Lecture 18: Transactional Memories II

Papers:
- LogTM: Log-Based Transactional Memory, HPCA'06, Wisconsin
- LogTM-SE: Decoupling Hardware Transactional Memory from Caches, HPCA'07, Wisconsin

LogTM

- Underlying conventional directory-based protocol

- Eager Versioning: writes happen "in place", old values are stored away in virtual memory (hence, can handle large transactions), old values will be reinstated in case of an abort (expensive abort), not much to be done on a commit (fast commit)

- Eager Conflict Detection: coherence permissions are acquired immediately, a conflict is signaled if the latest value is obtained from a transaction that has not yet committed the write

Handling Reads

- P2 detects that someone is trying to read a value in its uncommitted write set; P2 not ready to make A available (has other work that it wants to finish "atomically"); sends a "conflict" message to requestor
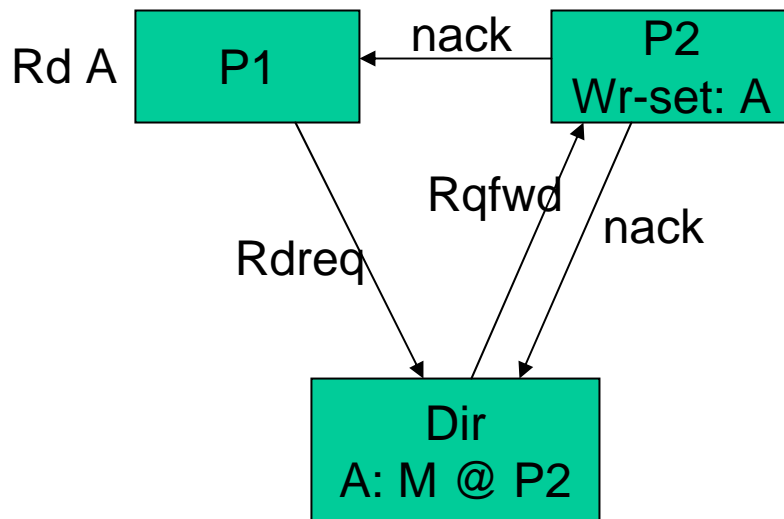
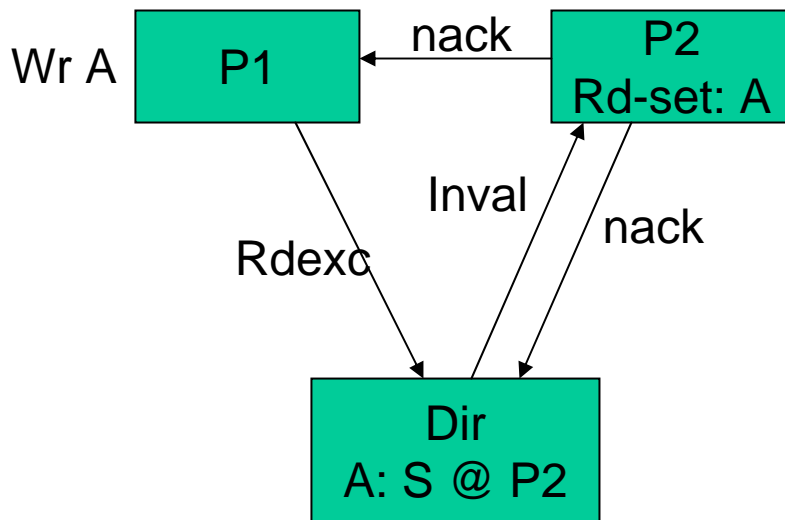Rd A | P1 ←nack P2 Wr-set: A

Rdreq   Rqfwd   nack

Dir
A: M @ P2

Dealing with the conflict:
- Both can abort – livelock!
- P1 can abort… it is possible that P2 also tries to read an uncommitted write of P1 and also aborts – livelock!
- P2 can abort… as above, can lead to livelock
- P1 waits for a while and re-tries; if T2 has finished, A is available; can cause deadlock as P2 may also wait for P1; eventually a software contention manager kicks in

Handling Writes

- The same rules apply, whether A is in P2's read or write set

Dealing with the conflict:
- Both can abort – livelock!
- P1 can abort… it is possible that P2 also tries to read an uncommitted write of P1 and also aborts – livelock!
- P2 can abort… as above, can lead to livelock
- P1 waits for a while and re-tries; if T2 has finished, A is available; can cause deadlock as P2 may also wait for P1; eventually a software contention manager kicks in

Block Replacement

- If a block in a transaction's rd/wr-set is evicted, the data is written back to memory if necessary, but the directory continues to maintain a "sticky" pointer to that node (subsequent requests have to confirm that the transaction has committed before proceeding)

- The sticky pointers are lazily removed over time (commits continue to be fast)

Versioning

- Every write first requires a read and a write to log the old value – the log is maintained in virtual memory and will likely be found in cache

- Aborts are uncommon – typically only when the contention manager kicks in on a potential deadlock; the logs are walked through in reverse order

- If a block is already marked as being logged (wr-set), the next write by that transaction can avoid the re-log

- Log writes can be placed in a write buffer to reduce contention for L1 cache ports

Results

SPLASH2 Benchmark Results
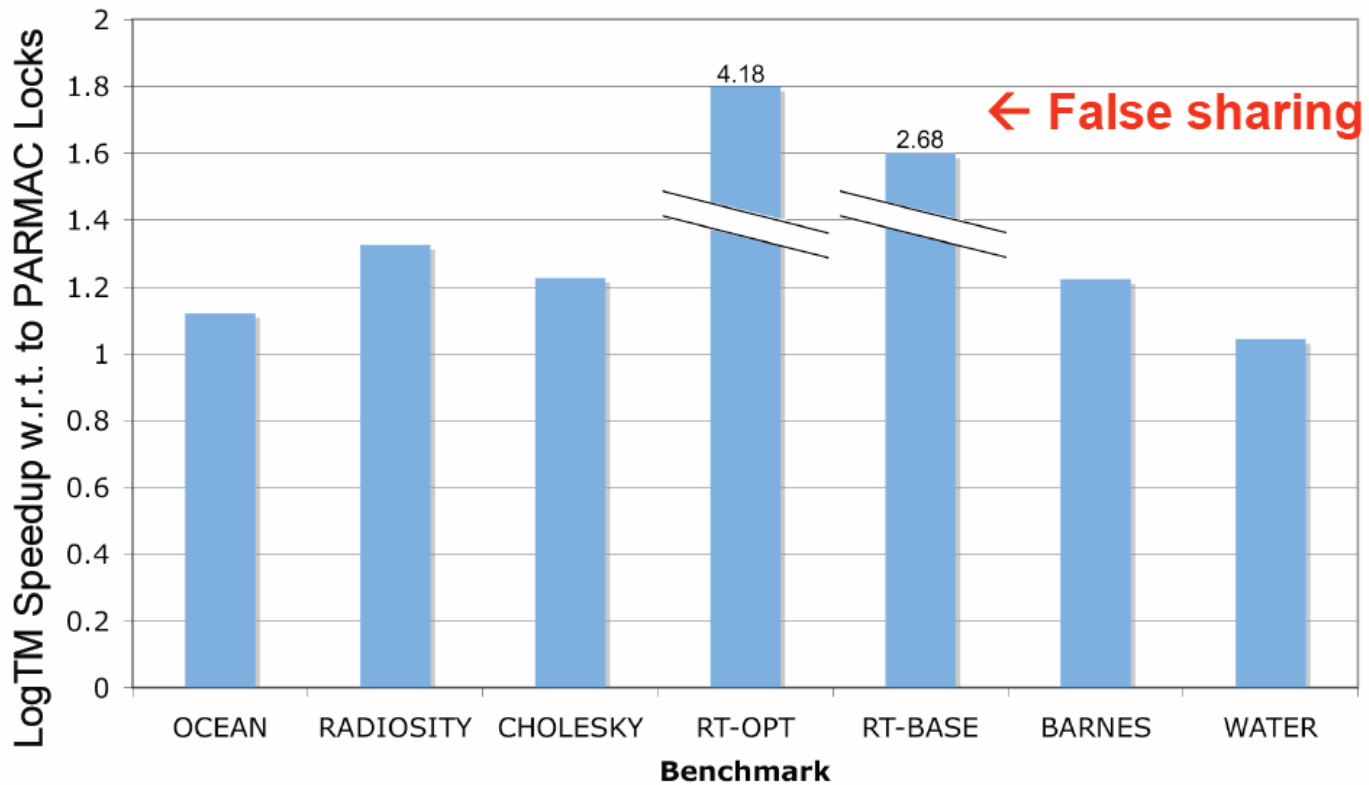
Results

Few stalls and aborts

Small write log buffer will avoid L1 port contention

Writes are preceded by reads, so a log imposes low overhead

Table 5: Selected transactional data

| Benchmark	# Trans.	% Stalls	% Aborts	Stores/Trans.	% Read-Modify-Writes
Barnes	3,067	4.89	15.3	5.50	27.9
Cholesky	22,309	4.54	2.07	1.68	82.3
Ocean	6,693	0.30	0.52	0.112	100
Radiosity	279,750	3.96	1.03	1.64	82.7
Raytrace-Base	48,285	24.7	1.24	1.96	99.9
Raytrace-Opt	47,884	2.04	0.41	1.97	99.9
Water	35,398	0.00	0.11	1.98	99.6

8
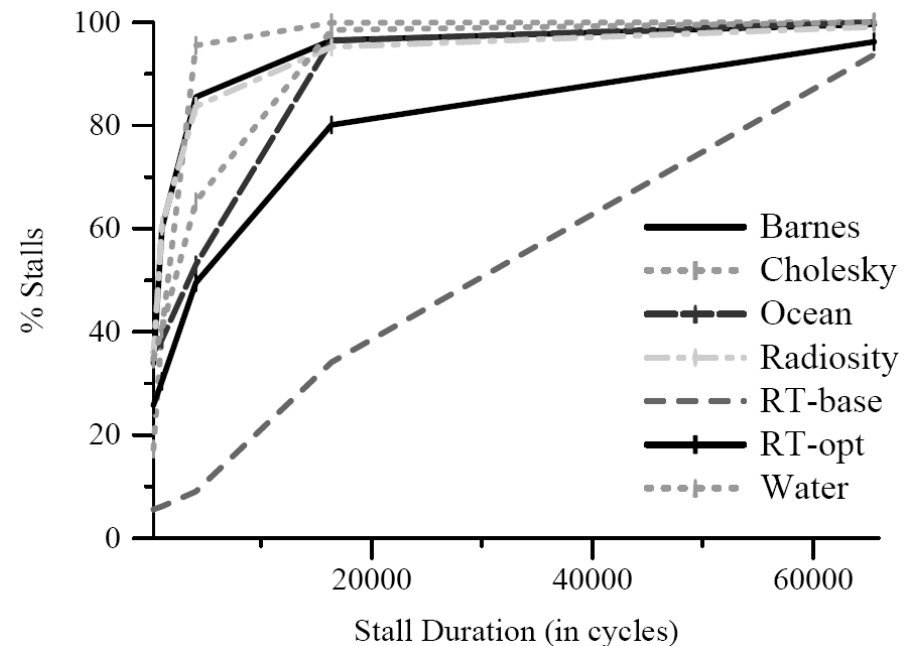
LogTM-SE

- Motivation: decouple conflict detection from cache implementation: maintain a signature to capture read and write sets
  - eliminates R and W bits for every cache block
  - eliminates the need to flash-clear above bits on commit
  - easy to save transaction state (simplifies nesting, context switching, etc.)

- Conflict detection can now yield false positives: signature indicates there is a conflict, but in reality, none exists
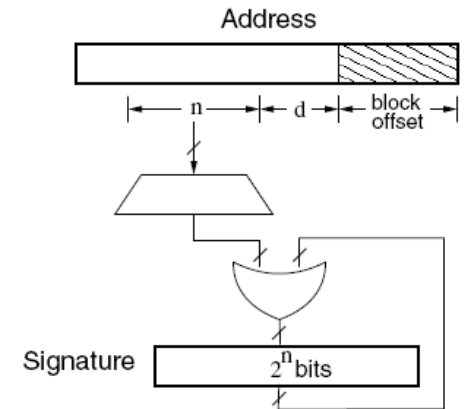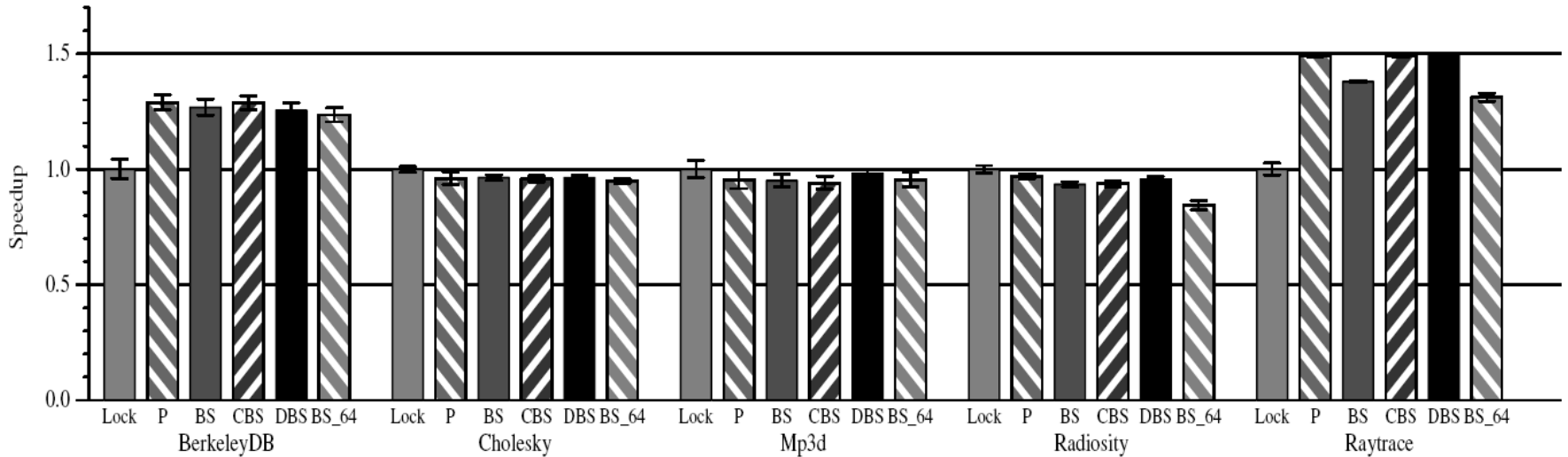
Signatures
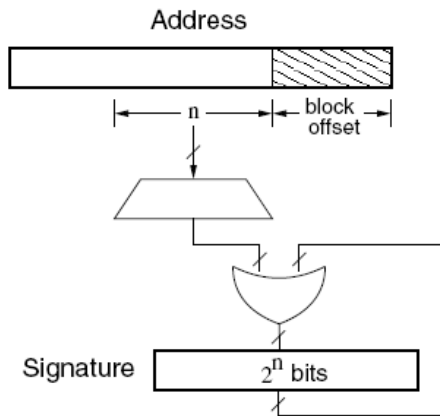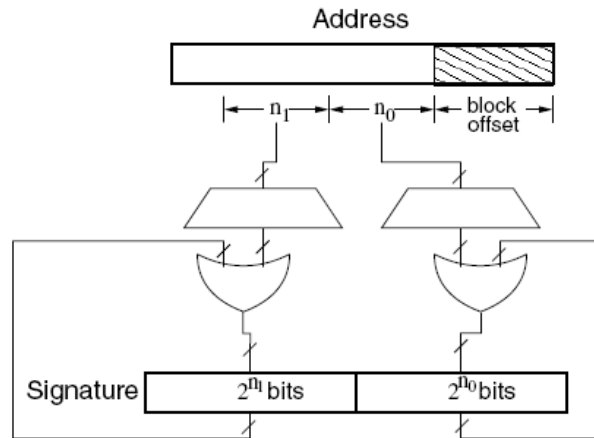
(a) bit−select (BS) (b) double−bit−select (DBS) (c) coarse−bit−select (CBS)

Signature Behavior

P   : perfect (rd and write sets)
BS   : bit select (2048 bit signature)
CBS   : coarse bit select
DBS   : double bit select
BS_64 : bit select (64 bit signature)

Title

- Bullet