H-Store : The End of an Architectural Era

Stonebraker et al., VLDB 2007

Joy Arulraj

CMU 15-799 : Paper Presentation

Carnegie Mellon

Talk Gist

- “One size fits all” databases excel at nothing
  - Specialized databases and languages

Motivation

- System R (1974)
  - Seminal database design from IBM
  - First implementation of SQL

- Hardware has changed a lot over 3 decades
  - Databases still based on System R's design
  - Includes DB2, SQL server, etc.

Hardware Evolution

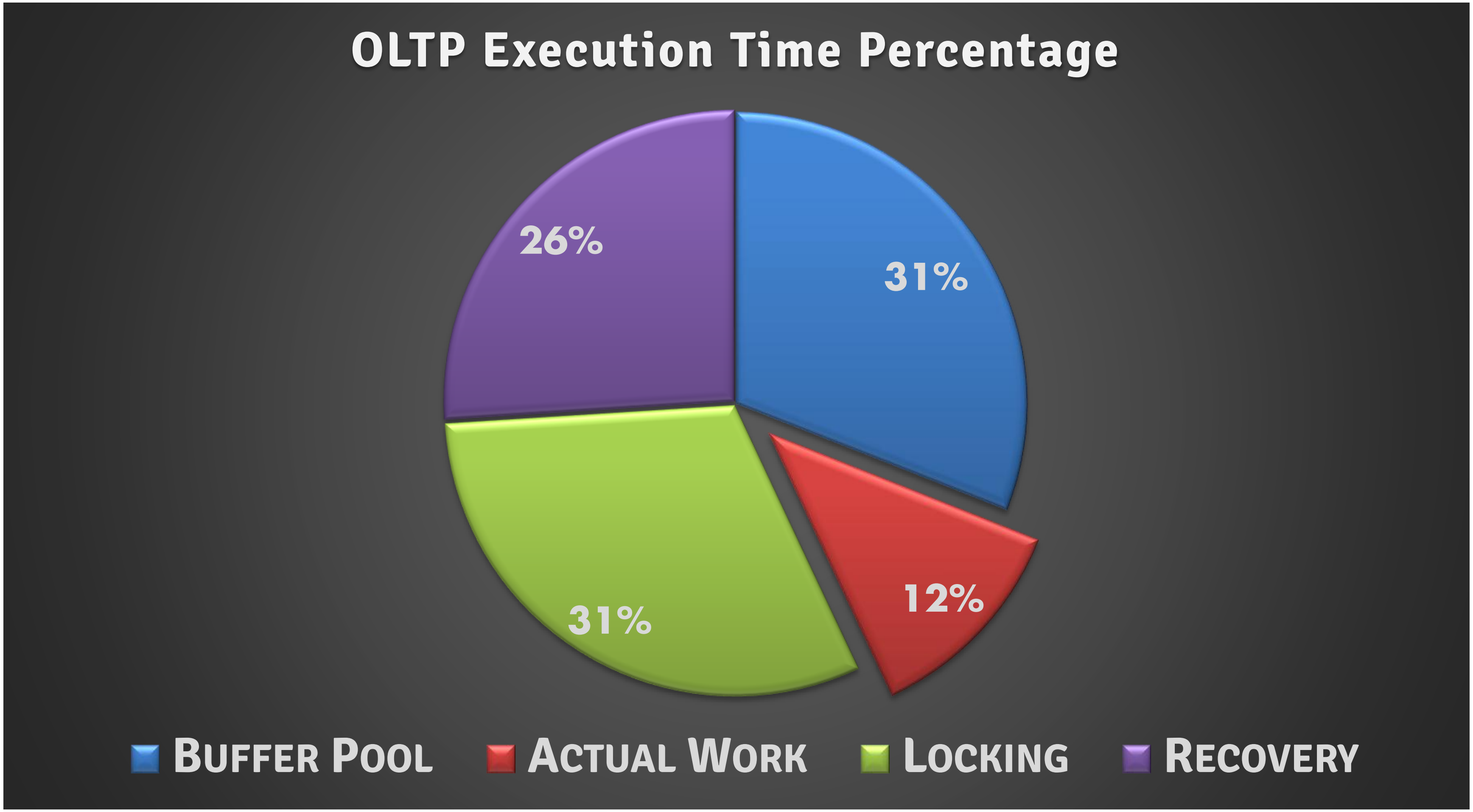
- Memory and disk capacity
  - 1000X larger
- Processors
  - 1000X faster
- But,
  - Disk bandwidth has grown very slowly
  - Disk latency for random accesses still high

Problem Statement

- Traditional database design
  - Disk oriented storage and indices
  - Multithreading to hide latency
  - Concurrency control using locks
  - Log based recovery

- Is traditional DB design still relevant ?

OLTP Bottlenecks



© OLTP Through the Looking Glass, and What We Found There, SIGMOD '08

Disk oriented storage

- Assumption
  - Main memory can't hold the database

- Main memory capacity has increased
  - A lot .. commodity devices can hold 32 GB
  - OLTP workloads <1TB => 32 node cluster

Multithreading

- Assumption
  - Disk accesses are slow => Must hide latency
  - Multiple threads => Need concurrency control

- Disk accesses are still slow
  - But, what if we store the database in memory ?
  - Single threaded model => No need for isolation

Concurrency control

- Assumption
  - Transactions used to be long (user input, disks)
  - Isolation obtained using locks
  - Pessimistic approach – blocks at txn. start

- Transactions now much shorter
  - Main memory latency, stored procedures

Log based recovery

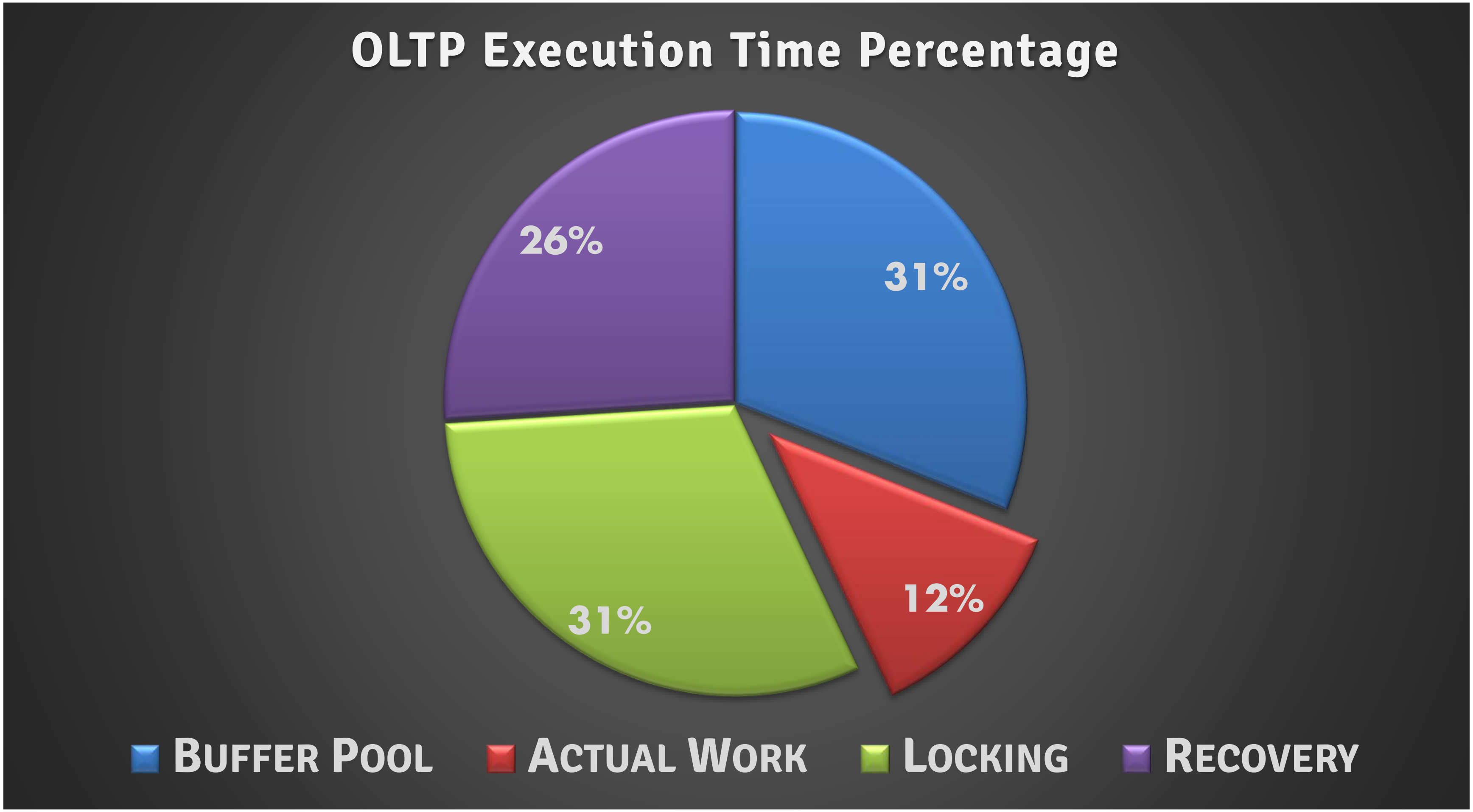
- Assumption
  - Logs needed for faster recovery (few machines)
  - Redo log brings state till crash point
  - Undo log then removes effect of failed txns.
- Machines cheaper, and availability crucial
  - Hot standby or peer-to-peer model
  - Simplify logging – remote replica for recovery

What just happened ?

- Assumptions are from a bygone era
  - Need a clean design from scratch

- Design a specialized database
  - Each world has its own constraints
  - This paper targets OLTP world

OLTP Bottlenecks



© OLTP Through the Looking Glass, and What We Found There, SIGMOD '08

New Design

- Buffering overhead
  - Main memory holds database
- Locking overhead
  - Single-threaded execution engine
- Latching overhead
  - No shared data structures

New Design

- Logging overhead
  - Replication for recovery => No redo log
  - Transient undo log sufficient for rollback

- Transaction classes
  - Optimize concurrency control protocol

New Design

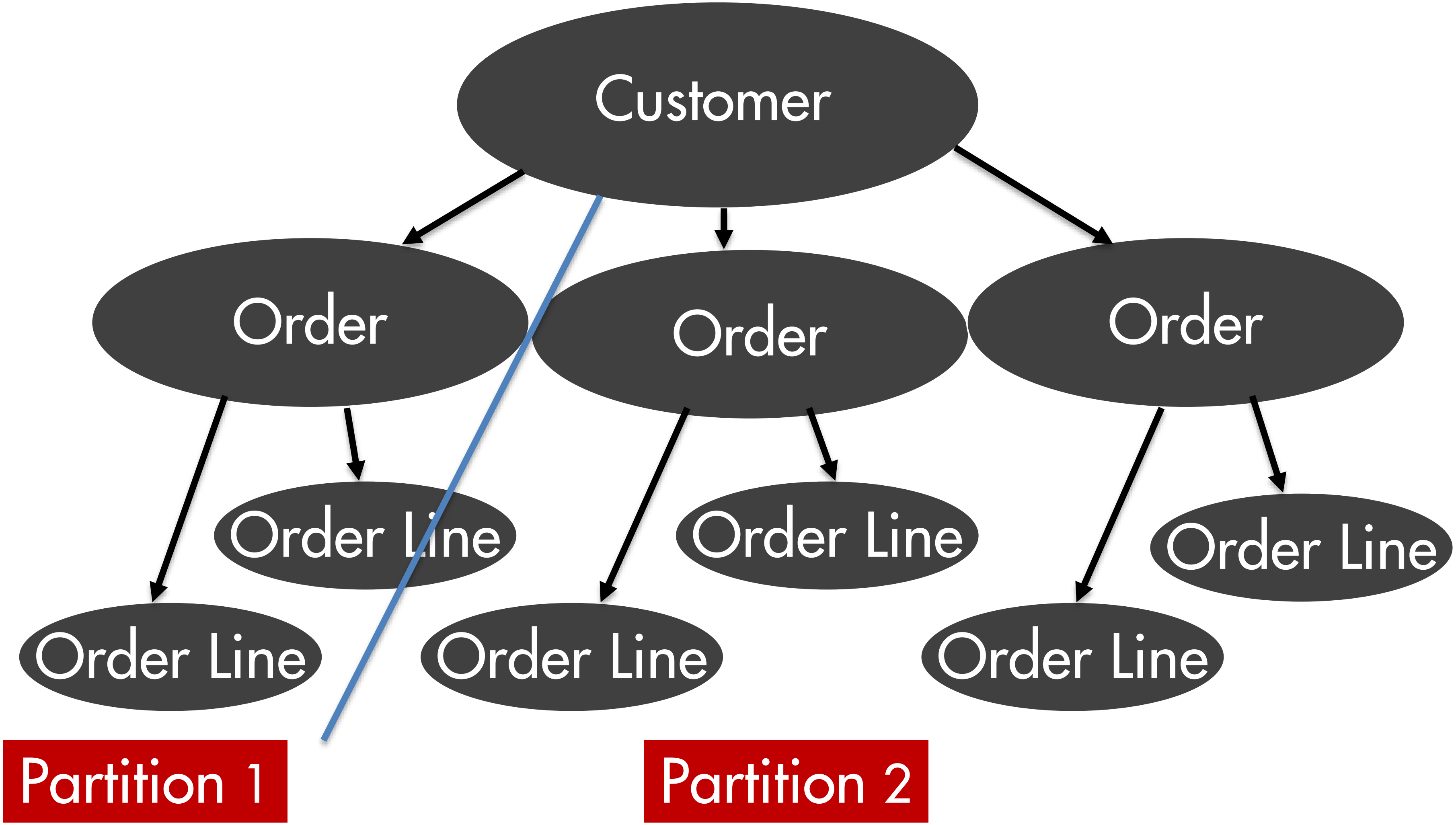
- Incremental scalability
  - Shared nothing architecture

- Remove knobs/tuning parameters
  - Personnel costs higher than machine costs
  - Automatic physical database designer

Transaction Classes

- Example
  - Class : “Insert record in History where customer = $(customer-Id) ; more SQL statements ;”
  - Runtime instance supplies $(customer-Id), etc.
- Each transaction class has certain properties
  - Optimize concurrency control protocols
  - And commit protocols

Constrained Tree Schema



Single-sited transactions

- All queries hit same partition

- Constrained Tree Schemas
  - Root table can be horizontally hash-partitioned
  - Collocate corresponding shards of child tables
  - No communication between partitions

One-shot transactions

- No inter-query dependencies
- Execute in parallel without communication
  - Replicate read only parts
  - Vertical partitioning
  - Can decompose into single-sited plans
  - Local decisions => No redo log required

Two-phase and sterile classes

- Two-phase classes
  - Phase 1 : Read-only operations
  - Phase 2 : Updates can't violate integrity
  - No undo log required
- Sterile classes
  - Commute with other classes
  - No concurrency control needed

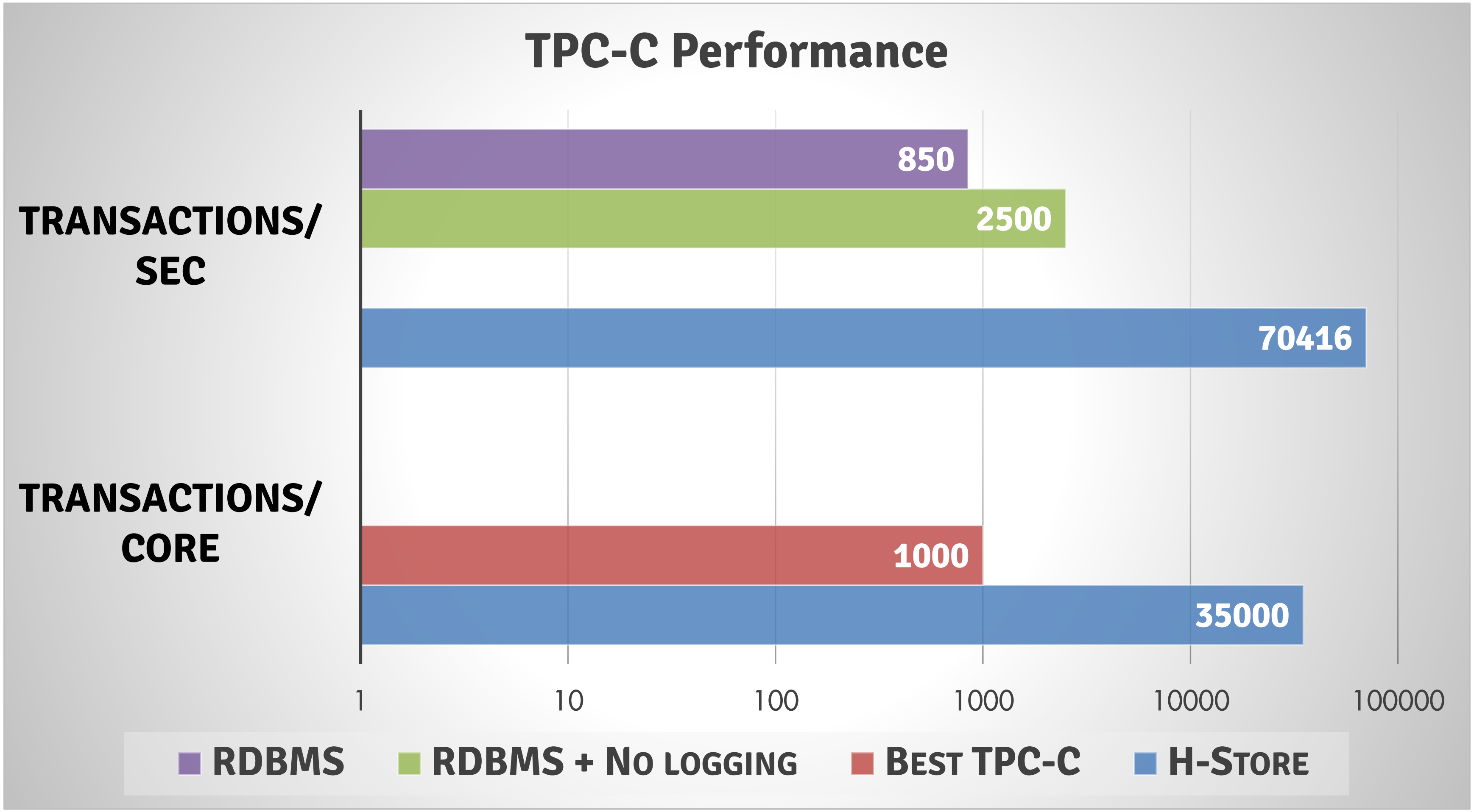
General transactions

- Basic Strategy
  - Timestamp ordering
  - Wait for “small period of time”
  - Preserve timestamp order (network delay)
- Advanced Strategy
  - Increase wait latency if too many aborts
  - Track read and write sets

Results

- H-Store
  - – Targets OLTP workload
  - – Shared-nothing main memory database
- TPC-C benchmark
  - – All classes made two-phase => No coordination
  - – Replication + Vertical partitioning => One-shot
  - – All classes still sterile in this schema => No waits

Results


TPC-C Performance
TRANSACTIONS/ SEC
TRANSACTIONS/ CORE
850
2500
70416
1000
35000
1
10
100
1000
10000
100000
RDBMS
RDBMS + No logging
Best TPC-C
H-Store

Conclusions

- "One size does not fit all"
  - OLTP : relational model
  - OLAP : entity-relational model
  - Stream processing : hierarchical model
  - Scientific : arrays

- SQL is not the answer
  - No one size fits all language (PL world)
  - Need more specialized little languages

Talk Summary

- "One size fits all" databases excel at nothing
  - – Specialized databases and languages

- H-Store
  - – Clean design for OLTP domain from scratch
  - – Emerging hardware support – NVM, TM ?

Thanks !