TOWARDS SCALABLE REAL-TIME ANALYTICS:
AN ARCHITECTURE FOR SCALE-OUT OF OLXP WORKLOADS

Anil K Goel, Jeffrey Pound, Nathan Auch, Peter Bumbulis, Scott MacLean, SAP Labs Canada
Franz Fä rber, Francis Gropengiesser, Christian Mathis, Thomas Bodner, SAP SE, Germany
Wolfgang Lehner, TU Dresden, Dresden, Germany

Joy Arulraj, Reading Group Fall 2015

CARNEGIE MELLON
DATABASE GROUP

Motivation

- Real-time analytics
  - *Business value through data scientists*
  - *Transactions (OLTP) + Analytics (OLAP)*
- Hardware trends
  - *Low-latency network interconnect*
  - *Storage class memory*

Motivation

- Elastic scale
  - *Large load fluctuation*
  - *Dynamic resource provisioning*
- Rapid code-shipping
  - *Separating functional components*
  - *Decouple development and release cycles*

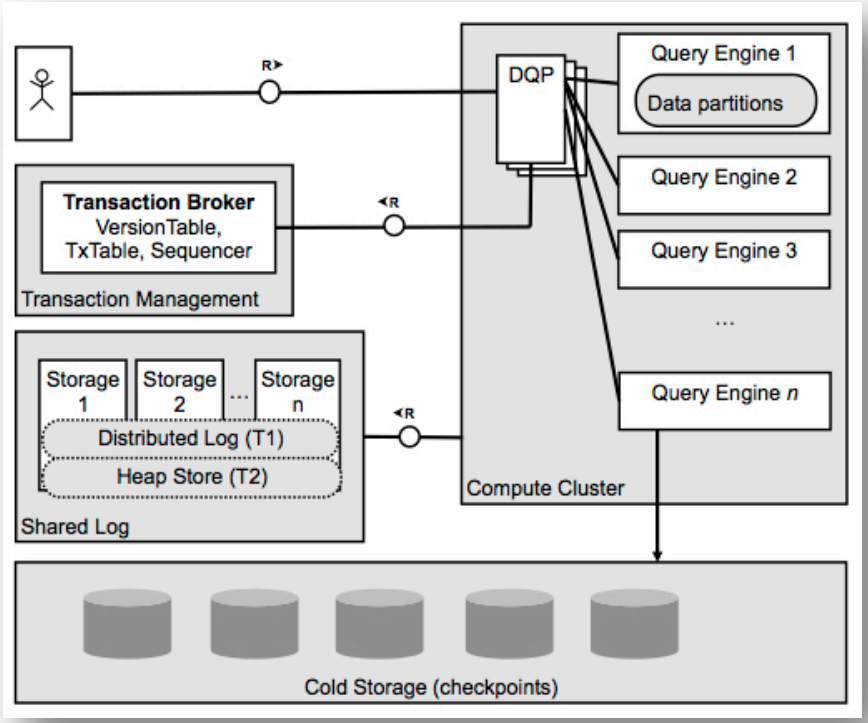
Key Question

How can we scale-out OLTP and OLAP workloads independently in a cluster ?

HANA-SOE Architecture

Transaction Broker

Shared Log



Query Engine

Three Components

COMPONENTS	PURPOSE
Transaction Broker	Shared state for concurrency control
Shared Log	Durability mechanism on top of storage units
Query Engine	Slice-oriented in-memory SQL engine

Interplay of components

- Transaction Broker
  - *Issues timestamps and does validation*
- Shared Log
  - *Persists updates and supports versioning*
- Query Engine
  - *Performs local changes*

1. Transaction Broker

- Decouple txn from query processing
  - *MVCC layer on top of the slice abstraction*
- Strong snapshot isolation
  - *Asynchronous update propagation to slices*
- Independent OLAP scale-out
  - *No distributed commit protocol*

1. Transaction Broker

- Efficient cross-partition transactions
  - *Separate partition-level updates to the log*
  - *Shared log takes care of ordering*
- Epoch-based versioning scheme
  - *All current transactions aborted on broker failure*
  - *No "split-brain" scenario*

1. Transaction Broker

- Query engine contract
  - *Answer query at requested logical timestamp*
  - *In-memory snapshots kept in-sync using log*
- Scheduling analytics transactions
  - *Run at same read timestamp based on SLA*

2. Shared Log

- Distributed shared log
  - *Key-metadata-value store*
  - *Total order over all writes for linearizability*
- Scan operation
  - *Bulk-read log entries based on predicate*
  - *Metadata can be slice identifier*

2. Shared Log

- Each transaction corresponds to a LSN
  - *LSN acquired by transaction broker*
- Implementation
  - *Partitioned and replicated entries over a cluster*
  - *Distributed hash table for partitioning*
  - *Chain replication protocol*

2. Shared Log

- Storage units
  - *Asynchronous I/O operations on SSDs*
  - *NVM-optimized design*
  - *RDMA support to allow scatter-gather reads*
- Log compaction
  - *Entries corresponding to same key*

3. Query Engine

- Distributed query processor
  - *Distributed execution plan*
  - *Mapping from slices to compute nodes*
- Data manager
  - *Read log and apply updates to build versions*
  - *Main-memory column store*

3. Query Engine

- SQL-to-C code generator
  - *Physical query plan to C code*
  - *LLVM to transform C to native code*
- Late materialization
  - *Intermediate operators generate only row-ids*
  - *Result printer materializes the result*

3. Query Engine

- Update processing
  - *Perform locally cached updates*
  - *Write validated updates to log*
- Checkpoint
  - *Per-slice per-version by the engine to cold storage*

Summary

COMPONENTS	PURPOSE
Transaction Broker	Shared state for concurrency control
Shared Log	Durability mechanism on top of storage units
Query Engine	Slice-oriented in-memory SQL engine

Conclusions

- Decouple OLTP and OLAP processing
  - *Distinguish three types of services*
- Scale-out of mixed OLTP/OLAP workloads
  - *Strict SLA on data freshness for analytics*

END

@jarulraj