1. 什么是元数据？

任何文件系统中的数据中的数据分为数据和元数据。数据是指普通文件中的实际数据，而元数据指用来描述一个文件的特征的系统数据，诸如访问权限、文件拥有者以及文件数据块的分布信息等等，在集群文件系统中，分布信息包括文件在磁盘上的位置以及磁盘在集群中的位置。用户需要操作一个文件必须首先得到它的元数据，才能定位到文件的位置并且得到文件的内容或相关属性。

1. 元数据管理方式

元数据管理有两种方式。集中式管理和分布式管理。集中式管理是指在系统中有一个节点专门司职元数据管理，所有元数据都存储在该节点的存储设备上。所有客户端对文件的请求前，都要先对该元数据管理器请求元数据。分布式管理是指将元数据存放在系统的任意节点并且能动态的迁移。对元数据管理的职责也分布到各个不同的节点上。大多数集群文件系统都采用集中式的元数据管理。因为集中式管理实现简单，一致性维护容易，在一定的操作频繁度内可以提供较满意的性能。缺点是单一失效点问题，若该服务器失效，整个系统将无法正常工作。而且，当对元数据的操作过于频繁时，集中的元数据管理成为整个系统的性能瓶颈。

分布式元数据管理的好处是解决了集中式管理的单一失效点问题， 而且性能不会随着操作频繁而出现瓶颈。其缺点是，实现复杂，一致性维护复杂，对性能有一定影响。

1. Hadoop系统架构

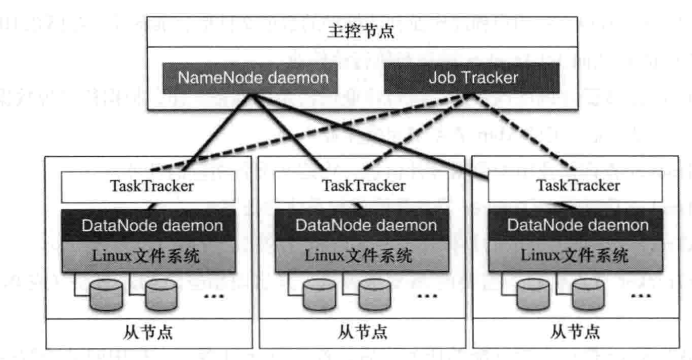
Hadoop系统的基本组成构架包括分布式存储和并行计算两部分。

分布式存储架构上，Hadoop系统使用NameNode作为分布式存储的主控节点、用以存储和管理分布式文件系统的元数据。同时使用DataNode作为实际存储大规模数据的从节点，每个从节点基于底层的Linux系统在本节点上存储实际数据。

并行计算架构上，Hadoop使用**JobTracker**作为MapReduce并行计算框架的主控节点，用以**管理和调度作业的执行**，用**TaskTracker管理每个计算从节点上计算任务的执行**。

为了实现Hadoop系统设计中本地化计算的原则，数据存储节点DataNode与计算节点TaskTracker将合并设置，让每个从节点同时运行作为DataNode和TaskTracker，以此让每个TaskTracker尽量处理存储在本地DataNode上的数据。

而数据存储主控节点NameNode与作业执行主控节点JobTracker既可以设置在同一个主控节点上，在集群规模较大或者这两个主控节点负载都很高以至于会相互影响时，也可以分开设置在两个不同的节点上。



Hadoop系统的基本组成架构

1. Hadoop MapReduce并行计算框架构建于HDFS之上，其中包含一个主控节点JobTracker以及众多从节点TaskTracker。JobTracker作为Hadoop的主控节点，主要负责调度、管理作业中的任务。TaskTracker作为从节点（任务节点），负责执行JobTracker分发过来的任务。

如上所述，当一个作业被提交给Hadoop系统时，这个作业的输入数据会被划分成很多等长的数据块，每个数据块都会对应于一个Map任务。这些Map任务会同时执行、并行化地处理数据。Map任务的输出数据会被排序，然后被系统分发给Reduce任务以做进一步的处理。在作业执行的整个过程中，JobTracker会对所有任务进行以下管理：重复执行失败的任务，更改作业的执行状态，等等。

1. 作业执行流程

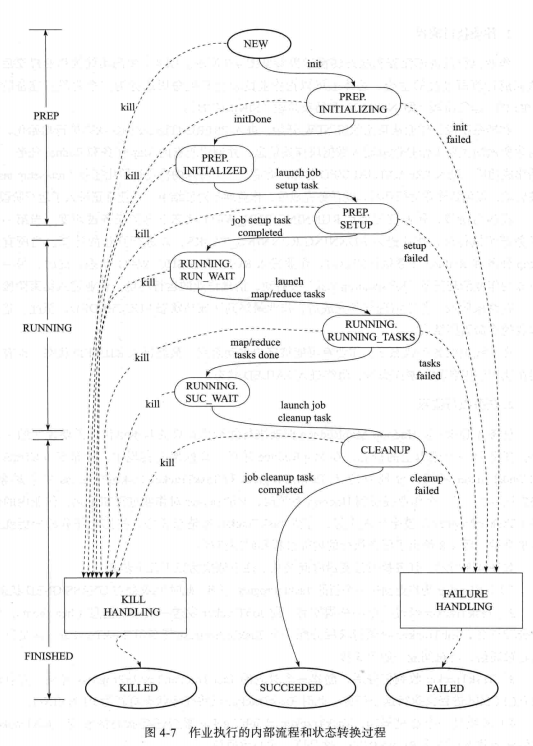
作业执行的内部流程和状态转换过程参见下图，图中实线为作业提交后成功完成计算所经过的主线。总体上可以把作业的运行和生命周期分为三个阶段：准备阶段（PREP）、运行阶段（RUNNING）和结束阶段（FINISHED）。

在准备阶段，作业从初始化状态NEW开始，进入PREP.INITIALIZING状态进行初始化，**初始化所做的主要工作是读取输入数据块描述信息，并创建所有的Map任务和Reduce任务。**初始化成功后，进入PREP.INITIALIZED状态。此后，一个特殊的作业初始化任务（job setup task）被启动，以**创建作业运行环境**，此任务完成后，作业准备阶段结束，作业真正进入运行阶段。

在运行阶段，作业首先处在RUNNING.RUN\_WAIT状态下等待任务被调度。当第一个任务开始执行时，作业进入RUNNING.RUNNING\_TASKS,以进行真正的计算。当所有的Map任务和Reduce任务执行完成后，作业进入RUNNING.SUC\_WAIT状态。此时，另一个特殊的作业清理任务（job cleanup task）被启动，清理作业运行环境，作业进入结束阶段。

在结束阶段，作业清理任务完成后，作业最终到达成功状态SUCCEEDED,至此，整个作业的生命周期结束。

在主线上的各个状态下，作业有可能被客户主动杀死，最终进入KILLED状态；也有可能在执行中因各种因素而失败，最终进入FAILED状态。



1. 任务执行流程

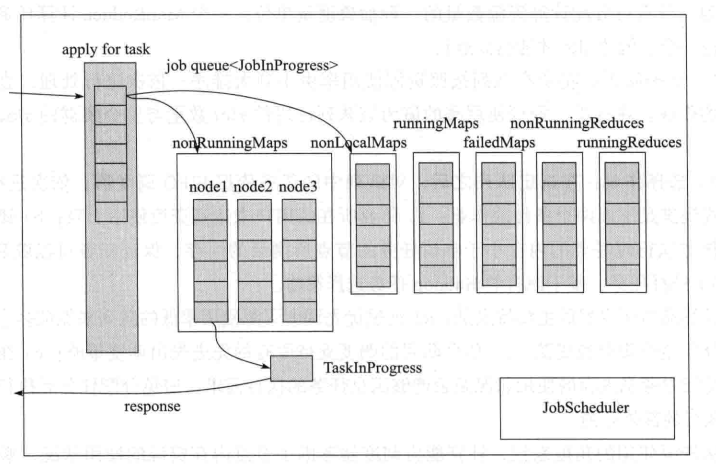
任务（Task）是Hadoop MapReduce框架进行并行化计算的基本单位。需要说明的是：任务是一个逻辑上的概念，在MapReduce并行计算框架的实现中，分布于jobTracker和TaskTracker两端，分别对应于TaskInProgress和TaskTracker.TaskInProgress两个对象。

1. 作业调度基本过程

简单地说，Hadoop MapReduce作业调度就是根据一定的策略，从作业队列中选择一个合适的作业，为了分配资源让他们得以执行。

与传统的作业调度不同，在Hadoop MapReduce并行计算框架中，每个作业都被划分为很多更小粒度的任务单元。因此，Hadoop作业调度在选择合适的作业之后还需从中选择合适的任务。不同的调度器对作业有着不同的组织结构，如单队列、多队列、作业池等，但是任务的组织结构都沿用了Hadoop本身的机制。

Hadoop MapReduce构建于Hadoop分布式文件系统HDFS之上。在默认情况下，每个文件都会被划分为64MB大小的一系列数据块（Block），每个数据块都会有三个备份存在集群中三个不同的节点上。



1. MapReduce执行框架所涉及的组件和执行流程

下图展示了Hadoop MapReduce执行框架所涉及的组件和执行流程。每个TaskTracker节点将从HDFS分布式文件中读取所要处理的数据。Hadoop MapReduce框架提供了一个InputFormat对象负责具体以什么样的输入格式读取数据。然后数据会被分为很多个分片（Split），每个分片将交由一个Map对象去处理。在进入Map之前，需要通过RecordReader对象逐个从数据分片中读出数据记录，并转换为Key-Value键值对，逐个输入到Map中处理。Map输出中间结果前，需要经过一个Combiner对象将该Map输出的相同主键下的所有键值对合并为一个键值对；Map所输出的中间结果在进入Reduce节点之前，先通过中间的Partition对象对数据分区，将数据发送到合适的Reduce节点上，避免不同Reduce节点上的数据的相关性，保证每个Reduce节点可独立完成本地计算；在传入Reduce节点之前还会自动将所有键值进行排序。Reduce节点完成计算后，经过OutputFromat对象指定输出数据的具体格式，最终将数据输出并写回到HDFS中。

