Pro-active Failure Handling Mechanisms for Scheduling in Grid Computing Environments

Benjamin Khoo B.T and Bharadwaj Veeravalli

National University of Singapore Department of Electrical and Computer Engineering,

{g0402607, elebv}@nus.edu.sg

September 9, 2009

Abstract

In this paper, we consider designing pro-active failure handling strategies for grid environments. These strategies estimate the availability of resources in the Grid, and also preemptively calculate the expected long term capacity of the Grid. Using these strategies, we create modified versions of the backfill and replication algorithms to include all three pro-active strategies to ascertain each of its effectiveness in the prevention of job failures during execution. Also, we extend our earlier work on a co-ordinate based allocation strategy. The extended algorithm also shows continual improvement when operating under the same execution environment. In our experiments, we compare these enhanced algorithms to their original forms, and show that pro-active failure handling is able to, in some cases, avoid all job failures during execution. Also, we show that NSA provides the best balance of enhanced throughput and job failures during execution in the algorithms we have considered.

1 Introduction

Grid computing has evolved over the past years from research and slowly reaching into the commercial space. As more people become aware of Grids, the types of computational environment has also changed. On one hand, large scale collaborative Grids continue to grow, allowing both intra and inter organizations to access vast amount of computing power, on the other hand, increasing number of individuals are starting to take part in voluntary computations, involved in projects such as Seti@Home or Folding@Home. Commercial organizations are also beginning to take notice of the potential capacities available within their organization if the workstations are aggregated into their computing resource pool.

This increase in awareness, has lead to various products, both in research and commercial, that handle resource allocation and scheduling of jobs to harness these computation powers. Products such as Platform LSF [1] or the Sun Grid Engine [2] provide algorithms and strategies that handle Dedicated Grid Computing Environments (GCE) well, but is unable to work optimally in Desktop Grid environments due to the high rate of resource failures. The same applies for technologies such as United Devices [3] or XGrid [4], whereby although

it excels in Desktop Grid (EG) environments, is unable to provide the same level of performance in Dedicated Grid (DG) environments. This is due to the assumptions made on the possibly high failure rates, resulting in simple scheduling algorithms used in such systems.

Given the ability to preemptively know about failures and to handle it adequately would allow the rise of a new class of scheduling algorithms that is able to prevent job failures resulting from the failure in the execution environment. Coupling this with the fact that handling job failures can help to reduce the turn-around time for a successful job completion, it would be then possible to create large scale scheduling algorithms where it is able to effectively estimate and allocate jobs to resources that can fulfill its task with minimal interruptions and re-scheduling. This will ultimately result in higher throughput, and a higher level of quality for jobs submitted to Grids. This motivates us to invent new strategies that take into account the failure possibilities to render the best services.

2 Related Works

We classify the current available work on Grid failures into pro-active and passive mechanisms. By pro-active mechanisms, we mean algorithms or heuristics where the failure consideration for the Grid is made *before* the scheduling of a job, and dispatched with hopes that the job does not fail. Passive mechanisms identify algorithms that handles the job failures *after* they have occurred. While there are existing works that looks at failure handling in the literature such as [6, 5] in other fields of engineering, most of these are not related to Grids. Many of these are also more complex requiring a large amount of information otherwise unavailable in a GCE.

Of those that look into the issues of failures within the Grid, many works are primarily passive in nature and deal with failures through Grid monitoring as mentioned in [10] and [8]. These methods mainly do so by monitoring for failures followed by either a checkpoint-resume or terminate-restart [15, 11, 12, 13]. Two passive failure mechanisms are introduced in [14, 21] and [16]. While [14, 21] operates by replicating jobs on Grid resources, [16] only looks at volunteer Grids. The former can possibly lead to an over allocation of resources, which will be reflected as an opportunity cost on other jobs in the execution queue. The latter addresses independent task executing on the resources, however, it does not address how these resources can potentially co-operate to run massively parallel applications.

In the literature referenced above, there have been limited proposals on the resource model suitable for Grids and the underlying mechanism to prevent failures of jobs in Grids by pro-actively estimating the state of the Grid prior to job dispatching. Other pro-active and dynamic fault-tolerant scheduling proposals such as [17] focus more on job reliability than on maximizing the total availability of resources with in the Grid. This results in a reduction in the total capacity of the Grid, very similar to replication based techniques. In [18], a mechanism to determine suitable candidate schedules based on Bayesian Networks and Genetic algorithms is introduced to modify candidate schedules. We find that this is highly complimentary to our proposed strategies, and can potentially be used to reduce run-time prediction errors within our algorithm. However, this is not demonstrated here as our proposal focuses on how we can propose a valid strategy based on minimal learning. A simple-state based and filtering model to handle prediction in computational Grids was also presented in [8]. While the

method works well in simplicity and looks at the reliability of a computational resource, it does not take into account additional un-certainties of recovery. Our proposed model proposes an alternative of how potentially well studied models of failures can assist in prediction failures in GCEs by making use of existing ideas and models. These can then continue to be further enhanced, perhaps using well established work, to create simple filtered models of pro-active failure handling to detailed analysis of individual node failure predictions.

[7] looks at the availability prediction of fine-grained cycle sharing systems using a Semi-Markov Process model. While this method has been shown to be effective, we agree more with the model presented in [9] as it is simpler and looks at more basic states and thus easier to implement. [9] looks at model fitting using parametric and non-parametric methods for predicting machine availability in a desktop Grid. While the parametric methods increases in complexity as conditions increase, non-parametric investigates the failure models based on a given statistical distribution. This differs from our proposed strategy as we view the probability of failure as independent events. The measurements in [9] also investigates only the accuracy of prediction on resource lifetime for desktop Grids. We differ in our strategy and experimentation by including job characteristics and simulating in other common GCEs such as to achieve a better job allocation schedule.

3 Scope of this work & our contributions

In this paper, we conclusively show that through pro-active failure handling, it is possible to improve the behaviour of existing scheduling strategies and algorithms such that it is able to prevent job failures during execution. We introduce pro-active failure handling strategies which allows existing scheduling algorithms to be modified to avoid job failures upon scheduling. Three strategies are introduced, namely (1) Site availability based allocation (SAA), (2) Node availability based allocation (NAA), and (3) Node and Site based allocation (NSA) strategies. These are then augmented into the backfill scheduling algorithm and the replication scheduling strategy. The modified and unmodified algorithms are then compared. We further introduce an extension of the co-ordinate based resource allocation strategy presented in [23]. We clearly show the improvement in job reliability by introducing pro-active failure handling to this algorithm using the proposed model.

In section 4, we first highlight how we estimate the resource availability of the Grid. The simulation environment is then described in section 5. This is followed by section 6, describing how these results are used to formulate the pro-active failure handling mechanisms for existing algorithms. The results are then discussed in 7. The conclusion of the study then follows in section 8.

4 Estimating Resource Availability

In this paper, we define *Failure* to be the breakdown of communication between computing resources, thereby leading to a loss in status updates in the progress of an executing job. This failure can be due to a variety of reasons such as hardware or software failures as stated in [10]. In our proposed strategy, we do not specifically identify the cause of the failure, but generalize it for any possible kind. This is because while differentiating between different forms of failure might be useful in prevention during implementation and system recovery, we feel that inability to account for failure during allocation will still cause a slow-down in job completion time if

it is to occur in the midst of job execution. This could be avoided if the system is made aware of it. As our proposal looks at being able to pro-actively avoid failure issues during the allocation process with minimum complexity, we find this generalization acceptable in view of failure avoidance. This pro-active strategy can then be used together with a passive failure mechanism to recover from the different types of failure, such as those stated in [10], if required to do so. We also look upon a failed resource as one that will have to be restarted and all history of past executions be cleared without checkpoint.

We also use the phrase *"availability of the nodes"* and *"capacity of the system"* interchangeably. This is taken from a point of view where every node meant to be in the GCE will contribute a certain set of resource that would affect the capacity of the GCE. A change in availability of the nodes thus affects the capacity of the GCE. Capacity in this case refers to the GCE state consisting of processing, memory and/or disk capabilities to assist in completing jobs sent into the Grid. In the event where there is a change in the contribution of resources during different phases of availability, a single physical system, or "real" node, can be modeled as multiple "virtual" nodes. Each of the "virtual" nodes, would then go on or offline during the different phases of its life-cycle, affecting the capacity of the GCE. In either case, the scheduling system would not make any differentiation between "real" or "virtual" nodes, and consider all "virtual" nodes as individual "real" nodes. "Virtual" nodes of a single "real" system will continue to have describing characteristics that would relate them to each other, forming a set of closely related resources. This this case, jobs passing through the scheduling system can then be potentially ran in the GCE the same way.

In order to build a model for resource availability, we first define the various stages of availability that it needs to go through from the perspective of an external agent. We place these stages in the following order:-

1. Resource coming online
2. Resource participation in Grid Computing Environment (GCE)
3. Resource going offline
4. Resource undergoing a offline or recovery period
5. Resource coming back online (return to first stage)

We do not identify the reason *why* the resource has gone online or offline from the view of the external agent. The agent, however, does register that if the resource goes offline, the possibility that any process that has been executing on that resource could possibly be interrupted and might not be restored. Unless the mechanism of execution allows for some form of check-point or recovery, the past computation cycles on the machine can be assumed to be lost.

Taking these 5 stages viewed by the external agent, and generalizing the states of the resource on the GCE, we easily classify that a resource has entered a state of a general *failure* or has *recovered* from its unavailable failed state. Thus, under these assumptions, from the resource perspective, we similarly break down the participation of a resource in a GCE into the following stages:-

1. Resource becomes available to the GCE
2. Resource continues to be available pending that none of the components within itself has failed

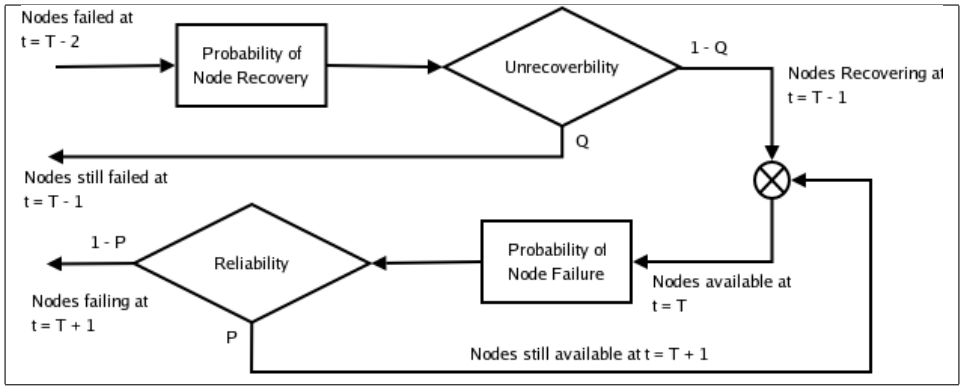


Figure 1: Resource Life Cycle Model for resources in the GCE

3. Resource encounters a failure in one of its components and goes offline for maintenance and fix
4. Resource goes through a series of checks, replacements or restarts to see if it is capable to re-join the GCE
5. Resource comes online and becomes available to the GCE (return to first stage)

From the above, it was observed that in Stages (2) and (4), the resource undergoes a period of uncertainty. This uncertainty stems from the fact that the resource probably might not fail or recover for a certain period of time. Based on these stages the model presented in [19] was constructed. The Resource Life Cycle (RLC) Model shown in Figure 1 identifies the stages where by Grid resources under-go cycles of failures and recovery, and also accounts for the probabilities of *each* resource being able to recover or fail in the next epoch of time. Thus using this model, we are able to describe any general form of resource failure that would cause an external agent to lose job control or connectivity to the said resource.

4.1 Pro-active Failure Handling versus Passive Failure Handling

In most of the mechanisms that improves the resilience of a scheduling strategy, it has been observed that steps were taken to re-schedule a troubled job, or replicate jobs hoping that one of them is successful. Mechanisms such as those in [15, 11] works in this fashion. In general, it was observed that the handling of failures by allocation strategies can occur either before the actual allocation itself, or after the allocation of the resources. We term these methods as Pro-active or Passive methods respectively.

While Passive methods using techniques of job monitoring are relatively easier to implement, Pro-active methods require more information from the GCE and works in a probabilistic fashion. While there exist pro-active methods such as replication where the decision of how to address possible failures in the GCE are made *before* the job is executed, we find that such static mechanisms are unable to cope with the dynamism of the GCE. An effective pro-active strategy should provide a way, with all information considered, deny any job from any possible failures. This potentially reduces the failure rates within a GCE, and also increases the capacity and throughput in a system. This is unlike passive methods where re-submission of jobs typically leads to a decrease in throughput in the system. It is, however, worth while to note, as shown in Figure (2), that both

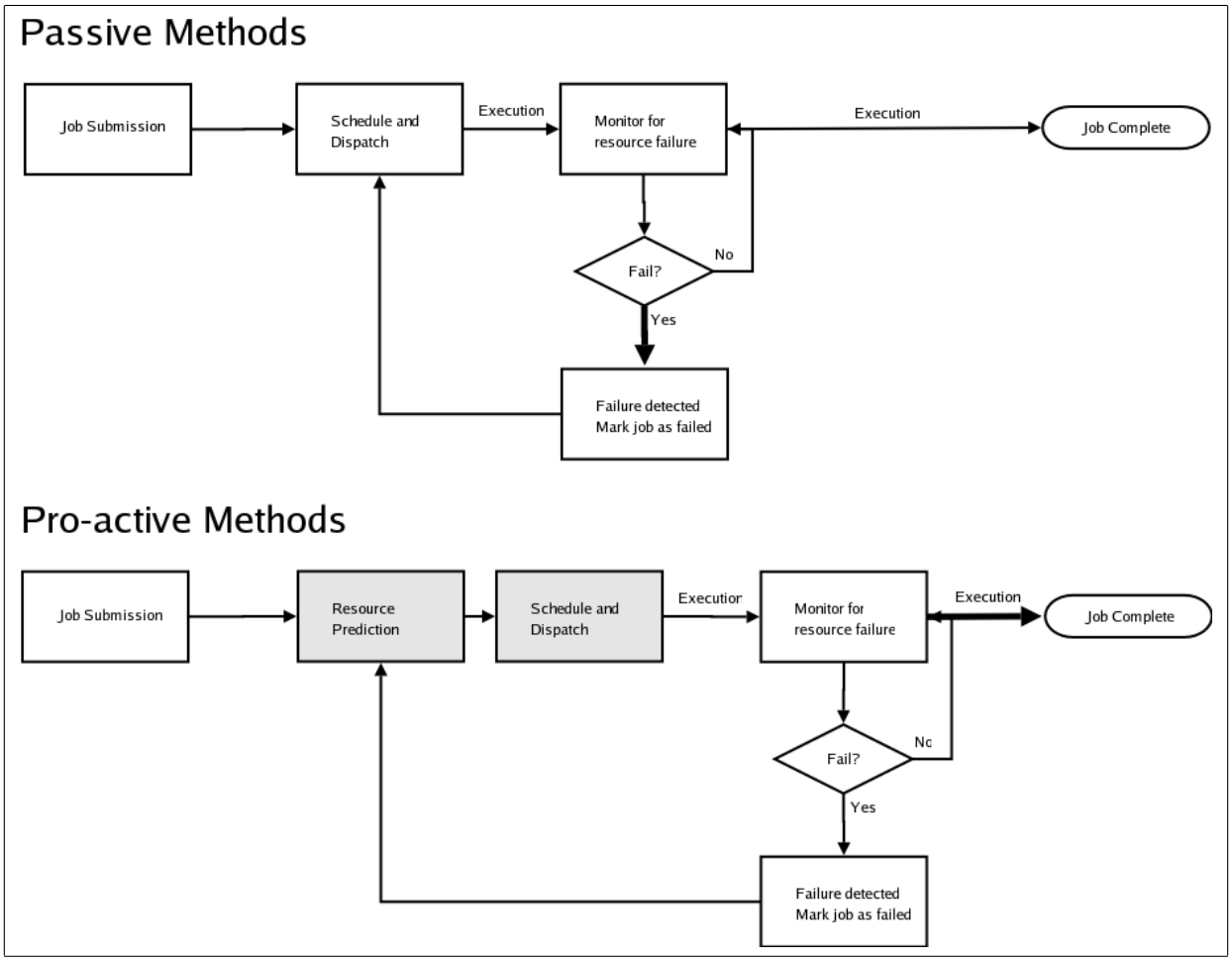


Figure 2: Passive and Pro-active mechanisms used to handle failure

pro-active and passive methods are not substitutes to each other, but rather, they are complimentary. One will never be able to fully predict the state of the GCE, and every pro-active method will have cases where it is unable to accurately reflect the state of the GCE. It is thus beneficial to continue to include passive failure handling mechanisms to assist in such situations.

4.2 Mathematical Modelling

In order to construct a pro-active scheduling strategy, we first construct a mathematical model based on the above mentioned Resource Life Cycle so as to be able to predict the capacity in a GCE given a total fixed number of resource that can possibly participate in the environment. The purpose of the mathematical model is to allow us to answer the following questions:-

1. How many nodes, as an estimate, would there be in the Grid at a certain time?
2. What is the probability of a job being able to complete its execution?

Addressing these two important questions will allow our strategy to dispatch jobs only to resources that will more likely guarantee the successful completion of the job, and know ahead the likely capacity of the GCE at a point in the future.

We first define the following notations and definitions associated with them:-

- $MTTF_j$ *and* λ_j^F : The Mean Time to Failure represents the average amount of time a resource is available to the GCE before going offline. We also term the average rate of failure to be $\lambda_j^F = \frac{1}{MTTF_j}$. Where j denotes the node index in the GCE.
- $MTTR_j$ *and* λ_j^R : The Mean Time to Recovery represents the average amount of time taken for a resource to rejoin the GCE after going offline. We also term the average rate of recovery to be $\lambda_j^R = \frac{1}{MTTR_j}$.
- τ, τ_j^D *and* τ_j^U : τ represents a specific time instance after the time period T, while τ_j^D and τ_j^U are defined as the duration of the state times of the jth node in either DOWN or UP states. We note that for a node, if $\tau_j^D > 0$ then $\tau_j^U = 0$ and vice versa.
- S_T : The number of nodes available for a period of time T.
- M_T : The number of nodes unavailable for a period of time T.
- K_T : This equals to the total number of nodes in the GCE that we would like to consider, and $K_T = S_T + M_T$, for all values of T.
- P_j : Denotes the resource reliability is a single value representing the likely-hood of a resource staying online at any given time. This value is influenced by information such as the resource availability pattern to the GCE, the reliability of the various components in the resource and the reliability value provided by the creators of this resource.

- Q_j : Denotes the resource unrecoverability is a single value representing the likely-hood of a resource recovering from its offline state at any given time. This value is influenced by information such as resource unavailability pattern to the GCE, the difficulty to replace parts in the resource that has failed and the service level provided by the creators of this resource.

- Pr_j^{UP} and Pr_j^{REC} : The probabilities of a resource remaining in its UP, or online, state and recovering from its DOWN, or offline, state respectively.

Note that the $MTTF$ and the $MTTR$ values are collectively termed as MTT values in the rest of this paper.

The above questions can now be paraphrased more specifically as:-

1. How many resources are there at $T+\tau$ time given that there are S_T resources available and M_T resources unavailable at time T?

2. What is the probability of a set of resources staying up over a period of time τ?

The answer to these questions will allow one to estimate the capacity of the Grid in the future. It would also aid us to approximate the likely-hood of a successful job completion when dispatched to a known group of resources. Alternatively, one can also choose to dispatch jobs only to resources that are likely to remain available to ensure successful job completion.

We note that in the RLC model, a resource can have exactly one failure or recovery before it switches its state from being *online* to *offline*, or vice versa. We also note that if given that the MTT, P and Q values are reliable, the duration of a resource being online would highly affect the probabilities of a resource remaining in steady state.

We assume that each event of a state switch is independent of each other. This is a reasonable assumption when we consider a very small instance in time between $\tau-1$, τ and $\tau+1$.

Using the Poisson Distribution to model the event of a single independent change in state, we obtain the probabilities of this event as the following:-

- Probability of a failure on node j due to $MTTF$ after period of UP state at τ_j^U is given by,

$$\left(\frac{1}{MTTF_j}\right)\sum_{t=0}^{\tau_j^U} e^{-(\lambda_j^F t)}(\lambda_j^F t) = \lambda_j^F \sum_{t=0}^{\tau_j^U} e^{-(\lambda_j^F t)}(\lambda_j^F t) \tag{1}$$

- Probability of a resource recovery on node j due to $MTTR$ after a period of DOWN state at τ_j^D

$$\left(\frac{1}{MTTR_j}\right)\sum_{t=0}^{\tau_j^D} e^{-(\lambda_j^R t)}(\lambda_j^R t) = \lambda_j^R \sum_{t=0}^{\tau_j^D} e^{-(\lambda_j^R t)}(\lambda_j^R t) \tag{2}$$

In addition to a resource changing states due to the MTT values, it has to be noted that there are other factors that could cause a change in state which was represented by P_j and Q_j. As the probabilities of P_j and Q_j are independent from the MTT values, it is possible to obtain these values for a single jth resource. Therefore, its probability of remaining in its UP or DOWN state can be immediately realized as:-

- Probability of a resource j remaining in its UP state at τ_j^U

$$Pr_j^{UP} = 1 - \lambda_j^F (1 - P_j) \sum_{t=0}^{\tau_j^U} e^{-(\lambda_j^F t)} (\lambda_j^F t) \tag{3}$$

- Probability of a resource j recovering from its DOWN state at τ_{jD}

$$Pr_j^{REC} = \lambda_j^R (1 - Q_j) \sum_{t=0}^{\tau_j^D} e^{-(\lambda_j^R t)} (\lambda_j^R t) \tag{4}$$

Considering that there is a set of n resources where $1 \leq n \leq S_T$, the probability of this set of resources remaining in the UP state at $T+1$, will be given by,

$$Pr^{UP}\{n_{T+1}\} = 1 - \prod_{j=1}^{n} \lambda_j^F (1 - P_j) \sum_{t=0}^{\tau_j^U} e^{-(\lambda_j^F t)} (\lambda_j^F t) \tag{5}$$

Similarly, for a set of n resources where $1 \leq n \leq M_T$, the probability of this set of resources recovering from its DOWN state at $T+1$, will be given by the following equation.

$$Pr^{REC}\{n_{T+1}\} = \prod_{j=1}^{n} \lambda_j^R (1 - Q_j) \sum_{t=0}^{\tau_j^D} e^{-(\lambda_j^R t)} (\lambda_j^R t) \tag{6}$$

Equations (5) and (6) suggest a methodology whereby it is possible to estimate the number of resources available at $(T+1)$. Under the assumption that the resources remain in constant state within τ period of time, it is possible to extend equations (5) and (6) to estimate the probability of Pr^{UP} and Pr^{REC} at time $(T+\tau)$. This is captured by equations (7) and (8) respectively.

$$Pr^{UP}\{n_{T+\tau}\} = 1 - \prod_{j=1}^{n} \lambda_j^F (1 - P_j) \sum_{t=0}^{\tau_j^U + \tau - 1} e^{-(\lambda_j^F t)} (\lambda_j^F t) \tag{7}$$

$$Pr^{REC}\{n_{T+\tau}\} = \prod_{j=1}^{n} \lambda_j^R (1 - Q_j) \sum_{t=0}^{\tau_j^D + \tau - 1} e^{-(\lambda_j^R t)} (\lambda_j^R t) \tag{8}$$

From the RLC model and equations (7) and (8), it is therefore possible to estimate the number of resources available at S_{T+1} as $S_T Pr^{UP}\{S_{T+1}\} + M_T Pr^{REC}\{M_{T+1}\}$. This can be further extrapolated to obtain an estimate on the number of resources available at $S_{T+\tau}$ given by equation (9).

$$S_{T+\tau} = S_T Pr^{UP}\{S_{T+\tau}\} + M_T Pr^{REC}\{M_{T+\tau}\} + e_T \tag{9}$$

In Equation (9), e_T is the error adjustment in prediction based on the average historical error predictions made. This can be easily captured by recording and taking the average of the difference between the number of resources predicted to be available at T and the actual number of resources available at $(T+1)$. Equation (9) ultimately states that the number of UP nodes available at $(T+\tau)$ is the sum of the number of nodes staying up and recovering at time $(T+\tau-1)$. $Pr^{UP}\{S_{T+\tau}\}$ and $Pr^{REC}\{M_{T+\tau}\}$ are the probabilities of a node staying in

the UP state and recovering from a DOWN state at time $(T + \tau)$ respectively. This enables us to approximate the acceptance of a job and subsequently run it at any point of time in the future.

While simulations of the prediction mechanism based on (9) has shown to be able to estimate the number of resources in a Grid Computing Environment (GCE) within the bounds of ± 2[1]. It is clear from the equations that this, however, requires analysis of each resource and can be unwieldy in both computation and information required. The advantage remains, however, that all equations leading up to Equation (9) provide a way to approximate the availability of a single or a group of resources.

In seeking a less computationally intensive mechanism to estimate the number of resources, it was noted that the MTT values alone were able to estimate the capacity of the GCE over a long period of time. The resulting GCE capacity obtained shows that the average availability of the GCE can be estimated by using the General Availability Equation (GAE) $\frac{\sum MTTF}{\sum MTTF + \sum MTTR}$. This provides the average capacity in the GCE, allowing the allocation strategy to be able to define an upper limit to the number of resources requested by the job at the point of submission. This also prevents users from over-requesting resources thereby leading to failures that can affect throughput. However, while the GAE provides the average number of resources in the GCE, the shortcoming of the GAE is that it does not provide any information as to *which* resource will be leaving or rejoining the GCE. This lends itself to be unable to determine the availability of a specific set of resources within the GCE.

4.3 Comparing Replication and Prediction

In this section we theoretically compare the difference between two pro-active allocation strategies, namely (1) Replication and (2) Prediction. We show that it is meaningful to try to approximate the capacity of the GCE before job submission and how it benefits the allocation strategy if it attempts to do so.

4.3.1 Replication

Assume that a GCE consists of S resources. It is required to process a queue containing J number of jobs requiring T amount of time to process. Given that the value of GAE is α, the effective capacity of the GCE would be represented by αS. In the case of job replication, a job is submitted K times into the GCE (where $K \geq 2$) . This results in the GCE being unable to execute any job requests for exactly S resources, thereby limiting the maximum capacity by a factor of $\frac{1}{K}$. The maximum load of the GCE accounting for its effective capacity is thus given by $\frac{S\alpha}{K}$. Assuming further that the real expected time of each job j to complete is $E_T[j]$ and the theoretical time required for the job to complete its execution is $H_T[j]$, we can conclude that if the requested load $L_j \leq \frac{S\alpha}{K}$, there would be enough capacity in the GCE to be able to execute all the replicas of the job at the same time. If j is successfully executed, it is noted that $E_T[j] \nrightarrow H_T[j]$. If the MTT values of the replica set is identical to that of the GCE, the probability of all replicas failing will be given by $(1 - \alpha)^K$. It is clear to note that the probability of any replica to succeed is $1 - K(1 - \alpha)$. It is observed in Figure 3 that increasing the replication factor of K results in the rate of the probability of j to succeed in its execution also

[1]This value was determined by passing a GCE based on the resource availability model going through the stages of figure 1 through a prediction cycle and capturing the real and predicted results. This error can perhaps be further adjusted by making use of GA based methods as shown in [18].

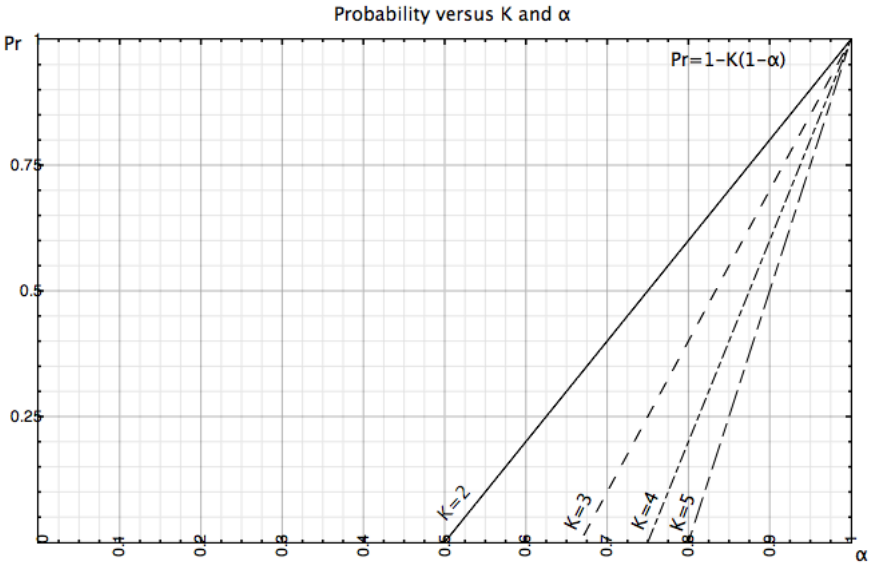


Figure 3: Probability of success versus α under varying replication factors K

by K, However, this benefit in replication is offset by the fact that the GCE is required to satisfy $\alpha = (1 - \frac{1}{K})$ before this advantage is realized. This defines a requirement on the GCE to satisfy this criteria. It is to be noted that as K increases, the site availability required for any of the job replica to succeed also increases. We find this conclusion consistent with many other experiments that demonstrate that the best level of job replication is when $K = 2$, providing the best balance between the requirement of the GCE versus the level of improvement when replicating.

In the event where the requested job load $L_j \geq \frac{S\alpha}{K}$, L_j and its replicas will not be able to execute at the same time. Instead, the replicas would queue in the GCE and be executed in a serial fashion. The replicas will only be able to execute simultaneously, at an instance in time whereby $\frac{S\alpha}{K} = L_j$. This will however be highly subject to failure as the expected GCE capacity is less than that of L_j. In the best case, the first replica will complete and $E_T[j] = H_T[j]$. In the worse case, $E_T[j] \rightarrow K H_T[j]$. Assuming that the average time to complete any job j under these circumstances is $(\frac{1+K}{2})H_T[j]$, we can effectively conclude that the average time taken to process the entire queue will be $T = J(\frac{1+K}{2})H_T[j]$, which is on the average $\frac{1+K}{2}$ times longer than the theoretical time. This effectively decreases the throughput of the GCE which is given by $\frac{2}{H_T[j](1+K)}$.

In both circumstances of job replication shown above, it was established that such a strategy always results in either a lowered capacity in the GCE, or a reduction in throughput in the GCE, and in some cases, both.

4.3.2 Prediction

Assuming a similar setup of a GCE as that used above, the effective capacity of the GCE continues to be $S\alpha$ where α is the GAE value obtained for the entire GCE with J jobs. For the sake of prediction, we introduce the probability that a wrong prediction will be made for each resource Er. We further assume that a wrong prediction on a resource will always result in a failure. We maintain that the expected time for a job j to

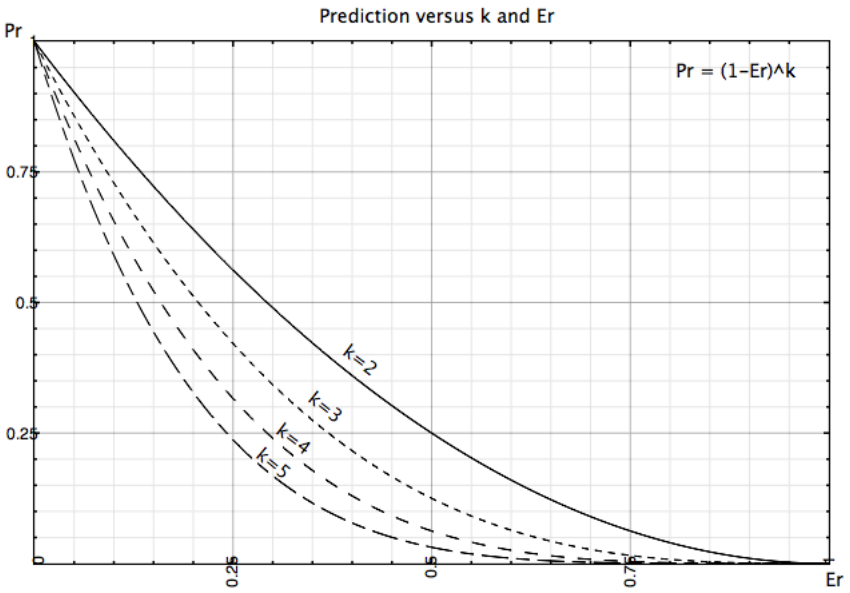


Figure 4: Probability of success Pr versus Er under varying division factors k

complete continues to be $E_T[j]$, while the theoretical time for it to complete is $H_T[j]$. Given that j will be divided into k nodes for execution, where $2 \leq k \leq S$, the probability of a job succeeding in its execution is dependent on all the subdivisions successfully executing. This is given by $Pr = (1 - Er)^k$ as shown in Figure 4.

In the figure, it is noted that the probability of success for each job j depends on the factor k. This is consistent with the fact that the more a job is divided into different resources, the more likely it is to fail. We also note that this probability of success is not affected by the capacity S of the GCE and does not impose a minimum requirement of the GCE to be available before a job can succeed in execution. It is noted that errors in prediction result in an exponential decline in the probability of success of j. When prediction is used with other failure detection techniques and subsequently re-submitted for R number of times, the probability of success is improved by a factor of R. The probability of success is thus changed to $Pr = R(1 - Er)^k$. A variation of Pr is shown in Figure 5. We note that the resubmission of j by $R = 2$ can result in a definite completion of j when Er is 0.15. This threshold of prediction error is even higher when $k = 2$, meaning that even a prediction error of 0.25 when split over two resources can almost absolutely result in a successful execution of j. Once again, this certainty varies from job to job and is not dependent on the capacity of the GCE as long as $L_j \leq S\alpha$.

This certainty allows one to be able to choose the R factor considering the type of workload the GCE is subjected to. If given that all jobs in J has $k \leq 4$, it can be then decided that a $R = 4$ with $Er = 0.25$ will result in all jobs being completed in the queue with $T < JRH_T[j]$. The throughput of such a strategy is therefore equal or greater than $\frac{1}{RH_T[j]}$. The actual throughput is once again dependent on the workload model applicable in the GCE. However, it is noted that, as prediction operates independently of the variables required in replication, these two strategies can be used together to improve the successful throughput of the GCE.

From the above comparisons, we can clearly see that the ability to predict the resource states does not act

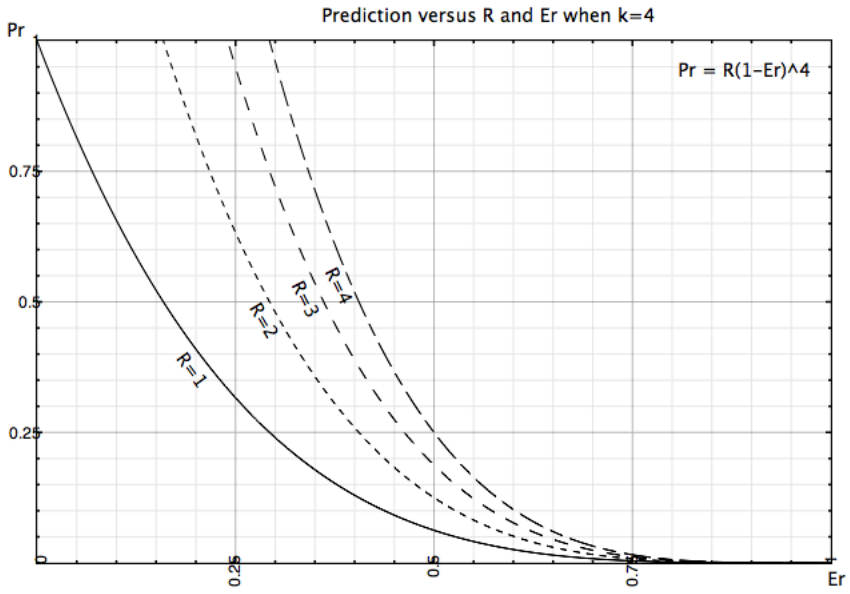


Figure 5: Probability of success Pr versus Er under varying R with division factor $k = 4$

as a substitute to existing strategies. This is due to the fact that prediction does not depend on site capacity but rather on the accuracy of the prediction and the workload model in the GCE. This results in the ability for prediction mechanisms to enhance existing strategies to assist in the assurance of the completion of a job, while trying to make use of the entire site's capacity. When both pro-active and passive methods are combined, premature job terminations resulting from environment failures should be considerably reduced.

5 Simulation Environment

In our experiments, we use a workload model based on [22] to generate synthetic workloads consisting of both massively parallel and embarrassingly parallel jobs.

We investigate several operating environments in order to ascertain the different performance that will be exhibited under various circumstances. We map the simulations based on table 1. We base these environments on the fact that it is possible to distinguish GCEs into Dedicated Grids, Desktop Grids and Hybrid Grids. The general availability levels of these GCEs are stated as a fixed index but is randomly generated for every node in the environment. These normally generated values for the entire site has a mean of the availability value and a variance of 0.2. This provides a fluid environment where by actual $MTTF$ and $MTTR$ values are unknown but the expected performance of the site is known.

We refer to Dedicated Grids as those that are pre-planned and negotiated. These Grids are typically made up of servers, clustered computers and super-computers. Dedicated teams of people or organization are also usually tasked to ensure the availability of these resources. This results in high expectation of the resources being on-line and the Grid capacity is usually known. Such GCEs are usually results of high level collaborations

Type	Availability	Run-Factor
Dedicated Grid (DG)	0.9	[0.1, 0.01, 0.001]
Desktop Grid (EG)	0.3	[0.1, 0.01, 0.001]
Hybrid Grid (HG)	0.5	[0.1, 0.01, 0.001]

Table 1: Table of Simulated Environments

between institutes. Examples of such Grids include the UK e-Science Grid[2], the Asia-Pacific Grid[3] as well as the NC BioGrid[4]. We assume the availability of such grids to be 90%. We assume that 10% of the time when resources are unavailable is due to maintenance, service outages, software upgrades and any other elements that might contribute to connectivity issues to other GCE resources worldwide.

Desktop Grids operate in an environment that is more dynamic and voluntary. Such Grids operate very much in a peer-to-peer fashion, where resources join or leave the Grid without any pre-arranged schedule. These Grids are typically made up of desktops or portable devices and are participated by users who do not usually know who else is also providing computation capability to the cause. The true capacity of such a GCE is thus hard to obtain at any instance in time as these computational resources can go offline regardless of the job state allocated. Examples of such Grids include Seti@Home[5], Korea@Home[6] and Folding@Home[7]. Availability values of such grids can fluctuate given the type of users participating in the GCE. In our case, we assume that participants of such GCEs would be home users who power off their resources at the end of each day. If the home user is to provide their computing resource for 8 hours a day, corresponding to the amount of working time in the day, this would result in approximately 30% availability of the resource to the Grid. If this assumption is stretched to include users worldwide, the average availability would continue to hover around 30%, at times providing more or less in transient depending on the timezone. In our simulations, based on the selected workload model, this assumption of an average availability of 30% availability seems to be consistent. Even if more resources are available in transient, the frequency of failures of these resources will result in the jobs within the workload undergoing extremely high number of failures. The assumption of 30% availability of resources within the EG GCE provided the best trade off allowing the simulated workload to complete its simulation time within reasonable time. This serves as a good estimate of the performance levels of the algorithms in such a GCE.

Hybrid Grids are Grids that we envision the future of Grids to become, given that this is an environment where both dedicated and voluntary resources will co-exist within a large computing resource pool. Such a Grid allow jobs to make use of dedicated computing resources, at the same time make use of a large dynamic pool of desktop resources when the job's applications determines that it is possible to do so. The assumption made is that there is a 3:7 ratio between DGs and EGs in such a GCE in the future. We take the capacity of such GCEs to be approximately 50%.

In our simulations, each of these type of execution environments are tagged with environmental availability

[2]http://www.grid-support.ac.uk/
[3]http://www.apgrid.org/
[4]http://biogrid.icm.edu.pl/
[5]http://setiathome.ssl.berkeley.edu/
[6]http://www.koreaathome.org/
[7]http://folding.stanford.edu/

values such that the average availability values are maintained. This however does not dictate the state of resources at any point of time.

The Run-Factor of the simulated environments describes the ratio of the maximum simulated runtime of a job to the mean $MTTF$ values. As workload models usually generates workloads that have much lesser jobs requiring very long run times, this value is used to induce situations where there will be a high volume of job failures in the environment. It is noted that we do not present the simulation results when the Run-Factor ratio is 1 as this will result in the maximum simulated runtime of the job to be equal to that of the $MTTF$. In which case, we observe no job failures in many of our simulations and is thus unable to study the effects of the various pro-active failure handling schemes.

5.1 Performance Indicators

In order to measure the performance of the modified algorithms, we capture the job failure and rejection rates in each run. We define a job to have failed when its execution is terminated due to a resource failure. A job is rejected when its resource request exceeds what is stated available in the scheduling algorithm. The job processing rate was also captured as an indication of throughput of the resulting algorithm. The $TotalQueueCompletionTime$ is given as the time it takes for the first job to enter the GCE, till the time where the last job leaves the GCE. We compute the various indexes as follows.

1. Job Processing Rate (JPR):

   $$JPR = \frac{NumberOfJobsSuccessfullyCompleted}{TotalQueueCompletionTime} = \frac{J_{Success}}{T_Q}$$

   A higher JPR will indicate larger number of successfully completed jobs or a lower queue completion time. A high JPR will therefore indicate that an algorithm is capable of high throughput.

2. Job Failure Rate (JFR):

   $$JFR = \frac{NumberOfJobsFailedAtRuntime}{TotalQueueCompletionTime} = \frac{J_{Fail}}{T_Q}$$

   A low JFR is desired as it signifies the number of jobs failing during the course of its queue completion is low. This indicates that a strategy is able to allocate resources will to reduce the number of jobs failing in its course of execution.

3. Job Rejection Rate (JRR):

   $$JRR = \frac{NumbeOfJobsRejected}{TotalQueueCompletionTime} = \frac{J_{Rej}}{T_Q}$$

   A low JRR indicates the ability of an algorithm to handle all types of jobs submitted to the queue based on the workload model used. A high JRR will therefore mean that the algorithm is unable to execute jobs due to insufficient capacity. A low JRR is thus desired to indicate that an allocation strategy is able to handle the workload presented using the workload model.

6 Improving Resilience of Existing Algorithms

From Section 4, we have obtained two mechanisms whereby it is possible to pro-actively circumvent the possibility of failures during the course of job execution. This is achieved by (1) making use of individual node MTT values and predicting the availability of each node over a course of time τ, or (2) by using the GAE to obtain the long term capacity of the GCE. In both cases, once the expected capacity or the prediction of availability is available in the scheduling mechanism, it is possible for the scheduler to make informed decisions in its execution schedule. This is inherently different from other passive techniques [16, 15, 14]. While passive mechanisms, the scheduler is typically unaware of the Grid state *prior* to scheduling and only *reacts* to job failure when it detects abnormalities in the job, pro-active mechanisms allocates jobs based on past and existing states of the Grid. It does so in a manner that it best *avoids* any possible event of failure that can occur when the job is submitted.

6.1 Pro-active failure handling strategies

In this section, we introduce 3 pro-active strategies to assist in job allocation to avoid job failures. They are :

1. Site availability based allocation (SAA strategy)
   In this method, we make use of the GAE to estimate the largest job that the GCE is capable of accepting in the long run and reject the submissions of job requirements that are larger then the GAE computed capacity. This acts on the fact that resources are wasted when jobs that are allowed into the GCE fail during their execution. This avoidance of jobs that can cause this situation will therefore allow the remaining jobs to have a higher probability in executing successfully.

2. Node availability based allocation (NAA strategy)
   We make use of Equation (7) in this mechanism to obtain a sorted set of nodes with decreasing probability of staying in the UP state over a jobs expected runtime. Jobs are then only allocated to this set of nodes in order to ensure a higher probability of completion. This strategy tries not to cause a synthetic reduction in the number of resources available in the GCE as it tries to utilize all available resources at any point of time. This is unlike mechanism (1) where the estimated capacity will always be less then that of the total GCE capacity.

3. Node and Site based allocation (NSA strategy)
   This method combines mechanism (1) followed by (2) in order to first ensure that the job requirements are realistic in view of the long term availability of the GCE, followed by a resource allocation strategy such that the resources the job is dispatched to will have a higher probability to complete its execution. We use this strategy to observe if there is any significant advantage in the increase in allocation complexity versus results.

Based on these strategies, we will modify existing algorithms to ascertain the performance for each of these allocation schemes. In addition, we will propose an extension to the algorithm proposed in [23] by the addition of a probability dimension. We discuss the modifications to the algorithms in section 6.2.

6.2 Modifications to Algorithms

In order to verify the capabilities of pro-active failure handling within scheduling algorithms, we implemented SAA, NAA and NSA into the following algorithms for comparison:-

1. Backfill Algorithm [20] (BF)
2. Replicaiton Algorithm [21] (REP)
3. Multi-Resource Scheduling Algorithm [23, 24] (MRS)

In the above algorithms, BF and REP were selected as they are well known algorithms. BF serves as a baseline for comparison, allowing us to observe the advantages in implementing pro-active failure handling techniques in traditional algorithms for GCEs. The REP algorithm is implemented with a replication factor of 2. This provides a mechanism that allows us to be able to observe the advantage of combining predictive mechanisms with more common failure prevention techniques.

The MRS algorithm we have presented in [23, 24] was also extended as a novel approach to allocating resources with considerations of availability in the GCE. MRS is a scalable multi-dimensional strategy where by resource and network capacities are taken into consideration together with job requirements to derive a allocation schedule. The extension to MRS was simply done by extending an additional dimension within MRS. We refer to the modified version of MRS as 3D-MRS. This additional dimension is included as a Availability Index ranging between 0 and 1. This corresponds directly to the Pr_{UP} for each resource. In a similar fashion described in [23, 24], resource selection under MRS is based on the minimum euclidean distance to the origin based on values provided by all three axes. This allows us to consider factors such as computation, data as well as availability provided by that of a GCE resource with only linear increase in computational complexity of the allocation strategy.

In all three cases, SAA was implemented with no change in the scheduling strategy other than adding a filter before the actual allocation stage within the algorithm. This serves as a filter point that rejects jobs that exceeds the GAE percentage value of the GCE. In BF and REP, the NAA strategy was implemented by computing the Pr_{UP} value of all the available GCE resources in the period of τ defined as the runtime of the job. These values are then sorted in a decreasing order. Jobs are then allocated to these resources in the order sorted so as to provide allocation to resources that are more likely available. For MRS, NAA was implemented as a third dimension to the allocation strategy and the resource availability considered during the computation of the euclidean distance determining the "goodness of fit" to the intended resource. As described in [23, 24], the lower this value is, the better the defined resource is for allocation.

The NSA strategy for BF, REP and MRS are implemented as a combination of SAA and NAA. A filter is used to first reject jobs requesting for resources greater than the computed value of general availability of the GCE, and the availability of the individual nodes computed and sorted to obtain the nodes that are predicted to be more likely available. Jobs are then allocated in the order similar to that in the NAA strategy.

7 Results and Discussions

Based on the setup in Table 1, we setup a simulated GCE to generate resource failures to achieve the required capacities. The results are shown in Figure 6, 8 and 7. We normalize[8] the results to that of the unmodified BF algorithm in order simplify the comparisons between the various strategies.

It would be interesting to note that in Cluster/HPC-like systems, the effect of the addition of pro-active failure algorithms to standard algorithms should not differ much. This is because the strategy is based on the availability of the GCE and is additive in nature. It is the base allocation algorithm that continues to derive the optimal schedule to allocate resources matching the allocatable jobs. We would expect that the base algorithm implemented for job allocation would execute more effectively, potentially increasing the job processing rates due to closer proximity. However, if the allocation algorithm, such as MRS is able to inherently consider the match between job resource requirements and node failure prediction, we would expect the JPR to further improve and the JFRs to further decrease.

7.1 Performance of the unmodified algorithms

From the results, it is noted that there is always an improvement in JPR when comparing the unmodified version of MRS to that of BF within all three simulated environments. In DGs, we also note that the JPR of MRS is about 20-30% better than that of BF. This is consistent with our results presented in [23, 24]. It is noted that in some cases, where the JFR data is not available, the value of that is 0.

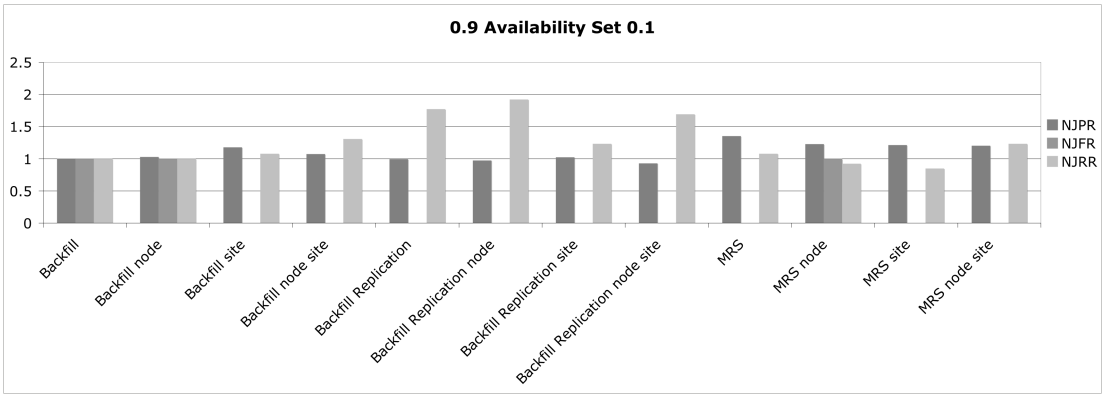
We also note that while REP might not always provide a higher JPR, its JFRs is consistently better than that of BF in all simulated environments. This again, is consistent with our expectation that replicating jobs should provide a greater likely-hood of job success. However, it is clear that this is done at the expense of throughput due to the effectively reduced capacity of the GCE due to replication.

It is noted JRRs in all simulated environments fluctuates. This is due to the fact that the changes in JPR and JFR can result in instances whereby there are more resources in the GCE at different instances in time. However, it is clear from the comparisons of the unmodified algorithms that both REP and MRS outperforms BF in terms of JFR and JPR. We also observe a much lower JRR with MRS compared to the other strategies. This can be due to the increase in throughput, thereby allowing more resources to be available in a shorter period of time.

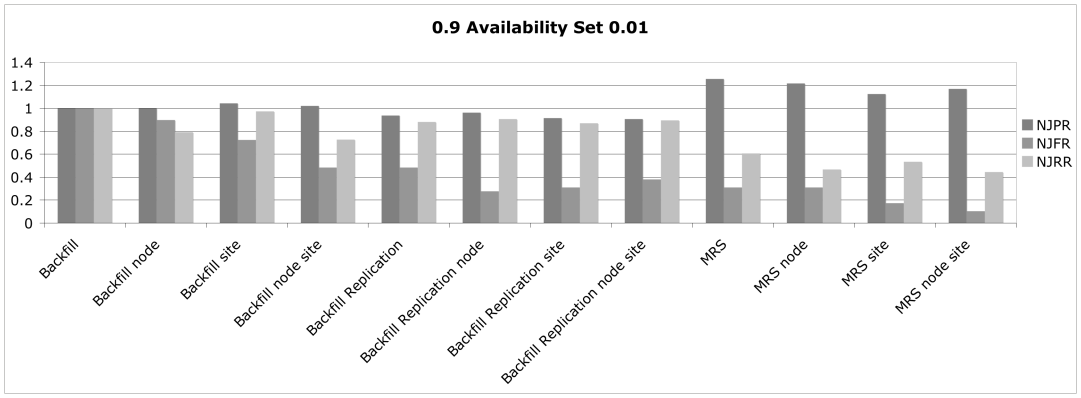
7.2 Performance of the modified algorithms in a DG environment

From the graphs shown in Figures 6, we observe that under a DG environment, BF is not able to derive much benefit from NAA. Making use of SAA or NSA type strategies however provides at least a 40% improvement in JFR and possibly increasing JPRs by up to 30% when Run-Factors are low. This can be due to the fact that DG environments tends to be online far longer than the $MTTF$ of the resource. Thus, predictions of exactly when a node will fail does not provide as much advantage as limiting the average resources available based on the entire site, as in the SAA and NSA strategy. The simulations also show that there is definite improvement in the assurance of job completion when pro-active strategies are introduced.

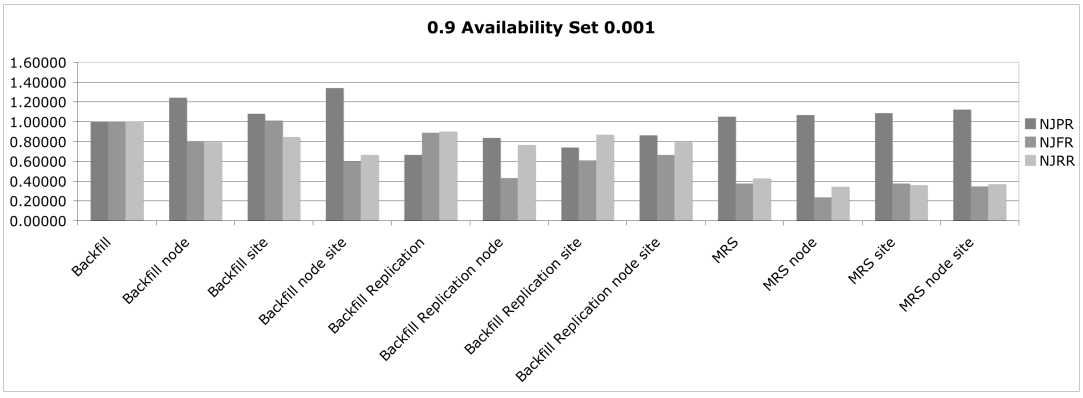
[8] Note that the prefix of "N" before JPR, JFR and JRR represents "Normalized"



(a) DG Environment with Run Factor 0.1

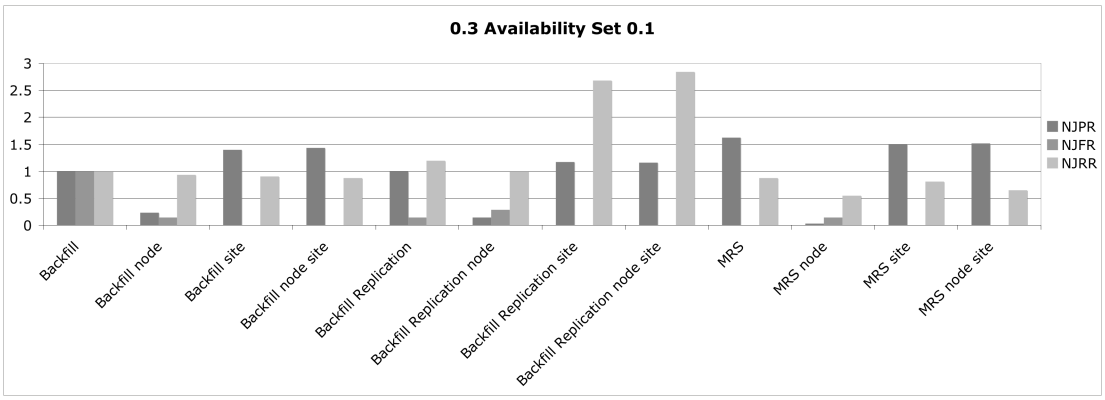


(b) DG Environment with Run Factor 0.01

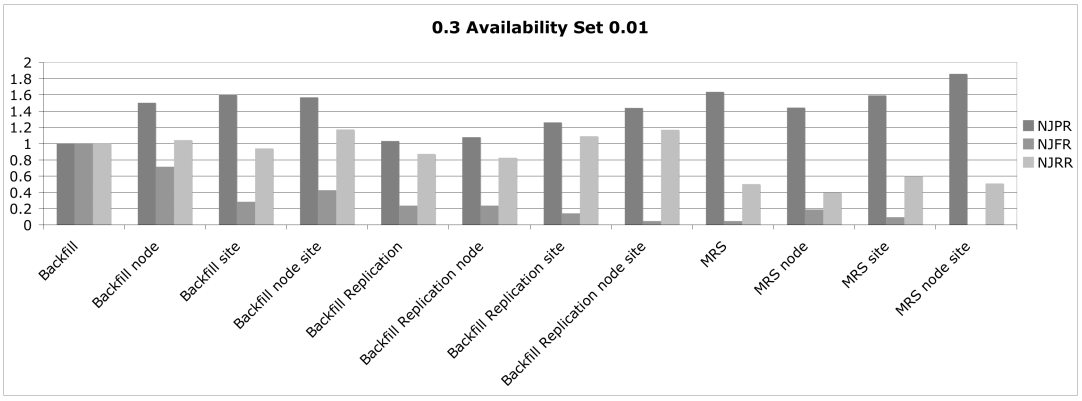


(c) DG Environment with Run Factor 0.001

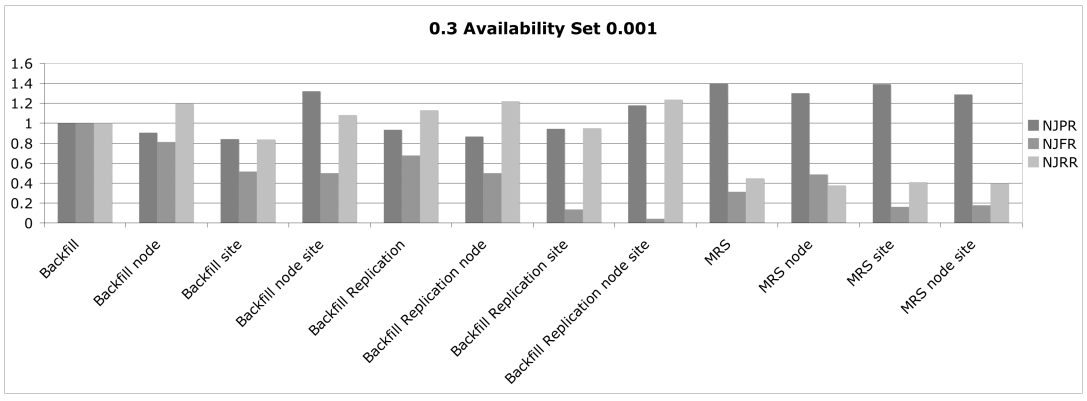
Figure 6: Simulation results for DG under different Run-Factors



(a) EG Environment with Run Factor 0.1

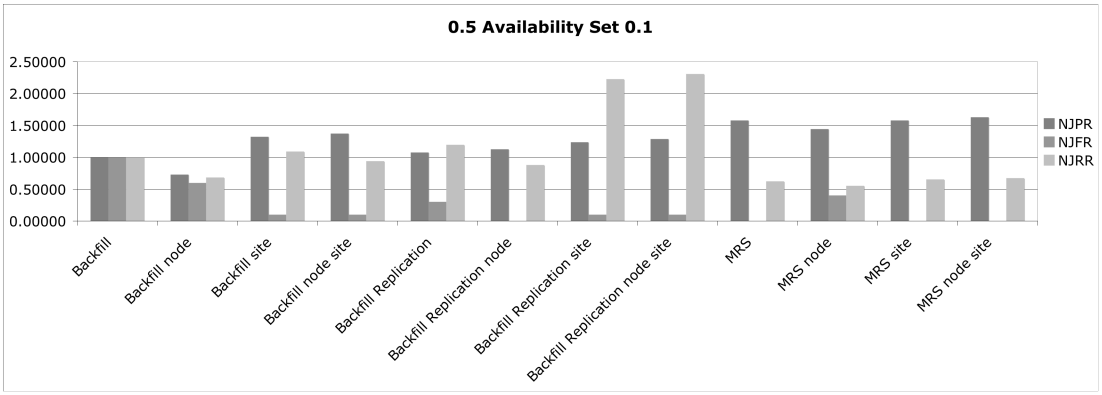


(b) EG Environment with Run Factor 0.01

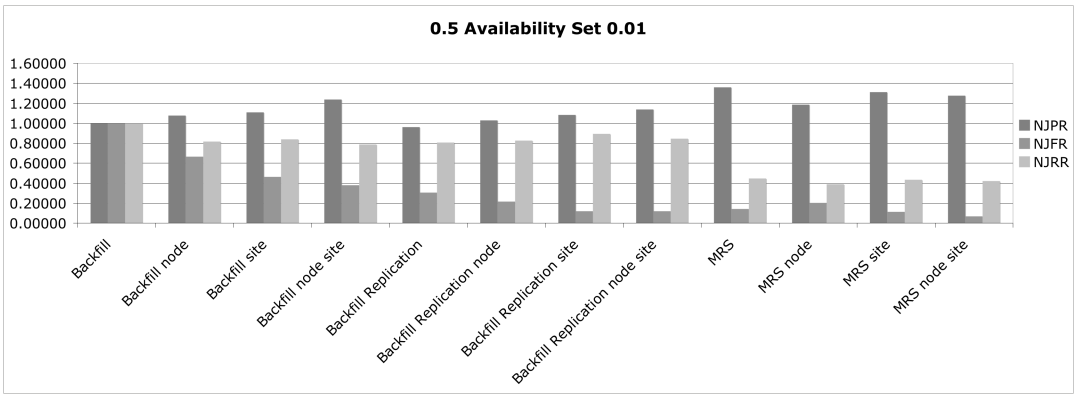


(c) EG Environment with Run Factor 0.001

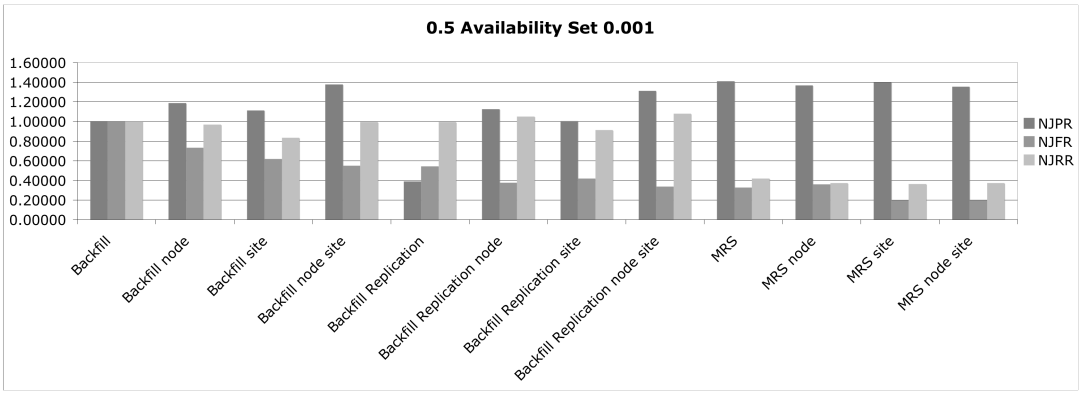
Figure 7: Simulation results for EG under different Run-Factors



(a) HG Environment with Run Factor 0.1



(b) HG Environment with Run Factor 0.01



(c) HG Environment with Run Factor 0.001

Figure 8: Simulation results for HG under different Run-Factors

The benefit of pro-active methods are also observed when introduced to the REP algorithms. In high Run-Factor situations, it is observed that although NAA type strategies are not able to reduce JRRs perhaps due to mistaken estimates in capacities in the GCE, the SAA strategy is able to show better performance through lowered JRRs values under this cirumstances. This is due to better management in resources, allowing more jobs to run before the simulation terminates. However, under Run-Factors of 0.01 and 0.001, we observe that the NAA type strategy performs marginally better than SAA. This can be due to the percieved changes in resource states that is not predicted accurately in situations where Run-Factors are 0.1. NSA, in general, derives its performance gains as a combination of SAA and NAA. This can be observed in the marginal improvements in the NSA strategy compared to either SAA or NAA. This is once again consistent to the additive nature of predictive pro-active failure handling strategies to other forms of failure handling techniques.

In the operation of 3D-MRS in DGs, we observe little benefit in the use of all three pro-active failure handling techniques. In fact, in high Run-Factor situations, the NAA strategy causes a higher JFR which is likely due to errors in node availability prediction. This lack in performance improvement can be due to the much higher throughput exhibited in the MRS algorithm compared to both BF and REP. This allows the job queue to be processed rather quickly before the resources exhibits failure. Therefore, this leads to the lack of improvement from the original JFR and JRR values as it was already allocating the resources to the jobs in a near optimal fashion.

In general, we find that under a DG environment, the inclusion of SAA, NAA or NSA into the selected algorithms provides marginal performance improvement over the original. While a decrease in JFR is observed, depending on the requirement of the GCE, one might feel that these marginal performance gain might not justify the inclusion of a proactive strategy, especially the NAA or NSA strategies. However, in view of the complexity of implementation, we suggest that strategies operating within DG environments to include SAA which is both simple to implement and invokes negligible overheads due to the filtering nature of its strategy. This will allow the strategy to continue providing inherent advantages of the algorithm while maintaining the ability to cope with changes in the GCE capacity. From the implementations compared, it was concluded that the the modified MRS provided the best balance in performance and prevention of job failures while utilizing the SAA modification.

7.3 Performance of the modified algorithms in a EG environment

In an EG environment, it is noted that resources join or leave the GCE fairly often resulting in an overall decrease in GCE capacity even though the resources participating can be large. In the normalized results, it was observed that REP and 3D-MRS continues to provide improvements compared to BF, exhibiting noticeably lower JFR. It was also noted that JRRs in REP strategies are much higher. This is due to the perceived capacity of the GCE when considering the result of the GAE, causing the SAA strategy in REP to reject jobs that are possibly over requesting resources from the environment. The NAA strategy when applied in REP resulted in less JRRs due to the lack of pre-filter of jobs, but exhibits a definitely higher JFR as jobs can fail due to mis-predictions as well as changes in resource states.

These detriments, however, are not observed in 3D-MRS. 3D-MRS consistently exhibits higher JPR, lowered JFRs and JRRs. In the cases where JFRs of the modified MRS strategies exceeds that of REP, the JPRs of

these algorithms always exhibits a much higher value. This signifies that the strategy is able to adapt itself, sacrificing some jobs in view that the entire job queue can be processed faster.

There is however, an exception of the 3D-MRS strategy modified with NAA that exhibited very poor JPR values when the Run-Factor is at 0.1. This can be due to mistakes in resource state prediction due to the volatility of the resources. However, it was found that in such cases, SAA modifications provides very good results, where the jobs that are executed experienced either no failure, or a 50% improvement over the NAA strategy. Similar improvements were also observed in the NSA strategies where there is also a slightly reduced rejection rate of 0%-10% with the aid of node prediction occurring after filtering from SAA. This is observed in all Run-Factors for 3D-MRS.

In such EG environments, we therefore conclude that making use of 3D-MRS with the modification of NSA provided the most reasonable performance while reducing JFRs. This allows greater assurance of job completion when executing in a volatile environment such as a EG.

7.4 Performance of the modified algorithms in a HG environment

It is noted that in HG environments, the performance of the modified BF, REP and 3D-MRS strategies falls intermediate to the extremes represented by both DG and EG environments. It is noted that 3D-MRS with NSA continues to provide the best balance in terms of JPR while exhibiting the lowest JFRs. At the same time, JRR is kept to a minimum. Once again, this can be due to the fact that MRS as a strategy tries to optimize the throughput of the jobs within the GCE. Indirectly, this results in a higher level of resources available for processing. This thus allows for more jobs to enter the system before being filtered as being classified as unable to run.

The continual improvements of the BF and REP strategies, once again, highlights the additive nature of SAA and NAA. Observation of the simulation results clearly shows the advantage of introducing the SAA, NAA or the NSA strategy under different GCEs, workloads as well as algorithms. The continual advantage of NSA also highlights that both the proposed SAA and NAA algorithms can be implemented, without detriments, while continuing to maintain the best balance of performance and minimizing job failures in all cases.

8 Conclusions

In this paper, we have presented three forms of pro-active failure handling strategies, mainly (1) the Site availability based allocation strategy (SAA-strategy), (2) the Node availability based allocation strategy (NAA-strategy) and (3) the Node-Site availability based allocation strategy (NSA-strategy). We simulated three different types of GCEs in order to try to capture different possible types of resource capacities in Grids. The backfill and replication algorithms were modified and used to allow us to observe the advantages of the different pro-active strategies. 3D-MRS, which is an extension to the MRS strategy presented in [23, 24] is also presented with integration to the various pro-active failure handling strategies. The results clearly shows the continued advantage in utilizing the MRS model in resource allocation, and clearly demonstrates the ability of the MRS strategy to be able to extend itself and cope with failure.

Through our experiments, we were able to show that the inclusion of any type of pro-active handling mechanism is able to cause a significant improvement above conventional algorithms. Pro-active strategies also have an additive effect, which is observed in the simulations involving the replication algorithm, where original advantages of the strategy is preserved. This provides higher assurance to the availability of resource allocation both pre and post runtime. Our simulations has also shown that including NSA strategies into various resource allocation strategies is able to demonstrate improvements whereby job failures are significantly reduced. The superior performance of 3D-MRS with the failure handling strategies in the simulations also shows conclusive evidence that the resource allocation strategy is able to handle failures effectively and optimally under various operating environments, when compared to backfill and replication algorithms.

It may be noted that failure prediction mechanisms for non-grid systems can also be used in our context if the mechanism allows for multiple elements being considered in the prediction. Since it is possible that multiple resources can be allocated to a single job, a low MTTF in any of the resources would mean that the probability of the job requirements not being met would occur very soon, possibly before even the job completes. This would also mean that the entire job would have failed. This is especially true in tightly coupled jobs, working on loosely coupled systems. For instance, studies related to power-grid systems described in [25] can serve as an allied technique. The work in [25] attempts to combine structural information with statistical analysis thus proposing a predictive mechanism. Thus, as long as the non-grid based failure prediction mechanism is able to handle the same scope as the above, it is entirely possible that such methods can also be used. Similarly it has been argued in [26] that failure prediction mechanism is an effective solution to prevent HPC clusters from data losses for maximizing MTBF. Our results also conclusively show that the inclusion of pro-active failure handling strategies is able to reduce job failures during runtime. The ability to predict the resource states thus paves way for higher assurance of a successful job execution when jobs are dispatched into a GCE.

Future work can definitely be done to discover the best passive methods that would work with pro-active methods to further address failures that occur during runtime. The contributions in this paper therefore conclusively demonstrate that pro-active failure handling strategies can lead to better Grid scheduler performance especially in a GCE experiencing any form of failure. The extension of the MRS allocation strategy also continues to perform much better when compared to other common algorithms in the GCE.

References

[1] Platform Computing, http://www.platform.com/Products/Platform.LSF.Family/

[2] Sun Grid Engine, http://gridengine.sunsource.net/

[3] United Devices, http://www.ud.com/index.php

[4] XGrid, http://www.apple.com/server/macosx/features/xgrid.html

[5] Chillarege, R.; Biyani, S.; Rosenthal, J.,"Measurement of failure rate in widely distributed software", *in the Twenty-Fifth International Symposium on Fault-Tolerant Computing*, 1995. FTCS-25. Digest of Papers., Pg 424-433, 27-30 Jun 1995.

[6] Liyang Xie, Jingyu Zhou, Xuemin Wang, "Data mapping and the prediction of common cause failure probability", *in the IEEE Transactions on Reliability*, Vol 54, Issue 2, Pg 291-296, June 2005

[7] R. Xiaojuan, S. Lee, R. Eigenmann and S. Bagchi, "Prediction of Resource Availability in Fine-Grained Cycle Sharing Systems", *in the Journal of Grid Computing*, Vol 5 No. 2, Pg 173-195, 2007.

[8] Woochul Kang, Grimshaw A, "Failure Prediction in Computational Grids", *in the 40th Annual Simulation Symposium 2007, ANSS 07*, Pg 275-282, March 2007

[9] J. Brevik, D. Nurmi and R. Wolski, "Automatic Methods for Predicting Machine Availability in Desktop Grid and Per-to-peer Systems", *in the 2004 IEE International Symposium on cluster and the Grid*, 19-22 April, Pg 190-199, 2004

[10] R. Medeiros, W. Cirne, F. Brasileiro and J. Sauve, "Faults in Grids: Why are they so bad and What can be done about it?," *in the proceedings of the Fourth international Workship on Grid Computing (GRID'03)*, 2003.

[11] M. Litzkow, M. Livny and M. Mutka, "Condor - A hunter of Idle Workstations," *in the Proceedings of the 8th International Conference of Distributed Computing Systems*, pp. 104-111, June 1988.

[12] James Frey, Todd Tannenbaum, Ian Foster, Miron Livny, and Steven Tuecke, "Condor-G: A Computation Management Agent for Multi-Institutional Grids", *Proceedings of the Tenth IEEE Symposium on High Performance Distributed Computing (HPDC10) San Francisco*, California, August 7-9, 2001

[13] Ayyub, S., Abramson, D., Enticott, C., Garic, S., Tan, J. "Executing Large Parameter Sweep Applications on a Multi-VO Testbed", *CCGrid 2007*, Brazil.

[14] V. Subramani, R. Kettimuthu, S. Srinivasan and P. Sadayappan, "Distributed Job Scheduling on Computational Grids Using Multiple simultaneous Requests", *in the Proceedings of 11th IEEE International Symposium on High Performance Distributed Computing HPDC-11*, 2002 (HPDC'02), Edinburgh, Scotland, July 24-26, 359-368, 2002

[15] H. M. Lee, S. H. Chin, J. H. Lee, D. W. Lee, K. S. Chung, S. Y. Jung and H. C. Yu, "A Resource Manager for Optimal Resource Selection and Fault Tolerance Service in Grids", *in the Proceedings of 4th IEEE International Symposium on Cluster Computing and the Grid,* Chicago, Illinois, USA, 2004.

[16] S. Choi, M. Baik and C. S. Hwang, "Volunteer Availability based Fault Tolerant Scheduling Mechanism in Desktop Grid Computing Environment", *in the Proceedings of the 3rd IEEE International Symposium on Network Computing and Applications*, Boston, Massachusetts, August 30th - September 1st, pp. 366-371, 2004.

[17] J.H. Abawajy, "Robust Parallel Job Scheduling on Service-Oriented Grid Computing," *O. Gervasi et al. (Eds.): Lecture Notes in Computer Science (LNCS) 3483*, Springer-Verlag Berlin Heidelberg, pp. 1272–1281, 2005.

[18] Zeng Bin, Luo Zhaohui and Wei Jun, "Grid Scheduling Optimization Under Conditions of Uncertainty", *in Lecture Notes in Computer Science,* Volume 4672, 2007

[19] Benjamin Khoo and Bharadwaj Veeravalli, "Cluster Computing and Grid 2005 Works in Progress: A Dynamic Estimation Scheme for Fault-Free Scheduling in Grid Systems," *IEEE Distributed Systems Online*, vol. 6, no. 9, 2005.

[20] V. Hamscher, and U. Schwiegelshohn, and A. Streit, "Evaluation of Job-Scheduling Strategies for Grid Computing", *In the Proceedings of 1st The 1st IEEE/ACM International Workshop on Grid Computing, Brisbane Australia*, 2000.

[21] Y. Li and M. Mascagni, "Improving Performance via Computational Replication on a Large-Scale Computational Grid", *In the Proceedings of IEEE/ACM International Symposium on Cluster Computing and the Grid (IEEE/ACM CCGRID2003)*, Tokyo, 2003.

[22] B. Song, C. Ernemann and R. Yahyapour, "User Group-based Workload analysis and Modelling," *Cluster and Computing Grid Workshop 2005, Cardiff United kingdom, 2005*

[23] Benjamin Khoo B.T, Bharadwaj Veeravalli, Terence H. and Simon S. C. W, "A Co-ordinate Based Resource Allocation Strategy for Grid Environments, *In the Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid , CCGrid 2006,* Singapore, 16-19 May, pp561-567, 2006

[24] Benjamin Khoo Boon T, Bharadwaj Veeravalli, Terence Hung, Simon See, "A Multi-Dimensional Scheduling Scheme in a Grid Computing Environment", *Journal of Parallel and Distributed Computing (JPDC),* Volume 67 , Issue 6 Pages: 659-673, June 2007.

[25] Zengyu H., Pak C. W., Patrick M., Chen Y., Jian M., K. Schneider, and F. L. Greitzer, "Managing Complex Network Operation with Predictive Analytics", *In the Proceedings of AAAI 2009 Spring Symposium Series*, Stanford California, USA, pp.59-65, March 2009.

[26] Pereslavi-Zalessky, "Cluster Hardware Monitoring for Failure Predictive Analysis", RCMS Poster session, Program System Institute of the Russian Academy of Sciences, May 2006 (http://skif.pereslavl.ru/psi-info/rcms/rcms-posters.eng/poster-cluster-monitoring-19-05-2006.pdf)