GridMate: A Portable Simulation Environment for Large-Scale Adaptive Scientific Applications

Xiaolin Li
Scalable Software Systems Laboratory
Department of Computer Science
Oklahoma State University
Stillwater, OK 74078, USA
Email: xiaolin@cs.okstate.edu

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

Abstract—In this paper, we present a portable simulation environment GridMate for large-scale adaptive scientific applications in multi-site Grid environments. GridMate is a discrete-event based simulator, consisting of abstractions of trace-based applications, computing resources, partitioners and schedulers, a 3D visualization tool, and user interfaces. It supports the analysis of runtime management strategies that address spatial and temporal heterogeneity in both adaptive scientific applications and geographically distributed resources in Grid computing environments. The targeted applications are a class of emerging large-scale dynamic Grid applications that require large amount of computational resources typically spanning multiple sites and exhibit long execution times. The underlying partitioning and scheduling algorithms are based on our previous work on the hybrid space-time runtime management strategy (HRMS). HRMS defines a set of flexible mechanisms and policies to adapt to state transitions of both applications and resources. The major components of GridMate are developed in Java, making GridMate highly portable and extensible. The design of GridMate and simulation results using GridMate are presented.

Keywords: Discrete-Event Simulation, Runtime Management, Resource Management, Grid Computing, Adaptive Applications

I. INTRODUCTION

The rapid progress in computing, networking and storage technologies has opened new opportunities and challenges for using the vast distributed and heterogeneous resources across organizations, regions, countries, and continents. Inspired by the pervasiveness, convenience, economics and open standards of the electrical power grid, Grid computing is emerging as the new computing paradigm for solving grand challenge problems in a wide spectrum of applications in science, engineering, business, and millitary [1], [2], [3]. Grid technologies are aimed to provide a service-oriented infrastructure that leverages open standard protocols and services to enable coordinated resource sharing and problem solving in dynamic multi-institutional virtual organizations [3]. A number of major Grid infrastructures are being developed and deployed [4], [5], [6] and many grand challenge problems, e.g. scientific applications employing Structured Adaptive Mesh Refinement (SAMR) techniques, are being tackled by exploiting the power of the Grid [7], [8], [9].

Resources on the Grid, including geographically distributed computers and storage systems, are inherently heterogeneous and dynamic. Due to the dynamics nature of grid platforms, results of most large-scale experiments obtained by one research group are hardly reproducible by other research groups (sometimes non-reproducible by the same research group) under identical system settings. In such case, simulation has been used extensively to validate the performance, conduct comparison study and gain insights from certain behaviors. To aid the evaluation of partitioning and scheduling strategies for large-scale adaptive applications in Grid environments [10], [11], we build a portable simulation environment, GridMate. Grid-based SAMR applications exhibit three key distinguishing characteristics: (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 modeled. (3) They

IEEE computer society

are highly dynamic and heterogeneous in space and time. In addition, their dynamics and heterogeneity patterns are not known a priori. Thus, the desired simulator needs to model the systems and applications such that these realistic characteristics are well reflected. The coupled heterogeneity and dynamism of resources and applications pose a significant challenge to managing resources and applications. It further complicates the evaluation of runtime management algorithms in such scenarios.

The rest of the paper is organized as follows. Section II presents related work and compares GridMate with several popular grid simulators. Section III describes the background and motivation. Section IV presents the conceptual architecture, scheduling mechanisms, and visualization of the GridMate. Section V shows experimental evaluation results and discussions. Section VI concludes the paper.

II. RELATED WORK

Grid environments are inherently heterogeneous and highly dynamic. Further, the ever-increasing system complexity, scale and diversity of software and hardware make system management a significant challenge. To attack this challenge, many resource management systems have been developed, such as Globus [12], [13], [1], [3], Condor [14], [15], AppLeS [16], [17] and Legion [18].

Performance evaluation of these resource management systems and algorithms plays a critical role in studying, calibrating, and comparing various techniques and systems for Grids. However, in Grid environments, the capacity and availability of resources change with time, along with a wide spectrum of dynamic applications. Due to the inherent dynamism and heterogeneity in Grid environments, it is quite difficult to obtain repeatable and comparable performance evaluation under identical system setups. Hence, simulation techniques are adopted. Simulation has been widely used for modeling and studying real-world systems and phenomena. A large number of simulation tool kits and libraries have been developed, including NS2 [19], Ptolemy [20], Sim-Java [21]. However, because Grid computing involves complex interacting components, there are only a few simulators that can model Grid environments, including MicroGrid [22], SimGrid [23], and GridSim [24].

MicroGrid, developed at UCSD, is based on the Globus Toolkit [12]. It offers a virtual Grid environment for simulating the execution of real applications. As an emulator, MicroGrid produces quite accurate simulation results. However, the simulation modeling and configuration process is quite demanding. In addition, due to its emulation nature, simulation based on MicroGrid is quite time-consuming. The SimGrid toolkit, developed also at UCSD, features flexible application scheduling mechanisms. It supports modeling of time-shared resources and applications from realistic traces. The GridSim toolkit, developed at University of Melbourne, is a Java-based simulation tool mainly for bag-of-task applications. It supports modeling of space-shared and time-shared large-scale resources in Grid environments and also supports the simulation of economy-based resource scheduling policies.

The Grid simulators described above enable simulating a wide spectrum of scenarios in Grid environments. However, they consider mainly resource heterogeneity and do not explicitly address the coupled space-time heterogeneity of both resources and SAMR applications. As a result, using these simulation toolkits, one needs to manually create the graph representing the SAMR domain, manually partition it, and assign the partitions to the heterogeneous resources.

III. MOTIVATION

SAMR applications are highly dynamic and exhibit space-time heterogeneity. We use a representative SAMR application, the 3-D compressible turbulence simulation kernel solving the Richtmyer-Meshkov (RM3D) instability, for our case study. RM3D application is part of the virtual test facility (VTF) developed at the Caltech ASCI/ASAP Center [25]. 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 1. 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 are dynamically changing as time goes by.

In Grid environments, we are confronted with a new dimension of complexity. Particularly, Grid systems consists of largely different software and hardware resources with changing capacity and availability and are inherently dynamic and heterogeneous. To demonstrate these characteristics, we show a typical scenario with two
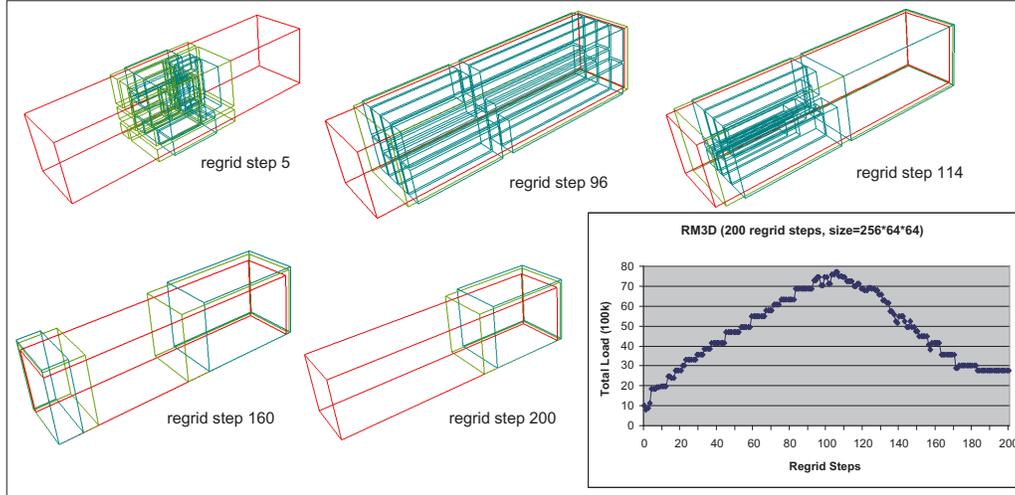
Fig. 1. Spatial and Temporal Heterogeneity and Load Dynamics of a 3D Richtmyer-Meshkov Simulation using SAMR
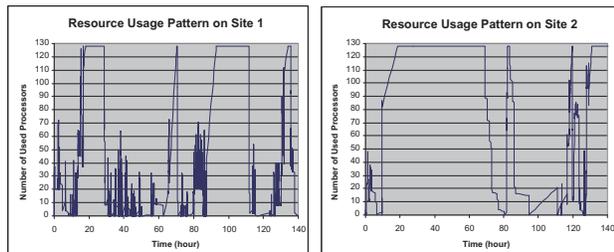

Fig. 2. Spatial and Temporal Heterogeneity of Resources on Two Sites

resource sites in Figure 2. The temporal heterogeneity is represented by the variation of available capacity (number of available processors) of a single resource site over time. The spatial heterogeneity is represented by the variation in the available resources across sites. In this paper, we consider the resource 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 [26]. More details will be presented in the experimental evaluation section.

IV. GRIDMATE: COMPANION FOR LARGE-SCALE ADAPTIVE APPLICATIONS IN GRIDS

To explicitly address the coupled heterogeneity of both applications and resources, we design the simulator to leverage both application partitioning and resource scheduling techniques. Thus, our main tasks are to model systems, applications, application partitioning and resource scheduling heuristics. Following this rationale, we build the Grid simulator GridMate. This section presents the trace-based simulation, the conceptual architecture, the scheduling and operations and visualization features in details.

A. Trace-based Simulation

GridMate can process either trace-based application profiles or randomly generated application runtime states. To achieve realistic simulation results, application traces obtained through execution of realworld applications can yield more precise insights of the applications. In GridMate, the trace-based simulation process is illustrated in Figure 3.

The input to the simulator is obtained by the following three steps: (1) A trace of the Grid hierarchy is obtained by executing the application for a single processor and the resulting parameters are dumped into a trace file. (2) This trace file is fed to a partitioner. The partitioner implements the partitioning scheme of choice. The partitioning scheme allocates various bounding boxes at different levels and timesteps to processors. This result is dumped to an output parameters file. (3) This output parameters file is further fed into the scheduler and execution engine. The scheduler applies two-level scheduling strategies to map to multi-site grid resources. The execution engine then executes the trace and measures application performance in terms of communication overheads, execution time, waiting time, and response time, and system (partitioning and scheduling
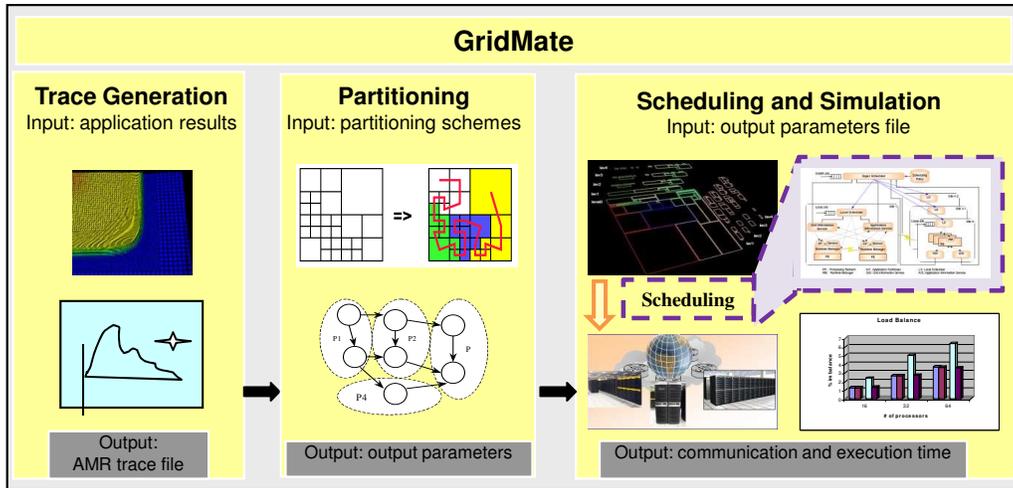
Fig. 3.   Trace-based Simulation in GridMate

efficiency) performance in terms of load balance, overheads, and other metrics. From the trace, the computation time is a function of the associated workload on the size of the domain (e.g. number of cells in a subdomain in 2D applications or number of unit blocks in a subdomain in 3D applications). Similarly, the communication time is a function of the associated workload on the boundary cells for exchanging information.
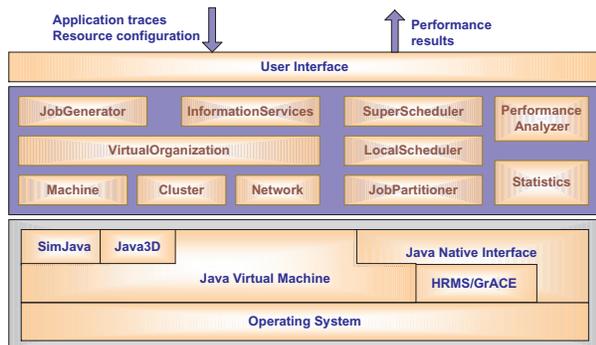
B. Conceptual Architecture

Fig. 4.   System Architecture of GridMate

The multi-layer system architecture of GridMate is illustrated in Figure 4. The bottom layer is the operating system. On the top of the operating system, we have three components, Java Virtual Machine (JVM), GrACE and Java Native Interface (JNI). JVM provides the portable runtime support for Java bytecoded files. Because our specific application/job partitioners are implemented using the GrACE toolkit which is

implemented in C++, we add a JNI wrapper layer to expose partitioning and other services to the simulator. On top of the JVM, we use the discrete-event simulation tool SimJava [21]. GridMate is built based on SimJava and GrACE (via JNI) [27]. It consists of the following major components: job generators, information services, virtual organization (machines, clusters, networks), local scheduler, super scheduler and performance monitor and analyzer. The input to the GridMate are a set of application traces including local jobs and SAMR jobs, resource configuration and scheduling policies. The output from the GridMate are various performance results based on the performance metrics defined in the next section.

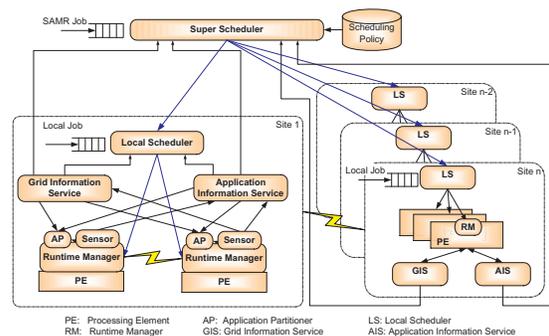### C. Scheduling Architecture and Operations

Fig. 5.   Scheduling Architecture of HRMS on a Multi-Site Grid

Handling the coupled heterogeneity of SAMR applications in complex grid computing environments is challenging. Figure 5 shows the scheduling architecture

85

of GridMate. It attempts to incorporate mechanisms to enable hierarchical runtime management and self-adaptivity. In particular, the conceptual architecture consists of three different levels: the overall system level (Grid), local site level (VO) and individual machine level. Key components are:

- Super Scheduler (SS): The target SAMR job is submitted to the super scheduler. SS makes scheduling decisions according to scheduling policies and current runtime states of SAMR jobs and resources. SS is in charge of dynamic co-allocation of the SAMR job to different resource sites through their local schedulers. In the Grid, there can be a variety of SSs for different classes of big jobs. In this paper, we only consider the case when there is only one SS for the SAMR job.
- Local Scheduler (LS): There is a local job queue associated with each local scheduler. These local jobs could be batch or interactive with various job specifications, such as number of processors required, execution time, deadline etc. We differentiate these local jobs with SAMR jobs. LS makes scheduling decisions according to its local scheduling policies for local jobs and SAMR jobs.
- Runtime Manager (RM): Runtime managers are organized in a hierarchical fashion. An RM is composed of application partitioners and sensors.
- Application Partitioner (AP): These are adaptive application-centric partitioners specialized for SAMR jobs. An AP resides in each processor in order to monitor the runtime requirements of applications and strives to improve performance by repartitioning and balancing for the target dynamic applications. APs take into account the application runtime characteristics to make partitioning or repartitioning decisions on behalf of SAMR jobs. Several partitioning strategies have been presented in our previous papers [10], [11].
- Sensors: These sensors monitor both resource and application runtime status. Grid information service (GIS) pulls the resource information from these sensors. Application information service (AIS) gathers the application runtime information from these sensors also. Resource sensors can be implemented using NWS [17], while application sensors are embedded into each application sub-task.
- Application Information Service (AIS): AIS collects the updated information of application runtime states from application sensors. AIS resides in each

local site and thus enables the aggregation of application states in a hierarchical fashion.
- Grid Information Service (GIS): GIS collects the updated information of resource states from resource sensors. It provides resource information to super scheduler and application partitioners so that they can make scheduling and partitioning decisions according to policies and current resource status.
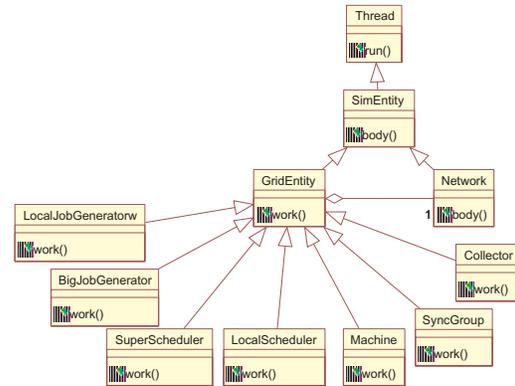

Fig. 6.   Class Diagram of GridMate Entities

As required by SimJava, all the simulation entities are derived from the "Sim_entity" class. Figure 6 illustrates a class diagram describing the inheritance hierarchy of major simulation entities. In the figure, GridEntity defines an abstract method *work()* to wrap the *body()* method as required by SimJava. Further, all these entities are derived from *Thread* of Java runtime library. When the simulation starts, all these entities will be instantiated and will execute in parallel as independent threads. This multi-thread feature is crucial to simulate parallel applications. Every GridEntity object and its derived object is equipped with a *Network* communication channel, which enables flexible and transparent communication modeling. In the simulation model, the communication cost is associated with the sender's messaging overhead, which is proportional to the message size plus a small constant overhead. The non-blocked send approach is adopted such that a sender will not be blocked even though the corresponding receiver is not ready receiving. On the receiver's side, if the message is available, there is no communication delay for the receiver; if the message is not available, the receiver has options to process other tasks or to be blocked until the message arrives. This communication model simulates the non-blocking message passing paradigm (such as *MPI_Isend()* and

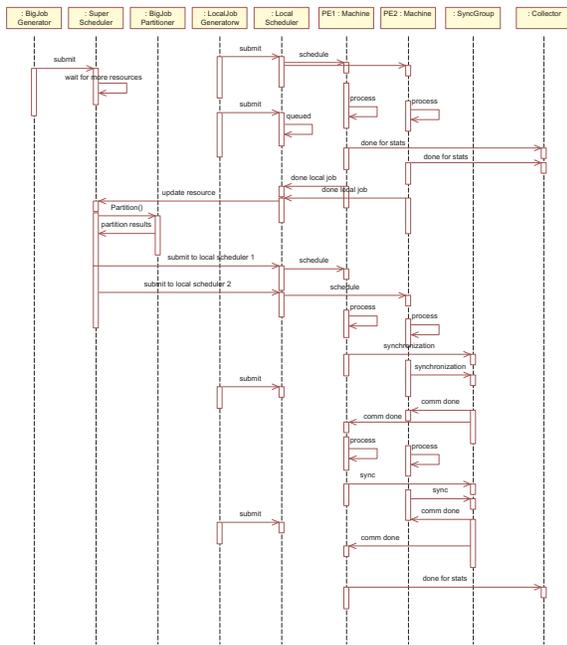*MPI_Irecv()*), which is a common practice in parallel applications.

Fig. 7.   Sequence Diagram for Interaction among GridMate Entities

The sequence diagram in Figure 7 illustrates the primary interactions among GridMate entities. Note that the local job generators and global job generator execute in parallel. Their job arrivals can be overlapped and result in resource contention and jobs queued at local schedulers or super scheduler. For local jobs, we assume that the parallel execution times obtained from application traces include all processing time, communication time and other overheads. Since we intend to model the exact computation and communication patterns in parallel SAMR applications, for SAMR jobs, we explicitly consider the communication cost incurred during synchronization after each iteration due to resource heterogeneity and load imbalance.

D. 3D Visualization Tool

GridMate also features a 3D visualization tool that provides an interface to study the application behaviors at runtime and may yield more insights on improving partitioning and scheduling algorithms for such highly dynamic applications. The visualization tool is developed based on the Java 3D library and specification, which is portable on multiple platforms. The visualization inter-
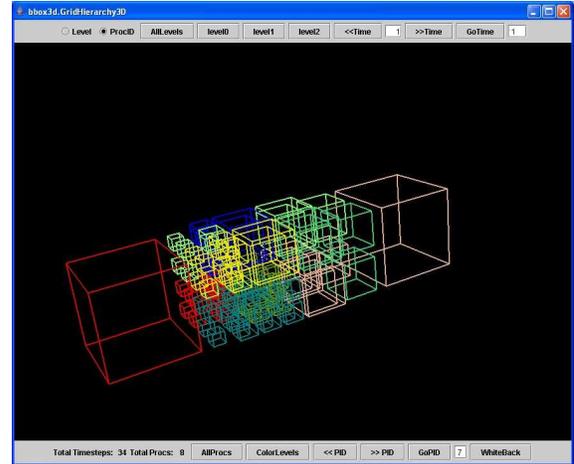

Fig. 8.   The Visualization Interface of GridMate

face is shown in Figure 8. In the figure, a snapshot of the Richtmyer-Meshkov (RM3D) instability application [25] is visualized.
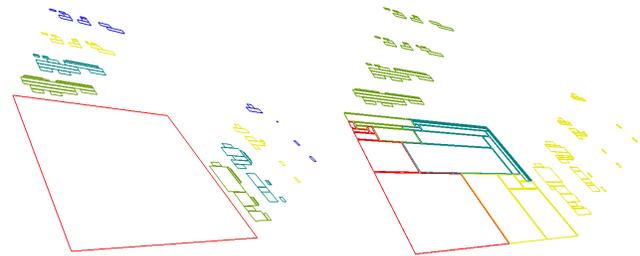

Fig. 9.   Visualizing Decomposable Computational Domains in Vertically or Horizontally. The figure in the left depicts patches on five refine levels (horizontal/level-based decomposition) in different colors; the figure in the right depicts detached subdomains of five refine levels (vertical/domain-based decomposition).

The visualization tool is flexible and capable to visualize decomposed computation domains as shown in Figure 9. This figure shows a snapshot of the transport2D application that solves heat transport equations. This feature is very useful at studying the dynamic behaviors of the adaptive applications and also analyzing the effects of various partitioning algorithms including patch-based and domain-based partitioning algorithms.

V. EXPERIMENTAL EVALUATION

A. System Setup

In GridMate, the Grid system is composed of several computer/resource sites. For the simulation results shown in this section, we set up totally four 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 follow the workload model presented in [26]. This 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. In this paper, we will focus our study mainly on partitioning and scheduling SAMR jobs. For local job scheduling, a substantial research effort already exists [28].

The target application is RM3D [25]. Its execution trace is submitted to the super-scheduler and executed across sites. The performance evaluation of HRMS strategies is compared with the baseline scheme. The baseline scheme statically allocates resources such that they meet the peak requirement of the SAMR application.

B. Evaluation Metric

The performance evaluation metrics used are waiting time, execution time and response time for the SAMR job. Additionally, to compare with the baseline scheme, we define a processor efficiency factor η and processor-time factor ς as follows.

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

where, n is the total number of application iterations/phases, 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 denoted for the HRMS scheme and ς^b for the baseline scheme. This equation represents the normalized total computational resource consumption. The physical meaning of ς could be interpreted as the total execution time if the application is assigned to a single standard processor (its $NC = 1$).

Thus the mean number of processors used is defined by,

$$\overline{N} = \frac{\varsigma}{T_{exe}} \qquad (2)$$

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.

$$\eta = \frac{\varsigma^b}{\varsigma^h} = \frac{\overline{N}^b \times T_{exe}^b}{\overline{N}^h \times T_{exe}^h} \qquad (3)$$
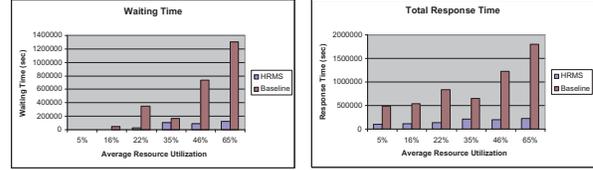
C. Simulation Results

Fig. 10.   Waiting Time and Response Time: HRMS and Baseline Schemes

Figure 10 shows the waiting time and response time of the SAMR 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. While using HRMS scheme, we observe a significant performance boost for the SAMR job due to its adaptive policies taking full advantages of resource-centric and application-centric approaches. Compared to the baseline scheme, HRMS scheme achieves significant speedups.
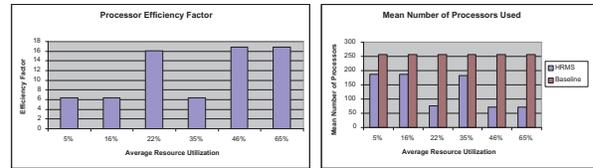

Fig. 11.   Processor Efficiency Factor and Mean Number of Processors Used: HRMS and Baseline Schemes

To demonstrate the resource usage of HRMS and baseline schemes, Figure 11 shows the processor efficiency factor and mean number of processors used, which are defined in equations (3) and (2) respectively. For the baseline scheme, its 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

88

not monotonically increase or decrease with respect to the resource utilization. This is because of the definition of $\overline{N}$ in the equation (2). Compared to the baseline scheme, HRMS scheme results in reduction on both the numerator and the denominator of the equation (2). As a comparison of these two schemes, the processor efficiency factor ranges from 6 to 17. These simulation results demonstrate the benefits of using HRMS strategies compared to the baseline scheme.

VI. CONCLUSION

We presented the design and performance evaluation of a grid simulator GridMate for a class of large-scale adaptive scientific applications. The trace-based simulation process, conceptual architecture, scheduling architecture, detailed operations, and visualization of GridMate are described. Simulation results confirm our observations from the real experiments on a supercomputer cluster: HRMS strategies outperform the baseline scheme through judiciously taking into consideration of the space-time heterogeneity. The future work to extend the simulator for online simulation, i.e., to simulate and predict the runtime behavior on multi-site grids for a SAMR application that is executing on a single-site cluster, is ongoing. In addition, based on Java, GridMate also makes it possible to be deployed on a web portal for public usage, evaluation, and analysis.

REFERENCES

[1] 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.

[2] 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.

[3] I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 2nd edition, 2004.

[4] European DataGrid. URL: http://eu-datagrid.web.cern.ch/eu-datagrid/.

[5] TeraGrid. URL: http://www.teragrid.org/.

[6] Chinagrid. URL: http://www.chinagrid.edu.cn/.

[7] BIRN, Biomedical Informatics Research Network project. URL: http://www.nbirn.net.

[8] Earth System Grid. URL: https://www.earthsystemgrid.org/.

[9] GriPhyN, Grid Physics Network project. URL: http://www.griphyn.org/.

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

[11] X. Li and M. Parashar. Hybrid runtime management of space-time heterogeneity for parallel structured adaptive applications. *IEEE Transactions on Parallel and Distributed Systems*, 18(9):1202–1214, 2007.

[12] Globus. URL: http://www.globus.org.

[13] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, and S. Tuecke. A resource management architecture for metacomputing systems. In *Lecture Notes in Computer Science*, volume 1459, 1998.

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

[15] C. Liu, L. Yang, I. Foster, and D. Angulo. Design and evaluation of a resource selection framework for grid applications. In *11 th IEEE International Symposium on High Performance Distributed Computing HPDC-11 20002 (HPDC'02)*, 2002.

[16] F. Berman, R. Wolski, H. Casanova, W. Cirne, H. Dail, M. Faerman, S. Figueira, J. Hayes, G. Obertelli, J. Schopf, G. Shao, S. Smallen, S. Spring, A. Su, and D. Zagorodnov. Adaptive Computing on the Grid Using AppLeS. *IEEE Transactions on Parallel and Distributed Systems*, 14(5):369–382, 2003.

[17] R. Wolski, N. T. Spring, and J. Hayes. The network weather service: a distributed resource performance forecasting service for metacomputing. *Future Generation Computer Systems*, 15(5-6):757–768, 1999.

[18] Legion. URL: http://legion.virginia.edu/.

[19] L. Breslau, D. Estrin, K Fall, S Floyd, J Heidermann, A Helmy, P Huang, S McCanne, K Varadhan, Y Xu, and H. Yu. Advances in network simulation. *IEEE Computer*, 33(5):5967, 2000.

[20] X. Liu, J. Liu, J. Eker, and E. A. Lee. Heterogeneous modeling and design of control systems. In T. Samad and G. Balas, editors, *Software-Enabled Control: Information Technology for Dynamical Systems*. IEEE Press, New York, 2003.

[21] Simjava. URL: http://www.dcs.ed.ac.uk/home/hase/simjava/.

[22] H.J. Song, X. Liu, D. Jakobsen, R. Bhagwan, X. Zhang, K. Taura, and A. Chien. The microgrid: A scientific tool for modeling computational grids. In *IEEE Supercomputing (SC2000)*, Dallas, TX, November 2000. IEEE Computer Society Press.

[23] A. Legrand, L. Marchal, and H. Casanova. Scheduling distributed applications: The simgrid simulation framework. In *the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid2003)*, Tokyo, Japan, May, 2003. IEEE Computer Society Press.

[24] R. Buyya and M. Murshed. Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *Concurrency and Computation: Practice and Experience*, 14(13-15):11751220, 2002.

[25] 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.

[26] 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.

[27] Grace. URL: http://www.caip.rutgers.edu/TASSL/Projects/GrACE/.

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

89