Summary
Compiler Optimization of Scalar Value Communication between Speculative Threads
Kiran Yellajyosula
February 9, 2005

1 To be completed before class

What are the problems solved by this paper? (50 words)
Thread Level Speculation attempts to run sequential programs on multiple processors with the extra hardware at the cache to ensure the correctness of program execution. The compiler inserts instructions to create threads and forward values between the threads. The authors identified that the critical forwarding path on multi-threaded processors is a performance bottleneck.

What are the approaches attempted by this paper? (50 words)
The authors reduce the forwarding critical path by instruction scheduling based on dataflow analysis for induction variables and other scalar values. They also present an aggressive technique based on speculation to optimize the frequent path and also insert additional code on the infrequent paths for ensuring correctness.

What are the main conclusions of this paper? (50 words)
The authors performed a potential improvement on performance that can be obtained by value prediction. The authors proposed and evaluated the scheduling algorithms and showed that the aggressive code motion improves the performance of some SPECint benchmarks. The authors employed their techniques to some selected regions which are good candidates for the TLS optimization.

2 To be completed after class

Did this paper address an important issue? Explain. (100 words)
This work tries to reduce the critical forwarding path of selected loops by synchronization and instruction scheduling. The wait is placed as late as possible and signal is placed as early as possible to decrease the synchronization time. The paper discusses the performance of conservative compiler analysis and develops a dataflow based scheduling algorithm. The paper further discusses an aggressive scheme based on control and data speculation to decrease the critical path for frequently executed paths. The authors extend their dataflow infrastructure with the profiled data to support their aggressive scheme. Loops of very small or large iteration size are not chosen for speculative execution as there is no sufficient parallelism to exploit in the former and the probability of a failure is high in the latter.

Are the proposed approaches valid? Describe its strength and weakness. (100 words)
The authors propose a scalable technique to parallelize sequential applications on a TLS supporting processor. The strengths of this approach are as follows:

1. The scheme is independent of the underlying architecture
2. The approach is scalable.
3. The compiler can easily identify these scalar variables that are forwarded.
4. The approach can be integrated with other hardware optimizations to achieve better performance.

The weaknesses of the approach are as follows:
1. Data is forwarded through the shared cache between the processors.
2. The time to profile the loops and decide good candidates is long.
3. It requires recompiling the source files of the software.

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

The authors implemented their scheduling algorithm and generated the code for the 4 processor TLS machine used in their simulation. The results depict a significant improvement in the performance of good regions and applying the scheduling technique to all the regions in the benchmarks decreased the performance. For some benchmarks, the reduced synchronization time caused an increase in the failure time as expected. The aggressive scheduling technique improved the performance of gcc, go and m88ksim further.

Describe the potential future works? (100 words)

The paper tries to reduce the synchronization time of the execution time. The failure time of the TLS execution can be reduced by synchronization and selective speculation. Efficient hardware can be developed to reduce the signal and wait without going through the cache. Further, a new hardware paradigm has to be designed to reduce the “other” time (time when instructions are not graduating). Compiler techniques to parallelize the iterations with large dynamic instruction size are to be developed.