PDPTA'99, Las Vegas, Nevada, USA. June 28 - July 1, 1999.

An Application-Centric Characterization of Distribution Techniques for Dynamic Adaptive Grid Hierarchies

Samip Bhavsar, Mausumi Shee, and Manish Parashar
Department of Electrical and Computer Engineering , Rutgers University, 94 Brett Road, Piscataway, NJ 08854
Tel: (732) 445-5388; Fax: (732) 445-0593; Email: {samip,mshee,parashar}@caip.rutgers.edu

Abstract

Dynamically adaptive methods for the solution of partial differential equations that employ locally optimal approximations can yield highly advantageous ratios for cost/accuracy. Distributed implementations of these methods offer the potential for accurate solution of physically realistic models of important physical systems. These implementations however, lead to interesting challenges in dynamic data-distribution and load balancing. This paper presents ongoing work on characterizing the performance of dynamic partitioning and load-balancing techniques for distributed adaptive grid hierarchies that underlie adaptive mesh-refinement algorithms (AMR). The overall goal of this characterization is to enable the selection of the most appropriate mechanism based on application and system parameters.

Keywords*: Dynamic load balancing; Performance characterization; Adaptive mesh refinement.*

1. Introduction

This paper presents ongoing work on the performance characterization of dynamic partitioning and load-balancing techniques for distributed adaptive grid hierarchies that underlie parallel adaptive mesh-refinement (AMR) techniques for the solution of partial-differential equations. The overall goal of this characterization is to enable the selection of the most appropriate mechanism and based on application and system parameters.

Dynamically adaptive methods for the solution of partial differential equations that employ locally optimal approximations can yield highly advantageous ratios for cost/accuracy when compared to methods based upon static uniform approximations. These techniques seek to improve the accuracy of the solution by dynamically refining the computational grid in regions of high local solution error. Distributed implementations of these methods offer the potential for accurate solution of physically realistic models of important physical systems. We believe that the next generation simulations of complex physical phenomenon will be built on top such dynamically adaptive techniques executing on dynamic and heterogeneous computational grids, and will provide dramatic insights into complex systems such as interacting black holes and neutron stars, formations of galaxies, oil reservoirs and aquifers, and seismic models of the whole earth.

Distributed implementations of adaptive applications lead to interesting challenges in dynamic resource allocation, data-distribution and load balancing, communications and coordination, and resource management. The overall efficiency of the algorithms is limited by the ability to partition the underlying data-structures at run-time so as to expose all inherent parallelism, minimize communication and synchronization overheads, and balance load. A critical requirement while partitioning 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 while the latter minimizes the total communication and synchronization overheads. Furthermore application adaptivity results in application grids being created, moved and deleted on the fly, making it is necessary to efficiently re-partition the hierarchy on the fly so that it continues to meet these goals.

This paper first defines an application-centric performance characterization of distribution mechanism for AMR grid hierarchies. It then uses it to characterize the performance of a suite partitioning and load-balancing mechanisms used by distributed AMR infrastructures.

2. Problem Description

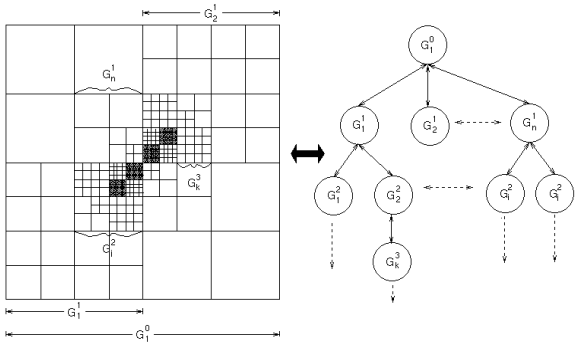

Figure 1 - Adaptive Grid Hierarchy - 2D (Berger-Oliger AMR Scheme)

Dynamically adaptive numerical techniques for solving differential equations provide a means for concentrating computational effort to appropriate regions in the computational domain. In the case of hierarchical adaptive mesh refinement (AMR) methods, this is achieved by tracking regions in the domain that require additional resolution and dynamically overlaying finer grids over these regions. AMR-based techniques start with a base coarse grid with minimum acceptable resolution that covers the entire computational domain. As the solution progresses, regions in the domain requiring additional resolution are tagged and finer grids are over laid on the tagged regions of the coarse grid. Refinement proceeds recursively so that regions on the finer grid requiring more resolution are similarly tagged and even finer grids are overlaid on these regions. The resulting grid structure is a dynamic adaptive grid hierarchy. The adaptive grid hierarchy corresponding to the AMR formulation by Marsha Berger and Joseph Oliger is shown in Figure 1.

Distribution of adaptive methods based on hierarchical AMR consists of appropriately partitioning the adaptive grid hierarchy across available computing nodes, and concurrently operating on the local portions of this domain. Parallel AMR applications require two primary types of communication: (a) Inter-grid Communications: Inter-grid Communications are defined between component grids at different levels of the grid-hierarchy and consist of prolongations (coarse to fine transfers) and restrictions (fine to coarse transfers). These communications typically require a gather/scatter type operations based on an interpolation or averaging stencil. Inter-grid communications can lead to serialization bottlenecks for naïve decompositions of the grid hierarchy. (b) Intra-grid Communications: Intra-grid Communications are required to update the grid-elements along the boundaries of local portions of a distributed grid. These communications consist of near-neighbor exchanges on the stencil defined by the difference operator. Intra-grid communications are regular and can be scheduled so as to overlap with computations on the interior region of the local portions of a distributed grid. Note that on the same processor, these communications translate to memory copies. Key requirements for a decomposition scheme used to partition the adaptive grid hierarchy across processors can be summarized as: (1) expose available data-parallelism (2) minimize communication overheads (3) balance overall load distribution and (4) enable dynamic load redistribution with minimum overheads. A balanced load distribution and efficient re-distribution is particularly critical for parallel AMR based applications as different levels of the hierarchy have different computational loads. The AMR scheme for time dependent applications have large number of grid elements, which are frequently updated, which makes efficient dynamic re-distribution difficult.

3. Parallel/Distributed AMR Infrastructures

There already exists wide spectrum of software systems that support parallel and distributed implementations of AMR applications. Five such infrastructures are introduced below. Each system represents a unique combination of design decisions in terms of algorithms, data-structures, decomposition, mapping and distribution mechanism, and communication mechanism. In this paper we characterize the partitioning and load-balancing schemes that underlie these infrastructures.

BATSRUS [1] is implemented in FORTRAN90, using a block-based domain-decomposition approach. Blocks of cell (stored as 3D F90 arrays) are locally stored on each processor so as to achieve a reasonable balanced load. The application starts out with a pool of processors, some of which are possibly unused. Every utilized processor has a block of equal *memory size*, but possibly at a different resolution and/or a different sized partition of physical space. As the application adapts and

new (adapted) grids are created, these are allocated, in units of the same fixed block size to the unused processors. No more refinement can occur once all the virtual processors are used up.

PARAMESH [7] is another FORTRAN 90 package designed to provide an application developer with an easy route to extend an existing serial code which uses a logically cartesian structured mesh into a parallel code with adaptive mesh refinement (AMR). The PARAMESH distribution strategy is based on partitioning a hierarchical tree representation of the adaptive grid structure.

SCOREC Parallel Mesh Databases (PMDB) [6] provides a generic mesh database for the topological, geometric and classification information that describes a finite element mesh. The database supports meshes of non-manifold models and multiple meshes on a single model or multiple models. Operators are provided to retrieve, store and modify the information stored in the database. PMDB provides three static partitioning procedures for initial mesh distribution, three dynamic load-balancing schemes and mesh migration operators.

SAMRAI [7] is an object-oriented framework that provides computational scientists with general and extensible software support for the prototyping and development of parallel structured adaptive mesh refinement applications. SAMRAI makes extensive use of object-oriented techniques and various design patterns, such as Abstract Factory, Strategy, and Chain of Responsibility.

DAGH [2] is an object-oriented toolkit for the development of parallel and distributed applications based on a family of adaptive mesh-refinement and multigrid techniques. DAGH is built on a "semantically specialized" distributed shared memory substrate that implements a hierarchical distributed dynamic array (HDDA) [5][4]. HDDA provides uniform array access to heterogeneous dynamic objects spanning distributed address spaces and multiple storage types. Communication, synchronization and consistency of HDDA objects are transparently managed for the user. Distribution of the HDDA is achieved by partitioning its array index space across the processors. The index-space is directly derived from the application domain, using locality preserving space-filling mapping.

4. A Characterization of AMR Distribution Mechanisms

We use four criteria to characterize distribution mechanism for AMR adaptive grid hierarchies, viz. load balance, distribution quality, grid interaction overheads (inter-processor communication and memory copy), and data-movement overheads. These criteria are described below.

4.1. Load Balance

The load balance metric measures a combination of the distribution of load across the processors, the time taken to achieve the distribution. AMR applications require re-distribution and load balancing at regular intervals; consequently the time spent in this effort is critical. The goal of this metric is to define operational points that represent the best balance between the effort spent in balancing the load and the balance achieved. In this paper we only address the quality of the load-balance and not the effort required.

4.2. Distribution Quality

Distribution quality is quantified by the number of grid components created on each processors and the quality (size, aspect ratio) of these components. The former captures the overheads due to the allocation, operation, and management and de-allocation of grid components. Large number of small grid increases the number of memory copies required for inter-level and intra-level communications. The size and shape of the grids also effects the communication/memory copy behavior. Bad aspect ratios result in larger interfaces between sibling grids and increased intra-level communications. Finally grid size also effects the overall cache behavior. Our goal is to use this metric to determine an acceptable range for the shape and size of grid components for different architectures, and use this to drive the distribution. In this paper we evaluate the number of boxes for each scheme studied.

4.3. Grid Interaction Overheads

The grid interaction overhead metric aims at characterizing the ability of the distribution scheme to capture and maintain application locality. Here we measure the overheads of four kinds of communications: inter-grid communications between grids at different levels, intra-Grid communication along ghost boundaries, and inter- and intra grid memory copies for co-located grid components.

Maintaining locality to minimize these overheads can lead to conflicting optimizations. The objective of this metric is to identify a balance between the two overheads based on system memory architecture and communication characteristics, that can achieve best overall performance.

4.4. Data Movement

Every refinement step in the AMR algorithms typically causes the adaptive grid hierarchy to change requiring redistribution. The redistribution should be incremental so as to minimize the data that has to be relocated. The objective of the data movement metric is to characterize the ability of the distribution scheme to minimize redistribution costs by reassigning grids to their original location. Optimizing this metric can lead to conflicts with requirements for optimizing load balancing and interaction overheads.

5. Run-Time Partitioning Dynamic AMR Grid Hierarchies

The section describes the six dynamic partitioning and load balancing schemes that we have implemented and evaluated in this paper. These schemes encapsulate key ideas underlying the approaches used by the AMR infrastructures described in Section 3.

5.1. Space-Filling Curves

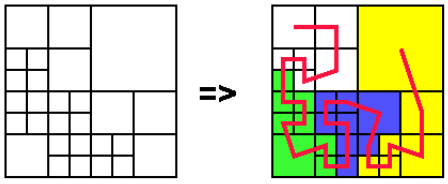

Figure 2 - Space-Filling Curve Representation of an Adaptive Grid Hierarchy

Space-filling curves (SFC) [3] are a class of locality preserving mappings from d-dimensional space to 1-dimensional space i.e., $N^d \rightarrow N^1$, such that each point in N^d is mapped to a unique point or index in N^1. The self-similar or recursive nature of these mappings can be exploited to represent a hierarchical structure and to maintain locality across different levels of hierarchy. The SFC representation of the adaptive grid hierarchy is a 1-D ordered list of composite grid blocks where each composite block represents a block of the entire grid hierarchy and may contain more than one grid level; i.e. inter-level locality is maintained within each composite block. Figure 2 illustrates the composite representation for a two dimensional grid hierarchy. Using the space-filling curve representation, the adaptive grid hierarchy can be simply partitioned by partitioning the composite list to balance the total work assigned to each processor. This decomposition using the Peano-Hilbert space-filling ordering for a 1-D grid hierarchy is shown in Figure 3. As inter-level locality is inherently maintained by the composite representation, the decomposition generated by partitioning this representation eliminates expensive gather/scatter communication and allows prolongation and restriction operations to be performed locally at each processor.

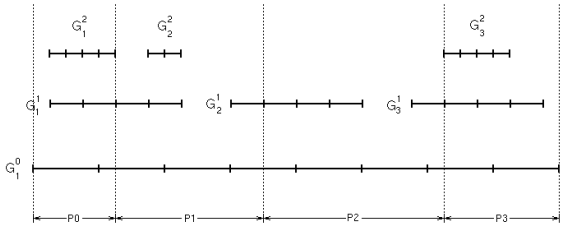

Figure 3 - Space-Filling (Composite) Distribution

5.2. Independent Grid Distribution

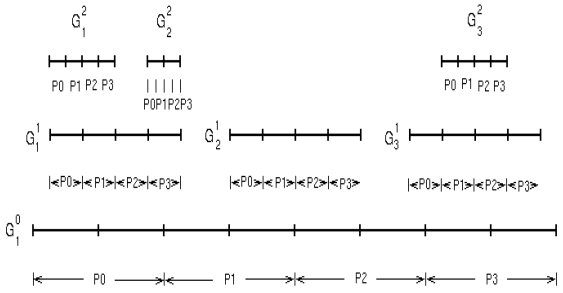

Figure 4 - Independent Grid Distribution

The independent grid distribution (IGD) scheme, shown in Figure 4, distributes the grids independently across the processors. This distribution leads to balanced loads and no redistribution is required when grids are created or deleted. However the decomposition scheme can be very inefficient with regard to inter-grid communication. In the adaptive grid hierarchy, a fine grid typically corresponds to a small region of the underlying coarse grid. If both, the fine and coarse grid are distributed over the entire set of processors, all the processors will

communicate with the small set of processors corresponding to the associated coarse grid region, thereby causing a serialization bottleneck. For example, a restriction from grid G22 to grid G11 requires all the processors to communicate with processor P3.

5.3. Combined Grid Distribution

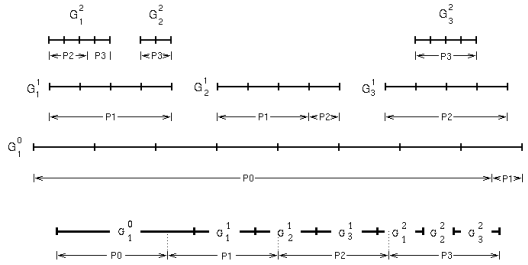

Figure 5 - Combined Grid Distribution

The combined grid distribution (CGD), shown in Figure 5, distributes the total work load in the grid hierarchy by first forming a simple linear structure by abutting grids at a level and then decomposing this structure into partitions of equal load. The combined decomposition scheme also suffers from the serialization bottleneck described above but to a lesser extent. For example, in Figure, G21 and G22 update G11 requiring P2 and P3 to communicate with P1 for every restriction. Regriding operations involving the creation or deletion of a grid are extremely expensive in this case, as they require an almost complete redistribution of the grid hierarchy. The combined grid decomposition does not exploit the parallelism available within a level of the hierarchy. For example, when G01 is being updated, processors P2 and P3 are idle and P1 has only a small amount of work. Similarly when updating grids at level 1 (G11, G12 and G13) processors P0 and P3 are idle, and when updating grids at level 2 (G21, G22 and G23) processors P0 and P1 are idle.

5.4. Independent Level Distribution

In the independent level distribution (ILD) scheme (see Figure 6), each level of the adaptive grid hierarchy is individually distributed by partitioning the combined load of all component grids at the level is distributed among the processors. This scheme overcomes some of the drawbacks of the independent grid distribution. Parallelism within a level of the hierarchy is exploited. Although the inter-grid communication bottleneck is reduced in this case, the required gather/scatter communications can be expensive. Creation or deletion of component grids at any level requires a re-distribution of the entire level.

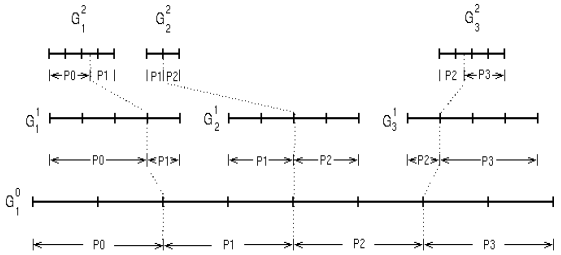

Figure 6 - Independent Level Distribution

5.5. Iterative Tree balancing

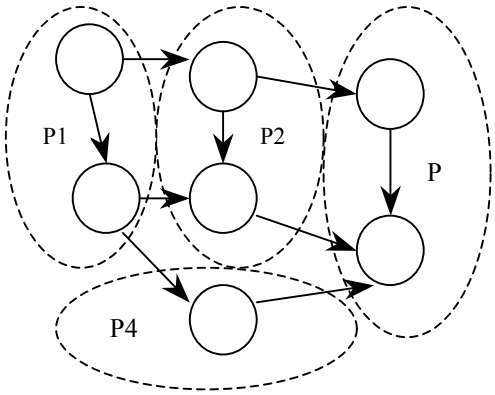

Figure 7 - Iterative Tree Balancing

The iterative tree balancing (ITB) scheme (see Figure 7) treats the dynamic partitioning and load-balancing problem as a graph-partitioning problem. A table is created from the grids at each timestep, which keeps pointers to neighboring and parent grids. A breadth first search is made on this graph i.e. for every grid immediate neighbors and children are also considered along with load distribution. Thus load balancing, inter level communication and intra level communication are addressed together. This scheme is promising from the point of view that all the constraints are dealt with to some extent.

5.6. Weighted Distribution

The weighted distribution scheme is a heuristic based hybrid scheme that attempts to combine the features of the other schemes described in this section. As previously observed, there are three primary parameters that

need to be controlled to minimize the overheads of an adaptive grid hierarchy distribution, viz. intra-level communication, inter-level communication and data movement at each regrid. In the weighted distribution we first assign a weight to each of these overheads. This weight defines the significance and contribution of the overhead to the overall application performance and depends on the system architecture and dynamic application behavior. The next step uses these weights to compute the affinity of each component grid to the different processors. Initially grids have no affinity for any processor. For each grid, the affinity of the processor(s) housing its parents is now increased by the inter-level communication weight. Similarly the affinities of processors housing the neighbors of the grid are increased by the intra-level communication weight, and affinity of the original location of the grid is increased by the data-movement overhead weight. The grid is now assigned to the available processors (i.e. total assigned load is below threshold for load balancing) to which the grid has maximum affinity. If the grid has equal affinity to more than one processor, the grid is either split among the processor (if its size is greater than the size threshold) or assigned to the processor with least load. Weights assigned to the different parameters can change dynamically depending on the current application and system states. For example if the application has many component grids and uses a large stencil, then the dominating weight is associated with intra-level communication. Similarly if the application is very dynamic and needs to regrid very often, the data-movement weight dominates.

6. Experimental Evaluation

To evaluate the six distribution and load-balancing schemes outlined in Section 5, we use the trace of grid adaptations from a 3D AMR application. The chosen application is the simulation of the Buckley-Leverette equations used in oil reservoir simulations. The trace was generated from a single processor run with 5 levels of factor 2 refinement and consisted of 100 regrids steps. The trace was then fed to a partitioning module that partitioned the boxes across the required number of processor using each of the six schemes. An AMR simulator then evaluated various costs for the partition generated. The plots presented in this section represent cumulative costs for each scheme. The vertical axis in each of these plots represents the relevant metric. In case of communication and data-movement overheads, this corresponds to the amount of information communicated. In case of distribution quality metrics this corresponds to the number of grids per processor and the percentage load imbalance (a perfect load balance corresponds to 0%).

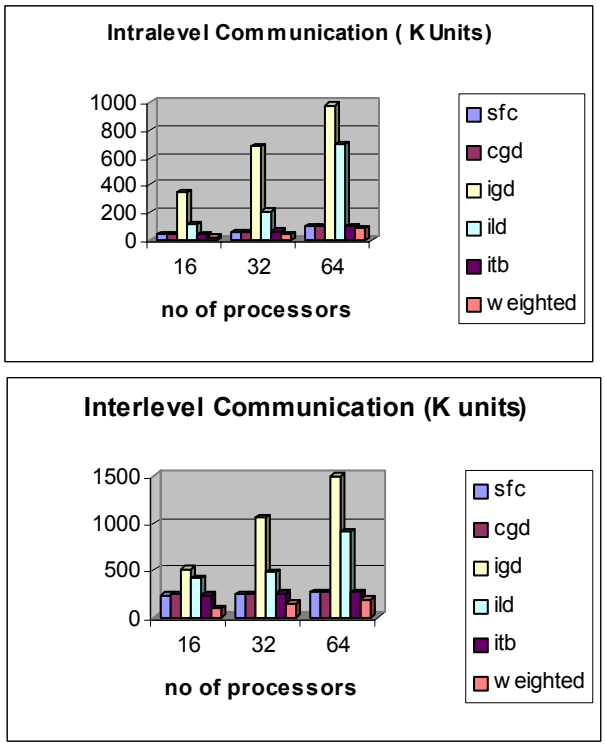

Figure 8 - Communication Overheads

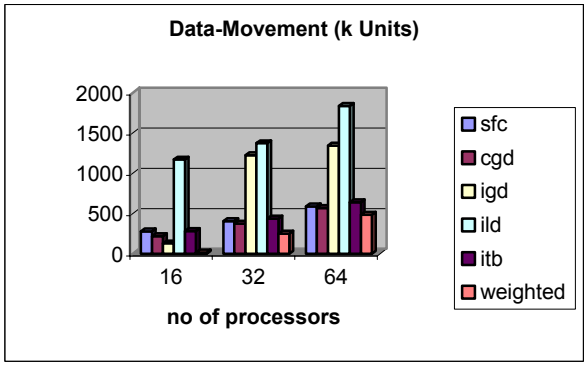

Figure 9 - Data-Movement Overheads

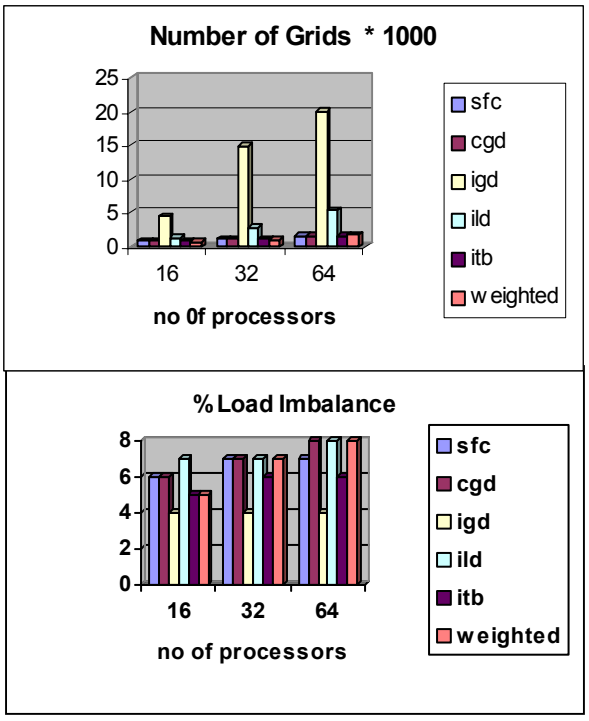

Figure 10 - Distribution Quality

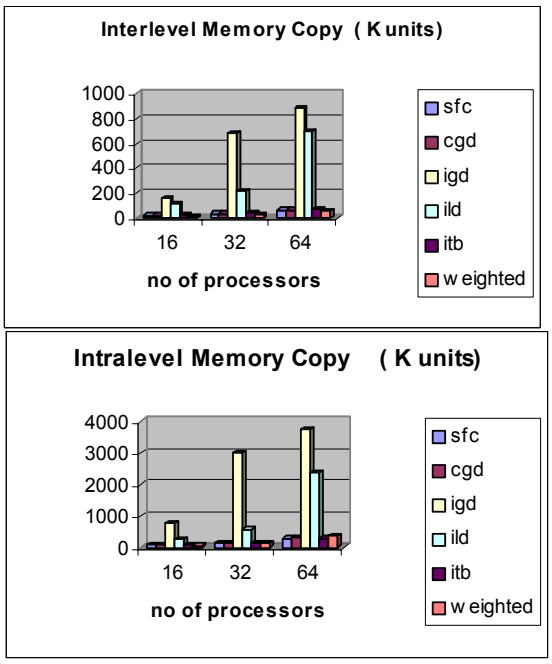

Figure 11 - Memory Copy Overheads

7. Conclusions and Future Work

In this paper we presented a performance characterization of six dynamic partitioning and load-balancing techniques for distributed adaptive grid hierarchies that underlie parallel adaptive mesh-refinement (AMR) techniques for the solution of partial-differential equations. This is part of an ongoing project for developing policy driven "smart" tools for automated distribution/load balancing of there problems in heterogeneous distributed environments. The characterization consisted of 3 metrics: Interaction overheads (inter- and intra- level communications and copies), Distribution Quality (load-balance, number of grids) and Data Movement. The presented results show that space-filling curve, iterative tree balancing, and weighted distribution clearly superior for all metrics. The reason is that these techniques use application information to determine the partitioning rather than pure heuristic. The weighted scheme in particular is tuned to this class of applications and tends to do better than the other schemes. Current work look at completing this characterization and encoding the results into a policy rule base that can drive an automated partitioning and load-balancing tool.

8. Refernces

[1] BATSRUS: hpcc.engin.umich.edu/HPCC/codes\\/2/BATSRUSv2.html.

[2] Distributed Adaptive Grid Hierarchies, www.caip.rutgers.edu/~parashar/DAGH/.

[3] Hanan Samet, The Design and Analysis of Spatial Data Structures, Addison - Wesley Publishing Company, 1989.

[4] M. Parashar and J.C. Browne, ``On Partitioning Dynamic Adaptive Grid Hierarchies", Proceedings of the 29th Annual Hawaii International Conference on System Sciences, Jan. 1996.

[5] M. Parashar and J.C. Browne, `Distributed Dynamic Data Structures for Parallel Adaptive Mesh Refinement', Proceedings of the International Conference for High Performance Computing, Dec. 1995.

[6] SCOREC Parallel Scientific Computation: www.scorec.rpi.edu/programs/parallel/ParallelScientific.html

[7] PARAMESH: sdcd.gsfc.nasa.gov/ESS/eazydir/\\inhouse/macneice/paramesh/paramesh.html.

[8] SAMRAI: Structured Adaptive Mesh Refinement Applications Infrastructure, www.llnl.gov/CASC/SAMRAI/.