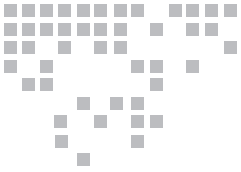
第一部分 Part 1

准备工作

■ 第 1 章　基础环境准备



Chapter 1 第 1 章

基础环境准备

1.1 软件环境准备

软件版本选择：

操作系统：CentOS 6.6 版本；JDK：1.7 版本；Maven：3.2 版本；Scala：2.10 版本。

所有软件安装目录：/data/soft。

确定了软件版本后，我们将具体介绍软件的安装，本节主要介绍基础的软件安装方式。

1. JDK 安装

JDK 是 Java Development Kit 的简称，为 Java 语言开发的程序提供开发工具包和运行环境。JDK 安装的步骤如下：

（1）下载 JDK 二进制安装包

```
wget http://download.oracle.com/otn-pub/java/jdk/7u15-b03/jdk-7u15-linux-x64.tar.gz
```

（2）解压安装

```
tar -zxvf jdk-7u15-linux-x64.tar.gz
```

（3）创建软连接

软连接相当于快捷方式，便于后续版本更新升级。

```
ls -s /data/soft/jdk-7u15-linux-x64 /usr/local/jdk
```

（4）配置环境变量

```
vim /etc/profile
```

```
export JAVA_HOME=/usr/local/jdk
export JRE_HOME=$JAVA_HOME/jre
exportCLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar
                :$JRE_HOME/lib:$CLASSPATH
export PATH=$PATH: $JAVA_HOME/bin
```

刷新环境变量使其生效：source /etc/profile

（5）验证安装是否成功

查看 JDK 版本命令：java -version

2. Maven 安装

Maven 是 Apache 开源的一个目前比较流行的项目管理和整合工具，能够自动完成项目的构建，并根据配置文件自动下载依赖组件，提供代码编译、打包、发布等功能。下面介绍 Maven 的详细安装过程。

Maven 安装的步骤如下：

（1）下载 Maven 二进制安装包

```
wget http://mirror.bit.edu.cn/apache/maven/maven-3/3.3.9/binaries/
apache-maven-3.3.9-bin.tar.gz
```

（2）解压安装

```
tar -zxvf apache-maven-3.3.9-bin.tar.gz
```

（3）创建软连接

软连接相当于快捷方式，便于后续版本更新升级。

```
ls -s/data/soft/apache-maven-3.3.9-bin /usr/local/maven
```

（4）配置环境变量

```
vim /etc/profile
export M2_HOME=/usr/local/maven
export PATH=$PATH: $JAVA_HOME/bin:$M2_HOM/bin
```

刷新环境变量使其生效：source /etc/profile

（5）验证安装是否成功

查看 Maven 版本命令：mvn -version

3. Scala 安装

Scala 编程语言是一种面向对象的函数式编程语言，充分展现了函数式编程语言简约、高效的特点，在程序开发的过程中可以引入 Java 语言，可扩展性强。由于 Scala 具有很多优秀的特性，越来越多的开源项目使用 Scala 语言开发，比如 Spark、Kafka 等。下面详细介绍 Scala 开发环境的安装过程。

Scala 安装的步骤如下：

（1）下载 JDK 二进制安装包

```
wget http://downloads.lightbend.com/scala/2.10.6/scala-2.10.6.tgz
```

（2）解压安装

```
tar -zxvf scala-2.10.6.tgz
```

（3）创建软连接

软连接相当于快捷方式，便于后续版本更新升级。

```
ls -s /data/soft/scala-2.10.6 /usr/local/scala
```

（4）配置环境变量

```
vim /etc/profile
export SCALA_HOME=/usr/local/scala
export PATH=$PATH: $JAVA_HOME/bin:$M2_HOM/bin:$SCALA_HOME/bin
```

刷新环境变量使其生效：`source /etc/profile`

（5）验证安装是否成功

查看 scala 版本命令：`scala-version`

1.2 集群环境准备

1.2.1 Zookeeper 集群部署

Zookeeper 是大数据系统中常用的分布式框架，主要用于公共配置管理、集群资源一致性管理、状态管理、部分分布式系统 Leader 选举等，下面通过完全分布式搭建方式进行介绍。

1. 集群规划

由于 Zookeeper 采用 FastLeaderElection 算法选举 Leader，集群中过半的机器正常运行才能够成功选举 Leader，为保证集群正常运行，集群部署的节点数为奇数个，最少节点个数为 3，生产环境建议部署 5 个以上的奇数个节点，因为 3 个实例其中只要有一个实例不可用，整个 Zookeeper 集群将无法成功选举，仍然不可以提供服务。

2. 部署过程

本例将以三个节点的部署为例，分别在 192.168.1.1、192.168.1.2、192.168.1.3 三台服务器部署一个 Zookeeper 实例。详细部署过程如下：

（1）下载安装包并解压

```
wget http://apache.fayea.com/zookeeper/zookeeper-3.4.6/zookeeper-3.4.6.tar.gz
```

解压到 /data/soft 目录下：

```
tar -zxvf http://apache.fayea.com/zookeeper/zookeeper-3.4.6/zookeeper-3.4.6.tar.gz
```

```
-C /data/soft
```

（2）创建软连接

创建软连接便于以后升级版本，方便统一管理。

```
ls -s /data/soft/zookeeper-3.4.6. /usr/local/zookeeper
```

（3）设置环境变量

```
vim /etc/profile
export ZOOKEEPER_HOME=/usr/local/zookeeper
export PATH=$PATH: $JAVA_HOME/bin:$M2_HOM/bin:$SCALA_HOME/bin
                 : $ZOOKEEPER_HOME/bin
```

刷新环境变量使其生效：`Source/etc/profile`

（4）配置

进入到 Zookeeper 安装目录：`cd /usr/local/zookeeper`

拷贝一份 conf 目录下的配置文件，重命名为 zoo.cfg：`cp ./conf/zoo_sample.cfg ./conf/zoo.cfg`

编辑配置文件设置关键参数：

```
tickTime=2000
initLimit=5
syncLimit=3
dataDir=/data/zookeeper/data
dataLogDir=/usr/local/zookeeper/logs
clientPort=2181
server.1=192.168.1.1:2888:3888
server.2=192.168.1.2:2888:3888
server.3=192.168.1.3:2888:3888
```

关键参数说明：

- tickTime：Zookeeper 中的基础参考时间，所有与时间相关的设置都为 tickTime 时间的整数倍，单位是毫秒。
- initLimit：Zookeeper Leader 与 Follower 初始连接时，Follower 需要从 Leader 同步最新数据，该值表示 Follower 同步数据的最大超时时间，一般为整数，表示是 tickTime 的整数倍时间。
- syncLimit：Leader 和 Follower 之间心跳检测的最大超时时间，超过这个时间则认为 Follower 已经下线。该参数值为整数，表示是 tickTime 的整数倍时间。
- dataDir：Zookeeper 持久化数据目录，建议与安装路径不在同一个路径下。
- dataLogDir：日志文件目录。
- clientPort：监听客户端连接的端口号，默认值为 2181。
- server.X=A:B:C。其中 X 是一个数字，表示这是第几号 server；A 是该 server 所在

的 IP 地址；B 配置该 server 和集群中的 leader 交换消息所使用的端口；C 配置选举 leader 时所使用的端口。

（5）创建 myid 文件

在配置参数 dataDir 对应的路径下新建 myid 文件，写入单独的一个数字，表示集群中该实例的编号，该值在集群中是唯一值，不可以重复，数字必须和 zoo.cfg 配置文件中的 server.X 中的 X 一一对应。

（6）启动 Zookeeper

```
bin/zkServer.sh start
```

（7）验证安装是否成功

`bin/zkServer.sh status`（一个 leader，两个 follower）

或者在 Zookeeper 安装的任何一个节点执行客户端连接命令：

```
bin/zkCli.sh -server 192.168.1.1:2181
```

1.2.2 Hadoop 部署

1. Hadoop 简介

Apache Hadoop 是由著名的 Apache 基金会开源的分布式存储计算系统，能够在廉价的硬件上轻松实现高可靠、高扩展、高性能、高容错等特性。通过增加机器即可直线增加集群的存储和计算能力。Hadoop 在大规模分布式系统中起着重要的作用，目前已经形成一套完整的 Hadoop 生态系统，并且在不断发展扩大。随着 Hadoop 生态系统的不断发展，Hadoop 已应用到互联网、大数据交通、智能医疗、气象监测、金融服务、人工智能等众多领域。

HDFS（Hadoop Distributed File System，Hadoop 分布式文件系统）：通过对文件分块多备份分布式存储的方式保证数据具有高效的容错能力，并且有效提高数据的吞吐量。

MapReduce：应用于规模分布式计算的编程模型，该模型包含 Map 和 Reduce 两种编程原语。Map 阶段常用于接入数据源，数据划分、过滤、整理等操作。Reduce 阶段常用于接收 Map 阶段的数据，聚合计算，持久化结果数据。

YARN：作业调度和集群资源管理框架。目前已经有很多开源项目部署到 YARN 上运行，将 YARN 作为统一的作业调度和资源管理框架，如 Spark、HBase、Tez 等。

2. Hadoop 集群部署

本节主要介绍 Hadoop2.6.4 版本的 Hadoop 集群部署。

1. 集群规划

为保证集群的高可用能力，NameNode 和 ResourceManager 都采用 HA 部署方式，各组件详细分布情况如表 1-1 所示。

表 1-1　Hadoop 集群规划

主机名	IP	运行进程
hadoop01、nn1、rm1	192.168.1.1	NameNode、DataNode、JournalNode、DFSZKFailoverController ResourceManager、NodeManager JobHistory Server QuorumPeerMain
hadoop02、nn2、rm2	192.168.1.2	NameNode、DataNode、JournalNode、DFSZKFailoverController ResourceManager、NodeManager QuorumPeerMain
hadoop03	192.168.1.3	DataNode、JournalNode NodeManager QuorumPeerMain

2. 部署过程

（1）SSH 免密码登录

使用 root 用户登录进入到 .ssh 目录下

```
cd ~/.ssh
```

执行 `ssh-keygen -t rsa` 生成公钥和私钥。系统会一直提示信息，一直按回车就可以。生成私钥文件 id_rsa，公钥文件 id_rsa.pub，认证文件 authorized_keys。

将公钥文件内容追加到认证文件中

```
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

在免密码登录的机器之间互相拷贝公钥然后追加到认证文件中，即可完成 SSH 免密码登录配置。

（2）创建 hadoop 用户和组

```
groupadd hadoop
useradd -m -g hadoophadoop
```

（3）下载安装包并解压

先安装 hadoop01，然后将配置好的安装包拷贝到其他节点。

```
wget http://www.apache.org/dyn/closer.cgi/hadoop/common/hadoop-2.6.5/hadoop-2.6.5.tar.gz
```

解压到指定目录 /data/soft/ 下

```
tar -zxvf hadoop-2.6.5.tar.gz-C /data/soft/
```

（4）创建软连接并修改属主为 hadoop

创建软连接便于以后升级版本，方便统一管理。

```
ln -s /data/soft/ hadoop-2.6.5 /usr/local/hadoop
```

```
chown -R hadoop:hadoop /usr/local/hadoop
```

（5）设置环境变量

```
vim /etc/profile
export HADOOP_HOME=/usr/local/hadoop
export PATH=$PATH: $JAVA_HOME/bin:$M2_HOM/bin:$SCALA_HOME/bin
                 : $ZOOKEEPER_HOME/bin:$ HADOOP_HOME/bin
```

刷新环境变量使其生效

```
source /etc/profile
```

（6）设置配置文件

a）HDFS 相关的配置文件 core-site.xml 和 hdfs-site.xml。

core-site.xml 配置信息如下：

```
<configuration>
    <!-- 指定 hdfs 的 nameservice 为 ns1 -->
    <property>
        <name>fs.defaultFS</name>
        <value>hdfs://ns1</value>
    </property>
    <!-- 指定 hadoop 临时目录 -->
    <property>
        <name>hadoop.tmp.dir</name>
        <value>/usr/local/hadoop/tmp</value>
    </property>
    <!-- 指定 zookeeper 地址 -->
    <property>
        <name>ha.zookeeper.quorum</name>
        <value>hadoop01:2181,hadoop02:2181,hadoop03:2181</value>
    </property>
</configuration>
```

hdfs-site.xml 配置信息如下：

```
<configuration>
    <!-- 指定 hdfs 的 nameservice 为 ns1，需要和 core-site.xml 中的保持一致 -->
    <property>
        <name>dfs.nameservices</name>
        <value>ns1</value>
    </property>
    <!-- ns1 下面有两个 NameNode，分别是 nn1，nn2 -->
    <property>
        <name>dfs.ha.namenodes.ns1</name>
        <value>nn1,nn2</value>
    </property>
    <!-- nn1 的 RPC 通信地址 -->
```

```
<property>
    <name>dfs.namenode.rpc-address.ns1.nn1</name>
    <value>hadoop01:9000</value>
</property>
<!-- nn1 的 http 通信地址 -->
<property>
    <name>dfs.namenode.http-address.ns1.nn1</name>
    <value>hadoop01:50070</value>
</property>
<!-- nn2 的 RPC 通信地址 -->
<property>
    <name>dfs.namenode.rpc-address.ns1.nn2</name>
    <value>hadoop02:9000</value>
</property>
<!-- nn2 的 http 通信地址 -->
<property>
    <name>dfs.namenode.http-address.ns1.nn2</name>
    <value>hadoop02:50070</value>
</property>
<!-- 指定 NameNode 的元数据在 JournalNode 上的存放位置 -->
<property>
    <name>dfs.namenode.shared.edits.dir</name>
    <value>
        qjournal://hadoop01:8485;hadoop02:8485;hadoop03:8485/ns1
    </value>
</property>
<!-- 指定 JournalNode 在本地磁盘存放数据的位置 -->
<property>
    <name>dfs.journalnode.edits.dir</name>
    <value>/usr/local/hadoop/journal</value>
</property>
<!-- 开启 NameNode 失败自动切换 -->
<property>
    <name>dfs.ha.automatic-failover.enabled</name>
    <value>true</value>
</property>
<!-- 配置失败自动切换实现方式 -->
<property>
    <name>dfs.client.failover.proxy.provider.ns1</name>
    <value>org.apache.hadoop.hdfs.server.namenode.ha
            .ConfiguredFailoverProxyProvider
    </value>
</property>
<!-- 配置隔离机制 -->
<property>
    <name>dfs.ha.fencing.methods</name>
    <value>sshfence</value>
</property>
<!-- 使用隔离机制时需要 ssh 免登陆 -->
<property>
```

```
        <name>dfs.ha.fencing.ssh.private-key-files</name>
        <value>/root/.ssh/id_rsa</value>
    </property>
    <!-- 导致 DN 停止工作的坏硬盘最大数，默认 0 就是只要有 1 个硬盘坏了，DN 就会 shutdown -->
    <property>
        <name>dfs.datanode.failed.volumes.tolerated</name>
        <value>2</value>
    </property>
    <!--Block 块副本数为 3 -->
    <property>
        <name>dfs.replication</name>
        <value>3</value>
    </property>
    <!-- fsimage 和 edit 文件存储路径 -->
    <property>
        <name>dfs.namenode.name.dir</name>
        <value>/data/hadoop/data1/dfs/name</value>
    </property>
    <!-- 数据存储物理路径，可以配置多块盘 -->
    <property>
        <name>dfs.datanode.data.dir</name>
        <value>/data/hadoop/data1/dfs/data,/data/hadoop/data2/dfs/data </value>
    </property>
    <!-- block 块大小 512M-->
    <property>
        <name>dfs.block.size</name>
        <value>536870912</value>
    </property>
</configuration>
```

向 slaves 文件添加 datanode/nodemanager 节点的 hostname：

```
hadoop01
hadoop02
hadoop03
```

b）YARN 相关配置文件。

yarn-site.xml 配置信息如下：

```
<configuration>
    <!-- 开启 ResourceManager HA -->
    <property>
        <name>yarn.resourcemanager.ha.enabled</name>
        <value>true</value>
    </property>
    <!-- 开启 ResourceManager 失败自动切换 -->
    <property>
        <name>yarn.resourcemanager.ha.automatic-failover.enabled</name>
        <value>true</value>
    </property>
```

```
<!--RM 失败后正在运行的任务在 RM 恢复之后重新启动 -->
<property>
    <name>yarn.resourcemanager.recovery.enabled</name>
    <value>true</value>
</property>
<!-- 运行 ResourceManager 的两个节点 -->
<property>
    <name>yarn.resourcemanager.ha.rm-ids</name>
    <value>rm1,rm2</value>
</property>
<!-- 当应用程序未指定队列名时，指定用户名作为应用程序所在的队列名 -->
<property>
    <name>yarn.scheduler.fair.user-as-default-queue</name>
    <value>true</value>
</property>
<!-- RM 状态信息存储方式 -->
<property>
    <name>yarn.resourcemanager.store.class</name>
    <value>org.apache.hadoop.yarn.server.resourcemanager.recovery
        .ZKRMStateStore
    </value>
</property>
<property>
    <name>yarn.resourcemanager.cluster-id</name>
    <value>yarn-ha</value>
</property>
<property>
    <name>yarn.resourcemanager.hostname.rm1</name>
    <value>rm1</value>
</property>
<!-- RM1 HTTP Web 访问地址 -->
<property>
    <name>yarn.resourcemanager.webapp.address.rm1</name>
    <value>${yarn.resourcemanager.hostname.rm1}:8088</value>
</property>
<property>
    <name>yarn.resourcemanager.hostname.rm2</name>
    <value>rm2</value>
</property>
<!-- RM2 HTTP Web 访问地址 -->
<property>
    <name>yarn.resourcemanager.webapp.address.rm2</name>
    <value>${yarn.resourcemanager.hostname.rm2}:8088</value>
</property>
<!-- NodeManger 节点可使用的总内存大小 -->
<property>
    <name>yarn.nodemanager.resource.memory-mb</name>
    <value>81920</value>
</property>
<!-- NodeManger 节点可使用的总 vcore 数量 -->
```

```
<property>
    <name>yarn.nodemanager.resource.cpu-vcores</name>
    <value>10</value>
</property>
<!-- Zookeeper 链接地址，为了容错应配置多个 -->
<property>
    <name>yarn.resourcemanager.zk-address</name>
    <value>hadoop01:2181,hadoop02:2181,hadoop03:2181</value>
</property>
<!-- NM 本地任务运行日志存储路径 -->
<property>
    <name>yarn.nodemanager.log-dirs</name>
    <value>file:///data/hadoop/data1/yarn/log
           ,file:///data/hadoop/data2/yarn/log
    </value>
</property>
<!-- ApplicationMaster 占用的内存大小 -->
<property>
    <name>yarn.app.mapreduce.am.resource.mb</name>
    <value>2048</value>
</property>
<!-- 单个任务可申请的最少物理内存量 -->
<property>
    <name>yarn.scheduler.minimum-allocation-mb</name>
    <value>1024</value>
</property>
<!-- 单个任务可申请的最多物理内存量 -->
<property>
    <name>yarn.scheduler.maximum-allocation-mb</name>
    <value>8192</value>
</property>
<!-- 单个任务可申请的最少 vcore 量 -->
<property>
    <name>yarn.scheduler.minimum-allocation-vcores</name>
    <value>1</value>
</property>
<!-- 单个任务可申请的最多 vcore 量 -->
<property>
    <name>yarn.scheduler.maximum-allocation-vcores</name>
    <value>10</value>
</property>
<!-- 开启日志聚合到 HDFS-->
<property>
    <name>yarn.log-aggregation-enable</name>
    <value>true</value>
</property>
<!-- 聚合日志保存时长，单位秒 -->
<property>
    <name>yarn.log-aggregation.retain-seconds</name>
    <value>259200</value>
</property>
```

```
    <!-- 聚合日志 HDFS 存储路径 -->
    <property>
        <name>yarn.nodemanager.remote-app-log-dir</name>
        <value>/data/hadoop/yarn-logs</value>
    </property>
    <!-- 使用公平调度器 -->
    <property>
        <name>yarn.resourcemanager.scheduler.class</name>
        <value>org.apache.hadoop.yarn.server.resourcemanager.scheduler.fair
            .FairSchedule
        </value>
    </property>
    <!-- 使用公平调度器配置文件路径 -->
    <property>
        <name>yarn.scheduler.fair.allocation.file</name>
        <value>/usr/local/hadoop/etc/hadoop/fair-scheduler.xml</value>
    </property>
</configuration>
```

mapred-site.xml 配置信息如下：

```
<configuration>
    <!--History Server 配置 -->
    <property>
        <name>mapreduce.jobhistory.address</name>
        <value>hadoop01:10020</value>
    </property>
    <property>
        <name>mapreduce.jobhistory.webapp.address</name>
        <value>hadoop01:19888</value>
    </property>
    <property>
        <name>mapreduce.jobhistory.joblist.cache.size</name>
        <value>200000</value>
    </property>
    <!--MapReduce 作业运行在 Yarn 上 -->
    <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
    </property>
    <property>
        <name>mapreduce.map.memory.mb</name>
        <value>1024</value>
    </property>
    <property>
        <name>mapreduce.reduce.memory.mb</name>
        <value>8192</value>
    </property>
    <property>
        <name>mapreduce.map.java.opts</name>
        <value>-Xmx1700m -Xms900m</value>
```

```
    </property>
    <property>
        <name>mapreduce.reduce.java.opts</name>
        <value>-Xmx7168m -Xms3000m</value>
    </property>
    <property>
        <name>mapreduce.client.submit.file.replication</name>
        <value>20</value>
    </property>
    <!--
        默认情况下是 user.name，即每个用户独自一个 pool；group.name，即一个 linux group
        一个 pool，mapred.job.queue.name，即一个 queue 一个 pool。
    -->
    <property>
        <name>mapred.fairscheduler.poolnameproperty</name>
        <value>group.name</value>
    </property>
</configuration>
```

fair-scheduler.xml 公平调度策略配置信息如下（按组分配不同的内存和 vcore 资源）：

```
<allocations>
    <pool name="group1">
    <maxResources>50000 mb,10 vcores</maxResources>
    <maxRunningApps>10</maxRunningApps>
    <weight>1.0</weight>
    <schedulingPolicy>fair</schedulingPolicy>
    </pool>
    <pool name="group1">
    <maxResources>80000 mb, 20 vcores</maxResources>
    <maxRunningApps>20</maxRunningApps>
    <weight>1.0</weight>
    <schedulingPolicy>fair</schedulingPolicy>
    </pool>
    <userMaxAppsDefault>99</userMaxAppsDefault>
    <queuePlacementPolicy>
    <rule name="primaryGroup" create="false" />
    <rule name="secondaryGroupExistingQueue" create="false" />
    <rule name="reject"/>
    </queuePlacementPolicy>
</allocations>
```

将配置好的 hadoop 拷贝到其他节点：

```
scp -r /data/soft/hadoop-2.6.5 hadoo02: /data/soft/
scp -r/data/soft/hadoop-2.6.5 hadoo03: /data/soft/
```

（7）集群启动

从 root 用户切换到 hadoop 用户：

```
su - hadoop
```

启动 journalnode（在 hadoop01 上启动所有 journalnode）：

```
cd /usr/local/hadoop
sbin/hadoop-daemons.sh start journalnode
```

jps 验证，后台进程增加 JournalNode 进程。

格式化 HDFS：

在 hadoop01 上执行命令：`hadoop namenode -format`

格式化后会根据在 core-site.xml 中的 hadoop.tmp.dir 配置生成一个文件，拷贝该文件到另外一个 NameNode 节点 hadoop02 的 /usr/local/hadoop/tmp 目录下：

```
scp -r /usr/local/hadoop/tmp/ hadoop02:/usr/local/hadoop/
```

格式化 ZK（在 hadoop01 上执行即可）：

```
hdfs zkfc -formatZK
```

启动 HDFS（在 hadoop01 上执行）：

```
sbin/start-dfs.sh
```

启动 YARN（在 hadoop01 上执行）：

```
sbin/start-yarn.sh
```

Hadoop 部署完成后在各个节点中使用 jps 命令查看各组件进程是否运行正常。如果发现有问题则查看日志进行排查。

（8）可以通过浏览器访问查看

http://192.168.1.1:50070

页面显示：NameNode 'hadoop01:9000' (active)

http://192.168.1.2:50070

页面显示：NameNode 'hadoop02:9000' (standby)

http://192.168.1.1:8088

1.3　小结

本章主要介绍了构建基础环境所需要软件的安装方法，目前使用的大部分开源软件依赖于 JVM，使用较为广泛的开发语言为 Java.Scala，而源码编译普遍使用 Maven 工具。Hadoop 目前已经可以作为大数据应用系统的基础系统，提供分布式数据存储、集中式资源调度、大规模分布式计算等功能。通过本章的学习初步构建了一套大数据应用系统的基础环境。