Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844
Vol. IV (2009), No. 2, pp. 158-166

Task Resource Allocation in Grid using Swift Scheduler

K. Somasundaram, S. Radhakrishnan

Arulmigul Kalasalingam College of Engineering
Department of Computer Science and Engineering
Krishnankoil-626190, Tamilnadu, India
E-mail: soms72@yahoo.com

Abstract:
In nature, Grid computing is the combination of parallel and distributed computing where running computationally intensive applications like sequence alignment, weather forecasting, etc are needed a proficient scheduler to solve the problems awfully fast. Most of the Grid tasks are scheduled based on the First come first served (FCFS) or FCFS with advanced reservation, Shortest Job First (SJF) and etc. But these traditional algorithms seize more computational time due to soar waiting time of jobs in job queue. In Grid scheduling algorithm, the resources selection is NP-complete. To triumph over the above problem, we proposed a new dynamic scheduling algorithm which is the combination of heuristic search algorithm and traditional SJF algorithm called swift scheduler. The proposed algorithm takes care of Job's memory and CPU requirements along with the priority of jobs and resources. Our experimental results shows that our scheduler reduces the average waiting time in the job queue and reduces the over all computational time.
Keywords: Grid Computing, Swift Scheduler, Dynamic Scheduling Algorithm, First Come First Serve, Shortest Job First.

1 Introduction:

A computational Grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities. According to the function, Grid is classified into three types: Computing Grid, Data Grid, and Service Grid. Computing Grid is used to connect varied computing resource on the network to construct a virtual high performance computer, which could offer high performance computer [1]. The traditional computing Grid systems involve many technologies such as certification, task scheduling, communication protocols, fault tolerance and so on. The task of Grid resource broker and scheduler is to dynamically identify and characterize the available resources and to select and allocate the most appropriate resources for a given job [2]. The resources are typically heterogeneous locally administered and accessible under different local policies. Advance reservation [3] is currently being added to Portable Batch System (PBS).

In a Grid Scheduler, the mapping of Grid resources and an independent job in optimized manner is so hard where we couldn't predict optimized mapping. So the combination of uninformed search and informed search will provide the good optimal solution for mapping a resources and jobs, to provide minimal turn around time with minimal cost and minimize the average waiting time of the jobs in the queue. A heuristic algorithm is an algorithm that ignores whether the solution to the problem can be proven to be correct, but which usually produces a good solution. Heuristics are typically used when there is no known way to find an optimal solution, or when it is desirable to give up finding the optimal solution for an improvement in run time.

The primary objective of this research is to investigate effective resource allocation techniques based on computational economy through simulation. We like to simulate millions of resources and thousands of users with varied requirements and study scalability of systems, algorithms, efficiency of resource allocation policies and satisfaction of users. In our simulation we would like to model applications in the areas of biotechnology, astrophysics, network design, and high-energy physics in order to study usefulness of our resource allocation techniques. The results of our work will have significant impact on the way resource allocation is performed for solving problems on grid computing systems.

The organization of this paper is as follows. In Section 2, the related works are discussed. In section 3, we introduce our scheduling algorithm model. In section 4 we present and discuss the experimental results. We conclude this study in section 5.

2 Related Work:

Job scheduling in parallel system has been extensively researched in the past. Typically this research has focused on allocating a single resource type (e.g., CPU usage) to jobs in the ready queue. The use of many of these scheduling algorithms has been limited due to restriction in application designs, runtime system, or the job management system itself. Therefore simple allocation scheme such as first come First Serve (FCFS) or FCFS with first fit back fill (FCFS/FF) are used in practice [4].

Current job scheduling practices typically support variable resource allocation to a job, and run to completion scheduling. Scheduling policies are also heavily based on First-come-First-serve (FCFS) methods [5]. A FCFS scheduling algorithm allocates resources to jobs in the order that they arrive. The FCFS algorithm schedules the next job in ready queue as soon as sufficient system resources become available to meet all of the job requirements. The advantage is that this provides level of determinism on the waiting time of each job[6]. The disadvantage of FCFS shows up when the jobs at the head of the ready queue cannot be scheduled immediately due to insufficient system resources, but jobs further down the queue would be able to execute given the currently available system resources. These latter jobs are essentially blocked from executing while the system resource remains idle.

Fidanova [7] compared the simulated annealing approach with the ant algorithm for scheduling jobs in Grid. David Beasley, Marek Mika and Grzegorz Waligora [8] formulated the scheduling problem as a linear programming problem and proposed local search meta-heuristic to schedule workflow jobs on a Grid. Fair Share scheduling [12] is compared with simple fare task order scheduling, adjusted fair task order scheduling and Max-min fair share scheduling algorithm are developed and tested with existing scheduling algorithms.

Rafael A. Moreno [9] addresses the issues that the resource broker has to tackle like resource discovery and selection, job scheduling, job monitoring and migration etc. Resource Management System [10, 11] was discussed and models of grid RMS availability by considering both the failures of Resource Management (RM) Servers and the length limitation of request queues. The resource management systems (RMS) can divide service tasks into execution blocks (EB), and send these blocks to different resources. To provide a desired level of service reliability, the RMS assigns the same EB to several independent resources for parallel (redundant) execution.

3 Swift Scheduler (SS) Model:

Let N be the number of jobs in Job Queue 'J_q' which is indicated as,

$$J_q = \{J_1, J_2, J_3, ..., J_N\} \tag{1}$$

Jobs are allotted to M number of resources in Resource Queue'R_q'which is indicated as,

$$R_q = \{R_1, R_2, R_3, ..., R_M\} \tag{2}$$

Let $F(J_i, R_j)$ be the overall job completion time for the i^{th} job in j^{th} resources can be calculated as,

$$\sum_{i=0}^{N}\sum_{j=0}^{M} F(J_i, R_j) = \sum_{i=0}^{N}\sum_{j=0}^{M} G(J_i, R_j) + \sum_{i=0}^{N}\sum_{j=0}^{M} H(J_i, R_j) \tag{3}$$

Let $G(J_i, R_j)$ be the expected job completion time of the i^{th} job in j^{th} resources which can be calculated as,

$$\sum_{i=0}^{N}\sum_{j=0}^{M} G(J_i, R_j) = \sum_{i=0}^{N}\sum_{j=0}^{M} (JL_i/RC_j) \tag{4}$$

JL_i be the Job length of i^{th} Jobs and RC_j be the capacity of the j^{th} resources.
Let $H(J_i, R_j)$ be the heuristic function of the i^{th} job in j^{th} resources which can be calculated as

$$\sum_{i=0}^{N}\sum_{j=0}^{M} H(J_i, R_j) = \sum_{i=0}^{N}\sum_{j=0}^{M} (JL_i/RC_j) + Communicationoverhead \tag{5}$$

3.1 Working Principle of Swift Scheduler (SS):

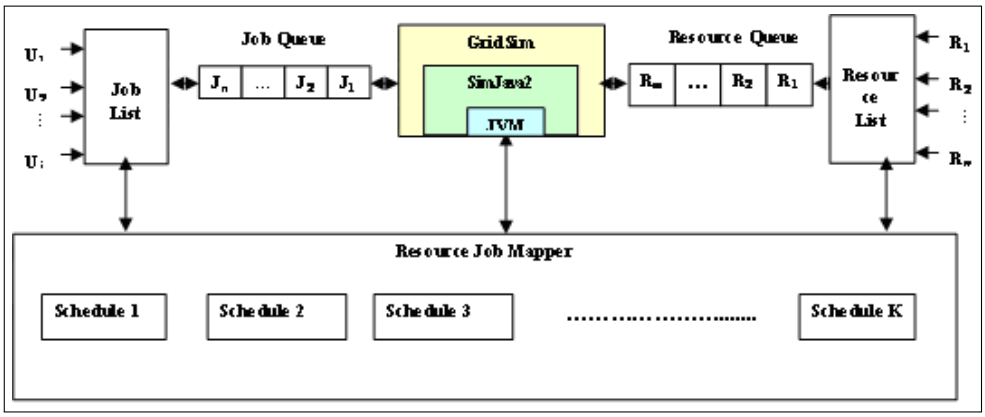

Figure 1: Architecture of SS

Figure 1 shows working principles and architecture of our proposed algorithm SS as follows: incoming jobs from different users are collected and stored in job list and available Resources are stored in resource list. Jobs are randomly arrived to job queue as well as resources are selected based on the availability. The swift scheduler in GridSim [13] maps the jobs from job queue and resources from resource queue where the resources are selected using the heuristics function. The function will select the optimized resource for the particular job to complete it with minimum time.

3.2 Pseudo Code for Swift Scheduler (SS):

```
  procedure SwiftScheduler()
begin:
   initialize job_queue, resource_queue;
   Loop exec for 'N' jobs
   begin:
        initialize job_servicetime;
        addJobs(N) to job_queue;
    end;
    Loop exec for 'M' Resources
    begin:
        addResources(R) to resource_queue;
    end;
    arrange jobs in ascending order based on job length and maintained in
    jobsQueue;
    Loop exec for 'N' jobs
    begin:
      Loop exec for 'M' resources
      begin:
       calculate the processing time of 'N' jobs in 'M' resources;
      end;
     end;
     Loop exec for 'N' jobs
 begin:
    Loop exec for 'M' resources
    begin:
      select the lowest processing time resource for the each jobs using
      heuristic function;
      allocate(n,m);
    end;
 end;
end;
```

4 Performance Analysis:

In this section, we analyze the performance of Swift Scheduler with existing Simple Fair Task Order Scheduling (SFTO) against large set of independent jobs with varying size and large number of heterogeneous resources. Assume, the arrival rates of jobs are based on the Poisson distribution. The following Fig.(2) and Fig.(3) shows the job allocation methods used in SFTO and SS respectively and the following Table (1) and Table (2) shows the arrival order of the particular jobs and at what time, the jobs will start its execution in the particular resource and its service time of the jobs in the particular resource where the selection of resources are based on the algorithms.

For example, In SFTO, the jobs J0 and J3 are allotted in resource R1. The residing time (T_r) of jobs J0 is the combination of jobs J0 waiting time (T_w) in queue and service time (T_s) of J0 (i.e $T_r = T_w + T_s$ = 2820.03ms + 61.35ms = 2881.38ms). Similarly, the residing time of job J3 is 1490 ms. The job J2 is allotted in resource R2 where the residing time of J2 is 1257.19 ms and jobs J4 and J1 are chosen

JobID	Resource Name	Start time in ms	Residing time in ms
4	R3	1851.12	1886.86
1	R3	3337.99	3391.73
2	R2	1214.79	1257.19
3	R1	1330.66	1490.66
0	R1	2820.03	2881.38

Table 1: Job, Resource, start and residing time for SFTO

JobID	Resource Name	Start time in ms	Residing time in ms
2	R1	1210.02	1316.02
0	R1	2805.39	2980.39
3	R2	1093.04	1178.27
1	R2	2688.17	2772.17
4	R3	1453.51	1489.25

Table 2: Job, Resource, start and residing time for SS

resources R3 where the residing time of J4 is 1886.86 ms and J1 is 3391.73 ms. The average waiting time of all jobs in grid system is 2181.56ms.

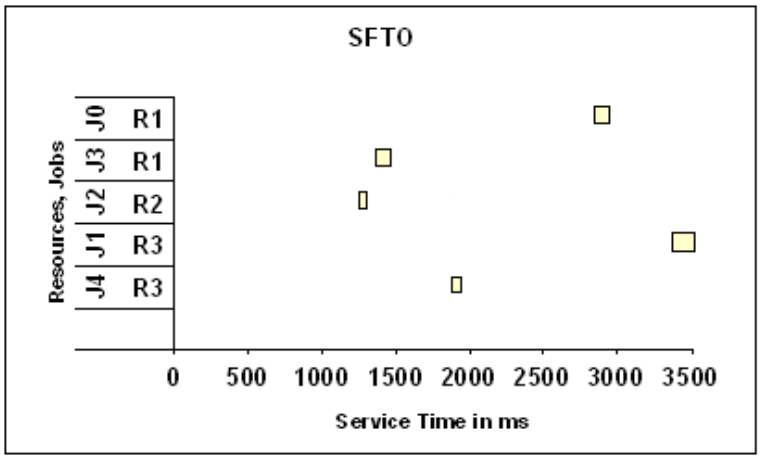

Figure 2: Job and Resource Allocation for SFTO

In SS, resource selection and jobs allocation are based on heuristic searching algorithm on SJF, which reduces the average waiting time of the jobs in queue. So, overall turn around time is reduced and resource utilization is increased. For example, in SS, the jobs J2 and J0 are allotted to resource R1 where its residing time is 1316.02 ms and 2980.39 ms respectively. Similarly, Jobs J3, J1 and J4 are chosen by resources R2, R2 and R3 respectively where residing time of Jobs J3,J1 and J4 are 1178.29 ms, 2772.17 ms and 1489.25 ms respectively. The average waiting time of all jobs in grid system is 1947.22 ms which is less than SFTO. The statistical data presented here is acquired by averaging the scheduling performance over different runs. The following figs and tables shows the cost based, total processing time and resource utilization based comparison test results of FCFS, SJF, SFTO and SS against varying number of jobs and resources.

Our proposed algorithm Swift Scheduler (SS) is compared with FCFS, SJF and Simple Fare Task Order (SFTO) scheduler. We have tested SS in GridSim by varying number of resources, no. of jobs

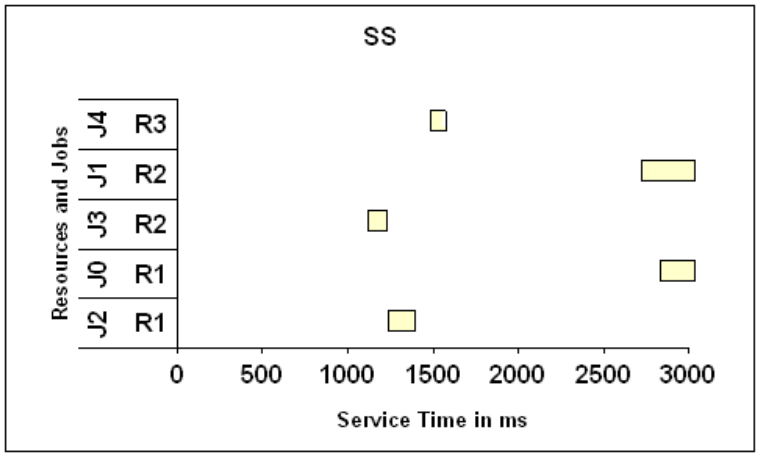

Figure 3: : Job and Resource Allocation for SS

against total processing time, cost, and resource utilization. We can vary the number of resources like 5, 10, 15, 20,... etc. For experimental purposes, two sample simulation results are shown in Figure 4, 5, 6, 7, 8 & 9. By analyzing the obtained results from the simulator, Swift Scheduler completed all jobs with minimum time and cost by utilizing maximum amount of resources compares with other scheduler like FCFS, SJF and SFTO.

Total Processing Time Analysis

Figure 4: : No of Jobs Vs Total Processing Time

Figure 5: : No of Jobs Vs Total Processing Time

Cost Analysis

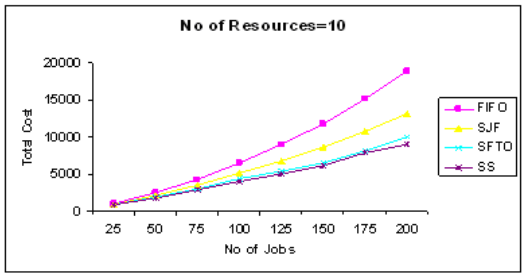

Figure 6: : No of jobs Vs Total cost

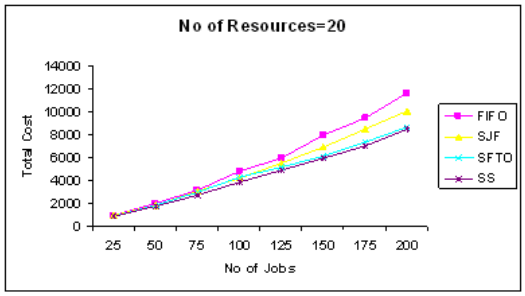

Figure 7: : No of jobs Vs Total cost

Resource Utilization

Figure 8: : No. of jobs Vs Resource Utilization

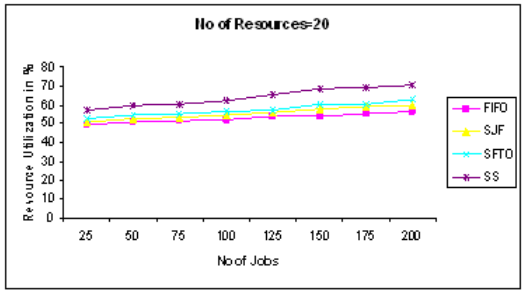

Figure 9: : No. of jobs Vs Resource Utilization

5 Conclusion and Future Work:

In this paper, we have presented the design and analyze the new scheduling algorithm Swift Scheduler. Our proposed Swift scheduler completed a task by using highly utilized low cost resources with minimum computational time. Our scheduling algorithm uses the heuristic function to select the best resources to achieve a higher throughput while maintaining the desired success rate of the job completion. This algorithm is performing better for real time job parameters and suitable for different job sizes in real environment. However, in all conditions, the proposed algorithms outperform the traditional ones. The SS policy is more effective than the FCFS, SJF and SFTO in the extent of computational complexity with lower cost but higher resources utilization. In future, we can hybrid the Swift Scheduler with any evolutionary scheduling algorithm like Genetic algorithm, Particle Swarm Optimization technique to achieve a high throughput and high resource utilization.

Bibliography

[1] Ranganathan,K, and I.Foster, "Decoupling Computation and Data Scheduling in Data Intensive Applications", *11th International Symposium on High Perfromance Distributed Computing*, Edinburgh, Scotland, Condor Project, Condor-G, 2002.

[2] Mitrani I, Palmer J," Dynamic Server Allocation Heterogenous Clusters ", *1st International working conference on Heterogeneous Networks*, Ilkley,UK, 2003.

[3] Foster,I, et al, "The Grid 2003 Production Grid : Principles and Practice", *13th International Symposium on High Performance Distributed Computing*, 2004.

[4] Vijay Subramanian, Rajkumar Kettimuthu, et al, "Distributed Job Scheduling on Computational Grids using Multiple Simultaneous Requests", *Proceedings 11th IEEE International Symposium on High Performance Distributed Computing*, 2002. HPDC-11 2002. Pages: 359- 366

[5] C.Bitten, J. Gehring, et. Al, "The NRW - Meta Computer : building block for a worldwide computational Grid", *proceeding of the 9th Heterogeneous Computing workshop*, pp.31-40, 2000.

[6] C.Ememann, V. Hamscher, et. al, "On Advantageous of Grid Computing for parallel job scheduling", *proceeding 2nd IEEE/ACM Int'l Symp. On cluster- computing and the Grid (CCGRID 2002)*, Berlin, 2002, IEEE press.

[7] Stefka Fidanova, "Simulated annealing for Grid Scheduling problem", *IEEE International Symposium on International Symposium on Modern Computing*, 2006. JVA apos;06 3-6 Oct. 2006 Page(s):41 - 45

[8] Marek Mika, Grzegorz Waligora and Jan Weglarz, A Meta-Heuristic Approach to scheduling Workflow jobs on a Grid, Grid resource management: *state of the art and future trends*, ISBN : 1-4020-7575-8 , Kluwer Academic Publishers, Norwell, MA, USA; pp : 295 - 318, 2004.

[9] Moreno, R., Alonso-Conde A.B, Job Scheduling and Resource Management Techniques in Dynamic Grid Environments In et al., F.F.R., ed.: *Across Grids 2003, Volume 2970 of Lecture Notes in computer science*, Springer, pp : 25 - 32, 2004.

[10] Yuan-Shun Dai, Min Xie and Kim-Leng Poh "Availability Modeling and Cost Optimization for the Grid Resource Management System" *IEEE Transactions on Systems, Man and Cybernetics*, Part A Volume 38, Issue 1, Jan. 2008 Page(s):170 - 179.

[11] Yuan-Shun Dai and Gregory Levitin "Optimal Resource Allocation for Maximizing Performance and Reliability in Tree-Structured Grid Services" *IEEE Transactions on Reliability* Volume 56, Issue 3, Sept. 2007 Page(s):444 - 453.

[12] Doulamis, N.D.; Doulamis, A.D.; Varvarigos, E.A.; Varvarigou, T.A "Fair Scheduling Algorithms in Grids" *IEEE Transactions on Parallel and Distributed Systems*, Volume 18, Issue 11, Nov. 2007 Page(s):1630 - 1648

[13] Anthony Sulistio, Uros Cibej, Srikumar Venugopal, Borut Robic and Rajkumar Buyya "A Toolkit for Modelling and Simulating Data Grids: An Extension to GridSim", *Concurrency and Computation: Practice and Experience (CCPE)*, Online ISSN: 1532-0634, Printed ISSN: 1532-0626, 20(13): 1591-1609, Wiley Press, New York, USA, Sep. 2008.