A Novel Agent Based Load Balancing Model for Maximizing Resource Utilization in Grid Computing

Ali Wided and Kazar Okba
Department of Computer Science, Mohamed Khider University, Biskra, Algeria
E-mail: aliwided1984@gmail.com

Keywords: grid computing, load balancing, multi agent system, performance metrics, agent based load balancing

Received: July 15, 2019

Grid is the collection of geographically distributed computing resources. For effective management of these resources, the manager must maximize its utilization, which can be achieved by efficient load balancing algorithm, The objective of load balancing algorithms is to assign the load on resources to optimize resource use while reducing total jobs execution time. The proposed agent based load balancing model aims to take advantage of the agent characteristics to generate an autonomous system. It also addresses similar systems drawbacks such as instability, scalability or adaptability. The performance of the proposed algorithms were tested in Alea 2 simulator by using different parameters such as response time, resources utilization and overall queue time. The performance evaluation suggests that the proposed algorithm can enhance the overall performance of grid computing.

Povzetek: Predstavljena in s simulatorjem analizirana je agentna metoda razporejanja obremenitev v omrežju.

1 Introduction

Due to the emergence of grid computing on the Internet, a hybrid load balancing algorithm, which takes into account various factors such as grid architecture, computer heterogeneity, communication delays, network bandwidth, resource availability, unpredictability and job characteristics, is now required.

For grids, scalability and adaptability are two major issues. As for the centralized resource scheduling problem, the limitation of scalability and computational performance is inevitable. Moreover, due to resource heterogeneity, resource variations, application diversity and grid environments are dynamic. Therefore, adaptive and robust scheduling techniques are preferred [1][2].

Multi-agent systems offer promising features for resource managers. The reactivity, proactivity, scalability, cooperation, robustness, flexibility and autonomy that characterize agents can help in the complex task of managing resources in dynamic and changing environments.

This paper presents a new Agent Based Load Balancing Algorithm, called ABLBA. A hierarchical architecture with coordination is designed to ensure scalability and efficiency. In addition, a multi-agent approach is applied to improve the adaptability. The proposed algorithm aims to reduce the average response time, as much as possible, of jobs submitted to the Grid, and to maximize throughput and resource utilization.

2 Related works

Authors in [3] proposed a multi-agent load balancing model by analyzing the load of compute nodes and the subsequent migration of virtual machines from overloaded nodes to underloaded nodes. The proposed system involves multiple nodes that interact to implement MapReduce jobs. The multi-agent system consists of a group of agents: node sensor agent, simulation model sensor agent, analysis agent, migration agent and distribution agent. Analysis and distribution agents are defined as reasoning agents.

In [4], a decentralized computing algorithm was proposed to assign and schedule jobs on a distributed grid. Using the properties of multi-agent systems, the proposed distributed resource allocation protocol (dRAP) is described as follows:

An agent in the system is simply a node. Each agent has a vector including the number of CPUs in its cluster and the residual time to complete the execution of its current process. Each agent is assured to be in exactly 1 out of 4 cases during the simulation.

A main feature of this algorithm is that nodes ask their neighbors to form clusters. This reduces waiting time and communication costs. One optimization to consider would be to delay the disconnection of the cluster in state 4, which would guide learning or memory in the system where the planner would be able to remember the requirements of the past process. The problem with this algorithm is its decentralized nature, it is neither a centralized control nor a precise synchronization on nodes (agents).

The study in [5] presented the development of an agent-based model for managing network resources with defined operations so that the user can perform jobs efficiently and effectively and thus significantly improve management by a gLite Grid middleware. The proposed solution provides a platform based on a collection of agents in a virtual organization. The key aspects of this proposal architecture are: resource tracking, load balancing and agent hierarchy.

In [6] the authors proposed a new load balancing structure based on the moving agent and a technique for optimizing ant colonies. In the proposed structure, a dispatcher agent is involved in distributing the tasks received to the worker agents according to the right decisions to minimize the overall execution time (makespan). The proposed framework is constructed using three layers which are the producer of user tasks, the scheduling load balancing layer and the workers' layer. This study should be complemented by comparing their results with other methods, minimizing task movements and resulting in additional costs in the migration process.

Authors in [7] presented the design and implementation of a priority scheduling and fuzzy load balancing model in a computing grid. In this grid template, the user sends his jobs to the grid agent, after the grid scheduler uses the priority-based scheduling algorithm to schedule jobs from the grid agent to the available resource. Load balancing is done using the fuzzy logic technique Propose, in which a set of fuzzy rules are produced using the resource and the work parameter. As fuzzy control rules are collected using linguistic variables, perceptual knowledge and inspection are easily integrated into the control mechanism.

3 Proposed agent based load balancing model

A grid computing was modelled as a set of clusters. Each cluster was composed of nodes and belonged to a LAN local domain (Local Area Network). Every cluster was connected to the WAN global network (World Area Network) by a Switch [8].

The proposed Agent Based load balancing model was based on mapping the Grid architecture into a tree structure. This tree was built by aggreGAtion as follows: first, for each cluster, a two level subtree was created. The leaves of this sub-tree correspond to the cluster nodes, and its root, called cluster manager, represents a virtual node associated with the cluster. Secondly, sub-trees corresponding to all clusters were collected to generate a three level sub-tree whose root is a virtual node designated as a Grid manager. The concluding tree is referred to as C/N, where C is the number of clusters that constitute the Grid and N the number of worker nodes [8].

This study aims to develop a hierarchical load balancing model based on a multi-agent system. There are two key challenges for Grid computing: heterogeneity and scalability. The authors propose a three-layer architecture to address the scalability issue. Connecting or disconnecting resources (worker nodes or clusters) correspond to simple operations in a tree (adding or removing leaves or sub-trees). The proposed agent based load balancing model aims to take advantage of the agent's characteristics to create an autonomous system. It also addresses similar disadvantages such as instability, scalability, adaptability, etc., and other specific issues related to grid computing.

3.1 Model characteristic

The proposed model is characterized as hierarchical; this characteristic facilitates the circulation of information through the tree and defines the flow of messages in the proposed strategy.

Three types of load information movements can be identified:

- Ascending movement: this movement relates to the load information movement, to get current load state. from Level 2 (node Agents) towards Level 1 (Cluster Agents). or from Level 1(Cluster Agents) towards Level 0 (grid Agents). With this movement, the cluster manager can have a global view of the cluster load or the grid manager can have a glob view of the grid load.
- Horizontal movement: it concerns the useful parameters for the execution of load balancing operations. This movement relates to task assignment intra-cluster in Level 2.
- Descending movement: this movement allows to take decisions for task assignment or jobs migration, the decisions taken by cluster Agents at levels 1 to the Migration Agents at same level. And from Migration Agents at level 1 to Node Agents at level 2, also from Grid Agent at level 0 to Cluster Agents in level 1.

The proposed model:

- supports the scalability and heterogeneity of grids: insertion or elimination entities (processing elements, nodes or clusters) are very simple operations in the proposed model (insertion or elimination nodes, subtrees);
- is totally independent of any physical structure of a grid: the conversion of a grid into a tree is a unique conversion. Each grid corresponds to one and only one tree;
- is based on the exchange of information between Nodes and clusters through their respective agents.

Level 0: At this level, Grid Agent is located, the Grid users send their jobs to the Grid Agent, for which it is responsible:

- receiving jobs from Grid users
- sending jobs for Node Agents
- all Cluster Agents are started by Grid Agent
- initiating a global load balancing process

Level 1: At this level, Cluster Agent is associated with a physical grid cluster; this Agent is responsible for:

- the maintenance of the load information relating to each of its Node Agents.
- estimating the load of the associated cluster and sending this information to Grid Agent.
- the decision to start local load balancing
- sending load balancing decisions to Migration Agent

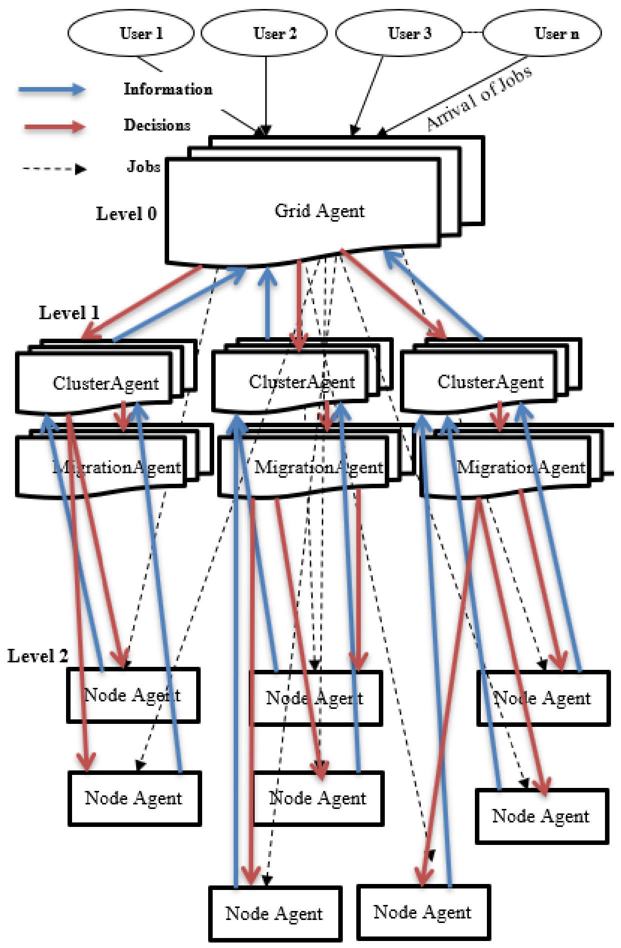

Figure 1: Agent based load balancing model in grid.

- Migration Agent is started by its associated cluster Agent
- all Node Agents are started by their corresponding Cluster Agent

Migration Agent is also present at this level, whose role is to:

- start the migration process
- send the migration decisions to the Node Agents.
- wait for an acknowledgement from receiver node and ensure that the migrated jobs are received and successfully resumed at the destination node

Level 2: At this level, Node Agent is present; it is necessary to have one Node Agent on each node; every Node Agent at this level is responsible for:

- maintaining its load information
- sending this information to its associated
- Cluster Agent
- working in cooperation with the Migration
- Agent to execute the migration process
- collect information about the jobs (number of jobs queued at node, arrival time, waiting time, submission time, start time, processing time and finish time of each job on the local node)
- remove the terminated, leaving or migrated jobs from queue of jobs
- calculate the total load of node
- receive jobs sent by Grid Agent

3.2 Proposed algorithms

According to the proposed model, two levels of load balancing are considered: Intra-cluster Agent based load balancing algorithm and Inter-Clusters Agent based load balancing algorithm.

There are certain specific events that change the load configuration in Grid computing and can be classified as follows:

- Any new job is arrived
- Accomplishment of execution of any job
- Any new node is arrived
- Any existing node is removed
- Failure of Machine at any node
- The node become overloaded

When any of these events happen, the local load value is changed. Table 1 summarizes the notations used in the proposed algorithms.

Parameter	Description
N	Node
$\mathbf{Load_N}$	Load of Node
Qlength	Queue length
CPU-U	CPU utilization of Node
Mem	**Memory utilization of node**
$\mathbf{TH_H}$	The higher threshold
$\mathbf{TH_L}$	The lower threshold
OLD-list	Overloaded List
ULD-list	Underloaded List
BLD-list	Balanced List
$\mathbf{Load_{avg}}$	Average Load
$\mathbf{NBR_N}$	Number of Nodes of cluster
C	Cluster

Table 1: Notations used in the proposed algorithms.

3.2.1 Intra-cluster agent based load balancing algorithm

Depending on its current load, each Cluster Agent decides to start a Job Migration operation. In this case, the Cluster Agent tries, in priority, to balance its load among its nodes.

Load estimation

The node load at a given time was simply described by the CPU queue length. It indicates the number of processes awaiting execution. The proposed algorithm considers CPU-U (CPU Utilization), Q length (Queue length) and Mem (memory utilization) as load information parameters to measure the load of a node.

These parameters are calculated as follows:

Load (CPU-U)= (U1+U2+……+UT)/T, where: U1+U2+……+UT is the value of CPU-U in a previous one second interval.

Load (Qlength) = (Q1+Q2+…...+QT)/T, where: Q1,Q2,……...,QT is the value of Qlength in a previous one second interval.

Load(Mem)=(M1+M2+……...+MT)/T Where: M1,M2,……...,MT is the value of Mem in a previous one second interval. T is the number of time intervals.

The averaged information of CPU-U, Qlength and Mem are the load parameters used to describe the node load.

Algorithm 1. An algorithm for Node Agent

```
1:  T←5 seconds
2:  Waiting for jobs;
3:  Create jobs queue for related node;
4:  In each one second of T intervals do
5:  Calculate (CPU-U);
6:  Calculate (Qlength);
7:  Calculate (Mem);
8:  End do
9:  Load (CPU-U) = $(U_0+U_1+…..U_T)/T$;
10: Load (Qlength) = $(Q_0+Q_1+…..Q_T)/T$;
11: Load (Mem) = $(M_0+M_1+…..M_T)/T$;
12: Send load information for related Cluster Agent
13: Wait for load change // happening of any of
    defined events
14: If (events_happens ()=1 or events_happens ()=4)
    then // Termination or migration of job
15: Remove terminated or migrated job from the
    waiting queue
16: Subtract their load value from the total local load
    of node.
17: Send new load to its Cluster Agent associated;
18: End if
19: If (events_happens ()=2 or events_happens ()=3)
    then // new or incoming job
20: Add the newly created or incoming job for the
    waiting queue
21: Add their load value for the total local load of
    node
22: Send new load to its Cluster Agent associated;
23: End if
```

Function events_happens ()

```
output Type: integer
1: If (Job.state=Termination) then events_happens ()
   =1; End If
2: If (Job.state=Start) then events_happens () =2;
   End If
3:If (Job.state=Incoming Migrating) then
   events_happens ()=3; End If
4: If (Job.state = migrated) then events_happens
   ()=4; End If
5:If (Arrival of any new resource) then
   events_happens ()=5; End If
6: If (Cluster.state=saturated )then events_happens
   ()=8; End If
7:If (Cluster.state=unbalanced) then events_happens
   ()=9; End If
```

Location policy

In the next step, the nodes must be classified according to their load. Three states were used for classification: overloaded, underloaded and balanced. First, Cluster Agent must calculate two threshold values, which are calculated as follows:

- cluster Agent calculates load average of each parameter (CPU-U and Qlength) over all related nodes.
- $Load_{avg}$(Qlength)=($load_1$+$load_2$+….$load_{NBRN}$)/NBR N, where $Load_{avg}$(Qlength) is the average load of Qlength over all related nodes.
- $load_1$,$load_2$,….$load_n$ are the current Qlength of each node calculated by Node Agent.
- $Load_{avg}$ (CPU-U) =($load_1$+$load_2$+….$load_{NBRN}$)/ NBRN, where $Load_{avg}$ (CPU-U) is the average load of CPU-U over all related nodes.
- $load_1$,$load_2$,….$load_{NBRN}$ are the current load of CPU-U of each node calculated by Node Agent.

Calculation of threshold values

The higher and lower threshold values of Qlength and CPU-U of parameters are calculated by multiplying the average load of (Qlength or CPU-U) and a constant value.

- THH(Qlength) =H*Loadavg(Qlength)
- THL(Qlength) =L* Loadavg(Qlength)
- THH(CPU-U) =H*Loadavg(CPU-U)
- THL(CPU-U) =L* Loadavg(CPU-U)

where, TH_H is the high threshold and TH_L is the low threshold. H and L are constants. The next step is to divide the nodes for balanced, overloaded and underloaded nodes using the threshold values as follows:

- Overloaded: the node will be added for overloaded list if queue length is high, or CPU utilization is high, or memory usage is greater than 85%, then the node is classified as overloaded node.
- Underloaded: the node will be added for underloaded list if queue length is low, or CPU utilization is low.
- Balanced: the node is not into the overloaded list or the underloaded list. The node is in a balanced load state. They are considered to be more loaded than the low state and less loaded than the high state.

Algorithm 2. An algorithm for Cluster Agent

```
1:  Startup its related Node Agent
2:  Startup its related Migration Agent
3:  Receive load information($Load_N$(Qlength),
    $Load_N$(CPU-U)) from its related nodes.
4:  Calculate and send its load information for
    Grid Agent.
5:  somme ←0; somme1←0;
6:  For every Node N of cluster C do
7:  Somme← Somme+ LoadN(Qlength);
8:  Somme1← Somme1+ LoadN(CPU-U);
9:  End For
10: $Load_{avg}$(Qlength)= somme1/NBR-N;
11: $Load_{avg}$(CPU-U)= somme/NBR-N;
12: $TH_H$(Qlength)= $Load_{avg}$(Qlength)*H;
13: $TH_L$(Qlength)= $Load_{avg}$(Qlength)*L;
14: $TH_H$(CPU-U)= $Load_{avg}$(CPU-U)*H;
15: $TH_L$(CPU-U)= $Load_{avg}$(CPU-U)*L;
16: Partition Nodes into overloaded list OLD-
```

```
    list, underloaded list ULD-list and
    balanced list BLD-list
17: OLD-list←∅; ULD-list←∅; BLD-list←∅;
18: For every Node N of cluster C do
19: If (($Load_N$(Qlength)>$TH_H$(Qlength)) or
    ($Load_N$(CPU-U)>$TH_H$(CPU-U))or
    (Load(Mem)>85%)) then
20: OLD-list ←OLD-list ∪ N;
21: End If
22: Else If ((LoadN(Qlength))<
    $TH_L$(Qlength))or($Load_N$(CPU-
    U)<$TH_L$(CPU-U))) then
23: ULD-list← ULD-list ∪ N;
24: Else BLD-list← BLD-list ∪ N;
25: End If
26: End For
27: Sort OLD_list by descending order relative
    to their $Load_N$(Qlength).
28: Sort ULD_list by ascending order relative
    to Their $Load_N$(Qlength).
29: If (events_happens ()=7) then //cluster is
    unbalanced
30: While (OLD-list ≠ ∅.AND. ULD-list ≠ ∅ ) do
31: For i = 1 To ULD-list. Size() do
32: send the decision of migration for
    AgentMigration (with address of first
    sender node of OLD List and its receiver
    node of ULD-list);
33: If an Acknowledgment received from
    Migration Agent
34: Update the current $Load_N$ of receiver and
    sender nodes
35: Update OLD-list, ULD-list and BLD-list;
36: Sort OLD-list by descending order of their
    $Load_N$( Qlength).;
37: End For
```

Job Migration Decision

After classifying the nodes, in the next step Cluster Agent decide to transfer jobs from overloaded to underloaded nodes. It sends this decision for Migration Agent.

Algorithm 3. An algorithm for AgentMigration

```
1: Receive decision of migration from its related
   Cluster Agent.
2: Sending the migration decisions to the Node
   Agents of sender and receiver node.
3: Wait for an Acknowledgment from Node agent of
   receiver node.
4: Send an Acknowledgment for its related Cluster
   Agent
```

3.2.2 Inter-cluster agent based load balancing algorithm

This algorithm applies a global load balancing among all clusters of the Grid. The Inter-cluster load balancing at this level is made if Cluster Agent fails to balance its load among its associated nodes. In this case the cluster agent transfers jobs to under loaded clusters based on the Decision taken by Grid Agent. the following algorithms are proposed:

Algorithm 4. An algorithm for Grid Agent

```
1:  Startup all Cluster Agents
2:  Receive jobs from grid user
3:  Send jobs for Node Agents
4:  If (events_happens ()=6) then //one of cluster is
    saturated
5:  Create underloaded_clusters_table;
6:  Sort clusters Cr of underloaded _clusters_table by
    Ascending order of their Load
7:  While (underloaded _clusters_table ≠ Φ) Do
8:  Sort the clusters Cr of underloaded _clusters_table
    by ascending order of inter clusters(Ci-Cr) WAN
    bandwidth sizes.
9:  Sort nodes of saturated cluster by descending
    order of their load
10: Sort Jobs of first node of saturated cluster by
    FCFS algorithm and communication cost
11: Migrate the selected job from the first node of
    saturated cluster to $j_{th}$ cluster of
    underloaded_clusters_table
12: Update load of sender and reciever cluster
13: Update ULD_clusters_table.
```

The last algorithm is implemented in Grid Agent which determines the way a receiver cluster is selected for a job migrated from overloaded cluster. Grid Agent calculates the minimum communication cost of sending jobs from saturated cluster to receiver underloaded cluster based on the information collected in the last exchange interval. Grid Agent selects the cluster that gives minimum overall cost.

3.3 Agents interactions

The proposed agent based load balancing algorithm is intended to take advantage of the agent characteristic to create a self-adaptive and self-sustaining load balancing system. It consists of five types of agents, in unbalanced situations, and if the Cluster Agent finds that there is a load imbalance between the nodes under its control, it uses the gathering event information policy to receive the load information from each Node Agent. On the basis of this information and the estimated equilibrium threshold, it analyses the current load of the cluster.

Depending on the result of this analysis, it decides whether to start a local balancing in case of an unbalanced state, or simply inform Grid Agent of its current load. Node Agent sends the updated local load value to Cluster Agent, which updates its load information. The local node load is calculated by the Node agent residing at each calculation node. Node Agent creates the task queue at the local node and updates it if necessary, and sends it for Cluster Agent based on the defined events. Migration Agent is responsible for migrating jobs to the selected underloaded node.

There is a Migration Agent in each cluster, who expects an acknowledgement of receipt from the receiving node once it receives the migrated job. The Migration Agent ensures that the work is successfully received and resumed or started at the destination node. The last agent is Grid Agent, it is the role of the distribution of work between clusters, all Cluster Agents are started by this type of agent and it decides whether to start a global load balancing in case of a saturated state.

4 Experimental results

4.1 Experimental environment

An experimental environment using Alea 2 as a grid simulator and JADE (Java Agent DEvelopment Framework) for agent implementation was set up to evaluate the effectiveness of the proposed algorithms. In the proposed infrastructure, management agents can communicate in a Grid environment using the Jade agent platform. In addition to Alea 2, a class library was developed that simulates the activities of an agent platform. This library, called ABLB (Agent based load balancing), includes the classes: Grid Agent Cluster Agent, Migration Agent and Node Agent.

4.2 Workload

The complex data set was modelled from the national Grid of the Czech Republic's MetaCentrum, which allowed to carry out very realistic simulations. It also provides information on machine failures and specific work requirements and this information influences the -quality of solu tions generated by scheduling algorithms. The job description includes (job ID, user, queue, number processors used, etc.).

The cluster description also includes detailed information such as RAM size, CPU speed, CPU architecture , operating system and list of supported properties (allowed queue(s), cluster location, network interface, etc.). In addition, the information machines were under maintenance (failure/restart). Finally, the list of queues containing their time limits and priorities is provided. More details on the trace file used can be found at [9].

4.3 Performance evaluation

The important performance factors in estimating the proposed algorithm is maximizing resource utilization. the use of resources was the main focus (%). The number of clusters was assumed to be 14, and each cluster was considered to be composed of different numbers of resources. The number of jobs was 3000. Figure 3 shows the use of the cluster with and without the proposed algorithm. It can be noticed that the agent based load balancing algorithm is more effective in maximizing resource utilization.

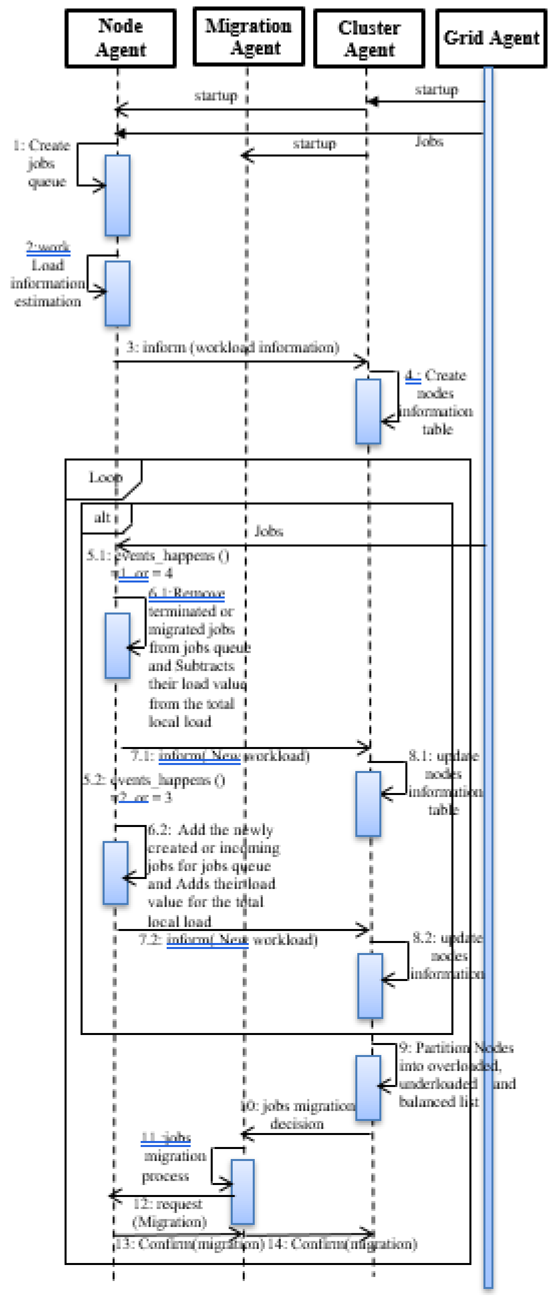
Node Agent
Migration Agent
Cluster Agent
Grid Agent
startup
startup
Jobs
startup
1: Create jobs queue
2:work Load information estimation
3: inform (workload information)
4: Create nodes information table
Loop
alt
Jobs
5.1: events_happens ()
n1_or = 4
6.1:Remove terminated or migrated jobs from jobs queue and Subtracts their load value from the total local load
7.1: inform(New workload)
8.1: update nodes information table
5.2: events_happens ()
n2_or = 3
6.2: Add the newly created or incoming jobs for jobs queue and Adds theirload value for the total local load
7.2: inform(New workload)
8.2: update nodes information
9: Partition Nodes into overloaded, underloaded and balanced list
10: jobs migration decision
11 jobs migration process
12: request (Migration)
13: Confirm(migration)
14: Confirm(migration)

Figure 2: UML Sequence Diagram describes agent interactions in intra cluster load balancing process.

The proposed algorithm allows job to be scattered over the most available resources when there was no appropriate resource, unlike other traditional algorithms that try to select the best resource that resembles the work requirements; otherwise, the job will remain in the global queue, indicating an underutilization of those resources.

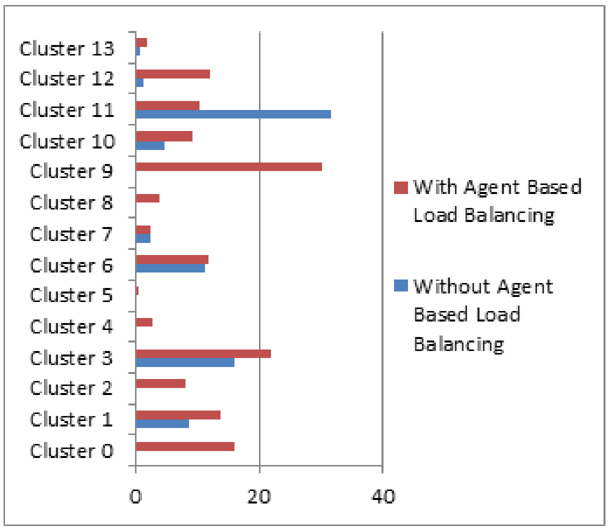

Figure 3: Comparison of cluster utilization (%) with and without Agent Based Load Balancing using 14 clusters.

5 Conclusion

The algorithms proposed under the Alea 2 simulator written in Java were developed to test and estimate the performance of the load balancing model based on the proposed agents. Experimental results showed that the proposed model allows a better balance of load and the correct use of resources. There are several approaches to improve resource utilization and reduce response time through coordination and cooperation among agents.

Therefore, the proposed model supports heterogeneity, scalability and dynamics of grids. In addition, a multi-agent architecture for grid load balancing was suggested, as well as a job migration technique to reduce the difference between overloaded and underloaded nodes. Finally, to estimate node load, the combination of CPU usage, memory usage and queue length was applied.

However, the problems of the model implemented included the reliability problem; there is no certainty that migrating work will resume in the reception node. The sender node does not keep a copy of the job until it is left at its new receiver node. Other solutions must be found to offer more reliability for migrating jobs. Moreover, the time required to complete a migration process is not explicitly calculated.

Hence, this study considered the comparison of the proposed algorithm with other agent-based load balancing algorithms, the cost of negotiation between agents, the use of a moving agent for load balancing, and the improvement and use of the proposed model in real grid environments.

6 References

[1] Brugnoli, M., Heymann, E., Senar, M.A., et al. "Grid scheduling based on collaborative random early detection strategies". 18th Euromicro Conf. Parallel,Distributed and Network-based Processing, Pisa, Italy, pp. 35–42 ,February 2010. https://doi.org/10.1109/PDP.2010.57

[2] Wu, J., Xu, X., Zhang, P.C., et al. "A novel multi-agent reinforcement learning approach for job scheduling in grid computing", Future Gener. Comput. Syst., 27, (5), pp. 430–439,2011. https://doi.org/10.1016/j.future.2010.10.009

[3] M.N. Satymbekov, I.T. Pak, L. Naizabayeva, and Ch.A. Nurzhanov. "Multi-agent grid system Agent-GRID with dynamic load balancing of cluster nodes", Open Engineering 7(1):485-490, December 2017.

[4] Soumya Banerjee and Joshua P. Hecker."Multi-Agent System Approach to Load-Balancing and Resource Allocation for Distributed Computing", First Complex Systems Digital Campus World E-Conference ,2015.

[5] Rina Suros, Juan Francisco Serrano. "Communication complexity in high-speed distributed computer network in an agent based architecture for grids service Ray Tracing View project", International Journal of Advanced Computer Research, Vol 8(35) ISSN (Print): 2249-7277, 22-February-2018. https://doi.org/10.19101/IJACR.2018.836002

[6] Hajoui Younes et al., "New load balancing Framework based on mobile AGENT and ant-colony optimization technique", Conference Intelligent Systems and Computer Vision (ISCV), At Fès,2017. https://doi.org/10.1109/ISACV.2017.8054961

[7] Rathore, N. "Efficient Agent Based Priority Scheduling and Load Balancing Using Fuzzy Logic in Grid Computing" , i-manager's Journal on Computer Science, 3(3), 11-22. 2015. https://doi.org/10.26634/jcom.3.3.3661

[8] B. Yagoubi, and M. Meddeber." Distributed Load Balancing Model for Grid Computing" , Revue ARIMA,Vol. 12,pp. 43-60,2010. http://www.cs.huji.ac.il/labs/parallel/workload/l_metacentrum/