A Dynamic Solution-Adaptive Unstructured Parallel Solver

Roy Williams

Concurrent Supercomputing Facilities
California Institute of Technology
roy@ccsf.caltech.edu

1. Introduction

This paper introduces CARDS: a Concurrent Adaptive Reaction-Diffusion Solver. CARDS is designed for solving time-dependent nonlinear parabolic PDE's in 2 or 3 dimensional domains with quantitative error control, using an extremely flexible mesh structure. The target machine is a distributed-memory parallel processor with vector or pipeline hardware on the processors, but CARDS also runs efficiently on shared-memory and uniprocessor machines with and without vector hardware.

The parallelism is built on the idea of a Voxel Data Base (VDB). This is a distributed shared memory, where entities which share memory are those at the same geometric position. A VDB may be thought of as a dictionary of position-subscript pairs, so that data may be associated with points in space by using the subscript corresponding to that point as an index into data arrays. The use of subscripts allows ease of programming and vectorization. The VDB may be sorted to put special points at the beginning or end of the subscript list, and also to reduce cache-miss inefficiencies. Support is provided for distribution of data entities between processors by a recursive bisection method for load-balancing.

CARDS uses linear finite elements. At present, it solves non-linear Poisson equations on polygonal and polyhedral domains, using Newton's method, solving the associated linear equations by diagonally preconditioned conjugate gradient. Spatial accuracy is achieved [2] by refining the mesh sufficiently that a discrete second derivative times the square of the mesh spacing is everywhere below a user-specified bound. This spatial solver will be used with an implicit finite element time-integration scheme described in [2, 3], which has quantitative error bounds even for adaptive meshes.

Simplices

The mesh used by CARDS consists of simplices: a simplex is the collective term for line-segments, triangles and tetrahedra. The major advantage of a simplex mesh is that software may be written with the dimensionality d as a compile-time or runtime variable. For example the measure (length, area, volume) of a simplex is a determinant of order d+1.The boundary of a d-simplex consists of d+1 simplices of dimension d-1, defining a hierarchy of lower-dimensional simplices. We shall refer to 0-dimensional simplices as points or nodes, 1-dimensional simplices as edges, d-1 dimensional simplices as faces, and d-dimensional simplices as elements. In the special case of d=2 dimensions, faces are the same as edges.

2. Voxel Data Bases

In this paper we shall use a method for writing geometrically local algorithms [20]. Such an algorithm is one where objects at the same geometric position may communicate: here such communication is effected by a shared memory at that point.

This view of data movement is exemplified, but not limited to, the Finite Element method [7]. A Finite Element mesh is a partitioning of the problem domain into elements, each of which is defined by a set of points in the domain, called nodes. The basic structure is that of a set of elements, where each element is a set of nodes. The computational kernel consists of each element reading and writing from and to its subset of the nodes. Writes to the nodes are separated from reads of those data by a loose synchronization [5]. Each element refers to a node by local numbering; for example the nodes of a tetrahedron might be 1, 2, 3, 4. In addition to the local numbering, there must also be a global numbering, so that the nodes may be thought of as shared memories: two elements referring to the same node will refer to the same global node number and thus share memory.

We may split a Finite Element code into two parts, one which deals with the properties and nature of the data stored with elements and nodes, and the calculations done with those data, and another part which maintains the shared memories at the nodes and supplies the mapping from local to global numbering. The latter part may be done with a Voxel Database (VDB), producing code to run on a uniprocessor or on a distributed machine.

A Set of Sets of Points

The data structure for a VDB is a collection of subsets of a set of points. We shall call these sets of points *pointsets*. A