Distributed Load Balancing Model for Grid Computing

Belabbas Yagoubi * — Meddeber Meriem **

* University of Oran
Department of Computer Science,Oran, Algeria
byagoubi@yahoo.fr

** University of Mascara
Department of Computer Science,Mascara, Algeria
m.meddeber@yahoo.fr

ABSTRACT. Most existing load balancing strategies were developed, assuming homogeneous set of resources linked with homogeneous and fast networks. However, these assumptions are not realistic in Grid computing because their specific characteristics like *heterogeneity*, *scalability* and *dynamicity*. Hence, load balancing problem represents a new challenge for these new environments where many research projects are under way.
Our contributions in this paper are two folds. First we propose a distributed load balancing model transforming any Grid topology into a forest structure. Second we develop, over this model, a two level load balancing that reduces the average response time of tasks and their transferring cost. The proposed strategy is fully distributed with local decision making avoiding, if possible, the use of wide area network.

RÉSUMÉ. La plupart des stratégies d'équilibrage de charge existantes se sont intéressées à des systèmes distribués supposés avoir des ressources homogènes interconnectées à l'aide de réseaux homogènes et à hauts débits. Pour les Grilles de calcul, ces hypothèses ne sont pas réalistes à cause des caractéristiques d'*hétérogénéité*, de *passage à l'échelle* et de *dynamicité*. Ainsi, pour ces environnements, le problème d'équilibrage de charge constitue un nouveau défi pour lequel plusieurs recherches sont actuellement investies.
Notre contribution dans cette perspective à travers ce papier est double: premièrement, nous proposons un modèle distribué d'équilibrage de charge, permettant de représenter n'importe quelle topologie de grille en une structure de forêt. Nous développons ensuite sur ce modèle, une stratégie d'équilibrage à deux niveaux ayant comme principaux objectifs la réduction du temps de réponse moyen et le coût de transfert de tâches. La stratégie proposée est de nature distribuée avec une prise de décision locale, ce qui permettra à chaque fois que c'est possible d'éviter le recours au réseau de communication à large échelle.

KEYWORDS : Load balancing,Grid computing, Distributed model, Transferring cost, Workload.

MOTS-CLÉS : Équilibrage de charge,Grilles de calcul, Modèle distribué, Coût de transfert, Charge de travail.

1. Introduction

Recent researches on computing architectures are allowed the emergence of a new computing paradigm known as *Grid computing*. Grid is a type of distributed system which supports the sharing and coordinated use of resources, independently from their physical type and location, in dynamic virtual organizations that share the same goal [5]. This technology allows the use of geographically widely distributed and multi-owner resources to solve large-scale applications like meteorological simulations,data intensive applications, research of DNA sequences, etc. [3].
In order to fulfill the user expectations in terms of performance and efficiency, the Grid system needs efficient load balancing algorithms for the distribution of tasks. The main goal is to prevent, if possible, the condition where some processors are overloaded with a set of tasks while others are lightly loaded or even idle[10].

The essential objective of a load balancing can be, depending on the user or the system administrator, defined by:

– The aim for the user is to minimize the *makespan* of its own application, regardless the performance of other applications in the system.

– The main goal for administrator is to maximize the *mean tasks deadline* by ensuring maximal utilization of available resources.

Typically, a load balancing scheme consists of four policies:

- The *information policy* is responsible to define when and how the information on the Grid resources availability is updated.
- The *location policy* determines a suitable transfer partner (*server* or *receiver*) once the transference policy decided that this resource is server or receiver.
- The *selection policy* defines the task that should be transferred from the busiest resource to the idlest.
- The *transference policy* classifies a resource as server or receiver according to its availability status.

Although load balancing problem in conventional parallel and distributed systems has been intensively studied, new challenges in Grid computing still make it an interesting topic, and many research projects are under way. Load balancing algorithms in classical parallel and distributed systems, which usually run on homogeneous and dedicated resources, cannot work well in the Grid architectures [1]. Grids has a lot of specific characteristics, like *heterogeneity*, *autonomy*, *dynamicity* and *scalability*, which make the load balancing problem more difficult.

Our main contributions in this paper are two folds. First we propose a distributed load balancing model transforming, univocally, any Grid topology into a forest structure. Second we develop a two level strategy to load balance tasks among resources of computational Grid.
Our strategy privileges local load balancing than global ones in order to achieve two main objectives:
(i) The reduction of the average response time of tasks; and,
(ii) The reduction of the communication cost induced by task transferring.

The rest of this paper is organized as follows: Some related works are described in Section 2. The mapping of any Grid architecture into a forest is explained in Section 3. Section 4 describes the main steps of the proposed load balancing strategy. In Section 5, we present and discuss the performance of our strategy through some experimental results. Finally, Section 6 concludes the paper and providing a preview of future research.

2. Related works

Most load balancing approaches are oriented on application partitioning via graph algorithms [8]. However, it does not address the issue of reducing migration cost, that is, the cost entailed by load redistribution, which can consume order of magnitude more time than the actual computation of a new decomposition. Some works [9] have proposed a latency - tolerant algorithm that takes advantage of overlapping the computation of internal data and the communication of incoming data to reduce data migration cost. Unfortunately, it requires applications to provide such a parallelism between data processing and migration, which restricts its applicability. Genaud et al. [6] enhance the MPI_Scatterv primitive to support master-slave load balancing by taking into consideration the optimization of computation and data distribution using a linear programming algorithm. However, this solution is limited to static load balancing. In [7], Hu et al. propose an optimal data migration algorithm in diffusive dynamic load balancing through the calculation of Lagrange multiplier of the Euclidean form of transferred weight. This work can effectively minimize the data movement in homogenous environments, but it does not consider the network heterogeneity.

As mentioned above, a large number of load balancing techniques and heuristics presented in the literature, target only homogeneous resources. However, modern computing systems, such as the computational Grid, are most likely to be widely distributed and strongly heterogeneous. Therefore, it is essential to consider the impact of these characteristics on the design of load balancing techniques.

The traditional objective, when balancing sets of computational tasks, is to minimize the overall execution time called *makespan*. However, in the context of heterogeneous distributed platforms, makespan minimization problems are in most cases *NP-complete*, sometimes even *APX-complete* [10]. In addition, when dealing with large scale systems, an absolute minimization of the total execution time is not the only objective of a load balancing strategy. The communication cost, induced by load redistribution, is also a critical issue. For this purpose, yagoubi proposed in [11], a hierarchical load balancing model as a new framework to balance computing load in a Grid. Unfortunately, the root of the proposed model can become a bottleneck.

We propose, in this paper, a novel load balancing strategy to address the new challenges in Grid computing. Comparatively to the existing works, the main characteristics of our strategy are:
(i) It is a *fully distributed* strategy, witch allows to solve bottleneck of the model proposed in[11];
(ii) It uses a *task-level* load balancing; and, (iii) It privileges local tasks transfer than global ones, in order to *reduce communication costs* induced by the migration of tasks.

3. Mapping a Grid into a forest-based model

From the topological point of view, we regard a Grid as a set of clusters in a multi-nodes platform. Each cluster owns a set of worker nodes and belongs to a local domain (LAN). Every cluster is connected to the global network (WAN). As illustrated by Figure 1 The load balancing algorithm proposed in this paper is based on mapping model of any Grid topology into a forest structure. Each cluster is modeled by a two levels tree: The root represents the ***cluster manager*** and the leaves are associated to the ***worker nodes***.

Level 0: Each *cluster manager* of this level is associated to a physical cluster of the Grid.
In our load balancing strategy, this manager is responsible to:

– Maintain the workload information relating to each one of its worker nodes;

– Estimate the workload of associated cluster and diffuse this information to other cluster managers;

– Decide to start a local load balancing, which we will call ***intra-cluster*** load balancing;

– Send the load balancing decisions to the worker nodes which it manages, for execution;

– Initiate a global load balancing, which we will call ***inter-clusters*** load balancing.

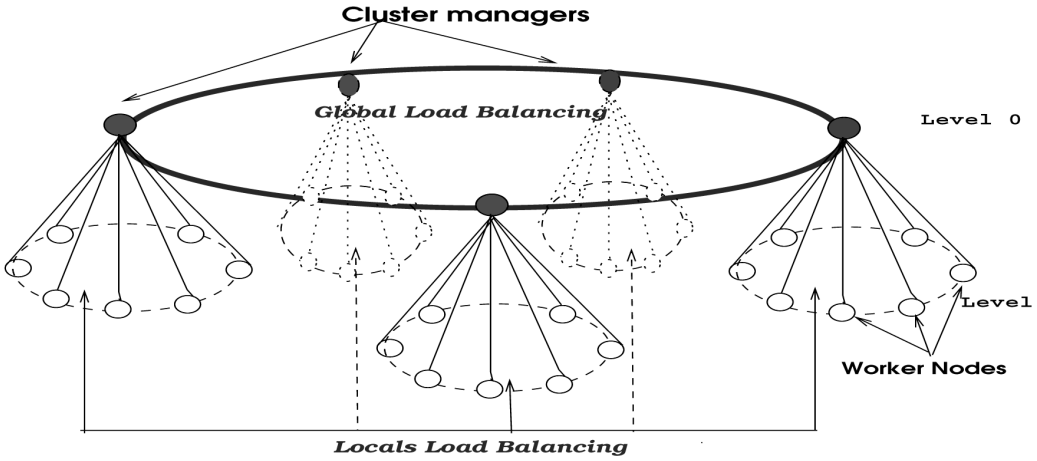

Figure 1. *Two level representation model of a Grid*

Level 1: At this level, we find the *worker nodes* of a Grid linked to their respective clusters. Each node at this level is responsible to:

– Maintain its workload information;

– Send this information to its cluster manager;

– Perform the load balancing decided by its cluster manager.

4. Load Balancing Strategy

In accordance with the structure of the proposed model, we distinguish between two load balancing levels: *Intra-cluster* load balancing (Inter-worker nodes) and *Inter-clusters* (Intra-Grid) load balancing.

4.1. Intra-cluster load balancing

Depending on its current load, each cluster manager decides to start a load balancing operation.
In this case, the cluster manager tries in priority, to load balance its workload among its worker nodes. At this level, communication costs are not taken into account in the task transfer since the worker nodes of the same cluster are interconnected by a LAN network, of which communication cost is constant.
To implement this local load balancing, we propose the following three steps strategy:
Step 1: *Estimation of the current cluster workload*
Each cluster manager estimates its associated cluster capability by performing the following actions:
(i) Estimates current workload of the cluster based on workload information received from its nodes;
(ii) Computes the standard deviation over the workload index [1] in order to measure the deviation between its involved nodes;
(iii) Sends workload information to the other cluster managers.
Step 2: *Decision making*
In this step the cluster manager decides whether it is necessary to perform a balancing operation or not. For this purpose it executes the two following actions:

1. To consider the heterogeneity between worker nodes capabilities, we propose to take as workload index the processing time denoted (TEX). We define the processing time of an entity (node or cluster) as ratio between its workload (LOD) and its capability (SPD): $TEX = \frac{LOD}{SPD}$.

1. Defining the imbalance/saturation state of cluster. If we consider that the standard deviation σ measures the average deviation between the processing times of worker nodes and the processing time of their cluster, we can say that this cluster is in balance state when σ is small. Then, we define the balance and saturation states as follows:

Balance state: We define a *balance threshold* ε, from which we can say that the σ tends to zero and hence the group is balanced: **If** $(\sigma \leq \varepsilon)$ **Then** the cluster is balanced **Else** It is imbalanced.
Saturation state: A cluster can be balanced while being saturated. In this particular case, it is not useful to start an intra cluster load balancing since its nodes will remain overloaded. To measure saturation, we introduce another threshold called *saturation threshold*, denoted as δ.

2. Cluster partitioning. For an imbalance case, we determine the overloaded nodes and the under loaded ones, depending on processing time of every node relatively to their associated cluster.
Step 3: *Tasks transferring.*
To transfer tasks from overloaded nodes to under loaded ones, we propose the following heuristic:
a- Evaluate the total amount of load "**Supply**", available on receiver nodes;
b- Compute the total amount of load "**Demand**", required by source nodes;
c- If the supply is much lower than the demand, it is not recommended to start local load balancing. We introduce a third threshold, called expectation threshold denoted as ρ , to measure relative deviation between supply and demand. We can then write **If** (Supply/Demand > ρ) **Then** perform Local load balancing **Else** perform Global load balancing;
d- Performs tasks transfer according to selection criteria [2].

For the sake of the paper length we give in the following only the inter-clusters load balancing algorithm. A similar intra-cluster load balancing algorithm can be found in [11].

4.2. Inter-clusters load balancing

The load balancing at this level is used if a cluster manager fails to balance their workload among their associated nodes. If we have such case, each overloaded cluster manager transfer tasks from its overloaded worker nodes to under loaded clusters. Contrary to the intra-cluster level, we should consider the communication cost between clusters. Knowing the global state of each cluster, the overloaded cluster manager can distribute its overload tasks between under loaded clusters. The chosen under loaded clusters are those than need minimal communication cost for transferring tasks from overloaded clusters. A task can be transferred only if the sum of its latency in the source cluster and cost transfer is lower than its latency on the receiver cluster. This assumption will avoid making useless task migration.

Inter-clusters load balancing algorithm

1. For Each cluster manager of cluster G_i **AND** according to its specific period **do**
  a- Computes speed SPD_i , capacity SAT_i and processing time TEX_i of G_i.
  b - Sends its current workload information to all other cluster managers.
end For
2. Let GES , GER and GEN are respectively overloaded, under loaded and balanced cluster sets.
$GES \leftarrow \Phi$; $GER \leftarrow \Phi$; $GEN \leftarrow \Phi$
3. Sort items of GES by descending order of their processing times.

2. As criterion selection we propose to transfer in first the task with *longest remaining processing time.*

4. For Every cluster G_j of set GES **do**
(i) Sort the clusters G_r of GER by ascending order of inter clusters (G_j-G_r) WAN bandwidth sizes.
(ii) Sort the nodes of G_j by descending order of their processing times.
(iii) **While** (($GES \neq \Phi$ **And** $GER \neq \Phi$)) **do**
For i = 1 **To** # (GER) **do**
(a) Sort tasks of first node of G_j by selection criterion and communication cost,
(b) Transfer the higher priority task from first node of G_j to i^{th} cluster of GER,
(c) Update the current workloads of source and receiver clusters,
(d) Update sets GES, GER and GEN,
(e) **If** ($GES = \Phi$ **OR** $GER = \Phi$) **then** Return **end If**
(f) Sort GES by descending order of their processing times.
end For
done
end For

5. Experimental study

In order to evaluate the performance of the strategy we have implemented our algorithms on the ***GridSim*** simulator [2], which we extended to support simulation of varying Grid load balancing problems.
The experiments were performed, based on the variation of several performance parameters in a Grid, namely the number of clusters, their worker nodes and the number of tasks.
We focused on the following objectives: *Average waiting time*, *Average execution time* and *Average response time*.
To evaluate the benefit realized, we compute the above metrics *before* (denoted Bef) and *after* (Aft) execution of our algorithms. All the experiment were performed on 3Ghz P4 Intel Pentium with 1 GB main memory, running on Linux Redhat 9.0.
After many evaluation tests, various thresholds was set to: $\varepsilon = 0.5$, $\delta = 0.8$ and $\rho = 0.75$.

Experiments 1: Intra-cluster load balancing
In the first set of experiments we focused results relating to objective metrics, according to various numbers of tasks and worker nodes. We have varied the nodes number from 100 to 400 by step of 100. For each node we randomly generate associated speed varying between 5 and 40 MIPS. The number of tasks have been varied from 6000 to 10000 by step of 2000, with sizes randomly generated between 1000 and 200000 MI (Million of Instructions).
Table 1 shows the variation of the average response time (in seconds) before and after execution. We can note the following:

– Proposed strategy allowed to reduce in a very clear way the mean response time of the tasks. We obtain a gain varying between 7% and 57%.
– In more than 95% of cases, this improvement is greater than 15%.
– The lower improvements were obtained when the number of nodes exceed 350. We can justify this by the instability of the Grid state (nodes are largely under-loaded).
– The best gains were realized when the number of nodes was between 200 and 300. In this case, we can say that our strategy is optimal.

Table 1. *Improvement realized by intra-cluster strategy*

# Tasks	# Nodes	100	200	300	400
6000	*Bef*	6.00E+05	3.03E+04	6.44E+03	2.28E+03
	Aft	4.21E+05	1.49E+04	3.81E+03	2.11E+03
	Gain	**30%**	**51%**	**41%**	**7%**
8000	*Bef*	9.06E+05	4.86E+04	1.05E+04	3.63E+03
	Aft	6.46E+05	2.43E+04	5.76E+03	3.04E+03
	Gain	**29%**	**50%**	**45%**	**16%**
10000	*Bef*	1.30E+06	7.42E+04	1.63E+04	5.63E+03
	Aft	9.39E+05	3.93E+04	7.03E+03	4.30E+03
	Gain	**28%**	**47%**	**57%**	**24%**

Experiments 2: Inter-clusters load balancing
During these experiments , we interested to the inter-clusters load balancing behaviors. We have considered different numbers of clusters and we supposed that each cluster involves 30 worker nodes. Figure 2 illustrates the improvement of the mean response time, obtained by our load balancing strategy.

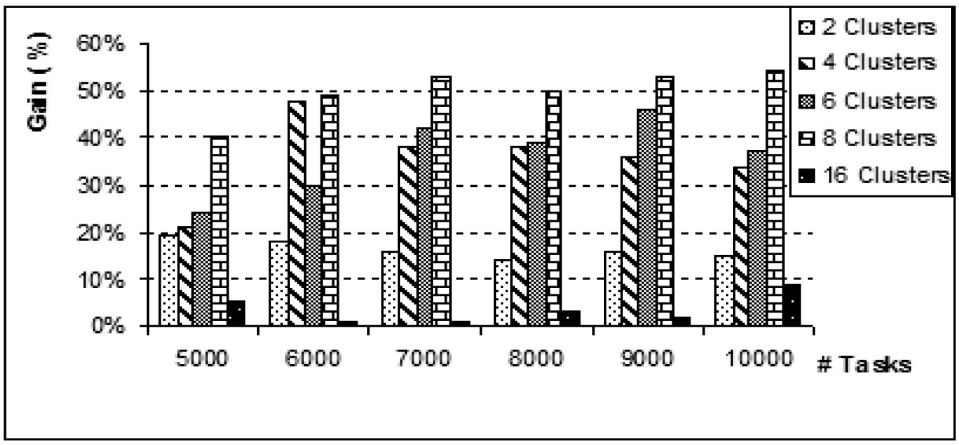

Figure 2. *Gain according to various number of clusters*

– Except the case of 16 clusters, all the profits are higher than 10%. We consider this important behaviour very promising.
– Best improvements are obtained when the Grid is in a stable state: (For # clusters$\in \{4, 6, 8\}$).
– The lower benefits were obtained when the number of clusters were set to 16. We can justify this by the instability of the Grid state (Most nodes are underloaded or even idle).

6. Conclusion

This paper addressed the problem of load balancing in Grid computing. We proposed a distributed load balancing model which takes into account the heterogeneity of the resources and it is completely independent from any Grid physical architecture.
Basing on this model, we defined a distributed load balancing strategy having two main objectives:
(i) Reduction of the mean response time of tasks submitted to a Grid computing;
(ii) Reduction of the communication costs during tasks transferring.
Relatively to the existing works, our strategy is fully distributed, uses a task-level load balancing and privileges, as much as possible, a local load balancing to avoid the use of WAN communication.
The first results obtained on ***GridSim*** simulator are very promising. We have appreciably improved the average response time with a weak communication cost.
In the future, we plan to integrate our load balancing algorithm on others known Grid simulators. This will allow us to measure the effectiveness of our algorithm in existing simulators. We also envisage developing our algorithm as a service of GLOBUS [4]. Finally,it is significant to take account, in a balancing algorithm of application characteristics. More these characteristics are known, more the algorithm will be adapted, which suggests adapting a strategy of balancing to a class of applications.

7. References

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

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

[3] 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, num. 3, Pages:187–200, 2000.

[4] FOSTER I. , "Globus toolkit version 4: Software for service oriented systems", *IFIP: International Conference on Network and Parallel Computing*, pages 2–13, Beijing, China, November 2005.

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

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

[7] 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, 199

[8] 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.

[9] 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.

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

[11] YAGOUBI B., "Modèle d'équilibrage de charge pour les grilles de calcul", *Revue Africaine de la Recherche en Informatique et Mathématiques Appliquées: ARIMA*, vol 7:1–19, Juin 2007.