An Experimental Study of Adaptive Application-Sensitive Partitioning Strategies for SAMR*Applications†

Sumir Chandra, Johan Steensland & Manish Parashar
The Applied Software Systems Laboratory
Department of Electrical and Computer Engineering
Rutgers, The State University of New Jersey
94 Brett Road, Piscataway, NJ 08854 USA
{sumir, johans, parashar}@caip.rutgers.edu

Dr. Julian C. Cummings,
Center for Advanced Computing Research
California Institute of Technology
1200 E. California Blvd. Pasadena, CA 91125
cummings@cacr.caltech.edu

Abstract

Dynamic adaptive mesh refinement methods for the numerical solution to partial differential equations yield highly advantageous ratios for cost/accuracy when compared to methods based upon static uniform approximations. Parallel/distributed implementations of these techniques, however, present significant challenges in dynamic data-distribution and load-balancing. This is because the choice of the "best" partitioning technique and associated partitioning parameters depends on the nature of the application and its run-time state. This paper presents an experimental study of an adaptive application-sensitive meta-partitioner for structured adaptive mesh refinement (SAMR) applications that dynamically selects and configures partitioning strategies at run-time based on system parameters and the current application state. The selection is based on a run-time classification of application state and an application-centric classification of the partitioners. Experimental results show that adaptive partitioning can improve application performance.

Keywords: Adaptive partitioning; Run-time management; Dynamic applications; Performance characterization; Structured adaptive mesh refinement.

1 Introduction

This paper presents an experimental study of an *adaptive* application-sensitive meta-partitioner for adaptive applications that improves application performance by dynamically selecting and configuring partitioners at run-time based on system parameters and the current application state. The

*Structured Adaptive Mesh-Refinement

†The work presented here was supported by the National Science Foundation via grants numbers ACI 9984357 (CAREERS), WU-HT-99-19 P029786F (KDI) and by DOE ASCI/ASAP (Caltech) via grant number PC295251, awarded to Manish Parashar.

overall goal of this project is to develop active system and application sensitive mechanisms to manage and optimize dynamically adaptive applications in distributed, heterogeneous and dynamic execution environments such as the computational "grid" [19].

Dynamic adaptive mesh refinement (AMR) methods [16], employing locally optimal approximations to partial differential equations, can yield highly advantageous ratios for cost/accuracy when compared to methods based upon static uniform approximations. These techniques seek to improve the accuracy of the solution by dynamically allocating additional computational resources to regions with large local solution error. Distributed implementations of these techniques offer the potential for realistic simulations of complex physical phenomena. These implementations, however, lead to interesting challenges in data-distribution, load-balancing, and resource management. A critical challenge in structured AMR (SAMR) is the partitioning of the adaptive grid hierarchy to balance load, optimize the communication pattern, minimize synchronization and data migration cost, maximize grid quality (e.g. aspect ratio), and exploit all available parallelism.

The research presented in this paper is motivated by the observation that for parallel/distributed SAMR, *no single partitioning scheme performs the best* for all types of applications and systems. Even for a single application, the most suitable partitioning technique depends on input parameters and the application run-time state [9, 15]. As a result, it becomes necessary to actively manage these dynamic applications at run-time. This includes using application and system run-time state to select and configure the partitioning and load-balancing strategy to be used, so as to maximize performance. The theoretical foundation for the meta-partitioner was developed in earlier research, which focused on characterizing the state of SAMR applications, identifying the partitioning requirements for different application/system states, and characterizing partitioning techniques [13] based on their ability to satisfy these requirements. The work provides a basis for dynamically mapping partitioners to application states. The goal of this paper is to build on this foundation to formulate strategies for dynamically selecting and configuring partitioners at run-time, and experimentally demonstrate that adaptive partitioning can improve application performance.

In this paper we use the structure of the adaptive grid hierarchy to characterize its current state and determine its current partitioning requirements. These requirements are then used by the meta-partitioner to select and configure the partitioning techniques to be used. The meta-partitioner is experimentally evaluated using a 3-D compressible turbulence application kernel, solving the Richtmyer-Meshkov instability. This application is a part of the ASCI/ASAP effort at California Institute of Technology. The partitioners used in this experiment are a pick from popular software libraries such as GrACE [5] and Vampire [14]. Experiments were performed on the IBM SP2 "Blue Horizon" system at the San Diego Supercomputing Center (SDSC). Experimental results show that adaptive partitioning can improve application performance.

The rest of the paper is organized as follows. Section 2 presents a brief overview of the SAMR methods and structured grid hierarchies. Section 3 describes related work in dynamic partitioning and load balancing. Section 4 presents an outline of the adaptive meta-partitioner. Section 5 describes the experimental study of the adaptive partitioner and presents the experimental results. Section 6 presents some concluding remarks and outlines future work.

2 AMR and Structured Grid Hierarchies

Dynamically adaptive solution techniques for partial differential equations provide a means for concentrating additional grid resolution and computational resources to regions in the application do-

main with large error. These techniques potentially lead to more efficient and cost-effective solutions to time-dependent problems exhibiting localized features. In the case of structured adaptive mesh refinement (SAMR) methods, dynamic adaptation is achieved by tracking regions in the domain that require higher resolution and dynamically overlaying finer grids on 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 with high solution error, requiring additional resolution, are identified and refined. Refinement proceeds recursively so that the refined regions requiring more resolution are similarly tagged and even finer grids are overlaid on these regions. The resulting grid structure is a dynamic adaptive grid hierarchy. The adaptive grid hierarchy corresponding to the SAMR formulation by Berger and Oliger [1] is shown in Fig. 1.

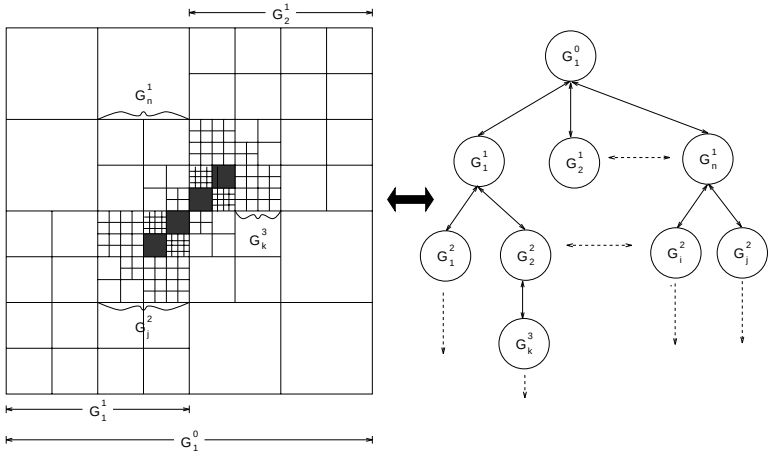

Figure 1: Berger-Oliger formulation of adaptive grid hierarchies for SAMR applications

Distributed implementations of SAMR methods present interesting challenges in dynamic resource allocation, data-distribution, and load balancing. The overall efficiency of parallel/distributed SAMR applications is limited by the ability to partition the underlying grid hierarchies at run-time to expose all inherent parallelism, minimize communication and synchronization overheads, and balance load. A critical requirement while partitioning these 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. Furthermore, application adaptation results in application grids being created, moved and deleted "on the fly", making it necessary to efficiently re-partition the hierarchy "on the fly" so that it continues to meet these goals. A sequence of grid hierarchies for the Buckley-Leverette 2D SAMR application is shown in Fig. 2. In this figure, the refined grids (levels 1 through 4) are tracking the front as it moves across the diagonal of the grid from bottom left to top right.

3 Related Work

Research on partitioning/load-balancing techniques can be broadly divided into static and dynamic techniques. *Static partitioning* techniques are used when the application domain is partitioned only once (or very few times) and there is no dynamic redistribution involved. In this case, the initial

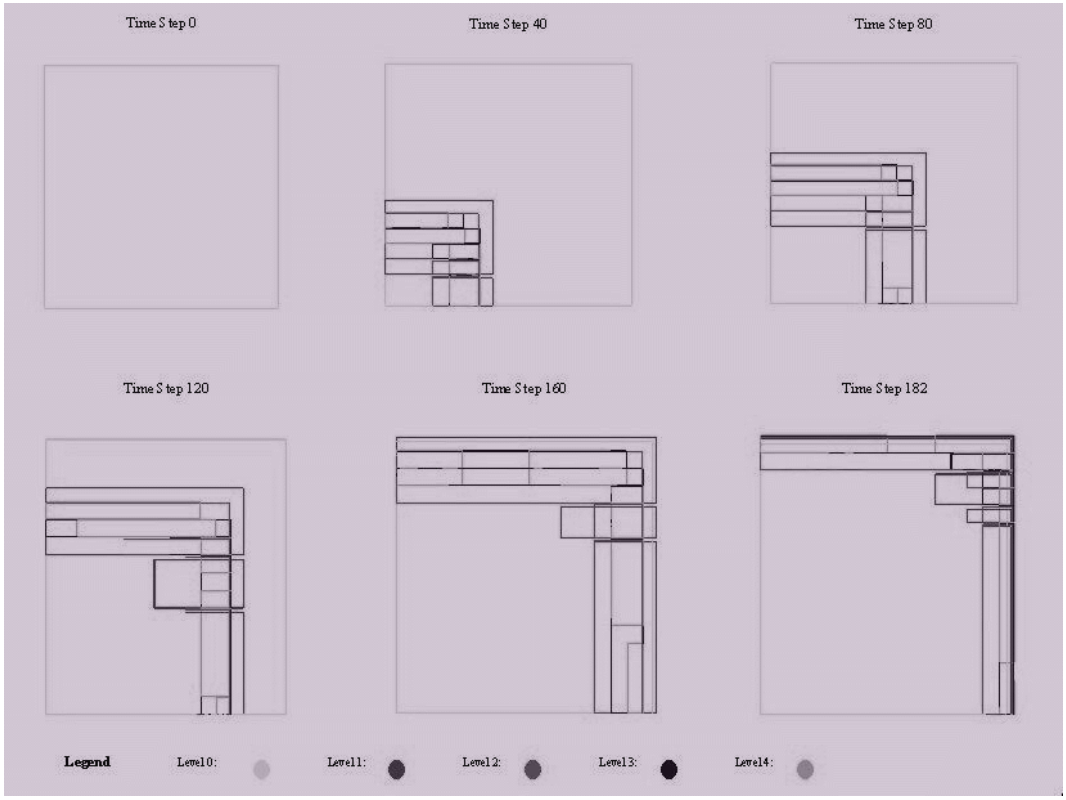

Figure 2: A sequence of 2-D adaptive grid hierarchies for a time-varying SAMR application

partitioning is maintained through the execution of the application. Static techniques tend to focus on the partitioning quality rather than partitioning speed. *Dynamic partitioning* routines are used by adaptive applications to repartition and redistribute the dynamic application domain at run-time. In addition, these techniques have to minimize data movement overheads and partitioning time, as grid adaptations occur at regular intervals. Consequently, partitioning quality is often sacrificed for speed.

Partitioners for SAMR applications can be classified as patch-based [12, 17, 9], domain-based, and hybrid partitioners. In the case of *patch-based partitioners*, distribution decisions are made independently for each newly created grid. A grid may be kept on the local processor or entirely moved to another processor. If the grid is considered too large, it may be split. *Domain-based partitioners* partition the physical domain, rather than the grids themselves. The domain is partitioned along with all contained grids on all refinement levels. *Hybrid partitioners* use a 2-step partitioning approach. The first step uses domain-based techniques to generate meta-partitions that are mapped to a group of processors. The second step uses a combination of domain and patch based techniques to optimize the distribution of each meta-partition within its processor group. The SAMR framework SAMRAI [20] (based on the LPARX [6] and KeLP [8] model) fully supports patch-based partitioning. However, domain-based partitioning techniques are preferable and tend to be more scalable [12, 17, 9, 15] (as long as they are applicable). The study presented in this paper involves domain-based techniques exclusively.

Partitioning of unstructured adaptive meshes and general graphs is a large field of research. Graph

partitioners, while establishing load balance, focus on either minimizing the communication, expressed as a edge-cut, [3, 10, 18] or minimizing the data redistribution cost [2, 4, 7]. In [11], a unified repartitioning algorithm is used along with a relative cost factor for dynamic load-balancing. Repartitioning is performed twice with alternate methods during the initial partitioning phase, and the scheme yielding the lowest cost is selected.

4 Adaptive Application-Sensitive Meta-Partitioner for SAMR Grid Hierarchies

The effective partitioning of dynamic SAMR grid hierarchies is a significant challenge. The SAMR partitioner has to dynamically partition the current hierarchy to expose all inherent parallelism, minimize data-movement overheads, minimize communication and synchronization overheads, and balance load. Furthermore, it must do this in a minimum amount of time. Clearly, optimizing all the above requirements leads to conflicting objectives. For example, optimizing load balance and communications together is an NP-hard problem. As a result, partitioners typically optimize a subset of the components at the expense of others.

The goal of the *adaptive* application-sensitive meta-partitioner is to reduce overall run-times of parallel SAMR applications by dynamically selecting and configuring partitioners at run-time to match the current state of the application. The structure of the adaptive grid hierarchy is used to characterize its current state and define its current partitioning requirements. These requirements are then used by the adaptive partitioner to appropriately select and configure the partitioning technique that best satisfies them. For example, consider a hypothetical SAMR application where the refined regions track a shock hitting a boundary and the refinements are localized before impact and scattered after impact. Initially, the refined regions may be relatively coarse-grained. In this state, the application run-times would be dominated by computations and the structure of the refinements would not change significantly from one refinement to the next. After impact, the refinements will be relatively fine-grained and scattered. In this state, the application will be dominated by communication costs and the structure of the refinements will be highly dynamic. In the former state, a partitioner that emphasizes load-balance is the most suitable. In the latter case, a partitioner that minimizes communication and data-migration costs will be appropriate.

The run-time partitioner selection is based on an application-centric characterization that defines the performance of a {*partitioner, application-state*} tuple using a 5-component metric. This metric allows the trade-offs for each partitioner to be readily identified. The components are (1) load-balance generated, (2) application communication requirements due to partitioning, (3) data migration requirements due to dynamic repartitioning, (4) time required for partitioning, and (5) partitioning quality. Partitioning quality is measured in terms of the number of sub-grid generated and their aspect ratios.

Application state is characterized using the *octant approach* shown in Fig. 3. In this approach, application state and computer system state are classified with respect to (a) the adaptation pattern (scattered or localized), and (b) whether run-time is dominated by computations or communications, and (c) the activity dynamics in the solution, e.g. adaptation pattern changes rapidly.

Applications may start off in one octant, then, as solution progresses, migrate to any other. For example, in the case of the example application discussed above where the refined regions track a shock hitting a boundary, the refinements may be localized before impact and scattered after impact. Note that a similar dynamic behavior may also be exhibited by the system. For example, load

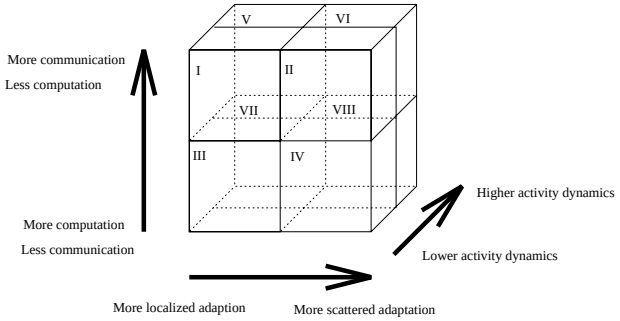

Figure 3: Octant approach for characterizing SAMR application state

variations on a shared system can cause a transition from one octant to another. As the network load increases, a previously computation-dominated application can become communication-dominated.

5 An Experimental Study of the Adaptive Application-Sensitive Meta-Partitioner

5.1 Characterizing Partitioner Behavior

This experimental study uses 3 partitioners, viz. space-filling curve-based partitioning (SFC), p-way binary dissection inverse space-filling curve partitioning (pBD-ISP), and geometric multi-level inverse space-filling curve partitioning with sequence partitioning (G-MISP+SP). These partitioners were selected from a suite of seven techniques as they were best suited for the application considered in the experiments [13]. The techniques used for evaluation are summarized in Table 1. The SFC partitioner is a part of Grid Adaptive Computational Engine (GrACE) [5] [21] while pBD-ISP and G-MISP+SP schemes are part of the Vampire [14] SAMR partitioning library.

Table 1: Summary of partitioning techniques

Scheme	Suite	Description
SFC	GrACE	Decomposition of the recursive linear representation of a multi-dimensional grid hierarchy, generated using space-filling mappings, to balance load across processors
G-MISP+SP	Vampire	Successive refinement of a one-vertex graph (matrix of workloads) by splitting vertices with total weight greater than a threshold, and using sequence partitioning to assign blocks to processors
pBD-ISP	Vampire	Generalized binary dissection of the domain into p partitions, dividing load as evenly as possible, with the orientation of the cuts determined by the Hilbert space-filling curve

SFC: The SFC scheme uses computationally efficient space-filling mappings to maintain logical locality and decomposes the one-dimensional representation of the grid hierarchy to balance load across processors. This decomposition results in average-to-good load balance at the expense of increased communication and data migration overheads. This scheme is suited for application states associated with low-to-moderate activity dynamics and more computation than communication.

G-MISP+SP: The G-MISP+SP scheme creates a one-dimensional list of similar-weight blocks as a result of the splitting of graph vertices obtained from the matrix representation of the grid hierarchy. It then uses sequence partitioning [1] to assign blocks to processors. G-MISP+SP is aimed towards speed and simplicity, and favors simple communication patterns and partitioning speed over amount of data migration. Sequence partitioning significantly improves the load balance but can be computationally expensive.

pBD-ISP: The pBD-ISP scheme partitions the domain into p partitions such that each split divides the load as evenly as possible, taking the available numbers of processors into account. This scheme is fast and the coarse granularity partitionings generated result in low overheads and communication costs. However, as the cuts are rectilinear, the overall load balance is average and may deteriorate when refinements are strongly localized. This scheme is suited for application states with greater communication and data requirements and lesser emphasis on load balance.

5.2 Characterizing Application State

The experimental evaluation of adaptive meta-partitioning uses RM3D, a 3-D compressible turbulence application solving the Richtmyer-Meshkov instability. RM3D is part of the virtual test facility developed at the ASCI/ASAP center at the California Institute of Technology[2]. The Richtmyer-Meshkov (RM) 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. Application characterization consists of two steps. First, the adaptive behavior of the application was captured in an adaptation trace generated using a single processor run. The adaptation trace contains snap-shots of the SAMR grid hierarchy at each regrid step. This trace was then analyzed using the octant approach and the adaptive partitioning strategy was defined. The application characterization presented in this paper was performed manually. However, we are currently developing agent-based mechanisms for automatically performing the characterization at run-time.

In this experiment we used a base grid of size 128*32*32 and 3 levels of factor 2 space-time refinements. Regriding was performed every 4 time-steps at each level. The application was run for 800 coarse level time steps and the trace consisted of over 200 snap-shots. A selection of these snap-shots are shown in Fig. 4 and illustrate the dynamics of the application.

Time-step	Octant State	Partitioner
0	IV	G-MISP+SP
5	VII	G-MISP+SP
25	I	pBD-ISP
106	VI	pBD-ISP
137	VIII	G-MISP+SP
162	II	pBD-ISP
174	V	pBD-ISP
201	III	G-MISP+SP

Table 2: Characterizing RM3D application run-time state for partitioning behavior

[1] Sequence partitioning is the problem to partition a sequence of n numbers into k intervals ($k < n$) by finding $k-1$ delimiters such that the sum of the numbers in the interval with the largest sum is minimized.

[2] http://www.cacr.caltech.edu/ASAP/

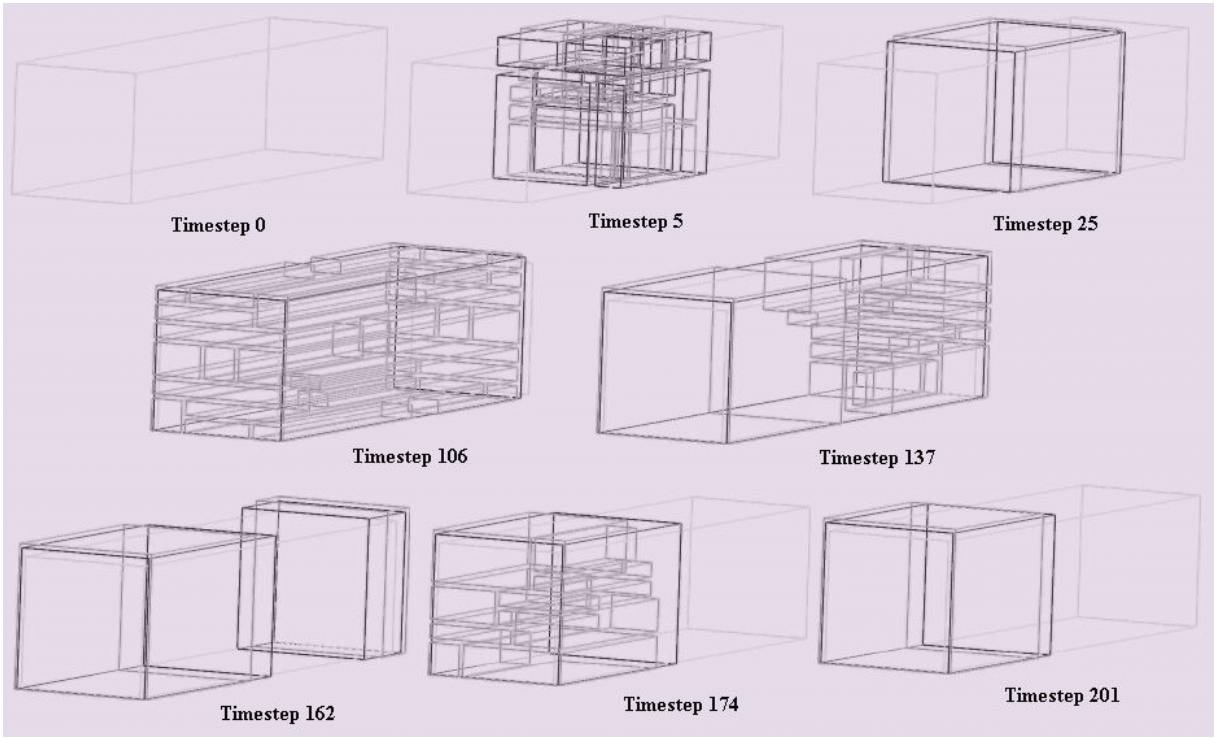

Figure 4: Sequence of profile views at sampled time-steps for RM3D application

The octant-based characterization of the RM3D trace is summarized in Table 2. RM3D starts out with scattered adaptation, lower activity dynamics and more computation, thus placing it initially in octant IV. G-MISP+SP is the most appropriate among the 3 partitioners for this octant and is selected. As the application evolves, its adaptation patterns cause it to move between octants and require different partitioning schemes. At time-step 106, the RM3D application has high communication requirements, high dynamics, and a scattered adaptation placing it in octant VI. pBD-ISP is the partitioner of choice in this octant. At the end of the simulation, the application adaptations are localized, with increased computation and lower activity, placing it in octant III with G-MISP+SP as the appropriate partitioner.

5.3 Experimental Evaluation of the Adaptive Meta-Partitioner

The experimental study of the adaptive meta-partitioner was performed on the NPACI IBM SP2, *Blue Horizon*, at the San Diego Supercomputing Center. Blue Horizon is a teraflop-scale Power3 based clustered SMP system from IBM. It supports shared-memory symmetric multi-processing (via OpenMP or PThreads) within a node and message passing (via MPI) between nodes. The machine contains 1,152 processors and 512 GB of main memory, arranged as 144 SMP compute nodes, with AIX as the operating system. Nodes are connected via a proprietary IBM communications network capable of a peak bi-directional data transfer rate of approximately 115 MBps.

The experiments consisted of measuring application execution times for different processor configurations using the meta-partitioning strategies outlined above. The partitioner as well as the partitioning parameters were switched on-the-fly while the application was executing using the re-

grid step as an indicator. Run-times from single partitioner runs were also measured.

Table 3: Performance of different partitioners for RM3D application on 64 processors

Partitioner	Run-time(s)	Max. Load Imbalance(%)	AMR Efficiency(%)
SFC	484.502	24.878	98.8207
G-MISP+SP	405.062	11.3178	98.7778
pBD-ISP	414.952	35.0317	98.8582
"adaptive"	352.824	8.11825	98.7633

Run-times for experiments on 32 and 64 processors are plotted in Figures 5 and 6 respectively. A sampling of the results for 64 processors are shown in Table 3. From these graphs, it can be seen that dynamically switching partitioners (and associated partitioning parameters) improves application performance and results in reduced run-times. For 64 processors, the improvement is 27.2% from the slowest partitioner. These results demonstrate that adaptive partitioning can improve application performance. In the adpative partitioner, G-MISP+SP is used to improve load-balance when the application is computationally dominated, while pBD-ISP reduces communication and data-migration overheads. This combination, based on run-time state, results in reduced partitioning time and lower load imbalance while maintaining high AMR efficiency. We are currently experimenting on larger processors and the results will be included in the final paper.

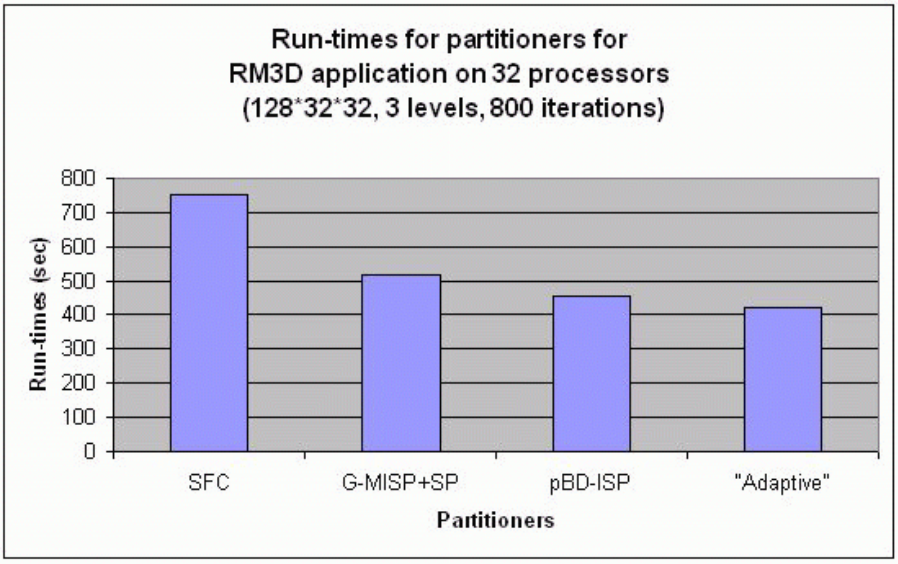

Figure 5: Run-times for partitioning techniques for RM3D application on 32 processors

6 Conclusions

This paper presented an experimental study of an adaptive application-sensitive meta-partitioner for structured adaptive mesh refinement (SAMR) applications. The meta-partitioner dynamically selects and configures partitioning strategies at run-time based on system parameters and the current

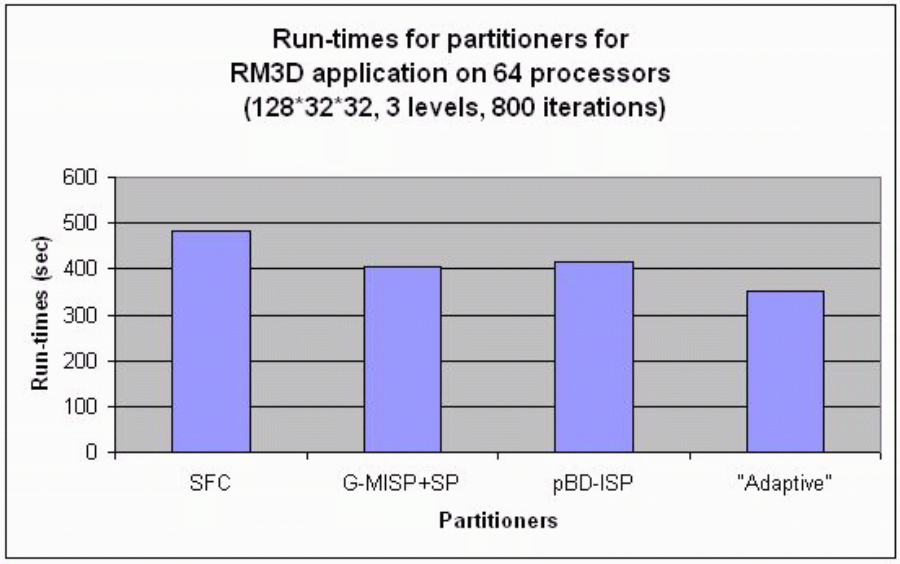

Figure 6: Run-times for partitioning techniques for RM3D application on 64 processors

application state. This research was motivated by the observation that the choice of the "best" partitioning technique and associated partitioning parameters depends on the nature of the application and its run-time state. In this paper, we used the structure of the adaptive grid hierarchy to characterize its current state and determine its current partitioning requirements. These requirements were then used by the meta-partitioner to select and configure the partitioning techniques to be used. The meta-partitioner was experimentally evaluated using a 3-D compressible turbulence application kernel, solving the Richtmyer-Meshkov instability. This application is a part of the ASCI/ASAP effort at California Institute of Technology. The partitioners used in this experiment are a part of the GrACE and Vampire libraries. Experiments were performed on the IBM SP2 "Blue Horizon" system at the San Diego Supercomputing Center (SDSC). Experimental results show that adaptive partitioning can improve application performance.

The overall goal of this project is to develop active system and application sensitive mechanisms to manage and optimize dynamically adaptive applications in distributed, heterogeneous and dynamic execution environments such as the computational "grid". We are currently working on the design of an automated recommender system that uses application and system state to select and configure partitioners. The results presented in this paper will be used to define the policies that will drive such a system.

Acknowledgments

The authors would like to thank Ravi Samtaney for making RM3D available for use in this research.

References

[1] Marsha J. Berger and Joseph Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. Journal of Computational Physics, 53, 1983.

[2] P. Diniz, S. Plimpton, B. Hendrickson, and R. Leland. Parallel algorithms for dynamically partitioning unstructured grids. Proc. 7th SIAM Conf. Parallel Proc., 1995.

[3] C. Ou and S. Ranka. Parallel incremental graph partitioning using linear programming. Proceedings Supercomputing '94, 458–467, 1994.

[4] C. Ou, S. Ranka, and G. Fox. Fast and parallel mapping algorithms for irregular and adaptive problems. Journal of Supercomputing, 10:119–140, 1996.

[5] M. Parashar, J.C. Browne, C. Edwards, and K. Klimkowski, A common data management infrastructure for adaptive algorithms for PDE solutions. Technical Paper at Supercomputing, 1997.

[6] S. Baden, S. Kohn, and S. Fink, "Programming with LPARX", Technical report, University of California, San Diego, 1994.

[7] J. Pilkington and S. Baden. Dynamic partitioning of non-uniform structured workloads with space-filling curves. Technical Report, Dept. of Computer Science and Engineering, Univ. of California, 1995.

[8] S. Fink, S. Baden, and S. Kohn, "Flexible communication mechanisms for dynamic structured applications", *IRREGULAR*, 1996.

[9] J. Rantakokko. Data Partitioning Methods & Parallel Block-Oriented PDE Solvers. PhD thesis, Uppsala Univ, 1998.

[10] K. Schloegel, G. Karypis, and V. Kumar. Wavefront diffusion and LMSR: Algorithms for dynamic repartitioning of adaptive meshes. Technical Report TR 98-034, Dept. of Computer Science and Engineering, University of Minnesota, 1998.

[11] K. Schloegel, G. Karypis, and V. Kumar, "A unified algorithm for load-balancing adaptive scientific simulations", *International Conference on Supercomputing*, 2000.

[12] M. Shee, S. Bhavsar, and M. Parashar. Characterizing the performance of dynamic distribution and load-balancing techniques for adaptive grid hierarchies. IASTED International Conference on PDCS, 1999.

[13] J. Steensland, S. Chandra, M. Thuné, and M. Parashar, "Characterization of domain-based partitioners for parallel SAMR applications", *IASTED International Conference on Parallel and Distributed Computing and Systems*, 2000.

[14] J. Steensland. http://www.tdb.uu.se/~johans/research/vampire/vampire1.html. Vampire homepage, 2000.

[15] J. Steensland, Domain-based partitioning for parallel SAMR applications, Licentiate thesis, 2001-002, Dept. of Scientific Computing, Uppsala University, Uppsala, Sweden, 2001.

[16] Erlendur Steinthorsson and David Modiano. Advanced methodology for simulation of complex flows using structured grid systems. ICOMP, 28, 1995.

[17] M. Thuné. Partitioning strategies for composite grids. Parallel Algorithms and Applications, 11:325–348, 1997.

[18] C. Walshaw, M. Cross, and M. G. Everett. Parallel dynamic graph partitioning for adaptive unstructured meshes. Journal of Parallel and Distributed Computing, 47(2):102–108, 1997.

[19] I. Foster and C. Kesselman The Grid: Blueprint for a Future Computing Infrastructure Morgan Kaufmann Publishers, 1998.

[20] R.D. Hornung, and S. Kohn, "SAMRAI: A Software Framework for Structured Adaptive Mesh Refinement," DOE Conference on High Speed Computing, Salishan Lodge, Gleneden Beach, OR, April 19-22, 1999. Also available as Lawrence Livermore National Laboratory technical report UCRL-MI-133555.

[21] M. Parashar and J. C. Browne, "System Engineering for High Performance Computing Software: The HDDA/DAGH Infrastructure for Implementation of Parallel Structured Adaptive Mesh Refinement", *in IMA Volume 117: Structured Adaptive Mesh Refinement Grid Methods, IMA Volumes in Mathematics and its Applications.* Springer-Verlag, pp. 1-18, 2000.