https://doi.org/10.31449/inf.v45i2.3230 Informatica **45** (2021) 257–266 **257**

# Load Balancing Mechanism Using Mobile Agents

Sarra Cherbal
University of Ferhat Abbas Setif 1, Department of Computer Science, 19000 Setif, Algeria
E-mail : sarra_cherbal@univ-setif.dz

**Keywords:** distributed systems, load balancing, mobile agents, agents' technology

**Received:** July 8, 2020

*Distributed systems need to perform load balancing on their hosts, so that the computation runs as quickly as possible. Research in this field has seen the emergence of mobile agents' paradigm as a promising solution. In this present work, we use this paradigm to propose a load balancing approach that benefits from the advantages of agent mobility, more particularly, in the information-gathering phase. In which, we aim to have an overall system vision while reducing network communication overhead, as well as other benefits such as fault tolerance and extensibility for large scale networks. Thus, the purpose of our contribution is to improve the distribution of loads in a balanced way to get loads as close as possible to the system average load. The experimental results show the efficiency of our approach on balancing the loads and in decreasing the response time.*

*Povzetek: V prispevku je opisana nova agentna metoda za uravnoteženje obremenitev v porazdeljenih sistemih.*

## 1 Introduction

The trend of computer world towards "distributed systems" is no longer envisaging the operation of one single computer, without interacting or cooperating with other computers. These systems must be designed to meet the new requirements [1] [2] [3]. In this work, we present two related research areas: "Mobile agents" and "load balancing".

The technology of Mobile software agent is one of the known technologies in the field of distributed computing. It has emerged as an alternative to the classic "client / server" paradigm that presents the most widely used approach in building distributed applications [4] [5]. Mobile agent technology has interesting prospects for various application areas, among which, we find e-commerce, web-based information retrieval, and the domain of load balancing (LB).

In order to achieve better performances in distributed systems, the load balancing problem has been intensively studied by researchers [6] [7] [8]. Balancing allows maximum use of available resources, and this can be achieved by distributing tasks in a smart manner.

In this work, we study the use of mobile agents in the load-balancing domain. Based on this study, we propose an approach that aims to balance the system load using agent mobility by adopting three agents, and defining tasks for each of these agents. Where, the basic idea is to have local loads close to the system average load.

The rest paper is organized as follows. In section 2, we present an overview of related works and we give a corresponding discussion and the motivation of the proposed approach. Section 3 summaries the benefits of using mobile agents in distributed systems. The details of the proposal are explained in Section 4. In section 5, we present our simulation results to prove the efficiency of the proposed techniques. Finally, Section 6 concludes the paper.

## 2 Related work

### 2.1 Load balancing mechanisms

In [9], the authors employ a new concept of transferring virtual loads, which allows nodes to predict the future loads they will receive in the subsequent iterations. Accordingly, the notified node will take into account this predicted load when transferring a part of its actual load to some of its neighbors.

The work of [10] introduces a resource scheduling and a load balancing approach for efficient cloud service provisioning. They increase the use of virtual machines, and achieve load balancing by dynamically selecting a request from a class using Multidimensional Queuing Load Optimization algorithm.

The paper of [11] presents a load balancing methodology for container loading problem in road transportation. They propose a random-key genetic algorithm (BRKGA), with a new fitness function that takes static stability and load balance into account.

Authors of [12] propose a Dynamic Data Replication Algorithms (DDRA) of three phases to improve the information duplication under the cloud storage system. The first two phases are to determine the adequate service nodes to achieve the workload balance based on the nodes' workloads. The third phase presents a scheme of a dynamic duplication deployment proposed to realize a higher access performance and a better load balancing. In the first phase, for realizing the initial LB, the service nodes with probability lower than a defined threshold are filtered.

The authors of [13] propose a new variant of directed diffusion routing protocol in wireless sensor networks. This variant tries to improve the existing protocol by using

a load balancing mechanism in order to balance the energy of sensors.

## 2.2 Mobile agents in load balancing

Authors of [14] propose a dynamic load balancing in a small world P2P network using mobile agents. Firstly, they cluster the peers that have the same set of sharing resources. Then, they balance the query loads in intra-cluster nodes to avoid network congestion. For this purpose, three agents are defined, Host Agent (HA), Detection Agent (DA) and the Reconnection Agent (RA). The DA migrates between the intra-nodes to detect node congestions. This agent is generated and controlled by the HA that walks through all the groups. The RA is generated to reconnect the links between congested nodes and under-loaded nodes of the same cluster. An attractiveness parameter is measured for each node according to some criteria as node degree, processing capacity and the resources contained in the node. The DA migrates to the node with high attractiveness parameter, when the DA arrives, it checks if the node is overloaded or underloaded. However, the selection algorithm chooses a partner node that may refuse to accept the DA agent requests if it is overloaded. In this case, the DA agent migrates to other nodes until finding the right node. Thus, the right partner may not be correctly chosen before migration process.

In [15] they propose two different models of load management in large-scale distributed systems. The first model is based on mobile agents in collecting the status of node actual resources (RAM and CPU). To balance the loads, they integrate a second model based on fuzzy function. The agent system is consisting of three modules as Agent monitoring module (AMM) to collect the available status of current resources. The Agent Decision module (ADM) used to make a decision of migration when the node is overloaded if the measured probability crosses the threshold. The Agent migration module (AMMO) selects the destination node where to migrate, according to some criteria like the high level of processing and computing resources. However, in the agent's model, when defining the node status (overloaded/under-loaded), the authors did not mention how the used threshold is defined or measured. Thus, if it is a constant or a variable value and if it is measured locally or globally according to different node status.

In [16], load balancing is realized using mobile agents that migrate through all nodes based on a credit value factor. This factor is defined at initial stage to decide the node selection for migration. The selection of the destination node to which the agent will be migrated, is based on comparing the local load to a threshold value. If the destination node refuses the load reception, the agent will be directed to the next selected node. However, how to measure the threshold value is not defined. Besides, the agents keep migrating until finding the right partner for each overloaded node.

The paper of [17] presents a load balancing algorithm for parallel virtual architectures. They use a host agent to perform a diagnostic test to evaluate the computation performance and latency of each node. Each parallel task is assigned to a VPU (virtual processing unit) presented by a mobile agent. These VPUs communicate by exchanging messages containing data and tasks to be performed. However, the communication between VPUs of different nodes leads to more traffic.

The paper of [18] distinguishes five populations of agents. The authors use the ant-colony optimization approach to distribute the tasks between the worker agents in a parallel way. Such as, a dispatcher agent (load balancer agent) collects the necessary information for the scheduling decisions, and then the ant colony approach is used to take this decision. In the context of their research, the scheduler is used as an ant that chooses for the current job, the machine having the higher rate of pheromone. To measure this pheromone probability, each worker machine sends via its controller agent, to the dispatcher agent, the number of tasks that it has in its queue.

In [19], the authors combine the least time of first byte algorithm (LFB) and the mobile agent concept. Where, the mobile agent role is monitoring the state of each server resources. However, the authors focus on achieving a reliable approach without taking into consideration the system throughput and latency.

The paper of [20] presents a load balancing scheme in heterogeneous P2P systems. They propose three agents. The first mobile agent is for information gathering and the second one is to decide the receiver partner for the overloaded node. The third is a stationary agent, which resides on each node to update its routing table. Its role is to inform all the network nodes of the other nodes' addresses and spread the node's failure information in the network. This mechanism of using messages to spread information each time to all the nodes lead to more overhead.

A load balancing algorithm for heterogeneous P2P systems is proposed in [21]. In which, one type of mobile agent is used with its essential components such as collection, analysis and location. A utilization rate is measured according to the load and capacity of each node. However, to migrate the overload, the authors propose to choose the neighbor node with the smallest utilization rate. Thus, this one under-loaded node is going to be chosen by the most of overloaded nodes or all of them. Consequently, the chosen node will be overloaded or will refuse the receiving tasks, which leads to rerun location process and try again with other nodes, thus, more time is wasting in location, migration and execution.

## 2.3 Discussion

In this section of related work, we have reviewed different papers of load balancing approaches existing in the literature. We can notice that this field in general still relevant with the mentioned recent approaches. Sub-section 2.2 concerns propositions of load balancing based on mobile agents. According to this literature study, in our proposed approach, we aim to avoid and improve some existing limits or weak points, mentioned as follows.

In the selection process, it concerns selecting the partner node to which the overload will be migrated, i.e. selecting an under-loaded node for an overloaded node. In

this process, in [14], the authors propose to migrate a mobile agent (MA) between a set of nodes until finding an under-loaded node. When the MA arrives to a node, it checks if this node can receive the holding overload, if it rejects the reception, the MA will pass to another from a sorted list [14] [15] [21] and repeat the same verification steps. Thus, this method leads to a wasting time in migration process especially that the MA is holding the overload when migrating, which makes its migration slower. Thus, the system latency is increased and it could find a good partner but not the suitable one. Therefore, in our work, we propose a selection process that aims to find the suitable partner node for each overloaded node and that before launching the load migration to avoid the mentioned disadvantages. In other words, our migration agent goes directly to a defined and a right node destination. In addition, in the selection process, some works like [21] propose to choose the node with the smallest load to be the partner node. consequently, most of overloaded nodes going to choose the same partner, this partner is going to be overloaded and the next receiving agents will be rejected and they should migrate to find another partner. To avoid this, in our approach we propose a method to choose one under-loaded node for each overloaded node without causing its overload.

In general, the node state is defined as overloaded or under loaded according to a threshold value. In some works like [15] [16], there is no mention on how the threshold is defined or measured. Such, if this threshold is measured locally according to the node local information or globally according to the system global information. Therefore, in our approach, we measure the threshold value according to the global overview, based on all the states of nodes. This, in order to have almost equally states for all the nodes, which is the main principle of load balancing mechanism.

Unlike [19], in our approach, we are interested not only by achieving a balanced system but also by avoiding the increase of throughput and by reducing the latency. Thus, we use mobile agents and their migration in a way to avoid exchanging messages between the nodes, unlike [17] [20].

In the next section, we summary the benefits of using mobile agents in distributed systems.

# 3 Mobile agents and load balancing

Mobile agents' technology provides a load balancing support with three main characteristics:

- They can move from one platform to another
- They can move across platforms of heterogeneous nature (e.g. OS, capacity of CPU, Storage, …)
- They carry the application specific code, instead of requiring a pre-installation of this code on the destination machine.

The two main advantages that mobile agent approach brings to load balancing are:

- Reducing network traffic
- Migrating processes from one site to another.

# 4 The proposed architecture

In this section, we present our proposed contribution of load balancing based on mobile agent paradigm.

In the following, we assume that the nodes are in a cooperative network, which allows us to study only the characteristics of the considered algorithm, and we assume that the tasks are independent i.e. there is no communication between them.

In this work, we are interested by the "dynamic" load balancing, in which, the movement of tasks depends on the current load of the processors. For the balancing decision, we chose the "source-initiative" approach in which a site called source tries to transfer its surplus (excess) towards a weakly charged site called receiver. For the choice of this latter, we propose a method that tries to obtain local loads close to the average load (AL) of the system, which is the goal of load balancing.

To determine the state of each machine, thresholds must be defined: there are static thresholds that are unchangeable fixed values and other dynamic thresholds that are varied according to the evolution of the system. In our approach, we use the second solution where the threshold is defined according to the average load of the system, which is more suitable to a dynamic environment.

Our contribution consists of three types of agents, a fixed agent and two mobile agents. Each one implements one or more load balancing policies (Figure 1):

- ***Observer Agent (OA):*** a fixed agent located on each site of the system. It evaluates and monitors the local load of the site and so launches the migration process to the partner site (which is indicated by the SMA agent).
- ***Supervisor Mobile Agent (SMA):*** a mobile agent that moves from one site to another in order to build a global vision of the system. It puts the collected information (the local load value + the site identifier) in a table. After finishing the course between the sites, SMA calculates the average load, according to which it builds two tables: a table of overloaded sites (in a descending order) and other table of under loaded sites (in an ascending order). With these two tables, SMA determines the receiving site for each overloaded site (section 4.2.2.).
- ***Transporter Mobile Agent (TMA):*** a mobile agent that migrates between two unbalanced machines in order to balance the system load. It is launched by the OA of an overloaded site, in order to transport the excess load (tasks) of this site to the defined partner by SMA.

The scenario of each of these agents is explained in the following of this section (section 4.2).

## 4.1 Suggestions and critics

In this sub-section, we aim to explain the benefit of each proposed process in this approach, by showing the different cases that can arise.

If we eliminate the concept of "tables and their ordering" (i.e. SMA agent only calculates the average), three cases arise:

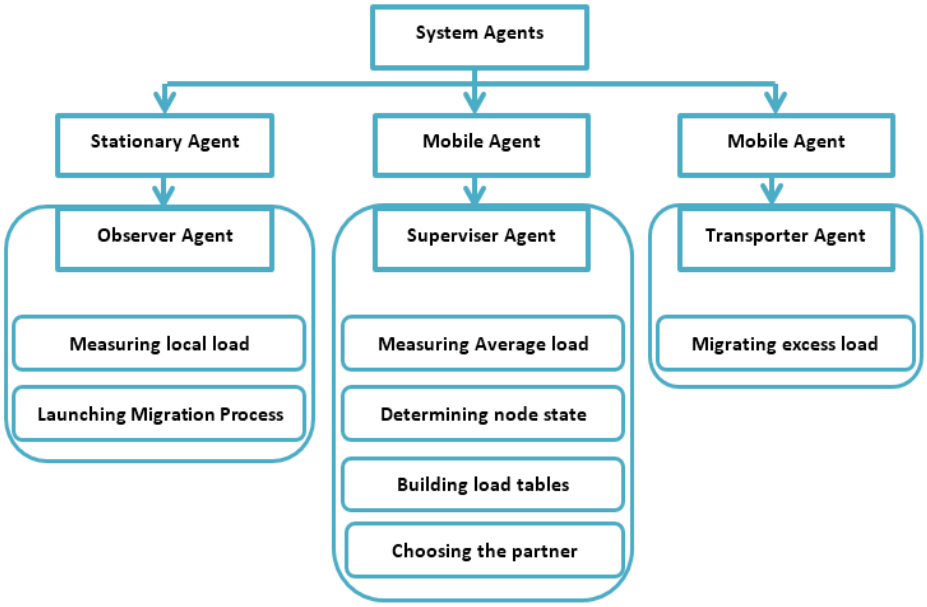


Figure 1: The role of each proposed agent.

## 4.2 Suggestions and critics

In this sub-section, we aim to explain the benefit of each proposed process in this approach, by showing the different cases that can arise.

If we eliminate the concept of "tables and their ordering" (i.e. SMA agent only calculates the average), three cases arise:

### 4.2.1 First case

When an observer agent $OA_i$ detects that its site is under-loaded, it sends a broadcast message to all other sites in the system (including underloaded sites) to inform them of its state. In case of an overloaded site Sj, its corresponding OAj sends a message to $OA_i$ to see if it can transmit its excess load. Three cases can be distinguished:

a. $OA_i$ has already accepted the offer of a site $S_k$, so it will reject the offer of OAj.
b. The excess load of site Sj can cause the overloading of Si, thereby, a refusal message will be sent to the agent OAj,
c. The site $S_i$ accepts the offer of Sj, so an acceptance message will be sent to Sj.

***Disadvantages***

A very high cost of communication, especially in case of a large network (N sites for example). There will be:

- N-1 broadcast messages for each imbalance ,
- M messages from overloaded sites (M < N),
- M messages of refusal / acceptance.

### 4.2.2 Second case

TMA agent is responsible for determining the partner node of the overloaded node Sj on which it was created. TMA moves from one node to another while handling the surplus of the node Sj, then, it chooses the first node that has a less load than Sj.

***Disadvantages***

a. The size of TMA becomes larger when it is handling the excess load, and it has no vision about its destination, thus, it can go through several overloaded sites before arriving to the suitable one, which is not favored in a dynamic load-balancing system. The size of TMA should be as small as possible, in order to speed up the load transfer before that the local loads of the sites change again, otherwise, this transfer becomes unnecessary.
b. This surplus can cause the overloading of the recipient site, while it is possible to find a more suitable recipient. This latter can receive the surplus while still having a load close to the system AL (which is the purpose of load balancing).

### 4.2.3 Third case

If we eliminate the order of tables (ascending/descending), the agent OAj of an overloaded site Sj will choose the node with the lowest load to ensure effective balancing.

***Disadvantage***

All OAs that are on overloaded nodes will choose the same node that is weakly loaded, thus, this latter becomes overloaded.

## 4.3 Description of the proposed agents

### 4.3.1 Observer Agent (OA)

The aim of our algorithm is to make the system in a balanced state, and thus having balanced machines. It is the role of the OA to observe the machine load, on which it is located:

**a. Measuring the node local load**

There are two methods to measure the load, one is on demand, and the other is periodic:

- On demand: in this case, the OA measures the node load when the SMA asks him to do it (i.e. when the

SMA arrives on this node). Thus, OA must know the arrival time of SMA, which means that SMA must inform OA with a signaling message. Otherwise, SMA must wait until OA calculates the local load, which increases the load collection time of SMA.

- Periodic: in this case, OA does not need to know the arrival time of SMA since it measures its site local load each period Tinfo. When SMA arrives on this site, the OA provides him the last measured load value. The Tinfo must be an appropriate period to avoid system overload. If the loads change quickly, then the Tinfo must be small. Otherwise, it is necessary to increase the duration of Tinfo.

In our contribution, we chose to use the second method because it is simple to implement, it does not require remote message exchanges between SMA and OA and it reduces the time allocated to the information collection policy.

**b. Start the migration process**

When the OA of site Sj receives a message from SMA, he gets that his site is overloaded. Therefore, this OA launches the TMA agent, by indicating the surplus to transport, using the average load sent by SMA. In addition, the destination site is declared in the message of SMA.
This load transport between two unbalanced machines makes their loads closer to the system average load, in order to make them better balanced.

**Algorithm.1: Observer Agent script**

```
For each site Sk {
For each period Tinfo {
Measure local load Lk  //Lk is the Local load of site Sk
}
Receive_info(SMA, Sr, AL);   //SMA indicates the recipient site Sr.
                             //AL is the average load
If (Received_info != NULL) {
        Launch the TMA agent and send it to the site Sr
Else                         //Sk is not overloaded
        OA declares: "My site is lightly loaded"
OA declares: "I am waiting for the reception of transporter agents from other sites"
}
}
```

### 4.3.2 Supervisor Mobile Agent (SMA)

The SMA moves cyclically between the various sites of the network. For each site, SMA retrieves the site name and its load value and put them in a table (Tab_System). After finishing this movement, SMA makes the two next steps.

**a. Calculation of the system average load (AL)**

At the arrival of the SMA agent on a site, it contacts the OA, to retrieve the local load. Then, SMA adds this load value to the values already collected. When the SMA finishes its trajectory, it calculates the average load "AL". This AL allows to :

- Achieve a global load balancing, since each machine load is compared to the system overall load.
- Manage load variations that may occur in the system, which is not the case when determining a fixed threshold.

**b. Determining the node state and building the two tables**

We use the average load (AL) as "threshold": according to which we can distinguish 3 states:

- The site is Overloaded If its load > AL;
- The site is Underloaded If its load < AL;
- Neutral If its load = AL.

From the Tab_System table and the calculated average load, the SMA agent builds two sub-tables:

- Tab_Less : contains underloaded sites.
- Tab_plus : contains overloaded sites.

After that, SMA will order the load values for each of these tables:

- Tab_Less: in ascending order (from the smallest load value to the greatest).
- Tab_Plus: in descending order.

**c. Choosing the partner**

After these two steps, a global vision is built, and now SMA plays the role of a distributor. It determines for each overloaded site, a partner (a receiver under-loaded site), in a way that the overloaded site and its partner have the same table index, respectively in tab_plus and tab_less.
In a detailed way, the method that we proposed to make the choice of partner is to assign the first entry of Tab_Less to the first entry of Tab_Plus. This, in order to assign the site of the biggest load to the site of the smallest load), and assign the second entry of Tab_Less to the second entry of Tab_Plus, and so on. I.e. each underloaded site Tab_Less [i] is the partner of the overloaded site Tab_Plus [i].
Noting that the mobile agent have to be fast during its move so that the system load information doesn't change during this movement i.e. the agent must be up to date, thus, the size of the SMA code must be minimized.
In our architecture, we propose to use a single supervisor agent but we can always consider multiplying the number of these agents and this depending on the network size (number of participating machines in the application). In this case, it is necessary to provide a cooperation mechanism between the different supervisory agents in order to build a global vision on the system load.

**Algorithm.2: script of Supervisor Mobile Agent**

```
Som=0 ; AL=0 ;
For m = (from 1 to nbSites) {  // nbSites is the number of machines or nodes
Request the local load from the OA of site Sj
Add(site name Sj + Lj load) as an entry in tab_system
Som=Som+Lj     // Lj: local load provided by agent OA of site Sj
doMove (Sj, Sk)    // SMA moves to the next node
}
AL = Som / nbSites ;
While Tab_Sys[i] not_empty {
        If (Tab_System [i] > AL)
```

```
                Add this site as an entry to tab_plus
        Else {
                If (Tab_Systeme [i] < AL)
                Add this site as an entry to tab_less
            }
i++;   //next hut in tab_sys
}
Ordering Tab_Less in ascending order;
Ordering Tab_Plus in descending order;
While tab_plus [i] not_empty {  //indicating partners
So = Tab_Plus[i]  // source overloaded site
Su = Tab_Moins[i]  //receiver underloaded site
Send_info (Sk, Sj ,AL)  // SMA informs the overload site by the receiver of its surplus load
}
```

#### 4.3.3 Transporter Mobile Agent (TMA)

It is a mobile agent launched by OA agent of an overloaded site and it represents the surplus of this latter. The TMA purpose is to transport this surplus to the recipient site, on which the task(s) will be executed.
It is the OA agent that indicates to the TMA the work to handle. This work presents a set of tasks. The number of TMA agents depends on how many tasks they can handle.

**Algorithm.3: script of Transporter Mobile Agent**

```
doMove(Sk,Sj) ;
// TMA moves from the overloaded site to the receiver site
Send (L_Plusk, Sk, Sj) ;          // Sk : the source site name
                                  // L_Plusk : excess load of site Sk
    // Sj: the recipient site name (indicated in SMA script)
```

An overview of our system architecture with a scenario of the three agents on 4 sites, is presented in Figure 2.

# 5 Experimental results

## 5.1 State of loads

To test the behavior and to evaluate the performances of the proposed approach, our agents' algorithms are developed using Java eclipse IDE on top of the JADE agent platform. We use the "FSMBehaviour" for OA and SMA agents. This behavior serves to present complex tasks, it responds to the needs of the compound behaviors of these two agents. For the third TMA agent that has only one task to perform, a simple behavior meets the need, so, in this case it is the "OneShotBehaviour". The source code is composed of three classes and of their sub-classes according to the roles in Figure 1.

In order to launch our algorithm execution under Jade, we first launch the observer agents (we implement eight agents OA0,OA1,…), each on a container. Then, we launch SMA agent on the main container. Next, the AMT agent is launched by the corresponding OA agent (that of an overloaded site).
A load ratio is assigned randomly to each container. Figure 3 presents the loads' state in the three phases: before applying LB, and when applying the proposed LB with and without table approach. Where, in the third phase (without table approach), the TMA transports the overload to the first found under-loaded site. Figure 3 shows that in the first phase, the loads are unbalanced (from very high to very low loads). In the third phase, the loads start to be balanced which shows the impact of the LB approach. Thereafter, in the second phase, the loads are more balanced compared to the system overall state, which demonstrates the efficiency of the proposed table approach in choosing the right partner to receive the overload.

## 5.2 The impact of LB on response time

In distributed computing, the response time is a significant issue and the fact of reducing it is a very important requirement in improvement approaches. However, when the loads are bigger, the system response time is higher. In this type of environment, a set of tasks is distributed between some nodes of the system. The system response time is the time taken by all the participating nodes to realize this set of tasks.
We measure the response time using the following formula:

$$T = (NbTasks \times HighL \times ReqAT)/100 \qquad (1)$$

Such that:

- NbTask is the total number of tasks,
- HighL is the higher load percentage in this system,
- ReqAT is the required average time to execute one task.

Figure 4 shows the change of response time in the three phases, while increasing the number of tasks to be executed by this system (between 200, 300, 400 & 500). The results prove the efficiency of our proposed LB algorithm in reducing the response time comparing to the two other phases.

## 5.3 Impact of our proposed selection technique

The main purpose of applying a LB process in a distributed system is that all the nodes work with almost equal workloads. In other words, minimize the difference in workload percentage between nodes and so avoid having a node with a high workload while another node is under-loaded. In a distributed system, where the nodes work in collaboration to complete a specific set of tasks, balancing the workload and thus the effort between the nodes is essential to reduce the execution time (or response time) of a task and thus of all the launched tasks. Therefore, in a system, we measure the difference of workload rates between the most heavily loaded nodes and the least loaded ones. Thus, the difference between the highest and the lowest workload rate.

In the selection technique, an overloaded node (site) selects one of the under-loaded nodes where to transmit the overload, what we call a partner. In our approach, we

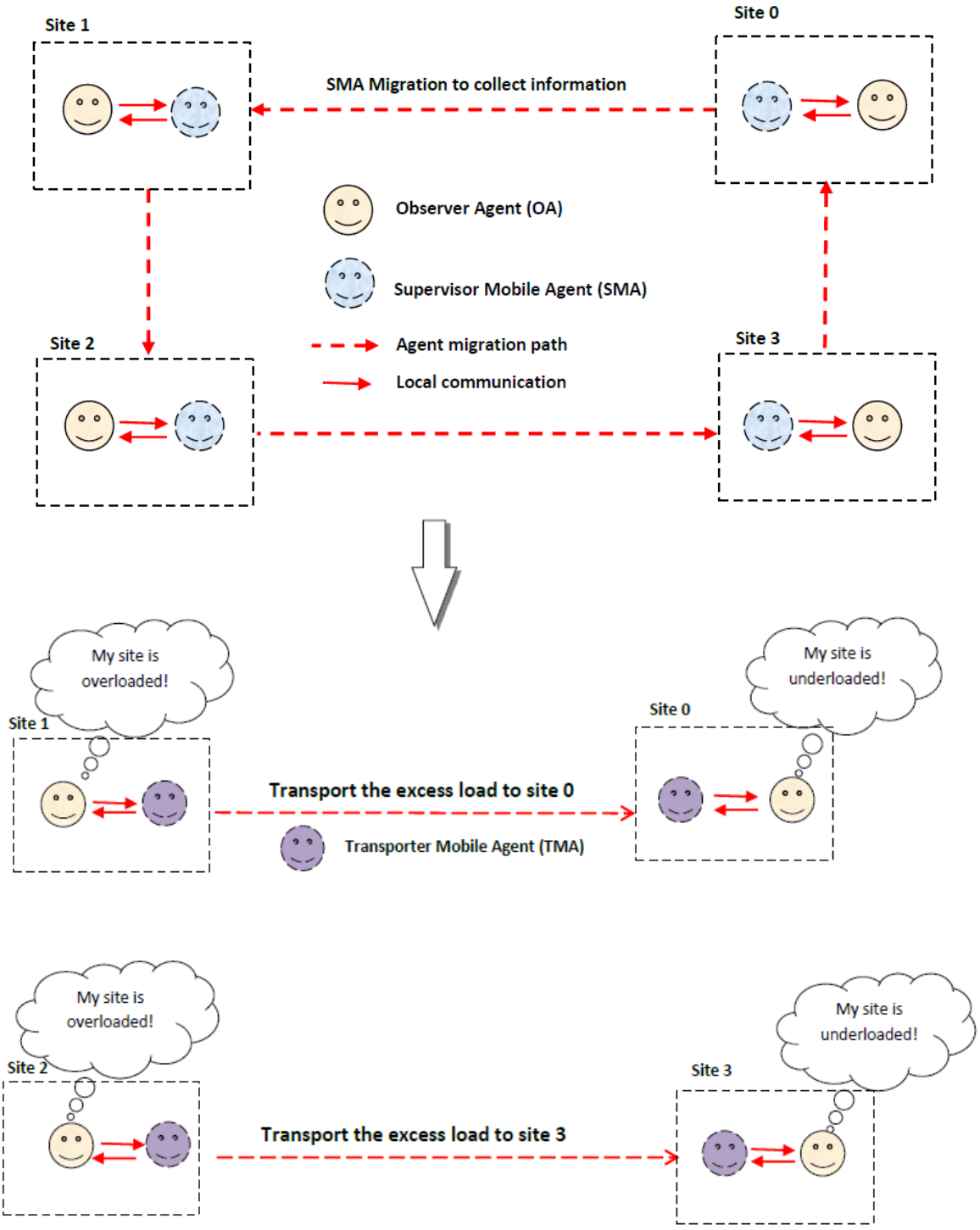


Figure 2: Architecture of the proposed system.

propose a selection technique based on a defined table as explained earlier in this paper.

To prove the efficiency of our proposal, in this sub section, we compare it to the selection technique proposed in [14] and in [21] in terms of workload differences and response time.

The selection technique of [14] is based on sorting the under-loaded nodes in a list, according to an attractiveness parameter and in [21] the list is sorted according to the utilization rate. Then, each overloaded node uses this list to find the least charged node to transmit him the overload. When the migration agent arrives to the selected node, it checks if this node is overloaded so it accepts the received overload or reject it if it is under-loaded. In case of rejection, the migration agent moves to the next node from the sorted list.

Figure 5 presents the difference of workload rates between the most heavily loaded site (node) and least loaded site in each simulation (case). This parameter is compared in three phases, the first phase is before applying LB, the second is when applying our proposed approach and the third is when applying LB with the


Figure 3: State of loads in the three phases.

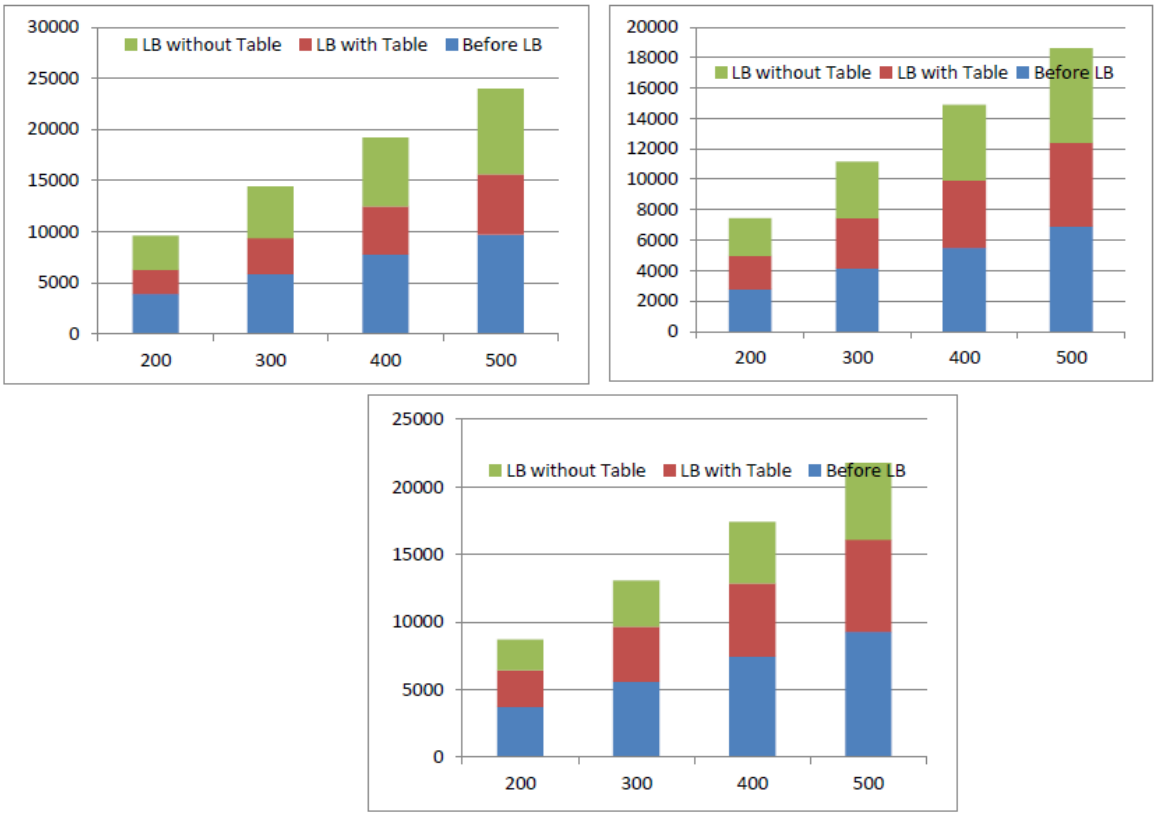


Figure 4: Response time as function of number of tasks.

selection techniques of [14][21]. A workload rate is assigned to each node in a random manner, and we have launched eight simulations (8 cases). The results in Figure 5 show that the differences in workload rates are more reduced after applying a LB approach and then are much reduced when applying our proposed approach. These results prove the efficiency of our proposed selection technique compared to existing techniques in selecting the

most suitable under-loaded node for each overloaded node and thus all the nodes work with almost equal workload rates.

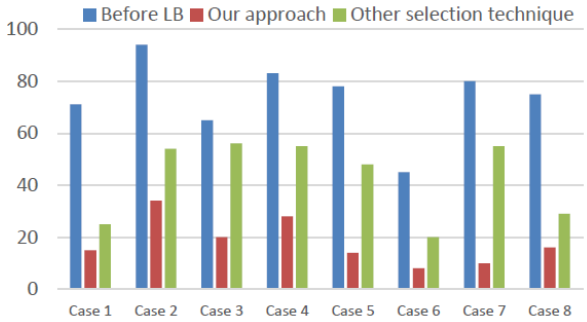


Figure 5: Differences in workload rates between most heavily loaded sites and least loaded sites

# 6 Conclusion

Load balancing is one of the keystones for the improvement of system performances. Its main objectives are to improve the execution time of tasks and to take advantage of the maximum system resources.

In this work, we are interested in the use of mobile agent technology in the field of load balancing. One of the important motivations of this paradigm is the minimization of remote communications thus saving the bandwidth consumption, which is favorable for an efficient load balancing system. For this reason, we have integrated in our proposed solution a mobile agent whose role is to collect the loads information to build a global vision on the system load, which reduces the communication cost compared to the classic collection method. In addition, the stationary agents must be aware


Figure 6: Impact of selection technique in response time as function of number of tasks.

Figure 6 shows the impact of the used selection technique on the Response time while changing the number of tasks from 200, 300, 400 to 500. In sub-section 5.2, we explain the signification of response time and we define formula (1) to measure it. The results presented in Figure 6 correspond to three launched simulations with different random workloads. Figure 6 shows that the response time increases with the increase of number of tasks with more workloads to perform. Besides, our proposed approach presents the most reduced response time in three launched simulations. This proves the efficiency of our selection technique in choosing the partner node and thus all the nodes participate with almost same effort to finish a task, and then the set of tasks achievement is done faster.

of the global load so they can detect the machine balance state and so select the tasks to be migrated. In our solution, the mobile supervisory agent only informs overloaded hosts which conducts in a traffic reduction compared to those approaches that inform all the hosts of the system.

Other benefits of mobile agents are robustness and fault tolerance, which are necessary in a load balancing system so that it can continue to operate when one of the members is disconnected. Mobile agents also offer the advantage of scalability, they adapt well to small networks as to large-scale networks. These benefits conduct us to use mobile agents in our solution, to have an extensible and a robust load balancing system.

In a large-scale network, increasing the number of supervisory agents is possible to reduce the agent

migration time and to avoid increasing the agent code size. This allows an improvement in load balancing. However, we find that determining the number of agents needs another study. Furthermore, we have implemented our proposal on Jade platform. By a comparative evaluation, the results showed the efficiency of the load balancing approach. Besides, we have shown its impact on reducing the execution time latency, which is an important factor in distributed systems. In addition, we have shown the impact of our proposed selection techniques compared to existing selection techniques in balancing the workloads and in reducing the system response time.

As a perspective, we aim to implement the proposal architecture on mobile nodes to show the effect on energy consumption.

# References


[1] Van Steen M, Tanenbaum AS (2016) A brief introduction to distributed systems. Computing. vol. 98, no. 10, pp. 967-1009. https://doi.org/10.1007/s00607-016-0508-7

[2] Anjomshoa MF, Salleh M, Kermani MP (2015) A Taxonomy and Survey of Distributed Computing Systems. Journal of Applied Sciences, Vol. 15, pp. 46-57. DOI: 10.3923/jas.2015.46.57

[3] Samolej S. and Rak T. (2009) Simulation and Performance Analysis of Distributed Internet Systems Using TCPNs, Informatica, vol. 33, pp. 405-415.

[4] Hakansson A, Hartung R (2013) Book: Agent and Multi-Agent Systems in Distributed Systems - Digital Economy and E-Commerce. Publisher: Springer-Verlag Berlin Heidelberg. DOI: 10.1007/978-3-642-35208-9

[5] Hajduk M, Sukop M, Haun M (2019) Cognitive Multi-agent Systems : Structures, Strategies and Applications to Mobile Robotics and Robosoccer. In Series of Studies in Systems, Decision and Control. DOI: 10.1007/978-3-319-93687-1

[6] Wided A. and Okba K. (2019) A novel Agent Based Load Balancing Model for maximizing resource utilization in Grid Computing, Informatica, vol. 43, no. 3. https://doi.org/10.31449/inf.v43i3.2944

[7] M. A. Salehi, H. Deldari, and B. M. Dorri (2009) Balancing load in a computational grid applying adaptive, intelligent colonies of ants, Informatica, vol. 33, no. 2, pp. 159–168.

[8] Aghdashi A, Mirtaheri SL (2019) A Survey on Load Balancing in Cloud Systems for Big Data Applications. In: Grandinetti L., Mirtaheri S., Shahbazian R. (eds) High-Performance Computing and Big Data Analysis. TopHPC 2019. Communications in Computer and Information Science, vol 891. Springer, Cham. DOI: https://doi.org/10.1007/978-3-030-33495-6_13

[9] Couturier R, Giersch A, Hakem M (2018) Best effort strategy and virtual load for asynchronous iterative load balancing. Journal of Computational Science. Vol. 26, Pages 118-127. DOI: https://doi.org/10.1016/j.jocs.2018.04.002

[10] Priya V, Kumar CS, Kannan R (2019) Resource scheduling algorithm with load balancing for cloud service provisioning. Appl Soft Comput Vol. 76, pp. 416–424. https://doi.org/10.1016/j.asoc.2018.12.021

[11] Ramos AG, Silva E, Oliveira JF (2018) A new load balance methodology for container loading problem in road transportation. European Journal of Operational Research, Vol. 266(3), pp. 1140–1152. https://doi.org/10.1016/j.ejor.2017.10.050

[12] Hsieh H, Chiang M (2019) The Incremental Load Balance Cloud Algorithm by Using Dynamic Data Deployment. J Grid Computing Vol. 17, pp. 553–575. https://doi.org/10.1007/s10723-019-09474-2

[13] Semchedine F, Bouallouche-Medjkoune L, Tamert M, Mahfoud F, Aïssani D (2015) Load balancing mechanism for data-centric routing in wireless sensor networks. J. Comput. Electrical Eng. Vol. 41, pp. 395–406. https://doi.org/10.1016/j.compeleceng.2014.03.005

[14] Shen X-J et al (2014) Achieving dynamic load balancing through mobile agents in small world P2P networks, Comput. Netw., vol. 75, pp. 134-148, Dec. https://doi.org/10.1016/j.comnet.2014.05.003

[15] Ali M, Bagchi S (2019) Design and analysis of distributed load management: Mobile agent. based probabilistic model and fuzzy integrated model. Appl Intell, Vol. 49, pp. 3464-3489. https://doi.org/10.1007/s10489-019-01454-z

[16] Metawei MA, Ghoneim SA, Haggag SM, Nassar SM (2012) Load balancing in distributed multi-agent computing systems, Ain Shams Engineering Journal, Vol. 3, pp. 237-249, May 2012. https://doi.org/10.1016/j.asej.2012.03.001

[17] Youssfi M, Bouattane O, Bensalah, M (2015) Efficient Load Balancing Algorithm for Distributed Systems Using Mobile Agents , Advanced Studies in Theoretical Physics Vol. 9, no. 5, pp.245 – 253. DOI:10.12988/ASTP.2015.5110

[18] Younes H, Bouattane O, Youssfi M, Illoussamen E (2017) New load balancing framework based on mobile AGENT and ant-colony optimization technique. Intelligent Systems and Computer Vision (ISCV), Fez, 2017, pp. 1-6. DOI: 10.1109/ISACV.2017.8054961

[19] Afriansyah MF, Somantri M, Riyadi MA (2017) Model of load balancing using reliable algorithm with multi-agent system. IOP Conference Series: Materials Science and Engineering, Volume 190, conference 1. doi: 10.1088/1757-899X/190/1/012033

[20] Nehra N, Patel RB, Bhat VK (2008) Load Balancing in Heterogeneous P2P Systems using Mobile Agents. International Journal of Electrical and Computer Engineering Vol:2, No:8, pp. 2740-2745.

[21] Li H (2011) Load Balancing Algorithm for Heterogeneous P2P Systems Based on Mobile Agent. in IJCA Proceedings on International Conference on Electronics, Information and Communication Engineering (ICEICE) (Foundation of Computer Science, New York, 2011), pp. 1446–1449. DOI: 10.1109/ICEICE.2011.5777758