THE CACHE COHERENCE PROTOCOL
OF THE DATA DIFFUSION MACHINE

Seif Haridi and Erik Hagersten
Swedish Institute of Computer Science
Box 1263, S-164 28 Kista, Sweden

ABSTRACT

The Data Diffusion Machine (DDM) is a scalable shared virtual memory multiprocessor in which the location of a datum in the machine is completely decoupled from its virtual address. In particular, there is no distinguished home location where a datum must normally reside. Instead data migrates automatically to where it is needed, reducing access times and traffic.

The hardware organisation consists of a hierarchy of buses and data controllers linking an arbitrary number of processors each having a large set-associative memory. Each data controller has a set-associative directory containing status bits for data under its control. The controller supports remote data access by "snooping" on the buses above it and below it. The data access protocol it uses provides for the automatic migration, duplication and replacement of data while maintaining data coherency.

The machine is scalable in that there may be any number of levels in the hierarchy. Only a few levels are necessary in practice for a very large number of processors. Most memory requests are satisfied locally. Requests requiring remote access generally cause only a limited amount of traffic over a limited part of the machine, and are satisfied within a small time that is logarithmic to the number of processors. Although designed particularly to provide good support for the parallel execution of logic programs, the architecture is very general in that it does not assume any particular processor, language or class of application.

1 INTRODUCTION

Message-passing machines and shared-memory machines are the two main classes of parallel (MIMD) computer, and are generally considered to be quite distinct. Message-passing machines typically have many processors with large private memories, linked together by a communications network. Shared-memory machines typically have only a limited number of processors with small private memories or caches, connected by a common bus to a large, physically shared, memory. Message passing machines usually require software to view memory access and communication with other processors as quite separate mechanisms. Software often simulates a form of shared virtual memory, by translating references to remote objects into appropriate messages. Shared-memory machines, on the other hand, usually support shared virtual memory directly, thereby allowing software to achieve communication implicitly through memory access, but require some locking mechanisms to support this. Message-passing machines are generally scalable to arbitrary numbers of processors, whereas in shared-memory machines the shared bus and memory is a bottleneck, placing a limit on the number of processors that can be attached. However, message-passing machines place a much heavier burden on software to partition the computation effectively, and so the scalability of the hardware is only useful insofar as the software can keep communication to a minimum.

The DDM is like a message-passing machine in that memory is distributed and the machine is scalable to an arbitrary number of processors. The DDM is like a shared-physical-memory machine in that it supports a shared address space and processors are connected via buses. The key idea behind the DDM, which distinguishes it from both message-passing machines and shared memory
machines, is that the location of a data item in the machine is completely decoupled from its virtual address.

The design of the DDM is based on the following considerations. Where a piece of data resides is not really relevant to the software. Ideally, the physical location of data should be transparent to the software. All the software needs is some means of identifying each data item, which is just the virtual address. Rather than have software control the physical placement of data, this should be taken care of automatically by hardware. Thus virtual addresses should be mapped into physical addresses in a totally flexible manner. The mapping should be dynamic, allowing data to migrate to where it is most needed. It may be desirable to have multiple copies of a particular data item, but they will all share the same virtual address. To summarize, from a software point of view there will be a number of processes sharing data that is arranged logically in a single virtual address space; from a hardware point of view, processes will be mapped into processors and virtual addresses into physical addresses in such a way that most of a processor's memory accesses can be satisfied by its local memory. In other words, the data structure that the software sees will distribute itself automatically over the machine in such a way as to reduce data access times and minimize data traffic.

The DDM was motivated by our work on logic programming execution models and represents our ideas on how these models can best be supported by hardware. The design, however, is very general in that it does not assume any particular kind of processor, language or application. We feel this is very important if the machine is to gain practical acceptance, and is an important factor in the commercial success of machines such as the Sequent. It should be noted that software designed for conventional shared-memory machines can run without change on a DDM.

The remainder of the paper is organized as follows. In the first section we describe the main feature of the hardware organization. The next section is an introduction to the protocol. Next is a discussion over the need for and implementation of replacement, followed by some remarks of the hardware requirements of the machine. Next we analyze performance characteristics of the machine, and compares it with other architectures. We conclude the paper by bringing up various other issues and a summary of the main novel features of the design. At the end of the paper protocol tables defining the protocols used is to be found.

2 OVERVIEW OF THE ARCHITECTURE

The machine is hierarchical (see fig. 1). At the tips of the hierarchy are processors each with a large local memory (possibly accessed via a conventional cache). The memory contains an image of some part of the global virtual address space. The memory is set-associative, and is organized like