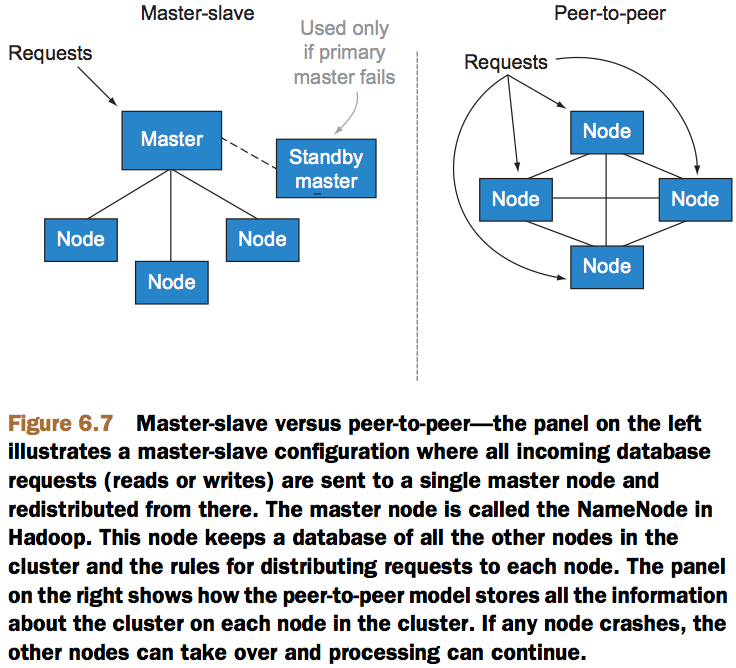
**Mention the advantage of peer to peer architecture over master slave architecture:**

HBase has a master-slave model, while Cassandra has a peer-to-peer model. I am aware that in a master-slave model, the master is a SPOF (Single Point of Failure) and there is no such thing in a peer-to-peer model.

Master-slave structure is more effective, P2P is more stable



Let’s look at the trade-offs. With a master-slave distribution model, the role of managing the cluster is done on a single master node. This node can run on specialized hardware such as RAID drives to lower the probability that it crashes. The cluster can also be configured with a standby master that’s continually updated from the master node. The challenge with this option is that it’s difficult to test the standby master without jeopardizing the health of the cluster. Failure of the standby master to take over from the master node is a real concern for high-availability operations.

Peer-to-peer systems distribute the responsibility of the master to each node in the cluster. In this situation, testing is much easier since you can remove any node in the cluster and the other nodes will continue to function. The disadvantage of peer-to-peer networks is that there’s an increased complexity and communication overhead that must occur for all nodes to be kept up to date with the cluster status.

The initial versions of Hadoop (frequently referred to as the 1.x versions) were designed to use a master-slave architecture with the NameNode of a cluster being responsible for managing the status of the cluster. NameNodes usually don’t deal with any MapReduce data themselves. Their job is to manage and distribute queries to the correct nodes on the cluster. Hadoop 2.x versions are designed to remove single points of failure from a Hadoop cluster.

Using the right distribution model will depend on your business requirements: if high availability is a concern, a peer-to-peer network might be the best solution. If you can manage your big data using batch jobs that run in off hours, then the simpler master-slave model might be best. As we move to the next section, you’ll see how MapReduce systems can be used in multiprocessor configurations to process your big data.

Similarities between Cassandra and MongoDB

Now, with the understanding of these two NoSQL databases, let us try to understand some of the similarities between these two:

* Both of these are NoSQL database types
* None of these is a replacement to the traditional RDBMS database types
* Both of these are not ACID compliant databases
* Consistency and Normalization are two concepts that these two database types not satisfy (as these lean more towards the RDBMS database types)

Cassandra Vs. MongoDB

In this section, we will take a look at the differences between Cassandra and MongoDB.

|  |  |  |
| --- | --- | --- |
| **Features** | **Cassandra** | **MongoDB** |
| Data Model | Cassandra has a more orthodox data model with rows and columns.  Data is structured in the case of Cassandra and each of these columns is of a specific type which gets assigned during the table creation itself.  In comparison, MongoDB provides more rich data model than that of Cassandra. | MongoDB has a data-oriented or an object-oriented data model.  This model can further be represented using any of the data structures based on the user domain.  Data can further be nested into multiple levels if there is a need. |
| Master Node | Cassandra has multiple master nodes in a cluster, and if one master node goes down, its place will be taken by another node.  Because of the above, there is no effect on the cluster and is always available.  In comparison, Cassandra has a higher availability over MongoDB. | MongoDB has only one master node in a cluster which further controls a set of slave nodes.  If the master goes down, a slave is elected as master and takes about 20-30 seconds for the same. In this duration, the cluster won’t be able to accept any incoming requests. |
| Secondary Indices | Cassandra has cursor support for the secondary index. This is limited to only a single column and equality comparison. | It is very easy to index any property that is stored in the MongoDB database.  MongoDB is better than Cassandra if your application requires secondary indices along with flexibility in the data model. |
| Scalability | Cassandra can have multiple master nodes in a cluster which makes it ideal in the case of Scalability.  Cassandra is more scalable in comparison with MongoDB as it can have more than one master node in a cluster. | MongoDB has only one master node in the cluster at any given point in time, which is the only point to cater incoming requests. Hence, it is not ideal when we think about scalability. |
| Query Language | There is a proprietary query language for Cassandra named CQL, which is very similar to SQL.  Cassandra has a user-friendly set of queries with CQL and is adaptable within the developers who have prior knowledge of SQL. | There is no support for any query language for MongoDB.  Queries are structured as JSON fragments in MongoDB. |
| Aggregation | Cassandra doesn’t have any built-in support for aggregation and heavily relies on tools like Hadoop or Apache Spark | MongoDB has built-in support for aggregation which can be used to run an ETL pipeline in transforming the required data.  MongoDB’s aggregation framework supports both small and medium data traffic. With the increased complexity, the framework gets tougher to debug as well.  MongoDB is better in comparison with Cassandra, as it has a built-in aggregation framework. |
| Schema | Cassandra doesn’t provide the facility to alter schema but provides static typing. | MongoDB provides the facility to alter schema for the Users |
| Performance | Cassandra performs better in applications with heavy data load as it can provide multiple master nodes in a cluster. | MongoDB is not ideal for applications with heavy data load as it can’t scale with the performance. |

**On what does the task tracker run?**

**Ans:** The **task tracker is** the one that actually **runs** the **task** on the data node. Job**tracker will** pass the information to the **task tracker** and the **task tracker will run**the job on the data node. Once the job **has** been assigned to the **task tracker**, there**is** a heartbeat associated with each **task tracker** and job **tracker**

**What are the key value pairs incase of keyvalueTextInputFormat?**

KeyValueTextInputFormat breaks the line itself into key and value by a tab character (‘/t’). Here Key is everything up to the tab character while the value is the remaining part of the line after tab character.

4 Some Limitations of RDF

Ground Assertions

• RDF allows us to make factual statements (assertions).

• These statements are always about individual objects.

• We can’t say things like the following:

– Every page was created by some person.

– Dogs are mammals.

– Dogs bark.

7

No Semantic Constraints

• By itself, RDF places no semantic restrictions on how predicates combine with subjects and objects.

• Even worse, RDF has no way of telling which URIs can semantically act as predicates!

Anomalous Statements

infcourses:masws terms:knows edstaff:104599

meals:lunch06 terms:homepage dc:title

mailto:kim@wanna.be edstaff:104599 ‘chicken’

RDF Vocabulary Definition

• RDF has been extended with mechanisms to allow new vocabularies to be defined.

• Resulting language known as RDF Schema (RDFS; cf. http://www.w3.org/TR/rdf-schema/)

• Basic idea is to allow statements like the following:

Example RDFS Constraints

The subject of ‘birthday’ must be an Agent.

The object of ‘homepage’ must be a Document.

Every instance of Person is an instance of Agent.

• These vocabularies, and similar sets of definitions, are frequently called ontologies.