1. Motivation

- Architectures that continuously execute Atomic Blocks or Chunks (e.g., TCC, BulkSC [Ceze ’07])
- Chunk: a group of dynamically contiguous instructions executed atomically
- Providing performance and programmability advantages [Hammond ’04][Ahn ’09]
- Chunk commit is an important operation: making the state of a chunk visible atomically
- Lazy detection of conflicts provides higher concurrency in codes with more conflicts
- In a lazy directory-based cache coherent system, parallelizing the commit is challenging
- Requires the consistent conflict resolution decision over all the distributed directory modules
- Therefore, the current schemes use sequential commit operations
- In addition, the current lazy conflict resolution is suboptimal
- No squash is required if the conflict is only Write-After-Write (WAW)

2. Inefficiencies of Current Approaches

- Execution: Reads and writes bring lines into cache
  - No written line is made visible to other processors
- Commit: Grouping: setting the relative order of any two conflicting chunks
  - Propagation: making the stores in a chunk visible to the rest of the system

3. BulkCommit = IntelliSquash + IntelliCommit

- IntelliSquash: no squash on WAW-only conflicts
- Challenge: the speculative data produced by a chunk cannot be lost when the chunk is ready to commit
- Solution: L1 cache as the “store buffer” for a chunk
  - Similar to the store buffer in the conventional system
- On receiving invalidation, the speculative dirty words of a line is preserved
- Extra hardware cost:
  - Absent bit: it is set when
    - The line is not presented
    - The line contains some speculative words
- Per-word dirty bit (not shown)

4. Experimental Evaluation

- Cycle accurate NOC simulation with processor and cache model
  - Number of cores: 16 and 64
  - 11 SPLASH-2 and 7 PARSEC applications
  - One or two outstanding chunks
  - Implemented most distributed commit protocols:
    - Scalable TCC (ST)
    - Scalable Bulk (SB)
    - BulkCommit without IntelliSquash (BC-SQ)
    - BulkCommit (BC)

5. Conclusion

- Proposed BulkCommit: commit protocol with parallel grouping and squash-free WAW-only conflict resolution
- Key properties:
  - Serializing WAW between chunks without squashing
  - Exploiting the similarity of the chunk commit and the individual store
  - Parallel grouping
  - Using preemption mechanisms to order two conflicting chunks consistently
- We hope that BulkCommit achieves the optimal design point