RenPar'19 / SympA'13 / CFSE'7
Toulouse, France, du 9 au 11 septembre 2009

Réduction de la consommation énergétique dans les centres de serveurs

Damien Borgetto, Georges Da Costa, Jean-Marc Pierson, Amal Sayah

Université de Toulouse,
IRIT,
{borgetto,dacosta,pierson,sayah}@irit.fr

Résumé
Cet article traite de la réduction d'énergie dans les systèmes à large échelle, plus particulièrement l'impact de la prise en compte de la consommation énergétique sur la consolidation de serveurs. Diminuer le nombre de machines physiques utilisées tout en garantissant une certaine qualité de services est au cœur de notre approche. Nous introduisons une métrique appelée le yield énergétique qui représente la qualité du placement de tâches sur un ensemble de machines, en prenant en compte l'économie d'énergie et la qualité de services. Elle est matérialisée par la différence entre ce qui est demandé par une tâche et ce que le système lui alloue in fine en prenant aussi en compte le facteur économie d'énergie. Nous cherchons bien entendu à minimiser cette différence. Nous proposons des heuristiques de placement que nous comparons avec l'existant et avec l'optimal. Nous présentons dans cet article un ensemble d'expérimentations qui montrent la pertinence de cette métrique pour réduire de manière significative la consommation énergétique.

Mots-clés : réduction, énergie, consolidation, minimisation, optimale

1. Introduction et motivation

Nous constatons depuis plusieurs années un accroissement de la consommation électrique dû aux serveurs de calculs et aux centres de données. Des études récentes [7],[2] aux Etats Unis et en Europe ont montré que la consommation électrique devient préoccupante. Les opérateurs de centres de calcul et de données investissent des sommes très importantes pour pouvoir alimenter en énergie leur infrastructure ou pour en disperser la chaleur émise.
Plusieurs solutions existent pour diminuer la facture énergétique, tant du point de vue financier que environnemental. Une première solution est basée sur l'efficacité énergétique de l'infrastructure elle-même, favorisant la circulation d'air ou le refroidissement alternatif des salles machines. La seconde solution consiste à déployer des matériels moins gourmands en énergie, ou tout au moins où le rapport flops/watts est le meilleur [5]. Les grands constructeurs de matériels, du CPU au disque dur ou au réseau améliorent continuellement leurs produits dans cette direction.
La troisième solution est algorithmique et s'intéresse à optimiser les traitements à effectuer pour diminuer l'empreinte énergétique. Cette solution regroupe de nombreuses approches, passant par les protocoles réseaux [10], la modélisation de la consommation [6], l'ordonnancement de tâches en prédisant les temps d'inactivité [4]... Nous utilisons l'approche consistant à placer efficacement les jobs sur les hôtes tout en garantissant une certaine qualité de service, pouvant ainsi conduire à des extinctions de jobs. Cette approche a été étudiée dans un contexte différent par Stillwell et al. [11]. Alors que ces travaux s'intéressent à la consolidation de serveurs pour optimiser l'utilisation d'un sous-ensemble de machines cohérent (voir détails en partie 2), nous allons plus loin en intégrant la dimension de la consommation électrique pour diminuer globalement l'impact énergétique. L'idée est de faire émerger un compromis entre un coût énergétique faible et une qualité de services maintenue. Trivialement, l'utilisation d'une seule machine, choisie pour consommer le moins d'énergie, optimise la consommation énergétique mais ne garantit pas la qualité de services des tâches. Réciproquement utiliser le maximum de machines peut garantir aux jobs d'être servis en fonction de leurs besoins mais consomme beaucoup d'énergie. Le

gaspillage de l'énergie est évité lorsque des tâches sont regroupées sur des machines physiques. Mais comme nous le montrons dans la suite de cet article, l'approche naïve consistant à éteindre les machines non utilisées, même après les avoir triées par ordre de consommation, peut être améliorée en définissant plus finement une métrique permettant de placer les tâches en direction d'un compromis.
La suite de cet article est organisé comme suit. En section 2 nous fournissons un ensemble de notations et nous expliquons l'approche de consolidation de serveurs utilisés par Stillwell et al. qui a servie de point de départ à ce travail. En section 3, nous présentons notre approche théorique pour intégrer la prise en compte du paramètre énergétique dans le placement de tâches, en étendant et généralisant le modèle précédent. En partie 4, nous déclinons un certain nombre de résultats montrant la pertinence de notre approche et nous évaluons les gains en termes d'énergie (potentiellement important) par rapport à la qualité de services des tâches. Enfin, nous concluons en section 5 et ouvrons un certain nombre de perspectives, notamment pour étendre notre modèle et pour sa mise en œuvre effective dans un système distribué par l'intermédiaire de l'utilisation de machines virtuelles.

2. Consolidation de serveurs

Les travaux de Stillwell et al. dans [11] concernent une heuristique adressant le problème du placement de jobs dans un cadre multi-contraintes, mémoire et charge processeur. Il y est fait l'hypothèse que le matériel est totalement homogène. Ce travail est basé sur une métrique, le yield, définissant la qualité d'un placement. Un bon placement aura pour but de maximiser cette métrique.
$Y_{ij} = \sum_{j=1}^{H} (\frac{\alpha_{ij}}{\alpha_i})$
α_{ij} : part de CPU de la machine *j* effectivement allouée au job *i*.
α_i : part de CPU que le job *i* a demandé.
H : Nombre d'hôtes

Le yield représente en quelque sorte le taux de satisfaction du job, c'est-à-dire la part de ressources affectée par rapport à la part de ressources allouée. Par exemple un job qui demande 60% des ressources d'un hôte, mais s'en voit affecter seulement 30%, aura un yield de $\frac{30}{60} = \frac{1}{2}$. Les travaux de [11] portent sur l'allocation de ressources, via du *multi capacity bin-packing* (MCB)[8] sur la mémoire et la charge processeur.
Le travail de [11] utilise cette métrique pour aggréger des services sur un nombre minimal de machines, tout en garantissant une bonne qualité de service en maximisant le yield minimum.
L'algorithme proposé se déroule en plusieurs phases. Il faut :

1 Fixer une valeur de Y du yield à atteindre.
2 Déduire du Y les besoins CPU afin que chaque job arrive à un yield Y.
3 Trier la liste des jobs en 2 listes, ceux qui ont plus de besoins CPU que de mémoire, et ceux qui ont plus de besoins mémoire que de CPU.
4 Appliquer le MCB, c'est à dire remplir chaque hôte, en prenant dans la liste le job qui va contre le déséquilibre actuel. Par exemple pendant l'algorithme, un hôte est chargé à 60% en CPU et à 50% en mémoire, il faut regarder en premier dans la liste des jobs dont les besoins mémoire sont supérieurs à ceux CPU. Si on ne trouve aucun job dans la première liste on regarde dans l'autre, si on ne trouve pas non plus l'algorithme échoue. Si l'hôte est plein on prend le suivant jusqu'à ce qu'il n'y ait plus d'hôtes (échec) ou plus de jobs (succès).
5 Recommencer 1, 2, 3 et 4 afin d'effectuer une dichotomie sur Y et trouver le meilleur compromis possible.

Cette heuristique est donc en temps polynomial en le nombre de jobs et de machines. Cette technique permet donc de rassembler les jobs sur un faible nombre de machines, et donc d'éteindre celles qui ne sont pas utilisées. Mais la prise en compte de l'énergie se situe uniquement au niveau de l'extinction des machines.
Les performances de cette heuristique dépendent fortement du tri utilisé pour trier les listes de jobs. Dans [11], le tri le plus efficace est MCB8 qui consiste à trier selon l'ordre décroissant le maximum de la mémoire et du processeur demandé (ie max(mémoire,CPU) en ordre décroissant). Dans la suite, après

avoir étudié le comportement des 8 différents tris [11], nous traiterons MCB4 en plus de MCB8 pour ses bonnes propriétés quand à l'économie d'énergie. Le tri MCB4 consiste à trier selon l'ordre croissant le maximum de la mémoire et du processeur demandé (ie max(mémoire,CPU) en ordre croissant).

3. Prise en compte de l'énergie

L'approche actuelle de Stillwell et al. est centrée autour de la métrique (le *yield*) mesurant la qualité d'un placemement. Plus un job reçoit les ressources qu'il a demandé, plus le yield est important. L'hypothèse de base était la totale homogénéité des machines. Nous avons choisi de conserver cette hypothèse car on trouve souvent dans les clusters un ensemble de machines ayant les mêmes spécifications. De plus, des études [9] ont montré que la localisation physique des machines induisait des différences de consommation d'énergie.

Pour tenir compte de la problématique de l'énergie sans trop complexifier le modèle, nous posons les hypothèses suivantes :

(H1) Les machines sont énergétiquement hétérogènes, mais offrent toutes la même puissance de calcul.

(H2) Une machine consomme C^{min} watts au repos et C^{max} en charge maximale.

(H3) Le surplus d'énergie consommée par un job sur un hôte est fonction f (linéaire) de ses ressources CPU ainsi que des caractéristiques de l'hôte. C'est à dire $\forall i \in [1..J], \forall j \in [1..H] : \delta C_{ij} = \alpha_{ij} \times f(j)$ où δC_{ij} est la consommation induite par le job i sur la machine j. En première approximation, $f(j) = (C_j^{max} - C_j^{min})$

(H4) Comme dans [11], nous nous plaçons pour commencer dans un contexte sans migration, avec des jobs infinis.

3.1. Yield énergétique

Il est donc nécessaire de modifier la métrique de façon à prendre en compte la problématique de l'énergie dans le cadre de ces hypothèses.

On utilise la formule suivante pour le yield energétique.

Pour un job $i \in [1..J]$, sur un hôte $j \in [1..H]$, on pose :

$$YE_{ij} = \frac{\left[\sum_{j=1}^{H}(\frac{\alpha_{ij}}{\alpha_i})\right]^{1-k}}{\left[\lambda(C_j^{max} - C_j^{min}) \times \alpha_{ij} + (1-\lambda)(A_j(1 - \sum_{i'=1,i'\neq i}^{J}(\alpha_{i'j})))\right]^{k}} = \frac{(Y_{ij})^{1-k}}{(E_{ij})^{k}}$$

Avec $0 \leq \lambda \leq 1$ et $0 \leq k \leq 1$.

On utilise l'hypothèse selon laquelle l'énergie consommée (δC_{ij}) par le job i lorsqu'il utilise la proportion α_{ij} de la puissance de calcul de la machine j est $(C_j^{max} - C_j^{min}) \times \alpha_{ij}$.

La composante Y_{ij} chiffre la qualité d'allocation des ressources de calcul aux jobs. La composante E_{ij} est chargée de tenir compte de la problématique de l'énergie. k permet de fixer l'importance relative entre les performances et l'économie d'énergie.

Le terme énergétique se décompose en deux parties : $E_{ij} = \lambda \delta C_{ij} + (1-\lambda) \times A_j(1 - \sum_{i'=1,i'\neq i}^{J}(\alpha_{i'j}))$

- Le terme δC_{ij} représente la contribution du job à la consommation d'énergie de la machine j.
- Le terme $A_j(1 - \sum_{i'=1,i'\neq i}^{J}(\alpha_{i'j}))$ permet le regroupement des jobs sur les hôtes attractifs. A_j représente l'attractivité de la machine j et est d'autant plus petit que la machine est attractive. Pour la suite nous prendrons $A_j = C_j^{min}$. Ce terme est d'autant plus petit (donc menant à un yield énergétique grand) qu'une machine est à la fois attractive et déjà chargée.

λ permet de pondérer entre les deux termes ci-dessus, afin de mettre plus ou moins d'importance sur l'un ou l'autre. Ainsi, si il est totalement impossible d'éteindre les machines, on prendra $\lambda = 1$. Notre approche permet de donner la qualité d'un job à un instant donné, sans tenir compte des placements à venir.

3.2. Propriétés

Cette métrique a les 3 propriétés principales suivantes.

Propriété 1 : Le scheduler préfère une machine qui consomme moins à une autre lorsque les autres propriétés sont égales. C'est-à-dire que si l'on a 2 machines dont la charge et le yield proposés sont

identiques, lors de l'arrivée d'un job, la métrique préfère celle dont l'augmentation de consommation d'énergie est la plus basse.
$\forall i \in [1, J]; \forall j, h \in [1, H]; j \neq h; k \neq 0; \lambda \neq 0 :$
$(\sum_{i'=1, i' \neq i}^{J} (\alpha_{i'j}) = \sum_{i'=1, i' \neq i}^{J} (\alpha_{i'h})$
$\wedge \sum_{j=1}^{H} (\frac{\alpha_{ij}}{\alpha_i}) = \sum_{h=1}^{H} (\frac{\alpha_{ih}}{\alpha_i})$
$\wedge A_j = A_h$
$\wedge \delta C_{ij} < \delta C_{ih})$
$\Rightarrow YE_{ij} > YE_{ih}$

Propriété 2 : À caractéristiques équivalentes, le scheduler préfère placer un job sur une machine possédant déjà des jobs (et ne pouvant donc pas être éteinte) que sur une machine vide. Ainsi, si l'on a 2 hôtes et 2 jobs, on préfère mettre les 2 jobs sur une seule machine plutôt que de les répartir les 2 hôtes.

$\forall i \in [1, J]; \forall j, h \in [1, H]; j \neq h; k \neq 0; \lambda \neq 1 :$
$(\sum_{i'=1, i' \neq i}^{J} (\alpha_{i'j}) > \sum_{i'=1, i' \neq i}^{J} (\alpha_{i'h})$
$\wedge \sum_{j=1}^{H} (\frac{\alpha_{ij}}{\alpha_i}) = \sum_{h=1}^{H} (\frac{\alpha_{ih}}{\alpha_i})$
$\wedge A_j = A_h$
$\wedge \delta C_{ij} = \delta C_{ih})$
$\Rightarrow YE_{ij} > YE_{ih}$

Propriété 3 : Il est possible de régler la sensiblité du système à l'énergie. Ici, le paramètre k varie entre 0 et 1, tel que si $k = 0$ on prend en compte uniquement le yield et si $k = 1$ on ne prend en compte que l'énergie. Ainsi, lorsque k augmente, les propriétés précedentes sont petit à petit étendues pour accepter une perte de yield.

3.3. Optimisations

L'algorithme proposé dans [11] repose sur un algorithme de bin-packing utilisant le yield pour placer correctement les jobs sur les hôtes. L'article compare plusieurs techniques pour trier les jobs et évaluer ainsi l'impact du tri sur la qualité de l'ordonancement résultant.
Un des apports de notre approche est de tenir compte de l'hétérogénéité des machines. L'ordre de prise de ces machines par le bin-packing a alors un impact sur la performance de l'algorithme. Nous comparerons dans la suite les tris suivants :
– TH1 : C^{min}, ordre croissant
– TH2 : C^{max}, ordre croissant
– TH3 : $C^{min} + C^{max}$, ordre croissant
– TH4 : $C^{max} - C^{min}$, ordre croissant

4. Expérimentations

Afin d'évaluer notre approche, nous avons effectué différentes simulations, en regardant l'évolution d'une part de la dégradation du yield avec l'augmentation de k, et d'autre part la consommation d'énergie du système. L'énergie du système étant la somme des énergies consommées par chaque machine dont la charge induite par les jobs n'est pas nulle.

4.1. Méthodologie

Nous avons généré des séries de problèmes en nous basant sur la génération des problèmes dans [11] , un problème étant une génération d'hôtes et de jobs, selon la méthode expliquée ci-après. Les besoins CPU des jobs sont générés selon une loi normale de variance 0.25 et 0.75, et de moyenne 0.5. Les besoins mémoire suivent une loi normale de variance 0.25 et 0.75 et de moyenne $H * (1 - slack)/J$. H étant le nombre d'hôtes, J le nombre de jobs, et *slack* représentant le pourcentage de mémoire libre restant sur le système une fois les jobs générés.
Ensuite les problèmes sont générés avec, pour les petits problèmes, 4 hôtes, 6, 8, 10 et 12 jobs. Ce qui nous fait un total de 144 spécifications de problèmes différentes. Pour chaque spécification, on génère 10 problèmes. Ce qui nous fait un total de 1440 simulations pour les petits problèmes. Nous faisons de

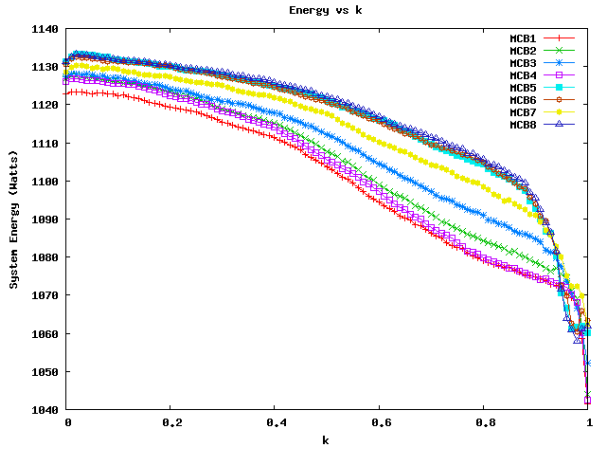

FIG. 1 – Variation de l'énergie du système en fonction de k, petits problèmes

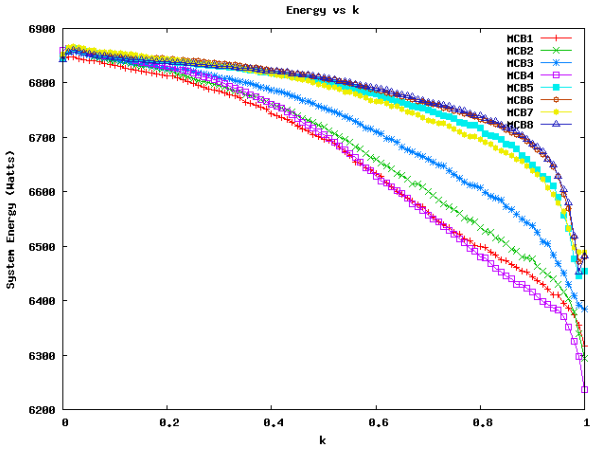

FIG. 2 – Variation de l'énergie du système en fonction de k, problèmes moyens

même pour des problèmes moyens, en multipliant le nombre d'hôtes et de jobs par 6, passant ainsi à 24 hôtes, 36, 48, 60, 72 jobs.
Pour mettre en évidence l'influence de notre métrique sur l'énergie du système, nous effectuons la recherche d'une solution proche optimale avec les algorithmes MCB de [11], en faisant, pour chaque problème généré, varier k (l'influence énergétique de la métrique) de 0 à 1 par pas de 0.01. De plus, ces tests ont été conduits à des valeurs de λ de 0 à 1 par pas de 0.1.
Enfin, pour pouvoir caractériser les hôtes, nous avons attribué à chaque hôte une valeur C^{min} comprise entre 100 et 200 watts, une valeur C^{max} comprise entre 200 et 400 watts, générées selon une loi uniforme. Pour plus de précision vis-à-vis de l'existant, il sera nécessaire de générer selon un modèle plus réaliste.

4.2. Résultats

Pour analyser les résultats de nos simulations, nous nous focalisons sur l'énergie du système ainsi que sur le yield moyen. En effet, l'énergie du système sera directement un guide pour évaluer l'impact de notre méthode sur la réduction de l'énergie, alors qu'analyser le yield moyen nous permettra de regarder à quel point les performances en sont affectées. Nous présentons tout d'abord l'évolution de l'énergie d'un système pour observer comment celle-ci se comporte lorsque l'on fait varier k. La figure 1 a été générée sur des petits problèmes, avec les 8 algorithmes de MCB, sans tri des hôtes (ainsi, le premier est rempli indifféremment de ses caractéristiques). Comme on peut le voir, plus on se rapproche de $k = 1$, plus l'énergie diminue. Ceci est dû en majeure partie au fait que si l'importance est accordée en majorité à la composante énergie, seule la mémoire sera limitante dans le placement des jobs, et on aura un regroupement maximum pour un k proche de 1.
Nous voyons aussi qu'il subsiste une différence au niveau énergétique entre les différents MCB. En effet, le MCB8 par exemple étant celui qui arrive à s'approcher le plus de l'optimal, on pourrait le penser meilleur au niveau énergétique. Cependant, comme nous pouvons le voir dans la figure 1, c'est l'inverse qui se produit, bien que de manière faible. Ceci est dû au fait que l'algorithme MCB4 par exemple qui est le plus faible en énergie, échoue plus souvent que l'algorithme MCB8. Rappelons ici que MCB4 trie les jobs par max(CPU,mémoire) en ordre croissant et MCB8 par ordre décroissant, et les place dans cet ordre. Ainsi, lorsque MCB4 n'échoue pas, l'agencement résultant sera tel qu'après la phase d'augmentation du yield moyen, la charge globale du système sera plus faible que pour le MCB8, ayant ainsi non seulement une énergie plus faible, mais aussi un yield moyen plus faible.
La figure 2 donne un aperçu de la variation de l'énergie lorsque nous passons à des problèmes un peu plus larges. Ici ce sont les problèmes moyens. Contrairement à la figure 1, nous avons une baisse nette plus rapidement lorsque l'on augmente le k avec par exemple le MCB8. On voit ici que l'écart entre le meilleur MCB et le moins bon s'accentue à mesure que l'on augmente la taille du problème. On peut

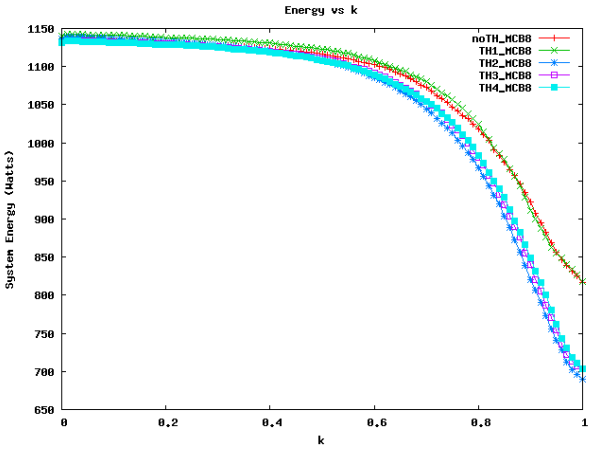

FIG. 3 – Comparaison des différents tris des hôtes, petits problèmes

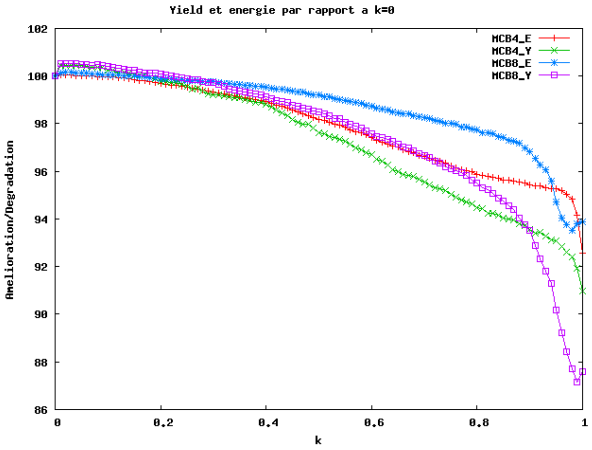

FIG. 4 – Comparaison des pertes d'énergie et de yield moyen en fonction de k, petits problèmes

penser ici que l'agencement issu du MCB4 par exemple, qui est superieur à celui du MCB8 en terme d'énergie, donnera de meilleurs résultats sur de gros problèmes. Cependant, le yield moyen est aussi plus haut pour le MCB8 que pour le MCB4.

Comme dit dans [11], une fois le binpacking effectué, il est possible d'éteindre les machines qui n'ont aucun job à exécuter, une fois que les jobs sont regroupés sur les autres machines. Il est possible d'aller plus loin dans la réduction de l'énergie, en dépassant l'approche naïve pour prendre en compte globalement l'énergie. Ainsi on économise de l'énergie supplémentaire en plaçant de manière éclairée les jobs sur les hôtes en fonctions de leurs caractéristiques respectives. En effet, dans [11], un milieu homogène est considéré. En pratique ce n'est évidemment pas le cas, car même si l'on a des machines identiques, elles ne consommeront pas forcément la même électricité en fonction de l'endroit où elles sont placées physiquement [9]. La figure 3 montre l'importance de tenir compte de l'hétérogénéité de la configuration des hôtes. On peut y voir qu'avec des tris d'hôtes simples, sur seulement 4 hôtes, on peut déjà économiser un montant non négligeable d'énergie. Sans tri des hôtes, on a une performance plus faible car l'hôte à remplir en premier est le premier de la liste. On peut ainsi facilement imaginer que sur un nombre d'hôtes largement supérieur à un nombre de jobs, cas qu'il n'est pas rare de rencontrer dans les grilles de calcul ou les clusters lors des périodes creuses, on peut avoir facilement une augmentation de l'efficacité énergétique du système.

La figure 4 nous montre le pourcentage de perte d'énergie et le pourcentage de perte de yield moyen, en fonction de la variation de k. On peut y voir que, par exemple, avec l'algorithme MCB8 et avec $k = 0.6$, une perte de yield de 2.5% nous permet d'économiser environ 1.5% d'énergie. Si l'on imagine par exemple les cas de jobs peu contraints par le temps, cette perte de yield est acceptable. De plus, une perte en yield de 2.5% n'est pas forcément une augmentation de 2.5% du temps de calcul, ni une augmentation de 2.5% de l'énergie, du fait que la majorité de la consommation énergétique d'une machine vient de C^{min}. Bien entendu, ceci n'est que pour les petits problèmes, donc sur seulement 4 hôtes, ce qui veut dire que la marge de manœuvre est relativement restreinte. Il faut ajouter aussi que les pourcentages sont générés en fonction des valeurs respectives des algos. Autrement dit par exemple une perte de yield de 2% pour le MCB8 est plus grande en valeur que 2% pour le MCB4, car le yield moyen du MCB8 est plus élevé que celui du MCB4.

Pour un $k = 0.99$, c'est à dire à l'endoit où la différence entre perte de yield moyen et gain d'efficacité énergétique est maximale, nous avons un peu plus de 6% de gain d'énergie, pour une perte de yield d'un peu moins de 13%. Cela correspond au cas où on a la moins bonne efficacité du yield par rapport à l'énergie, mais cela correspond aussi au cas où l'on consomme le moins.

Les figures 5 et 6 montrent l'évolution du yield moyen et de l'énergie à mesure que l'on varie les paramètres k et λ. Comme nous pouvions nous y attendre, là où l'énergie est la plus faible, aux alentours de

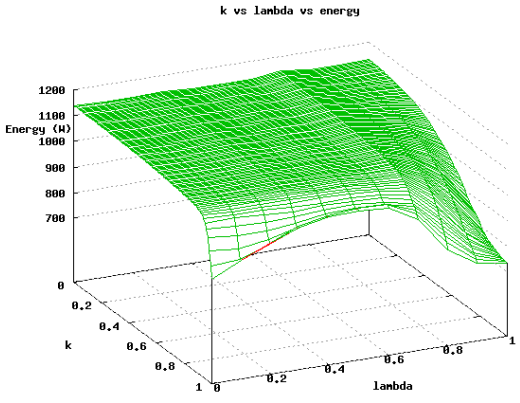

FIG. 5 – Evolution de l'énergie par rapport aux variations de k et lambda, petits problèmes

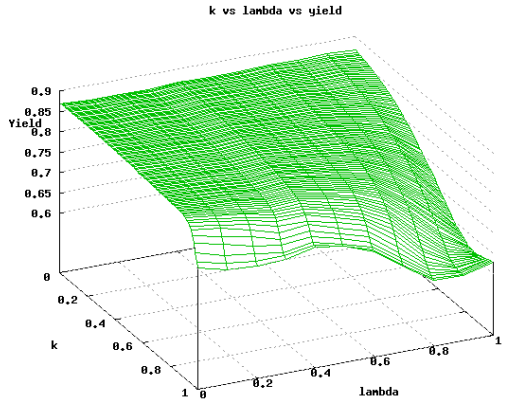

FIG. 6 – Evolution du yield moyen par rapport aux variations de k et labmda, petits problèmes

$k = 1$ et $\lambda = 1$, le yield moyen est aussi le plus faible. Cela correspond au cas où l'on considère dans la composante énergétique seulement le placement des jobs sur les bons hôtes. De plus, il faut noter que les performances atteintes avec $\lambda = 0$ ne sont probablement pas fidèles ici, car il n'y a que peu de marge de manœuvre pour regrouper les jobs sur des hôtes. Pour de telles valeurs, nous économisons environ 40% d'énergie par rapport au cas où l'on ne prend pas en compte l'énergie ($k = 0$), et nous avons une dégradation de 45% du yield moyen. Il est cependant à noter que les simulations donnant ces résultats ont été faites avec le tri TH3 (cf section 3.3) pour les hôtes et le tri MCB8 pour les jobs.

5. Conclusion et perspectives

Dans cet article, nous avons présenté une étude théorique du placement de jobs sur un ensemble de machines homogènes, prenant continuellement en compte les paramètres *réduction de la consommation d'énergie* et satisfaction des jobs en terme d'obtention de ressources sur les sites choisis. L'impact du regroupement des jobs sur un nombre plus réduit de machines a été également géré, ce qui permet aussi d'envisager ou non l'extinction de machines. Les premières simulations réalisées dans un contexte *petits problèmes* ont :

- validé l'intérêt de la prise en compte du paramètre consommation d'énergie en tant que tel et non comme uniquement un effet induit par un placement tendant à charger le plus possible un nombre plus réduit de machines ;
- montré l'impact du paramètre économie d'énergie sur la satisfaction des jobs : plus le paramètre économie d'énergie est pris en considération, plus le yield obtenu se détériore. Le cas extrême conduit ainsi à économiser 40% d'énergie, avec en contre-partie une dégradation du yield moyen de 45% ;
- montré l'importance du tri et donc du choix de la machine hôte la plus énergétiquement favorable pour l'augmentation de l'économie d'énergie ;
- confirmé que le contexte *petits problèmes* offre peu de marge de manœuvre pour favoriser à la fois une économie significative d'énergie et une dégradation faible du yield.

Nous espérons qu'à partir des différents cas étudiés, nous pourrons caractériser des stratégies de placement adaptées aux configurations pour obtenir la meilleure économie possible d'énergie. Le contexte de cette étude (machines homogènes, jobs infinis dont il ne s'agit de gérer que le placement initial, etc.) doit être en outre élargi pour être plus proche de situations réelles et pour prendre en compte des aspects tels que :

- l'hétérogénéité des machines considérées ;
- la création/disparition dynamique de jobs et le réexamen dynamique des affectations des jobs aux hôtes, avec comme corollaire la migration des activités[3] ;
- la prise en compte de paramètres de qualité de service plus contraignants que le niveau d'obtention

des ressources demandées : temps de réponse, échéances, planification, famine, etc. ;
– l'overhead généré par la gestion de l'énergie.

Dans le même temps, nous cherchons à vérifier les résultats théoriques obtenus lors de ces simulations par la mise en œuvre d'un système autonome gérant le placement d'activités (machines virtuelles [1]) sur une architecture matérielle donnée et visant à réduire la consommation d'énergie en utilisant le triptyque *Capteurs* pour observer (des ressources, des activités, etc.) - *Décision* (placement, arrêt/mise en marche de matériels, ...) - *Actionneurs* pour la mise en œuvre des décisions.

Bibliographie

1. Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. In *SOSP '03 : Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 164–177, New York, NY, USA, 2003. ACM.
2. P. Bertoldi and B. Atanasiu. Electricity consumption and efficiency trends in the enlarged european union. Technical Report EUR 22753EN, Institute for Environment and Sustainability, 2007.
3. Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, and Andrew Warfield. Live migration of virtual machines. In *NSDI'05 : Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*, pages 273–286, Berkeley, CA, USA, 2005. USENIX Association.
4. Georges Da-Costa, Jean-Patrick Gelas, Yiannis Georgiou, Laurent Lefèvre, Anne-Cécile Orgerie, Jean-Marc Pierson, Olivier Richard, and Kamal Sharma. The green-net framework : Energy efficiency in large scale distributed systems. In *HPPAC 2009 : High Performance Power Aware Computing Workshop in conjunction with IPDPS 2009*, Rome, Italy, may 2009.
5. W. Feng, M. Warren, and E. Weigle. Honey, i shrunk the beowulf ! In *International Conference on Parallel Processing, 2002. Proceedings.*, pages 141–148, 2002.
6. Helmut Hlavacs, Georges Da Costa, and Jean-Marc Pierson. Energy consumption of residential and professional switches. Rapport de recherche IRIT//RR-2009-7-FR, IRIT, Université Paul Sabatier, Toulouse, mars 2009.
7. J. G. Koomey. Estimating total power consumption by servers in the u.s. and the world. Technical report, Stanford University, 2007.
8. William Leinberger, George Karypis, and Vipin Kumar. Multi-capacity bin packing algorithms with applications to job scheduling under multiple constraints. In *ICPP '99 : Proceedings of the 1999 International Conference on Parallel Processing*, page 404, Washington, DC, USA, 1999. IEEE Computer Society.
9. A.-C. Orgerie, L. Lefevre, and J.-P. Gelas. Chasing gaps between bursts : Towards energy efficient large scale experimental grids. In *Ninth International Conference on Parallel and Distributed Computing, Applications and Technologies, 2008. PDCAT 2008.*, pages 381–389, Dec. 2008.
10. Barry Rountree, David K. Lowenthal, Shelby Funk, Vincent W. Freeh, Bronis R. de Supinski, and Martin Schulz. Bounding energy consumption in large-scale mpi programs. In Becky Verastegui, editor, *Proceedings of the ACM/IEEE Conference on High Performance Networking and Computing, SC 2007, November 10-16, 2007, Reno, Nevada, USA*, page 49. ACM Press, 2007.
11. M. Stillwell, D. Schanzenbach, F. Vivien, and H. Casanova. Resource allocation using virtual clusters. In *Proceedings of the 9th IEEESymposium on Cluster Computing and the Grid (CCGrid'09)*, may 2009.