Integrated Risk Analysis for a Commercial Computing Service

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

Utility computing has been anticipated to be the next generation of computing usage. Users have the freedom to easily switch to any commercial computing service to complete jobs whenever the need arises and simply pay only on usage, without any investment costs. A commercial computing service however has certain objectives or goals that it aims to achieve. In this paper, we identify three essential objectives for a commercial computing service: (i) meet SLA, (ii) maintain reliability, and (iii) earn profit. This leads to the problem of whether a resource management policy implemented in the commercial computing service is able to meet the required objectives or not. So, we also develop two evaluation methods that are simple and intuitive: (i) separate and (ii) integrated risk analysis to analyze the effectiveness of resource management policies in achieving the required objectives. Evaluation results based on five policies successfully demonstrate the applicability of separate and integrated risk analysis to assess policies in terms of the required objectives.

1. Introduction

The next era of computing is envisioned to employ the utility model [28], where users no longer need to invest heavily to maintain their own computing resources. Instead, users outsource jobs to dedicated commercial computing services to be completed and thus only pay for what they use whenever they want. With the advance of parallel and distributed technologies, such as cluster computing [18] and grid computing [10] that enable resource sharing across various organizations, commercial vendors such as Amazon [2], HP [3], IBM [4], and Sun Microsystems [5] are now progressing aggressively towards providing a service market that provides dynamic service delivery.

A commercial computing service is initiated to achieve certain objectives which are very different from that of a non-commercial computing service. The most distinct objective between them is that a commercial computing service aims to earn profit as Return On Investment (ROI) for running the service, since commercial computing services are businesses driven by monetary performance. Even though monetary performance is the ultimate objective, there are also other user-centric objectives. One user-centric objective is to meet the Service Level Agreement (SLA) that have been negotiated and agreed upon with the user as the user pays for the expected service to be delivered. Another user-centric objective is to maintain reliability so that users are not disappointed with the poor service quality, otherwise they will switch to other commercial competitors. These user-centric objectives are important since a commercial computing service focuses on providing value-added services for users who then in turn pay for using the services.

Our work focuses on evaluating suitable resource management policies for a commercial computing service, in particular with respect to its required objectives. As there are numerous resource management policies [16][17][13][12][20][25][27][26] available, it is non-trivial to identify the best policy that truly meets the required objectives of the commercial computing service. Therefore, the contributions for this paper are:

- Identifying three essential objectives for a commercial computing service: (i) meet SLA, (ii) maintain reliability, and (iii) earn profit.
- Developing two evaluation methods that are simple and intuitive: (i) separate and (ii) integrated risk analysis to analyze the effectiveness of resource management policies in achieving the required objectives.

- Providing comprehensive performance analysis of five policies (FCFS-BF, EDF-BF, Libra, LibraRisk, and FirstReward) thru trace-based simulation to reveal the best policy for various objectives.

Evaluation results demonstrate the applicability of separate and integrated risk analysis based on various scenarios (including varying workload, job mix, deadline, budget, and penalty) to assess policies in terms of the required objectives. These methods can thus help commercial computing service providers to identify and implement resource management policies that suit their objectives.

Section 2 discusses related work. Section 3 describes three possible objectives that a commercial computing service aims to achieve and how they can be measured. Section 4 develops two evaluation methods to facilitate the comparison of selected resource management policies in achieving the required objectives. Section 5 describes the evaluation methodology to assess these policies. Section 6 analyzes the effectiveness of these policies with regards to the required objectives. Section 7 concludes.

2. Related Work

There are many Resource Management Systems (RMS) [24][11][19][6][23] available, providing different policies to allocate jobs. However, a commercial computing service need to consider other service parameters, such as the deadline to complete the job, the budget the user will pay for its completion, and the penalty for any deadline violation. So, several new policies [21][12][13][20][25][27] have been proposed to support quality-driven computing services, such as using an admission control to selectively accept new jobs based on certain service parameters.

However, there is no work done to identify essential objectives that a commercial computing service is aiming to achieve, which is what we are addressing in this paper. We also propose separate and integrated risk analysis methods to evaluate whether policies are able to achieve the required objectives.

Various works [14][15][12][20] have addressed some form of risk in computing jobs. In [12] and [20], the risk of paying penalties to compensate users is minimized so as not to reduce the profit of service providers. Computation-at-Risk (CaR) [14][15] determines the risk of completing jobs later than expected based on either the makespan (response time) or the expansion factor (slowdown). GridIS [25] shows that a conservative provider earns much less profit due to accepting too few jobs to run, as compared to an aggressive provider who earn more profit even though more jobs result in deadline violations. In contrast, we provide a way to evaluate these policies with respect to the required objectives.

3. Objectives of a Commercial Computing Service

A commercial computing service operates based on objectives that it aims to achieve. This section explains the importance of achieving three possible objectives: (i) meet SLA, (ii) maintain reliability, and (iii) earn profit, and how they can be measured.

Since various users have different requirements and needs for running their jobs, they specify specific SLA for completing their jobs. An example of SLA requirement is the deadline within which a job needs to be completed in. Users expect their specified SLA requirements to be fulfilled as they are paying for the required service. Users who always have jobs being rejected can easily opt for other service providers, thus increasing competition and demand for SLA satisfaction. So, the relevant objective of the commercial computing service will be to meet SLA. The SLA metric can be computed as the percentage of n_{SLA} jobs with SLA fulfilled (in this case, completed within deadlines), out of the total number of m jobs submitted to the computing service:

$$SLA = \frac{n_{SLA}}{m} * 100 \tag{1}$$

With users specifing the level of service they required thru SLA, the commercial computing service needs to ensure that it can really deliver the negotiated service. Otherwise, users can simply switch to other competitors offering better service. So, another objective is to maintain reliability. The $reliability$ metric can be calculated as the percentage of n_{SLA} jobs with SLA fulfilled, out of the number of n jobs that are accepted by the computing service:

$$reliability = \frac{n_{SLA}}{n} * 100 \tag{2}$$

The most important objective for a commercial computing service is to earn profit as commercial businesses are always driven by monetary performance. The $profit$ metric can be set as the percentage of total utility earned from n jobs accepted by the computing service, out of the total maximum budget of m jobs that are submitted to the computing service:

$$profit = \frac{\sum_{i=1}^{n} utility_i}{\sum_{i=1}^{m} budget_i} * 100 \tag{3}$$

4. Risk Analysis

An effective evaluation method is essential to determine whether a commercial computing service is able to meet its objectives. This section develops two methods based on risk management techniques: (i) separate and (ii) integrated risk analysis.

4.1. Separate Risk Analysis

To evaluate an objective, we analyze the risk involved for that objective thru its corresponding metric as defined in Section 3. Risk analysis requires two parameters: (i) performance and (ii) volatility that can be computed based on the results obtained in a particular scenario. For example, in the scenario of varying workload, a total of n results can be obtained for a specific metric (eg. SLA) using each different workload, whereas the rest of the experiment settings remains the same.

Thus, we can compute performance μ_{sep} of a policy for an objective as the mean of all n results obtained in the scenario:

$$performance, \mu_{sep} = \frac{\sum_{i=1}^{n} result_i}{n} \tag{4}$$

Volatility σ_{sep} of the policy for the objective can then be derived by the standard deviation of these n results:

$$volatility, \sigma_{sep} = \sqrt{\frac{\sum_{i=1}^{n} (result_i)^2}{n} - (\mu_{sep})^2} \tag{5}$$

A higher volatility means that the results of the policy fluctuates more, thus increasing the risk that the policy does not always return the same performance under various conditions. So, given two policies with the same performance, a policy with lower volatility is preferred over the one with higher volatility.

4.2. Integrated Risk Analysis

As separate risk analysis only examines a single objective, we need to be able to examine a combination of objectives. Moreover, there is often more than one objective for a commercial computing service, so it is critical to be able to assess all these objectives in an integrated fashion.

Given that there is a total of n objectives to examine, the performance μ_{int} and volatility σ_{int} of the integrated risk analysis can be computed using the performance $\mu_{sep,i}$ and volatility $\sigma_{sep,i}$ measures from the separate risk analysis for each objective i:

$$performance, \mu_{int} = \sum_{i=1}^{n} w_i * \mu_{sep,i} \tag{6}$$

$$volatility, \sigma_{int} = \sum_{i=1}^{n} w_i * \sigma_{sep,i} \tag{7}$$

where w_i is a weight to denote the importance of a particular objective with respect to other objectives. For instance, for our experiments, we treat all objectives as equal, meaning that all w_i are the same. So, for the results in Section 6.2, w_i is 0.5 for two objectives and w_i is 0.33 for all three objectives.

Table 1. Policy parameter consideration.

Policy	Arrival time	Deadline	Budget with penalty
FCFS-BF	✓		
EDF-BF		✓	
Libra		✓	
LibraRisk		✓	
FirstReward			✓

5. Performance Evaluation

5.1. Resource Management Policies

We examine five resource management policies, namely: (i) FCFS-BF, (ii) EDF-BF, (iii) Libra, (iv) LibraRisk, and (v) FirstReward. Table 1 lists the differences between these policies thru the parameters they consider in allocating resources to jobs.

FCFS-BF and EDF-BF are backfilling policies which prioritize jobs based on arrival time (First Come First Serve) and deadline (Earliest Deadline First) respectively. Both policies adopt EASY backfilling [16][17] to increase resource utilization. A queue is used to store incoming jobs as only a single job can run on a processor at any time (i.e. space-shared). When insufficient number of processors is available for the first job (highest priority) in the queue, EASY backfilling assigns these unused processors to the next waiting jobs in the queue based on their runtime estimates, provided that they do not delay the first job. In other words, jobs that skip ahead must finish before the time when the required number of processors by the first job is expected to be available.

These two variations of EASY backfilling policy are chosen for performance evaluation because EASY backfilling is currently the most widely used policy for scheduling parallel jobs in commercial cluster batch schedulers [9]. We find that these policies without job admission control perform much worse, especially when deadlines of jobs are short. So, we implement an admission control that checks whether a job should be rejected based on two conditions before running it: (i) the job is predicted to exceed its deadline based on its runtime estimate, and (ii) the job has already exceeded its deadline while waiting in the queue. This generous admission control enables FCFS-BF and EDF-BF to select their highest priority job at the latest time, while ensuring that earlier jobs whose deadlines have lapsed do not incur propagated delay for later jobs.

Libra [21] uses deadline-based proportional processor share with job admission control to enforce the deadlines of jobs. A minimum processor time share is computed for each job i as $runtime_i/deadline_i$ using runtime estimate so that job i is accepted only if there are sufficient required number of processors with the free minimum processor time share. This means that multiple jobs can run on a processor at any time, using its allocated minimum processor time share (i.e. time-shared). Unlike the above backfilling policies, no queue is maintained so a new job is checked during submission and rejected immediately if its deadline is not expected to be fulfilled. Libra chooses suitable processors based on the best fit strategy, i.e. processors that have the least available processor time left with the new job will be selected first so that every processor is saturated to its maximum. Any remaining free processor time is then distributed among all jobs on the processor according to the processor time share of each job.

LibraRisk [27] is an improvement of Libra and uses the same deadline-based proportional processor share. The difference is that LibraRisk considers the risk of deadline delay when selecting suitable nodes for a new job. Nodes are selected for a new job only if they have zero risk of deadline delay. This enables LibraRisk to manage the risk of inaccurate runtime estimates more effectively than Libra. Given that actual runtime estimates from traces are quite inaccurate, LibraRisk is able to complete more jobs with deadline fulfilled and achieve lower average slowdown than Libra.

FirstReward [12] determines possible future earnings PV_i with possible opportunity cost penalties $cost_i$ based on estimated remaining runtime RPT_i of a job i. The reward $reward_i$ is then calculated thru a α-weighting function as: $reward_i = ((\alpha * PV_i) - ((1-\alpha) * cost_i))/RPT_i$. The earnings PV_i of a job i is computed as: $PV_i = budget_i/(1 + (discount\_rate * RPT_i))$. For unbounded penalties (as in the case of our simulation), the penalty cost $cost_i$ of a job i is the sum of penalty for all other n accepted jobs based on RPT_i: $cost_i = \sum_{j=0;j\neq i}^{n}(penalty\_rate_j * RPT_i)$. The admission control of FirstReward computes the slack $slack_i$ of a new job i during submission and rejects the job immediately if $slack_i$ is less than a specified slack threshold: $slack_i = (PV_i - cost_i)/penalty\_rate_i$. The slack threshold determines the balance of earnings and penalties where a high threshold avoids future commitments that can result in possible penalties. Setting the correct slack threshold is not trivial as the ideal slack threshold changes depending on the workload. After testing various slack threshold values for our simulated workload, we derive the following ideal simulation settings for FirstReward: α is 1, the discount rate is 1%, and the slack threshold is 25. We have also extended the FirstReward to consider multiple-processor parallel jobs for our simulation since the original one only considers single-processor jobs. However, we do not make FirstReward to support backfilling, so delays may occur due to waiting for the required number of processors.

5.2. Evaluation Methodology

Our evaluation uses a discrete event simulator called GridSim [7][22] to run the experiments. The experiments are generated from a subset of the last 5000 jobs in the SDSC SP2 trace (April 1998 to April 2000) version 2.2 from Feitelson's Parallel Workload Archive [1].

The SDSC SP2 trace is chosen because it has the highest resource utilization of 83.2% among other traces to ideally model the heavy workload scenario for a computing service. This 5000 job subset based on the last 3.75 months of the SDSC SP2 trace requires an average of 17 processors and has an average inter arrival time of 1969 seconds (32.8 minutes) and average runtime of 8671 seconds (2.4 hours). The computing service that is simulated resembles the IBM SP2 at San Diego Supercomputer Center (SDSC) with 128 computation nodes, each having a SPEC rating of 168.

However, jobs submitted to a commercial computing service need to have three other significant parameters (deadline, budget, and penalty) which is unfortunately unavailable in this trace and from an actual commercial computing service. Therefore, we adopt a similar methodology in [12] to model these parameters through two job classes: (i) high urgency and (ii) low urgency.

Each job in the *high urgency* class has a deadline of low $deadline_i/runtime_i$ value, budget of high $budget_i/f(runtime_i)$ value, and penalty of high $penalty_i/g(runtime_i)$ value. $f(runtime_i)$ and $g(runtime_i)$ are functions to represent the minimum budget and penalty that the user will quote with respect to $runtime_i$. Conversely, each job in the *low urgency* class has a deadline of high $deadline_i/runtime_i$ value, budget of low $budget_i/f(runtime_i)$ value, and penalty of low $penalty_i/g(runtime_i)$. This model is realistic since a user who submits a more urgent job to be completed within a shorter deadline is likely to offer a higher budget for the job to be finished on time and also specify a higher penalty if the job is delayed beyond its deadline. The arrival sequence of jobs from the high urgency and low urgency classes is randomly distributed.

Values are normally distributed within each of the three parameters. The ratio of the means for each parameter's high-value and low-value is thus known as the *high:low ratio*. So, a higher deadline high:low ratio indicates that low urgency jobs have longer deadlines than that of a lower ratio. For instance, a deadline high:low ratio of 8 means the $deadline_i/runtime_i$ mean of low urgency jobs is two times more than that of a deadline high:low ratio of 4. On the other hand, a higher budget or penalty high:low ratio denotes that high urgency jobs have larger budget or penalty than that of a lower ratio.

Since the deadline, budget and penalty rate of a job will now always be set as a larger factor of runtime, we in-

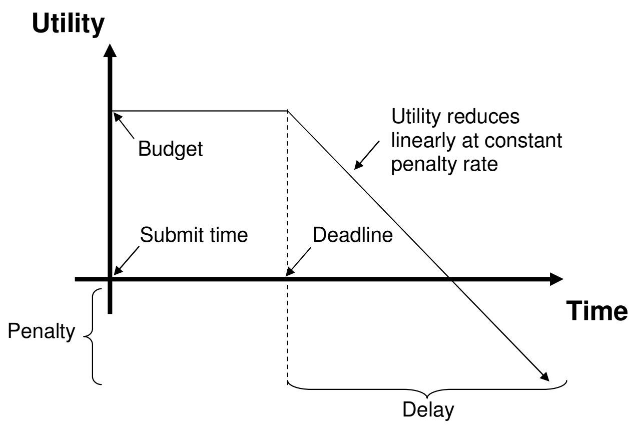


Figure 1. Impact of penalty function on utility.

troduce a *bias* parameter value. A *deadline bias* means that a job i with longer $runtime_i$ (more than the average runtime) has $deadline_i = deadline_i / deadline\_bias_i$ (i.e. shorter deadline). But if job i has shorter $runtime_i$ (less than the average runtime), then it has $deadline_i = deadline_i * deadline\_bias_i$ (i.e. longer deadline). This works likewise for budget and penalty bias.

Different levels of workload are modeled thru the *arrival delay factor* which sets the arrival delay of jobs based on the inter arrival time from the trace. For example, an arrival delay factor of 0.1 means a job with 600 seconds of inter arrival time from the trace now has a simulated inter arrival time of 60 seconds. Hence, a lower delay factor represents higher workload by shortening the inter arrival time of jobs.

For our experiments, the runtime estimates of jobs are taken from the actual runtime estimates available from the trace. A point to note is that actual runtime estimates are highly inaccurate and often over estimated.

5.3. Settings, Scenarios, and Metrics

For the experiments, we consider unbounded penalty as shown in Figure 1. The penalty function penalizes the computing service by reducing the budget of a job over time after the lapse of its deadline. For simplicity, we model the penalty function as linear, as in other previous works [8][12][20]. For every job i, the computing service earns a utility $utility_i$ depending on its penalty rate $penalty\_rate_i$ and delay $delay_i$:

$$utility_i = budget_i - (delay_i * penalty\_rate_i) \quad (8)$$

Job i has a delay $delay_i$ if it needs a longer time to complete than its given deadline $deadline_i$:

$$delay_i = (finish\_time_i - submit\_time_i) - deadline_i \quad (9)$$

Table 2. Default simulation settings.

Parameter	Default value	
	Set A	Set B
% of high urgency jobs	20	same
% of low urgency jobs	80	same
Deadline bias	1	14
Deadline high:low ratio	4	same
Deadline low mean	4	same
Budget bias	1	same
Budget high:low ratio	4	same
Budget low mean	4	same
Penalty bias	1	same
Penalty high:low ratio	4	same
Penalty low mean	4	same
Arrival delay factor	1	same

Table 3. Scenario settings.

Scenario	Varying value	
	Set A	Set B
Workload	0.02	same
(arrival delay factor)	0.10	
	0.25	
	0.50	
	0.75	
	1.00	
Job mix	0	same
(% of high urgency jobs)	20	
	40	
	60	
	80	
	100	
Deadline bias	1	10
	2	12
	4	14
	6	16
	8	18
	10	20
Budget bias	1	same
	2	
	4	
	6	
	8	
	10	
Penalty bias	1	same
	2	
	4	
	6	
	8	
	10	

where $submit\_time_i$ is the time when job i is submitted into the computing service and $finish\_time_i$ is the time when job i is completed. Thus, job i has no delay (i.e. $delay_i = 0$) if it finishes before the deadline and the computing service earns the full budget $budget_i$ as utility $utility_i$. But, if there is a delay (i.e. $delay_i > 0$), $utility_i$ drops linearly until it turns negative (i.e. exceeds $budget_i$) and becomes a penalty (i.e. $utility_i < 0$). The penalty is unbounded till the time when the job is finally completed. This model implies that a commercial computing service must be careful about accepting new jobs to ensure that too many jobs are not accepted such that heavily penalized jobs dramatically erode previously earned utility.

We use two sets of experiments: (i) Set A and (ii) Set B to better examine how each different policy performs for different experiment settings. Table 2 lists the default simulation settings for both Set A and B. The only difference is that Set B has a deadline bias of 14, whereas Set A has a deadline bias of 1. So, the aim of Set B is to minimize the advantage that FCFS-BF and EDF-BF have over other policies for Set A since jobs with the shortest runtime also has the shortest deadline and is thus always assigned first in EASY backfilling.

We first execute the policies for each of these five different scenarios: (i) varying workload, (ii) varying job mix, (iii) varying deadline bias, (iv) varying budget bias, and (v) varying penalty bias. Table 3 shows the six varying values in each scenario, thus deriving six results for a particular metric in each scenario. The three metrics (as described in Section 3) are: (i) SLA, (ii) reliability, and (iii) profit. We then apply the two proposed evaluation methods (introduced in Section 4) to assess the policies with respect to the objectives: (i) separate risk analysis (Section 6.1) and (ii) integrated risk analysis (Section 6.2).

6. Performance Results

6.1. Separate Risk Analysis

Figure 2 shows the separate risk analysis results for each objective (SLA, Reliability, and Profit) using Set A and B.

For SLA objective (Figure 2(a) and 2(b)), LibraRisk is the best policy as it returns the most number of jobs with deadline fulfilled. However, it is also more volatile compared to the other policies. EDF-BF is the least volatile policy to achieve the highest number of jobs as most deadlines are set as a larger factor based on runtime, so shortest jobs also has short deadlines. We can also notice that both Libra and LibraRisk are less volatile in Set B than Set A. This shows that Libra and LibraRisk are able to exploit changes in deadlines for better outcome.

Figure 2(c) and 2(d) shows the risk analysis for Reliability objective. Both FCFS-BF and EDF-BF has all jobs (100%) that are accepted complete within their deadlines, due to the generous admission control that we implemented. So, they are immune against the inaccuracy of runtime estimates. FirstReward is also more volatile for Set B than Set A as more shorter deadline jobs are being delayed, due to their lower penalty rates.

For Profit objective (Figure 2(e) and 2(f)), all policies experience high volatility for both Set A and B. However, performance varies greatly between Set A and B. FCFS-BF and EDF-BF achieves the highest profit in Set A, whereas LibraRisk achieves the highest profit in Set B. LibraRisk has similar profit as Libra in Set A, but is able to perform much more effectively when deadline bias is high in Set B. This proves that FCFS-BF and EDF-BF are able to achieve highest profit in Set A, largely due to the adverse unfairness in their generous admission controls.

FirstReward has the lowest performance for all three SLA, Reliability, and Profit objectives. This is possibly due to FirstReward not saturating the processors to their maximum, unlike the backfilling policies and Libra that maximizes the utilization of the processors.

6.2. Integrated Risk Analysis

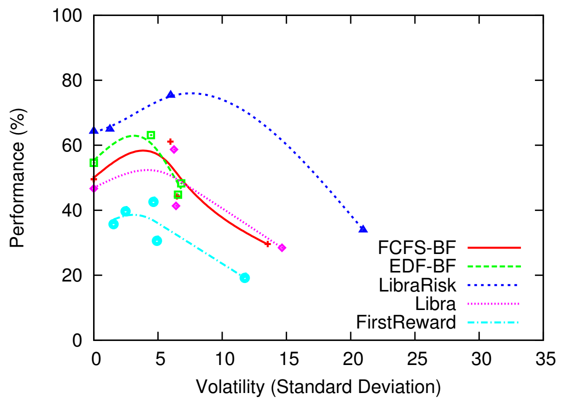
Figure 3 presents the integrated risk analysis results for each combination of two objectives using Set A and B, while Figure 4 shows the integrated risk analysis results for all three objectives.

For SLA and Reliability objectives in Figures 3(a) and 3(b), LibraRisk has the best performance and volatility for both Set A and B, while FCFS-BF, EDF-BF and Libra performs about the same.

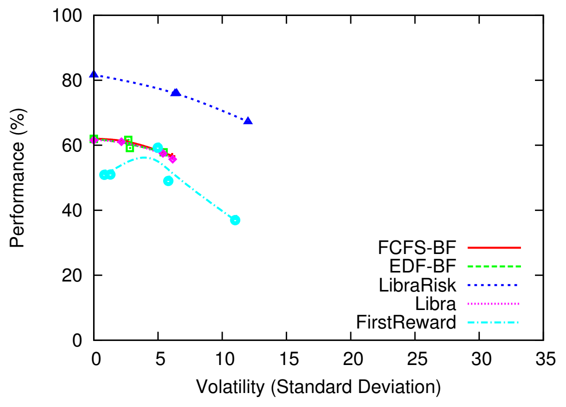
Figure 3(d) shows LibraRisk performs a lot better for SLA and Profit objectives in Set B due to its high performance in SLA objective, as seen thru a convex shape of its plot. Other policies perform not as well as seen thru the concave shapes of their plots.

For Reliability and Profit objectives in Figure 3(e) and 3(f), all the policies appear to perform somewhat similar in Set B, except LibraRisk and FirstReward having higher volatility. In Set A, FCFS-BF and EDF-BF performs best, followed by Libra and LibraRisk.

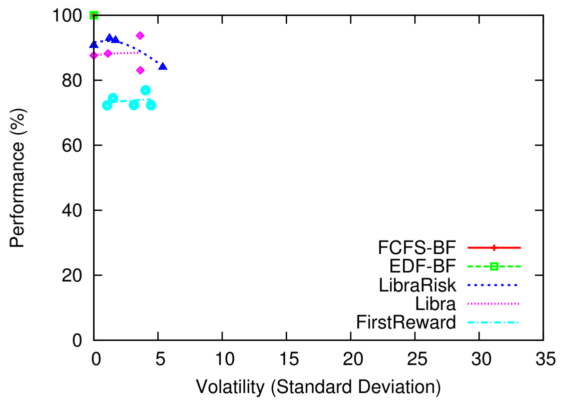
For all three objectives (SLA, Reliability, and Profit) (Figure 4(a) and 4(b)), both FCFS-BF and EDF-BF perform well in Set A, but not in Set B, highlighting the adverse advantage both policies have when deadlines are always set as larger factors of runtimes. LibraRisk is able to meet SLA and earn profit more effectively by considering the risk of deadline delay when runtime estimates are inaccurate. We can see that LibraRisk emerge as the overall best policy for Set B. LibraRisk also performs quite well for Set A with only slightly lower performance and higher volatility than FCFS-BF and EDF-BF, even though LibraRisk ob-



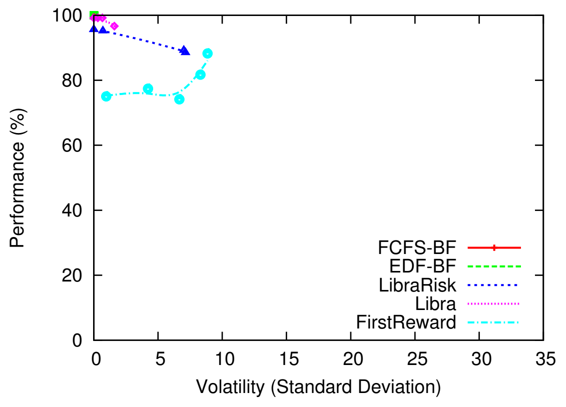
(a) Set A: SLA



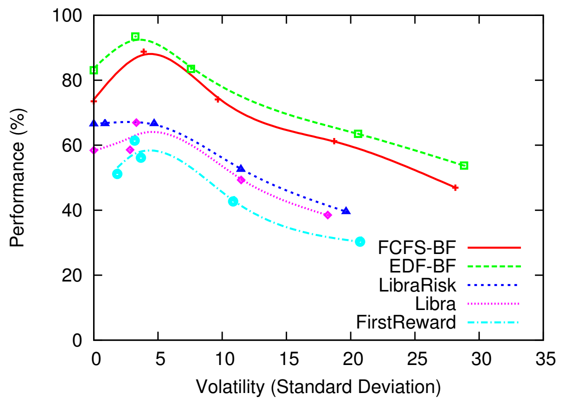
(b) Set B: SLA



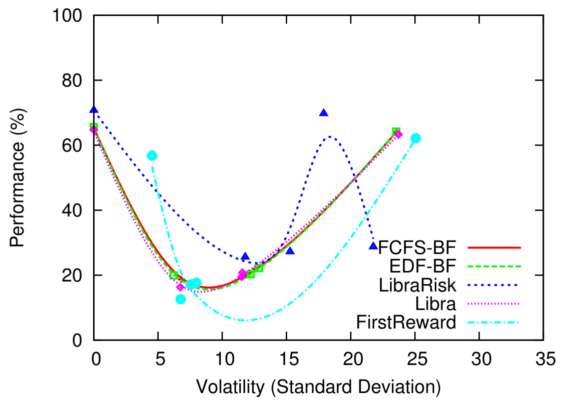
(c) Set A: Reliability



(d) Set B: Reliability

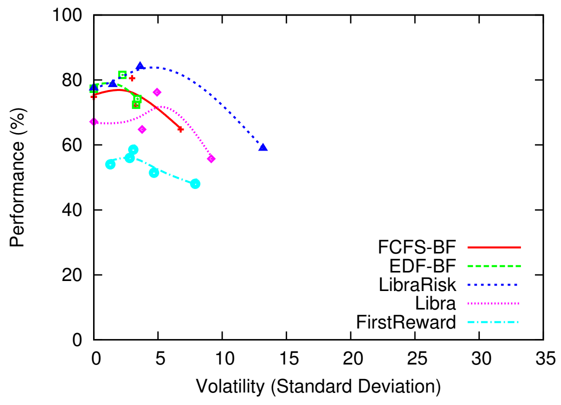


(e) Set A: Profit

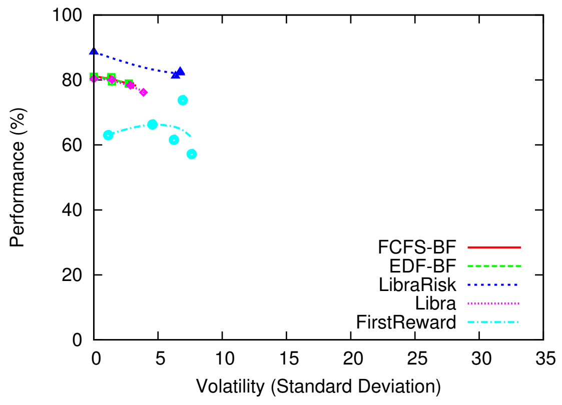


(f) Set B: Profit

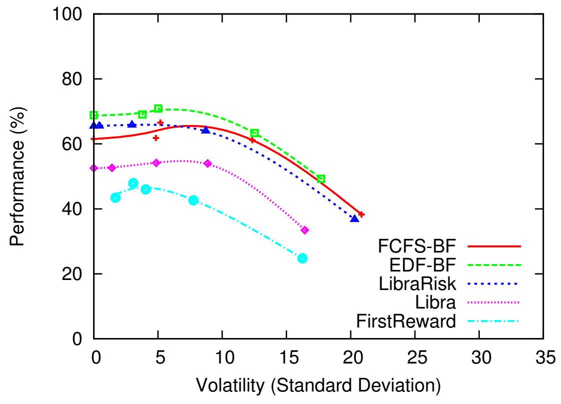
Figure 2. Separate Risk Analysis for one objective.



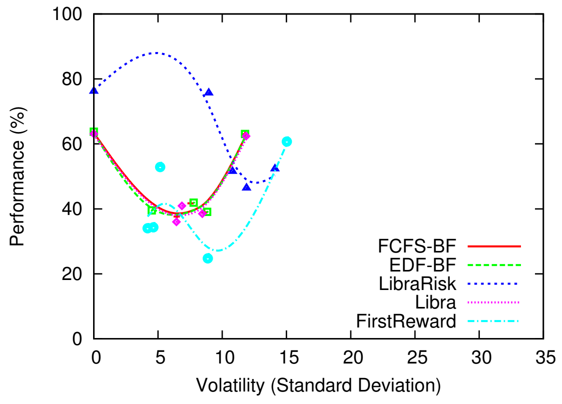
(a) Set A: SLA + Reliability



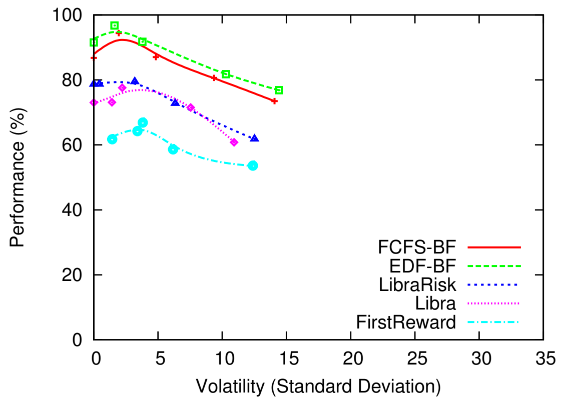
(b) Set B: SLA + Reliability



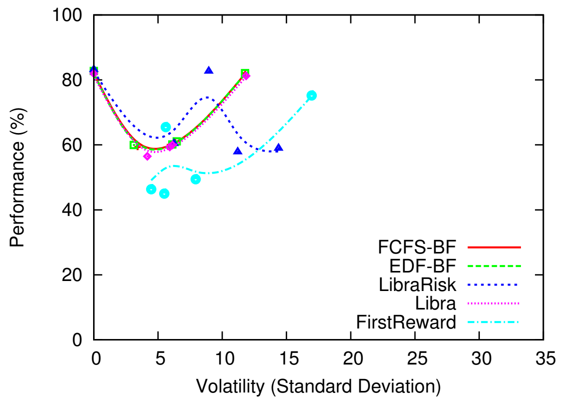
(c) Set A: SLA + Profit



(d) Set B: SLA + Profit

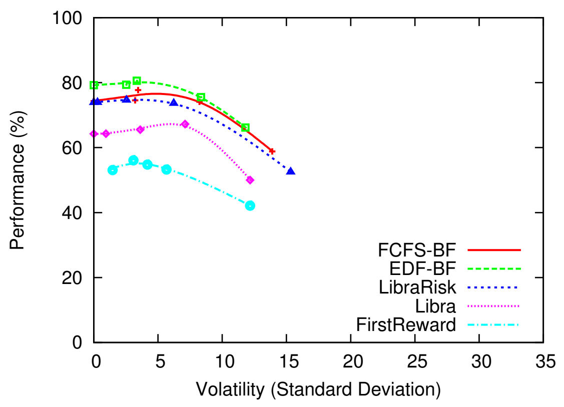


(e) Set A: Reliability + Profit

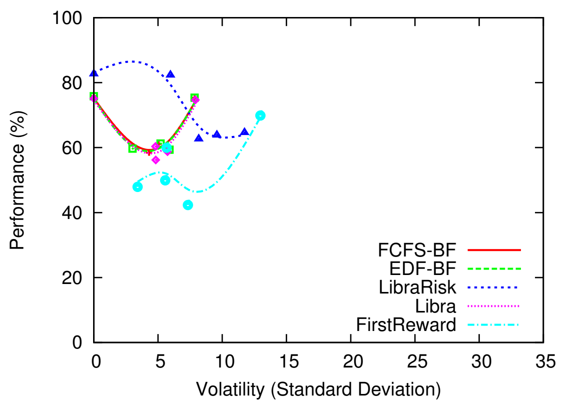


(f) Set B: Reliability + Profit

Figure 3. Integrated risk analysis for two objectives.



(a) Set A: SLA + Reliability + Profit



(b) Set B: SLA + Reliability + Profit

Figure 4. Integrated risk analysis for all three objectives.

tains about 20% lower profit than FCFS-BF and EDF-BF in Set A. This highlights the importance of considering all objectives, rather than a single objective to truly identify a resource management policy that can meet all the required objectives.

7. Conclusion

This paper discusses several important objectives that need to be considered by a commercial computing service. Two evaluation methods called separate and integrated risk analysis are then proposed. Evaluation results have shown that both separate and integrated risk analysis are able to determine how different resource management policies perform with respect to a single objective and a combination of objectives respectively. In particular, an objective that is not achieved can severely impact on the overall achievement of other objectives. This work has thus addressed the important need of identifying and analyzing the achievement of key objectives by resource management policies implemented in a commercial computing service.

Acknowledgments

We thank Anthony Sulistio for his support with the use of GridSim. This work is partially supported by a Discovery Project grant from the Australian Research Council (ARC) and an International Science Linkage grant from the Australian Department of Education, Science and Training (DEST).

References

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

[2] Amazon. *Elastic Compute Cloud (EC2)*, http://www.amazon.com/ec2.

[3] HP. *Adaptive Enterprise*, http://www.hp.com/go/adaptive.

[4] IBM. *On Demand Business*, http://www.ibm.com/ondemand.

[5] Sun Microsystems. *Sun Grid*, http://www.sun.com/service/sungrid.

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

[7] R. Buyya and M. Murshed. GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing. *Concurrency and Computation: Practice and Experience*, 14(13–15):1175–1220, Nov.–Dec. 2002.

[8] B. N. Chun and D. E. Culler. User-centric Performance Analysis of Market-based Cluster Batch Schedulers. In *Proceedings of the 2nd International Symposium on Cluster Computing and the Grid (CCGrid 2002)*, pages 22–30, Berlin, Germany, May 2002.

[9] Y. Etsion and D. Tsafrir. A Short Survey of Commercial Cluster Batch Schedulers. Technical Report 2005-13, Hebrew University, May 2005.

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

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

[12] 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.

[13] 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.

[14] 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, April 2004.

[15] 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.

[16] D. Lifka. The ANL/IBM SP Scheduling System. In *1st Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP 1995)*, Santa Barbara, CA, April 1995.

[17] 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.

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

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

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

[21] 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.

[22] 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.

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

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

[25] L. Xiao, Y. Zhu, L. M. Ni, and Z. Xu. GridIS: an Incentive-based Grid Scheduling. In *19th International Parallel and Distributed Processing Symposium (IPDPS 2005)*, Denver, CO, April 2005.

[26] C. S. Yeo and R. Buyya. A Taxonomy of Market-based Resource Management Systems for Utility-driven Cluster Computing. *Software: Practice and Experience*, 36(13):1381–1419, 10 Nov. 2006.

[27] C. S. Yeo and R. Buyya. Managing Risk of Inaccurate Runtime Estimates for Deadline Constrained Job Admission Control in Clusters. In *35th International Conference on Parallel Processing (ICPP 2006)*, Columbus, OH, Aug. 2006.

[28] C. S. Yeo, R. Buyya, M. D. de Assuncao, J. Yu, A. Sulistio, S. Venugopal, and M. Placek. Utility Computing on Global Grids. Technical Report GRIDS-TR-2006-7, Grid Computing and Distributed Systems Laboratory, The University of Melbourne, April 2006.