Studying state-of-the-art HTAP systems

- Satya Sai Bharath Vemula
- Rwitam Bandyopadhyay
- Shubham Pandey

PURDUE UNIVERSITY

Milestone 1

Table of Contents

PURDUE UNIVERSITY

Introduction

HTAP

OLTP

Online Transaction Processing

OLAP

Online Analytical Processing

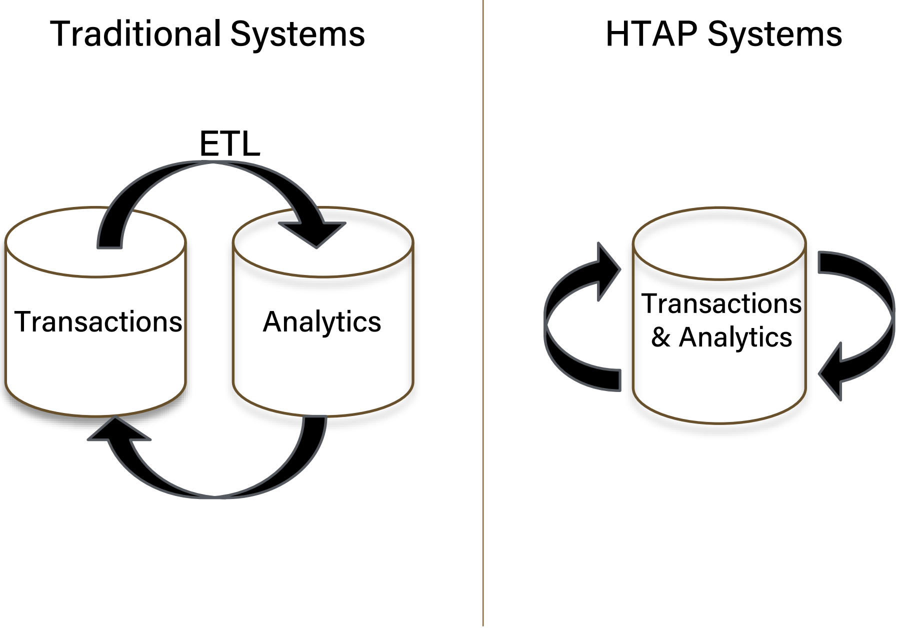
ETL

Source: https://phoenixnap.com/kb/oltp-vs-olap

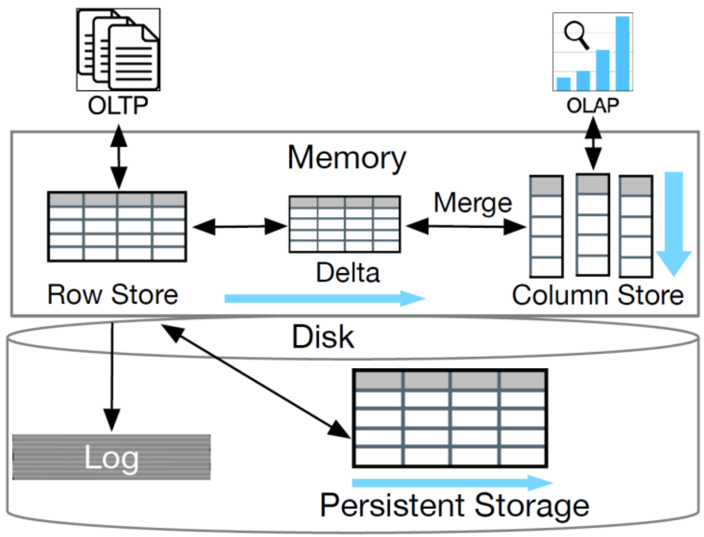
Challenges of HTAP Systems

What is it not easy to imagine both systems in one?

- Data modeling complexity
- Performance optimization
- Consistency and correctness
- Scalability
- Cost



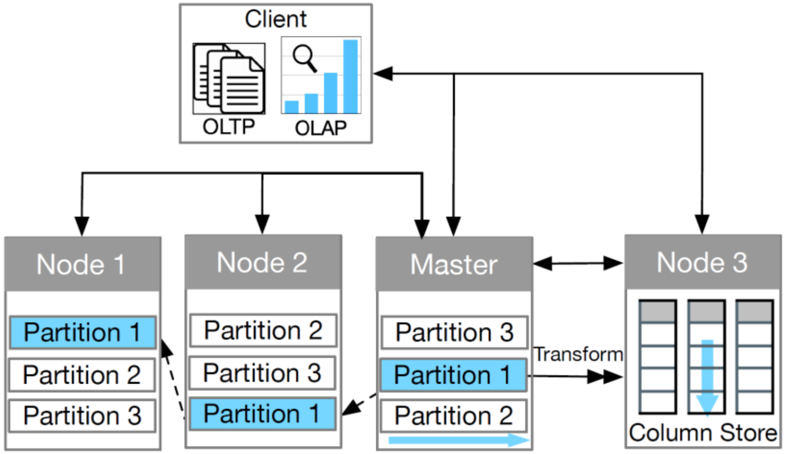
Storage Architectures of HTAP Databases



Primary Row Store + In-Memory Column Store

Source: Guoliang Li and Chao Zhang. 2022. HTAP Databases: What is New and What is Next. (SIGMOD '22)

Storage Architectures of HTAP Databases

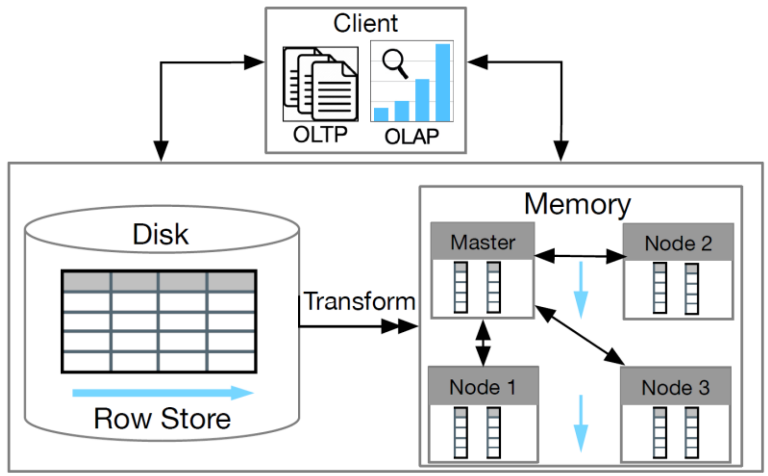


Distributed Row Store + Column Store Replica

PURDUE UNIVERSITY

Source: Guoliang Li and Chao Zhang. 2022. HTAP Databases: What is New and What is Next. (SIGMOD '22)

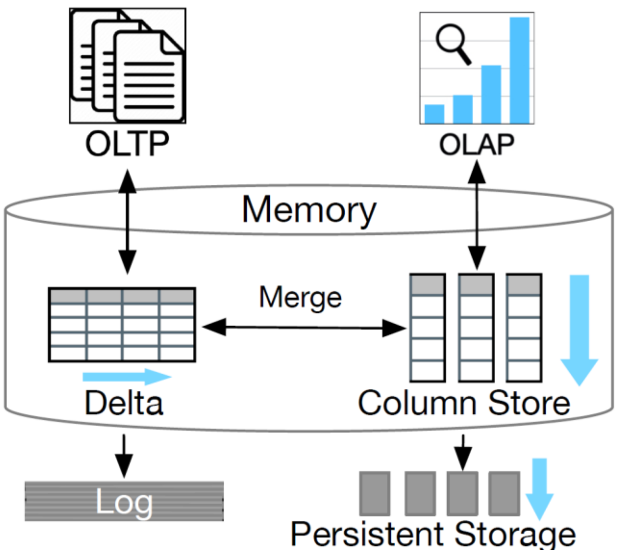
Storage Architectures of HTAP Databases



Disk Row Store + Distributed Column Store

Source: Guoliang Li and Chao Zhang. 2022. HTAP Databases: What is New and What is Next. (SIGMOD '22)

Storage Architectures of HTAP Databases



Primary Column Store + Delta Row Store

Source: Guoliang Li and Chao Zhang. 2022. HTAP Databases: What is New and What is Next. (SIGMOD '22)

Oracle Dual Format

Disagg1

Presented by Rwitam Bandyopadhyay

PURDUE UNIVERSITY

Motivation

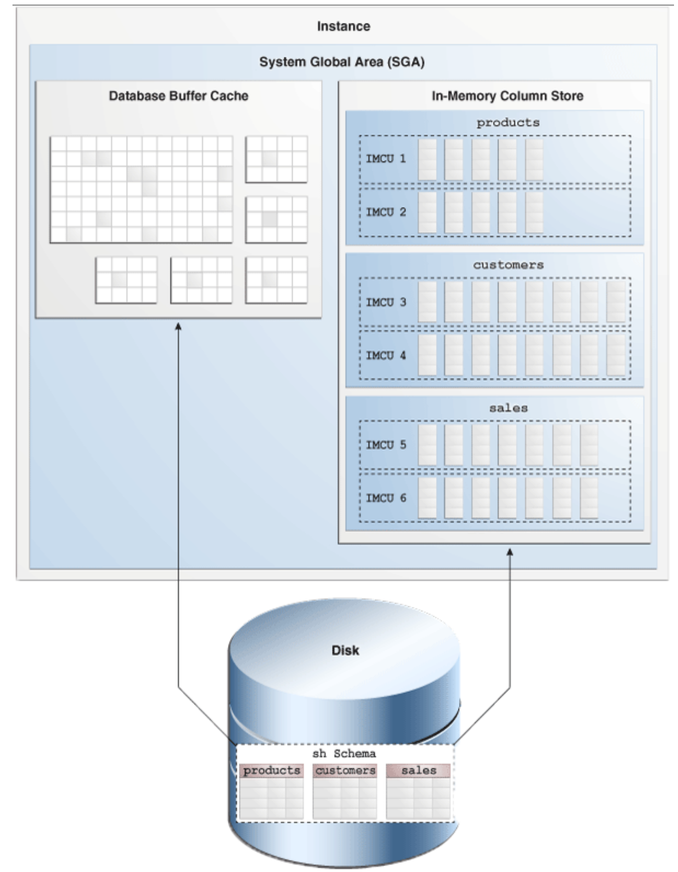
Oracle introduced the Database In-Memory option in 2014 as the industry's first dual-format, in-memory RDBMS

- No single data format is ideal for all type of workloads
- "Primary Row Store + In-Memory Column Store" architectural style
- The new columnar format is a pure in-memory format with no impact to the disk representation
- Tables required for fast analytics can be populated into the In-Memory column store
- Can be plugged in to any existing system with zero changes

Dual-Format, Dual Memory?

Dual format representation doesn't double memory requirements!

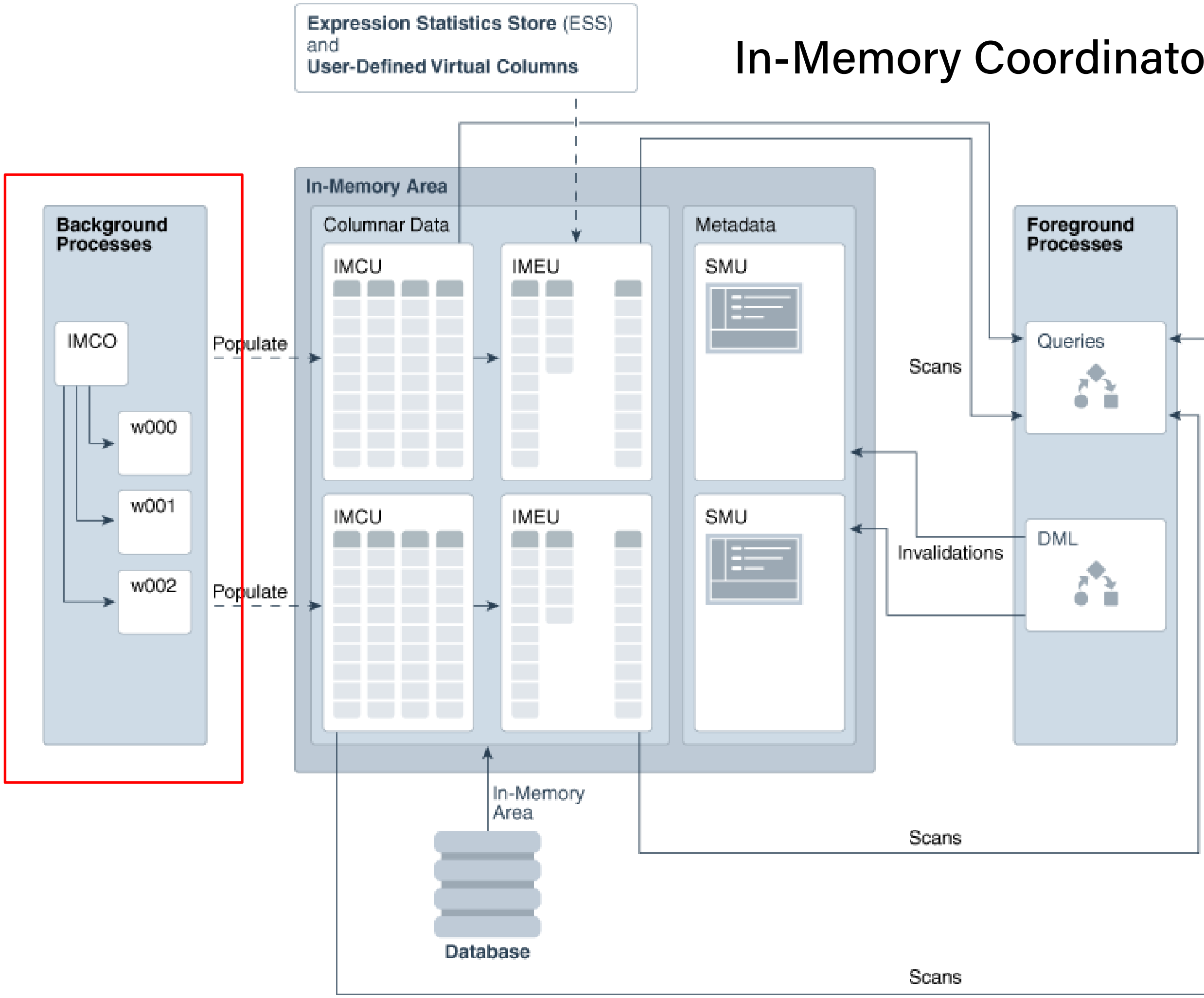
- Tables required for fast analytics
  → In-Memory Store
- OLTP updates/ highly selective lookups
  → Buffer Cache !
- Analytic Indexes can be dropped from the DB
- The IM column store is built into the DB data access layer



Source: Oracle Database R21 In-Memory Column Store Architecture

In-Memory Columnar Format (Building Blocks)

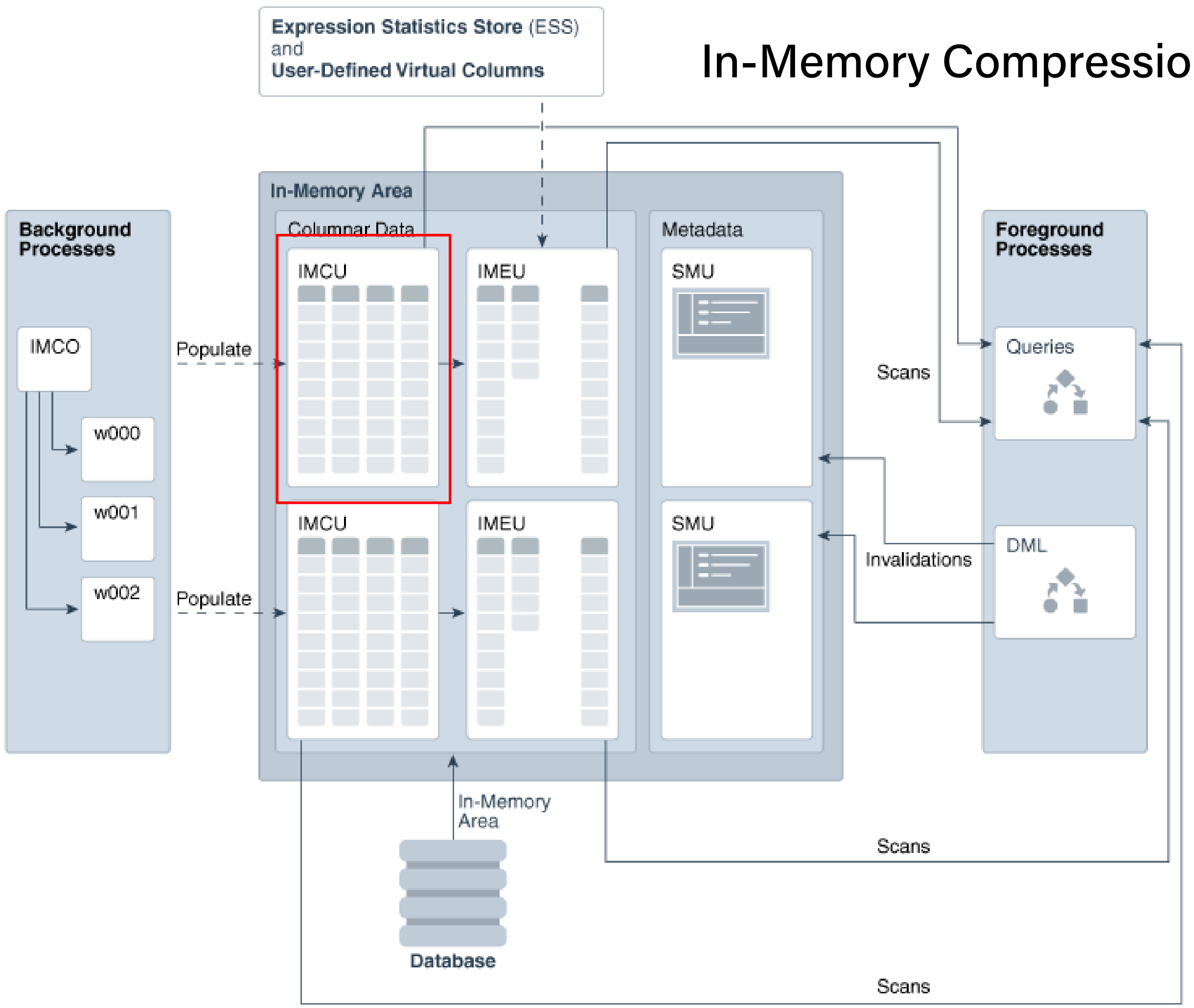
In-Memory Coordinator Process (**IMCO**)



Source: Oracle Database R21 In-Memory Column Store Architecture

In-Memory Columnar Format (Building Blocks)

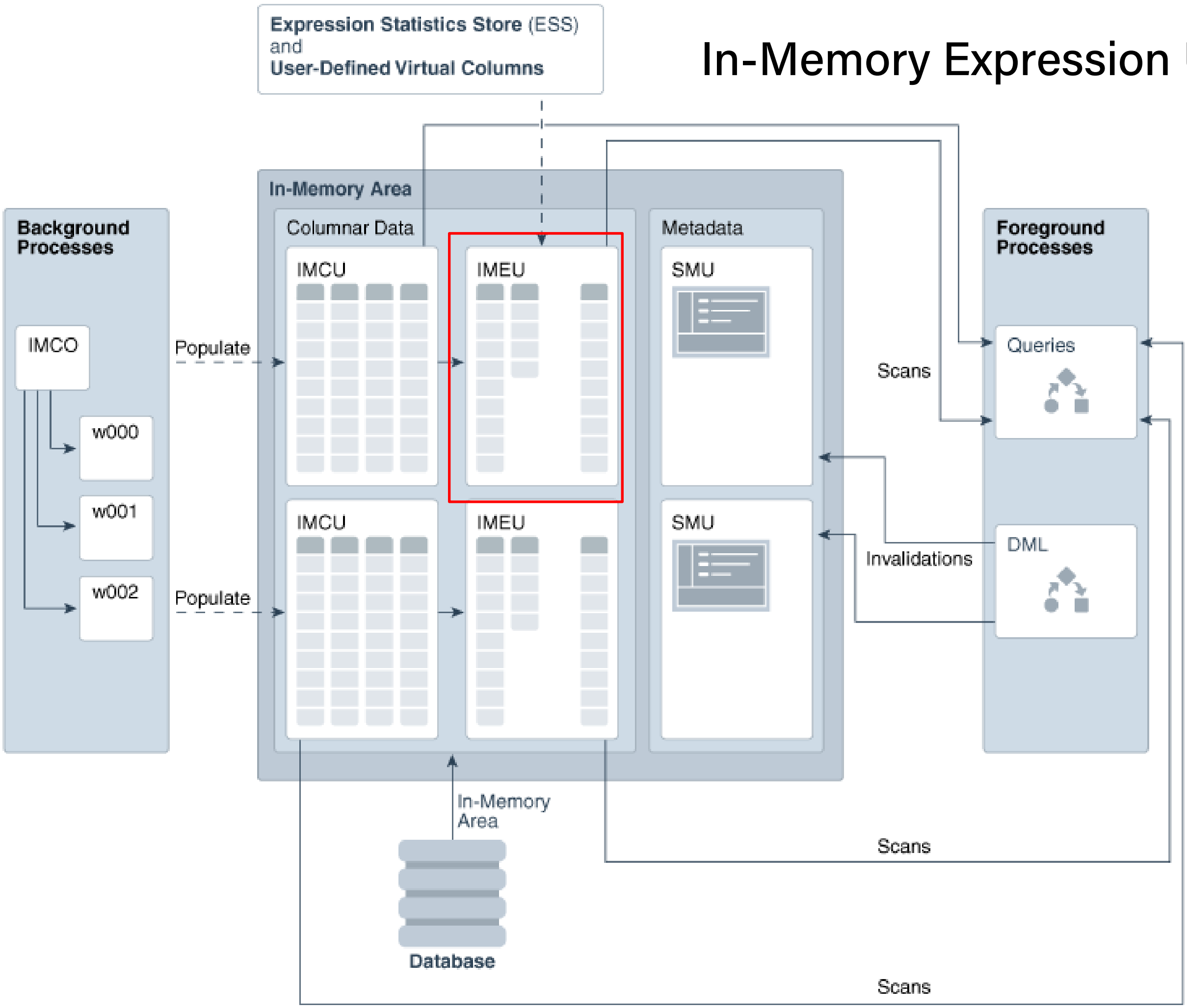
In-Memory Compression Units (**IMCUs**)



Source: Oracle Database R21 In-Memory Column Store Architecture

In-Memory Columnar Format (Building Blocks)

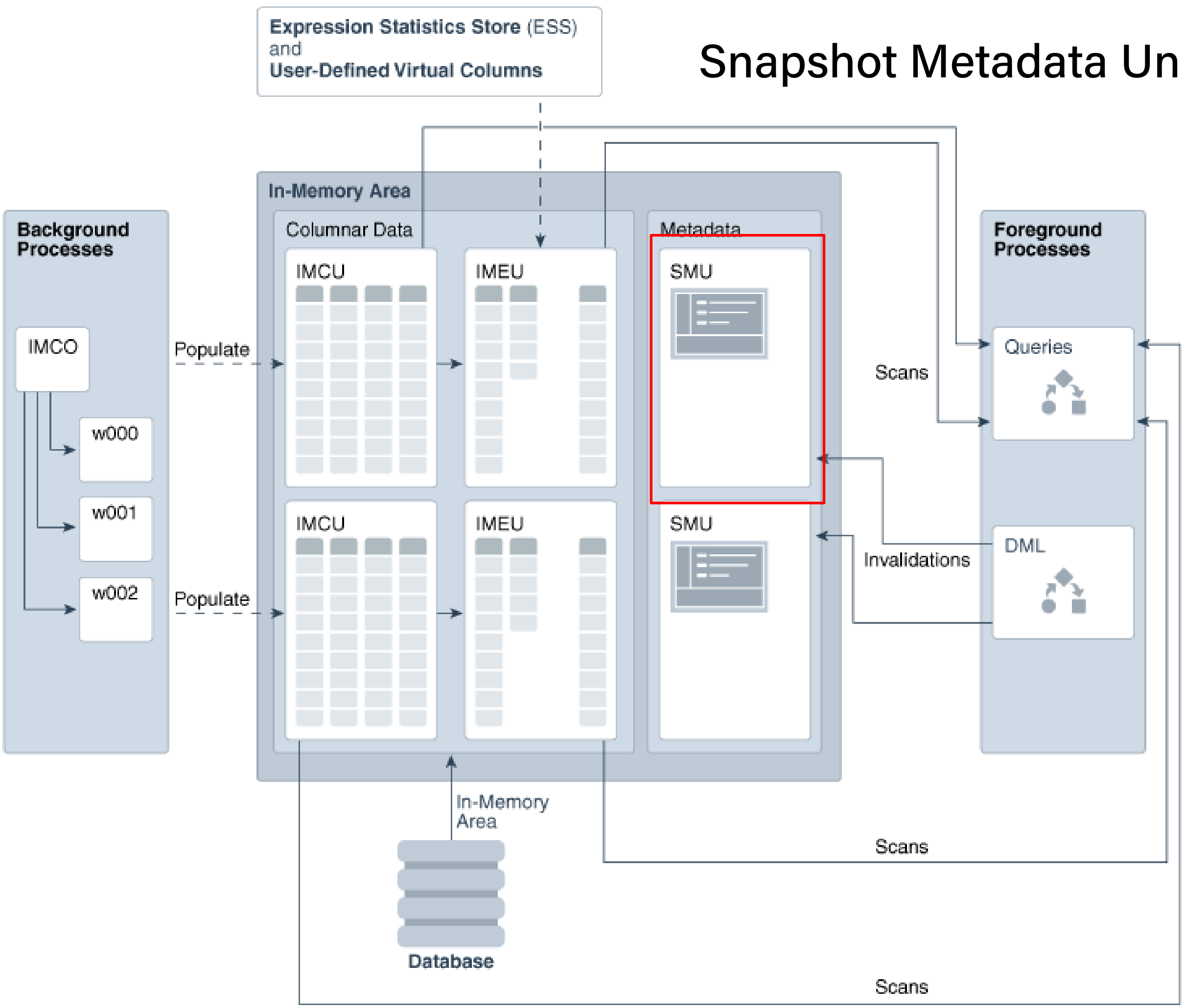
In-Memory Expression Units (**IMEUs**)



Source: Oracle Database R21 In-Memory Column Store Architecture

In-Memory Columnar Format (Building Blocks)

Snapshot Metadata Units (**SMUs**)



Source: [Oracle Database R21 In-Memory Column Store Architecture]()

In-Memory Columnar Format (Data Population)

It is possible to selectively populate the IM column store with a chosen subset of the database

- IM column store is populated using a pool of background server processes
- No application downtime during table population since it continues to be accessible via the buffer cache.
- Most other in-memory databases require a wait time for all objects to be brought in-memory

Priority	Description
CRITICAL	Object is populated **immediately** after the database is opened.
HIGH/ MEDIUM/ LOW	Objects are populated after CRITICAL in order of priority.
NONE	Object is only populated after it is **scanned for the first time** (Default).

Source: T. Lahiri, et al. Oracle Database In-Memory: A Dual Format In-Memory Database. In 2015 IEEE 31st International Conference on Data Engineering

In-Memory Columnar Format (Data Compression)

It is possible to selectively populate the IM column store with a chosen subset of the database

- In-memory storage is populated in contiguously allocated units called **In-Memory Compression Units.**
- It is possible to use different compression levels for different partitions within the same table. For example, a SALES table might have →
  - Current week partition - DML Compression
  - Earlier year partitions - Query Compression
  - Decade old partitions - Capacity Compression

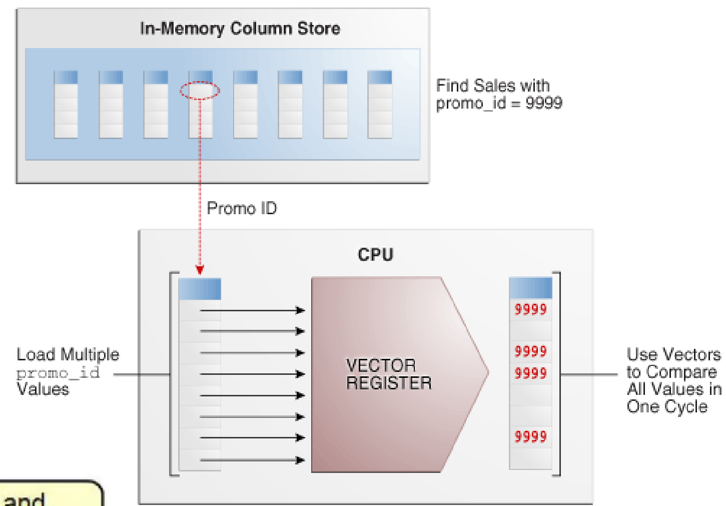
MEMCOMPRESS	Description
DML	Minimal compression optimized for DML Performance
QUERY LOW/HIGH	Optimized for fastest query performance (Default)
CAPACITY LOW/HIGH	Optimized for space saving

PURDUE UNIVERSITY®

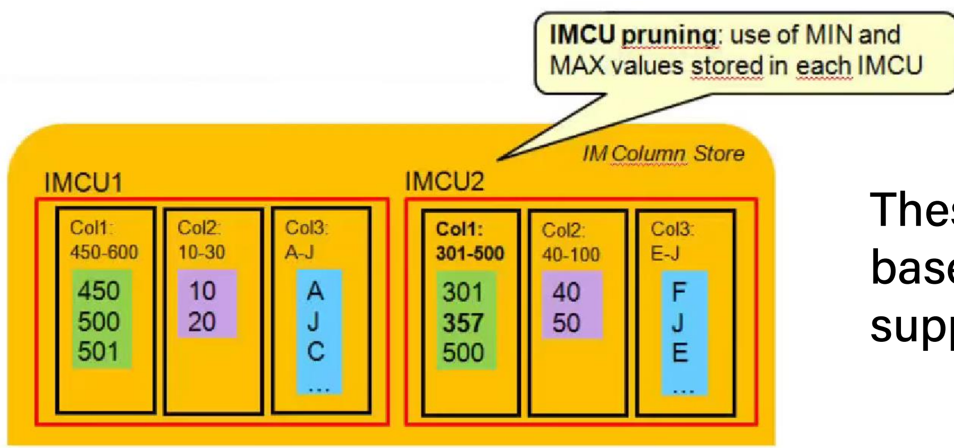
Source: T. Lahiri, et al. Oracle Database In-Memory: A Dual Format In-Memory Database. In 2015 IEEE 31st International Conference on Data Engineering

Query Processing

- In-Memory Scans utilize **SIMD** Vector Processing



- **In-Memory Storage Indexes** are automatically created and maintained on each of the columns.

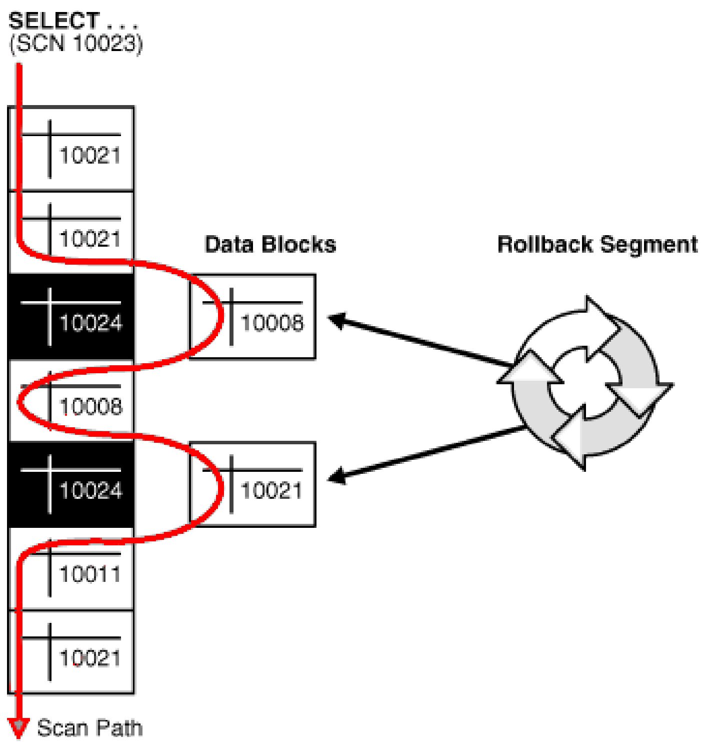


These allow data pruning to occur based on the filter predicates supplied in a SQL statement.

Source: [Oracle Database R21 In-Memory Column Store Architecture]

Transactional Consistency

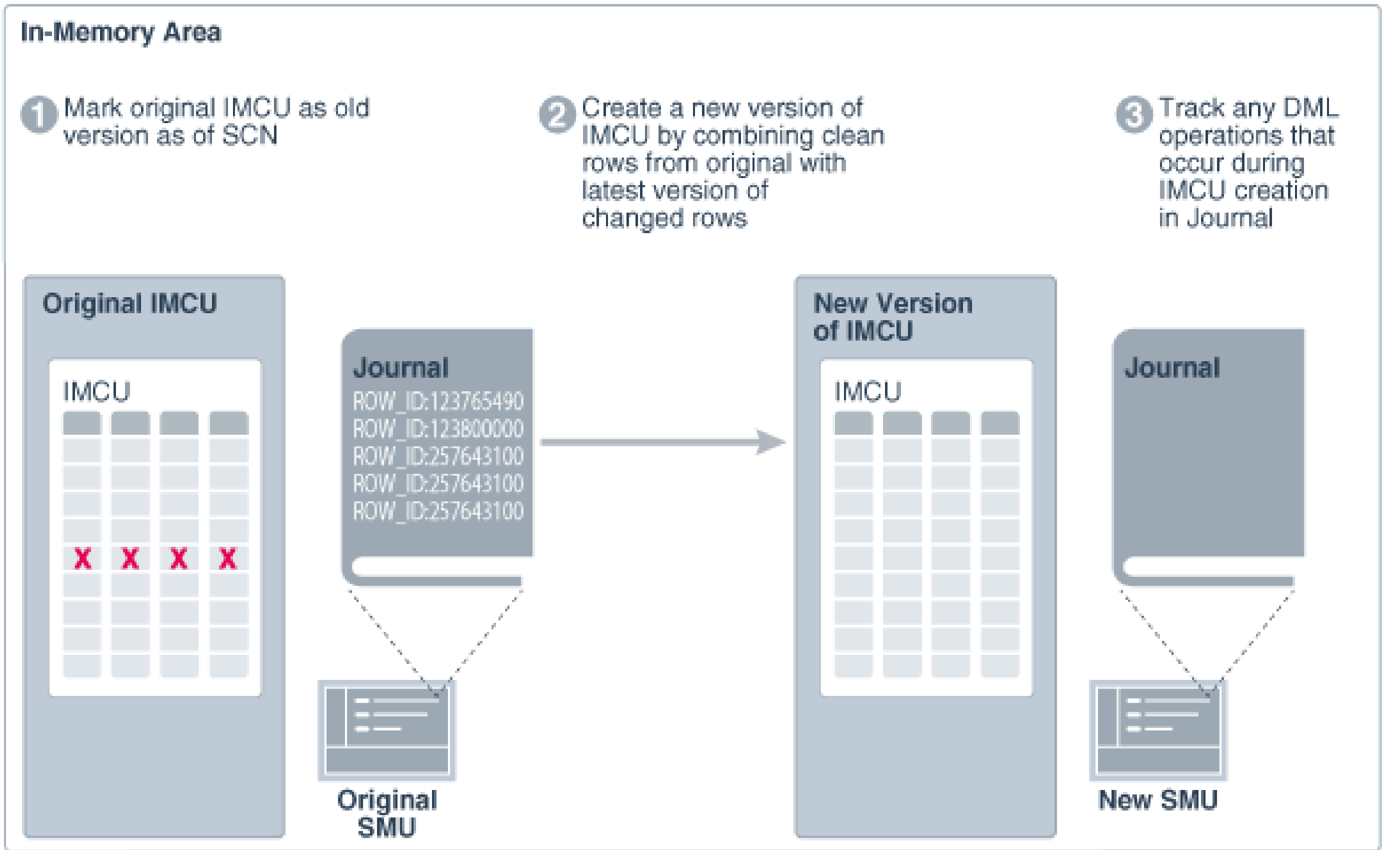
The default isolation level of Oracle DB is "Consistent Read"



- As a query enters the execution stage, the current system change number (**SCN**) is determined.
- Each query returns all committed data with respect to the SCN recorded at the time that query execution began.
- The In-memory Column store follows similar semantics!

Source: Oracle Database Concepts 10g - Data Concurrency and Consistency

Transactional Consistency (Double Buffering)

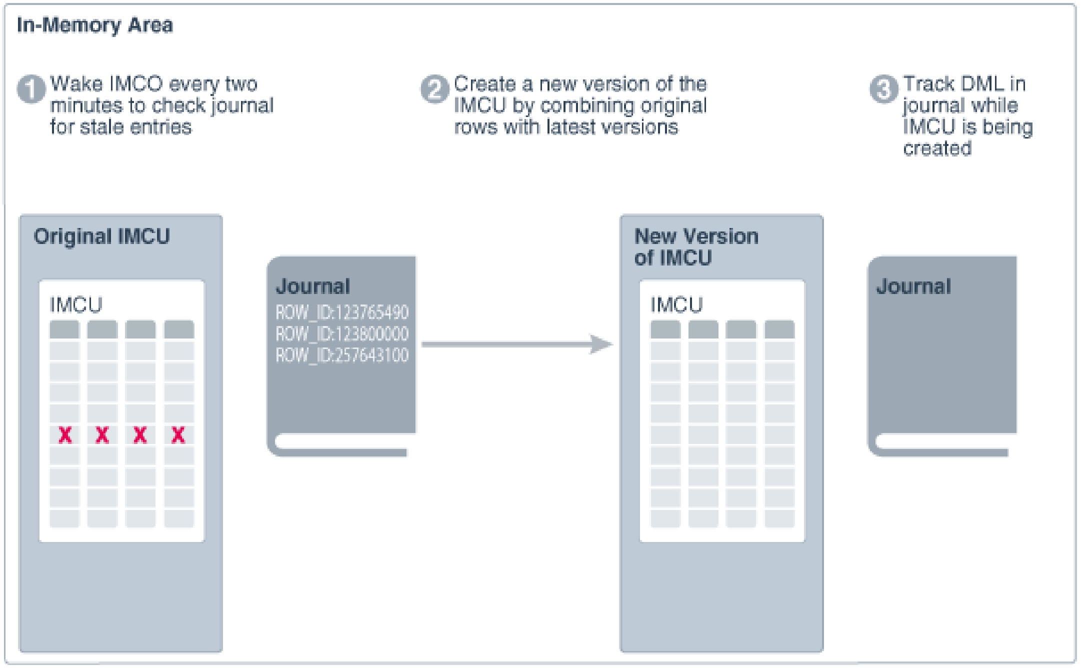


Source: Oracle Database R21 In-Memory Optimizing Repopulation of the IM Column Store

Transactional Consistency (Repopulation Overhead)

There are two types of strategies for data repopulation:

- **Threshold Repopulate**
- **Trickle Repopulate**

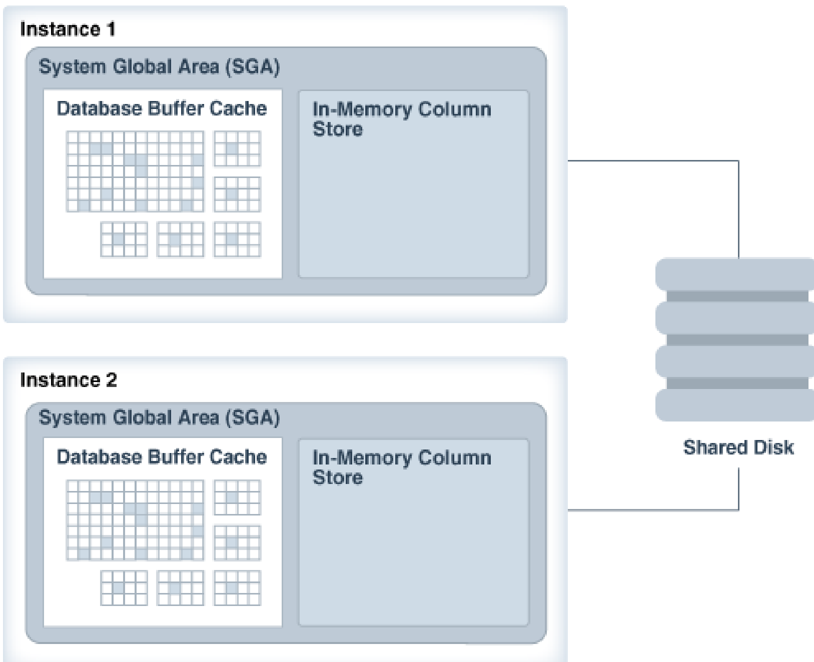


At most $\frac{1}{10}^{th}$ of a single CPU core is dedicated to trickle repopulate

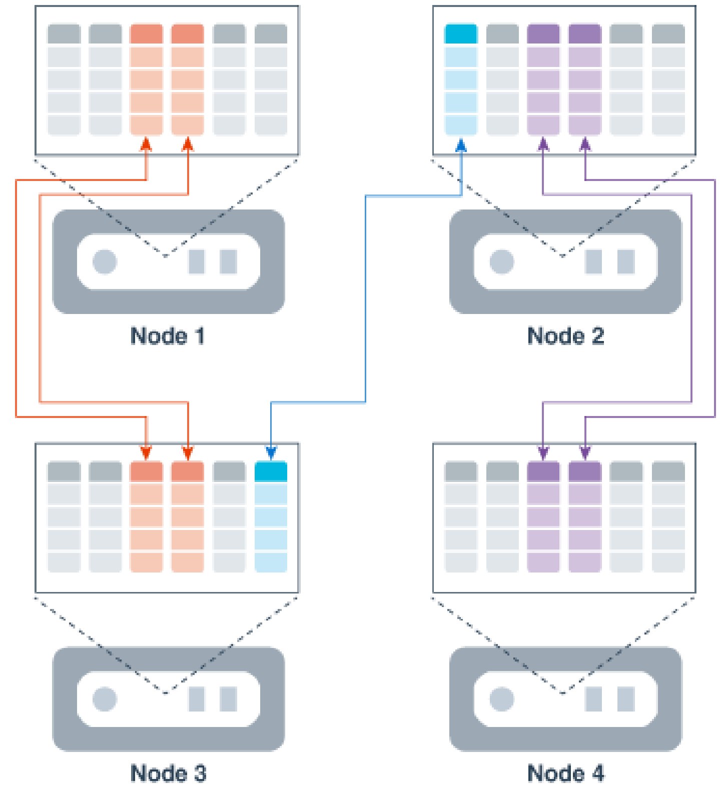
Source: Oracle Database R21 In-Memory Optimizing Repopulation of the IM Column Store

In-Memory Column Store Scale Out

Distribution



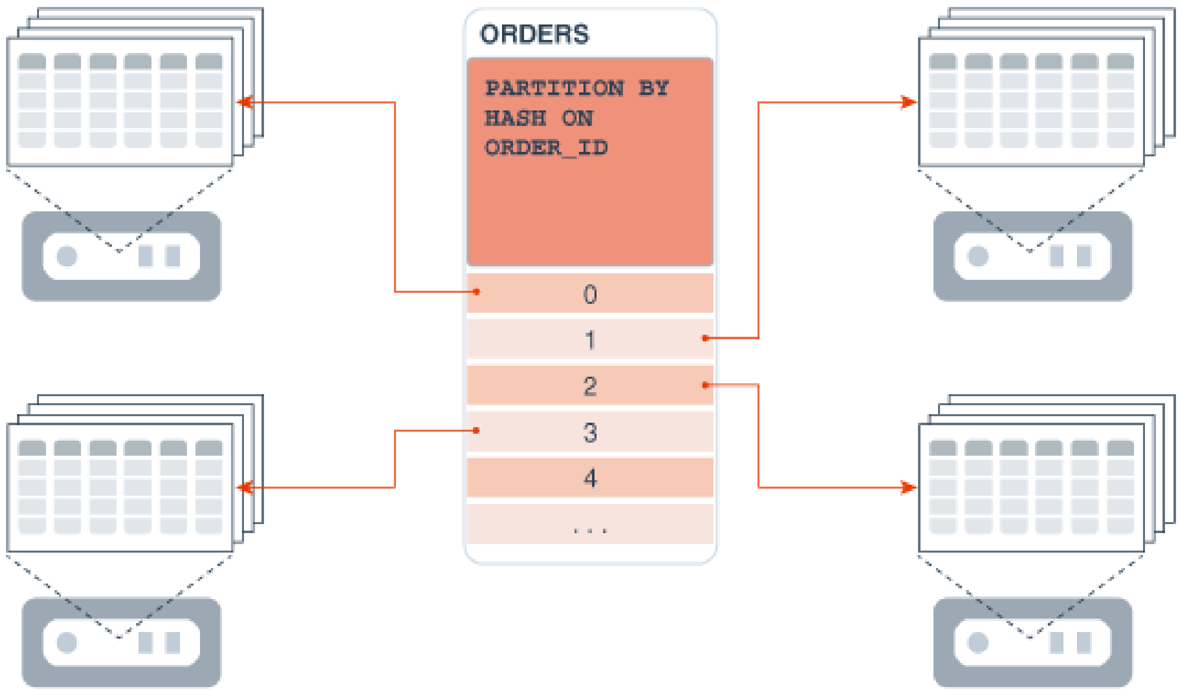
Duplication



Source: Oracle Database R21 In-Memory Deploying IM Column Stores in Oracle RAC

In-Memory Column Store Scale Out (Distribution)

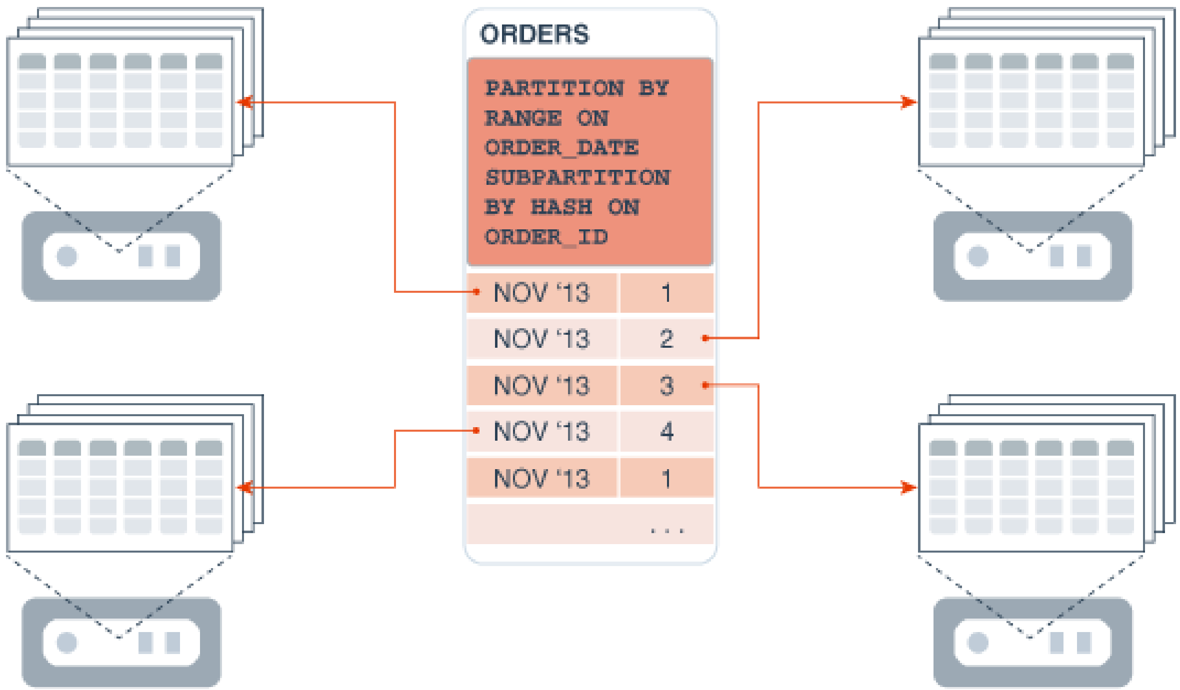
Distributing Partitions by Hash



Source: Oracle Database R21 In-Memory Deploying IM Column Stores in Oracle RAC

In-Memory Column Store Scale Out (Distribution)

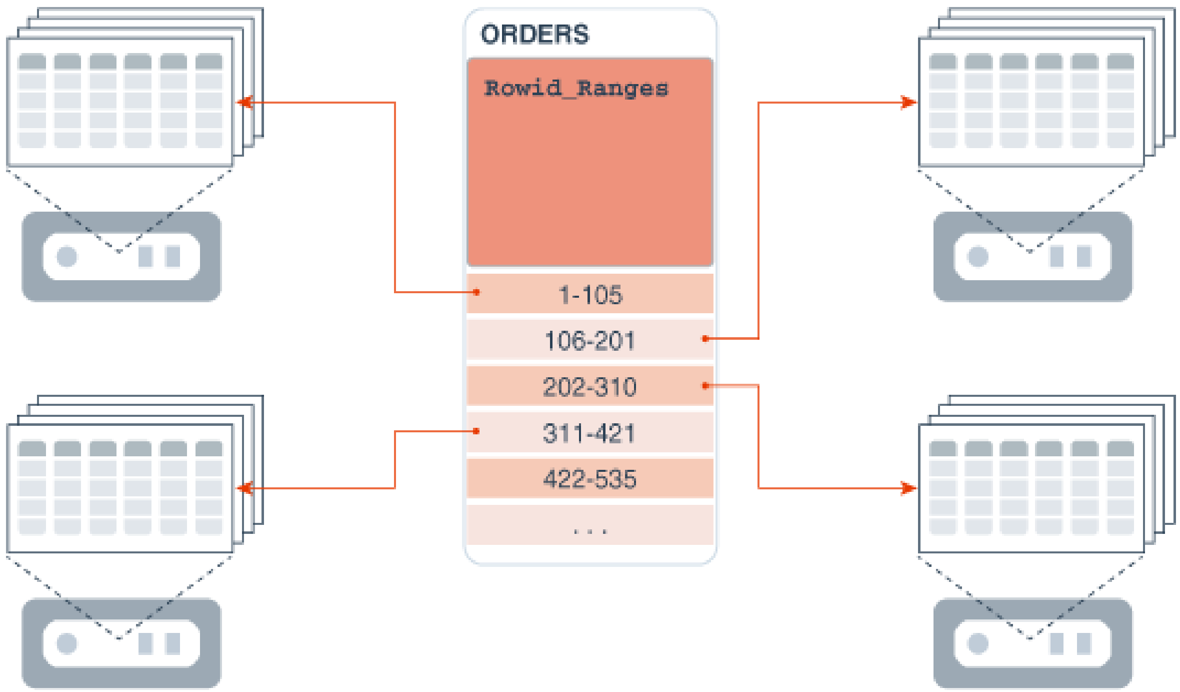
Distributing Partitions by Range and Sub partitions by Hash



Source: [Oracle Database R21 In-Memory Deploying IM Column Stores in Oracle RAC]

In-Memory Column Store Scale Out (Distribution)

Distribution by Rowid Range



Source: [Oracle Database R21 In-Memory Deploying IM Column Stores in Oracle RAC]()

Handling Node Failures

- If an Oracle RAC instance fails, then the IMCUs on the failed instance are unavailable.

- Consequently, a query that needs data stored in the inaccessible IMCUs must read it from somewhere else: the database buffer cache, flash storage, disk, or mirrored IMCUs in other IM column stores.

- Duplication helps to provide fault tolerance because if one node fails, then the mirrored columnar data is accessible from a different node.

- Worst case, queries issued against missing data do not fail. Instead, queries access the data either from the database buffer cache or permanent storage, which may just negatively affect performance.

PURDUE UNIVERSITY

Source: Oracle Database R21 In-Memory Deploying IM Column Stores in Oracle RAC

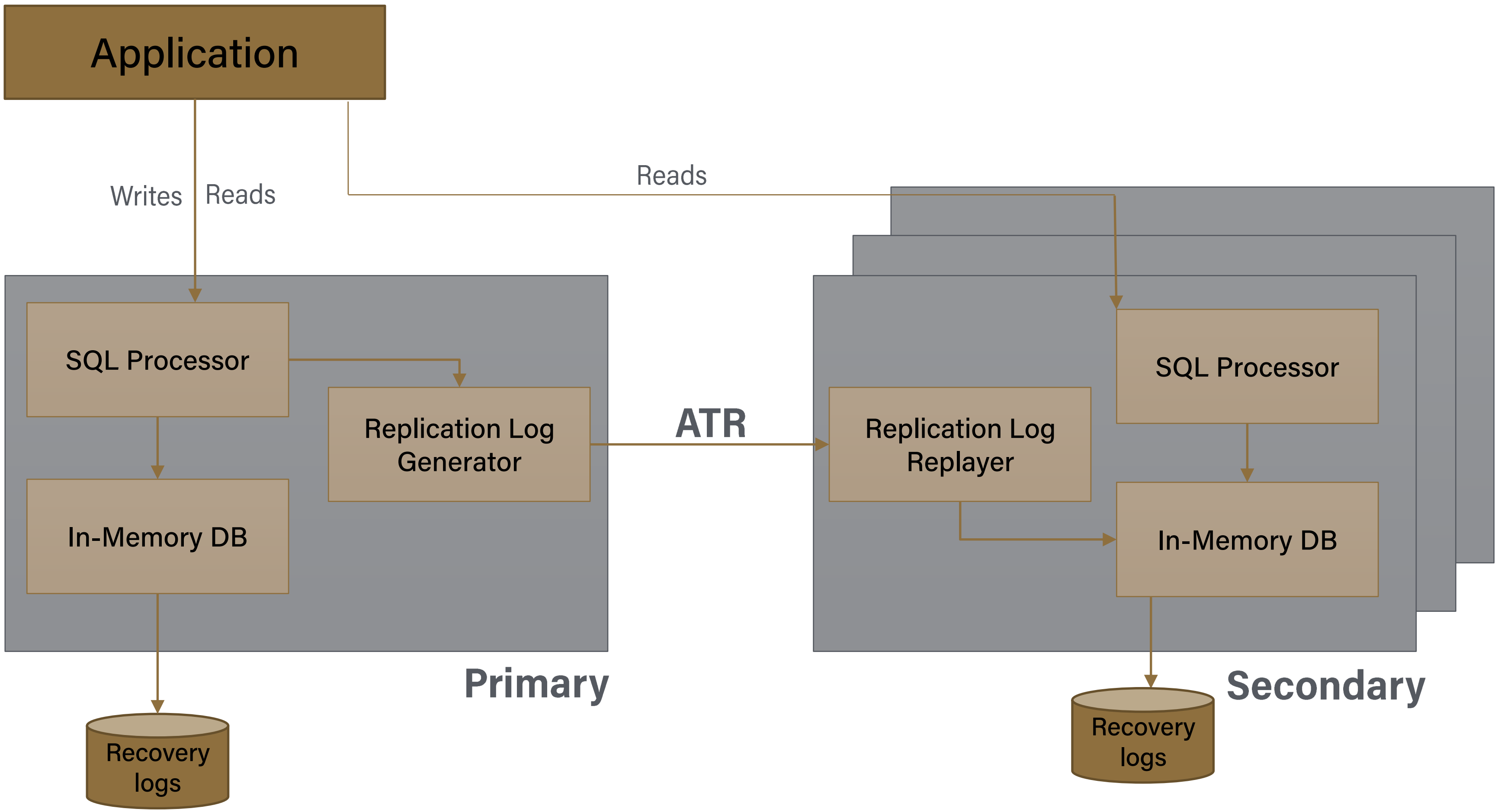
SAP HANA

Disaggl

Presented by
Shubham Pandey

PURDUE UNIVERSITY

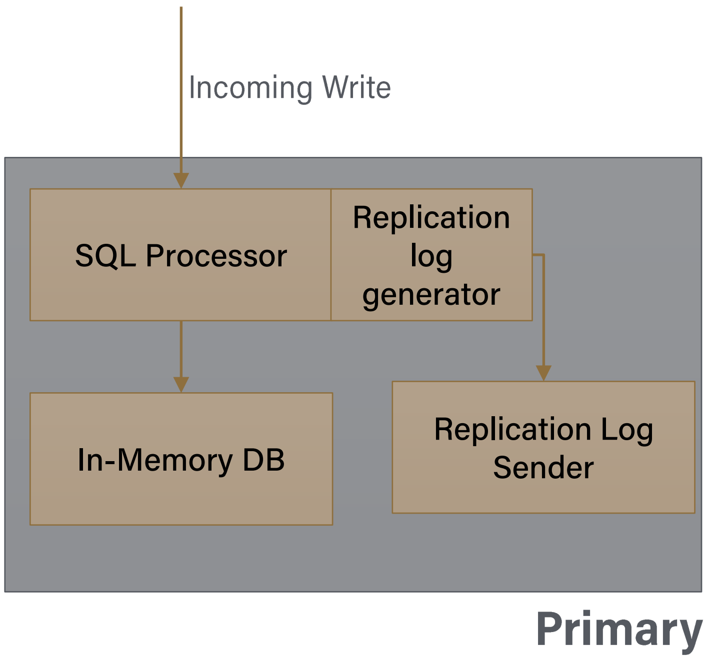
Architecture



Source: http://www.vldb.org/pvldb/vol10/p1598-han.pdf

Design Decisions

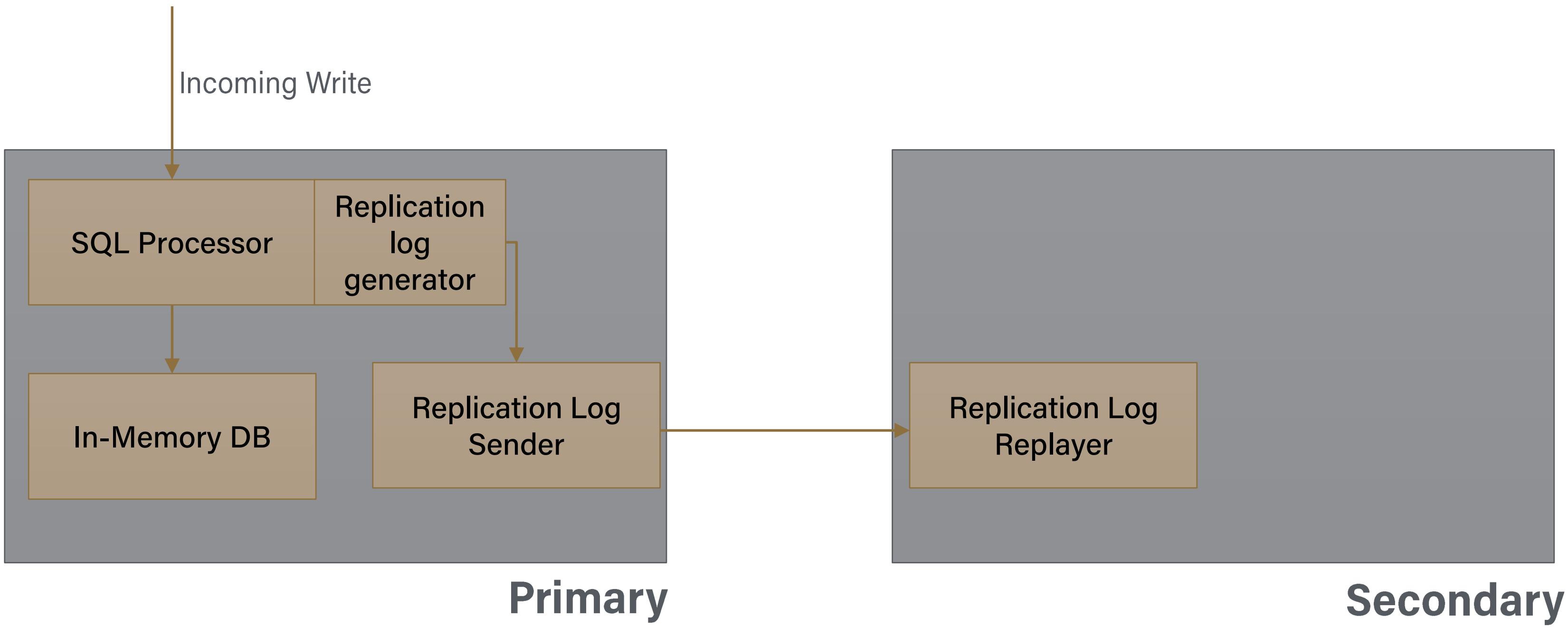
In-Database Replication



Source: http://www.vldb.org/pvldb/vol10/p1598-han.pdf

Design Decisions

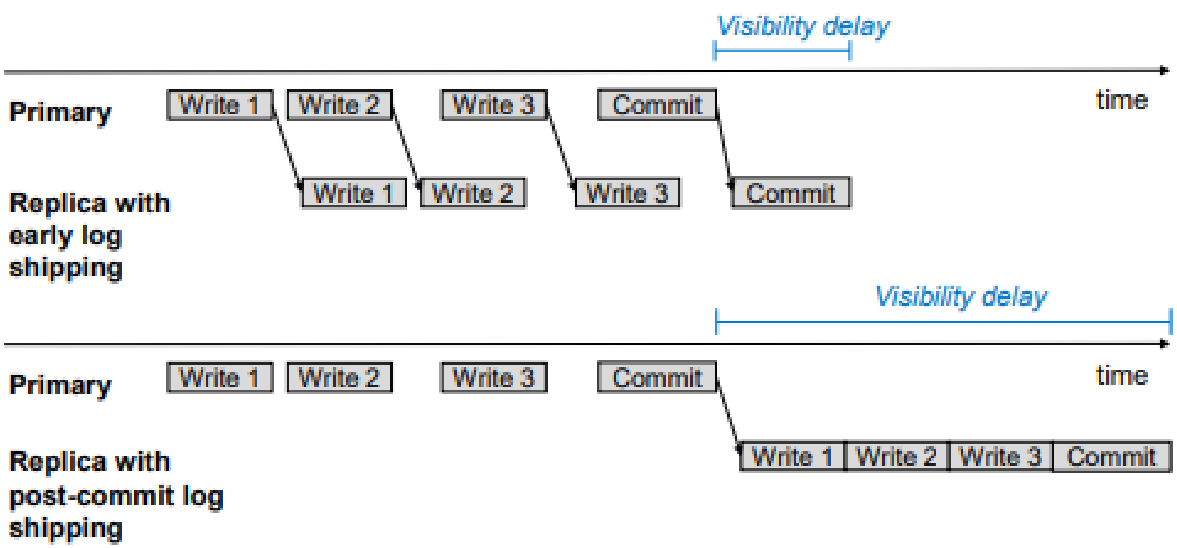
Asynchronous Replication



Source: http://www.vldb.org/pvldb/vol10/p1598-han.pdf

Design Decisions

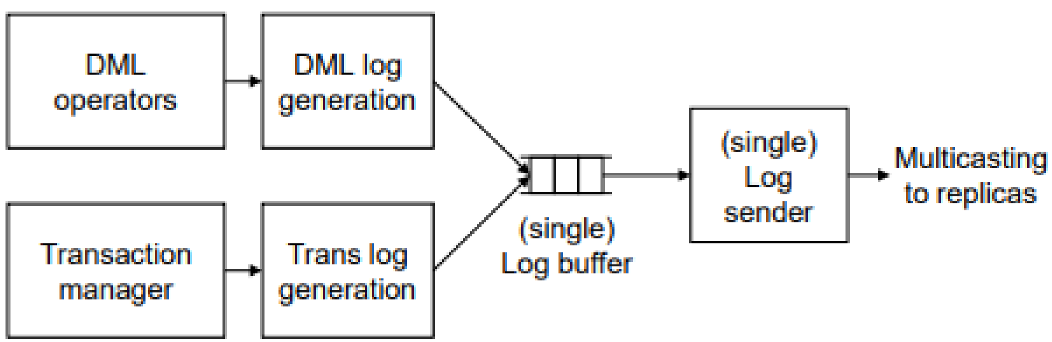
Early Log Shipping



Source: http://www.vldb.org/pvldb/vol10/p1598-han.pdf

Design Decisions

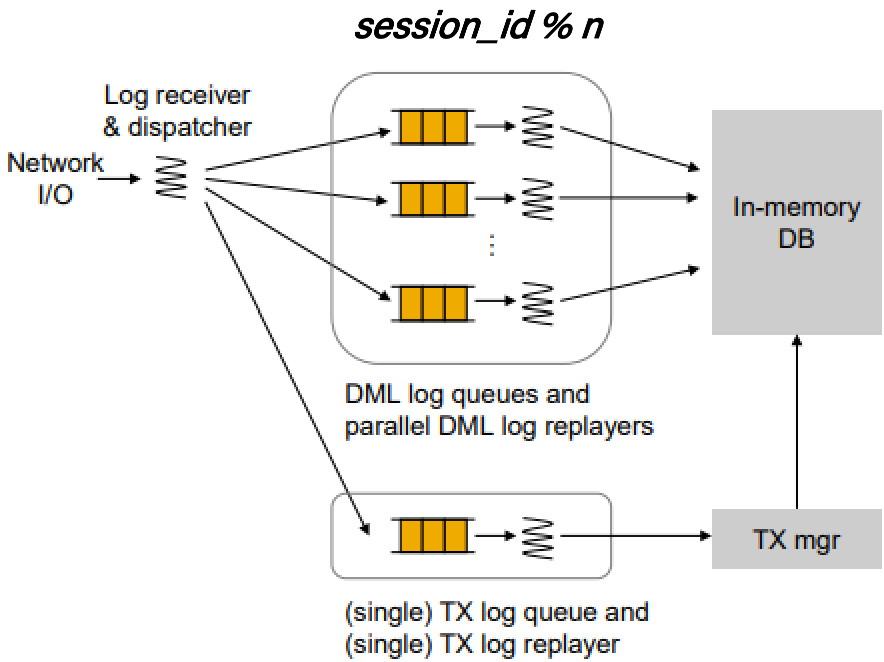
Log generator and Sender



Source: http://www.vldb.org/pvldb/vol10/p1598-han.pdf

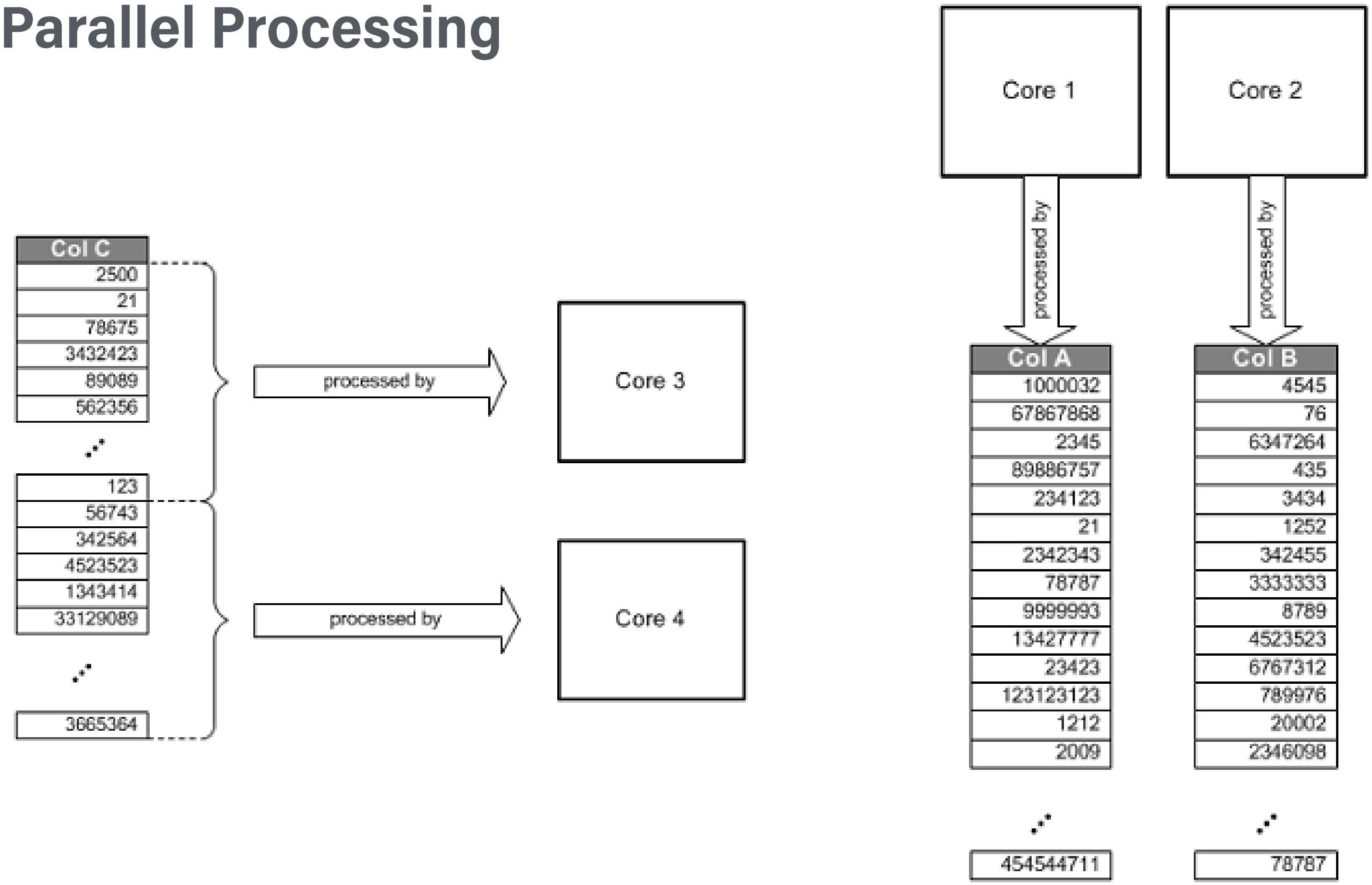
Design Decisions

Parallel Log Replay



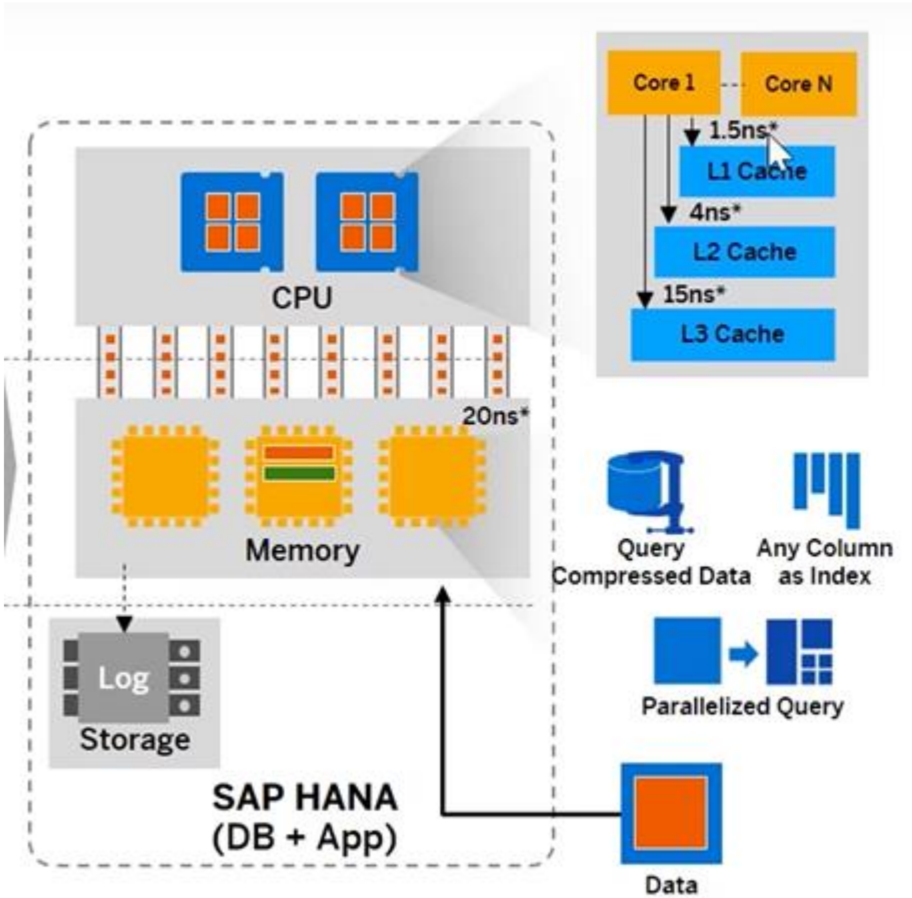
Source: http://www.vldb.org/pvldb/vol10/p1598-han.pdf

Parallel Processing



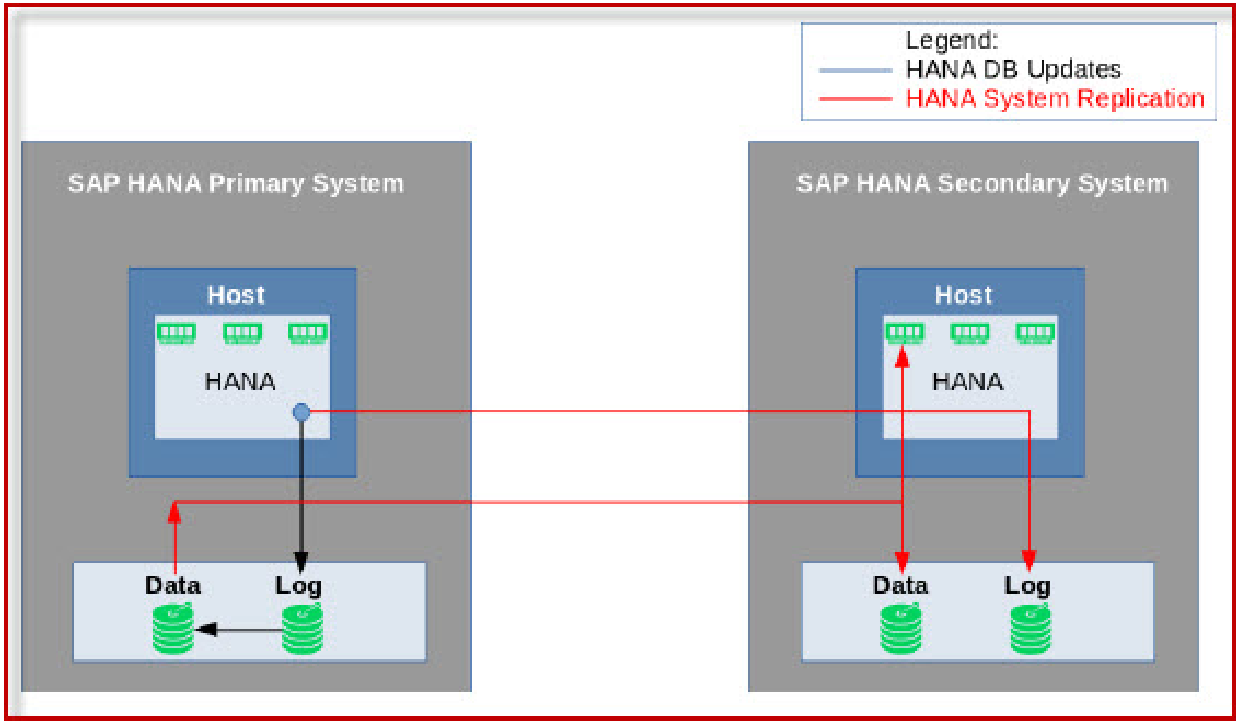
Source: https://help.sap.com/docs/SAP_HANA_PLATFORM

Exploiting Modern Hardware



Source: https://www.youtube.com/watch?v=ljdvqRtSHd4

Fault Tolerance



- Continuous synchronization of HANA DB to secondary location
- Secondary location can be in same data center or a different one

Source: https://blogs.sap.com/2021/12/31/sap-hana-platform-fault-tolerance-features-high-availability/

TiDB: A Raft Based HTAP Database

Disaggl

Presented by
Satya Sai Bharath Vemula

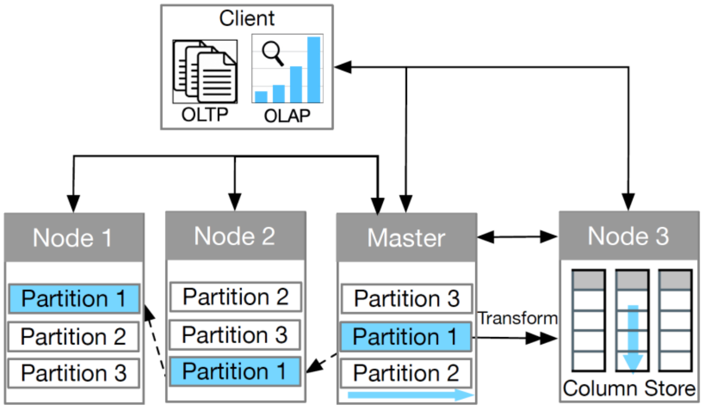
System Goals

System on Higher Level

- Provide Isolation for OLTP and OLAP Queries
    - Providing separate data stores (Row-based and Column-based) for OLTP and OLAP

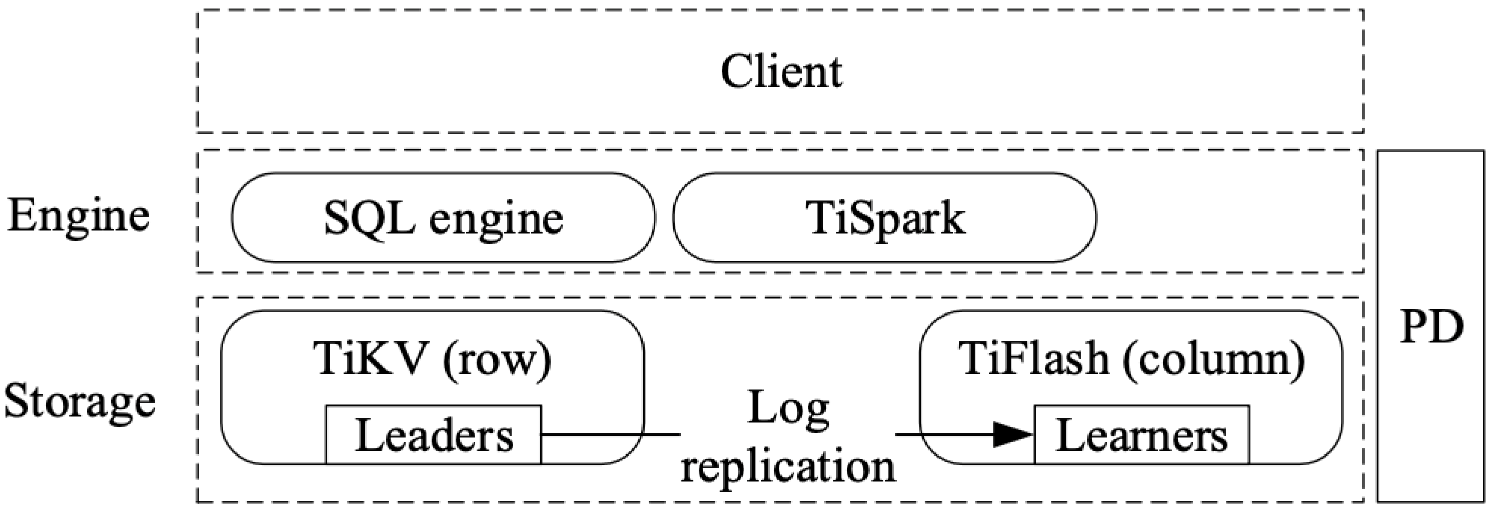
- Providing Consistent View of Data / Real time data synchronization between the two data stores by extending using state machine-based consensus algorithm (*RAFT*)

- Highly Scalable and Available System with efficient query processing



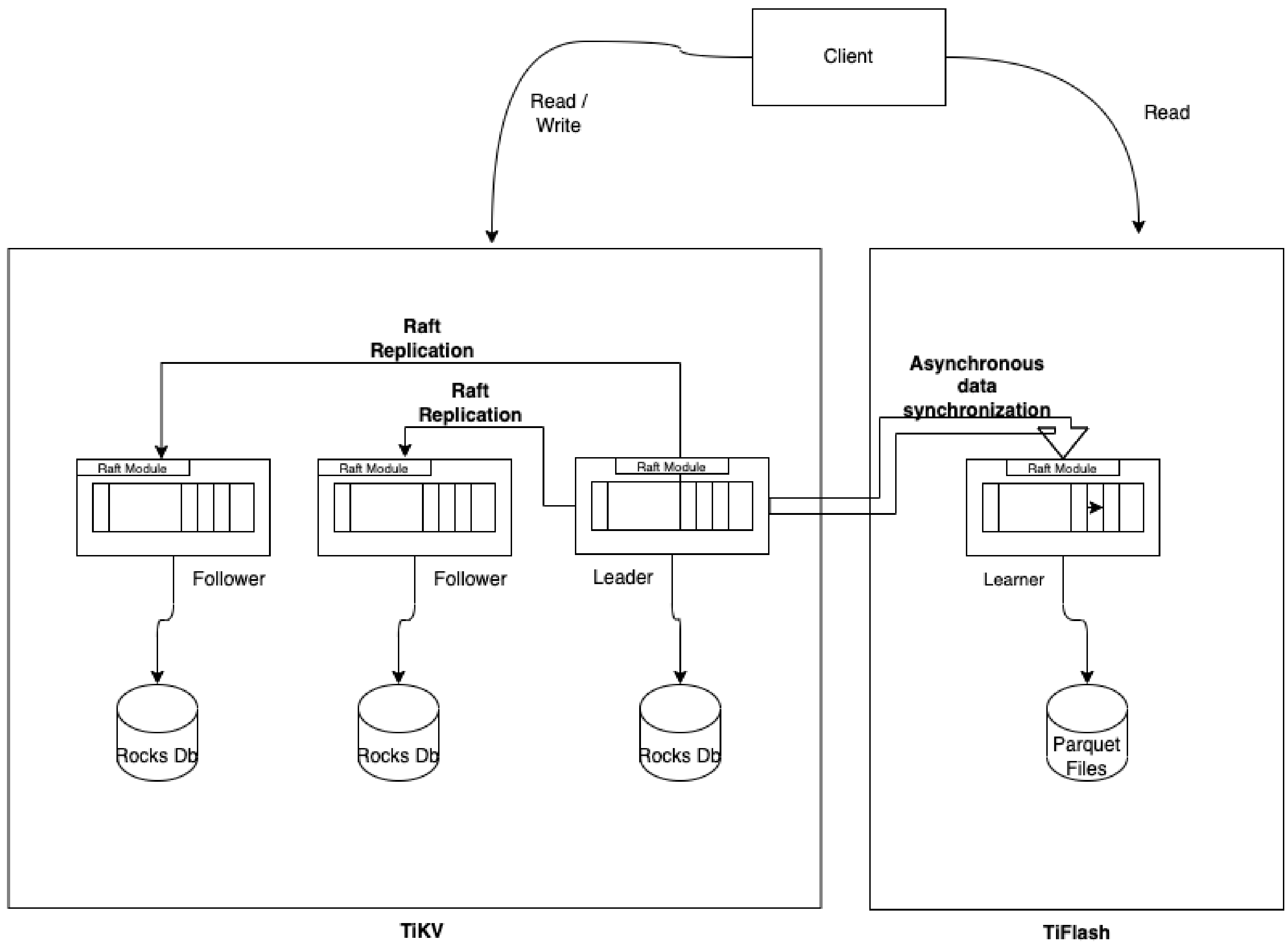
Source: Guoliang Li and Chao Zhang. 2022. HTAP Databases: What is New and What is Next. (SIGMOD '22)

TiDB Architecture



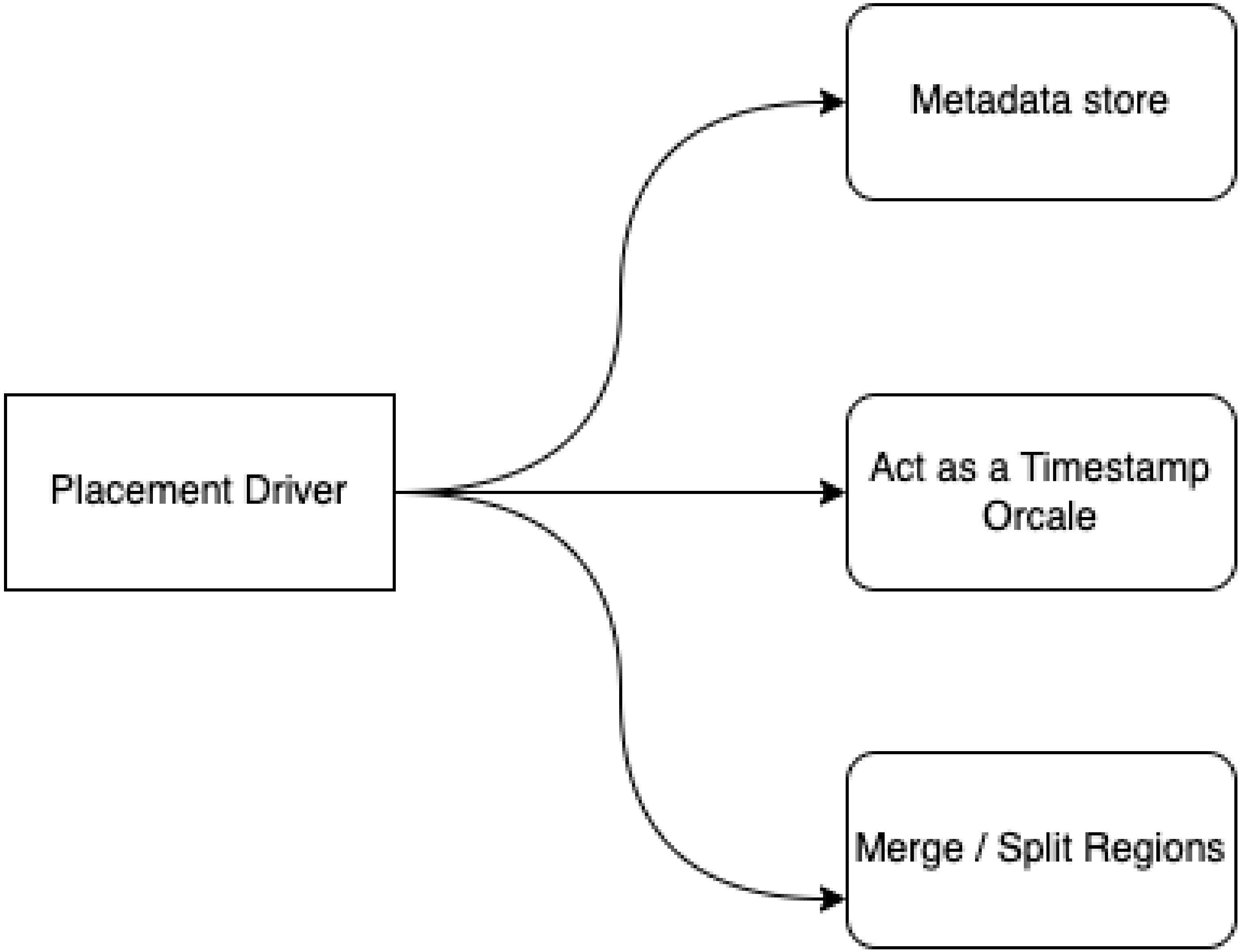
Source: https://www.vldb.org/pvldb/vol13/p3072-huang.pdf

TiDB Raft

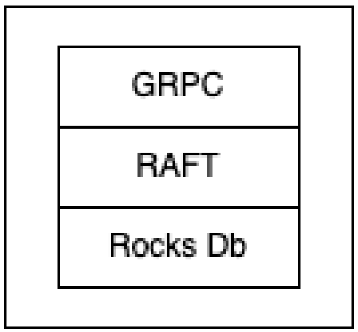


Source: https://www.vldb.org/pvldb/vol13/p3072-huang.pdf

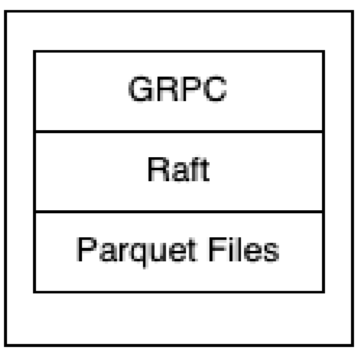
Placement Driver



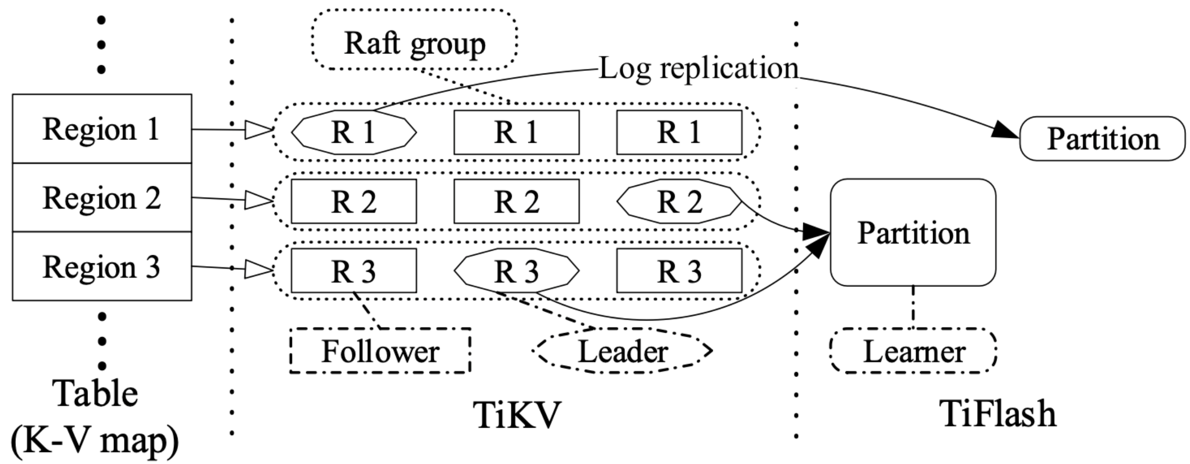
TiDB Scaling



Tikv Node



TiFlash Node



Availability / Failure Tolerance

Failure Tolerance in both TikV and TiFlash is maintained through replicas

- For each TiKV region if there are 2*m + 1 nodes there will be no observed down time if m + 1 nodes are functional in the same network partition.
  - This Property is because of RAFT Availability Semantics
  - OLTP queries for a region will not be functional if more than m + 1 nodes in a region have failed.

- TiFlash supports multiple replicas for the same data
  - If there are N replicas for a single TiFlash Node, The system can bear a failure of N –1 replicas
  - OLAP queries can still be functional if all the TiFlash Nodes are down as the system can also read data from TiKV nodes for serving analytical queries.

Log Replay

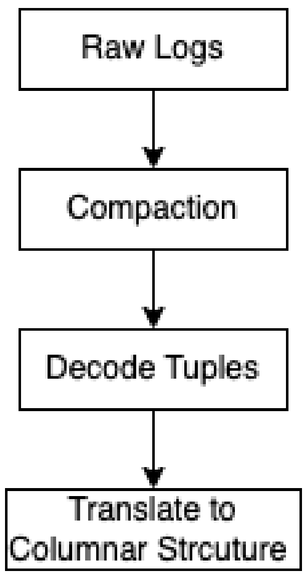


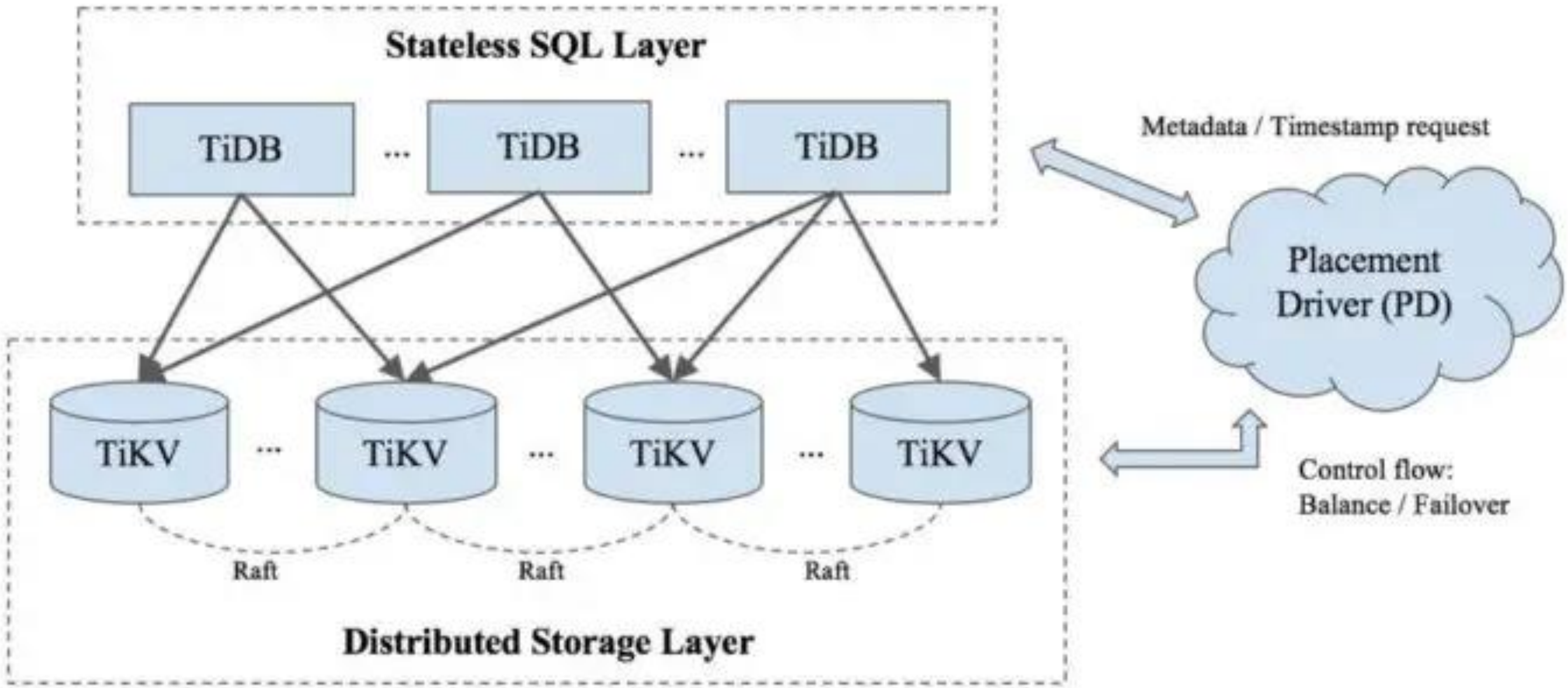
Table 1: Log replaying and decoding

Raw logs	{1}{insert}{prewritten@1}{k1→ (a1, b1)} {2}{insert}{prewritten@2}{k2→ (a2, b2)} {3}{update}{prewritten@3}{k3→ (a3, b3)} {1}{insert}{rollbacked@1} {2}{insert}{committed#4} {3}{update}{committed#5} {4}{delete}{prewritten@6}{k4} {4}{delete}{committed#7}
Compacted logs	{2}{insert}{prewritten@2}{k2→ (a2, b2)} {3}{update}{prewritten@3}{k3→ (a3, b3)} {2}{insert}{committed#4} {3}{update}{committed#5} {4}{delete}{prewritten@6}{k4} {4}{delete}{committed#7}
Decoded tuples	{insert}{#4}{k2→ (a2, b2)} {update}{#5}{k3→ (a3, b3)} {delete}{#7}{k4}
Columnar data	{insert,update,delete,} {#4,#5,#7,} {k2,k3,k4,} {a2,a3,,} {b2,b3,,}

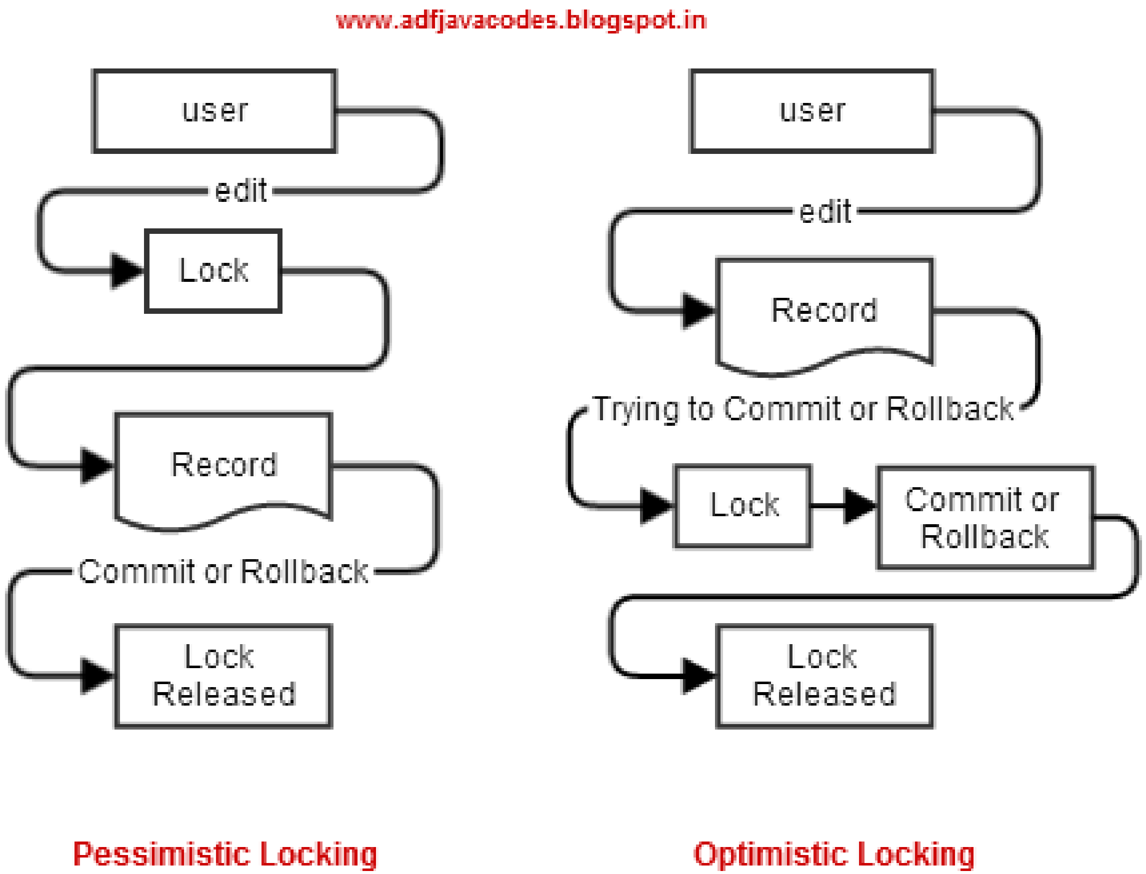
PURDUE UNIVERSITY

Source: https://www.vldb.org/pvldb/vol13/p3072-huang.pdf

OLTP Query Processing



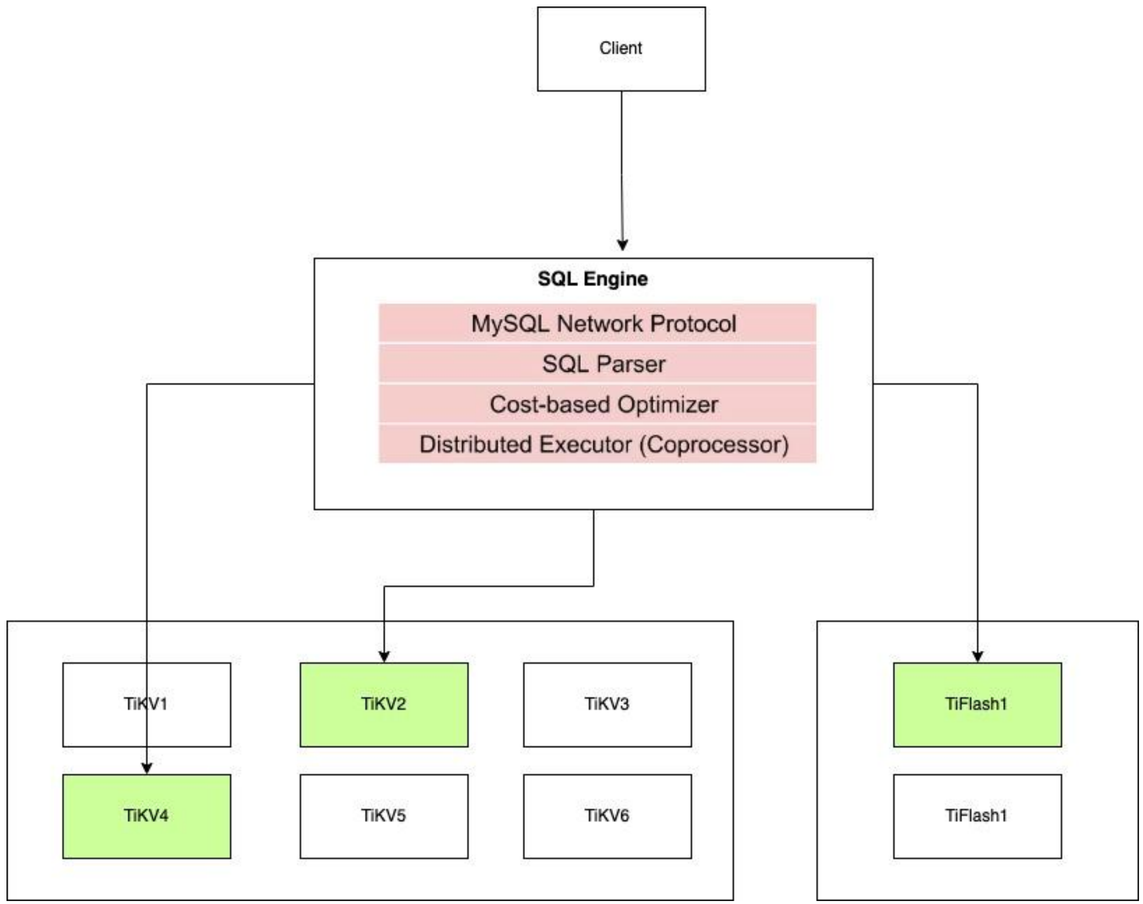
OLTP Distributed Transaction



PURDUE UNIVERSITY

Source: https://adfjavacodes.blogspot.com/2014/05/adf-locking-mechanism-in-adf-find-out.html

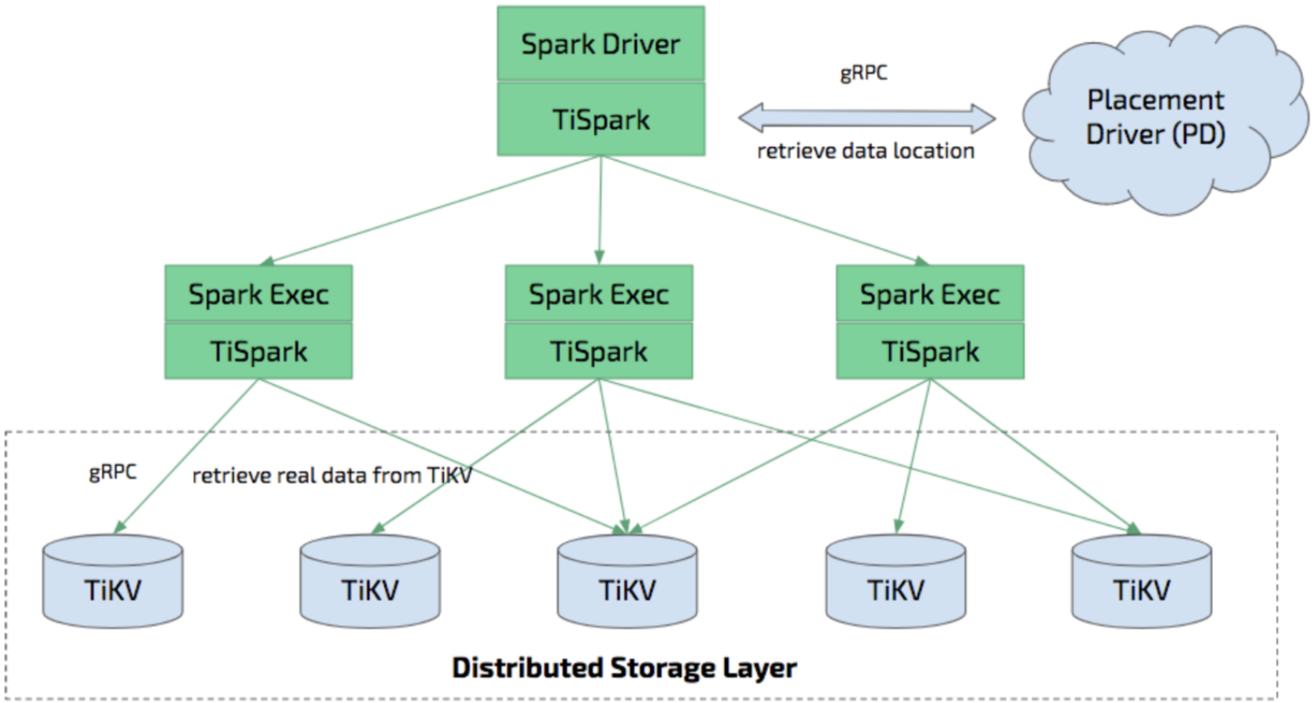
OLAP Query Processing



PURDUE UNIVERSITY®

Source: https://www.vldb.org/pvldb/vol13/p3072-huang.pdf

TiDB TiSpark



Source: https://docs.pingcap.com/tidb/dev/tispark-overview

Project Plan

- **Study HTAP systems with diverse architectures (e.g. data organisation and synchronization)**
  - Compare and Contrast such systems based on their design decisions
  - Assess the strengths and weakness of these systems (in terms of AP/TP throughput, scalability, among others)
- **Study the systems that have evolved from OLTP to HTAP (e.g. MemSQL, IBM dashDB, and more)**

APPENDIX

DETAILED DESCRIPTION

SAP Hana

Replay Algorithms – DML log entry

Require: A DML log entry L.

```
Find the transaction object T for L.TransactionID.
if T is empty then
    Create a transaction object for L.TransactionID.
end if
if L.OperationType = Insert then
    Insert L.Data into the table L.τ.
    Set the inserted record's RVID as L.α.
else if L.OperationType = Delete then
    while true do
        Find the record version R whose RVID equals
    to L.β in the table L.τ.
        if R is not empty then
            Delete R. return
        end if
    end while
else if L.OperationType = Update then
    while true do
        Find the record version R whose RVID equals
    to L.β in the table L.τ.
        if R is not empty then
            Update R with L.Data and L.α. return
        end if
    end while
end if
```

τ – *Table ID*

β – *Before update RVID*

α – After update RVID

PURDUE UNIVERSITY

Source: http://www.vldb.org/pvldb/vol10/p1598-han.pdf

Replay Algorithms

Replay a precommit log entry

Require: A precommit log entry L.

1: Find the transaction object T for $L.TransactionID$.

2: Mark Tś state as *precommitted*.

Replay an abort log entry

Require: An abort log entry L.

1: Find the transaction object T for $L.TransactionID$.

2: Abort T.

Replay a commit log entry

Require: A commit log entry L.

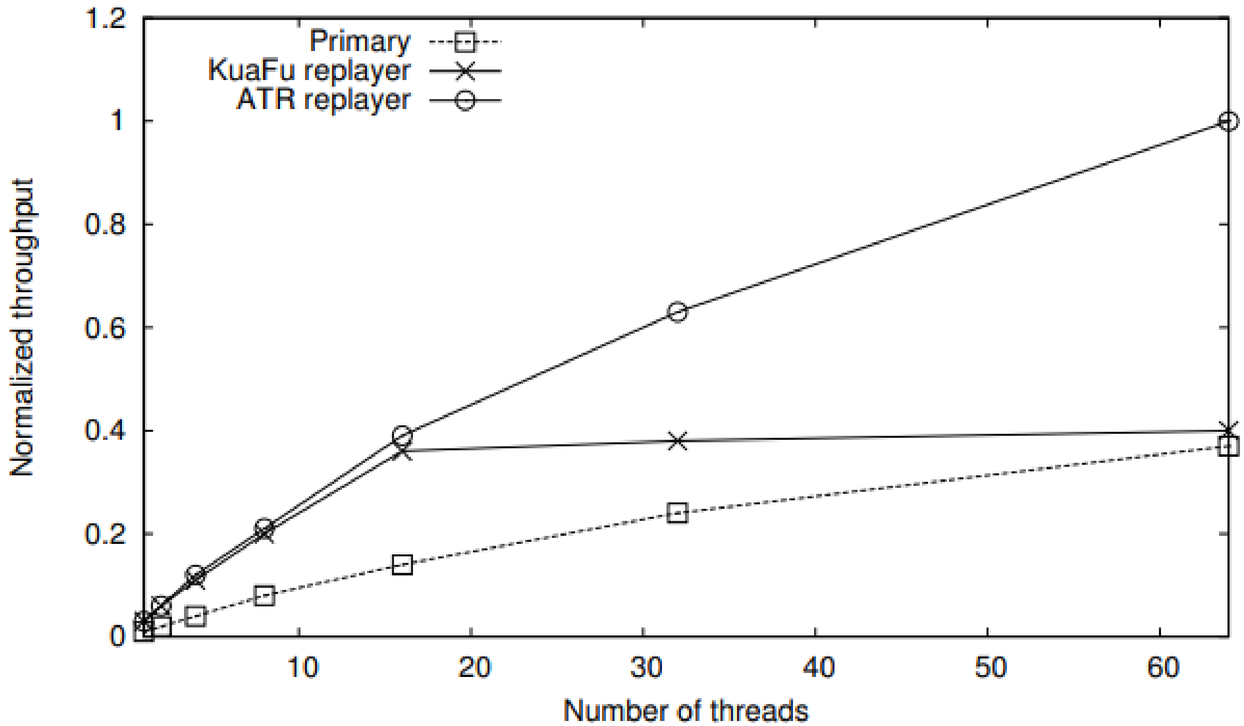
1: Find the transaction object T for $L.TransactionID$.

2: Wait until Tś state becomes *precommitted*.

3: Increment the transaction commit timestamp of the replica server by marking the Tś generated record versions with a new commit timestamp value.

Source: http://www.vldb.org/pvldb/vol10/p1598-han.pdf

Multi-core Scalability with Parallel Log Replay



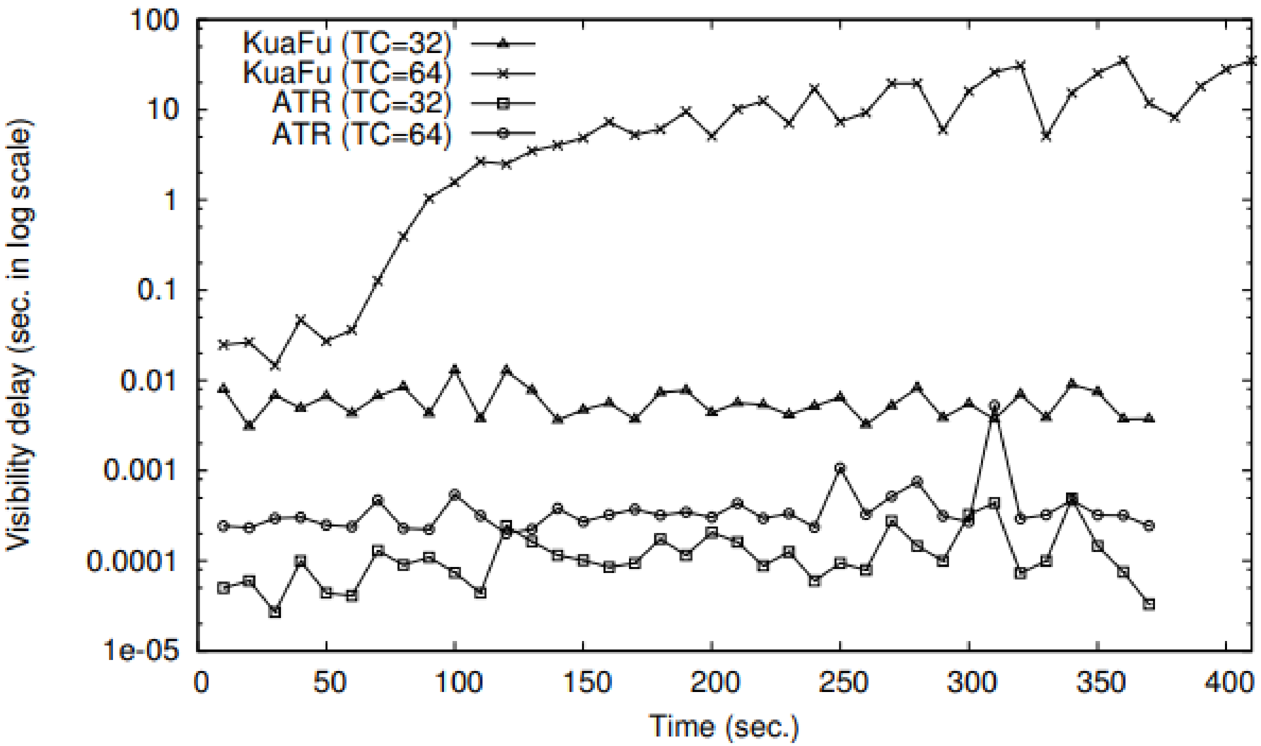
TPC-C benchmark

PURDUE UNIVERSITY

Source: http://www.vldb.org/pvldb/vol10/p1598-han.pdf

SAP Hana

Visibility Delay

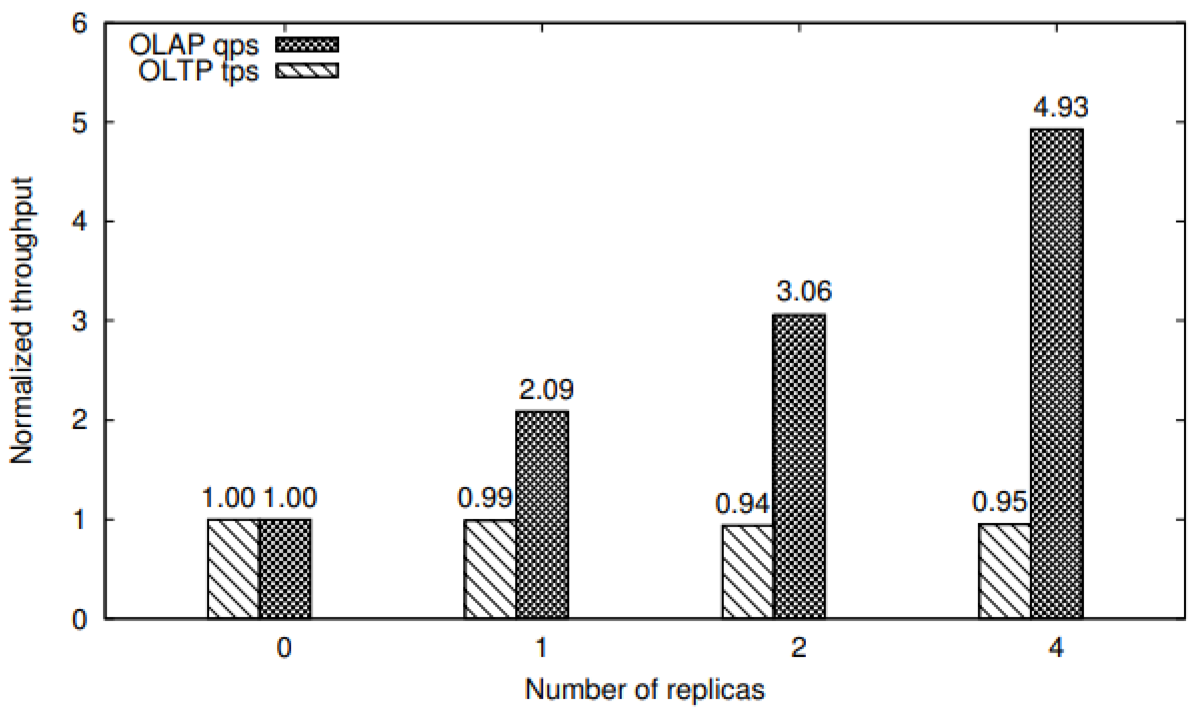


Visibility delay at replica while running TPC-C benchmark at primary

Source: http://www.vldb.org/pvldb/vol10/p1598-han.pdf

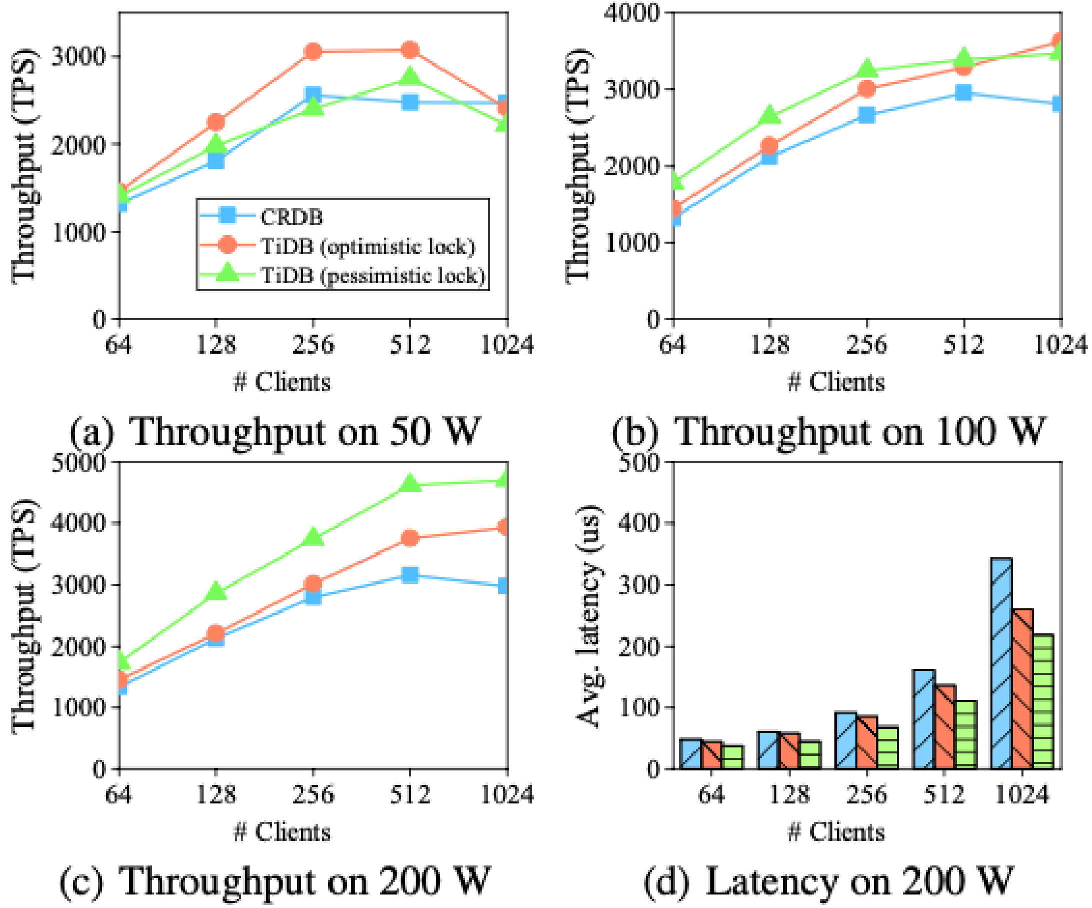
SAP Hana

Multi Replica scalability under mixed (OLTP+OLAP) workload



Source: http://www.vldb.org/pvldb/vol10/p1598-han.pdf

TiDb OLTP Performance Comparison



Source: https://www.geeksforgeeks.org/three-phase-commit-protocol/

OLAP Performance

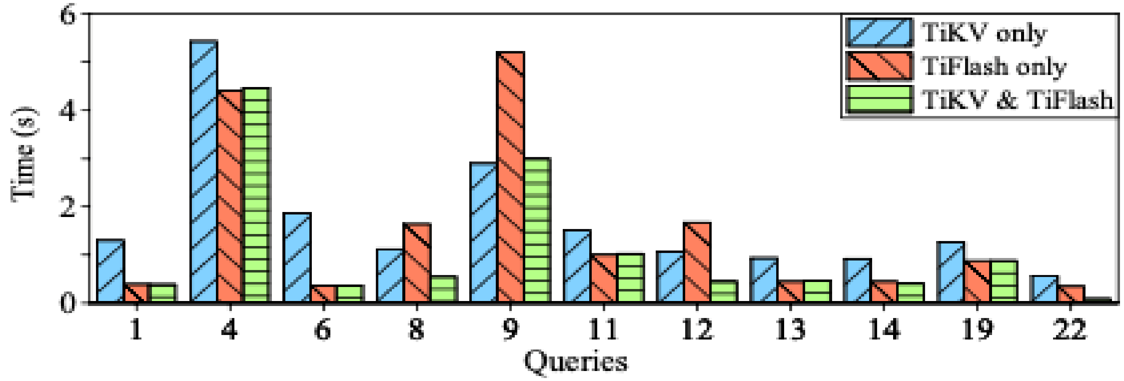


Figure 8: Choice of TiKV or TiFlash for analytical queries

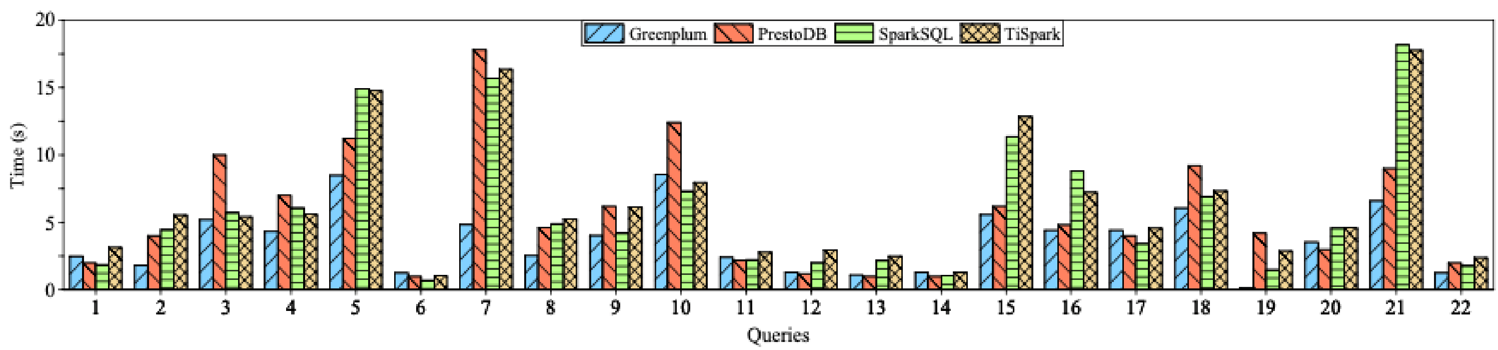


Figure 9: Performance comparison of CH-benCHmark analytical queries

PURDUE UNIVERSITY

Source: https://www.vldb.org/pvldb/vol13/p3072-huang.pdf

HTAP Performance Results

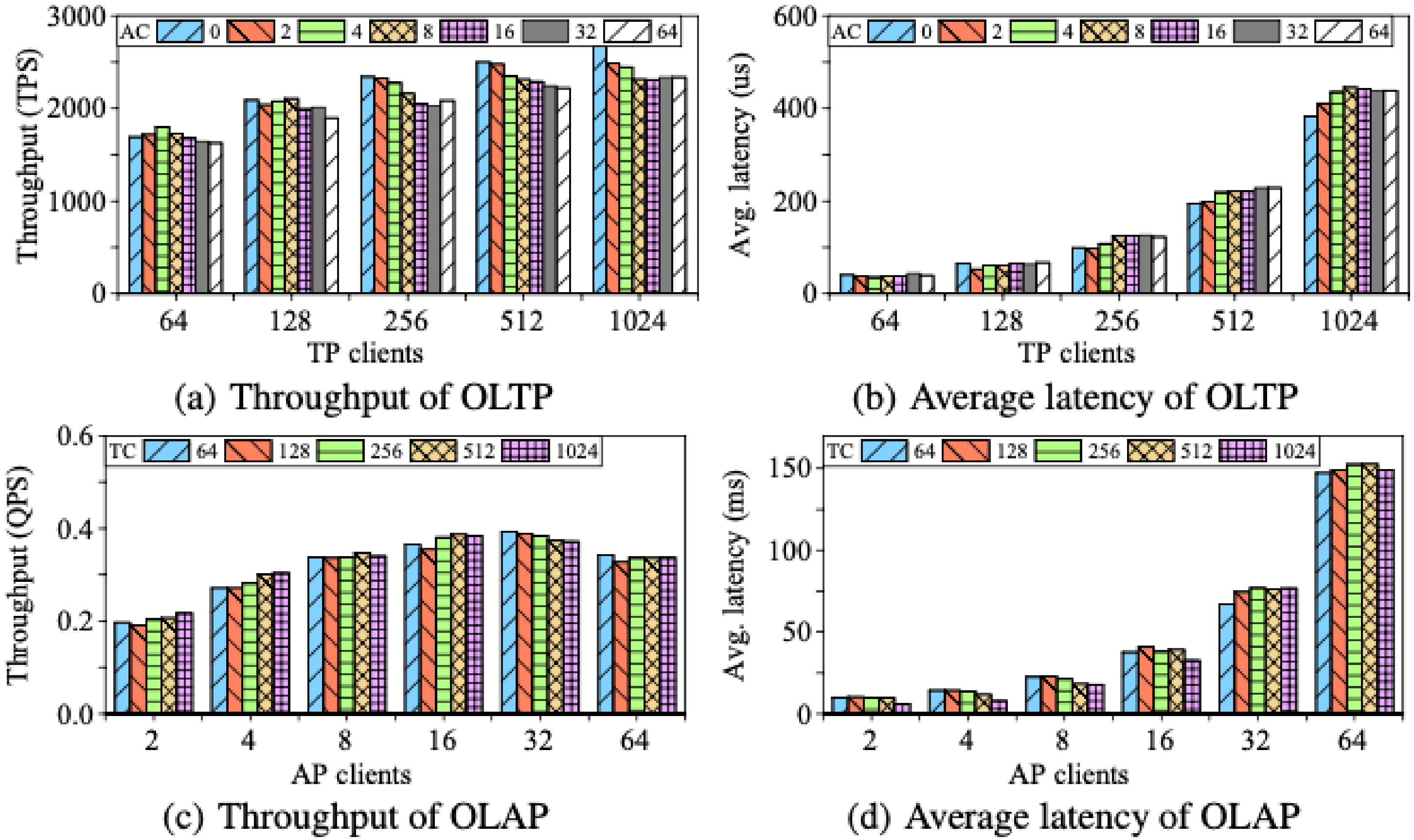


Figure 10: HTAP performance of TiDB

PURDUE UNIVERSITY

Source: https://www.vldb.org/pvldb/vol13/p3072-huang.pdf

Raft Learner

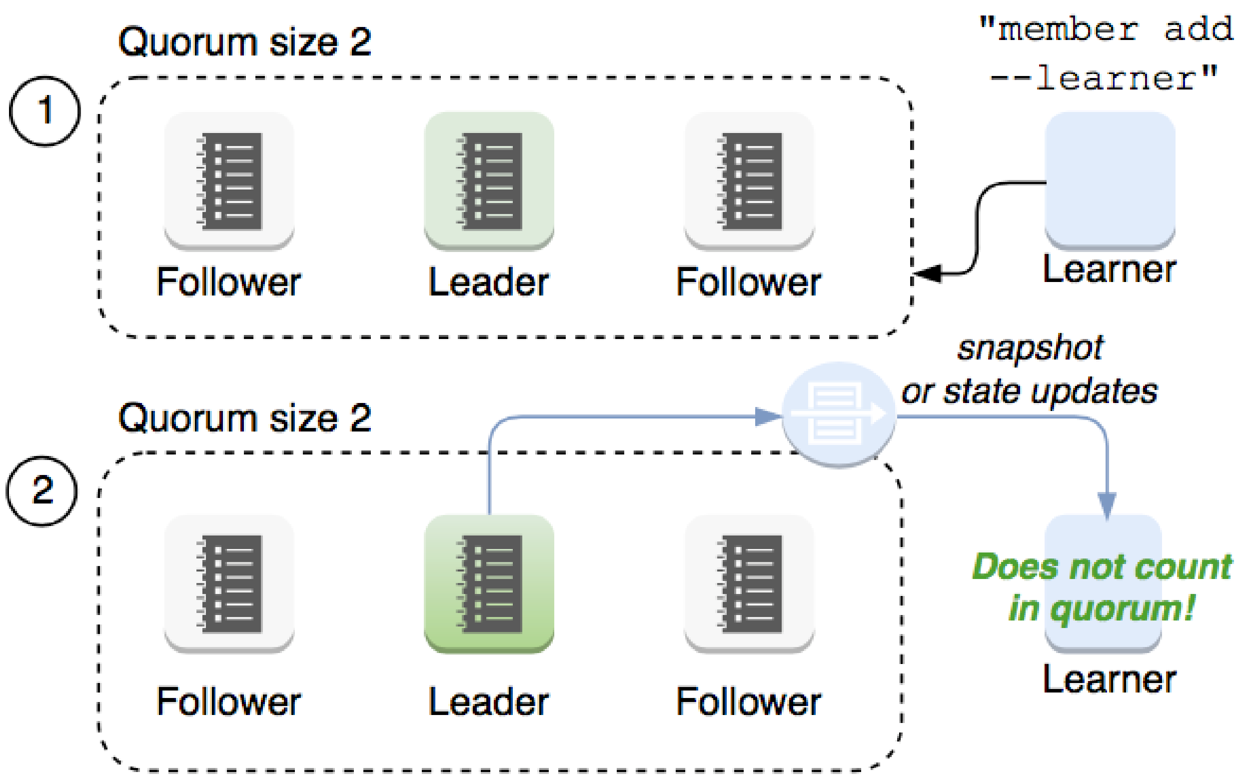
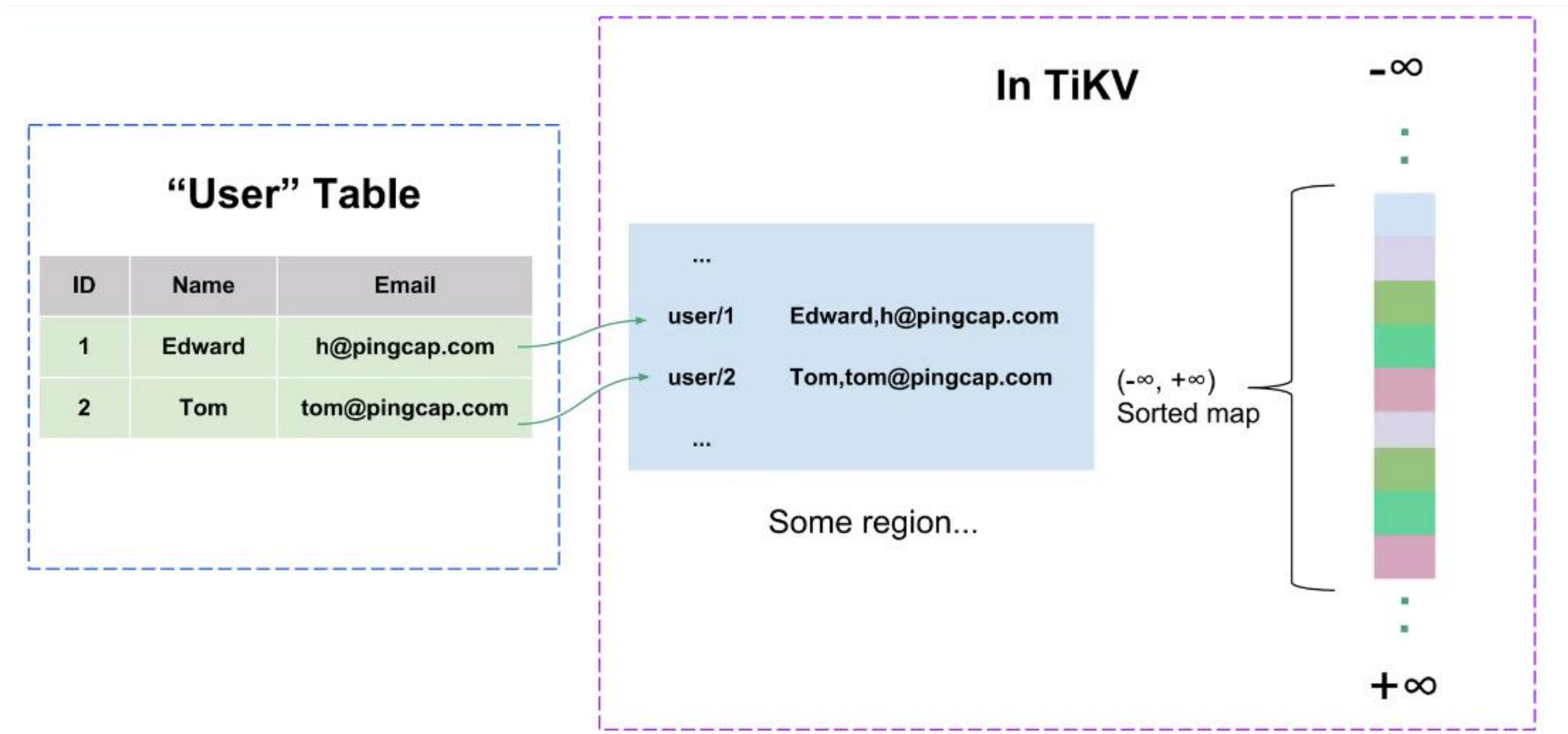


Figure 10. Add a learner node as a non-voting member. Wait until learner node catches up to leader's logs. Until then, learner node neither votes nor counts towards quorum.

PURDUE UNIVERSITY

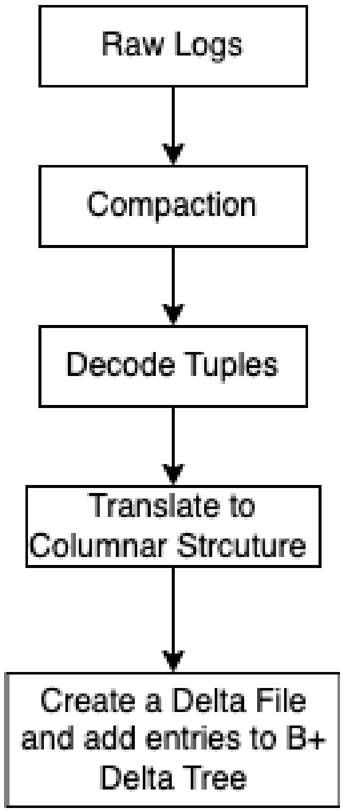
Source: https://assets.ctfassets.net/oxjq45e8ilak/1N3hH0IB4uDWS9zrcMaNA3/5fbd6c5db4835b022202a385505625b2/TiDB-EdHuang.pdf

TiKV Table Storage



Source: https://www.slideshare.net/morgo/tidb-introduction

Appendix (OLAP Optimization)



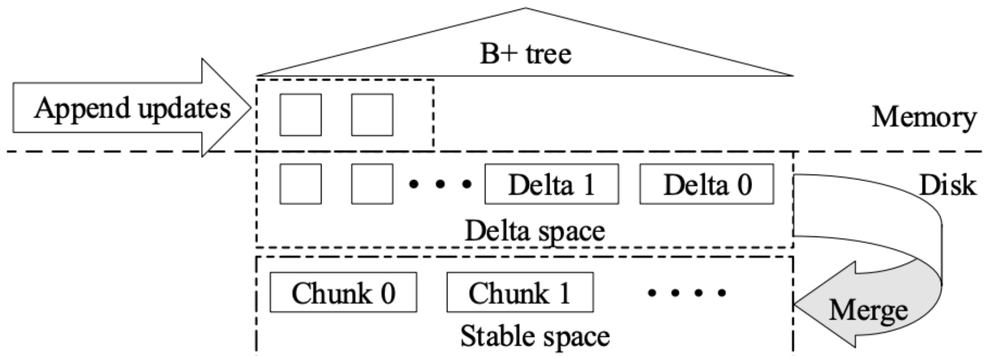


Figure 4: The columnar delta tree

Source: https://www.vldb.org/pvldb/vol13/p3072-huang.pdf

Appendix (Cost Modeling)

$$\mathtt{C_{opt\_scan}} = \min(\mathtt{C_{col\_scan}}, \mathtt{C_{row\_scan}}, \mathtt{C_{index\_scan}})$$

$$\mathtt{C_{row\_scan}} = \mathtt{S_{tuple}} \cdot \mathtt{N_{tuple}} \cdot \mathtt{f_{scan}} + \mathtt{N_{reg}} \cdot \mathtt{f_{seek}}$$

$$\mathtt{C_{col\_scan}} = \sum_{j=1}^{m} (\mathtt{S_{col\_j}} \cdot \mathtt{N_{tuple}} \cdot \mathtt{f_{scan}} + \mathtt{N_{reg\_j}} \cdot \mathtt{f_{seek}})$$

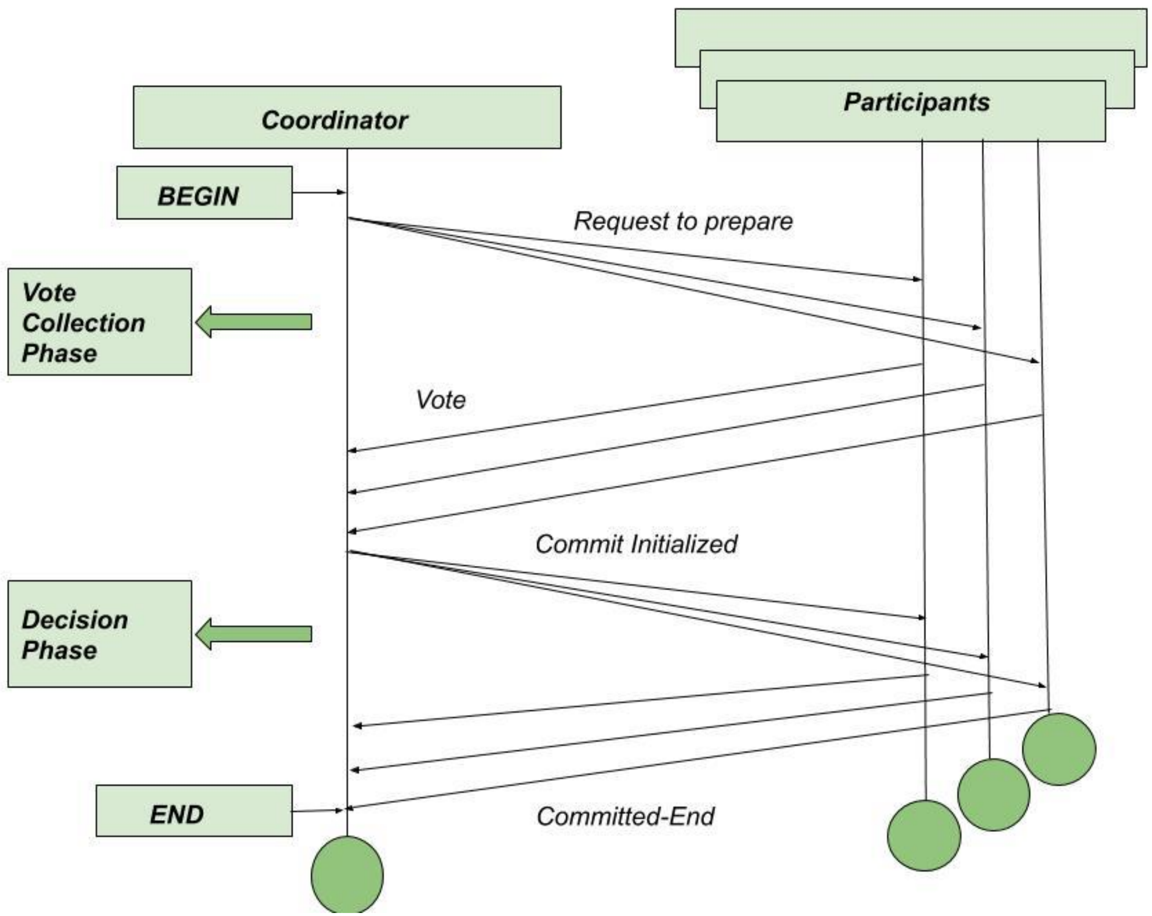
$$\mathtt{C_{index\_scan}} = \mathtt{S_{index}} \cdot \mathtt{N_{tuple}} \cdot \mathtt{f_{scan}} + \mathtt{N_{reg}} \cdot \mathtt{f_{seek}} + \mathtt{C_{double\_read}}$$

$$\mathtt{C_{double\_read}} = \begin{cases} 0 & \text{(if without double read)} \\ \mathtt{S_{tuple}} \cdot \mathtt{N_{tuple}} \cdot \mathtt{f_{scan}} + \mathtt{N_{tuple}} \cdot \mathtt{f_{seek}} \end{cases}$$

C -> Cost
S -> Size
F -> Seek / Scan Cost
N -> Number

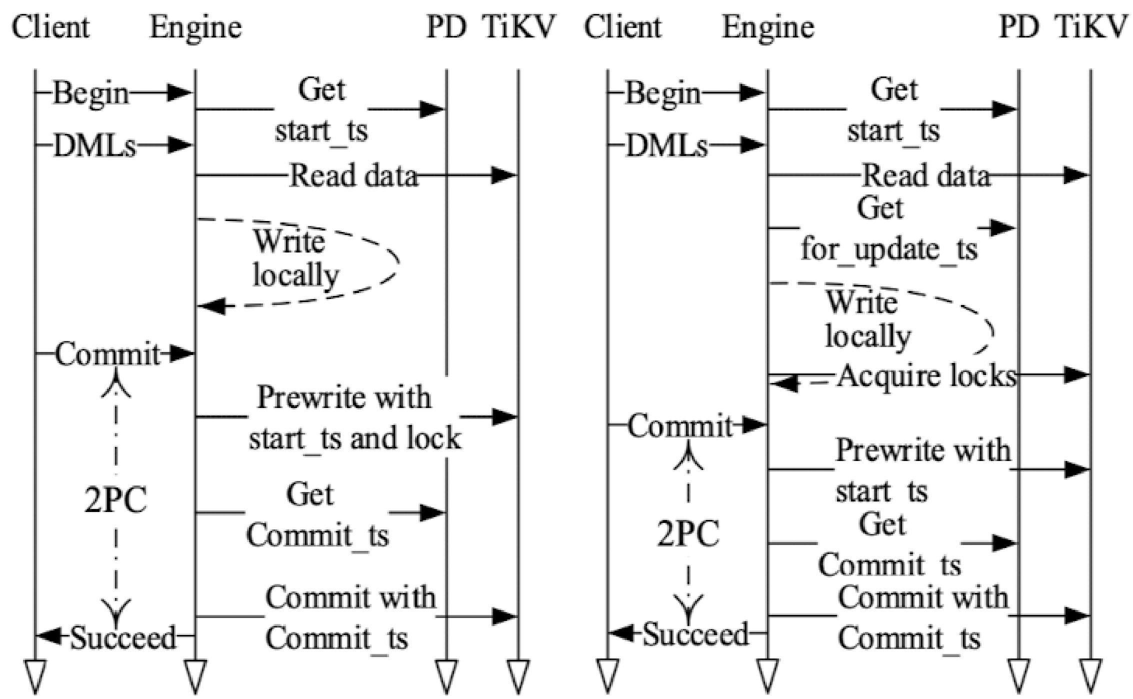
Source: https://www.vldb.org/pvldb/vol13/p3072-huang.pdf

Appendix (General Two-Phase Commit)



Source: https://www.geeksforgeeks.org/three-phase-commit-protocol/

Appendix (TiDB Two-Phase Commit)



(a) Optimistic transaction (b) Pessimistic transaction

Figure 5: The process of optimistic and pessimistic transaction