Understanding The Architecture Of NoSql Databases

Shivika Sodhi
Computer Science Department
San Jose State University
San Jose, CA 95192
408-924-1000
shivika.sodhi@sjsu.edu

ABSTRACT

With the advent of 21st century, arose the need of databases that can handle large amount of data. Today the world needs databases to be able to store and process big data effectively, there is a demand for very high-performance when reading and writing, especially in large scale and high concurrency applications.

In order to store that data, a huge amount of memory is required. Hence, more than one storage mechanism is needed so handle colossal data. Thus, NoSql, which means "Not only Sql" came into picture. It improved efficiency and balanced CPU memory utilization by having shared CPUs. Moreover, NoSql allows the development of a program without having to convert in-memory structures to relational structures, hence there's comparatively less chance of a mismatch.

The purpose of this paper is to introduce the architecture of various NoSql tools and talk about the scalability of NoSql. Scalability is a very important factor for a database for faster and more efficient computation. It is also said that successful web services do not only have big user bases, the performance of computer hardware is also a big factor for them to be reliable. So at one point or another a web application which attempts to become big needs the ability to scale

1. INTRODUCTION

In recent years, a huge number of systems have shifted to horizontal scalability for simple read/write database operations distributed over many servers, as there are thousands or millions of users doing updates as well as reads on the database at the same time. In contrast to that, traditional database products have comparatively little or no ability to scale horizontally and thus have a limited scope. In this paper, we shall discuss various types of systems for horizontal scalability.

A database can be scalable in three different ways. It can be scalable with the amount of read operations, the number of write operations and the size of the database.

Scalability of web applications means that any workload or any amount of data can be processed in a defined time if the application can run on enough servers, which will be referred to in this text as nodes. The relations between workload, time and needed machines would be linear in an ideal case. Relational databases (traditionally) reside on one server, which can be scaled by adding more processors, more memory and external storage. Relational database residing on multiple servers usually uses replications to keep database synchronization. There are several different applications nowadays and every application has different requirement towards scalability. Some need the ability to serve a large number of users at the same time and others might need to be able to scale with the amount of data.

Web companies such as Google and Facebook requires processing huge quantity of data. The requirements of these companies resulted in the surge in popularity of NoSQL databases. NoSQL databases has several features, but only six key features, discussed below:

i. The ability to horizontally scale "simple operation" throughput over many years.
ii. The call level interface, in contrast to a SQL binding is simple.
iii. A comparatively weaker model than ACID transactions of SQL.
iv. The use of distributed indexes and RAM for data storage is efficient.
v. One of the most crucial ones is the ability to dynamically add new attributes to data records.

2. TYPES OF NOSQL DATABASES

2.1 Key-Value stores

These are the simplest kind of NoSQL databases where the data is stored in the form of key-value pairs. The values are indexed for retrieval by keys. Values can contain any kind of data, structured or unstructured.
The advantages of key value stores includes its scalability and simplicity of its query model which usually consists of set, get and delete queries.

The disadvantages are that it provides no mechanism for querying for keys based on the content of the value.

2.2 Document stores

These databases store the data in the form of collection of documents. Documents are addressed in a database by a unique key which represents that document. The documents are stored in JSON or JSON like format.

The advantage is that the database provides query language which enables retrieval of documents based on their content.

The disadvantage is the complexity in its implementation.

2.3 Column store

In these databases, one extendable column consist of closely related data. A key identifies a row, which contains data stored in one or more column families. It stores versioned chunks of data in large table.

The advantages are that it provides versioning and has good scalability.

The disadvantage is that row and column designs are critical.

2.4 Graph Stores

These kinds of databases store data in a series of nodes, relationships and properties. The data such as social networks, road maps can be stored using such databases as the relation between them can be represented as graphs.

The advantage is that it provides fast network search and works with public linked data sets.

The disadvantages are that it provides poor scalability when graphs don't fit into RAM and also it needs specialized query languages.

3. ADVANTAGES OF NOSQL

3.1 Automatic Sharding

Automatic sharding is a phenomenon which states that when one node in a cluster has too much load the system should be able to automatically rebalance the load distribution. A NoSQL database automatically spreads the data across different nodes without requiring the participation of applications.

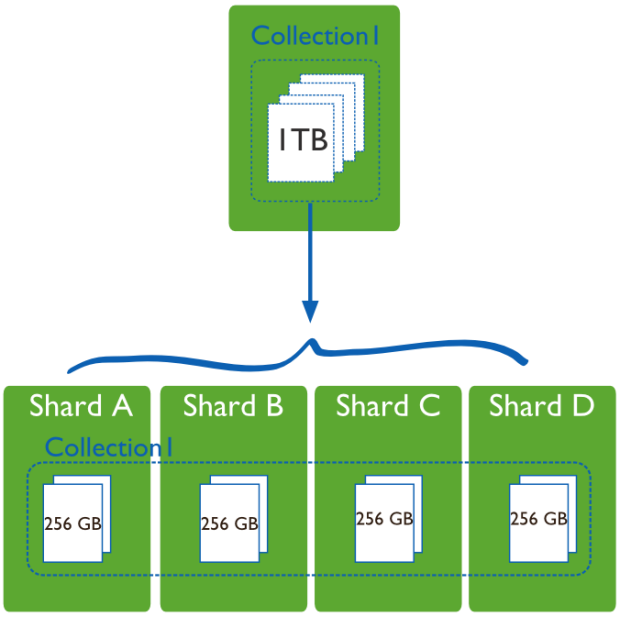


Figure 1: Sharding

3.2 Schema-Free Integration

NoSQL has a schema free approach that is it does not follow any pre-defined schema to store the data. It does not have any preconceived notion about the kind of data it will get. This approach helps NoSQL databases to store any kind of data.

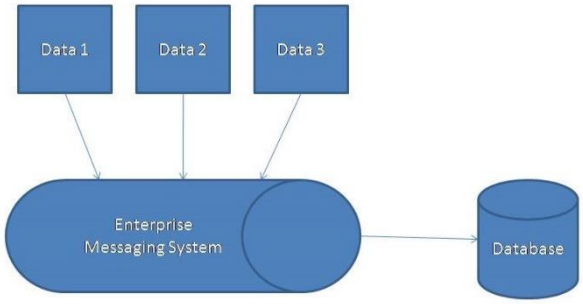


Figure 2: Schema-free integration

3.3 Horizontal Scaling

This is probably the biggest advantage that NoSQL databases provide. Relational Databases fail to provide effective horizontal scaling. NoSQL are designed to work across a cloud of servers.

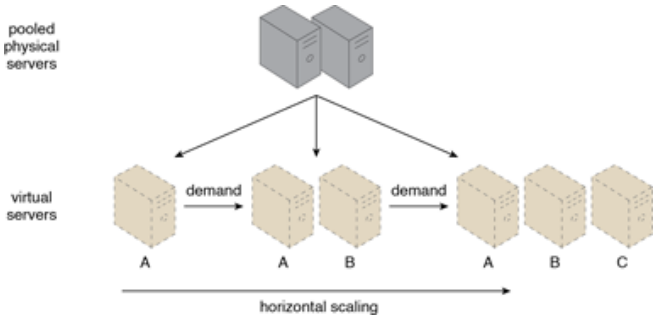


Figure 3: Horizontal Scaling

3.4 Integrated caching

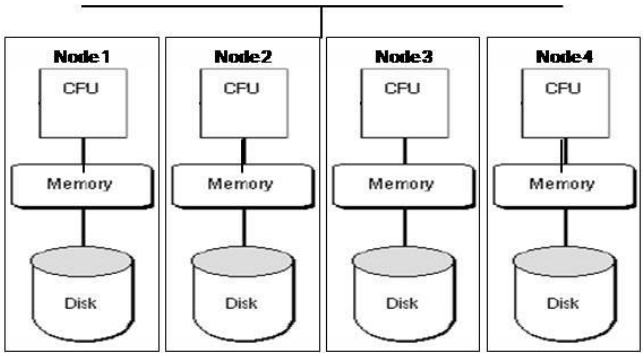
Advanced NOSQL databases transparently cache data in system memory to reduce latency and increase sustained data throughput.

4. NoSQL ARCHITECTURE

NoSQL architecture tries to achieve the shared nothing architecture. The term shared nothing means that when the data repository is being sharded the nodes should not be dependent on each other for any kind of updates. Any change inside any cluster should not affect any other cluster.

This kind of architecture can be achieved by introducing an extra scalable state which lies in the application layer and data is stored inside the memory RAM which is much quicker as compared to the disk. This new layer should manage data as objects, because objects are what the application needs. These objects should be kept in memory, in order to improve performance and avoid the disk input/output bottlenecks that burden databases. The layer should be able to transparently load the missing data from the persistent store behind it. The data modifications are propagated to persistent store asynchronously. The data is stored in the persistent data store asynchronously because the access to the disk takes time. So the read and write request of the user are taken care by the data stored in memory so that that the request are responded at high speed and

then the data is stored inside the disk so that it won't be lost if any damage occurs.



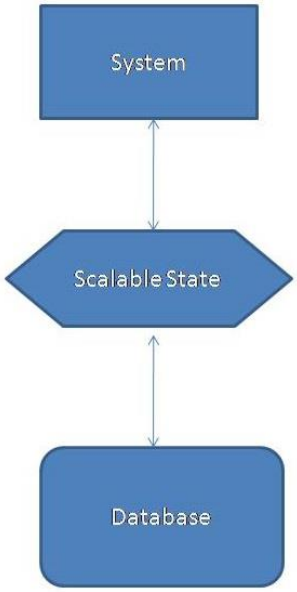


Figure 4: Shared nothing Architecture

5. NoSQL ARCHITECTURE - EXAMPLES

This section deals with the examples of few famous NoSQL databases and their architecture. We will discuss the architecture of few famous NoSQL databases such as MongoDB, CouchDB, Cassandra and HBase. The examples showcase different types of NoSQL databases.

5.1 MongoDB Architecture

MongoDB uses memory-mapped files for storage. This lets the operating system's virtual memory manager decide which parts of the database are stored in memory and which one only on the disk. This is the reason why MongoDB cannot control, when the data is written to the hard disk. The memory mapped files are used to instrument as much of the available memory as possible to boost the performance. Currently an alternative storage for MongoDB is under development, which will allow MongoDB more control over the timing of read and write operations. In MongoDB, the indexes are stored as B-Trees.

Mongo DB architecture generally consists of three components which are shard nodes, configuration servers and mongos or routing services.

Shard nodes: Shard nodes store the actual data. The shard can consist of either one node or a replication pair. Developments are being made to enable shard to consist of more than two nodes for better read and write operations and for better redundancy.

Configuration servers: The configuration servers are used to store the routing information of the MongoDB cluster and the metadata. The configuration servers are accessed from the routing services and from the shard nodes.

Mongos: They are the routing services which are responsible for the performing of the tasks requested by client. Depending on the type of operation the mongos send the requests to the necessary shard nodes and merge results before returning the result to the client.

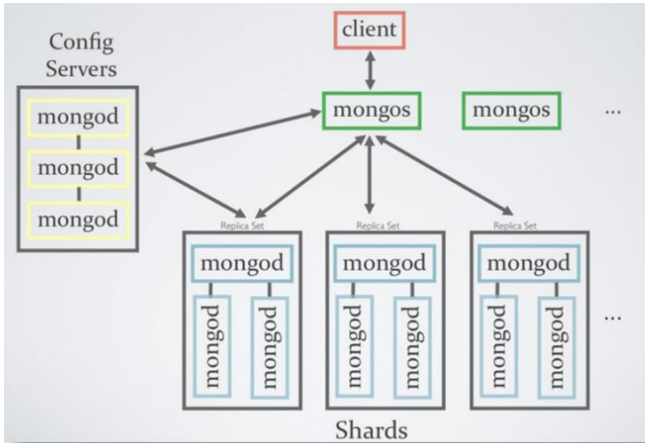


Figure 5: MongoDB Architecture

5.2 Cassandra Architecture

Cassandra is designed to handle big data workloads across multiple nodes. Cassandra has peer-to-peer distributed system across its nodes, and data is distributed among all the nodes in a cluster. Each node is independent as well as interconnected to each other. In Cassandra, one or more of the nodes in the cluster act as replicas for a piece of data. If during a read operation it is found that some of the nodes are reflecting out of date values then Cassandra will return the most recent value and performs a read repair in the background to update the out of date values. Cassandra uses Gossip protocol to allow the nodes to communicate with each other.

In Cassandra there is one node which acts as leader node (the node with which the client interacts) which assign and ranges the replicas to the storage nodes. The storage nodes are responsible for storage. The data is partitioned using consistent hashing. Each node is assigned a number that represents its position in the ring. The data items are assigned to a node by hashing its key and moving clockwise in the ring to find the first node with larger position. Each node is the coordinator for the data item that falls in the region between the node itself and the previous node. The coordinator is

responsible for copying the data item and replicating it on different nodes in the ring. Data is replicated on a number of nodes based on a replication factor and a replica replacement strategy. The node which acts as leader node is responsible for load balancing over the nodes so that no node is over burdened. Meta data like item-to-node mappings are stored in and cached in nodes.

Cassandra can be accessed through its nodes using Cassandra Query Language (CQL). It treats the database as a container of tables. Users approach any of the nodes with the read-write operations. The node works between the client and the nodes holding the data.

Cassandra uses an ordered hash index which provides most of the benefit of both has and B-tree indexes. Sorting though would be slower than with B-tress.

Cassandra's key structures include nodes, data center, cluster, commit log, table and SS table.

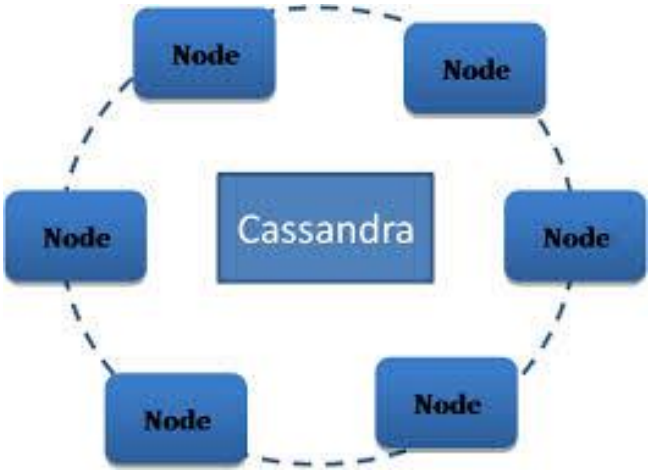
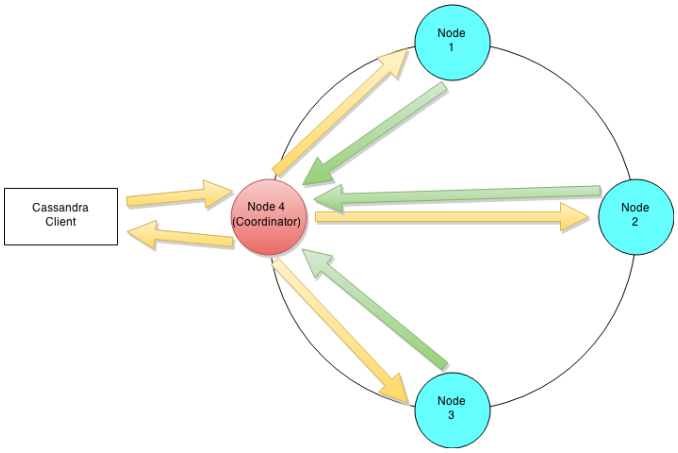


Figure 6: Node arrangement in Cassandra



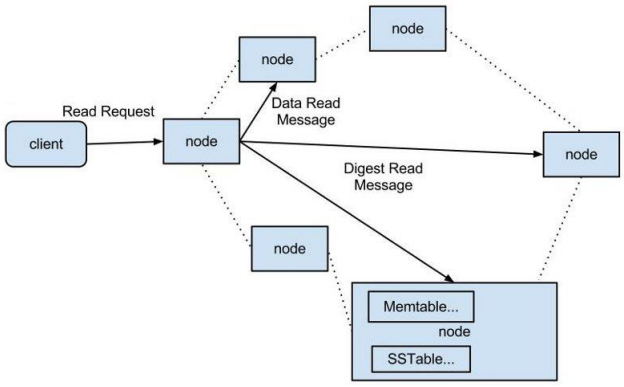


Figure 7: Cassandra interaction diagram

5.3 CouchDB

CouchDB has one server application, which contains the CouchDB integrated web server and the database service. CouchDB is a document oriented database which uses JSON to store data. The MapReduce functions are used for using JavaScript as a query language. CouchDB also has a pluggable language support which allows using other programming languages for the user functions. CouchDB can have both type of consistency eventual as well as strong. The type of consistency mainly depends on how the replication feature is used. If the replication feature is used in master-master configuration, then it provides eventual consistency. Else when used in master-slave configuration it provides strong consistency.

The query model for CouchDB consists of two things. First is Views which are build using MapReduce functions and second is a HTTP query API, which allows users to access and query the views.

There are three main components of CouchDB which are a storage engine, a view engine and a replicator

Storage Engine: The storage system is binary tree based. It is the core of the database which manages storing internal data, documents and views. In CouchDB, the data is accessed by keys or keys ranges which are mapped directly to the underlying binary tree operations. This results in significant improvement in speed.

View Engine: The view engine is based on Mozilla SpiderMonkey and written in JavaScript. It allows creation of adhoc views which are made of MapReduce jobs. When a user views a data in a view, CouchDB makes sure the data is up to date. They can be used for extracting data from documents and creating indices.

Replicator: The replicator is mainly responsible for replication of data to a remote or local database and synchronizing design documents.

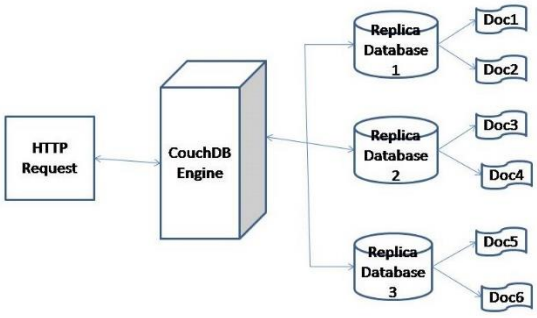


Figure 8: Simple architecture of CouchDB

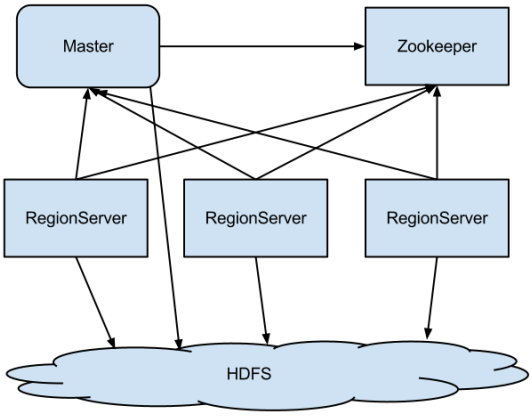


Figure 9: HBase Cluster

5.4 HBase Architecture

Built of top of HDFS, HBase is a scalable distributed database that supports structured data storage for large data tables. When the application requires real time read and write random access to large data set, it uses HBase. It is built from scratch just by adding nodes and is designed to solve the scaling problem from a different perspective than most other solutions. Unlike RDBMS HBase can store large data table on cluster made from commodity hardware. It lacks several features such as secondary indexes, triggers, typed columns and advanced query language.

An HBase cluster is actually two different clusters working together. The first cluster is the HDFS cluster which consists of one name node, which acts as the entry point of the cluster. This means that this nodes have the information regarding which are the data nodes that store any information required by the client.

The second cluster which is the HBase cluster is composed of one master cluster which acts as the entry point of the cluster and finally the region servers which serve the data. The master does the administrative tasks such as keeping the cluster balanced and moving regions from failing servers to other region servers.

6. SUMMARY/CONCLUSION

This paper has shown that the usage of a NoSql database can increase the performance significantly, as one CPU is dedicated to certain amount of data. If more data needs to be added, it can just be appended, unlike relational databases. This saves CPU cycles as it doesn't have to fit that data to specific places.

Since, speed and scalability are need of an hour, applications nowadays are shifting towards In-Memory data storage, which could boost the data access and the system could look forward for databases which could work according to the use cases and NoSql is a solution for them. It automatically and transparently fails over and redistributes its clustered data management services when a server becomes imperative or is disconnected from the network. It transparently redistributes the cluster load. NoSQL databases provide us with three important functionalities which are speed, scalability and fault tolerance.

One more important aspect of the NoSQL databases has been the varieties of database that are available to developers. Developers can choose the database according to their needs. The combination of relational databases and NoSQL will bring a big change in data storage.

The NoSQL database movement is still in its infancy and many of the popular NoSQL databases will change their design and architecture as they go ahead. But seeing the recent trend it seems the NoSQL databases are here to stay. They provide solution to the problems which are not addressed by relational databases.

.

7. REFERENCES

[1] https://en.wikipedia.org/wiki/NoSQL

[2] https://www.mongodb.com/nosql-explained

[3] http://www.couchbase.com/nosql-resources/what-is-no-sql

[4] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.184.483&rep=rep1&type=pdf

[5] http://macc.foxia.com/files/macc/files/macc_mccreary.pdf

[6] http://www.planetcassandra.org/what-is-nosql/

[7] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.403.8869&rep=rep1&type=pdf

[8] https://en.wikipedia.org/wiki/CAP_theorem

[9] http://www.researchgate.net/profile/Rabi_Padhy/publication/251442820_RDBMS_to_NoSQL_Reviewing_Some_Next-Generation_Non-Relational_Database's/links/5476b2940cf2778985b081b7.pdf

[10] http://www.aosabook.org/en/nosql.html

[11] https://en.wikipedia.org/wiki/Shared_nothing_architecture