Modèles économiques pour le partage des ressources de grilles de calcul

Bernard MIEGEMOLLE*

Directeur(s) de thèse : Thierry MONTEIL

Laboratoire d'accueil :
LAAS - CNRS
7, avenue du Colonel Roche
31077 Toulouse Cedex 4

Etablissement d'inscription :
Institut National des Sciences Appliquées
135, avenue de Rangueil
31077 Toulouse Cedex 4

Résumé

L'objectif principal des grilles de calcul est de regrouper un nombre important de ressources géographiquement distribuées dans le but de répondre à l'augmentation constante des besoins en termes de puissance de calcul des applications scientifiques ou industrielles.
Les utilisateurs de ces ressources et leurs propriétaires ont des objectifs différents qu'il convient de statisfaire au mieux. Dans cette présentation, une approche basée sur les modèles économiques utilisés dans l'économie humaine est présentée. Cette approche permet de gérer la décentralisation inhérente aux grilles de calcul, de s'affranchir des problèmes que posent l'hétérogénéité des ressources, ainsi que de satisfaire les exigences à la fois des propriétaires et des utilisateurs des ressources disponibles sur les grilles.

Mots-clés

Grilles de calcul, Modèles économiques, Placement.

1 INTRODUCTION

1.1 LE GRID COMPUTING

L'augmentation constante des besoins en termes de puissance de calcul informatique a toujours été un défi auquel la communauté informatique s'est confrontée. Même si les évolutions technologiques de ces dernières années ont permis d'aboutir à la création de machines de plus en plus puissantes, celles-ci ne fournissent toujours pas suffisamment de puissance pour effectuer des calculs d'une complexité trop élevée, ou traitant un trop grand nombre de données.

Le calcul parallèle, ou distribué, est une réponse à ce problème. Il s'agit de répartir le calcul sur un ensemble de machines, reliées entre elles par des réseaux rapides, afin d'augmenter les performances pour l'effectuer. Cette idée est à l'origine du calcul sur grille, ou *grid computing*.

Une grille de calcul est une infrastructure matérielle et logicielle qui fournit un accès consistant et peu onéreux à des ressources informatiques [1, 2]. Le but est ainsi de fédérer des ressources provenant de diverses organisations désirant collaborer en vue de faire bénéficier les utilisateurs d'une capacité de calcul et de stockage qu'une seule machine ne peut fournir.

Les grilles sont constituées de ressources fortement hétérogènes, qui sont géographiquement distribuées et qui appartiennent à différentes organisations indépendantes, ayant chacune

*bmiegemo@laas.fr

leurs propres politiques de gestion et de sécurité. Ces ressources sont soumises à une forte dynamique du point de vue de leur charge, mais également de leur disponibilité. Des interfaces standards et des mécanismes doivent être fournis afin de masquer à l'utilisateur la complexité d'un tel système, et de lui offrir un accès transparent à toutes les ressources de la grille.

1.2 MODELES ECONOMIQUES POUR LA GESTION DES RESSOURCES DES GRILLES

Une approche basée sur les modèles économiques du marché réel peut être employée dans le cadre du calcul sur grille [3]. Ces modèles économiques constitueront une métaphore pour la gestion des ressources et le placement d'applications, permettant ainsi de réguler l'offre et la demande au niveau des ressources de la grille tout en garantissant une certaine qualité de service aux clients.

1.2.1 Objectifs des modèles économiques

L'objectif des modèles économiques est de permettre à la fois aux producteurs (les propriétaires des ressources) et aux consommateurs (les utilisateurs des ressources) d'atteindre leurs objectifs [4, 3]. Ils doivent ainsi inciter les producteurs à partager leurs ressources, en leur fournissant une contrepartie, et également inciter les consommateurs à réfléchir aux compromis entre temps de calcul et coût dépendant de la qualité de service désirée.

Ainsi, une approche basée sur les modèles économiques permet de mettre en relation les propriétaires des ressources et leurs utilisateurs potentiels, et place le pouvoir de décision entre leurs mains.

1.2.2 Les différents modèles économiques existants

Il existe cinq principaux modèles économiques, empruntés à l'économie du marché réel, qui peuvent être appliqués à la gestion des ressources des grilles de calcul [5] :

- **Le modèle de marché :** Dans ce modèle, les fournisseurs de ressources fixent un prix, et les utilisateurs sont facturés à ce prix, proportionnellement à la quantité de ressources consommées.
- **Le modèle des négociations :** Ce modèle permet à l'utilisateur, contrairement au précédent, d'agir sur le prix des ressources qu'il désire consommer. Ici, producteurs et consommateurs auront leurs propres objectifs, et devront négocier pour les atteindre.
- **Le modèle de l'appel d'offres :** Le modèle de l'appel d'offres est inspiré des mécanismes mis en place dans le monde des affaires pour gouverner les échanges de biens et de services. Il permet au client de trouver le fournisseur de service approprié pour travailler sur une tâche donnée.
- **Le modèle de la vente aux enchères :** Ce modèle économique permet d'avoir une négociation entre un fournisseur et plusieurs consommateurs potentiels, qui permet d'aboutir à l'établissement d'un prix et au choix d'un consommateur.
- **Le modèle du partage proportionnel des ressources :** Selon ce modèle, une ressource peut être partagée entre plusieurs utilisateurs. La part de ressource allouée à chaque utilisateur par rapport aux autres est alors proportionnelle à son offre par rapport à celles des autres utilisateurs.

Différentes ressources pourront être facturées à leurs utilisateurs, telles que le temps processeur consommé, la quantité de mémoire consommée, la quantité de stockage utilisée, la bande-passante du réseau consommée, les signaux reçus et les changements de contexte, ou bien les logiciels et librairies utilisés.

2 DEFINITION D'UN MODELE ECONOMIQUE

2.1 CARACTERISTIQUES DU MODELE ECONOMIQUE

Le modèle économique que nous allons étudier ici se rapproche du modèle de marché. Le prix à payer, après l'exécution d'une application, sera proportionnel à la quantité de ressources consommées, le prix unitaire étant fixe et connu à l'avance.

Les ressources que nous prendrons en compte dans notre modèle sont celles concernant les processeurs des machines utilisées. Une application devant être exécutée sera tout d'abord découpée en plusieurs tâches. Nous supposerons ces tâches identiques du point de vue des ressources qu'elles consomment. Chaque tâche occupera un et un seul processeur (chaque processeur disponible pourra accueillir plusieurs tâches), pendant un temps donné. Notons que, même si toutes les tâches d'une application sont identiques, le temps pendant lequel elles seront exécutées pourra varier d'une tâche à l'autre en fonction de la puissance des processeurs qu'elles occupent.

Dans un tel contexte, nous proposons de facturer une application selon le temps processeur consommé (somme du temps réel pendant lequel chaque tâche de l'application aura occupé un processeur), le nombre de processeurs utilisés, ainsi que la puissance de chaque processeur utilisé.

Ainsi, selon le coût de chacune de ces ressources, différentes stratégies pourront être adoptées pour minimiser le coût global de l'application. Le problème consistera donc à choisir les machines (leur nombre et leur puissance), de telle sorte que le placement des tâches sur celles-ci offre un bon compromis entre le coût de l'application et sa durée d'exécution. Il s'agit donc d'un problème d'optimisation à variables entières.

2.2 DEFINITION DES VARIABLES

Dans cette section, nous allons présenter les variables permettant de traiter correctement notre problème. Notons qu'elles sont réparties en deux grandes catégories : les données du problème, et les inconnues, ces dernières fournissant l'ordonnancement optimal cherché.

2.2.1 Les données du problème

P^R = puissance du processeur de référence. Lorsque l'utilisateur effectue une requête, il doit renseigner la durée totale de l'application, en prenant pour référence cette puissance.

T^R = temps de référence (temps global de l'application demandé par l'utilisateur).

A = nombre de tâches de l'application.

N^P = nombre de processeurs disponibles.

N^P_{min} et N^P_{max}= bornes imposées par l'utilisateur sur le nombre de processeurs à utiliser.

$P(j)_{1\leq j\leq N^P}$ = puissance de chaque processeur disponible.

$Ch(j)_{1\leq j\leq N^P}$ = charge de chaque processeur.

C^T = coefficient pondérant le coût correspondant au temps processeur consommé.

C^N = coefficient pondérant le coût correspondant au nombre de processeurs utilisés.

C^P = coefficient pondérant le coût correspondant à la puissance des processeurs utilisés.

$C^P_{coef}(j)_{1\leq j\leq N^P}$ = coefficient appliqué sur le coût de chaque processeur. Ce coefficient permet de modifier localement le coût d'un processeur particulier.

2.2.2 Les inconnues

$Taches(i,j)_{1\leq i\leq A, 1\leq j\leq N^P}$ = matrice associant chaque tâche de l'application à un des processeurs disponibles. Le résultat de l'ordonnancement est donc contenu dans cette matrice. Il s'agit ainsi de l'inconnue principale du problème. Toutes les inconnues suivantes sont déduites de cette matrice.

$U(j)_{1\leq j\leq N^P}$ = vecteur indiquant, pour chaque processeur, s'il est utilisé ou non.

N = nombre de processeurs utilisés.

$y(j)_{1 \leq j \leq N^P}$ = variables d'écart entrant en jeu dans le calcul du vecteur U. Elles permettent d'avoir des résultats entiers dans ce vecteur.

$Temps(j)_{1 \leq j \leq N^P}$ = temps cumulé occupé par l'application sur chaque processeur. Il s'agit du temps passé réellement par les tâches sur chaque processeur (correspondant à la différence entre la date de fin de la dernière des tâches et la date de début de la première). Il dépend donc de la puissance de ces derniers, ainsi que de leur charge.

C_{App} = coût de l'application.

T_{App} = durée d'exécution de l'application.

2.2.3 Fonction objectif

La fonction objectif est un compromis entre le coût de l'application et sa durée d'exécution. Ces deux paramètres sont pondérés par des coefficients ajustables selon ce qu'il est souhaité de privilégier. La fonction objectif est donc la suivante :

$Objectif = PoidsCout \times C_{App} + PoidsDuree \times T_{App}$

Ainsi, il faudra calculer le contenu de la matrice $Taches$ de manière à minimiser cette fonction objectif.

2.3 MODELE MATHEMATIQUE

Dans cette section, nous allons établir toutes les équations et contraintes régissant notre modèle économique.

2.3.1 Contraintes sur la matrice $Taches$

$\forall i \in \{1, ..., A\} \qquad \forall j \in \{1, ..., N^P\} \qquad Taches(i,j) \in \{0,1\}$

$\forall i \in \{1, ..., A\} \qquad \sum_{j=1}^{N^P} Taches(i,j) = 1$

Nous avons ainsi spécifié que chaque case de la matrice contient un 0 ou bien un 1. Nous avons également contraint chaque ligne à contenir un et un seul 1 (chaque tâche est placée sur un et un seul processeur).

2.3.2 Calcul du vecteur U

$\forall j \in \{1, ..., N^P\} \qquad U(j) \in \{0,1\}$

$\forall j \in \{1, ..., N^P\} \qquad U(j) = \Big[\big(\sum_{i=1}^{A} Taches(i,j) \big) + y(j) \Big] / A$

$\forall j \in \{1, ..., N^P\} \qquad y(j) \in \mathbb{N} \qquad 0 \leq y(j) \leq A - 1$

$U(j)$ est un entier égal à 1 si la colonne j de la matrice $Taches$ contient au moins un 1, ou égal à 0 sinon. Ainsi, nous introduisons une variable d'écart $y(j)$ dans le calcul de $U(j)$. Cette variable sera ajustée de telle sorte que $U(j)$ soit entier. On aura ainsi :

- $y(j) = 0$ si $\sum_{i=1}^{A} Taches(i,j) = 0 \qquad (\Rightarrow U(j) = 0)$,
- $y(j) = A - \sum_{i=1}^{A} Taches(i,j)$ sinon $\qquad (\Rightarrow U(j) = 1)$.

2.3.3 Calcul de N

$N = \sum_{j=1}^{N^P} U(j)$

$0 \leq N_{min}^P \leq N \leq N_{max}^P \leq N^P$

$N_{max}^P \leq A$

Nous voyons ainsi que plusieurs contraintes sur le nombre de processeurs utilisés entrent en jeu. Ce nombre ne doit pas franchir les bornes fixées par l'utilisateur. D'autre part, ces bornes doivent respecter les limites physiques du système. De plus, l'utilisateur ne peut pas demander plus de processeurs que ce qu'il y a de tâches à exécuter, puisque, dans ce cas-là, des processeurs ne seraient pas utilisés.

2.3.4 Calcul du vecteur $Temps$

Soit T le temps processeur consommé par une tâche sur un processeur de puissance P, on a :

$T \cdot P = \frac{T^R.P^R}{A} \qquad \Rightarrow \qquad T = \frac{T^R.P^R}{A} \cdot \frac{1}{P}$

Soit T_i le temps processeur consommé par la tâche i (lorsqu'elle est active), on a :

$\forall i \in \{1, ..., A\} \qquad T_i = \frac{T^R.P^R}{A} \cdot \sum_{j=1}^{N^P} \frac{Taches(i,j)}{P(j)}$

Soit T_i^* le temps occupé par la tâche i sur un processeur (qu'elle soit active ou non), on a :

$\forall i \in \{1, ..., A\} \qquad T_i^* = \frac{T^R.P^R}{A} \cdot \sum_{j=1}^{N^P} \frac{Taches(i,j)}{P(j)\cdot\left(1-Ch(j)\right)}$

On en déduit le temps pendant lequel l'application occupera ainsi chaque processeur (que les tâches soient actives ou non) :

$\forall j \in \{1, ..., N^P\} \qquad Temps(j) = \frac{T^R.P^R}{A} \cdot \sum_{i=1}^{A} \frac{Taches(i,j)}{P(j)\cdot\left(1-Ch(j)\right)}$

2.3.5 Calcul du coût de l'application

$C_{App} = C^T \cdot \frac{T^R.P^R}{A} \cdot \sum_{i=1}^{A} \sum_{j=1}^{N^P} \frac{Taches(i,j)}{P(j)} + C^N \cdot N + C^P \cdot \sum_{i=1}^{A} \sum_{j=1}^{N^P} Taches(i,j) \cdot P(j) \cdot C_{coeff}^P(j)$

Le coût de l'application dépend du temps processeur consommé par chaque tâche, du nombre de processeurs utilisés, ainsi que de leur puissance.

2.3.6 Calcul de la durée d'exécution de l'application

$T_{App} = \max_{j=1,...,N^P} Temps(j)$

Nous supposerons que les temps de communication et de synchronisation entre les tâches parallèles sont négligeables. Ceci est possible dans le cadre d'applications " gros grains ", c'est-à-dire des applications dont les tâches calculent beaucoup et communiquent peu. Dans un tel contexte, la durée d'exécution de l'application correspond au maximum des temps qu'elle occupe sur chacun des processeurs.

3 RESULTATS DE SIMULATIONS

Nous pouvons maintenant tester le comportement du modèle économique défini précédemment. Nous avons développé à ce propos un algorithme génétique, dont nous comparerons les résultats avec ceux donnés par le logiciel *Xpress*, un outil de modélisation et d'optimisation numérique, dont le but est de trouver la solution optimale d'un problème linéaire.

Nous traiterons ici l'exemple d'une application de vingt tâches ($A = 20$) à placer sur trente processeurs ($N^P = 30$) de charge initiale nulle ($Ch(j)_{1 \leq j \leq N^P} = 0$). La puissance de chacun des processeurs suit la loi suivante : $P(j)_{1 \leq j \leq N^P} = 100 \times j$.

La puissance du processeur de référence est de 100 ($P^R = 100$). Le temps global de l'application demandé par l'utilisateur sur le processeur de référence est de 200000 ($T^R = 200000$). Notons enfin que l'utilisateur demande à utiliser entre 4 et 20 processeurs ($N_{min}^P = 4$ et $N_{max}^P = 20$).

Dans ces conditions, nous allons tenter de minimiser un compromis entre le coût de l'application et sa durée d'exécution. Nous fixerons les différents paramètres du modèle de la manière suivante :

- $C^T = 1$, $C^N = 1$, $C^P = 50$ et $C_{coef}^P(j)_{1 \leq j \leq N^P} = 1$ afin de privilégier le coût relatif à la puissance des processeurs utilisés.
- $PoidsCout = 1$ et $PoidsDuree = 500$ afin que le coût et la durée d'exécution aient une influence comparable sur le placement (ces valeurs peuvent être automatiquement

calculées à partir du coût optimal lorsque le temps n'est pas pris en compte, et du temps d'exécution optimal lorsque le coût est ignoré).

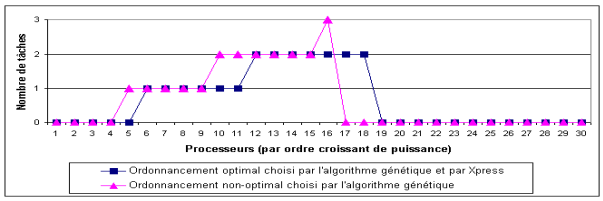

FIG. 1: Minimisation d'un compromis entre le coût d'une application et sa durée d'exécution

La figure 1 montre plusieurs ordonnancements. Le premier est l'ordonnancement optimal, retourné à la fois par l'algorithme génétique et par Xpress. Nous pouvons constater que les processeurs choisis se situent dans un domaine de puissances intermédiaires, permettant d'avoir une certaine rapidité d'exécution pour un coût assez faible. Il s'agit du meilleur compromis possible.

Cet ordonnancement optimal n'est pas le seul que retourne l'algorithme génétique. En réalité, trois ordonnancements différents sont retournés dans 95% des cas. Les deux autres ordonnancements sont toutefois très proches de la solution optimale (environ 1% en termes de valeur du critère). La figure 1 montre, parmi ces deux ordonnancements, celui qui est le plus souvent obtenu.

4 CONCLUSION

Les modèles économiques empruntés au marché réel peuvent être appliqués à la gestion des ressources des grilles de calcul, en permettant à leurs propriétaires et à leurs utilisateurs de satisfaire leurs objectifs respectifs.

Nous avons présenté un modèle économique particulier, et montré les résultats que donne son implémentation par un algorithme génétique. Ces résultats sont concluants, et montrent qu'une approche économique peut être utilisée pour la gestion des ressources d'une grille.

Ce modèle peut être étendu à un modèle dynamique, pour lequel la charge des processeurs ainsi que leur coût varient au cours du temps. Le problème revient ainsi à déterminer, en plus du placement, la date optimale du démarrage de l'application. Les premiers résulats obtenus montrent que la démarche statique décrite ici se généralise bien au cas dynamique.

Références

[1] I. Foster and C. Kesselman, *The Grid : Blueprint for a New Computing Infrastructure*, San Francisco, Morgan Kauffman, 1999.

[2] I. Foster, C. Kesselman and S. Tueke, *The Anatomy of the Grid : Enabling Scalable Virtual Organizations*, Intl. J. Supercomputer Appl., 2001.

[3] R. Buyya, *Economic-based Distributed Resource Management and Scheduling for Grid Computing*, PhD Thesis, Monash University, 2002.

[4] D. Ferguson, C. Nikolaou, J. Sairamesh and Y. Yemini, *Economic Models for Allocating Resources in Computer Systems*, In Scott Clearwater, editor, Market-Based Control : A Paradigm for Distributed Resource Allocation, Scott Clearwater. World Scientific, Hong Kong, 1996.

[5] R. Buyya, H. Stockinger, J. Giddy and D. Abramson, *Economic Models for Management of Resources in Peer-to-Peer and Grid Computing*, SPIE International Conference on Commercial Applications for High-Performance Computing, August 20-24, 2001, Denver, USA.