Extended Max-Min Scheduling using Petri Net and Load Balancing

El-Sayed T. El-kenawy, Ali Ibraheem El-Desoky, Mohamed F. Al-rahamawy

Abstract*— Max-min algorithm is based on comprehensive study of the impact of RASA algorithm in scheduling tasks and the atom concept of Max-min strategy. An Improved unique version of Max-min algorithm is proposed to outperform scheduling map at least similar to RASA map in total complete time for submitted jobs. Improved Max-min is based on the expected execution time instead of complete time as a selection basis. We employ Petri nets which are well suited for modeling the concurrent behavior of distributed systems. Experimental results show availability of load balance in small cloud computing environment and total small makespan in large-scale distributed system; cloud computing. In turn scheduling tasks within cloud computing using Improved Max-min demonstrates achieving schedules with comparable lower makespan rather than RASA and original Max-min.

Index Terms*— Distributed System, Job Dispatching Algorithms and Cloud Computing.

Petri net, Load Balance, Quality of Service, Meta Task Scheduling, Max-min Algorithm, Min-min Algorithm

I. INTRODUCTION

Cloud computing is known as a provider of dynamic services using very large scalable and virtualized resources over the Internet. Cloud computing can be defined as a collection of computing and communication resources located over distributed datacenters; that is shared by many different users [1]. As shown before, cloud computing is considered as internet based computing service provided by various infrastructure providers on an on-demand basis, so that cloud is subject to Quality of Service (QoS), Load Balance (LB) and other constraints which have direct effect on user consumption of resources controlled by cloud infrastructure. Cloud Computing is considered nowadays to be a very popular because of the many advantages provided by the Cloud infrastructure. Hardware, software and other services are available to users as a utility under an on-demand basis that is charged proportionally to the amount of resources consumed by them. In some cases, Cloud providers use a portion of their datacenter infrastructure for private purposes and provide the rest unused capacity as a cloud service to public clients.

Such setting enables cloud to increase the complexity of its resources efficiently and makes providers earn money from such deployments. On the other side of service providing, the users become more comfortable and make them not worry about the infrastructure required and its troubles shooting for their services [1], [2]. To make a set of cloud services an effective provider infrastructure, one of its requirements is an effective task scheduling algorithm. Task scheduling algorithm is responsible for mapping jobs submitted to cloud environment onto available resources in such a way that the total response time, the makespan, is minimized [2]. Many task scheduling algorithms are applied by resources manager in distributed computing to optimally allocate resources to tasks [4], [5], [6], [7], [8], [9], [10], [11]. While some of these algorithms try to minimize the total completion time. Where the minimization is not necessarily related to the execution time of each single task, but the aim is to minimize overall the completion time of all tasks [5], [11], [13], [14], [15], [16].

There have been many algorithms used to schedule tasks on their resources, some of these algorithms are used in grid computing which is large scale distributed system concerned with resource sharing and coordination for problem solving. Three well known examples of such algorithms intended to be applied in cloud computing environment are Max-min, Min-min and RASA [2], [5], [11], [13], [14], [16]. Each of these algorithms estimate the completion and execution time of each submitted task on each available resource. RASA is a hybrid algorithm of two other ones. In the RASA, an estimation of the completion time of each task on the available resources is calculated then Max-min and Min-min algorithms are applied alternatively to take advantage of both algorithm and avoids their drawbacks [2].

One of the features of the Max-min strategy is that chooses large tasks to be executed firstly, which in turn small task delays for long time. On the other hand, Min-min is perfect in executing smaller tasks then large ones that is the reverse of Max-min. So that, in RASA, alternating between small and large is reason for executing small tasks before large and avoids delays of executing large tasks, also support concurrency in execution of large and small tasks. Max-min strategy resolves the difficulty of Min-min, by giving priority to large tasks. The Max-min algorithm selects the task with the maximum completion time and assigns it to the resource on which achieve minimum execution time. It is clear the Max-min seems better choice whenever the number of small tasks is much more than large ones. But in other cases, early executing large tasks leads for increasing in total completion time of submitted tasks so Min-min is better choice and visa-verse [2].

This paper, as RASA, offers an improved task scheduling algorithm based on Max-min to resolve the mentioned above problems with both Max-min and Min-min. The basic idea of an improved version of Max-min assign task with maximum execution time to resource produces minimum complete time rather than original Max-min assign task with maximum completion time to resource with minimum execution time.

The remaining parts of this paper are organized as follows: Section 2 presents some related works. Next, Section 3 describes the methodologies such as Petri net and the concept of Task scheduling algorithm in distributed environment using Max-min strategy which is modified in Section 4. Then in 4, our improved modified version of Max-min schema is proposed and illustrated using pseudo code and flowchart.

Manuscript received September 02, 2012.

El-Sayed T. El-kenawy Dep. of Comp. and Sys. Eng., Faculty of Engineering, Mansoura University, Egypt.

Ali Ibraheem El-Desoky Dep. of Comp. and Sys. Eng., Faculty of Engineering, Mansoura University, Egypt

Mohamed F. Al-rahamawy Dep. of Computer scienecs, Faculty of Computer and Info., Mansoura University, Egypt.

In Section 5, compare the scheduling algorithms in typical environment and present the result of comparison using illustrative simple example. Finally, Section 6 concludes the paper and presents future work.

II. RELATED WORKS

Due to novelty of cloud computing field, there is no many standard task scheduling algorithm used in cloud environment. Especially that in cloud, there is high communication costs that prevents well known task schedulers to be applied in large scale distributed environment [5], [9], [10]. Today, researchers attempt to build job scheduling algorithms that are compatible and applicable in Cloud Computing environment.

L. Mohammed Khanli et al. have proposed QoS tasks scheduling algorithm as an aggregation formula in a specific architecture called Grid-JQA [6], [7]. Such formula is a combination of parameters and weighting factors to evaluate QoS. Khanli's scheduling algorithm is not practical as it hasn't a practical mathematical solution [2], [7].

X. He et al. have proposed an algorithm depends on the original Min-min algorithm [5]. It is called QoS guided Min-min, and it assigns tasks with high bandwidth before others. QoS acts similar to Min-min when available tasks have the same bandwidth so it preferred to use QoS guided Min-min whenever submitted tasks have large bandwidth. At that moment, QoS guided Min-min produces better results.

Similar to QoS guided Min-min, new algorithm called QoS priority grouping scheduling that is proposed by F. Dong et al [14]. QoS priority grouping scheduling algorithm considers deadline and acceptation rate of the tasks and makespan of the whole system as major factors for task scheduling. It achieves better acceptance rate and completion time for submitted tasks compared with Min-min and QoS guided Min-min.

QoS Sufferage is new task scheduling algorithm presented by E. Ullah Munir [15]. This algorithm considers network bandwidth and assigns tasks based on their bandwidth requirement as the QoS guided Min-min does. It achieves smaller makespan compared to Max-min, Min-min; QoS guided Min-min and QoS priority grouping algorithms.

K. Etminani et al. provided a new algorithm, that uses Max-min and Min-min algorithms to select one of these two algorithms depending on standard deviation of the expected completion times of the tasks on each of the resources [16].

Saeed Parsa et al. proposed a new task scheduling algorithm called RASA [2]. It takes advantage of both Max-min and Min-min algorithm. RASA uses the Min-min strategy to execute small tasks before large ones and applies the Max-min strategy to avoid delays in the execution of the large tasks and to support concurrency in the execution of large and small tasks.

III. METHODOLOGY

A. *TASK SCHEDULING ALGORITHMS*

Task scheduling process is an allocation of one or more time intervals to one or more resources [18]. In cloud computing, the scheduling is a problem of scheduling a set of submitted tasks from different users on a set of computing resources to minimize the completion time of a specific task or the makespan of a system. There are many other parameters can be mentioned as factor of scheduling problem to be considered such as load balancing, system throughput, service reliability, service cost, system utilization and so forth. Through comprehensive study of scheduling, Task scheduling algorithm is a decision making process about assigning and finding the best match between tasks and resources. So scheduling is NP-complete problem [5], [13], [18].

For producing a schedule, assume that we have m Resources Rj (R1, R2, .., Rm) and we process n tasks Ti (T1, T2, .., Tn) to be mapped on these resources. Also expected execution time Eij of task Ti on resource Rj is defined as required time of resource Rj to finish task Ti provided that Rj has no load when assignment occurs. On the other side, expected completion time Cij of task Ti on resource Rj is defined as the overall time consumption till finishing any assigned task previously assigned. Assume ri denote the beginning of execution task Ti. From previous mentions, it can be concluded that Cij = ri + Eij. The makespan of complete schedule is defined as Max (Ci) where Ci is the completion time for a task Ti [2].

Makespan is defined as a measure of the throughput of the heterogeneous computing system; like the Cloud Computing environment [11], [13]. Scheduling algorithms can be categorized according to many polices as immediate and batch scheduling, preemptive and non-preemptive scheduling, static and dynamic scheduling, etc [20]. In Immediate mode, tasks are scheduled as soon as arrive the computing environment, while in the batch mode, tasks are grouped into a batch; that is a set of meta-tasks would be allocated at times called mapping events [21]. For example, in the Minimum Execution Time (MET) algorithm estimating the execution time of the submitted tasks on available resources is calculated, choosing each task to a resource would produce the minimum execution time for that task [5], [11], [13], [16].

In contrast, the Max-min, Min-min and RASA algorithms estimate the execution time and the completion time of each task in meta-tasks; then assign the tasks on suitable resource; each based on its decision rule. The Max-min algorithm is commonly used in distributed environment which begins with a set of unscheduled tasks. Then calculate the expected execution matrix and expected completion time of each task on the available resources. Next, choose the task with overall maximum expected completion time and assign it to the resource with minimum overall execution time. Finally recently scheduled task is removed from the meta-tasks set, update all calculated times, then repeat until meta-tasks set become empty [11].

In the Max-min algorithm, shown in Fig 1, rj represents the ready time of resource Rj to execute a task, while Cij and Eij represent the expected completion time and Execution time respectively. As shown, task Tk with maximum expected completion time is chosen to be assigned for corresponding resource Rj that gives minimum execution time.

1. for all submitted tasks in meta-task; Ti
2. for all resources; Rj
3. Cij = Eij + rj
4. While meta-task is not empty
5. find task Tk consumes maximum completion time.
6. assign Tk to resource Rj with minimum execution time.
7. remove Tk from meta-tasks set
8. update rj for selected Rj
9. update Cij for all j

Fig 1: The Max-Min Algorithm

Each of Max-min, Min-min and RASA algorithms have running time complexity of O(mn2), where m is the number of resources currently in the system and n is the number of submitted tasks which should be scheduled [2].

B. PETRI NETS

A Petri net [22] consists of *places*, *transitions*, and arcs. Arcs run from a place to a transition or vice versa, never between places or between transitions. The places from which an arc runs to a transition are called the *input places* of the transition; the places to which arcs run from a transition are called the *output places* of the transition. Petri nets are state-transition systems that extend a class of nets called elementary nets, with nondeterministic execution.

Graphically, places in a Petri net may contain a discrete number of marks called *tokens*. Any distribution of tokens over the places will represent a configuration of the net called a *marking*. In an abstract sense relating to a Petri net diagram, a transition of a Petri net may *fire* whenever there are sufficient tokens at the start of all input arcs; when it fires, it consumes these tokens, and places tokens at the end of all output arcs. A firing is atomic step.

Definition 1. A *net* is a triple $N = (P, T, F)$ where:

1. P is a set of states, called *places*.
2. T is a set of *transitions*.
3. F where $F \subset (P \times T) \cup (T \times P)$ is a set of flow relations called "arcs" between places and transitions (and between transitions and places). A net is a bipartite graph, where P is one partition and T is the other. Moreover, for every t in T there exist p and q in P so that (p, t) and (t, q) are in F and for every p and q in P, if (p, t) and (t, q) are in F then $p \neq q$.

The set $P \cup T$ are the net *elements*. The set of places define the local states of a net, however, the global state of a net can be defined by place subsets.

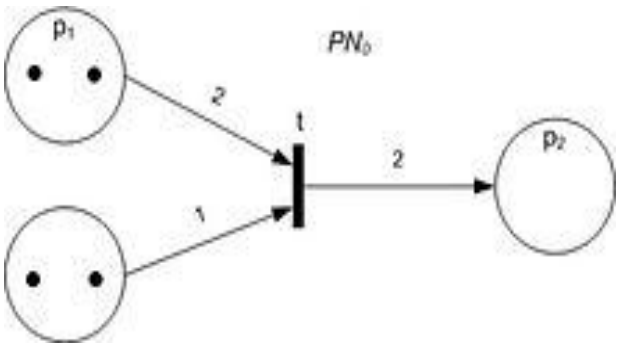
Definition 2. Given a net $N = (P, T, F)$, a *configuration* is a set C so that $C \subseteq P$.

Definition 3. An *elementary net* is a net of the form $EN = (N, C)$ where:

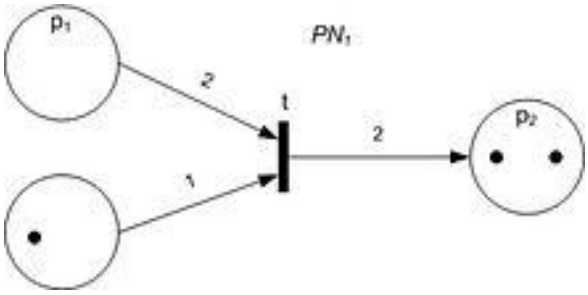
1. $N = (P, T, F)$ is a net.
2. C is such that $C \subseteq P$ is a *configuration*.

Definition 4. A *Petri net* is a net of the form $PN = (N, M, W)$, which extends the elementary net so that:

1. $N = (P, T, F)$ is a net.
2. M so that $M : P \rightarrow Z$ is a place multiset, where Z is a countable set. M extends the concept of *configuration* and is commonly described with reference to Petri net diagrams as a *marking*.
3. W so that $W : F \rightarrow Z$ is an arc multiset, so that the count for each arc is a measure arc *multiplicity*.



a) before



b) after

Fig 2 Petri net before and after the transition fires

If a Petri net is equivalent to an elementary net, then Z can be the countable set {0,1} and those elements in P that map to 1 under M form a configuration. Similarly, if a Petri net is not an elementary net, then the multiset M can be interpreted as representing a non-singleton set of configurations. In this respect, M extends the concept of configuration for elementary nets to Petri nets.

IV. PROPOSED ALGORITHM

Max-min algorithm allocates task Ti on the resource Rj where large tasks have highest priority rather than smaller tasks. For example, if we have one long task, the Max-min could execute many short tasks concurrently while executing large one. The total makespan, in this case is determined by the execution of long task. But if meta-tasks contains tasks have relatively different completion time and execution time, the makespan is not determined by one of submitted tasks. It would be similar to the Min-min makespan. For these cases, original Max-min algorithm losses some of its major advantages as load balance between available resources in small distributed system configuration and small total completion time for all submitted tasks in large scale distributed environment. We can't use the Max-min and wait submitted tasks to decide what would be the allocation map, makespan, load balance, etc. We try to minimize waiting time of short jobs through assigning large tasks to be executed by slower resources.

On the other hand execute small tasks concurrently on fastest resource to finish large number of tasks during finalizing at least one large task on slower resource. Based on these cases, where meta-tasks contains homogeneous tasks of their completion and execution time, we proposed a substantial improvement of Max-min algorithm that leads to increase of Max-min efficiency.

The algorithm calculates the expected completion time of the submitted tasks on each resource. Then the task with the overall maximum expected execution time is assigned to a resource that has the minimum overall completion time. Finally, this scheduled task is removed from meta-tasks and all calculated times are updated and the processing is repeated until all submitted tasks are executed. The algorithm focuses on minimizing the total makespan which is the total complete time in large distributed environment, for example, cloud computing environment also, executing tasks concurrently on available resources achieving load balance in small distributed system. The proposed algorithm produces mapping schema similar to RASA in such concurrency executing tasks and minimization of total completion time required to finish all tasks. Selecting task with maximum execution time leads to choose largest task should be executed. While selecting resource consuming minimum completion time means choosing slowest resource in the available resources. So allocation of the slowest resource to longest task allows availability of high speed resources for finishing other small tasks concurrently. Also, we achieve shortest makespan of submitted tasks on available resources beside concurrently. Not as original Max-min which recommended to be used if and only if submitted tasks is heterogeneous in their completion time and execution time, by means, there are clearly large tasks and small tasks.

Improved Max-min pseudo code is represented in Fig 3. We denotes the expected completion time matrix as Cij that is defined as rj, which represents ready time of resource Rj and Eij, that is Execution Time of task Ti on resource Rj. Fig 3 is a flowchart of proposed algorithm.

```
0.  Generate PetriNet based on  Gantt Charts
1.  for all submitted tasks in meta-task; Ti
2.  for all resources; Rj
3.   Cij = Eij + rj
4.  While meta-task is not empty
5.  find task Tk with maximum complete execution
    time.
6.  assign Tk to resource Rj with min. execution time.
7.  remove Tk from meta-tasks set
8.  update rj for selected Rj
9.  update Cij for all i
```

Fig 3: The Improved Max-Min Algorithm

Our algorithm derived from Max-min so that it has the same time complexity $O(mn^2)$, similar to original Max-min, Min-min and RASA where m is the number of resources and n is the number of tasks. Next section explains simple example to expose results.Step zero is responsible for generating PetriNet based on Gantt Charts. A similar process is achieved in generating the resource allocation graphs, which is commonly done in database systems especially for handling concurrency issues such as deadlock and inconsistency [12].

V.IMPLEMENTATION AND EXPERIMENTS

A. Illustrative Example

In order to illustrate our algorithm, assume we have four tasks T1, T2, T3 and T4 are in meta-tasks and scheduling manager has two resources R1 and R2 as problem set 2. Table 1, represents processing speed and bandwidth of communication links of each resource while Table 2, represents the volume of instructions and data in tasks T1 to T4. Using data given in Table 1 and Table 2, to calculate the expected completion time and execution time of the tasks on each of the resources.

Table 1. Resources Specification

Resource	Processing Speed (MIPS)	Bandwidth (MBBS)
R_1	150	300
R_2	300	15

Table 2. Meta-Tasks Specification

Task	Instruction Vol. (MI)	Data Vol. (MB)
T_1	256	88
T_2	35	31
T_3	327	96
T_4	210	590

Table 3 demonstrates calculated complete time of the tasks and execution time at the same time. On next step of the algorithm iteration, data in table 3 will be updated until all tasks are allocated. Fig 4 includes Gantt Charts representing the results of using original Max-min strategy on meta-tasks while Fig 5 includes two Gantt Charts representing the results of applying RASA and Improved Max-min, respectively. In Fig 4, the original Max-min achieves total makespan 9 seconds and uses only one resource R1. For next Fig, 5.a, RASA algorithm achieves total makespan 9 seconds, choose alternatively between large tasks and small tasks respectively because of number of resources is even [2] and uses just only one resource. Fig 5.b, describes Gantts Charts of our proposed scheduling algorithm which achieves makespan 8 seconds, introduces load balance between R1 and R2 and concurrency execution of tasks. Although the orders of the tasks scheduled in RASA and Improved Max-min is different, the makespan of each is at least equally if not smaller due to Improved Max-min. Based on experimental results, Improved Max-min algorithm produces mapping schema with better total makespan.

Table 3. Completion time of the tasks on each of the resources

Task / Resource	R_1	R_2
T_1	2.0	6.0
T_2	1.0	3.0

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

T_3	3.0	8.0
T_4	3.0	40.0

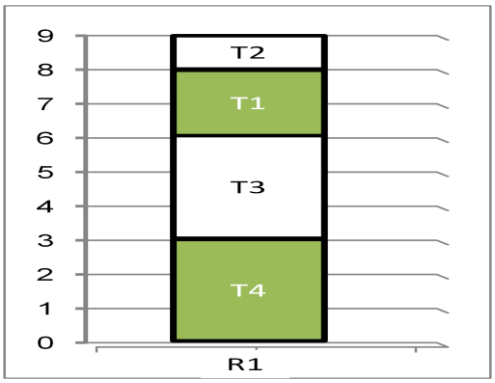
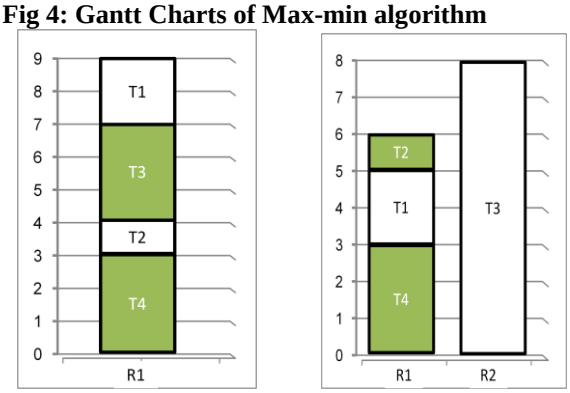


Fig 4: Gantt Charts of Max-min algorithm



a. RASA b. Imp. Max-min

Fig 5: Gantt Charts of Improved Max-min and RASA.

B. Evaluation of Experiments

The Improved algorithm is simulated using JAVA 6 Technology. Table 4 demonstrates different available resources of problem samples used for evaluation. Table 5 represents different submitted tasks in meta-tasks for each problem samples. We use data in table 4 and 5 to calculate makespan of each problem sample using different scheduling algorithms. Fig 6 is used to describe the problem samples and total time for completion; makespan using considered algorithms Max-min and Improved Max-min. While Fig 7 compares makespan of Min-min, Max-min, RASA and Improved Max-min as whole. We use data in Table 5 to construct Fig 6 and 7. It is obviously that the proposed algorithm schedules tasks with same makespan or less. Based on results, our proposed Improved Max–min produces the same total completion time or smaller than RASA and always smaller than original Max-min. Also, Improved Max-min scheduling presents concurrency execution of tasks using available resources and load balance in small distributed environment, cloud computing.

Table 4. Problem Samples Resources Specification

Problem Sample	Resource	MIPS	MBBS
P 1	R_1	50	100
	R_2	100	5
P 2	R_1	150	300
	R_2	300	15
P 3	R_1	300	300
	R_2	30	15

Table 5. Problem Samples Meta-Tasks Specification

Problem Sample	Task	MI	MB
P 1	T_1	128	44
	T_2	69	62
	T_3	218	94
	T_4	21	59
P 2	T_1	256	88
	T_2	35	31
	T_3	327	96
	T_4	210	590
P 3	T_1	20	88
	T_2	350	31
	T_3	207	100
	T_4	21	50

Table 6. Makespan of Problem Samples using algorithms

Problem Sample	Min-min	Max-min	RASA	Imp. Max-Min
P 1	11	11	10	10
P 2	9	9	9	8
P 3	5	5	5	4

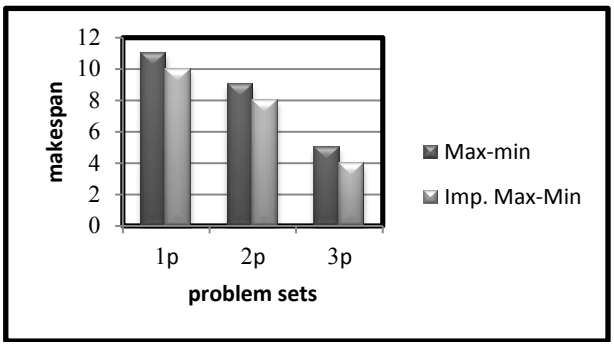


Fig 6: Comparison of makespan

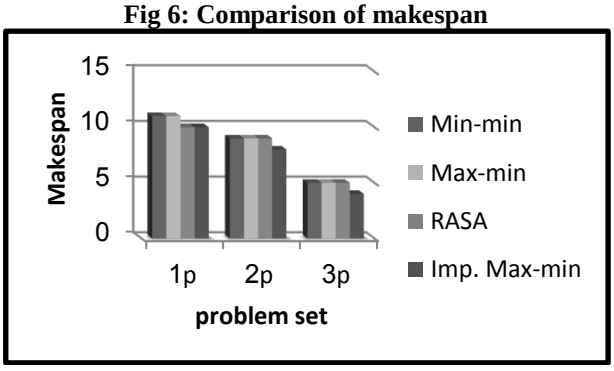


Fig 7: Comparison of makespan

VI. CONCLUSIONS AND FUTURE WORKS

Min-Min and Max-Min algorithms are common applicable in small scale distributed systems [21]. When the number of small tasks is more than number of the large tasks in a meta-task, the Max-min algorithm schedules tasks, in which the makespan of the system relatively depends on how many, executing small tasks concurrently with large one. If can't execute tasks concurrently, makespan become large. To overcome such limitations of Max-Min algorithm, a new modification is applied for Max-min scheduling algorithm. It uses the advantages of Max-Min and covers its disadvantages. We employ Petri nets which are well suited for modeling the concurrent behavior of distributed systems. This study is only concerned with the number of the resources and the tasks. The study can be further extended by applying the proposed algorithm on actual cloud computing environment and considering many other factors.

REFERENCES

1. Salim Bitam, "Bees life algorithms for job scheduling in cloud computing", International Conference on Computing and Information Technology, 2012.
2. Saeed Parsa and Reza Entezari-Maleki , "RASA: A New Grid Task Scheduling Algorithm", International Journal of Digital Content Technology and its Applications,Vol. 3, pp. 91-99, 2009.
3. I. Foster, and C. Kesselman, The Grid 2: Blueprint for a New Computing Infrastructure, Second Edition, Elsevier and Morgan Kaufmann Press, 2004.
4. L. Chunlin, et al, "QoS based resource scheduling by computational economy in computational grid," Journal of Information Processing Letters, Vol. 98, pp. 119-126, 2006.
5. X. He, X-He Sun, and G. V. Laszewski, "QoS Guided Min-min Heuristic for Grid Task Scheduling," Journal of Computer Sci. & Technology, Vol. 18, pp. 442-451, 2003.
6. L. Mohammad Khanli, and M. Analoui, "Resource Scheduling in Desktop Grid by Grid-JQA," The 3rd International Conference on Grid and Pervasive Computing, IEEE, 2008.
7. L. Mohammad Khanli, and M. Analoui, "Grid_JQA: A QoS Guided Scheduling Algorithm for Grid Computing," The Sixth International Symposium on Parallel and Distributed Computing (ISPDC'07), IEEE, 2007.
8. E. Elmroth, et al, "Grid resource brokering algorithms enabling advance reservations and resource selection based on performance predictions," J. of Future Generation Computer Systems, Vol. 24, pp.585-593, 2008.
9. B.T. Benjamin Khoo, B. Veeravalli, T. Hung, and C.W. Simon See, "A multi-dimensional scheduling scheme in a Grid computing environment," Journal of Parallel and Distributed Computing, Vol. 67, pp. 659-673, 2007.
10. B. Yagoubi, and Y. Slimani, "Task Load Balancing Strategy for Grid Computing," Journal of Computer Science, Vol. 3, No. 3, pp. 186-194, 2007.
11. M. Maheswaran, Sh. Ali, H. Jay Siegel, D. Hensgen, and R. F. Freund, "Dynamic Mapping of a Class of Independent Tasks onto Heterogeneous Computing Systems, Journal of Parallel and Distributed Computing, Vol. 59, pp. 107-131, 1999.
12. William Stallings, Operating Systems, 6th Ed. Chapter 6 - Concurrency: Deadlock and Starvation, Pearson Education International, 2008
13. T. D. Braun, H. Jay Siegel, N. Beck, L. L. Boloni, M.Maheswaran, A. I. Reuther, J. P. Robertson, M. D.Theys, and B. Yao, "A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems, "Journal of Parallel and Distributed Computing, Vol. 61, pp. 810-837, 2001.
14. F. Dong, J. Luo, L. Gao, and L. Ge, "A Grid Task Scheduling Algorithm Based on QoS Priority Grouping," In the Proceedings of the Fifth International Conference on Grid and Cooperative Computing (GCC'06), IEEE, 2006.
15. E. Ullah Munir, J. Li, and Sh. Shi, 2007. QoS Sufferage Heuristic for Independent Task Scheduling in Grid. Information Technology Journal, 6 (8): 1166-1170.
16. K. Etminani, and M. Naghibzadeh, "A Min-min Max-min Selective Algorithm for Grid Task Scheduling,"The Third IEEE/IFIP International Conference on Internet, Uzbekistan, 2007.
17. A. Afzal, A. Stephen McGough, and J. Darlington, "Capacity planning and scheduling in Grid computing environment," Journal of Future Generation Computer Systems, Vol. 24, pp. 404-414, 2008.
18. P. Brucker, Scheduling Algorithms, Fifth Edition, Springer Press, 2007.
19. R. Buyya, and M. Murshed, "GridSim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing," Journal of Concurrency and Computation Practice and Experience, pp 1175–1220, 2002.
20. D.I. George Amalarethinam and P. Muthulakshmi, "An Overview of the scheduling policies and algorithms in Grid Computing ", International Journal of Research and Reviews in Computer Science, Vol. 2, No. 2, pp. 280-294, 2011.
21. T. Kokilavani and Dr. D.I. George Amalarethinam, "Load Balanced Min-Min Algorithm for Static Meta-Task Scheduling in Grid Computing", International Journal of Computer Applications, Vol. 20, No. 2, pp. 43-49, 2011.
22. G. Rozenburg, J. Engelfriet, Elementary Net Systems, in: W. Reisig, G. Rozenberg (Eds.), Lectures on Petri Nets I: Basic Models - Advances in Petri Nets, volume 1491 of Lecture Notes in Computer Science, Springer,1998, pp. 12-121