Cloud·TM

A novel programming paradigm for the Cloud

SEVENTH FRAMEWORK PROGRAMME

Autonomic mechanisms for transactional replication in elastic cloud environments

Paolo Romano

inesc id lisboa

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-Jun2012)
  - 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)

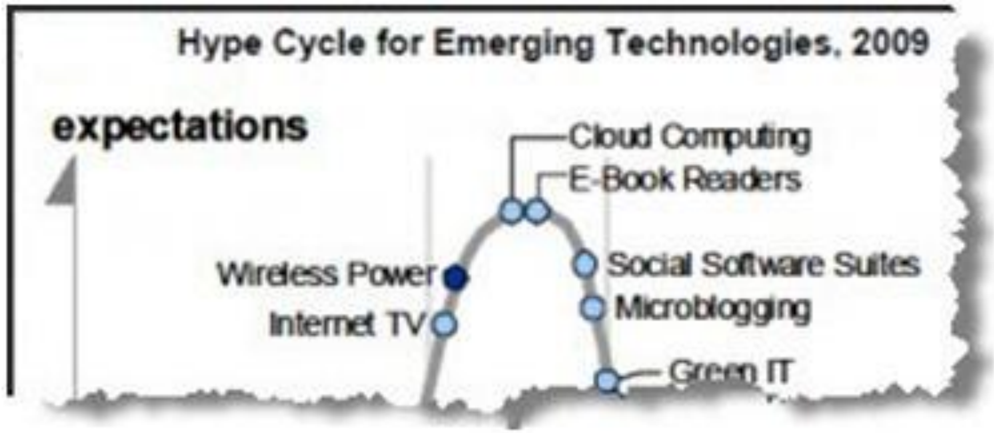
Duration:
From June 2010 to May 2013

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…



SIMPLIFYING THE DEVELOPMENT AND ADMINISTRATION OF CLOUD APPLICATIONS

- 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

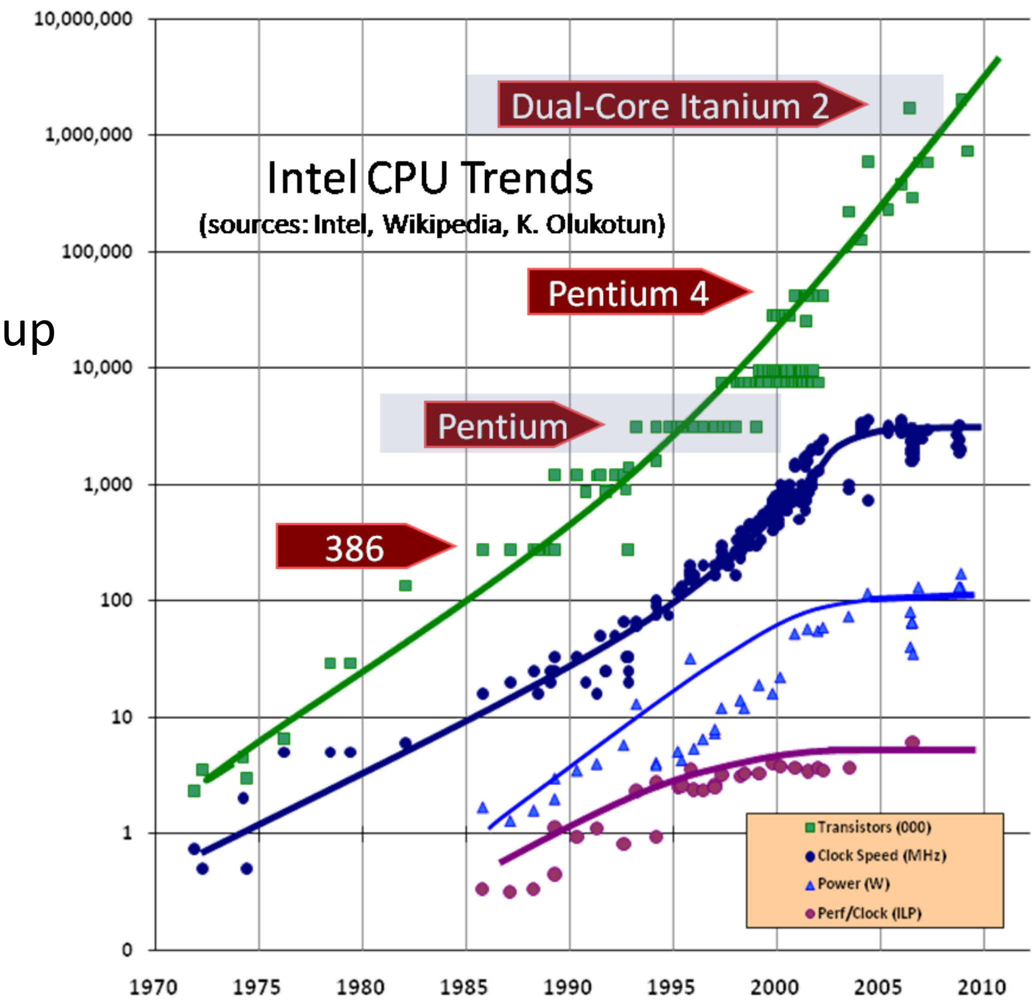
Background on the Cloud-TM Programming Paradigm….

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 acquistion 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

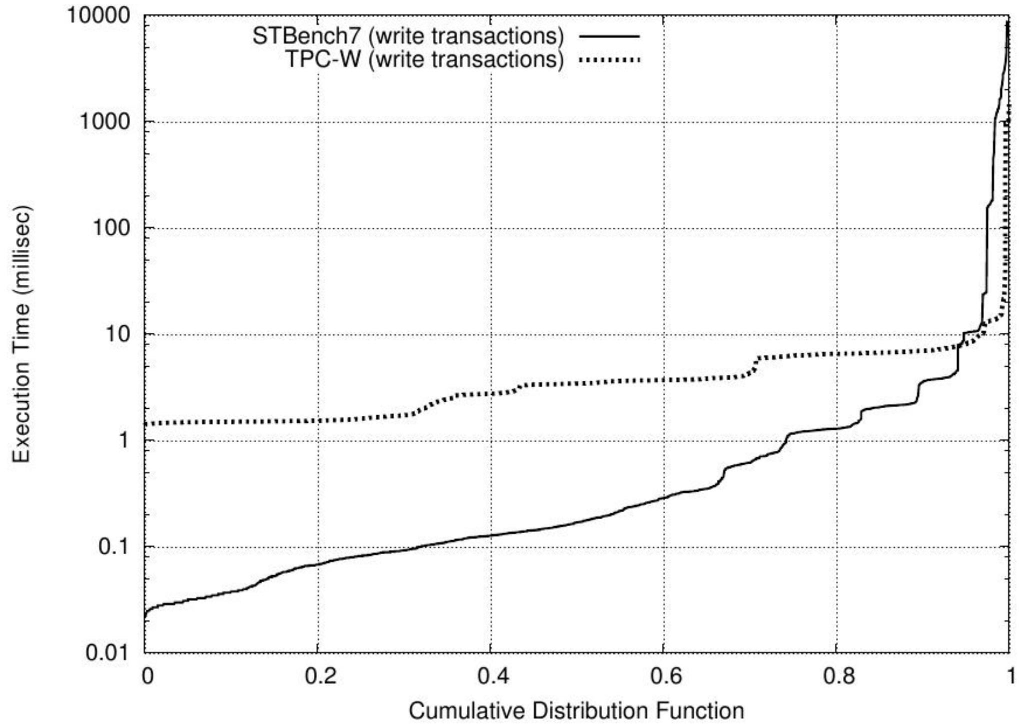
- Real, complex STM based applications are starting to appear:
  - Apache Web Server
  - FenixEDU
  - Circuit Routing
  - …
- …and are being faced with classic production environment's challenges:
  - scalability
  - high-availability
  - fault-tolerance

$\Big\}$ **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

unique, challenging requirements!

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!

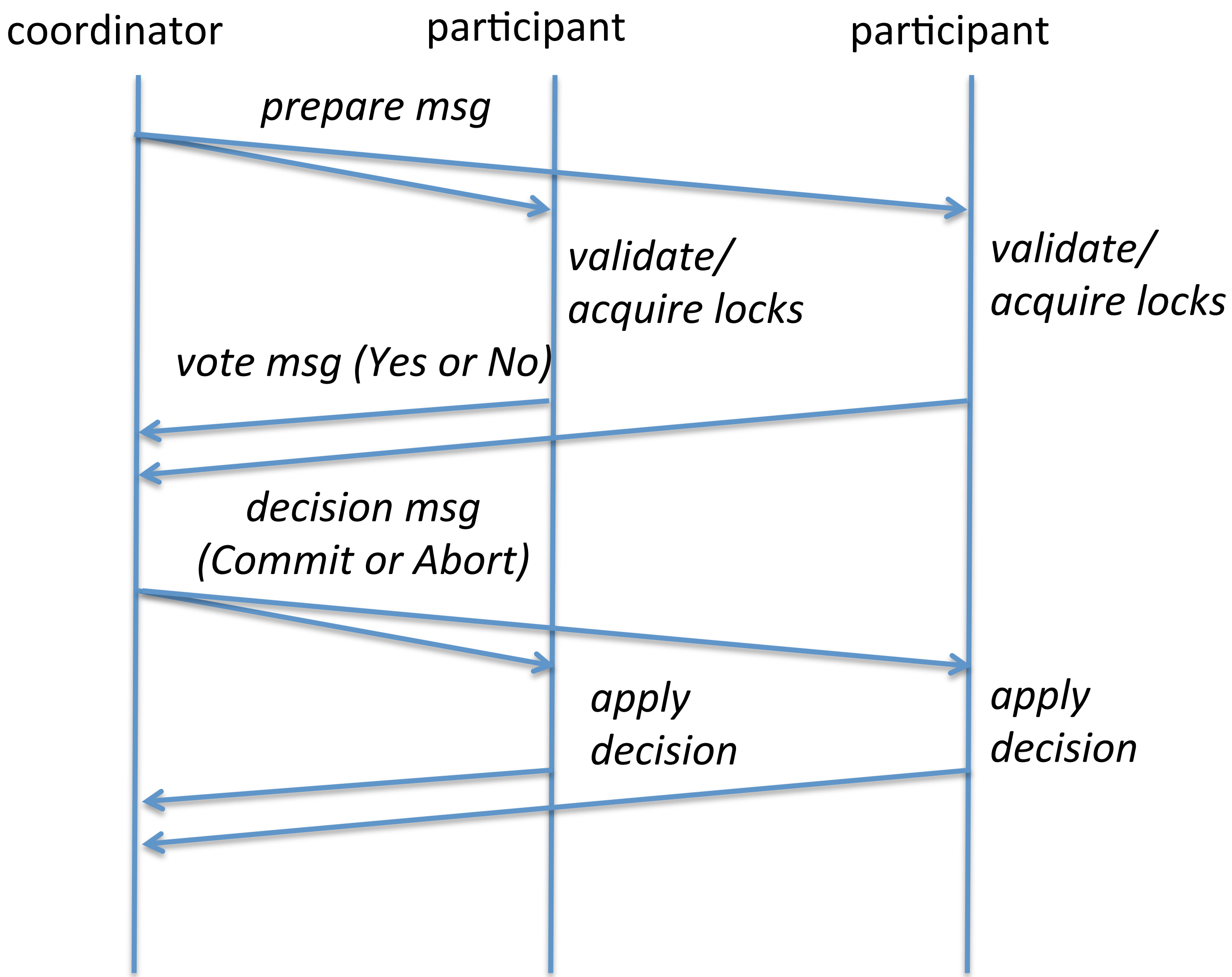
Toolbox for Replication

- **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



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

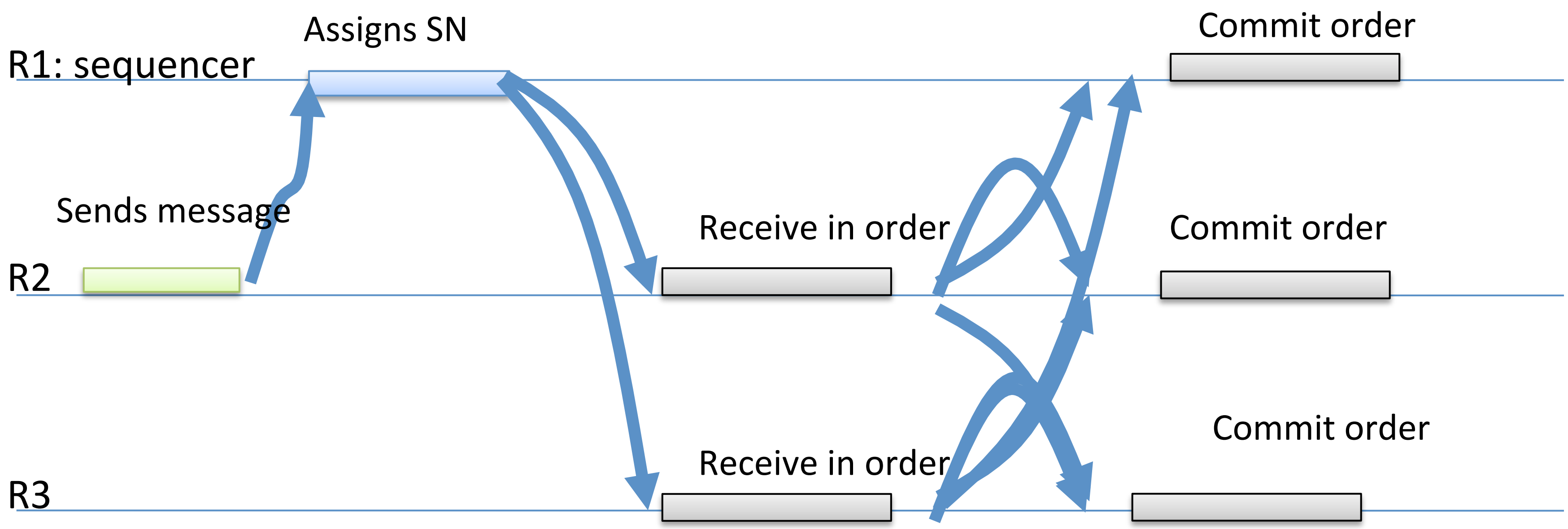
Toolbox for Replication

- Atomic Commitment
- Reliable Broadcast
- **Atomic Broadcast**

Atomic Broadcast

- Reliable broadcast with total order
- If replica R1 receives **m1** before **m2**, any other replica Ri also receives **m1** before **m2**
- Can be used to allow different nodes to obtain locks in the same order

Sequencer-based ABcast



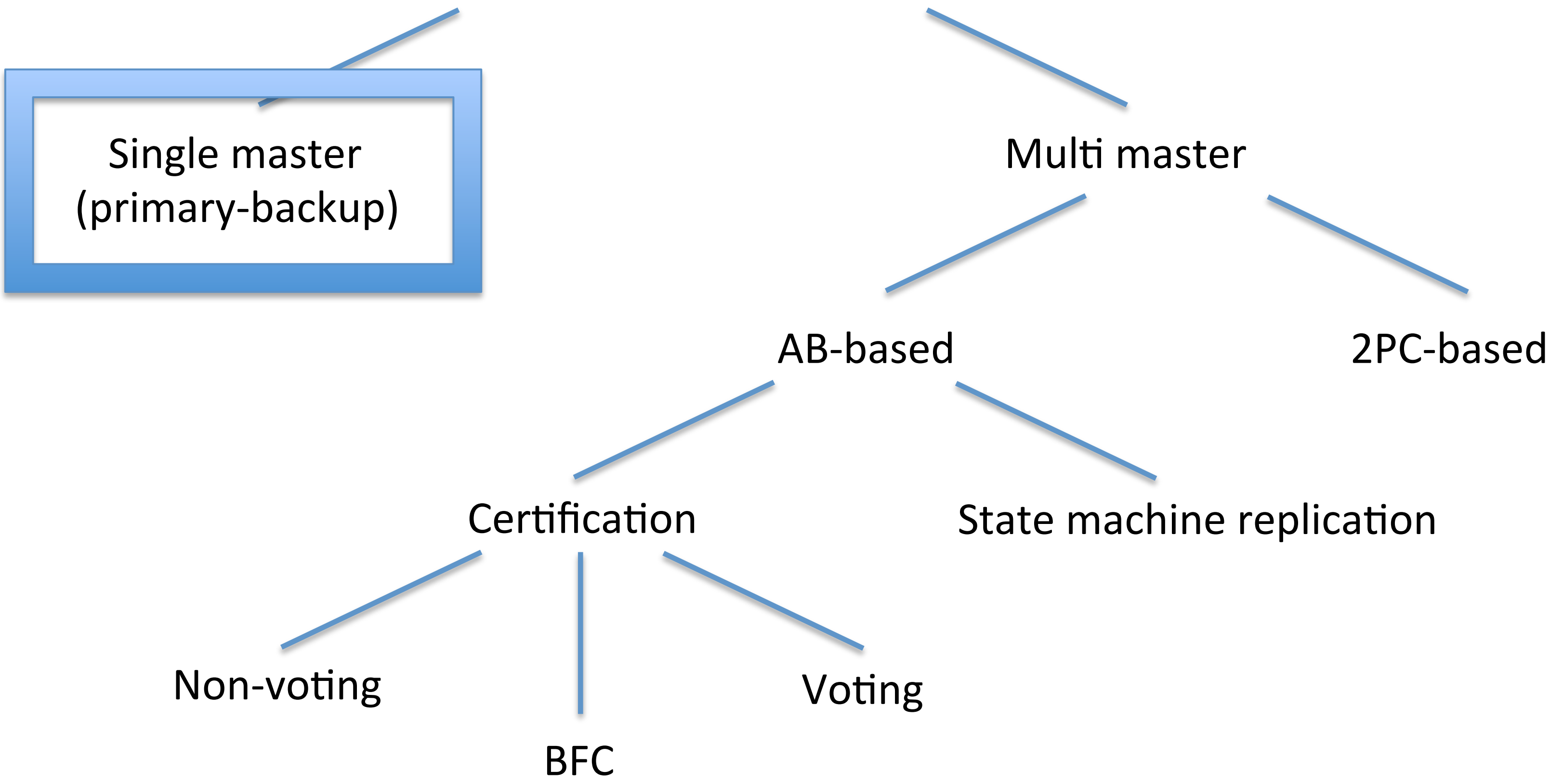
CLASSIC PROTOCOLS FOR TRANSACTIONAL REPLICATION

Classic Replication Protocols

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

Classic Replication Protocols

- Focus on full replication protocols



Single Master

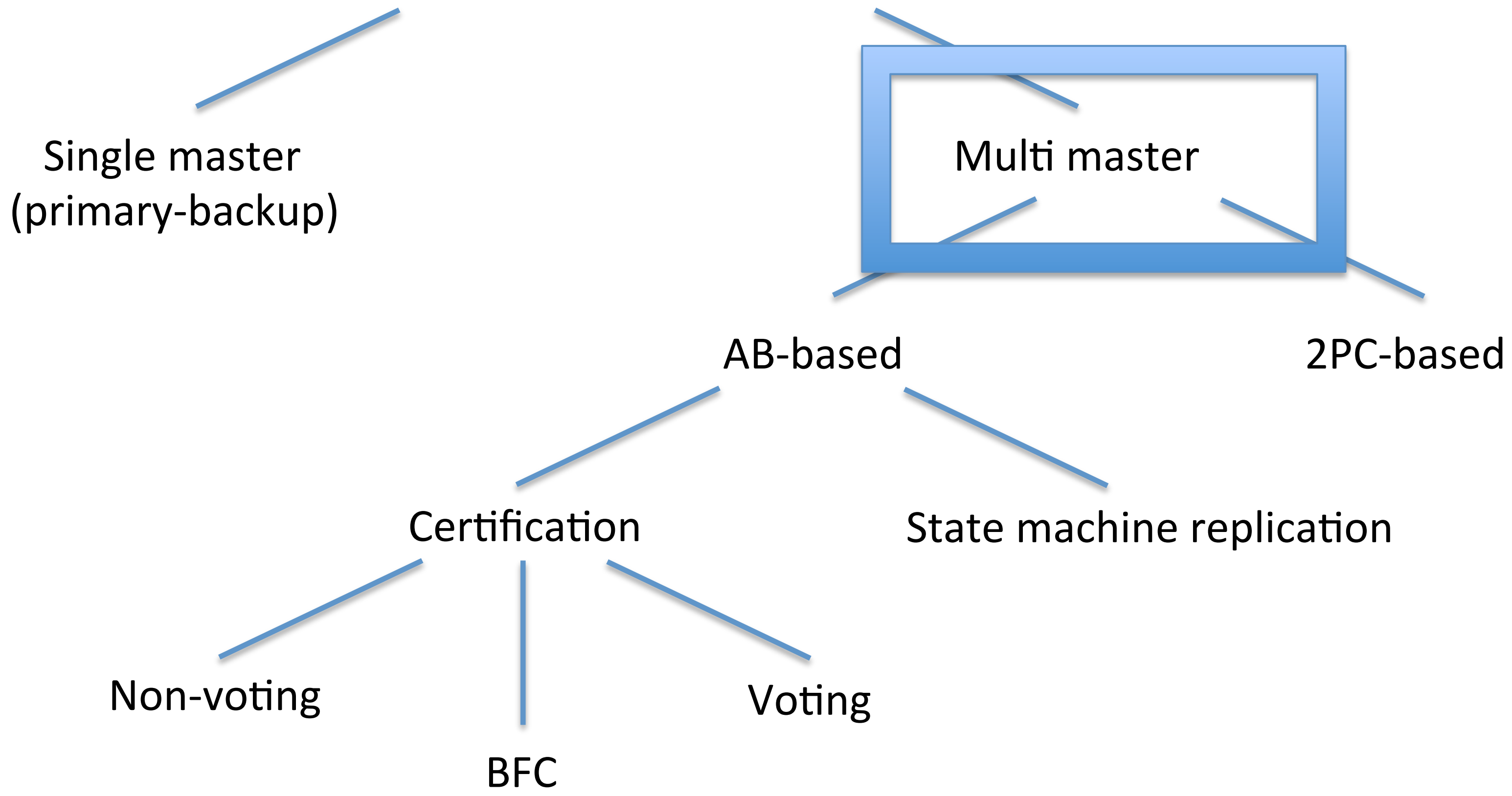
- Write transactions are executed entirely in a single replica (the primary)
- If the transaction aborts, no coordination is required.
- If the transaction is ready to commit, coordination is required to update all the other replicas (backups).
  - Reliable broadcast primitive.
- Read transactions can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

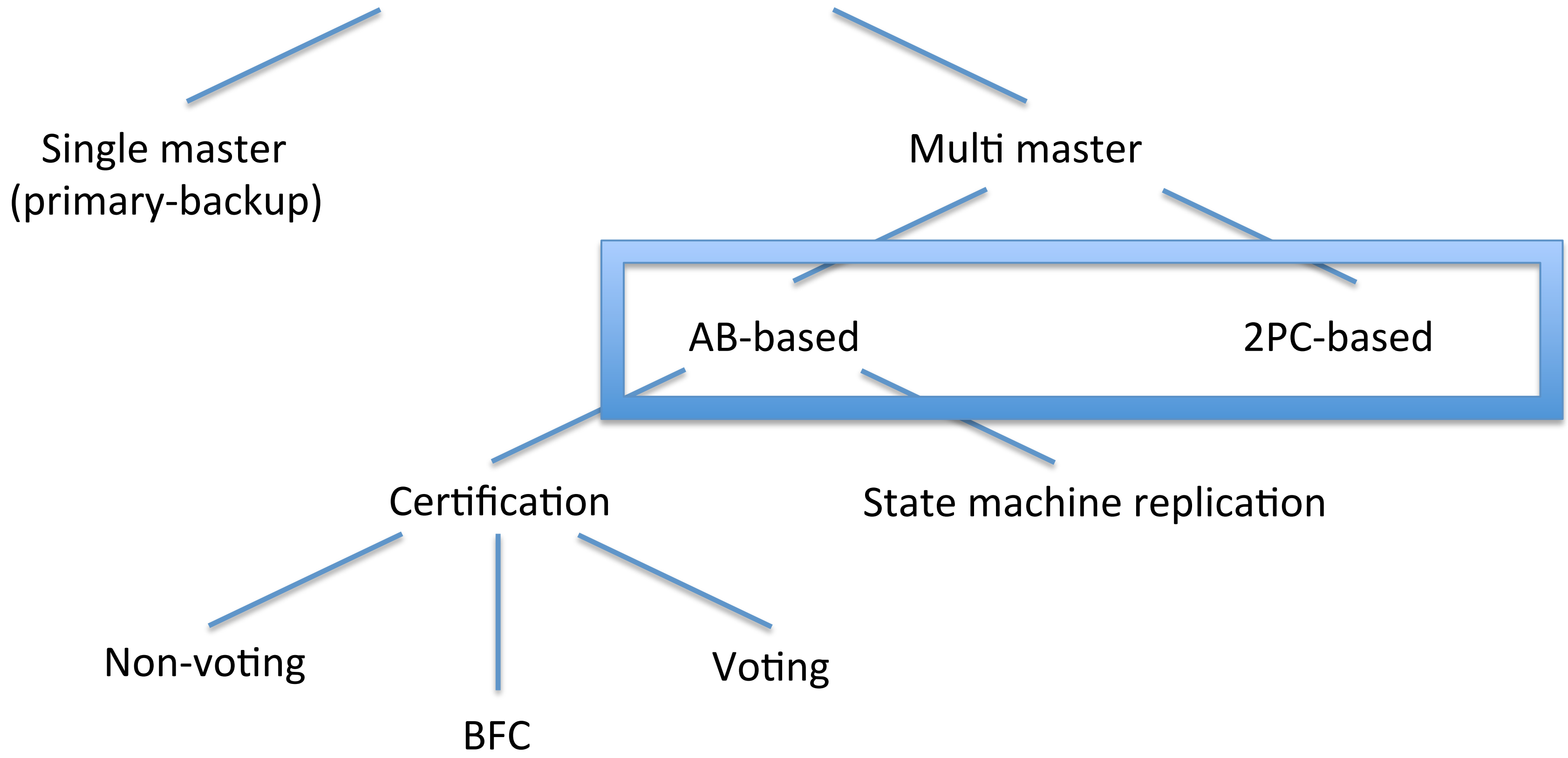


Multi master replication

- Write and read transactions can be processed anywhere
- Access Synchronization:
  - **Eager: upon each access (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



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

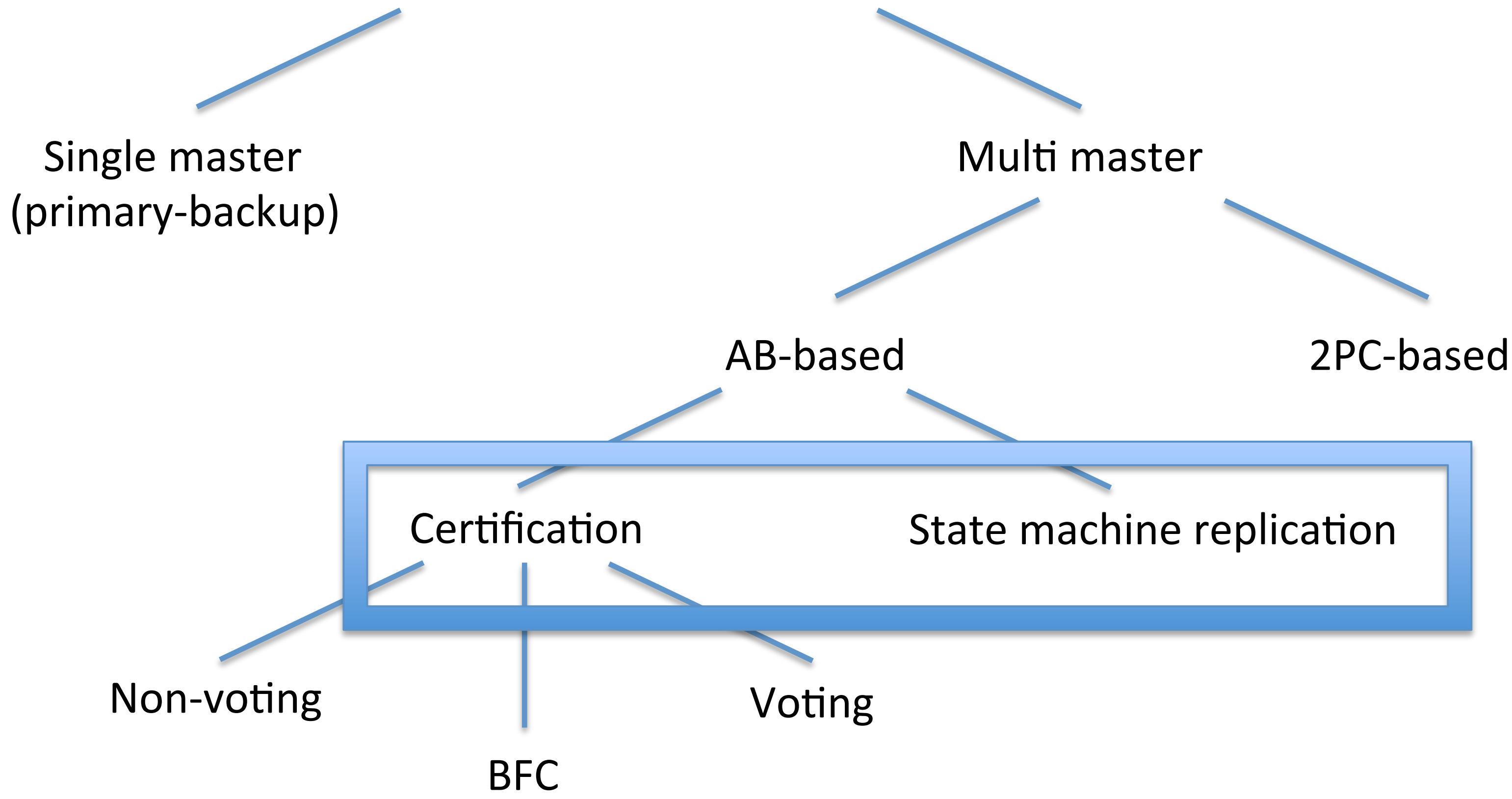
- family of (distributed) deadlock free algorithms
- Serialize transactions in the total order established by AB

+ strong gains at high conflict rates

- AB latency typically higher than 2PC

Classic Replication Protocols

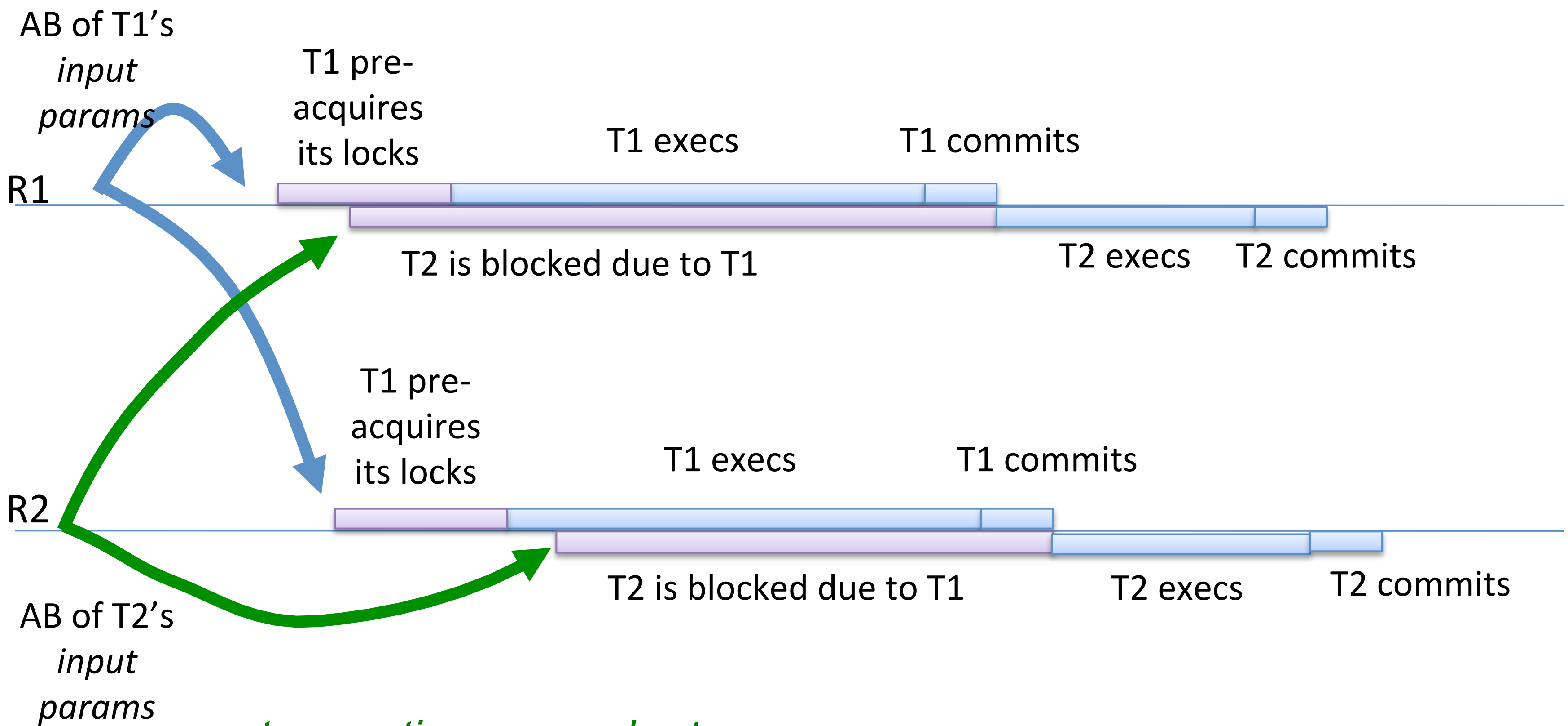
- Focus on full replication protocols



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

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

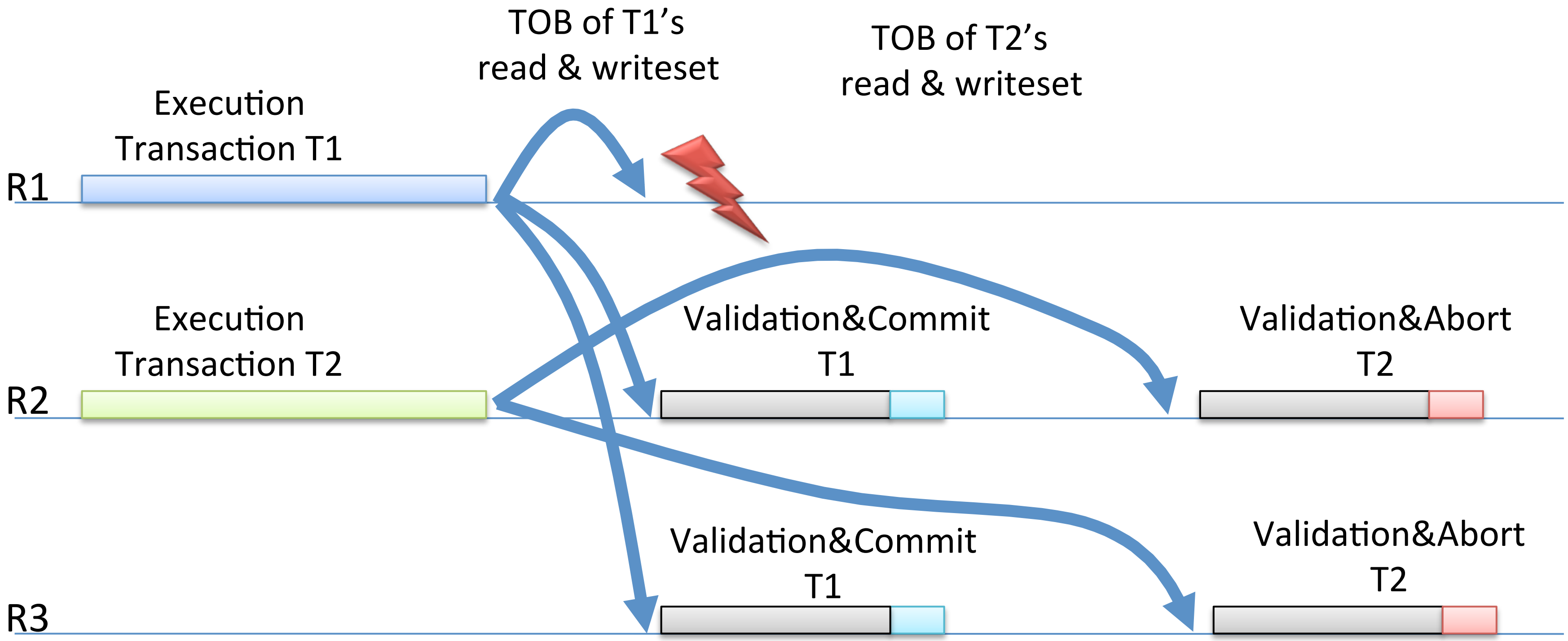
Classic Replication Protocols

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

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



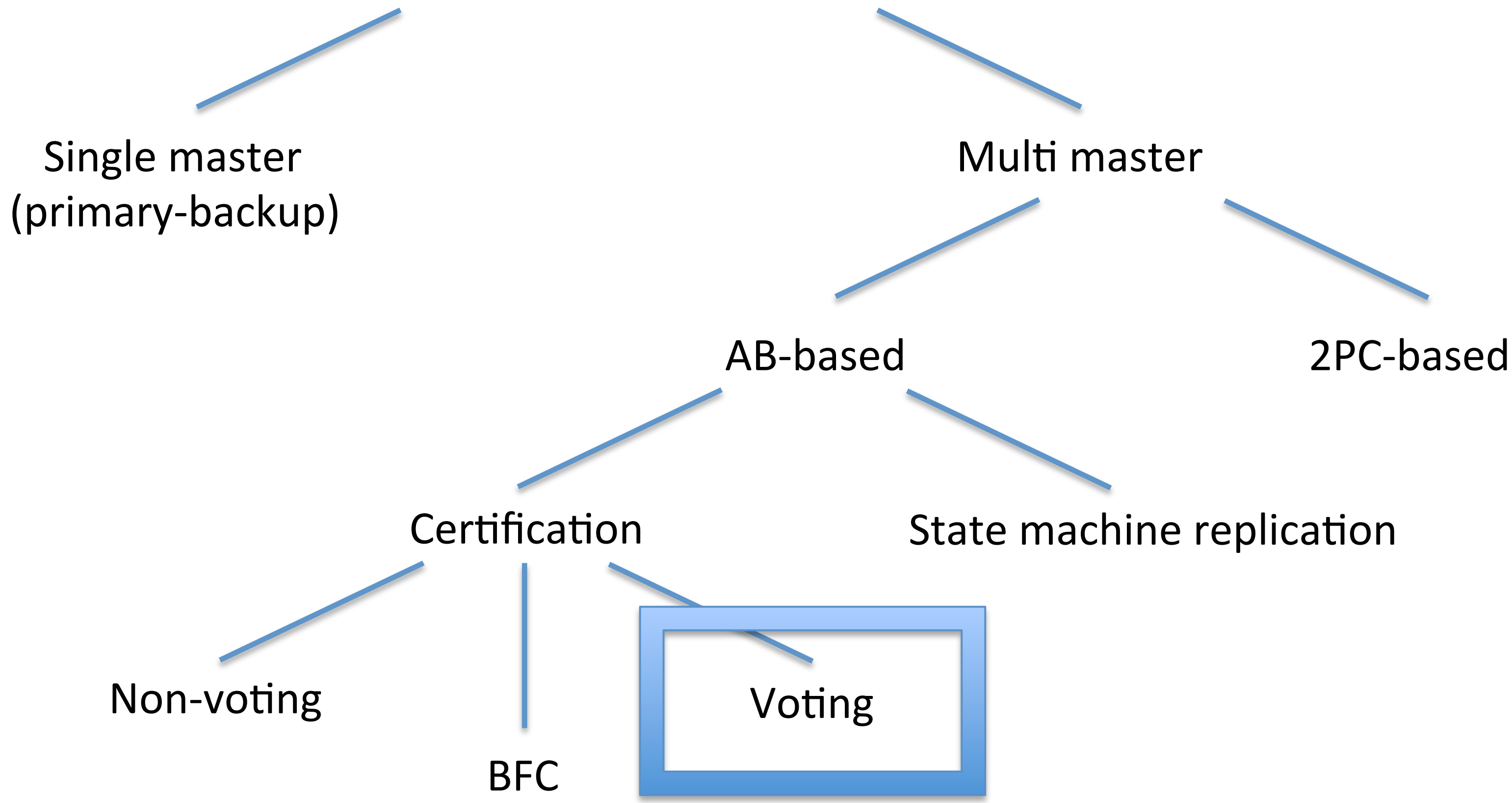
+ only validation executed at all replicas:

high scalability with write intensive workloads

- need to send also readset: often very large!

Classic Replication Protocols

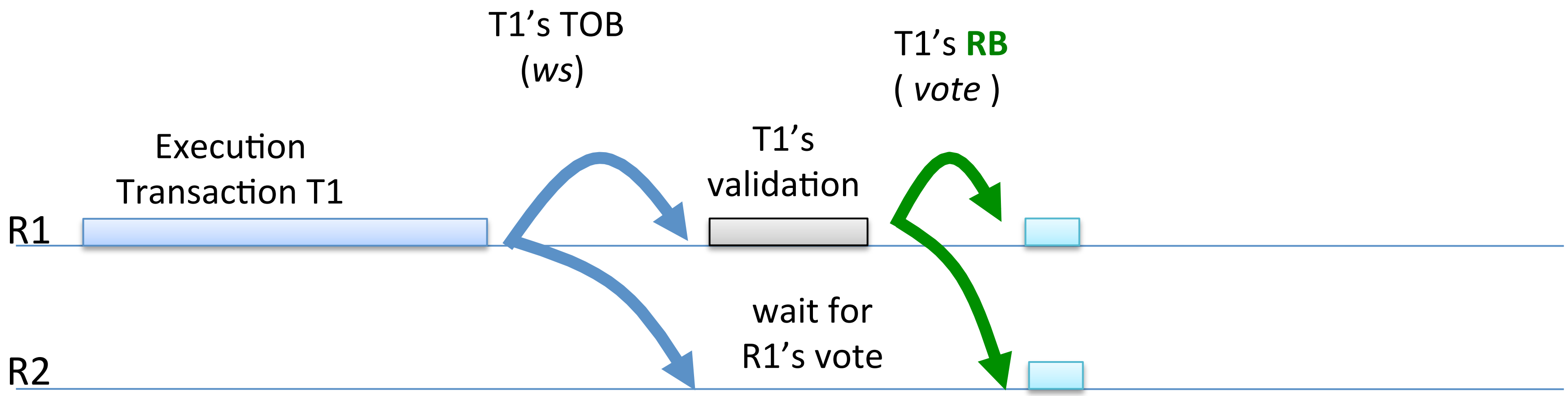
- Focus on full replication protocols



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

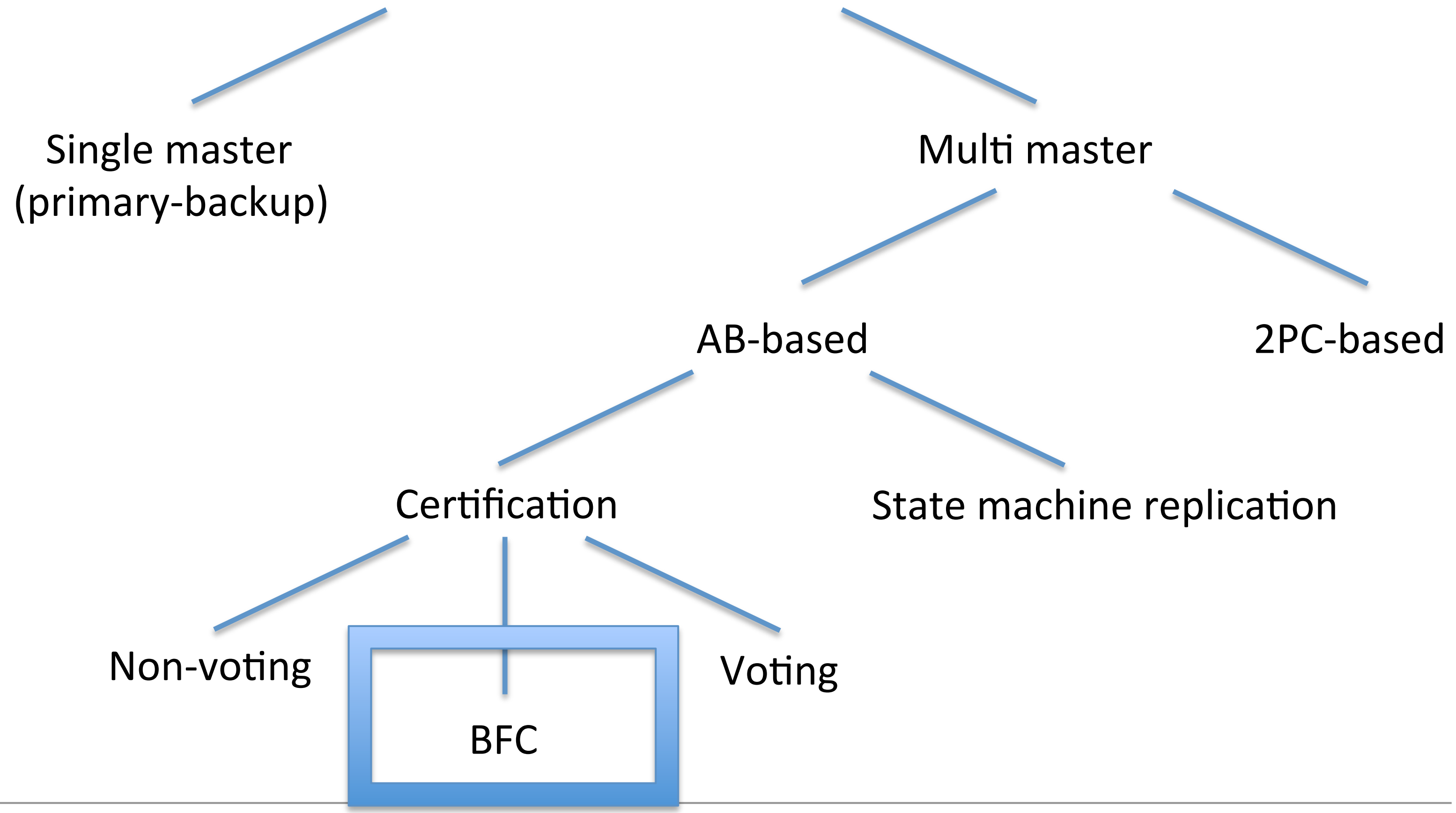


+ sends only write-set (much smaller than read-sets normally)

- Additional communication phase to disseminate decision (vote)

Classic Replication Protocols

- Focus on full replication protocols



Bloom Filter Certification (BFC)

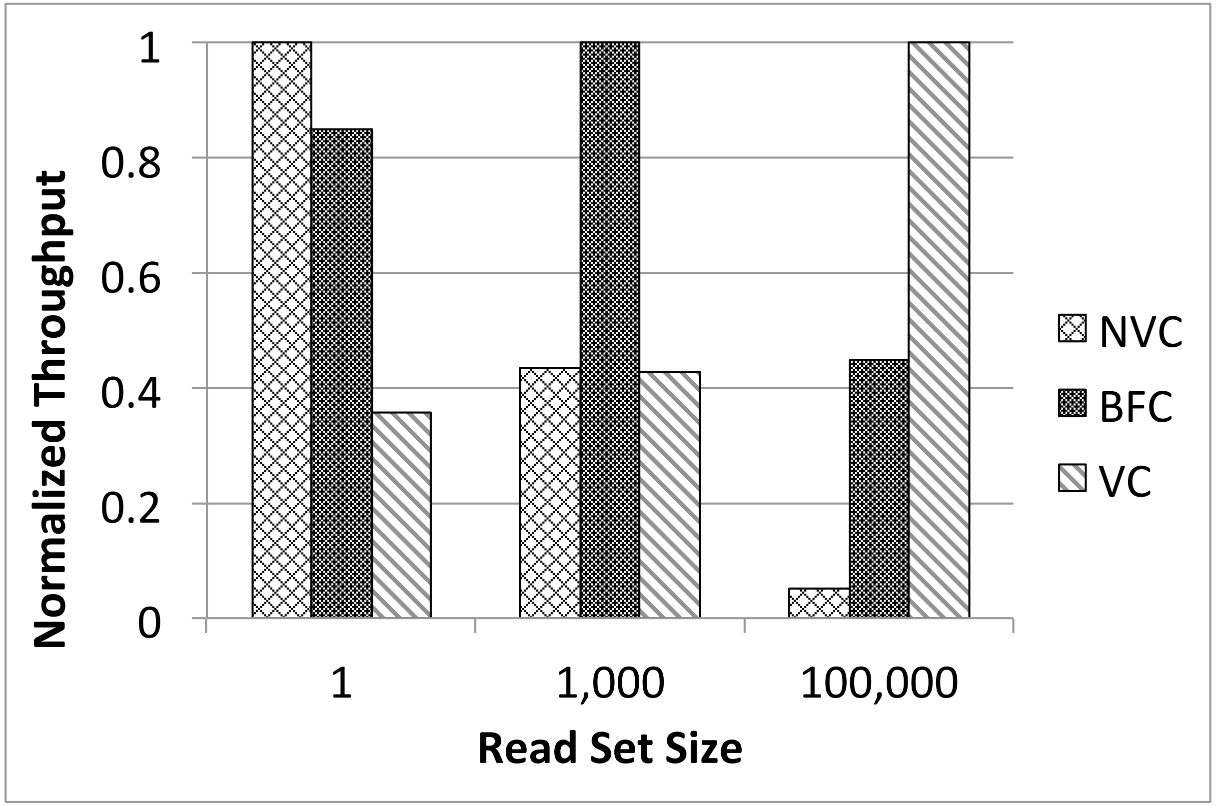
- Bloom filters:
  - space-efficient data structure for test membership queries
  - Probabilistic answer to "Is elem contained in BF?"
    - No false negatives: A "no" answer is always correct
    - False positives: A "yes" answer may be false
  - Compression is a function of a (tunable) false positive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

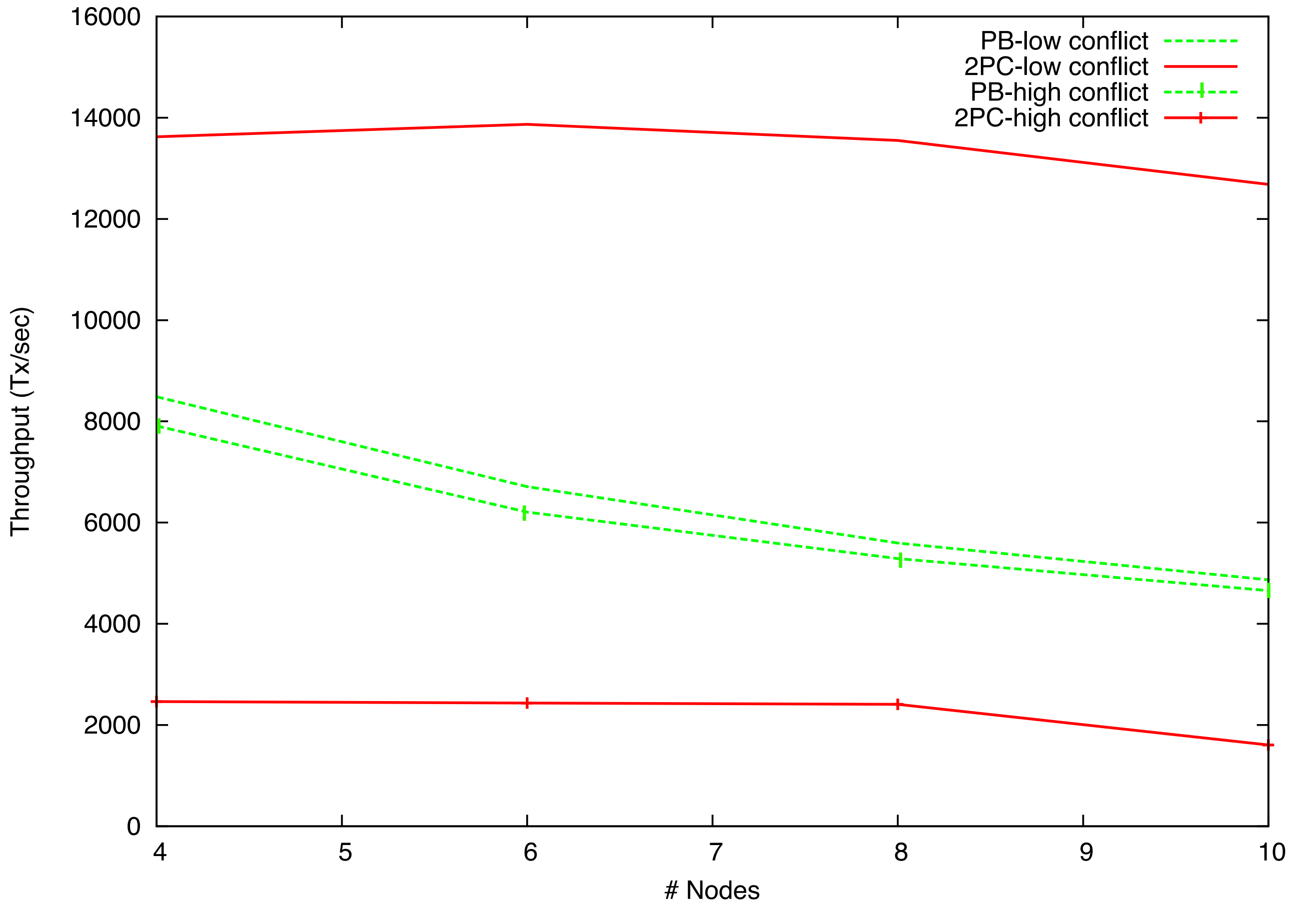
BFC vs Voting vs Non-Voting

+ optimal for "medium sized" readsets

- suboptimal for large and small readset sizes

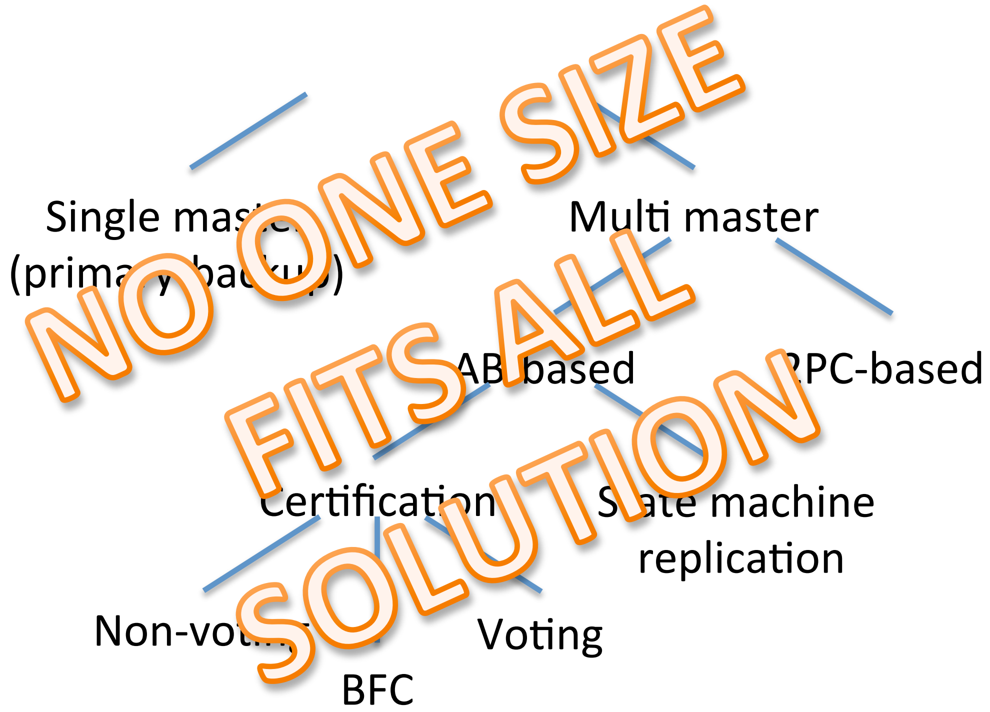


2PC-based vs single master

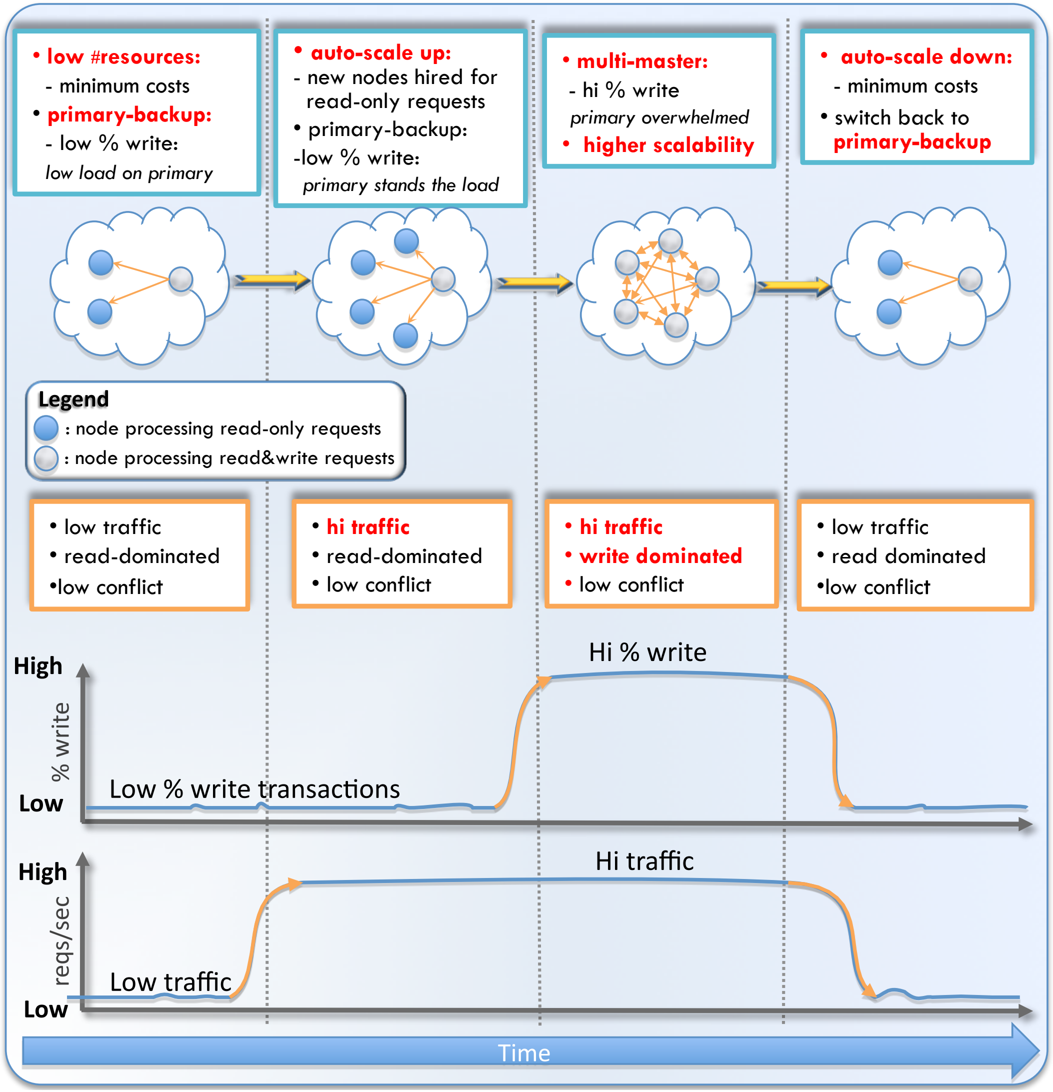


Summing up

- Existing solutions are optimized for specific workload/scale scenarios



Autonomic adaptation at play

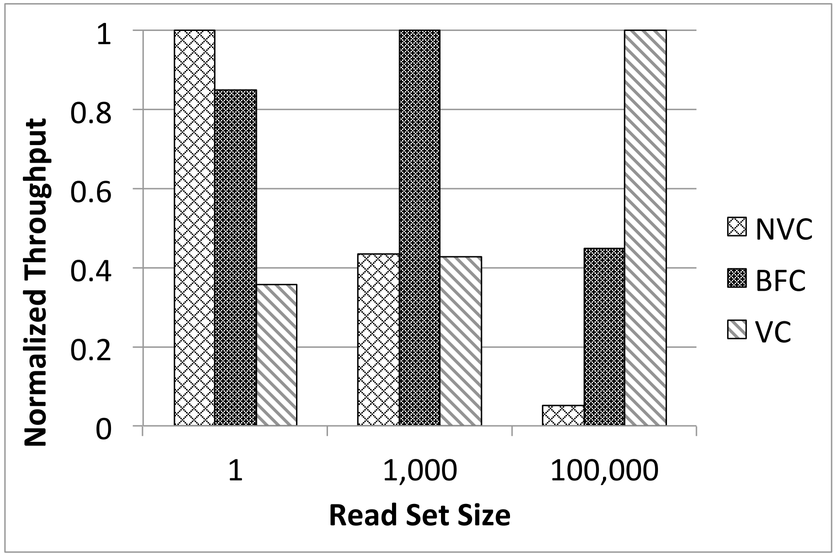


Self-optimizing replication

- 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)

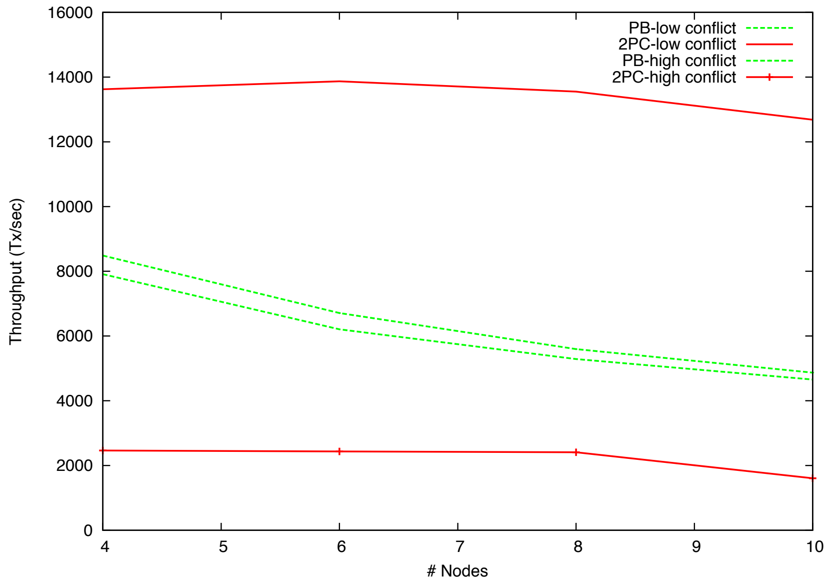
Two case studies

Certification Schemes
NVC vs VC vs BFC



joint work with
M. Couceiro, and L. Rodrigues

Single vs multi-master
2PC vs PB

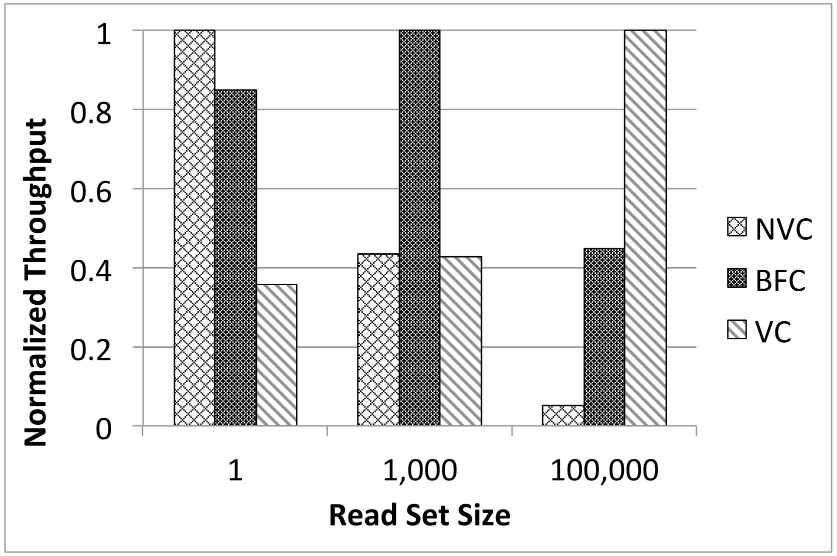


joint work with
D. Didona, S. Peluso and F. Quaglia

Two case studies

Certification Schemes

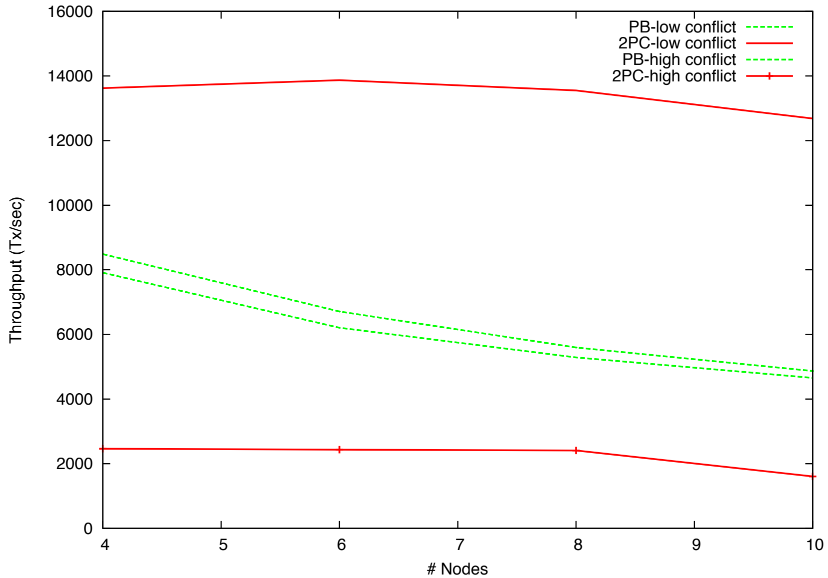
NVC vs VC vs BFC



joint work with
M. Couceiro, and L. Rodrigues

Single vs multi-master

2PC vs PB

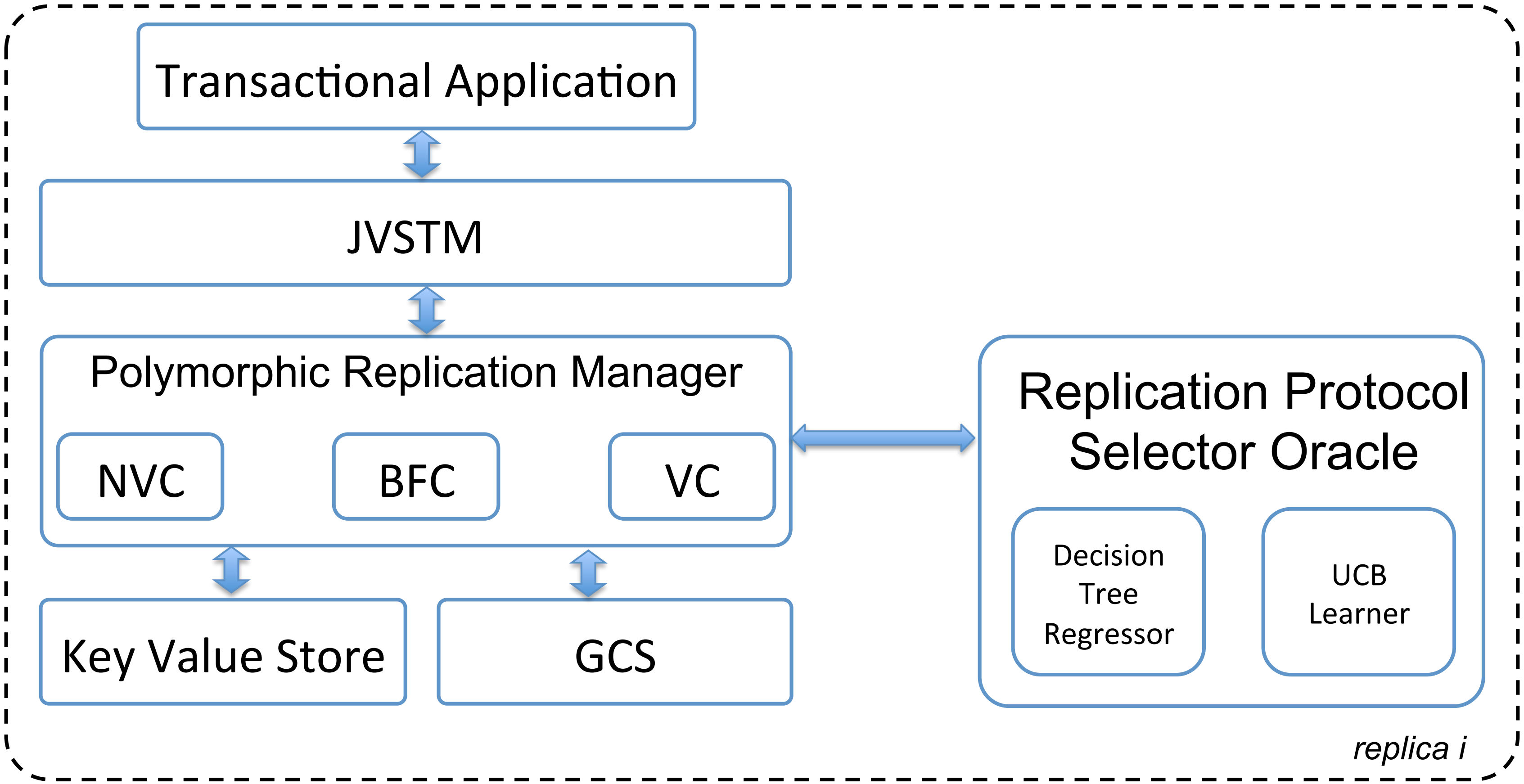


joint work with
D. Didona, S. Peluso and F. Quaglia

Certification Schemes NVC vs VC vs BFC

- The famous 2 keys issues:
  - Allow coexistence of multiple certification schemes via the Polymorhic 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)

PolyCert: System Architecture



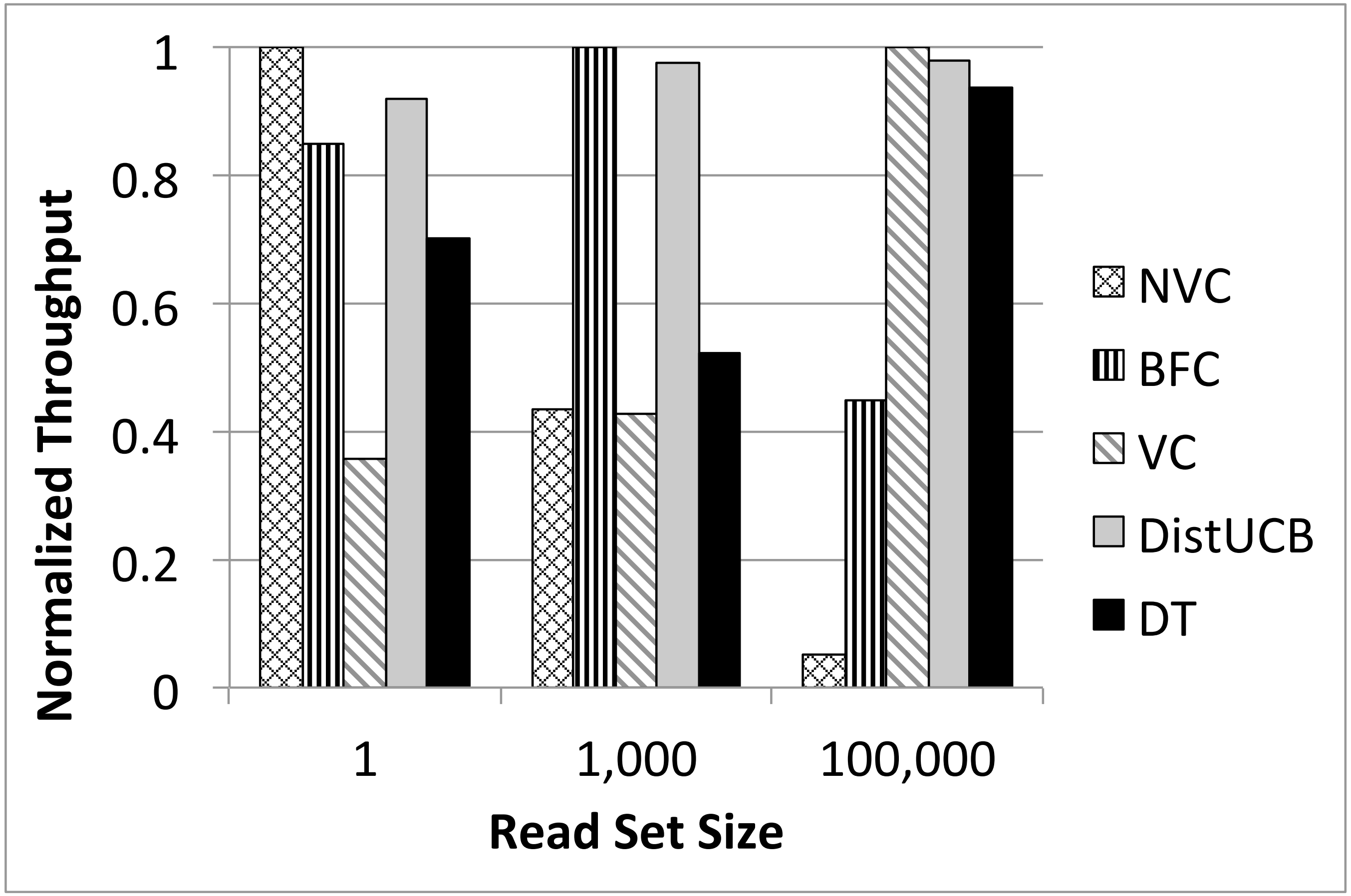
Off-line ML techniques

- 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

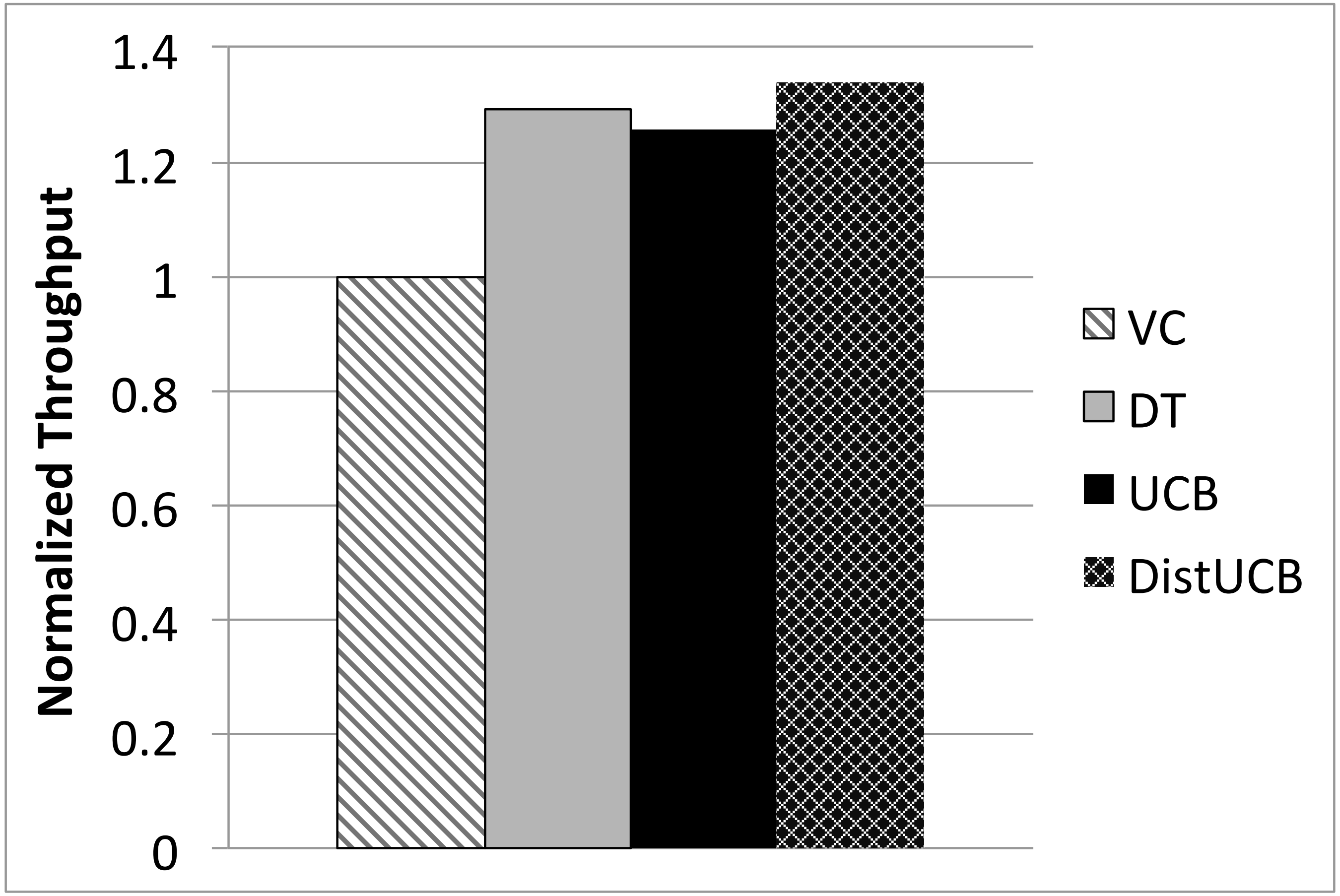
On-line reinforcement learning

- 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

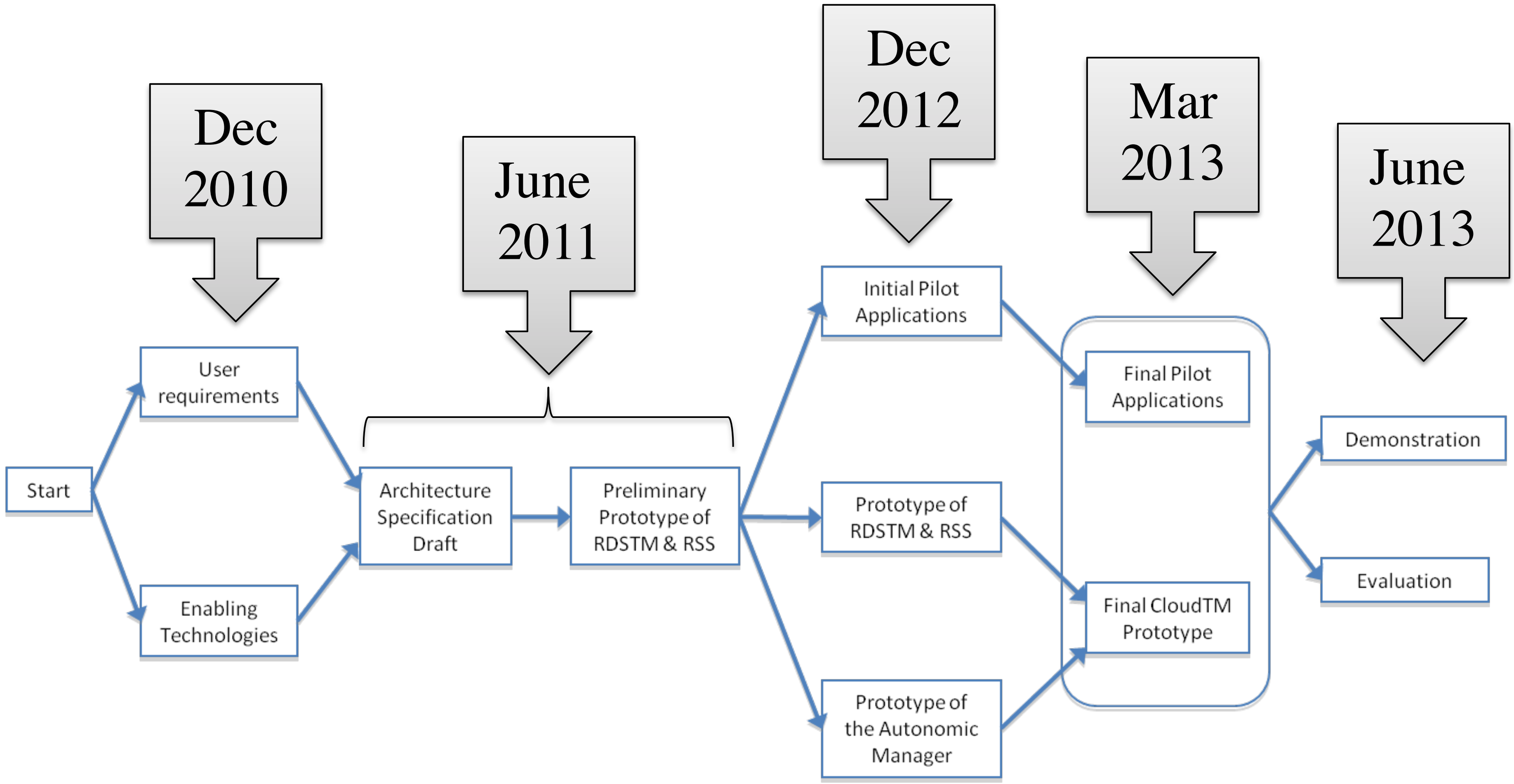
Chasing the optimum…



...and beating it!



Project's timeline



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*