HAL
open science

Agrégation spatiotemporelle pour la visualisation de traces d'exécution

Damien Dosimont

To cite this version:

Damien Dosimont. Agrégation spatiotemporelle pour la visualisation de traces d'exécution. Systèmes embarqués. Université Grenoble Alpes, 2015. Français. NNT : 2015GREAM075 . tel-01176440v2

HAL Id: tel-01176440

https://inria.hal.science/tel-01176440v2

Submitted on 9 Jan 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Informatique**

Arrêté ministériel : 7 août 2006

Présentée par

Damien DOSIMONT

Thèse dirigée par **Guillaume HUARD**
et codirigée par **Jean-Marc VINCENT**

préparée au sein d'**Inria**
et de l'**Ecole Doctorale Mathématiques, Sciences et Technologies de l'Information, Informatique**

Agrégation spatiotemporelle pour la visualisation de traces d'exécution

Thèse soutenue publiquement le **10 juin 2015**,
devant le jury composé de :

Mme Brigitte PLATEAU
Professeur, Grenoble INP, Présidente
M. João COMBA
Professeur, Universidade Federal do Rio Grande do Sul, Rapporteur
M. Martin QUINSON
Maître de conférences, Université de Lorraine, Rapporteur
M. Noël DE PALMA
Professeur, Université Joseph Fourier, Examinateur
M. Jesús LABARTA
Professeur, Barcelona Supercomputing Center, Examinateur
M. Raymond NAMYST
Professeur, Université de Bordeaux, Examinateur
M. Miguel SANTANA
Directeur du centre IDTEC, STMicroelectronics, Examinateur
M. Guillaume HUARD
Maître de conférences, Université Joseph Fourier, Directeur de thèse

RÉSUMÉ

Les techniques de visualisation de traces sont fréquemment employées par les développeurs pour comprendre, déboguer, et optimiser leurs applications. La plupart des outils d'analyse font appel à des représentations *spatiotemporelles*, qui impliquent un axe du temps et une représentation des ressources, et lient la dynamique de l'application avec sa structure ou sa topologie. Toutefois, ces dernières ne répondent pas au problème de passage à l'échelle de manière satisfaisante. Face à un volume de trace de l'ordre du Gigaoctet et une quantité d'évènements supérieure au million, elles s'avèrent incapables de représenter une vue d'ensemble de la trace, à cause des limitations imposées par la taille de l'écran, des performances nécessaires pour une bonne interaction, mais aussi des limites cognitives et perceptives de l'analyste qui ne peut pas faire face à une représentation trop complexe. Cette vue d'ensemble est nécessaire puisqu'elle constitue un point d'entrée à l'analyse ; elle constitue la première étape du *mantra* de Shneiderman — *Overview first, zoom and filter, then details-on-demand* —, un principe aidant à concevoir une méthode d'analyse visuelle.

Face à ce constat, nous élaborons dans cette thèse deux méthodes d'analyse, l'une temporelle, l'autre spatiotemporelle, fondées sur la visualisation. Elles intègrent chacune des étapes du *mantra* de Shneiderman — dont la vue d'ensemble —, tout en assurant le passage à l'échelle. Ces méthodes sont fondées sur une méthode d'agrégation qui s'attache à réduire la *complexité* de la représentation tout en préservant le maximum d'*information*. Pour ce faire, nous associons à ces deux concepts des mesures issues de la théorie de l'information. Les parties du système sont agrégées de manière à satisfaire un compromis entre ces deux mesures, dont le poids de chacune est ajusté par l'analyste afin de choisir un niveau de détail. L'effet de la résolution de ce compromis est la discrimination de l'hétérogénéité du comportement des entités composant le système à analyser. Cela nous permet de détecter des anomalies dans des traces d'applications multimédia embarquées, ou d'applications de calcul parallèle s'exécutant sur une grille.

Nous avons implémenté ces techniques au sein d'un logiciel, Ocelotl, dont les choix de conception assurent le passage à l'échelle pour des traces de plusieurs milliards d'évènements. Nous proposons également une interaction efficace, notamment en synchronisant notre méthode de visualisation avec des représentations plus détaillées, afin de permettre une analyse descendante jusqu'à la source des anomalies.

ABSTRACT

Trace visualization techniques are commonly used by developers to understand, debug, and optimize their applications. Most of the analysis tools contain *spatiotemporal* representations, which is composed of a time line and the resources involved in the application execution. These techniques enable to link the dynamic of the application to its structure or its topology. However, they suffer from scalability issues and are incapable of providing overviews for the analysis of huge traces that have at least several Gigabytes and contain over a million of events. This is caused by screen size constraints, performance that is required for an efficient interaction, and analyst perceptive and cognitive limitations. Indeed, overviews are necessary to provide an entry point to the analysis, as recommended by Shneiderman's *mantra* — *Overview first, zoom and filter, then details-on-demand* —, a guideline that helps to design a visual analysis method.

To face this situation, we elaborate in this thesis several scalable analysis methods based on visualization. They represent the application behavior both over the temporal and spatiotemporal dimensions, and integrate all the steps of Shneiderman's *mantra*, in particular by providing the analyst with a synthetic view of the trace. These methods are based on an aggregation method that reduces the representation *complexity* while keeping the maximum amount of *information*. Both measures are expressed using information theory measures. We determine which parts of the system to aggregate by satisfying a trade-off between these measures; their respective weights are adjusted by the user in order to choose a level of details. Solving this trade off enables to show the behavioral heterogeneity of the entities that compose the analyzed system. This helps to find anomalies in embedded multimedia applications and in parallel applications running on a computing grid.

We have implemented these techniques into Ocelotl, an analysis tool developed during this thesis. We designed it to be capable to analyze traces containing up to several billions of events. Ocelotl also proposes effective interactions to fit with a top-down analysis strategy, like synchronizing our aggregated view with more detailed representations, in order to find the sources of the anomalies.

REMERCIEMENTS

J'aimerais commencer par remercier Cathy, qui m'a soutenu et m'a supporté pendant ces trois années. Je te remercie des sacrifices que tu as fait pour moi, de ton aide infaillible, et surtout de l'amour que tu m'as porté.

Je souhaite remercier Guillaume et Jean-Marc, qui, avant toute chose, m'ont incité à faire ce que j'aimais et à prendre du plaisir à travers cette première expérience de recherche. J'ai apprécié le fait que vous m'avez très vite fait confiance et poussé à l'autonomie, sans pour autant m'abandonner à mon triste sort. Trouver cet équilibre est particulièrement difficile, mais c'est quelque chose que vous avez cependant bien maîtrisé. J'ai également été sensible à votre enthousiasme, à votre tact et votre qualité d'écoute, à la justesse de vos conseils et de vos remarques, à votre honnêteté et votre éthique qui m'ont paru exemplaires, et à votre perfectionnisme qui m'a poussé à m'améliorer tout au long de ces trois années.

Je tiens à remercier João Comba et Martin Quinson, qui ont relu mon travail avec attention. J'ai apprécié la qualité de leurs rapports et la pertinence de leurs suggestions. Je remercie également Brigitte Plateau, João Comba, Martin Quinson, Noël de Palma, Jesús Labarta, Raymond Namyst, Miguel Santana, Guillaume Huard et Jean-Marc Vincent qui m'ont fait l'honneur d'évaluer mes travaux.

Je remercie chaleureusement Lucas, qui, tout au long de ma thèse, m'a accordé une attention particulière, même après son retour au Brésil, et que je considère comme un encadrant officieux. J'ai apprécié tes qualités scientifiques, pédagogiques, mais aussi humaines (et sportives ;-)). Je te suis également reconnaissant de ton accueil à Porto Alegre. Je remercie également Robin, dont les travaux m'ont profondément inspiré, et avec qui j'ai pris un réel plaisir à collaborer, en particulier car nous avons su mettre à profit notre complémentarité. Je te suis reconnaissant de l'intérêt que tu as manifesté pour mes travaux tout au long de ma thèse. Je remercie aussi Generoso, dont le rôle a été primordial dans l'accomplissement de ce travail. Ta persévérance, ta rigueur et ton professionnalisme m'auront été salutaires. J'espère que nous aurons l'occasion de compléter ce tour du monde que nous avons déjà bien entamé. Je tiens à remercier Youenn, dont l'aide m'a été on ne peut plus précieuse. J'ai apprécié ton investissement, ton soutien et ta générosité. Je souhaite également remercier Annie, dont le professionnalisme et la gentillesse m'auront épargné bien des tracas. Un grand merci à Arnaud, qui a manifesté un grand in-

térêt pour mes travaux, et m'a offert de nombreuses opportunités de les mettre en avant. Merci également à Augustin et Luka. Je tiens à remercier, mais aussi encourager David, avec qui je souhaite réellement approfondir notre collaboration. Merci à Matthieu pour son aide quant à la prise en main de Grid'5000. Merci à Maxime, Alexis, Xavier, Alex et Élodie, grâce à qui j'ai pu sortir la tête du guidon. Merci à Joseph, Marie, Pierre et Bruno qui, à l'inverse, m'ont mis le nez dedans, mais au sens propre. J'aimerais également remercier tous les membres de l'équipe MOAIS et MESCAL. J'ai réellement apprécié de travailler au sein de ces deux équipes, qui constituent un environnement particulièrement stimulant.

Un grand merci à Miguel, Jérôme, Carlos et Roberto pour leur accueil à STMicroelectronics. Ces deux mois ont été particulièrement formateurs. J'aimerais également remercier tous les SoC-Traciens, et en particulier Vania, Damien, Alexandre, Serge, Christiane, Léon et Rémy, pour les discussions intéressantes que nous avons pu avoir et l'intérêt qu'ils ont porté à mes travaux. Merci à tous les doctorants brésiliens de l'institut d'informatique de l'UFRGS, à mes colocataires, Carla et Gisele, et mes professeurs de l'école de portugais, qui se sont évertués à rendre mon séjour à Porto Alegre agréable. Merci également à Swann pour son accueil à Kobé.

Je remercie ma famille et mes amis sans le soutien inconditionnel desquels je n'aurais jamais réussi cette thèse. Je pense en particulier à mon père, ma mère, à Emilie et à Nathan. Je tiens à remercier chaleureusement Elisa, dont la présence surprise à Grenoble pour ma soutenance m'a fait particulièrement plaisir, mon cher Valentin, qui nous a illuminé une fois de plus de sa verve flamboyante, Yves, qui a fait le déplacement malgré sa soutenance de doctorat une semaine après, et que je tiens à féliciter chaleureusement. Mention spéciale à Nicolas, alias Travis, le *Deus ex machina* de la logistique du pot de thèse, dont l'enthousiasme et l'humour raffiné auront marqué ce moment. Je remercie aussi Sofiane et Antoine qui n'ont pas pu venir mais m'ont fait preuve à maintes reprises de leur amitié en venant me soutenir sur les sentiers de VTT grenoblois ;-).

Je remercie sincèrement mon équipe de choc de relecteurs, Emilie, Maman, Youenn, Valentin, Robin, Lucas, Generoso, Luka, Shuai, Antoine, Gabriel et Cathy, qui ont fait preuve d'une persévérance sans faille à traquer les moindres fautes. Enfin, merci à tous ceux que j'ai malheureusement oubliés, mais dont l'aide a permis, directement ou indirectement, la réalisation de ce travail.

TABLE DES MATIÈRES

Résumé **iii**

Abstract **v**

Remerciements **vii**

Table des matières **ix**

Thèse en français **1**

1 Introduction **3**
1.1 Approches analytique et systémique 4
1.2 Problématique 4
1.2.1 Limitations du support d'affichage 5
1.2.2 Performances 6
1.2.3 Limites perceptives et cognitives de l'utilisateur 6
1.3 Objectif de la thèse et contributions 7
1.4 Contexte 9
1.4.1 SoC-Trace 9
1.4.2 SONGS 10
1.4.3 Projets MOAIS et MESCAL 10
1.5 Organisation du manuscrit 11

2 Visualisation de traces **15**
2.1 Description des traces d'exécution 16
2.1.1 Espace des ressources 16
2.1.2 Dimension temporelle 16
2.1.3 Évènements 17
2.2 Visualisation de traces 18
2.2.1 Visualisation globale fondée sur les opérateurs statistiques 18
2.2.2 Visualisation détaillée de la trace 19

2.3 Passage à l'échelle de l'analyse 23

3 Agrégation et passage à l'échelle 25
3.1 Définition et propriétés de l'agrégation 25
3.1.1 Agrégation mathématique 26
3.1.2 Processus de réduction de complexité dans la visualisation 27
3.2 Critères d'Elmqvist & Fekete 29
3.3 Techniques de visualisation employant de l'agrégation 30
3.3.1 Techniques fondées sur l'agrégation visuelle 30
3.3.2 Techniques hybrides 31
3.3.3 Techniques fondées sur l'agrégation de données 32
3.3.4 Discussion 34

4 Analyse macroscopique des grands systèmes 39
4.1 Formalisation du processus d'agrégation 40
4.1.1 Représentation microscopique d'un système 40
4.1.2 Partitionnement et agrégation 41
4.2 Problème des partitions optimales et mesures de qualité 43
4.2.1 Mesures de qualité 44
4.3 Partitions admissibles 46
4.4 Construction des algorithmes d'agrégation 49

5 Agrégation temporelle 51
5.1 Structure des modèles microscopiques temporels 52
5.1.1 Dimension temporelle discrète 52
5.1.2 Définition du modèle microscopique 53
5.2 Métriques des modèles microscopiques temporels 53
5.2.1 Analyse des évènements 53
5.2.2 Analyse des états 54
5.2.3 Analyse des variables 55
5.3 Effets de la discrétisation 55
5.4 Modèle d'agrégation temporelle 58
5.5 Extension aux modèles multidimensionnels 59
5.5.1 Structures de données 59
5.5.2 Sortie de l'algorithme 61
5.5.3 Description de l'algorithme 61
5.5.4 Complexité théorique de l'algorithme 63
5.5.5 Interprétation de la sortie de l'algorithme 64
5.6 Visualisation d'une partition du système 64
5.6.1 Histogramme 64
5.6.2 Partition 65
5.6.3 Histogramme empilé 66
5.6.4 Mode 68
5.7 Analyse temporelle 68
5.7.1 Applications multimédia 68
5.7.2 Applications parallèles 83
5.8 Bilan 86

5.8.1 Respect des critères d'Elmqvist-Fekete 86
5.8.2 Conclusions sur l'analyse 88

6 Agrégation spatiotemporelle 89
6.1 Proposition de modèles microscopiques 90
6.2 Modèle d'agrégation spatiotemporelle 91
6.2.1 Caractérisation du processus d'agrégation 91
6.2.2 Comparaison avec d'autres types d'agrégations 94
6.3 Algorithme 97
6.3.1 Structures de données 97
6.3.2 Sortie de l'algorithme 98
6.3.3 Description de l'algorithme 98
6.3.4 Complexité théorique de l'algorithme 101
6.4 Visualisation d'une partition du système 101
6.4.1 Partition 102
6.4.2 Amplitude relative 102
6.4.3 Mode 102
6.4.4 Agrégation visuelle 102
6.5 Analyse spatiotemporelle 105
6.5.1 Applications parallèles 106
6.6 Bilan 112
6.6.1 Respect des critères d'Elmqvist-Fekete 112
6.6.2 Conclusions sur l'analyse 113

7 Ocelotl 115
7.1 Framesoc 116
7.2 Architecture 117
7.2.1 Interface graphique 117
7.2.2 Noyau fonctionnel 120
7.3 Intégration dans la méthodologie de Shneiderman 121
7.3.1 Overview 123
7.3.2 Zoom 123
7.3.3 Filter 123
7.3.4 Details on demand 124
7.4 Performances et interactivité 124
7.4.1 Lecture de la trace et construction du modèle microscopique 124
7.4.2 Processus d'agrégation 131
7.4.3 Rendu 134
7.4.4 Performances globales et interactivité 134
7.5 Bilan 138

8 Conclusion 141
8.1 Bilan 142
8.2 Perspectives 143
8.2.1 Méthode d'agrégation d'information 144
8.2.2 Méthode d'agrégation visuelle 145
8.2.3 Performances 145

8.2.4 Applications 146

Bibliographie **149**

Annexes **157**

Annexe A Cas d'études supplémentaires **159**

Annexe B Optimisation des requêtes **163**

Annexe C Publications **165**

Annexe D Logiciels **167**

Extended Abstract in English **169**

1 Introduction **171**

2 Visualization of Execution Traces **175**

3 Aggregation and Scalability **177**

4 Macroscopic Analysis of Complex Systems **179**

5 Temporal Aggregation **181**

6 Spatiotemporal Aggregation **185**

7 Ocelotl **189**

8 Conclusion **193**

Agrégation spatiotemporelle pour la visualisation de traces d'exécution

CHAPITRE 1

INTRODUCTION

Il y a trois ans, lorsque j'ai commencé mon doctorat, la segmentation du travail des chercheurs en un aussi grand nombre de tâches indépendantes et concurrentes s'est vite imposée à mon attention. Puis, lorsque mon service d'enseignement a débuté, j'ai compris qu'à mon tour, je n'échapperai pas à la gestion minutieuse de mon agenda : les horaires des différents travaux pratiques dans lesquels j'officiais avaient en effet une certaine variabilité, mais leur répartition était aussi singulièrement inhomogène au cours de l'année. Ajoutez à cela les réunions au laboratoire, les échéances des appels à communications, et les divers séminaires et conférences qui régulent la vie d'un chercheur, et vous obtenez un agenda d'une complexité certaine.

Imaginons qu'un tiers un petit peu curieux se mette à suivre cet agenda, et pourquoi pas d'autres sources d'information, comme l'historique de mes *commits* sur *github*, ou encore les différentes notes ou rapports que j'ai écrits, et ce, afin d'évaluer l'avancement de mes travaux, ma ponctualité, ou la durée de mes sessions de babyfoot à la pause de 16 heures. Quelques jours d'analyse me semblent envisageables, quoique requérant déjà une obstination inquiétante. Mais la quantité d'information s'accumulant vite, en particulier à cause de mes sessions de codage intensives certains soirs et le nombre impressionnant de *micro-commits* qui en résultent, il serait impossible, en analysant directement toutes ces informations brutes, de comprendre le déroulement global de mon doctorat sur trois ans. Celui-ci n'est pourtant pas si complexe : il a commencé par une imprégnation du contexte et une étude bibliographique, suivie d'une phase plus productive où j'ai conçu la majeure partie de mes contributions et rédigé quelques articles scientifiques, pour finir par l'écriture de la thèse.

Prenons maintenant l'ensemble des personnes avec qui je travaille régulièrement. Il existe des synergies, des dépendances dans certaines tâches de développement logiciel, ou d'écriture d'articles scientifiques, des synchronisations lors de réunions, des séminaires rythmant la vie de l'équipe. Nous partons en vacances en général aux mêmes périodes, et nous ressentons l'impact du rythme universitaire sur celui du laboratoire. Intuitivement, nous avons la sensation que l'équipe a un comportement général proche de celui de la plupart de ses membres. Pourtant, si nous analysons nos agendas à une granularité très fine, tout semble différer : nos heures d'arrivée, de départ, un rendez-vous ponctuel chez le médecin pour l'un, une mission au Brésil d'une semaine pour un autre. Appréhender

le comportement de l'ensemble de l'équipe semble ainsi une tâche particulièrement complexe : nous avons déjà constaté qu'analyser correctement celui d'un seul de ses membres s'avérait plutôt ardu.

1.1 Approches analytique et systémique

Cet exemple illustre les limitations de ce qu'on appelle l'*approche analytique*, qui consiste à analyser séparément les entités microscopiques constituant un système, plutôt que le système dans son intégralité. Avec une *approche systémique*, c'est-à-dire l'analysant dans sa globalité, nous observerions un comportement *macroscopique* des systèmes « équipe de recherche » ou « laboratoire ». Celui-ci suivrait, par exemple, le rythme de l'année universitaire ou des grandes conférences. Le fait que quelques chercheurs partent en délégation et ne donnent pas d'enseignements ou n'assistent pas auxdites conférences pour des raisons de budget n'étant pas significatif vis-à-vis du comportement global du laboratoire. Cependant, pour parvenir à percevoir cette vue d'ensemble d'un système, il est indispensable de réduire la complexité des informations collectées avant de les analyser. Ainsi, par exemple, je conseillerais à toute personne qui serait tentée de regarder un par un tous mes *commits* et mon agenda, de plutôt continuer à lire cette thèse, qui en est leur représentation synthétique.

Les systèmes et applications embarqués ou parallèles fonctionnent un peu comme un laboratoire. Les processus sont comparables aux chercheurs, les fonctions, à leurs tâches quotidiennes, les interruptions, à des imprévus. Mais sans aller plus loin dans cette comparaison, le point commun le plus significatif entre mon exemple et ces applications concerne le fait que l'on puisse également définir chez elles un comportement macroscopique, c'est-à-dire une description simple du comportement des ressources impliquées dans leur exécution, entre elles, et au cours du temps. Le traçage est une méthode consistant à collecter et sauvegarder dans un fichier des données sur les ressources et les évènements survenant au cours de l'exécution d'une application. C'est l'équivalent de notre agenda. Deux facteurs conditionnent le succès de la méthode employée pour leur analyse : la masse de données générée et contenue dans la trace, et la complexité de la méthode employée. Un système de faible taille peut être analysé avec une méthode complexe, comme c'est le cas avec des algorithmes de fouille de données, tandis qu'un système de grande taille peut être analysé avec une méthode de faible complexité, en employant une approche fondée sur une synthèse à l'aide d'opérateurs statistiques, par exemple. Mais lorsque la masse de données est importante, l'analyse selon une méthode complexe, comme c'est le cas avec l'approche analytique, devient difficile voire impossible. Dans cette thèse, nous nous plaçons dans la situation où la quantité de données générées lors du traçage est conséquente.

1.2 Visualisation de traces et problématique du passage à l'échelle

Les techniques de visualisation de traces sont employées par les développeurs d'applications, embarquées comme parallèles, afin de comprendre, déboguer et optimiser ces dernières. En particulier, ils utilisent traditionnellement des techniques axées sur la re-

présentation du comportement de l'application au cours du temps et de la structure des ressources impliquées dans son exécution, comme le diagramme espace-temps, inspiré du diagramme de Gantt. Cette thèse s'attaque aux problématiques de passage à l'échelle de cette catégorie de représentations.

Le *mantra* de Shneiderman [1] aide à concevoir une méthode d'analyse fondée sur la visualisation :

Overview first, zoom and filter, then details-on-demand

Il préconise en fait une approche descendante de l'analyse, qui consiste à commencer par une vue d'ensemble, pour se focaliser progressivement sur les détails grâce à des interactions comme le zoom et le filtrage. Le concept de vue d'ensemble est cohérent avec l'approche systémique consistant à analyser le système comme un tout.

Les représentations spatiotemporelles classiques tendent à représenter l'ensemble de l'information contenue dans la trace, à l'instar de l'approche analytique. Cela gêne le passage à l'échelle et les outils s'avèrent incapables de générer une vue d'ensemble de la trace, ce qui ne permet pas d'aborder l'analyse de manière optimale. Nous distinguons plus précisément trois facteurs expliquant l'échec de ce genre de techniques à fournir une vue d'ensemble satisfaisante : les limitations du support d'affichage [2], les performances [3], et les limites perceptives et cognitives de l'utilisateur [4, 5, 6].

1.2.1 Limitations du support d'affichage

Les écrans d'ordinateurs, supports les plus couramment utilisés par les analystes, possèdent une résolution de l'ordre du million de pixels. Avec des objets graphiques d'une surface d'un pixel, une visualisation peut représenter au mieux un million d'objets simultanément. Dans cette thèse, nous générons des traces issues d'exécutions d'applications contenant jusqu'à 218 millions d'évènements. Il n'est donc pas envisageable d'associer chaque évènement de la trace à un objet graphique. Deux « stratégies » existent. La première consiste à *ignorer le problème* et à laisser la bibliothèque responsable du rendu graphique gérer automatiquement le placement et la taille des objets graphiques. Cela aboutit à la superposition et donc à la disparition de certains objets, à la déformation des proportions, et à une incohérence entre les visualisations lorsque l'on redimensionne ou déplace la fenêtre d'affichage. Ces représentations deviennent illisibles et difficiles à analyser, voire trompeuses. Elles forment donc des *overviews* inadaptés. Un exemple de ce qu'on peut obtenir avec cette stratégie est donné par la Figure 1.1. La deuxième stratégie consiste à ne pas représenter la trace en entier afin que les objets graphiques puissent être affichés correctement. On s'écarte alors des recommandations de Shneiderman, puisque l'analyste n'a pas accès à une vue d'ensemble de la trace. Cela provoque des pertes de contexte fréquentes : les déplacements selon les axes verticaux, horizontaux, et les changements d'échelles finissent par gêner l'analyste qui situe avec difficulté la position de la zone affichée.

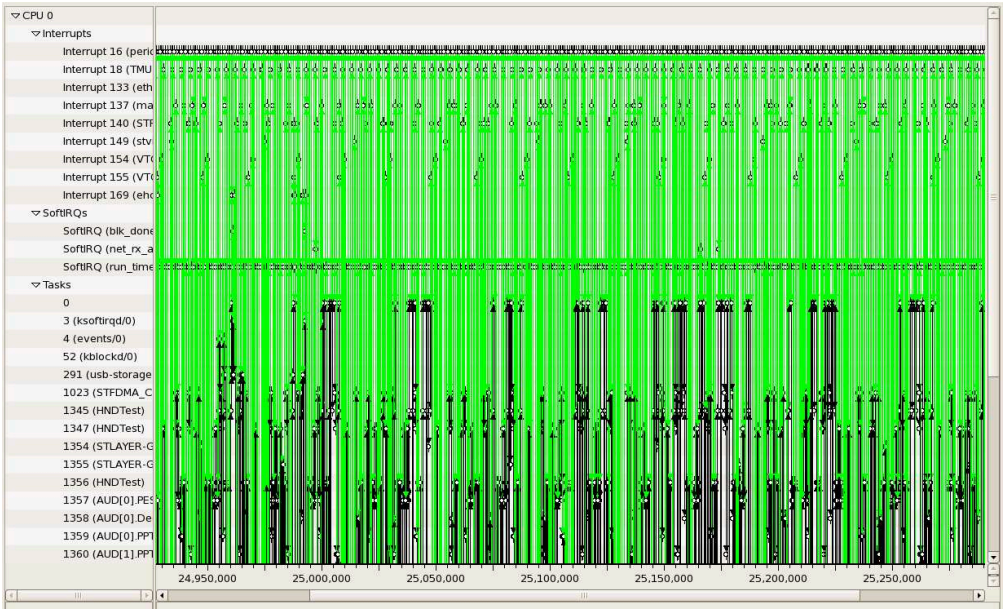


FIGURE 1.1 – Exemple d'une tentative d'afficher une grande quantité d'évènements avec le diagramme espace-temps T-Charts [7], qui ne possède pas de technique de réduction de la complexité sur l'axe du temps. La représentation est illisible.

1.2.2 Performances

La nécessité d'une analyse interactive pose des contraintes en terme de performances, qui n'existeraient pas si l'on générait une visualisation de manière statique. On distingue trois étapes dont la faisabilité et la durée ont un impact sur l'analyse : la lecture de la trace, l'interprétation de son contenu, et l'affichage. Pour chacune d'elles, il est possible de subir l'arrêt intempestif et systématique du programme, en général à cause d'une saturation de la mémoire : l'analyse de la trace est alors impossible. Cela est généralement dû à la mise en mémoire de tous les évènements de la trace lors de la lecture, stratégie inadaptée à de gros volumes de trace, à la complexité spatiale trop importante de l'algorithme interprétant les données, ou à la multitude d'objets graphiques affichés, eux aussi stockés en mémoire. Passé ce problème critique, l'utilisateur doit parfois faire face à un temps de calcul *non interactif*, c'est-à-dire que le résultat est retourné dans un délai déraisonnablement long et décourageant. Dans le cas de la lecture, cela dépend majoritairement du format de trace et de la technologie employée pour la stocker. Pour l'interprétation, ceci est généralement lié à la complexité temporelle des algorithmes mis en jeux. La fluidité et la réactivité de l'outil lors de l'interaction sont principalement liées à la bibliothèque d'affichage. Ce sont des facteurs importants, puisqu'une navigation aisée dans la trace facilitera l'analyse.

1.2.3 Limites perceptives et cognitives de l'utilisateur

Imaginons que l'on ait à disposition une surface d'affichage gigantesque, et que l'on fasse abstraction des limitations matérielles (puissance de calcul, mémoire) et logicielles

(complexité algorithmique). L'analyste, en tant qu'être humain, est lui aussi un facteur limitant puisqu'il n'est pas capable de traiter une quantité de données déraisonnablement importante. Ironiquement, la Figure 1.2 illustre particulièrement bien ce phénomène. Deux analystes sont photographiés dans le cadre d'une campagne de communication de la Maison de la Simulation, à Saclay. Les performances graphiques et calculatoires du mur d'image rendent possible l'affichage une trace extrêmement complexe avec le diagramme espace-temps de l'outil d'analyse ViTE[1]. Cependant, il est peu crédible que le procédé soit réellement efficace au niveau de l'analyse. La limitation des capacités visuelles humaines gêne l'observation des objets graphiques de faible taille à grande distance du mur, lorsque l'on souhaite observer la trace en entier. Mais surtout, la complexité de cette représentation, composée de millions d'objets graphiques dont la variété des couleurs est importante, chacune d'elle étant associée à une sémantique particulière, est bien trop importante pour en tirer une interprétation sur le comportement global de l'application : la limite est ici notre cerveau !

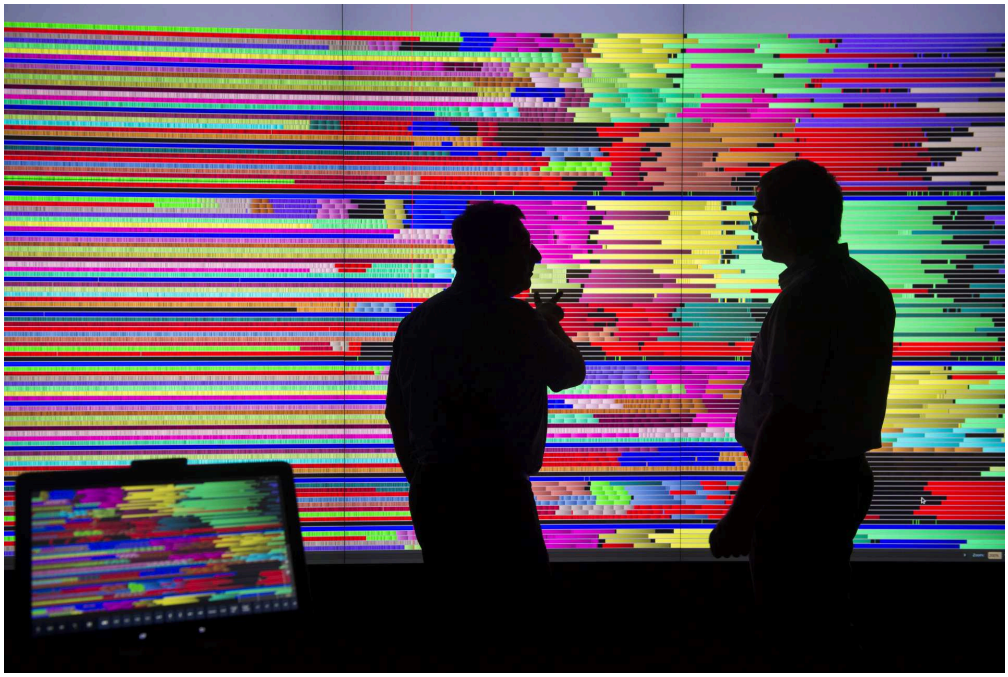


FIGURE 1.2 – Pour les besoins d'une campagne de communication, deux individus « analysent » le diagramme espace-temps d'une trace complexe, composé de millions d'objets graphiques, généré à l'aide de l'outil ViTE et affiché sur le mur d'image de la Maison de la Simulation.

1.3 Objectif de la thèse et contributions

L'objectif de cette thèse est l'élaboration d'une méthode d'analyse de traces fondée sur la visualisation, passant à l'échelle et conforme au *mantra* de Shneiderman, dont il est

[1] http://vite.gforge.inria.fr/

nécessaire de proposer une solution satisfaisant chacune des étapes. Les applications visées à l'origine sont les applications multimédia embarquées temps réel, mais nous élargissons ce champ aux applications parallèles et de calcul haute performance desquelles la structure et la complexité des plateformes et applications embarquées se rapprochent de plus en plus.

Pour assurer l'étape d'**overview** en prenant en compte les aspects relatifs aux limitations du support d'affichage, de l'interactivité, et de l'utilisateur lui-même, nous avons conçu une méthode d'analyse temporelle destinée initialement aux applications embarquées temps-réel pour lesquelles l'observation de l'évolution du comportement au cours du temps est primordiale, et une méthode d'analyse spatiotemporelle, principalement destinée aux systèmes parallèles structurés. Elles forment une extension significative d'une méthode d'agrégation existante, issue des travaux de thèse de Robin Lamarche-Perrin [8] sur l'analyse de systèmes multi-agents, qui permet de réduire la complexité d'un système en amont de sa représentation, tout en donnant à l'analyste la possibilité de choisir le niveau de détail et de maîtriser la perte d'information. L'un des principaux intérêts de cette méthode est que la réduction de complexité n'est pas employée dans le seul but d'assurer le passage à l'échelle de l'analyse : elle apporte une connaissance supplémentaire sur le comportement du système en discriminant l'homogénéité et l'hétérogénéité du comportement des entités qui le composent.

Cette méthode n'est toutefois pas applicable directement aux mesures contenues dans les traces brutes. Il est d'abord nécessaire d'abstraire la trace sous la forme d'une représentation intermédiaire dont les dimensions sont discrètes, le *modèle microscopique*, fondée sur une métrique représentant les phénomènes que l'analyste cherche à discriminer. Dans le cas de l'analyse temporelle, nous avons adapté et étendu l'algorithme d'agrégation d'un ensemble ordonné proposé par Lamarche-Perrin [8, 9]. Pour l'analyse spatiotemporelle, nous avons conçu un algorithme agrégeant simultanément les deux dimensions. Pour chacune de ces méthodes, nous associons des visualisations capables de représenter la trace abstraite. La réduction de complexité étant potentiellement inhomogène sur l'ensemble de la représentation, nous assurons le passage à l'échelle complet grâce à des techniques complémentaires fondées sur l'agrégation des éléments graphiques trop petits pour être affichés correctement. Ce processus nous permet de répondre positivement aux limitations du support visuel, et cognitives et perceptives de l'analyste.

Nous avons développé un logiciel pendant cette thèse, Ocelotl, dont les choix de conception visent à répondre aux contraintes de performances, mais aussi à une intégration complète des recommandations de Shneiderman : Ocelotl propose une vue d'ensemble de la trace, mais offre également une interaction poussée afin de **zoomer**, **filtrer**, et accéder à plus de **détails**. Cette dernière étape est réalisée notamment grâce à des représentations statistiques auxiliaires synchronisées, ou en commutant sur une sous-partie de la trace vers des représentations comme le diagramme espace-temps. La Figure 1.3 est une capture d'écran d'Ocelotl et montre un exemple de visualisation temporelle obtenue lors de l'analyse d'une trace issue de l'exécution d'une application parallèle.

Des cas d'études issus d'applications multimédia embarquées et de calcul parallèle valident la pertinence des méthodes d'analyse et de leur implémentation : nous y décelons des anomalies impossibles ou difficiles à détecter avec les techniques de visualisation spatiotemporelles classiques. Des tests exhaustifs valident les capacités de passage à l'échelle de l'infrastructure logicielle, capable de traiter des traces de plus de deux milliards d'évè-

nements, et d'une taille atteignant jusqu'à 68 gigaoctets dans un temps raisonnable.

L'ensemble de ces travaux a fait l'objet de plusieurs publications : trois en conférences internationales [10, 11, 12], une dans atelier international [13], deux dans des ateliers nationaux [14, 15], et deux rapports de recherche [16, 17].

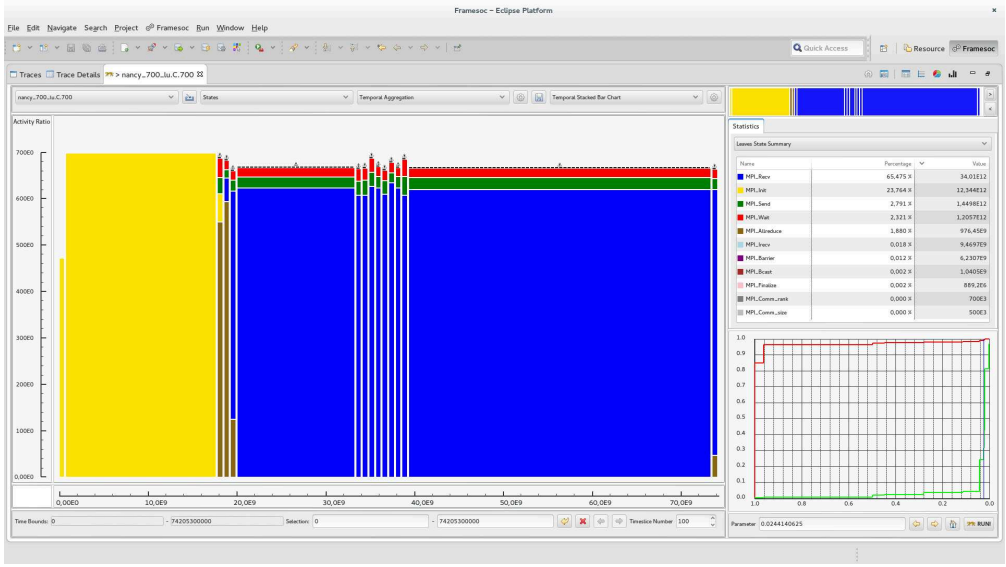


FIGURE 1.3 – Capture d'écran d'Ocelotl, l'outil d'analyse de trace développé pendant cette thèse.

1.4 Contexte

Les difficultés liées à l'analyse de traces issues de l'exécution d'applications embarquées, et en particulier multimédia, dans un contexte industriel, ont motivé le montage du projet SoC-Trace[2], qui a financé cette thèse. Cette problématique est partagée avec la communauté HPC (*High Performance Computing*), et constitue l'une des thématiques du projet SONGS[3] dans lequel j'ai eu l'occasion d'intervenir. Ces deux projets impliquent mes équipes-projets d'accueil MOAIS et MESCAL.

1.4.1 SoC-Trace

SoC-Trace est financé par le Fonds unique interministériel (FUI). Il implique des partenaires académiques tels qu'Inria Rhône-Alpes et l'Université Joseph Fourier, et des industriels tels que STMicroelectronics, acteur mondial dans le secteur de la production de semi-conducteurs, ProbaYes, une PME spécialisée dans l'analyse prédictive fondée sur une approche probabiliste, et Magillem Design Services, une PME de service et de conseil en informatique spécialisée dans le domaine de la gestion et l'exploitation de grands volumes de données.

[2]http://www.minalogic.com/TPL_CODE/TPL_PROJET/PAR_TPL_IDENTIFIANT/2717/15-annuaire-innovations-technologiques-nanotechnologie-systeme-embarque.htm

[3]http://infra-songs.gforge.inria.fr/

Le projet s'attaque à plusieurs problèmes liés à l'analyse de traces issues de l'exécution d'applications embarquées :

— la gestion de traces de gros volume (plusieurs gigaoctets) contenant des millions d'évènements. Cela pose en particulier des problèmes de stockage mais aussi de performances lorsqu'il s'agit de lire la trace pour l'analyser ;

— l'inadaptation des outils actuellement employés par les industriels à l'analyse de traces volumineuses générées par des systèmes embarqués de plus en plus complexes (multicœurs, hétérogènes, voire distribués).

Pour répondre à ces deux points, une infrastructure de gestion de traces, Framesoc [11, 12, 18], a été développée. Elle centralise un ensemble d'outils d'analyse développés par les partenaires du projet, chacun proposant des méthodes novatrices, s'appuyant sur des techniques telles que la fouille de données [19], l'analyse probabiliste, et bien entendu l'agrégation de données et la visualisation qui sont l'objet de cette thèse. L'intérêt est de permettre un flot d'analyse et une coopération entre ces différents outils avec l'objectif de détecter et d'identifier des anomalies. Le projet vise à l'intégration de ces différents composants au sein d'un produit, destiné à être utilisé par les développeurs d'applications multimédia afin d'améliorer leur productivité, la qualité des logiciels développés, et à diminuer le coût des systèmes embarqués en optimisant à la fois le logiciel qui s'y exécute et la plateforme matérielle.

Nos travaux se positionnent ici d'abord comme point d'entrée à l'analyse. Les méthodes que nous proposons fournissent une vue d'ensemble de la trace, afin que les autres méthodes conçues dans le cadre du projet, dont la complexité algorithmique est plus grande, puisse être appliquées sur des sous-parties de la trace intéressantes, et non sur son intégralité. Mais il est également possible d'employer nos méthodes en fin de chaine, pour représenter le comportement de traces préalablement abstraites et simplifiées.

1.4.2 SONGS

Le projet SONGS, qui poursuit l'objectif d'USS-Simgrid[4], se concentre sur la simulation de systèmes parallèles, aussi bien de grilles que de réseaux pair-à-pair, de systèmes nuagiques ou de calcul haute performance. La visualisation de traces est employée comme support pour valider le bon comportement de la simulation par rapport à un système réel. L'étude de l'influence de la structure, de l'architecture et de la topologie sur le comportement d'applications parallèles, comme les applications MPI, est ici un aspect important, auquel nous nous efforçons de répondre grâce à notre méthode d'analyse spatiotemporelle.

1.4.3 Projets MOAIS et MESCAL

Les équipes-projets MOAIS (pour Multi-programmation et Ordonnancement sur ressources distribuées pour les Applications Interactives de Simulation) et MESCAL (pour Middleware Efficiently SCALable), mixtes Inria-CNRS-LIG-UJF-UPMF-INPG, cultivent toutes les deux une forte sensibilité au domaine des systèmes et applications parallèles. MOAIS s'intéresse aux aspects liés à l'ordonnancement, à la conception d'algorithmes adaptatifs parallèles et distribués, ou encore aux applications de simulations interactives s'exécutant sur des grilles de calcul. MESCAL se focalise sur des thématiques comme les

[4] http://uss-simgrid.gforge.inria.fr/

modèles stochastiques de grands systèmes à évènements discrets, l'évaluation de performance et la simulation de systèmes déterministes et probabilistes, ou encore la conception d'intergiciels et les systèmes distribués. Les problématiques liées aux techniques d'analyses des traces issues de l'exécution d'applications sur des systèmes embarqués ou parallèles, réels ou simulés sont transverses à la plupart des sujets abordés par les deux équipes.

1.5 Organisation du manuscrit

Ce manuscrit de thèse est organisé en huit chapitres. La Figure 1.4 fait la correspondance entre ces derniers, les concepts et méthodologies empruntés et nos contributions.

Le **Chapitre 2** pose le contexte dans lequel se situe cette thèse : l'analyse de traces d'exécution d'applications à travers la visualisation et les problématiques de passage à l'échelle qui en découlent. Pour cela, nous introduisons d'abord la notion de traces d'exécution dont nous décrivons l'intérêt, le contenu, et des exemples de formats et d'infrastructures de traçage traditionnellement employées. Nous introduisons trois concepts que nous réutiliserons tout au long de la thèse : l'espace des ressources, la dimension temporelle, et enfin les évènements. Nous proposons ensuite une synthèse des principales techniques de visualisation de traces existantes illustrant notre problème : une première catégorie regroupe des visualisations synthétiques, fondées sur des opérateurs statistiques tandis que la seconde est constituée de visualisations détaillées focalisées sur la représentation de l'évolution du comportement de l'application au cours du temps ou de sa structure. Nous abordons les limitations de ces différentes méthodes : tandis que les premières passent à l'échelle mais au prix d'une perte d'information importante sur les dimensions temporelles ou structurelles, les secondes sont victimes des limitations liées à la taille de l'écran, aux performances et à l'utilisateur que nous avons évoquées dans la Section 1.2.

Le **Chapitre 3** porte sur l'agrégation et à son emploi dans les techniques de visualisation pour assurer le passage à l'échelle. Nous définissons l'agrégation et ses propriétés, d'abord d'un point de vue mathématique, pour introduire ensuite un formalisme plus général qui s'applique à deux types de techniques présentes dans le domaine de la visualisation d'information : l'*agrégation de données* et l'*agrégation visuelle*. Ce formalisme sera repris par la suite pour la description des méthodes d'agrégation. Afin de jauger la qualité d'une représentation agrégée, nous introduisons des critères présentés par Elmqvist & Fekete dans leurs travaux [4], grâce auxquels nous évaluons des techniques d'agrégation implémentées au sein d'outils d'analyse de traces. Cela nous permet de mettre en exergue les points forts et les faiblesses de chacune d'elles, afin d'établir un cadre dans lequel concevoir une technique d'agrégation en minimisant ses inconvénients. Ce bilan montre le potentiel d'une méthode de partition et d'agrégation d'un système multi-agents complexe, en vue de réduire la complexité tout en contrôlant la perte d'information, et qui semble bien adaptée à notre problème.

Le **Chapitre 4** décrit cette méthode, développée par Lamarche-Perrin durant sa thèse. Ce Chapitre n'apporte aucune contribution, mais est nécessaire pour expliquer le processus d'agrégation, et les concepts et les notations dont nous nous servons dans les Chapitres 5 et 6.

Le **Chapitre 5** se concentre sur l'adaptation à l'analyse de traces de l'algorithme

d'agrégation temporelle conçu par Lamarche-Perrin. Dans un premier temps, nous traitons les opérations nécessaires pour abstraire la trace brute afin de générer un modèle, la *représentation microscopique*, qui sera le niveau le plus fin pour l'analyse. Nous modifions l'algorithme d'agrégation d'un ensemble ordonné de Lamarche-Perrin. Puis, nous réalisons des techniques de visualisation capables de retranscrire la sortie de l'algorithme de manière graphique, en utilisant des techniques d'agrégation supplémentaires complétant celle de la *méthode de Lamarche-Perrin*. Enfin, nous proposons d'appliquer cette méthode à l'analyse de cas d'études. Les premiers s'inscrivent dans le domaine des applications multimédia embarquées, et correspondent à des scénarios ciblés par le projet SoC-Trace. Les seconds concernent des applications parallèles s'exécutant sur des grilles de calcul. Nous concluons le chapitre par un bilan évaluant la réponse de notre contribution aux critères d'Elmqvist & Fekete, comparé à d'autres techniques d'analyse existantes.

Le **Chapitre 6** présente notre méthode d'agrégation spatiotemporelle, en suivant globalement la même trame que le Chapitre précédent. Nous construisons une représentation microscopique, puis concevons l'algorithme et les techniques de visualisation de sa sortie, faisant elles aussi appel à des techniques d'agrégation complémentaires. Nous poursuivons par une application à l'analyse de cas d'études. Les applications parallèles précédemment évoquées dans le Chapitre 5 sont cette fois analysées sous le prisme de l'agrégation spatiotemporelle. Nous terminons également par un bilan relatif au respect des critères d'Elmqvist & Fekete, et nos conclusions sur l'intérêt mais aussi les limitations de notre technique.

Le **Chapitre 7** concerne l'implémentation des contributions des deux Chapitres précédents : l'outil d'analyse Ocelotl, développé pendant cette thèse. Nous abordons trois points. Nous commençons par les aspects relatifs à la conception du logiciel, son architecture et ses fonctionnalités, et son intégration au sein de Framesoc [11, 12, 18], l'infrastructure d'analyse et de gestion de traces développée dans le cadre de SoC-Trace. Nous poursuivons par les éléments, en particulier l'interaction, qui permettent à l'outil de s'intégrer dans le flot d'analyse préconisé par la méthodologie de Shneiderman. Enfin, nous justifions la capacité de l'outil à répondre à la problématique des performances et de l'interactivité lors du passage à l'échelle en détaillant l'ensemble des techniques qui optimisent ces deux aspects. Nous décrivons l'influence de certains paramètres (taille de la trace, taille des dimensions de la représentation microscopique) sur les temps de calcul de chacun des traitements. Nous concluons en situant les performances de notre outil par rapport à l'état de l'art.

Enfin, le **Chapitre 8** conclut cette thèse à travers une synthèse de nos contributions et un bilan général. Nous décrivons également nos perspectives et un certain nombre d'axes de recherches envisageables comme suite de cette thèse.

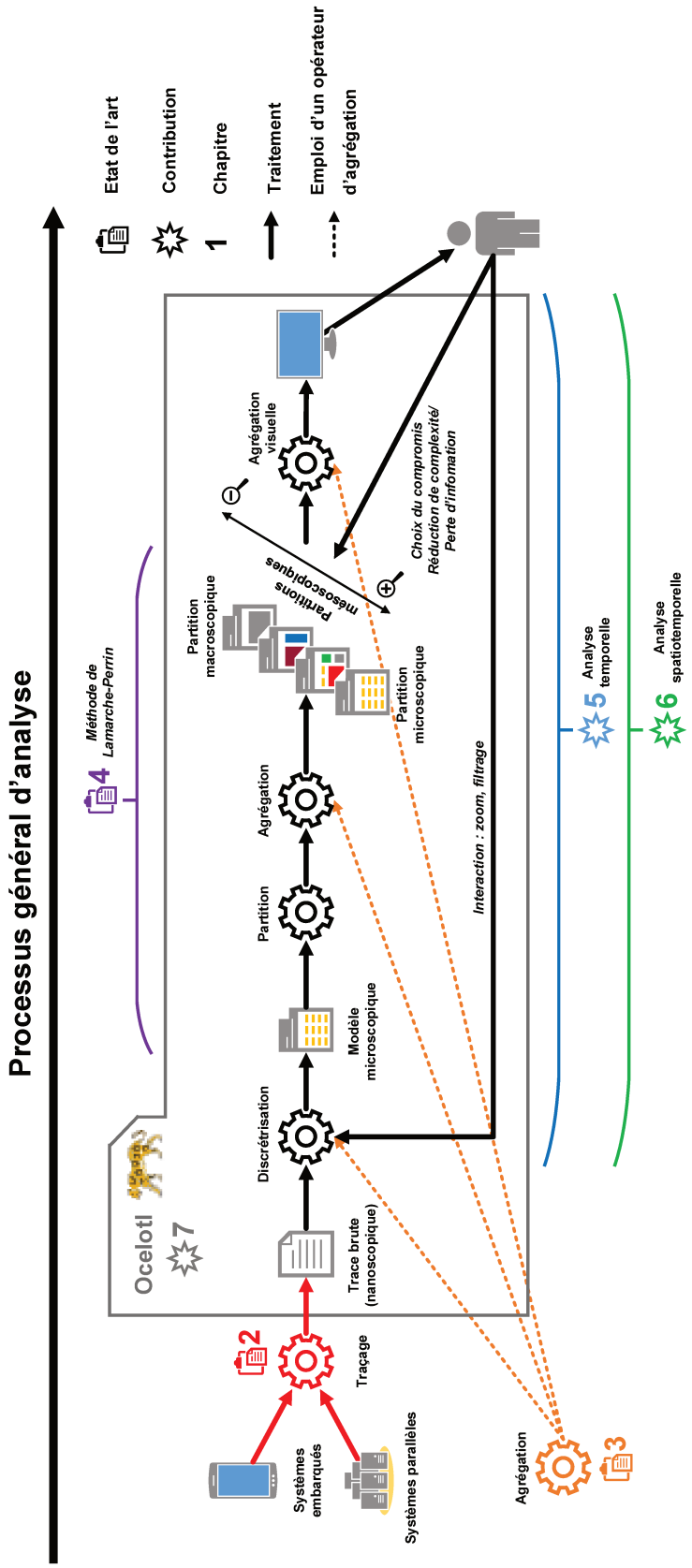


FIGURE 1.4 – Organigramme montrant le processus d'analyse, à travers les concepts, méthodologies et contributions présentés dans cette thèse, et les chapitres dans lesquels ils sont abordés.

CHAPITRE 2

Traces d'exécutions et techniques de visualisation

L'analyse du comportement d'applications logicielles à des fins de compréhension, de déverminage et d'optimisation est une tâche de plus en plus difficile, en raison de la complexité croissante des systèmes sur lesquels elles sont exécutées. Dans le cas des systèmes embarqués et des ordinateurs personnels, les principales difficultés sont liées à la pile logicielle complexe [20]. En effet, l'analyse concerne les aspects bas-niveau proches du matériel, les couches intergicielles abstrayant celui-ci [21] et les applications. Les systèmes parallèles, et en particulier les systèmes dédiés au calcul, se voient, quant à eux, distribués dans le monde, à travers une hiérarchie complexe (processeur, machine, grappe, site) et de plus en plus souvent hétérogène [22]. L'échelle de ces systèmes est contraignante pour les outils d'analyse. Pour tirer au mieux parti des multiples unités de calcul, les applications utilisent des bibliothèques logicielles, comme OpenMP ou MPI [23]. Cependant, elles se caractérisent par un fort indéterminisme inhérent au paradigme de la programmation parallèle (conditions de concurrences d'accès aux ressources, ordre des messages, interblocages), et pour lesquelles les méthodes d'analyse traditionnelles ne sont pas suffisantes [24].

Parmi les solutions les plus fréquemment employées pour l'analyse d'un système informatique figure le traçage, qui collecte des données pendant l'exécution de l'application. Ces traces sont ensuite analysées *post-mortem*. Tandis que certaines techniques, par exemple la fouille de données ou la reconnaissance de motifs, s'attèlent à chercher des anomalies dans la trace brute, nous nous intéressons, dans cette thèse, aux techniques d'analyse visuelle des traces, où le comportement de l'application, matérialisé par les données de la trace, est représenté à l'aide d'objets graphiques. Nous nous focalisons plus particulièrement sur les aspects temporels et architecturaux, avec l'objectif de relier la dynamique des comportements temporels à la structure et à la topologie de l'application.

2.1 Description des traces d'exécution

Tracer consiste à placer des observateurs puis à collecter et stocker des données sur l'exécution de l'application sous la forme d'un ou plusieurs fichiers, contenant des informations sur l'état de l'exécution et des mesures sur l'état de la plateforme, fournies par exemple par des compteurs matériels ou logiciels. Pour ce faire, le code de l'application est instrumenté à l'aide de bibliothèques logicielles, d'outils de traçage, ou simplement de fonctions d'affichage ou d'écriture dans des fichiers. Parmi les infrastructures les plus connues, on peut citer Score-P [25] et TAU [26] pour tracer des applications parallèles MPI ou OpenMP, ou encore LTTng [27, 28] pour analyser le noyau Linux et l'espace utilisateur.

La trace est générée au fur et à mesure de l'exécution dans un format spécifié (générique ou standardisé, textuel ou binaire selon les outils utilisés). Parmi les formats de traces les plus répandus, on peut citer les formats binaires et multi-fichiers OTF2 [29] et TAU dédiés à l'analyse de systèmes parallèles, qui permettent la lecture en parallèle pour de meilleures performances. On distingue aussi des formats textuels et génériques, comme celui de l'outil de visualisation Pajé [30, 31], Pablo [32], qui l'a inspiré, ou encore le format Paraver [33]. Le format binaire CTF [34] est utilisé par LTTng. Parmi les formats employés pour l'analyse des systèmes embarqués, les partenaires du projet SoC-Trace [35] utilisent KPTrace [36], un format de trace inspiré par Pajé, mais spécifique aux plateformes de STMicroelectronics.

Les éléments à tracer sont déterminés par l'analyste ou par l'outil servant à instrumenter le code. Chaque format de trace possède une syntaxe qui lui est propre, afin de représenter ces éléments. On distingue cependant trois concepts communs à la majorité des formats de trace : l'espace des ressources, le temps, et les évènements.

2.1.1 Espace des ressources

La ressource est un composant logiciel ou matériel de l'application ou du système. L'ensemble des ressources forme l'espace des ressources, que l'on nomme aussi *dimension spatiale*.

Définition 2.1 $S = \{s_1, \ldots, s_n\}$ est l'ensemble des éléments s définissant la dimension spatiale de la trace. Un élément s est aussi appelé *ressource*.

Les ressources peuvent être structurées, sous la forme d'un graphe ou d'une hiérarchie, par exemple, ou ordonnées (selon l'ordre alphabétique, un numéro, etc.).

2.1.2 Dimension temporelle

Le temps est la dimension permettant de dater dans la trace l'ensemble des phénomènes survenant pendant l'exécution du programme. Il est en général donné par l'horloge du processeur ou du système.

Définition 2.2 T est la dimension temporelle d'une trace. On peut le considérer comme un intervalle $[\tau_{debut}, \tau_{fin}]$ d'un corps commutatif K, dont la nature dépend du type de système qui est analysé et de la granularité temporelle de la trace (par exemple $\mathbb{N}$, $\mathbb{R}$, ou encore $\mathbb{R}^+$).

2.1.3 Évènements

Un évènement représente un point de l'espace-temps, c'est-à-dire un phénomène localisé dans le temps, et associé ou produit par un élément de l'espace des ressources.

Définition 2.3 $E = \{e_1, \ldots, e_n\}$ est l'ensemble des évènements e de la trace. On associe à chaque évènement e un couple de coordonnées spatiales et temporelles (s, τ).

Chaque évènement est typé, et on lui associe une ou plusieurs valeurs.

Il est à noter qu'il peut être nécessaire de resynchroniser à posteriori les évènements, c'est-à-dire modifier leurs coordonnées temporelles, en particulier dans le cas du traçage d'applications parallèles où différentes ressources génèrent simultanément des sous-parties de la trace [37]. En effet, si l'horloge de ces dernières n'est pas parfaitement synchrone, cela peut aboutir à des incohérences dans l'ordre des évènements.

Les évènements permettent de représenter plusieurs concepts d'un niveau d'abstraction supérieur, que l'on nomme catégories [30]. Celles-ci ne sont pas forcément définies explicitement par les formats de trace, mais la manière dont elles sont interprétées par les outils d'analyse, et en particulier, de visualisation, permet d'en établir une classification. On distingue :

- les *évènements ponctuels*. Ce sont les plus génériques. Ils correspondent à la définition de base de l'évènement, et représentent en pratique des phénomènes ponctuels comme des synchronisations, des drapeaux ou des changements de contexte ;
- les *états*. Il s'agit de phénomènes bornés dans le temps, possédant un début, une fin, et par conséquent une durée. On les utilise pour représenter des appels de fonction, l'état du système, d'un processus ou d'une ressource. Il est à noter que la plupart des formats de trace les modélisent sous la forme d'un couple d'évènements ponctuels correspondant au début et à la fin de l'état ;
- les *variables* sont employées pour représenter la valeur de compteurs matériels et logiciels, de variables internes du programme, ou encore l'utilisation de la mémoire ou du processeur ;
- les *liens*. Ils correspondent à un évènement modélisant l'interaction entre deux ressources, comme des communications ou un changement de contexte.

On peut enfin distinguer les types d'évènements, qui sont une organisation des évènements de la trace en sous-ensembles partageant des propriétés sémantiques (phénomènes qu'ils représentent) et syntaxiques (les champs et métriques qui leurs sont associés).

Définition 2.4 $\mathcal{J}(E) = \{J_1, \ldots, J_o\}$ définit l'ensemble des types d'évènements de la trace, ce qui correspond à un ensemble de sous-ensembles de E. $\mathcal{J}(E)$ contient l'ensemble E au complet ($E \in \mathcal{J}(E)$), et est tel que deux parties de $\mathcal{J}(E)$ sont toujours disjointes ($\forall (J_i, J_j) \in \mathcal{J}(E)^2,\ J_i \cap J_j = \emptyset$). Chaque type d'évènements possède un ensemble de métriques communes à tous les évènements de ce type.

Les types d'évènements sont par ailleurs des sous-ensembles des catégories.

2.2 Visualisation de traces

Nous présentons ici différentes catégories de techniques de visualisations de traces : la représentation de données statistiques présentant une synthèse globale de la trace et des représentations détaillées ayant pour objectif de comprendre l'évolution du comportement de l'application au cours du temps, ou l'influence des particularités de sa structure.

2.2.1 Visualisation globale fondée sur les opérateurs statistiques

La visualisation de données statistiques en vue d'une synthèse globale est une approche pertinente pour débuter une analyse, puisqu'elle permet de saisir une tendance générale. Plusieurs techniques sont fréquemment employées. Les graphiques à deux dimensions forment une représentation simple, constituée par deux axes et une courbe, traduisant la variation de la valeur d'une variable en fonction de la valeur d'un de ses paramètres. L'outil PerfExplorer [38] offre la possibilité de comparer l'efficacité relative ou le *speed-up* en fonction du nombre de processeurs grâce à ce type de représentations. Les histogrammes et diagrammes circulaires sont des représentations statistiques classiques traditionnellement utilisées pour comparer la répartition d'un certain nombre de valeurs. ParaGraph [39] permet de représenter, par exemple, le nombre de messages reçus par les processus, le taux de mémoire utilisé par les différentes machines. Paradyn [40] propose des histogrammes horizontaux contenant plusieurs types de valeurs, chacun possédant des échelles différentes. Les diagrammes circulaires sont présents dans Pajé [30] et permettent de quantifier le temps passé dans les différents états par chaque processus. La possibilité est laissée à l'utilisateur de définir la tranche de temps sur laquelle faire l'analyse. Cette méthode est idéale pour comparer deux processus et isoler des problèmes de performance reliés à un déséquilibre de répartition des tâches, par exemple. Le nuage de points est une représentation graphique donnant des indications sur le degré de corrélation entre deux ou plusieurs variables liées. Elle permet ainsi d'observer des relations (directes, inverses), des dépendances (fortes ou faibles), des tendances (linéaires, non linéaires) entre les variables, d'avoir un aperçu de l'homogénéité des répartitions ou d'isoler des données aberrantes. Une visualisation de ce type, illustrée par la Figure 2.1, est disponible au sein de l'outil PerfExplorer [38]. Les auteurs se servent de la corrélation pour ignorer des données qui fourniraient des informations redondantes (comme celles fournies par certains compteurs matériels), en vue de simplifier l'analyse. Les « area charts » consistent en une représentation en deux dimensions, fondée sur un graphique à courbes, où sont superposées différentes quantités à observer, mises en exergue par des zones de couleurs différentes. L'intérêt est de pouvoir comparer la taille des aires respectives et leur évolution en fonction de l'abscisse. Dans PerfExplorer [38], cette représentation peut être utilisée afin de comparer l'évolution du temps relatif passé dans différentes fonctions

ou états en fonction du nombre de processeurs, comme montré par la Figure 2.2. Des représentations en trois dimensions sans axe temporel permettent d'élargir l'analyse en affichant une donnée en fonction de trois variables. ParaProf [41] possède ainsi ce type de visualisations pour des nuages de points, des histogrammes ou à d'autres représentations à base de maillages triangulaires.

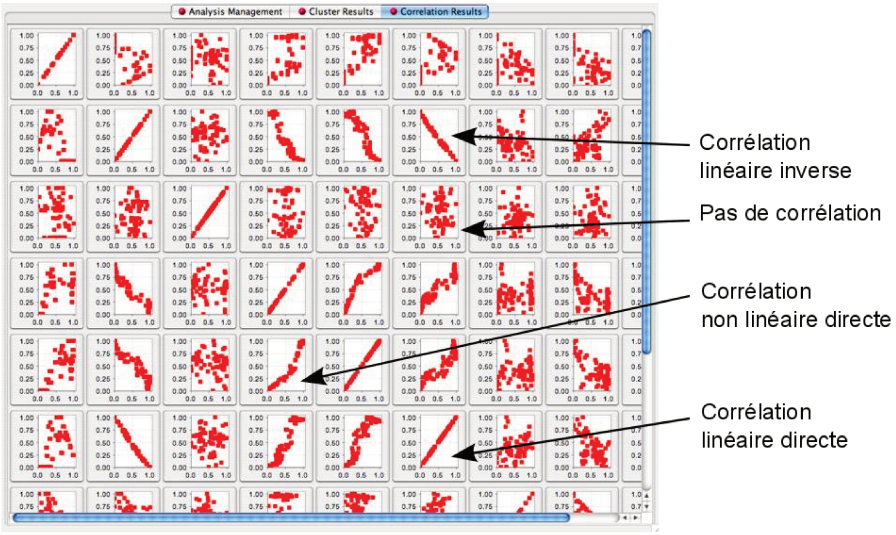


FIGURE 2.1 – Exemples de nuages de points issus de PerfExplorer mettant en évidence plusieurs types de corrélations entre différents évènements.
Source : http://www.cs.uoregon.edu/research/tau/vihps14/PerfExplorer.pdf

Les visualisations synthétiques fondées sur les statistiques sont un point d'entrée efficace à l'analyse de performance mais ne suffisent pas à elles seules à permettre une rétro-action sur le code de l'application. Les informations fournies permettent de déceler un comportement général problématique (un processus gourmand en ressources, par exemple) mais lorsque les phénomènes qui correspondent à un comportement indésirable sont limités dans le temps (comme des échéances non respectées, des interblocages ou des interruptions trop longues), ce type de visualisations est incapable de les mettre en évidence. De même, les composants de l'application, sa structure ou sa topologie ne sont pas représentés en détail, ce qui rend une analyse fine de l'influence de ces paramètres sur le comportement de l'application difficile. Les représentations statistiques décrites ici incitent donc l'utilisateur à affiner son examen de la trace à travers des méthodes de visualisation permettant une analyse temporelle et structurelle détaillée de celle-ci.

2.2.2 Visualisation détaillée de la trace

Nous nous intéressons ici aux techniques de visualisation fournissant des informations détaillées sur le comportement de l'application mais aussi sur la structure de la trace. Les plus plébiscitées sont issues de domaines divers, comme la gestion de projet (diagramme de Gantt), le stockage de données (visualisation treemap), l'algèbre (représentations matricielles) ou l'analyse mathématique (graphes d'appels de fonctions). Nous reprenons les catégories établies par Schnorr [42] durant sa thèse et les complétons. Il est

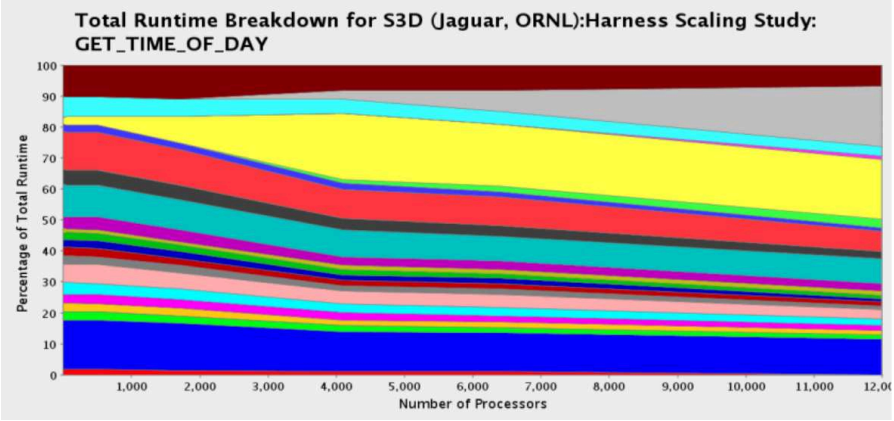


FIGURE 2.2 – Exemple de représentation statistique issue de PerfExplorer : un *area chart* représentant le pourcentage de temps passé dans différentes fonctions en fonction du nombre de processeurs.
Source : http://www.cs.uoregon.edu/research/tau/vihps14/PerfExplorer.pdf

à noter que des représentations statistiques focalisées sur un aspect temporel ou structurel précis de la trace existent. De par la perte de leur caractère synthétique, nous les classons parmi les visualisations détaillées.

Visualisation du déroulement de l'exécution

Certaines techniques de visualisation permettent d'étudier l'évolution du comportement d'une application au cours du temps à travers l'analyse de la séquence des événements, états, de la valeur de ses variables et des relations de causalité.

Diagramme espace-temps Le diagramme de Gantt [43], que l'on nommera aussi diagramme espace-temps, est apparu en 1896 et a d'abord été employé dans la gestion de projets. Dans le cas des outils de visualisation de traces, il est composé d'un axe temporel en abscisse tandis que l'axe des ordonnées correspond à l'ensemble des ressources dont les états vont être représentés au cours du temps. Ces derniers sont affichés sous la forme de rectangles disposés horizontalement, parfois colorés, dont les extrémités correspondent aux dates de début et de fin. LTTng Eclipse viewer [44] possède deux diagrammes de Gantt séparés représentant d'un côté les éléments logiciels (processus et fonctions) et de l'autre les entités matérielles (processeurs, IRQ). À l'inverse, Pajé [30] (Figure 2.3), utilisant un format de trace générique, laisse la possibilité d'associer la hiérarchie matérielle et logicielle.

Graphiques à courbes, histogrammes, « area charts », avec axe temporel Représentations similaires à celles présentées dans la Section 2.2.1, elles se distinguent ici par la présence d'un axe temporel en abscisse servant à montrer l'évolution de la valeur d'une ou plusieurs variables au cours du temps. Ces variables peuvent être des informations relatives au matériel : la « Streamline » de l'outil ARM DS5 [45] possède des représentations de ce type, sous la forme d'histogrammes (utilisation du processeur par le

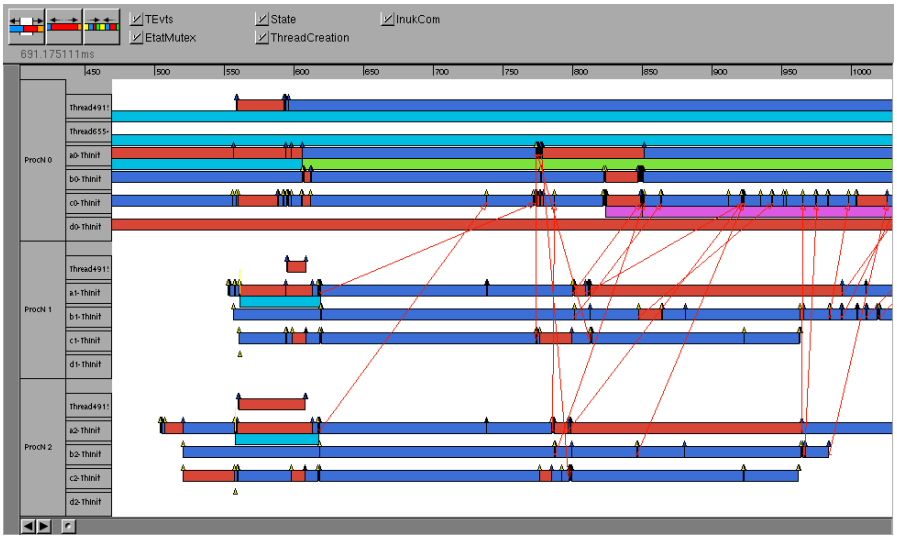


FIGURE 2.3 – Exemple de diagramme de Gantt issu de l'outil Pajé et montrant l'exécution de trois processus (états et communications).

système et l'utilisateur), ou d'*area charts* (afin de comparer la proportion d'interruptions logicielles et matérielles, les *cache hits* et *misses*, les écritures et lectures sur disque, la mémoire utilisée et disponible). Dans Pajé [30], il est possible de représenter les valeurs de variables du programme sous la forme d'une courbe intégrée au diagramme de Gantt principal. Ces techniques peuvent être transposées en trois dimensions, comme la *3D Terrain Visualization* de Paradyn [40], où les données représentées sont dépendantes de deux variables et du temps.

« 3D View » Dans les visualisations comportementales, un axe est utilisé pour le temps. Cela oblige à projeter les ressources sur une dimension, au détriment de la compréhension de leur structure. Une solution proposée par l'auteur de Triva [42] est la *3D view* qui combine la structure, à travers une représentation de la topologie, et un axe temporel.

Visualisation de la structure de l'application

Les techniques de visualisation structurelles se caractérisent par la représentation des composants constituant l'application ainsi que leurs interactions. La dimension temporelle, si elle est présente, n'est pas centrale.

Graphes d'appels de fonctions Les graphes d'appels de fonction sont des graphes orientés, dans lesquels sont représentés les appels (symbolisés par des liens) entre les différents composants (fonctions ou méthodes représentés par des nœuds). Cette technique de visualisation est efficace pour l'analyse d'applications parallèles, en particulier celles conçues sous la forme d'un graphe de flot de données. Virtue [46] contient un graphe d'appel de fonctions intégré au sein d'un environnement en trois dimensions, permettant à l'utilisateur de manipuler la représentation (rotation, sélection de parties) afin

d'obtenir plus d'informations sur les éléments affichés. L'outil Extravis [47] propose une représentation circulaire (Figure 2.4) sur le bord de laquelle les différentes fonctions sont indiquées. Un gradient de couleur appliqué sur les liens donne au choix des informations sur le sens des appels, ou sur leur survenue dans le temps.

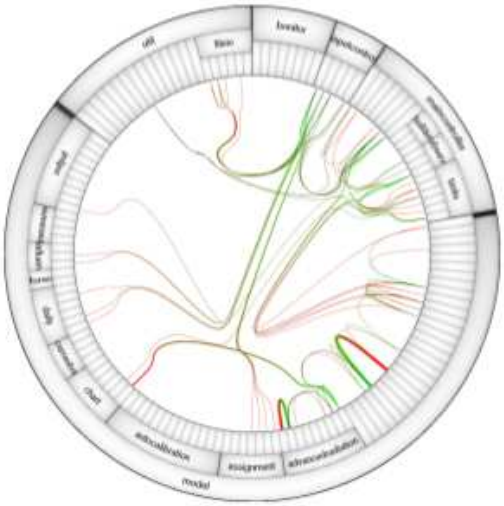


FIGURE 2.4 – Graphe d'appels de fonctions circulaire d'Extravis. Les liens symbolisent les appels entre les fonctions, le gradient de couleur indiquant l'appelant et l'appelé.

Communications Les communications entre différents processus à un instant donné peuvent être matérialisées par des matrices. Ces dernières consistent en une représentation à deux dimensions, sous la forme d'une grille. Des couleurs peuvent être employées pour indiquer le type de communications ou la taille des données transmises. Elles sont un outil idéal pour des applications parallèles en permettant la détection de goulets d'étranglement, et figurent parmi les choix de visualisation proposées par Vampir [48]. Une méthode alternative pour représenter les communications est le graphe orienté. Les entités sont symbolisées par les nœuds, tandis que les liens indiquent les communications. On en trouve une implémentation dans ParaGraph. Une version étendue est implémentée dans Triva [49, 50]. La position relative des entités représente la topologie réelle du réseau et un mécanisme d'animation temporel permet d'analyser au cours du temps la variation des volumes de communications.

Structure de la plateforme La hiérarchie des ressources et la topologie d'une plateforme peuvent être représentées à l'aide d'arbres. On peut citer la représentation *treemap* inventée par Shneiderman [51] qui consiste à diviser l'espace de visualisation (l'écran) en représentant chacune des entités sous la forme d'un rectangle, dont la surface est proportionnelle à la valeur de la variable que l'on veut mettre en évidence. La treemap est particulièrement utile pour représenter un grand nombre d'entités grâce au fait qu'elle exploite tout l'espace disponible de l'écran. Elle offre également la possibilité de choisir quel niveau hiérarchique afficher. Elle est employée dans Triva [52], pour montrer la

répartition des tâches sur de larges architectures distribuées.

2.3 Passage à l'échelle de l'analyse

Nous avons évoqué, Section 1.2, la problématique liée au passage à l'échelle des techniques de visualisations, et notamment dues aux limitations de l'écran, aux performances, et aux limitations cognitives et perceptives de l'utilisateur. Les méthodes présentées ici ne répondent pas toutes de la même manière aux contraintes liées à la représentation des ressources (nombre important d'entités produisant les événements, hiérarchie complexe) ou à la diversité et à la quantité des informations à analyser (durée de la trace, nombre d'événements, hétérogénéité des durées des événements, et plus globalement, taille de la trace).

Les représentations statistiques globales ne souffrent pas des limitations de l'écran. Le fait de ne pas associer à chaque évènement de la trace un objet graphique minimise l'utilisation de la mémoire, tandis que l'information transmise par la représentation, de nature synthétique, se concilie bien avec les limites perceptives et cognitives de l'utilisateur. Cependant, ces statistiques ne fournissent pas une première approche satisfaisante selon les dimensions spatiales et temporelles. De même, la spécialisation sur une métrique particulière des statistiques axées sur l'espace ou le temps ne leur permet pas de fournir une vue d'ensemble idéale du comportement de l'application.

Les limitations de l'écran sont particulièrement gênantes dans le cas des diagrammes espace-temps, où les deux axes sont concernés. Ce sont également les représentations qui souffrent le plus de problèmes de performances (arrêt intempestif de l'application, lenteur et manque de fluidité dans la navigation), du fait de la quantité d'évènements à traiter, à mettre en mémoire, à afficher, et avec lesquels il faut assurer l'interaction. Ces phénomènes combinés participent à accroître l'inconfort de l'utilisateur déjà limité par ses capacités à analyser une grande quantité de données.

Il est donc nécessaire de proposer des solutions pour le passage à l'échelle de ce genre de représentations spatiotemporelles, et ce, pour pouvoir fournir un point d'entrée à l'analyse, et permettre la visualisation de traces dont la complexité et le volume deviennent de plus en plus imposants. Dans le Chapitre suivant, nous abordons les techniques d'agrégation, employées par un certain nombre de techniques d'analyse spatiotemporelles pour résoudre cette problématique.

CHAPITRE 3

L'agrégation comme support au passage à l'échelle de la visualisation de traces

La représentation de l'information est une tâche complexe qui se heurte souvent à un problème de passage à l'échelle : la quantité d'information à représenter devient généralement trop importante en regard des capacités d'analyse de l'observateur à mesure que le sujet analysé gagne en détails et en réalisme. Il devient alors crucial d'élaborer des méthodes permettant d'obtenir, à partir d'informations brutes, une représentation plus compacte qui conserve la partie la plus pertinente de l'information à laquelle elle est associée. Cet intérêt d'agréger et fusionner de l'information dès lors que l'on cherche à réduire sa quantité ou simplifier sa compréhension est évoqué par Detyniecki dans son manuscrit [53]. Les méthodes que nous désignons de manière générale sous le terme d'agrégation se retrouvent dans des domaines aussi variés que le traitement d'images, la prise de décision, la reconnaissance de motifs, l'apprentissage automatique, et la visualisation d'information, dont fait partie la visualisation de traces. Leur objectif commun est de permettre d'obtenir une interprétation globale à partir de plusieurs données potentiellement fournies par de multiples sources d'information. Dans ce chapitre, nous étudions ainsi les méthodes d'agrégation employées par les techniques de visualisation de traces pour permettre leur passage à l'échelle, que nous définissons, classons et comparons selon différents critères. Cette classification est une contribution originale qui a fait l'objet d'une publication [10].

3.1 Définition et propriétés de l'agrégation

Une définition générale de l'agrégation est l'action de réunir ou de combiner un ensemble d'objets ou d'éléments distincts mais de même nature, produisant un objet ou élément unique qui les représente et les synthétise : l'agrégat.

3.1.1 Agrégation mathématique

Définition

Au niveau mathématique, l'agrégation est une application réalisée par un opérateur qui possède certaines propriétés. Detyniecki [53] en donne une définition simple :

Définition 3.1 L'opérateur d'agrégation est une fonction qui associe un nombre réel y à un n-uplet $(x_1, x_2, ..., x_n)$ de nombres réels :

$$y = \text{Aggreg}(x_1, x_2, ..., x_n) \tag{3.1}$$

Néanmoins, cette définition est limitée : il est en effet imaginable de l'étendre à d'autres objets mathématiques (dans cette thèse, nous agrégerons des vecteurs et des matrices sur le même principe), et il n'est pas nécessaire de la contraindre aux nombres réels.

Propriétés

En principe, toute fonction à plusieurs variables est un opérateur d'agrégation. Cependant, agréger sous-entend que le résultat produit puisse apporter une connaissance, une information utile sur les données d'origine. Plusieurs auteurs [54, 55, 56] ont proposé des propriétés qui garantissent la pertinence et le bon comportement d'un opérateur d'agrégation. On distingue les propriétés mathématiques et les propriétés comportementales [57, 58, 59].

Les propriétés mathématiques associées à un opérateur d'agrégation sont des propriétés *algébriques* ou *topologiques*. Elles traduisent généralement une absence de comportement chaotique de l'opérateur, en particulier une relative indépendance aux faibles variations des données en entrée ou à la manière de décomposer l'agrégation en étapes. Les plus importantes sont l'*unanimité pour les valeurs extrêmes* [56,59], la *monotonie* [56], la *continuité* [53, 59], l'*associativité* [53, 59], la *symétrie* [53], la *bisymétrie* [53], la présence d'un *élément absorbant* [53], d'un *élément neutre* [53], l'*idempotence* [53], la *compensation* [53], le *contrebalancement* [53], le *renforcement* [60], la *stabilité pour le changement d'échelle* [59,61,62], ou encore l'*invariance* [63]. Un opérateur d'agrégation ne respecte généralement pas l'ensemble de ces propriétés, mais les propriétés qu'il respecte renseignent sur son comportement.

Les propriétés comportementales sont plus spécifiques. Leur intérêt dépend principalement du contexte de l'analyse. Elle sont essentiellement liées à la nature de l'information à analyser et aux particularités de son interprétation. Les opérateurs peuvent, par exemple, exprimer des *comportements décisionnels*, comme tolérant, optimiste, pessimiste ou strict [59]. La propriété d'*interprétabilité* interdit d'utiliser des boites noires, dans le but de fournir une interprétation sémantique compréhensible pour le décisionnaire [59]. L'expression de *pondérations* permet de moduler l'importance de chacun des arguments, certains pouvant avoir une importance plus élevée dans l'expression du comportement du système ou la prise de décision [59].

Exemples d'opérateurs d'agrégation

Grabisch & Perny [59] définissent quatre grandes classes d'opérateurs d'agrégation : les opérateurs *conjonctifs*, les opérateurs *disjonctifs*, les opérateurs de *compromis*, et enfin, les opérateurs *hybrides*.

Les opérateurs *conjonctifs* agrègent des quantités à la manière d'un *et* logique, c'est-à-dire que pour que le résultat d'une agrégation soit élevé, toutes les quantités à agréger doivent être élevées. Ce type d'opérateurs a un comportement *strict* ou *pessimiste*. La famille des *normes triangulaires* ou t-normes [57, 64] est un cas particulier d'opérateurs conjonctifs. Le *minimum* en est le représentant le plus connu.

À contrario, les opérateurs *disjonctifs* se comportent comme un *ou* logique : pour que le résultat d'une agrégation soit élevé, au moins une des quantités à agréger doit être élevée. Ce type d'opérateurs peut être associé à un comportement *tolérant* ou *optimiste*. La famille des *conormes triangulaires* ou *t-conormes* [57, 64] est un cas particulier d'opérateurs disjonctifs, à laquelle appartient le *maximum*.

Les opérateurs de *compromis* permettent des nuances plus grandes dans la représentation d'un comportement. Ils ont tendance à être plus représentatifs de l'ensemble de l'information agrégée au prix d'une perte de détails sur les singularités dans la distribution des données. Parmi les plus connus, on compte la somme et la moyenne arithmétique. D'autres opérateurs de moyenne (géométrique, harmonique, quadratique) [65,66,67] permettent de faire ressortir ou de masquer certaines particularités dans la distribution des données. La médiane fournit une valeur intermédiaire obtenue à partir de la distribution des données, ce qui annule l'influence des valeurs extrêmes. La statistique d'ordre k [59] et les travaux de Calvo & Mestar [68] en sont une généralisation.

Il existe des versions pondérées de ces opérateurs : moyenne arithmétique pondérée [53, 59], moyennes quasi arithmétiques [53, 59], maximum et minimum pondérés [69], ce qui leur permet d'exprimer un compromis. La somme pondérée ordonnée (OWA, pour Ordered Weight Average) [58, 70, 71] est une moyenne arithmétique pondérée, dont les arguments sont triés par rangs. Elle est une généralisation des opérateurs moyenne, minimum, maximum, médiane et autres statistiques d'ordre k, ce qui lui permet d'exprimer des caractéristiques de la distribution.

Les intégrales floues proposées par Sugeno [72] et Choquet [73] sont fréquemment employées pour des décisions multicritères. Ces intégrales sont basées sur la notion de mesure floue, qui permet de quantifier l'importance d'un groupe de critères à travers des pondérations. Elles forment une généralisation plus poussée encore que l'opérateur OWA et peuvent exprimer l'ensemble des opérateurs évoqués ici [74, 75, 76].

3.1.2 Processus de réduction de complexité dans la visualisation

Dans la visualisation d'information, les objets graphiques sont associés à de l'information fournie par les données brutes contenues dans le système à analyser. Le processus de génération de la représentation est donc composé d'étapes successives : l'interprétation des données conduit à la représentation d'objets graphiques compréhensibles par l'utilisateur. Pour simplifier la représentation, ou mettre en évidence des types de comportement particuliers, il est possible d'introduire une méthode d'agrégation dans cette séquence de traitements. Chaque outil de visualisation employant de l'agrégation procède selon une technique et une implémentation particulière. On peut cependant distinguer

deux tendances : l'*agrégation de données* s'applique aux données brutes contenues dans le système à analyser, dont les agrégats sont ensuite représentés lors de la visualisation, tandis que l'*agrégation visuelle* s'applique aux objets graphiques pendant leur rendu.

Ce que nous appelons ici agrégation est en réalité un processus de réduction de complexité faisant appel à des opérateurs d'agrégation, mathématiques dans le cas de l'agrégation de données, graphiques dans le cas de l'agrégation visuelle. En effet, l'ensemble des éléments constituant le système n'est pas agrégé en un seul agrégat : le processus de réduction de complexité définit des ensembles éléments à agréger ensemble selon certains critères, communs à l'ensemble des techniques de visualisation d'information, et que nous emploierons au cours de cette thèse pour définir nos propres techniques [14]. Nous distinguons ainsi :

— les *dimensions* selon lesquelles agréger (le temps, l'espace, etc.) ;
— les *opérandes*[1], c'est-à-dire l'ensemble des objets évalués par le processus de réduction (mais pas forcément agrégés) : il peut s'agir de l'ensemble des données, des objets graphiques, ou d'un sous-ensemble spécifique d'objets du système ou de la représentation (si l'on souhaite, par exemple, n'agréger ensemble que des états, des liens ou des variables) ;
— les *contraintes*, qui définissent quelle relation les objets doivent avoir entre eux pour pouvoir être agrégés. En d'autres termes, quels sont les agrégats *autorisés* et ceux qui sont *interdits*. Dans le cas de l'agrégation temporelle, on peut penser notamment à la préservation de la contiguïté temporelle en n'agrégeant que des objets contigus selon l'axe temporel et en interdisant l'agrégation d'éléments disjoints. Dans le cas de l'agrégation spatiale (c'est-à-dire des ressources), il peut s'agir également de préserver la position (déterminée selon un ordre arbitraire, comme le numéro de processus, par exemple), ou de satisfaire des contraintes hiérarchiques où les nœuds d'un arbre ne peuvent être agrégés qu'au sein du nœud père ;
— les *conditions*, qui sont les facteurs qui *déclenchent* l'agrégation. La condition permet de sélectionner plusieurs sous-ensembles d'opérandes. Sur chacun de ces sous-ensembles est alors appliquée une opération d'agrégation. Dans le cas de l'agrégation visuelle, on emploie fréquemment comme condition la taille et la position des opérandes, qui, s'ils ne peuvent pas être représentés correctement à cause des limites de la résolution, sont agrégés. Dans le cas d'agrégation de données, il peut s'agir par exemple d'une contrainte fournie par l'utilisateur, liée à la quantité d'information perdue par l'agrégation ou à la réduction de complexité qu'elle permet, telle que proposée par la *méthode de Lamarche-Perrin* décrite dans le Chapitre suivant. Enfin, des techniques comme les représentations guidées par les pixels combinent des contraintes graphiques (la dimension est discrétisée en fonction des pixels disponibles) à une agrégation de données (les évènements situés dans l'intervalle correspondant à un pixel sont agrégés entre eux) ;
— l'*opérateur*, qui est la fonction ou la transformation appliquée sur chaque sous-ensemble d'opérandes destinés à être agrégés. Il s'agit des opérateurs mathématiques définis 3.1.1, ou de la production d'un nouvel objet graphique dont les

[1] Attention à la distinction entre les opérandes du processus de réduction de complexité dans son ensemble et les opérandes d'un opérateur d'agrégation, dont les seconds représentent ici un sous-ensemble des premiers.

caractéristiques dépendent des éléments qu'il représente.

La Figure 3.1 donne un exemple d'agrégation visuelle synthétique qui nous permet d'illustrer ces différents critères[2]. On agrège selon l'unique dimension. Les opérandes sont l'ensemble des objets graphiques. On ne peut agréger que des objets contigus. La condition d'agrégation est le partage d'une caractéristique commune parmi la couleur et la forme. Enfin, l'opération d'agrégation produit un objet graphique dont la largeur est la largeur totale des objets qu'il agrège, dont la couleur est celle de ses opérandes si elle est identique, gris sinon, et dont la forme est celle de ses opérandes, contenue dans un rectangle si elle est identique, ou un rectangle coloré sinon.

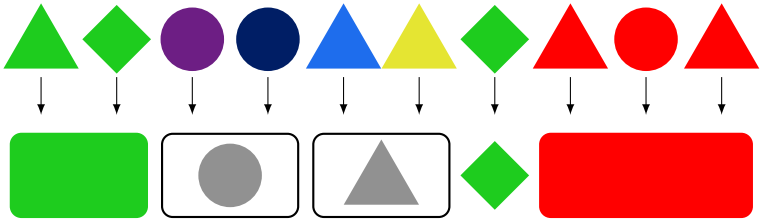


FIGURE 3.1 – Exemple d'un processus d'agrégation visuelle. Les objets graphiques sont agrégés ensemble si ils sont contigus et si leur couleur ou leur forme est identique. L'agrégat obtenu possède les attributs qui sont communs aux opérandes.

3.2 Critères qualitatifs d'évaluation d'une représentation agrégée

L'agrégation est une opération qui possède un coût : tous les opérateurs, qu'ils soient mathématiques ou graphiques, provoquent une perte d'information sur les données agrégées si ces dernières ne sont pas strictement égales. Ainsi, bien qu'elle aide au passage à l'échelle de l'analyse, l'agrégation est susceptible de modifier l'interprétation de la représentation. Il est donc primordial de respecter certaines règles de conception afin de ne pas introduire de biais dans l'analyse.

Elmqvist & Fekete [4] proposent une méthodologie pour construire une vue d'ensemble. Bien que destinée originellement à des représentations faisant appel à de l'agrégation hiérarchique, elle est pertinente pour d'autres types de visualisations agrégées. Nous présentons ici les six critères qui la composent (dont nous n'avons volontairement pas traduit le nom) que nous notons Gx, et qu'une visualisation cohérente doit fournir. Ultérieurement, dans le cas où l'un de ces critères n'est pas respecté par une technique de visualisation, nous utiliserons la convention $\overline{\text{Gx}}$.

G1. Entity Budget La résolution d'un écran d'ordinateur ne permet d'afficher qu'un certain nombre d'entités graphiques, dont la taille la plus petite possible est le pixel. Un autre facteur à prendre en compte est la capacité d'un être humain à percevoir et analyser un grand nombre d'objets graphiques. Remplir le critère G1 nécessite donc d'adapter le nombre d'entités afin que leur rendu soit graphiquement possible et que la complexité de la représentation ne nuise pas à sa perception par un analyste.

[2]Ce genre d'agrégation visuelle n'est pas employé en pratique et sert seulement d'exemple.

G2. Visual Summary Les agrégats visuels doivent contenir de l'information sur les données qu'ils représentent.

G3. Visual Simplicity Les agrégats visuels doivent être nets et avoir une apparence visuelle simple. Utiliser des formes ou dessins trop complexes, ou tenter de représenter trop d'information dans un agrégat peut être responsable d'une visualisation trouble ou peu lisible.

G4. Discriminability Dans le cas où des données non agrégées et des agrégats visuels peuvent être présents simultanément dans la même représentation, les agrégats visuels doivent être facilement distinguables des objets graphiques associés aux données non agrégées.

G5. Fidelity Les opérations d'abstraction de données comme l'agrégation réduisent la complexité d'un ensemble de données au détriment de la quantité d'information. Cette simplification peut être responsable d'une perte de fidélité, selon les opérateurs mis en jeux : par exemple, une moyenne fera disparaître les informations sur la distribution et les valeurs extrêmes, alors que ces dernières peuvent traduire un phénomène particulier [77]. Au niveau de la visualisation, cela peut conduire à déformer voire ignorer un effet [78, 79]. A défaut de pouvoir corriger la perte d'information et de fidélité liées au processus d'agrégation, il est important d'en informer l'utilisateur explicitement ou au travers de la représentation des agrégats.

G6. Interpretability Ce principe contre-balance le critère G1 : l'intensité de l'agrégation doit être adaptée de manière à ce que la visualisation contienne toujours du sens. Comme l'agrégation est responsable d'une perte d'information (G5), il est important d'en minimiser les effets en agrégeant seulement quand cela est nécessaire. Enfin, la symbolique utilisée pour indiquer un agrégat visuel devrait inciter l'utilisateur à *aller y chercher* plus d'information, au moyen d'une interaction avec la visualisation, par exemple.

3.3 Techniques de visualisation employant de l'agrégation

Nous présentons ici un certain nombre de techniques de visualisations temporelles, spatiales ou spatiotemporelles employant de l'agrégation, en évaluant les critères de qualité décrits Section 3.2. Elles sont listées dans la Table 3.1.

3.3.1 Techniques fondées sur l'agrégation visuelle

Nous énumérons ici différentes techniques de visualisations faisant appel à l'agrégation visuelle pour assurer le passage à l'échelle.

Agrégation visuelle par contiguité C'est un cas de l'agrégation visuelle où le rendu essaie de préserver l'échelle des objets graphiques, dont la taille dépend de leur contenu (par exemple, la durée d'un état, dans un diagramme espace-temps, détermine la largeur du rectangle associé). Quand il est impossible d'allouer une place suffisante à la représentation d'un objet graphique (la taille calculée est par exemple inférieure à un

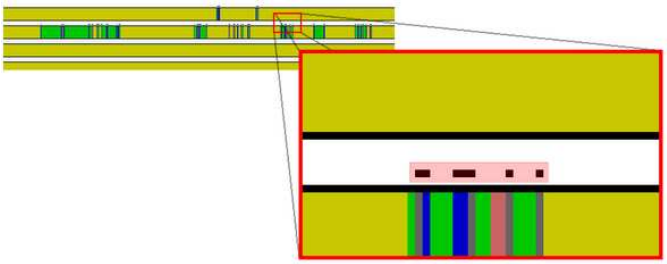


FIGURE 3.2 – Démonstration du principe d'agrégation de LTTng Eclipse Viewer [44], qui s'active en cas de zoom, lorsque des états deviennent trop petits pour être visibles. Les agrégats visuels sont indiqués à l'aide de points. Si plusieurs états identiques sont agrégés, on les représente en utilisant leur couleur originale. A l'inverse, s'il s'agit de deux états différents, on utilise le gris.

pixel), ce dernier est agrégé avec l'un de ses voisins. Un tel mécanisme est employé pour l'agrégation des états dans le diagramme espace-temps de LTTng Eclipse Viewer [44], illustré Figure 3.2, ou dans Pajé [30], et est signifié à l'utilisateur à l'aide d'un symbole. Cependant, un agrégat mixant plusieurs types d'évènements ne contient pas de détails sur ces derniers [44], ou ne représente que la proportion de l'état majoritaire [30]. De manière générale, ce type d'agrégation visuelle est peu utilisable pour fournir une vue synthétique compréhensible d'une trace volumineuse contenant beaucoup d'évènements, car les agrégats peuvent englober trop d'états, ce qui rend la représentation peu pertinente ($\overline{G2}$). Son domaine d'action se situe surtout à une échelle intermédiaire, où elle s'avère confortable pour analyser des états de durées hétérogènes, en agrégeant les plus petits et en laissant intacts les plus longs.

Compression temporelle T-Charts, le diagramme espace-temps de STWorkbench [7], l'infrastructure d'analyse de systèmes embarqués de STMicroelectronics, possède un mécanisme de compression temporelle purement graphique, distordant une zone de temps si les processus restent tous dans le même état, plutôt que de l'afficher en entier. Cependant, cette technique n'est efficace qu'à une échelle détaillée, puisque le déclencheur de la compression requiert que les ressources ne changent pas d'état sur l'intervalle compressé. De plus, cette technique peut modifier la perception du comportement des ressources par l'utilisateur ($\overline{G4}$), en particulier en donnant l'illusion de motifs périodiques : la zone compressée est représentée avec une largeur fixe, et ce, quelle que soit la durée de l'intervalle original. La Figure 3.3 illustre la compression temporelle de T-Charts.

3.3.2 Techniques hybrides

Ces techniques combinent agrégation de données et agrégation visuelle.

Représentations guidées par les pixels Les représentations guidées par les pixels [48, 80] associent à chaque pixel disponible un ensemble de données. Comme un pixel est incapable de représenter toute l'information associée, l'algorithme de rendu décide de ce qui est montré ou caché. Cette technique permet de représenter n'importe quel système mono ou bidimensionnel (si on ne considère pas les limitations liées au temps de calcul

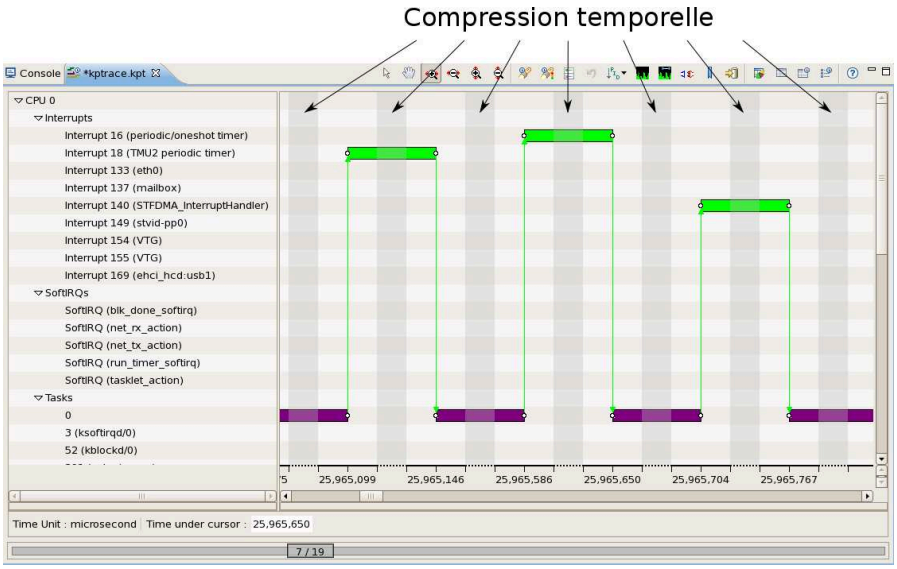


FIGURE 3.3 – Mécanisme de compression temporel de T-Charts [7]. On note que les intervalles agrégés n'ont pas tous la même taille, mais que l'utilisation d'une largeur fixe pour les représenter donne une fausse impression de régularité.

et à la mémoire), quel que soit le nombre d'évènements survenant au cours du temps et le nombre de ressources. L'agrégation est uniforme, et chaque pixel est un agrégat spatiotemporel. Néanmoins, il est possible qu'un état, par exemple, dure plus longtemps que la tranche de temps associée à un pixel. Dans ce cas, il est impossible de distinguer ($\overline{G4}$) si :

— un pixel représente un ensemble de données agrégées ;
— un pixel représente une seule donnée ou une partie d'une donnée.

L'opérateur d'agrégation employé n'est pas clairement indiqué à l'utilisateur dans Vampir [48] ($\overline{G6}$). À l'inverse, Paraver [80], permet de choisir l'opérateur conduisant à la représentation du pixel (moyenne, maximum, minimum, ou même aléatoire). On peut noter une instabilité vis-à-vis du redimensionnement lors de l'emploi de ce genre de techniques, due à la discrétisation basée sur les pixels : la variation de la taille allouée à la fenêtre change la durée de tranche de temps ou le nombre de ressources associées à chaque pixel. Cela pose des problèmes de cohérence et donc de fidélité des comportements retranscrits ($\overline{G5}$). Ce défaut est mis en évidence par la Figure 3.4 qui montre trois *timelines* de Vampir [48] de la même trace, générées en employant une résolution différente, et remises à la même largeur : la trame globale de chacune possède la même allure, néanmoins, l'observation des détails permet d'identifier des différences (disparition ou apparition de certains pics).

3.3.3 Techniques fondées sur l'agrégation de données

Nous évoquons ici des techniques de réduction de complexité fondées sur l'agrégation de données.

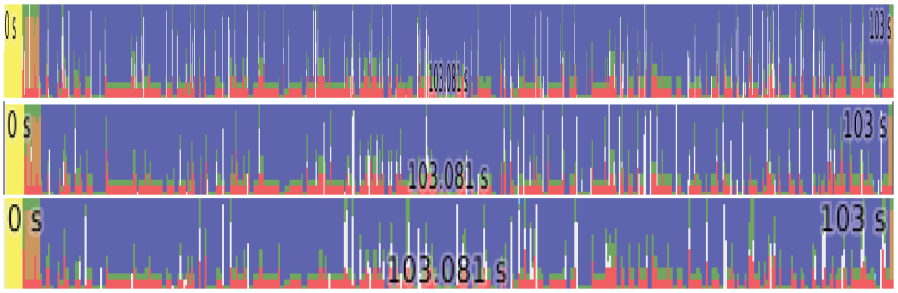


FIGURE 3.4 – Trois timelines de Vampir [48] de la même trace, générées en employant une résolution différente, et remises à la même largeur, montrant l'influence de la discrétisation basée sur les pixels sur le résultat de l'agrégation.

Agrégation de données par tranches de temps La « Streamline » d'ARM DS5 [45] possède une représentation proche du diagramme espace-temps. L'axe du temps est discrétisé sous la forme de tranches de temps, chacune d'elles étant associées à une couleur dont l'intensité dépend de l'amplitude de la valeur d'une métrique, comme le nombre d'évènements sur cette tranche de temps. La lisibilité est cependant difficile, car l'intensité est difficile à discriminer pour certaines couleurs. Ce problème est accentué par la faible taille des objets graphiques ($\overline{\text{G3}}$).

Abstraction contenue dans la trace Jumpshot [81] tire parti d'un format de trace particulier, SLOG-2, qui permet de regrouper les évènements au sein d'une hiérarchie de « bounding boxes », des sortes de *conteneurs* d'évènements. Ainsi, selon le niveau de détail désiré (représentation de la trace en entier, ou zoom sur une sous-partie), un niveau de cette hiérarchie est sélectionné, ce qui évite de lire la trace en entier et simplifie le rendu en affichant uniquement les « bounding boxes » abstrayant les évènements.

Agrégation hiérarchique T-Charts [7] possède un mécanisme d'agrégation hiérarchique interactif, où l'utilisateur peut décider d'agréger des ressources au sein de leur nœud parent, ce qui transfère la représentation de leurs évènements au niveau de ce dernier. Il est à noter que cette technique n'est possible que si il n'y a pas de chevauchement temporel entre les états des ressources à agréger.

Dans la treemap de Triva [52], l'utilisateur, en interagissant, choisit un niveau de la hiérarchie matérielle ou logicielle à afficher, et peut éventuellement zoomer sur une sous-partie de cette hiérarchie : l'élément sur lequel on se focalise devient la racine de la nouvelle hiérarchie représentée par la treemap. Les rectangles de la treemap affichée correspondent aux nœuds associés au niveau de la hiérarchie. Les valeurs qui leur sont associées sont calculées en sommant celles des nœuds qu'ils contiennent. Leur surface est ensuite déterminée à partir de ces valeurs. Les Figures 3.5 A forment un exemple de l'agrégation hiérarchique de Triva : A représente le niveau processus, A.1 le niveau machine, A.2 le niveau cluster et A.3 l'ensemble du système.

La « Scalable Topology-based Visualization » [49, 50] du même outil permet de représenter la topologie et les communications au moyen d'un graphe, où les nœuds sont les ressources et les arcs les communications. Un exemple en est donné Figure 3.6. Cette technique est idéale pour observer les goulets d'étranglement au niveau des communica-

tions. Elle bénéficie des mêmes mécanismes d'interaction que la treemap, ce qui permet d'agréger ou d'« exploser » les entités. Il est aussi possible de déplacer interactivement les éléments de la représentation pour plus de lisibilité, un mécanisme simulant des forces de répulsion et d'attraction faisant progressivement revenir le graphe dans sa configuration initiale afin de maintenir la cohérence de la représentation.

Enfin, une version récente de Triva, Viva [5, 6], introduit une représentation treemap multirésolution. Cette technique est une application des travaux de Robin Lamarche-Perrin [8] effectués durant sa thèse de doctorat, qui propose une méthode de partition et d'agrégation de données, guidée par l'information contenue dans la représentation, et par des contraintes exprimées grâce à la connaissance de la structure du système. Lamarche-Perrin a également conçu deux algorithmes : l'un agrège un ensemble ordonné, l'autre, implémenté au sein de Viva, un ensemble hiérarchique. Ce dernier agrège en priorité les nœuds dont les valeurs sont les plus homogènes, en respectant une contrainte hiérarchique, tandis que ceux dont les valeurs sont les plus hétérogènes restent désagrégés, ce qui met en évidence les « outliers » et corrige la perte d'information sur la distribution des deux précédentes techniques (G5). Son principe est illustré par les treemaps B et C de la Figure 3.5. L'utilisateur fait varier le niveau de détails, exprimé sous la forme d'un compromis entre la perte d'information et la complexité de la représentation : le niveau B est ici plus détaillé que le niveau C. Dans les deux représentations, on détecte une zone particulièrement hétérogène et potentiellement problématique, qui reste désagrégée. L'analyste est ainsi incité à aller explorer cette zone. En comparaison avec l'agrégation hiérarchique classique, il peut toutefois être difficile de maintenir le budget d'entités graphiques, puisque certaines configurations peuvent mener à laisser désagréger une zone de la trace alors que les objets graphiques possèdent une petite surface (G1).

Clustering spatial Vampir [48] contient une technique de partitionnement spatial (Figure 3.7) qui regroupe les processus les plus ressemblants au moyen d'une mesure de distance. Elle montre, pour chaque groupe, le temps d'exécution total moyen des fonctions appelées par les processus, sous la forme d'un histogramme empilé. Cette représentation est plutôt destiné à du profilage, car elle ne montre pas l'évolution du comportement au cours du temps.

3.3.4 Discussion

Nous discutons ici de certaines caractéristiques structurelles des représentations pouvant influencer le respect des critères d'Elmqvist & Fekete.

Influence de la multidimensionnalité

Parmi les techniques synthétisées dans la Table 3.1, certaines, comme Triva/Viva, le « task profile » et la « timeline » de Vampir, ne représentent que l'espace ou le temps. Cette particularité a une forte influence sur l'analyse, puisqu'elle est responsable d'un lissage de la dimension agrégée ($\overline{\text{G5}}$) : des représentations ne montrant pas l'espace des ressources de l'application ne permettent pas d'étudier l'influence de sa structure sur son comportement, tandis que les représentations s'affranchissant d'un axe temporel éprouvent des difficultés à mettre en évidence un comportement problématique localisé dans le temps et sont plutôt destinées au profilage de l'application. L'absence d'une

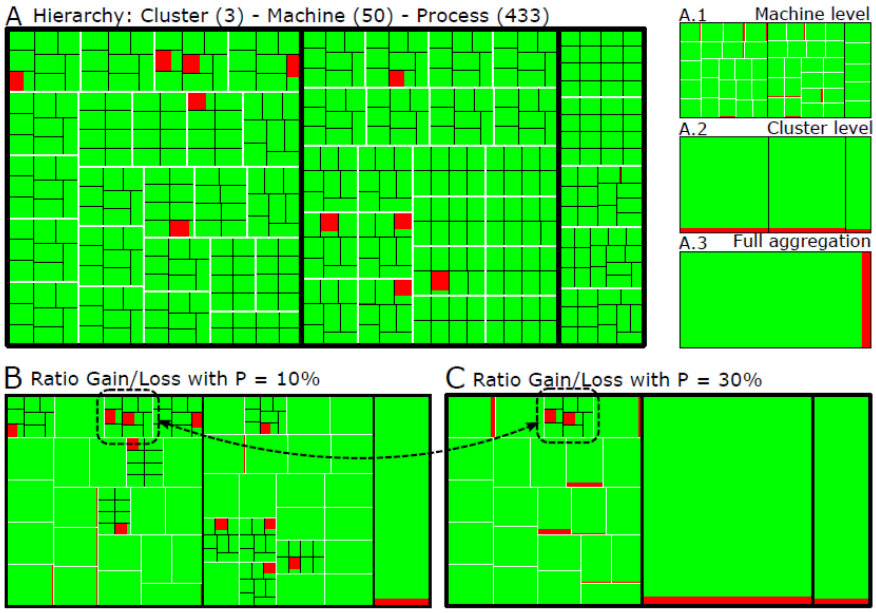


FIGURE 3.5 – Treemaps proposées par Viva [6] (ex-Triva [50]). Les représentations A sont un exemple d'agrégation selon la hiérarchie matérielle et logicielle. Les représentations B et C emploient de l'agrégation multirésolution. Le niveau de détail diminue quand p augmente.

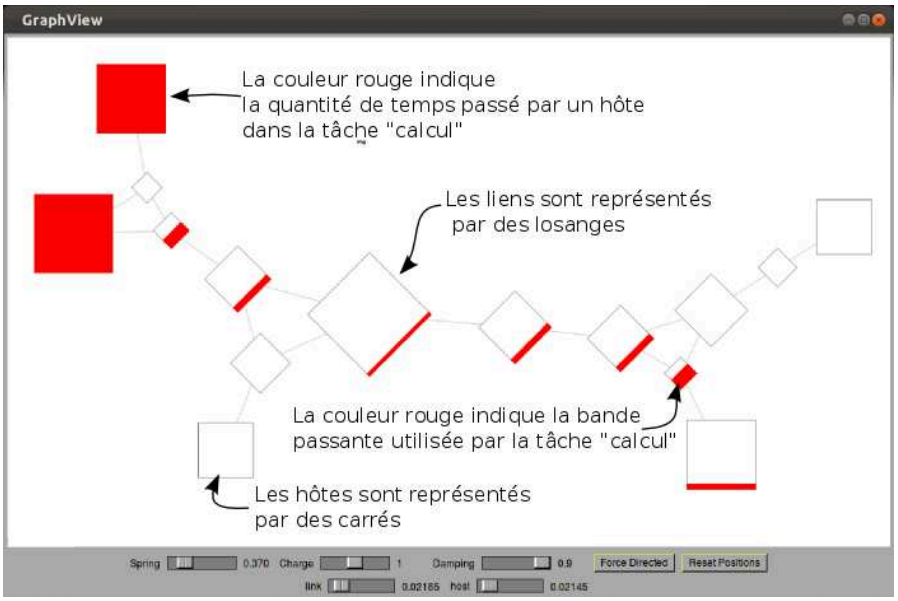


FIGURE 3.6 – La « Scalable Topology-based Visualization » de Triva [49, 50], montrant la topologie de l'application, les nœuds, les liens, et leur taux d'utilisation.

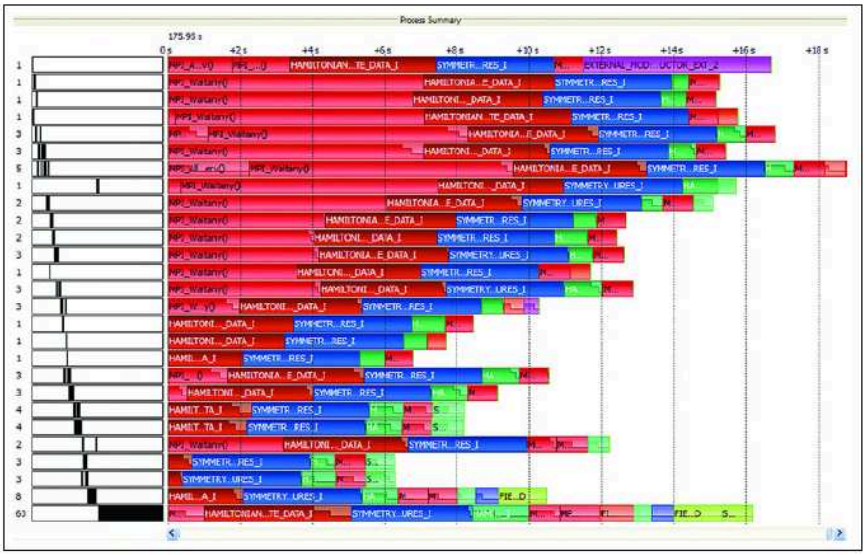


FIGURE 3.7 – Le « task profile » de Vampir [48] regroupe les processus les plus ressemblants et montre le temps moyen des fonctions associées aux processus agrégés au sein de la grappe.

dimension donne cependant l'opportunité de mieux maîtriser la place allouée à la représentation.

À l'inverse, les représentations multidimensionnelles sont capables de lier la structure et le comportement temporel de l'application, mais ont plus de difficultés à satisfaire la contrainte liée au respect du budget des éléments graphiques ($\overline{\text{G1}}$). En effet, la plupart des techniques d'agrégation employées traitent l'espace et le temps séparément, et de manière inégale. Les diagrammes espace-temps de Vampir [48], Jumpshot [81], Pajé [30], LTTng Eclipse viewer [44] et la Streamline d'ARM DS 5 [45] ne proposent pas d'agrégation spatiale. La technique d'agrégation hiérarchique de T-Charts [7] est relativement efficace, mais la compression temporelle n'est pas suffisante pour assurer le passage à l'échelle pour la représentation de la trace entière. Parmi les représentations multidimensionnelles, seul Paraver [80] est à même de proposer un diagramme espace-temps avec une agrégation dont le procédé est identique sur l'espace et le temps et qui respecte le critère G1. Cette constatation nous conduit à élaborer un nouveau critère que nous intégrons à la Table 3.1 : la cohérence du processus d'agrégation sur le temps et l'espace, que nous notons **M**.

M. Cohérence de l'agrégation sur l'ensemble des dimensions Une agrégation est cohérente sur les deux dimensions si elle les agrège au cours d'un même processus[3] de telle sorte à ce que les paramètres liés à l'agrégation d'une des deux dimensions influent sur le résultat de l'agrégation de la seconde[4]. L'intérêt d'une agrégation cohérente est de considérer le maintien du budget d'entités (G1), à travers le processus de réduction de complexité, sur l'ensemble de la représentation, et non séparément sur chaque dimension.

[3]Cela ne signifie pas qu'elles sont forcément agrégées de la même manière.

[4]Mais il n'est pas nécessaire que les paramètres de l'agrégation de la seconde dimension influent sur le résultat de l'agrégation de la première pour considérer qu'il y a cohérence.

Les techniques proposant une vue d'ensemble monodimensionnelle remplissent M de par leur construction (le fait de ne pas montrer une dimension forçant à l'agréger complètement).

Conception d'une technique agrégation répondant à l'ensemble des critères de qualité

Nous avons discriminé un certain nombre de défauts liés aux techniques d'agrégation présentées. D'après ce constat, nous estimons qu'il est important que l'agrégation soit cohérente (M), en particulier dans le cas de représentations multidimensionnelles. Les techniques fondées sur l'agrégation de données sont globalement les plus satisfaisantes, quoique souffrant encore de certains défauts. L'agrégation multirésolution employée par Viva [5, 6] nous semble pertinente puisqu'elle est cohérente sur l'ensemble des dimensions, et permet de remplir la plupart des critères, hormis G1 pour certaines valeurs du compromis entre la réduction de complexité et la perte de d'information. Nous envisageons d'employer la même méthode sur d'autres dimensions, en la complétant par des techniques d'agrégation visuelle pour assurer le critère G1. Nous estimons que ces dernières peuvent fournir un bon complément à l'agrégation de données : le processus de réduction de complexité étant déjà amorcé, elles ne sont appliquées que sur un ensemble restreint d'objets graphiques, ce qui limite la perte d'information (G5) dont elles souffrent habituellement.

Nous nous proposons donc, dans un premier temps, d'appliquer l'algorithme d'agrégation d'un ensemble ordonné issu des travaux de Lamarche-Perrin à une analyse temporelle, en vue de l'analyse d'applications embarquées dont l'homogénéité temporelle est déterminante. Notre contribution ici est de l'adapter à l'analyse de traces en définissant comment construire l'entrée de l'algorithme, afin que ses structures et sa métrique soient adaptées à la représentation du comportement de l'application. Ce modèle microscopique étant plus complexe que ceux employés par les applications initiales de l'algorithme d'agrégation d'un ensemble ordonné, il est nécessaire d'adapter ce dernier à des attributs multidimensionnels. La *méthode de Lamarche-Perrin* ne préconise pas comment visualiser la sortie de l'algorithme d'agrégation, si bien qu'il est nécessaire de concevoir des techniques de visualisation adaptées. Enfin, la réduction de complexité n'étant pas uniforme dans la représentation, nous ajoutons des mécanismes d'agrégation visuelle afin de remédier à la présence ponctuelle d'objets graphiques trop petits pour être affichés correctement. Bien que notre technique ne représente pas l'espace des ressources, nous n'agrégeons pas cette dimension lors de l'agrégation des données, afin de la prendre en compte pour déterminer comment agréger temporellement. Cela diffère de l'agrégation hiérarchique de Viva où le temps est complètement agrégé au travers d'une intégration temporelle. Dans un second temps, nous fusionnons l'agrégation temporelle et spatiale au sein d'un algorithme unique, proposant ainsi une méthode d'agrégation multidimensionnelle de données uniforme, également complétée par de l'agrégation visuelle.

Avant de décrire ces deux techniques d'analyse, nous présentons, Chapitre 4 une synthèse des travaux de Lamarche-Perrin afin de familiariser le lecteur aux concepts issus de sa méthode.

TABLE 3.1 – Passage à l'échelle des techniques d'analyse spatiotemporelles implémentées dans les outils d'analyse de traces. Les critères d'Elmqvist & Fekete sont symbolisés par Gx, et le critère de cohérence multidimensionnelle par M. Un critère peut être satisfait : uniquement pour le temps ($\star$), l'espace ($\circ$) ou les deux dimensions ($\bullet$). On symbolise l'agrégation de données par $\sharp$, l'agrégation visuelle par $\flat$, et l'agrégation hybride par $\natural$.

		Axes		**Agrégation**								
Outil	**Visualisation**	*Temporel*	*Spatial*	*Temporelle*	*Spatiale*	**G1**	**G2**	**G3**	**G4**	**G5**	**G6**	**M**
Vampir [48]	Diagramme espace-temps	$\bullet$	$\bullet$	Guidée par les pixels $\natural$		$\star$	$\bullet$	$\bullet$				
	Timeline	$\bullet$		Guidée par les pixels $\natural$	Guidée par les pixels $\natural$	$\bullet$	$\star$	$\bullet$				$\bullet$
	Task Profile		$\bullet$	Moyenne $\sharp$	Clustering $\sharp$	$\bullet$	$\bullet$	$\bullet$	$\bullet$		$\bullet$	$\bullet$
Paraver [80]	Diagramme espace-temps	$\bullet$	$\bullet$	Guidée par les pixels $\natural$	Guidée par les pixels $\natural$	$\bullet$	$\bullet$	$\bullet$			$\bullet$	$\bullet$
LTTng Eclipse Viewer [44]	Diagramme espace-temps	$\bullet$	$\bullet$	Agrégation par contiguïté $\flat$		$\star$		$\bullet$	$\bullet$		$\bullet$	
Pajé [30]	Diagramme espace-temps	$\bullet$	$\bullet$	Agrégation par contiguïté $\flat$		$\star$		$\bullet$	$\bullet$		$\bullet$	
T-Charts [7]	Diagramme espace-temps	$\bullet$	$\bullet$	Compression temporelle $\flat$	Agrégation hiérarchique $\sharp$	$\circ$		$\bullet$	$\circ$		$\bullet$	
ARM DS-5 Streamline [45]	Diagramme espace-temps	$\bullet$	$\bullet$	Discrétisation + intensité $\sharp$		$\star$	$\bullet$		$\bullet$	$\bullet$	$\bullet$	
Jumpshot [81]	Diagramme espace-temps	$\bullet$	$\bullet$	Abstraction $\sharp$		$\star$	$\bullet$	$\bullet$	$\bullet$	$\bullet$	$\bullet$	
Triva/Viva	Treemap [52]		$\bullet$	Intégration $\sharp$	Agrégation hiérarchique $\sharp$	$\bullet$	$\bullet$	$\bullet$	$\bullet$		$\bullet$	$\bullet$
	Scalable Topology-based [49, 50]		$\bullet$	Intégration $\sharp$	Agrégation hiérarchique $\sharp$	$\bullet$	$\bullet$	$\bullet$	$\bullet$		$\bullet$	$\bullet$
	Treemap multirésolution [5, 6]		$\bullet$	Intégration $\sharp$	Agrégation hiérarchique $\sharp$		$\bullet$	$\bullet$	$\bullet$	$\bullet$	$\bullet$	$\bullet$

CHAPITRE 4

Adaptation d'une méthode d'analyse macroscopique des grands systèmes à l'analyse de traces

Lamarche-Perrin oppose deux approches visant à l'analyse d'un système multi-agents complexe de grande taille [8]. L'approche *analytique* considère chacun des constituants d'un système séparément, tandis que l'approche *systémique* l'appréhende dans sa globalité [82]. Selon lui, l'étude d'un système complexe n'est pas possible avec l'approche analytique : celle-ci nécessite des outils d'observations dont la taille est de l'ordre de celle du système, ce qui génère une quantité d'information, et donc de phénomènes microscopiques à expliquer, trop importante pour un analyste. À l'inverse, l'approche systémique s'attache à « distinguer ce qui est important de ce qui est de l'ordre du détail », ou, en d'autres termes, à proposer « un point de vue *macroscopique* » [8], ce qui permet à l'analyste de ne pas être gêné par la taille du système à étudier.

Dans leurs travaux, Lamarche-Perrin *et al.* [8, 9, 83, 84, 85, 86, 87, 88] conçoivent une méthode permettant l'élaboration d'un point de vue macroscopique selon l'approche systémique. Elle part du postulat que le système est observé avec des outils microscopiques, et s'attache à répondre à la problématique de « *mettre en évidence les phénomènes macroscopiques à partir des résultats de l'observation microscopique* » [8]. Pour cela, la méthode s'appuie sur un *processus d'abstraction* des données microscopiques, reposant sur des techniques d'agrégations. Cette abstraction est élaborée en tenant compte du contexte de l'analyse et de connaissances à priori de l'observateur sur le système, ce qui permet d'engendrer une définition macroscopique représentant des phénomènes pertinents pour l'analyste. Nous noterons l'ensemble du corpus conceptuel et méthodologique issu des ces travaux *méthode de Lamarche-Perrin*.

La définition que nous donnons d'une trace d'exécution d'un programme, Section 2.1, en fait un cas particulier d'un système multi-agents complexe, constitué d'entités et d'évènements produits par ces entités. Les problématiques soulevées par Lamarche-Perrin [8] englobent ainsi celles du passage à l'échelle de l'analyse de traces basée sur la visualisation, où il faut composer avec les *capacités de calcul* de la machine servant à l'analyse, *les capacités graphiques* de l'affichage et les *capacités analytiques* de l'observateur. Les

visualisations dont le passage à l'échelle est délicat, décrites en Section 2.3, se positionnent du côté de l'approche analytique et souffrent de ses défauts. Nous proposons donc, dans cette thèse, d'aborder l'analyse et la visualisation de traces de grands volumes en employant une approche systémique, et, plus précisément, en suivant la *méthode de Lamarche-Perrin*.

Cette approche est adaptée à notre problème car elle n'est pas uniquement focalisée sur le passage à l'échelle : elle apporte une connaissance supplémentaire sur le comportement du système en discriminant l'homogénéité et l'hétérogénéité du comportement des entités qui le composent. Cela est possible grâce à un compromis entre la réduction de complexité et la perte d'information, dont la résolution privilégie l'agrégation des éléments les plus homogènes — du point de vue de la théorie de l'information. L'homogénéité du comportement des applications embarquées temps réel au cours du temps nous intéresse particulièrement, puisque les principales caractéristiques des dysfonctionnements liés à des pertes d'image ou à des artefacts sonores se manifestent par une rupture momentanée du comportement normal de l'application, qui est censé être périodique. Dans le cas des applications parallèles, nous cherchons à discriminer des phases, et éventuellement des ruptures de comportement locales dues à des facteurs externes (contentions sur le réseau, par exemple) ou internes (interblocage, par exemple), ce qui se traduit aussi par la notion d'homogénéité temporelle. Concernant l'aspect structurel, l'étude de la répartition de la charge de travail, de la proportion de tâches ou de fonctions sur les ressources est un indicateur du bon dimensionnement de la plateforme, ou des mauvaises performances d'une sous-partie du système. Là encore, la notion d'homogénéité comportementale est présente, au niveau spatiale cette fois-ci.

Dans ce chapitre, nous reprenons et synthétisons différents travaux [8, 9, 83, 84, 85, 86, 87, 88] pour en dégager les concepts principaux et les éléments clés de la *méthode de Lamarche-Perrin*. Nous détaillons le concept de modèle microscopique, une abstraction du système à étudier qui est le niveau le plus fin de l'analyse. Nous poursuivons par le processus de partitionnement et d'agrégation de ce modèle microscopique pour en générer une représentation moins complexe, et les propriétés qui caractérisent les agrégats composant cette représentation. Nous abordons les mesures de qualités, employées pour déterminer comment choisir une partition, puis les contraintes qui sont appliquées au processus de partitionnement, pour enfin finir par des exemples d'applications. Ces points sont essentiels pour aborder les chapitres 5 et 6, dans lesquels nous expliquons comment nous avons appliqué et adapté cette méthode pour l'intégrer dans un processus d'analyse de traces.

4.1 Formalisation du processus d'agrégation

Nous formalisons le processus d'agrégation de la *méthode Lamarche-Perrin* en définissant la représentation microscopique du système, son partitionnement et son agrégation.

4.1.1 Représentation microscopique d'un système

Dans la *méthode Lamarche-Perrin*, « l'agrégation a pour point de départ une représentation microscopique du système, discrétisant une dimension de l'analyse (espace, temps, entités, *etc.*) en objets microscopiques (portions d'espace, périodes de temps, membres

du système, *etc.*) » [8]. Trois concepts permettent de décrire cette représentation microscopique : la *population*, les *individus*, et les *attributs*.

Définition 4.1 La *population* est l'*ensemble* des éléments d'une dimension discrète du système. Nous la notons Ω dans ce chapitre. Un système multidimensionnel à k dimensions est représenté par une population constituée du produit cartésien de k populations : $\Omega = \Omega_1 \times ... \times \Omega_k$. [8].

Définition 4.2 Les *individus* sont les objets microscopiques du système, c'est-à-dire les éléments x de la population Ω. Pour une population multidimensionnelle d'ordre k, un individu multidimensionnel est un k-uplet d'individus monodimensionnels associés à chacune de ces dimensions : $x = (x_1, ..., x_k) \in \Omega_1 \times ... \times \Omega_k$ [8].

Définition 4.3 Les *attributs* désignent des valeurs associées aux individus, selon une métrique représentative d'un aspect du système étudié. Plus formellement, « un attribut v est une application de Ω dans un ensemble de valeurs V qui associe à chaque individu $x \in \Omega$ une valeur $v(x) \in V$ » [8]. Dans la définition de Lamarche-Perrin, les attributs expriment un *dénombrement*, c'est-à-dire des valeurs entières positives ($V = \mathbb{N}^+$). Elles sont « interprétées comme les *quantités d'unités atomiques* associées aux individus par l'instrument d'observation ». Cependant, nous étendrons cette définition à l'ensemble des valeurs réelles positives ($V = \mathbb{R}^+$). On définit $p(x)$ comme la *probabilité d'apparition* d'un individu, selon un tirage aléatoire uniforme, parmi toutes les unités observées : $\forall x \in \Omega$, $p(x) = \frac{v(x)}{v(\Omega)}$.

Les systèmes sur lesquels nous travaillons, les traces, sont potentiellement constitués de dimensions continues (comme le temps). Afin de pouvoir définir une population, des individus et des attributs, il est nécessaire de discrétiser ces dimensions : dans le cas du temps, il peut s'agir, par exemple, de constituer des intervalles dont chacun correspond à une portion du temps. Puisqu'une étape d'abstraction est alors nécessaire pour passer d'un système comme la trace au modèle microscopique, nous souhaitons introduire le concept de *représentation nanoscopique*, qui est la représentation du système à partir de laquelle est générée la représentation microscopique, telle que la trace d'exécution en question. Le processus d'agrégation est alors constitué par trois étapes. Les dimensions continues du modèle nanoscopique sont discrétisées pour produire un nombre fini d'individus. Des attributs sont associés à ces individus par l'intermédiaire d'une opération d'agrégation appliquée sur les données du modèle nanoscopique. Les Chapitres 5 et 6 expliciteront ce processus dans le cas de l'analyse de traces. Enfin, le modèle microscopique est lui-même agrégé selon la *méthode de Lamarche-Perrin*.

4.1.2 Partitionnement et agrégation

On définit un *agrégat* comme une partie de l'ensemble des individus définis par les populations, constituée d'un ou plusieurs individus, dont on agrège les attributs en fonction d'un opérateur d'agrégation mathématique, tel que ceux définis en Section 3.1.1. La *méthode de Lamarche-Perrin* est générique et s'applique en théorie à n'importe quel type

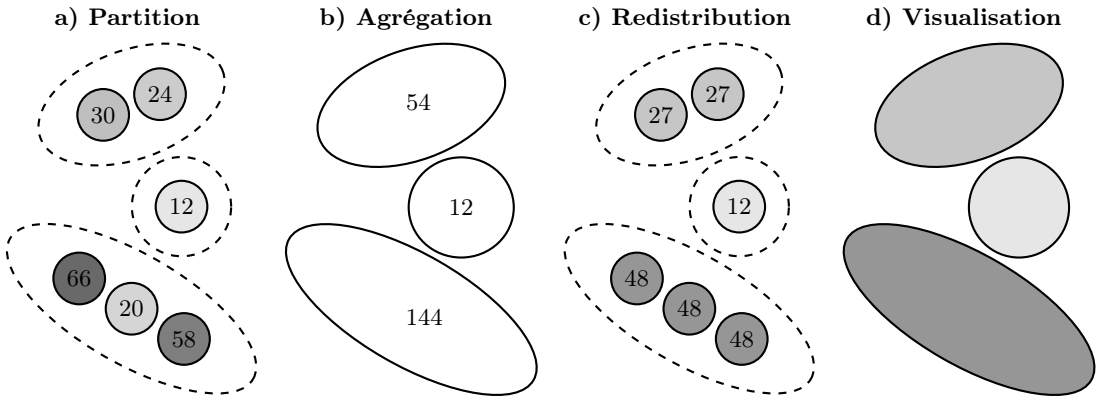


FIGURE 4.1 – Décomposition du processus d'agrégation, inspiré de la Figure 3.1 de la thèse de Lamarche-Perrin [8], à laquelle nous rajoutons l'étape de visualisation.

d'opérateur. Néanmoins, les applications actuelles fondées à partir de cette méthode sont entièrement bâties autour de l'opérateur somme, qui est le plus intuitif pour exprimer l'agrégation d'attributs de dénombrement : $v(X) = \sum_{x \in X} v(x)$ et $p(X) = \frac{v(X)}{v(\Omega)}$ où X est un agrégat. L'emploi d'opérateurs alternatifs nécessite d'adapter l'ensemble des mesures dont dépend le processus d'agrégation (décrites dans la Section 4.2). Cela sera l'objet de travaux futurs.

Une représentation macroscopique du système est définie par un ensemble d'agrégats d'éléments du modèle microscopique, que nous appelons *partition*, tel que les agrégats soient *disjoints* (un élément n'appartient qu'à un et un seul agrégat de la partition) et *couvrants* (tous les éléments du modèle microscopique appartiennent au moins à un agrégat) [8]. On notera $\mathcal{X}$ une telle partition.

Ce procédé réduit la complexité du système en fournissant une représentation dont le nombre d'éléments est plus faible que celui de la description microscopique. Ces données agrégées sont transmises à l'analyste au travers de la visualisation. Pour ce faire, il faut en interpréter les valeurs pour pouvoir retranscrire les phénomènes microscopiques sous-tendus. Pour cela, la *méthode de Lamarche-Perrin* préconise de redistribuer les attributs associés aux agrégats au niveau microscopique [8]. On note $v_{\mathcal{X}}(x)$ la valeur redistribuée associée à x. Dans le cas d'une redistribution uniforme, $v_{\mathcal{X}}(x) = \frac{v(X)}{|X|}$, avec $x \in X$ où X est un agrégat.

La Figure 4.1 récapitule la succession des étapes de partition, agrégation, redistribution, et enfin visualisation du modèle microscopique : a) on partitionne d'abord les individus — la teinte du niveau de gris de chaque individu est liée à la valeur de son attribut — ; b) on agrège les individus à l'intérieur de chaque partie ; c) on redistribue ensuite uniformément la valeur de l'agrégat à chaque individu associé, afin de pouvoir interpréter le résultat du processus d'agrégation — on associe de nouveau à chaque individu une teinte de gris liée à la valeur redistribuée de son attribut — ; d) comme représenter les individus microscopiques séparément est trop complexe, on préfère visualiser les agrégats — la teinte de ces derniers est la même que celle associée aux individus qui les composent, après redistribution.

Certaines propriétés algébriques caractérisent l'ensemble des partitions d'un modèle microscopique [8, 9]. Ainsi, l'ensemble des partitions de Ω, qu'on notera $\boldsymbol{\mathcal{X}}(\Omega)$, est un

treillis, « c'est-à-dire un ensemble *partiellement ordonné* dont chaque couple d'éléments admet une *borne inférieure* et une borne *supérieure* » [8, 89]. Cette relation d'ordre est définie par ce qu'on appelle le raffinement : la partition $\mathcal{X}$ raffine la partition $\mathcal{Y}$ « si et seulement si chaque partie appartenant à $\mathcal{X}$ est incluse dans une partie appartenant à $\mathcal{Y}$ ». Cette propriété permet de déterminer les partitions qui peuvent être atteintes en agrégeant ou en désagrégeant une partition. De même, on définit une relation de couverture : une partition $\mathcal{X}$ est couverte par la partition $\mathcal{Y}$ « si et seulement si $\mathcal{X}$ raffine $\mathcal{Y}$ et s'il n'existe pas de raffinement intermédiaire entre $\mathcal{X}$ et $\mathcal{Y}$ » [8,89]. Ces propriétés sont intéressantes pour plusieurs raisons. Les processus d'agrégation et de désagrégation sont tous les deux convergents, respectivement vers la partition totalement agrégée et vers la partition désagrégée (qu'on nomme partition microscopique) ; combiné à la relation d'ordre, cela assure la cohérence de l'analyse lorsqu'on passe d'une partition à une autre. Dans la Section 4.2, nous verrons aussi l'intérêt que représente la relation de raffinement dans le calcul des mesures de qualités associées aux partitions.

4.2 Problème des partitions optimales et mesures de qualité

Le choix d'une partition n'est pas fait de manière arbitraire. Il doit être réalisé avec l'objectif de répondre à deux des problématiques générales que nous avons définies dans l'introduction, à savoir le passage à l'échelle de la représentation et son interprétabilité. Afin de guider la partition et l'agrégation pour satisfaire au mieux ces contraintes, Lamarche-Perrin articule sa méthode autour de deux principes : la réduction de complexité et la perte d'information.

Selon lui, « la *complexité* d'une représentation désigne la difficulté qu'a un observateur à analyser le système à partir de cette représentation. La complexité est alors caractérisée par le « coût » d'une telle analyse, c'est-à-dire la *quantité de ressources nécessaires à l'exploitation de la représentation* » [8]. Les données contenues dans le modèle analysé fournissent de l'*information*, c'est-à-dire « un fait significatif servant à décrire et à expliquer l'état ou la dynamique d'un système (*e.g.*, un évènement, une perturbation, une transition) » [8]. La quantité d'information contenue dans une représentation dépend donc de la valeur des données plus que de la quantité de données elle-même. On distingue ainsi le cas de données dites *redondantes*, le phénomène qu'elles sous-tendent pouvant être exprimé avec une quantité de données plus faible, et le cas où le phénomène représenté ne peut pas être exprimé avec moins de données. Lors de l'agrégation, la perte d'information est faible dans le premier cas, forte dans le second.

La problématique initiale, à savoir assurer une représentation passant à l'échelle tout en assurant son interprétabilité, peut ainsi se reformuler comme la recherche d'un compromis entre la réduction de sa complexité et de la perte de l'information provoqué par le processus d'agrégation. Cette démarche est dynamique et guidée par le contexte et les besoins de l'analyse, comme la nature et la taille du système étudié, le type de comportements à analyser, ou encore l'abord de l'analyse (un aperçu général du comportement du système ou une analyse plus détaillée).

4.2.1 Mesures de qualité

Les mesures de qualité établies par Lamarche-Perrin sont un moyen de quantifier la réduction de complexité et la perte d'information. Elles sont définies, dans un cadre plus générique, comme des mesures associées aux partitions et exprimant leur qualité face à un critère particulier, lequel est positif (c'est-à-dire à maximiser, telle que la réduction de complexité) ou négatif (à minimiser, telle que la perte d'information). Les *partitions optimales* sont les partitions qui maximisent une mesure de qualité positive donnée (ou minimisent une mesure négative), et sont donc considérées comme les meilleures représentations du système selon ce critère. Le *problème des partitions optimales* consiste à trouver ces partitions [8].

Propriétés

Afin d'assurer une cohérence avec le processus d'agrégation, plusieurs propriétés doivent être vérifiées par les mesures de qualités, en particulier, la *monotonie* et la *décomposabilité*. Cette dernière garantit que « la qualité d'une partition est entièrement déterminée par la qualité de ses parties » [8] ; les parties possèdent ainsi elles-mêmes des mesures de qualités. La mesure de qualité d'une partition (ou d'un ensemble de parties) est dite *additive* si elle est la somme de la mesure de qualité de ses constituants.

Mesures de complexité

Complexité L'entropie de Shannon [90], issue de la théorie de l'information, est employée par Lamarche-Perrin *et al.* [88, 91] pour décrire la complexité d'un système. Elle mesure la quantité d'information nécessaire pour encoder un message ou une description (texte, signal ou encore fichier informatique). D'un point de vue probabiliste, l'entropie de Shannon est une mesure de désordre. Elle exprime l'incertitude qu'un observateur a sur les éléments d'une description dont il ne connaît que la probabilité d'apparaître des différents symboles qui leur sont associés. Ainsi, l'entropie croît avec le nombre de symboles possibles. Si la distribution des symboles suit une loi équiprobable, alors l'entropie est maximale, puisque l'incertitude sur la valeur de ces éléments est maximale. À contrario, une description où il n'existe qu'un seul symbole possible possède une entropie nulle : il n'existe aucune incertitude sur la valeur des éléments de la description. L'entropie de Shannon satisfait les propriétés énoncées Section 4.2.1, à savoir la *monotonie*, et l'*additivité* [92]. L'entropie de Shannon d'une partie $X \in \mathcal{X}$, en bits/unité, est donnée par la formule suivante :

$$\mathrm{info}(X) = \sum_{x \in X} p(x) \log_2 p(x) \tag{4.1}$$

et celle d'une partition $\mathcal{X} \in \boldsymbol{\mathcal{X}}(\Omega)$ est calculée grâce à ses propriétés d'additivité :

$$\mathrm{info}(\mathcal{X}) = \sum_{X \in \mathcal{X}} \mathrm{info}(X) \tag{4.2}$$

Réduction de complexité De manière générale, la réduction de complexité peut être exprimée comme la différence entre la complexité de la description microscopique et de la représentation agrégée. Dans le cas où l'on mesure la complexité avec l'entropie de

Shannon, cette différence correspond à la quantité d'information que permet de sauvegarder l'encodage de la représentation agrégée par rapport à l'encodage de la représentation microscopique [88]. Plus la représentation est agrégée, plus la quantité d'information sauvegardée est grande. Mécaniquement, diminuer la quantité d'information utilisée pour encoder la représentation se traduit par une perte en précision, ce qui diminue la quantité d'information qu'elle fournit. Ainsi, dans le cas d'une agrégation totale, toute l'information sur la distribution du modèle microscopique est perdue. La réduction de complexité définie en tant que différence d'entropies de Shannon est, de par sa construction, elle-même monotone et additive. La réduction de complexité d'une partie $X \in \mathcal{X}$, en bits/unité, est donnée par la formule suivante :

$$\text{gain}(X) = \sum_{x \in X} \text{info}(x) \quad \text{info}(X) \tag{4.3}$$

et celle d'une partition $\mathcal{X} \in \boldsymbol{\mathcal{X}}(\Omega)$ est calculée grâce à ses propriétés d'additivité :

$$\text{gain}(\mathcal{X}) = \sum_{X \in \mathcal{X}} \text{gain}(X) \tag{4.4}$$

Mesures de perte d'information

Comme pour la complexité, la théorie de l'information fournit des mesures qui se prêtent bien à l'expression de la perte d'information. Parmi elle, la divergence de Kullback-Leibler [93], qui est une mesure de dissimilarité asymétrique permettant de comparer deux distributions de probabilités, en général une source et un modèle. Appliquée aux représentations microscopiques (la source) et agrégées (le modèle), elle s'interprète comme le nombre de bits supplémentaires moyen nécessaire pour déterminer l'individu du modèle microscopique associé à un échantillon, tiré aléatoirement et de manière uniforme, de la représentation redistribuée générée à partir de la description macroscopique [8]. De manière plus conceptuelle, la divergence de Kullback-Leibler exprime la quantité d'information supplémentaire à apporter pour reconstituer le modèle microscopique à partir de la représentation agrégée et de la quantité d'information qui n'est pas transmise lors du processus d'agrégation (connue sous le nom d'entropie conditionnelle [94]). La divergence de Kullback-Leibler est *monotone* [95] et *additive* [88]. La divergence d'une partie $X \in \mathcal{X}$, en bits/unité est donnée par la formule suivante (dans le cas d'une redistribution uniforme) :

$$\text{loss}(X) = \sum_{x \in X} p(x) \log_2 \left(\frac{p(x)}{p(X)} \times |X| \right) \tag{4.5}$$

et celle d'une partition $\mathcal{X} \in \boldsymbol{\mathcal{X}}(\Omega)$ est calculée grâce à ses propriétés d'additivité :

$$\text{loss}(\mathcal{X}) = \sum_{X \in \mathcal{X}} \text{loss}(X) \tag{4.6}$$

Compromis entre réduction de complexité et perte d'information

Lamarche-Perrin propose enfin un troisième type de mesures de qualités, un compromis entre la réduction de complexité et la perte d'information [8], nommé pIC, pour *parametrized Information Criterion*. Il est formulé sous la forme d'une combinaison linéaire des

deux mesures de complexité, normalisées pour assurer leur homogénéité, tel qu'il suit :

$$\mathrm{pIC}(\mathcal{X}) = p \times \frac{\mathrm{gain}(\mathcal{X})}{\mathrm{gain}_{max}(\boldsymbol{\mathcal{X}}(\Omega))} \quad (1 \quad p) \times \frac{\mathrm{loss}(\mathcal{X})}{\mathrm{loss}_{max}(\boldsymbol{\mathcal{X}}(\Omega))} \tag{4.7}$$

où $\mathrm{gain}_{max}(\boldsymbol{\mathcal{X}}(\Omega))$ et $\mathrm{loss}_{max}(\boldsymbol{\mathcal{X}}(\Omega))$ sont la réduction de complexité et la perte d'information maximales parmi toutes les réductions de complexité et pertes d'information des partitions de $\boldsymbol{\mathcal{X}}(\Omega)$, et $p \in [0, 1]$ est un coefficient de compromis influant sur la pondération.

Dans le cas particulier où l'on emploie l'entropie de Shannon comme mesure de réduction de complexité et la divergence de Kullback-Leibler comme mesure de perte d'information, le gain et la perte sont homogènes (même unité), et il n'est pas nécessaire de normaliser :

$$\mathrm{pIC}(\mathcal{X}) = p \times \mathrm{gain}(\mathcal{X}) \quad (1 \quad p) \times \mathrm{loss}(\mathcal{X}) \tag{4.8}$$

Le pIC est une mesure de qualité positive, qui se maximise, pour un coefficient p donné, en maximisant la réduction de complexité et en minimisant la perte d'information. L'utilisateur fait varier p de manière à influer sur la pondération des deux mesures, et ainsi ajuster le niveau de détail de la représentation en fonction des objectifs de l'analyse :

- si $p = 0$, la réduction de complexité est ignorée, et la partition optimale est celle qui minimise la perte d'information, à savoir la partition microscopique ;
- si $p = 1$, la perte d'information est ignorée, et la partition optimale est celle qui maximise la réduction de complexité, c'est-à-dire la partition composée d'un unique agrégat ;
- si $0 < p < 1$, les deux mesures sont prises en compte. Une valeur de p faible fournira des partitions minimisant la perte d'information, c'est-à-dire détaillées. Au fur et à mesure que p croît, maximiser la réduction de complexité devient prépondérant, ce qui se traduit par des partitions contenant moins de parties et plus simples à analyser.

4.3 Partitions admissibles

Certaines partitions de $\boldsymbol{\mathcal{X}}(\Omega)$ ne sont pas compatibles avec les caractéristiques syntaxiques et sémantiques de la population Ω, et leur représentation n'est pas exploitable dans le cadre de l'analyse. Lamarche-Perrin *et al.* donnent l'exemple d'un espace géographique où l'on cherche à analyser certaines variables démographiques [8, 83]. Certains phénomènes peuvent être reliés à des caractéristiques de cet espace, connus de l'observateur (facteurs politiques, économiques, culturels, etc.). Agréger sans tenir compte de ces paramètres peut fournir des représentations optimales vis à vis des mesures de qualité, mais non pertinentes au regard de l'analyse.

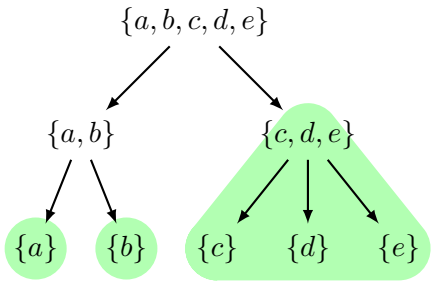
De plus, la résolution générale du problème des partitions optimales est de complexité exponentielle quand elle est appliquée à l'ensemble des partitions $\boldsymbol{\mathcal{X}}(\Omega)$ [8]. L'intérêt de la méthode étant de proposer une représentation macroscopique d'un système composé de nombreuses entités, le passage à l'échelle de l'algorithme résolvant le problème des partitions optimales est une condition obligatoire.

La solution à ces deux problématiques est de réduire l'espace des solutions à un sous-ensemble de $\boldsymbol{\mathcal{X}}(\Omega)$, appelé ensemble des partitions admissibles [8, 9], que nous noterons

$\boldsymbol{\mathcal{X}}_a(\Omega)$. Ces partitions sont choisies de manière à représenter des abstractions cohérentes du système au vu des connaissances que l'analyste possède sur ce dernier.

Dans le cas de Viva [5,6], par exemple, l'analyse est axée sur l'influence de la structure de la plateforme d'une application parallèle. Il est proposé de la structurer sous la forme d'un arbre enraciné représentant la hiérarchie matérielle et logicielle, et de contraindre les partitions de manière à ce que les individus soient agrégés par branches entières [8,9,83]. La Figure 4.2 donne des exemples de partitions admissibles et non admissibles selon cette contrainte, et la Figure 4.3 montre le parallèle avec une représentation treemap hiérarchique. Le diagramme de Hasse du treillis des partitions de la hiérarchie de ces deux Figures est décrit par la Figure 4.4 a).

a) Parties admissibles



b) Parties non admissibles

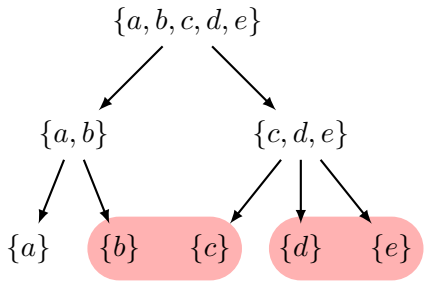
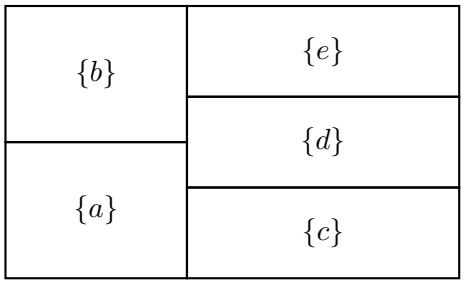


FIGURE 4.2 – Reproduction de la Figure 5.2 de la thèse de Lamarche-Perrin [8], montrant une hiérarchie, sous la forme d'un arbre enraciné, à 3 niveaux : a) exemples de parties admissibles, composant une partition $\{a\},\{b\},\{c,d,e\}$, b) exemples de parties non admissibles, telles qu'une partie partiellement superposée sur deux branches ($\{b,c\}$), ou ne s'étendant pas complètement sur une branche ($\{d,e\}$).

a) Partition microscopique



b) Partition de la Figure 4.2 a)

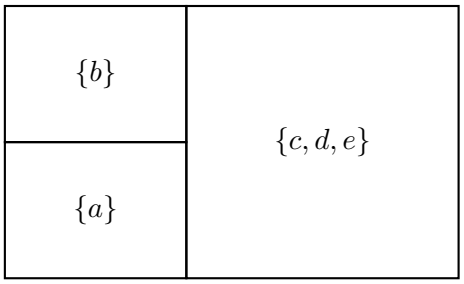


FIGURE 4.3 – Exemple de représentations treemap hiérarchiques multirésolutions fondées sur la *méthode Lamarche-Perrin*, telles que proposées par l'outil Viva [5,6] : a) partition microscopique de la hiérarchie de la Figure 4.2, b) partition correspondant à celle de la Figure 4.2 a).

Une organisation ordonnée du système convient bien aux représentations temporelles [8,9] : nous considérons que l'agrégation doit être limitée aux intervalles de temps contigus. Ici, il existe un ordre total $<$ sur Ω, et l'ensemble des parties admissibles est constitué des intervalles d'éléments $[x,y]$ définis dans Ω par la relation d'ordre [8]. L'en-

a) Population hiérarchique

b) Population ordonnée

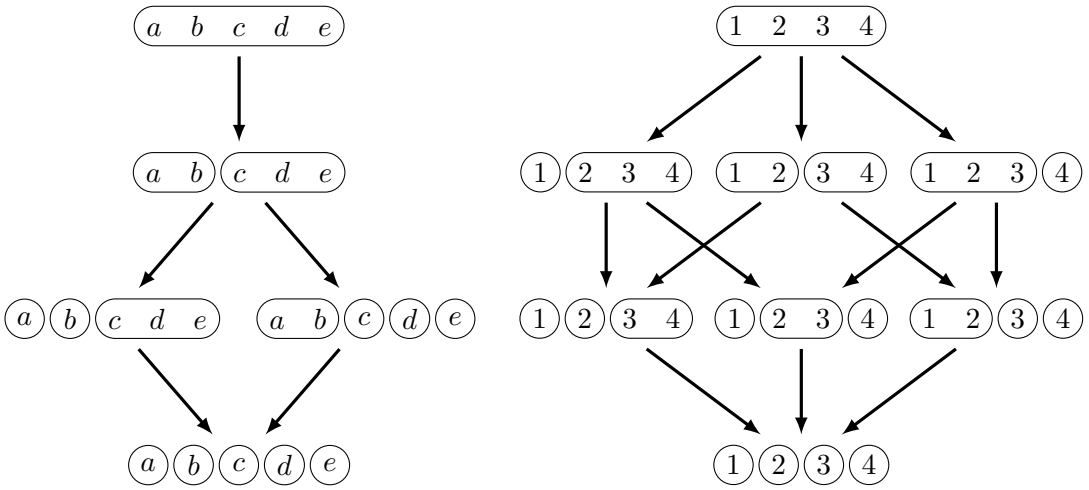


FIGURE 4.4 – Diagramme de Hasse représentant le treillis de l'ensemble des partitions admissibles ordonné d'une population hiérarchique et d'une population ordonnée (Figure 5.4 de la thèse de Lamarche-Perrin [8]). La relation de couverture et la décomposabilité des mesures de qualité permettent de simplifier le calcul des partitions optimales.

semble des partitions admissibles, dont le treillis est montré par la Figure 4.4 b), est alors composé des partitions construites à partir de ces intervalles. La Figure 4.5 représente un exemple d'application de cette technique à l'analyse de l'évolution temporelle de l'attention d'un journal à la Grèce, construite à partir des citations. L'agrégation b) permet d'identifier plus facilement des phénomènes macroscopiques comme la chute de l'attention au cours du temps, cependant entrecoupée par des pics relatifs à des évènements ayant un impact sur le plan international.

a) Partition microscopique

b) Partition préservant au moins 50% de l'information

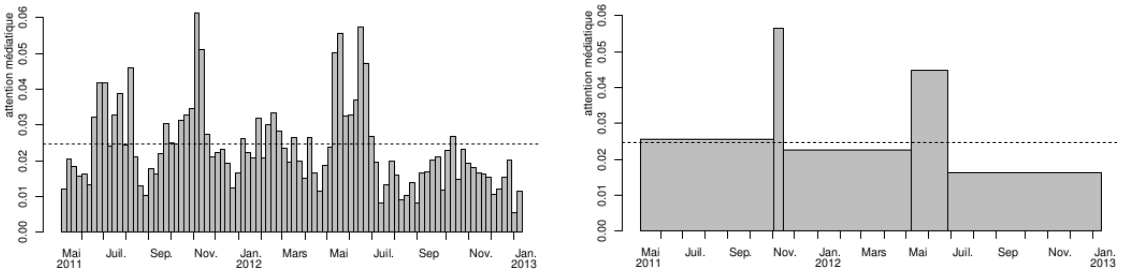


FIGURE 4.5 – Exemple d'application de la *méthode Lamarche-Perrin* à une population ordonnée : variation temporelle de l'attention médiatique du journal *Guardian* au sujet de la Grèce (tiré des Figures 8.11 et 8.13 de sa thèse [8]).

4.4 Construction des algorithmes d'agrégation

La dernière étape de la *méthode Lamarche-Perrin* consiste à construire un algorithme de résolution du problème des partitions admissibles optimales qui soit de complexité polynomiale afin de le rendre utilisable en pratique. Différents travaux [8,9,83] proposent ainsi des algorithmes de types *diviser pour régner*, en utilisant la relation de couverture définie par le treillis des partitions admissibles optimales pour décomposer le problème en sous-problèmes, et en profitant des propriétés de décomposabilité des mesures de qualité, qui permettent d'appliquer l'algorithme sur ces sous-problèmes. L'algorithme est exécuté récursivement sur chaque sous-problème jusqu'au cas trivial. Une description étendue de l'algorithme des partitions ordonnées optimales (version optimisée) est donnée Section 5.5. L'algorithme des partitions hiérarchiques optimales est détaillé dans la thèse de Lamarche-Perrin [8].

CHAPITRE 5

Agrégation temporelle de traces d'exécution

Les traces collectées lors de l'exécution d'un programme informatique, évoquées Section 2.1, correspondent à un cas particulier des *systèmes multi-agents* décrits par Lamarche-Perrin. Sa méthode d'agrégation, décrite dans le chapitre précédent, trouve ici une application intéressante, puisque la notion d'homogénéité peut nous permettre de déceler des comportements problématiques.

Dans ce chapitre, nous nous intéressons à une analyse du comportement temporel d'applications multimédia embarquées, mais aussi d'applications de calcul parallèle. Pour cela, nous employons l'algorithme d'agrégation d'un ensemble ordonné de Lamarche-Perrin référencé Section 4.4. En effet, la représentation de l'homogénéité du comportement de l'application grâce aux partitions se prête bien à l'analyse des applications multimédia temps-réel embarquées dont le comportement est composé de cycles de traitements pipelinés, et dont la durée doit être la plus régulière possible. Des défauts dans la régularité de ces cycles se matérialisent par des perturbations dans le flux vidéo (ralentissement, perte d'images), ce qui peut être décelé par ce type d'analyse. De même, certaines applications parallèles de calcul scientifique, comme le calcul matriciel, se découpent en différentes phases comportementales, à l'intérieur desquelles le comportement temporel est censé être homogène. Notre technique d'analyse permet, d'une part, de retrouver ces phases, et d'autre part, d'appréhender l'homogénéité de chacune d'elles, tout en garantissant le passage à l'échelle dans le cas de traces volumineuses.

Comme la trace brute (le modèle nanoscopique) possède une dimension temporelle continue, il est nécessaire de la discrétiser pour s'intégrer dans la *méthode de Lamarche-Perrin*. Pour cela, nous détaillons la construction d'un modèle microscopique bâti à l'aide d'une discrétisation du temps et d'une technique d'agrégation produisant des valeurs décrivant le comportement que l'on souhaite analyser. Ceci correspond à la première étape du flot d'agrégation. Nous appliquons ensuite l'algorithme de partition d'un ensemble ordonné, qui détermine comment agréger les éléments temporels du modèle microscopique de manière optimale. Nous étendons cet algorithme pour des modèles microscopiques faisant intervenir d'autres dimensions que le temps. Enfin, nous concevons une technique de visualisation qui représente le résultat fourni par l'algorithme de partition. Nous en

proposons plusieurs, fondées sur des histogrammes. Ces techniques de visualisation impliquent potentiellement d'autres types d'agrégations (de données, sur la dimension spatiale, visuelles) dans le but de fournir une représentation lisible véhiculant le maximum d'information.

La technique d'agrégation temporelle décrite dans ce chapitre a donné lieu à la publications de plusieurs articles scientifiques [11, 15, 17] et rapports de recherche [12, 16]. Ces travaux sont issus, en partie, de la collaboration avec Robin Lamarche-Perrin, chercheur à l'Institut Max Planck de Leipzig, concepteur de la méthodologie d'analyse et de la technique d'agrégation temporelle originale (voir Chapitre 4), dont j'ai repris certains éléments de formalisme ici, et Lucas Mello Schnorr, professeur adjoint à l'Université Fédérale de Rio Grande do Sul, Porto Alegre, co-auteur d'un rapport de recherche [16], et qui a réalisé des expérimentations sur la plateforme de calcul GridRS[1][2].

5.1 Structure des modèles microscopiques temporels

Nous présentons dans cette section les différents éléments composant la structure des modèles microscopiques générés à partir de la trace d'exécution et destinés à être agrégés temporellement. Celle-ci reflète en partie l'organisation de la trace, décrite Section 2.1, à savoir la présence des dimensions temporelle, spatiale, et des types d'évènements.

5.1.1 Dimension temporelle discrète

Tous les modèles microscopiques que nous proposons ont la particularité d'être construits autour de la dimension temporelle. La dimension temporelle contenue dans la trace est la dimension continue T. Il est nécessaire d'appliquer une opération de discrétisation afin que le modèle microscopique soit bâti autour d'une dimension temporelle discrète que l'on pourra agréger par la suite.

Définition 5.1 $T = \{t_1, \ldots, t_n\}$ est l'ensemble des éléments t définissant la dimension temporelle discrète et ordonnée d'un modèle microscopique. On note $|T|$ le nombre d'éléments de T. Chaque période t, que l'on nomme aussi *tranche de temps*, a une durée $d(t) \in \mathbb{R}^+$ tel que $d(t) = \frac{\tau_{fin} \quad \tau_{debut}}{|T|}$ et l'ensemble est ordonné selon l'axe temporel. Le concept d'intervalle est définit comme il suit : $T_{(i,j)} = \{t \in T \mid t_i \leq t \leq t_j\}$ avec $t_i \leq t_j$.

Définition 5.2 Soit un temps $\tau \in \mathsf{T}$. On note $\tau \in t_i$ avec t_i i-ème élément de T le respect de l'inégalité suivante : $\tau_{debut} + (i \quad 1).d(t) \leq \tau < \tau_{debut} + i.d(t)$. On note $\tau_d(t_i) = \tau_{debut} + (i \quad 1).d(t)$ et $\tau_f(t_i) = \tau_{debut} + i.d(t)$.

[1] http://gridrs.lad.pucrs.br/

[2] Qui ne figurent pas dans cette thèse, mais ont permis l'adaptation de la méthode à l'analyse de trace issues de systèmes parallèles.

Définition 5.3 $\mathcal{I}(T)$ est l'ensemble des intervalles de T.
$\mathcal{I}(T) = \{T_{(1)}, \ldots, T_{(1,m)}, T_{(2)}, \ldots, T_{(2,m)}, \ldots, T_{(m)}\}$

5.1.2 Définition du modèle microscopique

Le modèle microscopique associe une valeur à des coordonnées de l'espace-temps-type d'évènements selon une métrique capable de représenter l'état du système ou les phénomènes s'y produisant, sur cette « zone ».

Définition 5.4 Le modèle microscopique temporel tridimensionnel en fonction des ressources, d'un temps discret et des types d'évènements est une application de $S \times T \times \mathcal{J}(E)$ dans $\mathbb{R}^+$, que l'on note $M^{\mu/t3}$. L'image de (s_k, t_i, J_l) par l'application $M^{\mu/t3}$, avec $s_k \in S$, $t_i \in T$ et $J_l \subset \mathcal{J}(E)$ est noté $\mu^{t3}(s_k, t_i, J_l)$.

L'attribut associé à une tranche de temps t_i est la matrice de dimension $(S, \mathcal{J}(E))$ $\mu^{t3}(t_i)$:

$$\mu^{t3}(t_i) = \begin{pmatrix} \mu^{t3}(s_1, t_i, J_1) & \mu^{t3}(s_1, t_i, J_2) & \cdots & \mu^{t3}(s_1, t_i, J_m) \\ \mu^{t3}(s_2, t_i, J_1) & \mu^{t3}(s_2, t_i, J_2) & \cdots & \mu^{t3}(s_2, t_i, J_m) \\ \vdots & \vdots & \cdots & \vdots \\ \mu^{t3}(s_n, t_i, J_1) & \mu^{t3}(s_n, t_i, J_2) & \cdots & \mu^{t3}(s_n, t_i, J_m) \end{pmatrix} \tag{5.1}$$

À partir de cette définition, il est possible d'exprimer des modèles microscopiques mono ou bidimensionnels, où la dimension spatiale ou de types d'évènements est un singleton.

5.2 Métriques des modèles microscopiques temporels

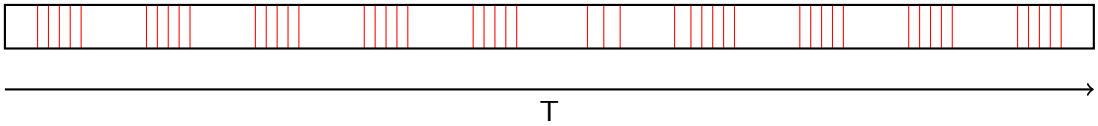
Il est possible d'associer plusieurs types de métrique différents à l'image $\mu^{t3}(s_k, t_i, J_l)$. Ces métriques expriment un comportement particulier de l'application à partir des données bas niveaux contenues dans le modèle nanoscopique. Il y a une relation entre la nature des évènements que l'on souhaite analyser et la métrique que l'on choisit d'associer aux coefficients du modèle microscopique. Selon que l'on se focalise sur les évènements, sans distinction de leur catégorie, ou que l'on s'attache plus spécifiquement à l'analyse des *états* ou des *variables*, les métriques employées diffèrent.

5.2.1 Analyse des évènements

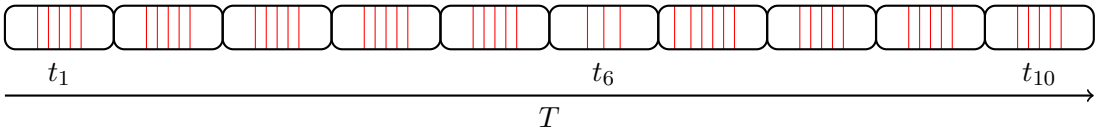
L'observation de l'évolution de la quantité d'évènements au cours du temps est une technique d'analyse classique. En effet, certaines perturbations peuvent induire un blocage dans l'application qui se traduisent par une diminution brutale du nombre d'évènements à ce moment. Pour l'adapter à un modèle microscopique temporel, on associe la quantité d'évènements de type $J_l \in \mathcal{J}(E)$ produite par la ressource $s_k \in S$ pour une tranche de temps $t_i \in T$ à $\mu^{t3}(s_k, t_i, J_l)$. Les Figures 5.1, 5.2, 5.3 illustrent la génération de modèles microscopiques de différentes dimensions, représentant la quantité d'évènements en fonction du temps. Dans chacun de ces modèles microscopiques, la tranche de temps

$T_{(6)}$ a une valeur, scalaire ou vectorielle, différente des autres, illustrant une variation du nombre d'évènements (Figure 5.1), de leur répartition au sein des ressources (Figure 5.2), ou un changement de leur type (Figure 5.3) dans cette partie de la trace, et donc potentiellement un changement de comportement. C'est un exemple d'irrégularité que notre technique d'agrégation mettra en évidence pour des modèles plus complexes.

Modèle nanoscopique :



Discrétisation :



Modèle microscopique :

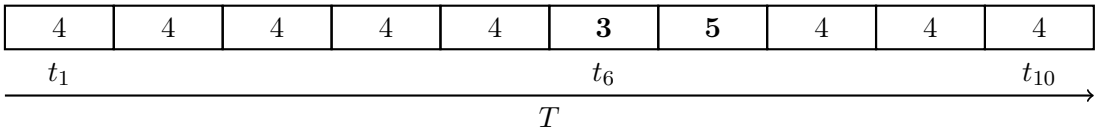


FIGURE 5.1 – Exemple de génération d'un modèle microscopique monodimensionnel représentant la quantité d'évènements en fonction d'un temps discret.

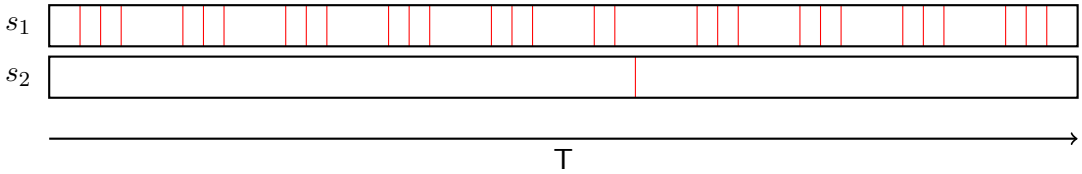
5.2.2 Analyse des états

Un certain nombre d'informations issues de la trace font appel à des notions de durée, représentées par des états. C'est le cas, par exemple, des appels de fonctions, qui sont bornés par une date d'entrée dans la fonction et par une date de sortie, des interruptions, mais, de manière générale, d'un traitement de durée finie ou d'un état du système. C'est ce type d'informations que les diagrammes espace-temps mettent principalement en évidence. Les métriques employées pour analyser les états permettent ainsi de mettre en relief un changement dans leur durée au cours du temps.

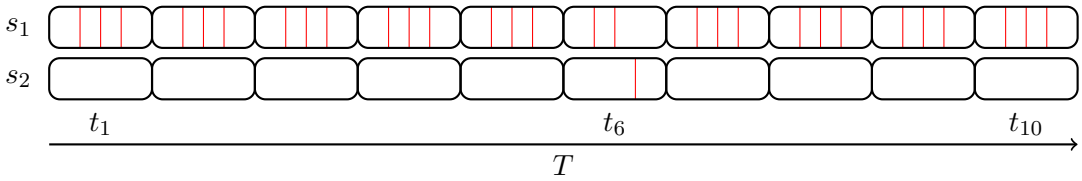
Par exemple, la *quantité de temps passée dans chaque type d'états* est obtenue en additionnant la durée de chaque état de type $J_l \in \mathcal{J}(E)$, de la ressource $s_k \in S$ dans la tranche de temps $t_i \in T$ au sein de $\mu^{t3}(s_k, t_i, J_l)$. *La quantité de temps moyen par type d'états* est calculée en moyennant leur durée sur chaque tranche de temps par le nombre d'états actifs sur cette tranche de temps. Elle est utile pour discriminer des états anormalement longs ou courts. D'autres types d'opérateurs d'agrégation sont envisageables (minimum, maximum, médiane, etc.), selon le comportement que l'on cherche à mettre en évidence.

La Figure 5.4 illustre la génération du modèle microscopique tridimensionnel représentant les quantités de temps passé dans un état en fonction d'un temps discret, des ressources et des types d'évènements. Les valeurs associées aux tranches de temps $T_{(4)}$

Modèle nanoscopique :



Discrétisation :



Modèle microscopique :

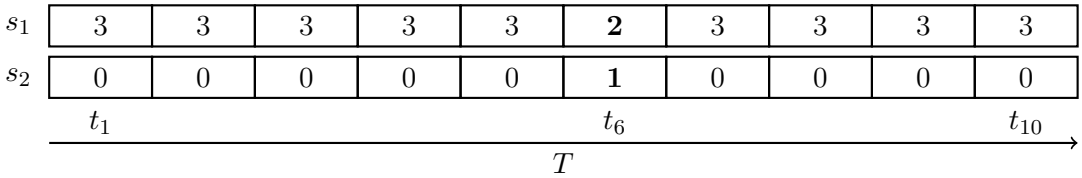


FIGURE 5.2 – Exemple de génération d'un modèle microscopique bidimensionnel représentant la quantité d'évènements en fonction d'un temps discret et des ressources.

à $T_{(7)}$ matérialisent un changement de comportement dans la trace (présence dun état vert et d'un état bleu).

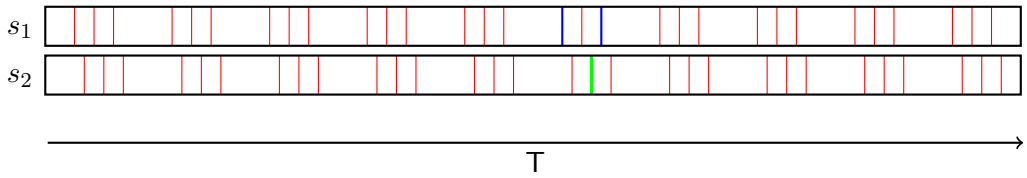
5.2.3 Analyse des variables

L'analyse des variables est fondée sur un principe similaire aux métriques précédentes, à savoir agréger leur valeur au sein de $\mu^{t3}(s_k, t_i, J_l)$, selon un opérateur particulier (moyenne, minimum, maximum, etc.).

5.3 Granularité du modèle microscopique et effets de la discrétisation

Les différents exemples illustrant les modèles microscopiques que nous avons présentés sont de taille très réduite (5 à 10 tranches de temps, 2 ressources, 3 types d'évènements, et quelques dizaines d'évènements maximum). En ce qui concerne la dimension temporelle, nous avons adapté la discrétisation à la période des motifs que constituent les évènements afin de mettre en évidence l'influence d'une rupture comportementale sur les données du modèle microscopique. Comme le montre la Figure 5.5, un redimensionnement de la dimension temporelle dans ce genre d'exemples peut fausser l'analyse à cause d'une répartition différente des évènements dans les tranches de temps. À gauche, la discrétisation à 5 tranches de temps coïncide avec la période des évènements. À droite,

Modèle nanoscopique :



Modèle microscopique :

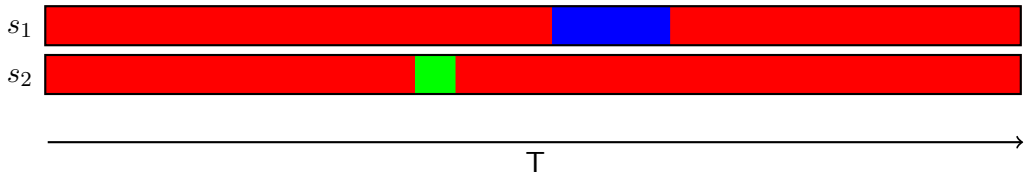
(s_1, J_R)	3	3	3	3	3	**1**	3	3	3	3
(s_1, J_G)	0	0	0	0	0	0	0	0	0	0
(s_1, J_B)	0	0	0	0	0	**2**	0	0	0	0
(s_2, J_R)	3	3	3	3	3	**2**	3	3	3	3
(s_2, J_G)	0	0	0	0	0	**1**	0	0	0	0
(s_2, J_B)	0	0	0	0	0	0	0	0	0	0

t_1 t_6 t_{10}

T

FIGURE 5.3 – Exemple de génération d'un modèle microscopique tridimensionnel représentant la quantité d'évènements en fonction d'un temps discret, des ressources et des types d'évènements.

Modèle nanoscopique :



Modèle microscopique :

(s_1, J_R)	5	5	5	5	5	**1**	**3**	5	5	5
(s_1, J_G)	0	0	0	0	0	0	0	0	0	0
(s_1, J_B)	0	0	0	0	0	**4**	**2**	0	0	0
(s_2, J_R)	5	5	5	**4**	**4**	5	5	5	5	5
(s_2, J_G)	0	0	0	**1**	**1**	0	0	0	0	0
(s_2, J_B)	0	0	0	0	0	0	0	0	0	0

t_1 t_6 t_{10}

T

FIGURE 5.4 – Exemple de génération d'un modèle microscopique tridimensionnel représentant les quantités de temps passé dans un état en fonction d'un temps discret, des ressources et des types d'évènements.

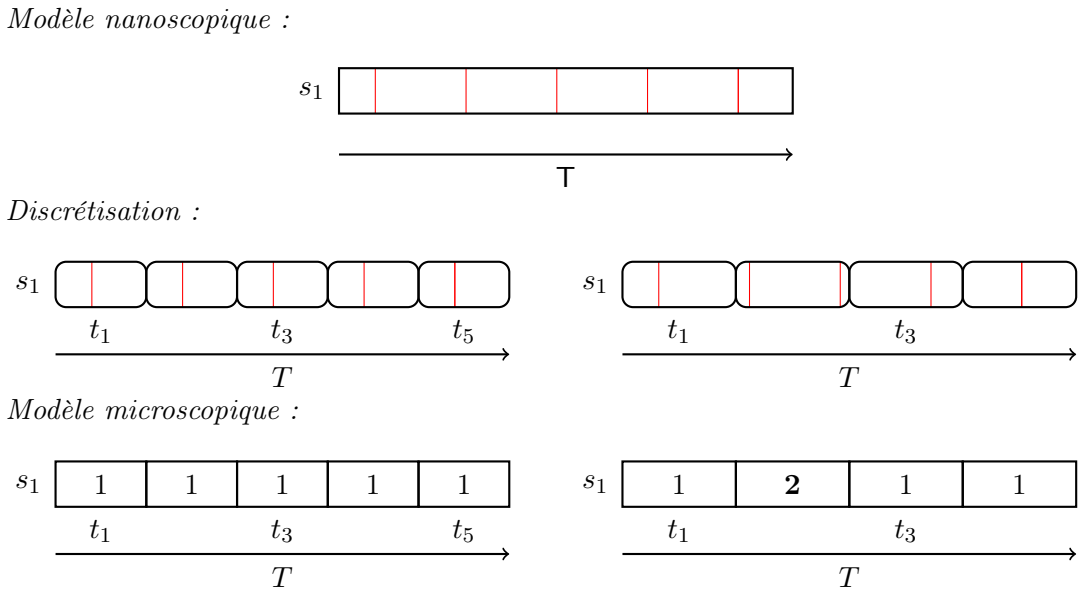


FIGURE 5.5 – Exemple d'un artefact dû à la discrétisation dans le cas d'un système simple comportant peu d'évènements.

en diminuant le nombre de tranches de temps à 4, la valeur du modèle microscopique associée à t_2 double : l'homogénéité du comportement de la trace, pourtant bien réelle, n'est plus correctement retranscrite.

De manière plus réaliste, l'analyse de trace fait intervenir des quantités d'évènements de plusieurs millions à plusieurs milliards. Les capacités de l'outil d'analyse (taille de l'écran, performances CPU, mémoire) quant à représenter et agréger des tranches de temps permettent d'atteindre plusieurs milliers de tranches de temps. Une tranche de temps contient donc, en principe, plusieurs milliers à plusieurs millions d'évènements agrégés, et la granularité d'un comportement périodique de l'application est bien en deçà de la durée d'une tranche de temps. Au vu des ordres de grandeur des paramètres impliqués, nous estimons donc que le modèle microscopique est peu sensible à une légère variation de ses paramètres.

Néanmoins, la granularité du modèle microscopique possède une influence sur l'analyse. Elle détermine, d'une part, sa précision, c'est-à-dire sa capacité à borner un phénomène temporel de durée finie contenu dans le modèle nanoscopique avec le moins d'erreur possible. En considérant l'imprécision sur la première et sur la dernière tranche de temps bornant un phénomène de ce type, l'imprécision totale est donc la durée de deux tranches de temps. Ainsi, plus la durée associée à tranche de temps décroît, plus cette précision augmente. La granularité détermine aussi la discriminabilité du modèle microscopique, c'est-à-dire sa capacité à retranscrire un phénomène particulier sans souffrir d'effets de compensation. En effet, plus la granularité est grosse, plus il y a risque qu'un comportement soit atténué par l'agrégation à l'intérieur des tranches de temps. Cet effet est dépendant du type d'opérateur utilisé pour générer le modèle microscopique. La taille des tranches de temps a donc, là encore, une influence importante.

Les limites se situant, en pratique, au niveau des capacités de l'algorithme d'agrégation (vitesse de calcul, mémoire) et de la visualisation (taille minimale affichable d'une

tranche de temps), le nombre de tranches de temps devrait être la valeur maximale permise par ces différents critères. Avec notre implémentation, pour une résolution d'écran standard de 1920×1080 pixels, nous estimons qu'il est préférable de ne pas dépasser 500 tranches de temps pour ne pas générer d'artefacts lors du rendu.

5.4 Modèle d'agrégation temporelle

Nous avons évoqué, Section 4.4, un algorithme d'agrégation d'un ensemble ordonné proposé par Lamarche-Perrin, qui est une application de sa méthodologie. Le modèle microscopique fourni en entrée est un ensemble ordonné comparable aux modèles microscopiques monodimensionnels temporels $M^{\mu/t1}$ que nous proposons. Il est donc possible de réutiliser intégralement cet algorithme pour ces modèles monodimensionnels. Pour les modèles multidimensionnels, une adaptation de cet algorithme est toutefois nécessaire, et sera détaillée dans la Section suivante. Nous caractérisons le processus d'agrégation de nos modèles microscopiques, en utilisant les critères définis dans la Section 3.1.2.

Note *Dans cette section et la suivante, les exemples qui illustrent nos propos seront appliqués sur des modèles microscopiques génériques (la sémantique de leurs données n'est pas définie). De plus, nous ne présenterons que les calculs et algorithmes appliqués à un modèle microscopique tridimensionnel. Pour les appliquer sur les cas mono ou bidimensionnels, il suffira de considérer que* $|S| = 1$ *ou* $|\mathcal{J}(E)| = 1$.

Dimensions d'agrégation

La dimension d'agrégation est la dimension temporelle discrète T. Il est important de noter que ni les ressources S, ni les types d'évènements $\mathcal{J}(E)$ ne sont agrégés par l'algorithme des partitions ordonnées optimales (mais ils pourront être agrégés après, lors de la visualisation).

Opérandes

Les opérandes sont l'ensemble des tranches de temps $t \in T$.

Contraintes sur l'agrégation

L'agrégation temporelle est contrainte de manière à n'autoriser que des partitions qui sont composées d'intervalles de temps contigus et sans chevauchement.

Définition 5.5 On note $\mathcal{P}(T)$ une partition temporelle quelconque d'un modèle microscopique temporel. Celle-ci est composée par un sous-ensemble de parties (ou intervalles) $T_{(i,j)} \in \mathcal{I}(T)$ telle que $\forall (T_{(a,b)}, T_{(m,n)}) \in \mathcal{P}(T)^2$, $T_{(a,b)} = T_{(m,n)}$ ou $T_{(a,b)} \cap T_{(m,n)} = \emptyset$, avec $a \leq b$, $m \leq n$, et $\bigcup_{T_{(i,j)} \in \mathcal{P}(T)} T_{(i,j)} = T$.

$\mathcal{P}(T)_p$ est la partition temporelle optimale d'un modèle microscopique temporel pour une valeur du paramètre p donné (p étant le coefficient de pondération entre la réduction de complexité et de la perte d'information de la mesure de qualité pIC (voir Section Section 4.2).

$\boldsymbol{\mathcal{I}}(T)$ est l'ensemble de toutes les partitions temporelles possibles.

Opération d'agrégation

L'opération d'agrégation est la somme des valeurs (scalaires ou vectorielles) associées aux tranches de temps du modèle microscopique agrégées.

Définition 5.6 Soit $T_{(i,j)}$ un agrégat temporel du modèle microscopique $M^{\mu/t3}$ agrégeant les individus t contigus de t_i à $t_j \in T$. L'attribut associé à cet agrégat est la matrice $\mu^{t3}_{sum}(T_{(i,j)})$ de dimensions $(S, \mathcal{J}(E))$ telle que $\mu^{t3}_{sum}(T_{(i,j)}) = \sum_{t \in T_{(i,j)}} \mu^{t3}(t)$.

Nous notons également $\mu^{t3}_{sum}(s_k, T_{(i,j)}, J_l) = \sum_{t \in T_{(i,j)}} \mu^{t3}(s_k, t, J_l)$.

Conditions d'agrégation

La condition d'agrégation peut être reformulée comme la résolution du problème des partitions optimales, où la mesure de qualité qui doit être maximisée est le pIC, le compromis entre la perte d'information et la réduction de complexité, pondéré par la valeur du paramètre p fourni par l'utilisateur. Ceci est décrit dans la Section 4.2.

5.5 Extension de l'algorithme des partitions ordonnées optimales aux modèles microscopiques multidimensionnels

Dans le cas des modèles multidimensionnels, l'algorithme nécessite d'être étendu : il est nécessaire d'adapter le calcul des qualités à des valeurs vectorielles ou matricielles. Nous détaillons les modifications apportées dans cette Section. Nous reprenons aussi le calcul des meilleures coupes proposé par Lamarche-Perrin [8], dans sa version optimisée, en adaptant les notations.

5.5.1 Structures de données

Nous détaillons ici l'ensemble des structures de données utilisées par l'algorithme.

Forme globale des structures

Pour modéliser les structures de données correspondant aux mesures de qualité, nous représentons l'ensemble des intervalles de temps $\mathcal{I}(T)$ sous la forme d'une matrice triangulaire supérieure où chaque couple (i, j) correspond à un intervalle $T_{(i,j)}$.

Note *Par soucis de lisibilité, nous emploierons ponctuellement, dans la description de l'algorithme, la notation* $A[i,j]$ *pour désigner* $A(T_{(i,j)})$*,* $A[k,i,j]$ *pour* $A(s_k, T_{(i,j)})$ *et* $A[k,i,j,l]$ *pour* $A(s_k, T_{(i,j)}, J_l)$*.*

Qualités

Dans les modèles multidimensionnels, la valeur associée à un agrégat temporel $T_{(i,j)}$ n'est pas un scalaire. Nous adaptons donc le calcul des mesures de qualité à des valeurs matricielles. Les mesures de complexité d'un agrégat temporel $T_{(i,j)}$ du modèle $M^{\mu/t3}$ sont les sommes des mesures de complexité de chacun de ses coefficients.

Définition 5.7 Entropie de Shannon

$$\text{info}^{\text{t3}}(s_k, T_{(i,j)}, J_l) = \sum_{t \in T_{(i,j)}} \mu^{t3}(s_k, t, J_l) \log_2 \mu^{t3}(s_k, t, J_l) \tag{5.2}$$

Définition 5.8 Réduction de complexité

$$\text{gain}^{\text{t3}}(S_k, T_{(i,j)}, J_l) = \mu^{t3}_{sum}(s_k, T_{(i,j)}, J_l) \log_2 \mu^{t3}_{sum}(s_k, T_{(i,j)}, J_l) \quad \text{info}^{\text{t3}}(s_k, T_{(i,j)}, J_l) \tag{5.3}$$

$$\text{gain}^{\text{t3}}(T_{(i,j)}) = \sum_{J \in \mathcal{J}(E)} \left(\sum_{s \in S} \text{gain}^{\text{t3}}(S_k, T_{(i,j)}, J) \right) \tag{5.4}$$

Définition 5.9 Divergence de Kullback-Leibler

$$\text{loss}^{\text{t3}}(S_k, T_{(i,j)}, J_l) = \sum_{t \in T_{(i,j)}} \mu^{t3}(s_k, t, J_l) \log_2 \left(\frac{|T_{(i,j)}| \mu^{t3}(s_k, t, J_l)}{\mu^{t3}_{sum}(s_k, T_{(i,j)}, J_l)} \right) \tag{5.5}$$

$$\text{loss}^{\text{t3}}(T_{(i,j)}) = \sum_{J \in \mathcal{J}(E)} \left(\sum_{s \in S} \text{loss}^{\text{t3}}(S_k, T_{(i,j)}, J)) \right) \tag{5.6}$$

Structure « trade-off »

On définit une structure de données tradeoff qui contient les trois mesures de qualité utilisées pour déterminer les coupes temporelles pour un p donné, ainsi que des méthodes, permettant de les calculer, de les sommer ou de les comparer. On note tradeoff(i) la structure associée à la zone définie par $T_{(i,|T|)}$.

Données de « trade-off »

- le pIC, tel que défini par la formule pIC $= p \times$ gain $\quad (1 \quad p) \times$ loss ;
- la valeur de gain employée pour calculer pIC ;
- la valeur de loss employée pour calculer pIC.

Fonctions associées

— COMPUTEPIC$(p, \text{gain}, \text{loss})$ calcule la valeur du pIC de pour p, gain et loss (qu'elle enregistre dans la structure de données) et retourne une structure tradeoff contenant ces trois données ;
— INIT() retourne une structure tradeoff dont tous les champs sont à 0 ;
— Opérateur + : somme respectivement pIC, gain et loss des deux structures tradeoff_a et tradeoff_b ;
— Opérateur < (resp. >) : compare tradeoff_a et tradeoff_b, et retourne le résultat de l'expression booléenne :

$$(\text{pIC}_a < \text{pIC}_b) \vee ((\text{pIC}_a == \text{pIC}_b) \wedge ((\text{gain}_a < \text{gain}_b) \vee ((\text{gain}_a == \text{gain}_b) \wedge (\text{loss}_a > \text{loss}_b))))$$

5.5.2 Sortie de l'algorithme

Chaque partition $\mathcal{P}(T) \subset \boldsymbol{\mathcal{I}}(T)$ peut être représentée comme une séquence de coupes temporelles. Chaque coupe $\text{cut}(i)$ partitionne un intervalle $T_{(i,|T|)}$ en deux intervalles $\{T_{(i,\text{cut})}, T_{(\text{cut}+1,|T|)}\}$, tel que $i \leq cut < |T|$. L'algorithme d'agrégation temporelle calcule itérativement, pour chaque intervalle $T_{(i,|T|)}$, une valeur de coupe définissant une partition optimale de la zone correspondante :

— $\text{cut}(i) = x \in \{i, \ldots, |T| \quad 1\}$ indique une coupe temporelle, où x est l'indice de la période de temps microscopique à la fin de laquelle se situe la coupe ;
— $\text{cut}(i) = |T|$ indique qu'il n'y a pas de découpage.

Après exécution de l'algorithme, la partition optimale résultante est obtenue grâce à la séquence des coupes. On part de $T_{(1,|T|)}$: pour chaque intervalle $T_{(i,|T|)}$, on récupère $\text{cut}(i)$ et on répète cette opération récursivement sur le sous-intervalle de droite $T_{(\text{cut}+1,j)}$ jusqu'à ce que plus aucune coupe ne soit possible ($\text{cut}(i) = |T|$).

5.5.3 Description de l'algorithme

Les algorithmes 5.1 sont des fonctions employées au cours du calcul de qualités et des meilleures coupes.

Calcul des qualités

La première étape consiste à calculer les qualités (réduction de complexité et perte d'information) associées à chaque agrégat. Elle consiste principalement en plusieurs boucles imbriquées, où l'on réutilise le résultat des itérations précédentes. On minimise le stockage des données intermédiaires en profitant de l'additivité des mesures de qualité afin de contrôler la complexité spatiale. L'algorithme 5.2 décrit cette étape pour les modèles microscopiques $M^{\mu/t3}$.

Calcul des meilleures coupes

La seconde étape est le calcul des meilleures coupes associées à la partition optimale. Elle consiste en une itération pour calculer les coupes temporelles. On compare $\text{pIC}(T_{(i,|T|)})$ avec la somme des pIC des partitions optimales $\text{pIC}(T_{(i,\text{cut})}) + \text{pIC}(T_{(\text{cut}+1,|T|)})$ obtenues pour chaque coupe $\text{cut} \in \{i, \cdots, |T| \quad 1\}$ possible, et calculées au cours des itérations précédentes. On choisit cut tel que le pIC associé soit le plus grand. De cette manière, toutes

Algorithme 5.1 Fonctions annexes.

```
1: function COMPUTEPIC(p, gain, loss)
2:     return tradeoff{
3:     gain = gain
4:     loss = loss
5:     pIC = p × gain   (p   1) × loss
6:     }
7: end function
1: function ENTROPY(value)
2:     return value × log_2 value
3: end function
1: function ENTROPYREDUCTION(value, entropy)
2:     if value > 0 then
3:         return ENTROPY(value)   entropy
4:     else
5:         return 0
6:     end if
7: end function
1: function DIVERGENCE(size, value, entropy)
2:     return value × log_2 size   ENTROPYREDUCTION(value, entropy)
3: end function
```

Algorithme 5.2 Algorithme de calcul des qualités (étendu)

```
1: procedure COMPUTEQUALITIES
2:     ∀T_(i,j) ∈ I(T), gain^t3[i,j] et loss^t3[i,j] sont initialisés à 0
3:     for k = 1, ..., |S| do
4:         for l = 1, ..., |J(E)| do
5:             for i = |T|, ..., 1 do
6:                 μ^t3_sum/temp[i,i] = μ^t3[k,i,l]
7:                 info^t3_temp[i,i] = ENTROPYREDUCTION(μ^t3_sum/temp[i,i], 0)
8:                 for j = i + 1, ..., |T| do
9:                     μ^t3_sum/temp[i,j] = μ^t3_sum/temp[i+1,j] + μ^t3_sum/temp[i,i]
10:                    info^t3_temp[i,j] = info^t3_temp[i+1,j] + info^t3_temp[i,i]
11:                    gain^t3[i,j]+= ENTROPYREDUCTION(μ^t3_sum/temp[i,j], info^t3_temp[i,j])
12:                    loss^t3[i,j]+= DIVERGENCE((j   1 + 1), μ^t3_sum/temp[i,j], info^t3_temp[i,j])
13:                end for
14:            end for
15:        end for
16:    end for
17: end procedure
```

les coupes possibles sont évaluées, et donc toutes les partitions possibles. L'algorithme 5.3 décrit le calcul des meilleures coupes.

Algorithme 5.3 Algorithme de calcul des meilleures coupes.

```
1:  procedure COMPUTEBESTCUTS(p)
2:      for i = |T|, ..., 1 do
3:          cut[i] = |T|
4:          tradeoff[i] = COMPUTEPIC(p, gain^t[i, |T|], loss^t[i, |T|])
5:          for cut_t = i, ..., |T|    1 do
6:              tradeoff_t = COMPUTEPIC(p, gain^t[i, cut_t], loss^t[i, cut_t])
7:              tradeoff_t += tradeoff[cut_t +1]
8:              if tradeoff_t > tradeoff[i] then
9:                  cut[i] = cut_t
10:                 tradeoff[i] = tradeoff_t
11:             end if
12:         end for
13:     end for
14: end procedure
```

Note *On peut éventuellement normaliser les valeurs de qualités, tel que décrit par l'équation 4.7. Cette étape est effectuée avant le calcul des meilleurs coupes. On note* $p_{normalized}$ *la valeur de* p *associée à une partition* $\mathcal{P}_{p_{normalized}}$ *obtenue à partir des valeurs de gain et perte normalisés. À l'exception de* $p = 0$ *et* $p = 1$*,* $p_{normalized} = p$ *n'implique pas que* $\mathcal{P}_{p_{normalized}} = \mathcal{P}_p$*.*

Calcul des meilleures partitions

Il est possible de récupérer par dichotomie un ensemble de paramètres p pour lesquels les partitions optimales sont différentes. Un algorithme récursif, que l'on nommera algorithme de calcul des meilleures partitions, compare les mesures de qualités de deux partitions $\mathcal{P}(T)_{p_1}$ et $\mathcal{P}(T)_{p_2}$. Si elles diffèrent, alors on sauvegarde p_1 dans une liste, et on compare les mesures des qualités des partitions associées à p_1 et $p'_1 = \frac{p_1+p_2}{2}$, et à p'_1 et p_2, en s'arrêtant dans la récursion quand les mesures de qualités sont égales, ou quand $|p_1 \quad p_2|$ est inférieur à un seuil fourni par l'utilisateur.

5.5.4 Complexité théorique de l'algorithme

La complexité théorique de l'algorithme est démontrée par Lamarche-Perrin dans sa thèse de doctorat [8]. Dans sa version monodimensionnelle, le calcul des qualités possède une complexité spatiale et temporelle quadratique $\mathcal{O}(|T|^2)$. Le calcul des meilleures coupes possède la même complexité spatiale et temporelle $\mathcal{O}(|T|^2)$ (dans sa version optimisée). Dans notre version multidimensionnelle, seul change le calcul des qualités : la complexité temporelle devient $\mathcal{O}(|T|^2 \times |S| \times |\mathcal{J}(E)|)$. La complexité spatiale reste $\mathcal{O}(|T|^2)$ car nous profitons de l'additivité des mesures de qualité pour ne pas avoir à stocker les valeurs intermédiaires. Le calcul des meilleures partitions possède une complexité qui est, dans

le pire cas, $\mathcal{O}(\log_2(\text{seuil}) \times |\boldsymbol{\mathcal{I}}(T)| \times |T|^2)$. On distingue l'influence du nombre de comparaisons entre les valeurs de p, limité par le seuil, $\mathcal{O}(\log_2(\text{seuil}))$, une borne supérieure délimitée par le nombre maximal de partitions optimales possibles $\mathcal{O}(|\boldsymbol{\mathcal{I}}(T)|)$, qui limite la descente dans la récursion, et la partie associée au calcul des qualités de chaque partition évaluée $\mathcal{O}(|T|^2)$.

5.5.5 Interprétation de la sortie de l'algorithme

La Figure 5.6 montre le résultat fourni par l'algorithme d'agrégation appliqué sur un modèle microscopique monodimensionnel temporel pour plusieurs valeurs de p. Ici, il est aisé d'interpréter la signification des partitions : on agrège en priorité les valeurs scalaires qui sont les plus proches, en tenant compte de la contrainte de contiguïté. On constate ainsi que l'élément qui résiste le plus à l'agrégation est l'intervalle $T_{(5)}$, qui possède la valeur la plus différente. L'homogénéité s'interprète donc intuitivement par la proximité des valeurs numériques scalaires. On notera la faible de valeur de p (0,0357056) pour la partition totalement agrégée, ce qui signifie que la minimisation de la perte d'information reste largement majoritaire dans le compromis matérialisé par le pIC. La représentation n'est en effet constituée que de dix entités dont les attributs sont scalaires, ce qui correspond à un système très simple, dont la perte d'information reste faible même lorsque l'on agrège l'ensemble de la représentation.

Note *Les paramètres p donnés dans les Figures de ce Chapitre et du suivant possèdent un nombre de chiffres significatifs susceptible de varier. Il s'agit de valeurs récupérées automatiquement par notre programme, et dont un arrondi serait susceptible de fournir une valeur aboutissant à une partition différente.*

Dans le cas multidimensionnel, on considère que deux éléments sont homogènes si chacun des coefficients de leurs valeurs matricielles sont proches deux à deux. Dans le cas d'une trace bidimensionnelle (espace et temps), on considère donc que des intervalles de temps sont homogènes si chaque ressource a un comportement homogène avec elle-même au cours du temps (dans le cas d'un modèle bidimensionnel), **sans que cela n'implique un comportement homogène entre elles**. Ce principe s'étend à des traces tridimensionnelles. La Figure 5.7 illustre que l'intervalle $T_{(8)}$ est le plus difficile à agréger, malgré l'homogénéité spatiale des deux ressources : en effet, seule l'homogénéité temporelle est prise en compte.

5.6 Visualisation d'une partition du système

La sortie de l'algorithme n'étant pas facile à interpréter puisque le nombre d'entités qu'elle contient peut être élevé, nous avons conçu plusieurs techniques de visualisation qui la retranscrivent de manière graphique.

5.6.1 Histogramme

L'histogramme est une méthode élémentaire pour représenter la partition $\mathcal{P}(T)$ d'un modèle monodimensionnel $M^{\mu/t1}$. On en trouve un exemple Figure 5.8. L'intérêt de cette méthode est de montrer à la fois la partition et l'amplitude des valeurs associées à chaque agrégat.

Modèle microscopique :

23	25	22	30	55	21	26	35	19	30

t_1 t_6 t_{10}

T

Agrégation : $p = 0$

23	25	22	30	55	21	26	35	19	30

Agrégation : $p = 0,00823975$

70	30	55	47	35	19	30

Agrégation : $p = 0,0167847$

100	55	131

Agrégation : $p = 0,0357056$

286

FIGURE 5.6 – Exemple d'agrégations appliquées sur un modèle microscopique monodimensionnel temporel par l'algorithme d'agrégation d'un ensemble ordonné, avec différents paramètres p.

5.6.2 Partition

Dans le cas des modèles multidimensionnels, les valeurs associées à chaque agrégat n'étant pas des scalaires, il n'est pas possible d'afficher un histogramme. La solution la plus simple consiste à afficher la partition. Afin de distinguer visuellement chaque agrégat, on crée une séquence de couleurs sous la forme d'un vecteur $\overrightarrow{\text{couleur}}$ de dimension T. Voici quelques critères que cette séquence doit remplir :

— La couleur de chaque rectangle associé à un intervalle $T_{(i,j)}$ est déterminée par couleur(i). Cela permet de garder une cohérence entre la couleur des agrégats et leur position temporelle lorsque l'on change la partition en utilisant un paramètre p différent ;

— La séquence de couleur n'est pas aléatoire et est donc toujours identique d'une session d'analyse à une autre ;

— On minimise le nombre de couleurs proches ou identiques dans le vecteur $\overrightarrow{\text{couleur}}$, en particulier pour les indices contigus, dans le but de pouvoir distinguer les différents agrégats facilement.

Cette visualisation s'applique à tous les modèles microscopiques (mono et multidimensionnels). La Figure 5.9 représente la construction d'une visualisation des partitions à partir d'un modèle bidimensionnel. En particulier, on visualise la correspondance entre

Modèle microscopique :

s_1	23	25	22	30	**55**	21	26	35	19	30
s_2	75	72	71	69	**92**	73	75	**35**	70	71

t_1 t_6 t_{10}

T

Agrégation : $p = 0,00952148$

s_1	100	**55**	47	35	49
s_2	287	**92**	148	**35**	141

Agrégation : $p = 0,0181885$

s_1	202	35	49
s_2	527	**35**	141

FIGURE 5.7 – Exemple d'agrégations temporelles appliquées sur un modèle microscopique bidimensionnel, avec différents paramètres p : on agrège en prenant en compte le comportement temporel de chacune des ressources mais pas leur comportement spatial.

les agrégats de différentes partitions générées avec différentes valeurs de p.

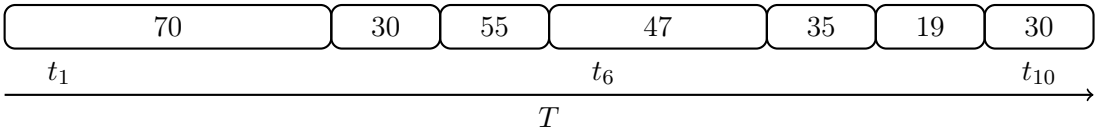
5.6.3 Histogramme empilé

Principe

La représentation des partitions est pratique pour observer des zones de la trace qui sont entièrement désagrégées et donc symptomatiques d'un comportement hétérogène. Il est plus difficile, pour l'analyste, d'étudier le découpage en phases de l'application quand ces dernières sont plus homogènes car il ne possède pas d'information sur le comportement de celles-ci (ressources impliquées, types d'évènements prédominants, etc.). Pour palier à cet inconvénient, nous proposons un histogramme empilé appliqué à la représentation des modèles tridimensionnels, où nous agrégeons la dimension S à l'aide de l'opérateur somme et représentons l'amplitude de la métrique associée à chaque type d'évènements de l'ensemble $\mathcal{J}(E)$ pour chaque agrégat. Nous attribuons à chacun de ces types d'évènements J une couleur couleur(J). La Figure 5.10 représente la construction d'un histogramme empilé généré à partir d'un modèle tridimensionnel.

Note *Nous ne proposons pas de solution à base d'histogramme empilé pour les modèles bidimensionnels temps-ressources, car le nombre d'éléments de la dimension spatiale ne permet généralement pas le passage à l'échelle de ce genre de représentations.*

Partition :



Histogramme :

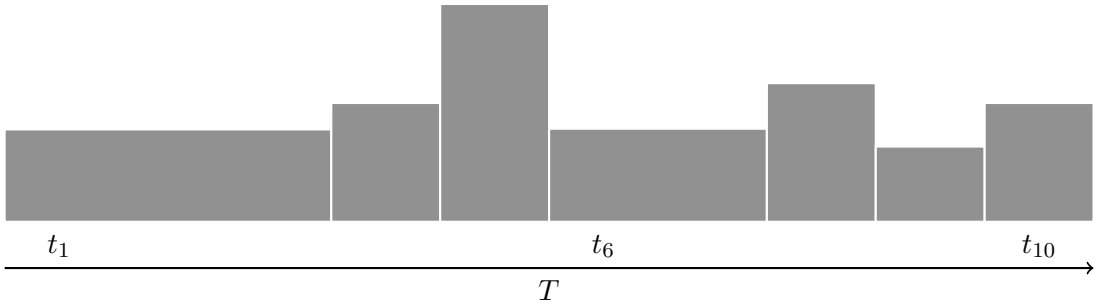


FIGURE 5.8 – Représentation d'une partition temporelle d'un modèle microscopique monodimensionnel sous la forme d'un histogramme.

Modèle microscopique :

s_1	23	25	22	30	**55**	21	26	35	19	30
s_2	75	72	71	69	**92**	73	75	**35**	70	71

t_1 t_6 t_{10}

T

Partition : $p = 0$

Partition : $p = 0,00952148$

Partition : $p = 0,0181885$

FIGURE 5.9 – Représentation d'une partition temporelle d'un modèle microscopique bidimensionnel.

Agrégation visuelle des types d'évènements

Il existe des situations où la hauteur hauteur$(T_{(i,j)}, J)$ d'un ou plusieurs rectangles est trop petite pour être affichée correctement. Dans ce cas, nous faisons appel à une technique d'agrégation visuelle. Le processus est le suivant : on affiche l'empilement des rectangles dont les hauteurs sont supérieures à un seuil, selon un ordre arbitraire appliqué à $\mathcal{J}(E)$. On affiche au-dessus de l'empilement une figure représentant l'agrégation de tous les types dont les rectangles associés ont une hauteur inférieure au seuil. Si la somme h_{aggreg} des hauteurs de ces rectangles est supérieure au seuil, la figure est dessinée sous la forme d'un rectangle de hauteur h_{haggreg}, d'une couleur particulière (noire dans nos exemples). Sinon, on représente un symbole indiquant à l'analyste l'agrégation. La Figure 5.11 représente le processus d'agrégation visuelle.

5.6.4 Mode

En statistique, le mode ou valeur dominante désigne la valeur la plus représentée d'une variable dans une population. Dans notre cas, c'est une représentation intermédiaire entre la partition et l'histogramme empilé, idéale quand la surface disponible, et en particulier la hauteur, est faible. Ici, chaque partie $T_{(i,j)}$ est représentée avec la couleur couleur(J_{mode}) associée au type d'évènement $J_{\text{mode}} \in \mathcal{J}(E)$ dont la valeur $\sum_{s \in S} \mu^{t3}_{sum}(s, T_{(i,j)}, J_{\text{mode}})$ est la plus élevée. Il est possible d'employer une mesure de transparence (le *canal alpha* du format de couleur RGBA, par exemple) pour représenter son amplitude en faisant ainsi varier l'intensité de la couleur.

5.7 Analyse temporelle

Nous consacrons cette section à décrire l'analyse de plusieurs cas d'usage à l'aide d'Ocelotl, l'outil qui implémente notre technique d'agrégation temporelle. Ocelotl est un module de Framesoc [11, 12, 18], l'infrastructure de gestion et d'analyse de traces du projet SoC-Trace [35]. Ces logiciels sont décrits plus en détail dans le Chapitre 7. Toutes les traces générées lors de l'exécution des différentes applications sont d'abord importées dans Framesoc puis stockées sous la forme de bases de données à l'aide d'un module dédié, pour être enfin analysées avec Ocelotl. Des cas d'études supplémentaires sont présentés en Annexe A.

5.7.1 Applications multimédia

L'objectif original du projet SoC-Trace [35] concerne l'analyse du déroulement d'applications embarquées multimédia issues d'un contexte industriel, comme la lecture d'un programme télévisé en streaming grâce à une set-top box, l'enregistrement d'une émission sur le disque dur, ou encore le décodage d'un fichier accédé à travers le réseau. Les applications sur téléphone mobile et autres tablettes sont aussi visées.

Le décodage ou la lecture d'une vidéo implique des contraintes temps réels souples : un certain nombre de traitements *pipelinés* doivent être exécutés dans un temps imparti afin de pouvoir afficher un nombre d'images par seconde nécessaire pour la fluidité du rendu. Il existe une certaine tolérance, et il est possible de dépasser ponctuellement une *deadline* sans que cela ne soit gênant pour l'utilisateur. Néanmoins, si plusieurs d'entre elles sont

Modèle microscopique :

(s_1, J_R)	23	25	22	30	55	21	26	35	19	30
(s_1, J_G)	12	11	11	12	0	12	11	13	14	11
(s_1, J_B)	0	0	5	6	8	12	6	5	0	6
(s_2, J_R)	75	72	71	69	92	73	75	68	70	71
(s_2, J_G)	10	10	10	12	11	15	25	12	13	15
(s_2, J_B)	0	0	10	12	15	13	18	20	0	10

t_1 t_6 t_{10}

T

Histogramme de la partition $p = 0,0280151$:

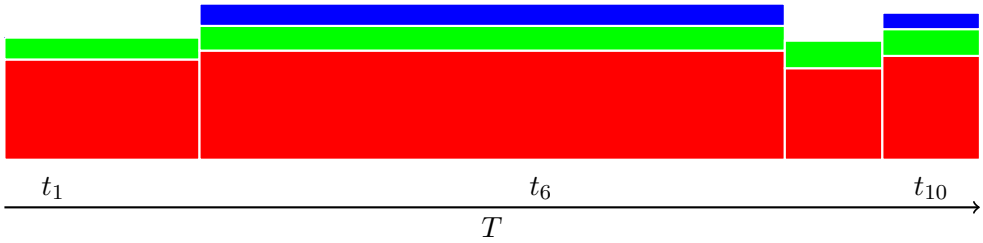
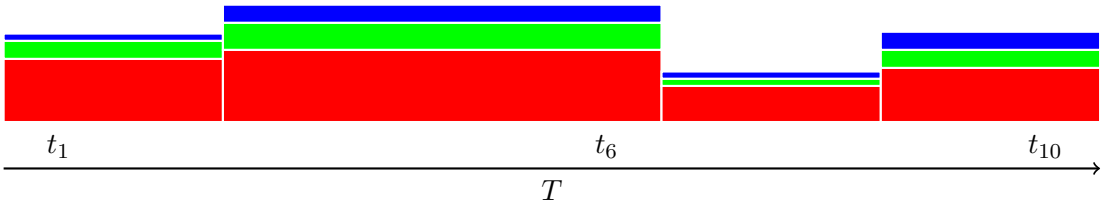


FIGURE 5.10 – Représentation d'un histogramme empilé généré à partir d'une partition d'un modèle tridimensionnel.

Histogramme sans agrégation visuelle, seuil $> a$:



Histogramme avec agrégation visuelle, seuil $< a$:

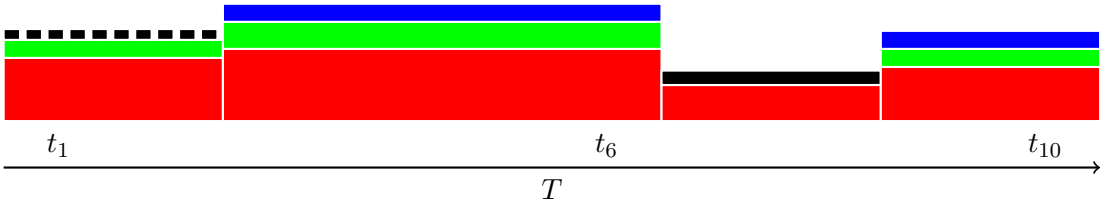


FIGURE 5.11 – Processus d'agrégation visuelle d'un histogramme empilé : la diminution du seuil force la représentation de bleu par un symbole, en pointillé ($T_{(1,2)}$), et fusionne vert et bleu sous la forme d'un seul agrégat, en noir ($T_{(7,8)}$).

dépassées dans un laps de temps court, alors la lecture de la vidéo devient saccadée au niveau de l'image et du son. Lorsque ce problème est récurrent, on soupçonne alors un problème de conception de l'application ou de dimensionnement de la plate-forme. Le rôle de l'analyse est de déterminer ce qui est responsable de ce comportement.

TABLE 5.1 – Cas d'études *applications multimédia*. Les cas notifiés par (*) sont traités dans l'Annexe A.

Cas d'étude	**Conditions expérimentales**	**Format**	**Durée**	**Ressources**	**Évènements**	**Taille de la trace**	**Taille de la DB**
GST-A	Référence	Pajé	34 s	1581	750045	40,5 Mo	79,2 Mo
GST-B	Stress à 15 s	Pajé	34 s	1581	749876	40,0 Mo	79,6 Mo
GST-C	Référence	Pajé	314 s	1528	8290576	455,2 Mo	874,9 Mo
GST-D	Stress à 60 s	Pajé	314 s	1593	8302874	455,9 Mo	876,2 Mo
TSR-S	Version courte	KPTrace	82 s	55	258519	12,7 Mo	16,7 Mo
TSR-L	Version longue	KPTrace	482 s	57	1642033	83,3 Mo	109,1 Mo
UNI-H*	Haut niveau	KPTrace	45 s	33	149857	7,0 Mo	10,0 Mo
UNI-B*	Bas niveau	KPTrace	39 s	28	324447	27,7 Mo	28,8 Mo

Lecture d'une vidéo perturbée par un stress du système

GStreamer[3] est une bibliothèque logicielle destinée à construire des applications multimédia sous la forme de graphes de composants, connectés entre eux via des *pipelines*[4]. Il permet la lecture de différents formats vidéo et audio, notamment grâce à l'utilisation de greffons. Nous proposons d'employer GStreamer afin d'illustrer, de manière pédagogique, le comportement d'une application multimédia lisant une vidéo MP4 et perturbée par un stress du système.

Nous traçons l'exécution du programme grâce à la fonctionnalité `GST_DEBUG`[5], qui affiche des messages estampillés, et fournissant des informations sur le déroulement de l'application. Ces messages sont associés à des niveaux de criticité, dans l'ordre décroissant : `ERROR`, `WARNING`, `FIXME`, `INFO`, `DEBUG`, `LOG`, `TRACE`, `MEMDUMP`. Il est possible de filtrer les niveaux les plus faibles pour éviter une trop grande intrusivité et diminuer le bruit dans la trace (dans notre cas, nous ne tracerons pas le niveau `MEMDUMP`). La Figure 5.12 montre l'association de ces types de messages avec la couleur employée au sein d'Ocelotl et Framesoc pour les représenter.

Nous enregistrons les traces issues de l'exécution de deux vidéos de différentes longueurs (34 et 314 secondes), cela ayant un impact sur le nombre d'évènements contenus dans la trace et sur la mémoire requise par GStreamer. Pour chacune des vidéos, nous traçons une référence qui correspond à un comportement normal sans perturbation, et une exécution perturbée à l'aide du programme *stress*[6]. La perturbation consiste en un processus appelant la fonction `sync()`[7] en boucle pendant une seconde :

```
$ stress --io 1 --timeout 1s
```

[3]http ://gstreamer.freedesktop.org/
[4]ressemblants, sur le principe, aux tubes UNIX.
[5]http://gstreamer.freedesktop.org/data/doc/gstreamer/head/gstreamer/html/gst-running.html
[6]http://people.seas.harvard.edu/~apw/stress/
[7]Cette fonction vide les tampons du disque en forçant l'écriture des blocs modifiés sur ce dernier.

Elle est déclenchée une certaine durée après le démarrage de la vidéo grâce à la commande `sleep`. Les conséquences sur la fluidité de la vidéo sont perceptibles par l'utilisateur à ce moment : il y a une légère saccade au moment de l'exécution du programme *stress*. La perturbation est cependant volontairement légère, afin de rendre sa détection par un outil d'analyse plus difficile. L'ensemble de ces opérations est réalisé sur la machine *birdy*, dont les spécifications sont données par la Figure 5.13.

Afin de prendre en considération les durées entre les évènements GStreamer, nous les considérons comme des états, dont le début est l'estampille associée lors du traçage, et la fin, l'estampille de l'évènement survenant directement après. Cela permet d'analyser la quantité d'évènements GStreamer en les considérant comme des évènements ponctuels (on ne tient alors pas compte de leur durée), mais aussi la variation des durées entre deux évènements GStreamer consécutifs, ce qui peut nous renseigner sur une rupture dans la régularité des motifs.

Pour garantir une parfaite séquentialité de ce procédé, nous forçons l'exécution de GStreamer sur un seul cœur. Les quatre cas sont récapitulés dans la Table 5.1 sous appellation **GST-[X]**.

L'intérêt de ces cas d'études est de confirmer que notre technique d'agrégation décèle le moment où l'application est perturbée. Les cas GST-A (la référence) et GST-B (le cas perturbé) concernent des traces de taille inférieure à 100 Mo et il est possible, en pratique, de retrouver la perturbation dans le second cas avec des techniques classiques comme le diagramme espace-temps. Le point faible du diagramme espace-temps est ici l'espace, car il n'est pas possible de représenter correctement plus de 1500 ressources. À contrario, la visualisation fournie par Ocelotl est synthétique et nous donne un aperçu des quantités de temps passé dans chaque type d'états au cours du temps (Figures 5.14 et 5.15) en s'affranchissant de la représentation de l'espace. Ocelotl permet également une comparaison qualitative entre la trace de référence GST-A, Figure 5.14, et GST-B, Figure 5.15, où l'on observe clairement leur similarité, à l'exception de la perturbation survenant dans le second cas.

Pour ces deux figures, nous avons choisi une partition que nous avons jugée significative. Ce choix s'effectue, en pratique, grâce aux courbes de qualités, que nous représentons dans l'outil d'analyse. La Figure 5.16 représente la réduction de complexité et la perte d'information des partitions optimales $\mathcal{P}(T)_{p_{normalized}}$ en fonction de $p_{normalized}$, dont les valeurs sont obtenues grâce à l'algorithme de calcul des meilleures qualités évoqué Section 5.5.3, associées au cas GST-B. Il est intéressant d'observer l'allure de ces courbes, composées de paliers et de sauts d'une grande amplitude. Ces paliers laissent émerger ce que l'on nomme des niveaux d'abstractions : ils correspondent à un ensemble de partitions optimales pour lesquelles les valeurs de réduction de complexité et de perte d'information associées sont relativement proches, et l'échelle des phénomènes qu'ils représentent est du même ordre. Dans le cas GST-B, on distingue quatre niveaux d'abstractions. La Figure 5.17 associe à chacun de ces niveaux une ou plusieurs partitions représentatives, visualisées selon la technique décrite dans la Section 5.6.2. Le niveau D représente les phénomènes de plus hauts niveaux : il comprend la partition $\mathcal{P}(T)_{p_{normalized}=1}$ composée d'un unique agrégat, mais aussi des partitions mettant en exergue l'initialisation et la terminaison de l'application. Le niveau C donne plus de détails sur ces deux comportements. Le niveau d'abstraction B est celui qui intéresse potentiellement le plus l'analyste, puisqu'il met en évidence la perturbation autour de 15 secondes. Enfin, le niveau A désa-

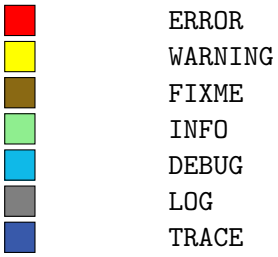


FIGURE 5.12 – Association entre les types d'évènements GStreamer et la couleur utilisée pour les représenter au sein d'Ocelotl et Framesoc.

```
# Système d'exploitation :
Linux birdy 3.17.3-200.fc20.x86_64 #1 SMP
Fri Nov 14 19:45:42 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux
# Matériel :
Nombre de coeurs : 8
Spécifications des coeurs
- model name : Intel(R) Core(TM) i7-2760QM CPU @ 2.40GHz
- cache size : 6144 KB
- hyperthreading : actif
cpufreq gouverneurs : performance
RAM : 8131804 Ko
Disques
- Disque 1 (SSD 256 Go)
/dev/sda:
 Model=LITEONIT LAT-256M2S, FwRev=PNDA,
 SerialNo=TW0PVGPX550851B21854
 Config={ Fixed }
 RawCHS=16383/16/63, TrkSize=0, SectSize=0, ECCbytes=0
 BuffType=unknown, BuffSize=unknown, MaxMultSect=16, MultSect=16
 CurCHS=16383/16/63, CurSects=16514064, LBA=yes, LBAsects=500118192
 IORDY=on/off, tPIO={min:120,w/IORDY:120}, tDMA={min:120,rec:120}
 PIO modes:  pio0 pio3 pio4
 DMA modes:  mdma0 mdma1 mdma2
 UDMA modes: udma0 udma1 udma2 udma3 udma4 udma5 *udma6
 AdvancedPM=no WriteCache=enabled
 Drive conforms to: unknown:  ATA/ATAPI-1,2,3,4,5,6,7
 * signifies the current active mode
```

FIGURE 5.13 – Spécifications de la machine *birdy*.

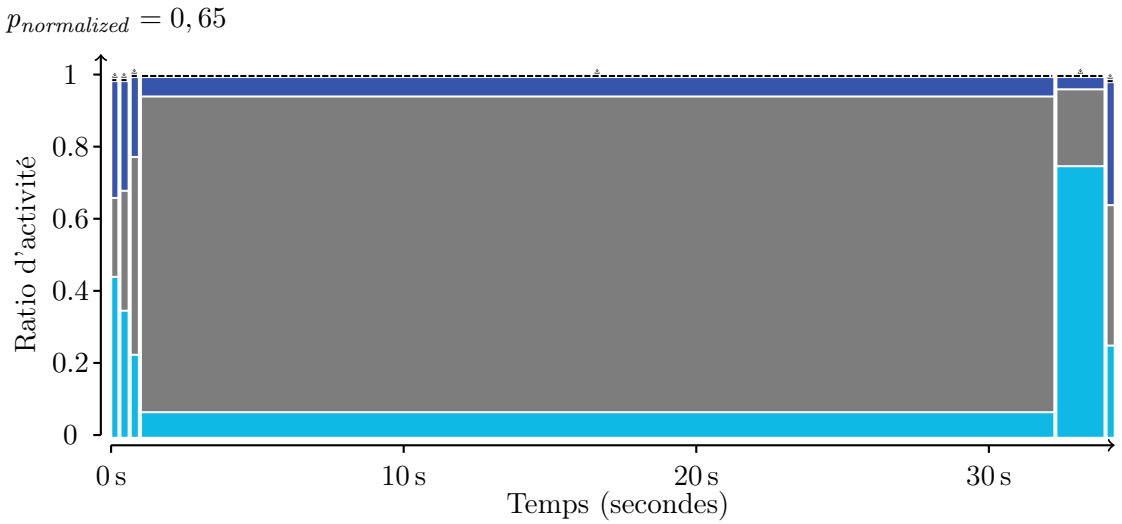


FIGURE 5.14 – Histogramme empilé représentant une partition optimale et le ratio d'activité de GST-A. Le comportement apparait homogène, excepté au début et à la fin de la trace, où l'on distingue une phase d'initialisation et de terminaison. Trois types d'états sont apparents : `DEBUG` (bleu clair), `LOG` (gris), `TRACE` (bleu foncé).

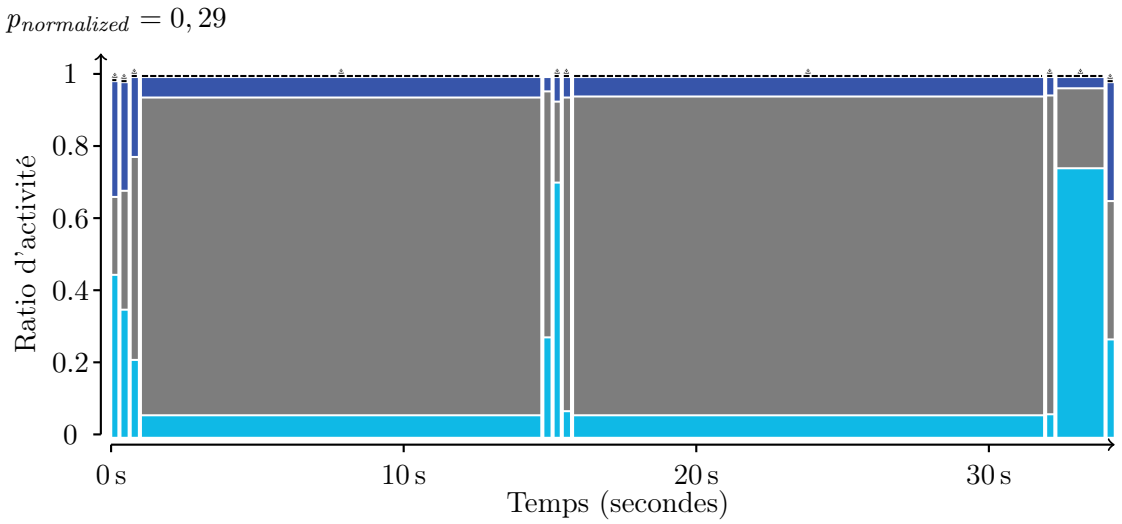


FIGURE 5.15 – Histogramme empilé représentant une partition optimale et le ratio d'activité de GST-B. On distingue clairement une rupture de comportement à 15 secondes, en plus des motifs d'initialisation et de terminaison de l'application identiques à celle de la trace de référence. Celle-ci se matérialise, entre autre, par une proportion de la durée des états différente de la phase homogène (plus grande proportion d'états `DEBUG`).

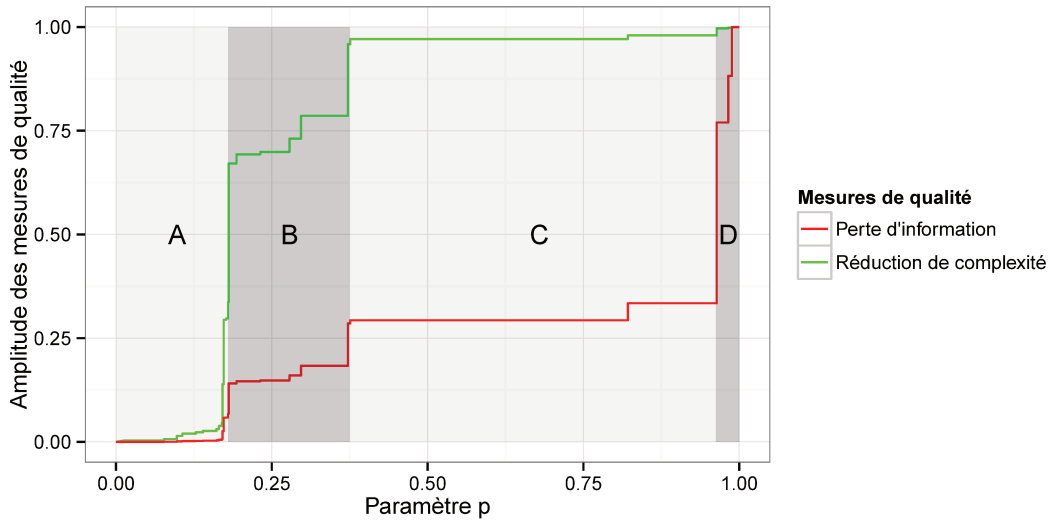


FIGURE 5.16 – Courbes de gain et de perte (normalisées) associées à GST-B. On distingue quatre niveaux d'abstraction (A, B, C, D, du plus complexe au moins complexe), séparés par des sauts dans chacune des deux courbes. Chacun de ces niveaux correspond à un ensemble de partitions qui représentent des phénomènes d'une même échelle.

grège fortement la représentation, car la sensibilité à la perte d'information est très forte. Les deux niveaux extrêmes A et D correspondent aux cas où il n'y a pas ou peu de compromis entre la réduction de complexité et la perte d'information : ils sortent de la zone de respect des critères G1 (entity budget) et G5 (fidélité) d'Elmqvist & Fekete. Les deux niveaux intermédiaires B et C, à l'inverse, nous donnent une information pertinente sur les différentes phases de l'application, et sur la perturbation, qui est un phénomène d'une échelle plus fine que les séquences d'initialisation et de terminaison de l'application, ce qui explique qu'on ne la distingue que dans un niveau d'abstraction plus bas. Cette approche dynamique fournit donc à l'analyste un moyen de hiérarchiser des phénomènes d'échelles différentes en faisant varier la puissance de l'agrégation.

Le choix des métriques utilisées pour la construction du modèle microscopique influe sur l'agrégation. Nous profitons de leur complémentarité pour obtenir plus d'information sur le comportement de l'application. Dans la Figure 5.15, nous représentons le ratio d'activité de chaque type d'états (c'est-à-dire la quantité de temps passé dans chaque type d'états par unité de temps). Cette métrique nous permet de constater que la proportion des états de type `DEBUG` dans la perturbation est plus importante que dans le *régime permanent* de l'application. La Figure 5.18 montre, quant à elle, la quantité d'évènements par unité de temps : la diminution de cette quantité aux alentours de la perturbation nous laisse penser qu'il y a moins d'évènements, et que ceux-ci durent plus longtemps. Ceci est corroboré par la Figure 5.19, qui représente la quantité de temps moyen par état passé dans chaque type d'état, par unité de temps. Notre hypothèse est donc que nous avons affaire à des états `DEBUG` d'une durée anormalement longue.

Notre implémentation s'inscrit dans la méthodologie de Shneiderman [1] : nous fournissons à l'analyste un *overview* de l'application, mais aussi la possibilité de *zoomer*, ou plus exactement, de ré-effectuer l'analyse sur une sous-partie de la trace. La Fi-

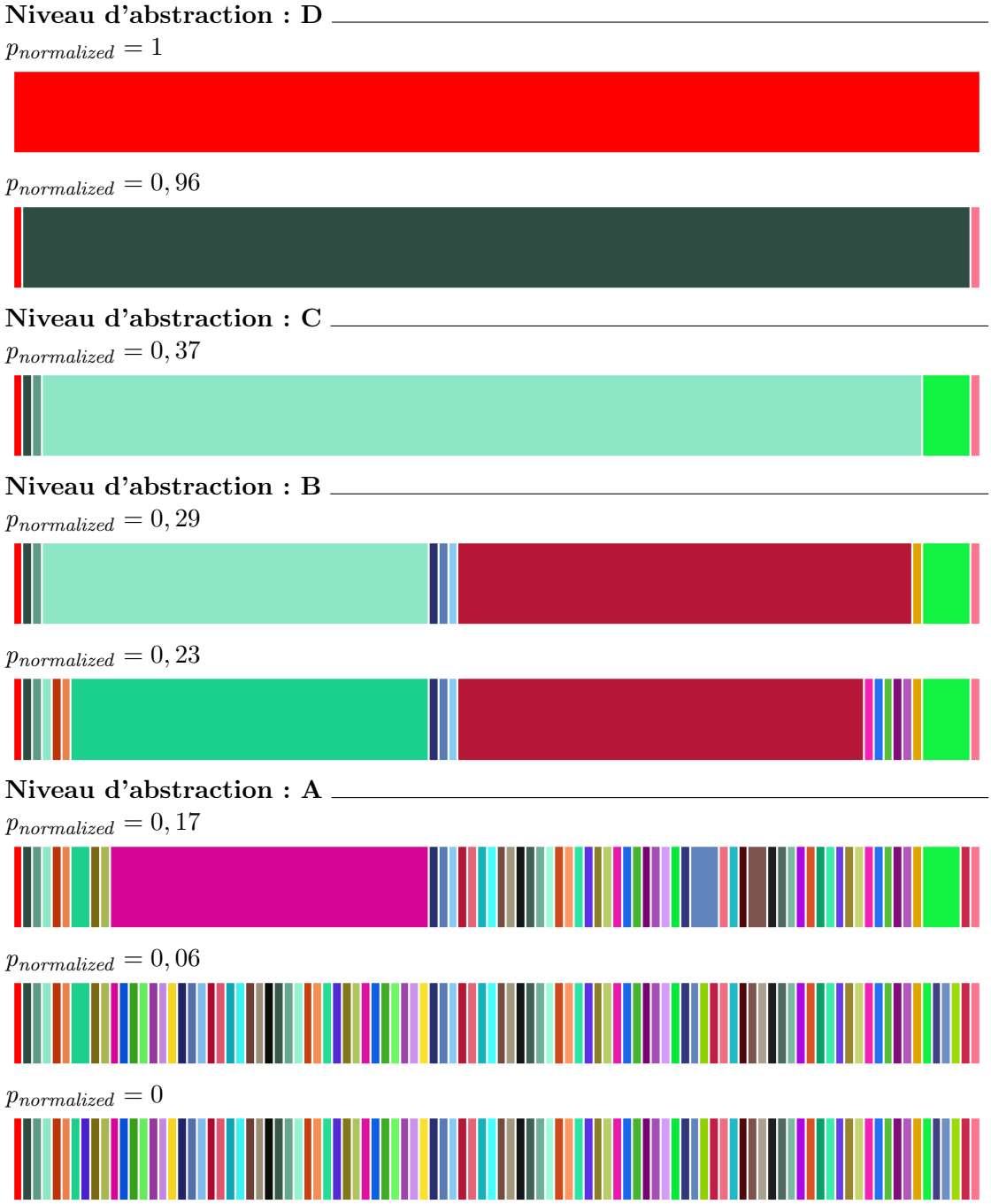


FIGURE 5.17 – 8 partitions du cas GST-B répartis en 4 niveaux d'abstractions (voir Figure 5.16), et montrant la désagrégation progressive du système.

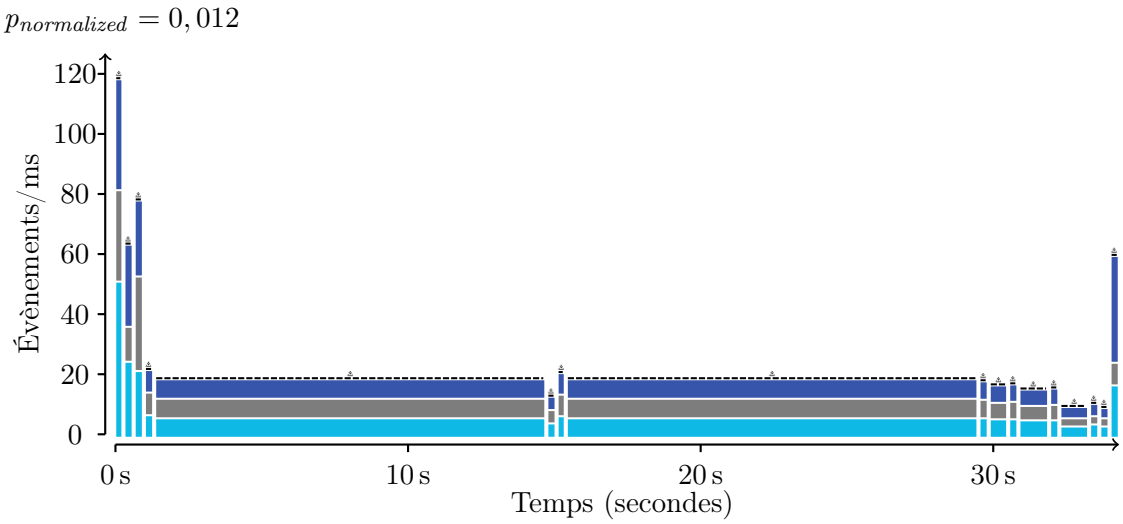


FIGURE 5.18 – Histogramme empilé représentant une partition optimale et la quantité d'évènements par unité de temps de GST-B. Nous retrouvons la perturbation autour de 15 secondes, matérialisée par une légère fluctuation de la quantité d'évènements par unité de temps.

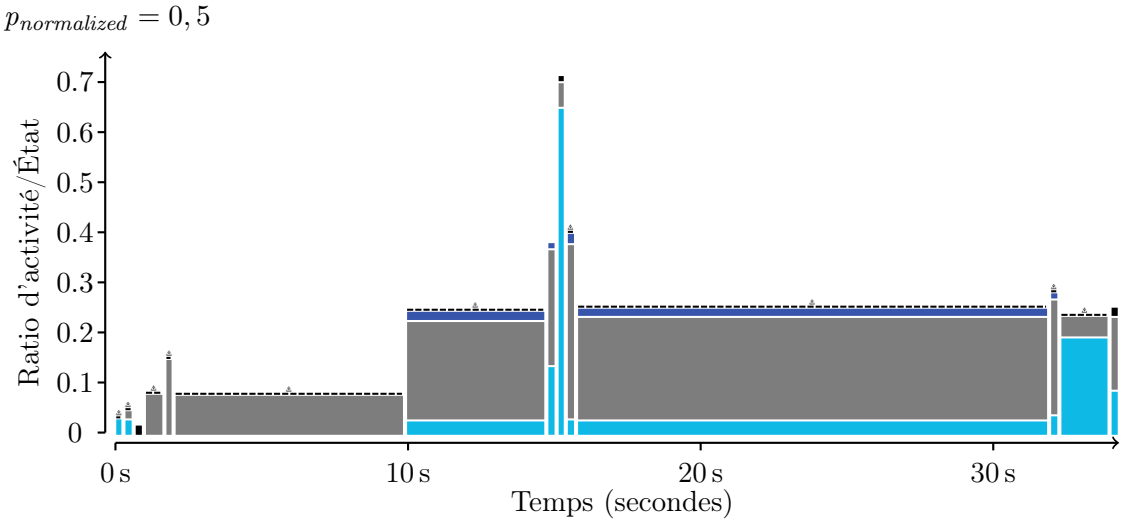


FIGURE 5.19 – Histogramme empilé représentant une partition optimale et la quantité de temps moyen de chaque type d'états par unité de temps de GST-B. La perturbation à 15 secondes est visible. Elle se distingue par un temps moyen par état plus long que dans le reste de la trace.

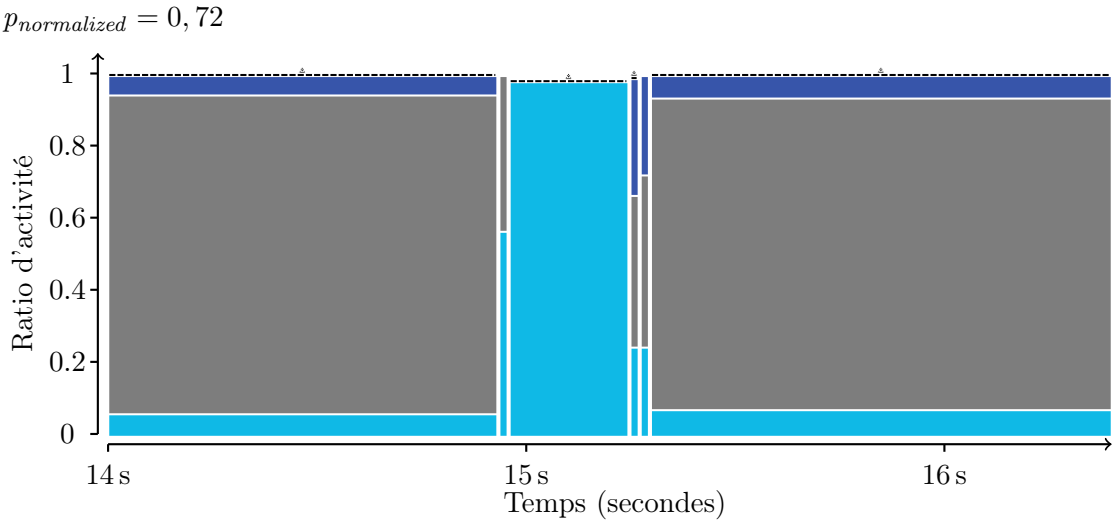


FIGURE 5.20 – Histogramme empilé représentant une partition optimale et le ratio d'activité d'une sous-partie temporelle de GST-B, focalisée sur la perturbation survenant à 15 secondes. Celle-ci apparait très visiblement, et est majoritairement constituée par l'état `DEBUG`.

gure 5.20 montre ainsi un focus temporel sur la perturbation nous donnant plus de détails sur sa composition. Enfin, l'interaction fournie par l'outil permet de commuter avec un diagramme espace-temps sur ces bornes de temps et d'accéder aux évènements matérialisant la perturbation, comme montré Figure 5.21. Nous visualisons ici l'impact du *stress* qui a bloqué momentanément l'application GStreamer dans la fonction `gst_system_clock_id_wait_jitter_unlocked_dbg`.

Dans les cas GST-C (la référence) et GST-D (le cas perturbé), le volume des traces et la quantité d'évènements rendent l'analyse avec un diagramme espace-temps beaucoup plus difficiles. Certains diagrammes espace-temps (Pajé [30], T-Charts [7]) ne permettent pas d'afficher autant d'évènements. Celui de LTTng Eclipse Viewer [44] (repris au sein de Framesoc) possède une plus grande tolérance, mais l'agrégation employée empêche de détecter la perturbation dans GST-D, comme montré par la Figure 5.22. La Figure 5.23 confirme que grâce à Ocelotl, nous retrouvons la perturbation générée à l'aide de *stress* à 60 secondes d'exécution.

TSrecord

Le cas d'étude TSrecord est une trace fournie par STMicroelectronics aux partenaires du projet SoC-Trace, et représente un scénario type de déverminage d'une application multimédia embarquée autour duquel s'articulent et coopèrent l'ensemble des méthodes d'analyse développées dans le cadre de ce projet. Nous reprenons ici la description du cas d'étude faite par Martin *et al.* [96, 97].

Les traces sont produites au format KPTrace [36] en enregistrant les différents évènements survenant sur chacune des couches logicielles (noyau, intergiciel, application), tels que les appels de fonctions, les interruptions logicielles, matérielles, les changements

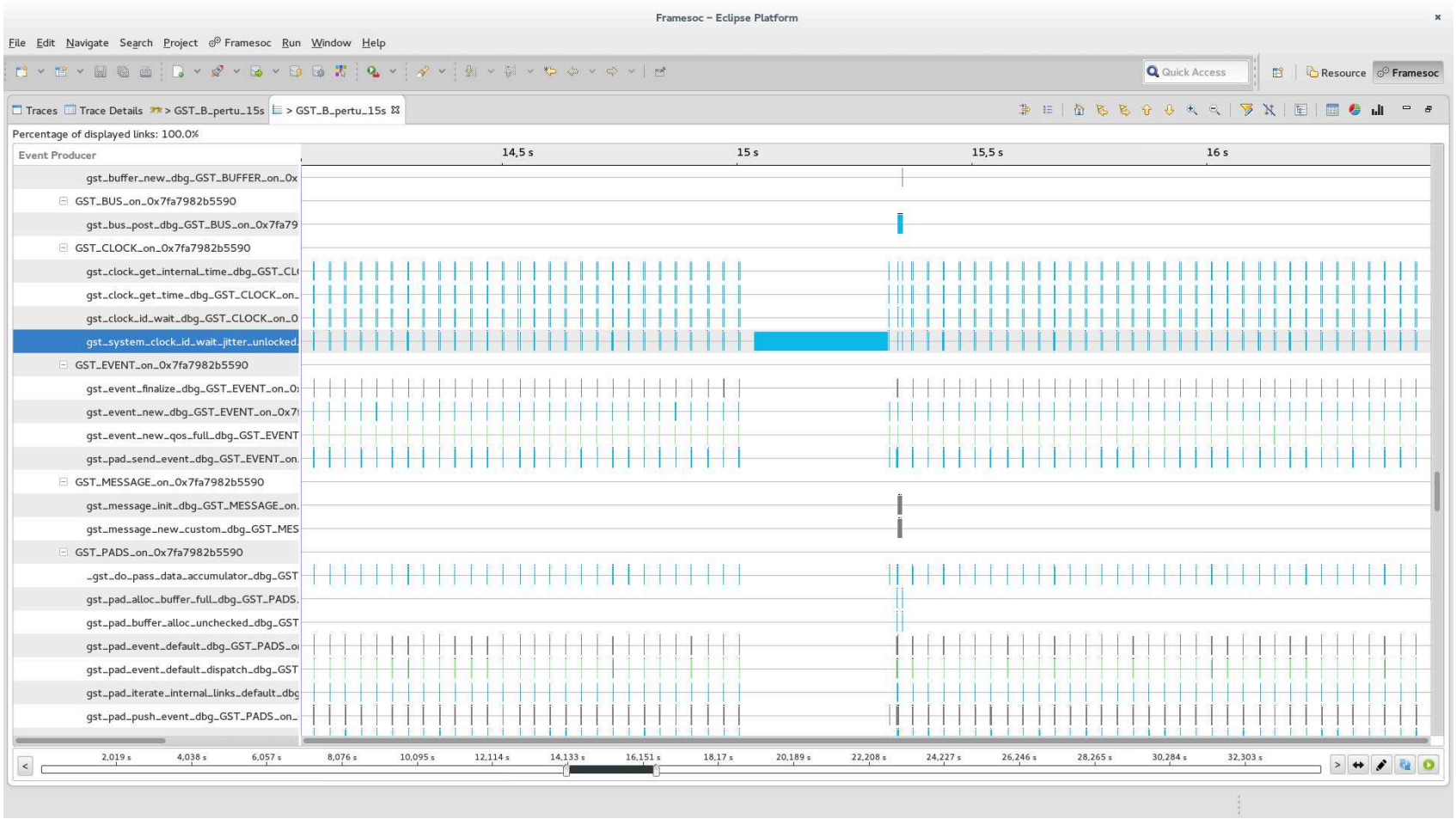


FIGURE 5.21 – Diagramme espace-temps de Framesoc du cas GST-B, focalisé sur les bornes de temps de la Figure 5.20. On détecte ici un état de type `DEBUG` d'une durée anormalement longue, associé à la fonction `gst_system_clock_id_wait_jitter_unlocked_dbg`.

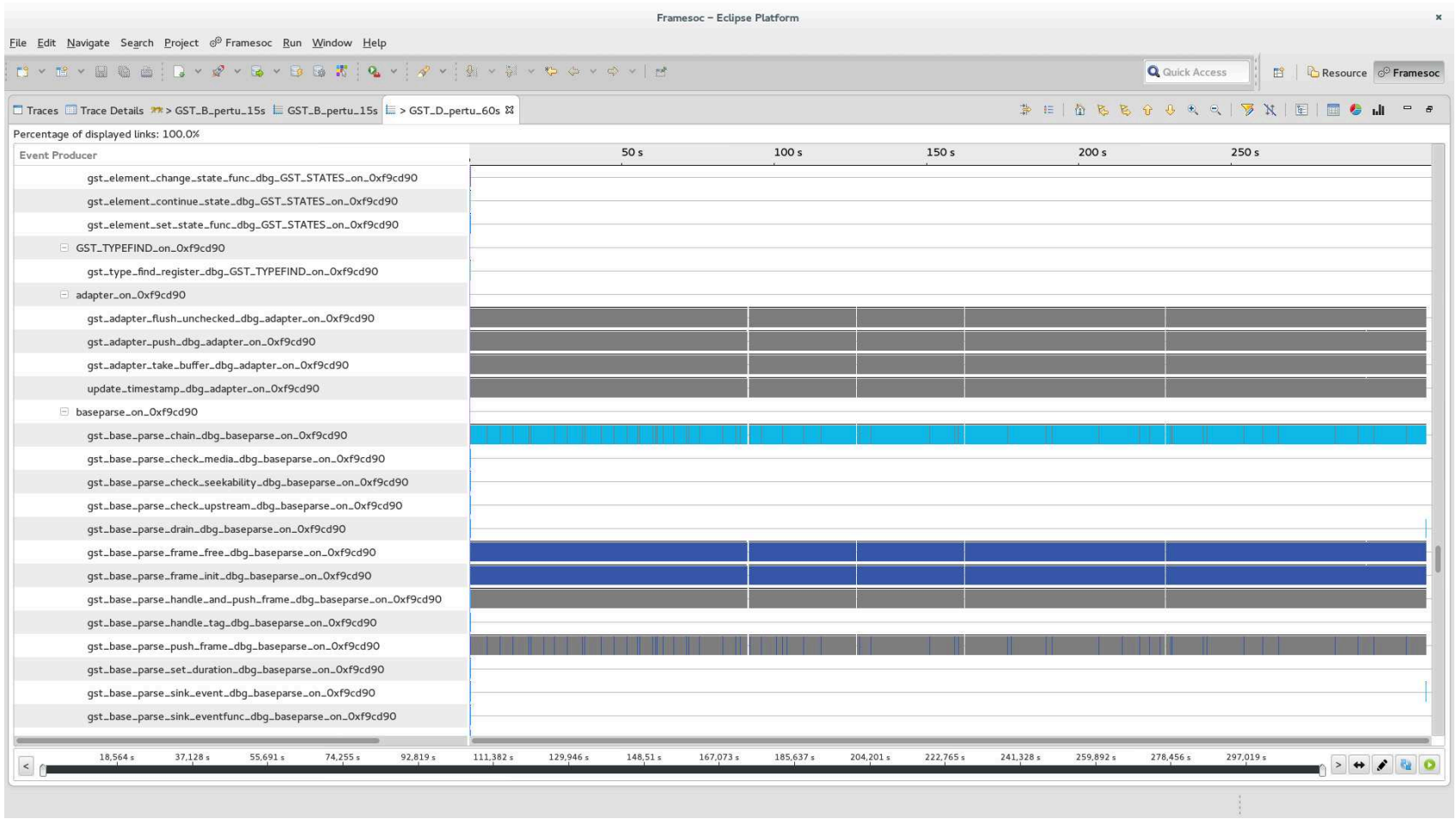


FIGURE 5.22 – Diagramme espace-temps de Framesoc, montrant l'ensemble de la trace GST-D. L'agrégation employée masque la perturbation à 60 secondes.

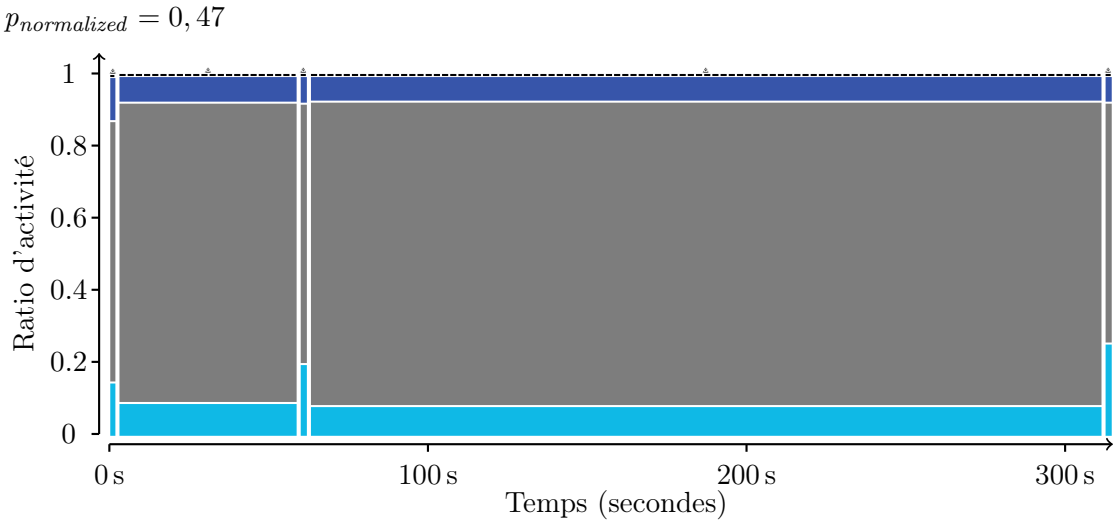


FIGURE 5.23 – Histogramme empilé représentant une partition optimale et le ratio d'activité de GST-D. Une perturbation émerge à 60 secondes.

de contexte, etc. La Figure 5.24 associe à chacun de ces évènements une couleur, utilisée par Ocelotl et Framesoc pour les représenter. Les traces que nous employons ici ont été générées par Oleg Iegorov, doctorant au LIG et chez STMicroelectronics. Elles sont indiquées dans la Table 5.1 par l'identifiant **TSR-[X]**. Nous nous focalisons sur l'étude de la plus longue, TSR-L.

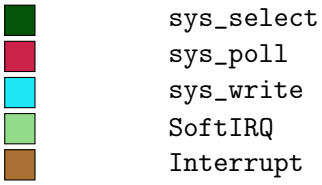


FIGURE 5.24 – Association entre les principaux types d'évènements contenus dans les traces **TSR-[X]** et la couleur employée par Ocelotl et Framesoc pour les représenter.

L'application TSrecord consiste en la lecture d'un flux vidéo depuis le réseau qui est enregistré sur un disque dur USB. Elle correspond à un cas d'utilisation classique d'une set-top box où l'utilisateur enregistre une émission. La vidéo obtenue n'est pas d'une qualité satisfaisante : on constate des sauts d'images et des artefacts sonores, dus à une perte d'information durant l'enregistrement. En effet, les tampons IP stockant temporairement le flux vidéo sont vidés toutes les 100 ms, et les données qu'ils contiennent sont copiées vers le disque, en transitant par son cache. Toutes les 5000 ms, le cache est verrouillé, le temps d'effectuer la copie sur le disque. Pendant cette opération, les données provenant des tampons IP sont perdues.

Le but de l'analyse avec notre technique d'agrégation temporelle est de réussir à détecter les périodes où le comportement de l'application est problématique, sans prendre en compte les informations connues à priori par les développeurs sur le phénomène res-

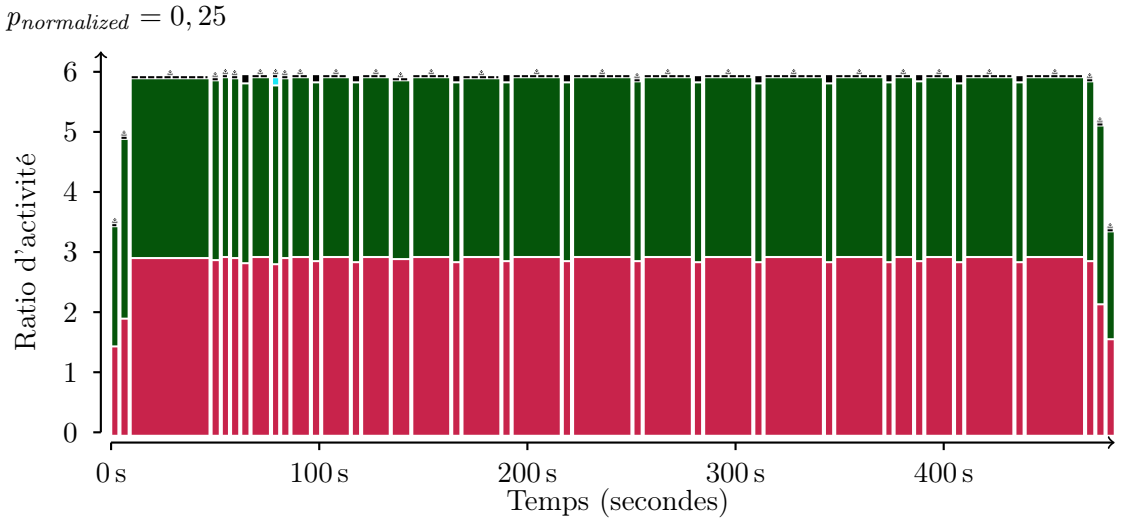


FIGURE 5.25 – Histogramme empilé représentant une partition optimale et le ratio d'activité de TSR-L. Trois types d'états sont apparents : `sys_select` (vert foncé), `sys_poll` (magenta), `sys_write` (cyan). On distingue des perturbations régulières (matérialisées, pour la plupart, par la présence visible d'états `sys_write`).

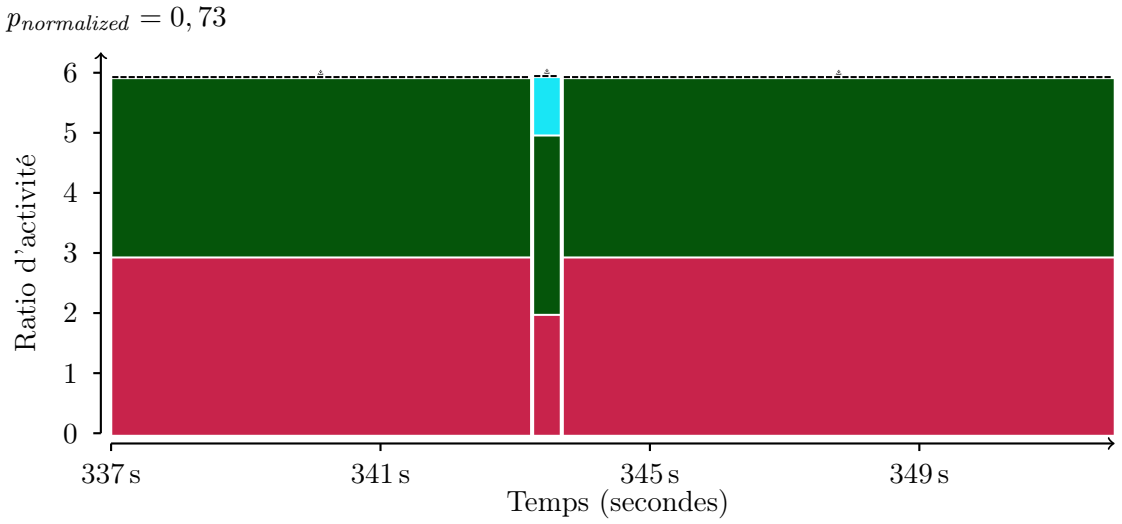


FIGURE 5.26 – Histogramme empilé représentant une partition optimale et le ratio d'activité d'une sous partie temporelle de TSR-L, focalisée sur une perturbation. Celle-ci est caractérisée par une forte activité de l'état `sys_write`.

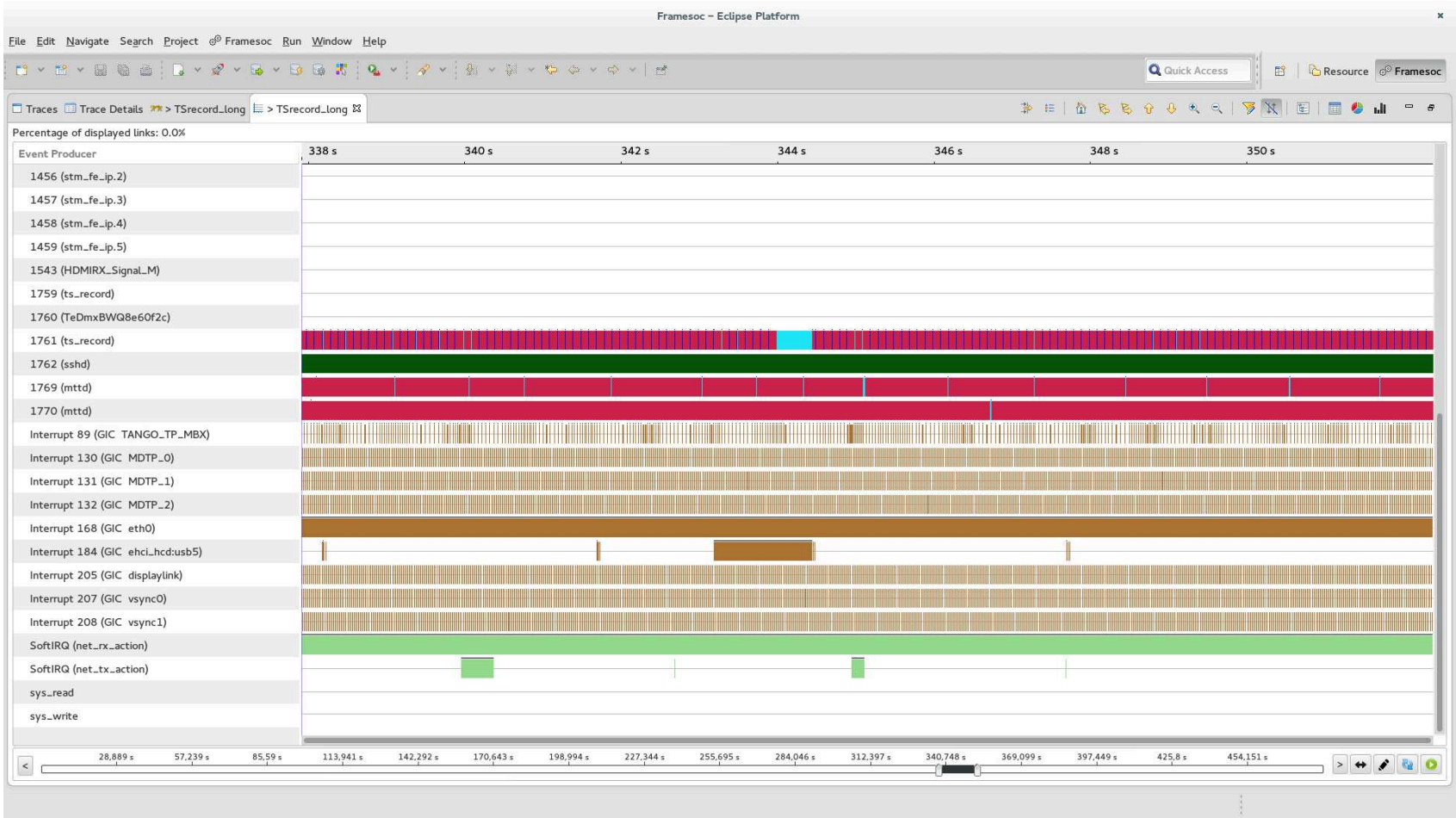


FIGURE 5.27 – Diagramme espace-temps de Framesoc du cas TSR-B, focalisé sur les bornes de temps de la Figure 5.27. On détecte ici que le processus `ts_record` reste anormalement bloqué dans l'état `sys_write`. Ce phénomène est concomitant avec un fort taux d'interruptions logicielles relatives à l'écriture sur le disque USB (`Interrupt 184`).

ponsable de la mauvaise qualité de la vidéo. Pour cela, nous suivons la méthodologie décrite pour l'application GStreamer, consistant à détecter les niveaux d'abstractions à l'aide des courbes de qualités. La Figure 5.25 illustre une partition optimale où des perturbations sont apparentes. En se focalisant sur l'une d'entre elles, comme montré Figure 5.26, on décèle une forte proportion de l'état `sys_write`, comparé à la phase homogène. Commuter vers le diagramme espace-temps de Framesoc, Figure 5.27, nous aide à faire le lien entre l'activité de `sys_write` et la forte quantité d'interruptions relatives à l'écriture sur le disque USB. La même méthodologie d'analyse est ensuite réitérée sur plusieurs perturbations. L'analyste procède finalement à une étude bas niveau du système et du code de l'application axée sur ces deux types d'évènements. Notre technique d'agrégation apporte ici un moyen rapide de se focaliser sur les *évènements intéressants* de la trace, à contrario des techniques d'analyse statistique usuelles, où l'on évalue des distances entre des occurrences ou encore des corrélations, ce qui requiert de choisir à tâtonnement un sous-ensemble d'évènements avec lesquels commencer l'analyse.

5.7.2 Applications parallèles

Nous nous focalisons ici sur l'analyse d'applications parallèles, et plus précisément les applications de calcul algébrique gradient conjugué (CG) et décomposition LU fournies par les benchmarks NASPB [98] (version MPI). Nous les exécutons sur la plate-forme Grid'5000, et nous instrumentons et traçons les appels MPI à l'aide de Score-P [25], sur différents sites, et avec des paramètres différents (machines, clusters impliqués). Les spécifications des composants matériels et du réseau sont consultables sur le site internet de Grid'5000[8]. Les traces sont générées au format OTF2 [29], converties au format Pajé [31] grâce au convertisseur *otf22paje*[9], et importées dans Framesoc. Seuls les appels MPI sont enregistrés lors de l'exécution de l'application, afin de minimiser l'intrusivité et la taille des fichiers générés. La Table 5.2 récapitule l'ensemble des cas d'études CG et LU. La Figure 5.28 montre les couleurs associées aux appels MPI.

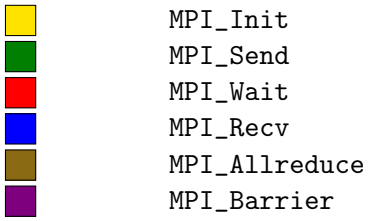


FIGURE 5.28 – Association entre les principaux types d'évènements contenus dans les traces issues des cas d'études *NASPB* et la couleur employée par Ocelotl et Framesoc pour les représenter.

Gradient conjugué

La méthode du gradient conjugué est un algorithme résolvant des systèmes d'équations linéaires dont la matrice est symétrique définie positive. Ce noyau est généralement employé pour tester les communications, et en particulier la latence du réseau d'une grille

[8] https://www.grid5000.fr
[9] https://github.com/schnorr/akypuera

TABLE 5.2 – Cas d'études *NASPB*. Les cas notifiés par * sont traités dans l'Annexe A.

Cas d'étude	Bench-mark	Site	Grappes (Machines)	Proces-sus	Durée	Évènements	Taille trace	Taille DB
CG.C-Gre-64	CG.C	Grenoble	adonis(2) edel(3) genepi(3)	64	26 s	3838144	205,3 Mo	261,5 Mo
CG.C-Nan-64	CG.C	Nancy	graphene(4) graphite(2) griffon(2)	64	128 s	3838144	205,0 Mo	261,3 Mo
CG.C-Ren-64	CG.C	Rennes	paradent(2) parapide(3) parapluie(1)	64	54 s	3838144	205,8 Mo	261,5 Mo
CG.C-Gre-512	CG.C	Grenoble	adonis(9) edel(24) genepi(31)	512	22 s	49149440	2,7 Go	3,4 Go
CG.C-Nan-512	CG.C	Nancy	graphene(20) graphite(4) griffon(46)	512	288 s	49149440	2,7 Go	3,4 Go
LU.C-Gre-700*	LU.C	Grenoble	adonis(10) edel(47) genepi(31)	700	28 s	218457456	12,2 Go	14,9 Go
LU.C-Nan-700*	LU.C	Nancy	graphene(26) graphite(4) griffon(67)	700	74 s	218457456	12,3 Go	15 Go

faiblement structurée, car il communique de manière intensive en envoyant des petits messages [98].

Nous avons exécuté CG, classe C, 64 processus sur 3 sites différents (Grenoble, Figure 5.29, Nancy, Figure 5.30 et Rennes, Figure 5.31). On constate que les performances globales contrastent d'une plateforme à l'autre, de 28 (Grenoble) à 128 secondes (Nancy) de temps de calcul. La partition sépare la phase `MPI_Init`, en jaune, qui a une durée du même ordre de grandeur (2 à 3 secondes) dans chaque cas, de la phase de calcul. C'est la durée de cette dernière qui varie selon la plate-forme. L'histogramme empilé nous aide à établir la proportion des différents appels MPI. On constate que le ratio activité de `MPI_Send` reste compris entre 11 et 16 dans les trois cas, tandis que l'amplitude du ratio d'activité `MPI_Wait` varie de 20 à 50, une valeur élevée étant corrélée avec un temps d'exécution plus long. Ici, on peut relier facilement les performances avec la technologie des interconnects présente sur les différents sites :

— Grenoble : l'ensemble des grappes et machines communiquent ensemble par Infiniband, le débit à l'intérieur de chacune des grappes étant de 40G pour *edel* et *adonis*, 20G pour *genepi*, et le débit entre les grappes de 10G, ce qui explique les bonnes performances de ce benchmark ;

— Rennes : les machines des grappes *parapide* et *parapluie* possèdent des cartes Infiniband 10G (les deux clusters communiquent entre eux par l'intermédiaire de cet interconnect) et Ethernet 1G. Les machines de la grappe *paradent* sont équipées de cartes Ethernet 1G seulement, d'où une latence et un débit de communications plus longs que pour Grenoble. Il est cependant à noter que la machine parapluie-1 (la seule que nous utilisons sur ce cluster) possède 24 cœurs, ce qui permet à ses 24 processus de communiquer ensembles sans passer par le réseau ;

— Nancy : les machines de la grappe *graphite* possèdent des cartes Ethernet 10G,

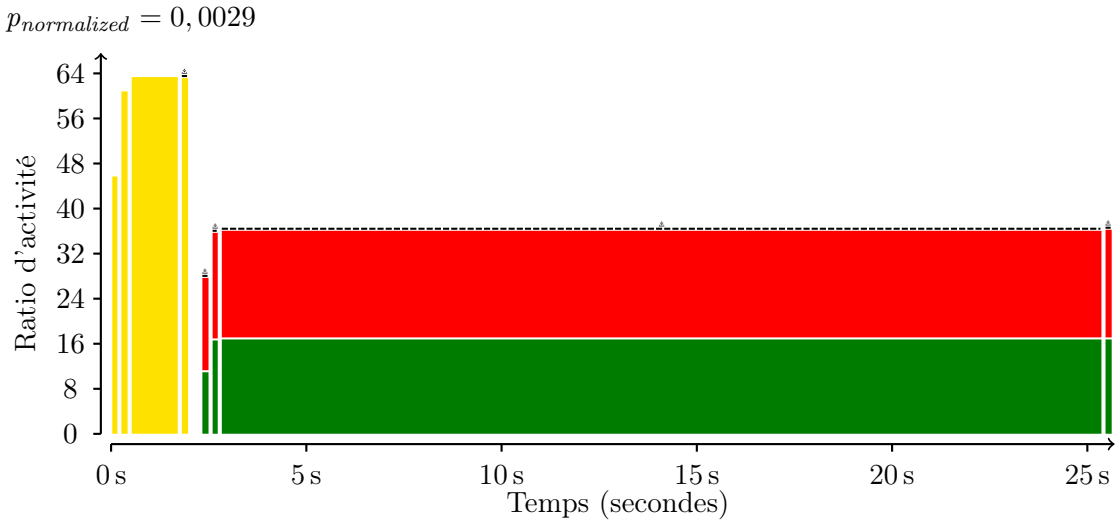


FIGURE 5.29 – Histogramme empilé représentant une partition optimale et le ratio d'activité de CG.C-Gre-64.

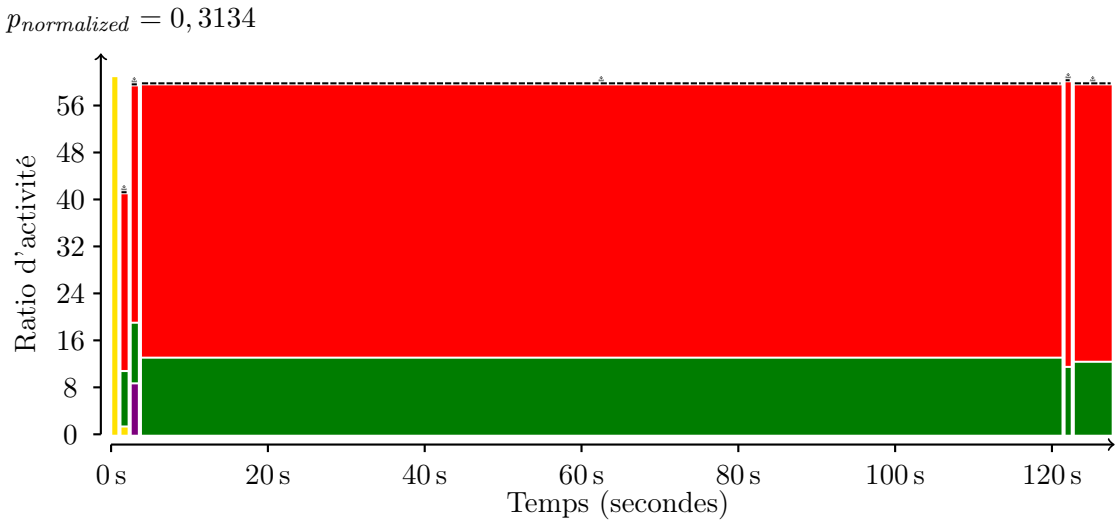


FIGURE 5.30 – Histogramme empilé représentant une partition optimale et le ratio d'activité de CG.C-Nan-64.

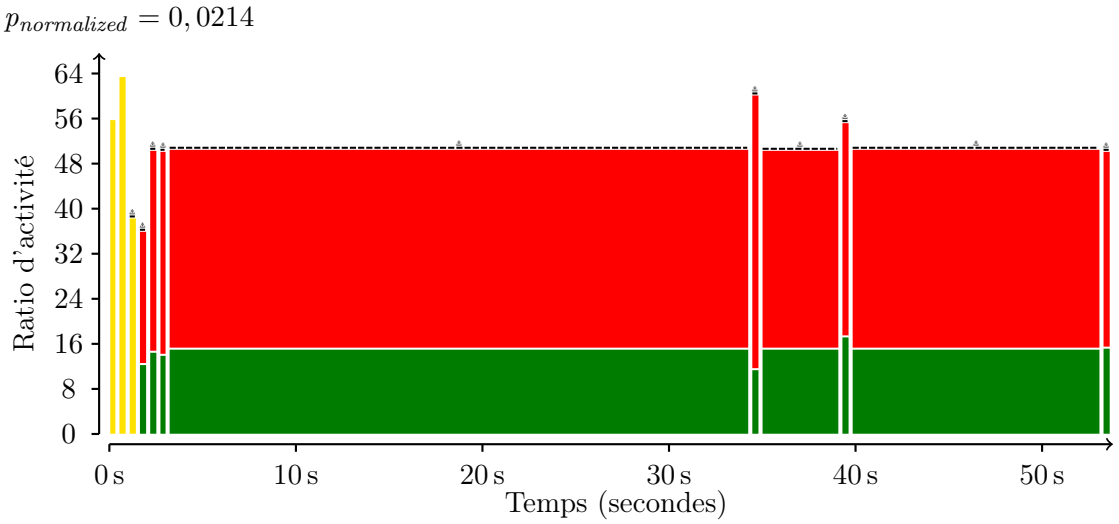


FIGURE 5.31 – Histogramme empilé représentant une partition optimale et le ratio d'activité de CG.C-Ren-64. Il est à noter deux perturbations visibles entre 30 et 40 secondes.

celles de *graphene* et *griffon* des cartes Infiniband 20G (débit interne et entre les deux grappes) et Ethernet 1G (débit extérieur). La configuration réseau entre *graphite*, et *graphene* et *griffon* (1G), et le fait que les machines impliquées ont au maximum 8 cœurs, comparé à parapluie-1 (24 cœurs) à Rennes, expliquent les mauvaises performances du benchmark.

Une analyse statistique sur les durée des appels MPI conduirait aux mêmes conclusions. Cependant, notre technique apporte des informations plus précises concernant le déroulement de la phase de calcul. Dans le cas de Grenoble, l'homogénéité temporelle de la phase de calcul est forte, ce qui est matérialisé par l'allure des courbes de mesures de qualité et le passage brutal sans intermédiaire d'une partition très agrégée à une partition complètement désagrégée. Dans le cas de Nancy, une légère perturbation est mise en évidence vers 122 secondes d'exécution. Enfin, pour Rennes, deux perturbations beaucoup plus nettes sont détectées entre 30 et 40 secondes. La grille de calcul étant accédée en concurrence avec d'autres utilisateurs, le partage des ressources réseau, qui sont, dans ce benchmark, un facteur limitant, peut expliquer cette fluctuation.

L'exécution de CG avec 512 processus sur Grenoble et Nancy confirme le comportement du benchmark en fonction de la plateforme, avec un rapport de 1:13 entre leur durée d'exécution. Ici, l'analyse avec Ocelotl ne met pas en évidence de perturbation.

5.8 Bilan

5.8.1 Respect des critères d'Elmqvist-Fekete

Nous confrontons ici notre technique d'agrégation temporelle face aux critères d'Elmqvist-Fekete développés Section 3.2. Il est important de noter que les techniques de visualisation que nous proposons ne se comportent pas toutes de la même manière face à ces critères. La Table 5.3 en fait une synthèse. Nous mettons en évidence l'effet de l'agrégation vi-

Table 5.3 – Évaluation de notre technique d'agrégation temporelle et des visualisations associées face aux critères Elmqvist-Fekete.

Visualisation	Modèles compatibles	Présence d'un axe d'ordonnées	Agrégation visuelle	G1	G2	G3	G4	G5	G6	M
Histogramme	Monodimensionnel			•		•	•	•	•	•
Partition	Multidimensionnel			•		•	•	•	•	•
Histogramme empilé (1)	Multidimensionnel	•			•	•	•	•	•	•
Histogramme empilé (2)	Multidimensionnel	•	•	•	•	•	•	•	•	•
Mode	Multidimensionnel		•	•	•	•	•	•	•	•

suelle en montrant deux versions de l'histogramme empilé : sans (1) et avec (2) agrégation visuelle. Dans le cas du critère G1 (« entity budget »), la discrétisation temporelle permet un contrôle par l'utilisateur du nombre d'éléments graphiques selon l'axe du temps. Concernant l'axe des ordonnées, l'histogramme empilé satisfait ce critère grâce à une technique d'agrégation visuelle appliquée aux types d'évènements. Les autres techniques présentées étant monodimensionnelles, il n'y a pas de problème de passage à l'échelle sur cette dimension. Les histogrammes, et dans une moindre mesure le mode, transcrivent les valeurs associées aux agrégats (G2) à l'inverse de la partition ($\overline{G2}$). L'ensemble des représentations est simple (G3), les agrégats étant constitués de formes géométriques basiques (rectangles, pointillés). Tous les éléments de la représentation sont des agrégats des données de la trace brut, le problème de discriminabilité des agrégats de données ne se pose pas (G4). Concernant les agrégats visuels, leur représentation dans l'histogramme empilé les différencie grâce à l'emploi de la couleur noire, de pointillés, et d'un symbole avertissant l'utilisateur de leur présence. Nous estimons remplir également le critère de fidélité (G5). En effet, bien qu'il existe une perte d'information due à l'étape de discrétisation du modèle microscopique, l'utilisateur en contrôle certains paramètres (le nombre de tranches de temps, par exemple), tandis que l'opérateur d'agrégation employé est explicite. De plus, la perte d'information et de la réduction de complexité associée à la représentation macroscopique sont portées à la connaissance de l'analyste. Enfin, la complémentarité des métriques et des techniques de visualisation permettent de compenser la perte d'information liée à chaque opération d'agrégation. En terme d'interprétabilité (G6), l'intensité de l'agrégation est belle et bien adaptée de manière à garder du sens dans la représentation, puisque l'accès à différents niveaux d'abstraction est explicite et choisi par l'utilisateur. La principale difficulté vis à vis de ce critère est le fait d'agréger en tenant compte de la dimension spatiale, qui n'est pas représentée, ce qui peut induire une certaine confusion vis-à-vis du processus d'agrégation.

Face aux représentations temporelles ou spatiotemporelles de la Table 3.1, et en particulier les diagrammes espace-temps, notre technique a une meilleure réponse aux critères d'Elmqvist-Fekete, au prix, toutefois, de la perte de la dimension spatiale dans l'affichage. Il est à noter que les domaines d'action d'Ocelotl et des diagrammes espace-temps sont différents : notre technique représente efficacement une vue synthétique et permet d'interagir et de se focaliser sur des sous-parties de la trace, mais n'est pas capable de représenter une vue détaillée au niveau évènements. À l'inverse, les diagrammes espace-temps ne fournissent pas de vue synthétique satisfaisante, mais s'avèrent indispensables pour une analyse bas niveau suite à celle effectuée à l'aide d'Ocelotl.

5.8.2 Conclusions sur l'analyse

Nous avons montré la pertinence de la technique d'analyse décrite dans ce chapitre grâce à nos cas d'études. La détection d'erreurs et de perturbations dans une application censée avoir un comportement homogène, comme les applications multimédia temps réel, est efficace. Les cas d'études issus d'applications parallèles mettent en évidence la capacité de notre technique à fournir une synthèse du comportement de la trace comparable avec une étude statistique, tout en préservant l'aspect temporel et la détection de phases et d'anomalies ponctuelles. La taille des traces employées dans notre démonstration (jusqu'à 12 Go, 218 millions d'évènements, ou 1500 ressources) confirme les capacités de passage à l'échelle de notre technique en terme d'analyse.

Pour l'analyse d'applications multimédia, d'autres techniques sont envisageables. On peut, par exemple, citer les travaux de Lopez Cueva [99], focalisés sur la reconnaissance de motifs périodiques dans le but de simplifier la trace, ou de Kengne [19] qui propose une méthode de diagnostic fondée sur la comparaison avec des traces de références grâce à l'emploi d'une distance sémantique.

À l'inverse de techniques automatiques comme celles précitées, notre approche visuelle permet une analyse dynamique de systèmes de natures variées, sans la nécessité d'avoir une connaissance à priori sur ces derniers ou la nature des perturbations qui peuvent s'y produire. La part allouée à l'interprétation de l'analyste est importante, ce qui requiert un certain effort cognitif, mais permet l'étude d'une plus grande variété de comportements et d'un champ d'application plus vaste. Cela se confirme par notre analyse d'applications parallèles, dont la séquence des évènements n'est pas périodique. La génération des modèles microscopiques est de complexité linéaire par rapport au nombre d'évènements de la trace. Comme la complexité de l'agrégation dépend uniquement des dimensions de ce modèle microscopique, notre technique possède une meilleure robustesse au passage à l'échelle que des algorithmes de fouilles de données travaillant directement avec les évènements bas niveaux.

Cependant, nous décelons un certain nombre de limitations imposées par notre approche. La discrétisation, décidée par l'analyste, possède une influence sur la détection des perturbations ou des phénomènes de petite échelle contenus dans la trace. Il sera donc potentiellement nécessaire de modifier le nombre de tranches de temps au cours de l'analyse, en fonction de la nature des comportements problématiques recherchés, tout en restant dans les bornes permises par le support de visualisation. Les approches automatiques évoquées précédemment ne possèdent pas cet inconvénient. Dans le cas de programmes dont le comportement attendu est fortement hétérogène ou peu structuré, il semble difficile de tirer des conclusions à partir de notre technique de visualisation. Nous ne proposons pas de modèles microscopiques faisant intervenir plusieurs catégories d'évènements (états, liens, variables) simultanément, alors que cela est le cas dans les diagramme espace-temps. Nous ne proposons, d'ailleurs, pas encore de solutions pour l'analyse des communications, qui représente un aspect important de l'analyse des applications parallèles. Enfin, le fait de ne pas représenter la dimension spatiale empêche l'analyste de lier la dynamique de l'application et les phénomènes temporels à sa structure. Le diagramme espace-temps s'avère souvent inexploitable pour une grande quantité de ressources et ne permet pas d'accomplir cette tâche. Nous verrons dans le Chapitre suivant l'apport d'une représentation spatiotemporelle sur l'analyse de nos cas d'études.

CHAPITRE 6

Agrégation spatiotemporelle de traces d'exécution

Nous avons montré, dans le Chapitre 5, l'intérêt d'une analyse temporelle pour comprendre et déverminer des applications multimédia embarquées. Ces applications se distinguent par un comportement régulier au cours du temps, mais aussi par une structure assez hétérogène des ressources qui ne rend pas les problématiques liées au passage à l'échelle de la dimension spatiale, ni même de sa représentation, prioritaires pour le point d'entrée à l'analyse. Au contraire, les applications MPI qui sont présentées dans le même chapitre se distinguent par une structure spatiale très marquée : il est aisé de construire une hiérarchie en se basant sur les ressources de la plateforme qui sont impliquées dans l'exécution de l'application : le système en entier, les pays, les sites, les clusters, les machines (nœuds), les cœurs et les processus. Nous avons réussi à détecter, dans ce genre d'applications, des phénomènes temporels ponctuels proches des perturbations observées dans les applications multimédia. Cependant, le manque d'une dimension spatiale empêche d'étudier l'impact des ressources sur les performances de l'application. L'organisation structurelle et les différents composants matériels employés aussi bien pour le calcul (fréquence et architecture des processeurs, taille et technologie de la mémoire) que pour les entrées-sorties (technologie des disques durs) ou les communications réseaux (type d'interconnect, configuration des commutateurs) peuvent occasionner un comportement différent sur des sous-parties du système. Cela est exacerbé dans le cas d'un système hétérogène comme Grid'5000 où plusieurs technologies différentes sont mélangées. Ainsi, les mauvaises performances d'un programme peuvent être par exemple causées par une grappe de calcul dont le matériel est d'une ancienne génération, et dont les données sont attendues par les machines d'autres grappes, plus performantes.

Face à cette limitation, nous présentons ici une méthode d'analyse spatiotemporelle, destinée à identifier l'influence de la structure de l'application sur son comportement (aspect spatial), l'évolution du comportement de l'application (aspect temporel), mais aussi à révéler des comportements localisés dans le temps et l'espace (aspect spatiotemporel). De la même manière que pour la technique d'agrégation temporelle, nous prenons comme support la *méthode de Lamarche-Perrin*. Deux dimensions sont représentées, le temps et l'espace des ressources, qui est organisé sous la forme d'une hiérarchie. Cette dernière

peut être la hiérarchie matérielle et logicielle de l'application, ou toute autre hiérarchie laissée au choix de l'analyste et qui peut expliquer le comportement spatiotemporel de l'application. La surface délimitée par les deux dimensions est alors partitionnée, et les parties ainsi générées représentent des zones de l'espace-temps où le comportement de l'application est considéré comme homogène.

La construction du flot d'agrégation est similaire à la technique d'agrégation temporelle : nous élaborons d'abord un modèle microscopique de la trace brute en discrétisant le temps, mais aussi en hiérarchisant les ressources, et concevons un algorithme de partition calculant les gains en complexité et la perte d'information associés à chaque partie spatiotemporelle possible, et fournissant la meilleure coupe du modèle microscopique pour un p donné. Enfin, nous expliquons comment visualiser la sortie de l'algorithme. Nous proposons différentes représentations afin de fournir à l'analyste des informations sur chaque partie, auxquelles nous ajoutons un mécanisme d'agrégation visuel aidant à gérer le passage à l'échelle de la dimension spatiale.

Les travaux décrits dans ce chapitre sont issus de la collaboration avec Robin Lamarche-Perrin, qui a reformalisé le processus d'agrégation spatiotemporelle et l'algorithme que j'ai conçu afin d'assurer la cohérence des concepts et notations avec ses travaux précédents. Ce formalisme n'a cependant été repris que partiellement dans ce Chapitre. Nous avons publié ces travaux dans une conférence internationale [10]. Lucas Schnorr est également co-auteur de cette publication [10], et a participé à l'interprétation des résultats expérimentaux.

6.1 Proposition de modèles microscopiques

De la même manière que pour les modèles microscopiques temporels définis Section 5.1, il est possible de construire des modèles de complexités différentes incluant plus ou moins de dimensions. Voici une définition générale associant une valeur à des coordonnées de l'espace-temps-types d'évènements.

Définition 6.1 Le modèle microscopique spatiotemporel tridimensionnel en fonction des ressources, d'un temps discret et des types d'évènements est une application de $S \times T \times \mathcal{J}(E)$ dans $\mathbb{R}^+$, que l'on note $M^{\mu/st2}$. L'image de (s_k, t_i, J_l) par l'application $M^{\mu/st2}$, avec $s_k \in S$, $t_i \in T$ et $J_l \subset \mathcal{J}(E)$ est noté $\mu^{st2}(s_k, t_i, J_l)$.

L'attribut associé à un couple spatiotemporel (s_k, t_i) est la matrice unidimensionnelle de dimension $(\mathcal{J}(E))$ $\mu^{st2}(s_k, t_i)$:

$$\mu^{st2}(s_k, t_i) = \begin{array}{cccc} \mu^{st2}(s_k, t_i, J_1) & \mu^{st2}(s_k, t_i, J_2) & \cdots & \mu^{st2}(s_k, t_i, J_n)) \end{array} \tag{6.1}$$

La Figure 6.7 représente la structure et le contenu d'un modèle microscopique spatiotemporel où $\mathcal{J}(E)$ est un singleton. Par souci de lisibilité, nous utilisons un gradient de couleur pour représenter l'amplitude de chaque coefficient $\mu^{st1}(s_k, t_i)$. Nous détaillons cette représentation Section 6.4.2.

Les métriques associées aux coefficients des modèles microscopiques spatiotemporels sont les mêmes que pour le modèle temporel, et sont décrites Section 5.2.

6.2 Modèle d'agrégation spatiotemporelle

Dans cette Section, nous définissons et comparons le processus d'agrégation avec d'autres techniques non optimales.

6.2.1 Caractérisation du processus d'agrégation

De la même manière que pour l'agrégation temporelle, nous caractérisons le processus d'agrégation de nos modèles microscopiques, en utilisant les critères définis dans la Section 3.1.2.

Dimensions d'agrégation

On agrège simultanément selon S et T : $S \times T$ est la dimension spatiotemporelle définie comme le produit cartésien des dimensions temporelles T et spatiales S.

Opérandes

Les opérandes sont les individus contenus dans la dimension spatiotemporelle définie par le produit cartésien de la dimension temporelle T et de la dimension spatiale S. Ces individus spatiotemporels sont définis par le couple $(s, t) \in S \times T$.

Contraintes sur l'agrégation

L'élaboration de contraintes sur l'agrégation spatiotemporelle nécessite d'introduire la notion de hiérarchie.

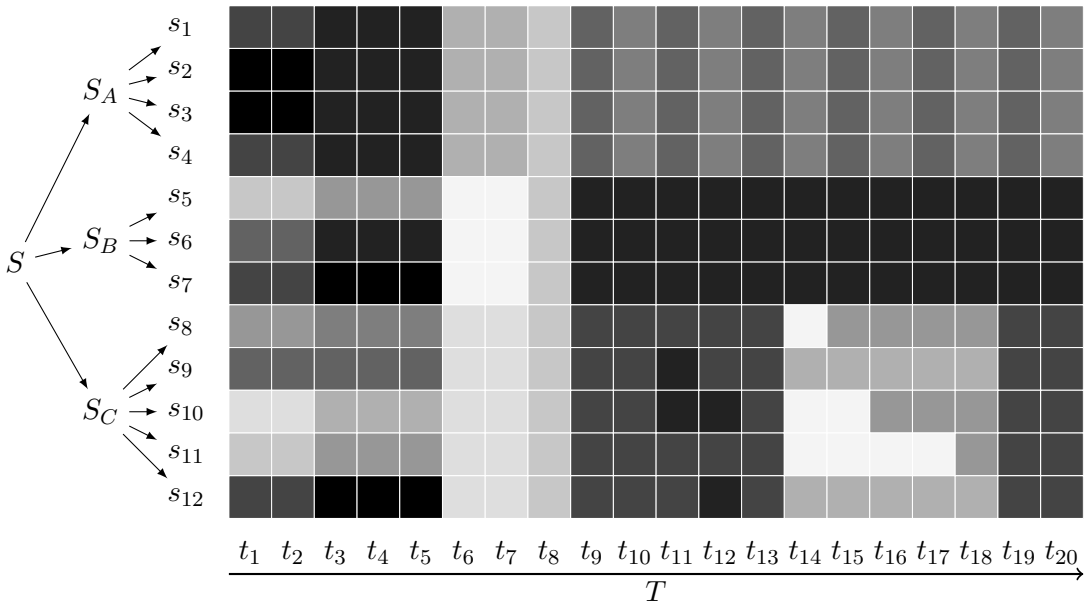


FIGURE 6.1 – Exemple de modèle microscopique représentant une métrique en fonction d'un temps discret et d'une hiérarchie de ressources.

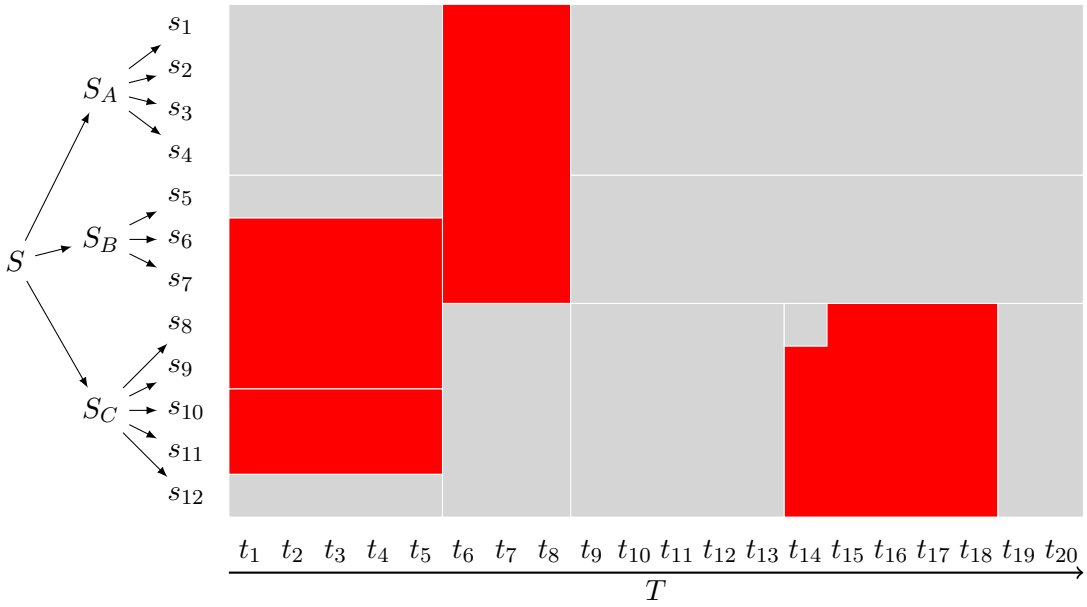


FIGURE 6.2 – Exemple d'agrégation du modèle microscopique décrit par la Figure 6.7 contenant des parties interdites (en rouge).

Définition 6.2 Une hiérarchie $\mathcal{H}(S) = \{S_1, \ldots, S_p\}$ est un ensemble de sous-ensembles de S tel que $\mathcal{H}(S)$ contient l'ensemble S au complet ($S \in \mathcal{H}(S)$), chaque singleton s ($\forall s \in S$, $\{s\} \in \mathcal{H}(S)$), et tel que deux parties de $\mathcal{H}(S)$ soient soit disjointes, soit inclues l'une dans l'autre ($\forall (S_i, S_j) \in \mathcal{H}(S)^2$, soit $S_i \cap S_j = \emptyset$, soit $S_i \subsetneq S_j$, soit $S_i \supsetneq S_j$).

Une hiérarchie est un arbre enraciné, où chaque singleton correspond à une feuille, la racine à l'ensemble complet, et où la relation d'ordre est l'inclusion.

En pratique, cette hiérarchie correspond par exemple à la hiérarchie matérielle (site, cluster, machine, cœur), logicielle (processus, thread), ou hybride (mélangeant les deux hiérarchies) des composants de l'application. Il est donc nécessaire que cette information soit contenue dans la trace ou soit renseignée à posteriori par l'analyste.

Nous formalisons à présent l'ensemble des agrégats spatiotemporels et leurs propriétés.

Définition 6.3 $\mathcal{A}(S \times T) = \mathcal{H}(S) \times \mathcal{I}(T)$ est l'ensemble des agrégats spatiotemporels définis par le produit cartésien d'un nœud $S_k \in \mathcal{H}(S)$ et d'un intervalle de temps $T_{(i,j)} \in \mathcal{I}(T)$.

$S_k \times T_{(i,j)} \in \mathcal{A}(S \times T)$ est le couple définissant un agrégat spatiotemporel du modèle macroscopique. Cette notation est équivalente à $(S_k, T_{(i,j)}) \in \mathcal{H}(S) \times \mathcal{I}(T)$.

La dimension spatiotemporelle hérite des relations d'ordre hiérarchique et du temps : pour chaque intervalle $T_{(i,j)}$, il y a une hiérarchie sur $\{\{s\} \times T_{(i,j)}\}_{s \in S}$, et un ordre total $<$ sur $\{S_k \times \{t\}\}_{t \in T}$ pour chaque nœud S_k.

Enfin, nous définissons l'ensemble des partitions admissibles.

Définition 6.4 Soit $\mathcal{P}(S \times T)$ est une partition spatiotemporelle admissible d'un modèle microscopique. Celle-ci est composée par un sous-ensemble d'agrégats spatiotemporels $(S_k, T_{(i,j)})$ tel que $\forall ((S_a, T_{(b,c)}), (S_l, T_{(m,n)})) \in \mathcal{P}(S \times T)^2$

— soit $(S_a, T_{(b,c)}) = (S_l, T_{(m,n)})$,

— ou soit $(S_a, T_{(b,c)}) \cap (S_l, T_{(m,n)}) = \emptyset$,

avec $b \leq c$, $m \leq n$, et $\bigcup_{(S_k, T_{(i,j)}) \in \mathcal{P}(S \times T)} (S_k, T_{(i,j)}) = S \times T$.

$\mathcal{P}(S \times T)_p$ est une partition spatiotemporelle d'un modèle microscopique obtenue pour un paramètre p donné.

$\boldsymbol{\mathcal{A}}(S \times T)$ est l'ensemble de toutes les partitions spatiotemporelles possibles.

La Figure 6.2 représente des exemples de parties ne respectant pas les contraintes sur l'agrégation. L'agrégat interdit $(S_{10\ \ 11}, T_{(1,5)})$ agrège partiellement des individus du nœud S_C, de même que $(S_{A\ \ B}, T_{(6,8)})$ n'agrège pas totalement S. L'agrégat $(S_{9\ \ 12}, T_{(14)}) + (S_C, T_{(15,18)})$ est également une partie interdite : $(S_8, T_{(14)})$ devrait être impliqué dans l'agrégat. Enfin, $(S_{6\ \ 9}, T_{(1,5)})$ est à cheval sur deux nœuds.

Opération d'agrégation

L'opération d'agrégation est la somme des valeurs des attributs associées aux individus spatiotemporels du modèle microscopique agrégés.

Définition 6.5 Soit $(S_k, T_{(i,j)})$ un agrégat spatiotemporel du modèle microscopique $M^{\mu/st2}$. L'attribut associé à cet agrégat est la matrice de dimension $(\mathcal{J}(E))$ $\mu^{st2}_{sum}(S_k, T_{(i,j)})$ telle que $\mu^{st2}_{sum}(S_k, T_{(i,j)}) = \sum_{(s,t)\in(S_k,T_{(i,j)})} \mu^{st2}(s,t)$.

Nous notons également $\mu^{st2}_{sum}(S_k, T_{(i,j)}, J_l) = \sum_{s,t\in(S_k,T_{(i,j)})} \mu^{st2}(s,t,J_l)$.

Conditions d'agrégation

La condition d'agrégation est la résolution du problème des partitions optimales pour une valeur de p fournie par l'utilisateur (voir Section 4.2). Les Figures 6.3 et 6.4 sont des exemples de partitions optimales pour deux valeurs de p, p_a et p_b, telles que $0 < p_a < p_b < 1$: la réduction de complexité, mais également la perte d'information, croissent entre la première et la deuxième représentation.

6.2.2 Comparaison avec d'autres types d'agrégations

Nous comparons ici notre technique d'agrégation spatiotemporelle avec d'autres types d'agrégations.

Condition d'agrégation ne dépendant pas de l'information

La Figure 6.5 est un exemple d'agrégation spatiotemporelle dont les conditions d'agrégation ne dépendent pas de l'information contenue dans la représentation : le temps est ici partitionné de manière fixe (cinq tranches de temps par agrégat), tandis que l'espace est partitionné au deuxième niveau de la hiérarchie. Cette représentation illustre un phénomène de lissage qui nuit à l'analyse : ainsi, les agrégats $(S_C, T_{(11,15)})$ et $(S_C, T_{(16,20)})$, par exemple, semblent représenter un comportement proche alors que les valeurs associées aux parties qui les composent sont très disparates dans le modèle microscopique montré Figure 6.7. En conclusion, le choix non optimal des partitions modifie l'interprétation des phénomènes contenus dans le modèle microscopique en atténuant les disparités entre les parties. À complexité relativement proche, la Figure 6.4 fournit une représentation plus fidèle du comportement du modèle microscopique.

Combinaison d'agrégations spatiale et temporelle

La Figure 6.6 illustre un exemple de combinaison d'une agrégation temporelle et d'une agrégation spatiale. Chacune d'elles est appliquée séparément sur le modèle microscopique, et leur résultat est ensuite fusionné. Cette technique apporte plus d'information utile que dans le cas de l'agrégation dont les conditions ne dépendent pas de l'information, comme montré par la Figure 6.5, puisqu'elle ne masque pas les disparités. Elle s'avère cependant moins riche que l'agrégation appliquée de manière simultanée sur les deux dimensions, puisque l'ensemble des partitions de cette combinaison d'agrégation est entièrement contenu dans l'ensemble des partitions de l'agrégation spatiotemporelle que nous proposons. $(S_B, T_{(9,20)})$, qui est parfaitement homogène, est ainsi séparé en trois parties à cause de la partition temporelle qui s'applique sur chacun des éléments spatiaux, de même que la relative homogénéité de $(S, T_{(6,8)})$ n'est pas mise en évidence à cause de la partition spatiale qui s'applique sur l'ensemble du temps.

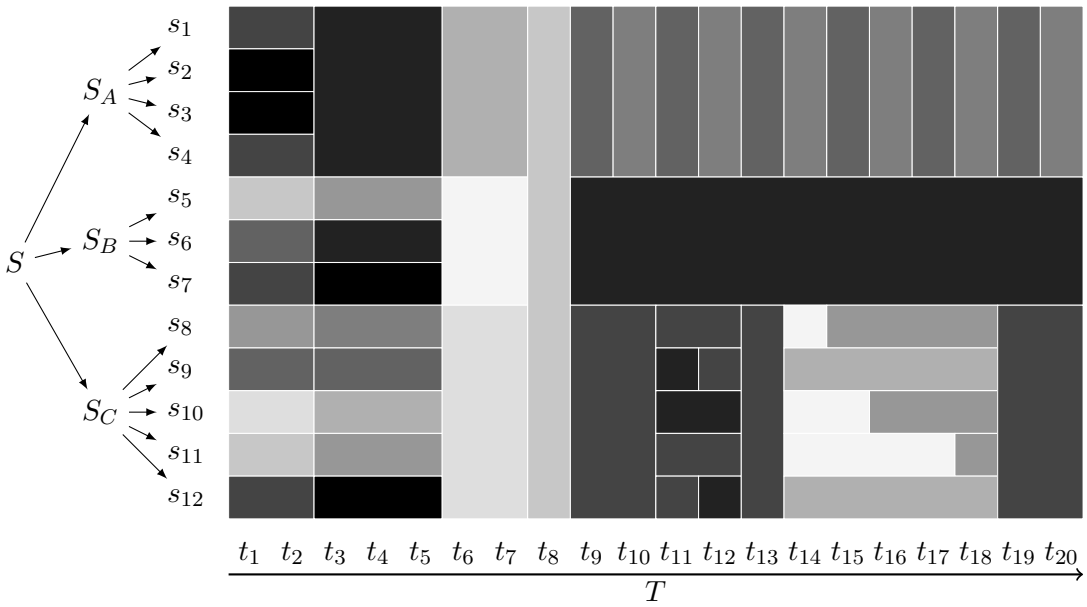


FIGURE 6.3 – Exemple d'agrégation optimale du modèle microscopique décrit par la Figure 6.7, avec p_a tel que $0 < p_a < 1$.

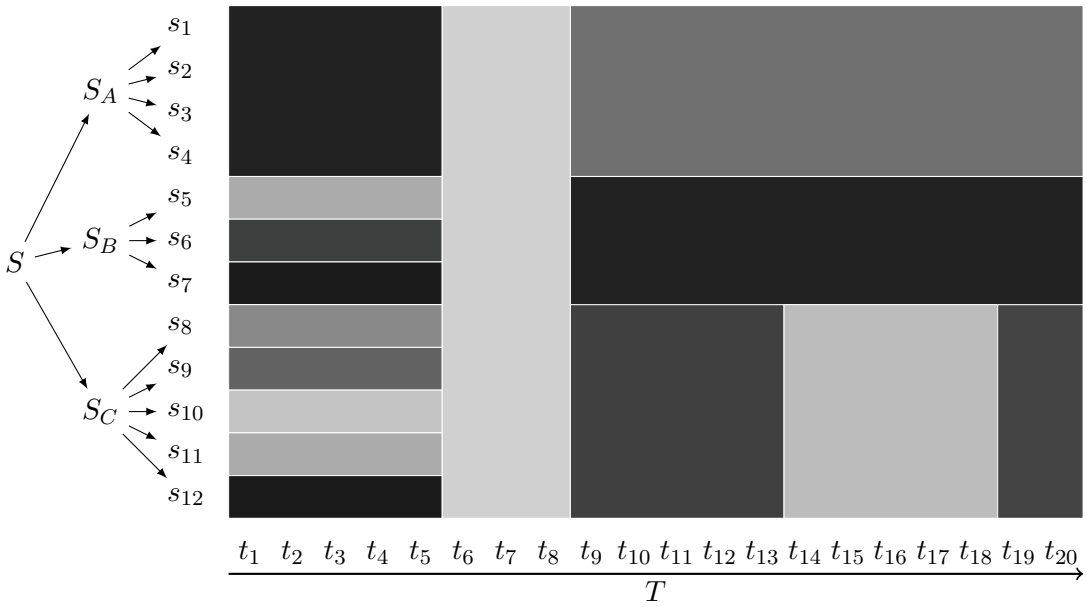


FIGURE 6.4 – Exemple d'agrégation optimale du modèle microscopique décrit par la Figure 6.7, avec p_b tel que $0 < p_a < p_b < 1$.

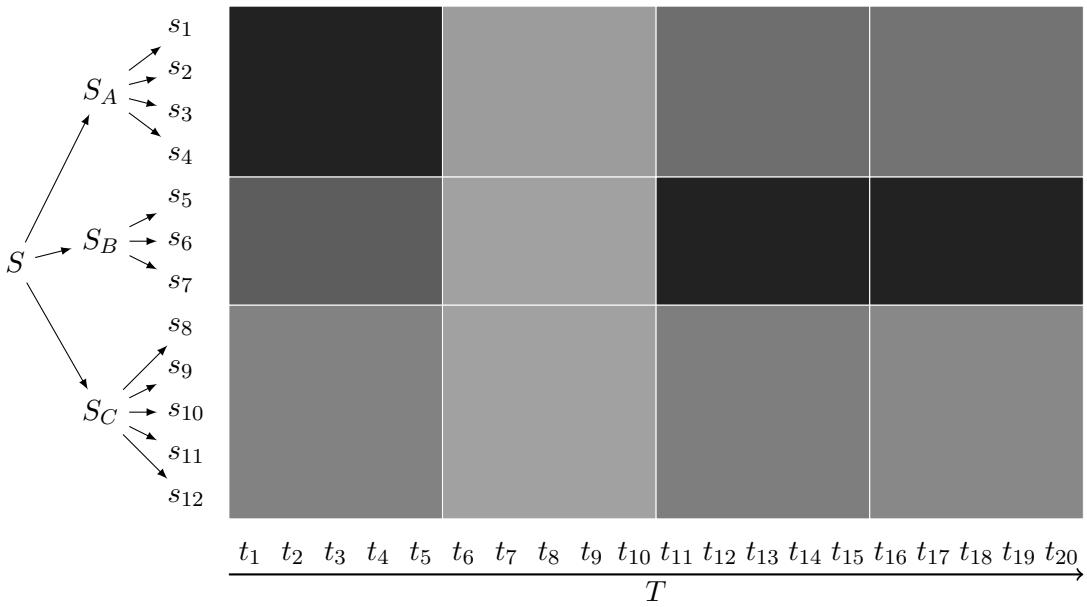


FIGURE 6.5 – Exemple d'agrégation non optimale du modèle microscopique décrit par la Figure 6.7.

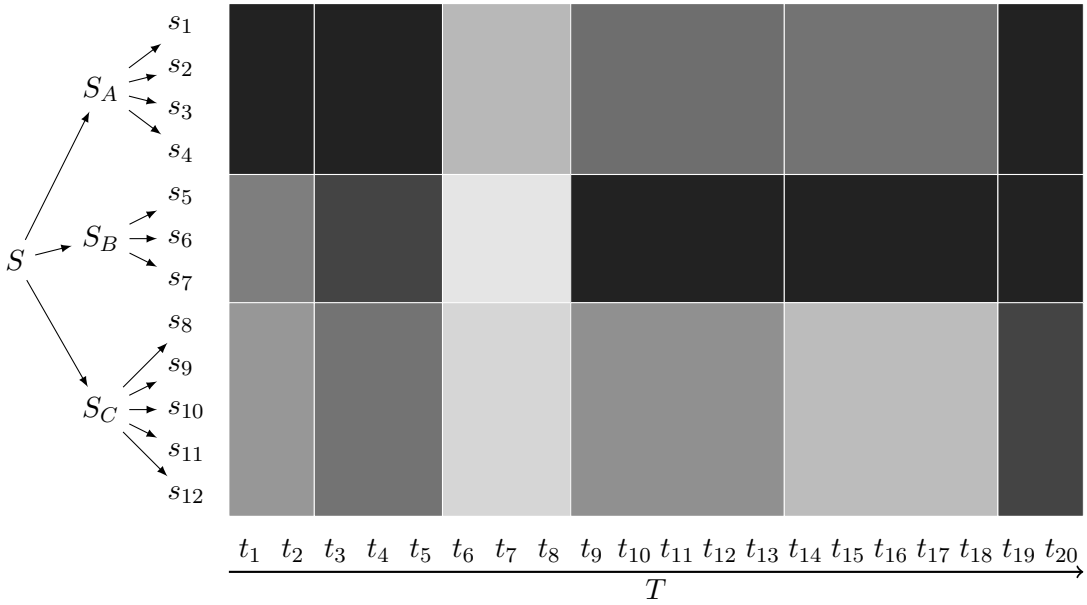


FIGURE 6.6 – Exemple d'agrégation multidimensionnelle, combinant l'agrégation temporelle et l'agrégation spatiale, du modèle microscopique décrit par la Figure 6.7.

6.3 Algorithme des partitions hiérarchiques et ordonnées optimales

De la même manière que pour son équivalent temporel, nous détaillons l'algorithme des partitions hiérarchiques et ordonnées optimales à travers une description de ses structures de données, et l'expression du calcul des qualités et des meilleures coupes.

6.3.1 Structures de données

Nous décrivons l'ensemble des structures de données utilisées par l'algorithme.

Forme globale des structures

Dans l'algorithme d'agrégation spatiale de Lamarche-Perrin évoqué Section 4.4, la hiérarchie $\mathcal{H}(S)$ est modélisée sous la forme d'un arbre de structures de données, où chaque structure est associée à une sous-partie S_k de l'ensemble S. Dans l'algorithme d'agrégation temporelle, l'ensemble des intervalles $\mathcal{I}(T)$ est représenté sous la forme d'une matrice triangulaire supérieure où chaque couple (i, j) correspond à un intervalle $T_{(i,j)}$. Les structures de données de l'algorithme d'agrégation spatiotemporelle regroupent ces deux types de structures de manière à ce que l'ensemble des surfaces $\mathcal{A}(S \times T) = \mathcal{H}(S) \times \mathcal{I}(T)$ soit représenté sous la forme d'un arbre de matrices triangulaires supérieures telles que chaque couple (i, j) de chaque nœud S_k corresponde à une partie $(S_k, T_{(i,j)})$.

Qualités

De même que pour l'algorithme d'agrégation temporel, les qualités associées à chaque agrégat spatiotemporel $(S_k, T_{(i,j)})$ sont les sommes des qualités de leurs coefficients.

Définition 6.6 Entropie de Shannon

$$\text{info}^{\text{st2}}(S_k, T_{(i,j)}, J_l) = \sum_{(s,t)\in(S_k,T_{(i,j)})} \mu^{st2}(s, t, J_l) \log_2 \mu^{st2}(s, t, J_l) \tag{6.2}$$

Définition 6.7 Réduction de complexité

$$\text{gain}^{\text{st2}}(S_k, T_{(i,j)}, J_l) = \mu^{st2}_{sum}(S_k,T_{(i,j)},J_l) \log_2 \mu^{st2}_{sum}(S_k,T_{(i,j)},J_l) \quad \text{info}^{\text{st2}}(S_k,T_{(i,j)},J_l) \tag{6.3}$$

$$\text{gain}^{\text{st2}}(S_k, T_{(i,j)}) = \sum_{J\in\mathcal{J}(E)} \text{gain}^{\text{st2}}(S_k, T_{(i,j)}, J) \tag{6.4}$$

Définition 6.8 Divergence de Kullback-Leibler

$$\text{loss}^{\text{st2}}(S_k, T_{(i,j)}, J_l) = \sum_{(s,t)\in(S_k,T_{(i,j)})} \mu^{st2}(s,t,J_l) \log_2\left(\frac{|S_k||T_{(i,j)}|\mu^{st2}(s,t,J_l)}{\mu^{st2}_{sum}(S_k,T_{(i,j)},J_l)}\right) \tag{6.5}$$

$$\text{loss}^{\text{st2}}(S_k, T_{(i,j)}) = \sum_{J\in\mathcal{J}(E)} \text{loss}^{\text{st2}}(S_k, T_{(i,j)}, J)) \tag{6.6}$$

Structure « trade-off »

On emploie la structure tradeoff décrite Section 5.5.1. On note tradeoff$(S_k, T_{(i,j)})$ la structure associée à la zone définie par $(S_k, T_{(i,j)})$.

6.3.2 Sortie de l'algorithme

Chaque partition $\mathcal{P}(S \times T) \in \boldsymbol{\mathcal{A}}(S \times T)$ peut être représentée comme une séquence de coupes imbriquées de surfaces spatiotemporelles appartenant à $\mathcal{A}(S \times T)$. On distingue les coupes spatiales, où la surface $(S_k, T_{(i,j)})$ est partitionnée au niveau hiérarchique directement inférieur $\{(S_{k_1}, T_{(i,j)}), \ldots, (S_{k_q}, T_{(i,j)})\}$, où $S_{k_1}, \ldots, S_{k_q}$ sont les enfants de S_k, et les coupes temporelles, où les surfaces sont partitionnées en deux intervalles $\{(S_k, T_{(i,\text{cut})}), (S_k, T_{(\text{cut}+1,j)})\}$, avec $i \leq \text{cut} < j$. Il est à noter qu'une séquence de coupes ne peut définir qu'une seule partition $\mathcal{P}(S \times T)$, alors qu'une partition $\mathcal{P}(S \times T)$ peut être définie par différentes séquences[1]. L'algorithme d'agrégation spatiotemporelle calcule récursivement, pour chaque intervalle $T_{(i,j)}$ de chaque nœud S_k, une valeur de coupe définissant une partition optimale de la zone correspondante :

- $\text{cut}(S_k, T_{(i,j)}) = \quad 1$ indique une coupe spatiale de $(S_k, T_{(i,j)})$;
- $\text{cut}(S_k, T_{(i,j)}) = x \in \{i, \ldots, j \quad 1\}$ indique une coupe temporelle, où x est l'indice de la période de temps microscopique à la fin de laquelle se situe la coupe ;
- $\text{cut}(S_k, T_{(i,j)}) = j$ indique que $(S_k, T_{(i,j)})$ est un agrégat de la partition (pas de découpage).

Après exécution de l'algorithme, la partition optimale résultante est obtenue grâce à la séquence des coupes. On part de $(S_{root}, T_{(1,|T|)})$ et on regarde récursivement les sous-parties jusqu'à ce que chaque agrégat de la partition soit atteint.

6.3.3 Description de l'algorithme

Nous décrivons ici les différentes étapes de l'algorithme d'agrégation spatiotemporelle. Nous réutilisons l'ensemble des fonctions annexes décrites par les algorithmes 5.1.

Calcul des qualités

La première étape, décrite par l'algorithme 6.1 consiste à calculer les qualités (réduction de complexité et perte d'information) associées à chaque agrégat. Elle consiste principalement en plusieurs boucles imbriquées, où l'on réutilise le résultat des itérations précédentes.

[1] Néanmoins, on définit un ordre sur les coupes dans l'algorithme afin qu'une partition soit toujours représentée par la même séquence.

Algorithme 6.1 Algorithme de calcul des qualités

```
1: procedure COMPUTEQUALITIES(S_k)
2:   if S_k ∉ S then                                   ▷ Si S_k n'est pas une feuille de H(S)
3:     for S_{k_q} ⊊⃗ S_k do                            ▷ ⊊⃗ signifie descendant direct de
4:       COMPUTEQUALITIES(S_{k_q})
5:     end for
6:     for i = 1, ..., |T| do
7:       for j = i, ..., |T| do
8:         gain^{st2}[k, i, j] = 0
9:         loss^{st2}[k, i, j] = 0
10:        for l = 1, ..., |J(E)| do
11:          μ^{st2}_{sum}[k, i, j, l] = 0
12:          info^{st2}[k, i, j, l] = 0
13:          for S_{k_q} ⊊⃗ S_k do
14:            μ^{st2}_{sum}[k, i, j, l] += μ^{st2}_{sum}[k_q, i, j, l]
15:            info^{st2}[k, i, j, l] += info^{st2}[k_q, i, j, l]
16:          end for
17:          gain^{st2}[k, i, j] += ENTROPYREDUCTION(μ^{st2}_{sum}[k, i, j, l], info^{st2}[k, i, j, l])
18:          loss^{st2}[k, i, j] += DIVERGENCE((j  i + 1) × |S_k|, μ^{st2}_{sum}[k, i, j, l], info^{st2}[k, i, j, l])
19:        end for
20:      end for
21:    end for
22:  else
23:    for i = 1, ..., |T| do
24:      gain^{st2}[k, i, i] = 0
25:      loss^{st2}[k, i, i] = 0
26:      for l = 1, ..., |J(E)| do
27:        μ^{st2}_{sum}[k, i, i, l] = μ^{st2}[k, i, l]
28:        info^{st2}[k, i, i, l] = ENTROPYREDUCTION(μ^{st2}[k, i, l], 0)
29:      end for
30:    end for
31:    for i = |T|, ..., 1 do
32:      for j = i, ..., |T| do
33:        gain^{st2}[k, i, j] = 0
34:        loss^{st2}[k, i, j] = 0
35:        for l = 1, ..., |J(E)| do
36:          μ^{st2}_{sum}[k, i, j, l] = μ^{st2}_{sum}[k, i + 1, j, l] + μ^{st2}_{sum}[k, i, i, l]
37:          info^{st2}[k, i, j, l] = info^{st2}[k, i + 1, j, l] + info^{st2}[k, i, i, l]
38:          gain^{st2}[k, i, j] += ENTROPYREDUCTION(μ^{st2}_{sum}[k, i, j, l], info^{st2}[k, i, j, l])
39:          loss^{st2}[k, i, j] += DIVERGENCE((j  i + 1), μ^{st2}_{sum}[k, i, j, l], info^{st2}[k, i, j, l])
40:        end for
41:      end for
42:    end for
43:  end if
44: end procedure
```

Calcul des meilleures coupes

Algorithme 6.2 Algorithme de calcul des meilleures coupes.

```
1: procedure COMPUTEBESTCUTS(S_k, p)
2:     if S_k ∉ S then
3:         for S_{k_q} ⊊⃗ S_k do
4:             COMPUTEBESTCUTS(S_{k_q}, p)
5:         end for
6:     end if
7:     for i = |T|, ..., 1 do
8:         for j = i, ..., |T| do
9:             cut[k, i, j] = j
10:            tradeoff[k, i, j] = COMPUTEPIC(p, gain^st[k, i, j], loss^st[k, i, j])
11:            if S_k ∉ S then
12:                tradeoff_s =INIT()
13:                for S_{k_q} ⊊⃗ S_k do                                  ▷ Coupe spatiale
14:                    tradeoff_s += tradeoff[k_q, i, j]
15:                end for
16:                if tradeoff_s > tradeoff[k, i, j] then
17:                    cut[k, i, j] =   1
18:                    tradeoff[k, i, j] = tradeoff_s
19:                end if
20:            end if
21:            for cut_t = i, ..., j   1 do                              ▷ Coupe temporelle
22:                compromise_t = tradeoff[k, i, cut_t] + tradeoff[k, cut_t +1, j]
23:                if tradeoff_t > tradeoff[k, i, j] then
24:                    cut[k, i, j] = cut_t
25:                    tradeoff[k, i, j] = tradeoff_t
26:                end if
27:            end for
28:        end for
29:    end for
30: end procedure
```

La seconde étape est le calcul des meilleures coupes associées à la partition optimale, décrit par l'algorithme 6.2. Elle consiste en une succession d'itérations imbriquées dans une récursion pour calculer alternativement les coupes spatiales et temporelles. On commence la récursion depuis le sommet de la hiérarchie, c'est à dire le nœud S_{root}.

Coupe spatiale : pour chaque zone spatiotemporelle $(S_k, T_{(i,j)})$, le $\mathrm{pIC}(S_k, T_{(i,j)})$ de l'agrégat correspondant est comparé avec la somme des $\mathrm{pIC}(S_{k_q}, T_{(i,j)})$ associés aux partitions optimales des enfants S_{k_q} de S_k (qui ont été calculées précédemment récursivement). Un découpage spatial est détecté si $\mathrm{pIC}(S_k, T_{(i,j)}) < \sum_{S_{k_q} \in S_k} \mathrm{pIC}(S_{k_q}, T_{(i,j)})$.

Coupe temporelle : on compare $\mathrm{pIC}(S_k, T_{(i,j)})$ avec la somme des pIC des partitions optimales $\mathrm{pIC}(S_k, T_{(i,\mathrm{cut})}) + \mathrm{pIC}(S_k, T_{(\mathrm{cut}+1,j)})$ obtenues pour chaque coupe $\mathrm{cut} \in \{i, \cdots, j \quad 1\}$ possible, et calculées au cours des itérations précédentes. On choisit cut tel que le pIC associé soit le plus grand. De cette manière, toutes les coupes possibles sont évaluées, et donc toutes les partitions possibles.

Calcul des meilleures partitions

De même, nous employons aussi un algorithme de calcul des partitions optimales fournissant une liste de paramètres p pour lesquels les partitions optimales associées sont différentes, construit sur le même principe que celui évoqué en Section 5.5.3.

6.3.4 Complexité théorique de l'algorithme

Complexité spatiale

La complexité spatiale du calcul des qualités est une fonction linéaire du nombre de nœuds $\mathcal{H}(S)$, multiplié par le nombre d'agrégats temporels possibles ($|T|^2$) multiplié par le nombre de types d'évènements ($|\mathcal{J}(E)|$). Elle est donc $\mathcal{O}(|\mathcal{H}(S)| \times |T|^2 \times |\mathcal{J}(E)|)$.

De même, la complexité spatiale du calcul des meilleures coupes est une fonction linéaire du nombre de nœuds multiplié par le nombre d'agrégats temporels possibles. Elle est donc $\mathcal{O}(|\mathcal{H}(S)| \times |T|^2)$.

Complexité temporelle

La complexité temporelle du calcul des qualités est une fonction linéaire de la somme du nombre d'itérations, égale au nombre d'agrégats temporels possibles ($|T|^2$), pour chaque nœud ($|\mathcal{H}(S)|$), et pour chaque type d'évènements ($|\mathcal{J}(E)|$). Elle est donc $\mathcal{O}(|\mathcal{H}(S)| \times |T|^2 \times |\mathcal{J}(E)|)$.

La complexité temporelle du calcul des meilleures coupes est une fonction linéaire du nombre de nœuds dans la hiérarchie (on effectue une addition et une comparaison pour chaque nœud). Pour un nœud donné, pour chaque intervalle $T_{(i,j)}$, avec $1 \leq i \leq j < |T|$, l'algorithme effectue $j \quad i$ itérations pour évaluer la meilleure coupe temporelle sur cet intervalle. Ainsi, $\sum_{n=1}^{|T|} i(|T| \quad i) = \mathcal{O}(|T|^3)$ itérations par nœud sont effectuées, ce qui équivaut à une complexité temporelle totale de $\mathcal{O}(|\mathcal{H}(S)| \times |T|^3)$. Le calcul des meilleures partitions possède une complexité qui est, dans le pire cas, $\mathcal{O}(\log_2(\mathrm{seuil}) \times |\boldsymbol{\mathcal{A}}(S \times T)| \times |\mathcal{H}(S)| \times |T|^3)$. On distingue l'influence du nombre de comparaisons entre les valeurs de p, limité par le seuil, $\mathcal{O}(\log_2(\mathrm{seuil}))$, une borne supérieure délimitée par le nombre maximal de partitions optimales possibles $\mathcal{O}(|\boldsymbol{\mathcal{A}}(S \times T)|)$, qui limite la descente dans la récursion, et la partie associée au calcul des qualités de chaque partition évaluée $\mathcal{O}(|\mathcal{H}(S)| \times |T|^3))$.

6.4 Visualisation d'une partition du système

De la même manière que dans la Section 5.6, nous détaillons ici les visualisations de la sortie de l'agrégation spatiotemporelle. Toutes ces visualisations sont bâties autour d'une représentation bidimensionnelle, où l'axe des abscisses est associé à la dimension temporelle T et l'axe des ordonnées à la hiérarchie $\mathcal{H}(S)$.

6.4.1 Partition

Cette visualisation simple représente la partition $\mathcal{P}(S \times T)$ sans montrer les valeurs de chaque agrégat. Elle consiste en une représentation « matricielle » où chaque partie est associée à un rectangle $rect(S_k, T_{(i,j)})$, placé sur un repère cartésien, dont l'axe des abscisses est associé au temps et celui des ordonnées à l'espace des ressources. Les caractéristiques de ce rectangle (hauteur, largeur, position) dépendent des individus spatiaux et temporels agrégés par l'agrégat associé.

De même que pour la partition temporelle, décrite Section 5.6.2, on utilise la couleur pour distinguer visuellement les agrégats. On crée une séquence de couleur $\overrightarrow{\text{couleur}}$ de dimension $S \times T$. On attribue à chaque agrégat $(S_k, T_{(i,j)})$ de la partition $\mathcal{P}(S \times T)$ un identifiant $\mathrm{id}\,(S_k, T_{(i,j)}) \in \mathbb{N}^+$ tel que $0 < \mathrm{id}\,(S_k, T_{(i,j)}) < |S| \times |T|$, et on lui associe la couleur $\text{couleur}(\mathrm{id}(S_k, T_{(i,j)}))$.

6.4.2 Amplitude relative

L'amplitude relative est l'équivalent spatiotemporel de l'histogramme décrit Section 5.6.3, dans la mesure où il permet de représenter aisément des valeurs scalaires (modèle microscopique $M^{\mu/st1}$). On utilise la même structure (rectangles symbolisant les agrégats, leur taille et leur position) que pour la partition spatiotemporelle que nous venons de décrire. Néanmoins, nous n'utilisons qu'une seule couleur appliquée à l'ensemble de tous les agrégats. Pour représenter la valeur des agrégats, on utilise le *canal alpha* du format de couleur RGBA permettant de gérer la transparence, afin de modifier l'intensité de cette couleur en fonction de la valeur associée à l'agrégat.

Les Figures 6.3 et 6.4 représentent deux niveaux d'agrégation de complexité décroissante du modèle microscopique décrit par la Figure 6.7. Tous les trois sont des exemples de l'amplitude relative.

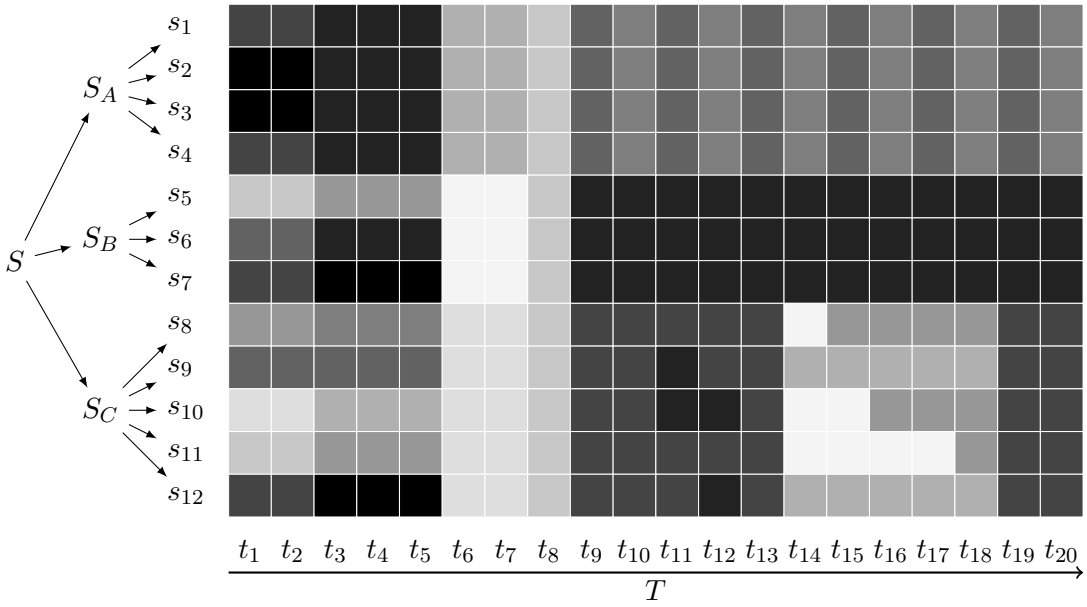
6.4.3 Mode

L'amplitude relative n'est pas adéquate à l'analyse du modèle microscopique $M^{\mu/st2}$ contenant plusieurs types d'évènement. De plus, à la différence de l'histogramme empilé employé pour l'agrégation temporelle Section 5.6.3, il n'est pas possible de représenter simultanément l'ensemble des types d'évènements pour chaque agrégat du fait d'une surface allouée à chacun d'eux potentiellement trop faible. Nous choisissons de visualiser seulement une partie de cette information à travers le *mode*, c'est à dire en attribuant à l'agrégat la couleur $\text{couleur}(J_{mode})$ associée au type d'évènement $J_{\text{mode}} \in \mathcal{J}(E)$ dont l'amplitude est la plus grande. La transparence utilisée pour l'amplitude relative peut aussi être employée pour ajouter de l'information au mode. La Figure 6.7 représente un modèle microscopique à deux types d'évènements différents (noirs et rouges), dont les valeurs associées à chaque individu spatiotemporel sont complémentaires. La Figure 6.8 montre le mode de ce modèle microscopique, tandis que la Figure 6.9 en montre une agrégation.

6.4.4 Agrégation visuelle

La partition, l'amplitude relative et le mode souffrent d'un même défaut structurel. Le passage à l'échelle de la dimension temporelle est assuré par l'opération de discrétisation

Type d'évènements noir :



Type d'évènements rouge :

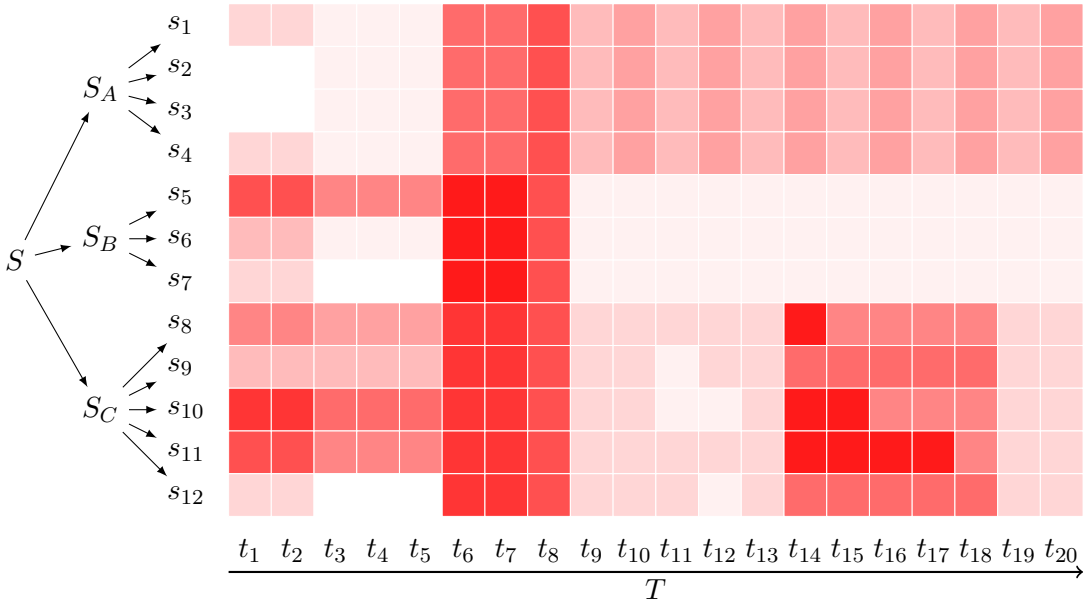


FIGURE 6.7 – Exemple de visualisation d'un modèle microscopique à deux types d'évènements (noir, en haut et rouge, en bas), dont la transparence traduit la valeur.

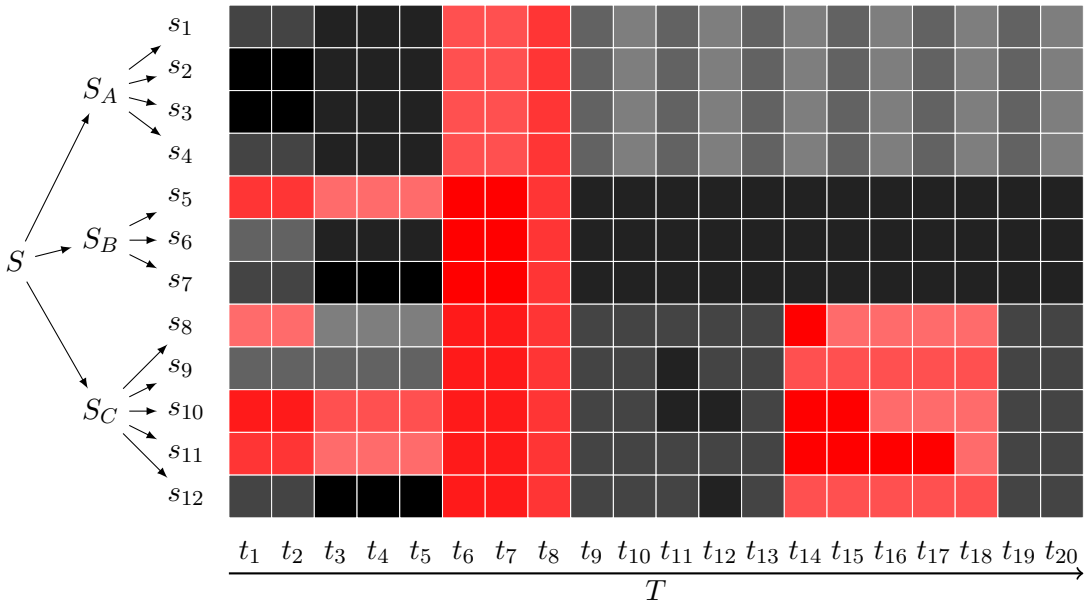


FIGURE 6.8 – Exemple de visualisation du modèle microscopique décrit par la Figure 6.7 représentant le mode (avec transparence) d'une métrique en fonction d'un temps discret et d'une hiérarchie de ressources.

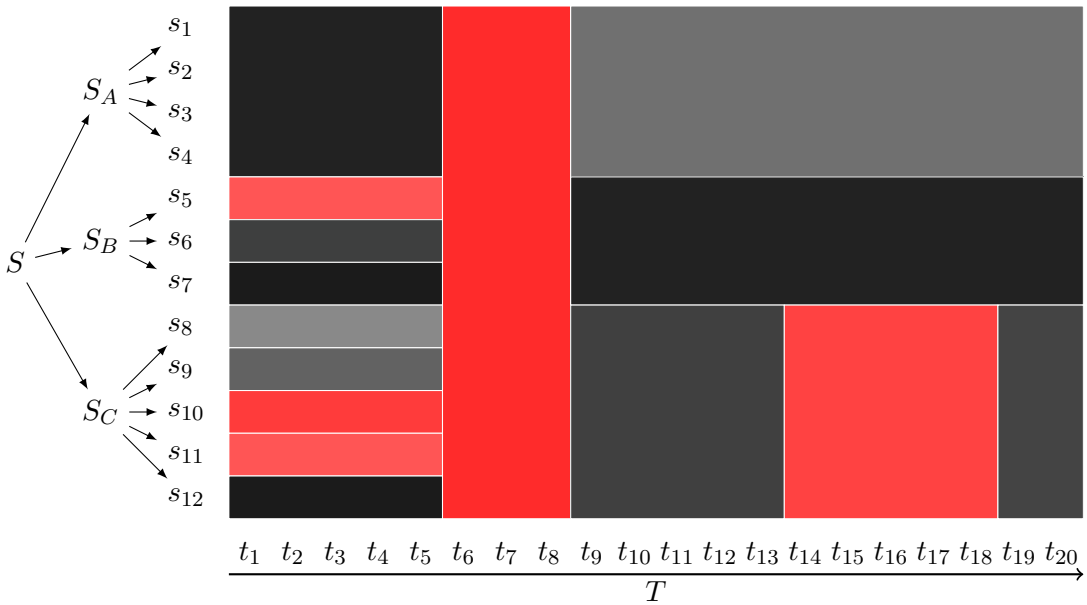


FIGURE 6.9 – Exemple de visualisation d'une agrégation optimale du modèle microscopique décrit par la Figure 6.7 représentant le mode (avec transparence).

temporelle qui permet de borner le nombre de tranches de temps de manière à leur allouer une taille en pixel raisonnable. Cependant, le passage à l'échelle de la dimension spatiale est quant à lui dépendant du résultat fourni par l'agrégation de données qui est appliquée sur le modèle microscopique. Ainsi, il est possible que pour des modèles microscopiques contenant une grande quantité de ressources, un certain nombre de partitions n'agrègent pas suffisamment la dimension spatiale, conduisant à ce que la taille allouée aux éléments spatiaux soit inférieure à un pixel et que ces derniers ne puissent être représentés de manière satisfaisante. Deux solutions s'offrent à nous : agréger les niveaux les plus bas de la hiérarchie à priori, ou appliquer une agrégation visuelle à posteriori. La première solution est préférable dans les cas où la taille de la dimension spatiale ou la profondeur de la hiérarchie sont si conséquentes que cela a un impact sur le temps de calcul de l'algorithme d'agrégation ou l'utilisation de la mémoire, ou que plusieurs niveaux de la hiérarchie sont impossibles à représenter à cause de leur taille. L'utilisateur fournit un seuil maximal de feuilles, et les niveaux les plus profonds de la hiérarchie des ressources sont agrégés de manière à satisfaire à cette condition, pour chaque tranche de temps et chaque type d'évènements. Bien que cette technique provoque une perte d'information, elle reste une alternative satisfaisante face à l'absence de possibilité d'analyse. Elle est, de plus, cohérente avec le processus de construction du modèle microscopique : on peut en effet la voir comme le pendant spatial de la discrétisation temporelle que nous employons. La deuxième solution est plus adaptée à des cas où seul un niveau partiel ou entier de la hiérarchie n'est pas représentable, puisqu'elle conserve de l'information sur l'hétérogénéité de ces éléments spatiaux.

On caractérise le processus de la manière suivante :

— Cette agrégation est appliquée sur les **dimensions spatiales** et **temporelles** ;
— Les **opérandes** sont les agrégats de données définis par la partition $\mathcal{P}(S \times T)$;
— Les **parties admissibles** sont les mêmes que celles définies pour l'agrégation de données : $\boldsymbol{\mathcal{A}}(S \times T)$;
— L'**opération** d'agrégation est la somme. On distingue l'agrégation visuelle de l'agrégation de données par l'utilisation de symboles ;
— La **condition** d'agrégation est la hauteur d'une partie $(S_k, T_{(i,j)})$. Si cette dernière est inférieure à un certain seuil, on agrège le nœud à l'intérieur duquel elle est contenue, et ce, sur l'intervalle de temps $T_{(i,j)}$.

On distingue deux situations, représentées par la Figure 6.10 : si un ensemble d'agrégats de données à agréger visuellement au sein du même nœud possède exactement le même découpage temporel (début et fin, sans coupure intermédiaire), cela génère des agrégats dits de type A. À contrario, s'il existe des coupures intermédiaires non partagées par l'ensemble des agrégats de données à agréger au sein d'un nœud, cela génère des agrégats dits de type B.

6.5 Analyse spatiotemporelle

Dans cette section, nous reprenons les cas d'études d'applications parallèles présentés dans la Section 5.7.2 pour les analyser avec notre technique d'agrégation spatiotemporelle, également implémentée dans Ocelotl. Nous avons deux objectifs. D'une part, vérifier que le comportement temporel de l'application est toujours retranscrit, et est cohérent avec nos observations dans le Chapitre précédent. D'autre part, mettre en relief des com-

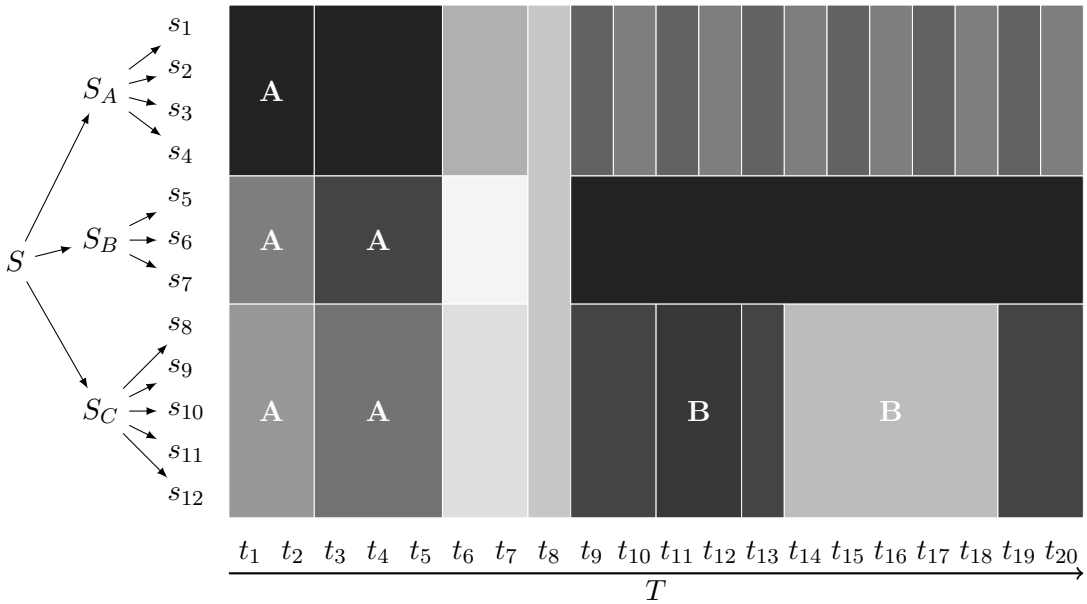


FIGURE 6.10 – Exemple d'agrégation visuelle appliquée sur la visualisation de la Figure 6.3 : les agrégats visuels de type A indiquent une désagrégation spatiale, mais une partition temporelle des agrégats de données sous-jacents identique, les agrégats visuels de type B, une désagrégation spatiale, et une partition temporelle des agrégats sous-jacents sans coupure commune.

portements spatiotemporels impliquant la structure de l'application, et non détectables avec notre technique d'agrégation temporelle.

6.5.1 Applications parallèles

Nous analysons de nouveau les cas d'études synthétisés dans la Table 5.2. Nous organisons les ressources au sein d'une hiérarchie hybride matérielle et logicielle comme il suit : *site* > *grappe* > *machine* > *processus*. Chaque processus est fixé sur un cœur de la machine sur laquelle il est exécuté. La Figure 6.11 rappelle l'association des appels MPI avec les couleurs employées par Ocelotl et Framesoc pour les représenter. La Figure 6.12 montre la représentation des différents types d'agrégats visuels dans Ocelotl.

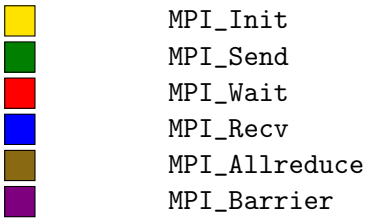


FIGURE 6.11 – Association entre les principaux types d'évènements contenus dans les traces issues des cas d'études *NASPB* et la couleur employée par Ocelotl et Framesoc pour les représenter.

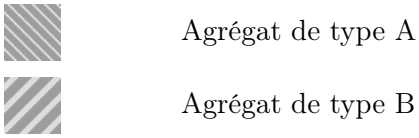


FIGURE 6.12 – Représentation des agrégats visuels de types A et B.

Gradient conjugué

Lors de l'analyse avec la technique d'agrégation spatiotemporelle des traces CG, nous retrouvons le comportement temporel mis en évidence dans le chapitre précédent : les phases d'initialisation et de calcul sont toujours apparentes, comme le montre les Figures 6.13 (CG.C-Ren-64) et 6.14 (CG.C-Gre-64). Dans le premier cas, on retrouve également les perturbations survenant entre 30 et 40 secondes, bien que moins marquées dans la représentation. L'aspect spatial apporte des informations supplémentaires : il apparaît que ces perturbations concernent l'ensemble des grappes, tout en épargnant certaines machines à l'intérieur de chacune d'elles. Pour CG.C-Gre-64, on distingue un déséquilibre entre *adonis* et *edel*, qui ont respectivement une proportion de 47 et 46 % de temps passé dans `MPI_Wait` et 32 % dans `MPI_Send` dans la phase de calcul (en comptant le temps passé dans l'application), et *genepi*, qui a 4 % seulement de temps passé dans `MPI_Wait` et 18 % dans `MPI_Send`[2]. L'analyse avec un diagramme espace-temps montre que les calculs prennent plus de temps sur *genepi* que sur les autres grappes. Les processus s'exécutant sur *adonis* et *edel* restent alors bloqués dans `MPI_Wait` le temps que les résultats de la phase de calcul soient partagés par l'ensemble des processus. Cette différence de comportement peut s'expliquer par les processeurs utilisés (Intel Xeon E5520 sur *adonis* et *edel* contre Intel Xeon E5420 QC sur *genepi*).

Décomposition LU

De même, les analyses de l'application LU corroborent le comportement temporel, au niveau des phases de calcul, et des anomalies temporelles, par exemple dans le cas de LU.C-Nan-700, Figure 6.15. Mais là encore, l'analyse spatiotemporelle se révèle capable d'apporter des informations inaccessibles avec la représentation exclusivement temporelle. Ainsi, dans ce même cas, on distingue que la grappe *graphite* possède un comportement spatialement et temporellement hétérogène. De plus, on découvre que la perturbation temporelle est limitée à la seule grappe *griffon*. Les Figures 6.16 et 6.17 montrent un focus sur ces deux perturbations. Dans le cas de *graphite*, plusieurs facteurs sont potentiellement responsables du comportement spatiotemporel : son isolement, au niveau réseau, des deux autres grappes (communication par Ethernet), ou encore la présence de serveurs de stockage sur le même commutateur et pouvant être accédés pendant l'exécution de l'application par des utilisateurs tiers peuvent expliquer ces irrégularités. Dans le cas de *griffon*, la perturbation étant ponctuelle, nous supputons un accès concurrent au réseau avec une autre application exécutée par un autre utilisateur.

La Figure 6.18 représente aussi la visualisation de la trace LU.C-Nan-700, mais les niveaux intermédiaires *grappes* et *machines* ont été supprimés. Lors de cette analyse,

[2]L'interaction proposée par Ocelotl permet d'obtenir ces informations statistiques en cliquant sur un agrégat. Nous présentons cette technique dans le Chapitre suivant.

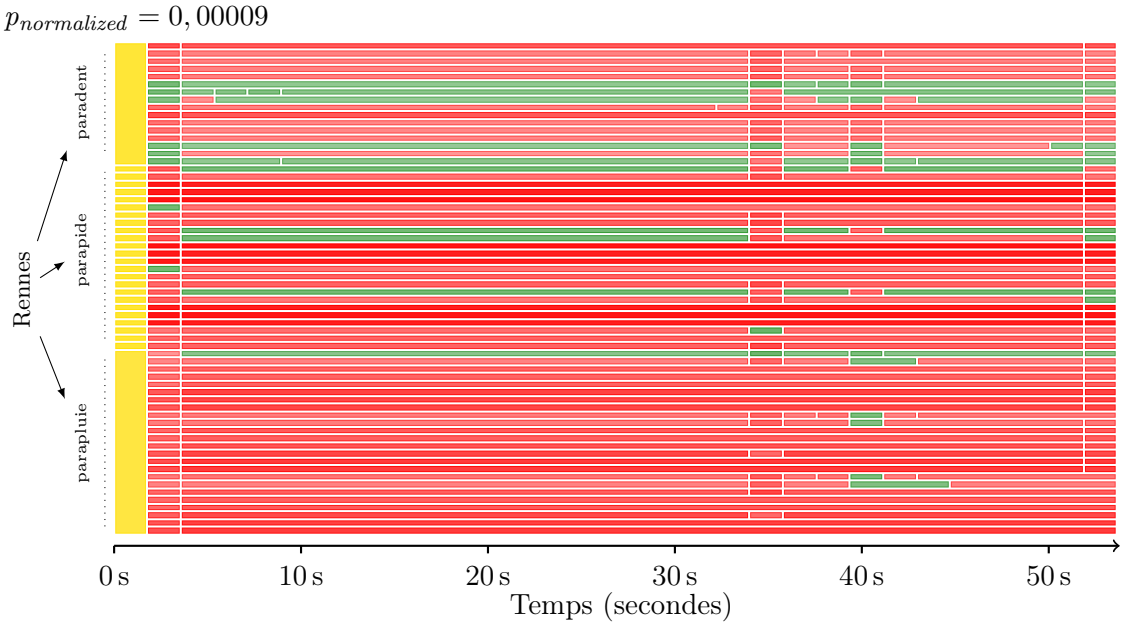


FIGURE 6.13 – Partition spatiotemporelle de CG.C-Ren-64 montrant le mode (ratio d'activité). Les perturbations temporelles, illustrées lors de l'analyse temporelle par la Figure 5.31, sont toujours détectées. L'analyse spatiotemporelle permet de voir leur incidence sur l'ensemble des grappes.

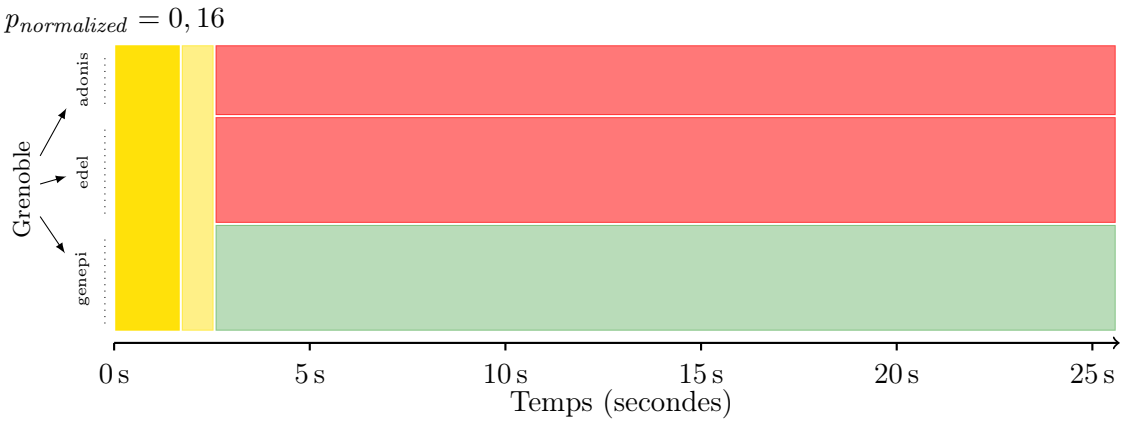


FIGURE 6.14 – Partition spatiotemporelle de CG.C-Gre-64 montrant le mode (ratio d'activité). La visualisation, en plus du découpage en phase décelé lors de l'analyse temporelle, montre que les grappes ont un comportement différent.

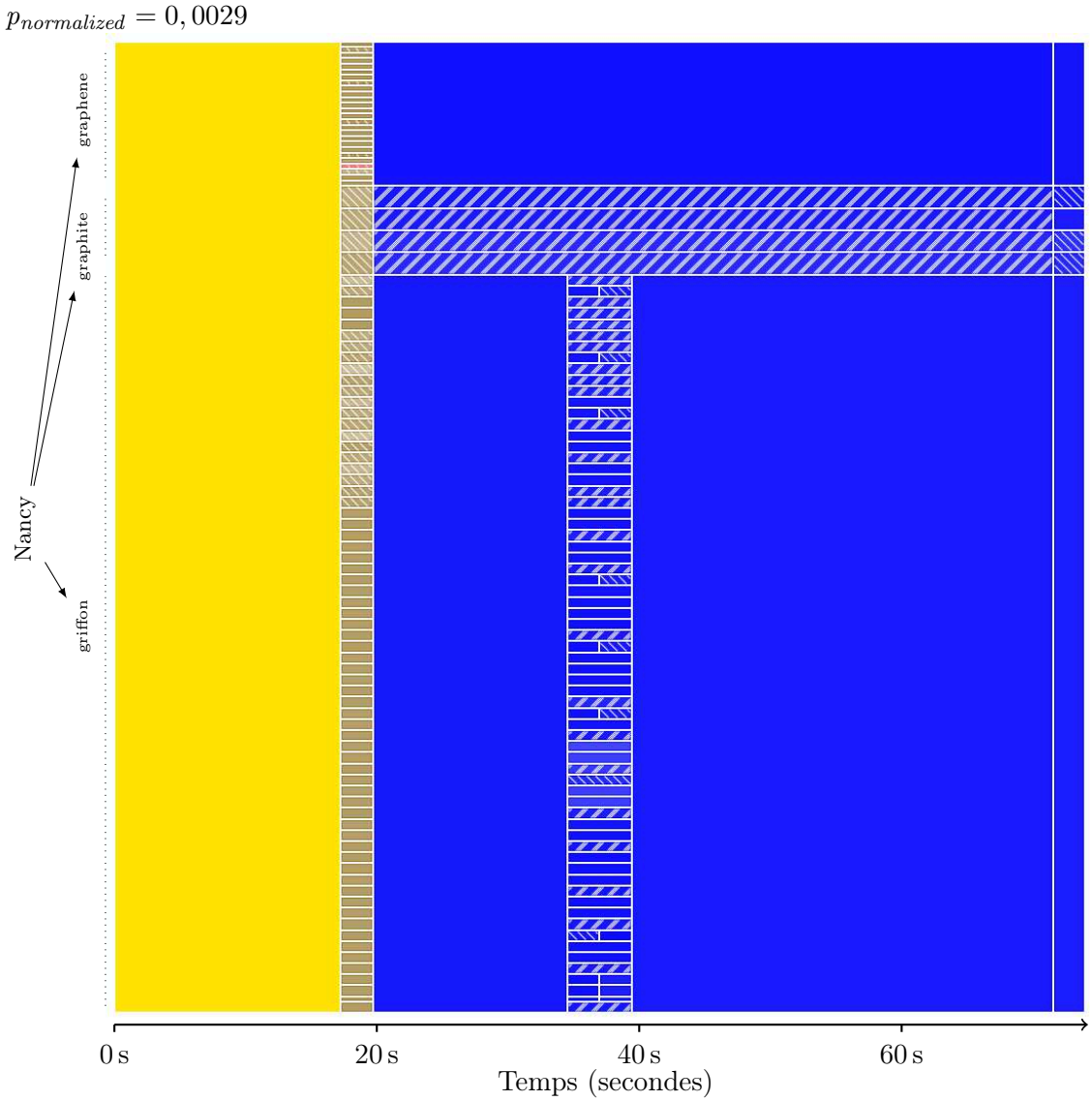


FIGURE 6.15 – Partition spatiotemporelle de LU.C-Nan-700 montrant le mode (ratio d'activité). En plus du découpage en phase (initialisation, de 0 à 20 secondes, suivi de la phase de calcul), deux comportements problématiques émergent dans la phase de calcul : une perturbation temporelle touche la grappe *griffon* aux alentours de 38 secondes, tandis que la grappe *graphite* possède un comportement spatialement et temporellement hétérogène tout au long de la phase de calcul.

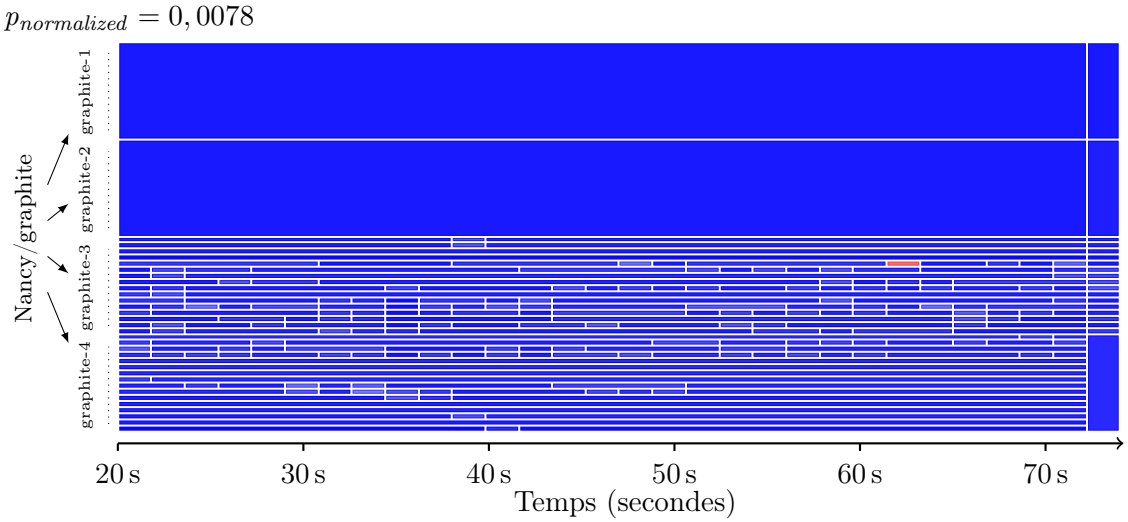


FIGURE 6.16 – Partition spatiotemporelle de LU.C-Nan-700 montrant le mode (ratio d'activité), et focalisée sur la phase de calcul de la grappe *graphite*. On met en évidence que les machines graphite-3 et 4 ont un comportement très hétérogène au cours du temps.

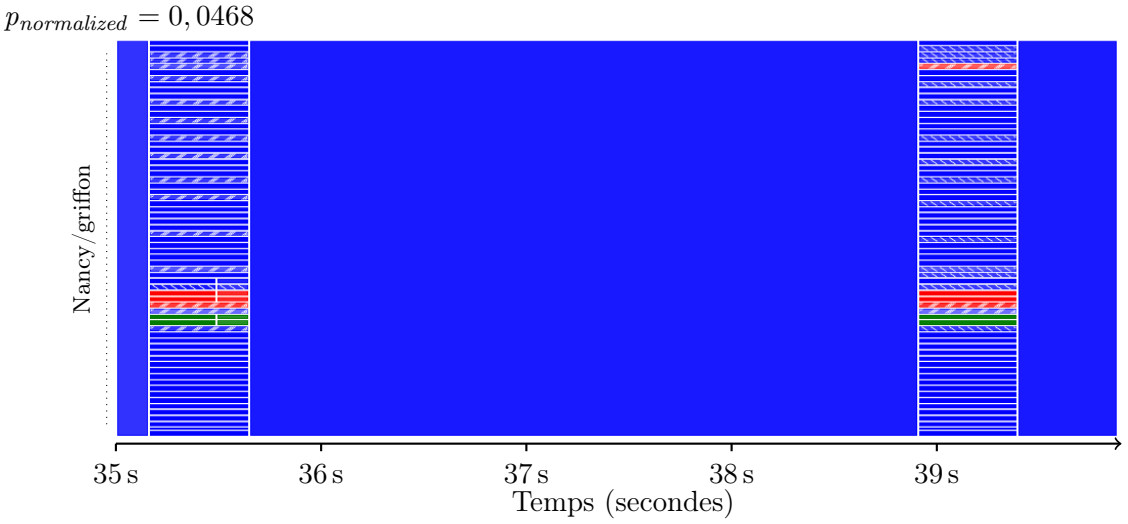


FIGURE 6.17 – Partition spatiotemporelle de LU.C-Nan-700 montrant le mode (ratio d'activité), et focalisée sur la perturbation temporelle touchant la grappe *griffon*. Celle-ci se décompose en deux perturbations où l'on observe clairement que certaines machines sont bloquées dans l'état `MPI_Wait` ou `MPI_Send`. La hiérarchie niveau machine n'est pas représentée pour des raisons de lisibilité.

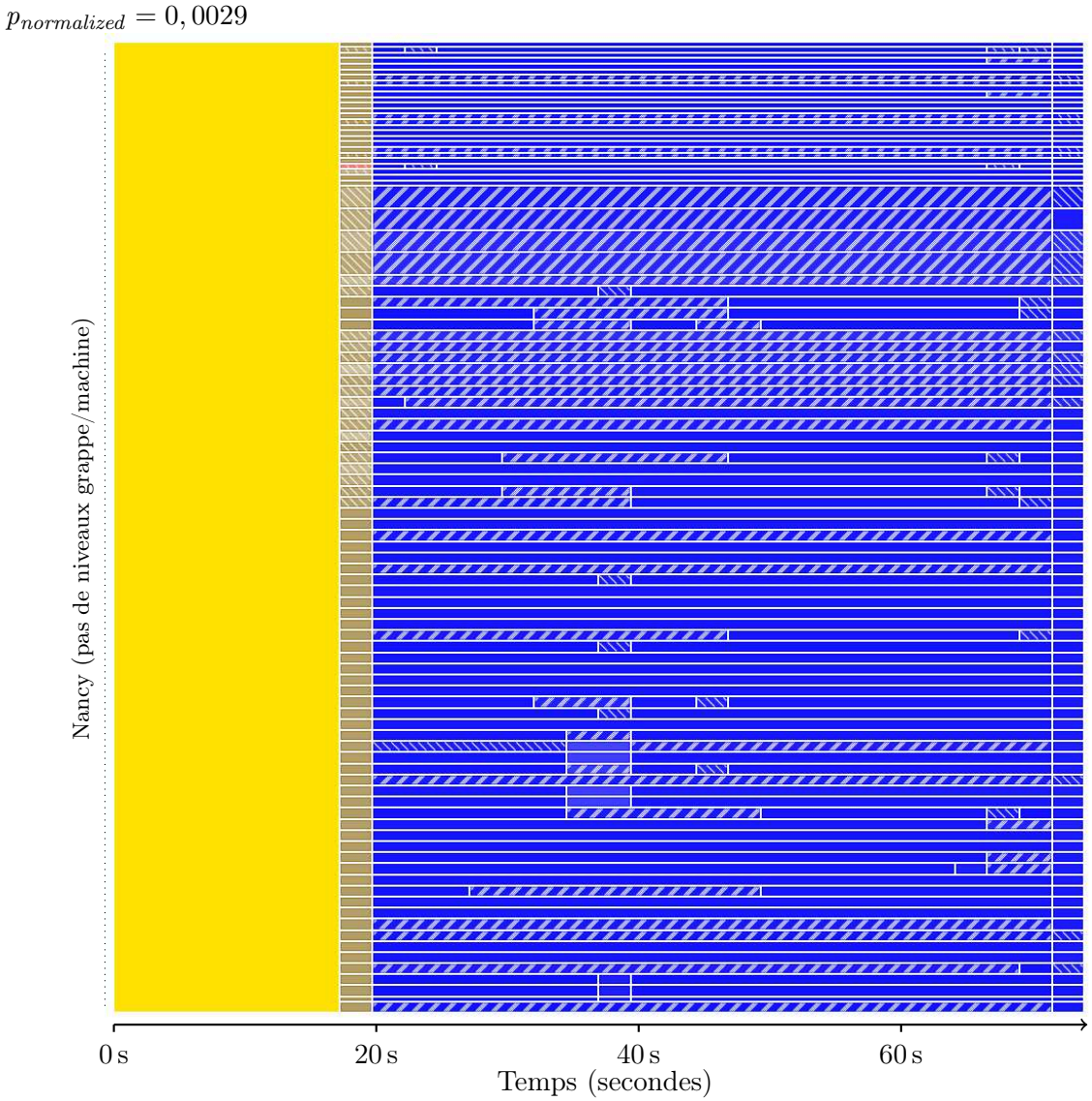


FIGURE 6.18 – Partition spatiotemporelle de LU.C-Nan-700 montrant le mode (ratio d'activité), en employant une hiérarchie sans les niveaux intermédiaires grappe et machine. Cette visualisation souligne l'incidence de la hiérarchie sur le résultat de l'agrégation, et donc sur la détection des anomalies : il est difficile de trouver une partition mettant en relief l'anomalie temporelle sur *griffon* et le comportement hétérogène de *graphite* à cause de la contrainte sur l'agrégation spatiale de type *tout ou rien*.

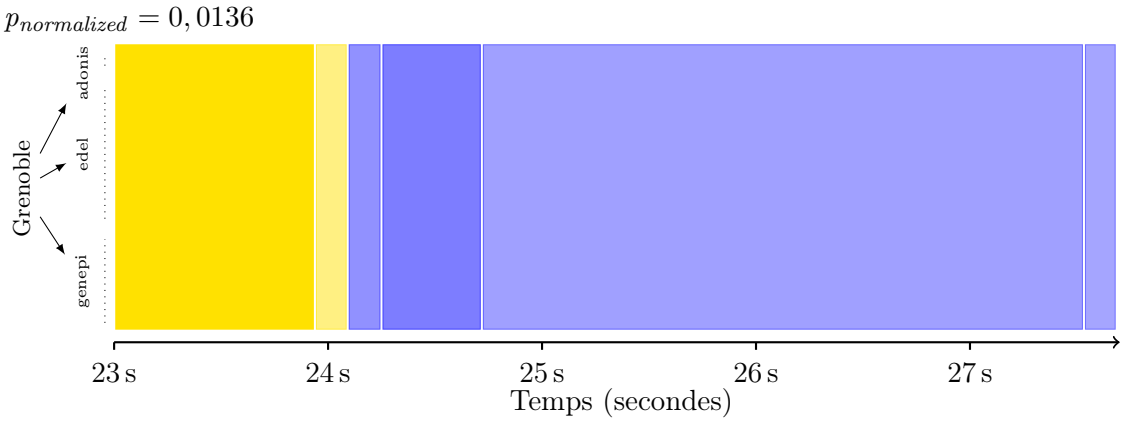


FIGURE 6.19 – Partition spatiotemporelle de LU.C-Gre-700 montrant le mode (ratio d'activité). Nous ne représentons qu'une partie de la phase d'initialisation pour permettre une comparaison visuelle avec la Figure 6.15. Ici, le comportement spatiotemporel de l'application est homogène. Le temps passé dans `MPI_Recv` (environ 23% sur cette zone) est beaucoup plus faible que pour LU.C-Nan-700 (65%), ce qui est mis en évidence grâce à l'intensité de la couleur.

il est difficile de trouver une représentation avec un niveau de détail satisfaisant : on passe instantanément d'une représentation agrégeant l'ensemble de la phase de calcul à la représentation complexe de la Figure 6.18. La perturbation temporelle localisée sur *griffon* est difficile à déceler, de même que l'hétérogénéité du comportement spatiotemporel de *graphite*. Les contraintes sur l'agrégation, de type *tout ou rien* sur l'espace, rendent l'analyse peu intéressante lorsque l'espace n'est pas structuré puisqu'il n'existe pas de niveaux d'agrégation spatiale intermédiaires. La perte d'information lors de l'agrégation spatiale étant importante du fait de la quantité de ressources impliquées, l'algorithme considère comme plus rentable, en terme d'optimalité des mesures de qualité, de laisser cette dimension désagrégée et d'agréger localement sur le temps.

Nous comparons enfin LU.C-Nan-700 avec LU.C-Gre-700, représentée Figure 6.19. Ici, le processus de désagrégation, dans le cas de Grenoble, montre l'homogénéité du comportement du système. L'intensité des couleurs nous renseigne que le temps relatif passé dans les communications MPI est moins important que pour Nancy, ce qui confirme l'impact du réseau sur les performances de ce benchmark.

6.6 Bilan

6.6.1 Respect des critères d'Elmqvist-Fekete

Nous confrontons ici notre technique d'agrégation spatiotemporelle face aux critères d'Elmqvist-Fekete développés Section 3.2. La Table 6.1 en fait une synthèse. Nous mettons en évidence l'effet de l'agrégation visuelle en montrant deux versions de chaque visualisation : sans (1) et avec (2) agrégation visuelle. Le critère G1 (entity budget), est respecté grâce à plusieurs techniques. La discrétisation temporelle, comme dans le cas de l'analyse temporelle, réduit le nombre d'éléments affichés selon cette dimension. La

Table 6.1 – Évaluation de notre technique d'agrégation spatiotemporelle et des visualisations associées face aux critères Elmqvist-Fekete. Un critère peut être satisfait : uniquement pour le temps (⋆) ou pour les deux dimensions (•).

Visualisation	**Attributs**	**Agrégation visuelle**	**G1**	**G2**	**G3**	**G4**	**G5**	**G6**	**M**
Partition (1)	Vectoriels		⋆		•	•	•	•	•
Partition (2)	Vectoriels	•	•		•	•	•	•	•
Amplitude relative (1)	Scalaires		⋆	•	•	•	•	•	•
Amplitude relative (2)	Scalaires	•	•	•	•	•	•	•	•
Mode (1)	Vectoriels		⋆	•	•	•	•	•	•
Mode (2)	Vectoriels	•	•	•	•	•	•	•	•

technique d'agrégation spatiotemporelle réduit la complexité globale de la représentation. Enfin, la technique d'agrégation visuelle appliquée sur l'espace évite de dépasser le budget si la réduction de complexité n'a pas été assez forte sur cette dimension. La couleur et l'intensité permettent de représenter partiellement les valeurs associées aux agrégats (G2) dans le cas du mode ou de l'amplitude relative (ce qui n'est pas le cas de la partition). Les agrégats visuels représentent l'homogénéité du comportement temporel à l'aide de deux types de symboles. L'ensemble des représentations est simple (G3), les agrégats étant constitués de formes géométriques basiques (rectangles). Des éléments comme l'intensité aident à montrer si un agrégat contient des données sous-jacentes qui n'ont pas pu être représentées à cause de l'emploi du mode (G4). Les agrégats visuels sont, quant à eux, signifiés à l'aide de diagonales. La réponse au critère de fidélité est la même que pour l'agrégation temporelle : l'utilisateur contrôle les paramètres de discrétisation (nombre de tranches de temps, pré-agrégation spatiale), et l'opérateur d'agrégation employé est explicite de même que la perte d'information et de la réduction de complexité de la représentation macroscopique (G5). La visualisation est responsable d'une perte d'information, le mode, par exemple, ne montrant que le type d'évènement majoritaire, de même que l'agrégation visuelle masque des informations sur la partition des éléments sous-jacents. Ici, les solutions que nous proposons seront plutôt fondées sur l'interaction (statistiques synchronisées, accès au détail des agrégats en cliquant), décrites dans le chapitre suivant. En terme d'interprétabilité (G6), comme pour l'agrégation temporelle, l'intensité de l'agrégation est adaptée de manière à garder du sens dans la représentation. Néanmoins, il est important que l'analyste ait connaissance des différents types d'agrégation mis en jeux au niveau des données et de la visualisation pour que le processus menant à la visualisation lui apparaisse clair.

6.6.2 Conclusions sur l'analyse

La technique d'analyse spatiotemporelle présentée dans ce chapitre s'avère efficace pour mettre en évidence des comportements localisés dans le temps et l'espace, et qui ne sont pas décelables avec la technique d'analyse temporelle présentée dans le chapitre précédent. Le passage à l'échelle est assuré pour des traces d'un volume de 12 Go et de 218 millions d'évènements, contenant 700 processus.

Nos cas d'études ont montré l'influence de la hiérarchie spatiale sur le résultat de l'agrégation et donc l'interprétation des phénomènes spatiotemporels représentés. Cette hiérarchie est contenue dans la trace. Au niveau technique, cela implique que le logiciel

de traçage connaisse la structure du système, et en particulier qu'il soit capable de lier la hiérarchie matérielle et logicielle correctement[3]. Certaines topologies ne peuvent pas être représentées sous la forme de hiérarchies (c'est le cas des interconnects en tore), ce qui rend notre technique inapplicable. De manière plus générale, le choix de la hiérarchie part du principe que cette dernière explique, en partie au moins, le comportement spatial au cours du temps de l'application : un exemple simple est que la technologie employée diffère selon l'endroit où l'on se situe dans la hiérarchie, ce qui peut être responsable de déséquilibres en terme de performances. Ceci n'est pas toujours le cas, et il est alors possible que la hiérarchie logicielle ou matérielle ne soit pas celle qui convienne le mieux à l'analyse du comportement du système. Ainsi, la principale faiblesse de notre technique est le fait que l'analyste ou l'outil de traçage doivent « deviner » la hiérarchie optimale.

[3]cela peut être problématique lorsque l'affinité des processus n'est pas fixe, par exemple.

CHAPITRE 7

Implémentation des techniques d'agrégation multidimensionnelles au sein de l'outil d'analyse Ocelotl

Nous avons décrit, dans la partie précédente, plusieurs techniques d'analyse fondées sur la *méthode de Lamarche-Perrin*, appliquées aux dimensions temporelles et spatiotemporelles. Dans ce Chapitre, nous abordons leur implémentation au sein d'un logiciel informatique, Ocelotl[1]. Ocelotl est écrit en Java et est un *plug-in* Eclipse. C'est aussi un module de Framesoc [11, 12, 18][2], l'infrastructure de gestion de traces et d'analyse du projet SoC-Trace. Nous décrivons plusieurs aspects d'Ocelotl. D'abord, ses fonctionnalités et son architecture, à travers la description de l'interface graphique et du noyau fonctionnel. En particulier, nous mettons l'accent sur la conception modulaire de l'outil permettant à un utilisateur tiers d'y ajouter ses propres techniques d'agrégation ou de visualisation. Nous abordons aussi l'intégration de l'outil dans la méthodologie de Shneiderman, assurant une analyse cohérente partant d'une vue synthétique jusqu'à une représentation détaillée, en décrivant les mécanismes d'interaction présents dans Ocelotl. Enfin, les aspects performances et interactivité complètent cette description de l'outil, et confirment sa capacité de passage à l'échelle pour des traces de gros volumes. Nous décrivons précisément les techniques servant à réduire le temps de calcul des différents traitements appliqués successivement aux données de la trace.

Plusieurs articles [10, 11, 13, 17] et rapports de recherche [12, 16] traitent de l'outil Ocelotl, de ses fonctionnalités et de ses performances. Ces travaux sont issus, en partie, de la collaboration avec Generoso Pagano, ingénieur de recherche à Inria, affilié au projet SoC-Trace, concepteur et développeur de l'infrastructure Framesoc [11, 12, 18], dont les fonctionnalités et performances d'Ocelotl sont en partie dépendantes, et avec Vania Marangozova-Martin, Maître de Conférences à l'Université Joseph Fourier, Grenoble, responsable de l'infrastructure logicielle Framesoc dans le cadre du projet SoC-Trace. Ils sont tous deux coauteurs de plusieurs articles et rapports de recherche qui décrivent à la fois Framesoc et les techniques d'agrégation d'Ocelotl [11, 12, 17]. J'ai développé Ocelotl

[1] http://soctrace-inria.github.io/ocelotl/
[2] http://soctrace-inria.github.io/framesoc/

principalement pendant la seconde année de mon doctorat, ce qui constitue la majorité des éléments et fonctionnalités décrits dans ce Chapitre. Youenn Corre a par la suite été recruté en tant qu'ingénieur de recherche à Inria, affilié au projet SoC-Trace, afin d'assurer le paquetage, la maintenance et le nettoyage du code d'Ocelotl (mise en conformité lors de l'évolution de l'API de Framesoc), et l'intégration de fonctions avancées, comme les caches, la sélection spatiale, les visualisations statistiques, ou la vue d'ensemble d'appoint, que nous avons conçues ensemble et qu'il a par la suite implémentées dans l'outil. Il a, en outre, développé les programmes réalisant les tests de performances sur la lecture des traces et du cache. Il est également co-auteur d'une publication [13].

7.1 Framesoc

Framesoc est une infrastructure d'analyse et de gestion de traces, développée dans le cadre du projet SoC-Trace, sous la forme d'un ensemble de plug-ins Eclipse écrits en Java, et répondant à plusieurs problématiques, comme la gestion efficace de traces volumineuses, d'outils d'analyse, ou encore la définition d'un protocole pour communiquer des résultats d'analyse entre outils. Framesoc possède aussi une interface graphique, et des outils d'analyse classiques, comme un certain nombre de visualisations statistiques ou un diagramme espace-temps de type diagramme de Gantt.

Framesoc fournit un stockage des traces, importées sous la forme de bases de données (nous utilisons, dans nos expériences, SQLite) à l'aide d'importateurs spécifiques à un format de trace, et dont le contenu est représenté via un modèle de données relationnel et générique, inspiré, entre autre, par les concepts de catégories d'évènements issus de Pajé [30,31]. Cela donne l'opportunité d'importer un grand nombre de formats de traces : Framesoc est actuellement compatible avec les formats Pajé, OTF2 [29], Paraver [33], CTF [34], KPTrace [36], ou encore GStreamer. Concernant la lecture, une API fournit plusieurs niveaux d'abstraction des requêtes SQL, et un mécanisme de reconstruction des données et évènements de la trace sous forme d'objets. Pour l'écriture, effectuée par un importateur ou un générateur de traces, un certain nombre de fonctionnalités rendent l'interface avec la base de données transparente.

Des outils externes peuvent être ajoutés à Framesoc, comme c'est le cas avec Ocelotl, et tirer parti de ces différentes fonctionnalités. Les points d'extension[3] fournis par Eclipse facilitent l'ajout de ces nouveaux modules sans avoir à modifier le code de Framesoc. Un mécanisme de type *Publish-Subscribe* permet, entre autres, de synchroniser ou de faire interagir les différentes vues correspondant aux différents outils d'analyse. La Figure 7.1 synthétise l'architecture de Framesoc. Elle montre, de bas en haut :

— le modèle de données générique utilisé pour représenter les traces (stockées sous la forme de bases de données relationnelles), identifier les outils, et sauvegarder les résultats d'analyse ;
— l'API faisant le lien entre modèle de données et outils ;
— l'interface graphique constituée de modules servant à gérer les traces et les outils, comme des importateurs, des exportateurs et des outils d'analyse (visuels ou non).

Plusieurs facteurs justifient la décision de se greffer à Framesoc :

[3] http://help.eclipse.org/luna/index.jsp?topic=%2Forg.eclipse.pde.doc.user%2Fconcepts%2Fextension.htm

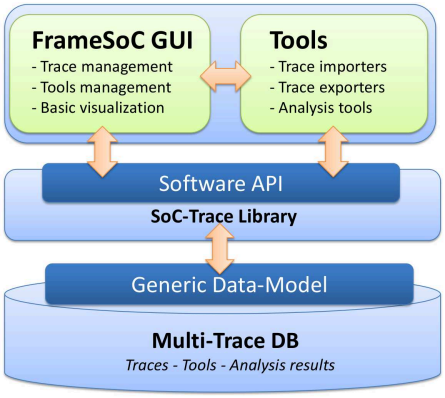


FIGURE 7.1 – Architecture de l'infrastructure Framesoc.

— un gain en terme de temps de développement, puisqu'on profite des mécanismes d'import et de lecture de la trace ;
— les performances relativement bonnes des bases de données pour les types de requêtes et filtrages effectués par notre outil d'analyse ;
— l'interaction possible avec d'autres outils d'analyse ;
— la portabilité ;
— l'intégration de l'outil dans les distributions du Système d'Exploitation des Traces proposées par le projet SoC-Trace, comprenant Framesoc et l'ensemble des outils conçus par les partenaires du projet ;
— la possibilité de profiter d'un support développeur.

7.2 Architecture

L'architecture d'Ocelotl obéit globalement au modèle MVC. On distingue, en particulier, un noyau fonctionnel et une interface graphique. Nous commençons par décrire l'interface graphique et ses différents composants, puis nous évoquons des caractéristiques du noyau fonctionnel.

7.2.1 Interface graphique

L'interface graphique est constituée de plusieurs composants graphiques. Nous utilisons la Figure 7.2 pour les situer et les décrire.

Paramètres d'agrégation et de visualisation Le composant A correspond aux paramètres qu'il est obligatoire de définir avant une session d'analyse. Quatre *combo boxes* permettent, respectivement de choisir :
— une trace importée dans Framesoc ;
— une métrique pour construire le modèle microscopique (comme la quantité d'évènements, le temps total ou moyen passé dans un type d'état par tranche de temps, la moyenne de la valeur d'une variable, décrits dans la Section 5.2) ;
— une agrégation temporelle ou spatiotemporelle ;
— une visualisation compatible avec les sélections des autres *combo boxes*.

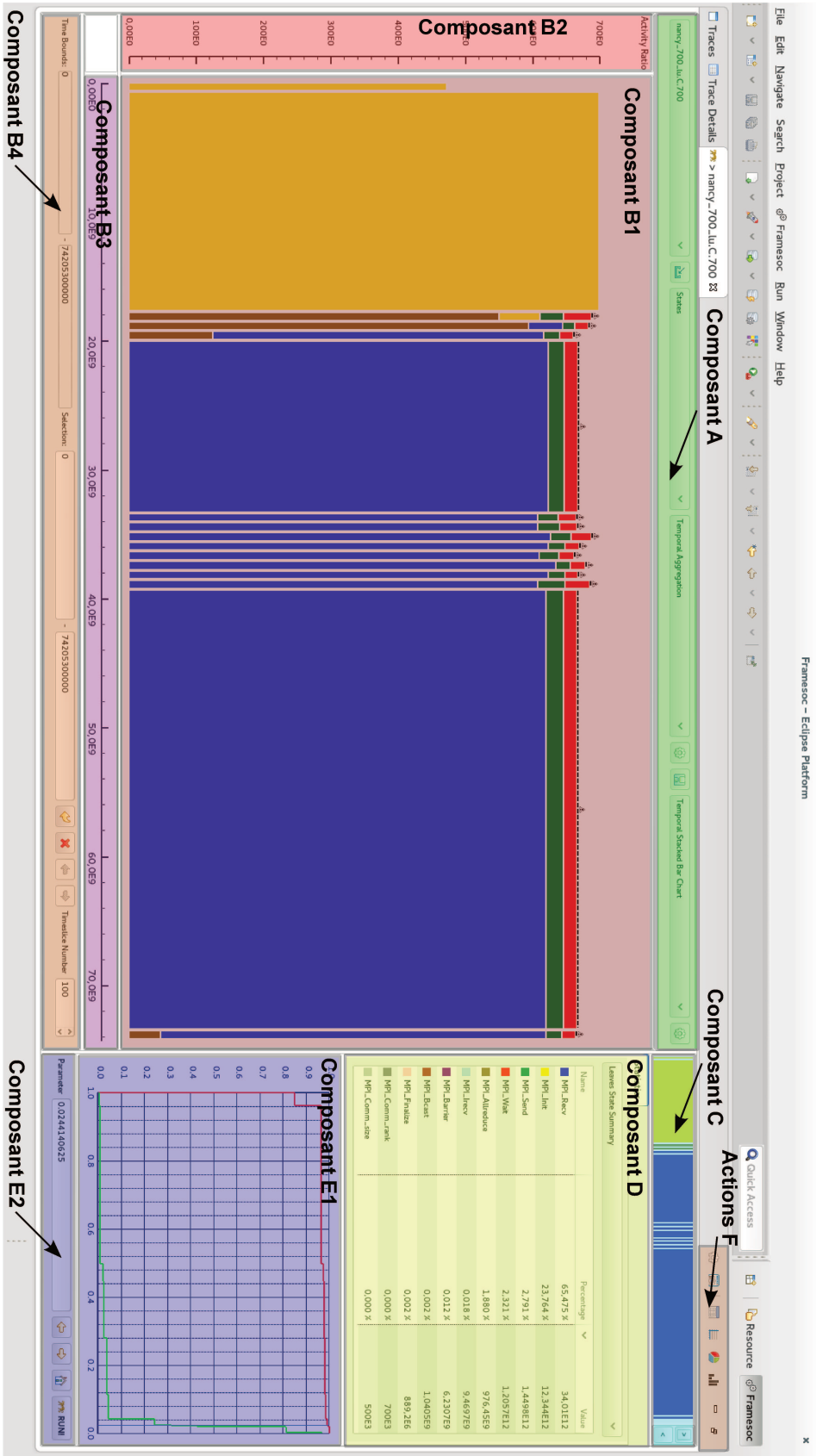


FIGURE 7.2 – Interface graphique d'Ocelotl montrant ses différents composants : A - Paramètres d'agrégation et de visualisation, B - Visualisation agrégée principale, C - Overview, D - Statistiques, E - Courbes de qualités, F - Barre d'actions.

Visualisation agrégée principale Le composant B1 constitue la visualisation principale. On y représente le résultat de l'agrégation (par exemple l'histogramme empilé pour l'agrégation temporelle). Le composant B2 correspond à l'axe des ordonnées, qui représente, selon la visualisation employée, l'amplitude des valeurs de l'histogramme empilé pour les analyses temporelles ou la hiérarchie des ressources pour les analyses spatiotemporelles. Le composant B3 est l'axe temporel. Le composant B4 contient :

— la zone temporelle affichée ;
— la zone temporelle sélectionnée (champ éditable) ;
— un bouton remettant les valeurs de la zone sélectionnée à leur valeur par défaut ;
— un bouton annulant la sélection ;
— des boutons de navigation (précédent et suivant) ;
— le nombre de tranches de temps (champ éditable).

Overview Le composant C est une vue d'ensemble de la trace calculée selon la méthode d'agrégation temporelle, et utilisant la même métrique que celle employée par la vue principale, afin d'indiquer à l'utilisateur où ce dernier se situe dans la trace lorsqu'il zoome temporellement, pour ne pas perdre le contexte. Les bornes de la zone temporelle affichée sont reportées sur cette vue d'ensemble, ainsi que la sélection. Pour minimiser la taille prise par ce composant, on emploie les représentations du mode (Section 5.6.4) ou de la partition temporelle (Section 5.6.2) plutôt que des histogrammes.

Statistiques Le composant D est associé à la vue statistique. La *combo box* permet de choisir une visualisation statistique. Un système de correspondance permet de ne proposer que les opérateurs statistiques compatibles avec la métrique et le type d'agrégation employés. La visualisation statistique est synchronisée avec la vue principale d'Ocelotl (composant B1) et permet de donner des informations statistiques complémentaires sur la zone temporelle ou spatiotemporelle affichée ou sélectionnée, comme la quantité de chaque type d'évènements, la durée totale ou moyenne de chaque type d'états ou la valeur moyenne des variables.

Courbes de gain d'information et de gain en complexité Le composant E1 représente ici deux courbes : l'amplitude des gains en information et en complexité relatifs, calculés par rapport à la partition la plus agrégée, chaque point de ces deux courbes étant associé à une partition $\mathcal{P}_p$ optimale obtenue pour une valeur de p donnée. Les différentes valeurs de p sont obtenues grâce à l'algorithme des partitions optimales, décrit dans la Section 5.5.3. Cette représentation, inversée par rapport à la réduction de complexité et la perte d'information que nous avons évoquées dans les Chapitres précédents, est plus intuitive, puisque l'on part généralement de la représentation la plus agrégée. Il est néanmoins possible de commuter vers les courbes de réduction de complexité et de perte d'information relatives à la partition la plus désagrégée. Les courbes sont interactives et cliquer dessus change la valeur de p, et donc la partition affichée. Dans le composant E2, deux boutons incrémentent ou décrémentent p, tandis qu'un champ texte permet d'éditer manuellement p. Le bouton RUN permet, quant à lui, de lancer l'analyse.

Barre d'actions Dans le coin supérieur droit d'Ocelotl (Actions F) sont présents un certain nombre de boutons permettant d'accéder à diverses fonctions. On distingue :

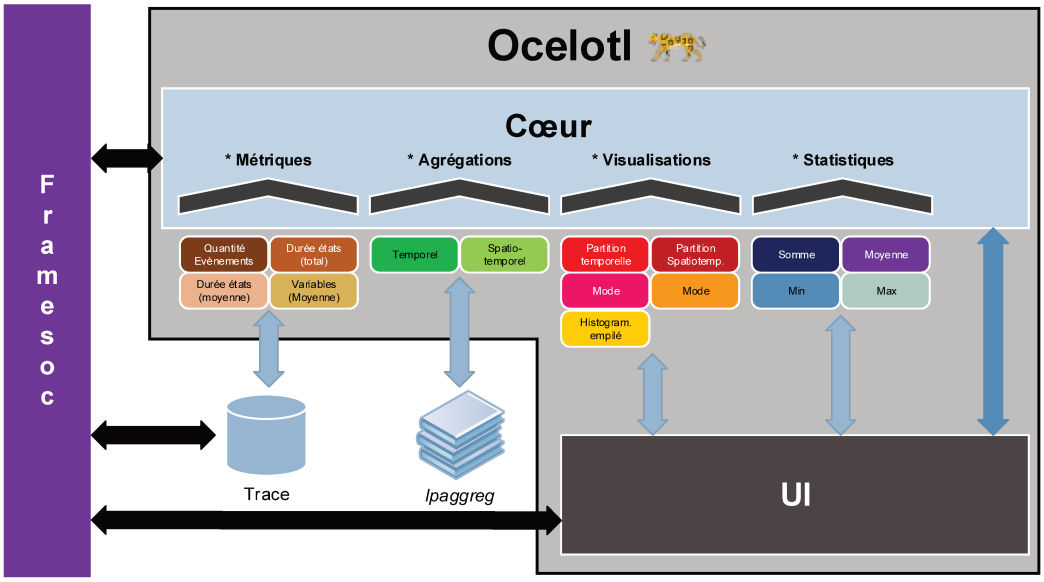


FIGURE 7.3 – Schéma montrant la structure globale d'Ocelotl, divisée en un cœur et une interface graphique, et les interactions entre les composants. Quatre types de modules (métriques, agrégations, visualisations, statistiques) se greffent au cœur, sous la forme d'extensions.

— les paramètres avancés d'Ocelotl ;
— l'enregistrement d'un *snapshot* de la visualisation en cours ;
— des boutons de commutation vers d'autres outils de Framesoc (table d'évènements, diagramme espace-temps, diagramme circulaire, ou encore histogramme).

7.2.2 Noyau fonctionnel

Nous ne détaillons pas de manière exhaustive le noyau fonctionnel d'Ocelotl, mais plutôt certaines de ces caractéristiques.

Modularité de la structure

Comme le montre la Figure 7.3, la structure d'Ocelotl est modulaire. Ainsi, on peut voir Ocelotl comme un squelette fournissant un certain nombre d'interfaces, sur lequel viennent se greffer des extensions (grâce au système de points d'extension d'Eclipse) détectées dynamiquement par le noyau fonctionnel qui ne possède aucune dépendance envers elles :

— les *métriques* définissent le modèle microscopique à utiliser et la métrique employée pour reconstruire ce modèle ;
— les *techniques d'agrégation*, temporelle ou spatiotemporelle ;
— les *visualisations* sont associées au code calculant les valeurs des agrégats, et à la vue associée ;
— les *opérateurs statistiques* calculent les données qui seront affichées dans la vue statistique.

Il est donc aisé d'ajouter de nouveaux modules de chaque sorte à Ocelotl, afin d'étendre ses capacités en terme d'analyse, et de l'adapter à un besoin particulier. Il existe un système de compatibilité entre les différents types de modules : les points d'extensions requièrent de définir, pour chaque type de module, un ou plusieurs *tag* relatifs à leur compatibilité avec les modules des autres types. Ainsi, suite à la sélection de la métrique, la combo box relative à la technique d'agrégation ne proposera que celles qui partagent un *tag* en commun. Puis, la combo box relative à la visualisation ne proposera à son tour que celles qui partagent un *tag* avec la métrique et avec la technique d'agrégation. Un exemple trivial : il est impossible de choisir une visualisation spatiotemporelle pour une analyse temporelle. Les restrictions de compatibilité peuvent bien entendu être plus complexes. Enfin, il est possible d'attribuer des paramètres propres à chaque technique d'agrégation ou visualisation, qui seront modifiables au niveau de l'interface graphique via une boîte de dialogue, permettant, par exemple, de filtrer certaines dimensions, ou de modifier le rendu graphique.

Pipeline d'analyse

L'analyse fonctionne ensuite comme un traitement séquentiel faisant intervenir successivement chacun de ces modules. Le module *métrique* lit la trace et génère un modèle microscopique. Il est à noter que nous utilisons le même modèle pour les analyses temporelles et spatiotemporelles. Le module *technique d'agrégation* applique l'algorithme d'agrégation sur le modèle microscopique. Dans le cas d'une analyse spatiotemporelle, c'est lui qui se charge de reconstituer la hiérarchie spatiale. Enfin, le module de visualisation attribue une valeur (scalaire ou vectorielle) à chacun des agrégats, qui sera utilisée pour le rendu (choix de l'évènement à montrer, amplitude ou transparence). La Figure 7.4 montre l'intégralité du pipeline d'analyse d'Ocelotl, de la lecture de la trace à la visualisation, en incluant les éventuelles interactions de l'utilisateur.

Bibliothèque lpaggreg

La bibliothèque *lpaggreg*[4] implémente les algorithmes d'agrégation d'un ensemble ordonné (temporel), hiérarchique (spatial), et ordonné et hiérarchique (spatiotemporel). C'est une bibliothèque extérieure à Ocelotl, écrite en C++ pour en maximiser les performances, mais aussi permettre sa diffusion pour d'autres types d'applications. Elle est interfacée avec les modules d'agrégation d'Ocelotl grâce à JNI[5]. Afin de ne pas limiter la portabilité d'Ocelotl et de simplifier sa distribution et son installation, nous y intégrons différentes versions de cette bibliothèque, compilées pour Linux et Windows, sous la forme de fragments Eclipse[6].

7.3 Intégration dans la méthodologie de Shneiderman

Nous détaillons les solutions qui nous permettent de remplir chaque étape du mantra de Shneiderman [1] (« overview first, zoom and filter, then details-on-demand »).

[4]https://github.com/dosimont/lpaggreg
[5]https://github.com/dosimont/lpaggregjni
[6]http://help.eclipse.org/indigo/index.jsp?topic=%2Forg.eclipse.platform.doc.isv%2Fguide%2Fproduct_def_plugins.htm

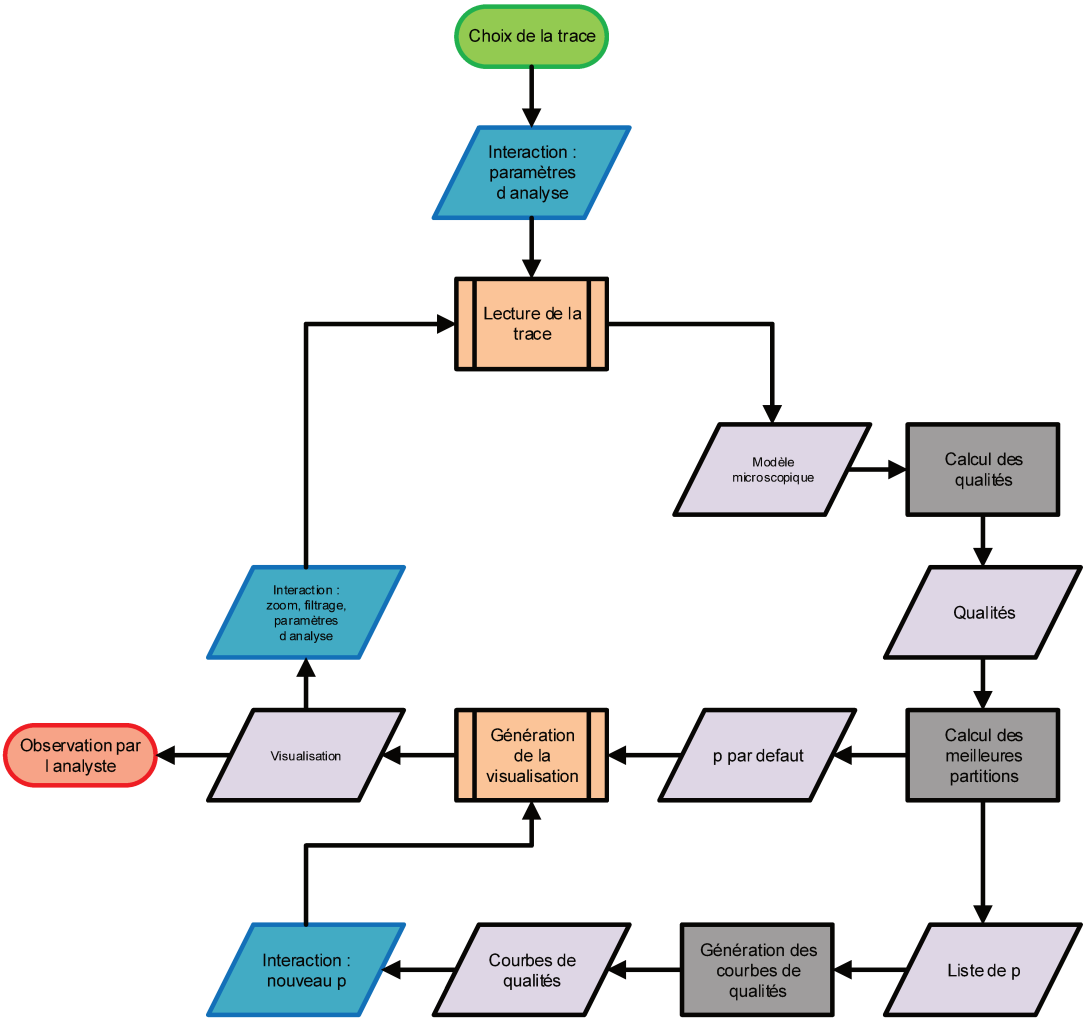


FIGURE 7.4 – Organigramme montrant le pipeline d'analyse d'Ocelotl.

7.3.1 Overview

Nous considérons que nous remplissons le contrat lié à la présence d'une étape de vue d'ensemble par la technique d'analyse que nous proposons dans cette thèse. De surcroît, l'*overview* temporel décrit par le composant C de la Figure 7.2 permet de garder le contexte et de situer, après un zoom sur une sous-partie de la trace, cette dernière par rapport à l'intervalle de temps global.

7.3.2 Zoom

Le zoom est une fonctionnalité employée par un grand nombre d'outils d'analyse dans leurs représentations spatiotemporelles [5, 6, 7, 30, 44, 45, 48, 49, 50, 52, 80, 100]. Dans le cas d'Ocelotl, il ne s'agit pas d'un zoom graphique, c'est-à-dire grossissant les objets graphiques ou désagrégeant les agrégats visuels, mais d'un zoom sur les données : le processus d'agrégation temporelle ou spatiotemporelle est réitéré sur une sous-partie de la trace.

Pour ce faire, nous avons implémenté un mécanisme de sélection d'une sous-partie de la trace, à l'aide de la souris. Dans le cas d'une analyse temporelle, la sélection est monodimensionnelle sur le temps. La granularité de la sélection est la tranche de temps : ainsi, la sélection s'ajuste de manière à inclure automatiquement les tranches de temps complètes si l'utilisateur ne les englobent pas complètement. Formellement, $\text{Select}(T) = T_{(i,j)}$ tel que $T_{(i,j)} \in \mathcal{I}(T)$.

Cette restriction existe pour plusieurs raisons :

— elle permet, pour les modules synchronisés sur cette sélection (comme les statistiques), de pouvoir réutiliser le modèle microscopique courant ;

— elle augmente la probabilité de réutiliser les caches (décrits plus loin dans la Section 7.4.1) ;

— au niveau de la sémantique de la sélection, nous considérons que sélectionner avec une granularité plus fine que le pas de temps de la discrétisation n'est pas cohérent, puisque la granularité des phénomènes représentés est de l'ordre de la tranche de temps. Procéder ainsi serait donc hasardeux.

Toutefois, cette restriction peut être levée si l'utilisateur saisit les timestamps de sélection manuellement.

Dans le cas spatiotemporel, la sélection est multidimensionnelle sur le temps et l'espace. Les restrictions sur le temps sont identiques à celles de la sélection de l'analyse temporelle, tandis que pour l'espace, nous n'autorisons pas la sélection partielle d'un sous-ensemble d'un nœud. Formellement, $\text{Select}(S) = S_k$ tel que $S_k \in \mathcal{H}(S)$. Ainsi, si l'utilisateur sélectionne deux nœuds S_a et $S_b \in \mathcal{H}(S)$ tels que $S_a \cup S_b \notin \mathcal{H}(S)$, alors $\text{Select}(S) = S_k \in \mathcal{H}(S)$ tel que $S_a \subset S_k$ et $S_b \subset S_k$ (et que la profondeur de S_k soit maximale).

Après sélection, on relance alors le processus d'analyse (lecture de la trace, création du modèle microscopique, agrégation et visualisation) sur cette sous-partie de la trace.

7.3.3 Filter

Il est possible d'effectuer un filtrage des types d'évènements et de la dimension spatiale (en faisant fi des contraintes hiérarchiques imposées par la sélection spatiale) grâce à une fenêtre de configuration.

7.3.4 Details on demand

Enfin, nous réalisons la dernière étape de la méthodologie de Shneiderman de plusieurs manières :

— Il est possible de commuter d'Ocelotl vers un autre outil d'analyse de Framesoc, comme le diagramme espace-temps, sur les bornes spatiotemporelles sélectionnées, afin d'obtenir plus de détails. Nous avons illustré cette fonctionnalité lors de l'analyse de différents cas d'études, Section 6.5.

— Les statistiques donnent des informations supplémentaires sur les agrégats ou la zone sélectionnée. La Figure 7.5 en donne une illustration.

— Un clic droit avec la souris sur un agrégat visuel, dans les représentations spatiotemporelles, ouvre une boîte de dialogue affichant une nouvelle représentation de cet agrégat, avec une résolution plus grande. Cette technique, illustrée par la Figure 7.6 permet de désagréger l'agrégat visuel et de détailler son contenu, sans avoir à zoomer dans la représentation principale et donc perdre le contexte.

7.4 Performances et interactivité

Une des contraintes majeures imposée par l'analyse visuelle est l'interactivité, c'est-à-dire la réactivité de l'outil face à une action de l'utilisateur et sa capacité à lui fournir un résultat dans un délai raisonnable. La notion de délai raisonnable est cependant en partie subjective, puisqu'elle s'appuie sur le ressenti de l'utilisateur face à l'utilisation de l'outil, et sur sa tolérance à l'attente, qui peut être modulée par les informations qu'il possède sur la complexité et les enjeux liés à chacune des tâches effectuées suite à son interaction.

Dans le cas d'Ocelotl, nous distinguons trois points particuliers affectant les performances et l'interactivité, et dont les solutions d'optimisation sont différentes. D'abord, la lecture de la trace, c'est-à-dire la récupération des évènements nécessaires à la construction du modèle microscopique, et la construction de ce dernier. Ensuite, le processus d'agrégation de ce modèle microscopique qui fournit une partition. Enfin, le rendu graphique de la partition. Il est à noter que ces trois étapes fonctionnent comme un cycle : on lit la trace, on génère la partition, on l'affiche. L'interaction, à l'aide d'actions comme un zoom ou un changement de paramètre p, déclenche un nouveau cycle (entier ou partiel). L'amélioration de l'interactivité consiste donc à augmenter les performances de chacune des étapes du cycle (en se focalisant prioritairement sur les plus coûteuses), d'une part, et à éviter au maximum de réitérer un cycle complet depuis la lecture de la trace, d'autre part.

Tous les tests de performances évoqués dans ce Chapitre ont été réalisés sur la machine *flutin*, dont les spécifications sont données par la Figure 7.7.

Pour certains tests de performances de cette Section, nous employons des traces artificielles. La Figure 7.8 décrit leur contenu.

7.4.1 Lecture de la trace et construction du modèle microscopique

Nous proposons deux techniques visant à améliorer les performances de la lecture de la trace et de la construction du modèle microscopique : l'optimisation des requêtes à la base de données, et l'emploi d'un cache pour stocker et réutiliser les modèles microscopiques dans les sessions d'analyses ultérieures.

Statistics

Leaves State Summary

Name	Percentage	Value
MPI_Wait	53,949 %	1,8607E12
MPI_Send	23,125 %	797,57E9
MPI_Init	2,483 %	85,648E9
MPI_Irecv	0,311 %	10,732E9
MPI_Finalize	0,007 %	257,8E6
MPI_Barrier	0,005 %	165,34E6
MPI_Bcast	0,001 %	43,75E6
MPI_Comm_size	0,000 %	30,0E3
MPI_Comm_rank	0,000 %	110E3
MPI_Reduce	0,000 %	13,6E6

FIGURE 7.5 – Capture d'écran de la vue statistique d'Ocelotl (Composant D de la Figure 7.2) donnant des informations supplémentaires sur la zone affichée ou sur la sélection en cours.

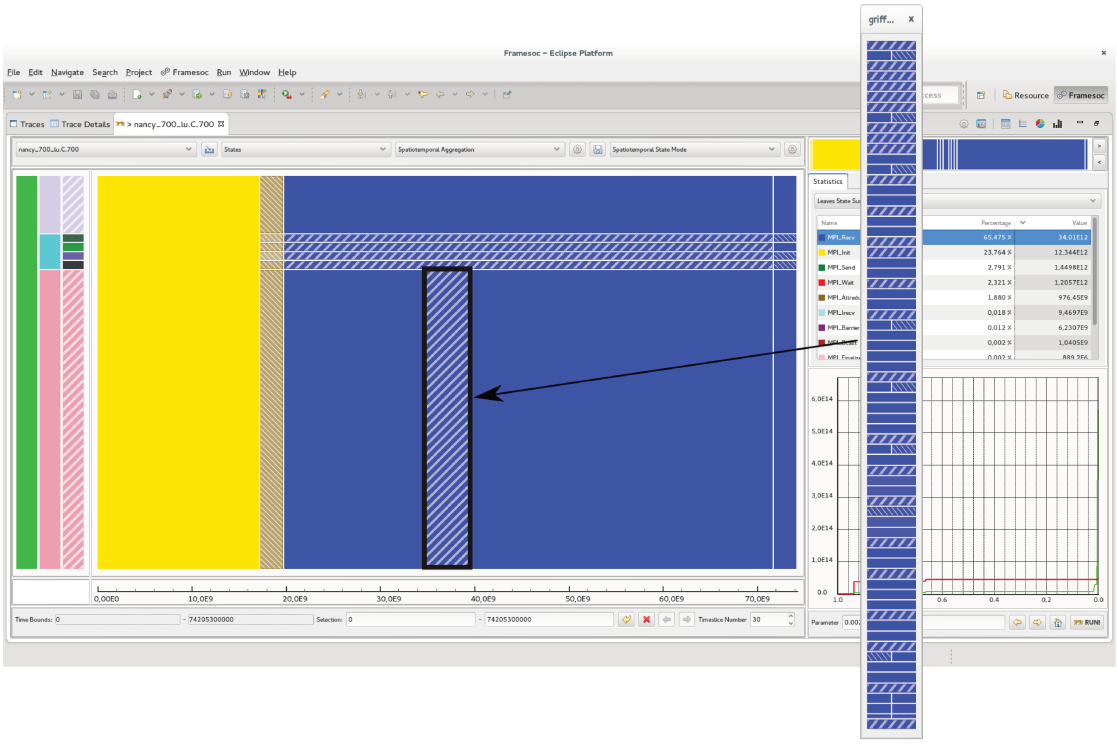


FIGURE 7.6 – Illustration de la fonctionnalité d'interaction avec un agrégat visuel, permettant d'ouvrir une fenêtre flottante et redimensionnable afin d'avoir plus de détails sur celui-ci. L'agrégat visuel sélectionné pour cette interaction est bordé de noir.

```
# Système d'exploitation :
Linux flutin 3.17.4-200.fc20.x86_64 #1 SMP
Fri Nov 21 23:26:41 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux
# Matériel :
Nombre de coeurs : 4
Spécifications des coeurs
- Modèle : Intel(R) Xeon(R) CPU E3-1225 v3 @ 3.20GHz
- Taille de cache : 8192 Ko
- Hyperthreading : actif
Gouverneur cpufreq : performance
RAM : 32890568 Ko
Disques
- Disque 1 (SSD 256 Go)
/dev/sda:
 Model=LITEONIT LCS-256M6S 2.5 7mm 256GB, FwRev=DC8110E,
 SerialNo=TW0XFJWX5508541E5214
 Config={ Fixed }
 RawCHS=16383/16/63, TrkSize=0, SectSize=0, ECCbytes=0
 BuffType=unknown, BuffSize=unknown, MaxMultSect=16, MultSect=16
 CurCHS=16383/16/63, CurSects=16514064, LBA=yes, LBAsects=500118192
 IORDY=on/off, tPIO={min:120,w/IORDY:120}, tDMA={min:120,rec:120}
 PIO modes:  pio0 pio3 pio4
 DMA modes:  mdma0 mdma1 mdma2
 UDMA modes: udma0 udma1 udma2 udma3 udma4 udma5 *udma6
 AdvancedPM=no WriteCache=enabled
 Drive conforms to: unknown:  ATA/ATAPI-1,2,3,4,5,6,7
  * signifies the current active mode
```

FIGURE 7.7 – Spécifications de la machine *flutin*.

```
#Ressources :
Hiérarchie spatiale de 1111 éléments à 4 niveaux constituée par un arbre décennaire :
un sommet (niveau 1), 10 ressources (niveau 2),
10*10 ressources (niveau 3), 10*10*10 ressources (niveau 4).
Seules les feuilles produisent des évènements
#10 types d'évènements
#Méthode de génération des évènements :
- On choisit un nombre d'évènements total.
- Chaque feuille produit un nombre d'évènements égal
au nombre d'évènements total divisé par 1000.
- Ces évènements sont des états.
- Leur type est choisi selon un tirage aléatoire uniforme.
- Ils sont tous contigus.
L'estampille de début d'un état est celle de fin de l'état précédent.
La durée de chaque état est déterminée à l'aide d'un tirage uniforme
sur l'ensemble des nombres entiers compris entre 1 et 100.
```

FIGURE 7.8 – Spécifications des traces synthétiques.

Requêtes optimisées

La construction du modèle microscopique consiste, dans son approche la plus simple, en la lecture de la trace, afin de récupérer ses évènements et remplir le modèle microscopique en agrégeant ces derniers grâce à un opérateur. Cette lecture de la trace est réalisée par l'intermédiaire de Framesoc, auquel on fournit une requête générée grâce à des paramètres déterminés par l'utilisateur :

— la période temporelle sur laquelle doivent survenir les évènements ;
— les ressources associées aux évènements à prendre en compte ;
— les types d'évènements souhaités.

Cela va générer un résultat d'analyse, géré par le DBMS, lequel contient les évènements correspondant à la requête spécifiée et accessibles à travers un itérateur. Ce dernier peut être accédé avant la récupération de tous les évènements, l'accès à l'élément suivant étant bloquant si ce dernier n'est pas encore disponible, ce qui permet de procéder à la construction du modèle microscopique en parallèle de la lecture de la trace.

La complexité de la requête influe sur le temps de génération du résultat d'analyse, qui est le facteur limitant de cette étape. Ainsi, deux requêtes sémantiquement identiques mais exprimées de deux manières différentes peuvent aboutir à un temps de lecture différent [101]. Il est donc essentiel de simplifier la requête au maximum, et d'éliminer toute condition redondante ou inutile. La requête totale se décompose en trois parties, relatives respectivement au temps (*condition* T), à l'espace (*condition* S), et aux types d'évènements (*condition* J), tel qu'il suit :

$$condition\ Total = condition\ T \wedge condition\ S \wedge condition\ J \tag{7.1}$$

Une première étape de simplification de cette requête consiste à ignorer les conditions dont le résultat est toujours vrai. Si l'intervalle de temps sélectionné correspond à l'ensemble de la trace, ou s'il n'y a aucun filtrage sur l'ensemble des ressources ou des types d'évènements, alors nous ignorons la condition respective. Une deuxième étape correspond à une simplification plus poussée. Dans le cas du temps, on ignore la condition relative à la borne minimale ou maximale de l'intervalle requêté si celle-ci est égale à l'estampille minimale ou maximale de la trace. Si l'ensemble des types d'évènements requêté est l'ensemble des types d'évènements d'une catégorie (évènements ponctuels, états, liens, variables), alors on remplace la condition *condition* J par une condition *condition* C, qui évalue la catégorie d'un évènement (la condition est une simple égalité, de complexité $\mathcal{O}\,1$) plutôt que son type (la condition est l'appartenance à un ensemble, de complexité $\mathcal{O}\,|\mathcal{J}(E)|$). Plus de précisions sur ces simplifications sont données en Annexe B. La Figure 7.9 montre l'influence de la présence ou pas de conditions superflues lors de la lecture de la trace entière. On constate que le temps de lecture et de génération du modèle microscopique est linéaire au nombre d'évènements mais que le coefficient directeur de la droite dépend de l'optimisation des requêtes. L'optimisation totale permet d'obtenir un temps 35% plus court que l'absence d'optimisation. On constate aussi que l'optimisation la plus rentable concerne les ressources, puisque c'est la requête la plus lourde dans notre scénario (1111 comparaisons à effectuer), suivie par les types d'évènements (10 comparaisons).

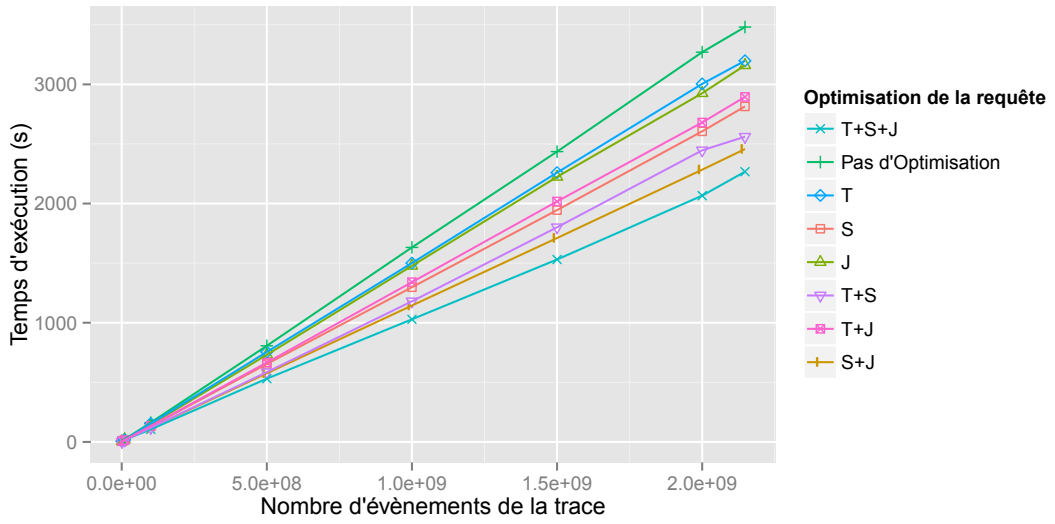


FIGURE 7.9 – Durée de lecture et de reconstruction du modèle microscopique (100 tranches de temps) en fonction du nombre d'évènements de la trace (générée selon le processus décrit par la Figure 7.8. Optimisation de la requête sur : T - Temps, S - Ressources spatiales, J - Types d'évènements.

Réutilisation du modèle microscopique

Bien que l'optimisation des requêtes offre un gain substantiel, les performances de la lecture de traces d'un grand volume n'assure pas une bonne interactivité. La Figure 7.9 montre que le temps de lecture de la trace et de génération du modèle microscopique à partir de la base de données atteint près de 2000 secondes pour une trace de 2 milliards d'évènements (environ 68 Go, la taille étant une fonction linéaire du nombre d'évènements), avec une requête optimisée.

Néanmoins, notre technique d'agrégation repose sur une abstraction de la trace, le modèle microscopique, dont le contenu et la structure sont identiques pour un ensemble de paramètres donné (intervalle de temps choisi, nombre de tranches de temps, métrique utilisée, ressources et types d'évènements sélectionnés). Grâce à l'agrégation utilisée pour générer ses valeurs, son empreinte mémoire est bien plus faible que la trace originale. Un fichier contenant ce modèle microscopique est donc beaucoup plus rapide à charger que la trace entière. Une autre propriété remarquable est que le modèle microscopique, qui est représenté en mémoire sous la forme d'une matrice, possède souvent une faible densité (due au fait qu'il est rare, en pratique, que tous les types d'évènements possibles surviennent au moins une fois, pour chaque ressource, au cours de chaque tranche de temps). En ne sauvegardant que les valeurs non nulles de la matrice dans ce même fichier, on réduit donc encore plus sa taille et sa vitesse de chargement. Tirant parti de ces propriétés, nous proposons de sauvegarder ce modèle microscopique lors de la première session d'analyse sous la forme d'un cache persistant, et de le réutiliser lors des sessions d'analyse suivantes. La Figure 7.10 montre le pipeline de lecture de la trace en incluant l'utilisation d'un cache.

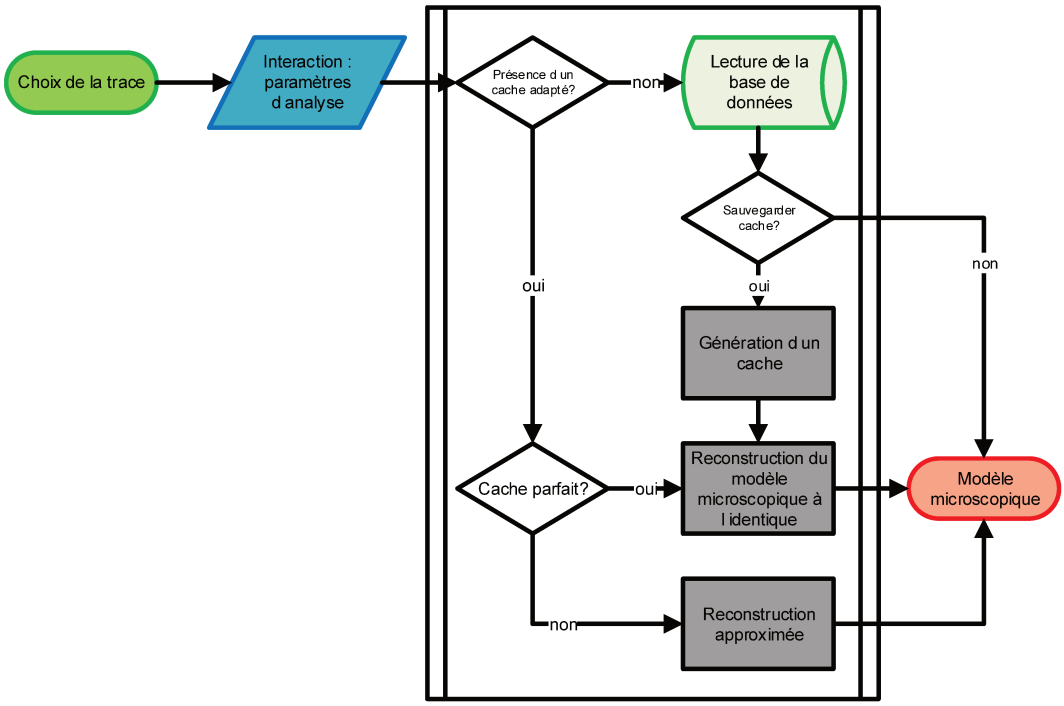
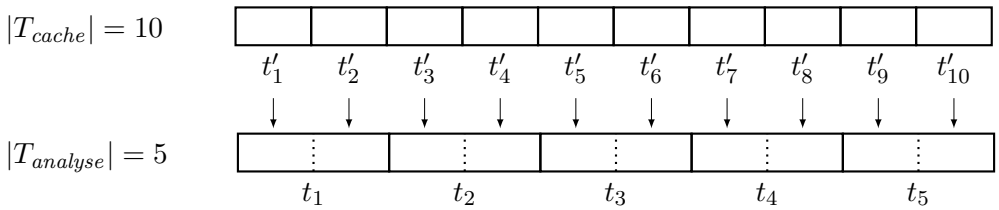


FIGURE 7.10 – Organigramme montrant le pipeline de lecture de la trace. Il correspond au processus *Lecture de la trace* de la Figure 7.4.

Cache parfait On note M^{μ}_{cache} le modèle microscopique mis en cache. Seuls trois paramètres déterminent comment générer M^{μ}_{cache} : l'intervalle de temps sélectionné, le nombre de tranches de temps, et bien entendu la métrique employée. Il est en effet facile d'ignorer, lors de la lecture du cache, des valeurs associées à des ressources ou des types d'évènements que l'on souhaite filtrer. C'est pourquoi nous sauvegardons le cache sans filtrage spatial ni évènementiel. M^{μ}_{cache} possède une granularité temporelle plus fine que ce qui peut être représenté par l'outil en utilisant la technique d'agrégation : $|T_{cache}| > |T_{visu}|$. Ce nombre de tranche de temps peut-être modifié dans les paramètres d'Ocelotl. De surcroît, lors de la première analyse qui va déclencher la génération du cache, si $|T_{cache}|$ n'est pas un multiple du nombre de tranches de temps $|T_{initial}|$ souhaité pour cette analyse, alors nous choisissons le plus petit multiple supérieur. Il est en effet simple de générer un modèle microscopique à partir d'un cache dont le nombre de tranches de temps est un multiple. Ceci est illustré par la Figure 7.11 a) : les attributs associés à chaque tranche de temps du modèle microscopique utilisé pour l'analyse sont obtenus en sommant ou en moyennant, selon la métrique utilisée, les attributs des tranches de temps correspondantes du cache. Dans cet exemple, chaque tranche de temps du modèle microscopique à générer est composée de deux tranches de temps du cache : $t_i = t'_{2i\ \ 1} + t'_{2i}$. L'intérêt de générer un cache avec un nombre de tranches de temps multiple à celui du modèle servant à la première analyse est d'augmenter la probabilité de sa réutilisabilité si l'on fait varier le nombre de tranches de temps lors d'une analyse suivante, dans le cas où l'on souhaite, par exemple, augmenter la précision de l'analyse ou de diminuer le temps de calcul : si $|T_{cache}|$ est un multiple de $|T_{initial}|$, la probabilité qu'il soit également un multiple de

a) Exemple de reconstruction d'un modèle microscopique en utilisant un cache parfait



b) Exemple de reconstruction d'un modèle microscopique à partir d'un cache imparfait (les tranches invalides sont indiquées en rouge)

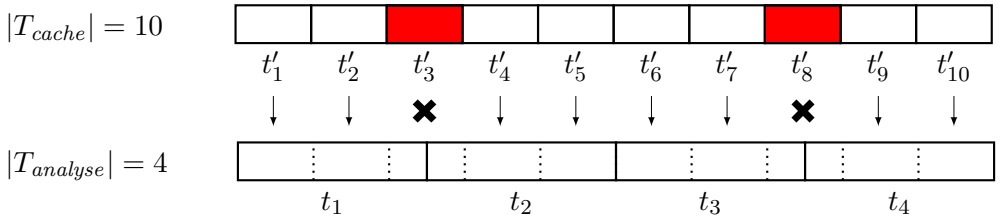


FIGURE 7.11 – Illustration de la reconstruction d'un modèle microscopique à partir d'un cache : a) le nombre de tranches de temps du cache est un multiple du nombre de tranches de temps du modèle microscopique à reconstruire. b) le nombre de tranches de temps du cache n'est pas un multiple et certaines tranches de temps du cache sont invalides (elles ne peuvent être réutilisées telles quelles).

$|T_{analyse_2}|$, choisi lors d'une nouvelle analyse, est plus grande que si $|T_{cache}| = |T_{initial}|$.

Modèle microscopique approximé La reconstruction du modèle microscopique fonctionne bien dans le cas où le nombre de tranches de temps du modèle microscopique en cache est un multiple du nombre de tranches de temps utilisé pour le modèle microscopique à générer pour l'analyse, d'où le fait que nous imposons cette contrainte lors de la première lecture d'une trace générant un cache. Cependant, quand cela n'est pas le cas, certaines tranches de temps du cache sont partagées entre deux tranches de temps du modèles microscopique. Ce phénomène est illustré par la Figure 7.11 b), où les tranches t'_3 et t'_8 chevauchent chacune deux tranches de temps et ne peuvent être utilisées telles quelles pour calculer les valeurs de ces tranches de temps. Nous appelons ces tranches du cache *tranches invalides*.

Nous proposons une solution fournissant une approximation de la valeur de ces tranches de temps. Pour cela, nous calculons la proportion du temps de la tranche invalide correspondant à celle du modèle à reconstruire. Puis, nous multiplions la valeur associée à cette tranche invalide par cette proportion. Selon le type de modèle à reconstruire, nous sommons ou moyennons cette valeur à ou avec celles associées aux autres tranches du cache utilisées pour reconstruire cette tranche de temps.

Cette stratégie possède seulement un léger surcoût en comparaison d'un cache sans tranches de temps invalides. Cependant, les approximations peuvent être responsables de différences entre les valeurs du modèle microscopique généré de cette manière et celles

calculées directement à partir des évènements de la trace, en particulier quand la proportion de tranches de temps invalides est forte et que la densité d'évènements est faible. Nous recommandons donc à l'utilisateur d'employer cette méthode avec précaution. Elle s'avère utile pour reproduire une séquence d'analyse rapidement, mais ne devrait pas être utilisée pour une analyse précise. Nous fournissons une option désactivant cette stratégie et n'autorisant que l'utilisation d'un cache parfait ou de la base de données. Il est aussi à noter que plusieurs caches avec des paramètres différents (comme le nombre de tranches de temps employées) peuvent être sauvegardés simultanément. Le cache le plus approprié (approximant le moins possible) est alors choisi en priorité.

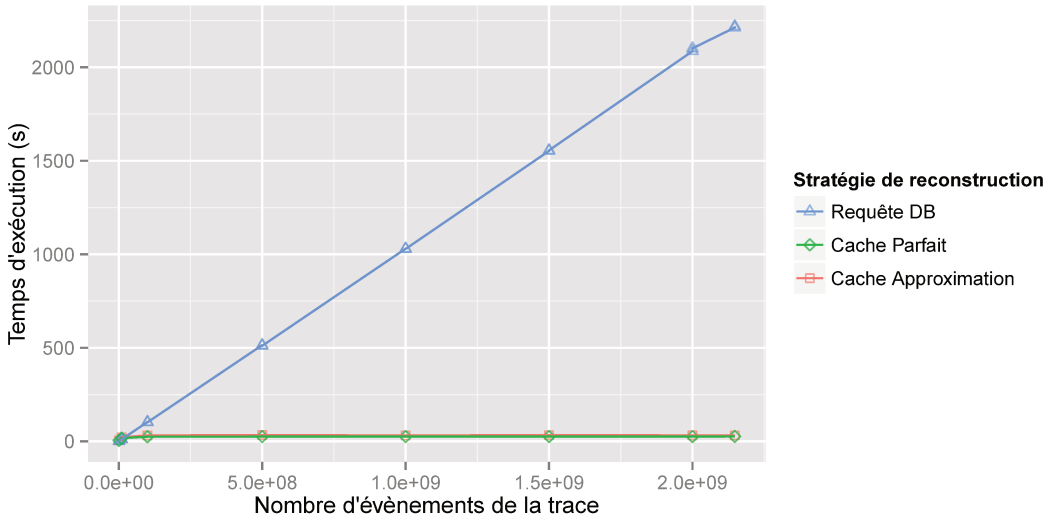


FIGURE 7.12 – Reconstruction du modèle microscopique en fonction du nombre d'évènements contenus dans la trace. On distingue plusieurs stratégies : les évènements sont lus grâce à une requête à la base de donnée, ou alors le modèle microscopique est généré à partir d'un cache. On différencie l'emploi d'un cache *parfait* d'un cache contenant des *tranches invalides*. Le cache généré possède 2000 tranches de temps.

Performances La Figure 7.12 indique le temps nécessaire à la reconstruction d'un modèle microscopique de 100 tranches de temps à partir de la base de données, d'un modèle microscopique de 100 tranches de temps à partir d'un cache de 2000 tranches de temps, c'est-à-dire sans approximation, ou d'un modèle microscopique de 90 tranches de temps à partir de ce même cache, c'est-à-dire avec approximation. Le gain substantiel dû à l'utilisation d'un cache y apparaît clairement.

7.4.2 Processus d'agrégation

Décomposition du traitement

Le processus d'agrégation est la seconde phase du processus d'analyse de la trace. Elle est constituée de plusieurs étapes successives.

1. Dans un premier temps, le **calcul des qualités** associe à chaque agrégat possible les valeurs de réduction de complexité et de perte d'information.
2. Ensuite, l'algorithme de **calcul des partitions optimales** retourne une liste de paramètres p associés à des partitions optimales $\mathcal{P}_p$ qui sont toutes différentes, ce qui permet ainsi de construire les courbes de réduction de complexité et de perte d'information (ou de gain en complexité et en information relatifs à la partition la plus agrégée) en fonction de p.
3. Enfin, l'outil sélectionne automatiquement une valeur de p par défaut (par exemple 1), et l'algorithme de **calcul des meilleures coupes** fournit la partition correspondante.

Après affichage, l'utilisateur a la possibilité de changer la valeur de p. Dans ce cas, seule la dernière étape est relancée.

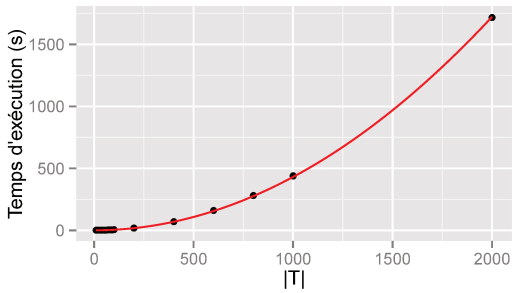
Complexité des algorithmes

Nous avons évalué expérimentalement la complexité des algorithmes. Pour cela nous lui fournissons en entrée des modèles microscopiques artificiels, générés en faisant varier, à l'aide de boucles imbriquées, $|T|$, $|S|$, $|\mathcal{J}(E)|$, et en assignant à chaque coefficient une valeur aléatoire tirée de manière uniforme entre 0 et `MAX_INTEGER` = 2147483647.

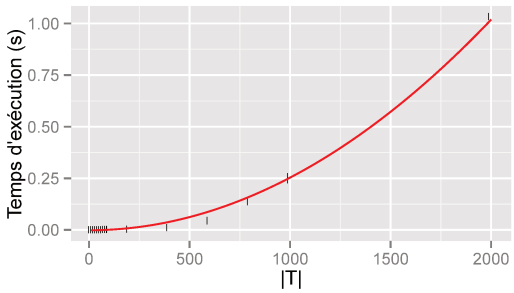
Agrégation temporelle Nous avons expérimentalement vérifié la complexité théorique de l'algorithme d'agrégation temporelle. La Figure 7.13 a) montre le temps de calcul des qualités, et sa dépendance quadratique au nombre de tranche de temps. Nous avons fixé le nombre de ressources $|S|$ à 1000 et de types d'évènements $|\mathcal{J}(E)|$ à 10. Le temps de calcul possède une dépendance linéaire à chacune de ces deux mesures : la Figure 7.14 a) met en évidence la dépendance linéaire au nombre de ressources, pour $|T| = 1000$ et $|\mathcal{J}(E)| = 10$. La Figure 7.13 b) représente le temps de calcul des meilleures partitions, et souligne sa dépendance quadratique au nombre de tranches de temps. Le seuil de profondeur de la récursivité est fixé à 0,001. Nous avons également fixé le nombre de ressources $|S|$ à 1000 et de types d'évènements $\mathcal{J}(E)$ à 10 . Pour rappel, nous avions borné sa complexité théorique à $\mathcal{O}(\log_2(\text{seuil}) \times |\boldsymbol{\mathcal{I}}(T)| \times |T|^2)$. La Figure 7.13 c) représente le temps de calcul des meilleures coupes, et sa dépendance quadratique au nombre de tranche de temps. Ici, l'influence de $|S|$ et $|\mathcal{J}(E)|$ n'est pas décelable. La Figure 7.13 d) met en évidence que le temps de calcul total de l'algorithme est majoritairement composée du temps de calcul des qualités.

Le temps d'exécution de l'algorithme peut donc être modulé principalement par le choix du nombre de tranches de temps, qui a une influence quadratique sur la complexité. Ce choix est un compromis entre la précision de l'analyse, le temps de calcul, mais aussi les capacités d'affichages. En pratique, nous choisissons une valeur dont l'ordre de grandeur est la centaine de tranches de temps, ce qui permet d'obtenir un résultat dans un temps interactif, et est suffisant pour détecter des anomalies dans nos cas d'études.

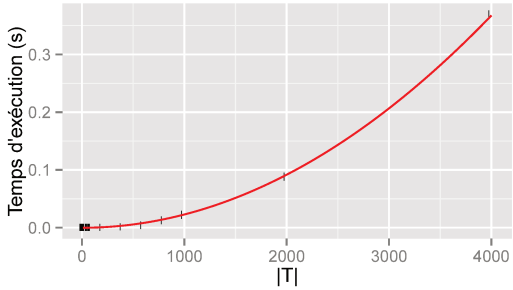
Agrégation spatiotemporelle Nous avons également expérimentalement vérifié la complexité théorique de l'algorithme d'agrégation spatiotemporelle. La Figure 7.15 a) montre le temps de calcul des qualités, pour $\mathcal{J}(E) = 10$. Nous montrons plusieurs arbres



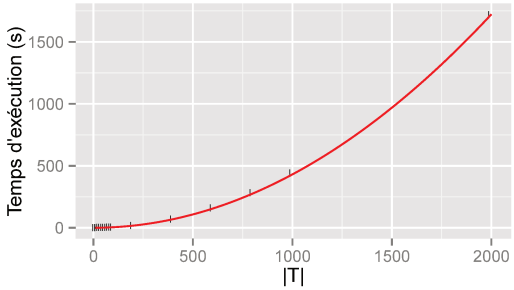
a) Temps de calcul des qualités ($|S| = 1000, |\mathcal{J}(E)| = 10$)



b) Temps de calcul des meilleures partitions (seuil $= 0,001$; $|S| = 1000; |\mathcal{J}(E)| = 10$)



c) Temps de calcul des meilleures coupes



d) Temps de calcul total

FIGURE 7.13 – Analyse temporelle : temps de calcul a) des qualités, b) des meilleures partitions, c) des meilleures coupes, d) total, en fonction du nombre de tranches de temps du modèle microscopique. En rouge, les courbes de régression polynomiale, toutes de la forme $y = ax^2$.

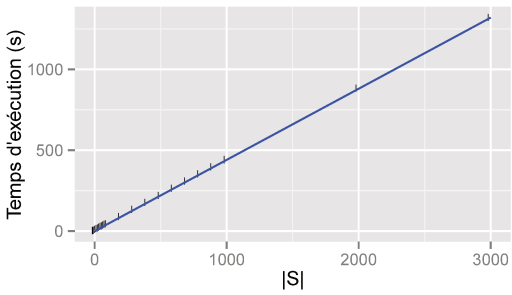


FIGURE 7.14 – Analyse temporelle : temps de calcul des qualités en fonction du nombre de ressources du modèle microscopique ($|T| = 1000; |\mathcal{J}(E)| = 10$). En bleu, la courbe de régression linéaire.

$\mathcal{H}(S)$ d'arité différente, dont la profondeur est fixée à 4 niveaux. Voici le nombre de nœuds de chaque arbre :

$$|H(S)_2| = 30 \qquad |H(S)_3| = 120 \qquad |H(S)_5| = 780 \qquad |H(S)_{10}| = 1111$$

Dans tous les cas, la courbe de régression polynomiale confirme la dépendance quadratique au nombre de tranches de temps. Le temps de calcul possède une également une dépendance linéaire au nombre de nœuds et au nombre de types d'évènements : la Figure 7.16 a) montre la dépendance linéaire au nombre de ressources, pour $|T| = 30$ et $|\mathcal{J}(E)| = 10$, pour des arbres $\mathcal{H}(S)$ d'arités différentes. Nous constatons par ailleurs la légère influence de l'arité sur le coefficient directeur de la courbe de régression linéaire. La Figure 7.15 b) représente le temps de calcul des meilleures partitions, et sa dépendance cubique au nombre de tranches de temps. Le seuil de profondeur de la récursivité est fixé à 0,001 et $\mathcal{J}(E)$ à 10. Nous montrons également quatre arbres $\mathcal{H}(S)$ d'arités différentes. Pour rappel, nous avions borné sa complexité théorique à $\mathcal{O}(\log_2(\text{seuil}) \times |\boldsymbol{\mathcal{A}}(S \times T)| \times |\mathcal{H}(S)| \times |T|^3)$. La Figure 7.15 c) représente le temps de calcul des meilleures coupes, et sa dépendance cubique au nombre de tranche de temps, également pour quatre hiérarchies matérielles, tandis que la Figure 7.16 b) fait ressortir sa dépendance linéaire au nombre de nœuds. Enfin, la Figure 7.15 d) montre que le temps de calcul total de l'algorithme est majoritairement composée du temps de calcul des meilleures partitions.

Le principal moyen d'agir sur le temps d'exécution est le choix du nombre de tranches de temps, de la même manière que pour l'algorithme temporel. De par son influence cubique, nous nous sommes cependant limités à une valeur de 30, déterminée empiriquement dans le cadre de nos cas d'études, ce qui limite la précision temporelle. Cette limite rend l'emploi complémentaire des algorithmes spatiotemporel et temporel intéressant, puisque ce dernier permettra une détection des comportement temporels plus fine.

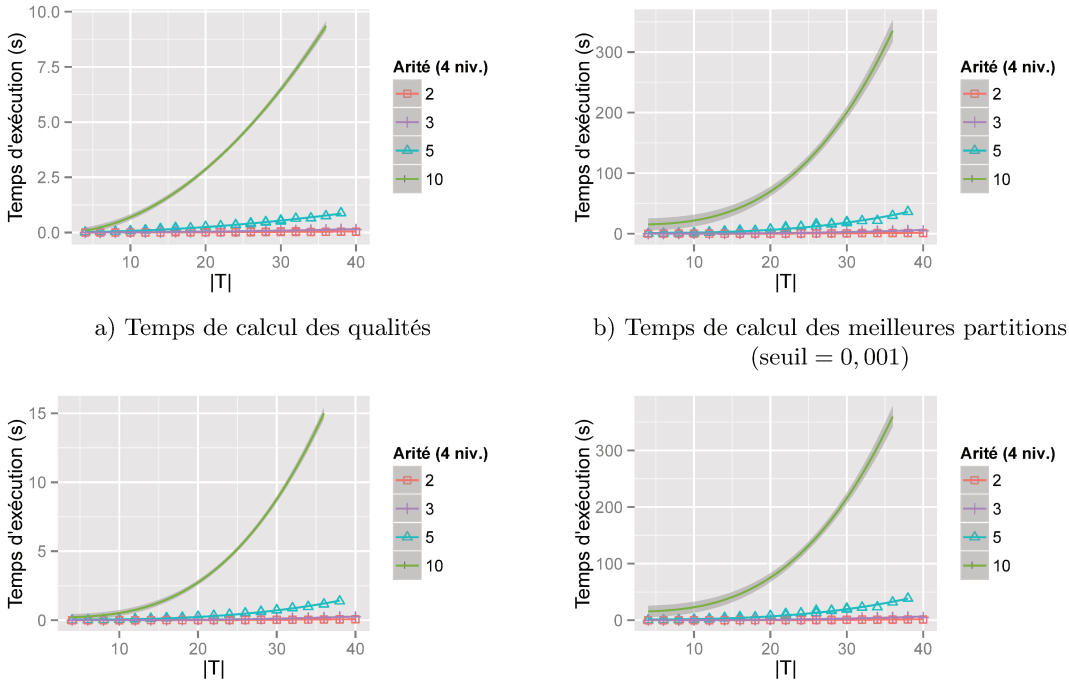
7.4.3 Rendu

Le rendu graphique est la dernière étape du traitement. Son temps dépend principalement du nombre d'objets graphiques à afficher, et est donc dépendant des tailles de dimensions temporelles et évènementielles (dans le cas de l'analyse temporelle), ainsi que de la taille de la dimension spatiale (dans le cas de l'analyse spatiotemporelle). Le type de représentation influe également le temps de calcul (l'histogramme et le mode spatiotemporel sont plus complexes que la partition). Enfin, la technique d'agrégation visuelle, et en particulier dans le cas spatiotemporel décrit Section 6.4.4, possède également un coût. La Table 7.1 donne des exemples du temps nécessaire pour reconstituer une partition à partir des meilleures coupes, et générer la représentation associée, en incluant le processus d'agrégation visuelle. La Figure 7.17 représente le processus de génération de la visualisation. Nous y avons inclus le calcul de la meilleure coupe de l'algorithme d'agrégation.

7.4.4 Performances globales et interactivité

Lecture et agrégation

Nous considérons que le premier contact avec une trace volumineuse n'est pas effectué dans un temps interactif (avec, par exemple, une durée de 20 minutes pour la lecture



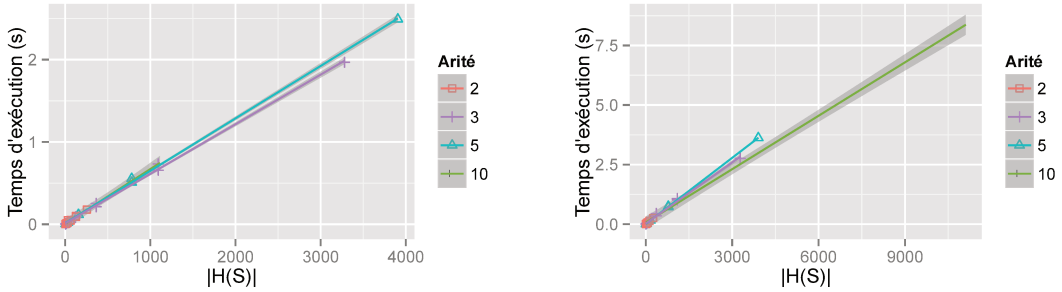
a) Temps de calcul des qualités

b) Temps de calcul des meilleures partitions (seuil $= 0,001$)

c) Temps de calcul des meilleures coupes

d) Temps de calcul total

FIGURE 7.15 – Analyse spatiotemporelle : temps de calcul a) des qualités, b) des meilleures partitions, c) des meilleures coupes, d) total, en fonction du nombre de tranches de temps du modèle microscopique (profondeur des arbres : 4 ; $|\mathcal{J}(E)| = 10$). On montre les courbes de régression polynomiale, de la forme a) $y = ax^2$, b) et c) $y = ax^3$, d) $y = ax^3 + bx^2$.



a) Temps de calcul des qualités

b) Temps de calcul des meilleures coupes

FIGURE 7.16 – Analyse spatiotemporelle : temps de calcul a) des qualités b) des meilleurs coupes, en fonction du nombre de nœuds de la hiérarchie du modèle microscopique ($|T| = 30$; $|\mathcal{J}(E)| = 10$). Les courbes de régression linéaire sont également affichées dans chaque cas.

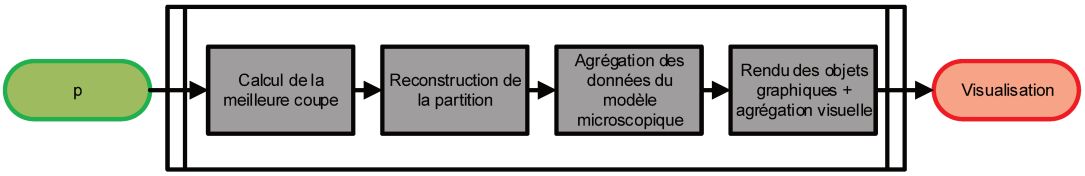


FIGURE 7.17 – Organigramme montrant le pipeline de génération de la visualisation. Il correspond au processus *Génération de la visualisation* de la Figure 7.4.

d'une trace de 1 milliards d'évènements). Néanmoins, nous outrepassons les limites de la mémoire que possèdent d'autres outils, comme Paraver [80] ou Pajé [30] grâce à l'utilisation de bases de données SQLite dont les résultats des requêtes sont stockés sur le disque dur. L'optimisation des requêtes est la technique la plus efficace pour réduire le temps de reconstruction du résultat. Le cache fournit, quant à lui, un confort d'utilisation non négligeable pour les sessions d'analyses suivantes, mais aussi pour certaines actions effectuées par l'utilisateur (zoom, filtrage, navigation *suivant*, *précédent*). Les étapes d'agrégation et de rendu ne dépendent pas de la taille initiale de la trace, mais principalement de ses dimensions spatiales et de types d'évènements. La complexité de ces algorithmes nécessite de trouver un compromis entre la précision temporelle, principal paramètre sur lequel agir, et l'interactivité.

Au niveau des limites de faisabilité de l'analyse obtenues en pratique, nous disposons d'un budget maximal dépendant des tailles des dimensions du modèle microscopique, et déterminé empiriquement sur la machine *flutin*. Dans le cas d'une analyse temporelle, ce budget est d'environ de dix milliards. Il est calculé de la manière suivante :

$$Budget_{t3} = 10^10 = |T|^2 \times |S| \times |\mathcal{J}(E)| \tag{7.2}$$

Cela nous permet, par exemple, d'analyser une trace de 100 000 ressources comportant 10 types d'évènements en discrétisant le temps en 100 tranches. Dans le cas d'une analyse spatiotemporelle, ce budget est d'environ 3 milliards :

$$Budget_{st2} = 3 \times 10^9 = |T|^3 \times |S| \times |\mathcal{J}(E)| \tag{7.3}$$

ce qui correspond à une trace de 10 000 ressources et de 10 types d'évènements, dont le temps est discrétisé en 30 tranches de temps.

Ces résultats placent Ocelotl parmi les outils gérant le mieux le passage à l'échelle selon l'état de l'art sur la visualisation de performances d'Isaacs *et al.* [3], qui indique, Figure 1, le volume de trace maximal (en évènement ou en taille de fichier) et le nombre de ressources pouvant être analysés. Seuls Vampir [48] et Scalasca [102], qui bénéficient d'une lecture parallèle des traces et de la possibilité d'une analyse à distance à partir d'un système distribué, sont capables de gérer des traces plus volumineuses (de l'ordre du téraoctet). Aucun outil ne permet l'analyse de plus de 100 000 ressources. Il est à noter que nos mesures ne prennent pas en compte la possibilité d'effectuer une pré-agrégation spatiale avant d'employer les algorithmes d'agrégation temporelle et spatiotemporelle.

La Table 7.1 récapitule, pour l'ensemble des cas d'études des deux Chapitres précédents (décrits Tables 5.1 et 5.2), les temps de calcul nécessaires pour chaque étape du pipeline d'analyse, dans le cas temporel et spatiotemporel, avec et sans cache.

TABLE 7.1 – Temps nécessaires à l'analyse des traces issus des cas d'études.

Cas d'étude	**Caractéristiques trace**			**Analyse temporelle ($\|T\| = 100$) (s)**					**Analyse spatiotemporelle ($\|T\| = 30$) (s)**				
	Taille	Évènements	Cache ($\|T\| = 900$)	M^μ (DB)	M^μ (cache)	Qual.	Dicho.	Partition + Rendu	M^μ (DB)	M^μ (cache)	Qual.	Dicho.	Partition + Rendu
GST-A	40,5 Mo	750045	1,5 Mo	1,08	0,432	1,377	0,004	1,098					
GST-B	40,0 Mo	749876	1,5 Mo	1,379	0,423	1,351	0,003	0,937					
GST-C	455,2 Mo	8290576	2 Mo	10,406	0,392	1,333	0,002	0,903					
GST-D	455,9 Go	8302874	2 Mo	13,901	0,391	1,33	0,002	0,879					
TSR-S	12,7 Mo	258519	0,3 Mo	0,241	0,014	0,049	0,003	0,158					
TSR-L	83,3 Mo	1642033	0,4 Mo	1,487	0,016	0,052	0,001	0,065					
UNI-H	7,0 Mo	149857	0,1 Mo	0,172	0,007	0,041	0,003	0,099					
UNI-B	27,7 Mo	324447	0,2 Mo	0,504	0,007	0,024	0,002	0,048					
CG.C-Gre-64	205,3 Mo	3838144	4 Mo	4,007	0,368	0,165	0,001	0,078	3,923	0,124	0,036	3,913	0,142
CG.C-Nan-64	205,0 Mo	3838144	3,4 Mo	3,874	0,094	0,16	0,0015	0,089	3,899	0,096	0,034	2,559	0,108
CG.C-Ren-64	205,8 Mo	3838144	4,1 Mo	3,874	0,119	0,161	0,001	0,089	3,941	0,118	0,038	2,881	0,135
CG.C-Gre-512	2,7 Go	49149440	17,4 Mo	54,313	0,852	1,191	0,001	0,234	50,916	0,737	0,251	27,396	0,98
CG.C-Nan-512	2,7 Go	49149440	25,5 Mo	50,931	1,164	1,306	0,0005	0,235	53,897	1,048	0,259	14,187	0,9185
LU.C-Gre-700	12,2 Go	218457456	23,2 Mo	223,224	1,262	1,735	0,001	0,361	225,467	1,168	0,356	30,214	1,5455
LU.C-Nan-700	12,3 Go	218457456	32,6 Mo	223,91	1,685	2,244	0,002	0,443	222,879	1,596	0,432	37,715	1,5765

Interaction

L'interaction profite globalement de toutes les techniques évoquées dans cette Section.

Le changement de paramètre p ne nécessite que le recalcul de la meilleure coupe, de la partition associée, et du rendu graphique. On peut le considérer comme interactif, puisque son temps d'exécution est de l'ordre de la seconde.

Le zoom nécessite de reconstruire le modèle microscopique et d'effectuer toutes les étapes de l'algorithme d'agrégation. Cependant, des contraintes imposées sur le zoom temporel (on ne peut sélectionner que des tranches de temps entières) augmentent la probabilité de réutiliser le cache, dont la granularité est plus fine que celle de l'analyse. Quant au zoom spatial, il est parfaitement compatible avec une réutilisation d'une sous-partie spatiale du cache. De manière générale, si le cache est réutilisé, l'interactivité est garantie. Si le cache ne convient pas, alors il faut reconstruire le modèle microscopique à partir d'une requête, ce qui diminue l'interactivité pour les traces volumineuses. Il est à noter qu'on peut regénérer un cache pour une sous-partie temporelle de la trace. Ainsi, l'ajout systématique, pour une même trace, de nouveaux caches avec des paramètres différents, augmente à long terme l'interactivité. L'analyse étant en général focalisée sur certaines sous-parties particulières de la trace, cette stratégie s'avère, en pratique, efficace.

Les actions suivant et précédent tirent parti, elles aussi, de la présence du cache, ce qui évite de relire la trace à l'aide d'une requête. L'intérêt est aussi d'éviter de stocker plusieurs modèles microscopiques en mémoire, ce qui pourrait mener à sa saturation.

La vue statistique associée à Ocelotl (Figure 7.2, composant D) utilise les données stockées dans le modèle microscopique afin d'éviter une requête vers la base de données, lorsque l'opérateur statistique sélectionné représente la même métrique que celle du modèle microscopique. Cela permet de la synchroniser avec la sélection sans latence. Cette technique est rendue possible grâce aux contraintes spatiotemporelles sur le zoom (sélection de tranches de temps entières et sélection hiérarchique).

7.5 Bilan

Nous avons présenté Ocelotl, un outil d'analyse qui implémente les différentes techniques d'agrégation et de visualisation que nous avons décrites dans les deux Chapitres précédents. Ocelotl présente plusieurs caractéristiques qui lui permettent de répondre aux différentes problématiques liées à l'analyse de traces. Nous distinguons celles relatives au passage à l'échelle, tant sur le plan visuel, calculatoire, que cognitif en ce qui concerne l'utilisateur, et celles liées au respect du mantra de Shneiderman, qui requiert une interaction efficace pour pouvoir gagner en niveau de détail au fur et à mesure de l'analyse sans perte du contexte. Dans cette optique, nous proposons différentes techniques d'interaction, comme le zoom, la commutation vers un diagramme espace-temps, ou encore des visualisations statistiques synchronisées. Ocelotl emploie un certain nombre de techniques afin d'optimiser les temps de calcul des différentes étapes du processus d'analyse. Nous proposons une optimisation des requêtes à la base de données contenant la trace afin de réduire le temps de récupération des évènements, tandis qu'un cache persistant permet d'éviter de relire la trace à chaque session d'analyse. Enfin, l'architecture modulaire d'Ocelotl offre à l'analyste la possibilité d'ajouter ses propres techniques d'agrégation et de visualisation sans avoir à modifier le code de l'outil. Les traces issues des cas d'études présentés dans les chapitres précédents (voir Tables 5.1 et 5.2) ont un volume maximal

(LU.C-Nan-700) de 218 millions d'évènements, ce qui correspond à 12,3 Go en format initial, et 15 Go après import dans Framesoc. Dans le cas de l'analyse temporelle, nous avons analysé avec succès des traces contenant environ 1500 ressources (les traces GST). Dans le cas de l'analyse spatiotemporelle, nous agrégeons efficacement deux traces contenant 700 ressources (LU.C-Nan-700 et LU.C-Gre-700). Néanmoins, Ocelotl est techniquement capable de lire des traces de 2 milliards d'évènements (environ 68 Go en base de données Framesoc), et possédant jusqu'à 10 000 ressources, dans le cas d'une analyse spatiotemporelle et 100 000 ressources, dans le cas d'une analyse temporelle, ce qui est validé grâce à des traces synthétiques. Ces deux mesures sont obtenues sans utiliser de pré-agrégation spatiale.

CHAPITRE 8

CONCLUSION

Les techniques de visualisation de traces sont fréquemment employées par les développeurs d'applications afin de comprendre, déboguer et optimiser leur code. En particulier, les représentations faisant intervenir la dimension temporelle tout en montrant la structure des ressources matérielles et logicielles impliquées dans l'exécution de l'application sont présentes dans la plupart des infrastructures d'analyse. Dans le domaine des applications embarquées multimédia, les développeurs cherchent à assurer une qualité de service relative au débit et à la résolution des flux audio et vidéo. Lorsque cette dernière n'est pas respectée et que des pertes d'images ou des artefacts sonores sont perceptibles, on recherche souvent des irrégularités dans la séquence des évènements, comme des durées d'interruptions ou de fonctions anormalement longues. Dans le domaine des applications HPC, le développeur s'intéresse à la représentation des différentes phases de l'application, au temps passé dans les communications et dans les phases de calcul et à l'adéquation de la plateforme avec les caractéristiques de l'application.

Néanmoins, ces techniques de visualisation spatiotemporelles n'assurent pas de manière satisfaisante le passage à l'échelle, c'est-à-dire qu'elles ne sont pas adaptées à la représentation du comportement global d'une trace volumineuse de plusieurs gigaoctets, contenant plusieurs centaines de millions d'évènements, et dont la structure des ressources de l'application est complexe et implique jusqu'à plusieurs millions d'entités. Cette vue d'ensemble, première étape du *mantra* de Shneiderman, « overview first, zoom and filter, then details-on-demand » [1], est nécessaire afin de fournir à l'analyste un point d'entrée destiné à identifier les zones de la trace sur lesquelles se concentrer.

Face à ce constat, nous avons élaboré une méthode d'analyse fondée sur la visualisation, faisant intervenir les dimensions temporelles et structurelles, passant à l'échelle et respectant le *mantra* de Shneiderman.

Nous résolvons les problèmes de passage à l'échelle de l'overview grâce à des techniques de réduction de complexité faisant intervenir des opérations d'agrégation. Après une analyse des solutions existantes dans le domaine de la visualisation de traces, et notamment une confrontation face à des critères de qualité de la représentation proposés par Elmqvist & Fekete, [4], nous arrivons à la conclusion que la *méthode de Lamarche-Perrin*, qui formalise un processus de partition et d'agrégation de données, se prête bien à notre problématique. Face à des techniques d'agrégation visuelle agrégeant uniformément

la représentation sans prendre en compte les caractéristiques des données, *la méthode de Lamarche-Perrin* cherche à diminuer la complexité de la représentation en conservant le maximum d'information utile possible. Pour cela, elle agrège en priorité les données les plus proches — au sens de la théorie de l'information —, ce qui apporte à l'analyste une connaissance supplémentaire sur le comportement d'un système *via* la discrimination de l'homogénéité du comportement de ses entités. L'utilisateur, par l'intermédiaire d'un coefficient, module le niveau de détail, grâce à un compromis entre la réduction de complexité souhaitée et la perte d'information tolérée. Cette démarche dynamique permet de faire émerger des phénomènes de nature et d'échelle différentes. Les étapes suivantes du *mantra* sont assurées grâce à notre implémentation, qui permet d'interagir avec l'analyse en zoomant, filtrant, et en obtenant plus de détails en synchronisant notre vue d'ensemble avec des représentations plus fines.

8.1 Bilan

Nous proposons une première contribution : une méthode d'analyse temporelle, pour laquelle nous avons adapté un algorithme d'agrégation temporel existant et fondé sur la *méthode de Lamarche-Perrin*. Le processus menant à l'élaboration de cette technique consiste à construire une abstraction de la trace, le *modèle microscopique*, qui puisse être agrégée par cet algorithme. Nous définissons la structure de cette abstraction et les métriques et opérations d'agrégation employées pour générer les valeurs qui lui sont associées, valeurs représentant les phénomènes que nous cherchons à discriminer dans la trace. Puis, Nous étendons l'algorithme à l'agrégation de valeurs multidimensionnelles. Enfin, nous proposons différentes techniques de visualisation de la sortie de cet algorithme, dont la plus aboutie est un histogramme empilé montrant la partition temporelle. Cette visualisation souffrant de défauts liés à l'affichage de valeurs de faibles amplitudes, nous employons une agrégation visuelle ponctuelle. Nous considérons ses inconvénients mineurs lorsqu'elle arrive à la fin de la chaine de traitement, par rapport aux représentations dont l'agrégation visuelle est le socle principal sur lequel repose la réduction de complexité. Nous appliquons la méthode d'analyse temporelle constituée de ces trois traitements successifs (la construction du modèle microscopique, la partition et l'agrégation, et enfin la visualisation) à l'examen du comportement de plusieurs cas d'études. Une première catégorie correspond à des applications multimédias dont les flux vidéo et audio sont perturbés, ce qui correspond à des scénarios classiques auxquels sont confrontés les industriels qui conçoivent ces applications. Nous démontrons qu'elle aide l'utilisateur à trouver la localisation temporelle de ces perturbations et à les connecter, au moyen d'une interaction avec une représentation plus détaillée, à la fonction, au processus, ou à la séquence d'évènements posant problème. Une second catégorie d'applications comprend des noyaux et des applications de calcul matriciel issues du benchmark NASPB, évaluant entre autres les performances de calcul et de communications de la plateforme sur laquelle elles sont exécutées. Les phases de l'application sont mises en évidence, et l'outil fournit une synthèse sur les durées des appels MPI, ce qui permet une comparaison entre plusieurs exécutions, et la détection de perturbations ponctuelles.

La seconde contribution est une méthode d'analyse spatiotemporelle pour laquelle nous avons conçu un nouvel algorithme d'agrégation. Comme pour la technique d'analyse temporelle, nous définissons la structure du modèle microscopique et son contenu,

et introduisons un formalisme du processus de partition et d'agrégation. Nous proposons également des techniques de visualisation de la sortie de l'algorithme, formées à partir d'une représentation matricielle à deux dimensions du système. La *méthode de Lamarche-Perrin* n'assurant pas une agrégation homogène de la représentation sur l'espace des ressources, nous la complétons par une agrégation visuelle ponctuelle, qui préserve une partie de l'information sur la partition temporelle des éléments agrégés grâce à l'emploi de plusieurs symboles différents. Nous procédons à l'analyse des applications parallèles NASPB préalablement examinées avec la technique d'analyse temporelle. Ce nouvel angle d'observation permet d'évaluer l'influence des ressources et de la topologie de la plateforme sur le comportement de l'application. On distingue des phénomènes globaux, concernant l'ensemble du système, de phénomènes locaux, touchant seulement une sous-partie des ressources. La présence de la dimension temporelle permet quant à elle de délimiter ces phénomènes : nous montrons des exemples de perturbations de courtes durées, ou d'un comportement hétérogène des ressources tout au long de l'exécution de l'application.

Enfin, la troisième contribution est l'implémentation de ces techniques au sein d'un outil de visualisation, Ocelotl. Nos choix de conception visent à assurer le passage à l'échelle en minimisant la durée des différents traitements survenant dans le processus d'analyse (lecture de la trace, étapes de l'algorithme d'agrégation), en exploitant, en particulier, le concept de modèle microscopique pour générer des caches persistants réutilisables pour les sessions d'analyse ultérieures. Mais nous nous attaquons aussi à l'intégration de la méthodologie de Shneiderman en implémentant des mécanismes d'interaction associés à chacune des étapes de son *mantra*. Enfin, la structure de l'outil permet d'ajouter à moindre effort de nouvelles métriques associées aux modèles microscopiques, de nouveaux algorithmes d'agrégation ou de nouvelles visualisations, sous la forme de greffons. Nous avons évalué les performances d'Ocelotl et les situons par rapport à l'état de l'art. L'analyse est réalisée avec succès sur des traces réelles de près de 700 processus, 218 millions d'évènements et 12 gigaoctets en 5 minutes lors d'une première analyse, et quelques secondes à partir des caches lors des analyses suivantes. Des traces synthétiques permettent d'évaluer les limites pratiques d'utilisation de l'outil à des traces de 2 milliards d'évènements (environ 68 gigaoctets), et jusqu'à 100000 ressources, sur un ordinateur personnel.

Nous affirmons donc que l'ensemble de ces trois contributions constitue une méthode d'analyse complète s'intégrant dans la méthodologie d'analyse descendante préconisée par Shneiderman, grâce à une vue d'ensemble temporelle et spatiotemporelle d'une trace de grand volume — résolvant les problèmes de la limitation de la surface d'affichage, des performances, et des limitations perceptives et cognitives de l'utilisateur —, et des interactions permettant de se focaliser sur une sous partie de la trace pour en obtenir plus de détails.

8.2 Perspectives

Les travaux détaillés dans cette thèse laissent envisager un grand nombre de perspectives, sur des aspects théoriques ou sur leur mise en œuvre.

8.2.1 Méthode d'agrégation d'information

La méthode de Lamarche-Perrin décrite dans le Chapitre 4 constitue un cadre générique. Les opérateurs d'agrégation (ici la somme), les mesures de qualité (entropie de Shannon, divergence de Kullback-Leibler), la structure des modèles microscopiques (ensemble ordonné, hiérarchique) et les contraintes définissant les ensembles des partitions admissibles sont des applications particulières de cette méthodologie. Ainsi, il est par exemple imaginable d'employer d'autres opérateurs d'agrégation que la somme, tels que ceux évoqués dans la Section 3.1.1 pour agréger des parties du modèle microscopique. Néanmoins, il y a une dépendance entre cette opération et les mesures de qualité employées, qu'il faut alors redéfinir en fonction de l'opérateur d'agrégation.

Les perspectives évoquées dans la thèse de Lamarche-Perrin [8] prévoient l'élaboration de nouveaux algorithmes d'agrégation. L'agrégation de graphes est notamment citée, et serait idéale pour étudier des graphes de dépendance de tâches complexes, par exemple, ou permettrait d'outrepasser certaines limitations liées à l'emploi de hiérarchies pour structurer les ressources. D'autres techniques sont également envisageables, comme l'agrégation de hiérarchies entrelacées (à plusieurs sommets), qui permettrait de séparer l'influence des ressources matérielles et logicielles lors de l'analyse. L'agrégation de surfaces correspondant à une agrégation de deux dimensions ordonnées s'avérerait, quant à elle, utile pour l'analyse des allocations mémoire, par exemple. Néanmoins, l'algorithme de recherche d'une partition optimale associé est de complexité exponentielle lorsque l'on contraint les parties à être des rectangles. Il est donc nécessaire d'exprimer une contrainte plus forte, ou de faire appel à des heuristiques. Des combinaisons à plus de deux dimensions ordonnées ou hiérarchiques sont également imaginables.

Nous avons, dans la Section 5.2, évoqué un certain nombre de métriques associées aux attributs du modèle microscopique (quantité d'évènements, durée totale des types d'états par tranche de temps, durée moyenne des types d'états, etc.) et généré à l'aide d'une opération d'agrégation. D'autres métriques sont possibles. Nous pensons, par exemple, en ne citant que les plus simples, au temps maximum, minimum, ou médium d'un type d'état sur une tranche de temps (voir les opérations d'agrégation données en exemple de la Section 3.1.1). Dans le cas où la technique d'agrégation employée par la *méthode de Lamarche-Perrin* est une somme suivie d'une redistribution uniforme (ce qui équivaut en pratique à une moyenne), il est nécessaire que les métriques associées au modèle microscopique soient sommables, ce qui exclut les variables booléennes (pour lesquelles il faut alors redéfinir l'opération d'agrégation dans la *méthode de Lamarche-Perrin*).

Concernant l'utilisation des courbes de qualités, un des axes d'améliorations possible est l'aide à la détection des niveaux d'abstraction, en utilisant une méthode automatique, ce qui permettrait une sélection des partitions les plus intéressantes pour l'utilisateur.

Dans le cas de l'agrégation spatiotemporelle, les mesures de qualités des partitions possèdent des amplitudes élevées à cause du nombre d'individus du modèle microscopique impliqués ($|S| \times |T|$). Cela peut être gênant car des approximations sur les calculs de flottants se cumulent et finissent par fausser les résultats des comparaisons lors du calcul des meilleures coupes. Cela nous empêche aussi de normaliser les mesures de qualités. Il est envisagé de replacer ces mesures sur une échelle plus restreinte pour pallier à ces inconvénients (en employant, par exemple, une échelle logarithmique, seyant mieux à la représentation de l'information contenue dans des systèmes aussi volumineux).

8.2.2 Méthode d'agrégation visuelle

La réduction de complexité proposée dans nos méthodes est inhomogène puisque les processus de partition et d'agrégation dépendent des valeurs des attributs du modèle microscopique. Elle n'assure pas intégralement le passage à l'échelle de la visualisation dans le cas où ce modèle microscopique ne peut pas être représenté intégralement à cause de la taille des dimensions qui le composent. La compléter avec de l'agrégation visuelle fournit une représentation qui ne souffre pas des artefacts dus à la limitation de l'écran. Ce procédé transmet également de l'information sur le contenu de l'agrégat, en particulier dans le cas spatiotemporel où la symbolique employée décrit le découpage temporel à l'intérieur de celui-ci. Nous préconisons toutefois d'améliorer cette technique, grâce à l'emploi de textures procédurales d'apparences pseudo-aléatoire, comme le bruit de Perlin [103], qui pourraient permettre de visualiser la complexité ou l'information contenue dans un agrégat visuel. Holten *et al.* [104] énumèrent un certain nombre de méthodes de génération de textures colorées dont les paramètres permettent de représenter une ou plusieurs variables, et qui pourraient être employées pour représenter la valeur d'une des mesures de qualité au sein de l'agrégat visuel.

D'autres aspects concernant la visualisation peuvent se montrer intéressants : hiérarchiser les types d'évènements, grâce à une ontologie par exemple, peut permettre de meilleures représentations, en particulier pour l'analyse spatiotemporelle et sa visualisation modale.

8.2.3 Performances

Les performances des techniques d'agrégation, liées à la complexité des algorithmes mis en jeux, et de l'outil d'analyse, de manière générale, sont un critère important, comme nous l'avons évoqué tout au long de cette thèse. Nous distinguons plusieurs axes d'améliorations.

Parallélisation

Des techniques de parallélisation sont déjà employées dans la reconstruction du modèle microscopique à partir de la trace (les évènements issus d'une requête sont lus alors que le résultat est en cours de reconstruction, et le modèle est généré en *streaming* par plusieurs tâches parallèles). D'autres parties du traitement sont parallélisables : cela concerne le calcul des qualités des algorithmes d'agrégation, en particulier pour des modèles multidimensionnels, où il n'y a pas de dépendances entre les qualités liées aux individus associés à certaines dimensions (comme les types d'évènements, par exemple). L'algorithme de recherche des partitions optimales employant une dichotomie est lui aussi parallélisable, chaque nouvelle branche dans la récursion pouvant être attribuée à une tâche ou un thread différent. D'après la loi d'Amdahl, nous pouvons espérer gagner au mieux un ordre de grandeur dans le temps de traitement (pour une parallélisation à une dizaine de tâches et une exécution sur un ordinateur de bureau possédant une dizaine de cœurs), ce qui améliorera l'interactivité et permettra de gagner en finesse sur l'analyse (en particulier dans le cas de l'analyse spatiotemporelle). À plus long terme, il semble judicieux de déporter la lecture de la trace et les calculs (comme l'agrégation) sur des systèmes distribués. Une lecture de trace en parallèle, telle qu'employée par Vampir [48], est efficace pour des volumes de plusieurs teraoctets. Il est, de surcroît, possible de décomposer la

trace en plusieurs fichiers (ou bases de données) selon une dimension particulière (comme les ressources). La génération des modèles microscopiques est alors réalisée en combinant plusieurs sous-modèles indépendants dont chacun est associé à un fichier de la trace, ce qui minimise les dépendances entre les threads générant le modèle microscopique, et la synchronisation lors de l'écriture des attributs de ce dernier.

Structures de données

Les modèles microscopiques formant l'abstraction de la trace sont des matrices creuses, à plus de 99% de 0 dans les cas d'études présentés dans cette thèse. Néanmoins, mis à part lorsque l'on génère le cache permanent associé à une trace, où ne sont sauvegardés que les attributs du modèle microscopiques non nuls, cette propriété n'est pas exploitée par Ocelotl lorsque le modèle microscopique est mis en mémoire. Cela peut aboutir à saturer cette dernière si le modèle microscopique à générer est très détaillé (par exemple avec une quantité de tranches de temps particulièrement élevée). Cette propriété n'est pas non plus prise en considération par la bibliothèque d'agrégation *lpaggreg* lors du calcul des qualités. Savoir qu'un sous-ensemble du modèle microscopique est constitué uniquement de 0 permettrait d'éviter le calcul de la somme des valeurs des attributs et des mesures de qualité associées à chaque partie admissible de ce sous-ensemble, toutes nulles.

Contrôle dynamique de l'application

Les contraintes liées à l'interaction nécessitent d'optimiser chaque étape du flot d'analyse. Parmi les solutions envisagées, nous envisageons la sauvegarde des données calculées lors des sessions d'analyse précédentes, sur le même principe que la sauvegarde du modèle microscopique. Nous prévoyons par exemple l'ajout, dans Ocelotl, d'un cache supplémentaire contenant les paramètres p obtenus par l'algorithme de calcul des partitions optimales et les valeurs des courbes associées, lié à un contexte d'analyse constitué d'une trace et de plusieurs paramètres (zoom temporel, filtrage, normalisation, etc.). Le gain portera principalement sur l'agrégation spatiotemporelle dont le temps de calcul de cette étape est perceptible par l'utilisateur. Le calcul des qualités lié à un contexte d'analyse peut également être sauvegardé. Dans ce cas, on étendra la bibliothèque *lpaggreg* pour pouvoir fournir les qualités pré-calculées plutôt que le modèle microscopique. Des méthodes fondées sur la prélecture et l'anticipation des actions de l'utilisateur, profitant des périodes de temps où le programme est en attente d'interaction, réduiraient également les temps de calcul. Il peut également être envisagé de générer plusieurs modèles microscopiques parmi ceux dont les métriques sont les plus couramment employées dès l'importation d'une trace dans Framesoc, plutôt que d'attendre une première analyse avec Ocelotl.

8.2.4 Applications

Les techniques d'agrégation que nous avons mis au point sont génériques : elles peuvent être appliquées à l'analyse de différents types de systèmes.

Combinaison avec la recherche de motifs locaux

L'analyse des cas d'études issus de SoC-Trace utilise Ocelotl en amont, pour employer un diagramme espace-temps focalisé sur une sous-partie de la trace par la suite. D'autres flots d'analyses sont envisageables. L'outil MegaLog, développé par ProbaYes, et également implémenté sous la forme d'une plug-in Framesoc, permet de détecter des séquences d'évènements en se basant sur un langage de requête dont la syntaxe est proche de SQL. La sortie de MegaLog est une suite de motifs ordonnés, abstrayant les évènements bas niveau de la trace, et représentés sous la forme d'états. MegaLog dispose de sa propre interface de visualisation des résultats, mais utilise les capacités d'agrégation d'Ocelotl afin d'assurer le passage à l'échelle de la représentation lorsque le nombre de motifs est élevé.

Comparaison de traces

La comparaison de traces est une tâche importante pour valider le comportement d'une simulation par rapport à un système réel [105]. Ocelotl est adapté à une analyse à gros grain des différences entre deux traces, comme nous l'avons illustré avec la comparaison d'exécutions d'applications parallèles sur des plateformes différentes. Néanmoins, il semble nécessaire de compléter cette analyse qualitative avec des méthodes plus fines, inspirées des recherches de séquences ADN [106], par exemple. Ces algorithmes étant complexes, notre proposition est de l'assister grâce à l'analyse haut-niveau d'Ocelotl, en établissant des zones de correspondance entre les deux traces, et en appliquant l'algorithme de séquençage uniquement sur ces sous-parties.

Analyse de la mémoire

L'agrégation spatiotemporelle est actuellement employée dans les travaux de David Beniamine, doctorant au LIG ayant développé un logiciel, HeapInfo, permettant l'analyse des allocations mémoire d'applications parallèles [107]. La visualisation employée à l'origine est une cartographie de la mémoire générée de manière statique et souffrant de défauts de passage à l'échelle, produisant des artefacts du fait de la quantité d'objets graphiques. L'utilisation d'Ocelotl et d'une structuration de la mémoire sous la forme d'une hiérarchie permet une analyse sans ces inconvénients.

Visualisation d'information

Dans la Section 4.3, nous avons évoqué des exemples d'applications géographiques des algorithmes originaux développés par Lamarche-Perrin. Un grand nombre de systèmes peuvent en fait être représentés sous la forme d'un système multidimensionnel faisant intervenir le temps, un espace de ressources produisant des évènements, ou les deux. Les techniques que nous proposons peuvent ainsi trouver des applications dans la gestion de gros volumes de ressources (humaines, stocks de matériels), ou encore dans l'analyse de données économiques, financières ou démographiques.

BIBLIOGRAPHIE

[1] B. Shneiderman. The eyes have it : A task by data type taxonomy for information visualizations. In *Visual Languages, 1996. Proceedings., IEEE Symposium on*, pages 336–343, 1996.

[2] Lucas Mello Schnorr, Arnaud Legrand, and Jean-Marc Vincent. Detection and analysis of resource usage anomalies in large distributed systems through multi-scale visualization. *Concurrency and Computation : Practice and Experience*, pages n/a–n/a, November 2011.

[3] Katherine E. Isaacs, Alfredo Giménez, Ilir Jusufi, Todd Gamblin, Abhinav Bhatele, Martin Schulz, Bernd Hamann, and Peer-Timo Bremer. State of the Art of Performance Visualization. In *Proceedings of Eurographics Conference on Visualization (EuroVis)*. R. Borgo, R. Maciejewski, and I. Viola, 2014.

[4] N. Elmqvist and J. D Fekete. Hierarchical aggregation for information visualization : Overview, techniques, and design guidelines. *Visualization and Computer Graphics, IEEE Transactions on*, 16(3) :439–454, 2010.

[5] R. Lamarche-Perrin, L.M. Schnorr, J.-M. Vincent, and Y. Demazeau. Evaluating Trace Aggregation for Performance Visualization of Large Distributed Systems. In *2014 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 139–140, March 2014.

[6] Robin Lamarche-Perrin, Lucas Mello Schnorr, Jean-Marc Vincent, and Yves Demazeau. Agrégation de traces pour la visualisation de grands systèmes distribués. *Technique et Science Informatiques*, 2014.

[7] C. Prada-Rojas, V. Marangozova-Martin, K. Georgiev, J. F. Mehaut, and M. Santana. Towards a Component-based Observation of MPSoC. In *Parallel Processing Workshops, 2009. ICPPW'09. International Conference on*, pages 542–549, 2009.

[8] Robin Lamarche-Perrin. *Analyse macroscopique des grands systèmes : émergence épistémique et agrégation spatio-temporelle*. PhD thesis, Université de Grenoble, October 2013.

[9] Robin Lamarche-Perrin, Yves Demazeau, and Jean-Marc Vincent. A Generic Algorithmic Framework to Solve Special Versions of the Set Partitioning Problem.

In *2014 IEEE 26th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 891–897, November 2014.

[10] Damien Dosimont, Robin Lamarche-Perrin, Lucas Mello Schnorr, Guillaume Huard, and Jean-Marc Vincent. A Spatiotemporal Data Aggregation Technique for Performance Analysis of Large-scale Execution Traces. In *Proceedings of the 2014 IEEE International Conference on Cluster Computing (CLUSTER'14)*, Madrid, Spain, September 2014.

[11] Damien Dosimont, Generoso Pagano, Guillaume Huard, Vania Marangozova-Martin, and Jean-Marc Vincent. Efficient Analysis Methodology for Huge Application Traces. In *Proceedings of the 2014 International Conference on High Performance Computing & Simulation (HPCS)*, Bologna, Italia, July 2014.

[12] Generoso Pagano, Damien Dosimont, Guillaume Huard, Vania Marangozova-Martin, and Jean-Marc Vincent. Trace Management and Analysis for Embedded Systems. In *2013 IEEE 7th International Symposium on Embedded Multicore Socs (MCSoC)*, pages 119–122, Tokyo, Japan, September 2013.

[13] Damien Dosimont, Youenn Corre, Lucas Mello Schnorr, Guillaume Huard, and Jean-Marc Vincent. Ocelotl : Large Trace Overviews Based on Multidimensional Data Aggregation. In *Proceedings of the 8th International Parallel Tools Workshop*, Stuttgart, Germany, October 2014.

[14] Damien Dosimont, Guillaume Huard, and Jean-Marc Vincent. La visualisation de traces, support à l'analyse, déverminage et optimisation d'applications de calcul haute performance. In *Actes de l'Atelier Visualisation d'informations, interaction et fouille de données de ECG'2013 - 13e Conférence Francophone sur l'Extraction et la Gestion des Connaissances*, January 2013.

[15] Damien Dosimont, Guillaume Huard, and Jean-Marc Vincent. Agrégation temporelle pour l'analyse de traces volumineuses. In *Actes du 10ème Atelier en Évaluation de Performances*, Sophia Antipolis, France, June 2014.

[16] Damien Dosimont, Lucas Mello Schnorr, Guillaume Huard, and Jean-Marc Vincent. A Trace Macroscopic Description based on Time Aggregation. Technical Report HAL RR-8524, Inria, April 2014.

[17] Generoso Pagano, Damien Dosimont, Guillaume Huard, Vania Marangozova-Martin, and Jean-Marc Vincent. Trace Management and Analysis for Embedded Systems. Technical Report HAL RR-8304, Inria, May 2013.

[18] Generoso Pagano and Vania Marangozova-Martin. The frameSoC Software Architecture for Multiple-view Trace Data Analysis. In *Proceedings of the 2014 ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, EICS '14, pages 217–222, New York, NY, USA, 2014. ACM.

[19] C.K. Kengne, N. Ibrahim, M.-C. Rousset, and M. Tchuente. Distance-Based Trace Diagnosis for Multimedia Applications : Help Me TED ! In *2013 IEEE Seventh International Conference on Semantic Computing (ICSC)*, pages 306–309, September 2013.

[20] Maurice Herlihy and Nir Shavit. *The art of multiprocessor programming.* Elsevier/Morgan Kaufmann, Amsterdam ; London, 2008.

[21] A.T. Campbell, G. Coulson, and M.E. Kounavis. Managing complexity : middleware explained. *IT Professional*, 1(5) :22 –28, October 1999.

[22] Kai Hwang, J. J Dongarra, and Geoffrey C Fox. *Distributed and cloud computing*. Elsevier/Morgan Kaufmann, Amsterdam ; London, 2012.

[23] Michael J Quinn. *Parallel programming in C with MPI and OpenMP*. McGraw-Hill, Boston [etc.], 2004.

[24] J. Chassin de Kergommeaux, É Maillet, and J.-M. Vincent. Parallel Program Development for Cluster Computing. In *Parallel program development for cluster computing*, pages 131–150. Nova Science Publishers, Inc., Commack, NY, USA, 2001.

[25] Score-P – HPC Profiling and Event Tracing Infrastructure. http ://www.vi-hps.org/projects/score-p, 2015.

[26] S. S. Shende. The Tau Parallel Performance System. *International Journal of High Performance Computing Applications*, 20(2) :287–311, May 2006.

[27] D. Toupin. Using tracing to diagnose or monitor systems. *Software, IEEE*, 28(1) :87–91, 2011.

[28] P. M. Fournier, M. Desnoyers, and M. R. Dagenais. Combined tracing of the kernel and applications with LTTng. In *Proceedings of the 2009 Linux Symposium*, 2009.

[29] Eschweiler Dominic, Wagner Michael, Geimer Markus, Kn&uuml ;pfer Andreas, Nagel Wolfgang E, and Wolf Felix. Open Trace Format 2 : The Next Generation of Scalable Trace Formats and Support Libraries. *Advances in Parallel Computing*, pages 481–490, 2012.

[30] J Chassin de Kergommeaux. Pajé, an interactive visualization tool for tuning multi-threaded parallel applications. *Parallel Computing*, 26(10) :1253–1274, August 2000.

[31] Lucas Mello Schnorr, Benhur de Oliveira Stein, and Jacques Chassin de Kergommeaux. Paje Trace File Format, Version 1.2.5. Technical Report, Laboratoire d'Informatique de Grenoble, France, February 2013.

[32] D.A. Reed, R.A. Aydt, T.M. Madhyastha, R.J. Noe, K.A. Shields, and B.W. Schwartz. The Pablo performance analysis environment. *Dept. of Comp. Sci., Univ. of Ill*, 1992.

[33] Paraver Tracefile Description Version 3.0, June 2001.

[34] Mathieu Desnoyers. Common Trace Format (CTF) Specification (v1.8.2). http ://git.efficios.com/ ?p=ctf.git ;a=blob_plain ;f=common-trace-format-specification.txt ;hb=master, 2012.

[35] Vania Marangozova-Martin and Generoso Pagano. SoC-TRACE : Handling the Challenge of Embedded Software Design and Optimization. In *Proceedings of the Posters and Demo Track*, Middleware '12, pages 10 :1–10 :2, New York, NY, USA, 2012. ACM.

[36] Dynamic system tracing with KPTrace | STLinux. http ://www.stlinux.com/devel/traceprofile/kptrace, 2015.

[37] Daniel Becker, Rolf Rabenseifner, Felix Wolf, and John C. Linford. Scalable timestamp synchronization for event traces of message-passing applications. *Parallel Computing*, 35(12) :595 – 607, 2009.

[38] K. A. Huck and A. D. Malony. Perfexplorer : A performance data mining framework for large-scale parallel computing. In *Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, page 41, 2005.

[39] M. T Heath and J. A Etheridge. Visualizing the performance of parallel programs. *IEEE Software*, 8(5) :29–39, September 1991.

[40] B. P Miller, M. D Callaghan, J. M Cargille, J. K Hollingsworth, R. B Irvin, K. L Karavanic, K. Kunchithapadam, and T. Newhall. The Paradyn parallel performance measurement tool. *Computer*, 28(11) :37–46, November 1995.

[41] Robert Bell, Allen D. Malony, and Sameer Shende. ParaProf : A Portable, Extensible, and Scalable Tool for Parallel Performance Profile Analysis. In Gerhard Goos, Juris Hartmanis, Jan Leeuwen, Harald Kosch, László Böszörményi, and Hermann Hellwagner, editors, *Euro-Par 2003 Parallel Processing*, volume 2790, pages 17–26. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[42] Lucas Mello Schnorr. *Quelques Modèles de Visualisation pour l'Analyse des Applications Parallèles*. PhD thesis, Institut Polytechnique de Grenoble, 2009.

[43] J.M. Wilson. Gantt charts : A centenary appreciation. *European Journal of Operational Research*, 149(2) :430–437, 2003.

[44] Eclipse LTTng plugin | LTTng Project. http ://lttng.org/eclipse, 2012.

[45] ARM Streamline Performance Analyzer - ARM. http ://www.arm.com/products/tools/software-tools/ds-5/streamline.php, 2012.

[46] E. Shaffer and D.A. Reed. Real-time immersive performance visualization and steering. *ACM SIGGRAPH Computer Graphics*, 34(2) :11–14, 2000.

[47] B. Cornelissen, A. Zaidman, D. Holten, L. Moonen, A. Van Deursen, and J.J. van Wijk. Execution trace analysis through massive sequence and circular bundle views. *Journal of Systems and Software*, 81(12) :2252–2268, 2008.

[48] Holger Brunst, Daniel Hackenberg, Guido Juckeland, and Heide Rohling. Comprehensive Performance Tracking with Vampir 7. In Matthias S. Müller, Michael M. Resch, Alexander Schulz, and Wolfgang E. Nagel, editors, *Tools for High Performance Computing 2009*, pages 17–29. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[49] L.M. Schnorr, A. Legrand, and J.-M. Vincent. Interactive analysis of large distributed systems with scalable topology-based visualization. In *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 64–73, April 2013.

[50] Lucas Mello Schnorr, Arnaud Legrand, and Jean-Marc Vincent. Detection and Analysis of Resource Usage Anomalies in Large Distributed Systems Through Multi-Scale Visualization. *Concurrency and Computation : Practice and Experience*, 24(15) :1792–1816, 2012.

[51] Ben Shneiderman. Tree visualization with Tree-maps : A 2-d space-filling approach. *ACM Transactions on Graphics*, 11 :92–99, 1991.

[52] Lucas Mello Schnorr, Guillaume Huard, and Philippe Olivier Alexandre Navaux. A hierarchical aggregation model to achieve visualization scalability in the analysis of parallel applications. *Parallel Computing*, 38(3) :91–110, March 2012.

[53] M. Detyniecki. Fundamentals on aggregation operators. *This manuscript is based on Detyniecki's doctoral thesis and can be downloaded from*, 2001.

[54] G Mayor and E Trillas. On the representation of some aggregation functions. In *Proceeding of ISMVL*, pages 111–114, 1986.

[55] Sergei Ovchinnikov. On Robust Aggregation Procedures. In Dr Bernadette Bouchon-Meunier, editor, *Aggregation and Fusion of Imperfect Information*, number 12 in Studies in Fuzziness and Soft Computing, pages 3–10. Physica-Verlag HD, January 1998.

[56] R Mesiar and M Komorníková. Aggregation operators. In *Proceeding of the XI Conference on applied Mathematics PRIM'96*, pages 193–211. Institute of Mathematics, Novi Sad, 1997.

[57] J. C. Fodor and M. R. Roubens. *Fuzzy Preference Modelling and Multicriteria Decision Support*. Springer, October 1994.

[58] Michel Grabisch, Sergei A. Orlovski, and Ronald R. Yager. Fuzzy Aggregation of Numerical Preferences. In Roman Słowiński, editor, *Fuzzy Sets in Decision Analysis, Operations Research and Statistics*, number 1 in The Handbooks of Fuzzy Sets Series, pages 31–68. Springer US, January 1998.

[59] M. Grabisch and P. Perny. Agrégation multicritère. In B. Bouchon and C. Marsala, editors, *Utilisations de la logique floue*. Hermès, 1999.

[60] R. R. Yager and A. Rybalov. Full reinforcement operators in aggregation techniques. *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society*, 28(6) :757–769, 1998.

[61] Gustave Choquet. Theory of capacities. *Annales de l'institut Fourier*, 5 :131–295, 1954.

[62] Roger M. Cooke. *Experts in uncertainty : opinion and subjective probability in science*. Environmental ethics and science policy series. Oxford University Press, New York, 1991.

[63] David H Krantz, R Duncan Luce, Patrick Suppes, and Amos Tversky. *Foundations of Measurement : Vol. 1 : Additive and Polynomial Representations*. Academic Press New York, 1971.

[64] Masaharu Mizumoto. Pictorial representations of fuzzy connectives, part I : cases of t-norms, t-conorms and averaging operators. *Fuzzy Sets and Systems*, 31(2) :217–242, 1989.

[65] Andreï Kolmogorov. Sur la notion de la moyenne. In *Sur la notion de la moyenne*, volume Volume 12, Issue 6, pages 388–391. e, Atti della R. Academia nazionale dei Lincei, 1930.

[66] Joseph Aczél. Lectures on functional equations and their applications. *New York : Academic Press, 1966, edited by Oser, Hansjorg*, 1, 1966.

[67] János Aczél. On mean values. *Bulletin of the American Mathematical Society*, 54(4) :392–400, 1948.

[68] T. Calvo and R. Mesiar. Generalized medians. *Fuzzy Sets and Systems*, 124(1) :59–64, November 2001.

[69] D. Dubois and H. Prade. Weighted minimum and maximum operations in fuzzy set theory. *Information Sciences*, 39(2) :205–210, September 1986.

[70] Ronald R Yager. On ordered weighted averaging aggregation operators in multicriteria decisionmaking. *Systems, Man and Cybernetics, IEEE Transactions on*, 18(1) :183–190, 1988.

[71] Didier Dubois, Hélène Fargier, and Henri Prade. Beyond Min Aggregation in Multicriteria Decision : (Ordered) Weighted Min, Discri-Min, Leximin. In Ronald R. Yager and Janusz Kacprzyk, editors, *The Ordered Weighted Averaging Operators*, pages 181–192. Springer US, January 1997.

[72] Michio Sugeno. *Theory of fuzzy integrals and its applications.* Tokyo Institute of Technology, 1974.

[73] Toshiaki Murofushi and Michio Sugeno. A theory of fuzzy measures : representations, the Choquet integral, and null sets. *Journal of Mathematical Analysis and Applications*, 159(2) :532–549, 1991.

[74] Dieter Denneberg. *Non-additive measure and integral.* Kluwer Academic, Dordrecht ; Boston [etc.], 1994.

[75] Michel Grabisch, Michio Sugeno, and Toshiaki Murofushi. *Fuzzy Measures and Integrals : Theory and Applications.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2000.

[76] Michel Grabisch, Hung T. Nguyen, and Elbert A. Walker. Fuzzy Measures and Integrals. In *Fundamentals of Uncertainty Calculi with Applications to Fuzzy Inference*, number 30 in Theory and Decision Library, pages 107–171. Springer Netherlands, January 1995.

[77] Natalia. Andrienko and Gennady. Andrienko. *Exploratory analysis of spatial and temporal data a systematic approach.* Springer, Berlin ; New York, 2006.

[78] Bernice E. Rogowitz, Lloyd A. Treinish, and Steve Bryson. How Not to Lie with Visualization. *Computers in Physics*, 10(3) :268–273, May 1996.

[79] Edward R. Tufte and Glenn M. Schmieg. The Visual Display of Quantitative Information. *American Journal of Physics*, 53(11) :1117–1118, November 1985.

[80] V. Pillet, J. Labarta, T. Cortes, and S. Girona. PARAVER : A Tool to Visualize and Analyze Parallel Code. Technical Report RR-95/03, CEPBA/UPC, 1995.

[81] Anthony Chan, William Gropp, and Ewing Lusk. An Efficient Format for Nearly Constant-time Access to Arbitrary Time Intervals in Large Trace Files. *Sci. Program.*, 16(2-3) :155–165, April 2008.

[82] Joel de Rosnay. *Le macroscope : vers une vision globale.* Editions du Seuil, Paris, 1975.

[83] Robin Lamarche-Perrin, Yves Demazeau, and Jean-Marc Vincent. Building Optimal Macroscopic Representations of Complex Multi-agent Systems. In Ngoc Thanh Nguyen, Ryszard Kowalczyk, Juan Manuel Corchado, and Javier Bajo, editors, *Transactions on Computational Collective Intelligence XV*, Lecture Notes in Computer Science, pages 1–27. Springer Berlin Heidelberg, January 2014.

[84] Robin Lamarche-Perrin, Yves Demazeau, and Jean-Marc Vincent. Macroscopic Observation of Large-scale Multi-agent Systems. In *Proceedings of the 2014 Brazilian Conference on Intelligent Systems (BRACIS'14)*, 2014.

[85] Robin Lamarche-Perrin, Yves Demazeau, and Jean-Marc Vincent. How to Build the Best Macroscopic Description of Your Multi-Agent System ? In Yves Demazeau, Toru Ishida, Juan M. Corchado, and Javier Bajo, editors, *Advances on Practical Applications of Agents and Multi-Agent Systems*, number 7879 in Lecture Notes in Computer Science, pages 157–169. Springer Berlin Heidelberg, January 2013.

[86] R. Lamarche-Perrin, Y. Demazeau, and J.-M. Vincent. The Best-Partitions Problem : How to Build Meaningful Aggregations. In *2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, volume 2, pages 399–404, November 2013.

[87] Robin Lamarche-Perrin, Yves Demazeau, and Jean-Marc Vincent. Analysis of International Relations through Spatial and Temporal Aggregation. In Yves Demazeau, Toru Ishida, Juan M. Corchado, and Javier Bajo, editors, *Advances on Practical Applications of Agents and Multi-Agent Systems*, number 7879 in Lecture Notes in Computer Science, pages 296–299. Springer Berlin Heidelberg, January 2013.

[88] Robin Lamarche-Perrin, Jean-Marc Vincent, and Yves Demazeau. Informational Measures of Aggregation for Complex Systems Analysis. Technical Report RR-LIG-026, LIG, July 2012.

[89] B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, April 2002.

[90] Claude Elwood Shannon. A Mathematical Theory of Communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1) :3–55, 2001.

[91] Timothée Giraud, Claude Grasland, Robin Lamarche-Perrin, Yves Demazeau, Jean-Marc Vincent, and Thomas Thévenin. Identification of international media events by spatial and temporal aggregation of RSS flows of newspapers. In *Proceedings of the 18th European Colloquium in Theoretical and Quantitative Geography (ECTQG)*, pages 112–114, 2013.

[92] Imre Csiszár. Axiomatic Characterizations of Information Measures. *Entropy*, 10(3) :261–273, September 2008.

[93] S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1) :79–86, March 1951.

[94] Mehdi Aghagolzadeh, Hamid Soltanian-Zadeh, and Babak Nadjar Araabi. Information Theoretic Hierarchical Clustering. *Entropy*, 13(2) :450–465, February 2011.

[95] Shun-ichi Amari. Information geometry in optimization, machine learning and statistical inference. *Frontiers of Electrical and Electronic Engineering in China*, 5(3) :241–260, September 2010.

[96] Alexis Martin, Generoso Pagano, Jérôme Correnoz, and Vania Marangozova-Martin. Analyse de systèmes embarqués par structuration de traces d'exécution. In *Actes de ComPAS 2014 : conférence en parallélisme, architecture et systèmes*, April 2014.

[97] Alexis Martin and Vania Marangozova-Martin. Analyse de traces d'exécutions pour les systèmes embarqués : détection d'anomalies par corrélation temporelle. report, Inria, October 2014.

[98] David H. Bailey, Eric Barszcz, John T. Barton, David S. Browning, Russell L. Carter, Leonardo Dagum, Rod A. Fatoohi, Paul O. Frederickson, Thomas A. Lasinski,

Rob S. Schreiber, and others. The NAS parallel benchmarks. *International Journal of High Performance Computing Applications*, 5(3) :63–73, 1991.

[99] Patricia López Cueva, Aurélie Bertaux, Alexandre Termier, Jean François Méhaut, and Miguel Santana. Debugging Embedded Multimedia Application Traces Through Periodic Pattern Mining. In *Proceedings of the Tenth ACM International Conference on Embedded Software*, EMSOFT '12, pages 13–22, New York, NY, USA, 2012. ACM.

[100] Omer Zaki, Ewing Lusk, and Deborah Swider. Toward Scalable Performance Visualization with Jumpshot. *High Performance Computing Applications*, 13 :277–288, 1999.

[101] Surajit Chaudhuri. An Overview of Query Optimization in Relational Systems. In *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, PODS '98, pages 34–43, New York, NY, USA, 1998. ACM.

[102] Markus Geimer, Felix Wolf, Brian J. N Wylie, Erika Ábrahám, Daniel Becker, and Bernd Mohr. The Scalasca performance toolset architecture. *Concurrency and Computation : Practice and Experience*, 22(6) :702–719, April 2010.

[103] Ken Perlin. An Image Synthesizer. In *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '85, pages 287–296, New York, NY, USA, 1985. ACM.

[104] D. Holten, R. Vliegen, and J.J. Van Wijk. Visual realism for the visualization of software metrics. In *Visualizing Software for Understanding and Analysis, 2005. VISSOFT 2005. 3rd IEEE International Workshop on*, pages 1–6, 2005.

[105] Luka Stanisic, Samuel Thibault, Arnaud Legrand, Brice Videau, and Jean-François Méhaut. Modeling and Simulation of a Dynamic Task-Based Runtime System for Heterogeneous Multi-core Architectures. In Fernando Silva, Inês Dutra, and Vítor Santos Costa, editors, *Euro-Par 2014 Parallel Processing*, number 8632 in Lecture Notes in Computer Science, pages 50–62. Springer International Publishing, August 2014.

[106] Dan Gusfield. *Algorithms on Strings, Trees and Sequences : Computer Science and Computational Biology*. Cambridge University Press, May 1997.

[107] David Beniamine. Cartographier la mémoire virtuelle d'une application de calcul scientifique. In *ComPAS'2013 / RenPar'21*, Grenoble, France, 2013.

Annexes

ANNEXE A

Cas d'études supplémentaires

Nous détaillons dans cette Annexe des cas d'études supplémentaires relatifs à l'analyse fondée sur la technique d'agrégation temporelle.

Unicast

Unicast, référencé dans la Table 5.1 par l'identifiant **UNI-[X]**, est une trace fournie par STMicroelectronics, générée au format KPTrace de la même manière que pour TSrecord (voir Section 5.7.1). Il s'agit d'un décodage vidéo, où le fichier est lu depuis le réseau. La vidéo diffusée est d'une mauvaise qualité, contenant des sauts d'images et des artefacts sonores. Ceci est dû à un problème de décodage, que l'on corrèle à des interruptions réseau trop longues [96, 97]. Deux traces sont générées avec des niveaux d'abstraction différents.

Comme pour TSrecord, notre analyse vise en premier lieu à retrouver les zones de perturbations, pour ensuite se focaliser sur le comportement détaillé des ressources grâce à une représentation de type diagramme espace-temps. L'analyse avec Ocelotl décèle des pics réguliers, dans la trace bas niveau UNI-B, montrée Figure A.1, matérialisés par une augmentation de la quantité de temps passé dans les interruptions logicielles, et la présence des fonctions `netif_receive_skb`, `dev_queue_xmit` et `sock_sendmsg`. Une étude de la trace haut-niveau UNI-H et du diagramme espace-temps corrèle le blocage de l'application et la présence de ces pics d'interruptions réseaux.

Décomposition LU

La décomposition LU (LU pour Lower-Upper symmetric Gauss-Seidel) résout un système synthétique d'équations différentielles partielles non linéaires grâce à une méthode de surrelaxation successive (SSOR). On a ici une alternance de phases de calculs et de communications, et on attend un comportement globalement homogène de l'application [98]. Ce cas d'étude est référencé dans la Table 5.2 par l'identifiant **LU.C-[X]**

Nous avons exécuté LU, classe C, 700 processus, sur Grenoble et Nancy. Comme pour CG, les performances de Grenoble sont meilleures (ratio de 1:2,6 entre Grenoble et Nancy) et s'expliquent par les mêmes facteurs (la technologie des interconnects mis en

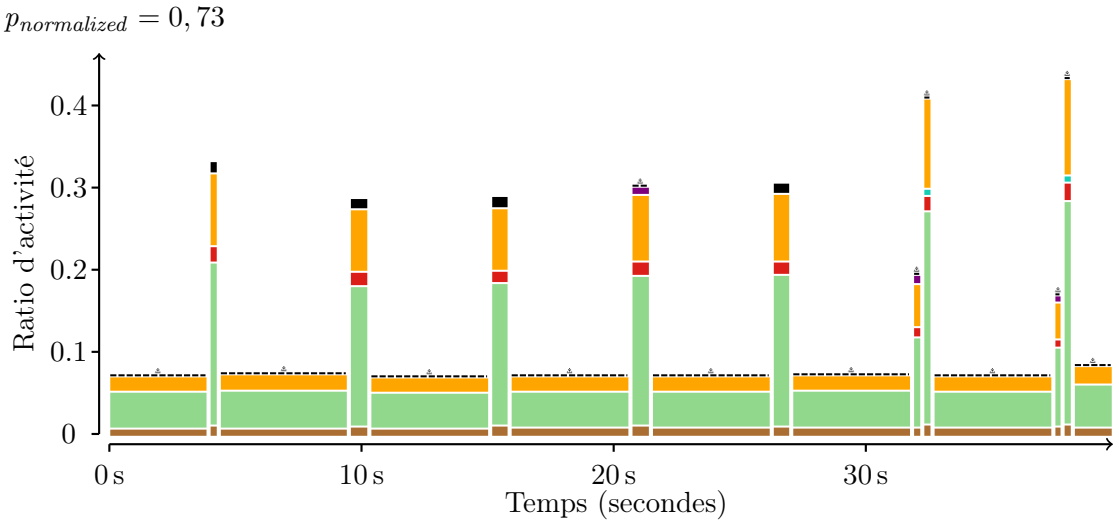


FIGURE A.1 – Histogramme empilé représentant une partition optimale et le ratio d'activité de UNI-B. Apparaissent les états suivants : `SoftIRQ` (vert), `netif_receive_skb` (orange), `Interrupt` (marron), `fe_ip_v4_nethook` (cyan), `dev_queue_xmit` (rouge) et `sock_sendmsg` (violet). On distingue la présence régulière de pics correspondant à une hausse des interruptions réseau.

jeu et la topologie de la plateforme). Le ratio d'activité de `MPI_Recv` est significativement plus long dans le cas de Nancy (600, contre 200 à Grenoble). De plus, notre technique met en évidence une perturbation temporelle d'une durée anormalement longue lors de la phase de calcul, vers 38 secondes d'exécution, dans le cas de Nancy, Figure A.2. Nous montrons, en comparaison, la phase de calcul de Grenoble, beaucoup plus homogène, Figure A.3.

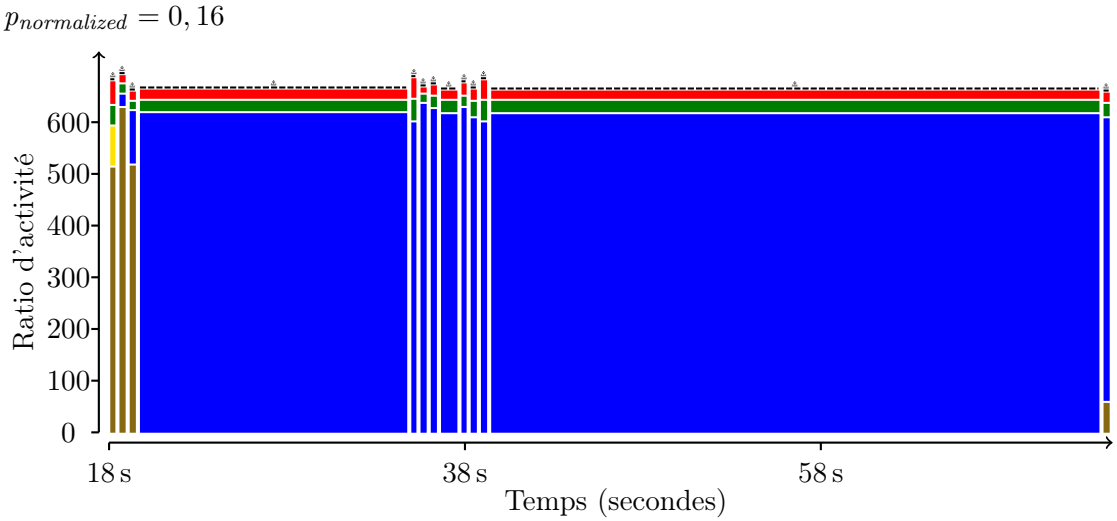


FIGURE A.2 – Histogramme empilé représentant une partition optimale et le ratio d'activité de la phase de calcul de LU.C-Nan-700. On distingue une perturbation autour de 38 secondes.

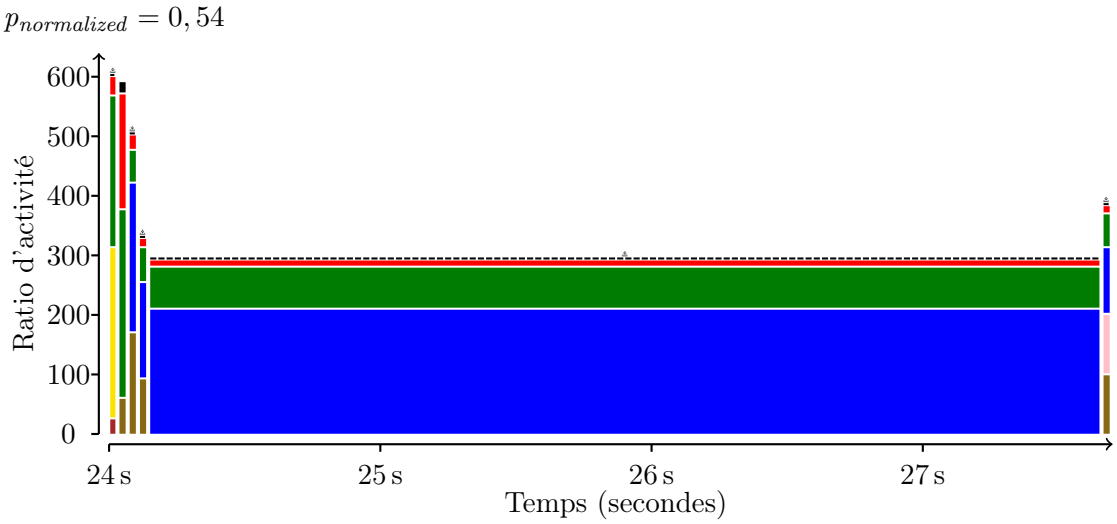


FIGURE A.3 – Histogramme empilé représentant une partition optimale et le ratio d'activité de la phase de calcul de LU.C-Gre-700. Le comportement de la phase de calcul est homogène.

ANNEXE B

Optimisation des requêtes

Nous expliquons ici plus en détail comment réduire la complexité des requêtes à la base de données lors de la lecture de la trace, décrites en Section 7.4.1, et destinée à obtenir un ensemble d'évènements filtrés selon la dimension temporelle, spatiale, et de types d'évènements. Pour rappel, la requête totale se décompose en trois parties, relatives respectivement au temps (*condition* T), à l'espace (*condition* S), et aux types d'évènements (*condition* J), tel qu'il suit :

$$condition\ Total = condition\ T \wedge condition\ S \wedge condition\ J$$

Ces simplifications peuvent être cumulées.

Simplification de la condition temporelle La partie de la requête *condition* T liée à l'intervalle de temps se présente sous la forme de la condition *condition* T_e pour les évènements ponctuels :

$$condition\ A = \tau_d(e) \geq \tau_{min} \tag{B.1}$$

$$condition\ B = \tau_d(e) \leq \tau_{max} \tag{B.2}$$

$$condition\ T_e = condition\ A \wedge condition\ B \tag{B.3}$$

ou de la condition *condition* T_s pour les états et les variables :

$$condition\ C = \tau_f(e) > \tau_{min} \tag{B.4}$$

$$condition\ T_s = condition\ T_e \vee \overline{condition\ A} \wedge condition\ C \tag{B.5}$$

avec $\tau_d(e)$ l'estampille de l'évènement e évalué (ou de début de l'état évalué), $\tau_f(e)$ celle de fin de l'état e évalué, et $[\tau_{min}, \tau_{max}]$ l'intervalle de temps requêté. Ces requêtes peuvent être simplifiées comme il suit :

- Si $\tau_{min} = \tau_{debut}$, avec τ_{debut} l'estampille la plus petite de la trace, alors *condition* A = 1. Cela donne : *condition* T_e = *condition* B et *condition* T_s = *condition* B ;
- Si $\tau_{max} = \tau_{fin}$, avec τ_{fin} l'estampille la plus grande de la trace, alors *condition* B = 1. Cela donne : *condition* T_e = *condition* A et *condition* T_s = *condition* $A \vee \overline{condition\ A} \wedge condition\ C$;

— Si $\tau_{min} = \tau_{debut}$ et $\tau_{max} = \tau_{fin}$, alors *condition* $T_e = 1$ et *condition* $T_s = 1$. Dans ce cas, il n'est pas nécessaire d'exprimer de condition sur le temps : *condition* $Total =$ *condition* $S \wedge$ *condition* J

Simplification de la condition spatiale La condition spatiale est exprimée comme il suit : *condition* $S = S(e) \in S_r$ où $S_r \in S$ est l'ensemble des éléments spatiaux désignés par la requête, et $S(e)$ la ressource associée à l'évènement évalué e. On peut effectuer la simplification suivante : si $S_r = \mathcal{H}(S)$, alors *condition* $S = 1$ et *condition* $Total =$ *condition* $T \wedge$ *condition* J.

Simplification de la condition sur les types d'évènements Cette condition est exprimée comme il suit : *condition* $J = J(e) \in \mathcal{J}(E)_r$ où $\mathcal{J}(E)_r \in \mathcal{J}(E)$ est l'ensemble des types d'évènements désignés par la requête, et $J(e)$ le type associé à l'évènement évalué e. La première simplification possible, comparable à celle apportée à la condition spatiale, est la suivante : si $\mathcal{J}(E)_r = \mathcal{J}(E)$, alors *condition* $J = 1$ et *condition* $Total =$ *condition* $T \wedge$ *condition* S. Dans le cas où l'ensemble des types d'évènements requêté forme une catégorie, c'est-à-dire que $\forall J_l \in \mathcal{J}(E)_r, J_l \in C_m$, où $C_m \in \mathcal{C}(\mathcal{J}(E))$ est une catégorie d'évènements (états, variables) associés à la métrique du modèle microscopique et $\mathcal{C}(\mathcal{J}(E))$ l'ensemble de toutes ces catégories d'évènements, alors on préférera employer la requête suivante : *condition* $J = C(e) == C_m$ où $C(e)$ est la catégorie associée à l'évènement évalué e. Cette requête évalue une égalité, de complexité $\mathcal{O}(1)$, plus efficace que la requête initiale, qui évalue l'appartenance à un ensemble, de complexité $\mathcal{O}\,|\mathcal{J}(E)_r|$. Ici, on peut tirer parti du modèle de données Framesoc qui associe explicitement les catégories définies par le format Pajé, et détaillées Section 2.1, aux évènements. Il est à noter que cette simplification est de priorité inférieure à la précédente (si $\mathcal{J}(E)_r = \mathcal{J}(E)$, alors *condition* $J = 1$).

ANNEXE C

Publications

Liste des publications relatives aux travaux décrits dans cette thèse.

A Spatiotemporal Data Aggregation Technique for Performance Analysis of Large-scale Execution Traces [10] *IEEE Cluster 2014*, Madrid, Espagne.

Cet article décrit la technique d'agrégation spatiotemporelle et la technique de visualisation matricielle associée. Nous présentons plusieurs cas d'études correspondant à l'exécution d'applications et noyaux du NASPB sur la grille de calcul Grid'5000.

Efficient Analysis Methodology for Huge Application Traces [11] *2014 International Conference on High Performance Computing & Simulation (HPCS)*, Bologne, Italie

Cet article présente Framesoc, l'infrastructure d'analyse de traces, et Ocelotl, l'outil d'analyse conçu durant cette thèse, et leurs capacités à permettre un flot d'analyse cohérent suivant le mantra de Shneiderman. Il décrit également des solutions aux problématiques liées à la multiplicité des formats de trace en proposant un modèle de données générique et à l'interaction et la réutilisation de résultats d'analyse entre différents outils.

Trace Management and Analysis for Embedded Systems [12] *2013 IEEE 7th International Symposium on Embedded Multicore Socs (MCSoC)*, Tokyo, Japon

Ce papier court de 4 pages décrit plusieurs contributions résolvant les problèmes de gestion de traces, de la multiplicité de leurs formats, des limites de la coopération entre les outils d'analyse à cause de l'incompatibilité entre leurs entrées et leurs sorties respectives, et de passage à l'échelle des outils de visualisation. Ces solutions sont implémentées au sein de Framesoc et d'Ocelotl.

Ocelotl: Large Trace Overviews Based on Multidimensional Data Aggregation [13] *8th International Parallel Tools Workshop*, Stuttgart, Allemagne

Ce chapitre de livre traite d'Ocelotl et de l'implémentation des visualisations fondées sur l'agrégation temporelle et spatiotemporelle, mais aussi de techniques assurant les performances et la réactivité du logiciel pour des traces volumineuses.

La visualisation de traces, support à l'analyse, déverminage et optimisation d'applications de calcul haute performance [14] *Atelier Visualisation d'informations, interaction et fouille de données (VIF) de la 13e Conférence Francophone sur l'Extraction et la Gestion des Connaissances (EGC'2013)*, Toulouse, France

Ce papier présente un état de l'art des techniques de visualisation de traces employées dans le domaine de l'analyse d'applications parallèles ou embarquées, et les solutions qu'elles adoptent pour assurer le passage à l'échelle.

Agrégation temporelle pour l'analyse de traces volumineuses [15] *10ème Atelier en Evaluation de Performances*, Sofia Antipolis, France

Il s'agit d'un court papier de deux pages résumant les travaux relatifs à la technique d'agrégation temporelle évoqués dans cette thèse.

A Trace Macroscopic Description based on Time Aggregation [16] *Hal RR-8524*, 2014

Ce rapport de recherche présente en détail la technique d'agrégation temporelle et son implémentation dans Ocelotl. Plusieurs cas d'études sont présentés, notamment l'application GStreamer décrite dans cette thèse et des applications et noyaux NASPB exécutés sur Grid'5000.

Trace Management and Analysis for Embedded Systems [17] *Hal RR-8304*, 2013

Ce rapport de recherche est la version longue de l'article soumis à MCSoC 2013 et accepté en tant que papier court.

ANNEXE D

Logiciels

Liste des principaux logiciels développés dans le cadre de cette thèse.

Ocelotl Le logiciel d'analyse décrit dans cette thèse implémentant les techniques d'agrégation temporelles et spatiotemporelles.
▷ https://github.com/soctrace-inria/ocelotl

Bibliothèque lpaggreg La bibliothèque C++ où sont implémentés les algorithmes d'agrégation d'un ensemble ordonné (temporel), d'un ensemble hiérarchique (spatial) et d'un ensemble ordonné et hiérarchique (spatiotemporel).
▷ https://github.com/dosimont/lpaggreg

Bibliothèque lpaggregjni L'interface JNI permettant d'appeler la bibliothèque LPAggreg à partir d'un code Java.
▷ https://github.com/dosimont/lpaggregjni

Importateurs Framesoc Pajé et Paraver Ces deux modules permettent d'importer dans Framesoc des traces au format Pajé et au format Paraver.
▷ https://github.com/soctrace-inria/framesoc.importers

Plug-in Framesoc de filtrage Ce plug-in permet de filtrer une trace selon différents critères et de générer un résultat d'analyse Framesoc attaché à cette trace.
▷ https://github.com/soctrace-inria/framesoc.various

Convertisseur d'une trace GStreamer au format Pajé Le procédé requiert l'exécution d'un script nettoyant puis transformant la trace GStreamer dans un fichier CSV grâce à un script développé par Serge Emteu, doctorant au LIG et chez STMicroelectronics.
▷ https://github.com/dosimont/gst2paje/blob/master/gstparser.py

Le fichier CSV généré est ensuite converti au format Pajé à l'aide d'un second script.
▷ https://github.com/dosimont/gst2paje/blob/master/gst2paje.py

Spatiotemporal Aggregation for Execution Trace Visualization

CHAPTER 1

INTRODUCTION

Tracing is a method that consists in collecting and saving information about the resources and the events that occur during the execution of a computer program. The objective of this process is to understand, debug, and optimize applications *post-mortem* using analysis techniques. Visualization techniques are traditionally used by application designers to perform such analysis. They generally use techniques based on the representation of the application behavior over the time and the structure of resources involved in its execution, such as the space-time diagram, inspired by the Gantt chart. This thesis tackles the scalability issues of this category of representations.

Shneiderman's *mantra* [1] helps to design an analysis method based on visualization:

Overview first, zoom and filter, then details-on-demand

Actually, this guideline prescribes a top-down approach for the analysis, which consists in starting by an overview, and progressively focusing on details by using interactions like zooming and filtering.

Classic spatiotemporal approaches usually represent all the information contained in traces. Consequently, in the case of huge traces containing million of events, the tools are incapable of generating correct overviews, which prevents the analysts from starting their analysis in a proper manner. More specifically, we distinguish three factors that explain why these techniques fail to provide a satisfying entry-point to the analysis:

- *Screen limitations* [2]. As the screen resolution is limited by the number of pixels, the amount of graphical objects that can be represented is finite. When this problem is ignored by the visualization technique, this leads to cluttered representations, as shown in Figure Ext. 1.1, which contains visualization artifacts, incorrect proportions, and incoherence between the views after resizing or moving the visualization window;

- *Performance* [3]. The necessity of interacting with the representation requires performance constraints. These would not exist if the view was generated statically. We distinguish issues related to the memory management, which can prevent the analysis program from returning a result, and issues related to the computation duration, which can be too long to ensure a good interactivity;

- *Perceptive and cognitive capabilities of the user* [4, 5, 6]. Even if a gigantic screen was available and if we could gloss over material and computing limitations, the analysis would remain difficult. Indeed, the analyst, as a human, is a limiting factor himself. He is not capable of dealing with a quantity of data that is too large: his perception of the view is dependent of his sight, while his cognitive capabilities limit his comprehension of the scene.

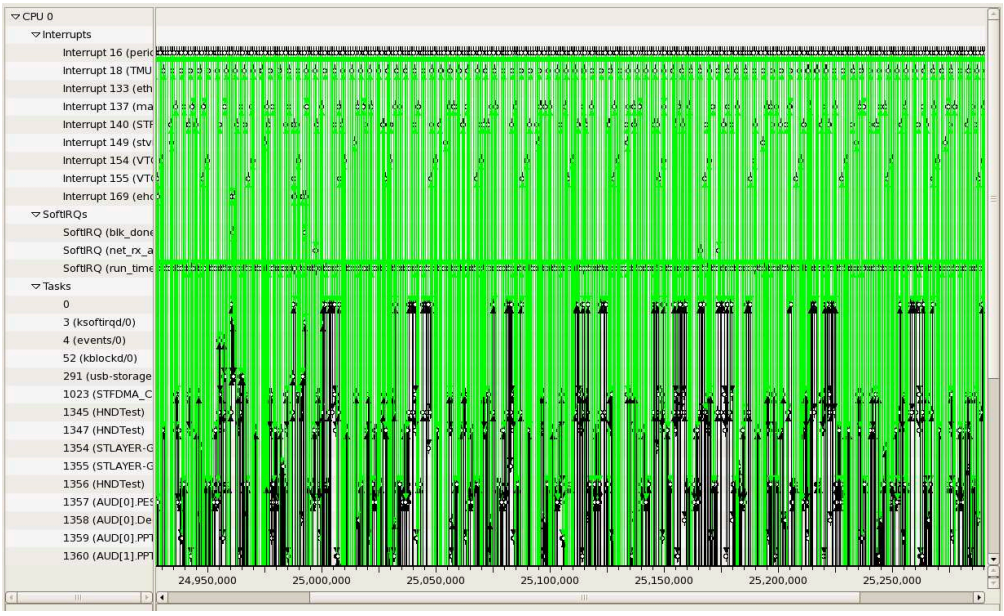


Figure Ext.1.1: Example of a classic Gantt chart trying to represent an important quantity of events, without complexity reduction techniques over the time axis. The representation is cluttered and unreadable.

The objective of this thesis is the elaboration of a visualization-based trace analysis method, which is scalable and follows Shneiderman's *mantra*. It is thus necessary to propose a solution that satisfies each of its steps. To provide an *overview* that takes into account the screen limitations, the interactivity constraint, and the user capabilities, we design two visual analysis methods. The first one, which is temporal, is initially destined to real-time embedded applications whose behavior evolution observation is essential. The second one, which is spatiotemporal, is mainly designed for structured parallel systems. They form a significant extension of an existing aggregation method, resulting from Lamarche-Perrin's doctorate thesis works [8] on the analysis of multi-agent systems. These works enable the complexity reduction of a system upstream of its representation, while giving the analyst the choice of the level of details and the

management of the information loss. One of the main interests of this technique is that the complexity reduction is not only performed to enable the analysis scalability : it brings a knowledge on the behavior of the system by discriminating the homogeneity and the heterogeneity of the entities that compose it.

However, this method cannot be applied directly on the measures contained in raw traces, since the dimensions of the system to aggregate must be discrete. It is thus required to abstract the trace first, in the form of an intermediate representation, the *microscopic model*, based on a metric representing the phenomenons the user wants to discriminate. This microscopic model is then aggregated. To perform this aggregation in the case of the temporal analysis, we adapt and extend an existing aggregation algorithm previously designed by Lamarche-Perrin [8,9]. In the case of the spatiotemporal analysis, we design an algorithm that aggregates simultaneously both dimensions. For each of these methods, we associate visualizations capable of representing the aggregated trace. As the complexity reduction is potentially not homogeneous over the whole representation, we ensure the complete scalability thanks to complementary visual aggregation techniques, applied on the graphical elements that are too small to be correctly represented by themselves. The full process enables to address the issues related to the limits of the screen, the performance, and the user.

We design a software, Ocelotl, which implements both aggregation methods. It addresses the performance issues, but also fully integrates Shneiderman's guideline. Ocelotl proposes an *overview* of the trace, but also advanced interaction mechanisms that enable to *zoom*, *filter* and access to more *details*. In particular, this last step can be realized by switching to more detailed representations, like Gantt charts, on a subpart of the trace.

Use cases provided by embedded multimedia and parallel computing applications validate the relevancy of our analysis methods and their implementation. In particular, Ocelotl can highlight anomalies that are difficult to detect with classic spatiotemporal visualization techniques. Exhaustive tests confirm the scalability capabilities of our software infrastructure. It is able to handle in a reasonable time traces up to two billion events and whose size reaches 68 gigabytes.

These works has been published in international conferences **[10,11,12]**, in an international workshop **[13]**, in national workshops **[14,15]**, and in research reports **[16,17]**. In this extended abstract, these references will be printed in **bold** fonts to relate them to the contributions we describe.

The plan of this thesis is organized as follows. **Chapter 2** summarizes the context of trace analysis by describing the tracing process, the trace contents, and usual visualization techniques used to analyze them. **Chapter 3** focuses on the aggregation and its use in the visualization techniques to ensure their scalability. **Chapter 4** describes a particular aggregation method, provided by Lamarche-Perrin's works, that we significantly extend in this thesis. **Chapter 5** deals with the temporal analysis method we have designed, while **Chapter 6** describes the spatiotemporal one. **Chapter 7** relates to the implementation of these methods in Ocelotl analysis tool. Finally, **Chapter 8** concludes this thesis through a summary of our contributions and their overall evaluation. We also present our perspectives and research axes that can be envisaged as a follow up to these works.

CHAPTER 2

Visualization of Execution Traces

Analyzing software application to understand their behavior, debug and optimize them, is a difficult task, since they are executed on systems that are of increasing complexity. In the case of embedded systems and personal computers, the main difficulties are due to the complex software stack [20]. Indeed, the analysis involves low-level aspects, close to the hardware, middleware layers that abstract it [21] and applications running on top of them. Regarding parallel systems, they might be distributed around the world, through a complex hardware and software hierarchy that is often heterogeneous [22]. The scale of these systems is a constraint for the analysis tools. Software libraries like OpenMP and MPI [23] are used to parallelize applications. However, they are characterized by a strong indeterminism, inherent to the parallel programming paradigm (concurrent access to resources, message ordering, deadlocks), which cannot be handled by traditional analysis methods [24].

Tracing consists in placing observers, collecting and saving data about the application execution, in one or several files. These files contain information about the state of the execution, and measures about the state of the platform, provided by hardware or software counters. These traces are finally analyzed *post-mortem*. Some techniques, such as data mining and pattern recognition, focus on finding anomalies directly in raw traces without the help of the analyst. In this thesis, we are interested in methods based on trace visualization, where the application behavior, which is materialized by the trace data, is represented thanks to graphical objects. In particular, we have concerns about temporal and architectural aspects; our objective is to relate the dynamic of the temporal behavior to the structure and the topology of the application.

In **Chapter 2** of this thesis, we present the context of our works, first through the description of the traces, their contents, some examples of formats [26, 29, 30, 31, 32, 33, 34, 36], and examples of instrumentation infrastructures [25, 26, 27, 28]. In particular, we introduce three concepts we use all along this thesis: *the resource space*, which are the elements composing its structure, *the temporal dimension*, and *the events*.

Then, we propose a summary of existing trace visualization techniques that illustrate the problematic of this thesis. A first set of techniques computes a synthesis of the trace using statistical operators. For instance, the area chart shown in Figure Ext.2.1 enables to compare several executions, as a function of a particular parameter. A second set

gathers detailed visualizations that focus on the evolution of the application behavior over time or its structure. The most wide-spread, the space-time diagram, also named Gantt chart [43], is present in many tools [7, 30, 44, 48, 80]. A sample is shown in Figure Ext.2.2.

At last, we discuss about the limitations of both approaches: while synthetic views based on statistics are scalable but loses much information on temporal and structural dimensions, spatiotemporal visualization techniques suffer of screen limitations, performance issues and user's cognitive and perceptive limits.

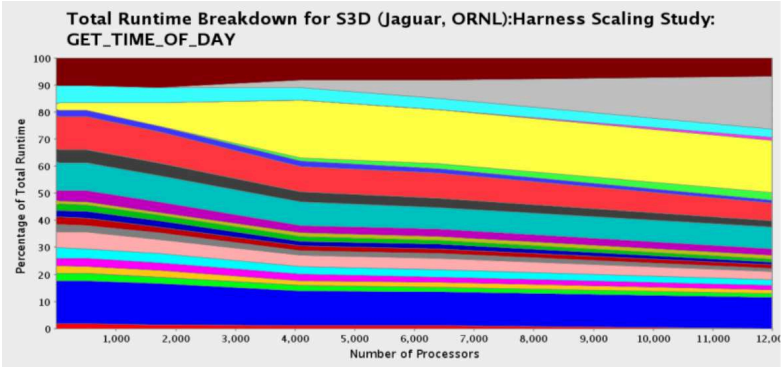


Figure Ext.2.1: Example of statistic representation provided by PerfExplorer: an area chart showing the percentage of time passed in several functions as a function of the number of processors.
Source: http://www.cs.uoregon.edu/research/tau/vihps14/PerfExplorer.pdf

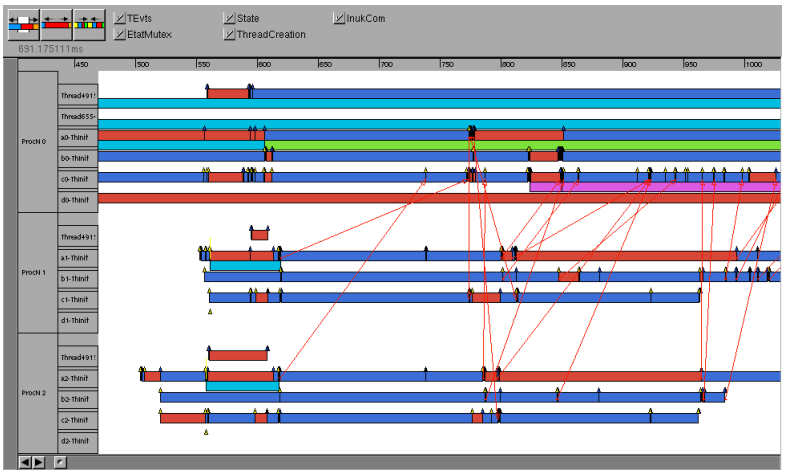


Figure Ext.2.2: Example of Gantt chart provided by Pajé and showing the execution of three processes through their states and their communications.

CHAPTER 3

Aggregation and Scalability

Information representation of large data sets is hindered by the limitations imposed by the screen size, the performance and the analyst's capabilities. Aggregation methods address these issues by providing a more compact representation while trying to maintain the relevant information. These methods are present in various domains, such as image processing, decision-making, machine learning and information visualization, which includes trace visualization. Their common objective is to provide a global interpretation from data potentially generated by several sources of information. In **Chapter 3**, we are interested in the aggregation techniques used by trace visualization tools.

We start by defining the aggregation and its properties from a mathematical point of view. First, we remind a formalism (Equation 3.1, page 26) established by Detyniecki [53], applied to real numbers, but that can be extended to other mathematical objects and fields. Then, we summarize some properties [57, 58, 59] that ensure the relevance and the good behavior of an aggregation operator. Mathematical properties are algebraic and topological and help to show that an operator has a non chaotic behavior, i.e., a relative impedance to slight variations of the values in input. Behavioral properties express how to interpret the aggregation operation result and help the decision making [59]. At last, we review examples of usual aggregation operators, divided in four classes: *conjunctive*, *disjunctive*, *trade-off* and *hybrid* operators.

We then introduce a more general formalism, describing the principle of complexity reduction based on aggregation. This formalism is applied on two types of techniques present in the information visualization domain: data and visual aggregations. It is composed of several points: the *dimensions* in which the aggregation is done, the *operands*, i.e., the objects taken into account by the complexity reduction process, the *constraints* defining the set of possible aggregates, the *conditions* triggering the aggregation, and the aggregation *operation* itself.

In order to evaluate the quality of an aggregated representation, we introduce criteria previously published by Elmqvist & Fekete [4], namely the *entity budget*, the *visual summary*, the *visual simplicity*, the *discriminability*, the *fidelity*, and the *interpretability*. We use these criteria to compare several aggregation techniques implemented in trace analysis tools. Table 3.1 (page 38) summarizes this effort **[10]**. Aggregation techniques that are presented are divided into three main groups: those based on visual aggregation

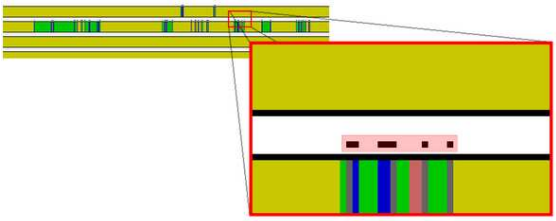


Figure Ext.3.1: Demonstration of the aggregation principle of LTTng Eclipse Viewer [44]. After a zoom out, if the states become too small to be visible, then they are aggregated and highlighted using dots.

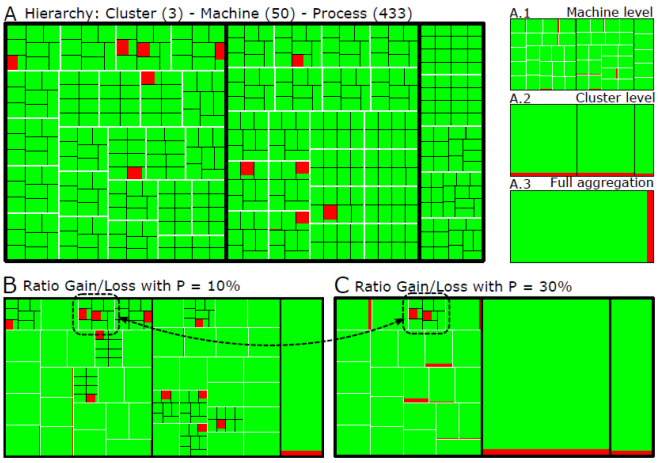


Figure Ext.3.2: Example of treemaps proposed by Viva [6] (ex-Triva [50]). Representations A are an example of classical hierarchical aggregation, while representations B and C use multiresolution aggregation, based on *Lamarche-Perrin's method*: the user adjusts the level of details by setting a parameter p, which does a trade-off between the complexity reduction and the information loss.

(an example is shown in Figure Ext.3.1), data aggregation (Figure Ext.3.2) and the hybrid ones, which combine both techniques. The comparison enables to highlight their strengths and their weaknesses: our objective is to determine the approach that fit the best to our needs.

At last, we conclude this chapter by a discussion. We determine the influence of multidimensionality on the respect of Elmqvist & Fekete's criteria, and in particular, we formalize a new criterion that evaluates the aggregation process coherence over several dimensions. This coherence is necessary to correctly manage the number of graphical objects over the whole representation. We also demonstrate that data-aggregation-based techniques fulfill Elmqvist & Fekete's criteria better than visual-aggregation-based ones. Finally, we show the potential of a partition and aggregation method, provided by Lamarche-Perrin [8], able to reduce the complexity while controlling the information loss, and which seems well-fitted for our problem. An application of this technique to trace analysis is shown in Figure Ext.3.2 B and C.

CHAPTER 4

Macroscopic Analysis of Complex Systems

Lamarche-Perrin [8] opposes two approaches whose aim is the analysis of a large scale and complex multi-agent system. The *analytic approach* considers separately each component of a system, whereas the *systemic approach* apprehends it globally [82]. According to Lamarche-Perrin, the analytic approach does not enable the analysis of very large systems. Indeed, it requires observation tools whose size is close to the system size. This generates a large quantity of information, and thus a large number of microscopic phenomenons to explain, which is too much for a human analyst. On the contrary, the systemic approach is more suited to the analysis of complex systems, since it proposes a macroscopic point of view that helps to distinguish what is important from what is not.

In their works, Lamarche-Perrin *et al.* [8, 9, 83, 84, 85, 86, 87, 88] designed a complexity reduction method following the systemic approach. This method is based on an abstraction process involving aggregation techniques. The analysis context and the observer's knowledge about the system help to elaborate this abstraction, leading to a macroscopic representation that has a meaning according to the analyst.

In this thesis, we propose to reuse and significantly extend this method to apply it to trace analysis. We consider that this approach is suited to our problem, since it is not solely focused on the complexity reduction. Indeed, the aggregation process brings knowledge about the system by discriminating the homogeneity and the heterogeneity in the behavior of the entities that compose it. The aggregation is chosen according to a trade-off between the complexity reduction and the information loss. As a consequence, the most homogeneous elements are aggregated in priority. The measures of complexity reduction and information loss are both coming from the information theory.

This process can help to express the temporal homogeneity in the behavior of embedded systems, in order to detect disruptions related to frame losses or sound processing artifacts. In the case of parallel system, we are interested in discriminating temporal phases, but also local temporal perturbations, potentially caused by internal or external factors. Regarding the structural dimension, we want to analyze aspects like the load-balancing, and the task or function execution distribution on the resources, which give relevant information about the proper dimensioning of the platform or help to detect poor performance on a sub-part of the system.

Chapter 4 focuses on the description of what we name *Lamarche-Perrin's method*.

It does not bring contributions, but is necessary to explain the aggregation process, and the concepts and notations we use in Chapters 5 and 6. We introduce the concept of microscopic model, which constitutes the finest level of detail for the analysis. This microscopic model is a *multidimensional set* composed of *individuals*, each one associated with an *attribute*, which is its value. We then describe the partition and aggregation process, which reduces the system complexity. We present properties about the set of partitions, such as the relations of refinement and coverage.

We associate quality measures to the partitions. We distinguish the complexity reduction, expressed using the Shannon entropy and the information loss, provided by the Kullback-Leibler divergence. These quality measures are used to solve the *best partition problem*: depending on the nature of the measure taken into account (positive, like the complexity reduction, or negative, like the information loss), the *best partition* is the one that maximizes or minimizes this quality measure. In our applications, this problem is solved using a trade-off measure, the pIC (*parametrized Information Criterion*), which can be tuned using a parameter, which we name p, provided by the analyst. Quality measures are *monotone* and *additive*. These properties, combined with the refinement and coverage relations, enable to compute them as the sum of the quality measures of the parts that compose them.

In order to reduce the complexity of the best partition problem, which is exponential, but also to bring meaning to the aggregation process, we reduce the set of possible partitions by expressing explicitly which ones are allowed or forbidden. These constraints are built by taking into account syntactic and semantic characteristics of the system. We give examples of applications previously designed by Lamarche-Perrin *et al.*, such as Triva [5, 6], which proposes a hierarchical aggregation whose constraints are defined by the tree branches, and the temporal analysis of citations in journals [8, 9], based on a temporal aggregation and where aggregated parts must be contiguous.

Last but not least, we evoke the design of the hierarchical and the temporal aggregation algorithms used in the examples above. Lamarche-Perrin *et al.* use a dynamic programming approach, and take advantage of the refinement and coverage relations, and of the additivity of the quality measures, to decompose the problem into sub problems. Both algorithms are described in Lamarche-Perrin's thesis [8].

CHAPTER 5

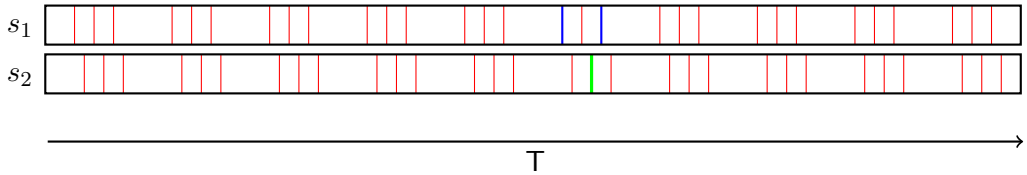
Temporal Aggregation of Execution Traces

The traces described in Chapter 2 are a particular case of multi-agent systems. In both cases of embedded multimedia and parallel applications, we are interested in the analysis of the temporal behavior. We estimate that the method proposed by Lamarche-Perrin is a framework that can help their analysis by addressing the scalability issue.

In **Chapter 5**, we propose to adapt this method to analysis of traces, and in particular, to extend an existing temporal aggregation algorithm to make it suited to our needs. As the raw trace, which we name the *nanoscopic model*, contains a continuous temporal dimension, it is necessary to discretize it, in order to fit with *Lamarche-Perrin's method*. We detail how to build a microscopic model as required by the method, by discretizing the time in slices. The structure representing the microscopic model is given by the Equation 5.4 (page 53): it is an application of the time, the resource and the event type dimensions, in $\mathbb{R}^+$. Each value of this application is generated using an aggregation technique, and describe the behavior we wish to analyze. For instance, we propose the following metrics: the event occurrence number, the total duration of each type of states for each time slice, their average duration, or the mean of a variable value. Figure Ext.5.1, for instance, depicts the process leading to a microscopic model representing the event occurrence number, we get from a raw trace. We also discuss about the effects of the granularity and the variation of the time slice number on the stability of the microscopic models. The order of magnitude of the time slice number allowed by the algorithm complexity and the screen size limitations is about a hundred. If we take this into account, slightly change the time slice number will not have much effect on the analysis result for realistic use cases containing millions of events.

Since the original algorithm works with scalar values, we need to adapt it to aggregate the microscopic model get from the trace, which contains several dimensions. First, we formalize the aggregation process using the five points we define in Chapter 3 (the *dimensions*: the time – the *operands*: the time slices – the *constraints*: only contiguous time slices can be aggregated, without overlapping between the aggregates – the *conditions*: solve the best partition problem using as quality measure the pIC – the *operator*: the sum) **[13]**. Then, we describe the algorithm and its data structures **[11, 12, 16, 17]**. Algorithm 5.2 (page 62), the *quality computation*, computes the quality measures (complexity reduction and information loss) associated with each possible temporal aggregate.

Nanoscopic model (raw trace):



Microscopic model:

(s_1, J_R)	3	3	3	3	3	**1**	3	3	3	3
(s_1, J_G)	0	0	0	0	0	0	0	0	0	0
(s_1, J_B)	0	0	0	0	0	**2**	0	0	0	0
(s_2, J_R)	3	3	3	3	3	**2**	3	3	3	3
(s_2, J_G)	0	0	0	0	0	**1**	0	0	0	0
(s_2, J_B)	0	0	0	0	0	0	0	0	0	0

t_1 t_6 t_{10}

T

Figure Ext.5.1: Example of microscopic model generation, from the raw trace. It contains three dimensions : the time (T), the resource space (S), composed of s_1 and s_2, and the event types ($\mathcal{J}(E)$), composed of J_R, J_G, J_B. The metric used here is the number of event occurrences.

Algorithm 5.3 (page 63), the *best cut computation*, returns the partition solving the best partition problem, represented as a sequence of cuts. The user provides, as input to this algorithm, the parameter p, which sets the relative importance of the complexity reduction and the information loss. An example of aggregations of a bidimensional microscopic model is given in Figure Ext.5.2.

Lamarche-Perrin's method does not specify how to represent the aggregated system. We propose several visualization techniques to provide the analyst with an understandable representation of the partition. For monodimensional microscopic models, the view consists in a simple bar chart, since it enables to represent the amplitude of the values associated with the aggregated parts over a time axis. However, this kind of techniques is unsuitable to multidimensional microscopic models. To address this issue, our first approach is a representation presenting the temporal partition through colored rectangles that are associated with the aggregates **[11, 12, 16, 17]**. Although this visualization technique enables to detect heterogeneity, it lacks information about the aggregate contents. This motivates our second approach, stacked bar charts: using a spatial dimension aggregation through the sum operator, we represent the total value associated with each event type, for each time aggregate **[13, 15, 16]**. We also propose a representation showing the mode, i.e., for each aggregate, we show only the event type whose associated value has the highest value. In the case of the stacked bar chart, we also design a visual aggregation technique gathering the stacks that are too small to be represented correctly without graphical artifacts **[13, 16]**.

We dedicate a section to validate our analysis method through several use cases. We

Microscopic model:

s_1	23	25	22	30	**55**	21	26	35	19	30
s_2	75	72	71	69	**92**	73	75	**35**	70	71

t_1 t_6 t_{10}

T

Aggregation: $p = 0.00952148$

s_1	100	**55**	47	35	49
s_2	287	**92**	148	**35**	141

Aggregation: $p = 0.0181885$

s_1	202	35	49
s_2	527	**35**	141

Figure Ext.5.2: Example of temporal aggregations applied on a bidimensional microscopic model, with several parameters p.

distinguish multimedia and parallel applications, respectively summarized by Table 5.1 (page 70) and Table 5.2 (page 84). The first category highlights typical problems encountered by the industrial partners involved in the SoC-Trace project[1]. We start our demonstration by a pedagogic use case, obtained by voluntarily perturbing a GStreamer video execution with an I/O stress. This allows us to detail our approach and the method we use to identify aggregations that are relevant, thanks to complexity reduction and information loss curves. These are generated using an algorithm that gives a list of best partitions, their associated value p, and their quality measures. We successfully show the perturbation with our implementation (Figure Ext.5.3), even in the case where it is not visible with a classic space-time representation. Our interaction mechanisms enable to focus on it, zoom in and finally, switch to a space-time diagram on this temporal subpart of the trace, to get more details on the resources that are involved **[11, 12, 15, 16, 17]**. Other use cases are traces coming from the execution of streaming applications developed by STMicroelectronics. Frame loss and sound artifacts occur regularly during the lecture of the video. Thanks to our implementation, we highlight regular perturbations related to these phenomenons. Then, the interaction features enable us to get to the root of this problem by discriminating the resources that have a abnormal behavior.

Regarding the parallel applications, we trace the execution of MPI applications and kernels of the NAS Parallel Benchmark [98]. We run Conjugate Gradient and Lower-Upper Gauss-Seidel solver on Grid'5000[2], using several hardware resource configurations and making the process count vary, and we compare the executions with our analysis

[1] http://www.minalogic.com/TPL_CODE/TPL_PROJET/PAR_TPL_IDENTIFIANT/2717/15-annuaire-innovations-technologiques-nanotechnologie-systeme-embarque.htm

[2] https://www.grid5000.fr

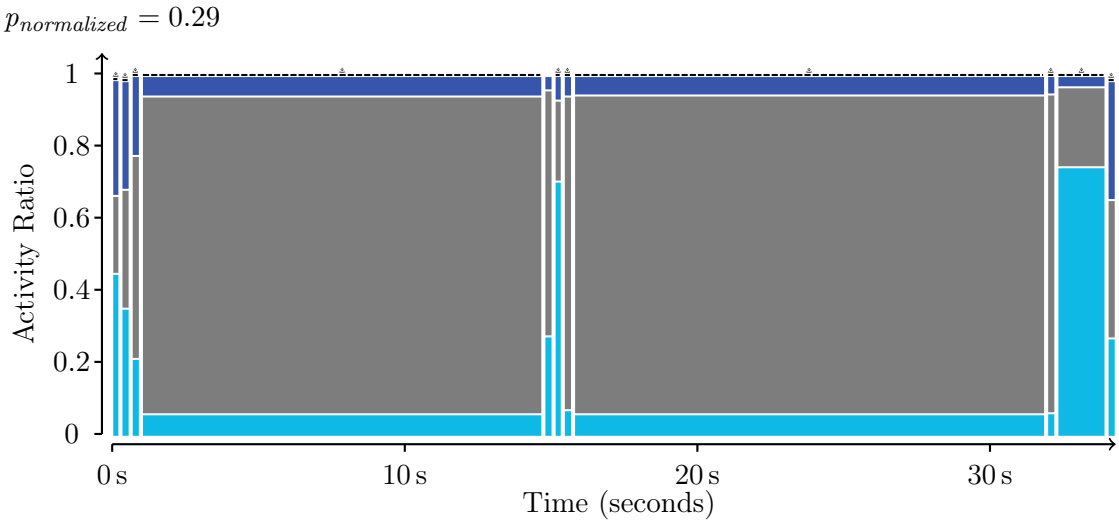


Figure Ext.5.3: Stacked bar chart showing the best partition of the perturbed GStreamer application trace, for $p_{normalized} = 0.29$, and the activity ratio (state total duration per time) for each event types (each one associated with a color: `DEBUG` (light blue), `LOG` (gray), `TRACE` (dark blue)). We clearly distinguish a behavior disruption around 15 seconds, which corresponds to the I/O stress occurrence.

method. We are able to highlight phases, as well as punctual perturbations, provoked by the resource sharing. We relate the performance and the behavior of these executions to the hardware resources and their topology **[13, 16]**.

We conclude this Chapter by confronting our method and the associated visualizations to the criteria defined by Elmqvist & Fekete. This analysis is summarized in Table 5.3 (page 87). The combination of *Lamarche-Perrin's method*, based on data aggregation, and punctual visual aggregation enables to meet all these criteria, in particular in the case of the stacked bar chart, which is the most detailed visualization we propose. We also justify the relevance of our method compared to automatic approaches that use pattern recognition [99] or trace comparison [19]. Indeed, our analysis method let us analyze a larger domain of applications, which are not necessarily periodic or regular. Moreover, our abstraction of the trace, the microscopic model, enables us to ensure a good performance whatever the number of events, compared to automatic approaches whose algorithmic complexity depends on the number of low level events.

However, our approach has some limitations: the temporal discretization influences the detection of perturbations and low scale phenomenons contained in the trace. The user potentially needs to adjust this setting, but he is limited by the bounds related to the algorithmic complexity and the screen size. We also detect potential issues with applications or systems whose behavior is particularly heterogeneous or loosely structured. Finally, the lack of the spatial dimension in the representation prevents us to relate the dynamic of the application and the temporal phenomenons to its structure. This motivates the contribution described the next Chapter, a spatiotemporal view also based on *Lamarche-Perrin's method*.

CHAPTER 6

Spatiotemporal Aggregation of Execution Traces

We have presented, in the previous Chapter, the benefits of our temporal analysis method to understand and debug embedded multimedia applications. We succeed in performing our analysis without representing the spatial dimension in our overview, because the temporal aspect is the most important. We also note that the software components that mainly compose our spatial dimension are loosely structured in our use cases. Consequently, we only start to analyze the spatial dimension once the *details-on-demand* step of Shneiderman's mantra has been reached, after having focused on the perturbations. In the case of MPI applications, the spatial dimension is particularly well structured, since we can easily build a deep hierarchy upon software and hardware components (*system* > *country* > *site* > *cluster* > *machine* > *process*). With our temporal aggregation method, we highlight phases and perturbations, but the lack of a spatial dimension prevents us to study the impact of the resources on the application performance. Indeed, the structural organization and the hardware component technologies, potentially heterogeneous, which compose the execution platform (CPU, memory, I/O, network) might induce behaviors and performance that are different depending on system subparts.

In **Chapter 6**, we face this limitation by proposing a novel contribution, a spatiotemporal analysis method. Its aim is to identify the influence of the application structure on its behavior (spatial aspect), to represent the evolution of its behavior over time (temporal aspect), but also to reveal behaviors bounded in time and space (spatiotemporal aspect). In the same way as the temporal aggregation method, we build this contribution upon *Lamarche-Perrin's method*. We represent both spatial and temporal dimensions using two axes. Time is still discretized in the form of intervals, whereas resources constituting the space are organized as a hierarchy. In general, this one follows the software and hardware component hierarchy. The surface defined by both dimensions is partitioned and the values associated with the spatiotemporal individuals contained in each part are aggregated. Thus, aggregated parts symbolize spatiotemporal area where the application behavior is considered as homogeneous.

The flow leading to the aggregated view is similar to the temporal aggregation method. We first elaborate a microscopic model discretizing the temporal dimension,

whereas the resource space is organized as a hierarchy. Its structure is given by the Equation 6.1 (page 91): it is an application of the time, the resource and the event type dimensions, in $\mathbb{R}^+$. The metrics we propose to instantiate this model are the same as the temporal microscopic model ones.

We design a new algorithm to aggregate simultaneously on space and time dimensions. We first start by the formalization of the aggregation process defined in Chapter 3 (the *dimensions*: the space and the time – the *operands*: the elements defined by the Cartesian product of the time slices and the resources (leaves of the spatial hierarchy) – the *constraints*: an aggregate is constituted by the Cartesian product of a time interval composed of contiguous time slices, and a node of the hierarchy; there is no overlap between the aggregates – the *conditions*: solve the best partition problem using the pIC as quality measure – the *operator*: the sum) **[10, 13]**. Then, we compare this method to other ways to aggregate over space and time. We show that an aggregation that is not determined by quality measures provides potentially the user with misleading representations. Indeed, it might aggregate parts that are heterogeneous, which leads to smoothing effects. In the case of an aggregation based on quality measures, but aggregating time and space sequentially instead of simultaneously, we prove that some partitions cannot be reached, which makes it less effective to represent some behaviors. At last, we describe the data structures we use **[10]** and detail the *quality computation*, given by Algorithm 6.1 (page 99), and the *best cut computation*, given by Algorithm 6.2 (page 100) **[10]**, as well as their respective complexity **[10, 13]**.

As the case of temporal analysis, we design visualizations to represent the parts and their contents. They are all built upon the same structure: a two dimensional matrix, whose abscissa axis is associated with the time and ordinate axis with the hierarchy, and whose cells correspond to the spatiotemporal individuals. These cells are then aggregated in the form of rectangles, according to the best cut computation result. In the case where attributes are scalars, we propose to use the color intensity to represent the amplitude of the values associated with each part. When these attributes are multidimensional, our first approach, the simplest, is a colored partition, based on the same principle than for the temporal aggregation. Because of the lack of information about the aggregate contents, our second approach is a modal representation: we represent the color of the event type whose associated attribute has the highest amplitude, while this amplitude is materialized by the color intensity **[10, 13]**. Finally, to ensure the fulfillment of Elmqvist & Fekete's criteria, each of these techniques has a visual aggregation mechanism. It consists in aggregating parts that are too small to be represented correctly alone into the parent node, using the same constraints than for the data aggregation (allowed aggregates are defined by the Cartesian product between the hierarchy nodes and the time intervals). We distinguish two types of visual aggregates. In the first one, aggregated parts share the same temporal pattern, i.e., no one has an intermediate temporal cut. It is the contrary in the second one **[10, 13]**.

We validate our method through the analysis of the NAS Parallel Benchmark use cases presented in Chapter 5. This time, we structure the space as a hybrid hierarchy composed of software and hardware components. The spatiotemporal analysis corroborates the temporal patterns we previously found with the temporal aggregation method. However, it adds new information: in one of the Conjugate Gradient executions (Figure Ext.6.1), we illustrate that the whole platform is subject to temporal perturbations **[10, 13]**. In

one of the Lower-Upper Gauss-Seidel solver executions (Figure Ext.6.2), we show that the clusters behave differently, and distinguish local temporal perturbations involving a cluster, from a completely heterogeneous behavior of another cluster during the execution **[10, 13]**. We also compare several executions of the same application on different platforms, whose differences appear clearly to the analyst thanks to our technique. Finally, we show the necessity of using a hierarchy thanks to an example where most of the hierarchy levels are removed, and whose analysis becomes difficult.

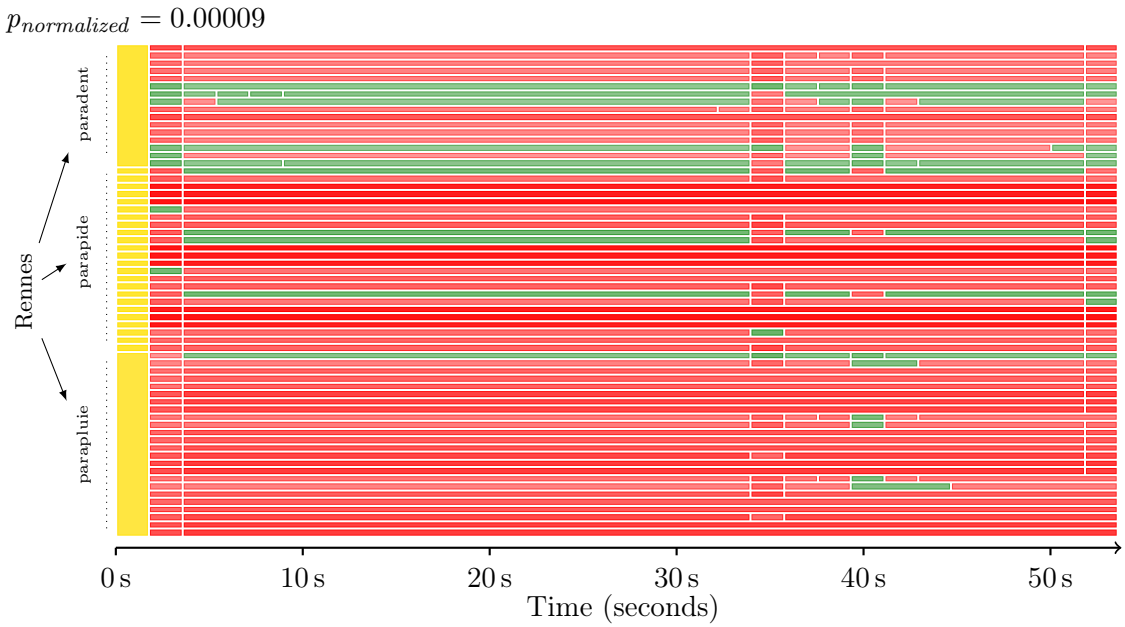


Figure Ext. 6.1: Spatiotemporal partition of the CG class C application execution on the Rennes' site of Grid'5000, 64 processes, showing the activity ratio mode (`MPI_Init` (yellow), `MPI_Send` (green), `MPI_Wait` (red)). We detect two temporal perturbations, touching machines whatever the cluster they belong to.

We also conclude the Chapter by confronting our contribution to the criteria defined by Elmqvist & Fekete. This is summarized in Table 6.1 (page 113). The combination of data aggregation and visual aggregation enables to respect all these criteria, in particular in the representation based on the mode. We also highlight the scalability issue fulfillment, since we provide a relevant overview for realistic use cases containing up to 700 processes. The main drawbacks of our technique are the limitations imposed by the hierarchical structure. For instance, some topologies cannot be represented. From a general way, our method requires that the user or the tracing infrastructure expresses a hierarchy which can potentially explain the application behavior. This operation is not always trivial, in particular if the hardware or software hierarchies are not the most convenient.

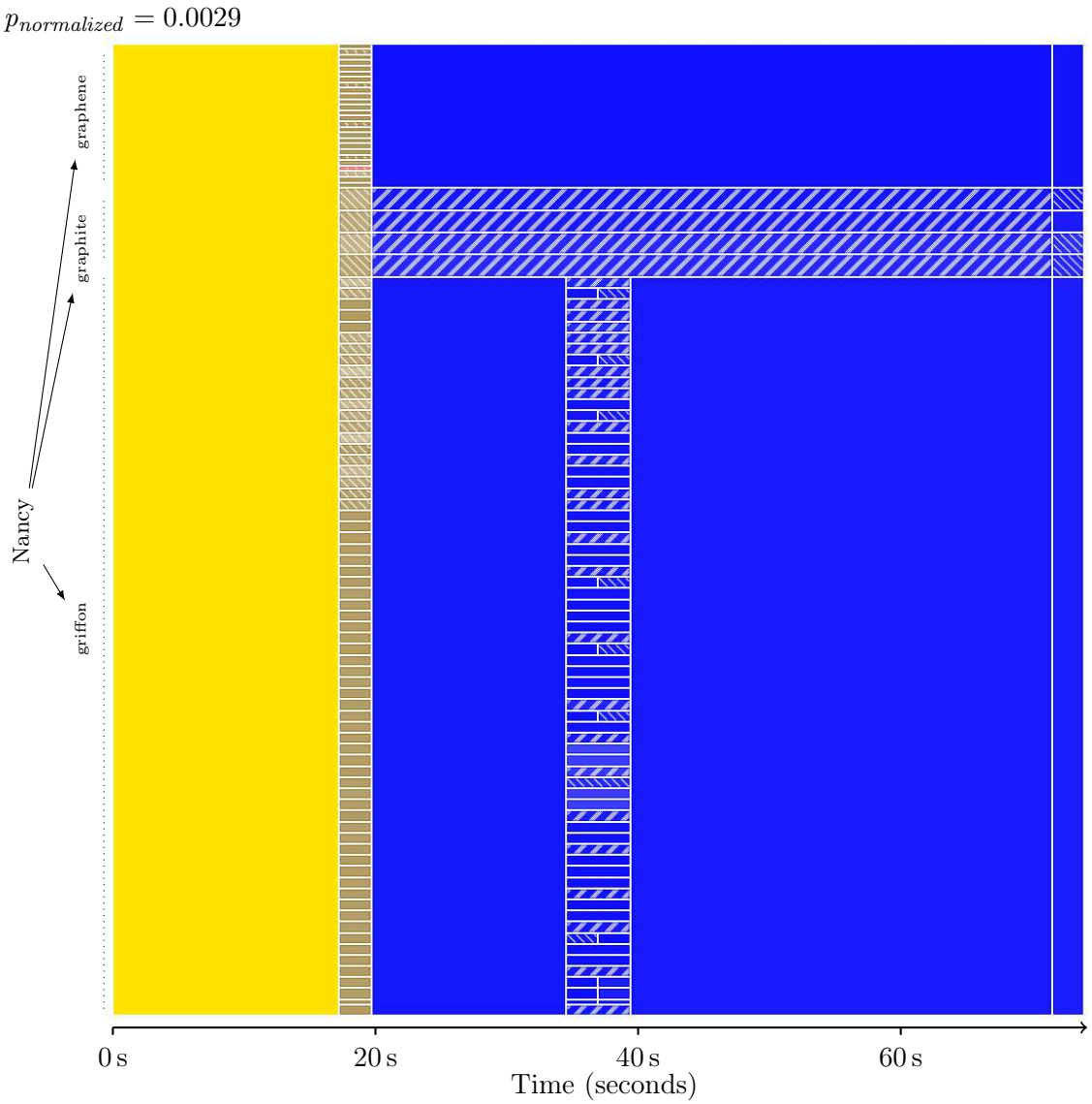


Figure Ext.6.2: Spatiotemporal partition of the LU class C application execution on the Nancy's site of Grid'5000, 700 processes, showing the activity ratio mode. (`MPI_Init` (yellow), `MPI_Recv` (blue), `MPI_Barrier` (brown)). In the computation phase, mainly composed of the `MPI_Recv` function in blue, we distinguish a punctual temporal perturbation located only on the cluster *griffon*, occurring around 38 seconds. *graphite*'s behavior is particularly heterogeneous all along the computation phase, whereas *graphene*'s is homogeneous.

CHAPTER 7

Implementation of the Multidimensional Aggregation Techniques into the Ocelotl Analysis Tool

In Chapter 7, we describe the implementation of both contributions of Chapters 5 and 6 into a software, Ocelotl[1], whose a screenshot is given by Figure Ext.7.1.

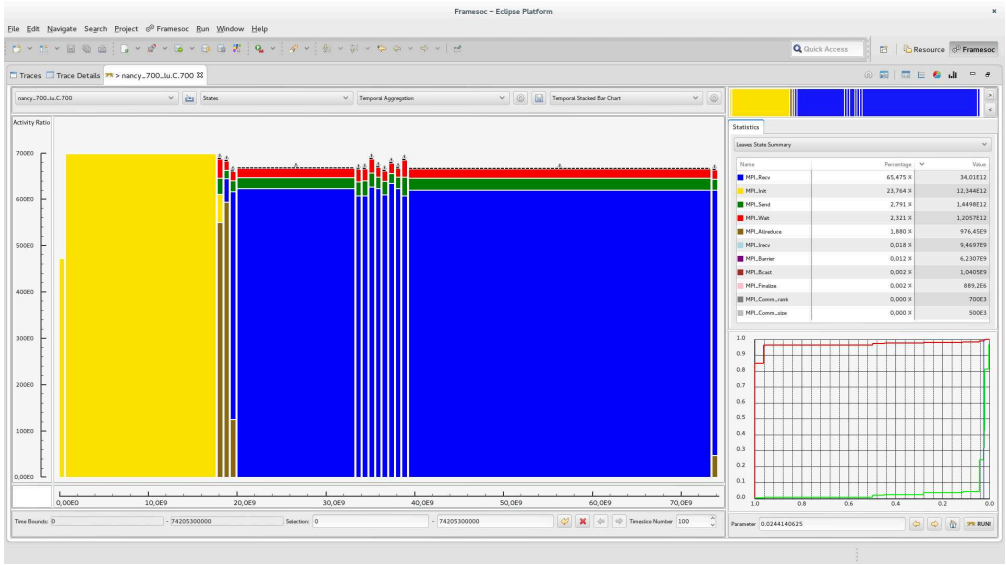


Figure Ext.7.1: Screenshot of Ocelotl, the analysis tool developed during this thesis.

We start with the description of the aspects related to the software design, its architecture and its functionality. Ocelotl is an Eclipse plug-in mainly written in Java. It is also a module of Framesoc [11, 12, 18][2], the infrastructure of trace management and

[1] http://soctrace-inria.github.io/ocelotl/
[2] http://soctrace-inria.github.io/framesoc/

analysis developed in the context of the SoC-Trace project. This infrastructure tackles several problems : the efficient management of large traces, the variety of trace formats and analysis techniques and the need of a protocol to share analysis results between several modules. Traces and analysis results are stored into databases, thanks to a generic data model. Framesoc provides the user with an API to easily access the trace. Several importers are natively present, which enables to work with multiple trace formats (Pajé [30,31], OTF2 [29], Paraver [33], CTF [34], KPTrace [36] and GStreamer). Framesoc also features classic analysis tools, such as bar charts, pie charts and a space-time diagram. At last, Framesoc allows to add dynamically new plug-ins, like Ocelotl, thanks to the extension point mechanism proposed by Eclipse[3]. Ocelotl's architecture follows globally the MVC model. We distinguish, in particular, a core and a graphical interface. This last one is composed of several components, summarized in Figure 7.2 (page 118). The most remarkable are the following. The main view (B) corresponds to our aggregation technique. A secondary temporal overview (C) is automatically computed using the temporal aggregation method, in order to keep track of the position in the trace during a temporal zoom. Statistics (D) are synchronized with the interaction, giving more details on a particular area. The user can interact with the information loss and complexity reduction curves (E), to select a new value of p and get the corresponding best partition. Regarding the core, we highlight several particularities. Its modular structure gives the analyst the opportunity to add new modules dynamically (*metrics* used to fill the microscopic model, *aggregation algorithms* based on temporal, structural, both, or even other dimensions, *visualizations* representing the partition, and *statistic components*) without modifying Ocelotl's code thanks to the extension points provided by Eclipse. The developer expresses the compatibility between the different modules through a mechanism based on the correspondence between *tags* associated with each of them. Ocelotl performs the analysis as a sequential treatment, which involves successively each of these modules. At last, we present the *lpaggreg* library[4], which implements the aggregation algorithms presented in Chapters 5 and 6. It is written in C++, and interfaced with Ocelotl through JNI[5].

Then, we demonstrate how our implementation integrates completely Shneiderman's mantra. We describe, for each step, the techniques that help its fulfillment. Regarding the *overview*, we consider that our temporal and spatiotemporal aggregation methods and their related visualizations meet the objective. Moreover, we note the presence of a second overview, keeping track of the position in the trace. The *zoom* step is realized through interaction mechanisms. We propose temporal and spatiotemporal selections. After selecting an area, the user performs the analysis again, on this subpart of the trace. *Filtering* is possible through advanced configuration. It can be applied on the space and the event type dimensions. At last, the *details-on-demand* step is accomplished thanks to the statistics view, which is synchronized with a selection, but also thanks to the possibility to switch to a more detailed view, like a space-time diagram, on the trace subpart defined by this selection. Furthermore, the user can get more information about a spatiotemporal visual aggregate by interacting with it, which opens a new window showing its content.

[3]http://help.eclipse.org/luna/index.jsp?topic=%2Forg.eclipse.pde.doc.user%2Fconcepts%2Fextension.htm

[4]https://github.com/dosimont/lpaggreg

[5]https://github.com/dosimont/lpaggregjni

At last, we broach the performance and interactivity aspects of our implementation. We list three points impacting the overall performance of our tool: the trace reading, which gets the events from the database to build the microscopic model, the aggregation process, and the visualization rendering, which involves the visual aggregation. These three steps constitute a cycle. Interacting triggers a new cycle, partial or complete. Improving the overall performance consists thus in increasing the performance of each of these steps, and in avoiding to reiterate a full cycle whenever possible.

The trace reading is the step having the longest duration. It mainly depends on the trace size and the storage support technology. This operation is realized through the Framesoc's API, thanks to a query to the database, which is composed of three conditions to evaluate: a temporal period, the resources to take into account, and the desired event types. This generates a result set, managed by the DBMS, which contains the events corresponding to the specified query. The microscopic model is computed sequentially: the events are accessed through an iterator, which is blocking while the next event is not available. The query complexity influences the duration of the result generation [101], which is the limiting factor. We propose, in order to minimize the reading time, to simplify this query and remove useless and redundant conditions. The first simplification step consists in removing unnecessary filtering on each dimension (for instance, when the evaluated time period is the same as the trace temporal bounds, filtering the time dimension is not necessary). The second simplification step is more advanced. For example, we propose to replace the event type filtering by a simpler condition, based on the categories, whenever possible.

Query simplification offers a substantial gain, up to 35% for the whole trace reading, as shown by Figure 7.9 (page 128). However, it is not enough to ensure a good interaction in the case of voluminous traces containing billion of events (2000 seconds are needed for a 2 billion event trace). Our aggregation method is based on an abstraction of the trace, the microscopic model, whose content and structure are identical for a particular trace and settings (metric, time interval, time slice number, resources and event types selected). We propose to save this microscopic model as a file, and reuse it for the upcoming analysis sessions **[13]**. In particular, we describe two policies: the microscopic model building from a cache whose time slice number fits perfectly (same number or multiple), and the use of an approximation in the case where it does not. Figure 7.12 (page 131) shows, in both cases, the important gain provided by this technique.

We dedicate a section to the aggregation process **[13]**. We describe its pipeline and show that changing the partition necessitates only the recomputation of the best cuts. Then, we confirm experimentally the aggregation algorithm complexity. In particular, we explain how to reduce the computation duration by adjusting the time slice number, and, in a second time, by applying a pre-aggregation on the spatial dimension. At last, we discuss about the influence of the visual aggregation on the rendering duration.

Finally, we position Ocelotl's performance in the state of the art [3]. Regarding the computation time, we first summarize the time required by each step of the analysis pipeline for our use cases, in Table 7.1 (page 137). Concerning the analysis feasibility, we give, for each method (temporal or spatiotemporal), the maximum size allowed for the microscopic model dimensions, formalized as a global *budget* that involves temporal, spatial, and event type dimension size (Equation 7.2 and 7.3, page 136). For example, we are able to aggregate microscopic models composed of 100 000 resources, 10 event

types and 100 time slices in the case of a temporal aggregation, and composed of 10 000 resources, 10 event types, and 30 time slices in the case of a spatiotemporal aggregation. These numbers does not depend on the trace size and its event number (but on its space and event type dimensions). This ranks Ocelotl among the most scalable tools of the domain, regarding the spatial dimension. Regarding the trace size, we are able to read traces up to 2 billion events (about 68 gigabytes). Only Vampir [48] and Scalasca [102] are better, since they use parallel trace formats.

CHAPTER 8

CONCLUSION

In this thesis, we have presented several contributions that solve the scalability problems of traditional spatiotemporal visualizations: a temporal overview and a spatiotemporal overview, both built upon data and visual aggregations, and a software that implements them. We integrate Shneiderman's guideline that prescribes a top-down approach and take into account the limits imposed by the screen resolution, the performance, and the user's perceptive and cognitive capabilities.

Our contributions are based on a method, originally designed by Lamarche-Perrin, which reduces the complexity by aggregating in priority data whose values are close to each other, from the information theory point of view. This brings an additional knowledge on the system behavior by discriminating the behavioral homogeneity of its entities. The user can change the level of detail through a trade-off between the desired complexity reduction and the tolerated information loss, whose balance can be adjusted. This dynamic approach enables the emergence of phenomenons of different magnitude and scale.

We adapt this method to trace analysis. We first abstract the traces to discretize their dimensions and make them compatible with Lamarche-Perrin's process. Then, in the case of the temporal aggregation, we significantly extend an existing aggregation algorithm, while, in the case of the spatiotemporal aggregation, we design an algorithm from scratch. For both methods, we associate visualizations. Since the data aggregation is potentially applied in a non homogeneous way and might produce views where some elements are still too small to be correctly represented, we complete it with visual aggregation. We validate our methods with embedded multimedia and MPI parallel computing applications where we highlight perturbations, execution phases and spatiotemporal behavior translating the influence of the structure and the resources. All these behaviors are difficult to detect using traditional approaches.

We implement these analysis methods into Ocelotl, a tool designed and implemented during this thesis. It supports the steps of Shneiderman's guideline requiring interaction, like zooming and filtering, and proposes several solutions to get more details, for instance by switching to a more detailed representation like a space-time diagram, or, by synchronizing a statistics view with the current selection. Ocelotl is extensible and analysts can add their own aggregation or visualization modules that fit heir particular

needs. At last, we also focuses on the performance aspects and implement techniques to minimize the interaction time. Performance is evaluated through tests that enable to position Ocelotl according to the state of the art. We succeed in analyzing realistic use cases containing up to 700 processes, 218 million events and 12 gigabytes in 5 minutes, if we read the whole trace, and in few seconds, if we use permanent caches. We evaluate practical limits on a personal computer to 2 billion events (about 68 gigabytes), and 100,000 resources for the temporal aggregation and 10,000 for the spatiotemporal aggregation, thanks to benchmarks realized on synthetic traces.

We foresee several perspectives as a continuation to the works described in this thesis, related to theoretical aspects, or their implementation. Some of them relate to *Lamarche-Perrin's method*. Although this method is a generic framework, all the applications employ the same aggregation operations, based on the sum, and the same quality measures, which are dependent on this operator. We propose to extend the method to different aggregation operators and design quality measures that are compatible. We also are interested in elaborating new aggregation algorithms, applied to different dimensions. In particular, we think about applications like graph aggregation, surface aggregation (ideal to represent the memory allocation, for instance) or, in a more general way, algorithms involving more than two hierarchical or ordered dimensions. The metrics we propose in our microscopic models are mainly based on the sum or the mean. We think about other kind of metrics involving other aggregation operators, the simplest being the minimum, the maximum and the median.

The data complexity reduction is non homogeneous since the partition and aggregation process depend on the values of the microscopic model attributes. For this reason, it does not completely ensure the scalability. We fulfill it with visual aggregation to prevent the representation from containing visual artifacts. We also give information about the aggregate content, in particular in the spatiotemporal case where aggregates indicate the presence or the absence of non shared temporal cuts. Nevertheless, we consider that efforts have to be done to improve this technique. We think about pseudorandom procedural textures, like the Perlin's noise [103], to represent the level of complexity reduction or information loss of a visual aggregate. Holten *et al.* [104] enumerate several methods of colored texture generation which could serve as support for this new feature.

Some potential improvements concern the performance. The aggregation algorithm can be parallelized, in particular the quality computation part, to reduce its duration. The trace reading offers also a important potential gain regarding the analysis computation time: the parallel files used by Vampir [48] are efficient for traces of several terabytes. In the long run, we wish to spread our tool computations on a distributed system. Regarding the data structures, we can take advantage of the low density of the microscopic models, which contain more than 99% of 0 in all of our use cases, for example during the quality computation. At last, increasing the use of data cache seems to be a good solution to minimize computation on further analysis sessions. We also think about prefetching and anticipation of the user actions.

Finally, we can think about other kind of applications, since our aggregation methods are generic. We intend to use Ocelotl to represent traces previously abstracted by pattern recognition and data mining techniques. We also think about the combination with fine grained trace comparison techniques, inspired of DNA sequences researches for instance [106], which would be guided by our overview, this last one offering an overall

qualitative comparison. Other applications are possible, like the memory allocation analysis [107]. In a more general way, we can extend the application field to the information visualization domain, where our method can fit to the analysis of economic, financial or demographic data.