SANDIA REPORT

SAND2003-8725
Unlimited Release
Printed March 2004

A Partitioner-Centric Model for SAMR Partitioning Trade-Off Optimization: Part II

Johan Steensland and Jaideep Ray

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation,
a Lockheed Martin Company, for the United States Department of Energy's
National Nuclear Security Administration under Contract DE-AC04-94AL85000.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865)576-8401
Facsimile: (865)576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.doe.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800)553-6847
Facsimile: (703)605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2003-8725
Unlimited Release
Printed March 2004

A Partitioner-Centric Model for SAMR Partitioning Trade-Off Optimization: Part II

Johan Steensland and Jaideep Ray
Advanced Software Research and Development
Sandia National Laboratory
P.O. Box 969
Livermore, CA 94550-9915, USA

Abstract

Optimal partitioning of structured adaptive mesh applications necessitates dynamically determining and optimizing for the most time-inhibiting factor, such as data migration and communication volume. However, a trivial monitoring of an application evaluates the current partitioning rather than the inherent properties of the grid hierarchy. We present a model that given a structured adaptive grid, determines *ab initio* to what extent the partitioner should focus on reducing the amount of data migration to reduce execution time. This model contributes to the meta-partitioner, our ultimate aim of being able to select and configure the optimal partitioner based on the dynamic properties of the grid hierarchy and the computer. We validate the predictions of this model by comparing them with actual measurements (via traces) from four different adaptive simulations. The results show that the proposed model generally captures the inherent optimization-need in SAMR applications. We conclude that our model is a useful contribution, since tracking and adapting to the dynamic behavior of such applications lead to potentially large decreases in execution times.

Acknowledgments

The authors thank Manish Parashar and Sumir Chandra at the Center for Advanced Information Processing, Rutgers University, NJ, USA, and Michael Thuné, Jarmo Rantakokko, and Henrik Johansson at Information Technology, Uppsala University, Sweden for scientific collaboration. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94-AL85000.

Contents

This page intentionally blank

1 Introduction and Background

Significantly improving the scalability of large structured adaptive mesh refinement (SAMR) applications requires sophisticated capabilities for using the underlying parallel computer's resources in the most efficient way. The *meta-partitioner* [35, 36] is a tool providing such capabilities. Previous research has offered design, proofs-of-concept and evaluation of major components.

Part I of this research [39] introduced a model for classifying SAMR application and parallel computer system state to best provide a partitioner with information required for trade-off optimization, and validated this model by showing its effectiveness to accurately capture the dynamic behavior of SAMR applications. Particular attention was paid to dimension I in the classification space (characterizing the relative importance of achieving a good load-balance versus reducing the overall communication cost) and the Richtmyer-Meshkov instabilities simulation.

The present paper completes the theoretical part of this research by presenting a model for dimension III in the classification space (characterizing the need for optimizing for data migration) and experimentally validating this model. This paper also lays the theoretical foundation for dimension II (determining the trade-off between partitioning speed vs. overall quality) and demonstrates the dynamic behavior of the Buckley-Leverette SAMR application.

The presented work is part of the ongoing research project [35, 36, 37, 10, 38, 39] with the overall goal of the *dynamically adaptive* meta-partitioner for SAMR grid hierarchies capable of selecting the most appropriate partitioning strategy at run-time, based on current system and application state. Such a meta-partitioner can significantly reduce the execution time of SAMR applications [13, 12, 11].

Dynamically adaptive mesh refinement (AMR) [41] methods for the numerical solution to partial differential equations (PDE's) [7, 8, 31] employ locally optimal approximations, and can yield highly advantageous ratios for cost/accuracy when compared to methods based on a static uniform mesh. These techniques seek to improve the accuracy of the solution by dynamically refining regions with large solution error. Structured adaptive mesh refinement methods are based on uniform patch-based refinements overlaid on a structured coarse grid and provide an alternative to the general, unstructured AMR approach. Methods based on SAMR can lead to computationally efficient implementations as they require uniform operations on regular arrays and exhibit structured communication patterns. Due to their regular structure, these methods tend to be easier to implement and manage.

The primary motivation for our research is that *no single partitioning scheme performs the best* for all types of SAMR applications and systems. For a given application, the most suitable partitioning technique depends on input parameters and the application's run-time state [29, 36]. Adaptive management of these dynamic applications at run-time is necessary. This includes using application run-time state to select and configure the partitioning strategy to maximize performance. The goal of the adaptive *meta-partitioner* is to provide such a capability for parallel SAMR applications.

Large scale SAMR applications place vastly different requirements on the partitioning strategy to enable efficient use of computer resources and consequently good scalability. Sometimes this requires focusing on optimizing load balance; and sometimes on lowering the interprocessor communication costs [38] or the data migration. A means to classify these requirements in a way that conforms to the partitioner is crucial.

The support for tuning and choosing trade-off impacts maturing in graph-based partitioning techniques [17, 33, 15] for *unstructured* AMR, is so far lacking in the field of *structured* AMR. Whereas recent research efforts have targeted the scalability of *specific* applications executing on *specific* parallel computers [5, 46, 27], our line of research is in the opposite direction; the development of a *general* partitioning tool enabling good scalability for *general* SAMR applications executing on *general* parallel computers. We engineer the components of the adaptive meta-partitioner with the requirement that they are able to adapt to changing requirements derived from the monitoring of system and application state.

In this research, we advance towards the meta-partitioner by introducing a key component: a model for the classification of application and system state. The key contributions of this paper are (1) a model for sampling and translating these samples of the given application parameters (such as the grid hier-

archy) and system parameters (such as CPU speed and communication bandwidth) into dimension III of the partitioner-centric classification space, and (2) an experimental evaluation and validation of this model showing its effectiveness to accurately capture the dynamic behavior of four vastly different SAMR applications, (3) grid-relative metrics allowing for inter-application comparisons of data migration and communication, (4) a theoretical model for dimension II, including the (so-far-little-researched) problem of determining the relative importance of an absolute metric-component value, and (5) complementary communication results for dimension I using the new metric.

2 SAMR and Related Work

2.1 Introduction to SAMR

Structured adaptive mesh methods are being widely used for adaptive PDE solutions in many domains, including computational fluid dynamics [2, 6, 28], numerical relativity [14, 30], astrophysics [1, 9, 23], and subsurface modeling and oil reservoir simulation [45, 25].

Dynamic adaptation is achieved by tracking regions in the domain that require higher resolution and dynamically overlaying finer grids on these regions. These methods start with a 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 structure is a dynamic adaptive grid hierarchy.

Some of the software infrastructures for SAMR are Paramesh [21, 22], a FORTRAN library for parallelization of and adding adaption to existing serial structured grid computations, SAMRAI [18, 46] a C++ object-oriented framework for implementing parallel structured adaptive mesh refinement simulations and GrACE [26] and CHOMBO[3], both of which are adaptive computational and data-management engines for enabling distributed adaptive mesh-refinement computations on structured grids.

2.2 Partitioning SAMR Grid Hierarchies

Parallel implementations of SAMR methods offer the potential for accurate solutions of physically realistic models of complex physical phenomena. However, they present interesting challenges in dynamic resource allocation, data-distribution, load-balancing, and run-time management. The overall efficiency of parallel SAMR applications is limited by the ability to partition the underlying grid hierarchies at run-time to expose all inherent parallelism, minimize communication and synchronization overheads, and balance load. A critical requirement for the partitioning is the maintenance of logical locality, both across different levels of the hierarchy under expansion and contraction of the adaptive grid structure, and within partitions of grids at all levels when they are decomposed and mapped across processors. The former enables efficient computational access to the grids and minimizes the parent-child (inter-level) communication overheads, while the latter minimizes overall communication and synchronization overheads. Furthermore, application adaptation results in grids being dynamically created, moved and deleted at run-time, making it necessary to efficiently repartition the hierarchy "on the fly" so that it continues to meet these goals.

Classification and comparative studies for *unstructured*-grid/mesh partitioning and dynamic load-balancing appear in the literature [43, 44]. For *structured* grid hierarchies, partitioners can be classified as patch-based, domain-based, or hybrid.

For *patch-based partitioners* [5, 19], distribution decisions are independently 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. Grids may also be distributed uniformly over all processors. The SAMR framework SAMRAI [18, 46] (based on the LPARX [4] and KeLP [16] model) supports patch-based partitioning. The distribution scheme maps the patches at a refinement level of the AMR hierarchy across processors. The advantages are manageable load imbalance and that re-partitioning at re-griding could be avoided. Shortcomings inherent in patch-based techniques are communication serialization bottlenecks, inability to exploit available parallelism both across grids at the same level and different levels [36].

Domain-based partitioners [24, 29, 42, 35] partition the physical domain, rather than the grids themselves. The domain is partitioned along with all contained grids on all refinement levels. The advantages are elimination of inter-level communication and better exploiting of all available parallelism. The disadvantages are intractable load imbalance for deep hierarchies and the occurrence of "bad cuts" leading to increased overhead costs [36].

Hybrid partitioners [24, 42, 20] combining patch-based and domain-based approaches, can be used for coping with the shortcomings present in these techniques. They typically use a 2-step 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.

Developed at Uppsala University, Sweden and Rutgers University, New Jersey, USA, `Nature+Fable` (**Natur**al **Re**gions + **Fa**ctional blocking and **bi-le**vel partitioning) [36] hosts a variety of hybrid partitioning options for partitioning SAMR grid hierarchies. All involved parts are engineered as components of the meta-partitioner. They offer parameters to steer component behavior enabling adaptation to varying partitioning requirements. As `Nature+Fable` matures, it is intended to transform into the meta-partitioner.

`Nature+Fable` separates homogeneous, unrefined (Hue) and complex, refined (Core) domains of the grid hierarchy and clusters refinement levels into *bi-levels* [36]. The Hues contain the portions of the grid hierarchy without refinements; consequently they contain only parts of the base grid (refinement level 0). The Cores contain the portions of the grid where refinements are present. The Cores are separated from the Hues in a strictly domain-based fashion, meaning that each Core contains a portion of the base grid and all its overlaid, refined grids. Expert blocking algorithms are used for the Hues. The Cores are subjected to a coarse partitioning, creating "easy-to-block" bi-levels. Then the same expert algorithms operating on the Hues are re-used for these bi-levels.

3 Previous Approaches

This section recapitulates from Part I and provides a problem description and a survey of relevant previous research efforts including the *octant approach* [36] and the *ArMADA framework* [13]. While conceptually similar to the octant approach in that it strives to capture the application and system state for optimizing the partitioning, our model is quantitative and is rigorously derived from a set of assumptions, primarily borne out by observations in SAMR simulations.

The PAC-triple defines the partitioner (P), the SAMR application (A), and the parallel computer system (C). While the A and C components are highly dynamic entities, the P component is usually selected once and for all. A static P component, i.e., not exploiting the dynamic nature of the A and C components, seriously inhibit the potential for increasing scalability and reducing execution time. The octant approach and the meta-partitioner are means for allowing fully dynamic PACs, i.e. , it enables

$$P_t = f(A_t, C_t),$$

where the partitioner P at a particular time t should be a function of the application A and computer system C, both of which are functions of time t.

To illustrate the dynamic behavior of SAMR applications, consider a static choice of P for the BL2D application (further described in section 5) illustrated in Figure 1. This figure plots load imbalance and communication amount as a function of time. Clearly, with a dynamic selection of P (a fully dynamic PAC) appropriately reflecting the inherent dynamics of the application, the total execution time could have been reduced.

The octant approach is depicted in Figure 3 (left). It is an extension of the quadrant approach [40] and constitutes an important part of the conceptual meta-partitioner, illustrated in Figure 2. The octant approach is a discrete classification space and a set of rules for selecting and configuring the most appropriate partitioning technique, based on application and system state. The model consists of the following: (a) classifying application state, (b) classifying system state, (c) combining the results in (a) and (b), (d) translating the result in (c) into an octant, and finally (e) mapping from octant onto partitioning technique.

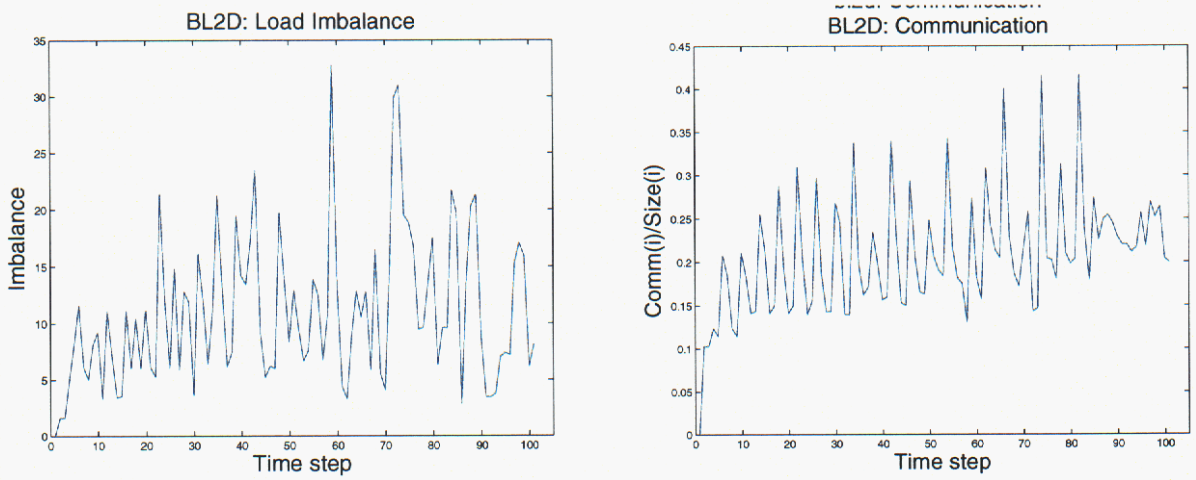

Figure 1: The dynamic behavior of the BL2D SAMR application illustrated by a static selection of partitioner (P). Clearly, with a dynamic selection of P (a fully dynamic PAC) appropriately reflecting the inherent dynamics of the application, the total execution time could have been reduced.

This model has evolved over the years to reflect and incorporate our growing intuition and experience. It requires determining the classification space, i.e., the actual axes in the cube (of octants), and the characterization of involved partitioners.

However, the model does not say *how* — only *what* and *why*; it outlines what seems to be a promising concept. For example, it does not define a "state". It does not offer ways to compute it, nor does it list the necessary variables. It does not define how to translate the information regarding state into an octant. The model in itself does not provide the specifics about the mapping from octant onto partitioning technique, but such a mapping has been derived for a set of partitioners [37].

The ArMADA framework offers a first attempt at an actual implementation of the model. ArMADA disregards the system component and uses simple box operations like e.g. volume-to-surface ratio on the grid hierarchy to determine the corresponding octant. The classification is relative to the previous state (octant) and the mappings used were those previously derived [37] . The project provided an important proof of concept: even with such a simple model, execution times were reduced.

In the following, we examine each of the dimensions of the classification space in the octant approach in detail. The original thought was to capture the current state of the application executing on the system, with the selection and configuration of the best partitioning technique determined by this state. We show that for the purpose, this space is inadequate.

3.1 The Refinement Pattern

The first dimension of the cube, *localized — scattered*, reflects the nature of the refinement pattern. For unstructured meshes, diffusion schemes are suited to scattered patterns while scratch/re-map work well for strongly localized patterns [32]. But how does this pattern affect partitioning of structured grids?

Assuming a strictly domain-based partitioning technique, the refinement pattern is crucial. A small base-grid, many processors, and many levels of refinement cause domain-based techniques to generate intractable amounts of load imbalance. However, the case improves with scattered refinement, and worsens with strongly localized refinement. Consequently, for an already strained domain-based, badly load-balanced scenario, the refinement pattern is crucial to sample. In other cases, it might be of little consequence.

3.2 Time Domination

The second dimension of the cube, *computation dominated — communication dominated*, reflects whether run-time of an application (at a given moment) is dominated by communication or by computation. The

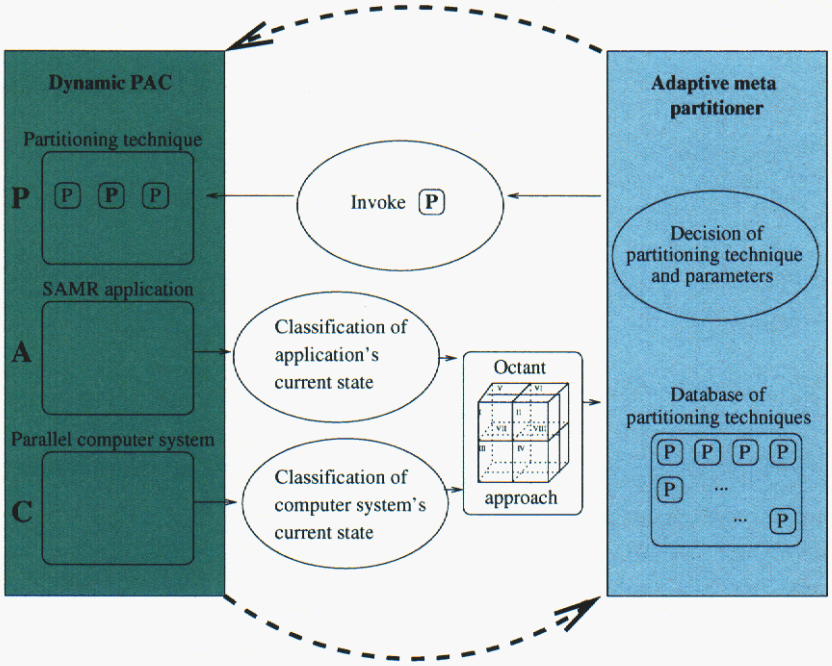

Figure 2: The conceptual meta-partitioner incorporating the octant approach. The most appropriate partitioner is selected and configured, based on the current application and system state. Consequently, fully dynamic PAC:s are enabled.

idea is to distinguish between scenarios where load balance should be the target for optimization (computation dominated) and where communication pattern/amount (communication dominated) should be the main target.

This is problematic. First, the other axes in the cube (refinement pattern and activity dynamics) reflect the state of the application and system *independently* of partitioning technique and current distribution of the hierarchy. It is impossible to determine the time domination axis without involving assumptions about how the grid is distributed.

This dependency accounts for a "circle" in the model. We wish to classify the state of the application / system to select the partitioner. But this classification is strongly influenced by the partitioner used to achieve the current domain decomposition. In the worst case, the partitioner is classifying itself and the answer cannot be regarded very general. For example, assume the classification of the application / system is "communication dominated". This means that *this* particular partitioner generated lots of communication for the application / system. Moreover, it is a classification of a *PAC-triple*, while the other dimensions in the cube classify the AC-double.

Furthermore, the original idea of interpreting a computation dominated simulation as one requiring an optimizing of load balance is flawed. A parallel SAMR application consists of a series of steps separated by synchronization points. Load imbalance among processors dictates the cost of (time spent at) these synchronizations. But this load imbalance is the *total* load imbalance, i.e. it is the sum of computational load imbalance and communicational load imbalance. Assume for example a perfectly (computationally) load-balanced application with a localized refinement. When the processors involved with the localized refinement start communicating, they will take more time than their counterparts working on the non-refined parts of the hierarchy. As a consequence, there will be lots of load imbalance (lots of time spent waiting at synchronization points) despite a perfect computational load balance.

To summarize the complexity of this dimension:

1. An application spending significant time at synchronization points is badly load balanced.

Based on this information only, it is impossible to determine whether optimizing (computational) load balance or communication would be the best remedy.

2. Synchronization points are implemented by variants of `MPI_Wait`. Thus, they are part of (global) communication. This means that a communication-dominated application might be in great need of optimizing load balance.

We conclude that this dimension must be reformulated so as to explicitly expose the cause of the load imbalance, i.e., whether the imbalance is due to unbalanced load distribution or unbalanced communication schedules. There are ways to steer the trade-off communication/load imbalance in tools like `Nature+Fable` . We must find a suitable model for giving these tools pertinent information to base their trade-offs on.

3.3 Activity Dynamics

The third dimension reflects how fast things are *changing* in the solution. The idea is that high activity dynamics implies more frequent regridding. Consequently, partitioning speed and low data migration costs are crucial. This is not necessarily true.

An SAMR application might have a very long compute cycle. There may be minutes between synchronization points and possible re-gridding. Such an application might exhibit significant activity dynamics from time-step to time-step. However, if each time-step is computationally expensive (i.e. takes minutes to perform) a fairly expensive partitioner and significant data migration costs can be justified. Thus, frequent partitioning does not automatically indicate a "cheap" partitioner — one needs to compare the partitioning and data migration costs with the time between successive repartitioning.

4 A New Model

This section derives a new continuous classification space and a model to characterize the "state" of a SAMR application. The section starts by investigating the fundamental requirements and suitable properties of such a classification space. Based on these requirements and properties, it proceeds with the designing of this space. Finally, it derives the methods for sampling application and system state and mapping this state onto the proposed classification space.

What the Partitioner Needs

This sub-section recapitulates from Part I and discusses the fundamentals of grid hierarchy partitioning. Independent of partitioning approach, any serious partitioner should either (a) exhibit a strong preference for optimizing a specific metric or (b) have parameters that allow one to bias its behavior so as to better optimize a given metric. Partitioners are generally confronted with the same problem-specific trade-offs and merely use different algorithms to achieve them. As a consequence, the classification space should be constructed to conform to these "universal truths" of SAMR partitioners — only then can the properties of the application and system be fully exploited.

A sophisticated partitioning tool offers various parameters for influencing the outcome. Most prominent is the trade-off between communication costs and load balance. For example, to focus on load balance in `Nature+Fable` we may choose a small *atomic unit*, select a large Q, choose *fractional blocking* and so forth. Working with other partitioners, we might migrate from domain-based techniques toward more elaborate patch-based techniques specializing in optimizing load balance.

Furthermore, it is a fairly straight-forward idea to trade-off overall quality for speed. For example, there are SAMR applications that compute for many minutes between synchronization and re-griding. For these applications, it would make sense to spend more than fractions of a second in order to generate a more high-quality partitioning.

The third and last important trade-off occurs when there is need to optimize the amount of data migration. As opposed to the two trade-offs above, there is no unique or apparent trade-off to optimizing data migration. Depending on the current partitioning strategy and the state of the grid hierarchy, optimizing data migration may be obtained by e.g. invoking some kind of post mapping technique or switching methods to a more "diffusion-like" one, or investing more time in creating a more fully ordered SFC mapping. Depending on the circumstances, any of the metrics load imbalance, communication, speed or overall remaining quality might suffer. Note that the optimal

amount of data migration is zero, translating to keeping all data where currently allocated. The trade-off for inducing longer waits between re-partitioning is the penalty of keeping the same partitioning during this time. Hence, attacking data migration this way trades-off whatever shortcomings the current partitioning is suffering from.

These are the three major and general trade-off possibilities, and it is imperative that *any* classification model (in the present context) should expose these fairly explicitly.

In view of the above, we propose that the partitioner-centric classification space should host exactly these three dimensions: (1) Communication versus load balance, (2) Speed versus overall quality, and (3) Data migration.

Illustrated in Figure 3 (right), this partitioner-centric classification space is obviously quite different from the octant approach. Moreover, we propose that the classification space is *absolute* and *continuous*, as opposed to *relative* and *discrete* in the ARMaDA framework. Consequently, a state sampling will generate a mapping onto a point defined in a continuous coordinate space within the classification space. The locus of all such points, as a simulation evolves, will be a curve in the same space. Unlike ARMaDA, one does not discretely transition between octants, but rather follow a smooth curve. This enables not only a coarse grained partitioner selection, but also an extremely fine grained partitioner configuration.

Part I of this work presented the theory for Trade-Off 1: Load Balance vs Communication. The following sub-sections will first present new, grid-relative metrics and discuss the absolute importance of relative metrics. Then, the theory for using Trade-Off 2 and Trade-Off 3 with any SAMR application is presented.

4.1 Grid-Relative Metrics

Measuring load imbalance in percent (the load of the heaviest loaded processor divided by the average load) has become the da facto standard. It allows for comparisons, both for different partitioners for a particular application, and for different application with the same partitioner.

Similar metrics for data migration and communication are so far lacking. These quantities are often presented in absolute terms, e.g. as number of data words transmitted, number of "packages", or number of transmissions. While this allows for rudimentary analysis, it lacks the flexibility of the relative load imbalance metric.

Our SAMR simulator outputs the number of grid points transmitted for a certain coarse time step and we propose to normalize this data with the regards to the grid at this particular time-step. Data migration between time-steps $t-1$ and t should be normalized with respect to *grid size*, i.e. the number of grid points, in the grid hierarchy at time-step $t-1$. Consequently, a 100-percent data migration translates to that all points in the grid are moved. Communication should be normalized with respect to *work load*. A 100-percent communication at a coarse time-step would translate to all points in the grid being involved in communications at all local time steps involved in the particular coarse time-step.

These grid-relative metrics allow for a) inter-application comparisons, and b) validation of our suggested Trade-Off models, as they build on theory for, and assumptions on, the current grid hierarchy.

4.2 Absolute Importance of Relative Metrics

Previous research has established metrics for gauging the quality of a domain decomposition. Mostly, they have been *ad hoc*, e.g. load imbalance and argued that for a particular application a load imbalance of, say, x percent can be tolerated, while an imbalance of, say, y percent cannot. This is problematic.

The vast range of dynamics present in SAMR applications, clearly visible in metrics such as load imbalance and communication, is due to dynamics in the grid hierarchy. The hierarchy changes, not only with regards to shape and structure, but also with respect to size. For example, SAMR applications often exhibit patterns where the total number of grid points is doubled (or cut in half) for two consecutive time steps.

When studying load imbalance per-time-step, it is tempting to conclude that large imbalances translate to great need of optimization. But these large imbalances might occur at local minima of the grid size. Optimizing them at these points would have little or no impact on overall application execution time. However, large imbalances detected at a grid-size peaks offer potential for huge savings after optimization.

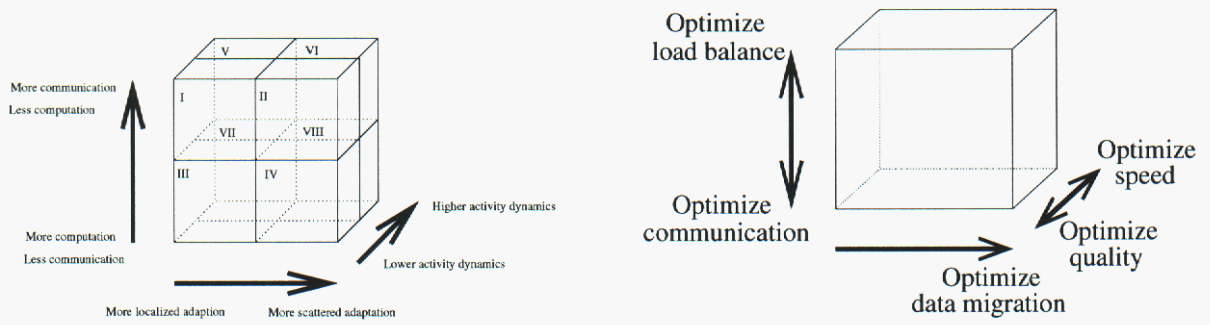

Figure 3: Left: The octant approach. The application and system is classified with respect to (1) communication / computation domination, (2) scattered / localized refinements, and (3) activity dynamics. Right: The absolute and continuous partitioner-centric classification space. Note the absence of unique trade-off for data migration, as opposed to the general trade-offs communication vs load balance and speed vs quality.

A successful model for determining partitioning trade-offs must take the absolute importance of relative metrics into consideration.

4.3 Trade-off 2: Speed vs. Overall Quality

The trade-off between speed and overall quality translates to a comparison of two entities. They are quantifications of 1) how much time the partitioner would like to spend to obtain its goals, and 2) what time-slot the application realistically can offer it.

Since Trade-Off 1 is derived from two penalties, it can provide the present trade-off (speed versus quality) with important information regarding the quantification of (1). β_l and β_c are compared by an equation that disregards the "amplitude" of the inputs. For example, $\beta_l = \beta_c = 0.1$ would yield the same result as $\beta_l = \beta_c = 0.4$. But this information is important to the present trade-off.

As the penalties approach 1, the more prominent is the need to optimize. Consequently, a first version of a quantification of (1) could be obtained by taking the average of the other penalties, including that for data migration. An average close to one translates to the greatest request for partitioning time, and an average close to zero would translate to a simple partitioning case with no particular demands on the partitioning (any will do).

To summarize, Trade-Off 1 tells us to what extent we should focus on load imbalance or communication, and Trade-Off 3 tells us how much data migration we can expect. The first version of our quantification of Trade-Off 2 tells us how bad *the relative* load imbalance, communication, or data migration is. This information is insufficient to determine how severe the case is and ultimately how much time we need to remedy it.

We need to quantify the absolute importance of the relative metrics. This is done by using the size of the current grid hierarchy as indicated above. Optimally, we would like to normalize the current grid size with respect to the largest of all grid hierarchies in the simulation. Since this information is unavailable, we propose to normalize the current grid size with respect to the largest grid encountered so far in the simulation.

To finalize the quantification of (1), we multiply our first version with the normalized grid size. This would give a good, normalized estimation of how much time the partitioner would like to spend to achieve it goals.

The quantification of (2) is more problematic. We propose[1] that the partitioner when invoked calls a timer to determine the invocation intervals. These timing calls will impose insignificant overhead, provided that the invocation frequency is small. Most large SAMR applications do not re-partition multiple times every second. Consequently, the more infrequently the partitioner is invoked, the greater the time-slots it can claim.

To finalize the model for this Trade-Off, we compare the quantities (1) and (2) as derived above, which place us along dimension II in the classification space.

[1] Actually, a reviewer for Part I proposed some kind of coarse grained timing calls.

In the present paper, we have outlined the detailed foundation for implementing Trade-Off 2. But only with hands-on, practical experimenting can we complete the work and investigate how to best a) normalize (2), and b) compare (1) and (2).

4.4 Trade-off 3: Data migration

To predict the amount of data migration, or rather capture the grid's inherent potential for data migration, we suggest studying the amount of perturbation. Chandra proposes keeping a history of grid hierarchies as a sliding window to allow for the comparison of adaptation patterns at different time steps and to prevent thrashing and over-reacting to sudden changes [13]. This model builds on tracking and reflecting *change*, i.e., all metrics are relative to the previous state.

Our normalized data migration penalty β_m is similar but absolute in the sense that each pair of time-consecutive grid hierarchies is mapped onto a value $\in [0, 1]$ in dimension III of the classification space, independently of any previous mapping at any other time-step. Our β_m also conforms to our proposed model in that it is comparable to our grid-relative metric for data migration.

By intersecting the boxes in the hierarchy at time-step $t-1$ with those at time-step t, we get an indication of how much the grid has changed during this time-step. A large intersection means little change and vice versa. Let the hierarchy at time-step t be H_t and let G_t^l denote the set of all patches in the grid at time t and at level l. Furthermore, let $G_t^{i,l}$ denote the i:th grid patch in G_l^t. Then, the data migration penalty

$$\beta_m\left(H_{t-1}, H_t\right) = 1 - \frac{1}{|H_t|} \sum_{l=0}^{l_{\max}} \sum_{i=1}^{|G_{t-1}^l|} \sum_{j=1}^{|G_t^l|} \left| G_{t-1}^{l,i} \times G_t^{l,j} \right|,$$

where the operator $\times$ denotes grid intersection.

The choice for $|H_t|$ as the denominator (as opposed to e.g. $|H_{t-1}|$ as for the data migration metric) stems from the following observations. When we move from a small to a large grid, we expect that a large fraction of the small grid will be moved, since partitioners generally (and preferably) focus on the topmost refinement levels at the expense of the lower levels. Conversely, when we move from a large to a small grid, we expect a small fraction of this large grid to actually being moved, since most of it is probably deleted. Formally, $|H_{t-1}| < |H_t|$ suggests $|H_t|$ as a denominator to yield a larger value when it is subtracted from 1. Analogously, $|H_{t-1}| > |H_t|$ also suggests $|H_t|$.

5 Validation

This section explains the experimental process for validating the proposed model and presents the results.

5.1 Methods — Experimental Setup

A trace file (described below) from each of the four SAMR applications (described below) is used in two different ways. First, the trace-file is processed by a program implementing our proposed model. This program outputs β_m and β_c for each time-step. Second, the trace-file is partitioned by `Nature+Fable` and processed by the SAMR simulator (described below). This program outputs the actual partitioning result in terms of relative communication and data migration for each time step. The two sets of output, viz. β_m vs actual data migration, and β_c vs actual communication amount, are then for each application plotted in the same figure to enable visual comparison. The idea is *not* that the plots should coincide — rather, the idea is to examine whether the model (i.e., β_c and β_m) succeeds in capturing the overall *behavior* of the different applications, i.e., for a static and non-optimized partitioning setup, will the analytical model correctly capture the difficult-to-reduce-data-migration and difficult-to-reduce communications configurations of the grid hierarchy.

5.1.1 Four SAMR Applications

A suite of 4 "real-world" SAMR application kernels taken from varied scientific and engineering domains are used to evaluate the effectiveness of the proposed model to capture application behavior. These applications demonstrate different runtime behavior and adaptation patterns. Application domains include numerical relativity (Scalarwave), oil reservoir simulations (Buckley-Leverette), and computational fluid dynamics (compressible turbulence - RM). 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 5 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 2. The application kernels are described below.

The numerical relativity application (Scalarwave/SC) 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[2].

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[3].

The RM is a 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[4]. 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.

5.1.2 Partitioning Set-Up

All partitioning is done with `Nature+Fable` set-up with static "default" values [36, 38]. The goal is not to obtain a particularly good-quality partitioning, but rather to partition the applications with a static "neutral" setting so that behavior patterns in the applications are clearly visible.

5.1.3 Deriving Application Behavior

The derivation of data migration and communication amount is performed using software [34] developed at Rutgers University in New Jersey by The Applied Software Systems Laboratory, that simulates the execution of the Berger-Colella SAMR algorithm. This software 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 experimental process allows the user to select the partitioner to be used, the partitioning parameters (e.g. block size), and the number of processors. The trace is then run and the performance of the partitioning configuration at each regrid step is computed using a metric [36] with the components load balance, communication, data migration, and overheads.

5.1.4 Model Evaluation

To evaluate the ability of our penalties β_m and β_c to accurately capture the behavior of the applications, we plotted each application's relative data migration as a function of time and super imposed β_m without any scaling. The same was done for communication versus β_c. No numerical results, e.g. in terms of error norms, were derived. The purpose of this experimental process was to examine whether our model indeed reflects the inherent and dynamic optimization-need in the applications. This was most easily examined visually.

5.2 Results

Figure 4 through 7 display the results. Examining the plots, it seems that the proposed model generally captures the essence of application behavior i.e., a larger β_m generally corresponds to a greater amount of data migration and a larger β_c generally corresponds to larger communication amount. The trends are similar, and in case of oscillatory behavior, the model captures the time period of the oscillation. Note that the values obtained from the simulations are governed by the domain decomposition achieved by the actual (hybrid) partitioner used, while the model predictions are

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

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

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

based on derivations from the unpartitioned grid hierarchy.

Generally, β_c is a bit aggressive, i.e., it "jumps" at potentially communication-heavy grids. These "jumps" did indeed reflect time-steps with more communication, but the partitioner could in reality cope relatively easy. This result meets our goals, since β_c reflects a "worst-case scenario" and is used in the model to trade-off the importance of load imbalance. Also, `Nature+Fable` hosts only hybrid partitioning techniques, and is therefore expected to produce substantially less communication than is indicated by β_c.

The penalty β_m on the other hand, is somewhat cautious in its predictions. The penalty reflected fairly well the actual migration amount, but the "amplitude" was generally slightly lower. Whether this was due to relatively high migration amount for the particular partitioning technique — perhaps due to the partially ordered space-filling curve — or is inherent in β_m can only be determined by future investigations.

Below, we discuss the results for each application.

RM2D (See Figure 4) Both penalties successfully capture the essence of this application. Both communication and data migration change seemingly randomly, and the penalties accurately reflect that.

BL2D (See Figure 5) Both the data migration and the communication exhibited oscillatory behavior for this application. Both β_m and β_c followed the time periods and accurately showed the same "peaks" and "valleys" in most cases. It seems that β_m peaks one time-step before the relative data migration occasionally. The fit between β_c and communication was very good.

SC2D (See Figure 6) This application exhibited oscillatory behavior both in load imbalance and communication volume. Both β_m and β_c followed the time periods and accurately showed the same "peaks" and "valleys". It seems that β_m peaks one time-step before the relative data migration occasionally. The fit between β_c and communication was very good.

TP2D (See Figure 7) This application exhibited seemingly random data migration and communication dynamics and the penalties accurately reflected this. Both penalties captured their respective metrics very well.

6 Conclusions and Future Work

We have developed a model that, *ab initio*, predicts the inherent potential for data migration in grid hierarchies. This prediction is used for determining one of the parameters for selecting and configuring the best partitioner for a given problem. The predictions were validated against data obtained from four different SAMR simulations.

From the results we draw the conclusion that our model could be useful for decreasing execution time for large SAMR applications. Most such applications exhibit a highly dynamic behavior and consequently the partitioning requirements change dynamically during execution. Accurately tracking and adapting to this dynamic behavior lead to potentially large decreases in execution times.

In the future, we will address the remaining, practical part of this research. This includes executing the applications on a number of different parallel platforms and letting our model provide the partitioner with trade-off settings reflecting the applications' inherent dynamical properties. We will refine our model with these experiments and investigate how it would be best implemented as part of the adaptive meta-partitioner with the ultimate goal to reduce execution times for general large-scale SAMR applications.

References

[1] The ASCI alliance. http://www.llnl.gov/asci-alliences/asci-chicago.html, University of Chicago, 2000.

[2] The ASCI/ASAP center. http://www.carc.caltech.edu/ASAP, California Institute of Technology, 2000.

[3] CHOMBO. http://seesar.lbl.gov/anag/chombo/, NERSC, ANAG of Lawrence Berkeley National Lab, CA, USA, 2003.

[4] Scott B. Baden, Scott R. Kohn, and S. Fink. Programming with LPARX. Technical Report, University of California, San Diego, 1994.

[5] Dinshaw Balsara and Charles Norton. Highly parallel structured adaptive mesh refinement using language-based approaches. *Journal of parallel computing*, (27):37'–70, 2001.

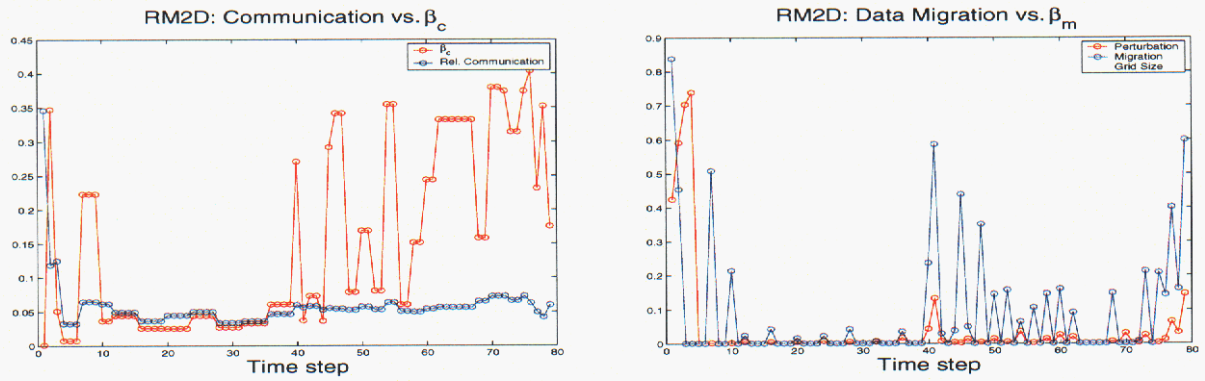

Figure 4: The ability of the penalties to predict application behavior for RM2D. Left, the actual relative communication (in blue) and the penalty β_c (in red). Right, the actual relative data migration (in blue) and the penalty β_m (in red).

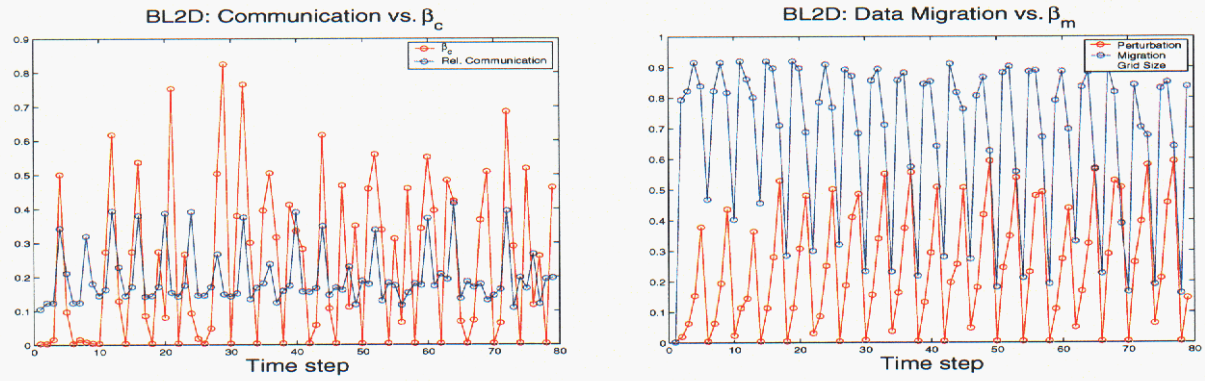

Figure 5: The ability of the penalties to predict application behavior for BL2D. Left, the actual relative communication (in blue) and the penalty β_c (in red). Right, the actual relative data migration (in blue) and the penalty β_m (in red).

[6] M. Berger, et al. Adaptive mesh refinement for 1-dimensional gas dynamics. *Scientific Computing*, 17:43–47, 1983.

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

[8] Marsha J. Berger and Joseph Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of Computational Physics*, 53:484–512, 1984.

[9] G. Bryan. Fluids in the universe: Adaptive mesh refinement in cosmology. *Computing in Science and Engineering*, pages 46–53, 1999.

[10] S. Chandra and M. Parashar. An evaluation of partitioners for parallel SAMR applications. *Lecture Notes in Computer Science*, 2150:171–174, 2001. Euro-Par 2001.

[11] S. Chandra, J. Steensland, and M. Parashar. An experimental study of adaptive application sensitive partitioning strategies for SAMR applications, 2001. Research poster presentation at Supercomputing Conference, November 2001.

[12] S. Chandra, J. Steensland, M. Parashar, and J. Cummings. An experimental study of adaptive application sensitive partitioning strategies for SAMR applications. Santa Fe, NM, USA, 2001.

[13] Sumir Chandra. ARMaDA: a framework for adaptive application-sensitive runtime management of dynamic applications. Master's Thesis, Graduate School, Rutgers University, NJ, USA, 2002.

[14] Mattew W. Choptuik. Experiences with an adaptive mesh refinement algorithm in numerical relativity. *Frontiers in Numerical Relativity*, pages 206–221, 1989.

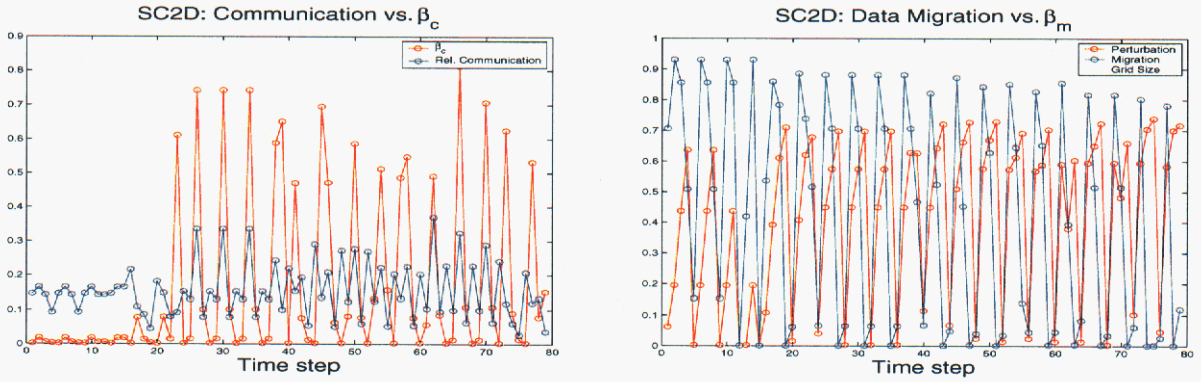

Figure 6: The ability of the penalties to predict application behavior for SC2D. Left, the actual relative communication (in blue) and the penalty β_c (in red). Right, the actual relative data migration (in blue) and the penalty β_m (in red).

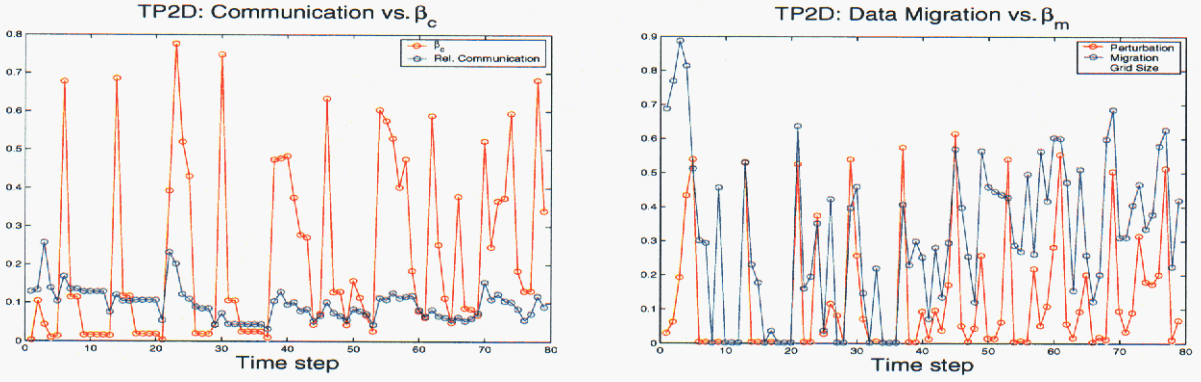

Figure 7: The ability of the penalties to predict application behavior for TP2D. Left, the actual relative communication (in blue) and the penalty β_c (in red). Right, the actual relative data migration (in blue) and the penalty β_m (in red).

[15] Karen Devine et al. Design of dynamic load-balancing tools for parallel applications. Technical report, Sandia national Laboratories, Albuquerque, NM, USA, 2000.

[16] Stephen J. Fink, Scott B. Baden, and Scott R. Kohn. Flexible communication mechanisms for dynamic structured applications. In *Proceedings of IRREGULAR '96*, 1996.

[17] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, 20:359–392, 1998.

[18] Scott Kohn. SAMRAI homepage, structured adaptive mesh refinement applications infrastructure. http://www.llnl.gov/CASC/SAMRAI/, 1999.

[19] Z. Lan, V. Taylor, and G. Bryan. Dynamic load balancing for structured adaptive mesh refinement applications. In *Proceedings of ICPP 2001*, 2001.

[20] Z. Lan, V. Taylor, and G. Bryan. Dynamic load balancing of SAMR applications on distributed systems. In *Proceedings of Supercomputing 2001*, 2001.

[21] Peter MacNeice. Paramesh homepage, 1999. sdcd.gsfc.nasa.gov/ESS/macneice/paramesh/-paramesh.html.

[22] Peter MacNeice et al. PARAMESH: A parallel adaptive mesh refinement community toolkit. *Computer physics communications*, (126):330–354, 2000.

[23] M. Norman and G. Bryan. Cosmological adaptive mesh refinement. *Numerical Astrophysics*, 1999.

[24] M. Parashar and J. C. Browne. On partitioning dynamic adaptive grid hierarchies. In *Proceedings of the 29th Annual Hawaii International Conference on System Sciences*, 1996.

[25] M. Parashar, J.A. Wheeler, G. Pope, K.Wang, and P. Wang. A new generation EOS compositional reservoir simulator: Part II - framework and multiprocessing. *Proceedings of the Society of Petroleum Engineerings Reservoir Simulation Symposium, Dallas, TX*, June 1997.

[26] Manish Parashar and James 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*, pages 1–18, 2000.

[27] S.G. Parker. A component-based architecture for parallel multi-physics PDE simulations. In *Proceedings of ICCS 2002*, number 2331, pages 719–734. Springer Verlag, 2002.

[28] R. Pember, J. Bell, P. Colella, W. Crutchfield, and M. Welcome. Adaptive cartesian grid methods for representing geometry in inviscid compressible flow, 1993. *11th AIAA Computational Fluid Dynamics Conference*, Orlando, FL, July 6-9.

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

[30] Hawley S. and Choptuic M. Boson stars driven to the brink of black hole formation. *Physic Rev*, D 62:104024, 2000.

[31] Jeffrey Saltzman. Patched based methods for adaptive mesh refinement solutions of partial differential equations, 1997. Lecture notes.

[32] K. Schloegel, G. Karypis, and V. Kumar. Multilevel diffusion schemes for repartitioning of adaptive meshes. Technical Report 97-013, Dept. of Computer Science, Univ. of Minnesota, Minneapolis, MN, 1997.

[33] K. Schloegel, G. Karypis, and V. Kumar. A unified algorithm for load-balancing adaptive scientific simulations. In *Proceedings of Supercomputing 2000*, 2000.

[34] Mausumi Shee. Evaluation and optimization of load balancing/distribution techniques for adaptive grid hierarchies. M.S. Thesis, Graduate School, Rutgers University, NJ, 2000
http://www.caip.rutgers.edu/TASSL/Thesis/mshee-thesis.pdf, 2000.

[35] Johan Steensland. Domain-based partitioning for parallel SAMR applications, 2001. Licentiate thesis. Uppsala University, IT, Dept. of scientific computing. 2001-002.

[36] Johan Steensland. *Efficient partitioning of dynamic structured grid hierarchies*. PhD thesis, Uppsala University, 2002.

[37] Johan Steensland, Sumir Chandra, and Manish Parashar. An application-centric characterization of domain-based SFC partitioners for parallel SAMR. *IEEE Transactions on Parallel and Distributed Systems*, December:1275–1289, 2002.

[38] Johan Steensland and Jaideep Ray. A heuristic remapping algorithm reducing inter-level communication in SAMR applications. In *Proceedings of The 15th IASTED International Conference on Parallel and distributed computing and systems PDCS03*, volume 2, pages 707–712. ACTA PRESS, 2003.

[39] Johan Steensland and Jaideep Ray. A partitioner-centric classification space for SAMR partitioning trade-offs : Part I. In *Proceedings of LACSI 2003, Los Alamos computer science symposium on CD-ROM*, 2003.

[40] Johan Steensland, Stefan Söderberg, and Michael Thuné. A comparison of partitioning schemes for blockwise parallel SAMR applications. In *Proceedings of PARA2000, Workshop on Applied Parallel Computing*, volume 1947 of *LNCS*, pages 160–169. Springer Verlag, 2001.

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

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

[43] 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*, 26:1535–1554, 2000.

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

[45] 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. *Proceedings of the Society of Petroleum Engineerings Reservoir Simulation Symposium, Dallas, TX*, June 1997.

[46] Andrew M. Wissink et al. Large scale parallel structured AMR calculations using the SAMRAI framework. *In proceedings of Supercomputing 2001*, 2001.

Distribution List

External Distribution

1 Manish Parashar
Department of Electrical & Computer Engineering
Rutgers, The State University of New Jersey
94 Brett Road, Piscataway, NJ 08854-8058

1 Sumir Chandra
CAIP Center at Rutgers University
CORE Building, Busch Campus
96 Frelinghuysen Rd. Piscataway, NJ 08855-1390

1 Michael Thuné
Information Technology
Department of Scientific Computing
P.O. Box 337, SE-751 05 Uppsala, Sweden

1 Jarmo Rantakokko
Information Technology
Department of Scientific Computing
P.O. Box 337, SE-751 05 Uppsala, Sweden

1 Henrik Johansson
Information Technology
Department of Scientific Computing
P.O. Box 337, SE-751 05 Uppsala, Sweden

Internal Distribution

2	MS 9915	Johan Steensland, 8964
1	MS 9051	Jaideep Ray, 8964
1	MS 0969	Mike Hardwick, 8964
3	MS 9018	Central Technical Files, 8945-1
1	MS 0899	Technical Library, 9616
1	MS 0612	Classification Office, 8511 for Technical Library, MS 0899, 9616
		DOE OSTI via URL