Cloud Computing

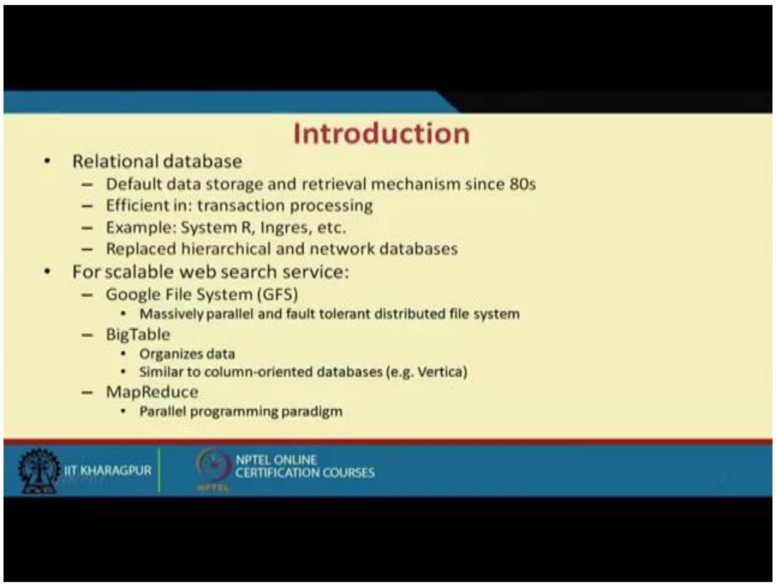
Prof. Soumya Kanti Ghosh
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 13
Managing Data

Hello. So, we will continue our discussion on cloud computing. Today, we will discuss about some aspects of managing data in cloud, right. So, as we understand that in cloud; as we have discussed in our earlier lectures that in cloud, one of the major aspect is the data because at the end of the day, your data and even processing applications are in somebody else's domain, right. So, they are being executed at somewhere else which is beyond your direct control. So, it is virtually host in some virtual data; a virtual machine somewhere in the cloud. So, it becomes tricky to on the security point of view that we have discussed; not only that if you look at from the other point of view; so, from the clouds provider point of view, managing huge volume of data keeping their replicas and making them queriable and these becomes a again a major issue.

So, all our conventional relational or object oriented model may not directly fit into the thing, right. So, long you are doing on a small instances experimental some database application or some small experimentations, then it is fine, but when you have a large scale thing where huge amount of read write going on or the volume of data is much much higher than the normal operations, then it is; we need to look in a different way. These are the things which come into not only for the cloud, it was there a little earlier also; like how this parallel database accesses; parallel database execution; read-write execute operations can be done. So, those things become more prominent or a de facto mechanisms; when we talk about in context of cloud. So, what we will try to do is more of a overview of how data can be managed in cloud or what are the difference strategies or schemes people or this ISPs follows and it is not exactly the security point of view; it is more of a management data management point of view, right.

(Refer Slide Time: 03:02)



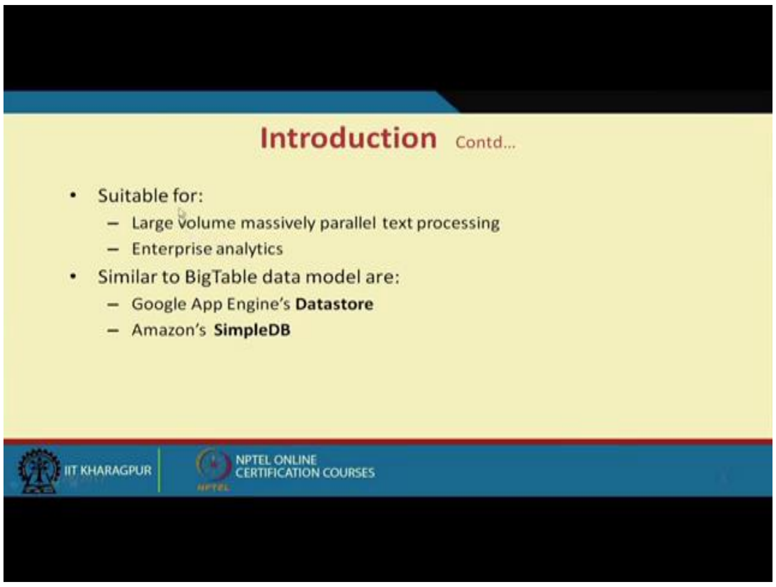
Introduction

- Relational database
  - Default data storage and retrieval mechanism since 80s
  - Efficient in: transaction processing
  - Example: System R, Ingres, etc.
  - Replaced hierarchical and network databases
- For scalable web search service:
  - Google File System (GFS)
    - Massively parallel and fault tolerant distributed file system
  - BigTable
    - Organizes data
    - Similar to column-oriented databases (e.g. Vertica)
  - MapReduce
    - Parallel programming paradigm

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, we will talk about a little bit of relational database already known to you then what you known to do that scalable data bases or data services like one of the couple of things are important one is Google file system big table and there is a Mapreduce parallel programming paradigm; those are the things which comes in back to back, when we are doing to the things. So, what we want to do when we were we are managing anything on a cloud platform; whether it is application or data we want to make it scalable in the sense the it suites scale as the requirement goes up. So, scale-up scale-down in a ubiquitous way or minimum interference from the; or minimum human or management interference. So, that type of infrastructure; we want to come up with, right, it is true for data also.

(Refer Slide Time: 04:09)


Introduction Contd...

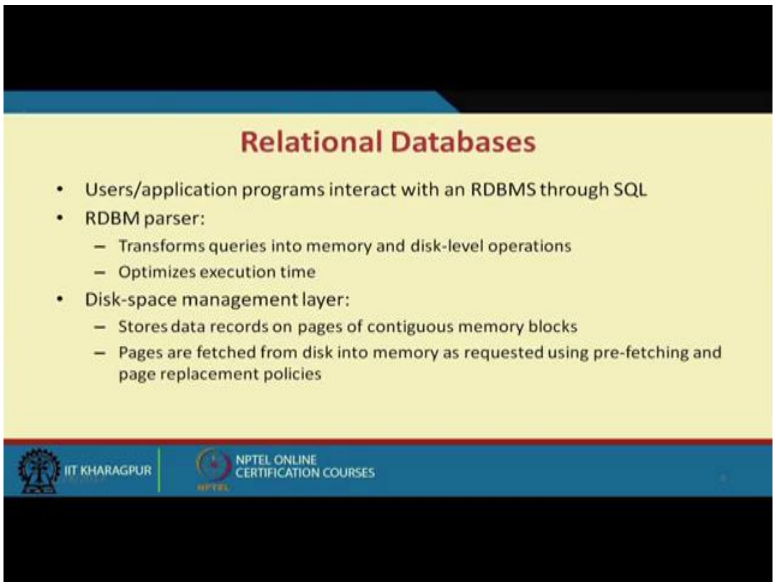
- Suitable for:
  - Large volume massively parallel text processing
  - Enterprise analytics
- Similar to BigTable data model are:
  - Google App Engine's **Datastore**
  - Amazon's **SimpleDB**

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, these are primarily suitable for large volume of massively parallel text processing, right that is one of the major thing or it is suitable for environment say enterprise analytics, right, I want to have a; if we want to do analytics on a distributed data stores, right, it may be a chain of a shopping or commercial staff or it may be a banking organization or financial any financial organization, even it is something to do with large volume of other type of data like it metrological data, it maybe climatological data something which need to be chant or has a distributed things, I need to do some parallel processing down the line where the actual effect comes into play. If you have a simple database with a simple instant, then you may not have gone to cloud for that; right. So, it may be a simple system or you buy a very a VM and work on it then the actual effect of cloud things are actual advantages of cloud you are not taking out.

So, we will see that similar to big table models there are Google app engines datastore, Amazon simple DB which are which different provides provide in different flavor, but the basic philosophy are same.

(Refer Slide Time: 05:46)



Relational Databases

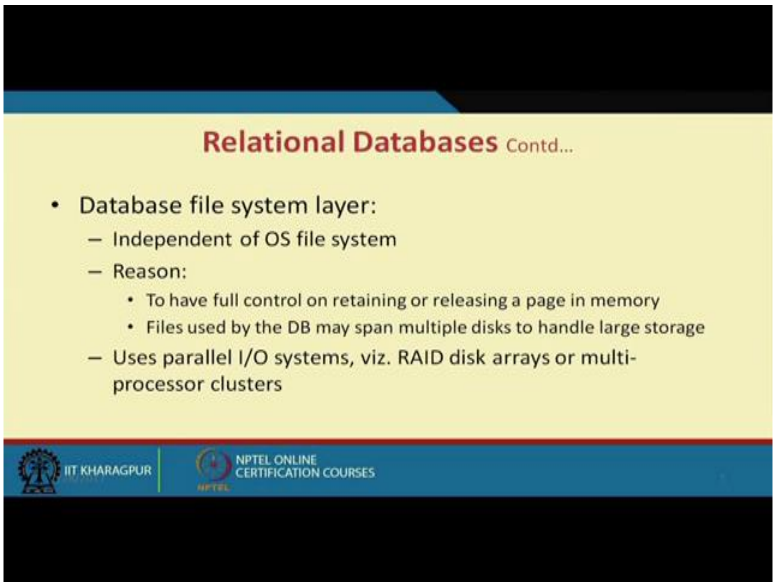
- Users/application programs interact with an RDBMS through SQL
- RDBM parser:
  - Transforms queries into memory and disk-level operations
  - Optimizes execution time
- Disk-space management layer:
  - Stores data records on pages of contiguous memory blocks
  - Pages are fetched from disk into memory as requested using pre-fetching and page replacement policies

So, if we look quickly look at the relational data base which is known to all of you or most of you users application programs interact with RDBMs through SQL, right. So, it is the structured query language or SQL by which we interact with the user programs, etcetera.

So, there is a relational database management parser which transforms queries into memory and disk label operations and optimize the execution time. So, in any query, we need to optimize the execution time of the query, right. So, if it is a large data base like you, whether you do project before select join before or after select that makes a lot of difference; though the query may be same the query output will be same, but the execution time may vary to a great extent, right, like I have a huge 2 data bases like R1 say relational databases R1, R2 and I do some projection or selection of some of the things, right I select A1, A2 and then do a; then do the; join whether I do the join before or after makes the things like suppose; if I do the select on R1; the number of tuples come down from 1 million to say few 1000s. Similarly for R2, if I do a select on that; right. So, then joining is much less costlier. So, whether you do the join first or it said that becomes a thing that is a database optimization problem nothing to do specifically for cloud, but relational database allows you to optimize those things.

Disk space management layer, this another property that stores data records on pages of contiguous memory block. So, that the disk movement is minimized pages are fetched from the disk into memory as requested state using pre fetching and page replacement policies. So, this is another aspects of the things like one is looking at that property making it more efficient in the query processing, other aspect it make it more efficient in storage terms of things like nearby things if the query requires the some 5 tables if they are nearby store then the access rate is high. So, database file system layer.

(Refer Slide Time: 08:15)



Relational Databases Contd...

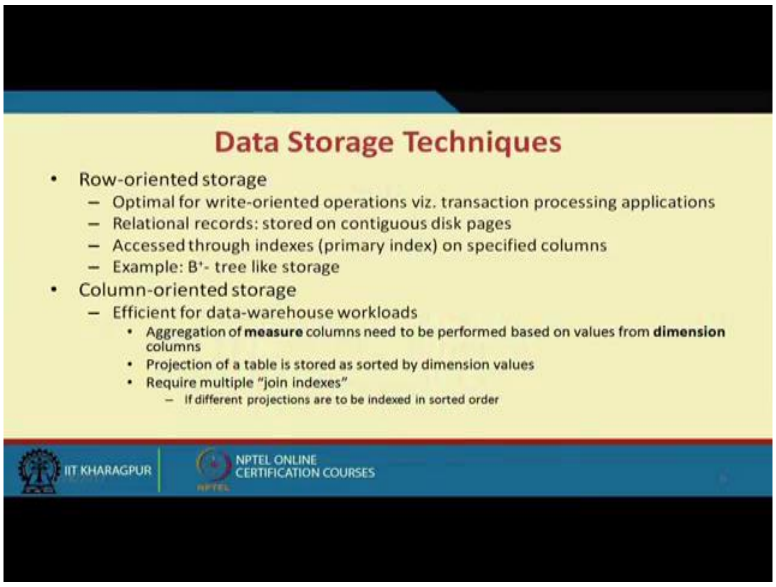
- Database file system layer:
  - Independent of OS file system
  - Reason:
    - To have full control on retaining or releasing a page in memory
    - Files used by the DB may span multiple disks to handle large storage
  - Uses parallel I/O systems, viz. RAID disk arrays or multi-processor clusters

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, previously we have seen that RDBM parser then disk space management layer then database file system layer. So, it is independent of OS file system, it is a separate file system. So, it is in order to have full control on retaining or realizing the page in the memory, files used by the DB or database may span multiple disk to handle large storages, right.

So, in other sense like if I dependent on the operating system for phase all those things then it is fine when your again database load is less if it is pretty large then the number of hope you take it text it becomes costly. So, what you need to do we need to do directly interact at the at the much lower level with the with the hardware or the available resources and that exactly this database file system layer tries to immolate uses parallel IO like we have heard about Raid disk Raid1, Raid2, Raid5, Raid 6eN type of things arrays or multiple clusters. So, which keeps a redundant redundancy into the thing. So, the your this failure down time is much less so; that means, is it is basically full failure proof implementation of the database.

(Refer Slide Time: 09:42)



Data Storage Techniques

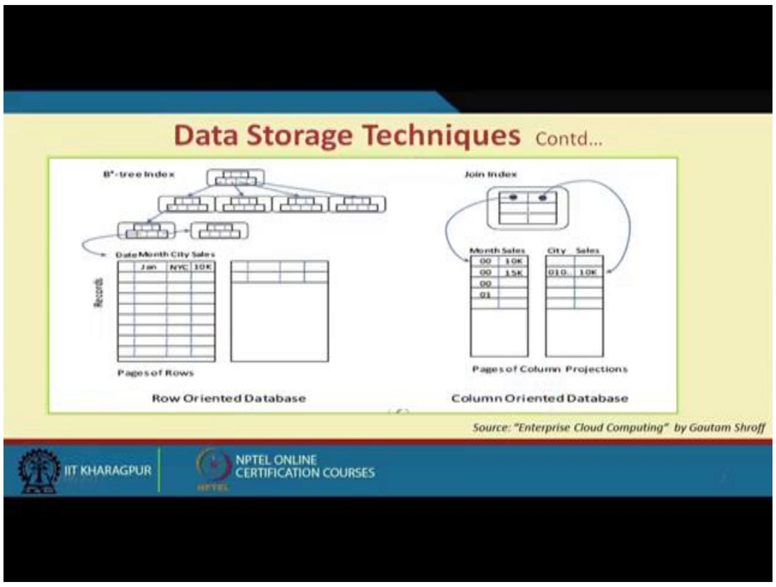
- Row-oriented storage
  - Optimal for write-oriented operations viz. transaction processing applications
  - Relational records: stored on contiguous disk pages
  - Accessed through indexes (primary index) on specified columns
  - Example: B^+- tree like storage
- Column-oriented storage
  - Efficient for data-warehouse workloads
    - Aggregation of **measure** columns need to be performed based on values from **dimension** columns
    - Projection of a table is stored as sorted by dimension values
    - Require multiple "join indexes"
      - If different projections are to be indexed in sorted order

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

So, usually the databases storage as row oriented that is we had tuples and its a set of row of the same schema optimal for write oriented operation the transaction processing applications, relational records stored in contiguous disk pages access through indexes primary key on specific columns, B plus tree is one of the favorite storage mechanisms for this sort of thing. Column oriented efficient for data warehouse workloads right. So, those who have gone through data warehouses. So, it is a high dimensional data huge volume of data and being collected and populated by different things. So, it is more of a warehouse, rather than a simple database. So, this is this column oriented storage are more suitable for data warehouse type of loads aggregate of measures where rather than individual data it is more of the analysis on analytics come into play. So, it is aggregation of measure columns need to be performed based on the values of the dimension columns. So, we are not going to the data warehouse. So, it has a different dimension tables and type of things and we need to the operations are more aggregate operations, right, we want to do some sort of analysis and type of things.

So, projection of a table is stored on as a stored on a dimension table dimension values in case of a column oriented require multiple join indexes if different projection are to be indexed in a sorted order right. So, it is; if it is a different-different thing because the organization may have different views for different type of data and need to be stored in that fashion.

(Refer Slide Time: 11:31)



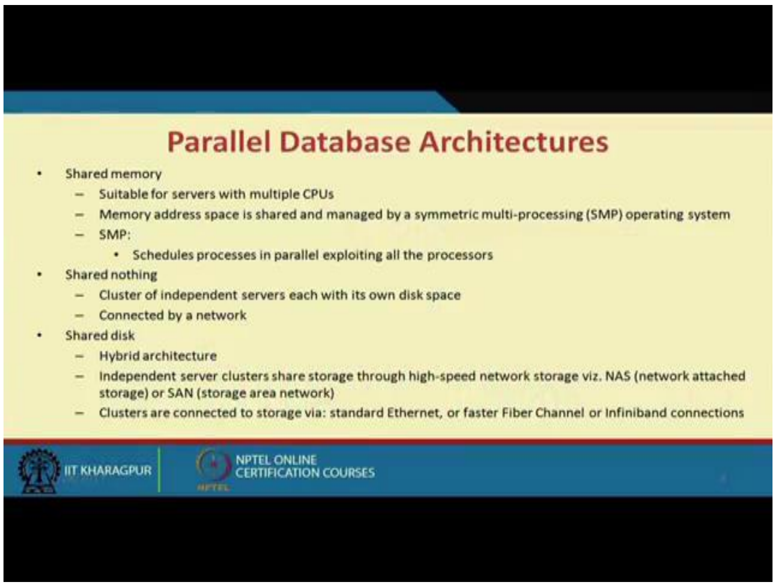
Data Storage Techniques Contd...

Source: "Enterprise Cloud Computing" by Gautam Shroff

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

So, data storage techniques as we have seen; it is B plus tree or join indexes. So, one is row oriented, other one is column oriented. So, this is row oriented data and this is column oriented data and we need to have a join index which allows this data to be linked to one another. So, these all these we will get in any standard database book or in standard literature; primarily as we are following that Gautam Shroff's Enterprise cloud computing book for this particular thing. So, that is why we have mentioned, but this is a very standard operation and you can get in any standard books.

(Refer Slide Time: 12:16)



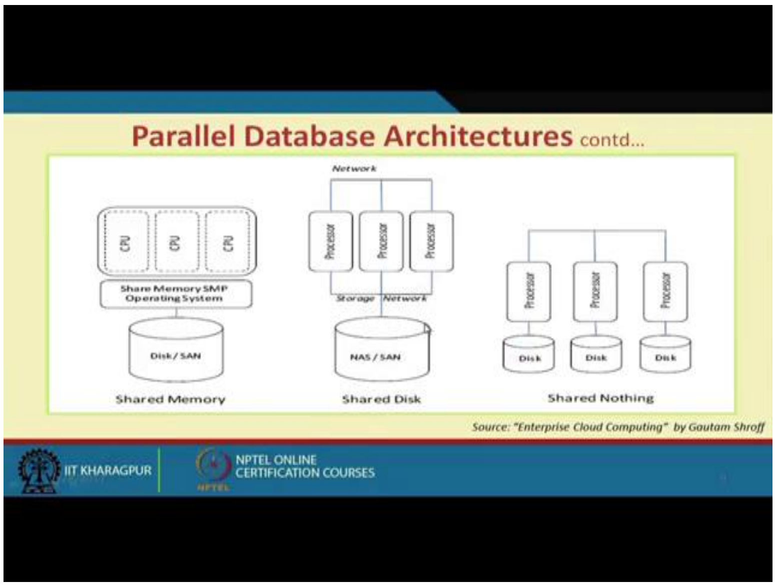
Parallel Database Architectures

- Shared memory
  - Suitable for servers with multiple CPUs
  - Memory address space is shared and managed by a symmetric multi-processing (SMP) operating system
  - SMP:
    - Schedules processes in parallel exploiting all the processors
- Shared nothing
  - Cluster of independent servers each with its own disk space
  - Connected by a network
- Shared disk
  - Hybrid architecture
  - Independent server clusters share storage through high-speed network storage viz. NAS (network attached storage) or SAN (storage area network)
  - Clusters are connected to storage via: standard Ethernet, or faster Fiber Channel or Infiniband connections

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

So, if we look at the parallel database architectures. So, it is broadly divided into 3 aspects one is shared memory one is shared nothing another is shared disk, right.

(Refer Slide Time: 12:30)


Parallel Database Architectures contd...

Source: "Enterprise Cloud Computing" by Gautam Shroff

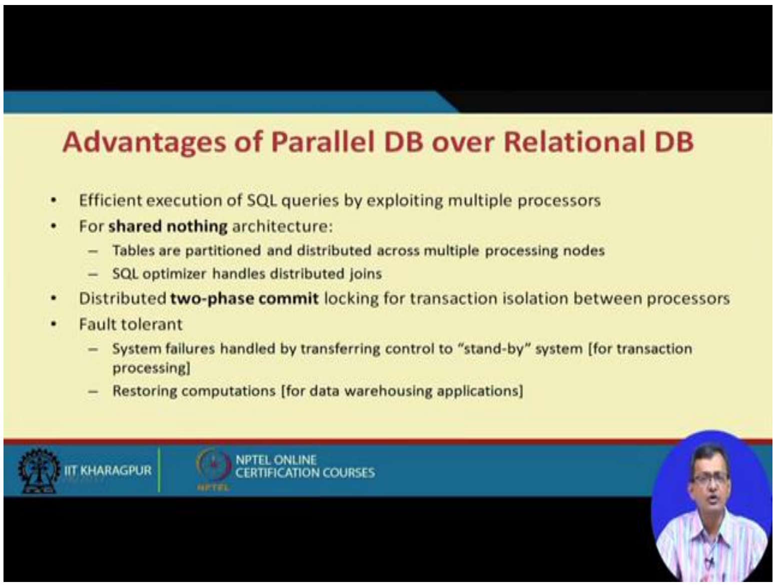
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, I just see the picture fast then come back. So, this is a typical structure of the shared memory, right. So, these processors different processors shared the memory, here it is a shared disk. So, different processors shared the disk, here we have shared nothing. So, individual processor has individual disk; so, in case of a shared memory suitable for servers with multiple CPUs. So, if there are multiple CPUs. So, if there are multiple CPUs memory address space is shared and managed by SMP operating systems like the memory address. This is shared among these SMPs and schedule processors in parallel exploiting the processors. So, it schedules small things so; that means, I have a shared memory space and I basically do a execution in a parallel mode.

So on the extreme other end is shared nothing. So, cluster independent servers with each of its having own disk space and connected by a network. So, at the with a back bone high speed network if any server shared its own disk space and then do the rest of the execution and if we look at that in between the thing is the shared disk like it is a hybrid architecture. So, to say independent server cluster storage through high speed network that can be NAS or SAN and clusters are connected to storage data via standard Ethernet fiber, etcetera what we have shown here. So, it is a shared storage and these different processor access this. So, based on your application type of parallelisms you need we can go for any of this structure.

So, here we see that it is more this more efficient if the memory things are more compact where in the other end we if the processors are individually working on separate data sets and there are machine to say then this could have been a advantage.

(Refer Slide Time: 14:32)



Advantages of Parallel DB over Relational DB

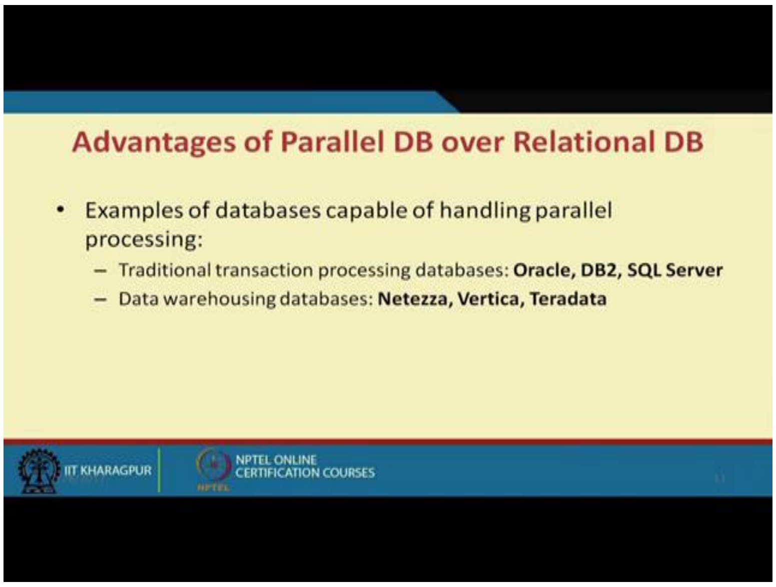
- Efficient execution of SQL queries by exploiting multiple processors
- For **shared nothing** architecture:
  - Tables are partitioned and distributed across multiple processing nodes
  - SQL optimizer handles distributed joins
- Distributed **two-phase commit** locking for transaction isolation between processors
- Fault tolerant
  - System failures handled by transferring control to "stand-by" system [for transaction processing]
  - Restoring computations [for data warehousing applications]

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, if we look at the advantages of parallel DB of relational database, if you do not want to put that; what are the features of relational parallel database structures which is more advantages for parallel this sort of operations, then the relational database efficient execution of SQL query by exploiting multiple processors, for shared nothing architecture tables partition and distributed across possessing table, right. So, happened that I can partition the table and every the data accountant in the table can be executed parallely they can be distributed in the different days and the processor can work that totally depends on your; what is your working mechanisms out there.

So, SQL optimizer handles this distributed joint. So, whenever we need to do some join then we need to fall on the; distribute your SQL optimizer. So, distributed 2 phase commit locking for transaction isolation between the processors. So, these are the some of the features, fault tolerant like system failures handled by transferring control to standby system. So, I can have different standby system or some with some protocol or some policy and then if there is a failure, then I can shift that particular execution to some of the standby system. So, that is possible in this sight of things and restoring computation for data though these are the things which are more required for data warehouse type of applications.

(Refer Slide Time: 16:15)



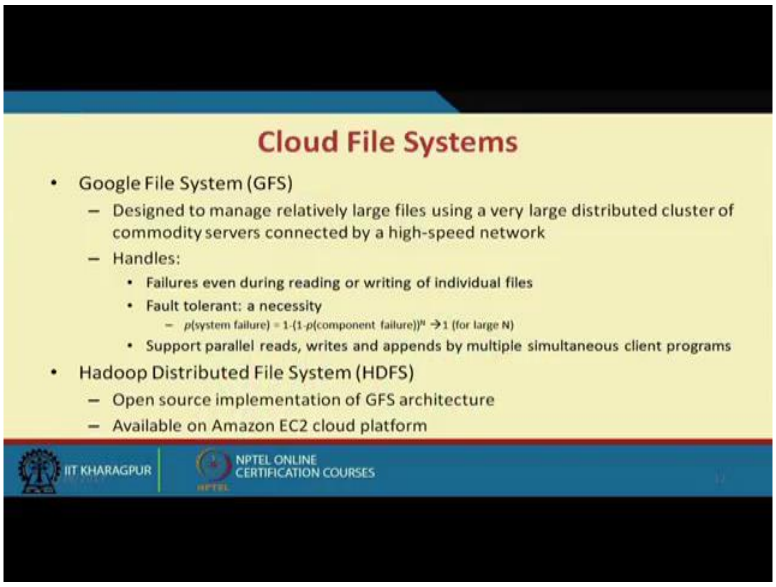
Advantages of Parallel DB over Relational DB

- Examples of databases capable of handling parallel processing:
  - Traditional transaction processing databases: **Oracle, DB2, SQL Server**
  - Data warehousing databases: **Netezza, Vertica, Teradata**

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, there are examples of databases capable of handling parallel processing traditional transaction processing things are oracle, DB2, SQL server data warehouse application are some of the Vertica, Teradata, Netezza; these are the some of the things which are more of a data warehouse type of database. Now with these background or with these things in our in our store what we say we look at that cloud file system.

(Refer Slide Time: 16:50)



Cloud File Systems

- Google File System (GFS)
  - Designed to manage relatively large files using a very large distributed cluster of commodity servers connected by a high-speed network
  - Handles:
    - Failures even during reading or writing of individual files
    - Fault tolerant: a necessity
      - $p(\text{system failure}) = 1-(1-p(\text{component failure}))^N \rightarrow 1$ (for large N)
    - Support parallel reads, writes and appends by multiple simultaneous client programs
- Hadoop Distributed File System (HDFS)
  - Open source implementation of GFS architecture
  - Available on Amazon EC2 cloud platform

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

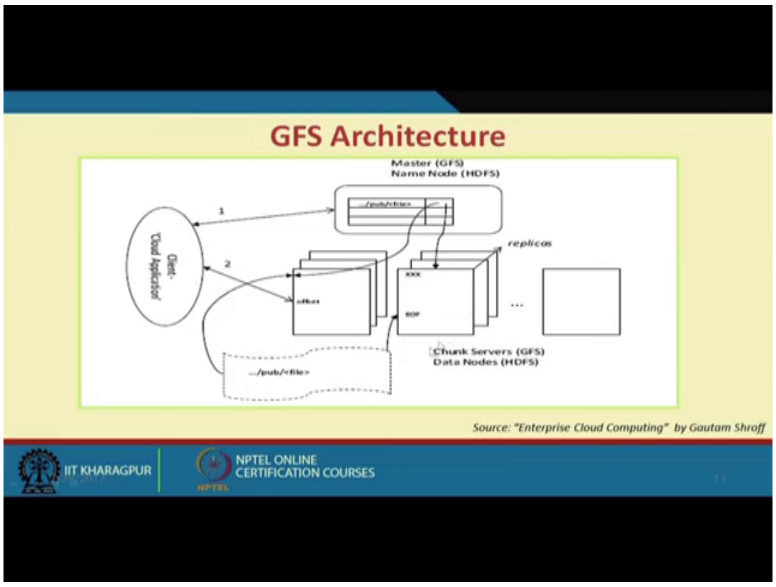
Now, as we understand it will not go something become totally we cannot through the whole thing out of the thing and start doing something new because this database has grown; they are fault tolerant, they are efficient we have raids and type of things we need to exploit some of the things and put some more philosophy of which behind the cloud.

So, one of the predominant thing is cloud file Google file system was GFS and back to back; we have a open source stuff called HDFS; Hadoop distributed file system. So, which is what we say someone to one mechanism set Google file system. So, Google file system, design to manage relatively large files using a very large distributed clusters of commodity servers connected by high speed things. So, it is whether GFS or HDFS, they are enable to work on very large data files which are distributed over this commodity servers; typically some of the things are Linux servers which are interconnected through a very high speed line.

So, they can handle failure even during read write of individual files, right, during the read-write operation if there is a failure it can handled. Fault tolerant it is definitely a necessity. So, if we have any that is any simple system term that *P(system failure)* probability of system failure is *1-(1-P(component failure))*^{N}. So, for if the N is pretty large, then you can say that we can go for that is the risk of this failure is minimum. So, supports parallel reads writes appends multiple simultaneous client program. So, it is parallel read parallel write and update by the client program and we have HDFS that is

Hadoop distributed file system which is open source implementation of GFS architecture available on Amazon EC2 cloud platform from. So, we have HDFS which is there.

(Refer Slide Time: 19:18)



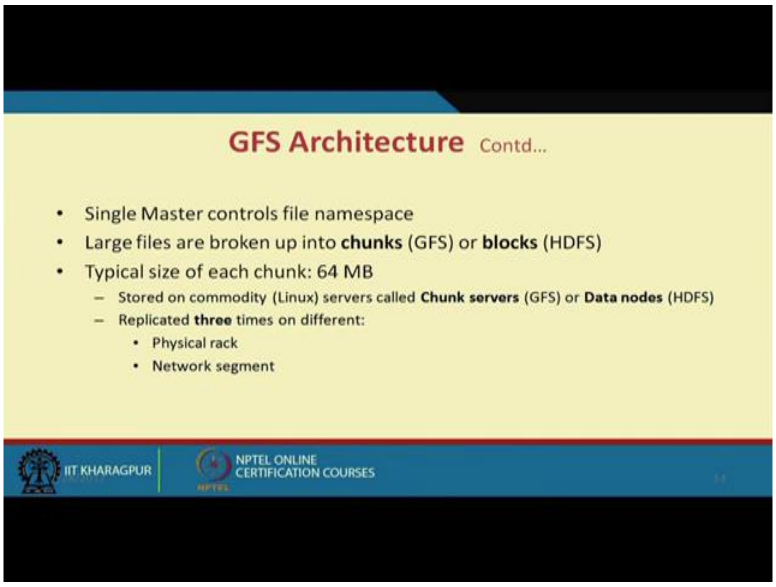
GFS Architecture

Source: "Enterprise Cloud Computing" by Gautam Shroff

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, if we have a big picture. So, that how a typical GFS are there. So, there are some of the components are there is master or the name nodes master node in GFS or name node is HDFS and there are client applications and we have different chunk server in case of GFS and data nodes in the case of HDFS in a typical cloud environment. So, single master controls the namespace.

(Refer Slide Time: 19:47)



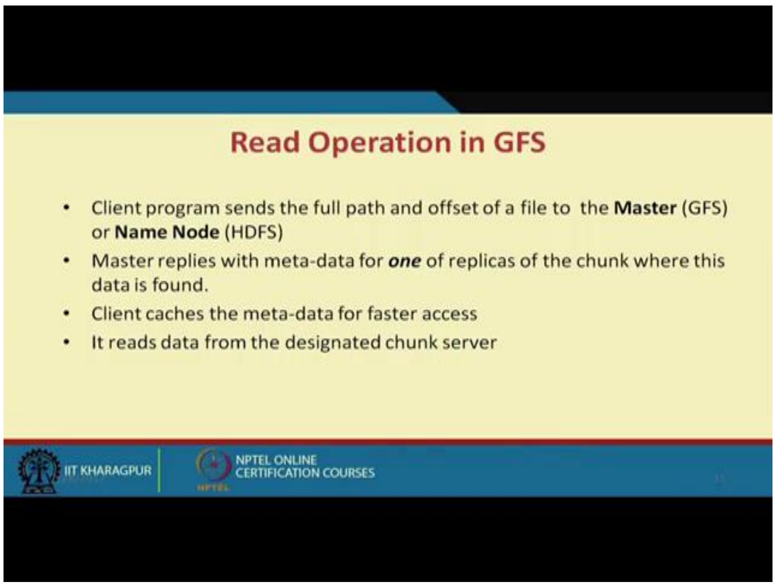
GFS Architecture Contd...

- Single Master controls file namespace
- Large files are broken up into **chunks** (GFS) or **blocks** (HDFS)
- Typical size of each chunk: 64 MB
  - Stored on commodity (Linux) servers called **Chunk servers** (GFS) or **Data nodes** (HDFS)
  - Replicated **three** times on different:
    - Physical rack
    - Network segment

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, logically a single master is there which control the namespace. So, namespace is important because it gives us that how there are stored; how data can be referred; it is more of a; it may modes of a meta-data sort of information which is controlled by the master large files are broken into chunks, in case of a GFS and block; what we called in case of a HDFS stored on commodity server, typically Linux servers called chunk servers in GFS and data nodes in HDFS, so replicated 3 times on different physical rack network segment. So, this chunk; so, what we have? We have the GFS or HDFS in the things below that we are having a chunk servers which are basically Linux servers chunk server or data nodes in the things which are the main custodian of the data and they are the every data D_i is replicated on different 3 times at least 3 time on different physical rack and network segments.

(Refer Slide Time: 21:11)



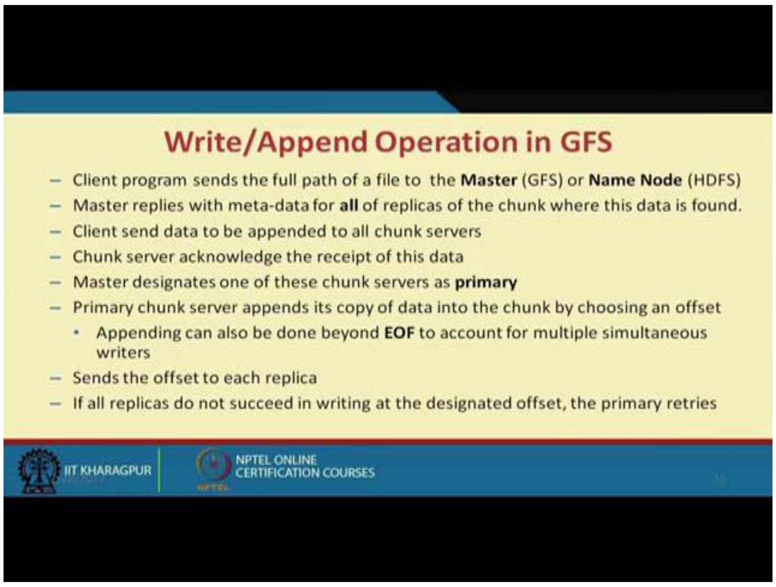
Read Operation in GFS

- Client program sends the full path and offset of a file to the **Master** (GFS) or **Name Node** (HDFS)
- Master replies with meta-data for ***one*** of replicas of the chunk where this data is found.
- Client caches the meta-data for faster access
- It reads data from the designated chunk server

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

So, if you look at the read operation in GFS, client program sends the full path offset of a file to the master, right where it wants to read or name node in case of HDFS. So, we will refer the GFS master node and which is back to back when we it is refer to the name node in HDFS master replies on meta-data for one of the replicas of the chunk where these data is found, right, client caches the meta-data for faster access. It reads the data from the designated chunk server. So, master from the master; it gets that and gets the mirror this meta-data and from there it basically access this chunk server.

(Refer Slide Time: 22:01)



Write/Append Operation in GFS

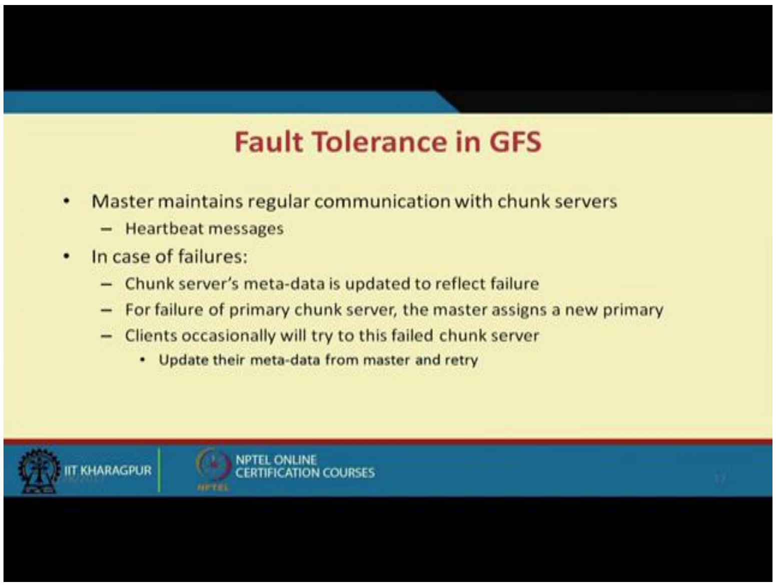
- Client program sends the full path of a file to the **Master** (GFS) or **Name Node** (HDFS)
- Master replies with meta-data for **all** of replicas of the chunk where this data is found.
- Client send data to be appended to all chunk servers
- Chunk server acknowledge the receipt of this data
- Master designates one of these chunk servers as **primary**
- Primary chunk server appends its copy of data into the chunk by choosing an offset
  - Appending can also be done beyond **EOF** to account for multiple simultaneous writers
- Sends the offset to each replica
- If all replicas do not succeed in writing at the designated offset, the primary retries

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

So, for read operation any of these chunk server or replicated chunk server will do where write append operation in GFS is little tricky, client program sends a full path of file to the master GFS on name node HDFS right, the master replies on the meta-data for all replicas of the chunks where the data is found. The client send data to be appended into the all chunk servers; chunk server acknowledges the receipt of the data, master designate one of the chunk server as primary, the primary chunks server appends its copy of the data into the chunk by offset choosing an offset, right. So, that it do it appending; appending can also be done beyond end of file to account for the multiple simultaneous, right.

So, this is a pretty interesting thing that even if you can have append end of EOF beyond EOF because there are simultaneous writers which are writing and it basically consolidated at later stage. Sends offset to the replica, if all replica do not success in writing in the designated offset, the client retries, right. So, the all offset; so, idea is that whenever I am looking for a data, I need to know that for all the 3 replicas, it should be at the same offset ideally. So, that I the read processed as there is no delay in that things because once its calculates it is directly access the other chunks on that offset, right.

(Refer Slide Time: 23:42)



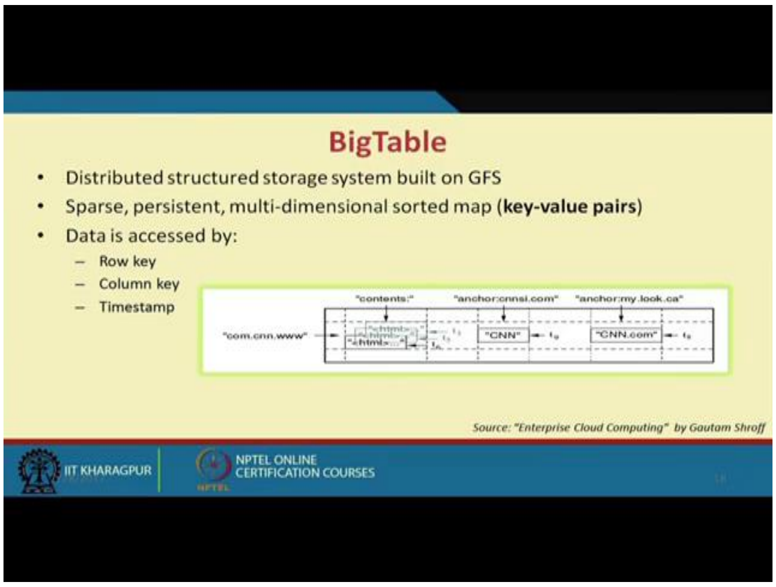
Fault Tolerance in GFS

- Master maintains regular communication with chunk servers
  - Heartbeat messages
- In case of failures:
  - Chunk server's meta-data is updated to reflect failure
  - For failure of primary chunk server, the master assigns a new primary
  - Clients occasionally will try to this failed chunk server
    - Update their meta-data from master and retry

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

So, fault tolerant in Google file system; the master maintains regular communication with the chunk server what we say heart beat messages sort of a are you alive type of thing and in case of a failure chunk server meta-data is updated to reflect failure for failure of primary chunk server the master assigns a new primary clients occasionally we will try to this failed we will try to this failed chunk server, update their meta-data from the master and retry. So, in case of a failure the chunk server meta-data after reflect the failure. So, the chunk server meta-data says that there is a failure. So, the next time you do not allocate or like that and for failure of the primary server itself, the master assigns a new primary. So, it assigns a new primary to work on the thing.

(Refer Slide Time: 24:47)



BigTable

- Distributed structured storage system built on GFS
- Sparse, persistent, multi-dimensional sorted map (**key-value pairs**)
- Data is accessed by:
  - Row key
  - Column key
  - Timestamp

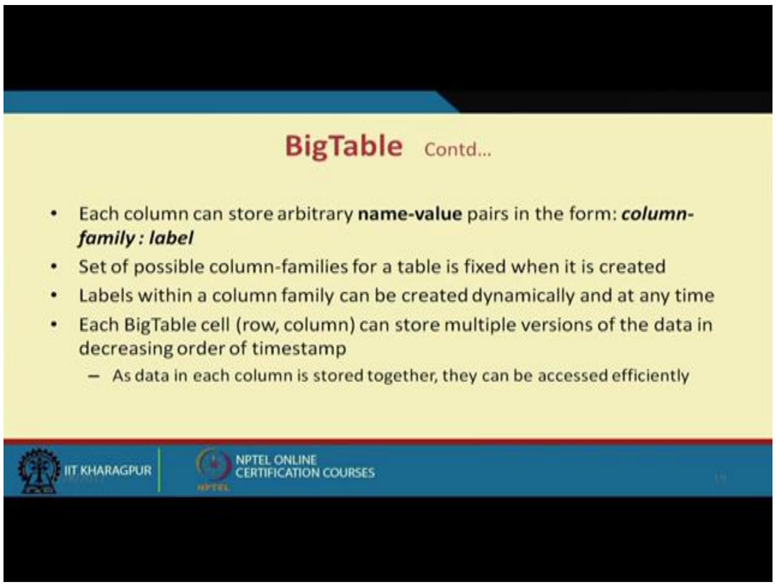
"contents:"
"anchor:cnnsi.com"
"anchor:my.look.ca"
"com.cnn.www"
"<html>..."
"CNN"
"CNN.com"

Source: "Enterprise Cloud Computing" by Gautam Shroff

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

And update the clients; occasionally we will try to this failed chunk server because it will be flagged, right. Now another related stuff is big data or related concept of big data, distributed structure storage 5 system build on GFS, right. So, it is build; it is a structure distributed structure storage file system it is build on GFS, right. So, data is accessed by row key, column key, timestamp. So, if you look at. So, it is a multiple instances are stored. So, there is a time key column key and of course, say row key which says that where the data is there.

(Refer Slide Time: 25:26)



BigTable Contd...

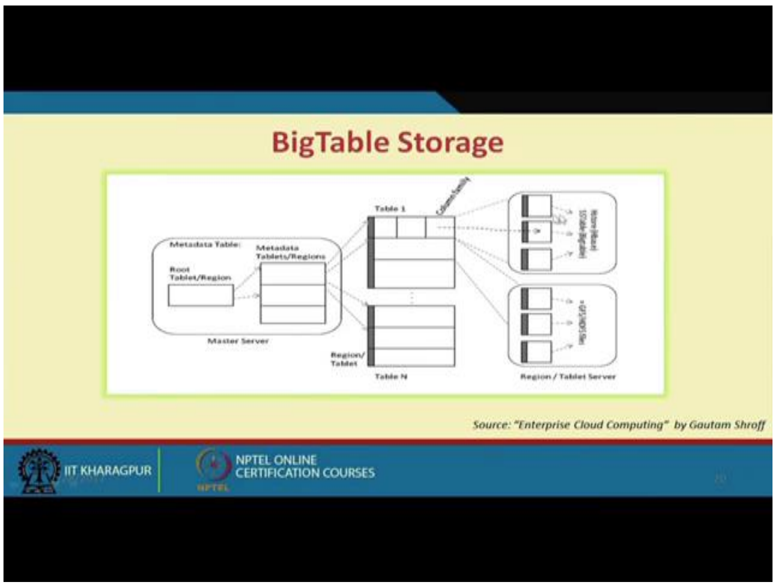
- Each column can store arbitrary **name-value** pairs in the form: ***column-family : label***
- Set of possible column-families for a table is fixed when it is created
- Labels within a column family can be created dynamically and at any time
- Each BigTable cell (row, column) can store multiple versions of the data in decreasing order of timestamp
  - As data in each column is stored together, they can be accessed efficiently

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, in big table each column can store arbitrary name value pair in the form of column family and label right. So, here if you can see that these are column families and it is labeled and they store a name value pair. Set of possible column family is of a table is fixed when it is created. So, which are the different column families will be there. So, that is somewhat fix. Labels within a column family can be created dynamically and at any time. So, I can recreate or create the table each big table cell row and column can store multiple versus of the data in decreasing order of the time stamp.

So; that means, it is the chronology is meant it in that fashion. So, it is multiple persons are stored in a decreasing time stamp.

(Refer Slide Time: 26:20)

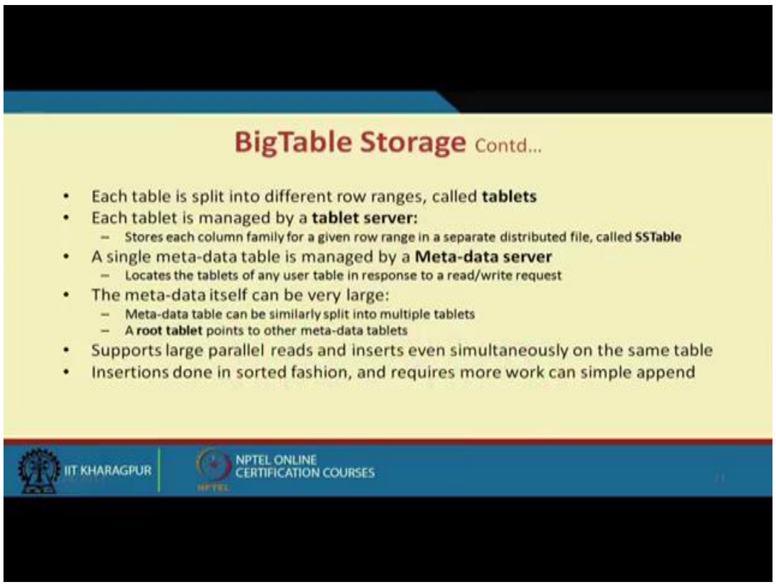


BigTable Storage

Source: "Enterprise Cloud Computing" by Gautam Shroff

So, again we see these things. So, there are different tables there are different tablets which are referred to this table and it is a hierarchical structure and we have a master server it is primarily a registry or a meta-data repository. So, each table in big data is split into rangers called tablets, each table is manage by tablet server. So, its stores each column family for a given row range in a separate distributed file called SS table. So, this type of management goes into play. So, that my access rate end of the day the access rate or will be pretty high.

(Refer Slide Time: 27:03)



BigTable Storage Contd...

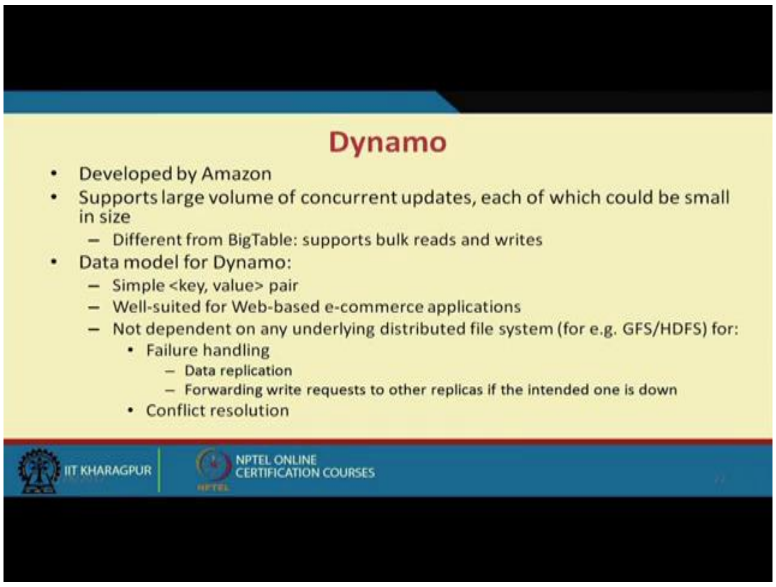
- Each table is split into different row ranges, called **tablets**
- Each tablet is managed by a **tablet server:**
  - Stores each column family for a given row range in a separate distributed file, called **SSTable**
- A single meta-data table is managed by a **Meta-data server**
  - Locates the tablets of any user table in response to a read/write request
- The meta-data itself can be very large:
  - Meta-data table can be similarly split into multiple tablets
  - A **root tablet** points to other meta-data tablets
- Supports large parallel reads and inserts even simultaneously on the same table
- Insertions done in sorted fashion, and requires more work can simple append

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

So, a single meta-data table is maintained by the maintained by the many meta-data server the meta-data itself can be very large. So, the meta-data while storing this itself can be very large, in that case; it is again broken down into split into different tablets a root tablet points to the other meta-data tablets.

So, if the meta-data are repository a pretty large, it is again broken down into different tablets and there is a root tablet which coordinates with your meta-data; this tablets and want to real a want to emulate or realize that meta-data services. Supports large parallel reads and inserts even simultaneously on the same table, insertion done in sorted fashion, requires more work can be more work than the simple append, right. There is true for a other databases also because once you insert it is basically you need to push the data aside and create a insertion point where as in case of a append you are putting data at the end of the end of that storage or data or the tables.

(Refer Slide Time: 28:22)



Dynamo

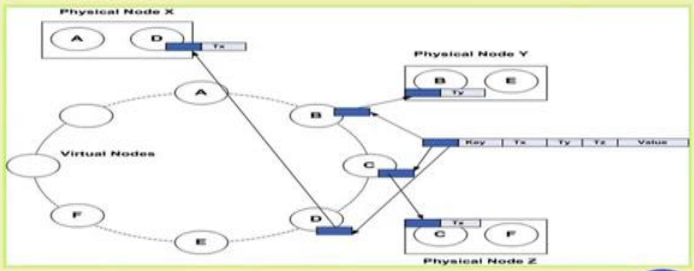
- Developed by Amazon
- Supports large volume of concurrent updates, each of which could be small in size
  - Different from BigTable: supports bulk reads and writes
- Data model for Dynamo:
  - Simple <key, value> pair
  - Well-suited for Web-based e-commerce applications
  - Not dependent on any underlying distributed file system (for e.g. GFS/HDFS) for:
    - Failure handling
      - Data replication
      - Forwarding write requests to other replicas if the intended one is down
    - Conflict resolution

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, Dynamo; it is developed by Amazon that supports large volume or concurrent updates each of which can be small in size different from big table supports bulk read and writes right end is. So, data model for Dynamo; it is a simple key value pair well suited for web based e-commerce type of applications and not dependent underlining distributed file systems, right for failure handling conflict resolution, etcetera, they do it their self.

(Refer Slide Time: 28:59)

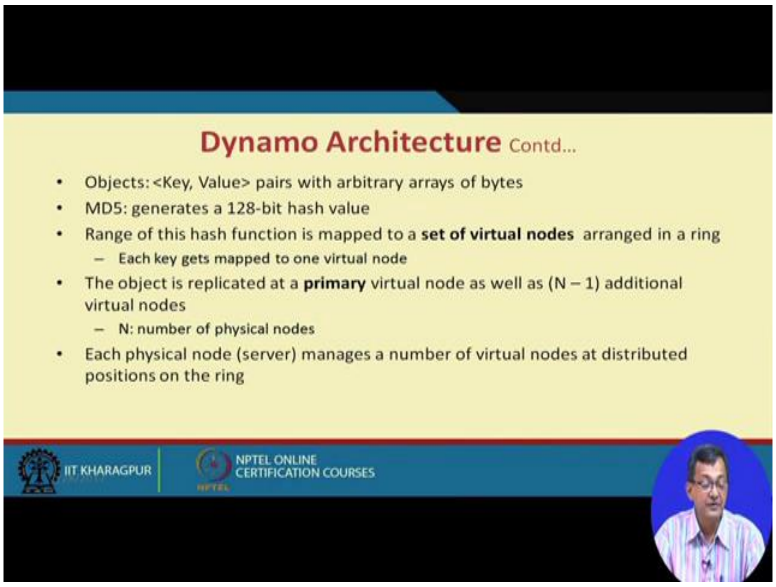
Dynamo Architecture



IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, this is typical architecture of the Dynamo where there are several virtual nodes and different physical nodes and they are logical connectivity are zone.

(Refer Slide Time: 29:13)



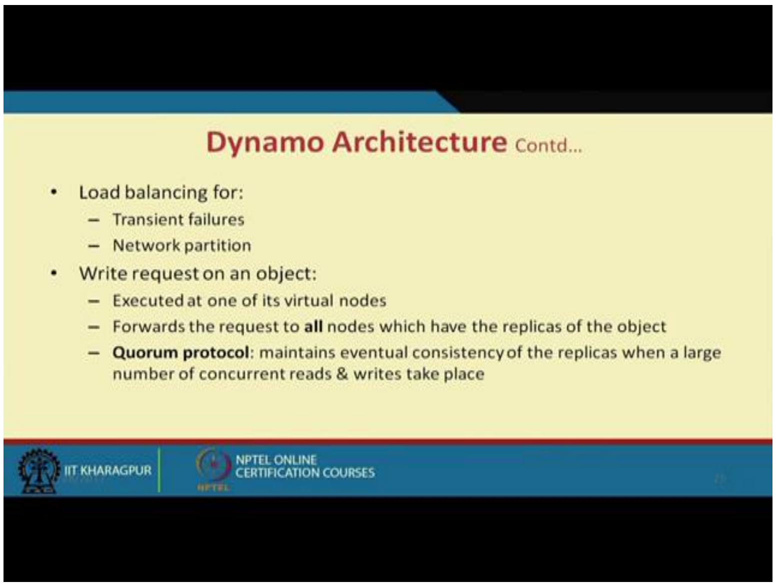
Dynamo Architecture Contd...

- Objects: <Key, Value> pairs with arbitrary arrays of bytes
- MD5: generates a 128-bit hash value
- Range of this hash function is mapped to a **set of virtual nodes** arranged in a ring
  - Each key gets mapped to one virtual node
- The object is replicated at a **primary** virtual node as well as (N – 1) additional virtual nodes
  - N: number of physical nodes
- Each physical node (server) manages a number of virtual nodes at distributed positions on the ring

So, if you look at the Dynamo architecture. So, it is a key value pair with arbitrary value key value pair with arbitrary arrays of bytes like it uses MD 5 generates a one twenty eight bit1hash table hash value.

So, it basically try to map that were virtual node will be mapping to by using this has function. Range of this has function is mapped as we are discussing that set of virtual nodes arrange in a ring type of thing. The object is replicated as a primary virtual node as well *N-1* additional virtual nodes, the N is the number of physical nodes. So, that any the objectives replicated into the things. Each physical nodes are managed is a number of virtual node at a distributed position on the ring. So, if you look at that this physical node server they are basically linked with this virtual node server.

(Refer Slide Time: 30:12)



Dynamo Architecture Contd...

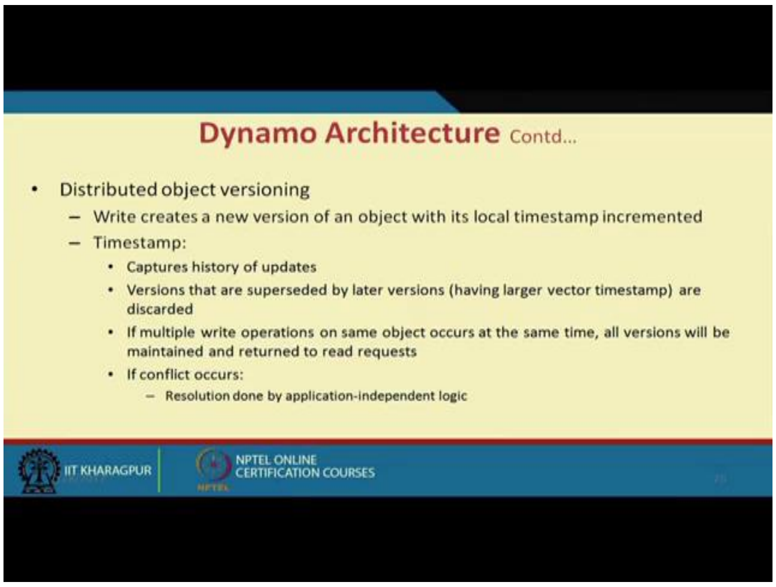
- Load balancing for:
  - Transient failures
  - Network partition
- Write request on an object:
  - Executed at one of its virtual nodes
  - Forwards the request to **all** nodes which have the replicas of the object
  - **Quorum protocol**: maintains eventual consistency of the replicas when a large number of concurrent reads & writes take place

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Dynamo architecture, load balancing for transient failure network partition this can handle write request on object that executed at one of its virtual nodes, right.

Forward all the request to all other nodes; it is executed one of the virtual node and say in all other all other nodes which have a replicas of the object so; that means, if I am a object; if it is replicated into another *N-1* node. So, one is updated rest are being communicate. So, there is a quorum protocol that maintains eventual consistency of the replicas when a large number of concurrent reads and writes going on. So, this quorum tries to find out that which are the minimum level of replica will be there to handle this large read write of person.

(Refer Slide Time: 31:02)

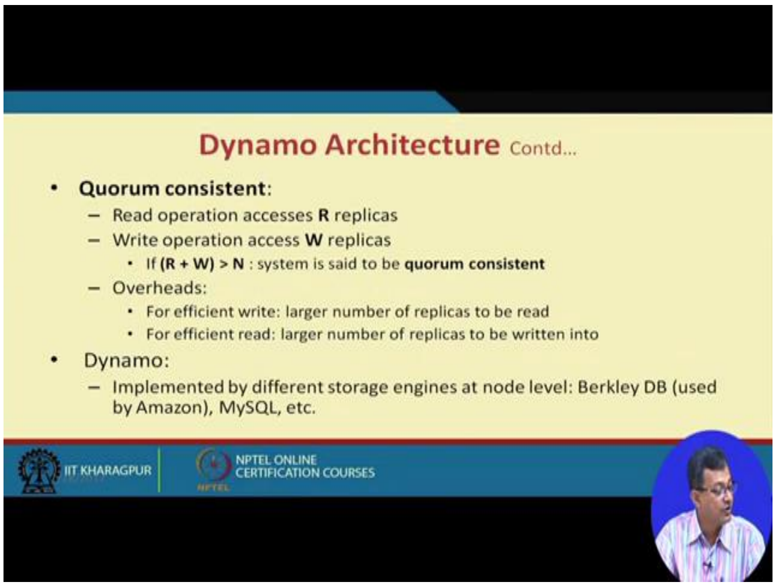


Dynamo Architecture Contd...

- Distributed object versioning
  - Write creates a new version of an object with its local timestamp incremented
  - Timestamp:
    - Captures history of updates
    - Versions that are superseded by later versions (having larger vector timestamp) are discarded
    - If multiple write operations on same object occurs at the same time, all versions will be maintained and returned to read requests
    - If conflict occurs:
      - Resolution done by application-independent logic

So, in next, we are having this dynamo distributed object version right creates a new version of the objects in his local time stamp created. There are algo for column consistency.

(Refer Slide Time: 31:12)

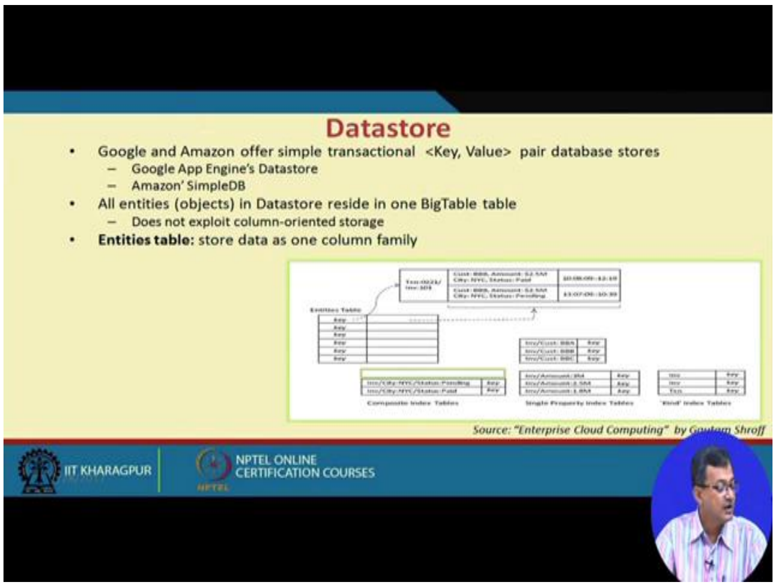


Dynamo Architecture Contd...

- **Quorum consistent**:
  - Read operation accesses **R** replicas
  - Write operation access **W** replicas
    - If **(R + W) > N** : system is said to be **quorum consistent**
  - Overheads:
    - For efficient write: larger number of replicas to be read
    - For efficient read: larger number of replicas to be written into
- Dynamo:
  - Implemented by different storage engines at node level: Berkley DB (used by Amazon), MySQL, etc.

So, read operation R; write operation E. So, read plus write operation should be greater than any of the system is quorum consistent there are overheads which will be coming there is a efficient write large number of replicas are to be read and if it is for a, b, c and read large number of large number of replicas need to be written. So, these are the 2 things which are they are; so, it is implemented by different storage engines at node level Berkley DB used by Amazon and can be implemented to using MySQL and etcetera.

(Refer Slide Time: 31:51)



Datastore

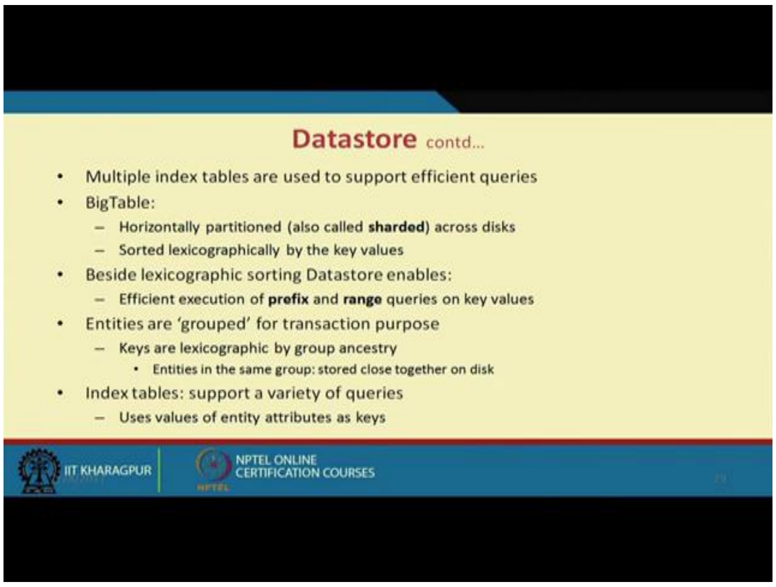
- Google and Amazon offer simple transactional <Key, Value> pair database stores
  - Google App Engine's Datastore
  - Amazon' SimpleDB
- All entities (objects) in Datastore reside in one BigTable table
  - Does not exploit column-oriented storage
- **Entities table:** store data as one column family

Source: "Enterprise Cloud Computing" by Gautam Shroff

Another; the final concept what we are having is the data store. Google and Amazon of a simple traditional key value pair database stores, right, Google app engines data store in case of Amazon what we say simple DB; all entities objects in the data store reside on in one big table, right.

Data store exploit column oriented storage right, data store as I mean store data as a column families. So, unlike our rational traditional thing is a more of a row family or tuple based it is called column family.

(Refer Slide Time: 32:30)



Datastore contd...

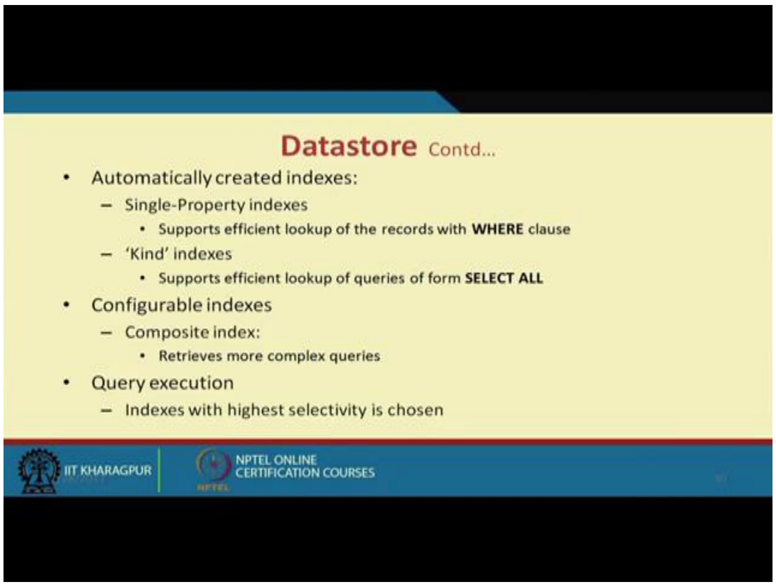
- Multiple index tables are used to support efficient queries
- BigTable:
  - Horizontally partitioned (also called **sharded**) across disks
  - Sorted lexicographically by the key values
- Beside lexicographic sorting Datastore enables:
  - Efficient execution of **prefix** and **range** queries on key values
- Entities are 'grouped' for transaction purpose
  - Keys are lexicographic by group ancestry
    - Entities in the same group: stored close together on disk
- Index tables: support a variety of queries
  - Uses values of entity attributes as keys

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

So, there are several advantages or several features or characteristics like multiple index tables are used to support efficient query. Big table horizontally partitioned call sharded and across the disk whereas, stored lexicographically in the key values other thing. Beside lexicographic sorting of the data enables there is a execution of prefix and range queries on key values entities are grouped for transactional purpose because if there is if when we are having transaction. So, that is a set of entities which are accessed in a more frequent way and index table to support varied varieties of queries.

So, we can have different indexes or different type of queries. So, it is not we should understand is not a simple a low a database it is a large database. So, in order to do that; I cannot churn the whole database. So, need to slice them appropriately. So, that based on the different variety different queries it can be executed more efficiently.

(Refer Slide Time: 33:37)



Datastore Contd...

- Automatically created indexes:
  - Single-Property indexes
    - Supports efficient lookup of the records with **WHERE** clause
  - 'Kind' indexes
    - Supports efficient lookup of queries of form **SELECT ALL**
- Configurable indexes
  - Composite index:
    - Retrieves more complex queries
- Query execution
  - Indexes with highest selectivity is chosen

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

And there are few more properties like automatically it creates indexes single property index or there is a kind index supports the efficient lookup queries of form select all type of things the configurable in indexes and there is a query execution indexes with highest selectivity is chosen, right. So, it is when we do the query execution.

So, with this we will stop our discussion here. So, what we tried to discuss over see is there different aspects we have the notion of our traditional databases which is established, fault tolerant, efficient and there are different mechanism to do that. So, we have we have also already this parallel execution things and its present. So, when we deal with a large volume of data in the cloud which are likely to be there, then what is are the different aspects we need to look at. So, we may not be able to follow the this column oriented or tuple oriented relational database we need to a sorry row oriented database we need to four for column oriented data base and there are different file system like GFS, HDFS and over that this data store Dynamo and your simple DB and those things what which are being implemented by various inter cloud service providers CSPs for efficient storage access, nread write execution of very very large databases.

Thank you.