Lecture 29(a)
Intro to Thread-Level Speculation

Automatic Parallelization

Proving independence of threads is hard:
- complex control flow
- complex data structures
- pointers, pointers, pointers
- run-time inputs

How can we make the compiler's job feasible?

Example of Thread-Level Speculation

```
while (...){
    x = hash[index1];
    ...
    hash[index2] = y;
    ...
}
```
Example of Thread-Level Speculation

Epoch 1
hash[3]
hash[10]
commit?

Epoch 2
hash[19]
hash[21]
hash[30]
commit?

Epoch 3
hash[33]
hash[25]
commit?

Epoch 4
hash[10]
hash[25]
commit?

Time

Overview of Our Approach

System requirements:
1) Detect data dependence violations
   • extend invalidation-based cache coherence
2) Buffer speculative modifications
   • use the caches as speculative buffers
**Life Cycle of an Epoch**

- **Time**
  - Init
  - Spawned
  - Becomes Speculative
  - Speculative Work
  - Commit?
  - Complete, Pass Homefree?
  - Slow Commit
  - Becomes Speculative
  - Fast Commit

**Simulation Infrastructure**

- Compiler system and tools based on SUIF
  - help analyze dependences, insert synchronization
  - produce MIPS binaries containing TLS primitives
- **Benchmarks (all run to completion)**
  - buk, compress95, ijpeg, equake
- **Simulator**
  - superscalar, similar to MIPS R10K
  - models all bandwidth and contention

- Detailed simulation!

**Performance on a 4-Processor CMP**

- **Speedup**
  - buk: 2.26
  - compress95: 1.27
  - equake: 1.77
  - ijpeg: 1.94

- **Parallel Coverage**
  - buk: 56.6%
  - compress95: 47.3%
  - equake: 39.3%
  - ijpeg: 22.1%

- **Program speedups are limited by coverage**
Varying the Number of Processors

- **buk and equake** are memory-bound
- **compress95 and jpeg** are computation-intensive

Scaling Beyond Chip Boundaries

- simulate architectures with 1, 2 and 4 nodes
- multi-chip systems benefit from TLS
Scaling Beyond Chip Boundaries

Conclusions

The overheads of our scheme are low:
- mechanisms to squash or commit are not a bottleneck
- per-word speculative state is not always necessary

It offers compelling performance improvements:
- program speedups from 8% to 46% on a 4-processor CMP
- program speedups up to 75% on multi-chip architectures

It is scalable:
- coherence provides elegant data dependence tracking

seamless TLS on a wide range of architectures