Cassandra Concepts

Table of Contents

Aim 2
Common Concepts 2
What is meant by Horizontal Scalability? 2
Horizontal scalability refers to the ability to add/remove nodes to a network. 2
Does Cassandra support transaction rollback? 2
Why adding a cache layer with RDBMS is not enough to handle the large amount of data and ensure fast access? 2
What are Drawbacks of Vertical Scalability? 2
Which type of consistency is involved in e-commerce sites? 2
Does Cassandra ensure availability when any node is removed? 2
What is elastic scalability? 2
Where to use a NoSQL database and where not to 2
High Level Cassandra Architectural Description 3
Top down structure of the various elements in Cassandra 3
High Level Cassandra Architectural Layers 4
Factors to consider when deploying the Cassandra cluster 6
How does more memory power help in tuning Cassandra? Does it help in the reads as well? 7
What is a node tool and where is it configured. What can be accomplished with a node tool? 7
Description of the Partitioners and the partitioning strategies 7
Description of the Snitches, their function and how their effect on the Cassandra configuration 7
Description of snapshots 8
Cassandra Tuning 8
Description of the decommissioning process in Cassandra 8
Is it usually good design to have multiple applications per key space? 8
Super column VS composite column 8
Index discussion 9

Aim

The aim of this document is to summarise some important concepts relating to the Cassandra database (NoSQL). This document does not review the limitations of the Cassandra solution (for example the usage of numerous indexes large volume databases). It is only meant as an introduction to the main concepts.

Common Concepts

What is meant by Horizontal Scalability?

Horizontal scalability refers to the ability to add/remove nodes to a network.

Does Cassandra support transaction rollback?

By design, Cassandra values availability and partition tolerance over consistency. Therefore transaction rollback is not supported.

Why adding a cache layer with RDBMS is not enough to handle the large amount of data and ensure fast access?

The main drawback of increasing scalability in this fashion is that the number of write operations increase significantly. The later bock the read operations, and therefore deteriorates the performance of the system.

What are Drawbacks of Vertical Scalability?

Vertical scalability is limited by the maximum memory and CPU capability available on a server. Once the limit is reached, no more performance gain can be achieved. Therefore, increasing the horizontal scalability is required (i.e. adding more nodes to the network).

Which type of consistency is involved in e-commerce sites?

Consistency is not a major component of e-commerce sites, availability and partition-tolerance are the most important factors.

Does Cassandra ensure availability when any node is removed?

Yes. Cassandra supports the concept of elastic scalability, i.e. another node will pick up the work.

What is elastic scalability?

Elastic scalability refers to the ability to scale horizontally up or down (i.e. adding/removing more nodes to a network) without major impact on the availability of the system running.

Where to use a NoSQL database and where not to

Where to use NoSQL dB	**Where not to use NoSQL dB**
Need to store and search Big Data	Need for secondary indexes
Need to very velocity of Random Reads and Writes	Need for relational data
Need for sparse / wide column requirements	Records that needs transactional consistency at heart
No need for multiple secondary index	Need for transaction rollback
Examples	
Ecommerce Inventory use cases	Financial Trade Booking system
Times Series / Events use cases	
Feed based activities use cases	

High Level Cassandra Architectural Description

Top down structure of the various elements in Cassandra

- When data is inserted into a keyspace, it will be stored in one of the node in the cluster, which will in turn be replicated, based on the replication factor.

- Replication can take two forms: the 'Simple Replication Strategy' or the 'Network Replication Strategy'

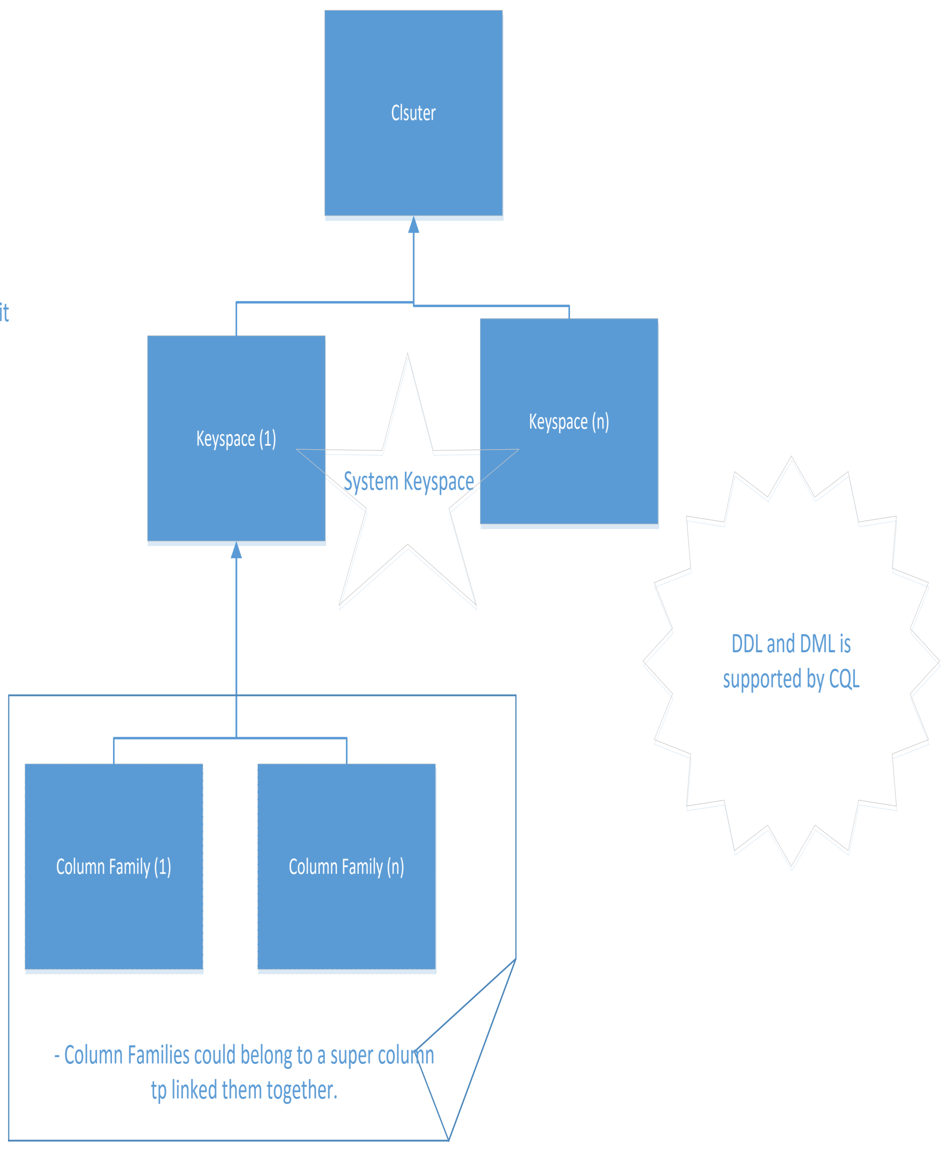
- Each column supports a primary index (on Key Row), and an optional secondary index.

- There are two types of Column Families: the 'Static' and 'Dynamic' column Family.

- Each column contains the following: 'Name', 'Type' (aka 'Comparator'), 'Timestamp' The 'Name' datatype is called a 'Validator'

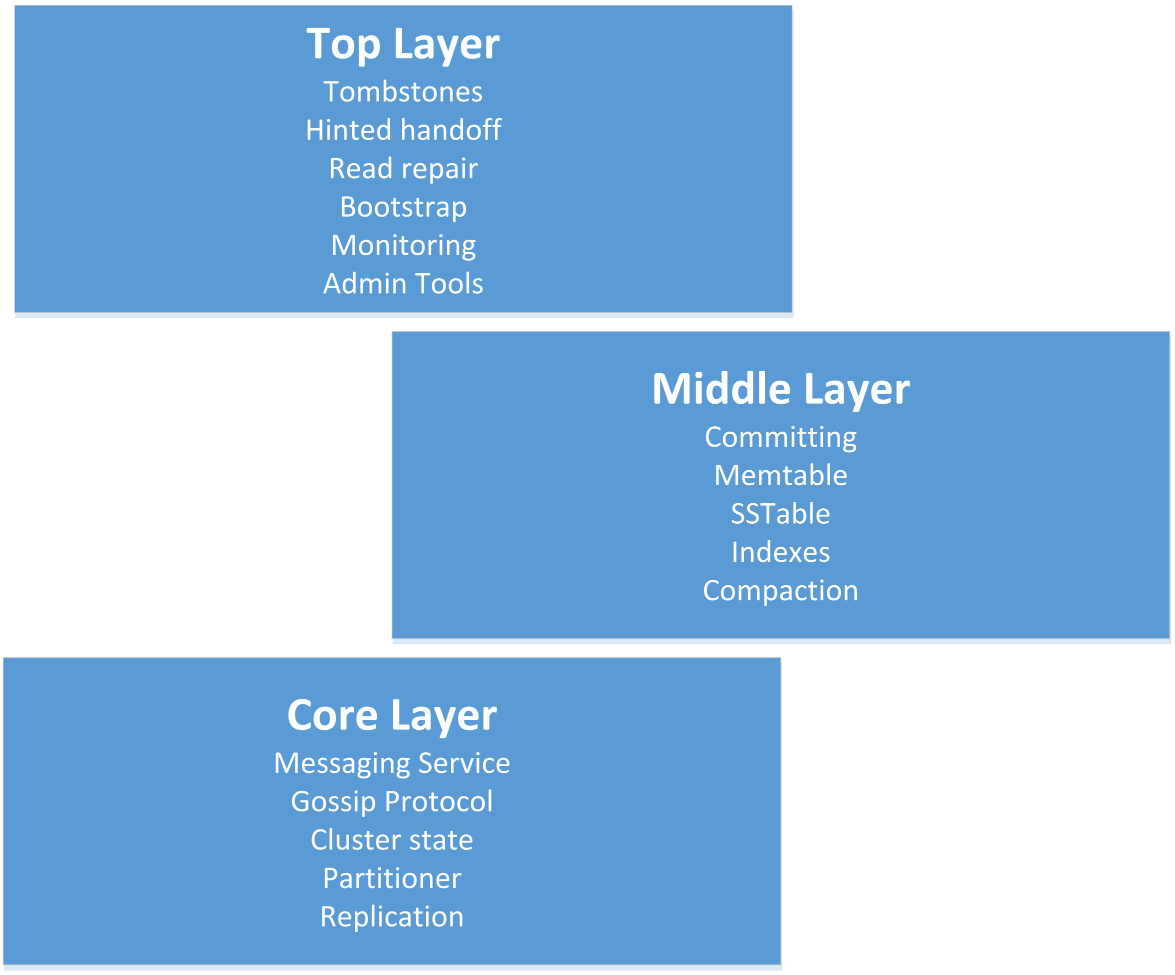
- Columns could be set as 'Expiring' or 'Counter' columns.

- Column Families could be aggregated into a composite key to form a unique column key.



- Column Families could belong to a super column tp linked them together.

High Level Cassandra Architectural Layers



Top Layer	
Tombstones	This is a ‘soft’ deletion mechanism. It sets deletion marker so that a ‘hard’ delete can planned later on.
Hinted handoff	This is the coordinator node. It ‘knows’ about ‘dead’ nodes and prevent requests to be sent to them. This mechanism is also used internally to Cassandra to capture and maintain data for of failed node, so that other nodes can pick-up these unattended requests.
Read repair	Read repair is an important component of keeping data consistent in a Cassandra cluster. Every time a read request occurs, it provides an opportunity for consistency improvement. Replicas are compared. When digest differs a repair is requested to synchronise all replicas.
Bootstrap	Bootstrapping is the process in which a newly-joining node gets the required data from the neighbours in the ring.
Monitoring	Support for monitoring is done via JMX. It provides information on low available memory, thread information, application uptime and class path, etc.
Admin Tools	It helps with the Cassandra database administration, such as access permission.

Middle Layer	
Committing	A commit log is a crash and recovery mechanism that supports Cassandra durability. When a node crashes, and the data contained in its MemTable (i.e. in-memory) has not been written to disk yet, the data can be recovered from the recover the commit log.
MemTable	This is an in-memory table that always gets the latest data. The data comes from the writes and commit log. The data is then flushed to disk for persistence (on a timer). Keeping the data in-memory improves reads performance.
SSTable	SSTable are memory space stored on the disk to ensure durably. Reads can get data from the SSTable or the MemTable.
Indexes	A mechanism which enables filtering on records.
Compaction	This is the process of freeing up space by merging large accumulated data files.
Core Layer	
Messaging Service	A service that supports the gossip message communication between nodes.
Gossip Protocol	This is used for intra-ring communication, so that each node 'knows' the state of all the other nodes in the ring.
Cluster state	It indicates which nodes are 'dead' and 'live'.
Partitioner	It takes care of the load balance of data across the nodes.
Replication	It enables the replication of data across nodes. The replication factor describes how many copies of the data exist.

Factors to consider when deploying the Cassandra cluster

The several factors to consider when deploying a Cassandra cluster:

- **Capacity planning**. This enables to establish the usable disk space. A rule of thumb provides the 'Usable Disk Space' as follows:
  - Usable Disk Space = (Disk Size * Number of Disks) * Formatting Overhead * Compaction and Repair Disk Discount Rate.
  - Formatting Overhead is usually set to 10%
  - Compaction and Repair Disk Discount Rate is usually set to 50%.
- **Hardware selection**. It consists of selecting:
  - The network. As Cassandra is a distributed data store it puts load on the network. It should have reliable and redundant network interfaces. It should handle traffic between nodes without bottlenecks.
  - The number of CPU per box – The minimum required for Cassandra is 8 CPU.
  - The memory per box. More memory means a more performant system, e.g. the larger the MemTables, the lesser the need for flushing to SSTables.
  - The disk space per box. The larger the better, as it directly impacts on the Usable Disk Space.
- **Keyspace and replication options**.
  - Keyspace: they are used to group Column Families together. Keyspace has a name and set of attributes that define the Keyspace-wide behaviour.
  - There is a choice of two replication strategies :
    - The Simple Strategy. It uses Simple Single Datacentre Clusters. It places the first replica on a node determined by the Partitioners.Additional Replicas are placed on the next nodes in a clockwise fashion in the ring without consideration for data centres or location.
    - The Network Topology Strategy that assigns a new node based on:
      - The partition strategy (see below for more information)
      - It finds a different rack for copying the data. In case there is not available rack, then the copy the data to the node in the same cluster (clockwise).
- **Node Configuration**. The following elements needs to be set:
  - The Storage settings. Depending on the installation type, e.g. development or production cluster deployment, the path for the data it manages might need to be altered (/var/lib/Cassandra). Furthermore, the 'commitlog_directory' needs to be changed, so that it is on a different disk device than the 'data_file_directories'.
  - The Partitioner settings. The Partitioner defines how the data is distributed across the nodes. In other words, it takes care of the load balance between nodes. There are two possible settings:
    - Random: the data is evenly distributed across the nodes. The advantage is that there is no potential for a node hotspot. The drawback is that it is not efficient in case of range query, as the data might be distributed on different nodes and therefore the querying requires more I/O.
    - Byte Order: the token used for data distributed are generated manually. Therefore, the data distribution is not balanced. This creates a potential for node hotspot and the cluster performance degradation. However, it maximise the efficient of range queries. This option is not recommended and should only apply when the data is well understood.
  - The Snitch settings. It is used to determine the relative host proximity by gathering information about the network topology, so that Cassandra can effectively route requests. There are four type of snitches (Simple snitch / Rack inferring snitch / Property file snitch / EC3 snitch)
  - The Gossip settings. It consists of five elements:
    - Listen address: the IP address or hostname that other Cassandra nodes use to connect
    - Cluster name: the name of the cluster node.
    - Initial token: it is used to determine the range of data the node is responsible for
    - Storage port: the intra-node communication port (default to 7000). It should be the same for every node in the cluster.
    - Seeds: it is a comma-delimited list of node IP addresses used to bootstrap the gossip process. Every node should have the same list of seeds. In a multi-data centre clusters,, the seed list should include a node for each data centre.

How does more memory power help in tuning Cassandra? Does it help in the reads as well?

The more memory means i) Larger MemTables ii) Few SSTables and ii) Larger Cache size. It also helps with the read, as it reduces the amount of disk read, which is slower than memory reads.

What is a node tool and where is it configured. What can be accomplished with a node tool?

- Node Configuration Tool enables the configuration of each node in the cluster. The following elements needs to be set:
  - The Storage settings. Depending on the installation type, e.g. development or production cluster deployment, the path for the data it manages might need to be altered (/var/lib/Cassandra). Furthermore, the 'commitlog_directory' needs to be changed, so that it is on a different disk device than the 'data_file_directories'.
  - The Partitioner settings. It defines how the data is distributed across the nodes. In other words, it takes care of the load balance between nodes. There are two possible settings:
    - Random: the data is evenly distributed across the nodes. The advantage -> there is no potential for a node hotspot. The drawback -> it is not efficient in case of range query, as the data might be distributed on different nodes and therefore the querying requires more I/O.
    - Byte Order: the token used for data distributed are generated manually. Therefore, the data distribution is not balanced. This creates a potential for node hotspot and the cluster performance degradation. However, it maximise the efficient of range queries. This option is not recommended and should only apply when the data is well understood.
  - The Snitch settings. It is used to determine the relative host proximity by gathering information about the network topology, so that Cassandra can effectively route requests. There are four type of snitches (Simple snitch / Rack inferring snitch / Property file snitch / EC3 snitch)
  - The Gossip settings. It consists of five elements:
    - Listen address: the IP address or hostname that other Cassandra nodes use to connect
    - Cluster name: the name of the cluster node.
    - Initial token: it is used to determine the range of data the node is responsible for
    - Storage port: the intra-node communication port (default to 7000). It should be the same for every node in the cluster.
    - Seeds: it is a comma-delimited list of node IP addresses used to bootstrap the gossip process. Every node should have the same list of seeds. In a multi-data centre clusters,, the seed list should include a node for each data centre.

Description of the Partitioners and the partitioning strategies

- The Partitioner settings. The Partitioner defines how the data is distributed across the nodes. In other words, it takes care of the load balance between nodes. There are two possible settings:
  - Random: the data is evenly distributed across the nodes. The advantage is that there is no potential for a node hotspot. The drawback is that it is not efficient in case of range query, as the data might be distributed on different nodes and therefore the querying requires more I/O.
  - Byte Order: the token used for data distributed are generated manually. Therefore, the data distribution is not balanced. This creates a potential for node hotspot and the cluster performance degradation. However, it maximise the efficient of range queries. This option is not recommended and should only apply when the data is well understood.

Description of the Snitches, their function and how their effect on the Cassandra configuration

- The Snitch settings. It is used to determine the relative host proximity by gathering information about the network topology, so that Cassandra can effectively route requests. There are four type of snitches:
  - Simple snitch: this is the default snitch and uses a simple strategy of placing the copy of the row on the next available node walking clockwise through the nodes.
  - Rack inferring snitch: it automatically tries to place copies of rows on different racks in the data centre.
  - Property file snitch: it gives more control when using a Rack-Aware Strategy by specifying node locations in a standard key/value properties file called Cassandra-Topology-properties.
  - EC3 snitch: this is used when Cassandra is used in the cloud. It uses the AWS API to request the region and the respective availability zone.

Description of snapshots

The purpose of snapshoat is to make a copy of some or all of the keyspaces in a node and save it to what is essentially a separate database file. This is useful in case of a restore scenario (e.g. when a node is shut down and restarted), when the snapshot can be reloaded and re-sync to the latest data. It is analogous to a snapshot table in an RDBMS database.

Cassandra Tuning

- Reads Tuning –> they will be more performant with larger MemTables and larger Cache.
- Writes Tuning –> they will be more performant with a separate device for Commit Logs and concurrent writes.

Description of the decommissioning process in Cassandra

Decommissioning means removing a node from the cluster. It involves a number of steps:

- Stop the Gossiper
- Stop the Messaging Service
- Stop the SEDA Manager
- Check the node is in the state 'decommissioned'
- The storage service check what nodes available to receive the 'decommissioned' node data and assign the data ranges to the available nodes that can consume these ranges.

Is it usually good design to have multiple applications per key space?

Generally no, a key space represent a domain space, having multiple applications per domain space could create confusion.

Super column VS composite column

- A super column family is a NoSQL object that contains column families. It is a tuple (pair) that consists of a key-value pair, where the key is mapped to a value made of column families. In other words, a super column wraps column families.

An example of a super column is shown below:

```
Footwear =
{
  Sock: {
             name:{"Cotton Sock"}
             uniqueRefNumber:{124}
             priceUsd:{"10.00"}
             isAvalable:{"1"}
        }
Shoe:{
             name:{"Trainer Sock"}
             uniqueRefNumber:{123}
            priceUsd:{" 125.00"}
           isAvalable:{"1"}
       }
}
```

- In DataStax words, "composite comparators can be thought of simply as a comparator composed of several other types of comparators." For more information, please refer to 'http://www.datastax.com/dev/blog/introduction-to-composite-columns-part-1'

Super column should not be used in Cassandra due to the high performance impact. Composite column should be used instead.

Index discussion

Indexes are useful for fast retrieval of information and ordering. A secondary index should be used on the name column, so that users looking for an item can find all items relating to the name of the selected item.

A small set of interesting CQL commands

Help	Returns all available command in CQL
select columnfamily_name from system.schema_columnfamilies where keyspace_name='my_keyspace' ;	Rep--turns all families columns for the keyspace_name = 'my_keyspace'
DESCRIBE FULL (CLUSTER | SCHEMA) | KEYSPACES | (KEYSPACE keyspace_name) | TABLES | (TABLE table_name) | TYPES | (TYPE user_defined_type) | INDEX | (INDEX index_name)	It provides information about the connected Cassandra cluster, or about the data objects stored in the cluster. For more details see http://docs.datastax.com/en/cql/3.1/cql/cql_reference/describe_r.html

A few interesting web sites

Category	**Web Site**
Architecture	https://dzone.com/refcardz/apache-cassandra
Best Practice Design	http://www.ebaytechblog.com/2012/07/16/cassandra-data-modeling-best-practices-part-1/ http://www.ebaytechblog.com/2012/08/14/cassandra-data-modeling-best-practices-part-2/
CQL Tutorial	http://www.tutorialspoint.com/cassandra/cassandra_shell_commands.htm
CQL Where Clause In Depth	http://www.datastax.com/dev/blog/a-deep-look-to-the-cql-where-clause
CQL/CLI Command list	Type HELP (and hen HELP <Command>) in cqlsh
Composite Key Tutorials	http://www.planetcassandra.org/blog/composite-keys-in-apache-cassandra/ http://outworkers.com/blog/post/a-series-on-cassandra-part-2-indexes-and-keys
Hector API	https://hector-client.github.io/hector/build/html/documentation.html
Hector User Guides	https://github.com/hector-client/hector/wiki/User-Guide https://github.com/rantav/hector/wiki/Getting-started-%285-minutes%29
Hector Examples	http://www.programcreek.com/java-api-examples/index.php?api=me.prettyprint.hector.api.query.RangeSlicesQuery
Hector Slice Query options	https://irfannagoo.wordpress.com/2013/02/27/hector-slice-query-options-with-cassandra/
Hector Quick Start	http://stones333.blogspot.co.uk/2013/02/quick-start-cassandra-with-hector.html