# On Closed Nesting in Distributed Software Transactional Memory

Alex Turcu    Binoy Ravindran    Mohamed Saad

February 26, 2012

# Overview

- Background
  - Nesting in Transactional Memory
  - TFA (Transactional Forwarding Algorithm)
- System model
- Nested Transactional Forwarding Algorithm
- Evaluation
- Conclusion

# Background

# Nesting in Transactional Memory

- Three kinds of nesting: flat, closed, open
  - Flat nesting is the most common in implementations, does not support partial rollback
  - Closed nesting allow aborting sub-transactions without aborting the parent
  - Open nested sub-transactions commit directly to memory, releasing isolation
- Code composability is main reason for nesting. Others:
  - Potentially increased concurrency
  - Conditional synchronization (retry when precondition is met)
  - Fault management (try...orElse)

# Transactional Forwarding Algorithm (1/2)

- TFA is a protocol for distributed STM
  - Based around Transactional Locking II and Lamport clocks
  - (distribution model: nodes communicating through a message passing links)
- Provides a way to establish ”happens before” relationships
  - Each node holds a node-local clock
  - Clock value affixed to all messages
  - Clock incremented on local transactions’ commits
  - When a message from a node with a higher clock is received, local clock is updated

# Transactional Forwarding Algorithm (2/2)

- Each txn stores its starting time
- When a txn communicates with a node with a higher clock:
  - Attempt to update txn's starting time (i.e. *transactional forwarding*)
  - Must validate read-set before forwarding
    - Success → update txn starting time and continue
    - Failure → abort txn
- Redo log approach (buffered writes), deferred lock acq
- Properties:
  - correctness: opacity
  - liveness: strong progressiveness

Overview

Background

System model

Base model

Nesting model

N-TFA

Evaluation

Conclusion

# System model

# Base model

- n nodes $\{N_1, N_2, ...N_n\}$
- Nodes communicate via message passing links
- Objects accessed using transactions $\{O_1, O_2, ...\}$
  - Shared registers, get/set
  - Each object $O_j$ has an ID, $id_j$
  - Each object has an owner, $owner(O_j)$
  - Objects can migrate (i.e. change owners)
- Transactions $\{T_1, T_2, ...\}$
  - Transactions are immobile and execute on a single node from start to finish

## Nesting model

- Sub-txns executed on the same node as parent/root txn
- A txn can have at most one active child (linear nesting)
- Operations in closed nesting:
  - Sub-txn commit = merge read and write-sets into those of parent’s
  - Read = Find location in read and write-sets from crt txn until root; read location from memory if not found
- No changes compared to the flat nesting model:
  - Write = add new value to write-set of current txn
  - Root txn commit = write to shared memory
  - Abort = discard read and write-sets for current txn

# Nested Transactional Forwarding Algorithm

# N-TFA Introduction

- Nested Transactional Forwarding Algorithm: an extension of TFA with support for closed nesting
- Defines two types of commit, inherited directly form the nesting model definition:
  - merge commit model
  - top-level commit model

Overview

Background

System model

N-TFA
Introduction
Transactions
Forwarding
Merge commit
Aborts

Evaluation

Conclusion

# N-TFA Transactions

- Root transaction:
  - Stores node-local clock on start
  - Increments node-local clock on commit
  - Acquires locks on commit
- Sub-transactions:
  - Do not change the shared memory and thus are not *globally important*
    - Do not record their starting time
    - Do not increment node-local clocks on commit
  - Do not acquire any locks

# N-TFA Forwarding

- When a txn communicates with a node having a higher clock value, it gets forwarded (read-set is validated; starting time updated)
- Sub-transactions do not store their starting time
  - Compare remote clock value with root txn’s starting time
  - Update root txn’s starting time
- Must validate read-sets of all transactions using the same starting time
  - Current sub-transaction and all its ancestors

# N-TFA Merge Commit Model

- When sub-transactions commit, they merge read and write-sets into those of parent’s
- Validating read-set at this stage is possible, but not required
- Validating pros:
  - conflicts can be detected earlier
  - may need to retry smaller sections of work
- Validating cons:
  - network access cost
- Choose not to validate (performance always lower with validation enabled)

Overview

Background

System model

N-TFA

Introduction

Transactions

Forwarding

Merge commit

Aborts

Evaluation

Conclusion

# N-TFA Aborts

- N-TFA benefit comes from partial rollback
  - Only applicable for conflicts detected during early validation
  - Abort as many sub-transactions as needed to resolve the conflict
  - In DTM, the invalid object needs to be retrieved again from the network, so transaction that originally opened the object must retry (there is no automatic re-opening)

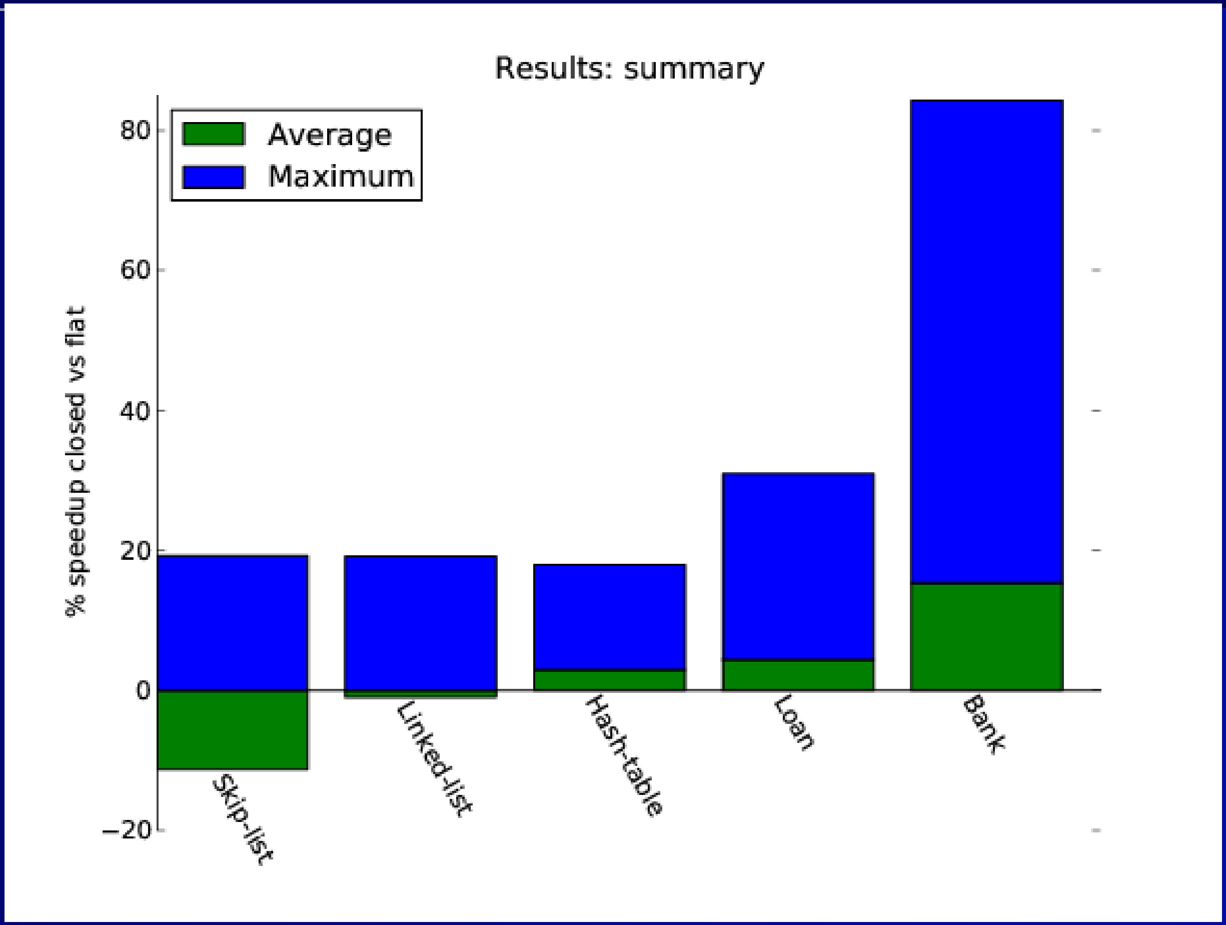
# Evaluation

# Experimental settings

- Implemented in HyFlow, a Java DTM framework[1]
- Benchmarks: two monetary applications (bank and loan) and three micro-benchmarks (linked-list, skip-list, and hash-table).
- Evaluated using up to 48 nodes (AMD Opteron at 1.9GHz) running Ubuntu Server 10.04

[1] available at http://hyflow.org

# Summary



- Avg improvement 2% compared to flat nesting
- Max improvement 84% (max degradation 42%)

# Observations

- Inconsistent/unreliable parameters:
  - Transaction length (in milliseconds)
  - Read-only ratio
- Reliable parameters:
  - N-TFA performs best when transactions consist of around 2-5 sub-transactions

Overview

Background

System model

N-TFA

Evaluation

Settings

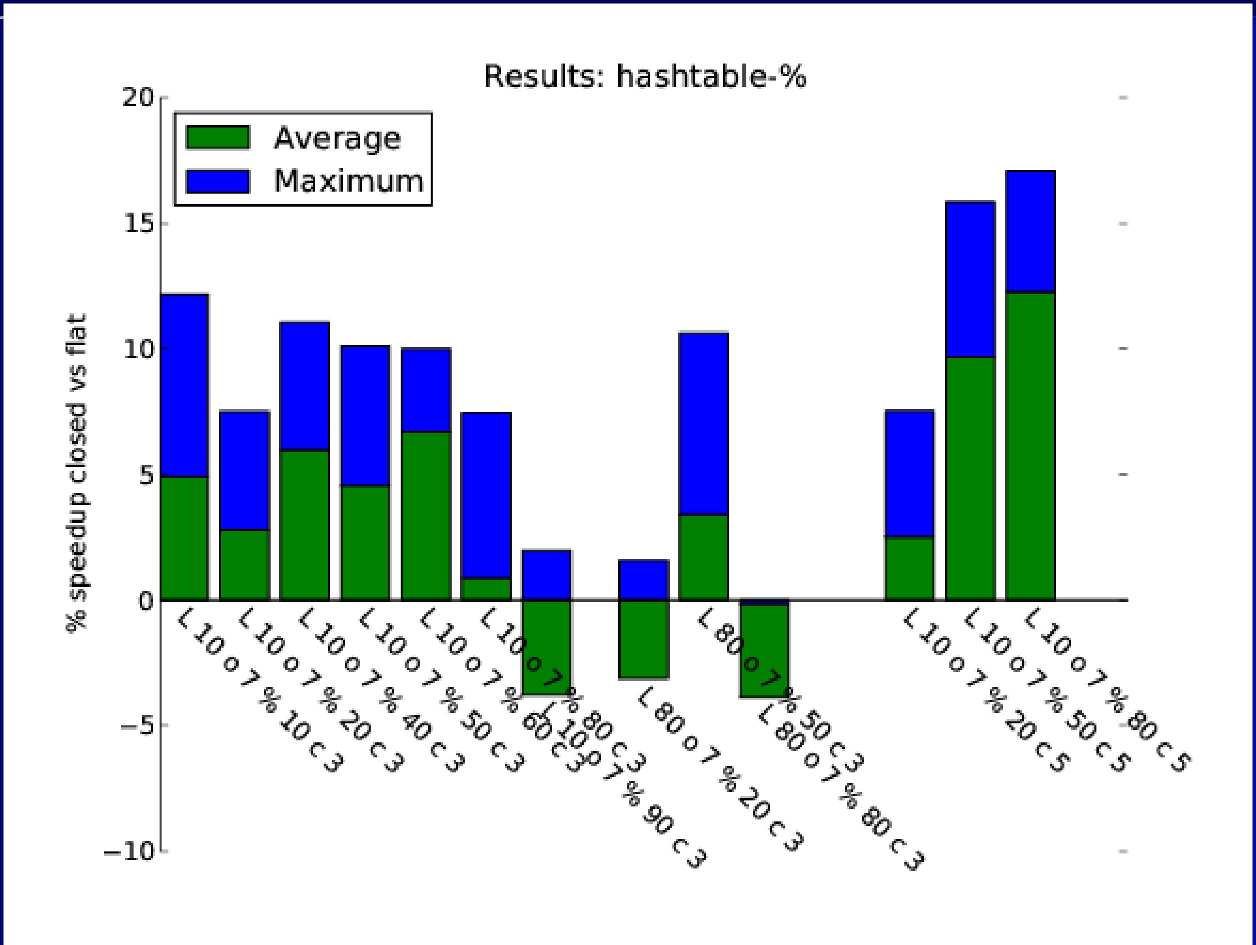
Summary

Observations

Sample

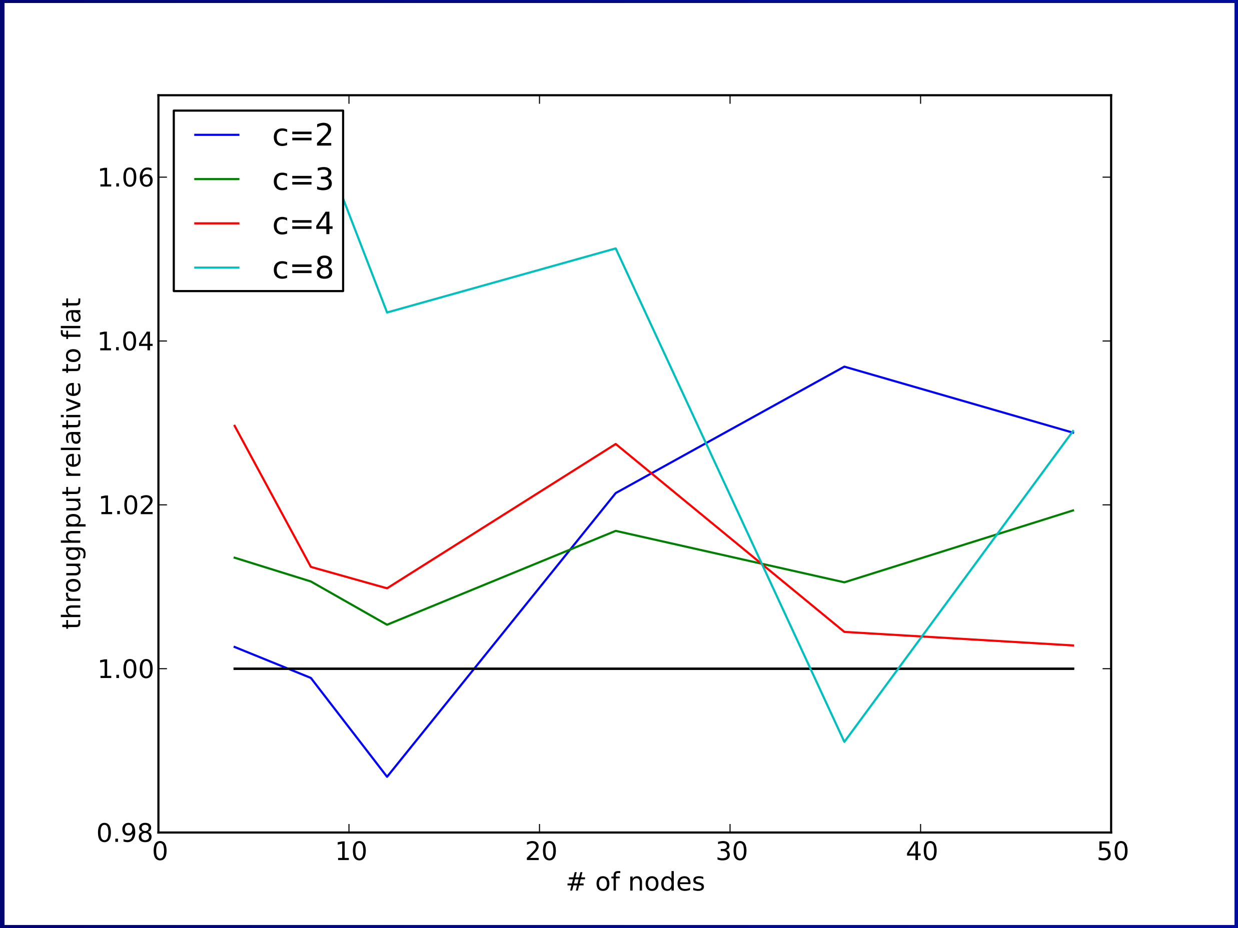
Conclusion

# Sample plots: hash-table



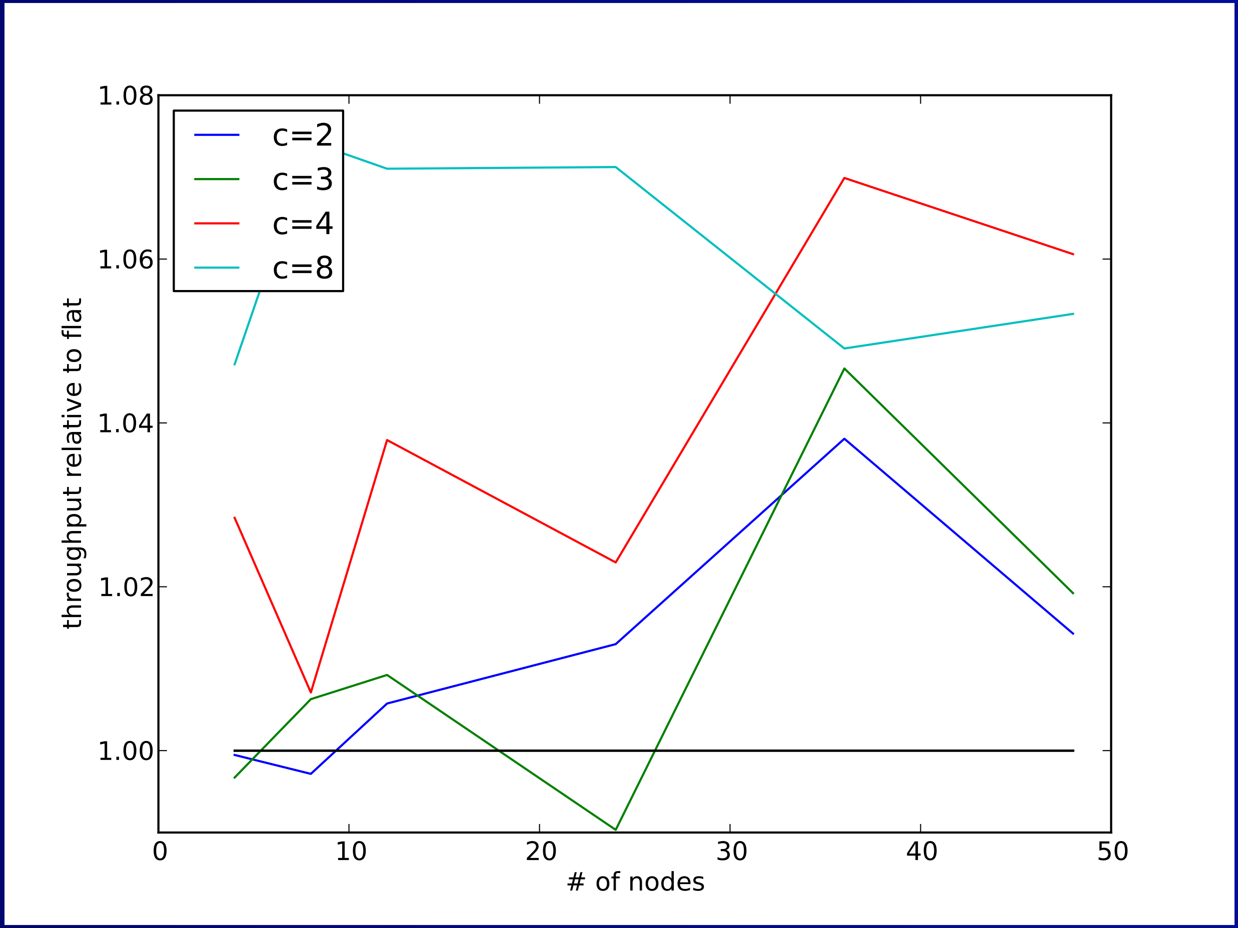
# Sample plots: hash-table

20% reads



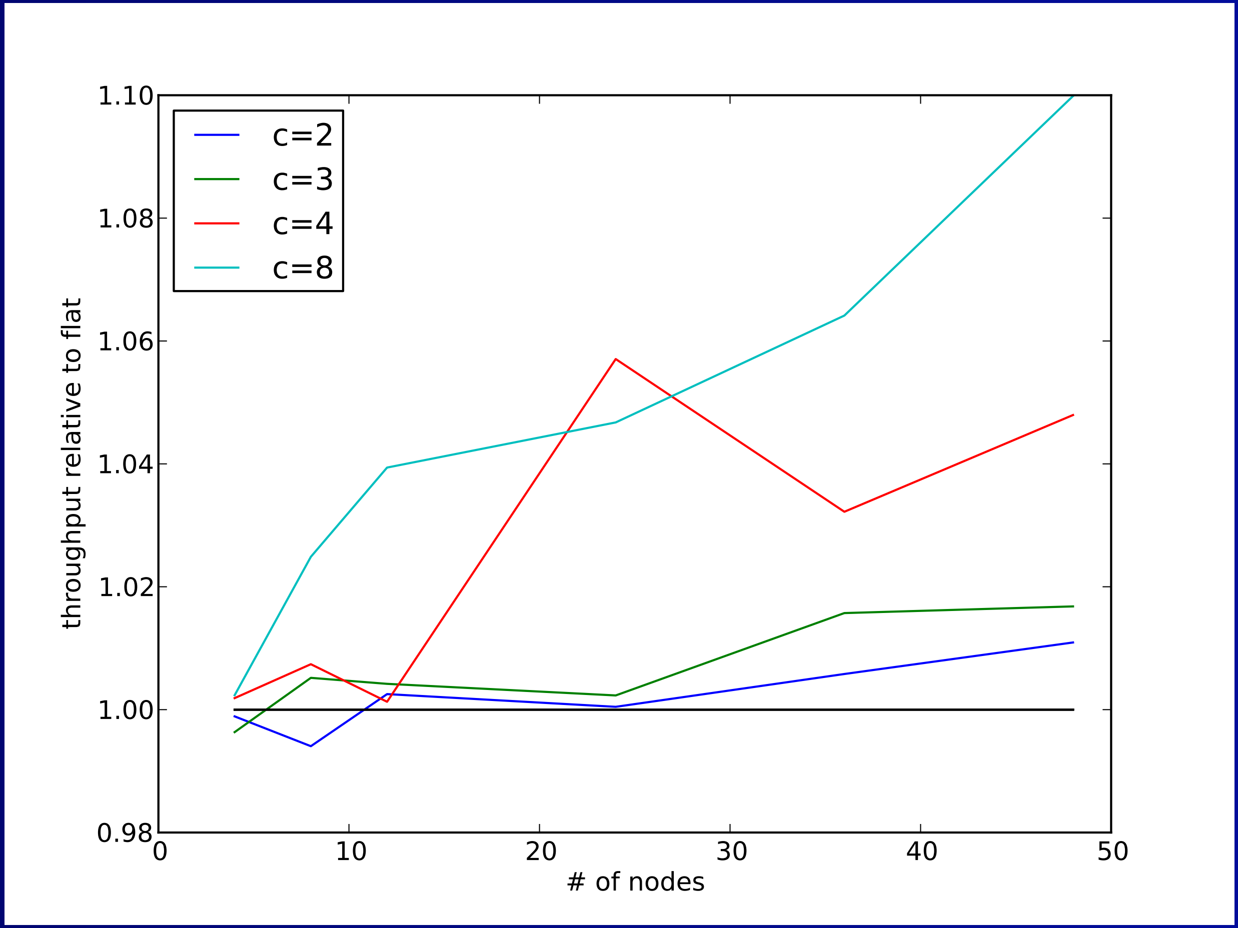
# Sample plots: hash-table

## 50% reads



# Sample plots: hash-table

## 80% reads



# Conclusion

# Conclusion

- N-TFA is a Distributed Transactional Memory protocol with support for partial rollback through closed nesting
- N-TFA benefits when invalid objects are detected in the middle of transaction execution via already existing early validation
- Can not perform extra validations due to network costs
- Transaction length and read-only ratio have a benchmark-dependent influence (can not generalize)
- Maximum benefit around 2-5 sub-transactions

Overview

Background

System model

N-TFA

Evaluation

Conclusion

# Questions

?