CockroachDB & Raft

DS8003 – MGT OF BIG DATA AND TOOLS

Ryerson University

Instructor: Kanchana Padmanabhan

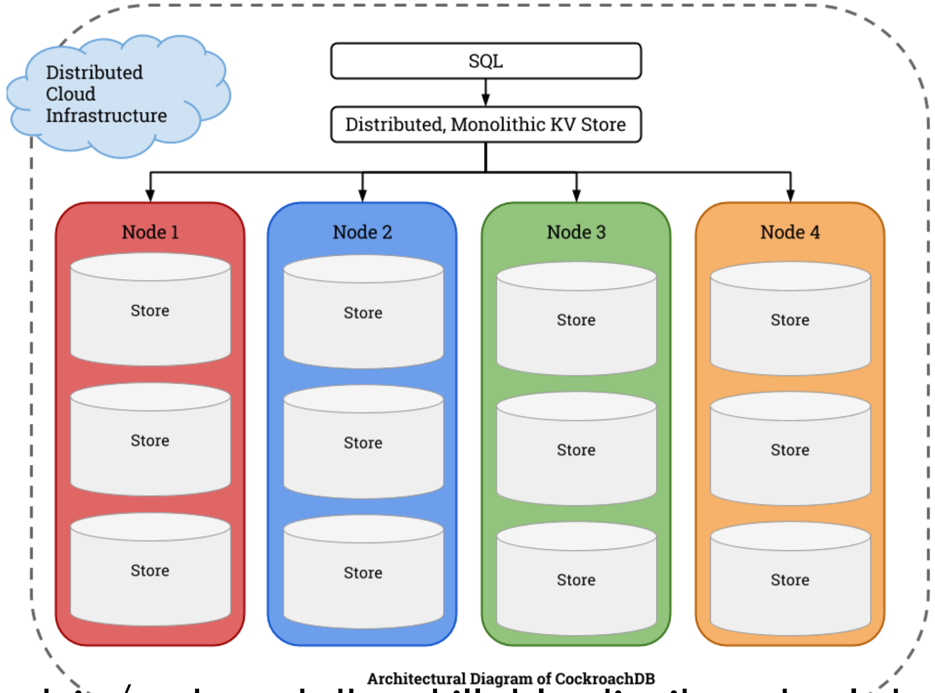
What is CockroachDB?

- CockroachDB is a distributed SQL database built on top of a transactional and consistent key-value store
- Design goals: ACID transactions, horizontal scalability and survivability (hence the name).
- Uses Raft consensus algorithm for consistency
- Aims to tolerate disk, machine, rack and even data center failures with minimal disruption and no manual intervention.
- Simple: Single binary with no external dependencies (https://www.cockroachlabs.com/docs/install-cockroachdb.html)

http://thenewstack.io/cockroachdb-unkillable-distributed-sql-database/

Architecture: Layered

- Structured in layers that make complexity an easier task to manage.
- Each higher level in the architecture treats the lower levels as functional black boxes
- While the lower layers remain completely unaware of the higher ones.


Distributed Cloud Infrastructure
SQL
Distributed, Monolithic KV Store
Node 1
Node 2
Node 3
Node 4
Store
Architectural Diagram of CockroachDB

http://thenewstack.io/cockroachdb-unkillable-distributed-sql-database/

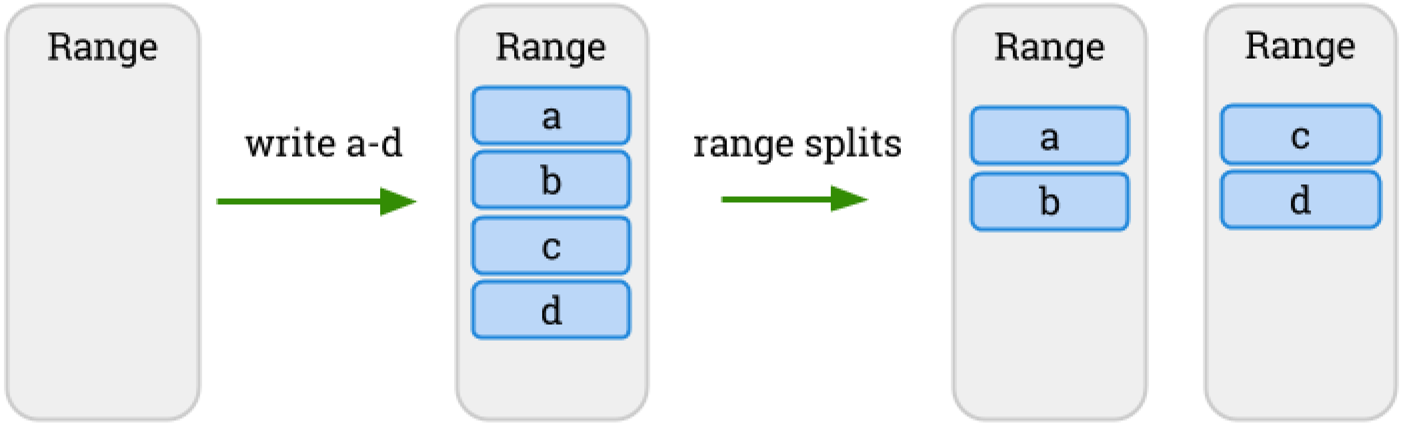
Architecture: Layered

- **SQL Layer:** Relational concepts such as schemas, tables and indexes using a derivative of the Postgres grammar
- **Distributed Key:Value Store:** We implement our distributed key-value store as a monolithic sorted map, helps create large tables and indexes (Hbase, BigTable, and Spanner all use similar architectures). Keys and values are both strings which can contain unrestricted byte values
- **Nodes:** The physical machines, virtual machines, or containers that contain stores. The distributed KV store routes messages to nodes.
- **Store:** Each node contains one or more stores, and each store contains potentially many ranges. Every store is managed with RocksDB
- **Range:** Every store contains ranges, which are the lowest-level unit of key-value data. Each range covers a contiguous segment of the larger key-space. Together, the ranges make up the entire monolithic sorted map. The range is where we do synchronous replication, usually three or five way, using the Raft consensus algorithm, a variant of Paxos.

http://thenewstack.io/cockroachdb-unkillable-distributed-sql-database/

Horizontal Scaling

- Data is logically organized into tables, rows, columns
- Individual pieces of data (think of a single column value) are stored on-disk in a sorted key-value map.
- starts off with a single, empty range of key-value data encompassing the entire key space

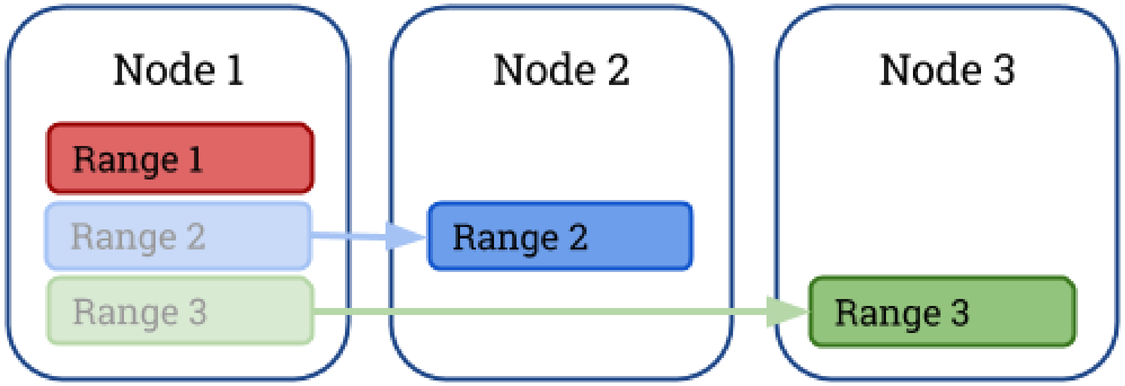


- monolithic sorted map in CockroachDB is made up of the sorted set of all ranges

http://thenewstack.io/cockroachdb-unkillable-distributed-sql-database/

Horizontal Scaling

- Create small ranges as they're easily moved between machines when repairing or rebalancing data.
- Existing ranges will continue to split into new ranges, aiming to keep a relatively consistent range size somewhere between 32MB and 64MB.

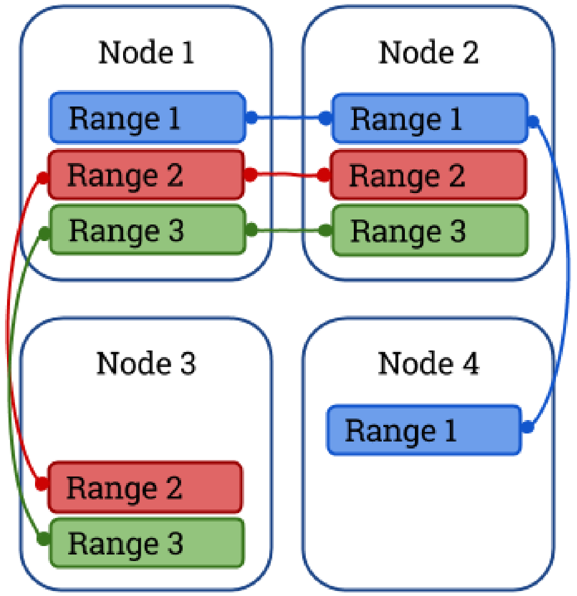


Range Distribution Across Nodes in CockroachDB

http://thenewstack.io/cockroachdb-unkillable-distributed-sql-database/

Replication

- Each range is replicated on three nodes
- Range replicas are intended to be located in disparate datacenters for survivability



Range Replication in CockroachDB

http://thenewstack.io/cockroachdb-unkillable-distributed-sql-database/

Replication: Consistency

- Data that is stored across multiple machines, it’s important that the data be consistent across replicas
- Raft as its consensus protocol.
- Each range is an independent instance of the Raft protocol, so we have many ranges all independently running Raft.

http://thenewstack.io/cockroachdb-unkillable-distributed-sql-database/

Raft Algorithm

- https://raft.github.io/slides/craftconf2014.pdf
- https://raft.github.io/
- https://raft.github.io/raftscope-replay/
- **https://www.cockroachlabs.com/blog/scaling-raft/**
- **https://raft.github.io/raftscope-replay/**

https://www.cockroachlabs.com/blog/scaling-raft/

Distributed Transactions

- Strong consistency and full support of distributed ACID transactions
- Distributed transactions using multi-version concurrency control (MVCC).
- MVCC data is stored and managed on each local storage device with an instance of RocksDB.
- Mutations to MVCC data are consistently replicated using Raft.

http://thenewstack.io/cockroachdb-unkillable-distributed-sql-database/

Multiversion concurrency control

- Each user connected to the database sees a *snapshot* of the database at a particular instant in time.
- Any changes made by a writer will not be seen by other users of the database until the changes have been committed
- When an MVCC database needs to update an item of data, it will not overwrite the old data with new data,
- It marks the old data as obsolete and adds the newer version elsewhere. Thus there are multiple versions stored, but only one is the latest.

https://en.wikipedia.org/wiki/Multiversion_concurrency_control

Distributed Transactions

- Snapshot isolation (SI) allows externally consistent, lock-free reads and writes, both from a historical snapshot timestamp and from the current wall clock time
- **Snapshot isolation** is a guarantee that all reads made in a transaction will see a consistent snapshot of the database (in practice it reads the last committed values that existed at the time it started)
- The transaction itself will successfully commit only if no updates it has made conflict with any concurrent updates made since that snapshot.

https://en.wikipedia.org/wiki/Snapshot_isolation
https://www.cockroachlabs.com/blog/how-cockroachdb-distributes-atomic-transactions/

SQL implementation (API)

- Leverages the monolithic sorted key-value map to store all of the SQL table data and indexes
- Encode, store, and retrieve the SQL table data and indexes.
- The SQL grammar supported is a derivative of PostgreSQL
- Provide query parsing, query analysis, query planning, query execution

https://www.cockroachlabs.com/blog/cockroachdbs-first-join/

http://thenewstack.io/cockroachdb-unkillable-distributed-sql-database/

https://www.cockroachlabs.com/blog/sql-in-cockroachdb-mapping-table-data-to-key-value-storage/

THE END