CumuloNimbo:
A Cloud Scalable SQL Database

Rui Carlos Gonçalves

HASLab, INESC TEC & U. Minho

Porto Linux, 2016

RoadMap

- Background Research
- Scalable Transaction Processing
- Scalable Analytics

Research

CoherentPaaS

Yousuf Ahmad Ainhoa Azqueta Ivan Brondino Fábio Coelho

Francisco Cruz Bettina Kemme Rui Gonçalves Ricardo Jimenez

Miguel Matos Rui Oliveira Marta Patiño José Pereira Ricardo Vilaça

These projects received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreements no 257993, 611068 and 619606.

HASLab
HIGH-ASSURANCE SOFTWARE LABORATORY

Background

- Big Data: Explosion of data being generated and stored every day
  - Holds valuable knowledge for organizations' operation
- Opportunity to improve efficiency
- New challenges to store and process massive amounts of data

Scalable Transaction Processing

Scalable Transaction Processing

- To offer the classical ACID properties with useful isolation levels, current clustered databases scale poorly both in terms of performance and efficiency

Scalable Transaction Processing

- To offer the classical ACID properties with useful isolation levels, current clustered databases scale poorly both in terms of performance and efficiency
- To scale, current systems need to sacrifice atomicity, consistency and overall ease of use

Scalable Transaction Processing

- To offer the classical ACID properties with useful isolation levels, current clustered databases scale poorly both in terms of performance and efficiency
- To scale, current systems need to sacrifice atomicity, consistency and overall ease of use
- The Big Data challenge to current clustered databases is to allow analytical processing over the freshest data

Scalable Transaction Processing

- To offer the classical ACID properties with useful isolation levels, current clustered databases scale poorly both in terms of performance and efficiency
- To scale, current systems need to sacrifice atomicity, consistency and overall ease of use
- The Big Data challenge to current clustered databases is to allow analytical processing over the freshest data
- Requires **H**ybrid **T**ransactional and **A**nalytical **P**rocessing - simultaneous transactional and analytical workloads

CumuloNimbo

CumuloNimbo

- CumuloNimbo is a framework for multi-tier applications that provides ultra-scalable and fault-tolerant processing of OLTP and OLAP workloads

CumuloNimbo

- CumuloNimbo is a framework for multi-tier applications that provides ultra-scalable and fault-tolerant processing of OLTP and OLAP workloads
- It provides a standard SQL interface and full ACID transactional support without resorting to sharding

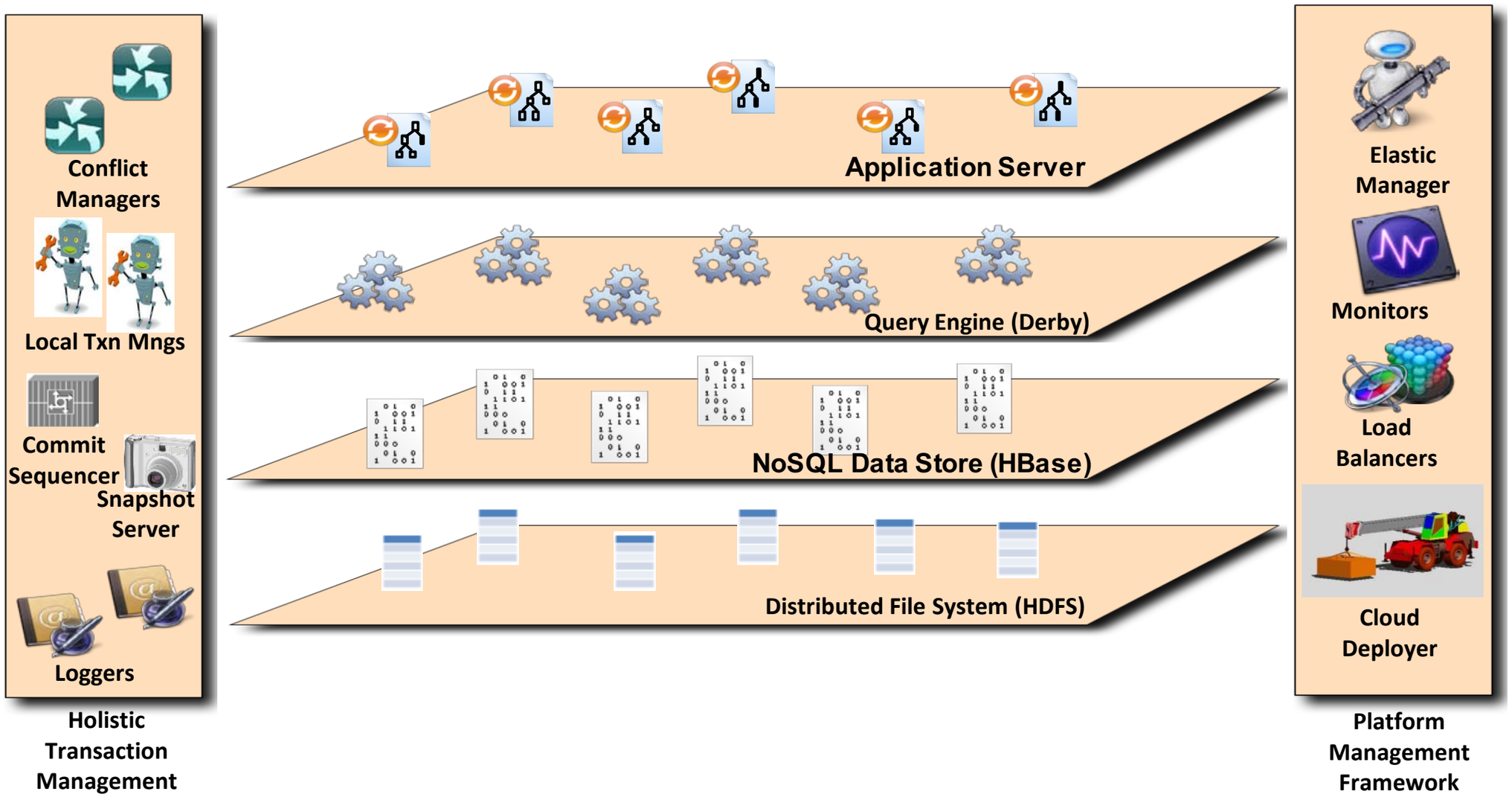
CumuloNimbo

- CumuloNimbo is a framework for multi-tier applications that provides ultra-scalable and fault-tolerant processing of OLTP and OLAP workloads
- It provides a standard SQL interface and full ACID transactional support without resorting to sharding
- Data is persisted in a distributed fault-tolerant data store

CumuloNimbo

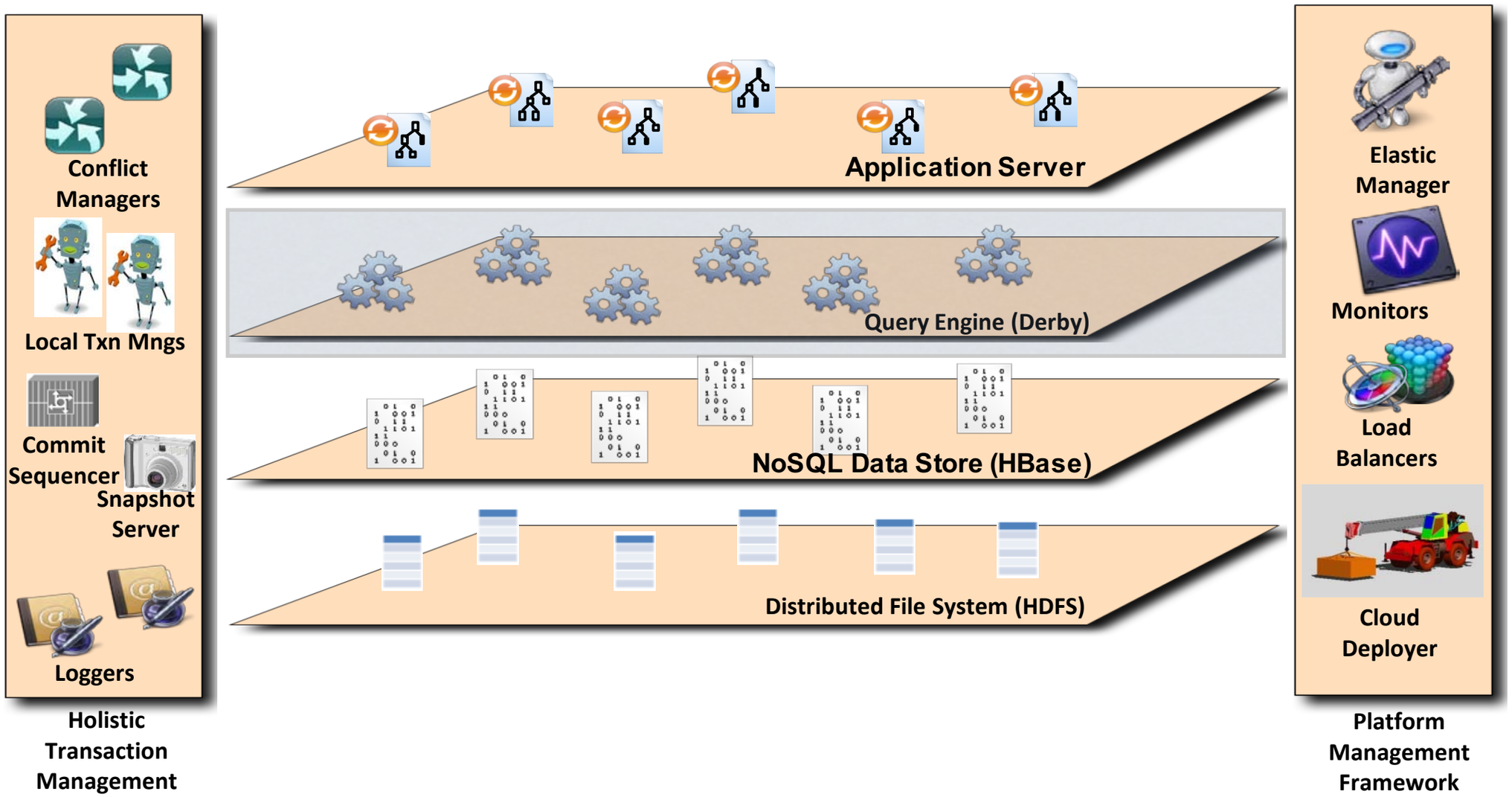
- CumuloNimbo is a framework for multi-tier applications that provides ultra-scalable and fault-tolerant processing of OLTP and OLAP workloads
- It provides a standard SQL interface and full ACID transactional support without resorting to sharding
- Data is persisted in a distributed fault-tolerant data store
- Scalability is achieved by distributing request execution and transaction control across many compute nodes

Architecture



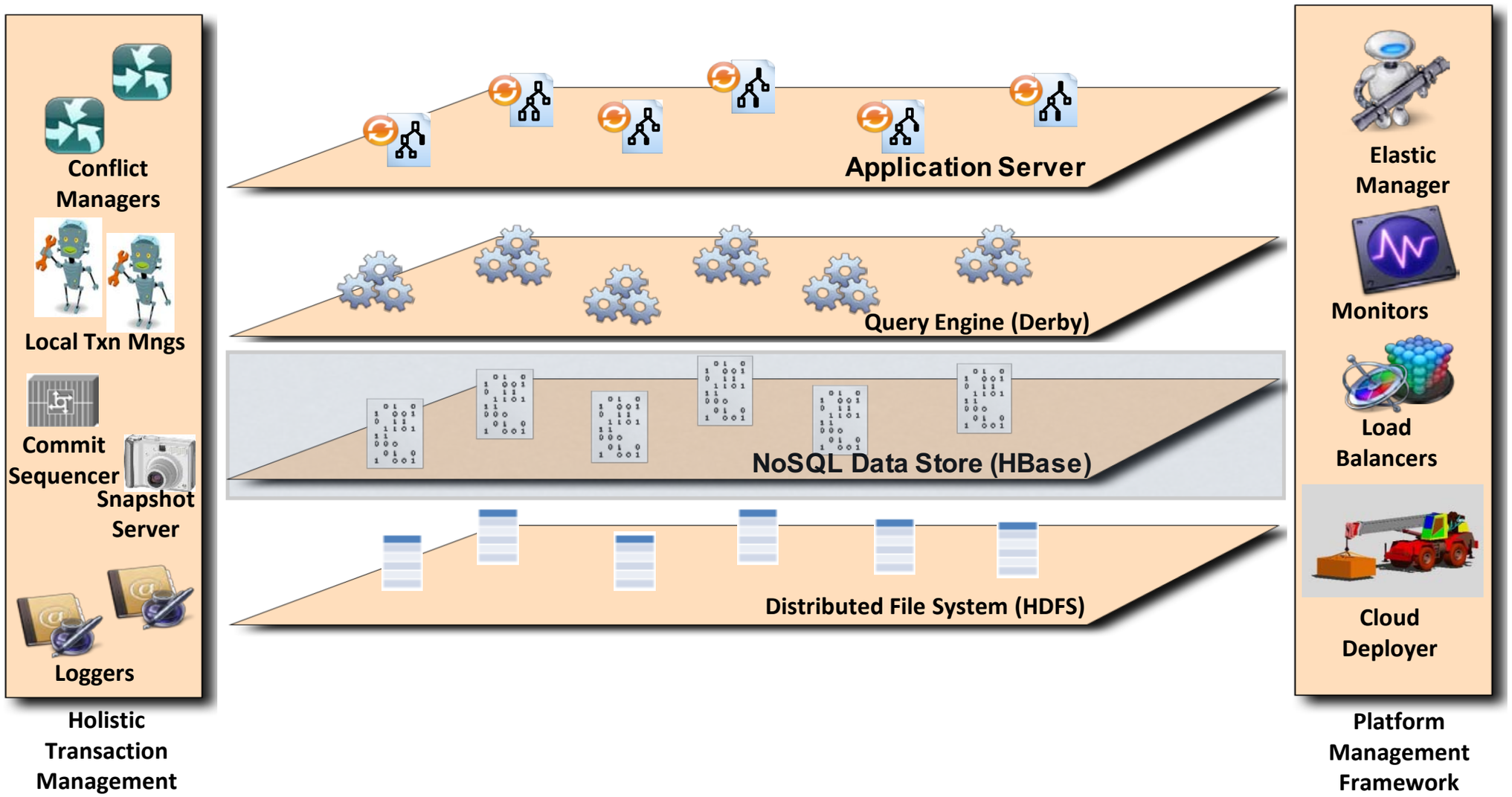
The **CumuloNimbo** system consists of multiple layers each of them having a specific functionality. In order to achieve the desired throughput, each layer can be scaled independently by adding more servers. Instances of different layers might be collocated to improve response time performance.

Architecture



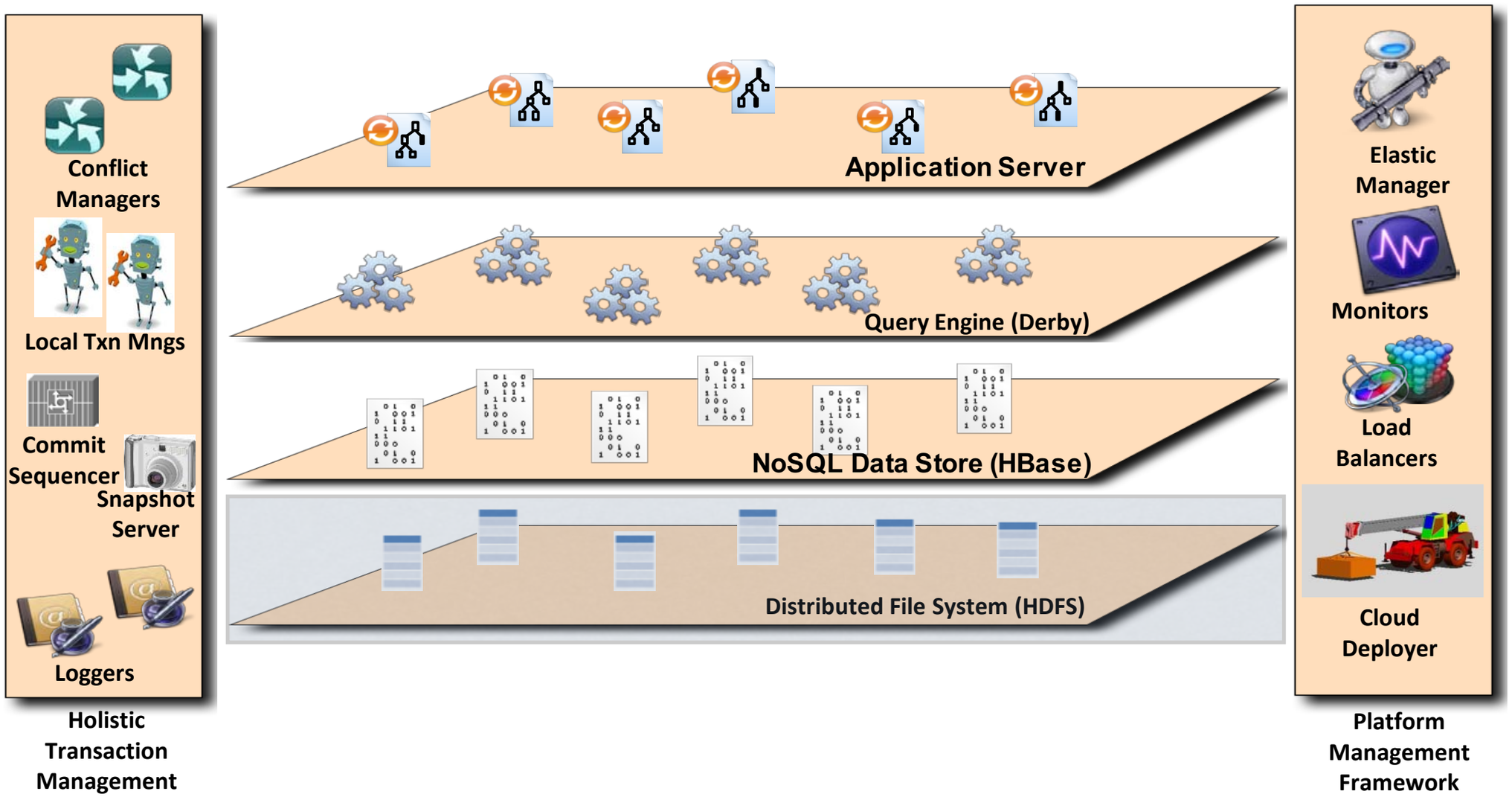
The **Query Engine** layer, based on **Apache Derby** query engine, provides the standard relational interface (SQL) including the transaction commands (commit, abort, and begin). The query layer itself is responsible for planning, optimizing and executing queries. However, it does not perform transaction management nor is it responsible for data storage.

Architecture



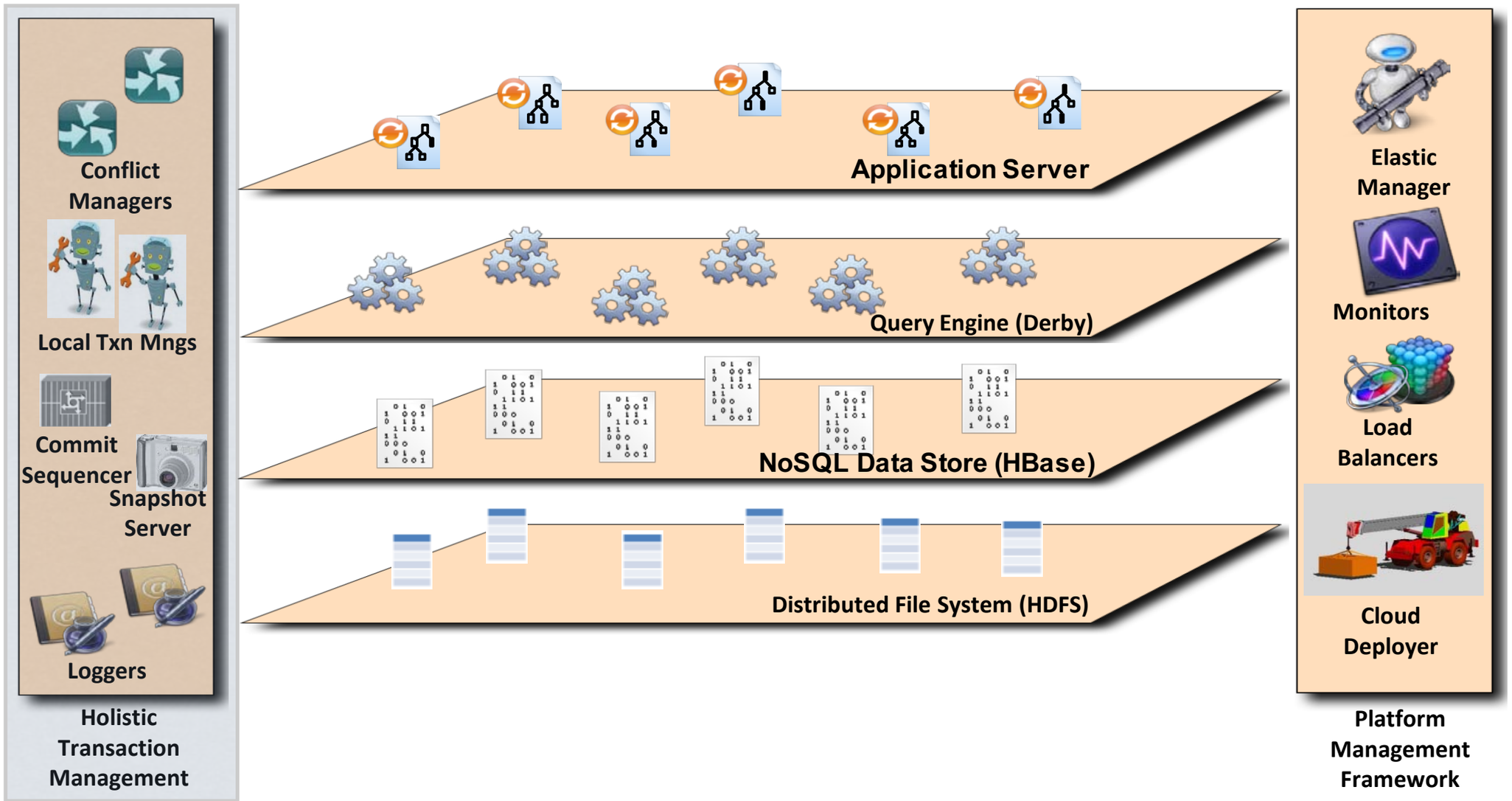
The **NoSQL Data Store** layer uses an advanced data store that is designed for inherent scalability in terms of data size and processing requirements. Our system is based on **HBase**, an elastic key-value store.

Architecture



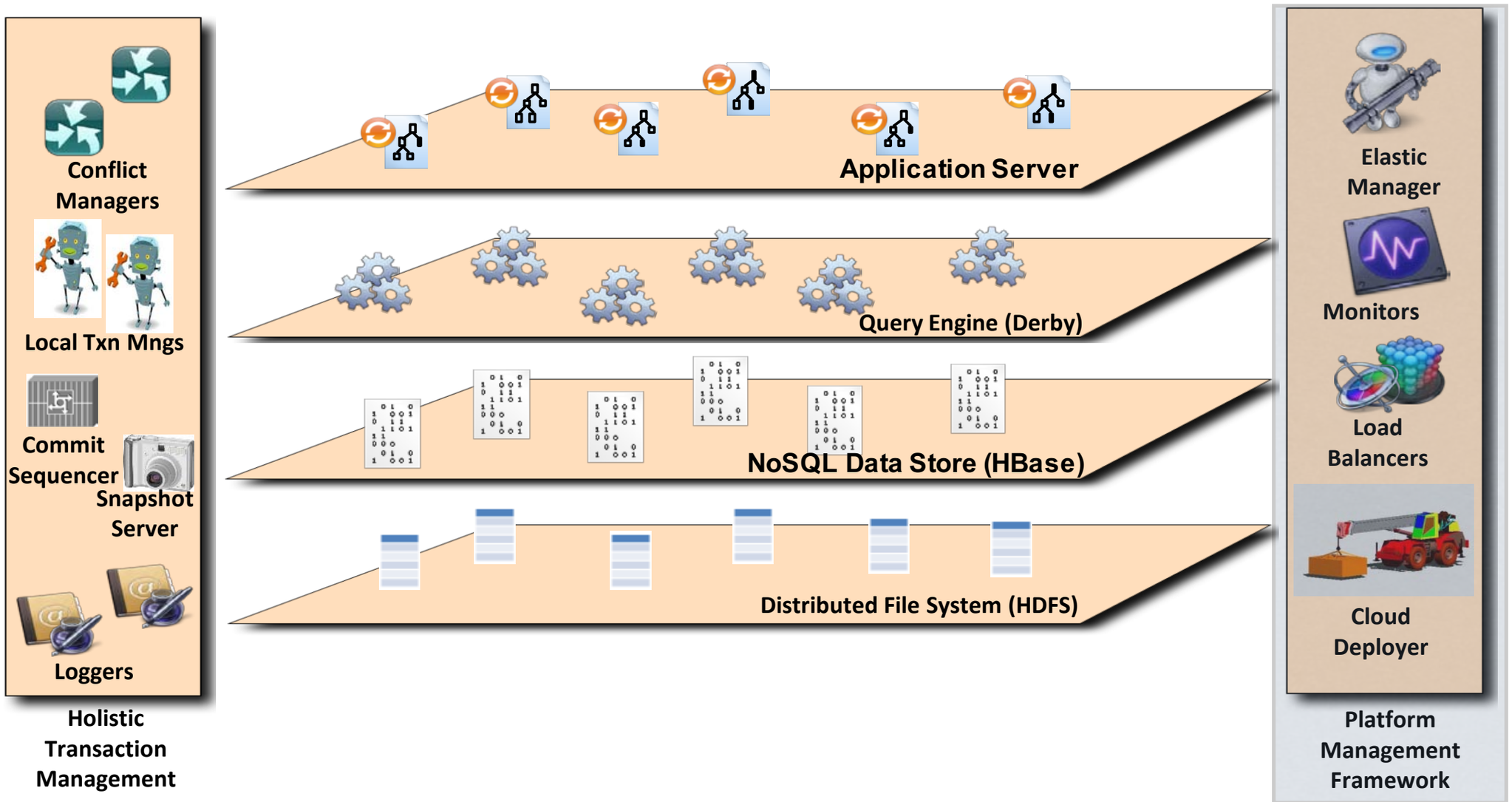
The NoSQL Data Store sits on top of the Hadoop **Distributed File System** that provides persistent and fault-tolerant data storage. The NoSQL Data Store tables, as well as its write-ahead log, are persisted in **HDFS**.

Architecture



The **Transaction Management** is handled in a holistic manner providing full ACID properties across the entire stack.

Architecture



The **Platform Management Framework** takes care of tasks such as deployment, monitoring, dynamic load balancing and elasticity. Each instance on each layer has a monitor collecting data about resource usage and performance metrics that are reported to a central monitor.

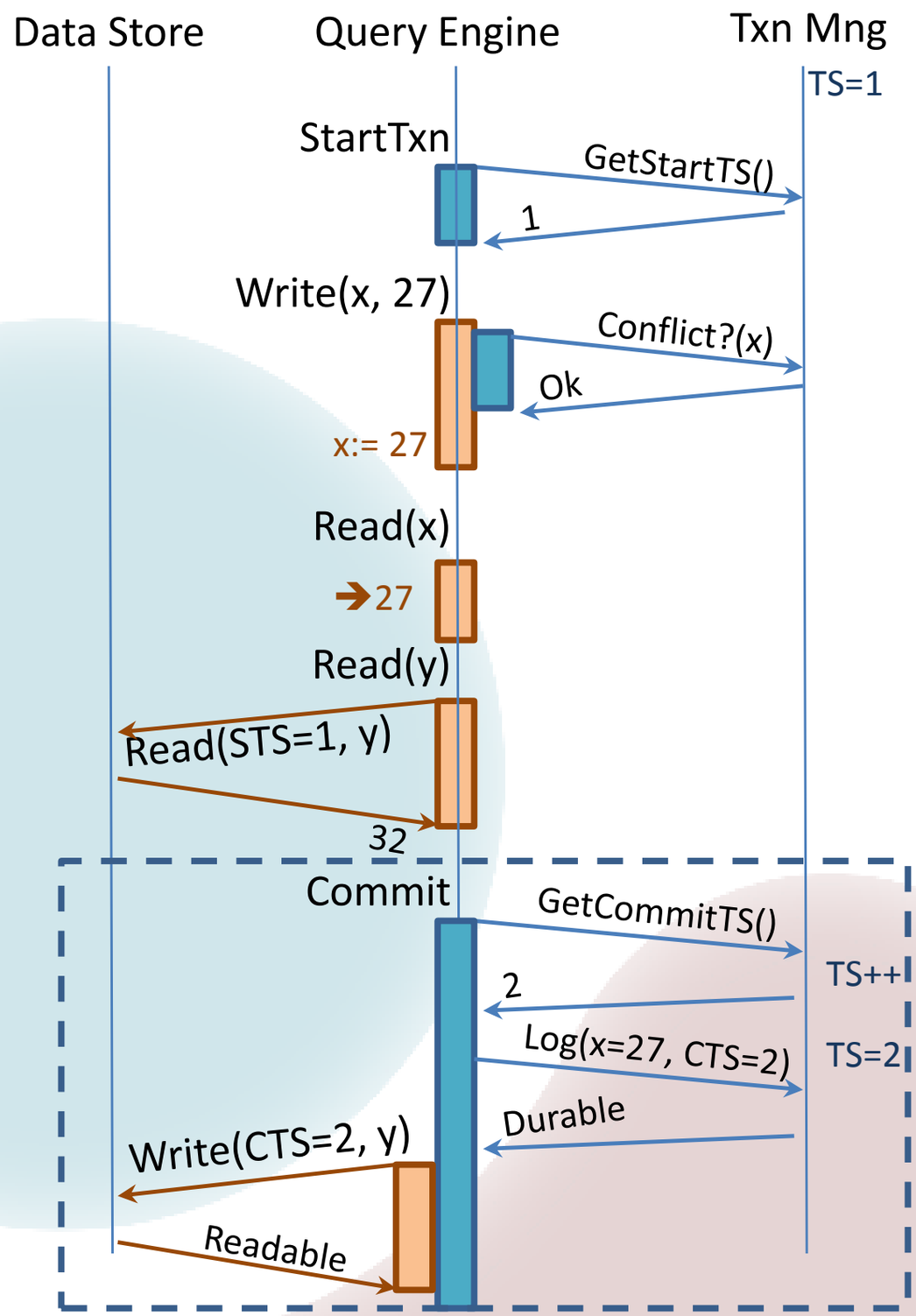
OLTP

Scalable Transaction Processing

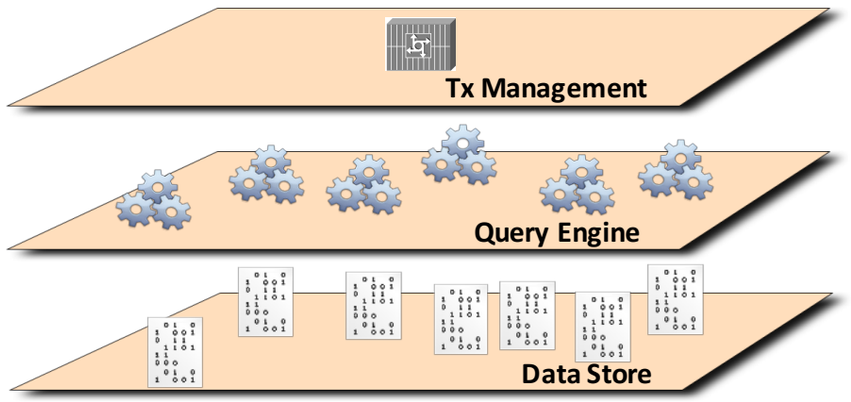
- Snapshot Isolation level
  - Each transaction reads the data from a snapshot that represents all committed values at the start time of a transaction
  - Conflicts are detected on updates: Whenever two transactions are concurrent (neither commits before the other starts), and they want to write a common data item, one of them has to abort
- Snapshot isolation requires a multi-version system providing snapshot read and snapshot write properties

Scalable Transaction Processing

Transaction execution under snapshot isolation in a centralized system

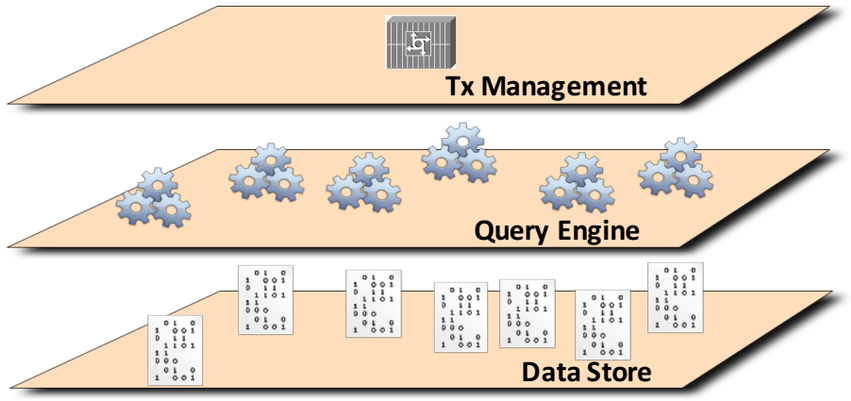


Scalable Transaction Processing



- Naive scale up approach
  - Limitation 1: Serial and Atomic Commit Processing
    Making commit processing appear as a single atomic action that is spread across multiple nodes becomes extremely expensive in terms of the time it takes to complete the action, as well as the negative impact it has on concurrency

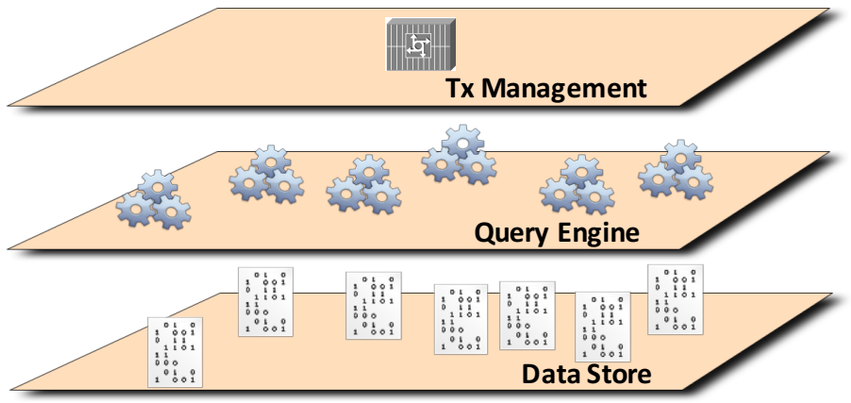
Scalable Transaction Processing



- Naive scale up approach
  - Limitation 2: Monolithic Transactional Processing

The global transaction management component can easily become a bottleneck, performing many different tasks and connecting to a large number of other components.

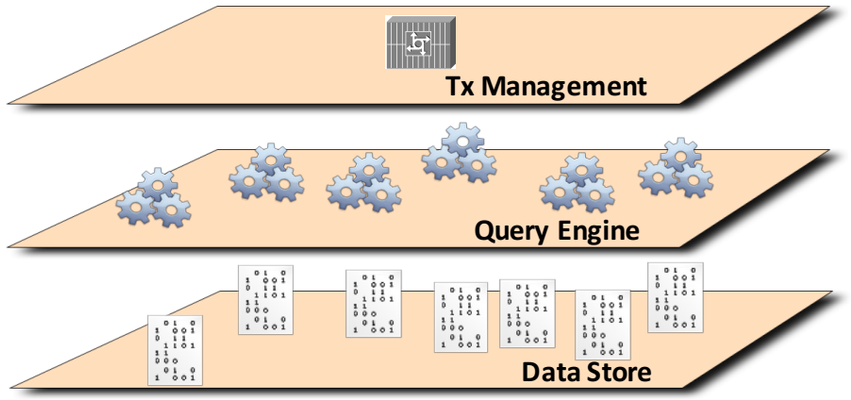
Scalable Transaction Processing



- Naive scale up approach
  - Limitation 3: Synchronous Communication

    Transaction execution is prolonged by including many synchronous message exchanges. For each write operation, the transaction first checks for conflicts at the transaction manager, and then writes the changes to the data store. Furthermore, the start and commit of the transaction require additional message rounds with the transaction manager and/or the data store.

Scalable Transaction Processing

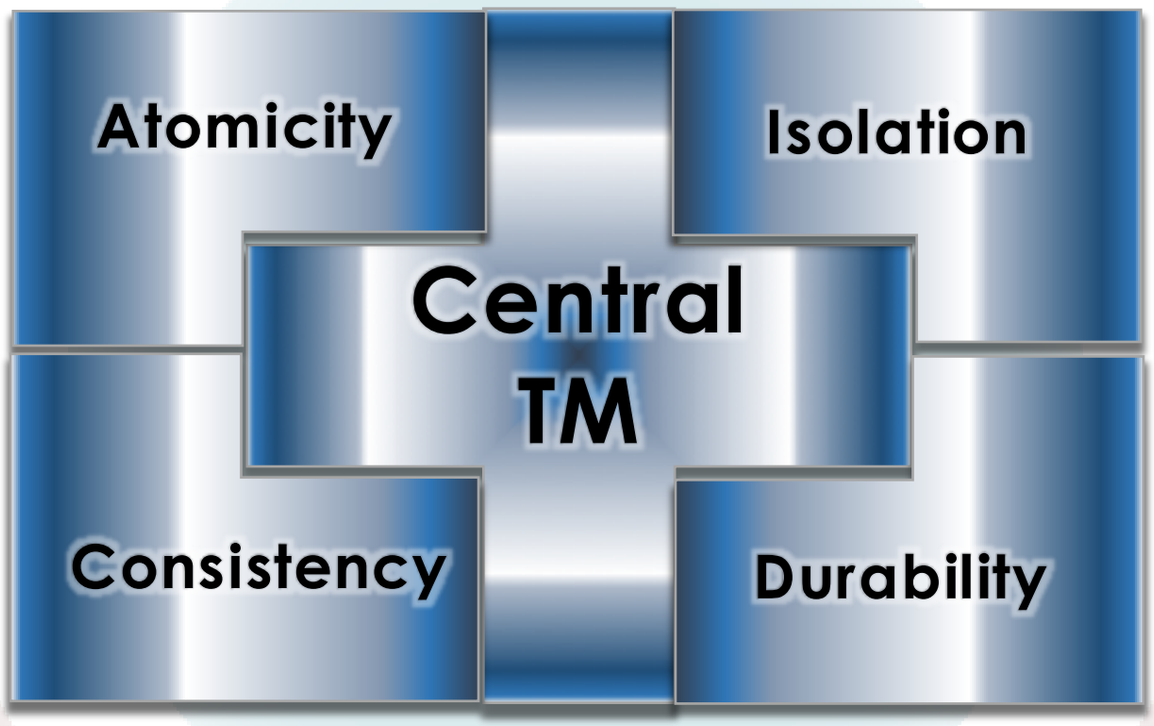


- Naive scale up approach
  - Limitation 4: **Message Overhead**

At large transaction rates all components have to handle a **large amount of messages** which can quickly become a major **overhead**, for instance, a round-trip message per updated item.

The CumuloNimbo Breakthrough

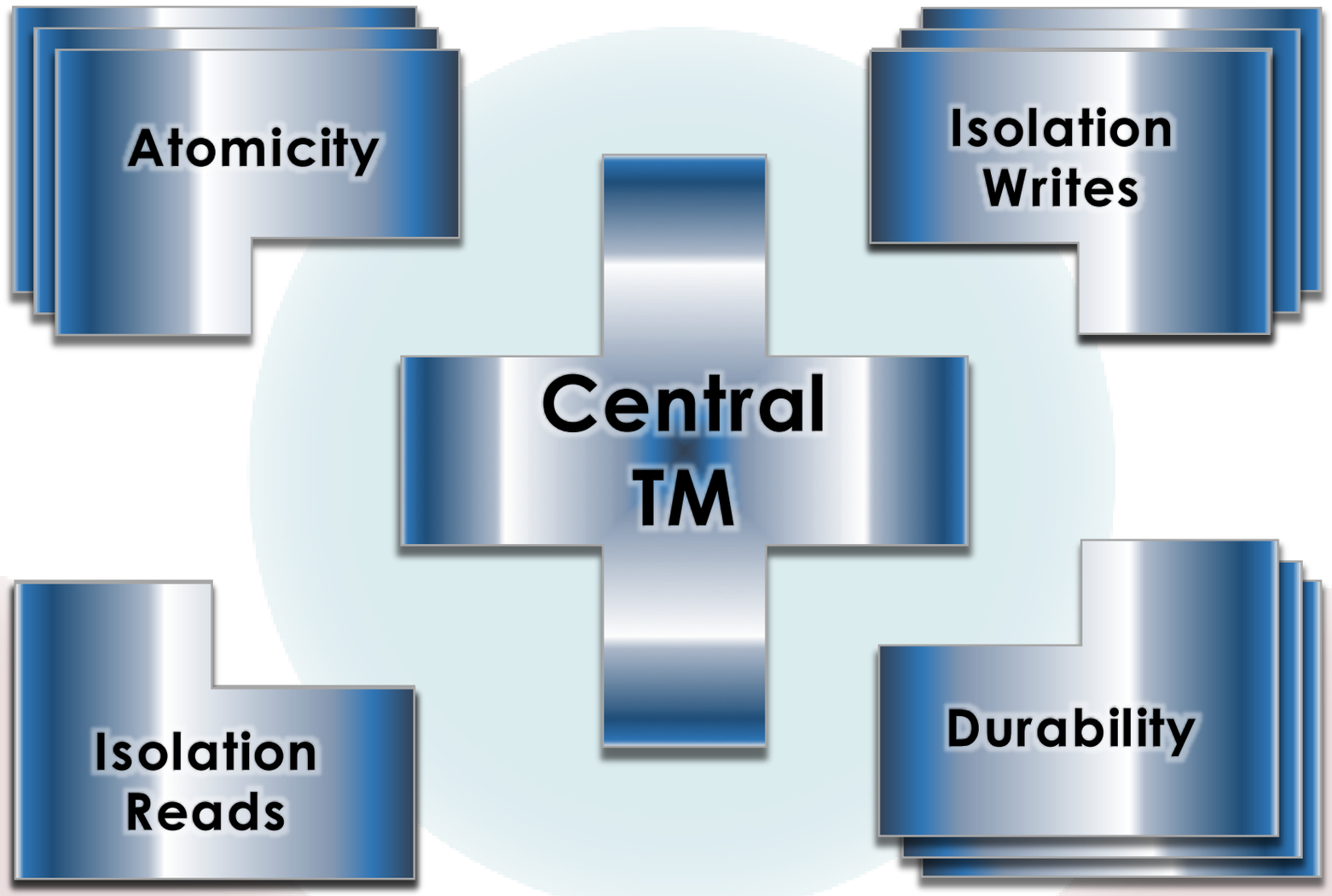
Untangle and scale out the ACID properties independently



HASLab
HIGH-ASSURANCE
SOFTWARE LABORATORY

The CumuloNimbo Breakthrough

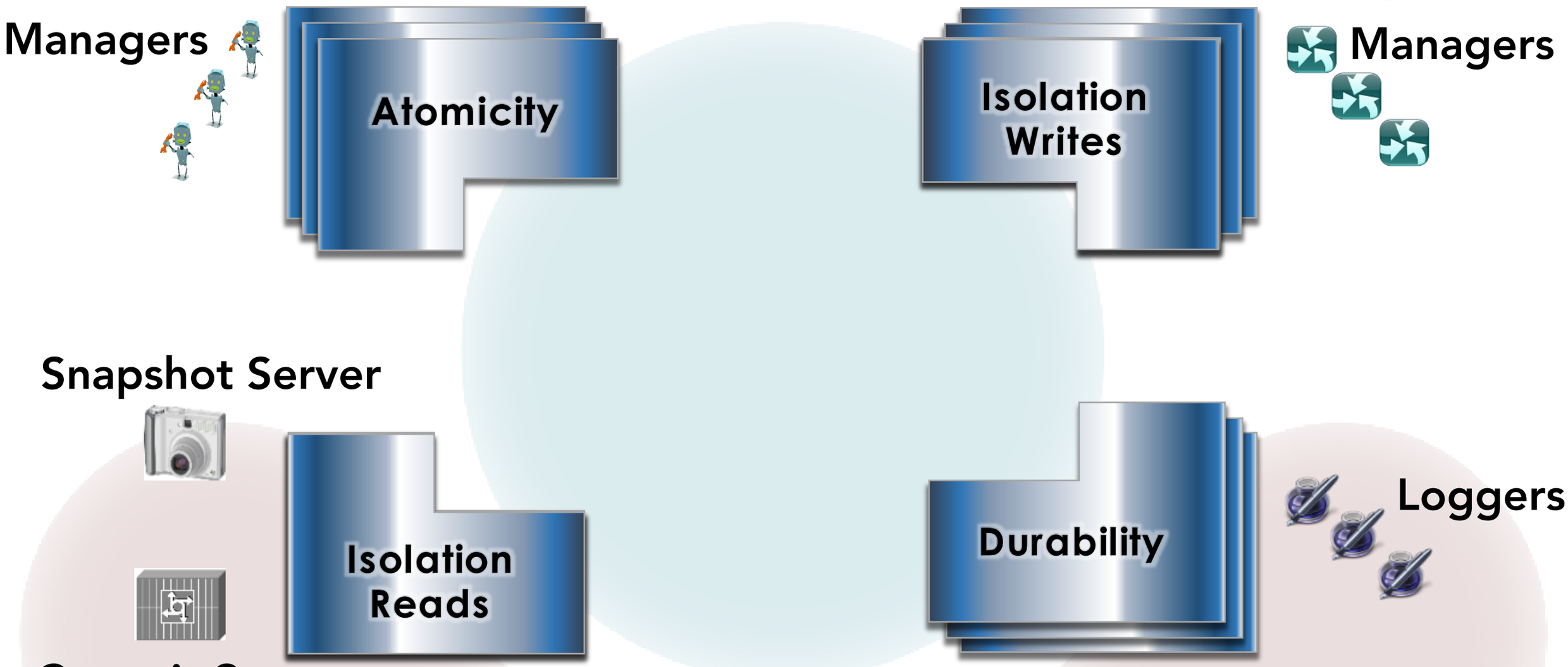
Untangle and scale out the ACID properties independently



HASLab
HIGH-ASSURANCE SOFTWARE LABORATORY

The CumuloNimbo Breakthrough

Untangle and scale out the ACID properties independently



HASLab
HIGH-ASSURANCE SOFTWARE LABORATORY

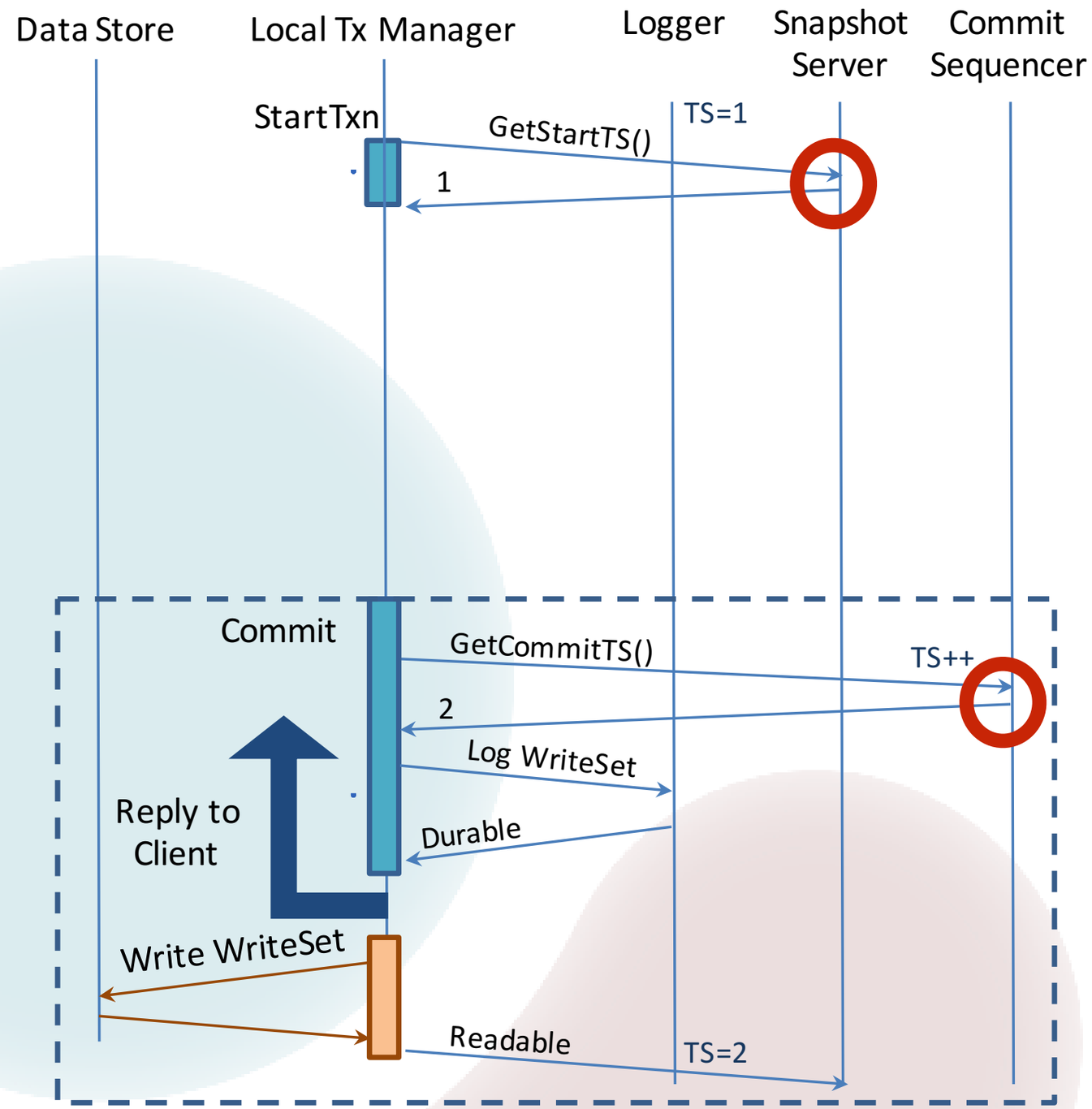
Principles of the Approach

- Decoupling Update Visibility and Atomic Commit
- Proactive Timestamp Management
- Parallelization and Distribution
- Asynchronous Messaging and Batching

Decoupling Update Visibility and Atomic Commit

Distinguish Snapshot Timestamp from Commit Timestamp.

Updates to the Data Store are now outside of the response path.



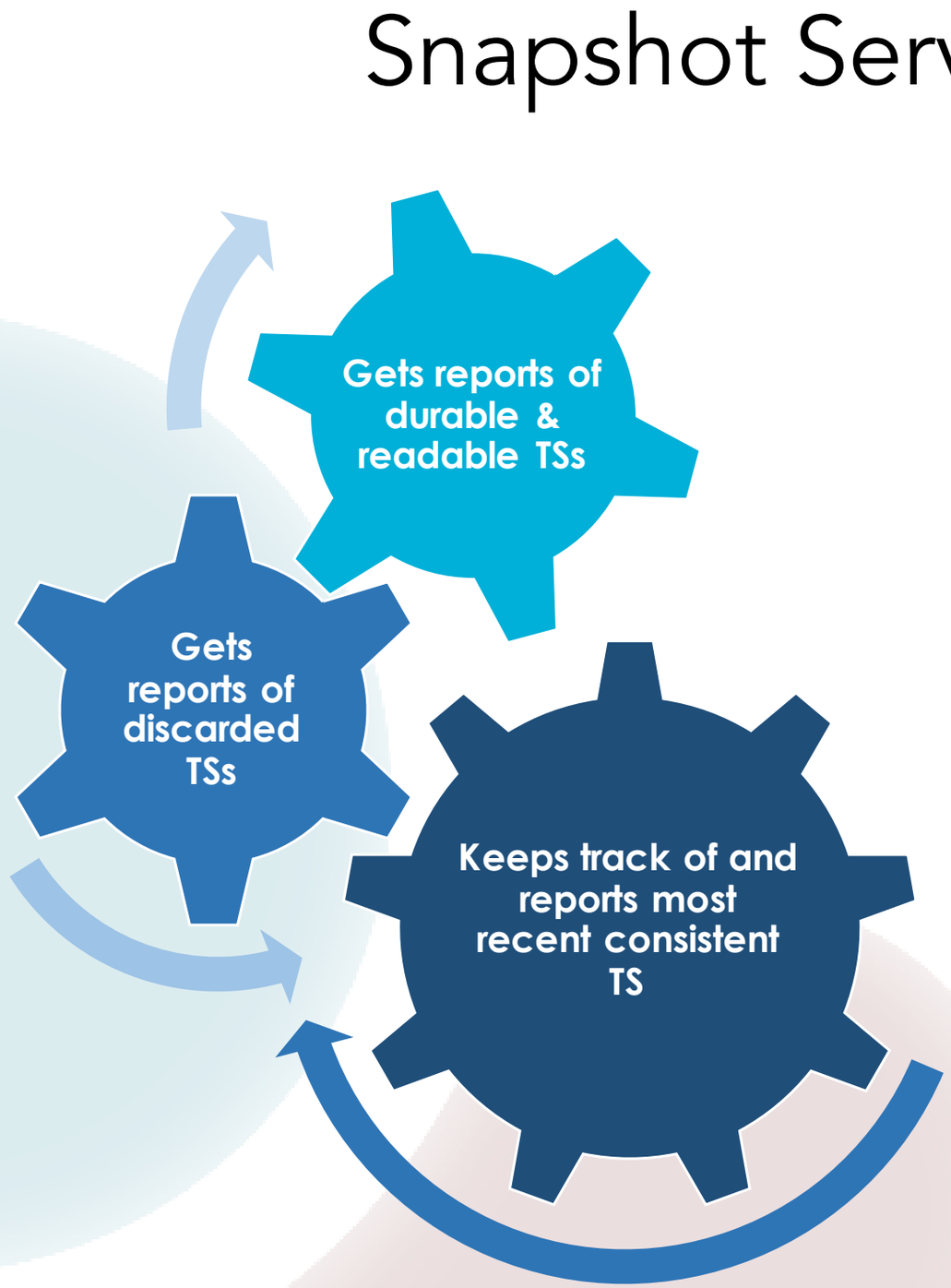
HASLab
HIGH-ASSURANCE SOFTWARE LABORATORY

Decoupling Update Visibility and Atomic Commit

The Snapshot server keeps track of the most recent snapshot that is consistent.

Its TS should be such that there is no previous Commit TS that a) is not yet readable or b) has been discarded.

This way, **transactions can commit in parallel** and consistency be preserved.



Decoupling Update Visibility and Atomic Commit

Coordination between Snapshot and Commit servers

Sequence of timestamps received by the Snapshot Server

Evolution of the current snapshot at the Snapshot Server

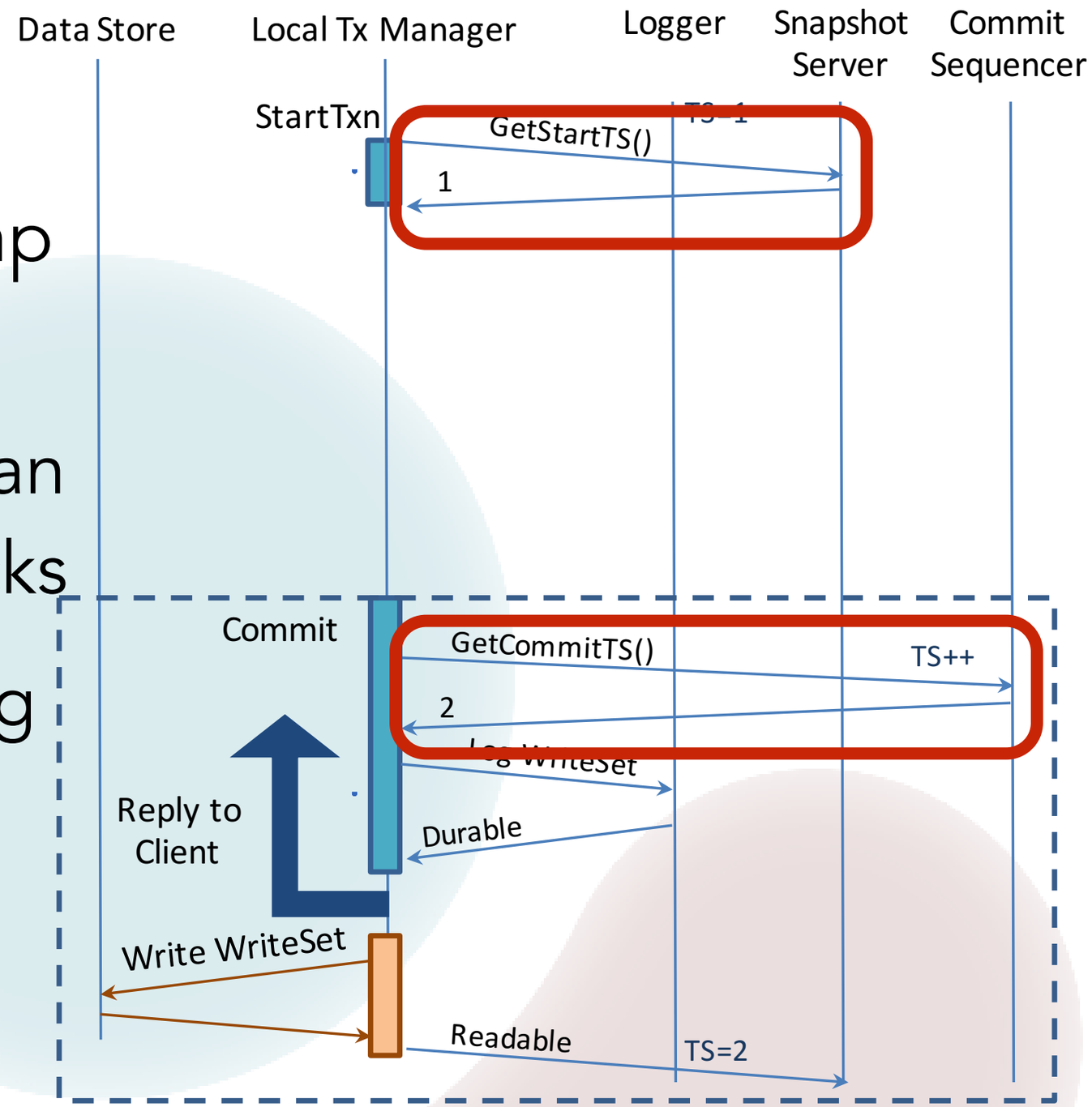
HASLab
HIGH-ASSURANCE SOFTWARE LABORATORY

Proactive Timestamp Management

Two independent timestamp services

Timestamp management can become ultimate bottlenecks

Proactive timestamp serving to the rescue



HASLab
HIGH-ASSURANCE SOFTWARE LABORATORY

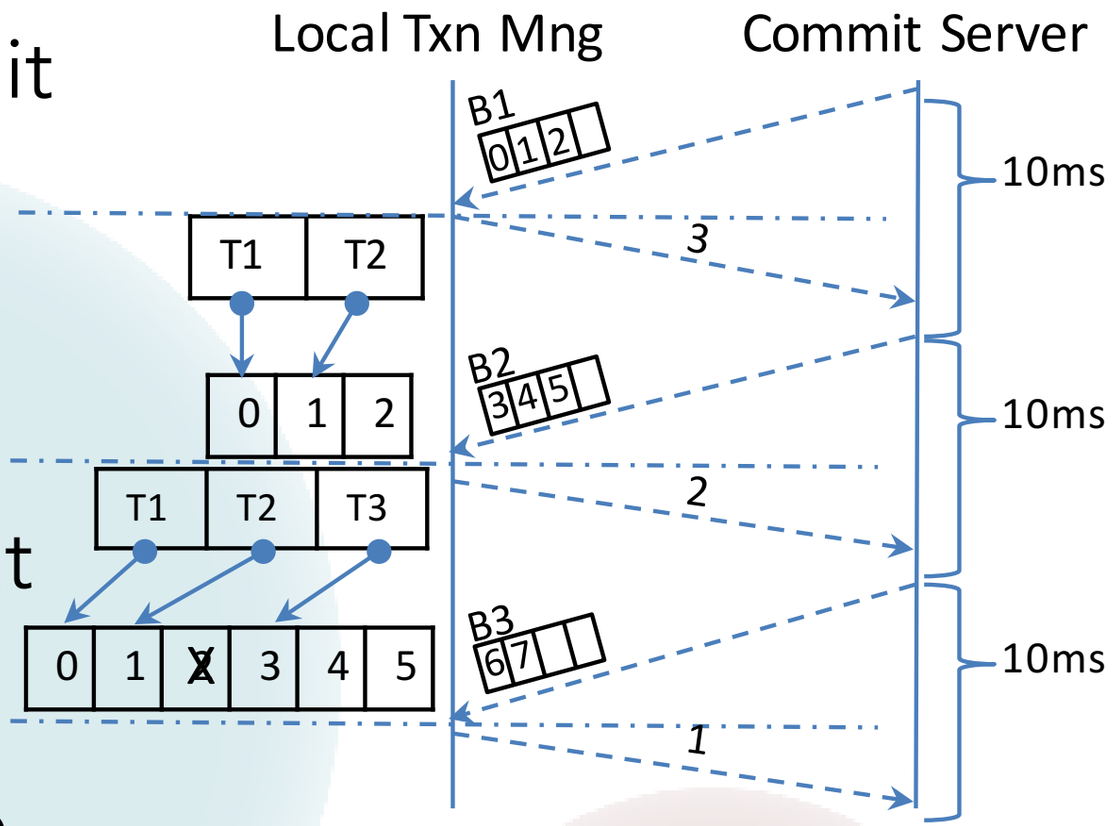
Proactive Timestamp Management

Commit Sequencer

Once a transaction enters the commit phase it is guaranteed that it does not conflict with any other transaction

The commit timestamp is only needed to tag the updates with that commit timestamp

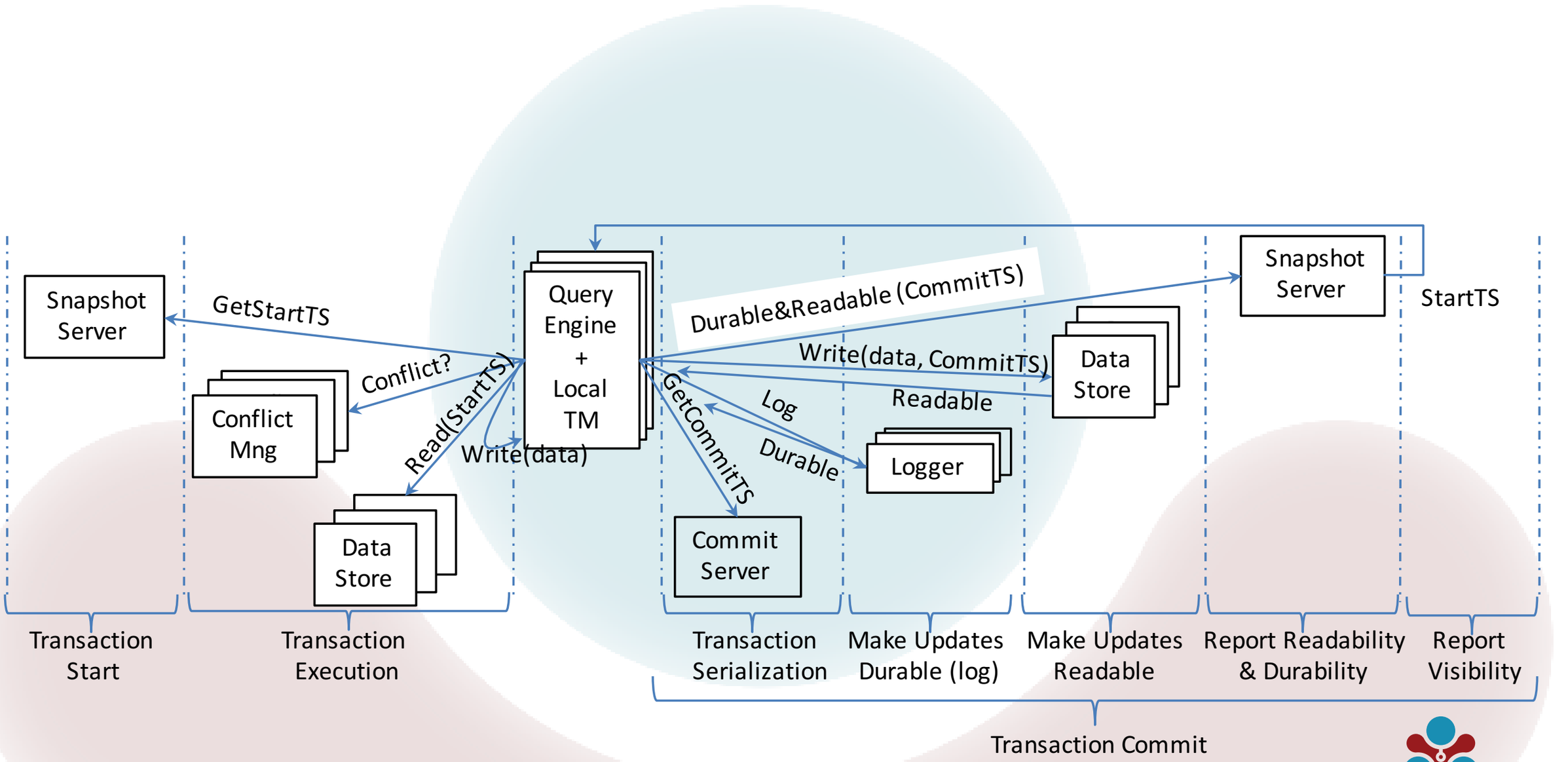
It does not matter which timestamp each transaction gets!



HASLab
HIGH-ASSURANCE SOFTWARE LABORATORY

Parallelization and Distribution

All CumuloNimbo architecture is distributed and most components can be run in parallel

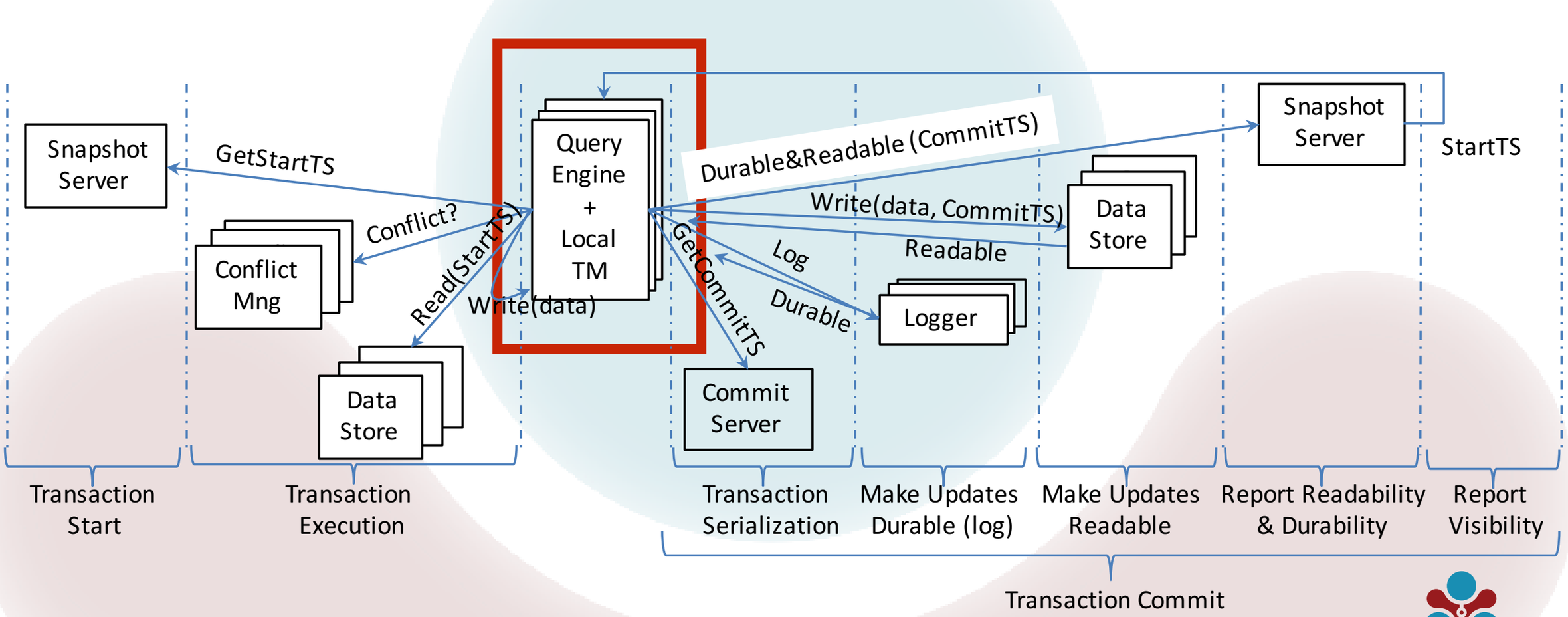


HASLab
HIGH-ASSURANCE SOFTWARE LABORATORY

Parallelization and Distribution

Local Transaction Managers

- Manage the transactions life cycle
- Can have parallel instances
- Collocated with Query Engines



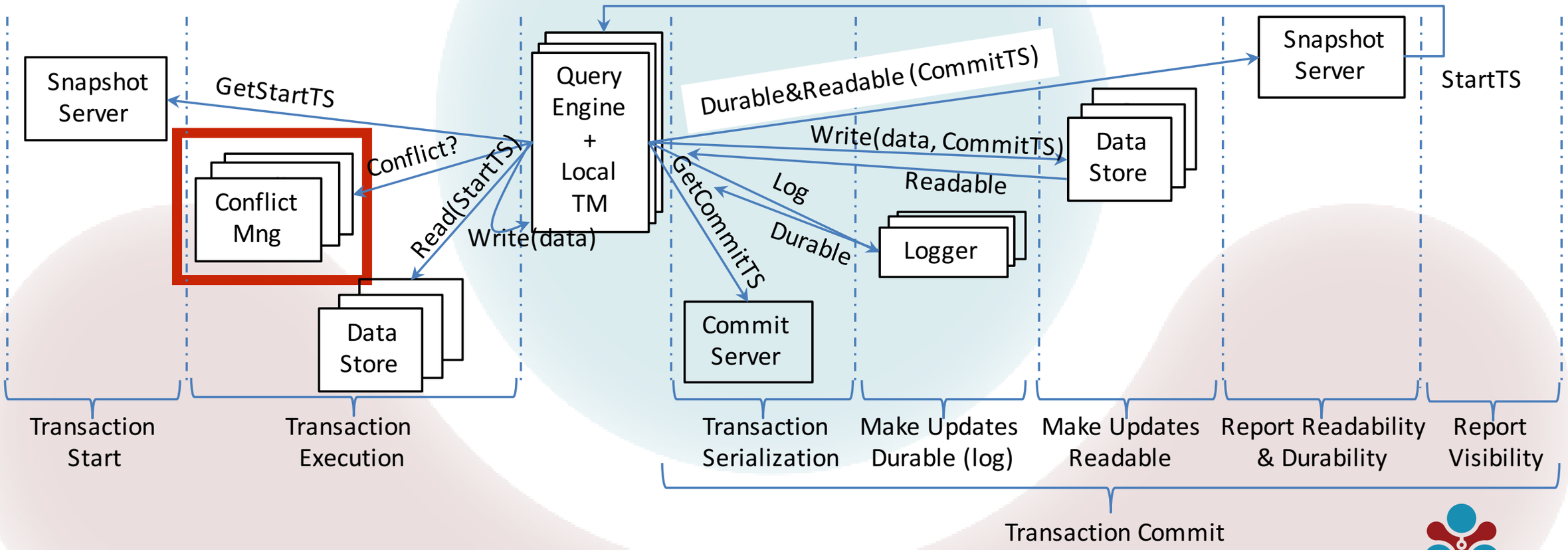
HASLab
HIGH-ASSURANCE SOFTWARE LABORATORY

Parallelization and Distribution

Conflict Managers

Each CM handles a disjoint subset of data record keys

Can have parallel instances

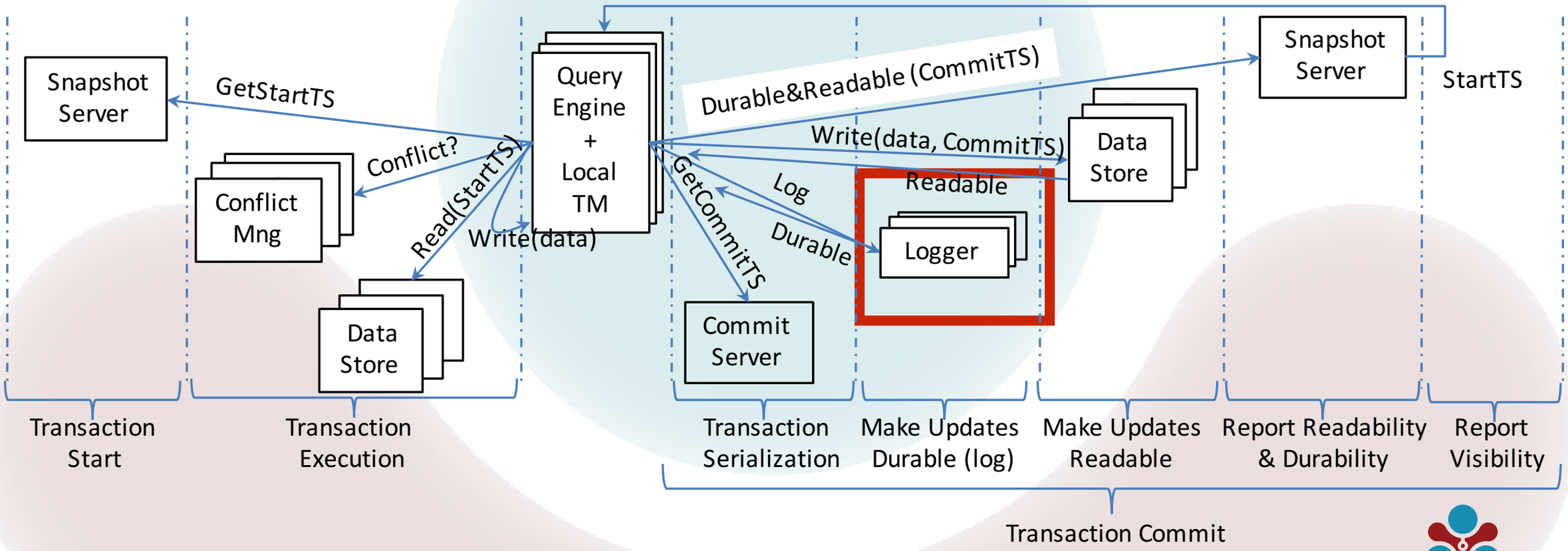


HASLab
HIGH-ASSURANCE SOFTWARE LABORATORY

Parallelization and Distribution

Loggers

- Each Logger handles a fraction of the log records
- Can have parallel and replicated instances
- Independent; do not coordinate with each other



HASLab
HIGH-ASSURANCE SOFTWARE LABORATORY

Asynchronous Messaging and Batching

- Conflict Managers
  - Conflict checks are asynchronous: non-conflicting transactions run faster vs. abort detection is delayed
  - Conflict checks are batched
    - On commit, outstanding conflict checks are synchronized
- Update propagation
  - While read operations are synchronous, updates are made asynchronously and batched per data store

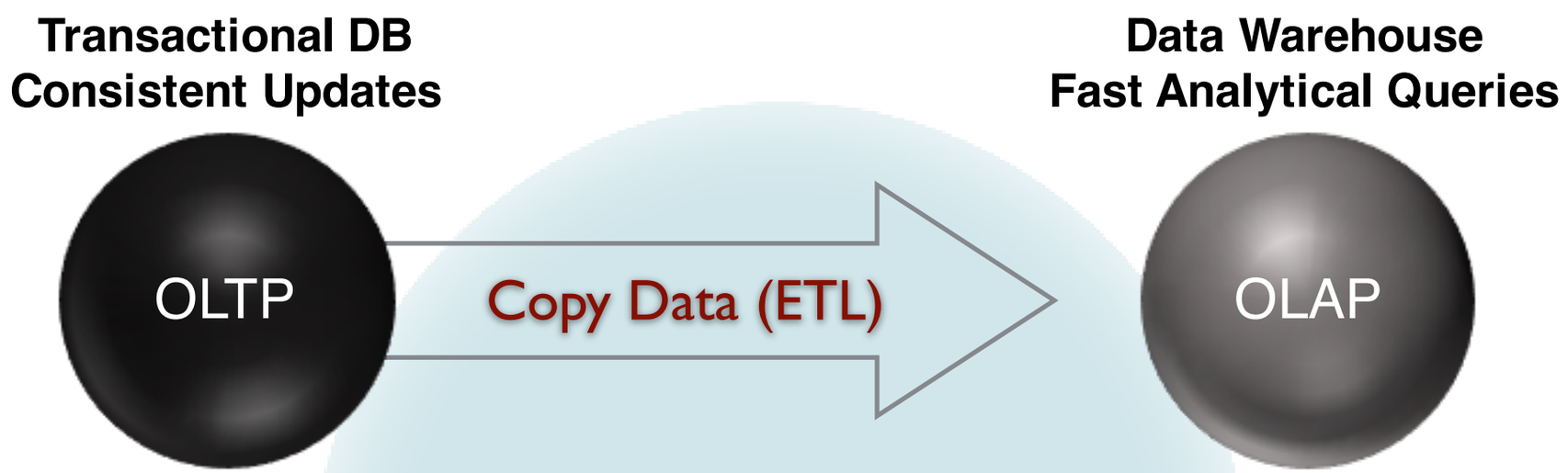
Asynchronous Messaging and Batching

- The local transaction managers report periodically about the number of committed update transactions per second
- The commit sequencer distributes batches of commit timestamps to the local transaction managers
- The snapshot server gets periodically batches of timestamps (both used and discarded) from local transaction managers
- The snapshot server reports periodically to local transaction managers the most current consistent snapshot

OLAP

Scalable Analytics

Current Data Warehousing solutions rely in ETLs to update data.



Cost of ETLs 80% of Business Analytics !

The Big Data challenge to current clustered databases is to allow analytical processing over the freshest data.

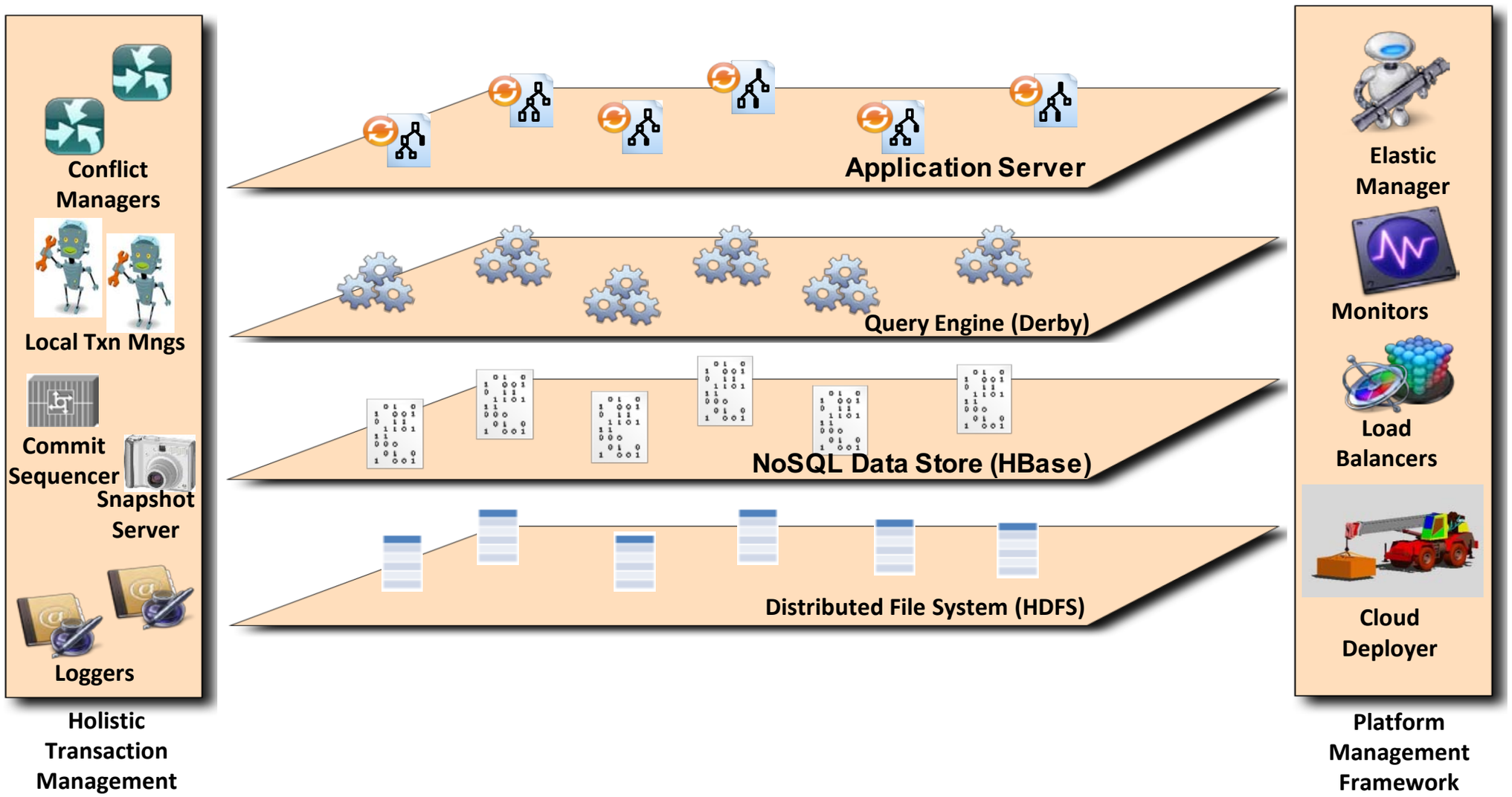
Scalable Analytics

Analytical queries over the transactional DB:

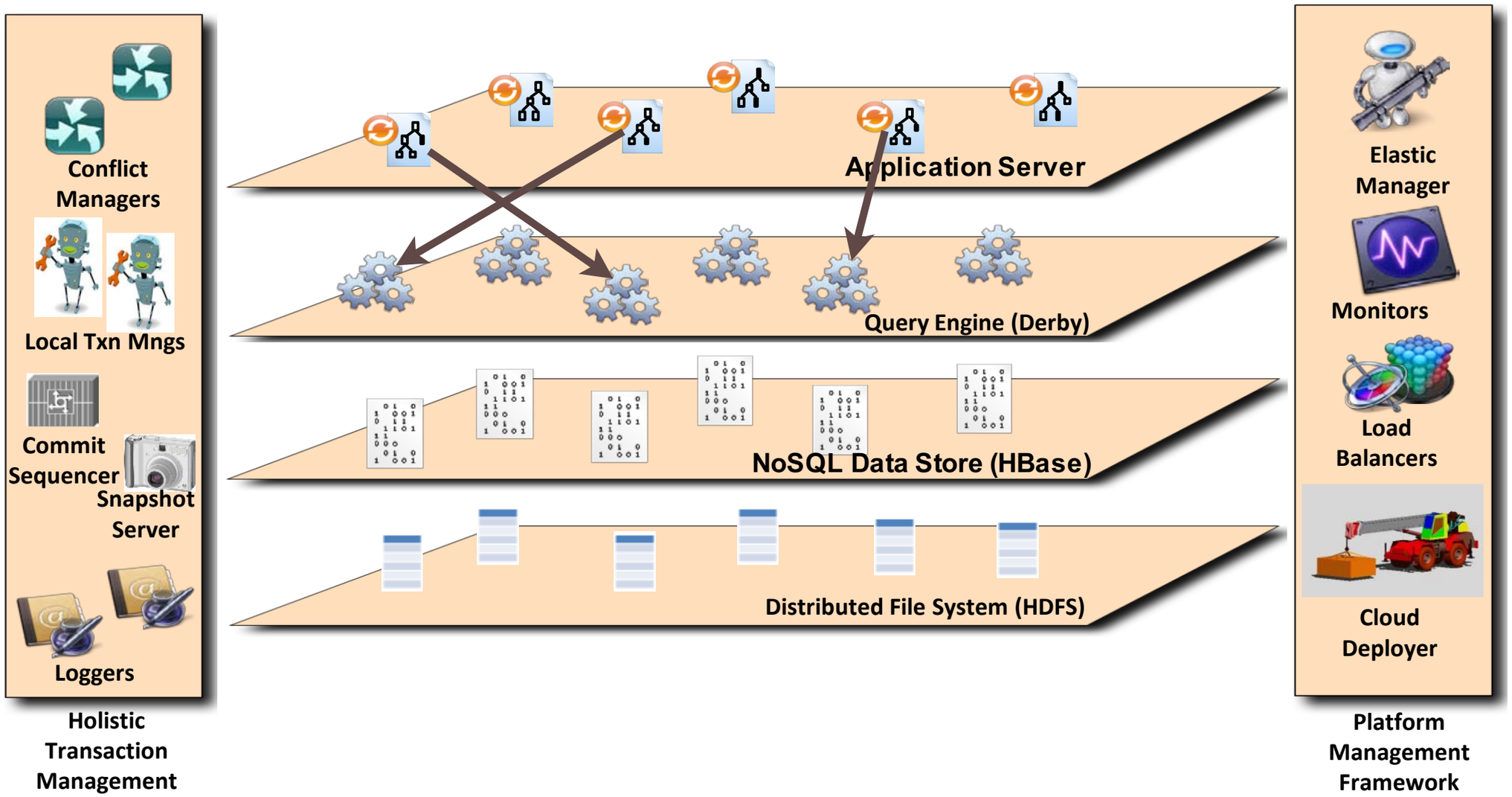
Real-Time Analytics (No ETLs)

HTAP -80% cost in Business Analytics !

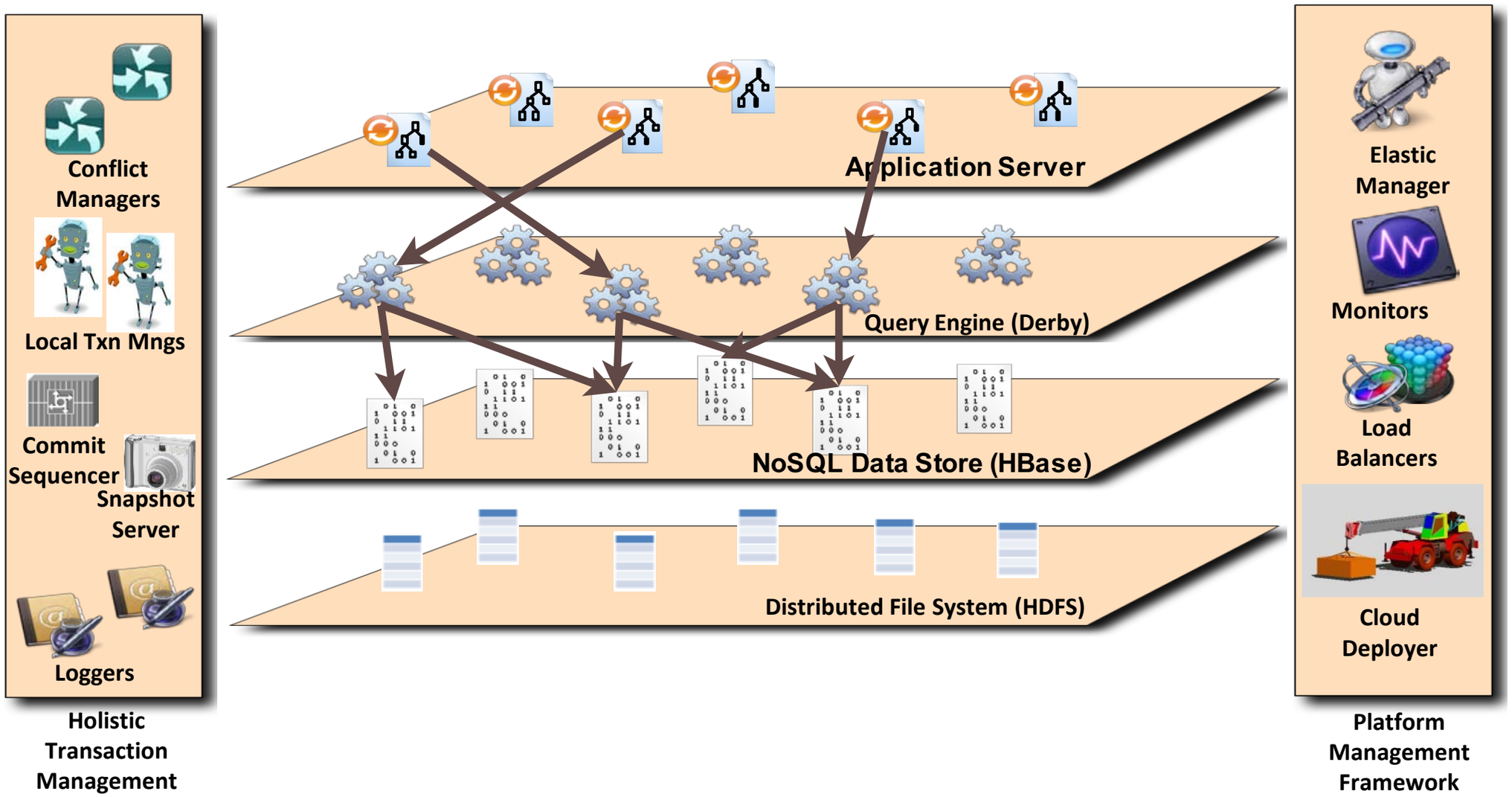
OLTP Architecture



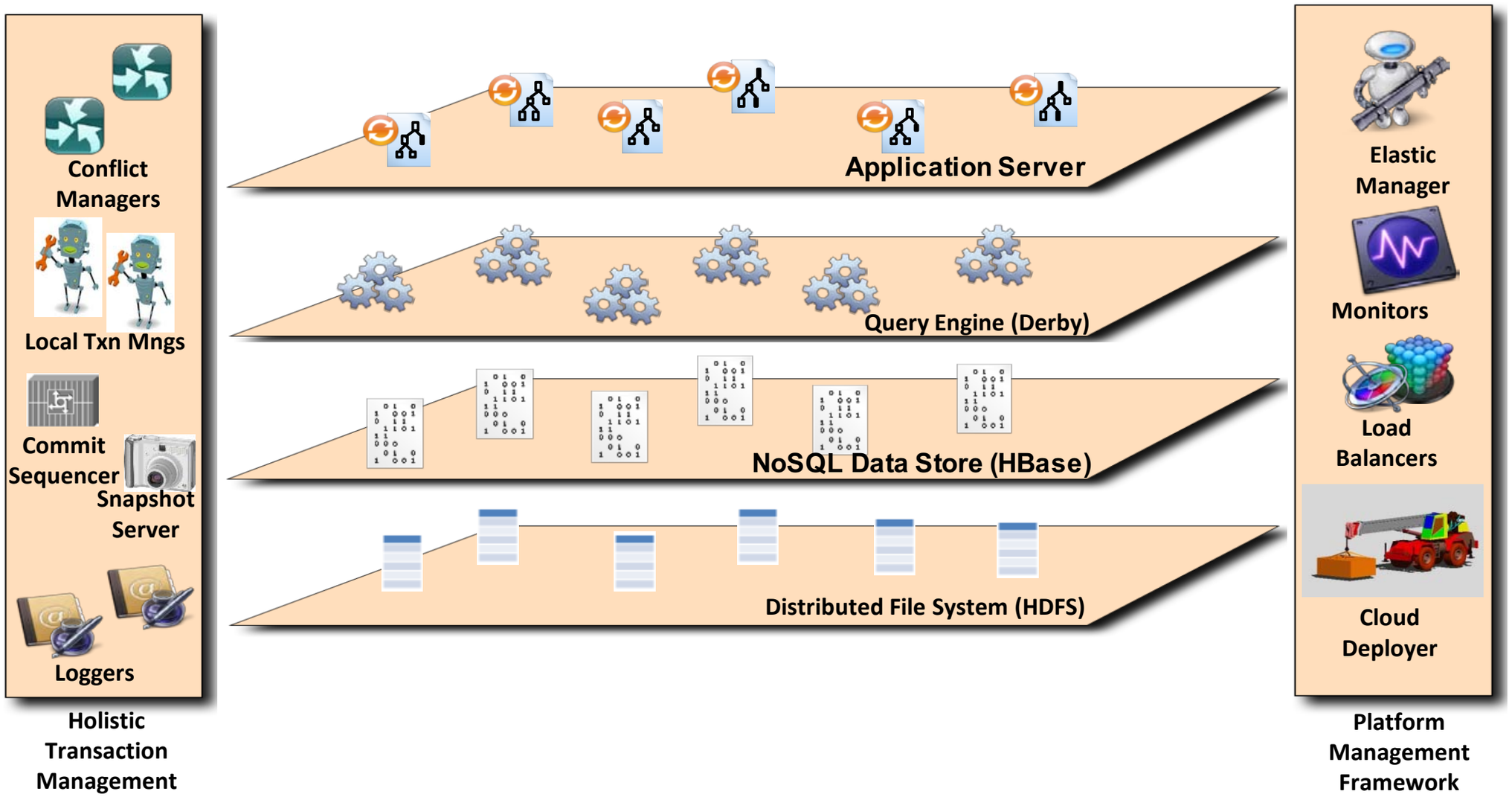
OLTP Architecture



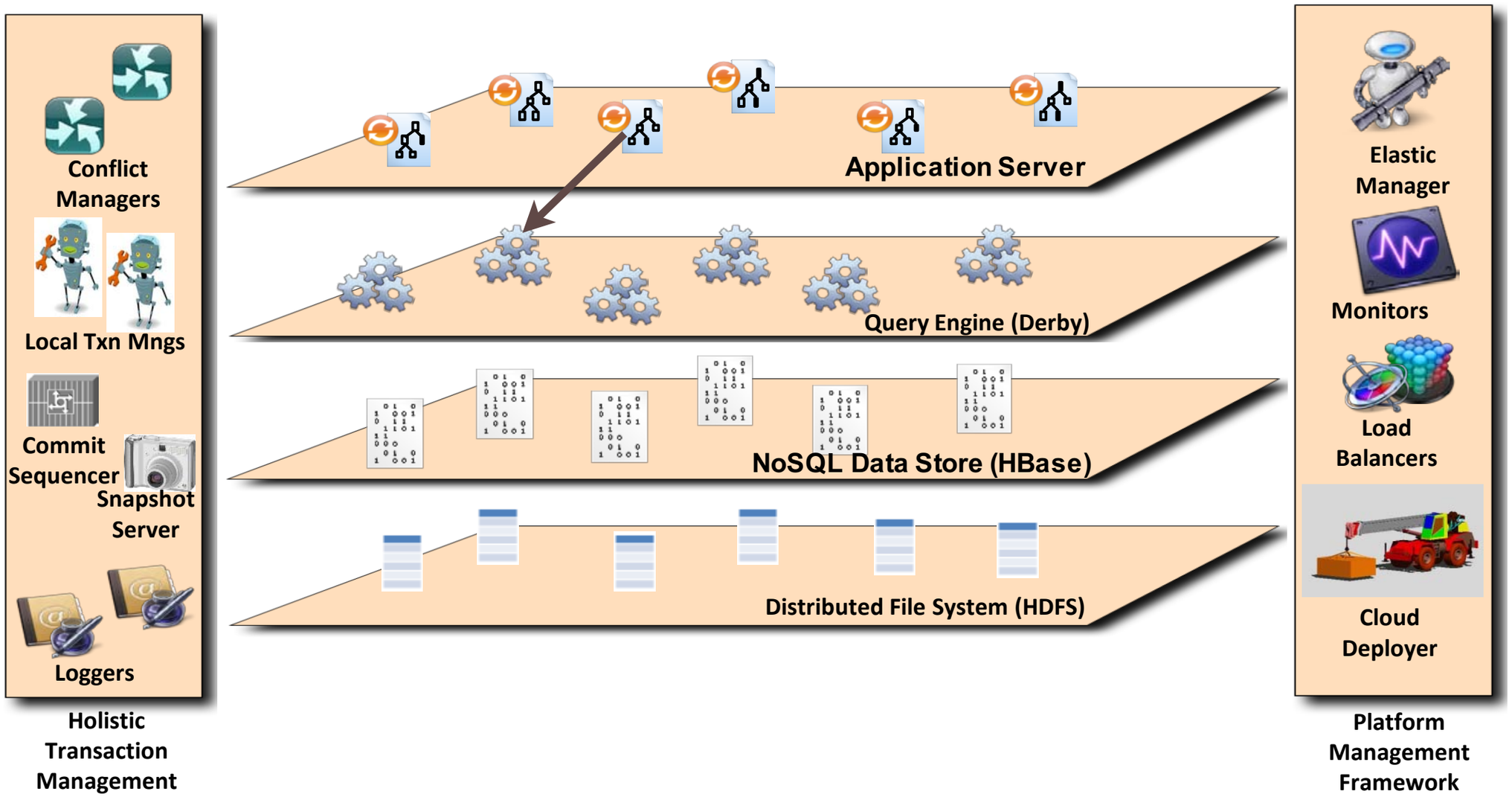
OLTP Architecture



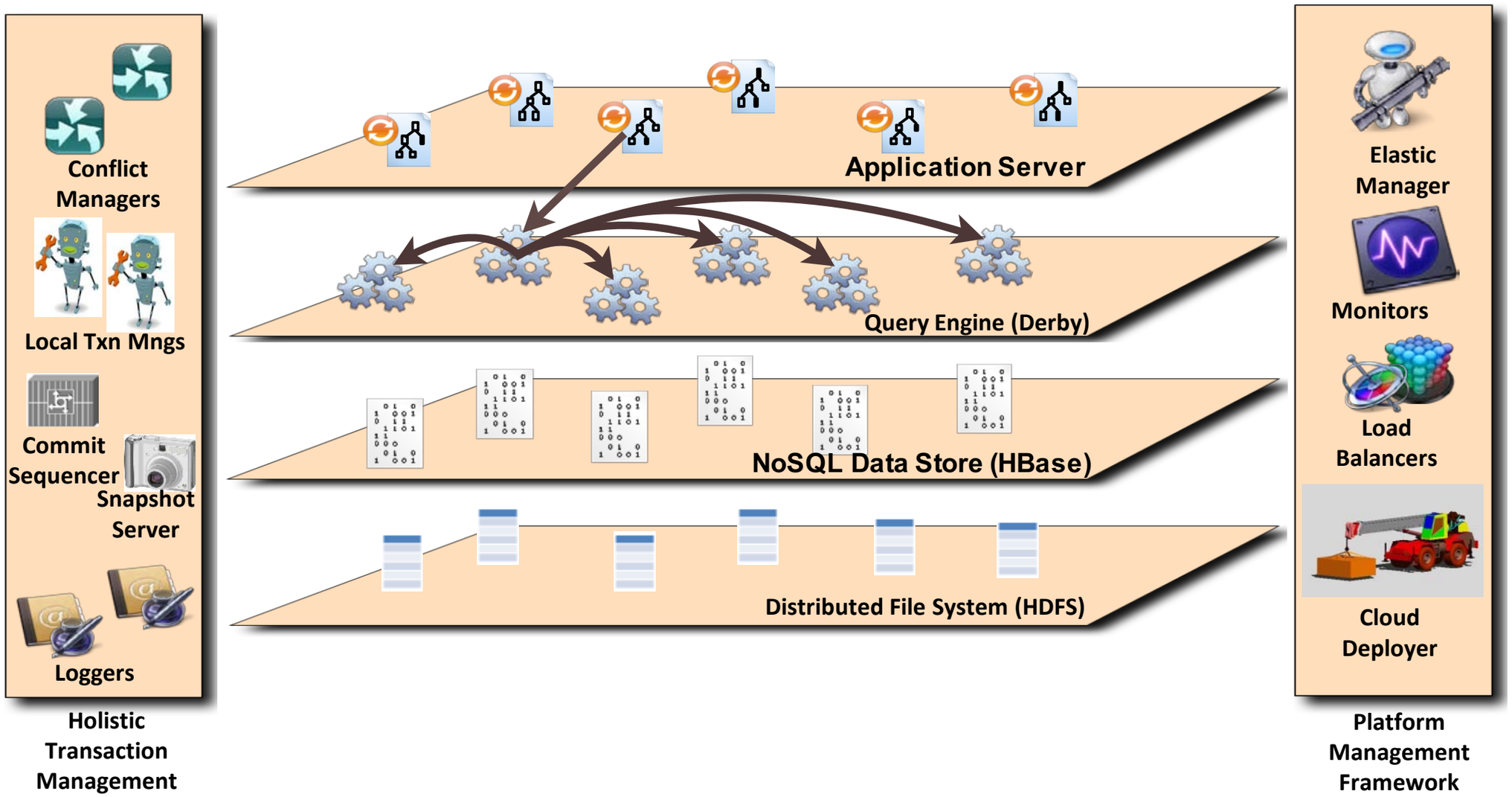
OLAP Architecture



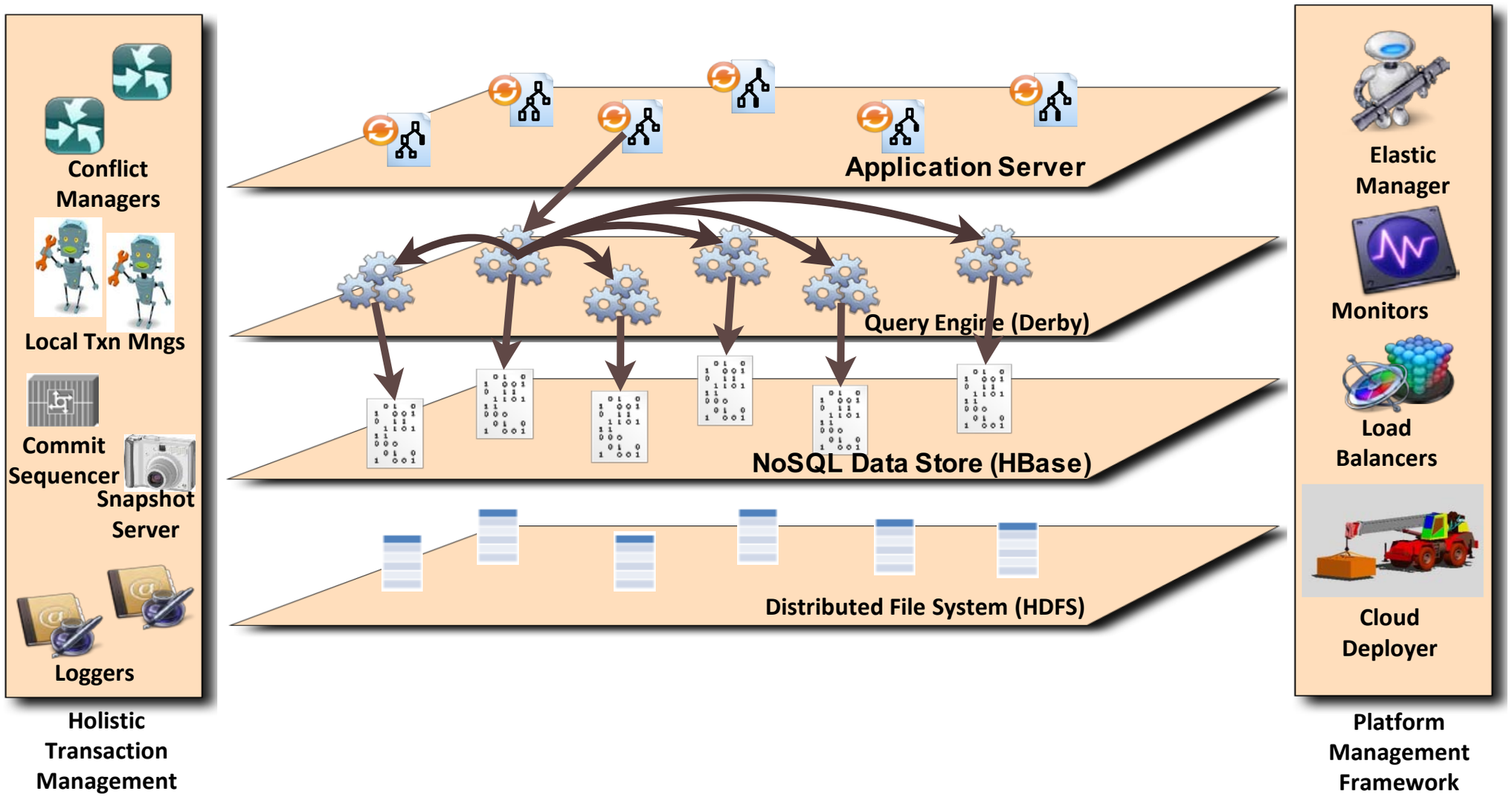
OLAP Architecture



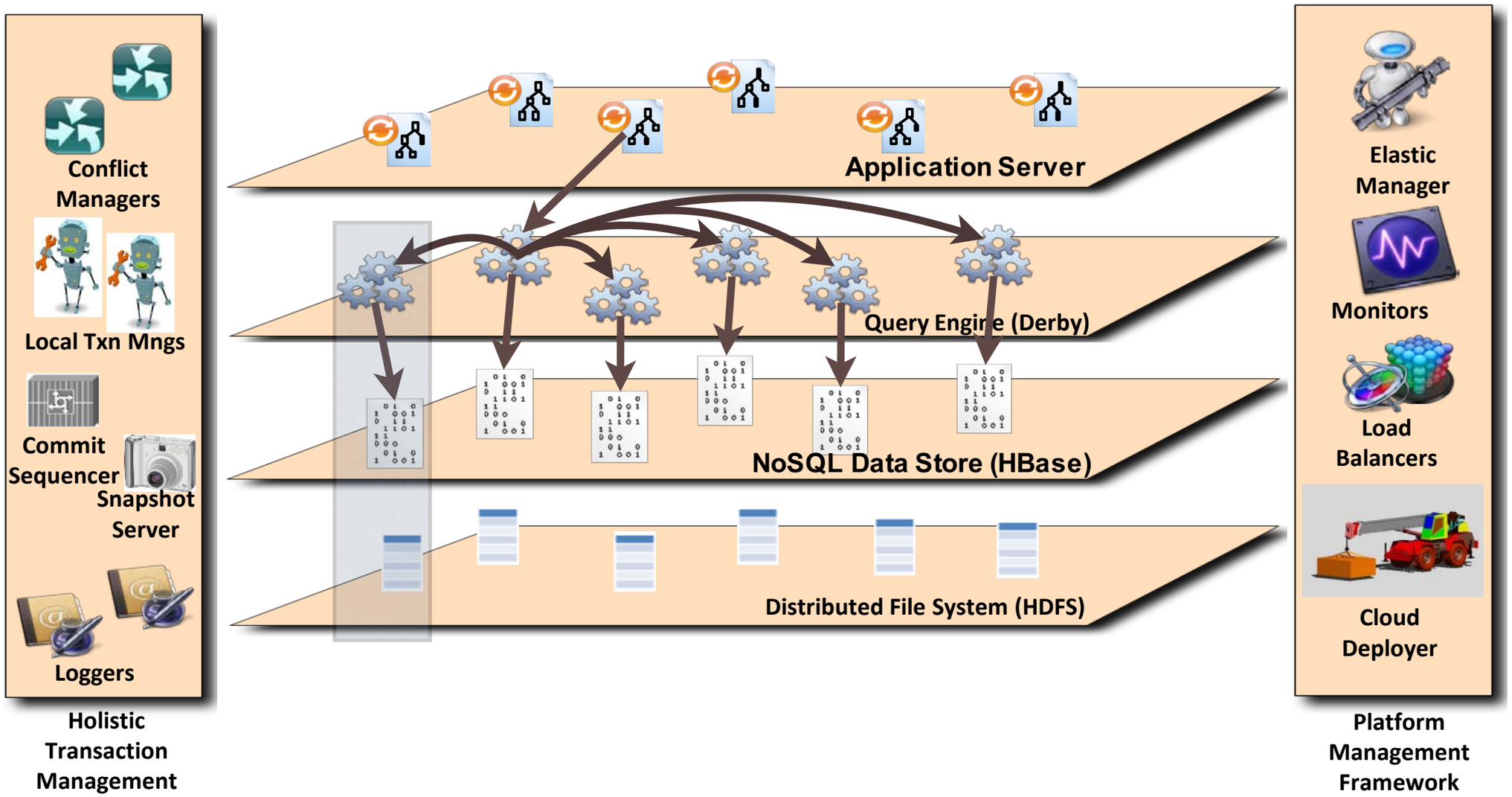
OLAP Architecture



OLAP Architecture

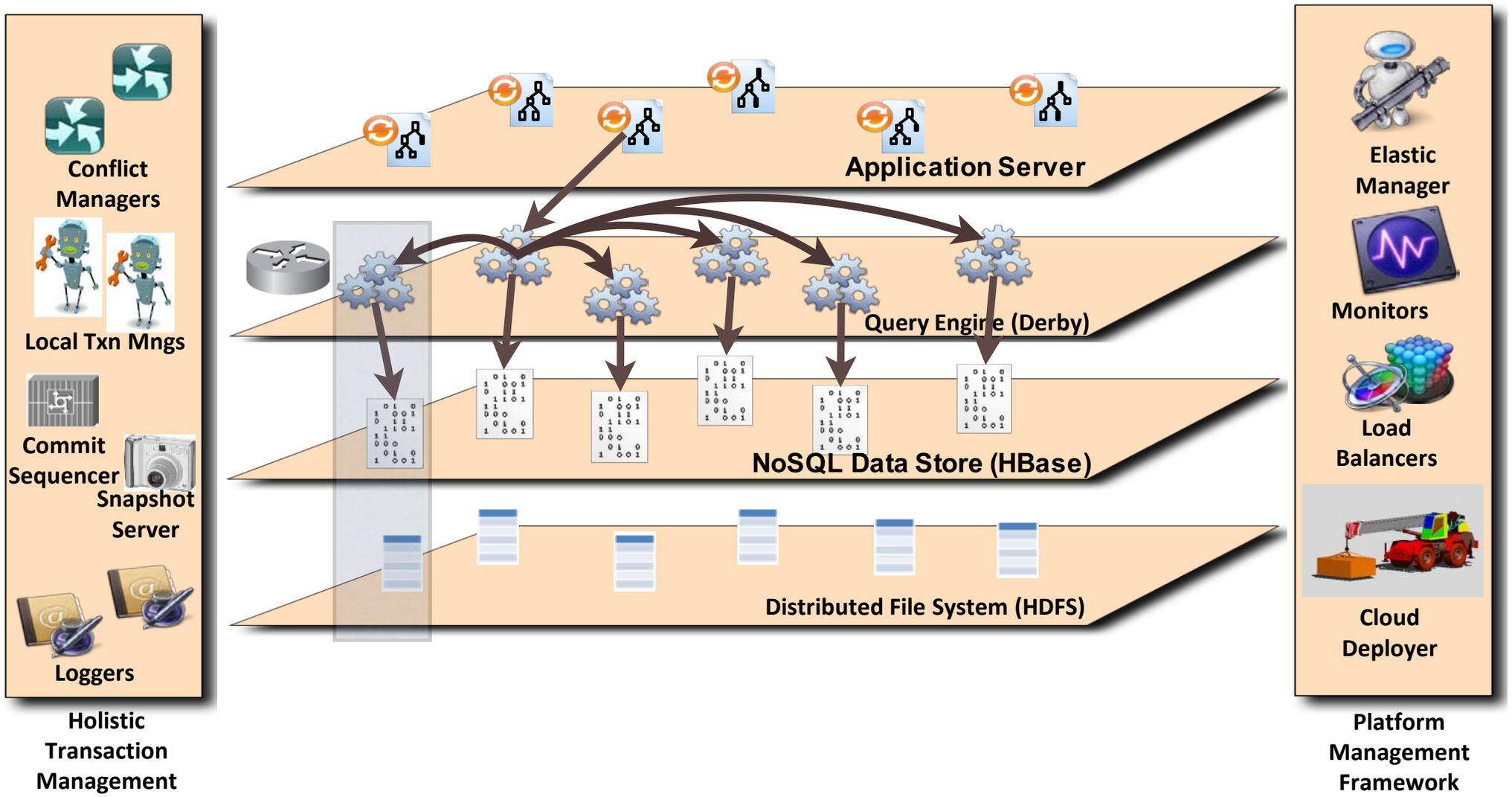


OLAP Architecture



CumuloNimbo can leverage data locality, when query engines, HBase and HDFS are co-located on the same physical machine.

OLAP Architecture

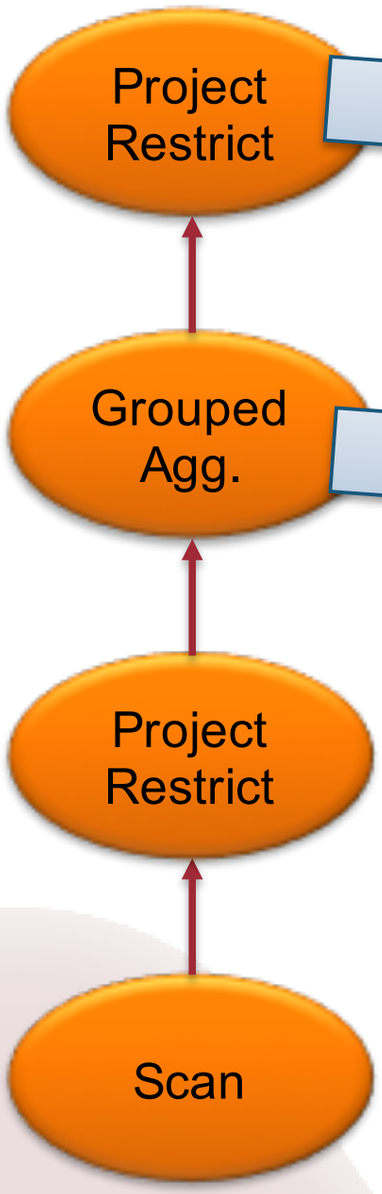


CumuloNimbo can leverage data locality, when query engines, HBase and HDFS are co-located on the same physical machine.

Parallel Coordination

- Maintains a set of symmetric workers for a user connection
- Initializes a **communication middleware** for efficient intra-query row exchange among workers
- Spawns a **parallel query plan** for all workers to execute
- Schedules disjoint subsets of data to scan to the different workers
- Collects results and errors

Parallel Query Plans

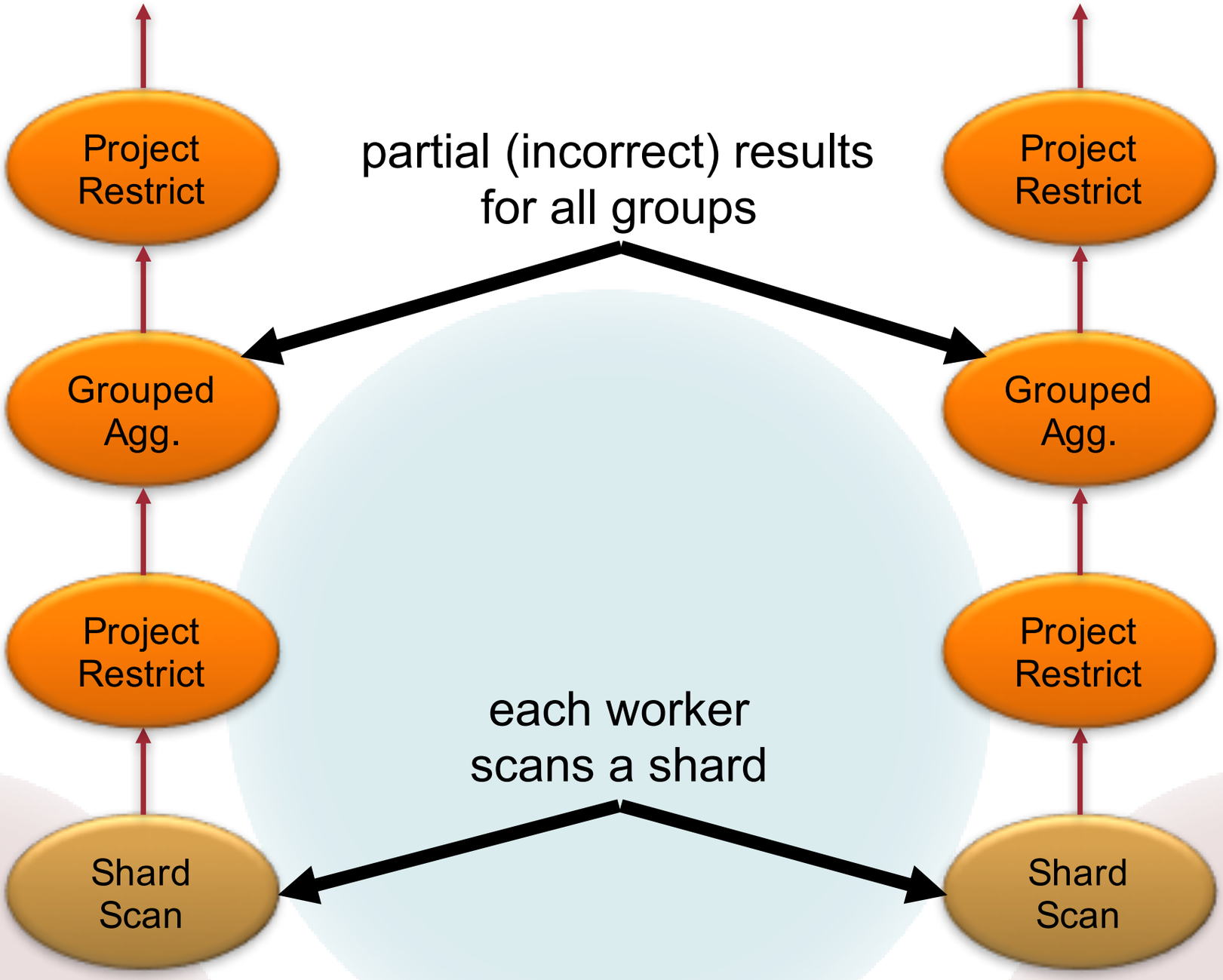


stateless operator

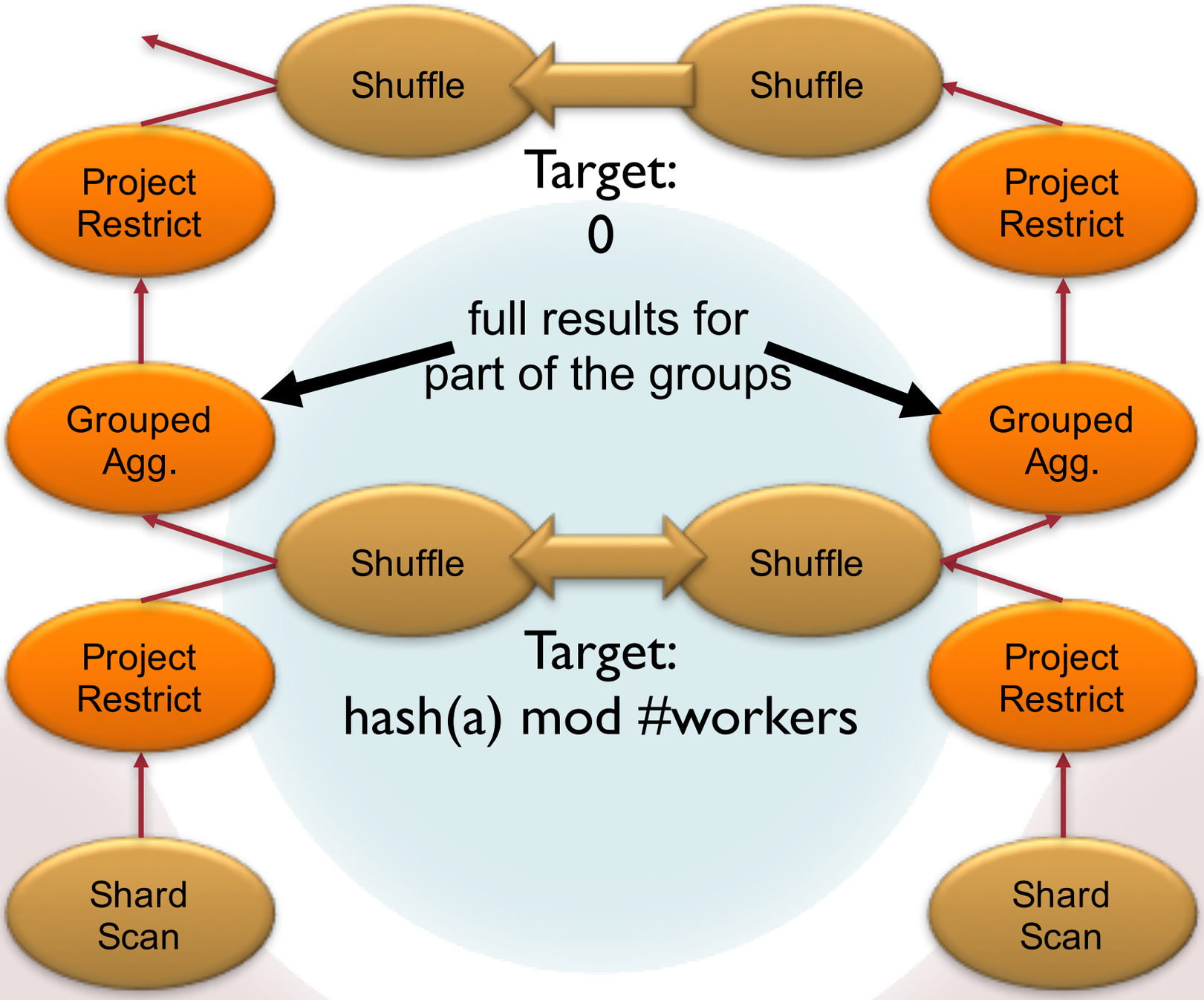
stateful operator

```
select a, avg(b) from t
        where c > 0
        group by a
        having avg(b) > 0
```

Parallel Query Plans



Parallel Query Plans



Communication Middleware

- Provides efficient row exchange for shuffle operators
- Supports hash-based shuffling
- Asynchronous and multi-stage
- Multiple implementations available (e.g. sockets, RDMA Verbs)

Conclusions

- CumuloNimbo platform offers distributed and highly scalable SQL processing, with full ACID properties
  - OLTP components iterated over 5 years: beta ready
  - OLAP has a fully working implementation
- Ongoing work
  - XA
  - HTAP (OLAP+OLTP)

Thank you for your attention

HASLab
HIGH-ASSURANCE SOFTWARE LABORATORY