ABSTRACT
Cassandra is open source and is in development at Apache. The Apache Cassandra project brings together Dynamo’s fully distributed design and Bigtables Column family based data model. Cassandra is adapting to recent advances in distributed algorithms like Accural style failure detection and others. Cassandra is proven as it is in use by Digg, Facebook, Twitter, Reddit, Rackspace, Cloudkick, Cisco. The largest production cluster has over 100TB of data in over 150 machines. It is Fault tolerant, decentralizes and gives the control to developers to choose between synchronous and asynchronous data replication. It offers rich data model, to efficiently compute using key and value pairs. It is financially viable through elastic scalability. It is durable and supports third party applications.

Cloud infrastructure particularly changed the way data is stored, accessed and computed. In the event of modern advances in social networking sites like Facebook, Twitter data storage creates new scenarios. Typically we can still use Relational databases, IBM DB2 or other solutions to solve the same problems. But, they might require running many scripts and superior algorithms that might be hard to be hidden from the developer.

Cassandra is a distributed storage system for managing structured data that is designed to scale to a very large size across many commodity servers, with no single point of failure. The idea is to run on top of an infrastructure of hundreds of nodes, where small and large components in the data centers fail continuously. Over the edge, Cassandra achieves scalability, high performance, high availability and applicability. It does not support a full relational data model. Instead it provides clients with a simple data model as explained later.

Cassandra is currently opensource, and it is with the Apache project and it is currently used by some of the most popular websites on the internet. It bases it’s distribution design on Amazon’s Dynamo and it’s data model on Google’s Bigtable.

The rest of the paper is organized as follows. Section 2 details related work and answers on why this technology became popular. Section 3 details on the design of Cassandra. Section 4 presents on the advances to Cassandra, open problems.
2. Related work
In this section, we contrast some of the other technologies with Cassandra, from the perspective of the features that are offered by Cassandra’s design. Whether or not these features are required by the application depends on the application.

Usability

RDMS had been a popular model for databases. It is handled using SQL queries, for insert, delete and update data elements. But database management is often more complex in some scenarios. For example, database used by Google or Facebook are more on the cloud and should guarantee availability even if some of the components fail. Which means data is replicated on several nodes. We can still use SQL queries in this scenario but the algorithms for fetching and updating will be more complex and adds burden onto the developers.

High Availability - Decentralized

MySQL, Bigtable and so many others provide solutions to the above problems. But then, they are not completely decentralized. They have a master-slave relationship, which means if the master fails then the availability decreases as the nodes below them fail as well.

Certainly, decentralized approach improves availability. But, if this degree of availability is needed or not depends on the application.

Elastic scalability

Data allocation is probabilistic and depends on the rate at which data is being created and destroyed. For example, gmail offers more than 7 GB of gmail space to its users. But definitely, if everyone on gmail starts using all the 7GB space, then gmail will crash. Because, the data allocation is probabilistic depending on the rate of information being generated. Therefore, a solution that keeps into the account and manages nodes well, where it can easily add more nodes when usage is high and delete them if they are not useful, would save lot of money for a company.

Cassandra internally, makes gives this option thereby making it a financially viable option.

Tuneable Consistency

When data is replicated, synchronized, it is often difficult to get the most recent form of it. Some distributed algorithms provide strict consistency where the most recent write is the one that is displayed. Some other techniques, without sound algorithms also make it possible to delay the transaction until unless it is known that this is the most recent write. But such a system, would have less availability in the case of multiple writes.

In either case, Cassandra offers tunable or close to eventual consistency. It is a tradeoff between consistency and replication, in such a way that performance and availability are maintained. CAP theorem states that in any given system, you can strongly support only two if the three (consistency, availability and Partition Tolerance). Therefore, Cassandra pushes itself onto high availability, partition tolerance and tradeoffs with consistency to eventual.

Column vs Row oriented

Cassandra is not relational and it does not represent data structures in sparse multi dimensional tables. As a result it is not row or column oriented. It is up to developers to choose either of the strategies, imagining Cassandra is just using a multi dimensional table. We will illustrate it more in the design section.
3. Design

In this section we describe the data model and the architecture and security of Cassandra. The data is modeled in the flowing way.

Data model:
Every row is identified by a unique key. The key is a string, an instance of Cassandra has one table which is made up of one or more column families as defined by the user. The number of column families and the name of each of the above must be fixed at the time the cluster is started. There is no limitation the number of column families but it is expected that there would be a few of these. Each column family can contain one of two structures: super columns or columns. Both of these are dynamically created and there is no limit on the number of these that can be stored in a column family. Columns are constructs that have a name, a value and a user-defined timestamp associated with them. The number of columns that can be contained in a column family is very large. Columns could be of variable number per key. For instance key K1 could have 1024 columns/super columns while key K2 could have 64 columns/super columns. “Supercolumns” are a construct that have a name, and an infinite number of columns associated with them. The number of “Supercolumns” associated with any column family could be infinite and of a variable number per key. They exhibit the same characteristics as columns.

Architecture:
Data is distributed in the cluster using Consistent Hashing based and on an order preserving Hash function. This allows performing range scans over the data for analysis as well. Cluster membership is maintained by a Gossip Style membership algorithm. Failures of the nodes within the cluster are monitored using an Accrual Style Failure Detector. High availability is achieved using replication and we actively replicate data across data centers. Since eventual consistency is the mantra of the system reads execute on the closest replica and data is repaired in the background for increased read throughput. System exhibits incremental scalability properties which can be achieved as easily as dropping nodes and having them automatically bootstrapped with data.

Gossip Style Membership:
To support decentralization and partition tolerance, Cassandra uses a gossip protocol for intra-ring communication so that each node can have state information about other nodes. The gossiper runs every second on a timer. Hinted handoff is triggered by gossip, when a node notices that a node it has hints for has just come back online. Anti-entropy, on the other hand, is a manual process; it is not triggered by gossip.

Security:
Security is not set up to the standards in Cassandra. By default, Cassandra allows any client on a network to connect to a cluster. It rather allows all clients, without requiring that they provide credentials. The security mechanism is pluggable, which means that we can easily swap out one authentication method for another, or write your own.
However, not all cloud mechanisms are in a trusted environment. The authentication between nodes should have been more robust, than just announcing their public keys.
For example, in a trusted distributed system, all nodes can merely announce their public keys and then perform encrypted transactions. But, data centers are no more private networks; hence such a strategy would not work. At the same time, verifying public keys in a completely decentralized environment using certificate chaining is going to beat the performance of the system.
On the other hand, utilizing range detection could help defend against Sybil attacks, when intrusions take place.

4. Conclusions

Cassandra is open source and is in development at Apache. The Apache Cassandra project brings together Dynamo’s fully distributed design and Bigtables Column family based data model. Cassandra is adapting to recent advances in distributed algorithms like Accural style failure detection and others.

Cassandra is proven as it is in use by Digg, Facebook, Twitter, Reddit, Rackspace, Cloudkick, Cisco. The largest production cluster has over 100 TB of data in over 150 machines. It is Fault tolerant, decentralizes and gives the control to developers to choose between synchronous and asynchronous data replication. It offers rich data model, to efficiently compute using key and value pairs. It is financially viable through elastic scalability. It is durable and supports third party applications. It is not entirely secure, which remains a open problem in distributed communication.

5. REFERENCES

[2] CAP Theorem
http://www.julianbrowne.com/article/viewer/brewers-cap-theorem