ScalableBulk: Scalable Cache Coherence for Atomic Blocks in a Lazy Environment

Xuehai Qian, Wonsun Ahn, Josep Torrellas
University of Illinois

http://iacoma.cs.uiuc.edu/
Motivation

• Architectures that continuously execute Atomic Blocks or Chunks (e.g., TCC, BulkSC)
  • Chunk: group of dynamically contiguous instructions executed atomically
  • Provide performance and programmability advantages [Hammond 04], [Ahn 10]
  • An important operation is commit: makes the state of chunk visible atomically

• Designs with lazy detection of chunk conflicts
  • Commit involves updating cache states and checking for conflicts
  • In lazy directory-based cache coherent systems, commit is very challenging
    • Requires updating the states of the distributed caches in a way that appears that chunks execute in a total order
    • In large systems, it results in an execution bottleneck
Commit in Directory-Based Machine

Conventional Cache Coherence

Chunk-based Cache Coherence

Xuehai Qian

Scalable Bulk Protocol
Recent Commit Protocols

• BulkSC [Ceze 07]:
  • Centralized commit arbiter

• Scalable TCC [Chafi 07]:
  • First distributed scheme
  • Enforce total order by grabbing a commit token
    • Serialization and global communication

• SEQ Protocol [Pugsley 08]:
  • Extends Scalable TCC by eliminating global communication
  • Still requires serialization of commits that use the same directory module
SEQ Protocol

P=Processor    D=Directory module

Commits of chunk 0 and chunk 1 are serialized.

Commits that use the same directory module are serialized even when they touch non-overlapped lines.
Outline

• Motivation
• ScalableBulk
• Evaluation
Goals for Scalable Commit

- No centralized structure
- Committing processor communicates only with the relevant directory modules
- Allow concurrent commits of chunks that use the same directory module, as long as the accessed addresses do not overlap
Contribution: ScalableBulk

- **ScalableBulk**: protocol for bottleneck-free commit of chunks in directory-based system

- **Key properties**:
  - Enable multiple concurrent chunk commits that use the same directory module
  - Made possible by integrating signatures into directory design
  - Better tolerance to commits that use many directories
  - Eliminates all centralized structures and global communications
  - Committing processor only communicates with the relevant directories (the homes of the addresses accessed by the chunk)
  - More scalable than previous schemes
  - Results: practically eliminates all commit stall overhead for 64 processors
ScalableBulk Protocol Primitives

- Allowing multiple non-overlapping commits to use the same directory module
- Grouping directory modules
- Initiating the commit optimistically

Many-Core Architecture Considered
Primitive 1: Allowing Concurrent Non-overlapping Commits

- Read and write footprint of a chunk is summarized in read and write signatures using Bloom filters
- On commit: signatures are sent to the relevant directory modules
- Only the addresses in the signature are locked in the directory during the commit
- Other chunks can commit using the same directory if their signatures do not conflict with the committing signatures
Primitive 1: Allowing Concurrent Non-overlapping Commits

- Enables more concurrent commits
Primitive 2:  
Grouping Directory Modules

- On a chunk commit: the relevant directory modules
- Coordinate their transitions by exchanging messages
- Form a Directory Group
- Identify a leader module that sends messages to the caches and the committing processor on behalf of the group

Grouping Protocol:
- Complete distributed operation
- Few messages are required
- Leader: lowest-numbered directory module in the group
Primitive 2: Grouping Directory Modules

P=Processor    D=Directory module

Leader

Group is formed

Sig{R,W} Sig{R,W} Sig{R,W}
D0 grab D1 grab D2 grab

Commit finished

Chunk 0

Xuehai Qian

Scalable Bulk Protocol
Distributed Conflict Detection

P=Processor    D=Directory module

Chunk 0    Chunk 1

P0    P1

D0    D1    D2

Sig{R,W}    Sig{R,W}

grab    grab

D0    D1    D2

Conflict is detected in D1

g Failure    commit failure

D0    D1    D2

Xuehai Qian

Scalable Bulk Protocol
Primitive 2: Grouping Directory Modules

- Deadlock is avoided by following a fixed directory-module traversal order
- Multiple groups can commit concurrently
Concurrent Commit

P = Processor    D = Directory module

No conflict is detected in D1.
Two groups commit concurrently.

Chunk 0
Chunk 1

D2
D1
D0
P0
P1

P=Processor    D=Directory module

Sig{R,W}
grab

Xuehai Qian
Scalable Bulk Protocol
Primitive 3: Optimistic Commit Initiation

- Idea:
  - Committing processor (CP) assumes its commit transaction will succeed
  - CP consumes incoming messages before receiving OK to commit
- Advantages: it enables more overlapping of commits
- Details in the paper
Summary: Scalable Commit

- Commit has no centralization point
- Committing processor communicates only with relevant directory modules (no message broadcasting)
- Multiple committing chunks can use the same directory modules, if the addresses that they access do not overlap
- Similar to how conventional protocols support concurrent writes
- Optimistic Commit Initiation removes operations from critical path of the commit

Commit is truly scalable
Outline

• Motivation
• ScalableBulk
• Evaluation
Evaluation

- Cycle-accurate execution-driven simulator based on SESC and Pin
- Number of cores: 32 and 64
- 11 SPLASH-2 and 7 PARSEC applications
- Implemented all existing protocols:
  - ScalableBulk
  - Scalable TCC
  - SEQ
  - BulkSC
ScalableBulk practically eliminates all commit stall time

- Other existing protocols suffer commit stall (see paper)
Directories Used Per Commit

- Chunk commits use about 6 directories on average.
- ScalableBulk is able to overlap the commits that use the same directories if the signatures do not conflict.
Network Message Characterization

- ScalableBulk sends fewer messages than other distributed protocols
Also in the paper

- Mechanism to handle fairness and avoid starvation
- Many details on the implementation of the ScalableBulk protocol
- Detailed results characterizing various aspects of the protocol
Conclusion

• Proposed *ScalableBulk*: protocol for bottleneck-free commit of chunks in lazy directory-based system

• Key properties:
  • Enables multiple concurrent chunk commits that use the same directory module
    • Thanks to use of signatures
  • Eliminates all centralized structures and global communications

• Results: practically eliminates all commit stall overhead for 64 processors
• Effectively enables a large-scale chunk-based machine
ScalableBulk: Scalable Cache Coherence for Atomic Blocks in a Lazy Environment

Xuehai Qian, Wonsun Ahn, Josep Torrellas
University of Illinois

http://iacoma.cs.uiuc.edu/