

## Multi-Threaded Systems: Issues, Solutions And Future

**Krishna M. Kavi**  
**The University of Texas at Arlington**

**and**  
**Ali R. Hurson**  
**The Pennsylvania State University**

Recent studies have shown that the single-threaded paradigm used by conventional programming languages and run-time systems can utilize less than 50% of the processor capabilities. Yet advances in VLSI technology have led to faster clocks and processor designs that can issue multiple instructions per cycle with more on-chip cache memories. In order to garner the potential performance gains from these technological advances, it is necessary to change the programming paradigm. Multithreading has emerged as one of the most promising and exciting avenues for exploiting the technological advances. A multithreaded architecture contains multiple "loci of control" (or threads) within a single program; the processor is shared by these multiple threads leading to higher utilization. The processor may switch between the threads to hide memory latencies or interleave instructions from multiple threads to minimize pipeline breaks due to dependencies among instructions within a single stream.

The idea of multithreading is not new. Fine-grained multithreading was implicit in the dataflow model of computation. Multiple hardware contexts (i.e., register files, PSW's) to aid switching between threads were implemented in systems such as Dorado and HEP. These systems were not successful due to a lack of innovations in programming languages, run-time systems and operating system kernels. There is, however, a renewed interest in multithreading primarily due to a confluence of several independent research directions which have united over a common set of issues and

techniques. A number of research projects are underway for designing multi-threaded systems including new architectures, new programming languages, new compiling techniques, more efficient interprocessor communication and customized microkernels. Some of these projects have produced substantial improvements over single threaded abstractions. For example, Tera Computers has developed a multithreaded architecture (MTA) where each processor of a multiprocessing system can interleave instructions from as many as 128 active threads. There is a need for further research into architecture as well as operating systems and compilers. For example, research into cache memory management to achieve acceptable hit rates when multiple threads share the cache memories is needed. Compile time optimization to properly interleave multithreaded programs is needed. Support for multi-level scheduling and efficient switching between multithreaded tasks are needed. Programming environments to permit users to write multithreaded applications is needed. The success of multithreading as a viable computational model depends on the integration of these efforts.

The main goal of this minitrack is to facilitate a forum for the presentation of successful research on the various aspects of multithreading and exchange of ideas among the participants. A total of 20 papers were submitted to the minitrack and 9 very high quality papers were accepted for presentation at the minitrack. The papers are divided into 3 sessions.