Multiphase Scalable Grid Scheduler Based on Multi-QoS Using Min-Min Heuristic

Nawfal A. Mehdi, Ali Mamat, Hamidah Ibrahim, Shamala A/P K
Faculty of Computer Science and Information Technology
University Putra Malaysia,
Serdang, 43400,Selangor, Malaysia

Abstract—**In scheduling, the main factor that affects searching speed and mapping performance is the number of resources or the size of search space. In grid computing, the scheduler performance plays an essential role in the overall performance. So, it is obvious the need for scalable scheduler that can manage the growing in resources (i.e. scalable). With the assumption that each resource has its own specifications and each job has its own requirements; then searching the whole search space (all the resources) can waste plenty of scheduling time. In this paper, we propose a two-phase scheduler that uses min-min algorithm to speed up the mapping time with almost the same efficiency. The scheduler is also based on the assumption that the resources in grid computing can be classified into clusters. The scheduler tries first to schedule the jobs to the suitable cluster (i.e. first phase) and then each cluster schedule the incoming jobs to the suitable resources (i.e. second phase). The scheduler is based on multidimensional QoS to enhance the mapping as much as it can. The simulation results show that the use of two-phase strategy can support the scalable scheduler.**

Keywords- Multi-phase; QoS; Grid Scheduling.

I. Introduction

With the development of the network technology, grid computing used to solve larger scale complex problems becomes a focus technology. The goal of schedulers is to utilize all available computational resources to overcome difficulties brought about by complicated tasks with enormous computing workloads.[1]

One of the nearest grid definition to our work is given by Ian Foster [2] "*The real and specific problem that underlies the Grid concept is coordinated resource sharing and problem solving in dynamic, multi-institutional Virtual Organizations (VO)*"[2]. We can conclude from Foster's definition: although the Grid has the characteristics of heterogeneity and dynamicity, these features are not flatly distributed in resources, but are rather distributed hierarchically and locally in many cases, due to the composition of the Grid resources. Current Grid resources are usually distributed in a clustered fashion[3]. The key technologies that affect the Grid efficiency involve Grid resource allocation, management and task scheduling algorithm.

Task scheduling is a challenging problem in grid computing environment [4] and has shown to be NP-complete in its general as well as in some restricted forms[5]. According to [6], a valid schedule is the assignment of tasks to specific time intervals of resources, such that no two tasks use any resource simultaneously, or such that the capacity of the resource is not exceeded by the tasks. The schedule of tasks is optimal if it minimizes a given optimality criterion (objective function).

Grid scheduler (GS) receives applications from grid users, selects feasible resources for these applications according to the acquired information from the Grid Information Service module, and finally generates application-to-resource mappings based on certain objective functions and predicted resource performance. Unlike their counterparts in traditional parallel and distributed systems, Grid schedulers usually cannot control Grid resources directly, but they work like brokers or agents[7]. One of the most issues in grid scheduling is the QoS; the quality of services (QoS) becomes a big concern of many Grid applications in such a non-dedicated dynamic environment. The meaning of QoS is highly dependent on particular applications, from hardware capacity to software existence. Usually, QoS is a constraint imposed on the scheduling process instead of the final objective function.[3]

This paper addresses the problem of resources growing in one search space and the ability of the main scheduler to control this growing by two phase mapping. The work in this paper is concerned with scheduling computing intensive independent task; each task requires multi QoS specification. Each task should be mapped to a cluster that can fulfill its requirement with a minimum completion time.

This work introduces the ability to schedule the tasks to a cluster to be scheduled later by the cluster's local scheduler. The main scheduler should have full information about the clusters starting from number of resources in each one to the common characteristics of the resources. Also, the main scheduler receives a set of tasks from the clients each one (i.e. task) with its QoS constraints to be mapped to the best fit cluster that can give the minimum execution time with the respect to its restrictions.

The remainder of this paper is organized as follows: in the next section II, we provide the related works. Section III, introduces task problem modeling and the new algorithm and its time complexity analysis. Section IV, shows the implementation and experiments results. Recommendations and future plan are given in section V.

II. Related Works

Over the years, task scheduling problem has become a well-recognized discipline in Grid computing and is identified as NP complete problem[8]. Many scheduling heuristics have been proposed to solve the mapping process in grid computing. Min-min heuristic depends on the minimum completion time, such that the task that has the minimum completion time is executed first. X. He *et al*.[9], proposed a QoS Guided Min-Min heuristic which can guarantee the QoS requirements of particular tasks and minimize the makespan at the same time. Wu, Shu and Zhang[10], proposed an algorithm that ordered tasks list by completion time, then segmenting the ordered list to be applied in Min-Min algorithm. They show in their results that, the algorithm can outperform the typical Min-Min. Another popular heuristic for independent scheduling is called Suffrage. The rationale behind Suffrage is that a task should be assigned to a certain host and if it does not go to that host, it will suffer the most. This algorithm has been studied by Maheswaran *et al* [11]. Muthuvelu *et al* [12] proposed a dynamic task grouping scheduling algorithm to deal with these cases. Once a set of fine grained tasks are received, the scheduler groups them according to their requirements for computation (measured in number of instructions) and the processing capability that a grid resource can provide in a certain time period. All tasks at same group are submitted to the same resource which can finish them all in the given time. Hence, the overhead for scheduling and job launching is reduced and resource utilization is increased. S´ebastien Noel *et al*[13], studied the use of a framework called YML for developing HPC applications on Grids, and proposed a multi-level scheduling architecture for it. K. Etminani and M. Naghibzadeh introduced a new scheduling algorithm based on two conventional scheduling algorithms, Min-Min and Max-Min, to use their cons and at the same time, cover their pros. It selects between the two algorithms based on the standard deviation of the expected completion time of tasks on resources. They evaluated their scheduling heuristic, the selective algorithm, within a grid simulator called GridSim. They also compared their approach to its two basic heuristics. F. M. Ciorba *et al* [15], studied the problem of scheduling loops with iteration dependencies for heterogeneous (dedicated and non-dedicated) clusters. The presence of iteration dependencies incurs an extra degree of difficulty and makes the development of such schemes quite a challenge. They extended three well known dynamic schemes (CSS, TSS and DTSS) by introducing synchronization points at certain intervals so that processors compute in pipelined fashion. Their scheme is called Dynamic Multi-Phase Scheduling (DMPS) and they applied it to loops with iteration dependencies. They implemented their new scheme on a network of heterogeneous computers and studied its performance. Through extensive testing on two real-life applications (the heat equation and the Floyd-Steinberg algorithm), they showed that the proposed method is efficient for parallelizing nested loops with dependencies on heterogeneous systems.

III. Task Scheduling Problem

This work is based on scheduling the tasks in two phases to reduce the search space for the scheduler. The proposed algorithm should already have the set of clusters that is available at that time. Each cluster should come with its specifications that is used to fit with user's QoS restrictions. Also, the algorithm takes a set of tasks, each one with its QoS restrictions.

A. *Problem Modeling*

We model the scheduling problem by $E_i=(J_i,C_j)$, where J_i is a job, C_j is a cluster and E_i is the mapping.

Jobs are defined in this work as:

- J is the set of M jobs such that $J=\{J_1,J=,\ldots,J_M\}$. Each job J_i has four QoS characteristics that are described in details in the next point.
- Q is the set of QoS dimensions that is attached with each job J_i such that $Q=\{L_i, S_i, SE_i, BW_i\}$,where
  - ➢ L_i is the length of the job J_i.
  - ➢ S_i is the maximum cost that can be paid by job J_i.
  - ➢ SF_i is the security value that represents the amount of security needed by J_i.
  - ➢ BW_i is the amount of network bandwidth that is needed by J_i.

Clusters are defined in this work as:

- C is the set of N clusters such that $C=\{C_1,C_2,\ldots.,C_N\}$. Each cluster C_i has four properties.
- P is the set of five properties attached with each cluster C_i such that, $P_i=\{SP_i,CB_i,CC_i,CS_i,Z_i\}$ where:
  - ➢ SP_j is the speed of cluster C_j.
  - ➢ CB_j is the bandwidth offered by cluster C_j.
  - ➢ CC_j is the cost/hour offered by cluster C_j.
  - ➢ CS_j is the security value that represents the amount of security offered by C_j.
  - ➢ Z_j is the size of cluster C_i (i.e. the number of resources)
- R_j is a set of size Z_j represent the resources' ready time for cluster C_j.

To model the servers in our work, we suppose:-

- RS is the set of W resources such that $RS=\{RS_1,RS_2,\ldots.,RS_W\}$. Each resource RS_i has four properties.
- PS is the set of five properties attached with each resource RS_i such that, $PS_i=\{SP_i,RN_i,RC_i,RS_i,RD_i\}$ where:
  - ➢ RSP_j is the speed of Resource RS_i.

- *RB_i* is the bandwidth offered by Resource *RS_i*.
- *RC_i* is the cost/hour offered by Resource *RS_i*.
- *RS_i* is the security flag that is set if the Resource *RS_i* offered security.
- *RD_i* is the ready time for resource *RS_i*.

In the cluster's class there is an *R_j* field that is responsible for holding the ready time for each host inside the cluster. This list should be always in ascending order to facilitate selecting the best cluster. Initialed to zero, this list is firstly. Each job's class has two fields, first one (*TCT*) is responsible for holding the best completion time offered by a cluster that its address is held in the second field cluster index (*Clr_ndx*).

In this work, *ET_{ij}* represents the expected completion time of task *J_i* on a host in cluster *C_j*. First(*R_j*) represents the best ready time for cluster *C_j*. *CE_{ij}* represents the expected completion time of task *J_i* on a host in cluster *C_j*. *EC_{ij}* represents the expected cost to execute job *J_i* in cluster *C_j*.

```
1.  While (J is not empty) do
2.    For each job $J_i$ in J do
3.      $J_i$.TCT=Double.Max_value    //TCT= Temporary Completion Time
4.        For each cluster $C_j$ do
5.          $ET_{ij}=L_i/SP_j$
6.          $CT_{ij}=ET_{ij}$+first($R_i$)
7.          $EC_{ij}=(L_i /SP_i)* CC_i$
8.          If ($J_i$.TCT>$CT_{ij}$) and ($J_i$.SF=$C_j$.CS) and ($J_i$.S<= $EC_{ij}$)
            and($J_i$.BW<=$C_j$.CB) then
9.            $J_i$.TCT= $CT_{ij}$
10.           $J_i$.Clr_ndx=j
11.         End if
12.       End For
13.       If (there is no match) then
14.         Print out Job $J_i$ has no match
15.         Delete $J_i$ from J
16.       End if
17.       Else
18.         If ($J_i$ has minimum Completion time) then
19.           Set Min_Clstr=j
20.           Set Min_Job=i
21.         End if
22.   End For
23.   Map $J_{Min\_job}$ to $C_{Min\_Clstr}$
24.   Delete $J_{Min\_job}$ from J
25.   Update $R_{Min\_Clstr}$ such that the set should stay sorted in ascending
      order.
26. End While
```

Figure 1. Global grid Scheduler Algorithm

This algorithm computes the expected completion time for all tasks on all clusters using these equations:

$$ET_{ij} = SP_j / L_i \qquad (1)$$

$$CT_{ij} = ET_{ij} + first(R_j) \qquad (2)$$

Then it computes the expected cost using (3):

$$EC_{ij} = ET_{ij} * CC_j \qquad (3)$$

This algorithm has loop *J_i* (line 4..line 12) that finds the best cluster that fulfill *J_i* QoS constraints and has the minimum completion time by using equations (1,2,3). After the loop *J_i*, an If condition (line 13) checks if the *J_i* got any host that can fulfill its constraints. If there is no such a host then delete this job (*J_i*) from the job list, otherwise check again (line 18) if this *J_i* has the minimum execution time and save its index if true. At line 23 we have *J_{Min_job}* that holds the index for the minimum completion time job, so we map it to its cluster *C_{Min_Clstr}*. Line 25 is responsible for updating the list *R* in such a way it stays in ascending order.

B. *Algorithm Analysis*

The time complexity of the proposed algorithm is:

$$f(n,m) = O(m \log m) \cdot O(n) \qquad (4)$$

Where m is the number of jobs and n is the number of clusters. From above, we can see that this algorithm has a little effect by the increase in the number of servers inside the clusters because updating servers list required just *log(Zj)* where *Zj* is the number of servers inside cluster *Cj*.

In comparison with this algorithm, the time complexity for the old algorithm is:

$$f(n,m) = O(m \log m) \cdot O(w) \qquad (5)$$

Where *w* is the number of servers in the cluster.
Therefore, it is quite clear the effect of increasing the number of servers on the proposed algorithm is not that much intense.

C. *Quality of Service (QoS)*

This work uses QoS restriction to find the suitable cluster that can execute user's tasks. Multi-dimensions QoS have been used so that the users should submit their tasks with many parameters. These parameters are:

- **Bandwidth**: The user should submit his task with the minimum amount of bandwidth needed to execute it. Bandwidth is set to zero in case it does not need any bandwidth.
- **Security**: These days, the most important issue in distributed system is the security and its type. In this work, we proposed a multi-type QoS security check. It means the algorithm can check for the user the suitable type that he needs to execute the task. Security parameter is an integer value, where each value represents a type or level of security.
- **Cost**: Budget cost is the amount of payment from a user to a resource for its service. Here the user should specify the maximum cost, which can be afforded.

IV. Implementation and Experiment Results

This algorithm is used in the first or higher level, while in the second level, the normal MM is used.

We use Java programming language in order to implement the simulator to test the proposed algorithm. The implementation consists of several classes, these are:

- **Create population**: This class is responsible for creating the set of tasks with its QoS restrictions, set of clusters with their specifications and a set of servers to be used as for old algorithm. The size of set of servers is equal to the number of clusters multiplied by the number of servers in each cluster. The number of tasks, clusters and servers/cluster are fixed, and the QoS restrictions and the clusters specifications are generated randomly.
- **New Min-Min**: this class is an implementation for the improved min-min that (2PMM) is responsible for mapping the tasks to the appropriate cluster.
- **Old Min-Min**: this class is an implementation for the old algorithm to be used for performance comparison.

Firstly, Create population class generates 1000 tasks in one list, N clusters each one with Z servers and list of W servers such that:

$$W = N * Z \qquad (6)$$

Secondly, *Old Min-Min* and *New Min-Min* start working to make the mapping and compute the performance metrics which is the makespan. Makespan can be define as the time spent from the beginning of the first job to the end of the last job.

Two experiments have been made to test the performance of 2PMM algorithm. Each experiment consists of six sizes (i.e. number of clusters and servers). The test for each size is made ten times and the average has been taken for the comparison.

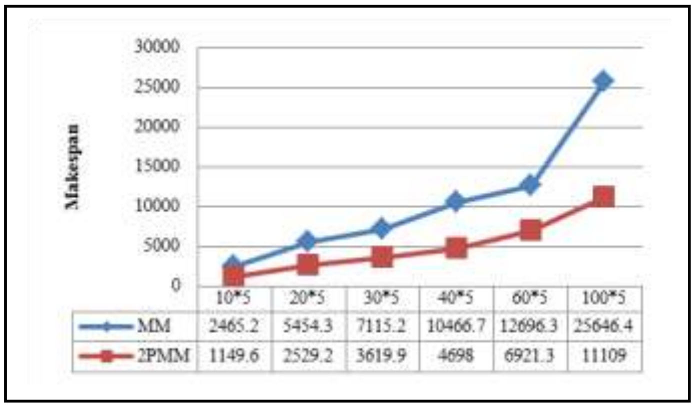
The first experiment compares the performance and cost in both old and new algorithms (figure 2). This figure shows the effect of increasing of servers on mapping time. The Y-axis in this figure represents the total execution time for the mapping process, while the x-axis represents the number of servers and it is written in form of equation (6)(i.e. 10*5=50 means, 10 clusters and 5 servers in each cluster as a test bed for the new algorithm and 50 servers as a test bed for the old algorithm). In this experiment, we fixed the number of tasks to 1000 and the number of clusters to 10 and changed the number of servers in each cluster. It is quite clear that the effect of increasing the number of servers (i.e. increasing the search space) on the execution time of the scheduler is not that much intense. Figure (4) shows the improvement mapping time between 2PMM and MM algorithms.

The second experiment (figure 3) shows the influence of increasing the number clusters on the new algorithm. As in figure (2), the y-axis represents the mapping execution time while the x-axis represents the number of clusters, number of servers for each cluster and total number of servers. We can see that the total execution time is directly affected by the number of clusters in its search space but its time is still far from the time needed in the old algorithm.

V. Conclusion and Future Work

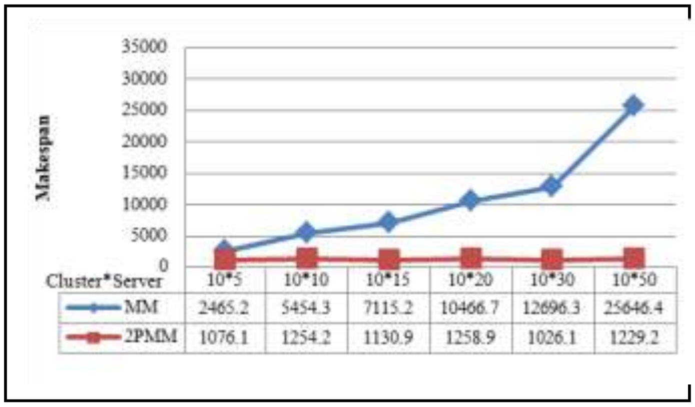
This paper investigates the job scheduling algorithm in grid environments as an optimization problem. The proposal is to minimize the scheduling time for urgent jobs, by mapping the jobs to the best cluster as the first phase and then reschedule to the best resource in the selected cluster.

The algorithm is developed based on Min-Min Algorithm to find the proper cluster that can execute the job with minimum execution time with respect to QoS job requirements. The improved algorithm is compared with the previous Min-Min algorithm. The results show a better performance in scheduling time point of view. It can map the jobs faster than the normal Min-Min. The future work will focus on clustering algorithms and study the effect of three phase clustering on the system.



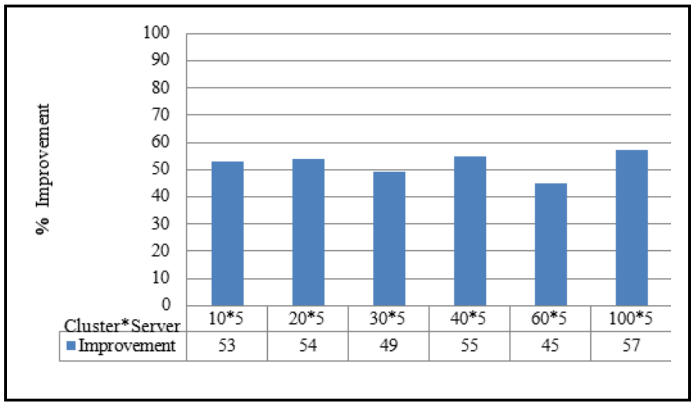
	10*5	20*5	30*5	40*5	60*5	100*5
MM	2465.2	5454.3	7115.2	10466.7	12696.3	25646.4
2PMM	1149.6	2529.2	3619.9	4698	6921.3	11109

Figure 2. The effect of increasing the number of clusters with fixed number of servers on makespan



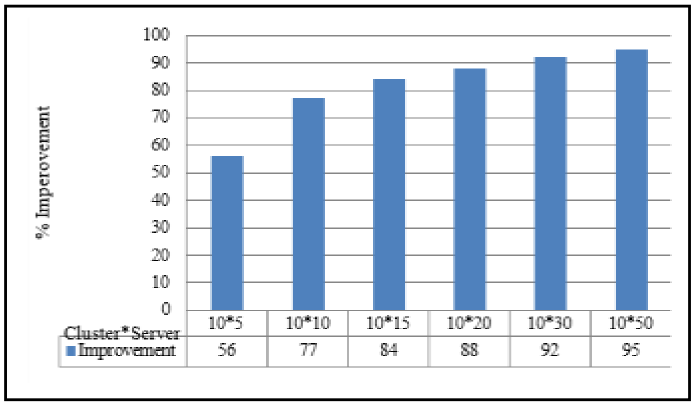
Cluster*Server	10*5	10*10	10*15	10*20	10*30	10*50
MM	2465.2	5454.3	7115.2	10466.7	12696.3	25646.4
2PMM	1076.1	1254.2	1130.9	1258.9	1026.1	1229.2

Figure 3. The effect of increasing the number of servers with fixed number of clusters on Makespan.



Cluster*Server	10*5	20*5	30*5	40*5	60*5	100*5
■ Improvement	53	54	49	55	45	57

Figure 4. The percentage of makespan improvement when increasing the number of clusters with fixed number of servers.



Cluster*Server	10*5	10*10	10*15	10*20	10*30	10*50
■ Improvement	56	77	84	88	92	95

Figure 5. The percentage of makespan improvement when increasing the number of servers with fixed number of clusters

REFERENCES

[1] Ehsan Ullah Munir, Jianzhong Li, and Shengfei Shi, "QoS Sufferage Heuristic for Independent Task Scheduling in Grid," Information Technology, vol. 6, no. 7. pp.1166-1179, 2007.
[2] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," International Journal of High Performance Computing Applications, vol. 15, no. 3. pp.200, 2001.
[3] F. Dong and S. G. Akl, "Scheduling Algorithms for Grid Computing: State of the Art and Open Problems," Queen's University School of Computing.January, 2006.
[4] I. Foster and C. Kesselman, The Grid: Blueprint for a New Computing Infrastructure: Morgan Kaufmann, 2004.
[5] H. El-Rewini, T. G. Lewis, and H. H. Ali, Task scheduling in parallel and distributed systems, 1994.
[6] P. Fibich, L. Matyska, and H. Rudovb, "Model of Grid Scheduling Problem," Exploring Planning and Scheduling for Web Services, Grid and Autonomic Computing. pp.05-03
[7] F. Berman, R. Wolski, H. Casanova et al., "Adaptive computing on the Grid using AppLeS," Parallel and Distributed Systems, IEEE Transactions on, vol. 14, no. 4. pp.369-382, 2003.
[8] O. Sinnen and I. NetLibrary,Task Scheduling for Parallel Systems: Wiley-Interscience, 2007.
[9] H. E. XiaoShan, S. U. N. XianHe, and G. von Laszewski, "QoS Guided Min-Min Heuristic for Grid Task Scheduling," Journal of Computer Science and Technology, vol. 18, no. 4, 2003.
[10] M. Y. Wu, W. Shu, and H. Zhang, "Segmented min-min: A static mapping algorithm for meta-tasks on heterogeneous computing systems." 9th IEEE Heterogeneous Computing Workshop (HCW 2000) , pp. 375-385. 2000.
[11] M. Maheswaran, S. Ali, H. J. Siegel et al., "Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems." 8th Heterogeneous Computing Workshop (HCWÆ99) , pp. 30-44. 1999.
[12] N. Muthuvelu, J. Liu, N. L. Soe et al., "A dynamic job grouping-based scheduling for deploying applications with fine-grained tasks on global grids." Proceedings of the 2005 Australasian workshop on Grid computing and e-research-Volume 44 , pp. 41-48. 2005. Australian Computer Society, Inc. Darlinghurst, Australia, Australia.
[13] S. Noel, O. Delannoy, N. Emad et al., "A Multi-level Scheduler for the Grid Computing YML Framework," LECTURE NOTES IN COMPUTER SCIENCE, vol. 4375. pp.87, 2007.
[14] K. Etminani and M. Naghibzadeh, "A Min-Min Max-Min selective algorihtm for grid task scheduling." Internet, 2007.ICI 2007.3rd IEEE/IFIP International Conference in Central Asia on , pp. 1-7. 2007.
[15] F. M. Ciorba, T. Andronikos, I. Riakiotakis et al., "Dynamic Multi Phase Scheduling for Heterogeneous Clusters." Proc.of the 20th IEEE IntÆl Par.& Dist.Proc.Symp.(IPDPSÆ06), Greece . 2006.