# GridARM: AN AUTONOMIC RUNTIME MANAGEMENT FRAMEWORK FOR SAMR APPLICATIONS IN GRID ENVIRONMENTS*

**Sumir Chandra†, Manish Parashar‡, and Salim Hariri§**

**Abstract**

Structured adaptive mesh refinement (SAMR) techniques offer the potential for accurate solutions of physically realistic models of complex physical phenomena, and are being effectively used in many domains including computational fluid dynamics, numerical relativity, astrophysics, subsurface modeling and oil reservoir simulation. However, the inherent space-time heterogeneity and dynamism of SAMR applications coupled with a similarly heterogeneous and dynamic execution environment such as the computational Grid present significant challenges in application composition, runtime management, optimization, and adaptation. This paper presents the design of GridARM – an autonomic runtime management framework that monitors application and system state and provides appropriate distribution, configuration, scheduling, and adaptation strategies to optimize performance and support the efficient and scalable execution of SAMR implementations in Grid environments.

***Keywords:*** *GridARM framework, Autonomic runtime management, Structured adaptive mesh refinement, Grid computing.*

## 1 Introduction

Dynamically adaptive techniques such as structured adaptive mesh refinement (SAMR) (Berger 1984) can yield highly advantageous ratios for cost/accuracy when compared to methods based upon static uniform approximations. SAMR provides a means for concentrating computational effort to appropriate regions in the computational domain. These techniques can lead to more efficient and cost-effective solutions to time dependent problems exhibiting localized features. Distributed implementations of SAMR methods offer the potential for accurate solutions of physically realistic models of complex physical phenomena and are being effectively used in several application domains including computational

* This research was supported in part by NSF via grants numbers ACI 9984357 (CAREERS), EIA-0103674 (NGS) and EIA-0120934 (ITR), and by DOE ASCI/ASAP (Caltech) via grant numbers PC295251 and 1052856.

† ECE Dept., Rutgers University, Piscataway, NJ, USA; E-mail: sumir@caip.rutgers.edu

‡ ECE Dept., Rutgers University, Piscataway, NJ, USA; E-mail: parashar@caip.rutgers.edu

§ ECE Dept., University of Arizona, Tucson, AZ, USA; E-mail: hariri@ece.arizona.edu

fluid dynamics (Berger 1983), astrophysics (Bryan 1999), and subsurface modeling and oil reservoir simulation (Parashar 1997). The phenomena being modeled by the SAMR applications are, however, inherently multi-phased, dynamic, and heterogeneous (in time, space, and state) requiring very large numbers of software components and very dynamic compositions and interactions between these components for efficient execution.

Furthermore, the Grid (Foster 1998) is rapidly emerging as the dominant paradigm for wide area distributed computing. Its goal is to provide a service-oriented infrastructure that leverages standardized protocols and services to enable pervasive access to, and coordinated sharing of geographically distributed hardware, software, and information resources. The Grid infrastructure is heterogeneous and dynamic, globally aggregating large numbers of independent computing and communication resources, data stores, and sensor networks. The inherent space-time heterogeneity and dynamism of SAMR applications coupled with a similarly heterogeneous and dynamic execution environment such as the computational Grid results in development and management complexities that break current paradigms, and present significant challenges in application composition, runtime management, optimization, and adaptation. The complexity, heterogeneity, and dynamism associated with scientific applications has made current programming environments and infrastructure unmanageable and insecure, and has led researchers to consider alternative programming paradigms and management techniques that are based on strategies used by biological systems. Systems based on this approach, known as autonomic computing (Kephart 2003), have the capabilities of being self-defining, self-healing, self-configuring, self-optimizing, self-protecting, contextually aware, and open.

This paper presents the design of GridARM – an autonomic runtime management framework that monitors application and system state and provides appropriate distribution, configuration, scheduling, and adaptation strategies to optimize the performance of SAMR applications. The GridARM framework models the computational resources as a hierarchy of virtual resource units (VRUs) and the application domain as a hierarchy of virtual computational units (VCUs). GridARM monitors and characterizes application and system state, deduces the appropriate optimization strategy to manage the application at runtime, and then maps and executes VCUs onto available VRUs in a hierarchical manner. The overall goal of GridARM is to manage dynamism and space-time heterogeneity and support the efficient and scalable execution of SAMR implementations in Grid environments.

The rest of the paper is organized as follows. Section 2 presents an overview of SAMR and the motivations and challenges for autonomic runtime management of SAMR applications. Section 3 presents an architectural overview of the GridARM framework. Section 4 details the design and operation of the components of the GridARM framework. Section 5 describes the current status of the framework and evaluation of its prototype components. Section 6 presents concluding remarks.

# 2 SAMR: Motivations and Challenges

## 2.1 Overview of SAMR

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 overlaid 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 adaptive grid hierarchy of the SAMR formulation by Berger and Oliger (Berger 1984) is shown in Figure 1. Figure 2 shows a 2-D snapshot of a sample Grid-based SAMR application that investigates the simulation of flames (Ray 2003). This figure shows the mass-faction plots of various radicals produced during the ignition of $H_2$-Air mixture in a non-uniform temperature field with 3 "hot-spots". The combustion application is highly dynamic and heterogeneous in space and time, and is representative of the nature of simulations targeted by this research.

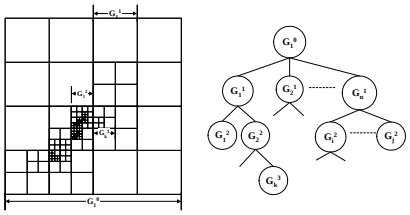

Figure 1: 2-D adaptive grid hierarchy (Berger-Oliger AMR scheme)

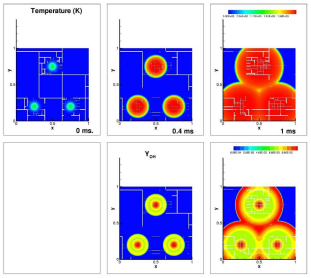


Figure 2: Flames simulation: ignition of $H_2$-Air mixture in a non-uniform temperature field (Courtesy: J. Ray, et al, Sandia National Labs, Livermore)

## 2.2 Challenges in SAMR

Parallel/distributed implementations of SAMR applications lead to interesting computational and computer science challenges in dynamic resource allocation, data-distribution and load balancing, communications and coordination, and runtime management. As a result, key requirements for an efficient runtime management framework for SAMR applications include:

- **Dynamic Partitioning Support:** The overall efficiency of the algorithms is limited by the ability to partition the underlying data-structures at runtime so as to expose all inherent parallelism, minimize communication and synchronization overheads, and balance load.

- **Adaptive Communication Support:** A critical requirement while partitioning adaptive grid hierarchies is the maintenance of logical locality, both across different levels of the hierarchy under expansion and contraction of the adaptive grid structure, and within partitions of grids at all levels when they are decomposed and mapped across processors. The former enables efficient computational access to the grids while the latter minimizes the total communication and synchronization overheads.

- **Dynamic Application Configuration Support:** Application adaptation results in application grids being dynamically created, moved and deleted on-the-fly, making it necessary to efficiently re-partition the hierarchy so that it continues to meet these goals. Furthermore, Grid computing environments require selecting and configuring application components based on available resources.

### 2.3 Application and System Heterogeneity in SAMR

Unlike static applications where requirements are typically known a priori, the dynamic behavior of SAMR applications is based on the current state of the physical phenomenon being simulated and can only be determined at runtime. Thus, it is important to abstract the state of the SAMR application in order to determine its current computational, communication, and storage requirements. This information can then be used to determine an appropriate decomposition of the application and mapping of the computations to available processing elements of the computational environment, and to drive the selection of appropriate algorithms and implementations, both at the application level (solvers, preconditioners) as well as the system level (communication mechanism).

Furthermore, networked computational environments such as the computational Grid are highly dynamic in nature. Thus, it is imperative that the application management system be able to react to this dynamism and make runtime decisions to ensure that the application's requirements are satisfied and its performance optimized. These decisions include selecting the appropriate number, type, and configuration of the computing elements, appropriate distribution and load-balancing schemes, the most efficient communication mechanism, as well as the right algorithms and parameters at the application level.

However, the dynamism and heterogeneity of Grid environments introduces a new level of complexity and makes the selection of a "best" match between system resources, application algorithms, problem decompositions, mappings and load distributions, communication mechanisms, etc., non-trivial. System dynamics coupled with application adaptation makes application composition, runtime management, optimization, and adaptation a significant challenge.

## 3 GridARM: Architectural Overview

The overall goal of GridARM autonomic runtime framework is to reactively and proactively manage and optimize SAMR application execution using current sys-

tem and application state, online predictive models for system behavior and application performance, and an agent based control network. It builds on the concept of *vGrid* proposed by M. Parashar and S. Hariri (Khargharia 2003). The GridARM architecture provides application developers with a convenient abstraction of a virtual Grid that may be significantly larger and more reliable than currently available resources. The autonomic runtime framework manages physical Grid resources, allocates them "on-demand", and spatially and temporally maps the virtual resources to these physical nodes. The mapping exploits the space, time, and functional heterogeneity of the simulations and underlying numerical methods to define application "working-sets". GridARM infrastructure services are responsible for collecting and characterizing the operational, functional, and control aspects of the application and using this information to define autonomic components, decomposing the application into *natural regions* (NRs) and the NR into *virtual computational units* (VCUs), and applying innovative allocation and scheduling strategies to map VCUs to physical Grid resources. Together, these solutions will allow application developers to concentrate on the science and its formulations without having to worry about explicitly addressing the number, limitations, and availability of resources or targeting and tuning their implementations to specific architectures and machines.

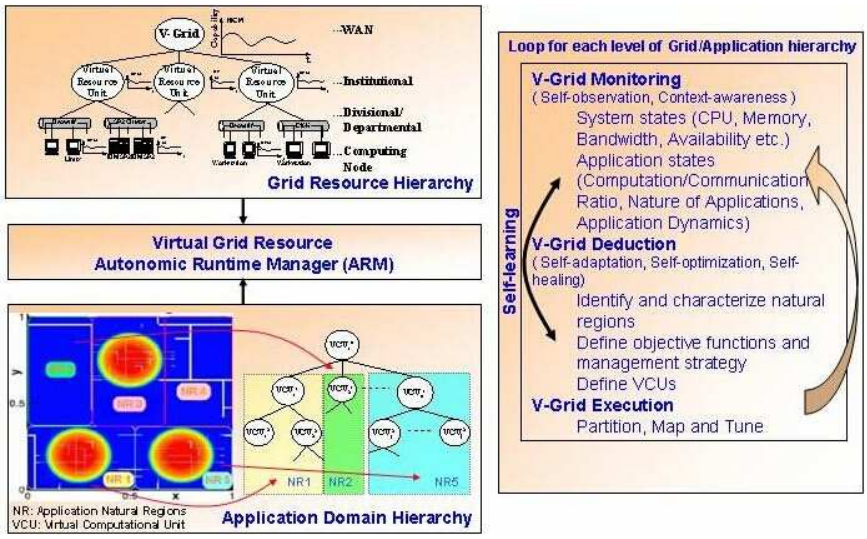


Figure 3: Conceptual model of GridARM framework

The conceptual GridARM architecture is shown in Figure 3. The framework has three components: (1) services for monitoring Grid resource capabilities and application dynamics and characterizing the monitored state into natural regions; (2) deduction engine and objective function that define the appropriate optimization strategy based on runtime state and policies; and (3) autonomic runtime manager which is responsible for hierarchically partitioning, scheduling, and mapping VCUs onto VRUs, and tuning application execution within the Grid environment.

# 4 GridARM: Operation

## 4.1 Monitoring and Characterization

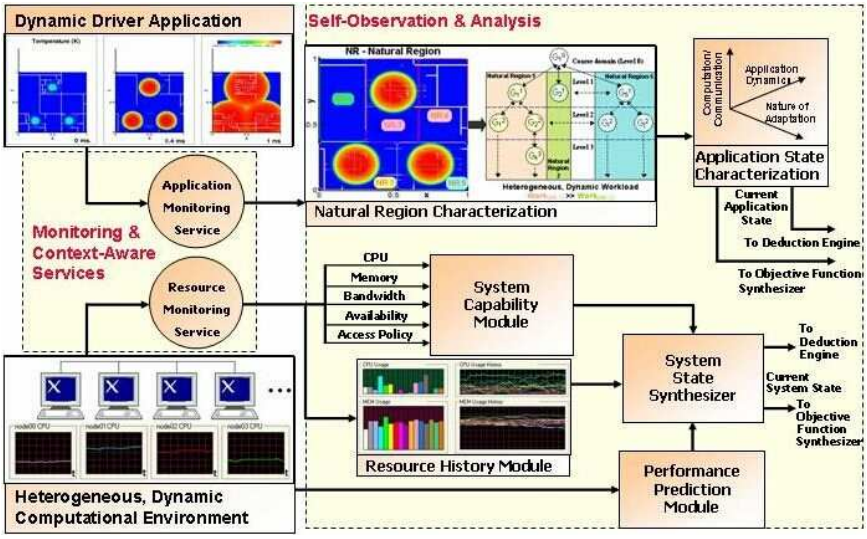


Figure 4: GridARM operation: monitoring and characterization

The monitoring and characterization mechanisms in the GridARM framework consist of embedded application-level and system-level sensors/actuators and are illustrated in Figure 4. The application is characterized into "natural regions" (NRs) which are regions of relatively homogeneous activity in the application domain and can span various levels of the SAMR grid hierarchy. Application sensors monitor the structure and state of the SAMR grid hierarchy and the nature of the refined regions. One way to track such natural regions for SAMR applications is using the refinement patterns based on local truncation errors. For example, a "hot-spot" in the flame simulation application described in Section 2 represents a natural region in the application domain. The application state is abstracted using these natural regions and is characterized in terms of application-level metrics such as computation/communication requirements, storage requirements, activity dynamics, and the nature of adaptations (Steensland 2002).

Similarly, system sensors, built on existing infrastructures such as NWS (Network Weather Service) and MDS (Metacomputing Directory Service), sense the current state of underlying computational resources in terms of CPU, memory, bandwidth, availability, and access capabilities. These are fed into the system state synthesizer along with history information (current state stored over time in the history module) and performance estimates (obtained using performance functions from the prediction module) to determine the overall system runtime state. The current application and system state are provided as inputs to the deduction engine and are used to define the autonomic runtime objective function.

## 4.2 Deduction and Objective Function

The deduction engine and the autonomic runtime manager provide the primary decision making capabilities within the GridARM framework. As shown in Figure 5, the current application and system state and the overall "decision space" are the inputs to the deduction engine. The decision space comprises the adaptation policies, rules, and constraints defined in terms of application metrics, and enables autonomic configuration, adaptation, and optimization. Application metrics include application locality, communication mechanism, data migration, load balancing, memory requirements/constraints, adaptive partitioning, adaptation overheads, and granularity control. Based on current runtime state and policies/-constraints within the decision space, the deduction engine formulates prescriptions for algorithms, configurations, and parameters that are used to define the objective function for adapting the behavior of the SAMR application. The deduction engine may be capable of self-learning by augmenting its decision space with new rules and constraints. The prescriptions provided by the deduction engine along with the objective function yield two metric – normalized work metric (NWM) and normalized resource metric (NRM) that characterize the current application state and current system state, respectively. These metric are self-defined based on the current application/system context and enable autonomic runtime management by helping to configure the SAMR application with appropriate parameters and execute optimally within the heterogeneous Grid environment.

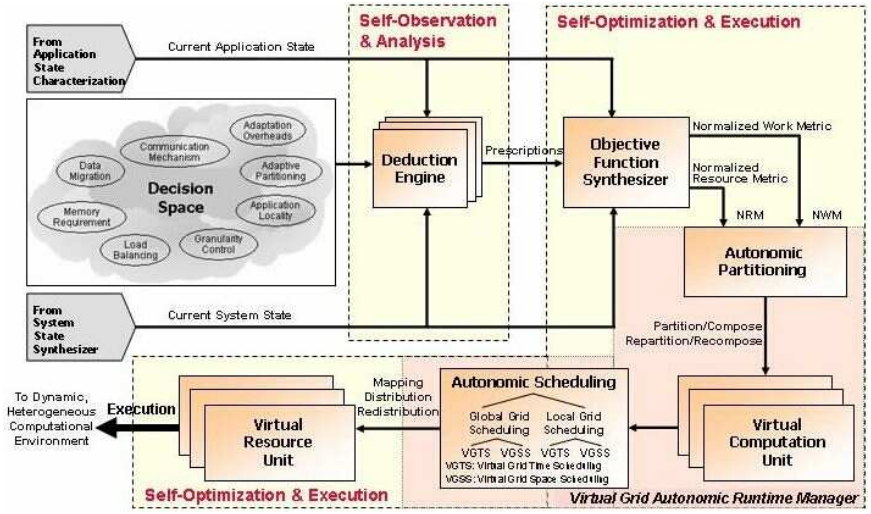


Figure 5: GridARM operation: deduction and optimization

## 4.3 Autonomic Runtime Manager

The normalized metric, NWM and NRM, form the inputs to the autonomic runtime manager (ARM). Using these inputs, ARM defines a hierarchical distribution mechanism, configures and deploys appropriate partitioners at each level of the hierarchy, and maps the application domain onto virtual computational units. A vir-

tual computational unit (VCU) is the basic application work unit that is scheduled by the GridARM framework and may consist of computation patches on a single refinement level of the SAMR grid hierarchy or composite patches that span multiple refinement levels. VCUs are dynamically defined at runtime to match the natural regions (NRs) in the application. Using natural regions to define VCUs can significantly reduce coupling and synchronization costs.

Subsequent to partitioning, scheduling operations on the virtual Grid are performed first across VRUs (Global-Grid Scheduling (GGS)) and then within a VRU (Local-Grid Scheduling (LGS)). During GGS, VCUs are hierarchically assigned to sets of VRUs, whereas LGS is used to schedule one or more VCU within a single VRU. The entire process is first spatial and then temporal, and combines a range of partition techniques (domain-based, patch-based, tree-based, etc.) and scheduling techniques (gang, backfilling, migration, etc.). A virtual resource unit (VRU) may be an individual resource (compute, storage, instrument, etc.) or a collection (cluster, supercomputer, etc.) of physical Grid resources. A VRU is characterized by its computational, memory, and communication capacities and by its availability and access policy. Finally, the VRUs are dynamically mapped onto physical system resources at runtime and the SAMR application is tuned for execution within the dynamic Grid environment.

Note that the work associated with a VCU depends on the state of the computation, the configuration of the components (algorithms, parameters), and the current ARM objectives (optimize performance, minimize resource requirements, etc.). Similarly, the capability of a VRU depends on its current state as well as the ARM objectives (minimizing communication overheads implies a VRU with high bandwidth and low latency has higher capability). The normalized metric NWM and NRM are used to characterize VRUs and VCUs based on current ARM objectives.

# 5 Current Status

The GridARM framework is currently under development at The Applied Software Systems Laboratory (TASSL) at Rutgers University, with current efforts focussed on the design, implementation, and evaluation of the core building blocks of the framework. The application used in the experimental evaluation of the GridARM prototype components is the 3-D Richtmyer-Meshkov instability solver (RM3D[1]) encountered in compressible fluid dynamics.

Application aware partitioning (Chandra 2002) uses current runtime state to characterize the SAMR application in terms of computation/communication, application dynamics, and the nature of adaptations. This adaptive strategy selects and configures the appropriate partitioner that matches current application requirements, thus improving overall execution time by 5-30% as compared to non-adaptive partitioning schemes. The adaptive hierarchical partitioning scheme (Li 2003) dynamically creates a group topology based on SAMR natural regions

[1] RM3D has been developed by Ravi Samtaney as part of the virtual test facility at the Caltech ASCI/ASAP Center.

and helps to reduce the synchronization costs needed to maintain the global hierarchy state, resulting in improved application communication time by up to 70% as compared to non-hierarchical schemes. System sensitive partitioning (Sinha 2001) uses current system state obtained using NWS to select and tune distribution parameters by dynamically partitioning and load balancing the SAMR grid hierarchy based on the relative capacities for each processor. In contrast to the non-heterogeneous scheme, the system sensitive approach improves overall execution time by 10-40%.

In addition, various optimizations have been incorporated within the GridARM framework that aim to improve SAMR application runtime parameters. Architecture sensitive communication mechanisms select appropriate messaging schemes that are suited for the underlying hardware architecture and help to improve application communication time by up to 50%. The workload sensitive load balancing strategy uses binpacking-based partitioning to distribute the SAMR workload among available processors while satisfying application constraints such as minimum patch size and aspect ratio. This approach reduces application load imbalance to 2-15% as compared to default schemes that employ greedy algorithms. Furthermore, performance prediction using performance functions can be used to estimate the application execution time based on current loads, available communication bandwidth, current latencies, and available memory. This approach helps to determine when the costs of dynamic load redistribution exceed the costs of repartitioning and data movement, and can result in 25% improvement in the application recompose time.

# 6 Conclusions and Future Work

This paper presented the design of the GridARM autonomic runtime management framework that monitors application and system state and provides appropriate distribution, configuration, scheduling, and adaptation strategies to optimize the performance of SAMR applications. The overall goal of the GridARM framework is to manage dynamism and space-time heterogeneity and support the efficient and scalable execution of SAMR applications in Grid environments. Future work in this research aims to integrate all autonomic components (currently developed and evaluated as independent building blocks) within the GridARM framework using a component-based architecture with an intelligent deduction engine to achieve highly flexible and orchestrated self-adapting, self-optimizing performance and self-learning behavior for Grid-based SAMR implementations.

# Acknowledgements

The authors would like to thank Xiaolin Li and Taher Saif for collaboration and valuable research discussions, and Ravi Samtaney for making the applications available for use.

# References


Berger, M., Hedstrom, G., Oliger, J., and Rodrigue, G. (1983), "Adaptive Mesh Refinement for 1-Dimensional Gas Dynamics", In *Scientific Computing* (IMACS/North Holland Publishing), pp. 43-47.

Berger, M. and Oliger, J. (1984), "Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations", In *Journal of Computational Physics*, Vol. 53, pp. 484-512.

Bryan, G. (1999), "Fluids in the Universe: Adaptive Mesh Refinement in Cosmology", In *Computing in Science & Engineering*, pp. 46-53.

Chandra, S. and Parashar, M. (2002), "ARMaDA: An Adaptive Application-sensitive Partitioning Framework for Structured Adaptive Mesh Refinement Applications", In *IASTED International Conference on Parallel and Distributed Computing Systems (PDCS '02)*, pp. 446-451, Cambridge, MA.

Foster, I. and Kesselman, C. (1998), "Computational Grids", In *The Grid: Blueprint for a New Computing Infrastructure*, editors: I. Foster and C. Kesselman, Morgan Kaufmann Publishers.

Kephart, J. and Chess, D. (2003), "The Vision of Autonomic Computing", In *IEEE Computer*, Vol. 36, No. 1, pp. 41-50.

Khargharia, B., Hariri, S., and Parashar, M. (2003), "vGrid: A Framework for Building Autonomic Applications", In *1st International Workshop on Heterogeneous and Adaptive Computing - Challenges of Large Applications in Distributed Environments (CLADE '03)*.

Li, X. and Parashar, M. (2003), "Dynamic Load Partitioning Strategies for Managing Data of Space and Time Heterogeneity in Parallel SAMR Applications", In *9th International Euro-Par Conference (Euro-Par '03)*, Austria.

Parashar, M., Wheeler, J., Pope, G., Wang, K., and Wang, P. (1997), "A New Generation EOS Compositional Reservoir Simulator: Part II - Framework and Multiprocessing", In *Society of Petroleum Engineering Reservoir Simulation Symposium*, Dallas, TX.

Ray, J., Najm, H., Milne, R., Devine, K., and Kempka, S. (2003), "Triple Flame Structure and Dynamics at the Stabilization Point of an Unsteady Lifted Jet Diffusion Flame", To be published in *Proc. Combust. Inst.*

Sinha, S. and Parashar, M. (2001), "Adaptive Runtime Partitioning of AMR Applications on Heterogeneous Clusters", In *3rd IEEE International Conference on Cluster Computing*.

Steensland, J., Chandra, S., and Parashar, M. (2002), "An Application-centric Characterization of Domain-based SFC Partitioners for Parallel SAMR", In *IEEE Transactions on Parallel and Distributed Systems*, Vol. 13, No. 12, pp. 1275-1289.