Summary
Helper threads via Virtual multithreading on an experimental itanium 2 processor-based platform

Venkatesan Packirisamy
February 9, 2005

1 To be completed before class

What are the problems solved by this paper? (50 words)
Widening processor - memory performance gap lead to many techniques which try to improve cache performance. One such successful techniques is to use helper threads to prefetch hard-to-predict delinquent data accesses. This technique was successfully used in systems which hardware support for multithreading, like the hyperthreading in P4. But this technique cannot be applied directly to Itanium processor, since it cannot support multiple threads without invoking the OS. This paper designs a hardware mechanism called virtual multithreading which can run the helper threads to prefetch data.

What are the approaches attempted by this paper? (50 words)
The paper uses a technique calle virtual multithreading. Since we have only one thread of execution, we cannot overlap the execution of the helper thread with the main thread. Only way to run the helper thread is when the main thread suffers a cache miss in the L3 cache. So we need some event triggered multithreading support. Existing multithreading techniques involve the OS, which is a huge overhead. So a hardware mechanism is built on the Processor abstraction layer (PAL) of the itanium processor. This acts along with the performance monitoring unit. New instructions are inserted (yield and yield conditional) to start the helper thread. When the cache miss occurs and if the yield instruction occurs, the PAL switches control to the helper thread. Techniques are also used to control the helper thread from going too far. Advanced compiler techniques are used to generate the helper threads. Thus a multithreading execution is emulated in a single thread itanium processor using VMT.

What are the main conclusions of this paper? (50 words)
The VMT technique lead to wall-clock speedups of 5.8% to 38.5% for the workstation benchmarks. The technique also showed good performance for database workloads. So it is showed that through VMT helper thread execution is possible even in a single threaded machine. To achieve high performance with helper threads, the helper threads have to adaptive to the current situation. Since the VMT is tied to program’s dynamic behavior, helper threads through VMT shows good performance. The same techniques can be applied to increase throughput of some programs.

2 To be completed after class

Did this paper address an important issue? Explain. (100 words)
The widening gap between memory and processor performance has led to the development of many techniques which try to improve the cache locality. One such widely used technique is prefetching with the help of helper threads. Though prefetching is possible without the helper threads, those prefetching can be done only for variables whose address is known at some point (either compile time or every early in runtime with predictable pattern). But there are some variables such recursive pointer indirections, whose address cannot be easily calculated. To find those addresses, complete calculations
of their address is needed. This cannot be done by inserting instructions, since this may involve speculative loads. To prefetch such loads we need helper threads. One problem with helper threads is that we need the processor to support multithreading. This paper solves this problem of using helper thread prefetching in a uniprocessor which supports only one thread of execution by using Virtual Multithreading (VMT).

Are the proposed approaches valid? Describe its strength and weakness. (100 words)
The helper thread is called only if there is a cache miss at the L3 cache. Usually the L3 cache latency is few hundred cycles. This gives a time gap for which the processor is idle (Itanium is in-order so the miss cannot be overlapped). The paper uses this time to start the helper thread. Usually in a uniprocessor to start a new thread, we need OS involvement. But this can take hundreds of cycles. To avoid this the paper uses VMT. Here the PMU(performance monitoring unit) keeps track of L3 miss, if it occurs and if the next instruction is a yield instruction, then it transfers the control to the helper thread. And the helper thread starts from where it left (co-routine like execution). The logic for this is implemented in the PAL (processor abstraction layer) thus avoiding OS overhead.

- Strength: The paper exploits the existing hardware support in the Itanium like the PAL and PMU to implement the helper thread. Also since the thread is closely tied with the main thread execution, it is more flexible.
- Weakness: The approach is not applicable to other processes. It depends on specific Itanium support.

Do the results support the conclusions? Explain. (100 words)
Some of the benchmarks show very good improvement. For example mcf shows an improvement of 38.5%. Also the results show that the targeted threads are very few and the helper threads involved are usually 1 or 2. This indicates that there are very few opportunities for helper threads. This is true since helper thread is costly and has to be used only when other prefetching techniques are not applicable. Also the results are shown for real database applications and it shows very good speedup. The results show that the method is very scalable i.e., if we change the L3 cache, or change some cache configuration, as a result of which our cache misses may change, the method still works as the threads are triggered only when L3 miss actually occurs.

Describe the potential future works? (100 words)
This paper shows how effectively the PAL feature can be exploited. So we could use the same mechanism to do other optimizations and all of this will happen transparent of the OS. Since the memory and CPU performance gap is every increasing, we will get more time during a cache miss to do something useful. One difficulty in this approach is to have a efficient helper thread. Different compiler optimizations specific to the helper thread has to be studied. Effectiveness of prefetching depends a lot on how ahead we prefetch the data. Since the loop count in integer programs can be less, we have a challenge of how to prefetch our data much ahead of the loop which actually causing cache misses. The helper thread technique as such may not be applicable, since the data needed to compute the address may not be available at the fork point we are aiming for. So other techniques like value profiling or prediction can be used to predict the address and prefetch.