On the Viability of Speculative Transactional Replication in Database Systems: a Case Study with PostgreSQL

Sebastiano Peluso, Roberto Palmieri, Francesco Quaglia, Binoy Ravindran

"Sapienza" University of Rome, Italy

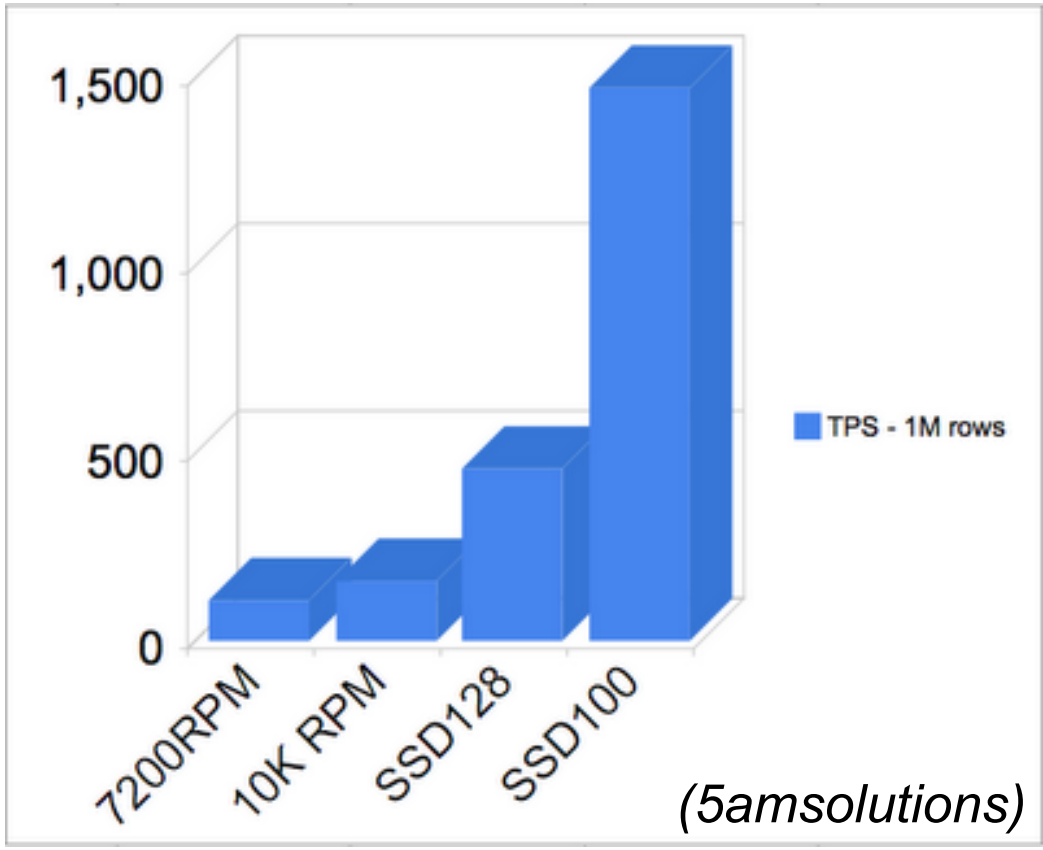
Virginia Tech. USA

The 12th IEEE International Symposium on Network Computing and Applications (IEEE NCA13)

Solid State Drives (SSDs)

A data storage device that uses solid-state memory to persist data

- PostgreSQL benchmark over one million rows
- SSD
  - 1472 TPS = 679,35 µsec/Tx
- HHD (10K RPM)
  - 23 TPS = 43,47 msec/Tx



Context

- Fault tolerance is a desirable property of transactional systems
- Replication is the typical mean
- Replication protocol suited for distributed transactional systems relying on:
  - Active Replication, as a paradigm for coordinating replicas;
  - Speculative processing, for boosting local processing.

Why Active Replication?

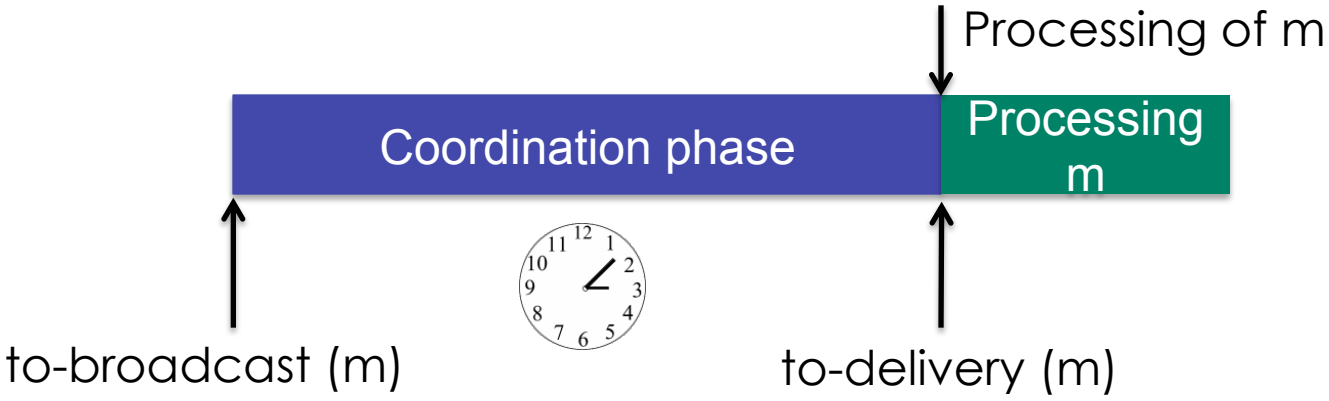
- Each replica keeps all shared data and executes the same transactions in the same order
- PRO (+)
  - Full failure masking
  - No coordination for read-only transactions
  - No communication during transaction processing
  - Prone to target performance issues
- CONS (-)
  - Agreement on common execution order
  - Deterministic business logic

Why Speculative Processing?

- A way to exploit as much as possible available resources (tailored for multicore architectures)
- A mean of anticipating work filling waiting periods

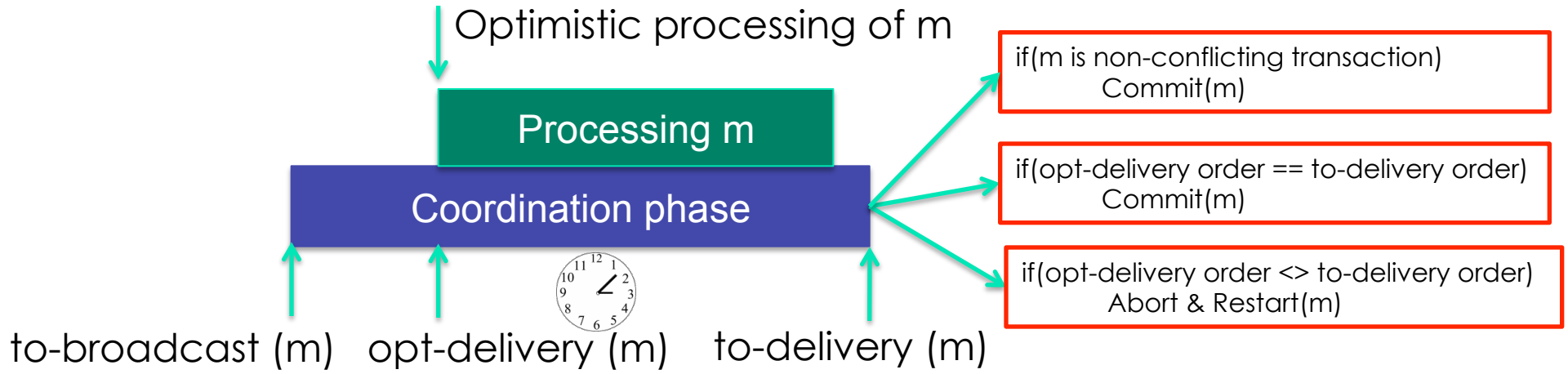
State Machine Approach (SM)

- Implements Active Replication paradigm
- Based on Atomic Broadcast as group communication system
- Does not exploit any kind of optimism



Optimistic Approach (OPT)

- Based on Optimistic Atomic Broadcast as a group communication system
- It processes in optimistic manner:
  - At most one conflicting transaction
  - Any non-conflicting transactions



Limited Overlapping

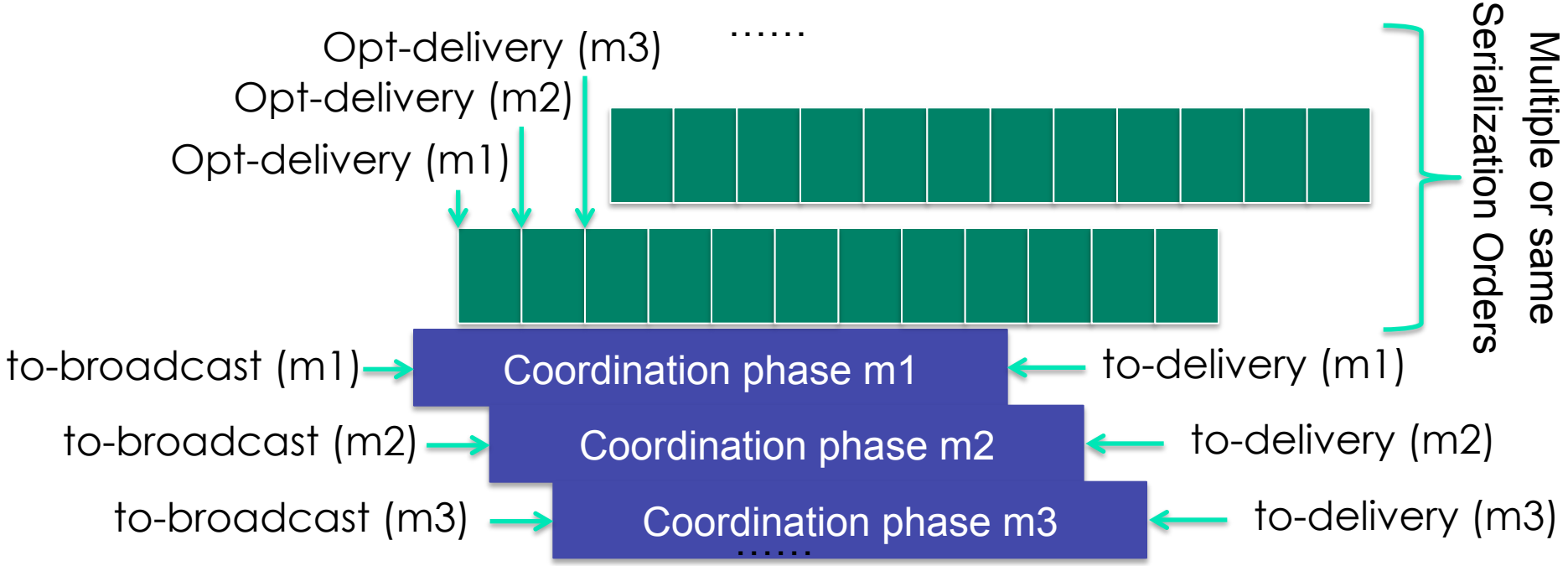
- In case of fine-grain transactions, the overlapping between the coordination phase and local processing is very limited

Mean Transaction Execution Time

Traditional Scenarios	Modern Scenarios
≈ 2m/10m sec	≈ 2m/500μ sec

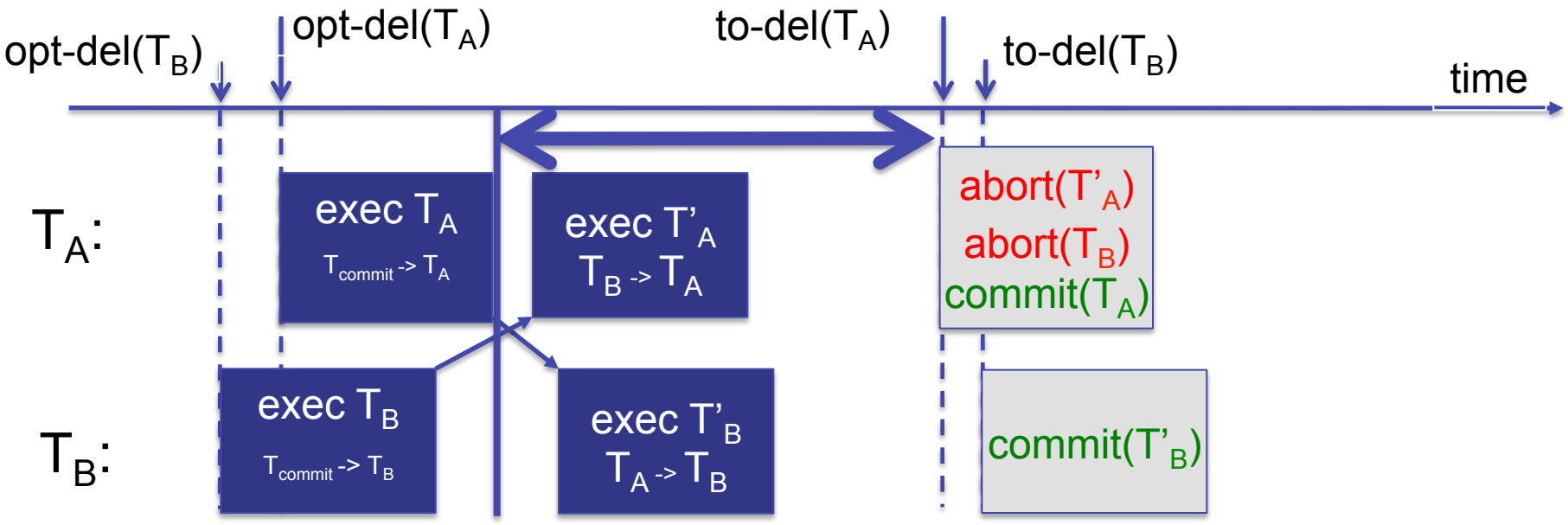
Target: Maximize the overlap

- Coordination delay Vs Local transaction processing



How: Speculative Processing

- Basic ideas:
  - Activate all transactions as soon as they are optimistic delivered
  - Explore (in depth and/or in breadth) multiple serialization orders



Contribution

- Integration of the active replication paradigm with speculation in existing PostgreSQL
  - Centralized
  - Non fault-tolerant
  - Open-source DBMS

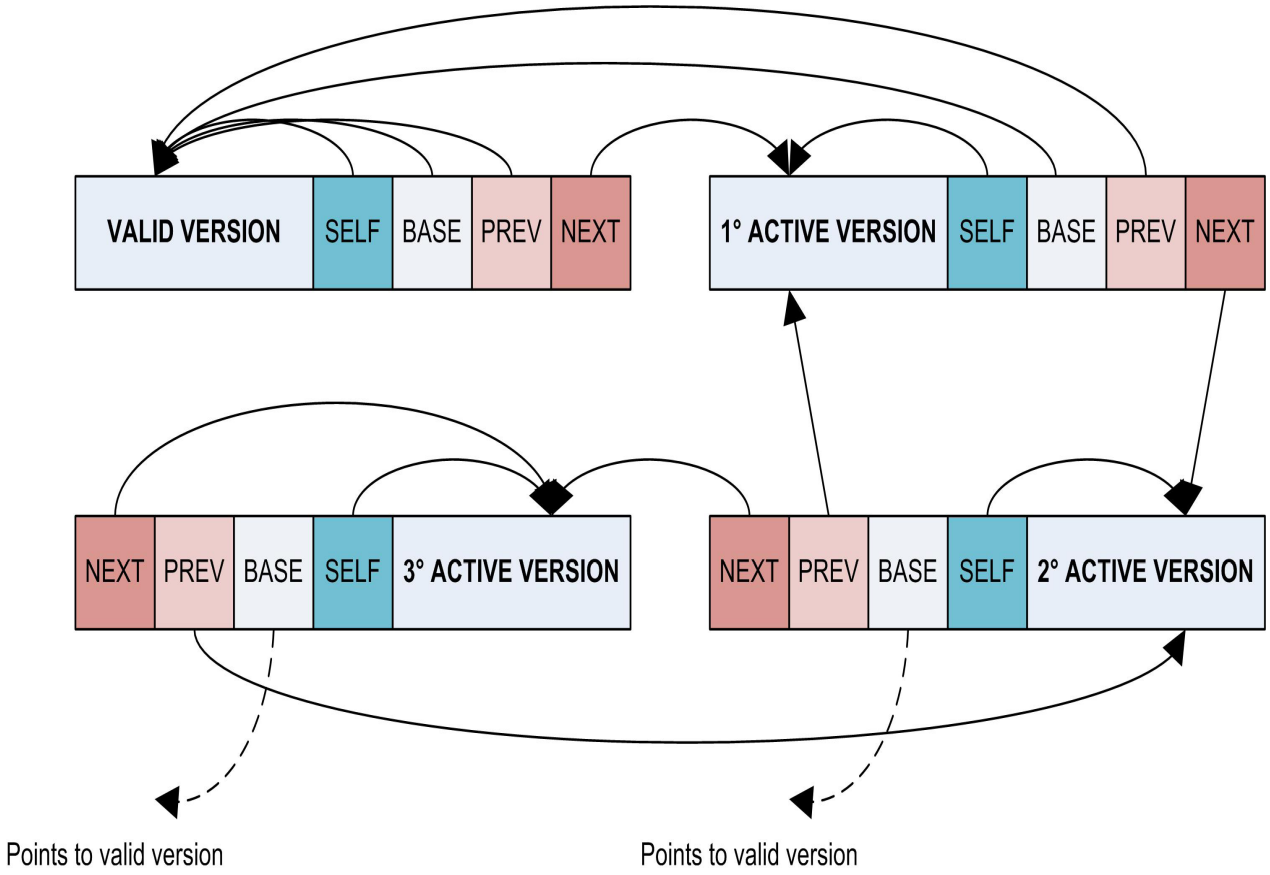
- Speculative supports embedded:
  - Transaction Demarcation and Commit
  - Enhanced Multiversioning
  - Speculative Transactions Forking

Transaction Demarcation and Commit

- Each transaction requested submitted by clients is a transaction *family*
- Speculative transactions belonging to the same *family* (activated on different snapshots) are *siblings*
- Transactions are identified with:
  - FAMILY ID: the family's identification
  - INSTANCE ID: the unique id valid in the context a family
- Hash-Map stores families' information and their speculative transactions to always have a picture of the current execution status.

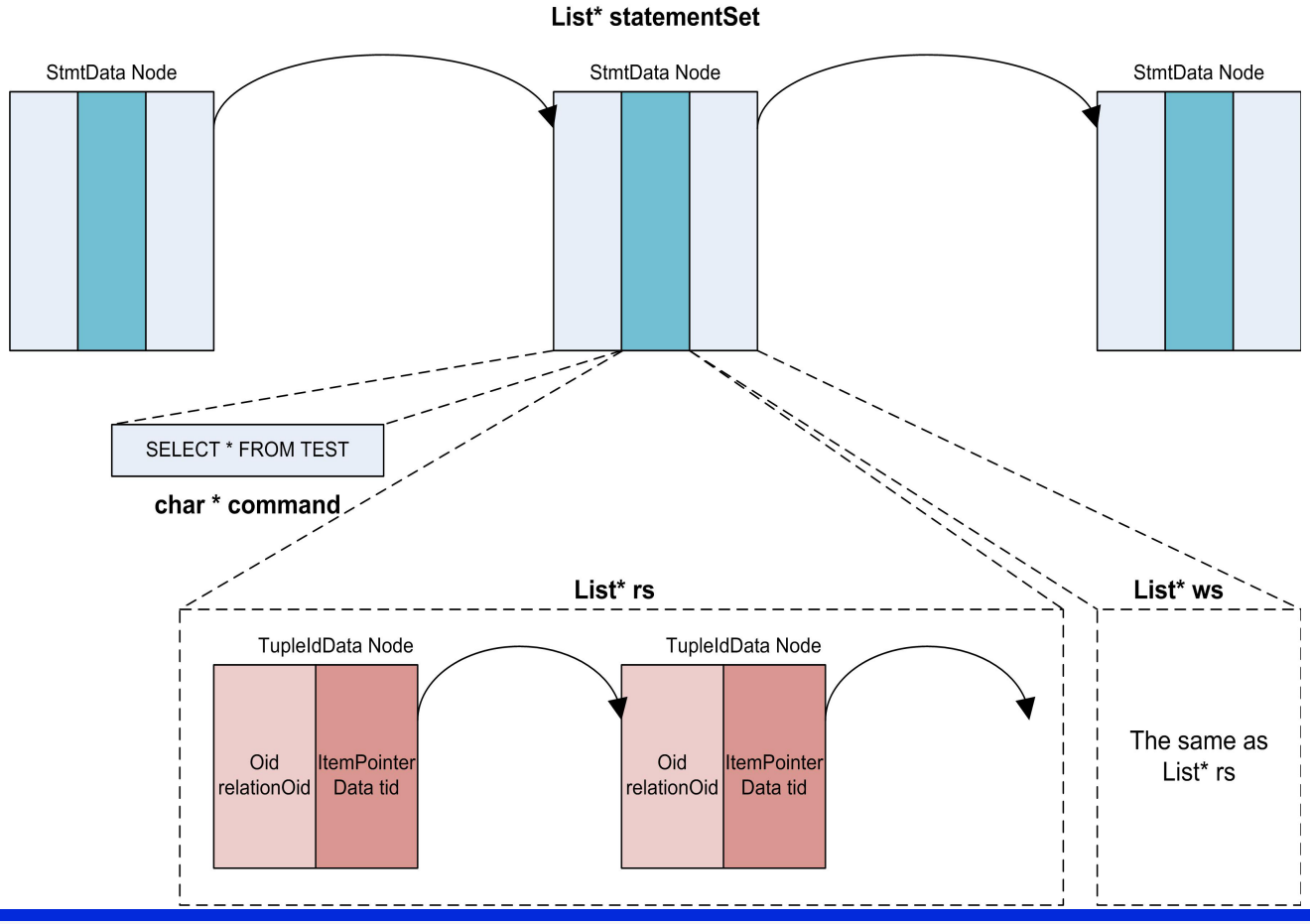
Enhanced Multiversioning

- Speculation requires non- blocking tuple access
- Post-images of inserted/updated tuples must be visible before the writing transaction is committed



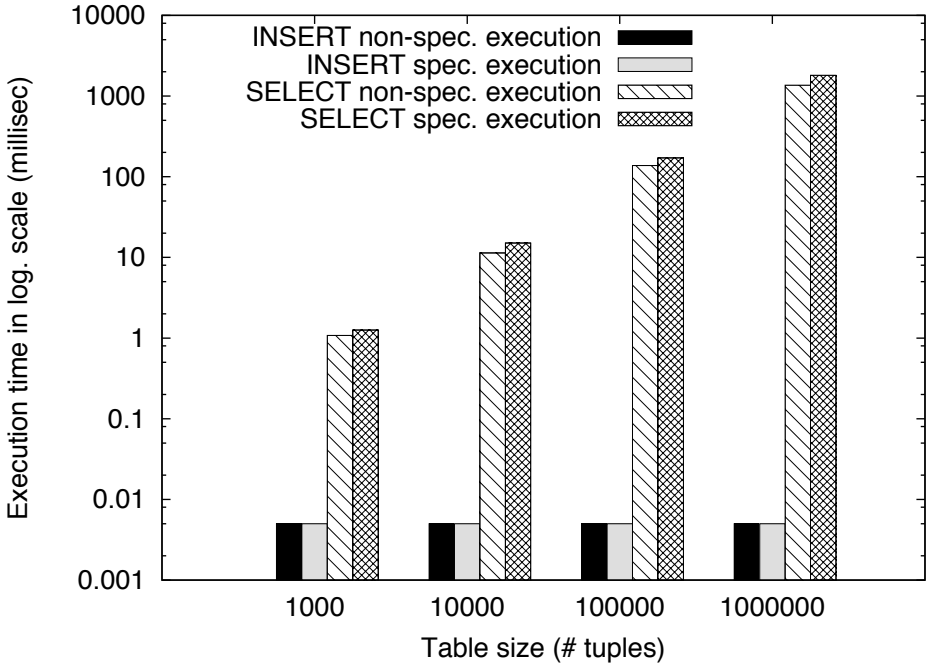
Speculative Transactions Forking

- Read operations return more than one version per tuple
- A forking mechanism is needed for allowing transactions to explore different serialization order

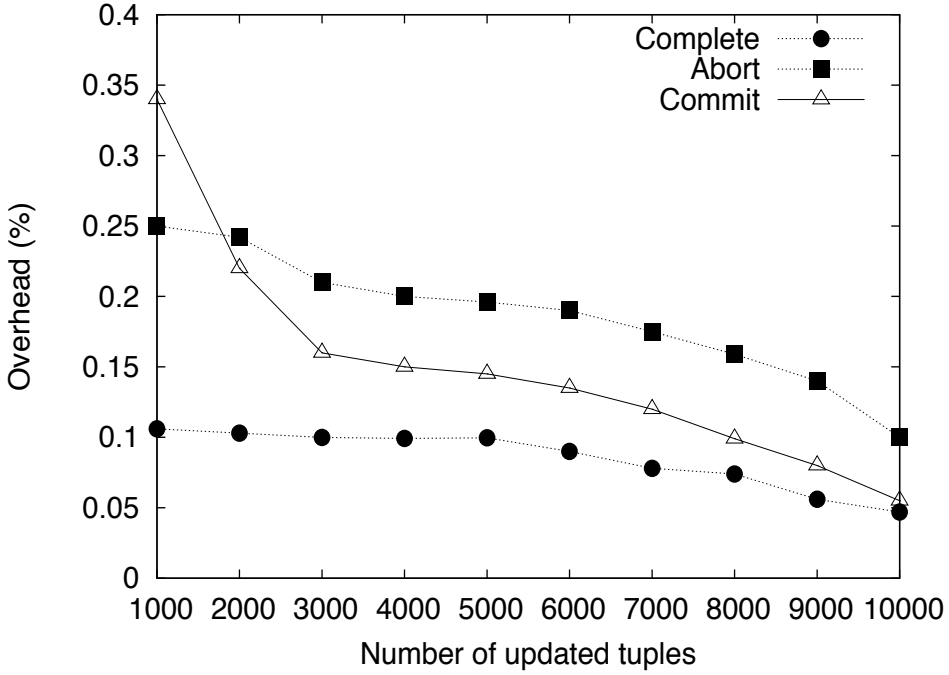


Evaluation

- Synthetic benchmarks and TPC-C
- 32 core machine HP ProLiant server, equipped with four 2GHz AMD Opteron 6128 and 64GB RAM



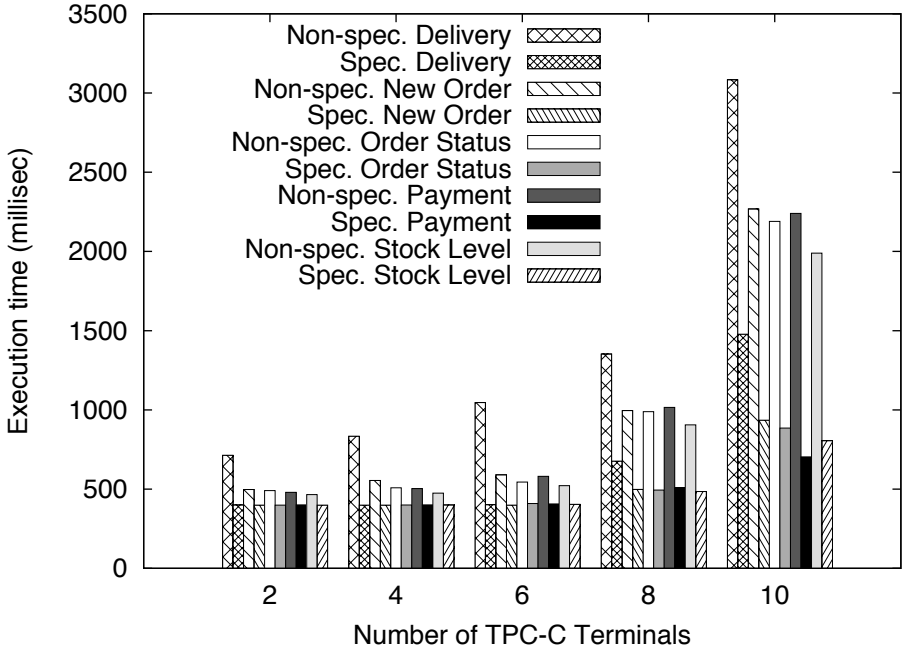
Overhead of synthetic bench SELECT commands



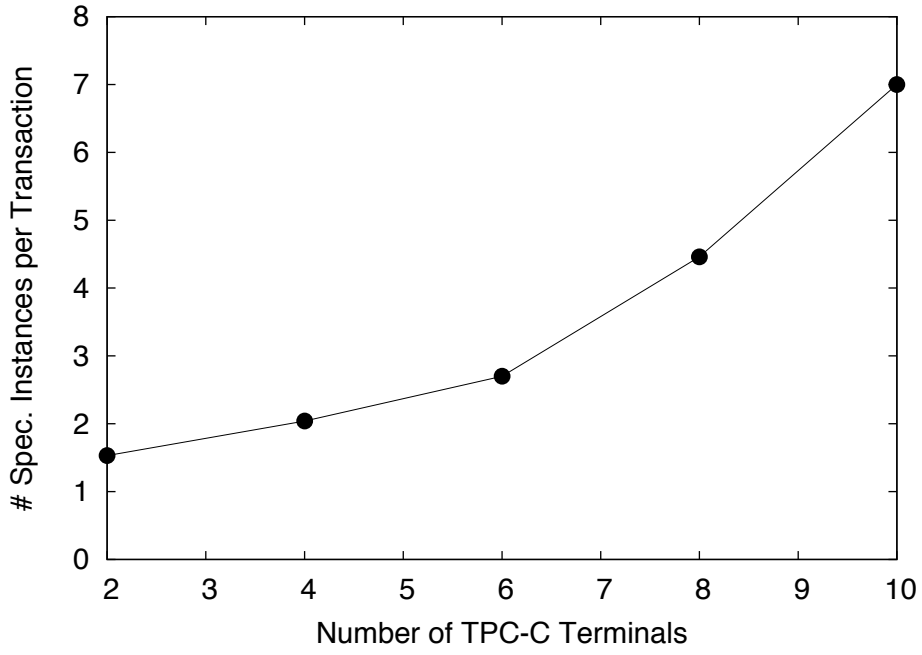
Overhead of synthetic bench UPDATE commands

Evaluation

- Performance of TPC-C



Performance comparison
Spec Vs Non Spec execution



Actual number of speculative transactions generated

Lesson, Conclusion & Questions

- Speculative processing in DBMS is doable
- Overhead is limited especially having multicore architecture
- The number of speculative transactions generated depends on the concurrency control rules