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 examined 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 A rack is a collection of 30 or 40 nodes that are authent stored close together and are all connected to the same network switch. Network bandwidth between any two nodes in the same rack is greater than bandwidth between two node.

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 Spectrum Scale, are supported.

- the MapReduce component, which is a framework for hard 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 separate. 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 They are only need to analyze a small subset of your dataset. Since Hadoop is designed to run over your your entire dataset, it is best to minimize from by using large files. Hadoop is designed for streaming or sequential data access. Access means minim seeks, since Hadoop only seeks to the beginning of each block and begins reading from the case. Hadoop uses blocks to store a file or parts of a file. This is shown in the figure. Let us now copy file blocks in more Detail. A Hadoop block is a file on the underlying filesystem. Since the underlying filesystem files as blocks, one Hadoop block may consist of many blocks in the underlying file system. Blocks are large. They default t O 64 megabytes each and most systems run with block sizes of 128 megabytes or larger. Blocks have are several times: First, they are fixed in size. This makes it easy to calculate how many can fit on a disk. Of blocks that can be spread over multiple nodes, a file can be larger than any single disk in the cluster. HDFS blocks also do not waste space. If a file is not an even multiple of the block size, the block containing the remainder Does Not as 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&#39;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&#39;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&#39;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. A 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.

欢迎来到Hadoop架构的Hadoop基础知识单元。我将从术语审查开始，然后涵盖Hadoop的主要组件。我们将看到Hadoop集群中可以存在什么类型的节点，并谈论Hadoop如何使用复制来减少数据丢失。最后我将解释Hadoop的一个重要特性，称为“机架意识”;或“网络拓扑意识”;

在我们检查Hadoop组件和体系结构之前，我们来回顾一下在本讨论中使用的一些术语。节点只是一台电脑。这通常是包含数据的节点的非企业级商品硬件。所以在这个例子中，我们有了Node 1.然后我们可以添加更多的节点，比如Node 2，Node 3等等。这将是一个机架是30或40个节点的集合，它们被认证存储在一起，并且都连接到相同的网络交换机。同一机架中任何两个节点之间的网络带宽大于两个节点之间的带宽。

一个Hadoop集群（或从现在开始就是集群）是一个机架的集合让我们来看一下Hadoop 2.2之前的架构。 Hadoop有两个主要组成部分：

- 分布式文件系统组件，其主要示例是Hadoop分布式文件系统，尽管其他文件系统（如IBM Spectrum Scale）也受支持。

- MapReduce组件，它是数据硬计算的框架

在分布式文件系统中。

Pre-Hadoop 2.2 MapReduce被称为MapReduce V1，并有自己的内置资源管理器和进度。本单元涵盖Hadoop分布式文件系统，MapReduce分开覆盖。 HDFS在Hadoop集群中每个节点上的现有文件系统之上运行。它不符合POSIX标准。它旨在通过复制数据来容忍高组件故障率。 Hadoop最适合使用非常大的文件。文件越大，Hadoop花费在磁盘上下一个数据位置的时间越少，Hadoop在磁盘带宽限制下运行的时间越长。当他们只需要分析数据集的一小部分时，寻求通常是便宜的操作。由于Hadoop旨在运行您的整个数据集，因此最好通过使用大文件进行最小化。 Hadoop专为流式或顺序数据访问而设计。访问意味着最小化寻求，因为Hadoop只寻求每个块的开始，并开始从案例中读取。 Hadoop使用块来存储文件或文件的一部分。如图所示。现在让我们更详细地复制文件块。 Hadoop块是底层文件系统上的一个文件。由于底层文件系统文件作为块，一个Hadoop块可能由底层文件系统中的许多块组成。块很大。它们每个默认为64兆字节，大多数系统以128兆字节或更大的块大小运行。块有几次：首先，它们的大小是固定的。这样可以很容易地计算出可以在磁盘上安装多少。可以分散在多个节点上的块，文件可以大于集群中的任何单个磁盘。 HDFS块也不浪费空间。如果文件不是块大小的偶数倍，则包含余数的块不作为整个块的空间。如图所示，具有A 128兆字节块大小的450兆字节文件消耗四个块，但第四个块不消耗完整的128兆字节。最后，块与复制配合良好，这允许HDFS在商品硬件上具有容错能力和可用性。如图所示：每个块被复制到多个节点。例如，块1存储在节点1和节点2上。块2存储在节点1和节点3上。块3存储在节点2和节点3上。这允许节点发生故障而没有数据丢失。如果节点1崩溃，节点2仍然运行并具有块1的数据。在这个例子中，我们只是跨两个节点复制数据，但是您可以通过更改Hadoop的配置，甚至为每个单独的文件设置复制因子来将复制设置为跨越更多节点。

在另一场讲座中详细描述的Hadoop的第二个主要组件是MapReduce组件。 HDFS是基于Google发布的关于他们的Google文件系统的文章，Hadoop的MapReduce的灵感来自于Google在MapReduce技术上发表的论文。它被设计为使用大量节点处理某些种类的可分配问题的巨大数据集。 MapReduce程序由两种类型的转换组成，可以将数据应用于数据，即地图变换和减少转换。 MapReduce作业是一个执行的MapReduce程序，它被划分为彼此并行运行的映射任务，并减少彼此并行运行的任务。让我们来看一下Hadoop 2.2之前的节点的主要类型。它们被归类为HDFS或MapReduce V1节点。对于HDFS节点，我们有NameNode和DataNodes。对于MapReduce V1节点，我们有JobTracker和TaskTracker节点。在本演示文稿的后面将详细讨论这些。还有其他HDFS节点，如本节中没有讨论的Secondary NameNode，Checkpoint节点和Backup节点。该图显示了系统上不同类型节点之间的一些通信路径。客户端显示为与JobTracker通信。它还可以与NameNode和任何DataNode进行通信。

集群中只有一个NameNode。虽然构成文件的数据存储在数据节点的块中，但文件的元数据存储在NameNode中。 NameNode还负责文件系统命名空间。为了补偿只有一个NameNode的事实，应该配置NameNode以将其状态信息的副本写入多个位置，例如本地磁盘和NFS安装。如果集群中有一个节点花在最佳企业硬件上以获得最大的可靠性，那么它就是NameNode。 NameNode也应该具有尽可能多的RAM，因为它将整个文件系统元数据保存在内存中。典型的HDFS集群有许多DataNodes。 DataNode存储数据块和不同文件的块可以存储在同一个DataNode上。当客户端请求文件时，客户端从NameNode中发现哪个DataNodes存储构成该文件的块，客户端直接从单个DataNodes读取块。每个DataNode还会定期向NameNode报告其存储的块列表。 DataNodes在硬件层不需要昂贵的企业硬件或复制。 DataNodes设计用于在商品硬件上运行，并在软件层提供复制。

JobTracker节点管理MapReduce V1作业。集群中只有其中之一。它接收客户提交的作业。它以对机架感知的方式调度Map任务并在适当的TaskTrackers（即数据所在的位置）上减少任务，并监视需要重新安排在不同TaskTracker上的任何失败的任务。为了实现地图的并行性并减少任务，Hadoop集群中有许多TaskTrackers。每个TaskTracker都会产生Java虚拟机来运行地图或减少任务。它与JobTracker进行通信，并从DataNodes读取块。

本课继续下一个视频。

Hadoop 2.2 brought about architectural changes to MapReduce. As Hadoop has matured, people have found that it can be used for more than running MapReduce jobs. But to keep each new framework from having its own resource manager and scheduler, that would compete with the other framework resource managers and schedulers, it was decided to have the recourse manager and schedulers to be external to any framework. This new architecture is called YARN. (Yet Another Resource Negotiator) . You still have DataNodes but there are no longer TaskTrackers and the JobTracker. You are not required to run YARN with Hadoop 2.2. as MapReduce V1 is still supported. There are two main ideas with YARN. Provide generic scheduling and resource management. This way Hadoop can support more than just MapReduce. The other is to try to provide a more efficient scheduling and workload management. With MapReduce V1, the administrator had to define how many map slots and how many reduce slots there were on each node. Since the hardware capabilities for each node in a Hadoop cluster can vary, for performance reasons, you might want to limit the number of tasks on certain nodes. With YARN, this is no longer required. With YARN, the resource manager is aware of the capabilities of each node via communication with the NodeManager running on each node. When an application gets invoked , an Application Master gets started. The Application Master is then responsible for negotiating resources from the ResourceManager. These resources are assigned to Containers on each slave-node and you can think that tasks then run in Containers. With this architecture, you are no longer forced into a one size fits all. The NameNode is a single point of failure. Is there anything that can be done about that? Hadoop now supports high availability. In this setup, there are now two NameNodes, one active and one standby. Also, now there are JournalNodes. There must be at least three and there must be an odd number. Only one of the NameNodes can be active at a time. It is the JournalNodes, working together , that decide which of the NameNodes is to be the active one and if the active NameNode has been lost and whether the backup NameNode should take over. The NameNode loads the metadata for the file system into memory. This is the reason that we said that NameNodes needed large amounts of RAM. But you are going to be limited at some point when you use this vertical growth model. Hadoop Federation allows you to grow your system horizontally. This setup also utilizes multiple NameNodes. But they act independently. However, they do all share all of the DataNodes. Each NameNode has its own namespace and therefore has control over its own set of files. For example, one file that has blocks on DataNode 1 and DataNode 2 might be owned by NameNode 1. NameNode 2 might own a file that has blocks on DataNode 2 and DataNode 3. And NameNode 3 might have a file with blocks on all three DataNodes.

Hadoop has awareness of the topology of the network. This allows it to optimize where it sends the computations to be applied to the data. Placing the work as close as possible to the data it operates on maximizes the bandwidth available for reading the data. In the diagram, the data we wish to apply processing to is block B1, the dark blue rectangle on node n1 on rack 1. When deciding which TaskTracker should receive a MapTask that reads data from B1, the best option is to choose the TaskTracker that runs on the same node as the data. If we can&#39;t place the computation on the same node, our next best option is to place it on a node in the same rack as the data. The worst case that Hadoop currently supports is when the computation must be processed from a node in a different rack than the data. When rack-awareness is configured for your cluster, Hadoop will always try to run the task on the TaskTracker node with the highest bandwidth access to the data.

Hadoop 2.2带来了MapReduce的架构更改。随着Hadoop已经成熟，人们发现它可以用于运行MapReduce工作。但是要保持每个新框架不具备自己的资源管理器和调度程序，这将与其他框架资源管理器和调度程序竞争，决定让追索管理器和调度程序在任何框架之外。这种新架构叫做YARN。 （又一资源谈判者）。您仍然拥有DataNodes，但不再有TaskTrackers和JobTracker。您不需要使用Hadoop 2.2运行YARN。因为仍然支持MapReduce V1。 YARN有两个主要思路。提供通用调度和资源管理。这样Hadoop可以支持多于MapReduce。另一个是尝试提供更有效的调度和工作负载管理。使用MapReduce V1，管理员必须定义每个节点上有多少个映射槽位和多少个存储槽。由于Hadoop集群中每个节点的硬件功能可能会有所不同，出于性能原因，您可能希望限制某些节点上的任务数量。使用YARN，这不再需要。使用YARN，资源管理器通过与在每个节点上运行的NodeManager进行通信来了解每个节点的功能。当应用程序被调用时，应用程序主机将启动。然后应用程序主人负责从ResourceManager协商资源。这些资源分配给每个从节点上的容器，您可以认为任务然后在容器中运行。有了这个架构，你就不再需要一个适合所有人的尺寸了。 NameNode是单点故障。有什么可以做的吗？ Hadoop现在支持高可用性。在此设置中，现在有两个NameNodes，一个活动和一个备用。此外，现在有JournalNodes。必须至少有三个，必须有一个奇数。 NameNodes中只有一个可以一次处于活动状态。 JournalNodes一起工作，决定哪个NameNodes是活动的，并且活动的NameNode已经丢失以及备份NameNode是否应该接管。 NameNode将文件系统的元数据加载到内存中。这就是我们说NameNodes需要大量RAM的原因。但是当您使用这种垂直增长模式时，您将在某些时候受到限制。 Hadoop联盟允许您水平扩展您的系统。此设置也使用多个NameNodes。但他们独立行事。但是，它们都共享所有的DataNodes。每个NameNode都有自己的命名空间，因此可以控制自己的一组文件。例如，在DataNode 1和DataNode 2上具有块的一个文件可能由NameNode 1拥有。NameNode 2可能拥有DataNode 2和DataNode 3上具有块的文件。NameNode 3可能在所有三个DataNodes上都有一个带有块的文件。

Hadoop已经意识到了网络的拓扑。这允许它优化发送要应用于数据的计算的位置。将工作尽可能靠近其操作的数据最大化可用于读取数据的带宽。在图中，我们希望应用处理的数据是块B1，机架1上节点n1上的深蓝色矩​​形。当决定哪个TaskTracker应该接收到从B1读取数据的MapTask时，最好的选择是选择TaskTracker在与数据相同的节点上运行。如果我们不能将计算放在同一个节点上，那么我们的下一个最佳选择就是把它放在与数据相同的机架中的一个节点上。 Hadoop目前支持的最糟糕的情况是必须从与数据不同的机架中的节点处理计算。当您的集群配置了机架感知功能时，Hadoop将始终尝试在TaskTracker节点上运行任务，并对数据进行最高带宽访问。