Supporting Software Transactional Memory in Distributed Systems based on the Data-Flow Model

Bo Zhang and Binoy Ravindran

Real-Time Systems Laboratory
Department of Electrical and Computer Engineering
Virginia Tech
Blacksburg, VA, USA

Sep 17, 2010
Distributed STM based on the data-flow model

- Distributed STM is a promising alternative to lock-based distributed concurrency control
- Data-flow distributed STM model [HS07]
  - Nodes communicate by message passing links
  - Transactional contention on an object is now distributed
  - Object requests can be ordered using a distributed queue
  - Distributed cache-coherence using distributed queuing
The data-flow model: two elements

- **A** Txn requesting object o
- **B** Txn holding object o

```
CC.locate(o)  
CC.move(o)  
```

```
req(o)  
not found  
```

**Network**

```
TM Proxy  
Local Cache  
TM Proxy  
Local Cache  
```

```
CR(A,B)  
```

- **Distributed cache-coherence protocol (CC)**
  - Ballistic [HS07]: based on simple distributed queuing protocols
- **Conflict resolution (CR) strategy**
  - Relay [ZR09]: reduces the worst-case # of aborts
  - Contention management strategy: migrated from multiprocessor TMs
  - Can we do anything more?
Current work summary

- Cache-coherence protocol design
  - Competitiveness of distributed queue-based cache coherence? [BA, PODC ‘10]
    - Distributed queuing cache-coherence (DQC) protocols
    - Distributed priority queuing cache-coherence (DPQC) protocols
    - DPQC protocols performs an order of N better than DQC protocols for N concurrent write txns

- Conflict resolution strategy
  - Distributed dependence-aware (DDA) model: relax the restriction of aborting at least one transaction when a conflict occurs by adopting different conflict resolution strategies based on transaction types. [BA, PODC ‘10]
    - Allows maximum concurrency for read-only and write-only txns: those txns never abort
    - Use contention management policy to treat non-write-only update txns
Future work

- Distributed STM scheduling
  - A more flexible strategy in resolving conflicts: not only decides which transaction to proceed upon conflicts, but also schedules a transaction: when a new transaction enters the system, when an aborted transaction restarts.
  - Taking into account the locations of transactions in scheduling transactions.
- Supporting nested transactions
  - Correctness criterion has to be relaxed to some extent
  - Lock-free conflict resolution strategy
  - Cache-coherence protocols design
- Supporting linked data structures
  - Moving objects are parts of larger linked data structures.
  - Trade-off: keeping multiple copies and significant bookkeeping vs efficient garbage-collection
  - Different strategies based on workloads
Future work

- Supporting nested transactions
  - Correctness criterion has to be relaxed to some extent
  - Lock-free conflict resolution strategy
  - Cache-coherence protocols design

Example [ALS06]: the execution $A_1 I_1 B C A_2$ is permitted by open nesting. However, txn A reads an inconsistent value of b. In this case, A does not execute atomically.
Future work (Cont’d)

- Supporting linked data structures
  - Moving objects are parts of larger linked data structures.
  - Trade-off: keeping multiple copies and significant bookkeeping vs efficient garbage-collection
  - Different strategies based on workloads