第2章

Cloudera 大数据平台介绍

由于 Hadoop 深受客户欢迎，因此许多公司都推出了各自版本的 Hadoop，也有一些公司围绕 Hadoop 开发产品。在 Hadoop 生态系统中，规模最大、知名度最高的公司是 Cloudera。2008 年成立的 Cloudera 是最早将 Hadoop 商用的公司，为合作伙伴提供 Hadoop 的商用解决方案。Cloudera 企业解决方案包括 Cloudera Hadoop 发行版（Cloudera Distribution Hadoop，CDH）、Cloudera Manager（CM）等。概括起来说，Cloudera 提供一个可伸缩的、稳定的、综合的企业级大数据管理平台，它拥有最多的部署案例，提供强大的部署、管理和监控工具。

2.1 Cloudera 简介

众所周知，Hadoop 是一个开源项目，所以很多公司在这个基础上进行商业化，在 Hadoop 生态系统中，规模最大、知名度最高的公司则是 Cloudera，目前 Intel 已经成为 Cloudera 最大的战略股东。Cloudera 的客户有很多知名公司，如哥伦比亚广播公司、eBay、摩根大通、迪士尼等。

Cloudera 提供一个可扩展的、灵活的、集成的企业级大数据管理平台，可用来方便地管理你的企业中快速增长的多种多样的数据。业界领先的 Cloudera 产品和解决方案使你能够部署并管理 Apache Hadoop 及其相关项目、操作和分析你的数据，以及保护数据的安全。

Cloudera 提供下列产品和工具：

（1）CDH：Cloudera 分发的 Apache Hadoop 和其他相关开放源代码项目，包括 Impala 和 Cloudera Search。CDH 还提供安全保护以及与许多硬件和软件解决方案的集成。

（2）Cloudera Impala：一种 MPP（大规模并行处理） SQL 引擎，用于交互式查询分析。它非常适合用于具有连接、聚合和子查询的传统 BI 商业智能的查询。它可以查询来自各种源的 Hadoop 数据文件，包括由 MapReduce 作业生成的数据文件或加载到 Hive 表中的数据文件。你可以通过 Cloudera Manager 用户界面管理 Impala 及其他 Hadoop 组件，并通过 Sentry 授权框架

保护其数据。

（3）Cloudera Search：提供近实时访问已存储的数据，或者摄取数据到 Hadoop 以及 HBase 中去。Search 提供了近实时的索引、批量索引、全文检索和 Drill-Down（下钻）的导航，以及一个简单的全文检索的接口，只需要一些 NoSQL 或者编程基础（技能）即可使用。完全集成的数据处理平台 Search 使用了在 CDH 中灵活的、可扩展的、可靠的存储系统。这样就不再需要在基础设施层或者业务层移动大量的数据了，也不需要产生新的任务。

（4）Cloudera Manager：一个复用于部署、管理和监控 CDH 大数据平台的应用程序。Cloudera Manager 提供 Admin Console，这是一种基于 Web 的用户界面，使得企业数据管理更加容易方便。Cloudera Manager 易于升级和安装 Hadoop 组件，还提供了在几分钟之内建立集群主节点的高可用（High Availability）。它还包括 Cloudera Manager API，可用来获取群集运行状态信息以及配置 Cloudera Manager。

（5）Cloudera Navigator：定位为 Hadoop 提供数据管理和监管的工具，它简化了存储和密钥的管理。Cloudera Navigator 中强大的数据审计和数据保护使企业能够满足严格的规范限制并遵从相关法规。

2.2　Cloudera 的 Hadoop 发行版 CDH 简介

2.2.1　CDH 概述

Cloudera 提供了 Hadoop 的商业发行版 CDH，能够十分方便地对 Hadoop 集群进行安装、部署和管理。如图 2-1 所示，CDH 是 Cloudera 发布的一个自己封装的 Hadoop 商业版软件发行包，里面不仅包含了 Cloudera 的商业版 Hadoop，同时 CDH 中也包含了各类常用的开源数据处理与存储框架，如 Spark、Hive、HBase 等。

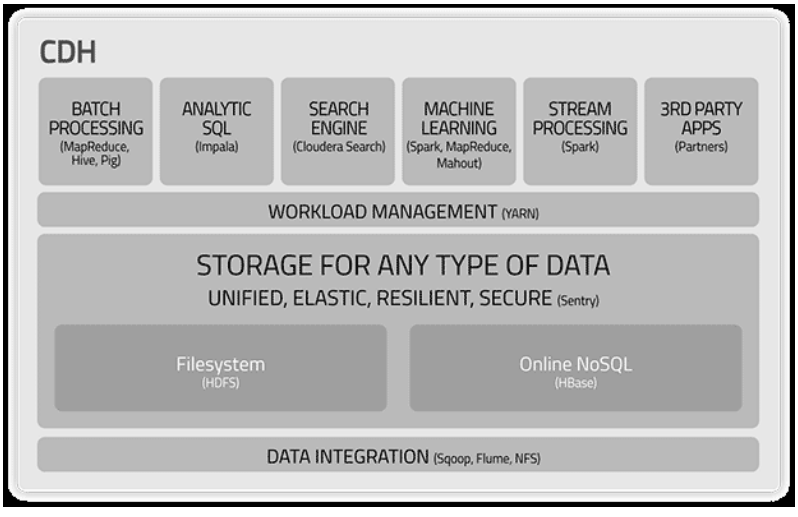


图 2-1

部署 Hadoop 群集的时候，可以选择 Cloudera Express 免费版本。这个版本包含了 CDH 以及 Cloudera Manager 核心功能，提供了对集群的管理功能，比如自动化部署、中心化管理、监控、诊断功能等。另外，Cloudera Express 免费版本对群集节点数目是无限制的。收费的 Cloudera Enterprise 拥有高级管理功能，如提供商业技术支持、自动化备份和灾难恢复、记录配置历史及回滚等，而这些功能 Cloudera Express 则没有。

2.2.2 CDH 和 Apache Hadoop 对比

Hadoop 大致可分为 Apache Hadoop 和第三方发行版 Hadoop。考虑到 Hadoop 集群部署的高效性、集群的稳定性以及后期集中的配置管理，业界大多使用 Cloudera 公司的发行版 CDH。

Apache Hadoop 社区版本虽然完全开源免费，但是也存在诸多问题：

（1）版本管理比较混乱，让人有些无所适从。

（2）集群部署配置较为复杂，通常按照集群需要编写大量的配置文件，分发到每一台节点上，容易出错，效率低下。

（3）对集群的监控、运维，需要安装第三方的其他软件，运维难度较大。

（4）在 Hadoop 生态圈中，组件的选择和使用，比如 Hive、Mahout、Sqoop、Flume、Spark、Oozie 等，需要大量考虑兼容性的问题，经常会浪费大量的时间去编译组件，解决版本冲突问题。

CDH 版本的 Hadoop 的优势在于：

（1）基于 Apache 协议，100%开源，版本管理清晰。

（2）在兼容性、安全性、稳定性上比 Apache Hadoop 有大幅度的增强。

（3）运维简单方便，对于 Hadoop 集群提供管理、诊断、监控、配置更改等功能，使得运维工作非常高效，而且群集节点越多，优势越明显。

（4）CDH 提供成体系的文档、很多大公司的应用案例以及商业支持等。

2.3 Cloudera Manager 大数据管理平台介绍

2.3.1 Cloudera Manager 概述和整体架构

Cloudera Manager（简称 CM）是为了便于在集群中进行 Hadoop 等大数据处理相关的服务安装和监控管理的组件，对集群中主机、Hadoop、Hive、Spark 等服务的安装配置管理做了极大简化。它是 Hadoop 集群的软件分发及管理监控平台，通过它可以快速地部署好一个 Hadoop 集群，并对集群的节点及服务进行实时监控。

Cloudera Manager 的整体架构如图 2-2 所示。

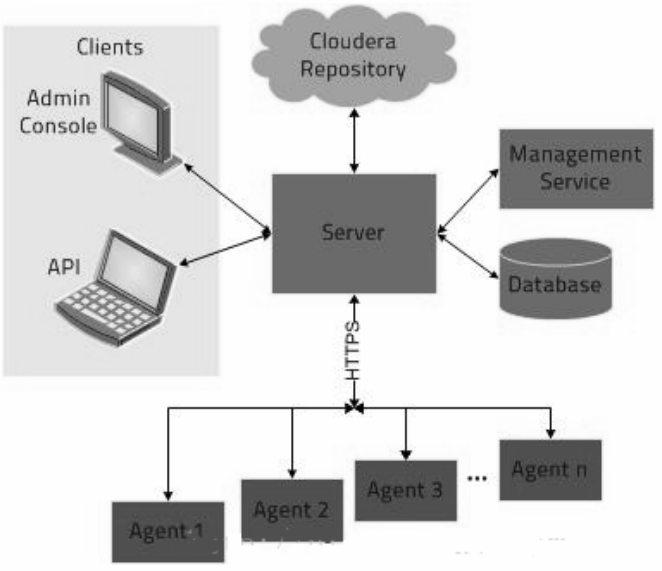


图 2-2

Cloudera Manager 的核心是 Cloudera Manager Server，它包括以下组件。

- Server：托管 Admin Console Web Server 和应用程序逻辑。它负责安装软件、配置、启动和停止服务以及管理运行服务的群集。
- Agent：安装在每台主机上。它负责启动和停止进程，解压缩配置，触发安装和监控主机。默认情况下，Agent 每隔 15 秒向 Cloudera Manager Server 发送一次检测信号。但是，为了减少用户延迟，在状态变化时会提高频率。如果 Agent 停止检测信号，主机将被标记为运行状况不良。
- Management Service：执行各种监控、报警和报告功能的一组角色的服务。
- Database：存储配置和监控信息。
- Cloudera Repository：可供 Cloudera Manager 分配的软件的存储库（repo 库）。
- Client：用于与服务器进行交互的接口。
- Admin Console：管理员控制台。
- API：Cloudera 产品具有开发的特性，所有在 Cloudera Manager 界面上提供的功能，通过 API 都可以完成同样的工作，这些 API 都是标准的 REST API。开发人员使用 API 甚至可以创建自定义的 Cloudera Manager 应用程序。

Cloudera Management Service 可作为一组角色实施各种管理功能：

- Activity Monitor：收集有关服务运行活动的信息。
- Host Monitor：收集有关主机的运行状况和指标信息。
- Service Monitor：收集有关服务的运行状况和指标信息。
- Event Server：聚合组件的事件并将其用于警报和搜索。
- Alert Publisher ：为特定类型的事件生成和提供警报。
- Reports Manager：生成图表报告，提供用户、用户组的目录的磁盘使用率、磁盘 IO 等历史视图。

2.3.2 Cloudera Manager 的基本核心功能

Cloudera Manager 作为 Hadoop 大数据平台的管理工具，能够有效地帮助用户更容易地使用 Hadoop。它的基本核心功能分为四大模块：管理功能、监控功能、诊断功能和集成功能。

Cloudera Manager 提供的管理功能如下：

（1）批量自动化部署节点：CM 提供强大的 Hadoop 集群部署能力，能够批量地自动化部署节点。安装一个 Hadoop 集群只需添加需要安装的节点、安装需要的组件和分配角色这三步，大大缩短了 Hadoop 的安装时间，也简化了 Hadoop 的安装过程。

（2）可视化的参数配置功能：Hadoop 包含许多组件，不同组件都包含各种各样的 XML 配置文件。CM 提供界面 GUI 可视化参数配置功能，如图 2-3 所示，能自动部署到每个节点。

图 2-3

（3）智能参数验证以及优化：当用户配置部分参数值有问题时，CM 会给出智能错误提示，帮助用户更合理地修改配置参数，如图 2-4 所示。

图 2-4

（4）高可用配置：CM 对关键的组件使用 HA 部署，如 NameNode 高可用可以通过 CM 的 Web 管理界面，根据向导启用 HDFS HA，如图 2-5 所示。

图 2-5

（5）权限管理：提供不同级别的管理权限，比如只读用户访问 Cloudera Manager 的界面时，所有服务对应的启停等操作选项都不可用，如图 2-6 所示。

图 2-6

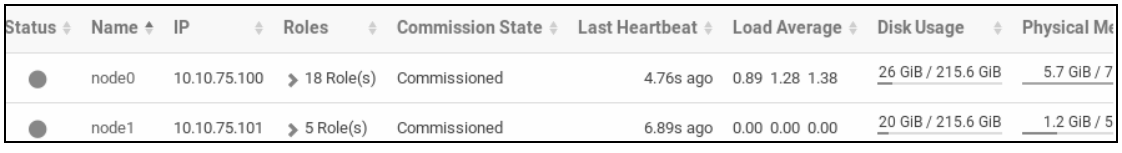
Cloudera Manager 提供的监控功能如下：

（1）服务监控：查看服务和实例级别健康检查的结果，对设置的各种指标和系统运行情况进行全面监控，如图 2-7 所示。如果任何运行状况测试是不良（Bad），则服务或者角色的状态就是不良（Bad）。如果任何运行状况测试是存在隐患（Concerning，没有任何一项是不良（Bad）），则角色或者服务的状况就是存在隐患（Concerning），而且系统会对管理员应该采取的行动提出建议，如图 2-8 所示。

图 2-7

图 2-8

（2）主机监控：监控群集内所有主机的有关信息，包括主机上目前消耗的内存、主机上运行的角色分配等，如图 2-9 所示，不但显示所有群集主机的汇总视图，而且能进一步显示单个主机关键指标详细视图。



Status	Name	IP	Roles	Commission State	Last Heartbeat	Load Average	Disk Usage	Physical Me
●	node0	10.10.75.100	18 Role(s)	Commissioned	4.76s ago	0.89 1.28 1.38	26 GiB / 215.6 GiB	5.7 GiB / 7
●	node1	10.10.75.101	5 Role(s)	Commissioned	6.89s ago	0.00 0.00 0.00	20 GiB / 215.6 GiB	1.2 GiB / 5

图 2-9

（3）行为监控：CM 提供了列表和图表来查看群集上进行的活动，不仅显示当前正在执行的任务行为，还可以通过仪表盘查看历史活动。

（4）事件活动：监控界面可以查看事件，系统管理员可以通过时间范围、服务、主机、关键字等字段信息过滤事件。

（5）报警：通过配置 CM 可以对指定的事件产生警报，并通过电子邮件或者 SNMP 的事件得到制定的警报通知。

（6）日志和报告：可以轻松点击一个链接查看相关的特定服务的日志条目，并且 Cloudera Manager 可以将收集到的历史监控数据统计生成报表。

Cloudera Manager 提供的诊断功能如下：

（1）周期性服务诊断：CM 会对集群中运行的服务进行周期性的运行状况测试，以检测这些服务的状态是否正常。如果有异常情况，就会进行告警，有利于更早地让用户感知集群服务存在的问题，如图 2-10 所示。

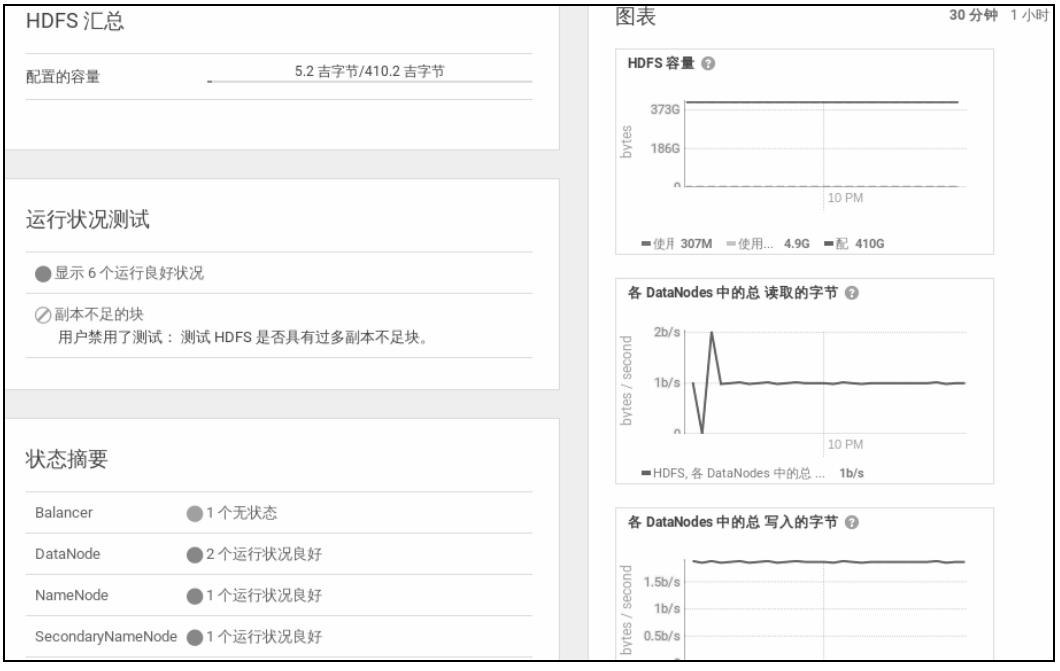


图 2-10

（2）日志采集及检索：对于一个大规模的集群，CM 提供了日志的收集功能，能够通过统一的界面查看集群中每台机器、各项服务的日志，并且能够根据日志级别等不同的条件进行检索，如图 2-11 所示。

（3）系统性能使用报告： CM 能够产生系统性能使用报告，包括集群的 CPU 使用率、单节点的 CPU 使用率、单个进程的 CPU 使用率等各项性能数据，这对于 Hadoop 集群的性能调试很重要。

图 2-11

Cloudera Manager 提供的集成功能如下：

（1）安全配置：为了方便 Hadoop 大数据平台与原有身份认证系统如 AD、LDAP 等的集成，CM 只需在界面上配置即可完成。

（2）Cloudera Manager API：通过 Cloudera Manager API，能够方便地将 CM 集成到企业原有管理系统集成。

（3）SNMP 集成：CM 也提供了方便的 SNMP 集成能力，只要简单的配置，就能够将 SNMP 进行集成，并且将集群中的告警信息进行转发。

2.3.3 Cloudera Manager 的高级功能

Cloudera manager 的高级功能在免费的 Express 版本中是不提供的。

（1）软件滚动升级：Hadoop 版本升级和 bug 修复，通常会影响业务的连续性。CM 提供了滚动升级的功能，支持 Hadoop 平台进行升级时继续对外提供服务以及应用。

（2）参数版本控制：任何时候进行配置修改并保存之后，Cloudera Manager 会对该配置生成一个版本。Cloudera Manager 支持查看历史配置，并能回滚到不同版本，从而为集群恢复、问题诊断等提供了可靠的依据和方便的工具。

（3）备份及容灾系统 BDR：Cloudera 为 Hadoop 平台提供了一个集成的、易用的灾备解决方案。BDR 为灾备方案提供了丰富的功能，CM 为 BDR 提供了完整的用户界面，实现界面化的数据备份与灾难恢复。

（4）数据审计：Cloudera Navigator 的审计功能支持对于数据的审计和访问。

（5）安全集成向导：启用 Kerberos 集成和外部安全认证集成，如支持通过内部数据库和外部服务进行用户认证。

2.4　Cloudera 平台参考部署架构

按照 Cloudera 官方手册，这里归纳总结出 Cloudera 大数据平台参考部署的架构指导。

2.4.1　Cloudera 的软件体系结构

Cloudera 的软件体系结构中包含系统部署和管理，数据存储，资源管理，处理引擎，安全，数据管理，工具仓库以及访问接口模块。一些关键组件的角色信息如表 2-1 所示。

表 2-1　Cloudera 一些关键组件的角色信息

模块	组件	管理角色	工作角色
系统部署和管理	Cloudera Manager	Cloudera Manager server	Cloudera Manager Agent
		Host monitor	
		Service monitor	
		Reports manager	
		Event server	
数据存储	HDFS	NameNode	DataNode
		SecondaryNameNode	
		JournalNode	
		FailoberController	
	HBase	HBase Master	RegionServer
资源管理	YARN	ResourceManager	NodeManager
		Job HistoryServer	
处理引擎	Spark	History Server	
	Impala	Impala Catalog Server	Impala Daemon
		Impala StateStore	
	Search		Solr Server
安全、数据管理	Sentry	Sentry Server	
	Cloudera navigator	Navigator keyTrustee	
		Navigator Metadata Server	
		Nagivator Audit Server	
数据仓库	Hive	Hive Metastore	
		Hive Server2	

2.4.2　群集硬件规划配置

集群服务器按照节点承担的任务分为管理节点和工作节点。管理节点上一般部署各组件的管理角色，工作节点一般部署有各角色的存储、容器或计算角色。根据业务类型不同，集群具体配置

也有所区别。

（1）实时流处理服务集群：由于性能的原因，Hadoop 实时流处理对节点内存和 CPU 有较高要求，基于 Spark Streaming 的流处理消息吞吐量可随着节点数量增加而线性增长，配置可参考图 2-12。

	管理节点	工作节点
处理器	两路 Intel®至强处理器，可选用 E5-2630 处理器	两路 Intel®至强处理器，可选用 E5-2660 处理器
内核数	6 核/CPU（或者可选用 8 核/CPU），主频 2.3GHz 或以上	6 核/CPU（或者可选用 8 核/CPU），主频 2.0GHz 或以上
内存	128GB ECC DDR3	128GB ECC DDR3
硬盘	2 个 2TB 的 SAS 硬盘（3.5 寸），7200RPM，RAID1	4-12 个 4TB 的 SAS 硬盘（3.5 寸），7200RPM，不使用 RAID
网络	至少两个 1GbE 以太网电口，推荐使用光口提高性能。 可以两个网口链路聚合提供更高带宽。	至少两个 1GbE 以太网电口，推荐使用光口提高性能。 可以两个网口链路聚合提供更高带宽。
硬件尺寸	1U 或 2U	1U 或 2U
接入交换机	48 口千兆交换机，要求全千兆，可堆叠	
聚合交换机（可选）	4 口 SFP+万兆光纤核心交换机，一般用于 50 节点以上大规模集群	

图 2-12

（2）在线分析业务集群：在线分析业务一般基于 Impala 等 MPP SQL 引擎，复杂的 SQL 计算对内存容量有较高要求，因此需要配置 128GB 甚至更多的内存。硬件参考规划如图 2-13 所示。

	管理节点	工作节点
处理器	两路 Intel®至强处理器，可选用 E5-2630 处理器	两路 Intel®至强处理器，可选用 E5-2650 处理器
内核数	6 核/CPU（或者可选用 8 核/CPU），主频 2.3GHz 或以上	6 核/CPU（或者可选用 8 核/CPU），主频 2.0GHz 或以上
内存	128GB ECC DDR3	128GB -256GB ECC DDR3
硬盘	2 个 2TB 的 SAS 硬盘（3.5 寸），7200RPM，RAID1	12 个 4TB 的 SAS 硬盘（3.5 寸），7200RPM，不使用 RAID
网络	至少两个 1GbE 以太网电口，推荐使用光口提高性能。 可以两个网口链路聚合提供更高带宽。	至少两个 1GbE 以太网电口，推荐使用光口提高性能。 可以两个网口链路聚合提供更高带宽。
硬件尺寸	1U 或 2U	2U
接入交换机	48 口千兆交换机，要求全千兆，可堆叠	
聚合交换机（可选）	4 口 SFP+万兆光纤核心交换机，一般用于 50 节点以上大规模集群	

图 2-13

（3）云存储业务集群：云存储业务主要面向海量数据和文件的存储和计算，强调单节点存储容量和成本，因此配置相对廉价的 SATA 硬盘，满足成本和容量需求。硬件规划配置如图 2-14 所示。

	管理节点	工作节点
处理器	两路 Intel®至强处理器，可选用 E5-2630 处理器	两路 Intel®至强处理器，可选用 E5-2660 处理器
内核数	6 核/CPU（或者可选用 8 核/CPU），主频 2.3GHz 或以上	6 核/CPU（或者可选用 8 核/CPU），主频 2.0GHz 或以上
内存	128GB ECC DDR3	48GB ECC DDR3
硬盘	2 个 2TB 的 SAS 硬盘（3.5 寸），7200RPM，RAID1	12-16 个 6TB 的 SATA 硬盘（3.5 寸），7200RPM，不使用 RAID
网络	至少两个 1GbE 以太网电口，推荐使用光口提高性能。 可以两个网口链路聚合提供更高带宽。	至少两个 1GbE 以太网电口，推荐使用光口提高性能。 可以两个网口链路聚合提供更高带宽。
硬件尺寸	1U 或 2U	2U 或 3U
接入交换机	48 口千兆交换机，要求全千兆，可堆叠	
聚合交换机	4 口 SFP+万兆光纤核心交换机，一般用于 50 节点以上大规模集群	

图 2-14

2.4.3　Hadoop 集群角色分配

Hadoop 大数据平台集群角色简称如图 2-15 所示，请读者务必熟悉这些简称。

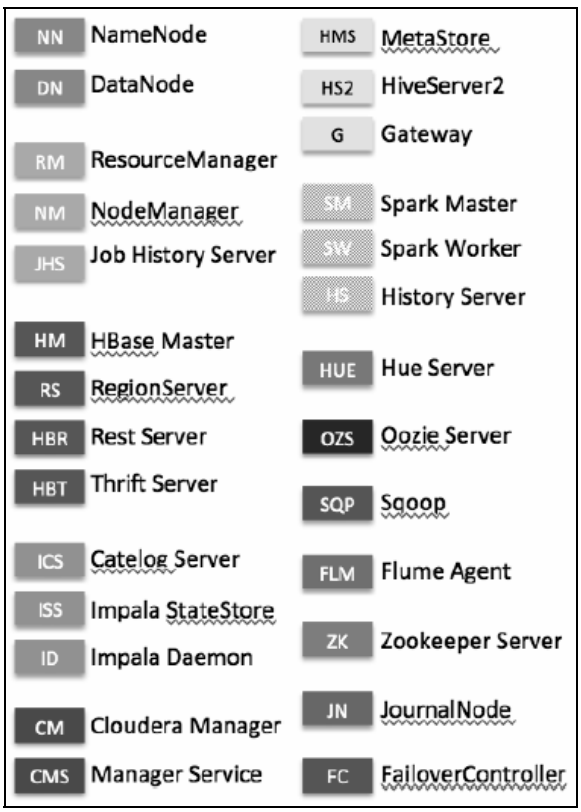


图 2-15

（1）搭建小规模集群一般是为了支撑专有业务，受限于集群的存储和处理能力，不太适合用

于多业务的环境。可以部署成一个 HBase 的集群，也可以部署成一个分析集群，包含 YARN、Impala。在小规模集群中，为了最大化利用集群的存储和处理能力，节点的复用程度往往比较高，如图 2-16 所示。对于那些需要两个以上节点来支持 HA 功能的，集群中分配有一个工具节点可以承载这些角色，并可以同时部署一些其他工具角色（这些工具角色本身消耗不了多少资源），其余节点可以部署为纯工作节点。

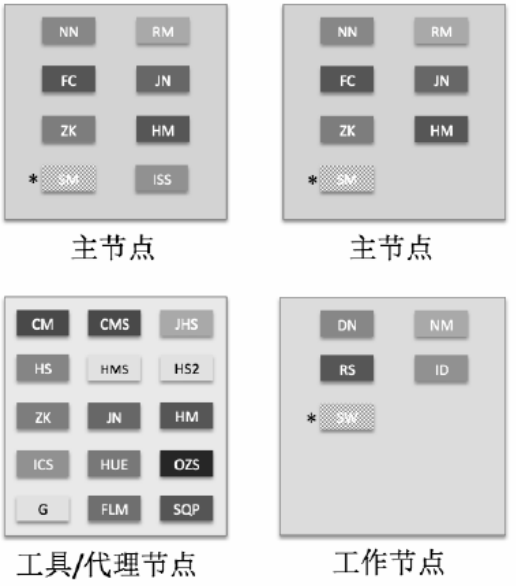


图 2-16

（2）对于一个中等规模的集群，节点数一般在 20～200，通常的数据存储可以规划到几百太字节，适用于一个中型企业的数据平台或者大型企业的业务部门数据平台。节点的复用程度可以降低，可以按照管理节点、主节点、工具节点和工作节点来划分，如图 2-17 所示。

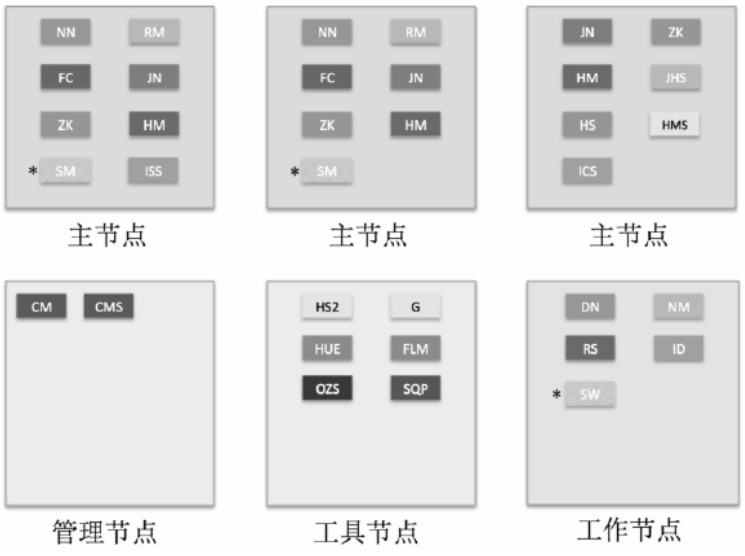


图 2-17

管理节点上安装 Cloudera Manager、Cloudera Management Service。主节点上安装 CDH 服务以及 HA 的组件。工具节点部署 HiveServer2、Hue Server、Oozie Server、Flume Agent、Sqoop Client、Gateway。工作节点的部署和小规模集群类似。

（3）大规模集群的数量一般会在 200 以上，存储容量可以是几百太字节（TB）甚至是拍字节（PB）级别，适用于大型企业搭建全公司的数据平台，如图 2-18 所示。

这里 HDFS JournalNode 由 3 个增加到 5 个，ZooKeeper Server 和 HBase Master 也由 3 个增加到 5 个，Hive Metastore 的数量由 1 个增加到 3 个。和中等规模的集群相比，部署的方案相差不大，主要是一些主节点可用性的增强。

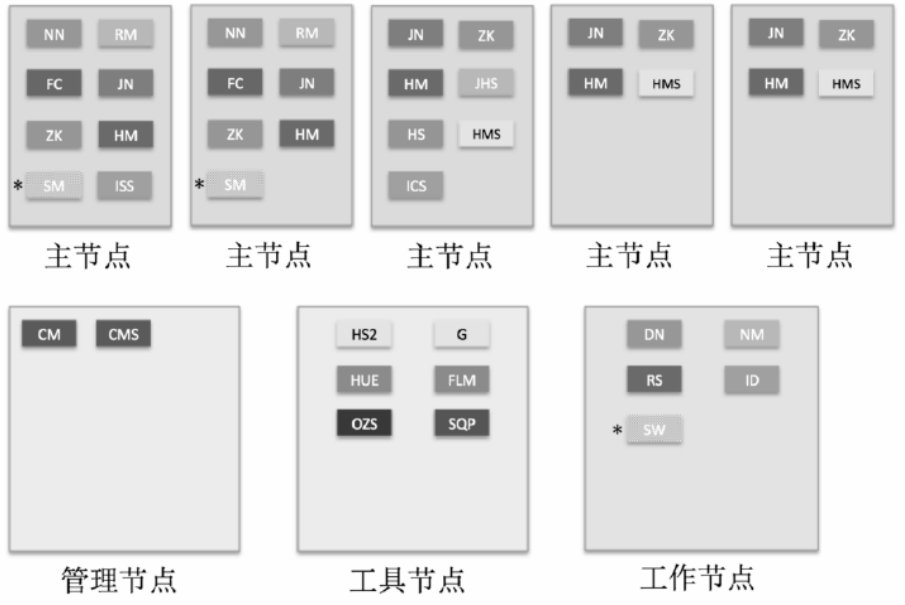


图 2-18

2.4.4　网络拓扑

对于一个小规模的集群或者单个 rack 的集群，所有的节点都连接到相同的接入层交换机。接入层交换机配置为堆叠的方式，互为冗余并增加了交换机吞吐。所有的节点两个网卡配置为主备或者负载均衡模式，分别连入两个交换机。在这种部署模式下，接入层交换机充当了聚合层的角色。

在多机架的部署模式下，除了接入层交换机，还需要聚合层交换机，用于连接各接入层交换机，负责跨 rack 的数据存取。

在机架上分配角色时，为了避免接入层交换机的故障导致集群的不可用，需要将一些高可用的角色部署到不同的接入层交换机之下（注意是不同的接入层之下，而不是不同的物理 rack 下，很多时候，客户会将不同物理 rack 下的机器接入到相同的接入层交换机下）。一个 80 个节点的物理部署示例如图 2-19 所示。

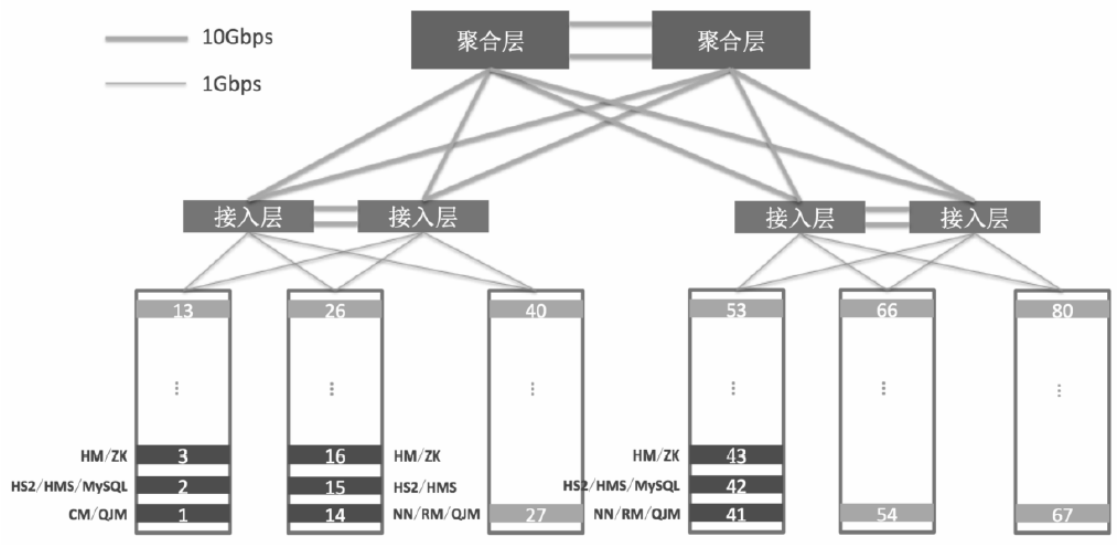

10Gbps
1Gbps
聚合层
聚合层
接入层
接入层
接入层
接入层
13
26
40
53
66
80
HM/ZK
HS2/HMS/MySQL
CM/QJM
3
2
1
16
15
14
HM/ZK
HS2/HMS
NN/RM/QJM
27
HM/ZK
HS2/HMS/MySQL
NN/RM/QJM
43
42
41
54
67

图 2-19

第 3 章

Cloudera Manager 及 CDH 离线安装部署

Cloudera Manager（CM）是由 Cloudera 公司提供的大数据组件自动部署和监控管理工具。CDH 是 Cloudera 公司在 Apache Hadoop 社区版的基础上做了商业化封装的大数据平台。Apache Hadoop 服务的部署非常烦琐，需要手动编辑配置文件、下载依赖包、协调版本兼容等。Cloudera Manager 以 GUI 的方式管理 Cloudera Hadoop 集群，并提供向导式的安装步骤。

3.1 安装前的准备工作

Cloudera 大数据平台默认采用在线自动化安装的方式，这给不能连接互联网或者网络不畅的用户带来了不便，很多时候是由于网络问题导致安装失败。所以这里我们采用离线安装 Cloudera Hadoop 集群的方法。

所需软件列表如表 3-1 所示。

表 3-1 离线安装 Cloudera Hadoop 集群所需软件列表

软件类型	名称
Linux 操作系统	CentOS-6.5-x86_64.ISO
sftp 文件传输	Winscp-5.9.2-Setup.exe
CDH（Cloudera 公司的 Hadoop 发行版）	CDH-5.11.2-1.cdh5.11.2.p0.4-el6.parcel
Cloudera Manager	cloudera-manager-el6-cm5.11.2_x86_64.tar.gz
JDK1.7	oracle-j2sdk1.7-x86_64.rpm
MySQL 数据库	MySQL 5.6.35
MySQL 的 JDBC 驱动	mysql-connector-java-5.1.42-bin.jar

但是 CDH 比 Apache Hadoop 对硬件的要求更高，如果节点分配内存太少，就很容易导致安装失败或服务无缘无故停止。哪怕只是做测试，也建议将主节点分配 8GB 以上的内存、从节点分配 5GB

内存。

本次部署的 Hadoop 测试环境是三个节点的 CDH 集群，node0 是主节点、node1 和 node2 是从节点，群集中的节点承担的角色如表 3-2 所示，这些角色在本书后续章节都会有详细阐述。

表 3-2　群集中的节点承担的角色

IP 地址	主机名	Hadoop 集群中的角色说明
10.10.75.100	node0	NameNode、DataNode（HDFS 集群的角色） ResourceManager（YARN 的核心角色） HMaster（HBase 的角色） JobHistory Server（MapReduce 的历史作业服务器角色） Hive Metastore Server（Hive 的角色）
10.10.75.101	node1	DataNode（HDFS 集群的角色） RegionServer（HBase 的角色） SecondaryNameNode（HDFS 集群的角色） NodeManager（YARN 框架的角色）
10.10.75.102	node2	DataNode（HDFS 集群的角色） RegionServer（HBase 的角色） NodeManager（YARN 的角色）

（1）下载介质软件

Cloudera Manager 的介质下载地址为 http://archive-primary.cloudera.com/cm5/cm/5/，节点的 Linux 操作系统是 Centos 6.5，CM 版本选择的是 5.11.2，所以选择 cloudera-manager-el6-cm5.11.2_x86_64.tar.gz，如图 3-1 所示。Hadoop 生态系统需要的所有组件都是通过 Cloudera Manager 统一管理和安装的。

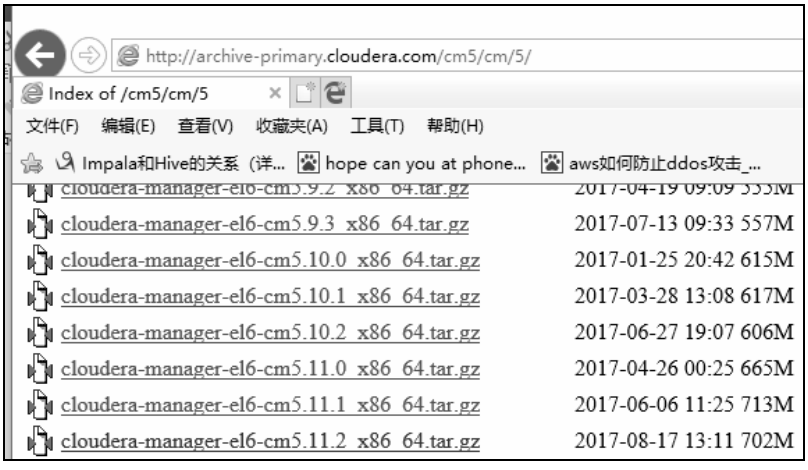


图 3-1

CDH（Cloudera 的 Hadoop 发行版）介质的下载地址是 http://archive-primary.cloudera.com/cdh5/parcels/，这里选择 5.11.2 版本。下载的 CDH 安装包一定要和 CM 包匹配。CDH 软件包以.parcel 结尾，相当于压缩包格式，这里要下载三个文件（包括 manifest.json），特别注意要将下载.sha1 文件后缀更改为.sha，如图 3-2 所示。

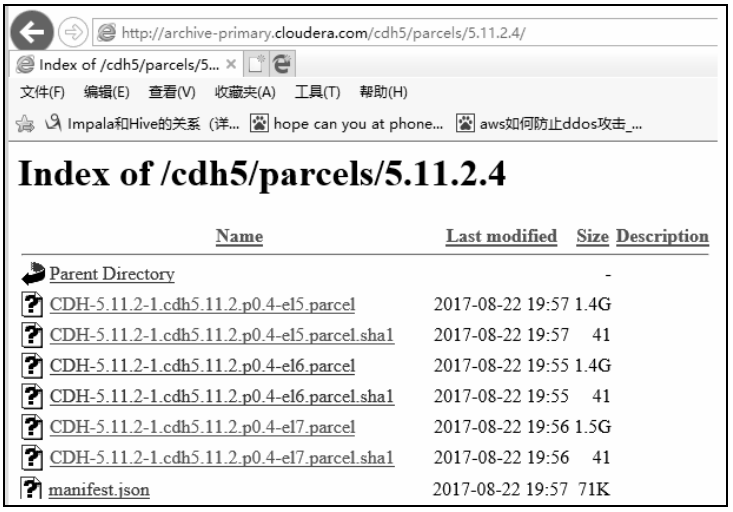


图 3-2

将下载的 CDH、CM、mysql 5.6 rpm 包、mysql jdbc 驱动、Oracle Java SDK 1.7 上传到群集主节点机器上的/opt 目录下，安装过程中就不需要从互联网上下载文件了，实现了离线安装。

（2）安装 Oracle 1.7 JDK

CDH 的运行依赖 JDK 的运行环境。所以在安装 CDH 之前一定要先安装 Oracle 1.7 JDK。通过 rpm -qa | grep jdk 命令来查询系统是否已经安装自带的 openjdk 的 RPM 包，如果有，请使用 rpm -e --nodeps 把它卸载，然后使用 rpm -ivh /opt/oracle-j2sdk1.7-x86_64.rpm 命令来安装 Oracle 1.7 JDK 的 RPM 包。

还要记得修改 vi /etc/profile，添加以下内容：

```
export JAVA_HOME=/usr/java/jdk1.7.0_67-cloudera
export PATH=.:$JAVA_HOME/bin:$PATH
export CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar
```

保存并退出 vi 编辑后，执行 source /etc/profile 使之生效。

（3）关闭 SELinux

SELinux 是一种安全子系统，它能控制程序只访问特定文件，但是在多数情况下我们还是要将其关闭，因为在不了解其机制的情况下，使用 SELinux 会导致软件安装或者应用部署失败。所有节点都要关闭 SELinux， 通过修改 gedit /etc/selinux/config 下的 SELINUX=disabled（重启后永久生效）完成。

（4）设置主机 hosts 文件，调整系统参数，关闭防火墙，禁用透明大页，调整 swap 和文件句柄，安装 Oracle JDK，添加环境变量。

编辑配置 hosts 文件，规划中的每一台机器都要配置集群中所有机器的 IP 和主机名称的对应关系。通过 vi 命令，编辑/etc/hosts 内容如下：

- 10.10.75.100 node0
- 10.10.75.101 node1
- 10.10.75.102 node2

（5）关闭防火墙

我们要搭建集群，集群之间就会有通信，服务器之间要是有通信，就要有相应的防火墙策略开放，因此我们要将防火墙关闭，操作命令如下：

```
service iptables stop
service iptables off
chkconfig iptables off
```

（6）调整 Linux 系统参数

修改 swappiness=0，最大限度使用物理内存，然后才是 swap 交换分区，命令如下：

```
echo 0 >/proc/sys/vm/swappiness
echo "vm.swappiness=0" >> /etc/sysctl.conf
echo "echo 0 > /proc/sys/vm/swappiness" >>/etc/rc.d/rc.local
cat /proc/sys/vm/swappiness
```

禁用 hugepage 透明大页，因为它可能会带来 CPU 利用过高的问题。将这个参数设置为 nerver，为了保证重启生效，要把命令写到/etc/rc.local 中，命令如下：

```
    echo "echo never > /sys/kernel/mm/transparent_hugepage/enabled"
>>/etc/rc.d/rc.local
    echo "echo never > /sys/kernel/mm/redhat_transparent_hugepage/defrag"
>>/etc/rc.d/rc.local
```

修改 Linux 最大文件句柄数（默认 Linux 最大文件句柄数为 1024），命令如下：

```
echo "* soft nofile 128000" >> /etc/security/limits.conf
echo "* hard nofile 128000" >> /etc/security/limits.conf
echo "* soft nproc 128000" >> /etc/security/limits.conf
echo "* hard nproc 128000" >> /etc/security/limits.conf
sed -i 's/1024/unlimited/' /etc/security/limits.d/90-nproc.conf
ulimit -SHn 128000
ulimit -SHu 128000
```

（7）配置时间同步

在所有要安装 CDH 环境的设备中需要设置统一时钟同步服务。如果我们有 NTP 时间同步器，那么我们需要在每一台设备上进行 NTP 客户端配置。如果没有，我们就将其中一台主机作为 NTP 时间同步服务器，对这台主机进行 NTP 服务器配置。其他服务器来同步这台服务器的时钟（修改 NTP 配置文件/etc/ntp.conf，指向企业自己的时间同步服务器 IP 地址）。

关于如何部署 NTP 时间同步服务器，可参阅作者的博客文章 http://blog.51cto.com/lihuansong/2172270。

（8）SSH 免密码登录

为什么要设置 SSH 免密码登录？其原因是在开启 Hadoop 的时候需要多次输入 yes 和 root 密码，这是我们所不能忍受的，迫切需要实现免登录的功能。

对于集群间免密的设置很简单，只要知道原理就好做了。分别在每台机器上配置本地免密登录，然后将其余的每台机器生成的公钥内容追加到其中一台主机的 authorized_keys 中，再将这台机器中包括每台机器公钥的 authorized_keys 文件发送到集群中所有的服务器，这样集群中每台服务器就都拥有所有服务器的公钥了，集群间任意两台机器都可以实现免密登录。关于如何配置 SSH

免密码登录，可参阅作者的博客文章 http://blog.51cto.com/lihuansong/2172326。

（9）安装 HTTP 服务

CM 的管理界面是 Web 访问方式，在主节点上安装并启动 HTTP 服务，命令如下：

```
yum install httpd
chkconfig httpd on
service httpd start
```

（10）安装 MySQL 数据库

CM 的元数据需要存储在数据库中。CM 支持 MySQL、PostgreSQL、Oracle 等数据库，通常我们会使用 MySQL 数据库（MySQL 的版本建议为 5.6 以上）。关于如何安装 MySQL，可参阅作者的博客文章 http://blog.51cto.com/lihuansong/2172326。

安装好 MySQL，在 MySQL 客户端执行命令 mysql -uroot -p，输入密码，在 MySQL 中创建相关数据库，命令如下：

```
create database oozie DEFAULT CHARSET utf8 COLLATE utf8_general_ci;
create database hive DEFAULT CHARSET utf8 COLLATE utf8_general_ci;
create database sentry DEFAULT CHARSET utf8 COLLATE utf8_general_ci;
create database scm DEFAULT CHARSET utf8 COLLATE utf8_general_ci;
create database monitor DEFAULT CHARSET utf8 COLLATE utf8_general_ci;
create database metastore DEFAULT CHARSET utf8 COLLATE utf8_general_ci;
create database amon DEFAULT CHARSET utf8 COLLATE utf8_general_ci;
```

（11）在主节点 node0 上操作，解压 CM，准备 Parcels

操作命令如下：

```
    tar xzvf /opt/cloudera-manager-el6-cm5.11.2_x86_64.tar.gz  -C /opt
    cp /opt/CDH-5.11.2-1.cdh5.11.2.p0.4-el6.parcel
/opt/cloudera/parcel-repo/
    cp /opt/CDH-5.11.2-1.cdh5.11.2.p0.4-el6.parcel.sha
/opt/cloudera/parcel-repo/
    cp /opt/manifest.json  /opt/cloudera/parcel-repo/
```

（12）修改 config.ini，同步 agent 到所有节点

通过 vi 编辑 /opt/cm-5.11.2/etc/cloudera-scm-agent/config.ini，把其中的 server_host 值改为主节点的主机名，这里主节点是 node0，如图 3-3 所示。

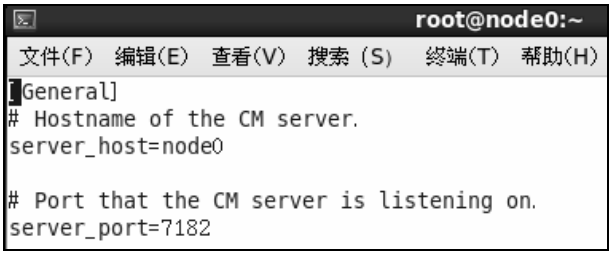


图 3-3

然后把 config.ini 同步到其他节点，命令如下：

```
scp -r /opt/cm-5.11.2 root@node1:/opt/
scp -r /opt/cm-5.11.2 root@node2:/opt/
```

（13）所有节点都建立 cloudera-scm 用户

命令如下：

```
    useradd --system --home=/opt/cm-5.11.2/run/cloudera-scm-server/ --no-
create-home --shell=/bin/false --comment  "Cloudera SCM User" cloudera-scm
```

（14）主节点上初始化配置数据库

CM Server 的主要数据库为 scm，里面包含了服务的配置信息，每一次配置的更改都会把当前页面的所有配置内容添加到数据库中，以保存配置修改历史。

初始化配置数据库 scm 的命令如下：

```
    /opt/cm-5.11.2/share/cmf/schema/scm_prepare_database.sh mysql cm
-hlocalhost -uroot -proot123 --scm-host localhost scm scm scm
```

（15）启动 CM 服务和 CM Agent

主节点上启动 CM Server 和 Agent，命令如下：

```
    /opt/cm-5.11.2/etc/init.d/cloudera-scm-server start
    /opt/cm-5.11.2/etc/init.d/cloudera-scm-agent start
```

所有从节点启动 Agent，命令如下：

```
    /opt/cm-5.11.2/etc/init.d/cloudera-scm-agent start
```

3.2 Cloudera Manager 及 CDH 安装

Cloudera Manager Server 和 Agent 都启动以后，就可以进行大数据基础平台的安装了。这时可以通过浏览器访问主节点 node0 的 7180 端口测试一下（由于 Cloudera Manager Server 的启动需要花点时间，这里可能要等待一会儿才能访问），默认的用户名和密码均为 admin，如图 3-4 所示。

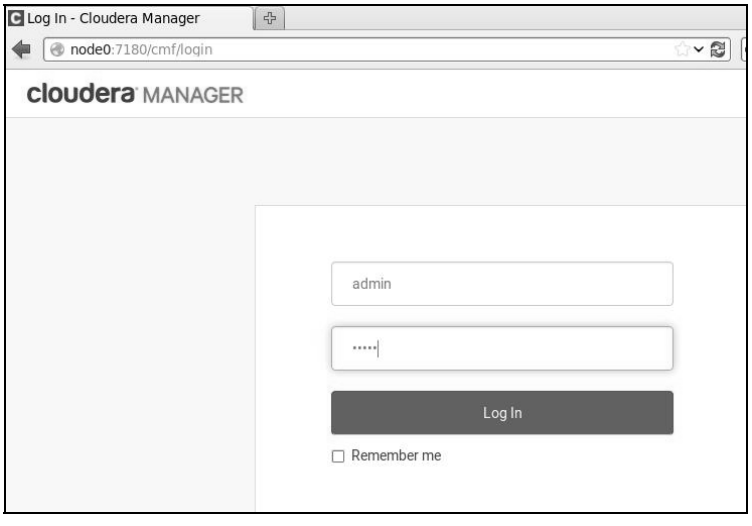


图 3-4

部署版本选择免费版本 Cloudera Express，免费版本除了拥有 CDH 和 Cloudera Manager 核心功能外，群集节点数量无任何限制，如图 3-5 所示。付费的 Cloudera Enterprise 企业版本还拥有 Cloudera Manager 高级功能、Cloudera Navigator 审核组件和商业技术支持。

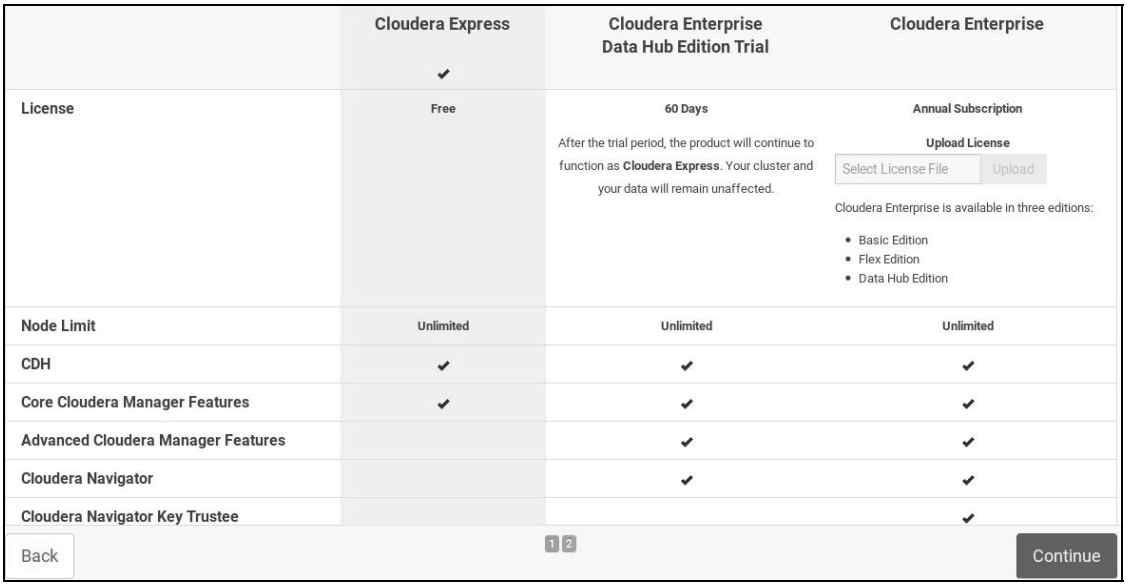


图 3-5

接下来，选择需要安装的节点主机。由于我们在各个节点都安装并启动了 Agent，各个节点的配置文件 config.ini 的 server_host 都指向主节点 node0，因此我们可以在“Currently Managed Hosts”（当前管理的主机）中看到三个主机，如图 3-6 所示，全部勾选并继续。如果 cloudera-scm-agent 没有启动，这里会检测不到主机。

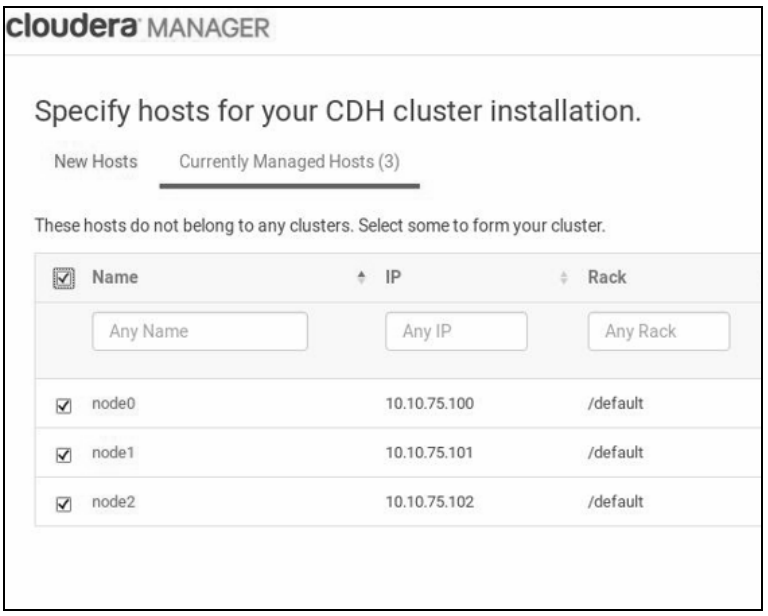


图 3-6

这里你会看到已经提前下载好的 Parcel 包对应的 CDH 版本，如 CDH-5.11.2，如图 3-7 所示。

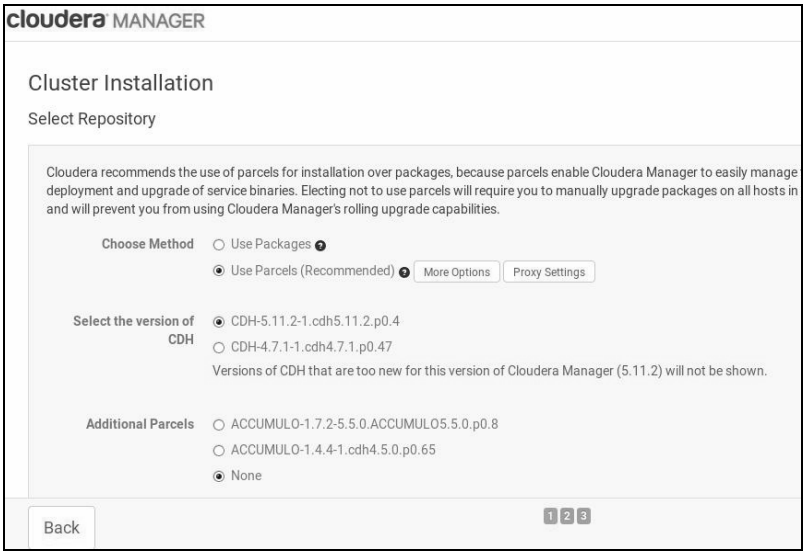


图 3-7

如果配置本地 Parcel 包无误，那么 Parcel 包的下载应该是瞬间就完成了，并由 CM 将 Parcel 文件包分发到各个节点。 Parcel 包分发完后，点击“Continue”按钮，进入到检查群集主机正确性的界面。Cloudera 会进行安装前各节点的检查工作，比如 Cloudera 建议将 swappiness 设置为 0 ，主机时钟要同步、禁用透明大页等，配置没有问题就打勾，如图 3-8 所示。

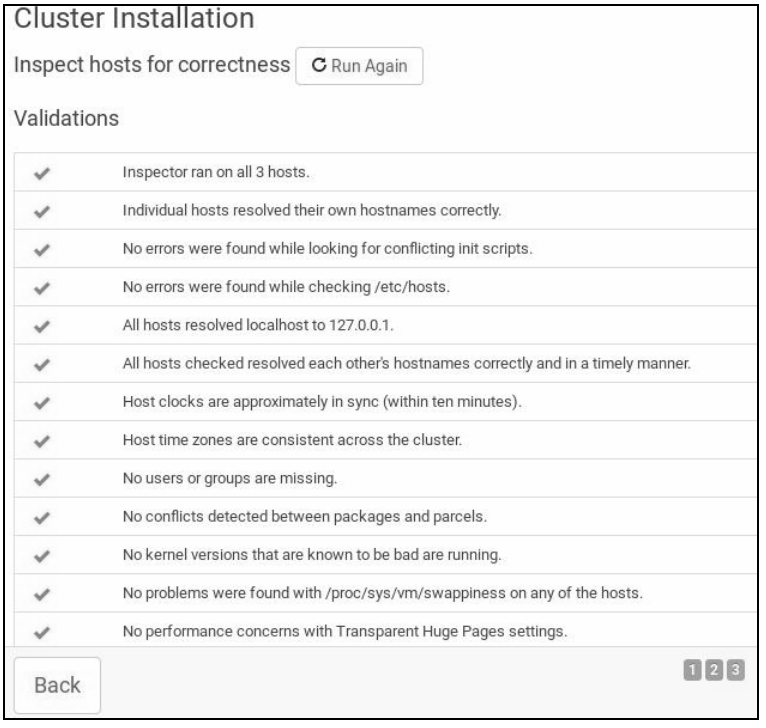


图 3-8

选择需要安装的大数据组件，我们可以选择自定义安装方式“Custom Services”，如图 3-9 所示。

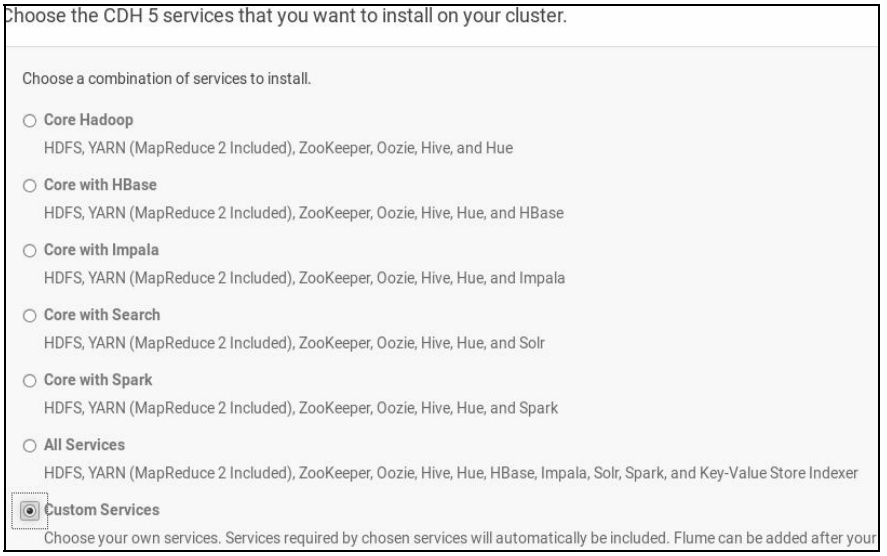


图 3-9

这里选择 HBase、HDFS、Hive、YARN、ZooKeeper 等服务组件，如图 3-10 所示。

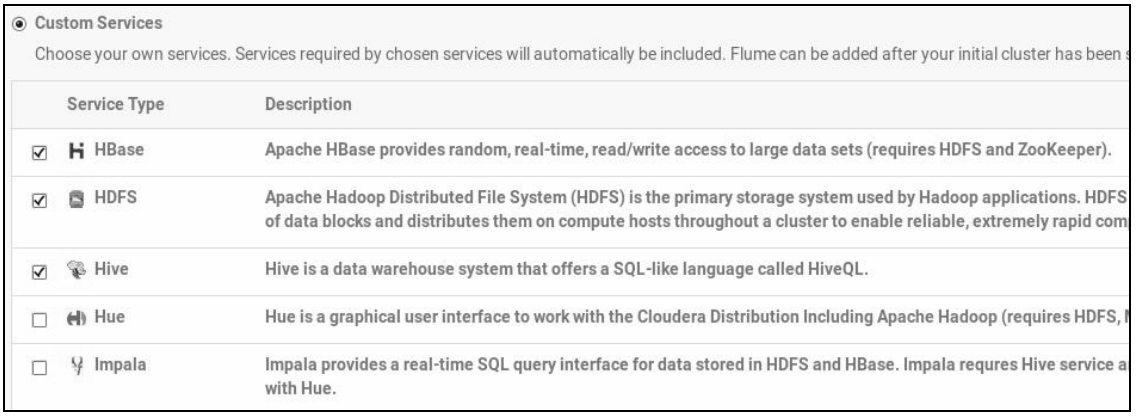


图 3-10

然后给集群各个节点分配角色，如 HDFS 需要的角色有 NameNode（名称节点，也称名称节点）、SecondaryNameNode（第二名称节点）、DataNode（数据节点），HBase 必需的角色有 HMaster、RegionServer（与 DataNode 在同一节点上）等，如图 3-11 所示。如果系统配置有什么问题，在安装过程中会有提示，根据提示安装组件就可以了。

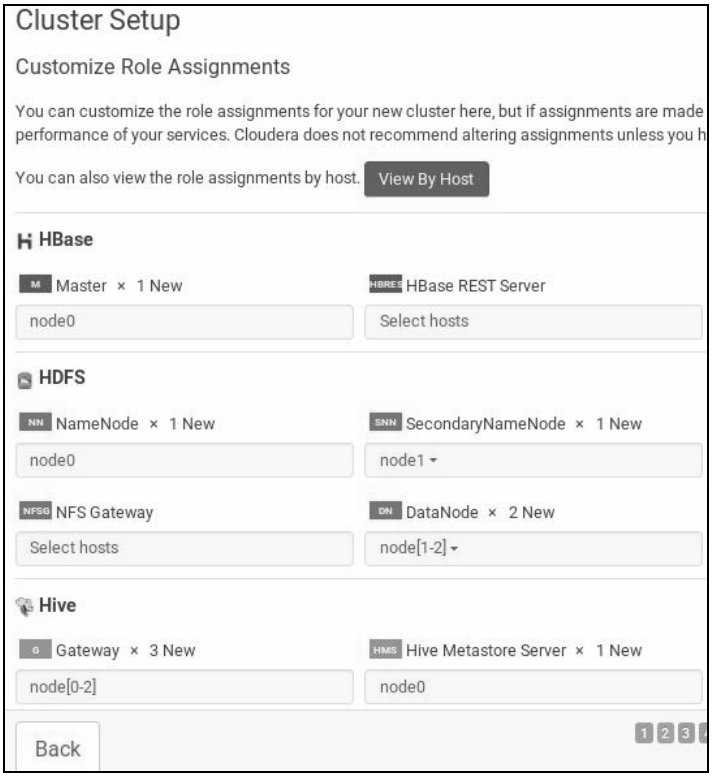


图 3-11

此处选择 Hive 组件的元数据库，使用 MySQL 来存储 Hive 元数据信息，如图 3-12 所示。

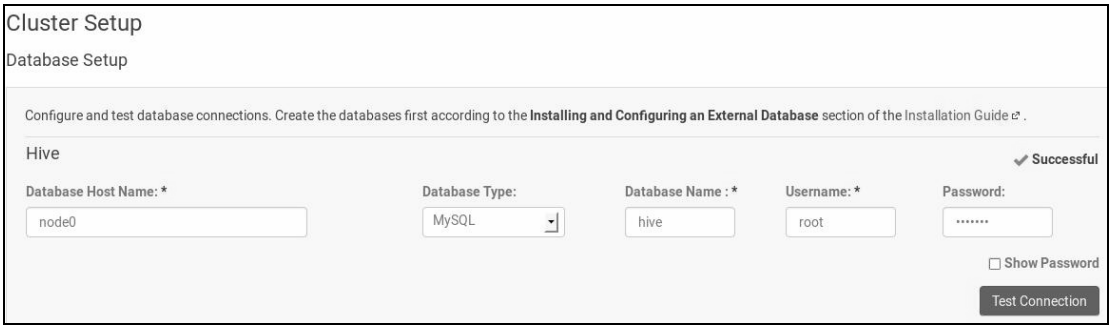


图 3-12

需要注意的是，若“Test Connection”确认数据库的连通性没有通过，则需要复制 MySQL 的 JDBC 驱动到相应目录，复制命令如下：

```
    cp /opt/mysql-connector-java-5.1.42-bin.jar /opt/cloudera/parcels/
CDH-5.11.2-1.cdh5.11.2.p0.4/lib/hive/lib/
    cp /opt/mysql-connector-java-5.1.42-bin.jar /opt/cm-5.11.2/share/
cmf/lib/
    cp /opt/mysql-connector-java-5.1.42-bin.jar /usr/share/java/mysql-
connector-java.jar
```

最后，CM 开始配置并启动各项服务，直到安装过程全部完成，CM 管理界面如图 3-13 所示。

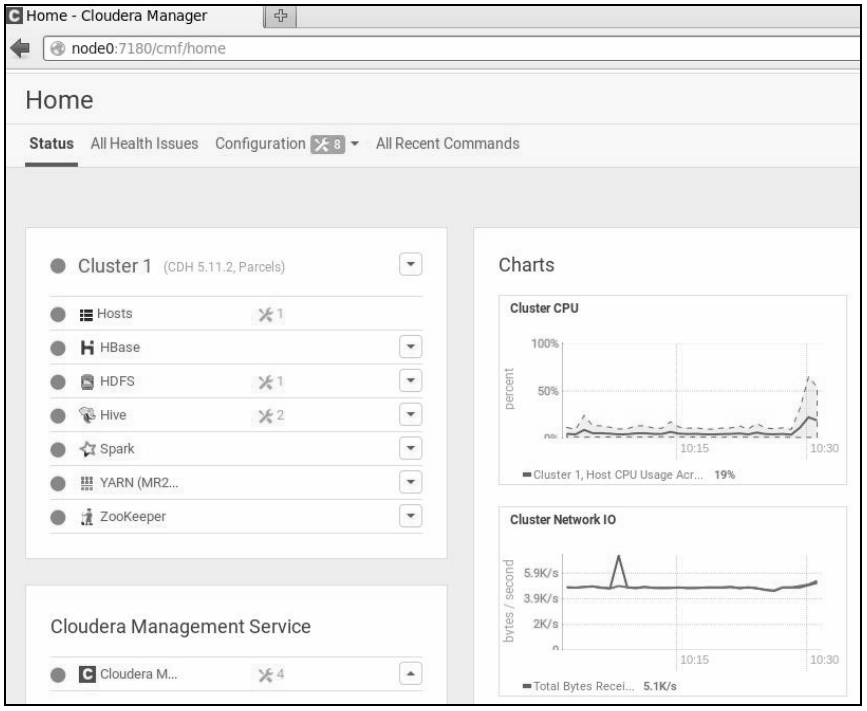


图 3-13

3.3　添加其他大数据组件

在 Cloudera Manager 中单击“添加服务”选项，如图 3-14 所示。

图 3-14

在添加服务向导中，选择要添加的服务组件，如“Spark（Standalone）”，如图 3-15 所示。

图 3-15

选择 Spark 角色分配。在 Spark 集群中重要的角色有 Master 和 Worker：Master 负责分配资源；Worker 负责监控自己节点的内存和 CPU 等状况，并向 Master 汇报。角色分配如图 3-16 所示。

图 3-16

Spark(Standalone)组件安装完成后，如图 3-17 所示。

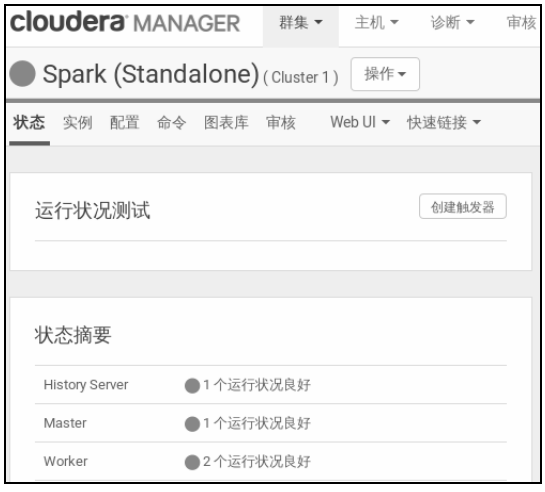


图 3-17

如果还要安装其他 Hadoop 生态系统的组件，也可以通过 Cloudera Manager 统一管理和安装。

第4章

分布式文件系统 HDFS

为了解决海量数据存储问题，Google 开发了分布式文件系统 GFS。HDFS 是 GFS 的开源实现，它是 Hadoop 的核心组件之一。HDFS 提供了在通用硬件集群中进行分布式文件存储的能力，是一个高容错性和高吞吐量的海量数据存储解决方案。

4.1 HDFS 简介

HDFS（Hadoop Distributed Filesystem，Hadoop 分布式文件系统）以流式数据访问模式来存储超大文件，运行在由廉价普通机器组成的集群上，是管理网络中跨多台计算机存储的文件系统。它的基本原理是将文件切分成同等大小的数据块，存储到多台机器上，将数据切分、容错、负载均衡等功能透明化。

HDFS 上的文件被划分为相同大小的多个 block 块，以块作为独立的存储单位（称为数据块）。为什么要弄成块来存储？第一，大文件用一个节点是存不下来的，势必分成块；第二，网络传输时万一宕掉，可以小部分重传；第三，简化了存储管理，同时元数据就不需要和块一同存储了，用一个单独的系统就可以管理这些块的元数据。所以 block 块是 HDFS 中最基本的存储单位。一个文件 Hadoop 2.x 版本的 HDFS 块默认大小是 128MB（Hadoop 1.X 版本默认块大小是 64MB）。默认块大小是可以修改的，可以通过 dfs.block.size 设置。

除了将文件分块，每个块文件也有副本，这是为了容错性。当一个机器挂了，想要恢复里面的文件，就可以去其他机器找文件的副本。默认是三个副本，也可通过 hdfs-site.xml 中的 replication 属性修改副本数量。

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。机架故障的概率远小于节点故障。将第三个副本放置在不同的机架上，这也防止了机架故

障时数据的丢失。

总之，HDFS 在设计之初就是针对超大文件存储的，小文件不会提高访问和存储速度，反而会降低。其次它采用了流式数据访问，特点是一次写入多次读取。再有就是它运行在普通的标准硬件（如 PC 服务器）之上，即使硬件故障，也可以通过副本冗余容错机制来保证数据的高可用。

4.2 HDFS 体系结构

4.2.1 HDFS 架构概述

HDFS 采用主从（Master/Slave）架构模型，分为 NameNode（名称节点）、SecondaryNameNode（第二名称节点）、DataNode（数据节点）这几个角色（遵从中国读者习惯，本文混用这 3 个中英文术语），如图 4-1 所示。

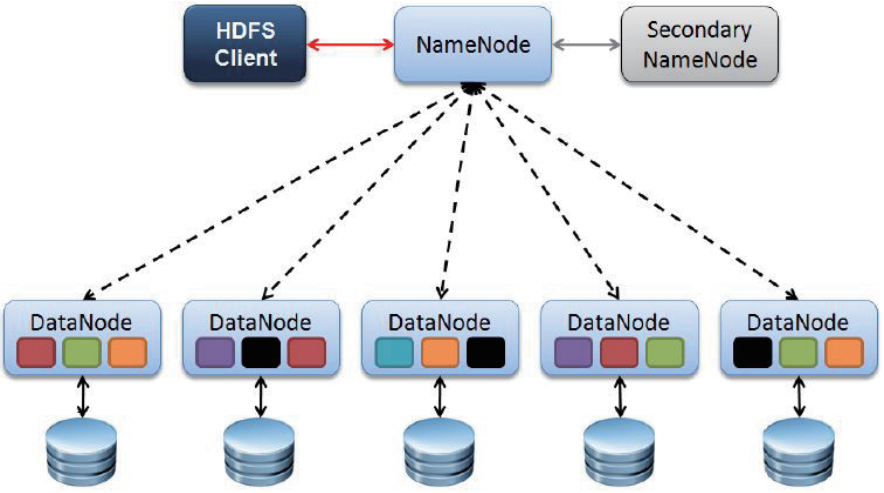


图 4-1

一个典型的 HDFS 集群是由一个 NameNode、一个 SecondaryNameNode 和若干个 DataNode（通常大于 3 个）组成的，通常是一个节点一个机器，它来管理对应节点的存储。

（1）NameNode：主要负责文件系统命名空间的管理、存储文件目录的 Metadata 元数据信息，主要包括文件目录、block 块和文件对应关系，以及 block 块和 DataNode 数据节点的对应关系。

（2）SecondaryNameNode：是 NameNode 的冷备份，用来减少 NameNode 的工作量。

（3）DataNode：负责存储客户端（Client）发来的 Block 数据块，执行数据块的读写操作。

4.2.2 HDFS 命名空间管理

HDFS 的命名空间包含目录、文件和块。在 HDFS1.0 架构中，在整个 HDFS 集群中只有一个命名空间，并且只有唯一一个 NameNode，负责对这个命名空间进行管理。HDFS 使用的是传统的

分级文件体系，因此用户可以像使用普通文件系统一样创建、删除目录和文件以及在目录间移动文件、重命名文件等。HDFS2.0 新特性 federation 联邦功能支持多个命名空间，并且允许在 HDFS 中同时存在多个 NameNode。

4.2.3　NameNode

HDFS 集群的命名空间是由 NameNode 来存储的。NameNode 使用 FsImage 和 EditLog 两个核心的数据结构，如图 4-2 所示。EditLog 事务日志文件记录每一个对文件系统元数据的改变，如在 HDFS 中创建一个新的文件，名称节点将会在 EditLog 中插入一条记录来记录这个改变。整个命名空间的信息包括文件块的映射表等都存放在 FsImage 文件中。

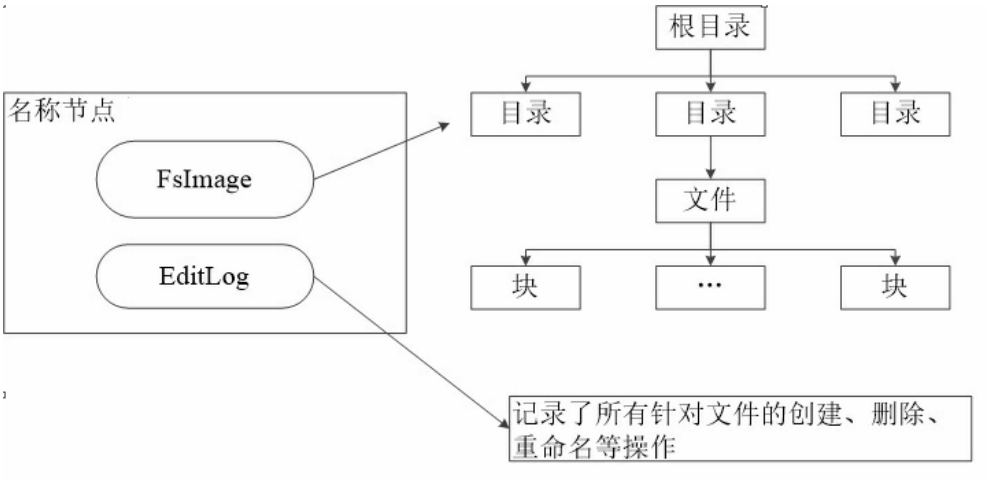


图 4-2

名称节点启动时，它将从磁盘中读取 FsImage 和 EditLog，将 EditLog 中的所有事务应用到 FsImage，然后将新的 FsImage 刷新到本地磁盘中，因为事务已经被处理并已经持久化到 FsImage 中，然后就可以截去旧的 EditLog。这个过程叫作检查点。

FsImage 和 Editlog 是 HDFS 的重要数据结构，如果这些文件损坏，就会导致整个集群的失效。因此可以配置成复制多个 FsImage 和 EditLog 的副本，一般会在本地磁盘和网络文件系统 NFS 中分别存放。

4.2.4　SecondaryNameNode

SecondaryNameNode 是 HDFS 架构中的一个组成部分，它用来保存名称节点中对 HDFS 元数据信息的备份，减小 Editlog 文件大小，从而缩短名称节点重启的时间。它一般是单独运行在一台机器上。

SecondaryNameNode 让 EditLog 变小的工作流程如下（见图 4-3）：

（1）SecondaryNameNode 会定期和 NameNode 通信，请求其停止使用 EditLog 文件，暂时将新的写操作写到一个新的文件 edit.new 中，这个操作是瞬间完成的，上层写日志的函数完全感觉不

到差别。

（2）SecondaryNameNode 通过 HTTP GET 方式从 NameNode 上获取到 FsImage 和 EditLog 文件，并下载到本地的相应目录下。

（3）SecondaryNameNode 将下载下来的 FsImage 载入到内存，然后一条一条地执行 EditLog 文件中的各项更新操作，使内存中的 FsImage 保持最新。这个过程就是 EditLog 和 FsImage 文件合并。

（4）SecondaryNameNode 执行完（3）操作之后，会通过 post 方式将新的 FsImage 文件发送到 NameNode 节点上。

（5）NameNode 将从 SecondaryNameNode 接收到的新的 FsImage 替换旧的 FsImage 文件，同时将 Edit.new 替换 EditLog 文件，从而减小 EditLog 文件大小。

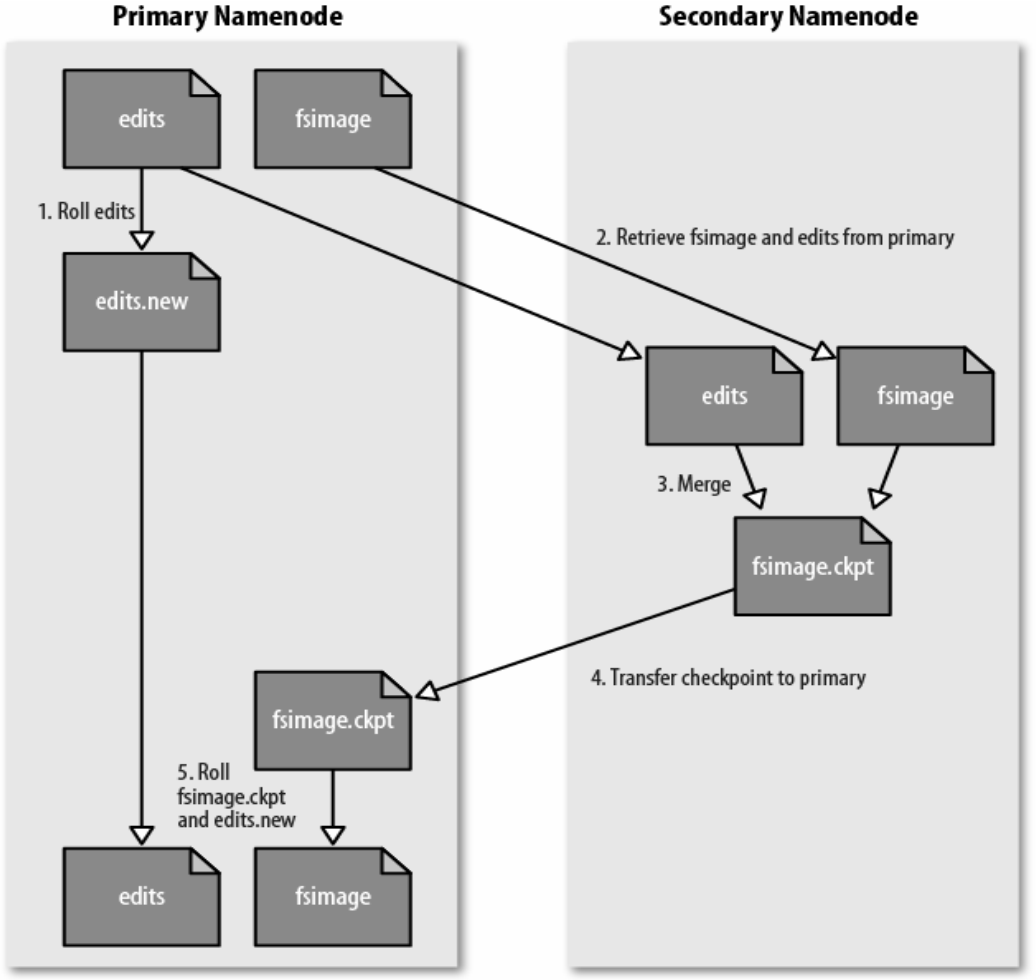


图 4-3

从上面的过程可以看出，第二名称节点相当于为名称节点设置一个“检查点”，周期性备份名称节点中的元数据信息，但第二名称节点在 HDFS 设计中只是一个冷备份，并不能起到“热备份”的作用。HDFS 设计并不支持当名称节点故障时直接切换到第二名称节点。

4.3　HDFS 2.0 新特性

4.3.1　HDFS HA

HDFS1.0 中虽然存在一个第二名称节点（SecondaryNameNode），但第二名称节点无法提供“热备份”功能，一旦名称节点发生故障，系统需要停机恢复。HDFS2.0 采用 HA（High Availability）架构，用于解决 NameNode 单点故障问题。该 HA 特性通过热备份的方式为主 NameNode 提供一个备用者，一旦主 NameNode 出现故障，可以迅速切换至备用 NameNode，从而实现不间断对外提供服务。

一个典型的 HDFS HA 架构如图 4-4 所示，它通常由两个 NameNode 组成：一个处于 Active 状态，另一个处于 Standby 状态。Active NameNode 对外提供服务，比如处理来自客户端的请求，而 Standby NameNode 则不对外提供服务，仅同步 Active NameNode 的状态，以便能够在它失败时快速进行切换。

注意，HA 中的两个 NameNode 属于同一命名空间。两个 NameNode 为了能够实时同步元数据信息（实际上是共享 EditLog），会通过一组称作 JournalNodes 的独立进程相互通信。

每个 Journal 节点暴露一个简单的 RPC 接口，允许 NameNode 读取和写入数据，数据存放在 Journal 节点的本地磁盘。当 Active NameNode 写入 EditLog 时，它向集群的所有 JournalNode 发送写入请求，当多数节点回复确认成功写入之后，EditLog 就认为是成功写入。

StandbyNameNode 负责监听，一旦发现有新数据写入，就读取这些数据，并加载到自己内存中，以保证自己内存状态与 Active NameNode 保持基本一致。

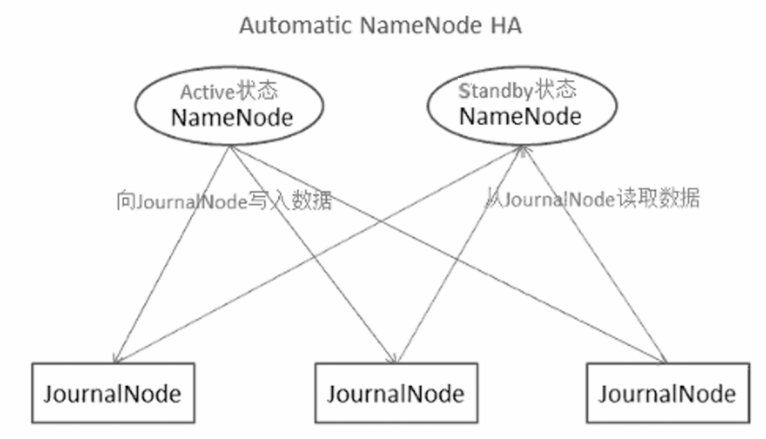


图 4-4

Hadoop 使用 ZooKeeper 支持自动故障转移，ZooKeeper 的任务包括 NameNode 失败检测和 NameNode 选举。

HDFS HA 集群的配置如下：

（1）NameNode 机器：运行 Active NameNode 和 Standby NameNode 的机器配置应保持一样。

（2）当 Active 状态的 NameNode 宕机后，需要手动切换到 Standby 状态的 NameNode 来继续提供服务。如果要实现自动故障转移，必须依赖 ZooKeeper。

（3）JournalNode 机器：运行 JournalNode 的机器，这些守护进程比较轻量级，可以部署在其他服务器上。至少需要部署 3 个 JournalNode 节点，以便容忍一个节点故障。通常配置成奇数，例如总数为 *N*，则可以容忍(*N*-1)/2 台机器发生故障后不影响集群正常运行。

（4）配置了 NameNode HA 后，客户端可以通过 HA 的逻辑名称去访问数据，而不用指定某一台 NameNode，当某一台 NameNode 失效自动切换后，客户端不必更改 HDFS 的连接地址，仍可通过逻辑名称去访问。

需要注意的是，Standby NameNode 同时完成了原来 SecondaryNameNode 的 checkpoint（检查点）功能，因此不需要再独立部署 SecondaryNameNode。

4.3.2 HDFS Federation

HDFS1.0 的单 NameNode 设计不仅存在单点故障问题，还存在可扩展性和性能问题。只有一个 NameNode，不利于水平扩展。HDFS Federation（HDFS 联邦）特性允许一个 HDFS 集群中存在多个 NameNode 同时对外提供服务，这些 NameNode 分管一部分目录（水平切分），彼此之间相互隔离，但共享底层的 DataNode 存储资源。每个 NameNode 是独立的，不需要和其他 NameNode 协调合作。

如图 4-5 所示，Federation 使用了多个独立的 NameNode/NameSpace 命名空间。这些 NameNode 之间是联合的，也就是说，它们之间相互独立且不需要互相协调，各自分工管理自己的区域。分布式的 DataNode 被用作通用的数据块存储设备。每个 DataNode 要向集群中所有的 NameNode 注册，且周期性地向所有 NameNode 发送心跳和块报告，并执行来自所有 NameNode 的命令。每一个 DataNode 作为统一的块存储设备被所有 NameNode 节点使用。

每一个 DataNode 节点都在所有的 NameNode 进行注册。 DataNode 发送心跳信息、块报告到所有 NameNode，同时执行所有 NameNode 发来的命令。

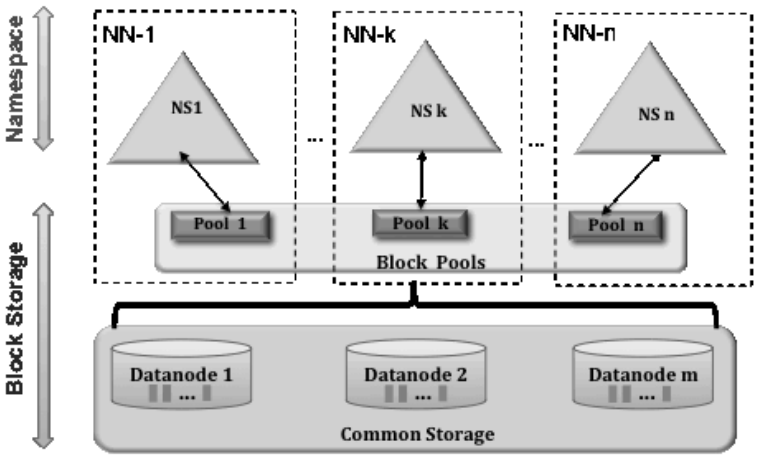


图 4-5

4.4　HDFS 操作常用 shell 命令

4.4.1　HDFS 目录操作和文件处理命令

我们可以利用 HDFS shell 命令对 Hadoop 进行操作，利用这些命令可以完成 HDFS 中文档的上传、下载、复制、查看文件信息、格式化名称节点等操作。使用 Cloudera CDH 版本安装 Hadoop 时，默认建立的 hdfs 用户是对集群文件的最高权限用户。如图 4-6 所示，在名称节点 node0 上运行 jps 命令查看进程，发现 NameNode 进程存在。在其他工作节点（如 node1）上运行 jps 命令查看进程，发现 DataNode 进程存在，如图 4-7 所示。

```
[root@node0 ~]# su - hdfs
[hdfs@node0 ~]$ jps
4087 NameNode
66949 Jps
```

图 4-6

```
[root@node1 ~]# su - hdfs
[hdfs@node1 ~]$ jps
3358 DataNode
3356 SecondaryNameNode
29670 Jps
[hdfs@node1 ~]$
```

图 4-7

在终端输入命令，查看 hdfs dfs 总共支持哪些操作，命令执行后会显示如图 4-8 所示的结果（这里只列出部分命令）。

```
[hdfs@node0 ~]$ hdfs dfs
Usage: hadoop fs [generic options]
        [-appendToFile <localsrc> ... <dst>]
        [-cat [-ignoreCrc] <src> ...]
        [-checksum <src> ...]
        [-chgrp [-R] GROUP PATH...]
        [-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
        [-chown [-R] [OWNER][:[GROUP]] PATH...]
        [-copyFromLocal [-f] [-p] [-l] <localsrc> ... <dst>]
        [-copyToLocal [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
        [-count [-q] [-h] [-v] [-x] <path> ...]
        [-cp [-f] [-p | -p[topax]] <src> ... <dst>]
        [-createSnapshot <snapshotDir> [<snapshotName>]]
        [-deleteSnapshot <snapshotDir> <snapshotName>]
        [-df [-h] [<path> ...]]
        [-du [-s] [-h] [-x] <path> ...]
        [-expunge]
        [-find <path> ... <expression> ...]
        [-get [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
        [-getfacl [-R] <path>]
        [-getfattr [-R] {-n name | -d} [-e en] <path>]
        [-getmerge [-nl] <src> <localdst>]
        [-help [cmd ...]]
        [-ls [-C] [-d] [-h] [-q] [-R] [-t] [-S] [-r] [-u] [<path> ...]]
        [-mkdir [-p] <path> ...]
        [-moveFromLocal <localsrc> ... <dst>]
        [-moveToLocal <src> <localdst>]
        [-mv <src> ... <dst>]
        [-put [-f] [-p] [-l] <localsrc> ... <dst>]
```

图 4-8

可以看出 hdfs dfs 命令的统一格式类似“hdfs dfs –ls”这种形式，即在“-”后面跟上具体的操作。需要查看某个命令的作用（例如，查询 ls 命令的具体用法）时，可以采用如图 4-9 所示的命令。

```
[hdfs@node0 ~]$ hdfs dfs -help ls
-ls [-C] [-d] [-h] [-q] [-R] [-t] [-S] [-r] [-u] [<path> ...] :
  List the contents that match the specified file pattern.  If path is not
  specified, the contents of /user/<currentUser> will be listed.  For a directory a
  list of its direct children is returned (unless -d option is specified).

  Directory entries are of the form:
        permissions - userId groupId sizeOfDirectory(in bytes)
  modificationDate(yyyy-MM-dd HH:mm) directoryName

  and file entries are of the form:
        permissions numberOfReplicas userId groupId sizeOfFile(in bytes)
  modificationDate(yyyy-MM-dd HH:mm) fileName

    -C  Display the paths of files and directories only.
    -d  Directories are listed as plain files.
    -h  Formats the sizes of files in a human-readable fashion
        rather than a number of bytes.
    -q  Print ? instead of non-printable characters.
    -R  Recursively list the contents of directories.
    -t  Sort files by modification time (most recent first).
    -S  Sort files by size.
    -r  Reverse the order of the sort.
    -u  Use time of last access instead of modification for
        display and sorting.
[hdfs@node0 ~]$
```

图 4-9

HDFS 目录操作和文件操作命令如图 4-10 所示。hdfs dfs -mkdir -p /doc 命令表示在 HDFS 根目录下创建一个称为 doc 的目录。hdfs dfs -ls / 命令表示列出 HDFS 根目录下的内容。使用 hdfs dfs -put 命令把本地文件系统的/var/lib/hadoop-hdfs/text.txt 上传到根目录的 doc 目录下，然后查看一下文件是否能成功上传到 HDFS 中。

```
[hdfs@node0 ~]$ hdfs dfs -mkdir -p /doc
[hdfs@node0 ~]$ hdfs dfs -ls /
Found 5 items
drwxr-xr-x   - hdfs  supergroup          0 2018-08-26 19:33 /doc
drwxr-xr-x   - hbase hbase               0 2018-08-25 22:13 /hbase
drwxr-xr-x   - root  supergroup          0 2018-05-19 14:38 /spool
drwxrwxrwx   - hdfs  supergroup          0 2018-08-26 19:05 /tmp
drwxr-xr-x   - hdfs  supergroup          0 2018-05-19 16:19 /user
[hdfs@node0 ~]$ cat >test.txt <<EOF
> test
> abc
> EOF
[hdfs@node0 ~]$ pwd
/var/lib/hadoop-hdfs
[hdfs@node0 ~]$ hdfs dfs -put /var/lib/hadoop-hdfs/test.txt /doc
[hdfs@node0 ~]$ hdfs dfs -ls /doc
Found 1 items
-rw-r--r--   2 hdfs supergroup          9 2018-08-26 19:35 /doc/test.txt
[hdfs@node0 ~]$
```

图 4-10

4.4.2 HDFS 的 Web 管理界面

HDFS 提供了 Web 管理界面，可以很方便地查看 HDFS 相关信息。需要在 Linux 系统打开浏览器，

在浏览器地址栏中输入 HDFS 的 NameNode 的 Web 访问地址，端口号为 50070，如图 4-11 所示。

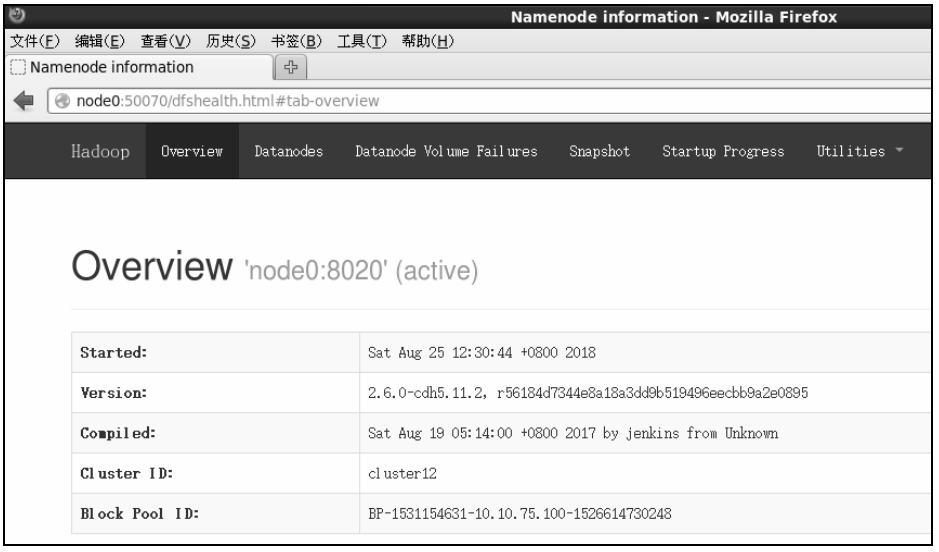


图 4-11

在 HDFS 的 Web 管理界面中，包含 Overview、DataNodes、DataNode Volume Failures、Snapshot、Startup Progress 和 Utilities 等菜单项。你可以点击每个菜单项，查询各种信息，如点击“Datanodes”，查看数据节点信息，如图 4-12 所示。

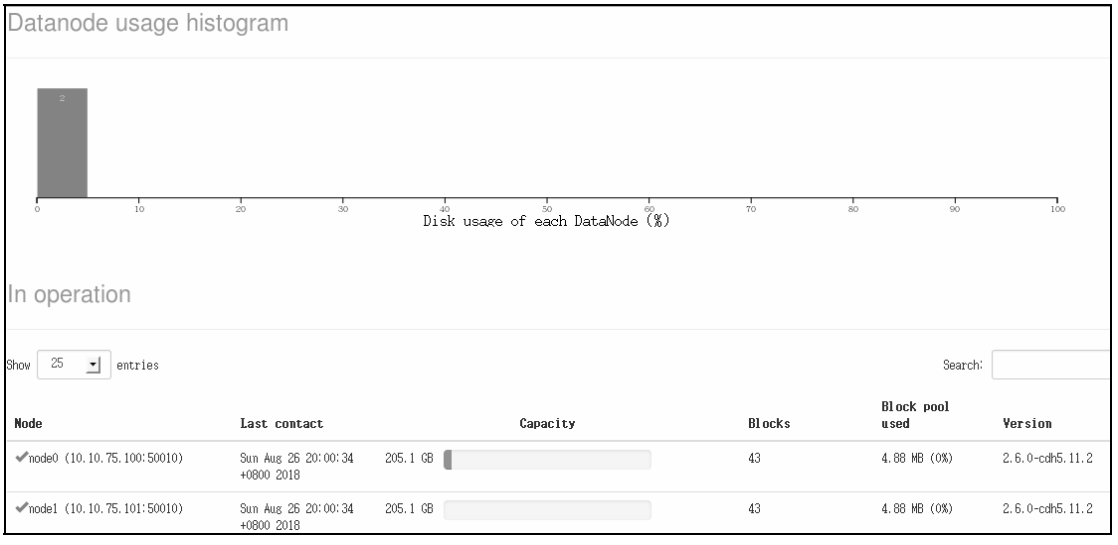


图 4-12

4.4.3　dfsadmin 管理维护命令

dfsadmin 是一个多任务客户端工具，用来显示 HDFS 运行状态和管理 HDFS，支持的命令如图 4-13 所示。

```
[hdfs@node0 ~]$ hdfs dfsadmin -help
hdfs dfsadmin performs DFS administrative commands.
Note: Administrative commands can only be run with superuser permission.
The full syntax is:

hdfs dfsadmin
        [-report [-live] [-dead] [-decommissioning]]
        [-safemode <enter | leave | get | wait>]
        [-saveNamespace]
        [-rollEdits]
        [-restoreFailedStorage true|false|check]
        [-refreshNodes]
        [-setQuota <quota> <dirname>...<dirname>]
        [-clrQuota <dirname>...<dirname>]
        [-setSpaceQuota <quota> <dirname>...<dirname>]
        [-clrSpaceQuota <dirname>...<dirname>]
        [-finalizeUpgrade]
        [-rollingUpgrade [<query|prepare|finalize>]]
        [-refreshServiceAcl]
        [-refreshUserToGroupsMappings]
        [-refreshSuperUserGroupsConfiguration]
        [-refreshCallQueue]
        [-refresh <host:ipc_port> <key> [arg1..argn]
        [-reconfig <datanode|...> <host:ipc_port> <start|status|properties>]
        [-printTopology]
        [-refreshNamenodes datanode_host:ipc_port]
        [-deleteBlockPool datanode_host:ipc_port blockpoolId [force]]
        [-setBalancerBandwidth <bandwidth in bytes per second>]
        [-fetchImage <local directory>]
        [-allowSnapshot <snapshotDir>]
        [-disallowSnapshot <snapshotDir>]
        [-shutdownDatanode <datanode_host:ipc_port> [upgrade]]
        [-getDatanodeInfo <datanode_host:ipc_port>]
        [-metasave filename]
        [-triggerBlockReport [-incremental] <datanode_host:ipc_port>]
```

图 4-13

例如，运行 hdfs dfsadmin -report 命令，显示 HDFS 文件系统的基本信息和统计信息，如图 4-14 所示，与 HDFS 的 Web 界面一致。

```
[hdfs@node0 ~]$ hdfs dfsadmin -report
Configured Capacity: 440457404416 (410.21 GB)
Present Capacity: 408493486080 (380.44 GB)
DFS Remaining: 408483254272 (380.43 GB)
DFS Used: 10231808 (9.76 MB)
DFS Used%: 0.00%
Under replicated blocks: 15
Blocks with corrupt replicas: 0
Missing blocks: 0
Missing blocks (with replication factor 1): 0

-------------------------------------------------
Live datanodes (2):

Name: 10.10.75.101:50010 (node1)
Hostname: node1
Rack: /default
Decommission Status : Normal
Configured Capacity: 220228702208 (205.10 GB)
DFS Used: 5115904 (4.88 MB)
Non DFS Used: 0 (0 B)
DFS Remaining: 208491122688 (194.17 GB)
DFS Used%: 0.00%
DFS Remaining%: 94.67%
Configured Cache Capacity: 657457152 (627 MB)
Cache Used: 0 (0 B)
Cache Remaining: 657457152 (627 MB)
Cache Used%: 0.00%
Cache Remaining%: 100.00%
Xceivers: 2
Last contact: Sun Aug 26 20:35:10 CST 2018
```

图 4-14

4.4.4　namenode 命令

运行 namenode 命令进行格式化、升级回滚等操作，支持命令如图 4-15 所示。

```
[hdfs@node0 ~]$ hdfs namenode -help
Usage: hdfs namenode [-backup] |
        [-checkpoint] |
        [-format [-clusterid cid ] [-force] [-nonInteractive] ] |
        [-upgrade [-clusterid cid] [-renameReserved<k-v pairs>] ] |
        [-upgradeOnly [-clusterid cid] [-renameReserved<k-v pairs>] ] |
        [-rollback] |
        [-rollingUpgrade <rollback|downgrade|started> ] |
        [-finalize] |
        [-importCheckpoint] |
        [-initializeSharedEdits] |
        [-bootstrapStandby] |
        [-recover [ -force] ] |
        [-metadataVersion ]  ]
```

图 4-15

4.5　Java 编程操作 HDFS 实践

Hadoop 主要是使用 Java 语言编写实现，这里介绍 HDFS 常用 Java API 及其编程实例。Hadoop 中关于文件操作类基本上全部都是在“org.apache.hadoop.fs”包中，这些 API 能够支持的操作包括打开文件、读写文件、删除文件等。

Hadoop 编程开发环境：安装好 Java JDK1.7 和 Eclipse 开发工具，解压 Hadoop 的源文件（不是源码文件，而是编译好的安装文件），操作系统环境采用 Windows 或 Linux 均可。Hadoop 源文件在整个 Hadoop 开发过程中都会用到，因为很多依赖包都出自里面。

启动 Eclipse 工具，编写 Java 程序。为了编写一个能够与 HDFS 交互的 Java 应用程序，一般需要向 Java 工程中添加 JAR 包，如图 4-16 所示，点击“Add External JARs…”按钮，导入相应的 Hadoop 的 JAR 包，因为这些 JAR 包含了 Hadoop 的 Java API。

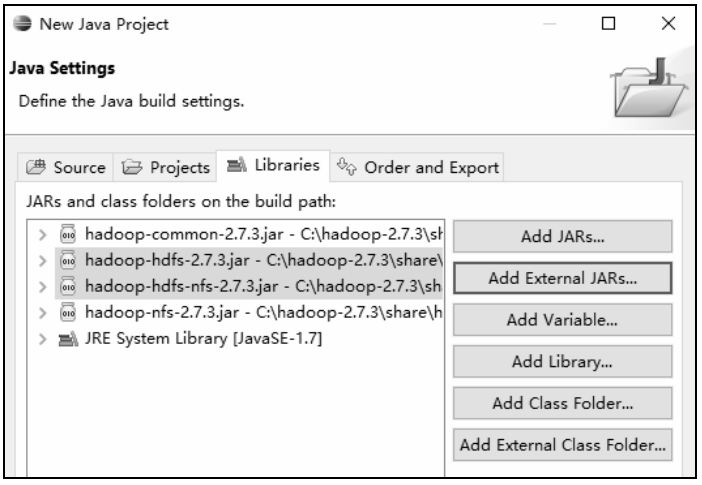


图 4-16

以下访问 HDFS 的应用程序用来检测 HDFS 文件系统 doc 目录下是否存在一个 test.txt 的文件：

```
package hadoopapi;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
public class hdfs_test {
  public static void main(String[] args) {
      try
      {
          String filename="/doc/test.txt";
      Configuration conf=new Configuration();
      conf.set("fs.defaultFS", "hdfs://node0:8020");

conf.set("fs.hdfs.impl","org.apache.hadoop.hdfs.DistributedFileSystem");

      FileSystem fs=FileSystem.get(conf);
      if(fs.exists(new Path(filename))) {
          System.out.println("this file is exist!");
      }else {
          System.out.println("this file is not exist!");
      }
      }catch(Exception e){
          e.printStackTrace();
      }
  }
}
```

编译无误，接下来把应用程序生成 JAR 包，部署到 Hadoop 大数据平台上运行。在 Eclipse 工作界面左侧的 Package Explorer 面板中，在工程项目名称上右击，在弹出的菜单中选择“Export...”，如图 4-17 所示。

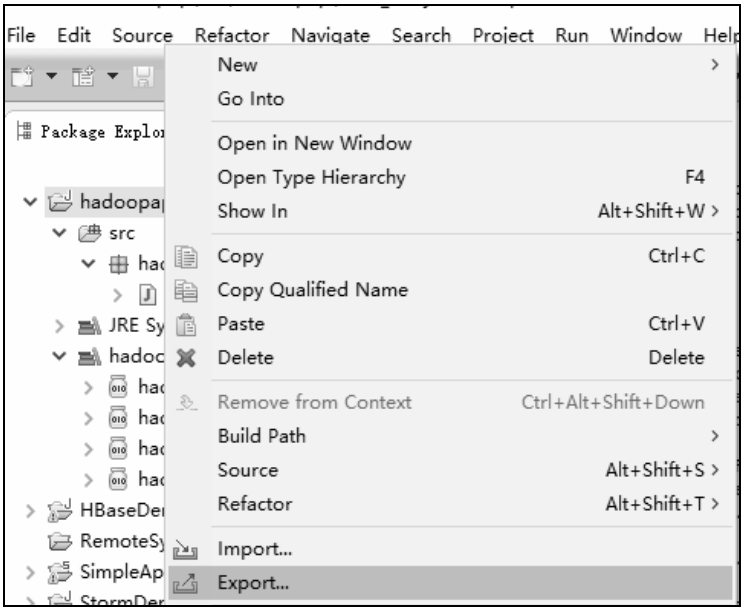


图 4-17

在弹出的“Export”对话框界面中选择 Runnable JAR file，然后点击“Next”按钮，弹出如图 4-18 所示的界面。

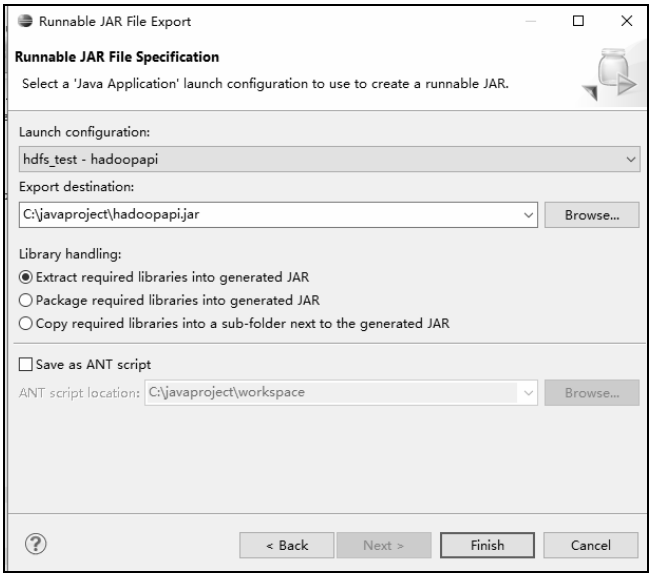


图 4-18

在该界面中 Launch configuration 用于设置生成的 JAR 包被部署启动时运行的主类，在 Export destination 中需要设置 JAR 包要输出保存到哪个目录，点击“Finish”按钮，启动打包过程。然后将打包生成的 JAR 包复制到 Hadoop 大数据平台上，使用 hadoop jar 命令运行，测试结果如图 4-19 所示。

```
[hdfs@node0 ~]$ hadoop jar /usr/local/hadoopapi.jar
this file is exist!
[hdfs@node0 ~]$ hdfs dfs -rm /doc/test.txt
18/08/26 22:37:01 INFO fs.TrashPolicyDefault: Moved: 'hdfs://node0:8020/doc/test
.txt' to trash at: hdfs://node0:8020/user/hdfs/.Trash/Current/doc/test.txt
[hdfs@node0 ~]$ hadoop jar /usr/local/hadoopapi.jar
this file is not exist!
[hdfs@node0 ~]$
```

图 4-19

HDFS 分布式文件系统很好地解决了大规模数据存储的需求，除了需要使用 shell 命令来操作 HDFS 外，使用 Eclipse 开发操作 HDFS 的 Java 应用程序也是必须掌握的技能。

4.6　HDFS 的参数配置和规划

CDH 集群建议使用 Web UI 配置界面修改参数配置。默认情况下，Hadoop 存储的副本数为 3（dfs.replication），也就是说副本数（块的备份数）默认为 3 份。如果你的集群只有两个 DataNode，就会报副本备份不足的错误，副本数调整如图 4-20 所示。因此对于 DataNode 节点，系统盘做 Raid

1、数据盘做 Raid 0。业务数据全部存储在 DataNode 上，所以 DataNode 的存储空间必须足够大，且每个 DataNode 的存储空间要尽量保持一致。

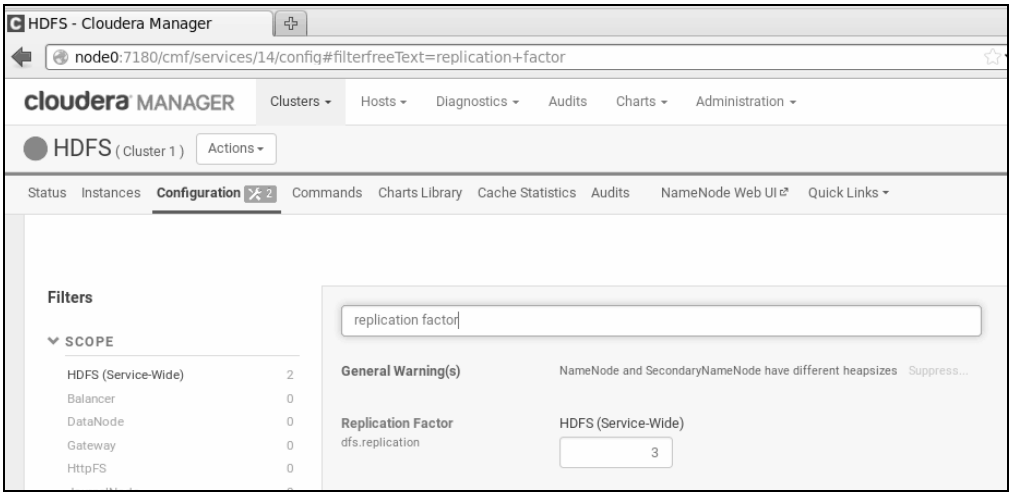


图 4-20

此外，NameNode 的 Java 堆栈大小至少要在 1GB 以上，调整参数如图 4-21 所示。如果是 128GB 内存的 NameNode 机器，建议把这个值改为 16GB 以上。

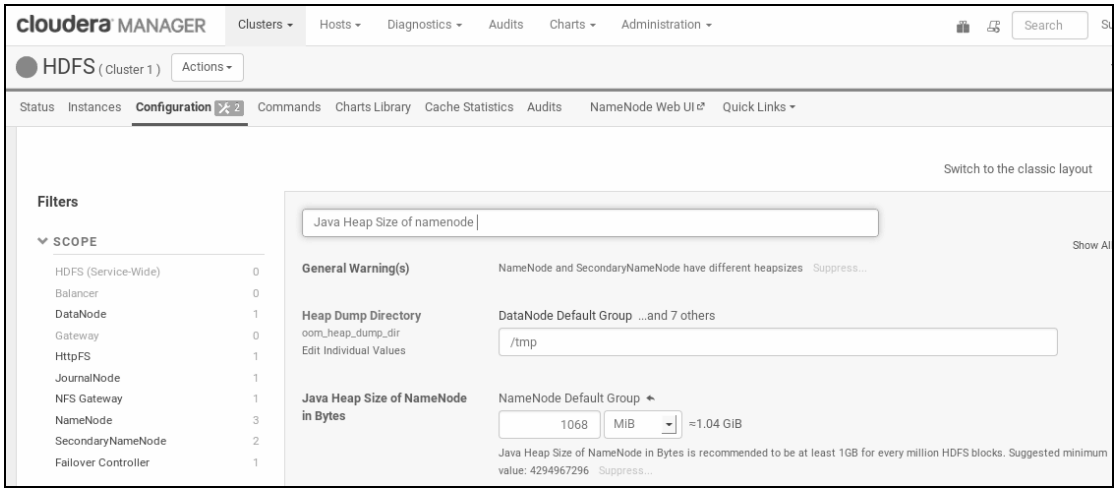


图 4-21

计算节点 DataNode 依靠的是数量优势，除了存储空间足够大之外，对机器配置要求不高。但是 NameNode 要跟所有的 DataNode 交互，接收处理各种请求，对机器配置要求较高。按以往项目测试数据来看，NameNode 存放 80GB 的元数据时，NameNode 机器建议使用 128GB 内存。

4.7　使用 Cloudera Manager 启用 HDFS HA

4.7.1　HDFS HA 高可用配置

在 HDFS 集群中 NameNode 存在单点故障，对于只有一个 NameNode 的集群，如果 NameNode 机器出现意外，将导致整个集群无法使用。为了解决 NameNode 单点故障的问题，Hadoop 给出了 HDFS 的高可用 HA 方案。HDFS 集群由两个 NameNode 组成，一个处于 Active 状态，另一个处于 Standby 状态。Active NameNode 可对外提供服务，而 Standby NameNode 则不对外提供服务，仅同步 Active NameNode 的状态，以便在 Active NameNode 失败时快速进行切换。

NameNode 之间共享数据有 NFS 和 JournalNodes 两种方案。CDH 支持 JournalNodes。两个 NameNode 为了数据同步，会通过一组称作 JournalNodes 的独立进程相互通信。当 Active 状态的 NameNode 的命名空间有任何修改时，会告知大部分的 JournalNodes 进程。Standby 状态的 NameNode 有能力读取 JournalNodes 中的变更信息，并且一直监控 EditLog 的变化，把变化应用于自己的命名空间。Standby NameNode 可以确保在集群出错时，命名空间状态已经完全同步了。

下面主要讲述如何使用 Cloudera Manager 启用 HDFS 的 HA。

步骤01 使用管理员用户 Admin 登录 Cloudera Manager 的 Web 管理界面，进入 HDFS 服务，单击“Enable High Avaiability”，如图 4-22 所示。

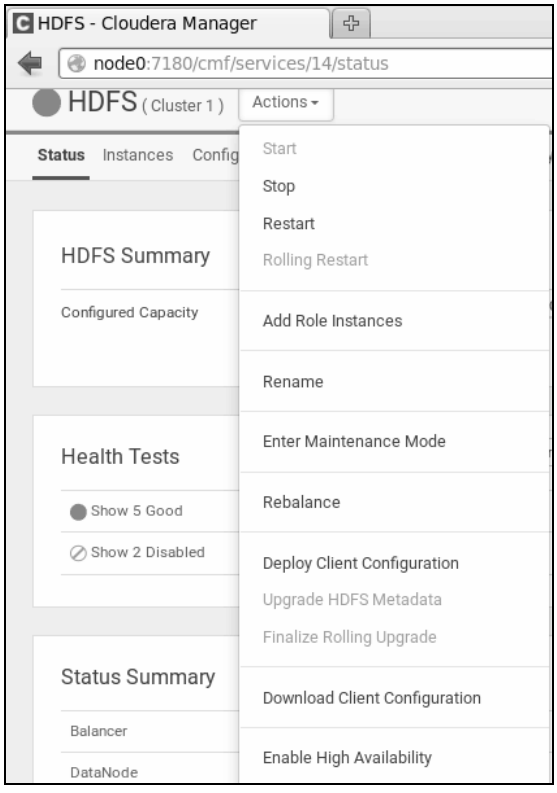


图 4-22

步骤02 设置 NameService Name，如图 4-23 所示。

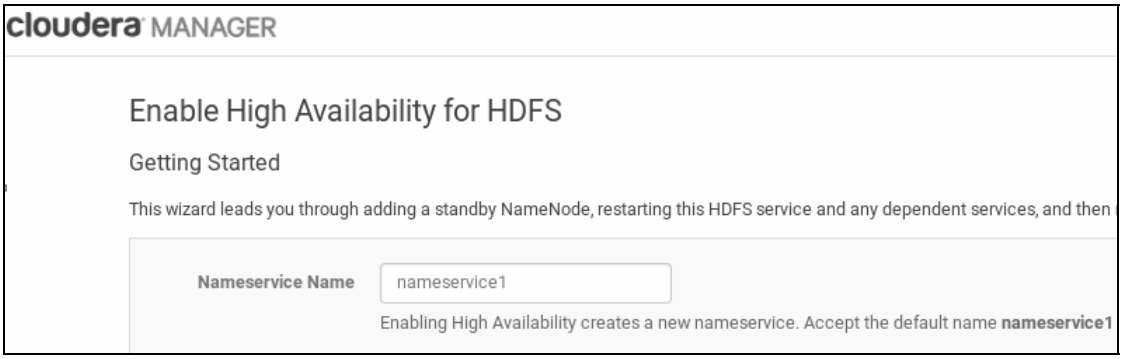


图 4-23

步骤03 选择 NameNode 主机及 JournalNode 主机，如图 4-24 所示。运行 NameNode 的服务器应该有相同的硬件配置。JournalNode 服务器上运行的 JournalNode 进程非常轻量，可以部署在其他的服务器上，但必须允许至少 3 个节点而且必须是奇数个，如 3、5、7、9 等。

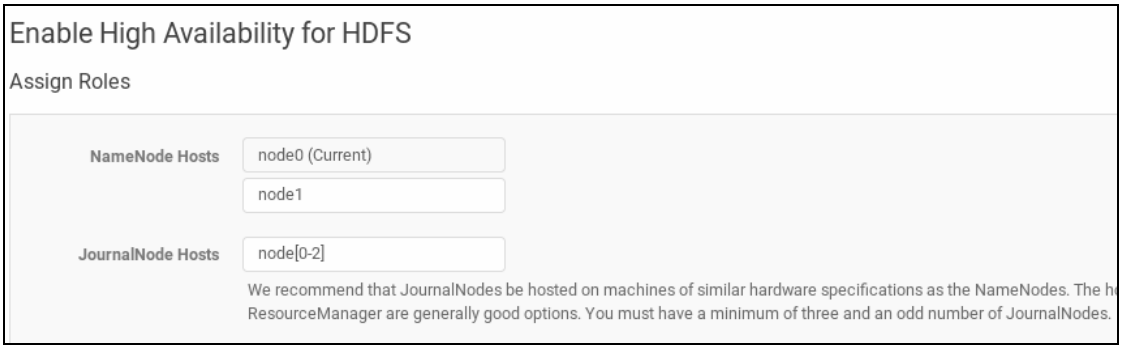


图 4-24

步骤04 设置 NameNode 的数据目录和 JournalNode 的 Edits 目录（把 NameNode 上的 namespace 元数据同步到 JournalNode 节点上），如图 4-25 所示。

步骤05 配置完毕，启用 HDFS 的 High Availability，如图 4-26 所示。如果集群已有数据，格式化 NameNode 会报错“Failed to format NameNode”，这个没有关系。

步骤06 完成 HDFS 的 High Availability ，如图 4-27 所示。屏幕提示进入 Hive 服务并停止 Hive 的所有服务，更新 Hive MetaStore NameNode，再启动 Hive 服务，完成 HiveMetastore NameNode 更新。

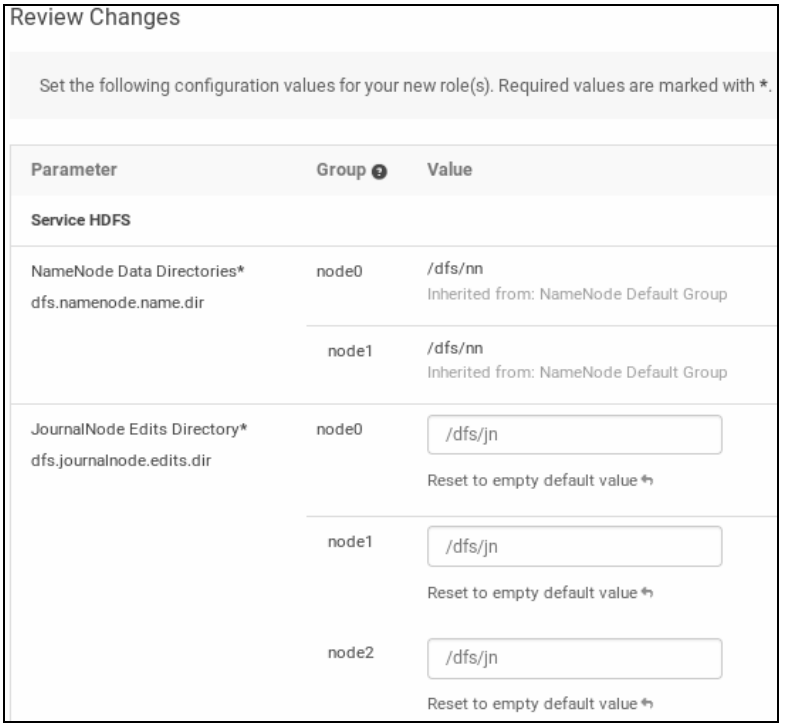

Review Changes
Set the following configuration values for your new role(s). Required values are marked with *.
Parameter
Group
Value
Service HDFS
NameNode Data Directories*
dfs.namenode.name.dir
node0
/dfs/nn
Inherited from: NameNode Default Group
node1
/dfs/nn
Inherited from: NameNode Default Group
JournalNode Edits Directory*
dfs.journalnode.edits.dir
node0
/dfs/jn
Reset to empty default value
node1
/dfs/jn
Reset to empty default value
node2
/dfs/jn
Reset to empty default value

图 4-25

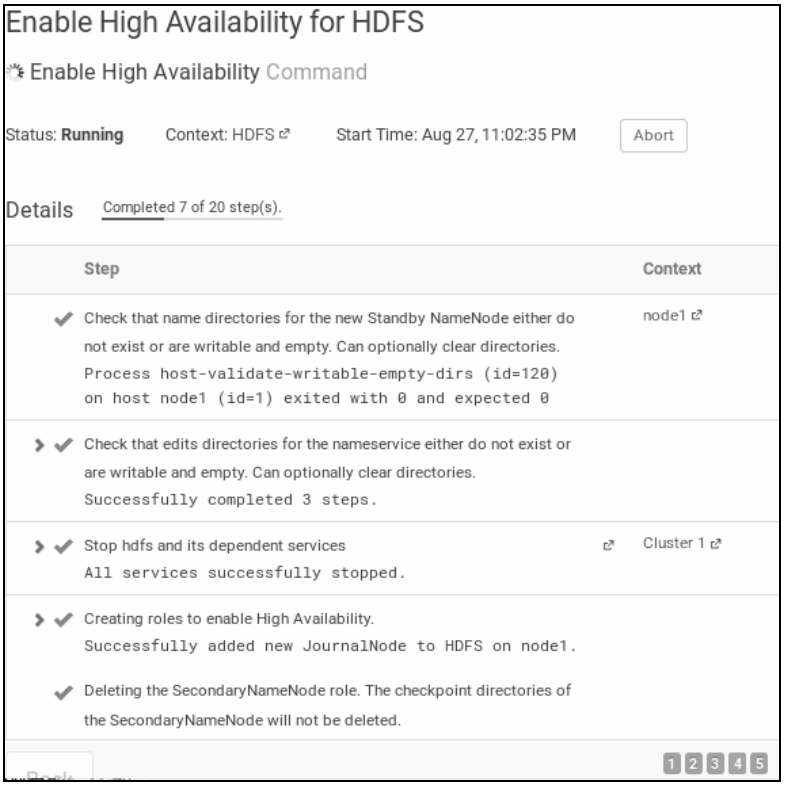

Enable High Availability for HDFS
Enable High Availability Command
Status: Running
Context: HDFS
Start Time: Aug 27, 11:02:35 PM
Abort
Details
Completed 7 of 20 step(s).
Step
Context
Check that name directories for the new Standby NameNode either do not exist or are writable and empty. Can optionally clear directories.
Process host-validate-writable-empty-dirs (id=120) on host node1 (id=1) exited with 0 and expected 0
node1
Check that edits directories for the nameservice either do not exist or are writable and empty. Can optionally clear directories.
Successfully completed 3 steps.
Stop hdfs and its dependent services
All services successfully stopped.
Cluster 1
Creating roles to enable High Availability.
Successfully added new JournalNode to HDFS on node1.
Deleting the SecondaryNameNode role. The checkpoint directories of the SecondaryNameNode will not be deleted.
1 2 3 4 5

图 4-26

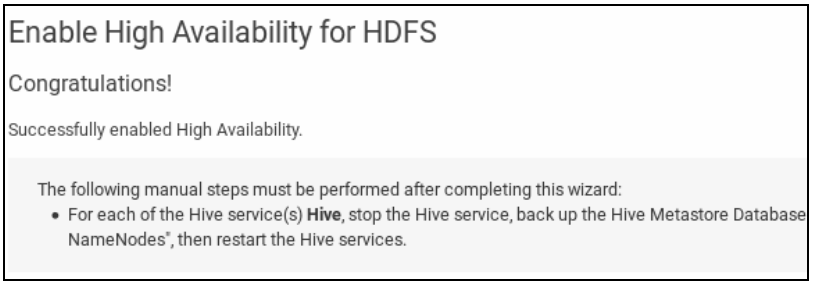


图 4-27

步骤07 HDFS 的 HA 配置成功后，通过实例列表（见图 4-28）可以看到启用 HDFS HA 后增加了 NameNode、Failover Controller 及 JouralNode 服务，并且服务都正常启动，至此已完成了 HDFS HA 的启用。

Role Type	State	Host	Commission State	Role Group
Balancer	N/A	node0	Commissioned	Balancer Default Group
DataNode	Started	node1	Commissioned	DataNode Default Group
DataNode	Started	node2	Commissioned	DataNode Group 1
DataNode	Started	node0	Commissioned	DataNode Default Group
Failover Controller	Started	node1	Commissioned	Failover Controller Default Group
Failover Controller	Started	node0	Commissioned	Failover Controller Default Group
JournalNode	Started	node1	Commissioned	JournalNode Default Group
JournalNode	Started	node2	Commissioned	JournalNode Default Group
JournalNode	Started	node0	Commissioned	JournalNode Default Group
NameNode (Standby)	Started	node1	Commissioned	NameNode Default Group
NameNode (Active)	Started	node0	Commissioned	NameNode Default Group

图 4-28

4.7.2 HDFS HA 高可用功能测试

接下来进行 HDFS HA 功能的可用性测试。Cloudera Manager 上 HDFS HA 的使用可以通过如图 4-29 所示的界面手动切换。

单击“Federation and High Availability”按钮，如图 4-30 所示，节点 node0 是 Active NameNode，节点 node1 是 Standby NameNode，单击“Actions”按钮，可以进行手动故障转移（Manual Failover）。

如图 4-31 所示，激活节点 node1，将另一个 NameNode 转换为 Standby 模式。

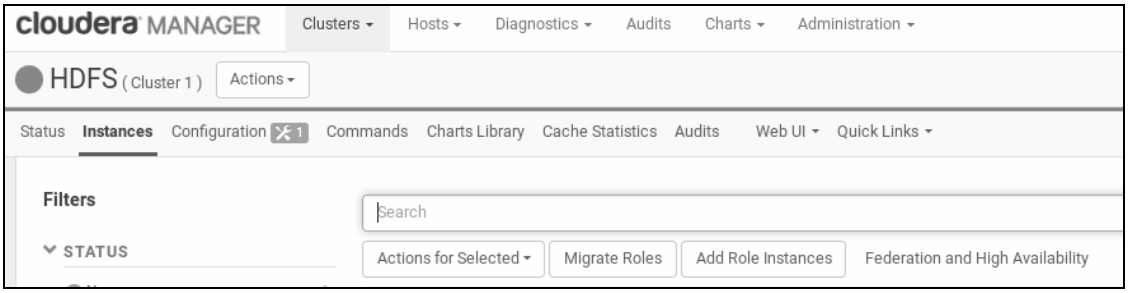


图 4-29

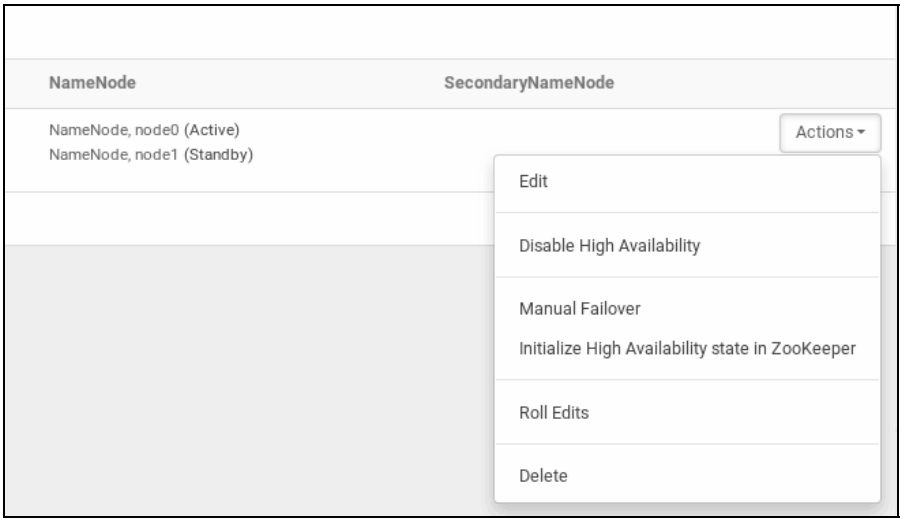


图 4-30

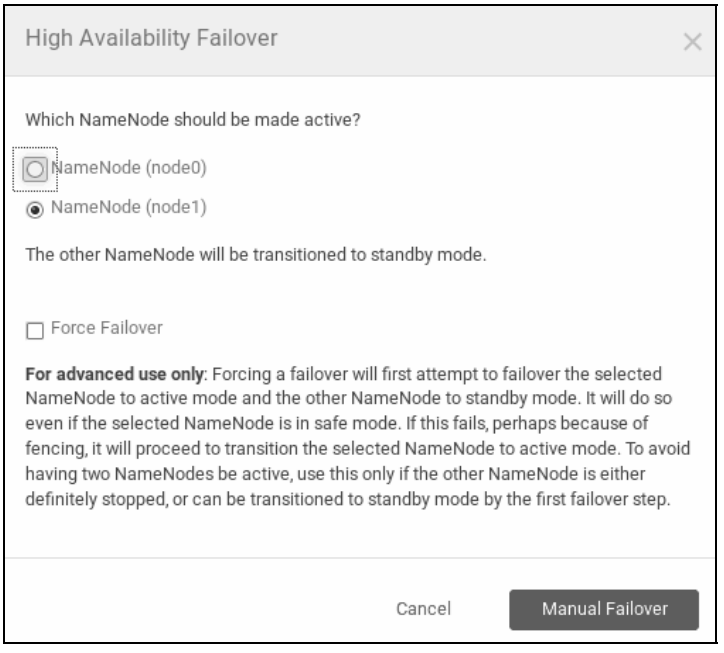


图 4-31

单击“Manual Failover”按钮，开始手动故障转移。如图 4-32 所示，故障转移成功。

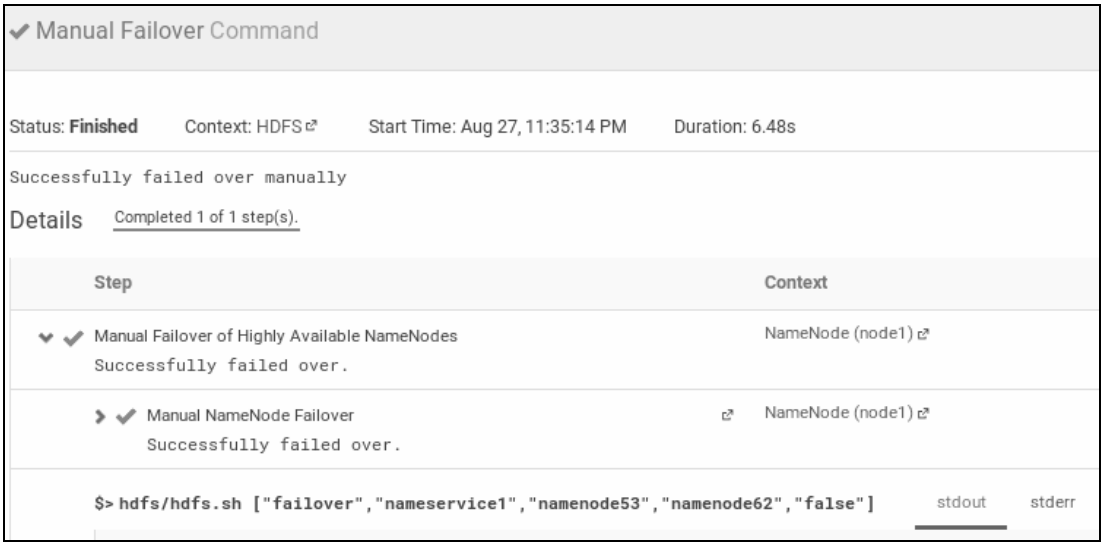


图 4-32

此时，节点 node1 已经变为 Active，节点 node0 已经变为 Standby，如图 4-33 所示。

图 4-33

另外，还可以进一步进行测试。比如使用 HDFS shell 命令 hdfs dfs -put test.tar.gz /tmp 向 HDFS 集群目录上传一个文件，上传文件的同时将 Active NameNode 服务停止，put 上传数据报错，但是 put 上传任务并没有终止。你可以查看到文件已成功上传到 HDFS 的目录，说明在 put 上传文件的过程中 Active 状态的 NameNode 停止后，会自动将 Standby 状态的 NameNode 切换为 Active 状态，未造成 HDFS 的任务终止。