BulkCompactor: Optimized Deterministic Execution via Conflict-Aware Commit of Atomic Blocks

Yuelu Duan, Xing Zhou, Wonsun Ahn, Josep Torrellas

University of Illinois at Urbana-Champaign

http://iacoma.cs.uiuc.edu

HPCA 2012, New Orleans, LA

i-acoma group

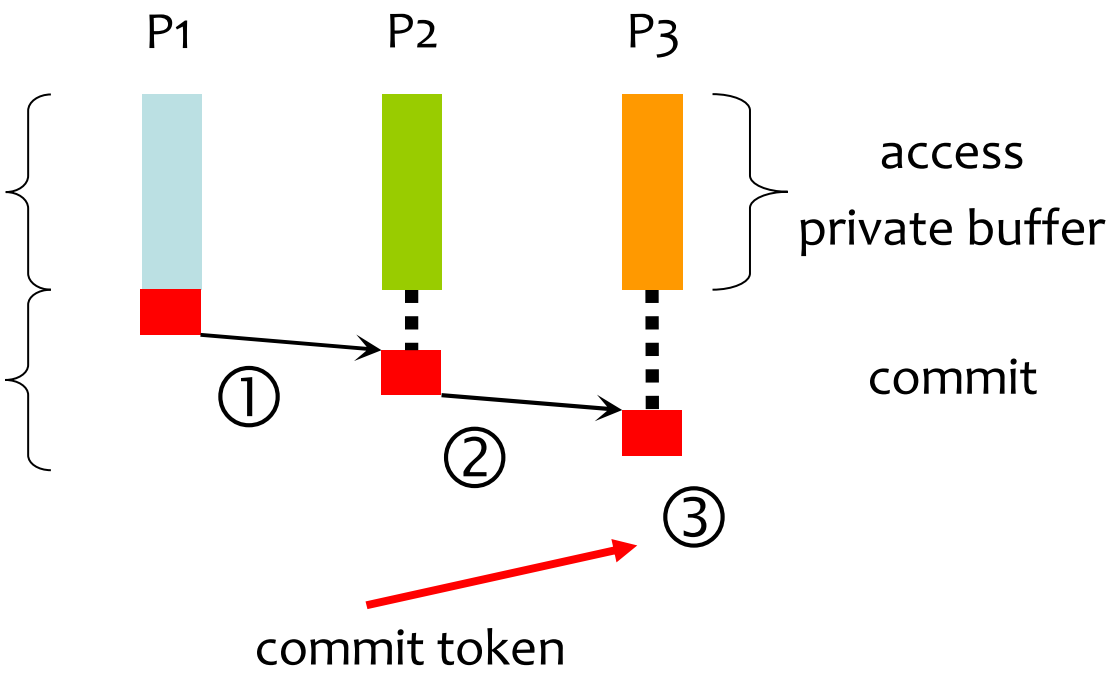
Why Deterministic Execution?

- Shared-memory multiprocessing is nondeterministic
  - Hard to reproduce execution/error
  - Complicates concurrent programming, testing and debugging
- Deterministic Execution
  - Given an input, always produces same result
  - Enforces an identical thread interleaving at each run

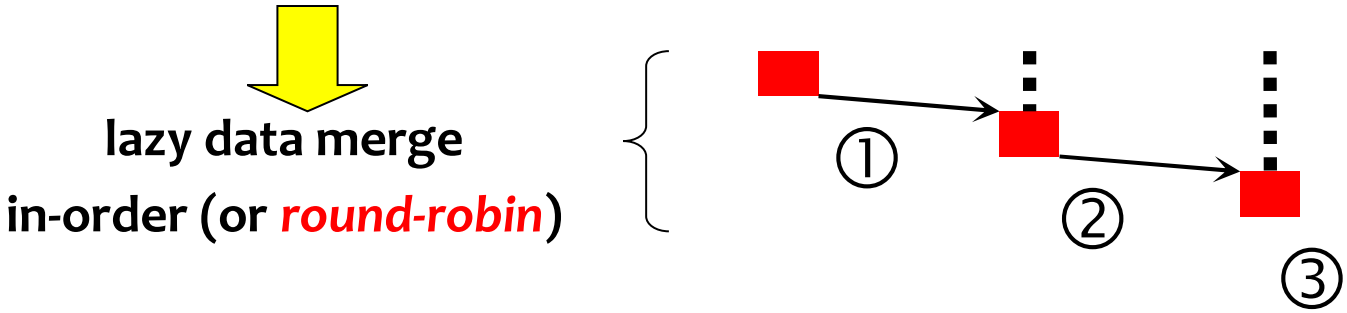
Current Deterministic Execution Models

parallel execution of
***chunks* w/o communication**

lazy data merge
in-order (or *round-robin*)



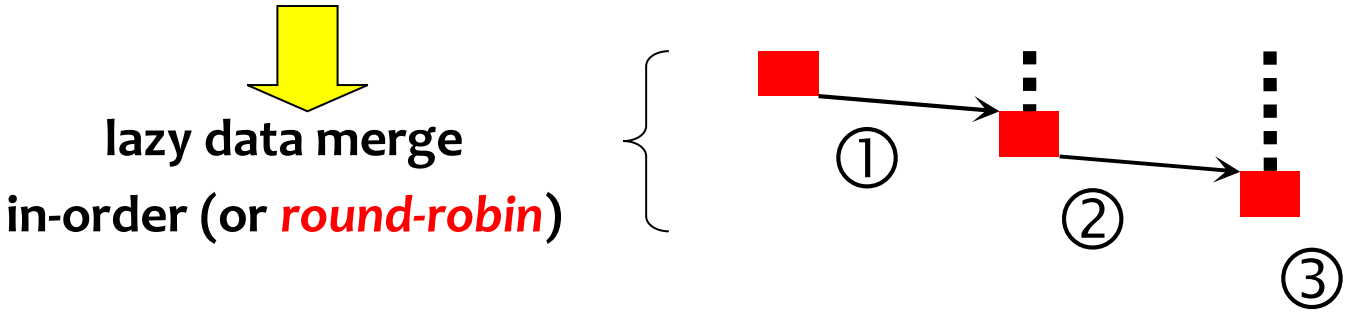
Current Deterministic Execution Models



On identifying a Data Conflict during the merge:

- Ignore it
  - + Fast
  - −Unintuitive memory model -- hard to reason

Current Deterministic Execution Models



On identifying a Data Conflict during the merge:

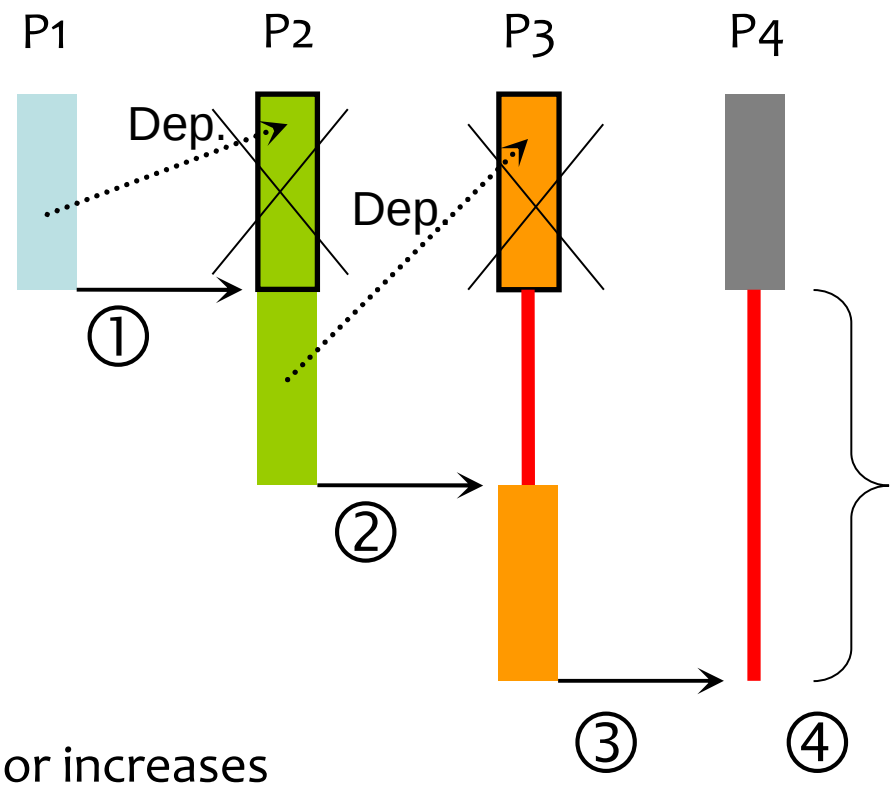
- Honor the dependence --> squash the chunk
  - + Intuitive memory model
  - − Becomes **performance bottleneck**

Understand & Fix this!

Problem in Round Robin: Transitive Squash Delay

Transitive Squash Delay: On conflict & squash, all the successor processors observe the squashed work overhead

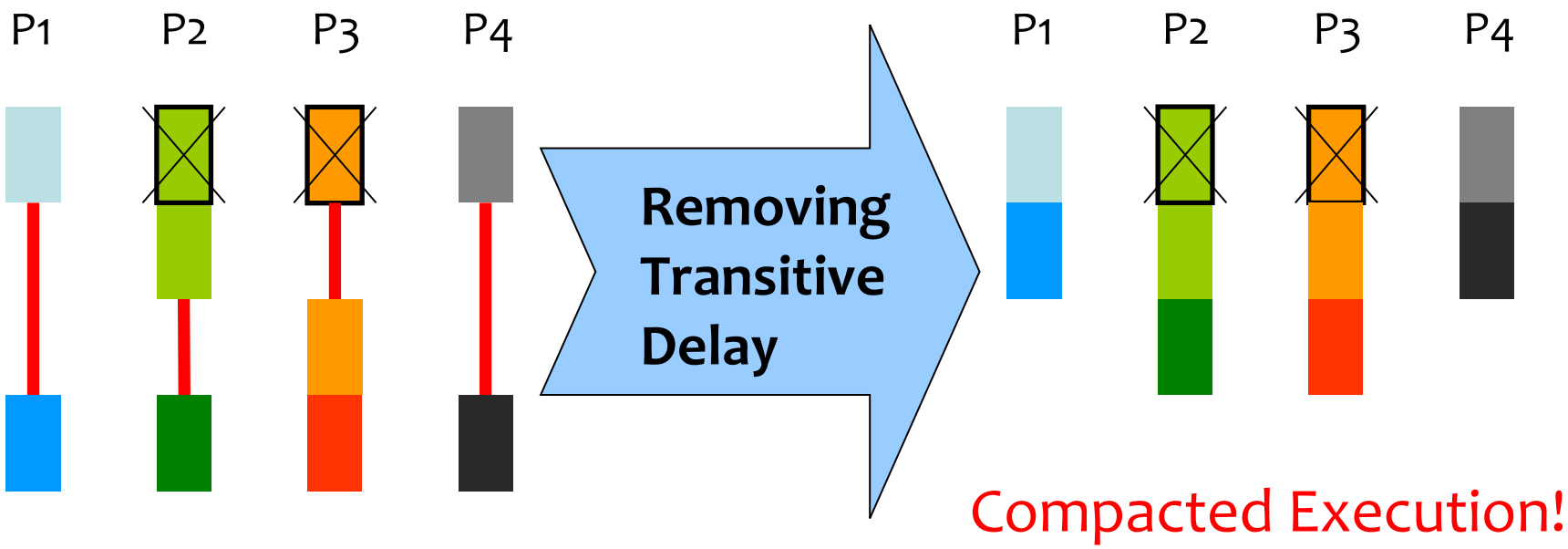
Multiple transitive squash delays accumulate



- With more processors:
  - Squash rate of each processor increases
  - More processors cause transitive squash delay
  - Overhead increases as $O(N^2)$

Our Contributions

- BulkCompactor: Conflict-Aware Deterministic Commit
  - Avoid transitive squash delay



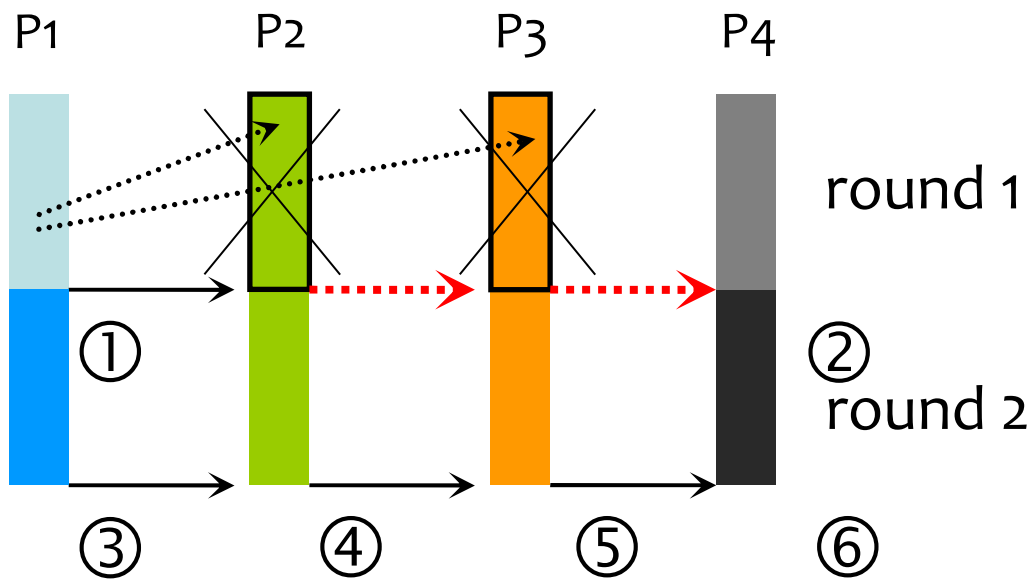
Our Contributions

- BulkCompactor: Conflict-Aware Deterministic Commit
  - Avoid transitive squash delay
- Design of a Centralized & Distributed Commit Machine
- Results for 32 processor runs
  - Only 22% execution overhead over a nondeterm. system
  - Overhead scalable as O(N)
  - Conventional round-robin deterministic execution: 133% overhead and $O(N^2)$

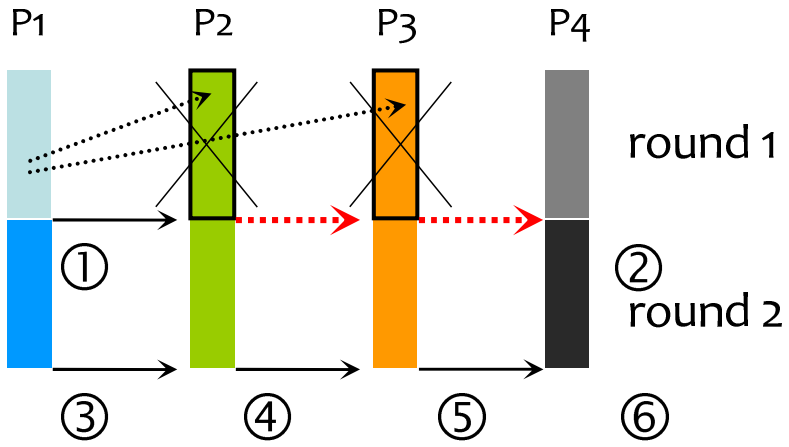
Conflict-Aware Commit

Key Idea:

Commit token bypasses a squashed chunk, and postpones it to a new round

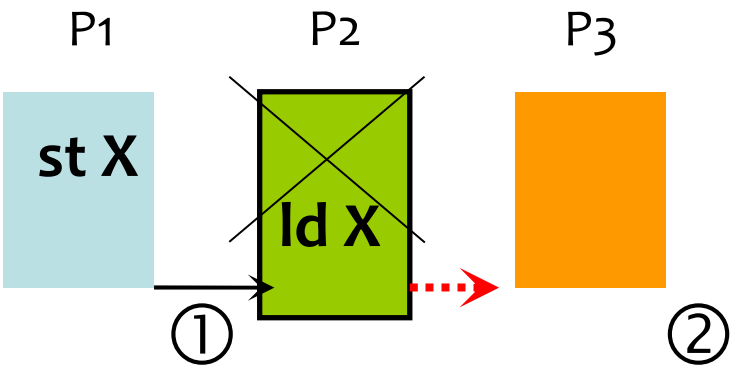


Conflict-Aware Commit

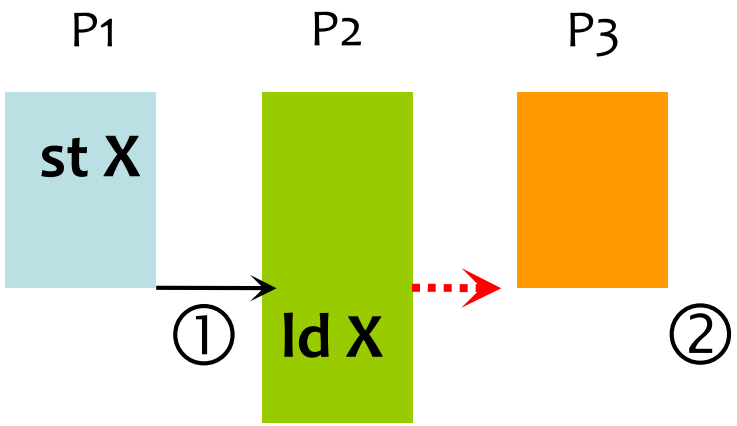


- Benefits:
  - A squashed chunk does not cause transitive squash delay
  - A conflict-free chunk can proceed to commit regardless of other's behavior

Squashes are not Deterministic



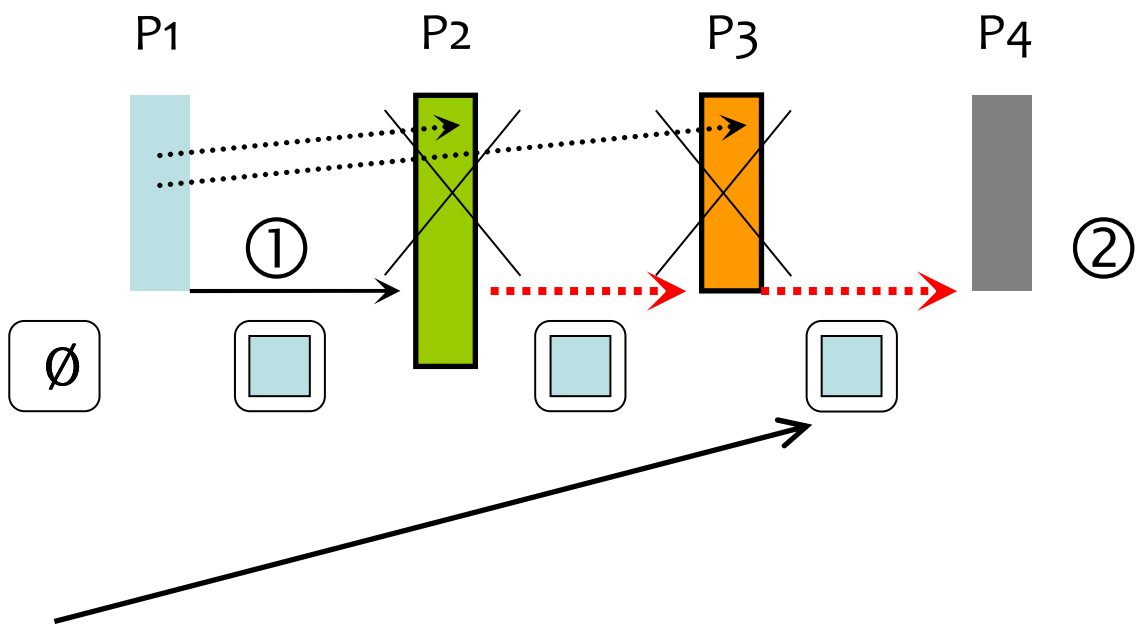
P2 is squashed
& postponed



P2 is not squashed,
but ***must*** be postponed!

➢ Dependences are Deterministic!

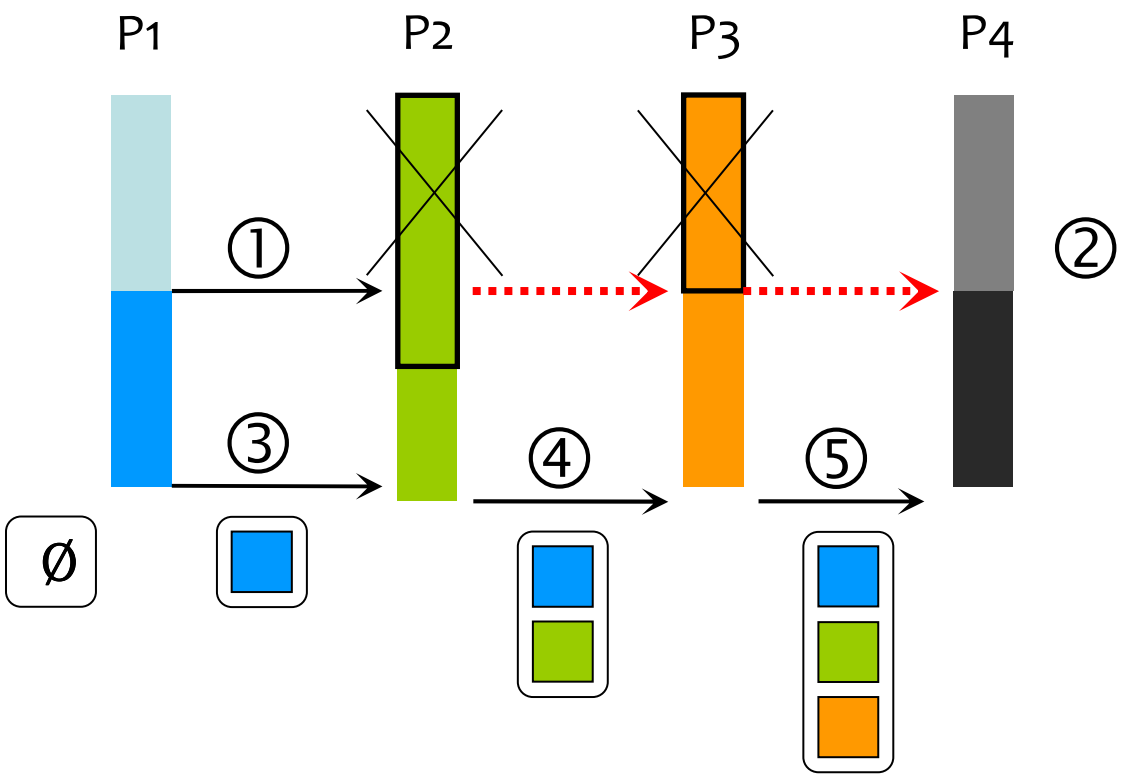
Enforcing the Dependences



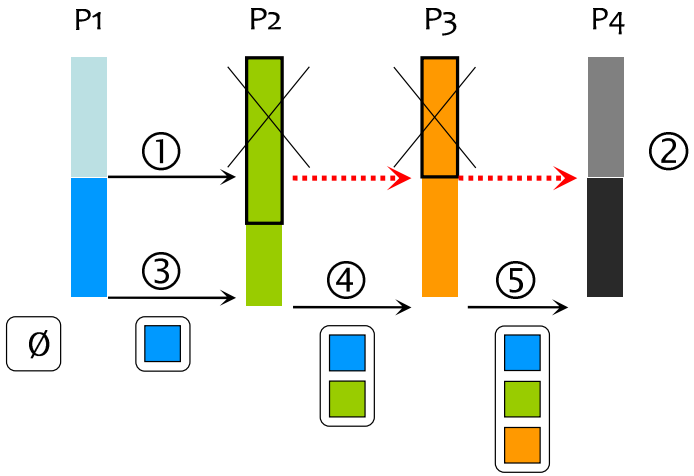
Non-Postponed Set:

Set of write addresses in the committed chunks

Enforcing the Dependences



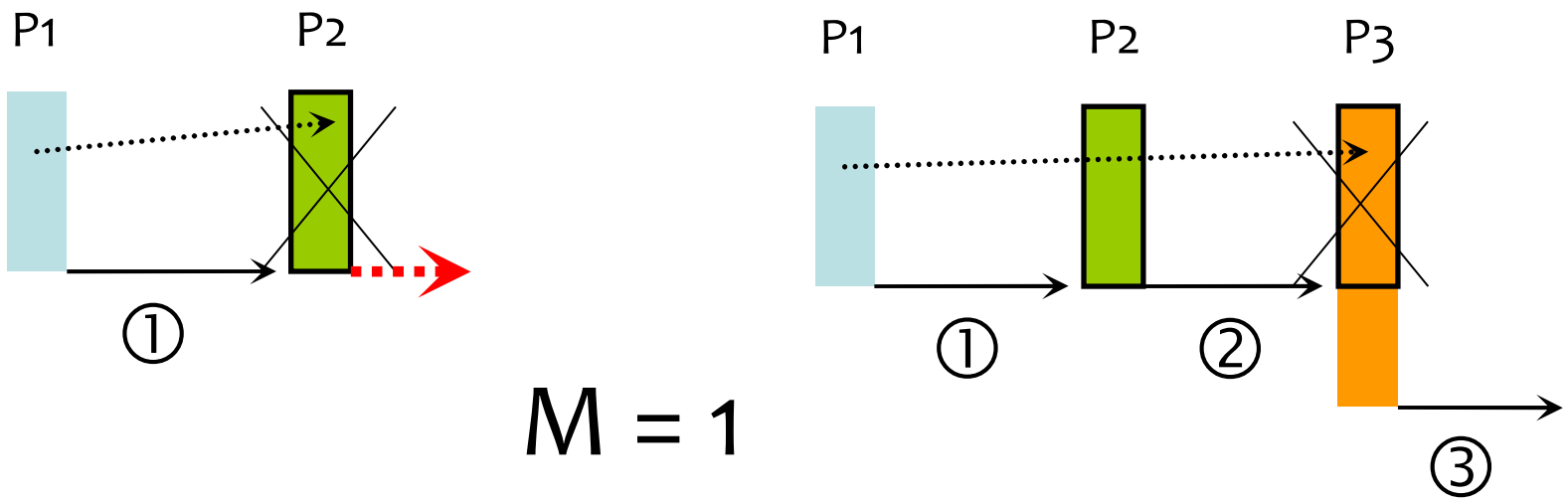
Enforcing the Dependences



- Limitation:
  - Non-Postponed Set may become very large
  - O(N) chunks are compared for each commit

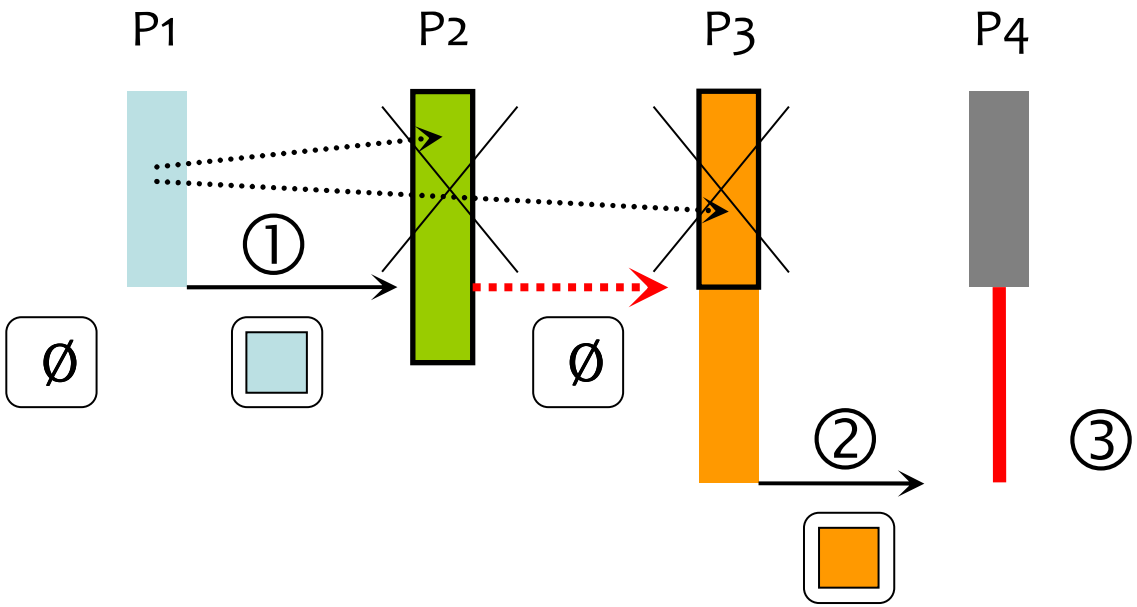
BulkCompactor-S: Scalable Postponement

- Dependences between M consecutive chunks:
  - As usual: cause chunk postponement
- Dependences beyond M consecutive chunks:
  - Cause squash and re-execution in critical path



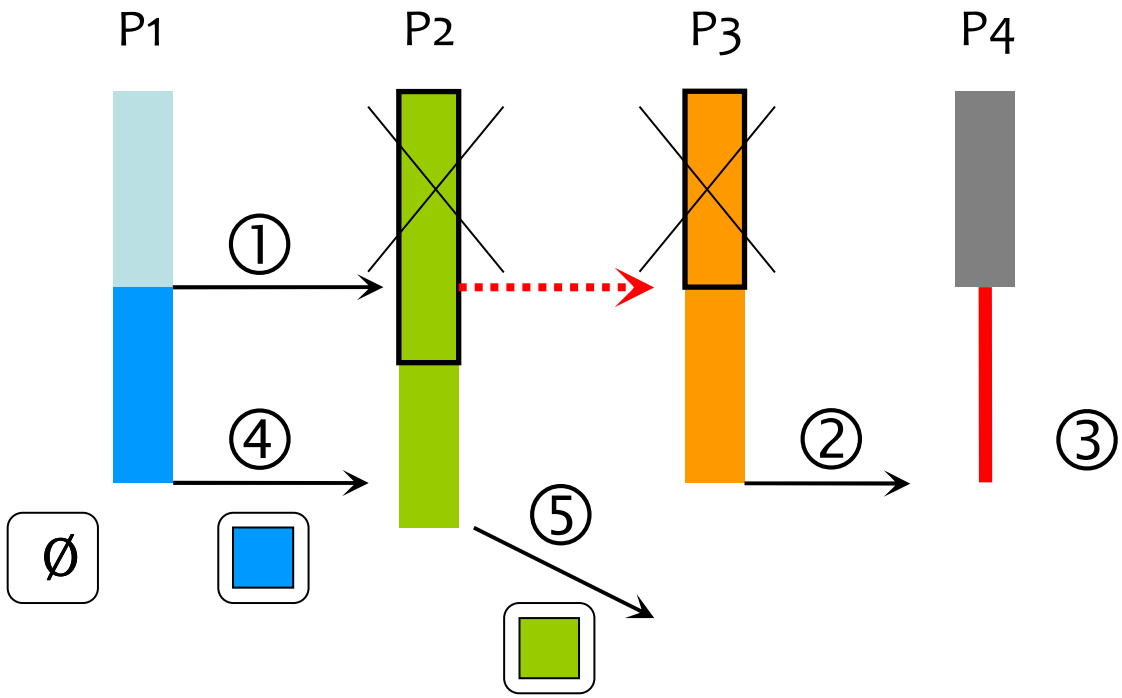
BulkCompactor-S Operation

M = 1



BulkCompactor-S Operation

M = 1



BulkCompactor-S

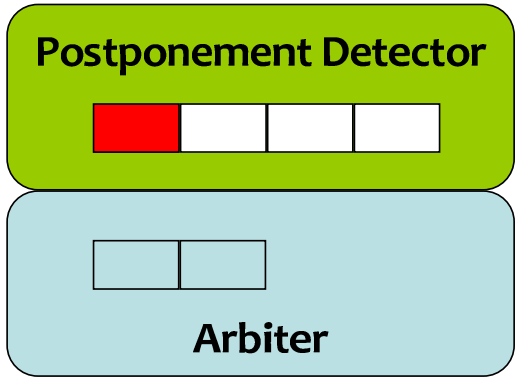
- Advantage:
  - Scalable: only O(M) chunks are compared
  - If most conflicts are between neighbor processors (***conflict-locality***) --> avoid most transitive delays
- Disadvantage:
  - Some transitive squash delay are not avoided

Outline

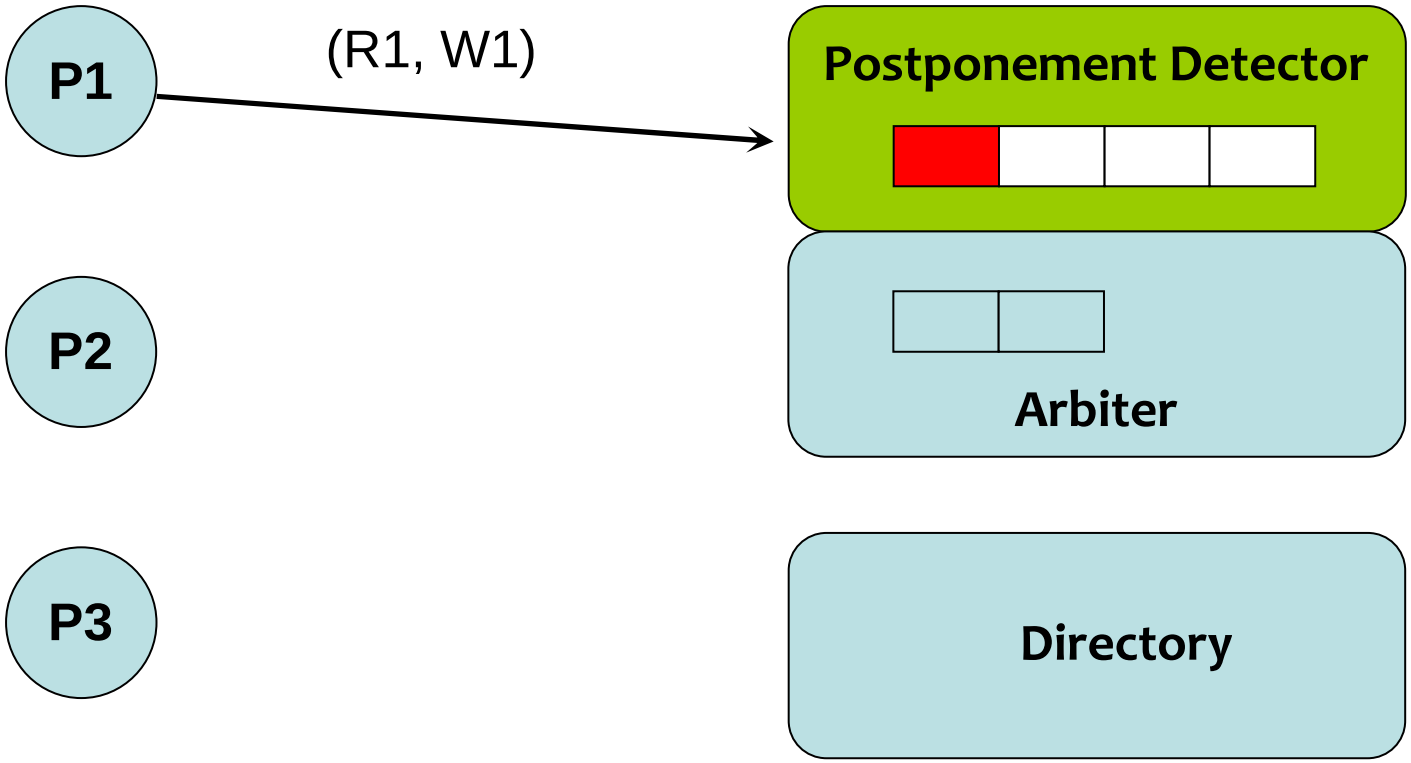
- BulkCompactor & BulkCompactor-S
- Design of a Centralized & Distributed Commit Machine

Centralized Commit Machine

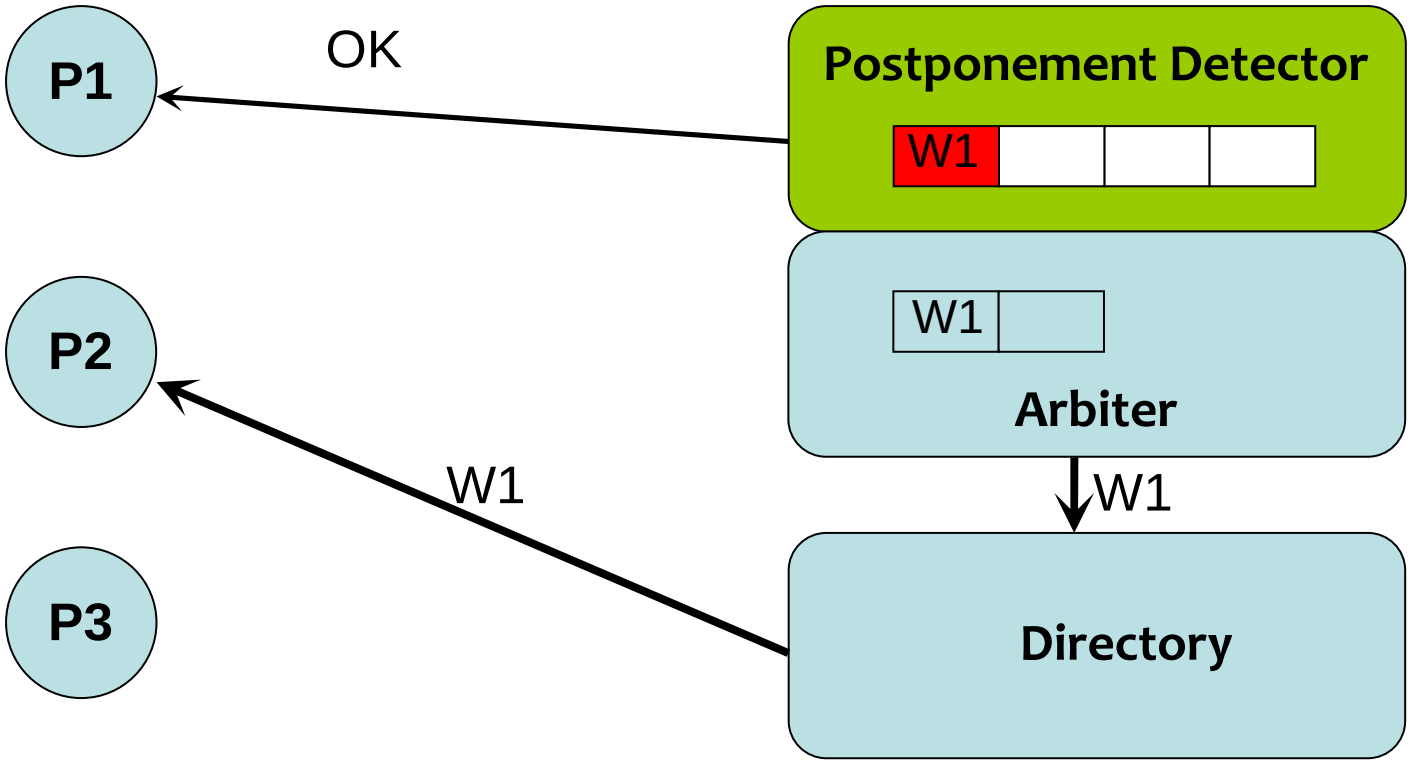
- Extensions to BulkSC[ISCA'07]
- Postponement Detector Module
  - Keeps Non-Postponed Set and commit token
  - Placed together with commit arbiter
  - Uses signatures to encode addresses in Non-Postponed Set and to perform conflict detection



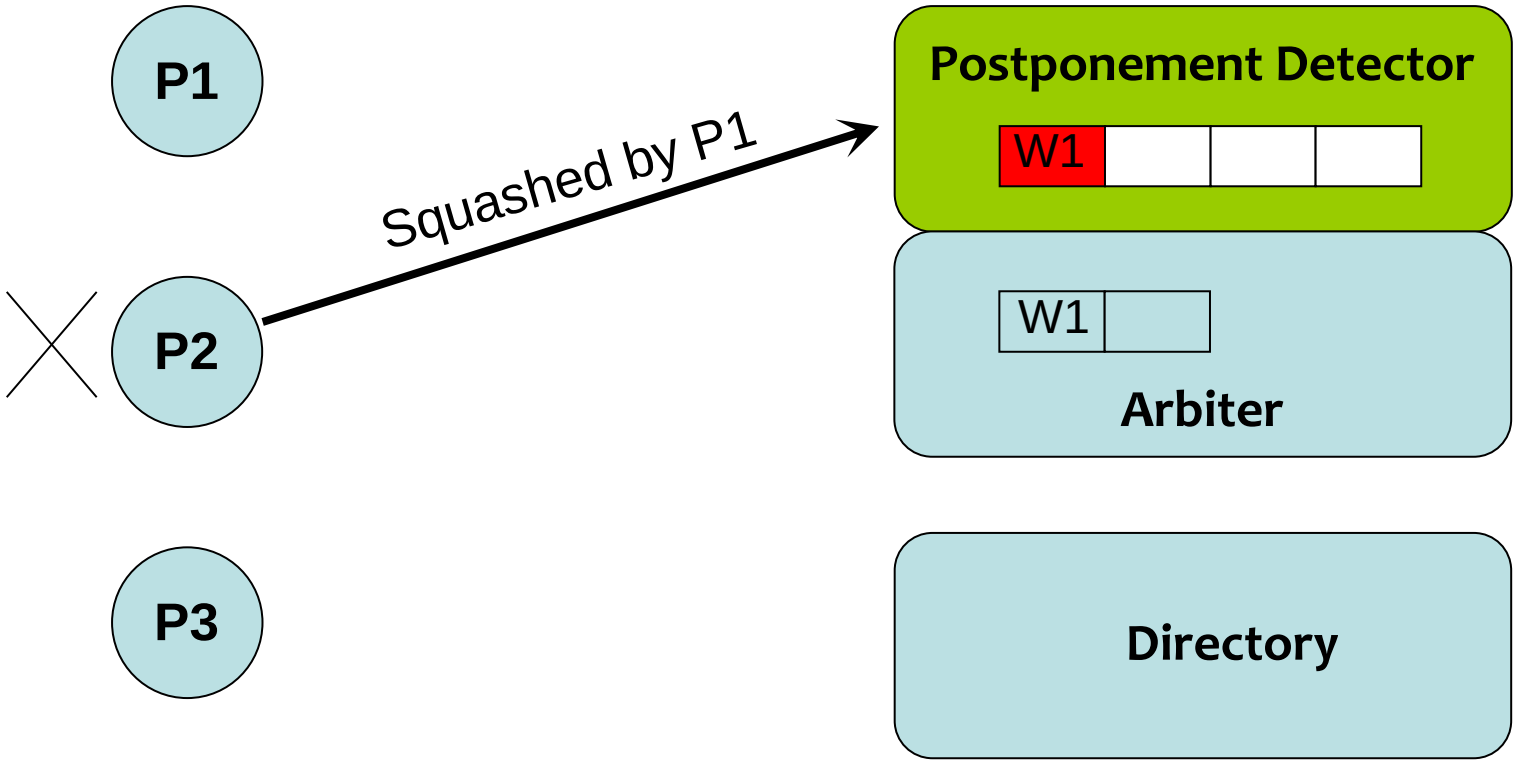
Complete Commit/Postponed Process



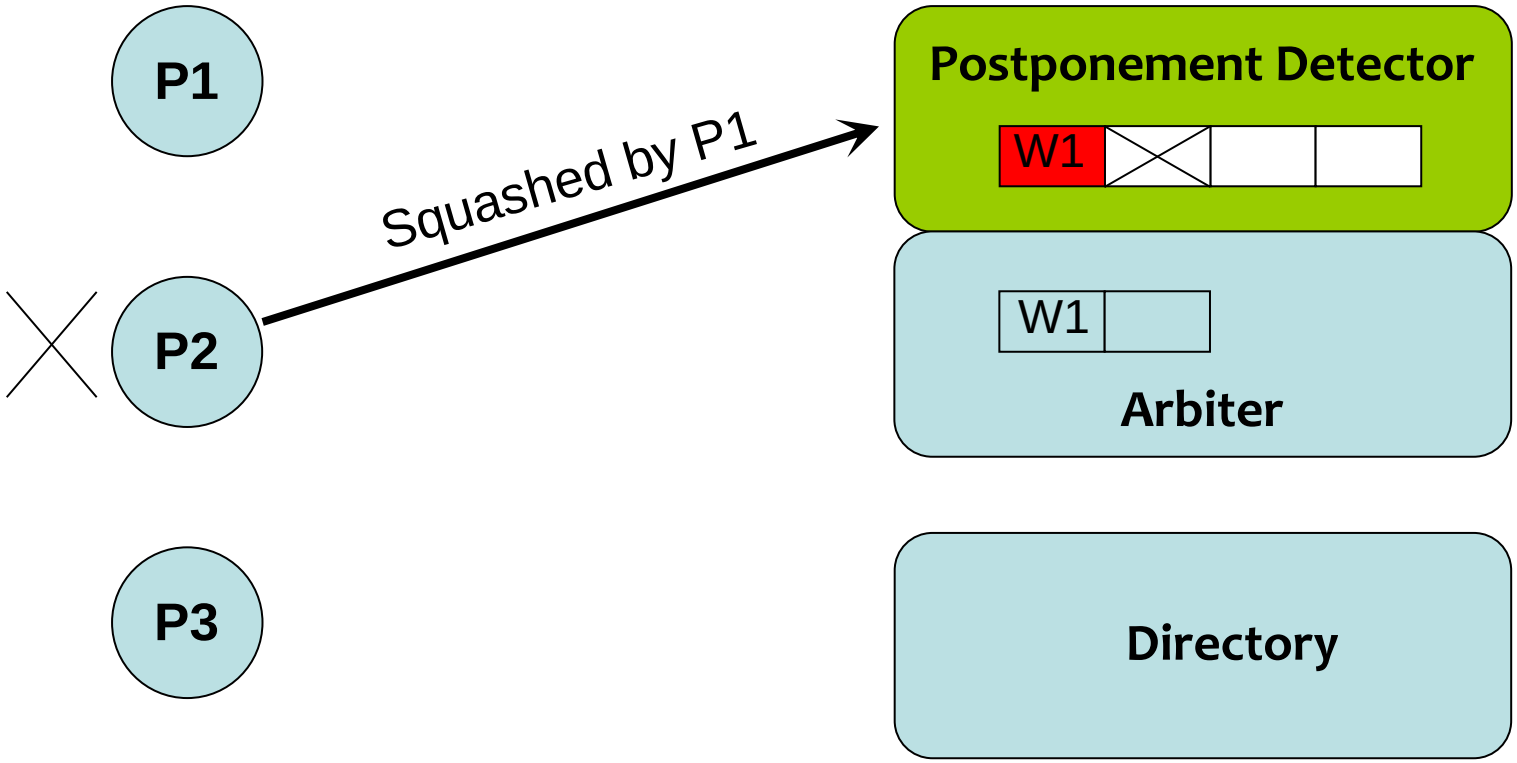
Complete Commit/Postponed Process



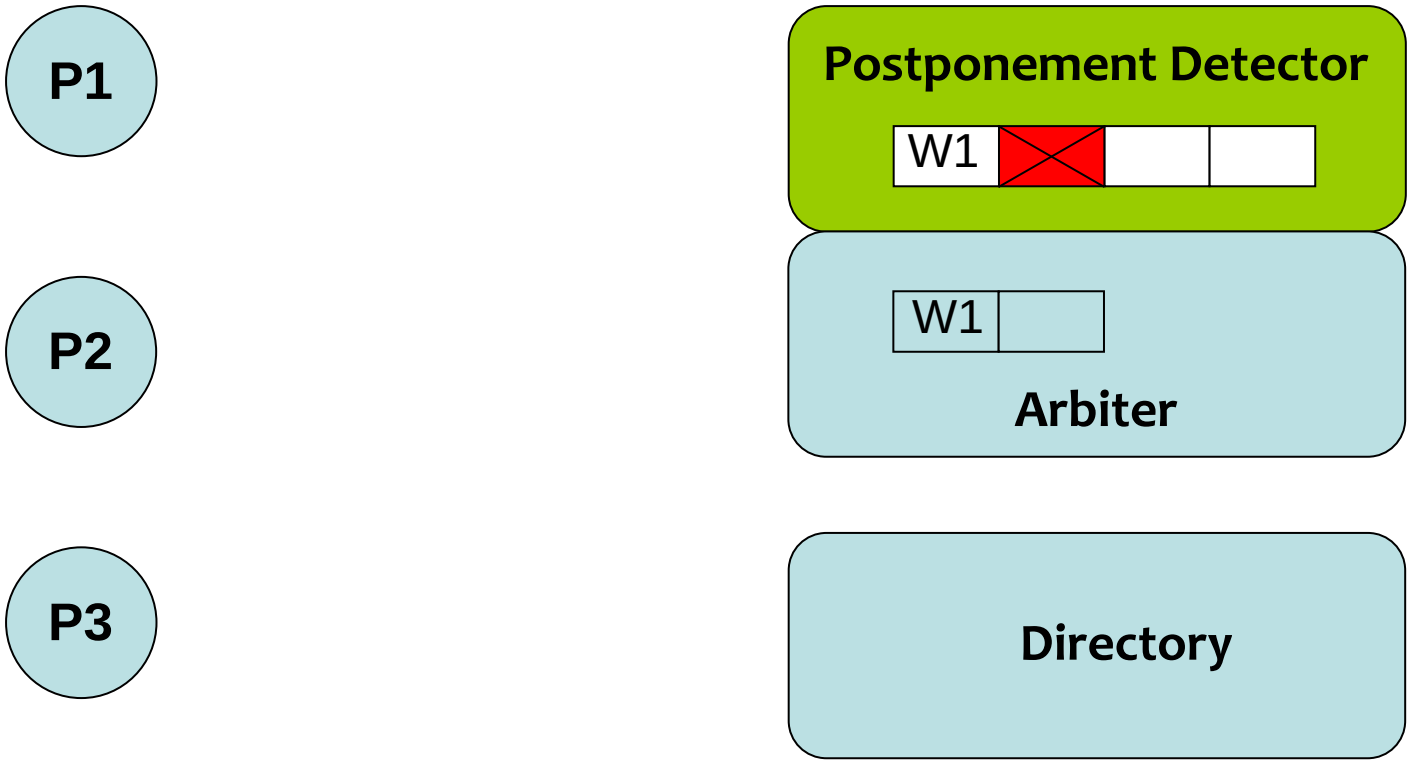
Complete Commit/Postponed Process



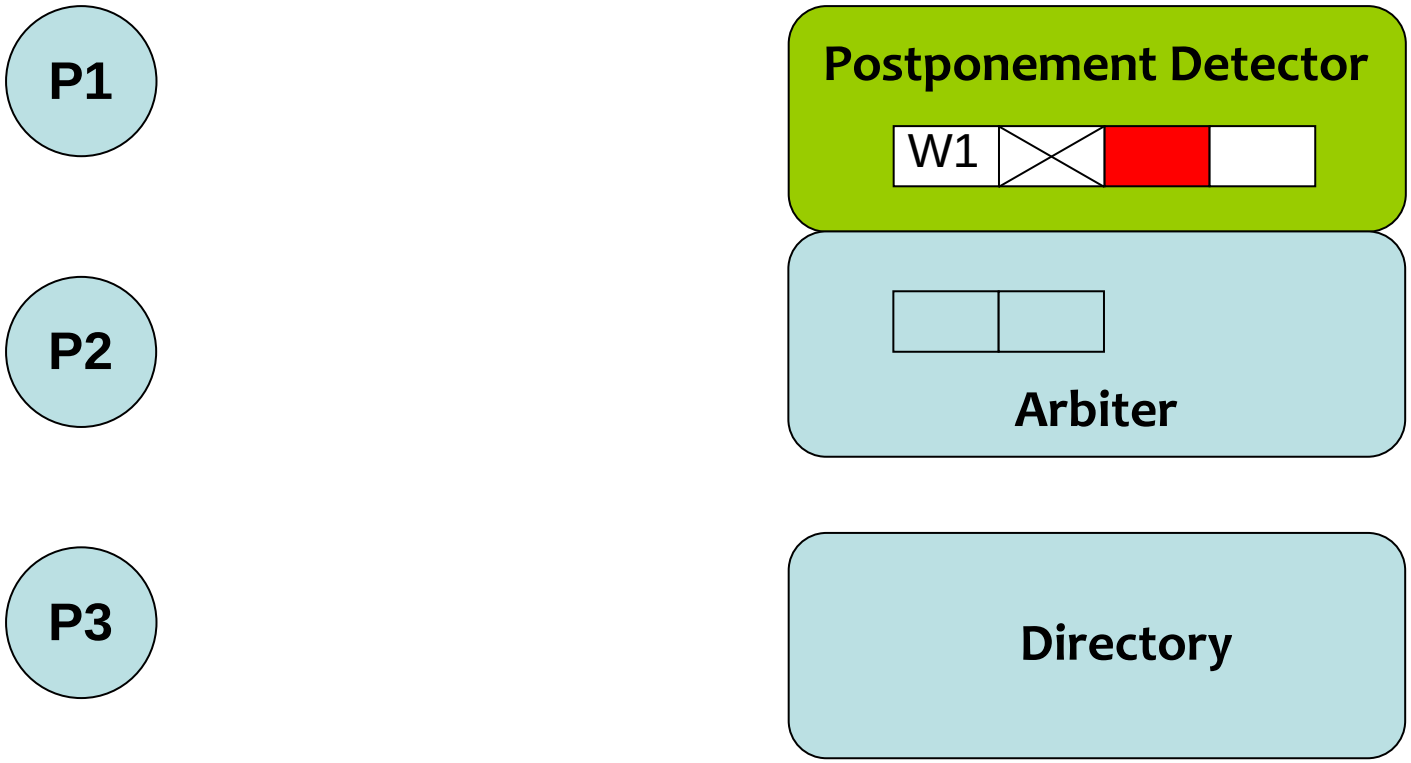
Complete Commit/Postponed Process



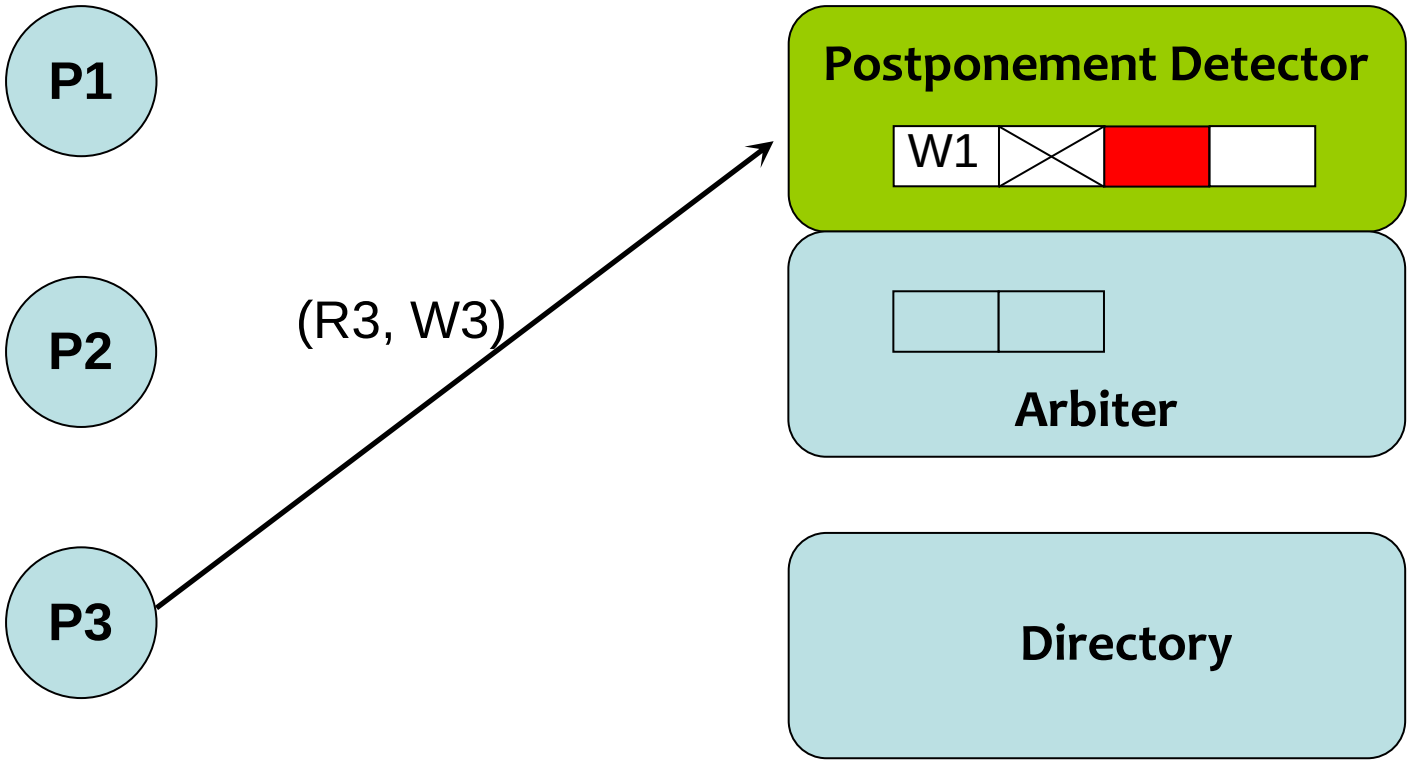
Complete Commit/Postponed Process



Complete Commit/Postponed Process

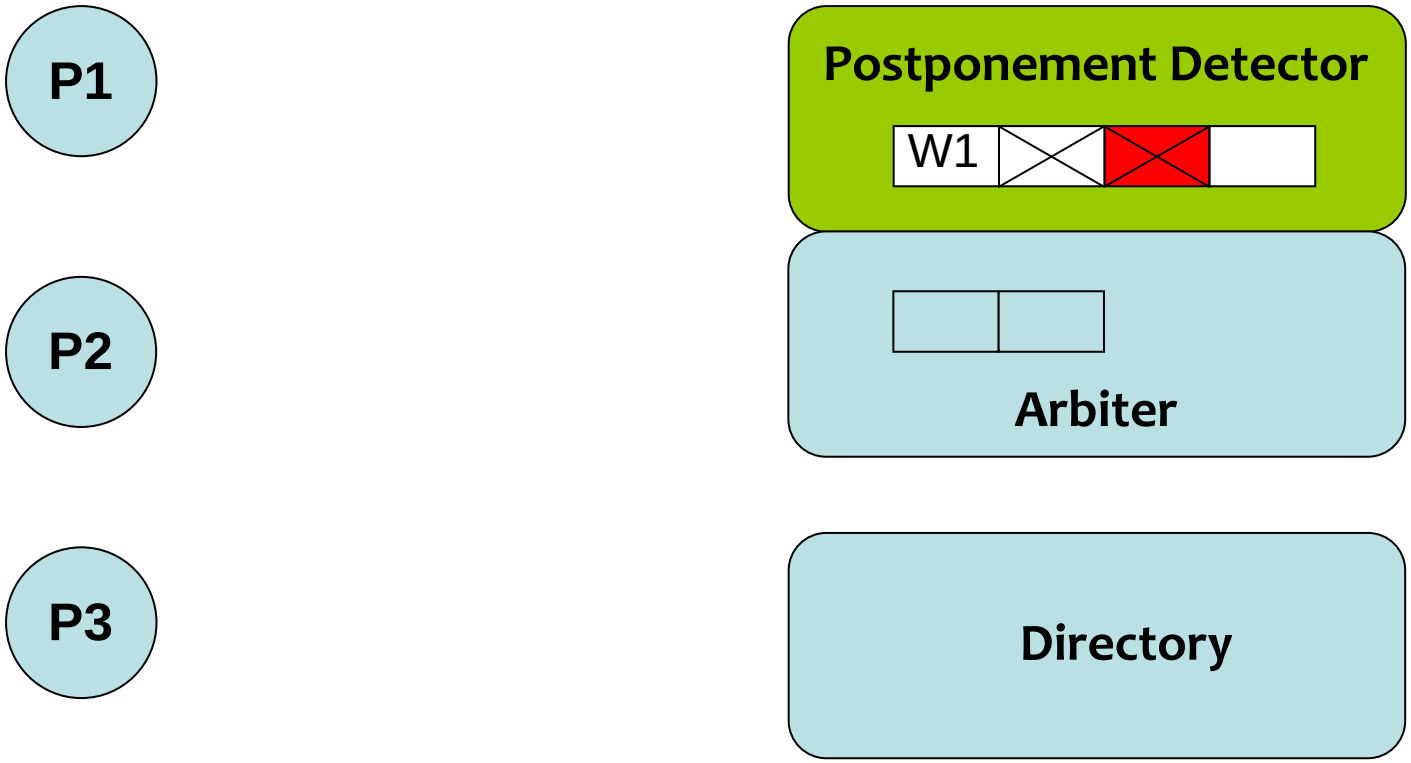


Complete Commit/Postponed Process



W1 ∩ (R3, W3) =Ø **?**

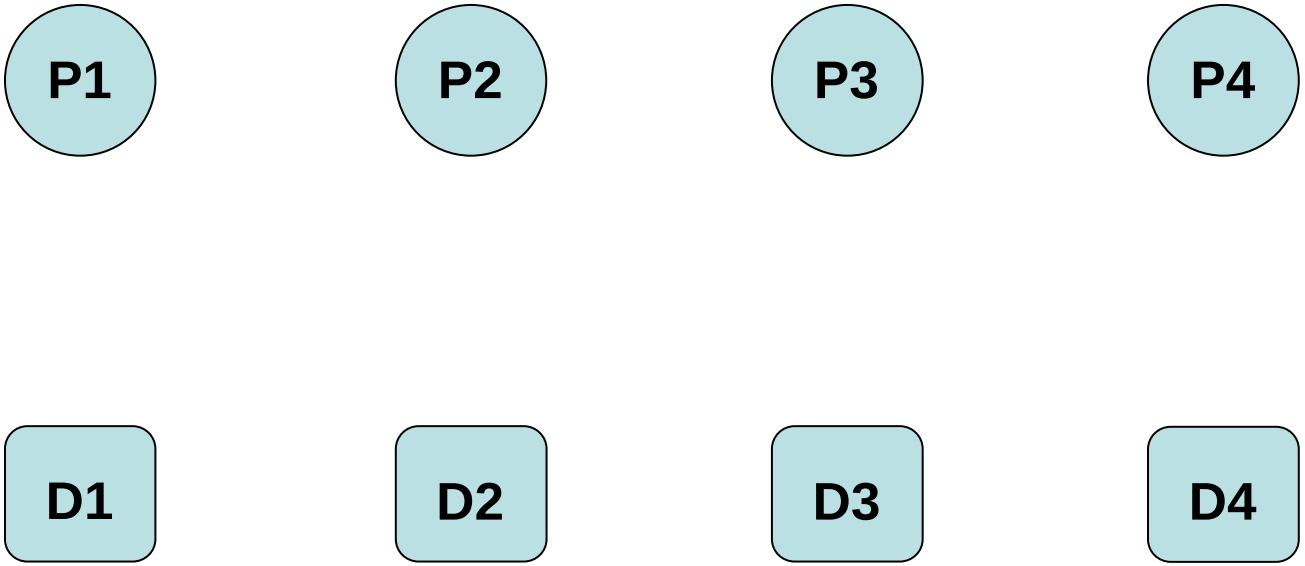
Complete Commit/Postponed Process



W1 ∩ (R3, W3) =Ø **?** NO!

Distributed Commit Design

Distributed P-detectors, each handle a range of addresses



See the paper for details..

Issues in Deterministic Chunk Execution

- Fix the chunk size
- Use modest sized chunk and enough buffer
  - Avoid cache overflow (nondeterministic)
- Process interrupts at chunk boundaries
- On an exception, re-execute the chunk only when the token arrives

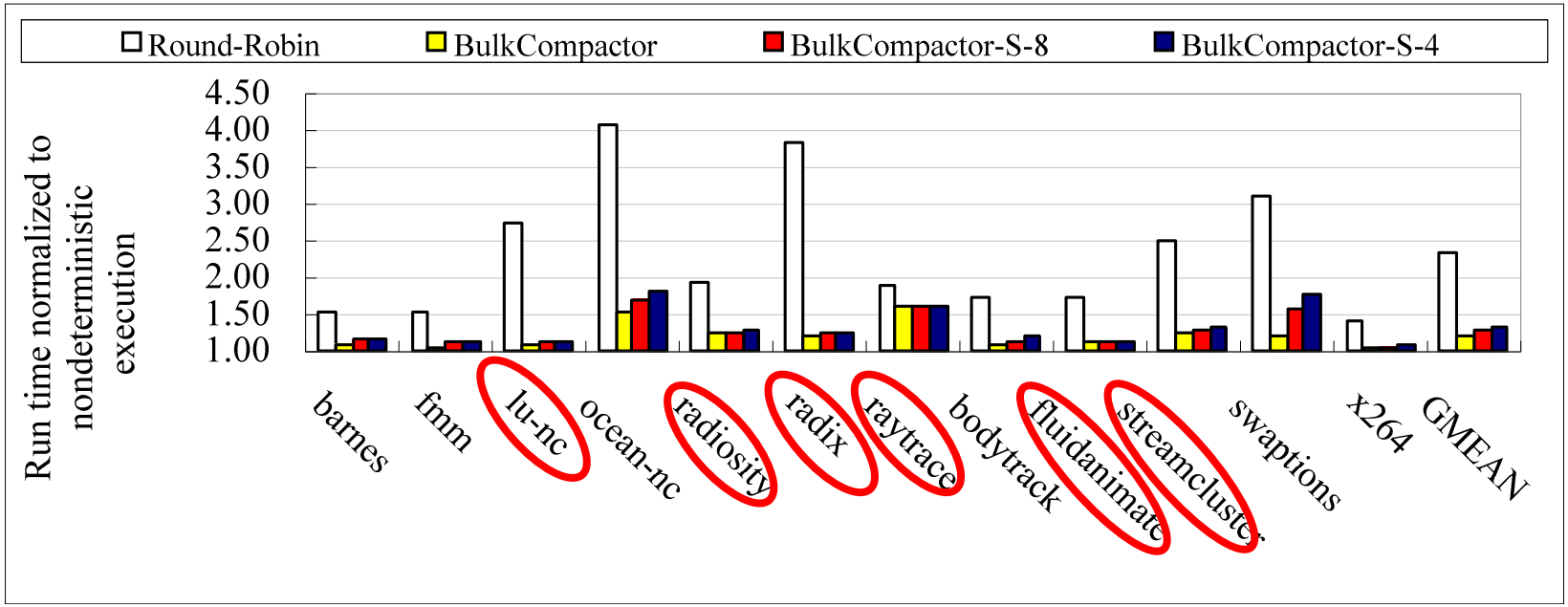
Outline

- BulkCompactor & BulkCompactor-S
- Design of a Centralized & Distributed Commit Machine
- Evaluation

Evaluation

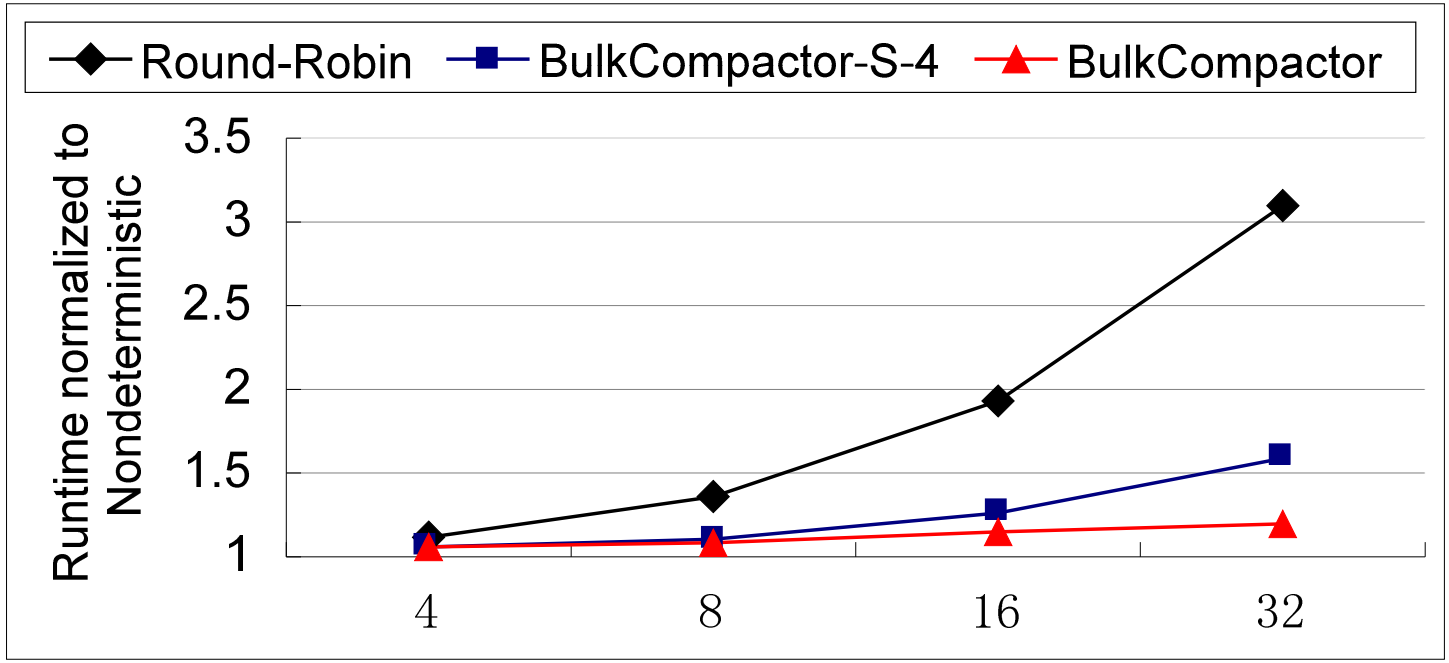
- Model the Bulk Multicore with 4 to 32 processors
  - Architecture uses chunks with lazy commit [CACM'09]
  - 2k instruction chunk size
  - Model Round-robin, BulkCompactor, BulkCompactor-S4, BulkCompactor-S8 commit
- Simulation based on PIN
- Apps from PARSEC and SPLASH2
  - Run to completion

Runtime Normalized to Nondeterministic System



- Round-robin incurs high overhead (~2.5x)
- BulkCompactor only slightly slower than nondeterministic system
- BulkCompactor-S4 & S8 perform like BulkCompactor for many apps
  - conflict locality!

Performance Scalability (fmm)



- Overhead of Round-robin grows fast
- BulkCompactor grows almost linearly

Also in the paper..

- Overhead analysis for each scheme
- Handling starvation and fairness
- Conflict behavior characterization
- Sensitivity to the chunk size
- Sensitivity to the token passing latency

Conclusions

- Presented BulkCompactor: Conflict-Aware Deterministic Commit System
  - Avoids transitive squash delay
  - Performs comparably to nondeterministic chunk-based system
  - Scalable
- Designed centralized and distributed commit machine
- Results for 32 processor runs
  - Only 22% execution overhead over a nondeterministic system

BulkCompactor: Optimized Deterministic Execution via Conflict-Aware Commit of Atomic Blocks

Yuelu Duan, Xing Zhou, Wonsun Ahn, Josep Torrellas

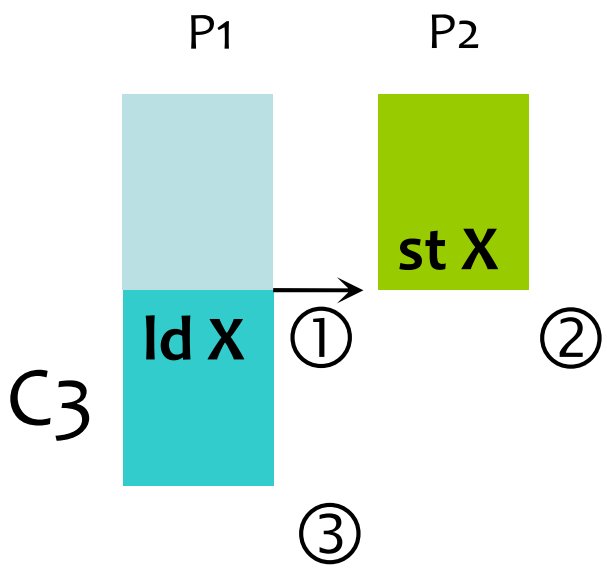
University of Illinois at Urbana-Champaign
http://iacoma.cs.uiuc.edu

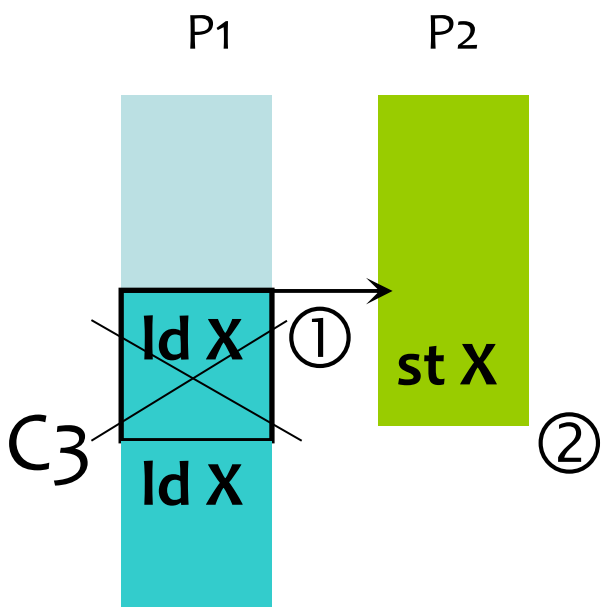
HPCA 2012, New Orleans, LA

i-acoma group

Backup Slides

Squash is not Deterministic (2)





C3 is neither squashed nor postponed

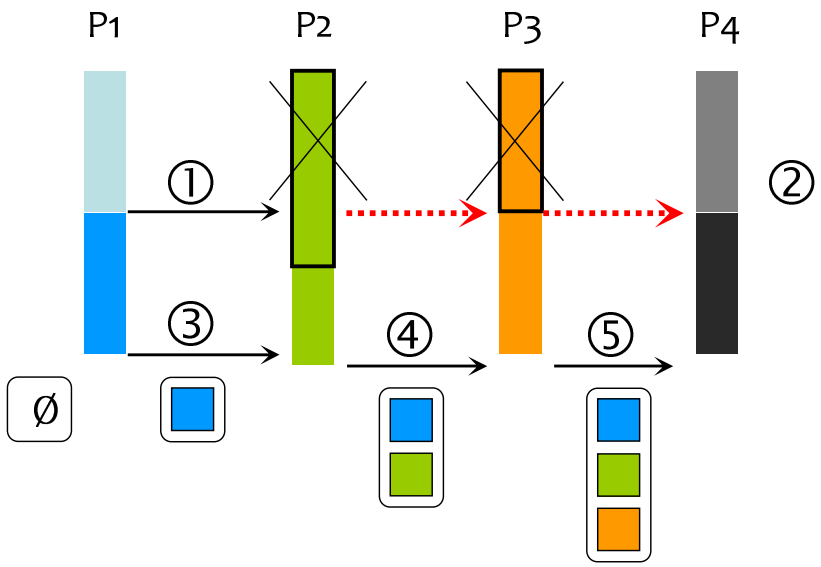
C3 is squashed, but **must not** be postponed

Rules of Deterministic Postponement

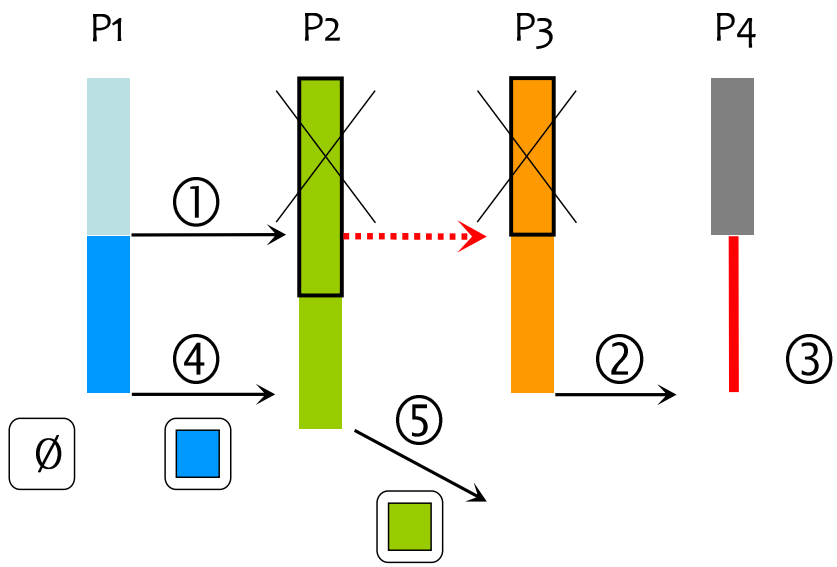
1. *Always* postpone a chunk if it conflicts with previously committed chunks in the same round

2. *Never* postpone a chunk if it conflicts with committed chunks from earlier round

BulkCompactor vs. BulkCompactor-S



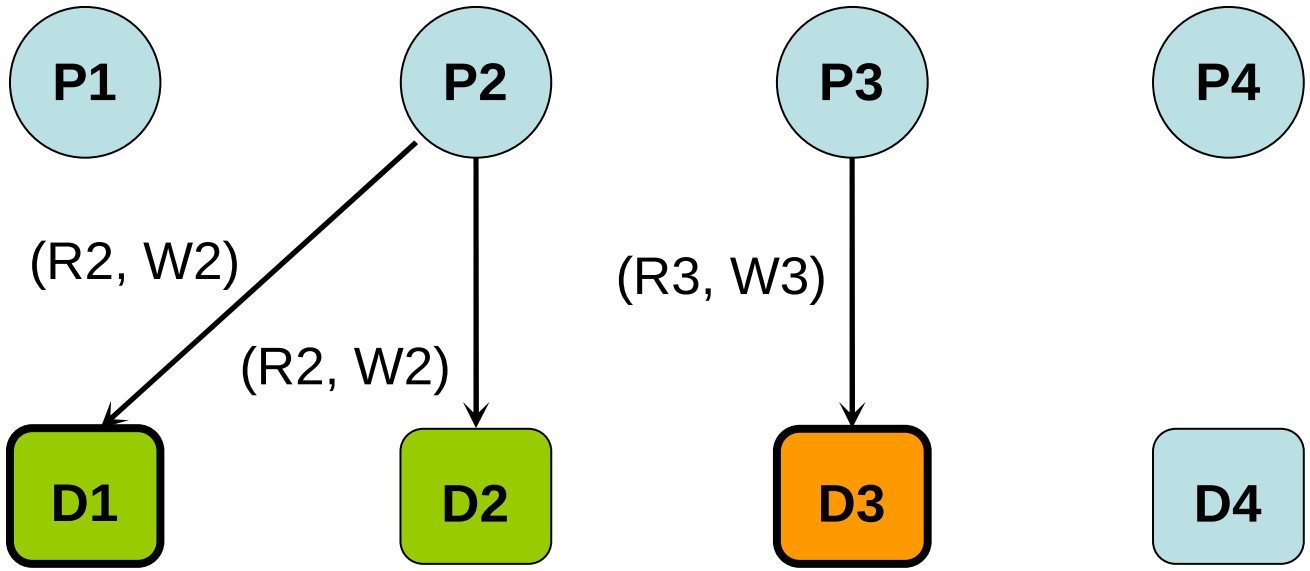
BulkCompactor



BulkCompctor-S (M = 1)

Signatures Sent in Parallel

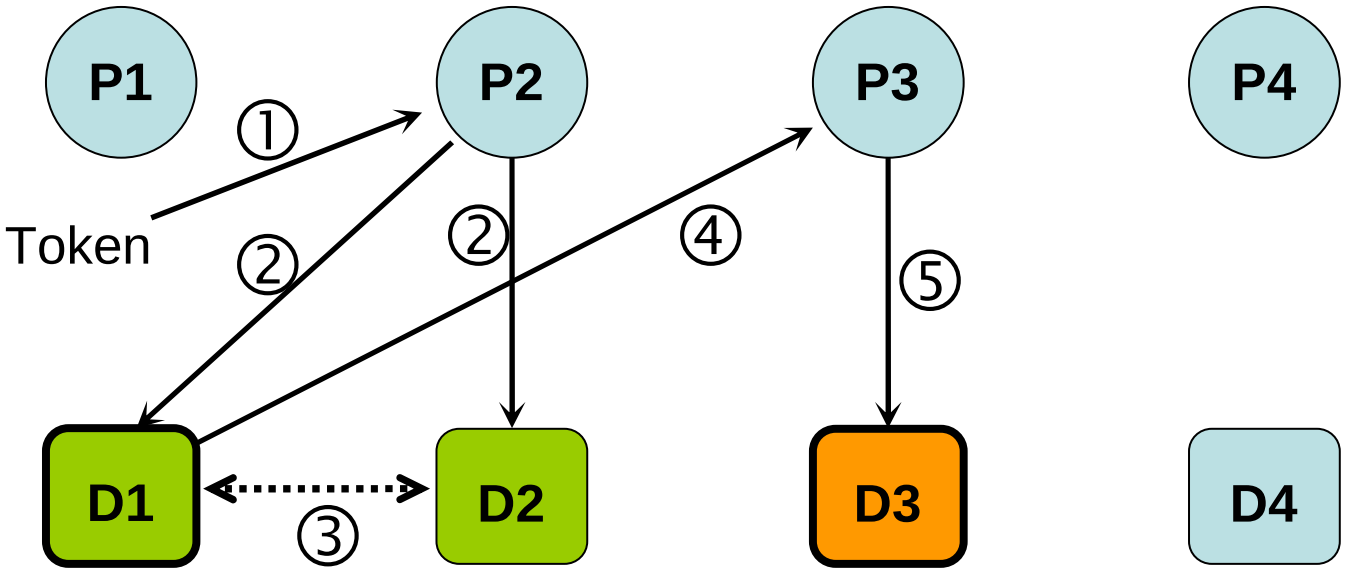
Distributed P-detectors, each handle a range of addresses



Signatures sent to *relevant* P-detectors, with a *leader* [MICRO'10]

Passing Commit Token

Distributed P-detectors, each handle a range of addresses



Token (Bit vector): indicates each Pi being postponed or not

Deterministic Conflict Detection

- Signature operations can introduce false positive conflicts
  - Use virtual address to encode signatures
    - Identical across program runs
  - Use same encoding algorithm across machines
- In distributed BulkCompactor:
  - Index p-detectors/directory deterministically