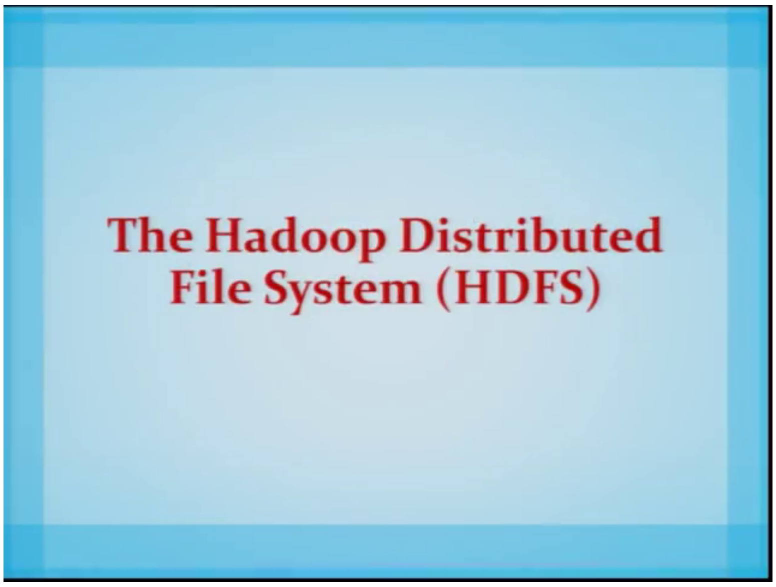
Distributed Systems
Dr. Rajiv Misra
Department of Computer Science and Engineering
Indian Institute of Technology, Patna

Lecture – 22
Case Studies HDFS

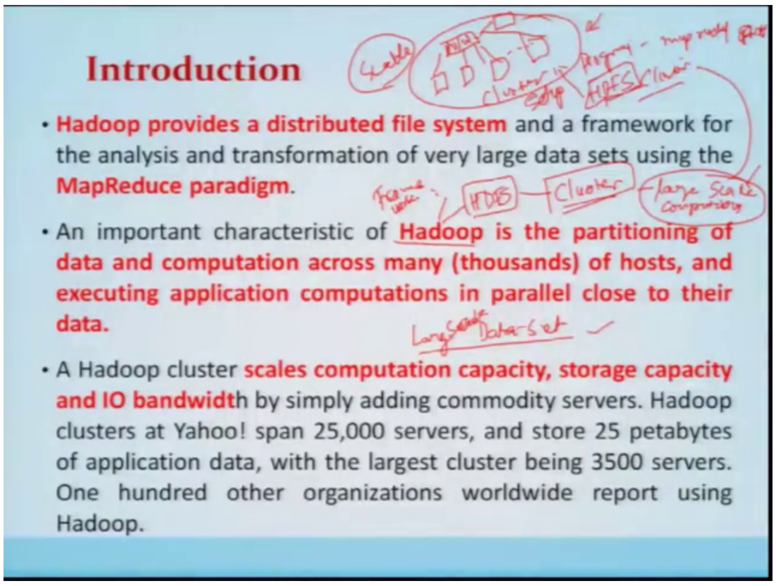
(Refer Slide Time: 00:16)



The Hadoop Distributed File System (HDFS)

HDFS and spark,

(Refer Slide Time: 00:22)



Introduction

- **Hadoop provides a distributed file system** and a framework for the analysis and transformation of very large data sets using the **MapReduce paradigm**.
- An important characteristic of **Hadoop is the partitioning of data and computation across many (thousands) of hosts, and executing application computations in parallel close to their data.**
- A Hadoop cluster **scales computation capacity, storage capacity and IO bandwidth** by simply adding commodity servers. Hadoop clusters at Yahoo! span 25,000 servers, and store 25 petabytes of application data, with the largest cluster being 3500 servers. One hundred other organizations worldwide report using Hadoop.

The Hadoop distributed file system HDFS, introduction Hadoop provides a distributed file system and a framework for the analysis and transformation of very large data sets using map reduce paradigm. An important characteristic of Hadoop is the partitioning of data and computation across many thousands of hosts, and executing application computations in parallel close to their data.

A Hadoop cluster scales computation capacity, storage capacity and IO bandwidth by simply adding commodity servers. Hadoop clusters at yahoo spans 25000 servers, and store 25 petabytes of application data with the largest cluster being 3500 servers 100 other organization worldwide report using Hadoop. So, let me give you a more intuitive notion of Hadoop file system and Hadoop environment. So, Hadoop is an environment where HDFS is a part of the Hadoop.

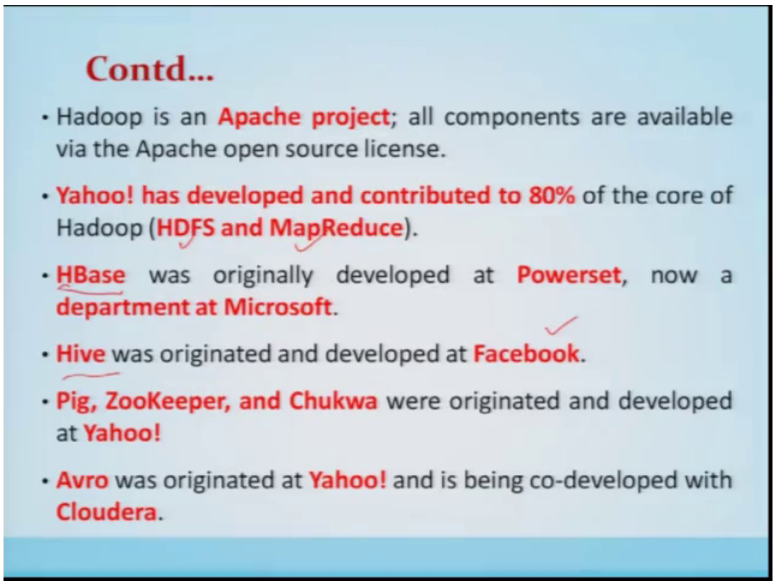
So, Hadoop also contains map reduce and other different application which are going to be used in this particular framework which is called a Hadoop. So, Hadoop file system is a particular file system which basically will be using the cluster setup for large scale computations, these computations will be designed using the programming environment such as map reduce or a spark.

So, when we say large scale computation; that means, the data set is of large scale data sets. So, this particular platform is good enough to support the computation of a large data set size. So, the cluster is basically nothing, but a set of hosts which are connected through the ne2rk. So, it is scalable in the sense as we grow the number of nodes without any problem of ne2rk bottlenecks this particular kind of cluster set up is scalable.

In contrast to a particular system becoming more powerful by adding more hardware it has a bottleneck of the scalability here the scalability is basically ensured in this particular kind of application. In this particular setup how this particular large scale computation can be made applicable, how this can be exploited for large scale computation. So, we are going to discuss the first the Hadoop environment and in particular the Hadoop distributed file system which will exploit or which will be running over this cluster setup.

So, HDFS will exploit this kind of; that means, cluster setup and this will be a point of discussion in this particular lecture.

(Refer Slide Time: 03:57)

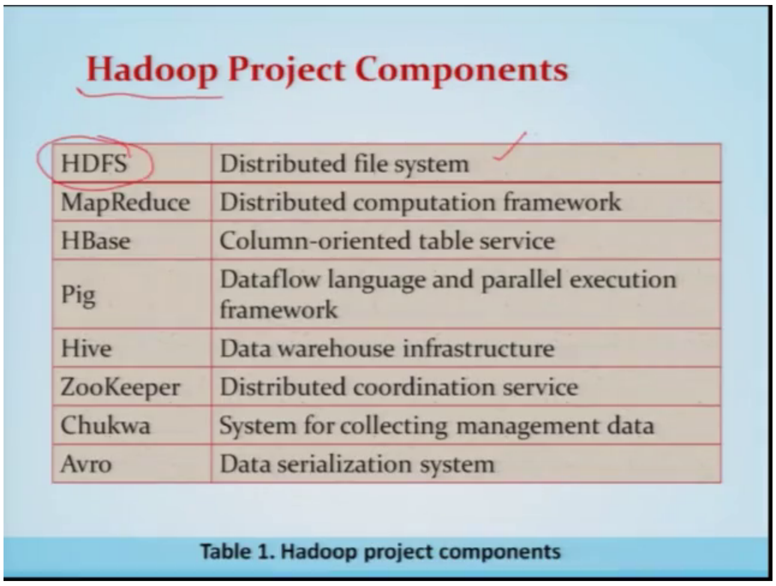


Contd...

- Hadoop is an **Apache project**; all components are available via the Apache open source license.
- **Yahoo! has developed and contributed to 80%** of the core of Hadoop (**HDFS and MapReduce**).
- **HBase** was originally developed at **Powerset**, now a **department at Microsoft**.
- **Hive** was originated and developed at **Facebook**.
- **Pig, ZooKeeper, and Chukwa** were originated and developed at **Yahoo!**
- **Avro** was originated at **Yahoo!** and is being co-developed with **Cloudera**.

So, Hadoop is an apache project all components are available via the Apache open source license. Yahoo has developed and contributed to 80 percent of Hadoop which comprises of HDFS map reduce and other components are also there such as, H base was originally developed at powerset, now a department at Microsoft such as hive also is developed at the facebook pig zookeeper and all these particular different variants are basically using this Hadoop file system and its basically the applications.

(Refer Slide Time: 04:35)



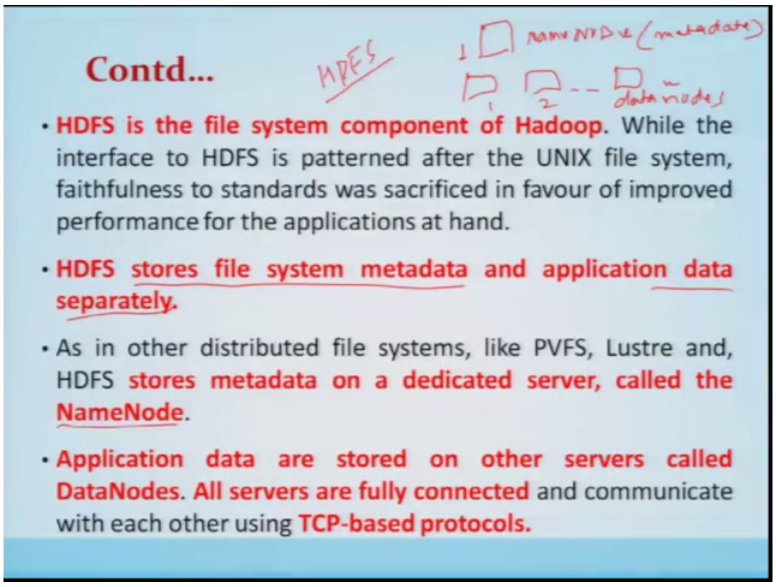
Hadoop Project Components

HDFS	Distributed file system
MapReduce	Distributed computation framework
HBase	Column-oriented table service
Pig	Dataflow language and parallel execution framework
Hive	Data warehouse infrastructure
ZooKeeper	Distributed coordination service
Chukwa	System for collecting management data
Avro	Data serialization system

Table 1. Hadoop project components

So, HDFS is a Hadoop file, Hadoop distributed file system is a part of Hadoop project. So, we are going to discuss this particular component here in this particular lecture.

(Refer Slide Time: 04:58)


HDFS
1 name NODE (metadata)
1 2 -- data nodes

Contd...

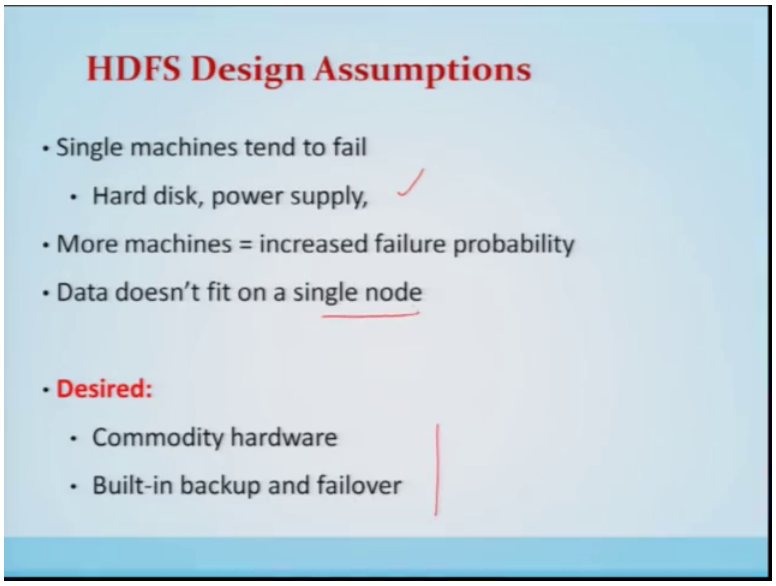
- **HDFS is the file system component of Hadoop.** While the interface to HDFS is patterned after the UNIX file system, faithfulness to standards was sacrificed in favour of improved performance for the applications at hand.
- **HDFS stores file system metadata and application data separately.**
- As in other distributed file systems, like PVFS, Lustre and, HDFS **stores metadata on a dedicated server, called the NameNode.**
- **Application data are stored on other servers called DataNodes. All servers are fully connected** and communicate with each other using **TCP-based protocols.**

So, HDFS is a file system, component of Hadoop while the interface to HDFS is patterned after the UNIX file system faithfulness to the standards were sacrified in favour of improved performance of for the applications at hand.

HDFS stores file system metadata and application data separately, with that we will see how this particular data and metadata are 2 separate entities here in HDFS and it manages them as in the other distributed file system like PVFS, Lustre and HDFS stores metadata on a dedicated server called the name node. So, there is a server which is called a name node in HDFS file system. So, name mode stores the metadata.

So, application data are stored on the other servers called the data nodes there will be a data nodes, many data nodes 1, 2 and so on up to n, but here there is only one name node all the servers are fully connected and communicate with each other using TCP based protocol. So, HDFS design assumption single machine tends to fail that is it is prone to fail, failing due to the different components also failing like hard disk power supply and so on.

(Refer Slide Time: 06:.35)

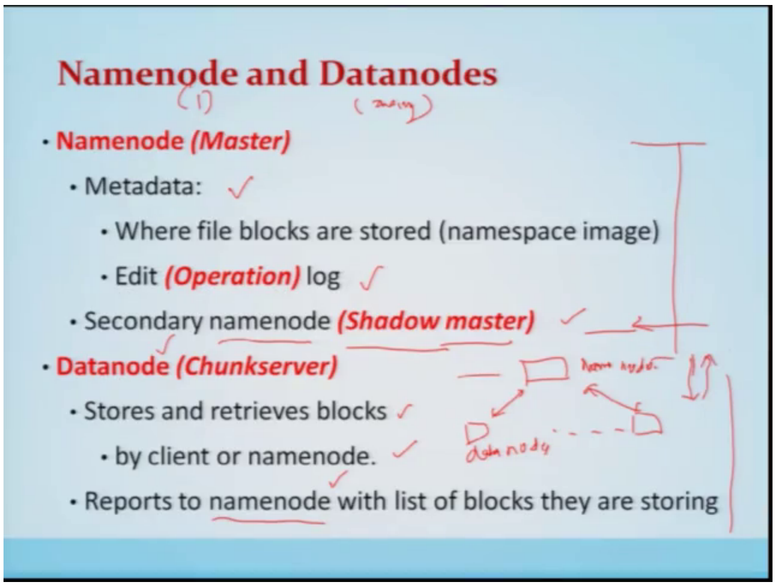


HDFS Design Assumptions

- Single machines tend to fail
  - Hard disk, power supply,
- More machines = increased failure probability
- Data doesn't fit on a single node
- **Desired:**
  - Commodity hardware
  - Built-in backup and failover

So, more machines means the increase failure probability will be there and data also does not fits in a single node. So, basically the so this will is a motivation to have a cluster which is basically a scalable and also is basically architecture can be a fault tolerant. So, the architecture of HDFS will involve one name node and many data nodes.

(Refer Slide Time: 07:03)



Namenode and Datanodes

(1)
(many)
name node
data node

- **Namenode *(Master)***
  - Metadata:
    - Where file blocks are stored (namespace image)
    - Edit ***(Operation)*** log
  - Secondary namenode ***(Shadow master)***
- **Datanode *(Chunkserver)***
  - Stores and retrieves blocks
    - by client or namenode.
  - Reports to namenode with list of blocks they are storing

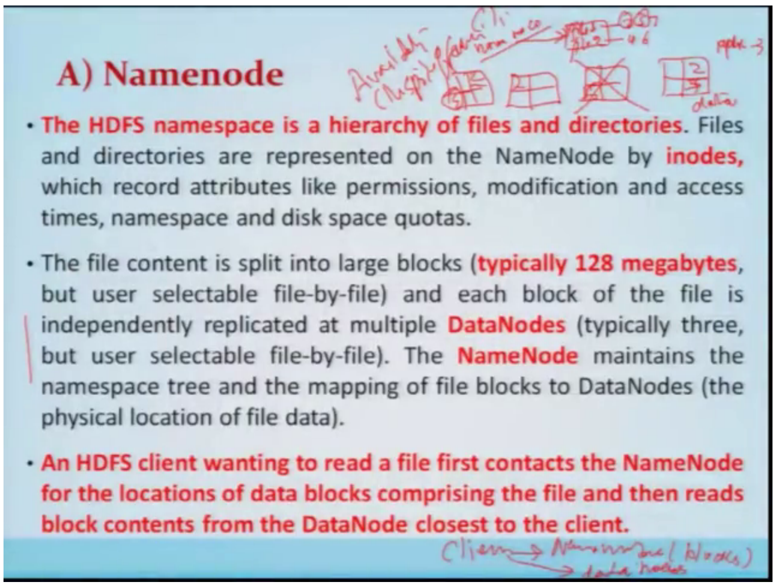
As name node is concern it stores the metadata, metadata is where the file blocks are stored that is the namespace image and basically also the edit and that is operation log also is maintained secondary name node that is it also basically keeps track off or maintaining the secondary name node which is also called as a master or a shadow master.

So; that means, if this particular master fails this particular shadow is basically up to date available and it will switch to the master in that case that is called secondary name node. Now, another thing is called data nodes this is also called chunk server stores and retrieves the file blocks that is the data blocks by the client or a name node, it can be this particular information about where these blocks are stored can be obtained through the name node by the clients these data nodes they report to the name node with a list of blocks that they are restoring.

So, the function of the data nodes which are many in number they are reporting with the help of a heartbeats with the name with the name node. So, it reports to the name node these data nodes they report to the name nodes with a list of blocks that they are restoring. So, whenever a request comes by the client to store the data nodes or through the name node whenever there is a request comes. So, they will be storing and this information will be reported back to the name node so that the metadata can be updated.

So, name node in HDFS name node is an hierarchy of files and directory.

(Refer Slide Time: 09:01)



A) Namenode

- The HDFS namespace is a hierarchy of files and directories. Files and directories are represented on the NameNode by inodes, which record attributes like permissions, modification and access times, namespace and disk space quotas.
- The file content is split into large blocks (typically 128 megabytes, but user selectable file-by-file) and each block of the file is independently replicated at multiple DataNodes (typically three, but user selectable file-by-file). The NameNode maintains the namespace tree and the mapping of file blocks to DataNodes (the physical location of file data).
- An HDFS client wanting to read a file first contacts the NameNode for the locations of data blocks comprising the file and then reads block contents from the DataNode closest to the client.

Client → Namenode (blocks) → data nodes

Files and directories are represented on a name node by the inodes the file content is split into a large blocks that is of 128mb and each block of a file is independently replicated at many data nodes the name node maintains the namespace tree and the mapping of the file blocks to the data nodes.

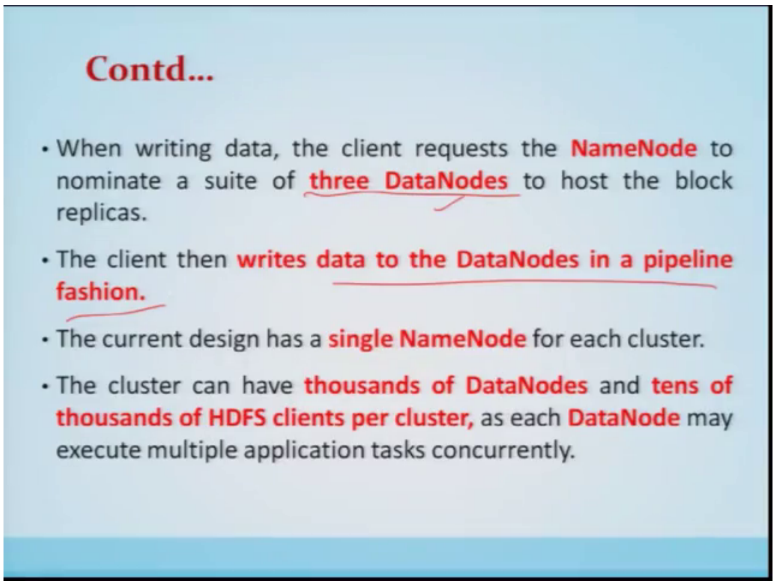
Let us see this particular concept of name node and if let us say a file let us say one is having the data blocks as 257 and another file 2 having the data blocks as 4, 6. So, this is the name node and there will be a data nodes which is storing the blocks of the file now here there is a replication of each block by default it is 3.

So; that means, if block number 2 is stored. So, out of 3 any 2 out of 4, let us say 4 blocks we are having out of 4 data nodes any 3 of them will be selected to store the first one that is block number 2, similarly block number 5 will be stored on any 3 of them. So, if one node crashes or is down or failed. So, a particular data is already available in 2 of the other nodes. So, that is why the availability in spite of failure is assured, that is what we have discussed here.

So, HDFS clients wanting to read the file first contacts the name node for the locations because it maintains the locations of the data blocks comprising the file and then reads the block contains from the data node closest to the client. So, once the client so the client contacts to the name node and we know knows the file blocks and then the addresses of the data nodes. So, then client in turn will contact to the data nodes and can access the file data that is in the form of data blocks.

So, when writing a data the client requests the name node to nominate the suit of 3 data nodes that I mentioned to host the block replicas, client then writes the data directly to the data node in a pipeline fashion.

(Refer Slide Time: 11:55)



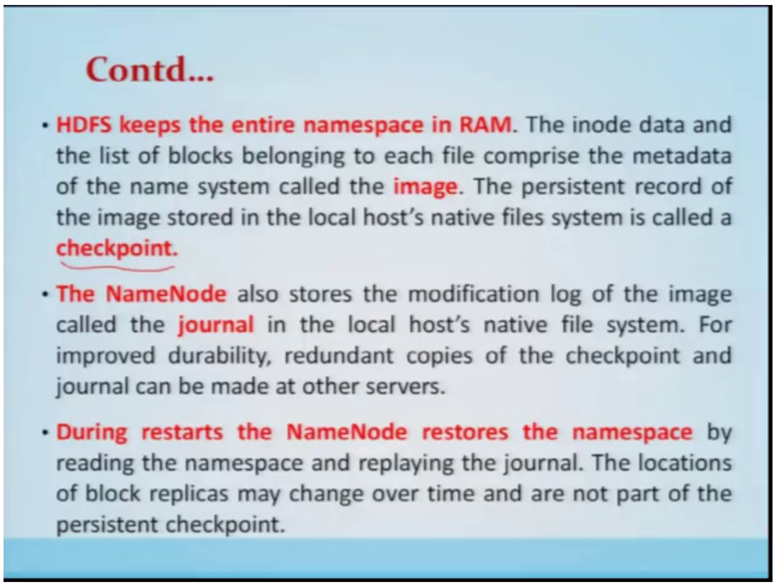
Contd...

- When writing data, the client requests the NameNode to nominate a suite of three DataNodes to host the block replicas.
- The client then writes data to the DataNodes in a pipeline fashion.
- The current design has a single NameNode for each cluster.
- The cluster can have thousands of DataNodes and tens of thousands of HDFS clients per cluster, as each DataNode may execute multiple application tasks concurrently.

That means it will push the data along 3 data nodes which is given by the name node and then single write operation will write them.

So, HDFS keeps the entire name space in ram.

(Refer Slide Time: 12:17)



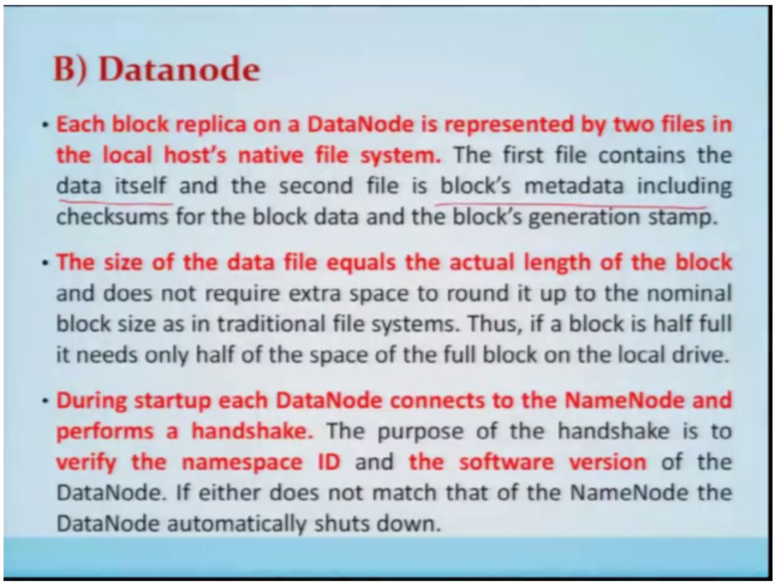
Contd...

- HDFS keeps the entire namespace in RAM. The inode data and the list of blocks belonging to each file comprise the metadata of the name system called the image. The persistent record of the image stored in the local host's native files system is called a checkpoint.
- The NameNode also stores the modification log of the image called the journal in the local host's native file system. For improved durability, redundant copies of the checkpoint and journal can be made at other servers.
- During restarts the NameNode restores the namespace by reading the namespace and replaying the journal. The locations of block replicas may change over time and are not part of the persistent checkpoint.

The inode data and the list of blocks belonging to each file comprise the metadata of the names of the name system called the image; the persistent record of the image is stored in the local hosts native file system called the check point. So, name node also stores the modifications a log image called the journals in the local host native file system. During restarts the name node restores the name space by reading the name space and replaying the journal.

Now, the another important component here is called data nodes, each block replicas on a data node is represented by 2 files in the local host native file system.

(Refer Slide Time: 12:53)

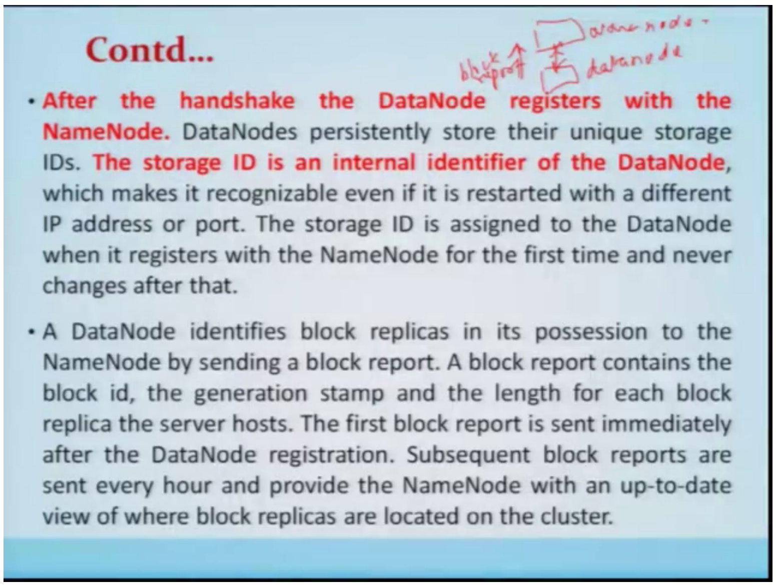


B) Datanode

- **Each block replica on a DataNode is represented by two files in the local host's native file system.** The first file contains the data itself and the second file is block's metadata including checksums for the block data and the block's generation stamp.
- **The size of the data file equals the actual length of the block** and does not require extra space to round it up to the nominal block size as in traditional file systems. Thus, if a block is half full it needs only half of the space of the full block on the local drive.
- **During startup each DataNode connects to the NameNode and performs a handshake.** The purpose of the handshake is to **verify the namespace ID** and **the software version** of the DataNode. If either does not match that of the NameNode the DataNode automatically shuts down.

The first file contains the data itself and the second file is the blocks metadata including the checksum for the block data and the blocks generation stamp. Now during startup each data nodes connects to the name node and performs the handshake.

(Refer Slide Time: 13:28)



Contd...

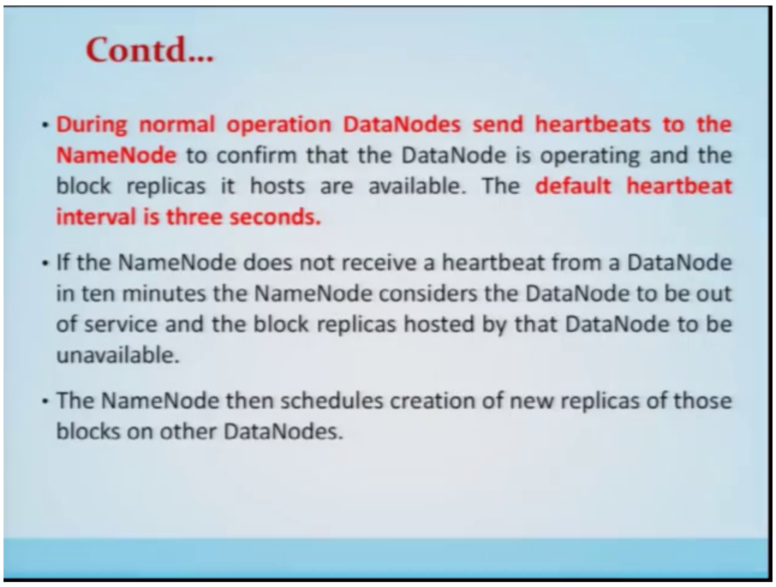
block report
name node
datanode

- **After the handshake the DataNode registers with the NameNode.** DataNodes persistently store their unique storage IDs. **The storage ID is an internal identifier of the DataNode**, which makes it recognizable even if it is restarted with a different IP address or port. The storage ID is assigned to the DataNode when it registers with the NameNode for the first time and never changes after that.
- A DataNode identifies block replicas in its possession to the NameNode by sending a block report. A block report contains the block id, the generation stamp and the length for each block replica the server hosts. The first block report is sent immediately after the DataNode registration. Subsequent block reports are sent every hour and provide the NameNode with an up-to-date view of where block replicas are located on the cluster.

The namespace ID is assigned to the file system instance when it is formatted, consistency of software version is important because incompatible version may cause data corruption or a loss data node that is newly installed and without any namespace ID is permitted to join the cluster and receive the clusters namespace.

So, after the handshake the data node registers with the name node and data nodes persistently stores their unique storage ID's, date node identifies the block replicas in its position to the name node by sending the block report, subsequent block reports are sent every hour and provides the namespace name node with the update up to date view of where the block replicas are stored on the cluster.

(Refer Slide Time: 14:24)



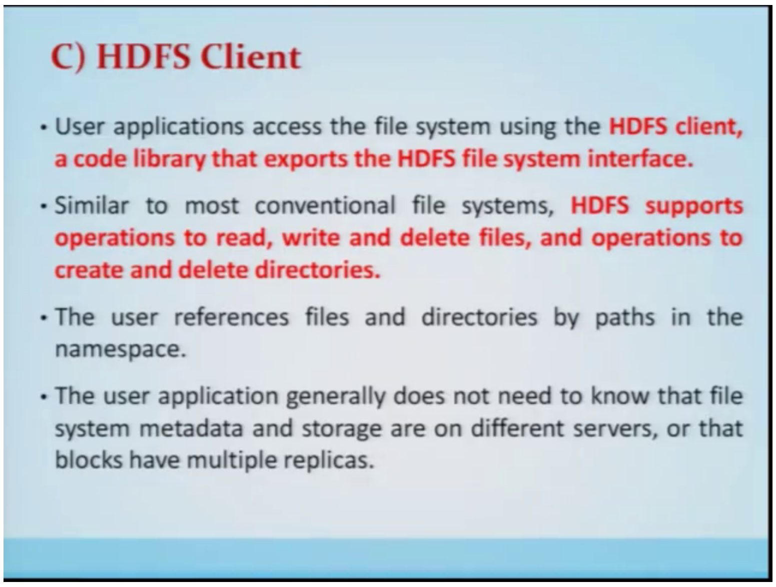
Contd...

- **During normal operation DataNodes send heartbeats to the NameNode** to confirm that the DataNode is operating and the block replicas it hosts are available. The **default heartbeat interval is three seconds.**
- If the NameNode does not receive a heartbeat from a DataNode in ten minutes the NameNode considers the DataNode to be out of service and the block replicas hosted by that DataNode to be unavailable.
- The NameNode then schedules creation of new replicas of those blocks on other DataNodes.

During the normal operation data nodes send heartbeats to the name node that I have told you, the default heartbeat interval is 3 seconds and if the name node does not receive the heartbeat from the data node in 10 minutes, the name node considered considers the data node to be out of service and the block replicas hosted by that by that data node to be unavailable.

So, heartbeats from the data node also carry information about the total storage capacity, fraction of the storage in use and the number of data transfer currently in progress. The name node does not directly call data nodes; it uses the replica heartbeats to send the instruction to the data nodes. So, these commands are important for maintaining the overall integrity and therefore, it is critical to keep heartbeats frequent on the big clusters.

(Refer Slide Time: 15:19)



C) HDFS Client

- User applications access the file system using the **HDFS client, a code library that exports the HDFS file system interface.**
- Similar to most conventional file systems, **HDFS supports operations to read, write and delete files, and operations to create and delete directories.**
- The user references files and directories by paths in the namespace.
- The user application generally does not need to know that file system metadata and storage are on different servers, or that blocks have multiple replicas.

HDFS client, third component is the HDFS client user application access the file system using HDFS client, similar to the most conventional file system HDFS supports operations to read, write, delete files operations to create and delete directories also. When an application reads the file HDFS client first asks the name node for the list of data nodes that I have explained and this interaction can be seen over here.

(Refer Slide Time: 15:45)

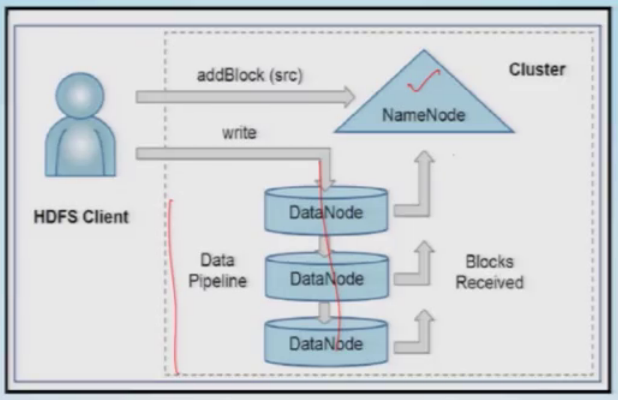
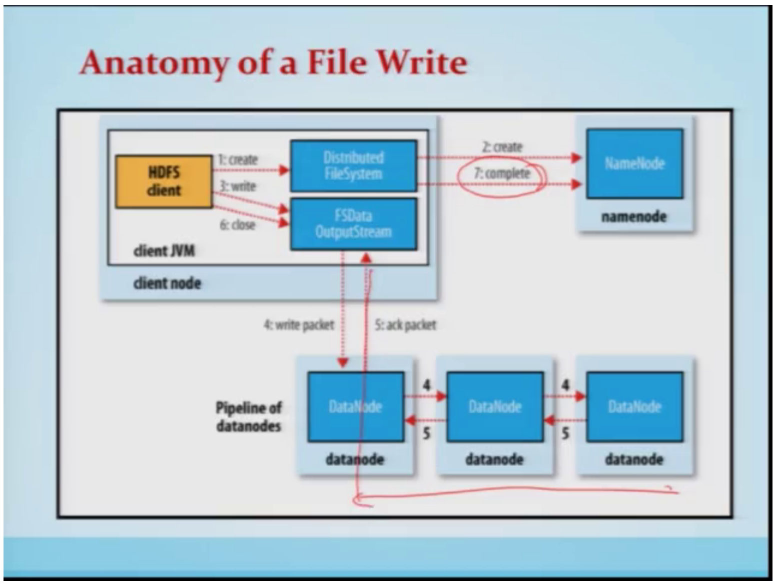


Figure 1. An HDFS client creates a new file by giving its path to the NameNode. For each block of the file, the NameNode returns a list of DataNodes to host its replicas. The client then pipelines data to the chosen DataNodes, which eventually confirm the creation of the block replicas to the NameNode.

So, client has to first contact to the name node because name node has the metadata and through that metadata it can directly access those set of data nodes which contains those block information data blocks. So, it first writes in a pipeline, this is called a pipeline and then issue a write command.

(Refer Slide Time: 16:20)



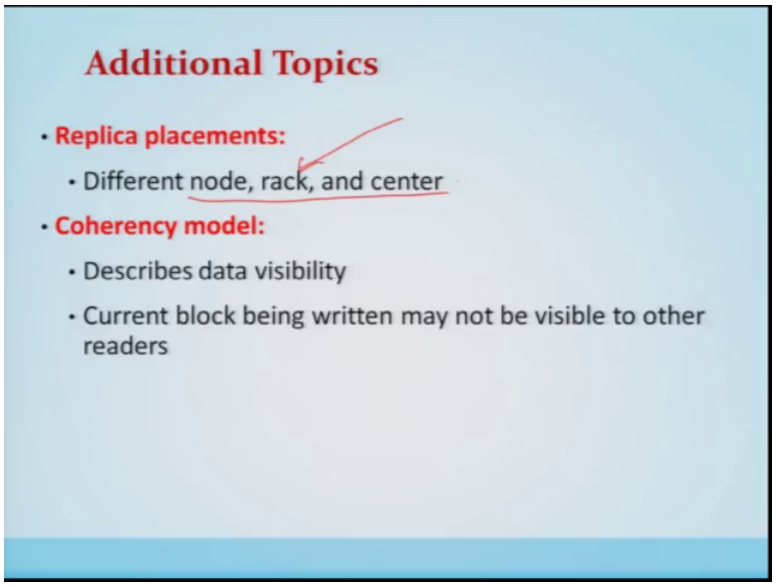
Anatomy of a File Write

So, all these particular data blocks are made or basically is written and then this particular information is basically informed to the name node about the writing of that particular data in a cluster.

Similarly, as far as file read is concerned. So, HDFS first step is to open a file and then contacts the name node to get the block locations and then perform the read operations on these block locations. Now since this particular blocks are stored in a 3 different copies. So, it will try to read any of these 3 copies if it is not successful in this read then it may read the another set of copies or may send simultaneously 2 different read out of 3 and once this particularly read operation is complete then it will close it.

In write it has to inform to the name node when the file is closed, but in read it will just close without informing. So, here this is the anatomy of a write. So, when once this write is complete then it has to inform to the name node that the entire operation is complete. So, here we see that for file writing it will it will contact to the name node, name node will inform about the data nodes and the data nodes then it will basically write packets, write files and this will be done in a pipeline fashion. It is shown as a pipeline of the data nodes and once it is done then basically the acknowledgement will flow back and then HDFS client will also inform the name node about that write operation is complete.

(Refer Slide Time: 18:01)



Additional Topics

- **Replica placements:**
  - Different node, rack, and center
- **Coherency model:**
  - Describes data visibility
  - Current block being written may not be visible to other readers

Now there are other issues because here the HDFS is having the information about the rack. So, most of the decisions of data node placement or a block placement on a data node basically is having the awareness of the position of the nodes in which rack and which center all these information are required. So, that the availability in spite of failure is to be ensured in a much comprehensive manner.

Thank you.