Using Speculative Retirement and Larger Instruction Windows to Narrow the Performance Gap between Memory Consistency Models

Parthasarathy Ranganathan
Vijay S. Pai
Sarita V. Adve

Department of Electrical and Computer Engineering
Rice University

`http://www-ece.rice.edu/~rsim/`

Motivation (1 of 2)

Shared memory systems are increasingly popular

Performance highly dependent on memory latency

Memory consistency model determines ability to hide latency

Tradeoff: programmability vs. performance

SC $\rightarrow$ PC $\rightarrow$ RC

Motivation (2 of 2)

Current processors agressively exploit instruction-level parallelism (ILP)

e.g. non-blocking loads, out-of-order issue, multiple issue

Allow optimized consistency implementations

BUT RC still higher performance [Pai et al. '96]

Store-to-load constraint impedes SC

Contributions (1 of 2)

Target store-to-load constraint of SC

- Speculative retirement
  - Within 11% of PC on all applications
  - Improves performance without sacrificing programmability

- vs. Larger instruction windows
  - Results depend on base window size

Contributions (2 of 2)

Optimized implementations of PC

- Significantly improve PC performance
- Release-consistency (RC) remains higher performance

Overview of Talk

Background

Speculative Retiremenet

Simulation Methodology

Performance Evaluation

Conclusions

Consistency Models

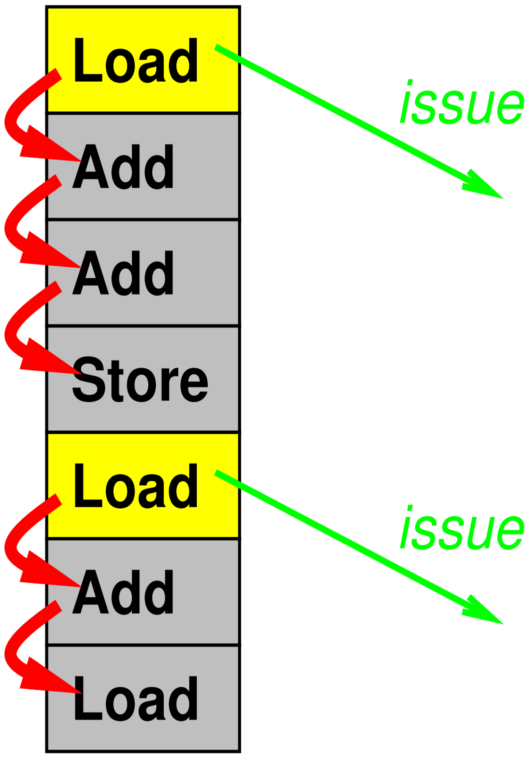
Flexibility to overlap or reorder memory operations

SC: All memory operations appear in program order

PC: Loads can bypass/overlap previous stores

RC: Freely overlap/reorder data;
must distinguish synchronization

Aggressive Microarchitectures



Issue out-of-order

Non-blocking loads

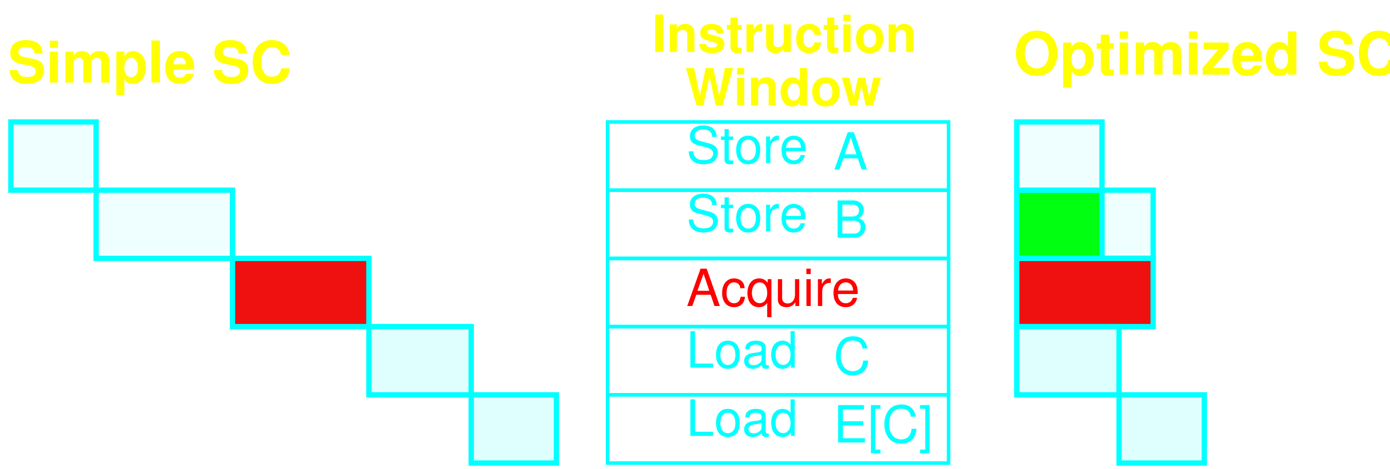
Instructions issued in parallel

Do strong consistency models limit aggressive processors?

Optimized Implementations

Issue prefetches for blocked stores

Use load values out-of-order (speculatively)



Load values are wrong if overwritten

- Check for invalidates/updates
- Rollback instructions on violation

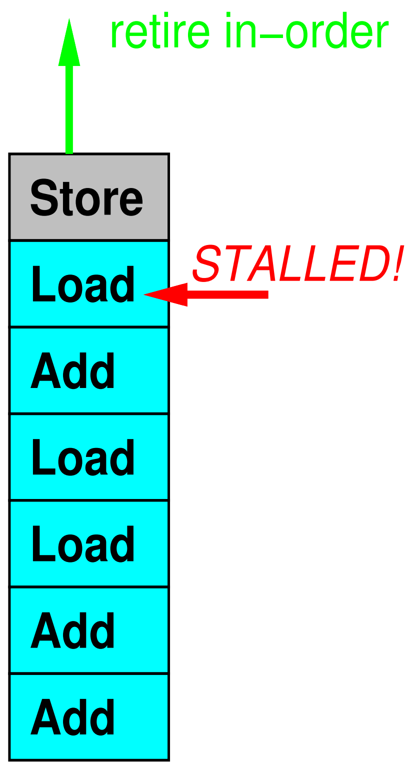
Results with Current Optimizations [Pai et al. ASPLOS '96]

SC performance dramatically improves

BUT significant performance gap remains

- Due to unoverlapped store latency

In-order Retirement



For precise interrupts

- Commit architectural state in-order
- Retire from window in-order

  Commit on retire

In SC,

Loads can't retire until previous stores complete

Speculative Retirement

Targets store-to-load constraint of SC

Lets loads retire even if previous stores incomplete

- Commit values (speculatively)
- Roll back in case of violation

How to save/restore processor state?

The History Buffer

Mechanism proposed for precise interrupts [SmithPleszkun '85]

Instr.	Reg. #	Old Value
load	24	−1
add	17	3
mult	9	5

Triggered when load speculatively retires

- Save old register value before writing new value
- On violation, restore old values from buffer

The History Buffer (cont'd)

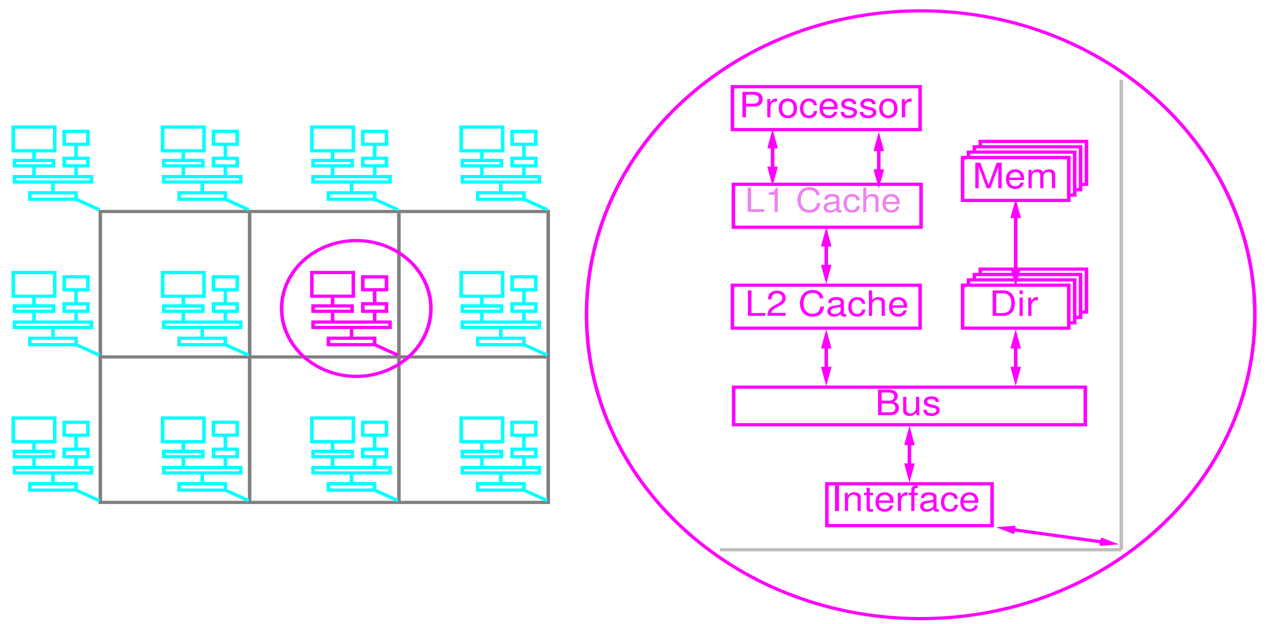
No entry needed if no register value written

E.g. stores, branches

May need additional register ports

- Needed only while speculatively retiring
- Can be multiplexed

Simulated System



System-size: 8-16 nodes

Invalidate-based directory protocol

Simulator and Applications

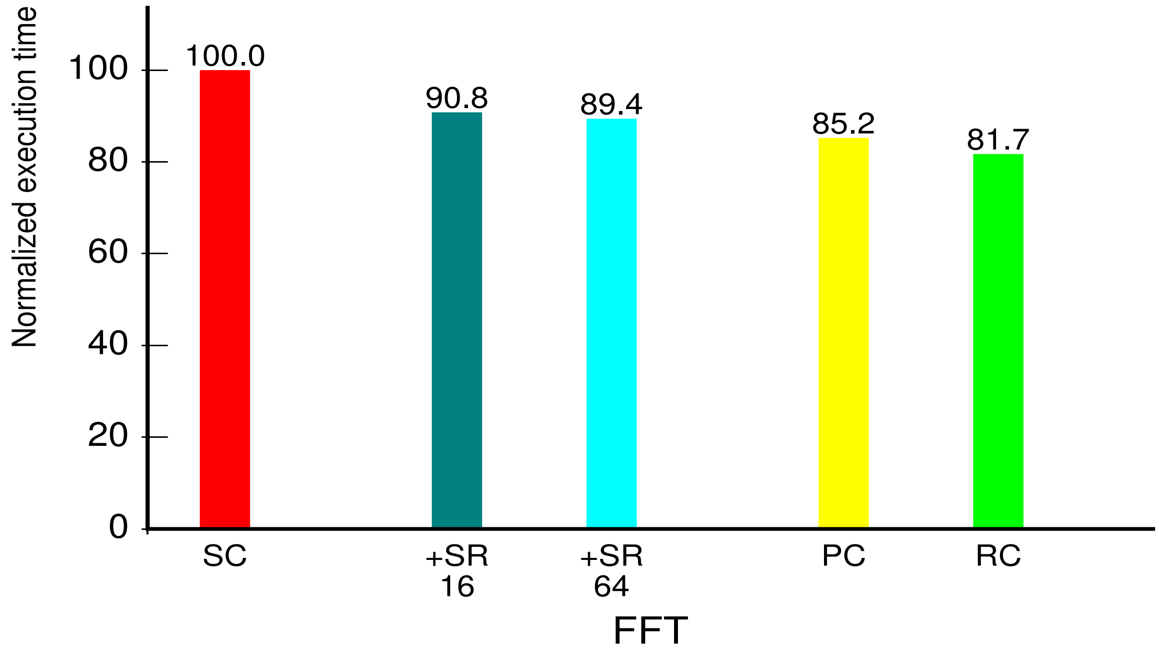
RSIM – Rice Simulator for ILP Multiprocessors

- Detailed execution-driven simulator

Applications

- LU, FFT, Radix (SPLASH-2)
- Mp3d, Water (SPLASH)
- Erlebacher (Rice parallel compiler group)

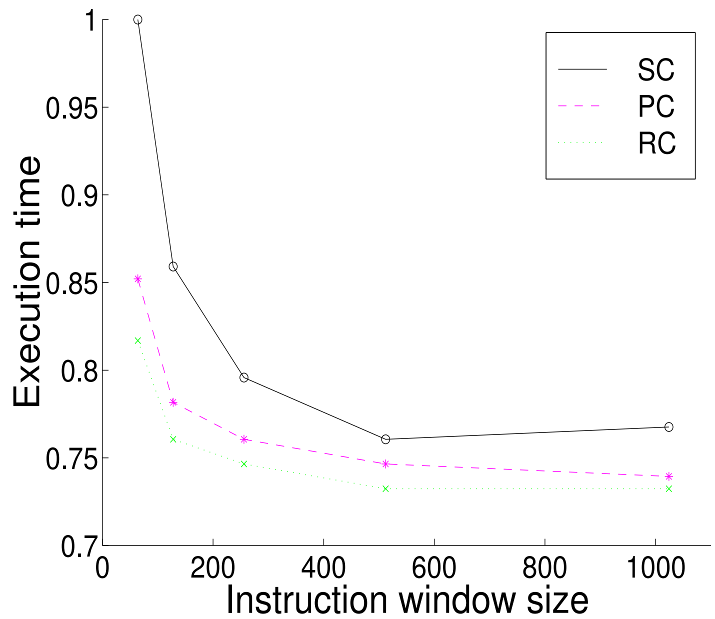
Effectiveness of Speculative Retirement



With speculative retirement,

- SC-PC difference $<$ 11% for all applications
- Performance w/o sacrificing programmability

Larger Instruction Window Sizes



FFT

Increases overlap for both stores and loads

RC reaches “knee” when load overlap saturated

SC may still benefit even when RC has reached knee

Larger Instruction Windows (IW) vs. Speculative Retirement (SR)

Before RC knee

IW	SR
More load overlap	Fewer early prefetches No entry for branch, store

Tradeoff application-dependent

At or beyond RC knee

- SR always comparable or better

SR vs. IW: Complexity

IW	SR
Larger register file	Additional register ports, but can be multiplexed

Same amount of storage required

Conclusions

Targetting store-to-load constraint of SC

- Speculative retirement
  - Within 11% of PC on all applications
  - Performance w/o sacrificing programmability

- vs. Larger instruction windows
  - Mixed results before RC knee
  - SR always comparable or better after RC knee
  - SR preferable for complexity?