Scheduling Tasks in Grid Computing Environments

A. Kadam[1], **V. Thool**[2]

[1]Department of Computer Engineering, All India Shri Shivaji Memorial College of Engineering, Pune

[2]Department of Instrumentation Engineering, Shri Guru Gobind Singhji Institute of Engineering and Technology, Nanded.
{kadam_in@yahoo.com, vrthool@yahoo.com}

Abstract: Scheduling tasks on different resources spread over a grid computing system is an NP complete problem. Assorted researchers are developing adaptation scheduling algorithms for getting optimality. Hence they have proved a good result for tasks scheduling about resources selection. In this review, we study the Group-based Parallel Multi-scheduler (GPMS). It is focused on effectively utilizing the advantages of multicore systems for Grid scheduling. Two job grouping methods are implemented, viz., Execution Time Balanced and Execution Time Sorted then Balanced termed as ETB, ETSB respectively. Two machine grouping methods; Evenly Distributed (EvenDist) and Similar Together (Sim Tog). We also see the MinMin Grid scheduling algorithm. We demonstrated that by assigning different tasks to the machines into batch before scheduling, the computation time for the scheduling process improves by 85% over the ordinary MinMin algorithm. We also study a new heuristic algorithm called Sort-Mid. It strives for maximizing the utilization and minimizing the time period.

Keywords—*Grid Computing, Scheduling, Parallelism.*

1 INTRODUCTION

Grid computing systems are distributed systems, enable large-scale resource sharing among millions of computer systems across a worldwide network such as the Internet. Grid resources are different from resources in conventional distributed computing systems by their dynamism, heterogeneity, and geographic distribution. The organization of the grid infrastructure consists of four levels as First: the foundation level, it includes the physical components. Second: the middleware level, it is the software responsible for task execution, task scheduling, resource management, and security. Third: the services level, it provides vendors/users with efficient services. Fourth: the application level, the application level contains the services such as business tools and operational utilities.

The scheduling has become one of the major research objectives, since it directly influences the performance of grid applications. Task scheduling [3] is the key step of grid resource management. It governs job to assign suitable resources by apply polices and scheduling algorithms. In static scheduling, all resources related data and all the tasks are assumed to be known in advance, application is scheduled. Moreover, each task is allocated once to a resource. Unlike static scheduling, in dynamic scheduling, the task assigning is done on the go as the application executes, but it is not possible to note the execution time. Tasks are coming dynamically scheduler has to work more in decision making to allocate resources. The advantage of the dynamic over the static scheduling is that the system does not require to posse the run time behavior of the application in prior it runs.

Multicore technology has come to stay and as Grid computing continues to grow, it will be worthwhile to scale Grid scheduling to benefit from the multicore technology. Multicore systems offer opportunity for parallelism and increased throughput. Parallelism takes programming away from the traditional serial execution approach by employing several processors to simultaneously execute independent tasks and is best suited for independent jobs which characterize a large percentage of users' jobs on the Grid.

B. Iyer, S. Nalbalwar and R.Pawade(Eds.)
ICCASP/ICMMD-2016. Advances in Intelligent Systems Research.
Vol. 137, Pp. 342-348.

Increased throughput is a direct increase in output over a set period resulting from more efficient processing. Current Grid scheduling algorithms do not utilize the advantages built in the underlying multicore systems, mostly focusing on parallel execution of jobs instead of parallelizing the scheduling function. Neglecting the underlying multicore hardware in the scheduling algorithm of the Grid will cause an unnecessary bottleneck in processing.

1.1 RELATED WORK

Reda et. Al. [1] present a new heuristic algorithm, Sort-Mid to maximize utilization and find appropriate resources in Grid. In [2], the authors propose a Particle Swarm Optimization (PSO) based scheduling algorithm which aims at optimization via rule generation to implement classifier system. Garg et. Al. presents a task scheduler which provides needful resources to the tasks according to the dynamic workflow. It also performs rescheduling to provide minimum execution time [3]. In [4], the authors discuss the implementation of Grid and Smart Grid applications over the Cloud environment. Some of the applications include Cyber Physical System (CPS) and Dynamic Internet Data Centers (IDCs). Finally in [5], the authors propose the Group-based Parallel Multi-scheduler (GPMS), which points effectively utilizing the advantages of multicore systems for Grid scheduling. It divides jobs and machines into paired groups and separately scheduling jobs in parallel from same groups.

1.2 Algorithms Used

1.2.1 Group-based Parallel Multi-scheduler for Grid (GPMS) [5]

The GPMS for Grid aims at exploring parallelism on multicore systems to raise scheduling algorithms inside Grid. To achieve this we assume that multicores are pervasive and constitute major part of Grid machines. We also assume that our scheduler runs on a multicore system.
The GPMS requires jobs to be split into groups. Two methods employed to achieve this are:
ETB: Execution Time Balanced —Estimate execution time and then balance across groups.
ETSB: Execution Time Sorted and Balanced —Estimate execution time, then sort jobs and then balance across groups.

A High level algorithm for GPMS is given below.

Step 1: Start
Step 2: Define number of threads
Step 3: Define number of groups to use
Step 4: Read jobs into the scheduler
Step 5: Estimate the execution time for each job from the job attributes,
Step 6: Group jobs using a chosen grouping method into number of specified groups
Step 7: Read machines and group them into the specified number of groups using a chosen grouping method
Step 8: Execute the scheduling algorithms within the groups
Step 9: Write results to output file
Step 10: Stop

1.2.2 Sort-Mid algorithm [1]

Scheduling is the main step of grid machines management. Machines may be homogeneous or heterogeneous. A grid scheduler choose the best machine to a particular job and submits that job to the selected machine[18]. The main aim of this heuristic algorithm for scheduling a group of pieces on a computational grid system is to maximize the machine use and to minimize the time period. Given a grid G and a restricted number of machines (resources) M; $M_1, M_2,...,M_m$, m> 1.
Let a finite nonempty set of n tasks be the T; $T_1, T_2,...,T_n$, n> 1 that needs to be executed in G. The steps to assign each task to a suitable machine are summarized below. It uses assignment function

$$\mathbf{S = T \rightarrow G}$$

which is described as follows. I≤ n and j ≤ m for every positive integer; such that $S(T_i)=M_j$. The first step is SCT: to sort the completion times of each task Ti in Tin increasing order. This scheduling decision is based on computing the average value AV of two consecutive completion times in SCT for each T_i. AV is computed by

$$\mathbf{(SCT_K+SCT_{K+1})/2}$$

where κ=[m/2]. In the second step, the task having the maximum AV is selected. In the third step, the task is allocated to the machine possessing minimum completion time. Next, the assigned task is deleted from T. Eventually; the waiting time of the machine that executes this task is updated. These steps are repeated until all **n** tasks are scheduled on **m** machines.

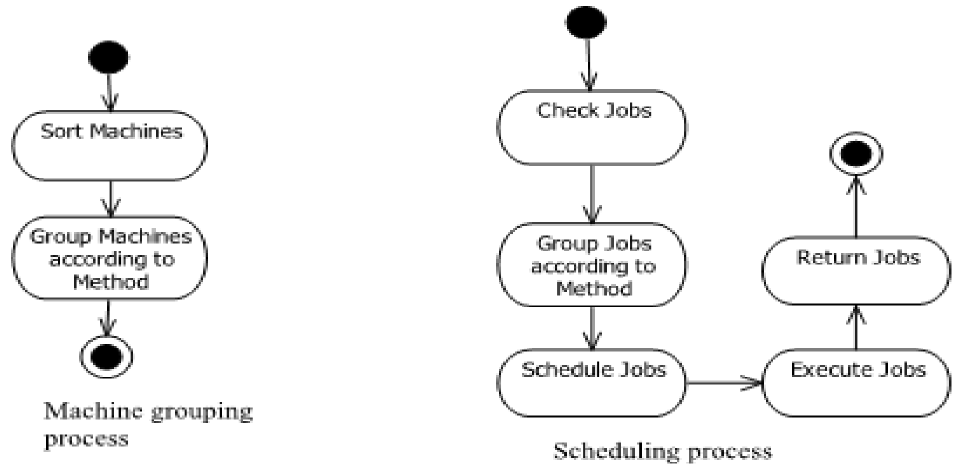

Fig. 1. Activity diagram showing steps of the GPMS.

1.2.3 Adaptive workflow scheduling (AWS) algorithm [3]

Proposed AWS has three phases: (i) Resource discovery and monitoring phase, (ii) static task scheduling and (iii) Rescheduling phase. The flow diagram for the algorithm is as given in Fig. 2.

1.2.4 Cloud Computing for Smart Grid applications [4]

In this paper, authors discussed various possibilities to apply CC towards implementation of SG applications. Although a specific algorithm for a particular application has not been mentioned, the authors have proposed many application, challenges and scenarios that can be studied for research thesis and so on.

1.2.5 Knowledge Acquisition with Rules as Particles (KARP)[2]

In this paper, the authors propose KARP, a novel strategy for classifier discovery systems for the generation of high quality fuzzy rules in Fuzzy Classifier Systems(s).

KARP operates in the following stages:

1. Create a RB made up of NP particles determined by its position to become the individuals of the swarm. Definition of particles position in the integer space.
2. Assign each particle a velocity that handles the modification of each rule in every step. Definition of velocity of a particle in the integer space.
3. Recognize the best particle discovered by every individual and the whole swarm. Quality of a particle.

4. Update the velocity of each particle with respect to their own and social experience.
5. Update location of every particle in the search space.
6. The detailed algorithm for implementation can be followed in [2].

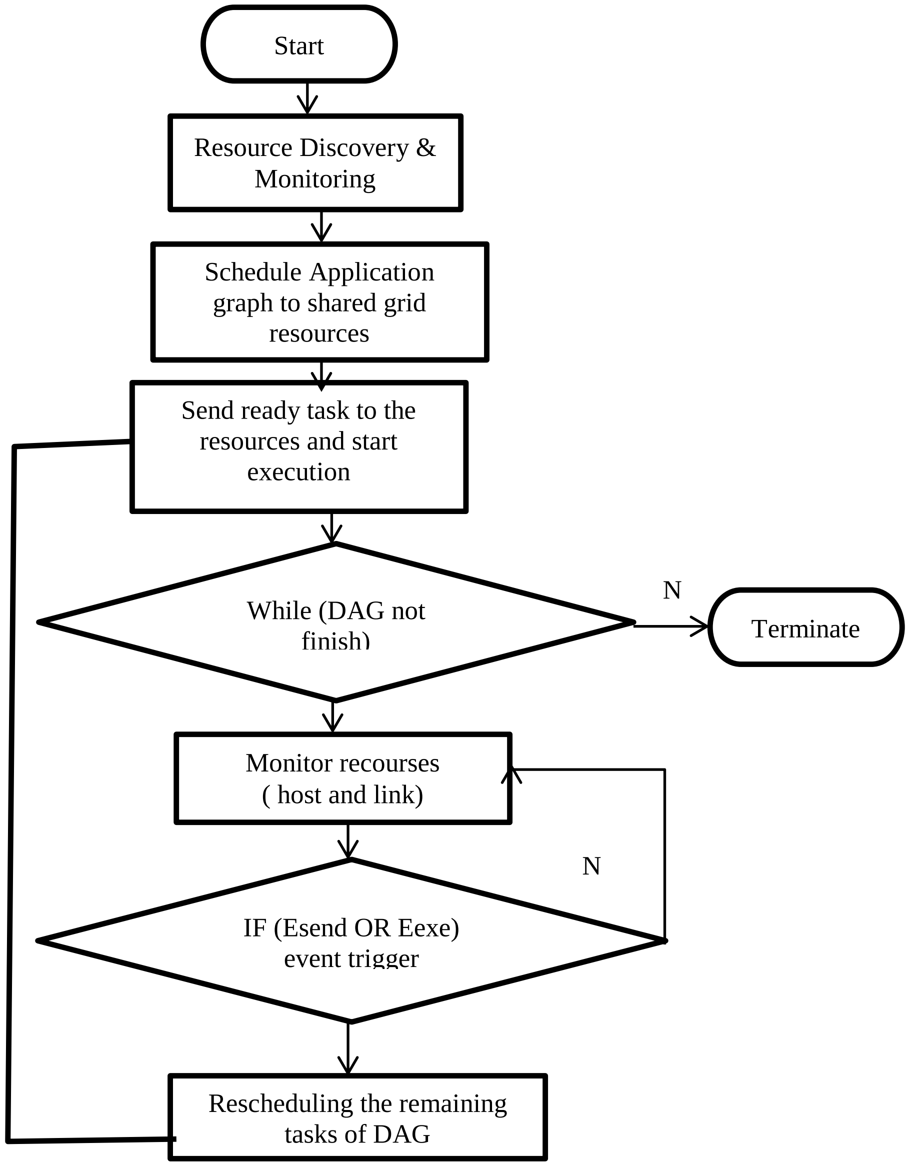
Start
Resource Discovery & Monitoring
Schedule Application graph to shared grid resources
Send ready task to the resources and start execution
While (DAG not finish)
N
Terminate
Monitor recourses (host and link)
N
IF (Esend OR Eexe) event trigger
Rescheduling the remaining tasks of DAG

Fig. 2. The flow diagram of the procedure for AWS

2 Results

2.1 Group-based Parallel Multi-scheduler for Grid (GPMS)

With four Grid sites author simulated a Grid environment which consisting of machines with different CPU speed and number of processors. Our scheduling aims directly at the CPUs on the individual machines. Jobs are scheduled to CPUs.
With the Priority job grouping method, the SimTog and EvenDist methods recorded a performance improvement of 5.90 and 6.76 over the MinMin, with times of 41 006 and 35 807 ms respectively.
With the ETSB grouping method, the SimTog and EvenDist methods recorded a 13 times and 52 times performance improvement over the MinMin using 17 569 and 4643 ms respectively, while the ETB job grouping method yielded 46 and 51 times performance improvement between the SimTog and EvenDist methods respectively using 5224 and 4701 ms respectively to perform the scheduling task. The ANOVA results which show the significance of the performance differences are given in table below.

Table 1. ANOVA result for Priority and all grouping methods

Test	Method	P value	Significance difference?
1	Priority vs GPMS (ETB and ETSB averaged)	0.027992	Yes
2	Priority vs. ETB	0.015965	Yes
3	Priority vs. ETSB	0.048583	Marginal
4	Priority-EvenDist vs. ETB- EvenDist	0.020335	Yes
5	Priority-SimTog vs. ETB- SimTog	0.013124	Yes
6	Priority-EvenDist vs. ETSB- EvenDist	0.020128	Yes
7	Priority-SimTog vs. ETSB- SimTog	0.109315	No

2.2 Sort-Mid algorithm

The following table shows the results obtained by the authors as compared to other competitive algorithms. Detailed experimentation results are available in [1].

Table 2. Computational Complexity

Algorithm	Complexity
MET	O(nm)
OLB	O(nm)
Mact-Min	O(nm)
Max-Min	$O(n_2m)$
Min-Mean	$O(n_2m)$
Min-Min	$O(n_2m)$
Sufferage	$O(n_2m)$
Sort-Mid	$n_2m \log m$

2.3 Adaptive workflow scheduling (AWS) algorithm

The results discussed in the paper clearly specifies that the proposed algorithm outperforms the other algorithms in all the cases and is the suitable algorithm for workflow scheduling in dynamic grid environment, where the number of resources and load on them are changing dynamically. In consideration of makespan under different amount of load change (for random task graph) we see AWS rising slightly above 1000 makespan at 30% load changes whereas other algorithms at the same percentage perform as follows: AHEFT-1150(app.), MinMin-1300(app.), and HEFT being the maximum at 1350. The detailed comparison has been shown in Fig.9, Fig.10[3].

KARP: Knowledge Acquisition with Rules as Particles
Relative performance of learning strategies has been tabulated below.

Table 3. Relative performance of the learning strategies

Comparison/iteration	50	100	200
Swarm-KARP 1 VS Genetic-Michigan 1 (%)	0.42	1.63	1.60
Swarm-KARP 1 VS Genetic-Michigan 1 (%)	-0.72	0.38	0.34
Swarm-KARP 2 VS Genetic-Michigan 1 (%)	2.10	2.50	2.47
Swarm-KARP 2 VS Genetic-Michigan 1 (%)	0.98	1.26	1.23

3 Conclusion

In the authors provide an algorithm for the challenge of selecting the appropriate resource for a specific task [1]. The implementation of Sort-Mid algorithm and various existing algorithms are tested using a benchmark simulation model. This algorithm overcomes the affection of large varies of task's execution times. The test results are indicate that Sort-Mid utilizes the grid by more than 99% at 6 instances and more than 98% at 4 instances.
A new strategy for scheduling, based on PSO is proposed. This new strategy, KARP, proves its efficiency for optimizing a wide range of functions based on the movement of swarms in the space towards best suited locations. In KARP approach each rule acts as peace that moves in the place with the final target of finding the best location and thus, the best quality to cooperate in obtaining RBs for the experts classifiers systems. KARP is intended to be an alternative to the classic genetic approach for rule discovery in FCS, Michigan approach. As simulation results show, KARP achieves an average greater accuracy (0.34–2.47%) and a faster convergence speed with the consequent reduction of computational effort. One of the most relevant advantages of KARP over the genetic approach is related to its simplicity. Finally, KARP is suggested as an efficient alternative for rule discovery in FCSs.
It is proposed a novel adaptive workflow scheduling (AWS) algorithm, to schedule workflow application in dynamic grid environment with the target to achieve the small execution time. By supporting rescheduling of tasks form overloaded resources the algorithm also provides load balancing. The simulation results apply randomly generated task graphs and corresponding to real world problems like GE and FFT demonstrates to 10% - 40% performance improvement (makespan minimization) of the proposed AWS algorithm over other scheduling algorithms considered.
In this paper we are provide architecture for hybrid computing and different methods and challenge to apply Cloud Computing in Smart Grid. Finally, Cloud platforms are inspected technical, security perspectives, and their compatibility with Smart Grid systems investigated.
We have presented a multi scheduler algorithm to show how different types of grouping can tackle parallelism in multicores get positively affect scheduling speed. In final Group Parallel Multi-Scheduler (GPMS) can be utilized in such environment, where there is a need to schedule a stream of jobs onto a set of limited resources. This type of typical environments which could benefit is Grid and Cloud environments.

References

[1] Reda NM et al., Sort-Mid tasks scheduling algorithm in grid computing, Journal of Advanced Research (2014),http://dx.doi.org/10.1016/j.jare.2014.11.010.
[2] S. García-Galán, R.P. Prado, J.E. Munoz Expósito, "Rules discovery in fuzzy classifier systems with PSO for scheduling in grid computational infrastructures," Applied Soft Computing 29 (2015) 424–435.
[3] R. Garg, A.K. Singh, Adaptive workflow scheduling in grid computing based on dynamic resource availability, Engineering Science and Technology, an International Journal (2015), http://dx.doi.org/10.1016/j.jestch.2015.01.001.
[4] Melike Yigita,V. Cagri Gungor, Selcuk Baktir, " Cloud Computing for Smart Grid applications", Computer

Networks 70 (2014) 312–329.

[5] Goodhead T. Abraham, Anne James, Norlaily Yaacob," Group-based Parallel Multi-scheduler for Grid computing", Future Generation Computer Systems, http://dx.doi.org/10.1016/j.future.2015.01.012

[6] Jinglian WANG, Bin GONG, Hong LIU, Shaohui LI, Juan YI, "Heterogeneous Computing and Grid Scheduling with Hierarchically Parallel Evolutionary Algorithms," *Journal of Computational Information Systems 10: 8 (2014)* pp. 3291–3298. Available at http://www.Jofcis.com.

[7] Pablo Ezzatti, Martín Pedemonte, Álvaro Martín, "An efficient implementation of the Min-Min heuristic" *Computers & Operations Research 40 (2013) pp.2670–2676.*

[8] Andrei Tchernykh, Uwe Schwiegelshohn, Ramin Yahyapour "ON-LINE HIERARCHICAL JOB SCHEDULING ON GRIDS," *unpublished original article.*

[9] Peng Xiao and Dongbo Liu, "Multi-scheme co-scheduling framework for high-performance real-time applications in heterogeneous grids," *Int. J. Computational Science and Engineering, Vol. 9, Nos. 1/2, (2014) pp. 55-63.*

[10] Mauro Canabe, Sergio Nesmachnow, "Parallel implementations of the MinMin heterogeneous computing scheduler in GPU," CLEI Electronic Journal, Vol. 15, No. 3, Paper- 8, Dec. 2012 pp. 1-11