1. Introduction

The Massive Memory Machine Project (MMM) was started at Princeton around 1983. Year by year, semiconductor memory was becoming cheaper (and still is) and memory chip densities were increasing dramatically. We were interested in studying how availability of very large amounts of memory would change the way data intensive problems could be solved. We believed that just like parallel computer architectures could speed up scientific computations, computer architectures and software that exploited massive main memories could yield order-of-magnitude performance improvements on many important computations.

By the end of 1985 we had received joint funding from the U.S. National Science Foundation and the Defense Advanced Research Projects Agency. This formal project has two main goals. One was to study key data intensive applications and to develop strategies for effectively utilizing large memories. The second was to design, implement, and test computer architectures that could support fast access to very large memories.

In this paper we will briefly summarize four of the MMM sub-projects having to do with data processing applications. For one sub-project, we implemented a memory resident transaction processing system (System M), of the type that might be used in an airline reservation or banking application. The second sub-project is an advanced file management system (iPcress). A third sub-project (HyperFile) involves a document management system that can be used to support hypertext style applications. Since memory resident data can be accessed very fast, a fourth sub-project studied real-time database management techniques.
This is not a complete summary of the MMM activities, as there were other projects in the architecture and performance evaluation areas. In addition to the authors of this paper, there were other individuals who contributed to the project in general and to the data management sub-projects. These include Rafael Alonso, Daniel Barbara, Richard Lipton, Arvin Park, Jonathan Sandberg, and Jacobo Valdes.

As part of our summary we will reference various papers we have written that give more details and results. These papers discuss in detail the relationship to other work in these areas, so to avoid repetition, we will not include here references to this other related work. We also note that even though NSF and DARPA funding for MMM ends in 1990, some of our sub-projects are continuing with other funding. In particular, we are currently actively working on the iPcress and HyperFile systems. We also have a strong interest in continuing to study real time and transaction processing issues related to massive memory.

2. System M

System M is an experimental transaction processing system with a memory resident database. System M was built with three goals in mind:

(1) To evaluate the potential gains of memory resident databases. Given the current costs of memory, it is now feasible to store in main memory realistically large databases. The improvements in performance can be great: I/O is substantially reduced, transaction context switches (and associated CPU cache flushes) are cut, lock contention is decreased, more efficient memory search structures and query processing can be used, and so on.

(2) To explore the architecture and algorithms best suited for memory resident databases. A memory-resident transaction processing system (MTPS) could be implemented simply as a disk-based system (DTPS) with a buffer that happens to be large enough to hold the entire database. This approach fails to capitalize on many of the potential advantages that memory residence offers. System M, on the other hand, has been implemented from scratch with memory residence in mind, leading to a novel internal process architecture and recovery strategies.

(3) To provide and experimental testbed for comparing algorithms. In particular, System M implements several checkpointing and logging strategies. (The checkpointer is the component that periodically sweeps memory, propagating changes to the backup database on disk.)

We focused on checkpointing and logging because we believe they are the major performance issues in a MTPS. In other words, conceptually, the data management algorithms for a MTPS and a DTPS are similar. Both types of systems have data in memory and on disk, and must propagate the updated data to disk. Both mechanisms must have concurrency control mechanisms. However, a MTPS will do many fewer I/O operations: the only I/O is to maintain transaction durability. This implies that recovery I/O (for durability) is relatively much more important than in a DTPS. Recovery I/O requirements should be satisfied without sacrificing the advantages that memory resident data can bring.

To achieve this, it is essential to decouple the checkpointing and logging components as much as possible from the rest of the system, so that transaction processing rarely has to wait for them. It is