A Novel Approach To Allocating QoS-constrained Workflow-based Jobs In A Multi-Cluster Grid

Yash Patel
London e-Science Centre, Imperial College
South Kensington Campus
London SW7 2AZ
yp03@doc.ic.ac.uk

John Darlington
London e-Science Centre, Imperial College
South Kensington Campus
London SW7 2AZ
jd@doc.ic.ac.uk

ABSTRACT

Clusters are increasingly interconnected to form multi-cluster systems, which are becoming popular for scientific computation. Grid users often submit their applications in the form of workflows with certain Quality of Service (QoS) requirements imposed on the workflows. These workflows detail the composition of Grid services and the level of service required from the Grid. This paper addresses workload allocation techniques for Grid workflows. We model a resource within a cluster as a $G/G/1$ queue and minimise failures (QoS requirement violation) of jobs by solving a mixed-integer non-linear program (MINLP). The novel approach is evaluated through an experimental simulation and the results confirm that the proposed workload allocation strategy not only provides QoS guarantee but also performs considerably better in terms of satisfying QoS requirements of Grid workflows than reservation-based scheduling algorithms.

1. INTRODUCTION

Clusters are becoming important contenders for both scientific and commercial applications. Clusters with different performance and architectures, owned by different organisations are now increasingly interconnected to form a multi-cluster computing system [7].

Complex scientific experiments within a Grid are increasingly specified in the form of workflows, which detail the composition of distributed resources such as computational devices, data, applications, and scientific instruments. Users who submit a workflow to the Grid will often have constraints on how they wish the workflow to perform. These may be described in the form of a Quality of Service (QoS) document which details the level of service they require from the Grid. This may include requirements on such things as the overall execution time for their workflow, the time at which certain parts of the workflow must be completed, cost to the user. In order to determine if these QoS constraints can be satisfied it is necessary to store performance information of resources and applications within the Grid. Such information could also be performance data for software to be run on a computational resource, resource information about speed and reliability, mean service time and mean arrival rate. Here we see that existing Grid middleware for resource descriptions [14] and performance repositories [3] may be used for the storage and retrieval of this data.

Job scheduling within Grid is mainly based on two techniques. Either scheduling is performed based on real time information such as waiting time in the queue, residual processing time; or on average-based metrics such as mean service rate, mean arrival rates. Real time information based algorithms generally perform better than average-based strategies [15]. However, obtaining real time information from a distributed system such as Grid, leads to high overheads. Moreover resources may be distributed geographically, which means that obtaining instantaneous information about the states of geographically distributed resources can lead to substantial delays and consequently to inaccurate scheduling decisions. Also, it may not be possible to obtain instantaneous information at any arbitrary point in time for some distributed systems. Thus, it is necessary to develop approaches which are not dependent on obtaining accurate instantaneous information. The use of average-based strategies seems to be an appropriate approach. Average-based scheduling, for jobs based on FCFS (First Come First Served) rule in a Grid, consists of distributing the workload received by a central entity such as a brokering service to underlying service providers.

The remainder of the paper is organised as follows. Section 2 describes the Grid model considered and assumptions held in this paper. Section 3 presents related work and compares our work with others in the field. Workload allocation strategy in terms of minimising job failures is obtained in Section 4 and the performance of the workload allocation strategy is evaluated in Section 5. Finally, we conclude the paper in Section 6.

2. THE MODEL

In our model of the Grid (see Figure 1) we envisage a number of co-operating Grid Services. We outline relevant services below and align them with existing Grid services. However for simplicity, services such as workflow management system, payment service and others are not shown in figure 1.

Workflows : Workflows are composed of individual tasks (jobs). These tasks get executed on computing boxes within the clusters. Workflows have overall deadline and cost constraints, which are explicitly specified by the end-user. There

MGC '06, November 27, 2006 Melbourne, Australia
Copyright 2006 ACM 1-59593-581-9/06/11 ...$5.00.

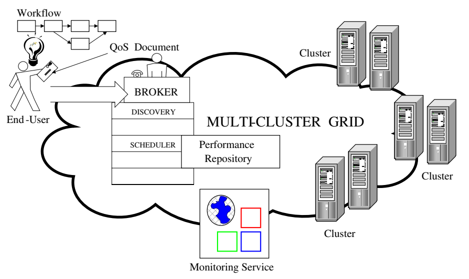

Figure 1: Grid Model

could also be other constraints such as network constraints, availability, reliability. A full list of constraints is beyond the scope of this paper. However for simplicity, we keep the QoS requirements of workflows limited to overall deadline and cost constraints. Deadlines and costs for individual tasks of workflows are calculated by the brokering service using a formula given in the experimental evaluation. We define a workflow failure as failure in meeting the overall workflow deadline or the cost limit. Failure in meeting deadlines or costs of intermediate workflow tasks is not a workflow failure.

Brokering service : End-users construct and submit workflows to a brokering service [10]. The brokering service facilitates the tranformation of an abstract workflow to a more concrete workflow through the discovery of software required by the workflow tasks and performing scheduling. Discovery service facilitates workflow tasks on computing resources by transporting required software on cluster resources. When a workflow task finishes, further tasks must be started. Hence the brokering service dispatches tasks (jobs) of new and old workflows to appropriate cluster resources using a workload allocation strategy developed in the next section. The jobs are executed by the computing resources in the order they are received.

Monitoring Service : Monitoring service takes care of collecting periodic information of the states of cluster resources from their respective management services. This service can be queried by the scheduling service in order to obtain estimates of various parameters such as queue length in order to make accurate scheduling decisions. We assume here that negligible time is spent by the scheduling service to obtain information about clusters from the monitoring service.

Performance Repository : Performance repository stores historical performance data of workflow tasks. The brokering service takes care of logging performance information of services. When the service completes its execution, the brokering service records its execution profile and stores in a performance repository [3]. The scheduling service can then interrogate performance information in order to obtain estimates on the execution times of workflow tasks.

3. RELATED WORK

In a distributed system such as Grid, job scheduling is performed at both global and local level [4] [13]. At global level, workload is distributed to resource clusters and within them, the local schedulers dispatch jobs to the underlying resources via a scheduling strategy. Kao et al. [1] use two homogenous non real time servers to provide a service that satisfies the QoS requirements of jobs. However they don't extend their approach to a distributed system such as Grid and QoS requirements of jobs are limited only to waiting times. Moreover it is assumed by Kao et al. that the waiting time requirements of jobs received by a server follow a uniform distribution. Kao et al. also model the server as an $M/M/1$ queue, which hardly exist in real world situations. Zhu et al. extend the work of Kao et al. by considering more than two servers that aim to satisfy QoS requirements of jobs [12]. The performance of scheduling based on minimising failures to meet waiting time requirements (the maximum time a job can wait before execution) of jobs is also evaluated in [12]. However, their work is confined to a single service with n processing nodes only and does not consider a distributed system such as Grid. Zhu et al. also assume that the waiting time requirements of jobs received by a server follow a uniform distribution and the server is modelled as an $M/M/k$ queue. He at al. [6] extend the work of Zhu et al. by developing a workload allocation strategy for a multi-cluster Grid. They obtain an analytical solution for miss rate (jobs failing to meet their waiting time requirements) of jobs having a slack (waiting time constraints). They minimise the miss rate by allocating an optimal workload to clusters. Moreover they also assume that the waiting time requirements of jobs received by a cluster follow a uniform distribution and again clusters are modelled as $M/M/k$ queues.

Our work focuses on developing a workload allocation strategy which minimises failures of jobs (tasks) of workflows with QoS requirements whilst providing QoS guarantee in a multi-cluster Grid.

4. MINLP FOR MINIMISATION OF JOB FAILURES

In this section we obtain a MINLP which minimises failures of jobs received by cluster resources. The MINLP obtains as solutions, the workload allocation and the job assignments for cluster resources. In mathematics, non-linear programming (NLP) is the process of solving a system of equalities and inequalities over a set of unknown real variables, along with an objective function to be maximized or minimized. The objective function and the functions associated with the unknown variables in the constraints may be non-linear in a NLP. If the unknown variables are all required to be integers, then the problem is called a non-linear integer programming (NLIP) problem. If only some of the unknown variables are required to be integers, then the problem is called a mixed-integer non-linear programming (MINLP) problem. These are generally NP-hard. MINLPs can be solved using advanced algorithms such as branch and bound, outer approximation, generalised Benders decomposition. Our workload allocation problem turns out to be a mixed-integer non-linear program which is developed in the next section. We provide table 1 as a quick reference to the parameters of the MINLP.

4.1 Workload allocation and job assignment based on failure minimisation of jobs

In this section, a workload allocation strategy using minimisation of failures of workflow tasks, using a MINLP is developed. The Grid consists of n clusters with i^{th} cluster having N_i resources. We model each resource in a cluster in

Table 1: MINLP Parameters

Symbol	Function
n	Number of clusters
N_i	Number of resources in cluster i
R_{ij}	Response time of resource j of cluster i for workload of λ_{ij}
c_{ij}	Cost per second of resource j of cluster i
d_y	Deadline allocation of a workflow task
e_y	Cost allocation of workflow task
x_{ijy}	Binary selection variable associated with workflow task y to be executed on resource j of cluster i
λ	Workload received by the brokering service
λ_{ij}	Workload allocation to resource j of cluster i
μ_{ij}	Service rate of resource j of cluster i
z_y^d, z_y^c	Penalty variable associated with deadline, cost for workflow task y

a novel way as a $G/G/1$ queue with infinite customer capacity, meaning the number of jobs that can wait in the queue of a resource is infinite. Hence essentially a resource is indeed a $G/G/1/\infty$ queue. Mean service rate of a resource is μ_{ij}. We consider that the brokering service receives jobs, with an arrival rate of λ, out of which, λ_{ij} is allocated to j^{th} resource of cluster i. The number of jobs that need to be scheduled is J. Thus the arrival rate can be expressed as the sum of workload proportions of resources, given by equation 1.

$$\sum_{i=1}^{n}\sum_{j=1}^{N_i}\lambda_{ij} = \lambda \tag{1}$$

We develop the two main constraints namely deadline and cost constraints one by one. Finally we model the objective function of the MINLP.

- **Deadline Constraint**

  The workload proportion allocated to resource j of cluster i is λ_{ij}. We can write the expected average response time R_{ij} [5] of resource j of cluster i for workload λ_{ij}, given by equation 2. The waiting time in equation 3 is calculated based on the parameters of the resource, whereas the *service time* parameter is calculated based on equation 23, as shown in the experimental evaluation section. The *service time* is the upper bound of expected execution time of workflow task on resource j of cluster i. W_{ij} is the waiting time in the queue, the best known upper bound for which is given by equation 3. The terms $\sigma^2_{ij}(A)$ and $\sigma^2_{ij}(S)$ are the variances of the inter-arrival times and service times of resource j of cluster i respectively.

$$R_{ij} = W_{ij} + service\ time \tag{2}$$

$$W_{ij} = \frac{\sigma^2_{ij}(A) + \sigma^2_{ij}(S)}{2(1-\frac{\lambda_{ij}}{\mu_{ij}})}\lambda_{ij} \tag{3}$$

  Expected average response time must be less than the deadline allocation of job assigned to resource j of cluster i. We can now write the following deadline constraint, given by equation 4.

$$\forall\ i,\ j,\ y,\ (R_{ij} - d_y)x_{ijy} \leq 0 \tag{4}$$

  The following equality constraints, given by equation 5 must also be met. These constraints take care of assigning a job to one and only one Grid service. At the same time they also take care of assigning every job. The binary variable x_{ijy} is 1 if job y is selected to be executed on resource j of cluster i, else it is 0. Equation 6 ensures that the number of assignments are less than or equal to the arrival rate λ_{ij} and also validates equation 4. Morevoer equation 7 ensures that the queue remains stable.

$$\forall y,\ \sum_{i=1}^{n}\sum_{j=1}^{N_i} x_{ijy} \ = \ 1 \tag{5}$$

$$\forall i,\ j,\ \sum_{y=1}^{J} x_{ijy} \ \leq \ \lambda_{ij} \tag{6}$$

$$\forall i,\ j,\ \lambda_{ij} \ < \ \mu_{ij} \tag{7}$$

- **Cost Constraint**

  The cost constraint is similar to deadline constraint, given by equation 8. Expected average cost must be less than the allocated cost of a job assigned to resource j of cluster i.

$$\forall\ i,\ j,\ y,\ (c_{ij}R_{ij} - e_{iy})x_{ijy} \leq 0 \tag{8}$$

The objective is to minimise total failures, i.e. to minimise the number of jobs failing to meet their QoS allocations. The constraints take care of workload allocation and job assignments. However these constraints may fail, thus making an infeasible program. Thus to allow for that we introduce a penalty term ($h^T z$) in the objective. We wish to minimise the penalty and in turn minimise failures. We introduce extra variables (z), one per inequality constraint, that make the constraints feasible at all times. These variables account for the penalty incurred in failing to meet the QoS requirements. The coefficient vector (h^T) of these variables is present in the objective of the MINLP. The values of this vector are the inverse of the terms d_{iy} and e_{iy} present in the LHS of deadline, cost and reliability constraints. We can now write the minimisation problem (MINLP) represented by equations 9 to 18.

$$minimise\ \mathbf{h^T z} \tag{9}$$

subject to

$$\forall\ i,\ j,\ y,\ (R_{ij} - d_{iy})x_{ijy} \leq z^d_{iy} \quad (10)$$

$$\forall\ i,\ j,\ y,\ (c_{ij}R_{ij} - e_{iy})x_{ijy} \leq z^c_{iy} \quad (11)$$

$$\forall y,\ \sum_{i=1}^{i=n}\sum_{j=1}^{j=N_i} x_{ijy} = 1 \quad (12)$$

$$\forall i,\ j,\ \sum_{y=1}^{J} x_{ijy} \leq \lambda_{ij} \quad (13)$$

$$\forall i,\ j,\ \lambda_{ij} < \mu_{ij} \quad (14)$$

$$\sum_{i=1}^{n}\sum_{j=1}^{N_i} \lambda_{ij} = \lambda \quad (15)$$

$$0 \leq \lambda_{ij} \leq \lambda \quad (16)$$

$$\forall i,\ j,\ y,\ x_{ijy} \in \{0,1\} \quad (17)$$

$$\forall i,\ y,\ z^d_{iy},\ z^c_{iy} \geq 0 \quad (18)$$

The above MINLP can be solved by using appropriate non-linear optimisation software. We use CPLEX, an industrial quality optimisation software by ILOG to solve the MINLP. MINLPs are NP-hard problems because apart from being non-linear, they also fall under combinatorial optimisation problems.

Table 2: Clusters setup for simulation 1 and 3

Cluster	Machines	Avg. Speed (MIPS)
1	6	14000
2	6	10000
3	6	5000
4	6	3000

Table 3: Clusters setup for simulation 2

Cluster	Machines	Avg. Speed (MIPS)
1	3	14000
2	3	10000
3	3	5000
4	3	3000

Table 4: Simulation parameters

Simulation	1	2	3
Mean λ (per sec)	1.5-10	0.1-2.0	1.5-3.6
CV λ	0.1-2.0	0.1-2.0	0.1-2.0
Task Mean (μ) (kMI)	7.5-35	10-30	7.5-35
Task CV = σ/μ	0.2-2.0	0.2-1.4	0.2-2.0
Cost per sec	0.07 - 0.7	0.07 - 0.7	0.07 - 0.7
Workflows	Type 1	Type 2	HW
Workflow deadline (sec)	40-60	80-100	40-60
Workflow Cost	1 - 5	1 - 5	1 - 5

5. EXPERIMENTAL EVALUATION

In this section we present experimental results for the workload allocation technique described in this paper.

5.1 Setup

Tables 2, 3 and 4 summarise the experimental setup. We have performed 3 simulations, the first with workflow type 1, second with workflow type 2 and in the third simulation, workload is made heterogenous. The workflows experimented with are shown in figure 2. Workflow type 1 is quite simple compared to type 2, which is a particle physics domain workflow. In the first two simulations, the workflows are all similar, but having different overall QoS requirements. In the third simulation, workload is made heterogenous (HW), meaning any of the three workflows shown as heterogenous workload, in figure 2 could be submitted. Apart from that, the workflows have different QoS requirements. Mean of a workflow task is measured in millions of instructions (MI), while speed of cluster resources is measured in millions of instructions per second (MIPS). We have taken workflow tasks' distributions as general arbitrary data distributions with a finite mean and variance. We have performed 10 runs in each different setup of a simulation and averaged out the values. Initially 500 jobs allow the system to reach steady state, the next 1000 jobs are used for calculating statistics such as mean execution time of workflows, mean workflow failures and mean utilisation of a Grid service. The last 500 jobs mark the ending period of the simulation. The simulation is developed on top of simjava 2 [2], a discrete event simulation package. The Grid size is kept small in order to get an asymptotic behaviour of workflow failures, as coefficient of variation (CV) of workflow task execution time or arrival rates (λ) of workflows are increased. Deadlines of individual tasks of workflows are calculated using equation 19. In order to compute deadlines of workflow tasks, we put no restriction on the nature of their execution time distributions (general distributions with finite mean and variance) and compute deadlines in a way such that 95% of jobs would execute in time under the calculated deadline. Equation 22 is the cumulative density function of execution time distribution associated with a workflow task. Such bounds or confidence intervals on the execution time can be computed using various techniques such as Chebyshev inequality [8], Monte Carlo approach [9] and Central Limit Theorem [8] or by performing finite integration, if the underlying execution time PDFs (Probability Density Functions) are available in analytical forms. Deadline calculation takes care of all possible execution paths in a workflow. $deadline_W$ is the overall workflow deadline for any possible path in a workflow, as shown in table 4, while $cost_W$ is the overall workflow cost. We provide an example for the first task of workflow 2 (HW) in figure 2. Equation 19 is scaled with reference to $deadline_W$, as it is for the first task of the workflow. Subsequent workflow tasks' deadlines are scaled with reference to the remaining workflow deadline. Similarly equation 20 is scaled with reference to $cost_W$, while subsequent workflow tasks' costs are scaled with reference to the remaining budget of the workflow. The *service time* parameter in equation 2 is calculated based on equation 23. The parameter X in the equations below is the upper bound of the 95^{th} confidence interval of the execution time of workflow tasks. The value k in equation 23 is the number of standard deviations required to compute the upper bound of the 95^{th} confidence interval of the service time of workflow tasks.

$$deadline_1 = \frac{X_1}{\sum_{i=1}^{5} X_i} deadline_W \quad (19)$$

$$cost_1 = \frac{X_1}{\sum_{i=1}^{5} X_i} cost_W \quad (20)$$

$$(21)$$

$$P(0 \leq x \leq X_i) = 0.95 \quad (22)$$

$$service\ time = \frac{(\mu + k\sigma)}{speed\ of\ resource} \tag{23}$$

$$P(0 \leq x \leq service\ time) = 0.95 \tag{24}$$

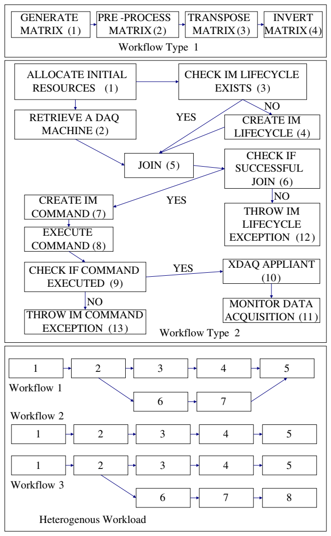

Figure 2: Workflows

5.2 Results

We compare our workload allocation scheme (FF (failure formula)) with traditional job dispatching strategies such as static (SR) and dynamic reservations (DR) based schedulers [11]. The reservations are back-filling enabled, meaning part of the reserved slot is returned to the pool of free slots if the execution completes before the reserved deadline. The static reservation scheduler makes reservations on cluster resources for all tasks within the workflow at the same time. It queries for reservation slot calculated based on equation 23. If all slots succeed within the allowed time and cost limit of the workflow, the scheduling operation is a success. In case of dynamic reservation scheduler, the scheduler obtains reservation slots when the workflow task is required to be scheduled. FF also schedules workflow tasks in a dynamic way, however FF schedules a collection of tasks of different workflows based on the MINLP developed in the previous section.

The workflows don't have any slack period, meaning they are scheduled without any delay as soon as they are submitted. The main comparison metrics between the schemes are mean execution time and cost of workflows, workflow failures and utilisation of Grid services as we increase λ and CV. However we will keep our discussion limited to failures as the main comparison between the schemes is their ability to satisfy QoS requirements.

5.3 Effect of arrival rate and workload nature

Referring to figures 3 and 4, for both low and high arrival rates, FF performs significantly better than SR and DR. However its performance compared to the reservation based schemes drops as λ increases. This trends continues, but the advantage gets reducing as arrival rates increase. This can be explained as follows. When arrival rates increase, more work needs to be scheduled in less time and the average response time and costs are increasing functions of arrival rate,

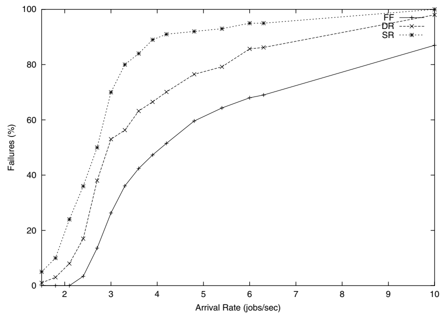

Figure 3: Failures vs λ, CV = 0.2 (Simulation 1)

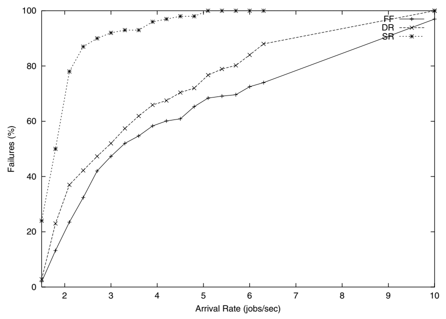

Figure 4: Failures vs λ, CV = 1.8 (Simulation 1)

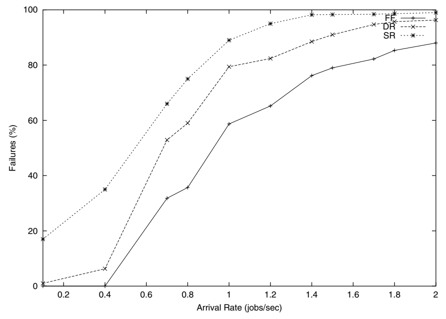

Figure 5: Failures vs λ, CV = 0.2 (Simulation 2)

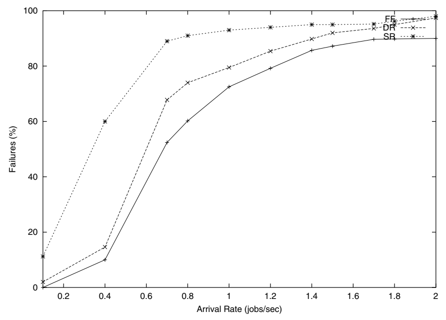

Figure 6: Failures vs λ, CV = 1.4 (Simulation 2)

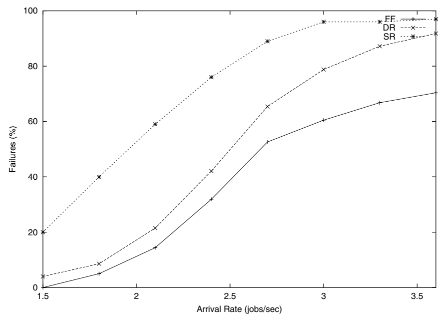

Figure 7: Failures vs λ, CV = 0.2 (Simulation 3)

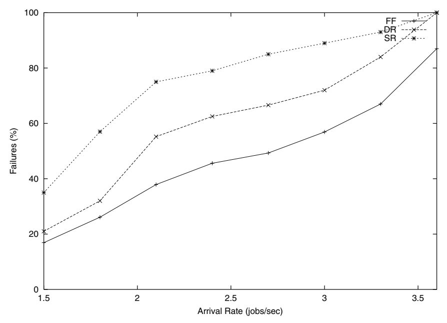

Figure 8: Failures vs λ, CV = 1.8 (Simulation 3)

as is evident from equation 2 in section 4.1. Hence failures due to missing deadline and cost assignments increase and as a consequence workflow failures increase. Referring to figures 5 and 6, for low arrival rates, FF performs significantly better than the other schemes. Referring to figures 7 and 8, the situation is similar to the above cases. Hence heterogenous workload does not change the behaviour of the schemes.

5.4 Effect of CV of execution time of workflow tasks

For both low and high CVs of execution time of jobs, the nature of graphs are similar, however failures increase as CV increases. In case of heterogenous workload, the graphs climb more steeply compared to the case of type 1 workflow. In all cases FF significantly outperforms SR and DR. This shows that the variability of execution time does not significantly affect the nature of graphs for different schemes. However the advantage of a particular scheme over others reduces as failures reach limiting values asymptotically. As CV is increased, failures increase because workflow jobs take longer time to execute and thus tend to complete near their assigned deadlines or even fail to meet their deadlines. Moreover they also may fail to meet their assigned costs.

6. CONCLUSION AND FUTURE WORK

The effectiveness of the workload allocation strategy is evaluated through experimental simulation. Results confirm that workload allocation strategy performs considerably better than the algorithms that do not use these strategies. For both low and high arrival rates of jobs, the workload allocation technique performs significantly better compared to reservation based schedulers. At the same time the technique provides the QoS guarantee as well just as reservation based schedulers provide.

Workflow and workload nature also don't change the performance of the scheme notably. Moreover execution time variability does not change the performance of the workload allocation strategy significantly for both low are high arrival rates. The queueing formulation allows us to get rid of advanced reservations. Moreover its performance over reservation based schedulers is significant at low and high arrival rates and CV of execution of workflow tasks.

As future work we would like to perform experiments with workflows having a slack period, meaning they can wait for some time before getting serviced. We would also like to develop a stochastic version of the MINLP that will help to further reduce the incurred penalty and in turn minimise job failures and also provide QoS guarantee for individual workflows.

7. REFERENCES

[1] B. Kao and H. Garcia-Molina. Scheduling Soft Real-Time Jobs over Dual Non-Real-Time Servers. *IEEE Trans. Parallel and Distributed Systems, vol. 7, no. 1*, pages 56–68, 1996.

[2] F. Howell et al. SimJava. *http://www.dcs.ed.ac.uk/home/hase/simjava*.

[3] G. Nudd and S. Jarvis. Performance-based middleware for Grid computing. *Concurrency and Computation: Practice and Experience*, 2004.

[4] Krauter, K., Buyya, R., Maheshwaran, M. A Taxonomy and Survey of Grid Resource Management Systems. *Technical Report 2000/80: Mannitoba University and Monash University*, 2000.

[5] L. Kleinrock. Queueing Systems. *John Wiley and Sons*, 1975.

[6] Ligang He, Stephen A. Jarvis, Daniel P. Spooner, Hong Jiang, Donna N. Dillenberger and Graham R. Nudd. Allocating Non-real-time and Soft Real-time Jobs in Multiclusters. *IEEE Transactions on Parallel and Distributed Systems*, 2006.

[7] M. Barreto, R. Avila, and P. Navaux. The MultiCluster Model to the Integrated Use of Multiple Workstation Clusters. *Proc. Third Workshop Personal Computer-Based Networks of Workstations*, pages 71–80, 2000.

[8] Milton Abramowitz and Irene A. Stegun. Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables. 1972.

[9] N. Metropolis and S. Ulam. The Monte Carlo Method. *Journal of the American Statistical Association*, 1949.

[10] R. Buyya et al. Economic Models for Resource Management and Scheduling in Grid Computing . *Concurrency and Computation*, 14(13-15):1507–1542, 2002.

[11] W. Smith, I. Foster, and V. Taylor. Scheduling with Advanced Reservations. *In Proceedings of the IPDPS Conference*, pages 127–132, 2000.

[12] W. Zhu and B. Fleisch. Performance Evaluation of Soft Real-Time Scheduling on a Multicomputer Cluster. *Proc. 20th International Conference Distributed Computing Systems (ICDCS 2000)*, pages 610–617, 2000.

[13] Weissman, J. B., Grimshaw, A. S. A Federated Model for Scheduling in Wide Area Systems. *Proceedings of the IEEE High Performance Distributed Computing*, 1996.

[14] Xuehai Zhang and Jennifer M. Schopf. Performance Analysis of the Globus Toolkit Monitoring and Discovery Service, MDS2. In *Proceedings of the International Workshop on Middleware Performance (MP 2004)*, Apr. 2004.

[15] X.Y. Tang and S.T. Chanson. Optimizing Static Job Scheduling in a Network of Heterogeneous Computers. *Proc. 29th International Conference on Parallel Processing*, pages 373–382, 2000.