Adaptive Runtime Management of Spatial and Temporal Heterogeneity for Dynamic Grid Applications*

Xiaolin Li and Manish Parashar
The Applied Software Systems Laboratory
Department of Electrical & Computer Engineering
Rutgers University, Piscataway, NJ 08854, USA
Email: {xlli, parashar}@caip.rutgers.edu

Abstract

This paper addresses the runtime management of spatial and temporal heterogeneity in both, scientific applications and geographically distributed resources in Grid computing environments. The targeted applications are large-scale dynamic Grid applications which require large amount of computational resources typically spanning multiple sites and exhibit very long execution times. An adaptive runtime management framework with a hybrid space-time runtime management strategy (HRMS) is proposed by combining adaptive application partitioning and resource scheduling techniques. HRMS defines a set of flexible mechanisms and policies to adapt to the state of both applications and resources. As a proof-of-concept, a simulator for key features of this framework is being developed. Preliminary evaluation demonstrates that HRMS scheme improves performance and provides better speedup while using fewer resources on average.

Keywords: Runtime Management, Grid Computing, Heterogeneous Computing, Dynamic Load Balancing, Structured Adaptive Mesh Refinement

1 Introduction

Grid computing is rapidly emerging as the dominant computing paradigm for tackling grand challenges in disciplines including science, engineering, medicine and business [8, 9]. Its goal is to enable the coordinated selection, sharing and aggregation of geographically distributed resources including computers, networks, storage systems and specialized devices.

Emerging large-scale Grid applications in science and engineering require increasing amount of computing and storage resources. Three distinct characteristics of these applications are: (1) They are inherently large and require large amount of computational resources, typically spanning multiple sites on the Grid. Furthermore, the exact resource requirements are often not known a priori and depend on the application runtime behavior. (2) They may execute for days, weeks or months and often the exact execution time is not known a priori. For instance, it is not always known how long a scientific and engineering simulation will have to run before it provides meaningful insights into the phenomenon being modelled. (3) They are highly dynamic and heterogeneous in space and time. In addition, their dynamics and heterogeneity patterns are not known a priori.

Similarly, Grid environments are inherently large, heterogeneous and dynamic. Efficient scheduling and management in these environments poses a challenging problem. Current research efforts in resource management fall into three categories. Resource-centric approaches, which are based on the aggregated view of resources, schedule different applications among resources to improve resource utilization through spatial or temporal sharing. Examples include FCFS, Gang Scheduling and Backfilling techniques [7, 10]. Application-centric approaches assume static resource allocation and focus on partitioning specific applications and mapping sub-tasks to resources to improve application performance [3, 11, 18]. The third research efforts address the parameter-sweep applications for high throughput, such as AppLeS and Condor [2, 5].

This paper presents the design of an adaptive runtime management framework to manage the spatial and temporal heterogeneity and dynamics exhibited in both Grid applications and resources. In particular, it targets applications based on parallel Structured Adaptive Mesh Refinement (SAMR) techniques [1]. SAMR provides means for concentrating computational effort to small and localized regions in the computational domain. These techniques can lead to more efficient and cost-effective solutions to time dependent problems exhibiting localized features. As a result, applications based on these adaptive techniques are dynamic and heterogeneous in both space and time. Combined with dynamics and heterogeneity exhibited by Grid resources, the existing resource-centric or application-centric approaches cannot achieve efficient scheduling and management by themselves. A simple application-centric approach (referred to as the baseline scheme hereafter) is to reserve maximum feasible resources for maximum feasible execution time of the application, which is not known a priori. This strategy is clearly not practical because it results in low resource utilization due to excessive resource allocation. A typical resource-centric strategy would ignore individual application characteristics and schedule all applications in the same manner.

*The research presented in this paper is supported in part by the National Science Foundation via grants numbers ACI 9984357, EIA 0103674, EIA 0120934, ANI 0335244, CNS 0305495, CNS 0426354 and IIS 0430826 and by DOE ASCI/ASAP via grant numbers PC295251 and 82-1052856.

Thus, it will result in large waiting times and long execution times because the inherent heterogeneity of SAMR applications will cause significant load imbalance when using simple partitioning schemes in a resource-centric strategy. Sophisticated application partitioning strategies would take the special characteristics of applications into account and dynamically repartition the application to achieve better load balancing at runtime. Without considering the system dynamics and availability, application partitioning strategies will also result in undesirable performance. We believe an application-aware and resource-sensitive adaptive runtime management strategy is a better choice to tackle this challenging scheduling problem as it can take full advantage of both resource-centric and application-centric approaches.

The objectives of this paper are as follows. A hybrid space-time runtime management strategy (HRMS) is proposed and preliminary simulation results are presented to validate the intuition that adaptive resource allocation using HRMS potentially improves the overall performance of distributed SAMR applications. HRMS leverages and combines resource scheduling and application partitioning techniques. Overall, HRMS defines a set of mechanisms and policies to adapt to the state of both applications and resources and strives to minimize application completion time, reduce waiting time and improve resource utilization. The HRMS framework consists of partitioning, clustering, scheduling and hybrid strategies. Conceptually, HRMS employs clustering algorithms to create a hierarchy of clique regions. A clique region in the context of SAMR applications is a quasi-homogeneous computational sub-domain that is composed of physically connected sub-regions. HRMS then maps this clique hierarchy to resource groups. Criteria for adapting to the dynamics of resources and applications are also defined.

The rest of the paper is organized as follows. Section 2 presents the problem description. Section 3 presents the formulation and operations of the proposed hybrid space-time runtime management strategy. Section 4 presents some performance evaluation metrics and preliminary evaluation results for SAMR applications. Section 5 concludes the paper.

2 Problem Description

SAMR techniques track regions in the domain that requires additional resolution and dynamically overlay finer grids over these regions. These methods start with a base coarse grid with minimum acceptable resolution that covers the entire computational domain. As the solution progresses, regions in the domain requiring additional resolution are tagged and finer grids are over laid on these tagged regions of the coarse grid. Refinement proceeds recursively so that regions on the finer grid requiring more resolution are similarly tagged and even finer grids are overlaid on these regions. The resulting grid structure for the Structured Berger-Oliger AMR is a dynamic adaptive grid hierarchy [1] as illustrated in Figure 1.

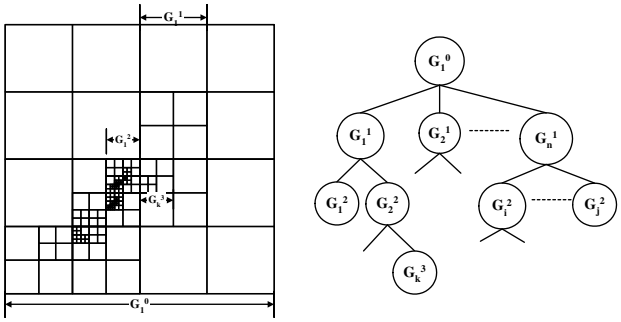

Figure 1: Adaptive Grid Hierarchy - 2D (Berger-Oliger AMR scheme)

We use a representative SAMR application, the 3-D compressible turbulence simulation kernel solving the Richtmyer-Meshkov (RM3D) instability, for our case study. The RM3D application is part of the virtual test facility (VTF) developed at the Caltech ASCI/ASAP Center [6]. The Richtmyer-Meshkov instability is a fingering instability which occurs at a material interface accelerated by a shock wave. This instability plays an important role in studies of supernova and inertial confinement fusion. A selection of snapshots and load dynamics for the RM3D adaptive SAMR grid hierarchy are shown in Figure 2. The load/workload in the figure is an abstraction of the computational requirement based on the number of numerical grid points on the grid, which is used to discretize the computational domain. Application variables are defined at each grid point and the numerical partial differential equation (PDE) operator is applied at each point. As a result, the total computational work (and storage) is proportional to the number of grid points. The heterogeneity in space is demonstrated in that, at each regriding step, the adaptively refined regions exhibit different computational, communication and storage requirements than other regions. The heterogeneity in time is demonstrated by the fact that the regions of refinement dynamically change as the simulation proceeds.

An inherent characteristic of the Grid is its heterogeneity in both time and space. A typical scenario of resource dynamics on two resource sites is illustrated in Figure 3. The temporal heterogeneity is represented by the variation of available capacity (number of available processors) of a single resource or resource site over time. The spatial heterogeneity is represented by the variation in the available resources across sites. In this paper, we consider the heterogeneity at a coarse-granularity. Specifically, we focus on space-sharing scenarios and leave the time-sharing cases for future work. The resource usage patterns presented are derived from synthesized traces based on the real traces from supercomputer centers [13]. More details will be presented in the experimental evaluation section.

3 Hybrid Space-Time Runtime Management Strategy

In this section, we develop a hybrid space-time runtime management strategy (HRMS). HRMS combines resource scheduling with application partitioning to address the application and system dynamics. Its overall operation flowchart is illustrated in Figure 4. From the flowchart, we can iden-

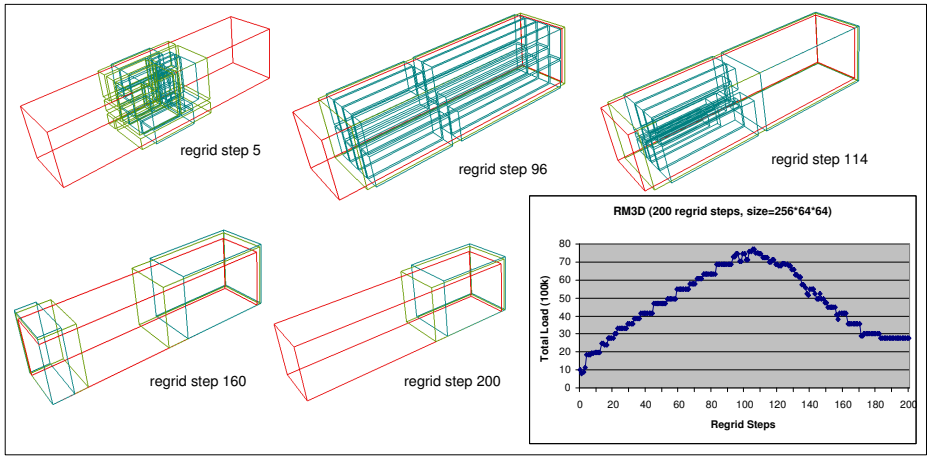

Figure 2: Spatial and Temporal Heterogeneity and Load Dynamics of a 3D Richtmyer-Meshkov Simulation using SAMR

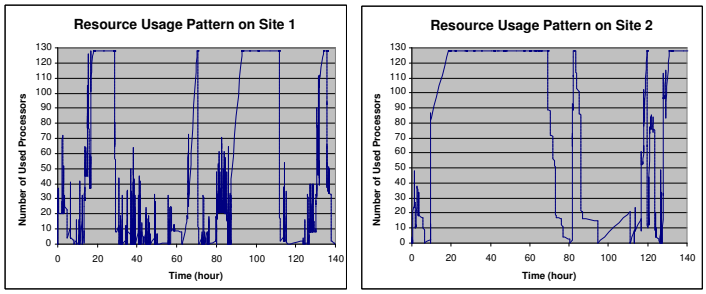

Figure 3: Spatial and Temporal Heterogeneity of Resources on Two Sites

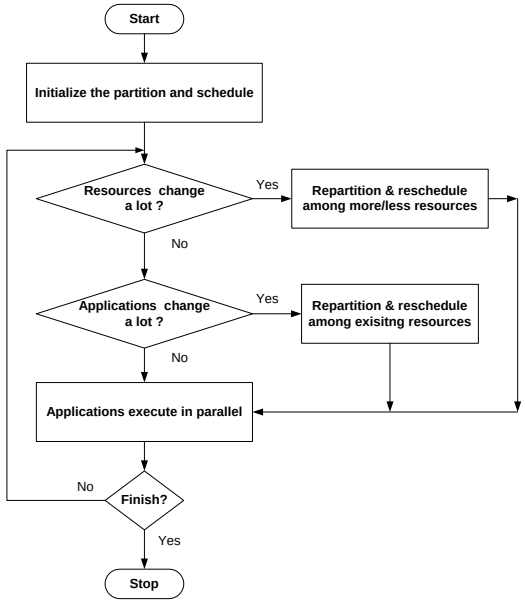

Figure 4: Flowchart of Hybrid Space-Time Runtime Management Strategy

tify four modes: "static" resource and application, "static" resource and dynamic application, dynamic resource and static application, dynamic resource and application. There are three major steps during the whole process, namely, initialization, repartitioning and rescheduling among existing resources, and repartitioning and rescheduling among more or less resources. HRMS primarily consists of the following components: partitioning, clustering, scheduling and hybrid strategies. Using these strategies, HRMS works in the following way. (1) It characterizes the application requirements hierarchically using the clustering algorithms to create a hierarchy of clique regions. (2) It schedules and maps the application hierarchy to the available resource group hierarchy. (3) Inside each resource group, it recursively applies the scheduling and partitioning algorithms. It may apply different partitioners for different cliques. (4) When application states change significantly, incremental repartitioning and rescheduling is performed in local resource groups. (5) When the available resource capacity changes significantly, HRMS performs rescheduling over all available resources. (6) When the resource capacity is sufficiently large and underutilized, we apply the application-level pipelining scheme by exploiting excessive resources. Alternatively, when the resource capacity is scarce, we adopt the application-level out-of-core scheme to enhance the survivability and performance of applications.

To ease the description, we define the following notations. L_Σ denotes the total amount of load/workload of the application; L_i denotes the workload assigned to the processor p_i; E_i denotes the time taken to process a unit load by the processor p_i. Thus, using these notations, we see that the computation time for processing L_i on the processor p_i is $L_i E_i$. Note that the processor p_i can denote a physical processor or a resource group. In the context of SAMR applications, the workload represents the computational and communication requirements for the physical domains. Due to the refinement on space and time, the workload at refinement level l is a function of number of grid points at the coarse level $L^{(0)}$, refinement level l, refinement factor r, and the dimension d of the physical domain, $L^{(l)} = (r^{d+1})^l \times L^{(0)}$. Thus the total load on a fully refined subdomain is $L^{all} = \sum_{l=0}^{nlev} L^{(l)}$, where $nlev$ denotes the number of refinement levels.

3.1 Load scheduling on heterogeneous systems

On heterogeneous systems, such as Grids, the load partitions shall be scheduled/mapped to processors in proportion to their computing power. Assume that we have a total work-

load of L_Σ to be scheduled onto np processors. The scheduling objective is to distribute load partitions such that all processors will stop processing at about the same time. Thus, we have the following $(np-1)$ equations for processing time.

$$L_{i+1}E_{i+1} = E_i \times L_i,\ i = 1, ..., np-1 \quad (1)$$

Since we have $L_\Sigma = \sum_{i=1}^{np} L_i$, we totally have np equations to solve np unknowns (L_i). Thus, we can obtain the load partitions assigned to each processor as follows.

$$L_1 = \frac{L_\Sigma}{1 + \sum_{i=2}^{np} \frac{E_1}{E_i}} \quad (2)$$

$$L_i = L_1 \times \frac{E_1}{E_i},\ i = 2, ..., np \quad (3)$$

Note that the above derivations are also applicable to homogeneous systems. These equations are used in the initialization and rescheduling phases as basic criteria for scheduling load portions to individual processors or resource groups with aggregated computing capacity. Communication cost is not taken into consideration in this simplified model. Since our clustering and partitioning schemes are based on space-filling curve (SFC) technique which possesses the desired locality-preserving property [15], the communication cost is potentially minimized.

3.2 Repartitioning and rescheduling among existing resources

In large parallel/distributed systems, the global information exchange and synchronization phase becomes a performance bottleneck. We have developed a hierarchical partitioning algorithm (HPA). The overall goal of HPA is to allow the distribution to reflect the state of the adaptive grid hierarchy and exploit it to reduce synchronization requirements, improve load-balance, and enable concurrent communications and incremental repartitioning and rescheduling. HPA partitions the computational domain into subdomains and assign these subdomains to dynamically configured hierarchical processor groups [12]. Furthermore, as mentioned in the introduction section, to exploit application characteristics, HRMS strives to cluster subregions with similar properties together to formulate a clique hierarchy. These clique regions are then further characterized and appropriate partitioning algorithms are applied to them [4, 17]. Two preliminary schemes have been proposed for clique generation, level-based clustering algorithm and segmentation-based clustering algorithm. The paper that describes these clique formulation schemes in detail is under preparation.

When resources remain relatively stable, we consider only the application dynamics. To specify when we need to repartition and reschedule the application subregions in a resource group A, we define the load imbalance factor (LIF) as follows:

$$LIF_A = \frac{\max_{i=1}^{A_n} L_i E_i - \min_{i=1}^{A_n} L_i E_i}{\sum_{i=1}^{A_n} L_i E_i} \quad (4)$$

where A_n denotes the total number of processors in the resource group A.

We set γ_A as the local imbalance threshold. When $LIF_A > \gamma_A$, the repartitioning will be conducted inside the local group.

3.3 Repartitioning and rescheduling among more or less resources

We measure the dynamics of resources by a simple parameter ΔR which is defined by the changing percentage of entire resource capability:

$$\Delta R(t) = \left| \frac{RC_\Sigma(t) - RC_\Sigma(t - \Delta t)}{RC_\Sigma(t - \Delta t)} \right| \quad (5)$$

where $RC_\Sigma(t)$ denotes the total available resource capacity at time t. If $\Delta R(t)$ is greater than a reschedule threshold β, HRMS repartitions and reschedules load among all resources (more or less).

When we have more resources, we trade in space (resources) for time (minimizing overall execution time) by applying the application-level pipelining scheme (ALP). In ALP scheme, multiple application patches with different refinement levels are processed in an overlapped manner. When the efficiency and benefits using ALP is below a certain threshold and the resource utilization is low, HRMS chooses to release unnecessary resources in order to improve the overall resource utilization. Alternatively, when there are insufficient resources available, we trade in time for space by applying the application-level out-of-core scheme (ALOC). ALOC scheme attempts to enhance the application survivability by explicitly managing application-level *pages*. The work on the details and analytical models of these two schemes are ongoing.

4 Experimental Evaluation

In this section, we present a preliminary simulation results to partially validate the HRMS framework. The simulation is focused on demonstrating the benefits of the adaptive resource allocation using HRMS over the static resource allocation using the baseline scheme. For that purpose, we have developed a Grid simulator, called GridMate [1]. In GridMate, the Grid system is composed of several computer sites. Totally, we set up 4 resource sites. On each site, there are 128 homogeneous processors. On different sites, computers are heterogeneous in computing speed, communication bandwidth and memory capacity. Each site has its local scheduler and its local job arrivals (as illustrated in Figure 3) which are synthesized from traces on several supercomputer centers [13]. Their workload model is based on workload logs from three sites, San Diego Supercomputer Center (SDSC), Los Alamos National Lab (LANL) and Swedish Royal Institute of Technology. In this model, the job sizes follow a two-stage uniform distribution, job execution times follow the hyper-Gamma distribution, and job arrivals follow two Gamma distributions [13].

[1] More details about GridMate can be found at `http://www.caip.rutgers.edu/~xlli/gridmate.htm`

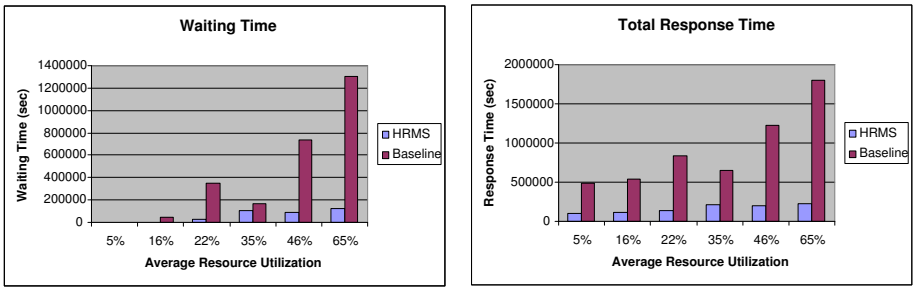

Figure 5: Waiting Time and Response Time: HRMS and Baseline Schemes

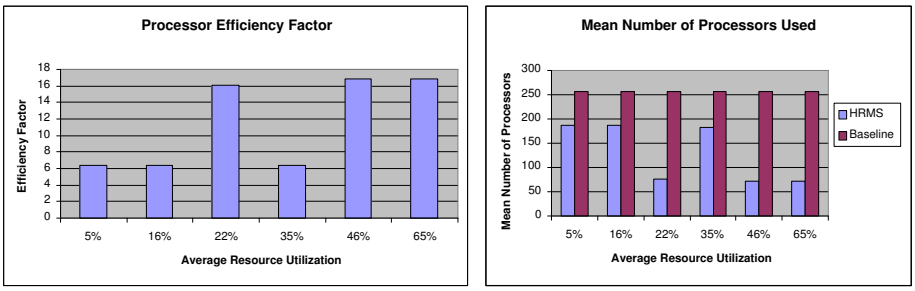

Figure 6: Processor Efficiency Factor and Mean Number of Processors Used: HRMS and Baseline Schemes

This system configuration resembles a scenario where a large-scale scientific application is submitted to a super-scheduler for execution on 4 geographically distributed supercomputer centers simultaneously. Furthermore, on each supercomputer center, there are local parallel/serial job arrivals. This is a representative scenario of the emerging Grid usage paradigms for solving grand challenge problems. While there exist several popular simulators for Grid computing, such as SimGrid, GridSim and MicroGrid [19], they cannot handle large-scale co-allocation, synchronized rescheduling and repartitioning scenarios as required in distributed SAMR applications.

GridMate adopts a two-level hierarchical partitioning and scheduling scheme. Initially, the super-scheduler coarsely partitions the physical domain into 4 subregions (cliques) with workload proportional to the available resources on 4 resource sites. These subregions are then submitted to local schedulers and further partitioned and assigned to the individual processor on each site. Repartitioning and rescheduling will be triggered when states of the SAMR application and resources change significantly. The targeted application is RM3D simulation kernel. Using GridMate simulator, its execution trace is submitted to the super-scheduler and executed across sites. The simulator is built on top of the primitive discrete-event simulation engine SimJava [16]. Additionally, using Java native interface (JNI), GridMate calls underlying partitioning routines implemented in GrACE, which is an object-oriented infrastructure in C++ for enabling parallel SAMR applications [14]. The performance evaluation metrics used are waiting time, execution time and response time for the RM3D job. Furthermore, to compare with the baseline scheme, we define a processor efficiency factor η and processor-time factor ς as follows.

$$\varsigma = \sum_{i=1}^{n} \sum_{j=1}^{s} (NC_j \times N_{i,j} \times \tau_{i,j}) \tag{6}$$

where, n is the total number of application iterations, s is the total number of sites, NC_j is the normalized capacity of one processor on site j, $N_{i,j}$ is the number of allocated processors, $\tau_{i,j}$ is the length of the i-th time interval and the subscript (i, j) denotes in the i-th time interval on the site j. ς^h is the processor-time factor for the HRMS scheme and ς^b is for the baseline scheme. The processor-time factor represents the normalized total computational resource consumption. Using the above equation, we define the mean number of processors used as,

$$\overline{N} = \frac{\varsigma}{T_{exe}} \tag{7}$$

where, T_{exe} is the total execution time.

Using the processor-time factor ς, we define the processor efficiency factor η by the following equation.

$$\eta = \frac{\varsigma^b}{\varsigma^h} = \frac{\overline{N}^b \times T^b_{exe}}{\overline{N}^h \times T^h_{exe}} \tag{8}$$

Figure 5 shows the waiting time and response time of the RM3D job with respect to the resource utilization using HRMS and baseline schemes respectively. The average resource utilization is measured for all resource sites with local job arrivals only. The simulation results show that the simple baseline scheme results in large waiting time due to its high requirement for large number of processors. The waiting time increases significantly as the resource utilization increases. However, using HRMS scheme, we observe a significant performance boost for the RM3D job due to its adaptive policies taking advantages of resource-centric and application-centric approaches. Compared to the baseline scheme, HRMS scheme achieves significant speedups.

To demonstrate the resource usage of HRMS and baseline schemes, Figure 6 shows the processor efficiency factor and mean number of processors used, which are defined in equations (7) and (8) respectively. For the baseline scheme, the

mean number of processors used is constant, 256 processors, due to its static resource allocation. Compared to the baseline scheme, the mean number of processors used for HRMS scheme is in the range from 70 to 190. One interesting observation is that the mean number of processors used for HRMS does not monotonically increase or decrease with respect to the resource utilization. This is because of the definition of $\overline{N}$ in the equation (7). Compared to the baseline scheme, HRMS scheme results in reduction on both the numerator and the denominator of the equation (7). As a comparison of these two schemes, the processor efficiency factor ranges from 6 to 17.

These preliminary simulation results demonstrate the feasibility and efficiency of HRMS strategy. Further simulation is ongoing, such as to perform sensitivity analysis of various rescheduling thresholds to gain more insights on the impact of these parameters. More realistic applications with various problem sizes are to be included to further evaluate its efficiency and scalability.

5 Conclusion

This paper presented an adaptive runtime management strategy HRMS to manage a new class of large-scale dynamic scientific applications with long execution times in Grid environments. The proposed strategy combines the resource scheduling and application partitioning techniques. Criteria for adapting to resource dynamics and application dynamics are defined. Preliminary simulation results demonstrate that HRMS outperforms the baseline scheme. Further simulations and experiments are ongoing to measure the performance of our strategies in various situations.

References

[1] M. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of Computational Physics*, 53:484–512, 1984.

[2] F. Berman and et al. Adaptive computing on the grid using apples. *IEEE Transactions on Parallel and Distributed Systems*, 14(5), May 2003.

[3] S. Chandra and M. Parashar. An evaluation of partitioners for parallel samr applications. In R. Sakellariou, J. Keane, J. Gurd, and L. Freeman, editors, *Euro-Par 2001*, volume 2150, pages 171–174. Springer-Verlag, August 2001.

[4] S. Chandra, J. Steensland, M. Parashar, and J. Cummings. An experimental study of adaptive application sensitive partitioning strategies for samr applications. In *2nd Los Alamos Computer Science Institute Symposium (also Best Research Poster at Supercomputing Conference 2001), October 2001.*, 2001.

[5] Condor. URL: `http://www.cs.wisc.edu/condor`.

[6] J. Cummings, M. Aivazis, R. Samtaney, R. Radovitzky, S. Mauch, and D. Meiron. A virtual test facility for the simulation of dynamic response in materials. *Journal of Supercomputing*, 23:39–50, 2002.

[7] D. G. Feitelson. A survey of scheduling in multiprogrammed parallel systems. Technical report, IBM Research Report RC19790(87657), Feb. 1995.

[8] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The physiology of the grid: An open grid services architecture for distributed systems integration, open grid service infrastructure wg, global grid forum, June 2002.

[9] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *International Journal of High Performance Computing Applications*, 15(3):200–222, 2001.

[10] E. Frachtenberg, D. G. Feitelson, F. Petrini, and J. Fernandez. Flexible coscheduling: Mitigating load imbalance and improving utilization of heterogeneous resources. In *International Parallel and Distributed Processing Symposium (IPDPS'03)*, Nice, France, 2003.

[11] X. Li and M. Parashar. Dynamic load partitioning strategies for managing data of space and time heterogeneity in parallel samr applications. In *The 9th International Euro-Par Conference (Euro-Par 2003)*, Klagenfurt, Austria, 2003.

[12] X. Li and M. Parashar. Hierarchical partitioning techniques for structured adaptive mesh refinement applications. *The Journal of Supercomputing*, 28(3):265 – 278, 2004.

[13] U. Lublin and D. G. Feitelson. The workload on parallel supercomputers: Modeling the characteristics of rigid jobs. *Journal of Parallel and Distributed Computing*, 63(11):1105–1122, 2003.

[14] M. Parashar. Grace. `http://www.caip.rutgers.edu/~parashar/TASSL/`.

[15] H. Sagan. *Space Filling Curves*. Springer-Verlag, 1994.

[16] SimJava. URL: `http://www.dcs.ed.ac.uk/home/hase/simjava/`.

[17] J. Steensland. *Efficient Partitioning of Structured Dynamic Grid Hierarchies*. PhD thesis, Uppsala University, 2002.

[18] J. Steensland, S. Chandra, and M. Parashar. An application-centric characterization of domain-based sfc partitioners for parallel samr. *Ieee Transactions on Parallel and Distributed Systems*, 13(12):1275–1289, 2002.

[19] Anthony Sulistio, Chee Shin Yeo, and Rajkumar Buyya. A taxonomy of computer-based simulations and its mapping to parallel and distributed systems simulation tools. *International Journal of Software- Practice and Experience*, 34(7):653–673, June 2004.