A Simulated Annealing Method to Cover Dynamic Load Balancing in Grid Environment

Mauricio Paletta[1] and Pilar Herrero[2]

[1] Departamento de Ciencia y Tecnología, Universidad Nacional Experimental de Guayana, Av. Atlántico, Ciudad Guayana, Venezuela
mpaletta@uneg.edu.ve
[2] Facultad de Informática. Universidad Politécnica de Madrid, Campus de Montegancedo S/N, 28.660 Boadilla del Monte, Madrid, Spain
pherrero@fi.upm.es

Abstract. High-performance scheduling is critical to the achievement of application performance on the computational grid. New scheduling algorithms are in demand for addressing new concerns arising in the grid environment. One of the main phases of scheduling on a grid is related to the load balancing problem therefore having a high-performance method to deal with the load balancing problem is essential to obtain a satisfactory high-performance scheduling. This paper presents SAGE, a new high-performance method to cover the dynamic load balancing problem by means of a simulated annealing algorithm. Even though this problem has been addressed with several different approaches only one of these methods is related with simulated annealing algorithm. Preliminary results show that SAGE not only makes it possible to find a good solution to the problem (effectiveness) but also in a reasonable amount of time (efficiency).

Keywords: Grid Computing, load-balancing, Simulated Annealing.

1 Introduction

Grid Computing (GC) [5, 11] emerged in the mid 1990s as an alternative to design distributed computing infrastructure for sharing resources. With the evolution of the computational grid, new scheduling algorithms are in demand for addressing new concerns arising in the grid environment. In this environment the scheduling problem is to schedule a stream of applications from different users to a set of computing resources to maximize system utilization [10].

But, the complexity of scheduling problem increases with the size of the grid and becomes highly difficult to solve effectively, so that the developed of techniques that provide an optimal or near optimal solution for large grids has been required [9]. In this sense, high-performance schedulers become a critical part of the programming environment for the computational grid, and therefore an area of very active research and development is evolving [3]

In the same order of ideas, there are three main phases of scheduling on a grid [8]: 1) resource discovery, which generates a list of potential resources; 2) gathering information about those resources and choosing the best set to match the application requirements (load-balancing problem); 3) task execution, which includes file staging and cleanup.

A load-balancing strategy, which relates with the second phase of scheduling on a grid, deals with partitioning a program into smaller tasks that can be executed concurrently and mapping each of these tasks to a computational resource (this problem is

J.M. Corchado et al. (Eds.): DCAI 2008, ASC 50, pp. 1–10, 2009.
springerlink.com

also known as task allocation or mapping problem). An efficient load balancing strategy avoids the situation where some resources are idle while others have multiple jobs queued up. Therefore, by developing strategies that can map these tasks to resources in a way that balances out the load, the total processing time will be reduced with improving the utilization of these resources.

In this sense, having a high-performance method to deal with the load-balancing problem related with the scheduling on a grid is an important part the programming environment for the computational grid should take into consideration.

On the other hand, and due to the fact that load-balancing is a combinatorial optimization problem the use of heuristics is useful to cope in practice with its difficulty [9]. Most of the research on load-balancing strategy in particular, and scheduling in general, focused on static scenarios that, in most of the cases, employ heuristic methods (see next section for details). Although static techniques have proved effectiveness, the dynamic nature of the grid requires effective and efficient dynamic allocation of available resources techniques, even though there is often a trade-off between solution quality and speed in achieving a solution [18].

This paper presents a new heuristic method designed to solve the dynamic load-balancing problem in grid environments. This method, called SAGE (Simulated Annealing to cover dynamic load balancing in Grid Environment), is a dynamic load-balancing method based on Simulated Annealing (SA) algorithm that is properly configured and developed whereby optimal or near-optimal task allocations can "evolve" during the operation of the grid system.

As part of the SA configuration, a valid representation of the current state of the grid environment must be defined. In this case the specifications given by Herrero et al in [13, 14] are used to do this. Thus, the results of preliminary experiments obtained using a test application and presented in this paper may be associated with possible results to achieve in real or simulated grid environments. These results show that SAGE not only makes it possible to find a good solution to the problem but also in a reasonable amount of time, obtaining the high-performance feature it is looking for.

SAGE could be integrated in any grid model as CAM [14] (Collaborative/Cooperative Awareness Management), specifically in the scheduling management component, with the aim to give an alternative way to deal with the load-balancing problem in grid environments. So that, users from existing grid models, in general, may benefit with another way to deal the load-balancing problem and decide which algorithm to use according to the results obtained.

The paper is organized as follow. Section 2 presents the existing related work in the area, a briefly description of the SA algorithm is presented in section 3, section 4 presents SAGE, results of some experimental tests are presented in section 5. Finally, section 6 exposes the paper conclusions as well as the future work related with this research.

2 Related Work

Intensive research has been done in the area of scheduling algorithms for grid computing and many results have been widely accepted. An interesting reading on the state of the art about this area is given by Fangpeng et al in [7].

Some heuristic approaches for this problem include local search [20], Tabu Search [1], genetic algorithms [1, 12, 22, 24], Ant Colony Optimization (ACO) algorithm [10, 17], intelligent agents [6, 13], particle swarm optimization [2], fuzzy based scheduling [16], economic-based approaches [5], Multi-Objective Evolutionary Algorithm (MOEA), and Great Deluge (GD) algorithm [18]. Regard to SA who is the subject in which the proposal presented in this paper was designed, some approaches can be reviewed in [1, 4, 21, 23]. The main difference between these previous SA-based works and SAGE is that they are offline methods (tasks are collected in a set and after that they are scheduled) and SAGE is designed to be an online method (every time a new task has to be executed it is scheduled based on current conditions).

Next section presents a briefly description of the SA method including the elements that are needed to define for solving combinatorial optimization problems using this method.

3 SA Method

SA [15, 19] is a random-search technique (a generalization of a Monte Carlo method) which exploits an analogy between the way in which a metal cools and freezes into a minimum energy crystalline structure (the annealing process) and the search for a minimum in a more general system; it forms the basis of an optimization technique for combinatorial and other problems. The concept originates from the way in which crystalline structures are brought to more ordered states by an "annealing process" of repeated heating and slowly cooling the structures.

In SA, a system is initialized at a temperature T with some configuration whose energy is evaluated to be E. A new configuration is constructed by applying a random change, and the change in energy dE is computed. The new configuration is unconditionally accepted if it lowers the energy of the system. If the energy of the system is increased by the change, the new configuration is accepted depending on the probability distribution of Boltzmann [19]; this process is repeated sufficient times at the current temperature to sample the search space; then the temperature is decreased; the process is repeated at the successively lower temperatures until a frozen state is achieved.

This procedure allows the system to move to lower energy states, while still jumping out of local minima (especially at higher temperatures) due to the probabilistic acceptance of some upward moves. SA theory states that if temperature is lowered sufficiently slowly (by carefully controlling the rate of cooling), the solid will reach thermal equilibrium, which is an optimal state (global optimum).

To solve a problem through this strategy it is necessary to define the following elements:

1) The representation of a valid solution to the problem.
2) The cost function (energy) that measures the quality of each solution.
3) The mechanism of transition of the space of solutions from t to t+1 (dynamic of the model).
4) The parameters that control the rate of cooling.

Next section presents the way in which these four elements are defined according to the load-balancing problem in grid environments to have a high-performance method to cover this problem.

4 SAGE: A SA Based Method to Cover Load Balancing in Grid Environments

4.1 The Grid Model

The grid specifications considered in this study can be reviewed in detail in [13, 14]. The distributed environment *DE* is composed of n hosts h_i ($1 \le i \le n$) each composed of several computational resources r_j each of a specific type (CPU power, memory, disk space, knowledge, etc.). If the following statements are considered:

1) There is a maximum of m types or classes of resources ($1 \le j \le m$).
2) A particular host h_i may have more than one resource of the same type.
3) A particular host h_i may not have a resource for a specific type.

$$\overline{DE} = \begin{vmatrix} & r_1 & \cdots & r_m \\ h_1: & f_{11} & \cdots & f_{1m} \\ \vdots & \vdots & \ddots & \vdots \\ h_n: & f_{n1} & \cdots & f_{nm} \end{vmatrix} \tag{1}$$

$$f_{ij} = \begin{cases} Maximum, \text{entire resource is available} \\ Medium, \text{part of the resource is available} \\ Null, \text{the resource is saturated or is not available} \end{cases} \tag{2}$$

Then and as can be seen in (1), *DE* can be represented using a matrix with n rows and m columns; each row represents a host with the corresponding vector of resources; each column is associated with a specific type of resource. In this sense a particular item f_{ij} of the matrix corresponds to the degree in which the resource type r_j of the host h_i is being used at a given time. As can be seen in (2) there are three possible values for setting each f_{ij}: *Maximum* when h_i has at the disposal all the resource associated to the type r_j; *Medium* when h_i has at the disposal only a part of the resource associated to the type r_j, and *Null* when h_i has not resources r_j at the disposal either because it is saturated or because it is not available in h_i.

A task T who needs resources from this grid-based system is composed by a collection of processes p_k each associated with the resource type necessary to complete the specific process. So that, T can be seen as a vector of p elements each associated to a valid value that represents a resource type:

$$\overline{T} = (p_1, \ldots, p_p);\ \forall k, 1 \le k \le p, 1 \le p_k \le m \tag{3}$$

In this sense, the load-balancing problem related to this system is as follows: for any task T, looks a valid relationship (given by the resource type) between all the processes p_k of T according to the current state of *DE*, and considering the following statements:

1) It must be done dynamically, so in a reasonable time.
2) It must be done considering the load-balancing of each host.
3) It must be done considering the load-balancing of each resource type.

In other words, it is necessary to find, in a efficient, effective and dynamical way, what $f_{ij}(t)$ must be changed for each p_k in order to obtain the new state of *DE* or $DE(t+1)$.

4.2 Representing a Valid Solution

A solution S for this problem is represented with a vector of size p; each element s_l ($1 \leq l \leq p$) has the host in which the process p_l of T has been assigned. So that, the value of s_l is the row index i and the value of p_l is the column index j for the element f_{ij} of *DE* in which the process l has been assigned.

This way, having the task $T = (p_1,\ldots,p_p)$ and the solution $S = (s_1,\ldots,s_p)$ it is possible to know all the f_{ij} involved in the distribution of the p processes of T among the n hosts of *DE*.

4.3 Measuring a Solution

As was previously read in Section 3, one of the elements to define for using the SA strategy is the cost function or energy that measures the quality of a specific solution. In this sense the following statements are considered:

1) While lower energy (system temperature) is better quality in the solution.
2) Energy is directly proportional to the current state of the resources.
3) There should be a restriction to balance the type of resources of each host.
4) There should be a restriction to balance the hosts of each type of resource.

First, it is necessary to define values for measuring the current state of the resources. Due to the fact that energy or solution quality is directly proportional to the current state of the resources, these values are defined according to the following guidelines:

1) Reward (low or no energy) to those resources that are full available.
2) Punish (high energy) those that are saturated.
3) Severely punish (very high energy) invalid situations (a solution which assigns a process to a saturated or not available resource).

If "τ(x)" is the function that transforms the current state f_{ij} of a resource then:

$$\tau(f_{ij}) = \begin{cases} 0, f_{ij} = Maximum \\ 1, f_{ij} = Medium \\ 2, f_{ij} = Null \\ 5, \text{other case} \end{cases} \tag{4}$$

Therefore, minimum energy ε_m (top quality) from the current state of *DE* depends on the accumulated resource use:

$$\varepsilon_m(DE) = \sum_{i,j}^{n,m} \tau(f_{ij}) \tag{5}$$

To define restrictions for balancing the type of resources of each host and the hosts of each type of resource, Th_i is the current resource accumulated for the host h_i (sum of the elements of the row i in DE) and Tr_j is the current host accumulated for the resource type r_j (sum of the elements of the column j in DE):

$$Th_i = \sum_{j=1}^{m} \tau(f_{ij}) \tag{6}$$

$$Tr_j = \sum_{i=1}^{n} \tau(f_{ij}) \tag{7}$$

Therefore, for balancing the resources from the same host h_i their values are in the order of Th_i / n, and for balancing the hosts from the same resource type r_j their values are in the order of Tr_j / m. Any difference to these values must be considered in the calculation of the restrictions. Using: 1) squares to handle the negative differences, 2) adjustment factors μ and δ to determine the degree of importance which we want to consider each restriction, and 3) divided between 2 to facilitate the derivation of the term, the restrictions Ψ_1 and Ψ_2 are defined as follow:

$$\Psi_1 = \frac{\mu}{2}\left(\sum_{j=1}^{m}\left(\sum_{i=1}^{n}(Tr_j/n - \tau(f_{ij}))^2\right)\right) \tag{8}$$

$$\Psi_2 = \frac{\delta}{2}\left(\sum_{i=1}^{n}\left(\sum_{j=1}^{m}(Th_i/m - \tau(f_{ij}))^2\right)\right) \tag{9}$$

Finally, for a current situation of DE at time t and a specific solution S the energy of the system with these conditions is calculated as follow:

$$\varepsilon(DE(t), S) = \frac{1}{2}\varepsilon_m(DE) + \Psi_1 + \Psi_2 \tag{10}$$

4.4 The Mechanism of Transition and Cooling Control Parameters

The rules governing change of DE between time t to t+1 in the dynamic of the system affect only those resources f_{ij} that have been assigned with any process in the optimal solution S obtained once a SA iteration was executed. In this sense, the new values are calculated according to the following statements:

1) $f_{ij}(t) = Maximum \Rightarrow f_{ij}(t+1) = Medium$.
2) $f_{ij}(t) = Medium \Rightarrow f_{ij}(t+1) = Null$.
3) $f_{ij}(t)$ = other case $\Rightarrow f_{ij}(t+1)$ = other case.

The first mapping or tentative solution S is generated from a uniform random distribution. The mapping is mutated randomly changing the host of a specific process, and the new solution is evaluated. If the new solution is better (less energy), it replaces the old one. If the new solution is worse, it is accepted depending on the probability distribution of Boltzmann. After each mutation, the system temperature is reduced to 95% of its current value. The heuristic stops when there is no change in the solution for 5 iterations or the system temperature approaches a value closes to zero

or the total of iterations reaches the value of n x m x p. The initial value of the temperature is equal to n x m.

Next section presents some experimental results based on the efficiency and effectiveness of the method previously defined.

5 Experimental Results

The results presented in this section are taken from a testing experimental application which is possible to indicate the number of hosts n, the number of resources types m, the maximum number of processes p for a task T, the initial state (null or random) of DE, and the values for the adjustment factors μ and δ for both restrictions. The hardware where the tests were done is 2.16GHz and 2 GB RAM.

Fig. 1 shows a screen shot of the testing application. At the bottom left is the matrix DE in the current state (6 hosts and 10 resource types in the example); at the bottom right are the current results of balancing including run time in the process execution (1 sec.). As it can be seen in the results all processes were assigned to resources that were in a state equal to *Maximum* (value of 0) which is the optimal situation for this problem.

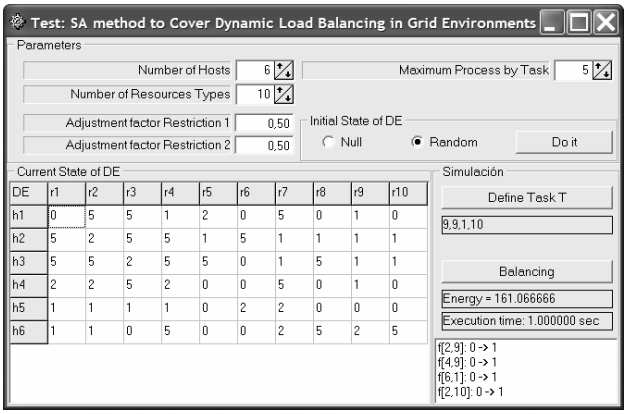

Fig. 1. A screen shot of the application for testing the method

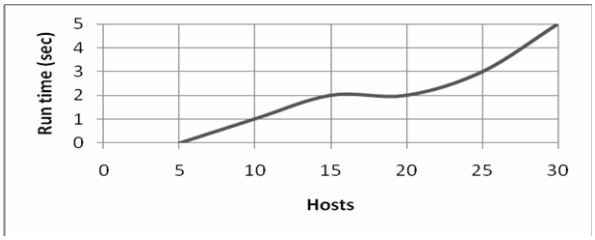

Fig. 2. Execution times with $m = 10$, $p = 5$ and $n \in \{5,10,15,20,25,30\}$

It is important to mention that all parameters can be changed dynamically previous to execute the balancing algorithm, as for example to increment the number of hosts in the system. Fig. 2 shows the execution times (in seconds) to obtain an optimal solution or close to the optimum of a grid with 10 different types of resources, tasks composed by 5 processes and varying the number of hosts from 5 to 30 (5 in 5).

6 Conclusions and Future Work

In this paper we present SAGE, a SA based method to cover dynamic load-balancing problem in Grid Environments. The results show that the heuristic strategy proposed finds good solutions (closer to the optimum) in a reasonable time (a few seconds). Therefore, this algorithms guarantees efficient and effective load-balancing of any task requires resources from a grid-based system and it also can be applied in dynamic way.

Even though this method has not yet been tested neither in a real grid environment nor a grid environment simulation, the manner in which the current state of a grid environment is represented is suitable for real grid environments. Therefore it is expected that the results of preliminary experiments are also achieved in real scenarios.

We are working on the term for calculating the energy that can be adapted to meet other needs in the scheduling problem of a grid. As a future work we will handle a large scale of possibilities for the degree in which the resource types are being used at a given time, as for example using fuzzy logic techniques. We are also working on integrating this algorithm in simulated applications schedulers by using grid modeling and simulating toolkit to obtain more accurate results in experiments and compare them with other similar work.

References

1. Abraham, A., Buyya, R., Nath, B.: Nature's Heuristics for Scheduling Jobs in Computational Grids. In: Sinha, P.S., Gupta, R. (eds.) Proc. 8th IEEE International Conference on Advanced Computing and Communications (ADCOM2000), pp. 45–52. Tata McGraw-Hill Publishing Co. Ltd, New Delhi (2000)
2. Abraham, A., Liu, H., Zhang, W., Chang, T.G.: Scheduling Jobs on Computational Grids Using Fuzzy Particle Swarm Algorithm. In: Proc. of 10th International Conference on Knowledge-Based & Intelligent Information & Engineering Systems, England, pp. 500–507 (2006)
3. Berman, F.: High-performance schedulers. In: The Grid: Blueprint for a New Computing Infrastructure, pp. 279–309. Morgan Kaufmann, San Francisco (1999)
4. Braun, R., Siegel, H., Beck, N., Boloni, L., Maheswaran, M., Reuther, A., Robertson, J., Theys, M., Yao, B., Hensgen, D., Freund, R.: A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems. Journal of Parallel and Distributed Computing 61(6), 810–837 (2001)
5. Buyya, R., Abramson, D., Giddy, J., Stockinger, H.: Economic Models for Resource Management and Scheduling in Grid Computing. Journal of Concurrency and Computation: Practice and Experience 14(13-15), 1507–1542 (2002)

6. Cao, J., Spooner, D.P., Jarvis, S.A., Nudd, G.R.: Grid load balancing using intelligent agents. Future Generation Computer Systems 21(1), 135–149 (2005)
7. Fangpeng, D., Selim, G.A.: Scheduling Algorithms for Grid Computing: State of the Art and Open Problems Technical Report No. 2006-504, Queen's University, Canada, 55 pages (2006), http://www.cs.queensu.ca/TechReports/Reports/2006-504.pdf
8. Fernandez-Baca, D.: Allocating Modules to Processors in a Distributed System. IEEE Transactions on Software Engineering 15(11), 1427–1436 (1989)
9. Fidanova, S.: Simulated Annealing for Grid Scheduling Problem. In: Proc. IEEE John Vincent Atanasoff International Symposium on Modern Computing (JVA 06), 10.1109/JVA.2006.44, pp. 41–44 (2006)
10. Fidanova, S., Durchova, M.: Ant Algorithm for Grid Scheduling Problem. In: Lirkov, I., Margenov, S., Waśniewski, J. (eds.) LSSC 2005. LNCS, vol. 3743, pp. 405–412. Springer, Heidelberg (2006)
11. Foster, I., Kesselman, C., Tuecke, S.: The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International Journal of Supercomputer applications and High Performance Computing 15(3), 200–222 (2001)
12. Grosan, C., Abraham, A., Helvik, B.: Multiobjective Evolutionary Algorithms for Scheduling Jobs on Computational Grids. In: Guimares, N., Isaias, P. (eds.) IADIS International Conference, Applied Computing 2007, Salamanca, Spain, pp. 459–463 (2007) ISBN: 978-972-8924-30-0
13. Herrero, P., Bosque, J.L., Pérez, M.S.: An Agents-Based Cooperative Awareness Model to Cover Load Balancing Delivery in Grid Environments. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM-WS 2007, Part I. LNCS, vol. 4805, pp. 64–74. Springer, Heidelberg (2007)
14. Herrero, P., Bosque, J.L., Pérez, M.S.: Managing Dynamic Virtual Organizations to get Effective Cooperation in Collaborative Grid Environments. In: Meersman, R., Tari, Z. (eds.) OTM 2007, Part II. LNCS, vol. 4804, pp. 1435–1452. Springer, Heidelberg (2007)
15. Kirkpatrick, S.: Optimization by Simulated Annealing: Quantitative Studies. Journal of Statistical Physics 34(5-6), 975–986 (1984)
16. Kumar, K.P., Agarwal, A., Krishnan, R.: Fuzzy based resource management framework for high throughput computing. In: Proc. of the 2004 IEEE International Symposium on Cluster Computing and the Grid, pp. 555–562 (2004)
17. Lorpunmanee, S., Sap, M.N., Abdullah, A.H., Chompoo-inwai, C.: An Ant Colony Optimization for Dynamic Job Scheduling in Grid Environment. International Journal of Computer and Information Science and Engineering 1(4), 207–214 (2007)
18. McMullan, P., McCollum, B.: Dynamic Job Scheduling on the Grid Environment Using the Great Deluge Algorithm. In: Malyshkin, V.E. (ed.) PaCT 2007. LNCS, vol. 4671, pp. 283–292. Springer, Heidelberg (2007)
19. Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E.: Equations of state calculations by fast computing machines. Journal of Chemical Physics 21(6), 1087–1091 (1953)
20. Ritchie, G., Levine, J.: A fast, effective local search for scheduling independent jobs in heterogeneous computing environments, Technical report, Centre for Intelligent Systems and their Applications, School of Informatics, University of Edinburgh (2003)
21. Yarkhan, A., Dongarra, J.: Experiments with Scheduling Using Simulated Annealing in a Grid Environment. In: Parashar, M. (ed.) GRID 2002. LNCS, vol. 2536, pp. 232–242. Springer, Heidelberg (2002)

22. Ye, G., Rao, R., Li, M.: A Multiobjective Resources Scheduling Approach Based on Genetic Algorithms in Grid Environment. In: Fifth International Conference on Grid and Cooperative Computing Workshops, pp. 504–509 (2006)
23. Young, L., McGough, S., Newhouse, S., Darlington, J.: Scheduling Architecture and Algorithms within the ICENI Grid Middleware. In: Proc. of UK e-Science All Hands Meeting, Nottingham, pp. 5–12 (2003)
24. Zomaya, A.Y.: The YH (2001) Observations on using genetic algorithms for dynamic load-balancing. IEEE Transactions On Parallel and Distributed Systems 12(9), 899–911 (2001)