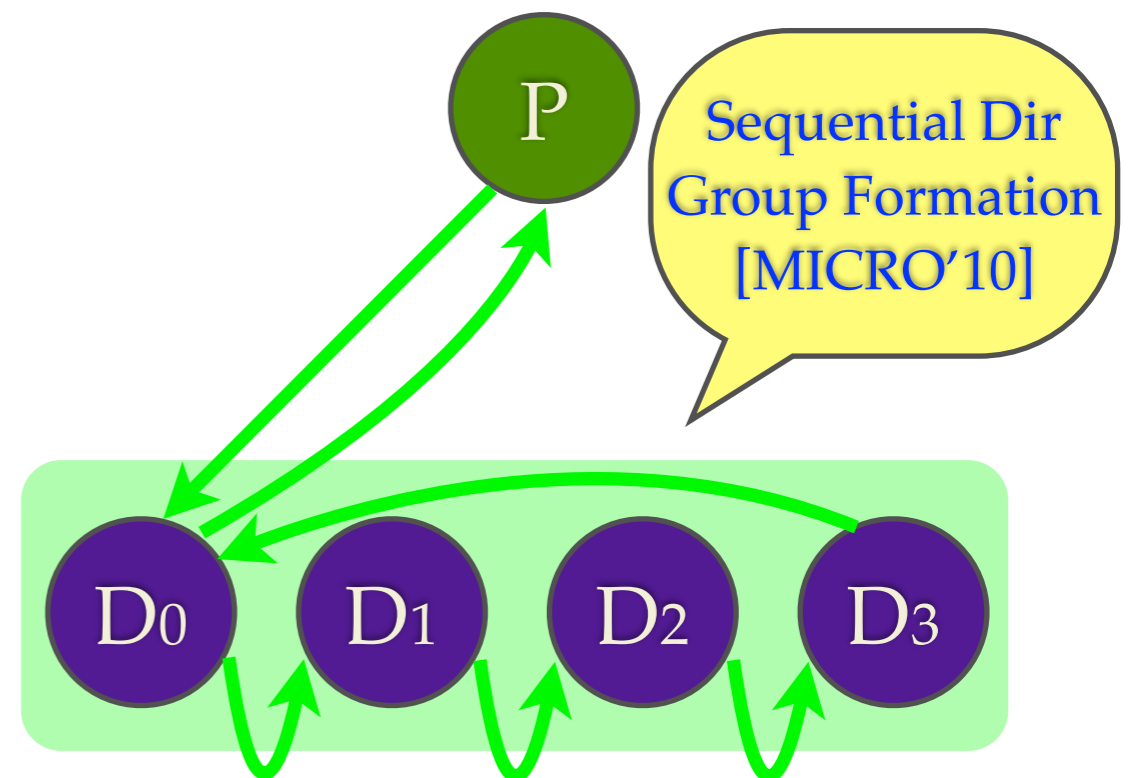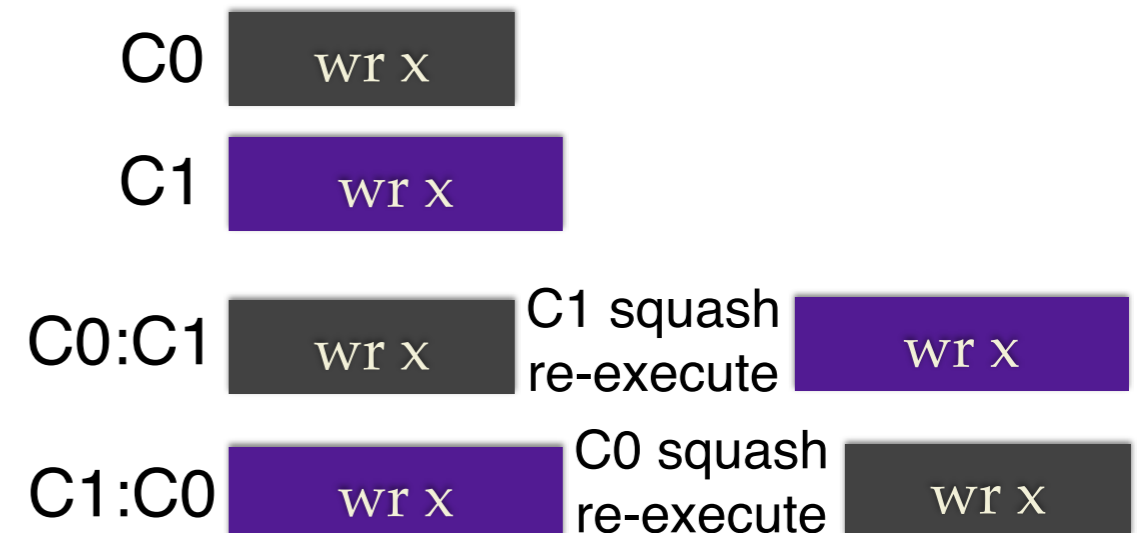# BulkCommit: Scalable and Fast Commit of Atomic Blocks in a Lazy Multiprocessor Environment
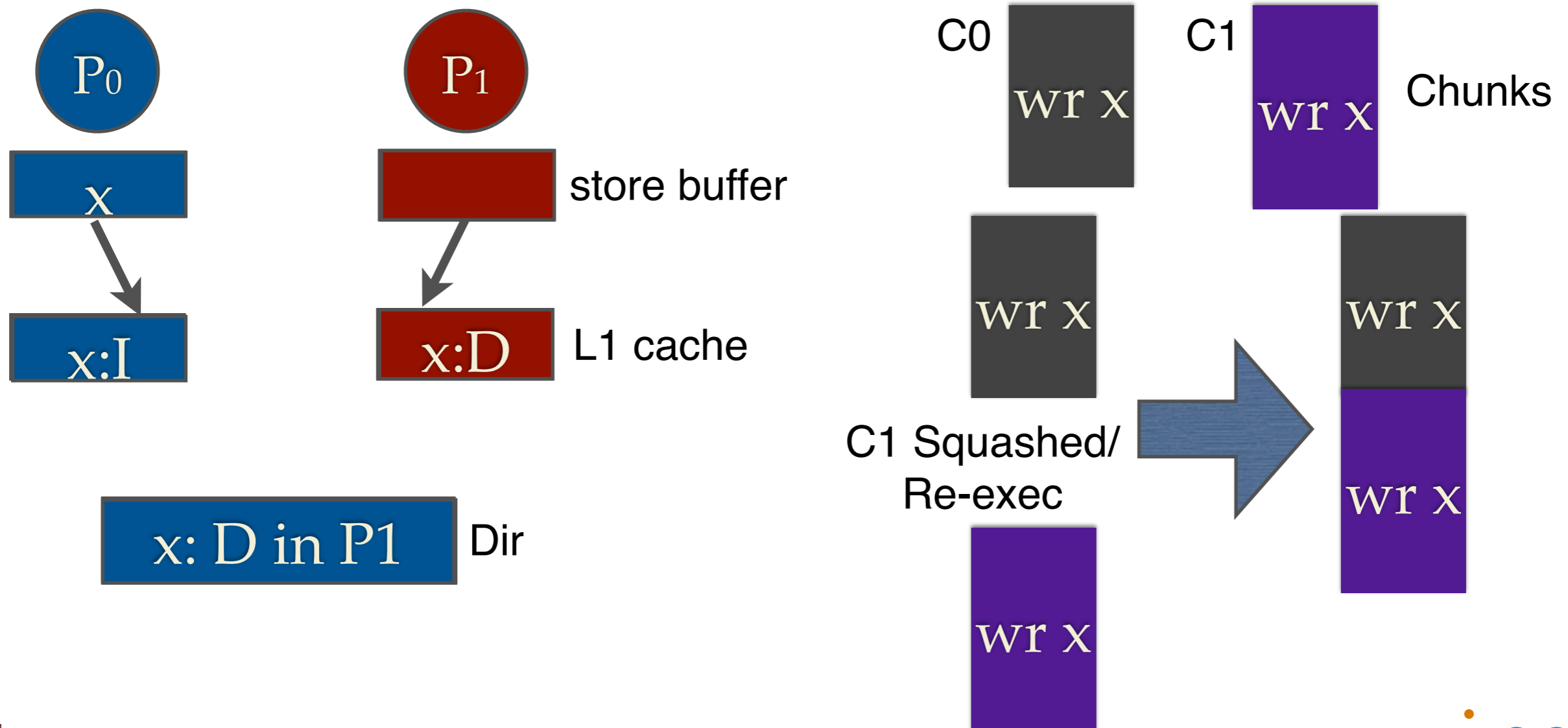
Xuehai Qian, Josep Torrellas (UIUC)

Benjamin Sahelices (Univ of Valladolid) and Depei Qian (Beihang Univ.)

- Problem:

  - Current atomic block (chunk) execution incurs unnecessary squashes

  - Atomic block commit operation in a lazy environment has sequential bottlenecks

- Our solution:

  - IntelliSquash: no squash on WAW-only conflict

  - IntelliCommit: parallel directory group formation

C0   wr x

C1   wr x

C0:C1   wr x   C1 squash re-execute   wr x

C1:C0   wr x   C0 squash re-execute   wr x

P

Sequential Dir Group Formation [MICRO'10]

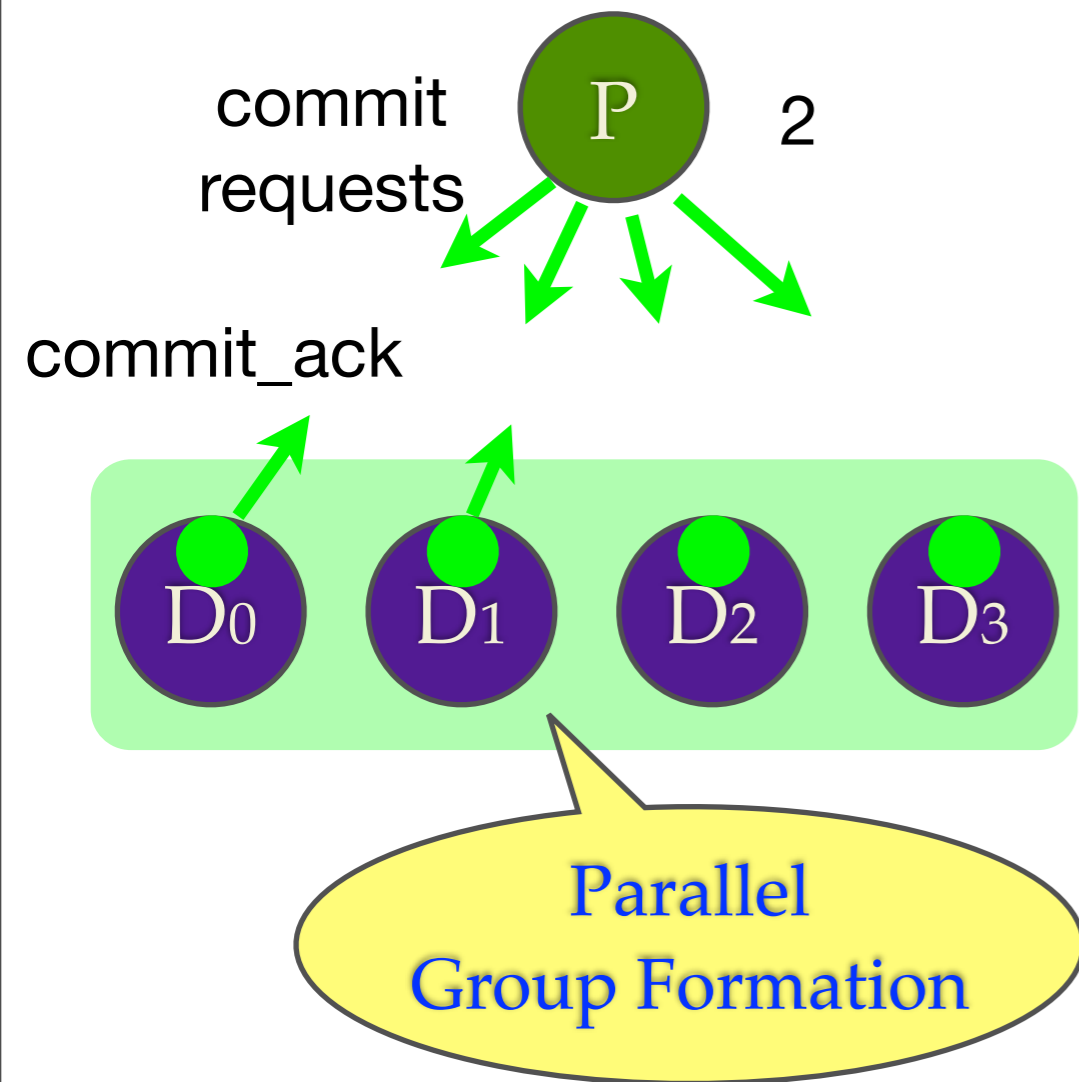D_0 D_1 D_2 D_3

i-acoma group

IntelliSquash: No Squash on WAW-only Conflict

- Insight: WAW is a name dependence. It does not break semantic atomicity

- Similarity with two conflict stores from two processors

- If two chunks only have WAW conflicts, IntelliSquash serializes them without squash

P_0

x

x:I

P_1

store buffer

x:D   L1 cache

x: D in P1   Dir

C0   C1

wr x   wr x   Chunks

wr x

C1 Squashed/ Re-exec

wr x

wr x

wr x

i-acoma group

IntelliCommit: Parallel Directory Group Formation

commit
requests

P   2

commit_ack

D0   D1   D2   D3

Parallel
Group Formation

- On chunk commit:

  - Processor sends commit requests to all the relevant directory modules

  - Directory module receives commit request:

    - Locks the memory lines

    - Responds with commit_ack

  - Processor counts the number of commit_acks received

  - Processor sends commit_confirm when it receives the expected number of commit_acks

- Challenge: resolving conflicts from two processors

- ChunkSort: ordering all the conflicting chunks in the same order in all relevant directories by **preemption**

i-acoma group