What Does the Future Hold for Parallel Languages

A Computer Architect's Perspective

Josep Torrellas

University of Illinois

http://iacoma.cs.uiuc.edu

October 2003

The Future (0-10 years)

Parallel architectures will become more complex:

1. Processing In Memory (PIM)
2. Thead-Level Speculation (TLS)
3. "Undo/redo" capabilities for debugging

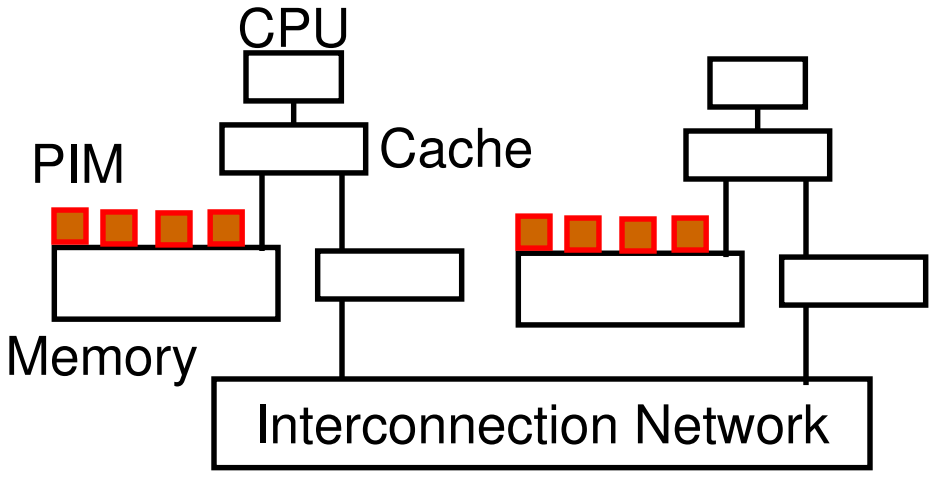
What are the language implications?

Processing In Memory (PIM)

- What? Weak processors close to the memory
- Language implications:
  - Map memory-intensive parts of the code on the PIMs [PPoPP'03]
    - Execute what parts of the computation where
    - How to re-arrange the data structures where
    - How to synchronize, maintain data coherence

PIM (Continuation)

- Language implications:
  - Off-load operations:
    - Cache coherence management
    - Active messages
    - Synchronization/reduction
    - Scatter/gather
    - Logging
    - Vector streaming
    - Checkpointing
    - Bit operations



Thead-Level Speculation (TLS)

- What? Take hard-to-analyze non-numerical codes and compiler parallelizes them into 2-4-6 threads (in a Chip Multiprocessor)
- Language implications:
  - Hints to the compiler, e.g.:
    - This section has few dependences: parallelize speculatively
    - This section has too many dependences
    - This is the best way to break the program into tasks
    - Do not spawn this task any earlier than here

"Undo/Redo" Capabilities for Debugging

- What? Hardware can undo section of execution with minimal overhead and deterministically re-execute it.
- Example of use: If an event occurs (e.g. found a bug), undo, fall into the debugger, ready to replay at user's command

- Language implications:
  - Directive to hardware: "be ready to undo"
    - Entering a section with likely bugs
    - Entering a section with likely data races
    - Entering a section with likely parallelization mistakes

"Undo/Redo" (Cont)

- Language implications:
  - Watch for any access to this address
  - Every access to this address, run this check
    - If check is not ok, fall into the debugger in that instruction. Examples:

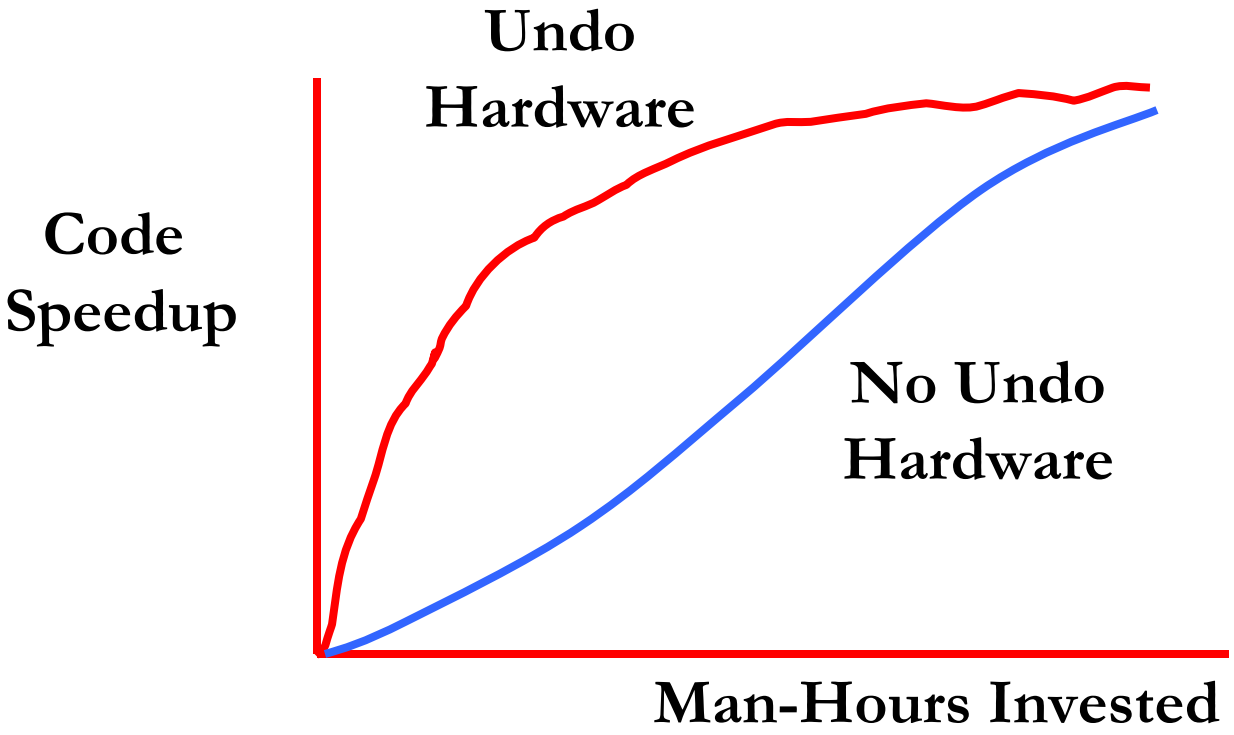
      Memory corruption

      Memory leak

      Invariant check

      Buffer overflow…

Parallelize Programs with a Fraction of the Effort



What Does the Future Hold for Parallel Languages

LCPC03 Panel

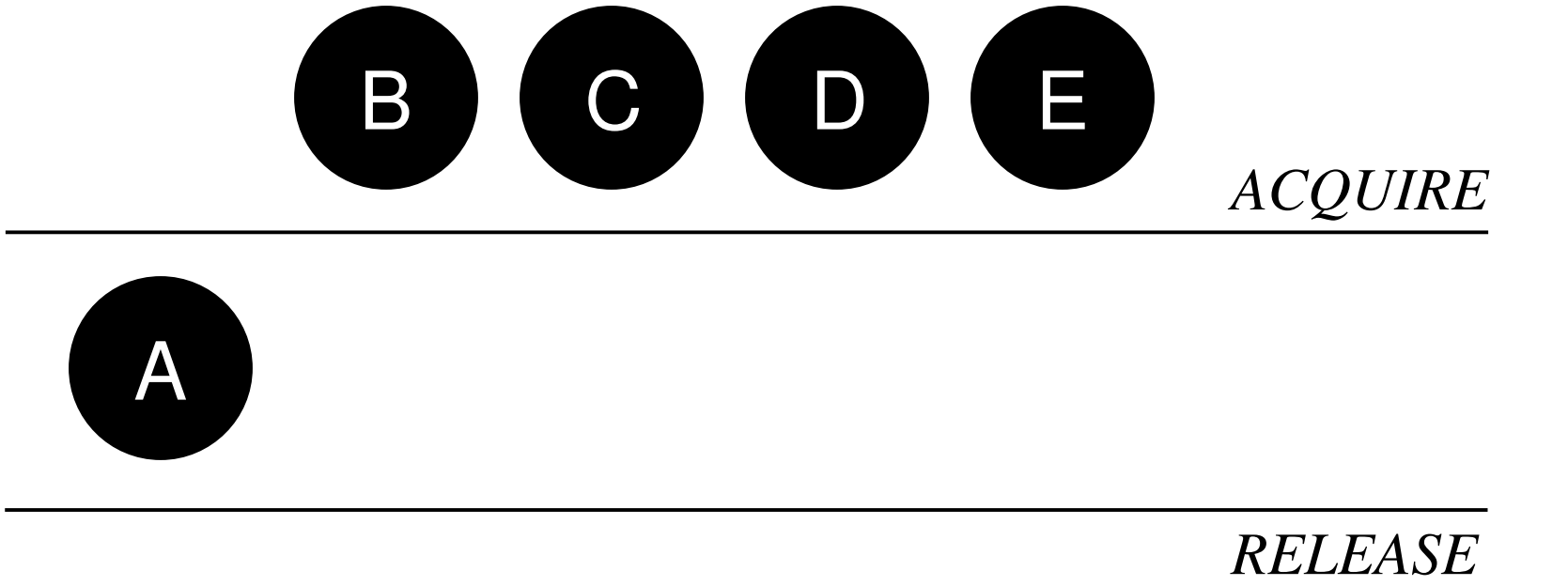
Josep Torrellas

University of Illinois

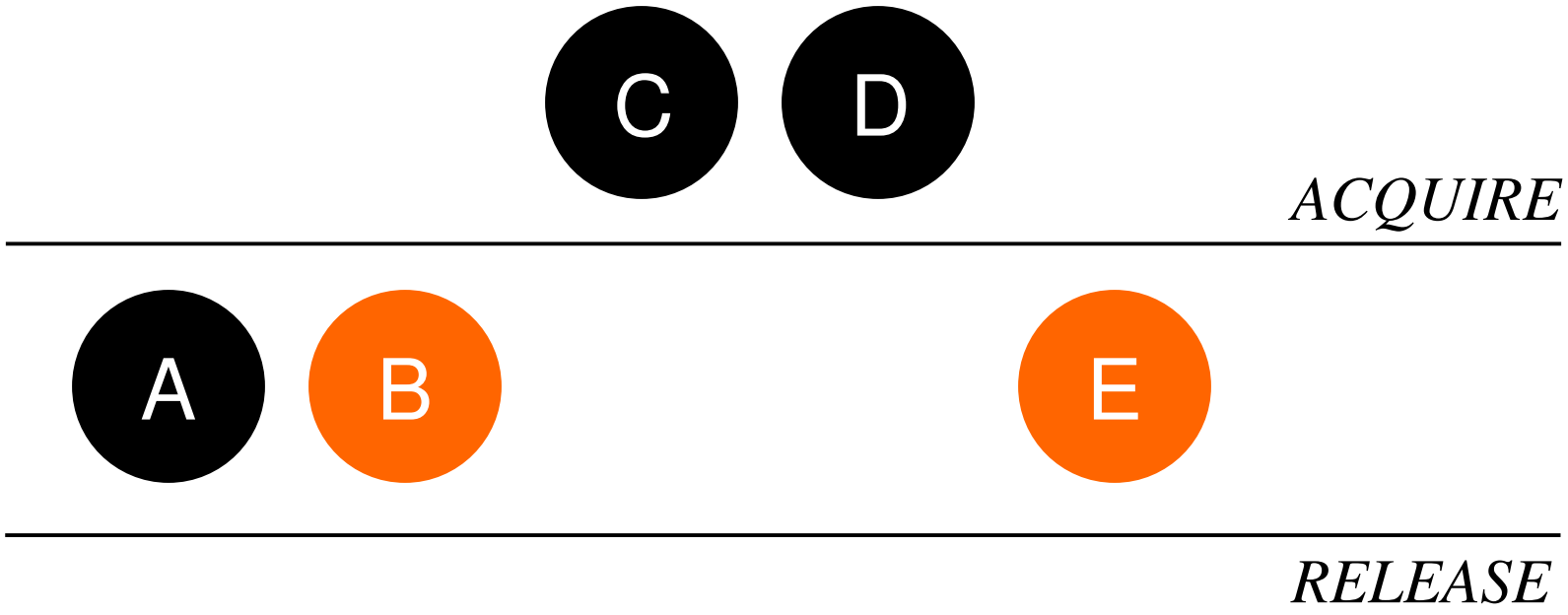
http://iacoma.cs.uiuc.edu

October 2003

Speculative Lock

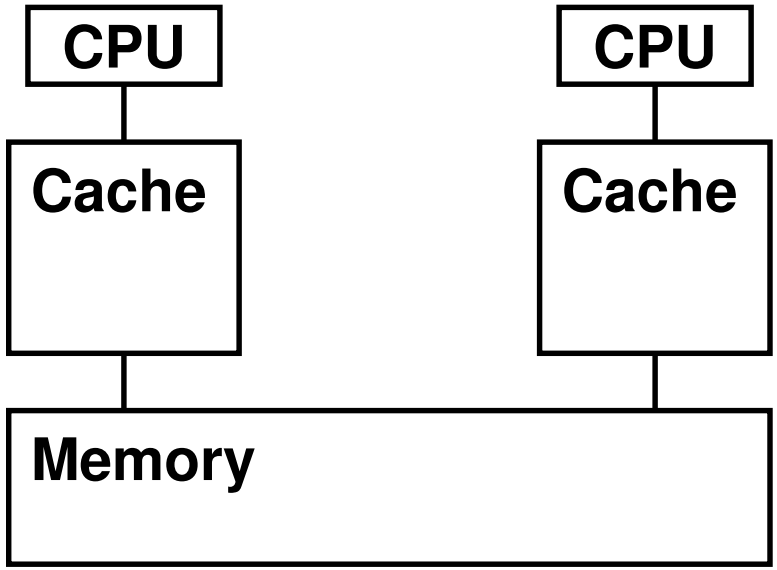


Speculative Lock



Primitive: Speculative Multithreading

- Buffer task state before committing (caches)
- Cleanly undo group of tasks (window of buggy code)
- Re-execute those tasks only
- Re-execution of tasks is deterministic even under multithreading



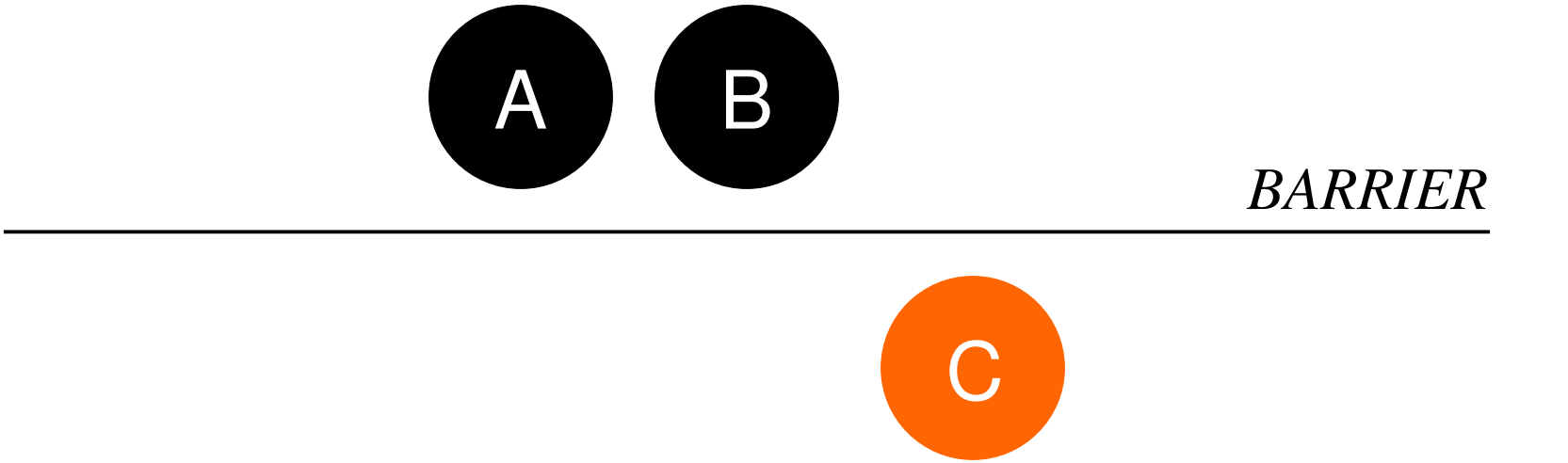
Coarse Synch: Speculative Barrier [ASPLOS02]

BARRIER

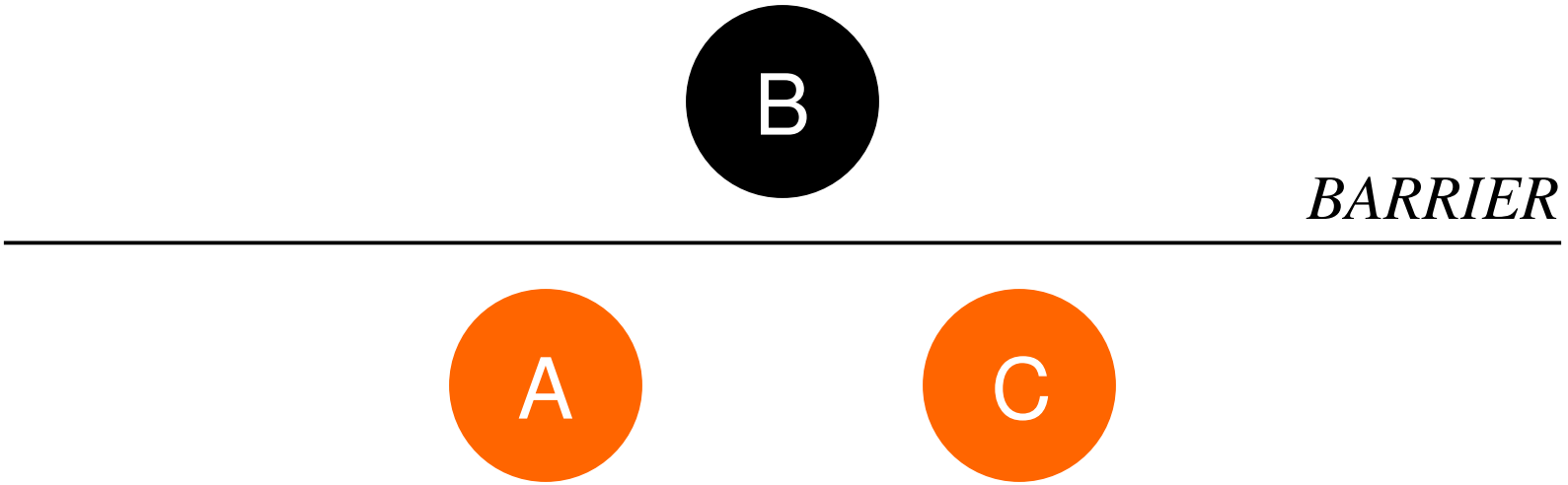
Safe

Speculative

Speculative Barrier



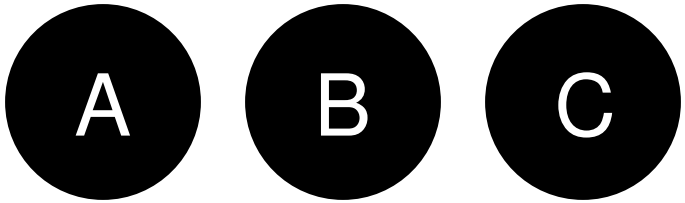
Speculative Barrier



Safe
Speculative

Speculative Barrier

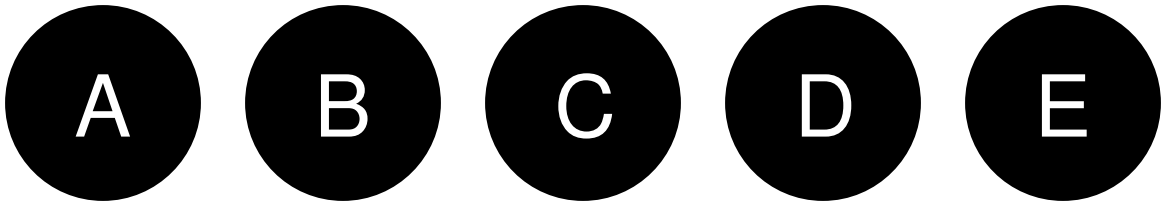
BARRIER



Safe

Speculative

Speculative Lock [ASPLOS02]



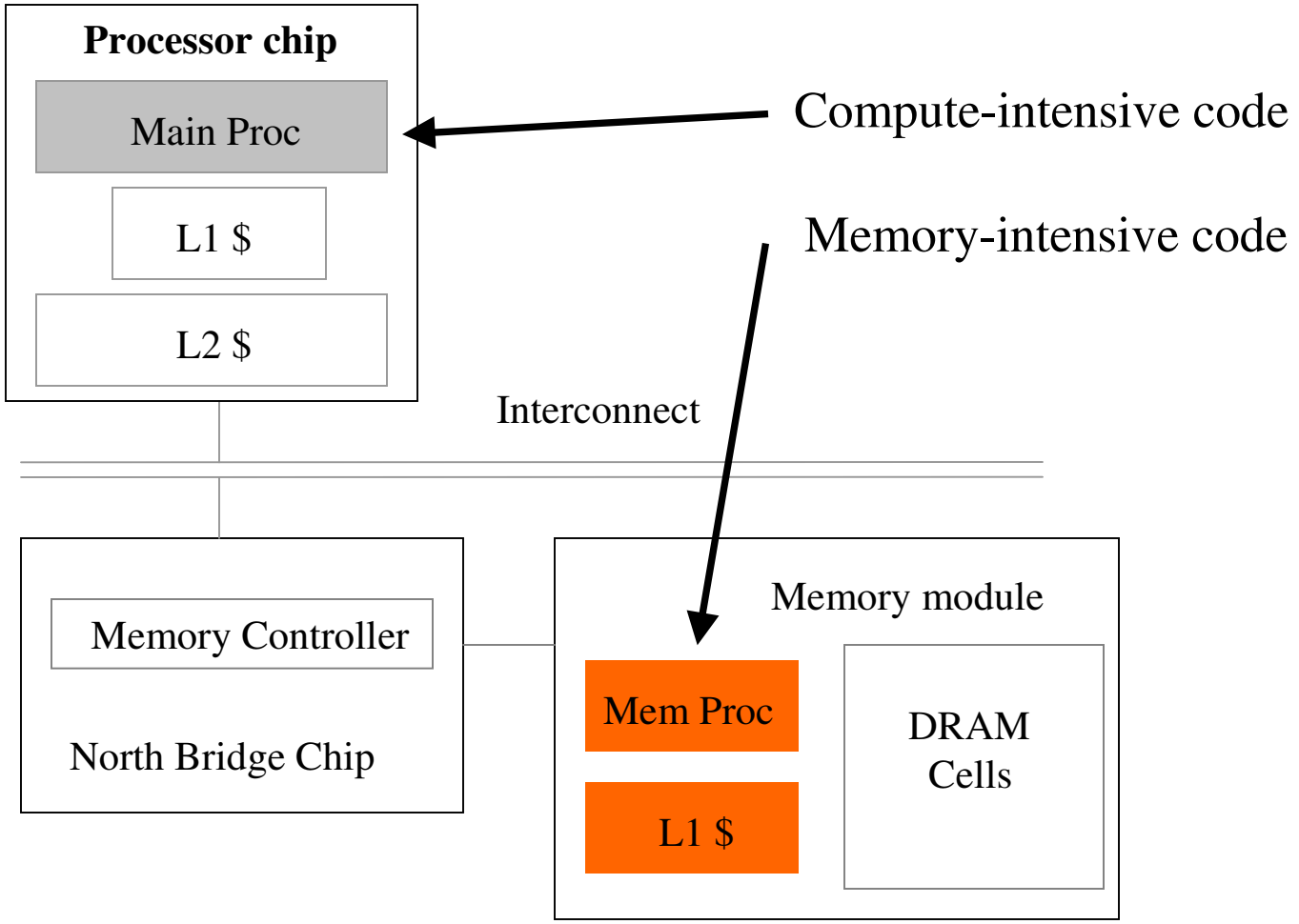
ACQUIRE

RELEASE

Safe

Speculative

Memory Processor Executes Code [HPCA01]



Memory Processor Prefetches [ISCA02]