HDDA[1]: A Distributed Dynamic Data-Structure for Parallel Adaptive Computations

Introduction

Dynamically adaptive techniques for the solution of partial differential equations (PDE's) seek to improve the solution accuracy achieved with a fixed amount of computation. This is done by dynamically refining the computational mesh to focus effort and resources (CPU and memory) to areas where solution errors are high. Adaptive methds thus employ locally optimal approximations to yield highly advantageous cost/accuracy ratios as compared to methods based upon static uniform approximations. Parallel implementation of these methods offer the potential for accurate solutions of physically realistic models for systems such as oil reservoirs, black hole and neutron star interaction, and the whole earth. The family of adaptive techniques includes Adaptive Mesh Refinement (AMR), hp-Adaptive Finite Elements and Adaptive Fast Multipole methods. A snapshot of the adaptive grid hierarchy associated with the AMR technique is shown in Figure 1.

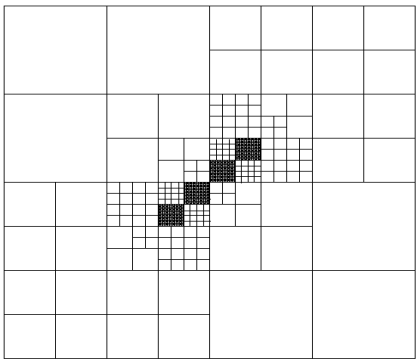

Figure 1 AMR Grid Structure

Data-Structures for Parallel Adaptive Techniques

Implementations of numerical algorithms for solving PDE's are typically formulated as operations on arrays. This is because vertices, cells and faces in a static grid can be directly represented as elements of an array. Such array-based formulations have proven to be simple, intuitive and efficient, and are extensively optimized by current compilers. A similar formulation for dynamically adaptive algorithms, then, warrants a dynamic array; i.e. an array that can grow and shrink dynamically, and where each element of the array can be an array. Furthermore, parallel implementations of the algorithms require the array to be distributed. Like conventional arrays the dynamic distributed array must translate index locality (corresponding spatial application locality) to storage locality. This array must additionally maintain this locality through its dynamics and distributions.

Hierarchical Distributed Dynamic Array (HDDA)

The HDDA data-structure implements a hierarchical distributed dynamic array, providing pure array access semantics to hierarchical, dynamic and physically distributed objects. The array is partitioned and distributed across multiple address spaces with communication, synchronization and consistency managed transparently. The lowest level of the array hierarchy is an object of arbitrary size and structure. The HDDA design is based on the fundamental software engineering design principle[i] of *Separation of Concerns*. Applying separation of concerns to a convention array data-structure, it can be decomposed into an ordered index-space, storage corresponding to each index in the index-space, and accesses mechanism that enable retrieval of data objects associated with an index. Similarly, the HDDA is composed of (1) a hierarchical, extendible index-space and (2) associated distributed dynamic storage and access mechanisms.

Hierarchical, Extendible Index Space: The hierarchical extendible index space component of the HDDA is derived directly from the application domain using space-filling mappings[ii], which are recursive mappings from N-dimensional space to 1-dimensional space. The solution space is first partitioned into segments. The space-filling curve then passes through the midpoints of these segments. A 2-dimensional mapping is shown in Figure 2. Space filling mappings encode application domain locality and maintain this locality through expansion and contraction. The self-similar or recursive nature of these mappings can be exploited to represent a hierarchical structure

[1] Hierarchical Distributed Dynamic Array

and to maintain locality across different levels of the hierarchy. The hierarchical index-space is used by the HDDA as the basis for application domain partitioning, as a global name-space for name resolution, and for communication scheduling.

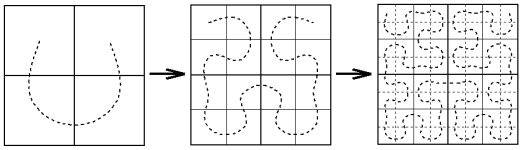

Figure 2 Space Filling Mapping

Distributed Dynamic Storage and Access: Data storage is implemented using extendible hashing[iii] techniques with contractions of the hierarchical index-space indices serving as hash keys. Extendible hashing provides efficient management mechanisms for dynamic databases. Spans of the hash keys space are mapped to units of storage called hash buckets and expansion and contraction of the key space are handled efficiently by splitting and merging these buckets. These operations are local involving at most two buckets. As spans of the index space are mapped to contiguous storage within buckets by the hashing scheme, index locality (which encodes applications domain locality) is translated into storage locality. Data locality is preserved without copying.

Partitioning of the applications domain and associated distribution of HDDA objects is achieved by partitioning the index space among processing elements and accordingly assigning ownership to HDDA buckets. Buckets are used as the units of communications and caching. The overall HDDA distributed dynamic storage scheme is shown in Figure 3

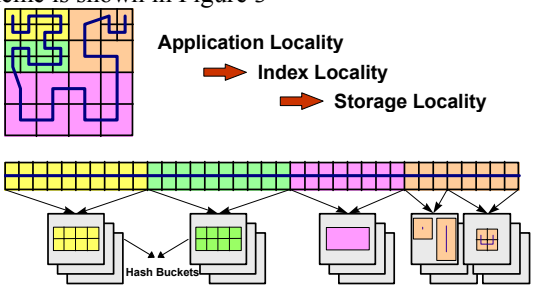

Figure 3 HDDA Storage Mechanism

Applications of HDDA

HDDA has been used as the underlying data-structure in the DAGH framework for parallel adaptive mesh-refinement applications. DAGH is being used for data-management in national grand-challenge projects involving the simulation black-hole collisions and neutron star interactions, for oil reservoir simulations, and for seismic whole earth models. HDDA/DAGH is also being used as the basis for integrating interactive visualization and steering capabilities with adaptive computations in these projects. For more information on HDDA and DAGH see http://www.caip.rutgers.edu/~parashar/DAGH.

---

[i] M. Parashar and J.C. Browne, "System Engineering for High Performance Computing: The HDDA/DAGH Infrastructure for Implementation of Parallel Structures Adaptive Mesh Refinement", IMA Volumes in Mathematics and its Applications, Springer-Verlag, 1997.

[ii] Hans Sagan, Space "Filling Curves", Springer-Verlag, 1994.

[iii] H.F. Korth, A. Silberschatz, "Database System Concepts", McGraw Hill. New York, 1991.