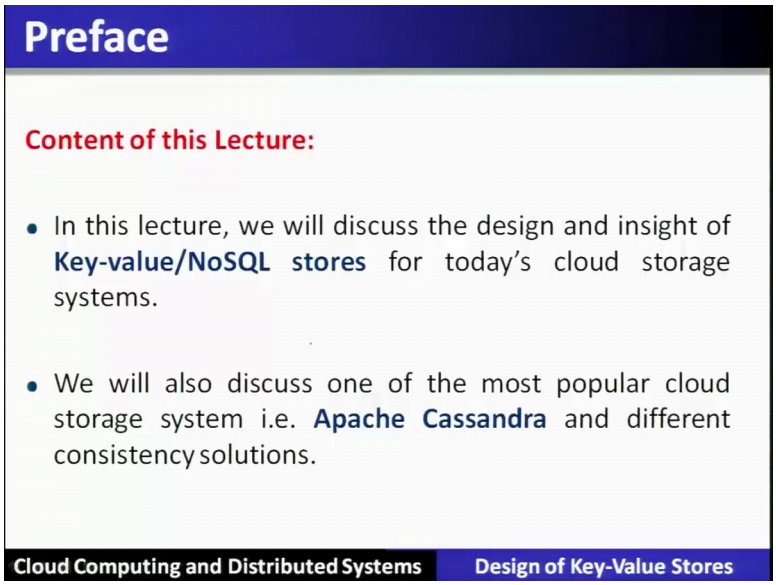
Cloud Computing and Distributed Systems
Dr. Rajiv Misra
Department of Computer Science and Engineering
Indian Institute of Technology, Patna

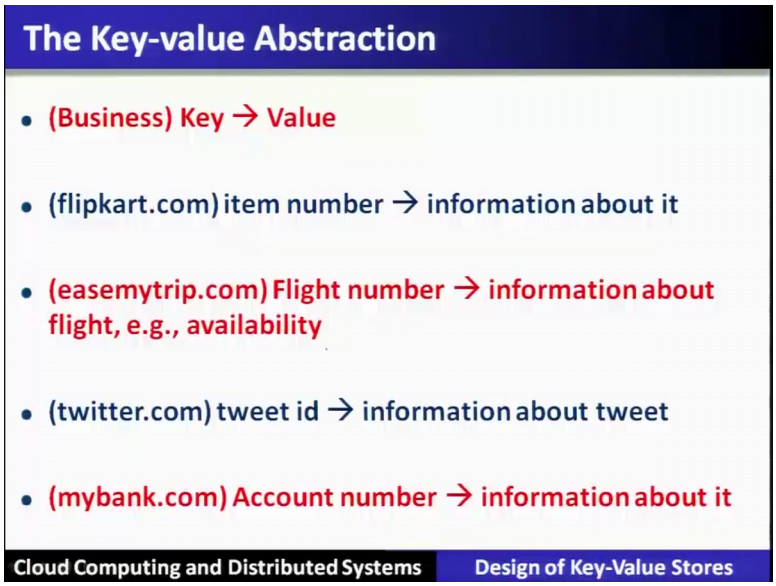
Lecture – 16
Design of Key-Value Stores

(Refer Slide Time: 00:16)



Design of Key Value Stores; preface content of this lecture, we will discuss the design and under the (Refer Time: 00:22) of new generation storage that is called key value store or it is also called as a NoSQL store, which is being provided as a services by the cloud storage system. We will also discuss one of the popular storage system provided by the cloud providers which uses underlying the key value store that is called Cassandra; we will also cover different consistency solutions.

(Refer Slide Time: 01:07)



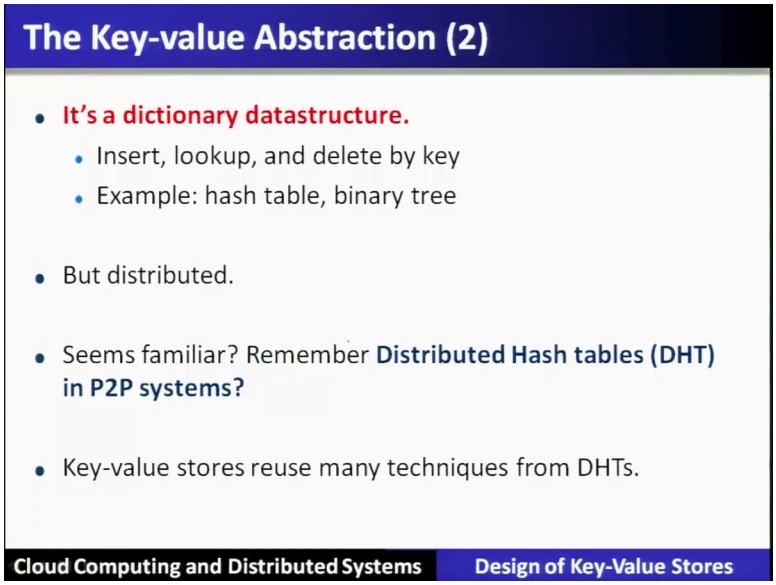
So, as I told you that the new generation storage system which are being provided by most of the cloud systems are now providing the abstraction of a key value store. So, let us understand what do you mean by key value store. So, key value store can be understood by different services which are running on the internet. For example, the Flipkart used to sell the items; every item is having a particular unique number that is called item number. And this item number has various information such as the product cost, name of the product, manufacturer, who purchased rate and how much quantity is available.

Item number becomes the key and all the information related to that item becomes the value. Similarly, if let us say we want to book a flight ticket through some service then the flight number will become the key and related to that flight number that information about the flight and the availability of seat and other information becomes the value.

Similarly, the Twitter; tweet every tweet is having an id call tweet id and tweet id is having the information about the tweet all the details about that particular tweet who send it and so on. Similarly, online banking provides the information in the form of the account number. So, account number becomes a key and who owns that account, how much money is there in the account and other details are the value. So, every business here deals with this a key and the value kind of notion; hence we will see that key value

abstraction is very much required in different businesses. And we will see how this abstraction is provided in a new generation storage system to using the cloud.

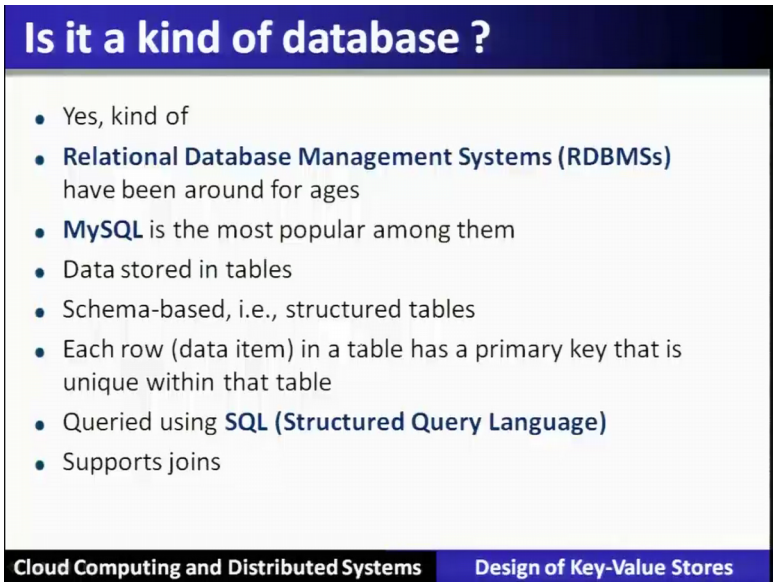
(Refer Slide Time: 03:43)



Key value abstraction though looks very simple can be implemented using the traditional approaches like dictionary data structures which supports insert, lookup, delete by the key; for example, hash table and binary tree. But the drawback or the problem with that is scheme is that it can run on single server machine as a process.

So, for a small data set this all was workable, but when the data becomes huge then the single server will not work; hence the distributed system is required to manage the key value store that is and also large number of clients who are accessing this huge amount of information for that the distributed system is required. So, instead of hash table the concept of similar to the hash table is being supported in distributed hash table in pear to pear systems that we have already seen can be used in the designing of such key value store.

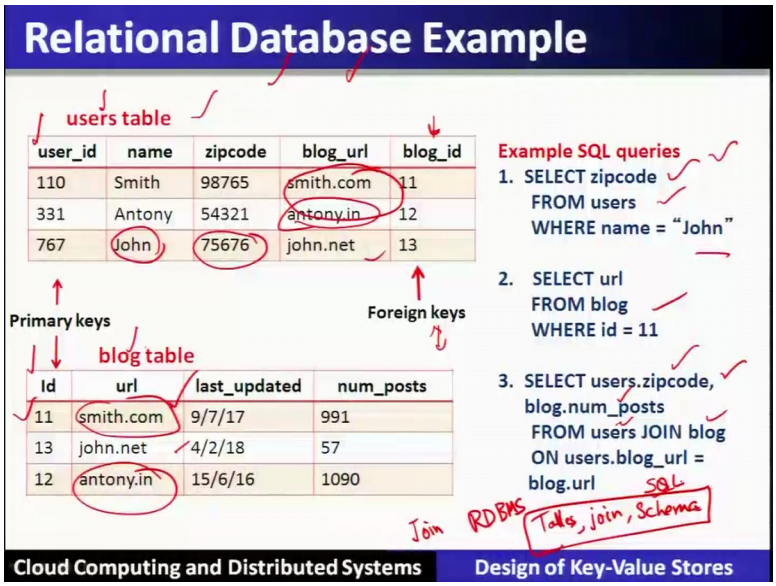
(Refer Slide Time: 05:00)



As far as key value store let us see whether is it a kind of database. Yes it is a kind of data base let us understand what is let us recall; what is the RDBMS: Relational Data Base Management System provided the language to access the data through the keys called MySQL and in RDBMS data is stored in the form of tables; which follows the schema that is the these tables are structured.

And every row is called a data item has a primary key that will uniquely identify a row or a tuple in that particular table. This kind of database supports the operations called join across multiple tables to fetch the data from it.

(Refer Slide Time: 06:12)

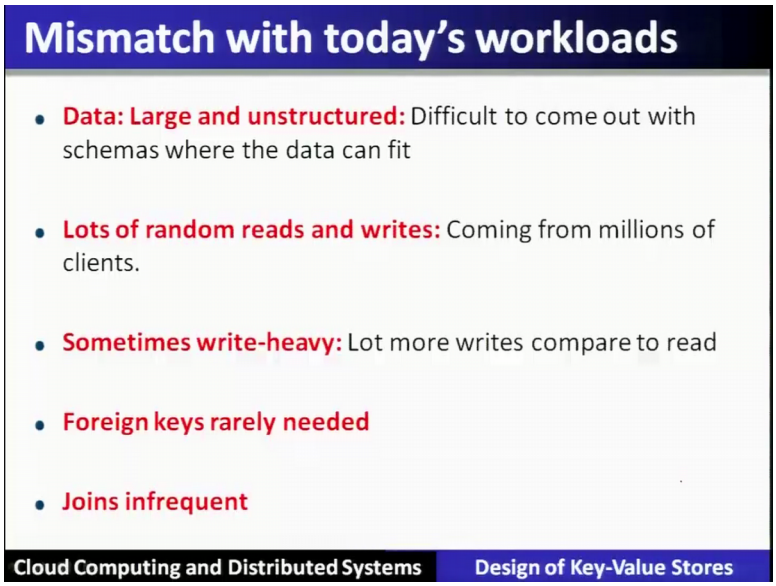


Let us see this particular example here; we have shown 2 different tables one is the users table the other is the blog table in the RDBMS format. The user table has the primary key as user id and the blog table has the primary key as the id in the blog table. This primary key a blog table is also being provided as blog id in the user table hence this is called a foreign key in the terms of RDBMS. Now you can generate the SQL queries for example, the query comprising of one table is shown over here. So, the name field which is having the John in this table user will print the zip code which is being queried by a customer. Similarly, from the blog table the id whose value is 11 his url will be printed.

Now comes a query which requires 2 different tables user and a blog to be joined where in the user blog url is equal to the blog url. So, here we can see this particular entry is common this is also common and so also this common. So, they will join using Cartesian product and then it will find out the users zip code and the blog num posts.

So, this is called a join operation; if 2 different tables are required for a Cartesian product and then it can basically satisfy the query and supports it. So, in this relational database we have seen the tables, we have seen the join operation, we have seen the schema which is being provided by RDBMS. And on top of it the SQL language which will basically allow us to do these operations on the database.

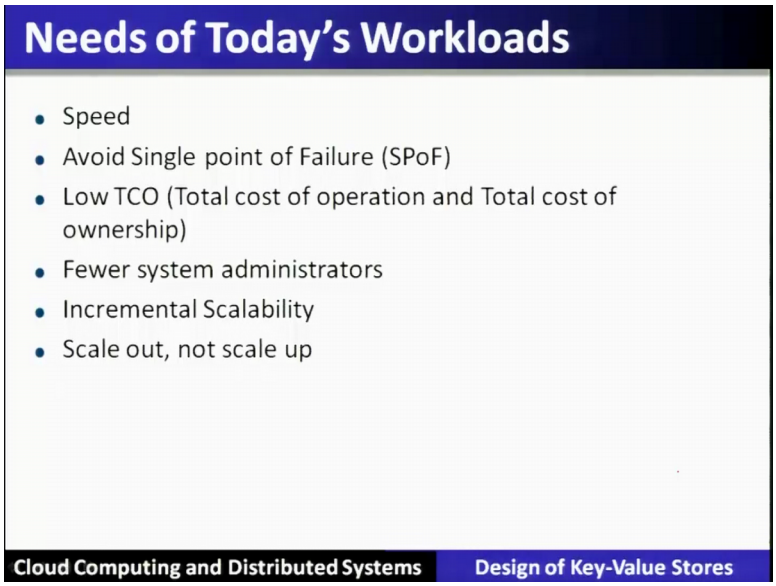
(Refer Slide Time: 09:10)



However, in the earlier times that was all fine to involve the database and serve a particular organization, but now today's workload is across the globe. Hence the workload is quite heavy to be tackled by RDBMS; there are different requirements of in today's work load. For example, data is very large and cannot be structured that is called unstructured; that means, it cannot follow any schemas.

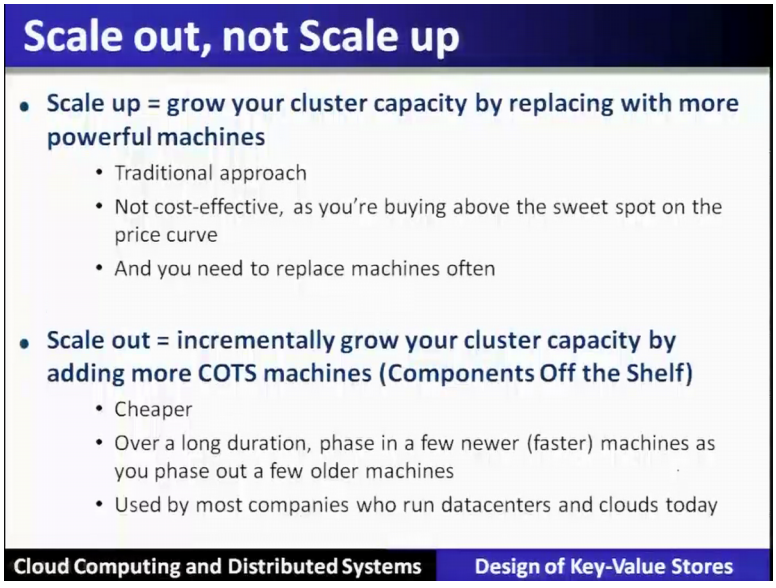
The second mismatch in today's workload is that lot of random reads and writes are required. So, it is a heavy reads and heavy write operations are being supported compared to the read. So, heavy write operations are supported sometimes and foreign key is rarely required and join is very infrequent in today's workload.

(Refer Slide Time: 10:24)



So, to deal with this workload these are the following characteristics which are required to be handled in the modern storage system; that is the speed single point of failure to be avoided total cost of operation. And a total cost of ownership should be low fewer system administration and incremental scalability and scale out not the scaling up.

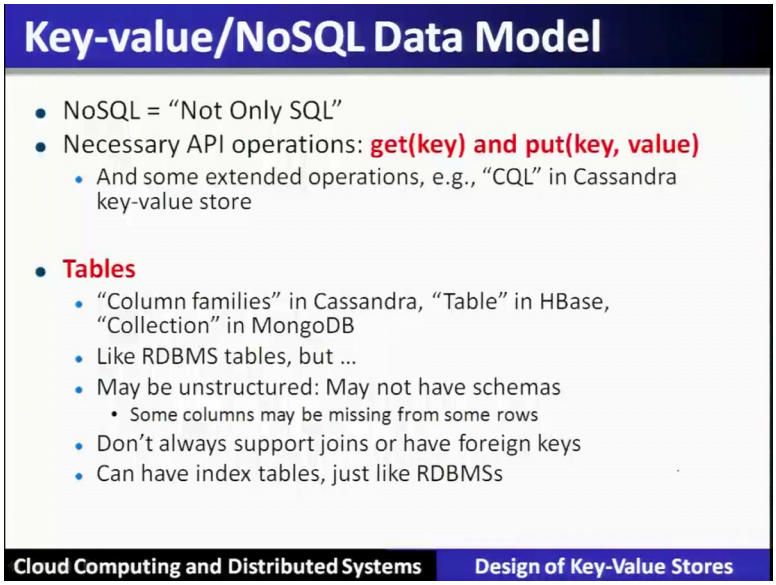
(Refer Slide Time: 10:56)



So, scale out means that we can incrementally grow your data center or your cluster by adding more machines; that becomes the cheaper way of managing the data center. On

the other hand, earlier times the entire cluster capacity was replaced by with more machines that is called scale of that is not required these days in the data center scenario.

(Refer Slide Time: 11:30)

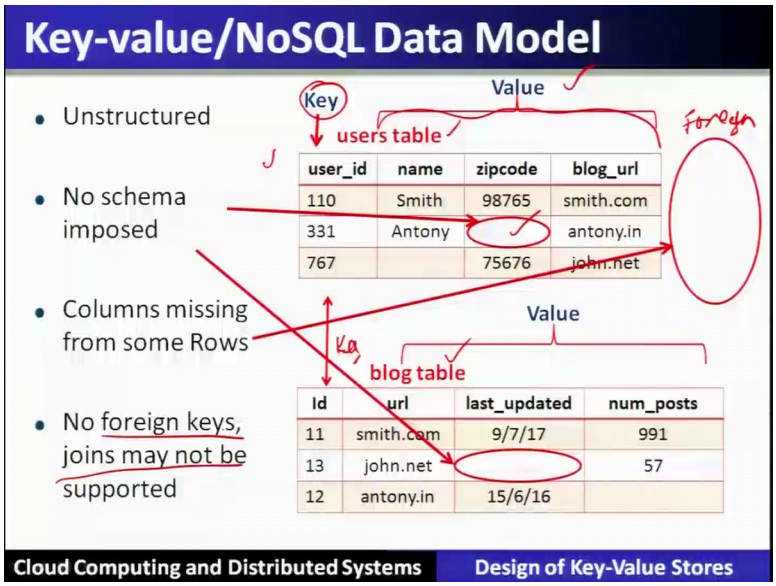


So, let us see that the solution here is the key value abstraction; key value is store which is also called as a no SQL data model. So, no SQL is not only SQL or not only SQL which provides the operations in the form of APIs that is the get by a key and a put the key and a value.

So; that means, get with the key; that means, it will fetch the value of the key which is maintained by the data store. Put the key value means that it is going to be updated with the new values using the put operation which is supported by an API. So, as far as the new generation data is store such as Cassandra provides a similar kind of language which is similar to SQL, but not exactly SQL it is called Cassandra query language which will allow to use the key value store as per as tables are concerned here the tables are called column families in Cassandra. But it is called table in Hbase it is called collection in MongoDB that is an NoSQL data storage.

These terms are now redefined now these particular tables are unstructured; that means, they will not follow any schema; that means, that is all the columns may be missing from some of the rows and so on. And, also this kind of NoSQL database do not supports join operations or do not have the foreign key concept they can have the index tables just like RDBMS.

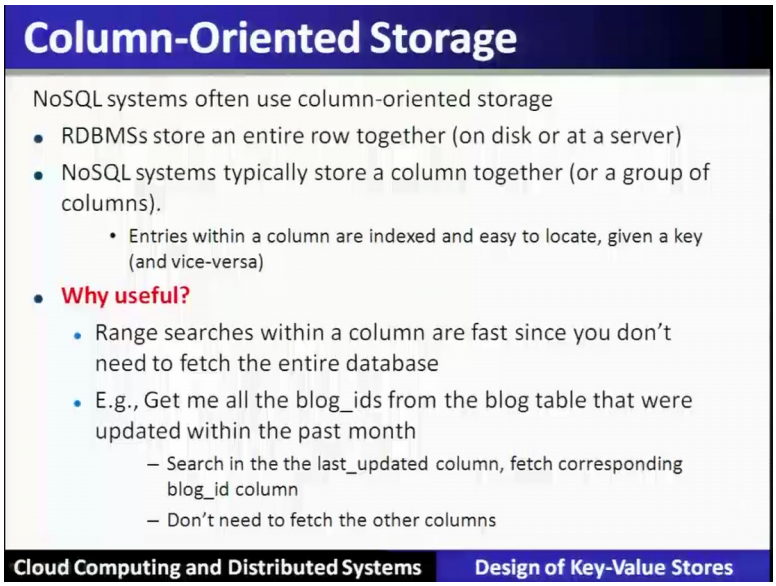
(Refer Slide Time: 13:47)



So, let us see these differences with RDBMS its highly unstructured database for example, here although these data stores are called tables. For example, the same user table if we consider user id will be the key and the remaining fields are called value. Similarly in another table that is a blog table id is called a key and rest are all called values.

This is highly unstructured in the sense it is not following the schema some of the column values are missing from some other row that is fine. And there is no foreign key which is being supported here, since there is no foreign key has the joins are also not supported.

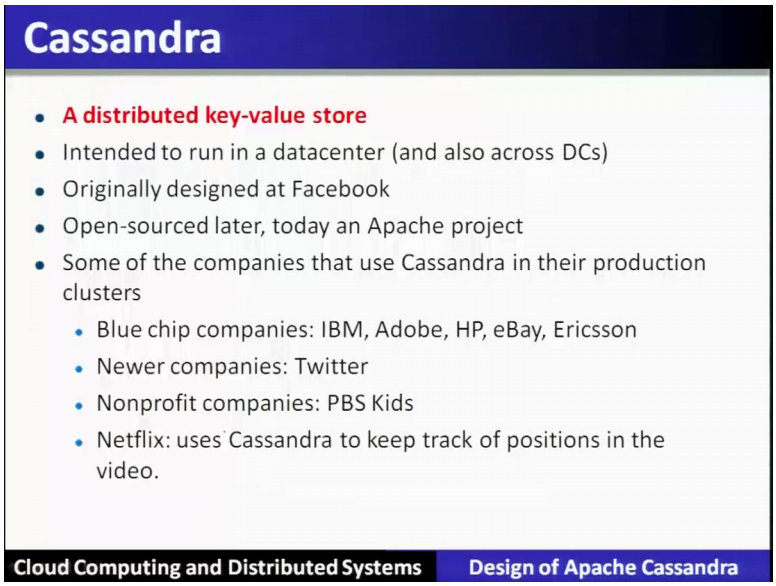
(Refer Slide Time: 14:52)



This particular new generation storage is also called as a column oriented storage unlike in RDBMS which is stores the entire rows together on a disc or a server NoSQL; NoSQL systems typically store the column together. So, the entries within the particular column are indexed and easy to locate given a key and vice versa. This is useful to support the range queries within a particular column, since they are fast and do not need to face the entire database for the retrieval of data for that query.

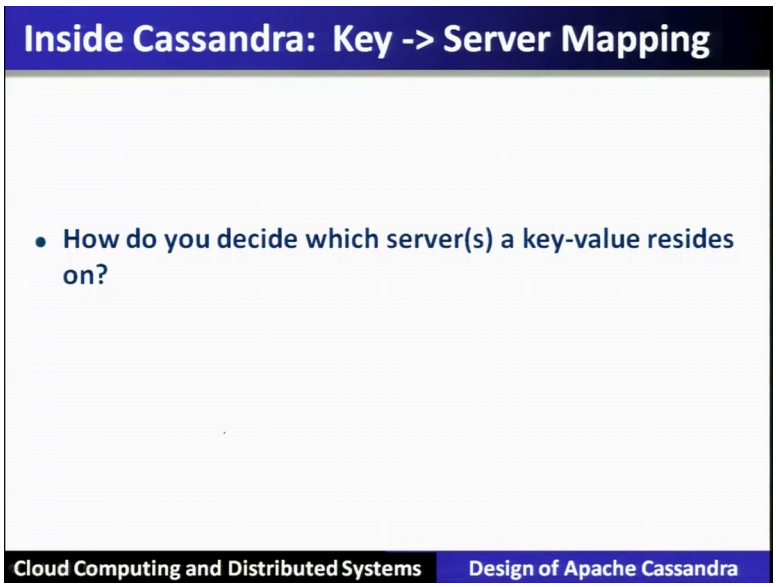
Let us go and see the design of one such system that is called apache Cassandra, which supports the new type of storage that is called key value storage or it is also called is a NoSQL. So, Cassandra is a key value store which is supported by the distributed system.

(Refer Slide Time: 16:04)



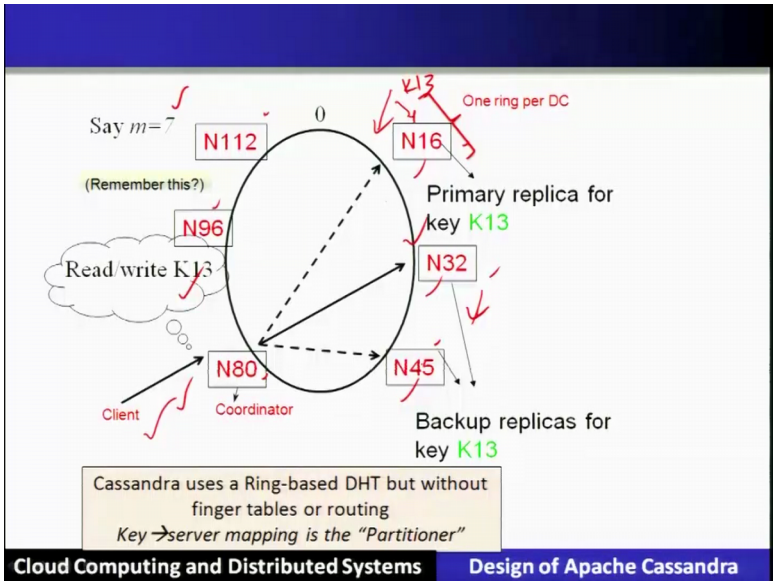
That is called distributed key value store; obviously, it runs on the data center not on a single server Cassandra was originally designed at the Facebook and was later very popular and open source did by the Apache project. And many companies' uses Cassandra in their production clusters such as IBM, HP, Ericsson, Twitter, PBS kids, Netflix and so on.

(Refer Slide Time: 16:39)



Let us go and see inside the Cassandra. So, the first task we will look into that how the keys are to be mapped on the server, which will store those key and their corresponding value that is key value store how that is being stored in which server that mapping.

(Refer Slide Time: 17:09)



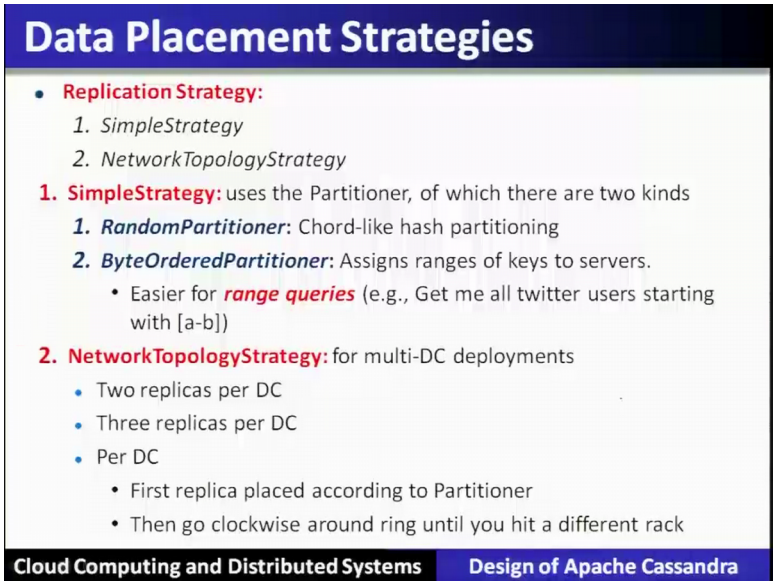
Cassandra uses the concept of a ring which is being provided by the chord distributed hash table. So, it uses chord like ring based distributed hash table to map the servers and the keys for key value storage, but with a difference that it does not use finger tables or it does not use routing and the key to the server mapping is done by the partitioner.

So, here for example, this is the ring which we have already seen defined for m is equal to 7; these are the different nodes which are mapped on the ring. As far as the client is concerned it want to read and write a key let us say 13. So, it will be done through the partitioner maybe for per customer there is a partitioner here. So, this particular key K 13 will be hashed that is using shravan function and it will try to map here. So, as per as this kind of storage is concerned; so, this particular will be map to the next nearest server or the node.

So, N 16 will be storing it, but it will also store the replications of it. So, the second replication will be stored on the next server; that is N 32 and the next server will be that and 40 5 will be storing up all are called replicas that is K 13 will be now maintained at 3 different servers consecutive services in Cassandra.

So, that was the key to the server mapping and this not only stores a particular key, but also maintains the replicas also here in this particular way.

(Refer Slide Time: 19:53)



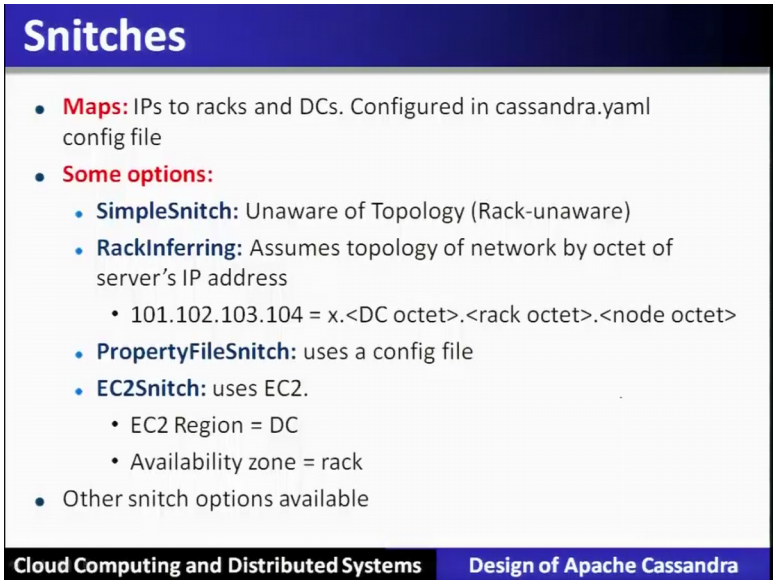
Now this particular data placement follows some strategies let us see what are the strategies which are supported in Cassandra for data placement. There are two strategies which are supported for data placement in Cassandra one is called simple strategy, the other one is called network topology strategy. Simple strategy uses the partitioner as we have seen earlier in the slide. So, simple strategy using the partitioner of which there are two different kind of partitioner here which is being applied.

So, the first kind of partitioner is called random partitioner that is nothing, but it is a chord like hash partitioning we have seen in the previous slide. This particular way that is called a random partitioner why random? Because, distributed hash table hashing using hashing it basically stores the data. The second kind of strategy of the partitioner or second type of partitioner is called byte ordered partitioner it assigns the range of keys to the server.

So, range of keys to the servers means that these range of keys are stored at one place and will become easier for the range queries; so that was the simple strategy for that replication. The second replication strategy for the data placement is called network topology strategy; that is meant for multi data center deployments. So, it says that two replicas per data center or three replica sometimes per data center.

When it calls per data center that is first replica will be placed according to the partitioner that we have seen in the simple strategy. And then it will go a clock wise around the ring until you find another server on a different rack. So, rack wise replication is done to support the rack failure that is called as network topology strategy is nowadays very much useful in the cloud scenario. So, both strategies are supporting the replication. So, there are 2 kinds of application we have covered one is called simple strategy; the other one is called network topology strategy.

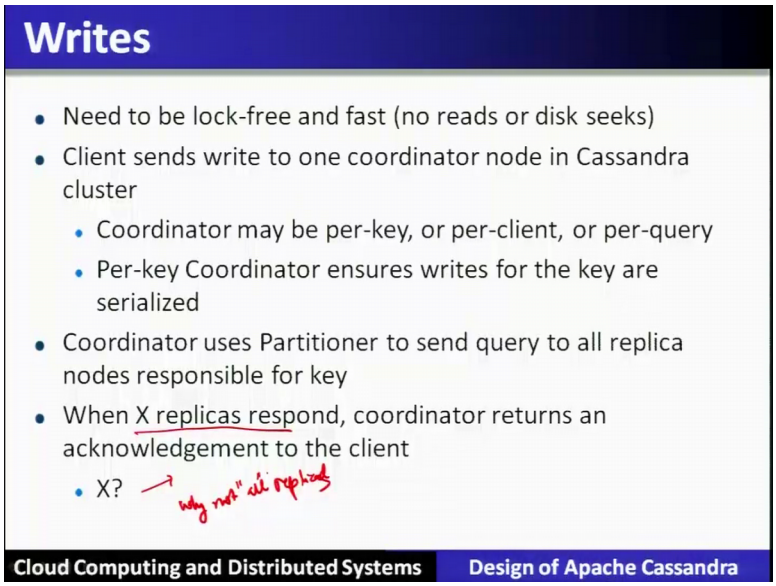
(Refer Slide Time: 22:34)



Now, there are snitches; that means, when IPs are mapped to the racks and data centers. These particular configurations are stored in cassandra dot yaml configuration files. So, there are some options for mapping first is called simply snitch. So, this particular strategy is unaware of the topology; that means, IPs when their map we will not know from that mapping which of the racks or a data center the IP is mapped. The second one is called rack inferring will assumes the topology of the network by updates of servers IP addresses.

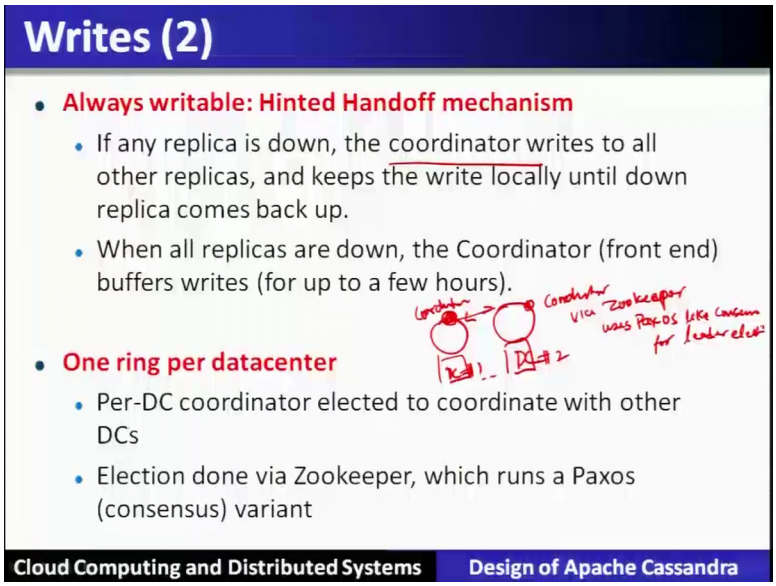
So; that means, by looking up using rack infer we can know that IP is mapping to which of the racks on which of the data centers and so on. Third property is called property files snitch uses a configuration file and easy to snitch uses the easy to that is easy to regions which data center and also the availability zone which rack and so on. Let us see the write operations on this kind of store.

(Refer Slide Time: 24:09)



So, it need to be log free and fast for write operations. So, the client sends the write request to one of the coordinator in the Cassandra cluster, the coordinator maybe per key basis or per client or per query basis. So, per key coordinator ensures the writes for the key are serialized for maintaining the consistency. The coordinator uses partitioner to send the query to all the replica nodes responsible for the key. Now an X replicas respond the coordinate returns and acknowledgement to the client; now what do you mean by X replicas, why not all replicas? That we will see; now that strategy which is applied in Cassandra.

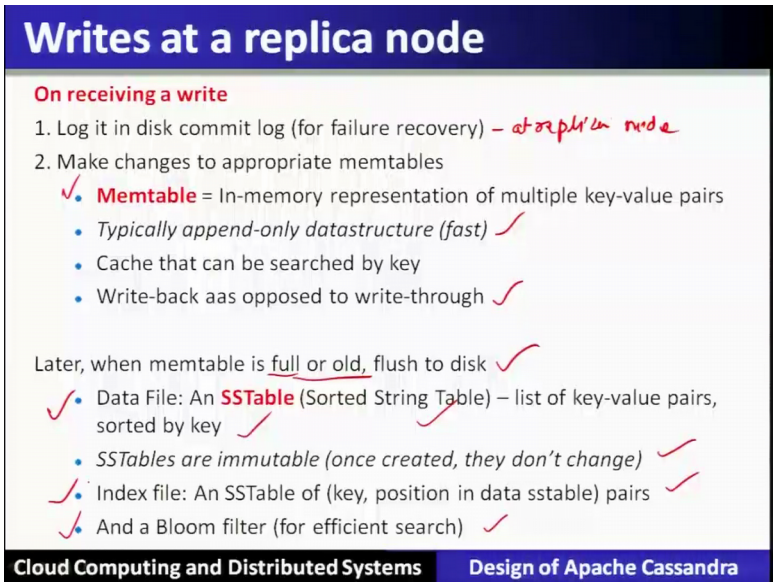
(Refer Slide Time: 25:15)



So, the first strategy which says that always writable we say that if any replica is down, the coordinator writes to all other replica and keep the write locally until the down replica comes up and then it will do the backup. So, it is the responsibility of the coordinator to take care of the replica which is down. When all the replicas are down then coordinator will be the at the front end and it will buffer all the writes, but it is not forever, but it will be for a few hours and one these replicas are up they will backed up. So, this is called a hinted handoff mechanism why because here the coordinator will manage to how these replicas updated it.

Now, another strategy is called one ring for data center that we know that if it is a multi data center; multiple data centers are involved then for each data center there is one ring; so, for every data center there is there is a ring. So, per data center one of these particular nodes will be elected as the coordinator. This particular coordinator job is to co ordinate with the other data centers and this election is done through the zookeeper which uses paxos like consensus for leader election.

(Refer Slide Time: 27:36)

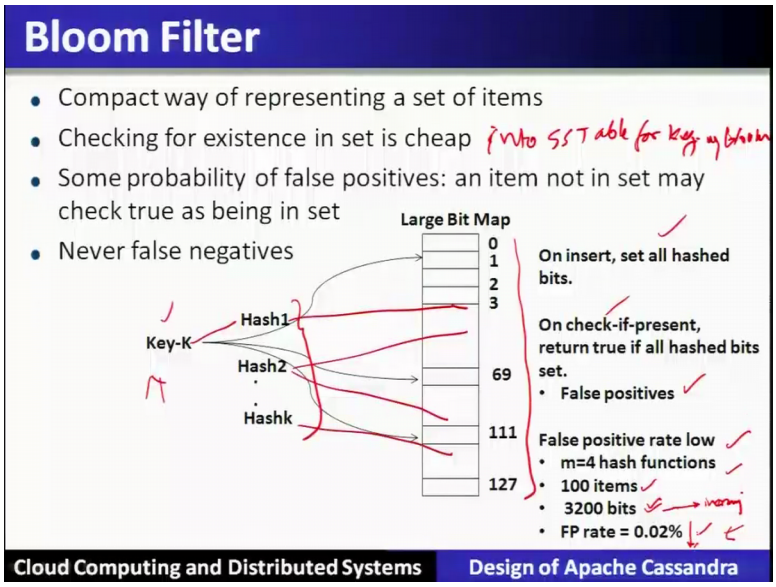


On receiving a write request at the replica node; it will log in its disc for ensuring the failure recovery. We will see the algorithms which use the log file to recover from the failure so, that it will be a minimal loss or the write operations which are incomplete and if it is filled in between they can be recovered.

Make changes to the appropriate memtables; memtable is an in memory representation of multiple key value pairs. So, typically only the append only data structure is there since its in memory; so, it becomes a fast operations. So, this is maintained in the cash that can be searched by the key; it is in contrast if it is these particular changes are directly done, then it is called a write back then it is called the write through as opposed to the write back. Now when the memtable because this is in memory when it is full or it becomes old, then it will be flush to the disc; that means, it will be made permanent.

So, for that there will be a file which is called SStable is stored string table; this is a list of key value pair which are sorted by the key. These SStable are immutable that is once created they do not change, now they are indexed using a file called index file. So, that they can be checked up for the key its position of the data in the SStable and so on; for that efficient search a bloom filter is being applied on SStable.

(Refer Slide Time: 30:00)



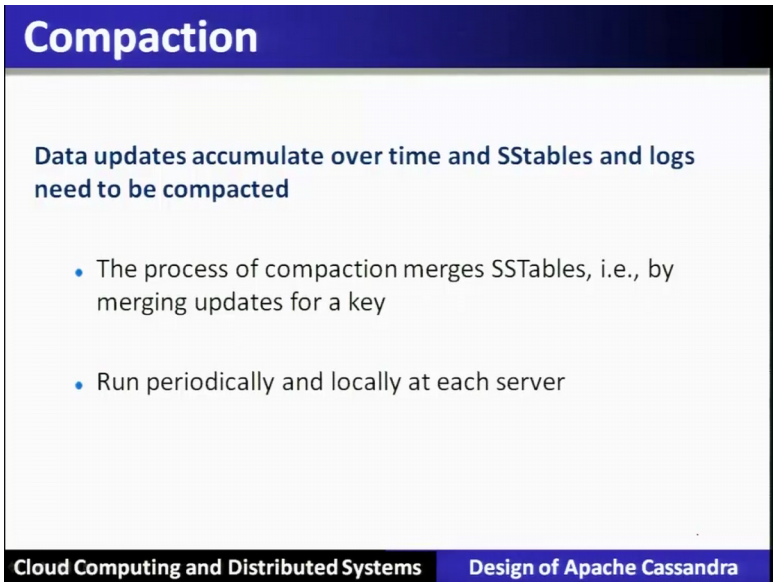
Let us understand what a bloom filter is; a bloom filter is a randomized data structure is a compact way or presenting beside of items checking for the existence in the set becomes cheap. Because here we are checking into SStable for a particular key using the bloom filter, to support the search operation in a fast manner; this is key bloom filter has some probability of false positives; that means, if a item is not present in the set may sometimes turn to be true. But if a item is present then it will never give a false information hence it is called negative.

Never have the false negative it may have the false positive, but that is not a false positive can be handled easily; that means, it has to look up again and search for that particular item in the set. So, let us see how the bloom filter works for a particular key; there are K different hash functions which are applied and every has function will hash into the values from 0 to 127. So, for a particular key and a particular hash function one of these bits will be mapped and it will be turn to 1; similarly this also and so on.

Since this particular SStable stores many different keys; so, all of them will be mapping the values of 1 is off if it is already there then is no change if it is 0, then return 1s. So, on insert therefore, the set all has bits will be turn on. So, on check if they are present all 1s, then it will turn true if one of them is 0 then it will be false; that means, it is not present.

There are some false positives; so false positive analysis if we see and here in the scenario when we have 4 different hash functions. And there are hundred different items stored for searching and if these number of bits are 3200; then it is analyze that the false positive rate is 0.02 percent. This can further be reduced by increasing these bits that is the bitmap; if you can increase then this can further go down. So, the bloom filter is used here to search the SStable of particular keys; hence as a index file, bloom filter is applied for an efficient search.

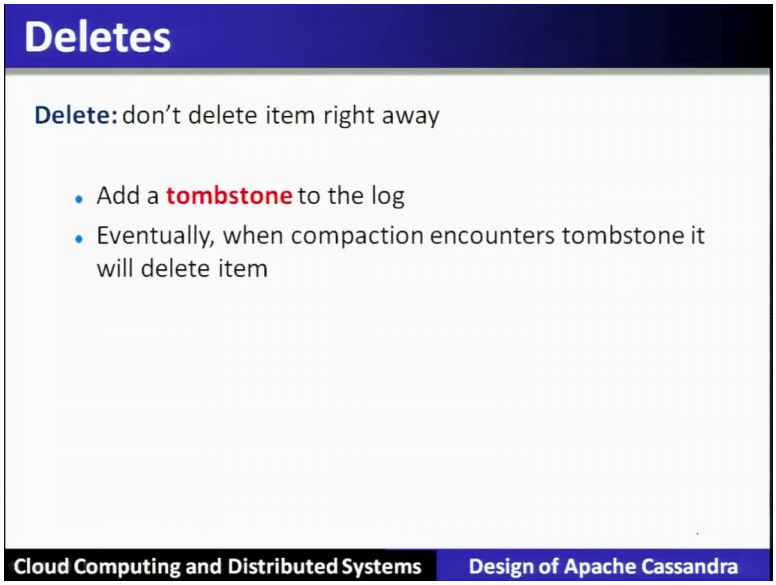
(Refer Slide Time: 33:32)



Now, another operation which is supported to deal with the multiple SStable; which will be accumulated over the period of time and the log tables is called compaction. So, the

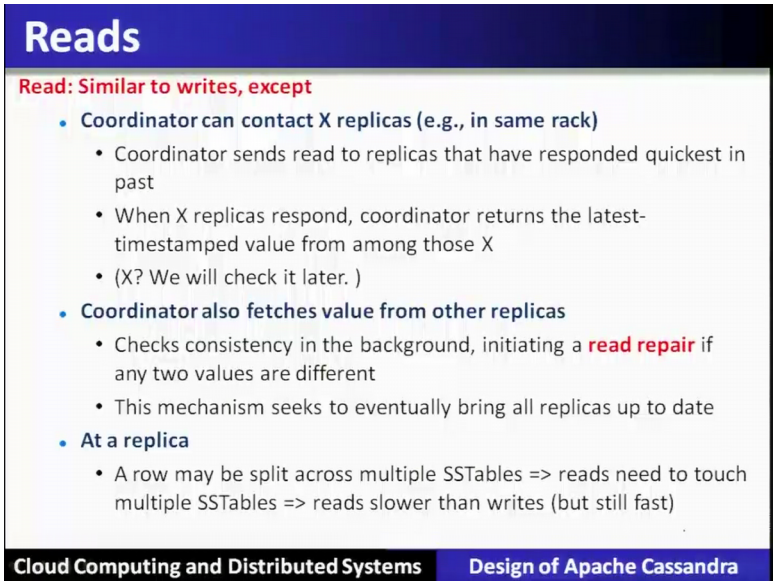
process of compaction is to merge different SStables that is merging and updates for a particular key and run periodically, locally at each server this compaction process.

(Refer Slide Time: 34:06)



Let us see how the delete operation is being supported in Cassandra. Delete is supported that do not delete the item right away; instead it will add in a tombstone to the log and eventually when the compaction and counters the tombstone then it will delete the item.

(Refer Slide Time: 34:36)

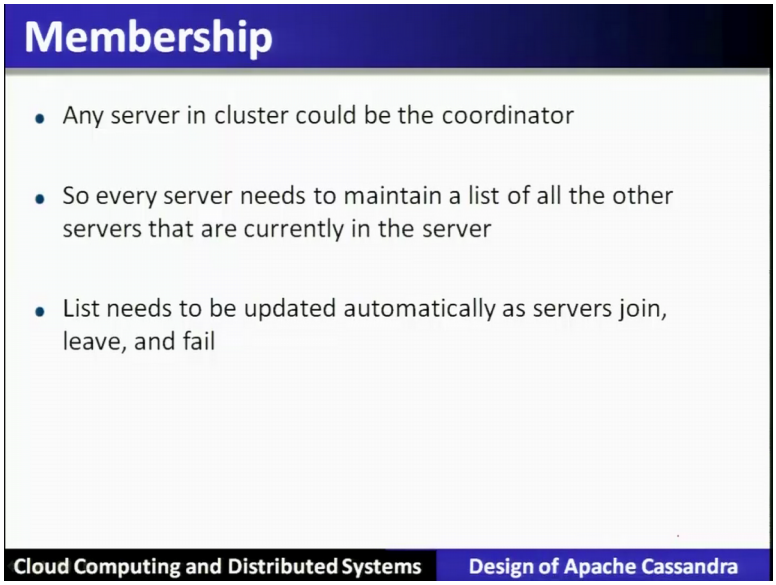


Now, let us see how the reads operations are supported read is similar to the write except the coordinator can contact X replicas not all; in the same rack. Coordinator sends the

read request to the replicas that have responded the quickest in the past, when X replica responded the coordinator returns the latest time stamped value from among those X replicas. So, the values of X; that means, not all we will see now the coordinator also fetches the values from other replicas.

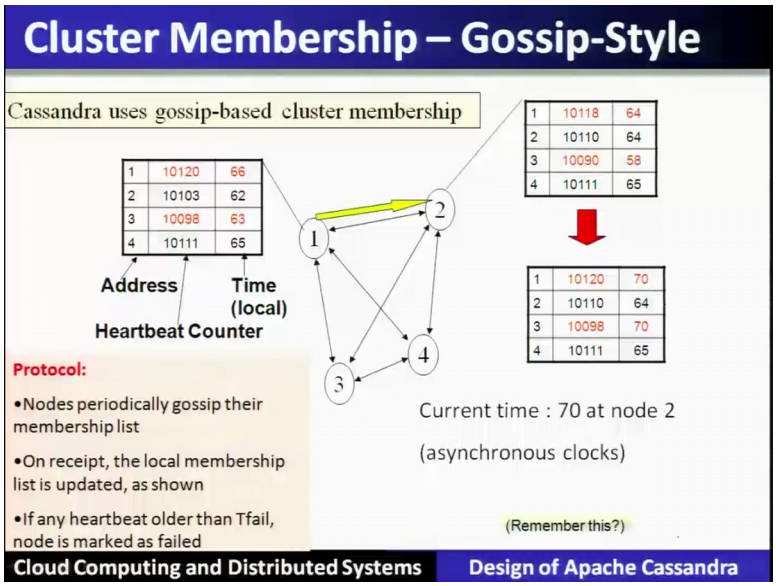
So, checks the consistency in the background and if multiple replicas have different values; that means, they are all not same. Then there is a then it will initiate the read repair if the two values are different; this mechanism seeks to eventually bring up all the replicas up to date eventually. So, at a replica a row maybe is split across multiple SStables. So, reads need to touch multiple SStables; that means, read becomes slower in that case than the writes, but it still it is the faster.

(Refer Slide Time: 36:05)



So, membership any server in the cluster could be the coordinator. So, every server need to maintain a list of all other servers; that are currently working or interacting as a servers. So, these list need to be updated automatically as new server join fail or leaves.

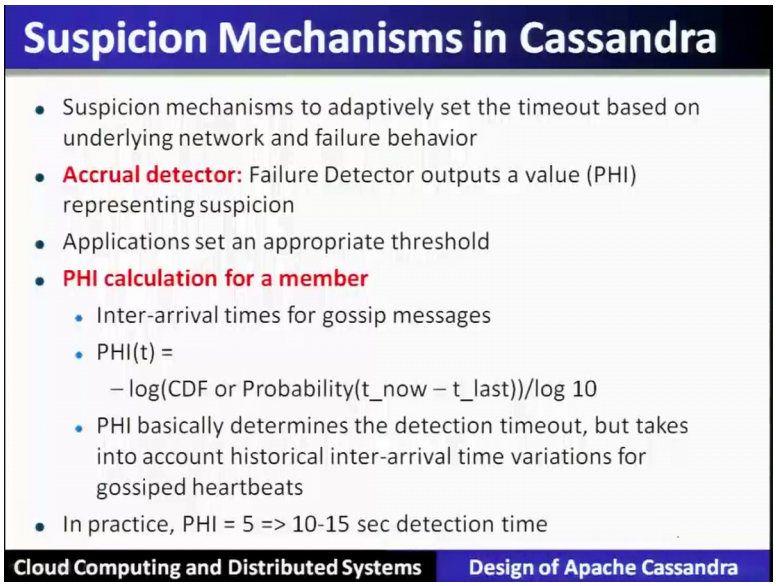
(Refer Slide Time: 36:29)



This membership is done using the gossip style failure detection. So, in this method or a protocol the nodes periodically gossip there membership information. When we say gossip; that means, randomly it will exchange its information to some randomly selected neighbors; not all neighbors hence it is a gossip. On the receipt of this local membership the list is updated and if any heartbeat is older than; than Tfail node then it is marked as the failed.

So, it is kind of membership; that means, it will try to figure out the nodes which are failed or which are active; hence this particular gossip style membership is maintained in Cassandra.

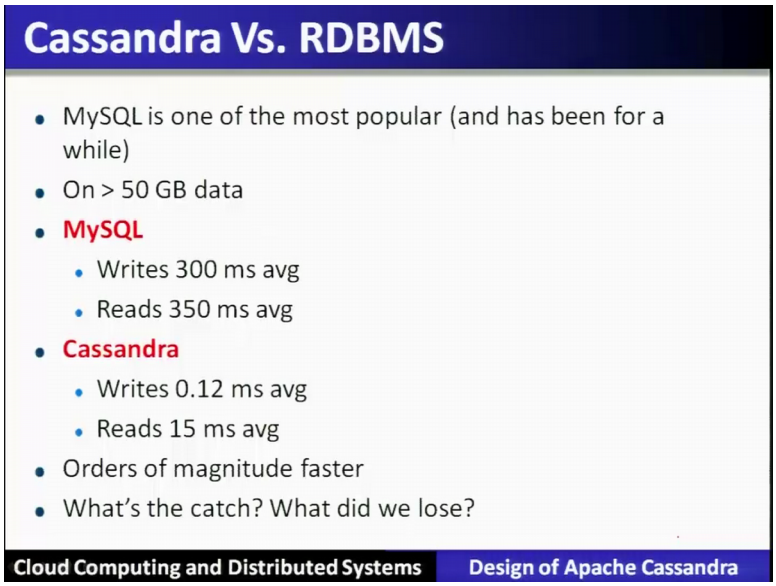
(Refer Slide Time: 37:38)



Now, there is a suspension mechanism in Cassandra to detect the failures. Accrual detector is being used here the failure detector output value which is called phi representing the suspicion level. So, applications will set an appropriate threshold to deal with that; phi calculations for a member is based on inter arrival times for the gossip messages. So, phi of a particular time is equal to the log of cumulative distribution function or a probability of t now minus t last divided by log 10.

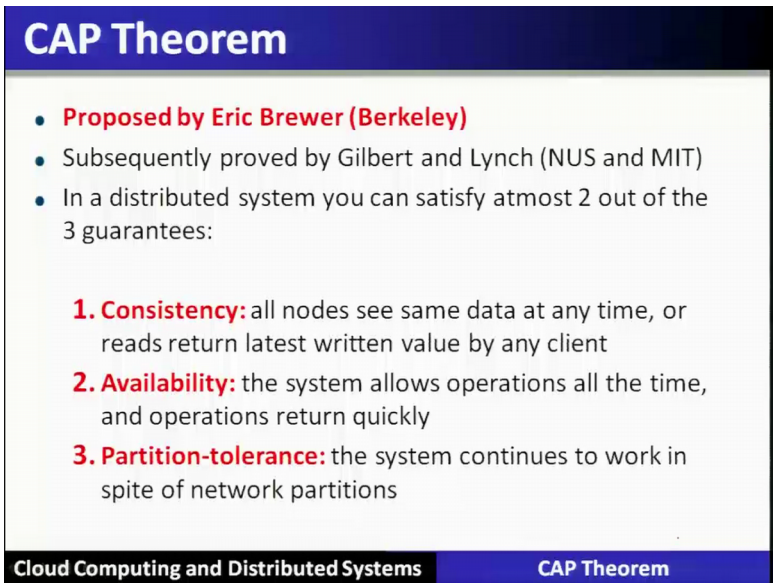
So, phi basically determines the detection time out, but takes into account the historical inter arrival time variations for the gossip heartbeats. In practice phi is equal to 5, this will imply that 10 to 15 seconds is basically the detection time in case of the failures.

(Refer Slide Time: 38:53)



Therefore, if we compare the Cassandra with RDBMS now we have seen that Cassandra is supporting the NoSQL data store in the form of key value store. So, here if we compare for the data which is more than 50 GB; MySQL has the write latency the write operation is 300 millisecond on an average speed is 350 milliseconds whereas, Cassandra will take 0.12 milliseconds and read will take 15 milliseconds. So, this is in the order of magnitudes much faster CAP theorem. So, CAP theorem was proposed by Eric Brewer of Berkeley.

(Refer Slide Time: 39:55)



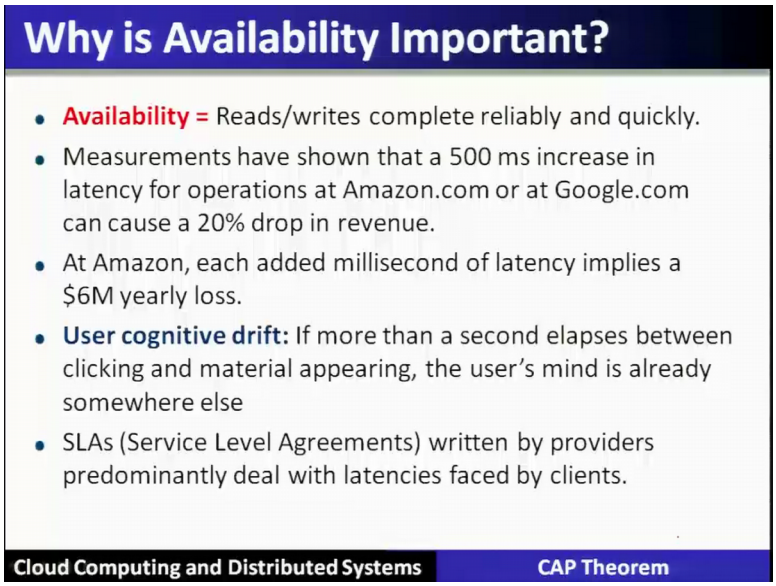
And which was subsequently proved by Gilbert and Lynch of NUS and MIT. In the distributed system, as per as their observation is that you can satisfy at most 2 out of 3 different guarantees. What are the 3 guarantees? The first one is called Consistency will form C in the CAP, second one is called Availability that is A in the CAP, third is the Partition tolerance that is P in the CAP. So, what you mean by consistency? That consistency says that all the nodes see the same data at any time or the read returns latest value by any client called consistency.

This is ensured in the RDBMS, second aspect is called availability that is the system allows the operation at all points of time and the operations return quickly that is called availability that also is supported in most of the RDBMS. Third is called partition tolerance which says that system continues to work in spite of network partition that is failures which lead to the network partition that is called partition tolerance. So, let us see what CAP theorems says; CAP theorems says in a distributed systems or in a cloud system you can satisfy at most 2 out of 3 guarantees; that means, either consistency or availability you can support in the cloud system; that means, not partition tolerance and so on.

So, we will see that partition tolerance is important why? Because now data in the cloud is stored over multiple data centers; which are geographically distributed hence it has to be partition tolerance. If partition tolerance is very much required, then out of consistency and availability one of these 2 things has to be also taken up because at most 2 can be satisfied. Availability is very much important, why?

Because these services which are serving customer will otherwise lose the revenue; so, availability is also one of the important where we can compromise is called consistency. So, a compromise consistency is very much to be create and developed in the modern scenarios in the cloud computing system; so, in that perspective we will see the CAP theorem.

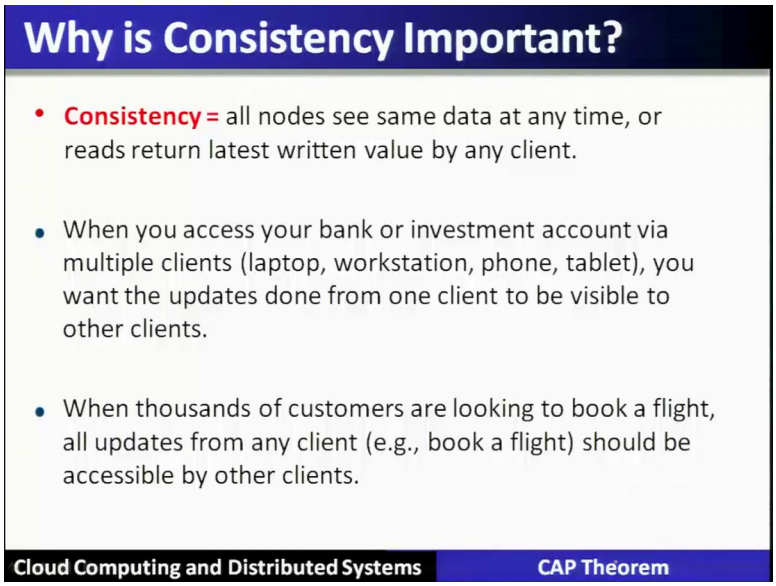
(Refer Slide Time: 42:46)



So, availability that is reads and writes will complete reliably and quickly measurements have shown that 5 millisecond increase in the latency; for the operations at Amazon dot com or a Google can cost 20 percent drop in the revenue in the sense many customer will move away and will buy from other commerce site.

So, at Amazon each added milliseconds of latency will imply that 6 million revenue loss per year will happen. So, latency is very important that is availability is one of the important factor or a guarantee in the cloud computing system. These guarantees are being supported by the; by the service providers to their clients for example, this Cassandra is being used by Netflix. And the Netflix requires the availability, this guarantees by for this particular service.

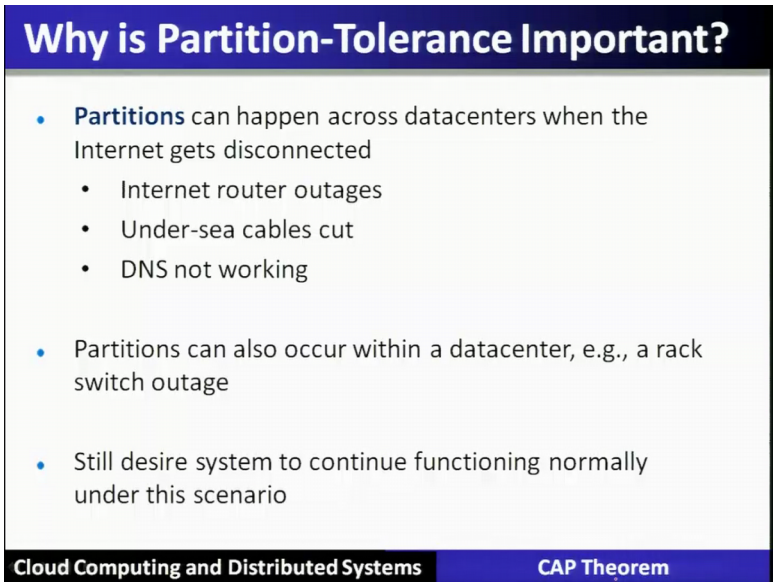
(Refer Slide Time: 44:12)



So, Service Level Agreements written by the providers predominately deal with the latency is which are faced by the by the client. The second component in CAP is called consistency let us see why the consistency is important to deal and what are the guarantees in what are the scenarios. So, consistency means all the node see the same data at anytime and the read returns the latest written value by any client. So, when you access your bank account; where multiple clients you want to update from one client to the visible to the other clients.

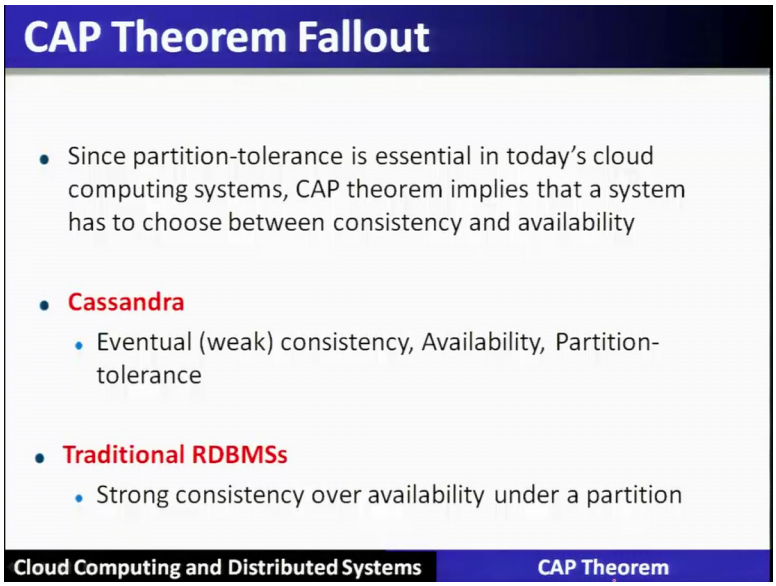
That is done in the RDBMS, but in our scenario when thousands of customers are booking a flight ticket all the updates from the client should be accessible by the other clients hence it is called the consistency.

(Refer Slide Time: 45:14)



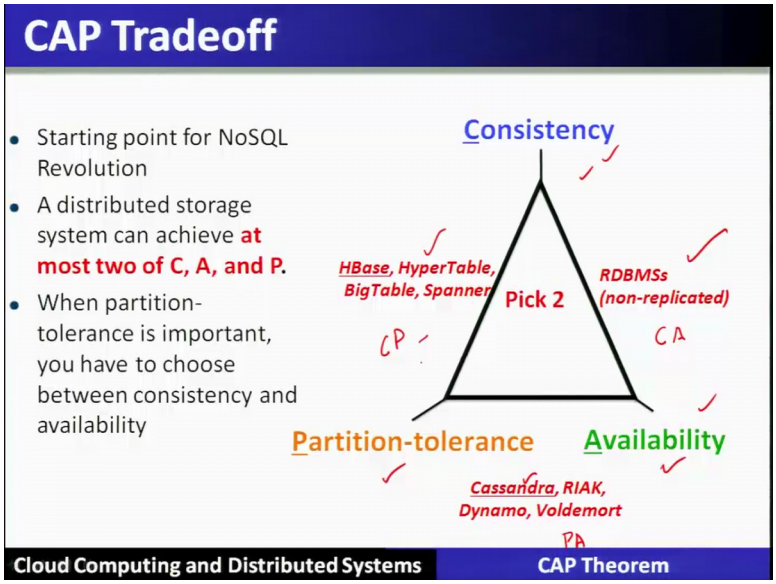
Third one is called partition tolerance why the partition tolerance is an important aspect in what condition that we will see. So, partitions can happen across the data centers when internet gets disconnected due to the internet router outages or the trans oceanic sea cables are cut or DNS is not working. So, partitions can occur within the data centers also for example, if the top of the racks switch is not functioning. So, the all the servers within the data center will be in the partition. When in spite of these issues we want that the services should continue as normal in the scenario, hence partition tolerance is one of the important factors.

(Refer Slide Time: 46:09)



But the CAP theorem guarantees that only 2 out of 3 important parameters can be ensured in any design of a cloud computing system. So, as far as the Cassandra is concerned Cassandra chooses to give the guarantees of availability and partition tolerance whereas, consistency it will be compromising it and that consistency is called eventual consistency that we will see now. In traditional database that is in traditional in RDBMS supports a strong consistency; that means, it supports a strong consistency over availability under the partition; so, it is not partition tolerant.

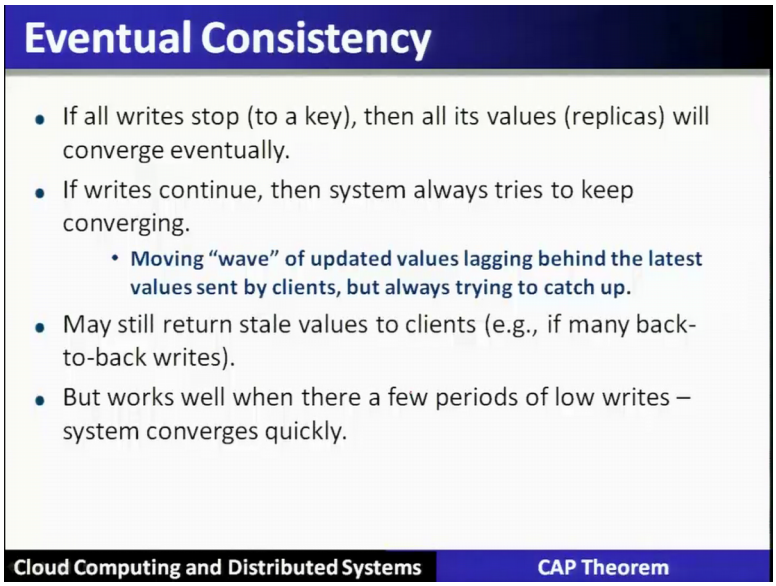
(Refer Slide Time: 47:01)



Let us see through this particular figure that CAP tradeoff. So, here this is an RDBMS; the RDBMS normally runs on a particular server it is not replicated. Hence, partition tolerance is not that important, but it supports consistency and availability. Similarly the NoSQL or a key value store like Cassandra, RIAK or Dynamo and Voldemort guarantees the partition tolerance and also guarantees the availability.

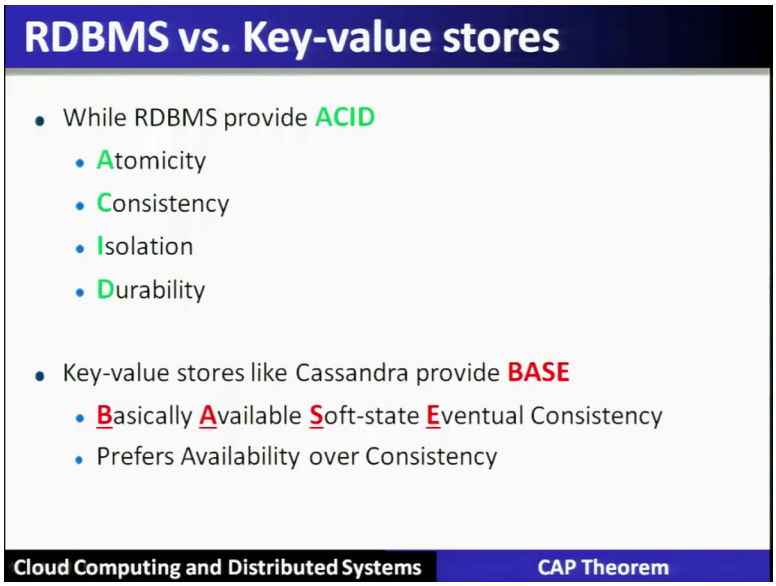
Whereas, consistency is compromised; similarly another type of key value store that is HBase, HyperTable, BigTable and Spanner guarantees consistency and partition whereas, availability is concerned it can be compromised. So, we will see different products are available around the CAP trade off.

(Refer Slide Time: 48:17)



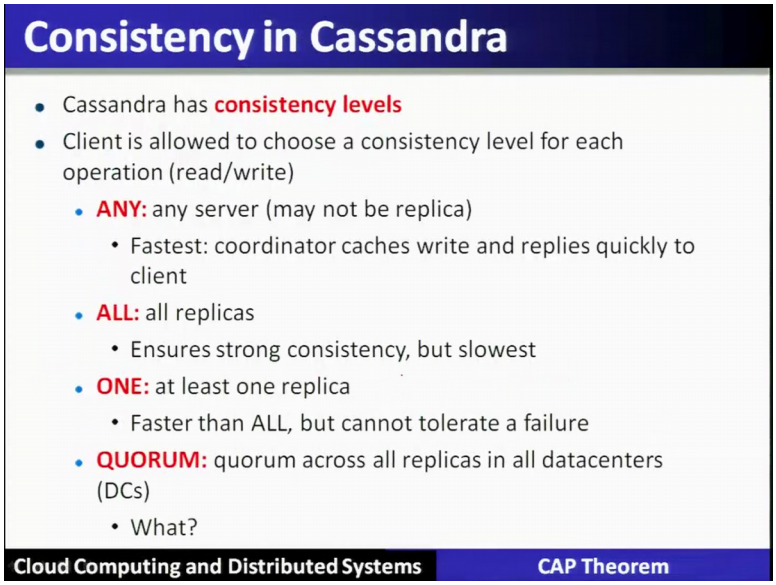
So, let us start with the different form of consistencies which are being supported in different products. So, eventual consistency which is supported in Cassandra let us go and see the details of it. So, if all the writes stopped to a particular key then all its values that is replica will converge eventually. And if the right continues then system always tries to keep converging so; that means, since the clients may return with the still values; if there are many back to back writes happening and they are all not converged. But it works well when there are a few periods of low writes where the system converges quickly.

(Refer Slide Time: 49:09)



Now, if you compare the RDBMS verses the key value stores; RDBMS supports an acid properties in the transaction whereas, the key values store like Cassandra supports just opposite to it that is called BASE; Basically Available Soft-state Eventual consistency which will prefer availability over the consistency as eventual consistency is mentioned here in the base condition.

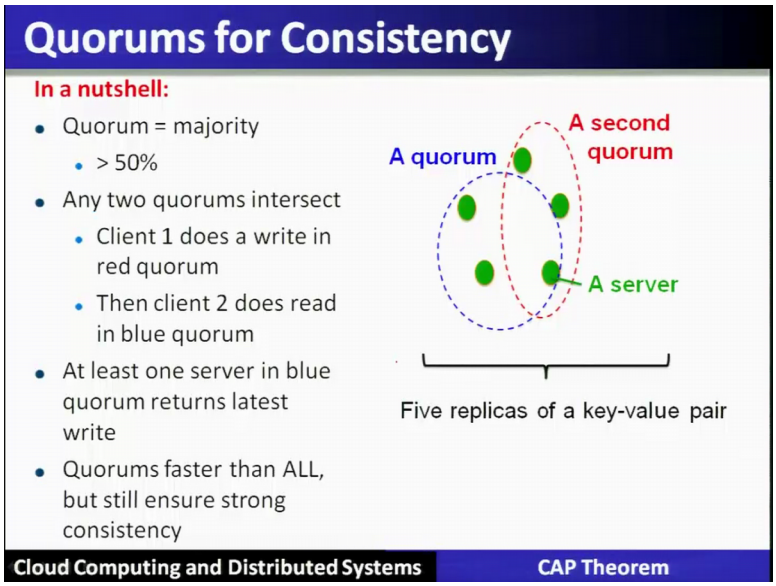
(Refer Slide Time: 49:40)



Let us see what are the different consistency levels supported in the Cassandra and how it is supported that we will see. So, client is allowed to chose the consistency level for each operation that is the read and write. If the consistency level is any; that means, any server is basically is good enough to provide the value, it is the fastest in terms of that coordinator caches the writes and replicas quickly response to the client request.

Now in the consistency level is all; that means, all the replicas are required to be consulted and the latest writes values will be given back to the client. It ensures the strong consistency no doubt, but it will be a slower operation; if the consistency level is 1 then at least one replica responds and that response will be given back. It is faster, but it cannot tolerate the failures; now if the consistency level is the quorum and quorum across all replicas in the data center is required to get their values. So, that it may be returned; quorums we have already seen in the previous discussions.

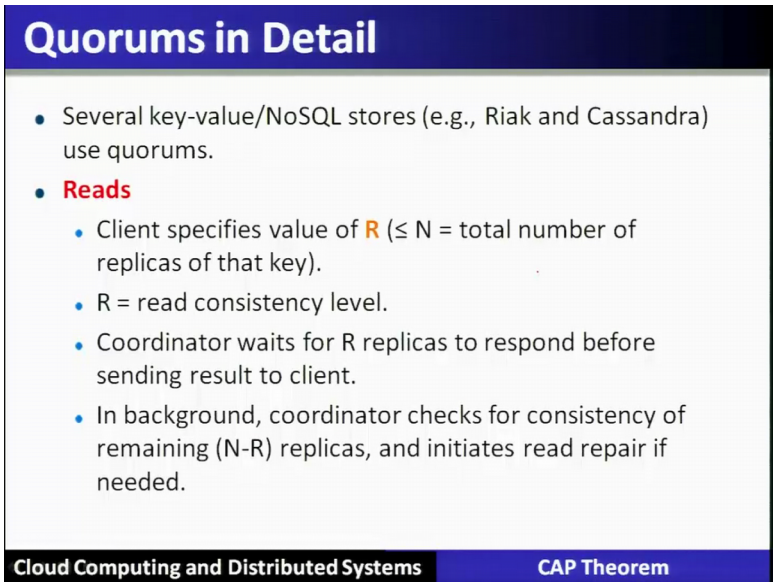
(Refer Slide Time: 51:18)



Let us see about the quorum how the quorum will support the consistency. So, quorum is the majority for example, here in the figure 5 different replicas are there. So, out of 5 the majority is 3 so; that means, out of 3; 2 different quorums can be found here in the example.

If these 2 quorums intersects; so, there exist at least one server which is common in both the quorums. Now whenever the client writes chooses through the quorum; so, the client 1 writes in the red quorum. So, when it writes in the red quorum then the client 2 want to read from the bloom quorum, then in the bloom quorum all 3 different replicas will give the values; this will be having the updated value. So, that way the client will be served with the latest value here in this case; quorums are faster than all, but still it ensure the strong consistency.

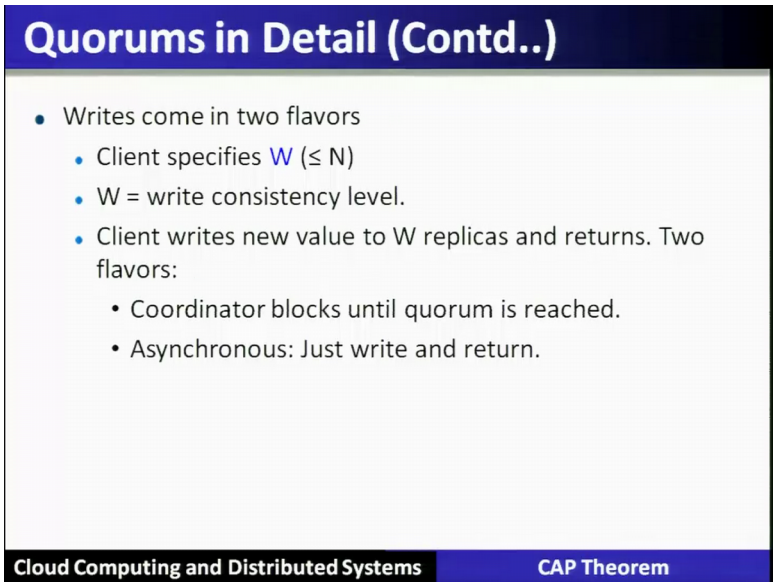
(Refer Slide Time: 52:34)



To several key value store and no SQL store such as Riak and Cassandra uses the concept of a quorum to support the strong consistency. Here the reads the client is specifies the value of R which is less than the total number of replicas of that particular key less than or equal to N that is bounded by N. So, R is the read consistency level.

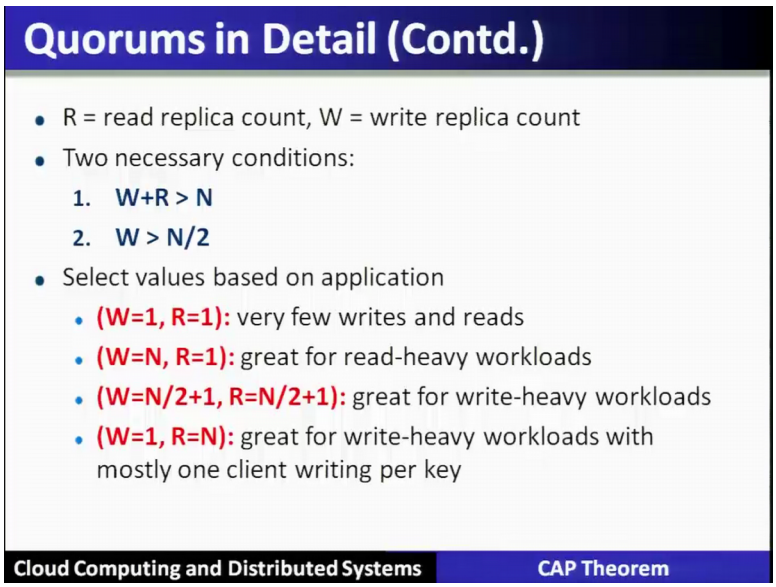
So, the coordinator waits for all replicas to respond before sending results to the client. On the background the coordinator checks for the consistency of the remaining N minus R replicas and initiates the read repair if needed. So, the reads client specifies the value of R which is bounded by the total number of replicas of that key; R will specify the read consistency level. So, the coordinator wait for R replicas to respond before sending results back to the client. In the background the coordinator checks for the consistency of remaining N minus R replicas and initiate the read repair if any is required.

(Refer Slide Time: 53:57)



So, that is quite simple; as far as write is concerned write will come into different flavors. So, client specifies the write consistency which is bounded by N. So, client writes the new value to W replicas and returns. That is why it is having 2 different flavors that is the coordinator blogs until the quorum is reached or it will be an asynchronous in the sense it was done write and return back. Now let us see the quorums in more details.

(Refer Slide Time: 54:40)

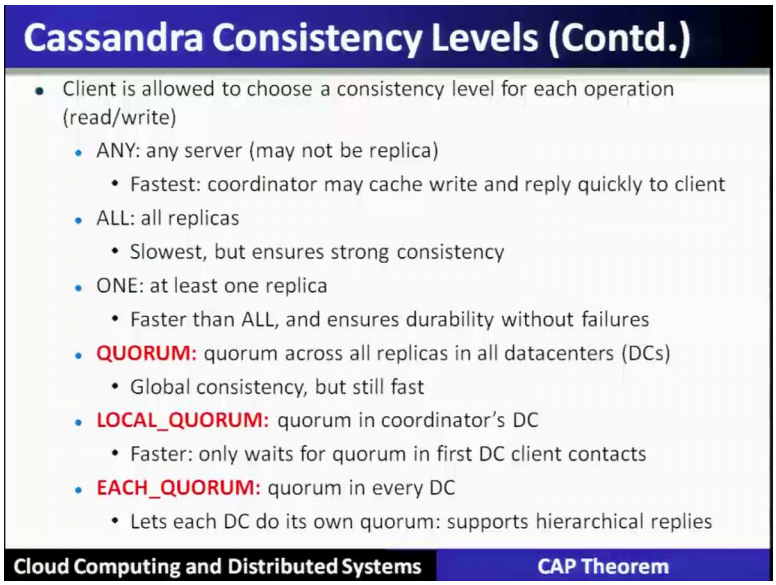


When R is a read replica count and W is the write replica count there are two necessary conditions have to be followed; write plus read replica count should be greater than N

and write is greater than N by 2. So, these are the necessary conditions in the quorum. So, select the values based on the applications; if there are very few writes and reads, then W is equal to 1 and R is equal to 1 when there are read heavy workloads then W is equal to N and read is equal to 1.

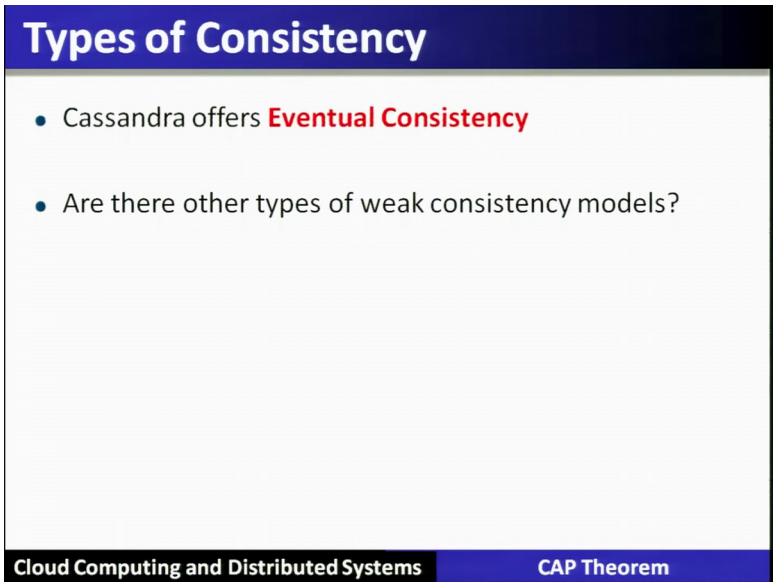
When there is a write heavy workload, then it is W is equal to N by 2 plus 1 and R is also N divided by 2 plus 1. Now N write heavy workloads with mostly one client writing per key; then W is equal to 1 and R is equal to N.

(Refer Slide Time: 55:49)



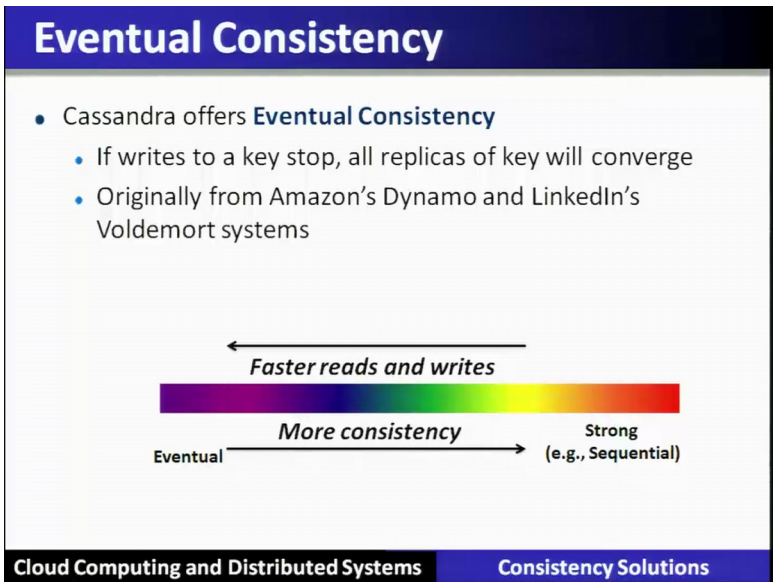
Now, let us see the Cassandra consistency level which is being supported there. So, client is allowed to choose the consistency level for each operations; that is read or write. So, any means any server may not be the replica; it is the fastest that is the coordinator may cache write and reply quickly to the client; all means all replicas it is slowest. One is means that at least one replica quorum is that quorum across all replicas in all the data centers. Local quorum means quorum in the coordinators data center each quorum means quorum in every data center.

(Refer Slide Time: 56:42)



There are different types of consistencies which Cassandra offers. So, Cassandra offers eventual consistency what are the other type of consistency that is the weaker form of consistency model that we will see.

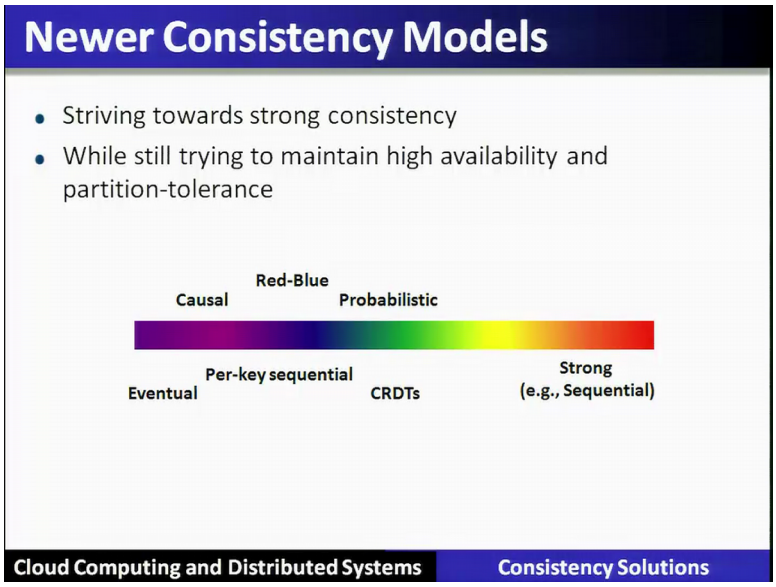
(Refer Slide Time: 56:55)



So, we will see different range of consistency solutions which are available starting from eventual that is the weakest, moving towards the strong consistency. Now if a strong consistency is supported then; obviously, the response will be slower and if we want a response to be faster, then we have to move towards eventual consistency.

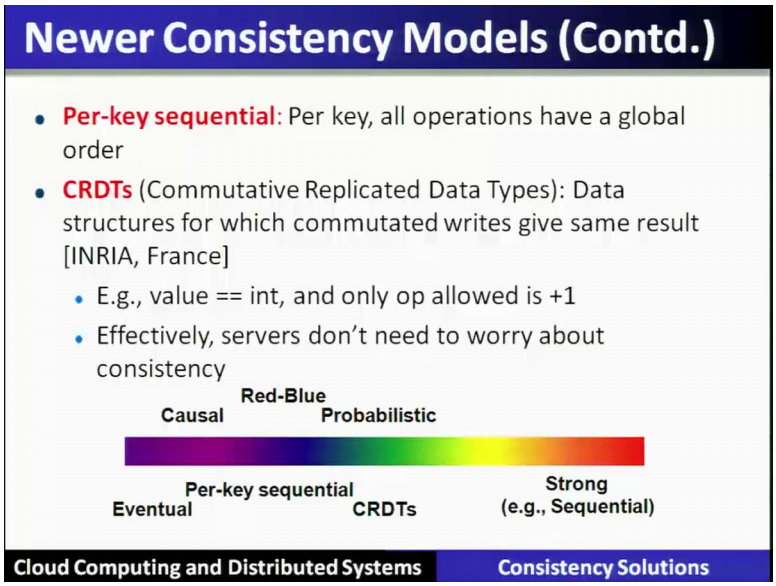
So, Cassandra supports eventual consistency. So, if write to a key will you stop; that means, all the replica of the key will converge. So, that was originally from Amazons dynamo and Linkedln's, Voldemort system this idea was taken up and used in the Cassandra.

(Refer Slide Time: 57:48)



Now, there are other newer consistency models; some are striving towards strong consistency, while still trying to achieve the high availability and partition tolerance. So, these are some of the different schemes which are shown over here causal, red blue, probabilistic, per keys sequential, CRDTs and so on.

(Refer Slide Time: 58:12)

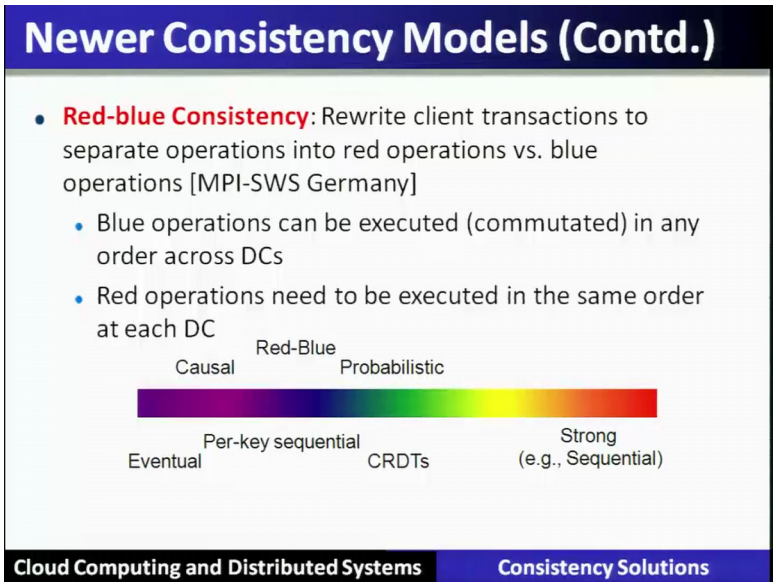


So, let us see these newer consistency models which are available in the literature and will also being applied in most of the products. So, per key sequential means for per key all the operations have to be ordered globally. So; that means, every client has to go through a particular coordinator and coordinator will ensure the per key sequential operations.

That means a global ordering is being followed, but the problem is of scale; scalability. The another one is called another consistency model is called CRDTs; it is a newer consistency model that is commutative replicated data types. Here the data structures for the commutated writes will give the same results. For example, if the operation plus 1 is there plus 1 means increment of a particular value. So, whose ever if let us say 2 processes uses this particular operation.

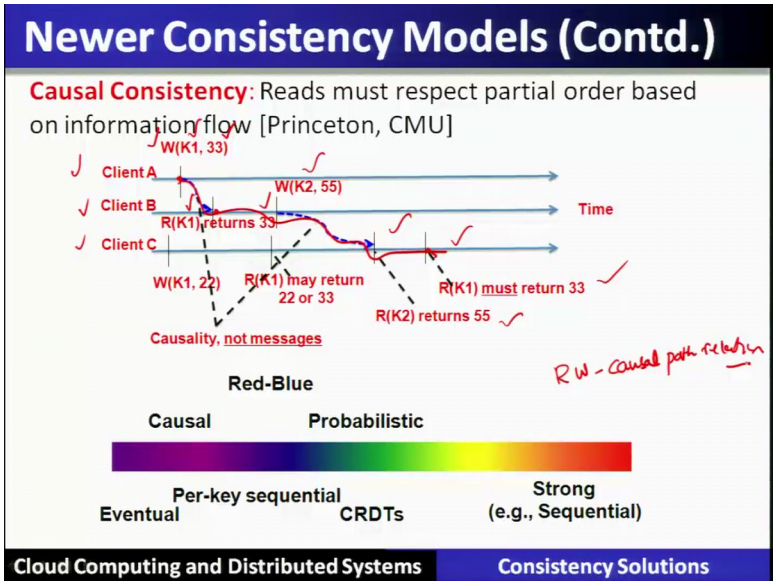
So, it is a immaterial if we can exchange or we can interchange the order in which these operations are being allowed; hence it is called commutative operations. So, besides plus operator there are other such operations which are being supported through CRDTs. So, they are effectively servers do not need to worry about the consistency, but they have to deal with the operations and this is supported in CRDTs.

(Refer Slide Time: 59:52)



Now, the next level is called next level of consistency another newer model is called blue red blue consistency model. In red blue consistency model, the client transactions are separated into two different operations. Red operation that is commutative type and blue operations; so, the blue operations can be executed that is using the commutative type in any order across the data centers. Whereas, red need to be executed in the same order; that means, they cannot be commutated this is called red blue consistency.

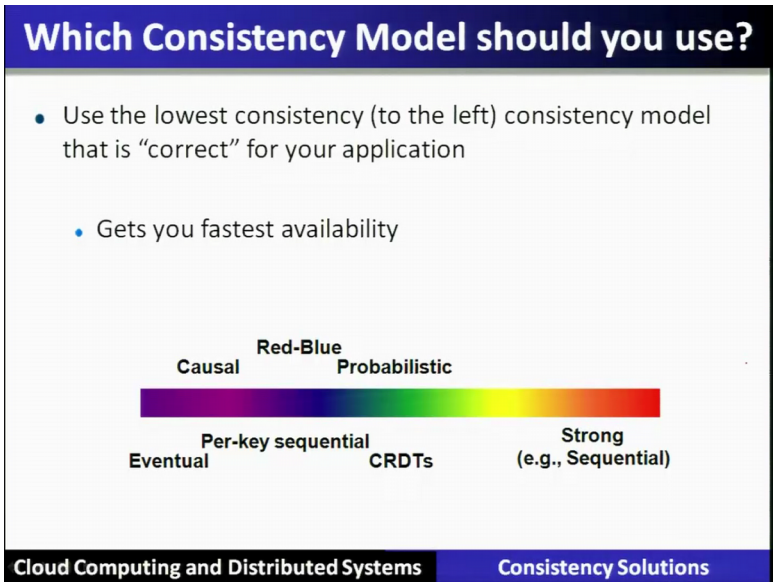
(Refer Slide Time: 60:33)



Another newer kind of consistency is called causal consistency; here the reads must respect the partial order based on the information flow; that is given by the Princeton University. For example, there are 3 different clients A B and C; the client A will write down in key K 1 value 33 and after that client B will read the value of key K 1, which will be return 33. So, this particular these 2 operations write and read they are causally related; thereafter in client B will write down key K 2 with the value 55 and thereafter the client C will read the value of K 2 which also will return 55.

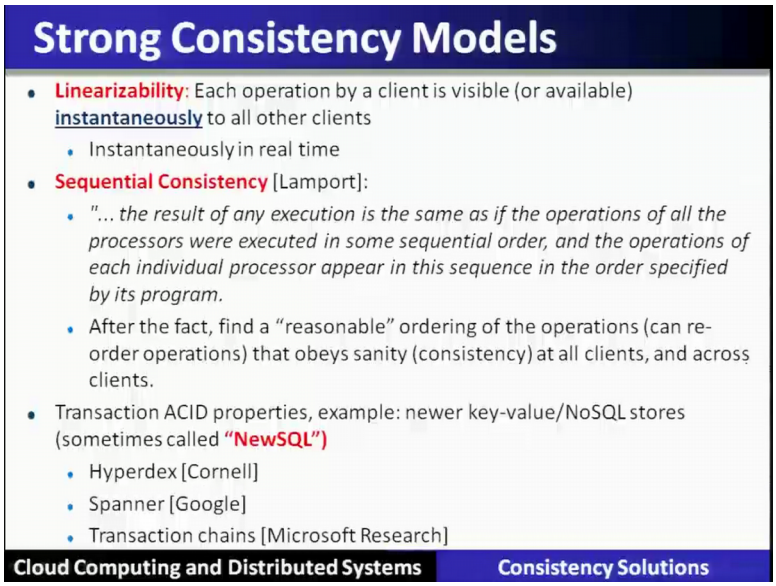
So, the client Bs K 2 and client Cs K 2 there and read of K 2 these 2 operations are also causally related. Finally, the client C when will read the key K 1; it must return 33 y because there is a causal path; although it is not through the message exchange, but using read and writes will follow the causal path and that is the relation which will bind this is called causal consistency model.

(Refer Slide Time: 62:15)



Now, here out of this range of consistency model which one will be useful for a particular applications is depends upon the availability; that means, which is supported by the applications.

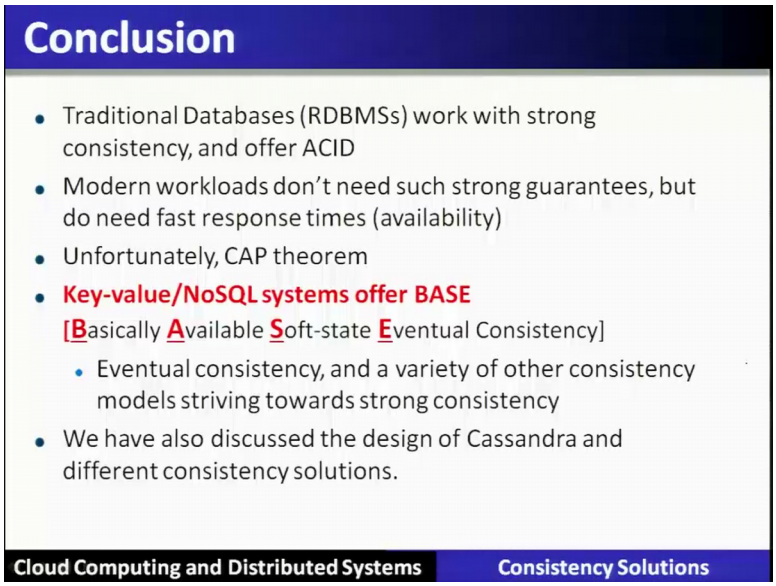
(Refer Slide Time: 62:31)



Let us see some of the strong consistency model the first one is called linearizability. So, here each operation by the client is visible or is available instantaneously to all other clients. So, this is just like RDBMS which supports this kind of consistency that is called linearizability. Now, in a the systems like key value store, there is another consistency which is also categorized as a strong consistency is called sequential consistency which is given by the Lamport here the result of any execution is the same as if the operations of all the processor for executive some sequential order and operations of each individual processor appear in this sequence in the order is specified by the program.

So, here the order is not that order which is being provided in linearizability, but eventually all the processor will see the same order and that is fine in some of the applications. Now comes the acid properties there are newer NoSQL database called NewSQL which supports the transactions acid properties such as hyperdex, spanner and transaction chains by different vendors.

(Refer Slide Time: 64:10)



Conclusion traditional databases RDBMS work with the strong consistency and offers the acid properties. Whereas, the modern workloads do not need such a strong guarantees, but do need fast response time that is the availability.

Unfortunately the CAP theorem says that out of 3 any 2 can be guaranteed. We have seen the key value store NoSQL; which is not offering CAP which is not offering acid, but it is offers base that is basically available soft state eventual consistency. Eventual consistency and a variety of other consistency model striving strives towards strong consistency, we have discussed the design of Cassandra and also different consistency solutions.

Thank you.