Chapter 1

Introduction

CAVEAT EMPTOR
(let the buyer beware)

Imagine having a fast, expensive parallel computer, and finding out that half the time one processors is holding up all the others, which need the results of this one processor to be able to compute further. Any means of distributing the single processor's work among all other processors would make the computation complete in little more than half the time, and would make the processor seem twice as fast to its users. Load balancing attempts to spread the work evenly among processors, so a given machine will be able to run as fast as possible.

As money is a measure of wealth, so *computational load* is a measure of the amount of work that will be done by a portion of a program. As money is also a measure of cost, computational load is likewise a measure of the cost of executing a part of a computation. The definition of load used in this dissertation is the time required for a given processor to complete a portion of a computation. In a parallel computer, different parts of a computation may be executed in parallel by different processors; *load balancing* is the process of distributing parts of the computation to processors, with the goal of having each processor complete the computation assigned to it in the same time as all other processors. A load balancing algorithm is *efficient* if the time required for balancing the load is short; a load balancing algorithm is *effective* for a given computation if balancing the load causes the computation to complete in less time than is required to run the computation in the absence of load

balancing. Because the parts of a program often need to coordinate their execution, load balancing may have to use criteria that are unrelated to load in deciding which parts should be assigned to which processor.

A *scan* is an operation that takes as input a vector and produces as its result a different vector with the same number of elements as the input vector. Each element in the result of a *left* scan is a function (such as the sum) of all the elements in the input vector that are to its left; each element in the result of a *right* scan is a function of all the elements of the input that are to its right. For many useful functions such as sum and product, scans can be computed efficiently in parallel on almost any parallel computer, and scans are therefore a good foundation for an algorithm that is designed to improve the speed of a computation.

The focus of this dissertation is Scan Directed Load Balancing, a new algorithm for load balancing. The algorithm uses scans to find out what the current load distribution is and to determine how the load distribution can be made more even.

Scan Directed Load Balancing is designed for parallel computers in which the processors are connected in a *rectangular mesh*: a network that has the same structure as a grid, with a processor at each grid intersection, and with every line in the grid representing a connection between adjacent processors. In this dissertation, the word *mesh* refers to a rectangular mesh of processors, and the word *grid* refers to a rectangular grid of data.

The dimension of a rectangular mesh is the dimension of the space in which the mesh can be laid out. A mesh of dimension one is a linear array of processors, with each processor connected to the processors before and after. A mesh of dimension two is a square or rectangular array of processors, and a mesh of dimension three is a cubic array of processors; meshes of higher dimension are possible but have no intuitive counterparts.

Each processor in the mesh has connections to other processors, referred to as *neighbors*. Processors in the interior of the mesh always have $2k$ neighbors. Processors that have fewer than $2k$ neighbors are *edge* processors, and may have as few as k neighbors. An "edge" processor may be an endpoints of a one dimensional array, in a corner or on a side of a square array, or may be in a corner, on an edge, or on a face of a cube.

Many parallel computations operate on data that is structured as a rectangular grid of a given dimension. In a *local computation*, each operation is computed using only data and intermediate results from a region within a

small fixed distance (on the grid) of the result, the *neighborhood* of the result. Normally the dimension of the data grid is the same as the dimension k of the processor mesh; if not, the data grid can be mapped to have dimension k by an appropriate transformation.

There are many ways to decide which processor must process which data points. A *hierarchical* distribution uses a coarse grid to divide the finer data grid into as many (k dimensional) rectangular areas as there are processors, and assigns the data from each cell of the coarser grid to a single processor. This distribution minimizes the communication requirements of local computations, because the neighborhood of most data points within a processor is in that processor, and those that aren't are split among neighboring processors that have direct connections to one another.

Scan Directed Load Balancing is designed to balance the load of processors in rectangular meshes that are executing local computations on rectangular grids with the data distributed hierarchically among the processors. First, a scan based computation determines where the coarse grid should be placed to make the loads on the processors as even as possible; then the data is shifted among the processors so that the actual distribution corresponds to placement of the coarse grid.

Scan Directed Load Balancing can be applied to a wide variety of problems that have data grids and that for any reason partition the data hierarchically; one possible reason is to minimize communication in local computations. Physical simulations of *continuum* problems, in which the computed values of physical quantities at each point are only dependent on the values within a neighborhood, are one large class of computations that can benefit from Scan Directed Load Balancing. The class includes simulations of the atmosphere, of integrated circuits, of the airflow around a wing or hull, and of the behavior of a gas of molecules at different temperatures. Other continuum computations for which Scan Directed Load Balancing is well suited include computations in image processing, dynamic data management, signal processing, and image generation. In this dissertation, Scan Directed Load Balancing is applied to two computations: Variable Conductance Diffusion, an image processing algorithm, and the SOP interpreter for the FFP language, which is based on the design for the FFP Machine parallel architecture.

Variable Conductance Diffusion is a diffusion algorithm for images, in which each *pixel* (picture element) repeatedly modifies its value to be closer to the average of its neighbors; the image is thereby somewhat blurred, but

any noise in the image is canceled out. The difference between Variable Conductance Diffusion and normal diffusion is that Variable Conductance Diffusion tries to prevent blurring of sharp edges by not modifying the value of a pixel when the pixel's neighbors differ by more than a certain amount. This produces results that are more pleasing to the eye and easier to understand and look at than standard diffusion. One application of Variable Conductance Diffusion is to improve the display of medical images such as the ones produced by CT and sonar scanners. Speeding up this process with Scan Directed Load Balancing is potentially a step towards the goal of real-time processing for these images.

The FFP Machine is a general-purpose parallel architecture that is particularly well-suited for executing programs written in the FFP functional language. In this architecture, the program is stored in a linear array of cells; some of these cells store the program and some store the data for the program. Each cell in the machine does different things depending on the program or function it is executing: it can copy itself to a different part of the string, rewrite itself to a different symbol, or perform symbolic or arithmetic operations. These operations define the SOP (String OPeration) language, the "machine language" of the interpreter, in which primitive operations modify structured strings. Many SOP operations either allocate ("create") new cells or erase one or more cells, and an efficient dynamic load and storage manager is therefore essential. The load and storage manager for SOP is based on Scan Directed Load Balancing. The SOP language has been used with the FFP Machine architecture to execute MIMD programs on SIMD computers. While the direct usefulness of this implementation is limited, it is part of continuing research to identify useful features of existing computers and to explore new, desirable features for future architectures.

This dissertation makes several original contributions. The development of Scan Directed Load Balancing itself from a different, one-dimensional algorithm that uses permutation instead of mesh communication is one contribution. Another is the extension of the algorithm to more than one dimension. The development of an algorithm for respecting storage constraints while balancing load is also a novel contribution. The identification and analysis of a class of domain decompositions that always preserve locality in rectangular processor meshes has been only done incompletely in the past. The experimental verification of the effectiveness of Scan Directed Load Balancing for Variable Conductance Diffusion and SOP is of course new. The development

of the SOP language is new, though its model of reduction is only marginally different from that of the FFP Machine architecture. The possibility of executing SOP efficiently on SIMD machines, and SOP's strategy of storing code in cells, whereas the FFP Machine has a separate mechanism for storing code, also distinguish the two.

The statement of this thesis is that Scan Directed Load Balancing is an effective algorithm for dynamic load balancing on multi-dimensional, mesh-connected computers

The rest of this dissertation is organized as follows. Chapter 2 gives an overview of existing load balancing schemes and describes in greater detail scans, continuum computations, and mesh connected parallel computers, as well as both the FFP Machine and the algorithm for Variable Conductance Diffusion. Chapter 3 presents the complete algorithm for Scan Directed Load Balancing and some theoretical results about its efficiency and effectiveness. Chapter 4 describes the implementation of Variable Conductance Diffusion on the MasPar MP-1, a mesh connected computer, and analyzes the performance of Scan Directed Load Balancing in load balancing the computation of Variable Conductance Diffusion. Chapter 5 describes the SOP language and the implementation of the SOP interpreter for the MP-1, and analyzes the effectiveness of both SOP and Scan Directed Load Balancing. Finally, Chapter 6 reviews the features of Scan Directed Load Balancing and suggests directions for future work that are likely to prove profitable and interesting.