Managing Risk of Inaccurate Runtime Estimates for Deadline Constrained Job Admission Control in Clusters

Chee Shin Yeo and Rajkumar Buyya
Grid Computing and Distributed Systems Laboratory
Department of Computer Science and Software Engineering
The University of Melbourne
VIC 3010, Australia
{*csyeo, raj*}*@csse.unimelb.edu.au*

Abstract

The advent of service-oriented Grid computing has resulted in the need for Grid resources such as clusters to enforce user-specific service needs and expectations. Service Level Agreements (SLAs) define conditions which a cluster needs to fulfill for various jobs. An example of SLA requirement is the deadline by which a job has to be completed. In addition, these clusters implement job admission control so that overall service performance does not deteriorate due to accepting exceeding amount of jobs. However, their effectiveness is highly dependent on accurate runtime estimates of jobs. This paper thus examines the adverse impact of inaccurate runtime estimates for deadline constrained job admission control in clusters using the Earliest Deadline First (EDF) strategy and a deadline-based proportional processor share strategy called Libra. Simulation results show that an enhancement called LibraRisk can manage the risk of inaccurate runtime estimates better than EDF and Libra by considering the risk of deadline delay.

1 Introduction

Commercial vendors are now rapidly deploying and providing service-oriented computing using Grid technologies [3]. Since the majority of Grid resources are *clusters* [10], there is a need for *cluster Resource Management Systems (RMSs)* to distinguish and enforce user-specific service requests and demands. The cluster RMS has to fulfill *Service Level Agreements (SLAs)* with precise *Quality of Service (QoS)* requirements that have been negotiated and agreed upon between the cluster and various users.

Recent studies [14][5][6][12] have shown that implementing job admission control is essential to support service-oriented computing since a cluster has limited resources and cannot meet unlimited demand of resources from all users. A job admission control accepts a limited number of jobs so that the overall service performance of the cluster does not deteriorate. It prevents workload overload and ensures that jobs submitted earlier do not delay jobs submitted later.

Our work focuses on job admission control in clusters based on a common SLA requirement – to complete a job within its user-specified deadline. In this case, the deadline constrained job admission control needs accurate runtime estimates of jobs to prioritize jobs effectively. Given that user runtime estimates are rather inaccurate [9][17], this paper examines how this inaccuracy can affect deadline constrained job admission control in clusters.

The first contribution is developing an enhanced deadline constrained job admission control called LibraRisk that determines whether accepting a new job will expose the risk of deadline delay in the cluster. The second contribution is conducting comprehensive performance analysis of LibraRisk and other deadline constrained job admission controls (EDF and Libra) using trace-based simulation. Comparisons cover various scenarios that includes varying workload, deadline high:low ratio, high urgency jobs, and inaccurate runtime estimates. Experiment results shows that the actual runtime estimates from traces of supercomputer centers are indeed highly inaccurate and often over estimated. However, LibraRisk is still able to perform significantly better, thus highlighting its effectiveness in managing the risk of inaccurate runtime estimates.

Section 2 discusses related work. Section 3 describes how LibraRisk manages the risk for deadline constrained job admission control thru a deadline delay metric. Section 4 describes the experimental methodology for performance evaluation. Section 5 compares the performance of LibraRisk with EDF and Libra under various scenarios inaccurate runtime estimates. Section 6 concludes.

2 Related Work

Existing cluster RMSs [18][4][11][2][16] neither consider SLAs for resource allocation nor implement job admission control, and thus are not able to enable service-oriented computing. So, several works [14][5][6][12] have proposed implementing job admission control to support service-oriented computing.

In [5] and [12], their job admission controls aim to improve the overall utility or profit of service providers and do not consider the deadline QoS in their service requirements. Both Libra [14] and QoPS [6] use deadline constrained job admission control to fulfill the deadline QoS of jobs. Libra enforces hard deadlines of jobs, while QoPS allows soft deadlines by defining a slack factor for each job so that earlier jobs can be delayed up to the slack factor if necessary to accommodate later more urgent jobs. In this paper, we focus on enforcing hard deadlines and thus enhance Libra to manage the risk of inaccurate runtime estimates.

Various works [7][8][5][12] have addressed some form of risk in computing jobs. In [5] and [12], job admission control minimizes the risk of paying penalties to compensate users that will reduce the profit of service providers. Computation-at-Risk (CaR) [7][8] determines the risk of completing jobs later than expected based on either the makespan (response time) or the expansion factor (slowdown) of all jobs in the cluster. In contrast, our work examines the risk of inaccurate runtime estimates on job admission control to enforce the deadline QoS of jobs.

Some other works [9][13][17] have studied how inaccurate runtime estimates affect job scheduling in general, but not in deadline constrained job admission controls for service-oriented computing.

3 Managing Risk for Deadline Constrained Job Admission Control

We consider deadline constrained job admission control in a cluster under the following scenario:

- The SLA requirements of a job do not change once the job has been accepted by the job admission control. This work focuses on one SLA requirement which is fulfilling the specified deadlines of jobs. Users specify hard deadlines so that jobs must be completed within deadlines to be useful.
- The cluster RMS supports non-preemptive job scheduling and provides the only single interface for users to submit jobs in the cluster. So, it is aware of all workloads in the cluster.
- Each job i has a runtime $runtime_i$ which is the time period required to complete job i if it is allocated the full proportion of processing power on a node. The runtime of a job does not include any waiting time and communication latency, and may be expressed as job length in million instructions (MI). As the runtime of a job varies between heterogeneous nodes, the runtime estimate of a job has to be translated to its equivalent value across heterogeneous nodes. Each job i also requires a minimum $numproc_i$ number of processors for execution.

In this section, we first introduce the deadline-based proportional processor share strategy with job admission control called Libra. Then, we propose how the risk of deadline delay can be modeled and incorporated into the enhanced version of Libra called LibraRisk to manage the risk of deadline delay.

3.1 Libra: Deadline-based Proportional Processor Share

Libra [14] implements deadline-based proportional processor share so that more jobs can be accepted, with allocated processor time shares spread across deadlines of jobs. Given that a job i still has an expected remaining runtime $remaining\_runtime_{ij}$ (initially its runtime $runtime_i$) on node j and has to be completed within a remaining deadline $remaining\_deadline_i$ (initially its deadline $deadline_i$), the minimum processor time share $share_{ij}$ required is:

$$share_{ij} = \frac{remaining\_runtime_{ij}}{remaining\_deadline_i} \quad (1)$$

Node j thus needs to have the total processor time share $total\_share_j$ to fulfill the deadlines of all its n_j jobs:

$$total\_share_j = \sum_{i=1}^{n_j} share_{ij} \quad (2)$$

So, in Libra, a new job is only accepted into node j if node j has at least $total\_share_j$ (including the new job) of processor time available. Otherwise, all the jobs on node j will encounter delays. If accepted, the new job starts execution immediately based on its allocated share $share_{ij}$.

3.2 Modeling Risk of Deadline Delay

Given that a job i is submitted to the cluster at time $submit\_time_i$ and completed at time $finish\_time_i$, job i incurs a delay $delay_i$ if it takes longer to complete than its specified deadline $deadline_i$:

$$delay_i = (finish\_time_i - submit\_time_i) - deadline_i \quad (3)$$

Otherwise, job i has no delay (i.e. $delay_i = 0$s) and its deadline is fulfilled if it completes before the deadline.

The deadline-constrained job admission control aims to maximize the number of jobs completed within their deadlines to enforce their SLAs, i.e. ideally, all accepted jobs are to be completed without any delay. Analogous to the CaR approach [7][8], we define a *deadline delay* metric to model the impact of a job's delay on its remaining deadline:

$$deadline\_delay_i = \frac{delay_i + remaining\_deadline_i}{remaining\_deadline_i} \quad (4)$$

The minimum and best value of $deadline\_delay_i$ is 1 when job i has zero delay. $deadline\_delay_i$ has a higher impact value when $delay_i$ is longer or $remaining\_deadline_i$ is shorter. For example, job 1 with $delay_1 = 20$s and $remaining_deadline_1 = 5$s has $deadline\_delay_1 = 5$ which is higher, compared to job 2 with the same $delay_2 = 20$s and $remaining_deadline_2 = 10$s having $deadline\_delay_2 = 3$. This metric therefore discourages incurring longer job delays and violating deadlines of more urgent jobs. We can then compute the mean deadline delay μ_j of a node j that has n_j scheduled jobs:

$$\mu_j = \frac{\sum_{i=1}^{n_j} deadline\_delay_{ij}}{n_j} \quad (5)$$

Next, we determine the risk of deadline delay σ_j on node j by deriving its standard deviation:

$$\sigma_j = \sqrt{\frac{\sum_{i=1}^{n_j} \left(deadline\_delay_{ij}\right)^2}{n_j} - \left(\mu_j\right)^2} \quad (6)$$

A high risk σ_j indicates a high uncertainty of jobs on node j not to experience deadline delays. Thus, not having any risk is the most ideal (i.e. $\sigma_j = 0$).

3.3 LibraRisk: Managing Risk of Deadline Delay

We now propose an improved deadline constrained job admission control called LibraRisk to manage the risk of deadline delay. LibraRisk uses the same deadline-based proportional processor share technique as Libra to determine the processor time allocation for each job on a node. Like Libra, the job admission control of LibraRisk enforces the deadlines of previously accepted jobs on each node by rejecting a new job if there are not enough processors as required to execute it based on current workload. However, LibraRisk applies two enhancements that differ from Libra.

First, in Libra, a node j is suitable if it has at least $total\_share_j$ of processor time available, i.e. node j can meet the deadlines of all its jobs (including the new job). On the other hand, in LibraRisk, a node j is suitable to accept a new job if its risk of deadline delay σ_j is zero. Second, Libra assigns the most suitable nodes to the new job based on the best fit strategy, i.e. nodes that have the least available processor time after accepting the new job will be selected first so that nodes are saturated to their maximum. In contrast, LibraRisk only selects nodes that have zero risk of deadline delay.

Algorithm 1: Pseudo-code for job admission control of LibraRisk.

```
1  for j ← 0 to m − 1 do
2      add job new temporarily into ListJobs_j ;
3      for i ← 0 to ListJobs_j_size −1 do
4          determine delay_i;
5      endfor
6      compute σ_j;
7      remove job new from ListJobs_j ;
8      if σ_j = 0 then
9          add node j into ListZeroRiskNodes_new ;
10     endif
11 endfor
12 if ListZeroRiskNodes_new_size ≥ numproc_new then
13     for j ← 0 to numproc_new − 1 do
14         allocate job new to node j of
           ListZeroRiskNodes_new ;
15     endfor
16 else
17     reject job new;
18 endif
```

Algorithm 1 outlines how LibraRisk works. Given that there are m computation nodes in the cluster, LibraRisk first determines the delay that will be incurred on all jobs (including accepted jobs and the new job new) on each node j if job new is scheduled on it (line 2–5). The delay of a job on node j is determined based on the deadlines and runtime estimates of all the jobs on node j as explained in Sections 3.1 and 3.2 (line 4). LibraRisk then computes the risk of deadline delay σ_j for each node j (line 6). A node is suitable if it has zero risk after accepting job new (line 8–10). Job new is finally accepted and allocated to these suitable nodes if there is $numproc_{new}$ number of suitable nodes as required by job new; else it is rejected (line 12–18).

4 Experimental Methodology

We use GridSim [15], an event-based simulator to run the experiments. The experiments use a subset of the last 3000 jobs in the SDSC SP2 trace (April 1998 to April 2002) version 2.2 from Feitelson's Parallel Workload Archive [1].

The SDSC SP2 trace is chosen as most other traces contain no information about actual runtime estimates of jobs. In addition, it has the highest resource utilization of 83.2% among other traces and thus ideally model the heavy workload scenario where having a job admission control can prevent overloading of the cluster. This 3000 job subset which represents about 2.5 months of the original trace has an average inter arrival time of 2131 seconds (35.52 minutes) and average runtime of 8880 seconds (2.47 hours), and requires

an average of 17 processors. The IBM SP2 located at San Diego Supercomputer Center (SDSC) has 128 computation nodes, each with a SPEC rating of 168. Unfortunately, the workload trace does not contain any information about the deadlines specified for each job. Therefore, we follow a similar experimental methodology in [5] to model deadline values for the workload.

By default, 20% of the jobs belong to a *high urgency* job class with a deadline of low $deadline_i/runtime_i$ value, while 80% of the jobs belong to a *low urgency* job class with a deadline of high $deadline_i/runtime_i$ value.

The *deadline high:low ratio* is the ratio of the means for high $deadline_i/runtime_i$ and low $deadline_i/runtime_i$ and its default value is 4. For instance, a deadline high:low ratio of 8 signifies that the $deadline_i/runtime_i$ mean of low urgency jobs is double than that of a deadline high:low ratio of 4. In other words, a higher deadline high:low ratio indicates that low urgency jobs have longer deadlines as compared to a lower ratio. The mean for low $deadline_i/runtime_i$ is 4 and values are normally distributed within each high and low $deadline_i/runtime_i$. The deadline of a job is thus always assigned a higher factored value based on the real runtime of a job. The arrival sequence of jobs from the high urgency and low urgency job classes is randomly distributed.

The *arrival delay factor* sets the arrival delay of jobs based on the inter arrival time available from the trace and its default value is 1. For example, an arrival delay factor of 0.1 denotes that a job with 600 seconds of inter arrival time from the trace now has a simulated inter arrival time of 60 seconds. Thus, a lower delay factor represents higher workload by shortening the inter arrival time of jobs.

The *inaccuracy* of runtime estimates is measured with respect to the actual runtime estimates of jobs obtained from the trace. An inaccuracy of 100% is equivalent to the actual runtime estimates from the trace, whereas an inaccuracy of 0% assumes runtime estimates are accurate and equal to the real runtimes of the jobs.

We also implement a non-preemptive Earliest Deadline First (EDF) strategy to facilitate comparison with Libra and LibraRisk. EDF selects the job with the earliest deadline to execute first. Unlike Libra and LibraRisk, EDF executes only a single job on a processor at any time (i.e. space-shared) and maintains a queue to store incoming jobs. So, EDF needs to wait for the requested number of processors for a selected job to be available, but can thus reselect a new job with an earlier deadline that arrives later during the waiting phase to improve its selection choice.

We find that EDF without job admission control performs much worse as compared to with job admission control, especially when deadlines of jobs are short. But, unlike Libra and LibraRisk, we incorporate a more relaxed job admission control for EDF where a job is not rejected immediately during job submission. Instead, EDF only rejects a selected job prior to execution if its deadline has expired or its deadline cannot be met based on its runtime estimate. Therefore, EDF has two advantages over Libra and LibraRisk: better selection choice and more generous job admission control.

5 Performance Evaluation

First, we assess the job admission controls based on three different scenarios: (i) varying workload (Section 5.2), (ii) varying deadline high:low ratio (Section 5.3), and (iii) varying high urgency jobs (Section 5.4). For each scenario, we compare how various job admission controls perform for both accurate runtime estimates and actual runtime estimates from the trace. The accurate runtime estimates shows the ideal performance of each control and how they differ with actual runtime estimates from the trace. Then, we evaluate based on varying inaccurate runtime estimate for both 20% and 80% of high urgency jobs (Section 5.5). This will demonstrate whether LibraRisk can manage the risk of inaccurate runtime estimates more effectively.

We examine the performance of each scenario based on two metrics: (i) percentage of jobs with deadlines fulfilled, and (ii) average slowdown. The percentage of jobs with deadlines fulfilled is the number of jobs that are completed within their specified deadlines, out of the total number of jobs submitted. The slowdown of a job is the ratio of its response time and minimum runtime required, where the response time is the time taken for a job to be completed after it is submitted and includes waiting time. Since the emphasis of this work is to meet the deadlines specified for jobs, the average slowdown is computed only for jobs with deadlines fulfilled.

5.1 Overview: Varying Workload, Deadline High:Low Ratio, and High Urgency Jobs

This overview explains similar results that are observed for all three scenarios (Figures 1, 2, and 3). Overall, more jobs are completed with their deadlines fulfilled for the ideal case of accurate runtime estimates than the case of using actual runtime estimates from the trace. This is because in practice, runtime estimates are often overestimated during job submission to reduce the possibility of jobs being terminated due to inadequate request of runtime. So, all three job admission controls will actually accept fewer jobs than expected, leading to fewer jobs with deadlines fulfilled.

With the assumption of accurate runtime estimates, Libra is able to fulfill more jobs within their deadlines than EDF (Figures 1(a), 2(a), and 3(a)). One reason is that Libra ensures that the deadline of a job can be fulfilled based on its runtime estimate before accepting it. Another reason

is that Libra adopts the best fit strategy so that nodes are saturated to their maximum to accommodate more jobs.

But, we can see that Libra only performs barely better than EDF with the use of actual runtime estimates from the trace (Figures 1(b), 2(b), and 3(b)). This exposes the core weakness in Libra as it relies heavily on the idealistic assumption of accurate runtime estimates. In contrast, LibraRisk can fulfill many more jobs than Libra for actual runtime estimates from the trace (Figures 1(b), 2(b), and 3(b)), while fulfilling as many jobs as Libra for accurate runtime estimates (Figures 1(a), 2(a), and 3(a)).

We now compare the results for average slowdown. Both Libra and LibraRisk incur the same average slowdown for accurate runtime estimates (Figures 1(c), 2(c), and 3(c)). However, LibraRisk achieves lower average slowdown than Libra for actual runtime estimates from the trace (Figures 1(d), 2(d), and 3(d)). EDF has the lowest average slowdown that remains unchanged for both cases because in all our experiments, the deadline of a job is always set as a factor higher than its runtime ($deadline_i/runtime_i$). Thus, EDF is implicitly executing smaller jobs first based on the deadlines. This is also why the average slowdown of EDF only marginally increases for increasing deadline high:low ratio (Figures 2(c) and 2(d)) and slightly drops for increasing number of high urgency jobs (Figures 3(c) and 3(d)).

Therefore, LibraRisk is not only able to perform comparatively well as Libra based on accurate runtime estimates, but is also capable of fulfilling many more jobs, and achieve lower average slowdown than Libra based on actual runtime estimates from the trace. This highlights the importance of considering the risk of deadline delay and clearly illustrates the effectiveness of LibraRisk in doing so.

5.2 Varying Workload

In Figure 1, we vary the arrival delay factor (from 0 to 1) to depict decreasing workload. As the workload decreases, increasing number of jobs are completed with deadlines fulfilled (Figures 1(a) and 1(b)), with decreasing (improving) average slowdown (Figures 1(c) and 1(d)). When the workload is heavy (arrival delay factor < 0.3), EDF fulfills more jobs within their deadlines than Libra and LibraRisk (Figures 1(a) and 1(b)), but fulfills less jobs as the workload decreases (arrival delay factor > 0.3).

Unlike Libra and LibraRisk that determine whether to accept or reject a job immediately during job submission, EDF maintains a queue to store incoming jobs and thus do not reject jobs instantaneously. So, as EDF waits for the requested number of processors to be available for a currently selected job, it can reselect a new job with an earlier deadline that arrives during the waiting period. Thus, EDF has a more favorable selection choice during heavy workload as more jobs arrive and improve the selection choice leading to EDF fulfilling more jobs than Libra and LibraRisk. However, this unfair advantage diminishes when the arrival delay factor is more than 0.3 as the selection choice decreases with increasing arrival delay factor.

With actual runtime estimates from the trace, LibraRisk progressively completes an increasing higher number of jobs with deadlines fulfilled (Figure 1(b)), with decreasing lower average slowdown (Figure 1(d)) than Libra as the workload decreases (arrival delay factor > 0.5).

5.3 Varying Deadline High:Low Ratio

Figure 2 shows the impact of increasing deadlines for low urgency jobs (deadline high:low ratio from 1 to 10). With increasing deadlines, more jobs have their deadlines fulfilled (Figures 2(a) and 2(b)), but the average slowdown (Figures 2(c) and 2(d)) rises as longer jobs are accepted.

For the case of using actual runtime estimates from the trace, LibraRisk finishes more jobs with deadlines fulfilled (Figure 2(b)) than Libra even though the improvement is higher when the deadline high:low ratio is low (< 4). LibraRisk also attains a gradually improving lower average slowdown than Libra as deadlines increase (Figure 2(d)).

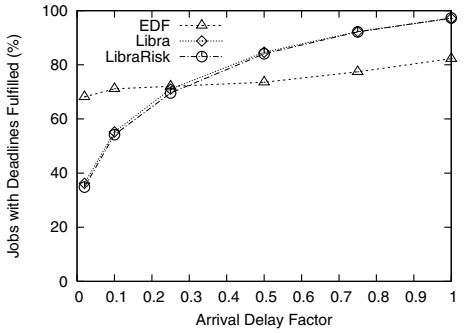
5.4 Varying High Urgency Jobs

Figure 3 shows the performance for various proportions of high urgency jobs (from 0% to 100%). As the number of high urgency jobs increases, fewer jobs are being accepted and completed, leading to decreasing number of jobs with deadlines fulfilled (Figures 3(a) and 3(b)) and decreasing average slowdown (Figures 3(c) and 3(d)).

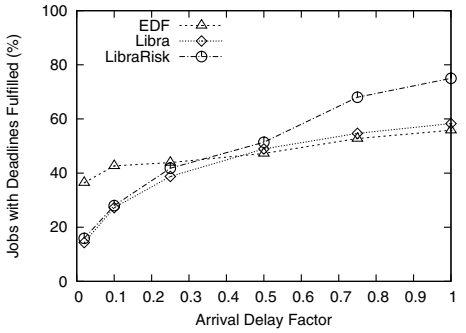
With actual runtime estimates from the trace, LibraRisk completes close to 40% more jobs than Libra where there is 100% high urgency jobs (Figure 3(b)), which is twice better than the 20% improvement over Libra for 0% high urgency jobs. LibraRisk also satisfies increasing number of jobs within their deadlines as the number of high urgency jobs increases, whereas both EDF and Libra satisfy decreasing number of jobs. So, LibraRisk can meet more jobs with shorter deadlines. It also maintains an improvement over Libra for average slowdown (Figure 3(d)).

5.5 Varying Inaccurate Runtime Estimates

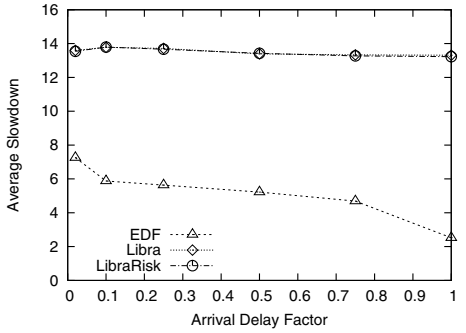
Figure 4 compares the difference between executing jobs with 20% and 80% of high urgency jobs in terms of varying inaccurate runtime estimates (from 0% to 100%). Generally, more jobs are completed with their deadlines fulfilled for 20% of high urgency jobs as compared to 80% of high urgency jobs since having more high urgency jobs results in fewer jobs being fulfilled. As such, the average slowdown is also lower (better) when there is 80% of high urgency jobs.



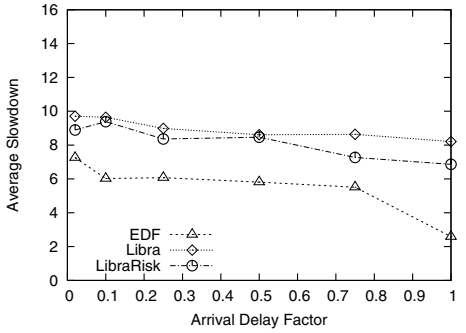
(a) Accurate runtime estimate



(b) Actual runtime estimate from trace

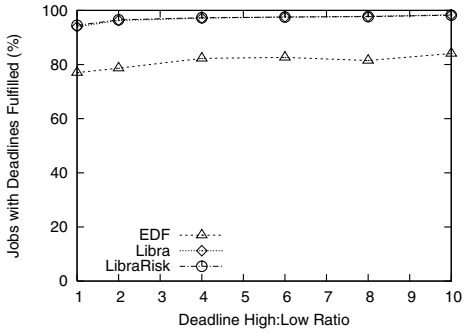


(c) Accurate runtime estimate

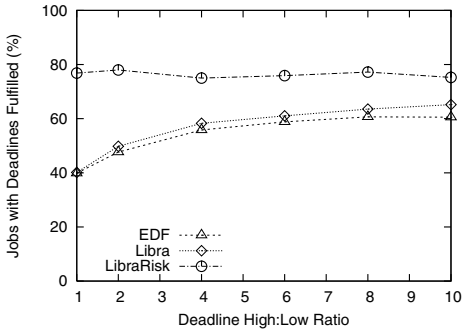


(d) Actual runtime estimate from trace

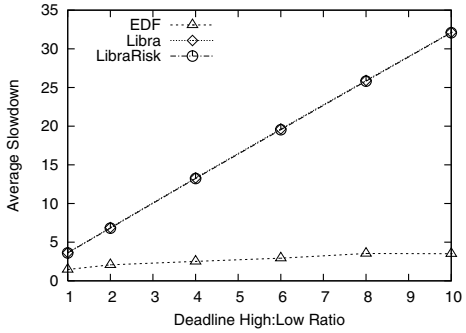
Figure 1. Impact of varying workload.



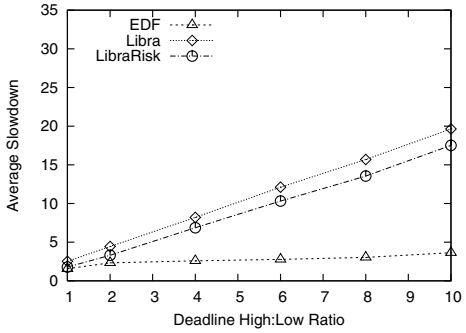
(a) Accurate runtime estimate



(b) Actual runtime estimate from trace

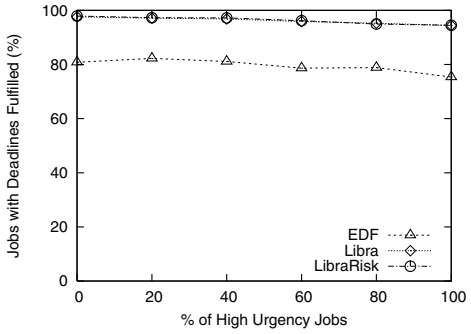


(c) Accurate runtime estimate

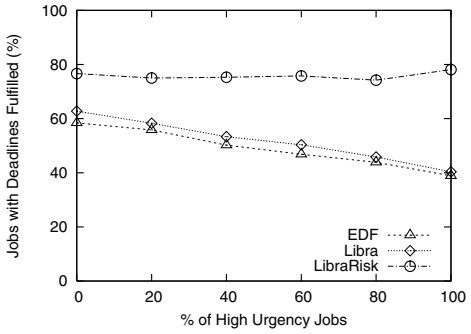


(d) Actual runtime estimate from trace

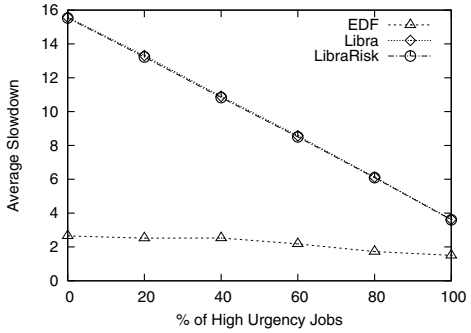
Figure 2. Impact of varying deadline high:low ratio



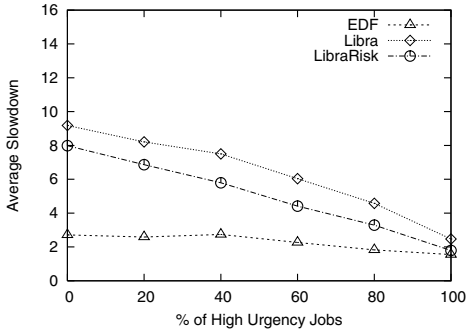
(a) Accurate runtime estimate



(b) Actual runtime estimate from trace

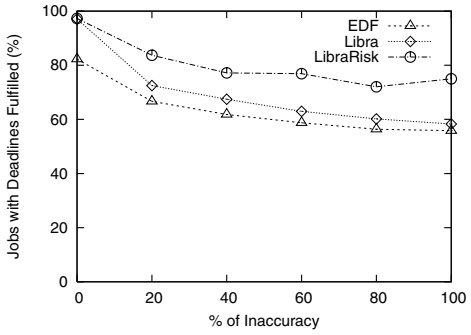


(c) Accurate runtime estimate

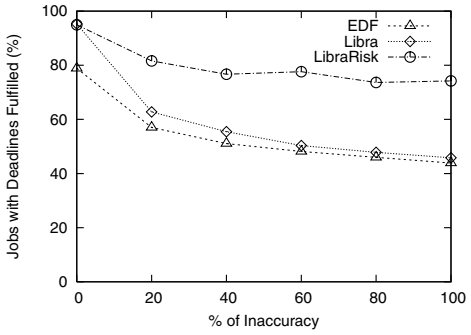


(d) Actual runtime estimate from trace

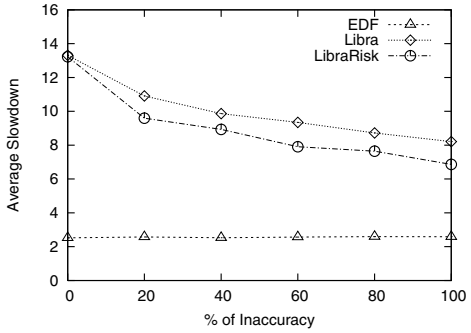
Figure 3. Impact of varying high urgency jobs.



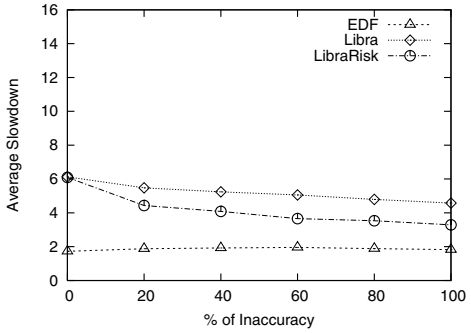
(a) 20% of high urgency jobs



(b) 80% of high urgency jobs



(c) 20% of high urgency jobs



(d) 80% of high urgency jobs

Figure 4. Impact of varying inaccurate runtime estimates.

In Figures 4(a) and 4(b), decreasing number of jobs are completed with deadlines fulfilled as the inaccuracy of runtime estimates increases. But, LibraRisk fulfills higher number of jobs than both EDF and Libra. We can see that LibraRisk completes much more (about twice as many) jobs when there is 80% of high urgency jobs as compared to 20% high urgency jobs. In fact, LibraRisk still maintains similar number of jobs with deadlines fulfilled for the case of 80% high urgency jobs (Figure 4(b)) as that of 20% high urgency jobs (Figure 4(a)), while both EDF and Libra experience drops in numbers.

Figures 4(c) and 4(d) shows that the average slowdown decreases for both Libra and LibraRisk as the inaccuracy of runtime estimates increases. EDF has the lowest average slowdown that remains the same for both cases as it also executes smaller jobs first based on their deadlines. Otherwise, as the inaccuracy of runtime estimates increases, LibraRisk maintains similar improvement in average slowdown over Libra when there is 20% or 80% of high urgency jobs.

In short, LibraRisk is able to fulfill more jobs for higher inaccuracies of runtime estimates as compared to Libra, especially when there are more high urgency jobs with shorter deadlines. This thus demonstrates its effectiveness in handling the risk of inaccurate runtime estimates.

6 Conclusion

This paper reveals that deadline constrained job admission controls perform worse than expected when using actual runtime estimates from traces of supercomputer centers because they rely on accurate runtime estimates, whereas actual runtime estimates are inaccurate and often over estimated. Therefore, we propose an enhanced deadline constrained job admission control called LibraRisk that can manage the risk of inaccurate runtime estimates more effectively via a deadline delay metric. Simulation results show that LibraRisk completes the most number of jobs with deadline fulfilled as compared to EDF and Libra in three cases when: (i) the cluster workload is lower, (ii) the deadlines of jobs are shorter (more urgent), or (iii) the runtime estimates are less accurate. LibraRisk also achieves considerably lower average slowdown than Libra. This work has thus addressed the importance of managing the risk of inaccurate runtime estimates for deadline constrained job admission control in clusters to support service-oriented computing.

Acknowledgments

We thank Anthony Sulistio for his support with the use of GridSim and the anonymous reviewers for their comments.

References

[1] Parallel Workloads Archive, http://www.cs.huji.ac.il/labs/parallel/workload, May 2005.

[2] Altair Grid Technologies. *OpenPBS Release 2.3 Administrator Guide*, Aug. 2000.

[3] I. Foster and C. Kesselman, editors. *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, San Francisco, CA, 2003.

[4] IBM. *LoadLeveler for AIX 5L Version 3.2 Using and Administering*, Oct. 2003.

[5] D. E. Irwin, L. E. Grit, and J. S. Chase. Balancing Risk and Reward in a Market-based Task Service. In *13th International Symposium on High Performance Distributed Computing (HPDC13)*, Honolulu, HI, June 2004.

[6] M. Islam, P. Balaji, P. Sadayappan, and D. K. Panda. Towards Provision of Quality of Service Guarantees in Job Scheduling. In *6th International Conference on Cluster Computing (Cluster 2004)*, San Diego, CA, Sept. 2004.

[7] S. D. Kleban and S. H. Clearwater. Computation-at-Risk: Assessing Job Portfolio Management Risk on Clusters. In *18th International Parallel and Distributed Processing Symposium (IPDPS 2004)*, Santa Fe, NM, Apr. 2004.

[8] S. D. Kleban and S. H. Clearwater. Computation-at-Risk: Employing the Grid for Computational Risk Management. In *6th International Conference on Cluster Computing (Cluster 2004)*, San Diego, CA, Sept. 2004.

[9] A. W. Mu'alem and D. G. Feitelson. Utilization, Predictability, Workloads, and User Runtime Estimates in Scheduling the IBM SP2 with Backfilling. *IEEE Transactions on Parallel and Distributed Systems*, 12(6):529–543, June 2001.

[10] G. F. Pfister. *In Search of Clusters*. Prentice Hall PTR, Upper Saddle River, NJ, second edition, 1998.

[11] Platform Computing. *LSF Version 4.1 Administrator's Guide*, 2001.

[12] F. I. Popovici and J. Wilkes. Profitable services in an uncertain world. In *18th Conference on Supercomputing (SC 2005)*, Seattle, WA, Nov. 2005.

[13] G. Sabin, G. Kochhar, and P. Sadayappan. Job Fairness in Non-Preemptive Job Scheduling. In *33rd International Conference on Parallel Processing (ICPP 2004)*, Montreal, Canada, Aug. 2004.

[14] J. Sherwani, N. Ali, N. Lotia, Z. Hayat, and R. Buyya. Libra: a computational economy-based job scheduling system for clusters. *Software: Practice and Experience*, 34(6):573–590, May 2004.

[15] A. Sulistio, G. Poduvaly, R. Buyya, and C.-K. Tham. Constructing A Grid Simulation with Differentiated Network Service Using GridSim. In *6th International Conference on Internet Computing (ICOMP 2005)*, Las Vegas, NV, June 2005.

[16] Sun Microsystems. *Sun ONE Grid Engine, Administration and User's Guide*, Oct. 2002.

[17] D. Tsafrir, Y. Etsion, and D. G. Feitelson. Modeling User Runtime Estimates. In *11th Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP 2005)*, Cambridge, MA, June 2005.

[18] University of Wisconsin-Madison. *Condor Version 6.7.1 Manual*, 2004.