Balanced Distribution of Load on Grid Resources using Cellular Automata

Amir Akbarian Sadeghi
Department of Computer Engineering
South Tehran Branch,
Islamic Azad University
Tehran, Iran

Ahmad Khademzadeh
Research Institute for Information & Communication Technology
Tehran, Iran

Mohammad Reza Salehnamadi
Department of Computer Engineering
South Tehran Branch,
Islamic Azad University
Tehran, Iran

***Abstract*—Load balancing is a technique for equal and fair distribution of load on resources and maximizing their performance as well as reducing the overall execution time. However, meeting all of these goals in a single algorithm is not possible due to their inherent conflict, so some of the features must be given priority based on requirements and objectives of the system and the desired algorithm smust be designed with their orientation. In this article, a decentralized load balancing algorithm based on cellular automata and fuzzy logic has been presented which has capabilities needed for fair distribution of resources in Grid level.**

Each computing node in this algorithm has been modeled as a Cellular Automata's cell and has been provided with the help of fuzzy logic in which each node can be an expert system and have a decisive role which is the best choice for tasking in dynamic environment and uncertain data.

Each node is mapped to one of the VL, L, VN, and H, VH conditions based on information **exchange on certain time periods with its neighboring nodes and based on fuzzy logic and tries to estimate the status of the other nodes in subsequent periods to reduce communication overhead with the help of Fuzzy Logic and the decision making to send or receive task loads is done based on the status of each node. So an appropriate structure for the system can greatly improve the efficiency of the algorithm. Fuzzy control does not use search and optimization and makes decisions based on inputs which are effective parameters of the system and are mostly based on incomplete and nonspecific information.**

Keywords—computing Grid; load balancing; cellular automata; fuzzy logic

I. INTRODUCTION

The need for high computational power and organizational limitations have created a new type of shared computing environment, which is called computing grid. Computing Grid is a computing infrastructure that makes effective access to high performance with computing resources possible. End users and applications see this environment as a large virtual computing system. Systems that are connected by Grid may be distributed globally and be running on different hardware platforms and operating systems and belong to different organizations. In a short definition, Grid can be considered as a system for distributed resource sharing in a large scale and indeed without borders. Requests should be divided evenly among the available resources in order to globally enhance the global throughput of computing grid. Management of resources is one of the major issues in this environment. Resource management is a major and infrastructure Grid component of environment. The overall objective of resource management is effective timing to run programs that need to use resources in Grid environment. In a general definition. The purpose of load balancing algorithms is uniform distribution of the load on resources and maximizing resource efficiency as well as reducing the overall running time which means the difference between the most and least productive resources should be minimal. The load balancing problem for Grid environment in which equitable distribution of resources is one of the most important issues is also considered as a basic necessity. The desirable characteristics of a load balancing solution include: Comparability, versatility, stability, clarity of vision of program's user, capability of fault tolerance and minimal overhead costs imposed on the system. The load balancing methods are generally divided into centralized and decentralized, static and non-static, cyclic or non-cyclic, and has a threshold and no threshold. Cellular automata answers this question that How complex systems can be studied? There is the ability to predict the next state of cells in this system based on the status of each cell and its adjacent cells which can help in proper distribution of load among nodes. Given that the distribution of load needs awareness of mentioned conditions and considers the functionality of each resource in a computing grid and cellular automata has this feature which means it can predict current and future situation of each resource, the load distribution is done in a balanced way based on the needs and abilities and capabilities of each of these resources [14-19].

This article tries to execute distribution of load in Grid resources by evaluating the effectiveness of cellular automata and fuzzy logic which have capabilities required in the fair distribution of load and grade level decision-making. The load balancing algorithms has been provided in this article based on Cellular automata- and use of fuzzy rules. The remainder of the paper is organized as follows: In section 2, definition of concepts such as Grid, load balancing, cellular automata and fuzzy logic. Section 3 describes our proposed algorithm in detail. Section 4 discusses our simulation and results of evaluation. Finally, section 5 concludes this paper.

II. DEFINITION OF CONCEPTS

Advances in areas technical constantly need to have faster computing but computer hardware manufacturers have reached fundamental limitations in the physical speed [1]. Electronics and hardware advances in technology alone cannot meet the demand for increased computing speed. Parallel processing is the emerging response to this problem in which different parts of a task are simultaneously tasking on several processors [2, 3].

Although writing code that is flexible enough to be split among several processors is generally more difficult for programmers but the tendency toward parallel processing hardware and software has increased [2]. Instead of limiting the time of implementation of a program running on a processor, parallel processing task load is divided among several processors and allow this issue to be solved through team work, thus parallel processing has become a viable alternative to the circuit and faster processors which can only reduce the time of initial cycle time of the single processor [3]. Reduced costs powerful computers along with advances in computer networking technologies have increased the tendency for the use of large-scale parallel systems and distributed computing systems. In fact, recent studies in the field of computing architecture has led to emergence of a new computing paradigm which is computing Grid [4]. A computing Grid creates a hardware and software infrastructure which is: Reliable, consistent, pervasive, and has inexpensive access to high performance computing [5]. This technology is a type of distributed system that supports the sharing and coordinated use of resources, independently from physical type and their location in virtual dynamic organizations which is the same shared goal. Nowadays, a variety of Grid systems are manufactured with various definitions and facilities which have different objectives. Thus, providing a single definition that covers all aspects of grid computing technology in not easy nor true. Various experts have provided different definitions according to different pursued goals with different views towards this technology and its various applications.

Ian Foster who was the main inventor of Grid and founder of Globus defines Grid as follows [6]:

"Grid technology is seeking to create the possibility of large-scale and controlled resource sharing which is flexible and is after creating protocols, services and software packages".

Gird is defined as follows in IBM Company which is among pioneers of Grid:

"Grid is a set of distributed computing resources in a local area network or a wide area network which seems like a computer and virtual computing system for end-user or applications. Its main goal is creating dynamic virtual organizations through sharing resources using coordinated and safe methods among users, universities and organizations".

A computing grid is a grid computing infrastructure which provides access to advanced computing resources which features such as being: High-End Computational Resources, Dependable, Consistent, Pervasive and having Coordinated Resource Sharing and problem solving in dynamic virtual organizations is multi-organ.

Generally, Grid is a distributed system which contains following items [8]:

- Resources (software and hardware) are heterogeneous
- Resources are coordinated but are not under a centralized management
- The use of all-purpose standard protocols and interfaces
- Grid may have Multiple administrative domains or in other words be made of several Virtual Organizations (VO)
- Ensuring the quality of the services provided

Resource management is one of the important issues in such environment. Resource management is among major components and infrastructure of Grid environment. The overall objective of resource management is effective timing of applications which need to use available resources in Grid environment for running. In a general definition. The purpose of load balancing algorithms is equal distribution of load on resources and maximizing their performance as well as reducing the overall execution time [9]. In another definition, the load balancing algorithm is an algorithm which ultimately allows all nodes to task at once [10-11]. The issue of load balancing for Grid environment has fair distribution of load on resources as a basic necessity. The desirable characteristics of a load balancing solution include: desirable characteristics of a load balancing solution include: Scalability, Adaptability, Stability, Application Transparency, Fault Tolerant and minimum overhead imposed on the system. The mentioned specifications are greatly interdependent. For example, delays such as Computation Delay and Communication Delay have abnormal effects on the stability and thus comparability of the algorithm. Due to the many parameters involved in the problem of load balancing as well as contradictory of some mentioned features, meeting all the features in the form of a single algorithm is practically difficult or even impossible. Most of the existing methods try to satisfy one or more of the above objectives [12-14].

For better efficiency and more use of dynamic algorithms and considering that the main focus of this article is on the same set of algorithms, in general, the process of dynamic load balancing algorithms has four main routines:

A. Load Measuring routine

B. Information Exchange routine

C. Initiation routine

D. The final load balancing operation

Load measuring routine is expression of CPU load in a way that heavier load on processors will increase it and its reduction will reduce it. Since the routine is repeatedly and with great frequency in use (run) of the load balancing algorithms, the Calculation of obtaining should be as simple and as efficient as possible [17]. Information Exchange routine determine the method of collecting necessary task load for load balancing

decisions. Initiation routine decides about the time of starting load balancing. This decision-making is along with determining the ratio of efficiency to imposing overhead (which means load balancing must be effective). Load balancing methods attempt to achieve goals such as minimizing the average response time for processing or maximizing resource efficiency by running processes on distributed resources. This this goal may initially be a demand or take place after the start of its implementation. Of course, in any case, a good and efficient algorithm must consider the cost of route as well [18]. Cellular Automata (CA) is an answer to this question that how to study complex systems. Cellular automata can be a complex system in itself and yet provide appropriate methods to study complex systems like these - Complex systems – [19-20].

III. The Proposed Algorithm of Fuzzy Load Distribution using Cellular Automata (Fuzzy Load Balancing Cellular Automata)

The main idea of this project is using a cell of cellular automata to show a computational node in which the status of cell shows the status of that node. A load balancing solution can be created just using local load balancing in this method. The method of load distribution is in form of a wave motion.

All parameters that each processor considers during the proposed load balancing algorithm are described below:

M: Number of heterogeneous computing nodes in the system (P_1, P_2, ..., P_M)

x: Number of job executed in the system (j_1, j_2, ..., j_x).

T_s: Iinformation exchange time.

T_e: The estimated time period.

N_i: Buddy set of node P_i.

$S_i(T_n)$: State of node i at time T_n.

ɱ: Number of migration of a job.

$Q_i(t)$: The number of jobs waiting in the execution queue at the node Pi at time t.

w_i: Processing power at P_i.

$z(j_x)$: Size of job(x).

$TET_{i,t}$: Total waiting time for execution of waiting job at p_i queue.

$RET_{i,t}$: The remaining execution time of the job being processed at the P_i.

$LD_{i,t}$: Load of Pi at time t, comes from (1).

$$LD_{i,t} = TET_{i,t} + RET_{i,t} \quad (1)$$

$NLD_{i,t}$: Normalized average load in the buddy set of node P_i at time t.

BW_{ij}: Bandwidth communication between processors i and j

$ArrTime(j_x)$: Arrival time of job J_x.

$endTime(j_x)$: End time of job j_x.

$ETC(j_x, p_i)$: Estimated execution time of j_x at p_i, comes from (2).

$$\text{ETC}(j_x, p_i) = \frac{\text{ETC}(j_x, p_{std})}{w_i} \quad (2)$$

$T_{com}(j_x, p_i, p_j, t)$: The time required for transfer job j_x from p_i to p_j at time t.

$EFC(j_x, , p_i, p_j, t)$: Estimation of finish time of job j_x when transfer from p_i to p_j at time t.

if $\text{T}_{\text{com}}(\text{j}_\text{x}, \text{s}_\text{i}, \text{s}_\text{j}, \text{t}) \geq \text{LD}_{\text{j,t}}$

$$\text{EFC}(\text{j}_\text{x}, \text{s}_\text{i}, \text{s}_\text{j}, \text{t}) = \text{T}_{\text{com}}(\text{j}_\text{x}, \text{s}_\text{i}, \text{s}_\text{j}, \text{t}) + \text{ETC}(\text{j}_\text{x}, \text{s}_\text{j})$$

ELSE

$$\text{EFC}(\text{j}_\text{x}, \text{s}_\text{i}, \text{s}_\text{j}, \text{t}) = \text{LD}_{\text{j,t}} + \text{ETC}(\text{j}_\text{x}, \text{s}_\text{j}) \quad (3)$$

$B_x(p_i, p_j)$: Benefit of execution of the job j_x at p_j compared to execution at pi.

$$B_x = EFC(j_x, p_i, p_i, t) - EFC(j_x, p_i, p_j, t) \quad (4)$$

The general routine of the proposed Load balancing algorithm is in a way that when a new task enters the computational node, that node will decide based on cell's conditions that it should carry out this task itself or migrate it to another node.

This algorithm consists of several main routines:

- Determining the status of nodes
- Making decision to migrate the task
- Selecting the best node to carry out the task

A. Determining the status of nodes

The overall basis for all decisions is the status of that node. In fact, the essential criterion in deciding to send a task is the status of that node and the main criterion for selecting a node to perform the task is also the status of that node. Thus, determining the status of each node is crucial in load balance in the whole system.

In order to determine the status of each node and its neighbors for each running and migration, the information are needed to determine the status of nodes. There will be a large overhead in the system if a series of messages are exchanged between nodes to exchange their status. Thus, regular intervals are used to determine the status of nodes which are called information exchange periods. T_s which is greater than the period of time for running and migration of tasks is performed between nodes which estimates the status of nodes between these time periods. T_e tries to reduce communication overhead and have a more accurate decisions (Fig. 1).

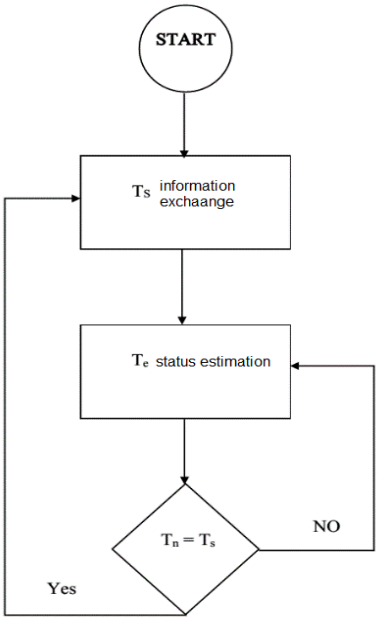

Fig. 1. Determining the status of nodes in T_n

Determining the status of the node is done in two methods:

- Information Exchange (T_s)
- State estimation (T_e)

1) Determining the status of nodes through the information exchange

Determining the status of nodes through the information exchange contains three parts of sending load information, calculating the average load and determining the status of nodes using Fuzzy Logic (Fig. 2).

a) Sending load information

All of the nodes send the information related to their load to other node in their neighboring collection in regular time periods of T_s which are called information exchange periods and receive their information. Then each node records the reviewed information from each neighboring node in neighbor table.

b) Calculate the average load

Each node calculates its average load and the load related to neighboring collection using (5) and considers the obtained index as normalized load average.

$$NLD_{i,t} = \frac{\sum_{j \in N_i} LD_{j,t}}{N \times \sum_{i \in N_i} w_i} \quad (5)$$

c) Determining the status of nodes using Fuzzy Logic

The normalized load average is considered as the point of balance and map the status of each node to one of Very light, Light, Normal, Heavy, Very heavy forms using fuzzy logic. This step is called mapping input values to fuzzy mode and the status of each node will be recorded in neighboring table.

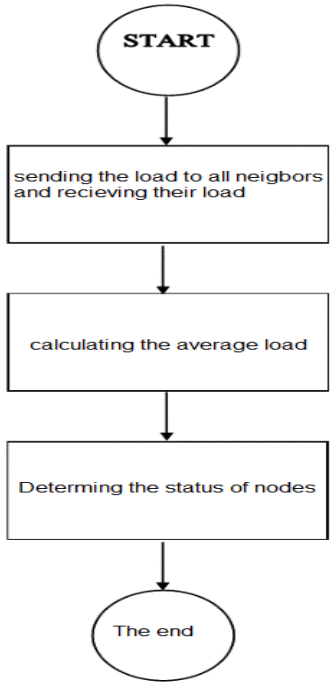

Fig. 2. Information exchange

2) Determining the status of node using estimation

If this data exchange is done in intervals with short distance, there will be a high communication overhead imposed on the system. Thus, these intervals must be increased and estimated T_e status in information exchange periods using fuzzy rules.

In order to discover these laws, the algorithm runs without estimation periods and records the data on each node. The related fuzzy rules are extracted by using Matlab software. In order to reduce communication overhead caused by data exchange, exchange periods will be increased and the status of each node using obtained rules in order to reduce errors in decision making will be estimated.

B. The decision to send task

When j_x task enters node and that node is in one of VL, L, VN states, it will be queued for processing and will be waiting to run according to respective priority and the higher rate of migration (ɱ) leads to increased running priority.

But when j_x enters node and that node is in one of H, VH states, if it is migrated from another node, it will not be accepted and rejected but if the job belongs to that node, then if the node is in VH status then it calculates the amount of its own load and the normal load and if it has a neighbor with VL status, it selects ½ of its additional load and if it has a neighbor with L status, it selects ¼ of its additional load for migration and if it has neighbors with both statuses, it sends with the same ratio to both of them. If it is in H status, it will calculate the difference between its own load and the normal load and if it has a neighbor with VL status, it will select ¼ of its additional load for migration and selects 1/8, if it has a neighbor with L status.

C. Selecting the most appropriate node to perform the task

Initially nodes with VL, L are marked in neighboring nodes with weight of 1 for L node and 2 for VL node. Then the implementation cost of performing the j_x task is estimated on specified nodes and based on $B_x(p_i, p_j)$ profit which is the difference between running cost in node p_j compared to p_i node, if this profit I positive and bigger than the threshold, a weight is given to each one of them. In this way that the higher running profit leads to higher score and the value of this profit close to the threshold will lead to having score near one. Finally, with regard to weight of node's status and the weight of running's profit in P_j node, the decision is made to send this node to P_j. the higher weight will lead to higher possibility of sending the task to that node (Fig. 3).

It should be noted that the cost of running a task may be equal in source node and the destination node which means running the task on that same node or migration it to another node may result in similar end time which has the running profit of zero and even have a little earlier end time but that task is not allowed to be migrated to that node because it imposes communication overhead on the system and the bandwidth of communication between processors is engaged even for small time.

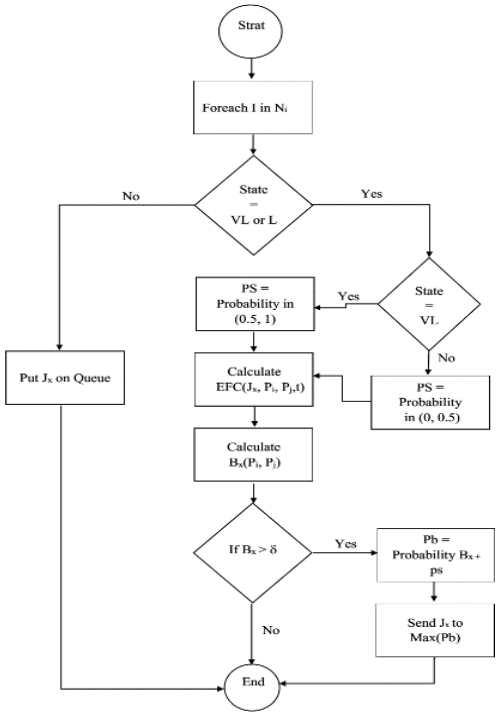

Fig. 3. Selecting the most appropriate node to perform the task

IV. Simulation and Results

Simulation is an imitation of the real world. Simulators enable program designers, instrument developers and grid developers who need to test programs, tools and services available to ensure the proper function of their program before final production and using them in real world.

The proposed FLBCA algorithm has been simulated using C# programming language in Visual Studio 2010 environment and its algorithm has been compared with MELISA algorithm.

A. Efficiency criteria

Following two criteria have been used in order to evaluate the performance of the proposed algorithm [21-23]:

1) Average response time (ART):
The average time from when the task enters the Grid until it successfully comes out of the Grid environment.

2) The average performance of computing nodes (CPU):
the ratio of working time to the total time of a system:

$$U_i = \frac{Busy_i}{Busy_i + Idle_i} \quad (6)$$

$$APU = \frac{\sum_{i=1}^{M} U_i}{M} \quad (7)$$

B. Simulation Model

This simulation is formed by 30 heterogeneous computing grids which their processing power follows random distribution in range of [1, 10] and the relation between two nodes has been formed by heterogeneous communication network in a way that their communication bandwidth is variable from 1 Mbps to 10 Mbps.

10,000 independent tasks have been used in this simulation in a way that running time of each task has been generated randomly in the range of [1, 100]. These tasks enter the system based on Poisson distribution with rate of [1, 4] and the volume of each task follows normal distribution with mean of 5 MB and standard deviation of 1 MB.

The time for information exchange (T_s) is assumed to be 20 units and the estimation time of status is considered to be 5 units

C. Simulation results

This algorithm has been evaluated in terms of performance criteria and under the effect of factors such as the number of tasks, time and period of service transition and estimation interval.

1) The effect of the works entered in the homogeneous environment

In a homogeneous environment where processing power of each CPU is 1 and the communication bandwidth between any two nodes is constant and equal to 10 Mbps, the number of tasks has been added from 0,000 to 50,000 in order to measure these factors. In these conditions where the number of tasks is 10000. The average response time is about the same among all three algorithms but with increased tasks, the efficiency of ELISA algorithm is better than MELISA algorithm and the proposed algorithm and the efficiency of FLBCA algorithm is slightly better than MELISA algorithm (Fig. 4). The total run time is about the same in all three algorithms (Fig. 5).

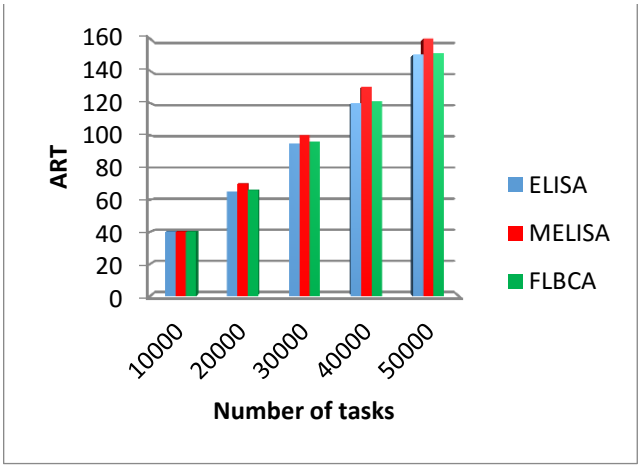

Fig. 4. The average response time in case of homogeneous

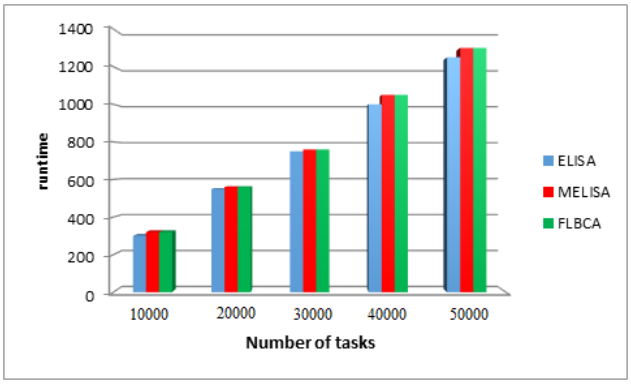

Fig. 5. The total runtime in homogeneous environment

2) The effect of tasks entered into the heterogeneous environment

Since this algorithm has been designed for heterogeneous environments, it is compared in heterogeneous environment with different processing power in the range of [1, 10] and various communication bandwidths between any two nodes in the range of 1 Mbps to 10 Mbps with ELISA and MELISA algorithms.

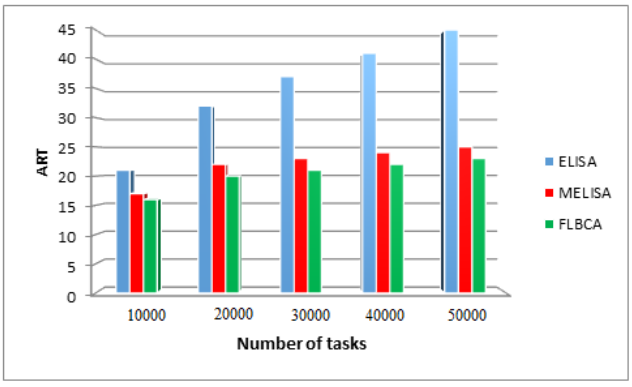

Fig. 6. Comparing the average response time in heterogeneous environment

In order to measure the effectiveness of the work, the number of tasks have been increased from 50,000 to 10,000. The average response time is much better in FLBCA and MELISA algorithms than the ELISA algorithm. The average response time is initially about the same in FLBCA and MELISA algorithms but is gets better with increasing number of tasks in FLBCA algorithm and shows better and faster decisions in fuzzy logic (Fig. 6).

The total runtime is similar among all three algorithms. The runtime is a function of the rate of entering data into the system and the runtime is about the same due to using same data (Fig. 7).

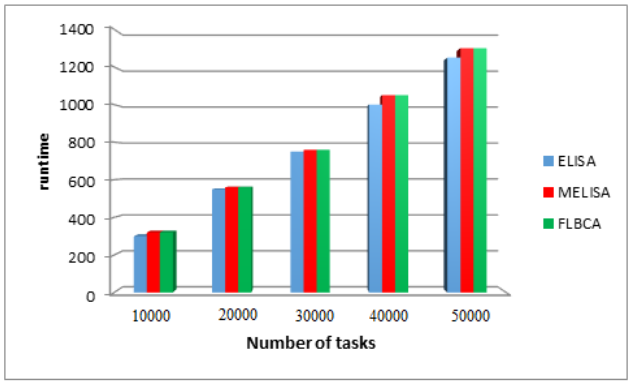

Fig. 7. Comparing the total runtime in heterogeneous environment

3) The effect of job size

This section tries to evaluate the effect of changing tasks volume from MB to 50 MB on the average response time in the proposed algorithm. The number of migration reduces and the average response time increases with increased runtime (Fig. 8).

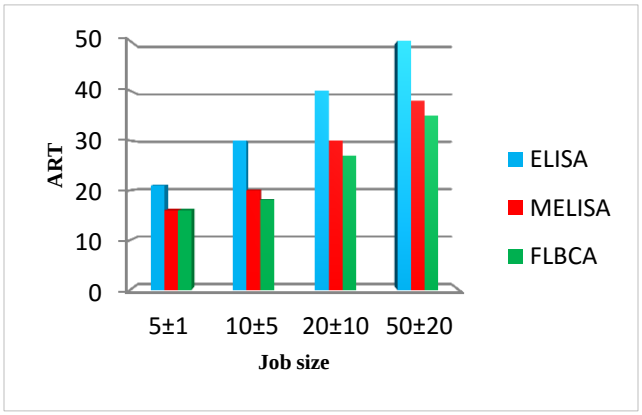

Fig. 8. Average response time for different job sizes

4) The effect of running tasks

This section tries to evaluate the effect of changing the average runtime of tasks from 10 to 150 units on the efficiency of the algorithm. The average response time increases with large rate by increasing the runtime (Fig. 9).

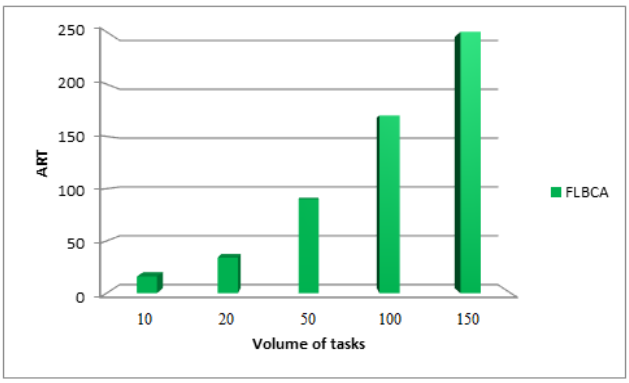

Fig. 9. The average response time for various run times

5) The effect of information exchange time

This algorithm was tested with different times in range of 2 units to 40 units in order to find a time for information exchange which is suitable in perspective of efficiency criteria (Fig. 10).

The accuracy of information reduces with increasing time of the information exchange. The distribute the load between nodes is done with less precision and average response time increases for this purpose

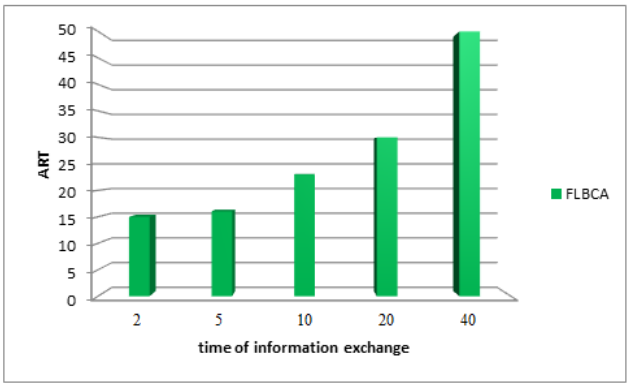

Fig. 10. The average response time for different times of exchanging information

6) The effect of state estimation time

In order to access a proper estimation time from the point of efficiency criteria, the exchange time must be considered to be 20 units by default. Then by changing the estimation time in different intervals from 2 units to 10 units, the most effective time will be found (Fig. 11).

The average response time increases by reducing estimation time due to increased computational overhead and the most optimal time is reached at times of 4 and 5 and the average response time increases again by increasing this time due to reduced accuracy of data.

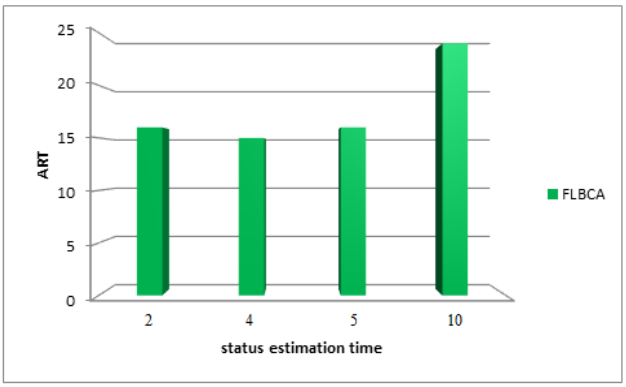

Fig. 11. Comparing the average response time for different times

V. CONCLUSION

A load balancing model has been provided in this research for grid computing environment which is called FLBCA. This algorithm tries to meet needs and characteristic required for a load balancing algorithm such as stability, versatility, transparency from the user's standpoint and minimize the communication overhead as much as possible. Since the fuzzy logic system has the ability to deal with imprecision and uncertainty. Algorithm based on fuzzy logic has been proposed for dynamic load balancing in computing grid. The main purpose of using this algorithm is increasing efficiency and productivity of Grid system which will lead to reducing organization's costs and increasing productivity and saving energy and since using energy efficiently saves it which is the top priority in our life today and helps to protect the environment and nature.

Among the tasks carried out in this paper and with respect to the fact that the issue of resource management is still discussed about in Grid as well as the importance of load balancing in it and the use of cellular automata which can be a good platform for the design of load balancing algorithms, it seems hierarchical cellular automata is an Appropriate structure for design of these types of algorithms in future which can be provided a better view of the whole condition of the Grid System. In addition to this, the use of fuzzy logic which leads to increased accuracy of decision-making in uncertain environments can be used in to improve the efficiency of parallel algorithms. The combination of fuzzy logic and cellular automata can be a good technique for a lot of parallel algorithms.

REFERENCES

[1] M. Fathi, F. Mehryari., "the principles and concepts of grid computing technology and its applications in various fields", Noorpardazan, .10, 2009 - Tehran, pp.

[2] M. Amini Salehi, H. Deldari, load balancing in the grid resources using agent-based resource management, Master's thesis, Department of Computer Engineering, Ferdowsi University of Mashhad, 2005.

[3] S. Ghanbari, balance and self-organization in the grid computing using learning automata, Master's thesis, Department of Computer Engineering, Amirkabir University of Technology, 2004.

[4] R. Tlili, Y. Slimani, A Hierarchical Dynamic Load Balancing Strategy for Distributed Data Mining, International Journal of Advanced Science and Technology, Vol. 39, February. 2012

[5] S. Adabi, reducing power consumption in wireless sensor networks based on cellular automata, Master's thesis, Department of Computer Engineering, Islamic Azad University of Science and Research Branch, 2010.

[6] L. Anand, D. Ghose, V. Mani, ELISA: An Estimated Load Information Scheduling Algorithm for Distributed Computing Systems, An International computers & mathematics with applications, 1999.

[7] L. Rostami, A. Rahmani, An adaptive Load Balancing Algorithm with use of cellular Automata for Computational Grid Systems, Euro-Par 2011 Parallel Processing, Lecture Notes in Computer Science Volume 6852, 2011, pp 419-430.

[8] A. Karimi, F. Zarafshan, A. Jantan, Anew Fuzzy Approach for Dynamic Load Balancing Algorithm, International Journal of Computer Science and Information Security, Vol. 6, No. 1, 2009.

[9] M. Marinov, Intuitinistic Fuzzy Load balancing in cloud computing, 8th Int. Workshop on IFSs, Ocy 2012.

[10] S. Mousavi Nejad, S. Mortazavi, B. Vosoughi Vahdat, Design and set optimal control and intelligent load balancing based on fuzzy logic in distributed systems, first regional conference on new approaches in computer engineering, 2011.

[11] I. Foster, and C. Kesselman, "The Grid: Blueprint for a New Computing Infrastructure" Morgan Kaufmann and Elsevier, Second Edition, USA, ISBN: 1-55860-933-4, 2004.

[12] I. FOSTER, C. KESSELMAN, M. NICK J, S. TUECKE, "Grid services for distributed system integration,« vol. 35, 6, 2002.

[13] C. J., K. E.,. L. M. Anderson D P., "SETI @ home: an experiment in public-resource computing," vol. 45 (11).

[14] S. Graupner, J. Pruyne, S. Singhal, "Making the Utility Data Center a Power Station for the Enterprise Grid," 2003.

[15] J. Liu, X. Jin, and Y. Wang, Agent-Based Load Balancing on Homogeneous Minigrids: Macroscopic Modeling and Characterization, IEEE TRANS.IONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 16 (7) 2005.

[16] L.SUN CHEUNG, A fuzzy approach to load balancing in a distributed object computing network, In Proc. Of 6th Int IEEE Conf. HPDC, 2000.

[17] T. L. Casavants and J. G. Kuhl, A taxonomy of scheduling in general-purpose distributed computing systems, IEEE Trans. Software Eng., Vol. SE-14 (2), pp. 141-154, 1988.

[18] H. Kameda, J. Li, C. Kim, and Y. Zhang, Optimal Load Balancing in Distributed Computer Systems. London, U.K .: Springer-Verlag, 1997.

[19] Z. Zeng and B. Veeravalli, Rate-Based and Queue-Based Dynamic Load Balancing Algorithms in Distributed Systems, 10th Int. Conference on Parallel and Distributed Systems, IEEE 2000.

[20] Abubakar, Haroon Rashid and Usman Aftab, Evaluation of Load Balancing Strategies, National Conference on Emerging Technologies 2004.

[21] J.Cao, Daniel P. Spooner, Agent-Based Grid Load Balancing Using Performance-Driven Task Scheduling, In Proc. of 17th IEEE Int. Parallel & Distributed Processing Symposium (IPDPS 2003), Nice, France, April 2003.

[22] A. Shaout and P. McAuliffe, Job scheduling using fuzzy load balancing in distributed system, in Proc. of 6st conf ICPAD, 1998.

[23] J. Liu ,, X. Jin, and Y. Wang, Agent-Based Load Balancing on Homogeneous Minigrids: Macroscopic Modeling and Characterization, IEEE TRANS.IONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 16 (7), 2005.