Summary
Compiler optimization Scalar value communication between speculative threads

Venkatesan Packirisamy
February 9, 2005

1 To be completed before class

What are the problems solved by this paper? (50 words)
Thread level speculation (TLS) has been successfully used to parallelize programs that cannot be parallelized by conventional parallelization techniques. One of the major overhead in TLS execution is the frequent squashes occurring due to mis-speculation. There are some dependences that can be easily predicted. If we synchronize on these dependences, we can easily reduce the number of mis-speculations. One problem with synchronization is it sometimes can serialize execution. This paper tries to reduce the number of mis-speculations and at the same time reduce the effect of synchrononization (by reducing the critical forwarding path).

What are the approaches attempted by this paper? (50 words)
The paper uses compiler dataflow algorithms to identify the scalar variables that needs to be synchronized. Then it inserts signal and wait to implement the synchronization. This will prevent the mis-speculation caused due to the dependence. But it can effectively serialize the execution. So the paper also tries to push the wait instruction downwards in the program. Similarly the signal is pushed up. Along with pushing the wait and signal, the instructions are also scheduled to reduce the critical forward path. Also aggressive scheduling is done using control speculation and data speculation. The resulting code will prevent frequent mis-speculations without serializing the execution.

What are the main conclusions of this paper? (50 words)
The compiler techniques implemented improve the performance by 6.2-28.5% in many applications. The effect of hardware techniques was also studied and compared with the compiler technique. It is showed that the compiler techniques can outperform hardware techniques. Also good performance is showed when hardware and compiler techniques are used together. The study shows that compiler scheduling is very important to remove the bottlenecks in TLS.

2 To be completed after class

Did this paper address an important issue? Explain. (100 words)
Though multi-core architectures have become common, they are not used to increase performance of a single program. The main reason is that a sequential program is very difficult to parallelize. To parallelize such programs, thread level speculation was introduced. But even with this speculation, the performance gained by the sequential program is very less. The main reason is that there are overheads on the speculative execution. One such major overhead is the frequent squashes due to data dependences. One was to reduce is to introduce synchronization. If a load-store pair is causing frequent dependences, the dependence synchronized. Now there will not be any squashes due to dependence violations. This can be done both in hardware and software. But due to these synchronizations, sometimes the code gets serialized. To avoid this the paper schedules the instructions in such a way
that the overlap between the threads is increased. Also to increase the overlap the paper uses aggressive scheduling mechanisms with control and data speculation. So the paper addresses the issue of frequent squashes and also it reduces the serial effect due to synchronizations.

**Are the proposed approaches valid? Describe its strength and weakness. (100 words)**
The approach used in the paper to reduce frequent squashes is to synchronize if we are sure that the dependence is going to occur. But inefficient synchronization can cause the code to be effectively serial. The paper addresses this issue by using code scheduling. But sometimes it is very difficult to schedule the code due to branches and pointer data accesses. So this paper uses aggressive scheduling by using control and data speculation.

- **Strength:** The major strength of the approach is that it needs very little hardware support. Also the approach was shown to identify data dependences that are not identified by hardware.
- **Weakness:** The approach is only for scalar variables. So it has to be supplemented with hardware to detect dependences due to pointer dependences.

**Do the results support the conclusions? Explain. (100 words)**
First the paper shows the need to reduce the critical forwarding path. The graph showed the importance of synchronization and scheduling. Even though handling induction variables gives the largest improvement in some benchmarks, the scheduling also leads to lot of performance. Though not much is improved due to control speculation, some benchmarks show good improvement after applying data speculation. Also it is shown that the compiler technique can complement the hardware technique since they both tackle different dependences in many cases. Also when compared to the multiscalar algorithm, the current algorithm shows some improvement. The improvement in the overall performance of the benchmarks shows the importance of the synchronization and scheduling. Also it shows that the proposed technique gives very good performance.

**Describe the potential future works? (100 words)**
The paper only handles scalar variables. The next thing is to apply the same technique to irregular pointer dependences (this is what is done in the next paper). The paper looked only one aspect of the overhead of speculative execution. The other overheads like load balancing, etc can also benefit from some compiler support. Also for some dependencies which are not very frequent to be synchronized and not that less frequent to be allowed to speculate, we can generate recovery code. So we always speculate on such dependences, and when that dependence occurs, then instead of squashing the thread we can execute the recovery code. There may some dependence whose behavior will vary at runtime. So we could think of some runtime code optimizations techniques for such dependencies.