Software-Managed Cache Coherence for fast One-Sided Communication

PMAM@PPoPP, Barcelona, Spain, March 12 2016

Steffen Christgau, Bettina Schnor

Operating Systems and Distributed Systems
Insititute for Computer Science
University of Potsdam, Germany

Motivation

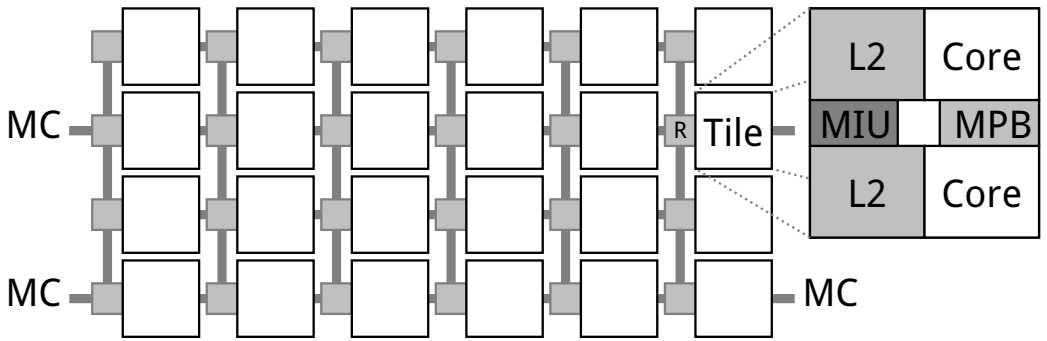
- Will future many-core systems provide hardware cache coherence?

Motivation

- Will future many-core systems provide hardware cache coherence? Not always $\rightarrow$ coherence islands in nCC systems

Motivation

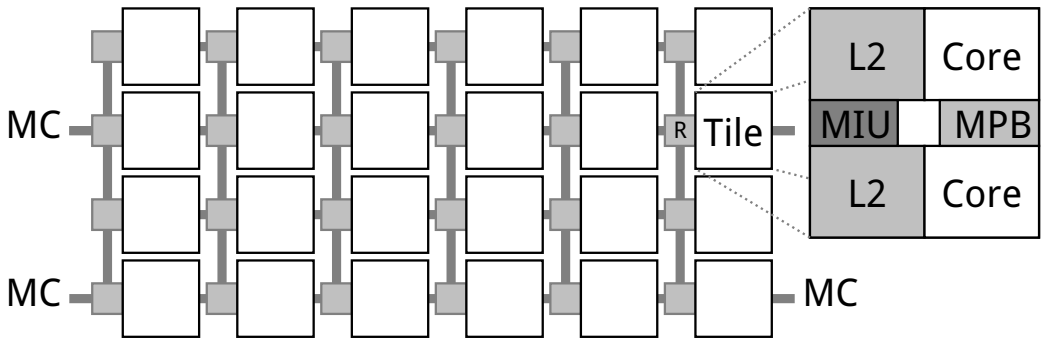
- Will future many-core systems provide hardware cache coherence? Not always $\rightarrow$ coherence islands in nCC systems
- nCC many-core research system: Intel SCC



  - 48 Pentium cores with L1/2 caches, **no HW cache coherence**
  - memory subsystem allows creation of shared memory

Motivation

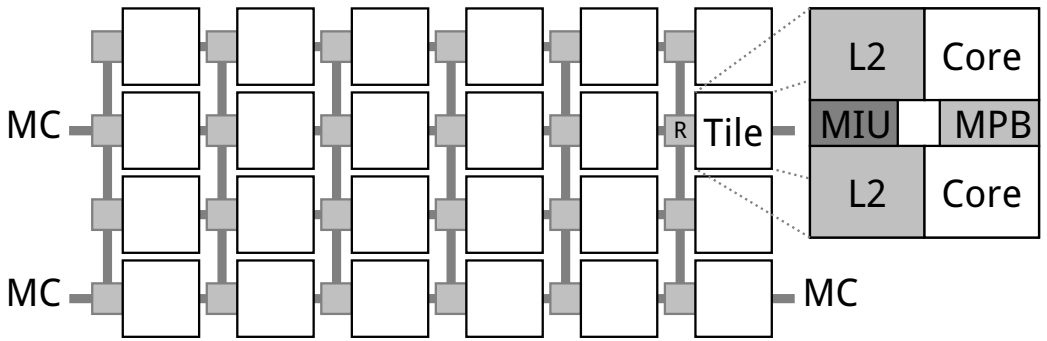
- Will future many-core systems provide hardware cache coherence? Not always $\rightarrow$ coherence islands in nCC systems
- nCC many-core research system: Intel SCC



  - 48 Pentium cores with L1/2 caches, **no HW cache coherence**
  - memory subsystem allows creation of shared memory
- previous research: focus on message passing (used MPB)

Motivation

- Will future many-core systems provide hardware cache coherence? Not always $\rightarrow$ coherence islands in nCC systems
- nCC many-core research system: Intel SCC

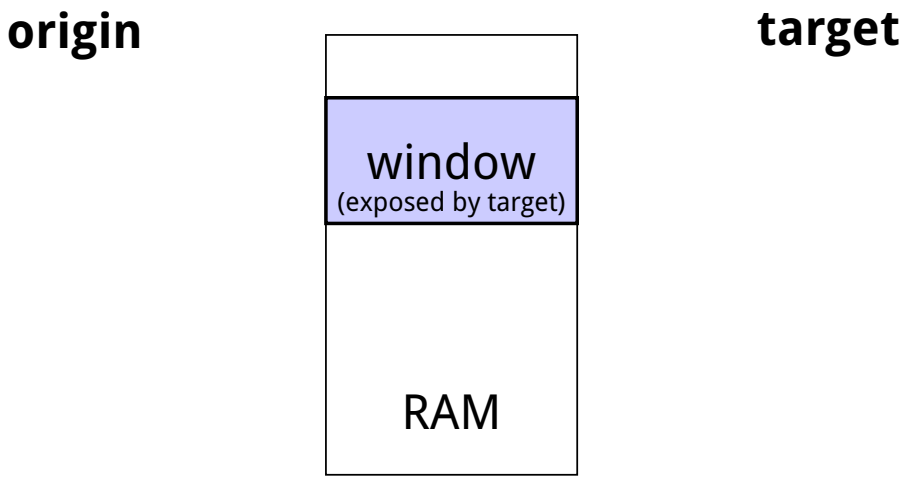


  - 48 Pentium cores with L1/2 caches, **no HW cache coherence**
  - memory subsystem allows creation of shared memory
- previous research: focus on message passing (used MPB)
- new approach: use shared memory on nCC CPU for **one-sided communication** and **manage cache coherence in software**

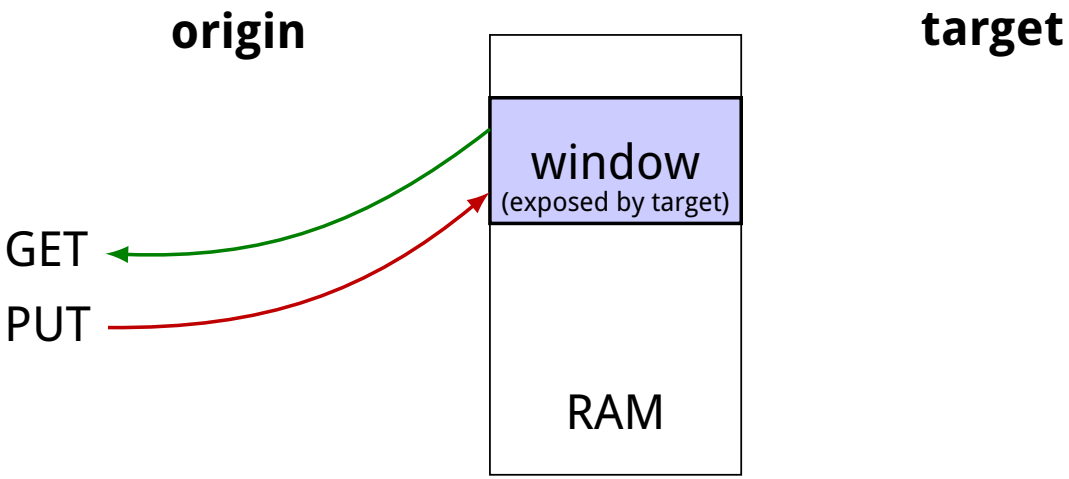
MPI One-Sided Communication

origin **target**

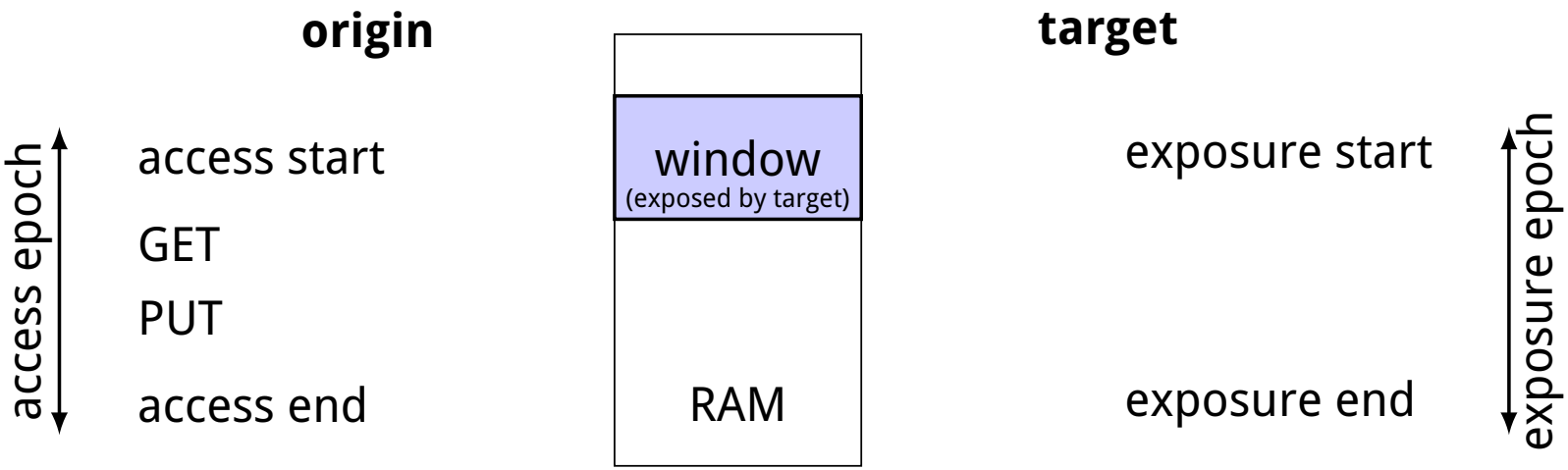
MPI One-Sided Communication



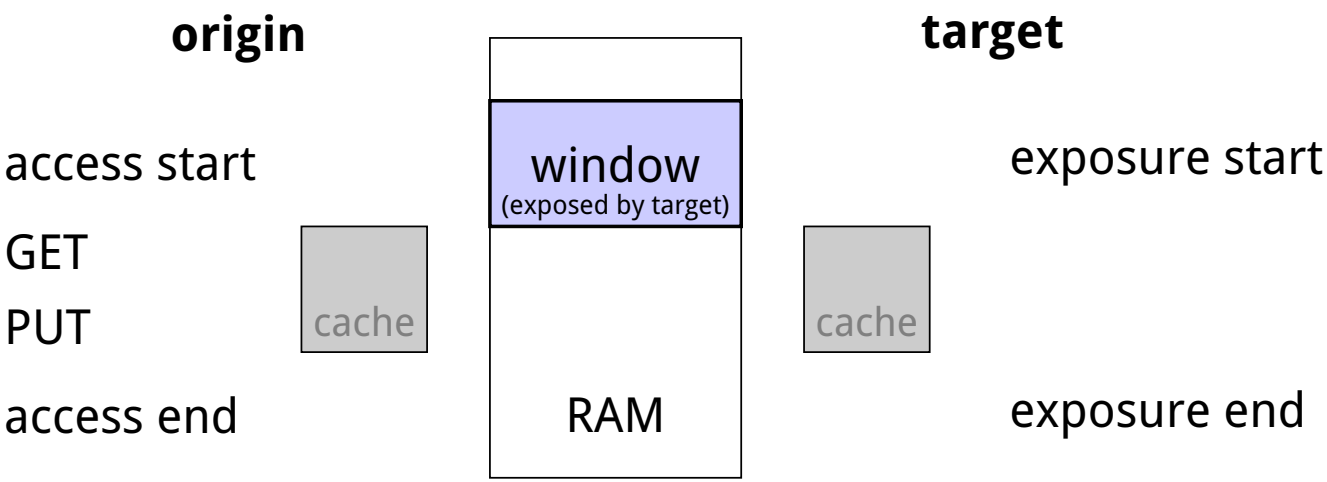
MPI One-Sided Communication



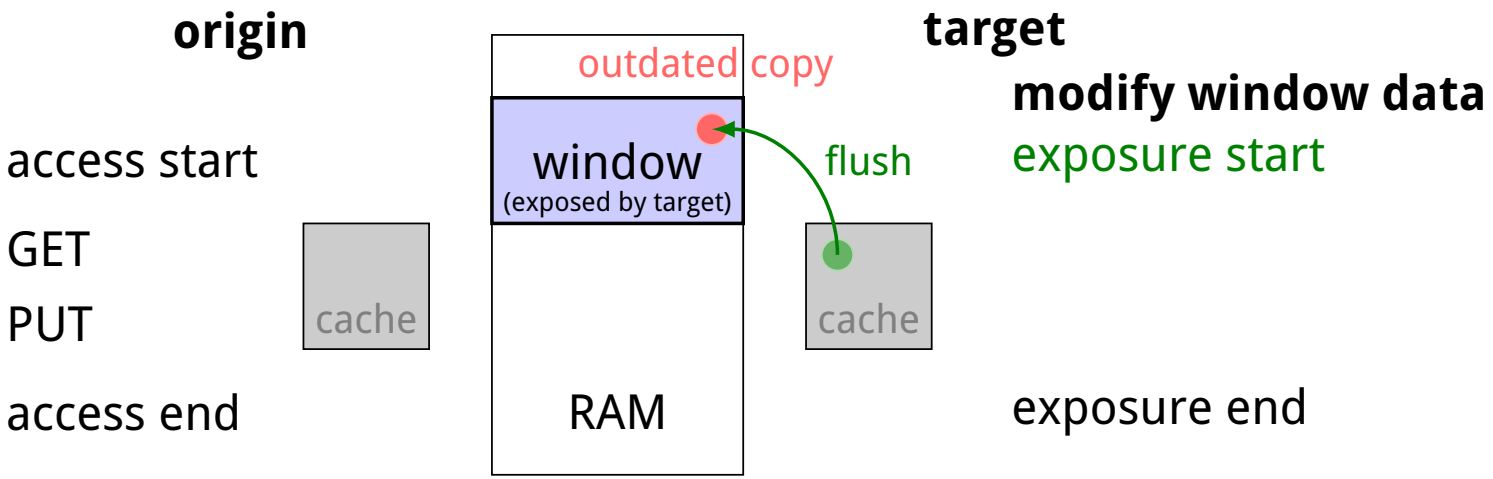
MPI One-Sided Communication



MPI One-Sided Communication



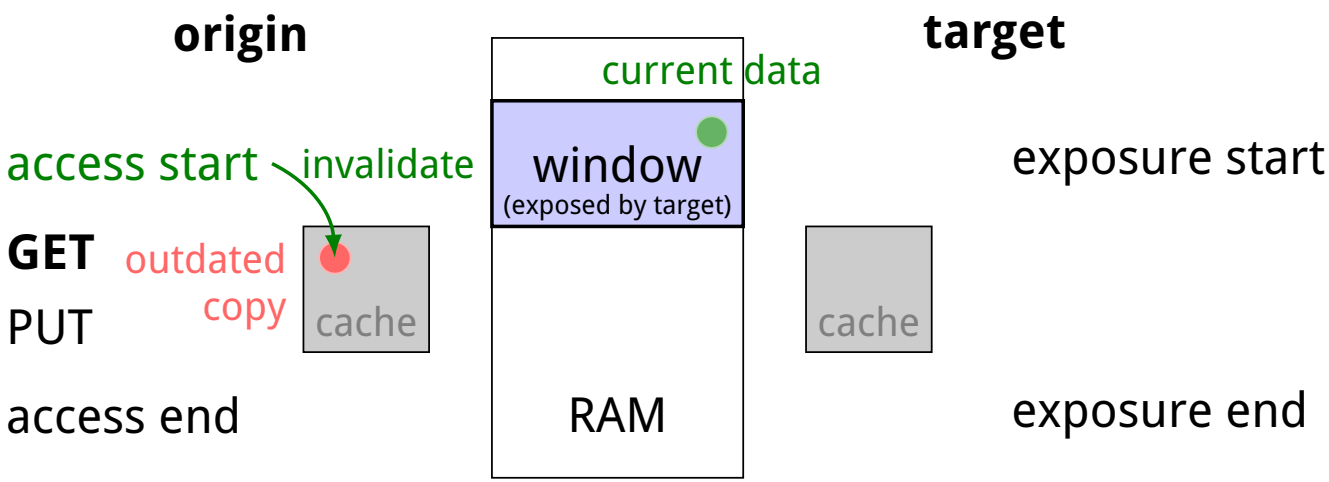
MPI One-Sided Communication



Requirements for Software-Managed Cache Coherence

target/exposure start: write window data back to RAM

MPI One-Sided Communication

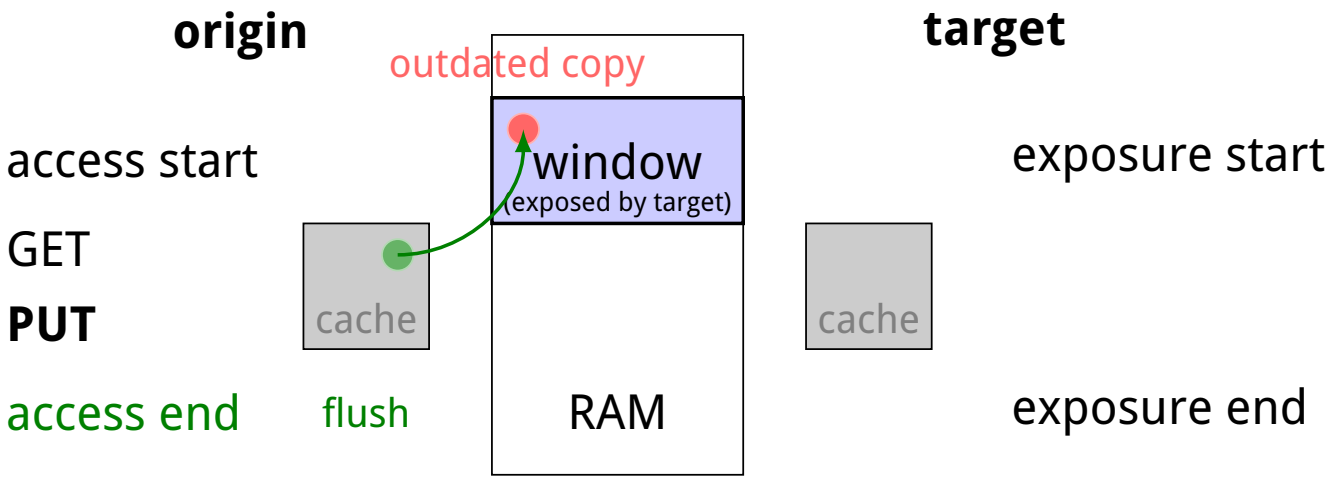


Requirements for Software-Managed Cache Coherence

target/exposure start: write window data back to RAM

origin/access start: invalidate cached window data

MPI One-Sided Communication



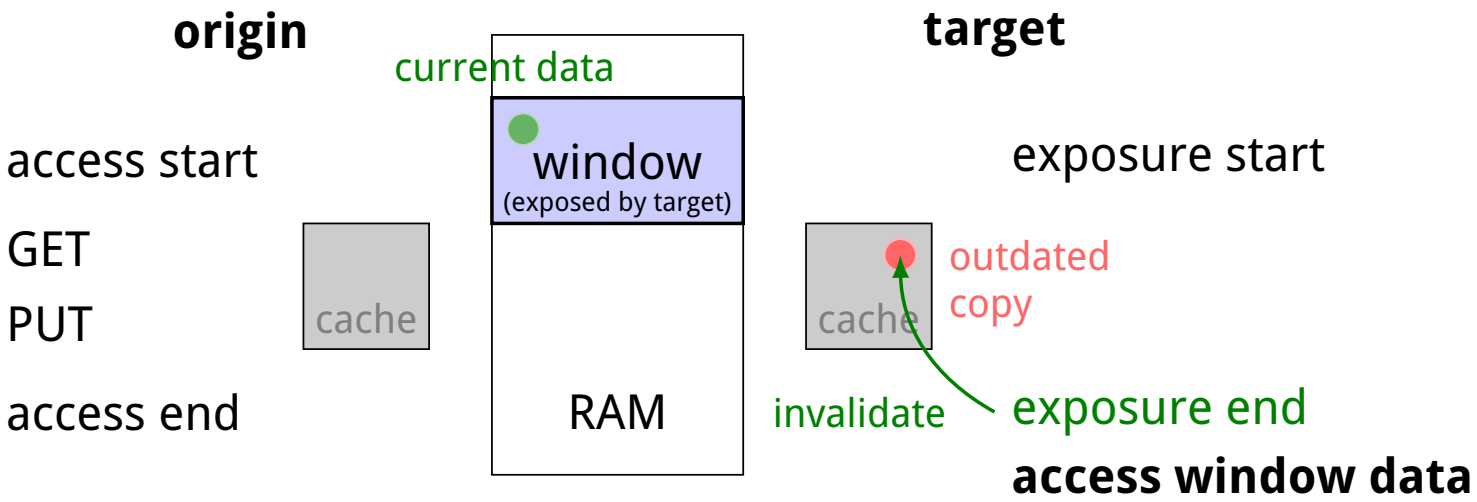
Requirements for Software-Managed Cache Coherence

target/exposure start: write window data back to RAM

origin/access start: invalidate cached window data

origin/access end: write window data back to RAM

MPI One-Sided Communication



Requirements for Software-Managed Cache Coherence

target/exposure start: write window data back to RAM

origin/access start: invalidate cached window data

origin/access end: write window data back to RAM

target/exposure end: invalidate cached window data

Implementation on the SCC: SCOSCo

- conventional cacheable/uncacheable memory types unsuited

Implementation on the SCC: SCOSCo

- conventional cacheable/uncacheable memory types unsuited
- new memory type on SCC
  - L1-cacheable, cache lines marked with special bit
  - new instruction: invalidate marked cache lines in few cycles
  - configurable L1 behavior: write-back or write-through
  - write-combine buffer

Implementation on the SCC: SCOSCo

- conventional cacheable/uncacheable memory types unsuited
- new memory type on SCC
  - L1-cacheable, cache lines marked with special bit
  - new instruction: invalidate marked cache lines in few cycles
  - configurable L1 behavior: write-back or write-through
  - write-combine buffer
- use new memory type + write-through for window

Implementation on the SCC: SCOSCo

- conventional cacheable/uncacheable memory types unsuited
- new memory type on SCC
  - L1-cacheable, cache lines marked with special bit
  - new instruction: invalidate marked cache lines in few cycles
  - configurable L1 behavior: write-back or write-through
  - write-combine buffer
- use new memory type + write-through for window
- requirement 1: write window data back to RAM
  $\rightarrow$ **write-through cache configuration** for window

Implementation on the SCC: SCOSCo

- conventional cacheable/uncacheable memory types unsuited
- new memory type on SCC
  - L1-cacheable, cache lines marked with special bit
  - new instruction: invalidate marked cache lines in few cycles
  - configurable L1 behavior: write-back or write-through
  - write-combine buffer
- use new memory type + write-through for window
- requirement 1: write window data back to RAM
  $\rightarrow$ **write-through cache configuration** for window
- requirement 2: invalidate cached data
  $\rightarrow$ issue fast **invalidate instruction**

Implementation on the SCC: SCOSCo

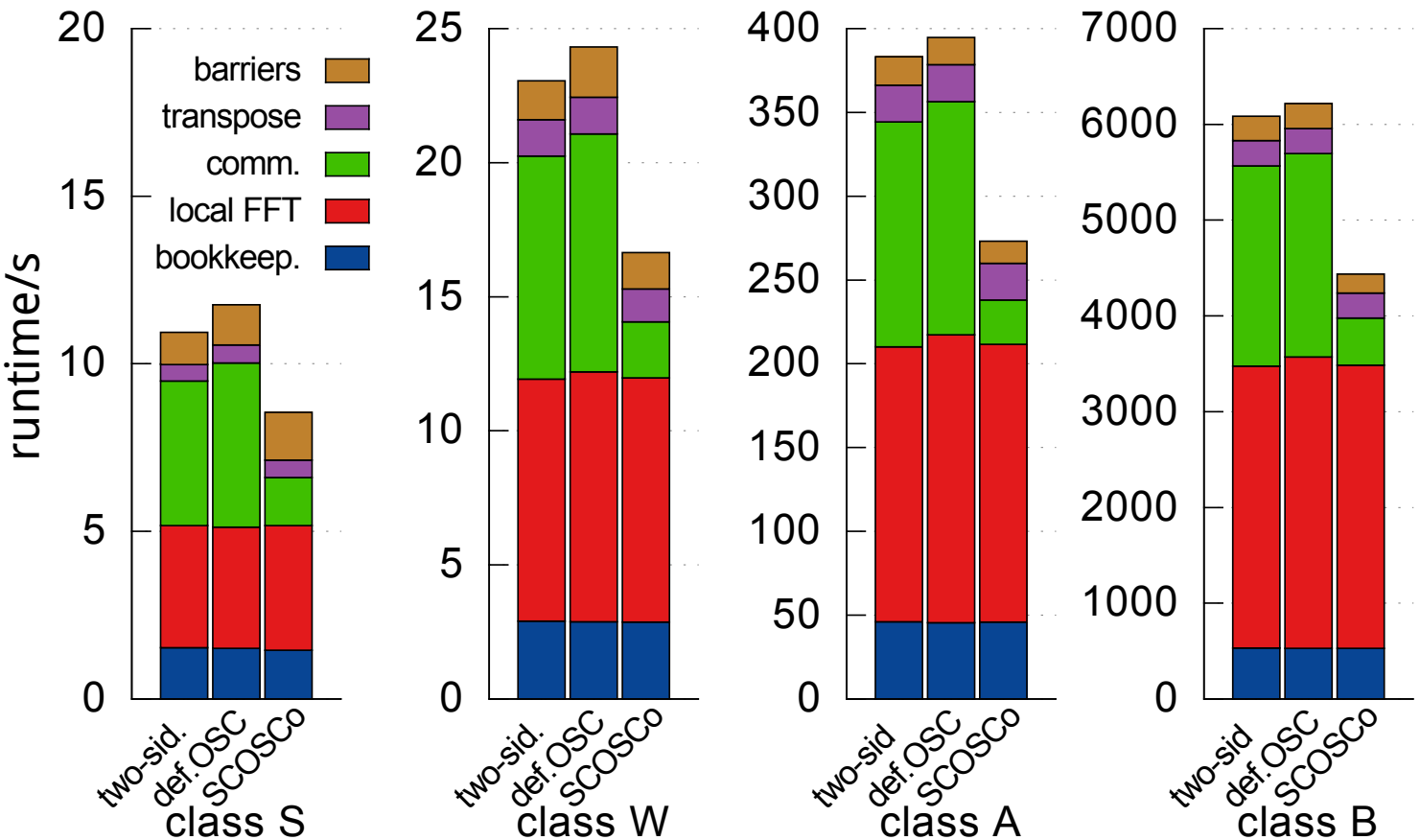
- conventional cacheable/uncacheable memory types unsuited
- new memory type on SCC
  - L1-cacheable, cache lines marked with special bit
  - new instruction: invalidate marked cache lines in few cycles
  - configurable L1 behavior: write-back or write-through
  - write-combine buffer
- use new memory type + write-through for window
- requirement 1: write window data back to RAM
  $\rightarrow$ **write-through cache configuration** for window
- requirement 2: invalidate cached data
  $\rightarrow$ issue fast **invalidate instruction**
- but: SCC's write-through has uncached memory performance
  - use write-back configuration as substitute
  - for benchmarks: ensure cache miss on write

Experimental Results: 3D FFT

- MPI implementation of NPB-FT
  - origins only PUT $\rightarrow$ window gets updated correctly
  - assume similar performance as working write-through memory

Experimental Results: 3D FFT

- MPI implementation of NPB-FT
  - origins only PUT $\rightarrow$ window gets updated correctly
  - assume similar performance as working write-through memory



Wishful Thinking for Fast OSC on nCC Systems

Recommendations for Future Systems

potentials for further improvements

Wishful Thinking for Fast OSC on nCC Systems

Recommendations for Future Systems

potentials for further improvements

- slow flush to RAM
  - slow performance of working write-through memory (?)
  - (full) cache flush slow anyway

Wishful Thinking for Fast OSC on nCC Systems

Recommendations for Future Systems

potentials for further improvements

- slow flush to RAM
- wasteful invalidation
  - removes all window data (not only modified)
  - affects other cached data of same type

Wishful Thinking for Fast OSC on nCC Systems

Recommendations for Future Systems

potentials for further improvements

- slow flush to RAM
- wasteful invalidation

desireable beneficial hardware features in future nCC systems:

Wishful Thinking for Fast OSC on nCC Systems

Recommendations for Future Systems

potentials for further improvements

- slow flush to RAM
- wasteful invalidation

desireable beneficial hardware features in future nCC systems:

- **flushable write-combine buffer** for write-through memory
  - alternative: address range-based flush of modified cache lines

Wishful Thinking for Fast OSC on nCC Systems

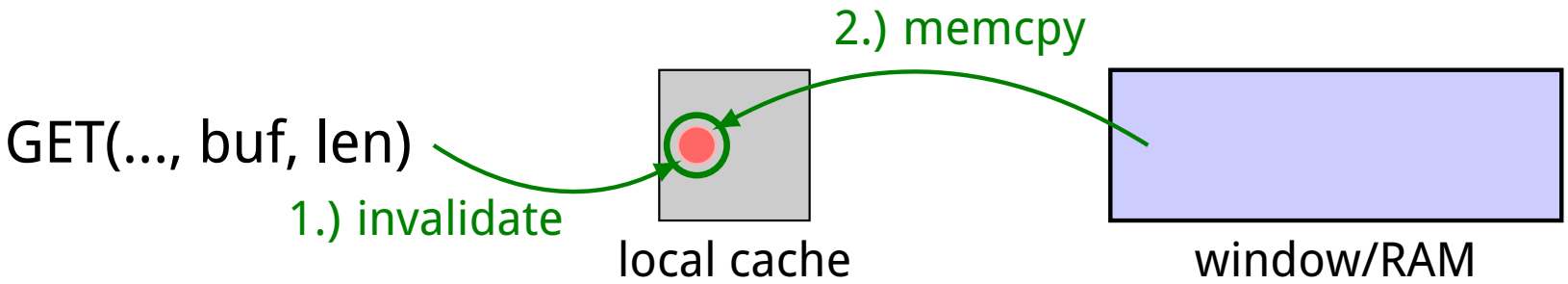
Recommendations for Future Systems

potentials for further improvements

- slow flush to RAM
- wasteful invalidation

desireable beneficial hardware features in future nCC systems:

- **flushable write-combine buffer** for write-through memory
- address **range-based cache invalidation** (for GETs)



Wishful Thinking for Fast OSC on nCC Systems

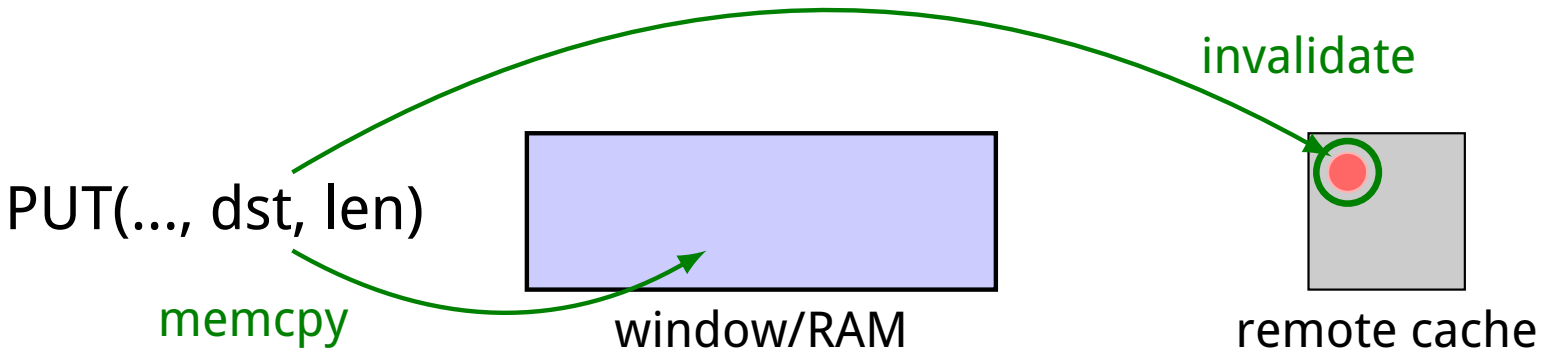
Recommendations for Future Systems

potentials for further improvements

- slow flush to RAM
- wasteful invalidation

desireable beneficial hardware features in future nCC systems:

- **flushable write-combine buffer** for write-through memory
- address **range-based cache invalidation** (for GETs)
- **remote cache invalidation** by address range (for PUTs)



Wishful Thinking for Fast OSC on nCC Systems

Recommendations for Future Systems

potentials for further improvements

- slow flush to RAM
- wasteful invalidation

desireable beneficial hardware features in future nCC systems:

- **flushable write-combine buffer** for write-through memory
- address **range-based cache invalidation** (for GETs)
- **remote cache invalidation** by address range (for PUTs)

$\rightarrow$ more explicit control over caches for software

Summary

- demonstrated implementation for software-based cache coherence on nCC many-core CPU
- shared memory approach outperforms message-based solution
  - 4–5x less communication time (40–45% app. speedup) for FFT
- identified beneficial features for future systems
  - local and remote address-based cache invalidation

Summary

- demonstrated implementation for software-based cache coherence on nCC many-core CPU
- shared memory approach outperforms message-based solution
  - 4–5x less communication time (40–45% app. speedup) for FFT
- identified beneficial features for future systems
  - local and remote address-based cache invalidation

Thanks for your attention!