Characterization of Domain-Based Partitioners for Parallel SAMR Applications*

Johan Steensland†
Michael Thuné‡
IT, Dept of Scientific Computing
Uppsala University
Box 120, S-751 04 Uppsala
Sweden

Sumir Chandra§
Manish Parashar¶
Electrical and Computer Engineering
Rutgers, The State University of NJ
94 Brett Road, Piscataway, NJ 08854
USA

July 5, 2000

Abstract

Dynamic adaptive mesh refinement methods for the numerical solution to partial differential equations yield highly advantageous ratios for cost/accuracy as compared to methods based upon static uniform approximations. Distributed implementations of these techniques have the potential for enabling realistic simulations of complex systems. These implementations however, present significant challenges in dynamic data-distribution and load balancing. This paper presents an experimental characterization of dynamic partitioning/load-balancing techniques for adaptive grid hierarchies. Techniques studied include newly proposed as well as existing approaches. The overall motivation is the formulation of policies required to drive a *dynamically adaptive* meta-partitioner for SAMR grid hierarchies capable of selecting the most appropriate partitioning strategy at runtime, based on current application and system state. We believe that such a partitioner can significantly decrease application execution time.

Keywords: Parallel/distributed algorithms; Dynamic partitioning/load-balancing; Adaptive meta-partitioner; Performance characterization; Run-time adaptation; Structured adaptive mesh refinement.

1 Introduction

This paper presents an experimental characterization of dynamic domain-based partitioning and load-balancing techniques for distributed adaptive grid hierarchies that underlie parallel structured adaptive mesh refinement (SAMR) methods for the solution to partial differential equations. The overall goal of this research is to support the formulation of policies required to drive a *dynamically adaptive* meta-partitioner for SAMR grid hierarchies. We believe that such a meta-partitioner will be capable of decreasing overall execution time by selecting the most appropriate partitioning strategy (from a family of available partitioners) at runtime, based on current application and system state.

*This paper is submitted to PDCS 2000 as category C, applications and algorithms. The work presented here was supported by the Swedish Foundation for Strategic Research via the Swedish Research Council for Engineering Sciences and the National Network in Applied Mathematics, and by the National Science Foundation via grants numbers WU-HT-99-19 P029786F (KDI) and ACI 9984357 (CAREERS) awarded to Manish Parashar

†Email: johans@tdb.uu.se
‡Email: michael@tdb.uu.se
§Email: sumir@caip.rutgers.edu
¶Email: parashar@caip.rutgers.edu

Dynamical adaptive mesh refinement (AMR) [20] methods for the numerical solution to partial differential equations employ locally optimal approximations and yield highly advantageous ratios for cost/accuracy compared to methods based upon static uniform approximations. Distributed implementations of these techniques lead to significant challenges in dynamic resource allocation, data-distribution and load balancing, communication and coordination, and resource management. Critical among these is the partitioning problem, where load balance, communications, data migration costs, and other overheads such as grid aspect ratios must be optimized simultaneously. The primary motivation for the research presented in this paper is the observation that *no single partitioning scheme performs the best* for all types of applications and computer systems. Even for a single application, the most suitable partitioning technique depends on input parameters and its runtime state [11, 16].

In this paper, we first present an experimental study of six dynamic domain-based partitioners, using a suite of five "real-world" (2D and 3D) SAMR application kernels. The partitioners studied include existing (ISP[10] and G-MISP[19]) as well as new (G-MISP+SP, pBD-ISP and SP) techniques and constitute a selection from popular software tools, viz. GrACE[10], ParMetis[6] and Vampire[18]. Application kernels are taken from varied scientific and engineering domains, and demonstrate very different run-time behavior. The experimental results are then used to characterize the behavior of the partitioners using a five-component goodness metric, and to associate the partitioner(s) with application and system states. The paper makes 4 key contributions: (1) It presents 3 new partitioning/load-balancing techniques for SAMR grid hierarchies. (2) It presents an experimental characterization of the new and existing partitioning/load-balancing techniques using 5 different SAMR applications. (3) It proposes a 5-component goodness metric to evaluate dynamic partitioning and load-balancing techniques for parallel/distributed SAMR grid hierarchies. (4) It outlines policies for the selection of SAMR partitioners based on application and system state.

The rest of the paper is organized as follows. In section 2, a brief description of SAMR methods, partitioning requirements, and related work is presented. Section 3 defines partitioning quality for SAMR applications and presents the metric used in our experimentations. Section 4 outlines our experimental evaluation. It describes the partitioning techniques, the SAMR applications used in the evaluation, and presents the numerical results obtained. Section 5 uses the experimental results to characterize the partitioning techniques. Section 6 presents some concluding remarks.

2 Partitioning of Structured Grids - Overview and Related Work

Dynamically adaptive numerical techniques for solving differential equations provide a means for concentrating additional resolution and computational effort to regions in the application domain with high error. In these techniques, refinement proceeds recursively so that regions requiring higher resolution are flagged, and finer grids are overlaid on these regions, forming a dynamic adaptive grid hierarchy [3, 1] (see Fig 1).

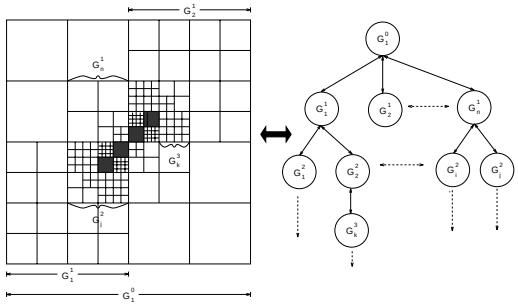

Figure 1: Berger-Oliger formulation of adaptive grid hierarchies for SAMR applications

The overall efficiency of the adaptive algorithms is limited by the ability to partition the underlying data-structures at run-time to expose all inherent parallelism, minimize communication and synchronization overheads, and balance load. A critical requirement 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.

Existing research on partitioners for SAMR applications can be broadly classified into *patch-based partitioners* [15, 21, 11] and *domain-based partitioners*. For 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* combine the two approaches described above. Partitioning is performed in two steps. 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 [7] and the tool KeLP [5] fully support patch-based partitioning and let you create parallel SAMR applications with minimum effort. However, for achieving scalable performance, domain-based partitioning techniques are preferable [15, 21, 11, 17].

All partitioners in this study are domain-based partitioners since our objective is to decrease run times for large scale SAMR applications.

3 Defining Partitioning Quality

The partitioning requirements for adaptive applications depend on the current state of the application and the computing environment. Therefore, it is of little consequence to discuss the absolute "goodness" of a particular partitioning technique. Hence, we base our characterization of partitioning behavior on the tuple (partitioner, application, computer system), (PAC). Further, we define a five-component metric to evaluate each PAC. The goal is to capture the overall runtime behavior of the partitioner. We believe that this is critical to understanding the suitability of a particular partitioner or *why* one PAC works better than some other PAC. The proposed metric for characterizing the quality of a PAC include:

1. **Communication requirement:** This component is a measure of how much the processors have to communicate with each other until the next repartition.

2. **Load imbalance:** This component measures the load imbalance created by a partitioner. It occurs when any processor has more than average load and can cause performance to quickly deteriorate.

3. **Amount of data migration:** This is a measure of the ability of the partitioner to consider the existing distribution. This is particularly important for SAMR applications as they require re-partitioning at regular intervals.

4. **Partitioning time:** This component tells how fast the partitioning algorithm computes the new partitioning, given the current partitioning.

5. **Partitioning induced overhead:** This component attempts to capture the quality of the partition generated by the partitioner, using the number of sub-grids and their aspect ratio.

Optimizing all the above components implies conflicting objectives. For example, optimizing components (1) and (2) together constitutes an NP-hard problem. Partitioners typically optimize a subset of the components at the expense of others. Our goal in defining the five-component metric is to facilitate the determination of trade-offs for each partitioning technique.

4 An Experimental Evaluation of Partitioning Techniques for SAMR Applications

The goal of the presented research is to construct a dynamic meta partitioner, that can adapt to (and optimize for) fully dynamic PACs. In this paper we move towards this goal by characterizing each partitioner (and partitioner configuration) for different applications and application states. That is, only the (A)pplication is allowed to be dynamic. The outline of the experiments is as follows. Partitioning quality is measured for each PAC, each with different application behavior. For each of the six partitioning schemes, we measure the five components of our quality metric, for each applications on 64 processors.

The partitioners are evaluated using an AMR simulator driven by application adaptation traces from a single processor run. The traces contain the state of the grid hierarchy (without any partitioning) at each stage of the adaptive application. The AMR simulator enables the user to select the partitioner to be used, partitioning parameter (block size, granularity), and the number of processors. Then, it invokes the selected partitioner on each set of grids in the trace and evaluates the generated partition using the metrics defined above. More information about the simulator can be obtained from [14].

4.1 The SAMR Applications

The experimentation presented in this paper uses a suite of five "real-world" (2D and 3D) SAMR application kernels taken from varied scientific and engineering domains, and demonstrate very different run-time behavior and adaptation patterns. Application domains include numerical relativity (scalarwave 2D & 3D), oil reservoir simulations (buckley-leverette 2D & 3D), and computational fluid dynamics (2D compressible turbulence - rm2D, and supersonic flows - enoamr2D). The applications use 3 levels of factor 2 refinements. Refinements are performed every 4 time-steps in each case. The applications are executed for 100 time-steps.

4.2 Partitioning Techniques for SAMR Grid Hierarchies

Six different partitioning techniques are included in this test. The first partitioning technique is a part of GrACE [10]. The partitioning techniques (2)-(5) are part of the parallel SAMR partitioning library *Vampire* [18], which combines structured and unstructured partitioning techniques in a fashion introduced by Rantakokko [11]. The techniques (3)-(5) are not described in the literature previously. The last algorithm is from ParMetis [6], a tool for unstructured graphs.

For some of the partitioning techniques, *granularity* may be varied. In this context, granularity is determined by the *atomic unit*, the smallest entity the partitioner can "see" and work with. By choosing a small atomic unit, the partitioner may split the grids (if necessary) into a myriad of tiny sub-grids, all having the size of the atomic unit. In this case, we have a fine granularity partitioning with potentially good load-balance. On the other hand, if we choose a big atomic unit, the partitioner may be limited by it and may not be able to split blocks that need to be split in order to achieve load balance. In this case, we have a coarse granularity partitioning. Note that large atomic units always result in coarse granularity partitionings. Small atomic units, however, may or may not result in fine granularity partitionings. In theory, fine granularity enables perfect load balance, but induces lots of communication and overhead. Conversely, coarse granularity promotes good communication patterns but prohibits good load balance.

4.2.1 Space-Filling Curve-Based Partitioning (SFC)

The SFC partitioning scheme is based on a recursive linear representation of a multi-dimensional grid hierarchy generated using space-filling mappings [12, 13]. These mappings are computationally efficient,

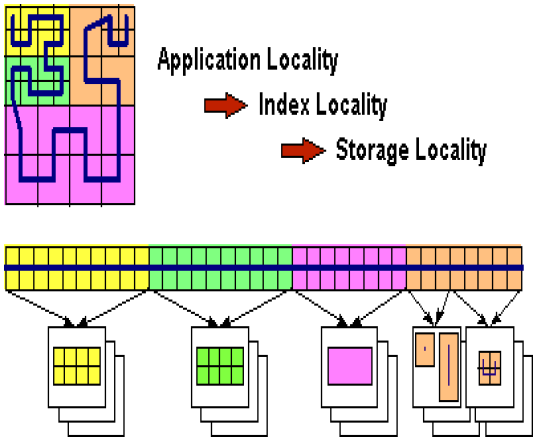

Figure 2: Space-filling mappings encode application domain locality and maintain locality through expansion and contraction

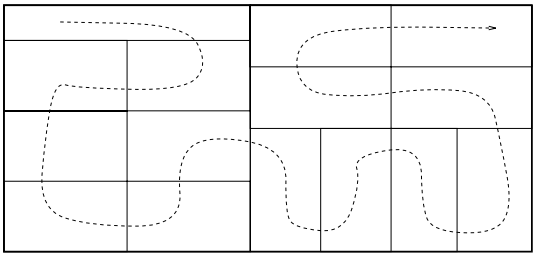

Figure 3: The pBD-ISP algorithm partitions the domain into p parts. Load balance is asserted in each cut. In the figure $p = 15$ and a uniform grid is used for simplification

recursive mappings from N-dimensional space to 1-dimensional space. The solution space is first partitioned into blocks of a minimum acceptable granularity that may be dictated by the system and/or the application. The space-filling curve then passes through these blocks. Space-filling mappings encode application domain locality and maintain this locality through expansion and contraction. The self-similar nature of these mappings is exploited to maintain locality across levels of the grid hierarchy while their locality preserving characteristics guarantee locality within partitions of individual grids in the hierarchy. Using the SFC-based linear representation, the problem of partitioning and dynamically re-partitioning the multi-dimensional grid hierarchy is thus reduced to decomposing its one-dimensional representation to balance load across the processors. Computational load is determined by the length of the SFC segment and the levels of recursion it contains. The former corresponds to block sizes while the latter corresponds to the levels of refinement. The SFC-based partitioning scheme is illustrated in Fig 2 and is used by parallel SAMR infrastructures such as GrACE and Paramesh [8].

4.2.2 Geometric multilevel inverse space-filling curve partitioning (G-MISP) with variable grain size

The G-MISP technique [19] is a multilevel algorithm similar to SFC described above. It starts by creating a matrix of workloads from the SAMR grid hierarchy, and viewing this matrix as a one-vertex graph. The matrix entries correspond to grid points in the application base grid along with their refinements, and its workload is the total work required to advance the solution in the area covered by the matrix entry. The G-MISP scheme successively refines the graph by splitting vertices with total weight greater that a threshold. The minimum vertex size is bounded by the *atomic unit* parameter. Therefore, the resulting partitioning granularity is a function of the threshold, the atomic unit, and the current state of the grid hierarchy. The splitting of the vertices is structured. A vertex is split at the geometrical mid point, and the orientation is determined by the Hilbert space-filling curve. As a result of this splitting, an one-dimensional list of blocks having approximately the same weight is created. Load balance is now trivially achieved by assigning the same number of blocks to each processor. The G-MISP approach favors speed at the expense of load-balance.

4.2.3 Geometric multilevel inverse space-filling curve partitioning with sequence partitioning (G-MISP+SP) and variable grain size

The G-MISP+SP is a "smarter" variant of the G-MISP technique. It uses sequence partitioning [9] [1] to assign blocks from the one-dimensional list to processors, instead of just assigning the same number of blocks to each processor. As a result, load balance improves. The scheme is however computationally more expensive.

4.2.4 The p-way binary dissection inverse space-filling curve partitioning (pBD-ISP)

The pBD-ISP scheme is a generalization of the binary dissection algorithm discussed in [2], extending this scheme in two ways. First, the domain is partitioned into p partitions without restrictions on p. The binary dissection is performed in such a way that each split divides the load as evenly as possible, taking the available numbers of processors into account. That is, if there are e.g. 15 processors available, the cut is made such that 8/15 *of the load* is in one part and 7/15 is in the other part. Second, the orientation of the cuts is determined by the Hilbert space-filling curve, hence incorporating unstructuredness into the scheme. The idea is illustrated in Fig 3.

4.2.5 "Pure" sequence partitioning with inverse space-filling curves (SP-ISP) and variable grain size

In this scheme, the domain is subdivided into $p * b$ equally sized blocks by a generalization of the parameterized binary dissection (BD) algorithm described in [4]. The BD algorithm is extended in two ways. First, it is a dual-level algorithm, enabling different parameter settings for each level. In the first level, the domain is split into b blocks. In the second level, all b blocks are furhter divided into p blocks each. Second, the orientation of the cuts is determined by the Hilbert SFC. In the case of SAMR grid hierarchies, blocks generated will have different workloads due to adaptation of the grid. These blocks are ordered according to the SFC to from a one-dimensional list, which is then partitioned using sequence partitioning [9]. SP-ISP is potentially a fine granularity partitioning scheme, leading to good load balance at the expense of increased communication, overhead, and a higher computational cost.

4.2.6 Wavefront diffusion (WD) based on global work load

The wavefront diffusion algorithm is based on global work load and is part of the suite of partitioning algorithms included in *ParMetis* [6]. Our motivation for including it in our study is that this algorithm specializes in repartitioning graphs where the refinements are scattered. This feature is lacking in the other techniques studied. ParMetis is designed for unstructured grids and requires a node graph as its input. In the case of SAMR block-structured grids, it is not feasible to work with grid graphs where each point in the computational grid corresponds to a vertex in the graph. This is because the time required to generate the graph will be prohibitive and the resulting partitioning will be extremely irregular, and jagged. As a result, we use grid blocks of size greater than or equal to the atomic unit as vertices of the node graph. The WD scheme results in fine grain partitionings with jagged partitioning boundaries and potentially increased communication costs and overheads.

[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.

Scheme	LB	Comm	DM	OH	Speed
G-MISP	-o- - - - -	ooooooo	ooooooo	ooooooo	+
G-MISP+SP	+++o+o+	ooooooo	ooooooo	ooooooo	o
pBD-ISP	ooo-o- -	+++o+oo	+o+o+o+	+++++++	+
SP	++- -o-o	-oo+o+o	ooo+- -o	-o- -o-o	-
ISP	- -o- - - -	- -o-o- -	- -oo-oo	o++++o+	+

Table 1: Conceptual view of the experimental evaluation of the partitioners for 7 SAMR applications on 64 processors. For each application (1 through 7 listed in Table 3) the grade (+,o,-) indicates how well the partitioner succeeded in optimizing the particular metric. A '+' means significantly better than the other schemes, a '-' means significantly worse, and 'o' is for average or OK. 'LB'=Load-Balance, 'Comm'=communication, 'DM'=data movement, and 'OH'=overhead

4.3 Experimental Results

An overview of the experimental evaluation of the partitioners using 64 processors is presented in Table 1. For each partitioning technique, this table indicates the behavior of each metric by grading it as '+' for good, as '-' for bad, or as 'o' for average or OK. All grades are relative to how hard it was in average for the partitioners to optimize this particular metric. That is, a load imbalance of 20 percent may be considered to be OK for a particular application, while the corresponding figure might be as low as 5 percent for another application. This is shown in Table 3, where each metric is given a '+' for large effort, a '-' for small effort, or an 'a' for average effort required to optimize. Due to lack of space, detailed results are shown only for the scalarwave and the Buckley-Leverette 2D (Fig 5) and 3D (Fig 4) applications. The performance of each partitioner is discussed below.

G-MISP

The G-MISP technique is fast and receives average grades in all other metrics except load balance. The load balance generated by this scheme is poor due to the rudimentary load balancing approach.

G-MISP+SP

The G-MISP+SP scheme uses sequence partitioning to remedy G-MISP's poor load balance. The result is that load balance gets close to optimal at the expense of an additional computational cost.

pBD-ISP

Clearly, the pBD-ISP is a good overall partitioner. It excels in the speed and overhead metrics, and generates little communication and data movement. Load balance is its weakest spot. It does, however, receive average grades for load balance for about half of the applications.

SP

The SP technique was not a great success. It is computationally more intense than the other schemes, and its behavior is somewhat unpredictable. It is interesting to see that SP generates a worse load balance than G-MISP+SP in many cases. We believe that this is due to the differences in the sequences fed to the sequence partitioner.

ISP

The ISP technique is also very fast and generates low overhead. It does however struggle with load balance. Its behavior is similar to G-MISP, which is expected since the two schemes are very similar. However, ISP favors the overhead metric at the expense of communication.

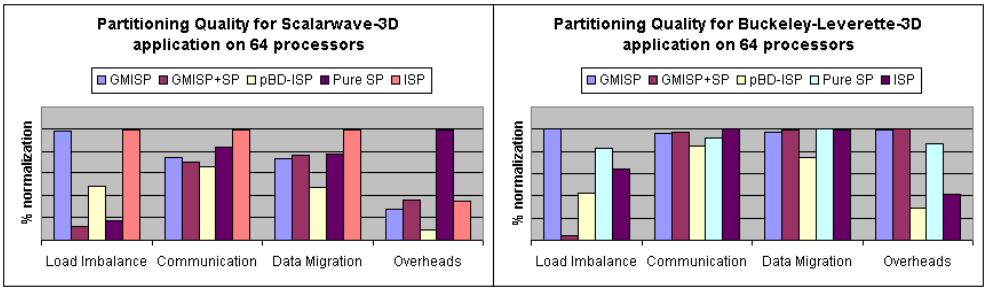

Figure 4: Partitioning quality for the scalar-wave (left) and Buckley-Leverette (right) 3D applications on 64 processors for each partitioning scheme

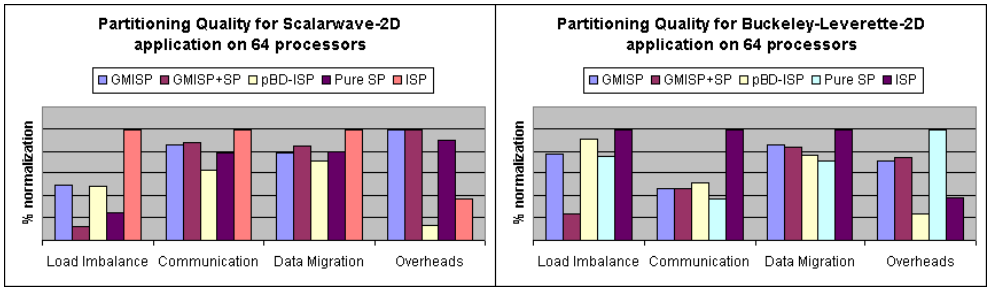

Figure 5: Partitioning quality for the scalar-wave (left) and Buckley-Leverette (right) 2D applications on 64 processors for each partitioning scheme

WD

Our integration of the ParMetis WD partitioner could not compete with the other partitioners due to extremely large partitioning costs. Even though ParMetis is known to produce high-quality partitionings at a low cost, we had to perform two extra steps to adapt it to SAMR grid hierarchies. Prior to partitioning, we had to generate a ParMetis graph from the grid. After partitioning, we had to use clustering to regenerate grid block from from graph partitions. As a consequence, the dedicated SAMR partitioners easily outperformed our ParMetis integration.

5 Application-Centric Characterization of the Partitioning Techniques

An application-centric characterization of the partitioners is obtained by first classifying the SAMR application state and identifying the partitioning requirements of each application state. We then assign partitioner(s) to the application states based on the experimental evaluation.

The *octant approach* (Fig 6) is used to classify the state of the SAMR application. The octant approach is an extension to the quadrant approach introduced in [19]. According to the octant approach, application state is 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.

In the "far part" of the cube, solutions have high activity dynamics. This gives rise to increased need for data movement. Furthermore, partitioning efficiency is crucial as we need frequent repartitioning. Techniques like pBD-ISP and even G-MISP+SP meet these requirements. In the "close part" of the cube, there is more time for computing the partitioning. Thus, schemes like G-MISP+SP are strong candidates here. In the "upper part" of the cube, communication dominates run-time. As a result partitioners such as

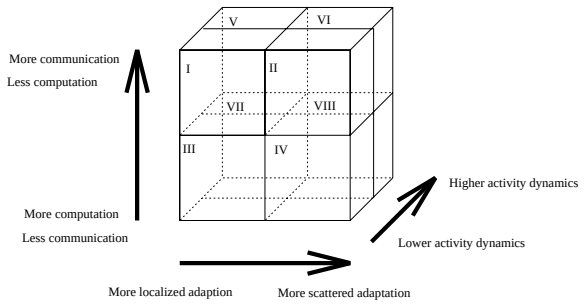

Figure 6: According to the octant approach, application state and computer system state are classified with respect to (a) the adaptation pattern (scattered or localized), and (b) whether run-time is dominated by computations or communications, and (c) the activity dynamics in the solution. Applications may start off in one octant, then, as solution progresses, migrate to any other

Octant	Scheme
I	pBD-ISP
II	pBD-ISP
III	G-MISP+SP
IV	G-MISP+SP, SP
V	pBD-ISP
VI	pBD-ISP
VII	G-MISP+SP
VIII	G-MISP+SP

Table 2: Mapping of partitioning techniques to application state octants. Clearly, the two most successful schemes are pBD-ISP and G-MISP+SP

Application	LB	Comm	DM	OH
SW3D	a	+	a	a
SW2D	-	a	+	+
BL3D	a	+	+	-
BL2D	a	-	a	+
RM2D	a	+	a	a
ENO2D	+	+	-	+
Tr2D	a	a	a	+

Table 3: Applications partitioning requirements. A '+' means significantly tougher to optimize this metric than for the other applications, a '-' means significantly easier, and 'a' is for average. 'LB'=Load-Balance, 'Comm'=communication, 'DM'=data movement, and 'OH'=overhead

pBD-ISP that optimize this metric are best suited. On the other hand, in the "lower part", computation dominates, which suggests that schemes that optimize load balance (such as G-MISP+SP or SP) are preferable. In the "right part" of the cube, adaptation is scattered over the computational domain. This typically means more overhead. Both ISP and pBD-ISP optimizes this metric well. Finally, in the "left part" of the cube, adaptation is strongly localized and results in lesser data movement and overheads. Moreover, load balance is harder to achieve, indicating that techniques like G-MISP+SP, SP, or pBD-ISP are suitable. The association of partitioning techniques to application state octants using the experimental evaluation presented in this papers are summarized in Table 2.

6 Conclusions

The goal of the research presented in this paper is to support the formulation of policies required to drive a *dynamically adaptive* meta-partitioner for SAMR grid hierarchies capable of decreasing overall execution time by selecting the most appropriate partitioning strategy (from a family of available partitioners) at runtime, based on current application and system state. This paper presents an application-centric characterization of domain based partitioners. Six dynamic partitioning schemes are evaluated using seven different SAMR applications. The results in Table 2 represent a first step towards enabling the dynamic meta-partitioner. It provides a mapping of partitioners onto application state octants, i.e. it enables cur-

rent application (and system) state to dictate partitioning requirements and the selection of an appropriate partitioner.

References

[1] M. J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics*, 82, 1989.

[2] Marsha J. Berger and Shahid Bokhari. A partitioning strategy for non-uniform problems on multiprocessors. *IEEE Trans. on Computers*, 85:570–580, 1987.

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

[4] S. H. Bokhari, T. W. Crockett, and D. M. Nicol. Binary dissection: variants & applications. Technical Report ICASE Report No. 97-29, NASA Langley Research Center, Hampton, VA, 1997.

[5] Stephen J. Fink, Scott B. Baden, and Scott R. Kohn. Flexible communication mechanisms for dynamic structured applications, 1996. To appear in IRREGULAR '96.

[6] G. Karypis, K. Schloegel, and V. Kumar. PARMETIS - parallel graph partitioning and sparse matrix ordering library, version 2.0. Univ. of Minnesota, Minneapolis, MN, 1998.

[7] Scott Kohn. Samrai, structured adaptive mesh refinement applications infrastructure. 1999.

[8] Peter MacNeice. http://sdcd.gsfc.nasa.gov/ESS/macneice/paramesh/paramesh.html. Paramesh homepage, 1999.

[9] Bjorn Olstad and Fredrik Manne. Efficient partitioning of sequences. Technical report, Dep of Computer Systems and Telematics, The Norweigian Institute of Technology.

[10] M. Parashar, et al. A common data management infrastructure for adaptive algorithms for PDE solutions. Technical Paper at Supercomputing '97, 1997.

[11] Jarmo Rantakokko. *Data Partitioning Methods and Parallel Block-Oriented PDE Solvers*. PhD thesis, Uppsala University, 1998.

[12] Hans Sagan. Space-filling curves. *Springer-Verlag*, 1994.

[13] H. Samet. The design and analysis of spatial data structure. *Addison-Wesley, Reading, Massachusetts*, 1989.

[14] Mausumi Shee. Evaluation and optimization of load balancing/distribution techniques for adaptive grid hierarchies. http://www.caip.rutgers.edu/ TASSL/Thesis/mshee-thesis.pdf. 2000.

[15] Mausumi Shee, Samip Bhavsar, and Manish Parashar. Characterizing the performance of dynamic distribution and load-balancing techniques for adaptive grid hierarchies. 1999.

[16] Johan Steensland. http://www.tdb.uu.se/~johans/. Work in progress, 1999.

[17] Johan Steensland. A theoretical discussion about inverse space-filling curve partitioning. Work in progress, 1999.

[18] Johan Steensland. Vampire homepage, http://www.caip.rutgers.edu/~johans/vampire/. 2000.

[19] Johan Steensland, Stefan Soderberg, and Michael Thuné. A comparison of partitioning schemes for blockwise parallel SAMR applications. *Springer...*, 2000.

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

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