A Characterization of a Hybrid and Dynamic Partitioner for SAMR Applications

Henrik Johansson, Johan Steensland

Department of Information Technology
Uppsala University
Box 337, SE-751 05 Uppsala, Sweden

Technical report 2004-009
March 2004
ISSN 1404-3203

A Characterization of a Hybrid and Dynamic Partitioner for SAMR Applications*

Henrik Johansson[1] and Johan Steensland[2]

[1] IT, Dept of Scientific Computing, Uppsala University, Box 337, S-751 05 Uppsala, Sweden, henrikj@it.uu.se
[2] Advanced Software Research and Development, Sandia National Laboratories, P.O. Box 969, Livermore, CA 94550, USA, jsteens@sandia.gov

Abstract. Significantly improving the scalability of large structured adaptive mesh refinement (SAMR) applications is challenging. It requires sophisticated capabilities for using the underlying parallel computer's resources in the most efficient way. This is non-trivial, since the basic conditions for how to allocate the resources change dramatically during run-time due to the dynamics inherent in these applications.
This report presents a first characterization of a hybrid and dynamic partitioner for parallel SAMR applications. Specifically, we determine optimal parameter settings for trade-offs like communication vs. load balance and speed vs. quality. The key contribution is that the characterization enables the partitioner to respond accurately to stimuli from system and application state, and hence adapting to various SAMR scenarios. This potentially reduces run-time for large SAMR applications.

Keywords: Dynamic run-time environments, structured adaptive mesh refinement, partitioning, load balancing.

1 Introduction

Structured adaptive mesh (SAMR) methods are being widely used for simulations of physical phenomena in domains like computational fluid dynamics [2,6,26], numerical relativity [14,28], astrophysics [1,9,21], and subsurface modeling and oil reservoir simulation [39,23]. Significantly improving the scalability of large SAMR applications is challenging. It requires sophisticated capabilities for using the underlying parallel computer's resources in the most efficient way. This is non-trivial, since the basic conditions for how to allocate the resources change dramatically during run-time due to the dynamics inherent in these applications.

Previous research has targeted execution time optimization for specific combinations of programming model, computer system, and application [5,40,25]. The present contribution is a part of a larger research effort [30,31,32,10,33,34]

* A shorter version of this report is submitted to Euro-Par 2004 in Pisa, Italy.

that aims to improve performance for general SAMR applications executing on general parallel computers. The main 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 [27,31]. Adaptive management of these dynamic applications at run-time is necessary.The *meta-partitioner* [13,12,11] provides such capabilities, by selecting and configuring the most suitable partitioner based on the current system and application state. Previous research has offered design, proofs-of-concepts, and evaluation of major components.

This report goes one step further by providing a detailed study of *Nature+Fable*[31], a core component of the meta-partitioner. *Nature+Fable*is a partitioning framework, hosting a wide set of parameters. Different parameter settings lead to different behavior. The work presented here is a characterization of this parameter space, based on five SAMR applications with different behavior. This characterization provides an intial basis for conclusions about how to adapt *Nature+Fable*to various SAMR scenarios, by dynamic adjustment of parameters.

2 Partitioning of SAMR Applications

2.1 Structured Adaptive Mesh Refinement

Dynamically adaptive mesh refinement (AMR) [35] methods for the numerical solution to partial differential equations (PDE's) [7,8,29] 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 the computational grid in regions with large local 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. Furthermore, these methods 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. However, they also pose new 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.

For 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 structure is a dynamic adaptive grid hierarchy.

Existing software infrastructures for SAMR include Paramesh [19,20], a FORTRAN library for parallelization of and adding adaption to existing serial structured grid computations, SAMRAI [16,40] a C++ object-oriented framework for implementing parallel structured adaptive mesh refinement simulations, and GrACE [24] 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

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. 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 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.[3]

For *patch-based partitioners* [5,17], 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 frameworks SAMRAI [16,40] (based on the LPARX [4] and KeLP [15] model) fully supports patch-based partitioning. The advantages are manageable load imbalance and 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 [31].

Domain-based partitioners [22,27,36,30] partition the physical domain, rather than the grids themselves. The domain is partitioned along with all contained

[3] Note that this report 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 [37,38].

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 [31].

Hybrid partitioners [22,36,18] combining patch-based and domain-based approaches, can be used for coping with the shortcomings present in these techniques. They use a 2-step partitioning approach. The first step uses domain-based techniques to generate coarse 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 coarse partition within its processor group.

2.3 *Nature+Fable*

Developed at Uppsala University, Sweden, Rutgers University, New Jersey, USA, and Sandia National Laboratories, USA, `Nature+Fable` (**Natur**al **Re**gions + **Fra**ctional blocking and **bi-le**vel partitioning) [31] aims to be the best possible tool for partitioning SAMR grid hierarchies. It hosts a variety of hybrid partitioning options. All involved parts are engineered to be components of the meta-partitioner. Thus, they offer carefully designed parameters to steer its behavior enabling adaptation to varying partitioning requirements imposed by applications and computer systems.

`Nature+Fable` separates homogeneous, un-refined (Hue) and complex, refined (Core) domains of the grid hierarchy and clusters refinement levels into *bi-levels* [31]. A bi-level consists of two refinement levels - a partition of grid level k together with all its superimposed refinements on the next level. The concept of bi-levels is illustrated in Fig. 1. 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 used for these bi-levels.

*Nature+Fable*is a hybrid partitioner with high degrees of freedom. It is equipped with a set of parameters, allowing it to adapt to the dynamic requirements imposed by applications and computer systems. To fully exploit the adaptation possibilities, the parameter space must be characterized with regards to the impacts of given parameters — or combination of parameters. Below, such a characterization is presented. It provides a basis for dynamically adapting *Nature+Fable*to various SAMR scenarios, within the concept of the meta-partitioner.

3 Experimental Setup

A suite of five "real-world" SAMR application kernels [32] taken from varied scientific and engineering domains were used for the characterization. Applica-

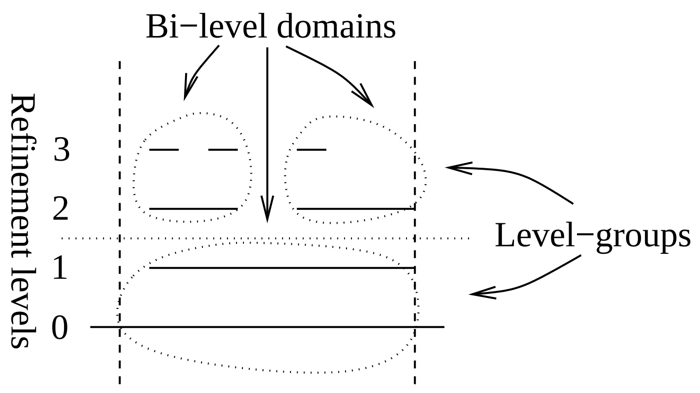

Fig. 1. Four refinement levels are grouped into bi-levels. Note how the two uppermost levels are split into two bi-levels.

tion domains include numerical relativity (Scalarwave), oil reservoir simulations (Buckley-Leverett), and computational fluid dynamics (compressible turbulence - Richtmyer-Meshkov). These applications demonstrate different runtime behavior and adaptation patterns. The 2D (3D) applications use 5 (3) levels of factor 2 refinements in space and time. Regridding and redistribution is performed every 4 time-steps on each level.

The evaluation was performed using software that simulates the execution of the Berger-Colella SAMR algorithm for 32 processors. The performance of the partitioning configuration at each regrid step was computed using a metric [31] with the components load balance, communication, data migration, partitioning time and number of boxes. Here, communication was measured as the sum of the amount of inter-processor communication taken over all time-steps. Data migration was captured by the sum of the total number of data points forced to migrate as a result of re-partitioning, taken over all time-steps. Both communication and data migration were normalized with respect to grid hierarchy size and workload [33]. Partitioning times were measured as the time to partition an entire trace file, minus the time for the same program calling a dummy partitioning.

Next, we turn to the *Nature+Fable*parameters.To isolate the effect of a given parameter, we base our experiments on a "default setting" and let only the parameter under study deviate from it. Our study includes the following specific items (summarized in Table 1, see [31] for a detailed description of the parameters):

Quantum The parameter `Q` affects partitioning granularity in the blocking step. A larger `Q` corresponds to a finer granularity and thus the ability to create smaller boxes. The granularity affects load imbalance and communication. We studied this by varying `Q` between 2, 4, and 6.

Mapping The mapping functionality [33] in *Nature+Fable*strives to reduce inter-level communication by assigning overlapping boxes (with respect to different bi-levels) to the same processor. If the overlap is above a specified amount, the mapping is deemed good enough. We compared runs without mapping with runs with mapping and an overlap of 50 percent.

Blocking Mode Two blocking modes are available in *Nature+Fable*, Multiple and Fractional blocking. Multiple blocking tries to achieve load balance by creating more blocks in accordance with the idea of virtual processors. Fractional blocking only splits a few key blocks. We studied overall quality and speed of both blockers.

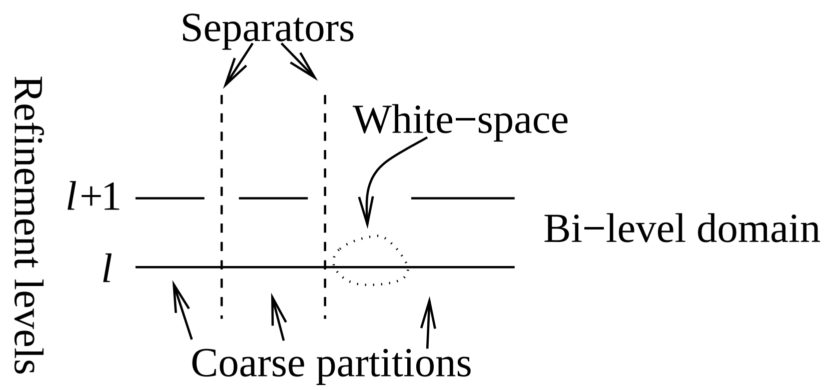

Fig. 2. The part of a bi-level that only consists of the lower refinement level is called white space. It is possible to turn this portion of the grid into a single-level with the parameter `WhiteSpace`. In this case it is possible to construct three bi-levels and two single-levels.

WhiteSpace The parameter `whiteSpace` determines how large fraction of a bi-level coarse partition that is allowed to have only the lower level (Fig. 2). A smaller value means that we cut the bi-levels into more boxes that are either strictly single- or bi-level. This affects load imbalance and communication.

4 Results

Quantum Finer partitioning granularity is achieved by increasing quantum. Finer granularity improved load balance at the expense of increased communication (Fig. 3). The gain in load balance was substantially greater than the loss in communication. Finer granularity also implied slightly more blocks and there seemed to be some dependance between this increase and the rise in communication. A more detailed study must be conducted to fully understand the mechanisms behind this behavior. Data migration exhibited a seemingly random behavior, as it decreased for some applications and increased for others.

Mapping The communication was reduced with at least ten percent when mapping was turned on (Fig. 4). The parameter succeeded in assigning boxes of

Parameter	Range	Exp values	Description
`actualLevels`	`on/off`	`off`	Regard Core's number of ref. levs. instead of max ref. levs.
`goodEnough`	$\in \mathbb{R}$	20.0	Stop searching for separators better than
`whiteSpace`	$\in [0,1]$	0.25, 0.5, 0.75, 1	White-space allowed
`bMode`	`FRAC/BLOCK`	`FRAC/BLOCK`	Blocking mode
`Q`	$\in \mathbb{N}$	2,4,6	Q for `FRAC` and k for `MULT`
`Mapping`	$\in [0,100]$	0, 50	Mapping on/off, and overlap pct
`smoothing`	$\in \mathbb{N}$	2	The smoothing factor
`growing`	$\in \mathbb{N}$	2	The growing factor
`mode` Bit 1	`on/off`	`off`	Strategy 2 on/off
`mode` Bit 2	`on/off`	`on`	Smoothing on/off
`mode` Bit 3	`on/off`	`on`	Connect regions on/off
`maxNRLoadImb`	$\in \mathbb{R}$	0.10	Max natural region induced load balance
`maxVirtualP`	$\in \mathbb{N}$	1	Maximum number of virtual processors
`atomicUnit`	$\in \mathbb{N}$	2	No grid side smaller than this allowed

Table 1. Summary of parameters in *Nature+Fable.* The values used in the experiments are found in column 3

different, but overlapping, bi-levels to the same processor. The amount of data migration was slightly increased. The extra time invested in the mapping function was smaller than expected. More research is however needed to determine the optimal use of this parameter, e.g. by investigating the result of other values than the ones presented here.

Blocking Mode Multiple blocking reduced migration at the expense of longer partitioning time and more boxes (Fig. 5). The reduction of migration was generally smaller than expected, but it was still substantial in some special cases (e.g. RM2D). Fractional blocking was on average generating a bit less communication, which was expected because of the smaller number of generated boxes. The load imbalance was roughly equal.

WhiteSpace Varying the parameter `whiteSpace` resulted in some counter-intuitive effects. As can be seen in Fig. 6, the impact on the 2D and 3D versions of the BL application was different. A low amount of white space resulted in a significant reduction of load imbalance for BL2D. The opposite was true for the 3D version (BL3D), where a larger amount of white space gave the best overall result. It appears as the result is dependent on the individual properties of each application, as it seems to be impossible to make any general conclusions. The

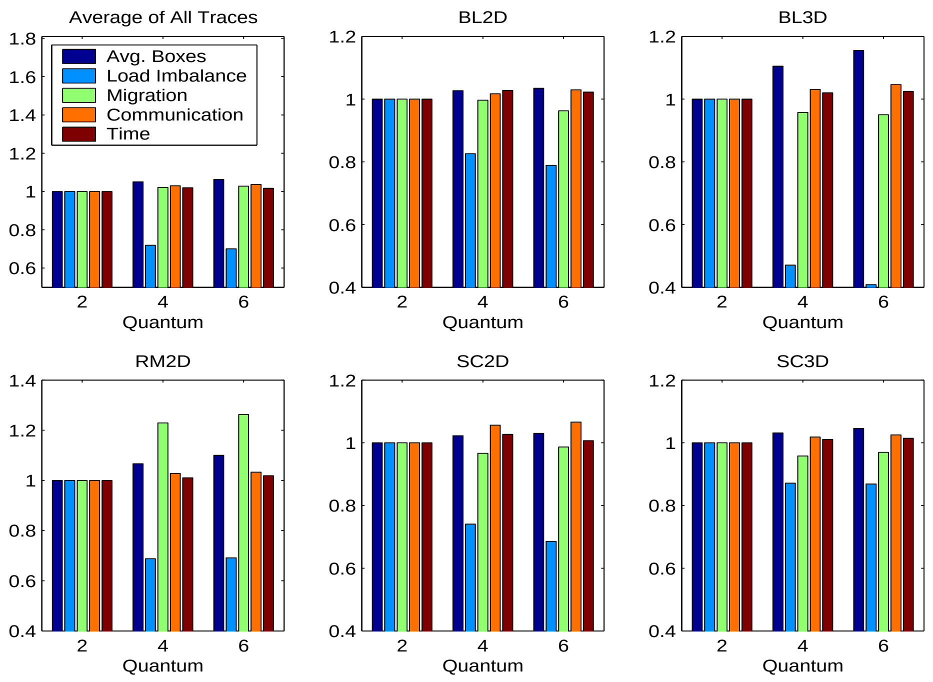

Fig. 3. Results for `Q`. The scale is normalized with respect to `Q`=2. Note the decrease in load imbalance for all applications.

mechanisms behind these differencies are not yet understood. Compared to load imbalance, the effects on communication, migration and partitioning time were less significant.

5 Conclusions and Future Work

Ultimately, the presented research strives to describe how the parameter space relates to the general partitioning trade-offs being (1) communication vs. load balance, (2) speed vs. overall quality, and (3) data migration. These trade-offs constitute the basis for our classification space [34].

By varying one parameter at the time, the quality of each metric-component could be significantly improved. Trade-off (1) was heavily influenced by `Q`, trade-offs (2) and (3) by `Mapping` and `Blocking Mode`. The parameter `whiteSpace` affected all metrics. We conclude that an optimal parameter setting will lead to significant run-time reductions for arbitrary SAMR applications. The research presented here has given valuable insight of how to adjust some of those parameters. The results also demonstrate the need for a deeper knowledge to fully understand the behavior of each parameter.

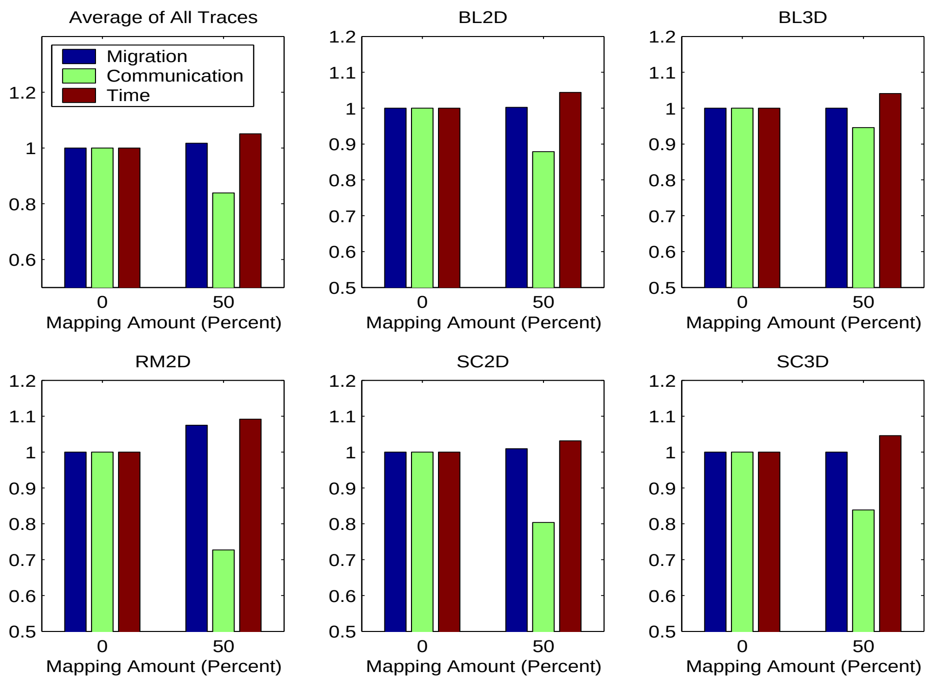

Fig. 4. Results for `Mapping`. The scale is normalized with respect to no Mapping. The time is only slightly increased with Mapping turned on.

The meta-partitioner [31] is our ultimate goal and it uses application and system state to determine the best combination of the trade-offs to decrease execution times. To function as a component of the meta-partitioner, *Nature+Fable* must be able to adapt to these recommendations. Our future research task is therefore to create an *abstract classification space layer* on top of the lower level function call to *Nature+Fable*. This abstraction layer should conform to the classification space described briefly above. The layer should host a front-end, providing input in form of a point in the classification space, and a back-end that calls *Nature+Fable*. In between the front and back ends, there should be an "engine" implementing translations based on the findings of the research.

Acknowledgments

The authors thank Jaideep Ray at Sandia National Laboratories, CA, USA, Manish Parashar and Sumir Chandra at the Center for Advanced Information Processing, Rutgers University, NJ, USA, and Michael Thuné and Jarmo Rantakokko at Information Technology, Uppsala University, Sweden for scientific collaboration. Sandia is a multiprogram laboratory operated by Sandia Corpora-

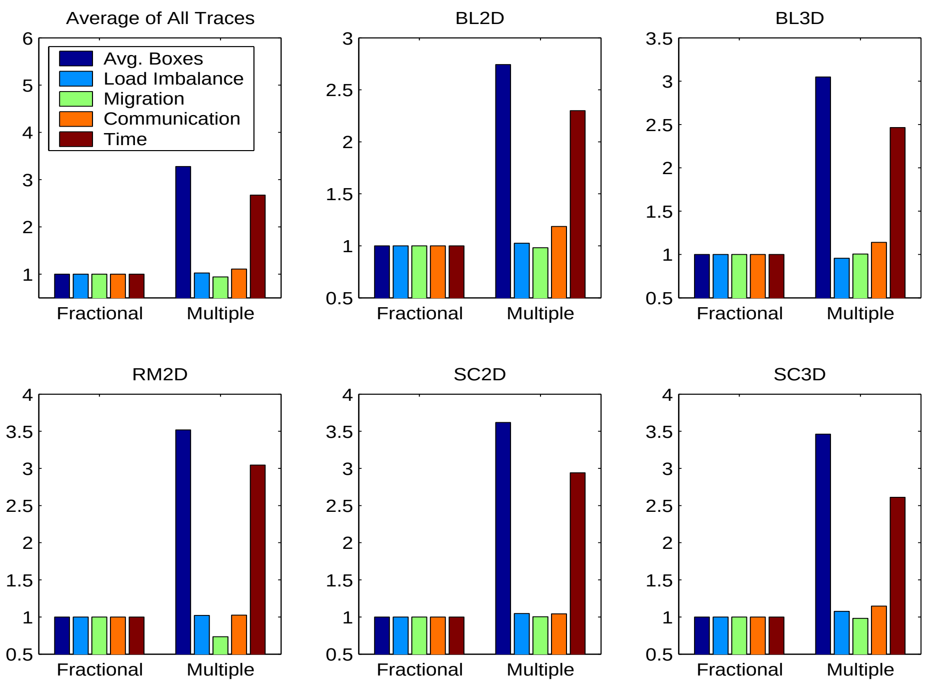

Fig. 5. Results for Fractional versus Multiple blocking. The scale is normalized with respect to Fractional blocking. Note the difference in time and number of boxes.

tion, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94-AL85000.

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.
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.

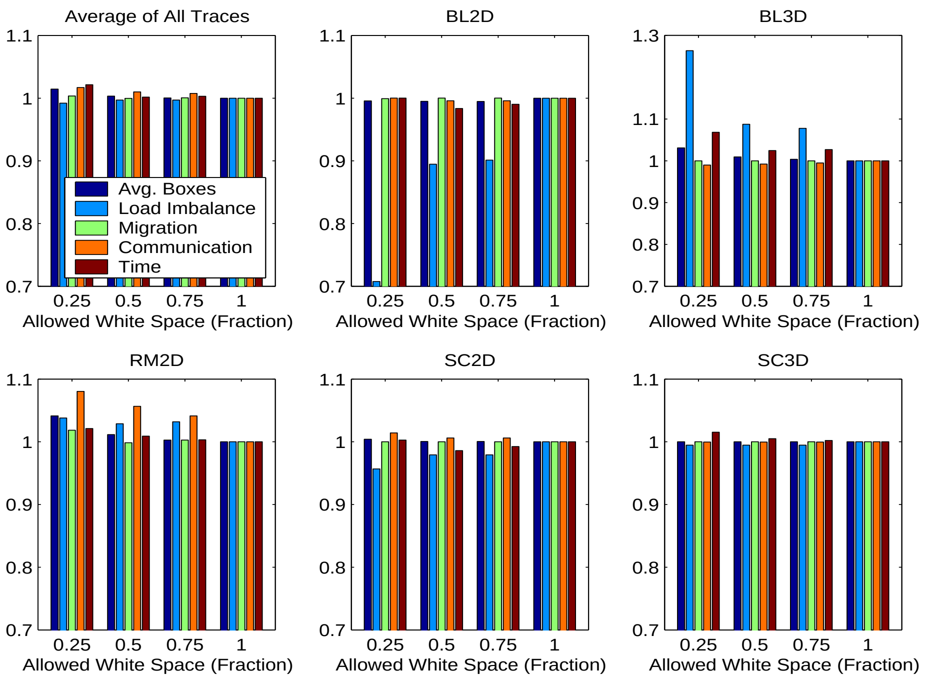

Fig. 6. Results for `whiteSpace`. The scale is normalized with respect to `whiteSpace` 1.00. The behavior of BL2D and BL3D is significantly different.

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

16. Scott Kohn. SAMRAI homepage, structured adaptive mesh refinement applications infrastructure. http://www.llnl.gov/CASC/SAMRAI/, 1999.
17. Z. Lan, V. Taylor, and G. Bryan. Dynamic load balancing for structured adaptive mesh refinement applications. In *Proceedings of ICPP 2001*, 2001.
18. Z. Lan, V. Taylor, and G. Bryan. Dynamic load balancing of SAMR applications on distributed systems. In *Proceedings of Supercomputing 2001*, 2001.
19. Peter MacNeice. Paramesh homepage, 1999. sdcd.gsfc.nasa.gov/ESS/macneice/paramesh/-paramesh.html.
20. Peter MacNeice et al. PARAMESH: A parallel adaptive mesh refinement community toolkit. *Computer physics communications*, (126):330–354, 2000.
21. M. Norman and G. Bryan. Cosmological adaptive mesh refinement. *Numerical Astrophysics*, 1999.
22. 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.
23. 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.
24. 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.
25. 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.
26. 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.
27. Jarmo Rantakokko. *Data Partitioning Methods and Parallel Block-Oriented PDE Solvers.* PhD thesis, Uppsala University, 1998.
28. Hawley S. and Choptuic M. Boson stars driven to the brink of black hole formation. *Physic Rev*, D 62:104024, 2000.
29. Jeffrey Saltzman. Patched based methods for adaptive mesh refinement solutions of partial differential equations, 1997. Lecture notes.
30. Johan Steensland. Domain-based partitioning for parallel SAMR applications, 2001. Licentiate thesis. Uppsala University, IT, Dept. of scientific computing. 2001-002.
31. Johan Steensland. *Efficient partitioning of dynamic structured grid hierarchies.* PhD thesis, Uppsala University, 2002.
32. 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.
33. Johan Steensland and Jaideep Ray. A heuristic re-mapping 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.
34. 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.

35. Erlendur Steinthorsson and David Modiano. Advanced methodology for simulation of complex flows using structured grid systems. *ICOMP*, 28, 1995.
36. M. Thuné. Partitioning strategies for composite grids. *Parallel Algorithms and Applications*, 11:325–348, 1997.
37. 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.
38. 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.
39. 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.
40. Andrew M. Wissink et al. Large scale parallel structured AMR calculations using the SAMRAI framework. *In proceedings of Supercomputing 2001*, 2001.

Recent technical reports from the Department of Information Technology

2003-055	Martin Nilsson: *Rapid Solution of Parameter-Dependent Linear Systems for Electromagnetic Problems in the Frequency Domain*
2003-056	Parosh Aziz Abdulla, Johann Deneux, Pritha Mahata, and Aletta Nylén: *Forward Reachability Analysis of Timed Petri Nets*
2003-057	Erik Berg and Erik Hagersten: *Low-Overhead Spatial and Temporal Data Locality Analysis*
2003-058	Erik Berg and Erik Hagersten: *StatCache: A Probabilistic Approach to Efficient and Accurate Data Locality Analysis*
2003-059	Jonas Persson and Lina von Sydow: *Pricing European Multi-asset Options Using a Space-time Adaptive FD-method*
2003-060	Pierre Flener: *Realism in Project-Based Software Engineering Courses: Rewards, Risks, and Recommendations*
2003-061	Lars Ferm and Per Lötstedt: *Space-Time Adaptive Solution of First Order PDEs*
2003-062	Emilio Tuosto, Björn Victor, and Kidane Yemane: *Polyadic History-Dependent Automata for the Fusion Calculus*
2003-063	Michael Baldamus, Joachim Parrow, and Björn Victor: *Spi Calculus Translated to π-Calculus Preserving May-Testing*
2003-064	Arnim Brüger, Bertil Gustafsson, Per Lötstedt, and Jonas Nilsson: *High Order Accurate Solution of the Incompressible Navier-Stokes Equations*
2003-065	Michael Baldamus, Richard Mayr, and Gerardo Schneider: *A Backward/Forward Strategy for Verifying Safety Properties of Infinite-State Systems*
2004-001	Torsten Söderström, Torbjörn Wigren, and Emad Abd-Elrady: *Maximum Likelihood Modeling of Orbits of Nonlinear ODEs*
2004-002	Pablo Giambiagi, Gerardo Schneider, and Frank D. Valencia: *On the Expressiveness of CCS-like Calculi*
2004-003	Johan Elf, Per Lötstedt, and Paul Sjöberg: *Problems of High Dimension in Molecular Biology*
2004-004	Torbjörn Wigren: *Recursive Prediction Error Identification of Nonlinear State Space Models*
2004-005	Håkan Zeffer, Zoran Radovic, Oskar Grenholm, and Erik Hagersten: *Evaluation, Implementation and Performance of Write Permission Caching in the DSZOOM System*
2004-006	Henrik Löf, Markus Nordén, and Sverker Holmgren: *Improving Geographical Locality of Data for Shared Memory Implementations of PDE Solvers*
2004-007	Erik Nordström, Per Gunningberg, and Christian Tschudin: *Comparison of Gateway Forwarding Strategies in Ad hoc Networks*
2004-008	Pär Samuelsson and Bengt Carlsson: *An Integrating Linearization Method for Static Input Nonlinearities*
2004-009	Henrik Johansson and Johan Steensland: *A Characterization of a Hybrid and Dynamic Partitioner for SAMR Applications*

UPPSALA
UNIVERSITET

March 2004
ISSN 1404-3203
http://www.it.uu.se/