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

Modèles fluides pour l'économie d'énergie dans les grilles par migration : une première approche.

Georges Da Costa, Guillaume Dufour, David Sanchez

IRIT, Université Paul Sabatier, 118 route de Narbonne 31062 TOULOUSE Cedex 9, dacosta@irit.fr
DTIM-M2SN, ONERA Toulouse, 2 av. E. Belin 31055 TOULOUSE Cedex 4, dufour@onera.fr
IMT, INSA de Toulouse, 135 av. de Rangueil 31077 TOULOUSE Cedex 4, dsanchez@insa-toulouse.fr

---

Résumé
La consommation énergétique des serveurs de calculs représente une partie non négligeable de la consommation électrique mondiale. Gérer ainsi de manière fine les *jobs* s'éxécutant sur les grilles permet d'améliorer sensiblement l'efficacité énergétique des centres de calculs, qui représentent une part importante de la consommation liée à l'informatique. Il est difficile de tester, d'évaluer et de comparer les techniques de gestion de *jobs* ainsi que leur impact énergétique. De plus, avec la généralisation de la possibilité de migration de ces tâches, de nouveaux leviers de réduction de consommation apparaissent. Il devient alors possible de réduire la consommation de manière proactive, en choisissant de manière pertinente les tâches à migrer, ainsi que leur lieu de destination. Dans cet article, nous allons présenter une nouvelle approche basée sur une modélisation fluide pour résoudre la problématique de l'évaluation rapide et efficace de gestion de *jobs* dans les grilles de calcul, et ceci afin de permettre une optimisation de celle-ci du point de vue énergétique. Le modèle mathématique utilisé provient d'une limite macroscopique formelle effectuée à partir d'une description particulaire des *jobs*. La résolution de ce modèle est beaucoup plus rapide que les simulations particulaires classiques. On passe d'un temps linéaire en le nombre de *jobs* à un temps indépendant de ce nombre. Nous présentons ce modèle ainsi qu'une validation par rapport au modèle particulaire.

Mots-clés : Grilles, Modélisation fluide, Migration, Economie d'énergie, Ordonnancement

---

1. Introduction

Nous constatons depuis plusieurs années un accroissement de la consommation électrique dû aux serveurs de calculs et aux centres de données. Des études récentes [1,2] aux Etats Unis et en Europe ont montré que la consommation électrique devient préoccupante. De plus, lors du CeBit 2008 à Hanovre, des estimations ont montré que l'utilisation d'Internet au niveau mondial nécessite l'équivalent de 14 centrales électriques pour alimenter les ordinateurs et serveurs, produisant une empreinte carbone équivalente à celle de l'industrie du transport aérien. 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, ces coûts étant actuellement équivalents au coût d'achat de l'infrastructure en moins de trois ans.

Plusieurs solutions existent pour diminuer la facture énergétique, tant du point de vue financier qu'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 du moins dont le rapport flops/watts est meilleur [3]. 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 [4], la modélisation de la consommation [5], l'ordonnancement de tâches en prédisant les temps d'inactivité [6]...

Ces approches reposent souvent sur une prise en compte de problématique de l'économie d'énergie lors de la création des logiciels, ou à des choix statiques de positionnement de tâches. La troisième possibilité, consiste à optimiser en temps réel la consommation en migrant [7] les tâches, par exemple en utilisant des machines virtuelles. Ainsi il devient par exemple possible de déplacer des tâches sur des sites où il fait nuit, réduisant les besoins en climatisation.
Le système SLM (*Subtle Load Management*) a pour objectif de gérer de manière fine les tâches présentes dans les grilles à grande échelle afin de réduire la consommation électrique. Cette réduction peut provenir de plusieurs effets, tels que le déplacement de tâches vers des sites énergétiquement efficaces ou vers des sites peu chargés et donc ne nécessitant pas une utilisation extrême des moyens de climatisation.
Cette première étude a pour but double de fournir des outils d'évaluation de la qualité de SLM ainsi que d'en fournir un exemple de simulation numérique efficace.
La suite de cet article est organisée comme suit. En section 2 nous préciserons le cadre dans lequel cette étude se place. En section 3 nous exposerons l'approche multi-résolution utilisée pour obtenir un système de migration efficace. En section 4 nous présenterons une validation de cette approche. Enfin, nous concluons en section 5 et ouvrons un certain nombre de perspectives, notamment pour étendre notre modèle.

2. Motivation

Ces dernières années, la demande en terme de simulations numériques a fortement augmenté, celles-ci permettant de réduire le nombre d'expérimentations réelles et donc de diminuer les coûts de développements. Avec cette demande croissante, l'équipement informatique des différents pays s'est adapté et il existe maintenant à l'échelle mondiale un réseau de clusters de calculs plus ou moins performants. Toutefois, la demande énergétique des moyens informatiques dédiés aux calculs intensifs a suivi la même croissance, jusqu'à devenir une part non négligeable dans le budget alloué aux laboratoires de recherche. Les répercussions vont même jusqu'à impacter la politique environnementale avec des demandes en climatisation de plus en plus importantes.
L'étude présentée dans ce document a pour but d'optimiser les dépenses énergétiques nécessaires à la réalisation d'un calcul, tout en essayant de limiter l'impact sur les temps d'exécution. Considérant que les différents clusters de calculs disponibles dans une région donnée, éventuellement sur l'intégralité du globe terrestre, sont interconnectés par des liaisons rapides permettant le transfert de données à haut débit, le principe est de proposer une répartition dynamique de la charge de travail de manière à optimiser le rendement énergétique. Cette approche d'optimisation sous contraintes pour la distribution de calcul sera appelée méthode SLM (Subtle Load Management).
L'organisation que l'on se propose d'étudier est composée d'éléments dont la taille relative diffère de plusieurs ordres de grandeur. En effet, l'élément qui sera considéré comme élémentaire est le processeur, ou plus exactement un cœur de processeur, chacun pouvant exécuter un et un seul processus. Le niveau immédiatement supérieur sera la 'machine', composée d'un nombre fixé de processeurs. Afin de simplifier un peu la description, on traitera chaque cœur de processeur comme un processeur à part entière, nous négligerons donc tous les aspects liés au partage de mémoire, de vitesses de bus internes dans l'évolution des performances d'une machine. Concrètement, une machine physique dont la carte-mère comporte deux processeurs quad-cores sera modélisée comme une machine avec huit processeurs et pourra donc exécuter jusqu'à huit processus simultanément. Un ensemble de machines *toutes identiques* sera appelé un cluster. Enfin, l'échelle la plus grande représente la répartition des clusters.
L'objectif sera de minimiser au mieux les paramètres suivants :
- *Energie consommée :* elle représente l'énergie utilisée par les machines en charge ainsi que celle nécessaire au fonctionnement de la climatisation.
- *Temps de traitement :* c'est le temps total mis pour exécuter le travail demandé.
- *Coût de location :* il s'agit du coût financier demandé pour la location de temps de calcul sur des clusters.
- *Bande passante :* les échanges entre clusters pouvant être fréquents, il s'agit de minimiser l'impact de ceux-ci sur le réseau.

3. Modélisation fluide des grilles de calcul : une approche hiérarchique

En amont de la mise en œuvre de cette méthode, nous nous intéressons essentiellement à la simulation et à la modélisation mathématique de celle-ci afin de pouvoir tester rapidement différents protocoles de répartition de charge. Une fois la structure du réseau de clusters donnée, nous considérons un grand nombre de *jobs* évoluant dynamiquement sur celui-ci, chaque cluster décidant à intervalles de temps donnés, ce qu'il advient de chaque *job*, c'est-à-dire si ce dernier continue à s'exécuter sur place ou bien s'il est envoyé vers un autre cluster afin d'optimiser les coûts ou le temps de traîtement restant. Une telle simulation, dite particulaire, faisant intervenir des *jobs* avec leur caractéristiques propres, leurs interactions *via* l'occupation des clusters et leur influence sur les performances globales du cluster se révèle numériquement lourde à mettre en œuvre quand on s'intéresse à son comportement à très grande échelle, c'est-à-dire quand le nombre de *jobs* considérés est très élevé. Notre objectif est de proposer des modèles moyennés du problème, valables à grande échelle, utilisant le formalisme des équations aux dérivées partielles (voir par exemple les travaux de P. Degond [8,9] sur les modèles de trafic routier, de G. Dufour [10] sur les gouttes) et qui pourront être simulés numériquement à un coût indépendant du nombre de *jobs* considérés. Nous nous restreignons ici à l'optimisation conjointe du coût énergétique des *jobs* et de leur temps de calcul, l'optimisation énergétique liés aux problèmes de climatisation étant réservée à un travail ultérieur.
L'approche présentée dans cet article repose sur une hiérarchisation des différents niveaux de description, laquelle permet, en partant d'une description précise de l'évolution des valeurs modélisées (niveau particulaire) de déterminer de façon rigoureuse l'évolution de la population complète des *jobs* selon leurs caractéristiques (niveau mésoscopique ou cinétique), *via* la définition d'une fonction de répartition. Les variations des quantités observables de l'ensemble des *jobs* peuvent alors être modélisées en calculant les moments de cette fonction de répartition. Cette méthodologie présente l'avantage de pouvoir recréer, sous la contrainte de quelques hypothèses de forme de distribution des *jobs*, une approximation de l'état du système au niveau particulaire à partir des quantités moyennées, lesquelles peuvent être calculées avec des algorithmes efficaces et rapides.

3.1. Description du réseau de clusters

Nous considérons ici un sytème de M clusters, notés C_j pour $j \in \{1, \dots, M\}$, en réseau. Chaque cluster est caractérisé par le nombre total de processeurs E_j contenus dans le cluster, l'indice de performance f_j de ses processeurs (supposés tous identiques) et leur consommation individuelle d'énergie en charge par unité de temps Z_j (la consommation de référence zéro - supposé identique pour tous les clusters - étant prise pour des machines allumées au repos). En ce qui concerne les communications entre chaque cluster, nous négligeons dans un premier temps tout effet de saturation lié à la bande passante et nous nous donnons une table de connectivité globale pour les temps de transfert entre les clusters. Le temps de transfert d'un fichier de taille l en mémoire du cluster C_j vers le cluster C_k est supposé affine en l :

$$\tau_{jk}(l) = \alpha_{jk} + \beta_{jk} l,$$

où α_{jk}, un temps de latence, et $1/\beta_{jk}$, le débit entre j et k, sont connus *a priori*. On ne fera aucun transfert interne à un cluster et on supposera que $\alpha_{jj} = 0$ et $\beta_{jj} = 0$.
Nous supposerons pour simplifier l'écriture des équations que tous les clusters sont synchronisés et effectuent leurs choix de transfert au même moment avec une périodicité T caractérisant la réactivité du système.

3.2. Description particulaire

On suppose avoir un nombre fini N de *jobs* lancés en même temps. Chaque *job* J_i, $1 \leq i \leq N$ est caractérisé par sa charge de travail restant à effectuer q_i, sa taille en mémoire l_i, le nombre de processeurs n_i requis pour son exécution et le cluster P_i sur lequel il s'exécute. Lors d'un transfert entre deux clusters, plutôt que de modéliser le temps de latence lié au temps de transfert, nous supposerons que le *job* est transféré immédiatement du cluster source au cluster cible, mais nous rajoutons un temps d'attente θ_i avant son exécution effective égal au temps de transfert. Plus précisément, le temps d'attente θ_i associé à un *job* sera soit nul (processus en cours d'exécution), soit strictement positif (processus en cours d'attente car venant d'être déplacé).

Le temps requis pour finir l'exécution du *job* J_i sera alors $t^{exe}(q_i, n_i, l_i, , P_i, P_i) = \dfrac{q_i}{n_i f_{P_i}}$ si J_i continue à s'exécuter sur P_i et $t^{exe}(q_i, n_i, l_i, P_i, k) = \dfrac{q_i}{n_i f_k} + \tau_{P_i k}(l_i)$ si J_i est transféré du cluster C_{P_i} vers le cluster C_k. Le côut énergétique du traîtement du *job* sera $K_e(q_i, k) = n_i Z_k \dfrac{q_i}{n_i f_k} = \dfrac{Z_k q_i}{f_k}$.
Soient c_t et c_e les coefficients de pondération donnant l'importance relative accordée au temps d'exécution et à la consommation énergétique. La fonctionnelle que l'on souhaite minimiser est alors :

$$K(q_i, n_i, l_i, k) = c_t t^{exe}(q_i, n_i, l_i, , P_i, k) + c_e K_e(q_i, k).$$

En particulier, soit $k^\star \in \{1, \dots, M\}$ qui minimise la fonctionnelle $K(q, n, l, \cdot)$ pour un *job* de charge q, de longueur l et nécessitant n processeurs, *i.e.* tel que $K(q, n, l, k^\star) = \min\{K(q, n, l, k),\ 1 \le k \le M\}$, on déplacera le *job* de sa position actuelle P vers $k^\star$ à un temps multiple de T uniquement si $k^\star$ est différent de P. On définit alors le processus de décision $Dec(q, n, l, k, P)$ par

$$\begin{array}{rccl} Dec\ : & \mathbb{R}_q^+ \times \mathbb{R}_n^+ \times \mathbb{R}_l^+ \times \{1, \dots, M\} \times \{1, \dots, M\} & \longrightarrow & \{0, 1\} \\ & (q, n, l, k, P) & \longmapsto & \begin{cases} 1 & \text{si } k = k^\star \text{ et } k^\star \neq P \\ 0 & \text{sinon.} \end{cases} \end{array}$$

L'évolution du *job* J_i sur le réseau de cluster s'effectue suivant les équations suivantes :

- la longueur du message à transmettre ne varie pas au cours de l'exécution : $\dfrac{dl_i}{dt} = 0$,
- le nombre de processeurs requis ne varie pas au cours de l'exécution : $\dfrac{dn_i}{dt} = 0$,
- la charge de travail restant à effectuer ne peut que décroître quand le *job* est effectué :

$$\frac{dq_i}{dt} = -n_i f_{P_i} \mathbf{1}_{q_i > 0} \delta_{\theta_i = 0},$$

- le temps d'attente décroît linéairement tant qu'il est strictement positif et ne peut augmenter qu'avec un changement de cluster à un temps multiple de T. Soit $\Psi_i = (\theta_i, P_i)$, on a :

$$\frac{d\Psi_i}{dt} = \frac{d}{dt}\begin{pmatrix} \theta_i \\ P_i \end{pmatrix} = \begin{pmatrix} -\mathbf{1}_{\theta_i > 0} \\ 0 \end{pmatrix} + \delta_{\theta_i = 0} \delta_{\sin(\pi t/T) = 0} \sum_{1 \le k \le M,\ k \neq P_i} \begin{pmatrix} \tau_{P_i k}(l_i) \\ k - P_i \end{pmatrix} Dec(q_i, l_i, n_i, P_i, k).$$

3.3. Equations cinétiques

L'approche précédente consiste à regarder beaucoup de *jobs* différents et à observer leur évolution sur le réseau. Afin d'éviter de les suivre tous individuellement, nous introduisons la fonction de répartition des *jobs*

$$f(t, q, \Psi = (\theta, P), l, n) = \frac{1}{N} \sum_{i=1}^{N} \delta_{q = q_i(t)} \otimes \delta_{\Psi = (\theta_i(t), P_i(t))} \otimes \delta_{l = l_i(t)} \otimes \delta_{n = n_i(t)}$$

qui permet de compter le nombre de *jobs* étant à l'instant t dans le cluster P avec un temps d'attente θ, de longueur l et nécessitant n processeurs pour fonctionner. Pour une fonction test $\phi \in \mathcal{D}(\mathbb{R}_q^+ \times \mathbb{R}_\theta^+ \times \{1, \dots, M\} \times \mathbb{R}_l^+ \times \mathbb{R}_n^+)$, on a :

$$< f(t), \phi >= \frac{1}{N} \sum_{i=1}^{N} \phi(q_i(t), \Psi_i(t), l_i(t), n_i(t))$$

En dérivant cette égalité, on obtient au sens des distributions :

$$\frac{\partial f}{\partial t} - \frac{\partial}{\partial q}(n f_P \mathbf{1}_{q>0} \mathbf{1}_{\theta=0} f) - \frac{\partial}{\partial \theta}(\mathbf{1}_{\theta>0} f) = \sum_{k \neq P} Dec(q, l, n, P, k) < f, \mathbf{1}_{\Psi=(0,P)} > \left(\delta_{\Psi=(\tau_{kP},k)} - \delta_{\Psi=(0,P)}\right).$$

Dans la suite, nous distinguons les *jobs* en attente ($\theta > 0$) des *jobs* en cours d'exécution ($\theta = 0$) en introduisant la fonction de densité f^{exe} associée aux processus en cours d'exécution et la fonction de densité f^{att} associée au processus en attente. On écrit :

$$f^{exe}(t, q, P, l, n) =< f(t, q, \cdot, P, l, n), \mathbf{1}_{\theta=0} > \ \text{ et } \ f^{att} = f - f^{exe} \delta_{\theta=0},$$

ce qui donne les équations suivantes sur f^{exe} et f^{att} :

$$\frac{\partial}{\partial t}f^{exe} - \frac{\partial}{\partial q}(n f_P \mathbf{1}_{q>0} f^{exe}) = f^{att}(\theta = 0^+) - \sum_{k \neq P} Dec(q, l, n, k, P) < f^{exe}, \mathbf{1}_P > \delta_P,$$

$$\frac{\partial}{\partial t}f^{att} - \frac{\partial}{\partial \theta}(\mathbf{1}_{\theta>0} f^{att}) = \sum_{k \neq P} Dec(q, l, n, k, P) < f^{exe}, \mathbf{1}_k > \delta_{\theta=\tau_{kP}} \otimes \delta_P.$$

3.4. Equations fluides

Les équations précédentes décrivent bien la répartition des *jobs* sur le réseau, en conservant l'essentiel de l'information. Le nombre de processeurs utilisés et la longueur des *jobs* n'apparaissent pourtant pas comme des informations essentielles à la compréhension de la dynamique du réseau décrit. Dans le cadre de simulations numériques, ces variables supplémentaires doivent également être discrétisées et compliquent d'autant la résolution. Afin de nous concentrer sur un modèle contenant l'essentiel de l'information, nous travaillons avec les moments en l et n des fonctions de répartitions f, f^{exe} et f^{att} précédentes :

$$\begin{aligned}
\rho(t, q, \theta, P) &= \int_{\mathbb{R}_l^+ \times \mathbb{R}_n^+} f(t, q, \theta, P, l, n)\, dl\, dn, \\
\rho l(t, q, \theta, P) &= \int_{\mathbb{R}_l^+ \times \mathbb{R}_n^+} l.f(t, q, \theta, P, l, n)\, dl\, dn, \\
\rho n(t, q, \theta, P) &= \int_{\mathbb{R}_l^+ \times \mathbb{R}_n^+} n.f(t, q, \theta, P, l, n)\, dl\, dn,
\end{aligned}$$

où ρ est la densités de *jobs* de charge q dans le cluster P à l'instant t et au temps d'attente θ (en exécution (exe) ou en attente (att) si on travaille à partir des fonctions de répartition f^{exe} ou f^{att}), $l = \frac{\rho l}{\rho}$ et $n = \frac{\rho n}{\rho}$ sont les tailles et nombres de processeurs moyens requis par les *jobs*.
En intégrant de même les équations sur f^{exe} et sur f^{att}, nous obtenons un jeu d'équations donnant l'évolution en temps des moments d'ordre k en fonction de conditions sur les moments d'ordre k et $k+1$. Nous avons besoin d'hypothèses supplémentaires concernant la répartition des *jobs* sur le réseau afin d'obtenir un système fini d'équations aux dérivées partielles. Cet aspect constituera un enjeu important des modélisations futures, mais nous pouvons déjà étudier l'évolution du système sous des hypothèses de fermetures simples : nous supposons pour cette étude que tous les *jobs* requièrent exactement le même nombre n_0 de processeurs et occupent la même taille l_0 en mémoire. Cette hypothèse impose la structure $f^{exe} = \rho^{exe}\delta_{l=l_0} \otimes \delta_{n=n_0}$ et $f^{att} = \rho^{att}\delta_{l=l_0} \otimes \delta_{n=n_0}$. Sous ces hypothèses, on obtient le système fluide suivant :

$$\frac{\partial}{\partial t}\rho^{exe} - \frac{\partial}{\partial q}(n_0 f_P \mathbf{1}_{q>0} \rho^{exe}) = \rho^{att}(\theta = 0^+) - \sum_{k \neq P} Dec(q, l_0, n_0, P, k) < \rho^{exe}, \mathbf{1}_P > \delta_P,$$

$$\frac{\partial}{\partial t}\rho^{att} - \frac{\partial}{\partial \theta}(\mathbf{1}_{\theta>0} \rho^{att}) = \sum_{k \neq P} Dec(q, l_0, n_0, k, P) < \rho^{exe}, \mathbf{1}_k > \delta_{\theta=\tau_{kP}} \otimes \delta_P.$$

On remarquera que sous ses hypothèses, nous retrouvons exactement les équations du modèle cinétique mais leur interprétation est sensiblement différente.

4. Validation numérique

Nous nous plaçons dans le cas particulier décrit dans la partie précédente, en supposant que tous les *jobs* requièrent le même nombre de processeurs et ont la même longueur. Dans l'optique de valider l'approche fluide, deux types de solveurs ont été développés. Le premier est de type particulaire et simule un traitement réaliste des *jobs* tel qu'il est habituellement effectué dans un *batch scheduler*, en rajoutant la notion de migration de *jobs*. Ce solveur constituera notre référence en termes de résultats et de coût de calcul et permettra de valider le solveur fluide basé sur la résolution des équations aux dérivées partielles obtenues dans le chapitre précédent.

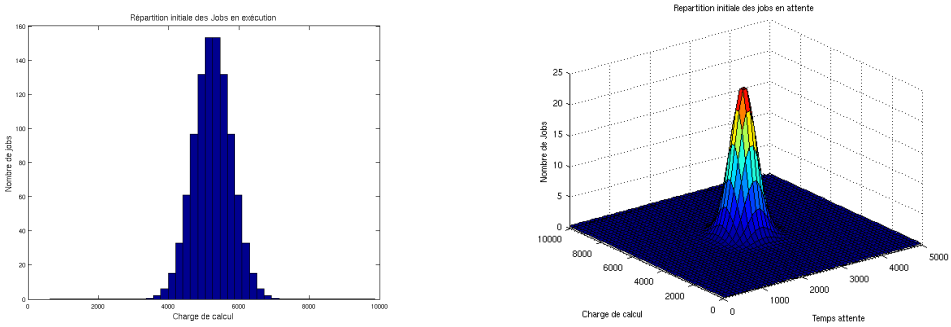

FIG. 1 – Condition initiale des *jobs* en execution (gauche) et en attente (droite)

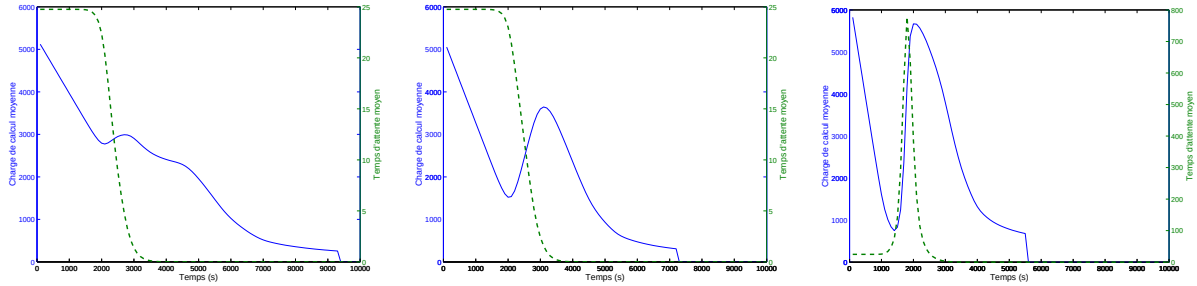

FIG. 2 – Evolution dans les clusters 1, 3 et 5 des charges de calcul (trait plein) et des temps d'attente (tirets) moyens.

4.1. Solveur particulaire

Le code du solveur particulaire a été réalisé en python. Il repose sur une modélisation directe du système afin de fournir une base de comparaison réaliste de référence pour le solveur fluide. Il simule le système par une modélisation discrète du temps. Régulièrement la boucle de décision est appelée avec comme argument les *jobs* et les sites. Le reste du temps, les caractéristiques des *jobs* sont simplement mises à jours (exécutions ou transferts).

4.2. Solveur fluide

Le code du solveur fluide a été réalisé en fortran 90. Les structures de données correspondantes sont basées sur les caractéristiques des *jobs*, ce qui amène à définir, pour chaque cluster, un espace des phases constitué de la charge de calcul q et du temps d'attente θ. En pratique, l'utilisateur est amené à définir des bornes maximales de ces variables, respectivement Q_{MAX} et θ_{MAX}, ainsi qu'une finesse de discrétisation, *i.e* le nombre d'intervalles N_Q et N_θ considérés respectivement sur les espaces $[0, Q_{MAX}]$ et $[0, \theta_{MAX}]$.
La résolution proprement dite des équations de type transport dans ces espaces est effectuée numériquement par une méthode de type Volumes Finis, la précision au second ordre en la variable d'espace étant obtenue grâce à une approche MUSCL [12]. Une caractéristique essentielle de l'algorithme utilisé est la préservation de bornes réelles telles que la positivité du nombre de *jobs* ayant des caractéristiques données. Ce type de conditions est assuré par la limitation des gradients ainsi que par l'utilisation d'un schéma en temps de type splitting [11].

4.3. Description des cas-tests

Pour simplifier l'interprétation des résultats de simulation, le cas-test présenté contient 5 clusters de caractéristiques proches, seul l'indice de performance étant modifié. Les valeurs de coût ont été normalisées et sont dans un premier temps identiques et le nombre de machines est choisi suffisamment grand (10^6) afin de ne pas avoir d'effet de saturation. Les indices de performance ont été choisis d'un rapport 1 à 3 afin d'avoir un ensemble de comportements suffisamment différents, respectivement 1.3, 1.6, 2, 2.6 et 4.

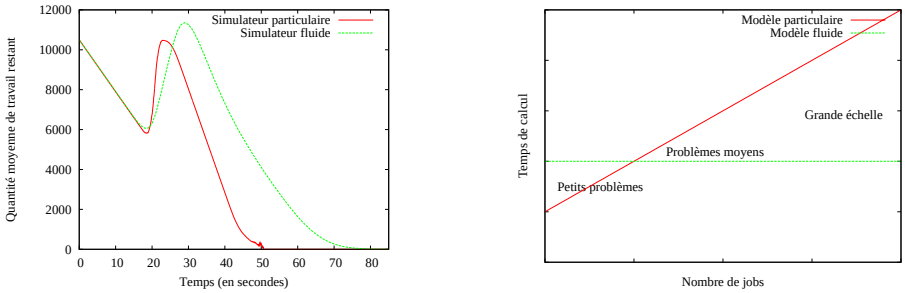

FIG. 3 – Comparaison entre les solveurs particulaire et fluide.

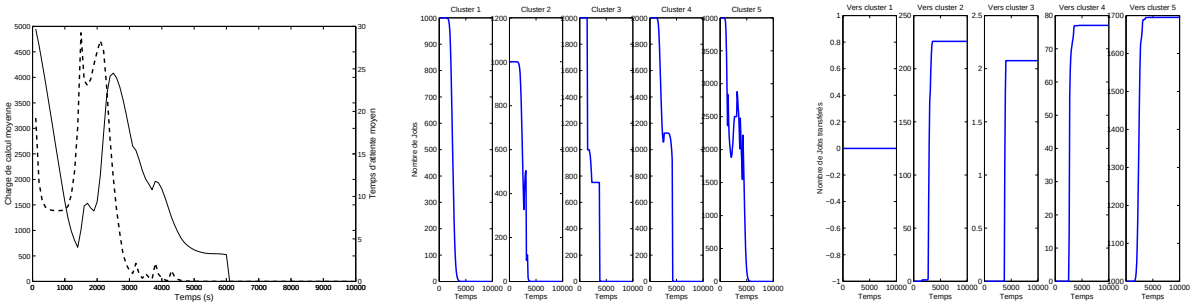

FIG. 4 – Evolution dans le cluster 5 des charges de calcul (trait plein) et des temps d'attente (tirets) moyens (gauche) ; nombre de *jobs* par cluster (centre) et évolution du transfert des *jobs* issus du cluster 1 (droite).

Dans un premier temps, nous nous intéressons à un cas où les temps de transit entre clusters sont pris suffisamment grands pour éviter les transferts des *jobs* afin de permettre une inteprétation simple des résultats et de valider la méthode utilisée. Dans un second temps, afin d'illustrer l'effet des transferts sur la dynamique des clusters, nous considérons une distribution de latences telle qu'il est coûteux d'atteindre le cluster le plus performant (numéro 5), tandis que les transferts entre les clusters 1 et 2 d'une part et 3 et 4 d'autre part sont un peu plus rapides.

4.4. Résultats numériques

Nouc commençons par le cas où il n'y a pas de transferts, la figure 2 représente l'évolution des valeurs moyennes de charge de calcul et de temps d'attente moyen au cours du temps dans les clusters 1, 3 et 5. Ces résultats correspondent qualitativement au comportement attendu. En effet, l'augmentation observée du temps d'attente est d'autant plus importante que le nombre de *jobs* en exécution décroît rapidement. De plus, l'augmentation brutale de la charge de calcul correspond à la disparition du premier pic de *jobs* en exécution. La variation est d'autant plus brutale que l'arrivée des *jobs* en attente se fait rapidement relativement à la vitesse d'exécution du cluster.
Sur la figure 3 sont représentées l'évolution des temps moyens pour les solveurs particulaire et fluide ainsi que la variation de la complexité de calcul en fonction de la taille des systèmes étudiés. La comparaison entre les résultats des deux solveurs permet alors de valider le comportement du solveur fluide. Il faut toutefois noter que la simulation particulaire ayant un coût temporel linéaire en le nombre de *jobs* - contrairement au modèle fluide qui ne dépend que de la discrétisation choisie - les queues de distribution étudiées dans le cas-test sont difficilement simulables en temps raisonnable. Cela explique la divergence entre les deux modèles observée sur des temps longs de simulation.
Dans le cas où les transferts sont effectifs, on observe que l'évolution des valeurs moyennes de charge de calcul et de temps d'attente moyen dans le cluster 5 est modifiée, les différents pics correspondants aux arrivées de *jobs* des autres clusters. Cette évolution, de même que les quantités de *jobs* transférées d'un cluster vers un autre, sont représentée sur la figure 4.
Les comportements observés sont conformes aux attentes, la correspondance entre les latences entre

clusters et les tranferts effectués étant bonne. Les proportions de *jobs* transférés obéissent effectivement à la règle selon laquelle il est intéressant de déplacer un *job* vers un cluster lointain si et seulement si la charge de calcul restant à effectuer est suffisamment importante au regard de la différence de performance entre les clusters.

5. Conclusions et perspectives

Dans cet article, nous avons présenté une nouvelle approche basée sur une modélisation fluide pour résoudre la problématique de l'évaluation rapide et efficace de gestion de *jobs* dans les grilles de calcul, afin de permettre une optimisation de celle-ci du point de vue énergétique. Le modèle mathématique utilisé provient d'une limite macroscopique formelle effectuée à partir d'une description particulaire des *jobs*. Le remplacement du système particulaire originel par un système d'équations aux dérivées partielles permet l'utilisation de techniques numériques de résolution efficaces et uniquement dépendantes de la discrétisation de l'espace des phases. Les premiers résultats numériques présentés, dont la comparaison avec un solveur particulaire, assurent de la capacité du modèle fluide à décrire le comportement attendu des grilles, pour un coût temporel indépendant du nombre total de *jobs*. Rappelons que le solveur particulaire a un coût temporel proportionnel à ce nombre.
Une première étape d'amélioration du modèle consistera à augmenter le nombre de paramètres pris en compte afin d'affiner le réalisme des situations étudiées (taille de clusters limitée, coût de la climatisation, etc.). Cela permettra d'étudier des configurations complexes hors d'atteinte des méthodes classiques particulaires actuelles. Une seconde étape sera alors l'application de cette méthode à l'évaluation de l'efficacité de différents algorithmes de décision pour optimiser l'énergie consommée, le temps de traitement, le coût et/ou la bande passante.

Bibliographie

1. J. G. Koomey. – Estimating Total Power Consumption by Servers in the U.S. and the World. – Tech Report, Stanford University, 2007.
2. P. Bertoldi and B. Atanasiu. – Electricity Consumption and Efficiency Trends in the Enlarged European Union. – Tech Report, Institute for Environment and Sustainability, 2007.
3. Feng, W. and Warren, M. and Weigle, E. – Honey, I shrunk the Beowulf ! – Proceedings. International Conference on Parallel Processing, 2002, pp. 141-148.
4. Barry Rountree and David K. Lowenthal and Shelby Funk and Vincent W. Freeh and Bronis R. de Supinski and Martin Schulz. – Bounding energy consumption in large-scale MPI programs – Proceedings of the ACM/IEEE Conference on High Performance Networking and Computing, SC 2007, November 10-16, 2007, Reno, Nevada, USA.
5. Hlavacs, Helmut and Da Costa, Georges and Pierson, Jean-Marc. – Energy Consumption of Residential and Professional Switches – Rapport de recherche, IRIT, Université Paul Sabatier, Toulouse, 2009.
6. Da-Costa, Georges and Gelas, Jean-Patrick and Georgiou, Yiannis and Lefèvre, Laurent and Orgerie, Anne-Cécile and Pierson, Jean-Marc and Richard, Olivier and Sharma, Kamal. – The GREEN-NET Framework : Energy Efficiency in Large Scale Distributed Systems – HPPAC 2009 : High Performance Power Aware Computing Workshop in conjunction with IPDPS 2009.
7. Clark, Christopher and Fraser, Keir and Hand, Steven and Hansen, Jacob Gorm and Jul, Eric and Limpach, Christian and Pratt, Ian and Warfield, Andrew. – Live migration of virtual machines – NSDI'05 : Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation, 2005, pp. 273-286.
8. P. Degond, M. Delitala. – Modelling and simulation of vehicular traffic jam formation. – Kinetic and Related Models 1 (2008), pp. 279-293.
9. F. Berthelin, P. Degond, V. Le Blanc, S. Moutari, J. Royer, M. Rascle. – A Traffic-Flow Model with Constraints for the Modeling of Traffic Jams. – Mathematical Models and Methods in Applied Sciences 18, Suppl. (2008), pp. 1269-1298.
10. G. Dufour. – Modélisation Multi-Fluides Eulerienne des écoulements diphasiques à inclusions dispersées. – Thèse de doctorat, Université Toulouse III, Décembre 2005.
11. G. Strang. – On the construction and comparison of difference schemes. – SIAM Journal for Numerical Analysis, (5) 507-517, (1968).
12. B. Van Leer. – A second order sequel to Godunov's method. – Journal of Comp. Physics, (32) 101-136, (1979).