A Co-ordinate Based Resource Allocation Strategy for Grid Environments

Benjamin Khoo Boon Tat, Bharadwaj Veeravalli, Terence Hung and Simon See Chong Wee
National University of Singapore Department of Electrical and Computer Engineering,
Institute of High Performance Computing, Sun Microsystems
{g0402607, elebv}@nus.edu.sg, terence@ihpc.a-star.edu.sg, simon.see@sun.com

Abstract*— In this paper, we propose a novel resource scheduling strategy, referred to as the Multi-Resource Scheduling (MRS) algorithm, which is capable of handling several resources to be used among jobs that arrive at a Grid Computing Environment. We propose a model in which the job and resource characteristics are captured together and are used in the scheduling strategy. To do so, we introduce the concept of virtual map and resource potential. Based on the proposed model, simulations with realistic workload traces were conducted to quantify the performance. We compare our strategy with some of the commonly used algorithms, and show that MRS renders a higher performance in all cases. Our experimental results clearly show that MRS outperforms other strategies and we highlight the impact and importance of our strategy.

I. 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, the use of these resources as a powerful computation platform has increased. Recent distributed computing trends has moved towards Grid Computing [1] were geographically distributed systems are binded to work together, hoping to bring huge amounts of computing resources together to solve problems that where previously too costly to handle.

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. This leads to a lack of robust scheduling algorithms that are available for Grids.

In this paper, we propose a novel scheme referred to as Multiple Resource Scheduling (MRS) which is applicable in a Grid Computing Environment (GCE). We consider the various resource requirements of jobs and its inter-resource dependencies while taking into consideration the computation environment where the job resides in. The MRS technique shall then devise an allocation schedule which can be used to provide what it believes as the most efficient job execution sequence to handle the jobs.

A. Related Work

In [3], queue completion time is reduced by executing each job at more than one location. It then obtains the results from the location where the job first completes. This method provides redundancy in view of potential node failures, and thus reduces the average job slowdown and turn-around times as the fastest submission time is used. Better job wait times and utilization if thus expected. However, the down side of this strategy is the opportunity cost of the resources spent in the duplicated jobs. In any event where a job execution succeeds, this would have lead to wastage of resources of degree n, where n is the number of times a job is replicated for submission. A similar approached is used in [2], however, it is felt that queue lengths are unnecessarily compromised. MRS eliminates this wastage, by identifying and allocating only the resources that are optimal for execution. This limits the amount of resources used to the amount only required by the job, thus allowing more jobs to be executed in an environment at any given time.

In [4], Zhang's proposal was to maintain a buddy set at each site such that the best CPU resource will be allocated quickly w.r.t the known entities within the set. However, the proposed system only investigates computational capabilities on the Grid. While Zhang's method looks at being able to schedule CPU intensive jobs in real-time, it does not address the other dependencies such as data and bandwidth requirements of a job. MRS takes the concept proposed and extends it to include considerations of job dependencies to data, bandwidth and locality. This allows MRS to be able to handle both local and distributed jobs effectively.

In the work presented in [5], the ability to schedule a job in accordance to multiple (K) resources is explored. Simulation results in [5] clearly show performance improvements when including resources-awareness in the scheduling algorithm. Studies in [5] further highlight the limitations of current approaches which can lead to unnecessary fragmentation of job and under-utilization when resources are poorly allocated. A resource balancing method was used to try to match the multiple resources available to that of a job's requirement. The results show high system utilization and better performance. However, [5] mainly considers resources within the same locality and considers resources independently based on job requirements. This causes inter-resource dependencies to be hard to capture. MRS builds on the multi-resource concept to create a strategy where dependencies can be capture and and considered easily. This results in better inter-resource co-operation and better resource and site allocation in the process.

In [6], Ranaganathan et. al. presented that Computation Scheduling and Data Scheduling can be considered asynchronously in Data-Intensive Applications, thereby separating the resource considerations of computation and data. The

study considered External Schedulers, local Schedulers and Data schedulers. It was concluded that data movement and computation need not always be coupled together for consideration. While this might be true, in some parameter sweep applications, we argue that this is not always the case when parallel applications, using libraries such as MPI or MPICH-G2 [7], [8], are concerned. This is due to the fact that distributed memory applications typically involve heavy data transmissions during the execution of the application itself. Decoupling the scheduling process into data and computation scheduling therefore is not possible due to this high dependence. There is also an assumption that little or no data is used during the execution which might also not be true as chunks of data can be picked up as events from other applications during execution which can contribute to latencies during execution. MRS resolves this by accounting for the inter-dependence of resources prior to allocation, thereby optimizing the performance of such applications.

Other works such as [13], [14] mainly concentrate specifically on either the communication and data aspects or computational requirements such as parameter sweep applications. These techniques, although applicable to certain applications, does not address instances where applications are subjected to two or more resource restrictions. Our proposed solution overcomes these limitations by determining a way of computing the desired type of resources through prudent aggregation of selected units. Our simulation results indeed testify this aggregation.

II. Our contributions

In this paper, we shall introduce a *Multiple Resource Scheduling (MRS)* strategy that would enable jobs with multiple requirements to be run effectively on a Grid Computing Environment (GCE). The performance of such a scheduling algorithm promises to provide respectable waiting times, queue completion time, while achieving better utilization across the entire GCE. This is due to the concurrent considerations of many requirements. The proposed strategy is also independent of the units used, as long as the entire GCE operates with the same base units. The algorithm also offers no resistance in accommodating considerations for more resources in its allocation strategy. We evaluate the performance of our strategy with respect to several key performance indicators to quantify performance. Our study shows that MRS outperforms some of the commonly available schemes in place for a GCE.

III. Grid System Model

This section defines the Grid environment in which we will consider designing MRS strategy. We first identify certain key characteristics of resources as well as the nature of jobs in place.

A GCE consists of many diverse machine types, disks, and networks. In our resource environment, we consider the following.

1) Computational resources can be made up of individual servers, clusters or large multi-processor systems. 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 reserve the required resource for a job when instructed by the Grid Meta-Scheduler (GMS). It will then dispatch the job to the reserved resource for execution. The GMS thus treats all resources exposed under a single LRM as a single aggregated resource. We find this assumption to be reasonable as GMS usually does not have the ability to directly contact resources controlled by the LRM and can usually only submit to it.
2) The computation resources are of varying resource capabilities, and their resource capabilities and its changes are known instantaneously throughout the GCE. While it is not possible for this to happen reliably in reality, it is possible to achieve information accuracy within a limited time period. 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.
4) All resources have prior agreement to participate on the Grid. From this, we safely assume that all resources are accessible by every other participating node in the Grid.
5) Resources can include a number of shared CPUs, amount of memory to be shared, and the quota of file system space allocated for distributed computation. All these information are shared with the same units across the GCE.
6) We assume that the importance of the resources with respect to each other is identical.
7) The capacity for computation in a resource is provided in the form of GFlops. This is used as a gauge to standardize the performance of different CPU architectures in different sites.

The creation of the job environment is done through the investigation of the workload models available in the Parallel Workload Archive Models [9] and the Grid workload model available in [15]. 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. 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, and (b) massively parallel types written in either MPI or PVM.
2) The job 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.
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

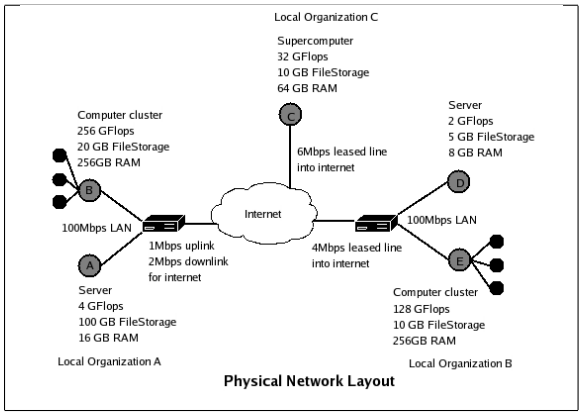

Fig. 1. Illustration of a physical network layout of a GCE.

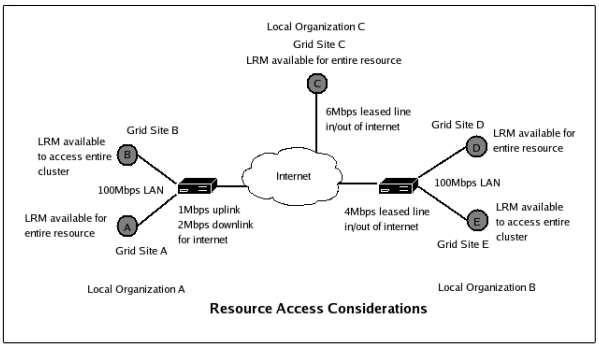

Fig. 2. Resource view of physical environment with access considerations

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.

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 is shown in the figure (2).

IV. Scheduling Strategy

The MRS strategy considers job requirements and resource capabilities based on some performance metrics and compute the best matching resource for a job to be dispatched to. This identification of resource will enable the job (regardless of the fact that if it is a parallel or serial application) to run most efficiently. MRS will also try to avoid over allocation of resources, so as to prevent the detrimental effects on other jobs which might need these resources to achieve efficiency in execution. By mapping a job to the best capable and matched resource, we hope to be able to improve the following metrics of performance measure.

1) Average Wait-Time (AWT)
2) Queue Completion Time (QCT)
3) Average Grid Utilization (AGU)

Our experiments consider two pairs of metrics for resource allocation from the perspectives of resources and jobs. These considerations include Resource Capabilities and Resource Requirements for both computation as well as data.

We define Computation capabilities, to mean *the capability of a resource to carry out a computation job efficiently*. This capability is dependent on many things, including CPU speed, the number of CPUs, the amount of RAM and network latencies. We also define Data capabilities as a representation of *the capability of a resource to be able to handle data efficiently*. This can be a factor of available bandwidth and the available file system. Computation and Data Requirements for a job simply means the respective *amount of computation or data resources required in order for the job to be executed successfully*.

A. Resource Capabilities

We first characterize the resource environment defined as a set $S = \{R_1, ..., R_m\}$, where m represents the total number of resources in our GCE. Each resource's capability to perform its computational role is identified to be dependent on effective GFlops (C), memory available (M) and file system (F).

These three parameters are essential in identifying the basic resources required for completing a job successfully when allocated to resources for processing. This however does not capture the overheads that might be incurred when a job requires inter-process communication or when it is distributed to multiple sites. An example of this is when a job makes use of MPI-G2 to distribute its job to multiple resources in a Grid. Such applications tend to run across multiple processors or resources, exploiting multiple CPUs, distributed memory architecture and distributed file systems. In order for such applications to achieve the best performance, the resources allocated to such an application must be able to satisfy the job requirements while being well connected to a set of co-operating resources.

In order to accommodate the requirements of such applications, we introduce a term called the *Resource Potential* to be included as part of a resource's computation capabilities. The potential of a resource R_i refers to 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 relate to Resource Potential as a form of "Virtual Distance", P_i of resource R_i, where $1 \leq i \leq m$, 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 removes all network complexities and "flattens" the bandwidth view of all the resources to the maximum achievable bandwidth between resources. We illustrate this "flattening" process in figure (3). The values C, M, F and P_i dynamically change with resource availability over time t. Thus we represent the computational capability of a site i as a set $S_c\{R_i, t\}$ and each item is represented by $f_i\{< C, M, F, P_i >, t\}$.

The data capabilities of a resource is mainly dependent on its available upload bandwidth. This basically allows us to

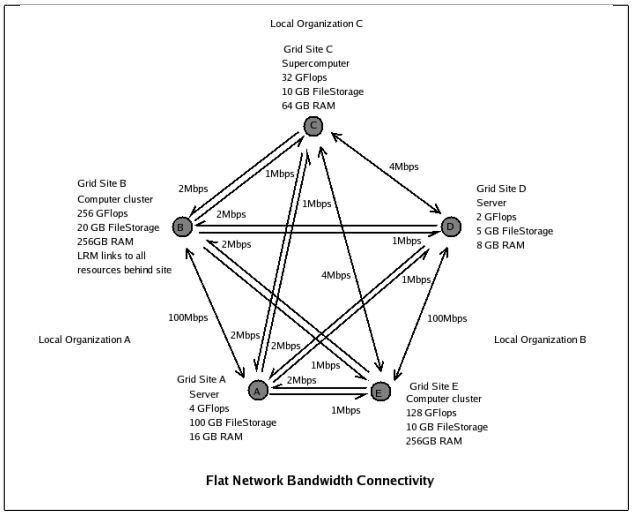

Fig. 3. Flattened network view of resources for computation of Potential

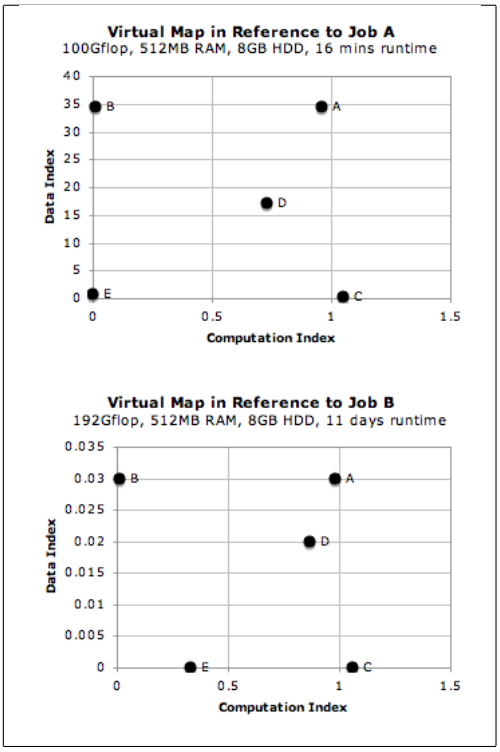

Fig. 4. A Virtual Map is created for each job to determine allocation

quantitatively determine how quickly data can be transmitted from one resources to the next. Available bandwidth also changes over time depending 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 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 this set is represented by $d_i\{< B >, t\}$ where B is defined above.

B. Job Resource Requirements

Like resource capabilities, jobs are also divided into computational as well as data resource requirements. These requirements are generally determined by the user (in our case, the workload model determines the requirements).

The job environment is characterized by $J = \{A_i, ..., A_j\}$. The computational requirement of each job A_j in the set of J jobs is represented by $g_j\{< C, M, F, P_{src} >, t\}$. The C, M, F parameters closely relate to the resource capabilities available in the GCE. The only difference is that in the case of resource requirements, these highlight the consumption of resources rather than the availability. P_{src} refers to the resource potential of where the data source file of job A_j resides. This follows from our assumption of the job execution environment in point 3 highlighted in section III. The data requirement is represented by $e_j\{< F, A^{runtime} >, t\}$. $A^{runtime}$ is the estimated runtime of the job j.

C. Allocation Strategy

With both the capability and requirement models defined, we now proceed to describe the allocation strategy used in the GMS. The MRS strategy is based on a co-ordinate based resource allocation scheme that aggregates various resources into a 2-dimensional index i.e. a co-ordinate based system. Sites participating in the Grid computes its own co-ordinate space within this 2-dimensional plane when presented with the job requirements. The resulting resource co-ordinates are then sent back and plotted back at the GMS which decides the order of nodes in which the job will be distributed to.

We call this resulting plot a *Virtual Map*. As the resources available in the GCE changes whenever a new job is submitted for execution, this Virtual Map must be re-created at each scheduling instance. An illustration of the virtual map is shown in figure (4). The euclidean distance from the origin denotes the "goodness of fit" for the resource requirements of a job.

In figure (4), the computation and data index is computed by equation (1) and (2) for each job in the queue. This is done through a series of operations. We first broadcast g_j and e_j to all resources in the GCE through a Grid Information Service[16]. These resources then concurrently takes a snapshot of its available resources, and obtains f_i and d_i. The calculation of the computation and data index (x_i, y_i) is also performed at this stage. These quantities are obtained by using equation (1) and (2). The results are subsequently sent back to the GMS which will automatically drop resources which don't reply within a time-out. The euclidean distance from the origin is then obtained from the resulting map and the order of resource distribution will be performed in an increasing order. We note that each resource requirement computed within x_i can be scaled by K to signify the importance of a job resource. The $floor()$ function truncates all values to be at a minium of 0.

$$x_i = \sqrt{\sum_{n=C,M,F} \frac{1}{K_n} \left\{ floor(0, (1 - \frac{f_i\{n\}}{g_j\{n\}})) \right\}^2}$$

$$+\frac{1}{K_P}\left\{floor(0,(1-\frac{f_i\{P_i\}}{g_j\{P_{src}\}}))\right\}^2} \quad (1)$$

$$y_i = \frac{e_j\{F\}}{d_i\{B\}}.\frac{1}{e_j\{A^{runtime}\}} \quad (2)$$

It should be noted that in this strategy, nodes evaluates its ability to accept a job in parallel due to the broadcast of the job requirements. This reflects well in a Grid, as participating resources retain its autonomy in how they want to share resources. thereby allowing them to alter the availability of resources in situ. This broadcasting also means that unit conversion can be carried out individually at the resource level so that there is no need for GCE resources to maintain the same base units, reinforcing the concept of distributed resource control. Although this broadcast and result collection mechanism can incur overheads in communication, it intrinsically provides a sync-ack protocol before a job gets dispatched, thereby reducing the need for very accurate Grid information mechanisms. The nature of broadcast also favours the MRS strategy in an environment where real-time scheduling is required as batching of jobs is not needed to create a run schedule.

From both equations (1) and (2), we note that (x_i, y_i) is dimensionless and x_i is a collective measure of how well a site is able to provide the individual resources required to execute a job. Sites that are able to provide more than available resources also do not suffer any penalty in selection. We also account, within the measure, for a way to ensure that the connectivity of the site is high. This is done by using a ratio of the Resource Potential as shown in the equation. This is such that in event of any job distribution, network latency will not become a significant overhead due to inappropriate resource allocation. We wish to draw the attention that at this point of co-ordinate computation, the resource is still unaware if any job distribution will happen. The y_i co-ordinate essentially computes the ratio of data transmission time to the runtime. This gives weight-age to the fact that if the job could possibly be processed more efficiently if it doesn't incur transmission overheads, it would prefer to execute locally. This therefore easily captures the resource dependencies of data and bandwidth, and illustrates how additional resource inter-dependencies can be addressed. We also note that in the calculation of the y_i co-ordinate, the $d_i\{B\}$ used to compute y_i for each resource would be the bandwidth required for each resource to obtain the data.

These independent considerations are then plotted on a Virtual Map, and brought together through the computation of euclidean distance from the origin. The result will then be the scheduling score of the site, where lower is better.

Figure (4) shows how two different jobs with different requirements[1] and run time would affect how the resulting plots on the Virtual Map. It is clear to see that as the ratio of the transmit time to run time of a job decreases, the strategy find it increasingly better to dispatch a job to a remote location for execution, while keeping in consideration the ability to best allocate its resources for minimal job distribution if possible.

[1]Figure (4) is illustrated with a data source location of node D from the preceeding figures.

D. System Implementation and Complexity

The system and strategy for MRS described in (IV-C) can be described as a class of Job Sharing strategies operating within a Multi-Site Computing model [18]. However, when MRS is compared to the models described in [18], clear stages in resource selection and scheduling algorithm does not exist. The computation of the indexes combines the selection and scheduling stages and thus reduces the fragmentation of resource considerations during resource allocation and scheduling.

In a strategy where 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-allocated and the time-to-schedule.

In MRS, the longest waiting job will always broadcast its resource requirements first. This gives the strategy the nature where the wait time for each job will tend to be reduced. This broadcast of requirements is of time complexity $O(n)$ as each site will only need to receive the resource requirements of a job once. This is delivered in a single transmission. However, due to broadcast, network latencies will be involved, which can lead to potential slow-downs in MRS. This can be easily prevented by "dropping" sites that does 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. It is noted that the time complexity affects the accuracy of the information within a given time window. This is because a higher ordered time complex algorithm will penalize the accuracy of the information at the point of job dispatch. The linear time complexity in MRS ensures that the information used in the strategy will never be more out-dated than a strategy that incurs a time complexity of $O(n^m)$, especially when there is a large number of resources to consider.

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 makes up the computational complexity of the strategy for each job. First, being 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 indexes provided by the participating sites is linear to the number of resources and axes we which 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.

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 has to be rescheduled due to the lack of resources or 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 SYNC-ACK communications protocol between the target site and the GMS was also used ensure that the site is reserved for the job dispatch. This component also allows for better synchronization between multiple GMSes if more than one is used. A cancellation broadcast was also introduced to notify all participating sites in the GCE to remove a unscheduled job from its cache. This keeps the entire GCE in sync of the jobs that are remaining to be scheduled. Potentially, multicast mechanisms could also be used to help reduce the overheads of broadcasting.

V. Performance Analysis and Discussions

In order to ascertain the performance of MRS, we compared it against the Backfilling strategy (BACKFILL) [11] and a job Replication (REP) strategy [3], which is similar to that used in SETI@Home [17]. We make use of similar Job Sharing and Multi-site environment as described earlier such that the intrinsic advantages of the algorithms can be observed.

The Lublin99 [10] workload model from [9] as well as workload models provided by[15] was used as the workload input. This provided workload profiles from San Diego Super-Computing Center (SDSC), Lawrence Livermore National Laboratory (LLNL) and The Royal Institute of Technology (KTH). An event driven simulator using a generated set of workload with 200 jobs running over 20 sites was implemented. This is to ensure that there will be a large number of jobs in the queue, such that the effects of the scheduling algorithm will be significant. The metrics described in section IV was used to quantify the performance and comparison. The

	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 I
EXPERIMENTAL RESULTS COMPARING BACKFILL, REP AND MRS

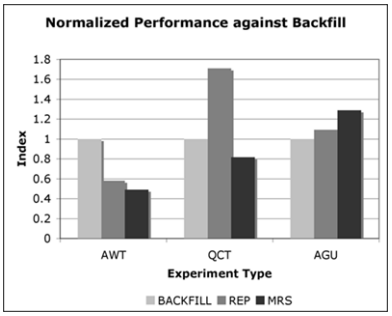

Fig. 5. Normalized comparison of simulation to Backfill Algorithm

results of our experiments are summarized in Table (I) and figure (5), and discussed in the section below.

In the simulation model, jobs are allowed to arrive in a stream over a span of 3 days. The various job requirements are modelled by the information provided in [10], [15] 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 point when the job begins its transmission to the execution node. In figure 5, we have normalised 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 greatly out-performs BACKFILL by 40% and 50%, when ran in a distributed environment, respectively. 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.

From Table I, we can clearly see that the utilization for BACKFILL is the lowest of the experiments. Average Grid Utilization (AGU) is measured as the average percentage of nodes in use throughout the execution time of the entire queue. REP and MRS both have increasing levels of utilization, which also accounts for the much shorter AWT. However, we note that the reason for REP achieving a shorter wait time is due to the nature of job replication. As a job gets replicated, the likelihood of being allocated a faster resource or bandwidth increases. As the first completed replicated job will result in the termination of the similar jobs, the wait time for other jobs will subsequently decrease as more jobs are injected into the system.

Although this has a positive effect on AWT, the effect of replication has produced a negative effect on the Queue Completion Time (QCT). We measure QCT as the time taken for the first job to enter the system up until the time the last job leaves the system. The degradation of performance is clearly reflected in figure 5, where REP is performing 71% slower in QCT. It is 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 has 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 lowered AWT if 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 when compared to a REP. The matching of resources using the computation and data indexes, also resulted 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 general, it is observed that MRS is able to render a performance that is much suited for scheduling resources over the Grid.

VI. Summary and Conclusions

In this paper, we have proposed a novel 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 shows how to obtain a minimal execution schedule through efficient management of available Grid resources. We have introduced the concept of virtual map that can be used by the scheduler to determine a best fit of resources for jobs prior to actual execution. We also introduced the concept of Resource Potential to identify a node that has least execution overheads for a job to execute.

The effectiveness of the MRS technique is studied against workloads based on SDSC, LLNL and KTH and our experiments have conclusively elicited several key performance features of MRS with respect to BACKFILL and REP. Possible extensions to MRS includes Quality-of-Service, economic considerations, achievable through the increase of dimensions in the MRS algorithm. This allows MRS to include other concepts for optimal scheduling in light of different resources and cost mechanisms considerations in Grid computing.

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] 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.

[4] 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.

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

[6] 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.

[7] 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.

[8] 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.

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

[10] 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.

[11] 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.

[12] D. A Lifka, "The ANL/IBM SP Scheduling System", *In the Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing*, Springer-Verlag, London, UK, Pages: 295 - 303, 1995.

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

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

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

[16] T. Zang, W. Jie, T. hung, Z. Lei, S. J. Turner, W. Cai, "The Design and Implementation of an OGSA-based Grid Information Service", *IEEE International Conference on Web Services (ICWS'04),* page 556, 2004

[17] 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.

[18] 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.