Autonomic mechanisms for transactional replication in elastic cloud environments

Paolo Romano

2nd Workshop on Software Services, Timisoara, Romania, 6 June 2011
About me

• Master (2002) and PhD (2007) from Rome University “La Sapienza”
• Researcher & Lecturer at Rome University “La Sapienza” (2007-2008)
• Senior Researcher at Distributed Systems Group, INESC-ID, Lisbon (since 2008)

• Coordinator of the FCT Aristos Project (Jan 2010-Jan 2012)
  – Bilateral Italian-Portuguese project
  – Autonomic Replication of Transactional Memories

• Coordinator of the FP7 Cloud-TM Project (Jun 2010-Jun 2012)
  – 4 international partners from industry and academy
  – Self-tuning, Distributed Transactional Memory platform for the Cloud

• Coordinator of the Cost Action Euro-TM (fall 2010-fall 2013)
  – Pan-European Research network on Transactional Memories
  – 56 experts, 42 institutions, 12 countries
Outline

• Overview of the Cloud-TM project

• Software Transactional Memories (STMs)

• Data Replication Protocols for STMs
  – No one size fits all solution

• Self-Optimizing Replication Protocols:
  – AB-based certification protocols
  – Single vs Multi-master schemes
Cloud-TM at a glance

Partners:

- INESC ID (PT)
- Algorithmica (IT)
- C.I.N.I. (IT)
- Red Hat (IE)

Project coordinator:
Paolo Romano, INESC ID (PT)

Duration:
From June 2010 to May 2013

Programme:
FP7-ICT-2009-5 – Objective 1.2

Further information:
http://www.cloudtm.eu
Project Motivations

- Cloud computing is at the peak of its hype...

- How to materialize the vision and maximize actual productivity?
Key Goals

Develop an open-source middleware platform for the Cloud:

1. Providing a simple and intuitive programming model:
   - hide complexity of distribution, persistence, fault-tolerance
   - let programmers focus on differentiating business value

2. Minimizing administration and monitoring costs:
   - automate elastic resource provisioning based on applications QoS requirements

3. Minimize operational costs via self-tuning
   - maximizing efficiency adapting consistency mechanisms upon changes of workload and allocated resources
Background on the Cloud-TM Programming Paradigm....

TRANSACTIONAL MEMORIES
The era of free performance gains is over

- Over the last 30 years:
  - new CPU generation = free speed-up

- Since 2003:
  - CPU clock speed plateaued...
  - but Moore’s law chase continues:
    - Multi-cores, Hyperthreading...

FUTURE IS PARALLEL
Fine grained locking?

• Simple grained locking is a **conundrum:**
  – need to reason about deadlocks, livelocks, priority inversions:
    • complex/undocumented lock acquisition protocols
    • scarce composability of existing software modules

... and a **verification nightmare:**
  • subtle bugs that are extremely hard to reproduce

• Make parallel programming **accessible to the masses!**
Transactional memories

• Key idea:
  – hide away synchronization issues from the programmer
  – replace locks with atomic transactions:
    • avoid deadlocks, priority inversions, convoying
    • way simpler to reason about, verify, compose
    • deliver performance of hand-crafted locking via speculation (+HW support)
An obvious evolution

- Real, complex STM based applications are starting to appear:
  - Apache Web Server
  - FenixEDU
  - Circuit Routing
  - ...

- ...and are being faced with classic production environment’s challenges:
  - scalability
  - high-availability
  - fault-tolerance

Distributed STMs
Distributed STMs

• At the convergence of two main areas:

>70% xacts are 10-100 times shorter:
  • larger impact of coordination

2. Boost performance by batching any remote synchronization during the commit phase
The Cloud-TM Programming Paradigm: Elastic Distributed Transactional Memory

- Elastic scale-up and scale-down of the DTM platform:
  - data distribution policies minimizing reconfiguration overhead
  - auto-scaling based on user defined QoS & cost constraints

- Transparent support for fault-tolerance via data replication:
  - self-tuning of consistency protocols driven by workload changes

- Language level support for:
  - transparent support of object-oriented domain model (incl. search)
  - highly scalable abstractions
  - parallel transaction nesting in distributed environments
Data replication

• Essential for in-memory data platforms for:
  – Performance
  – Fault-tolerance
• Performance
  – Read operations on local data
• Fault-tolerance
  – Ensure data availability in presence of crashes
Challenge

• Distributed coordination when:
  – The transaction commits (all-or-none the copies must be updated)
  – But also for ensuring same serialization order across all replicas!
Toolbox for Replication

• Atomic Commitment
• Reliable Broadcast
• Atomic Broadcast
Atomic Commitment

• Set of nodes, each node has input:
  – CanCommit
  – MustAbort

• All nodes output same value
  – Commit
  – Abort

• Commit is only output if all nodes CanCommit
2-phase commit

**coordinator**

- prepare msg

**participant**

- validate/ acquire locks
- vote msg (Yes or No)

**participant**

- validate/ acquire locks
- decision msg (Commit or Abort)

- apply decision

- apply decision
Toolbox for Replication

• Atomic Commitment
• Reliable Broadcast
• Atomic Broadcast
(Uniform) Reliable Broadcast

- Allows to broadcast a message \( m \) to all replicas
- If a process delivers \( m \), every correct node will deliver \( m \)
- Useful to propagate updates
Toolbox for Replication

- Atomic Commitment
- Reliable Broadcast
- Atomic Broadcast
Atomic Broadcast

- Reliable broadcast with total order
- If replica R1 receives $m_1$ before $m_2$, any other replica $R_i$ also receives $m_1$ before $m_2$
- Can be used to allow different nodes to obtain locks in the same order
Sequencer-based ABcast

R1: sequencer
Assigns SN

R2
Sends message
Receive in order
Receive in order
Commit order
Commit order
Commit order

R3
Receive in order
Commit order
Commit order
Commit order

2nd Workshop on Software Services, Timisoara, Romania, 6 June 2011
CLASSIC PROTOCOLS FOR TRANSACTIONAL REPLICATION
Classic Replication Protocols

• Focus on full replication protocols

Single master
(primary-backup)

Multi master

AB-based

2PC-based

State machine replication

Certification

Non-voting

BFC

Voting
Classic Replication Protocols

• Focus on full replication protocols

- Single master (primary-backup)
- Multi master
  - AB-based
  - 2PC-based
- Certification
  - Non-voting
  - BFC
- State machine replication
  - Voting
Single Master

• Write transactions are executed entirely in a single replica (the primary)
• If the transaction aborts, no coordination is required.
• If the transaction is ready to commit, coordination is required to update all the other replicas (backups).
  – Reliable broadcast primitive.
• Read transactions can be executed on backup replicas.

- No distributed deadlocks
- No distributed coordination during commit

• Throughput of write txs doesn’t scale up with number of nodes
Classic Replication Protocols

• Focus on full replication protocols

Single master (primary-backup)

Multi master

AB-based

2PC-based

Certification

Non-voting

BFC

State machine replication

Voting
Multi master replication

• Write and read transactions can be processed anywhere

• Access Synchronization:
  – Eager: upon each access (bad bad bad performance)
  – Lazy: at commit time

• Lazy multi-master are classifiable as:
  – 2PC-based
  – AB-based
Classic Replication Protocols

- Focus on full replication protocols

Single master (primary-backup)

Multi master

AB-based

2PC-based

Certification

Non-voting

BFC

State machine replication

Voting
2PC-based vs AB-based

2PC-based replication
• Transactions attempt to acquire atomically locks at all nodes
• 2PC materializes conflicts among remote transactions generating:

DISTRIBUTED DEADLOCKS

+ good scalability at low conflict
- thrashes at high conflict

AB-based replication
• family of (distributed) deadlock free algorithms
• Serialize transactions in the total order established by AB

+ strong gains at high conflict rates
- AB latency typically higher than 2PC
Classic Replication Protocols

• Focus on full replication protocols

- Single master (primary-backup)
- Multi master
  - AB-based
  - 2PC-based
- Certification
  - Non-voting
  - Voting
- State machine replication
  - BFC
State-machine replication

• All replicas execute the same set of transactions, in the same order.
• Transactions are shipped to all replicas using total order broadcast.
• Replicas receive transactions in the same order.
• Replicas execute transaction by that order.
  – Transactions need to be deterministic!
State-machine replication

Transaction never abort

Write transactions fully executed by all replicas: low scalability

+ transaction never abort

- Write transactions fully executed by all replicas: low scalability
Certification
(a.k.a. deferred update)

• A transaction is executed entirely in a single replica.
• Different transactions may be executed on different replicas.
• If the transaction aborts, no coordination is required.
• If the transaction is ready to commit, coordination is required:
  – To ensure serializability
  – To propagate the updates
Certification

• Two transactions may update concurrently the same data in different replicas.

• Coordination must detect this situation and abort at least one of the transactions.

• Three alternatives:
  – Non-voting algorithm
  – Voting algorithm
  – BFC
Classic Replication Protocols

• Focus on full replication protocols

- Single master (primary-backup)
  - Certification
    - Non-voting
  - BFC
- Multi master
  - AB-based
  - State machine replication
  - 2PC-based
  - Voting
Non-voting

• The transaction executes locally.
• When the transaction is ready to commit, the read and write set are sent to all replicas using total order broadcast.
• Transactions are certified in total order.
• A transaction may commit if its read set is still valid (i.e., no other transaction has updated the read set).
Non-voting

Execution
Transaction T1

Execution
Transaction T2

TOB of T1’s
read & writeset

TOB of T2’s
read & writeset

Validation&Commit
T1

Validation&Commit
T1

Validation&Abort
T2

Validation&Abort
T2

+ only validation executed at all replicas:
high scalability with write intensive workloads

- need to send also readset: often very large!
• Focus on full replication protocols

- Single master (primary-backup)
- Multi master
  - AB-based
  - 2PC-based
- Certification
  - Non-voting
  - BFC
- Voting
- State machine replication
Voting

• The transaction executes locally at replica R

• When the transaction is ready to commit, only the write set is sent to all replicas using total order broadcast

• Commit requests are processed in total order

• A transaction may commit if its read set is still valid (i.e., no other transaction has updated the read set):
  – Only R can certify the transaction!

• R send the outcome of the transaction to all replicas:
  – Reliable broadcast
Voting

- sends only write-set (much smaller than read-sets normally)
- Additional communication phase to disseminate decision (vote)
Classic Replication Protocols

• Focus on full replication protocols

Single master
(primary-backup)

Multi master

AB-based

2PC-based

Certification

State machine replication

Non-voting

Voting

BFC
Bloom Filter Certification (BFC)

• Bloom filters:
  – space-efficient data structure for test membership queries
  – Probabilistic answer to “Is elem contained in BF?”
    • No false negatives: A “no” answer is always correct
    • False positives: A “yes” answer may be false
  – Compression is a function of a (tunable) false positive rate

• Key idea:
  – encode readset in a BF and test if any of the items written by concurrent transactions results in BF:
    • False positives: additional (deterministic) abort
  – strongly reduce network traffic:
    • 1% false positive up to 30x compression
BFC vs Voting vs Non-Voting

+ optimal for “medium sized” readsets
- suboptimal for large and small readset sizes

![Graph showing throughput for different readset sizes for NVC, BFC, and VC]
2PC-based vs single master

![Graph showing throughput vs number of nodes for PB and 2PC in low and high conflict scenarios.](image-url)
Summing up

• Existing solutions are optimized for specific workload/scale scenarios
Autonomic adaptation at play

- **low resources:**
  - minimum costs
  - primary-backup:
    - low % write:
      - low load on primary

- **auto-scale up:**
  - new nodes hired for read-only requests
  - primary-backup:
    - low % write:
      - primary stands the load

- **multi-master:**
  - hi % write:
    - primary overwhelmed
  - higher scalability

- **auto-scale down:**
  - minimum costs
  - switch back to primary-backup

Legend:
- : node processing read-only requests
- : node processing read&write requests

- **low traffic**
  - read-dominated
  - low conflict

- **hi traffic**
  - read-dominated
  - low conflict

- **hi traffic**
  - write dominated
  - low conflict

- **low traffic**
  - read dominated
  - low conflict

High % write

Low % write

Low write transactions

Hi % write

Low traffic

Hi traffic

Low traffic

Time
Self-optimizing replication

- Entails devising solutions to 2 keys issues:
  - Allow coexistence/efficient switch among multiple replication protocols:
    - Avoid blocking transaction processing during transitions
  - Determine the optimal replication strategy given the current (or foreseen) workload characteristics:
    - machine learning methods (black box)
    - analytical models (white box)
    - hybrid analytical/statistical approaches (gray box)
Two case studies

Certification Schemes
NVC vs VC vs BFC

Single vs multi-master
2PC vs PB

joint work with
M. Couceiro, and L. Rodrigues

joint work with
D. Didona, S. Peluso and F. Quaglia
Two case studies

Certification Schemes
NVC vs VC vs BFC

Single vs multi-master
2PC vs PB

joint work with
M. Couceiro, and L. Rodrigues

joint work with
D. Didona, S. Peluso and F. Quaglia

2nd Workshop on Software Services, Timisoara, Romania, 6 June 2011
Certification Schemes
NVC vs VC vs BFC

• The famous 2 keys issues:
  – Allow coexistence of multiple certification schemes via the Polymorphic Certification (PolyCert) protocol:
    • simultaneous presence of txs using NVC/VC/BFC

  – Determine the optimal replication strategy depending on workload characteristics:
    • machine learning methods to predict certification latency
      – off-line: decision-trees, neural network, SVM
      – on-line: reinforcement learning (UCB)
Fig. 3. Architectural Overview wSingle Node Perspectivex

Collecting historical data about the execution of past transactions to improve the selection process. For this, it exports an interface that allows the PRM to register the following information: (i) the commit time experienced by a transaction, and (ii) the certification protocol that it used. This allows the RPSO to gather, store, and analyze whether online or offline statistical data on the distribution of the commit time of transactions.

The PRM also interacts with two other components, the Group Communication System (GCS) and a local Key Value Store (KVS). The GCS, Appia in our implementation, provides a number of communication abstractions required by the PRM: view synchronous membership, AB, and URB. Finally, the Key Value Store is a weak hash map, used to maintain the mapping between application-level transactional objects, namely, containing JVSTM VBoxes and replica-wide unique object identifiers, which are generated automatically by our framework upon creation of a new transactional object. More in detail, an entry of the key-value store contains the unique object identifier as its key and a weak reference to the local transactional object as its value. This information is used by the PRM upon reception of a commit request for a remote transaction to retrieve in the local JVSTM instance the objects that were read, updated by the transaction during its remote execution.

The usage of a weak hash map ensures that Java Garbage Collector is not prevented from discarding the object referenced by a hash map entry whenever all application-level references to the object have been removed, thus avoiding any interference with the local JVM garbage collection mechanism.

PolyCert is designed to allow the simultaneous use of all the three AB-based certification protocols introduced before, namely NVC, BFC, and VC. In particular, the selection of each protocol is made on a transaction-by-transaction basis, and multiple concurrent transactions may use different protocols without endangering the correctness of the system.
Off-line ML techniques

- Per each transaction:
  - predict size of AB message $m$ for the various certification schemes
  - forecast AB latency for each message size. We evaluated several ML approaches:
    - decision trees → best results
    - neural networks
    - support vector machine
  - uses up to 53 monitored system attributes:
    - CPU
    - Memory
    - Network
    - Time-series
  - requires computational intensive training phase
On-line reinforcement learning

• Each replica builds on-line expectations on the rewards of each protocol:
  – no assumption on rewards’ distributions

• Solves the exploration-exploitation dilemma:
  – did I test this option sufficiently in this scenario?

• Distinguishes workload scenario solely based on read-set’s size
  – exponential discretization intervals to minimize training time

• Replicas exchange statistical information periodically to boost learning
Chasing the optimum...

![Normalized throughput of the adaptive and non-adaptive protocols](image)

**Fig. 5.** Normalized throughput of the adaptive and non-adaptive protocols. Transactions thus allowing us to focus on the performance of the transactions that require the activation of a commit-time certification phase. As shown in Figure 3 around 15% of transactions—namely the so-called long traversal transactions—in this benchmark have read-set sizes larger than 100K items. As a consequence, when using either NVC or BFC, this benchmark generates a very high traffic volume that, in all our runs, eventually determined the saturation and the collapse of the GCS. This is the reason why in Figure 8 we only report the throughput of VC, DT, UCB, and DistUCB normalized with respect to the throughput of the optimal non-adaptive protocol, namely VC. In this scenario, the adaptive protocols clearly outperform the non-adaptive VC scheme, thanks to their ability to use the more efficient NVC and BFC protocols to handle transactions with smaller read-set sizes. The speedup of PolyCert when using the three alternative oracles ranges from 2 to 3, with the best performance also in this case achieved by DistUCB.

Overall, our experimental data demonstrated the effectiveness and viability of the proposed self-tuning polymorphic replication technique, highlighting in particular the efficiency of the DistUCB oracle, which, not needing any time-consuming offline training phases and being totally parameter-free, results extremely convenient for deployment in real-life practical scenarios. Interestingly, PolyCert does not only provide benefits in terms of performance, but also in terms of robustness, avoiding to saturate the GCS in presence of transactions with extremely large read-sets, a main source of instability for BFC and, in particular, NVC.

**Related Work**

Our work is clearly related to the vast literature on replication of transactional systems and in particular to the more recent works relying on AB to achieve a replica-wide...
...and beating it!

![Bar chart showing normalized throughput comparison between VC, DT, UCB, and DistUCB protocols.](chart.png)
Project’s timeline

Dec 2010
- User requirements
  - Start
  - Enabling Technologies

June 2011
- Architecture Specification Draft
  - Preliminary Prototype of RDSTM & RSS

Dec 2012
- Initial Pilot Applications
  - Prototype of RDSTM & RSS
  - Prototype of the Autonomic Manager

Mar 2013
- Final Pilot Applications
  - Final CloudTM Prototype

June 2013
- Demonstration
- Evaluation

Internet of Services Collaboration Meeting – Bruxelles 19-20 October 2010
Opportunities for collaboration

• Standards/tools to specify and negotiate SLAs
  – focus in Cloud-TM is on performance, reliability and cost

• Tools for monitoring provided QoS

• Auto-scaling/proactive reconfiguration:
  – challenging goal common to very projects
  – in Cloud-TM we will target data intensive applications

• Achieve interoperability with storage solutions for the cloud developed by other projects
Conclusions

• Cloud computing raises a number of research challenges for transactional replication:
  – elasticity:
    • self-tuning as an essential requirement
  – non-uniform transaction synchronization costs:
    • multi-core rack data-center cloud federation
  – unprecedented scalability challenge
THANKS FOR THE ATTENTION

Q&A

Webpage: www.cloudtm.eu  Contact: romano@inesc-id.pt