Multi-Dimensional Resource Allocation Strategy for Large-Scale Computational Grid Systems

Benjamin Khoo Boon Tat

(B. Eng (Hons), NUS)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2006

Abstract

In this thesis, we propose a novel distributed resource-scheduling algorithm capable of handling multiple resource requirements for jobs that arrive in a Grid Computing Environment. In our proposed algorithm, referred to as Multi-Dimension Resource Scheduling (MRS) algorithm, we take into account both the site capabilities and the resource requirements of jobs. The main objective of the algorithm is to obtain a minimal execution schedule through efficient management of available Grid resources. We first propose a model in which the job and site resource characteristics can be captured together and used in the scheduling algorithm. To do so, we introduce the concept of a n-dimensional virtual map and resource potential. Based on the proposed model, we conduct rigorous simulation experiments with real-life workload traces reported in the literature to quantify the performance. We compare our strategy with most of the commonly used algorithms in place on performance metrics such as, job wait times, queue completion times, and average resource utilization. Our combined consideration of job and resource characteristics is shown to render high-performance with respect to above-mentioned metrics in the environment. Our study also reveals the fact that MRS scheme has a capability to adapt to both serial and parallel job requirements, especially when job fragmentation occurs. Our experimental results clearly show that MRS outperforms other strategies and we highlight the impact and importance of our strategy.

We further investigate the capability of this algorithm to handle failures through dimension expansion. Three types of pro-active failure handling strategies for grid environments are proposed. These strategies estimates 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. A variation of MRS called 3D-MRS is presented. The extended algo-

rithm continues 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, achieve a 0% job failure rate during execution. Also, we show that a combination of node based prediction and site capacity filter used with MRS provides the best balance of enhanced throughput and job failures during execution in the algorithms we have considered.

Keywords: Grid computing, scheduling, parallel processing time, multiple resources, load distribution, failure, fault tolerance, dynamic grids, failure handling

Acknowledgments

I would like to express gratitude for my supervisor Bharadwaj Veeravalli for his guidance, advice and support throughout the course of this work. The assistance and lively discussions with him has provided much of the motivation and inspiration during the course of research. This thesis would not have been possible without his guidance, ideas and contributions.

I would also like to express my appreciation for my ex-colleagues from International Business Machines (IBM), IBM e-Technology Center (e-TC) and the Institute of High Performance Computing (IHPC). Without the opportunities from IBM and working with e-TC (John Adams and team), the ideas rooted for this thesis would never have materialized. The involvement in commercial Grid Computing projects with IBM also proved to be a great background to the understanding of real problems faced in the commercial sector. Also a big thank-you to Chia Weng Wai (IBM) for taking the time to explain the perspective of failure in the eyes of the commercial customers.

Many thanks also goes to good friend and colleague, Ganesan Subramanium (IHPC), for our many tea-breaks to discuss ideas that could be used in this research. While some of them might not have worked out, the ideas they represented certainly worked towards to goal of this research. Thanks also goes to Terence Hung (IHPC) for being an understanding manager, and allowing me to combine my work responsibilities and research interest during my stay in IHPC. His guidance and candid comments has also helped refine this work.

Special thanks also goes to Simon See Chong Wee (SUN Micro-systems) for encouraging me to put my initial ideas unto paper which became the basis of this thesis. His initial guidance and perspective in this work was encouraging and invaluable to its outcome.

What i have done during the pursuit of this degree would not have been possible without the support of my family, Veronica Lim. I cannot begin to express the gratitude for her on the sacrifices she made in order for me to

pursue this degree and finally put my ideas unto paper.

I would also like to acknowledge National University of Singapore for giving me the opportunity to pursue this degree with my ideas. Last but not least, i would like to thank anyone i have failed to mention that have made this work possible.

Contents

List of Figures

List of Tables

1 Introduction

With recent technological advances in computing, the cost of computing has greatly decreased, bringing powerful and cheap computing power into the hands of more individuals in the form of Commodity-Off-The-Shelf (COTS) desktops and servers. Together with the increasing number of high bandwidth networks provided at a lowered cost, use of these distributed resources as a powerful computation platform has increased. Vendors such as IBM [1, 2], HP [3] and Sun Micro-systems [4] have all introduced clusters that would effectively lower the cost-per-gigaflop of processing while maintaining high performance using locally distributed systems. The concept of Grid Computing [5] has further pushed the envelope of distributed computing, moving traditionally local resources such as memory, disk and CPUs to a wide area distributed computing platform sharing these very same resources. Consequently, what had used to be optimal in performance for a local cluster has suddenly become a serious problem when high latency networks, uneven resource distributions, and low node reliability guarantees, are added into the system. Scheduling strategies for these distributed systems are also affected as more resources and requirements have to be addressed in a Grid system. The lack of centralized control in Grids has also resulted in failure of traditional scheduling algorithms where different policies might hinder the sharing of specific resources. This leads to a lack of robust scheduling algorithms that are available for Grids.

At the same time, 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, 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.

These increase in awareness, has lead to various products, both in research and commercial, that handles resource allocation and scheduling of jobs to harness these computation powers. Products such as Platform LSF [34] or the Sun Grid Engine [35] provides algorithms and strategies that handles 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 [36] or XGrid [37], 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 the 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 know, estimate and allocate jobs to resources that can fulfill its task with minimal interruptions and re-scheduling. Together with a well designed multiple resource scheduling mechanism, it will ultimately result in higher throughput, and a higher level of quality for jobs submitted to Grids. This motivates to invent new strategies that take into account the failure possibilities to render best services.

1.1 Related Works

There have been other strategies introduced to handle resource optimization for jobs submitted over Grids. However, while some investigated strategies to obtain optimizations in the computational time domain, others looked at optimizations in data or I/O domain. Recently, more creative methods to achieve optimal scheduling have included the concept of the costs of resources in financial terms. Some of these techniques, which are relevant to the context of this paper, will

be introduced below.

In [6], job optimization is handled by redundantly allocating jobs to multiple sites instead of sending it only to the least loaded site. The rationale in this scheme was that the fastest queue will allow a job to execute before its replicas and this provides low wait times and improves turn-around time. Job allocation failures due site availabilities would also be better handled due to this redundancy. However, this strategy leads to problems where queue lengths of different sites are unnecessarily loaded handle the same job. The frequent changes in queue length can also potentially hamper on-site scheduling algorithms to work effectively as schedules are typically built by looking ahead in the queue. In addition, the method proposed does not investigate the problems that can arise when the data required for the job is not available at the execution site and needs to be transported for a successful execution. MRS works to eliminate these issues by allocating only the right amount of resources to jobs that requires it, thus freeing up queues from potentially non-executing jobs.

In [7], Zhang has highlighted that the execution profiles of many applications are only known in real-time, which makes it difficult for an "acceptance test" to be carried out. The study also broke down the various scheduling models into 1) Centralized, wherein all jobs are submitted at a central location for scheduling and dispatching, 2) Decentralized, wherein jobs are submitted at their local locations for dispatching, and 3) Hierarchical model, wherein jobs are submitted to a meta-scheduler but are dispatched to low-level schedulers for dispatching and execution. Effective virtualization of resources was also proposed in order to abstract the resource environment and hide the physical boundaries defined. A buddy set as in [8] was also proposed, and its effectiveness also highlighted in [18], where it was shown that when groups of trusted nodes co-operate, the resulting performance is superior compared to situations where there is no relationship establishment between nodes. However, in both cases, the strategies proposed looks plainly at the computational requirements of a job and does not consider the data resource required. It also does not address

resource allocation pertaining to both serial and parallel job requirements. MRS effectively applies the concept of co-operation and virtualization to exploit the advantages presented in [18, 8], but includes knowledge of bandwidth to account for I/O and communication overheads. While this allows us to apply MRS to both serial and parallel jobs, it also allows us to efficiently schedule in a Grid environment where the data resources are distributed.

In the work presented in [9], the ability to schedule a job in accordance to multiple (K) resources is explored. Although the approach was not designed with the Grid environment in mind, the simulation work presented in [9] shows clearly the potential benefits where scheduling with multiple resources is concerned. Performance gains of up to 50% were achieved when including effective resources-awareness in the scheduling algorithm. Similar resource awareness and multi-objective based optimizations where studied in [21]. In both cases, the limitations of in conventional methods was also identified as there was have no mechanism for utilizing additional information known about the system and its environment. However, in [9], there was no data resources identified, while in [21], we believe that the over simplicity of resource aggregation was in-adequate in capturing resource relationships. MRS proposes a more complex form of resource aggregation that allows for better expression of resource relationships, while maintaining simplicity in the algorithm construction. At the same time we continue to consider multiple resources which includes both computational and data requirements.

In [10] data replication and reuse of resources was looked into as a means of establishing a Grid being able to handle large data (i.e., Data Grid). Elizeu et. al. has looked into the classification of tasks that are processors of huge data (pHD), where by processes require large datasets and data reuse is possible. They introduced a term referred to as Storage Affinity, which takes into account on how reusable is a set of data by pHDs or a bag of tasks. This also determines if a task should be sent to a location where the required data resides or vice versa. Following this, task replication [44] is used to reduce the wait time of

the job. This method is useful to handle pre-replicated or re-usable data but does not address how the data would be best scheduled for applications with no reusable data. However, [10] has demonstrated that it is possible to improve response times for jobs through smart data management. We build on this concept of Affinity in our algorithm, combined with better resource relationship representation, to arrive at a strategy that would allow the overall overheads of data transmission to be minimized. This is done with no detrimental effect on the wait times of a job and the overall queue completion of the Grid environment.

Contributions in [11] considered the idea of replication and further included a data catalog method to discover and the best location to use. Making use of the Network Weather Service [12], it is possible to determine the best node to collect the data from/send a job to. Then, a compute-data pair is assigned with the earliest completion time. This method has again identified that data optimization is critical to the response time of a job. This however, does not exploit resource locality w.r.t the serial or parallel job requirements. This is thus unsuitable for jobs that are highly parallel in nature (i.e., for applications customized for distributed memory systems). We look upon parallel jobs as applications that requirements low latency and high bandwidth, and assign the resource allocation such that both parallel and serial jobs are optimized.

In [13], Ranaganathan et. al. presented that Computation Scheduling and Data Scheduling can be considered asynchronously in Data-Intensive Applications. The study considered External Schedulers, local Schedulers and Data schedulers. It concludes that data movement and computation need not always be coupled for consideration together. While this might be true, and demonstrated in [11], through High Energy Physics applications, this is not always the case when MPICH-G2 type applications [14, 17] are concerned. MRS recognizes parallel job requirements and, by using affinity and combined resource allocation, decides the best sites for the job to be dispatched to such that everything is in the same path.

Other projects such as the Storage Resource Broker [15] and OGSA-DAI

[23] mainly concentrates on assisting the access and integration of data in a distributed computing environment such as a Grid. By itself, these middle-ware does not decide nor allocate the availability of data resources.

While many other works such as [19, 20] continues to provide algorithms to effectively allocate resources, much of these works on the premise of [13] where data and computation resource requirements are handled separately. While these mechanisms are shown to be effective in Monte-Carlo or parameter sweep type applications where the tasks or sub tasks are considered to be independent, we hesitate to generalize on its effectiveness when the nature of jobs, such as MPI-G2 parallel class of applications, can lead to inter-resource dependence. Although many of these algorithms work effectively over a known set of resources, the complexity of the strategies makes it difficult to include additional resources to the Grid. MRS seeks to eliminate this limitation to allow additional resource considerations to be easily added for consideration through aggregation and representation of resource dependence. Our simulation demonstrates this aggregation to cater for data and communication overheads while at the same time, taking care of both requirements of serial and MPI parallel application, especially during fragmentation.

While the above literature provides many existing perspectives of resource allocation and scheduling, there has been no proposal on the resource model suitable for Grids and the underlying mechanism to prevent failures of jobs in Grids. We classify the current available work on Grid failures into pro-active and post-active 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. Post-active mechanisms identifies algorithms that handles the job failures *after* it has occurred. In the literature, very few works address failure on Grids. Of those that look into these issues, many works are primarily post-active in nature and deal with failures through Grid monitoring as mentioned in [38]. These methods mainly do so by either checkpoint-resume or terminate-restart [41, 39]. Two

pro-active failure mechanisms is introduced in [40, 44] and [42]. While [40, 44] operates by replicating jobs on Grid resources, [42] 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. While the latter only addresses independent task executing on the resources. It does not address how these resources can potentially co-operate to run massively parallel applications.

1.2 Our Contributions

In order to provide a more robust allocation strategy, we propose a novel methodology referred to as Multi-Dimension Resource Scheduling (MRS) strategy that would enable jobs with multiple resource requirements to be run effectively on a Grid Computing Environment (GCE). A job's resource dependencies in computational, data requirements and communication overheads will be considered. A parameter called Resource Potential is also introduced to ease in situations where in inter-resource communication relations need to be addressed. An n-dimensional resource aggregation and allocation mechanism is also proposed. The resource aggregation index and the Resource Potential sufficiently allows us to mathematically describe the relationship of resources that affects general job executions in a specific dimension into a single index. Each dimension is then put together to form an n-dimensional map that allows us to identify the best allocation of resources for the job. The number of dimensions considered depends on the number of job related attributes we wish to schedule for.

The combination of these two methodologies allows MRS to be able to respond more suitably in the execution of applications that are both highly parallel as well as serial in nature in GCEs. The performance of such a scheduling algorithm promises respectable waiting times, response times, as well as an improved level of utilization across the entire GCE.

As dimensional indices are computed at the resource sites itself, this vastly

improves the distributed control of the Grid over resources. It additionally unloads scheduling overheads due to resource comparison at the main scheduling server. This design also paves way in designing a distributed scheduling system as each additional resource is responsible for its own sharing of resources and computation of indexes. This naturally allows the MRS to be possibly implemented easily as both a central and distributed scheduling systems. In this paper, we restrict the scope of simulation to a central scheduling design of the MRS. However, we will present a discussion how a distributed MRS system can be easily achieved.

We begin our evaluation of the performance of our proposed strategy in 2 dimensions, namely computation and data, while addressing requirements of resources such as, FLOPS, RAM, Disk space, and data. We study our strategy with respect to several influencing factors that quantify the performance. Our study shows that MRS out performs most of the commonly available schemes in place for a GCE. We subsequently expand the same strategy into 3 dimensions (3D-MRS) to handle failure.

Using our pro-active failure model, we conclusively show that it is possible to improve existing scheduling strategies and algorithms such that they are able to prevent job failures during execution. Three strategies are introduced, namely the SAA, NAA and 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 and compare 3D-MRS using these strategies and clearly show the improvement in job reliability by introducing pro-active failure handling to this algorithm using the proposed model.

1.3 Organization of Thesis

In this thesis, we first look at the Grid Computing Model that we will operate in in section 2, investigating the resource environment and failure models in a GCE. We then look at how we would measure the performance of our proposed

strategies in section 2.3. The allocation strategy and algorithm is then described in section 3. This will include Multi Dimension Scheduling and the Failure Prediction model. The extension of a dimension to include failure knowledge in the MRS is then shown in section 3.3. The performance of these strategies are then discussed in section 4. This is followed by a conclusion in section 5 and proposed future work in section 6.

2 Grid Computing Model

In this section, we define the GCE in which the MRS strategy was designed. We also look at the ways a failure can be observed and build a failure model which can be practically used in a GCE. We then investigate the various performance measures that can be used to measure the effectiveness of our allocation strategies.

2.1 Resource Environment for Grid Computing

We first clearly identify certain key characteristics of resources as well as the nature of jobs. A GCE comprises many diverse machine types, disks/storage, and networks. In our resource environment, we consider the following.

1. Resources can be made up of individual desktops, servers, clusters or large multi-processor systems. They can provide varying amounts of CPU FLOPs, RAM, Harddisk space and bandwidth. Communication to individual nodes in the cluster will be done through a Local Resource Manager (LRM) such as SGE, PBS, or LSF. We assume that the LRM will dispatch a job immediately when instructed by the Grid Meta-Scheduler (GMS). The GMS thus treats all resources exposed under a single LRM as a single resource. We find this assumption to be reasonable as GMS usually does not have the ability to directly contact resources controlled by the LRM.

2. Changes in any shared resource at a site is known instantaneously to all locations throughout the GCE. Without loss of generality we assume that every node in the GCE is able to execute all jobs when evaluating the performance of the MRS strategy.

3. Each computation resource is connected to each other through different bandwidths which are possibly asymmetrical.

4. All resources have prior agreement to participate on the Grid. From this, we safely assume a trusted environment whereby all resources shared by

sites are accessible by every other participating node in the Grid if required to do so.

5. We assume that the importance of the resources with respect to each other is identical.

6. The capacity for computation in a CPU resource is provided in the form of GFlops. While we are aware that this is not completely representative of a processor's computational capabilities, it is at current one of the most basic measure of performance on a CPU. Therefore, this is used as a gauge to standardize the performance of different CPU architectures in different sites. However, the actual units used in the MRS strategy does not require actual performance measures, rather, it depends on relative measures to the job requirements. We will show how it is done in later sections.

The creation of the job environment is done through the investigation of the workload models available in the Parallel Workload Archive Models [16] and the Grid workload model available in [25]. The job characteristics are thus defined by the set of parameters available in these models and complemented with additional resource requirements that are not otherwise available in these two models. Examples of these resources includes information such as job submission locations and data size required for successful execution of the task. In our job execution environment, we assume the following.

1. Resource requirement for a job does not change during execution and are only of (a) Single CPU types, or (b) massively parallel types written in either MPI such as MPICH[1] or PVM[2].

2. The job resource estimates provided are the upper bound of the resource usage of a given job.

3. Every job submitted can have its data-source located anywhere within the GCE.

[1]MPICH: http://www-unix.mcs.anl.gov/mpi/mpich/
[2]Parallel Virtual Machines: http://www.csm.ornl.gov/pvm/pvm_home.html

4. A job submitted can be scheduled for execution anywhere within the GCE. Without loss of generality, we also assume that the applications to be executed are already available in all sites within the GCE.

5. Jobs resource requirements are divisible into any size prior to execution.

6. In addition to computational requirements (i.e. GFlops, RAM and File system requirements), every job also has a data requirement where-by the main data source and size is stated. These data resources required are accessible using GridFTP or GASS[3] services provided by the Globus Toolkit.

7. The effective run time of a job is computed from the time the job is submitted, till the end of its result file stage-out procedure. This includes the time required for the data to be staged in for execution and the time taken for inter-process communication of parallel applications.

8. Resources are locked for a job execution once the distribution of resources start and will be reclaimed after use.

A physical illustration of the resource environment that we consider is shown in figure (1), and the resource view of how the Grid Meta-Scheduler will access all resources through the LRM is shown in the figure (2).

2.2 Failure Model for Grid Computing

In this thesis, we define *Failure* to be the breakdown of communication links 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. We do not specifically identify the cause of the failure, but generalize it for any possible kind. We also assume that a failed resource will be restarted and all history of past executions will be cleared. We

[3]Grid Access to Secondary Storage: http://www.globus.org/gass

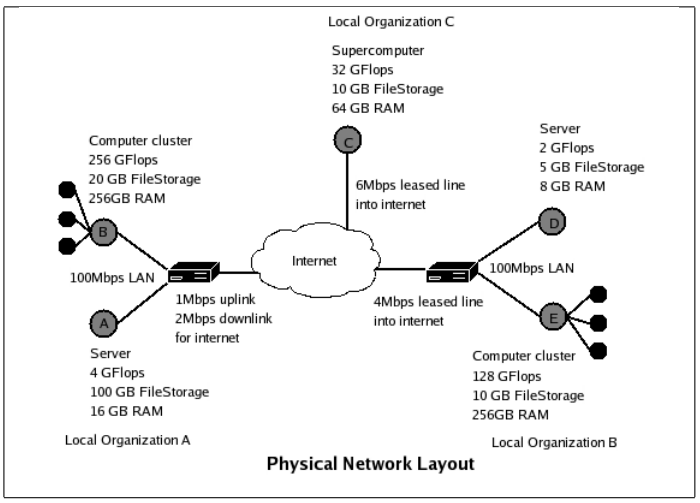


Figure 1: Illustration of a physical network layout of a GCE.

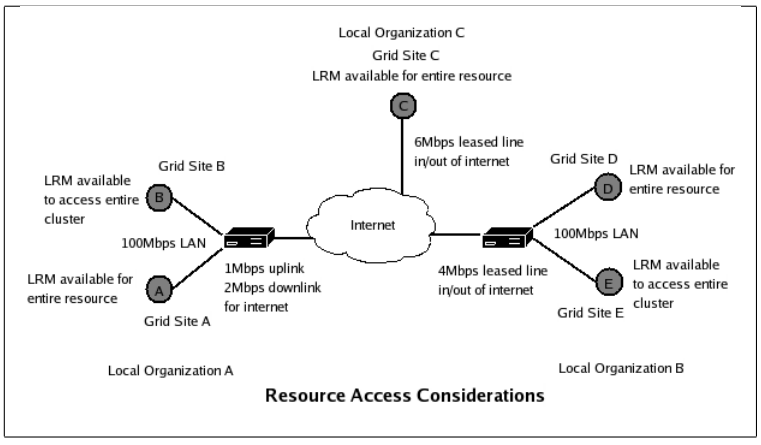


Figure 2: Resource view of physical environment with access considerations

also use the term *availability* and *capacity* interchangeably as they both refer to the number of resources that can be utilized at any point of time.

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
3. Resource encounters a failure in one of its components and goes offline for maintenance and fix

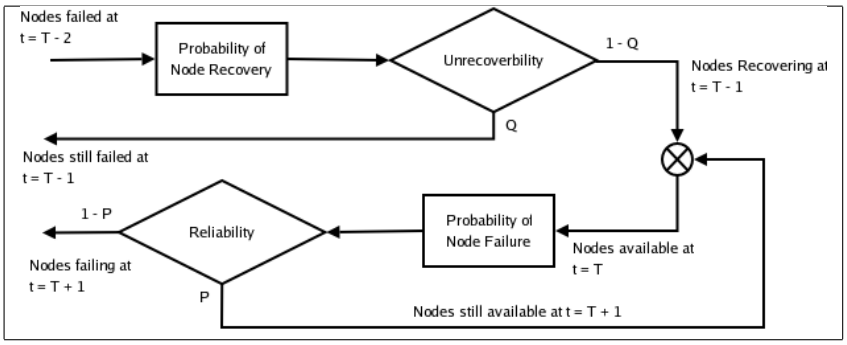


Figure 3: Resource Life Cycle Model for resources in the GCE

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 stages, 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 [43] was constructed. The Resource Life Cycle (RLC) Model shown in Figure 3 identifies the stages where by Grid resources undergoes 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.

The execution environment defined in section 2.1 and the failure model presented in 2.2 allows us to be able to create an environment whereby resources can join or leave the GCE at any point, at the same time, exhibit sudden failures, simulating that in a real environment. Resources will also be consumed and re-injected into the systems as they cycle through different states of load, allowing us to model the GCE subject to different workload models if required to do so.

2.3 Performance measures

In order to verify the effectiveness of the MRS algorithm, we make use of the following metrics of performance measure.

1. Average Wait-Time (AWT)

   This is defined as the time duration for which a job waits in the queue before being executed. The wait time of a single job instance is obtained by taking the difference between the time the job begins execution (e_j) and the time the job is submitted (s_j). This is computed for all jobs in the simulation environment. The average job waiting time is then obtained. If there are a total of J jobs submitted to a GCE, the AWT of a job is given by,

   $$AWT = \frac{\sum_{j=0}^{J-1}(e_j - s_j)}{J}$$

   This quantity is a measure of responsiveness of the scheduling mechanism. A low wait time suggests that the algorithm can potentially be used to schedule increasingly interactive applications due to reduced latency before a job begins execution.

2. Queue Completion Time (QCT)

   This is defined as the amount of time it takes for the scheduling algorithm to be able to process all the jobs in the queue. This is computed by tracking the time when the first job enters the scheduler until the time the last job exits the scheduler. In our experiments, the number of jobs entering the system is fixed, to make the simulation more traceable. This allows us a quantitative measure of throughput, where the smaller the time value, the better. The queue completion time is given by,

   $$QCT = e_J + E_J - s_0$$

   where, E_J is the execution time of the last job. This includes the I/O and

communication overheads that occurs during job execution.

This metric, when coupled with the average waiting time of a job, allows us to deduce the maximum amount of time a typical job will spend in the system for a given workload.

3. Average Grid Utilization (AGU)

   This quantity investigates how well the algorithm is capable of organizing the workload and the GCE resources so as to optimize the performance. Thus, the higher the utilization, the better optimized the environment is. The utilization of the GCE at each execution time step is captured and represented as $U(t) = \frac{M_u}{M}$, where M is the total computational resources available. M_u is the number of computational resources utilized. The average grid utilization is thus given by the following equation.

$$AGU = \frac{\sum_{t=s_0}^{QCT} U(t)}{QCT}$$

However, as these measures are not suited for investigating the effectiveness in event of faults in the GCE, we evaluate the effectiveness in such circumstances by capturing the job failure and rejection rates in each simulation. 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. We compute the various performance indexes as follows.

1. Job Processing Rate (JPR):

$$JPR = \frac{NumbeOfJobsSuccessfullyCompleted}{TotalQueueCompletionTime}$$

   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):

$$JFR = \frac{NumbeOfJobsFailedAtRuntime}{TotalQueueCompletionTime}$$

A low JFR is desired as it signifies the number of jobs failing during the course of its queue completion is low. This thus 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):

$$JRR = \frac{NumbeOfJobsRejected}{TotalQueueCompletionTime}$$

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.

3 Allocation strategy and Algorithms

This sections presents the n-dimensional MRS allocation strategy and the failure prediction model that can be used to augment existing allocation strategies. We then highlight how MRS is extended into 3D-MRS where the new dimension would include the knowledge of availability of a resources.

3.1 Multi-dimension scheduling

As stated earlier, MRS is a n-dimensional allocation strategy. In order to make use of this strategy, the dimensions to consider must first be decided. The dimensions should be the general classifications of resource requirements that would be required by a job. We make use of two basic dimensions (1) Computation, and (2) Data, in our simulations in order to verify the effectiveness of our strategy. These two dimensions are chosen due to the general requirement to achieve faster computation through proper resource allocation such as GFLOPs, RAM and disk, and better data resource allocation to achieve higher I/O throughput. Aggregation of the various available resources are then combined into two major indices based on these two dimensions. We refer to these indices as the Computational and Data Index respectively.

From the two indices, we create a 2-dimensional (2D) plot with the Computation and Data Index. This 2D plot describes the virtual topology of the job resource requirements, situated at the origin, to the resource providing sites in the GCE. We call this virtual topology a *Virtual Map*. It is thus clear that each site has two indices that describes its suitability for the job. The most suited resource providers will be the sites whereby it is located nearest to the origin. The sections below will demonstrate how we construct the two selected dimensions and the process of aggregation that leads to the final aggregated Indexes used in the Virtual Map.

3.1.1 Computation Dimension

Resources in the computation dimension consist of entities that would impact the efficient computation of a job. Each resource is in turn represented by a capability value and a requirement value. In our simulations, we make use of the following allocable resources as basis for scheduling in the computation dimension:

- GFLOP (C)
- RAM (M)
- Disk space (F)

However, we note that this is insufficient to represent a collection of sites and how they can possibly inter-operate with each other. A job submitted to a poorly connected site will be penalized when job fragmentation occurs or when the data required for processing is located in another location.

In order to minimize the detrimental effects in such cases, we introduce a parameter referred to as the *Resource Potential.* This is to assist in the evaluation of the Computation Index. The potential, denoted as P_i , of a resource R_i quantifies the level of network connectivity between itself and its neighboring sites. For simplicity, we assume that the network latencies as well as the communication overhead of a resource is inversely proportional to its bandwidth. With m representing the total number of sites, we refer to the Resource Potential, P_i of a resource R_i, as a form of "Virtual Distance", where $1 \leq i \leq m$. This is computed as $P_i = \sum B_{ij}$ where, B is the upload bandwidth, expressed in bits per sec, from R_i to R_j for $i \neq j$ and $B_{ij} = 0$ if $i = j$. This effectively eliminates all network complexities and "flattens" the bandwidth view of all the resources to the maximum achievable bandwidth between resources. This also inherently includes all sub-net routing overheads and communication overheads when a bandwidth monitoring system such as NWS [12] is employed. We illustrate this "flattening" process in figure (4). The values C, M, F and

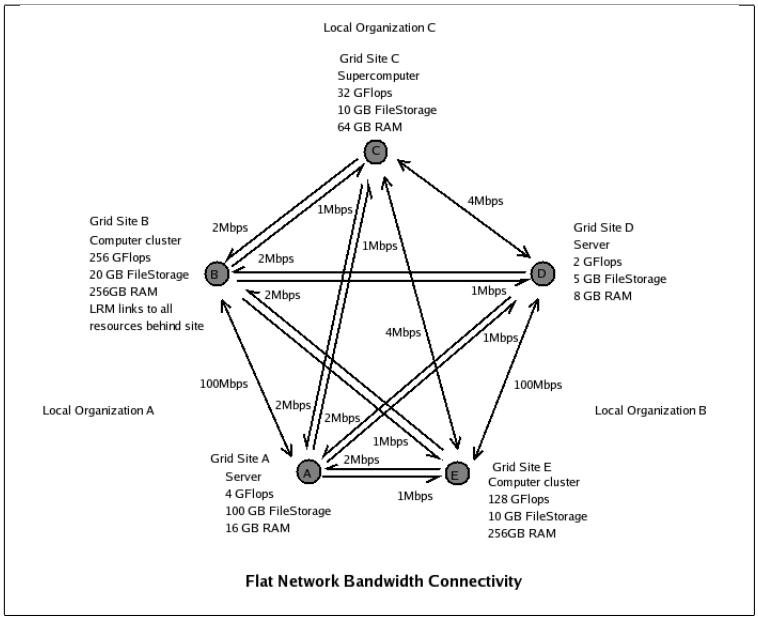


Figure 4: Flattened network view of resources for computation of Potential

P_i dynamically change with resource availability over time t, and is constantly monitored for changes in our simulation. Thus, in a GCE where we characterize the resource environment as a set $S = \{R_1, ..., R_m\}$, we can represent the allocate-able computational resources within a site i as a set $S_c = \{R_i, t\}$ where $S_c \subseteq S$. R_i further represented by 4-tuple of $f_i(< C, M, F, P_i >, t)$ denoting the four resources considered in our allocation strategy.

In order to ascertain an aggregated Computation Index of a site to a job, resources are also requested based on the same GFLOPs, RAM and Harddisk space required. Similar to a node's Resource Potential described earlier, jobs are also additionally characterized by a potential value. However, this potential value is not obtained from the location where the job is submitted from, rather, it is obtained from the location of the source file required for the job to execute efficiently. In our simulations, we assume that each job only requires data from one data resource. This data resource can be either local to the job submission site or remote. As MRS is expected to operate in a GCE, we also simulate

scenarios wherein users can submit jobs from different locations[4].

We characterize the job environment by $J = \{A_i, ..., A_j\}$, and the computational requirement of each job A_j in the set of J jobs is represented by $g_j(< C, M, F, P_{src} >, t)$.

3.1.2 Computational Index through Aggregation

Evaluation of various resource requirements of sites and jobs allows us to aggregate their values and encode inter-resource relationships in order to arrive at a single computational index such that it can be used to obtain the allocation score. This is done by obtaining a ratio of provision (R_{ij}), for site i and job j, between what is requested and what is possibly provided. For computational resources, it is given by, $R_{ij}\{C\} = 1 - \frac{f_i\{C\}}{g_j\{C\}}$. We consider only the positive values of $R_{ij}\{C\}$, such that and $R_{ij}\{C\} = 0$ if the above evaluates to be less than zero. $f_i\{C\}$ and $g_j\{C\}$ are the GFLOP resource provided at site i and GFLOP resource required by job j. We only consider positive values in the Virtual Map, and therefore truncate the values at zero. We make several observations in this equation.

1. Perfect ability to provision for a resource results in this value being 0.
2. Inability to provide for a resource results in $0 < \frac{f_i\{C\}}{g_j\{C\}} < 1$. The $R_{ij}\{C\}$ value would approach 1 as the inability to provision a resource to a job increases.
3. Over-ability to provision resources for a job results in the $R_{ij}\{C\} = 0$.

We apply the same ratio of provision to all resource and requirements within the computational dimension which also includes RAM (M) and Harddisk (F) requirements. Additionally we also include the ratio of provision between the potential value of the site (P_i) and the source file potential (P_{src}). This allows us to evaluate if a site connectivity is equal or better to where the source data file is located. This ensures that the possible target job submission site will not be

[4]Without loss of generality, we have assumed that applications are pre-staged at the sites.

penalized more than required if job fragmentation is to occur, when compared to executing the job in place at the data source location.

These ratios are then aggregated into a resulting dimensionless computation index (x_i) for site i on job j using the following equation. Constants K_C, K_M, K_F and K_P represents weights that provide modification to the importance of the respective provisioning ratios in terms of importance to each other. A value of $0 < K < 1$ signifies a lower relative importance of a specific computational resource while $K \geq 1$ represents equal or greater relative importance when compared to other resources. After the sites providing resources are indexed to obtain x_{ij} , the site i with the lowest computation index, x^*_{ij} is deemed to provide the best resources suited for a job j. In our simulations, we set the Kconstants such that $K = 1$.

$$x_{ij} = \sqrt{\frac{1}{K_C} R_{ij} \{C\}^2 + \frac{1}{K_M} R_{ij} \{M\}^2 + \frac{1}{K_F} R_{ij} \{F\}^2 + \frac{1}{K_P} R_{ij} \{P\}^2} \quad (1)$$

3.1.3 Data Dimension and indexing through resource inter-relation

In the data dimension, we wish to inter-relate resources that would affect the I/O of a job and evaluate an index that aids us in determining a good resource site that would best execute a job. The expected time for I/O is determined based on the estimated data communications required and the bandwidth between the source file location and the target job allocation site. The ratio between the I/O communication time to the estimated local job runtime is then taken. This ratio allows us to evaluate the level of advantage a job has in dispatching that job to a remote site. This is because a site capable of executing a job locally would incur a minimal (not-zero) I/O time as compared to any other remote location. Thus, allocation of a job to the intended target resource should be one whereby this ratio is as low as possible.

The I/O time is mainly dependent on the availability of bandwidth at a site. The available bandwidth also changes over time depending on if a resource

is sharing any of its network resources with other resources in the GCE. This is also captured as a sequence of complete network allocation for a job in our simulator. We annotate bandwidth B between two sites i and j as $B_{ij} = min\{B_{ij}^{downlоad}, B_{ji}^{upload}\}$ which changes over time t as data capabilities of a resource $S_d\{R_i, t\}$. Where each item in this set is represented by $d_i\{< B >, t\}$. The data requirement of a job j is thus represented by $e_j\{< F, A^{runtime} >, t\}$ where $A^{runtime}$ is the estimated runtime of the job.

We make use of this ratio to create the Data Index. This evaluation is an example of aggregation based on resource inter-relation. I/O time is affected by the amount of data for a job and the actual bandwidth resource available. In the worst case scenario, the amount of data required for the job would also be the amount of hard-disk resource required at the site to store the data to be processed. This, therefore inter-relates the data resources to the bandwidth resources available. The ratio is written as follows.

$$y_{ij} = \frac{e_j\{F\}}{d_i\{B_{ij}\}} . \frac{1}{A^{runtime}} \tag{2}$$

It is noted that y_{ij} continues to be dimension-less and a smaller value would represent a better site i preference when compared to a larger one. An (ascending) ordered y_{ij} would rank sites with the better advantage in handling job fragmentation compared to those ranked later.

3.1.4 Dimension Merging

From the individual Computation and Data Indices described above, we observe that the best allocated resources are represented by those with low index values. Each of the individual indices are also encoded with resource requirements considerations in its evaluation through aggregation. These points when plotted on a 2-dimensional axis creates what we termed as the Virtual Map that is described in section **??**. As we have observed, sites that position themselves closest to the origin are those that deviate from the resource requirements by

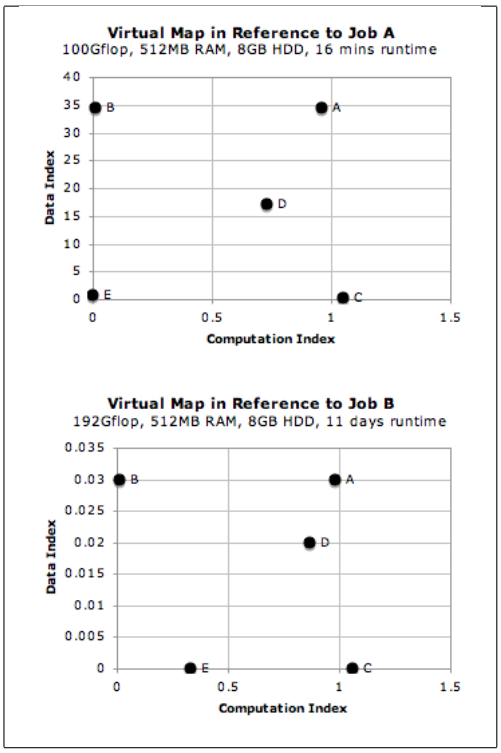


Figure 5: A Virtual Map is created for each job to determine allocation

the least amount. An illustration of the virtual map is shown in figure (5). The euclidean distance from the origin therefore denotes the best possible resources that matches the resource requirements of a job for an instance in time.

In figure (5), the computation and data index is computed by equation (1) and (2) for each job in the queue. As job requirements differs for each job, the Virtual Map is essentially different for each job submitted. This has to be computed at each job submission cycle.

3.2 Formulation for Failure Prediction

The formulation of the failure prediction model is based on the observation of pro-active and passive failure handling techniques stated below. This is then followed by the mathematical modeling of failure which will bring us to the ability to approximate or predict the failure events of a resource. We cover in

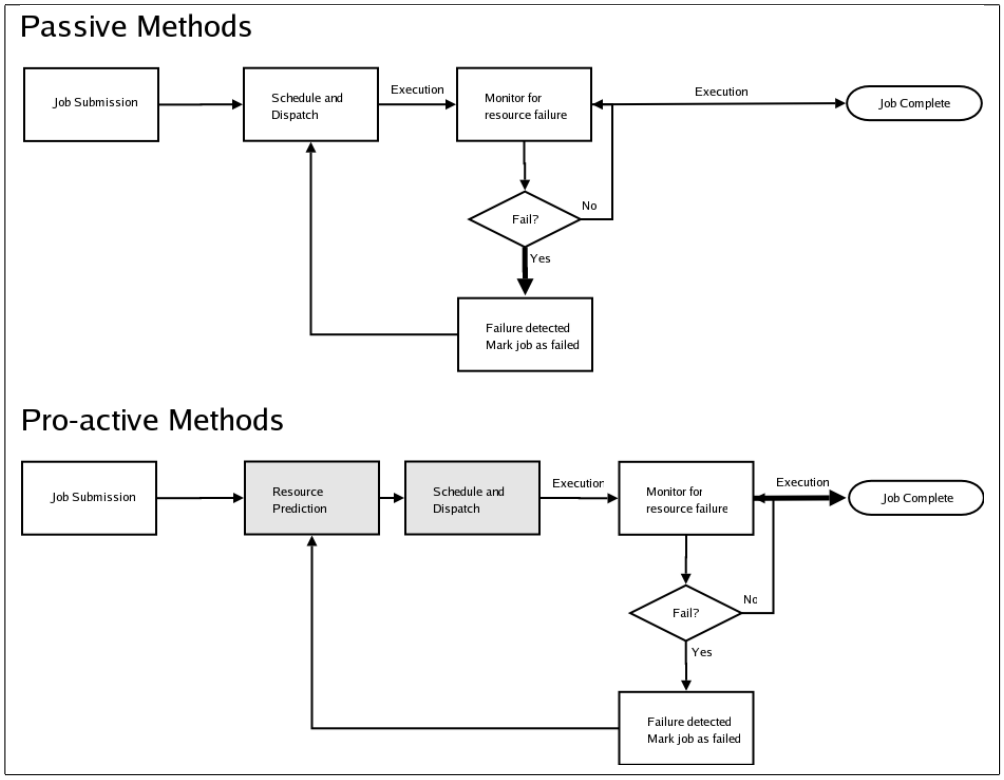


Figure 6: Passive and Pro-active mechanisms used to handle failure

detail the analysis and formulation in this section.

3.2.1 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 [41, 39] 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 requires 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 provides 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 (6), that both pro-active and passive methods are not substitutes to each other, but rather, they are compliments. 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.

3.2.2 Mathematical Modeling

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 be able to answer the following questions:-

1. How many nodes 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 questions will allow our strategy to be able 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 variables:-

- $MTTF$ *and* λ_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 $\lambda_F = \frac{1}{MTTF}$.

- $MTTR$ *and* λ_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 $\lambda_R = \frac{1}{MTTR}$.

- τ, τ_D *and* τ_U : τ represents a specific time instance after the time period T, while τ_D and τ_U are defined as the duration of the state times of a node either in DOWN or UP states. We note that for a node, if $\tau_D > 0$ then $\tau_U = 0$ 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 : 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 : The resource unrecoverbility is a single value representing the likely-hood of a resource recovering form 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_{UP} and Pr_{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 defined set of resource staying up over a period of time τ?

The answer to these questions will allow one to be able to estimate the capacity of the Grid in the future. It would also allow one 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 our 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 are independent of each other. While this assumption might not be true when observing the failures over an entire period of time such as T, we find that this is a reasonable assumption when only considering a very small instance in time between $\tau - 1$, τ and $\tau + 1$.

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

- Probability of a failure due to $MTTF$ after period of UP state at τ_U

$$\lambda_F \sum_{t=0}^{\tau_U} e^{-\lambda_F t} \lambda_F t \tag{3}$$

- Probability of a resource recovery due to $MTTR$ after a period of DOWN state at τ_D

$$\lambda_R \sum_{t=0}^{\tau_D} e^{-\lambda_R t} \lambda_R t \tag{4}$$

In addition to a resource changing states due to the MTT values, it has to be considered that there are other factors that could cause a change in state

which was represented by P and Q. As the probabilities of P and Q are independent from the MTT values, it is possible to obtain, for a single resource, its probability of remaining in its UP or DOWN state as:-

- Probability of a resource j remaining in its UP state at τ_{jU}

$$Pr_{jUP} = 1 - \lambda_{jF}(1 - P_j) \sum_{t=0}^{\tau_{jU}} e^{-\lambda_{jF} t} \lambda_{jF} t \tag{5}$$

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

$$Pr_{jREC} = \lambda_{jR}(1 - Q_j) \sum_{t=0}^{\tau_{jD}} e^{-\lambda_{jR} t} \lambda_{jR} t \tag{6}$$

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 the following equation.

$$Pr_{UP}\{n_{T+1}\} = 1 - \prod_{j=1}^{n} \lambda_{jF}(1 - P_j) \sum_{t=0}^{\tau_{jU}} e^{-\lambda_{jF} t} \lambda_{jF} t \tag{7}$$

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.

$$Pr_{REC}\{n_{T+1}\} = \prod_{j=1}^{n} \lambda_{jR}(1 - Q_j) \sum_{t=0}^{\tau_{jD}} e^{-\lambda_{jR} t} \lambda_{jR} t \tag{8}$$

Equations (7) and (8) provides a method whereby it is possible to estimate the number of resources available at $T + 1$. Under the assumption that the resources remains in constant state within τ period of time, it is possible to extend equations (7) and (8) to estimate the probability of Pr_{UP} and Pr_{REC} at time $T + \tau$. This is represented by equations (9) and (10) respectively. It is noted that we terminate the summation of the probability distribution at $\tau_U + \tau - 1$. This is due to the fact that the state of the resources at τ is dependent of its likely-hood of consistency at $\tau - 1$.

$$Pr_{UP}\{n_{T+\tau}\} = 1 - \prod_{j=1}^{n} \lambda_{jF}(1 - P_j) \sum_{t=0}^{\tau_{jU}+\tau-1} e^{-\lambda_{jF}t}\lambda_{jF}t \tag{9}$$

$$Pr_{REC}\{n_{T+\tau}\} = \prod_{j=1}^{n} \lambda_{jR}(1 - Q_j) \sum_{t=0}^{\tau_{jD}+\tau-1} e^{-\lambda_{jR}t}\lambda_{jR}t \tag{10}$$

From the RLC model and equations (9) and (10), it is therefore possible to estimated 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 is further extrapolated to obtain an estimation of the number of resources available at $S_{T+\tau}$ given by equation (11).

$$S_{T+\tau} = S_T Pr_{UP}\{S_{T+\tau}\} + M_T Pr_{REC}\{M_{T+\tau}\} + e_T \tag{11}$$

In Equation (11), 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 (11) 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 a job and subsequently run it at any point of time in the future.

While simulations of the prediction mechanism based on (11) has shown to be able to estimate the number of resources in a Grid Computing Environment (GCE) within the bounds of ± 2, 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 (11) provides a way to approximate the available 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 where 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 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.

3.2.3 Comparing Replication and Prediction

We theoradically 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 it would benefit the allocation strategy if it tries to do so.

- Replication

Assuming that a GCE consists of S resource. 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 requesting for exactly S resources, thereby limiting the maximum capacity by a factor of K. The maximum job load of the GCE accounting for its effective capacity is thus given by $\frac{S\alpha}{K}$. Assuming further that the expected time of each job j to complete is $E_T[j]$ and the theoradical time required for the job to

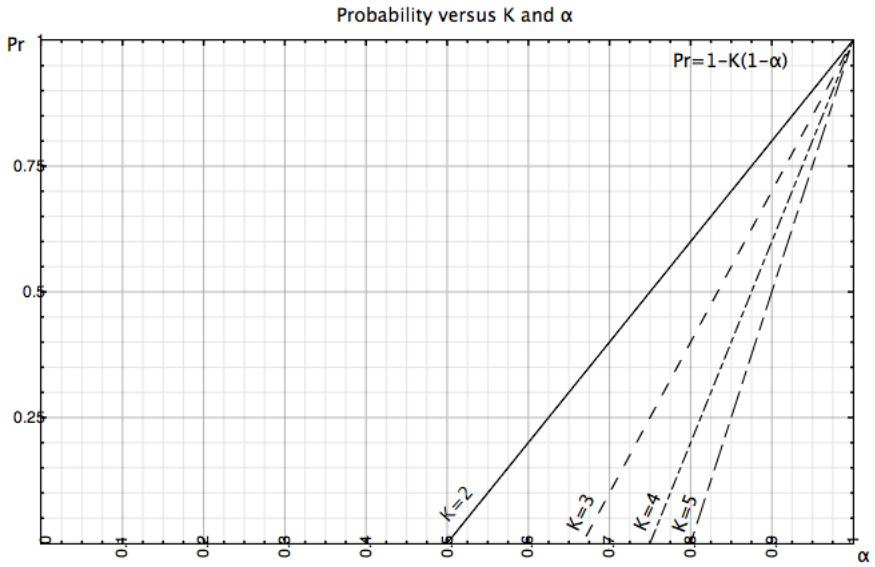


Figure 7: Probability of success versus α under varying replication factors K

complete its execution is $H_T[j]$, we can conclude that if the requested job 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 $E_T[j] \twoheadrightarrow H_T[j]$. 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 (7) that increasing the replication factor of K results in the rate of the probability of j to succeed in its execution also 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 noted that as K increased, the site availability required for any of the job replica to succeed also increases. We find this conclusion consistent with many other experiments that concludes 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 event where the requested job load $L_j \geq \frac{S\alpha}{K}$, j and its replicas will no longer expect to be executable all at the same time, but at an instance in time whereby $\frac{S\alpha}{K} = L_j$. This will however be highly subjected 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, $E_T[j] \twoheadrightarrow KH_T[j]$. 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 theoradical 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.

- 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 complete continues to be $E_T[j]$, while the theoradical 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 (8).

In the figure, it is noted that probability of success for each job j depends on the division factor k. This is analogous to the degree of parallalizm in the application, and 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

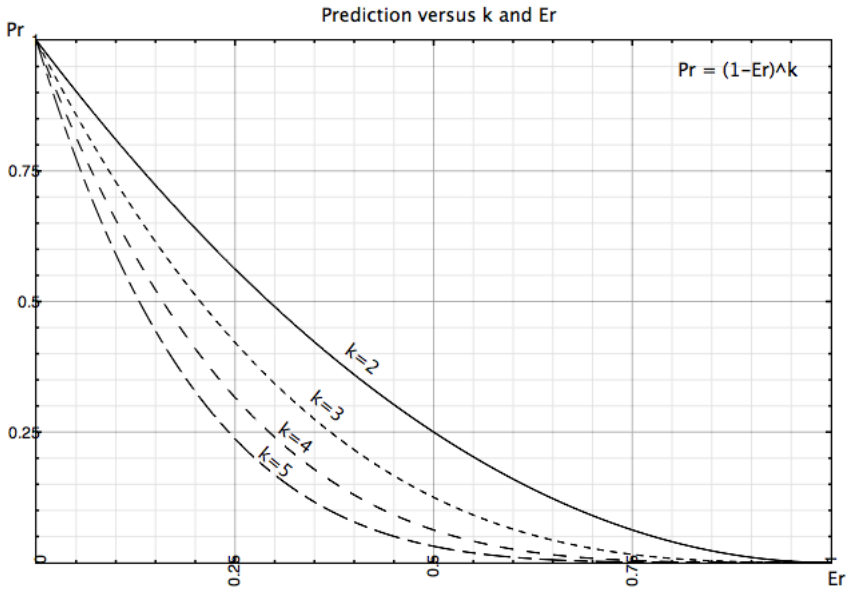


Figure 8: Probability of success Pr versus Er under varying division factors k

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 results in an exponential decline in the probability of success of j. However, 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 (9). 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 $L_j \leq S$.

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

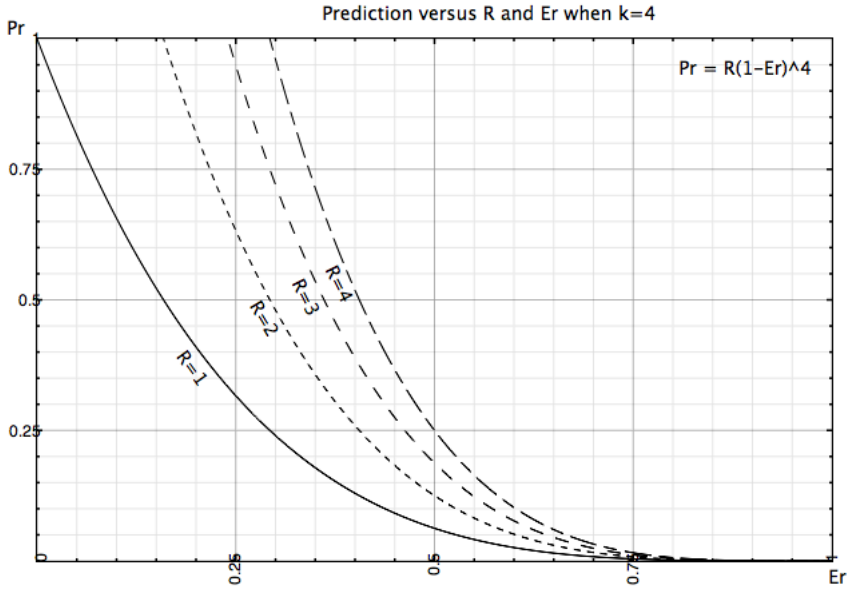


Figure 9: Probability of success Pr versus Er under varying R with division factor $k = 4$

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 acts 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. When both pro-active and passive methods are combined, premature job terminations resulting from environment failures should be greatly reduced.

3.3 Improving Resilience of Algorithms

From Section 3.2, 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 [42, 41, 40]. 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.

3.3.1 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 fails during its 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 (10) 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 [45] by the addition of a probability dimension. We discuss the modifications to the algorithms in section 3.3.2.

3.3.2 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 [27] (BF)
2. Replication Algorithm [44] (REP)
3. Multi-Resource Scheduling Algorithm [45] (MRS)

In these algorithms, BF and REP were selected as they are well known algorithms. BF serves as a baseline for comparison, allowing us to observe the ad-

vantages 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 [45] was also extended as a novel approach to allocating resources with considerations of availability in the GCE. This is 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 [45], 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 [45], 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.

4 Performance Evaluation

4.1 Simulation Design

From the MRS allocation strategy, we proceed to implement the scheduling mechanism, which will later be extended to accommodate failure events, in the GCE. There are several points observed in the implementation of the system made to support the MRS scheduling strategy.

- Each dimensional index is independent between sites and can therefore be computed locally at the participating sites within the GCE.
- A job can be submitted from any node within the GCE, a resource requirement broadcast mechanism with timeout was implemented for each job to announce itself to the sites within the GCE. This allows sites within the GCE to obtain the specific requirements for each job and evaluate its computation and data indexes accordingly. The timeout for requirements broadcast effectively truncates sites that do not reply within a certain delay. This is a simple mechanism to efficiently truncate sites that are responding too slowly to requests due to high load or congested bandwidth.
- A caching mechanism was implemented in the participating sites in order to help reduce the communications overheads. As resources are not always available to handle jobs at the time of job submission, these jobs requirements would have to be resent in a broadcast whenever a change in resource availability is detected.

A job allocation communication in event of a new job submission is as follows.

1. New job announces itself to entire GCE using a unique job ID together with its requirements.
2. Sites receive broadcast and acknowledge (ACK) with euclidean distance of its Virtual Map location to origin. Sites also cache the requirements locally.

3. Job submission location waits for a timeout and collects all ACK responses and sorts the results in ascending order

4. Full job description is dispatched to the target site

5. Target site acknowledges receipt of full job and begins processing execution request. Job submission location then issues a "cache clear" to all sites for this job ID.

In event of a job re-submission, the following process takes place.

1. Job re-announces submission request to the GCE.
2. Sites with requirements in cache ACK with euclidean measure. Sites without requirements in cache ACKs with requirements request.
3. Job submission location sends requirements to the other sites and waits for timeout.
4. Sites receives requirements and acknowledges (ACK) with euclidean distance of its Virtual Map location to origin. Sites also caches the requirements locally.
5. Job submission location waits for a timeout and compiles all ACK responses and sorts the results in ascending order
6. Full job description is dispatched to the target site
7. Target site acknowledges receipt of full job and begins processing execution request. Job submission location then issues a "cache clear" to all sites for this job ID.

From the above process, several advantages in the implementation of the MRS strategy is observed :-

- The indexes, being independent between sites, are evaluated within sites. It does not require any system monitoring mechanism to inform a master scheduler about its state. Thus reduces the complexity of the entire system during implementation.

- The main scheduler in the MRS does not contain complex algorithms and is only required to sort the resulting euclidean measure that is obtained from the GCE. Only job tracking functionalities are required at the various locations where job submission is permitted. This allows the strategy to scale better when more resources are added into the GCE to be considered. It also allows more inter-resource relations to be defined as separate dimensions without computational penalties as in a central scheduling strategy.

- Multiple MRS schedulers can co-operate in a large scale GCE. This is because the ability of resource provision is computed at the sites itself. Therefore, each site its willingness to accept a job. As multiple jobs arrive in a site, the Euclidean measure is computed sequentially and resources pre-emptively deducted. These resources will be re-included in the site as a "cache clear" is received for the intended job ID. This ensures that resources are correctly reported at every ACK to the job submission locations. This also provides a starting point for implementing a scalable distributed scheduling mechanism to support a large scale GCE.

- Independent resource policies can be implemented at every site as the Euclidean measure is calculated within the site. This allows the site administrators to be able to easily define the amount of shared resources available to the GCE without consulting a GCE administrator. Essentially, this reduces the involvement of the administrator in defining "rules" in resource allocation. Again, this reduces the implementation complexity of the MRS system.

- The broadcast and ACK mechanism used in MRS provides a way to identify sites that are disconnected from the site. The time-out function also allows the strategy to discard sites whose resources are possibly more scarce. This helps MRS in identifying sites that it can continue to schedule to even as sites leave and join the Grid environment.

The system and strategy for MRS can be described as a class of Job Sharing strategies operating within a Multi-Site Computing model [26]. However, when MRS is compared to the models described in [26], clear stages in resource selection and scheduling algorithm does not exist. The computation of the indices combines the selection and scheduling stages and thus reduces the fragmentation of resource considerations during resource allocation and scheduling.

In a strategy wherein resource matching is followed by allocation through a scheduling algorithm, where there are n computing sites in the GCE and m

resources to consider, the time complexity of the resource selection stage would be $O(n^m)$. This results in undesirable slow-downs when there are either a huge number of sites, or when there is a large number of resources to consider. The total time is therefore the sum of time-to-allocate and the time-to-schedule.

In MRS, the broadcast of requirements is of time complexity $O(n)$ as each site will only need to receive the resource requirements of a job once. However, due to broadcast, network latencies will be involved, which can possibly lead to slow-downs in MRS. This can be easily prevented by "dropping" sites that do not acknowledge the broadcast in a fixed amount of time. We, thus set an upper limit of the Time-To-Live (TTL) for each broadcast depending on the network environment MRS is operating in. The worst-case overall time taken for MRS to schedule can thus be written as $2n.TTL + max(CT_n)$, where $max(CT_n)$ is the maximum time taken for index computation for a single site. The time-complexity therefore remains linear with increase in sites as well as resources when using MRS.

We also investigated the computational complexity of MRS compared to other Job Sharing strategies in a Multi-Site Computing Model. When a strategy separates the resource selection and the scheduling phases, two main components contribute to the computational complexity of the strategy for each job. First, the sorting and filtering methodology used in the resource selection phase, and secondly, the scheduling complexity incurred in the algorithm used. In MRS, the creation of the Virtual Map for each job is essentially a sort of the (x, y) indexes provided by the sites participating in the MRS. This is simplified further when we use the Euclidean distance as a measure of match. The computational complexity is therefore only dependent on the sorting algorithm. This is because scheduling in MRS is a one step process. It is also noted that the computation complexity of the indices provided by the participating sites is linear to the number of resources and the number of dimensions we wish to consider in the Virtual Map. The increase in the number of sites or resources therefore has no effect on the overall allocation strategy provided in MRS, and thus limits the computation complexity to that of the sorting algorithm used in the system. This is unlike other strategies which can still incur computation complexities in the other stages of allocation. In our implementation, the sorting strategy used is a stable merge-sort where the complexity is $O(nlogn)$.

It should be noted that in MRS, resource considerations are not limited to dependencies. Additional requirements or dependencies can be easily added by extending the number of dimensions to be considered within MRS. This does not severely impact the complexity of MRS in both time and computation complexity when compared to other methods.

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

The broadcast of resource requirements in the GCE is done "all to all" due to the nature of Job Sharing. This is potentially wasteful when jobs have to be rescheduled due to the lack of resources or are delayed for some reason. We reduce the impact of broadcasting by allowing sites in the GCE to cache all requirements of unscheduled jobs upon reception of a broadcast. Subsequent notifications to try to schedule the same jobs will therefore incur much less overheads in communication. A cancellation broadcast was also introduced to notify all participating sites in the GCE to remove an unscheduled job from its cache. This keeps the entire GCE in sync of the jobs that are remaining to be scheduled.

In our experiments, we use a workload model based on [?] to generate a synthetic workloads consisting of both massively parallel and embarrassingly parallel jobs. We further include the resource model as discussed in section 2.2 into the described environment above such that at any point of time, resources will be able to "fail" based on a probability following a normal distribution based on the $MTTF$ and subsequently recover in the same manner based on the $MTTR$ values.

We investigate several operating environments in order to ascertain the different performance that will be exhibited under various failure 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.

We refer to Dedicated Grids as those that are pre-planned and negotiated. These Grids are typically made up of servers, clustered computers and supercomputers. 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 between institutes. Examples of

such Grids includes the UK e-Science Grid[5], the Asia-Pacific Grid[6] as well as the NC BioGrid[7]. We assume the availability of such grids to be 90%, barring certain maintenance down times.

Desktop Grids operates in an environment that are 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. Such 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 includes Seti@Home[8], Korea@Home[9] and Folding@Home[10]. 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. We are unable to simulate levels of availability lower than 30% due to the large amount of simulation time required. however, we feel that 30% availability 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. This is an environment where both Dedicated and voluntary resources will coexist within a large computing resource pool, allowing jobs to make use of these computing resources where required. we take the capacity of such GCEs to be at 50%.

In our simulations, each of these type of execution environments are tagged with environmental availability values such that the overall availability values are maintained. This however does not dictate the state of resources at any point of time.

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

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.

4.2 MRS Results, Analysis and Discussions

Based on the system design in Section 4.1, the GCE is simulated in order to ascertain the performance of MRS.

We compare our MRS with the Backfilling strategy (BACKFILL) [27, 24] and a job Replication (REP) strategy [44], which is similar to that used in SETI@Home [28]. We make use of similar Job Sharing and Multi-site environment as described earlier such that the intrinsic advantages of the algorithms can be elicited and quantified.

The workload model provided by[25] was used as the workload input. The workload profile is shown in Figure 10. The metrics described in section 4 were used to quantify the performance and comparison. The results of our experiments are summarized in Table (2) and figure (11). The significance of these results are discussed below.

In the simulation model, jobs are allowed to arrive in a stream over a span of 3 days. The various job requirements are modeled by the information provided in [29, 25] and are injected into the simulation model. Data requirements are also additionally generated in order to simulate the need for data to be transported from one location to another in order for a successful computation to take place.

Average Wait Time (AWT) is the average amount of time a job waits in the queue before being executed. This starts from the point of submission to the

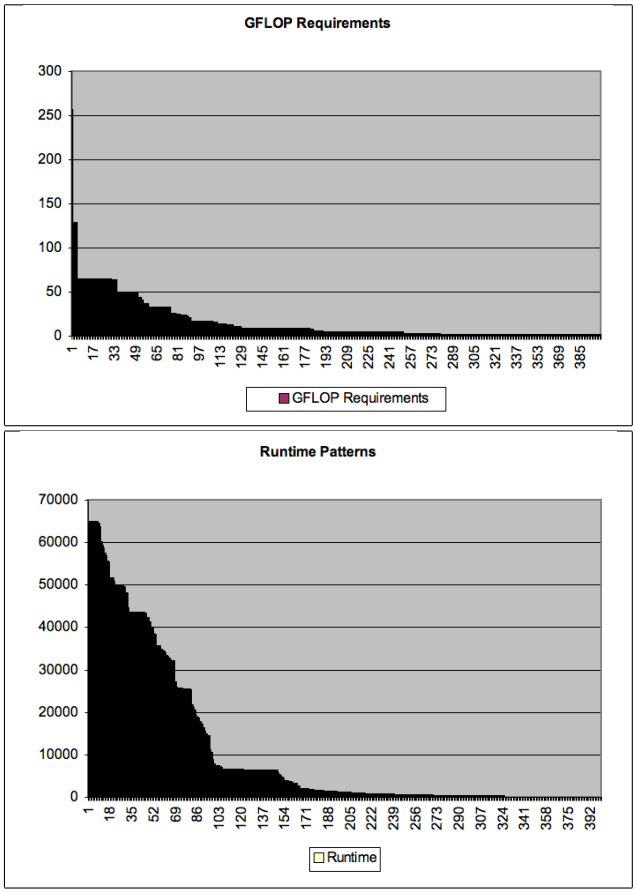


Figure 10: Workload model profile provided by [25]

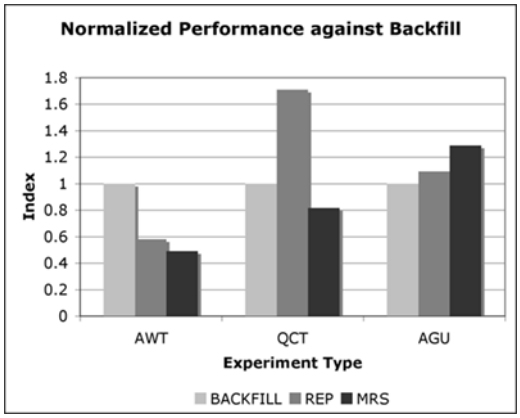


Figure 11: Normalized comparison of simulation to Backfill Algorithm

	AWT (Time units)	QCT (Time units)	AGU (%)
BACKFILL	2579.43	187302.22	57.78
REP	1500.40	320362.16	63.08
MRS	1266.71	152810.98	74.46

Tabulated Experimental Result

	AWT (%)	QCT (%)	AGU (%)
REP	41.83	(71.04)	9.16
MRS	50.89	18.41	28.86

Percentage Improvement over BACKFILL

	AWT (%)	QCT (%)	AGU (%)
BACKFILL	(71.91)	41.53	(8.40)
MRS	15.58	52.30	18.04

Percentage Improvement over REP

Table 2: Experimental results comparing BACKFILL, REP and MRS

point when the job begins its transmission to the execution node. In figure 11, we have normalized all the performance indicators to the BACKFILL algorithm in order to look at the performance differences of the experiments.

It was noted that in terms of AWT, both REP and MRS significantly outperforms BACKFILL by 40% and 50% respectively, when run in a distributed environment. This is due to the fact that the backfill algorithm does not allocate jobs in consideration of the data distribution time. The fact that jobs are streaming into the system also accounts for the inability for the algorithm to be able to obtain a good "packing" schedule where resources will be optimized. We observe that the AWT of REP is far better than BACKFILL. This is attributed to the fact that as a job gets replicated, the likelihood of being allocated to a faster resource or bandwidth increases. This is however non-optimal as it was achieved without making full use of the information available in the execution environment. This non-optimality is verified by the fact that MRS is able to achieve an even better AWT by making use of inter-resource relationships defined within its indices.

From Table 2, we can also clearly see that the utilization for BACKFILL is the lowest in all the experiments. REP and MRS exhibits increasing levels of utilization which accounts for a shorter AWT. However, it may be noted that in the replication algorithm, every job is essentially submitted twice in order to achieve better performance. This replication potentially hinders the execution of other jobs that might require more CPUs in the GCE. This can also artificially inflate the utilization of the GCE. This is clear from the fact that an increase in utilization using the REP strategy does not lead to any improvement in the QCT. It has, instead, induced a detriment to the GCE by almost 70% when compared to BACKFILL. In contrast, we can see an improvement of 18% when comparing the utilization between MRS and REP. This is also directly reflected in the overall QCT which has improved by 52%.

From our experiments, we observe that replication can lead to a degradation of performance when the entire queue is considered. This is clearly reflected in figure 11, where REP is performing 71% slower in QCT when compared to BACKFILL. It is to be noted that the time taken for a job to complete its execution is inclusive of the execution overheads and latencies that is associated with data and computation communications.

In contrast to BACKFILL and REP, our simulations have shown that MRS has been able to achieve a 50% improvement AWT, an 18% improvement over QCT and a 29% improvement in AGU. This is due to the fact that MRS makes use of comparative measures on the benefits of allocation to each node. This is inherent to the algorithm during the process of Virtual Map creation. A lower AWT is very much due to a good allocation decision of the resources when MRS is presented with a queue of jobs. This allows for more jobs to be allocated per unit time, which is reflected clearly in the 18% improvement in QCT over BACKFILL. This is achieved without the over allocation of resources as in REP, giving MRS a 52.3% improvement in QCT when compared to a REP. The matching of resources using the computation and data indexes, also results in a much higher utilization, dispatching jobs to nodes that are able to satisfy

the jobs while intelligently deciding which jobs to keep local and which jobs to dispatch.

In view of the workload model used, we observe that many of the jobs in the simulation model requires between 1-64 GFLOPs. A majority of the jobs also require run times less than half the longest running job. On comparing this workload model that we are using with those from San Diego Super-Computing Center (SDSC), Lawrence Livermore National Laboratory (LLNL) and Kungliga Tekniska hogkölan - Royal Institute of Technology (KTH), we find that our workload profile exhibited close similarities when compared to [29] and [30]. This provides further assurance that MRS is able to provide advantages in scheduling when applied to other common workload.

In general, it is observed that MRS is able to render a performance that is much suited for scheduling resources over a GCE.

4.3 Pro-active Failure Handling Results, Analysis and Discussions

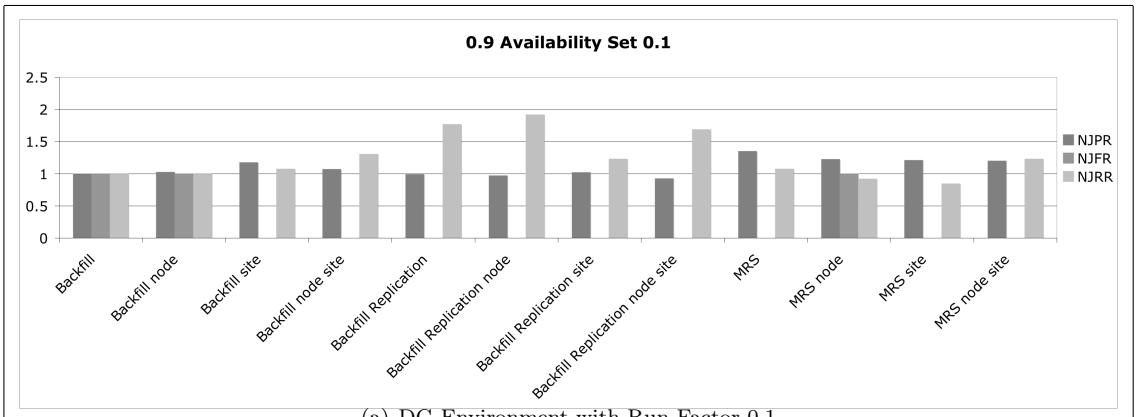
Based on the setup in table 1, we setup a simulated GCE to generate resource failures to achieve the required capacities. The results are shown in Figure 12, 14 and 13. We normalize[11] the results to that of the unmodified BF algorithm in order simplify the comparisons between the various strategies.

4.3.1 Performance of the unmodified algorithms

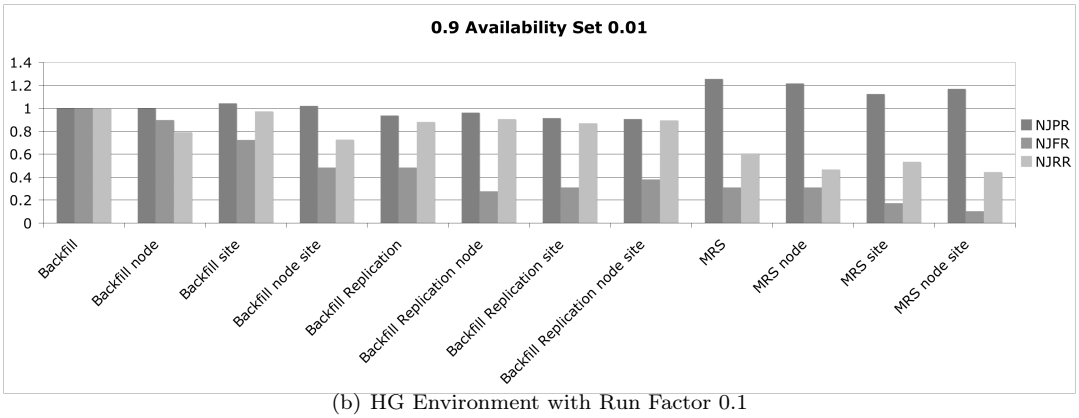
From the results, it is noted that there is always an improvement in JPR when comparing the unmodified version of MRS to that of BF within all three simulated environments. In DGs, we also note that the JPR of MRS is about 20-30% better than that of BF. This is consistent with our results presented in [45].

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

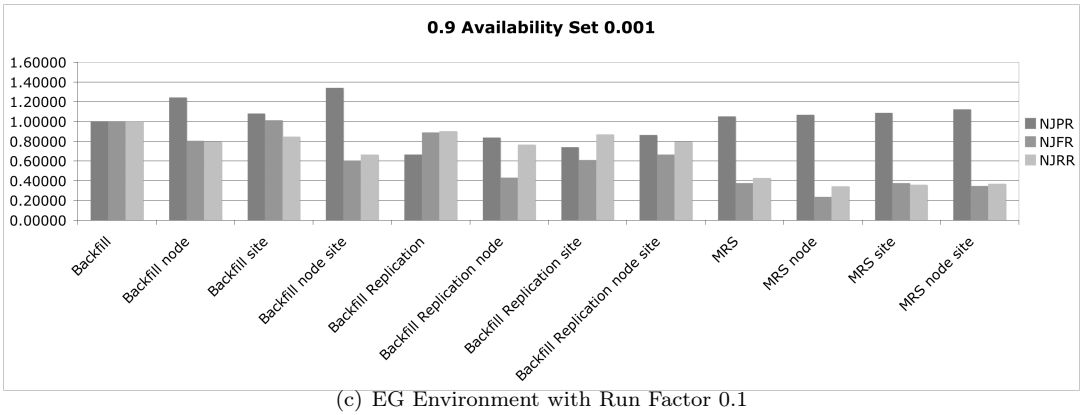
[11]Note that the suffix of "N" before JPR, JFR and JRR represents "Normalized"



(a) DG Environment with Run Factor 0.1

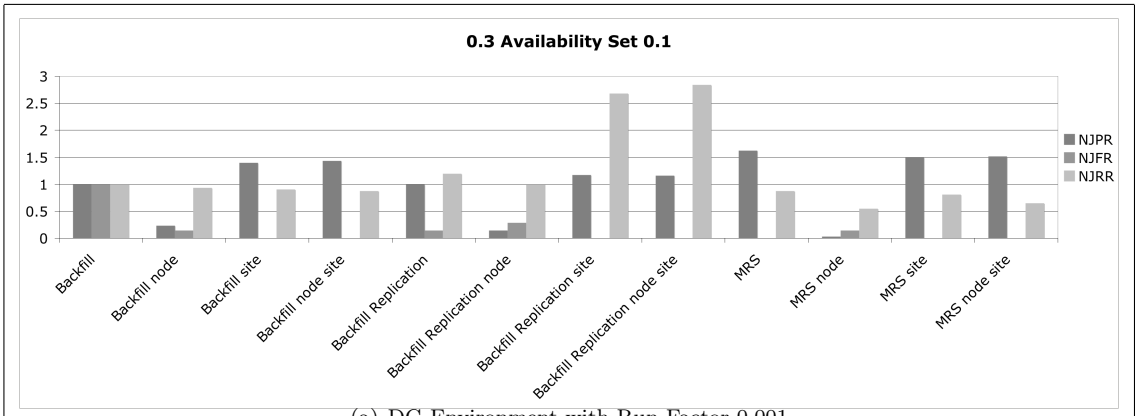


(b) HG Environment with Run Factor 0.1

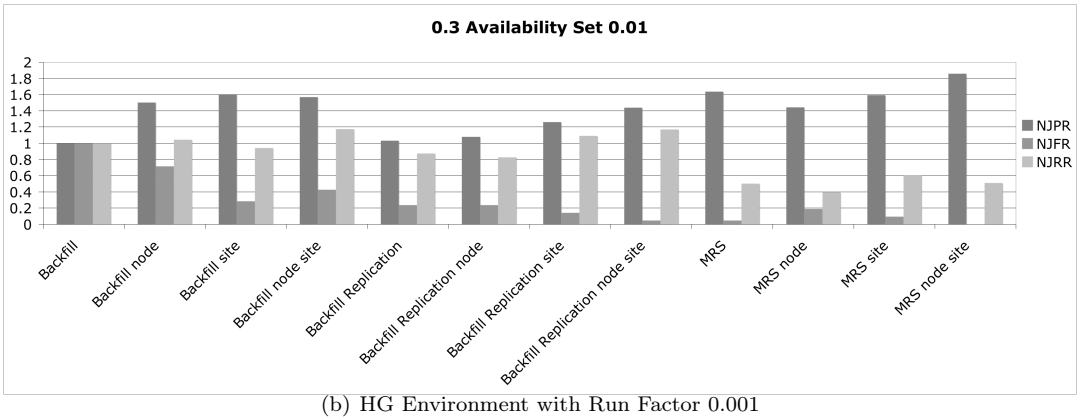


(c) EG Environment with Run Factor 0.1

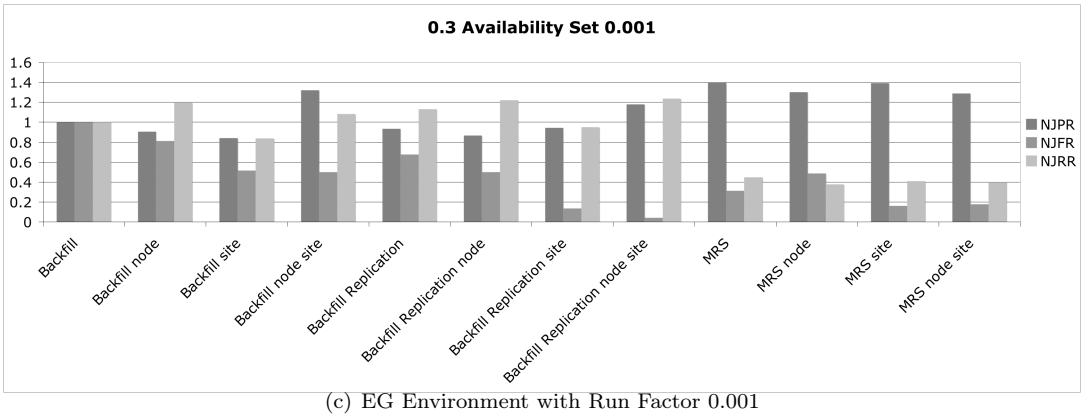
Figure 12: Simulation results for DG under different Run-Factors



(a) DG Environment with Run Factor 0.001

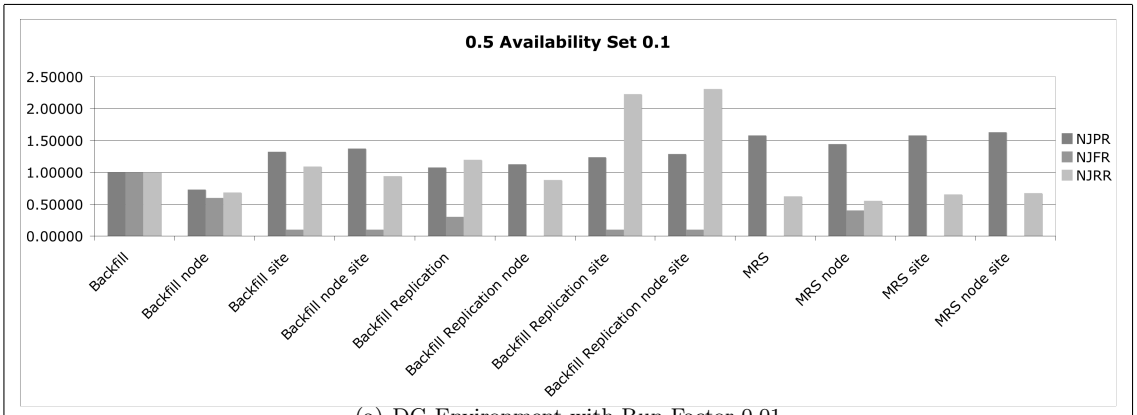


(b) HG Environment with Run Factor 0.001

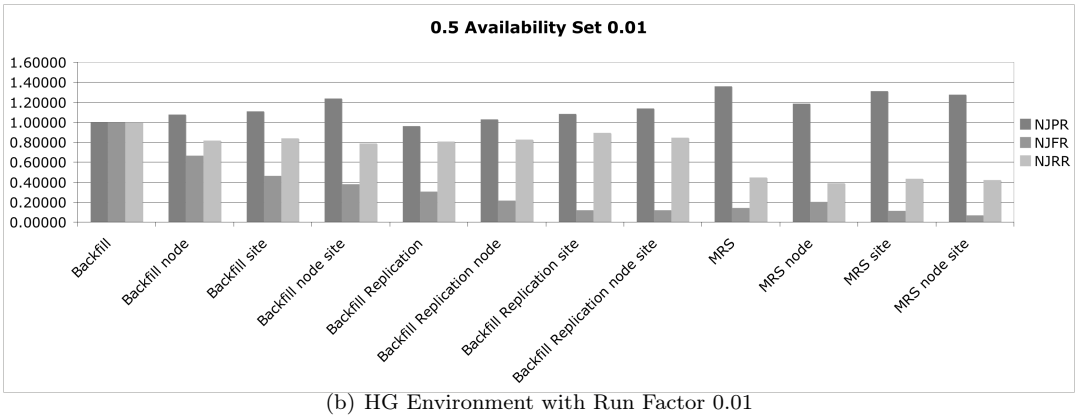


(c) EG Environment with Run Factor 0.001

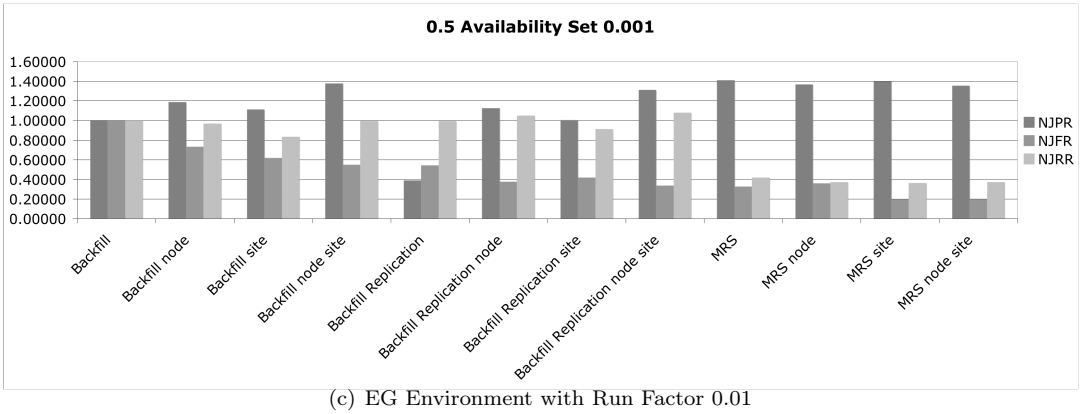
Figure 13: Simulation results for EG under different Run-Factors



(a) DG Environment with Run Factor 0.01



(b) HG Environment with Run Factor 0.01



(c) EG Environment with Run Factor 0.01

Figure 14: Simulation results for HG under different Run-Factors

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.

4.3.2 Performance of the modified algorithms in a DG environment

From the graphs shown in Figures 12, 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 shows that there is definite improvement in the assurance of job completion when pro-active strategies are introduced.

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 circumstances. 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 perceived 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. This, therefore 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 pro-active 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. Of 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.

4.3.3 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 from BF, exhibiting noticibly 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.

4.3.4 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. 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. However, in general we feel that the NSA algorithm provides the best balance in performance while minimizing job failures in all cases.

The above results offers conclusive evidence that 3D-MRS is able to exhibit effectiveness when handling failures pro-actively, while performing optimally under various operating environments, when compared to backfill and replication algorithms.

5 Conclusion

In this thesis, we have proposed a novel distributed resource scheduling algorithm capable of handling several resources to be catered among jobs that arrive at a Gird system. Our proposed algorithm, referred to as Multi-Resource Scheduling (MRS) algorithm, takes into account the different resource requirements of different tasks and shown to obtain a minimal execution schedule through efficient management of available Grid resources. We have proposed a model in which the job and resource relations are captured and are used to create an aggregated index. This allows us to introduce the concept of virtual map that can be used by the scheduler to efficiently determine a best fit of resources for jobs prior to execution. We also introduced the concept of Resource Potential to identify inter-relations between resources such as bandwidth and data. This allows us to identify sites that has least execution overheads with respect to a job.

In order to quantify the performance, we have used performance measures such as average job wait times, queue completion times, and average resource utilization factor, respectively. We considered practical workload models that are used in real-life systems to quantify the performance of MRS. Performance of MRS has been compared with conventional backfill and replication algorithms that are commonly used in a GCE. Workload models based on recent literature[25] was also used. Our experiments have also conclusively elicited several key performance features of MRS with respect to the backfill and replication algorithms, yielding performances improvements up to 50% on some performance measures.

We have also presented an extension of MRS (3D-MRS) with-in 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 or-

der 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 [45] 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.

In our experiments, we have been 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 has an additive effect, which is observed in the simulations involving the replication algorithm, where original advantages of the strategy is preserved. Our simulations has also shown that including NSA strategies into various resource allocation strategies is able to demonstrate improvements where by 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.

The results also conclusively shows 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 the way for higher assurance of a successful job execution when jobs are dispatched into a GCE. The contributions in this thesis 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.

6 Future works

Below we briefly discuss on some possible immediate extensions to the problem we have addressed in this paper. Having shown the effectiveness of MRS in a conventional scheduling environment, and the successful extension of MRS to encompass failure information, thus achieving better fault tolerance, we believe that MRS can be even further extended to include more dimensions. For instance, using the virtual map technique, it is possible that other parameters such as, Quality-of-Service [31, 32], economic considerations [33] can be included into the model by simply extending the number of dimensions of consideration. These new considerations and how it interacts with other parameters have to be studied carefully to quantify the inter and intra-resource relationship and then represented into an aggregation equation which can be used in MRS. It would be interesting to consider expanding our simulation environment to include latency information and not assume the direct relation between bandwidth and latency. Lastly, it would be more interesting to invent advanced techniques of job arrangement and fragmentation of jobs to thoroughly exploit the idling resources during the execution of jobs, especially when job queues are insufficient to fully utilize a Grid computing environment.

References

[1] Norm Snyder, "IBM Linux Clusters", http://linux.ittoolbox.com/documents/document.asp?i=2042, 2002.

[2] IBM, "Cluster Servers", http://www-1.ibm.com/servers/eserver/clusters/, 2004.

[3] Hewlett Packard, "High Performance Technical Computing", http://www.hp.com/techservers, 2004.

[4] Sun Microsystems, "High Performance Technical Computing", http://www.sun.com/solutions/hpc, 2004.

[5] I. Foster and C.Kesselman, "The Grid: Blueprint for a new Computing Infrastructure (2nd Edition)", *Morgan-Kaufman, 2004.*

[6] 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 20002 (HPDC?02), Edinburgh, Scotland, July 24-26, 359-368, 2002.*

[7] L. Zhang, "Scheduling algorithm for Real-Time Applications in Grid Environment", *In the Proceedings on IEEE International Conference on Systems, Man and Cybernetics, USA, Vol. 5, 2002.*

[8] K. G. Shin and Y. Chang, "Load sharing in distributed real-time systems with state change broadcasts", *IEEE Transactions on Computers, 38(8):1124–1142, August 1989.*

[9] W. Leinberger, G. Karypis, and V. Kumar, "Job Scheduling in the presence of Multiple Resource Requirements", *Proceedings of the IEEE/ACM SC99 Conference, Portland , Oregon, USA, Nov 13-18, pp. 47-48, 1999.*

[10] E. Santos-Neto, W. Cirne, F. Brasileiro, and A. Lima, "Exploiting Replication and Data Reuse to Efficiently Schedule Data-intensive Applications

on Grids", *Proceedings of the 10th Workshop on Job Scheduling Strategies for Parallel Processing, June 2004.*

[11] S. Venugopal, R. Buyya, and L.Winton, "A Grid Service Broker for Scheduling Distributed Data-Oriented Applications on Global Grids", *Technical Report, CoRR cs.DC/0405023: 2004. This can be located at: http://www.gridbus.com*

[12] R. Wolski and G. Obertelli, "Network Weather Service", http://nws.cs.ucsb.edu, 2003.

[13] [13] K. Ranganathan and I.Foster, "Decoupling Computation and Data Scheduling in distributed Data-Intensive Applications", *In the Proceedings of 11th IEEE International Symposium on High Performance Distributed Computing HPDC-11 (HPDC'02), Edinburgh, Scotland, July 24-26, 352-358, 2002.*

[14] N. Karonis, B. Toonen, and I. Foster, "MPICH-G2: A Grid-Enabled Implementation of the Message Passing Interface", *Journal of Parallel and Distributed Computing (JPDC), Vol. 63, No. 5, 551-563, May 2003.*

[15] A. Rajasekar, M. Wan, R. Moore, W. Schroeder, G. Kremenek, A. Jagatheesan, C. Cowart, B. Zhu, S-Yen Chen, and R. Olschanowsky, "Storage Resource Broker - Managing Distributed Data in a Grid", *Computer Society of India Journal, Special Issue on SAN, Vol. 33, No. 4, 42-54, Oct 2003.*

[16] Parallel Workload Archive: Models, http://www.cs.huji.ac.il/labs/parallel/workload/models.html

[17] K.-L.Park, H.-J. Lee, O.-Y. Kwon, S.-Y. Park, H.-W. Park and S.-D. Kim, "Design and Implementation of a dynamic communication MPI library for the grid", *International Journal of Computers and Applications, ACTA Press, Vol 26, No. 3, pages 165-171, 2004.*

[18] F. Azzedin and M. Mahewaran, "Integrating Trust into Grid Resource Management Systems", *Proc. ICPP 2002*

[19] H. Casanova, A.legrand, D. Zagorodnov, "Heuristics for Scheduling Parameter Sweep Applications in Grid Environments", *9th Heterogeneous Computing workshop 2000*

[20] K. Taura and A. Chien, , "A Heuristic Algorithm for Mapping Communicating Tasks on Heterogeneous Resources", *9th Heterogeneous Computing workshop 2000*

[21] K. N. Vijay, L. Chuang, L. Yang and J. Wagner, "On-line Resource Matching for Heteroeneous Grid Environments", *Cluster and Computing Grid, Cardiff, United Kingdom, 2005*

[22] Y. Li, and M. Mascagni, "Improving Performance via Computational Replication on a Large-Scale Compuational Grid", *IEEE/ACM CCGRID2003, Tokyo, 2003.*

[23] Open Grid Service Architecture Data Access and Integration, http://www.ogsadai.org.uk/

[24] Ahuva W. Mu'alem and Dror G. Feitelson, "Utilization, Predictability, Workloads, and User Runtime Estimates in Scheduling the IBM SP2 with Backfilling", *IEEE Transactions on Parallel & Distributed Systems, 12(6), pp. 529-543, June 2001.*

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

[26] C. Ernemann, V. Hamscher, U. Schwiegelshohn, R. Yahyapour, "On Advantages of Grid Computing for Parallel Job Scheduling," *Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid*, 2002.

[27] 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.

[28] E. Korpela, D. Werthimer, D. Anderson, J. Cobb, and M. Lebofsky, "SETI@home-Massively distributed computing for SETI," *Computing in Science and Engineering*, v3n1, 81, 2001.

[29] U. Lublin and D. G. Feitelson, "The Workload on Parallel Supercomputers: Modeling the Characteristics of Rigid Jobs." *Technical Report 2001-12, School of Computer Science and Engineering, The Hebrew University of Jerusalem*, Oct 2001.

[30] J. Jann, P. Pattnaik, H. Franke, F. Wang, J. Skovira, and J. Riodan, "Modeling of Workload in MPPs" Job Scheduling Strategies for Parallel Processing, D. G. Feitelson and L. Rudolph (Eds.), Springer-Verlag, Lecture Notes in Computer Science, vol. 1291, pp. 95-116, 1997.

[31] X.S. He, X.H. Sun, and G. von Laszewski, "QoS guided min-min heuristic for Grid Task Scheduling", Journal of Computer Science and Technology, Editorial Universitaria de Buenos Aires, Argentina, Vol 18, Issue 4, 442-451, July 2003.

[32] A. Takefusa, H. Casanova, S. Matsuoka, and F. Berman. "A study of deadline scheduling for client-server systems on the computational grid". In Proceedings of 10th IEEE International Symposium on High Performance Distributed Computing(HPDC-10), pages 406-415, 2001.

[33] R. Buyya, M. Murshed, D. Abramson, and S. Venugopal, "Scheduling Parameter Sweep Applications on Global Grids: A Deadline and Budget Constrained Cost-Time Optimisation Algorithm", International Journal of Software: Practice and Experience, Wiley Press, USA. This document can also be found at: http://www.gridbus.org/~raj/cv.html#papersj

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

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

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

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

[38] 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.

[39] 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.

[40] 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

[41] 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.

[42] 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.

[43] Benjamin Khoo, 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.

[44] 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.

[45] 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