On Closed Nesting and Checkpointing in Fault-Tolerant Distributed Transactional Memory

Aditya Dhoke, Binoy Ravindran, Bo Zhang (speaker Roberto Palmieri)
ECE Dept. Virginia Tech
Software Transactional Memory

- Centralized STM
  - Simple programming model
  - Easy debugging --- deadlocks, livelocks, lock convoying, priority inversion
- Distributed STM (DTM)
  - Implementation distributed locking
  - Support for synchronization of distributed application
Nesting Models

- Flat Nesting
  - Abort causes entire transaction to restart

- Closed Nesting
  - Enclosed inside parent transaction
  - Commit of inner transaction local
  - Abort independently of parent
  - Partial rollback mechanism
Checkpointing

- Checkpointing
  - Generalization of closed nesting
  - Checkpoint saves execution state and transactional metadata
  - Rollback to previous checkpoint for abort
  - Partial rollback mechanism
Motivation with an example

- Matrix: $m_1$, $m_2$, $m_3$
- Result = $m_1 + m_2 + m_3$
- Conflict on accessing $m_3$

- Flat Nesting will restart from $T_{\text{flat}}$
- Closed Nesting will restart $T_{\text{closed}}$

```
T_{\text{flat}}:
    m1 = getRemote(m1_Obj);
    m2 = getRemote(m2_Obj);
    m3 = getRemote(m3_Obj);
    intm = add(m1,m2);
    result = add(intm,m3);
    if commit()
        return result;
    retry $T_{\text{flat}}$
```

```
T_{\text{closed}}:
    result = add(intm,m3);
    if commit()
        return result;
    retry $T_{\text{flat}}$
```
Partial rollback benefits

- Do not repeat operation on $m_1$ and $m_2$
- Saved computation cost and remote calls
- Reduced abort rate as well, thus reduced transaction execution time
- Suited for replicated systems, where operations are costly
Questions?

- What application/workload will benefit from partial abort, as compared to flat nesting?
- What is the potential performance improvement or degradation of partial abort?
- Which parameters of a transaction will affect the partial abort performance?

IN THE CONTEXT OF DTM
Quorum-based Replication (QR-DTM)

- **Network layer:**
  - Nodes form logical ternary tree
  - Read and write quorum created from tree
  - Read and write quorum always intersect
  - Reduces the number of nodes to contact

- **DTM protocol (Full replication):**
  - Read quorum services read and write requests
  - Selecting the highest version from read quorum gives the latest copy
  - Write quorum services commit requests
Network organization

On Closed Nesting and Checkpointing in Fault-Tolerant Distributed Transactional Memory
27th IEEE International Parallel and Distributed Processing Symposium, IPDPS 2013

VirginiaTech
Invent the Future
QR-DTM: Early Validation

• For every read request:
  • Validate previously read objects (piggyback read-set on read request)
  • On success, proceed to service the request
  • On failure, send abort message to the transaction
Example /1

- Read quorum R1 and Write quorum W2 intersect at n2
- T1 reads o1, o2 & o3
Example /2

- T2 commits changes to o2
- T1 requests o4 and validates all the read-set
- n1 validates T1's read-set, finds o2 is old and sends abort
QR-CN: Closed Nested Operations

- Read operation (from remote node)
  - Validate current read-set
  - If fail
    - Find the transaction with object invalidated (that needs to be aborted)
    - Send an abort message with the object not valid
  - If success
    - Responds with latest copy of requested object
QR-CN : Closed Nested Operations

• Read operation (from local node)
  • If abort response
    • Abort child or parent
  • If success
    • Select the last version from all the versions sent by remote nodes
    • Return the object

• Same behavior for write operation
QR-CN: Closed Nested Operations

- Nested transaction:
  - Commit
    - Merge read and write set with the parent
    - Read quorum validation ensures that data-set is valid at commit time

- Parent transaction:
  - Commit using write quorum
QR-CHK: Checkpointing Operations

- Transaction creates checkpoints locally
- Conflict during execution phase, can restart from appropriate checkpoint
- Checkpoint saves
  - Program counter
  - Read/write set at that point
  - Records the checkpoint ID
QR-CHK: Checkpointing Operations

- Remote nodes record the checkpoint ID for each object

- Remote node
  - For each read request, remote node validates the data-set
  - On failure
    - Finds the least checkpoint ID among the conflicting objects that has all its objects valid
QR-CHK: Checkpointing Operations

• Local Node:
  • On success, returns latest copy of requested object
  • On failure, retrieves the checkpoint ID, restores to it and resumes execution

• Checkpoints are ordered following the objects access pattern. In case of multiple conflicts, the checkpoint with minimum id is restore.
Implementation /1

- QR-CN
  - Java Exceptions for aborting transaction
  - Transaction throws exception with ID of aborted transaction
  - Transaction catching compares the ID
  - If the matching fails, throws another exception caught by parent
Implementation /2

- QR-CHK
  - Java Continuations for creating checkpoints and rollback
  - Continuations save the execution state in Java object
  - Rollback retrieves this object and resumes execution

Possible bottleneck!
Experimental study

- For QR-CN, each operation is a closed nested transaction
- For QR-CHK, checkpoint created after reading every object
- Benchmarks
  - Bank, Hashmap, RBTree, SkipList, Vacation (STAMP)
- Test-bed: 40 nodes
On Closed Nesting and Checkpointing in Fault-Tolerant Distributed Transactional Memory

27th IEEE International Parallel and Distributed Processing Symposium, IPDPS 2013
On Closed Nesting and Checkpointing in Fault-Tolerant Distributed Transactional Memory
27th IEEE International Parallel and Distributed Processing Symposium, IPDPS 2013
Observations

- % read workload
  - Closed nesting outperforms flat nesting and checkpoints with higher margin for mostly write workload

- Transaction length
  - Increasing transactional calls, the gap between closed nesting and other increases

- Object variation
  - With a large read-set, closed nesting contention increases the gap against flat.
Conclusion

• Closed nesting has performance gain (vs flat-nesting) of 53% across all benchmarks

• 33% reduction in abort rate of closed nesting (vs flat-nesting)

• Checkpointing has 16% of performance degradation with 19% message overhead
...and the winner is...

CLOSED NESTING
Thanks

Questions?

http://www.hyflow.org/