Lecture 28(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?

Thread-Level Speculation (TLS)

Example

```
while (...) {
    x = hash[index1];
    ...
    hash[index2] = y;
    ...
}
```

Example of Thread-Level Speculation

```
Epoch 1
- hash[3]
  ...
  hash[10] = ...
  ...
  hash[19] = ...
  ...
  hash[21] = ...
  ...
  hash[33] = ...
  ...
  hash[30] = ...
  ...
  hash[25] = ...
Epoch 2
- hash[10] = ...
  ...
  hash[30] = ...
  ...
  hash[25] = ...
Epoch 3
- hash[33] = ...
  ...
  hash[30] = ...
  ...
  hash[25] = ...
Epoch 4
- hash[10] = ...
  ...
  hash[30] = ...
  ...
  hash[25] = ...
```
Example of Thread-Level Speculation

Time

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

Epoch 2
- hash[19]
  - …
Violation

Epoch 3
- hash[33]
  - hash[21]
  - …
commit?

Epoch 4
- hash[10]
  - …
commit?

Example of Thread-Level Speculation

Time

Epoch 1
- hash[3]
  - hash[10]
  - …
Violation
  - …
commit?

Epoch 2
- hash[19]
  - …
commit?

Epoch 3
- hash[33]
  - hash[21]
  - …
commit?

Epoch 4
- hash[10]
  - …
commit?

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

Carnegie Mellon
Life Cycle of an Epoch

- Spawned
- Init
- Becomes Speculative
- Speeds Speculative
- Commit?
- Slow Commit
- Fast Commit
- Complete, Pass Homefree
- Wait to be Homefree?
- Speculative Work

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

Performance on a 4-Processor CMP

- buk
- compress95
- equake
- ijpeg

Regional speedups are limited by coverage
Varying the Number of Processors

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

Scaling Beyond Chip Boundaries

- simulating architectures with 1, 2 and 4 nodes
- multi-chip systems benefit from TLS

Notes:
- *buk* and *equake* scale well
- passing the homefree token is not a bottleneck
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