Welcome to the unit of Hadoop Fundamentals on Hadoop architecture.

I will begin with a terminology review and then cover the major components of Hadoop. We will see what types of nodes can exist in a Hadoop cluster and talk about how Hadoop uses replication to lessen data loss. Finally I will explain an important feature of Hadoop called "rack awareness" or "network topology awareness".

Before we examine Hadoop components and architecture, let's review some of the terms that are used in this discussion.

A node is simply a computer. This is typically non-enterprise, commodity hardware for nodes that contain data. So in this example, we have Node 1. Then we can add more nodes, such as Node 2, Node 3, and so on.

This would be called a rack. A rack is a collection of 30 or 40 nodes that are physically stored close together and are all connected to the same network switch. Network bandwidth between any two nodes in rack is greater than bandwidth between two nodes on different racks.

You will see later how Hadoop takes advantage of this fact.

A Hadoop Cluster (or just 'cluster' from now on) is a collection of racks

Let us now examine the pre-Hadoop 2.2 architecture.

Hadoop has two major components:

- the distributed filesystem component, the main example of which is the Hadoop Distributed File System, though other file systems, such as IBM GPFS-FPO, are supported.

- the MapReduce component, which is a framework for performing calculations on

the data in the distributed file system. Pre-Hadoop 2.2 MapReduce is referred to as MapReduce V1 and has its own built-in resource manager and schedule. This unit covers the Hadoop Distributed File System and MapReduce is covered separately.

Let's now examine the Hadoop distributed file system - HDFS

HDFS runs on top of the existing file systems on each node in a Hadoop cluster. It is not POSIX compliant. It is designed to tolerate high component failure rate through replication of the data.

Hadoop works best with very large files. The larger the file, the less time Hadoop spends seeking for the next data location on disk, the more time Hadoop runs at the limit of the bandwidth of your disks. Seeks are generally expensive operations that are useful when you only need to analyze a small subset of your dataset. Since Hadoop is designed to run over your entire dataset, it is best to minimize seeks by using large files. Hadoop is designed for streaming or sequential data access rather than random access. Sequential data access means fewer seeks, since Hadoop only seeks to the beginning of each block and begins reading sequentially from there.

Hadoop uses blocks to store a file or parts of a file.

This is shown in the figure.

Let us now examine file blocks in more detail.

A Hadoop block is a file on the underlying filesystem. Since the underlying filesystem stores files as blocks, one Hadoop block may consist of many blocks in the underlying file system.

Blocks are large. They default to 64 megabytes each and most systems run with block sizes of 128 megabytes or larger.

Blocks have several advantages:

Firstly, they are fixed in size. This makes it easy to calculate how many can fit on a disk.

Secondly, by being made up of blocks that can be spread over multiple nodes, a file can be larger than any single disk in the cluster.

HDFS blocks also don't waste space. If a file is not an even multiple of the block size, the block containing the remainder does not occupy the space of an entire block.

As shown in the figure, a 450 megabyte file with a 128 megabyte block size consumes four blocks, but the fourth block does not consume a full 128 megabytes.

Finally, blocks fit well with replication, which allows HDFS to be fault tolerant and available on commodity hardware.

As shown in the figure:

Each block is replicated to multiple nodes. For example, block 1 is stored on node 1 and node 2. Block 2 is stored on node 1 and node 3. And block 3 is stored on node 2 and node 3. This allows for node failure without data loss. If node 1 crashes, node 2 still runs and has block 1's data. In this example, we are only replicating data across two nodes, but you can set replication to be across many more nodes by changing Hadoop's configuration or even setting the replication factor for each individual file.

The second major component of Hadoop, described in detail in another lecture, is

the MapReduce component.

HDFS was based on a paper Google published about their Google File System, Hadoop's MapReduce is inspired by a paper Google published on the MapReduce technology. It is designed to process huge datasets for certain kinds of distributable problems using a large number of nodes.

A MapReduce program consists of two types of transformations that can be applied to data any number of times - a map transformation and a reduce transformation.

A MapReduce job is an executing MapReduce program that is divided into map tasks that run in parallel with each other and reduce tasks that run in parallel with each other.

Let us examine the main types of nodes in pre-Hadoop 2.2. They are classified as HDFS or MapReduce V1 nodes. For HDFS nodes we have the NameNode, and the DataNodes. For MapReduce V1 nodes we have the JobTracker and the TaskTracker nodes. Each of these is discussed in more detail later in this presentation. There are other HDFS nodes such as the Secondary NameNode, Checkpoint node, and Backup node that are not discussed in this course.

This diagram shows some of the communication paths between the different types of nodes on the system. A client is shown as communicating with a JobTracker. It can also communicate with the NameNode and with any DataNode.

There is only one NameNode in the cluster. While the data that makes up a file is stored in blocks at the data nodes, the metadata for a file is stored at the NameNode. The NameNode is also responsible for the filesystem namespace. To compensate for

the fact that there is only one NameNode, one should configure the NameNode to write a copy of its state information to multiple locations, such as a local disk and an NFS mount. If there is one node in the cluster to spend money on the best enterprise hardware for maximum reliability, it is the NameNode. The NameNode should also have as much RAM as possible because it keeps the entire filesystem metadata in memory.

An typical HDFS cluster has many DataNodes. DataNodes store the blocks of data and blocks from different files can be stored on the same DataNode.

When a client requests a file, the client finds out from the NameNode which DataNodes stored the blocks that make up that file and the client directly reads the blocks from the individual DataNodes. Each DataNode also reports to the NameNode periodically with the list of blocks it stores. DataNodes do not require expensive enterprise hardware or replication at the hardware layer. The DataNodes are designed to run on commodity hardware and replication is provided at the software layer.

A JobTracker node manages MapReduce V1 jobs. There is only one of these on the cluster. It receives jobs submitted by clients. It schedules the Map tasks and Reduce tasks on the appropriate TaskTrackers, that is where the data resides, in a rack-aware manner and it monitors for any failing tasks that need to be rescheduled on a different TaskTracker.

To achieve the parallelism for your map and reduce tasks, there are many TaskTrackers in a Hadoop cluster. Each TaskTracker spawns Java Virtual Machines

to run your map or reduce task. It communicates with the JobTracker and reads blocks from DataNodes.

This lesson continues in the next video.