Topic 08
Parallel Computer Architecture

Silvia Müller, Per Stenström, Mateo Valero, and Stamatis Vassiliadis

Topic Chairpersons

Computer architecture is a truly fascinating field in that improvements in the basic technology and innovations how to make best use of the underlying technology has yielded a performance growth exceeding a million times over the past 50 years. What is even more amazing is the fact that the pressure on maintaining this rate of performance growth shows no decline. In fact, as performance thresholds are passed, application designers face new opportunities that give new challenging problems to work on for computer architects.

Parallelism and locality are the two fundamental concepts from an architecture point of view that have contributed to the impressive performance growth. Exploitation of parallelism has led to an increased pressure on the memory system. The increased speed-gap between processor and memory has in turn fueled innovations in memory hierarchy research that exploit locality.

Until now, the major form of parallelism that has been exploited at the microprocessor level is across instructions. Coarser-grained, or thread-level parallelism, is becoming increasingly important to consider for the following two reasons: First, there are always computational problems at any one time whose performance demands can not be accommodated by a single processor, such as various forms of transaction and database processing and scientific/engineering computing. Second, exploiting instruction-level parallelism is yielding diminishing returns owing to the complexity involved in considering larger instruction windows. Both of these observations prompt towards also exploiting thread-level parallelism and are the major motivating factors for parallel computer architecture – the topic of this session.

Thread-level parallelism can be exploited at the chip as well as at the system level. Two architectural styles at the chip level are currently being debated: chip multiprocessors and multithreaded architectures. Independent of the architectural style chosen at the chip level, how thread-level parallelism is exploited across microprocessor chips, which act as processing nodes, is an important issue in the area of parallel computer architecture.

Historically, message passing (or distributed memory) and shared-memory multiprocessors are the prevailing parallel computer architectural styles at the system level. In message passing, the software abstraction forces threads to explicitly exchange messages between disjoint address spaces whereas in shared-memory threads exchange messages implicitly in a common address space. In implementing any of these abstractions, a fundamental issue is to reduce the impact of inter-thread message communication latency on the execution time of parallel programs. All the papers in this session address in one or another way

A. Bode et al. (Eds.): Euro-Par 2000, LNCS 1900, pp. 537–538, 2000.

this fundamental problem by either proposing innovative solutions to reduce or tolerate the message latency, or by making important observations regarding the nature of the inter-thread communication pattern in parallel programs to be used to identify new approaches for efficient communication.

In shared-memory multiprocessors inter-thread communication results in coherency interactions between threads that may hurt performance. In the first paper, Acquaviva and Jalby study the nature of these interactions based on analysis of a suite of scientific codes and make interesting observations regarding what program behavior causes performance problems. These observations are important in order to find more efficient coherency mechanisms.

In message-passing systems, the thread that sends the message often has to wait until the receiver has copied the data into its address space. One interesting contribution in the second paper by May et al. is the introduction of a new message passing protocol that allows the sender to copy the data directly into the address space of the receiver.

Replication of data is an effective means to avoid some of the communication in shared-memory machines and the COMA concept enables replication also at the memory level. In the third paper, Ferraris et al. use the COMA concept to propose a multiprocessor architecture using workstations as building blocks. They report on a COMA protocol that reduces the overhead associated with replication.

For small-scale systems, bus-based multiprocessors have dominated the market for some time and are also considered for chip multiprocessors. A problem with these systems is that inter-thread communication can cause severe bus contention. In the fourth paper, Milenkovic and Milutinovic propose an innovative solution, called cache injection, to reduce the bus traffic.

Finally, the topic of the last paper by Talbot and Kelly is again on replication at the memory level in cache-coherent NUMA machines. In these machines, widely shared memory blocks can cause performance problems if the cache space is not sufficient. In their proposal, called adaptive proxies, a mechanism is proposed that adaptively replicates only the data that is simultaneously shared by a large number of nodes.

We hope you will enjoy and learn a lot from this collection of papers.