Journal of Theoretical and Applied Information Technology
10th November 2015. Vol.81. No.1
© 2005 - 2015 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

NEW MODEL OF FRAMEWORK FOR TASK SCHEDULING BASED ON MOBILE AGENTS

[1]YOUNES HAJOUI, [2]MOHAMED YOUSSFI, [3]OMAR BOUATTANE, [4]ELHOCEIN ILLOUSSAMEN

Laboratory SSDIA

ENSET Mohammedia, University Hassan II of Casablanca, Morocco

E-mail: [1]hajouiyounes@gmail.com , [2]med@youssfi.net, [3]o.bouattane@gmail.com, [4]illous@hotmail.com

ABSTRACT

Compute-intensive applications and applications with high volume of data need strong processing power and considerable storage resources. To reach the required performance, multiple machines should be associated in order to handle the distributed tasks. In this paper, we propose a new framework for task distribution based on mobile agents. In the proposed model, a dispatcher agent is used to distribute parallel tasks to worker agents. Each worker agent is deployed in a node of the distributed system according to the load balancing system. The proposed framework is build using three layers which are the user task producer, the scheduling load balancing layer and the workers layer. After presenting the architecture and the structure of the proposed model, an example of application, relating to the distributed image processing, is presented to improve the performance of this framework.

Keywords: *Task scheduling; Parallel Computing; Distributed System; Framework; Multi-Agent system; Load balancing.*

1. INTRODUCTION

In the intensive computation domain, the applications need more powerful processing models and high storage resources availability and computing. As examples, we find: weather forecasts, financial projects, scientific simulations of mechanical, aerodynamic, electrical or molecular biology problems, and other application of high volume of data such as image processing, multimedia and video games. In this domain, the use of a single processor machines proves the performance limit of these sequential models. Since the technologies of microprocessors, the data storage units, and the networking systems, are continuously evolving, the limits of the obtained supercomputers are quickly bounded by the large amount of data and processing applications. Subsequently, it is necessary to combine the performance of several processing and storage units to overcome these limitations. Parallel and distributed models became the natural solutions for several types of computational systems.

Several studies are published in the literature on the scheduling tasks on distributed system and scheduling policies in grid computing. It has been shown that scheduling has a great influence and impact on the performance and cost-efficiency [1,4].

In [5] authors define a task routing model allowing the use of shared resources. In this model, tasks are scheduled on the assigned workstation to exploit the considerable power of the distributed system. Two routing policies are defined: static and adaptive. In the static policy distributed jobs are based on using information about the average behavior of the system. In the adaptive policies, the scheduler use information about processor queues to take decision about load scheduling.

In the static policy, two techniques are used: deterministic and probabilistic. In deterministic policies, a routing algorithm is applied when tasks are ready and specified [5]. In the probabilistic case, tasks are sent to workstations in an arbitrary manner, the machines have the same probability of receiving these tasks from the scheduler.

Scheduling algorithms are also classified into immediate and delayed/batch modes. In the immediate mode, the algorithm transmits arrivals tasks as soon as possible to the workstations designed to perform distributed jobs [6]. While in the batch mode, algorithms distribute all tasks of a job queued in the scheduler [7].

To exploit the considerable power of the distributed system, jobs must be partitioned into several tasks using algorithms evaluated in [8, 9]. These tasks are distributed to multiple machines aiming to reduce the cost of communication latency and of execution time.

In this paper, we propose a new framework for task distribution based on mobile agents. In the proposed model, a dispatcher agent is used to distribute parallel tasks to worker agents. Each worker agent is deployed in a node of the distributed system according to the load balancing system. The proposed framework is build using three layers which are the user task producer, the scheduling load balancing layer and the workers layer. In this work we will present an implementation based on JADE [10] multi agent system framework.

The structure of this paper is as follows. In the second section, we describe the architecture of the proposed model. In this part we also present the load balancing system used in task routing. In Section III, an example of application using Multiple Program Multiple Data (MPMD) architecture and, relating to the distributed image processing, is presented to improve the performance of this framework. In the last section, conclusions and perspective are presented.

2. PROPOSED FRAMEWORK

2.1 Architecture of Framework

The proposed load distribution system aims to use the agent characteristics to create an autonomous system. A real multi-agent system is implemented using the JADE platform (Java Agent Development Framework). It simplifies the implementation of multi-agent systems

The solution that we propose aims to develop a multi-agent system to distribute tasks on a cluster of heterogeneous distributed machines, using intelligent agents or software entities which can delegate specific tasks.

These mobile agents are distributed in different nodes of the system and can dynamically collect the performance of grid nodes to calculate and execute the tasks that they receive. They do not know in advance what code or when they will run it. These same agents co-operate processors of different nodes to move the load of overloaded ones to those that are less loaded.

As shown in Figure 1, the system goes through three main stages, our proposed Framework is based on 3 layers: User Producer, Task Distribution - Load Balancing, and Layer3: Workers. Each layer will be detailed later in this article.

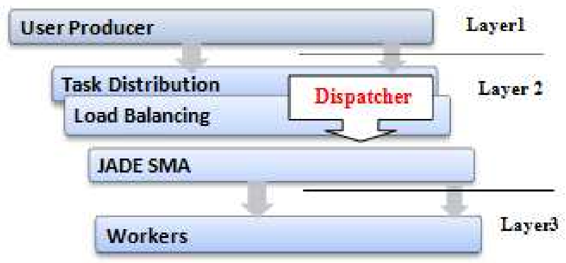

Figure 1 : The layers of our Framework

We distinguished 5 different types of intelligent agents in our proposal:

1. Producer Agent
2. Dispatcher Agent
3. Tester Agent
4. Controller Agent
5. Worker Agent

<u>Layer 1: User Producer</u>

In this first layer, users prepare the tasks they want to perform. Then these tasks migrate to layer 2 via: Producer Agent. The layer 2 is preoccupied with these tasks and sends them to Layer 3 via the Dispatcher agent, this agent checks the status of availability of workers machines then it affects Workers agent of layer 3 to received tasks. The results are then returned to the client applications: Producers Agent.

The preparation of tasks is made by extending from abstract class AbstractTask, the developer is asked to define the algorithm and the instructions to execute, and define the data needed by extending from the class AbstractTaskDataObject.

The following class diagram illustrates the preparation of a task by the user.

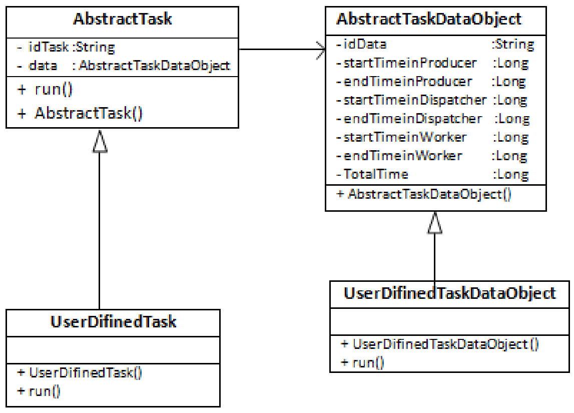

Figure 2: Class diagram illustrating preparation of a Task

Before beginning scheduling tasks to the worker machines, the first step consists to determine the performance and the cost of communication: Latency of these nodes, this will allow the distribution algorithm to take scheduling decisions. For this, a reference task T_0 is performed before starting the distribution process by the VPU (Virtual Processing Unit) Controller of all Workers.

Each VPU Controller communicates to Layer 2 these parameters: Performance and latency, which will allow the Framework to optimize the distribution.

Latency θLi is the cost of communication time taken by the task T_0; since his departure from the producer agent, through the Dispatcher that determines where it will go, until his return after its execution by the worker.

$\theta Pi(T_K)$: Processing duration of the task T_K, by the node i.

$\theta Li(T_K)$: Communication Latency required to perform the Task T_K, in the node i.

$\theta i(T_K)$: Total Time required to perform the task T_K.

The communication Latency of each node Ni, can be easily calculated by :$\theta Li = \theta i - \theta Pi$.

The following use case diagram illustrates the phase of referencing with the task T_0:

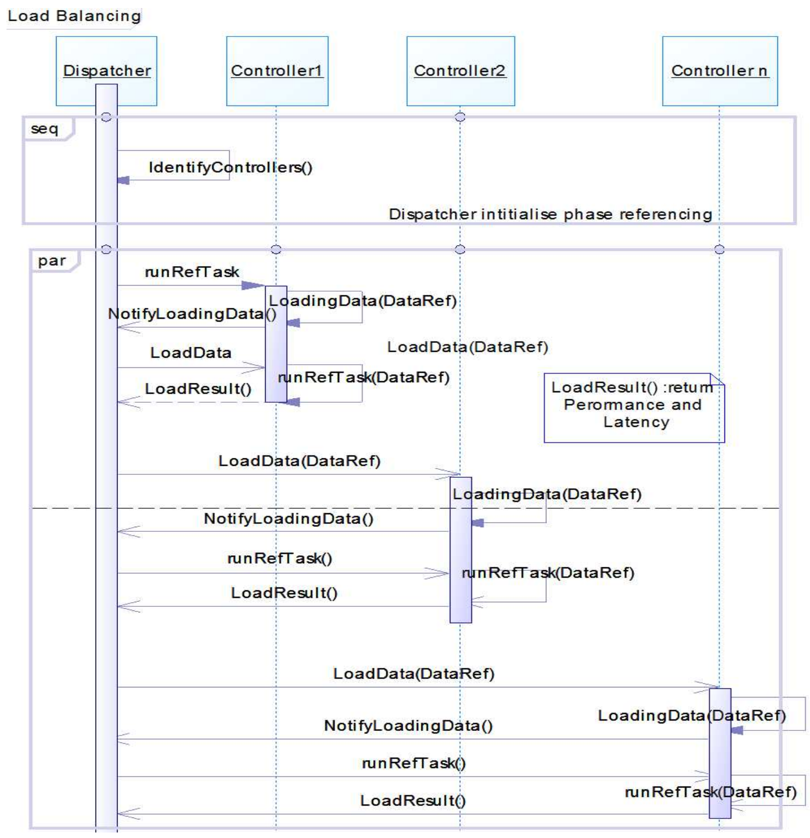

Figure 3: Sequence diagram of referencing phase

Layer 2: Task Distribution - Load Balancing

At this layer, the Dispatcher Agent aims to distribute and schedule tasks. It receives them from the layer 1, measures them with a Tester Agent, and then checks the status of availability of Worker processors of layer 3. Next, it sends them to be performed. The Dispatcher agent orders tasks following rules defined by the distribution algorithm which is based on a data structure that logs exchanges with Workers.

According to this history, the Dispatcher takes decisions to schedule and send tasks that will be received and performed by the VPUs. Figure 7 shows use case of the scheduling of tasks by the Dispatcher.

Layer 3: Workers

In this layer, a cluster of Worker machines which are referenced by the layer 2 are ready to contribute to the parallel computing and to execute the tasks received from the Dispatcher.

Each node is represented by a VPU called Controller, this Controller, as cited in layer 1, performs a reference task T_0 to determine the performance of the node to which it belongs and determine the cost of communication: latency.

After the referencing phase, the controllers periodically collect the state of the processor node Workers. This is done through the communication of each controller, using the located VPUs. These parameters are communicated to the Dispatcher by the ACL messages; the latter uses these parameters to feed the distribution algorithm.

The Dispatcher performs in a periodic manner the distribution algorithm. This algorithm assigns each received task, Based on its evaluation with the Tester Agent, to adequate Worker. In parallel this Dispatcher decides if it is necessary to perform a load balancing program.

Figure 8 illustrates our conceptual description of the Framework for Parallel task distribution.

2.1 Load Balancing

Load balancing is the important objective for multi-objective optimization in the task scheduling problem in the grid. It aims to optimize resource use, maximize throughput, minimize response time, and avoid the overload of any single resource. The authors have used in [11] a mobile agent, which migrates the loads from overloaded nodes to under loaded ones, Each task in the distributed system should be assigned to a VPU.

The VPUs can communicate with each other asynchronously by exchanging through their ports ACL messages. These messages contain data and tasks to be performed.

As already mentioned in the layer 3, Worker machines are referenced periodically by layer 2, these latter are ready to contribute in parallel computing and execute tasks which are sent by the Dispatcher. After the referencing phase, the Controllers Agent periodically collects the state of Worker processors based on their communication with different VPUs deployed in these machines, and then the Controllers communicate the status of the processors and latency to the Dispatcher via the ACL messages which decides if it is necessary to perform a load balancing. In case of imbalance, it determines the overloaded nodes (sources) and the under-loaded ones (receivers).

Migration task is one of the techniques used in load balancing problems, Agent migration is adopted to support load balance in multi-agent systems. Before taking decisions of migration, the Dispatcher must take the following decisions:

- Which agent should be migrated?
- Where should this agent go?

Then it starts the transfer of tasks to establish the balancing of the system, this transfer is done by the migration of VPUs from overloaded nodes to under-loaded ones

If the selection is made with care, migrant agent will not cause an overload on the destination node and the cost of migration and balancing will be Optimal.

3. APPLICATION AND REULTS

To test the proposed Framework of distribution, a multi-agent system is implemented to analyze the results obtained by our parallel design, and our distribution algorithm. We worked on a grid of 8 heterogeneous machines.

The referencing of Worker machines is done by a reference task T_0 in order to determine the latency and power of each machine. Next, it allows the distribution algorithm, the knowledge of the network topology of the platform and reduces scheduling to the slower machines as well as to those which have the most expensive communications. Table 1 above shows the latencies calculated during this referencing phase:

Node i	θi (Total Time) (ms)	θPi(Processing duration by node i)(ms)	θLi (Latency) (ms)
1	2690	2595	95
2	2700	2570	130
3	2830	2688	142
4	2845	2694	151
5	2910	2751	159
6	2870	2700	170
7	3000	2810	190
8	2900	2700	200

Table 1: Referencing phase

The curve of Figure 4 shows the latencies of different machines employed in this work. This parameter is calculated by the following formula: θLi = θi – θPi .

The diagram of Figure 5 shows the Performance of those machines. These parameters are considered very important to take the scheduling decisions by the Load Balancer Algorithm.

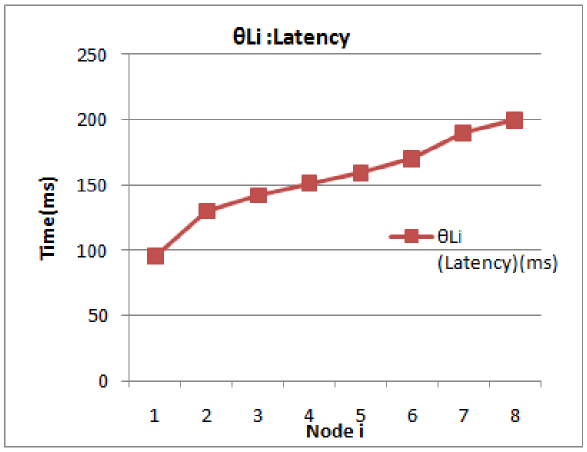

Figure 4: Latency diagram of different machines in Grid

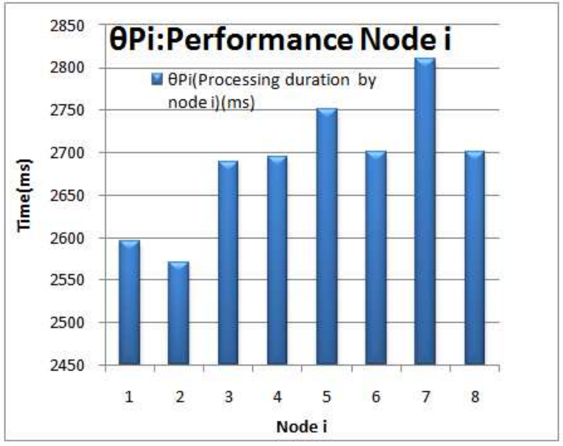

Figure 5: Performance diagram of different machines in Grid

After the referencing phase, that's aims the knowledge and classification of nodes in the grid according to their powers and their latencies, we start the distribution of tasks, it is launching the parallel execution of 300 tasks on a grid of 1, 3,5, and 8 machines. As shown in Table 2, this distribute system, allows reducing the overall execution time of these 300 tasks compared to the sequential system. The tasks that we are distributed are prepared by the extending of the **AbstractTask** class, the data of these tasks are defined by the extending of **AbstractTaskDataObject** class; in this case the algorithm that we have chosen to do the test, is Sobel ,for detecting the contour, on the list of 300 images.

	Number of Machines			
	Sequential 1 Machine	3 Machines	5 machines	8 machines
execution duration(ms) (90 images)	300	131	73	47
execution duration(ms) (300 images)	1000	400	250	180

Table 2: The execution time in different grid

Figure 6 above shows that the curve of the running time is decreasing; with more machines we optimize the distribution.

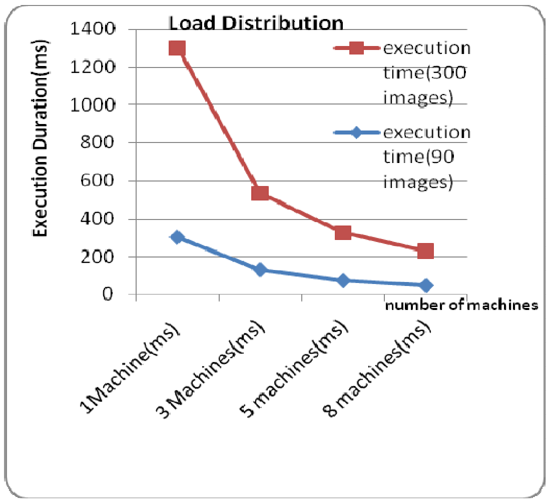

Figure 6: Curve of the execution time in different grid

4. CONCLUSION

In this paper we have proposed a new framework for parallel distribution based on mobile agents, this latter is based on agents deployed on every node of the distributed system that's provides dynamically, from their respective nodes, to the Dispatcher/Scheduler agent, all information needed to make scheduling decisions in a way that maximizes the system performance and minimizes execution time.

As perspective, we intend to complete our work in order to optimize the work of scheduler to maximize the use of all resources and minimize the overall maximum execution time, the scheduler will become a load balancer, which will cooperates different processors and will start the migration tasks from under loaded nodes to the overloaded ones.

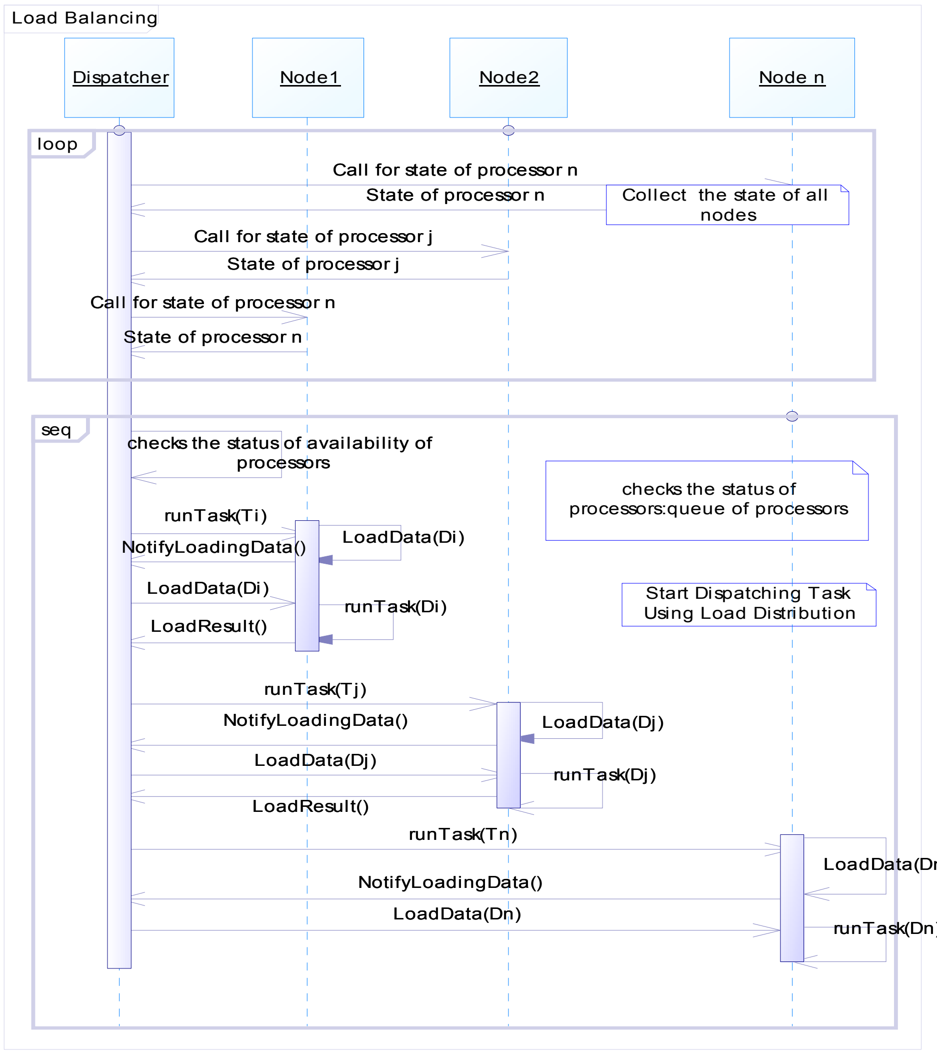

Figure 7: Sequence diagram of Tasks distribution phase

Journal of Theoretical and Applied Information Technology
10th November 2015. Vol.81. No.1
© 2005 - 2015 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

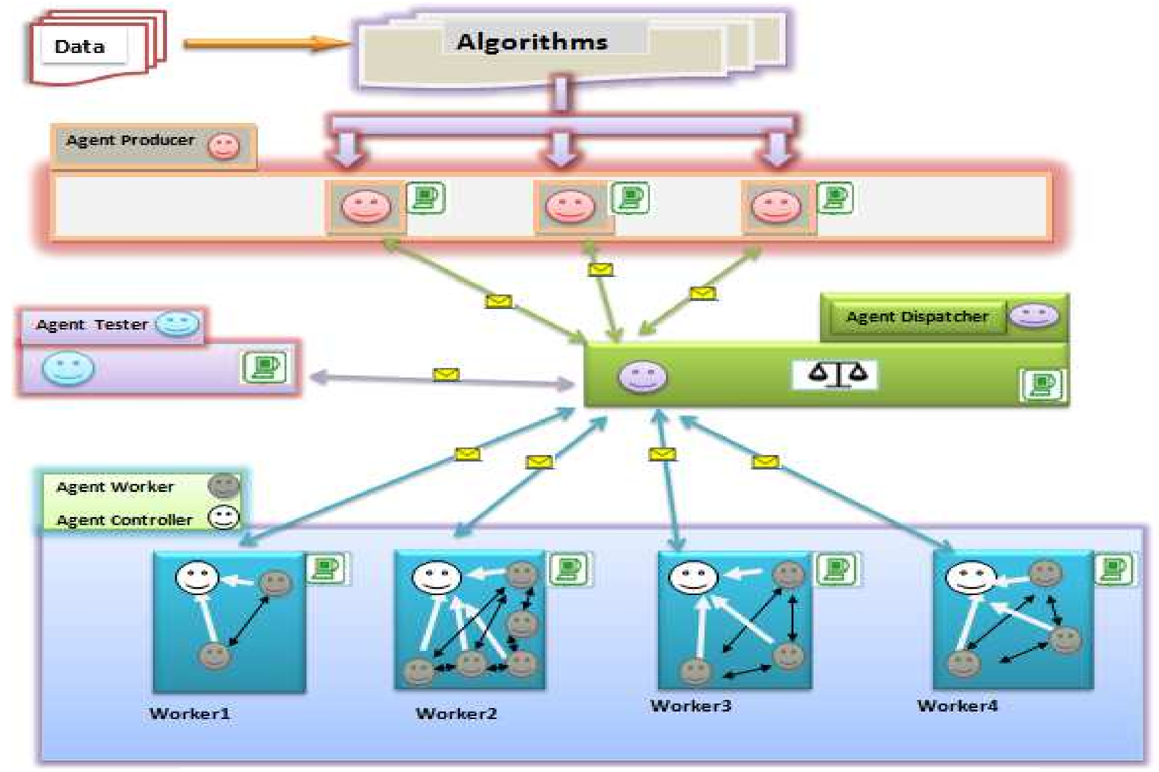

Figure 8: Conceptual description of the Framework for Parallel task distribution

REFERENCES

[1] A. Chowdhury, L.D. Nicklas, S.K. Setia, E.L. White, Supporting dynamic space-sharing on clustersof non-dedicated workstations, in: Proceedings of the 17th International Conference on Distributed Computing Systems (ICDS '97), IEEE, Baltimore, MD, 28–30 May 1997, pp. 149–158.

[2] H.D. Karatza, Task Scheduling Performance in Distributed Systems with Time Varying Workload, Neural, Parallel & Scientific Computations, Dynamic Publishers, Atlanta, 10 (3) (2002) 325–338.

[3] M.S. Squillante, Y. Zhang, A. Sivasubramaniam, N. Gautam, H. Franke, J. Moreira, Modeling and analysis of dynamic coscheduling in parallel and distributed environments, in: Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems, ACM, New York, NY, 2002, pp. 43–54.

[4] Y. Zhang, A. Sivasubramaniam, Scheduling best-effort and real-time pipelined applications on timeshared clusters, in: Proceedings of the 13th Annual ACM Symposium on Parallel algorithms and Architectures, Crete Island, Greece, 2001, pp. 209–219.

[5] S.P. Dandamudi, Performance implications of task routing and task scheduling strategies for multiprocessor systems, in: Proceedings of the IEEE Euromicro Conference on Massively Parallel Computing Systems, Ischia, Italy, May 1994, pp. 348–353.

[6] F. Xhafa, L. Barolli, A. Durresi, Immediate mode scheduling of independent jobs in computational grids, in: Proceedings of the 21st International Conference on Advanced Networking and Applications (AINA'07), May 2007, IEEE, 2007, pp. 970–977.

[7] F. Xhafa, L. Barolli, A. Durresi, Batch mode scheduling in grid systems, International Journal of Web and Grid Services 3 (1) (2007) 19–37

[8] J. Aguilar, E. Gelenbe, Task assignment and transaction clustering heuristics for distributed systems, Information Sciences, vol. 97, Elsevier Science, Amsterdam, Netherlands, 1997, pp. 199–219.

[9] Ming Wu, Xian-He Sun, Memory conscious task partition and scheduling in grid environments, in: Fifth IEEE/ACM International Workshop on Grid Computing (GRID'04), 2004, pp. 138–145

[10] F. L. Bellifemine, G.Caire, and D. Greenwood, “Developing MultiAgent Systems with JADE”. Wiley, 2007.

[11] Mohamed Youssfi, Omar Bouattane, Jamila Bakkoury, Mohammed Ouadi Bensalah, “A new massively parallel and distributed virtual machine model using mobile agents “ IEEE International Conference on Multimedia Computing and Systems (ICMCS), 2014, 14-16 April 2014, pp.407 – 414, ISBN: 978-1-4799-3823-0 Marrakech, Morocco DOI: 10.1109/ ICMCS.2014.6911306.