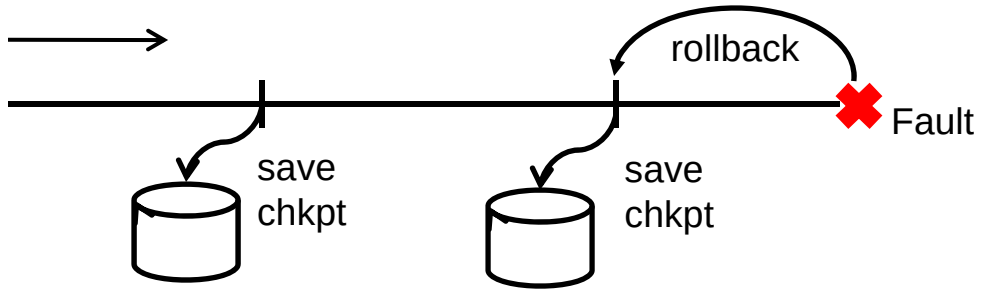
Rebound: Scalable Checkpointing for Coherent Shared Memory

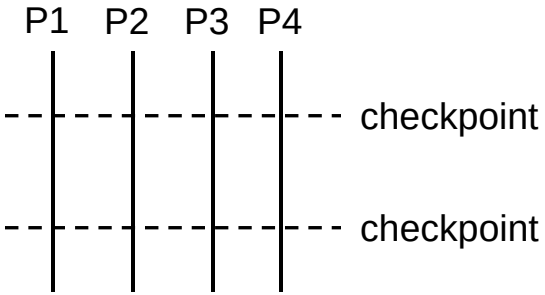
Rishi Agarwal, Pranav Garg, and Josep Torrellas

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

Checkpointing in Shared-Memory MPs



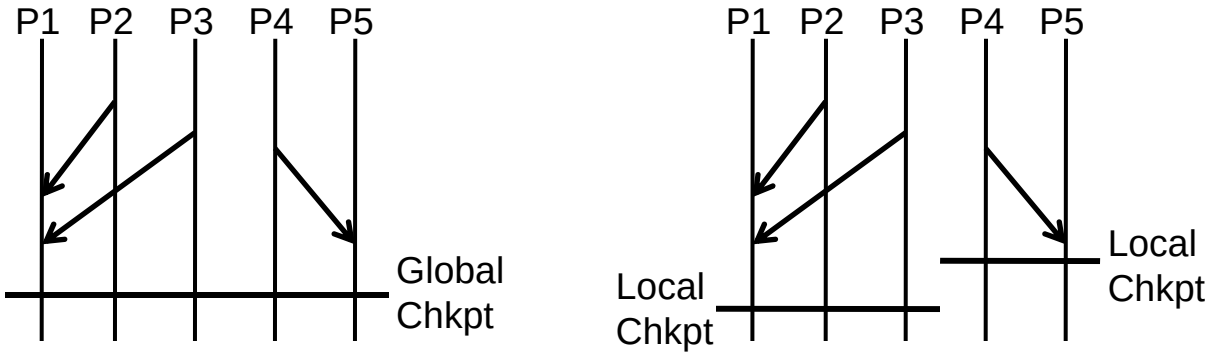
- HW-based schemes for small CMPs use Global checkpointing
  - All procs participate in system-wide checkpoints



- Global checkpointing is not scalable
  - Synchronization, bursty movement of data, loss in rollback…

Alternative: Coordinated Local Checkpointing

- Idea: threads coordinate their checkpointing in groups
- Rationale:
  - Faults propagate only through communication
  - Interleaving between non-comm. threads is irrelevant



+ Scalable: Checkpoint and rollback in processor groups

– Complexity: Record inter-thread dependences dynamically.

Contributions

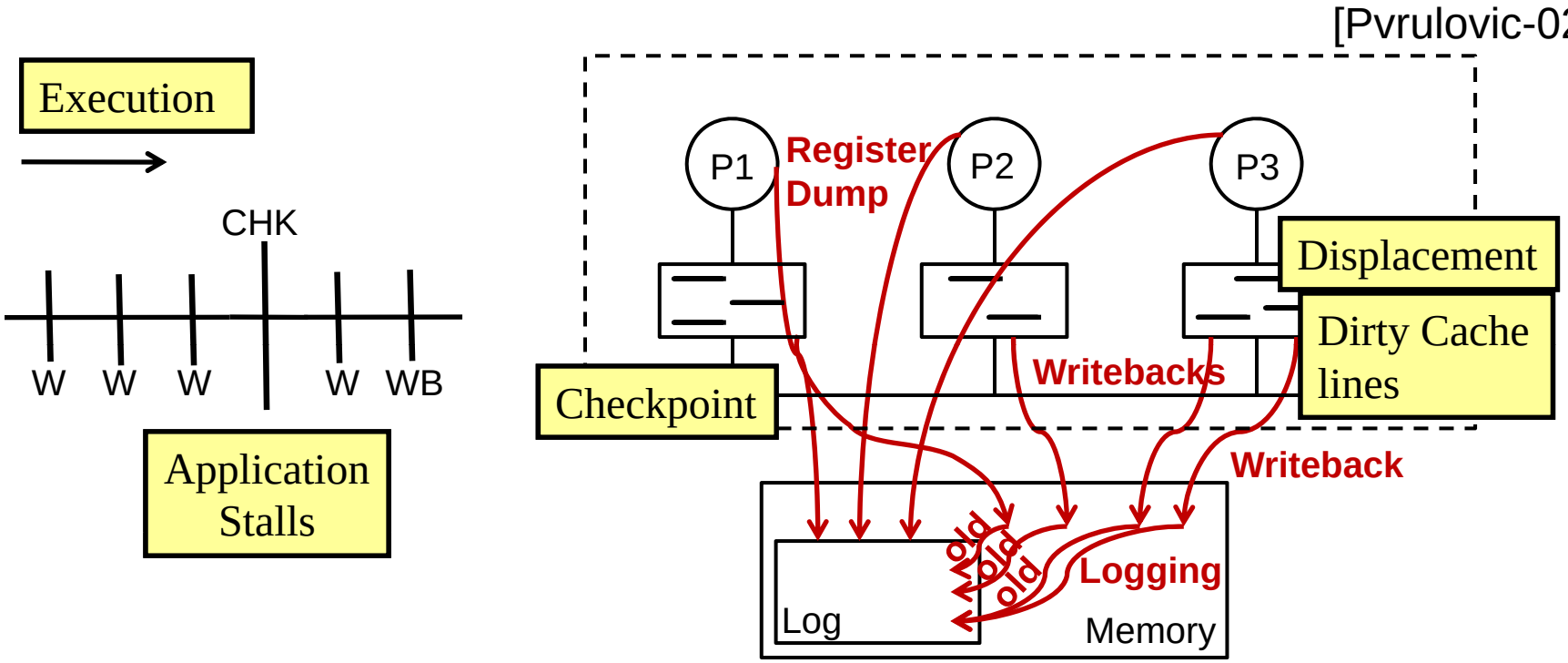
Rebound: First HW-based scheme for scalable, coordinated local checkpointing in coherent shared-memory

- Leverages directory protocol to track inter-thread deps.

- Opts to boost checkpointing efficiency:
  - Delaying write-back of data to safe memory at checkpoints
  - Supporting multiple checkpoints
  - Optimizing checkpointing at barrier synchronization

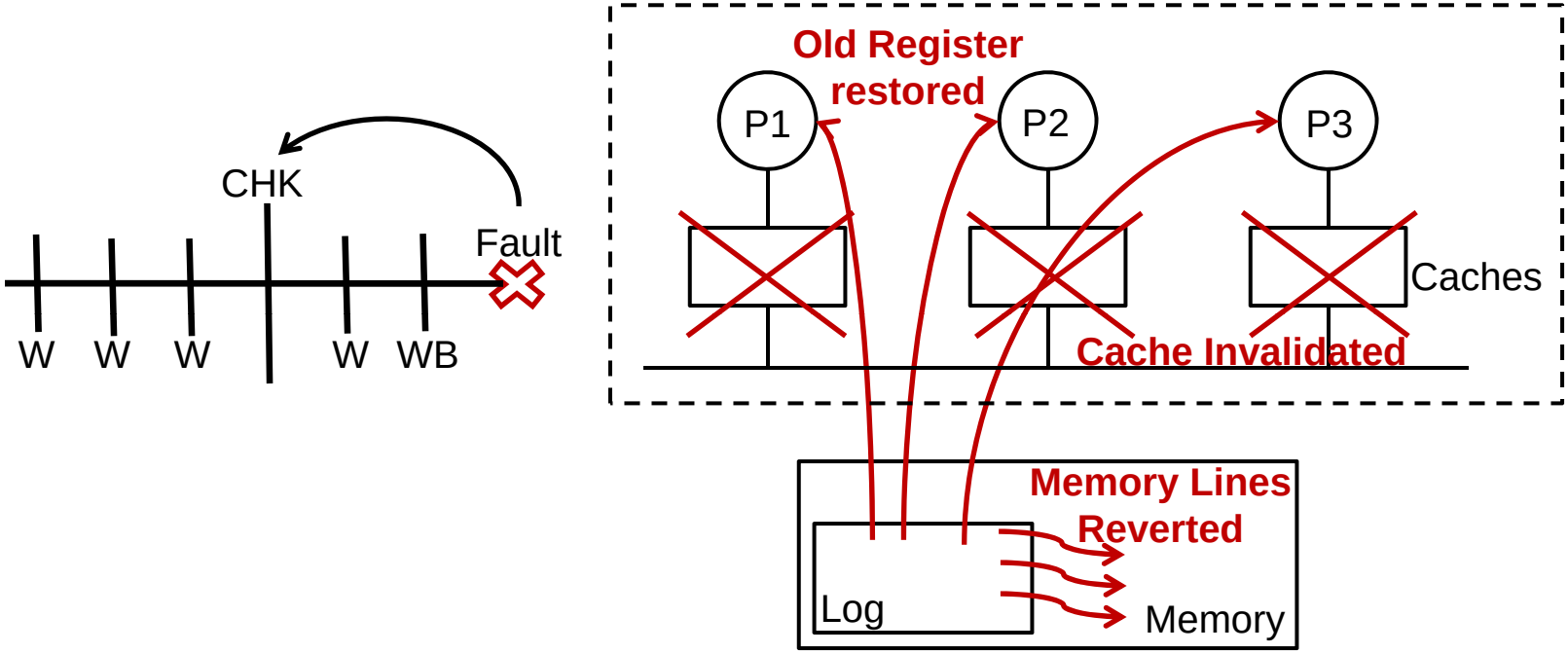
- Avg. performance overhead for 64 procs: 2%
  - Compared to 15% for global checkpointing

Background: In-Memory Checkpt with ReVive



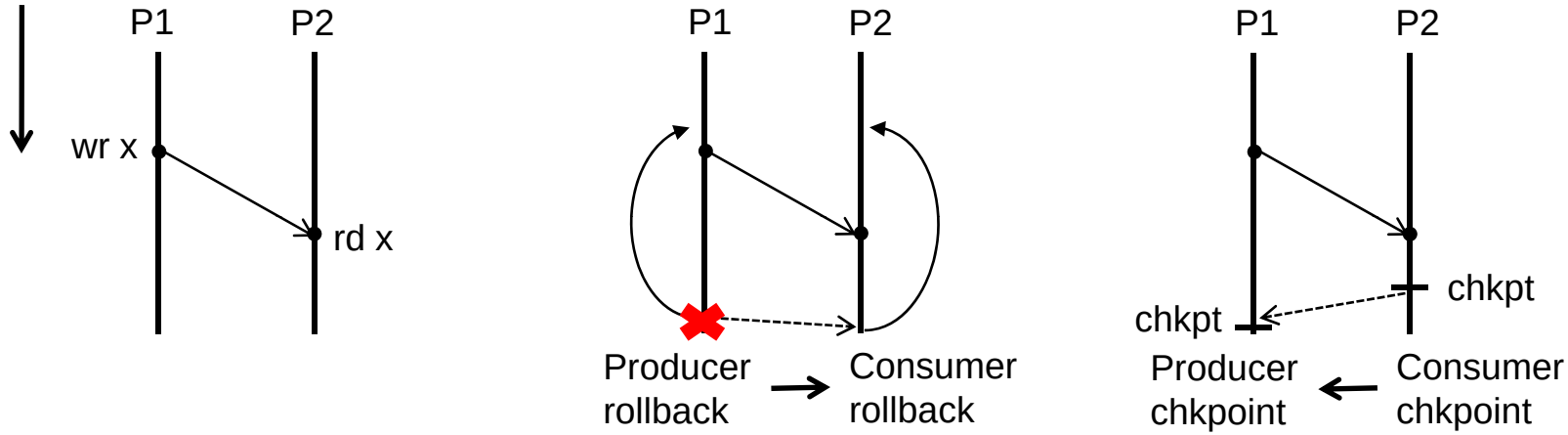
Background: In-Memory Checkpt with ReVive

[Pvrulovic-02]



~~Global Broadcast protocol~~ → Local Coordinated Scalable protocol

Coordinated Local Checkpointing Rules

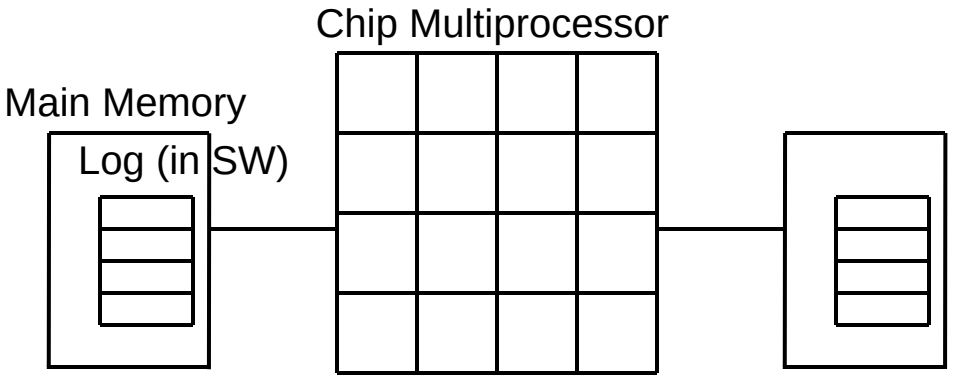


P checkpoints ➔ P's producers checkpoint

P rolls back ➔ P's consumers rollback

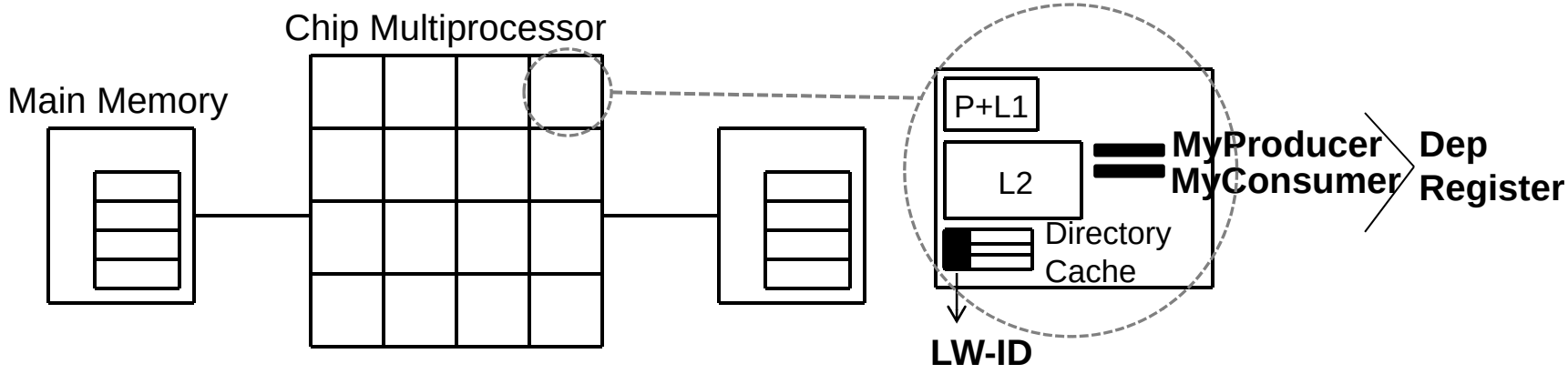
- Banatre et al. used Coordinated Local checkpointing for bus-based machines [Banatre96]

Rebound Fault Model

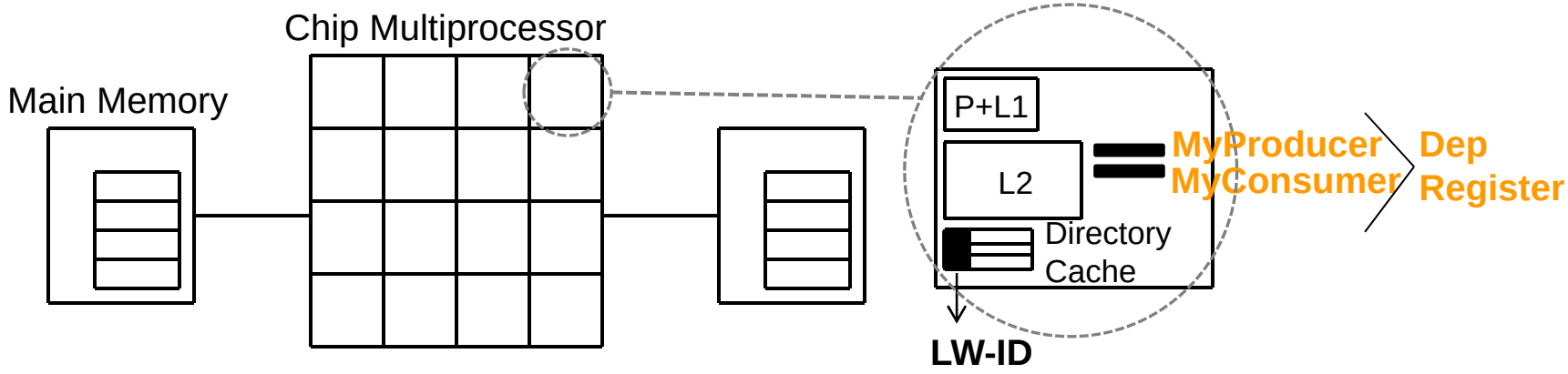


- Any part of the chip can suffer transient or permanent faults.
- A fault can occur even during checkpointing
- Off-chip memory and logs suffer no fault on their own (e.g. NVM)
- Fault detection outside our scope:
  - Fault detection latency has upper-bound of L cycles

Rebound Architecture

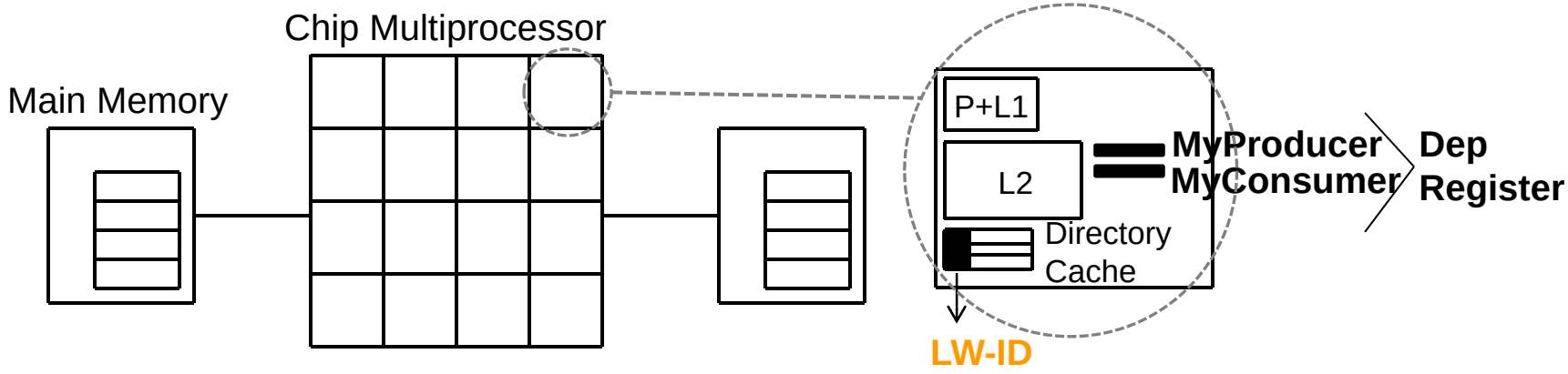


Rebound Architecture



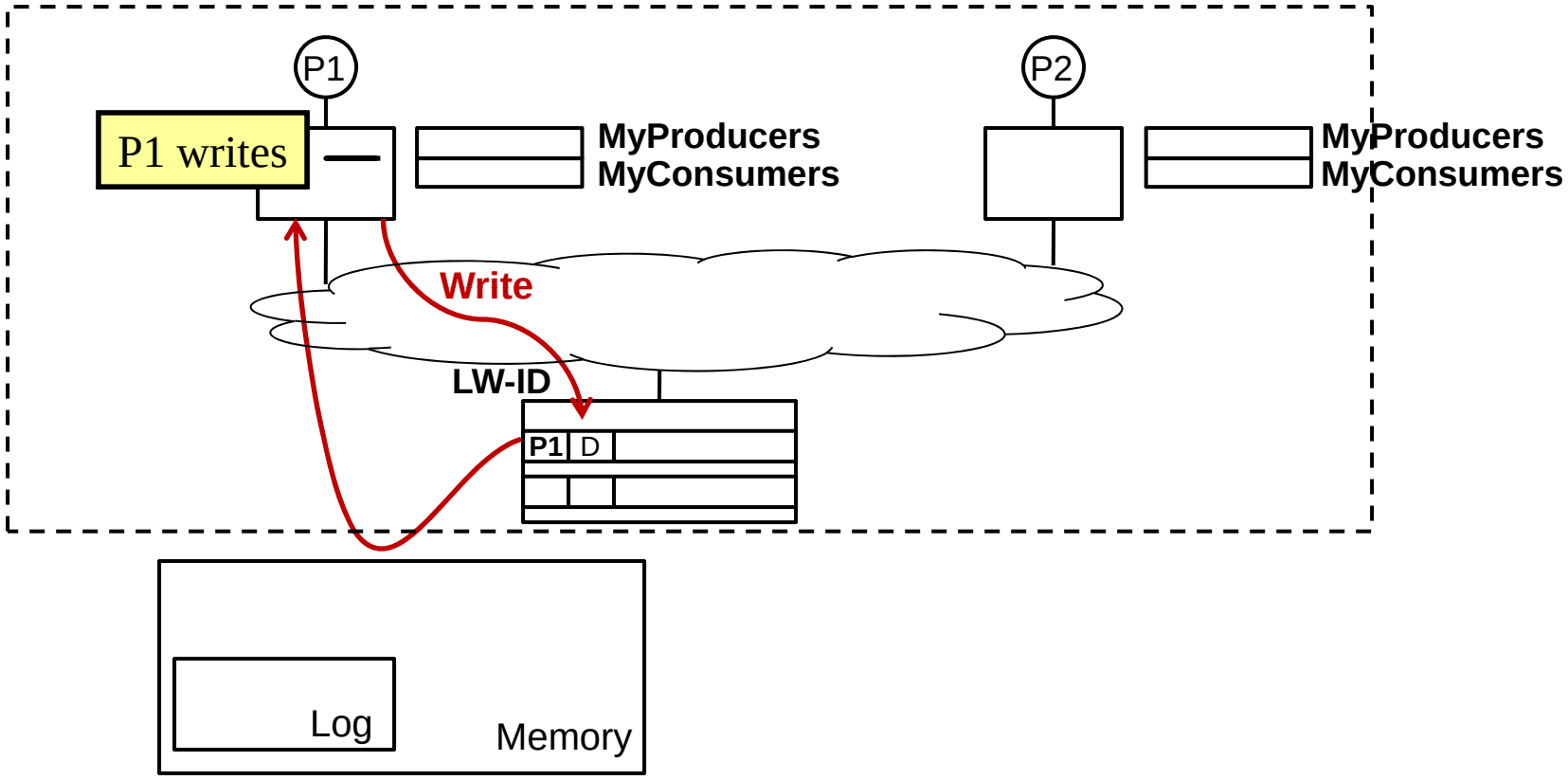
- Dependence (Dep) registers in the L2 cache controller:
  - **MyProducers** : bitmap of proc. that produced data consumed by the local proc.
  - **MyConsumers** : bitmap of proc. that consumed data produced by the local proc.

Rebound Architecture



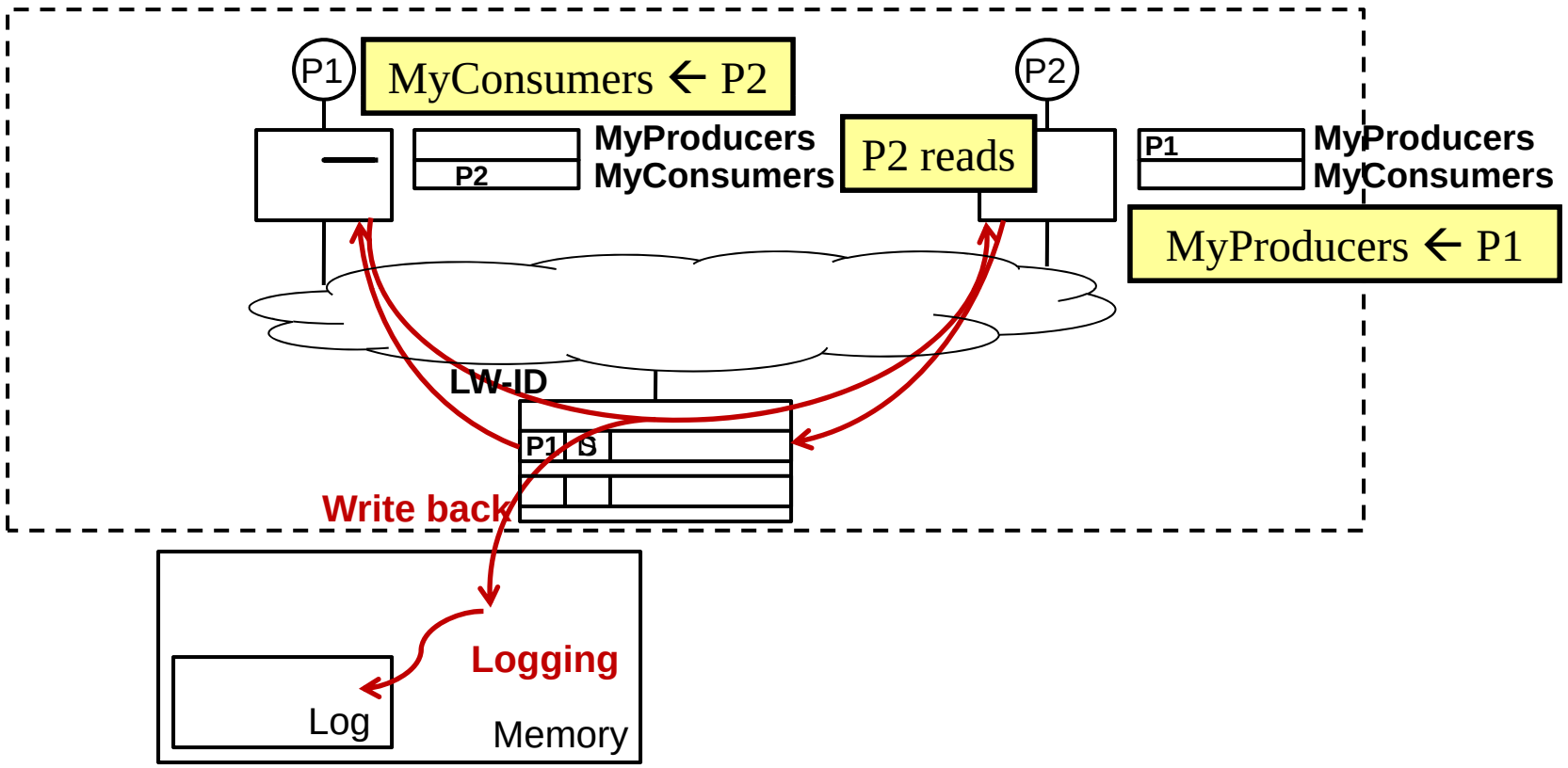
- Dependence (Dep) registers in the L2 cache controller:
  - **MyProducers** : bitmap of proc. that produced data consumed by the local proc.
  - **MyConsumers** : bitmap of proc. that consumed data produced by the local proc.
- Processor ID in each directory entry:
  - **LW-ID** : last writer to the line in the current checkpoint interval.

Recording Inter-Thread Dependences



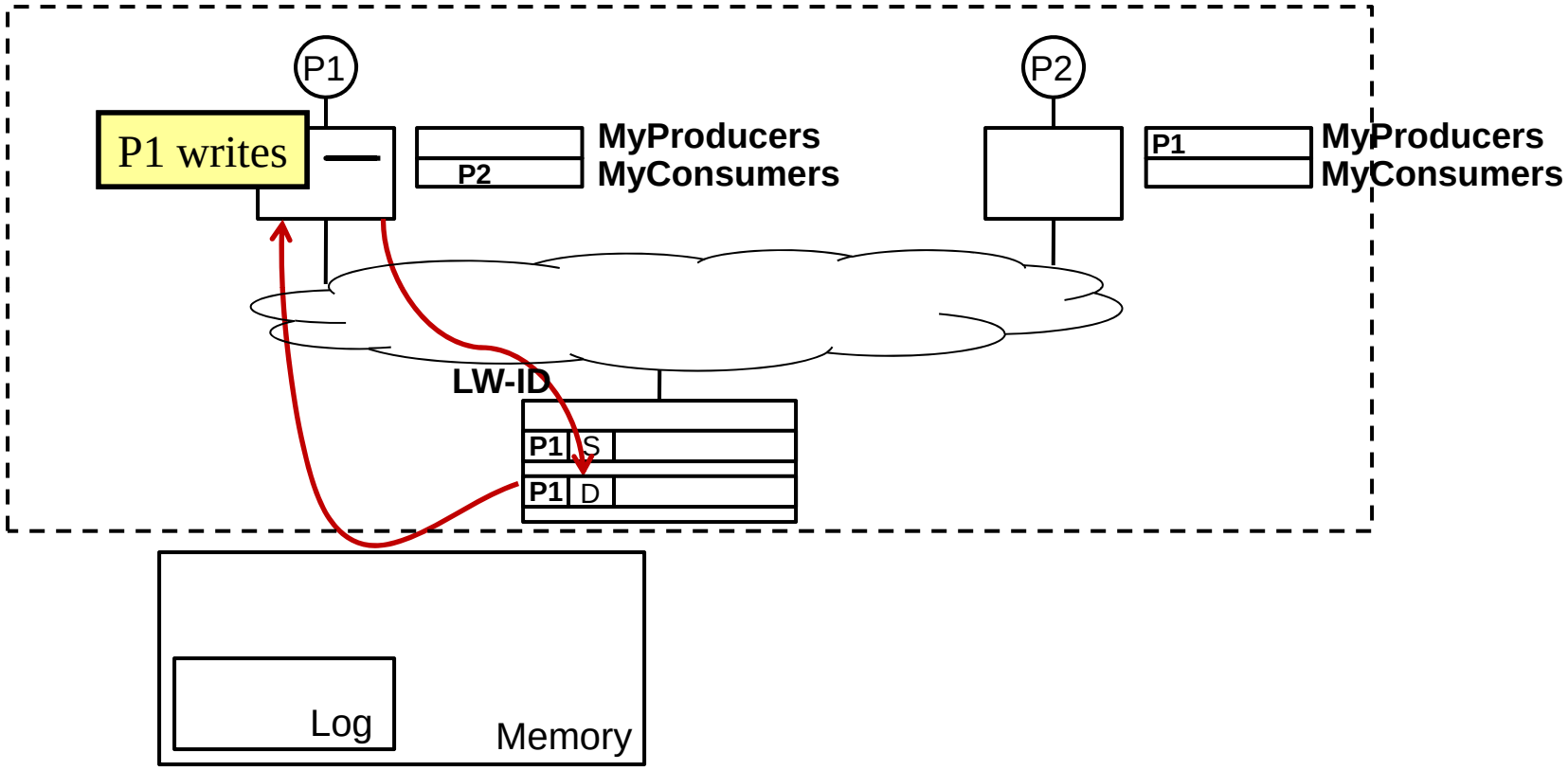
Assume MESI protocol

Recording Inter-Thread Dependences



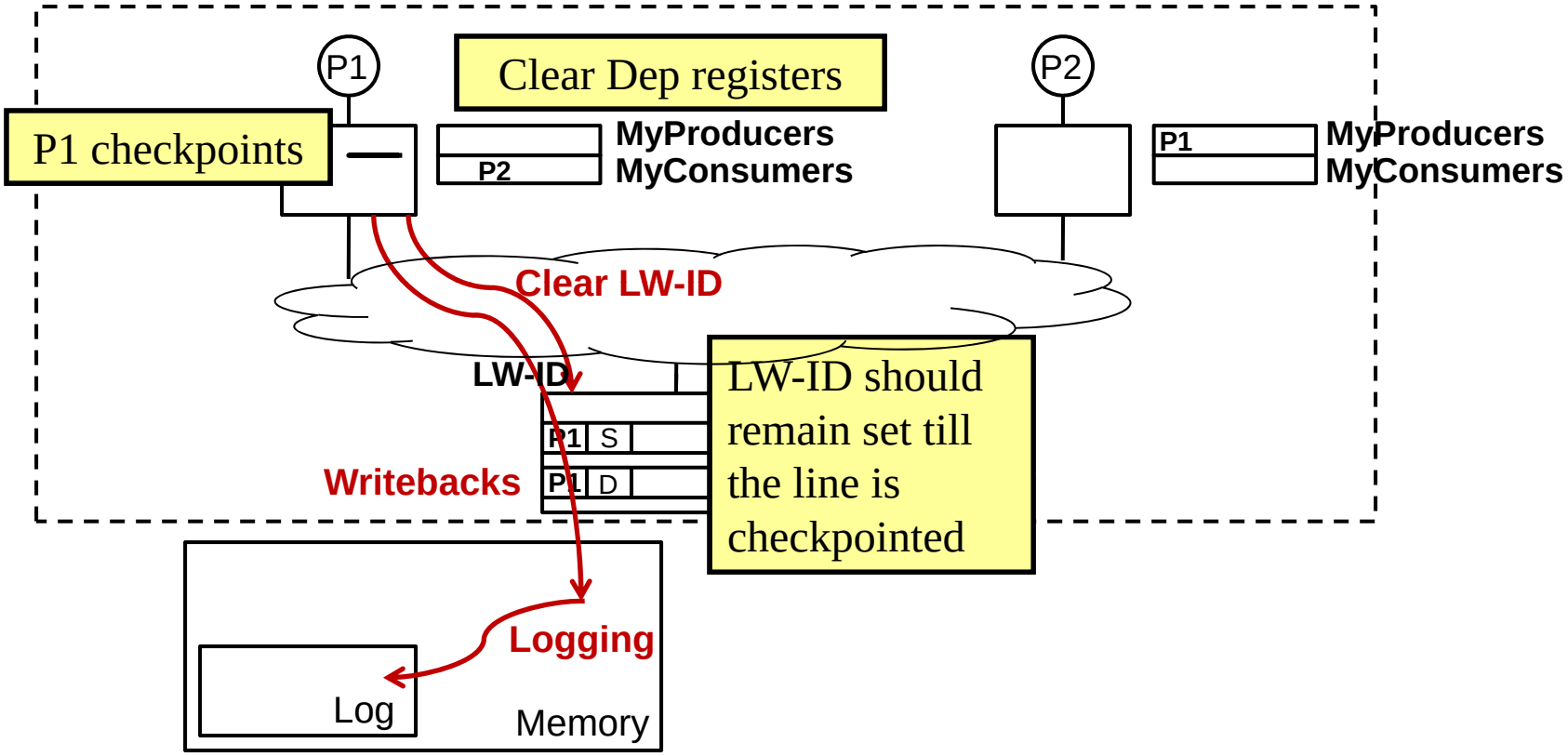
Assume MESI protocol

Recording Inter-Thread Dependences



Assume MESI protocol

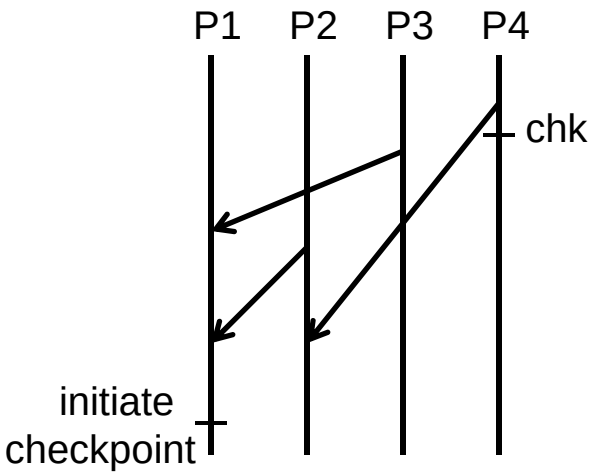
Recording Inter-Thread Dependences

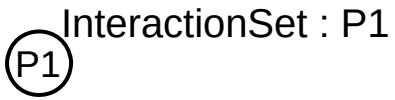


Assume MESI protocol

Distributed Checkpointing Protocol in SW

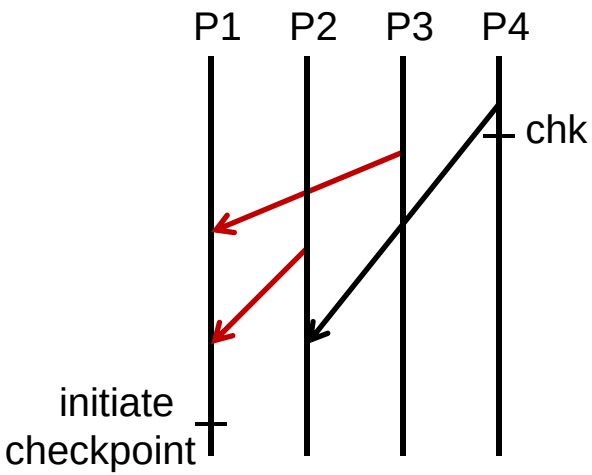
- Interaction Set [P_i]: set of producer processors (transitively) for P_i
  - Built using MyProducers

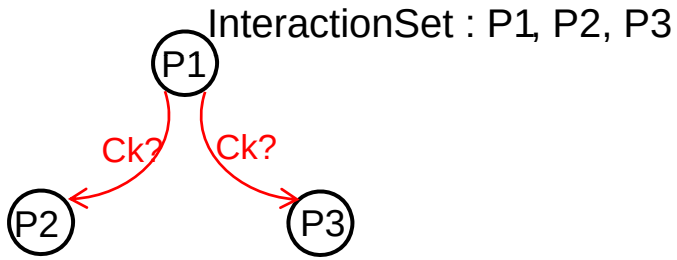




Distributed Checkpointing Protocol in SW

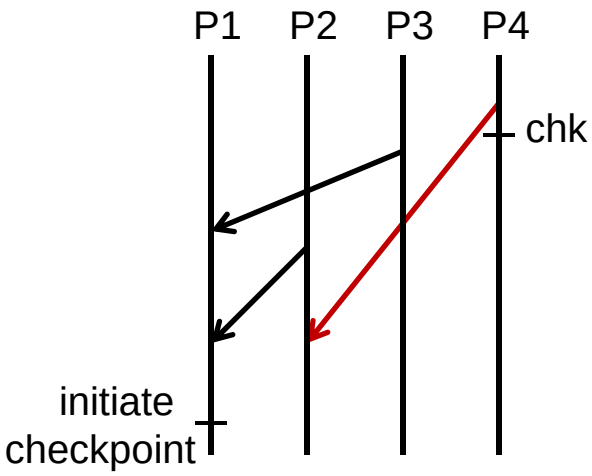
- Interaction Set [P_i]: set of producer processors (transitively) for P_i
  - Built using MyProducers

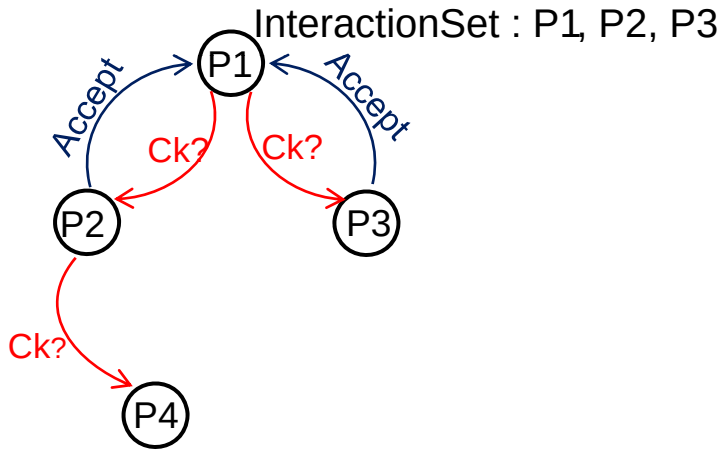




Distributed Checkpointing Protocol in SW

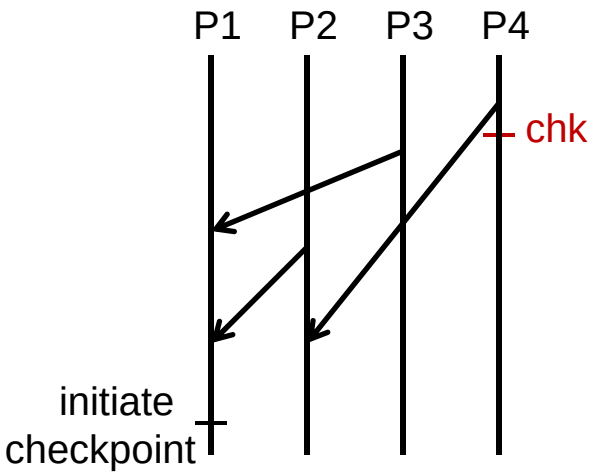
- Interaction Set [P_i]: set of producer processors (transitively) for P_i
  - Built using MyProducers

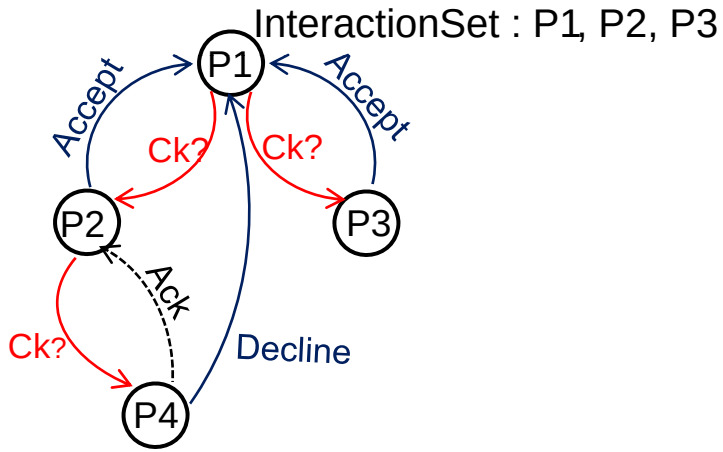




Distributed Checkpointing Protocol in SW

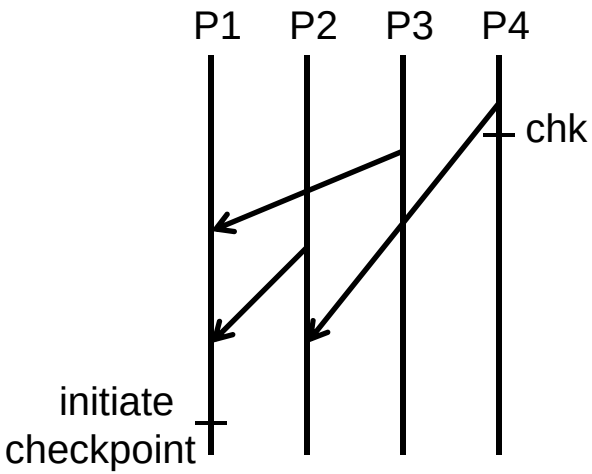
- Interaction Set [P_i]: set of producer processors (transitively) for P_i
  - Built using MyProducers

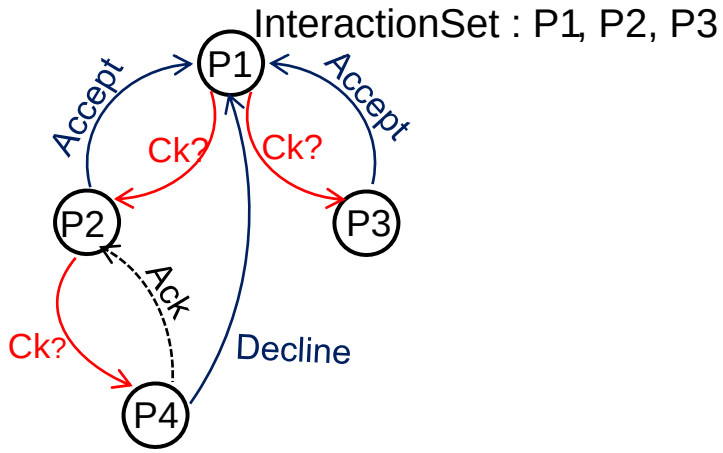




Distributed Checkpointing Protocol in SW

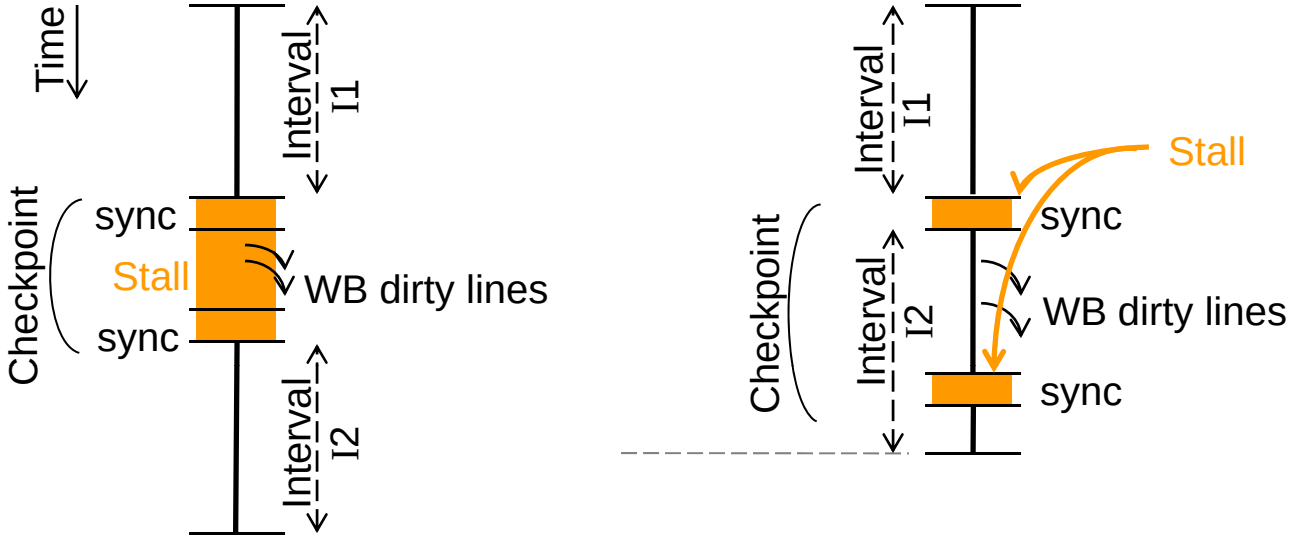
- Interaction Set [P_i]: set of producer processors (transitively) for P_i
  - Built using MyProducers





- Rollback handled similarly using MyConsumers

Optimization1 : Delayed Writebacks



- Checkpointing overhead dominated by data writebacks
- Delayed Writeback optimization
  - Processors synchronize and resume execution
  - Hardware automatically writes back dirty lines in background
  - Checkpoint only completed when all delayed data written back
  - Still need to record inter-thread dependences on delayed data

Delayed Writeback Pros/Cons

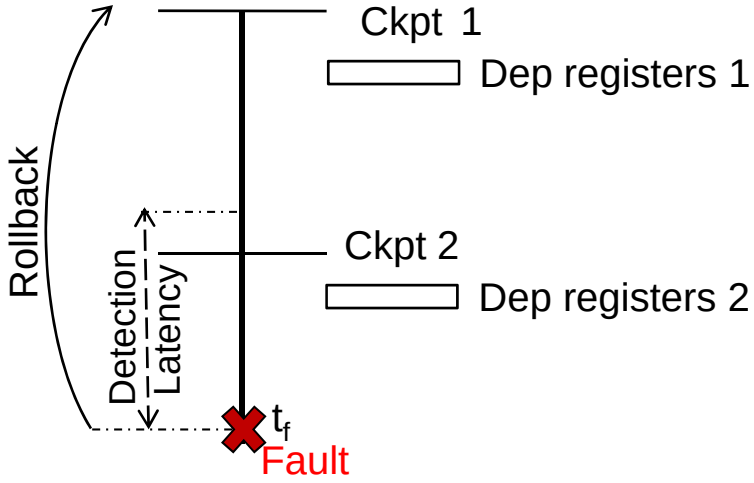
+ Significant reduction in checkpoint overhead

- Additional support:
  - Each processor has two sets of Dep. registers
  - Each cache line has a delayed bit

- Increased vulnerability
  - A rollback event forces both intervals to roll back

Optimization2 : Multiple Checkpoints

- Problem: Fault detection is not instantaneous
  - Checkpoint is safe only after max fault-detection latency (L)



- Solution: Keep multiple checkpoints
  - On fault, roll back interacting processors to safe checkpoints

- No Domino Effect

Multiple Checkpoints: Pros/Cons

+ Realistic system: supports non-instantaneous fault detection

- Additional support:
  Each checkpoint has *Dep* registers
  Dep registers can be recycled only after fault detection latency

- Need to track communication across checkpoints

- Combination with Delayed Writebacks: one more *Dep* register set

Optimization3 : Hiding Chkpt behind Global Barrier

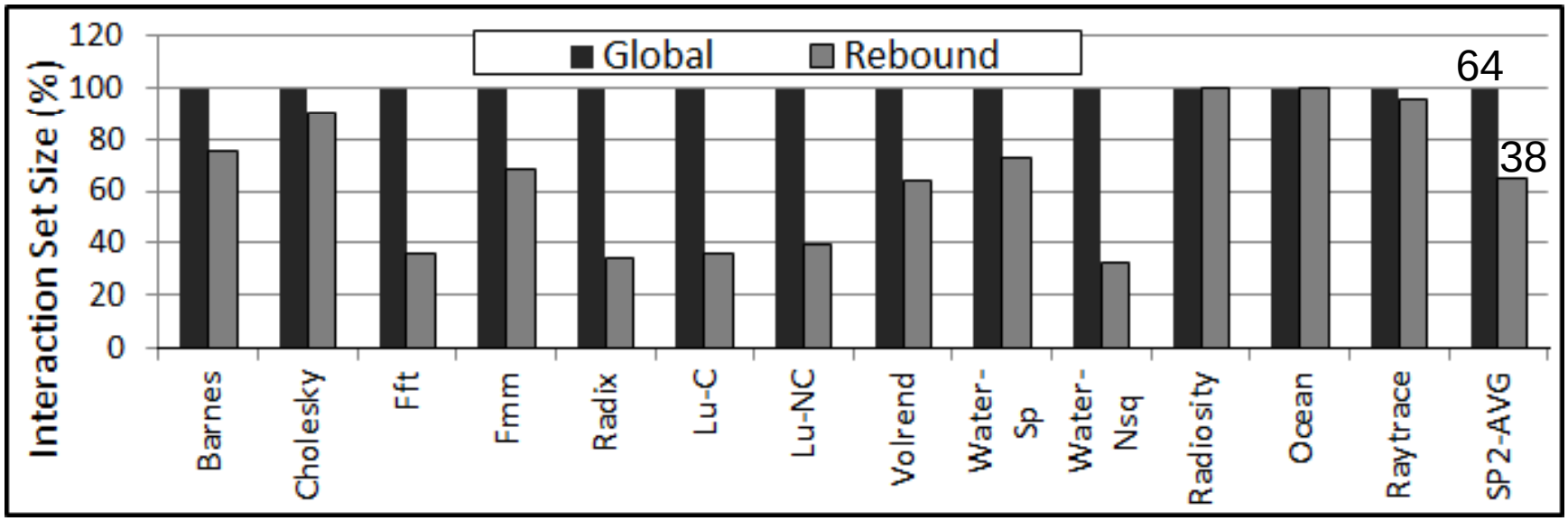
- Global barriers require that all processors communicate
  - Leads to global checkpoints

- Optimization:
  - Proactively trigger a global checkpoint at a global barrier
  - Hide checkpoint overhead behind barrier imbalance spins

Evaluation Setup

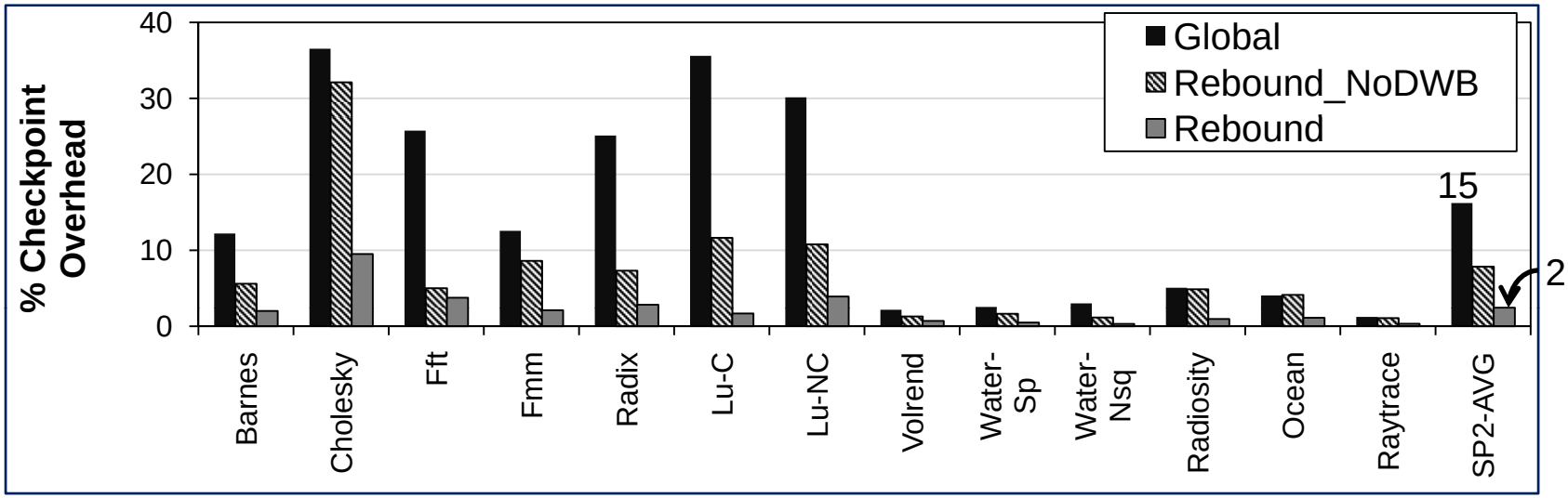
- Analysis tool using Pin + SESC cycle-acc. simulator + DRAMsim
- Applications: SPLASH-2 , some PARSEC, Apache
- Simulated CMP architecture with up to 64 threads
- Checkpoint interval : 5 – 8 ms
- Modeled several environments:
  - Global: baseline global checkpointing
  - Rebound: Local checkpointing scheme with delayed writeback.
  - Rebound_NoDWB: Rebound without the delayed writebacks.

Avg. Interaction Set: Set of Producer Processors



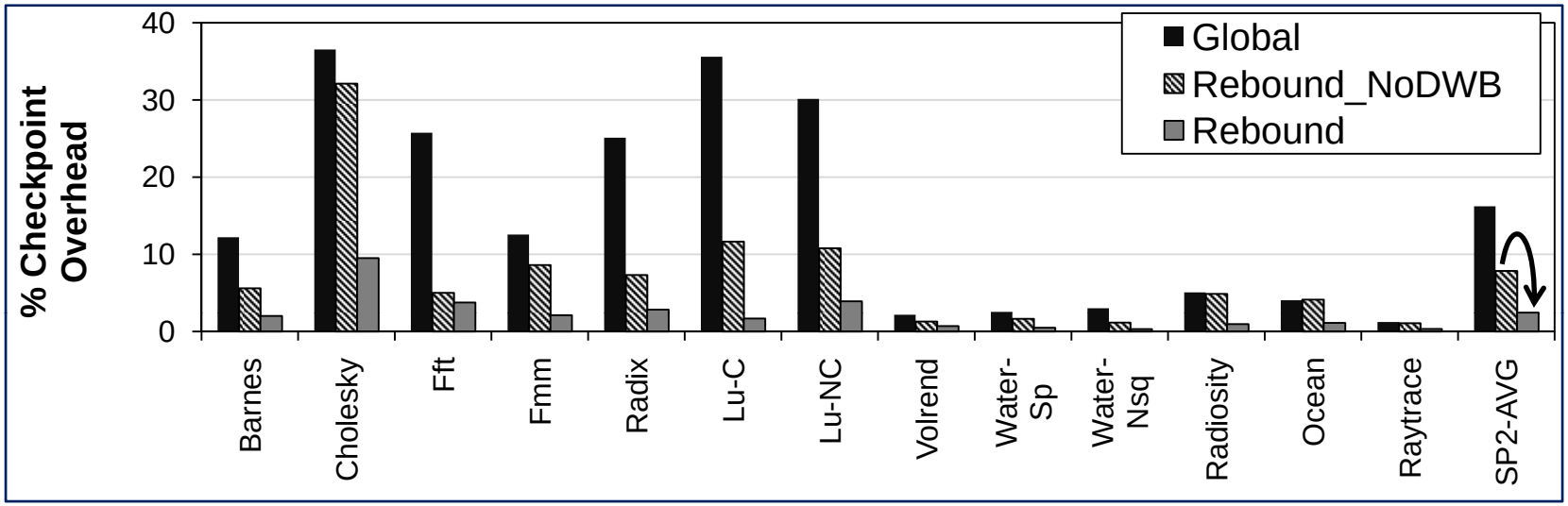
- Most apps: interaction set is a small set
  - Justifies coordinated local checkpointing
  - Averages brought up by global barriers

Checkpoint Execution Overhead



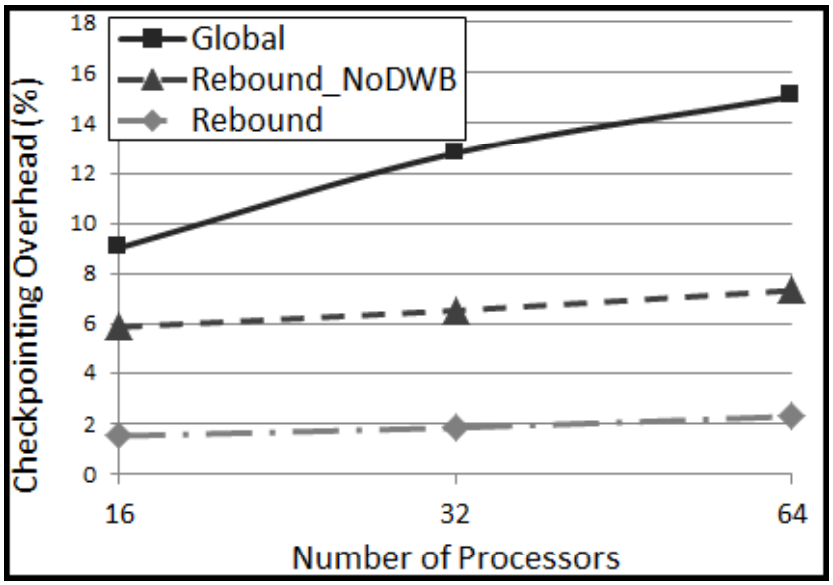
- Rebound's avg checkpoint execution overhead is 2%
  - Compared to 15% for Global

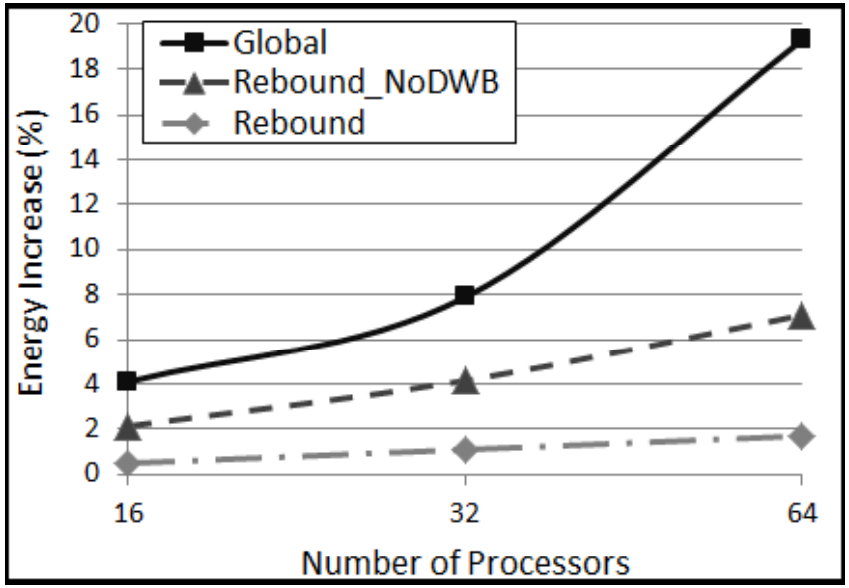
Checkpoint Execution Overhead



- Rebound's avg checkpoint execution overhead is 2%
  - Compared to 15% for Global
- Delayed Writebacks complement local checkpointing

Rebound Scalability





Constant problem size

- Rebound is scalable in checkpoint overhead
- Delayed Writebacks help scalability

Also in the Paper

- Delayed write backs also useful in Global
- Barrier optimization is effective but not universally applicable
- Power increase due to hardware additions < 2%
- Rebound leads to only 4% increase in coherence traffic

Conclusions

Rebound: First HW-based scheme for scalable, coordinated local checkpointing in coherent shared-memory

- Leverages directory protocol
- Boosts checkpointing efficiency:
  - Delayed write-backs
  - Multiple checkpoints
  - Barrier optimization
- Avg. execution overhead for 64 procs: 2%

- Future work:
  - Apply Rebound to non-hardware coherent machines
  - Scalability to hierarchical directories

Rebound: Scalable Checkpointing for Coherent Shared Memory

Rishi Agarwal, Pranav Garg, and Josep Torrellas

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