HAL
open science

Un service hiérarchique distribué de partage de données pour grille

Loïc Cudennec

To cite this version:

Loïc Cudennec. Un service hiérarchique distribué de partage de données pour grille. Rencontres francophones du Parallélisme (RenPar '18), Haute Ecole Spécialisée de Suisse Occidentale (HES-SO), Feb 2008, Fribourg, Suisse. inria-00256827

HAL Id: inria-00256827

https://inria.hal.science/inria-00256827v1

Submitted on 18 Feb 2008

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.

Un service hiérarchique distribué de partage de données pour grille

Loïc Cudennec

IRISA, INRIA Rennes,
Campus universitaire de Beaulieu,
35042 RENNES - France
loic.cudennec@irisa.fr

Résumé
Les besoins grandissants en terme d'espace de stockage requis par les applications scientifiques (en énergie atomique, cosmologie, génétique, etc) motivent la recherche de solutions adaptées pour la gestion des données à grande échelle. Il devient primordial d'offrir un accès efficace et fiable à de grandes quantités de données partagées. Nous pensons que, contrairement à l'approche actuelle basée sur le transfert explicite des données, l'utilisation d'un modèle d'accès transparent aux données permet de simplifier le modèle de programmation. Une telle approche est illustrée à travers les notions de système de fichiers distribué à l'échelle globale et de service de partage de données à l'échelle de la grille. Ce papier présente un système de stockage hiérarchique pour grille, tirant partie de la rapidité des accès en mémoire physique et de la persistance d'un stockage sur disque. Notre proposition a été validée par un prototype couplant le système de fichiers Gfarm avec le service de partage de données JuxMem.

Mots-clés : grille de calcul, partage de données, persistance

1. Introduction

La gestion des données prend une part de plus en plus importante dans le contexte du calcul scientifique. Le type d'application visé nécessite de partager des données entre différents sites, dans le cadre d'un travail collaboratif. C'est par exemple le cas pour les applications basées sur le couplage de code, où différents programmes s'exécutent en s'échangeant, de manière épisodique, de grandes quantités de données. La gestion est complexifiée par la taille des données manipulées et les besoins requis par les applications en terme de disponibilité, performance d'accès et persistance. Elle est aussi tributaire de la taille de l'infrastructure utilisée pour supporter le calcul. Dans ce papier nous nous focalisons sur les infrastructures de type grille de calculateurs que nous définissons comme une fédération de grappes.

Le partage des données a été largement étudié dans le contexte du stockage distribué en mémoire physique. Les systèmes à mémoire virtuellement partagée (MVP) permettent de partager des données de manière transparente à l'échelle de la grappe. L'accès aux données s'effectue grâce à l'utilisation d'un identifiant global unique sans se préoccuper de la localisation et du transfert. Cette approche est étendue à la grille à travers la notion de *service de partage de données*, un système qui tire partie de l'approche MVP pour les accès transparents et la gestion de la cohérence des données, ainsi que du modèle pair-à-pair (P2P) pour la tolérance à la volatilité et le passage à l'échelle. Une illustration d'un tel service est donnée par la plate-forme JuxMem[5].

Le service JuxMem stocke les données en mémoire physique, principalement pour profiter de la rapidité d'accès supérieure aux accès disque. Ce choix limite la taille maximale des données pouvant être accueillies par le service, l'espace de stockage en mémoire physique étant environ cent fois moins important que l'espace disque. Ceci est clairement une limitation pour les applications nécessitant la gestion de grandes masses de données. Une solution est d'adjoindre au service JuxMem un stockage secondaire sur disque afin d'augmenter l'espace de stockage. Ceci a par ailleurs l'avantage d'offrir des propriétés supplémentaires de persistance des données sur disque. Dans ce papier nous explorons cette approche, avec l'objectif de construire un service de partage de données hiérarchique distribué. Le stockage secondaire est assuré par Gfarm[15], un système de fichiers distribué pour des grilles.

La suite de ce papier est organisée comme suit. La section suivante propose un état de l'art. La section 3 présente les systèmes JuxMem et Gfarm. La contribution est expliquée en section 4 et une évaluation préliminaire est donnée en section 5. Enfin, la section 6 conclut et discute les travaux futurs.

2. État de l'art

Les projets de recherche touchant à des domaines tels que l'énergie atomique, la cosmologie ou la génétique utilisent des applications scientifiques qui génèrent une masse de données de plus en plus importante. À titre d'exemple, le projet LHC (pour *Large Hadron Collider*) du CERN devrait rapidement dépasser la production de plusieurs peta-octets par an, gérées par le projet MONARC [14]. Les deux principaux objectifs de tels systèmes de gestion de données sont 1) la possibilité de stocker de grandes quantités de données de manière persistante et 2) assurer des accès rapides en lecture et en écriture.

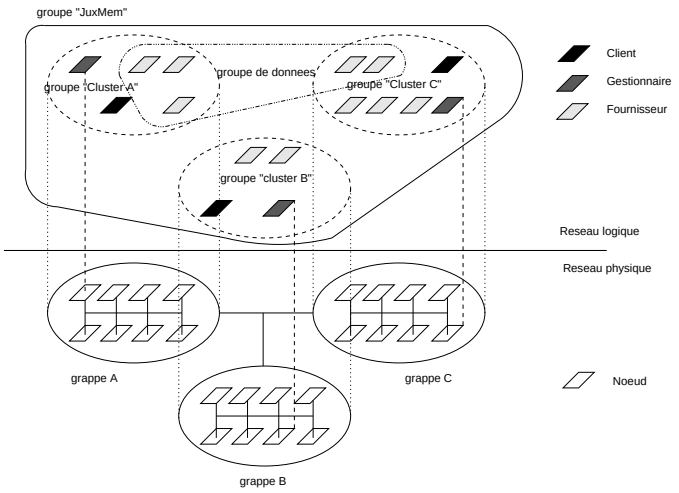

FIG. 1 – Organisation hiérarchique de JuxMem

Dans le contexte du partage de données sur grille, différentes techniques ont été proposées afin d'offrir un modèle d'accès aux données efficace pour le programmeur. La notion la plus largement répandue se base sur le transfert *explicite* des données entre clients et serveurs de calcul. C'est, par exemple, l'approche retenue par la plateforme Globus [10], dont le protocole associé, GridFTP [3], permet l'authentification, le transfert parallèle et la reprise du transfert des données. Plusieurs systèmes à base de catalogues de données ont été proposés pour recenser manuellement tous les réplicas. La cohérence des données ainsi que leur localisation restent cependant à la charge de l'utilisateur. Le réseau logistique de données IBP [8] propose un système de stockage à grande échelle composé de zones de mémoire distribuées sur internet. L'utilisateur loue ces zones mémoire pour y stocker temporairement ses données et y accèder de manière efficace. Là encore, les transferts doivent être opérés explicitement par l'utilisateur et la cohérence des données n'est pas gérée par le système. D'autres systèmes comme Stork [12] ou SRB [7] offrent également des méchanismes pour localiser et transférer les données de manière explicite.

Le problème du partage des données a aussi été traité au niveau des systèmes de fichiers. L'un des intérêts majeurs de tels systèmes est la propriété de persistance des données sur disque. GridNFS [11] est un système de fichiers distribué basé sur le protocole NFS. Il permet des accès transparents et sécurisés en lecture mais ne donne que de faibles garanties sur la cohérence et la tolérance aux défaillances. LegionFS [16] est un système de fichiers distribué offrant des accès parallèles et des performances supérieures à un NFS classique. Les mécanismes de réplication et de cohérence des données ne sont cependant pas disponibles. Gfarm[15] est un système de fichiers distribué conçu pour les fédérations de grappes. Il permet des accès transparents en lecture, de manière parallèle, à des fragments de fichiers répliqués. Dans sa version 1, Gfarm ne prend pas en charge la cohérence des données.

Par ailleurs, dans le contexte du calcul parallèle haute performance, les systèmes à mémoire virtuellement partagée (MVP) permettent de donner l'illusion au programmeur qu'un ensemble de mémoires physiques distribuées sur plusieurs machines n'en forment qu'une. Le modèle de programmation qui en découle permet donc aux développeurs de ne plus se soucier des problèmes liés à la localisation physique des données. Des systèmes tels que Ivy [13] ou TreadMarks [4] offrent des propriétés de transparence des accès en terme de localisation et de transfert, et prennent en charge la cohérence des données répliquées. Ces systèmes ont cependant été conçus pour des architectures matérielles homogènes de

type grappe et ne passent pas à l'échelle.
Le service de partage de données pour grille JuxMem[5] se présente comme une approche hybride entre les systèmes à mémoire partagée et les systèmes pair-à-pair (P2P). Il offre des accès transparents aux données dans des environnements dynamiques en utilisant des méchanismes pour la tolérance aux fautes et la cohérence des données. Nous pensons qu'un tel service constitue une solution pertinente au partage de données sur grille pour les applications scientifiques travaillant sur de grandes masses de données. Afin d'augmenter la capacité de stockage du système et de consolider les propriétés de persistance des données stockées dans JuxMem, nous faisons le choix d'adjoindre le système de fichiers distribué Gfarm comme stockage secondaire.

3. Deux systèmes de stockage pour une approche hiérarchique

Cette section présente les deux systèmes de gestion de données choisis pour construire une mémoire partagée, distribuée et hiérarchique : le système de fichiers distribué Gfarm au niveau du disque et le service de partage de données JuxMem au niveau de la mémoire physique.

3.1. JuxMem : Un service de partage de données

Pour la plupart des applications conçues pour s'exécuter sur un environnement de type grille, le partage de données repose sur le modèle d'accès *explicite* aux données, ce qui induit une complexité non négligeable dans le code des applications. Afin de laisser l'utilisateur se concentrer sur son code métier plutôt que sur la gestion des données, la notion de service de partage *transparent* de données à été introduite dans [5] et illustrée par la plate-forme logicielle experimentale JuxMem. Sa conception est basée sur une approche hybride tirant parti des systèmes à MVP et des systèmes P2P.

```
Processus1
BEGIN

 A = malloc()
 acquire(A)
 A = [[7,..]..[4,..]]
 release(A)

END
```

```
Processus2
BEGIN

 acquireRead(A)
 print(A)
 release(A)

END
```

FIG. 2 – Exemple de pseudo-code utilisant JuxMem

L'interface utilisateur associée au service de partage de données s'inspire du modèle *MVP*. L'utilisateur alloue dynamiquement l'espace nécessaire au stockage de la donnée partagée. Celle-ci est caractérisée par un *identifiant global unique* et les accès sont effectués par un pointeur en mémoire locale. Lors de la phase d'allocation, l'utilisateur indique le degré de réplication de la donnée pour déterminer le nombre de grappes ainsi que le nombre de réplicas par grappe devant être créés. À charge ensuite au service de sélectionner les fournisseurs JuxMem susceptibles de stocker une copie de la donnée. Ces fournisseurs forment un *groupe de données*. La procédure d'allocation retourne un identifiant global, le *data ID*, utilisé pour accèder à cette donnée à partir de n'importe quel client connecté au service.
Ces multiples copies introduisent la problématique de la cohérence. JuxMem permet d'associer à chaque donnée stockée dans le service un protocole de cohérence particulier. Le protocole actuellement implémenté se base sur le modèle de la cohérence à l'entrée introduit dans [9]. Dans ce modèle, un verrou est attaché à chaque donnée. Ce verrou permet : soit des accès exclusifs en écriture, soit des accès éventuellement concurrents en lecture seule (MRSW). Le protocole garantit la cohérence de la donnée si les accès s'effectuent après la prise du verrou comme montré en figure 2. Les modifications effectuées sur la donnée sont transférées aux fournisseurs lors du relâchement du verrou.
La réplication des données est utilisée dans JuxMem pour offrir le support à la tolérance aux fautes. Le groupe de données, formé de tous les fournisseurs hébergeant une copie, est aussi appellé le *groupe de données global* (GDG). Afin d'assurer un bon comportement lors du passage à l'échelle, le GDG est construit sur un modèle hiérarchique, partitionné en un ou plusieurs *groupes de données locaux* (LDG). Chaque LDG correspond à l'ensemble des copies stockées sur une même grappe. Ces groupes LDG et GDG sont dits auto-organisants, c'est à dire qu'ils restent stables, en terme du nombre de membres, sous certaines conditions de fautes (voir section 3.3). Ils sont régis par un algorithme de gestion de groupe tolérants aux fautes [6].
L'architecture de JuxMem, présentée sur la figure 1, montre une corrélation forte entre les topologies logiques et physiques. À l'instar de la grille, vue comme une fédération de grappes de calculateurs, l'architecture de JuxMem suit une organisation hiérarchique. Ceci est particulièrement pertinent pour

prendre en compte les différences de latences que l'on peut trouver entre des liens réseaux intra-grappes et inter-grappes. Les protocoles de cohérence et les algorithmes liés à la tolérance aux fautes sont ainsi conçus pour tirer parti de ces propriétés en minimisant les coûts des communications. L'implémentation de JuxMem utilise actuellement la plate-forme pair-à-pair Jxta [2].

3.2. Gfarm : Un système de fichiers distribué

Le système de fichiers distribué Gfarm[15] vise des applications manipulant des données de l'ordre du peta-octet. L'idée est d'offrir un service inter-organisations pour le partage de données, en fédérant les disques locaux des participants et en décentralisant les accès. Pour des raisons de performance et de tolérance aux fautes, les fichiers sont fragmentés et répliqués sur les différents nœuds de stockage. Si la procédure de réplication reste à la charge de l'utilisateur, les accès en lecture s'effectuent, eux, de manière transparente vis-à-vis de la localisation des données. La bande passante est par ailleurs améliorée et optimisée pour le passage à l'échelle grâce à l'utilisation des entrées-sorties parallèles.

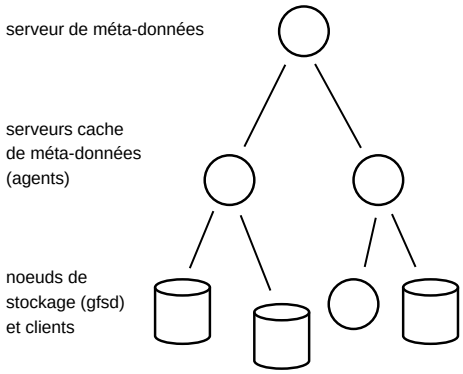

FIG. 3 – Topologie de Gfarm

L'écriture d'un fichier dans Gfarm (version 1) s'effectue de manière unique. Une fois créé, le fichier n'est plus modifiable et il faut en créer un nouveau lors de l'écriture suivante. Cette stratégie déplace le problème de la cohérence des données au niveau applicatif, les seuls accès concurrents possibles s'effectuant en lecture. La topologie de Gfarm(figure 3) est de type arbre et met en jeu un serveur centralisé ayant la charge de gérer les méta-données, des serveurs de cache pour distribuer la charge des accès aux méta-données (appellés *agents*), des nœuds de stockage et des clients connectés à un *agent*.

3.3. Vers un stockage distribué hiérarchique

JuxMem a été conçu pour prendre en compte la dynamicité d'un environnement de type grille grâce, notamment, à la technique de réplication des données. Ces données sont stockées en mémoire physique, ce qui les rend sensibles aux défaillance de nœuds. Le modèle de fautes considéré est le modèle de défaillance franche (*fail-stop*) dans lequel les participants se comportent normallement puis stoppent toute activité soudainement. Les canaux de communications sont considérés comme équitables (*fair lossy channels*) et garantissent que si l'emetteur envoie une infinité de fois le même message, le recepteur finit par recevoir ce message. L'algorithme de consensus utilisé pour assurer la gestion des groupes de copies ne peut tolérer qu'un nombre de défaillances inférieur à la moitié de la taille du groupe. Si l'ensemble des fournisseurs appartenant à un même groupe de données tombent en panne dans un espace de temps trop petit pour que le protocole d'auto-organisation du groupe puisse intégrer de nouveaux membres, alors la donnée est considérée comme perdue. Un tel scénario est envisageable, dans le contexte de la grille, lors de l'arrêt de fonctionnement non prévu d'une grappe entière, et donc d'un grand nombre de nœuds susceptibles de stocker des copies. La perte d'une donnée dans JuxMem est dommageable pour des applications scientifiques de longue durée, manipulant des données de taille conséquente, qui devront faire appel à des mécanismes classiques de points de reprises (*checkpointing*). Nous pensons qu'il est plus efficace dans ce type de scénario d'assurer la persistance des données en

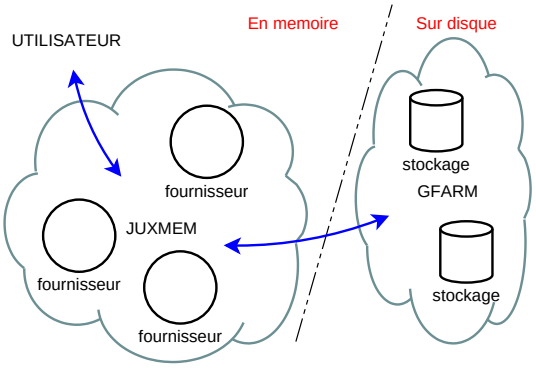

FIG. 4 – Architecture de JuxMem - Gfarm

utilisant un stockage secondaire sur disque.
Le stockage de données sur disque se distingue du stockage en mémoire physique par sa capacité à mieux supporter les redémarrages du matériel. Il offre par ailleurs un espace de stockage plus conséquent que la mémoire physique et est de ce fait tout indiqué pour remplir le rôle de stockage secondaire. Dans le contexte du service de partage de données pour la grille, ce stockage secondaire doit être distribué, global et permettre des accès rapides aux fichiers sur de larges infrastructures. Notre approche consiste donc à utiliser le système de fichiers distribué Gfarm pour offrir un stockage secondaire sur disque au service de partage de données JuxMem. La figure 4 montre qu'il n'y aucune intéraction directe entre le système de fichiers et l'utilisateur. Ce dernier utilise l'interface de programmation de JuxMem, l'accès aux disques étant rendu transparent par le service de partage de données.

4. Intérations entre les deux niveaux de la hiérarchie

Les intéractions mises en place entre les deux systèmes de stockage ne nécessitent pas de partager les mêmes nœuds physiques, laissant ainsi une grande liberté de configuration de leurs topologies respectives. Ce choix est motivé par le fait que les nœuds pouvant offrir de bonnes caractéristiques pour JuxMem (taille de la mémoire physique) ne sont pas nécessairement les mieux appropriés pour Gfarm (rapidité et capacité de stockage des disques dur).

4.1. Sauvegarde des données dans Gfarm

Le système de fichiers Gfarm est utilisé par JuxMem comme stockage secondaire. Le protocole d'auto-organisation du groupe utilisé dans JuxMem est basé sur la notion de *leader*. Le *leader* est élu parmi les fournisseurs ; il existe un *leader* pour chaque LDG et un pour le GDG. Une solution simple retenue dans un premier temps consiste à utiliser le *leader* du GDG, comme un mandataire ayant la charge de l'écriture dans Gfarm lors de chaque mise à jour de la donnée dans JuxMem. Un tel choix permet d'éviter les écritures concurrentes dans Gfarm.
Chaque fois que le verrou en écriture est relâché, la donnée est mise à jour dans la mémoire locale de tous les fournisseurs du groupe. Le fournisseur *leader* du groupe global crée alors un nouveau fichier contenant la donnée dans Gfarm, dont le nom est obtenu par concaténation de l'identifiant JuxMem avec le numéro de version de la donnée. Ce fichier est enfin répliqué sur plusieurs nœuds de stockage en utilisant la fonction de réplication fournie par Gfarm. Ce degré de réplication peut être indiqué par l'utilisateur lors de l'allocation de la donnée dans JuxMem.

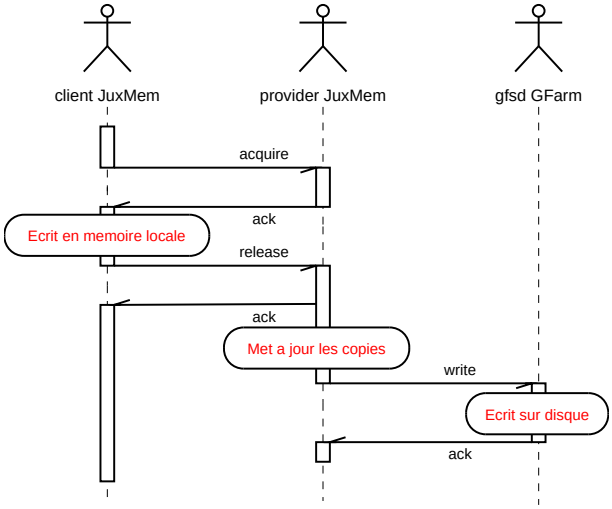

FIG. 5 – Intéraction simple entre JuxMem et Gfarm

La figure 5 décrit les intéractions mises en jeu entre les trois rôles principaux (client JuxMem, fournisseur et nœud de stockage Gfarm) lors d'une écriture. Ce diagramme de séquence propose un scénario simple composé de la séquence *acquisition du verrou*, *écriture* et *relâchement du verrou*. Une fois le verrou obtenu, le client JuxMem est le seul à modifier la donnée. Les modifications sont ensuite envoyées aux fournisseurs lors du relâchement du verrou. Un message d'acquittement permet au client de poursuivre son calcul pendant que le *leader* écrit dans Gfarm. Ce dernier point confère au système un fonctionnement asynchrone dont les clients peuvent tirer parti, n'ayant pas besoin d'attendre que toutes les copies soient mises à jour Gfarm pour continuer. Afin de maintenir la cohérence de la donnée et l'atomicité de sa mise à jour, le verrou qui lui est associé ne peut être re-attribué tant que l'écriture dans les deux systèmes ne soit effectuée. Le bénéfice d'une telle stratégie n'est possible que si la fréquence des écritures

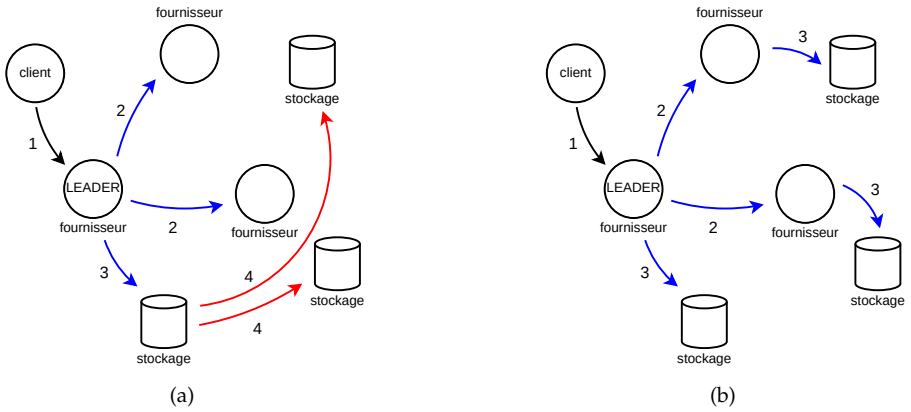

FIG. 6 – Prendre en compte le parallélisme

d'une même donnée ne dépasse pas un certain seuil au delà duquel les clients doivent attendre les mises à jour dans JuxMem et Gfarm, créant alors un point de contention. Ce seuil dépend principalement de la taille de la donnée et des propriétés des liens réseaux employés.

4.2. Recouvrement des données dans JuxMem

Le recouvrement consiste à lire une donnée dans le système de fichiers pour la stocker dans le service de partage de données. Cette opération peut être motivée par deux raisons : 1) une application cliente à JuxMem effectue un accès sur une donnée qui n'est plus stockée dans le service pour cause de défaillance et 2) une application cliente aimerait rehabiliter une ancienne version de la donnée en cas de reprise d'un calcul suite à une défaillance. À l'instar de l'écriture dans Gfarm, une solution simple consiste à utiliser le *leader* du groupe global de la donnée pour effectuer la lecture à partir du système de fichiers. Lorsque les fournisseurs sont solicités pour transférer une donnée vers un client JuxMem, ceux-ci vérifient d'abord que la donnée est stockée dans leur mémoire physique. Si non, le *leader* effectue une recherche dans Gfarm en utilisant son identifiant puis alloue de l'espace en créant un nouveau groupe pour cette donnée. Le *leader* récupère alors la donnée, met à jour sa mémoire physique et propage les modifications aux autres membres du groupe ainsi qu'au client.

4.3. Prendre en compte le parallélisme

La gestion des intéractions entre JuxMem et Gfarm de manière centralisée introduit les problèmes classiques de performance, de passage à l'échelle et de tolérance aux fautes. La figure 6(a) présente les transferts réseaux nécesssaires lors d'une écriture. Seuls les messages contenant la donnée à mettre à jour sont représentés. L'étape 4, qui correspond à une copie de disque à disque, peut s'avérer particulièrement coûteuse d'autant plus si les nœuds de stockage se situent dans des grappes différentes. Pour pallier ce problème, nous proposons une architecture parallèle comme montrée en figure 6(b). Dans cette approche, chaque fournisseur JuxMem est associé à un nœud de stockage du système de fichiers Gfarm choisi dans la même grappe. Cette association est effectuée en essayant de minimiser la latence des communications. Lorsqu'une intéraction entre JuxMem et Gfarm est requise, que ce soit pour une écriture ou un recouvrement, tous les fournisseurs appartenant au groupe global de la donnée concernée écrivent ou lisent, en parallèle, sur leur nœud Gfarm associé. Cette stratégie est mise en place pour réduire les communications inter-grappes et ainsi diminuer le surcoût lié à l'utilisation du stockage secondaire. Avec cette approche, chaque écriture déclenche chez les fournisseurs la création d'un fichier de même nom et de même contenu sur plusieurs nœuds de stockage. Une version modifiée de Gfarm permet d'enregistrer ces copies comme appartenant au même fichier logique Gfarm. Cette approche est actuellement en cours d'implémentation

5. Étude de faisabilité et évaluation préliminaire

Afin de valider notre approche, nous avons réalisé un prototype permettant à JuxMem d'écrire dans Gfarm, sans parallélisme, à chaque modification de la donnée. Un client JuxMem écrivain alloue de l'espace pour stocker la donnée sur un fournisseur JuxMem, sans réplication. L'identifiant de la donnée est transmis à un autre client en charge de la lecture. L'écrivain effectue vingt écritures et le lecteur vingt lectures. Ces deux types d'accès potentiellement concurrents sont synchronisés grâce à l'utilisation des primitives d'acquisition et de relâchement du verrou propres au modèle de cohérence à l'entrée. Chacune des écritures est répercutée, par le fournisseur, dans le système de fichiers Gfarm. Un nouveau fichier est créé pour chacune des écritures. Il n'est ni fragmenté, ni répliqué.

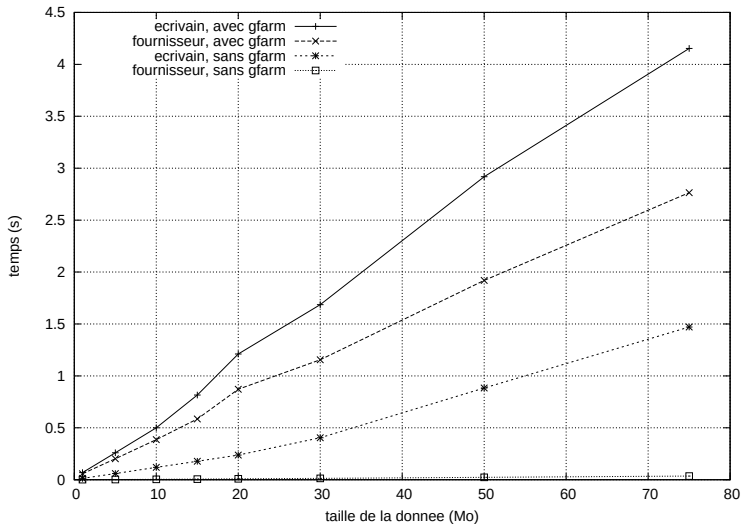

FIG. 7 – Évaluation du surcoût induit par le stockage secondaire

Pour exécuter le scénario décrit précédemment nous avons déployé une instance de Gfarm composée d'un serveur de méta-données, d'un serveur de cache pour les méta-données et d'un nœud de stockage. À ceci se rajoute une instance de JuxMem composée d'un client écrivain, d'un client lecteur, d'un gestionnaire de groupe *cluster* et d'un fournisseur. Afin de mettre en évidence les coûts des transferts réseaux dans l'évaluation des performances, les sept rôles sont placés sur des nœuds physiques différents. Le déploiement de l'ensemble des rôles s'effectue de manière automatique grâce à l'utilisation de l'outil de déploiement générique Adage [1]. Les expérimentations ont été réalisées sur la plate-forme Grid'5000. Cette grille expérimentale hautement reconfigurable est composée de neuf sites en France avec pour objectif de réunir jusqu'à 5000 processeurs dans un proche avenir. Les sites sont interconnectés par un réseau Ethernet 10 gigabits. La grappe utilisée, située à Rennes, est composée de nœuds Intel Xeon 5148 LV cadencés à 2.3 GHz, équipés de 4 Go de mémoire physique et interconnectés par un réseau éthernet gigabit.

Afin d'évaluer le surcoût introduit par l'utilisation de Gfarm, le temps moyen requis par une écriture a été mesuré sur le client JuxMem ainsi que sur le fournisseur. La mesure du temps sur le client comprend la demande du verrou en écriture, l'écriture en mémoire locale puis le relâchement du verrou. C'est pendant cette dernière étape que la donnée modifiée est transmise au fournisseur. La mesure du temps sur le fournisseur comprend l'écriture en mémoire physique ainsi que l'écriture dans Gfarm.

La figure 7 montre le temps moyen requis pour une écriture sur le client et le fournisseur en fonction de la taille de la donnée. Les expériences sont menées avec et sans écriture dans Gfarm. La courbe de l'écrivain sans Gfarm permet d'apprécier le débit moyen de JuxMem (55 Mo/s) en présence de lectures concurrentes [1]. Le temps mesuré sur le fournisseur est négligeable car uniquement composé d'une écriture en mémoire locale. La différence constatée entre les deux courbes fournisseur est directement liée à l'écriture dans Gfarm (25 Mo/s). Cette différence impacte l'écrivain, malgré le fait que l'acquittement pour le relâchement du verrou soit envoyé avant l'écriture dans Gfarm. En effet, après la réception de l'acquittement, le client tente de reprendre le verrou pendant que l'écriture précédente s'effectue dans Gfarm, et se retrouve bloqué.

Dans une phase de calcul où les accès en écriture sont fréquents, ce surcoût peut engendrer un point de contention. Or, il peut s'avérer que cette phase de calcul soit suffisamment courte, avec une probabilité de défaillance acceptable pour ne pas nécessiter de recopie dans Gfarm à chaque écriture. La

[1] Le débit moyen mesuré en écriture, sans concurrence d'accès, est de 89 Mo/s

fréquence des écritures dans Gfarm doit alors être paramétrée comme un compromis entre la fiabilité et la performance.

6. Conclusion

Ce papier explore la possibilité d'augmenter la capacité de stockage d'un service de partage de données pour la grille en lui couplant un sytème de fichiers distribué. Ce stockage secondaire sur disque offre de plus des propriétés de persistance des données. Les accès au système de fichiers sont rendus transparents pour l'utilisateur qui conserve tous les avantages du service de partage de données en terme de simplicité d'utilisation. Afin de valider cette approche, un prototype a été réalisé en utilisant JuxMem et Gfarm. L'évaluation préliminaire de ce prototype montre qu'il existe un surcoût lié à l'utilisation du stockage secondaire mais que celui-ci est proportionnel à la taille de la donnée. Ce coût peut néanmoins être diminué grâce à une gestion fine de la fréquence des écritures dans Gfarm.
Nos travaux actuels visent à évaluer un deuxième prototype tirant partie des lectures et écritures parallèles offertes par le système de fichiers. Cette deuxième version doit pouvoir être comparée avec les résultats déjà obtenus. Des expériences à plus grande échelle, mettant en jeu de la réplication et un nombre important de données de grande taille sont par ailleurs en cours de préparation.

Bibliographie

1. ADAGE. – http ://adage.gforge.inria.fr.
2. The JXTA (juxtapose) project. http ://www.jxta.org.
3. Allcock (B.), Bester (J.), Bresnahan (J.), Chervenak (A. L.), Foster (I.), Kesselman (C.), Meder (S.), Nefedova (V.), Quesnel (D.) et Tuecke (S.). – Data management and transfer in high-performance computational grid environments. *Parallel Comput.*, vol. 28, n5, 2002, pp. 749–771.
4. Amza (C.), Cox (A. L.), Dwarkadas (S.), Keleher (P.), Lu (H.), Rajamony (R.), Yu (W.) et Zwaenepoel (W.). – TreadMarks : Shared Memory Computing on Networks of Workstations. *IEEE Computer*, vol. 29, n2, février 1996, pp. 18–28.
5. Antoniu (G.), Bougé (L.) et Jan (M.). – JuxMem : An adaptive supportive platform for data sharing on the grid. *Scalable Computing : Practice and Experience*, vol. 6, n3, novembre 2005, pp. 45–55. – Extended version to appear in Kluwer Journal of Supercomputing.
6. Antoniu (G.), Deverge (J.-F.) et Monnet (S.). – How to bring together fault tolerance and data consistency to enable grid data sharing. *Concurrency and Computation : Practice and Experience*, vol. 18, n13, November 2006, pp. 1705–1723.
7. Baru (C.), Moore (R.), Rajasekar (A.) et Wan (M.). – The sdsc storage resource broker, 1998.
8. Bassi (A.), Beck (M.), Fagg (G.), Moore (T.), Plank (J. S.), Swany (M.) et Wolski (R.). – The internet backplane protocol : A study in resource sharing. *In : CCGRID '02 : Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid*. p. 194. – Washington, DC, USA, 2002.
9. Bershad (B. N.), Zekauskas (M. J.) et Sawdon (W. A.). – The Midway distributed shared memory system. *In : Proceedings of the 38th IEEE International Computer Conference (COMPCON Spring '93)*, pp. 528–537. – Los Alamitos, CA, février 1993.
10. Foster (I.) et Kesselman (C.). – Globus : A metacomputing infrastructure toolkit. *The International Journal of Supercomputer Applications and High Performance Computing*, vol. 11, n2, Summer 1997, pp. 115–128.
11. Honeyman (P.), Adamson (W. A.) et McKee (S.). – GridNFS : global storage for global collaborations. *In : Proceedings of the IEEE International Symposium Global Data Interoperability - Challenges and Technologies*. pp. 111–115. – Sardinia, Italy, juin 2005.
12. Kosar (T.) et Livny (M.). – Stork : Making data placement a first class citizen in the grid, 2004.
13. Li (K.). – IVY : a shared virtual memory system for parallel computing. *In : Proceedings of the 1988 International Conference on Parallel Processing*. pp. 94–101. – University Park, PA, USA, août 1988.
14. MONARC. – Models of networked analysis at regional centres for lhc experiments : Phase 2 report, 2000.
15. Tatebe (O.), Morita (Y.), Matsuoka (S.), Soda (N.) et Sekiguchi (S.). – Grid datafarm architecture for petascale data intensive computing. *In : CCGRID '02 : Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid*. p. 102. – Washington, DC, USA, 2002.
16. White (B. S.), Walker (M.), Humphrey (M.) et Grimshaw (A. S.). – LegionFS : a secure and scalable file system supporting cross-domain high-performance applications. *In : Proceedings of the 2001 ACM/IEEE Conference on Supercomputing (SC '01)*. pp. 59–59. – New York, NY, USA, 2001.