“big picture” for Cassandra

1. Stakeholders

Some major stakeholders in Cassandra:

- CERN uses Cassandra for its prototype ATLAS experiments.[1]
- Discord uses Cassandra to store messages. [2]
- Netflix uses Cassandra because it scales both horizontally and dynamically. [3]

More than 1500 other companies worldwide are using Cassandra for large amount data-storing.

DataStax created its own version database: DataStax Enterprise (DSE) based on Cassandra. DataStax also holds a community and an online academy for Cassandra users and developers.

2. Functionality

2.1. Overall domain

Apache Cassandra is a NoSQL, distributed database of high scalability and availability with no single point of failure. It is a prefect choice if you want to handle huge amount of data and take no risk in losing it .

2.2. Essential functional aspects[4]

- Store data: The base function of any database system

- High performance: With the help of distributed system, any authorized user can connect to any node at any data center using CQL language, with very fast reading and writing speed
- Reliability: Cassandra has no single point failure and ensures no data loss
- Recoverability: Any damaged node can be recovered very easily without losing any other data by removing that node temporarily, clear corrupted data and put that node back to the ring.

2.3. Essential non-functional aspects[4]

- Availability: Cassandra is an open source database management system that every one and every organization can use.
- Cloud friendly: Cassandra is compatible with cloud infrastructures
- Distributed: As one of the NoSQL databases, Cassandra is full distributed. With distributed data centers at different locations of the world as well as nodes in each center, the query activity can be done within a very short amount of time. Every node will perform a proportionate part of the activity.
- Scalable: Cassandra has linear scalability, which means that when more data needs to be stored in the system, simply increasing capacity horizontally (data center) or vertically (nodes) will do the job.

2.4. Uniqueness

There are a few features which make Cassandra attract attention out of all other databases.[5]

1. Cassandra allows for all kinds of data structures including structured, unstructured, and semi-structured data. It supports dynamic changes so that the data structure always fulfill your requirements.
2. Cassandra can be easily scaled to meet a sudden increase in demand by deploying multi-node Cassandra clusters.
3. Cassandra allows easy data distribution among various data centers by replication.
4. Cassandra is able to handle failures without affecting the performance at all because it has no single node failure, which makes it perfect for data-critical situations.
5. Cassandra supports for ACID (atomicity, consistency, isolation, and durability), which is supported by relational database systems.
6. Cassandra supports high speed of data writing without affecting the reading efficiency.

3. Key Developers

- Original authors:
  - Avinash Lakshman, one of the original authors, from Facebook
  - Prashant Malik, one of the original authors, from Facebook
- Top contributors on Github:
  - Jonathan Ellis (@jbellis) from DataStax, 4526 commits
  - Sylvain Lebresne (@pcmanus) from DataStax, 1617 commits
  - Brandon Williams (@driftx) from DataStax, 1208 commits
  - Dave Brosius (@mebigfatguy), 734 commits
  - Aleksey Yeschenko (@iamaleksey) from Apple, 610 commits

4. Open Issues

4.1. Add test coverage workflows to CircleCI config

According to the issues page of Cassandra on Apache #14788:

To support 4.0 testing efforts it's helpful to know how much of the code is being exercised by unit tests and dtests. Add support for running the unit tests and dtests instrumented for test coverage on CircleCI and then combine the results of all tests (unit, dtest with vnodes, dtest without vnodes) into a single coverage report.

4.2. Overflow of 32-bit integer during compaction.

According to the issues page of Cassandra on Apache #14733:

When rounding the expriration time which is close to **Cell.MAX_DELETION_TIME**(it is just **Integer.MAX_VALUE**) the math overflow happens. Data type for point was changed from Long to Integer in order to reduce memory footprint), as result point became negative and acts as silent poison for internal structures of StreamingTombstoneHistogramBuilder like **DistanceHolder** and **DataHolder**.

4.3. Add ability to track state in repair

According to the issues page of Cassandra on Apache #15399:

To enhance the visibility in repair, we should expose internal state via virtual tables; the state should include coordinator as well as participant state (validation, sync, etc.)

The main reason for exposing virtual tables rather than exposing through durable tables is to make sure what is exposed is accurate. In cases of write failures or node failures, the durable tables could become in-accurate and could add edge cases where the repair is not running but the tables say it is; by relying on repair's internal in-memory bookkeeping, these problems go away.

4.4. Add a warning when RF > N

According to the issues page of Cassandra on Apache #15548

When testing Cassandra with network partitions, we find that keyspace creation can succeed without any warning even if there are not enough nodes to support the replication factor. Here are the steps to reproduce:

1. Start a cluster w/ two nodes.
2. Create a keyspace with replication factor of three.
3. Notice that the creation succeeds without any warning.

4.5. Harry: generator library and extensible framework for fuzz testing Apache Cassandra

According to the issues page of Cassandra on Apache #15348:

Current testing tooling largely tests for common- and edge-cases, and most of the tests use predefined datasets. Property-based tests can help explore a broader range of states, but often require either a complex model or a large state to test against.

Harry allows to run tests that are able to validate state of both dense nodes (to test local read-write path) and large clusters (to test distributed read-write path), and do it efficiently. Main goals, and what sets it apart from the other testing tools is:

- The state required for verification should remain as compact as possible.
- The verification process itself should be as performant as possible.
- Ideally, we'd want a way to verify database state while *continuing* running state change queries against it.

---

1. https://cdsweb.cern.ch/record/1432912↩
2. https://blog.discordapp.com/how-discord-stores-billions-of-messages-7fa6ec7ee4c7↩
3. https://netflixtechblog.com/nosql-at-netflix-e937b660b4c↩
4. https://www.rapidvaluesolutions.com/tech_blog/cassandra-the-right-data-store-for-scalability-performance-availability-and-maintainability/↩↩
5. What is Cassandra↩