Immediate Mode Scheduling Methods for Independent Jobs on Open Online Heterogeneous Systems

Abhishek Kumar[†], Navneet Chaubey[†], Sireesha Yakkali[†]

Computer Science and Information Systems Department,
Birla Institute of Technology and Science, Pilani – Goa Campus
Zuarinagar, Goa – 403726, INDIA

Email: (*abhishek.kumar.ak*}@gmail.com, {*f2005512 and f2006081*}@bits-goa.ac.in

Abstract

Grid infrastructures and grid based applications are becoming common approaches for solving large scale science and engineering problems. The efficient scheduling of independent computational jobs in a heterogeneous computing (HC) environment is an important problem in domains such as grid computing. In this work, we consider an online scheduling problem in immediate mode, where jobs arrive over time and are allocated to machines as soon as they arrive. All jobs' characteristics are unknown before their arrival times. We implemented several scheduling algorithms and measured three metrics for comparison: response time, bounded slowdown and system utilization. Our simulation allowed us to identify which of the considered methods perform better for response time, bounded slowdown and utilization at different system loads. We also evaluate the usefulness of the methods if certain grid characteristics such as heterogeneity of jobs and resources are known in advance.

Keywords: Immediate mode scheduling, Heterogeneous systems

1. Introduction and Motivation

A grid is a heterogeneous system in which resources may belong to different organizations. Grid computing[1] started as a project to link geographically dispersed supercomputers[2]. It resulted in the development of several large scale applications(such as NetSolve[3]).

A complex computational problem benefits by using many nodes of the grid at the same. In open, online systems, jobs arrive over time and the characteristics of these jobs (such as expected computational time) are not known beforehand.

Grid systems usually span over multiple organizations and are dynamic in nature. They usually receive computational tasks or jobs from multiple organizations and users. A scheduler in such an environment needs to allocate different jobs to diverse machines, each with its own computational capacity. One important goal for the scheduler is to provide load balancing of the resources so that idle time is minimized. An important necessity for such a scheduler is to allocate jobs to resources as fast as possible. Immediate mode methods of scheduling fall into this category since they require very little time for allocation. They are very efficient in terms of scheduling time compared to sophisticated methods (Such as those involving genetic algorithms[4]).

In the immediate mode, a job is scheduled as soon as it arrives without any waiting interval. In [5], the authors have examined 11 heuristics for mapping jobs to resources statically. In [6], the authors have examined a subset of these methods that employ immediate mode scheduling. Both these results examine scheduling methods on an offline system.

In this work, we examine five different immediate mode methods on an open, online heterogeneous system: *Opportunistic Load Balancing* (OLB), *Minimum Execution Time* (MET), *Minimum Completion Time* (MCT), *Switching Algorithm* (SA) and *k-Percent Best* (kPB). We also propose a new method, *Modified MCT*, based on certain observed heuristics. We implemented these methods and tested them using a benchmark of instances proposed by Braun et al. [5]. This benchmark of instances is obtained from an expected time to compute model that simulates the job runtimes on different nodes of a heterogeneous system. This benchmark has been widely used in examining heterogeneous systems [6] and is known to be one of the most difficult benchmarks in the literature [6].

The rest of the paper is organized as follows. Section 2 presents the problem description. Section 3 provides an overview of the benchmark and metrics used for evaluation. It also explains the job arrival model that we have used to model the time interval between job arrivals. Section 4 discusses the immediate mode methods considered in this work. In section 5, we provide some computational results of our simulation and in section 6, we present our observations. We end in section 7 with conclusions and future work.

[†]Student authors from Birla Institute of Technology and Science, Pilani – Goa Campus.

2. Problem Description

We consider a scheduling problem where jobs are to be allocated immediately to resources in a global, heterogeneous and dynamic environment. The allocation should be as fast as possible, while at the same time optimizing several criteria such as response time, utilization and slowdown (explained in section 3).

The jobs have to be completed on a unique resource. There are no dependencies between jobs (each job is independent). The arrival rate of jobs determines the system load.

Since we consider a heterogeneous environment, the processing capacity of each resource in the system may vary significantly and thus yield different runtimes for a particular job on different machines. In order to formalize our definition, we use the ETC matrix model to simulate task and machine heterogeneity (explained in section 3).

An instance of the problem consists of:

- A number of independent jobs to be scheduled
- A number of heterogeneous machines (resources)
- The job's expected time to compute on each of the machines. (This depends on the workload of each job and computing capacity of each machine). The ETC matrix: ETC[*i*][*j*] is the expected execution time for job *i* on machine *j*.
- Ready time (*ready*[*m*]) the time when machine *m* will finish previously assigned jobs.

In this work, we assume that the computation time for each job is known accurately before the job begins execution.

3. Benchmarks and Metrics Used

Real world heterogeneous systems, such as a computational grid, are complex combinations of hardware, software and network components. In order to make fair comparisons of different techniques used for different systems, we require a benchmark simulation model. Braun et al. [5] describes such a model for HC environments taking into consideration task heterogeneity, machine heterogeneity and consistency. Essentially, the running time of each individual job on each processor (resource) must be known and this information can be stored in an 'expected time to compute' (ETC) matrix. A row in an ETC matrix[1] contains the ETC for a single job on each of the available processors and so any ETC matrix will have *n* x *m* entries where *n* is the number of jobs and *m* is the number of processors. A simple ETC matrix with 4 jobs and 2 processors is given in table 1. This is a consistent ETC matrix, as processor 1 is consistently faster than 2.

[1]We have used the range based method for ETC Matrix calculation with ranges for R_{mach} being 10 and 1000; Ranges for R_{task} being 100 and 3000 (lo and hi) as proposed by Braun et al.

Table 1. An example of ETC matrix.

	Processor 1	Processor 2
Job 1	3	5
Job 2	8	10
Job 3	1	6
Job 4	7	12

Thus, the ETC matrix model is able to capture most important characteristics of the heterogeneous system as well as the independent jobs.

In order to examine scheduling methods on an online open system, we also need to model the arrival rate of jobs. The arrival rate of jobs determines the system load. Various performance metrics (such as response time) are studied with respect to system load when studying online open systems [7]. We assume a scheduling environment where jobs arrive in a Poisson process. The mean inter arrival time is adjusted to match the desired load on the system. Hence, for instance, a mean inter arrival time of 0 units would put our system on infinite load (fig. 1).

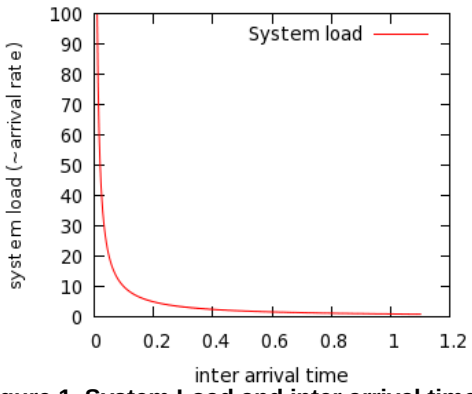

Figure 1. System Load and inter arrival times.

Various metrics have been described in the literature for performance evaluation[7]. The most common metric used for open online systems is response time(fig. 2).

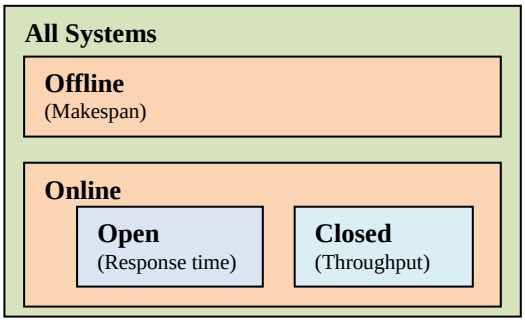

Figure 2. Commonly used metrics.

Response Time. For online open systems, response time is the metric used for performance evaluation. Response time of a job is defined as the sum of the waiting time and execution time.

Responsetime[*i*]= *ready*[*schedule*[*i*]] + ETC[*i*][*schedule*[*i*]]

Load and Utilization. Since we consider only rigid jobs for our analysis, the load and utilization are completely determined by the arrival rate of jobs [7]. System load can be increased by reducing the inter-arrival times between jobs. We model the arrival process as a memory-less poisson process with a mean inter arrival time that can be changed to adjust system load. The load variable we used is the arrival rate[7].

Bounded Slowdown. Slowdown is defined as the runtime of a job (sum of waiting and execution times) on a system when its loaded divided by runtime on the system when it is dedicated and not loaded. The importance of slowdown as a metric is that it is normalized as compared to the metric response time, which places greater emphasis on long jobs[7]. The problem with slowdown is that extremely short jobs with reasonable delays lead to excessive slowdown values. A commonly accepted solution is to apply a lower bound on job runtimes. This is known as *Bounded Slowdown*.

4. Immediate Mode Methods

In this work, we consider six immediate mode methods, namely *Opportunistic Load Balancing* (OLB), *Minimum Execution Time* (MET), *Minimum Completion Time* (MCT), *Switching Algorithm* (SA), *k*-Percent Best and *Modified MCT*.

OLB: This method assigns a job to the earliest available machine. This method does not take into account the execution times of the job. It considers the ready times of the machines and assigns the job to the machine with the least ready time. If two or more machines are idle, one of them is chosen arbitrarily. This method tries to keep each of the machines as loaded as possible and thus gives highest system utilization and load balancing.

MET: This method assigns a job to a machine that will execute it fastest. It does not consider the ready times of the machines and allocates jobs only on the basis of its expected time to compute. A disadvantage of this method is that it scores poorly on load balancing. However, the advantage is that jobs are allocated to machines that best suit their requirements regarding execution time. This method is also known as LBA (*Limited Best Assignment*) and UDA (*User Directed Assignment*)

MCT: This method assigns a job to the machine that gives the least completion time. The completion time of a job *i* is the sum of the waiting time and execution time of the job.

completion[i]=*ready*[*schedule*[*i*]] + ETC[*i*][*schedule*[*i*]]

In this case, it is possible that a job is assigned to a machine that does not have the smallest execution time. It is also possible that a job is assigned to a machine that does not get idle earliest. However the completion time for the job would be minimum on the machine scheduled. A disadvantage of MCT is that it may not give the least execution time for jobs.

SA: This method tries to combine the best features of MET and MCT. It switches between MCT and MET to achieve good load balancing and execution time respectively. The implementation of this method first calculates the ratio of minimum and maximum ready times in the system:

$r = r_{min} / r_{max}$

The value of r is compared to two threshold values, r_l and r_h, where $0 < r_l < r_h < 1$. Initially, r=0.0, and SA starts scheduling according to MCT until *r* becomes greater than r_h. Once r is greater than r_h, SA starts using MET to allocate jobs until *r* becomes lesser than r_l and then a new cycle starts again.

***k*PB:** This method first selects a subset of resources according minimum execution time (from the ETC matrix). It selects *k*% best resources, and within this subset, MCT is used. For *k*=100, *k*PB performs as MCT, while for *k*=100/*nb_machines* it behaves as MET. This is the only method that simultaneously tries to achieve the objectives of MCT and MET. One major disadvantage of *k*PB is that a machine with low processing capability may never be selected in the k% best subset and so will always remain idle or less loaded.

Modified MCT: We note that the OLB method provides highest system utilization values and keeps the system load balanced. The drawback is that it may schedule the jobs on inefficient machines where the execution time is very high. However, for 'short jobs', most of the time spent is in the waiting queue, whereas most of the time spent by a 'long job' is during execution. In this method, we first check the size of the job. If it's a small job, it is scheduled on the earliest idle machine(OLB). For medium and large jobs, the MCT method is used for allocation. The advantage of this method is that it increases system utilization and improves load balancing by allocating jobs to idle machines. The penalty incurred due to allocation on inefficient machines is very less for short jobs, which usually spend much larger times in the waiting queue.

5. Results

We implemented each of the immediate mode methods for the allocation of jobs to resources. The computation is a two step process: first, the ETC matrices and inter arrival times are calculated, then a simulation of the jobs and their execution is done while examining the metrics for performance evaluation. The immediate mode methods OLB, MET, MCT and SA have a time complexity of $O(N.M)$ where N is the

number of jobs and M is the number of machines in our system. The time complexity of the *k*PB method is $O(N.M.\log M)$.

We have used a common set of benchmark instances of the ETC model to allow a fair comparison. We have used 8 different types of benchmark instances denoted by the notation u_x_yyzz where

- *u* means that a uniform distribution has been used to generate the benchmark instance.
- *x* denotes the type of consistency (*c*-consistent, *i*-inconsistent). We have considered fully consistent and inconsistent cases in our simulation. A consistent ETC matrix model means that if a machine m_i executes a job *q* faster than machine m_j, then m_i executes all jobs faster than m_j.
- *yy* denotes the job heterogeneity (*hi*-high, *lo* - low).
- *zz* denotes the machine heterogeneity.

The resource utilization of each method at different load conditions is presented in figure 3. The mean completion, waiting and execution time for a particular instance on a moderately loaded system is presented in figure 4. We use the job arrival rate as our load variable and by 'moderately loaded' we mean that the arrival rate of job is approximately equal to the service rate of our system. The bounded slowdown values are presented in figure 5. We provide the response time for each method as a function of system load in figure 6.

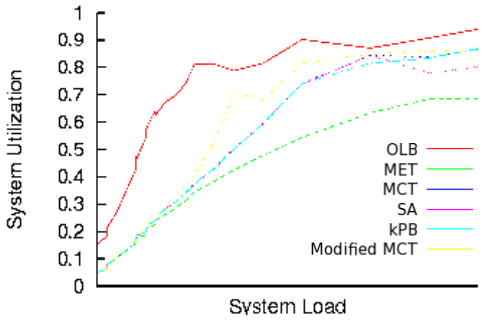

Figure 3. Average Resource Utilization.

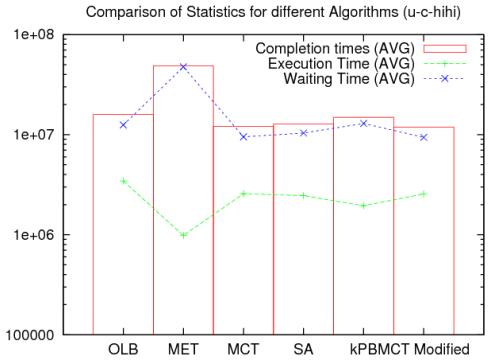

Figure 4. Average wait, execution and completion times for a moderately loaded system.

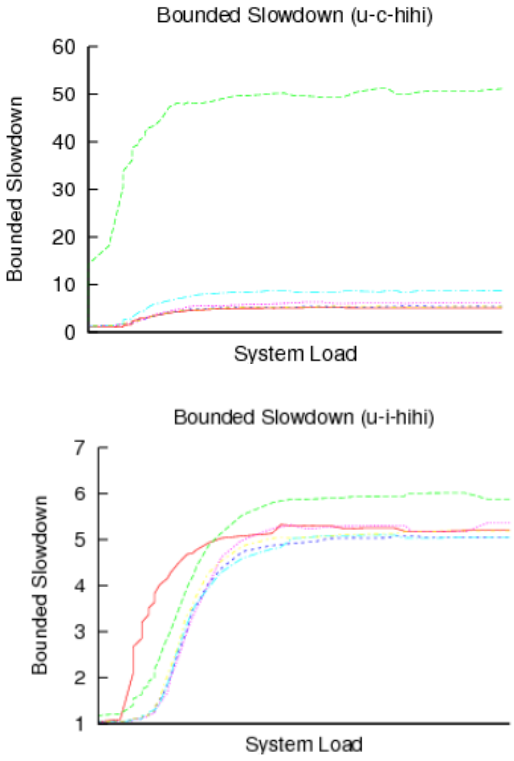

Figure 5. Bounded Slowdown Values. (u_i_hihi is an exceptional case.)

6. Observation

We found that response times provide greater disparity between the different methods at high system loads (fig. 5). Overall, in terms of response times, MET performs poorest for consistent instances, OLB performs poorly for inconsistent ones, while MCT and Modified MCT perform best. However the resource utilization of both MET and MCT is low (figure 3).

Resource utilization and load balancing are important metrics in a grid environment. The Modified MCT algorithm achieves much better resource utilization than all other methods (except OLB) at the cost of slightly degraded response time.

The SA and kPB methods provide response times and utilization values between those of MCT and MET. This is particularly evident when we consider the average waiting, execution and completion time of all the methods for a single instance as shown in figure 4. In this particular case, MET achieves best execution times, but worst waiting times for jobs. MCT provides the best completion time. SA and kPB have completion times higher than MCT but lower than MET.

The bounded slowdown values are the worst for MET in all cases but are more pronounced for consistent instances. With this metric, OLB is the best,

except for u_i_hihi where where it gives a higher slowdown than MCT(fig. 5).

7. Conclusions and future work

In this work, we have examined a set of methods for dynamic scheduling of jobs on an open online heterogeneous system, with the unique characteristic that they schedule jobs as soon as they arrive. We have examined most of the immediate mode methods from the literature and have tested them on the benchmark provided by Braun et al[5]. The results of our simulation show that none of the methods perform best in all the evaluated metrics. Their performance depends on the machine and job heterogeneity and system load. In the future, we plan to test these methods using a simulation model derived from the queue traces of existing heterogeneous systems. We also plan to improve our scheduling method to optimize different metrics for different jobs: *bounded slowdown* for short jobs and *response time* for long jobs.

8. References

[1] I. Foster and C. Kesselman., "The Grid - Blueprint for a New Computing Infrastructure",*Morgan Kaufmann*, 1998.

[2] R. Buyya., "Economic-based Distributed Resource Management and Scheduling for Grid Computing" *Phd thesis*, Monash University, Melbourne,Australia, 2002.

[3] H. Casanova and J. Dongarra, "NetSolve: Network enabled solvers.", *IEEE Computational Science and Engineering*, 1998, pp.55-67.

[4] V. D. Martino and M. Mililotti, "Sub optimal scheduling in a grid using genetic algorithms", *Parallel Computing*, 2004, pp. 553-565.

[5] T. Braun, H. Siegel, N. Beck, L. Boloni, M. Maheswaran, A. Reuther., J. Robertson, M. 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*, 2001, pp 810-837.

[6] F. Xhafa, L. Barolli, A. Durresi, "Immediate mode scheduling of independent jobs on computational grids", *21st International conference on Advanced Networking and Applications*, 2007.

[7] Dror. G. Feitelson, L. Rudolph, "Metrics and Benchmarking for Parallel Job Scheduling", *Proceedings of the workshop on Job Scheduling Stragetis for Parallel Processing*, 1998.

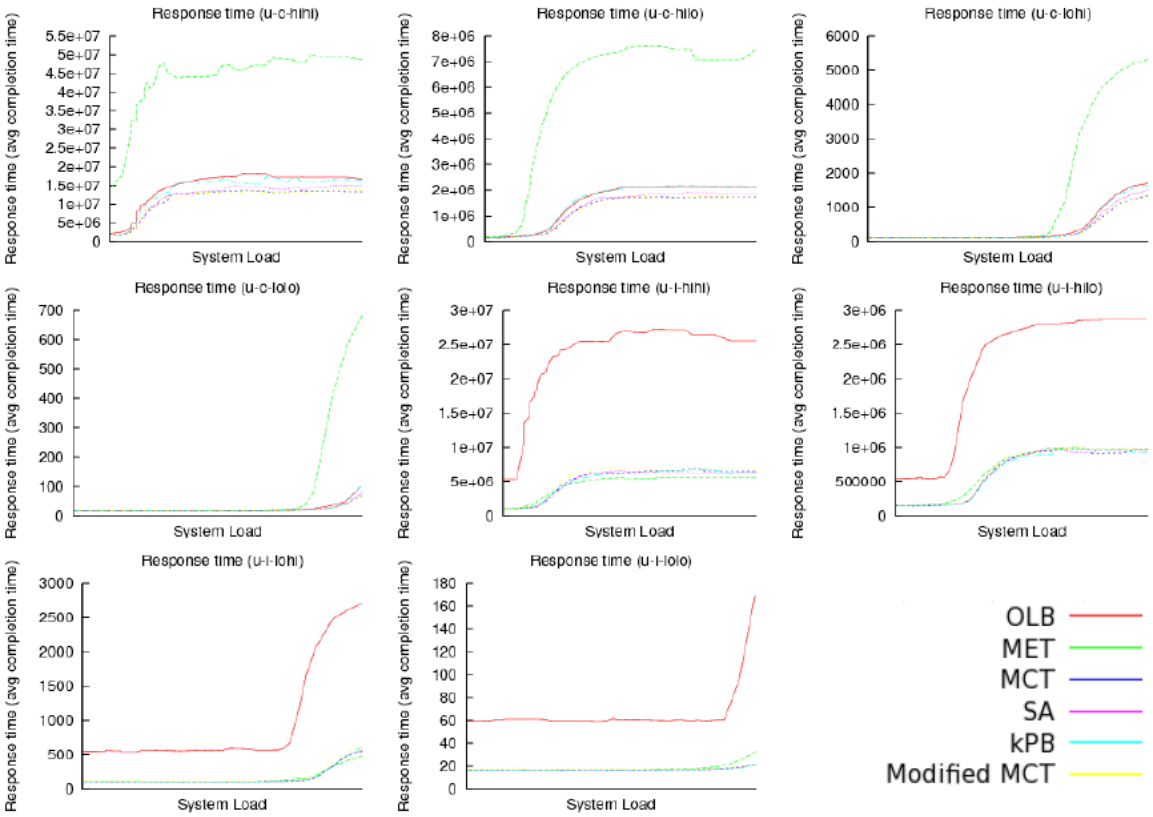

Figure 6. Response time as a function of System Load