Généralisation du concept "big.LITTLE" pour aller vers des infrastructures cloud énergétiquement proportionnelles

Violaine Villebonnet, Georges da Costa, Laurent Lefevre, Jean-Marc Pierson, Patricia Stolf

Compas'2015 : Parallélisme/ Architecture / Système
Lille, France, du 30 juin au 3 juillet 2015

Généralisation du concept « big.LITTLE » pour aller vers des infrastructures cloud énergétiquement proportionnelles

Violaine Villebonnet[1,2], Georges Da Costa[1], Laurent Lefevre[2], Jean-Marc Pierson[1], and Patricia Stolf[1]

[1]IRIT, Université de Toulouse, France
[2]Inria Equipe Avalon - LIP - Ecole Normale Supérieure de Lyon, Université de Lyon, France

Résumé
Les infrastructures de type cloud sont souvent surdimensionnées par rapport aux besoins et les serveurs sont rarement utilisés à leur capacité maximale. Seulement, leur consommation énergétique statique est importante. Pour que l'énergie consommée par ces infrastructures soit proportionnelle à la charge, il faut adapter l'architecture des serveurs en fonction des besoins des applications. Nous proposons une infrastructure composée de machines hétérogènes, ainsi qu'un système de placement dynamique des applications et une gestion intelligente d'allumage et d'extinction des ressources physiques. Notre but est d'exécuter les applications sur les machines les plus adaptées afin d'avoir une consommation énergétique en relation avec l'utilisation.

1. Introduction

A l'échelle mondiale, la consommation des datacenters était en 2012 de 270 TWh, ce qui correspond à environ 2% de la consommation énergétique mondiale [8]. Cette importante consommation d'électricité représente non seulement une limitation économique mais a surtout un impact non négligeable sur l'environnement. L'année 2014, qui a été la plus chaude jamais observée, nous prouve qu'il est crucial de faire des efforts pour limiter la consommation de ces infrastructures, dont le nombre est en constante augmentation. La conception des datacenters est loin d'être parfaite car l'électricité consommée n'est pas intégralement utilisée par les serveurs. L'indicateur PUE (Power Usage Effectiveness), introduite par l'organisation Green Grid en 2007, permet de montrer cela. Le PUE est une métrique qui mesure la part d'énergie consommée par le datacenter global par rapport à l'énergie effectivement consommée par les systèmes informatiques. On se rend ainsi compte des surcoûts de consommation causés par le système de refroidissement, l'alimentation électrique, l'éclairage, etc... Cependant, le PUE ne permet pas d'évaluer l'efficacité des systèmes informatiques. Au sein d'un datacenter, les serveurs sont la plupart du temps toujours allumés même s'ils ne sont pas utilisés. Dans ce cas, l'électricité est effectivement consommée par les ressources informatiques, mais elle est gâchée car elle n'est pas transformée en puissance de calcul. En effet, la consommation des serveurs n'est pas idéale : Quand un serveur est allumé mais sans activité, sa consommation électrique est déjà importante. Elle peut parfois atteindre la moitié de la consommation à pleine puissance. Or dans un datacenter, les serveurs sont rarement utilisés à leur puissance maximale. Ce problème a été mis en valeur par Barroso et Holzle en 2007 [5]. Ils ont montré que la charge moyenne des machines au sein d'un datacenter de Google était entre 10 et 50%. Or les machines sont conçues pour avoir une grande efficacité énergétique principalement à pleine charge. Une si faible utilisation correspond donc à la région où les serveurs sont les moins efficaces. Dans nos travaux, nous voulons réduire ces coûts statiques, c'est à dire viser une courbe de consommation qui commencerait à zéro et serait la plus proportionnelle possible. Une telle architecture, que l'on qualifie d'« énergétiquement proportionnelle »permettrait d'importantes économies d'énergie.
Notre contribution consiste à proposer une infrastructure originale, composée de ressources de calcul hétérogènes en termes de performances et de consommation d'énergie. Afin d'atteindre la proportionnalité énergétique, nous proposons d'exécuter dynamiquement les applications sur la ressource, ou com-

binaison de ressources, la moins consommatrice en énergie tout en satisfaisant les performances demandées. Le fait d'adapter le matériel aux demandes des applications permet d'avoir une consommation qui évolue en même temps que la charge. Notre idée est d'être capable d'utiliser uniquement les ressources les plus adaptées, pendant que les autres sont éteintes, ou dans un mode veille à faible consommation. Les mécanismes-clés utilisés sont la migration d'applications entre ressources hétérogènes, ainsi que l'extinction et rallumage dynamique des machines.
Cet article commence par un court état de l'art sur la proportionnalité énergétique et sur le concept d'architectures hétérogènes. Nous présentons ensuite notre infrastructure matérielle ainsi que le système qui va prendre les décisions de placement des applications et de gestion des ressources. Nous détaillons les expériences de profilage et la validation de notre concept par simulations, qui ont pour cas d'utilisation les serveurs web sans état. Enfin, nous commentons les premiers résultats, ainsi que les futures directions de nos travaux.

2. Etat de l'art

Les économies d'énergie dans les infrastructures de types cloud et HPC font partie d'un domaine de recherche assez récent [10]. De nombreux travaux ont été menés dans le but d'optimiser l'utilisation des ressources dans les datacenters. L'approche majeure est la consolidation, qui consiste à concentrer le plus de tâches possibles sur un nombre restreint de machines physiques, afin de pouvoir éteindre les machines inutilisées. Cela a été rendu possible grâce à la technologie de virtualisation, qui permet de faire cohabiter plusieurs systèmes d'exploitation sur une même machine physique. La migration à chaud est l'autre mécanisme-clé car elle permet de déplacer les machines virtuelles d'une machine physique à une autre sans avoir d'impact sur les applications en cours d'exécution. La consolidation n'est pas toujours facile à mettre en œuvre car les actions d'allumage et d'extinction représentent un surcoût en temps et en énergie, et doivent être utilisées judicieusement pour être pertinentes. La plupart des techniques de consolidation sont fondées sur des heuristiques qui sont souvent des variantes de l'algorithme « bin-packing », mais d'autres méthodes ont aussi fait leurs preuves comme la programmation par contraintes [9], les algorithmes génétiques, ou même une méta-heuristique inspirée des colonies de fourmis [7].
Le DVFS, « Dynamic Voltage and Frequency Scaling », permet de diminuer la consommation énergétique au niveau des serveurs. Son principe consiste à adapter la fréquence du processeur en fonction des besoins en calcul, et donc de réduire la consommation lorsqu'une haute fréquence n'est pas nécessaire. Cependant, tout comme l'extinction/allumage, cette technique n'apporte des gains que si elle est utilisée à bon escient. Pour cela, il est nécessaire d'avoir de bonnes connaissances sur l'évolution de la charge de travail. Dans [13], les auteurs proposent d'analyser les compteurs de performances pour acquérir le profil de charge des applications et ensuite prédire leurs évolutions. Plus les prédictions sont précises et plus les gains en énergie pourront être importants. Ces différentes approches montrent des limitations car elles permettent uniquement de réduire l'énergie de manière globale mais ne règlent pas le problème des coûts statiques élevés des serveurs.
Dans nos travaux, nous voulons proposer une solution qui réduise au maximum ces coûts statiques. Notre but, la proportionnalité énergétique, consiste à se rapprocher d'une consommation énergétique nulle lorsque la charge l'est aussi, et de suivre une courbe de consommation proportionnelle à la charge ensuite. Des travaux [14] [12] ont proposé des métriques afin d'évaluer la proportionnalité d'une architecture. Dans [12], une combinaison de deux métriques est proposée : la première mesure l'intervalle entre la consommation pour une charge nulle et la consommation à plein charge, tandis que la deuxième évalue la linéarité de la courbe de consommation. Ils appliquent ces métriques sur des architectures existantes et montrent l'évolution du matériel au cours des dernières années. On constate que les serveurs sont globalement de plus en plus proportionnels : l'intervalle des consommations est de plus en plus grand, mais la linéarité est souvent dégradée.
Comme une architecture purement proportionnelle n'existe pas encore, une suggestion pour se rapprocher de la proportionnalité est d'utiliser un ensemble de plusieurs architectures disposant de différentes caractéristiques de consommation et de performances : C'est l'idée d'hétérogénéité. On la retrouve dans les processeurs multi-cœurs hétérogènes. Plusieurs constructeurs ont proposé leur implémentation de ce concept comme ARM avec big.LITTLE [2] qui combine un processeur très basse performance avec un

plus puissant, ou Nvidia avec le processeur Tegra K1 [3] qui associe un processeur ARM à des accélérateurs GPU. Dans ces architectures, les applications sont exécutées sur le processeur qui correspond le plus à leurs besoins. Nous avons choisi de nous focaliser sur le processeur ARM big.LITTLE car il a été notre inspiration pour nos travaux. Ce processeur multi-cœur est composé du processeur Cortex-A15, qui joue le rôle du processeur « big »et du Cortex-A7 qui est le « LITTLE ». La particularité de cette architecture réside dans le module de cohérence cache qui interconnecte les deux processeurs. Cela permet de facilement déplacer les tâches entre les processeurs en conservant toute cohérence mémoire.
Les processeurs hétérogènes constituent une approche innovante, mais ils sont pour le moment dédiés aux dispositifs mobiles comme les tablettes et smartphones. L'idée derrière ce concept est d'économiser la batterie de tels dispositifs lorsqu'ils sont sans activité, tout en fournissant d'excellentes performances quand c'est nécessaire. Nous voulons étendre cette approche à l'échelle d'un datacenter. Les premiers aspects de cette idée ont été explorés dans [6]. Ces travaux montrent les potentiels gains d'utiliser un ensemble de ressources hétérogènes composés de Raspberry Pi, Intel Atom et Intel i7 pour héberger des serveurs web sans état. La courbe de consommation d'énergie se rapproche de la courbe proportionnelle particulièrement pour les faibles charges. Notre objectif est de poursuivre cette proposition et d'étudier des aspects supplémentaires comme la gestion des ressources et les surcoûts associés.

3. Infrastructure BML : Big,Medium,Little

Notre approche va au delà du concept d'ARM big.LITTLE en le généralisant à l'échelle d'un datacenter. Nous voulons exploiter les processeurs basse consommation quand la charge est faible et utiliser les serveurs classiques pour leur performance. Dans nos travaux nous ajoutons plus de flexibilité et plus d'hétérogénéité en utilisant une plus grande variété de processeurs, comme nous l'avons détaillé dans [15]. De plus, notre approche s'éloigne du concept de base en divers points. En effet, big.LITTLE est un processeur multi-cœur qui combine 2 processeurs différents, mais ils sont tous les 2 fondés sur la même architecture ARMv7-A et le même jeu d'instructions. L'hétérogénéité dans ce cas est relative car la différence entre les 2 processeurs se voit uniquement en terme de puissance de calcul et de consommation énergétique. Pour généraliser ce concept aux infrastructures à grande échelle, nous avons élargi la gamme de processeurs, c'est pourquoi l'hétérogénéité dans notre cas va jusqu'au jeu d'instructions.

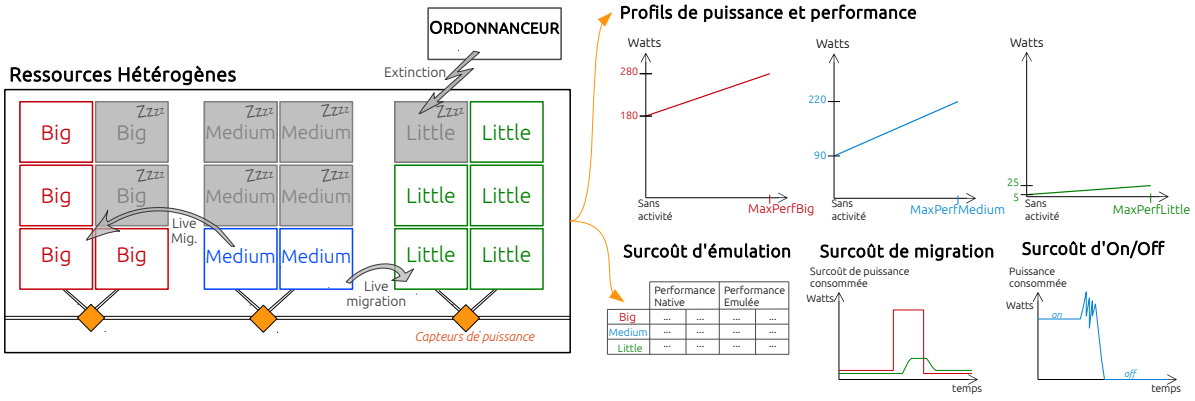


FIGURE 1 – Infrastructure composée de ressources hétérogènes avec informations de profilage collectées pour calibration du système

L'infrastructure que nous proposons est décrite sur la FIGURE 1. Dans cet exemple, nous considérons 3 différents types de machines que nous avons nommés 'Little', 'Medium' et 'Big' pour rester dans le même esprit qu'ARM big.LITTLE. Bien sûr, nous ne considérons pas que le nombre optimal d'architectures soit forcément trois, l'important étant la pertinence des architectures choisies et pas leur nombre. Nous imaginons une infrastructure composée de plusieurs nœuds de chaque type que l'ordonnanceur peut contrôler et accéder de manière totalement indépendante. Notre but est de toujours exécuter les

applications sur l'architecture ou combinaison d'architectures la plus adaptée. L'architecture la plus adaptée est définie comme celle qui consomme le moins pour les performances demandées par l'application. Evidemment, les exigences de performances d'une application peuvent varier au cours du temps donc le système doit être capable de transférer l'exécution d'une application vers une autre architecture plus adaptée si besoin. Par exemple si la charge CPU demandée décroit, l'application doit être migrée vers une architecture moins performante et donc moins consommatrice en énergie, mais si la charge augmente, l'application doit être déplacée vers une architecture plus puissante afin de satisfaire ses besoins en calcul et ne pas dégrader sa qualité d'exécution. Ce mécanisme est représenté par les flèches 'migration' sur la figure. Ces migrations à chaud peuvent avoir lieu entre n'importe quelles ressources de notre infrastructure. Quand les machines sont inutilisées, elles sont éteintes ou mises dans un mode veille. Sur le schéma, les nœuds inutilisés sont représentés en gris avec le signe 'Zzzz' pour montrer leur indisponibilité. Nous considérons que la consommation d'une machine indisponible est négligeable.
Une étape primordiale pour pouvoir utiliser cette infrastructure dans l'optique de proportionnalité énergétique est le profilage des machines. Sur la partie droite de la FIGURE 1, nous avons représenté chaque aspect de l'infrastructure que nous profilons. La consommation des ressources de calcul doit être parfaitement connue, c'est pourquoi les machines de l'infrastructure sont équipées d'outils de mesurage. Les profils énergétiques, couplés avec des profils de performances applicatives forment un profil complet de la machine. Le comportement des serveurs varie en fonction des types d'applications qu'ils exécutent, ainsi plusieurs profils peuvent être créés. Ces profils sont les clés de notre ordonnanceur car ils caractérisent les ressources et ils servent de bases aux décisions de placement des applications. En plus des profils de performance et de consommation, d'autres aspects doivent être analysés comme le surcoût d'émulation, de migration et d'allumage et extinction. La connaissance de ces paramètres permettra de prendre des décisions de placement plus efficaces.

4. Système d'ordonnancement BML

Dans cette partie nous décrivons les différents blocs qui composent notre système global BML :

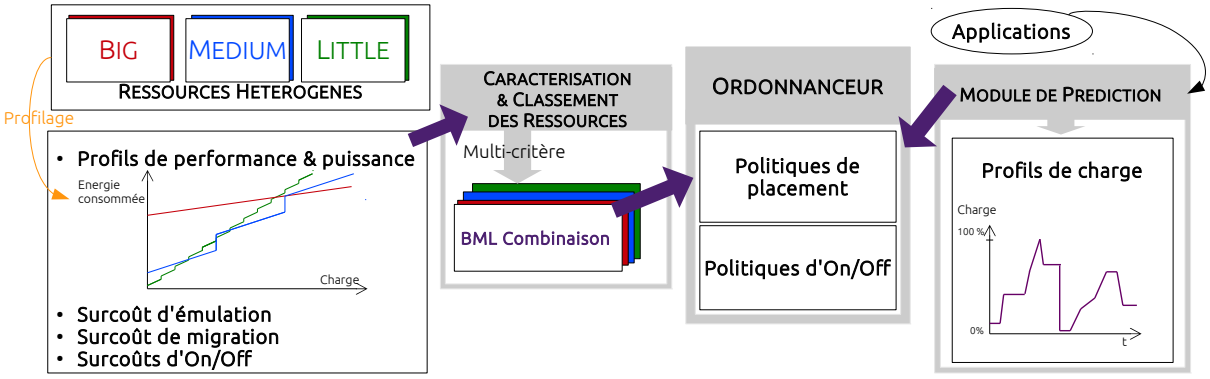


FIGURE 2 – Architecture du système d'ordonnancement BML

Sur la gauche de la FIGURE 2 on retrouve les profils de puissance et de performance qui ont été introduits dans la partie précédente et dont l'acquisition est détaillée dans la partie 5.2. On connaît donc la consommation de chaque type d'architecture pour chaque niveau de performance. Ainsi pour un niveau de performance donné on peut trouver l'architecture ou la combinaison d'architectures qui consommera le moins. En plus des décisions de placement, le système doit prendre les décisions concernant la gestion d'allumage et d'extinction des machines. Mais pour prendre des décisions intelligentes il est nécessaire d'avoir non seulement des informations sur l'infrastructure matérielle mais aussi sur les applications à exécuter et leurs profils de charge au cours du temps. Nous allons détailler les autres blocs de notre système qui sont la caractérisation et le classement des ressources, le module de prédiction de charge et

enfin l'ordonnanceur.

4.1. Caractérisation et classement des ressources

Le but de ce module est de caractériser et classer les différents types d'architectures en se basant sur leurs profils de performance et de consommation énergétique. On peut éventuellement prendre en compte dans le classement d'autres paramètres comme les surcoûts d'allumage et d'extinction. Ce classement est spécifique à l'application choisie comme cas d'étude. Les performances de chaque architecture ne sont pas les mêmes pour toutes les applications et le classement des architectures les unes par rapport aux autres peut différer en fonction de l'application choisie. Une fois que chaque architecture a été profilée individuellement, et classée par rapport aux autres, le but est de déterminer la combinaison d'architectures idéale.

4.2. Module de prédiction de charge

Le graphe sur la droite de la FIGURE 2 représente le profil de charge de l'application. Ce profil décrit l'évolution des besoins de l'application au cours du temps. Il est nécessaire de connaître cette information car on veut adapter le placement de l'application en fonction des ressources demandées pour qu'elle s'exécute toujours sur la machine la plus appropriée. La difficulté est de pouvoir acquérir de tels profils. On peut imaginer avoir des informations à priori sur certaines applications, mais ce n'est pas toujours le cas. Dans une situation plus générale, quand l'application n'est pas connue à l'avance, le profil de charge est plus compliqué à construire et il est nécessaire d'avoir un système de prédictions. Un tel système va analyser le comportement de l'application en regardant les ressources qu'elle utilise et en fonction des observations passées, le mécanisme prédit les ressources demandées dans le futur.

4.3. Ordonnanceur

L'ordonnanceur et ses politiques associées sont le cœur de notre système. L'ensemble des profils que nous venons de détailler représentent les paramètres d'entrée des politiques de placement. Tout au long de l'exécution d'une application, en fonction des besoins futurs prédits, l'ordonnanceur va choisir la localisation de l'exécution en s'assurant que les performances demandées sont satisfaites, mais aussi qu'il n'y a pas d'énergie consommée inutilement. Pour cela, il va activer les leviers de migration d'applications ainsi que d'extinction et d'allumage des machines. Les politiques d'extinction et d'allumage ont une grande importance. Les ressources doivent rapidement être éteintes ou mises en veille lorsqu'elles sont inutilisées pour éviter de gaspiller de l'énergie, mais elles doivent également être rapidement disponibles. Comme les actions d'allumage et d'extinction peuvent être longues et consommatrices en énergie, la qualité de prédiction de la charge va permettre de prendre de meilleures décisions qui auront le moins d'impact possible sur la qualité d'exécution de l'application. Plusieurs politiques peuvent être implémentées et il est nécessaire d'étudier lesquelles apportent les gains les plus importants.

5. Expérimentations

5.1. Plateforme expérimentale

La TABLE 1 rassemble la description des machines que nous avons choisies pour nos expériences de validation. Nous avons choisi le processeur ARM Cortex-A15 comme processeur faible consommation, que nous avons trouvé dans le Samsung Chromebook. Comme son nom l'indique ce notebook est équipé du système d'exploitation Chrome OS de Google, mais nous avons installé une distribution Linux pour nos expériences. Nous mesurons sa consommation électrique avec un wattmètre externe Watts up Pro. Concernant les serveurs classiques à architecture x86, le choix est plus large. Nous avons utilisé des nœuds de la plateforme expérimentale française Grid'5000 [4]. Nous avons choisi un serveur avec un processeur Intel Xeon et un avec un processeur AMD Opteron, localisés respectivement dans les clusters de Lyon et Rennes. Ces deux machines sont de deux générations différentes c'est pourquoi nous avons trouvé intéressant de les comparer. Leur consommation électrique est mesurée également avec des wattmètres externes : Omegawatt à Lyon et PDU EATON à Rennes. Les données de consommation sont récupérées et accessibles via l'API Kwapi [11]. On observe sur le tableau l'importante différence entre les puissances consommées. En particulier la puissance consommée quand les machines sont sans activité : pour le serveur HP elle est 20 fois plus grande que celle du Chromebook, et 2 fois plus grande que celle du serveur Dell. La borne maximale de l'intervalle de puissance est la consommation quand

tous les cœurs de la machines sont chargés au maximum. Bien sûr la puissance consommée brute n'est pas la seule chose à prendre en considération, et nous allons le voir dans la suite de l'article.

TABLE 1 – Résumé des machines sélectionnées

Nom	Samsung Chromebook	Dell PowerEdge R720	HP Proliant DL165 G7
Architecture	ARMv7 32 bits	x86 Intel 64 bits	x86 AMD 64 bits
CPU	2 x ARM Cortex-A15	2 x Intel Xeon E5-2630	2 x AMD Opteron 6164
Nb cœurs	2	12	24
Puissance	5 – 25 W	90 – 220 W	170 – 280 W
Année de sortie	2012	2012	2013

5.2. Acquisition des profils

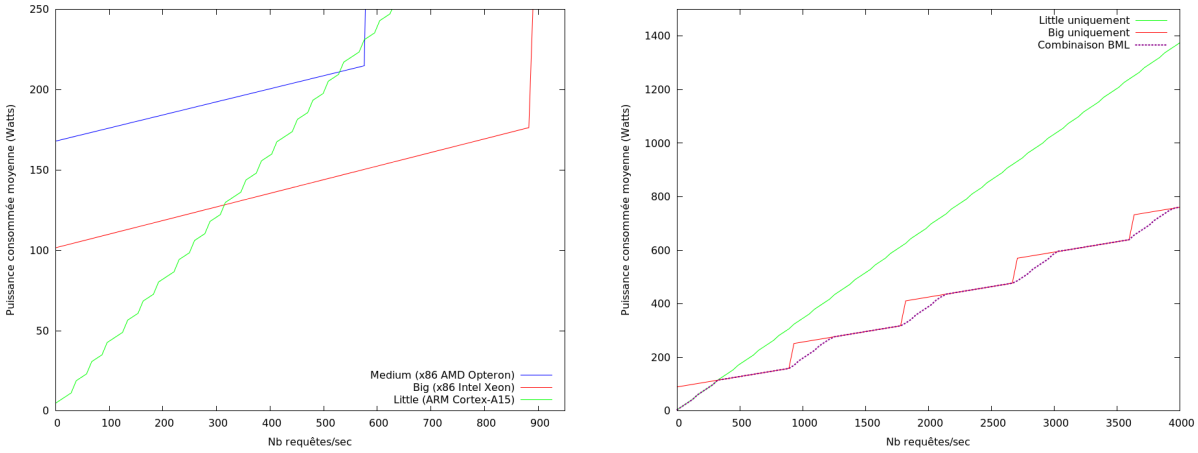
L'acquisition des profils est une étape très importante dans nos travaux. Nous avons besoin de connaître le comportement des applications sur les différentes machines afin de savoir où il est le plus judicieux de les placer. L'idée est de pré-exécuter l'application sur chaque architecture présentée dans la TABLE 1, et de mesurer sa performance et la consommation de la machine. Nous avons choisi un serveur web comme cas d'application. Ce type de service présente une forte variabilité en terme de demande, ce qui signifie qu'il y a des efforts à faire pour faire correspondre le matériel à l'évolution de la demande. Le challenge dans la gestion de serveurs web est d'être capable de répondre à toutes les requêtes avec une latence acceptable mais sans trop allumer de machines pour ne pas gâcher d'énergie. Pour nos expériences, nous avons choisi le serveur web lighttpd. La page web interrogée est un script cgi écrit en python qui retourne aléatoirement une image parmi 5, de taille entre 1 et 3 ko. Pour générer la charge nous avons utilisé Siege [1] qui est un outil de benchmarking web. Il génère des clients et des requêtes et renvoie des informations comme le nombre de requêtes par seconde effectivement traitées, le temps de réponse, la taille des données renvoyées, etc... Notre objectif est de trouver le nombre maximal de requêtes que peut traiter chaque architecture, ainsi que la puissance consommée pour faire ce traitement. Le serveur web tourne sur un coeur de la machine et l'outil qui génère les requêtes est lancé depuis une autre machine pour ne pas influer sur les mesures. Les résultats sont présentés dans la TABLE 2.

TABLE 2 – Performance maximale et puissance consommée moyenne pour chaque architecture

Processeur	NbReqs/s max	Watts max	Watts min
x86 Intel Xeon (Big)	888,74	175,25	93.61
x86 AMD Opteron (Medium)	583,35	221,16	172
ARM Cortex-A15 (Little)	31,54	11,96	5,5

Une fois ces profils acquis, nous les interpolons pour mieux les analyser. Sur la FIGURE 3(a), nous avons représenté la consommation de chaque architecture en fonction du nombre de requêtes traitées. Nous avons interpolé le profil du processeur 'Little' basse consommation en cumulant la performance et la consommation de plusieurs d'entre eux. Nous pouvons constater graphiquement que jusqu'à environ 340 requêtes par seconde, il est plus intéressant du point de vue énergie d'utiliser un ensemble de processeurs ARM plutôt qu'un seul processeur x86. La seconde chose à constater est le fait que le profil du processeur AMD n'est pas intéressant dans ce cas ci. En effet, il est moins performant que le processeur Intel mais sa consommation est la plus importante de toutes les architectures.

Avec ces profils nous pouvons savoir quelle est l'architecture, ou ensemble d'architectures la plus intéressante à choisir pour chaque niveau de performance. Nous appelons cet ensemble la combinaison BML, pour « Big, Medium, Little ». Le profil de la combinaison BML idéale est représenté en pointillés



(a) Comparaison entre 3 architectures différentes

(b) Création de la combinaison BML idéale

FIGURE 3 – Profils de puissance et de performance d'un serveur web lighttpd

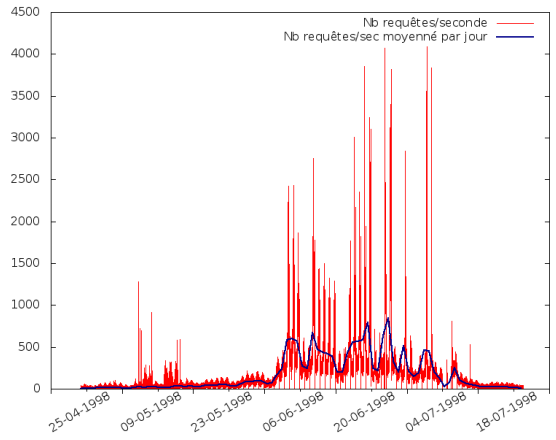
violets sur la FIGURE 3(b). Cette combinaison mélange uniquement les architectures les plus intéressantes des points de vue performance et énergie. Dans notre exemple seulement 2 architectures ont été conservés : Intel Xeon ('Big') et ARM Cortex-A15 ('Little). Jusqu'à 340 requêtes par seconde, il est plus pertinent d'utiliser une combinaison de processeurs 'Little' uniquement. A partir de ce point là, un seul processeur 'Big' devient plus intéressant, et ce jusqu'à sa performance maximale. Quand la demande est juste un peu au dessus du maximum d'une machine 'Big', il n'est alors pas judicieux d'allumer un deuxième 'Big', mais plutôt d'allumer un ou plusieurs 'Little'. Ici seulement la puissance instantanée moyenne est considérée et nous ne prenons pas en compte l'allumage des différentes machines.

6. Validations

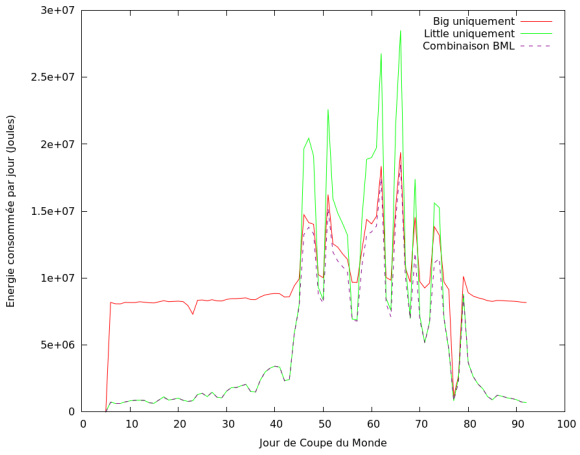
6.1. Scénario de simulations

Afin de valider notre proposition et d'évaluer les gains potentiels de la combinaison BML à grande échelle, nous avons développé un simulateur. Il se base sur les profils de puissance et de performance que nous venons de décrire. La simulation nous permet d'évaluer plus facilement et plus rapidement notre solution sur différents scénarios, et de tester différentes politiques de placement et de gestion des ressources. Techniquement, nous avons développé notre simulateur en langage Python, il prend en entrée les profils décrits dans la TABLE 2. En sortie, il calcule l'énergie consommée au cours du temps pour un certain scénario, ainsi que la composition du datacenter, et le nombre d'actions d'allumage et d'extinction. De cette façon, nous pouvons savoir précisément d'où vient la consommation d'énergie et sur quels points nous devons améliorer notre solution. Nous avons choisi l'application de service web car elle offre une grande variabilité de charge. Elle nous permet aussi de facilement tester différents scénarios en changeant les traces en entrée du simulateur. Dans cet article, nous présentons les résultats de simulation pour les traces d'accès au site de la Coupe du Monde de Football de 1998. (Disponibles ici : `http://ita.ee.lbl.gov/html/contrib/WorldCup.html`). Elles ont été collectées sur une période de 4 mois entre Avril et Juillet 1998. Le nombre total de requêtes reçues dépasse le milliard.
Sur la FIGURE 4(a) est représentée la répartition des requêtes au cours du temps. Nous constatons l'importante variabilité par la différence entre le nombre de requêtes par seconde (en rouge), et le taux moyen de requêtes par jour (en bleu). Par exemple le pic le plus important qui a été d'environ 4000 requêtes par seconde le 23 juin correspond à une moyenne sur la journée d'environ 900 requêtes par seconde. Notre simulateur calcule le nombre de machines nécessaires pour répondre à la totalité des requêtes entrantes avec un temps de réponse acceptable, c'est à dire en ne dépassant jamais la performance maximale de chaque architecture. La consommation énergétique des machines utilisées pour répondre aux requêtes est cumulée, et toutes les autres machines sont considérées inutilisées. Dans les résultats

suivants, nous considérons une quantité illimitée de machines de chaque architecture. Nous détaillons ensuite les différentes variations de scénarios et de paramètres pris en compte.



(a) Variabilité de la distribution des requêtes d'accès au site de la Coupe du Monde 98



(b) Energie consommée cumulée par jour pour 3 configurations différentes

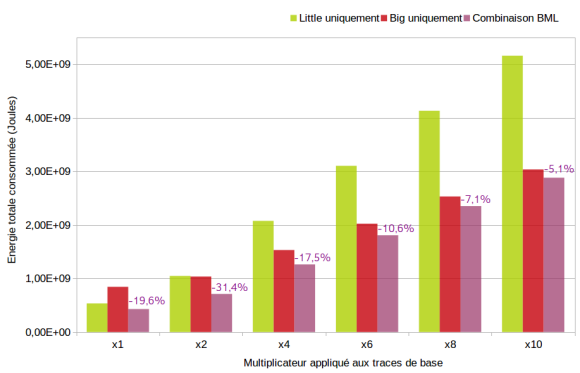
FIGURE 4 – Relation entre charge du serveur web et consommation énergétique des machines

6.2. Gains de la combinaison BML

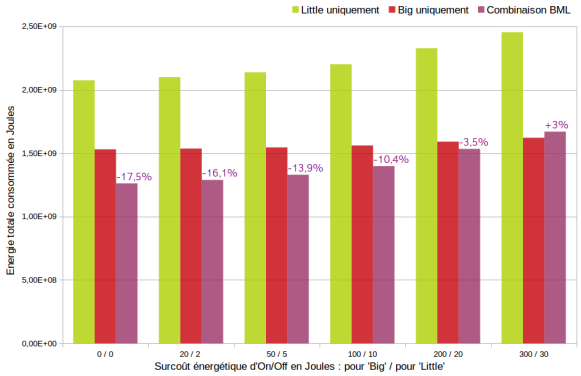
Dans un premier temps, nous avons comparé les deux configurations homogènes du datacenter ('Big' ou 'Little' uniquement) à la combinaison BML qui mélange les deux architectures de manière idéale. Ici les surcoûts d'allumage/extinction des machines ne sont pas pris en compte. De plus, le nombre de requêtes arrivantes est connu à l'avance. Nous sommes dans un cas idéal de prédiction parfaite et d'allumage/extinction instantanés. Sur la FIGURE 4(b) est représentée la consommation d'énergie cumulée pour chaque jour pour les 3 configurations. Nous retrouvons un résultat similaire à celui des profils d'architecture : Quand la charge est faible, pendant les 40 premiers jours et les 10 derniers, il est plus intéressant d'utiliser une combinaison de processeurs 'Little' uniquement. Au contraire pendant la période de Coupe du Monde, on se rapproche de la configuration 'Big' uniquement. Comme les résultats obtenus dépendent beaucoup du nombre absolu de requêtes par seconde, nous avons effectué d'autres simulations en faisant varier les traces d'entrée. Nous avons appliqué un multiplicateur aux traces de base pour voir l'impact sur les gains relatifs de la combinaison BML par rapport aux configurations homogènes. Ces résultats sont présentés sur la FIGURE 5(a). Le pourcentage représente à chaque fois les gains de la combinaison BML par rapport à la configuration homogène la moins consommatrice. C'est uniquement pour les traces de base (multiplicateur x1) que la configuration 'Little' est la moins consommatrice, pour toutes les autres les pourcentages sont calculés par rapport à la configuration 'Big' uniquement. On voit sur la figure que plus le nombre absolu de requêtes est grand et moins les gains de la combinaison BML sont importants. Dans ces cas là, l'apport de l'architecture 'Little' n'est pas remarquable car l'architecture 'Big' sera majoritairement utilisée comme elle est plus efficace pour traiter des taux de requêtes élevés.

Jusqu'à présent nos résultats ne prennent en compte aucun surcoût, ils représentent les gains maximaux atteignables avec la combinaison BML. Pour se rapprocher de la réalité, nous avons étudié l'influence de l'allumage/extinction des machines sur notre solution. Nous avons donc ajouté les surcoûts d'On/Off sur les résultats précédents. Dans notre situation, la configuration du datacenter est mise à jour chaque seconde en fonction du nombre de requêtes à traiter. Sur la FIGURE 5(b), nous montrons l'impact des surcoûts énergétiques d'On/Off. Nous avons choisi le scénario où les traces sont multipliées par 4 car il représente le cas moyen. Nous avons testé différentes valeurs de surcoût pour voir les limitations

de notre proposition. Par simplicité, nous avons considéré identiques les coûts d'allumage et d'extinction. Comme les architectures 'Big' et 'Little' ont dans notre cas un facteur 10 environ de différence en terme de consommation énergétique, nous avons supposé ce même facteur pour les coûts d'On/Off. Par exemple le cas '100/10' signifie qu'un surcoût de 100 Joules a été considéré pour l'allumage/extinction d'un nœud 'Big', et 10 Joules pour un 'Little'. Nous avons simulé pour les 3 configurations et calculer le pourcentage de gains de la combinaison BML. La configuration homogène 'Big' est toujours la moins consommatrice dans ce cas. En effet, la solution 'Little' est composée d'un nombre plus important de machines et donc beaucoup d'actions d'On/Off sont faites. Pour le cas où les surcoûts d'On/Off sont les plus grands, la combinaison BML consomme plus que la solution 'Big'. Cela nous montre les limites de notre solution et qu'il est important de prendre en compte avec attention les surcoûts d'On/Off. Ici les décisions d'allumage et d'extinction sont prises à chaque seconde ce qui implique un nombre important de changement d'état des machines. Il est donc nécessaire d'explorer des systèmes de décisions moins fréquents et peut être ainsi plus efficaces.



(a) Différentes versions des traces de la Coupe du monde 98



(b) Différents surcoûts d'On/Off pour les nœuds 'Big' et 'Little' en Joules (traces x4)

FIGURE 5 – Energie totale consommée, et gains de la combinaison BML par rapport aux solutions homogènes pour différents scénarios

7. Conclusions et perspectives

Dans ces travaux, nous proposons de généraliser le concept d'hétérogénéité, inspiré du processeur ARM big.LITTLE, à l'échelle d'un datacenter pour viser la proportionnalité énergétique. Notre infrastructure nommée BML pour « Big, Medium, Little »rassemble des machines hétérogènes à plusieurs niveaux : architecture, performance et consommation énergétique. Autour nous construisons un système qui va prendre les décisions de placement dynamique des applications ainsi que d'allumage et d'extinction des machines. Nous avons montré les gains de notre solution dans le cas d'un service web à charge variable à travers des simulations basées sur des profils acquis expérimentalement. Nous voulons maintenant améliorer notre simulateur pour tester et évaluer de nouvelles politiques de placement et d'allumage/extinction. Pour cela il est nécessaire de limiter la topologie du datacenter, et d'intégrer une notion de qualité de service pour mesurer la qualité de chaque politique. Dans un second temps, il sera intéressant d'évaluer l'impact d'une prédiction non parfaite de l'arrivée des requêtes sur notre solution.

Remerciements

Les expériences présentées dans cet article ont été effectuées en utilisant la plateforme Grid'5000, soutenu par un groupement d'intérêt scientifique organisé par Inria, le CNRS, RENATER et plusieurs universités ainsi que d'autres organisations (voir https ://grid5000.fr). Ces travaux sont partiellement soutenus par le projet ANR MOEBUS.

Bibliographie

1. Siege Benchmark. *https ://www.joedog.org/siege-home/*.
2. ARM big.LITTLE Technology : The Future of Mobile. *ARM White Paper*, 2013.
3. NVIDIA Tegra K1 : A New Era in Mobile Computing. *NVIDIA White Paper*, 2014.
4. Balouek (D.), Carpen Amarie (A.), Charrier (G.), Desprez (F.), Jeannot (E.), Jeanvoine (E.), Lèbre (A.), Margery (D.), Niclausse (N.), Nussbaum (L.), Richard (O.), Pérez (C.), Quesnel (F.), Rohr (C.) et Sarzyniec (L.). – Adding Virtualization Capabilities to the Grid'5000 Testbed. *In : Cloud Computing and Services Science*. – Springer, 2013.
5. Barroso (L.) et Holzle (U.). – The Case for Energy-Proportional Computing. *IEEE Computer*, 2007.
6. Da Costa (G.). – Heterogeneity : The Key to Achieve Power-Proportional Computing. *IEEE Int. Symp. on Cluster Computing and the Grid*, 2013.
7. Feller (E.), Rilling (L.) et Morin (C.). – Energy-Aware Ant Colony Based Workload Placement in Clouds. – In *IEEE/ACM Int. Conference on Grid Computing*, 2011.
8. Heddeghem (W.), Lambert (S.), Lannoo (B.), Colle (D.), Pickavet (M.) et Demeester (P.). – Trends in Worldwide ICT Electricity Consumption from 2007 to 2012. *Computer Communications*, 2014.
9. Hermenier (F.), Lorca (X.), Menaud (J.-M.), Muller (G.) et Lawall (J.). – Entropy : A Consolidation Manager for Clusters. – In *ACM Int. Conf. on Virtual Execution Environments*, 2009.
10. Orgerie (A.-C.), Dias De Asuncao (M.) et Lefèvre (L.). – A Survey on Techniques for Improving the Energy Efficiency of Large Scale Distributed Systems. *ACM Computing Surveys*, 2014.
11. Rossigneux (F.), Lefevre (L.), Gelas (J.-P.) et Dias de Assuncao (M.). – A Generic and Extensible Framework for Monitoring Energy Consumption of OpenStack Clouds. – In *IEEE 4th Int. Conf. on Big Data and Cloud Computing, BDCloud, Sydney, Australia*, 2014.
12. Ryckbosch (F.), Polfliet (S.) et Eeckhout (L.). – Trends in Server Energy Proportionality. *Computer*, 2011.
13. Tsafack Chetsa (G.), Lefèvre (L.), Pierson (J.-M.), Stolf (P.) et Da Costa (G.). – Exploiting Performance Counters to Predict and Improve Energy Performance of HPC Systems. *Future Generation Computer Systems*, 2014.
14. Varsamopoulos (G.), Abbasi (Z.) et Gupta (S.). – Trends and Effects of Energy Proportionality on Server Provisioning in Data Centers. – In *Int. Conf. on High Performance Computing*, 2010.
15. Villebonnet (V.), Da Costa (G.), Lefèvre (L.), Pierson (J.-M.) et Stolf (P.). – Towards Generalizing "Big Little" for Energy Proportional HPC and Cloud Infrastructures. – In *IEEE 4th Int. Conf. on Big Data and Cloud Computing, BDCloud, Sydney, Australia*, 2014.