Using Clustering to Address Heterogeneity and Dynamism in Parallel Scientific Applications[⋆]

Xiaolin Li[1] and Manish Parashar[2]

[1] Department of Computer Science, Oklahoma State University, OK 74078, USA
[2] Department of Electrical & Computer Engineering, Rutgers University, NJ 08854, USA
Email: xiaolin@cs.okstate.edu and parashar@caip.rutgers.edu

Abstract. The dynamism and space-time heterogeneity exhibited by structured adaptive mesh refinement (SAMR) applications makes their scalable parallel implementation a significant challenge. This paper investigates an adaptive hierarchical multi-partitioner (AHMP) framework that dynamically applies multiple partitioners to different regions of the domain, in a hierarchical manner, to match the local requirements of these regions. Key components of the AHMP framework include a segmentation-based clustering algorithm (SBC) for identifying regions in the domain with relatively homogeneous partitioning requirements, mechanisms for characterizing the partitioning requirements, and a runtime system for selecting, configuring and applying the most appropriate partitioner to each region. The AHMP framework has been implemented and experimentally evaluated on up to 1280 processors of the IBM SP4 cluster at San Diego Supercomputer Center.

Keywords: Parallel Computing, Adaptive Mesh Refinement, Dynamic Load Balancing, Hierarchical Multi-Partitioner

1 Introduction

Simulations of complex physical phenomena, modeled by systems of partial differential equations (PDE), are playing an increasingly important role in science and engineering. Dynamic structured adaptive mesh refinement (SAMR) techniques [1] are emerging as attractive formulations of these simulations. Compared to numerical techniques based on static uniform discretization, SAMR can yield highly advantageous ratios for cost/accuracy by adaptively concentrating computational effort to appropriate regions at runtime.

Parallel implementations of SAMR-based applications have the potential to accurately model complex physical phenomena and provide dramatic insights. However, while there have been some large-scale implementations [4] [6] [7] [8] [11], these implementations are typically based on application-specific customizations and general scalable implementations of SAMR applications continue to present significant challenges. This is primarily due to the dynamism and space-time heterogeneity exhibited

[⋆] The research presented in this paper is supported in part by the National Science Foundation via grants numbers ACI 9984357, EIA 0103674, EIA 0120934, ANI 0335244, CNS 0305495, CNS 0426354 and IIS 0430826.

by these applications. SAMR dynamism/heterogeneity has been traditionally addressed using dynamic partitioning and load-balancing algorithms, such as the mechanisms presented in [6] [11], that partition and load-balance the domain when it changes. The meta-partitioner approach proposed in [14] selects and configures partitioners at runtime to match the application's current requirements. However, due to the spatial heterogeneity of the SAMR domain, the computation/communication requirements can vary significantly across the domain, and as a result, using single partitioner for the entire domain can lead to decompositions that are locally inefficient. This is especially true for large-scale simulations that run on over 1000 processors.

The objective of the research presented in this paper is to address this issue. Specifically, we investigate an adaptive multi-partitioner framework that dynamically applies multiple partitioners to different regions of the domain, in a hierarchical manner, to match the local requirements of the regions. This research builds on our earlier research on meta-partitioning [14] and adaptive hierarchical partitioning [10] to define an adaptive hierarchical multi-partitioner framework (AHMP). The experimental evaluation of AHMP demonstrates the performance gains using AHMP on up to 1280 processors of the IBM SP4 cluster at San Diego Supercomputer Center.

The rest of the paper is organized as follows. Section 2 presents the problem description. Section 3 presents the AHMP framework and the SBC clustering algorithm. The experimental evaluation is presented in Section 4 . Section 5 reviews related work. Section 6 presents a conclusion.

2 Problem Description

SAMR formulations for adaptive solutions to PDE systems track regions in the computational domain with high solution errors that require additional resolution. SAMR methods start with a base coarse grid with minimum acceptable resolution. As the solution progresses, regions in the domain requiring additional resolution are tagged and finer grids are overlaid on these tagged regions of the coarse grid. Refinement proceeds recursively so that regions on the finer grid requiring more resolution are similarly tagged and refined. It results in a dynamic adaptive grid hierarchy [11].

Parallel implementations of SAMR typically partition the adaptive grid hierarchy across available processors, and each processor operates on its local portions of this domain in parallel. The overall performance of parallel SAMR applications is thus 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. Communication overheads of parallel SAMR applications primarily consist of four components: (1) *Inter-level communications*, defined between component grids at different levels of the grid hierarchy; (2) *Intra-level communications*, required to update the grid-elements along the boundaries of local portions of a distributed grid; (3) *Synchronization cost*, which occurs when the load is not balanced; (4) *Data migration cost*, which occurs between successive regridding and re-mapping steps.

The space-time heterogeneity of SAMR applications is illustrated in Figure 1 using the 3-D compressible turbulence simulation kernel solving the Richtmyer-Meshkov (RM3D) instability [3]. The figure shows a selection of snapshots of the RM3D adap-

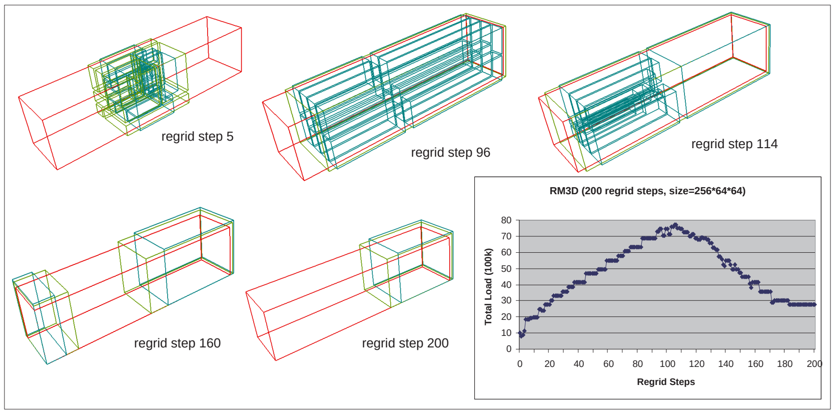

Fig. 1. Spatial and Temporal Heterogeneity and Workload Dynamics for RM3D Simulation

tive grid hierarchy as well as a plot of its load dynamics at different regrid steps. Since the grid hierarchy remains unchanged between two regrid steps, the workload dynamics and other features are measured in terms of regrid steps. The workload in this figure represents the computational/storage requirement, which is computed based on the number of grid points in the grid hierarchy. Application variables are typically defined at these grid points and are updated at each iteration of the simulation, and consequently, the computational/storage requirements are proportional to the number of grid points. The snapshots in this figure clearly demonstrate the dynamics and spatial and temporal heterogeneity of SAMR applications - different subregions in the computational domain have different computational and communication requirements and regions of refinement are created, deleted, relocated, and grow/shrink at runtime.

3 Adaptive Hierarchical Multi-Partitioner (AHMP) Framework

The operation of the AHMP framework is illustrated in Figure 2. The input of AHMP is the structure of the current grid hierarchy, which is represented as a list of regions and defines the runtime state of the SAMR application. AHMP operation consists of the following steps. First, a clustering algorithm is used to identify clique hierarchies. Second, each clique is characterized and its partitioning requirements identified. Available resources are also partitioned into corresponding resource groups based on the relative requirements of the cliques. Third, these requirements are used to select and configure an appropriate partitioner for each clique. The partitioner is selected from a partitioner repository using selection policies. Finally, each clique is partitioned and mapped to processors in its corresponding resource group. The strategy is triggered locally when the application state changes. State changes are determined using a load-imbalance metric defined below. Partitioning proceeds hierarchically and incrementally. The identification and isolation of cliques uses a segmentation-based clustering (SBC) scheme. Partitioning schemes in the partitioner repository include Greedy Partitioning Algorithm (GPA), Level-based Partitioning Algorithm (LPA), bin-packing partitioning algorithm (BPA), geometric multilevel + sequence partitioning (G-MISP+SP),

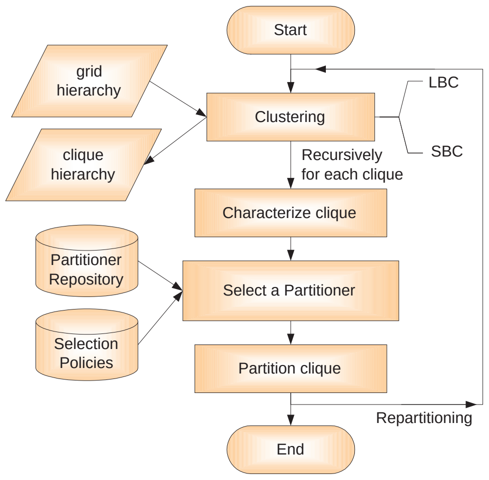

Fig. 2. Flowchart for the AHMP Framework

and p-way binary dissection algorithm (pBD-ISP) [10] [14]. AHMP extends our previous work on Hierarchical Partitioning Algorithm (HPA) [10], which applies single partitioner hierarchically, reducing global communication overheads and enabling incremental repartitioning and rescheduling.

The load imbalance factor (LIF) metric is used as the criterion for triggering repartitioning and rescheduling within a local resource group, and is defined as follows:

$$LIF_A = \frac{\max_{i=1}^{A_n} T_i - \min_{i=1}^{A_n} T_i}{\sum_{i=1}^{A_n} T_i / A_n}$$

where A_n is the total number of processors in resource group A, and T_i is the estimated relative execution time between two consecutive regrid steps for processor i, which is proportional to its load. The local load imbalance threshold is γ_A. When $LIF_A > \gamma_A$, the repartitioning is triggered inside the local group. Note that the imbalance factor can be recursively calculated for larger groups as well.

3.1 Clustering Algorithm for Clique Identification

The objective of clustering is to identify well-structured subregions in the SAMR grid hierarchy, called cliques. A clique is a quasi-homogeneous computational sub-domain with relatively homogeneous partitioning requirements.

This section presents the segmentation-based clustering (SBC) algorithm, which is based on space-filling curves (SFC) [12]. The algorithm is motivated by the locality-preserving property of SFCs and the localized nature of physical features in SAMR applications. Typical SAMR applications exhibit localized features and result in localized refinements. Moveover, refinement levels and the resulting adaptive grid hierarchy reflect the application runtime state. SBC hence attempts to cluster subregions with similar refinement levels. SBC defines the load density factor (LDF) as follows:

$$LDF(rlev) = \frac{\text{associated load on the subdomain}}{\text{volume of the subdomain at } rlev}$$

where $rlev$ denotes the refinement level and the volume is for the subregion of interest.

The SBC algorithm first smooths out subregions that are smaller than a predefined threshold, which is referred to as the template size (TS). TS is determined by the stencil size of the finite difference method and the granularity constraint that defines a certain computation communication ratio. SBC then follows the SFC index to extract subregions (defined by rectangular bounding boxes) from the subregion list until the size of the accumulated subregion set is over the template size. It calculates the load density for this set of subregions and computes a histogram of its load density. SBC continues to scan through the entire subregion list, and repeats the above process, calculating the load density and computing histograms. Based on the histogram of the load density obtained, it then finds a clustering threshold θ. A simple intermeans thresholding algorithm [5] is used. Using the threshold obtained, subregions are further partitioned into several clique regions. A hierarchical structure of clique regions is created by recursively calling the SBC algorithm for finer refinements.

Note that this algorithm has similarities to the point clustering algorithms proposed by Berger and Regoutsos in [2]. However, the SBC scheme differs from this scheme in two aspects. Unlike the Berger-Regoutsos scheme, which creates fine grained cluster, the SBC scheme targets coarser granularity cliques. In addition, SBC also takes advantage of the locality-preserving properties of SFCs to potentially reduce data movement costs between consecutive repartitioning phases.

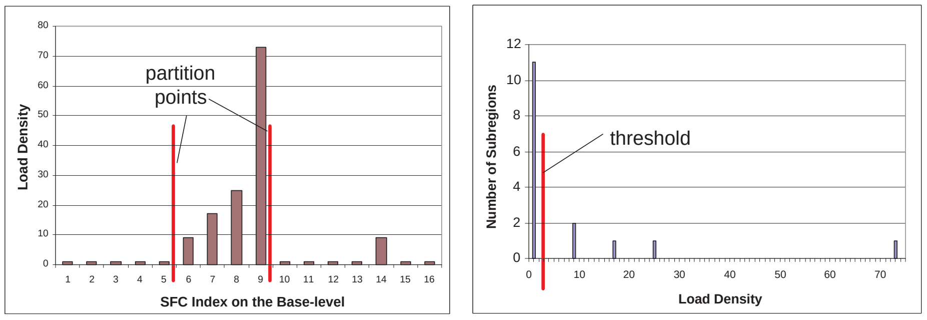

Fig. 3. Load Density Distribution and Histogram

Figure 3 shows the load density distribution and histogram for an SFC-indexed subdomain list. For this example, the SBC algorithm creates three cliques defined by the regions separated by the vertical lines in the figure on the left. The template size in this example is two boxes on the base level. The right figure shows a histogram of the load density. For this example, the threshold is identified in between 1 and 9 using the intermeans thresholding algorithm. While there are many more sophisticated approaches for identifying good thresholds for segmentation and edge detection in image processing [5], this approach is sufficient for our purpose. Note that we assume a predefined minimum size for a clique region. In this example, the subregion indexed 14 does not form a clique as its size is less then the template size. It is clustered with another subregion in its proximity.

3.2 Clique Characterization and Partitioner Selection

The characterization of a clique is based on its computation and communication requirements, and its refinement homogeneity is defined in Section 4.1. Using the characterization of applications and partitioners presented in [14], partitioner-selection policies are defined to select the partitioners. The overall goal of these policies is to obtain better load balance for less refined cliques, and to reduce communication and synchronization costs for highly refined cliques. For example, the policy dictates that the GPA and G-MISP+SP partitioning algorithms be used for cliques with refinement homogeneity below some threshold and partitioning algorithms LPA and pBD-ISP be used for cliques with refinement homogeneity greater than the threshold.

4 Experimental Evaluation

4.1 Evaluating the Effectiveness of the SBC Clustering Algorithm

To aid the evaluation of the effectiveness of the SBC clustering scheme, a clustering quality metric is defined below. The static quality of a clique is measured in terms of its refinement homogeneity and the efficiency of the clustering algorithm. The dynamic quality of the clique hierarchy is measured in terms of its communication costs (intra-level, inter-level, and data migration).

(1) **Refinement Homogeneity**: This measures the quality of the structure of a clique. Let $|R_i^{total}(l)|$ denote the total size of a subregion or a clique at refinement level l, which is composed of $R_i^{ref}(l)$, the size of refined regions, and $R_i^{unref}(l)$, the size of un-refined regions at refinement level l. Refinement homogeneity is recursively defined between two refinement levels as follows:

$$H_i(l) = \frac{|R_i^{ref}(l)|}{|R_i^{total}(l)|}, \;\; and \;\; H_{all}(l) = \frac{1}{n}\sum_{i=1}^{n} H_i(l), \text{if } |R_i^{ref}(l)| \neq 0$$

where n is the total number of subregions that have refinement level $l+1$. The goal of AHMP is to maximize the refinement homogeneity of a clique as partitioners work well on relatively homogeneous regions.

(2) **Communication Cost**: This measures the communication overheads of a clique and includes inter-level communication, intra-level communication, synchronization cost, and data migration cost as described in Section 2. The goal of AHMP is to minimize the communication overheads of a clique.

(3) **Clustering Cost**: This measures the efficiency of the clustering algorithm itself. As mentioned above, SAMR applications require regular re-partitioning and re-balancing, and as a result clustering cost become important. The goal of AHMP is to minimize the clustering cost.

Partitioning algorithms typically work well on highly homogeneous grid structures. Hence, it is important to have a quantitative measure to specify homogeneity. Intuitively, the refinement homogeneity metric attempts to isolate refined cliques that are potentially heterogeneous. In contrast, unrefined cliques are homogeneous at their finest refinement level.

The effectiveness of SBC-based clustering is evaluated using the metrics defined above. The evaluation compares the refinement homogeneity of 6 SAMR application kernels with and without clustering. These application kernels span multiple domains, including computational fluid dynamics (compressible turbulence: RM2D and RM3D, supersonic flows: ENO2D), oil reservoir simulations (oil-water flow: BL2D and BL3D), and the transport equation (TP2D). The detailed descriptions and characterizations of these applications are presented in [14].

Table 1. Average Refinement Homogeneity $H(l)$ for 6 SAMR Applications

Application	Level0	Level1	Level2	Level3
TP2D	0.067	0.498	0.598	0.6680
RM2D	0.220	0.680	0.830	0.901
RM3D	0.427	0.618		
ENO2D	0.137	0.597	0.649	0.761
BL3D	0.044	0.267		
BL2D	0.020	0.438	0.406	0.316

The average refinement homogeneity of 6 SAMR applications without clustering is presented in Table 1. The table shows that the refinement homogeneity $H(l)$ increases as the refinement level l increases. Typical ranges of $H(l)$ are: $H(0) \in [0.02, 0.22]$, $H(1) \in [0.26, 0.68]$, $H(2) \in [0.59, 0.83]$ and $H(3) \in [0.66, 0.9]$. Several outliers require some explanation. In case of the BL2D application, average $H(2) = 0.4$. However, the individual values of $H(2)$ are in the range $[0.6, 0.9]$ with many scattered zeros. Since the refinement homogeneity on level 3 and above is typically over 0.6 and refined subregions on deeper refinement levels tend to be more scattered, the clustering schemes will focus efforts on clustering level 0, 1 and 2. Furthermore, based on these statistics, we set the threshold θ for switching between different lower-level partitioners as follows: $\theta_0 = 0.4$, $\theta_1 = 0.6$, and $\theta_2 = 0.8$, where the subscripts denote the refinement level.

The effects of clustering using SBC for the 6 SAMR applications are presented in Table 2. In the table, the gain is defined as the ratio of the improved homogeneity over the original homogeneity at each level. The gains for RM3D and RM2D applications are smaller because these applications already exhibit high refinement homogeneity. These results demonstrate the effectiveness of the clustering scheme.

4.2 Performance Evaluation

This section presents an evaluation of the AHMP scheme using the clustering quality metrics defined above.

Table 2. Homogeneity Improvements using SBC

Application	Level0	Level1	Gain on Level0	Gain on Level1
TP2D	0.565	0.989	8.433	1.986
RM2D	0.671	0.996	3.050	1.465
RM3D	0.802	0.980	1.878	1.586
ENO2D	0.851	0.995	6.212	1.667
BL3D	0.450	0.583	10.227	2.184
BL2D	0.563	0.794	28.150	1.813

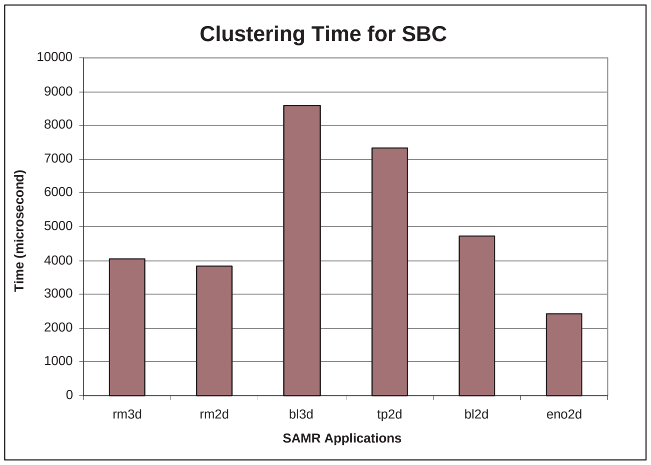

Fig. 4. Clustering Costs for the 6 SAMR Application Kernels

Clustering Costs: The cost of the SBC clustering algorithm is experimentally evaluated using the 6 different SAMR application kernels on a Beowulf cluster (Frea) at Rutgers University. The cluster consists of 64 processors and each processor has a 1.7 GHz Pentium IV CPU. The costs are plotted in Figure 4. As seen in this figure, the overall clustering time on average is less than 0.01 second. Note that the computational time between successive repartitioning phases is typically in the order of 10's of seconds, and as a result, the clustering costs are not significant.

Overall Performance: The overall performance benefit of the AHMP scheme is evaluated on DataStar, the IBM SP4 cluster at San Diego Supercomputer Center. DataStar has 176 (8-way) P655+ nodes (SP4). Each node has 8 (1.5 GHz) processors, 16 GB memory, and CPU peak performance is 6.0 GFlops. The evaluation uses the RM3D application kernel with a base grid of size 256x64x64, up to 3 refinement levels, and 1000 base level time steps. The number of processors used was between 64 and 1280.

The overall execution time is plotted in Figure 5. The figure plots execution times for GPA, LPA and AHMP. The plot shows that SBC+AHMP delivers the best performance. Compared to GPA, the performance improvement is between 30% to 42%. These improvements can be attributed to the following factors: (1) AHMP takes advantage of the strength of different partitioning schemes matching them to the requirements of each clique; (2) the SBC scheme creates well-structured cliques, which reduces the commu-

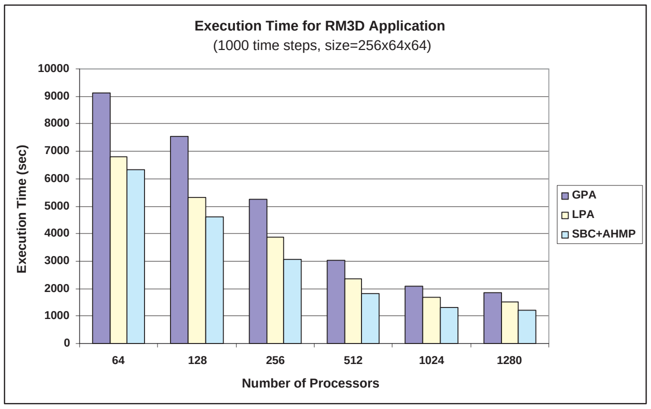

Fig. 5. Overall Performance for RM3D

nication between cliques; (3) AHMP enables incremental repartitioning/redistribution and concurrent communication between resource groups, which extends the advantages of HPA [10].

5 Related Work

Traditional parallel SAMR implementations presented in [6] [11] use dynamic partitioning and load-balancing algorithms. These approaches view the system as a flat pool of processors. They are based on global knowledge of the state of the adaptive grid hierarchy, and partition the grid hierarchy across the set of processors. Global synchronization and communication is required to maintain this global knowledge and can lead to significant overheads on large systems. Furthermore, these approaches do not exploit the hierarchical nature of the grid structure and the distribution of communications and synchronization in this structure. Dynamic load balancing schemes for distributed SAMR applications are proposed in [9], which consist of two phases: global load balancing and local load balancing. However, simplistic partitioning schemes are used without explicitly addressing the spatial and temporal heterogeneity exhibited by SAMR applications. The characterization of SAMR applications presented in [14] was based on the entire physical domain. The research in this paper goes one step further by considering the characteristics of individual subregions. The concept of natural regions was proposed in [13]. Two kinds of natural regions were defined: unrefined/homogeneous and refined/complex. The framework proposed in the paper then used a bi-level domain-based (BLED) partitioning scheme to partition the refined subregions. This approach is one of the first attempts to apply multiple partitioners concurrently to the SAMR domain. However, this approach restricts itself to applying only two partitioning schemes, one to the refined region and the other to the unrefined region.

6 Conclusion

This paper presented the adaptive hierarchical multi-partitioner (AHMP) scheme to address space-time heterogeneity in dynamic SAMR applications. The AHMP scheme applies multiple partitioners to different regions of the domain, in a hierarchical manner, to match the local requirements of the regions. A segmentation-based clustering algorithm (SBC) was used to identify clique regions with relatively homogeneous partitioning requirements in the adaptive computational domain. The partitioning requirements of these clique regions are then characterized, and the most appropriate partitioner for each clique is selected. The AHMP framework and its components have been implemented and experimentally evaluated using 6 SAMR application kernels. The evaluations demonstrated both, the effectiveness of the clustering as well as the performance improvements using AHMP.

References

1. M. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of Computational Physics*, 53:484–512, 1984.
2. M. Berger and I. Regoutsos. An algorithm for point clustering and grid generation. *IEEE Transactions on Systems, Man and Cybernetics*, 21(5):1278–1286, 1991.
3. J. Cummings, M. Aivazis, R. Samtaney, R. Radovitzky, S. Mauch, and D. Meiron. A virtual test facility for the simulation of dynamic response in materials. *Journal of Supercomputing*, 23:39–50, 2002.
4. K. Devine, E. Boman, R. Heaphy, B. Hendrickson, and C. Vaughan. Zoltan data management services for parallel dynamic applications. *Computing in Science and Engineering*, 4(2):90–97, 2002.
5. R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Prentice Hall, Upper Saddle River, NJ, 2nd edition, 2002.
6. R. D. Hornung and S. R. Kohn. Managing application complexity in the samrai object-oriented framework. *Concurrency and Computation - Practice & Experience*, 14(5):347–368, 2002.
7. L. V. Kale. Charm. URL: `http://charm.cs.uiuc.edu/research/charm/`.
8. G. Karypis and V. Kumar. Parmetis, 2003. URL: `http://www-users.cs.umn.edu/~karypis/metis/parmetis/index.html`.
9. Z. Lan, V. Taylor, and G. Bryan. Dynamic load balancing of samr applications on distributed systems. *Journal of Scientic Programming*, 10:4:319–328, 2002.
10. X. Li and M. Parashar. Dynamic load partitioning strategies for managing data of space and time heterogeneity in parallel samr applications. In *The 9th International Euro-Par Conference (Euro-Par 2003), Lecture Notes in Computer Science*, volume 2790, pages 181–188. Springer-Verlag, 2003.
11. M. Parashar and J. Browne. On partitioning dynamic adaptive grid hierarchies. In *29th Annual Hawaii Int. Conference on System Sciences*, pages 604–613, 1996.
12. H. Sagan. *Space Filling Curves*. Springer-Verlag, 1994.
13. J. Steensland. *Efficient Partitioning of Structured Dynamic Grid Hierarchies*. PhD thesis, Uppsala University, 2002.
14. J. Steensland, S. Chandra, and M. Parashar. An application-centric characterization of domain-based sfc partitioners for parallel samr. *Ieee Transactions on Parallel and Distributed Systems*, 13(12):1275–1289, 2002.