

SANDIA REPORT

SAND2006-2487
Unlimited Release
Printed May 2006

An Improved Bi-Level Algorithm for Partitioning Dynamic Grid Hierarchies

Johan Steensland and Jaideep Ray, Sandia National Laboratories, CA
Henrik Johansson, Uppsala University
Ralf Deiterding, California Institute of Technology

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-AC04-94-AL85000.

Approved for public release; further dissemination unlimited.



Sandia National Laboratories

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

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

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 ordering: <http://www.ntis.gov/ordering.htm>



SAND2006-2487
Unlimited Release
Printed May 2006

An Improved Bi-Level Algorithm for Partitioning Dynamic Grid Hierarchies

Johan Steensland and Jaideep Ray
Advanced Software Research and Development
Sandia National Laboratory
P.O. Box 969, Livermore, CA 94550-9915, USA
{jsteens, jairay}@ca.sandia.gov

Henrik Johansson
Department of Information Technology, Uppsala University
P.O. Box 337, SE-751 05 Uppsala, Sweden
henrikj@it.uu.se

Ralf Deiterding
California Institute of Technology
MC 158-79, Pasadena, CA 91125, USA
ralf@cacr.caltech.edu

Abstract

Structured adaptive mesh refinement methods are being widely used for computer simulations of various physical phenomena. Parallel implementations potentially offer realistic simulations of complex three-dimensional applications. But achieving good scalability for large-scale applications is non-trivial. Performance is limited by the partitioner's ability to efficiently use the underlying parallel computer's resources. Designed on sound SAMR principles, *Nature+Fable* is a hybrid, dedicated SAMR partitioning tool that brings together the advantages of both domain-based and patch-based techniques while avoiding their drawbacks. But the original bi-level partitioning approach in *Nature+Fable* is insufficient as it for realistic applications regards frequently occurring bi-levels as "impossible" and fails. This document describes an improved bi-level partitioning algorithm that successfully copes with all possible

bi-levels. The improved algorithm uses the original approach side-by-side with a new, complementing approach. By using a new, customized classification method, the improved algorithm switches automatically between the two approaches. This document describes the algorithms, discusses implementation issues, and presents experimental results. The improved version of `Nature+Fable` was found to be able to handle realistic applications and also to generate less imbalances, similar box count, but more communication as compared to the native, domain-based partitioner in the SAMR framework `AMROC`.

Acknowledgment

The authors thank 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.

Contents

1	Introduction	9
2	SAMR with AMROC and Nature+Fable	10
2.1	Parallel SAMR — Related Work	10
2.2	AMROC	10
2.3	Nature+Fable	11
3	The Improved Bi-Level Algorithm	12
3.1	The Original Approach	12
3.2	The Complementary Approach	13
3.3	The Classification Algorithm	14
3.4	Implementation	17
4	Experimentation	19
4.1	Hypotheses	19
4.2	Experiments	19
4.3	Results	22
4.4	Discussion: More Communication, Less Load Imbalance	22
5	Summary and Conclusions	25
6	Future Work: A New Bi-Level-to-Processor Mapping	25
6.1	The Current Mapping in Nature+Fable	25
6.2	Ideas for a A New Mapping	26

List of Figures

1	Representative AMROC scale-up test for fixed problem size	11
2	The three steps of the original approach (CDA): 1) Separation, 2) Cleaning, and 3) Blocking.	13
3	Two “impossible” bi-levels. Left: The “circular” children pattern. Right: A patch of atomic unit size P_a with an immediate neighbor P_b	14
4	The three steps of the complementary approach (PDA): 1) Cleaning, 2) Blocking, and 3) Derive kids.	15
5	A schematic view over the improved bi-level partitioning algorithm. New functions are yellow and old are green. The figure shows the relationship between the new and old functions, and how the old functions are re-used by the improved algorithm.	20
6	Upper: Iso-lines of density on refinement levels at the final time step of <i>Ramp2d</i> . Lower: An enlargement of the relevant triple region.	21
7	Iso-lines of density on refinement levels at different time step of <i>ConvShock2d</i> show the expansion of the shock wave after the reflection at $t \approx 0.3$ in the origin. . .	31
8	The improved bi-level algorithm versus PDA alone for <i>Ramp2d</i> and <i>ConvShock2d</i> for 16 processors: Load imbalance (Left) and box count (Right).	32
9	The improved bi-level algorithm versus PDA alone for <i>Ramp2d</i> and <i>ConvShock2d</i> for 16 processors: Communication volume.	32

10	Nature+Fable with the improved bi-level algorithm versus AMROC DB/SFC for ConvShock2d and 16 processors: Load imbalance (Left) and box count (Right). . .	33
11	Nature+Fable with the improved bi-level algorithm versus AMROC DB/SFC for ConvShock2d and 16 processors: Communication volume.	33
12	Nature+Fable with the improved bi-level algorithm versus AMROC DB/SFC for Ramp2d and 16 processors: Load imbalance (Left) and box count (Right).	34
13	Nature+Fable with the improved bi-level algorithm versus AMROC DB/SFC for Ramp2d and 16 processors: Communication volume.	34
14	For this simple 1D hierarchy, Nature+Fable generated a perfect (DB) partitioning.	35
15	For this simple 1D hierarchy, a BD partitioning will probably generate too much load imbalance. Nature+Fable fails to map all blocks optimally (only the yellow processor avoids part of its inter-level communication).	35

List of Tables

1	The statistical quantities used for defining the macro predicates used for classification of bi-levels	18
2	The definition of these thresholds is a result of lengthy trial-and-error	19
3	Key properties of the AMROC application traces.	21
4	Actual total synchronization time, including both communication and wait, in seconds for the two applications on both ACL and SHC using AMROC DB-SFC and 16 processors. Communication volume is increased by the factor f . The row for $f = 0$ contains extrapolated values, corresponding to net wait times caused by load imbalances. Note that $f = 1$ indicates using the applications' original ghost cell buffer width.	23
5	Communication time — assuming perfect load balance — computed as total synchronization time minus the extrapolated net wait times in Table 4. Note that the communication time for Nature+Fable would roughly correspond to $f = 4$ as the communication volume for Nature+Fable was about 4 times higher and load balance was significantly better as compared to AMROC DB-SFC.	23
6	Comparison of the actual total synchronization time for AMROC DB-SFC (top row) to the estimated synchronization time for perfect load balance but with communication volume increased by a factor of 4 (bottom row).	24
7	Percent of total run-times spent in synchronization for AMROC DB-SFC.	24

This page intentionally left blank

An Improved Bi-Level Algorithm for Partitioning Dynamic Grid Hierarchies

1 Introduction

Structured adaptive mesh refinement (SAMR) methods are being widely used for numerical solutions to partial differential equations (PDEs) [4] in many simulation areas, including climate modeling [13], computational fluid dynamics [4], numerical relativity [6], astrophysics [5], subsurface modeling, oil reservoir simulation [31], electromagnetic field simulations [10], and flame simulations [11]. As SAMR leads to uniform operations on regular arrays and exhibits structured communication patterns, implementations can be efficient.

Parallel SAMR potentially leads to realistic modeling of complex three-dimensional physical phenomena. However, large-scale SAMR applications place vastly diverse requirements on the partitioning strategy to enable efficient use of computer resources and consequently good scalability [28, 29]. Performance is limited by the partitioner’s ability to simultaneously trade-off and balance load, optimize communication and synchronization, minimize data migration costs, maximize grid quality, and expose and exploit available parallelism.

Traditionally, partitioning techniques for SAMR have been classified as either domain-based or patch-based. But due to the inherent shortcomings in both of these basic concepts, neither of them can provide good scalability for a broader spectrum of computer-application combinations. Designed on sound SAMR principles, *Nature+Fable* [25] is a hybrid, dedicated SAMR partitioning tool that brings together the advantages of both concepts while avoiding their drawbacks.

Unfortunately, the original bi-level partitioning approach in *Nature+Fable* has been proven to be insufficient as it for some complex and realistic applications considers frequently occurring bi-levels as “impossible” and fails. This document describes an improved bi-level partitioning algorithm that successfully copes with all bi-levels present in two complex and realistic applications derived from AMROC [7]. The improved algorithm uses the original approach side-by-side with a new, complementing approach. By using a new, customized classification method, the improved algorithm switches automatically between the two approaches. This document describes both the original approach and the new approach, as well as the classification method. Moreover, this document discusses implementation issues and presents experimental results.

2 SAMR with AMROC and Nature+Fable

2.1 Parallel SAMR — Related Work

Methods based on SAMR start with a coarse base grid that covers the entire computational domain and has the minimum acceptable resolution for the given required numerical accuracy of the PDE solution. As the simulation progresses, local features in the simulation will cause regions in the domain to exhibit unacceptable numerical errors. These regions are identified and the magnitude of the errors is reduced by overlaying grid patches with higher resolution. This refinement process proceeds recursively so that the refined regions requiring higher resolution are similarly identified and even higher resolution grids are overlaid on these regions [4]. The result is a dynamic adaptive grid hierarchy. Software infrastructures for SAMR include Paramesh [20], a FORTRAN library for parallelization of and adding adaption to existing serial structured grid computations, SAMRAI [32] a C++ object-oriented framework for implementing parallel structured adaptive mesh refinement simulations, and GrACE [22], AMROC [7], and CHOMBO[1], all of which are adaptive computational and data-management engines for parallel SAMR.

Parallel SAMR requires repeated partitioning of the dynamic grid hierarchy and subsequent mapping of the partitions onto physical processors. Traditionally, partitioners for SAMR can be classified as *patch-based* or *domain-based*. For patch-based partitioners [3, 16] (PB), distribution decisions are made independently for each newly created grid patch; it may be kept on the local processor or split and distributed (uniformly) over other processors. The SAMR framework SAMRAI [32] (based on the LPARX [2] and KeLP [14] model) supports PB. Domain-based partitioners [23, 30, 26] (DB) partition the physical domain, rather than the grids. The domain is partitioned in a recti-linear fashion along with all overset grids on all refinement levels.

Hybrid partitioners [25, 30, 17] combining patch-based and domain-based approaches, can be used for coping with the shortcomings present in PB and DB. Patch-based partitioners suffer from communication serialization bottlenecks, while granularity constraints prevent DB methods from providing acceptable load balance for deep hierarchies and many processors. *Hierarchical schemes* have been presented as a successful means to overcome synchronization costs [19]. But to reap the benefits of a hierarchical design, these methods need to be hybrid.

2.2 AMROC

AMROC (Adaptive Mesh Refinement in Object-oriented C++) is a dimension- and equation-independent framework for structured mesh adaptation that particularly supports the Berger-Colella SAMR algorithm [4], including conservative flux correction, on parallel computers with distributed memory. The parallelization strategy in AMROC is a generalization of the strictly domain-based space-filling curve approach in GrACE [21]. To derive load balance, the partitions are computed exclusively on the basis of the root level. All patches on higher refinement levels follow this “floor-plan”. The entire hierarchy is considered by projecting the accumulated work from higher levels

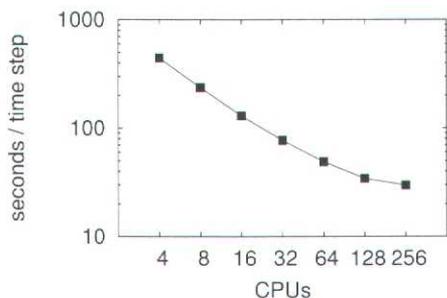


Figure 1. Representative AMROC scale-up test for fixed problem size.

onto the root level cells.

The only operations incurring parallel overhead for the Berger-Colella SAMR algorithm using strictly domain-based partitioning are ghost cell synchronization, redistribution of the SAMR hierarchy and the application of the flux correction terms. Inter-level operations such as interpolation and averaging, but in particular the calculation of the flux corrections, remain strictly local [9, 8].

Figure 1 shows a representative scalability test for AMROC. The test application is a three-dimensional spherical shock wave problem for the Euler equations for a single polytropic ideal gas and employs Roe’s approximate Riemann solver within the multi-dimensional Wave Propagation Method by LeVeque [18] as efficient single-grid scheme. The test was run on the ASC Linux cluster (ALC) at Lawrence Livermore National Laboratories that connects Pentium-4-2.4GHz dual processor nodes with Quadrics Interconnect. The base grid has 32^3 cells and two additional levels with refinement factors 2 and 4. The adaptive calculation uses approx. 7.0M cells in each time step instead of 16.8M cells for a uniform base grid with the equivalent numerical accuracy. The calculation on 16 CPUs employs approx. a total of 1100 grid patches per time-step. Displayed are the average costs for each root level time step, which involves two time steps on the middle level and eight on the highest. All components of the dynamically adaptive algorithm, including regridding and parallel redistribution, are activated to obtain realistic results. The finite volume scheme is incorporated into the C++ framework as a Fortran 77 routine with full compiler optimization. The fraction of the time spent in this Fortran routine are 90.5% on four, still 74.9% on 16 CPUs, but decrease down to 33.0% on 128 CPUs. The mere intra-level synchronization costs increase from 3.5% (4 CPUs) to 27.8% (128 CPUs)¹. As the data volume to be communicated actually decreases, this increase illustrates the influence of rising waiting times due to load imbalances in the core numerical update routine in the strictly domain-based partitioning approach.

2.3 Nature+Fable

Designed on sound SAMR principles, *Nature+Fable* (**Natural Regions + Fractional blocking and bi-level partitioning**) [25, 27, 15] is a hybrid, dedicated SAMR partitioning tool that brings together

¹Note that the necessarily increasing costs for setting up synchronization data structures are measured separately.

the advantages of both patch-based and domain-based partitioning approaches while avoiding their drawbacks. It separates homogeneous, un-refined (Hue) and complex, refined (Core) domains of the grid hierarchy. 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, *i.e.*, each Core contains a portion of the base grid and all its overlaid, refined grids. The Cores are then clustered into *bi-levels* [25]. A bi-level consists of patches from two adjacent refinement levels — a patch in the grid hierarchy at level k together with all its superimposed refinements at level $k + 1$. From the bi-levels, coarse “easy-to-block” partitions are created. Finally, expert blocking algorithms are used for the Hues and the coarse partitions in the Cores. To enable optimal partitioning for arbitrary application-computer combinations, the partitioning process is controlled by a large set of parameters.

3 The Improved Bi-Level Algorithm

3.1 The Original Approach

Nature+Fable groups refinement levels two by two. A *bi-level* is a patch — a parent — at the lower level in a group together with all its children. Bi-levels are transformed into coarse partitions, suitable for blocking. We define the original approach as the *child-driven approach* (CDA). The steps in the CDA are:

1. **Separation** Separate the children patches from each other: Without cutting any of the children, split up the parent in a domain-based fashion so each piece gets a maximum of one child (Figure 2, Left).
2. **Cleaning** Remove “white-space”: Cut off excessive parts of the parent if it has a relatively “small” child (Figure 2, Middle).
3. **Blocking** Send the coarse partitions to the blocker and *let the child (if there is one) dictate the cuts* (Figure 2, Right).

The CDA (illustrated in Figure 2) generally generates high-quality results for all inputs it can possibly handle. Quality does not deteriorate as the inputs get “harder”. Rather, there is a hard cut-off beyond where it can not go. Therefore, we wish to keep the CDA as a the corner-stone in the improved algorithm.

The CDA sometimes fail because the children patches are laid out in a way that makes the separation step impossible. For example, if the parent is “cluttered” with children, it is unlikely that there will be any clean cuts for the separation: it will be impossible to cut the parent without simultaneously cutting at least one of the children. Another problem is that the occurrence of children of the smallest possible size — *i.e.* the *atomic unit* — may result in that the only possible cuts create patches smaller than the atomic unit on the parent.

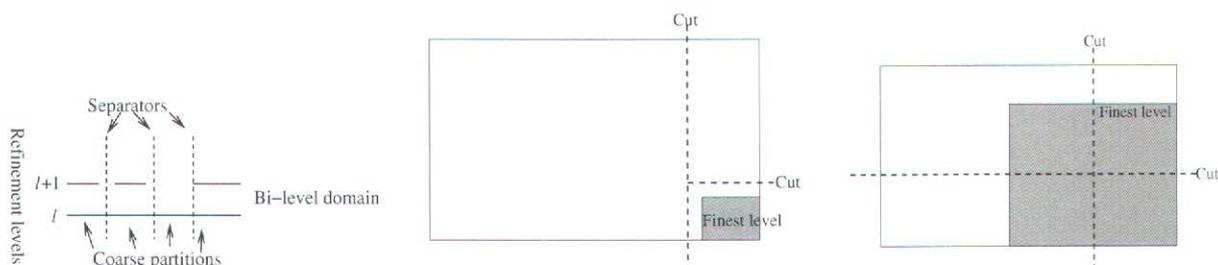


Figure 2. The three steps of the original approach (CDA): 1) Separation, 2) Cleaning, and 3) Blocking.

3.2 The Complementary Approach

We can now identify two criteria for a complementing approach: 1) It should cope with all possible bi-levels (as it will be fed all bi-levels that the CDA could not handle), and 2) it should generate high-quality results for bi-levels badly suited to the CDA.

To design a successful complementing approach, further analysis of the “impossible” bi-levels is needed. We list the two basic types of impossible bi-levels:

1. **Circular children pattern** If the children form a “circular pattern” as illustrated by Figure 3 (Left), there is no possible separation cut that does not cut through at least one of the children.
2. **Children of atomic unit size with close neighbors** If a child of atomic unit size resides at the border of the parent *and* has an immediate neighbor (illustrated by Figure 3, Right), these two children cannot be separated without creating a piece of the parent smaller than the atomic unit.

We make the following observation. Though not exclusively, both of the types listed above occur for many and spread-out children often including small patches. An implication of this observation is that for “impossible” bi-levels, there is a good chance that the children a) are reasonably spread-out, b) are many, and c) includes patches of atomicunit size. We design our complementing approach on the basis of this observation.

The complementing approach should be most efficient for exactly the bi-levels described by our observation. For now, we leave (c) for later and focus on (a) and (b) from the paragraph above. We make a further observation: A bi-level having many and spread-out children might not only be impossible to handle for the CDA. It may also be *unnecessary* to handle such a bi-level with the CDA. There may be a simpler way, exploiting the given characteristics. If the children are many and spread-out, a high-quality result should follow automatically by the following simple approach, which we define as the *parent-driven approach* (PDA) (illustrated in Figure 4):

1. **Cleaning** Make a bounding box around the children and re-use the original cleaning algo-

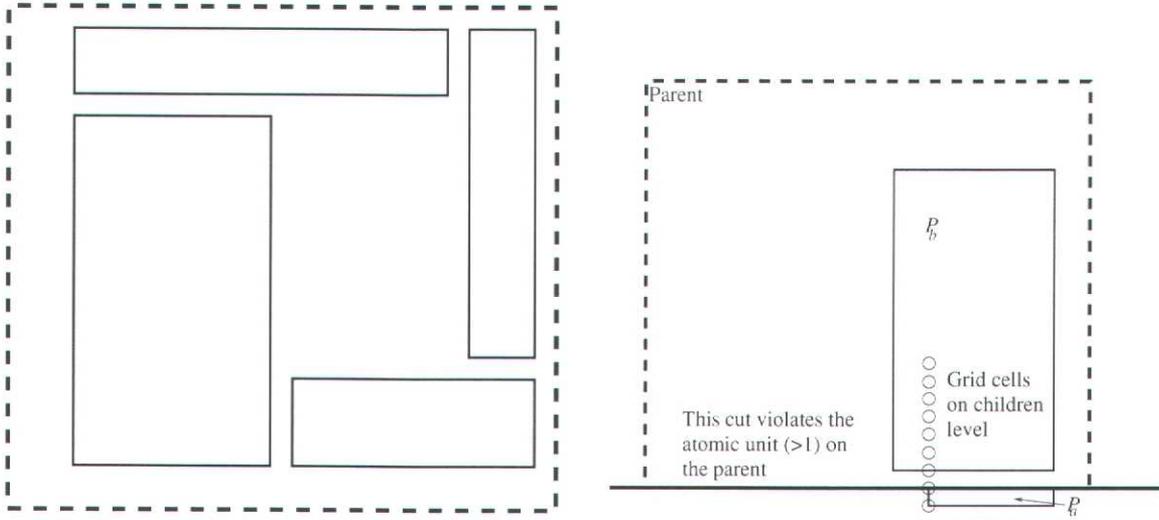


Figure 3. Two “impossible” bi-levels. Left: The “circular” children pattern. Right: A patch of atomic unit size P_a with an immediate neighbor P_b .

rithm (described in the previous sub-section) with the bounding box as the child (Figure 4, Left).

2. **Blocking** Block the parent and *let the parent alone dictate the cuts*. In this step we only regard the children’s combined workload. All other aspects of the children — such as position, size, and number — are disregarded (Figure 4, Middle).
3. **Derive kids** For each block, re-use the original `getKids` function to derive the block’s children (Figure 4, Right).

Letting the parent dictate the cuts is logical considering the class of input: many and spread-out children patches. Statistically, this would produce a reasonably well-balanced partitioning. Note that for the PDA, the result has a different structure compared to the CDA. The PDA generates bi-level blocks with one parent and potentially multiple children, while the CDA generates bi-level blocks with one or zero blocks per parent. This poses challenges in data structures and data handling, as will be discussed in Section 3.4.

3.3 The Classification Algorithm

Given the PDA, the task is now to design an efficient algorithm that classifies a given bi-level as suitable for either the CDA or the PDA. This algorithm’s job is to switch automatically and transparently between the two approaches, always producing a high-quality result.

As the CDA generally gives high-quality results, a goal for the classification algorithm is to classify

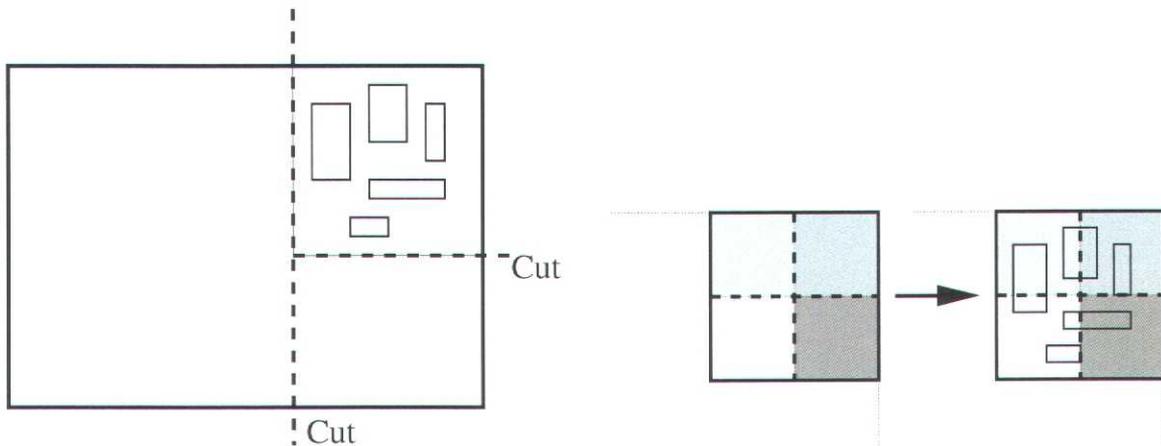


Figure 4. The three steps of the complementary approach (PDA): 1) Cleaning, 2) Blocking, and 3) Derive kids.

as many bi-levels as possible as suitable for the CDA. A crucial constraint is that it should never classify an “impossible” bi-level to the CDA as this will cause `Nature+Fable` to fail. We also want the classification algorithm to conform to the general `Nature+Fable` algorithm policy regarding efficiency, *i.e.*, for significantly complex tasks we prefer powerful heuristics over searching for an exact or optimal solution.

While deriving and computing mathematical quantities for classifying a bi-level might be possible and bears the potential of rigorous results, we considered this prohibitively expensive. Instead, we constructed a conceptually trivial, yet powerful, recursive algorithm:

```

Classify(bi-level BL)
{
  if (triviallySuitableForPDA(BL)) doPDA(BL);
  else if (triviallySuitableForCDA(BL)) doCDA(BL);
  else
  {
    /*=== This the non-trivial case ===*/
    if (split(BL, BLLow, BLHigh)==FAIL) doPDA(BL);
    else
    {
      /*=== Recursive case ===*/
      Classify(BLLow);
      Classify(BLHigh);
    }
  }
}

```

This algorithm singles out the trivially classifiable bi-levels. If trivial classification fails, the bi-level is split and both resulting parts are classified recursively. The strength in this classic divide-and-conquer algorithm is that it simultaneously separates and classifies the input bi-level. It re-uses the split function used by the CDA, meaning that each split is equivalent to the optimal separation cut. Thus each recursive call will generate a meaningful break-down (separation) of the input, effectively decreasing the number of bi-levels that would unnecessarily be classified as impossible.

The challenge inherent in this algorithm is to define the suitable predicates and determine the numerical thresholds associated with each predicate. First, we consider what is *trivially suitable for the PDA*. We identify the following sufficient conditions:

- The children pattern should be very dense, OR
- There should be a lot of children, exhibiting a not-very-sparse pattern, OR
- The children should be on average very small, exhibiting a not-very-sparse pattern.

The following line of C code implements the above conditions as the macro predicate `triviallySuitableForPDA`:

```
#define triviallySuitableForPDA \  
    (VERY_DENSE) || ((A_LOT) && !(VERY_SPARSE) ) ||\  
    ((VERY_SMALL) && !(VERY_SPARSE)))
```

The above macro, in turn, relies on a set of abstractions, which are implemented as C macro predicates, too. They will be defined below.

Now, we consider what is *trivially suitable for the CDA*. We identify the following sufficient conditions:

- The bi-level has only one child, which is not of atomic unit size, OR
- The bi-level has very few and large children and there is no patch of atomic unit size.

The following line of C code implements the above conditions as the macro predicate `triviallySuitableForCDA`:

```
#define triviallySuitableForCDA \  
    ( ((kids.number()==1) || ((VERY_FEW) && (VERY_LARGE))) \  
    && !existsAtomic)
```

As for the `triviallySuitableForPDA`, the abstractions will be defined below. Note how this predicate addresses point (c) from Section 3.2, *i.e.*, bi-levels containing children of atomic unit size

can not be classified as suitable for the CDA. Instead of automatically and possibly unnecessarily being classified as suitable for the PDA, the bi-levels that fail this predicate will be split and the two pieces will be classified recursively.

To define the six above described abstractions as C macro predicates, we rely on the statistical quantities described in Table 1. These quantities give information about the children’s average size and so forth, and are derived by looping once through the bi-level’s children. We define the remaining six macro predicates with these lines of C code:

```
#define VERY_SPARSE ( (howMany<REALLY_SPARSE) || \
                      (howMany<SPARSE && \
                       avgSize[RELATIVE]<2*TINY_RELATIVE) )
#define VERY_DENSE (howMany>DENSE)
//VERY_SMALL means very small but NOT tiny!
#define VERY_SMALL ( (avgSize[ABSOLUTE]<SMALL_ABSOLUTE && \
                     avgSize[ABSOLUTE]>TINY_ABSOLUTE)\
                     ||\
                     (avgSize[RELATIVE]<SMALL_RELATIVE && \
                      avgSize[RELATIVE]>TINY_RELATIVE) )
#define VERY_LARGE ( (avgSize[ABSOLUTE]>LARGE_ABSOLUTE) ||\
                     (avgSize[RELATIVE]>LARGE_RELATIVE) )
#define VERY_FEW ( (kids.number()<FEW_ABSOLUTE) )
//A_LOT means a lot but NOT a myriad!
#define A_LOT ( (kids.number()>MANY_ABSOLUTE && \
               kids.number()<MYRIAD_ABSOLUTE) )
```

The six above defined macro predicates use 12 constants, *i.e.*, thresholds that define what is considered *sparse*, *dense*, and so forth. These constants are listed in Table 2 along with their numerical thresholds. The numerical values are a results of a lengthy trial-and-error process. Further “tweaking” of the numbers is expected.

3.4 Implementation

Figure 5 offers a schematic view over the improved algorithm, showing the relationship between the old and new functions, and how the old functions are re-used by the new improved algorithm. Even though both the PDA and the classification algorithm are new, they required minimal additional code. The classification algorithm is as described above of trivial divide-and-conquer type. And as the PDA is comprised of the three steps *cleaning*, *blocking*, and *derive kids*, it re-uses these function calls as they already exists in Nature+Fable.

The primary data structure in Nature+Fable is the `GridBoxList`: a linear list of elements of the type `GridBox`. To accommodate a coarse partition, *i.e.*, a parent and its child, the Nature+Fable `GridBox` has been extended from the standard version in GrACE to include an additional `BBox`.

Quantity	Value	Explanation
<code>kids.number()</code>	The number of children	
<code>existsAtomic</code>	1 if there exists an atomic unit-sized patch and 0 otherwise	
<code>howMany</code>	<i>(The number of children) / (The maximum number of children possible for the parent)</i>	Indication of children pattern's density
<code>avgSize[ABSOLUTE]</code>	<i>(average child size) / (the size of the atomic unit)</i>	How many times bigger is the average child compared to the smallest possible child
<code>avgSize[RELATIVE]</code>	<i>(average child size) / (parent size)</i>	How large is the average child compared to the parent

Table 1. The statistical quantities used for defining the macro predicates used for classification of bi-levels

The choice of this data structure is a result of the original bi-level approach, *i.e.*, the CDA. As the PDA — as opposed to the CDA — produces parents having potentially multiple children, the requirement for the data structures changes. Instead of each `GridBox` having *one* additional `BBox`, the logical choice of data structure would be to include an additional `BBoxList` for each `GridBox` in the `GridBoxList`.

In retrospect, `Nature+Fable` should probably have been designed on the basis of a more general data structure. But a drastic modification of data structures in a thousands-of-lines software tool often requires re-engineering large parts of the tool. To avoid the daunting task of re-engineering `Nature+Fable`, we searched for an alternative solution.

The solution to the problem with the data structures was simpler than expected: For the PDA, each child is stored as a separate `GridBox` and is given the same space-filling curve (SFC) index as its parent. Having identical SFC indices forces the parent and its children to stay together through the subsequent partitioning steps (e.g. sorting). Each `GridBox` has an integer used internally by `Nature+Fable` as a flag. For the PDA, the children and the parent are flagged as `MULTI`, whereas for the CDA each `GridBox` is flagged as `SINGLE`. The flag is used by the subsequent mapping routines to correctly identify all kinds of `GridBoxes`.

Constant	Threshold	Used for
TINY_ABSOLUTE	10.0	avgSize[ABSOLUTE]
TINY_RELATIVE	0.001	avgSize[RELATIVE]
SMALL_ABSOLUTE	80.0	avgSize[ABSOLUTE]
SMALL_RELATIVE	0.005	avgSize[RELATIVE]
LARGE_ABSOLUTE	100.0	avgSize[ABSOLUTE]
LARGE_RELATIVE	0.20	avgSize[RELATIVE]
FEW_ABSOLUTE	4	kids.number()
MANY_ABSOLUTE	5	kids.number()
MYRIAD_ABSOLUTE	25	kids.number()
REALLY_SPARSE	0.002	howMany
SPARSE	0.003	howMany
DENSE	0.004	howMany

Table 2. The definition of these thresholds is a result of lengthy trial-and-error

4 Experimentation

4.1 Hypotheses

We have the following four hypotheses:

- H1 Using the improved bi-level algorithm, `Nature+Fable` can now cope with arbitrary and realistic grid hierarchies.
- H2 The improved algorithm — including the automatic switching between the PDA and the CDA — is “better” than using the PDA alone.
- H3 `Nature+Fable` will generate significantly better load balance than the native AMROC domain-based SFC partitioner without significantly increasing box count.
- H4 As `Nature+Fable` is a hybrid partitioner, it will separate parent patches from their kids whenever this might be justified. Consequently, there will be a significant increase in communication volume as compared to the native AMROC domain-based SFC partitioner.

4.2 Experiments

To determine which of our four hypotheses (if any) could be verified, we used two complex and realistic applications from the structured adaptive mesh refinement framework AMROC. Using

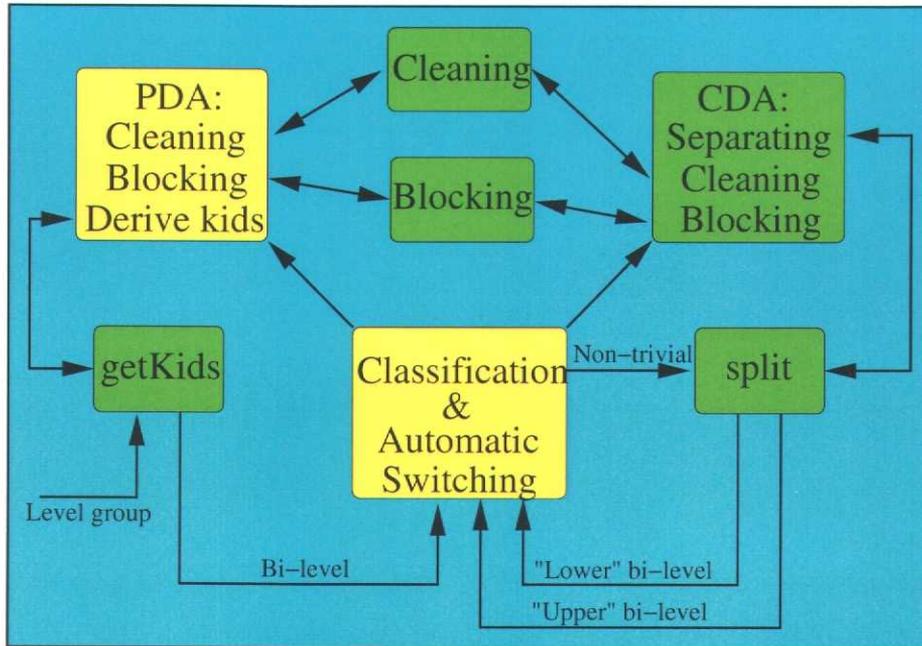


Figure 5. A schematic view over the improved bi-level partitioning algorithm. New functions are yellow and old are green. The figure shows the relationship between the new and old functions, and how the old functions are re-used by the improved algorithm.

16 processors, we compared the improved bi-level algorithm to PDA alone, and then compared Nature+Fable (using our improved algorithm) to the native partitioner in AMROC.

The applications

Two two-dimensional AMROC applications with a topological complexity comparable to the scalability test in Section 2.2 have been selected for this study. Table 3 lists their key properties.

The first application, *Ramp2d*, is a double Mach reflection of a Mach-10 shock wave impinging on a 30 degree wedge (also used by Berger and Colella [4]). In serial, this application uses 293 grid patches per time-step on average. There are three additional levels of refinement. Figure 6 depicts the final state of the hierarchy.

The second application, *ConvShock2d*, is a converging Richtmyer-Meshkov instability. In this simulation, a spherically converging Mach-5 shock hits an initially perturbed spherical interface between two different gases and drives it inward. The shock is reflected around $t \approx 0.3$ in the origin and inverts the direction of motion of the interface as it hits it again. The simulation is motivated by mixing phenomena related to inertial confinement fusion. Figure 7 shows the early stages of the simulation: the refinement is clustered around the origin. As a consequence of this refinement pattern, the application is particularly challenging for the PDA. In serial, this simulation uses 745 grid patches per time-step on average. There are four additional levels of refinement.

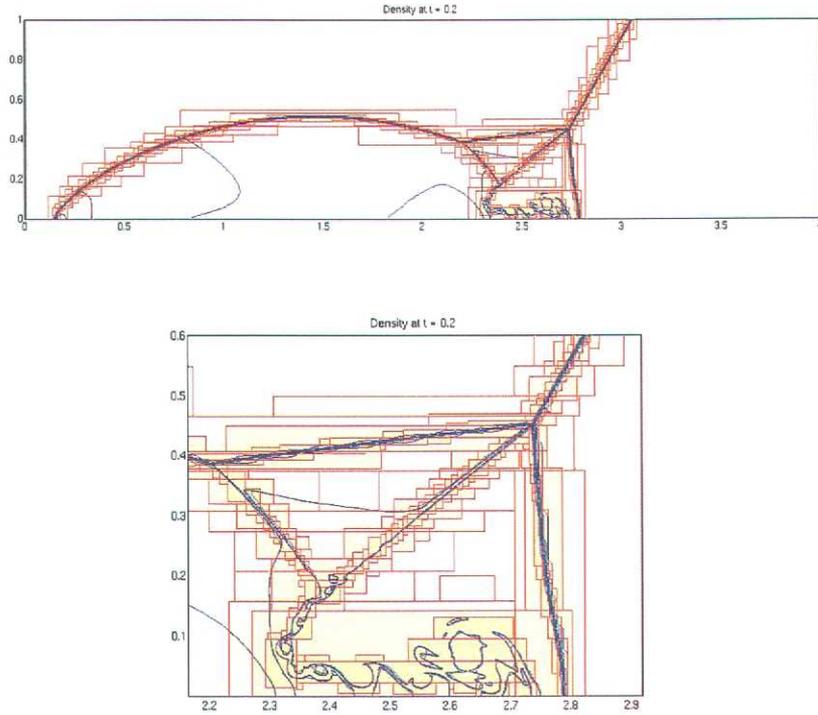


Figure 6. Upper: Iso-lines of density on refinement levels at the final time step of *Ramp2d*. Lower: An enlargement of the relevant triple region.

Both simulations use the second-order accurate multi-dimensional Wave Propagation Method with MUSCL slope limiting in the normal direction [8] in combination with linearized Riemann-solvers of Roe-type.

Application	Tot. levels	Ref. factors	Avg. boxes	Steps
Ramp2d	4	2,2,4	292.6	339
ConvShock2d	5	2,2,4,2	744.8	228

Table 3. Key properties of the AMROC application traces.

The SAMR simulator

Load imbalance, box count, and communication amount are derived using software [24] 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, and the number of processors. The trace is then run and the performance of the partitioning configuration at each regrid step is computed using the above mentioned metrics. For each coarse time-step, load imbalance is defined as the load of the heaviest loaded processor divided by the average load, box count as the average number of boxes per processor, and communication amount as the maximum number of ghost cells required to be transferred by any single processor.

4.3 Results

The improved bi-level algorithm vs. PDA alone

As can be seen in Figures 8 and 9, only load imbalance differs significantly. Clearly, the automatic switching has some success in classifying the current bi-level group and selecting the correct approach to partitioning it.

Nature+Fable vs. AMROC DB-SFC

As can be seen in Figures 10 through 13, Nature+Fable with the improved algorithm can cope with all grid hierarchies. Load imbalance is significantly lower — about half — as compared to AMROC DB-SFC, while box count on average is similar. Communication volume, on the other hand, is 4 to 4.5 times higher for Nature+Fable.

4.4 Discussion: More Communication, Less Load Imbalance

The results show that Nature+Fable generates significantly more communication but also significantly less load imbalances as compared to the native AMROC DB-SFC. To put these results into perspective, we investigated the impact of perfecting load balance while increasing the amount of communication for our two applications with 16 processors used on two different parallel platforms: ACL² and SHC³. The communication volume was increased by a factor $f \in \{1, 2, 3, 4, 5\}$ by increasing the ghost cell buffer width with the same factor. Note that the partitioning is independent of f as it disregards ghost cells. Table 4 shows the resulting synchronization times, including both actual communication time and wait time. The results were extrapolated to create the numbers for $f = 0$.

As both platforms have high-performance networks, no significant wait time should occur during communication for four to eight nodes and perfect load balance. Thus, $f = 0$ corresponds to net wait times caused by load imbalances generated by the AMROC DB-SFC. Assuming perfect

²Pentium 4 2.4GHz dual processor nodes with Quadrics High Speed Interconnect.

³Quad processor, dual core, 2.2 GHz AMD Opteron, PCI-X 4x Infiniband.

f	Ramp ALC	ConvShock ALC	Ramp SHC	ConvShock SHC
0	680	1800	510	1400
1	766	2263	560	1764
2	900	2884	640	
3	986	3715	728	
4	1143	4731	817	3404
5	1278	5904	908	4284

Table 4. Actual total synchronization time, including both communication and wait, in seconds for the two applications on both ACL and SHC using AMROC DB-SFC and 16 processors. Communication volume is increased by the factor f . The row for $f = 0$ contains extrapolated values, corresponding to net wait times caused by load imbalances. Note that $f = 1$ indicates using the applications’ original ghost cell buffer width.

load imbalance translates to subtracting the row for $f = 0$ from all other rows. The result from this subtraction is shown in Table 5. These numbers could be viewed as communication times for a fictitious partitioner generating perfect load balance but f times more communication than AMROC DB-SFC.

f	Ramp ALC	ConvShock ALC	Ramp SHC	ConvShock SHC
0	0	0	0	0
1	86	463	50	364
2	220	1084	130	
3	306	1915	218	
4	463	2931	307	2004
5	598	4104	398	2884

Table 5. Communication time — assuming perfect load balance — computed as total synchronization time minus the extrapolated net wait times in Table 4. Note that the communication time for Nature+Fable would roughly correspond to $f = 4$ as the communication volume for Nature+Fable was about 4 times higher and load balance was significantly better as compared to AMROC DB-SFC.

Table 6 shows a comparison between the actual DB-SFC and the fictitious partitioner generating perfect load balance but four times as much communication. The table shows that the combination perfect load balance/four times more communication is beneficial for Ramp2d but not for ConvShock2d. This is perhaps due to that ConvShock2d has roughly three times as many synchronizations per time step as Ramp2d. Moreover, ConvShock has a default ghost cell buffer width of three, whereas Ramp2d has only two.

So far, we have only considered the time spent in synchronization. To put these times into context,

Descr.	Ramp ALC	ConvShock ALC	Ramp SHC	ConvShock SHC
$f = 1$ Actual SFC	766	2263	560	1764
$f = 4$ Balanced Extrapolated	463	2931	307	2004

Table 6. Comparison of the actual total synchronization time for AMROC DB-SFC (top row) to the estimated synchronization time for perfect load balance but with communication volume increased by a factor of 4 (bottom row).

they have to be compared to the total run-time. The percent of total run time spent in synchronization is listed in Table 7.

Ramp ALC	ConvShock ALC	Ramp SHC	ConvShock SHC
26	26	27	24

Table 7. Percent of total run-times spent in synchronization for AMROC DB-SFC.

The above experiments show that the combination perfect load balance/four times more communication for a regular Berger-Collela SAMR application such as Ramp2d can reduce the synchronization costs by up to about 50 percent. If the fraction of the time spent in synchronization is about 25 percent as for our applications, this gain will translate to better than a ten percent improvement of overall run time. Conversely, for more complex applications requiring more frequent synchronization and has a higher communication-to-computation demand, the partitioning focus needs to be on the communication volume. An increased communication volume by a factor of four would for these applications be devastating to parallel performance.

Before relating the above reasoning to `Nature+Fable`, we must consider that `Nature+Fable` does not generate perfect load balance. Hence, it would perform slightly worse than the fictitious partitioner. On the other hand, the bulk of the communication increase for `Nature+Fable` comes from introducing *inter-level* communication, whereas only *intra-level* communication was increased in the experiments. As inter-level communication generally requires larger data sets in fewer packages than intra-level communication, `Nature+Fable` is expected to perform slightly better than the fictitious partitioner. Assuming that one minus and one plus roughly cancel each other, `Nature+Fable` is expected to perform in the neighborhood of the fictitious partitioner.

The conclusion is that the performance of Berger-Collela-type applications, like Ramp2d, can be significantly improved by using `Nature+Fable`. However, `Nature+Fable` needs to improve in communication to be considered a solution to scalability problems for general SAMR applications.

5 Summary and Conclusions

As a result of the improved bi-level partitioning algorithm presented in this document, `Nature+Fable` now successfully copes with realistic applications. The recursive classification algorithm divides the bi-levels and automatically switches between the parent-driven or the child-driven partitioning approach.

As compared to the native AMROC DB-SFC partitioner, `Nature+Fable` generates significantly less load imbalance, similar box count, but 4-4.5 times more communication.

As we move away from strictly domain-based partitioning, we introduce inter-level communication. This component exists for `Nature+Fable` but not for SFC. Thus, it is non-trivial to quantify the impact of the actual numbers. For some computer-application combinations, like the Ramp2d on ALC and SHC, `Nature+Fable` is expected to improve performance. For other combinations, using the AMROC DB-SFC or some other technique that better optimizes the pertinent metrics, gives better performance.

To be considered a solution to the scalability problem for general SAMR applications, `Nature+Fable` must lower the generated communication volume.

6 Future Work: A New Bi-Level-to-Processor Mapping

For some computer-application combinations, communication pattern needs to be specifically addressed and optimized. The current mapping between bi-level partitions and processors in `Nature+Fable` is insufficient as it clearly cannot compete with the domain-based partitioners. This section discusses how to improve the block-to-processor mapping in `Nature+Fable`.

6.1 The Current Mapping in `Nature+Fable`

Consider the simple 1D SAMR hierarchy in Figure 14. Level 0 and level 1 constitute the bottom level group, and level 2 and level 3 constitute the top level group. Assume four processors and note that the SFC mapping in 1D translates to the 1D grid coordinates.

In this figure, the bottom level group (level 0 and 1) is split up in four pieces, one per processor. These pieces are ordered according to the SFC. This means that they are ordered simply left to right (red, green, blue, and yellow). Then, the top level group is treated analogously: It is split up in four pieces, and these pieces are ordered according to the very same SFC (left to right: red, green, blue, and yellow).

The particular characteristic of this hierarchy is that the top level group is similar in composition and spacing to the lower level group. Due to this similarity, the SFC mapping is highly successful (optimal) as no inter-level communication is introduced.

Now, consider another simple 1D SAMR hierarchy depicted in Figure 15. In contrast to the previous hierarchy, the particular characteristic of this hierarchy is that the top level group is *dissimilar* to the bottom level group. The partitioning procedure is exactly the same as for the previous hierarchy: *Nature+Fable* splits the level groups in four pieces and orders these pieces according to the SFC.

For this hierarchy, only the blocks for the yellow processor were optimally mapped. Studying the figure, we realize that it would have been better to switch places in the top level group between the green (or red) and the blue processor pieces. This switch would eliminate inter-level communication for half of the data on the blue processor.

The problem with “misplaced” blocks is addressed by a heuristic re-mapping algorithm [27]. We note a few things:

- The SFC only *orders* blocks (relative to each other) within a level group.
- Blocks are mapped to processors exclusively based on this order (SFC index is disregarded).
- The two points above are the reason for the occurrence of sub-optimal mappings as in the second example above.
- Realistic hierarchies are 2D or 3D. As the dimensionality increases, the proximity preservation property of the SFC decreases. Thus, the two hierarchies and partitionings described above can be viewed as best-case scenarios for 2D and 3D.
- Even with an optimal SFC mapping we still incur large amounts of inter-level communication between level groups in the second example. This inter-level communication is between blue and red, and between yellow and green processors.
- The implication of the point above is: The price we pay for leaving DB partitioning is increased communication volume. For every single block we “orphan”, we may gain load balance but lose in communication. This is inevitable and independent of the SFC used.
- Even in the light of the above, or perhaps *because of* the points made above, there is need to create a new or improve on the present SFC mapping algorithms in *Nature+Fable*.

6.2 Ideas for a A New Mapping

Studying the sub-section above, we note that

1. In the first example, *Nature+Fable* created a strictly DB partitioning.
2. In the second example, a strictly DB partitioning would probably have given unreasonable load imbalance.

From (1) and (2) above we draw the following conclusions.

1. An optimal mapping in Nature+Fable should be such that a strictly DB partitioning is automatically generated for suitable grids.
2. For grids *not* suitable for strictly DB partitioning, an optimal mapping in Nature+Fable should guarantee a minimal amount of “orphaning” *given the current bi-levels blocks*.

The current mapping in Nature+Fable does *not* guarantee (1). Regardless of the quality of the SFC, there is no guarantee that the blocks on the higher levels will be mapped to beneficial processors if the composition and location of the boxes are not identical. On average, the more dissimilar these refinement patterns are, the more the mapping will differ.

For exactly the same reasons as for (1), the current mapping in Nature+Fable cannot guarantee (2) either. But the remapping function significantly improves on the results.

The conclusion so far is that it is not the quality of the current SFC implementation that causes the problems. Rather, the problems arise from the way the SFC mapping is used. Also, note that the problems discussed are independent of whether the SFC mapping is fully or partially ordered. Again, the problems arise in the mapping of the bi-levels to processors, based on the SFC *ordering*.

So the first remedy is not to implement an SFC index generator with perhaps better proximity preservation properties for general SAMR (even if this will perhaps be necessary eventually). Rather, the remedy should address how the SFC is used.

Below, we sketch two strategies for a remedy:

Strategy 1: Instead of letting the SFC order blocks (bi-levels) relatively to each other, we could look for ways to let the SFC tie blocks to processors. Put differently: Now, the mapping is *block relative*. It should really be more *processor absolute*. So instead of a mapping from $N \times M \rightarrow NM$ (and do the processor assignment based on the resulting order), we should look for a more direct mapping $N \times M \rightarrow P$, where P is the number of processors. Such a mapping would guarantee that a block straight on top of another block will be mapped onto the same processor regardless of the refinement patterns on these levels.

There are numerous problems with this strategy. The most prominent one is that it in its simplest form will always generate strictly DB partitionings. Thus, there must be a way to re-map after (or refine) this initial mapping. Performing these steps, in turn, is ridiculous as this strategy would be equivalent to standard SFC DB partitioning followed by a refinement step.

Strategy 2: We could use a SFC of higher dimensionality (tailored to the specific SAMR hierarchical structure). Then we should not map level group by level group, but rather bi-level by bi-level block as ordered by the (new) SFC.

This strategy also has numerous problems. The most prominent one is that the hierarchy *should* be mapped in a level (group) by level (group) fashion as the SAMR algorithm executes in a level

by level fashion (patches on level l are advanced before patches on level $l + 1$). Disregarding the computational order will allow for “perfect” load balance, which in practice is meaningless.

To conclude, we note that both strategies seem to hold great promise but are also inherently problematic. We hope to address this problem thoroughly in the future.

References

- [1] CHOMBO. <http://seesar.lbl.gov/anag/chombo/>, NERSC, ANAG of Lawrence Berkeley National Lab, CA, USA, 2003.
- [2] Scott B. Baden, Scott R. Kohn, and S. Fink. Programming with LPARX. Technical Report, University of California, San Diego, 1994.
- [3] Dinshaw Balsara and Charles Norton. Highly parallel structured adaptive mesh refinement using language-based approaches. *Journal of parallel computing*, (27):37–70, 2001.
- [4] M. J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics*, 82:64–84, 1989.
- [5] G. Bryan. Fluids in the universe: Adaptive mesh refinement in cosmology. *Computing in Science and Engineering*, pages 46–53, 1999.
- [6] Matthew W. Choptuik. Experiences with an adaptive mesh refinement algorithm in numerical relativity. *Frontiers in Numerical Relativity*, pages 206–221, 1989.
- [7] Ralf Deiterding. AMROC. <http://amroc.sourceforge.net/> VTF, California Institute of Technology, CA, USA, 2003.
- [8] Ralf Deiterding. *Parallel adaptive simulation of multi-dimensional detonation structures*. PhD thesis, Brandenburgische Technische Universität Cottbus, Germany, 2003.
- [9] Ralf Deiterding. Construction and application of an amr algorithm for distributed memory computers. volume 41, pages 361–372. Springer Verlag, 2005.
- [10] Fredrik Edelvik and Gunnar Ledfelt. A comparison of the GEMS time domain solvers. In Fredrik Edelvik et al., editors, *EMB 01—Electromagnetic Computations—Methods and Applications*, pages 59–66, Uppsala, Sweden, November 2001. SNRV.
- [11] Jaideep Ray et al. Triple flame structure and dynamics at the stabilization point of an unsteady lifted jet diffusion flame. In *Proceedings of the Combust. Inst. 2000*, 25(1):219–226, 2000.
- [12] Jiuxing Liu et al. Performance comparison of mpi implementations over infiniband, myrinet and quadrics. In *Proceedings of Supercomputing*, 2003.
- [13] T. Wilhelmsson et al. Increasing resolution and forecast length with a parallel ocean model. In *Operational Oceanography — Implementation at the European and Regional Scales*. Elsevier, 1999.

- [14] Stephen J. Fink, Scott B. Baden, and Scott R. Kohn. Flexible communication mechanisms for dynamic structured applications. In *Proceedings of IRREGULAR '96*, pages 203–215, 1996.
- [15] Henrik Johansson and Johan Steensland. A characterization of a hybrid and dynamic partitioner for SAMR applications. In *Proceedings of PDCS2004*, 2004.
- [16] Z. Lan, V. Taylor, and G. Bryan. Dynamic load balancing for structured adaptive mesh refinement applications. In *Proceedings of ICPP 2001*, pages 571–579, 2001.
- [17] Z. Lan, V. Taylor, and G. Bryan. Dynamic load balancing of SAMR applications on distributed systems. In *Proceedings of Supercomputing 2001*, pages 36–48, 2001.
- [18] R. J. LeVeque. Wave propagation algorithms for multidimensional hyperbolic systems. *J. Comput. Phys.*, 131(2):327–353, 1997.
- [19] X. Li and M. Parashar. Hierarchical partitioning techniques for structured adaptive mesh refinement applications. *Journal of Supercomputing*, 28(3):265–278, 2004.
- [20] Peter MacNeice et al. PARAMESH: A parallel adaptive mesh refinement community toolkit. *Computer physics communications*, (126):330–354, 2000.
- [21] 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.
- [22] 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.
- [23] Jarmo Rantakokko. *Data Partitioning Methods and Parallel Block-Oriented PDE Solvers*. PhD thesis, Uppsala University, 1998.
- [24] 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.
- [25] Johan Steensland. *Efficient partitioning of dynamic structured grid hierarchies*. PhD thesis, Uppsala University, 2002.
- [26] 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.
- [27] 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.

- [28] 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.
- [29] Johan Steensland and Jaideep Ray. A partitioner-centric classification space for SAMR partitioning trade-offs : Part II. In *Proceedings of the 2004 international conference on parallel processing workshops*, pages 231–238. IEEE Computer Society, 2004.
- [30] M. Thuné. Partitioning strategies for composite grids. *Parallel Algorithms and Applications*, 11:325–348, 1997.
- [31] 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*, pages 31–38, June 1997.
- [32] Andrew M. Wissink et al. Large scale parallel structured AMR calculations using the SAM-RAI framework. In *proceedings of Supercomputing 2001*, pages 6–6, 2001.

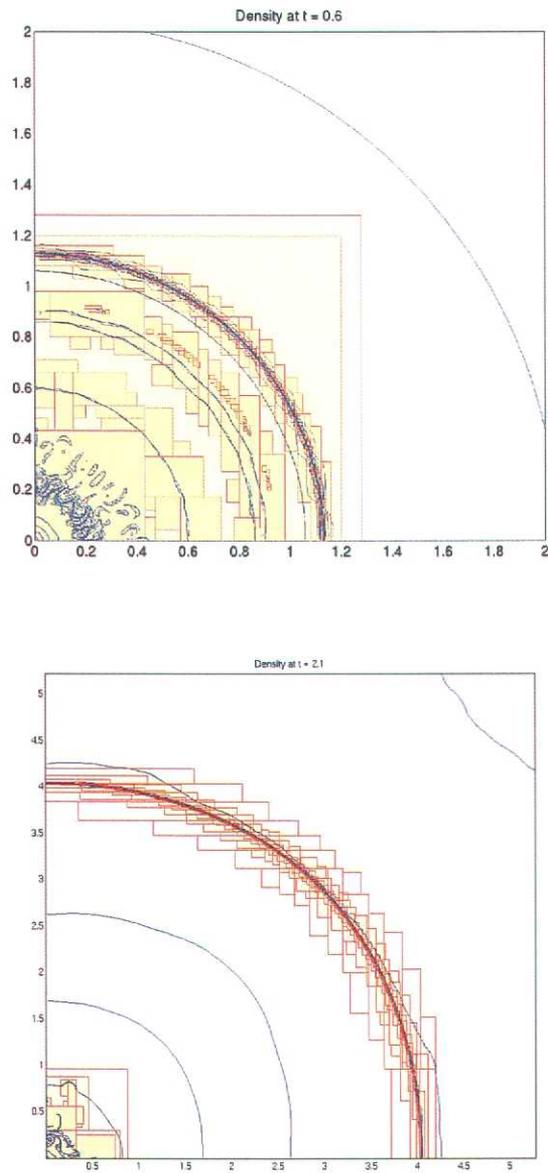


Figure 7. Iso-lines of density on refinement levels at different time step of *ConvShock2d* show the expansion of the shock wave after the reflection at $t \approx 0.3$ in the origin.

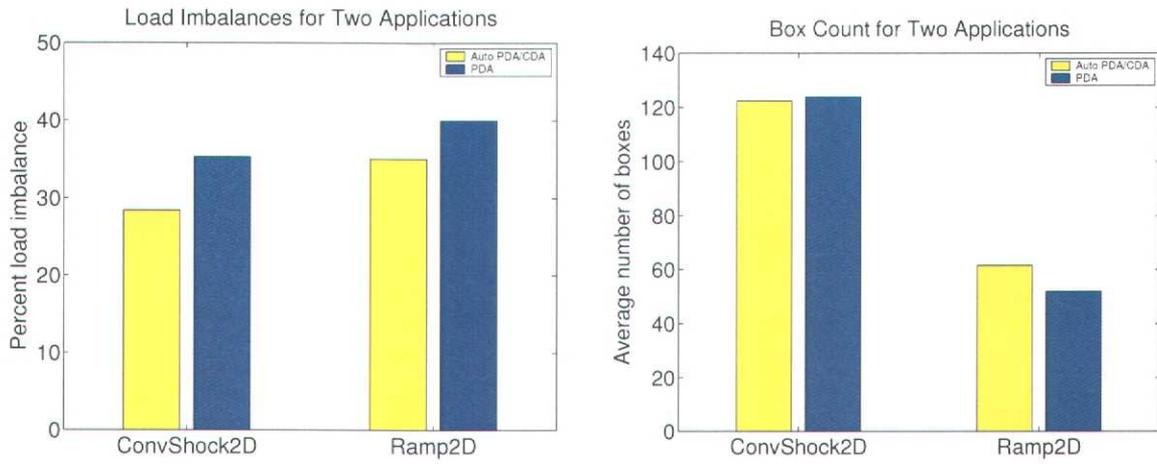


Figure 8. The improved bi-level algorithm versus PDA alone for Ramp2d and ConvShock2d for 16 processors: Load imbalance (Left) and box count (Right).

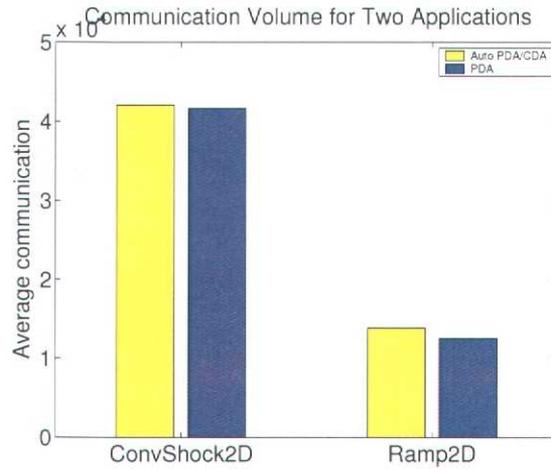


Figure 9. The improved bi-level algorithm versus PDA alone for Ramp2d and ConvShock2d for 16 processors: Communication volume.

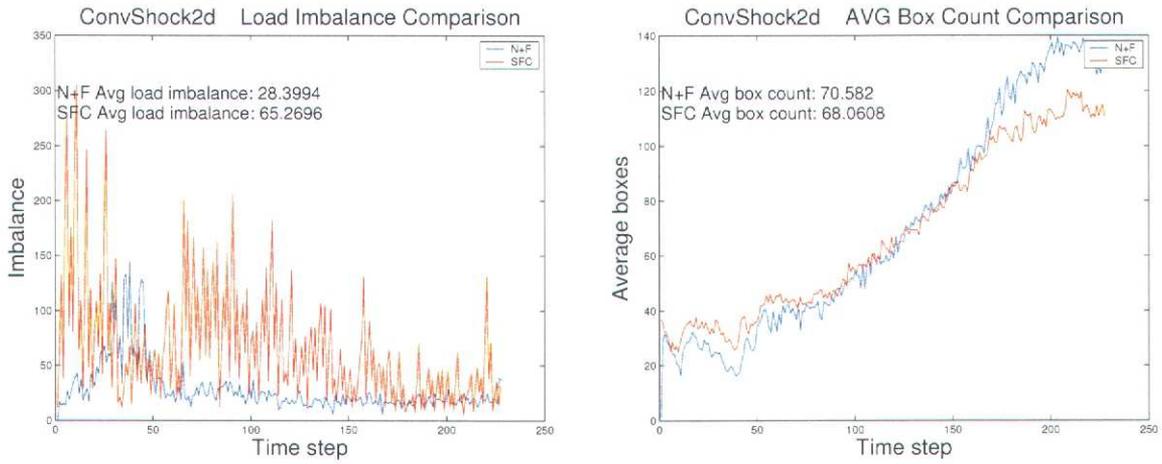


Figure 10. Nature+Fable with the improved bi-level algorithm versus AMROC DB/SFC for ConvShock2d and 16 processors: Load imbalance (Left) and box count (Right).

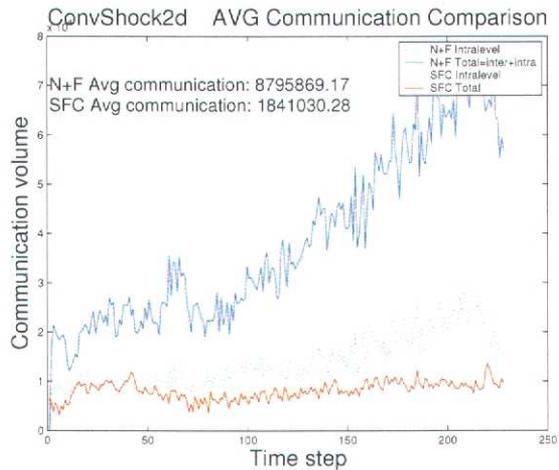


Figure 11. Nature+Fable with the improved bi-level algorithm versus AMROC DB/SFC for ConvShock2d and 16 processors: Communication volume.

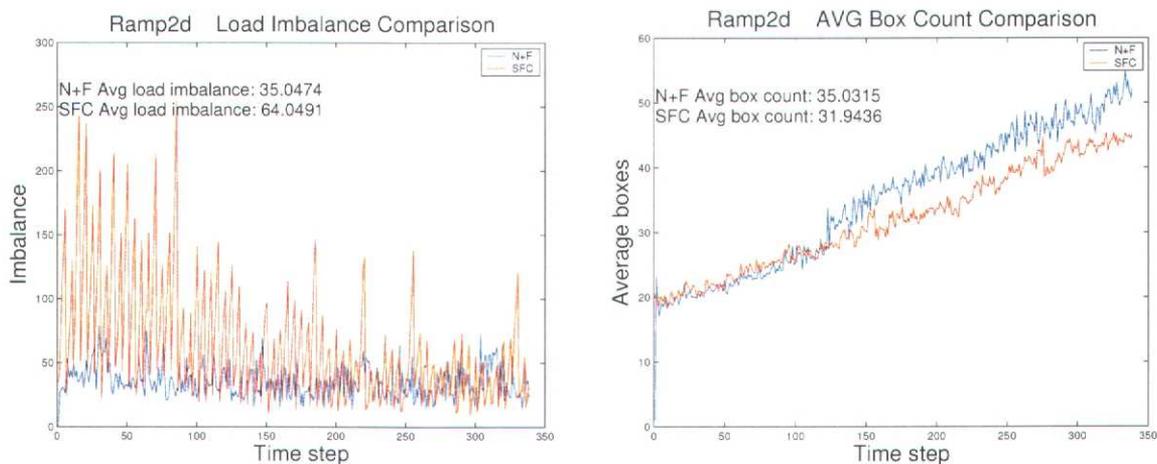


Figure 12. Nature+Fable with the improved bi-level algorithm versus AMROC DB/SFC for Ramp2d and 16 processors: Load imbalance (Left) and box count (Right).

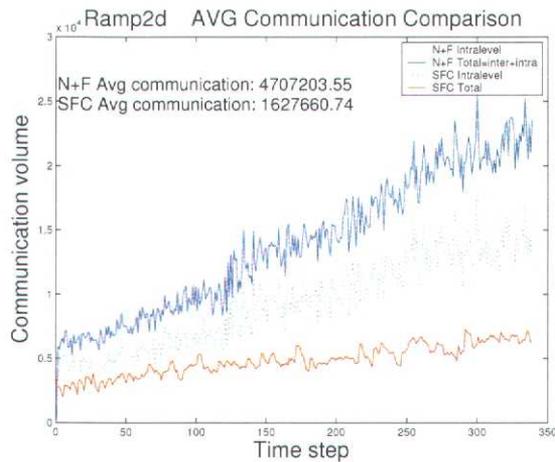


Figure 13. Nature+Fable with the improved bi-level algorithm versus AMROC DB/SFC for Ramp2d and 16 processors: Communication volume.

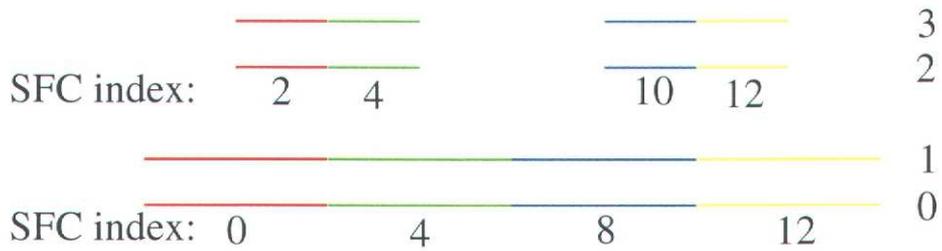


Figure 14. For this simple 1D hierarchy, Nature+Fable generated a perfect (DB) partitioning.

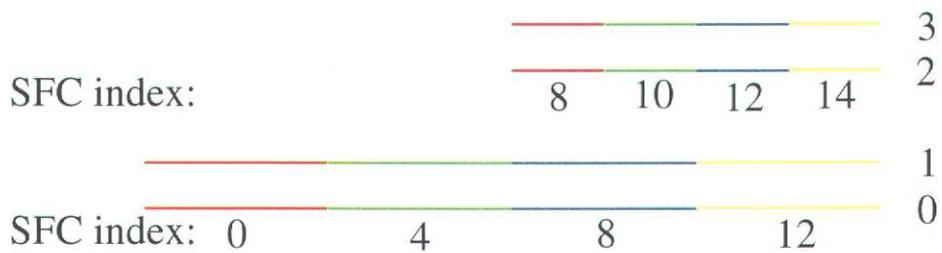


Figure 15. For this simple 1D hierarchy, a BD partitioning will probably generate too much load imbalance. Nature+Fable fails to map all blocks optimally (only the yellow processor avoids part of its inter-level communication).

DISTRIBUTION:

- | | | | |
|---|--|---|---|
| 4 | M9159
J. Ray and J. Steensland, 08964 | 1 | R. Deiterding
California Institute of Technology
MC 158-79, Pasadena, CA 91125,
USA |
| 2 | S. Chandra and M. Parashar
28 CoRE, Department of Electrical
& Computer Engineering
Rutgers, The State University of
New Jersey
94 Brett Road, Piscataway, NJ
08854-8058 | 2 | MS 9018
Central Technical Files, 8940-1 |
| 3 | M. Thúne, J. Rantakokko and H.
Johansson
Department of Information Tech-
nology
Uppsala University, Box 337, SE-
751 05 Uppsala, Sweden | 2 | MS 0899
Technical Library, 9616 |
| | | 1 | MS 9021
Classification Office, 8511, for
Technical Library, MS 0899, 9616
DOE/OSTI via URL |