第3章

视频讲解

Hadoop分布式系统

3.1 Hadoop 概述

众所周知，Hadoop 是一个由 Apache 基金会开发的分布式系统基础架构，可安装在一个商用机器集群中，使机器可彼此通信并协同工作，以高度分布式的方式共同存储和处理大量数据。简单地说，Hadoop 是一个海量数据分布式处理的开源软件框架，被部署到一个集群上。

3.1.1 Hadoop 简介

Hadoop 是一个能够让用户轻松架构和使用的分布式计算平台。用户可以轻松地在 Hadoop 上开发和运行处理海量数据的应用程序。

Hadoop 是以一种可靠、高效、可伸缩的方式进行处理的。Hadoop 是可靠的，因为它假设计算元素和存储会失败，所以它维护多个工作数据副本，确保能够针对失败的节点重新分布处理。Hadoop 是高效的，因为它以并行的方式工作，通过并行处理加快处理速度。Hadoop 还是可伸缩的，能够处理 PB 级数据。此外，Hadoop 依赖于社区服务器，因此它的成本比较低，任何人都可以使用。它主要有以下几个优点。

(1) 高可靠性。Hadoop 按位存储和处理数据的能力值得人们信赖。

(2) 高扩展性。Hadoop 是在可用的计算机集簇间分配数据并完成计算任务的，这些集簇可以方便地扩展到数以千计的节点中。

(3) 高效性。Hadoop 能够在节点之间动态地移动数据，并保证各个节点的动态平衡，因此处理速度非常快。

(4) 高容错性。Hadoop 能够自动保存数据的多个副本，并且能够自动将失败的任务重新分配。

3.1.2　Hadoop的发展历程

2003年，Google发表了第一篇关于其云计算核心技术GFS的论文。该论文呈现出Doug Cutting等在研发Apache开源项目Nutch搜索引擎时，正面临着如何将其架构扩展到可以处理数10亿规模网页的难题。继而，他们于2004年编写了一个开放源代码的类似系统——Nutch分布式文件系统（Nutch Distributed File System，NDFS）。同年，Google在著名的OSDI国际会议上发表了一篇题为*MapReduce：Simplified Data Processing on Large Clusters*的论文，简要介绍MapReduce的基本设计思想。该论文发表后，MapReduce编程模型在解决大型分布式并行计算问题上具有极大的可操作性。紧接着，Nutch团队尝试依据Google MapReduce的设计思想，模仿Google MapReduce框架的设计思路，用Java设计实现出了一套新的MapReduce并行处理软件系统，并将其与Nutch分布式文件系统NDFS结合，用以支持Nutch搜索引擎的数据处理。2006年，他们把NDFS和MapReduce从Nutch项目中分离出来，成为一套独立的大规模数据处理软件系统。有意思的是这个系统的命名是使用Doug Cutting小儿子当时牙牙学语称呼自己的玩具小象的名字“Hadoop”。Hadoop能支持PB级海量数据，可扩展性强。可靠、高效、可扩展和开源的特性，使得Hadoop技术得到了迅猛发展，并在2008年成为Apache的顶级项目。

自从2006年Hadoop正式成为Apache开源组织的独立项目后，由于其低成本、高性能，深受广大用户的欢迎，经过短短几年的发展，Hadoop及其技术在不断地改进完善中，目前已形成一个强大的系统。

下面列举了Hadoop在成为独立项目后的发展与演进中的重要事件，以便大家了解Hadoop的发展历程。

（1）Hadoop最初是由Apache Lucene项目的创始人Doug Cutting开发的文本搜索库。Hadoop源自于2002年的Apache Nutch项目———一个开源的网络搜索引擎，并且也是Lucene项目的一部分。

（2）2004年，Nutch项目也模仿GFS开发了自己的分布式文件系统NDFS（Nutch Distributed File System），也就是HDFS的前身。

（3）2004年，谷歌公司又发表了另一篇具有深远影响的论文，阐述了MapReduce分布式编程的思想。

（4）2005年，Nutch开源实现了谷歌的MapReduce。

（5）2006年2月，Apache Hadoop项目正式启动以支持MapReduce和HDFS的独立发展。

（6）2007年4月，雅虎公司实现了包含1000个计算节点的Hadoop集群。

（7）2008年，淘宝开始投入研究基于Hadoop的系统——云梯，并将其用于处理电子商务相关数据。云梯1的总容量大概为9.3PB，包含了1100台机器，每天处理约18 000道作业，扫描500TB数据。

（8）2008年1月，Hadoop成为Apache顶级项目，获得了业界更为广泛的关注。

（9）2008年2月，雅虎公司宣布其搜索引擎产品部署在一个拥有1万个内核的

Hadoop 集群上。

(10) 2008 年 7 月,Hadoop 打破 1TB 数据排序基准测试记录。雅虎公司的一个 Hadoop 集群用 209 秒完成 1TB 数据的排序,比上一年的纪录保持者保持的 297 秒快了近 90 秒。

(11) 2009 年 5 月,雅虎的团队使用 Hadoop 对 1TB 的数据进行排序只花了 62 秒时间。

(12) 2009 年 7 月,Hadoop Core 项目更名为 Hadoop Common; MapReduce 和 HDFS 成为 Hadoop 项目的独立子项目; Avro 和 Chukwa 成为 Hadoop 新的子项目。

(13) 2010 年 5 月,Avro 数据传输中间件和 HBase 数据库从 Hadoop 项目中脱离出来,成为 Apache 顶级项目。此外,IBM 提供了基于 Hadoop 的大数据分析软件——InfoSphere BigInsights,包括基础版和企业版。

(14) 2010 年 9 月,Hive 数据仓库工具和 Pig 数据分析平台从 Hadoop 项目中脱离出来,成为 Apache 顶级项目。

(15) 2011 年 1 月,ZooKeeper 脱离 Hadoop,成为 Apache 顶级项目。

(16) 2011 年 5 月,Mapr Technologies 公司推出分布式文件系统和 MapReduce 引擎——Mapr Distribution for Apache Hadoop。该项目由 Hortonworks 在 2010 年 3 月提出,HCatalog 主要用于解决数据存储、元数据的问题,主要解决 HDFS 的瓶颈,它提供了一个地方来存储数据的状态信息,这使得数据清理和归档操作可以很容易地进行。

(17) 2011 年 8 月,Cloudera 公布了一项有益于合作伙伴生态系统的计划——创建一个生态系统,以便硬件供应商、软件供应商及系统集成商可以一起探索如何使用 Hadoop 更好地洞察数据。

(18) 2011 年 12 月,Hadoop1.0.0 版本发布,标志着 Hadoop 技术进入成熟期。

(19) 2012 年 5 月,Hadoop 发布 2.0 Alpha 版本,对 MapReduce、HDFS 等部分进行了重大改进,标志着 Hadoop 技术进入一个新的发展阶段。

(20) 2013 年 8 月,Hadoop1.2.1 稳定版发布。

(21) 2014 年 2 月,Spark 逐渐代替 MapReduce 成为 Hadoop 的默认执行引擎,并成为 Apache 基金会顶级项目。

(22) 2015 年 10 月,Cloudera 公布继 HBase 以后的第一个 Hadoop 原生存储替代方案——Kudu。

(23) 2015 年 12 月,Cloudera 发起的 Impala 和 Kudu 项目加入 Apache 孵化器。

(24) 2017 年 12 月,Apache Hadoop 3.0.0 版本发布。

3.1.3 Hadoop 原理及运行机制

Hadoop 的核心由 3 个子项目组成: Hadoop Common、HDFS、和 MapReduce。其中,Hadoop Common 在 Hadoop 0.20 版本以前被称为 Hadoop Core。Hadoop Common 子项目为 Hadoop 整体构架提供基础支撑性功能。Hadoop Common 包括文件系统(File System)、远程过程调用协议(RPC)和数据串行化库(Serialization Libraries)。HDFS 是一个分布式文件系统,具有低成本、高可靠性、高吞吐量的特点。MapReduce 是一个计算模型,用于大数据量的计算。在实际应用环境中,这 3 个核心子项目配合默契,结合其他

子项目共同完成用户提交的大数据处理请求。下面主要讲述了 HDFS 和 MapReduce 这两个核心子项目所包含的主要逻辑组件。

1. HDFS 组件

HDFS(Hadoop Distributed FileSystem)是一种专门为 MapReduce 这类框架下的大规模分布式数据处理而设计的文件系统。可以把一个大数据集(100TB)在 HDFS 中存储为单个文件,大多数其他的文件系统无力实现这一点。

HDFS 的组件主要有 Namenode、SecondaryNamenode 及 Datanode。

1) Namenode

Namenode,即元数据节点。元数据节点用来管理文件系统的命名空间。它将所有文件和文件夹的元数据保存在一个文件系统树中。这些信息也会存储在 Namenode 维护的两个本地磁盘文件中,这两个本地磁盘文件是命名空间镜像文件(name space image)和编辑日志文件(editlog)。Namenode 还保存了一个文件包括哪些数据块、分布在哪些数据节点上这些信息。然而这些信息并不存储在硬盘上,而是在系统启动的时候从数据节点收集而来。

2) SecondaryNamenode

SecondaryNamenode,即从元数据节点。在 Hadoop 集群环境上,只有一个 Namenode 节点。那么,一旦 Namenode 节点出现故障,整个系统将会受到影响。为了提高 Namenode 的可靠性,从 Hadoop 0.23 版本开始引入了 Secondary Namenode。但是,SecondaryNamenode 并不是 Namenode 出现问题的时候的备用节点,它和 Namenode 负责不同的事情。SecondaryNamenode 的主要功能是周期性地将元数据节点命名空间的镜像文件和修改日志文件合并,以防日志文件过大。合并过后的命名空间镜像文件也在 SecondaryNamenode 中保存了一份,以便在元数据节点出现故障时,可以恢复数据。

3) Datanode

Datanode,即数据节点。Datanode 是文件系统中真正存储数据的地方,是 HDFS 文件系统中保存数据的节点。HDFS 中的文件通常被分割成多个数据块,以冗余备份的形式存储在多个 Datanode 中。客户端(client)或者元数据信息(Namenode)可以向数据节点请求写入或者读出数据块;而 Datanode 周期性地向 Namenode 汇报其存储的数据块信息。

2. MapReduce 组件

MapReduce 也采用了 Master/Slave(M/S)架构。它主要由 JobClient、JobTracker、TaskTracker 和 Task 组件组成。下面分别对这几个组件进行介绍。

1) JobClient

用户编写的 MapReduce 程序通过 JobClient 提交到 JobTracker 端;同时,用户可通过 Client 提供的一些接口查看作业运行状态。在 Hadoop 内部用“作业”(Job)表示 MapReduce 程序。一个 MapReduce 程序可对应若干个作业,而每个作业会被分解成若干个 Map/Reduce 任务(Task)。

2）JobTracker

JobTracker 主要负责 MapReduce 的资源监控和作业调度。JobTracker 监控所有 TaskTracker 与作业的状态情况，一旦发现失败情况后，其会将相应的任务转移到其他节点；同时，JobTracker 会跟踪任务的执行进度、资源使用量等信息，并将这些信息告诉任务调度器，而调度器会在资源出现空闲时，选择合适的任务使用这些资源。在 Hadoop 中，任务调度器是一个可插拔的模块，用户可以根据自己的需要设计相应的调度器。每一个 Hadoop 集群中只有一个 JobTracker。

3）TaskTracker

TaskTracker 主要负责执行由 JobTracker 分配的任务。TaskTracker 会周期性地通过 Heartbeat 将本节点上资源的使用情况和任务的运行进度汇报给 JobTracker，同时接收 JobTracker 发送过来的命令并执行相应的操作（如启动新任务、结束任务等）。

4）Task

Task 分为 MapTask 和 ReduceTask 两种，均由 TaskTracker 启动，负责具体地执行 Map 任务和 Reduce 任务的程序。

3.2 Hadoop 相关技术及生态系统

Hadoop 最为核心的技术是 HDFS 和 MapReduce。除此之外，为了满足大数据平台更高的存储和运算要求，Hadoop 技术不断拓展，在原来的基础上研发了很多其他技术，构成一个完整的分布式计算系统。Hadoop 生态系统主要包括 HDFS、MapReduce、Spark、Storm、HBase、Hive、Pig、ZooKeeper、Avro、Sqoop、Ambari、HCatalog、Chukwa、Flume、Tez、Phoenix、Mahout、Shark 等，Hadoop 生态系统如图 3-1 所示。

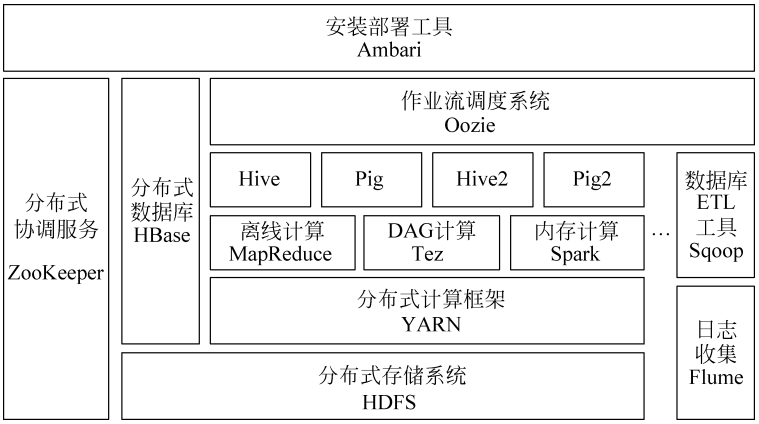

图 3-1 Hadoop 生态系统

（1）HDFS（Hadoop Distributed File System）：Hadoop 分布式文件系统，由早期的 NDFS 演化而来。它是一个高度容错的系统，能检测和应对硬件故障，用于在低成本的通用硬件上运行。并且，其简化了文件的一致性模型，通过流式数据访问，提供高吞吐量应用程序数据访问功能，适合带有大型数据集的应用程序。

(2) MapReduce：一个编程模型和软件框架，用于在大规模计算机集群上编写对大数据进行快速处理的并行化程序。

(3) Spark：一个开源的数据分析集群计算框架，最初由加州大学伯克利分校AMPLab开发，建立于HDFS之上。Spark与Hadoop一样，用于构建大规模、低延时的数据分析应用。Spark采用Scala语言实现，使用Scala作为应用框架，能够对大数据进行分析处理。

(4) Storm：一个分布式的、容错的实时计算系统，由BackType开发，后被Twitter收购。Storm属于流处理平台，多用于实时计算并更新数据库。Storm也可被用于“连续计算”(continuous computation)，对数据流做连续查询，在计算时就将结果以流的形式输出给用户。它还可用于“分布式RPC”，以并行的方式运行大型的运算。

(5) HBase：一个分布式和面向列的动态模式数据库，不同于一般的关系数据库，它是一个适合于非结构化大数据存储的数据库，支持随机、实时读/写访问。

(6) Hive：Hadoop中的一个重要子项目，最早由Facebook设计，是建立在Hadoop基础上的数据仓库架构，它提供了类似于SQL的查询语言，通过实现该语言，可以方便地进行数据汇总、特定查询，以及分析存放在Hadoop兼容文件系统中的大数据。

(7) Pig：运行在Hadoop上，是对大型数据集进行分析和评估的平台。Pig包括了一个数据分析语言和运行环境，其特点是其结构设计支持真正的并行化处理，因此适合应用于大数据处理环境。与Hive一样，Pig降低了对大型数据集进行分析和评估的门槛。

(8) Oozie：一个管理Hadoop作业、可伸缩、可扩展、可靠的工作流调度系统，它内部定义了三种作业：①工作流作业：由一系列动作构成的有向无环图(DAGs)；②协调器作业：按时间频率周期性触发Oozie工作流的作业；③Bundle作业：管理协调器作业。

(9) ZooKeeper：作为一个分布式的服务框架，解决了分布式计算中的一致性问题。在此基础上，ZooKeeper可用于处理分布式应用中经常遇到的一些数据管理问题，如统一命名服务、状态同步服务、集群管理、分布式应用配置项的管理等。

(10) Avro：由Doug Cutting牵头开发，是一个数据序列化系统。类似于其他序列化机制，Avro可以将数据结构或者对象转换成便于存储和传输的格式，其设计目标是用于支持数据密集型应用，适合大规模数据的存储与交换。

(11) Sqoop：SQL-to-Hadoop的缩写，是Hadoop的周边工具，它的主要作用是在结构化数据存储与Hadoop之间进行数据交换，即可以用于传统数据库(如SQL、Oracle)中的数据导入HDFS或者MapReduce，并将处理后的结果导出到传统数据库中。

(12) Ambari：一个用于安装、管理和监控Hadoop集群的Web界面工具，它提供一个直观的操作工具和一个健壮的Hadoop API，可以隐藏复杂的Hadoop操作，使集群操作大大简化。

(13) HCatalog：一个用于管理Hadoop产生的数据的表存储管理系统。它提供了一个共享的数据模板和数据类型的机制，并对数据表进行抽象，同时支持Hadooop不同数据处理工具之间的联系。

(14) Chukwa：开源的数据收集系统，用于监控大规模分布式系统。它构建在Hadoop的HDFS和MapReduce基础之上，继承了Hadoop的可伸缩性和鲁棒性。Chukwa

包含一个强大和灵活的工具集，提供了数据的生成、收集、排序、去重、分析和展示等一系列功能，是 Hadoop 使用者、集群运营人员和管理人员的必备工具。

(15) Flume：Flume 是 Cloudera 开发维护的分布式、可靠、高可用的日志收集系统。它将数据从产生、传输、处理并最终写入目标的路径的过程抽象为数据流，在具体的数据流中，数据源支持在 Flume 中定制数据发送方，从而支持收集各种不同协议的数据。

(16) Tez：一个基于 Hadoop YARN 之上的 DAG(有向无环图，Directed Acyclic Graph)计算框架。它把 Map/Reduce 过程拆分成若干个子过程，同时可以把多个 Map/Reduce 任务组合成一个较大的 DAG 任务，减少了 Map/Reduce 之间的文件存储。同时合理组合其子过程，减少任务的运行时间。

(17) Phoenix：一个构建在 Apache HBase 之上的 SQL 中间层，完全使用 Java 编写，提供 HBase scan，并编排执行以生成标准的 JDBC 结果集，直接使用 HBase API、协同处理器与自定义过滤器。对于简单查询来说，其性能量级是毫秒；对于百万级别的行数来说，其性能量级是秒。

(18) Mahout：一种基于 Hadoop 的机器学习和数据挖掘的分布式计算框架算法集，实现了多种 MapReduce 模式的数据挖掘算法。

(19) Shark：即 Hive on Spark，一个专为 Spark 打造的大规模数据仓库系统，兼容 Apache Hive，无须修改现有的数据或者查询，就可以用 100 倍的速度执行 Hive QL。Shark 支持 Hive 查询语言、元存储、序列化格式及自定义函数，与现有 Hive 部署无缝集成，是一个更快、更强大的替代方案。

3.3 操作实践：Hadoop 安装与配置


视频讲解

3.3.1 安装 JDK

由于 Hadoop 是由 Java 语言开发的，所以需要安装 JDK。安装 JDK 的具体步骤如下。

(1) 下载 JDK 安装包 jdk-8u141-linux-x64.tar.gz(本书将下载好的安装包放在 home/hadoop/software 目录下)。

(2) 卸载 Centos 自带的 OpenJDK(root 权限下)。

查看系统已有的 openjdk，如图 3-2 所示。

```
[root@master ~]# rpm -qa|grep jdk
```

图 3-2 查看 openjdk

卸载上述找到的 openjdk 包，如图 3-3 和图 3-4 所示。

```
[root@master ~]# yum -y remove java-1.8.0-openjdk-headless-1.8.0.102-4.b14.el7.x86_64
```

图 3-3 卸载 openjdk 包(一)

```
[root@master ~]# yum -y remove java-1.7.0-openjdk-headless-1.7.0.111-2.6.7.8.el7.x86_64
```

```
Removed:
  java-1.7.0-openjdk-headless.x86_64 1:1.7.0.111-2.6.7.8.el7
  libreoffice-langpack-en.x86_64 1:5.0.6.2-3.el7

Dependency Removed:
  java-1.7.0-openjdk.x86_64 1:1.7.0.111-2.6.7.8.el7
  jline.noarch 0:1.0-8.el7
  libreoffice-calc.x86_64 1:5.0.6.2-3.el7
  libreoffice-core.x86_64 1:5.0.6.2-3.el7
  libreoffice-draw.x86_64 1:5.0.6.2-3.el7
  libreoffice-graphicfilter.x86_64 1:5.0.6.2-3.el7
  libreoffice-impress.x86_64 1:5.0.6.2-3.el7
  libreoffice-pdfimport.x86_64 1:5.0.6.2-3.el7
  libreoffice-pyuno.x86_64 1:5.0.6.2-3.el7
  libreoffice-ure.x86_64 1:5.0.6.2-3.el7
  libreoffice-writer.x86_64 1:5.0.6.2-3.el7
  rhino.noarch 0:1.7R4-5.el7
  unoconv.noarch 0:0.6-7.el7

Complete!
```

图 3-4　卸载 openjdk 包(二)

这时,已有 Openjdk 卸载完了。(注:同样的操作在节点 slave1 和 slave2 上进行。)

(3) 传输文件,将文件夹 software 中的 jdk 安装包复制到文件夹/opt/java/里面。

```
[root@master ~]# cd /home/hadoop/software
[root@master software]# ls
hadoop-2.7.3-src.tar.gz jdk-8u141-linux-x64.tar.gz
[root@master software]# cp jdk-8u141-linux-x64.tar.gz/opt/java/
```

将/home/hadoo/下的安装包复制到/opt/目录下,如图 3-5 所示。

```
[root@master opt]# cp /home/hadoop/jdk-8u141-linux-x64.tar.gz /opt/
```

```
[root@master opt]# cp /home/hadoop/jdk-8u141-linux-x64.tar.gz /opt/
[root@master opt]# ls
jdk-8u141-linux-x64.tar.gz  rh
```

图 3-5　复制安装包

解压命令:

```
tar -zxvf 安装包名
[root@master opt]# tar -zxvf jdk-8u141-linux-x64.tar.gz
```

查看解压后的安装包:

```
[root@master opt]# ls
jdk1.8.0_141 jdk-8u141-linux-x64.tar.gz rh
```

同样的安装步骤在其余节点上进行,如图 3-6 和图 3-7 所示。

图 3-6 查看安装包(一)

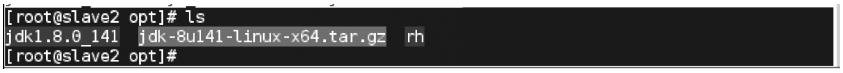

图 3-7 查看安装包(二)

进入目录下查看:

```
[root@master software]# cd/opt/java
[root@master java]# ls
jdk-8u141-linux-x64.tar.gz
[root@master java]# tar -zxvf jdk-8u141-linux-x64.tar.gz
```

安装完毕后,记录下 jdk 的路径。

(4) 将安装的 jdk 路径添加至系统环境变量中。

修改系统环境变量命令:

```
[root@master ~]# gedit /root/.bash_profile
```

在文件末尾加上如下内容,如图 3-8 所示。

```
export JAVA_HOME=/opt/java/jdk1.8.0_141/
export PATH=$JAVA_HOME/bin:$PATH
export PATH=$PATH:$JAVA_HOME/bin:$JRE_HOME/bin
export CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar
```

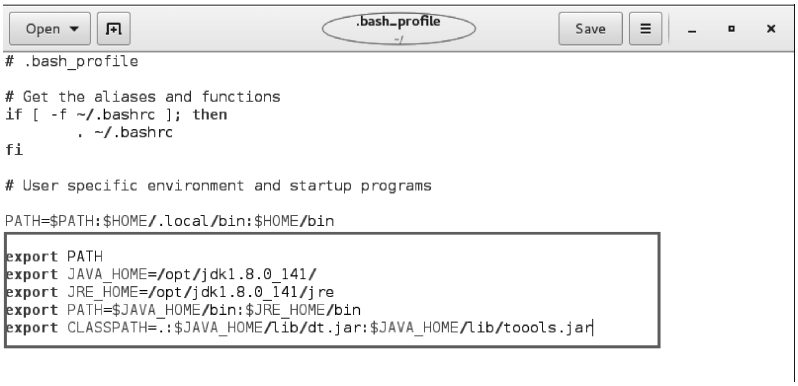

图 3-8 将安装后 jdk 路径添加至系统环境变量中

(5)关闭 profile 文件,执行下列命令使配置生效:

```
[root@master ~]# source /root/.bash_profile
```

此时，我们就可以通过 java -version 命令检查 jdk 路径是否配置成功。

Master，如图 3-9 所示。

```
root@master ~]# vi /root/.bash_profile
root@master ~]# source /root/.bash profile
root@master ~]# java -version
ava version "1.8.0_141"
ava(TM) SE Runtime Environment (build 1.8.0_141-b15)
ava HotSpot(TM) 64-Bit Server VM (build 25.141-b15, mixed mode)
root@master ~]#
```

图 3-9　Master

Slave1，如图 3-10 所示。

```
root@slave1 ~]# vi /root/.bash_profile
root@slave1 ~]# source /root/.bash profile
root@slave1 ~]# java -version
ava version "1.8.0_141"
ava(TM) SE Runtime Environment (build 1.8.0_141-b15)
ava HotSpot(TM) 64-Bit Server VM (build 25.141-b15, mixed mode)
root@master ~]#
```

图 3-10　Slave1

Slave2，如图 3-11 所示。

```
root@slave2 ~]# vi /root/.bash_profile
root@slave2 ~]# source /root/.bash profile
root@slave2 ~]# java -version
ava version "1.8.0_141"
ava(TM) SE Runtime Environment (build 1.8.0_141-b15)
ava HotSpot(TM) 64-Bit Server VM (build 25.141-b15, mixed mode)
root@master ~]#
```

图 3-11　Slave2

至此，所有节点的 jdk 1.8 安装完毕。

3.3.2　安装 Hadoop

Hadoop2.7.3 的安装步骤如下。

(1) 下载 Hadoop-2.7.3 安装包 hadoop-2.7.3.tar.gz。

(2) 在安装包所在目录下解压文件，如图 3-12 所示。

```
[root@master software]# tar -zxvf hadoop-2.7.3.tar.gz
#复制解压后的 hadoop-2.7.3 文件到/opt/目录下
[root@master ~]# cp -r /home/hadoop/hadoop-2.7.3/opt/
```

```
[root@master ~]# cd /opt/
[root@master ~]# ls -l h*
total 4
drwxr-xr-x. 16   root  root   4096 Jul 27 01:16 hadoop-2.7.3
[root@master ~]#
```

图 3-12　安装 Hadoop

3.3.3　配置 Hadoop

配置 Hadoop 的具体步骤如下。

(1) 在 master 节点上,首先在/home/hadoop/目录下创建 4 个新文件夹,如图 3-13 所示。

```
root@master ~]# mkdir -p /home/hadoop/hadoopdir/data
root@master ~]# mkdir -p /home/hadoop/hadoopdir/temp
root@master ~]# mkdir -p /home/hadoop/hadoopdir/logs
root@master ~]# mkdir -p /home/hadoop/hadoopdir/pids
root@master ~]# cd /home/hadoop/hadoopdir/
root@master hadoopdir]$ ls
data  logs  pids  temp
```

图 3-13 创建新文件夹

(2) 同样在 slave1、slave2 节点上创建 4 个新文件夹,如图 3-14 所示。

```
[root@slave1 ~]# mkdir -p /home/hadoop/hadoopdir/data
[root@slave1 ~]# mkdir -p /home/hadoop/hadoopdir/temp
[root@slave1 ~]# mkdir -p /home/hadoop/hadoopdir/logs
[root@slave1 ~]# mkdir -p /home/hadoop/hadoopdir/pids
[root@slave1 ~]# cd /home/hadoop/hadoopdir/
[root@slave1 hadoopdir]$ ls
data  logs  pids  temp
```

图 3-14 在 Slave1、Slave2 节点上创建新文件夹

进入到解压后的 hadoop-2.7.3/etc/hadoop 目录下,开始配置文件:

① hadoop-env.sh:

```
export JAVA_HOME = /opt/jdk1.8.0_141/
export HADOOP_LOG_DIR = /home/hadoop/hadoopdir/logs
export HADOOP_PID_DIR = /home/hadoop/hadoopdir/pids
```

② Mapred-env.sh:

添加下列语句:

```
export JAVA_HOME = /opt/ jdk1.8.0_141/
export HADOOP_MAPRED_LOG_DIR = /home/hadoop/hadoopdir/logs
export HADOOP_MAPRED_PID_DIR = /home/hadoop/hadoopdir/pids
```

③ yarn-env.sh:

```
export JAVA_HOME = /opt/ jdk1.8.0_141/
YARN_LOG_DIR = /home/hadoop/hadoopdir/logs
```

④ Slaves 文件,如图 3-15 所示。

图 3-15 配置 Slaves 文件

⑤ Core-site. xml：

```
<configuration>
    <property>
        <name>fs.defaultFS</name>
        <value>hdfs://master:9000</value>
    </property>
    <property>
        <name>io.file.buffer.size</name>
        <value>131072</value>
    </property>
    <property>
        <name>hadoop.tmp.dir</name>
        <value>file:///home/hadoop/hadoopdir/temp</value>
    </property>
</configuration>
```

⑥ Hdfs-site. xml：

```
<configuration>
    <property>
        <name>dfs.namenode.name.dir</name>
        <value>file:///home/hadoop/hadoopdir/name</value>
    </property>
    <property>
        <name>dfs.datanode.data.dir</name>
        <value>file:///home/hadoop/hadoopdir/data</value>
    </property>
    <property>
        <name>dfs.replication</name>
        <value>2</value>
    </property>
    <property>
        <name>dfs.blocksize</name>
        <value>64m</value>
    </property>
    <property>
        <name>dfs.namenode.secondary.http-address</name>
        <value>master:9001</value>
    </property>
    <property>
        <name>dfs.webhdfs.enabled</name>
        <value>true</value>
    </property>
</configuration>
```

⑦ Mapred-site. xml。

首先执行复制命令：

```
[root@master hadoop]# cp /opt/hadoop-2.7.3/etc/hadoop/mapred-site.xml.template /opt/hadoop-2.7.3/etc/hadoop/mapred-site.xml
```

然后打开文件进行添加：

```
<configuration>
    <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
        <final>true</final>
    </property>
    <property>
        <name>mapreduce.jobhistory.address</name>
        <value>master:10020</value>
    </property>
    <property>
        <name>mapreduce.jobtracker.http.address</name>
        <value>master:50030</value>
    </property>
    <property>
        <name>mapred.job.tracker</name>
        <value>http://master:9001</value>
    </property>
    <property>
        <name>mapreduce.jobhistory.webapp.address</name>
        <value>master:19888</value>
    </property>
</configuration>
```

⑧ Yarn-site. xml：

```
<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>
<property>
<name>yarn.resourcemanager.hostname</name>
<value>master</value>
</property>
<property>
<name>yarn.resourcemanager.scheduler.address</name>
<value>master:8030</value>
</property>
```

```
<property>
<name>yarn.resourcemanager.resource-tracker.address</name>
<value>master:8031</value>
</property>
<property>
<name>yarn.resourcemanager.address</name>
<value>master:8032</value>
</property>
<property>
<name>yarn.resourcemanager.admin.address</name>
<value>master:8033</value>
</property>
<property>
<name>yarn.resourcemanager.webapp.address</name>
<value>master:8088</value>
</property>
```

至此，所有配置文件已完成。

（3）在 master 机器上，把整个配置好的 hadoop2.7.3 文件夹复制到其他节点。

注意：在这里，为了统一，把 master 根目录下/opt/中的 hadoop-2.7.3 复制到/home/hadoop/目录下。

```
[root@master~]# scp -r /opt/hadoop-2.7.3 hadoop@master:/home/hadoop/
[root@master~]# scp -r /opt/hadoop-2.7.3 hadoop@slave1:/home/hadoop/
[root@master~]# scp -r /opt/hadoop-2.7.3 hadoop@slave2:/home/hadoop/
```

所以，现在所有节点配置好的文件都在/home/hadoop/目录下了，如图 3-16 所示。

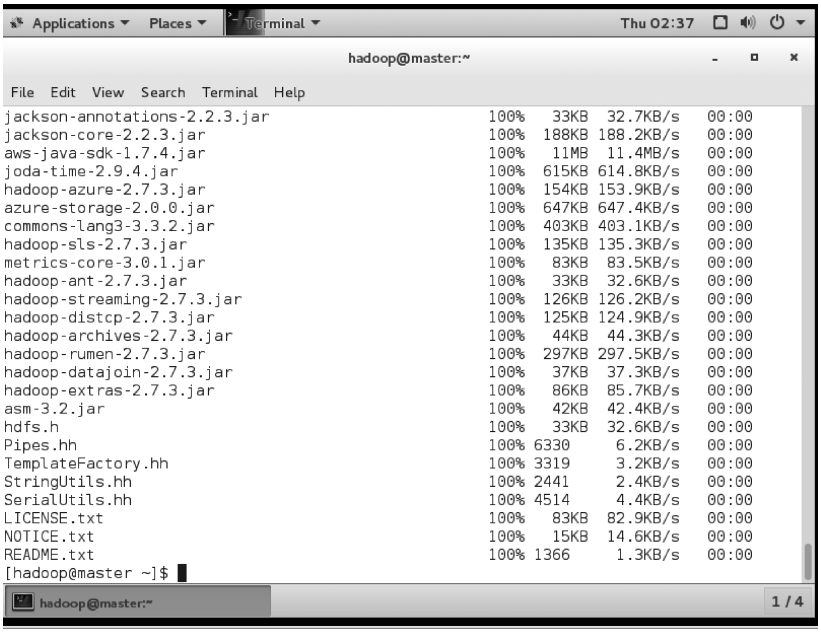

图 3-16　配置文件完成

3.3.4 格式化

进入 hadoop-2.7.3/bin 目录,进行格式化,如图 3-17 和图 3-18 所示。

```
[root@master ~]# cd /home/hadoop/hadoop-2.7.3/bin
[root@master bin]# ./hdfs namenode -format
```

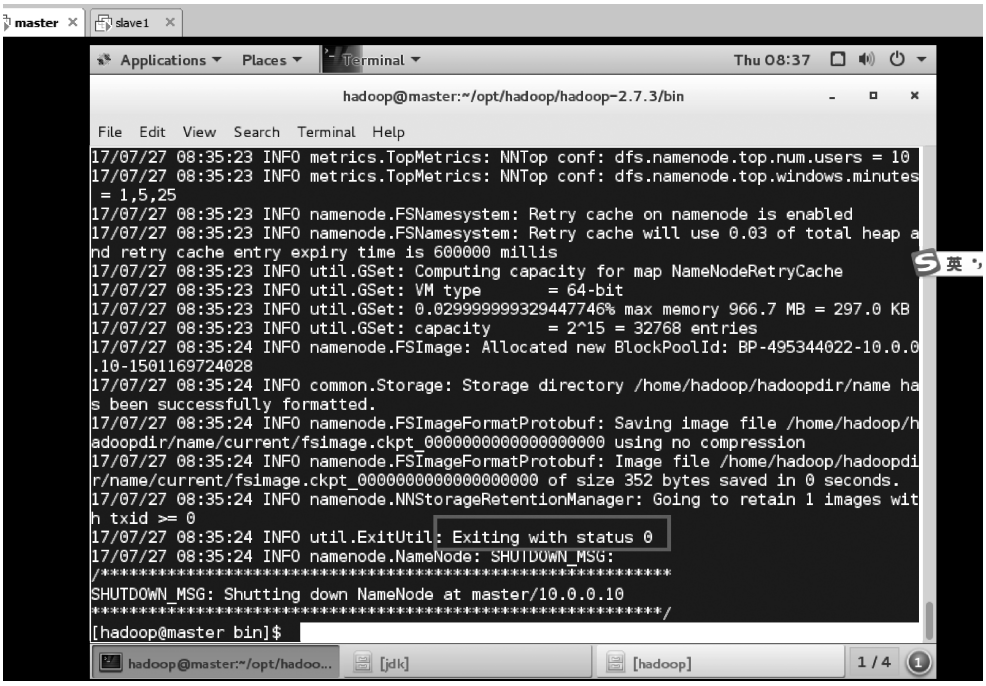

图 3-17 格式化(一)

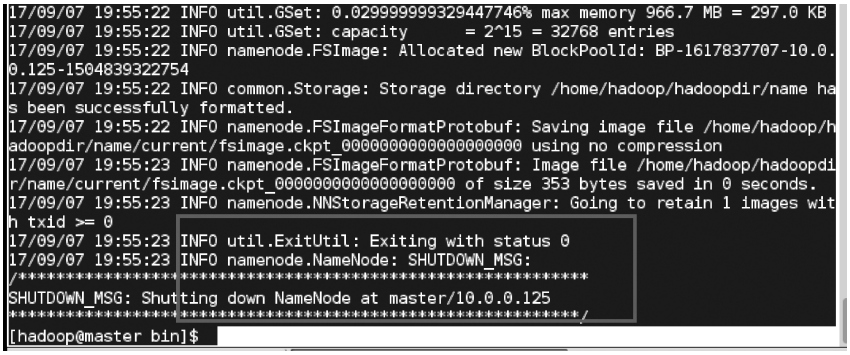

图 3-18 格式化(二)

3.3.5 运行 Hadoop

进入 cd/home/hadoop/hadoop-2.7.3/sbin 目录下。

（1）启动 hdfs 集群：

```
[root@master sbin]#./start-dfs.sh
```

（2）启动 yarn 集群：

```
[root@master sbin]# ./start-yarn.sh
```

这样，hadoop 集群就跑起来了。如果要关闭，在 sbin 目录下进行：

```
[root@master sbin]#./stop-dfs.sh
[root@master sbin]#./stop-yarn.sh
```

（3）使用命令 jps 查看节点：

Master，如图 3-19 所示。

图 3-19　Master

Slave1，如图 3-20 所示。

图 3-20　Slave1

Slave2，如图 3-21 所示。

图 3-21　Slave2

如果要关闭，在 sbin 目录下，执行以下命令：

```
[root@master sbin]# ./stop-dfs.sh
[root@master sbin]# ./stop-yarn.sh
```

然后需要用浏览器打开 http://master:50070，即 HDFS 的 Web 页面，可以看到集群信息和 Datenode 相关信息，如图 3-22 所示。

用浏览器打开 http://master:8088，即 YARN 的 Web 页面，可以看到集群相关信息，如图 3-23 所示。

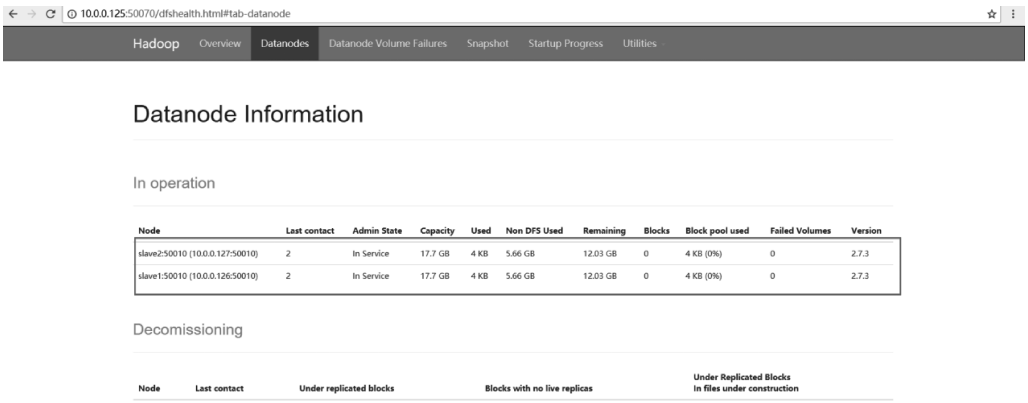

图 3-22　打开 HDFS 的 Web 页面

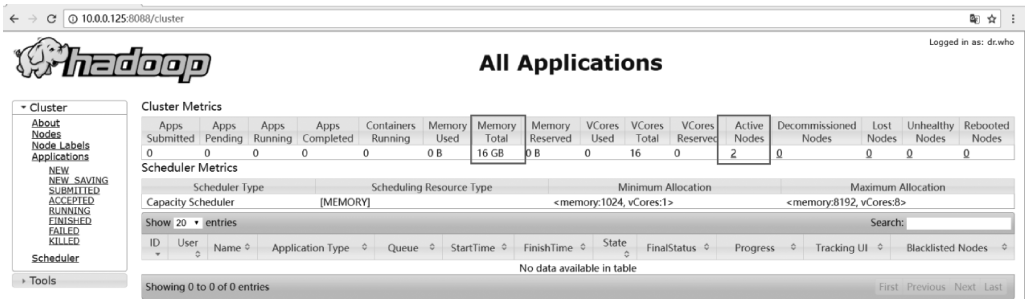

图　3-23

小结

本章首先介绍了 Hadoop 海量数据分布式处理框架、Hadoop 的优点及 Hadoop 的发展历程，然后详细描述了 Hadoop 原理、运行机制，以及 3 个重要的组件（Hadoop Common、HDFS 和 MapReduce），接着简述了 Hadoop 开源技术生态系统的相关组件，最后重点介绍了 Hadoop 安装与配置，包括 JDK 的安装与配置、Hadoop 的安装与配置、Hadoop 的运行。

习题

1. 简述 Hadoop 系统及其优点。
2. 简述 Hadoop 原理及运行机制。
3. 简述 Hadoop 技术生态系统。
4. 学会 JDK 的安装和配置。
5. 掌握 Hadoop 的安装和配置。