Hadoop 学习笔记

harli

2015/1/8

目录

[2 Hadoop学习 6](#_Toc410772892)

[2.1 Hadoop介绍 6](#_Toc410772893)

[2.2 NameNode 6](#_Toc410772894)

[2.2.1 NameNode HA 6](#_Toc410772895)

[2.3 MapReduce 7](#_Toc410772896)

[2.4 Yarn 7](#_Toc410772897)

[2.5 SecondaryNameNode 7](#_Toc410772898)

[2.6 Hadoop 安全控制 8](#_Toc410772899)

[3 Llama 8](#_Toc410772900)

[3.1 简介 8](#_Toc410772901)

[3.2 官网资料 8](#_Toc410772902)

[3.3 架构 8](#_Toc410772903)

[3.4 Llama 安装 8](#_Toc410772904)

[3.4.1 安装 llama 8](#_Toc410772905)

[3.4.2 配置 9](#_Toc410772906)

[3.4.3 启动和停止 9](#_Toc410772907)

[3.4.4 配置 HA 9](#_Toc410772908)

[3.4.5 修改 Impala 启动参数 10](#_Toc410772909)

[3.5 使用 11](#_Toc410772910)

[3.5.1 Llama Application Master 11](#_Toc410772911)

[3.5.2 Llama Admin Command Line tool 11](#_Toc410772912)

[3.5.3 Llama Node Manager Auxiliary Service 11](#_Toc410772913)

[4 Hadoop 集群安装、配置与管理 11](#_Toc410772914)

[4.1 集群安装和维护 11](#_Toc410772915)

[4.2 Hadoop 软件的安装和配置 11](#_Toc410772916)

[4.2.1 利用 Yum/Puppet 搭建 Hadoop 集群 12](#_Toc410772917)

[4.2.2 构建 Datanode/Tasktracker 12](#_Toc410772918)

[4.2.3 构建 Namenode(及辅助 Namenode) 13](#_Toc410772919)

[4.2.4 构建 JobTracker 14](#_Toc410772920)

[4.3 集群内存设置 14](#_Toc410772921)

[5 Hadoop 生态系统 15](#_Toc410772922)

[5.1 Apache Hive 数据仓库 15](#_Toc410772923)

[5.2 Apache Cassandra 15](#_Toc410772924)

[5.3 Hbase 15](#_Toc410772925)

[5.4 Apache Pig 15](#_Toc410772926)

[5.5 Apache ZooKeeper 16](#_Toc410772927)

[5.6 Chukwa 16](#_Toc410772928)

[5.7 Apache Mahout 16](#_Toc410772929)

[6 Hadoop 发行商的开源服务与工具 16](#_Toc410772930)

[6.1 Hortonworks 16](#_Toc410772931)

[6.2 MapR Technologies 16](#_Toc410772932)

[6.3 Cloudera 16](#_Toc410772933)

[6.3.1 CDH 16](#_Toc410772934)

[6.3.2 Cloudera Manager 17](#_Toc410772935)

[6.3.3 Crunch 框架 17](#_Toc410772936)

[6.3.4 Cloudera Impala：将Hadoop变为分析型数据库 17](#_Toc410772937)

[6.4 Attunity Ltd. 18](#_Toc410772938)

[6.5 Dataguise Inc. 18](#_Toc410772939)

[6.6 EMC 公司 18](#_Toc410772940)

[6.7 Intel 19](#_Toc410772941)

[6.8 微软 Azure 19](#_Toc410772942)

[7 官网学习 19](#_Toc410772943)

[8 Hadoop2.3+Hive0.12集群部署 19](#_Toc410772944)

[8.1 0 机器说明 19](#_Toc410772945)

[8.2 1 打通无密钥 20](#_Toc410772946)

[8.3 2 安装Hadoop2.3 20](#_Toc410772947)

[8.4 3 安装Hive0.12 28](#_Toc410772948)

[8.5 4 安装Mysql5.6 31](#_Toc410772949)

[8.6 5 Pi计算实例、Hive表的计算实例运行 31](#_Toc410772950)

[9 Hadoop 调试 36](#_Toc410772951)

[10 集群监控 38](#_Toc410772952)

[10.1 Ganglia、Nagios 38](#_Toc410772953)

[10.2 39](#_Toc410772954)

[11 扩展 39](#_Toc410772955)

[11.1 内存技术 39](#_Toc410772956)

[11.2 大规模并行处理（MPP）技术 39](#_Toc410772957)

[12 常用技巧 39](#_Toc410772958)

[12.1 开启垃圾箱(trash)功能 39](#_Toc410772959)

[13 FAQ 40](#_Toc410772960)

[13.1 Win下Hadoop执行报错 40](#_Toc410772961)

[14 参考 41](#_Toc410772962)

# Hadoop学习

## Hadoop介绍

Hadoop有两个核心的组件，即HDFS（分布式文件系统）和MapReduce架构（分布式处理平台）。而YARN的出现，则使Hadoop从一个分布式处理架构蜕变为一个分布式操作系统。

在处理大数据任务时，这个开源框架采用的方法就是将它们拆分成较小部分。在 Hadoop 部署中，数据分析任务会分布于一个计算集群的多个节点中。Hadoop 包含一些核心组件，如Hadoop 分布式文件系统（HDFS）和MapReduce，其中后者是一个负责将处理任务分散到 Hadoop 集群的系统。

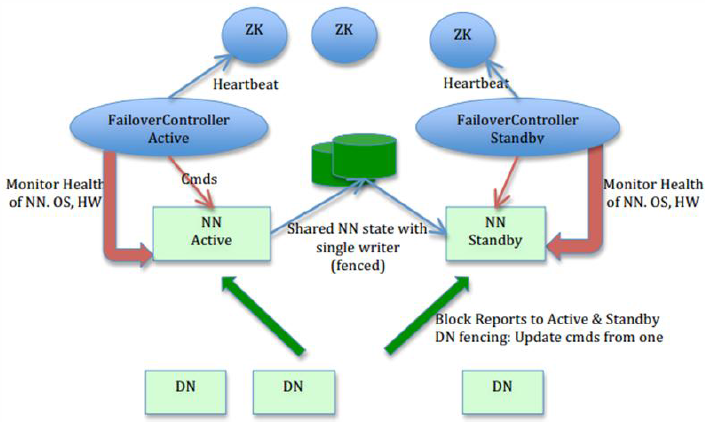
### 基本原理

1. 将文件切分成等大的数据块，存储到多台机器上
2. 将数据切分、容错、负载均衡等功能透明化
3. 可将HDFS看成一个容量巨大、具有高容错性的磁盘

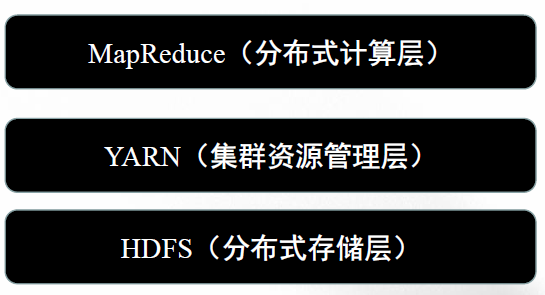
### 应用场景

1. 海量数据的可靠性存储
2. 数据归档

## Hadoop架构



## Hadoop的三大组件



1. 分布式存储系统HDFS（Hadoop Distributed File System）

* 分布式存储系统
* 提供了高可靠性、高扩展性和高吞吐率的数据存储服务

1. 资源管理系统YARN（Yet Another Resource Negotiator）

* 负责集群资源的统一管理和调度

1. 分布式计算框架MapReduce

* 分布式计算框架
* 具有易于编程、高容错性和高扩展性等优点

## NameNode

### NameNode HA

NN 是整个集群可能出现的单点故障。

Hadoop 通过在 hdfs.site.xml 文件的 dfs.name.dir 属性指定保持的 metadata 路径，如果希望保持到多个路径，可以使用逗号分割配置多个路径。

|  |
| --- |
| <property>  <name>dfs.name.dir</name>  <value>/data/cache1/dfs/nn,/data/cache2/dfs/nn</value>  </property> |

Hadoop 官方推荐配置为 metadata 配置多个 path，其中包含一个 NFS 的路径。但根据笔者一次集群严

重故障经验，即使这样，还是导致了所有镜像文件损坏，包括 SNN 上的镜像文件，所以定期备份一个可用的

副本还是很有必要的。

## HDFS

1. 源自于Google的GFS论文

* 发表于2003年10月
* HDFS是GFS克隆版

1. HDFS特点

* 良好的扩展性
* 高容错性
* 适合PB级以上海量数据的存储

## MapReduce

Hadoop 原生 MapReduce 需要 Java 语言编写，但是不会 Java 也没问题，通过 Hadoop streaming 框架管理员可以使用 Python，Shell，Perl 等语言进行 MapReduce 开发，但更简单的办法是安装和使用 Hive 或者 Pig。

## Yarn

对于YARN，大家可以拜读Hortonworks架构师Arun Murthy的[系列文章](http://hortonworks.com/blog/introducing-apache-hadoop-yarn/)，介绍非常详细。

<http://zh.hortonworks.com/blog/introducing-apache-hadoop-yarn/>

YARN资源管理器的应用：

1. [**Storm on YARN**](http://www.searchbi.com.cn/showcontent_76383.htm)**：**Twitter应用的运行在Hadoop上的流计算框架（Yahoo）
2. [**Apache Samza**](http://samza.incubator.apache.org/)**：**基于YARN开发的项目，可以作为Storm的替代品（Apache）
3. [**HOYA**](https://github.com/hortonworks/hoya)**：**HBase on YARN，集群上的HBase部署工具（Hortonworks）
4. [**Weave**](https://github.com/continuuity/weave)**：**基于YARN的封装，用于简化应用部署（Continuuity）
5. [**Giraph**](http://giraph.apache.org/)**：**图形处理系统（Apache）
6. [**Llama**](http://cloudera.github.io/llama/)**：**让外部服务器从YARN获取资源的框架（Cloudera）
7. [**Spark on Yarn**](https://github.com/tweetmagik/spark-yarn)**：**基于内存的分析集群技术
8. [**Tez**](http://hortonworks.com/hadoop/tez/)**：**通用的、高度可定制的框架，用于简化Hadoop中数据处理任务的创建，支持小规模（低延迟）和大规模（高吞吐量）负载（Hortonworks）

### **YARN**是什么

1. Hadoop 2.0新增系统
2. 责集群的资源管理和调度
3. 使得多种计算框架可以运行在一个集群中

### **YARN**的特点

1. 良好的扩展性、高可用性
2. 对多种类型的应用程序进行统一管理和调度
3. 自带了多种多用户调度器，适合共享集群环境

### Yarn对计算框架的支持



计算框架的支持架构



### Yarn的架构



## SecondaryNameNode

SecondaryNameNode（下称 SNN）的主要功能是帮助 NameNode（下称 NN）合并编辑日志，然后将合并后的镜像文件 copy 回 NN，以减少 NN 重启时合并编辑日志所需的时间。

SNN 不是 NN 的热备，但是通过以下步骤可以实现将 SNN 切换为 NN 的目的。

1. 首先，SNN 节点上导入从 NN Copy 过来的镜像文件；
2. 然后修改 SNN 机器名和 IP 与 NN 一致，最后重启集群。

特别注意的是 SNN 的内存配置要与 NN 一致，因为合并编辑日志的工作需要将 metadata 加载到内存完成。

另外，不仅仅是 SNN，任何保存 NN 镜像的节点都可以通过上面步骤变为 NN，只是 SNN 更适合罢了。

## Hadoop 安全控制

Hadoop 安全控制非常简单，只包含简单的权限，即只根据客户端用户名，决定使用权限。它的设计原则是：“避免好人做错事，但不阻止坏人做坏事”。

如果你知道某台 NN 的 IP 和端口，则可以很轻松获取 HDFS 目录结构，并通过修改本机机器用户名伪装成 HDFS 文件所属 owner，对该文件进行删除操作。

通过配置 kerberos，可以实现身份验证。但很多管理员使用更简单有效的办法——通过防火墙对访问 IP进行控制。

# Llama

## 简介

Llama (Low Latency Application MAster) 是一个 Yarn 的 Application Master，用于协调 Impala 和 Yarn 之间的集群资源的管理和监控。Llama 使 Impala 能够获取、使用和释放资源配额，而不需要 Impala 使用 Yarn 管理的 container 进程。Llama 提供了 Thrift API 来和 Yarn 交互。

可以认为Llama 的作用就是使 Impala 能够工作在 YARN 之上，使得 Impala 和 YARN 共享集群资源，提供低延迟的查询。

## 官网资料

1. Llama 官网地址：<http://cloudera.github.io/llama/>
2. Llama 源码：<https://github.com/cloudera/llama>

## 架构

## Llama 安装

### 安装 llama

Llama 需要安装在装有 Yarn 的节点上。

Llama 只能和 Yarn 配合工作，不能用于 MRv1。

在 rhel 系统上安装：

$ sudo yum install llama-master

### 配置

Llama 的配置文件：/etc/llama/conf/ 目录。

llama-site.xml 默认配置参考： <http://cloudera.github.io/llama/llama-site.html>。

### 启动和停止

1. 启动 ： $ sudo service llama start
2. 停止 ： $ sudo service llama stop

### 配置 HA

Llama 使用 Zookeeper 来实现 HA，任一时刻，只有一个 Llama-master 实例是 active的以确保资源不会被分区。

为了从 Yarn 获取资源，Llama 启动 YARN application 并且运行未管理的ApplicationMaster。当一个 Llama 实例宕掉的时候，分配给该实例启动的 application 的所有资源将会被回收，直到这些 application 超时（默认超时时间为10分钟）。当 Llama 运行失败的时候，这些资源将会被杀掉他启动的application的 Llama 回收。

HA 相关配置参数在 /etc/llama/conf/llama-site.xml：

|  |  |  |
| --- | --- | --- |
| 属性 | 描述 | 默认值 |
| llama.am.cluster.id | Cluster ID of the Llama pair, used to differentiate between different Llamas | llama |
| llama.am.ha.enabled | Whether to enable Llama HA | false |
| llama.am.ha.zk-quorum | ZooKeeper quorum to use for leader election and fencing |  |
| llama.am.ha.zk-base | Base znode for leader election and fencing data | /llama |
| llama.am.ha.zk-timeout-ms | The session timeout, in milliseconds, for connections to ZooKeeper quorum | 10000 |
| llama.am.ha.zk-acl | ACLs to control access to ZooKeeper | world:anyone:rwcda |
| llama.am.ha.zk-auth | Authorization information to go with the ACLs |  |

上面必填的两个参数为：

1. llama.am.ha.enabled ： true
2. llama.am.ha.zk-quorum ： cdh1:21088,cdh2:21088

### 修改 Impala 启动参数

使用 jdbc 方式提交查询到 Impala 时，会出现 number of running queries 20 is over limit 20 的异常，这时候在 impala的 源代码中搜索关键字 number of running queries，可以找到<https://github.com/cloudera/Impala/blob/cdh5-1.4_5.1.2/be/src/scheduling/admission-controller.cc>，从源代码中可以看到出现该问题和 Llama 有关系，在找不到 llama 的相关配置时，impala 一个队列中能够接受的最大请求数为 20。代码见:[RequestPoolService.java](https://github.com/cloudera/Impala/blob/c5c475712f88244e15160befaf4e99d6e165a148/fe/src/main/java/com/cloudera/impala/util/RequestPoolService.java" \t "_blank)

|  |
| --- |
| @VisibleForTesting  TPoolConfigResult getPoolConfig(String pool) {  TPoolConfigResult result = **new** TPoolConfigResult();  int maxMemoryMb = allocationConf\_.get().getMaxResources(pool).getMemory();  result.setMem\_limit(  maxMemoryMb == Integer.MAX\_VALUE ? -1 : (long) maxMemoryMb \* ByteUnits.MEGABYTE);  **if** (llamaConf\_ == **null**) { *//llama配置为空*  result.setMax\_requests(LLAMA\_MAX\_PLACED\_RESERVATIONS\_DEFAULT);  result.setMax\_queued(LLAMA\_MAX\_QUEUED\_RESERVATIONS\_DEFAULT);  } **else** {  *// Capture the current llamaConf\_ in case it changes while we're using it.*  Configuration currentLlamaConf = llamaConf\_;  result.setMax\_requests(getLlamaPoolConfigValue(currentLlamaConf, pool,  LLAMA\_MAX\_PLACED\_RESERVATIONS\_KEY,  LLAMA\_MAX\_PLACED\_RESERVATIONS\_DEFAULT)); *//20*  result.setMax\_queued(getLlamaPoolConfigValue(currentLlamaConf, pool,  LLAMA\_MAX\_QUEUED\_RESERVATIONS\_KEY,  LLAMA\_MAX\_QUEUED\_RESERVATIONS\_DEFAULT));  }  LOG.trace("getPoolConfig(pool={}): mem\_limit={}, max\_requests={}, max\_queued={}",  **new** Object[] { pool, result.mem\_limit, result.max\_requests, result.max\_queued });  **return** result;  } |

目前，参考 [Admission Control and Query Queuing](http://www.cloudera.com/content/cloudera-content/cloudera-docs/CDH5/latest/Impala/Installing-and-Using-Impala/ciiu_admission.html)，在不安装和使用 llama 情况下，找到的一种解决办法是：

修改 impala 启动参数（/etc/default/impala），添加 -default\_pool\_max\_requests=-1，该参数设置每一个队列的最大请求数，如果为-1，则表示不做限制。

## 使用

### Llama Application Master

### Llama Admin Command Line tool

### Llama Node Manager Auxiliary Service

# Hadoop 集群安装、配置与管理

## 集群安装和维护

安装和维护 Hadoop 集群涉及大量的管理工作，包括软件安装，设备管理（crontab、iptables 等）、配置分发等。

对于小型集群软件分发和节点管理可以使用 PDSH 这款软件，它可以通过免密钥的 SSH 将文件分发到目标服务器，以及为一组目标设备发送命令并获得反馈。如果是大型集群或者硬件配置差别很大的集群，推荐使用 puppet 这样的工具帮助你维护配置文件，或者通过 Cloudera Manager 以 GUI 的方式的管理集群（注意：Clodera Manager 不是开源软件，免费版最多支持 50 个节点）。

## 集群UI监控界面

Hadoop集群UI界面：

1. dfs.http.address   master:50070
2. dfs.secondary.http.address  SecondaryNode:50090
3. dfs.datanode.http.address  datanode:50075
4. yarn.resourcemanager.webapp.address  resourcemanager:50030
5. mapreduce.jobhistory.webapp.address jobhistory:19888

需要启动：hadoop/sbin/mr-jobhistory-daemon.sh start historyserver。

## Hadoop 软件的安装和配置

Hadoop 集群有多种构建方式：

1. 手工下载 tar 文件并复制到集群中；

2. 利用 Yum 仓库；

3. 利用 Puppet 等自动化部署工具。

我们不建议采用手工方式，那只适合很小的集群(4 节点以下)，而且会带来很多维护和排障上的问题，因为所有的变更都需要用 scp 或 ssh 的方式手工应用到所有的节点上去。

从以下方面来看，利用 Puppet 等部署工具是最佳的选择：

1. 安装
2. 配置
3. 维护
4. 扩展性
5. 监控
6. 排障

Puppet 是 Unix/Linux 下的一个自动化管理引擎，它能基于一个集中式的配置执行增加用户、安装软件包、更新服务器配置等管理任务。我们将主要讲解如何利用 Yum 和 Puppet 来安装 Hadoop。

### 利用 Yum/Puppet 搭建 Hadoop 集群

要利用 Puppet 搭建 Hadoop 集群，首先要符合以下前置条件：

1. 包含所有必需 Hadoop 软件的中央仓库；
2. 用于 Hadoop 部署的 Puppet 装载单(manifest)；
3. 用于 Hadoop 配置管理的 Puppet 装载单；
4. 用于集群维护的框架(主要是 sh 或 ksh 脚本)，以支持集群的 start/stop/restart；
5. 利用 puppet 构建整个服务器(包括操作系统和其它软件)。

注：如果要用 Yum 来安装 Hadoop 集群，则所有服务器应该预先构建完成，包括操作系统和其它软件都应安装完毕，yum 仓库也应在所有节点上设置完毕。

### 构建 Datanode/Tasktracker

如果用 Yum 安装 Datanode/Tasktracker，需在所有数据节点上执行以下命令：

|  |
| --- |
| yum install hadoop-0.20-datanode –y  yum install hadoop-0.20-tasktracker –y |

换成 Puppet 的话，则是：

|  |
| --- |
| class setup\_datanode {  if ($is\_datanode == true) {  make\_dfs\_data\_dir { $hadoop\_disks: }  make\_mapred\_local\_dir { $hadoop\_disks: }  fix\_hadoop\_parent\_dir\_perm { $hadoop\_disks: }  }  # fix hadoop parent dir permissions  define fix\_hadoop\_parent\_dir\_perm() {  …  }  # make dfs data dir  define make\_dfs\_data\_dir() {  …  }  # make mapred local and system dir  define make\_mapred\_local\_dir() {  …  }  } # setup\_datanode |

### 构建 Namenode(及辅助 Namenode)

如果用 Yum 安装 Namenode，需在所有数据节点上执行以下命令：

|  |
| --- |
| yum install hadoop-0.20-namenode –y  yum install hadoop-0.20-secondarynamenode –y |

换成 Puppet 的话，则是：

|  |
| --- |
| class setup\_namenode {  if ($is\_namenode == true or $is\_standby\_namenode == true) {  ...  }  exec {"namenode-dfs-perm":  ...  }  exec { "make ${nfs\_namenode\_dir}/dfs/name":  ...  }  exec { "chgrp ${nfs\_namenode\_dir}/dfs/name":  ...  }  if ($standby\_namenode\_host != "") {  ...  }  exec { "own $nfs\_standby\_namenode\_dir":  ...  }  }  # /standby\_namenode\_hadoop  if ($standby\_namenode\_host != "") {  ...  }  exec { "own $standby\_namenode\_hadoop\_dir":  ...  }  }  }  }  class setup\_secondary\_namenode {  if ($is\_secondarynamenode == true) {  ...  }  ....  }  exec {"namenode-dfs-perm":  ...  }  }  } |

### 构建 JobTracker

如果用 Yum 安装 Jobtracker，需在所有数据节点上执行以下命令:

|  |
| --- |
| yum install hadoop-0.20-jobtracker –y |

换成 Puppet 的话，则是使用与构建 Namenode 相同的装载单，唯一的区别在于，在 Jobtracker 机器上，会启动 Jobtracker——即将该机器上的 is\_jobtracker 设置为 true。

## 集群内存设置

Hadoop 集群安装完毕后，第一件事就是修改 bin/hadoop-evn.sh 文件设置内存。主流节点内存配置为32GB。

典型场景内存设置如下：

1. NN: 15-25 GB
2. JT：2-4GB
3. DN：1-4 GB
4. TT：1-2 GB，Child VM 1-2 GB

集群的使用场景不同相关设置也有不同，如果集群有大量小文件，则要求 NN 内存至少要 20GB，DN 内存至少 2GB。推荐理由：几个组件中 NN 对内存最为敏感，它有单点问题，直接影响到集群的可用性；JT 同样是单点，如果 JT 内存溢出则所有 MapReduce Job 都无法正常执行。

# Hadoop 生态系统

Hadoop 并不完全是孤立的解决方案，它由许多不同的技术组成，其中包括 Pig 过程编程工具、Hive 查询工具、HBase NoSQL 数据库等。事实上，Hadoop 发行版通常并不是完全商业化的私有产品，而是开源与闭源的整合。

## Apache Hive 数据仓库

Hive 是为 Hadoop 设计的数据仓库系统，支持简单数据汇总。

这个数据仓库基础基于Hadoop，并且包含一个基于 SQL 的查询语言 Hive QL（HQL）。HQL 支持复杂分析，但是定制的 Hadoop 解决方案要求能够使用 Java 或 Python 创建自定义的MapReduce 程序。

## Apache Cassandra

Apache Cassandra 是一款为大数据量场景设计的数据库管理系统。它的关键特性是容错、扩容、Hadoop集成性和复制支持。

## Hbase

Hbase 是非关系型的支持容错的分布式数据库，专为存储大量稀疏数据而设计。

## Apache Pig

Apache Pig 由创建数据分析程序的高级语言组成，还包括评估那些应用程序的基础。

## Apache ZooKeeper

Apache ZooKeeper 是用于分布式应用的一款集中服务。它维护配置信息，并提供命名注册、分布式同步和组服务。

## Chukwa

Chukwa 是一个数据收集系统，可以监控大型分布式系统，包括分析结果的一个工具包。

## Apache Mahout

Apache Mahout 项目的目的是生成丰富的实现，在 Hadoop 平台，可扩展的机器学习算法。

# Hadoop 发行商的开源服务与工具

什么是Hadoop 发行版（Hadoop distribution），简单来说就是基于开源Apache Hadoop 进行改造的商业解决方案，其中包括一系列定制的管理工具和软件。

一些Hadoop 发行商所提供的足够创建一个基本的集群运行环境，如 Hortonworks、MapR Technologies 和Cloudera。这些工具已经成为一种管理集群的标准可靠方法。

## Hortonworks

Hortonworks Data Platform 产品。

## MapR Technologies

## Cloudera

<http://www.cloudera.com/content/cloudera/en/home.html>

Cloudera 是一家企业软件公司，该公司在 2008 年开始提供基于 Hadoop 的软件和服务。

### CDH

CDH 基于稳定版 Apache Hadoop，并应用了最新 Bug 修复或者 Feature 的 Patch。

Cloudera 常年坚持季度发行 Update 版本，年度发行 Release 版本，更新速度比 Apache 官方快，而且在实际使用过程中 CDH表现无比稳定，并没有引入新的问题。

CDH 支持 Yum/Apt 包，Tar 包，RPM 包，Cloudera Manager 四种方式安装,官方网站推荐 Yum/Apt 方式安装.

优点：

1. 联网安装、升级，非常方便。可以下载 rpm 包到本地，使用 Local Yum 方式安装。
2. 自动下载依赖软件包，比如要安装 Hive，则会级联下载、安装 Hadoop。
3. Hadoop 生态系统包自动匹配，不需要寻找与当前 Hadoop 匹配的 Hbase，Flume，Hive 等软件，Yum/Apt 会根据当前安装 Hadoop 版本自动寻找匹配版本的软件包，并保证兼容性。
4. 自动创建相关目录并软链到合适的地方（如 conf 和 logs 等目录）；自动创建 hdfs, mapred 用户，hdfs用户是 HDFS 的最高权限用户，mapred 用户则负责 mapreduce 执行过程中相关目录的权限。

### Cloudera Manager

Cloudera Manager 已经成为一个运行 Hadoop 集群的优质工具

Cloudera Manager 属于 CDH Enterprise Hadoop 发行版的一部分。免费版 CDH 的 Manager 工具支持最多 50 个节点。Cloudera Manager 完整版支持无限主机数，属于评阅版 CDH。

### Crunch 框架

Cloudera 的 Crunch 框架（一个 Java 程序库）也是一个可以简化编码的工具。Crunch 是Cloudera的一个开发项目，它已经进入Apache孵化器。孵化器的作用就是在将代码贡献给Apache软件基金会时继续供外部组织使用。Crunch能够帮助编写一些自定义 MapReduce，加快 Hadoop 开发过程，同时有利于优化复杂的数据转换任务。

Crunch 与Karmasphere对比：

Karmasphere 公司的开发环境创建自定义的 MapReduce 程序。Karmasphere与 Crunch 的不同点是，前者是一个可下载的软件工具，而 Crunch 更像是一个 API 环境。

### Cloudera Impala：将Hadoop变为分析型数据库

Impala 旨在改进查询性能。

与Hive之间的异同点包括：

1. Impala的查询语言(HQL，相当于SQL)、元数据以及JDBC/ODBC驱动等都与Hive兼容；
2. Impala无需通过Hadoop MapReduce使用，这点与Hive有所不同；
3. 同MapReduce+Hive方式相比，Impala不会将中间结果存储到磁盘，能够显著提升性能。但对于运行时间较长的查询则存在单点故障的风险，一旦出现故障将需要重新运行查询；
4. 在Impala的第一版中将不包含部分Hive语法，如对UDF的支持；

总体来说，虽然Impala同Hadapt相比还略显不成熟，但是它还是有很多优势：

1. 它是免费的；
2. 它的可管理性更好；
3. 它针对一些环境的性能是非常好的；

Cloudera公司已经开放了Impala的源代码，用户可以通过[Github](https://github.com/cloudera/impala)下载。

## [Attunity Ltd.](http://www.searchbi.com.cn/showcontent_64525.htm)

专注于Hadoop数据传输。

## Dataguise Inc.

Dataguise 公司是一家数据安全公司，在 2012 年该公司推出了一款针对 Hadoop 的数据保护和风险评估。

专注于Hadoop数据安全性。推出了用于保护Hadoop数据的技术，在数据存储和由数据分析工具提取时提供保护。

Dataguise的DG for Hadoop还支持访问控制。

## EMC 公司

Pivotal HD Apache Hadoop。

虽然 EMC 还没有给出太多关于 Pivotal HD 的技术细节，但我们可以看出它的核心源自 EMC Greenplum长期以来的一个项目——HAWQ，能够以原生态的方式，将管道化的传统数据库集成到 Hadoop 当中去。然而纵观整个业界，SQL 与 Hadoop 的结合已经并不是什么新鲜的课题。

通过整合 Greenplum 数据库，Pivotal HD 向业务用户打开了大数据分析之门。但分析师提醒，EMC 的Hadoop 发行版显然只是针对自身的 Greenplum 数据库进行了优化，而是用其他外部数据库产品的用户则无法获得这一优势。

## Intel

Apache Hadoop

## 微软 Azure

Hadoop 服务HadoopOnAzure。

# 官网学习

# [Hadoop2.3+Hive0.12集群部署](http://www.cnblogs.com/Scott007/p/3614960.html)

<http://www.cnblogs.com/Scott007/p/3614960.html>

## 0 机器说明

|  | |
| --- | --- |
| **IP** | **Role** |
| **192.168.1.106** | **NameNode、DataNode、NodeManager、ResourceManager** |
| **192.168.1.107** | **SecondaryNameNode、NodeManager、DataNode** |
| **192.168.1.108** | **NodeManager、DataNode** |
| **192.168.1.106** | **HiveServer** |

## 1 打通无密钥

    配置HDFS，首先就得把机器之间的无密钥配置上。我们这里为了方便，把机器之间的双向无密钥都配置上。

**(1)**产生RSA密钥信息

ssh-keygen -t rsa

一路回车，直到产生一个图形结构，此时便产生了RSA的私钥id\_rsa和公钥id\_rsa.pub，位于/home/user/.ssh目录中。

**(2)**将所有机器节点的ssh证书公钥拷贝至/home/user/.ssh/authorized\_keys文件中，三个机器都一样。

**(3)**切换到root用户，修改/etc/ssh/sshd\_config文件，配置：

RSAAuthentication yes

PubkeyAuthentication yes

AuthorizedKeysFile .ssh/authorized\_keys

**(4)**重启ssh服务：service sshd restart

**(5)**使用ssh服务，远程登录：

http://images.cnitblog.com/i/392365/201403/202240390524436.jpg

 ssh配置成功。

## 2 安装Hadoop2.3

     将对应的hadoop2.3的tar包解压缩到本地之后，主要就是修改配置文件，文件的路径都在etc/hadoop中，下面列出几个主要的。

(1)core-site.xml

[复制代码](javascript:void(0);)

1 <configuration>

2 <property>

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

4 <value>file:/home/sdc/tmp/hadoop-${user.name}</value>

5 </property>

6 <property>

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

8 <value>hdfs://192.168.1.106:9000</value>

9 </property>

10 </configuration>

[复制代码](javascript:void(0);)

(2)hdfs-site.xml

[复制代码](javascript:void(0);)

1 <configuration>

2 <property>

3 <name>dfs.replication</name>

4 <value>3</value>

5 </property>

6 <property>

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

8 <value>192.168.1.107:9001</value>

9 </property>

10 <property>

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

12 <value>file:/home/sdc/dfs/name</value>

13 </property>

14 <property>

15 <name>dfs.datanode.data.dir</name>

16 <value>file:/home/sdc/dfs/data</value>

17 </property>

18 <property>

19 <name>dfs.replication</name>

20 <value>3</value>

21 </property>

22 <property>

23 <name>dfs.webhdfs.enabled</name>

24 <value>true</value>

25 </property>

26 </configuration>

[复制代码](javascript:void(0);)

(3)hadoop-env.sh

主要是将其中的JAVA\_HOME赋值：

export JAVA\_HOME=/usr/local/jdk1.6.0\_27

(4)mapred-site.xml

[复制代码](javascript:void(0);)

1 <configuration>

2 <property>

3 <!-- 使用yarn作为资源分配和任务管理框架 -->

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

5 <value>yarn</value>

6 </property>

7 <property>

8 <!-- JobHistory Server地址 -->

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

10 <value>centos1:10020</value>

11 </property>

12 <property>

13 <!-- JobHistory WEB地址 -->

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

15 <value>centos1:19888</value>

16 </property>

17 <property>

18 <!-- 排序文件的时候一次同时最多可并行的个数 -->

19 <name>mapreduce.task.io.sort.factor</name>

20 <value>100</value>

21 </property>

22 <property>

23 <!-- reuduce shuffle阶段并行传输数据的数量 -->

24 <name>mapreduce.reduce.shuffle.parallelcopies</name>

25 <value>50</value>

26 </property>

27 <property>

28 <name>mapred.system.dir</name>

29 <value>file:/home/sdc/Data/mr/system</value>

30 </property>

31 <property>

32 <name>mapred.local.dir</name>

33 <value>file:/home/sdc/Data/mr/local</value>

34 </property>

35 <property>

36 <!-- 每个Map Task需要向RM申请的内存量 -->

37 <name>mapreduce.map.memory.mb</name>

38 <value>1536</value>

39 </property>

40 <property>

41 <!-- 每个Map阶段申请的Container的JVM参数 -->

42 <name>mapreduce.map.java.opts</name>

43 <value>-Xmx1024M</value>

44 </property>

45 <property>

46 <!-- 每个Reduce Task需要向RM申请的内存量 -->

47 <name>mapreduce.reduce.memory.mb</name>

48 <value>2048</value>

49 </property>

50 <property>

51 <!-- 每个Reduce阶段申请的Container的JVM参数 -->

52 <name>mapreduce.reduce.java.opts</name>

53 <value>-Xmx1536M</value>

54 </property>

55 <property>

56 <!-- 排序内存使用限制 -->

57 <name>mapreduce.task.io.sort.mb</name>

58 <value>512</value>

59 </property>

60 </configuration>

[复制代码](javascript:void(0);)

　　注意上面的几个内存大小的配置，其中Container的大小一般都要小于所能申请的最大值，否则所运行的Mapreduce任务可能无法运行。

(5)yarn-site.xml

[复制代码](javascript:void(0);)

1 <configuration>

2 <property>

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

4 <value>mapreduce\_shuffle</value>

5 </property>

6 <property>

7 <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>

8 <value>org.apache.hadoop.mapred.ShuffleHandler</value>

9 </property>

10 <property>

11 <name>yarn.resourcemanager.address</name>

12 <value>centos1:8080</value>

13 </property>

14 <property>

15 <name>yarn.resourcemanager.scheduler.address</name>

16 <value>centos1:8081</value>

17 </property>

18 <property>

19 <name>yarn.resourcemanager.resource-tracker.address</name>

20 <value>centos1:8082</value>

21 </property>

22 <property>

23 <!-- 每个nodemanager可分配的内存总量 -->

24 <name>yarn.nodemanager.resource.memory-mb</name>

25 <value>2048</value>

26 </property>

27 <property>

28 <name>yarn.nodemanager.remote-app-log-dir</name>

29 <value>${hadoop.tmp.dir}/nodemanager/remote</value>

30 </property>

31 <property>

32 <name>yarn.nodemanager.log-dirs</name>

33 <value>${hadoop.tmp.dir}/nodemanager/logs</value>

34 </property>

35 <property>

36 <name>yarn.resourcemanager.admin.address</name>

37 <value>centos1:8033</value>

38 </property>

39 <property>

40 <name>yarn.resourcemanager.webapp.address</name>

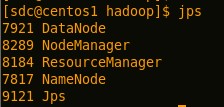
41 <value>centos1:8088</value>

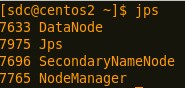
42 </property>

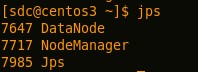
43 </configuration>

[复制代码](javascript:void(0);)

    此外，配置好对应的HADOOP\_HOME环境变量之后，将当前hadoop文件发送到所有的节点，在sbin目录中有**start-all.sh**脚本，启动可见：

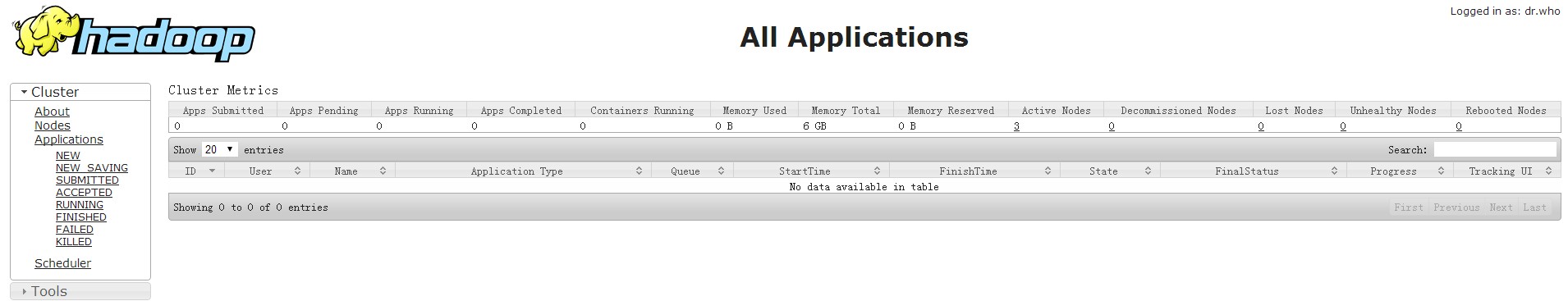




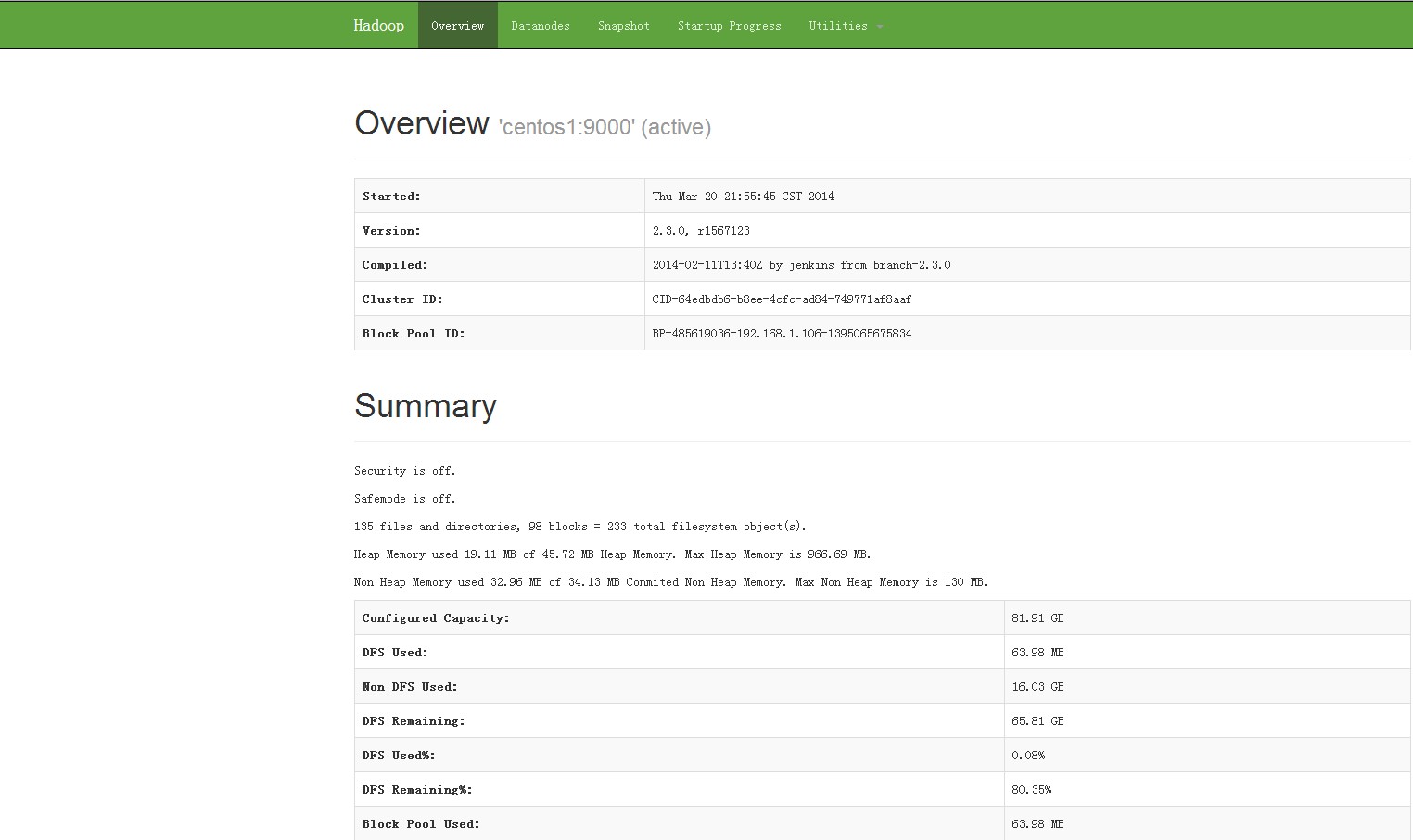


    启动完成之后，有如下两个WEB界面：

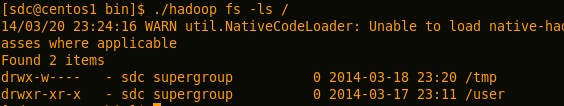
http://192.168.1.106:8088/cluster



http://192.168.1.106:50070/dfshealth.html



使用最简单的命令检查下HDFS：



## 3 安装Hive0.12

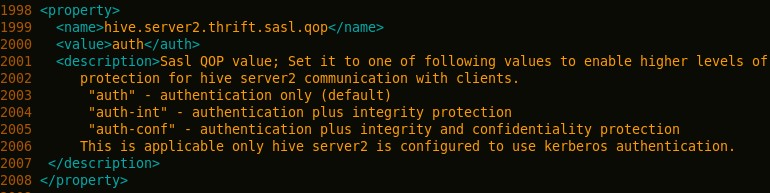
    将Hive的tar包解压缩之后，首先配置下HIVE\_HOME的环境变量。然后便是一些配置文件的修改：

(1)hive-env.sh

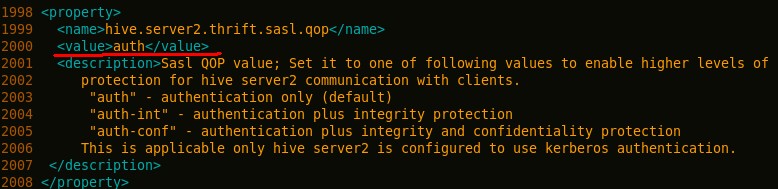
将其中的HADOOP\_HOME变量修改为当前系统变量值。

(2)hive-site.xml

* 修改hive.server2.thrift.sasl.qop属性



修改为：



1. 将hive.metastore.schema.verification对应的值改为false

        强制metastore的schema一致性，开启的话会校验在metastore中存储的信息的版本和hive的jar包中的版本一致性，并且关闭自动schema迁移，用户必须手动的升级hive并且迁移schema，关闭的话只会在版本不一致时给出警告。

* 修改hive的元数据存储位置，改为mysql存储：

[复制代码](javascript:void(0);)

1 <property>

2 <name>javax.jdo.option.ConnectionURL</name>

3 <value>jdbc:mysql://localhost:3306/hive?characterEncoding=UTF-8</value>

4 <description>JDBC connect string for a JDBC metastore</description>

5 </property>

6

7 <property>

8 <name>javax.jdo.option.ConnectionDriverName</name>

9 <value>com.mysql.jdbc.Driver</value>

10 <description>Driver class name for a JDBC metastore</description>

11 </property>

12

13 <property>

14 <name>javax.jdo.PersistenceManagerFactoryClass</name>

15 <value>org.datanucleus.api.jdo.JDOPersistenceManagerFactory</value>

16 <description>class implementing the jdo persistence</description>

17 </property>

18

19 <property>

20 <name>javax.jdo.option.DetachAllOnCommit</name>

21 <value>true</value>

22 <description>detaches all objects from session so that they can be used after transaction is committed</description>

23 </property>

24

25 <property>

26 <name>javax.jdo.option.NonTransactionalRead</name>

27 <value>true</value>

28 <description>reads outside of transactions</description>

29 </property>

30

31 <property>

32 <name>javax.jdo.option.ConnectionUserName</name>

33 <value>hive</value>

34 <description>username to use against metastore database</description>

35 </property>

36

37 <property>

38 <name>javax.jdo.option.ConnectionPassword</name>

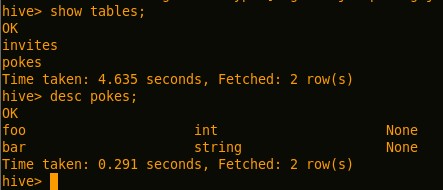
39 <value>123</value>

40 <description>password to use against metastore database</description>

41 </property>

[复制代码](javascript:void(0);)

    在bin下启动hive脚本，运行几个hive语句：



## 4 安装Mysql5.6

 见<http://www.cnblogs.com/Scott007/p/3572604.html>

## 5 Pi计算实例、Hive表的计算实例运行

     在Hadoop的安装目录bin子目录下，执行hadoop自带的示例，pi的计算，命令为：

./hadoop jar ../share/hadoop/mapreduce/hadoop-mapreduce-examples-2.3.0.jar pi 10 10

运行日志为：

[复制代码](javascript:void(0);)

1 Number of Maps = 10

2 Samples per Map = 10

3 14/03/20 23:50:04 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

4 Wrote input for Map #0

5 Wrote input for Map #1

6 Wrote input for Map #2

7 Wrote input for Map #3

8 Wrote input for Map #4

9 Wrote input for Map #5

10 Wrote input for Map #6

11 Wrote input for Map #7

12 Wrote input for Map #8

13 Wrote input for Map #9

14 Starting Job

15 14/03/20 23:50:06 INFO client.RMProxy: Connecting to ResourceManager at centos1/192.168.1.106:8080

16 14/03/20 23:50:07 INFO input.FileInputFormat: Total input paths to process : 10

17 14/03/20 23:50:07 INFO mapreduce.JobSubmitter: number of splits:10

18 14/03/20 23:50:08 INFO mapreduce.JobSubmitter: Submitting tokens for job: job\_1395323769116\_0001

19 14/03/20 23:50:08 INFO impl.YarnClientImpl: Submitted application application\_1395323769116\_0001

20 14/03/20 23:50:08 INFO mapreduce.Job: The url to track the job: http://centos1:8088/proxy/application\_1395323769116\_0001/

21 14/03/20 23:50:08 INFO mapreduce.Job: Running job: job\_1395323769116\_0001

22 14/03/20 23:50:18 INFO mapreduce.Job: Job job\_1395323769116\_0001 running in uber mode : false

23 14/03/20 23:50:18 INFO mapreduce.Job: map 0% reduce 0%

24 14/03/20 23:52:21 INFO mapreduce.Job: map 10% reduce 0%

25 14/03/20 23:52:27 INFO mapreduce.Job: map 20% reduce 0%

26 14/03/20 23:52:32 INFO mapreduce.Job: map 30% reduce 0%

27 14/03/20 23:52:34 INFO mapreduce.Job: map 40% reduce 0%

28 14/03/20 23:52:37 INFO mapreduce.Job: map 50% reduce 0%

29 14/03/20 23:52:41 INFO mapreduce.Job: map 60% reduce 0%

30 14/03/20 23:52:43 INFO mapreduce.Job: map 70% reduce 0%

31 14/03/20 23:52:46 INFO mapreduce.Job: map 80% reduce 0%

32 14/03/20 23:52:48 INFO mapreduce.Job: map 90% reduce 0%

33 14/03/20 23:52:51 INFO mapreduce.Job: map 100% reduce 0%

34 14/03/20 23:52:59 INFO mapreduce.Job: map 100% reduce 100%

35 14/03/20 23:53:02 INFO mapreduce.Job: Job job\_1395323769116\_0001 completed successfully

36 14/03/20 23:53:02 INFO mapreduce.Job: Counters: 49

37 File System Counters

38 FILE: Number of bytes read=226

39 FILE: Number of bytes written=948145

40 FILE: Number of read operations=0

41 FILE: Number of large read operations=0

42 FILE: Number of write operations=0

43 HDFS: Number of bytes read=2670

44 HDFS: Number of bytes written=215

45 HDFS: Number of read operations=43

46 HDFS: Number of large read operations=0

47 HDFS: Number of write operations=3

48 Job Counters

49 Launched map tasks=10

50 Launched reduce tasks=1

51 Data-local map tasks=10

52 Total time spent by all maps in occupied slots (ms)=573584

53 Total time spent by all reduces in occupied slots (ms)=20436

54 Total time spent by all map tasks (ms)=286792

55 Total time spent by all reduce tasks (ms)=10218

56 Total vcore-seconds taken by all map tasks=286792

57 Total vcore-seconds taken by all reduce tasks=10218

58 Total megabyte-seconds taken by all map tasks=440512512

59 Total megabyte-seconds taken by all reduce tasks=20926464

60 Map-Reduce Framework

61 Map input records=10

62 Map output records=20

63 Map output bytes=180

64 Map output materialized bytes=280

65 Input split bytes=1490

66 Combine input records=0

67 Combine output records=0

68 Reduce input groups=2

69 Reduce shuffle bytes=280

70 Reduce input records=20

71 Reduce output records=0

72 Spilled Records=40

73 Shuffled Maps =10

74 Failed Shuffles=0

75 Merged Map outputs=10

76 GC time elapsed (ms)=710

77 CPU time spent (ms)=71800

78 Physical memory (bytes) snapshot=6531928064

79 Virtual memory (bytes) snapshot=19145916416

80 Total committed heap usage (bytes)=5696757760

81 Shuffle Errors

82 BAD\_ID=0

83 CONNECTION=0

84 IO\_ERROR=0

85 WRONG\_LENGTH=0

86 WRONG\_MAP=0

87 WRONG\_REDUCE=0

88 File Input Format Counters

89 Bytes Read=1180

90 File Output Format Counters

91 Bytes Written=97

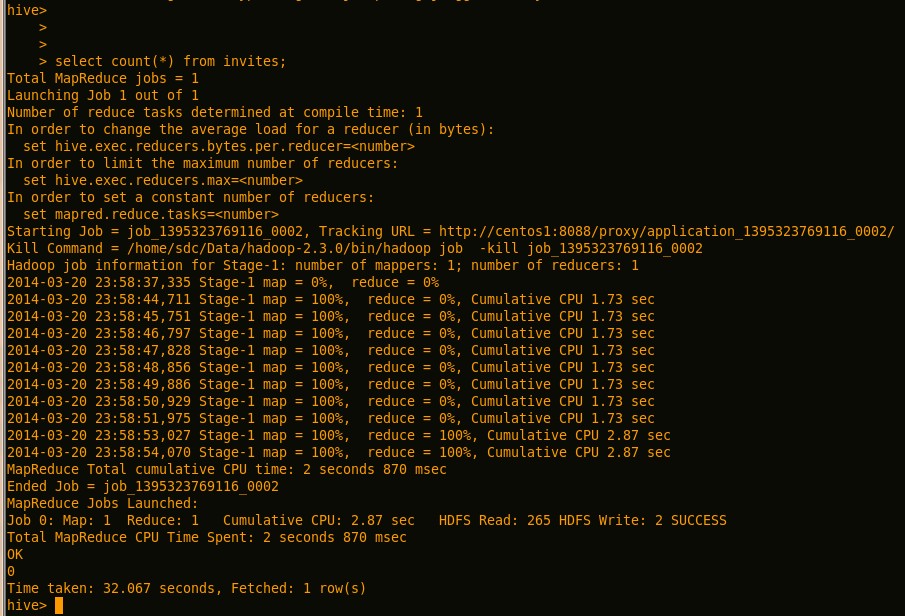
92 Job Finished in 175.556 seconds

93 Estimated value of Pi is 3.20000000000000000000

[复制代码](javascript:void(0);)

    如果运行不起来，那说明HDFS的配置有问题啊！

    Hive中执行count等语句，可以触发mapduce任务：



    如果运行的时候出现类似于如下的错误：

Error in metadata: java.lang.RuntimeException: Unable to instantiate org.apache.hadoop.hive.metastore.HiveMetaStoreClient

说明元数据存储有问题，可能是以下两方面的原因：

(1)HDFS的元数据存储有问题：

$HADOOP\_HOME/bin/hadoop fs -mkdir /tmp

$HADOOP\_HOME/bin/hadoop fs -mkdir /user/hive/warehouse

$HADOOP\_HOME/bin/hadoop fs -chmod g+w /tmp

$HADOOP\_HOME/bin/hadoop fs -chmod g+w /user/hive/warehouse

(2)Mysql的授权有问题：

在mysql中执行如下命令，其实就是给Mysql中的Hive数据库赋权

grant all on db.\* to hive@'%' identified by '密码';(使用户可以远程连接Mysql)

grant all on db.\* to hive@'localhost' identified by '密码';(使用户可以本地连接Mysql)

flush privileges;

    具体哪方面的原因，可以查看hive的日志。

s

# Hadoop 调试

调试环境：源码环境 + 部署环境

## Eclipse的Hadoop插件

在Eclipse下调试时，先安装插件，如Hadoop2.x eclispe-plugin。

Hadoop2.X的插件一般都需要自己编译，也可以从网上下载。

<https://github.com/winghc/hadoop2x-eclipse-plugin>

Hadoop installation directory里填写Win下的调试环境的hadoop home地址。

说明：

1. hadoop eclipse插件不是必须的，其作用在我看来就是如下三点(这个是一个错误的认识，具体请参考<http://zy19982004.iteye.com/blog/2031172>)。study-hadoop是一个普通project，直接运行(不通过Run on Hadoop这只大象)，一样可以调试到MapReduce。
2. 对hadoop中的文件可视化。
3. 创建MapReduce Project时帮你引入依赖的jar。
4. Configuration conf = new Configuration();时就已经包含了所有的配置信息。
5. 还是自己下载hadoop2.2的源码编译好，应该是不会有任何问题的(没有亲测)。

## Hadoop调试

<http://www.lifeba.org/arch/hadoop_debug.html>

### Java调试的JVM选项设置

JVM参数：

\_OPTS -Xdebug -Xrunjdwp:transport=dt\_socket,address=6666,server=y,suspend=y"

### 调试HDFS

调试对象NameNode和SecondaryNameNnode

1. bin/hdfs 脚本中

|  |
| --- |
| # 在namenode下加入  HADOOP\_OPTS="$HADOOP\_OPTS -Xdebug -Xrunjdwp:transport=dt\_socket,address=7777,server=y,suspend=y"  # 在secondarynamenode下加入  HADOOP\_OPTS="$HADOOP\_OPTS -Xdebug -Xrunjdwp:transport=dt\_socket,address=7778,server=y,suspend=y" |

如下图所示：

**[](http://www.lifeba.org/wp-content/uploads/2012/05/lifeba.org_hadoop_debug_1.jpg)**

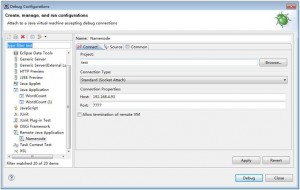
1. 执行相应的脚本，然后通过Eclipse IDE连接来下断调试

启动服务./start-dfs.sh，停止服务./stop-dfs.sh

启动后会停留在 Listening for transport dt\_socket at address: 7777 ，IDE连接后才会继续。

### 调试eclipse hadoop

打开Eclipse的 Debug Configurations ，新建一个Remote Java Application。指定project工程项目，以及需要调试的hadoop节点的ip和端口。

**[](http://www.lifeba.org/wp-content/uploads/2012/05/lifeba.org_hadoop_debug_3.jpg)**

在hadoop节点启动后，点击debug来连接远程的hadoop程序进行调试。hadoop节点在调试启动方式下，必须IDE连接上去后，才会继续，否则会停留在  
上面的Listening for transport dt\_socket at address: 7777，直到有IDE连接成功。

说明：

1. 调试datanode

在datanode节点下的bin/hdfs 脚本中的datanode节点加入JVM调试参数：

|  |
| --- |
| HADOOP\_OPTS="$HADOOP\_OPTS -Xdebug -Xrunjdwp:transport=dt\_socket,address=7777,server=y,suspend=y" |

然后启动脚本./hdfs datanode datanode，打开调试监听，这样就可以连调。

1. 不能使用外层hadoop启动或者start-all.sh启动，调试的话必须直接使用实际调用的sh启动。

如 ./hadoop namenode -format 要换为 ./hdfs namenode -format

### 调试MapReduce

1. 调试MapReduce先启动HDFS服务，然后在bin/mapred 脚本中

|  |
| --- |
| #jobtracker下加入  HADOOP\_OPTS="$HADOOP\_OPTS -Xdebug -Xrunjdwp:transport=dt\_socket,address=6666,server=y,suspend=y"  # tasktracker下加入  HADOOP\_OPTS="$HADOOP\_OPTS -Xdebug -Xrunjdwp:transport=dt\_socket,address=6667,server=y,suspend=y" |

如下图：

[](http://www.lifeba.org/wp-content/uploads/2012/05/lifeba.org_hadoop_debug_2.jpg)

1. 执行相应的脚本，同样通过Eclipse IDE连接来下断调试。

启动服务start-mapred.sh，停止服务stop-mapred.sh

如果需要驱动某个节点，可以执行单个启动命令。./mapred jobtracker 后面指定需要开启的服务的名称。

## [Win7 Eclipse调试Hadoop2.2-Mapreduce](http://zy19982004.iteye.com/blog/2024467)

<http://zy19982004.iteye.com/blog/2024467>

### **调试环境**

Centos5.3 + Hadoop2.2 + Hbase0.96.1.1

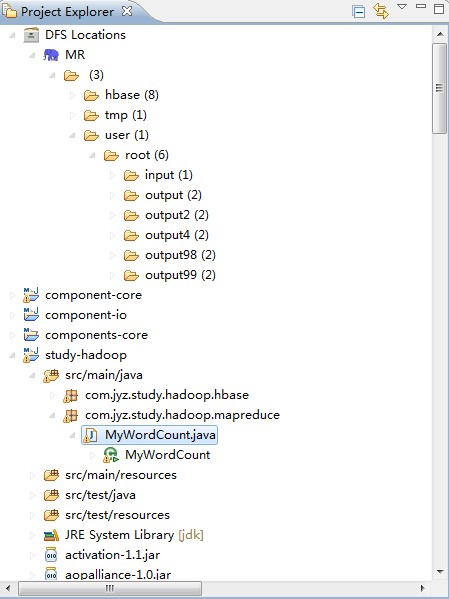
Win下搭建调试环境。

### 调试目的

在Win7下通过 Eclipse调试MapReduce。

创建一个MapReduce Project的调试环境。

### 调试结果

MapReduce的正确输出output99。

## **FAQ**

运行的问题及其解决方法如下：

### Win下Hadoop执行报错问题

参考FAQ的对应章节

代码如下：

* 1. **public** **static** **final** String getQualifiedBinPath(String executable)
  2. **throws** IOException {
  3. // construct hadoop bin path to the specified executable
  4. String fullExeName = HADOOP\_HOME\_DIR + File.separator + "bin"
  5. + File.separator + executable;
  7. File exeFile = **new** File(fullExeName);
  8. **if** (!exeFile.exists()) {
  9. **throw** **new** IOException("Could not locate executable " + fullExeName
  10. + " in the Hadoop binaries.");
  11. }
  13. **return** exeFile.getCanonicalPath();
  14. }
  16. **private** **static** String HADOOP\_HOME\_DIR = checkHadoopHome();
  17. **private** **static** String checkHadoopHome() {
  19. // first check the Dflag hadoop.home.dir with JVM scope
  20. String home = System.getProperty("hadoop.home.dir");
  22. // fall back to the system/user-global env variable
  23. **if** (home == **null**) {
  24. home = System.getenv("HADOOP\_HOME");
  25. }
  26. ...
  27. }

### setPermission问题

* 1. at org.apache.hadoop.util.Shell.execCommand(Shell.java:661)
  2. at org.apache.hadoop.fs.RawLocalFileSystem.setPermission(RawLocalFileSystem.java:639)
  3. at org.apache.hadoop.fs.RawLocalFileSystem.mkdirs(RawLocalFileSystem.java:435)

跟踪代码org.apache.hadoop.util.Shell.java

* 1. **public** **static** String[] getSetPermissionCommand(String perm, **boolean** recursive,
  2. String file) {
  3. String[] baseCmd = getSetPermissionCommand(perm, recursive);
  4. String[] cmdWithFile = Arrays.copyOf(baseCmd, baseCmd.length + 1);
  5. cmdWithFile[cmdWithFile.length - 1] = file;
  6. **return** cmdWithFile;
  7. }
  9. /\*\* Return a command to set permission \*/
  10. **public** **static** String[] getSetPermissionCommand(String perm, **boolean** recursive) {
  11. **if** (recursive) {
  12. **return** (WINDOWS) ? **new** String[] { WINUTILS, "chmod", "-R", perm }
  13. : **new** String[] { "chmod", "-R", perm };
  14. } **else** {
  15. **return** (WINDOWS) ? **new** String[] { WINUTILS, "chmod", perm }
  16. : **new** String[] { "chmod", perm };
  17. }
  18. }

 cmdWithFile数组的内容为{"D:\Hadoop\tar\hadoop-2.2.0\hadoop-2.2.0\bin\winutils.exe", "chmod", "755", "xxxfile"}，我把这个单独在cmd里执行了一下，发现

* 1. 无法启动此程序，因为计算机中丢失 MSVCR100.dll

 那就下载一个呗<http://files.cnblogs.com/sirkevin/msvcr100.rar>，丢到C:\Windows\System32里面。再次cmd执行，又来了问题

* 1. 应用程序无法正常启动(0xc000007b)

 下载<http://blog.csdn.net/vbcom/article/details/7245186> DirectX\_Repair来解决这个问题吧。记得修复完后要重启电脑。搞定后cmd试一下，很棒。

### UnsatisfiedLinkError问题

报错信息：

* 1. Exception in thread "main" java.lang.UnsatisfiedLinkError: org.apache.hadoop.io.nativeio.NativeIO$Windows.access0(Ljava/lang/String;I)Z

跟踪代码：

* 1. /\*\* Windows only method used to check if the current process has requested
  2. \*  access rights on the given path. \*/
  3. **private** **static** **native** **boolean** access0(String path, **int** requestedAccess);

显然缺少dll文件，还记得<https://github.com/srccodes/hadoop-common-2.2.0-bin>下载的东西吧，里面就有hadoop.dll，最好的方法就是用hadoop-common-2.2.0-bin-master/bin目录替换本地hadoop的bin目录，并在环境变量里配置PATH=HADOOP\_HOME/bin，重启电脑。

* 1. Exception in thread "main" java.lang.UnsatisfiedLinkError: org.apache.hadoop.io.nativeio.NativeIO$Windows.access0(Ljava/lang/String;I)Z

跟踪代码org.apache.hadoop.util.NativeCodeLoader.java：

* 1. **static** {
  2. // Try to load native hadoop library and set fallback flag appropriately
  3. **if**(LOG.isDebugEnabled()) {
  4. LOG.debug("Trying to load the custom-built native-hadoop library...");
  5. }
  6. **try** {
  7. System.loadLibrary("hadoop");
  8. LOG.debug("Loaded the native-hadoop library");
  9. nativeCodeLoaded = **true**;
  10. } **catch** (Throwable t) {
  11. // Ignore failure to load
  12. **if**(LOG.isDebugEnabled()) {
  13. LOG.debug("Failed to load native-hadoop with error: " + t);
  14. LOG.debug("java.library.path=" +
  15. System.getProperty("java.library.path"));
  16. }
  17. }
  19. **if** (!nativeCodeLoaded) {
  20. LOG.warn("Unable to load native-hadoop library for your platform... " +
  21. "using builtin-java classes where applicable");
  22. }
  23. }

 这里报错如下：

* 1. DEBUG org.apache.hadoop.util.NativeCodeLoader - Failed to load **native**-hadoop with error: java.lang.UnsatisfiedLinkError: HADOOP\_HOME\bin\hadoop.dll: Can't load AMD 64-bit .dll on a IA 32-bit platform

可以看出是32位jdk的问题，替换成64位：

* 1. 2014-03-11 19:43:08,805 DEBUG org.apache.hadoop.util.NativeCodeLoader - Trying to load the custom-built **native**-hadoop library...
  2. 2014-03-11 19:43:08,812 DEBUG org.apache.hadoop.util.NativeCodeLoader - Loaded the **native**-hadoop library

# 集群监控

## Ganglia、Nagios

使用 Ganglia 和 Nagios 监控你的集群。

Ganglia 可以帮你记录集群状态，方便诊断问题；Nagios 可以再遇到问题时第一时间通知你。

当运行一个大型 mapreduce 作业时，我们通常非常关心该作业对 TaskTracker（下称 TT）CPU、内存、磁盘，以及整个网络的带宽情况，这时候就需要 Ganglia 这个工具为我们生成相关图表来诊断、分析问题。

Ganglia 可以监控集群状态，但当你的服务器 down 机或者某个 TT 挂掉，它却无法通知到你，这时我们可以使用 Nagios 这款告警软件，它可以配置邮件告警和短息告警。通过编写 plugins，可以实现自己的监控功能。

我们的集群目前做了如下监控：

1. NameNode、JobTracker 内存
2. DataNode 和 TaskTracker 运行状态
3. NFS 服务状态
4. 磁盘使用情况
5. 服务器负载状态

## 

# 扩展

## 内存技术

如 SAP HANA，可以在高度结构化数据上实现低延迟的查询结果。

## 大规模并行处理（MPP）技术

大规模并行处理（MPP）技术（包括 Teradata 和 IBM Netezza）都可以处理高延迟的高度结构化数据。

# 常用技巧

## 开启垃圾箱(trash)功能

trash 功能默认是关闭的，开启后，被你删除的数据将会 mv 到操作用户目录的".Trash"文件夹，可以配置超过多长时间，系统自动删除过期数据。这样一来，当操作失误的时候，可以把数

据 mv 回来。

**开启垃圾箱**：修改文件core-site.xml ，添加下面配置，value 单位为分钟。

|  |
| --- |
| <property>  <name>fs.trash.interval</name>  <value>1440</value>  </property> |

开启垃圾箱后，如果希望文件直接被删除，可以在使用删除命令时添加“--skipTrash” 参数，如下：

|  |
| --- |
| hadoop fs –rm --skipTrash /xxxx |

# FAQ

## Win下Hadoop执行报错

1. 错误提示

java.io.IOException: Could not locate executable null\bin\winutils.exe in the Hadoop binaries.

由其中null可知，Hadoop路径未设置，在环境变量中设置HADOOP\_HOME，指向Hadoop的安装路径。

当使用IDEA时，Windows的环境变量设置无效…… 直接在IDEA运行设置中添加该环境变量，运行成功。

解决方法：配置环境变量，重启机器； 或在代码中添加System.setProperty("hadoop.home.dir", "...")进行设置。

再次执行，显示路径不再有null。

1. 错误提示

设置好环境变量后，再次执行，当访问hdfs时，报错：

Exception in thread "main" java.lang.NullPointerException  
at java.lang.ProcessBuilder.start(ProcessBuilder.java:1010)  
at org.apache.hadoop.util.Shell.runCommand(Shell.java:445)  
at org.apache.hadoop.util.Shell.run(Shell.java:418)  
at org.apache.hadoop.util.Shell$ShellCommandExecutor.execute(Shell.java:650)  
at org.apache.hadoop.util.Shell.execCommand(Shell.java:739)  
at org.apache.hadoop.util.Shell.execCommand(Shell.java:722)  
at org.apache.hadoop.fs.RawLocalFileSystem.setPermission(RawLocalFileSystem.java:633)

执行进程命令时失败 —— 可查询源码，再此，直接将之前报错时缺失的文件加入到Hadoop的bin目录下。

下载winutils.exe，并放入$HADOOP\_HOME/bin目录。

再次运行，OK。

说明：winutils.exe未放入时，进程执行命令为空，故抛出空指针异常。

# 参考

<http://www.lifeba.org/arch/hadoop_debug.html>

<http://www.searchbi.com.cn/showcontent_77668.htm>

计算机大牛Andrew S.Tanenbaum在[《现代操作系统》](http://book.douban.com/subject/1390650/)

<http://www.searchdatabase.com.cn/showcontent_67034.htm>

<http://download.techtarget.com.cn/bi/guide/2013/apachehadoop.pdf>

Apache hadoop 的 mail list : <http://hadoop.apache.org/mailing_lists.html>

Cloudera CDH google group: <https://groups.google.com/a/cloudera.org/forum/#!forum/cdh-userP>>

O'Reilly Strata 是 IT 业界最知名也是规模最大的数据管理大会：<http://strataconf.com/>

Llama：<http://blog.javachen.com/2014/09/09/llama/>

Hadoop安装：<http://www.cnblogs.com/xia520pi/archive/2012/05/16/2503949.html>

MapReduce的配置：<http://blog.sina.com.cn/s/blog_546abd9f0101i8b8.html>

Hadoop插件：

1. <https://github.com/winghc/hadoop2x-eclipse-plugin>
2. <http://blog.csdn.net/zythy/article/details/17397153>