国家一级出版社
全国百佳图书出版单位
China University of Mining and Technology Press

策划编辑：杨　洋
责任编辑：
封面设计：刘文东

ISBN 978-7-5646-6599-9
9 787564 665999 >

定价：55.00元

Spark 技术与应用

主编　王晓燕　袁　帅

中国矿业大学出版社
China University of Mining and Technology Press

大数据系列精品教材

中国矿业大学出版社
China University of Mining and Technology Press

大数据系列精品教材

- 内容科学严谨，紧跟行业发展
- 案例真实典型，贴近岗位需求
- 贯彻立德树人，落实课程思政
- 教学理念先进，教学资源丰富

Spark 技术与应用

主编　王晓燕　袁　帅

中国矿业大学出版社
China University of Mining and Technology Press

图书在版编目(CIP)数据

Spark 技术与应用 / 王晓燕，袁帅主编. -- 徐州：中国矿业大学出版社，2024.12. -- ISBN 978-7-5646-6599-9

Ⅰ. TP274

中国国家版本馆 CIP 数据核字第 2024G8Q822 号

书　　名　Spark 技术与应用
主　　编　王晓燕　袁　帅
责任编辑
出版发行　中国矿业大学出版社有限责任公司
（江苏省徐州市解放南路　邮编 221008）
营销热线　(0516)83885370　　83884103
出版服务　(0516)83995789　　83884920
网　　址　http://www.cumtp.com　**E-mail**：cumtpvip@cumtp.com
印　　刷　三河市骏杰印刷有限公司
开　　本　787 mm×1092 mm　　1/16　**印张**　17.25　**字数** 357 千字
版次印次　2024 年 12 月第 1 版　　2024 年 12 月第 1 次印刷
定　　价　55.00 元

PREFACE
前 言

在当今大数据时代，Apache Spark 作为一种强大的分布式计算框架，凭借其卓越的性能、易用性和广泛的适用性，在全球范围内被广泛应用于大规模数据处理、实时流处理、机器学习和图形计算等领域。本书旨在为对大数据处理技术感兴趣的学生、工程师提供一个系统且深入理解 Apache Spark 的基础知识与实践技能的学习平台。

随着企业对数据价值挖掘需求的日益增长，掌握 Spark 技术已成为现代数据工作者必备的核心技能之一。本书以实用性和前沿性相结合的原则，详细介绍了 Spark 的基本架构、核心概念、RDD（弹性分布式数据集）模型及其操作、DataFrame 与 Dataset API、Spark SQL 数据分析、Spark Streaming 实时处理以及 MLlib 与 GraphX 等高级组件的实践与应用。

全书任务设计循序渐进，从 Spark 基础环境搭建开始，逐步引领读者步入 Spark 的广阔世界。本书以任务驱动的方式，通过丰富的实例代码、详尽的操作解析以及实际项目案例电力数据统计，力求帮助读者牢固掌握 Spark 编程原理，并能够在实践中灵活运用，解决各类复杂的大数据问题。

此外，本书特别注重理论与实践并重，在每个具体的任务中，通过知识引入的方式阐述了 Spark 的工作机制及内部原理，然后再通过具体的需求点，在实践中理解知识，体验动手能力的重要性，鼓励读者在实践中深化对 Spark 的理解和应用能力。

本书共 9 个任务，以下是每个任务的介绍。

任务一为 Spark 的学习准备，搭建其开发测试环境，指导同学们下载并安装后续任务中所需要的各类软件。

任务二进行电力数据的采集，我们通过学习 Sqoop 实现批量电力数据的采集，通过学习 Flume 结合 kafka 实现实时电力数据的采集。

任务三主要介绍了 Scala 编程语言，学习并实践 Scala 语言的语法和编程知识。

任务四对 RDD 弹性分布式数据集进行了系统的介绍，包括 RDD 基础知识、

前 言 PREFACE

RDD 算子、分区和依赖、持久化和容错等知识，并利用电力数据创建 RDD，实现电力数据的分析处理。

任务五利用 Spark SQL 对数据做融合分析，并在知识引入中学习了 Spark SQL 原理，并介绍了 DataFrame、DataSet 两种数据抽象，并在任务实现中，理解 Spark SQL 的使用。

任务六我们实现了实时数据的接入和处理，学习并使用 Spark Streaming、DStream 来操作流式数据。

任务七介绍了数据可视化的相关知识，并通过 DataEase 实现了前面任务中所处理的结果数据的可视化呈现。

任务八介绍了 Spark GraphX 和 MLlib 两个高级组件的知识，并通过具体的案例理解两个组件的知识。

任务九我们总结了本书整个项目任务的建设过程，从需求分析、模型设计、需求实现到结果数据的可视化呈现。

最后，我们希望通过本书激发学生对大数据技术的热情，培养学生利用 Spark 进行高效数据处理的能力，并在未来的职业生涯中，无论是面对海量数据的批处理挑战还是实时数据分析的需求，都能游刃有余，实现个人价值与业务价值的最大化。

本书由王晓燕、袁帅担任主编。

由于编者水平有限，编写过程中难免存在不足之处，敬请广大读者批评指正。

编　者

CONTENTS
目 录

任务一
搭建 Spark 开发环境

任务二
项目数据采集

任务三

探索 Scala 编程方法

CONTENTS
目 录

任务四
揭秘弹性分布式数据集

任务五
Spark SQL——数据融合分析

任务六

Spark Streaming——实时智能分析

CONTENTS
目 录

任务七
数据可视化——让数据说话

任务八

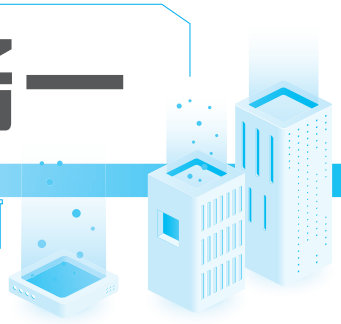
基于 Spark GraphX 与 MLlib 的智能化场景应用

任务九

综合实践——区域用电分析项目

任务一

搭建 Spark 开发环境



一、任务说明

（一）学习目标

知识目标

（1）掌握 Spark 的基础架构原理。

（2）了解 Spark 的主要组件。

（3）掌握Spark开发环境的搭建和集成开发环境（integrated development environment，IDE）的使用。

能力目标

（1）能独立完成 Spark 开发环境的搭建，包括虚拟机 Java SDK Hive 分布式环境的配置。

（2）能够运行简单的 Spark 应用程序，并通过 Shell 或编程接口将任务提交到本地或集群环境。

（3）能够独立完成全国职业院校技能大赛大数据应用开发方向环境的部署。

素质目标

（1）培养对分布式计算环境的敏感度，理解资源配置的重要性。

（2）了解国内相关技术的发展状况及国家在政策方面对大数据技术的支持。

（3）提高系统化思维和问题解决能力，能够适应技术栈变化带来的挑战。

（二）思维导图

本任务思维导图如图 1-1 所示。

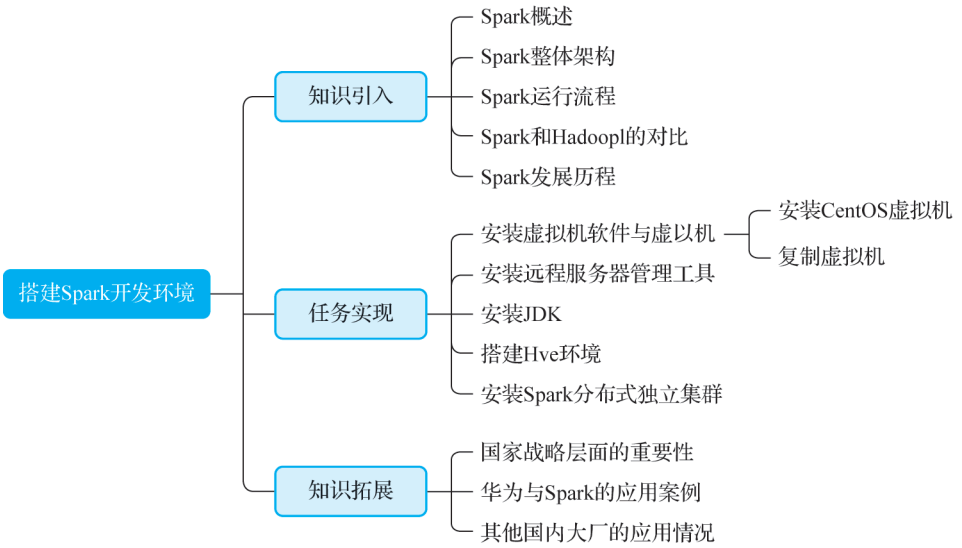


图 1-1　搭建 Spark 开发环境思维导图

二、知识引入

（一）Spark 概述

Spark 是一种通用的大数据计算框架，其目标是使用一个技术堆栈就完美地解决大数据领域的各种计算任务。Apache 官方对 Spark 的定义：通用的大数据快速处理引擎。Spark 基于内存进行计算，从而使其速度可以达到 MapReduce、Hive 的数倍甚至数十倍。

Spark 包含了大数据领域常见的各种计算框架。例如，Spark Core 用于离线计算，Spark SQL 用于交互式查询，Spark Streaming 用于实时流式计算，Spark MLlib 用于机器学习，Spark GraphX 用于图计算等。

现在已经有很多大公司在生产环境下深度地使用 Spark 作为大数据的计算框架，包括 eBay、Yahoo、百度、阿里、腾讯、网易、京东、华为等互联网公司，还有世界顶级 IT 厂商在使用 Spark，包括 IBM、Intel 等。

Spark+Hadoop 是未来大数据领域最热门、最有发展前景的组合。Spark 主要用于大数据的计算，而 Hadoop 主要用于大数据的存储（如 HDFS、Hive、HBase 等）及资源调度（YARN）。

（二）Spark 整体架构

图 1-2 所示为 Spark 架构图。

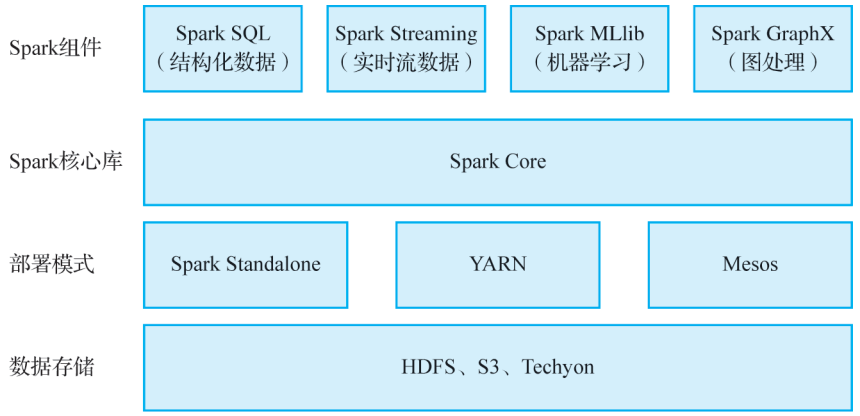


图 1-2　Spark 架构图

相关模块解释如下。

（1）Spark Core。包含 Spark 的基本功能，包含任务调度、内存管理、容错机制等，内部定义了 RDD（弹性分布式数据集）。Spark Core 提供了很多 API 来创建和操作这些 RDD，为其他组件提供底层的服务。

（2）Spark SQL。Spark 处理结构化数据的库，就像 Hive SQL、MySQL 一样，企业常用来做报表统计。

（3）Spark Streaming。实时数据流处理组件，定位类似于 Storm。Spark Streaming 提供了 API 来操作实时流数据。例如，可以从 Kafka 接收数据做实时统计。

（4）Spark MLlib。一个包含通用机器学习功能的包，Machine learning lib 包含分类、聚类、回归等机器学习算法，还支持模型评估和数据导入。MLlib 提供的这些方法均可支持集群上的横向扩展。

（5）Spark GraphX。处理图的库（如社交网络图），并进行图的并行计算。像 Spark Streaming、Spark SQL 一样，它也继承了 RDD API。它提供了各种图的操作和常用的图算法，如 PangeRank 算法。

（6）数据存储。支持 HDFS、S3、Techyon 等多种存储方式。

Spark 支持多种部署模式，可以选择任一种来部署和运行 Spark，以下为常用的三种部署模式。

① Spark Standalone。独立模式，在 Spark 自己的资源调度管理框架上运行，该框架采用 master/salve 结构。

② YARN。一种统一资源管理机制，在其上面可以运行多套计算框架。Spark 支持在 YARN 框架上运行，YARN 负责资源管理，Spark 负责任务调度和计算。

③ Mesos。Apache 下的开源分布式资源管理框架。Spark 支持在 Mesos 框架上运行，Mesos 负责资源管理，Spark 负责任务调度和计算。

（三）Spark 运行流程

Spark 运行流程图如图 1-3 所示。

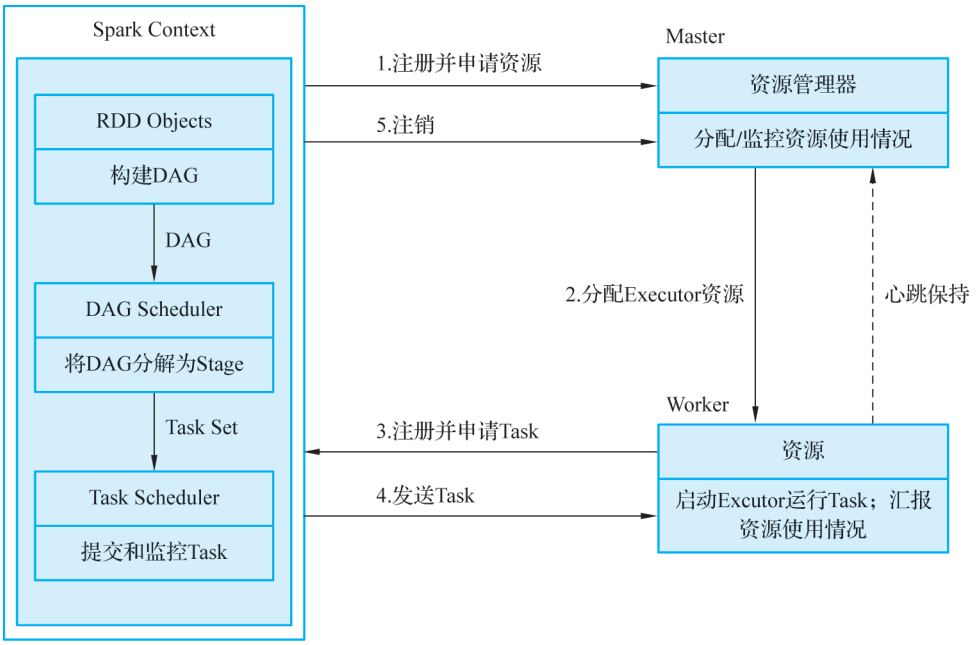


图 1-3　Spark 运行流程图

（1）构建 Spark Application 的运行环境（启动 SparkContext），SparkContext 向资源管理器（可以是 Standalone、Mesos 或 Yarn）注册并申请运行 Executor 资源。

（2）资源管理器分配 Executor 资源并启动 StandaloneExecutorBackend，Executor 运行情况将随着心跳发送到资源管理器上。

（3）SparkContext 构建成 DAG，将 DAG 分解成 Stage，并把 TaskSet 发送给 Task Scheduler。Executor 向 SparkContext 申请 Task。

（4）Task Scheduler 将 Task 发放给 Executor 运行，同时 SparkContext 将应用程序代码发放给 Executor。

（5）Task 在 Executor 上运行，运行完毕释放所有资源。

重要概念解释如下。

（1）Spark Application：表示创建的应用程序（见图 1-4）。

（2）Driver（见图 1-4）：指的是 main() 函数，作用是创建 SparkContext。通过 SparkContext 完成与资源管理器通信，申请运行资源，分配和监控任务。在程序执行完毕后需要关闭 SparkContext。

（3）Executor（见图 1-3）：Application 运行在 Worker 节点上的一个独立的 JVM 进程，在每个工作节点上会启动一个 Executor，主要用来执行 Task。一个 Executor 内可以同时执行多个 Task。

（4）Worker（见图 1-3）：集群中可以运行 Application 代码的节点。

（5）Task（见图 1-4）：在 Executor 进程中执行任务的工作单元，多个 Task 组成一个 Stage。

（6）Stage（见图 1-4）：每个 Job 会被拆分成很多组 Task，每组 TaskSet 定义为 Stage。

（7）Job（见图 1-4）：用户程序一个完整的处理流程。

（8）DAG Scheduler（见图 1-3）：根据 Job 构建基于 Stage 的 DAG（有向无环图），并提交 Stage 给 Task Scheduler。

（9）Task Scheduler（见图 1-3）：将 Stage 提交给 Worker（集群）运行，指定某个 Executor 运行具体的 Task。

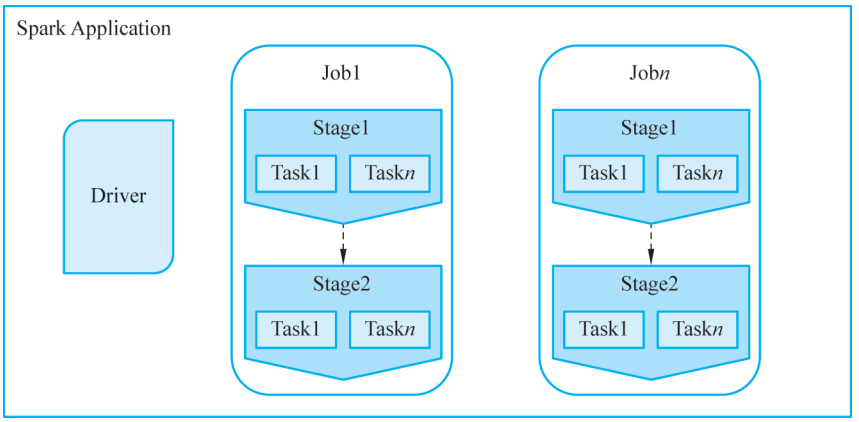


图 1-4　Spark 任务结构

（四）Spark 和 Hadoop 的对比

Hadoop 已经成了大数据技术的事实标准，Hadoop MapReduce 也非常适合对大规模数据集合进行批处理操作，但是其本身还存在一些缺陷。针对实时、快速计算需求，MapReduce 存在的延迟过高，使得需要进行多路计算和迭代算法的用例的作业过程比较低效。具体缺点如下。

（1）Hadoop MapReduce 的表达能力有限。所有计算都需要转换成 Map 和 Reduce 两个操作，不能适用于所有场景，对于复杂的数据处理过程难以描述。

（2）磁盘 I/O 开销大。Hadoop MapReduce 要求每个步骤间的数据序列化到磁盘，所以 I/O 成本很高，导致交互分析和迭代算法开销很大，而几乎所有的最优化和机器学习都是迭代的。所以 Hadoop MapReduce 不适合交互分析和机器学习。

（3）计算延迟高。如果想要完成比较复杂的工作，就必须将一系列的 MapReduce 作业串联起来然后顺序执行这些作业。每一个作业都是高时延的，而且只有在前一个作业完成之后下一个作业才能开始启动。因此，Hadoop MapReduce 不能胜任比较复杂的、多阶段的计算服务。

Spark 是借鉴了 Hadoop MapReduce 技术发展而来的，继承了其分布式并行计算的优点并改进了 MapReduce 明显的缺陷。

Spark 使用 Scala 语言进行实现，它是一种面向对象的函数式编程语言，能够像操作本地集合对象一样轻松操作分布式数据集。它具有运行速度快、易用性好、通用性强和随处运行等特点，具体优势如下。

（1）Spark 提供了内存计算，把中间结果放到内存中，带来了更高的迭代运算效率。通过支持有向无环图（DAG）的分布式并行计算的编程框架，Spark 减少了迭代过程中数据需要写入磁盘的需求，提高了处理效率。

（2）Spark 提供了一个全面、统一的框架，用于管理各种有着不同类型（文本数据、图表数据等）的数据集和数据源（批量数据或实时的流数据）的大数据处理的需求。Spark 使用函数式编程范式扩展了 MapReduce 模型以支持更多计算类型，可以涵盖广泛的工作流，这些工作流之前被实现为 Hadoop 之上的特殊系统。Spark 使用内存缓存来提升性能，因此进行交互式分析也足够快速，缓存同时提升了迭代算法的性能，这使得 Spark 非常适合数据理论任务，特别是机器学习。

（3）Spark 比 Hadoop 更加通用。Hadoop 只提供了 Map 和 Reduce 两种处理操作，而 Spark 提供的数据集操作类型更加丰富，从而可以支持更多类型的应用。Spark 的计算模式也属于 MapReduce 类型，但提供的操作不仅包括 Map 和 Reduce，还包括 Filter、FlatMap、Sample、GroupByKey、ReduceByKey、Union、Join、Cogroup、MapValues、Sort、PartionBy 等多种转换操作，以及 Count、Collect、Lookup、Save 等行为操作。

（4）Spark 基于 DAG 的任务调度执行机制比 Hadoop MapReduce 的迭代执行机制更优越。Spark 各个处理节点之间的通信模型不再像 Hadoop 一样只有 Shuffle 一种模式，程序开发者可以使用 DAG 开发复杂的多步数据管道，控制中间结果的存储、分区等。

（五）Spark 发展历程

Spark 是一种通用的大数据计算框架，使用了内存内运算技术。

2009 年，Spark 诞生于美国加州大学伯克利分校 AMP 实验室，最初是一个研究性项目。

2010 年，Spark 通过 BSD 许可协议正式对外开源发布。

2012 年，Spark 第一篇论文发布，同时发布了第一个正式版（Spark 0.6.0）。

2013 年，Spark 成为了 Apache 基金会下的项目，并发布了 Spark Streaming、Spark MLlib（机器学习）和 Shark（Spark on Hadoop）。

2014 年，Spark 成为 Apache 的顶级项目，并在 5 月底发布了 Spark 1.0.0。同时发布了 Spark GraphX（图计算）和 Spark SQL。同年，Spark 的母公司 Databricks 团队使用 Spark 刷新了数据排序世界纪录，展示了 Spark 的高效性能。

2015 年，Spark 推出了 DataFrame（大数据分析）。

至今，Spark 已成为大数据领域最活跃、最热门、最高效的大数据通用计算平台之一，广泛应用于各种行业和场景。

三、任务实现

（一）安装虚拟机软件与虚拟机

VMware Workstation 是一款功能强大的桌面虚拟计算机软件，提供用户可在单一的桌面上同时运行不同的操作系统和进行开发、测试、部署新的应用程序的最佳解决方案。

本书使用 VMware Workstation-pro-17.5，操作系统为 Windows，首先需要先下载 Windows 版软件并按照安装要求完成安装。下载链接：https://www.vmware.com/cn/products/workstation-pro/workstation -pro-evaluation.html。

Linux 是一套免费使用和自由传播的类 UNIX 操作系统，是一个基于 POSIX 和 UNIX 的多用户、多任务、支持多线程和多 CPU 的操作系统。本书采用基础的 Linux 发行版 CentOS 作为虚拟机环境部署相关软件。下载链接：https://mirrors.aliyun.com/centos/7.9.2009/isos/x86_64/。

打开网站后，选择 CentOS-7-x86_64-DVD-2207-02.iso 下载安装。本书项目实践需要采用 3 个虚拟机实现，主机名分别定义为 hadoop001、hadoop002、hadoop003。

在苹果计算机上，可以安装免费虚拟机软件 VirtualBox 来实现与 VMware Workstation 同样的功能。访问 VirtualBox 官网（https://www.virtualbox.org/），下载 dmg 格式的安装文件进行安装即可，建议选择最新版本 VirtualBox V7.0.14，考虑到网上资料丰富，本文不再赘述。虚拟机安装成功后，可以按照如下步骤继续安装虚拟机。

1. 安装 CentOS 虚拟机

（1）打开 VMware，选择 hadoop001 服务器，打开虚拟机启动界面，单击“编辑虚拟机设置”链接，如图 1-5 所示。

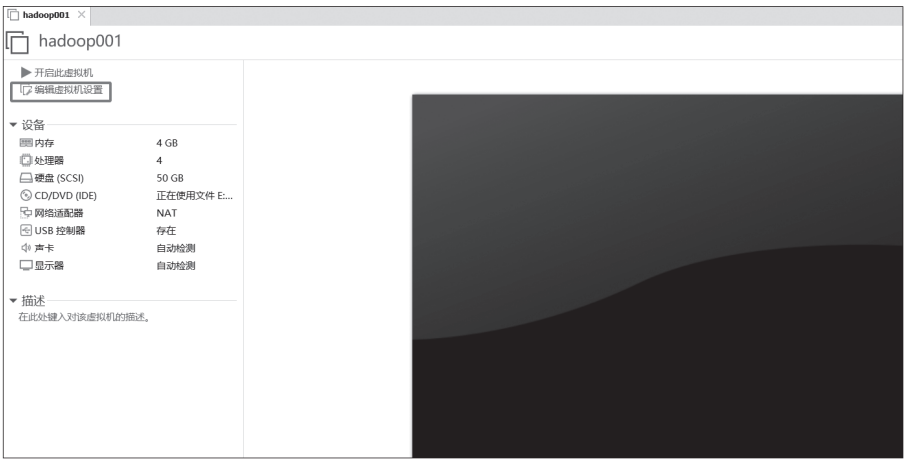


图 1-5　单击“编辑虚拟机设置”链接

（2）进入虚拟机编辑界面，选择 DVD 镜像文件，如图 1-6 所示。

图 1-6　选择 DVD 镜像文件

（3）单击“开启此虚拟机”链接，进入操作系统安装界面，选择“Install CentOS 7”选项，如图 1-7 所示。

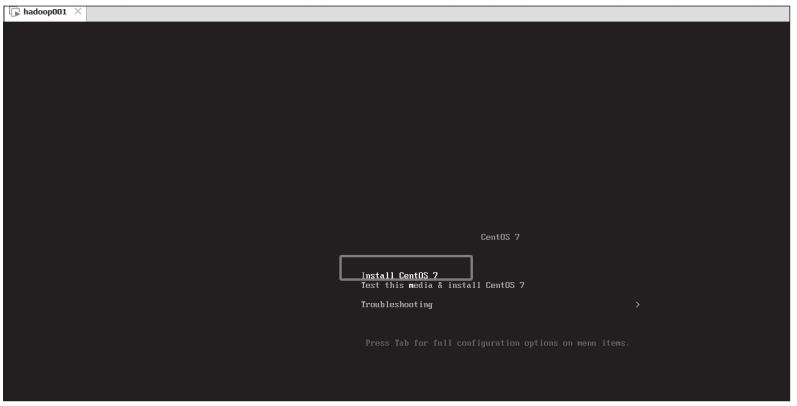


图 1-7　安装 CentOS 7

（4）进入系统安装界面，配置 CentOS 7 的语言为“简体中文（中国）”，如图 1-8 所示。

图 1-8　选择安装语言

（5）配置操作系统界面如图 1-9 所示。

图 1-9　配置操作系统界面

（6）进入“日期 & 时间”界面，配置时区，单击“完成”按钮，如图 1-10 所示。

图 1-10　配置时区

（7）进入“软件选择”界面，选择“带 GUI 的服务器”单选按钮，单击“完成”按钮，如图 1-11 所示。

图 1-11　选择“带 GUI 的服务器”单选按钮

（8）进入“安装目标位置”界面，保持默认即可，单击“完成”按钮，如图 1-12 所示。

图 1-12 “安装目标位置”界面

（9）进入“网络和主机名”界面，开启网络并修改主机名为“hadoop001”，单击“完成”按钮，如图 1-13 所示。

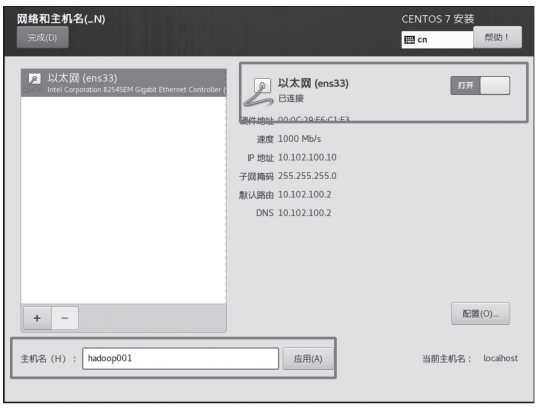


图 1-13 开启网络并修改主机名

（10）进入“ROOT 密码”界面，配置 root 账户密码。然后进入“创建用户”界面，创建用户“hadoop”，如图 1-14～图 1-16 所示。

图 1-14 配置 root 用户密码

图 1-15　选择“创建用户”选项

图 1-16　创建用户“hadoop”

（11）等待操作系统安装完成后单击“重启”按钮，如图 1-17 所示。

图 1-17　单击“重启”按钮

2. 复制虚拟机

（1）打开 VMware 软件，选中创建的虚拟机，右击并选择“管理”→“克隆”选项，克隆虚拟机，如图 1-18 所示。

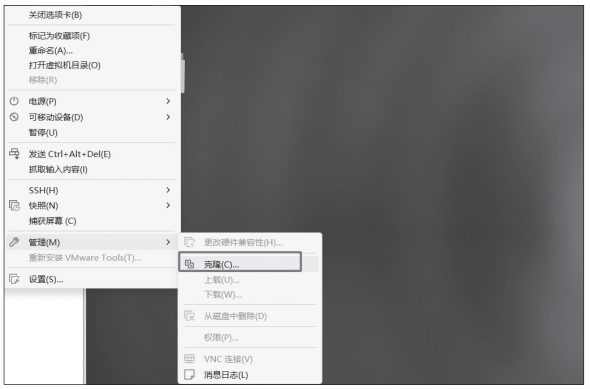


图 1-18　克隆虚拟机

（2）打开“克隆虚拟机向导”对话框，单击“下一页”按钮进入下一步操作，如图 1-19 所示。

图 1-19　“克隆虚拟机向导”对话框

（3）默认选择克隆自“虚拟机中的当前状态”，单击“下一页”按钮进入下一步操作，如图 1-20 所示。

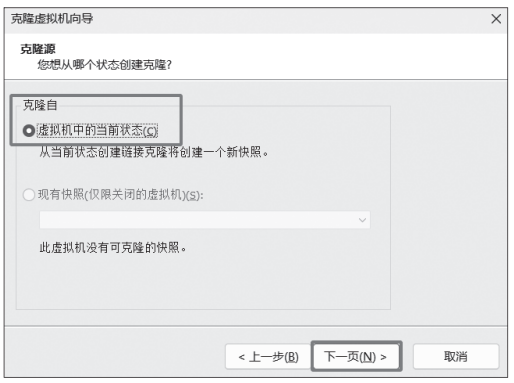


图 1-20　克隆状态选择

（4）克隆类型选择“创建完整克隆”单选按钮，单击“下一页”按钮进入下一步操作，如图 1-21 所示。

图 1-21　选择“创建完整克隆”单选按钮

（5）新虚拟机名称。设置虚拟机名称为“hadoop002”，虚拟机位置为本地存储，如图 1-22 所示。

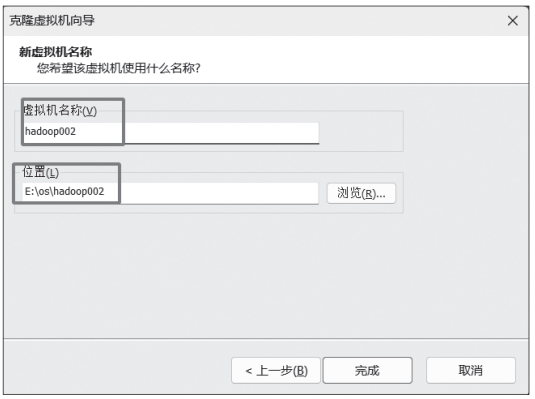


图 1-22　设置虚拟机名称及位置

（6）等待虚拟机克隆完成，如图 1-23 和图 1-24 所示。

图 1-23　正在克隆虚拟机

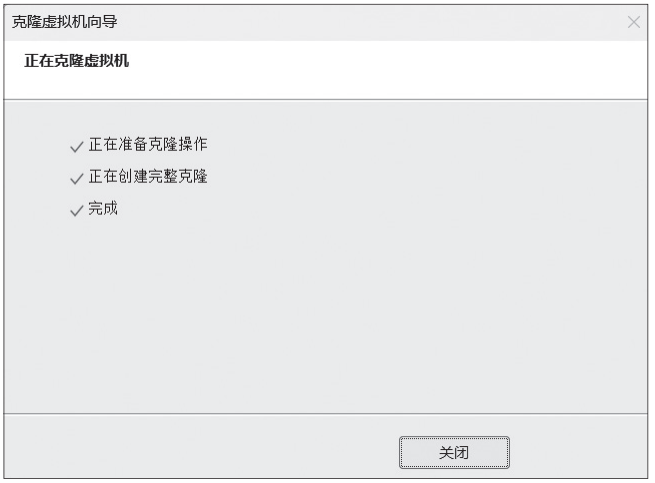


图 1-24　克隆虚拟机完成

（7）按照以上步骤再克隆一个虚拟机，如图 1-25 所示。

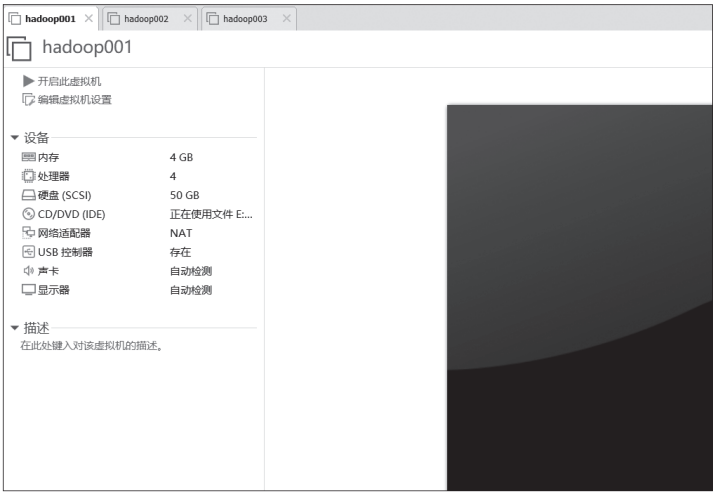


图 1-25　虚拟机创建完成

（二）安装远程服务器管理工具

远程服务器管理工具是允许用户管理远程服务器的实用工具。其通常提供图形用户界面 (GUI)，允许用户执行各种任务，例如，配置服务器、管理服务器用户、管理服务器资源和监视服务器性能。

市面上有很多远程服务器管理工具可用，选择远程服务器管理工具时需要考虑的一些因素包括所提供的特性、易用性、价格和可用的支持。

远程服务器管理有如下协议。

（1）SSH。SSH（secure shell）是一种远程管理协议，允许用户控制和修改远程服务器。SSH 通常用于登录远程服务器、执行命令和传输文件。SSH 是一种安全协议，它通过加密来保护数据不被拦截。

SSH 协议主要有以下几个版本：SSH-1、SSH-2 和 OpenSSH。SSH-1 由 Tatu Ylönen 开发，而 SSH-2 是 SSH-1 的改进版本，更安全且功能更强大。OpenSSH 是 SSH 协议的免费开源实现，也是 Linux 系统默认使用的 SSH 实现。

（2）RDP。RDP（remote desktop protocol）即远程桌面协议，是一种远程管理工具，允许用户控制和修改远程服务器。RDP 协议通常用于登录远程服务器、执行命令和传输文件。RDP 是一种私有协议，不像 SSH 那样安全。

（3）VNC。虚拟网络计算（virtual network computing，VNC）是一种远程管理协议，允许用户控制和修改远程服务器。VNC 通常用于登录远程服务器、执行命令和传输文件。VNC 不像 SSH 或 RDP 那样安全。

MobaXterm 是一款 SSH 客户端，它帮助人们在 Windows 操作系统下去连接并操作 Linux 服务器。MobaXterm 是一款增强型终端、X 服务器和 UNIX 命令集工具箱。MobaXterm 可以开启多个终端窗口，轻松地试用 UNIX/Linux 上的 GNUUNIX 命令。MobaXterm 还有很强的扩展能力，可以集成插件来运行 Gcc、Perl、Curl、Tcl/Tk/Expect 等程序。MobaXterm 分免费开源版和收费专业版，免费开源版又分便捷版（解压即用）和安装版（需要一步步安装）。

MobaXterm 支持 SSH、X11、RDP、VNC、FTP、MOSH 等连接，也支持 UNIX 命令，如 bash、ls、cat、sed、grep、awk、rsync 等。连接 SSH 终端后支持 SFTP 传输文件。

MobaXterm 的下载链接：https://mobaxterm.mobatek.net/download-home-edition，下载界面如图 1-26 所示。

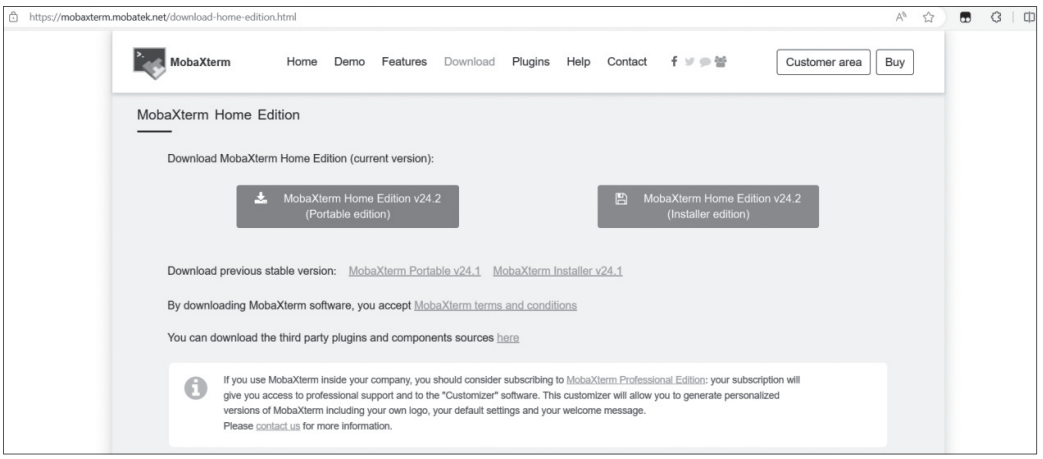


图 1-26　MobaXterm 下载界面

在 Windows 中安装 MobaXterm 工具的步骤如下。

（1）下载 Portable edition 免安装版本，如图 1-27 所示。

图 1-27　软件下载

（2）右击下载的压缩包，选择安装位置，完成解压，如图 1-28 所示。

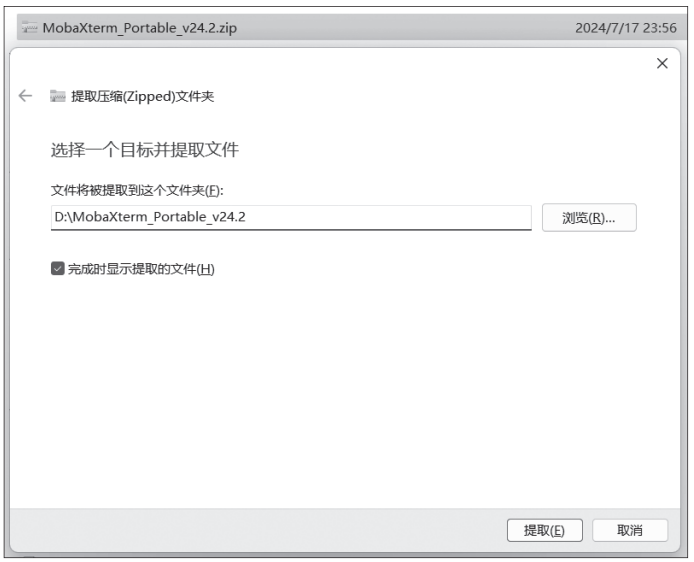


图 1-28　软件解压

（3）解压后，文件目录内有三个文件，双击“MobaXterm_Personal_24.2.exe”即可打开软件，如图 1-29 所示。

图 1-29　免安装版 MobaXterm

（4）在软件界面左侧窗口中，右击“User sessions”选项，在弹出的快捷菜单中选择“New session”选项，弹出“Session settings”对话框，再单击“SSH”按钮，在打开的界面中输入 IP 地址（Remote host）、用户名（Specify username），单击“OK”按钮，如图 1-30 所示。

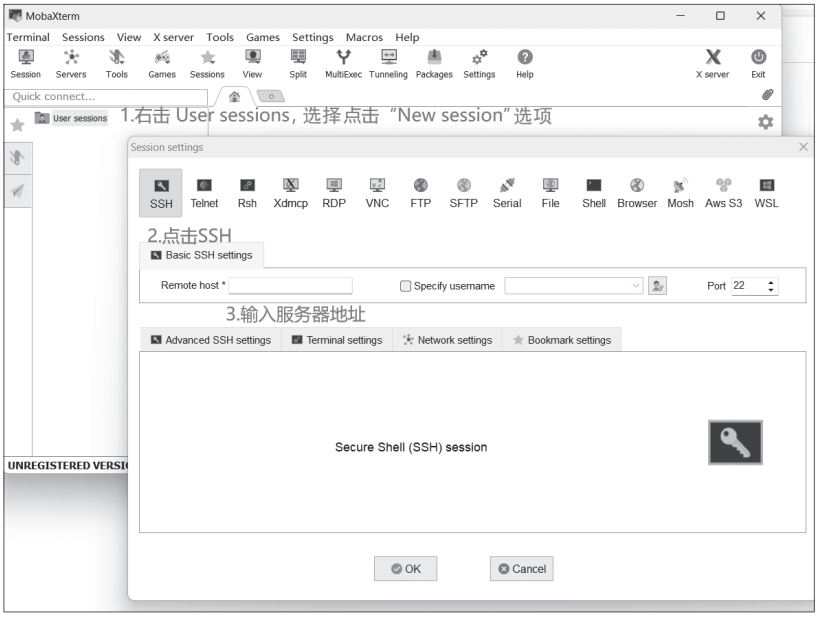


图 1-30　账号登录

（5）双击创建好的 session，输入密码即可登录服务器，并进行业务操作，如图 1-31 所示。

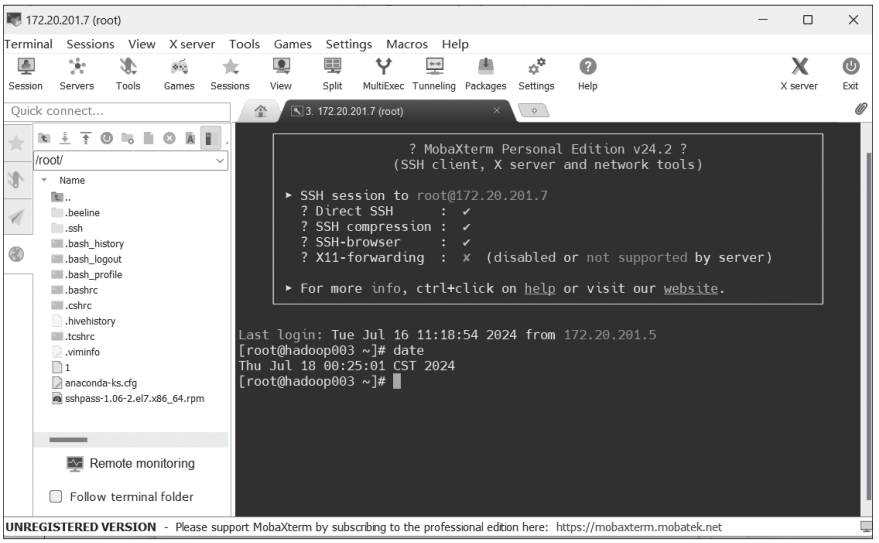


图 1-31　服务器登录

（三）安装 JDK

首先下载安装包，登录到 Oracle 官网，然后下载。下载链接：https://www.oracle.com/java/technologies/javase/javase8u211-later-arcHIVE-downloads.html。

（1）下载 JDK 1.8 安装包，将 JDK 安装包上传到 hadoop001 服务器的 /opt/software/package 目录下。

（2）解压 JDK 安装包到 /opt/software/bigdata 目录下。

```
tar -xvf jdk-8u381-linux-x64.tar.gz -C /opt/software/bigdata/jdk/
```

（3）配置环境变量，执行“vim /etc/profile”命令，将如下配置放到文件末尾，使用同样的方法在节点 hadoop002、hadoop003 上配置环境变量。

```
# JAVA_HOME
JAVA_HOME=/opt/software/bigdata/jdk/jdk1.8.0_381
JRE_HOME=${JAVA_HOME}/jre
CLASSPATH=.:${JAVA_HOME}/lib:${JRE_HOME}/lib
PATH=${JAVA_HOME}/bin:$PATH
```

（4）按“Esc”键，执行“:wq”命令，保存并退出。

（5）重新加载环境变量。

```
source /etc/profile
```

（6）验证 Java 是否安装成功。

```
java -version
```

如图 1-32 所示，出现 Java 版本号，则表示安装。

图 1-32　确认 Java 安装成功

（7）查看 jps 进程脚本。

```
#!/bin/bash
source /etc/profile
# 获取控制台指令
cmd=$*
# 判断指令是否为空
if [ ! -n "$cmd" ]
then
        echo "command can not be null !"
        exit
fi
```

```
    # 获取当前登录用户
    user='whoami'
    # 在从机执行指令，这里需要根据具体的集群情况配置，host 与具体主机名一致，同上
    for host in hadoop001 hadoop002 hadoop003
    do
            echo "================current host is $host==
===============
            echo "--> excute command \"$cmd\""
            ssh $user@$host source "/etc/profile;${cmd}"
    done
```

演示脚本功能，执行结果如图 1-33 所示。因为笔者连接的服务器已经安装了部分软件，所以能够通过脚本查看每个服务器上已经运行的 Java 程序。

```
[root@hadoop001 ~]# sh xcall.sh jps
================current host is hadoop001================
--> excute command "jps"
603560 NameNode
1011341 StandaloneSessionClusterEntrypoint
1011691 TaskManagerRunner
102532 RunJar
102660 RunJar
915361 Master
603748 DataNode
618052 ResourceManager
618724 JobHistoryServer
996898 HRegionServer
996129 HMaster
588935 QuorumPeerMain
1168 Jps
915548 Worker
618235 NodeManager
604286 DFSZKFailoverController
604054 JournalNode
================current host is hadoop002================
--> excute command "jps"
310213 DataNode
```

图 1-33　jps 脚本运行结果

（四）搭建 Hive 环境

在搭建 Hive 环境前，读者需要提前安装 Hadoop 环境和 MySQL 数据库，作为 Hive 的依赖环境。为了聚焦讲解本书重点知识，且网络资料较多，本书不再详解 Hadoop 环境和 MySQL 数据库的安装过程，读者可自行完成安装。

Hadoop 环境和 MySQL 数据库安装完成后，可继续安装 Hive 环境。

首先下载 Hive 安装包，使用浏览器访问以下链接。

https://arcHIVE.apache.org/dist/HIVE/HIVE-3.1.3/apache-HIVE-3.1.3-bin.tar.gz

（1）下载 Hive 安装包 apache-HIVE-3.1.3-bin.tar.gz，并上传到 Linux 的 /opt/software 目录下。

（2）将 Hive 安装包解压到 /opt/software/bigdata/HIVE 目录下。

```
tar -xvf apache-HIVE-3.1.3-bin.tar.gz -C /opt/software/bigdata/HIVE/
```

（3）配置环境变量，执行“vim /etc/profile”命令，将如下配置放到最后。

```
# HIVE_HOME
export HIVE_HOME=/opt/software/bigdata/HIVE/apache-HIVE-3.1.3-bin
export PATH=$PATH:$HIVE_HOME/bin
```

（4）按“Esc”键，执行“:wq”命令，保存并退出。

（5）重新加载环境变量。

```
source /etc/profile
```

（6）新建 Hive 元数据库。

```
# 登录 MySQL
MySQL -uroot -proot
# 创建 Hive 元数据库
MySQL> create database metastore;
MySQL> quit;
```

（7）将 MySQL 的 JDBC 驱动程序复制到 Hive 的 lib 目录下。

```
cp /opt/software/bigdata/MySQL/MySQL-connector-java-5.1.49/MySQL-connector-java-5.1.49.jar $HIVE_HOME/lib
```

（8）创建 HIVE-site.xml 配置文件。

①执行“vim HIVE-site.xml”命令，在打开的文件中输入以下内容。

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>
        <!-- jdbc 连接的 URL -->
    <property>
        <name>javax.jdo.option.ConnectionURL</name>
            <value>jdbc:MySQL://hadoop001:3306/metastore?useSSL=false</value>
    </property>

        <!-- jdbc 连接的 Driver-->
```

```
    <property>
        <name>javax.jdo.option.ConnectionDriverName</name>
        <value>com.MySQL.jdbc.Driver</value>
    </property>
        <!-- jdbc 连接的 username-->
    <property>
        <name>javax.jdo.option.ConnectionUserName</name>
        <value>root</value>
    </property>
        <!-- jdbc 连接的 password -->
    <property>
        <name>javax.jdo.option.ConnectionPassword</name>
        <value>root</value>
    </property>
        <!-- Hive 在 HDFS 中的默认工作目录 -->
    <property>
        <name>HIVE.metastore.warehouse.dir</name>
        <value>/user/HIVE/warehouse</value>
    </property>
        <!-- 指定 server2 连接的 host -->
    <property>
        <name>HIVE.server2.thrift.bind.host</name>
        <value>0.0.0.0</value>
    </property>
        <!-- 指定 server2 连接的端口号 -->
    <property>
        <name>HIVE.server2.thrift.port</name>
        <value>10000</value>
    </property>
        <!-- 指定 metastore 服务的地址 -->
    <property>
        <name>HIVE.metastore.uris</name>
        <value>thrift://hadoop001:9083</value>
    </property>
</configuration>
```

②修改 Hadoop 配置文件 etc/hadoop/core-site.xml，加入如下配置项。

```
<!-- 配置该 root 允许通过代理访问的主机节点 -->
<property>
```

```
    <name>hadoop.proxyuser.root.hosts</name>
    <value>*</value>
</property>
<!-- 配置该 root 允许代理的用户所属组 -->
<property>
    <name>hadoop.proxyuser.root.groups</name>
    <value>*</value>
</property>
```

保存并退出，并重启 hdfs 服务。

（9）初始化 Hive 元数据库（修改为采用 MySQL 存储元数据）。

```
cd $HIVE_HOME
bin/schematool -dbType MySQL -initSchema -verbose
```

（10）验证 Hive 功能，如图 1-34 所示。

```
# 进入 HIVE_HOME
cd $HIVE_HOME
bin/HIVE
-- 查看数据库
HIVE> show databases;
```

```
hive> show databases;
OK
default
temp
Time taken: 0.318 seconds, Fetched: 2 row(s)
```

图 1-34　查看数据库 1

```
-- 查看表，如图 1-35 所示
HIVE> show tables;
```

```
hive>  show tables;
OK
stu2
Time taken: 0.058 seconds, Fetched: 1 row(s)
hive>
```

图 1-35　查看表 1

```
-- 创建表，如图 1-36 所示
HIVE> create table stu(id int, name string);
```

```
hive> create table stu(id int, name string);
62557 [be6c287a-e151-491d-9253-781fae19fd81 main] WARN  org.apache.hadoop.hive.ql.
will be ignored, since hive.security.authorization.manager is set to instance of
OK
Time taken: 0.728 seconds
hive>
```

图 1-36　创建表 1

```
-- 插入数据，如图 1-37 所示
HIVE> insert into stu values(1,"ss");
```

```
hive> insert into stu values(1,"ss");
92244 [be6c287a-e151-491d-9253-781fae19fd81 main] WARN  org.apache.hadoop.hive.ql.D
nd may not be available in the future versions. Consider using a different executio
releases.
Query ID = root_20240122111405_946213d4-0910-4eb5-b27f-89e15d51f427
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks determined at compile time: 1
```

图 1-37　插入数据 1

```
-- 查询数据，如图 1-38 所示
HIVE> select * from stu;
```

```
hive> select * from stu;
OK
1       ss
Time taken: 0.291 seconds, Fetched: 1 row(s)
```

图 1-38　查询数据 1

（11）启动 metastore 和 HIVEserver2 服务。

```
cd $HIVE_HOME
# 启动 metastore 服务
bin/HIVE --service metastore
# 启动 HIVEServer2 服务
bin/HIVE --service HIVEserver2
```

（12）使用 beeline 连接 Hive。

```
[root@hadoop001 ~]# beeline -u jdbc:HIVE2://hadoop001:10000/
default -n root
-- 查看数据库，如图 1-39 所示
jdbc:HIVE2://hadoop001:10000/default>  show databases;
```

```
0: jdbc:hive2://hadoop001:10000/default>  show databases;
+----------------+
| database_name  |
+----------------+
| default        |
| temp           |
+----------------+
2 rows selected (0.14 seconds)
```

图 1-39　查看数据库 2

```
-- 查看表，如图 1-40 所示
HIVE> show tables;
```

```
0: jdbc:hive2://hadoop001:10000/default> show tables;
+-----------+
| tab_name  |
+-----------+
| stu       |
| stu2      |
+-----------+
2 rows selected (0.271 seconds)
```

图 1-40　查看表 2

```
-- 创建表，如图 1-41 所示
jdbc:HIVE2://hadoop001:10000/default>  create table stu2(id int, name string);
```

```
0: jdbc:hive2://hadoop001:10000/default>  create table stu2(id int, name string);
No rows affected (1.209 seconds)
0: jdbc:hive2://hadoop001:10000/default>
```

图 1-41　创建表 2

```
-- 插入数据，如图 1-42 所示
jdbc:HIVE2://hadoop001:10000/default> insert into stu2 values(1,"ss");
```

```
0: jdbc:hive2://hadoop001:10000/default> insert into stu2 values(1,"ss");
No rows affected (27.926 seconds)
```

图 1-42　插入数据 2

```
-- 查询数据，如图 1-43 所示
jdbc:HIVE2://hadoop001:10000/default> select * from stu2;
```

```
0: jdbc:hive2://hadoop001:10000/default> select * from stu2;
+----------+------------+
| stu2.id  | stu2.name  |
+----------+------------+
| 1        | ss         |
+----------+------------+
1 row selected (0.394 seconds)
```

图 1-43　查询数据 2

（五）安装 Spark 分布式独立集群

首先下载安装包，使用浏览器访问以下链接。

https://arcHIVE.apache.org/dist/Spark/Spark-3.2.3/Spark-3.2.3-bin-hadoop3.2.tgz

（1）下载 Spark 安装包 Spark-3.2.3-bin-hadoop3.2.tgz，并上传到 hadoop001 服务器的 /opt/software/package 目录下。

（2）将安装包解压到 /opt/software/bigdata/Spark 目录下。

```
[root@hadoop001 package]# tar -xvf Spark-3.2.3-bin-hadoop3.2.tgz
-C /opt/software/bigdata/Spark/
```

（3）配置环境变量，执行“vim /etc/profile”命令，将如下配置放到文件末尾，其他节点 hadoop002、hadoop003 的配置环境变量。

```
# Spark_HOME
export  Spark_HOME=/opt/software/bigdata/Spark/Spark-3.2.3-
bin-hadoop3.2
export PATH=$PATH:$Spark_HOME/bin
```

（4）按“Esc”键，执行“:wq”命令，保存并退出。

（5）重新加载环境变量。

```
[root@hadoop001]# source /etc/profile
```

（6）修改 workers 配置文件，保存并退出，集群为 3 个节点。

```
[root@hadoop001]# cd $Spark_HOME/conf
[root@hadoop001 conf]# vim workers
hadoop001
hadoop002
hadoop003
```

（7）修改 Spark-defaults.conf 文件。

```
[root@hadoop001]# cd $Spark_HOME/conf
[root@hadoop001 conf]# vim Spark-defaults.conf
# 开启记录事件日志的功能
Spark.eventLog.enabled           true
# 设置事件日志存储的目录
Spark.eventLog.dir               hdfs://nameservice/Spark/log
# 设置 HistoryServer 加载事件日志的位置
Spark.yarn.historyServer.address  http://hadoop001:19888/
jobhistory/logs
# 设置 HistoryServer UI 端口
Spark.history.ui.port 18080
```

（8）修改 Spark-env.sh 文件。

```
[root@hadoop001 ~]# cd $Spark_HOME/conf
[root@hadoop001 conf]# vim Spark-env.sh
# 配置 JAVA_HOME
export JAVA_HOME=/opt/software/bigdata/jdk/jdk1.8.0_381
# 配置 Spark Master 节点 IP
Spark_MASTER_HOST=hadoop001
# 配置 Spark Master 端口
Spark_MASTER_PORT=7077
# 配置 HADOOP_HOME
export HADOOP_HOME=/usr/local/ha/hadoop-3.3.5/
# 配置 Spark Master UI 端口
Spark_MASTER_WEBUI_PORT=8989
# 配置 Spark WORKER UI 端口
Spark_WORKER_WEBUI_PORT=8020
# 配置 HADOOP_HOME
export HADOOP_HOME=/opt/software/bigdata/hadoop/hadoop-3.3.4
# 配置 Hadoop 配置文件路径
export HADOOP_CONF_DIR=/opt/software/bigdata/hadoop/
hadoop-3.3.4/etc/hadoop
# 配置 YARN 配置文件路径
export YARN_CONF_DIR=/opt/software/bigdata/hadoop/
hadoop-3.3.4/etc/hadoop
# 设置通用 JVM 参数
export Spark_DAEMON_JAVA_OPTS="
-DSpark.deploy.recoveryMode=ZOOKEEPER
-DSpark.deploy.zookeeper.url=hadoop001,hadoop002,hadoop003
-DSpark.deploy.zookeeper.dir=/Spark"
# 配置 Hadoop 目录中的 CLASSPATH
export Spark_DIST_CLASSPATH=$(/opt/software/bigdata/hadoop/
hadoop-3.3.4/bin/hadoop classpath)
# 设置 Spark 日志配置
export Spark_HISTORY_OPTS="
-DSpark.history.ui.port=18080
-DSpark.history.fs.logDirectory=hdfs://nameservice/Spark/log
-DSpark.history.retainedApplications=30"
```

（9）分发 Spark 文件到其他两个节点。

```
[root@hadoop001 ~]# cd /opt/software/bigdata/
[root@hadoop001 bigdata]# scp -r Spark/ hadoop002:/opt/
software/bigdata/
```

```
[root@hadoop001 bigdata]# scp -r Spark/ hadoop003:/opt/software/bigdata/
```

（10）启动 Spark 集群。启动 hadoop001 上的 master 和所有的 slave 节点。

```
[root@hadoop001 bigdata]# sh $Spark_HOME/sbin/start-all.sh
```

在 hadoop001 上执行 jps 进程可以查看到 master 和 worker 进程是否启动，如图 1-44 所示。

```
sh xcall.sh "jps -ml |grep Spark"
```

```
[root@hadoop001 ~]# sh xcall.sh "jps -ml |grep spark"
================current host is hadoop001=================
--> excute command "jps -ml |grep spark"
17032 org.apache.spark.deploy.worker.Worker --webui-port 8020 spark://hadoop001:7077
16846 org.apache.spark.deploy.master.Master --host hadoop001 --port 7077 --webui-port 8989
================current host is hadoop002=================
--> excute command "jps -ml |grep spark"
612299 org.apache.spark.deploy.worker.Worker --webui-port 8020 spark://hadoop001:7077
================current host is hadoop003=================
--> excute command "jps -ml |grep spark"
768261 org.apache.spark.deploy.worker.Worker --webui-port 8020 spark://hadoop001:7077
[root@hadoop001 ~]#
```

图 1-44　查看 master/worker 是否启动

（11）分别在 hadoop002 上启动 standby master。

```
[root@hadoop002 ~]# sh $Spark_HOME/sbin/start-master.sh
```

执行 jps 进程可以查看 hadoop002 节点 master 是否启动，如图 1-45 所示。

```
[root@hadoop002 ~]# jps -ml| grep spark
364045 org.apache.spark.deploy.worker.Worker --webui-port 8020 spark://hadoop001:7077
364247 org.apache.spark.deploy.master.Master --host hadoop002 --port 7077 --webui-port 8989
[root@hadoop002 ~]#
```

图 1-45　查看 master 是否启动

（12）查看 Web 界面，登录到 hadoop001、hadoop002 的 UI 界面，如图 1-46 和图 1-47 所示。

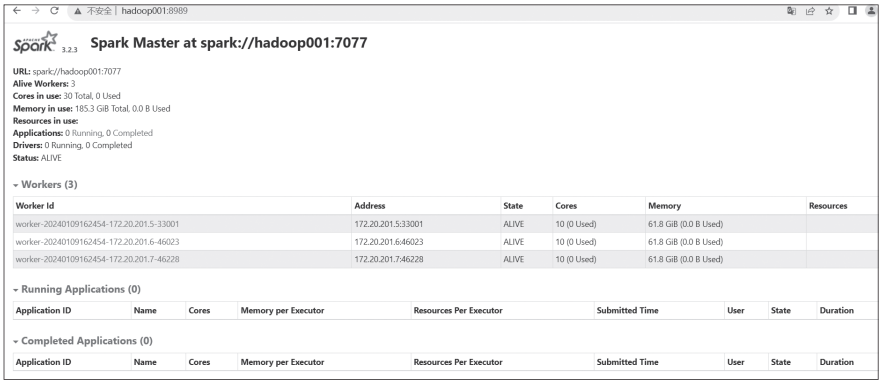


图 1-46　hadoop001 节点监控界面

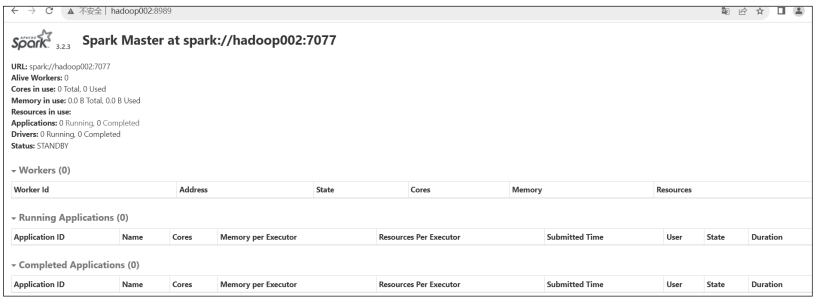


图 1-47　hadoop002 节点监控界面

（13）验证 Spark 功能，执行 Spark PI。这里运行官方示例“计算 PI 的值”，将计算任务提交到 Spark 集群。

```
[root@hadoop001 ~]# Spark-submit --class org.apache.Spark.examples.SparkPi \
--master Spark://hadoop001:7077,hadoop002:7077 \
--executor-memory 512MB \
--total-executor-cores 4 \
/opt/software/bigdata/Spark/Spark-3.2.3-bin-hadoop3.2/examples/jars/Spark-examples_2.12-3.2.3.jar 10
```

代码说明如下。

--master：master 的地址，指提交任务到哪里执行，例如，Spark://host:port, yarn, local。

--executor-memory：每个 executor 的内存，默认是 1 GB。

--total-executor-cores：所有 executor 的核数。仅在 mesos 或 standalone 植式下使用。

运行结果如图 1-48 所示，web-UI 监控界面如图 1-49 所示。

```
24/01/22 14:37:49 INFO TaskSchedulerImpl: Killing all running tasks in stage 0: Stage finished
24/01/22 14:37:49 INFO DAGScheduler: Job 0 finished: reduce at SparkPi.scala:38, took 2.740273 s
Pi is roughly 3.1435471435471434
24/01/22 14:37:49 INFO SparkUI: Stopped Spark web UI at http://hadoop001:4040
24/01/22 14:37:49 INFO StandaloneSchedulerBackend: Shutting down all executors
```

图 1-48　计算 PI 值

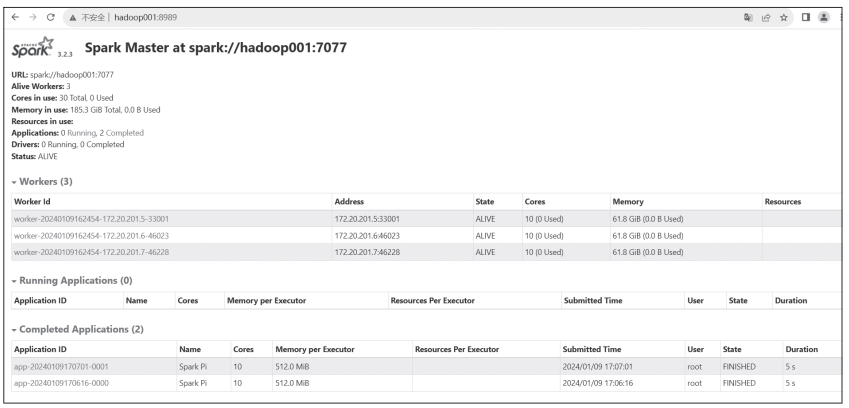


图 1-49　运行任务的 Web-UI

（14）Spark-SQL 对接 Hive。将 HIVE-site.xml 文件复制到 Spark_HOME/confvnf 目录下，然后重启 Spark 集群即可，如图 1-50 所示。

```
[root@hadoop001 ~]# cp $HIVE_HOME/conf/HIVE-site.xml $Spark_HOME/conf
```

```
[root@hadoop001 ~]# ll $SPARK_HOME/conf
total 40
-rw-r--r-- 1  501 sudo 1105 Nov 15  2022 fairscheduler.xml.template
-rw-r--r-- 1 root root 1351 Dec 26 15:06 hive-site.xml
-rw-r--r-- 1  501 sudo 2471 Nov 15  2022 log4j.properties
-rw-r--r-- 1  501 sudo 9141 Nov 15  2022 metrics.properties.template
-rw-r--r-- 1  501 sudo 1494 Dec 26 14:11 spark-defaults.conf
-rwxr-xr-x 1  501 sudo 5280 Dec 26 14:19 spark-env.sh
-rw-r--r-- 1 root root  887 Dec 26 14:06 workers
[root@hadoop001 ~]#
```

图 1-50　对接 Hive

（15）验证 Spark-SQL。

```
[root@hadoop001 ~]# Spark-sql \
--master Spark://hadoop001:7077,hadoop002:7077 \
--executor-memory 512MB \
--num-executors 4
-- 查看数据库，如图 1-51 所示
Spark-sql> show databases;
```

```
spark-sql> show databases;
default
temp
Time taken: 4.009 seconds, Fetched 2 row(s)
spark-sql>
```

图 1-51　查看数据库 3

```
-- 进入 temp 数据库，如图 1-52 所示
Spark-sql> use temp;
```

```
spark-sql> use temp;
Time taken: 0.139 seconds
spark-sql>
```

图 1-52　进入 temp 数据库

```
-- 查看表，如图 1-53 所示
Spark-sql> show tables;
```

```
spark-sql> show tables;
stu
test1
Time taken: 0.157 seconds, Fetched 2 row(s)
spark-sql>
```

图 1-53　查看表 3

```
-- 插入数据，如图 1-54 所示
Spark-sql> insert into stu2 values(2,"tom");
```

```
spark-sql> insert into stu values(2,"tom");
24/01/22 14:44:48 WARN SessionState: METASTORE_FILTER_HOOK will be ignored, since hive
instance of HiveAuthorizerFactory.
Time taken: 5.584 seconds
spark-sql>
```

图 1-54　插入数据 3

```
-- 查询数据，如图 1-55 所示
Spark-sql> select * from stu;
```

```
spark-sql>  select * from stu;
1       ss
102     lisi
103     lisi1
103     lisi1
103     tom
100     张三
2       tom
Time taken: 3.987 seconds, Fetched 7 row(s)
```

图 1-55　查询数据 3

（16）停止集群方法（在 hadoop001 上执行）。

```
[root@hadoop001 bigdata]# sh $Spark_HOME/sbin/stop-all.sh
```

在 hadoop001 上执行 jps 进程可以查看 mater 和 worker 进程是否停止，如图 1-56 所示。

```
sh xcall.sh "jps -ml |grep Spark"
```

```
[root@hadoop001 ~]#
[root@hadoop001 ~]# sh xcall.sh "jps -ml |grep spark"
================current host is hadoop001================
--> excute command "jps -ml |grep spark"
================current host is hadoop002================
--> excute command "jps -ml |grep spark"
364247 org.apache.spark.deploy.master.Master --host hadoop002 --port 7077 --webui-port 8989
================current host is hadoop003================
--> excute command "jps -ml |grep spark"
[root@hadoop001 ~]#
```

图 1-56　查看进程 1