# HDFS

目录

[第四章 HDFS 1](#_Toc24412)

[4.1 HDFS与Spark关系 1](#_Toc22510)

[4.2 HDFS概述 3](#_Toc14210)

[4.2.1 特点 3](#_Toc24217)

[4.2.2 主要组件与架构 3](#_Toc10802)

[4.3 HDFS架构分析 4](#_Toc21944)

[4.3.1 数据块 4](#_Toc30051)

[4.3.2 NameNode 5](#_Toc2078)

[4.3.3 DataNode 5](#_Toc12643)

[4.3.4 SecondaryNameNode 5](#_Toc11554)

[4.3.5 数据备份 6](#_Toc16239)

[4.4 文件操作过程 7](#_Toc29534)

[4.4.1 读文件 7](#_Toc25786)

[4.4.2 写文件 8](#_Toc16410)

[4.4.3 删除文件 9](#_Toc23243)

[4.5 HDFS安装与配置 10](#_Toc4789)

[4.5.1 安装Hadoop 10](#_Toc26543)

[4.5.2 修改Hadoop文件 11](#_Toc29834)

[4.5.3 启动hadoop 14](#_Toc25807)

[4.6 HDFS基本命令 16](#_Toc24817)

[4.6.1 创建目录 16](#_Toc4460)

[4.6.2 查看文件列表 17](#_Toc11844)

[4.6.3 上传文件到HDFS 17](#_Toc13843)

[4.6.4 下载文件到本地 18](#_Toc2065)

[4.6.5 查看HDFS文件内容 18](#_Toc2177)

[4.6.6 删除HDFS文件 18](#_Toc19429)

[4.7 HDFS与spark-shell 18](#_Toc31693)

HDFS（Hadoop Distributed File System），设计目标为将超大数据集存储到网络中的多台普通商用计算机上，并提供高可靠性和高吞吐量的服务。本章首先介绍HDFS的架构、运行原理、读写文件流程，然后介绍安装配置的具体流程，最后通过简单实例介绍基本的应用方法。

## HDFS与Spark关系

在Spark出现之前，Hadoop已经提供了在集群机器上实现容错、并行处理的框架。Hadoop解决了两个关键问题：数据的可靠存储和数据的分析处理。对应的，Hadoop的核心为HDFS和MapReduce。

HDFS：分布式文件系统。在集群中提供高可靠文件存储，通过将文件块保存多个副本解决故障问题。

MapReduce：分布式计算框架。

Hadoop对于大多数大规模分布式数据处理的商业场景都适用，但在迭代和交互的计算与工作、实时数据处理等场景无法胜任。Apache Spark的创造者意识到开发一个有效且高效地进行迭代和交互数据处理系统的重要性。因此，Spark不仅保留了Hadoop的批处理、可扩展能力、错误容忍、分布式数据处理的优点，而且创造一个新的体系结构避免MapReduce阶段昂贵的硬盘读开销，并且需要在集群节点上支持完全在内存中处理分布式的数据。

Spark提供了一个定义明确且层次化的体系结构。所有层次和组件之间都是松耦合的，且与外部应用和扩展的集成都是基于定义明确的契约。Spark的体系结构同时支持基于核心API的自定义库和扩展。如下图所示为Spark的体系结构。主要包括四个层次。

数据存储层：负责提供数据永久存储区，包括本地文件系统、HDFS或者任何其他的NoSQL数据库（如HBase、Cassandra、MongoDB、S3等）

资源管理API：负责在集群中为客户作业分配可用资源。

Spark Core：包括支持Spark通用执行引擎的API，支持在内存中分布式计算，同时支持一个包含多种应用和语言的通用执行模型。

Spark扩展库：基于Spark Core开发。支持不同场景下的附近功能。

Spark有效集成了Hadoop，是MapReduce的一种更优的替代方案，可以指定Hadoop，YARN的版本来编译出合适的发型版本，Spark也能够很容易地运行在EC2、Mesos上，或以Standalone模式运行，并从HDFS、HBase、Cassandra和其他Hadoop数据源读取数据。



图 4. 1 Spark架构

## HDFS概述

### 特点

HDFS的设计特点与目标如下：

**①应对硬件故障**

硬件故障是常态，而不是异常。整个HDFS系统将由数百或数千个存储着文件数据片段的服务器组成。实际上它里面有非常巨大的组成部分，每一个组成部分都很可能出现故障，这就意味着HDFS里的总是有一些部件是失效的，因此，故障的检测和自动快速恢复是HDFS一个很核心的设计目标。

**②流式数据访问**

运行在HDFS之上的应用程序必须流式地访问它们的数据集，它不是运行在普通文件系统之上的普通程序。HDFS被设计成适合批量处理的，而不是用户交互式的。重点是在数据吞吐量，而不是数据访问的反应时间，POSIX的很多硬性需求对于HDFS应用都是非必须的，去掉POSIX一小部分关键语义可以获得更好的数据吞吐率。

**③存储大数据集**

运行在HDFS之上的程序有很大量的数据集。典型的HDFS文件大小是GB到TB的级别。所以，HDFS被调整成支持大文件。它应该提供很高的聚合数据带宽，一个集群中支持数百个节点，一个集群中还应该支持千万级别的文件。

**④支持简单一致性模型**

大部分的HDFS程序对文件操作需要的是**一次写多次读取**的操作模式。一个文件一旦创建、写入、关闭之后就不需要修改了。这个假定简单化了数据一致的问题，并使高吞吐量的数据访问变得可能。

**⑤移动计算比移动数据更经济**

在靠近计算数据所存储的位置来进行计算是最理想的状态，尤其是在数据集特别巨大的时候。这样消除了网络的拥堵，提高了系统的整体吞吐量。一个假定就是迁移计算到离数据更近的位置比将数据移动到程序运行更近的位置要更好。HDFS提供了接口，来让程序将自己移动到离数据存储更近的位置。

**⑥异构软硬件平台间的可移植性**

HDFS被设计成可以简便地实现平台间的迁移，这将推动需要大数据集的应用更广泛地采用HDFS作为平台。兼容各种硬件和软件平台。

**HDFS不适合的场景主要有：**要求低时间延迟数据访问的应用，存储大量的小文件，多用户写入，任意修改文件。

### 主要组件与架构

HDFS主要由NameNode、DataNode和SecondaryNameNode三个组件构成。HDFS是一个Master/Slave(主从)结构，一个HDFS集群有一个NameNode，它是一个管理文件命名空间和调节客户端访问文件的主服务器，当然还有一些DataNode，通常是一个节点一个机器，它来管理对应节点的存储。NameNode、SecondaryNameNode运行在Master节点，DataNode运行在Slave节点上。HDFS对外开放文件命名空间并允许用户数据以文件形式存储。

内部机制是将一个文件分割成一个或多个块，这些块被存储在一组数据节点中。NameNode用来操作文件命名空间的文件或目录操作，如打开，关闭，重命名等等。它同时确定块与数据节点的映射。DataNode负责来自文件系统客户的读写请求。数据节点同时还要执行块的创建，删除，和来自NameNode的块复制指令。

集群中只有一个NameNode极大地简单化了系统的体系结构。NameNode是仲裁者和所有HDFS元数据的仓库，用户的实际数据不经过NameNode。

****

图 4. 2 HDFS架构

## HDFS架构分析

本小节将介绍HDFS架构中的数据块存储、三大组件（NameNode、DataNode、SecondaryNameNode）的具体作用以及数据备份。

### 数据块

HDFS上的文件被划分为块大小的多个分块，作为独立的存储单元，称为数据块，默认大小是64MB。数据块存储在DataNode中，为了能够容错，数据块是以多个副本的形式分布在集群中。副本数量默认为3。

HDFS使用块存储的好处主要有：一个文件的大小可以大于网络中任意一个磁盘的容量。文件的所有块不需要存储在同一个磁盘上，因此它们可以利用集群上的任意一个磁盘进行存储；简化了存储子系统的设计，将存储子系统控制单元设置为块，可简化存储管理，同时元数据就不需要和块一同存储，用一个单独的系统就可以管理这些块的元数据；数据块适合用于数据备份进而提供数据容错能力和提高可用性；有利于负载均衡。

### NameNode

HDFS的维护进程，用来管理文件系统的命名空间，负责记录文件是如何分割成数据块，以及这些数据块分别被存储到那些数据节点上，它的主要功能是对内存及IO进行集中管理。当一个客户端请求一个文件或者存储一个文件时，它需要先知道具体到哪个DataNode上存取，获取这些信息后，客户端再直接和这个DataNode进行交互，而这些信息的维护者就是NameNode。

NameNode中保存元信息的种类有：文件名目录名及它们直接的层级关系；文件目录的所有者极其权限；每个文件块的名及文件有哪些块组成。需要特别注意的是，NameNode元信息并不包含每个块的位置信息，这些信息会在NameNode启动时从各个DataNode获取并保存在内存中，因为这些信息会在系统启动时由数据节点重建。把块数据信息存放在内存中，在读取数据时，减少查询时间，增加读取效率。NameNode也会实时通过心跳机制和DataNode进行交互，实时检查文件系统是否运行正常。NameNode元信息会保存各个块的名称以及文件由哪些块组成。

元信息的持久化。fsimage文件中存放NameNode元信息。在系统运行期间，所有对元信息的操作都保存在内存中并被持久化到另一个文件edits中，并且edits文件和fsimage文件会被SecondaryNameNode周期性合并。

### DataNode

DataNode是HDFS中的Works节点，负责存储数据块，也负责为系统客户端提供数据块的读写服务，同时还会根据NameNode的指示进行创建、删除、复制等操作。此外，通过心跳定期向NameNode发送所存储文件块列表信息。当对HDFS文件系统进行读写时，NameNode告知客户端每个数据驻留在哪个DataNode，客户端直接与DataNode进行通信。DataNode还会与其他DataNode进行通信，复制这些块以实现冗余。

### SecondaryNameNode

SecondaryName的作用是定期合并edits和fsimage文件。合并的步骤为：合并之前告知NameNode把所有的操作写到新的edits文件并将其命名为edits.new；SecondaryNameNode从NameNode请求fsimage和edits文件；把fsimage和edits文件合并为新的fsimage文件；NameNode从SecondaryNameNode获取合并好的新的fsimage并将旧的替换掉，并把edits用edits.new文件替换掉；更换fstime文件中的检查点。合并过程中，涉及NameNode中的相关文件有：

fsimage： 保存上个检查点的HDFS的元信息。

edits：保存从上个检查点开始发生的HDFS元信息状态改变信息。

fstime：保存最后一个检查点的时间戳。

### 数据备份

HDFS设计成能可靠地在集群中大量机器之间存储大量的文件，它以块序列的形式存储文件，并通过备份数据块的形式实现容错。文件中除了最后一个数据块，其他块都有相同的大小。属于文件的块为了故障容错而被复制。块的大小和复制数是以文件为单位进行配置的，应用可以在文件创建时或者之后修改复制因子。HDFS中的文件是一次写的，并且任何时候都只有一个写操作。

NameNode负责处理所有的块复制相关的决策。它周期性地接受集群中数据节点的心跳和数据块报告。一个心跳的到达表示这个数据节点是正常的。一个块报告包括该数据节点上所有数据块的列表。

**副本存放位置**

数据块副本存放位置的选择严重影响HDFS的可靠性和性能。副本存放位置的优化是HDFS区分于其他分布式文件系统的的特征，这需要精心的调节和大量的经验。机架敏感的副本存放策略是为了提高数据的可靠性，可用性和网络带宽的利用率。副本存放策略的实现是这个方向上比较原始的方式。短期的实现目标是要把这个策略放在生产环境下验证，了解更多它的行为，为以后测试研究更精致的策略打好基础。

HDFS运行在跨越大量机架的集群之上。两个不同机架上的节点是通过交换机实现通信的，在大多数情况下，相同机架上机器间的网络带宽优于在不同机架上的机器。

在开始的时候，每一个数据节点自检它所属的机架id，然后在向名字节点注册的时候告知它的机架id。HDFS提供接口以便很容易地挂载检测机架标示的模块。一个简单但不是最优的方式就是将副本放置在不同的机架上，这就防止了机架故障时数据的丢失，并且在读数据的时候可以充分利用不同机架的带宽。这个方式均匀地将复制分散在集群中，这就简单地实现了组建故障时的负载均衡。然而这种方式增加了写的成本，因为写的时候需要跨越多个机架传输文件块。

默认的HDFS block放置策略在最小化写开销和最大化数据可靠性、可用性以及总体读取带宽之间进行了一些折中。一般情况下复制因子为3，HDFS的副本放置策略是将第一个副本放在本地节点，将第二个副本放到本地机架上的另外一个节点而将第三个副本放到不同机架上的节点。这种方式减少了机架间的写流量，从而提高了写的性能。机架故障的几率远小于节点故障。这种方式并不影响数据可靠性和可用性的限制，并且它确实减少了读操作的网络聚合带宽，因为文件块仅存在两个不同的机架， 而不是三个。文件的副本不是均匀地分布在机架当中，1/3在同一个节点上，1/3副本在同一个机架上，另外1/3均匀地分布在其他机架上。这种方式提高了写的性能，并且不影响数据的可靠性和读性能。

**副本的选择**

为了尽量减小全局的带宽消耗读延迟，HDFS尝试返回给一个读操作离它最近的副本。假如在读节点的同一个机架上就有这个副本，就直接读这个，如果HDFS集群是跨越多个数据中心，那么本地数据中心的副本优先于远程的副本。

**安全模式**

在NameNode启动的时候，会进入一个叫做安全模式的特殊状态。安全模式中不允许发生文件块的复制。NameNode接受来自DataNode的心跳和块报告。一个块报告包含数据节点所拥有的数据块的列表。

每一个块有一个特定的最小复制数。当名字节点检查这个块已经大于最小的复制数就被认为是安全地复制了，当达到配置的块安全复制比例时（加上额外的30秒），NameNode就退出安全模式。它将检测数据块的列表，将小于特定复制数的块复制到其他的数据节点。

## 文件操作过程

本小节将介绍HDFS读文件、写文件、删除文件的具体过程。

### 读文件

客户端将要读取的文件路径发送给NameNode，NameNode获取文件的元信息（主要是数据块的存放位置信息）返回给客户端，客户端根据返回的信息找到相应DataNode逐个获取文件的数据块并在客户端本地进行数据追加合并从而获得整个文件。具体过程为：

（1）客户端调用FileSyste对象的open()方法在分布式文件系统中打开要读取的文件。

　　（2）分布式文件系统通过使用RPC（远程过程调用）来调用NameNode，确定文件起始块的位置。

　　（3）分布式文件系统的DistributedFileSystem类返回一个支持文件定位的输入流FSDataInputStream对象，FSDataInputStream对象接着封装DFSInputStream对象（存储着文件起始几个数据块的DataNode地址），客户端对这个输入流调用read()方法。

　　（4）DFSInputStream连接距离最近的DataNode，通过反复调用read方法，将数据从DataNode传输到客户端。

　　（5）到达数据块的末端时，DFSInputStream关闭与该DataNode的连接，寻找下一个块的最佳DataNode。

（6）客户端完成读取，对FSDataInputStream调用close()方法关闭连接。

下图给出读文件的过程。



图 4. 3 读文件过程

需要注意的是，从某个DataNode获取的数据块有可能是损坏的，损坏可能是由DataNode的存储设备错误、网络错误或者软件bug造成的。HDFS客户端软件实现了对HDFS文件内容的校验和(checksum)检查。当客户端创建一个新的HDFS文件，会计算这个文件每个数据块的校验和，并将校验和作为一个单独的隐藏文件保存在同一个HDFS名字空间下。当客户端获取文件内容后，它会检验从Datanode获取的数据跟相应的校验和文件中的校验和是否匹配，如果不匹配，客户端可以选择从其他Datanode获取该数据块的副本。

### 写文件

客户端要向HDFS写数据，首先要跟NameNode通信以确认可以写文件并获得接收文件数据块的DataNode，然后客户端按顺序将文件逐个数据块传递给相应DataNode，并由接收到数据块的DataNode负责向其他DataNode复制数据块的副本。具体过程为：

1. 客户端向NameNode发送上传文件请求， NameNode对要上传目录和文件进行检查，判断是否可以上传，并向客户端返回检查结果。

2）客户端得到上传文件的允许后读取客户端配置，如果没有指定配置则会读取默认配置（例如副本数和块大小默认为3和128M，副本是由客户端决定的）。向NameNode请求上传一个数据块。

3）NameNode会根据客户端的配置来查询DataNode信息，如果使用默认配置，那么最终结果会返回同一个机架的两个DataNode和另一个机架的DataNode。这称为“机架感知”策略。

机架感知：HDFS采用一种称为机架感知(rack-aware)的策略来改进数据的可靠性、可用性和网络带宽的利用率。大型HDFS实例一般运行在跨越多个机架的计算机组成的集群上，不同机架上的两台机器之间的通讯需要经过交换机。在大多数情况下，同一个机架内的两台机器间的带宽会比不同机架的两台机器间的带宽大。通过一个机架感知的过程，NameNode可以确定每个DataNode所属的机架id。一个简单但没有优化的策略就是将副本存放在不同的机架上。这样可以有效防止当整个机架失效时数据的丢失，并且允许读数据的时候充分利用多个机架的带宽。这种策略设置可以将副本均匀分布在集群中，有利于当组件失效情况下的负载均衡。但是，因为这种策略的一个写操作需要传输数据块到多个机架，这增加了写的代价。在大多数情况下，副本系数是3，HDFS的存放策略是将一个副本存放在本地机架的节点上，一个副本放在同一机架的另一个节点上，最后一个副本放在不同机架的节点上。这种策略减少了机架间的数据传输，这就提高了写操作的效率。机架的错误远远比节点的错误少，所以这个策略不会影响到数据的可靠性和可用性。于此同时，因为数据块只放在两个（不是三个）不同的机架上，所以此策略减少了读取数据时需要的网络传输总带宽。在这种策略下，副本并不是均匀分布在不同的机架上。三分之一的副本在一个节点上，三分之二的副本在一个机架上，其他副本均匀分布在剩下的机架中，这一策略在不损害数据可靠性和读取性能的情况下改进了写的性能。

4）客户端在开始传输数据块之前会把数据缓存在本地，当缓存大小超过了一个数据块的大小，客户端就会从NameNode获取要上传的DataNode列表。之后会在客户端和第一个DataNode建立连接开始流式的传输数据，这个DataNode会一小部分一小部分（4K）的接收数据然后写入本地仓库，同时会把这些数据传输到第二个DataNode，第二个DataNode也同样一小部分一小部分的接收数据并写入本地仓库，同时传输给第三个DataNode，依次类推。这样逐级调用和返回之后，待这个数据块传输完成客户端后告诉NameNode数据块传输完成，这时候NameNode才会更新元数据信息记录操作日志。

5）第一个数据块传输完成后会使用同样的方式传输下面的数据块直到整个文件上传完成。

下图给出写文件的过程。



图 4. 4 写文件过程

细节：

a.请求和应答是使用RPC的方式，客户端通过ClientProtocol与NameNode通信，NameNode和DataNode之间使用DatanodeProtocol交互。在设计上，NameNode不会主动发起RPC，而是响应来自客户端或DataNode的RPC请求。客户端和DataNode之间是使用socket进行数据传输，和NameNode之间的交互采用nio封装的RPC。

b.HDFS有自己的序列化协议。

c.在数据块传输成功后但客户端没有告诉NameNode之前如果NameNode宕机那么这个数据块就会丢失。

d.在流式复制时，逐级传输和响应采用响应队列来等待传输结果。队列响应完成后返回给客户端。

c.在流式复制时如果有一台或两台（不是全部）没有复制成功，不影响最后结果，只不过DataNode会定期向NameNode汇报自身信息。如果发现异常NameNode会指挥DataNode删除残余数据和完善副本。如果副本数量少于某个最小值就会进入安全模式。

### 删除文件

HDFS删除数据比较流程相对简单，只列出详细步骤:

　　1）客户端向NameNode发起RPC调用，请求删除文件。NameNode检查合法性。

　　2）NameNode查询文件相关元信息，向存储文件数据块的DataNode发出删除请求。

　　3）DataNode删除相关数据块。返回结果。

　　4）NameNode返回结果给客户端。

　　当用户或应用程序删除某个文件时，这个文件并没有立刻从HDFS中删除。实际上，HDFS会将这个文件重命名转移到/trash目录。只要文件还在/trash目录中，该文件就可以被迅速地恢复。文件在/trash中保存的时间是可配置的，当超过这个时间时，NameNode就会将该文件从名字空间中删除。删除文件会使得该文件相关的数据块被释放。注意，从用户删除文件到HDFS空闲空间的增加之间会有一定时间的延迟。只要被删除的文件还在/trash目录中，用户就可以恢复这个文件。如果用户想恢复被删除的文件，可以浏览/trash目录找回该文件。/trash目录仅仅保存被删除文件的最后副本。/trash目录与其他的目录没有什么区别，除了一点：在该目录上HDFS会应用一个特殊策略来自动删除文件。目前的默认策略是删除/trash中保留时间超过6小时的文件。将来，这个策略可以通过一个被良好定义的接口配置。

当一个文件的副本系数被减小后，NameNode会选择过剩的副本删除。下次心跳检测时会将该信息传递给DataNode。DataNode遂即移除相应的数据块，集群中的空闲空间加大。同样，在调用setReplication API结束和集群中空闲空间增加间会有一定的延迟。

## HDFS安装与配置

### 安装Hadoop

1）解压

在/usr/software目录下鼠标右键点击【Open in Terminal】，打开终端，将hadoop压缩包到/usr/local

**sudo tar zxvf hadoop-2.7.2.tar.gz -C /usr/local**

第二步，配置环境变量：

**终端中输入sudo gedit ~/.bashrc**在.bashrc文档最后加入如下字段：

**#HADOOP**

**export HADOOP\_HOME=/usr/local/hadoop-2.7.2**

**export PATH=$PATH:$HADOOP\_HOME/bin**

**export PATH=$PATH:$HADOOP\_HOME/sbin**

点击【Save】保存修改，然后在终端中输入**source ~/.bashrc**使配置生效

2）设置ssh免密钥登录

免秘钥登录是指两台Linux机器之间使用SSH连接时不需要用户名和密码。SSH是一种在不安全网络上提供安全远程登录及其他安全网络服务的协议。默认情况下，SSH连接需要密码认证，本文中，我们修改系统认证，使系统通信免除密码输入和SSH认证。有利于部署和使用Spark集群。

设置SSH免密钥登录的步骤如下:

1）打开终端，依次输入执行：

**sudo apt-get install ssh**

**sudo apt-get install rsync**

2）生成ssh密码，在终端中输入：

**ssh-keygen –t rsa**

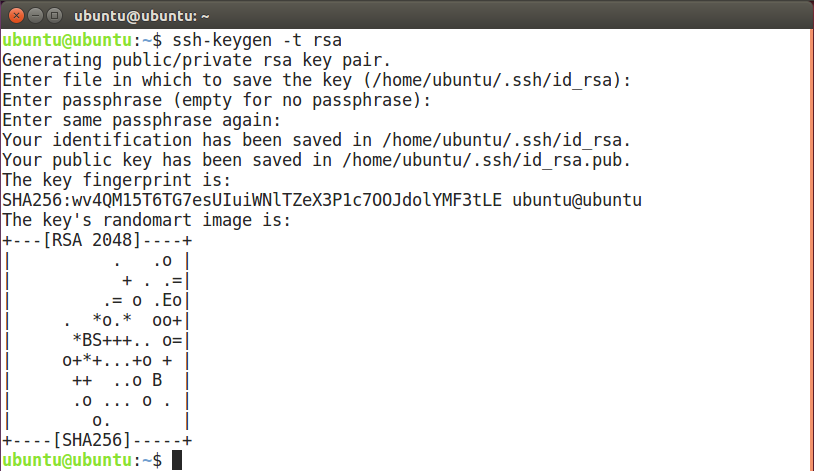


图 4. 5 生成密码

**ll /home/ubuntu/.ssh** #路径与设定的虚拟机名有关，本例中虚拟机名为ubuntu

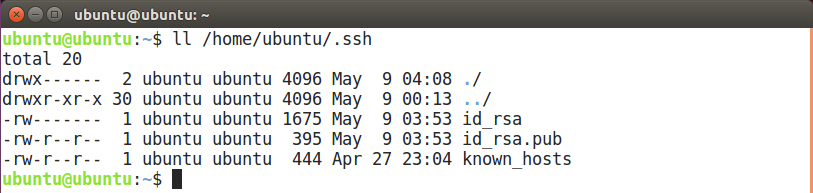


图 4. 6 查看密码位置方法1

**ll ~/.ssh**

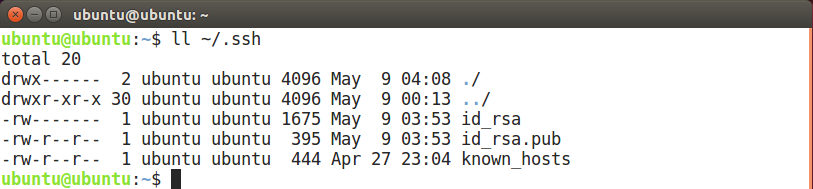


图 4. 7查看密码位置方法2

3）将公钥文件复制到.ssh目录

**cat ~/.ssh/id\_rsa.pub >> ~/.ssh/authorized\_keys**

### 修改Hadoop文件

**①修改环境变量文件hadoop-env.sh**

Hadoop的环境变量文件在hadoop的安装目录（/usr/local/hadoop-2.7.2）下，我们只需要配置该文件的JDK路径即可。打开终端，输入如下命令：

**sudo gedit etc/hadoop/hadoop-env.sh**

用gedit编辑器修改文件内容，修改

**export JAVA\_HOME = ${JAVA\_HOME}**为export JAVA\_HOME=/usr/local/jdk1.8.0\_171

保存并关闭

**②修改核心组件文件core-site.xml**

指定hadoop中HDFS的NameNode的地址。

在hadoop的安装目录（/usr/local/hadoop-2.7.2）下，打开终端，输入如下命令：

**sudo gedit etc/hadoop/core-site.xml**

在configuration中间加入以下配置：

<property>

<name>fs.default.name</name>

<value>hdfs://localhost:9000</value>

</property>

如下图：



图 4. 8修改core-site.xml

保存后退出

**③修改yarn-site.xml（指定YARN的ResourceManager的地址）**

在hadoop的安装目录（/usr/local/hadoop-2.7.2）下，打开终端，输入如下命令：

**sudo gedit etc/hadoop/yarn-site.xml**

在<configuration>和</configuration>中间加入以下配置：

**<property>**

**<name>yarn.nodemanager.aux-services</name>**

**<value>mapreduce\_shuffle</value>**

**</property>**

**<property>**

**<name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>**

**<value>org.apache.hadoop.mapred.ShuffleHandler</value>**

**</property>**

如下图：

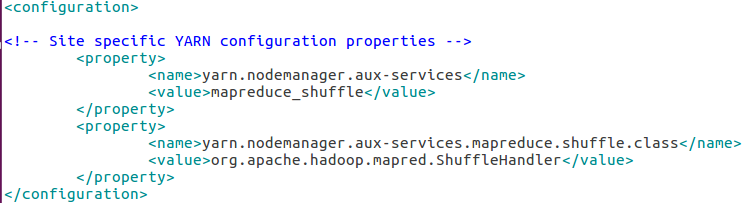


图 4. 9 修改yarn-site.xml

保存后退出

**④修改mapred-site.xml（指定mapreduce运行在YARN上）**

在hadoop的安装目录（/usr/local/hadoop-2.7.2）下，打开终端，输入如下命令：

复制模板然后修改：

**sudo cp etc/hadoop/mapred-site.xml.template etc/hadoop/mapred-site.xml**

**sudo gedit etc/hadoop/mapred-site.xml**

在<configuration>和</configuration>中间加入以下配置：

**<property>**

**<name>mapreduce.framework.name</name>**

**<value>yarn</value>**

**</property>**

如下图：

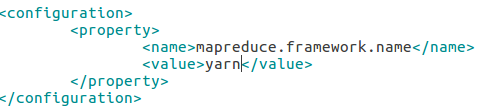


图 4. 10 修改mapred-site.xml

保存后退出

**⑤修改文件系统hdfs-site.xml（指定HDFS副本的数量，指定datanode、namenode的日志记录位置）**

在hadoop的安装目录（/usr/local/hadoop-2.7.2）下，打开终端，输入如下命令：

**sudo gedit etc/hadoop/hdfs-site.xml**

在configuration中间加入以下配置：

<property>

<name>dfs.replication</name>

<value>3</value>

</property>

<property>

<name>dfs.namenode.name.dir</name>

<value>file:/usr/local/hadoop-2.7.2/hadoop\_data/hdfs/namenode</value>

</property>

<property>

<name>dfs.datanode.data.dir</name>

<value>file:/usr/local/hadoop-2.7.2/hadoop\_data/hdfs/datanode</value>

</property>

如下图：

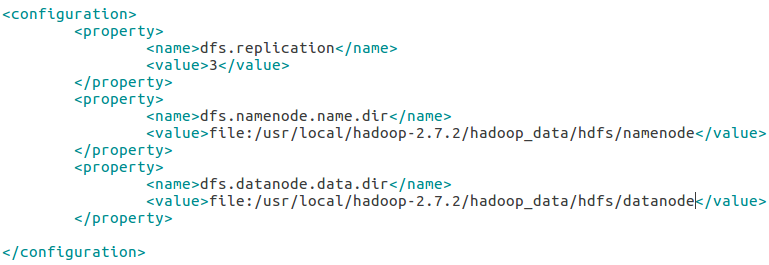


图 4. 11 hdfs-site.xml

保存后退出

**⑥建立与格式化HDFS目录**

**sudo mkdir -p /usr/local/hadoop-2.7.2/hadoop\_data/hdfs/namenode**

**sudo mkdir -p /usr/local/hadoop-2.7.2/hadoop\_data/hdfs/datanode**

修改权限：终端中输入命令

**sudo chown ubuntu:ubuntu -R /usr/local/hadoop-2.7.2**

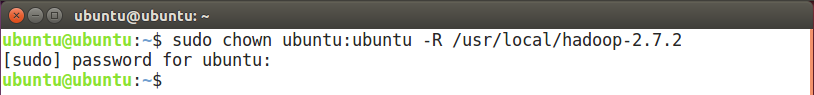


图 4. 12 设置所有者

**hadoop namenode -format**



图 4. 13 格式化目录

### 启动hadoop

两种启动方式：

①先启动**start-dfs.sh**再启动**start-yarn.sh**

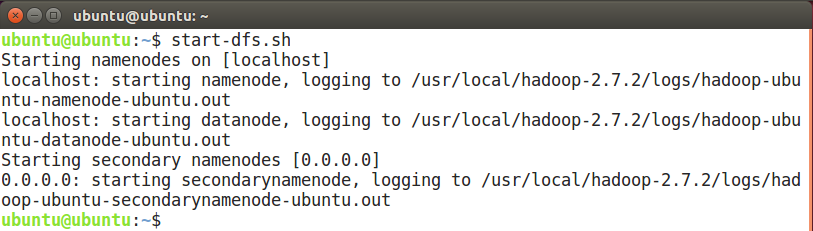


图 4. 14 start-dfs.sh

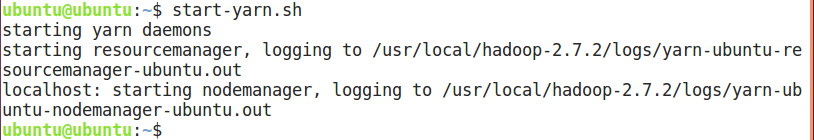


图 4. 15 start-yarn.sh

②启动全部

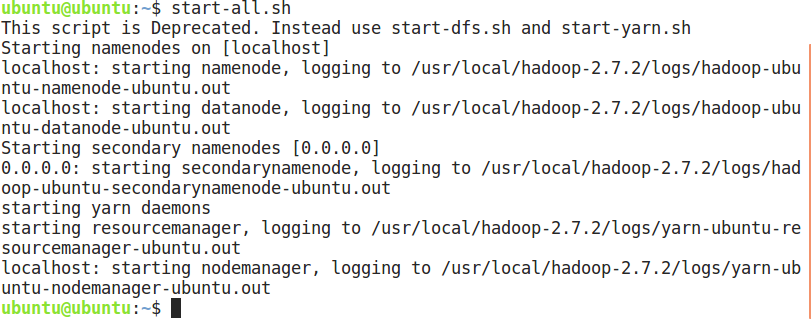


图 4. 16 start-all.sh

对应的，也存在两种结束方式：①stop-dfs.sh ，stop-yarn.sh②stop-all.sh

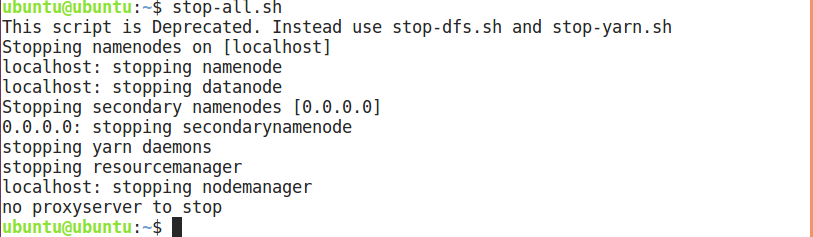


图 4. 17 stop-all.sh

5）查看目前所执行的状态

终端中输入 jps，如下图所示，则配置成功。

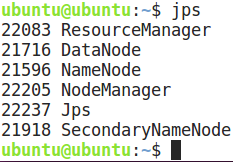


图 4. 18 jps查询状态

6）开启Hadoop ResourceManager Web界面

Hadoop ResourceManager Web界面网址：<http://localhost:8088/>

在Ubuntu自带的Firefox浏览器中打开上述网页：

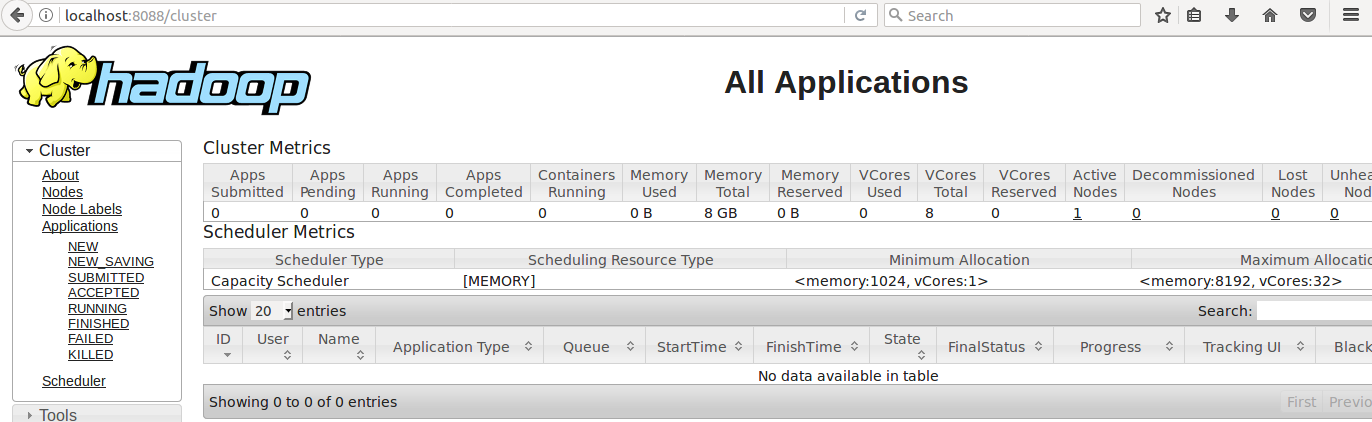


图 4. 19 ResourceManager状态

NameNode HDFS Web界面

开启HDFS Web UI网址：<http://lcoalhost:50070/>

在Ubuntu自带的Firefox浏览器中打开上述网页：

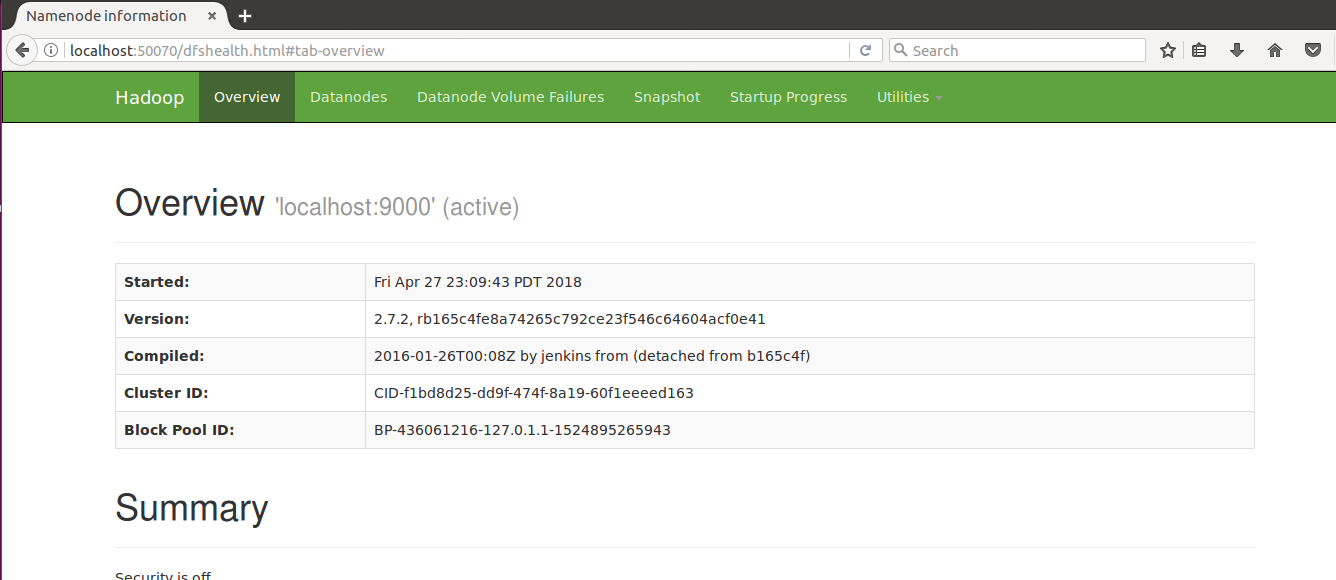


图 4. 20 NameNode状态

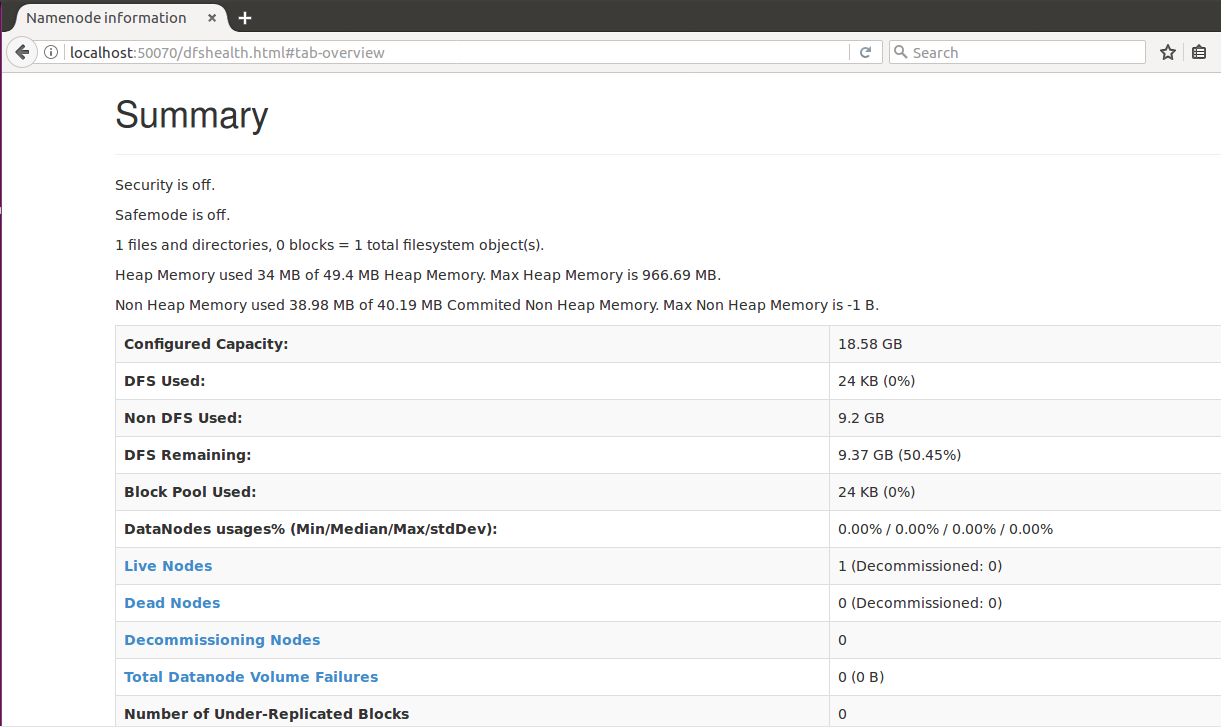


图 4. 21 NameNode状态详情

## HDFS基本命令

进行本节的学习，首先在终端中启动hadoop集群（输入start-all启动集群）。要想了解Hadoop命令，输入hadoop fs -help即可查看所有命令的帮助文件。多加练习，才会真正掌握命令的使用方法。本节将介绍创建目录、查询HDFS文件列表、上传文件到HDFS、下载文件到本地、查看HDFS文件内容、删除HDFS文件等常用命令。

### 创建目录

在Hadoop上创建目录与在Linux中创建目录相似，均用-mkdir命令，差别在于Hadoop所有的命令行均由bin/hadoop脚本引发，创建目录的命令为：

**hadoop fs -mkdir 路径/目录名**

例如在根目录下创建一个名为first的目录，命令为：**hadoop fs -mkdir /first**

命令的输入情形如下图所示：



图 4. 22 创建first目录

然后在first目录下，创建一个名为inputfile的目录。命令为：

**hadoop fs -mkdir /first/inputfile**

需要注意的是：未创建first目录情况下，直接创建first目录下的inputfile的目录会报错。

### 查看文件列表

查询文件列表的命令，与Linux中查询目录文件列表的命令，类似，均为ls。Hadoop脚本可触发的查询文件列表的命令的完整格式为：**hadoop fs -ls <args>**

其中，<args>为可选参数。例如，查询根目录下的子目录和文件，命令为：

hadoop fs -ls /

如果事先在根目录下已经创建了second和third两个目录，执行上述查询命令，会显示如下图所示信息。

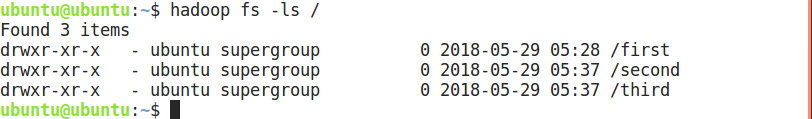


图 4. 23 查询文件列表

在上一小节中，first目录下，创建了inputfile目录。查询目录下所有子目录和子文件的命令为：**hadoop fs -ls -R 目录路径**

例如查询HDFS中所有目录和文件的命令为：hadoop fs -ls -R /

查询结果如下图所示：

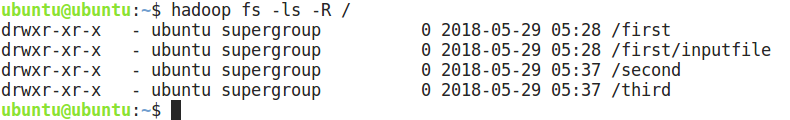


图 4. 24 查询目录下所有目录、文件

### 上传文件到HDFS

上传文件到HDFS的过程，即本地文件复制到HDFS集群中，有以下两种命令可以使用：

1. **hadoop fs -put 本地文件路径 HDFS保存文件路径**
2. **hadoop fs -copyFromLocal -f 本地文件路径 HDFS保存文件路径**

例如，将本地文件“/usr/sparkwc”上传到HDFS的“/data”目录下，首先需要确定本地文件存在，以及HDFS中根目录下data目录也存在。然后终端中输入命令：

hadoop fs -put /usr/sparkwc /data

或：

hadoop fs -copyFromLocal -f /usr/sparkwc /data

回车执行以后，可以使用查询文件列表命令，查询到文件已经上传到HDFS集群中。如下图所示。

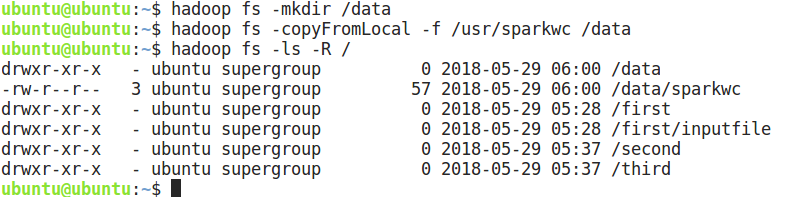


图 4. 25 上传文件

### 下载文件到本地

下载HDFS中文件到本地的过程，即文件从HDFS集群中复制到本地，有以下两种命令可以使用：

**1.hadoop fs -get HDFS文件路径 本地保存文件路径**

**2.hadoop fs -copyToLocal -f HDFS文件路径 本地保存文件路径**

例如，将HDFS的“/data”目录下文件sparkwc复制到本地目录“/home/ubuntu”下，首先需要确定HDFS中文件存在，以及本地目录合法（存在）。然后终端中输入命令：

Hadoop fs -get /data/sparkwc /home/ubuntu/sparkwc

或：

Hadoop fs -copyToLocal -f /data/sparkwc /home/ubuntu/sparkwc

回车执行以后，可以查询本地/home/ubuntu目录，查询到文件已经复制到本地。需要注意的是：目录读写权限。

### 查看HDFS文件内容

查询文本文件的内容，一般使用的命令为：

**hadoop fs -cat 文件路径**

例如查询HDFS中/data/sparkwc（此文件必须存在），在终端中输入命令：

hadoop fs -cat /data/sparkwc 查询到结果如下图所示。

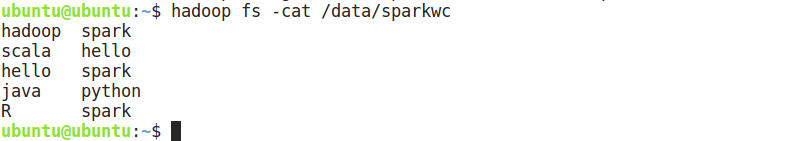


图 4. 26 查看文本内容

### 删除HDFS文件

删除HDFS中文件类似于Linux中删除文件，命令格式为：

**hadoop fs -rm 文件路径**

例如删除HDFS中“/data”中的sparkwc。在终端中输入命令为：

hadoop fs -rm /data/sparkwc

如下图所示：

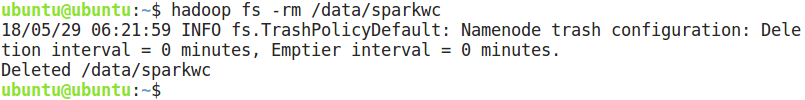


图 4. 27 删除文件

如果需要删除目录下所有的文件和子目录，则可以通过以下命令实现：

**hadoop fs -rm -R 目录路径**

## HDFS与spark-shell

本节中，我们将介绍wordcount示例读取hdfs中的文档，并将输出结果存入hdfs文件系统中的整个流程以及SparkWEB对于任务的监控。

**第一步，启动HDFS**，

在终端中，输入：start-all.sh(或者先执行start-dfs.sh，再执行start-yarn.sh）。然后输入jps查看状态。

hadoop命令：**hadoop fs -ls /** 查看根目录下的所有文件，初始为空。如下图所示。然后，新建一个data目录，终端中输入命令：

**hadoop fs -mkdir /data**

在hdfs中，成功新建一个data目录，存储wordcount计数输入的单词文档。

**hadoop fs -ls /**

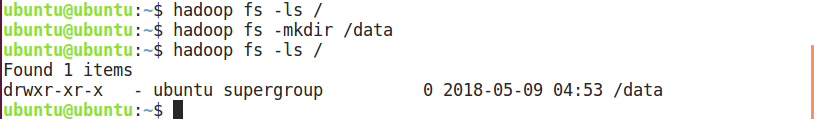


图 4. 28 hdfs目录中文件

复制/usr中的sparkwc文件到hdfs目录下的data目录中。命令如下：

**hadoop fs -put /usr/sparkwc /data**

输入hdfs中当前根目录下的所有文件：

**hadoop fs -ls -R /**

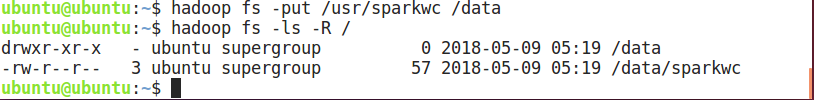


图 4. 29 复制文件到hdfs目录

**第二步，启动spark-shell**

新建一个终端，输入：spark-shell启动spark：

为展示scala函数编程的特点以及伪分布式：将上一章中的wordcount实例整理如下，在启动以后的spark中输入：

**sc.textFile(“hdfs://127.0.0.1:9000/data/sparkwc”).flatMap(\_.split(“\t”)).map((\_,1)).reduceByKey(\_+\_).saveAsTextFile(“hdfs://127.0.0.1:9000/data/out”)**



图 4. 30 基于HDFS的wordcount

在hdfs中查看/data/out目录下文件，查看运行结果：

**hadoop fs -ls /data/out**

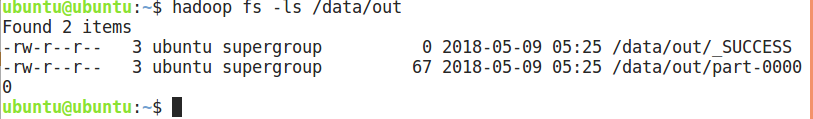


图 4. 31 查看运行结果

由上图可知，运行结果保存在/data/out/part-00000中，查看内容：

**hadoop fs -cat /data/out/part-00000**

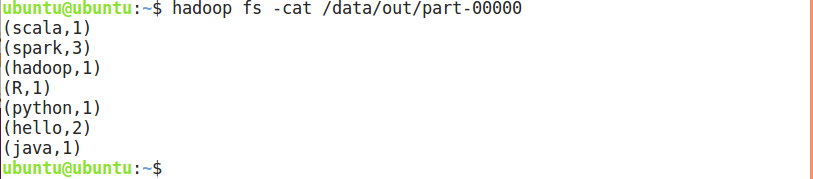


图 4. 32 查看结果内容

第三步，查看SparkWEB监控。

SparkWEB监控的地址在spark-shell启动时，显示如下图框中地址：



图 4. 33 查看结果内容

在ubuntu自带的FireFox浏览器中输入网址：localhost:4040或者192.168.241.128:4040，回车进入SparkWEB监控页面。如下图所示。

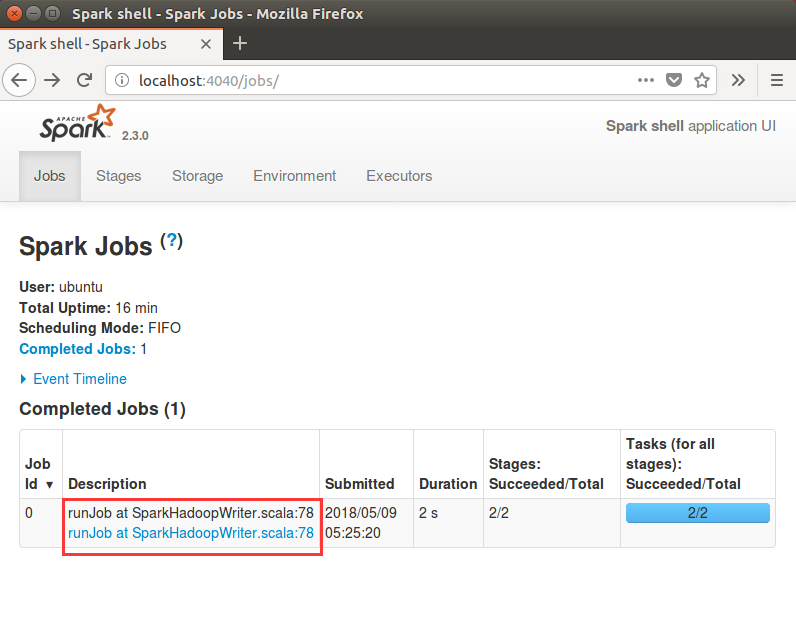


图 4. 34 SparkWEB

点击上图框中内容，进入任务Stages详情页面，如图所示。点击图中【DAG Visualization】，查看任务中的所有操作，以及根据宽依赖、窄依赖划分的stage。如图所示。

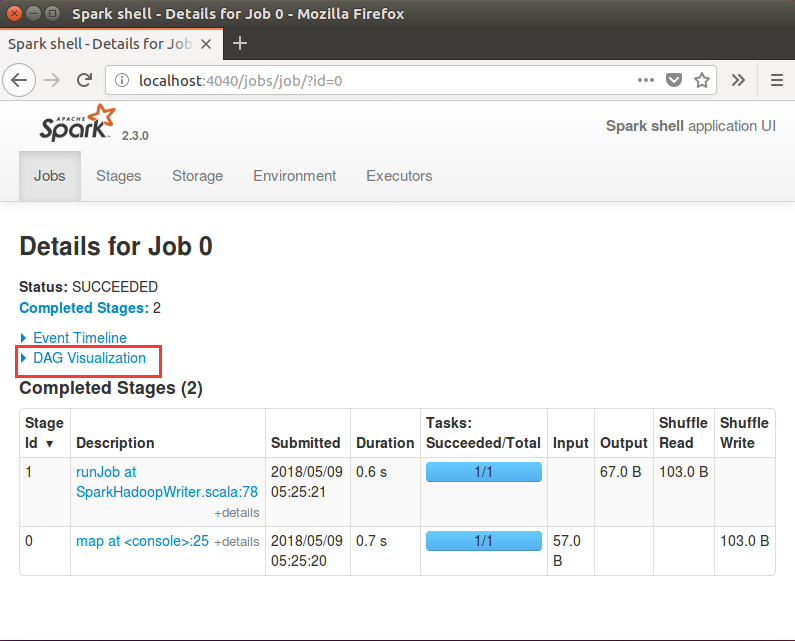


图 4. 35 查看stage数

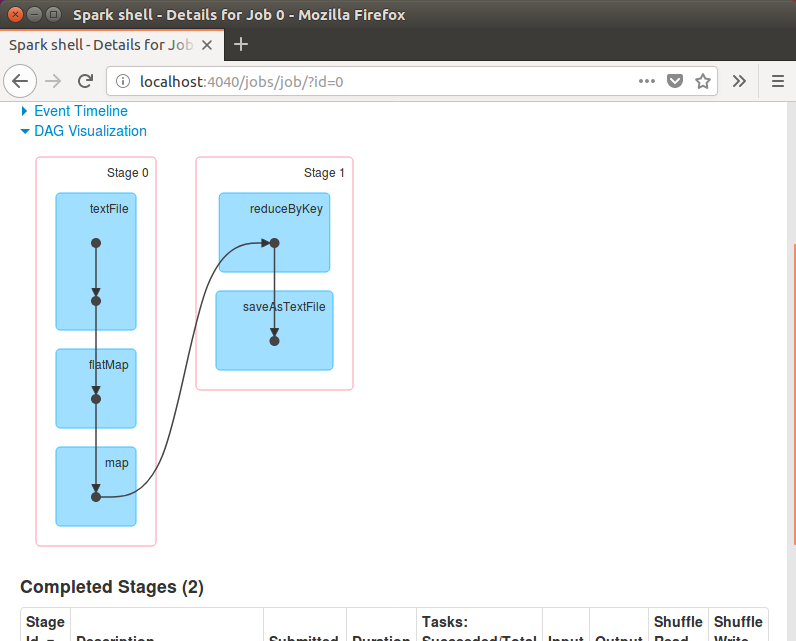


图 4. 36 查看DAG