A Novel Load Balancing Method in an Agent-based Grid

Mohsen Amini Salehi
Computer Department, Faculty of Engineering,
Ferdowsi University of Mashhad,
Mo_am88@stu-mail.um.ac.ir

Hossain Deldari
Computer Department, Faculty of Engineering,
Ferdowsi University of Mashhad,
hd@um.ac.ir

Abstract: **A computational grid is a widespread computing environment which provides huge computational power for large-scale distributed applications. One of the most important issues in such an environment is resource management. Load balancing which is a part of resource manager, has a considerable effect on resources' performance. There are several factors which affect the stability of the load balancing in the grid. Among these factors, accurate criterion for estimating the workload, considering the workload transmission cost, validity of the information about other nodes and the overhead imposed by the method are more effective. In this paper we propose a novel method for load balancing which tries to satisfy these factors. The proposed method is implemented on an agent-based resource management system, called ARMS. There are several simulations which indicate that the proposed method outperforms than similar methods.**

Key-words: **Grid computing, Load Balancing, Agent-based Resource management system (ARMS), Virtual routing, PACE.**

I. INTRODUCTION

Grid infrastructure provides us with the ability to dynamically link together resources as an ensemble to support the execution of large-scale, resource-intensive, and distributed applications [1]. Grid permits users to interface with the resources uniformly, providing a powerful platform for global computing. Resource management and scheduling are key issues in this platform [2, 4]. An ideal resource manager should efficiently utilize all of the grid resources [4, 5].

Load balancing is an important issue for the problem of utilization. Load balancing are those mechanisms which are designed to equally spread the load on resources and maximize their utilization [10]. These mechanisms can be broadly categorized as centralized or decentralized, dynamic or static, periodic or non-periodic [10].

Most of current load balancing methods in the grid environment are borrowed from those in distributed systems [13]. However lack of a common/realistic definition for load [6], communication overhead and ignoring transmission cost [8] are crucial limitations for most of current load balancing methods in grid and distributed systems. Furthermore vastness and high degree of dynamism are two additional difficulties in the grid environment.

Frequent workload variation is a major characteristic of the grid dynamism [1,2]. Therefore, for the sake of accuracy in load balancing methods, it is necessary to update load information frequently. However, frequent updating in the grid, leads to enormous communication overhead. In other words, here we have a trade-off between reliability and scalability [8]. In [16] Dahlin proposed a load estimating method (instead of load exchanging) which drastically reduces the load exchanging needed. Though, utilizing load estimation method needs a more precise load definition.

In [8] Dhaka1 et al. proposed a combinatorial definition for load. However In spite of more accuracy and estimation capability, it finally considers (counts) the number of jobs, which is not a precise scale for load measuring.

QLBVR [3] is a multi-level load balancing method for distributed systems which balances the load among nodes according to their queue length and request arrival rate. It also uses virtual routing method for checking the scattering profitability. Virtual routing changes load balancing difficulty to an optimal routing problem in the following way.

Virtual routing adds a *virtual* node, which is called as the *destination* node (node *d*), in the network system. Connect node *d* with each neighboring node *i* by a virtual direct link (*i, d*). Let the nodal delay of node *i* be considered as the communication delay on link (*i, d*). As shown in Fig. 1, a three-node system has been converted into a datagram network, in which node *i* essentially acts as a *router*. The loads which arrive at node *i,* are routed to destination node *d* via every node $j \neq i$, with the goal of minimizing the mean link delay. Then, each node *i* in the system only considers the routing paths of (*i, d*) and *(i, j)* → *(j, d)* where *j* is a neighbor of *i.*

This work was supported by the Iran Telecommunication Research Center (ITRC).

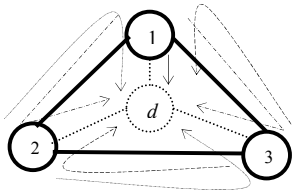


Fig. 1.Routing paths for each node

Considering the grid specifications and the above limitations, in this work we attempt to propose a novel method that does not need a lot of information to be exchanged, taking in to account a more accurate definition for load as well as considering transmission cost. This method is implemented in an agent-based resource manager called ARMS. We will discuss ARMS and the proposed method in detail, during the next sections.

The rest of the paper is organized as follows: Section 2 introduces the agent-based resource management system (ARMS). In Section 3, neighbor level load balancing method is described. The performance metrics and simulation results are included in Section 4. At the end, we present the conclusion of the article as well as the future works which can be done in the same direction as the inclination of this research.

II. Agent-based Resource Management

As mentioned above, Resource management is an important infrastructural component of a grid computing environment. Agent-based Mechanisms are considered to be suitable for grid resource management, since agents have the capability to control their own knowledge rather than relying on a fixed function query engine [2, 4].

ARMS is an Agent-based resource manger for grid computing which is aimed at provisioning scalability and adaptability [2]. In ARMS, each agent can simultaneously stand for a resource questioner, resource provider, and also a matchmaker. These agents cooperate with each other to achieve resource discovery.

An agent in the system can have many local resources that can provide services. The agent can take them as its own capabilities. It must decide how and when to advertise this service information to other neighboring agents. An agent can also receive many service advertisements from neighboring agents. However, service information can also be propagated to a large area after many steps of advertisement over a period of time. All of the service information is organized into Agent Capability Tables (ACTs) in each agent. The service information contains all performance related information of a grid resource, which can be used to estimate its performance. Therefore the agents are equipped with a performance prediction toolkit called PACE [5, 9, 14].

Application performance prediction provides the important functionality that enables the grid load balancing capabilities described in this research. The PACE toolkit [5] is used to supply this ability for both the local schedulers and the grid agents. The main components of the PACE toolkit include application tools (AT), resource tools (RT), and an evaluation engine (EE). The PACE evaluation engine can mingle application and resource models at execution time to produce performance data e.g. total execution time. In the work described in [9] an ASCI (Accelerated Strategic Computing Initiative) core application, Sweep3D, is used to exemplify the performance prediction capabilities of PACE. The validation results illustrate that a high level of precision can be attained, and the process benefits from a quick evaluation time. These characteristics allow PACE forecasting data to be used on-the-fly for grid resource load balancing.

While an ARMS system can achieve grid load balancing as a result of trying to meet QoS requirements specified explicitly by users [13], neighbor level load balancing is investigated in this research for agents to achieve load balancing for batch queuing jobs which are not explicitly coupled with execution deadlines.

In [4, 15] there are load balancing mechanisms on ARMS using ant colony optimization, however the basics of our work are different in the following ways:

- The new approach is a prevention mechanism while [4] is detection and recovery one. This implies that in the new approach we initially try to impede the unbalance.
- Ant colony is an offline mechanism, while the new one is online. In other words in the former, there might be nodes that had not been met for a long time [4], however in the new approach, all the nodes cooperate in the load balancing process periodically.

Neglecting the transmission cost and high communication overhead is another deficiency in [4].

However our mechanism and the previous one are not oppositions, instead they could be synergetic.

In the next section we propose our new mechanism.

III. Neighbor Level Load Balancing Using Virtual Routing

Considering stated disadvantages for current load measurement methods, we first provide a more accurate load measurement/estimation method which

relies on the time needed for executing current jobs (instead of number of current jobs). Then we propose our new load balancing method based on this new measurement/estimation policy.

In the ARMS, agents obtain their resource capabilities using PACE and exchange them with their neighbors periodically. An agent advertises its load information only to its neighbors, for the purpose of scalability needed in the grid [11]. It is possible to attach load characteristics of the nodes to this exchanging information.

Here, we consider the total execution time (gained through PACE evaluation engine in each node), average of the job arriving rate and job completion rate (considering the number of arrivals/completions in a certain fixed interval of time in each node) as the load information. This information helps to provide a more accurate measurement as well as estimation for load as follows:

$$L_i(t+\Delta t) = L_i(t) - C_i(t+\Delta t)\overline{T} + j_i(t,t+\Delta t)\overline{T} \quad (1)$$

Where $L_i(t)$ shows the total time needed for executing current waiting jobs, $C_i(t+\Delta t)$ indicates the number of jobs completed in the interval $[t,t+\Delta t)$. $j_i(t,t+\Delta t)$ is the number of jobs arriving in the same interval. Finally $\overline{T}$ is a constant coefficient which shows the average execution time.

Moreover the receiver agent can estimate the time gap between sending and receiving the information and applies it as an approximation for transmission cost between the two nodes.

The agents in the system exchange their status information at periodic interval of time T_s, which is called the *status exchange interval*. The instant at which this information exchange happens is called a *status exchange epoch*. Each status exchange interval is further divided into identical subintervals denoted as *estimation intervals Te*. The points of division are called *estimation epochs,* in which each node estimates its neighbors load according to (1).

In Fig. 2, T_{n-1}, T_n stand for the status exchange epochs and t_1, t_2 denote the estimation epochs. These estimation epochs postpone the status exchange epochs and therefore communication overhead decreases.

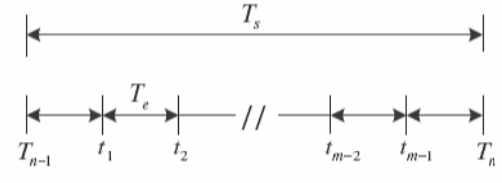


Fig. 2 .Intervals of estimation and status exchange [3].

These exchanges are not dedicated to the load balancing method and are inherently used in resource advertisement in ARMS. We opportunistically attach our favorite fields to them. Therefore we assert that the method has a few communications overhead on the whole system.

Now, the load scheduling decision is taken as follows: utilizing the load information received in the last T_s and using (1), the agents estimate the current load of their neighbors in each estimation epoch while estimation is not required in the exchange epochs. Then the agent computes the average load on its neighboring agents. An agent calls itself "*overloaded*" if its load is greater than the average load of its neighbors. Agents in the neighboring set, whose estimated load is less than the estimated average load by more than a threshold θ, form an *active set*. However, because of the transmission cost, sending the agent's load to all of the active set members might not be profitable [8]. We compass the problem of profitability using virtual routing and try to send the extra loads to those members of the active set which mostly satisfy the profitability. The sender each time finds a member of the active set which has the most profit (result in less response time) and then it sends its jobs (extra load) to the selected member of the active set.

It is clear that most of the equalizations occur only between nearby agents. However, load can be spread to a large area after many steps of equalizations over a period of time.

It is probable that an underloaded agent situated in active set of two or more than two overloaded agents simultaneously. In these circumstances, overloaded agents may send their extra load to the underloaded at the same time and make the underloaded agent, overloaded. Hence this condition causes instability for the proposed method. We use a locking technique to avoid these situations. Therefore each agent only sends its load information to one requester, and does not respond to any other agent at the same time. This continues until the agent dismisses by the requester.

IV. Performance Evaluation

In performance evaluation described in this section, we investigate several common statistics to show performance of the method described.

A. Performance Metrics

There are a number of performance metrics that can be used to describe grid resource management and scheduling systems. In this work several communal statistics are investigated and used to describe the effect of grid load balancing.

Let P be the number of agents of an ARMS system and W_{pk} (p=1,2,……,P) be the workload of the agent p at the period k. The average workload of all P agents is:

$$\overline{W_k} = \frac{\sum_{p=1}^{P} W_{pk}}{P} \qquad (2)$$

The mean square deviation of W_{pk} that characterizes the load balancing level of the system is defined as:

$$L_k = \sqrt{\frac{\sum_{p=1}^{P} (\overline{W_k} - W_{pk})^2}{P}} \qquad (3)$$

It is obvious that load balancing can not be achieved without any overhead. In at most all methods the load balancing process imposes additional network connections among nodes. However as mentioned earlier, this communication overhead is minimized in this method due to the ARMS characteristics and load estimation method. Nevertheless, in system load balancing efficiency (e) is another metrics that takes consideration of system costs for load balancing. Let T_k be the total time spent in all agents to achieve a load balancing level L_k. The system efficiency e_k is represented using the average of an agent connection time to load balancing level upturn during the latest period and calculated as follows:

$$e_k = \frac{L_0 - L_k}{T_k} \qquad (4)$$

Most of the time, these metrics explained recently are conflictive, that is not all metrics can be elevated at the same time. For example, a high load balancing level does not indicate high efficiency, as sometimes good load balancing may be accomplished through too many load balancing operations, resulting in low system efficiency.

B. System Modeling

In this work, agent systems and the method are simulated in a simplified way using several aspects of parameters sufficient to outline system characteristics statistically.

- *Agents*. The agents can be mapped in a graph. This simplification is done in some similar works [12, 13, and 4]. For the sake of simplification we use a 20*20 square to show the graph (grid environment), thus all of experiments described later include 400 agents. In this square each agent might have at most 8 neighbors.
- *Workload*. A workload value and corresponding distribution are used to characterize system workload. Initially the value is generated randomly; however it changes during time with regard to the load balancing, completion rate and arrival rate.
- *Processing rate* and *arrival rate*. In each period, processing (completion) rate and request arrival rate are assigned to each agent randomly.

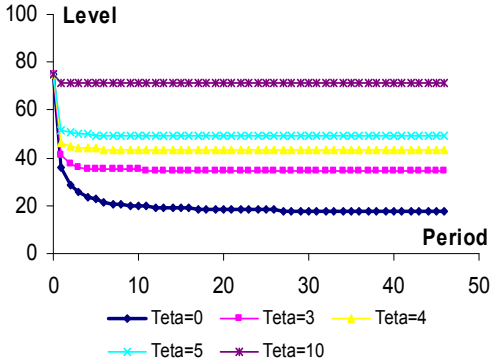
C. Simulation Results

The first experiment illustrates the relation between load balancing level and period count in two different ways. However both of them are repeated for different values of θ .

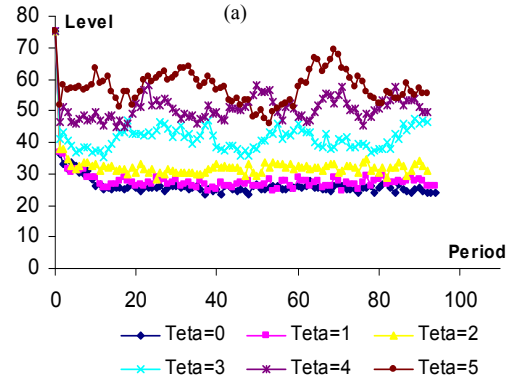
In the first way, we assume that the system load is unvarying during the load balancing process. As shown in Fig. 3.a as the time elapsed, the system approaches to the final convergence. However it doesn't improve more than a specific value. This is the point that the load balancing method prefers to execute jobs locally instead of sending them to the others. In other words, as time elapses a wavelike load distribution is created. As shown in Table1, the balancing level has a few fluctuations in last iterations because of that transmission delay exceeds the executing time locally.

In the second way, according to (1) we assume that the load of each agent changes proportional to the processing rate and arriving rate during the time. We achieved the experiment through hundred periods to show the final balancing level of the system. As shown in Fig. 3.b, even the load of the system fluctuated during the time; the load balancing method keeps the balancing level of the system steady especially when θ approaches to zero.

In the second experiment we concentrate on the effect of virtual routing on saving time spent for load balancing. The comparison is done with an algorithm which does not using virtual routing i.e. does not considering transmission cost. In this algorithm each agent sends the extra load to neighbors with the lowest load. In this work we name such an algorithm a "*Best fit*" algorithm. For the sake of comparison, we calculated the total time spent in all agents for sending the extra load to the neighbors in each period (*Cost*). As illustrated in Fig. 4 the new method spent less time (with different values for θ). This is because the new method considers the transmission overhead while the best fit algorithm does not.



(a)



(b)

Fig. 3. Relation between period count (*Period*) and balancing level (*Level*). (a) Load is constant, (b) load varying, during the time.

TABLE I
PROPORTION OF LOAD AND BALANCING LEVEL WHEN THE LOAD IS CONSTANT

Period(*P*)	Balance Level(*L*)
0	78.60925518
1	39.90507487
2	33.44248795
5	25.11827024
10	22.73581976
15	21.12841688
30	19.38846306
45	18.79893614
50	18.71910788

The last experiment shows the efficiency of the method (for different values of θ) in comparison with the best fit algorithm. It is obvious from (4) that smaller T_k result in better efficiency; however balancing level in *k*th iteration is effective. As shown in Fig. 5, the new method has a better efficiency.

This was predictable from the former experiment; however this experiment shows that while T_k is much smaller than the best fit algorithm, there is no much difference between L_k in both methods.

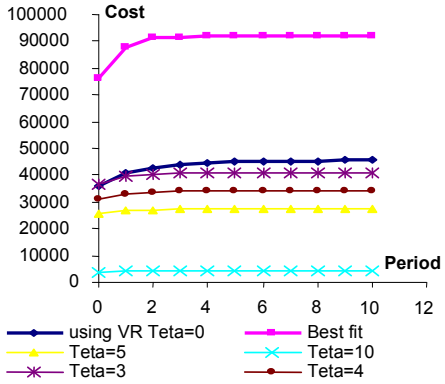


Fig. 4. Time overhead (*Cost*) comparison between using Neighbor level and Best fit algorithm during several periods (*Period*).

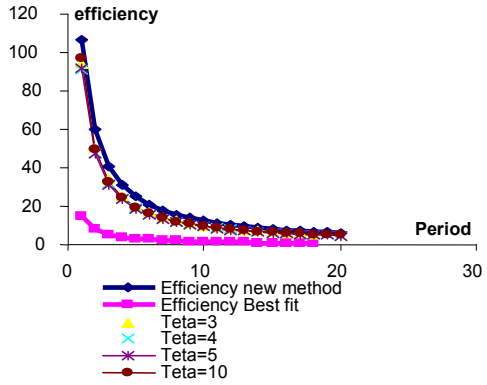


Fig. 5. Efficiency comparison between the new method and best fit algorithm during several periods.

V. CONCLUSION

The main contribution of this work includes providing a load balancing method for agent based grid which considers the time needed for sending jobs to other nodes. This method imposes less overhead to the underlying platform while increasing resource utilization and improving completion time of all jobs.

There are many important features required by a grid computing system and not discussed in this work, e.g. security, billing contracts between agents when they exchange the load of their customers, and data management. Future work will focus on the refinement of the system prototype and the proposed algorithm.

ACKNOWLEDGMENT

We are grateful Mrs. Bahare Mokarram for her helps and Mr. Adel Najjaran for his ideas.

REFERENCES

[1] F. Berman, G. Fox, and T. Hey, eds. John Wiley *Grid Computing: Making the Global Infrastructure a Reality,*and Sons, 2003.

[2] J.Cao, "Agent-Based Resource Management System (ARMS)," *PhD Thesis, Warwick University Dept. of Computer Science,*2001

[3] Z. Zeng, B. Veeravalli, Rate-Based and Queue-Based Dynamic Load Balancing Algorithms in Distributed Systems, "Proc. 10th IEEE Int. Conf. on Parallel and Distributed Systems.

[4] J. Cao, Self-Organizing Agents for Grid Load Balancing, in "Proc. 5th IEEE/ACM Int. Workshop on Grid Computing (GRID'04).

[5] G. R. Nudd, D. J. Kerbyson, E. Papaefstathiou, S. C. Perry, J.S. Harper, and D. V. Wilcox, PACE – a toolset for the performance prediction of parallel and distributed systems, *Int. J. High Performance Computing Applications Vol.* 3, 2000, pp. 228-251.

[6] B. S. Joshi, S. H. Hosseini, K. Vairavan, A Methodology for Evaluating Load Balancing Algorithms, in "Proc. 2nd IEEE Int. High Performance Distributed Computing Symp.", pp.216 – 222, 1993

[7] L. Anand, D. Ghose, and V. Mani, "ELISA: An Estimated Load Information Scheduling Algorithm for Distributed Computing System," *Computers and Mathematics with Applications*, 37, pp. 57-85, 1999.

[8] S**.** Dhaka1, B. S**.** Paskaleva , M**.** M. Hayat, E. Schamiloglu, C. T. Abdallah, Dynamical Discrete-Time Load Balancing in Distributed Systems in the presence of Time Delays, in "Proc. 42nd IEEE Decision and Control Conference" , Vol.5, pp.5128 – 5134, 2003.

[9] J. Cao, D. J. Kerbyson, E. Papaefstathiou, and G. R. Nudd, Performance modelling of parallel and distributed computing using PACE, in "Proc. 19th IEEE International Performance, Computing and Communication Conference", pp. 485-492, Phoenix, AZ, USA, 2000.

[10] A. Y. Zomaya, Y.Teh, Observations on using genetic algorithms for dynamic load-balancing, *IEEE Transactions on Parallel and Distributed Systems* , vol.12, No. 9, 2001, pp. 899-911 .

[11] Orly Kremien, Jeff Kramer. Methodical analysis of adaptive load sharing algorithms, *IEEE Transactions on Parallel and Distributed Systems 3(*Nov.1992*),* pp. 747-760.

[12] S.Viswanathan, B. Veeravalli, D. Yu, G. Robertazzi, Design and Analysis of a Dynamic Scheduling Strategy with Resource Estimation for Large-Scale Grid Systems. In "Proc. 5th IEEE/ACM Int. Grid Computing Workshop", pp.163-170, 2004.

[13] J. Cao, D. P. Spooner, S. A. Jarvis, S. Saini, and G. R. Nudd, Agent-Based Grid Load Balancing Using Performance-Driven Task Scheduling, in "Proc. of 17th IEEE Int. Parallel and Distributed Processing Symp.", Nice, France, 2003.

[14] J. Cao, D. J. Kerbyson, E. Papaefstathiou, and G.R.Nudd, "Modeling of ASCI High Performance Applications Using PACE", in Proc. of 15th Annual UK Performance Engineering Workshop, Bristol, UK, pp. 413-424, 1999.

[15] M. Amini, H. Deldari, "Grid Load Balancing Using an Echo System of Intelligent Ants", in Proc. Of 24th IASTED International Multi-Conference parallel and Distributed Computing and Networks (PDCN06), Innsbruck, Austria, pp. 47-52, 2006.

[16] M. Dahlin, "Interpreting stale load information", *IEEE TRANS. ON PARALLEL AND DISTRIBUTED SYSTEMS,* VOL. 11 (10), OCTOBER 2000, pp. 1033-1047.