Modèle d'Équilibrage de Charge pour les Grilles de Calcul

YAGOUBI Belabbas

Université d'Oran (Es-sénia),
Faculté des Sciences,
Département d'Informatique,
Campus Professeur Taleb Mourad, Oran, ALGÉRIE.
byagoubi@yahoo.fr

RÉSUMÉ. Afin d'obtenir de meilleures performances dans les systèmes répartis, le problème d'équilibrage de charge a été intensivement étudié ces dernières années. La plupart des travaux existants se sont intéressés à des systèmes qui sont plus ou moins homogènes et trouvent quelques difficultés à s'adapter aux caractéristiques des nouvelles infrastructures telles que les *grilles de calcul*, qui présentent un degré d'hétérogénéité assez élevé. Pour cela, il faut soit adapter, soit définir de nouvelles stratégies d'équilibrage pour ces infrastructures.
Dans cette perspective, nous proposons un modèle arborescent de représentation d'une grille de calcul, sur lequel nous développons une stratégie hiérarchique d'équilibrage de charge. Les caractéristiques principales de la stratégie proposée peuvent être résumées comme suit: (i) C'est une stratégie d'équilibrage au niveau des tâches; (ii) Elle favorise un transfert local de tâches dans le but de réduire les coûts de communication; (iii) C'est une stratégie distribuée avec prise de décision locale.

ABSTRACT. In order to get a better performance in distributed systems, load balancing problem has been extensively studied in recent years. Most of existing works focus on traditional systems where resources are generally homogeneous, like clusters. For grid infrastructures, this assumption is not totally true because resources of a grid are highly heterogeneous. Hence, load balancing problem for *grid computing* is a new challenge for scientists.
In this paper, we propose a tree-based representation model for grid computing, over which we develop a hierarchical load balancing strategy. The main characteristics of this strategy can be summarized as follows:(i) It uses a task-level load balancing; (ii) It privileges local tasks transfer to reduce communication costs; (iii) It is a distributed strategy with local decision making.

MOTS-CLÉS : Grilles de calcul, Équilibrage de charge, Modèle arborescent, Charge de travail.

KEYWORDS : Grid computing, Load balancing, Tree based model, Workload.

1. Introduction

Les systèmes répartis ont connu ces dernières années un essor assez particulier grâce au potentiel de puissance de traitement qu'ils peuvent offrir, au développement des vitesses des réseaux et au faible coût de leur mise en œuvre. D'un autre côté, nous assistons à l'émergence d'applications de plus en plus exigeantes en ressources de calcul et de stockage et pour lesquelles les architectures actuelles s'avèrent insuffisantes. Parmi ces applications, nous pouvons citer celles liées à la météorologie, au calcul scientifique intensif, au datamining et à la bioinformatique [10]. Pour répondre à de telles exigences, une solution consiste à agréger les ressources informatiques disséminées à l'échelle planétaire en les interconnectant grâce aux infrastructures réseaux existantes et en particulier Internet. C'est cette idée qui a permis l'émergence d'un nouveau concept de calcul appelé *calcul de grille* (*Grid computing*) [12]. Les architectures qui peuvent supporter ce type de calcul peuvent être classées selon différents points de vue. Si l'on se base sur le point de vue objectif, nous pouvons distinguer trois types de grilles : les *grilles de calcul* (agrégation de la puissance de calcul), les *grilles de données* (stockage à large échelle) et les *grilles d'information* (partage de la connaissance). Si par contre l'on se place du point de vue infrastructure, nous pouvons classer les grilles en *grilles de calcul*, *grilles de PC* (desktop grid) et *systèmes P2P*. Indépendamment de ces différentes classifications, les infrastructures de type grille se caractérisent par trois principaux éléments [3] :

- *Dynamicité* : Dans une grille, le nombre de ressources évolue de manière dynamique (connexion et déconnexion de ressources), ce qui rend leur gestion plus complexe que dans des systèmes stables. D'un point de vue gestion, il faut donc s'adapter à ces fluctuations qui peuvent avoir un effet négatif sur les performances des applications.

- *Hétérogénéité* : Plusieurs niveaux d'hétérogénéité existent dans une grille : matériel, systèmes d'exploitation, logiciels d'applications, réseaux, etc. Cette hétérogénéité doit évidemment être transparente par rapport aux utilisateurs et à leurs applications.

- *Domaines administratifs multiples* : Les ressources d'une grille de calcul sont géographiquement distribuées et gérées à travers plusieurs domaines administratifs qui appartiennent à différentes organisations. Ces domaines peuvent avoir des politiques de gestion et de sécurité qui peuvent être très différentes les unes par rapport aux autres.

La mise en œuvre d'infrastructures à large échelle peut se faire de manière relativement simple, dans la mesure où l'idée sous-jacente est de mettre en commun les infrastructures actuelles afin de construire un *super calculateur virtuel*. Ainsi, il est possible de construire une grille à partir de ressources existantes (PC, stations de travail, machines parallèles, etc.) en utilisant des réseaux d'interconnexion existants, et notamment l'Internet. Il est aussi possible d'utiliser une approche communautaire et volontariste dans laquelle chaque individu ou institution accepte de mettre ses ressources à la disposition de ceux qui en ont besoin et ce sans contraintes spécifiques [5].

La gestion de telles infrastructures pose naturellement des problèmes beaucoup plus complexes que dans le cas de systèmes répartis classiques. Parmi ces problèmes, l'optimisation de l'utilisation des ressources de calcul et de communication est un aspect très important qui mobilise beaucoup de chercheurs. En fonction des objectifs visés, cette optimisation peut être vue selon différents points de vue [2, 21] :

– Du point de vue de l'utilisateur, il s'agit notamment de minimiser le temps d'exécution global (makespan) de son application.

– Du point de vue de l'administrateur du système, l'objectif visé consiste à optimiser les performances globales du système, en maximisant le taux d'exploitation des machines.

Nous pensons que, si le critère *makespan* est un critère de performance important pour les systèmes parallèles, l'exploitation optimale des grilles de calcul peut se mesurer avec deux autres métriques : un équilibrage de charge entre les ressources d'une grille et une réduction des coûts de communication. Dans cette perspective, nous proposons, dans ce papier, un modèle arborescent d'équilibrage de charge, sur lequel nous développons une stratégie d'équilibrage qui puisse répondre aux deux objectifs suivants : (i) la réduction, autant que possible, du temps de réponse moyen des tâches soumises à une grille de calcul ; et, (ii) la réduction des coûts de communication en privilégiant une redistribution de la charge de calcul au sein d'une grappe plutôt qu'au niveau de la grille toute entière.

Le reste du papier est organisé comme suit : la Section 2 rappelle les principaux aspects du problème de l'équilibrage de charge. La Section 3 passe en revue les travaux existants relatifs à l'équilibrage de charge notamment pour les systèmes à large échelle. La Section 4 décrit le modèle arborescent que nous proposons pour représenter une grille de calcul. La stratégie d'équilibrage de charge et les algorithmes associés sont décrits, respectivement, dans les Sections 5 et 6. La Section 7 présente quelques résultats relatifs à l'expérimentation de la stratégie d'équilibrage proposée. Finalement, la Section 8 conclut cet article et liste quelques perspectives de recherche.

2. Problème de l'équilibrage de charge

Le problème de l'équilibrage étant un problème relativement ancien, beaucoup d'approches ont été proposées pour le résoudre. Casavant et Kuhl [9] ont défini une taxonomie largement adoptée par la communauté scientifique dont les principales classes sont :

1) *Approche statique Vs. approche dynamique* : Dans une approche statique, les tâches sont assignées aux machines avant l'exécution de l'application qui les contient. Les informations concernant le temps d'exécution des tâches et les caractéristiques dynamiques des machines sont supposées connues a priori. Cette approche est efficace et simple à mettre en œuvre lorsque la charge de travail est au préalable suffisamment bien caractérisée.
   Dans une approche dynamique, l'assignation des tâches aux machines se décide durant la phase d'exécution, en fonction des informations qui sont collectées sur l'état de charge du système. Ceci permet d'améliorer les performances d'exécution des tâches mais au prix d'une complexité dans la mise en œuvre de cette stratégie, notamment en ce qui concerne la définition de l'état de charge du système, qui doit se faire de manière continue.
2) *Approche centralisée Vs. approche distribuée* : Dans une approche centralisée, un site du système est choisi comme *coordinateur*. Il reçoit les informations de charge de tous les autres sites qu'il assemble pour obtenir l'état de charge global du système.
   Dans le cas d'une approche distribuée, chaque site du système est responsable de collecter les informations de charge sur les autres sites et de les rassembler pour obtenir l'état global du système. Les décisions de placement de tâches sont prises localement, étant donné que tous les sites ont la même perception de la charge globale du système.
3) *Approche source-initiative Vs. receveur-initiative* [20] : L'approche *source-initiative* est appliquée lorsqu'un site, appelé *source*, détecte qu'il a une surcharge de travail et qu'il cherche à transférer le surplus vers un site faiblement chargé. L'approche *receveur-initiative* s'applique lorsqu'un site faiblement chargé, appelé *receveur*, demande à recevoir tout ou partie du surplus des sites surchargés.

2.1. Politiques et mécanismes d'équilibrage de charge

Un système d'équilibrage de charge est composé de deux éléments essentiels : les *politiques* et les *mécanismes* [11]. Les politiques considèrent l'ensemble des choix à effectuer pour distribuer une charge de travail alors que les mécanismes réalisent physiquement la répartition de la charge et fournissent les informations exigées par les politiques. La figure 1 illustre la décomposition arborescente d'un système d'équilibrage de charge.

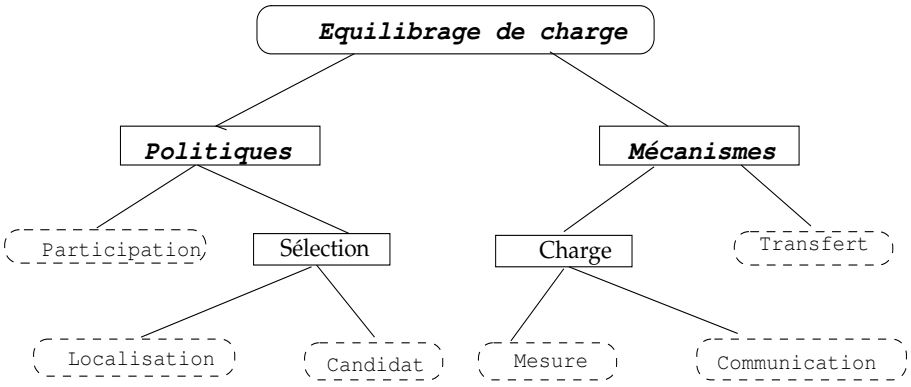


Figure 1. *Composants d'un système d'équilibrage de charge*

– *Politique de participation* : Le but de cette politique consiste à déterminer si un site est dans un état approprié pour participer à un transfert de tâches comme *source* (site surchargé) ou comme *receveur* (site sous-chargé).

– *Politique de sélection de la localisation* : Cette politique est responsable de trouver, pour un site donné, un partenaire (source ou receveur), une fois que la politique de participation a décidé que ce site était soit source, soit receveur.

– *Politique de sélection des tâches à transférer* : Une fois que les politiques de participation et de localisation ont décidé qu'un site S_i est source et qu'un autre site S_j est receveur, cette politique est responsable du choix des tâches à transférer de S_i vers S_j.

– *Mécanisme de mesure de la charge* : Dans toute approche d'équilibrage de charge, une des difficultés majeures est celle qui consiste à évaluer la mesure de la charge d'un site. Dans la plupart des travaux existants, c'est la longueur de la file d'attente qui détermine la charge d'un site. Certains auteurs [17, 21] préconisent comme indicateur de charge, une combinaison entre la longueur de la file d'attente CPU, celle des Entrées/Sorties et l'occupation mémoire. Dans le cas des grilles de calcul, il est nécessaire de tenir compte aussi de l'hétérogénéité des ressources et des réseaux de communication pour mesurer la charge d'un site.

– *Mécanisme de définition de la charge* : Ce mécanisme essaie de définir la charge globale d'un système en collectant les informations de charge (partielles) sur l'ensemble ou une partie des sites du système. Il faudra alors définir les méthodes selon lesquelles l'information de charge est collectée puis diffusée aux sites.

2.2. Problématiques particulières liées aux grilles de calcul

Les spécificités des grilles de calcul font que le problème d'équilibrage de charge devient beaucoup plus complexe que dans le cas des systèmes répartis classiques ou dans

le cas de grappes (clusters), qui peuvent offrir une certaine *homogénéité* et *stabilité* des ressources qui les composent. Par contre, dans des systèmes à large échelle comme les grilles de calcul, ces hypothèses ne sont pas tout à fait réalistes [17]. Les grilles de calcul sont susceptibles d'être largement distribuées et fortement hétérogènes. Par conséquent, il est essentiel de considérer l'impact de ces caractéristiques dans la conception et l'analyse de techniques d'équilibrage de charge. La plus grande difficulté réside dans l'estimation de la charge globale de toute la grille. Ainsi, tout système d'équilibrage de charge d'une grille devra, en premier lieu, permettre d'estimer l'état de charge de chaque ressource de cette grille. Il s'agira notamment de préciser [14] :

– Quels critères retenir pour la définition de la charge d'une ressource ?

– Comment mesurer cette charge ?

– Comment intégrer toutes les informations liées à l'hétérogénéité des ressources pour obtenir une moyenne représentative de la charge instantanée de tout le système ?

3. Travaux connexes

Une étude bibliographique relative aux travaux de recherche sur l'équilibrage de charge nous a permis de distinguer deux niveaux sur lesquels il serait possible de définir une stratégie d'équilibrage de charge : *application* et *système* [1, 4, 19].

– L'équilibrage au niveau application consiste à adapter les ressources du système aux caractéristiques particulières d'une application donnée, dans l'objectif de minimiser son makespan, sans se soucier de l'exécution des autres applications dans le système.

– L'équilibrage au niveau système, appelé aussi ordonnancement distribué, consiste à maximiser les performances globales du système, ce qui revient à minimiser le temps de réponse global des applications, en affectant convenablement les tâches aux différentes ressources d'un système.

La plupart des approches au niveau application [16] effectuent le partitionnement d'une application en utilisant des techniques issues de la théorie des graphes. Cependant, elles négligent le coût de migration qui peut être parfois très prohibitif. Pour réduire un tel coût, certains travaux [20] ont proposé un algorithme tolérant de latence qui tente de tirer profit, au niveau d'une machine, du recouvrement entre le calcul et les communications. Malheureusement, sa mise en œuvre est limitée car les applications doivent, elles-mêmes, trouver un parallélisme entre le traitement et la migration des données. Des approches d'équilibrage utilisant la technologie agent ont été également proposées pour des clusters de machines [8, 18]. Dans [13], Genaud et al. améliorent la primitive MPI_Scatterv pour supporter un équilibrage maître-esclave par l'optimisation de la distribution de calcul et de données en utilisant un algorithme de programmation linéaire. Malgré les bons résultats que procure une telle approche, elle reste néanmoins limitée à un équilibrage statique. Hu et al. [15] proposent un algorithme optimal de transfert de données à travers le calcul d'un multiplicateur de Lagrange de la forme Euclidienne d'un volume de données. Les résultats de l'application de cet algorithme ont montré que le transfert de données peut être minimisé efficacement dans le cas d'environnements homogènes. Par contre, cet algorithme ne considère pas l'hétérogénéité du réseau qui peut avoir des incidences négatives sur les performances d'une application. De plus, il est difficile de tenir compte du transfert de charge parce que les performances d'un réseau WAN sont dynamiques en termes d'exécution, d'instabilité, etc.

L'objectif traditionnel d'une approche d'équilibrage de charge orientée système consiste à minimiser le temps d'exécution global des applications. Ce problème est réputé, comme étant un problème NP-complet [21]. Dans le cas de systèmes à large échelle, tels que les grilles, une minimisation absolue du temps d'exécution global n'est pas obligatoirement le seul objectif. Nous pensons qu'il existe deux autres facteurs aussi importants que le makespan : le premier concerne l'utilisation rationnelle de toutes les ressources d'une grille (objectif d'équilibrage) ; le second est relatif à l'aspect communication lors du transfert de tâches (objectif de minimisation des coûts de communication).
Dans cette perspective, nous proposons, dans ce papier, une stratégie d'équilibrage de charge pour les grilles de calcul qui tente de répondre à ces deux objectifs. Comparativement aux travaux existants, notre stratégie se caractérise par les éléments suivants : (i) C'est une stratégie d'équilibrage au niveau des tâches ; (ii) Elle favorise un transfert local de tâches (intra-grappe) pour éviter un surcoût de communication ; (iii) C'est une stratégie distribuée, dans la mesure où plusieurs opérations d'équilibrage peuvent se faire en parallèle (équilibrage au niveau de chaque grappe) ; (iv) La décision d'effectuer une opération d'équilibrage de charge, pour une grappe, se fera uniquement sur la base des informations de charge locales à cette grappe.

4. Modèle arborescent d'équilibrage de charge

4.1. Topologie générale d'une grille

D'un point de vue topologique (voir figure 2), Berstis et al [6] répertorient les grilles en trois niveaux par ordre croissant d'étendue géographique et de complexité : *Intra-grappe*, *Extra-grappe* et *Inter-grappes*.

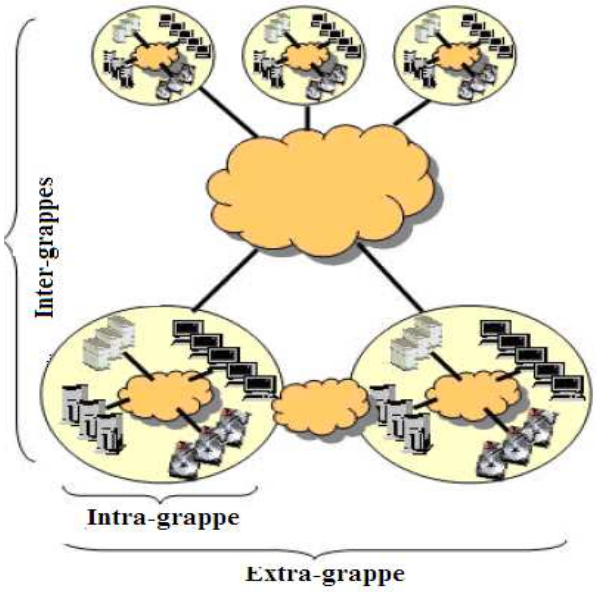


Figure 2. *Topologie générale d'une grille de calcul*

– *Intra-grappe* (par analogie à Intranet) : Cette grille est composée d'un ensemble relativement simple de ressources (Éléments de calcul et Éléments de stockage) appartenant à une organisation unique. Les principales caractéristiques d'une telle topologie sont la présence d'un réseau d'interconnexion performant et haut-débit et d'un ensemble relativement statique et homogène de ressources.

– *Extra-grappe* (par analogie à Extranet) : Ce type de grille étend la topologie précédente en agrégeant plusieurs grappes. Une extra-grappe est caractérisée par la présence d'un réseau d'interconnexion hétérogène haut et bas débit (LAN/WAN) et d'un ensemble plus ou moins dynamique de ressources.

– *Inter-grappes* (par analogie à Internet) : Cette topologie consiste à fédérer les grilles de multiples organisations en une seule *grille* générale. Ses principales caractéristiques sont la présence d'un réseau d'interconnexion très hétérogène haut et bas débit (LAN/WAN) et d'un ensemble dynamique et fortement hétérogène de ressources.

4.2. Modèle de représentation proposé

Pour représenter une grille nous proposons de la transformer, de manière univoque, en un arbre d'interconnexion virtuel. Tel qu'illustré par la figure 3, cet arbre peut être instancié en trois configurations notées, respectivement : **G/C/M**, **1/C/M** et **1/1/M**, où **G** représente le nombre d'*extra-grappes*, **C** le nombre d'*intra-grappes* et **M** le nombre total d'*éléments de calcul (**EC**'s)* d'une grille.

4.2.1. Modèle 1/1/M

Cette instance du modèle représente la plus petite grille possible, à savoir une seule grappe composé de M éléments de calcul. Le schéma relatif à cette instance est le modèle $1/1/M$ de la figure 3 qui comprend deux niveaux définis comme suit :

1) Le niveau racine de cet arbre correspond à l'unique grappe de la grille. Appelée *Gestionnaire des EC's*, cette racine a pour rôle de :
   - Gérer l'information de charge relative aux EC's de la grappe.
   - Maintenir l'état de charge de la grappe.
   - Décider de déclencher un équilibrage local, que nous appellerons *équilibrage intra-grappe*.
   - Informer les EC's, pour mettre en œuvre l'équilibrage décidé par le gestionnaire.
2) Le niveau feuilles de l'arbre, où chaque feuille correspond à un EC ayant pour fonction de :
   - Maintenir à jour l'information de l'état de charge de l'EC correspondant.
   - Envoyer périodiquement cette information à son gestionnaire.
   - Exécuter les opérations d'équilibrage décidées par son gestionnaire.

4.2.2. Modèle 1/C/M

Ce modèle représente une extra-grappe. C'est une extension du modèle précédent, dans le sens où l'on passe d'une seule grappe à C grappes. Nous obtenons ainsi le modèle $1/C/M$ représenté par un arbre à trois niveaux défini comme suit :

1) Le niveau racine, appelé *Gestionnaire des grappes*, a pour fonctions de :
   - Gérer l'information de charge relative à chaque grappe sous son contrôle.
   - Maintenir l'état de charge de l'extra-grappe.

- Décider d'un équilibrage local au niveau d'une grappe, que nous appellerons *équilibrage extra-grappe*.

- Envoyer les décisions d'équilibrage aux gestionnaires des EC's associés.

2) Le deuxième niveau est associé aux grappes de l'extra-grappe. Chaque nœud de ce niveau joue le même rôle que dans le modèle $1/1/M$.

3) Le troisième niveau correspond aux éléments de calcul (feuilles de l'arbre) tel que défini dans le modèle précédent.

4.2.3. Modèle G/C/M

C'est le modèle générique qui correspond à une grille (inter-grappes). L'arbre correspondant a quatre niveaux représentant l'agrégation de G modèles de type $1/C/M$, que l'on peut définir comme suit :

1) La racine de cet arbre, appelée *Gestionnaire de grille*, a pour fonctions de :

- Maintenir l'information de charge de l'ensemble de la grille.

- Décider d'un équilibrage global entre les extra-grappes de la grille, que nous appellerons *équilibrage inter-grappes*.

- Envoyer les décisions d'équilibrage aux gestionnaires de grappes pour exécution .

2) Les trois autres niveaux sont identiques à ceux du modèle $1/C/M$.

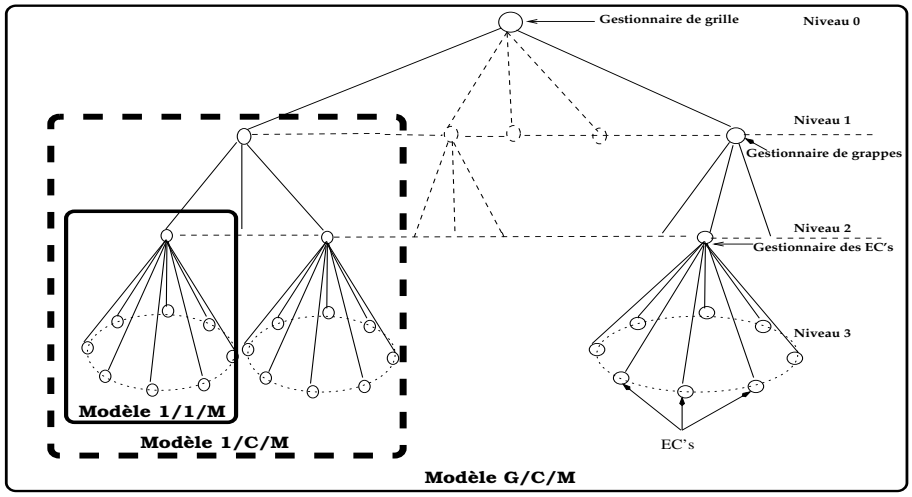


Figure 3. *Modèle générique de représentation d'une grille*

Remarque : Le nombre d'éléments de calcul par grappe ainsi que leurs caractéristiques (en particulier leurs vitesses) peuvent changer d'une grappe à l'autre. Il en est de même pour le nombre de grappes par extra-grappe.

4.2.4. Caractéristiques du modèle proposé

Le modèle de représentation défini ci-dessus peut être caractérisé comme suit :

1) La modélisation d'une grille en arbre s'effectue par une transformation univoque. A chaque grille correspond un et un seul arbre de représentation et ce quelque soit la complexité topologique de la grille. Nous pouvons considérer cet arbre comme un arbre de recouvrement d'une grille, qui nous permettra de définir une stratégie d'équilibrage.

2) Le modèle arborescent proposé supporte le passage à l'échelle en terme de ressources (ajout et retrait d'un élément de calcul, d'une grappe ou d'une extra-grappe).
3) La structure hiérarchique du modèle facilite les flux d'informations à travers les nœuds de l'arbre. En terme de flux d'informations, nous distinguons trois types :

- *Flux montant* : Ce flux concerne la circulation des informations de charge pour définir un état de charge à différents niveaux (intra/extra/inter-grappe(s)).
- *Flux horizontal* : Il concerne les informations nécessaires à l'exécution des opérations d'équilibrage de charge.
- *Flux descendant* : Ce flux permet de véhiculer les décisions d'équilibrage prises par les gestionnaires correspondants aux différents niveaux du modèle.

5. Stratégie d'équilibrage de charge

5.1. Principes

La structure arborescente du modèle proposé nous permet de développer une stratégie hiérarchique à trois niveaux d'équilibrage : *Intra-grappe*, *Extra-grappe* et *Inter-grappes*.

1) ***Équilibrage Intra-grappe*** : Dans ce premier niveau, chaque gestionnaire d'éléments de calcul décide de déclencher une opération d'équilibrage en fonction de la charge courante de la grappe qu'il gère. Cette charge est estimée à partir des différentes informations de charge envoyées périodiquement par les EC's qui composent la grappe. Le gestionnaire tente, en priorité, d'équilibrer la charge de la grappe localement en la répartissant entre les EC's qui lui appartiennent. Cette approche de localité a pour objectif de réduire les coûts de communication, en évitant les communications extra/inter grappes.
2) ***Équilibrage Extra-grappe*** : Dans ce deuxième niveau, l'équilibrage se fait à l'échelle des extra-grappes. Il intervient dans le cas où certains gestionnaires des EC's n'ont pas réussi à équilibrer localement leurs charges. Il y aura ainsi transfert de tâches entre grappes surchargées et grappes sous-chargées de la même extra-grappe. Dans un souci de réduire au maximum les coûts de communication, les grappes réceptrices seront sélectionnées en fonction des débits des réseaux.
3) ***Équilibrage Inter-grappes*** : Dans ce troisième niveau, l'équilibrage de charge n'est déclenché que si un ou plusieurs gestionnaires de grappes n'arrivent pas équilibrer leurs charges localement entre les grappes qu'ils gèrent. Dans ce cas extrême, il sera alors nécessaire au gestionnaire de grille de transférer un certain nombre de tâches à partir d'extra-grappes surchargées vers d'autres qui sont sous-chargées.

5.2. Description générique de la stratégie d'équilibrage

A n'importe quel niveau du modèle défini ci-dessus, nous proposons une stratégie d'équilibrage composée de trois étapes. Comme la description de cette stratégie se fera de manière générique, nous utiliserons les notions de *groupe* et d'*élément*. Un groupe peut désigner, selon les cas, soit une intra-grappe, une extra-grappe ou toute la grille. Un élément est un composant d'un groupe (un EC, une grappe ou une extra-grappe).

1) ***Estimation de la charge du groupe :*** Cette étape définit les mécanismes de mesure et de communication de charge. Connaissant le nombre de ses éléments ainsi que leurs capacités respectives, chaque gestionnaire estime les capacités du groupe auquel il est associé

en effectuant les actions suivantes :

- Estimation de la charge courante du groupe, sur la base des informations reçues périodiquement à partir de ses éléments.

- Calcul de l'écart type sur les charges de travail des éléments dans le but de mesurer l'étendue des variations de charge entre un groupe et ses éléments.
Afin de prendre en considération l'hétérogénéité des EC's, nous proposons comme indice de charge, le *temps d'exécution* [1] noté $TEX = \frac{LOD}{SPD}$.

- Envoi de l'information de charge à son gestionnaire associé.

2) ***Prise de décision*** : Durant cette étape, le gestionnaire du groupe décide de l'opportunité de déclencher un équilibrage de charge local. Pour cela, il exécute les actions suivantes :

- *Définition de l'état de charge d'un groupe* : Sachant que l'écart type σ mesure la variation moyenne entre le temps d'exécution des éléments et celui de leur groupe associé, nous pouvons dire qu'un groupe est en état d'équilibre lorsque cet écart est relativement faible. Cela signifie que le temps d'exécution de chaque élément converge vers le temps d'exécution de son groupe.

  État d'équilibre : En pratique il s'agit de définir un *seuil d'équilibre*, noté ε, à partir duquel nous pouvons dire que l'écart type σ tend vers zéro et donc le groupe est en état d'équilibre. Ainsi nous pouvons écrire :
  Si ($\sigma \leq \varepsilon$) **Alors** le groupe est équilibré **sinon** le groupe est en état de déséquilibre.

  État de saturation : Un groupe peut être déséquilibré tout en étant saturé. Dans ce cas précis, il n'est pas utile d'entamer un équilibrage local, puisque le groupe restera surchargé. Pour mesurer la saturation d'un groupe, nous définissons un autre seuil, noté δ, que nous appellerons *seuil de saturation*.

- *Partitionnement du groupe* : Lorsqu'un groupe non saturé est déséquilibré, nous pouvons envisager le déclenchement d'une opération d'équilibrage de charge. Pour déterminer si un élément d'un groupe est dans un état approprié pour participer à un transfert de tâches comme *source* ou comme *receveur*, nous partitionnons le groupe en trois classes d'éléments : les éléments surchargés (sources), les éléments équilibrés (neutres) et les éléments sous-chargés (receveurs). Cette classification dépend de l'écart entre l'indice de charge de chaque élément et celui de son groupe.

3) ***Transfert de tâches*** : Pour réaliser une opération d'équilibrage de charge, nous proposons l'heuristique suivante :

a) Calculer la disponibilité en terme de capacité de calcul, qui correspond à la charge totale offerte par les éléments receveurs.

b) Calculer la demande, i.e., la charge totale requise par l'ensemble des éléments sources.

c) Si l'offre n'est pas en mesure de satisfaire suffisamment la demande (écart trop grand), il n'est pas recommandé d'entamer un équilibrage local. Pour mesurer l'offre par rapport à la demande, nous définissons un seuil noté ρ, que nous appellerons *seuil d'espérance*.
Si ($\frac{OFFRE}{DEMANDE} > \rho$) **Alors** Équilibrage local **Sinon** Équilibrage au niveau supérieur.

d) Effectuer un transfert de charge en tenant compte des coûts de communication.

---

1. Le temps d'exécution d'une entité (groupe ou élément) est le rapport entre la charge (LOD exprimée en nombre d'unités de calcul) et la vitesse (SPD exprimée en nombre d'unités de calcul exécutées par unité de temps) de cette entité.

Pour la sélection des tâches à transférer, nous pouvons utiliser l'un des critères suivants :
- *Plus petit temps d'exécution* : Donner la priorité de transfert à la tâche qui dispose du plus petit temps d'exécution.
- *Plus grand temps d'exécution* : Priorité à la tâche ayant le plus grand temps d'exécution.
- *FIFO* : Transférer la tâche la plus âgée.
- *LIFO* : Transférer la tâche la plus jeune.
- *Aléatoire* : Choix aléatoire.

5.3. Estimation de l'offre et de la demande

L'offre d'un élément receveur E_r correspond à la charge X_r qu'il accepte de recevoir pour que son temps d'exécution TEX_r converge vers le temps d'exécution TEX_G du groupe auquel il appartient. Nous définissons un intervalle de confiance basé sur l'écart type σ, soit : $TEX_r \in [TEX_G - \sigma\ ,\ TEX_G + \sigma]$.
En réalité il s'agit de faire converger $TEX_r \rightarrow\ TEX_G$.
Soit $TEX_r = \frac{LOD_r + X_r}{SPD_r} \simeq \frac{LOD_G}{SPD_G} \Rightarrow X_r \simeq \frac{LOD_G.SPD_r}{SPD_G} - LOD_r$.
Ainsi, nous pouvons estimer l'offre totale de l'ensemble GER des éléments receveurs :

$$OFFRE = \sum_{E_r \in GER} \frac{LOD_G.SPD_r}{SPD_G} - LOD_r$$

Par un raisonnement analogue, nous pouvons déterminer la demande d'un élément source E_s, qui correspond à la charge Y_s qu'il souhaite faire migrer pour que $TEX_s \rightarrow TEX_G$.
$TEX_s = \frac{LOD_s - Y_s}{SPD_s} \simeq \frac{LOD_G}{SPD_G} \Rightarrow Y_s \simeq LOD_s - \frac{LOD_G.SPD_s}{SPD_G}$
La demande totale de l'ensemble GES des éléments sources est définie par :

$$DEMANDE = \sum_{E_s \in GES} LOD_s - \frac{LOD_G.SPD_s}{SPD_G}$$

6. Algorithmes d'équilibrage

Nous définissons 3 niveaux d'algorithmes : *Intra-grappe*, *Extra-grappe* et *Inter-grappes*.

– ***Algorithme d'équilibrage Intra-grappe :*** Cet algorithme constitue le noyau de notre stratégie. L'approche de localité adoptée fait que c'est le niveau d'équilibrage qui sera le plus fréquemment sollicité. Il est déclenché lorsqu'un gestionnaire d'EC's constate qu'il y a déséquilibre entre ses EC's. Pour faire ce constat, le gestionnaire reçoit, de manière périodique, les informations de charge à partir de chaque élément de calcul. Sur la base de ces informations et du seuil d'équilibrage estimé ε, il analyse de manière régulière la charge du groupe. En fonction du résultat de cette analyse, soit il décide de déclencher un équilibrage local en cas de déséquilibre, soit il décide d'informer son gestionnaire du niveau supérieur sur sa charge actuelle.

– ***Algorithme d'équilibrage Extra-grappe :*** Cet algorithme, qui utilise une approche *source-initiative*, est exécuté uniquement lorsque certains gestionnaires d'EC's n'ont pas réussi à équilibrer localement leur charge pour cause de saturation ou d'offre insuffisante. Dans ce cas, le gestionnaire de grappes tente d'équilibrer la charge globale de l'extra-grappe à travers les éléments qu'il gère. Contrairement à l'algorithme *intra-grappe*, l'équilibrage extra-grappe devra tenir compte des coûts de communication. Durant le transfert de tâches, nous choisirons comme grappe réceptrice, celle qui nécessite le

plus petit coût de transfert. Ainsi, le critère de sélection sera pondéré par le coût de communication (coût de transfert de tâches) pour assurer qu'une tâche ne peut être transférée que lorsque son temps de réponse estimé dans la grappe réceptrice, auquel nous rajoutons le temps de transfert à partir de la grappe source, est meilleur que son temps de réponse dans la grappe source.

– ***Algorithme d'équilibrage Inter-grappes :*** Ce troisième niveau d'algorithme procède à un équilibrage global à travers toutes les extra-grappes de la grille. Il est exécuté dans le cas extrême où la majorité des gestionnaires de grappes n'arrivent pas à équilibrer localement leur surplus de charge. Le recours à ce type d'équilibrage ne sera effectif que si les coûts de communication, associés aux différentes tâches à transférer, seront réellement intéressants. En d'autres termes, il serait plus judicieux de laisser un ensemble de tâches attendre la libération de ressources que de procéder à leur transfert vers d'autres sites sans un gain de temps substantiel.
Dans ce qui suit, nous allons décrire l'algorithme générique associé à notre stratégie.

– ***Algorithme d'équilibrage Intra/Extra/Inter-grappe : (cas d'un groupe G)***

Estimation de la charge du groupe

1. Collecte périodique de l'information de charge
Pour chaque élément E_i de G **faire**
  Envoi de sa charge actuelle LOD_i à son gestionnaire associé.
Fin Pour
2. Le gestionnaire de groupe G effectue les opérations suivantes :
a- Estimer la vitesse SPD_G et la capacité SAT_G du groupe G ;
b- Calculer la charge courante LOD_G et le temps d'exécution TEX_G de G ;
c- Calculer l'écart type σ_G sur les temps d'exécution des éléments de G ;
d- Envoyer l'information de charge de G à son gestionnaire correspondant.

Prise de décision

3. *Test d'équilibre* :
a- Cas intra-grappe : **Si** ($\sigma_G \leq \varepsilon$) **Alors** grappe en état d'équilibre **Fin Si**
b- Autres cas : **Si** (proportion d'éléments surchargés $\leq$ seuil donné) **Alors**
le groupe G est en état d'équilibre **Fin Si**
4. *Test de saturation* :
Si ($\frac{LOD_G}{SAT_G} > \delta$) **Alors** G est saturé **Fin Si**
5. Partitionnement des éléments de G en surchargés (GES), sous-chargés (GER) et équilibrés (GEN) : $GES \leftarrow \Phi$; $GER \leftarrow \Phi$; $GEN \leftarrow \Phi$.
Pour Chaque E_i de G **faire**
  Si (E_i Saturé) **Alors**
    $GES \leftarrow GES \cup \{E_i\}$
  Sinon
    Selon que
      $TEX_i > TEX_G + \sigma_G : GES \leftarrow GES \cup \{E_i\}$
      $TEX_i < TEX_G - \sigma_G : GER \leftarrow GER \cup \{E_i\}$
      $TEX_G - \sigma_G \leq TEX_i \leq TEX_G + \sigma_G : GEN \leftarrow GEN \cup \{E_i\}$
    Fin Selon que
  Fin Si
Fin Pour

Transfert de tâches

6. $OFFRE = \sum_{E_r \in GER} \frac{LOD_G.SPD_r}{SPD_G} - LOD_r$
$DEMANDE = \sum_{E_s \in GES} LOD_s - \frac{LOD_G.SPD_s}{SPD_G}$
Si ($\frac{OFFRE}{DEMANDE} \leq \rho$) **Alors** Échec d'équilibrage local **Fin Si**
7. Transfert de tâches :
Utiliser l'heuristique 1 en cas de transfert Intra-grappe
Autrement utiliser l'heuristique 2.

Heuristique 1 : Transfert de tâches intra-grappe

a- Trier GES par ordre décroissant des temps d'exécution ;
b- Trier GER par ordre croissant des temps d'exécution ;
c- Tant que ($(GES \neq \Phi$ **ET** $GER \neq \Phi)$) **faire**
  Pour i = 1 **jusqu'à** # (GER) **faire**
    (i) Transférer la tâche la plus prioritaire du premier élément de GES vers le i^{eme} élément receveur de GER.
    (ii) Mettre à jour les charges des éléments source et receveur.
    (iii) Mettre à jour les ensembles GES, GER et GEN.
    (iv) **Si** ($(GES = \Phi$ **OU** $GER = \Phi)$) **Alors** Retourner **Fin Si**
    (v) Trier GES par ordre décroissant des temps d'exécution.
  Fin Pour
Fait

Heuristique 2 : Transfert de tâches extra/inter-grappe

1- Trier GES par ordre décroissant des temps d'exécution.
2- Pour Chaque élément E_j de GES **faire**
  (i) Trier les éléments E_r de GER par ordre croissant de leur bande passante, par rapport aux grappes de E_r et E_j.
  (ii) Trier les éléments de E_j par ordre décroissant des temps d'exécution.
  (iii)- **Tant que** ($(GES \neq \Phi$ **ET** $GER \neq \Phi)$) **faire**
    Pour i = 1 **jusqu'à** # (GER) **faire**
      (a) Transférer la tâche la plus prioritaire du premier élément de E_j vers le i^{eme} élément receveur de E_r, en tenant compte des coûts de communication.
      (b) Mettre à jour les charges des éléments E_j et E_r.
      (c) Mettre à jour les ensembles GES, GER et GEN.
      (d) **Si** ($(GES = \Phi$ **OU** $GER = \Phi)$) **Alors** Retour **Fin Si**
      (e) Trier GES par ordre décroissant des temps d'exécution.
    Fin Pour
  Fait
Fin Pour

7. Etude expérimentale

Afin de tester le comportement du modèle et d'évaluer les performances de la stratégie proposée, nous avons développé un simulateur en utilisant une approche multi-agents. Le choix d'une telle approche est motivé essentiellement par deux facteurs :
(i) d'une part, les architectures multi-agents suscitent un intérêt grandissant en tant qu'outil facilitant la conception, le développement et le déploiement d'applications réparties. L'objectif est de développer un système d'équilibrage basé sur des agents qui soient coopératifs, de telle sorte que chaque agent évolue d'une manière autonome, et coopère avec les autres agents pour se répartir les tâches et les ressources d'une manière efficace ;
(ii) d'autre part, la plate-forme que nous utilisons traditionnellement pour nos différentes expérimentations est une plate-forme multi-agents (plate-forme JADE).
Le simulateur que nous avons développé utilise un fichier de configuration d'une grille comportant les éléments suivants :

1) *Au niveau des ressources de calcul* :
   - Répartition aléatoire des EC's à travers les intra-grappes et les extra-grappes.
   - Génération aléatoire des caractéristiques des éléments de calcul (vitesses, capacités, etc.)
   - Introduction des seuils, des périodes d'envoi des informations de charge, etc.
2) *Au niveau des tâches* :
   - Génération d'un ensemble de tâches avec leurs caractéristiques : période de soumission, nombre de tâches à soumettre, date de soumission, temps estimé d'exécution, etc.
   - Affectation de tâches aux EC's.
3) *Au niveau du réseau* : Génération aléatoire de différentes largeurs de bandes LAN/WAN.

Du point de vue agents, nous avons associé à chaque niveau du modèle un agent :

– *MainAgent* : Cet agent a pour rôle d'initialiser tous les autres agents en leur distribuant aléatoirement des tâches.

– *AgentExtra* : Chargé de la décision d'équilibrage extra-grappe.

– *AgentIntra* : Associé à une intra-grappe, il a pour rôle de :

- Collecter les informations de charge sur les EC's à partir des agents *AgentEC* qui leurs sont associés et prendre une décision d'équilibrage intra-grappe si nécessaire.
- Informer périodiquement son agent gestionnaire *AgentExtra*, sur sa charge actuelle.
- Participer à un équilibrage intra-grappe comme receveur ou comme source.

– *AgentEC* : Cet agent peut avoir trois comportements :

- Un comportement périodique qui consiste à calculer sa charge locale et à l'envoyer vers l'*AgentIntra* correspondant.
- Un second comportement qui se déclenche à chaque fois qu'il reçoit un message provenant de l'*AgentIntra* pour lui demander de participer à une opération d'équilibrage.
- Un troisième comportement qui se déclenche lorsqu'il reçoit un message contenant des tâches soumises par l'agent *MainAgent* ou par un autre agent *AgentEC* qui se trouve dans un état surchargé.

7.1. Résultats expérimentaux

Pour expérimenter la stratégie d'équilibrage proposée, nous avons utilisé un PC Pentium IV de 3GHz, doté d'une mémoire de 1Go et fonctionnant sous Windows-XP.

Compte tenu que nos algorithmes évoluent dans un système où il peut y avoir d'autres processus qui s'exécutent en parallèle, et qui peuvent donc influer sur les résultats obtenus, nous avons pris la moyenne des résultats en réitérant les mêmes expériences plus de dix (10) fois. Pour les besoins de nos expériences, nous avons adopté les hypothèses suivantes :

1) Répartition aléatoire d'un nombre d'EC's sur un nombre fixé d'intra-grappes et d'extra-grappes ;
2) Génération des caractéristiques variables pour chaque EC (vitesses comprises entre 500 et 3000 unités de calcul par seconde, capacités variant entre 5000 et 30000 unités de calcul) ;
3) La distribution d'un nombre variable de tâches d'une manière périodique et aléatoire selon une loi uniforme ;
4) Génération du temps d'exécution estimé pour chaque tâche (compris entre 300 et 1500 unités de calcul)
5) Comme critère de sélection, nous avons opté pour le *plus petit temps d'exécution*, qui donne la priorité de transfert à la tâche ayant le plus petit temps d'exécution ;
6) Enfin, nous avons adopté comme indice de charge le *temps d'exécution* défini dans la section 5.2.

Les valeurs attribuées aux seuils ont été fixées après plusieurs expériences. Nous avons constaté que les meilleurs résultats sont obtenus pour $\delta = 0.8$, $\rho = 0.75$, alors que le seuil ε dépend de la répartition initiale des tâches.
Les expérimentations réalisées se basent sur la variation de deux paramètres de performance, à savoir le nombre d'éléments de calcul et le nombre de tâches.
En ce qui concerne les mesures de performance, nous nous étions intéressés aux métriques relatives à l'ensemble des tâches soumises durant une période déterminée :
(i) le *Temps d'Attente Moyen* (TAM) ; (ii) le *Temps d'Exécution Moyen* (TEM) ; et, (iii) le *Temps de Réponse Moyen* (TRM). Ces métriques ont été calculées avant (notées $\_AV$) et après équilibrage ($\_AP$), et ceci afin d'évaluer les performances de l'algorithme.

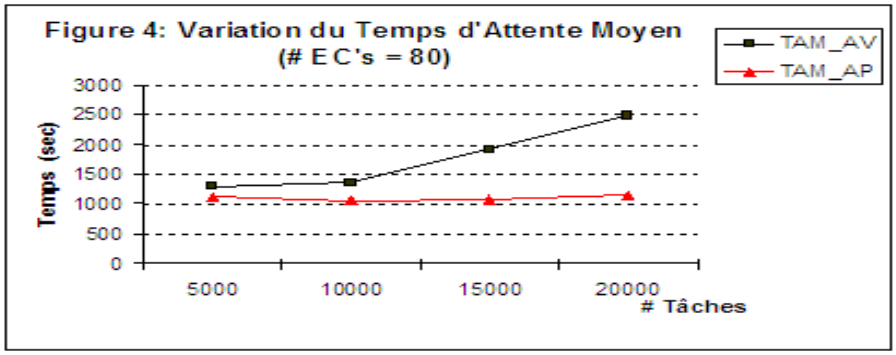
7.1.1. Expérience 1 : *Nombre d'EC's fixe et nombre de tâches variable*

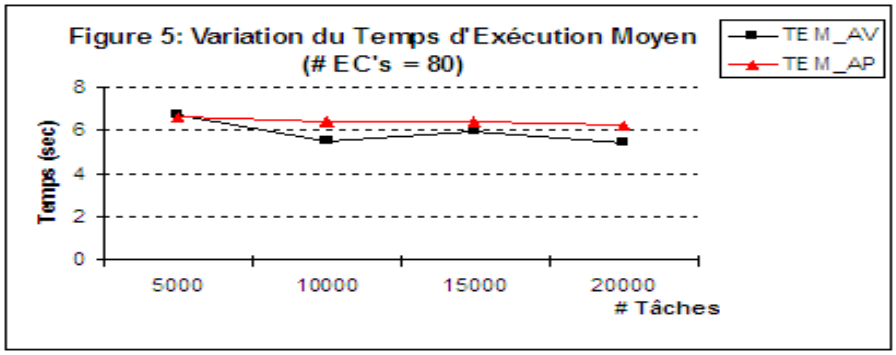
En ce qui concerne cette expérience, nous avons fixé le nombre d'éléments de calcul à 80 et nous avons fait varier le nombre de tâches de 5000 à 20000 par pas de 5000. Ceci nous a permis d'obtenir les résultats suivants :

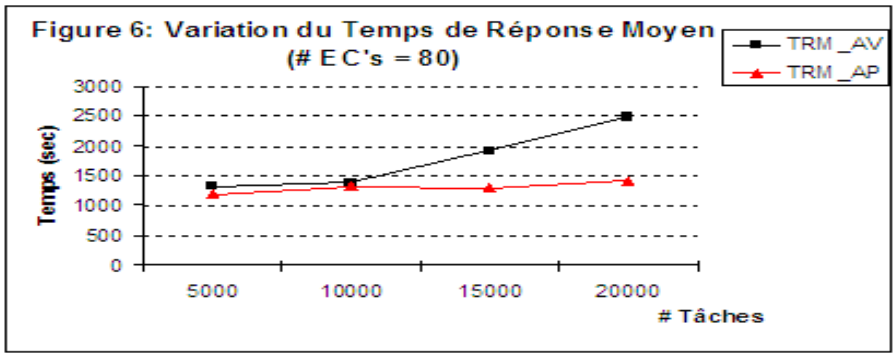
– ***Temps d'attente moyen*** : Nous avons remarqué, dans la figure 4, que le temps d'attente moyen augmente proportionnellement avec le nombre de tâches pour une distribution sans équilibrage, alors que notre stratégie le fait diminuer et surtout le stabilise par rapport à l'augmentation du nombre de tâches.

– ***Temps d'exécution moyen*** : Les deux courbes de la figure 5 montrent que la variation du temps d'exécution moyen est presque constante dans les deux cas (avant et après équilibrage). Ceci est un résultat que nous considérons important car cela veut dire que le coût de mise en œuvre de notre stratégie d'équilibrage n'a pas influé négativement sur les temps d'exécution moyen des tâches.

– ***Temps de réponse moyen*** : La figure 6 montre que la stratégie d'équilibrage proposée a permis de réduire de manière très sensible le temps de réponse moyen des tâches. Plus le nombre de tâches augmente plus nous notons une amélioration perceptible de ce paramètre.







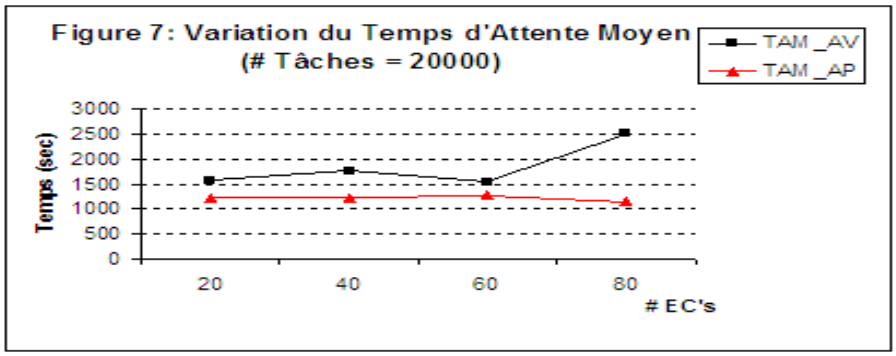
7.1.2. Expérience 2 : *Nombre de tâches fixe et nombre d'EC's variable*

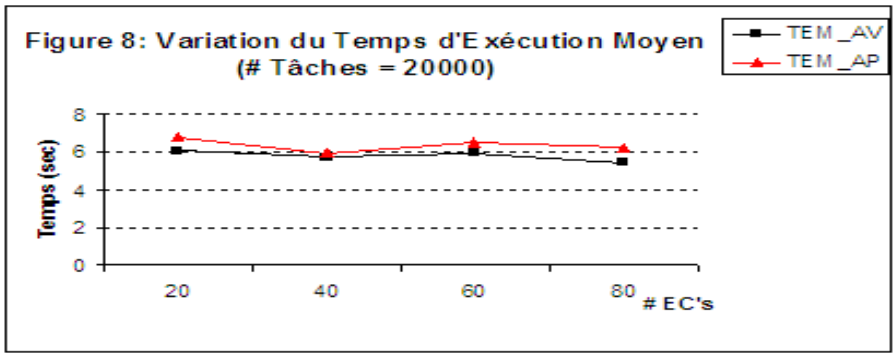
Pour cet ensemble d'expériences, nous avons fixé le nombre de tâches à 20000 et nous avons fait varier le nombre d'éléments de calcul entre 20 et 80 par pas de 20. Les résultats obtenus sont illustrés dans les figures 7-9 :

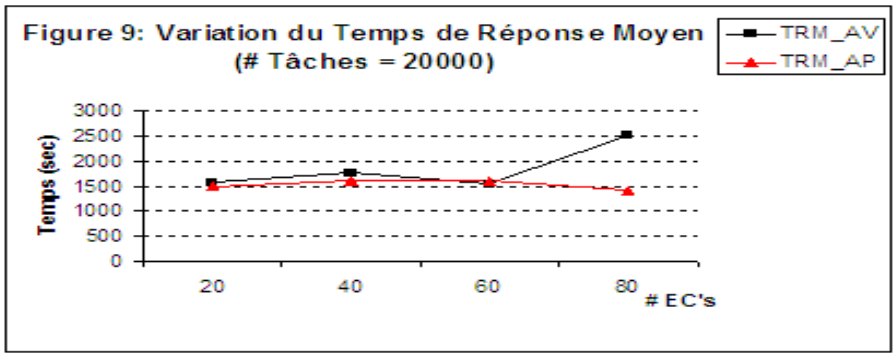
– ***Temps d'attente moyen*** : Dans la figure 7, nous remarquons que l'algorithme d'équilibrage a permis de réduire significativement le temps d'attente moyen des tâches. Ainsi, nous avons pu augmenter l'utilisation des ressources disponibles en minimisant la sensibilité de ce paramètre aux variations du nombre d'EC's.

– ***Temps d'exécution moyen*** : La figure 8 montre que les variations entre les deux stratégies (avant et après équilibrage) ne sont pas significatives. Ainsi, nous pouvons dire que le temps d'exécution de notre stratégie n'a pas eu d'incidence négative sur le temps d'exécution moyen des tâches.

– ***Temps de réponse moyen*** : Pour un nombre d'éléments de calcul égal à 80, nous remarquons que le temps de réponse moyen, après équilibrage, a diminué de manière remarquable (1000 secondes) par rapport à sa valeur avant équilibrage (voir figure 9).







Suite à cette série d'expériences, nous pouvons remarquer que la stratégie d'équilibrage proposée a permis d'améliorer les variations des métriques étudiées sans overhead important. Nous pouvons dire que ces résultats donnent de bons indices sur le comportement de notre stratégie.

8. Conclusion et perspectives

Dans ce papier, nous avons proposé un modèle arborescent permettant de représenter de manière univoque une infrastructure de type grille de calcul. Partant de ce modèle, nous avons ensuite développé un stratégie d'équilibrage ayant comme principaux objectifs : (i) la réduction du temps de réponse moyen des tâches soumises à une grille de calcul ; et, (ii) la réduction du coût de communication lors d'un transfert de tâches.
Comparativement aux travaux existants, notre stratégie d'équilibrage se situe au niveau des tâches et elle privilégie, autant que possible, un équilibrage de charge local pour éviter le recours aux communications extra et inter-grappes. C'est une stratégie distribuée avec prise de décision d'équilibrage sur la base d'informations locales. Compte tenu des surcoûts engendrés par l'exécution d'un système d'équilibrage de charge, notre stratégie n'entame un équilibrage que lorsqu'il est rentable (réduction du temps de réponse moyen des tâches et réduction des coûts de communication). Pour tester le comportement de la stratégie proposée, nous avons développé un simulateur de grilles à base de systèmes multi-agents sous la plate-forme JADE. Les systèmes multi-agents nous ont permis de représenter facilement les différents niveaux du modèle proposé et ont facilité les interactions et les échanges d'informations entre ces niveaux.
Les premiers résultats obtenus nous semblent assez encourageants dans la mesure où ils donnent de bons indices sur le comportement de la stratégie proposée. Nous arrivons à améliorer sensiblement les paramètres de performance que nous avons défini, notamment le temps de réponse moyen et le temps d'attente moyen des tâches, moyennant un surcoût relativement faible par rapport au temps de réponse total.
Comme perspectives, nous envisageons d'intégrer notre stratégie d'équilibrage dans des simulateurs connus dans le domaine des grilles de calcul tel que *GridSim* [7]. Ceci nous permettra d'évaluer les performances de notre stratégie en l'adaptant à des environnements de grilles existants. Comme autre perspective, nous pensons faire évoluer notre modèle arborescent vers un modèle complètement distribué (suppression de la racine de l'arbre), dans lequel nous utiliserons le principe de diffusion pour échanger les informations de charge entre extra-grappes. Nous pensons qu'il est également important de tenir compte, dans une stratégie d'équilibrage, des caractéristiques des tâches qui seront exécutées sur une grille. Plus ces caractéristiques sont connues, plus la stratégie d'équilibrage pourra donner de meilleurs résultats du point de vue performance. Ceci suggère d'ailleurs l'idée de définir une stratégie d'équilibrage adaptée à une classe d'applications bien déterminée. Enfin, il serait intéressant d'inclure dans le modèle d'équilibrage d'autres paramètres tels que l'espace mémoire et les accès disques notamment.

9. Bibliographie

[1] AGGARWAL A. K. , KENT R. D., « An adaptive generalized scheduler for grid applications », *In Proceedings of the 19th International Symposium on High Performance Computing Systems and Applications*, pages 15–18, Guelph, Ontario Canada, May 2005.

[2] ARORA M. , DAS S.K. , BISWAS R., « A decentralized scheduling and load balancing algorithm for heterogeneous grid environments », *Workshop on Scheduling and Resource Management for Cluster Computing*, Vancouver, Canada, pages 499–505, August 2002.

[3] BAKER M., BUYYA R. , LAFORENZA D., « Grids and Grid Technologies for Wide-Area Distributed Computing », *International Journal of Software : Practice and Experience (SPE)*, vol. 32, n° 15, Wiley Press, USA , Nov. 2002.

[4] BANINO C., BEAUMONT O., LEGRAND A. , ROBERT Y., « Scheduling strategies for master-slave tasking on heterogeneous processor grids », *Applied Parallel Computing : Advanced Scientific Computing, 6th International Conference (PARA 02)*, Lecture Notes in Computer Science, vol. 2367, Springer-Verlag, pages 423–432, 2002

[5] BERMAN F., FOX G. , HEY Y., *« Grid Computing : Making the Global Infrastructure a Reality »*, Wiley Series in Communications Networking & Distributed Systems, 2003.

[6] BERSTIS V., FERREIRA L. , AMSTRONG J., « *Introduction to Grid Computing with Globus* », IBM RedBook, December 2002.

[7] BUYYA R. , « *A grid simulation toolkit for resource modelling and application scheduling for parallel and distributed computing* », www.buyya.com/gridsim/.

[8] CAO J., DANIEL P. S., STEPHEN A. J., SAINI S. , GRAHAM R. N., « Agent-Based Grid Load Balancing Using Performance-Driven Task Scheduling », *IPDPS 2003 : Proceedings of the 17th IEEE International Symposium on Parallel and Distributed Processing*, April 22-26, Nice, France, 2003.

[9] CASAVANT T. , KUHL J., « A taxonomy of scheduling in general purpose distributed computing systems », *IEEE Transactions on Soft. Eng.*, vol. 14, n° 2, Pages :141-153, 1994.

[10] CHERVENAK A., FOSTER I., KESSELMAN C., SALISBURY C. , TUECKE S., « The data grid : towards an architecture for the distributed management and analysis of large scientific datasets », *Jour. of Network and Computer Applications*, vol. 23, n° 3, Pages :187–200, 2000.

[11] FERRARI D. , ZHOU S. , « An empirical investigation of load indices for load balancing applications », Technical Report UCB/CSD-87-353, EECS Department, University of California, Berkeley, 1987.

[12] FOSTER I. , KESSELMAN C. (EDITORS), « *The Grid2 : Blueprint for a New Computing Infrastructure* », Morgan Kaufmann (second edition), USA, 2004.

[13] GENAUD S., GIERSCH A. , VIVIEN F., « Load-balancing scatter operations for grid computing », *12th Heterogeneous Computing Workshop (HCW 2003)*, IEEE CS Press, 2003.

[14] HOULE M. , SYMNOVIS A. , WOOD D. , « Dimension-exchange algorithms for load balancing on trees », *Proceedings of 9th International Colloquium on Structural Information and Communication Complexity (SIROCCO'02)*, Pages 181–196, Carleton Scientific, 2002.

[15] HU Y.F., BLAKE R.J. , EMERSON D.R., « An optimal migration algorithm for dynamic load balancing », *Concurrency : Practice and Experience*, vol. 10, Pages 467–483, 1998.

[16] JOHANSSON H., STEENSLAND J. , « A performance characterization of load balancing algorithms for parallel SAMR applications », Technical Report 2006-047 from the Department of Information Technology, Uppsala University, 2006.

[17] LEINBERGER W., KARYPIS G., KUMAR V. , BISWAS R., « Load balancing across near-homogeneous multi-resource servers », *In HCW'00 : Proceedings of the 9th Heterogeneous Computing Workshop*, Pages 60–71, Washington, DC,USA, 2000. IEEE Computer Society.

[18] MYINT C. C. , LAR TUN K.M., « A Framework of Using Mobile Agent to Achieve Efficient Load Balancing in Cluster », *6th Asia-Pacific Symposium on Information and Telecommunication Technologies, APSITT 2005*, pages :66–70, 2005.

[19] OLUGBILE A., XIA H., LIU X. , CHIEN A., « New Grid Scheduling and Rescheduling Methods in the GrADS Project », *Workshop for Next Generation Software*, Santa Fe, New Mexico, April 2004.

[20] SHAN H. , OLIKER L., BISWAS R. , , SMITH W. , « Scheduling in heterogeneous grid environments : The effects of data migration », *In Proc. of ADCOM2004 : International Conference on Advanced Computing and Communication*, India, December 2004.

[21] XU C.Z. , LAU F.C.M., « *Load Balancing in Parallel Computers : Theory and Practice* », Kluwer, Boston, MA, 1997.