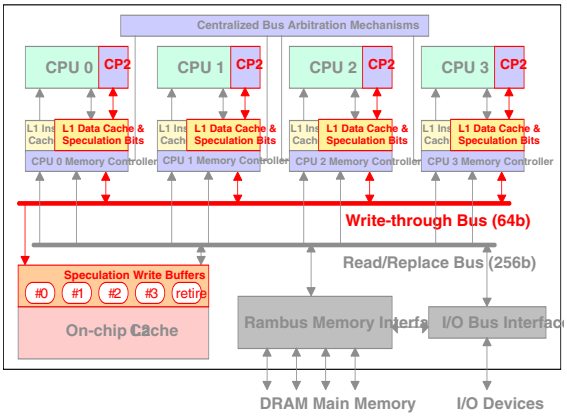
18-742

Lecture 24

Speculative Threading on CMPs



Spring 2005

Prof. Babak Falsafi

http://www.ece.cmu.edu/~ece742

Slides developed in part by Prof. Falsafi from Hill, Olukotun, Oplinger and Stets of Carnegie Mellon University, Google, Stanford University, and University of Wisconsin.

What is so hard about memory disambiguation across cores?

Think private writeback L1s and a shared L2

- **Need coherence among L1s**

But, also need SES memory dependence order

How do we change the coherence protocol to implement SES dependence order?

Requirements (changes to CMP):

- **Common case of cache hit should go fast**
- **Cache misses should not take much longer than in a CMP**
- **No sequential searches in caches upon thread invocation, completion or rollback**
- **Coherence protocol needs tens of states (see SVC by Gopal et al.)**

Stanford Hydra

Made the problem much simpler

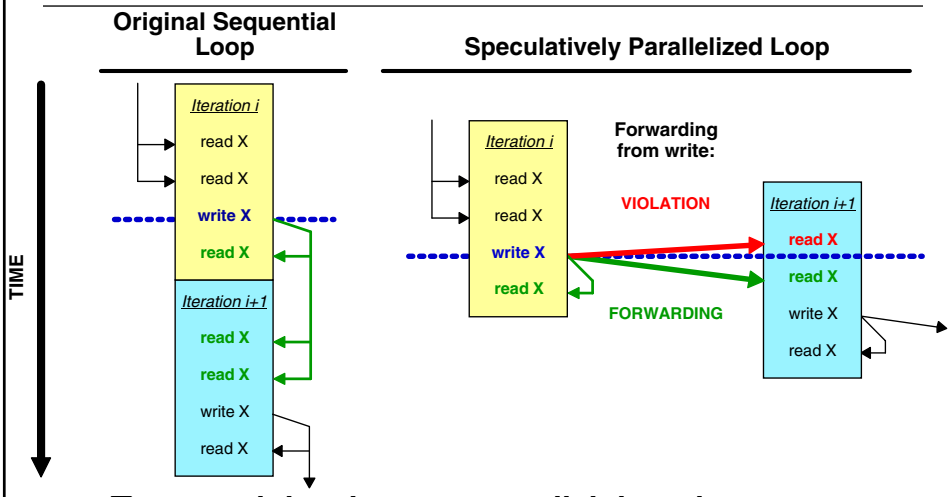
- At a performance cost?

No need for register communication

- Communicate register values through memory
- As in conventional register load/spills

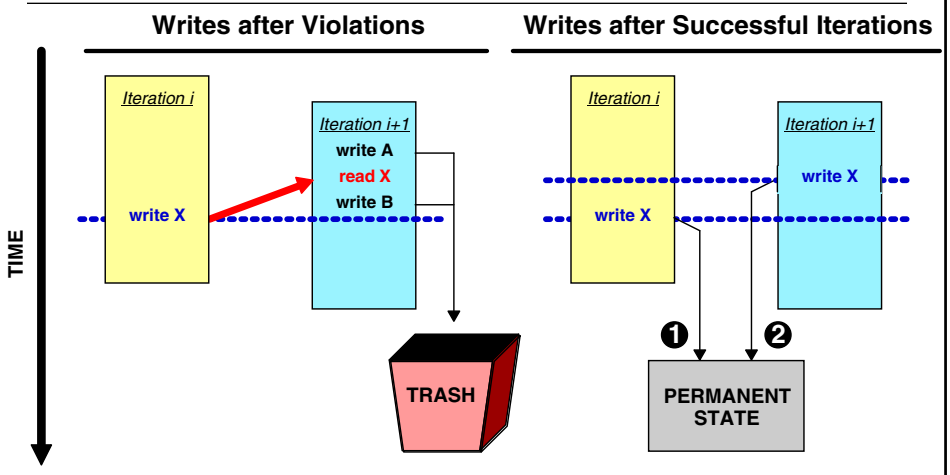
Make the L1 caches writethrough

Data Speculation in Hydra: Requirements (I)



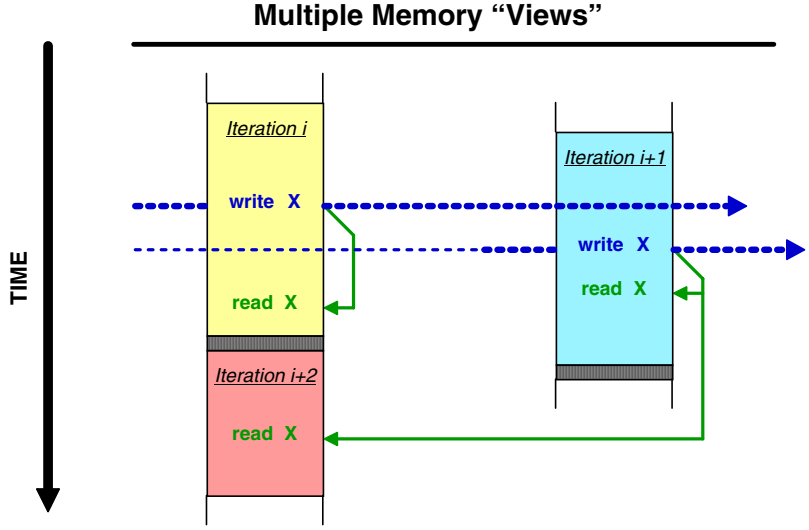
- ❑ Forward data between parallel threads
- ❑ Detect violations when reads occur too early

Data Speculation in Hydra: Requirements (II)



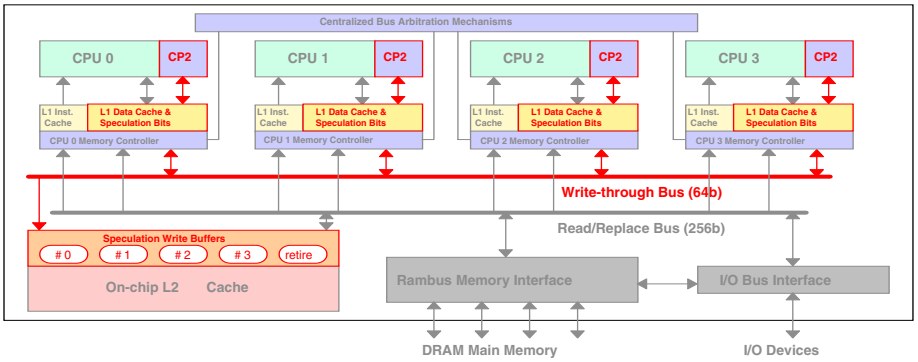
- Safely discard bad state after violation
- Correctly retire speculative state

Data Speculation in Hydra: Requirements (III)



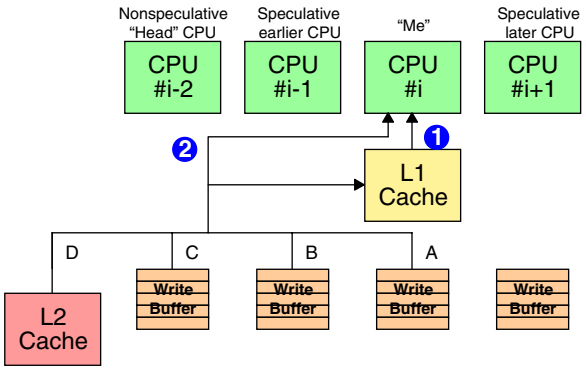
- Maintain multiple "views" of memory

Hydra Speculation Support



- Write bus & L2 buffers forward
- "Read" L1 tags for violations, "Dirty" L1 tags and wbuff provide backup
- Wbuff reorder & retire spec. state
- Speculation coprocessors to control threads

Speculative Reads



L1 hit

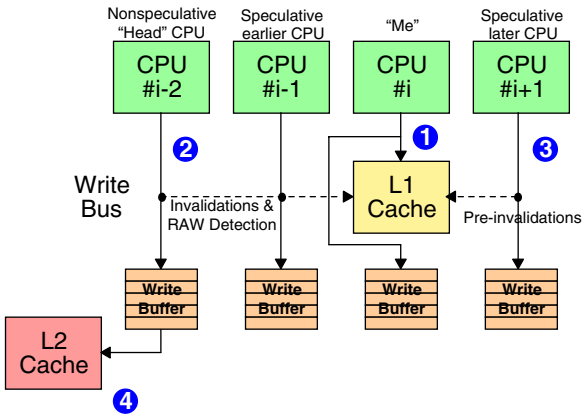
Read bits are set

L1 miss

L2 and write buffers are checked in parallel

Latest written values from a cache block are pulled in by priority encoders on each byte (priority A-D)

Speculative Writes

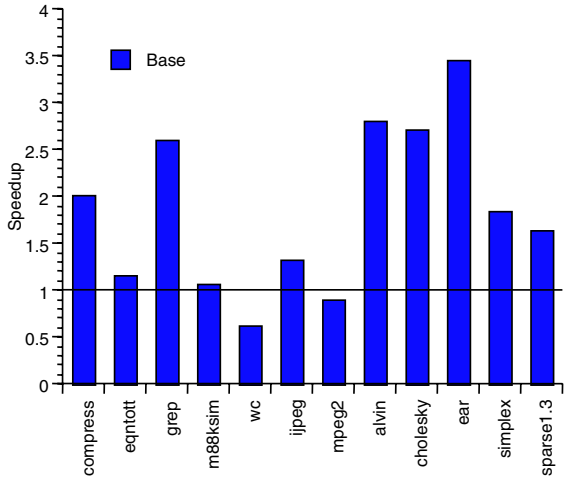


- ❑ A CPU writes to its L1 cache & write buffer
- ❑ "Earlier" CPUs invalidate our L1 & cause RAW hazard checks
- ❑ "Later" CPUs just pre-invalidate our L1
- ❑ Non-speculative write buffer drains out into the L2

Creating Speculative Threads

- **Speculative loops**
  - `for` and `while` loop iterations
  - Typically one speculative thread per iteration
- **Speculative procedures**
  - Execute code after procedure speculatively
  - Procedure calls generate a speculative thread
- **Compiler support**
  - `C` source to source translator
  - `Pfor, pwhile`
  - Analyze loop body and globalize any local variables that could cause loop-carried dependencies

Base Speculative Thread Performance

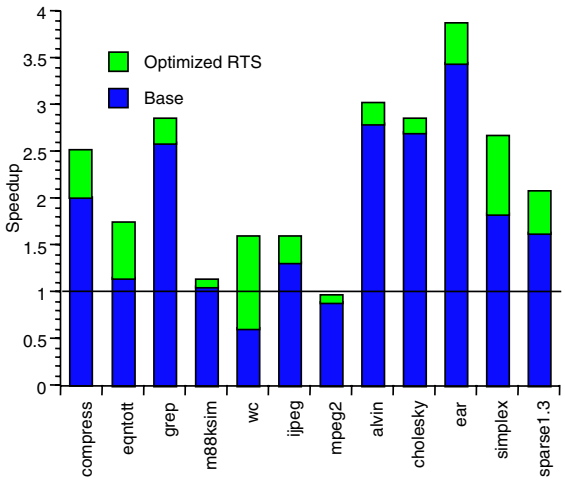


- Entire applications
- GCC 2.7.2 -O2
- 4 single-issue processors
- Accurate modeling of all aspects of Hydra architecture and real runtime system

Improving Speculative Runtime System

- **Procedure support adds overhead to loops**
  - Threads are not created sequentially
  - Dynamic thread scheduling necessary
  - Start and end of loop: 75 cycles
  - End of iteration: 80 cycles
- **Performance**
  - Best performing speculative applications use loops
  - Procedure speculation often lowers performance
  - Need to optimize RTS for common case
- **Lower speculative overheads**
  - Start and end of loop: 25 cycles
  - End of iteration: 12 cycles (almost a factor of 7)
  - Limit procedure speculation to specific procedures

Improved Speculative Performance



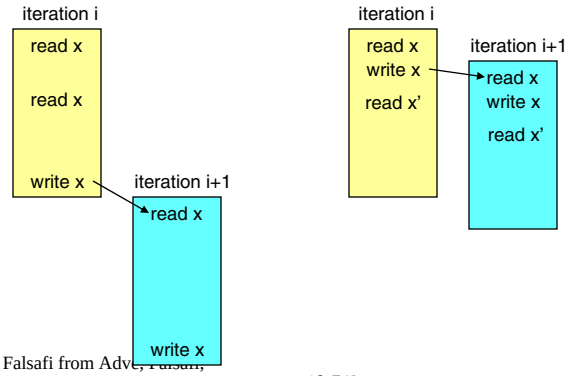
- Improves performance of all applications
- Most improvement for applications with fine-grained threads
- Eqntott uses procedure speculation

Feedback and Code Transformations

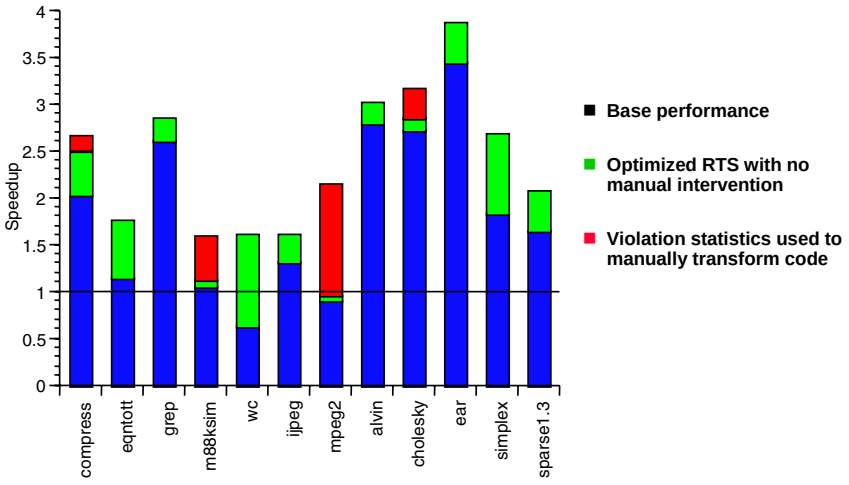
- **Feedback tool**
  - Collects violation statistics (PCs, frequency, work lost)
  - Correlates read and write PC values with source code
- **Synchronization**
  - Synchronize frequently occurring violations
  - Use non-violating loads
- **Code Motion**
  - Find dependent load-stores
  - Move loads down in thread
  - Move stores up in thread

Code Motion

- Rearrange reads and writes to increase parallelism
- Delay reads and advance writes
- Create local copies to allow earlier data forwarding



Optimized Speculative Performance



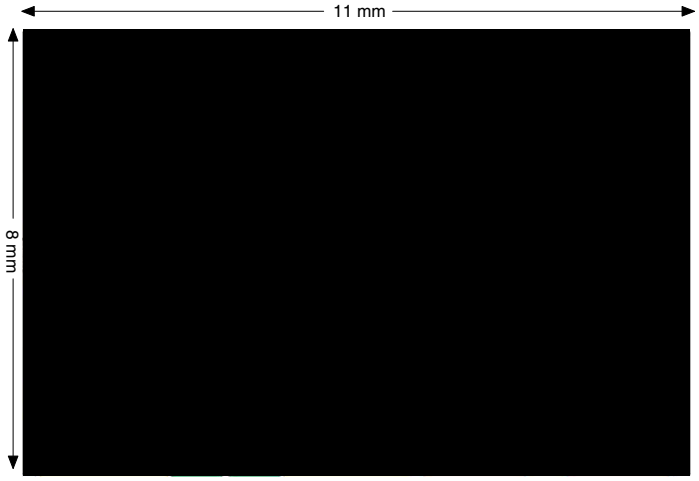
Size of Speculative Write State

- **Max size determines size of write buffer for max performance**
- **Non-head processor stalls when write buffer fills up**
- **Small write buffers (< 64 lines) will achieve good performance**

Max no. lines of write state	
compress	24
eqntott	40
grep	11
m88ksim	28
wc	8
ijpeg	32
mpeg	56
alvin	158
cholesky	4
ear	82
simplex	14

– 32 byte cache lines

Hydra Prototype



- **Design based on Integrated Device Technology (IDT) RC32364**
- **88 mm² in 0.25µm with 8 KB I, D and 128 KB L2**