Hadoop2.x 系列学习笔记

前言

学习Hadoop2.x主要分为四部分，大纲如下：

第一部分：Hadoop 2.x入门

1. 初步认识Hadoop2
2. 编译源代码 导入到Eclipse
3. Hadoop2模块架构原理
4. Hadoop2伪分布式安装部署
5. 启动hadoop2（HDFS YARN）
6. 深入理解hadoop2的相关配置
7. 阅读官方的Reference
8. Hadoop2.x分布式安装部署

第二部分：分布式文件系统HDFS

1. HDFS系统架构
2. NameNode HA
3. NameNode Federation
4. 分布式服务框架Zookeeper
5. 企业级HDFS集群：大数据集群的整体设计、集群机器角色的规划、集群配置、升级等
6. HDFS客户端调用：JavaAPI方式
7. 新特性：HDFS Snapshots HFTP Short-Circuit Local Reads NFSv3

第三部分：分布式资源管理系统YARN

1. YARN产生的背景
2. YARN系统架构
3. YARN运维管理
4. Spark & Storm：流式实时计算背景、内存计算框架Spark、实时计算框架Storm

第四部分：分布式计算框架MapReduce

1. 架构组成
2. MapReduce基础
3. MapReduce高级编程
4. 项目实战

01课时 Hadoop2介绍和生态系统

1. Hadoop——轻松应对海量数据存储和分析带来的挑战

海量数据存储方案：HDFS分布式文件系统

海量数据分析方案：MapReduce分布式计算框架

1. Apache Hadoop介绍hadoop

**What Is Apache Hadoop?**

The Apache™ Hadoop® project develops open-source software for reliable, scalable, distributed computing.

The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.

**The project includes these modules:**

**Hadoop Common**: The common utilities that support the other Hadoop modules.

**Hadoop Distributed File System (HDFS™):** A distributed file system that provides high-throughput access to application data.

**Hadoop YARN**: A framework for job scheduling and cluster resource management.

**Hadoop MapReduce**: A YARN-based system for **parallel** processing of large data sets.

**与Hadoop2相关的生态系统**

Other Hadoop-related projects at Apache include:

[**Ambari™**](http://incubator.apache.org/ambari/)**:** A web-based tool for provisioning, managing, and monitoring Apache Hadoop clusters which includes support for Hadoop HDFS, Hadoop MapReduce, Hive, HCatalog, HBase, ZooKeeper, Oozie, Pig and Sqoop. Ambari also provides a dashboard for viewing cluster health such as heatmaps and ability to view MapReduce, Pig and Hive applications visually alongwith features to diagnose their performance characteristics in a user-friendly manner.

[**Avro™**](http://avro.apache.org/)**:** A data serialization system.

[**Cassandra™**](http://cassandra.apache.org/): A scalable multi-master database with no single points of failure.

[**Chukwa™**](http://incubator.apache.org/chukwa/)**:** A data collection system for managing large distributed systems.

[**HBase™**](http://hbase.apache.org/): A scalable, distributed database that supports structured data storage for large tables.

[**Hive™**](http://hive.apache.org/): A data warehouse infrastructure that provides data summarization and ad hoc querying.

[**Mahout™**](http://mahout.apache.org/): A Scalable machine learning and data mining library.

[**Pig™**](http://pig.apache.org/): A high-level data-flow language and execution framework for parallel computation.

[**Spark™**](http://spark.incubator.apache.org/)**:** A fast and general compute engine for Hadoop data. Spark provides a simple and expressive programming model that supports a wide range of applications, including ETL, machine learning, stream processing, and graph computation.

[**Tez™**](http://tez.incubator.apache.org/)**:** A generalized data-flow programming framework, built on Hadoop YARN, which provides a powerful and flexible engine to execute an arbitrary DAG of tasks to process data for both batch and interactive use-cases. Tez is being adopted by Hive™, Pig™ and other frameworks in the Hadoop ecosystem, and also by other commercial software (e.g. ETL tools), to replace Hadoop™ MapReduce as the underlying execution engine.

[**ZooKeeper™**](http://zookeeper.apache.org/): A high-performance coordination service for distributed applications.

YARN：被称为“云操作系统”（ From Hadoop商业版本Hortonworks）

对于部署在YARN上面的应用，分配资源（CPU、内存、网络、IO），

管理资源

JOB/Application 对于任务的调度

Spark计算框架：（Kettle工具，对于数据的清理）流式计算、图形计算。

1. Java做开发相关的领域

服务总线：SOA/OSB，Dubble

全文检索：Solr Cloud，Netch， Lucene，将Solr与hadoop框架集成

云计算领域：Hadoop2.x Spark等

02课时 Hadoop起源和版本

1. Hadoop起源

Lucene Nutch:全文检索系统

Google: MapReduce GFS BigTable

Hadoop

1. Hadoop版本介绍

1）0.20.x 和1.x.x版本

2）0.21.x系列版本

3）2.x版本：2.0~2.1测试版本，2.2.x之后为实用版本

03课时 Hadoop源代码编译

Hadoop2.x系列在二进制包中只有32位的运行库，没有64位系统下的运行库，因此对于64位系统，需要我们自行编译源代码，生成对应的库。只是针对2.2版本，但是在2.6.x之后，又是默认64位的本地库。

[hadoop@hadoop-master native]$ file libhadoop.so.1.0.0

libhadoop.so.1.0.0: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, BuildID[sha1]=b3b5e812c2a91fa4b28aa33eb76dc6889d3b91e9, not stripped

解压官方的源代码，查看Build文件

Build instructions for Hadoop

----------------------------------------------------------------------------------

Requirements:

\* Unix System

\* JDK 1.6+

\* Maven 3.0 or later

\* Findbugs 1.3.9 (if running findbugs)

\* ProtocolBuffer 2.5.0

\* CMake 2.6 or newer (if compiling native code), must be 3.0 or newer on Mac

\* Zlib devel (if compiling native code)

\* openssl devel ( if compiling native hadoop-pipes )

\* Internet connection for first build (to fetch all Maven and Hadoop dependencies)

1. 安装linux系统包：

yum install autoconf automake libtool cmake

yum install ncurses-devel

yum install openssl-devel

yum install lzo-devel zlib-devel gcc gcc-c++

1. 安装MAVEN 3.0.x版本

tar -zxvf apache-maven-\*.tar.gz -C /usr/cloud

设置环境变量：

sudo ln -s apache-maven-\*/ maven

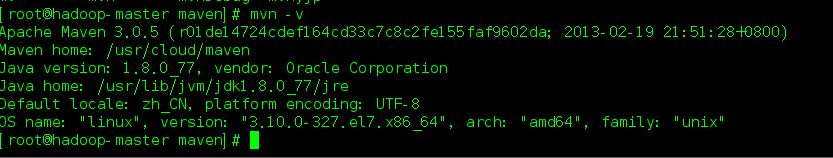
vim /etc/profile

*export MAVEN\_HOME=/opt/cloud/maven*

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

生效命令$ source /etc/profile

验证命令：mvn –v



配置Maven在国内的镜像（略过）

1. 安装protobuffer

tar -zxvf protobuf\*.tar.gz -C /usr/cloud

ln -s protobuf\* protobuf

cd protobuf

./configure

make

make install

1. 安装findbugs

tar -zxvf findbugs-3.0.1.tar.gz -C /usr/cloud/

ln -s

设置环境变量/etc/profile 并且生效

#findbugs environment

export FINDBUGS\_HOME=/usr/cloud/findbugs

export PATH=$PATH:$FINDBUGS\_HOME/bin

findbugs -version

下面就可以编译执行，在hadoop2.2.0源码中出现bugs，缺少一个pom依赖包，因此会出现编译错误，之后的版本不会没有此bug。

编译hadoop源代码，查看BUILD文件中的编译命令：

|  |
| --- |
| Building distributions:  Create binary distribution without native code and without documentation:  $ mvn package -Pdist -DskipTests -Dtar  Create binary distribution with native code and with documentation:  $ mvn package -Pdist,native,docs -DskipTests -Dtar  Create source distribution:  $ mvn package -Psrc -DskipTests  Create source and binary distributions with native code and documentation:  $ mvn package -Pdist,native,docs,src -DskipTests -Dtar  Create a local staging version of the website (in /tmp/hadoop-site)  $ mvn clean site; mvn site:stage -DstagingDirectory=/tmp/hadoop-site |

我们使用命令

mvn package –Pdist,native,docs –DskipTests -Dtar

此时会maven联网下载依赖的JAR包，然后在执行编译，下载的maven对应的JAR包在当前用户目录下面.m2文件夹中，有一个respository。

编译过程中出错误，可能出现Java内存溢出的错误，

export MAVEN\_OPTS="-Xms256m -Xmx512m"，设置Java的内存

使用JDK1.8编译Hadoop可能出现错误Failed to execute goal org.apache.maven.plugins:maven-javadoc-plugin:2.8.1:jar (module-javadocs) on。

我们需要更换JDK1.7

04课时Hadoop2目录结构

bin:hadoop最近本的管理脚本和使用脚本，这些脚本是sbin目录下管理脚本的基础实现，用户可以直接使用这些脚本管理和使用hadoop

etc：配置文件 core-site.xml hdfs-site.xml mapred-site.xml yarn-site.zml等等配置文件

include: 对外提供的编程库头文件，具体的动态库静态库在lib目录下面（C++格式）

lib:C++的库目录

libexec:各个五福的shell配置文件所在的目录，可用于配置日志输出目录、启动参数等基本信息

sbin:Hadoop管理脚本，HDFS 和YARN中各种服务的启动关闭脚本

share:Hadoop各个模块编译后的JAR包所在目录

05课时 将Hadoop源码到入Eclipse

BUILD.text文档中介绍的方法

----------------------------------------------------------------------------------

Importing projects to eclipse

When you import the project to eclipse, install hadoop-maven-plugins at first.

$ **cd hadoop-maven-plugins**

$ **mvn install**

Then, generate eclipse project files.[回到src目录]

$ **mvn eclipse:eclipse -DskipTests**

At last, import to eclipse by specifying the root directory of the project via

[File] > [Import] > [Existing Projects into Workspace].

06课时 Hadoop2.x架构介绍

NameNode：存储文件的元数据（文件名 文件属性 文件与DataNode的对应关系）

DataNode：存储文件块数据以及块校验和

SecondaryNameNode：监控HDFS状态的辅助后台程序，每隔一段时间获取NameNode的元数据快照。

07课时 YARN架构

ResourceManager:

处理客户端的请求

启动和监控ApplicationMaster

监控NodeManager

资源分配和调度

NodeManager：

单个节点上面的资源管理

处理来自ResourceManager的命令

处理来自ApplicationMaster的命令

ApplicationMaster:

数据的切分

为应用程序申请资源，并且分配给内部任务

任务的监控和容错

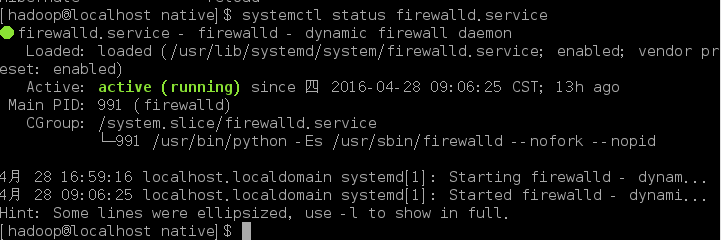
Container：

对于任务运行环境的抽象，封装CPU、内存等多维资源以及环境变量、启动命令等任务运行相关的信息

08 09课时 Hadoop2.x 伪分布式安装

之前学习了Hadoop1.x的伪分布式安装，这里学习Hadoop2.x的伪分布式安装，内容在之前的基础之上，添加了关于YARN的知识。

1. 系统介绍
2. CentOS7
3. 关闭iptables （firewall.service）防火墙和SELinux

在Centos7中使用的是systemctl start|status|stop| **disable** firewall.service

但是在之前的Centos或者是其他的linux中，使用的是

service iptables status |start| stop#暂时关闭服务，重新启动之后会再次启动

永久性关闭使用的是chkconfig iptables off

关闭SELinux(在文件/etc/sysconfig/selinux):设置SELINUX=disabled

1. 设置静态IP地址

/etc/sysconfig/network-scripts/ifcfg-eno\*

|  |
| --- |
| TYPE=Ethernet  BOOTPROTO=none  DEFROUTE=yes  IPV4\_FAILURE\_FATAL=no  IPV6INIT=yes  IPV6\_AUTOCONF=yes  IPV6\_DEFROUTE=yes  IPV6\_FAILURE\_FATAL=no  NAME=eno16777736  UUID=7d44b9cc-d955-455e-a55a-ede9eb47f4e4  DEVICE=eno16777736  ONBOOT=yes  IPADDR=192.168.1.201  PREFIX=24  GATEWAY=192.168.1.1  DNS1=192.168.1.1  DNS2=114.114.114.114  IPV6\_PEERDNS=yes  IPV6\_PEERROUTES=yes |

1. 修改主机名称：hostname 在文件中/etc/hostname
2. 修改hosts 文件： IP hadoop-yarn
3. 解压hadoop

创建/usr/cloud,修改cloud所有者和用户组，解压hadoop到/usr/cloud，创建软链接

tar -zxvf hadoop-2.6.4.tar.gz -C /usr/cloud/

1. 配置文件的修改
2. hadoop-env.sh

配置JAVA\_HOME : export JAVA\_HOME=/usr/lib/jvm/java8

1. yarn-env.sh

export JAVA\_HOME=/usr/lib/jvm/java8

1. mapred-env.sh

export JAVA\_HOME=/usr/lib/jvm/java8

1. core-site.xml

<property>

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

<value>hdfs://hadoop-yarn:8020</value>

</property>

<property>

<name>hadoop.tmp.dir</name>

<value>/opt/hadoop/tmp</value>

</property>

1. hdfs-site.xml

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

<property>

<name>dfs.permissions</name>

<value>false</value>

</property>

1. mapred-site.xml

<property>

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

<value>yarn</value>

</property>

1. yarn-site.xml

<property>

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

<value>mapreduce\_shuffle</value>

</property>

1. 启动Hadoop
   1. HDFS文件系统：NameNode DataNode SecondaryNameNode

bin/hdfs namenode –format [–clusterid yarn-cluster]指定clusterID

Cluster ID

Block Pool ID 数据块池ID

For NameNode Fedaration

sbin/hadoop-daemon.sh start namenode

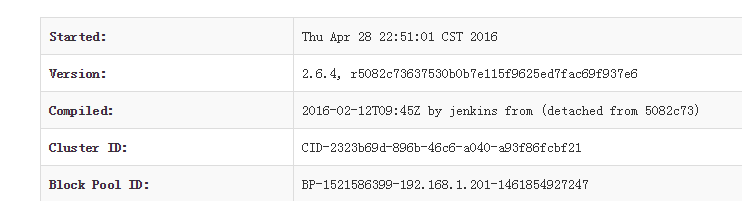
sbin/hadoop-daemon.sh start datanode

sbin/hadoop-daemon.sh start secondarynamenode

HDFS: 50070

SecondaryNameNoe: 50090

* 1. <http://hadoop-yarn:50070/dfshealth.html#tab-overview>



* 1. 日志名字格式

FramworkName-UserName-ServiceName-HostName.LogType

* 1. ulimit -a



3.5 Journal Status是客户端对于文件的操作，也就是NameNode内存中存放的fsimage文件系统镜像文件，然后客户端对于文件的操作记录放在edits编辑日志文件中，而SecondaryNameNode就会定期的合并两个文件，然后复制到NameNode。合并的过程被称为checkpoint。

10课时 HDFS Shell命令和文件系统权限设置

1. SecondaryNameNode（<http://hadoop-yarn:50090>）

主要是CheckPoint的参数

1. HDFS文件系统的命令

hdfs dfs -ls /

hdfs dfs –put <localsrc> <dest>

在hadoop文件系统中，可能会出现权限的问题，我们可以在hdfs配置文件中设置权限检查为false：dfs.permissions false

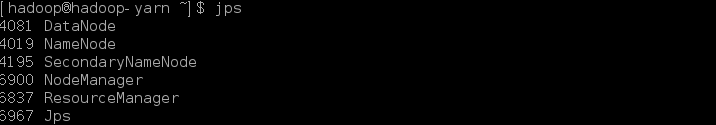
11课时 启动YARN服务 Web UI界面讲解 运行PI程序 MR历史服务器启动

之前在hadoop中的配置文件中有一个yarn-site.xml配置文件，启动YARN服务。

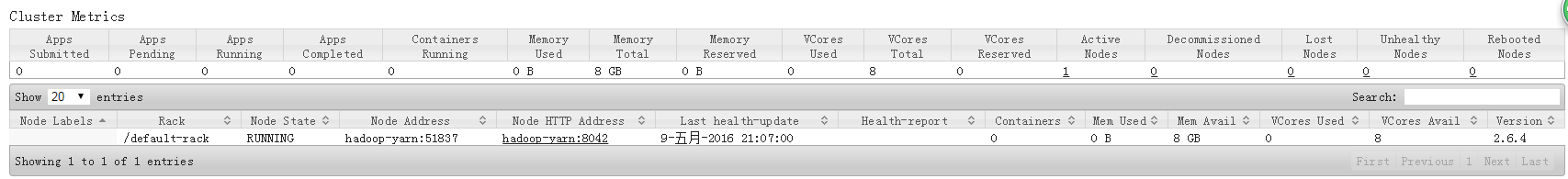
启动YARN服务：ResourceManager NodeManager

sbin/yarn-daemon.sh start resourcemanager

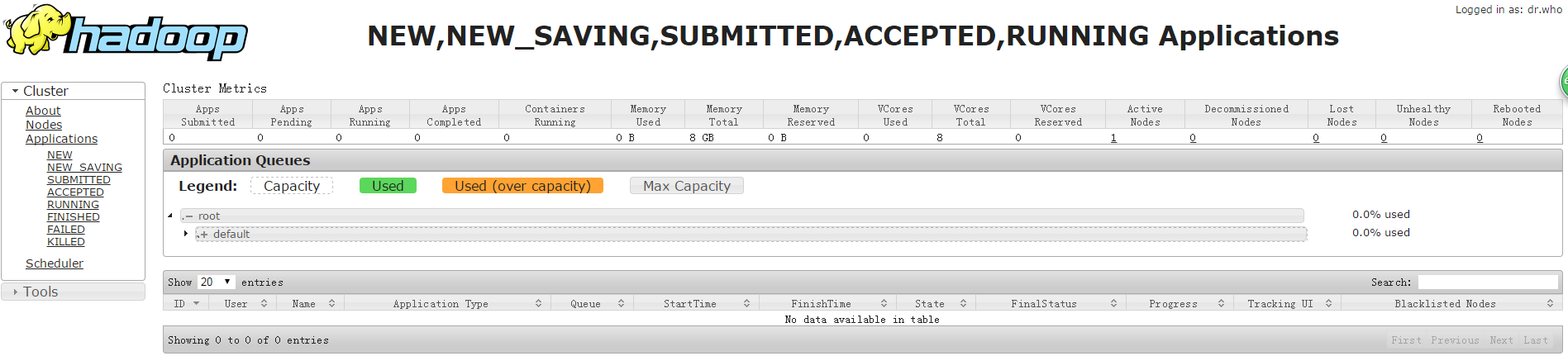
sbin/yarn-daemon.sh start nodemanager



Web UI 界面 ：8088



Application



运行MapReduce程序：

bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-\*.jar pi 10 20

bin/yarn jar share/hadoop/mapreduce/hadoop-mapreduce-examples-\*.jar pi 10 20

Logs: uber mode:false

Uber模式是hadoop2.x中实现的一种针对MR小作业的优化机制中，即作业比较小的话，所有的任务都运行在同一个JVM中（mr app master）中完成要比为每一个task启动一个container更划算。设置属性是 mapreduce.job.usertask.enable = true

计数器种类：

HDFS FileSystem

Job Counters

Map-Reduce Framework

Shuffle Error

FileInputFormat Counters

FileOutputFormat Counters

Job Id： job\_年月日时分\_job序列号，从0开始，上限为1000

Task Id： job\_年月日时分\_job序列号\_task序列号\_m [map task slot]

job\_年月日时分\_job序列号\_task序列号\_r [reduce task slot]

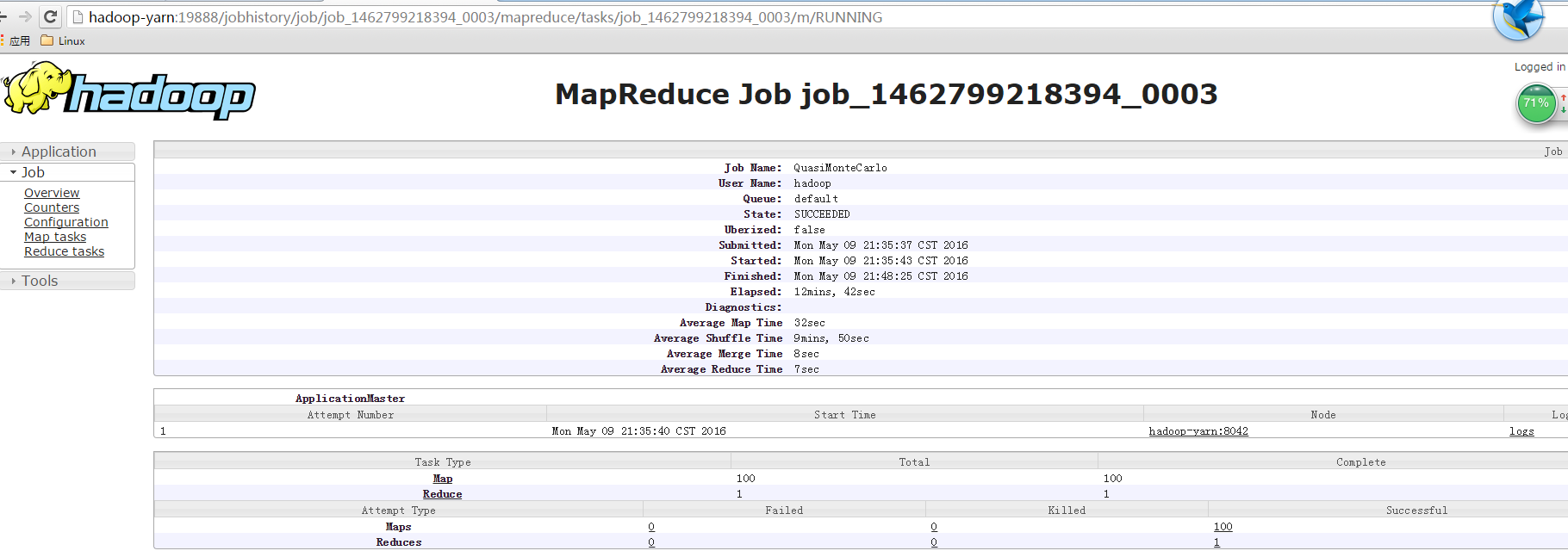
hadoop2 历史服务器

查看已经运行完成的MapReduce作业记录，比如Map Reduce数目，提交时间、完成时间等信息，默认情况下是不启动的。

sbin/mr-jobhistory-daemon.sh start historyserver

sbin/mr-jobhistory-daemon.sh stop historyserver

Web UI：<http://hadoop-yarn:19888>



我们这里没有配置SSH，并且启动方式不是使用的 sbin/start-dfs.sh start-yarn.sh start-all.sh，而是使用的sbin/hadoop-daemon.sh yarn-daemon.sh start <servicename>

12课时 复习伪分布式安装、启动HDFS YARN服务 WordCount程序

配置文件(四个)：

core-site.xml

hdfs-site.xml

mapred-site.xml

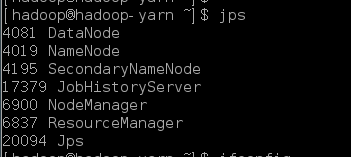
yarn-site.xml

启动各种服务：

sbin/hadoop-daemon.sh start|stop namenode|datanode|secondarynamenode

sbin/yarn-daemon.sh start|stop resourcemanager|nodemanager

sbin/mr-jobhistory-daemon.sh start|stop historyserver



13课时 hadoop的启动方式讲解

在没有配置ssh的情况下，采用我们的启动方式，依旧可以启动各种服务，不用输入密码。

启动方式一：逐一启动（我们采用的）

启动方式二：分开启动 start-dfs.sh start-yarn.sh

启动方式三：一起启动 start-all.sh（不推荐，已经过时）

分析启动脚本使用Notepad sftp方式查看文件

1. hadoop-daemon.sh

usage="Usage: hadoop-daemon.sh [--config <conf-dir>] [--hosts hostlistfile] [--script script] (start|stop) <hadoop-command> <args...>"

# Determine if we're starting a secure datanode, and if so, redefine appropriate variables

if [ "$command" == "datanode" ] && [ "$EUID" -eq 0 ] && [ -n "$HADOOP\_SECURE\_DN\_USER" ]; then

export HADOOP\_PID\_DIR=$HADOOP\_SECURE\_DN\_PID\_DIR

export HADOOP\_LOG\_DIR=$HADOOP\_SECURE\_DN\_LOG\_DIR

export HADOOP\_IDENT\_STRING=$HADOOP\_SECURE\_DN\_USER

starting\_secure\_dn="true"

fi

if [ "$HADOOP\_IDENT\_STRING" = "" ]; then

export HADOOP\_IDENT\_STRING="$USER"

fi

# some variables

export HADOOP\_LOGFILE=hadoop-$HADOOP\_IDENT\_STRING-$command-$HOSTNAME.log

export HADOOP\_ROOT\_LOGGER=${HADOOP\_ROOT\_LOGGER:-"INFO,RFA"}

export HADOOP\_SECURITY\_LOGGER=${HADOOP\_SECURITY\_LOGGER:-"INFO,RFAS"}

export HDFS\_AUDIT\_LOGGER=${HDFS\_AUDIT\_LOGGER:-"INFO,NullAppender"}

log=$HADOOP\_LOG\_DIR/hadoop-$HADOOP\_IDENT\_STRING-$command-$HOSTNAME.out

pid=$HADOOP\_PID\_DIR/hadoop-$HADOOP\_IDENT\_STRING-$command.pid

HADOOP\_STOP\_TIMEOUT=${HADOOP\_STOP\_TIMEOUT:-5}

case $startStop in

(start) #启动命令

case $command in

namenode|secondarynamenode|datanode|journalnode|dfs|dfsadmin|fsck|balancer|zkfc)

if [ -z "$HADOOP\_HDFS\_HOME" ]; then

hdfsScript="$HADOOP\_PREFIX"/bin/hdfs

else

hdfsScript="$HADOOP\_HDFS\_HOME"/bin/hdfs

fi

nohup nice -n $HADOOP\_NICENESS $hdfsScript --config $HADOOP\_CONF\_DIR $command "$@" > "$log" 2>&1 < /dev/null &

;;

(\*)

nohup nice -n $HADOOP\_NICENESS $hadoopScript --config $HADOOP\_CONF\_DIR $command "$@" > "$log" 2>&1 < /dev/null &

;;

1. start-dfs.sh

# Start hadoop dfs daemons.

# Optinally upgrade or rollback dfs state.

# **Run this on master node.**

usage="Usage: start-dfs.sh [-upgrade|-rollback] [other options such as -clusterId]"

#---------------------------------------------------------

# namenodes

**NAMENODES=$($HADOOP\_PREFIX/bin/hdfs getconf -namenodes)**

echo "Starting namenodes on [$NAMENODES]"

"$HADOOP\_PREFIX/sbin/**hadoop-daemons.sh**" \

--config "$HADOOP\_CONF\_DIR" \

--hostnames "$NAMENODES" \

--script "$bin/hdfs" start namenode $nameStartOpt

#---------------------------------------------------------

# datanodes (using default slaves file)

if [ -n "$HADOOP\_SECURE\_DN\_USER" ]; then

echo \

"Attempting to start secure cluster, skipping datanodes. " \

"Run start-secure-dns.sh as root to complete startup."

else

"$HADOOP\_PREFIX/sbin/hadoop-daemons.sh" \

--config "$HADOOP\_CONF\_DIR" \

--script "$bin/hdfs" start datanode $dataStartOpt

fi

#---------------------------------------------------------

# secondary namenodes (if any)

SECONDARY\_NAMENODES=$($HADOOP\_PREFIX/bin/hdfs getconf -secondarynamenodes 2>/dev/null)

if [ -n "$SECONDARY\_NAMENODES" ]; then

echo "Starting secondary namenodes [$SECONDARY\_NAMENODES]"

"$HADOOP\_PREFIX/sbin/hadoop-daemons.sh" \

--config "$HADOOP\_CONF\_DIR" \

--hostnames "$SECONDARY\_NAMENODES" \

--script "$bin/hdfs" start secondarynamenode

fi

#---------------------------------------------------------

# quorumjournal nodes (if any)

SHARED\_EDITS\_DIR=$($HADOOP\_PREFIX/bin/hdfs getconf -confKey dfs.namenode.shared.edits.dir 2>&-)

case "$SHARED\_EDITS\_DIR" in

qjournal://\*)

JOURNAL\_NODES=$(echo "$SHARED\_EDITS\_DIR" | sed 's,qjournal://\([^/]\*\)/.\*,\1,g; s/;/ /g; s/:[0-9]\*//g')

echo "Starting journal nodes [$JOURNAL\_NODES]"

"$HADOOP\_PREFIX/sbin/hadoop-daemons.sh" \

--config "$HADOOP\_CONF\_DIR" \

--hostnames "$JOURNAL\_NODES" \

--script "$bin/hdfs" start journalnode ;;

esac

#---------------------------------------------------------

# ZK Failover controllers, if auto-HA is enabled

AUTOHA\_ENABLED=$($HADOOP\_PREFIX/bin/hdfs getconf -confKey dfs.ha.automatic-failover.enabled)

if [ "$(echo "$AUTOHA\_ENABLED" | tr A-Z a-z)" = "true" ]; then

echo "Starting ZK Failover Controllers on NN hosts [$NAMENODES]"

"$HADOOP\_PREFIX/sbin/hadoop-daemons.sh" \

--config "$HADOOP\_CONF\_DIR" \

--hostnames "$NAMENODES" \

--script "$bin/hdfs" start zkfc

fi

# eof

1. hadoop-daemons.sh

# Run a Hadoop command on all slave hosts.

usage="Usage: hadoop-daemons.sh [--config confdir] [--hosts hostlistfile] [start|stop] command args..."

# if no args specified, show usage

if [ $# -le 1 ]; then

echo $usage

exit 1

fi

bin=`dirname "${BASH\_SOURCE-$0}"`

bin=`cd "$bin"; pwd`

DEFAULT\_LIBEXEC\_DIR="$bin"/../libexec

HADOOP\_LIBEXEC\_DIR=${HADOOP\_LIBEXEC\_DIR:-$DEFAULT\_LIBEXEC\_DIR}

. $HADOOP\_LIBEXEC\_DIR/hadoop-config.sh

exec "$bin/slaves.sh" --config $HADOOP\_CONF\_DIR cd "$HADOOP\_PREFIX" \; "$bin/hadoop-daemon.sh" --config $HADOOP\_CONF\_DIR "$@"

1. slaves.sh

在这个文件中使用了ssh命令远程主机列表，分别执行后面的命令。

在使用start-dfs.sh启动HDFS文件系统中为什么需要ssh协议，就是因为在slaves中使用了ssh

1. start-all.sh

调用的就是start-dfs.sh start-yarn.sh

1. start-yarn.sh

# Start all yarn daemons. Run this on master node.

echo "starting yarn daemons"

DEFAULT\_LIBEXEC\_DIR="$bin"/../libexec

HADOOP\_LIBEXEC\_DIR=${HADOOP\_LIBEXEC\_DIR:-$DEFAULT\_LIBEXEC\_DIR}

. $HADOOP\_LIBEXEC\_DIR/yarn-config.sh

# start resourceManager

"$bin"/yarn-daemon.sh --config $YARN\_CONF\_DIR start resourcemanager

# start nodeManager

"$bin"/yarn-daemons.sh --config $YARN\_CONF\_DIR start nodemanager

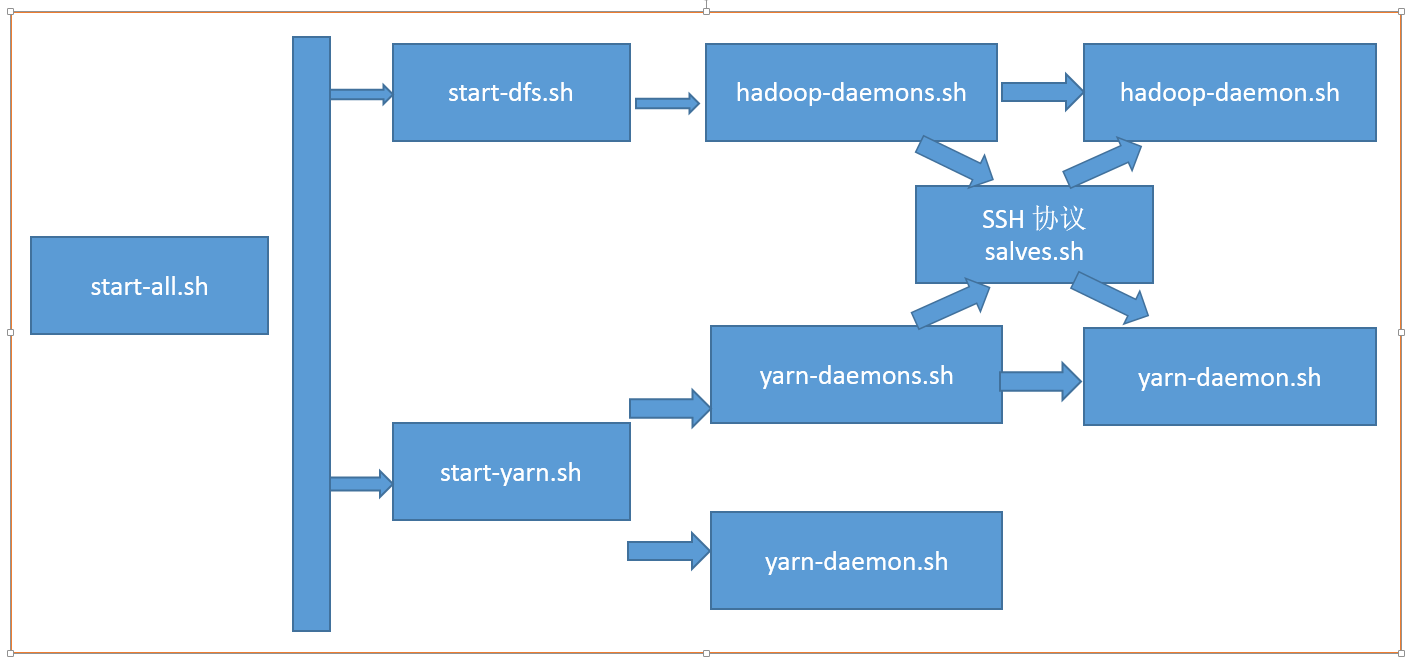
# start proxyserver

#"$bin"/yarn-daemon.sh --config $YARN\_CONF\_DIR start proxyserver

7．总结

start-all方式调用第二种方式

第二种start-dfs.sh start-yarn.sh调用的是hadoop-daemons.sh 和yarn-daemons.sh脚本，而这两个脚本又是分别调用第一种方式的hadoop-daemon.sh & yarn-daemon.sh



14课时SSH协议、公钥登录、无秘钥登录和配置

1. 使用SSH协议登录

SSH是一种网络协议，用于计算机之间的加密登录。

使用SSH协议登录，可以认为登录是安全的，即便是数据被截获，密码不会被泄露。

1. **公钥登录：**远程主机收到用户的登录请求，把自己的公钥发给用户；

用户使用这个公钥，将登录密码进行加密后，发送回来；

远程主机用自己的私钥，解密登录密码，如果密码正确，就同一用户登录。

1. 免密码登录

用户将自己的公钥存储在远程主机上面，登录的时候，远程主机会向用户发送一段随机的字符串，用户用自己的私钥加密之后，再发送回来。远程主机用事前存储的公钥进行解密，如果成功，就证明用户是可信的，直接允许shell登录，不在要求密码。

ssh-keygen –t rsa

ssh-copy-id hadoop@host

在远程主机.ssh文件夹下面就会出现一个authorized\_keys文件。

15课时 hadoop配置文件

在share jar包中分别复制出默认的配置文件core-default.xml hdfs-default.xml mapred-default.xml yarn-default.xml.

1. core-site.xml

<property>

<name>fs.defaultFS</name>

<value>hdfs://hadoop-yarn:8020</value>

</property>

<property>

<name>hadoop.tmp.dir</name>

<value>/opt/hadoop/tmp</value>

</property>

<property>

<name>hadoop.http.staticuser.user</name>

<value>yangtengfei</value>

</property>

1. hdfs-core.xml

<property>

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

<value>file://${hadoop.tmp.dir}/dfs/nn/name</value>

</property>

<property>

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

<value>file://${hadoop.tmp.dir}/dfs/nn/edits</value>

</property>

<property>

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

<value>file://${hadoop.tmp.dir}/dfs/snn/name</value>

</property>

<property>

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

<value>file://${hadoop.tmp.dir}/dfs/snn/edits</value>

</property>

<property>

<name>dfs.namenode.http-address</name>

<value>hadoop-yarn:50070</value>

</property>

<property>

<name>dfs.namenode.secondary.http-address</name>

<value>hadoop-yarn:50090</value>

</property>

1. yarn-site.xml

<property>

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

<value>mapreduce\_shuffle</value>

</property>

<property>

<name>yarn.resourcemanager.hostname</name>

<value>hadoop-yarn</value>

</property>

1. mapred-site.xml

<property>

<name>mapreduce.jobhistory.address</name>

<value>hadoop-yarn:10020</value>

</property>

<property>

<name>mapreduce.jobhistory.webapp.address</name>

<value>hadoop-yarn:19888</value>

<description>MapReduce JobHistory Server Web UI host:port</description>

</property>

1. slaves

DataNode 与NodeManager主机名称，这里全部设置成为一台主机hadoop-yarn

创建对象的目录 /nn

运行之前重新格式化 hdfs namenode –format

16课时 MapReduce运行日志讲解

MapReduce程序日志分为历史作业日志和Container日志

历史作业日志：保存在HDFS文件系统中

Container日志位于logs目录下面的userlogs目录下面。

17课时 阅读官方的Reference

<http://hadoop.apache.org/> 官方网站的参考



Scalable：可扩展性

Distributed：分布式

Computing：计算框架最原始的是MapReduce，随着发展，有了新的众多的计算框架，如Spark Storm Tez S4 -> 统一的管理平台 Yarn

Hadoop源代码下载<http://svn.eu.apache.org/viewvc/hadoop/common/branches/> 使用SVN工具便可以下载，在share/doc目录下面有关于hadoop的说明文档和API

总结：

学习hadoop2.x的云帆大数据视频公开部分，主要讲解了关于hadoop2.x的基础知。在之前我们学习过hadoop1.x系列的知识，并且深入到了MapReduce编程。这里主要讲解的是Hadoop2.x与hadoop1.x的不同之处。因为之前已经学习了hadoop1.x的知识，所以这里学习hadoop2.x比较快。