An Application-Centric Characterization of Domain-Based SFC Partitioners for Parallel SAMR*

Johan Steensland†, Sumir Chandra, and 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
{johans, sumir, parashar}@caip.rutgers.edu

Abstract

Structured adaptive mesh refinement (SAMR) methods for the numerical solution to partial differential equations yield highly advantageous ratios for cost/accuracy as compared to methods based on static uniform approximations. These techniques are being effectively used in many domains including computational fluid dynamics, numerical relativity, astrophysics, subsurface modeling and oil reservoir simulation. Distributed implementations of these methods lead to significant challenges in dynamic data-distribution, load-balancing, and runtime management. This paper presents an application-centric characterization of a suite of dynamic domain-based inverse space-filling curve partitioning techniques for the distributed adaptive grid hierarchies that underlie SAMR applications. The overall goal of this research is to formulate 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. Such a meta-partitioner can significantly reduce the execution time of SAMR applications.

Keywords: Dynamic domain-based partitioning and load-balancing; Inverse space-filling curve partitioning; Performance characterization; Structured adaptive mesh refinement; Adaptive meta-partitioner.

 This work was supported by the National Science Foundation via grants numbers WU-HT-99-19 P029786F (KDI) and ACI 9984357 (CAREERS) awarded to Manish Parashar and by the Swedish Foundation for Strategic Research via the Swedish Research Council for Engineering Sciences and the National Network in Applied Mathematics

†Visiting from Information Technology, Dept. of Scientific Computing, Uppsala University, Sweden

1 Introduction

This paper presents an experimental application-centric characterization of dynamic, domain-based, inverse space-filling curve partitioning (ISP) techniques for distributed, structured adaptive grid hierarchies. These hierarchies underlie parallel structured adaptive mesh refinement (SAMR) methods for the solution to partial differential equations (PDE's) [5, 7, 38]. The presented work is part of an ongoing research project with the overall goal of engineering a *dynamically adaptive* meta-partitioner for SAMR grid hierarchies capable of selecting the most appropriate partitioning strategy at runtime based on current system and application state. Such a meta-partitioner can significantly reduce the execution time of SAMR applications [13].

Dynamically adaptive mesh refinement (AMR) [45] methods for the numerical solution to PDE's employ locally optimal approximations, and 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 refining the computational grid in regions with large local solution error. Structured adaptive mesh refinement (SAMR) methods are based on uniform patch-based refinements overlaid on a structured coarse grid, and provide an alternative to the general, unstructured AMR approach. These methods are being effectively used for adaptive PDE solutions in many domains, including computational fluid dynamics [1, 6, 33], numerical relativity [14, 19], astrophysics [2, 10, 24], and subsurface modeling and oil reservoir simulation [31, 50]. Methods based on SAMR can lead to computationally efficient implementations as they require uniform operations on regular arrays and exhibit structured communication patterns. Furthermore, these meth-

ods tend to be easier to implement and manage due to their regular structure. Distributed implementations of these methods offer the potential for accurate solutions of physically realistic models of complex physical phenomena. These implementations lead to interesting challenges in dynamic resource allocation, data-distribution, load-balancing, and runtime management. Critical among these is the partitioning of the adaptive grid hierarchy to balance load, optimize communication and synchronization, minimize data migration costs, and maximize grid quality (e.g. aspect ratio) and available parallelism.

Techniques based on ISP are increasingly being used to partition structured grid hierarchies. The self-similar nature of space-filling curves (SFC's) [8, 37, 39] complement the recursive refinements of SAMR grids. Furthermore, the proximity preserving properties of SFCs can be exploited to translate application domain locality to storage locality [15, 23, 30]. Finally, these techniques are computationally efficient and are suited to SAMR algorithms, which require regular regridding steps. Techniques based on ISP are used by SAMR infrastructures such as Paramesh [22] and DAGH/GrACE [30].

The primary motivation for the research presented in this paper is the observation that in the case of parallel 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's runtime state [36, 42]. This necessitates an adaptive management of these dynamic applications at runtime. This includes using application runtime state to select and configure the partitioning strategy so as to maximize performance. The goal of the adaptive *meta-partitioner* is to provide such a capability for parallel SAMR applications. The application-centric characterization of ISP-

based SAMR partitioners presented in this paper is crucial to the formulation of the polices that would drive the adaptive meta-partitioner.

In this paper, we first present an experimental study of five dynamic domain-based ISP techniques using a suite of "real-world" (2D and 3D) SAMR application kernels. The partitioners studied include existing techniques (ISP [29] and G-MISP [44]) as well as new variants (G-MISP+SP, pBD-ISP, and SP-ISP) that extend the existing techniques. These partitioners constitute a selection from popular software tools such as GrACE [29] and Vampire [41]. Application kernels are taken from varied scientific and engineering domains and exhibit a wide spectrum of runtime behavior. The experimental results are then used to characterize the partitioners' behavior using a five-component goodness metric, and to associate partitioner(s) with corresponding application state. The paper makes four key contributions: (1) It introduces three new ISP variants for SAMR grid hierarchies, (2) It proposes a five-component goodness metric to evaluate dynamic partitioning techniques for distributed SAMR grid hierarchies, (3) It presents an experimental evaluation and application-centric characterization of the new and existing partitioning techniques using seven SAMR applications, and (4) It outlines policies for the selection of SAMR partitioners based on application state.

The rest of the paper is organized as follows. Section 2 presents a brief overview of structured adaptive mesh refinement. Section 3 outlines approaches for the partitioning of SAMR grid hierarchies and presents related work. Section 4 introduces the architecture and operation of the adaptive meta-partitioner. This section also defines partitioning quality for SAMR applications and presents the five-component metric used in our study. Section 5

describes the suite of SAMR partitioning techniques studied. Section 6 introduces the SAMR applications used, and presents the experimental evaluation and results obtained. Section 7 uses the experimental results to characterize partitioning techniques based on application state. Section 8 presents concluding remarks and outlines future work.

2 Structured Adaptive Mesh Refinement: An overview

The numerical solution to a PDE is obtained by first discretizing the problem domain. One approach is to introduce a structured uniform Cartesian grid. The unknowns of the PDE are then approximated numerically at each discrete grid point. The resolution of the grid (or grid spacing) determines the local and global error of this approximation, and is typically dictated by the solution-features that need to be resolved. The resolution also determines computational costs and storage requirements. Dynamically adaptive solution techniques for PDE's provide a means for concentrating additional grid resolution and computational resources to regions in the application domain with large error. These techniques potentially lead to more efficient and cost-effective solutions to time-dependent problems exhibiting localized features, viz. shocks, discontinuities, or steep gradients.

In the case of 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 techniques start with a coarse base grid with minimum acceptable resolution that covers the entire computational domain. As the solution progresses, regions in the domain with large solution error, requiring additional resolution, are identified and refined. Refinement proceeds recursively so that the refined regions requiring higher resolution are similarly tagged and even finer grids are overlaid on these regions. The resulting grid struc-

ture is a dynamic adaptive grid hierarchy. The adaptive grid hierarchy corresponding to the SAMR formulation by Berger and Oliger [7] is shown in Fig. 1. 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.

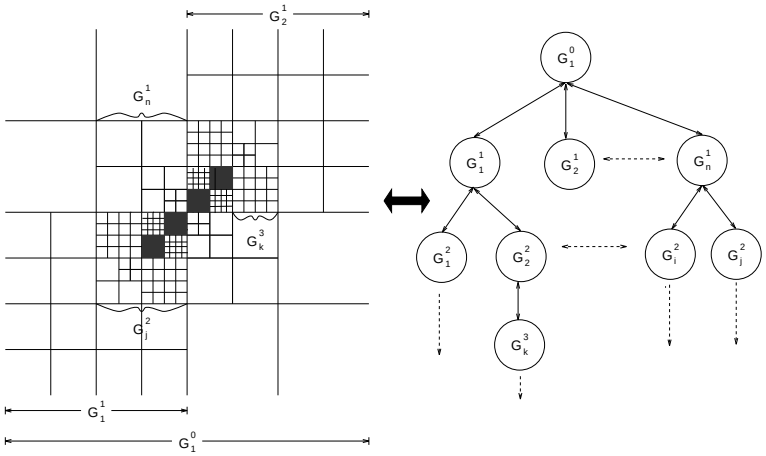

Figure 1: Berger-Oliger formulation of adaptive grid hierarchies for SAMR applications.

3 Related Work

3.1 Partitioning SAMR Grid Hierarchies

Parallel implementations of SAMR methods present interesting challenges in dynamic resource allocation, data-distribution, load-balancing, and runtime management. The overall efficiency of parallel SAMR applications is 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. A critical requirement when 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.

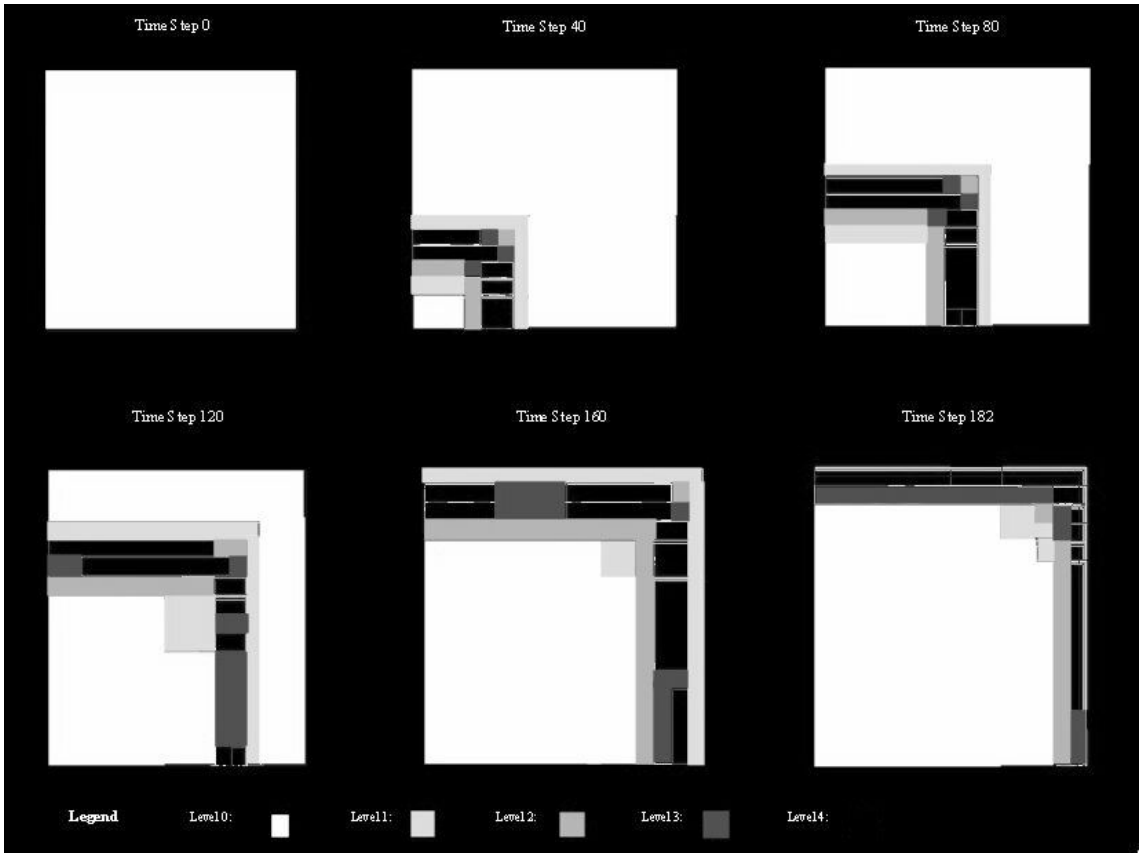

Figure 2: A sequence of dynamic adaptive grid hierarchies for a 2-D Buckley-Leverette SAMR simulation. Note the dynamic creation, deletion and migration of grids at different levels.

The former enables efficient computational access to the grids and minimizes the parent-child communication overheads, while the latter minimizes overall communication and synchronization overheads. Furthermore, application adaptation results in grids being dynamically created, moved and deleted at runtime, making it necessary to efficiently repartition the hierarchy "on the fly" so that it continues to meet these goals.

Partitioners for SAMR grid hierarchies can be classified as patch-based, domain-based, or hybrid.[1] In the case of *patch-based partitioners*, distribution decisions are independently

[1]Note that this study focuses exclusively on partitioning techniques for adaptive structured grids. Similar classification and comparative studies for unstructured-grid/mesh partitioning and dynamic load-balancing have been investigated in the literature [47, 48].

made for each newly created grid. A grid may be kept on the local processor or entirely moved to another processor. If the grid is too large, it may be split (see Fig. 3). *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, which 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 [3] and KeLP [16] model) fully supports patch-based partitioning. The distribution scheme maps the patches at a refinement level of the AMR hierarchy across processors. Domain-based partitioners are presented in [28, 36, 46] and are the primary focus of this paper. Some hybrid approaches are presented in [28].

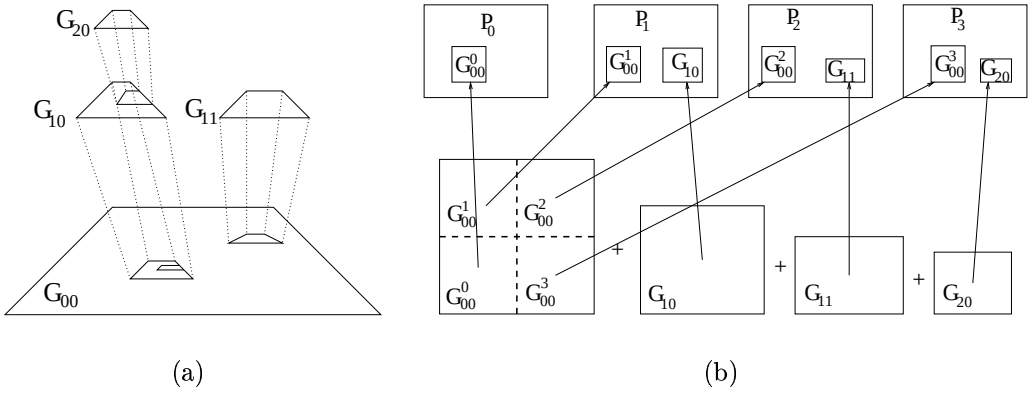

Figure 3: In patch-based partitioning, distribution decisions are made independently for each newly created patch. A patch may be kept on the local processor or entirely moved to another processor. If the patch is considered too large, it may be split. An example of such a patch is G_{00} in sub-figure (a), which is split into $G^0_{00} - G^3_{00}$ in sub-figure (b).

One possible strategy for patch-based partitioning is to distribute each patch in equal portions over all processors. This would result in a good load balance independently of the

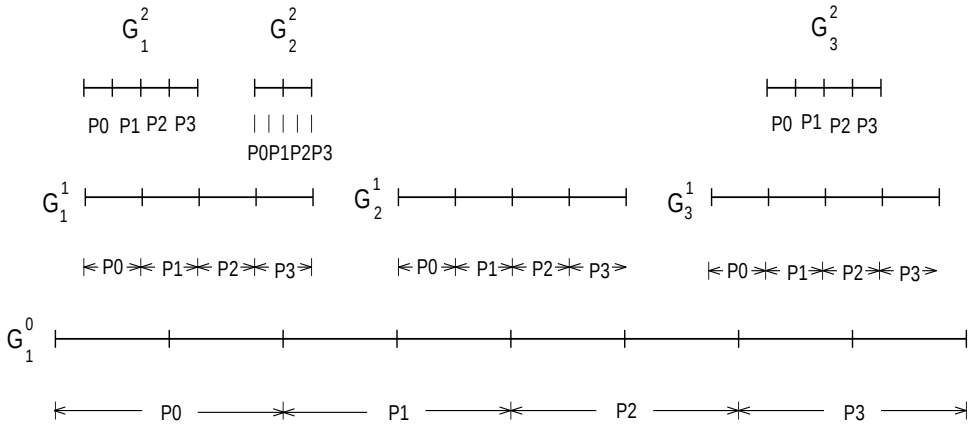

Figure 4: A patch-based partitioning of a 1-D SAMR grid hierarchy. Note that inter-level communication lead to expensive gather/scatter operations in this case. For example, a restriction from grid G_2^2 to grid G_1^1 requires all the processors to communicate with processor *P3*. Furthermore, parallelism across grids at a level is not exploited. For example, grids G_1^1, G_2^1, and G_3^1 are distributed across the same set of processors and have to be integrated sequentially.

depth and complexity of the grid hierarchy, but it requires each patch to be large enough to allow for this partitioning. With this strategy, redistribution when grids are created or deleted is not required. However, earlier studies [46] show that it is necessary to treat too small patches as special cases, hence adding complexity to the algorithms as well as communication and imbalance to the application. Patch-based techniques can lead to substantial "depth-wise" parent-child communication overheads. In an SAMR grid hierarchy, a fine grid typically corresponds to a relatively small region of the underlying coarse grid. If, like in the strategy outlined above, both the fine and coarse grids are distributed over the entire set of processors, all the processors will communicate with the small set of processors corresponding to the associated coarse grid region, thereby causing a serialization bottleneck. For example, in Fig. 4, a restriction from grid G_2^2 to grid G_1^1 requires all the processors to communicate with processor *P3*. All patch-based techniques suffer more or less from this type of shortcoming, regardless of strategy. Another problem with patch-based distributions is that

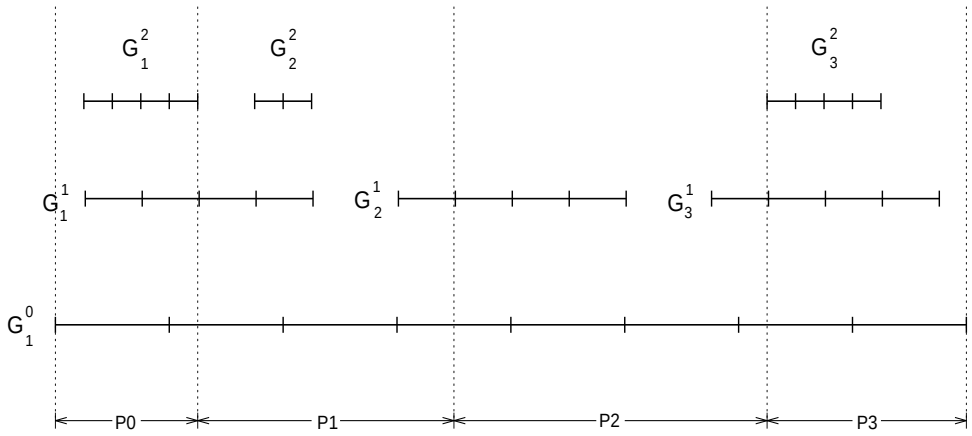

Figure 5: A domain-based partitioning of a 1-D SAMR grid hierarchy. By maintaining parent-child locality, inter-level communications are eliminated. Load redistribution is required when grids are created or destroyed.

all available parallelism is not exploited. In most SAMR algorithms, updates are performed level by level in a sequential manner. That is, the solution on all grids at refinement level l have to be advanced before the solution on any overlaid grid at level $l+1$ can be advanced. Thus, parallelism in SAMR applications primarily comes from operating on component grids at a level in parallel. For example, in Fig. 4, grids G_1^1, G_2^1, and G_3^1 are distributed across the same set of processors and have to be integrated sequentially. Another example is the simple patch-based scenario depicted in Fig. 4, where grids G_{10} and G_{11} are integrated *after* grid G_{00}. In both examples, processors will be temporarily idle and parallel efficiency will deteriorate.

Domain-based partitioners (Fig. 5), on the other hand, eliminate "depth-wise" communication by maintaining parent-child locality. Furthermore, these partitioners fully exploit the available parallelism at each level of the SAMR grid hierarchy. Domain-based partitioners, however, do require load redistribution when grids are created or destroyed. But this redistribution can be performed incrementally [28], thus minimizing the overheads involved

with creating a partitioning from scratch and remaping this partitioning onto the existing partitioning so as to minimize data redistribution.. Although domain-based partitioners can effectively support parallel SAMR, these techniques require the overall structure of the grid hierarchy to be maintained and partitioned, which can be challenging. Furthermore, the load-balance produced by domain-based partitioners for deep SAMR hierarchies with strongly localized regions of refinement can deteriorate significantly. In these cases, hybrid partitioners, combining patch-based and domain-based approaches, can be used.

3.2 Space-Filling Curves and SAMR Grid Hierarchies

Inverse space-filling curve partitioners (ISP) [11, 21, 27, 28, 32, 34, 42, 49] are widely used to implement domain-based partitionings of SAMR grid hierarchies. Space-filling curves (SFC) [8, 37, 39] are computationally efficient, locality preserving recursive mappings from N-dimensional space to 1-dimensional space i.e. $N^d \rightarrow N^1$, such that each point in N^d is mapped to a unique point or index in N^1. Two such mappings, the Morton order and the Peano-Hilbert order, are shown in Fig. 6. Two properties of space filling curves, viz. digital causality and self-similarity, make them particularly suitable for partitioning SAMR grid hierarchies. Digital causality implies that points close together in d-dimensional space will be mapped to points close together in 1-dimensional space, i.e. the mapping preserves locality [15, 17, 18, 23, 30]. Self-similarity implies that, as a d-dimensional region is refined, the refined sub-regions can be recursively filled by curves having the same structure as the curve filling the original (unrefined) region, but possibly a different orientation. Fig. 7 illustrates this property for a 2-dimensional region with refinements by factors of 2 and 3. In the case of SAMR grid hierarchies, this self-similar or recursive nature of SFC mappings

is exploited to represent the hierarchical structure and to maintain locality across different hierarchy levels. Finally, these techniques are computationally efficient and are suited to SAMR algorithms, which require regular regridding steps.

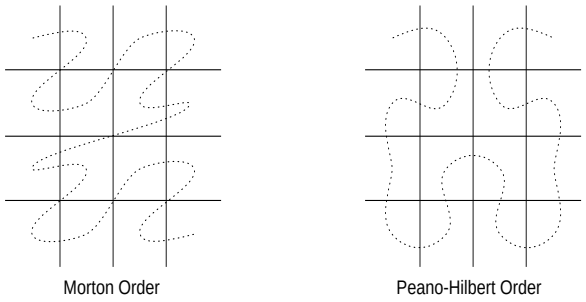

Figure 6: Examples of Space Filling Curves - Morton (left) and Peano-Hilbert (right).

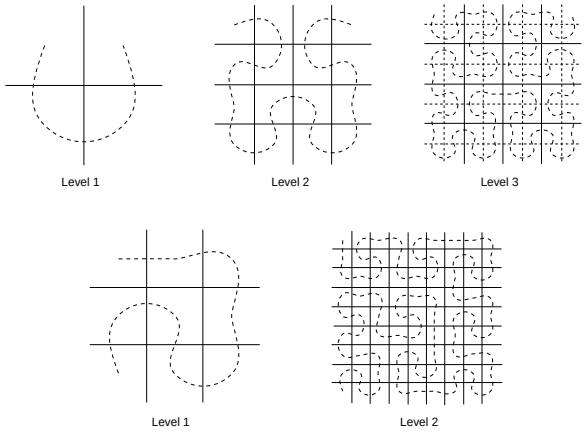

Figure 7: An illustration of the *Self-Similarity* property of Space Filling Curves.

Although it has been demonstrated that ISP techniques can be effectively used to support parallel applications [27, 35], and these techniques are used by SAMR infrastructures such as Paramesh and DAGH/GrACE, their effectiveness and scalability for a wide range of SAMR applications with varied number of refinement levels, refinement patterns, and refinement granularities have to be theoretically/experimentally verified (and understood). One goal of our research is to construct domain-based ISP methods based on theoretical models and experimental studies. This paper focuses on analyzing and characterizing the behavior of various domain-based ISP techniques for a range of dynamic SAMR applications with the objective of automating their selection and configuration.

4 Engineering an Adaptive Meta-Partitioner

The characterization of partitioners presented in this paper is part of a larger effort aimed at developing an adaptive meta-partitioner with the objective of *decreasing the run-times and improving the scalability of parallel SAMR applications.* A number of factors contribute to the overall performance of these applications. As a result, runtime optimization requires identifying an optimal combination of these parameters. Such an optimization can be a significant challenge as the relationships between the contributing factors are often complex, and depend on dynamic information such as the current state of the application and the system. One critical factor, the runtime selection and configuration of the partitioning strategy, is addressed by the adaptive meta-partitioner. The meta-partitioner uses the current state of the application and the system to dynamically select and configure the most appropriate partitioning strategy. This selection is based on the application-centric characterization of the partitioners. In this section, we first define partitioning quality. We then outline our model for classifying runtime state. Finally, a conceptual adaptive meta-partitioner is presented.

4.1 Defining Partitioning Quality

The partitioning requirements for an adaptive application (and the performance of a particular partitioner) 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 certain partitioning technique. We base our characterization of partitioning behavior on the tuple {*partitioner, application, computer system*}, (PAC). Furthermore, we define a five-component metric to evaluate each PAC. The goal of the metric 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 another PAC. The proposed metric for characterizing the quality of a PAC for the adaptive SAMR meta-partitioner include:

1. **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 deteriorate rapidly. Note that the load associated with a patch is the summation of the work associated with each cell/vertex in the patch (all cells/vertices in a patch may not require the same amount of work). This metric component assumes greater significance in a computation-dominated environment.

2. **Communication requirement:** This component is a measure of the inter-processor communication required by the SAMR algorithm, given a partitioning of the grid hierarchy. It includes communication between neighboring grid components and communication across refinement levels. This cost is determined by the geometrical intersections between SAMR grid components. This metric component assumes greater significance in a communication-dominated environment.

3. **Amount of data migration:** This component is a measure of the amount of data that has to be moved between processors when the application transitions from one partitioning to the next. It represents the ability of the partitioner to consider the existing distribution. This component is particularly important for SAMR applications as they require repartitioning at regular intervals. This metric component assumes greater significance for highly dynamic applications and scattered adaptations.

4. **Partitioning induced overhead:** This component attempts to capture the quality of the partitions generated by the partitioner, using the number of sub-grids and their aspect ratio. A large number of small grids may lead to a better load-balance, but can result in increased communication causing performance to deteriorate. Similarly, bad aspect ratio can adversely affect the computation-to-communication ratio, and can lead to bad memory access patterns.

5. **Partitioning time:** This component represents how fast the partitioning algorithm computes the new partitioning, given the current partitioning. It indicates the parallel efficiency of the partitioning algorithms employed. This component is particularly important for SAMR applications as they require repartitioning at regular intervals.

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 this metric is to be able to readily determine the trade-offs for each partitioning technique. Note that a cost-metric based on similar components is used by the PLUM [25] dynamic load-balancing strategy for adaptive unstructured meshes. The PLUM cost-metric uses computational, communication and data-remapping costs to estimate the computational gains of repartitioning. It then uses the metric to decide whether to commit the new partition or not. The quality metric used by our adaptive meta-partitioner for SAMR grid hierarchies additionally includes the speed of the partitioner and the partitioning overheads, and uses the metric to also select the type and configuration of the partitioner to be used.

4.2 Characterizing Runtime State

The *octant approach* (Fig. 8) [43] provides a means to classify the runtime state, i.e. the state of the application and the system, and is an extension to the quadrant approach introduced in [44]. 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 runtime is dominated by computations or communications, and (c) the activity dynamics in the solution, e.g. adaptation pattern changes rapidly.

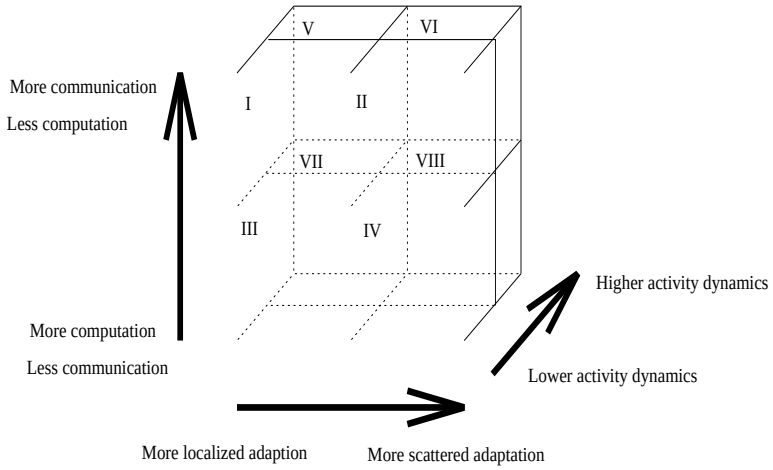

Figure 8: The *octant approach* for characterizing application state based on its adaptation pattern, communication/computation requirements and activity dynamics.

Applications may start in one octant, then, as solution progresses, migrate to others. For example, in the case of an application where the refined regions track a shock hitting a boundary, the refinements may be localized before impact and scattered after impact. Similar dynamic behavior may also be exhibited by the system. For example, load 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.

4.3 Dynamic Selection of Partitioning Techniques

Previous research has investigated the selection of the appropriate partitioner for a certain combination of the application and computer system to reduce runtime. Nevertheless, the PAC tuple is a dynamic entity. The (A)pplication and the (C)omputer system components are obviously changing. The (P)artitioner, however, is typically viewed as static, i.e. it is selected only once. This can be a serious limitation. A PAC should be fully dynamic and the *choice* of partitioning technique should be dynamic, rather than static. Consequently, we define the PAC relationship as $P_t = f(A_t, C_t)$, indicating that the partitioning technique P selected at a given time t should be a function of the state of the application A and the computer system C at that time.

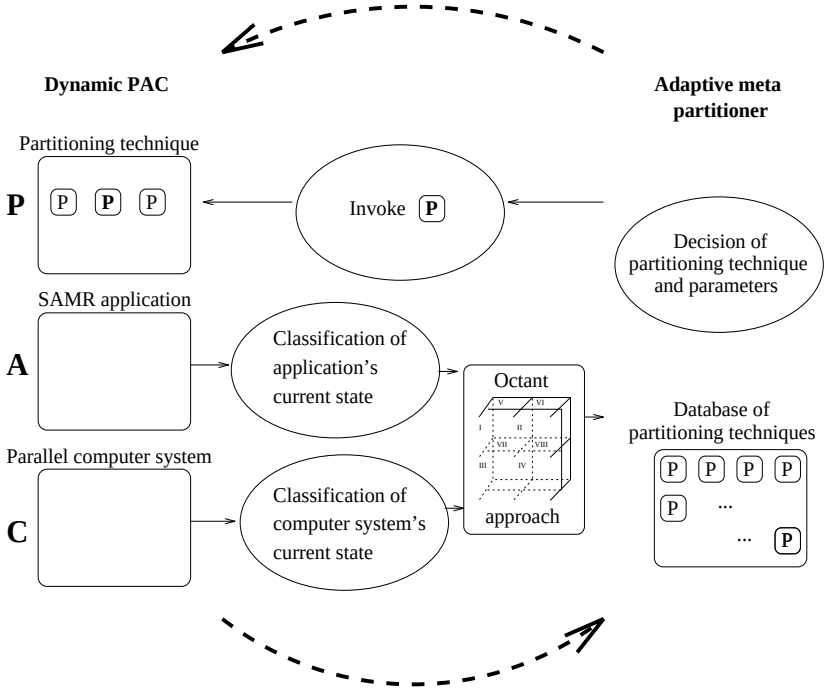

Figure 9: The adaptive meta-partitioner is fed at runtime with information about the current state of the (A)pplication and the (C)omputer system. On the basis of this information, it selects the appropriate (P)artitioning technique, hence forming a fully dynamic PAC.

The adaptive meta-partitioner for SAMR applications (illustrated in Fig. 9) is based on this idea. It enables the current state of the application and system to drive the selection of the partitioner at runtime. The selection is driven by a mapping from the application and system state to the partitioners. As shown in Fig. 9, the runtime environment is characterized using current application (A) and system (C) state, using the octant approach. The current octant is then used to select the most appropriate partitioning technique (P) from a database of available partitioning techniques. The selected partitioner is then configured with appropriate parameters such as partitioning granularity and threshold, and then invoked (using a common interface) to partition the SAMR grid hierarchy. This paper derives such a mapping based on an experimental characterization of partitioning techniques and application states [12, 43]. In this study, the (A) and (P) components are dynamic while the (C) component is assumed to be static.

5 SFC-Based Partitioners for SAMR Grid Hierarchies

This paper evaluates five different domain-based partitioning techniques for SAMR grid hierarchies that build on space-filling curve (SFC) representations of the problem domain. The first technique is the default partitioner in the *GrACE* [29, 30] distributed AMR infrastructure[2]. Techniques (2)-(5) are part of the parallel SAMR partitioning library *Vampire* [41], which combines structured and unstructured partitioning techniques as introduced by Rantakokko [36]. These techniques are summarized in Table 1.

[2]Note that GrACE allows users to select from a family of partitioners. It also allows users to provide their own partitioners.

The partitioning *granularity* for SFC-based partitioners may be varied. Granularity is determined by the *atomic unit*, i.e. the smallest entity the partitioner can "see" and work with. A small atomic unit (block dimension) can produce fine granularity partitioning with potentially good load balance. Nevertheless, the large number of sub-grids produced may increase the communication and synchronization overheads reducing overall performance. Using a large atomic unit results in a coarse granularity partitioning. In this case, the partitioner may not be able to achieve the best load balance. Furthermore, large atomic units may result in increased data migration costs. Communication/synchronization overheads however tend to decrease. Fig. 10 shows the effect of partitioning granularity on performance for the 2D numerical relativity application on 16 processors.

Scheme	Suite	Description
ISP	GrACE	Decomposition of recursive linear representation of a multi-dimensional grid hierarchy, generated using space-filling mappings
G-MISP	Vampire	Successive refinement of a one-vertex graph (matrix of workloads) by splitting vertices with total weight greater than a threshold
G-MISP+SP	Vampire	Similar to G-MISP, but uses sequence partitioning to assign internal blocks to processors
pBD-ISP	Vampire	Binary dissection of the domain into p partitions, with the orientation of the cuts determined by Hilbert space-filling curve
SP-ISP	Vampire	Dual-level parameterized binary dissection algorithm, with SFC-ordered one-dimensional list partitioned using sequence partitioning

Table 1: Summary of SFC-based partitioning techniques.

5.1 Space-Filling Curve Based Partitioning (ISP)

This partitioning scheme is based on the recursive linear representation of a multi-dimensional grid hierarchy generated using SFC's. In this scheme, the SAMR solution space is first partitioned into blocks of a minimum acceptable granularity dictated by the system and/or the application. The space-filling curve then passes through these blocks. As mentioned in Section 3.2, 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

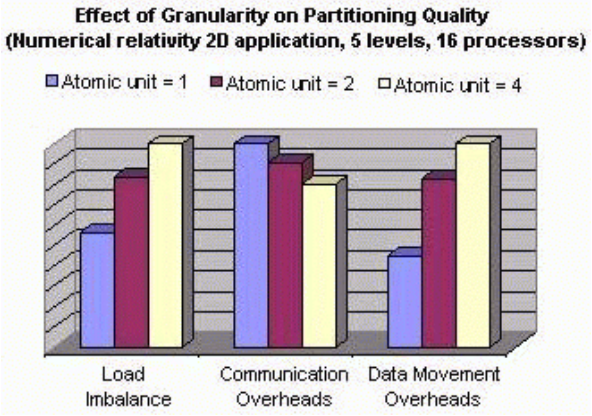

Figure 10: Effect of granularity on partitioning quality. Fine granularity enables good load balance but induces more communication and overhead. Conversely, coarse granularity promotes good communication patterns but prohibits good load balance.

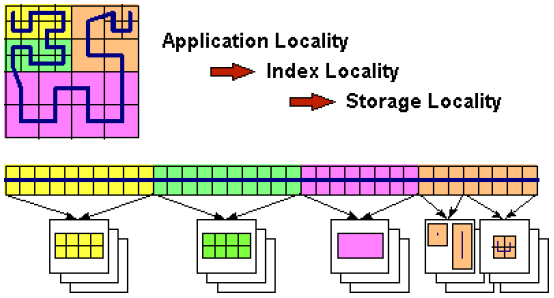

Figure 11: SFC-based partitioning of SAMR grid hierarchies.

within partitions of individual grids in the hierarchy. Space-filling mappings thus encode application domain locality and maintain this locality through expansion and contraction. Using the SFC-based linear representation, the problem of partitioning and dynamically repartitioning 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. 11 and is used by parallel SAMR infrastructures such as GrACE and Paramesh.

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

The G-MISP technique is a multilevel algorithm similar to SFC described above. It strives to create an list of blocks, each having approximately the same weight (workload). It starts

by creating a matrix of workloads stemming 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 where needed, by recursively splitting the matrix as long as the accumulated workload is greater than some threshold. Each cut is made at the geometrical mid point, and the orientation is determined by the Hilbert space-filling curve. As a result of this splitting, a one-dimensional list of blocks having approximately the same weight is created. The parameter "atomic unit" acts as an upper bound of the granularity. Therefore, the resulting partitioning granularity is a function of the atomic unit, the threshold, and the current state of the dynamic grid hierarchy. Load balance is trivially achieved by assigning the same number of blocks to each processor. The idea is to generate a granularity fine enough to make this strategy result in an acceptable load balance. The G-MISP technique (shown in Fig. 12) is designed for speed through simplicity (at the expense of load balance), and for giving acceptable communication patterns.

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

The G-MISP+SP is a variant of the G-MISP technique. It differs from the original G-MISP technique in that it uses *sequence partitioning*[3] (SP) [26] to assign the internal blocks to processors instead of simply assigning the same number of blocks to each processor. As a result, load balance improves, but the algorithm is more expensive. Apart from this last

[3]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.

load-balancing step, the G-MISP and G-MISP+SP techniques are identical.

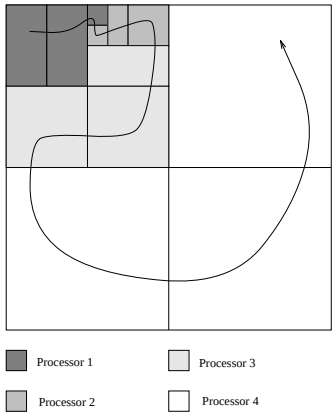

Figure 12: G-MISP: successive graph refinement.

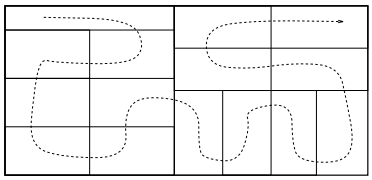

Figure 13: pBD-ISP: domain partitioned into p parts ($p = 15$) with approximately equal workloads.

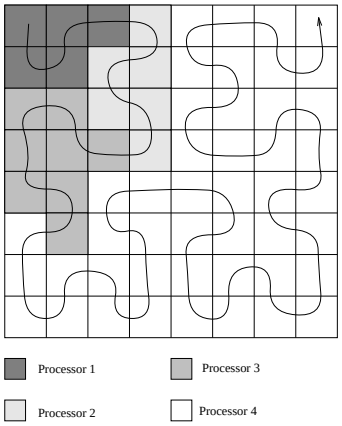

Figure 14: SP-ISP: domain subdivided into $p * b$ parts with approximately equal workloads. Sequence partitioning is then used to partition the resulting SFC list.

5.4 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 [4], 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. For example, if there are 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. 13. By dividing the domain recursively into exactly p parts, we gain speed, and get an extremely low granularity partitioning leading to low overhead costs and prospects of good communication patterns. Nevertheless, as the cuts are rectilinear, load balance might be a problem when the refinements are strongly localized.

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

In this scheme, the domain is divided into several equally-sized blocks by a generalization of the parameterized binary dissection (BD) algorithm described in [9]. 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 divided into b blocks. In the second level, all b blocks are further split into p blocks each, thereby resulting in the $p*b$ sub-division of the domain. Second, the orientation of the cuts is determined by the Hilbert SFC. For SAMR grid hierarchies, the blocks generated have different workloads due to grid adaptation. These blocks are ordered according to the SFC to form a one-dimensional list which is then partitioned using sequence partitioning [26]. SP-ISP (shown in Fig. 14) is potentially a fine granularity partitioning scheme leading to good load balance at the expense of more communication, overhead, and higher computational cost.

6 Experimental Characterization of Domain-Based Partitioning Techniques for SAMR Applications

Our overall research goal is to significantly decrease the runtime of large-scale parallel SAMR applications by introducing a dynamic meta-partitioner able to adapt to (and optimize for) fully dynamic PACs. In this paper, we move towards this goal by experimentally characterizing the overall behavior of each partitioning technique for a range of applications with different dynamic behaviors and partitioning requirements. This characterization is used to map partitioners to the set of application state octants (Section 4.2). This section describes the experimentation process and the applications, and presents the experimental results.

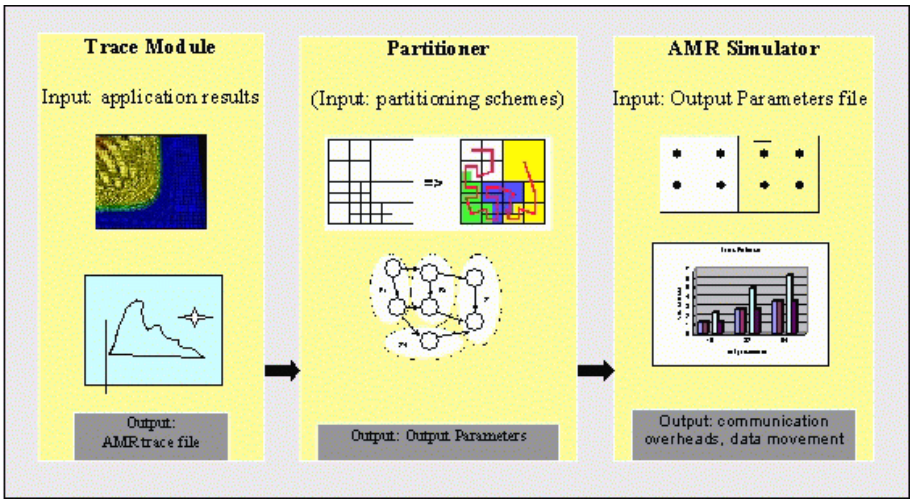

Figure 15: An overview of the experimentation process. An application refinement trace from a single processor run is used to drive the selected partitioner. The resulting partitioning is then evaluated by the AMR simulator.

The experiments presented in this section measure partitioning quality, using the five-component metric outlined in Section 4.1, for each of the applications and partitioning techniques on 64 processors. *Load imbalance* is measured as the load of the heaviest loaded processor divided by the average load. The final score is the average of these fractions taken over all time-steps. We subtract one to obtain the imbalance in percent. *Communication* is the sum of the amount of inter-processor communication taken over all time-steps. *Data migration* is the sum of the total number of data points forced to migrate as a result of re-partitioning, taken over all time-steps. *Overheads* are computed as the sum of the total number of patches, taken over all time-steps.

The evaluation is performed using an AMR simulator that simulates the execution of the Berger-Oliger AMR algorithm. The simulator is driven by an application execution trace obtained from a single processor run. This trace captures the state of the SAMR grid hierarchy for the application at the regrid (refinement and coarsening) step and is

independent of any partitioning. The simulator allows the user to select the partitioner to be used, the partitioning parameters (block size, granularity), and the number of processors. It then runs the trace and computes the performance of the partitioner at each regrid step using the metric outlined above. Information about the simulator can be obtained from [40]. The process is illustrated in Fig. 15.

6.1 The SAMR Applications

A suite of 7 "real-world" (2D and 3D) SAMR application kernels taken from varied scientific and engineering domains are used to evaluate and characterize the partitioners. These applications demonstrate different runtime behavior and adaptation patterns. Application domains include numerical relativity (Scalarwave 2D & 3D), oil reservoir simulations (Buckley-Leverette 2D & 3D), and computational fluid dynamics (compressible turbulence - RM 2D, and supersonic flows - EnoAMR 2D). Finally, we also use TportAMR 2D which is a simple benchmark kernel that solves the transport equation in 2D and is part of the GrACE distribution. The applications use 3 levels of factor 2 refinements in space and time. Regridding and redistribution is performed every 4 time-steps on each level. The applications are executed for 100 time-steps and the granularity (minimum block dimension) is varied between 1 and 2. The application kernels (summarized in Table 2) are described below.

The numerical relativity application (Scalarwave-2D and Scalarwave-3D) is a coupled set of partial differential equations. The equations can be divided into two classes: Elliptic (Laplace equation-like) constraint equations which must be satisfied at each time, and coupled Hyperbolic (Wave equation-like) equations describing time evolution. This kernel

addresses the Hyperbolic equations and is part of the Cactus numerical relativity toolkit [4].

The Buckley-Leverette model is used in Oil-Water Flow Simulation (OWFS) application for simulation of hydrocarbon pollution in aquifers. OWFS provides for layer-by-layer modeling of oil-water mixture in confined aquifers with regard to discharge/recharge, infiltration, interaction with surface water bodies and drainage systems, discharge into springs and leakage between layers. This kernel is taken from the IPARS reservoir simulation toolkit developed at the Center for Subsurface Modeling at the University of Texas at Austin [5]. The test suite includes 2D and 3D versions of this application.

RM is a 2D compressible turbulence application solving the Richtmyer-Meshkov instability. This application is part of the virtual test facility (VTF) developed at the ASCI/ASAP center at the California Institute of Technology[6]. The Richtmyer-Meshkov instability is a fingering instability which occurs at a material interface accelerated by a shock wave. This instability plays an important role in studies of supernova and inertial confinement fusion.

EnoAMR is a 2D computational fluid dynamics application that addresses the forward facing step problem, describing what happens when a step is instantaneously risen in a supersonic flow. The application has several features including bow shock, Mach stem, contact discontinuity, and a numerical boundary. EnoAMR2D is also a part of the virtual test facility developed at the ASCI/ASAP Center at Caltech.

Table 2 also summarizes the demands that each of the applications places on the partitioners. The effort required to optimize the different components of the metric is application

[4]Cactus Computation Toolkit - http://www.cactuscode.org

[5]IPARS: A New Generation Framework for Petroleum Reservoir Simulation - http://-www.ticam.utexas.edu/CSM/ACTI/ipars.html

[6]Center for Simulation of Dynamic Response of Materials - http://www.cacr.caltech.edu/ASAP/

Application	Domain	Dimension	LB	Comm	DM	OH
Scalarwave (SW)	Numerical relativity: Cactus toolkit, solving Einstein's and gravitational equations	SW-3D	a	+	a	a
		SW-2D	-	a	+	+
Buckley-Leverette (BL)	Oil reservoir simulation: IPARS test-bed for multiphase flow models and solvers	BL-3D	a	+	+	-
		BL-2D	a	-	a	+
Richtmyer-Meshkov (RM)	Computational fluid dynamics: compressible turbulence application solving RM instability	RM-2D	a	+	a	a
EnoAMR (Eno)	Computational fluid dynamics: forward facing step problem in supersonic flows	Eno-2D	+	+	-	+
TportAMR (Tport)	Transport equation solution: benchmark kernel included in the GrACE distribution	Tport-2D	a	a	a	+

Table 2: SAMR applications and their partitioning demands. A '+' means significantly tougher to optimize this metric than for other applications, a '-' means significantly easier, and 'a' is for average. 'LB'=Load-Balance, 'Comm'=communication, 'DM'=data movement, and 'OH'=overhead.

dependent. For example, due to application dynamics, EnoAMR-2D is significantly harder to load-balance as compared to Scalarwave-2D. In Table 2, each entry represents the relative effort required to optimize a metric (column) for an application (row). The entries are marked a '+' for large effort, a '-' for small effort, or an 'a' for average effort.

6.2 Experimental Results

An overview of the experimental evaluation of the partitioners using 64 processors is presented in Table 3. For each partitioning technique, this table indicates the behavior of each metric by grading it as '+' for good, as '-' for bad, or as 'a' for average or OK. All grades are relative to how hard it was on 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 value might be as low as 5 percent for another application. Detailed results are plotted in Fig. 16 for Scalarwave-3D and Scalarwave-2D, Fig. 17 for Buckley-Leverette-3D and Buckley-Leverette-2D, Fig. 18 for RM-2D and EnoAMR-2D and Fig. 19 for TportAMR-2D. Figures 17 through 19 are meant to reflect the relative characteristics of the partitioning techniques, displayed for each of the applications. Thus,

for each component in the five-component metric, values are normalized with respect to the largest value in this particular compnent. The performance of each partitioner is discussed below.

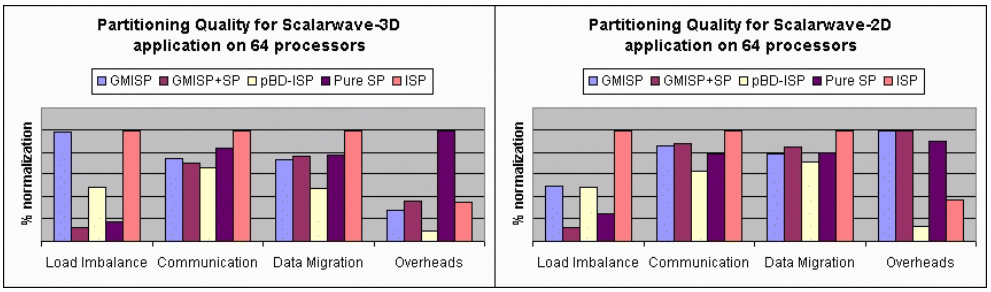

Figure 16: Partitioning quality for the Scalarwave 3D (left) and Scalarwave 2D (right) applications on 64 processors for each partitioning scheme.

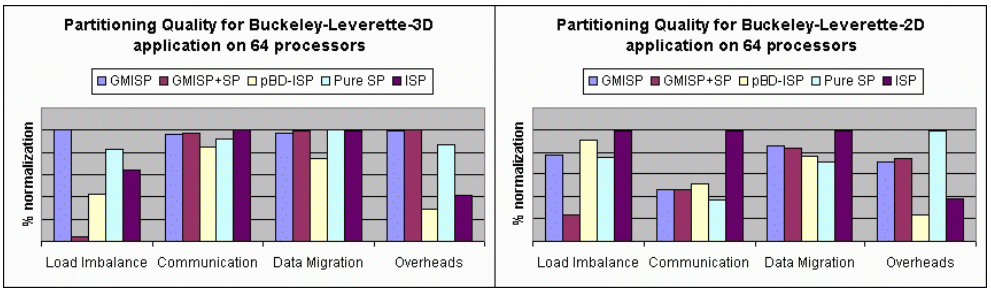

Figure 17: Partitioning quality for the Buckley-Leverette 3D (left) and Buckley-Leverette 2D (right) applications on 64 processors for each partitioning scheme.

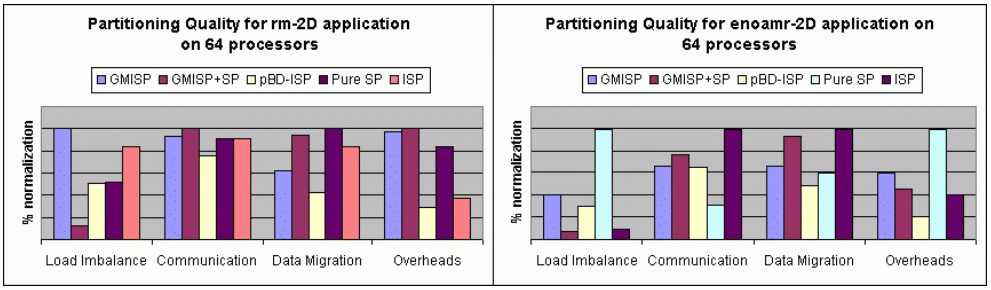

Figure 18: Partitioning quality for the RM 2D (left) and EnoAMR 2D (right) applications on 64 processors for each partitioning scheme.

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

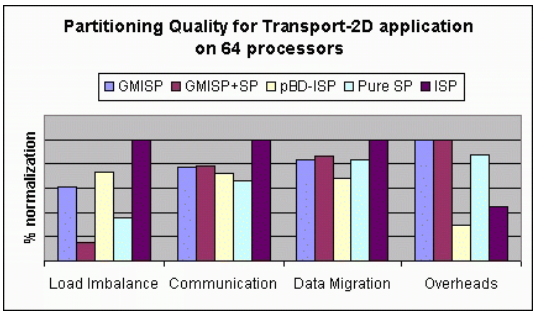

Figure 19: Partitioning quality for the TportAMR 2D application on 64 processors for each partitioning scheme.

Scheme	LB	Comm	DM	OH	Speed
G-MISP	-a-----	aaaaaaa	aaaaaaa	aaaaaaa	+
G-MISP+SP	+++a+a+	aaaaaaa	aaaaaaa	aaaaaaa	a
pBD-ISP	aaa-a--	+++a+aa	+a+a+a+	+++++++	+
SP-ISP	++--a-a	-aa+a+a	aaa+--a	-a--a-a	-
ISP	--a----	--a-a--	--aa-aa	a++++a+	+

Table 3: An overview of the experimental evaluation of the partitioners for 7 SAMR applications on 64 processors. For each application (1 through 7 listed in Table 2), the grade (+,a,-) indicates how well the partitioner succeeded in optimizing the particular metric. A '+' means significantly better than the other schemes, a '-' means significantly worse, and 'a' is for average or OK. 'LB'=Load-Balance, 'Comm'=communication, 'DM'=data movement, and 'OH'=overhead.

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 additional computational cost.

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

SP-ISP: The SP technique was not a great success. It is computationally more expensive 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, but struggles with load balance. Its behavior is similar to G-MISP, which is expected since the two schemes are very similar. Nevertheless, ISP favors the overhead metric at the expense of communication.

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

Table 4: Recommendations for mapping octants onto partitioning schemes.

7 An Application-Centric Characterization of the Partitioning Techniques

An application-centric characterization of the partitioners is obtained by first classifying the state of the SAMR application using the octant approach, and identifying the partitioning requirements of each octant (see Fig. 8). We then assign partitioner(s) to application state-octants based on their ability to meet the requirements of that octant using the experimental characterization presented in Section 6.

In the "far part" of the cube (octants V, VI, VII, VIII), solutions have high dynamics giving rise to large and rapid changes in the refinement. This results in increased data movement during regridding. Furthermore, partitioning speed is crucial as we need frequent repartitioning. Techniques such as pBD-ISP and even G-MISP+SP meet these requirements. In the "near part" of the cube (octants I, II, III, IV), changes in the refinement are more

gradual. Slower partitioners can be tolerated in these octants. Therefore, schemes such as G-MISP+SP are strong candidates here.

In the "upper part" of the cube (octants I, II, V, VI), communication dominates runtime. As a result, partitioners that optimize this metric (eg. pBD-ISP) are best suited. On the other hand, in the "lower part" (octants III, IV, VII, VIII), computation dominates, which suggests that schemes that optimize load balance (eg. G-MISP+SP or SP) are preferable.

In the "right part" of the cube (octants II, IV, VI, VIII), adaptation is scattered over the computational domain. This typically means more overhead. Both ISP and pBD-ISP optimize this metric well. Finally, in the "left part" of the cube (octants I, III, V, VII), adaptation is strongly localized and results in lesser data movement and overheads. Moreover, load balance is harder to achieve. Techniques such as G-MISP+SP, SP, or pBD-ISP are applicable in these octants.

The associations of application state octants to partitioning techniques are summarized in Table 4. To illustrate the association, let us assume an application that tracks an explosion, executing on a computer system with very low communication bandwidth. If the application has high activity dynamics, we will remain in "far part" of the cube (octants V - VIII). The explosion will give rise to many scattered clusters of adaptation, hence putting us in the "right part" of the cube (octants VI and VIII). Due to the low communication bandwidth, the runtime will probably be dominated by communication. Therefore, we end up in the "upper part" of the cube (octant VI). Our conclusion is that a good partitioner for this application on this machine would be pBD-ISP.

8 Conclusions

In this paper, we presented an application-centric experimental characterization of five dynamic domain-based inverse space-filling curve partitioning techniques for structured adaptive mesh-refinement applications using a suite of seven (2D and 3D) "real-world" application kernels. The overall goal was to obtain policies required to drive a *dynamically adaptive* meta-partitioner for SAMR grid hierarchies capable of decreasing overall execution time by selecting and configuring the most appropriate partitioning strategy (from a family of available partitioners) at runtime, based on current application and system state.

In the paper, we also proposed a five-component metric for evaluating partitioners for SAMR grid hierarchies and introduced an octant-based characterization of SAMR application state. The experimental results obtained were used to associate the appropriate partitioners to application state octants. The partitioners studied include new and existing techniques and constitute a selection from popular software tools such as GrACE and Vampire. Application kernels were taken from varied scientific and engineering domains, and exhibit very different runtime behavior. We are currently investigating the design of automated recommender systems for partitioning techniques. Such a system (which includes the characterization of system state) can suggest the choice of partitioner to be used based on the application, its input parameters, and the current runtime state.

Acknowledgements

The authors would like to thank Bruce Hendrickson and the referees for their valuable suggestions, which have greatly influenced the current version of the paper. The authors are grateful to Michael Thuné and Jarmo Rantakokko for their support, and Ravi Samtaney, Marco Arienti, Michael

Edwards, and Mijan Huq for making the applications available for use.

References

[1] ASCI/ASAP Center, http://www.cacr.caltech.edu/ASAP, California Institute of Technology.

[2] ASCI Alliance, http://www.llnl.gov/asci-alliances/asci-chicago.html, University of Chicago.

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

[4] M. Berger and S. Bokhari, "A Partitioning Strategy for Non-uniform Problems on Multiprocessors", *IEEE Transactions on Computers*, 85:570-580, 1987.

[5] M. Berger and P. Colella, "Local Adaptive Mesh Refinement for Shock Hydrodynamics", *Journal of Computational Physics*, vol. 82, pp. 64-84, 1989.

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

[7] M. Berger and J. Oliger, "Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations", *Journal of Computational Physics*, vol. 53, pp. 484-512, 1984.

[8] T. Bially, "A Class of Dimension Changing Mappings and its Application to Bandwidth Compression", Ph.D. Thesis, Polytechnic Institute of Brooklyn, 1967.

[9] S. Bokhari, T. Crockett, and D. Nicol, "Binary Dissection: Variants & Applications", Technical Report ICASE Report No. 97-29, NASA Langley Research Center, Hampton, VA, 1997.

[10] G. Bryan, "Fluids in the Universe: Adaptive Mesh Refinement in Cosmology", *Computing in Science & Engineering*, pp. 46-53, March-April 1999.

[11] A. Calder, B. Curtis, and et al, "High Performance Reactive Fluid Flow Simulations Using Adaptive Mesh Refinement on Thousands of Processors", *Supercomputing Conference*, 2000.

[12] S. Chandra and M. Parashar, "An Evaluation of Partitioners for Parallel SAMR Applications", *Euro-Par 2001*, Springer-Verlag Lecture Notes in Computer Science, editors: R. Sakellariou, J. Keane, J. Gurd, and L. Freeman, Vol. 2150, pp. 171-174, August 2001.

[13] S. Chandra, J. Steensland, M. Parashar, and J. Cummings, "An Experimental Study of Adaptive Application Sensitive Partitioning Strategies for SAMR Applications", *2nd Los Alamos Computer Science Institute Symposium*, October 2001.

[14] M. Choptuik, "Experiences with an Adaptive Mesh Refinement Algorithm in Numerical Relativity", *Frontiers in Numerical Relativity*, Cambridge University Press(London), editors: C. Evans, L. Finn, and D. Hobill, pp. 206-221, 1989.

[15] C. Edwards, "A Parallel Infrastructure for Scalable Adaptive Finite Element Methods and its Application to Least Squares C-infinity Collocation", Ph.D. Thesis, University of Texas at Austin, May 1997.

[16] S. Fink, S. Baden, and S. Kohn, "Flexible Communication Mechanisms for Dynamic Structured Applications", *IRREGULAR*, 1996.

[17] M. Griebel and G. Zumbusch, "Hash-Storage Techniques for Adaptive Multilevel Solvers and their Domain Decomposition Parallelization", *Domain Decomposition Methods 10*, editors: J. Mandel, C. Farhat, X. Cai, Contemporary Mathematics 218, pp. 279-286, AMS, 1998.

[18] M. Griebel and G. Zumbusch, "Parallel Multigrid in an Adaptive PDE Solver based on Hashing", *ParCo '97*, editors: E. D'Hollander, G. Joubert, F. Peters, U. Trottenberg, Elsevier, pp. 589-599, 1998.

[19] S. Hawley and M. Choptuik, "Boson Stars Driven to the Brink of Black Hole Formation", Phys. Rev. D 62:104024, 2000.

[20] S. Kohn, http://www.llnl.gov/CASC/SAMRAI, SAMRAI: Structured Adaptive Mesh Refinement Applications Infrastructure, 1999.

[21] S. Kohn and S. Baden, "Parallel Software Abstractions for Structured Adaptive Mesh Methods", *Journal of Parallel and Distributed Computing*, 61, (6), pp. 713-736, 2001.

[22] P. MacNeice, http://esdcd.gsfc.nasa.gov/ESS/macneice/paramesh/paramesh.html, Paramesh homepage, 1999.

[23] B. Moon, H. Jagadish, C. Faloutsos, and J. Saltz, "Analysis of the Clustering Properties of Hilbert Space-filling Curve", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 13, No. 1, February 2001. Also, University of Maryland Department of Computer Science Technical Report, CS-TR-3611 and UMIACS-TR-96-20, March 1996.

[24] M. Norman and G. Bryan, "Cosmological Adaptive Mesh Refinement", *Numerical Astrophysics 1998*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999.

[25] L. Oliker and R. Biswas, "PLUM: Parallel Load Balancing for Adaptive Unstructured Meshes", *Journal of Parallel and Distributed Computing*, Vol. 52, No. 2, pp. 150-177, 1998.

[26] B. Olstad and F. Manne, "Efficient Partitioning of Sequences", Technical Report, Dept. of Computer Systems and Telematics, The Norweigian Institute of Technology.

[27] C. Ou and S. Ranka, "Parallel Remapping Algorithms for Adaptive Problems", *Journal of Parallel and Distributed Computing*, 42, pp. 109-121, 1997.

[28] M. Parashar and J. Browne, "On Partitioning Dynamic Adaptive Grid Hierarchies", *29th Annual Hawaii International Conference on System Sciences*, pp. 604-613, January 1996.

[29] M. Parashar, J. Browne, C. Edwards, and K. Klimkowski, "A Common Data Management Infrastructure for Adaptive Algorithms for PDE Solutions", Technical paper at *Supercomputing Conference*, 1997.

[30] M. Parashar and J. Browne, "System Engineering for High Performance Computing Software: The HDDA/DAGH Infrastructure for Implementation of Parallel Structured Adaptive Mesh Refinement", *IMA Volume on Structured Adaptive Mesh Refinement (SAMR) Grid Methods*, editors: S. Baden, N. Chrisochoides, D. Gannon, and M. Norman, Springer-Verlag, pp. 1-18, January 2000.

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

[32] A. Patra, J. Long, and A. Laszloffy, "AFEAPI: A Simple Infrastructure for Parallel Adaptive hp FEM Using Self-Organizing Data and Computations", 1st Workshop on *Parallel Computing for Irregular Applications* held in conjunction with High Performance Computing Architectures-5, Orlando, FL, January 1999.

[33] R. Pember, J. Bell, P. Colella, W. Crutchfield, and M. Welcome, "Adaptive Cartesian Grid Methods for Representing Geometry in Inviscid Compressible Flow", *11th AIAA Computational Fluid Dynamics Conference*, Orlando, FL, July 6-9, 1993.

[34] J. Pilkington and S. Baden, "Dynamic Partitioning of Non-Uniform Structured Workloads with Spacefilling Curves", *IEEE Transactions on Parallel and Distributed Systems*, 7, (3), 288-300, 1996.

[35] J. Pilkington and S. Baden, "Partitioning with Space-filling Curves", Technical Report CS94-34, University of California, San Diego, 1994.

[36] J. Rantakokko, "Data Partitioning Methods and Parallel Block-oriented PDE Solvers", Ph.D. Thesis, Uppsala University, Sweden, 1998.

[37] H. Sagan, "Space-filling Curves", Springer-Verlag, 1994.

[38] J. Saltzman, M. Berger, J. Bell, and M. Welcome, "Three Dimensional Adaptive Mesh Refinement for Hyperbolic Conservation Laws", *SIAM Journal of Scientific Computing*, vol. 15, pp. 127-138, 1994.

[39] H. Samet, "The Design and Analysis of Spatial Data Structures", Addison-Wesley Publishing Company, Reading, MA, 1989.

[40] M. Shee, "Evaluation and Optimization of Load Balancing/Distribution Techniques for Adaptive Grid Hierarchies", M.S. Thesis, Graduate School, Rutgers University, NJ, 2000.

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

[42] J. Steensland, "Domain-based Partitioning for Parallel SAMR Applications", Licentiate Thesis, 2001-002, Dept. of Scientific Computing, Uppsala University, Sweden, 2001.

[43] 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*, Las Vegas, NV, pp. 425-430, November 2000.

[44] J. Steensland, S. Soderberg, and M. Thuné, "A Comparison of Partitioning Schemes for Blockwise Parallel SAMR Applications", Springer-Verlag, 2000.

[45] E. Steinthorsson and D. Modiano, "Advanced Methodology for Simulation of Complex Flows using Structured Grid Systems", *ICOMP*, 28, 1995.

[46] M. Thuné, "Partitioning Strategies for Composite Grids", *Parallel Algorithms and Applications*, 1997.

[47] N. Touheed, P. Selwood, P. Jimack, and M. Berzins, "A Comparison of Some Dynamic Load-Balancing Algorithms for a Parallel Adaptive Flow Solver", *Journal of Parallel Computing*, vol. 26, pp. 1535-1554, 2000.

[48] C. Walshaw, M. Cross, and M. Everett, "Parallel Dynamic Graph Partitioning for Adaptive Unstructured Meshes", *Journal of Parallel and Distributed Computing*, vol. 47, pp. 102-108, 1997.

[49] M. Warren and J. Salmon, "A Parallel Hashed Oct-Tree N-body Algorithm", *Supercomputing Conference*, pp. 12-21, 1993.

[50] P. Wang, I. Yotov, T. Arbogast, C. Dawson, M. Parashar, and K. Sepehrnoori, "A New Generation EOS Compositional Reservoir Simulator: Part I - Formulation and Discretization", *Society of Petroleum Engineering Reservoir Simulation Symposium*, Dallas, TX, June 1997.