Lecture 28(a)

Intro to Thread-Level Speculation

Automatic Parallelization

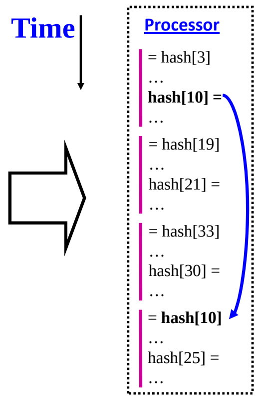
Proving independence of threads is hard:
- complex control flow
- complex data structures
- pointers, pointers, pointers
- run-time inputs

How can we make the compiler's job feasible?

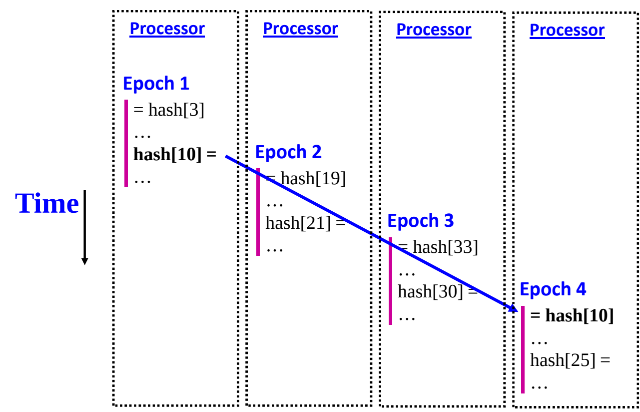
☞ **Thread-Level Speculation (TLS)**

Example

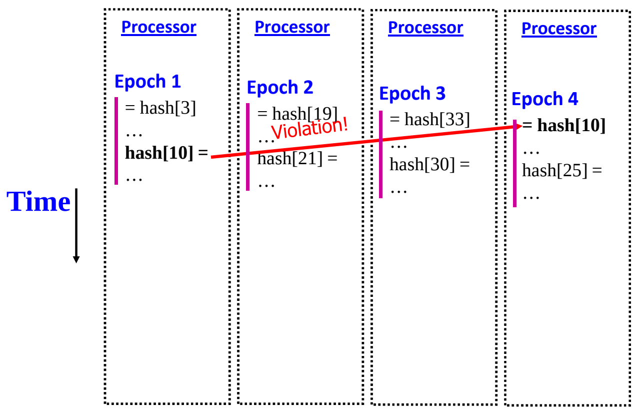
```
while (...){
    x = hash[index1];
    ...
    hash[index2] = y;
    ...
}
```



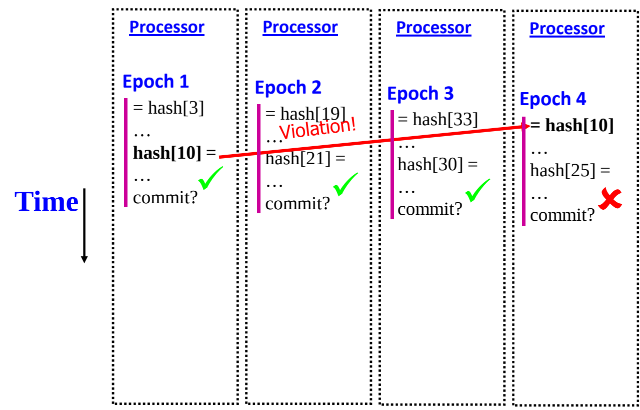
Example of Thread-Level Speculation



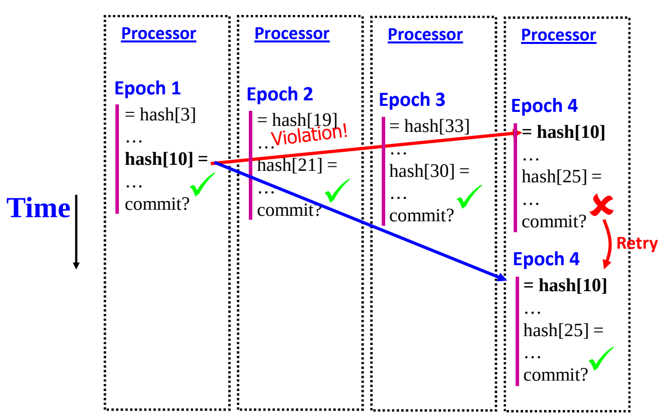
Example of Thread-Level Speculation



Example of Thread-Level Speculation



Example of Thread-Level Speculation

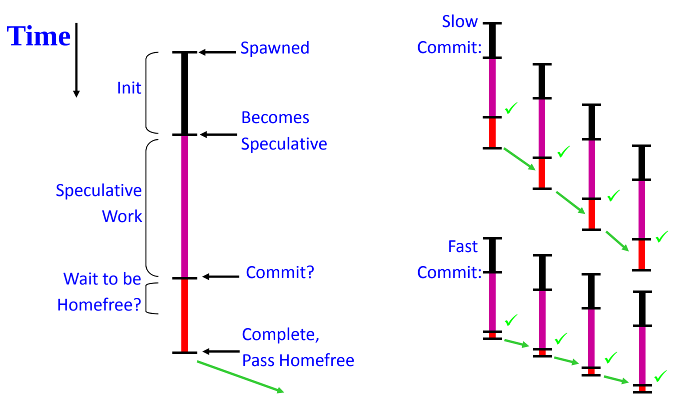


Overview of Our Approach

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

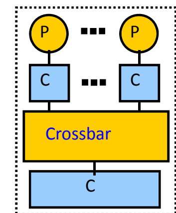


Simulation Infrastructure

Compiler system and tools based on SUIF
- help analyze dependences, insert synchronization
- produce **MIPS** binaries containing TLS primitives

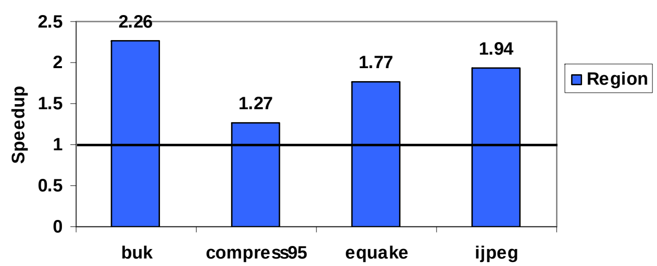
Benchmarks (all run to completion)
- buk, compress95, ijpeg, equake

Simulator
- superscalar, similar to **MIPS R10K**
- models all bandwidth and contention



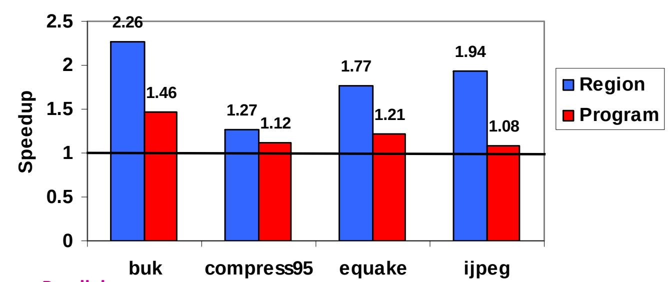
☞ **detailed simulation!**

Performance on a 4-Processor CMP



Parallel Coverage:	56.6%	47.3%	39.3%	22.1%

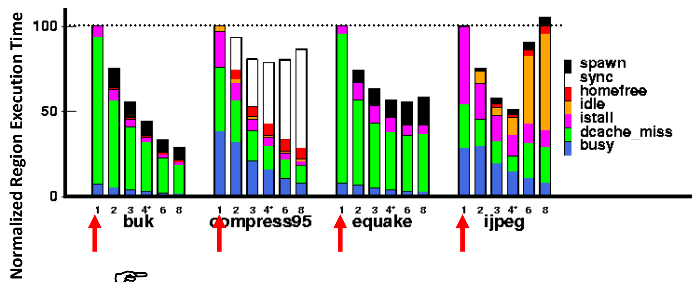
Performance on a 4-Processor CMP



Parallel Coverage:	56.6%	47.3%	39.3%	22.1%

☞ **program speedups are limited by coverage**

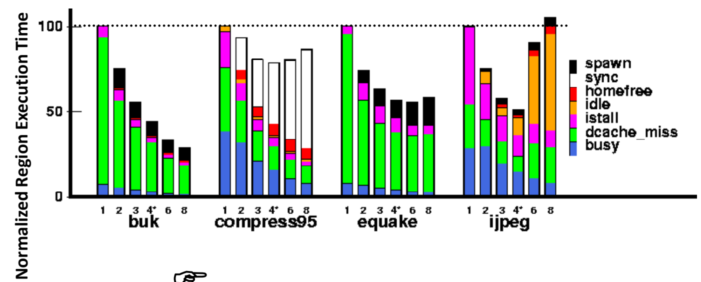
Varying the Number of Processors



☞ buk and equake are memory-bound

☞ compress95 and ijpeg are computation-intensive

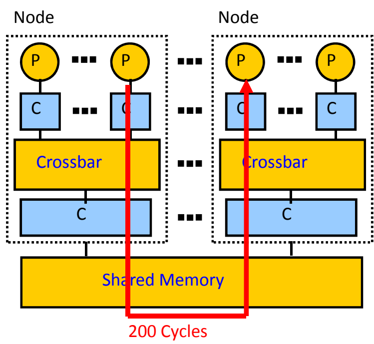
Varying the Number of Processors



☞ buk and equake scale well

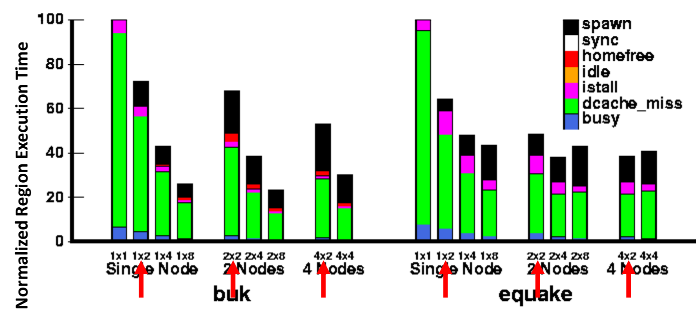
☞ passing the homefree token is not a bottleneck

Scaling Beyond Chip Boundaries



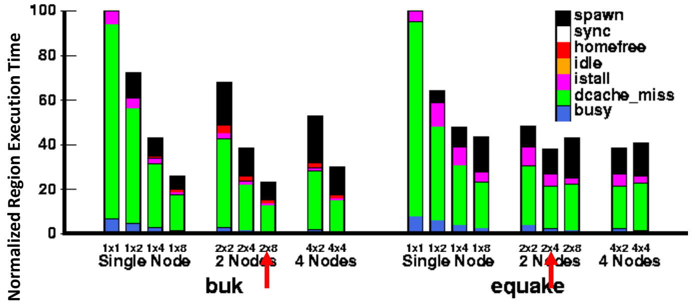
☞ simulate architectures with 1, 2 and 4 nodes

Scaling Beyond Chip Boundaries



☞ multi-chip systems benefit from TLS

Scaling Beyond Chip Boundaries



☞ **our scheme scales well**

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**