**Lecture 9 Notes:**  
  
**Different Type of databases and there comparisons:**

**Key-Value Stores:**  
  
What it provides?  
  
Key-Value Store is effectively an associative array stored on a disk; it is a single key lookup, a dictionary so to speak.

Pro and Con:  
The good thing about these databases is that they can be read very quickly, but these databases are not so good for reverse lookups or additional analytics.

Example:  
An example of this type of database is Redis.

**Column Stores:**

Column Stores is the subset of NoSQL databases that kept, somewhat, to the tabular form.

So, what does this mean?

**Column Stores vs Relational Databases**

**Relational Database Approach**

Well, as you probably know, relational databases keep all their data in tabular form (where every row represents one entity). Since every row is saved separately on the disk, we could say that rows align the data.

When reading this kind of database, it always reads the whole row, even if not all of the data is necessary (i.e., if we only want one column of values).

**Column Store Approach**

Column stores, on the other hand, change this approach a bit: they store data in so-called columned families (i.e., in column order).

For example:

First, Ids of all records are saved.

Then, all of their names, etc.

**Why is this a big deal?**

This is a big deal because it is possible to get the whole column in a more efficient manner than it was when you got all of the rows and had to pull specific values from each one

Basically, we can get more information from the database in a single seek. Also, these databases can be easily compressed and, it goes without saying that writes are very expensive.

Example:

A typical example of these databases is Cassandra.

**Graph Stores:**Graph stores use graph structures for queries, with nodes, edges, and properties, to represent and store data.

They are used for storing a network of connections or relationships (e.g., social networks).

Graph stores are a bit different from other NoSQL databases since they originated from a different problem with relational databases–they have a number of small records with a lot of relationships between them.

An example of such a database is AllegroGraph, GraphQL

**Document Stores:**

One of the most popular types of NoSQL databases is Document stores, which revolves around the concept of a document.

Documents are self-describing structures and usually similar to each other, but they don’t have to be the same.

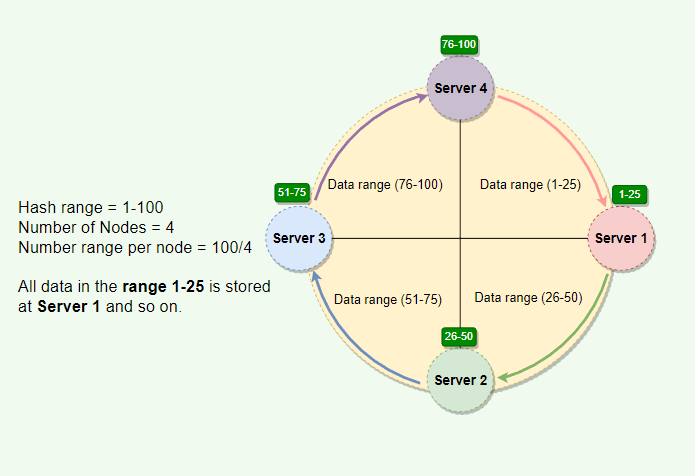
Unlike the rows in relational databases, where every row has to follow the same schema, documents can vary from each other and still belong to the same collection.

Example:

MongoDB and Couchbase are examples of document stores.

**Consistent Hashing:**

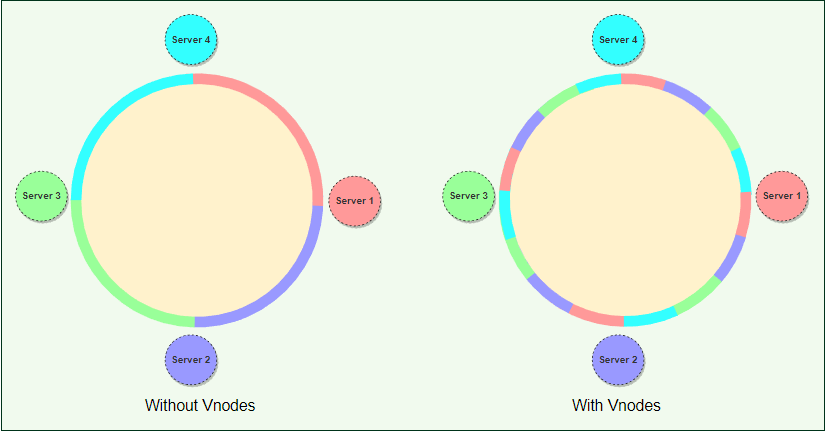
Use the Consistent Hashing algorithm to distribute data across nodes. Consistent Hashing maps data to physical nodes and ensures that only a small set of keys move when servers are added or removed.

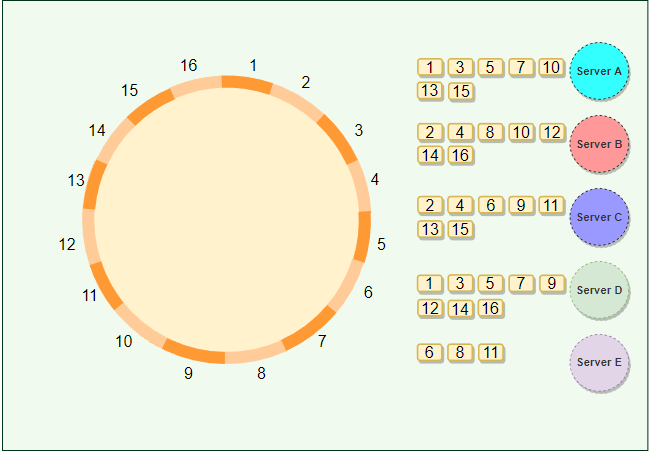
  
  
Issues with basic consistent Hashing:

Adding or removing nodes: Adding or removing nodes will result in recomputing the tokens causing a significant administrative overhead for a large cluster.

Hotspots: Since each node is assigned one large range, if the data is not evenly distributed, some nodes can become hotspots.

Node rebuilding: Since each node’s data might be replicated (for fault-tolerance) on a fixed number of other nodes, when we need to rebuild a node, only its replica nodes can provide the data. This puts a lot of pressure on the replica nodes and can lead to service degradation.





**Advantages of Vnodes**

Vnodes gives the following advantages:

* As Vnodes help spread the load more evenly across the physical nodes on the cluster by dividing the hash ranges into smaller subranges, this speeds up the rebalancing process after adding or removing nodes. When a new node is added, it receives many Vnodes from the existing nodes to maintain a balanced cluster. Similarly, when a node needs to be rebuilt, instead of getting data from a fixed number of replicas, many nodes participate in the rebuild process.
* Vnodes make it easier to maintain a cluster containing heterogeneous machines. This means, with Vnodes, we can assign a high number of sub-ranges to a powerful server and a lower number of sub-ranges to a less powerful server.
* In contrast to one big range, since Vnodes help assign smaller ranges to each physical node, this decreases the probability of hotspots.