proposal

November 9, 2018

1 Topic Proposal (initial draft, to be updated)

1.1 Introducing and Comparing NoSQL Database Management Systems

1.1.1 Abstract

With the exponentially generated and accumulated data from varied sources like mobile devices, mechanical sensors, financial transactions, satellite imaging, the task of data managing gets much more challenging day after day. And these collected raw data are in a wide range of forms, most are unstructured, like images, videos, audios, etc. In the whole data processing pipeline, starting from data integrating and storing, it's already a quite essential part. Since the consistency, accessibility, velocity of data inserting and retrieving might make a huge difference to the following data manipulation tasks, especially in fields like finance, social networks, etc. Thus making an optimal choice of the right database for different data format, different data volume, different business focus like whether velocity or consistency is more important for the client's perspective.

Widely used Database Management Systems are Relational DBMS such as SQL (MySQL, SQL Server), and NoSQL(Not Only SQL). Why the relational DBMS is limited is that due to its strong focus on consistency and reliability and ACID guaranteed (ACID: Atomicity, Consistency, Isolation and Durability), thus also making it difficult for scaling, which is one of the many reasons why NoSQL is considered better than SQL. To reach a balance between availability and consistency, NoSQL achieve more optimal level of consistency by slightly sacrificing its availability.

NoSQL also can reduce the level of availability to achieve higher consistency. However there are still many remaining problems which NoSQL databases cannot solve. As a result, providers produced distinct data model CDBMSs(Cloud Database Management System) to satisfy different requirements, whereas each of them are present an imperfection.

By design, NoSQL databases and management systems are relation-less (or schema-less). They are not based on a single model (e.g. relational model of RDBMSs) and each 'database, depending on their target-functionality, adopt a different one.

There are almost a handful of different operational models and functioning systems for NoSQL databases.:

1.1.2 Goal

The proposed goal of the paper is to give a precise introduction of the NoSQL DBMS and provide a comparison of some of the most frequently used in the industries nowadays, Cassandra, HBase and MongoDB, mainly from 4 aspects: their mechanism, important features, advantages, existing problems.

1.1.3 Proposing Outline

NoSQL Data Models There are four frequently used data models of NoSQL databases:

- Key / Value Store:
  - e.g. Redis, MemcacheDB
- Column Store:
  - e.g. Cassandra, HBase
- Document:
  - e.g. MongoDB, Couchbase
- Graph:
  - e.g. OrientDB, Neo4J, etc.

Comparison From the varied industrial level used NoSQL DBMS, we mainly discuss the following 3 and would make comprehensive comparisons among these DBMS:

- Cassandra
- Mongodb
- Hbase

on factors listed below:

- mechanism
  - Architecture
  - Data Model
- features
  - Partition
  - Replication
- advantage
- disadvantage

Cassandra (very rough example, more details will be concluded) Cassandra is an open source DBMS, with a peer-to-peer distributed system across its nodes, in this way it achieves a high availability. Its data model si inherited from BigTable, with the keyspaces being the container of data, in which contains column families. Row-keys are the identifiers of columns. Cassandra could always achieve a better performance by querying on row-keys.

As a distributed database management system, it partitions the data across all participating nodes in the cluster. There are two partitioning strategies in Cassandra: Random partitioning (default and recommended) and Ordered partitioning. If partitioning is initialized, every change made will not be processed without reloading the whole data again.

In every transaction, Cassandra replicates data to achieve high availability by storing replicas on multiple nodes in the cluster to guarantee reasonable fault tolerance. So Cassandra rely on a high speed and low latency network connection. Since it does not operate under the master-slave replication strategy like in HBase, thus situations like the single points failures do not exist in Cassandra, ensuring higher level of availability. But in this case, the extra transactions incurred between could be expensive.

In the CAP(Consistency, Availability, Partition tolerance) theorem, Cassandra satisfied the Availability and Partition tolerance, these are its major advantages. It needs to trade-off between availability and consistency depending on the actual context of tasks, it can achieve strong consistency and eventual consistency by configuration.