Exploiting Parallelism:The Tera Computer System and the Multiscalar Processors

Romarie U. Lorenzo
Department of Electrical and Electronics Engineering
University of the Philippines

Abstract--**Parallelism introduces complexities both in hardware and in software. Studying the Tera Computer System and the multiscalar processor gives us a glimpse of these.**

I. INTRODUCTION

Simultaneous multithreading (SMT)[1] was compared to different architectures in our class reading. The article mentions several alternative approaches to exploit parallelism. In this paper, we explore the Tera computer system[2] and the multiscalar processor[3]. We describe their basic characteristics and take a look at what happened in history: how it fared in industry for the case of the Tera architecture and the progress of present researches for the case of the multiscalar processor. Finally, we give some insights regarding the complexities in the implementation of parallelism.

II. THE TERA COMPUTER SYSTEM

A. Characteristics

The Tera Computer System was built to be suitable for very high speed computation, which is equivalent to having a short clock period. It was made to be scalable to many processors, applicable to different types of problems, even to those that do not vectorize well, and was made to have easy compiler implementation.

The Tera MTA (multithreaded architecture) uses fine-grained multithreading and long instruction words. Each clock cycle, a processor selects a stream ready to execute. Each stream has an associated state. Fast context switching gives no time to swap state. Instead, the stream state is replicated 128 times to make this work. Each instruction typically specifies three operations: a memory reference, an arithmetic operation and a control operation.

Its interconnection network features a uniform distribution of resources. That is, every processor has equal access to every memory location. This makes it easy to program because concerns for the memory layout are eliminated[10]. On the other hand, if p is the number of processors, the number of network nodes grows as $p^{3/2}$ rather than $plog(p)$ which results when using multistage networks. That is, the architecture has some additional overhead per processor.

Memory latency is fully masked by parallelism only when the number of messages being routed by the network is at least $p \times l$, where l is the round-trip latency. On the example implementation, 70 different streams would be required on each processor to hide the expected latencies if the next instruction was not allowed to execute before the previous one completes. Instruction lookahead or exposing the pipeline could help. However, the Tera architecture uses a new technique called explicit-dependence lookahead to help reach its peak performance. A three bit lookahead field specifies how many instructions from the stream will issue before a dependent instruction is encountered. Using this, at most eight instructions and twenty-four operations can be executing at the same time for each stream.

B. How the Tera Architecture Fared

MTA is the architecture developed by Tera Computers. It is capable of delivering high performance as promised. In March 1997, Tera Computer Company announced that MTA computer recorded the fastest single processor performance ever when running the NASA NAS Parallel Integer Sort benchmark. Tera ran the Integer Sort in only 1.54 seconds, beating any other supercomputers by more than 10 percent using a prototype of its 1 Gigaflop-peak multithreaded architecture[4].

Unfortunately, it is not just the performance that matters. In the year 2000, Tera Computer Company acquired Cray Research to form Cray, Inc. The Tera MTA system was relaunched as the Cray MTA-2. Although this computer is quite promising in performance due to its built-in scalability, it was not a commercial success and only shipped to two customers[5]. The original Tera computer turned out to be nearly unmanufacturable due to its aggressive packaging and circuit technology[6].

III. MULTISCALAR PROCESSORS

A. Characteristics

For a multiscalar processor, a single program is divided into a collection of *tasks* by a combination of software and

hardware. These multiple tasks execute in parallel on the processing units. The key, then, to making this work is the proper resolution of inter-task dependencies. The processor should be able to establish a large and dynamic window of instructions from which parallel execution is scheduled. The multiscalar processor walks through the control flow graph(CFG) task by task, as opposed to instruction by instruction.

This processing procedure imposes requirements for multiscalar programs. They would need to supply particular information to make this work out-- the actual code for the task which accomplishes the work, the details of the CFG structure and the communication characteristics of the individual tasks. To handle register values, a *create mask* is used to indicate the register values a task may produce.

On the hardware side, a *sequencer* will be needed to determine the order of the tasks and an Address Resolution Buffer(ARB) is provided to keep track of the data cache.

The best performance is achieved when the processor capability is matched to the useful computation cycles. There are many techniques that could be employed to minimize non-useful cycles. Data communication could be synchronized and the prediction could be validated early to decrease the non-useful computation cycles. Intra-task dependences and inter-task dependences were found to be sources for no computation cycles. Also, the load should be balanced by properly selecting the grain size of a task.

B. Multiscalar Work In Progress

Works on the multiscalar processor is being carried out at the University of Wisconsin[8]. As they also pointed out, this architecture requires new, never-before-designed hardware structures and compiler algorithms. From their site, we can see that research is being carried out to fine-tune different aspects of the architecture.

IV. INSIGHTS

From these two architectures, we got a glimpse of the complexities that could arise in exploiting parallelism. For the Tera MTA, the hardware made it hard to manufacture in its time. This reminds us that when we design, we should also take note of its manufacturability. But we should be open for these types of things because as we can see, this architecture could be possible in the future. It's high scalability may not have been needed in its time. But it is good to note that it is easy to program. Also, we should note that for commercial purposes, we should design a computer for manufacturability.

Meanwhile, for the multiscalar architecture, a lot of logic has to be implemented--from breaking down the program into tasks up to the synchronization of data communication. Programs for multiscalar also add burden to the programmer. These type of architecture naturally leads to the study of quite a number of different techniques. Other architectures could probably take a look at the design considerations from these studies that could be applicable for them as well.

REFERENCES

[1] S.J. Eggers et al., "Simultaneous Multithreading: A Platform for Next-Generation Processors," IEEE Micro, 1997, pp.12-19.
[2] R. Alverson et al., "The Tera Computer System," Proc. Int'l Conf. Supercomputing, Assoc. of Computing Machinery, N.Y., 1990, pp.1-6.
[3] G.S. Sohi, S.E. Breach, and T. Vijaykumar, "Multiscalar Processors," Proc. Int'l Symp. Computer Architecture, ACM, 1995, pp.414-425.
[4] "Tera Computer Company," HPCWire, 1998, http://www.hpcwire.com/hpc-bin/artread.pl?direction=Current&articlenumber=71097
[5] "Cray," Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/Cray_Inc.#Cray_Inc, accessed August 2006.
[6] "Cray MTA-2," Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/Cray_MTA-2, accessed August 2006.
[7] C.H.Stork, "The Tera Architecture: Concepts and Experiences," www.ics.uci.edu/~cstork/talk/index.htm
[8] Wisconsin Multiscalar Group. http://www.cs.wisc.edu/~mscalar/index.html
[9] J. Hickey, "Tera Computer's Multithreaded Architecture," Virtual Medical Worlds, November 1999.
[10] "Cray History," from Cray, Inc. website, http://www.cray.com/about_cray/history.html