Conception d'un Simulateur de Grilles Orienté Gestion d'Équilibrage

Fatima Kalfadj [1], Yagoubi Belabbas[2] et Meriem Meddeber[2]

[1] Université de Mascara, Faculté des Sciences, Département d'Informatique, 29000 Mascara, Algérie
amani_for@yahoo.fr

[2]Université d'Oran, Faculté des Sciences, Département d'Informatique, 31000 Oran, Algérie
byagoubi@yahoo.fr
m.meddeber@yahoo.fr

Résumé. Les dernières évolutions dans le calcul distribué ont conduit à l'apparition de nouvelles infrastructures appelées *grilles de calcul*. La gestion d'équilibrage de charge dans ce type d'infrastructure est complexe et exige donc des outils sophistiqués pour analyser les algorithmes avant de les appliquer aux vrais systèmes. Cependant une recherche étendue a été conduite dans le domaine de la simulation pour modéliser de tels systèmes et comprendre leur comportement. En conséquence, un nombre croissant d'outils de simulation ont été conçus et développés. Dans ce papier nous proposons un outil de simulation de grilles qui fournit des primitives pour la création et l'ordonnancement des tâches indépendantes. C'est un simulateur qui permet d'évaluer les performances d'un modèle distribué pour résoudre le problème d'équilibrage de charge dans les grilles de calcul.

Mots-clés: Grilles de calcul, Équilibrage de charge, Simulateur de grilles, Tâches indépendantes, Modèle d'équilibrage de charge.

1 Introduction

Dés les débuts de l'informatique, les scientifiques furent les plus gros consommateurs de puissance de calcul. La dernière décennie a également vu l'avènement des réseaux et d'Internet. De plus en plus de projets de recherche impliquent de multiples partenaires pouvant être repartis aux quatre coins du globe. Il devient alors nécessaire de disposer d'une infrastructure commune, facilitant les partages d'informations mais aussi de ressources. C'est dans ce contexte, qu'est né le concept de Grille de Calcul (*Grid Computing*): une infrastructure virtuelle constituée d'un ensemble coordonné de ressources informatiques potentiellement partagées,

distribuées, hétérogènes et sans administration centralisée [1]. Cependant la gestion de ressource dans ce type d'infrastructure pose évidemment des problèmes beaucoup plus complexes que ceux posés par les systèmes distribués traditionnels, et ce à cause notamment de leur hétérogénéité et de leur dimension dynamique. Parmi ces problèmes, la répartition de charge où il faut en effet éviter, dans la mesure du possible, les situations où certains nœuds sont surchargés alors que d'autres sont sous chargés ou complètement libres. Pour remédier à ce problème plusieurs algorithmes de répartition de charge ont été développés [2].

Lorsqu'il s'agit de comparer la performance de deux algorithmes, les conditions expérimentales doivent nécessairement être les mêmes. Or assurer la même évolution de multiples composants d'un environnement distribué dans le contexte de la grille est impossible: un trop grand nombre de phénomènes amène un non déterminisme de la plateforme de test. Il est d'usage de les simuler. Beaucoup d'outils standards et spécifiques à l'application ont été établis dans cette optique.

Cependant, ces outils de simulation ne permettent pas de tester facilement de nouveaux algorithmes de gestion d'équilibrage pour les grilles : ils leurs manquent les fonctionnalités permettant la mise en œuvre de la politique d'informations nécessaire à tout système d'équilibrage. Cette information concerne aussi bien l'état de charge des ressources disponibles que la charge du réseau de communication à un instant donné.

Dans cet article nous proposons un simulateur de grilles nommé OrientéSim qui : (i) fournit des primitives pour la création et l'ordonnancement des tâches indépendantes, (ii) permet d'évaluer les paramètres de performance d'un modèle distribué pour résoudre le problème d'équilibrage de charge dans les grilles de calcul.

Le reste de cet article est organisé comme suit : Dans la deuxième section, nous présentons une taxonomie des outils de simulation. La troisième section cite quelques outils de simulation et propose un tableau de comparaison. La quatrième section présente le simulateur proposé. Dans la cinquième section, nous présenterons et discuterons quelques résultats expérimentaux relatifs au model développé. La sixième section conclut cet article et présente quelques perspectives futures de recherche.

2 Taxonomie des outils de simulations

Les outils de simulation sont nombreux et il est difficile d'en faire une présentation détaillée. Pour cela, la nécessité d'avoir une taxonomie qui permet d'uniformiser les terminologies pour une meilleure description est indispensable. Ainsi, Anthony Sulistio, Chee Shin Yeo et Rajkumar Buyya [7] ont proposé une taxonomie largement adoptée.

1. ***Taxonomie des* utilisateurs :** l'outil de simulation peut être utilisé comme un simulateur ou comme un émulateur ; un simulateur est un outil qui représente un système réel par contre l'émulateur est un outil qui agie comme un système réel.

2. ***Taxonomie de simulation :*** en général, une simulation comporte trois propriétés :
   - *Présence du temps:* indique si la simulation d'un système prend en compte le facteur temps. Une simulation statique ne considère pas le temps en tant qu'élément de simulation, contrairement à une simulation dynamique.

   - *Valeur de base:* spécifie les valeurs que peut prendre une entité simulée. Une simulation discrète à des valeurs d'entités appartenant à un intervalle fini tandis qu'une simulation continue propose des valeurs d'entités appartenant à un intervalle infini.

   - *Comportement:* la simulation peut se dérouler d'une manière déterministe (sans événements aléatoires) ; Ainsi la répétition de la même simulation rendra toujours les mêmes résultats contrairement à une simulation probabiliste (avec événements aléatoires) ; la répétition de la même simulation rend souvent des résultats différents.

3. ***Taxonomie de conception :*** Cela consiste à classer les outils de simulations par catégories basées sur les composants et les dispositifs nécessaires à la simulation:
   - *Moteur de simulatio*n : la simulation peut être exécutée en mode séquentiel ou en mode parallèle ; Une simulation séquentielle est exécutée en utilisant un seul processeur, alors qu'une simulation parallèle ou distribuée est exécutée en utilisant plusieurs processeurs.

   - *Environnement de conception :* détermine comment l'utilisateur utilise l'outil pour concevoir des modèles de simulation. Un langage fournit un ensemble de constructions définies pour concevoir des modèles de simulation, alors qu'une bibliothèque fournit un ensemble de routines pour être utilisé avec un langage de programmation.

   - *Interface utilisateur:* détermine comment l'utilisateur agit avec l'outil de simulation. Une interface de conception visuelle permet à l'utilisateur de créer un modèle de simulation beaucoup plus facile et plus rapide. Alors qu'une interface de conception non-visuel exige à l'utilisateur d'écrire des codes de programme ce qui exige plus de temps et d'effort.

   - *Supports système*: fournit les dispositifs utiles et prêts à employer qui aident l'utilisateur à construire un modèle de simulation précis.

3 Quelques outils de simulation de grille

Dans la littérature beaucoup d'outils standards et spécifiques à l'application ont été établis parmi lesquels nous pouvons citer :

- Bricks [3], Il a été proposé et conçu pour des études de comparaisons d'algorithmes d'ordonnancement.
- OptorSim [4], conçu pour l'étude d'algorithmes d'ordonnancement traitant spécifiquement de la réplication ou de la migration de données,
- GridSim [5] ou SimGrid [6], des outils de modélisation de ressources et réseaux d'une grille de calcul pour tester des algorithmes d'ordonnancement distribués.
- MicroGrid [8], permet aux développeurs d'exécuter les applications dans une grille virtuelle. Plus précisément, il a été conçu pour émuler Globus[9].

Le tableau (Table 1) compare ces quatre simulateurs en respectant la taxonomie présentée dans la deuxième section.

Table 1. Taxonomie des outils de simulation

Taxonomie	***Brikcs***	***GridSim***	***MicroGrid***	***OptorSim***
Utilisateur	Simulateur	Simulateur	Emulateur	Simulateur
Simulation	Statique	Statique	Dynamique	Dynamique
	Discrète	Discrète	Discrète	Discrète
	Déterministe	Déterministe	Déterministe	Déterministe
Conception	Evénement	Multithread Avec événement	Parallèle avec événement	Parallèle
	Langage	Bibliothèque	Langage	Bibliothèque
	Non-Visuel	Non-Visuel	Non-Visuel	Graphe
	Génération Des statistiques	Génération Des statistiques	N/A	Génération De statistiques

4 Simulateur proposé (OrientéSim)

4.1 Architecture

La figure Fig. 1, présente les différents composants du simulateur que nous avons développé dans le but d'avoir un outil approprié à nos besoins, à savoir modéliser l'information de charge qui caractérise les ressources de calcul d'une grille, implémenter et évaluer les performances des stratégies d'équilibrage de charge dédiées aux environnements de grille de calcul.

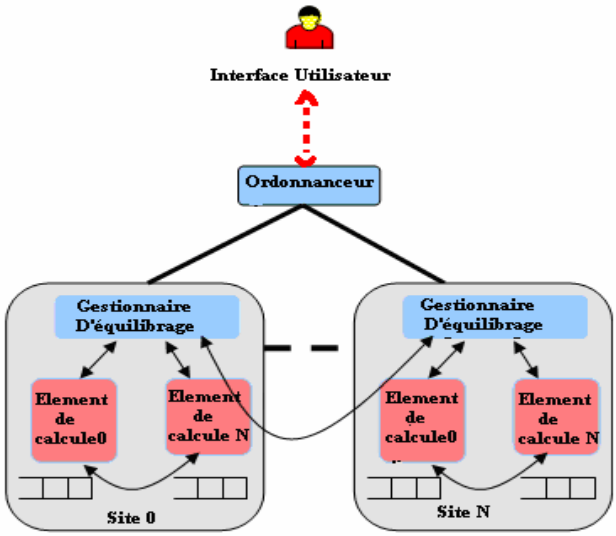

Fig. 1. Architecture d'OrientéSim.

D'un point de vue architectural OrientéSim est composé de:

- *Interface utilisateur:* à travers laquelle on peut générer le fichier de configuration d'une grille (nombre de sites, nombre d'éléments de calcul, leurs caractéristiques, période d'envoi des informations de charge, largeur de bandes, etc...) ;
- *Ordonnanceur:* effectue l'ordonnancement des tâches selon trois stratégies.
  - (i) *Aléatoire:* les tâches sont distribuées aléatoirement sur les éléments de calcul,
  - *(ii) A priorité fixe:* la tâche contenant le plus grands nombre d'instructions est assignée à l'élément de calcul le plus puissant,
  - (iii) *Round robin* : la première tâche est assignée au premier élément de calcul, la deuxième tâche au deuxième élément de calcul etc. …, d'une façon circulaire.

- *Sites:* fournit les ressources de calcul nécessaires à l'exécution des tâches soumises par l'ordonnanceur.

- *Gestionnaire d'équilibrage:* chaque gestionnaire participe au maintien des informations de charge et à l'équilibrage de la charge globale des éléments de calcul de la grille. Les différents gestionnaires peuvent échanger leurs informations de charge.

4.2 Modèle de ressource

Chaque ressource représente un élément de calcul, et se caractérise par :

- *Hétérogénéité :* sur le plan matériel (architecture des processeurs, nombre de processeurs, vitesse CPU mesurée en MIPS[1], *File d'attente)* et sur le plan logiciel (système d'exploitation).

- *Période :* durant laquelle, la ressource mesure sa charge courante.

4.3 Modèle de réseau

Dans OrientéSim il n'existe pas des protocoles réseaux ou de normes qui doivent être suivies. Tous les éléments sont reliés en utilisant des liens logiques.

4.4 Implémentation

OrientéSim est écrit en Java, pour les raisons principales suivantes :

- *Approche orientée objet* : où il y a plusieurs composants distincts qui agissent les uns sur les autres par l'intermédiaire de méthodes bien définies.
- *Capacité d'exécuter des threads concourants :* les threads sont assignés à chaque site et à chaque CE (élément de calcul), l'ordonnanceur et un thread simple.
- *Portabilité :* permettant à la simulation d'être distribuée facilement sans avoir à recompiler le code pour les différents systèmes.
- *Extensibilité :* Le code est structuré dans plusieurs packages, dont chacun traite une partie différente de la simulation.

[1] Million d'Instructions Par Seconde

4.5 Processus de simulation

La figure fig. 2, montre les différentes interactions entre les composants d'OrientéSim pendant la vie de la simulation, c'est un exemple de diagramme de séquence dans lequel le temps augmente de haut en bas.

Tout d'abord l'ordonnanceur affecte chaque tâche soumise par l'un des utilisateurs à un élément de calcul selon l'une des stratégies d'ordonnancement présentées ci-dessus. Les tâches sont insérées aux files d'attente de chaque élément de calcul. Quand l'élément de calcul est prêt à traiter la tâche il la dépile et l'exécute.

A chaque période de temps l'élément de calcul évalue sa charge courante, envoie son information de charge au gestionnaire d'équilibrage, ce dernier prend les décisions d'équilibrage de charge, la procédure se répète jusqu'à la fin de la simulation.

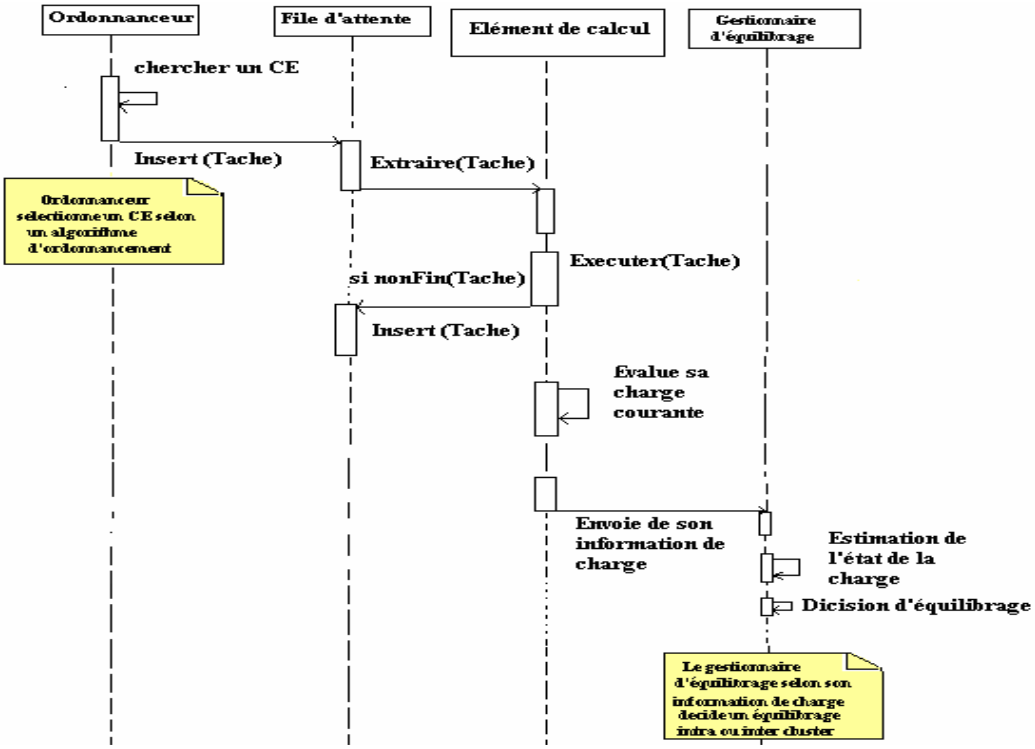

Fig. 2. Diagramme UML du processus de simulation

5 Résultats expérimentaux

Pour valider notre simulateur nous avons implémenté un modèle distribué reposant sur une architecture hybride. Sur la base de ce modèle nous avons développé une stratégie d'équilibrage centralisée locale**, *intra-cluster***, et une deuxième totalement distribuée, ***inter-clusters***. L'ensemble des expériences ont été réalisées sur un PC Pentium DUAL CPU de 2.00 GHz, doté d'une mémoire de 1 Go et fonctionnant sous Windows XP.

5.1 Modèle de la grille

Dans ce modèle, nous considérons qu'une grille de calcul est composée d'un ensemble de *Clusters* qui communiquent à travers un réseau WAN. Chaque cluster est à son tour composé d'un ensemble de *Nœuds* de calcul qui communiquent à travers un réseau LAN. Ces entités (nœuds de calcul, clusters et réseaux locaux) peuvent être hétérogènes. Nous pouvons représenter cette topologie par un modèle arborescent à deux niveaux (Fig. 3).

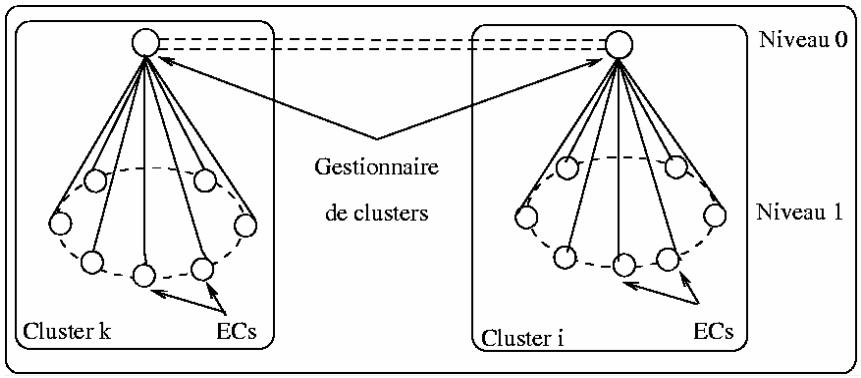

Fig. 3. Modèle générique de représentation d'une grille

Les deux niveaux sont définis comme suit :

- **Niveau 1 :** ce niveau est constitué d'un ensemble de nœuds qui représentent les clusters de la grille. Chaque nœud est appelé *gestionnaire du cluster*.
- **Niveau 2 :** ce niveau est à son tour constitué d'un ensemble de nœuds, qui correspondent aux nœuds de calcul de chaque cluster.

5.2 Stratégie d'équilibrage de charge

Selon la structure arborescente du modèle proposé nous avons développés une stratégie d'équilibrage de charge à deux niveaux : *Intra-cluster* et *Inter-clusters.*

- ***Équilibrage intra-cluster :*** Dans cette première phase, chaque gestionnaire de cluster et selon les informations de charge transmises par ses éléments de calcul, décide de lancer une opération d'équilibrage de charge locale à son cluster.
- ***Équilibrage intra-grille*** cette deuxième phase est réalisée entre les clusters de la grille, elle est lancée par chaque gestionnaire de cluster qui ne parvient pas à équilibré sa charge localement.

5.3 Résultats de l'algorithme Intra-cluster

Nous avons utilisé la stratégie aléatoire comme stratégie d'ordonnancement et nous avons varié le nombre de nœuds de calcul de 40 à 100, avec un nombre de tâches variant de 4000 à 8000.

Le tableau suivant, montre les gains obtenus en temps de reponse, temps d'attente et temps d'exécution.

Table. 2. Résultats de l'algorithme Intra-clusters

Taches / nœuds		**40**	**60**	**80**	**100**	**120**
Gains sur le temps de réponse	4000	39.49%	34.65%	29.51%	23.62%	18.94%
	5000	42.27%	40.50%	35.14%	29.23%	22.55%
	6000	37.52%	43.16%	38.67%	32.59%	28.26%
	7000	42.88%	43.33%	40.30%	36.65%	30.93%
	8000	35.26%	42.05%	43.35%	40.07%	35.41%
Gains sur le temps d'attente	4000	47.43%	44.91%	39.06%	32.30%	26.80%
	5000	51.93%	52.32%	46.32%	38.92%	31.15%
	6000	45.23%	54.38%	49.46%	42.82%	37.93%
	7000	52.72%	54.50%	50.98%	47.80%	40.80%
	8000	42.34%	51.58%	54.32%	51.64%	46.42%
Gains sur le temps d'exécution	4000	16.67%	14.62%	11.90%	9.72%	4.82%
	5000	15.82%	15.13%	13.95%	13.47%	6.29%
	6000	10.62%	17.01%	14.61%	14.61%	13.22%
	7000	15.39%	17.35%	14.25%	14.32%	14.61%
	8000	4.38%	17.03%	15.79%	15.89%	15.70%

Interprétation des résultats

Nous pouvons constater à partir des résultats ci-dessus que pour parvenir à stabiliser le système et avoir de bons gains, il faut avoir, pour une charge de 4000 à 8000 tâches et un nombre de ressources variant de 60 à 100 nœuds.

6 Conclusion

Dans cet article nous avons proposé un outil de simulation de grille OrientéSim, qui dispose d'une architecture extensible et modulaire, il fournit des primitives pour la création et l'ordonnancement des tâches indépendantes et permet de tester les paramètres de performances d'un modèle distribué pour l'équilibrage de charge dans les grilles de calcul.

Comme perspectives, nous pensons à :

- Implémenter d'autres modèles d'équilibrage de charge.
- Etendre notre simulateur par un module de gestion de réplication.
- Nous avons implémenté deux stratégies d'équilibrage de charge basé essentiellement sur un placement de taches comme mécanisme de transfert. Nous aimerions élargir notre simulateur pour étudier l'équilibrage de charge avec une migration de taches qui s'avère une opération très complexe à réaliser.
- Appliquer l'équilibrage de charge dans le cas ou les tâches sont dépendantes.
- Nous avons vue que dans OrientéSim il n'existe pas des protocoles et des normes qui doivent être suivie on aimerait bien d'implémenter des protocoles réseau.

References

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

2. B.Yagoubi and Y.Slimani, «*Dynamic Load Balancing Strategy for Grid computing*», Transactions on Engineering, Computing and Technology, vol 13, May 2006.

3. K. Aida, A.Takefusa, H. Nakada, S. Matsuoka, S. Sekiguchi, and U. Nagashima, «*Performance Evaluation Model for Scheduling in a Global Computing System*», Int. J. of High Performance Computing Applications, 14(3), 2000, 268–279.

4. W. H. Bell, D. G. Cameron, L. Capozza, A. P. Millar, K.Stockinger, and F. Zini, «*OptorSim – A Grid Simulator for Studying Dynamic Data Replication Strategies*», Int. J. of High Performance Computing Applications, 17(4), 2003, 403–416.

5. R. Buyya and M. Murshed, «*GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and scheduling for Grid Computing*», Journal of Concurrency and Computation : Practice and Experience,pages1175-1220, 2002

6. H. Casanova, «*Simgrid: A Toolkit for the Simulation of Application Scheduling*». Proc. of the First IEEE/ACM Int. Symposium on Cluster Computing and the Grid, Brisbane, Australia, 2001, 430–437.

7. A.Sulistio, C. Yeo and R Buyya «A taxonomy of computer-based simulations and its mapping to parallel and distributed systems simulation tools» sortware practice and experience.2004.

8. H. Song, X. Liu ,D. Jakobson, R.Bhagwan, X. Zhang, K. Taura, and A. Chien. *The microgrid*, proceedings of the 2000 ACM/IEEE conference on supercomputing, page 53. john wiley and Sons,Ltd.

9. I. Foster. Globus toolkit version 4 : Software for service oriented systems. *In IFIP: International Conference on Network and Parallel Computing*, pages 2-13, Beijing, China, November 2005.