Studies in Computational Intelligence 146

Fatos Xhafa
Ajith Abraham (Eds.)

Metaheuristics for Scheduling in Distributed Computing Environments

Springer

1

An Adaptive Co-ordinate based Scheduling Mechanism for Grid Resource Management with Resource Availabilities for Grid Computing Environments

Benjamin Khoo B. T, Bharadwaj Veeravalli* **

Computer Networks and Distributed Systems (CNDS) Laboratory, Department of Electrical and Computer Engineering, National University of Singapore, 4 Engineering Drive 3, Singapore 117576

Summary. In this chapter, we propose a novel resource-scheduling strategy capable of handling multiple resource requirements for jobs that arrive in a Grid Computing Environment. In our proposed algorithm, referred to as Multi-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 introduce the concept of a 2-dimensional virtual map and resource potential using a co-ordinate based system. To further develop this concept, a third dimension was added to include resource availabilities in the Grid environment. Based on the proposed model, rigorous simulation experiments shows that the strategy provides excellent allocation schedules as well as superior avoidance of job failures by at least 55%. The aggregated considerations is shown to render high-performance in the Grid Computing Environment. The strategy is also capable of scaling to address additional requirements and considerations without sacrificing performance. Our experimental results clearly show that MRS outperforms other strategies and we highlight the impact and importance of our strategy.

Key words: Grid Computing, Multiple Resource Management, Predictive Failure Handling, 3-Dimensional Resource Allocation, Virtual Maps

* Corresponding Author: Bharadwaj V, Dept of ECE, 4 Engineering Drive 3, NUS, Singapore 117576, Ph: +65-68745158, Fax: +65-67791103

** Parts of this work addressed certain objectives of a funded project in Grid Computing under the Grant 052 015 0024/R- 263-000-350-592 from National Grid Office via A*Star, Singapore. Bharadwaj Veeravalli would like to thank the funding agency in supporting this research.

1.1 Introduction

As computing systems has evolved from single monolithic systems to the networked multi-core systems of today, distributed computing has began to make an impact beyond research and has grown into the commercial space. This growth has accelerated the usage of distributed computing systems either as dedicated clusters, or as workstations that share its computing resources with an interactive user. The usage of Grids [1] adds further complexity into these two classes of systems by considering geographical separation, multi-organizational identity management as well as resource co-ordination.

Vendors such as IBM, HP and Sun Microsystems have all introduced hardware solutions that aims to effectively lower the cost-per-gigaflop of processing while maintaining high performance using locally distributed systems. Additionally, solution vendors such as Sybase, DataSynpase and United Devices have also further pushed the envelope of distributed computing beyond research and academia, moving traditionally local resources such as memory, disk and CPUs to a wide area distributed computing platform sharing these very same resources for commercial workloads. Consequently, what had used to be optimal in performance for a local environment has suddenly become a serious problem when high latency networks, uneven resource distributions, and low node reliability guarantees, are added into the system.

Allocation strategies for such distributed environments are also affected as more resources and requirements have to be addressed in a Grid system. Coupled with un-reliable information availability and possibilities of failures, such environments has also resulted in failure of traditional scheduling algorithms where changes adversely affects the robustness of scheduling algorithms that are available for Grids.

In this chapter, we propose a novel scheme that considers various resource requirements of jobs while taking into consideration the distributed computation environment where the job resides in. The technique we propose shall then devise an allocation, which can be used to provide what it believes as the most efficient job execution sequence to handle the jobs. Below we summarize our contributions in this chapter.

1.1.1 Our Contributions

We propose a novel methodology referred to as Multiple-Resource-Scheduling (MRS) strategy that would enable jobs with multiple resource requirements to be run effectively in 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, derived from the n-dimensional resource aggregation method, 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 virtual map that allows us to identify the best allocation of resources for the job. 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. The number of dimensions considered depends on the number of job related attributes we wish to schedule for.

We evaluate the performance of our proposed strategy firstly 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. We then further extend MRS into a third dimension to accommodate availability considerations in the Grid environment. Our study shows that MRS out performs most of the commonly available schemes in place for a GCE.

1.1.2 Organization of Chapter

The organization of this chapter is as follows. In Section 2, we describe the Grid Computing Environment and in Section 3 we introduce our MRS strategy and algorithm. Section 4 evaluates the performance of both MRS in 2 and 3 dimensions. Section 5 concludes the chapter.

1.2 Grid Environment Model

In this section, we define the GCE in which the MRS strategy was designed. 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 computing power, RAM, Hard disk space and bandwidth. Communication to individual nodes in the cluster will be done through a Local Resource Manager (LRM). 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. Negligible propagation delay of information is assumed in the GCE. We also 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 networks which are possibly asymmetrical in bandwidth.

4. All resources have prior agreement to participate on the Grid. From this, we safely assume an environment whereby all resources shared by sites are accessible by every other participating node in the Grid if required to do so.
5. In our simulations, 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 [13] and the Grid workload model available in [14]. 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[3] or PVM[4].
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.
4. A job submitted can be scheduled for execution anywhere within the GCE as applications are assumed to be available in all sites.
5. Jobs resource requirements are divisible into any size prior to execution.
6. Every job also has a data requirement where-by the main data source and size is stated.
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.

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

A physical illustration of the resource environment that we consider is shown in figure (1.1), and the resource view of how the Grid Meta-Scheduler will access all resources through the LRM is shown in the figure (1.2).

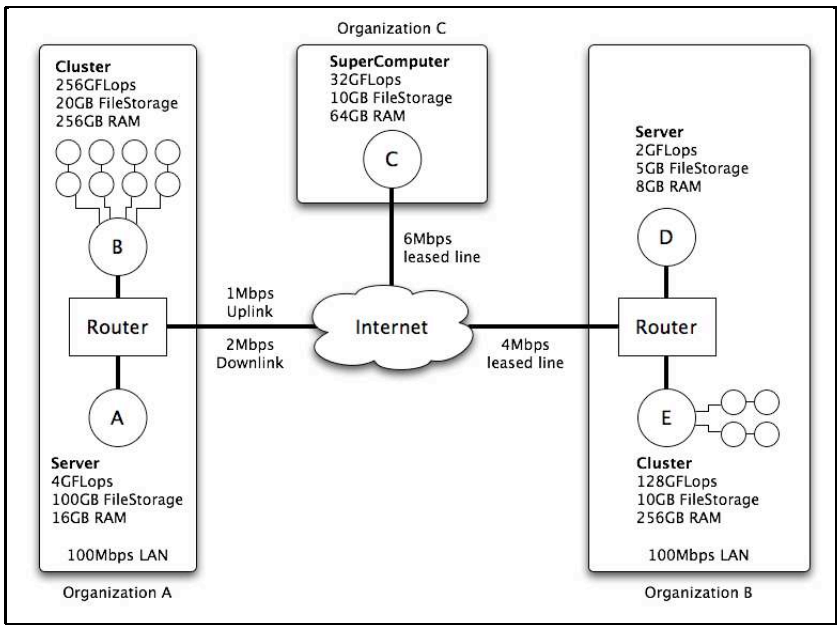

Fig. 1.1. Illustration of a physical network layout of a GCE.

In such an environment, we consider *Failure* to be the breakdown of network communication between computing resources, thereby leading to a loss in status updates in the progress of an executing job. This failure can be due to a variety of reasons such as hardware or software failures. 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 take the view of the resource by an external agent in order to classify if a resource has entered a state of a general *failure* or has *recovered* from its unavailable failed state. Thus, under these assumptions, we are able to 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

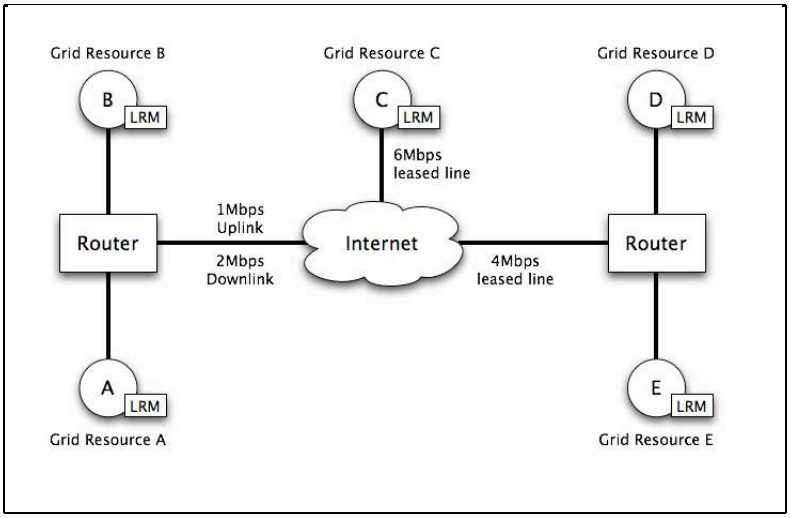

Fig. 1.2. Resource view of physical environment with access considerations

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 on-line once it is capable and becomes available to the GCE (return to first stage)

From the above, it was observed that in Stages (2) and (4), the resource undergoes a period of uncertainty. This uncertainty stems from the fact that the resource probably might not fail or recover for a certain period of time. Based on these stages the model presented in [15] was constructed. The Resource Life Cycle (RLC) Model shown in Figure 1.3 identifies the stages where by Grid resources under-go cycles of failures and recovery, and also accounts for the probabilities of *each* resource being able to recover or fail in the next epoch of time. Thus using this model, we are able to describe a general form of resource failure that would cause a loss of job control or connectivity to the said resource. This in turn affects the capacity of the GCE.

1.3 Scheduling Strategy

From the Grid Environment Model, we note that the system environment of the Grid consists of heterogeneous nodes. This results in an environment whereby a wide range of resources are available. These resources may or may not be well connected to each other depending on network connectivity and thus require proper allocation and grouping before jobs can be executed efficiently.

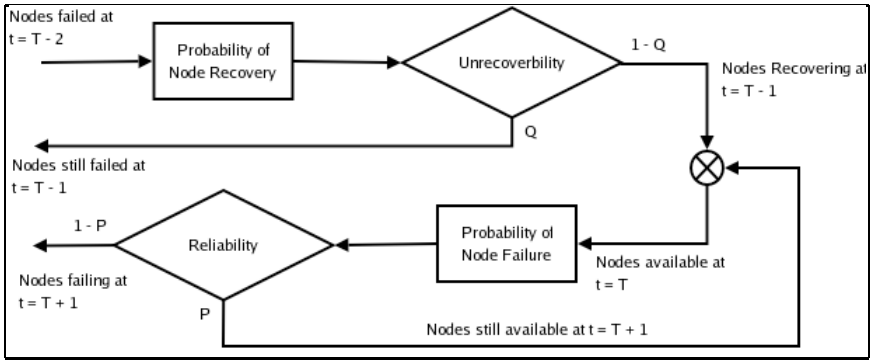

Fig. 1.3. Resource Life Cycle Model for resources in the GCE

MRS addresses the various job requirements and resource capabilities by dividing decision factors into separate dimensions. By using some performance metrics, it then decides which resources the jobs would be best dispatched to. It combines several inter-dependent factors within each selected dimension and simplifies it into a single index which is then used to decide how a job is to be sent or distributed to the resources. MRS also treats each submitted job as an independent entity and does not address work-flow requirements of any application. We feel that this is done without any loss of generality as work-flow requirements should be addressed at an orchestration layer independent of the scheduling middle-ware. With MRS always allocating every job to sites that best provides its resources, it ensures that the job execution environment will remain optimal for both serial as well as parallel jobs.

The jobs request and site representations of CPU resources is done in terms of GFLOPs as an indication of performance. Future changes in unit representations will not affect the strategy as the aggregation algorithm will result in dimensionless indexes as long as the request and site resource representation units are the same. This applies to all other resources shared within MRS.

In this chapter, we first consider 2 dimensions (2D) within MRS and then extend it to a third dimension (3D). The two basic dimensions (1) Computation, and (2) Data are used in our design. 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. It is to be noted that these two components are highly related to each other in the scheduling process. Each of them on its own, would be unable to provide optimality in resource allocation. Aggregation of the various available resources are then combined into two major indices based on these two basic dimensions. We refer to these indices as the Computational and Data Index respectively.

The third dimension of capacity is subsequently added to MRS as another component that affects the optimality of the allocation strategy. While in an

ideal GCE, this dimension can be ignored, the inclusion of this dimension would allow better representation of how an allocation strategy can adapt dynamically to changing GCEs. This makes the allocation strategy much more versatile compared to traditional algorithms.

The Computational and Data Index allows us to create a 2D plot which describes the virtual topology, which we call a *Virtual Map*, of the GCE. The distance to the origin will describe the matching proximity of the resource to the job. Similarly, the extension of the third dimension to include the availability of resources extends the *Virtual Map* into a *Virtual Space*. The most suited resource providers will continue to be the sites whereby it is located nearest to the origin. The sections below will demonstrate how we construct the two basic dimensions and the process of aggregation that leads to the final aggregated Indexes used in the Virtual Map. A description of the simplicity of extending this to a Virtual Space is then described.

1.3.1 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 allocatable 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. We denote m as the total number of sites in a GCE. 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. 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 [17] is employed. We illustrate this "flattening" process in figure (1.4). The values C, M, F and 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 allocatable computational resources within a site i as a set $S_c = \{R_i, t\}$ where $S_c \subseteq S$. R_i is further represented by 4-tuple of $f_i(< C, M, F, P_i >, t)$ denoting the four resources considered in our allocation strategy.

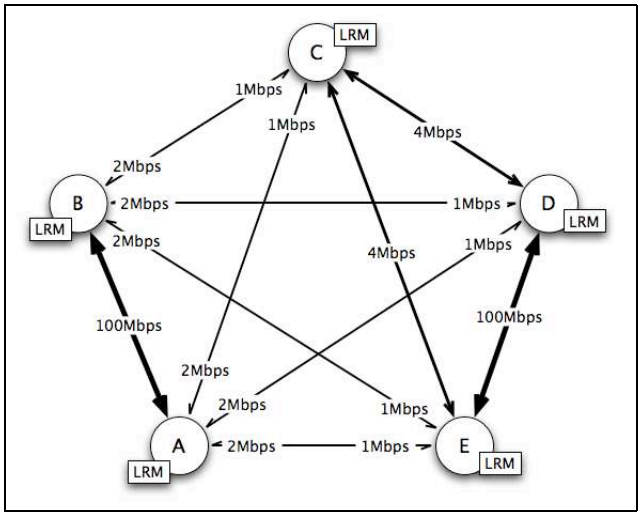

Fig. 1.4. Flattened network view of resources for computation of Potential

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[5].

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 $g_j(< C, M, F, P_{src} >, t)$.

[5] In our simulations, we have assumed that applications are pre-staged at the sites.

1.3.2 Computational Index through Aggregation

Evaluation of various resource requirements of sites and jobs allows us to aggregate and encode inter-resource relationships in order to arrive at a single index which 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, $R_{ij}\{C\} = 1 - \frac{f_i\{C\}}{g_j\{C\}}$. Only the positive values of $R_{ij}\{C\}$ are considered, 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 apply the same ratio of provision to all resource and requirements 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 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 dimensionless computation index (x_{ij}) 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. An increasing value of $K > 0$ signifies an increasing importance of a specific resource requirement relative to the other resources. This steers the strategy away from the default allocation to one that is weighted towards the more important resource.

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 K constants such that $K = 1$. This provides equal importance to all components making up the computational index. Detailed derivation and formulations can be found in [16]. Essentially, such a strategy does not restrict itself to specific units of measure for C, M or F. Potentially, any arbitrary unit is suitable for this approach, as long as the entire GCE is in agreement.

$$x_{ij} = \sqrt{(K_C R_{ij}\{C\})^2 + (K_M R_{ij}\{M\})^2 + (K_F R_{ij}\{F\})^2 + (K_P R_{ij}\{P\})^2} \tag{1.1}$$

1.3.3 Data Dimension and indexing through resource inter-relation

In the data dimension, we wish to determine the best resource that would execute a job considering its I/O requirements. 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 run-time 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 (non-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 time dependent resource which is based on the instantaneous bandwidth availability at a resource. We annotate bandwidth B between two sites i and j as $B_{ij} = min\{B_{ij}^{download}, B_{ji}^{upload}\}$ which changes over time t as data capabilities of a resource $S_d\{R_i, t\}$. Where each item in the 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 run-time 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.

$$y_{ij} = \frac{e_j\{F\}}{d_i\{B_{ij}\}} . \frac{1}{A^{runtime}} \tag{1.2}$$

1.3.4 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. As we have observed, sites that position themselves closest to the origin are those that deviate from the resource requirements by the least amount. An illustration of the virtual map is shown in figure (1.5). 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 (1.5), the computation and data index is computed by equation (1.1) and (1.2) 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 each time a job is to be submitted or re-submitted to the GCE.

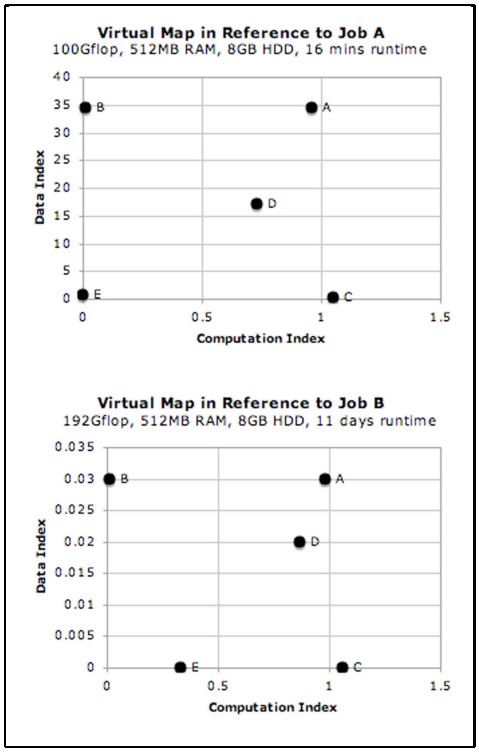

Fig. 1.5. A Virtual Map is created for each job to determine allocation

1.3.5 Availability Index

We first define the following notations used in this section, and the definitions associated with them:-

- $MTTF_j$ *and* λ_j^F : The Mean Time to Failure represents the average amount of time a resource is available to the GCE before going offline. We also term the average rate of failure to be $\lambda_j^F = \frac{1}{MTTF_j}$. Where j denotes the node index in the GCE.
- τ, τ_j^U : τ represents a specific time instance after the time period T, while τ_j^U is defined as the duration of the of the jth node in the UP state.
- P_j : Denotes the resource reliability is a single value representing the likelihood of a resource staying on-line 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.
- Pr_j^{UP}: The probability of a resource j remaining in its UP, or on-line, state.

Using the Poisson Distribution to model the event of a single change in state, with the assumption that the resources remain in constant state within τ period of time, it is possible to estimate the probability of Pr^{UP} at time $(T + \tau)$. This is captured by Equation (1.3).

$$Pr^{UP}\{n_{T+\tau}\} = 1 - \prod_{j=1}^{n} \lambda_j^F (1 - P_j) \sum_{t=0}^{\tau_j^U + \tau - 1} e^{-(\lambda_j^F t)} (\lambda_j^F t) \tag{1.3}$$

We make use of Equation (1.3) to obtain an index of Availability represented by $(1 - Pr^{UP})$. This is coupled as a third dimension in addition to the Compute and Data Index for each resource in consideration. One would notice that a resource that is more likely to stay up will have a value closer to zero than one that is likely to fail. The inclusion of this index allows one to pro-actively estimate which resources will be available during a jobs run-time such that the likelihood of job failure is reduced. This is different from other proposed failure handling strategies where-by the failed job is trapped and then restarted.

The merging process is similar to that described in section 1.3.4, where the shortest euclidean distance from the origin denotes the best possible resources that matches the resource requirements of a job for that instance in time.

1.4 Performance Evaluation

1.4.1 Computation and Data index in 2D MRS

We compare our basic 2-dimensional MRS with the Backfilling strategy (BACKFILL) [18, 19] and a job Replication (REP) strategy [20], which is similar to that used in SETI@Home [21]. The workload model provided by[14] was used as the workload input. The following metrics where used as the performance measure of the algorithm.

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,

   $$AWT = \frac{\sum_{j=0}^{J-1}(e_j - s_j)}{J} \tag{1.4}$$

   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 trackable. This allows us a quantitative measure of throughput, where the smaller the time value, the better. The queue completion time is given by,

$$QCT = e_{J-1} + E_{J-1} - s_0 \tag{1.5}$$

   where, E_{J-1} 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.

$$AGU = \frac{\sum_{t=s_0}^{QCT} U(t)}{QCT} \tag{1.6}$$

The results of our experiments is summarized in figure (1.6). The significance of these results are discussed below.

It was noted that in terms of AWT, both REP and MRS significantly out-performs BACKFILL by 40% and 50% respectively. This is due to the fact that the backfill algorithm does not allocate jobs in consideration of the data distribution time. The improved performance of REP compared to BACKFILL on AWT can be 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 the figure, 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

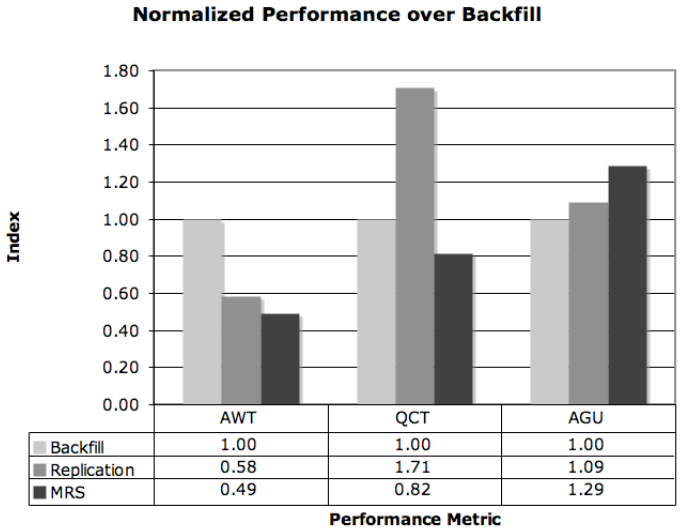

Fig. 1.6. Normalized comparison of MRS and REP simulation to Backfill Algorithm

order to achieve better performance. This replication potentially hinders the execution of other jobs that might require more CPUs in the GCE. This is clear from the fact that an increase in utilization using the REP strategy does not translate to an improvement in QCT. It has, instead, induced a detriment to the GCE by almost 70% when compared to BACKFILL. This could be attributed to in-efficient allocation of resources. In contrast, we can see that an improvement in utilization of 29% between MRS and BACKFILL is also directly reflected by a 18% improvement in QCT.

As discussed earlier, we have ascertained that replication can lead to a degradation of performance when the entire queue is considered. 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 clearly reflected in MRS's improvement in QCT over BACKFILL. This continues to be achieved when compared to REP, indicating that MRS is able to allocate resources more effectively when compared to REP. This is clearly shown when comparing the results in figure (1.6). 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.

1.4.2 Inclusion of Availability Index in 3D MRS

From the positive results in the implementation of 2D MRS in section 1.4.1, we extended our GCE to exhibit failures in resources and also included the Availability Index as the third dimension in MRS. This is to further investigate if the extension of the MRS methodology will continue to exhibit positive results even in higher dimensions. It is also to apply the strategy in a more realistic GCE environment where failures do affect how a job should be scheduled. We used the following metrics as a measure of performance.

1. Job Processing Rate (JPR):

$$JPR = \frac{NumberOfJobsSuccessfullyCompleted}{TotalQueueCompletionTime} = \frac{J_{Success}}{T_Q} \tag{1.7}$$

   A higher JPR will indicate larger number of successfully completed jobs or a lower queue completion time. A high JPR will therefore indicate that an algorithm is capable of high throughput.
2. Job Failure Rate (JFR):

$$JFR = \frac{NumberOfJobsFailedAtRuntime}{TotalQueueCompletionTime} = \frac{J_{Fail}}{T_Q} \tag{1.8}$$

   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):

$$JRR = \frac{NumbeOfJobsRejected}{TotalQueueCompletionTime} = \frac{J_{Rej}}{T_Q} \tag{1.9}$$

   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.

The GCE was simulated to include 50% of dedicated resources while the remaining are volatile resources that goes on and offline periodically based on a set of random generated normally distributed $MTTF$ value. The workload environment was also modified to address the problem where workload models

tends to generates much lesser jobs with long run-times. It is done such that the longest job run-time is 1000 times that of the average $MTTF$ in the GCE. This would induce a much larger number of failures in the volatile resources, providing a better view into the effectiveness of the algorithm. The normalized result of the simulation is shown in the Figure (1.7).

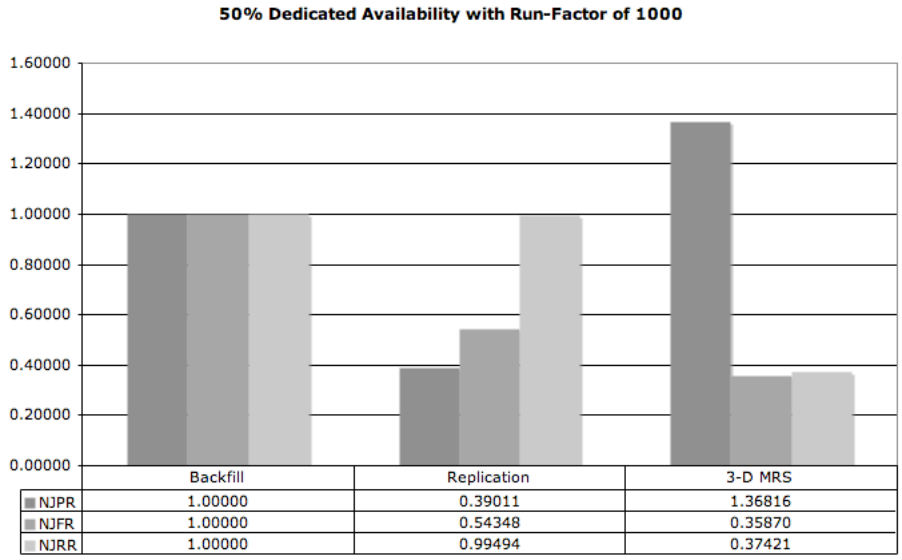

Fig. 1.7. Normalized comparison of 3-D MRS and REP simulation to Backfill Algorithm

From the results, we can clearly see an approximately 30% increase in the JPR of 3-D MRS when compared to BACKFILL and an even larger increase when compared to REP. This is likely due to the more effective job to resource matching strategy employed in MRS, allowing more jobs to complete within each period of high resource availability. The approximately 64% and 61% improvements in JFR and JRR also clearly demonstrates the effectiveness of being able to include availability information as part of the allocation strategy. It is clear from the simulations that the job failures resulting in a REP strategy is approximately half that of the replication factor (which was 2 in the simulation). It can be reasoned then that for every 2 jobs replicated, there is a 50% chance that one of them will fail due to the lack of knowledge in which half of the resources is likely to fail. One would also notice that the JRR of REP is similar to that of BACKFILL further pointing to limited improvement in resource utilization in the GCE.

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

to include availability factors of the GCE into the strategy while continuing to provide superior performance to traditional algorithms.

1.5 Conclusions

In this chapter, we have proposed a novel distributed resource scheduling algorithm capable of handling several resources to be catered among jobs that arrive at a Grid 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. A third dimension was also introduced to extend the idea of a virtual map into a virtual space. This new dimension proposes the use of availability of each resource such as to provide a more accurate resource allocation strategy.

In order to quantify the performance, we have used various measures to ascertain the performance of our strategy in both the 2D and 3D aspects. 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. 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. The strategy presented continues to exhibit performance gains even when extended to a GCE that exhibits failures. Presenting more than 60% improvements in job failure rates while improving throughput by up to 30%.

The strategy discussed in this chapter introduces a mechanism where individual resources can be compounded into dimensions. Within these dimensions, the inter-relations of resources are addressed. By placing multiple dimensions together, it creates a virtual map, which can be extended into a virtual space when more than 2 dimensions are used. The ability for this strategy to be able to extend itself through additional dimensions, provides a mechanism where other forms of specialized optimization heuristics can be used. These individual dimensions, which are optimal in itself, could then be aggregated in a strategy similar to MRS where each can be weighted and interrelated for further optimization over multiple resources. The idea of a virtual map or space also potentially provides a starting point where more complex

optimizations can be applied based on virtual "spatial" considerations in a GCE.

Some possible immediate extensions to the strategy we have proposed in this chapter could look at providing a computationally less intensive mechanism to compute the predicted availabilities in the GCE such as to provide a gauge of capacity in the GCE. It could also be possible that other parameters such as, Quality-of-Service, economic considerations as well as real-time applications can be included into the model by simply extending the number of dimensions of consideration.

Related Work

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. Very few works address failure on 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 [2], 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.

In [3], 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 Centralized, Decentralized and Hierarchical models where jobs are submitted to a meta-scheduler but are dispatched to low-level schedulers for dispatch. 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 [4] was also proposed, and its effectiveness also highlighted in [5], where it was shown that an establishment of relationships in resources can lead to better performance.

In the work presented in [6], the ability to schedule a job in accordance to multiple (K) resources is explored. This approach shows clearly the benefits

where scheduling with multiple resources is concerned. Effective resources-awareness in the scheduling algorithm provided performance gains of up to 50%. Similar resource awareness and multi-objective based optimizations where studied in [7]. In both cases, the limitations of conventional methods was also identified as there was have no mechanism for utilizing additional information known about the system and its environment. However, in [6], there was no data resources identified, while in [7], we believe that the over simplicity of resource aggregation was in-adequate in capturing resource relationships.

Of works that look into failures in the GCE, many works are primarily post-active in nature and deal with failures through Grid monitoring as mentioned in [8]. These methods mainly do so by either checkpoint-resume or terminate-restart [11, 9]. Two pro-active failure mechanisms are introduced in [10, 20] and [12]. While [10, 20] operates by replicating jobs on Grid resources, [12] 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.

The formulation of 2D MRS in [16] has allowed us to build on a effective mechanism, providing an alternative solution to the problem of resource allocation. This also highlights how the strategy can be extended to consider more complex environmental requirements, which is presented in this chapter.

References

1. I. Foster and C.Kesselman, "The Grid: Blueprint for a new Computing Infrastructure (2nd Edition)", *Morgan-Kaufman, 2004.*
2. 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.*
3. 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.*
4. K. G. Shin and Y. Chang, "Load sharing in distributed real-time systems with state change broadcasts", *IEEE Transactions on Computers, 38(8), Pages:1124–1142, August 1989.*
5. F. Azzedin and M. Mahewaran, "Integrating Trust into Grid Resource Management Systems", *Proc. ICPP, PAges: 47-52, 2002*
6. 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.*
7. K. N. Vijay, L. Chuang, L. Yang and J. Wagner, "On-line Resource Matching for Heterogeneous Grid Environments", *Cluster and Computing Grid, Cardiff, United Kingdom, Pages: 607-614, 2005*

8. 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), Pages: 18-24, 2003.*
9. 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.*
10. 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*
11. 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, Pages: 572-579, 2004.*
12. 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.*
13. Parallel Workload Archive: Models, http://www.cs.huji.ac.il/labs/parallel/workload/models.html
14. B. Song, C. Ernemann and R. Yahyapour, "User Group-based Workload analysis and Modelling," *Cluster and Computing Grid Workshop 2005, Cardiff United kingdom, Pages: 953-961, 2005*
15. Benjamin Khoo and Bharadwaj Veeravalli, "Cluster Computing and Grid 2005 Works in Progress: A Dynamic Estimation Scheme for Fault-Free Scheduling in Grid Systems," *IEEE Distributed Systems* Online*, vol. 6, no. 9, 2005.*
16. Benjamin Khoo Boon T, Bharadwaj Veeravalli, Terence Hung, Simon See, "A Multi-Dimensional Scheduling Scheme in a Grid Computing Environment", *Journal of Parallel and Distributed Computing (JPDC), Volume 67 , Issue 6 Pages: 659-673, June 2007.*
17. R. Wolski and G. Obertelli, "Network Weather Service", *http://nws.cs.ucsb.edu, 2003.*
18. 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, Pages 191-202,* 2000.
19. 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.*
20. Y. Li, and M. Mascagni, "Improving Performance via Computational Replication on a Large-Scale Compuational Grid", *IEEE/ACM CCGRID2003, Tokyo, Page 442, 2003.*
21. E. Korpela, D. Werthimer, D. Anderson, J. Cobb, and M. Lebofsky, "SETI@home-Massively distributed computing for SETI," *Computing in Science and Engineering, Volume 3 Issue 1*, Pages 78-83, 2001.