Summary
Helper Threads via Virtual Multithreading on An Experimental Itanium 2 Processor-based Platform.
Kiran Yellajyosula
February 9, 2005

1 To be completed before class

What are the problems solved by this paper? (50 words)
Helper threads are being employed to improve the throughput and performance of a processor by prefetching hard-to-predict data accesses. The processor has to stall whenever there is a last level cache miss and a subsequent time of invocation of a helper thread by the OS is unpredictable. Helper threads need to be adaptable and should not contend with the resources of the main thread.

What are the approaches attempted by this paper? (50 words)
The paper introduces a technology by which a single thread processor can support multiple thread contexts and switch between them without needing the help of the OS. The firmware of the Itanium 2 processor is programmed to track hardware events and jump to a helper thread whenever there is a pipeline stall or an explicit instruction.

What are the main conclusions of this paper? (50 words)
The paper improves on the performance of three workstation programs and six database queries of an 100 GB IBM database. They design a fly weight context switch process by partitioning the register set. As the performance gap between processor and memory increases, VMT helper threads can improve the performance latency.

2 To be completed after class

Did this paper address an important issue? Explain. (100 words)
The paper addresses the issues involved in helper thread invocation and helper thread adaptability. The helper threads are used to prefetch hard to predict data by running ahead of the main thread. The helper threads are also not to hinder the execution of the main thread. In the usual processors, helper threads are scheduled by the OS, which adds more overhead to the context switch time. This requires the helper threads to run much further in advance to do any useful prefetching. The helper threads also need to be able to throttle themselves when they are far ahead.

These issues were addressed by partitioning the Itanium registers and using a subset of them to run the helper threads. The invocation of the VMT helper thread requires around 70 cycles and is done without the OS, there by giving the VMT helper thread about 60 cycles of computation time. The helper threads can self-throttle themselves by using the yield-conditional instruction and can readjust itself based on the iteration counter of the main thread. The VMT system uses the firmware support to detect and jump to a compiler inserted helper thread code on a pipeline stall.
Are the proposed approaches valid? Describe its strength and weakness. (100 words)

The approach was emulated on the Itanium 2 processor and improves the performance of various tested application (3 workstation and 6 database server queries).

The strengths of the system are as follows:

1. It provides a multithreaded interface on a single processor.
2. It programs the firmware to detect such stalls and makes use of the available hardware resources.
3. The context switch between helper threads is faster
4. Inter thread value communication and synchronization is faster and easier
5. The OS is not involved with thread scheduling
6. A nifty approach of using NOPs to support two new instructions.
7. It takes the benefit from the compiler optimizations and profiling techniques to create threads.

The weaknesses of the approach are as follows:

1. Scalability : The number of helper threads involved at a time are limited. Any further approach to support a region with multiple helper threads complicates the whole framework.
2. Register set is explicitly partitioned for helper threads, there by requiring huge register files in the first place.
3. The approach requires a complicated firmware support to be available in the architecture.
4. It has the complexity of coding the firmware.

Do the results support the conclusions? Explain. (100 words)

Yes, the results support the conclusions derived. The authors test their framework on an Itanium processor and improve the performance of some workstation and database server applications. They profiled the number of cache misses and showed that they reduced the number of L3 cache misses by around 48%. They even tested the adaptability of the VMT helper threads by decreasing the size of the L3 cache and testing with various input sets. The results depicted that helper threads were invoked only when the applications encountered L3 cache misses.

Describe the potential future works? (100 words)

This work was emulated on the Itanium 2 processor. Hence a performance improvement can be acheived by implementing support for the VMT technology in the architecture. The issue of supporting a region with multiple helper threads depending on the phase of the program could be an interesting result. A further efficient scheme to shift between threads can be designed in hardware. The authors propose to fine-tune the compiler for further improving the performance. The authors claim that with the increasing processor memory gap, the VMT technology provides a valuable window for helper threads. A study to determine the amount of prefetching required and the upper limit of the obtainable performance with similar techniques would be helpful.