Chapter 1

Introduction: Enabling Large-Scale Computational Science—Motivations, Requirements, and Challenges

Manish Parashar and Xiaolin Li

1.1 MOTIVATION

The exponential growth in computing, networking, and storage technologies has ushered in unprecedented opportunities for parallel and distributed computing research and applications. In the meantime, emerging large-scale adaptive scientific and engineering applications are requiring an increasing amount of computing and storage resources to provide new insights into complex systems. Furthermore, dynamically adaptive techniques are being widely used to address the intrinsic heterogeneity and high dynamism of the phenomena modeled by these applications. Adaptive techniques have been applied to real-world applications in a variety of scientific and engineering disciplines, including computational fluid dynamics, subsurface and oil reservoir simulations, astronomy, relativity, and weather modeling. The increasing complexity, dynamism, and heterogeneity of these applications, coupled with similarly complex and heterogeneous parallel and distributed computing systems, have led to the development and deployment of advanced computational infrastructures that provide programming, execution, and runtime management support for such large-scale adaptive implementations. The objective of this book is to investigate the

Advanced Computational Infrastructures For Parallel and Distributed Adaptive Applications. Edited by Manish Parashar and Xiaolin Li

state of the art in the design, architectures, and implementations of such advanced computational infrastructures and the applications they support.

1.2 REQUIREMENTS AND CHALLENGES

Compared to numerical techniques based on static uniform discretization, adaptive techniques, for example, structured adaptive mesh refinement (SAMR), can yield highly advantageous ratios for cost/accuracy by adaptively concentrating computational effort and resources on appropriate regions of the domain at runtime. Large-scale parallel implementations of such adaptive applications have the potential for accurately modeling complex physical phenomena and providing dramatic insights. However, due to the dynamism and space–time heterogeneity, the scalable parallel implementation remains a significant challenge. Specifically, these adaptive applications are inherently dynamic because the physical phenomena being modeled and the corresponding adaptive computational domain change as the simulation evolves. Further, adaptation naturally leads to a computational domain that is spatially heterogeneous, that is, different regions in the computational domain and different levels of refinements have different computational and communication requirements. Finally, the adaptive algorithms require periodically regridding the computational domain, which causes regions of refinement to be created/deleted/moved to match the physics being modeled, that is, it exhibits temporal heterogeneity.

For example, parallel implementations of SAMR applications typically partition the adaptive grid hierarchy across available processors, and each processor operates on its local portions of this domain in parallel. The overall performance of parallel SAMR applications is thus limited by the ability to partition the underlying grid hierarchies at runtime to expose all inherent parallelism, minimize communication and synchronization overheads, and balance load.

Furthermore, communication overheads of parallel SAMR applications primarily consist of four components: (1) *interlevel communications*, defined between component grids at different levels of the grid hierarchy and consist of prolongations (coarse to fine transfer and interpolation) and restrictions (fine to coarse transfer and interpolation); (2) *intralevel communications*, required to update the grid elements along the boundaries of local portions of a distributed component grid, consisting of near-neighbor exchanges; (3) *synchronization cost*, which occurs when the load is not balanced among processors; and (4) *data migration cost*, which occurs between two successive regridding and remapping steps.

1.3 A TAXONOMY FOR ADAPTIVE APPLICATIONS AND MANAGEMENT STRATEGIES

To provide an overview of the applications and management strategies presented in this book, we develop a simple taxonomy that is based on the characteristics of the applications and application-level partitioners. This taxonomy, shown in Tables 1.1 and 1.2, decouples the classification of applications and their partitioners. This not

Table 1.1 Classification of Application Characteristics

Application characteristics	Categories
Execution	Computation intensive, communication intensive, IO intensive
Activity	Dynamic (localized adaptivity, scattered adaptivity), static
Granularity	Fine grained, coarse grained, indivisible
Dependency	Independent, workflow, hybrid

only provides a better understanding of both aspects but also implicitly indicates that different partitioners can be potentially applied to address different application characteristics. The SAMR application is used to illustrate the taxonomy in the discussion below.

In Table 1.1, applications are characterized based on their execution behavior, activity, granularity, and dependency. The execution behavior can be computation intensive, communication intensive, or IO intensive [1, 2]. For example, most SAMR applications are computation intensive, belonging to high-performance scientific computing category. Due to deep levels of adaptive refinements, SAMR applications can also be communication intensive. In some cases, when dealing with large amounts of data, SAMR applications can fall into IO-intensive category. Experiments show that during the entire course of execution, SAMR applications may run in different execution modes as the simulated physical phenomena evolve [1, 2]. Application activities are classified as dynamic or static. Many embarrassingly parallel applications belong to the static application category, for example, parallel geometrical transformations of images and Monte Carlo simulations [3]. SAMR applications, on the other hand, are dynamic in nature because of their dynamic adaptivity. The dynamic behavior of SAMR applications may demonstrate localized adaptivity pattern or scattered adaptivity pattern in different execution phases. From the perspective of divisibility, some applications are fine grained [4], some are coarse grained [5], while others are not divisible at all [6, 7]. Workloads in the divisible load scheduling class are assumed to be homogeneous and arbitrarily divisible in the sense that each portion of the load can be independently processed on any processor on the network. Coarse-grained divisible applications typically involve dependencies among subtasks. Indivisible tasks are atomic and cannot be further divided into smaller subtasks and have to be completely processed on a single processor. SAMR applications can fall into the fine-grained or coarse-grained divisible categories. When the underlying physical domain being modeled exhibits more homogeneity, the load associated with this domain belongs to the fine-grained divisible category. However, when the physical domain exhibits scattered heterogeneity with deep refinements, the load may be classified as coarse-grained divisible. Note that SAMR applications involve iterative operations and frequent communications and obviously cannot be considered as embarrassingly parallel. The last criterion in the table is dependency. Independent applications are common in the divisible load scheduling category, such as parallel low-level image processing and distributed database query processing [4]. Workflow

Table 1.2 Classification of Application Partitioners

Partitioner characteristics	Categories
Organization	Static single partitioner, adaptive single partitioner (meta-partitioner), adaptive hierarchical multiple partitioner
Decomposition	Data decomposition (domain based, patch based, hybrid), functional decomposition
Partitioning method	Binary dissection, multilevel, SFC based, and others
Operations	Dynamic Repartitioning policy (periodic, event driven), system sensitive
	Static

applications are composed of several modules or subtasks that must run in order, for example, data-driven parallel applications, scientific simulations, and visualization applications. In the case of SAMR applications, although load partitions can be processed independently, they need communications iteratively. If dynamic load balancing strategies are adopted, repartitioning may result in load movement or process migration, thus exhibiting a hybrid dependency.

As shown in Table 1.2, application-level partitioners/schedulers are classified with respect to their organization, decomposition method, and operations. The organization of partitioners falls into three categories: static single partitioner, adaptive single partitioner, and adaptive multiple partitioner. In the case of static single partitioning approaches, one predefined partitioning and repartitioning strategy is adopted throughout the entire life cycle of the applications. However, in the case of adaptive single partitioner approaches, also termed meta-partitioner in the literature [2], the most appropriate partitioning routine is selected based on the runtime behavior of the applications. Adaptive multiple partitioner approaches not only select appropriate partitioning strategies based on the runtime state of the application but also apply multiple partitioners simultaneously to local relatively homogeneous regions of the domain based on the local requirements.

Decomposition methods can be classified as data decomposition and functional decomposition. Functional decomposition exploits functional parallelism by dividing problems into a series of subtasks. Pipelining techniques can often be used to speed up applications with limited functional parallelism. Data decomposition is commonly applied to achieve data parallelism for applications that require the same operations to be performed on different data elements, for example, SPMD (single program multiple data) applications.

In the case of SAMR applications, data decomposition methods can be further classified as patch based and domain based. Patch-based decomposition methods make partitioning decisions for each patch at different refinement levels independently [8–10]. Patch-based techniques result in well-balanced load distribution as long as each patch is sufficiently large. Since the workload is balanced for each patch, an implicit feature of this scheme is that it does not need redistribution when a patch

is deleted at runtime. However, the scheme does not preserve locality and can result in considerable communication overheads. Moreover, these communications may lead to serializing of the computation and severe performance bottleneck. Interlevel communications occur during restriction and prolongation data transfer operations between parent–children (coarser–finer) patches.

In contrast, domain-based approaches partition the physical domain rather than the individual patches at each refinement level [9, 10]. A subdomain includes all patches on all refinement levels in that region. Domain-based schemes maintain the parent–child locality while striving to balance overall workload distribution. As a result, these techniques substantially eliminate interlevel communication overheads and the associated communication bottleneck. However, strict domain-based partitioning may result in a considerable load imbalance when the application has deep refinements on very narrow regions with strongly localized features. In these cases, hybrid schemes that combine patch-based and domain-based techniques are required.

1.4 A CONCEPTUAL ARCHITECTURE FOR COMPUTATIONAL INFRASTRUCTURES

The functionality of a computational infrastructure includes partitioning applications at runtime and distributing each partition to available resources to efficiently utilize resources and minimize the execution time of applications. Figure 1.1 presents an illustrative conceptual architecture for such an infrastructure as a middleware system

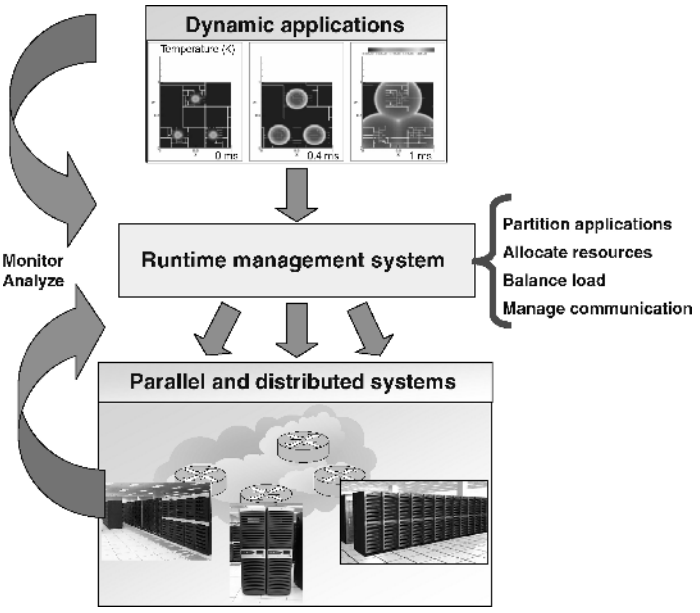

Figure 1.1 An illustrative conceptual architecture for computational infrastructures.

that bridges the gap between applications and operating systems and provides efficient runtime support, including monitoring, dynamic partitioning, resource allocation, coordination, load balancing, and runtime adaptation. In the figure, the illustrative input application is a combustion simulation of hydrogen–air mixture with three initial ignition spots. The infrastructure monitors the runtime states of applications and resources, analyzes requirements, and delivers a plan for partitioning, scheduling, and executing the application on parallel and distributed systems to maximize their performance, efficiency, and utilization. In case of such dynamically adaptive applications, the system should also address the dynamism and space–time heterogeneity in applications.

1.5 OVERVIEW OF THE BOOK

The objective of this book is to investigate the state of the art in the design, architectures, and implementations of such advanced computational infrastructures and the applications they support. The book is organized in three separate parts. Part I, Adaptive Applications in Science and Engineering, focuses on high-performance adaptive scientific applications and includes chapters that describe high-impact, real-world application scenarios. The goal of this part of the book is to motivate the need for advanced computational engines as well as outline their requirements. Part II, Adaptive Computational Infrastructures, includes chapters describing popular and widely used adaptive computational infrastructures. Part III, Dynamic Partitioning and Adaptive Runtime Management Frameworks, focuses on the more specific partitioning and runtime management schemes underlying these computational toolkits.

We do hope that it will lead to new insights into the underlying concepts and issues, current approaches and research efforts, and outstanding challenges of the field and will inspire further research in this promising area.

REFERENCES

1. 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)*, 2001.
2. J. Steensland, M. Thune, S. Chandra, and M. Parashar. Towards an adaptive meta-partitioner for parallel SAMR applications. In *IASTED PDCS 2000*, 2000.
3. B. Wilkinson and M. Allen. *Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers*, 1st edition. Pearson Education, 1999.
4. B. Veeravalli, D. Ghose, V. Mani, and T.G. Robertazzi. *Scheduling Divisible Loads in Parallel and Distributed Systems*. IEEE Computer Society Press, Los Alamitos, CA, 1996.
5. G. Allen, T. Dramlitsch, I. Foster, N.T. Karonis, M. Ripeanu, E. Seidel, and B. Toonen. Supporting efficient execution in heterogeneous distributed computing environments with cactus and globus. In *Proceedings of the 2001 ACM/IEEE Conference on Supercomputing (CDROM)*, Denver, CO, 2001, p. 52.
6. S. Bokhari. *Assignment Problems in Parallel and Distributed Computing*. Kluwer Academic Publishers, Boston, MA, 1987.

7. B.A. Shirazi, A.R. Hurson, and K.M. Kavi. *Scheduling and Load Balancing in Parallel and Distributed Systems*. IEEE Computer Society Press, Los Alamitos, CA, 1995.
8. S. Kohn. SAMRAI: structured adaptive mesh refinement applications infrastructure. Technical report, Lawrence Livermore National Laboratory, 1999.
9. M. Parashar and J. Browne. On partitioning dynamic adaptive grid hierarchies. In *29th Annual Hawaii International Conference on System Sciences*, 1996, pp. 604–613.
10. J. Steensland. *Efficient partitioning of structured dynamic grid hierarchies*. PhD thesis, Uppsala University, 2002.