A Novel Approach To Workload Allocation of QoS-constrained Workflow-based Jobs In A Utility Grid

Yash Patel
London e-Science Centre
Imperial College
London SW7 2AZ

John Darlington
London e-Science Centre
Imperial College
London SW7 2AZ

Abstract

The Grid can be seen as a collection of services each of which performs some functionality. 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 Grid service 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 performs considerably better in terms of satisfying QoS requirements of Grid workflows than scheduling algorithms that don't employ such workload allocation techniques.

1. Introduction

Grid Computing [6] has been evolving over recent years towards the use of service-oriented architectures [8]. Functionality within the Grid exposes itself through a service interface. This functionality may be exposing computational power, storage, software capable of being deployed, access to instruments or sensors, or potentially a combination of the above. As more and more functionality is made available through these services, users seek to combine the services in manners, which allows them to carry out larger and more complex tasks than an individual service can perform. Services are selected which perform the appropriate functionality for the overall task and are combined together through a workflow description. This functionality is often referred to as a component or a workflow task and without loss of generality these components can be composed themselves of a set of components with a workflow description. We refer to all components as workflow tasks in this paper.

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. These resources in turn could be exposed in the form of a web service. 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; reliability of execution. 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 related to execution of Grid services; information about speed and reliability; mean service time and mean arrival rate. Here we see that existing Grid middleware for resource descriptions [21] and performance repositories [9] may be used for the storage and retrieval of this data.

Job scheduling within a 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 [22]. However, obtaining real time information from a distributed system such as Grid, leads to high overheads. Moreover obtaining extremely volatile information such as service load, exact waiting times from geographically distributed Grid services 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 from some Grid services. Thus, it is necessary to develop approaches which are not dependent on obtaining accurate instantaneous informa-

Proceedings of the Second IEEE International
Conference on e-Science and Grid Computing (e-Science'06)
0-7695-2734-5/06 $20.00 © 2006 IEEE

tion. 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. This process is referred to as workload allocation strategy in this paper. The workload allocation scheme determines the proportion of workload directed to a matching Grid service. Once the workload gets collected in the queue of a Grid service, jobs are executed using a FCFS rule.

The remainder of the paper is organised as follows. Section 2 describes the Grid model (figure 1) 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.

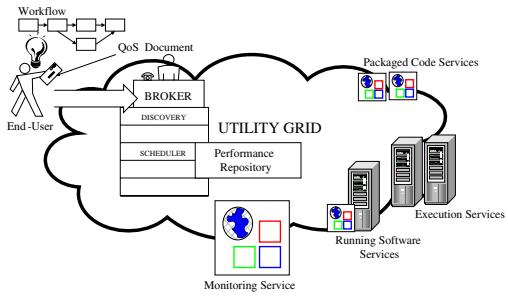

Figure 1. Utility Grid Model

Workflows : Workflows are composed of individual tasks (jobs), which may be abstract in nature. Workflows have overall deadline and cost constraints, which are explicitly specified by the end-user. The workflows also have reliability constraints for individual tasks, which are again specified by the end-user. There could also be other constraints such as network constraints, availability. 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 and reliability constraints for individual tasks within the workflows. Reliability of a Grid service is a constant, which is the number of successful executions divided by the total number of executions. 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. Unsuccessful execution of a workflow task at any point is also a workflow failure. Failure in meeting deadlines or costs of intermediate workflow tasks is not a workflow failure. Workflow tasks with higher reliability requirements, if successfully executed by Grid services with lower reliability is also not a workflow failure.

Running Software Service : The owner of a running software service may wish to expose this themselves to the Grid and provide both computational resources and software. Running software service providers have a fixed location of their service and charge users based on metered usage or subscription. In our model these services exhibit a queueing behaviour. For simplicity we have kept the model of these services limited to a single thread only, meaning only one thread is available at any instant for processing a job waiting in the queue. Hence the service is essentially a $G/G/1$ queue [13]. However this is not a restriction and other models such as multiple threading model could also be used in place.

Packaged Code Service : The owner of a software service may not have the ability to provide computational resources for their software. Therefore they make it available for other third parties to execute as packaged service. We are investigating this approach to software availability [11].

Execution service : These services are computing resources on which packaged code services can be deployed and executed. In our model, a combination of a packaged code service and an execution service together forms one Grid service. We model execution services in a novel fashion as a $G/G/1$ queue. In this way we provide both QoS guarantee as well as get rid of advanced reservations. It remains to be seen whether doing so can provide a performance gain. i.e. improve metrics such as average response times. A naive argument would be that queueing approach computes a joint response time expectation for a set of jobs, whereas reservation computes response time expectation for a single job. Hence performance gain is expected with modelling Grid services via queueing theory approach.

Brokering service : End-users construct and submit abstract workflows to a brokering service [19]. The brokering service facilitates the tranformation of an abstract workflow to a more concrete workflow through the discovery of existing services and performing scheduling. Discovery service finds services implementing a particular interface matching a workflow task, as defined in the workflow. As the workflow task descriptions that make up the abstract workflow only describe the meaning of what should be carried out, the first task of the discovery service is to match these abstract

Proceedings of the Second IEEE International
Conference on e-Science and Grid Computing (e-Science'06)
0-7695-2734-5/06 $20.00 © 2006 IEEE

meanings with their implementations. Here we see that existing technologies such as RDF [4], OWL [3] may be used to describe Grid services. There may be many Grid services matching any given workflow task. When a workflow task finishes, further tasks must be started. Hence the brokering service dispatches tasks (jobs) of new and old workflows to appropriate Grid services using a workload allocation strategy developed in the next section. The jobs are executed by the Grid services in the order they are received.

Monitoring Service : Monitoring service takes care of collecting periodic information of the states of Grid services 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 Grid services from the monitoring service.

Performance Repository : Performance repository stores historical performance data of Grid services. 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 [9]. The scheduling service can then interrogate performance information in order to obtain estimates on the execution times for services matching the abstract services.

3. Related Work

Various application domains and projects such as OMII-BPEL, GridCC are using web services as a means for enabling loosely coupled and extensible systems [10] [18]. Many web services flow specification languages are emerging, such as BPEL4WS [1].

In a distributed system such as Grid, job scheduling is performed at both global and local level [12]. 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. [5] 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 does not quite often 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 [20]. 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 [20]. 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. [15] extend the work of Zhu et al. by developing a workload allocation strategy for a multicluster 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.

Buyya et al. [19] propose a Grid Architecture for Computational Economy (GRACE) considering a generic way to map economic models into a distributed system architecture. The Grid resource broker (Nimrod-G) supports deadline and budget based scheduling of Grid resources. However real time information is required by the resource broker in order to make accurate scheduling decisions. Zeng et al. [14] investigate QoS-aware composition of web services using integer programming method. The services are scheduled using local planning, global planning and integer programming approaches. The execution time prediction of web services is calculated using an arithmetic mean of the historical invocations. However Zeng et al. focus on satisfying QoS requirements of individual workflows. Moreover every time scheduling is performed, real time QoS information of web services has to be known, so that service level agreements can be obtained for executing workflow jobs on web services. We have developed a scheduling strategy using two-stage stochastic programming approach that aims to satisfy QoS requirements of workflows with sufficient guarantee in a volatile Grid [23]. The strategy aims at satisfying QoS requirements of individual workflows. However the strategy developed by us does not require real time information for scheduling.

Our work focuses on developing a workload allocation strategy which minimises failures of jobs (tasks) of workflows with QoS requirements in a utility Grid.

4. MINLP For Minimisation Of Job Failures

In this section we obtain a MINLP which minimises failures of jobs received by Grid services. The MINLP obtains as solutions, the workload allocation and the job assignments for Grid services. 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 asso-

Proceedings of the Second IEEE International
Conference on e-Science and Grid Computing (e-Science'06)
0-7695-2734-5/06 $20.00 © 2006 IEEE

Table 1. MINLP Parameters

Symbol	Function
N_i	Number of services matching service type i
J_i	Number of jobs matching service type i
R_{ij}	Response time of service j of service type i for workload of λ_{ij}
c_{ij}	Cost per second of service j of service type i
f_{ij}	Reliability of service j of service type i
d_{iy}	Deadline allocation of job matching service type i
e_{iy}	Cost allocation of job matching service type i
g_{iy}	Reliability requirement of job matching service type i
x_{ijy}	Binary selection variable associated with job y to be executed on service j of service type i
λ	Workload received by the brokering service
λ_i	Workload matching services of service type i
λ_{ij}	Workload allocation to Grid service j of service type i
μ_{ij}	Service rate of service j of service type i
z^d_{iy}, z^c_{iy}, z^f_{iy}	Penalty variable associated with deadline, cost and reliability constraints for job y of service type i respectively

ciated 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 jobs for Grid services, using a MINLP is developed. The Grid consists of different services which has a specific service type or functionality associated with it. Each service is modelled as a $G/G/1$ queue with infinite customer capacity, meaning the number of jobs that can wait in the queue of a Grid service is infinite. Hence essentially a Grid service is indeed a $G/G/1/\infty$ queue. Mean service rate of a Grid service is μ_{ij}. The number of Grid services matching a service type i is N_i. The number of jobs matching a service type i is J_i. Grid service j of service type i has a unit cost (cost per sec) equal to c_{ij} and reliability equal to f_{ij} (constant). We consider that the brokering service receives jobs, with an arrival rate of λ, out of which λ_i corresponds to Grid services of service type i. Out of λ_i, λ_{ij} is allocated to j^{th} Grid service of service type i. Thus the arrival rate can be expressed as the sum of workload proportions of Grid services, given by equations 1 and 2.

$$\sum_{i=1}^{n} \lambda_i = \lambda \quad (1)$$

$$\sum_{j=1}^{N_i} \lambda_{ij} = \lambda_i \quad (2)$$

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

- **Deadline Constraint**

  The workload proportion allocated to Grid service j of service type i is λ_{ij}. We can write the expected average response time R_{ij} [13] of Grid service j of service type i for workload λ_{ij}, given by equation 3. Equation 3 holds true for running software services, whereas equation 4 for combined packaged code and execution service. The waiting time in equation 4 is calculated based on the parameters of the execution service, whereas the $service\ time$ parameter is calculated based on equation 24 as shown in the experimental evaluation section. W_{ij} is the waiting time in the queue, the best known upper bound for which is given by equation 5. The terms $\sigma^2_{ij}(A)$ and $\sigma^2_{ij}(S)$ are the variances of the inter-arrival times and service times of Grid service j of service type i respectively.

$$R_{ij} = W_{ij} + \frac{1}{\mu_{ij}} \quad (3)$$

$$R_{ij} = W_{ij} + service\ time \quad (4)$$

$$W_{ij} = \frac{\sigma^2_{ij}(A) + \sigma^2_{ij}(S)}{2(1 - \frac{\lambda_{ij}}{\mu_{ij}})}\lambda_{ij} \quad (5)$$

  Expected average response time must be less than the deadline allocation of job assigned to Grid service j of service type i. We can now write the following deadline constraint, given by equation 6.

$$\forall\ i,\ j,\ y,\ (R_{ij} - d_{iy})x_{ijy} \leq 0 \quad (6)$$

  The following equality constraints, given by equation 7 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 Grid service j of service type i, else it is 0. Equation 8 ensures that the number of assignments are less than or equal to the arrival rate λ_{ij} and also validates equation 6.

$$\forall i,\ y,\ \sum_{j=1}^{N_i} x_{ijy} = 1 \quad (7)$$

$$\forall i,\ j,\ \sum_{y=1}^{J_i} x_{ijy} \leq \lambda_{ij} \quad (8)$$

- **Cost Constraint**

  The cost constraint is similar to deadline constraint, given by equation 9. Expected average cost must be less than the allocated cost of a job assigned to Grid service j of service type i.

$$\forall\, i,\ j,\ y,\ (c_{ij}R_{ij} - e_{iy})x_{ijy} \leq 0 \quad (9)$$

- **Reliability Constraint**

  Reliability constraint is given by equation 10. Reliability requirement of a job must be less than the reliability of Grid service j of service type i.

$$\forall\, i,\ j,\ y,\ (g_{iy} - f_{ij})x_{ijy} \leq 0 \quad (10)$$

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}, e_{iy} and g_{iy} present in the LHS of deadline, cost and reliability constraints. We can now write the minimisation problem (MINLP) represented by equations 11 to 20.

$$minimise\ \mathbf{h^T z} \quad (11)$$

subject to

$$\forall\, i,\ j,\ y,\ (R_{ij} - d_{iy})x_{ijy} \leq z^d_{iy} \quad (12)$$

$$\forall\, i,\ j,\ y,\ (c_{ij}R_{ij} - e_{iy})x_{ijy} \leq z^c_{iy} \quad (13)$$

$$\forall\, i,\ j,\ y,\ (g_{iy} - f_{ij})x_{ijy} \leq z^f_{iy} \quad (14)$$

$$\forall i,\ y,\ \sum_{j=1}^{j=N_i} x_{ijy} = 1 \quad (15)$$

$$\sum_{j=1}^{N_i} \lambda_{ij} = \lambda_i \quad (16)$$

$$0 \leq \lambda_{ij} \leq \lambda_i \quad (17)$$

$$\forall i,\ j,\ \sum_{y=1}^{J_i} x_{ijy} \leq \lambda_{ij} \quad (18)$$

$$\forall i,\ j,\ y,\ x_{ijy} \in \{0,1\} \quad (19)$$

$$\forall i,\ y,\ z^d_{iy},\ z^c_{iy},\ z^f_{iy} \geq 0 \quad (20)$$

The above MINLP can be solved by using appropriate nonlinear optimisation software. We use CPLEX [2], an industrial quality optimisation software by ILOG to solve the MINLP. The number of variables and constraints appearing in the MINLP are around 100. Hence the time taken to solve the MINLP is negligible.

5. Experimental Evaluation

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

5.1 Setup

Table 2 summarises 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 real scientific 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 overall workflow deadlines. Mean of a packaged code service (PCS) is measured in millions of instructions (MI), while speed of Grid resources is measured in millions of instructions per second (MIPS). 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 [7], 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

Table 2. Simulation parameters

Simulation	1	2	3
Grid services per task	6-24	3-12	6-24
Service Speed (kMIPS)	3-14	3-14	3-14
PCS Mean (μ_p) (kMI)	7.5-35	10-30	7.5-35
PCS CV (σ_p/μ_p)	0.2-2.0	0.2-1.4	0.2-2.0
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) (sec)	3-12	3-10	3-12
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
Service Reliability(%)	50-100	50-100	50-100
Workflows	Type 1	Type 2	HW
Workflow deadline(sec)	40-60	80-100	40-60
Workflow Cost	1-5	1-5	1-5
Task Reliability(%)	60-95	60-95	60-95

increased. Deadlines of individual tasks of workflows are calculated using equation 21. 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 23 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 [16], Monte Carlo approach [17] and Central Limit Theorem [16] 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 2, while $cost_W$ is the overall workflow cost. We provide an example for the first task of workflow 2 (HW) in figure 2. Mean of the execution time (μ) and coefficient of variation of the execution time (CV) are specified in table 2 with respect to a reference machine. Equation 21 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 22 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 4 is calculated based on equation 24.

$$deadline_1 = \frac{X_1}{\sum_{i=1}^{5} X_i} deadline_W \quad (21)$$

$$cost_1 = \frac{X_1}{\sum_{i=1}^{5} X_i} cost_W \quad (22)$$

$$P(0 \leq x \leq X_i) = 0.95 \quad (23)$$

$$service\ time = \frac{(\mu_p + k\sigma_p)}{processing\ speed} \quad (24)$$

$$P(0 \leq x \leq service\ time) = 0.95 \quad (25)$$

5.2. Results

We compare our workload allocation scheme (FF) with traditional job dispatching strategies like global weighted allocation (GWA) and real time based least-loaded algorithm (RTLL). The GWA scheme calculates the proportion of workload based on the service rate of a Grid service based on the equation 26. Hence, higher the service rate, higher the workload proportion for the Grid service. Thus GWA has no notion of satisfying the QoS requirements of jobs. The least-loaded scheme selects the least-loaded Grid service which can satisfy the QoS requirements of jobs. Hence least-loaded scheme schedules jobs individually. For every schedule, it queries real time information from Grid services.

$$\lambda_{ij} = \frac{\mu_{ij}\lambda_i}{\sum_{j=1}^{N_i} \mu_{ij}} \quad (26)$$

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 low arrival rates, FF performs similar to RTLL. However its performance compared to RTLL drops as λ increases. However FF significantly outperforms GWA. 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, as is evident from equation 5 in section 4.1. Hence failures due to missing deadline and cost assignments increase and as a consequence workflow failures increase. For both low and high CVs, at low arrival rates, FF performs similar to RTLL. Referring to figures 5 and 6, for low arrival rates, FF performs similar to RTLL. However its advantage over GWA is significant. 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.

Proceedings of the Second IEEE International
Conference on e-Science and Grid Computing (e-Science'06)
0-7695-2734-5/06 $20.00 © 2006 IEEE

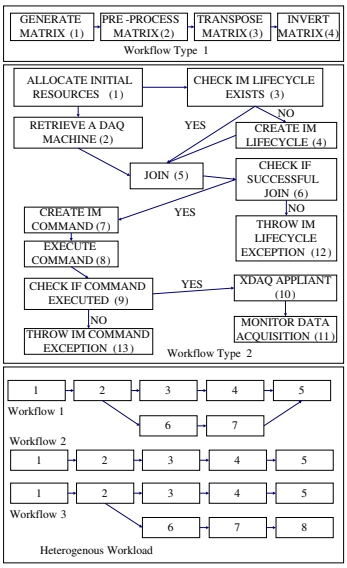

Figure 2. Workflows

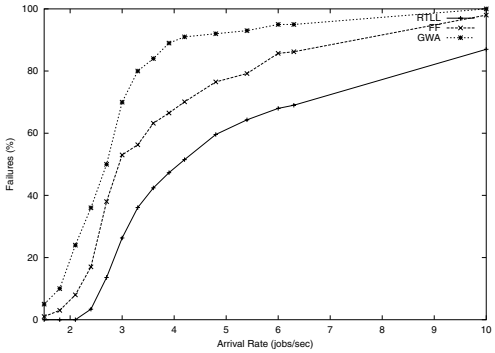

Figure 3. Failures vs λ, CV = 0.2 (Sim. 1)

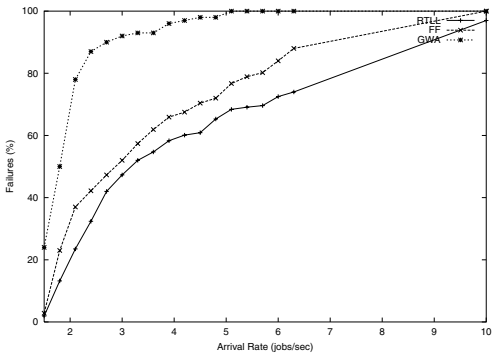

Figure 4. Failures vs λ, CV = 1.8 (Sim. 1)

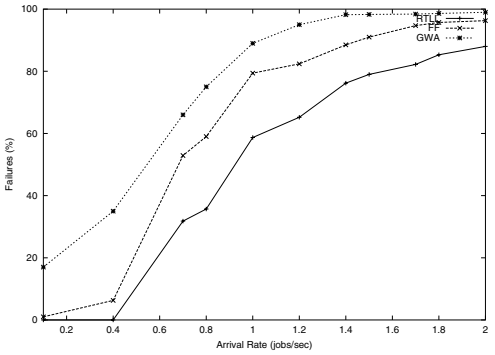

Figure 5. Failures vs λ, CV = 0.2 (Sim. 2)

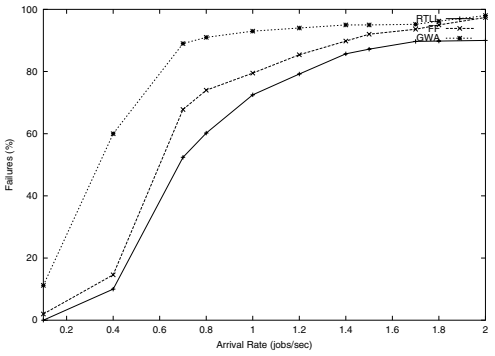

Figure 6. Failures vs λ, CV = 1.4 (Sim. 2)

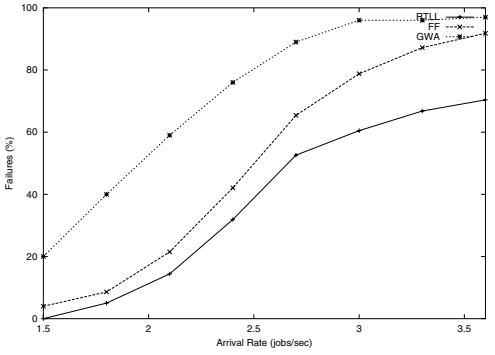

Figure 7. Failures vs λ, CV = 0.2 (Sim. 3)

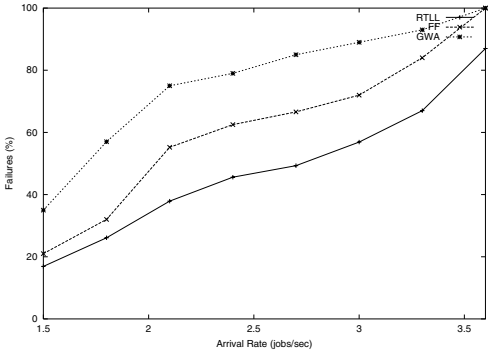

Figure 8. Failures vs λ, CV = 1.8 (Sim. 3)

Proceedings of the Second IEEE International
Conference on e-Science and Grid Computing (e-Science'06)
0-7695-2734-5/06 $20.00 © 2006 IEEE

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 GWA. 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 experimentatal simulation. Results confirm that workload allocation strategy performs considerably better than the algorithms that do not use these strategies. When the arrival rates are low, the workload allocation technique performs similar compared to scheduling algorithms based on real time performance information. 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.

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 evaluate the performance of our scheme against reservation based schedulers. 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.

References

[1] BPEL Spec., http://www-106.ibm.com/developerworks.

[2] ILOG, http://www.ilog.com.

[3] OWL, http://www.w3.org/TR/owl-features.

[4] RDF, http://www.w3.org/TR/rdf-concepts.

[5] 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.

[6] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, July 1998.

[7] Fred Howell and Ross McNab. simjava: a discrete event simulation package for Java with applications in computer systems modelling. *First International Conference on Web-based Modelling and Simulation*, 1998.

[8] N. Furmento, J. Hau, W. Lee, S. Newhouse, and J. Darlington. Implementations of a Service-Oriented Architecture on top of Jini, JXTA and OGSI. In *Grid Computing: Second European AcrossGrids Conference, AxGrids 2004*, volume 3165 of *Lecture Notes in Computer Science*, pages 90–99, Nicosia, Cyprus, Jan. 2004.

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

[10] GRIDCC Collaboration. Grid Enabled Remote Instrumentation with Distributed Control and Computation. *http://www.gridCC.org*.

[11] J. Cohen and W. Lee and J. Darlington. Payment and Negotiation for the Next Generation Grid and Web. In *UK e-Science All Hands Meeting*, Nottingham, UK, Sept. 2005.

[12] 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.

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

[14] L. Zeng et al. QoS-Aware Middleware for Web Services Composition. *IEEE Transactions on Software Engineering*, 30(5):311–327, May 2004.

[15] 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.

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

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

[18] OMII. Open Middleware Infrastructure Institute. *http://www.omii.ac.uk*.

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

[20] 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.

[21] 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.

[22] 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.

[23] Yash Patel, Andrew Stephen McGough and John Darlington. Workflow Scheduling in WOSE. In *Proceedings of the UK e-Science All Hands Meeting*, Nottingham, UK, Sept. 2006.

Proceedings of the Second IEEE International
Conference on e-Science and Grid Computing (e-Science'06)
0-7695-2734-5/06 $20.00 © 2006 IEEE