Node Allocation In Grid Computing Using Optimal Resouce Constraint (ORC) Scheduling

K.Somasundaram[1], S.Radhakrishnan[2]
[1]Research Scholar [2]Research Supervisor and Professor
[1,2]Department of Computer Science and Engineering,
Arulmigu Kalasalingam College of Engineering, Krishnankoil-626190, Tamilnadu, India

Abstract
The objective of grid computing is to distribute the jobs to heterogeneous environment and increase the speed of processing the jobs. Most of the Grid tool kits use the First Come First Served (FCFS) based scheduling algorithm to distribute the job to different executors or client nodes. In FCFS scheduling algorithm, doesn't consider the capabilities of executors or client and also many processes runs on a processor, has bottlenecks in Grid environment such as when scheduling workloads that are subject to time constraints, it increases the waiting time of list of processes. We proposed a new scheduling algorithm called Optimum Resource Constraint (ORC) Scheduling algorithm which includes the combination of both the Best fit allocation and Round Robin scheduling to allocate the jobs in queue pool. The algorithm was implemented in local and global grid schedulers. This improved the efficiency of load balancing and dynamicity capability of the Grid resources. We measured the performance of ORC algorithm on a Grid computing with maximum number of jobs and nodes.

Key words:
Queue pool, Grid computing, Load balancing, ORC scheduling algorithm

1. Introduction

The Grid is a collection of interconnected stand-alone computers working together as a single, integrated computing resource consisting of many of the same or similar types of machine. Grid computing, most simply stated that is distributed computing taken to the next evolutionary level. The goal is to create the illusion of a simple yet large and powerful self managing virtual computer out of a large collection of connected heterogeneous systems sharing various combinations of resources. Recently there has been increasing interest in service composition on the web and also in creating and managing workflows of tasks on computational grids. Scheduling techniques will play an essential role in managing workflows of grid services, including their composition, allocation of resources and execution.

The resource management is to run the jobs, monitor their status, and return outputs when jobs are completed. The resource manager will consult the broker about resource assignments and launch the jobs with the appropriate resources [4]. Moreover, it must authenticate the user and check if he/she is authorized person to access the resources before assigning the job. The schedulers combine the functionality of resource manager, information service and resource broker. The basic role of a broker is to provide match-making services between a service requester (jobs submitted for execution) and a service provider (available resources on the Grid) [5]. A request specifies what are required to execute a job such as CPU power, minimum memory, disk space, etc. It keeps track of which machines are available to run jobs, how the available machines can be utilized and given for all the users who want to run jobs on them, and when a machine is no longer available. It keeps track of a user's jobs and asks the job manager to show the job queue and submit the new jobs to the system, to put jobs on hold, and to request information about jobs that have completed, other part is to provide information service [6]. Our proposed ORC, capable to do all the above mentioned operations of resource management, also it helps to minimize the complexity of process allocation and reduce the waiting time of the process in the Grid environment. Remaining paper is organized as follows: Section 2 and 3 describe the related works and problem definition. Section 4 explains ORC algorithm. Study of ORC performances was done in section 5 and finally we conclude and our future work described in section 6.

2. Related Works

On-line Algorithm for Fair-Share Node Allocations in a Cluster [1] is an application with predefined portions of system resources based on fair share node allocation in cluster which describes the Single node operating systems use context-switch (preemption) to dynamically allocate the CPU(s) to running processes. They presented preemptive process migrations for dynamic allocations of nodes to users. They developed a proportional share allocations can be achieved in a relatively short time (minutes) on a MOSIX organizational Grid. Maximum number of Nodes is limited and moreover it reduces the CPU speed with the increased process.

Highest Response Next Scheduling [14] provides more responses with time, memory and CPU requirements. They had identified that the HRN model for scheduling system is much adaptive for Grid environment. They allotted the jobs

Manuscript received June 5, 2008.
Manuscript revised June 20, 2008.

to number of processor based on job's priority and processor's capability. This scheme is adaptive for local jobs and remote jobs without any loss of performance and highly adaptive for grid environment. It is not suitable for more number of jobs allocations. Because of finding priority of job is tedious one.

Optimal Multiple QoS-based Grid resource scheduling models [8] solves the scheduling problems using optimization techniques. They proposed an idea of decomposing a global optimization problem in multiple QoS based resource scheduling into two sub problems: task agent optimization and resource agent optimization. It simplifies the problem and makes it mathematically tractable (non linear model). The experiments shown that an optimal multiple QoS based resource scheduling involves less overhead and leads to more efficient resource allocation but the main issue is less possibility for optimal resource allocation.

Parallel Job Scheduling Policy for Workstation Cluster Environments [16] which shows the best solution to the processor allocation problem in a distributed multiprocessor environment is an adaptive scheduling policy that can adjust load distribution based on runtime scheduling algorithms. The main idea of adaptive space-sharing policies is that the number of processors assigned to a job is a compromise between the user's request and what the system can provide and they handle process fragmentation problem with desired solution based on system utilization and load sharing

Hiroyuki Ohsaki et.al introduced a DRM-DC (Dynamic Resource Management with Delay Compensator) [17] algorithm based on the amount of available resources in a site changes over time and sites are geographically distributed, the transfer delay between sites cannot be neglected for computing resource management. The main feature of DRM-DC is that it realizes high steady state and transient state performance in wide-area Grid computing using a delay compensator called Smith predictor to improve the resource management

A Load Balancing Model for Grid Environment [18] to improve the global throughput of these environments, effective and efficient load balancing algorithms are fundamentally important. This model is characterized by main features such as hierarchical, supports heterogeneity & scalability and totally independent from any Grid physical architecture. It uses a task-level load balancing it privileges local tasks transfer to reduce communication cost and it is a distributed strategy with local decision making.

Xiong Li et.al introduced Concurrent Negotiation for Agent Based Grid Computing" [4] and experimented on the Grid with the challenges about effective load balancing for Grid Computing, because of the highly heterogeneous and complex environments. To solve these problems, they introduced concurrent negotiations model, in which an auction is mapped into a one-to-many negotiation between one seller agent and many buyer agents in service-oriented contexts to provide optimize computing resources allocation. With the volume of the data transferred increases the traffic and load on the heterogeneous environment.

3. Problem Formulation

The proposed ORC algorithm was designed that the jobs are allocated to the processor based on the best fit algorithm followed by the round robin techniques as shown in Figure-2 and 3. In ORC each jobs are allotted to the processor based on its capabilities and also verified about the processor's capability where it was assigned. The processing time of each job are calculated by the fraction of jobs size and processor capability. The processors capabilities and jobs size are measured in terms of Gflops. The processing time and waiting time of each job are calculated as:

Processing time = (Processing capability needed for jobs (in Gflops)) / (Processing capacity of processor where the jobs were allotted (in Gflops))

$$\text{Waiting time} = \sum_{K=P_1}^{K=P_N} \min \text{ (processing time of jobs allotted in processor K)}$$

The main objective is to dynamically schedule the process based on pre-emption that best suits among the clients. The ORC scheduling algorithm which repeatedly runs through a list of jobs and giving opportunity to each job for using the processor in succession. The algorithm of ORC is shown below:

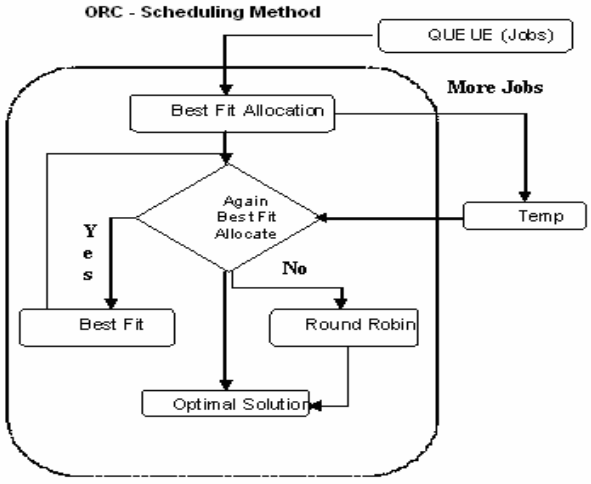

Figure-2: ORC scheduling basic frame work

Figure-3 Pseudo code for ORC

```
Procedure ORC_scheduler ( var jobs[j1..jn],
              var jobs_size_in_Gflops[j1..jn],
              var processor[p1..pn],
              var processor_capab_in_Gflops[p1..pn])
begin
  var processing_time[n];
  var queue[n];
  for(i=j1;i<=jn;i++)
  begin
    for(k=p1;k<=pn;k++)
    begin
      if (max (jobs_size_in_Gflops[i] <=
         max(processor_capab_in_Gflops[k])
      begin
         allocate (processor[k], job_processing_time[i]) :=
          (jobs_size_in_Gflops[i]/(processor_capab_in_Gflops[k]))
       else
         compute_waiting_time(not allotted jobs);
         RRQueue[i]:=jobs[i];
         if(free(p[k]= = true)
         begin
              allocate(p[k],RRQueue[i]);
          end
       end
     end
                                  end
end
```

4. Implementation

We simulated our work with following test scenarios. Table-1 shows the list of processors and its capabilities measured in terms of Gflops.

Name of the processor	Processing Capability in Gflops
P_1	10
P_2	5
P_3	2
P_4	20

Table-1: Processor's and its processing capability

Table-2 shows the list of jobs and its processor requirements in terms of Gflops

Jobs	**Processor requirements in Gflops**
J_1	100
J_2	10
J_3	200
J_4	50
J_5	5
J_6	60
J_7	1
J_8	20
J_9	2
J_{10}	75

Table-2: Jobs and its processor requirements

When jobs are coming in to the queue pool, the ORC scheduling algorithm searches the all jobs in the queue. It scans the processor and job's capability, apply best fit algorithm followed by round robin techniques. In our scenario, Jobs J_2, J_5, J_8 and J_9 are allotted first to the processors P_1, P_2, P_4, P_3 in first epochs based on best fit and its processing time is 1sec as shown in Table-3. The local scheduler will allocate the jobs to the processor based on its capability and job's size.

Processors	**Jobs**	**Processing Time in Sec**	**Jobs in Round Robin Queue**
P_1	**J_2**	**1**	**J_1, J_3, J_4, J_6, J_7,J_{10}**
P_2	**J_5**	**1**	
P_3	**J_9**	**1**	
P_4	**J_8**	**1**	

Table-3: Jobs allotted to the processor in epoch1

In second epochs, the jobs put it in Round Robin queue and again search the processor's capability to run the jobs. In our scenario, Jobs J_1, J_3, J_4, J_6, J_7 and J_{10} are in Round Robin queue. After the completion of allotted jobs by the processor, the jobs are allotted to the best fit free processors. This process will be continuing until all the jobs are completed.

5. Performance Analysis

The ORC scheduling in Grid environment includes the best fit followed by round robin scheduling which distribute the jobs among the available processors. The remaining un-allotted jobs are queued for next execution. Thus the order of jobs execution increases the performance of the processor and distribute load effectively across the network. This will reduce the waiting time of jobs in the queue and avoid the starvation. We compared the performance of ORC with First come First Served (FCFS), Shortest Job First (SJF) and Round Robin (RR).

The Table-4 shows the average waiting time for different no. of jobs, for different algorithms like First come First Served (FCFS), Shortest Job First (SJF), Round Robin (RR) and ORC.

PROCESS	**FCFS**	**SJF**	**RR**	**ORC**
5	17	15	14	11
10	47	44	42	38
15	80	72	69	62
20	115	106	101	93
25	152	139	128	112
30	196	176	170	139

Table-4: Average waiting time of process

The performance of FCFS, SJF, RR and ORC algorithms are studied in Figure-4 and shows ORC giving better performance than other algorithms.

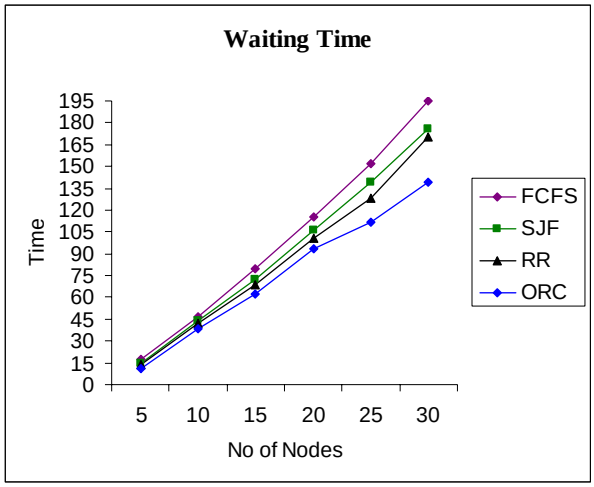

Figure-4: Process Vs Waiting Time for Jobs

Table-5 shows the average turn around time for different jobs, for different algorithms like FCFS, SJF, RR and ORC.

PROCESS	FCFS	SJF	RR	ORC
5	27	25	24	23
10	50	57	55	53
15	92	85	80	76
20	128	119	112	102
25	167	153	142	132
30	212	192	186	178

Table-5: Turn around time for running process

Figure-5 shows the performance analysis of turnaround time for different algorithm. From the figure, by using ORC the turnaround time for jobs are reduced and response time of the process is increased.

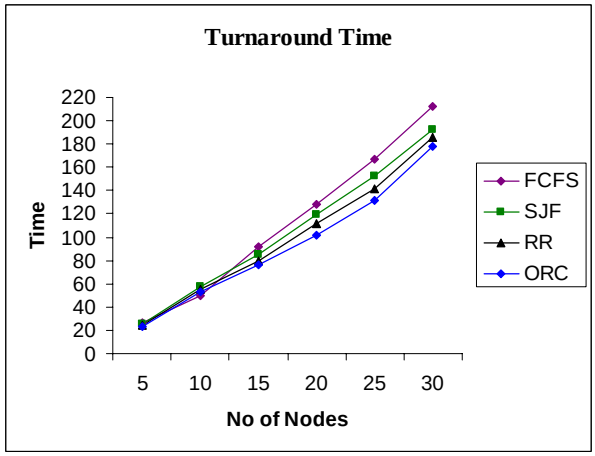

Figure-5: Process Vs Turnaround Time for Jobs

6. Conclusion and Future work

The major issues in Grid environment is resource management which is done by either local or global schedulers. Many tool kits will allocate the jobs by using FCFS scheduler. When allotting jobs to processor using FCFS, it doesn't consider the capability of the processor. Using HRN, the smallest jobs are processed immediately and longest jobs will wait for long time. This problem was overcome by ORC scheduling. The ORC scheduling reduces the waiting time of job and allocates the jobs to processor based on its capability and increase the processing time of job. The proposed method will calculate the capability of jobs and processor to allocate the jobs. It will reduce the waiting time of jobs in queue and turn around time. Thus the overall performance of the Grid has increased. We have implemented the ORC algorithm in local scheduler. In future, we plan to implement the ORC in Global scheduler and analyze the performance.

References

[1] Lior Amar, Amnon Barak, Ely Levy Michael Okun, "An On-line Algorithm for Fair-Share Node Allocations in a Cluster" Seventh IEEE International Symposium on Cluster Computing and the Grid- 2007

[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] Xiong Li, Yujin Wu, Kai Wang and Zongchang Xu," Concurrent Negotiation For Agent-Based Grid Computing", Proc. 5th IEEE Int. Conf. on Cognitive Informatics (ICCI'06)

[5] A. Barak, A. Shiloh and L. Amar, "An Organizational Grid of Federated MOSIX Grids", CCGrid- 05, Cardiff, 2005.

[6] Abdulla Othman Peter Dew Karim Djemame Iain Gourlay "Adaptive Grid Resource Brokering" Proceedings of the IEEE International Conference on Cluster Computing – 2003

[7] L. Amar, A. Barak, Z. Drezner, and I. Peer, "Gossip Algorithms for Maintaining a Distributed Bulletin Board with Guaranteed Age Properties", submitted for publication, 2006.

[8] Li Chunlin, Li Layuan, "Optimal Multiple QoS Resource Scheduling In Grid Computing", Proceedings of the 20th International Conference on Advanced Information Networking and Applications (AINA'06)-06

[9] B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, I. Pratt, A. Warfield, P. Barham and R. Neugebauer, "Xen and the Art of Virtualization", OSDI, 2003.

[10] S.D. Kleban and S.H. Clearwater, "Fair Share on HighPerformance Computing Systems: What Does Fair Really Mean?" CCGrid-03, 2003.

[11] K. Lai, L. Rasmusson, E. Adar, S. Sorkin, L. Zhang and B. A. Huberman, "Tycoon: an Implementation of a Distributed Market-Based Resource Allocation System", TRarXiv:cs.DC/0412038, HP Labs, 2004.

[12] E. Gabriel et al. “Open MPI: Goals, concept, and design of a next generation mpi implementation”. In 11th European PVM/MPI Users’ Group Meeting, 2004.
[13] R.T. Aulwes et al. "Architecture of LA-MPI, a network-fault-tolerant mpi” In 18th Intl Parallel and Distributed Processing Symposium, 2004.
[14]K.Somasundaram,S.Radhakrishnan,M.Gomathynayagam “Efficient Utilization of Computing Resources using Highest Response Next Scheduling in Grid” in Asian Journal of Information Technology 6 (5): 544-547, 2007.
[15] Lior Amar, Amnon Barak, Ely Levy and Michael Okun ” An On-line Algorithm for Fair-Share Node Allocations in a Cluster” IEEE International Symposium on Cluster Computing and the Grid-CCGrid-2007
[16] J. H. Abawajy and S. P. Dandamudi “Parallel Job Scheduling Policy for Workstation Cluster Environments” Proceedings of the IEEE International Conference on Cluster Computing (CLUSTER’00)
[17] Hiroyuki Ohsaki, Soushi Watanabe, and Makoto Imase “On Dynamic Resource Management Mechanism using Control Theoretic Approach for Wide-Area Grid Computing” Proceedings of 2005 IEEE Conference on Control Applications, 2005. CCA 2005 Volume , Issue , 28-31 Aug. 2005 Page(s):891 - 897
[18] Belabbas Yagoubi ,Meriem Medebber ” A Load Balancing Model for Grid Environment” IEEE International Symposium on Computer and Information Sciences, 2007. ISCIS 2007 Volume, Issue, 7-9 Nov. 2007