

# 大数据编程基础


- 张大为
- 辽宁师范大学计算机与信息技术学院·辽宁大连。116081
- 手机：13998465335，微信：aqdvxin，电邮：daweiz@lnnu.edu.cn

## 绪论

## 为什么？

Google 有很多创新性应用，如 Google 搜索引擎，Gmail，安卓，AppspotGoogle Maps，Google Earth，Google 学术，Google 翻译，Google+等，下一步 Google What? [fig:intro01]

+您 搜索 图片 地图 Play YouTube 新闻 Gmail 更多



搜索找到约 17,400,000 条结果（用时 0.15 秒）

所有结果

图片

地图

视频

新闻

购物

图书

博客

更多

网页

所有中文网页

简体中文网页

翻译的外文网页

更多搜索工具

Welcome to Apache™ Hadoop™!

[hadoop.apache.org/](#) - 网页快照 - 翻译此页

Official site of the Apache project to provide an open-source implementation of frameworks for reliable, scalable, distributed computing and data storage.

↳ [MapReduce](#) - [HDFS](#) - [Hadoop Common Releases](#) - [PoweredBy](#) - [Hadoop Wiki](#)

Hadoop快速入门

[hadoop.apache.org/common/docs/r0.19.2/.../quickstart.html](#) - 网页快照

这篇文档的目的是帮助你快速完成单机上的Hadoop安装与使用以便你对Hadoop分布式文件系统( HDFS )和Map-Reduce框架有所体会，比如在HDFS上运行示例 ...

↳ [目的](#) - [先决条件](#) - [下载](#) - [运行Hadoop集群的准备工作](#)

Hadoop 百度百科

[baike.baidu.com/view/908354.htm](#) - 网页快照

一个分布式系统基础架构，由Apache基金会开发。用户可以在不了解分布式底层细节的情况下，开发分布式程序。充分利用集群的威力高速运算和存储。Hadoop实现 ...

↳ [Hadoop名字的起源](#) - [hadoop起源](#) - [诸多优点](#) - [hadoop架构](#)

Hadoop - 分布式文件系统- 开源中国

[www.oschina.net](#)，[开源软件](#)，[软件分类](#)，[分布式应用/网路](#) - 网页快照

2008年9月14日 - Hadoop并不仅仅是一个用于存储的分布式文件系统，而是设计用来在由

{fig:intro01}

Hadoop 思想来源于 Google，是 Google 对所面临问题解决方案的一种再版。Google 面对的数据和计算难题包括：

1. 大量的网页怎么存储？
2. 搜索算法
3. Page-Rank 计算问题

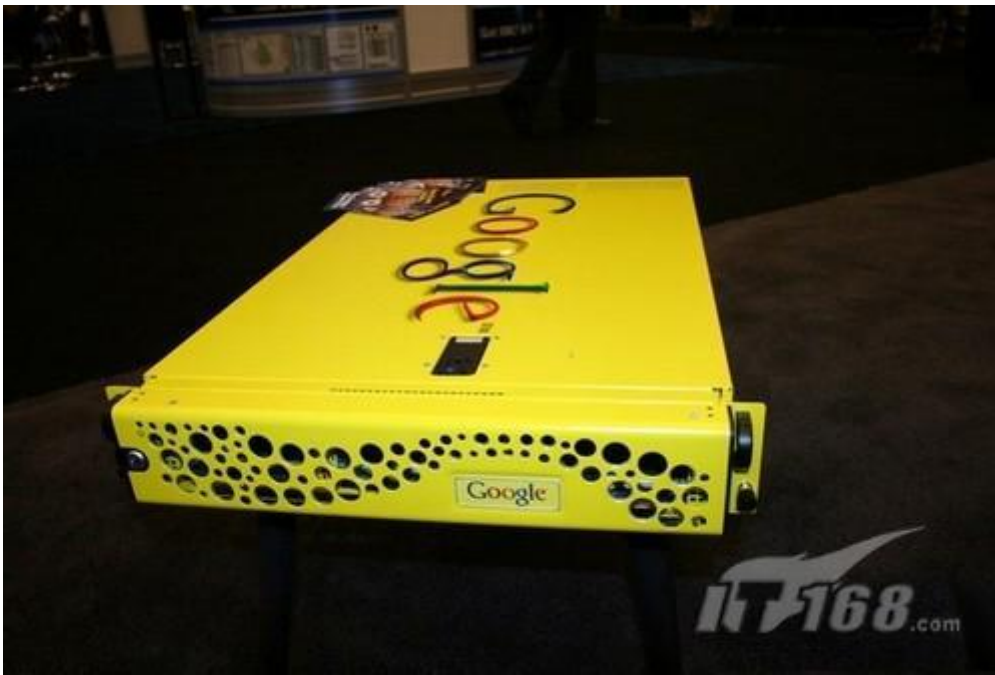
## 大量的网页如何存储？

Google 采用低成本之道存储搜索到的海量网页

1. 不使用超级计算机，不使用存储（淘宝的去 i，去 e，去 o 之路）。[@fig:intro02]
2. 大量使用普通的 pc 服务器（去掉机箱，外设，硬盘），提供有冗余的集群服务。[@fig:intro03]
3. 全世界多个数据中心，有些附带发电厂
4. 运营商向 Google 倒付费



{#fig:intro02}



{#fig:intro03}

## 建设集装箱数据中心

1. 位于 Mountain View, Calif 总部的数据中心
2. 总功率为 10000 千瓦，拥有 45 个集装箱，每个集装箱中有 1160 台服务器，该数据中心的能效比为 1.25（PUE 为 1 表示数据中心没有能源损耗，而根据 2006 年的统计，一般公司数据中心的能效比为 2.0 或更高。Google 的 1.16 已经低于美国能源部 2011 年的 1.2 的目标）。

[@fig:intro04], [@fig:intro05]



{#fig:intro04}

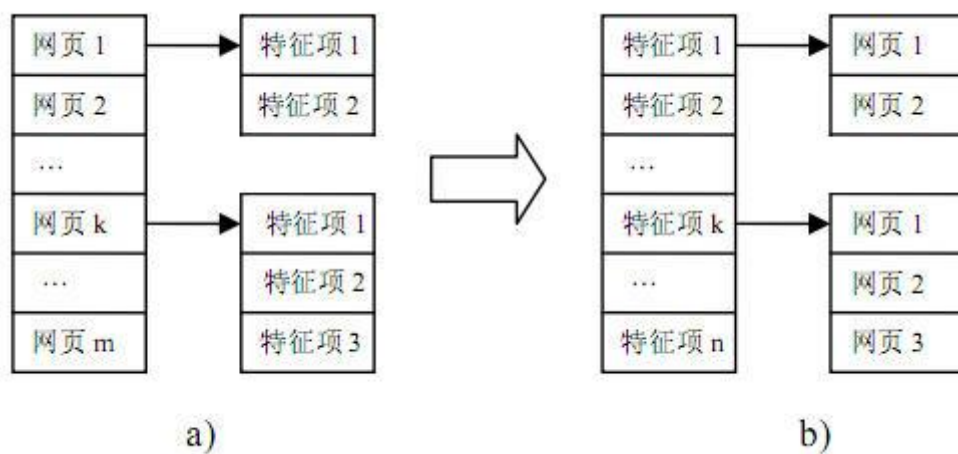


{#fig:intro05}

## 如何搜索网页?

采用倒排索引，通过关键字定位希望找到的网页。

1. 从每个网页中提取特征项，建立倒排索引。[@fig:intro06]
2. 倒排索引表。[@fig:intro07]
3. 利用倒排索引中的关键字找到所要的网页。[@fig:intro08]



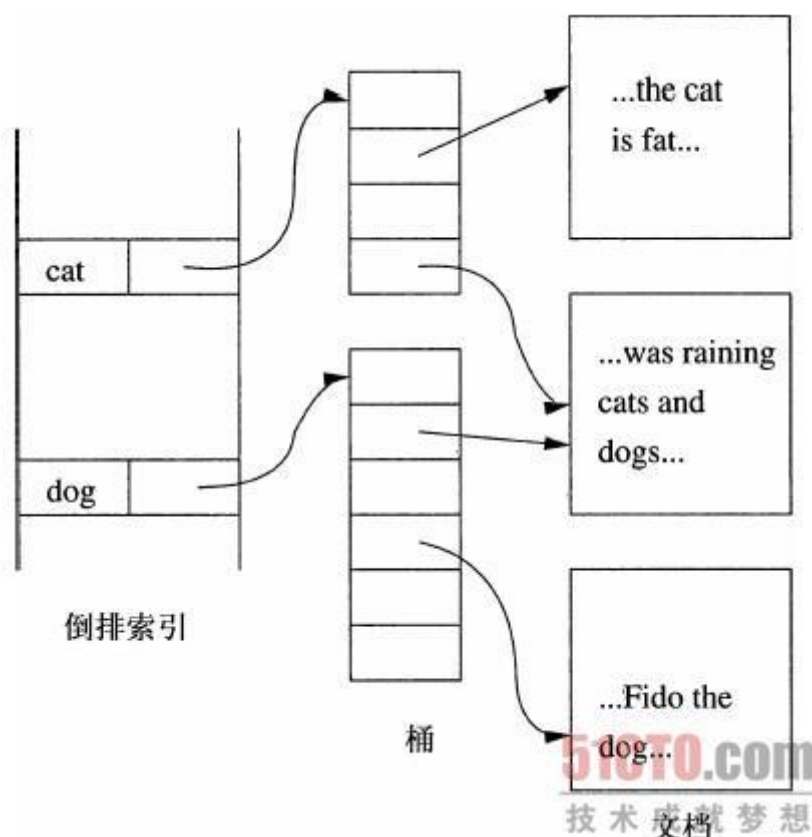
{#fig:intro06}



单词ID	单词	倒排列表 ( DocID:TF)
1	谷歌	(1;1),(2;1),(3;2),(4;1),(5;1)
2	地图	(1;1),(2;1),(3;1),(4;1),(5;1)
3	之父	(1;1),(2;1),(4;1),(5;1)
4	跳槽	(1;1),(4;1)
5	Facebook	(1;1),(2;1),(3;1),(4;1),(5;1)
6	加盟	(2;1),(3;1),(5;1)
7	创始人	(3;1)
8	拉斯	(3;1),(5;1)
9	离开	(3;1)
10	与	(4;1)
11	Wave	(4;1)
12	项目	(4;1)
13	取消	(4;1)
14	有关	(4;1)
15	社交	(5;1)
16	网站	(5;1)

织梦内容管理系统  
DEDECMS.COM

{#fig:intro07}



{#fig:intro08}

## Page Rank 计算每个网页的重要性

Page Rank 是 Google 最核心的算法，用于给每个网页价值评分，是 Google“在垃圾中找黄金”的关键算法，这个算法成就了今天的 Google。

1. PageRank 向量的定义。[@fig:intro09]
2. PageRank 计算方法。[@fig:intro10]
3. 网页链接矩阵。[@fig:intro11]

PageRank vector  $q$  is defined as  $q = Gq$

where  $G = \alpha S + (1 - \alpha) \frac{1}{n} U$

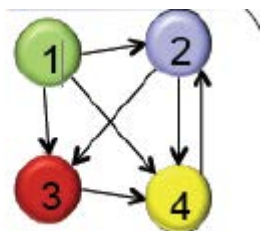
- ❑  $S$  is the destination-by-source stochastic matrix,
- ❑  $U$  is all one matrix.
- ❑  $n$  is the number of nodes
- ❑  $\alpha$  is the weight between 0 and 1 (e.g., 0.85)

{#fig:intro09}

Algorithm: Iterative powering for finding the first eigen-vector

$$q^{next} = Gq^{cur}$$

{#fig:intro10}



$$G = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1/3 & 0 & 0 & 1 \\ 1/3 & 1/2 & 0 & 0 \\ 1/3 & 1/2 & 1 & 0 \end{bmatrix}$$

{#fig:intro11}

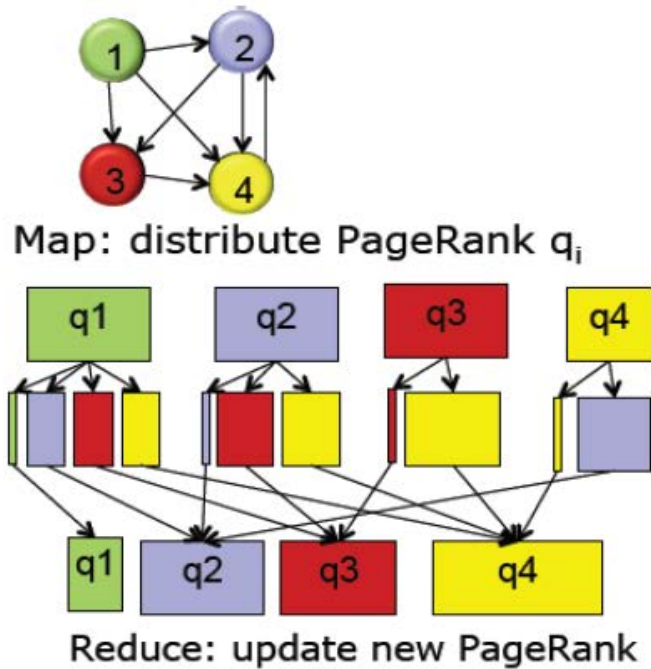
PageRank 向量，矩阵  $G$  实在太大了，单机无法计算完成，Google 工程师们发明了采用 Map-Reduce 进行分布式计算的方法：

1. PageRank Map 算法。[@fig:intro12]

2. Map 过程示例。[@fig:intro13]

3. Page Rank Reduce 算法。[@fig:intro14]

4. Reduce 过程示例。[@fig:intro15]



## PageRank Map()

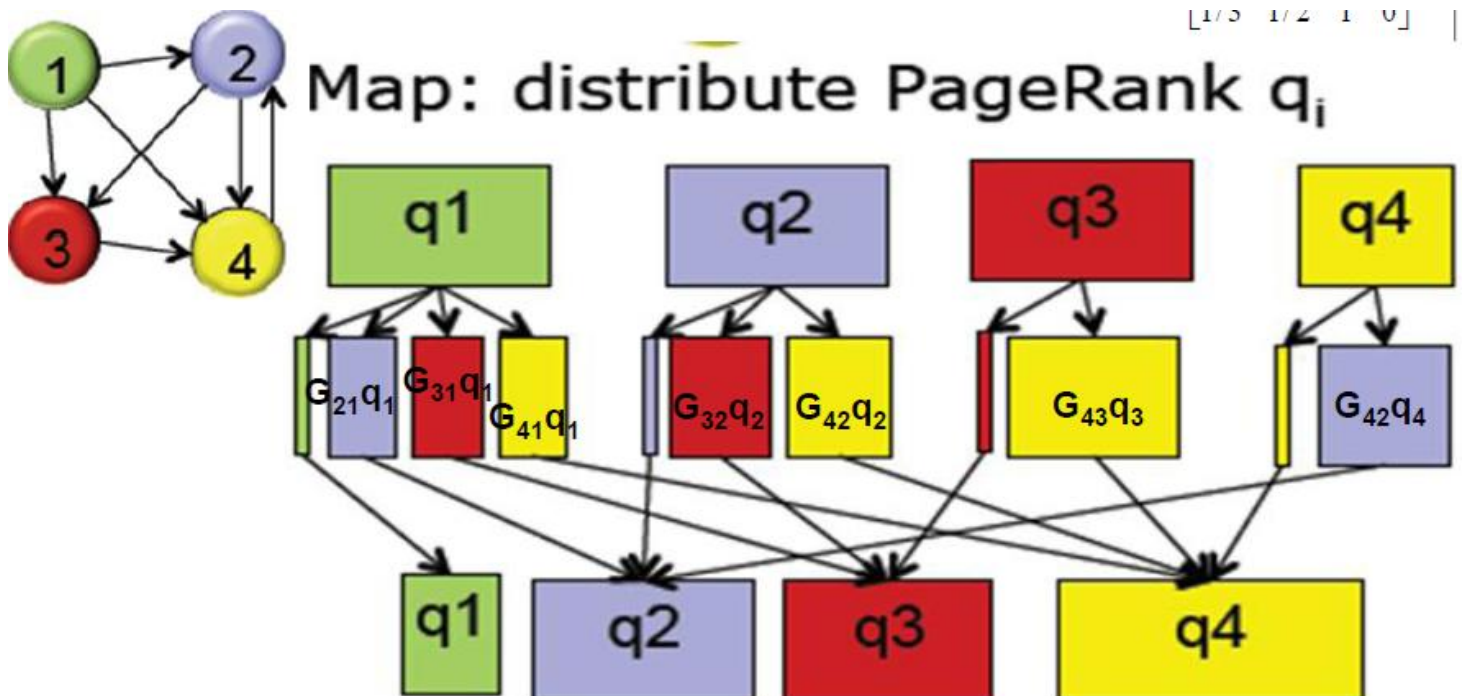
- Input:

- key = page  $x$ ,
- value =  $(q_x, \text{links}[y_1 \dots y_m])$

- Output:

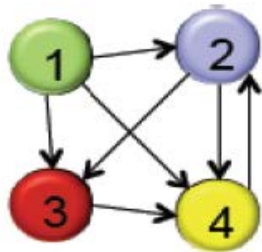
- key = page  $x$ ,
- value =  $\text{partial}_x$ 
  1.  $\text{Emit}(x, 0)$  //guarantee all pages will be emitted
  2. For each outgoing link  $y_i$ :  
 $\text{Emit}(y_i, G_{ix}q_x)$

{#fig:intro12}



{#fig:intro13}

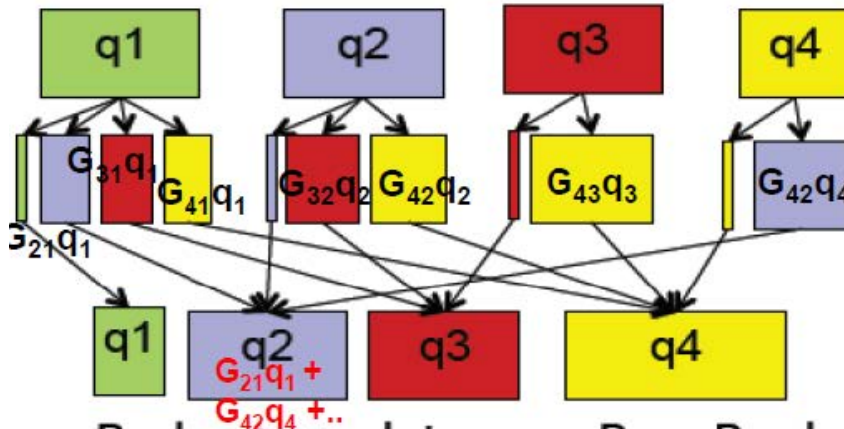




## PageRank Reduce()

- Input:
  - key = page  $x$ ,
  - value = the list of  $[\text{partial}_x]$
- Output:
  - key = page  $x$ ,
  - value = PageRank  $q_x$
  1.  $q_x = 0$
  2. For each partial value  $d$  in the list:
 
$$q_x += d$$
  3.  $q_x = \alpha q_x + (1 - \alpha)/n$
  4. Emit( $x, q_x$ )
$$q^{next} = Gq = \alpha Sq + (1 - \alpha) \frac{1}{n} Uq$$

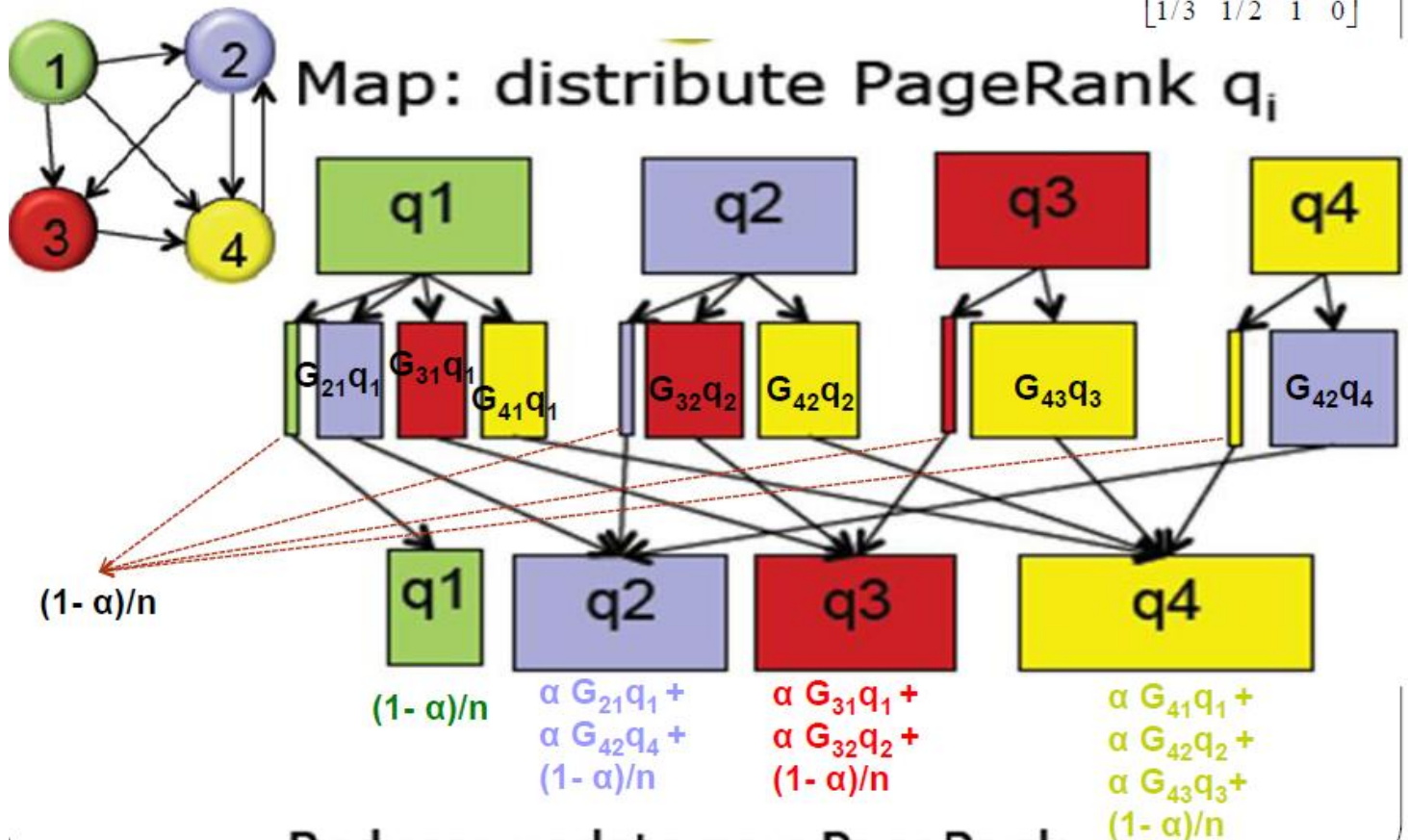
Map: distribute PageRank  $q_i$



Reduce: update new PageRank

{#fig:intro14}

[1/3 1/2 1 0]



{#fig:intro15}



Google 搜索引擎的建立和应用，带给我们的关键技术和思想包括：

1. GFS
2. Map-Reduce
3. Bigtable

为后来的大数据技术发展打下了坚实的基础。

## Hadoop 的源起

Doug Cutting 想要开发一款软件，类似于 Google 的搜索引擎，能广泛用于个人和企业。于是，用 Java 写了一个开源的搜索引擎，后被 Apache 基金会确认为其子项目 Lucene。

1. Doug Cutting 开创的开源软件，用 Java 书写代码，实现与 Google 类似的全文搜索功能，它提供了全文检索引擎的架构，包括完整的查询引擎和索引引擎。
2. 早期发布在个人网站和 SourceForge，2001 年年底成为 Apache 软件基金会 Jakarta 的一个子项目。
3. Lucene 的目的是为软件开发人员提供一个简单易用的工具包，以方便的在目标系统中实现全文检索的功能，或者是以此为基础建立起完整的全文检索引擎。
4. 对于大数量的场景，Lucene 面对与 Google 同样的困难，迫使 Doug Cutting 学习和模仿 Google 解决这些问题的办法。
5. 于是再写一个类似 Google 的微缩版：Nutch。

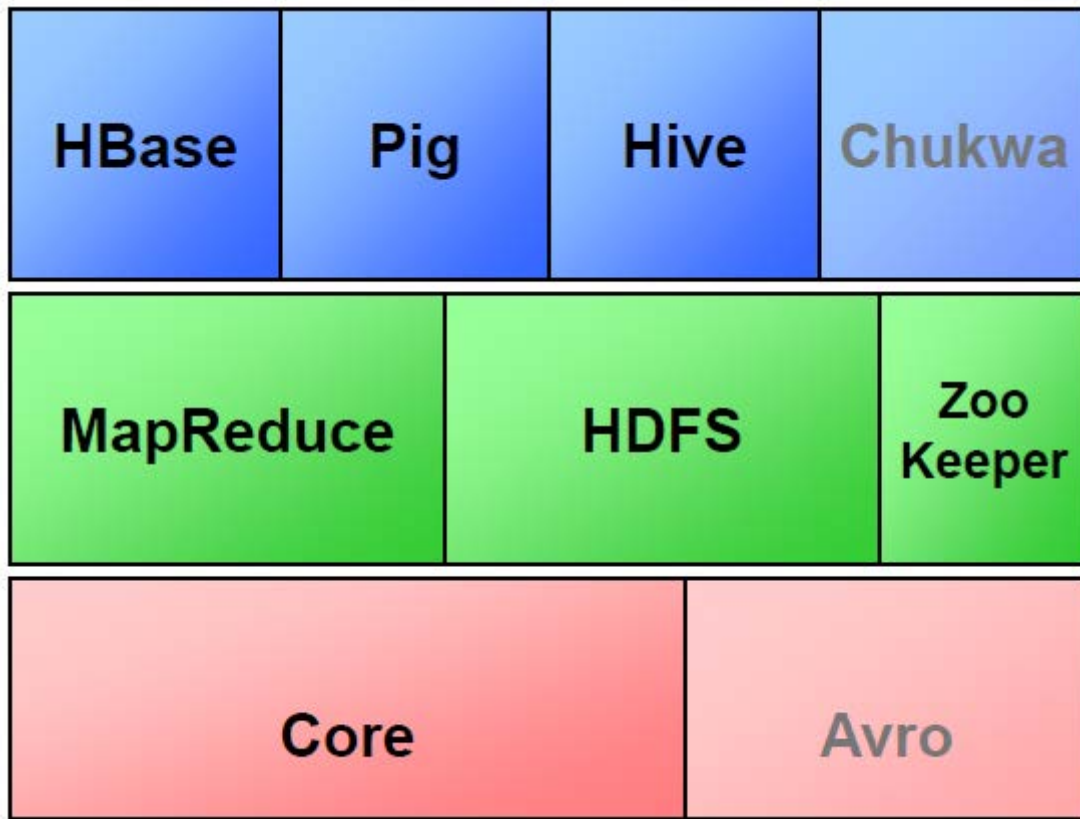
Nutch 中学习了 Google 技术，形成了自成系统的 Hadoop。

1. 2003-2004 年，Google 公开了部分 GFS 和 Mapreduce 思想的细节，以此为基础 Doug Cutting 等人用了 2 年的业余时间实现了 DFS 和 Mapreduce 机制，使 Nutch 性能飙升。
2. Yahoo 招安了 Doug Cutting 及其项目。
3. Hadoop 于 2005 年秋天作为 Lucene 的子项目 Nutch 的一部分正式引入 Apache 基金会。2006 年 3 月份，Map-Reduce 和 Nutch Distributed File System (NDFS) 分别被纳入称为 Hadoop 的项目中。
4. Hadoop 这个名字来源于 Doug Cutting 儿子的玩具大象。

目前 Hadoop 达到的高度包括：

1. 是实现云计算的事实标准的开源软件
2. 包含数十个具有强大生命力的子项目
3. 已经能在数千节点上运行，处理数据量和排序时间不断打破世界纪录

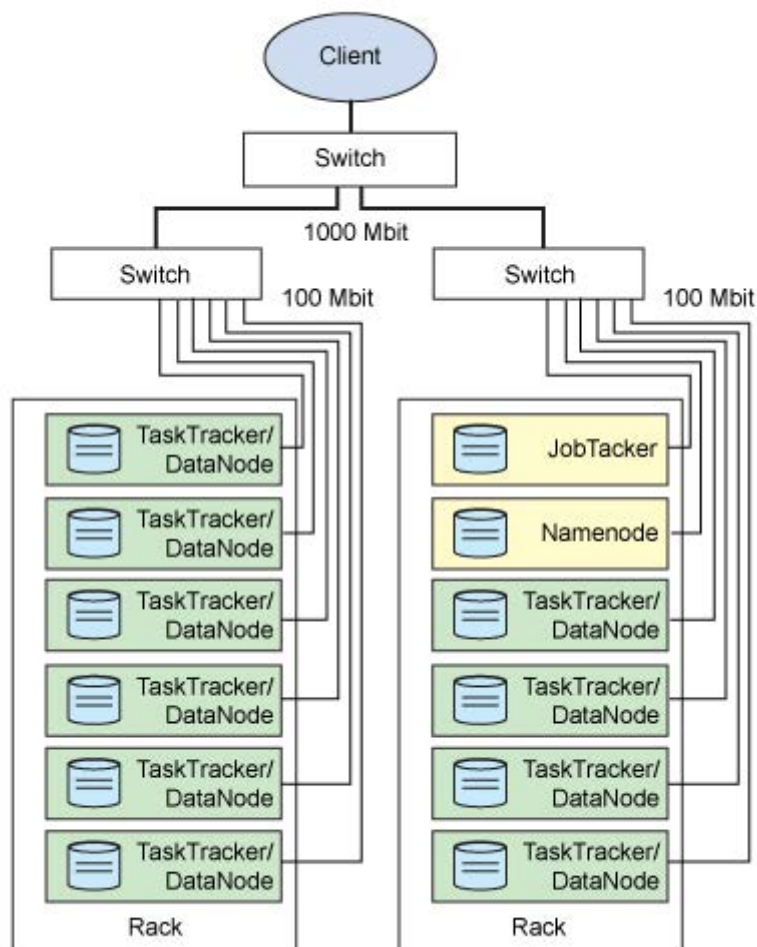
Hadoop 子项目家族见下图 [file:intro16]：



{#fig:intro16}

## Hadoop 的架构

Hadoop 的整体架构包括网络系统和节点系统两个部分。节点系统包括管理节点 Namenode 和 JobTacker（可部署在一台计算机），每个工作节点包括 DataNode 和 TaskTracker 两个工作分工。  
[@flg:intro17]



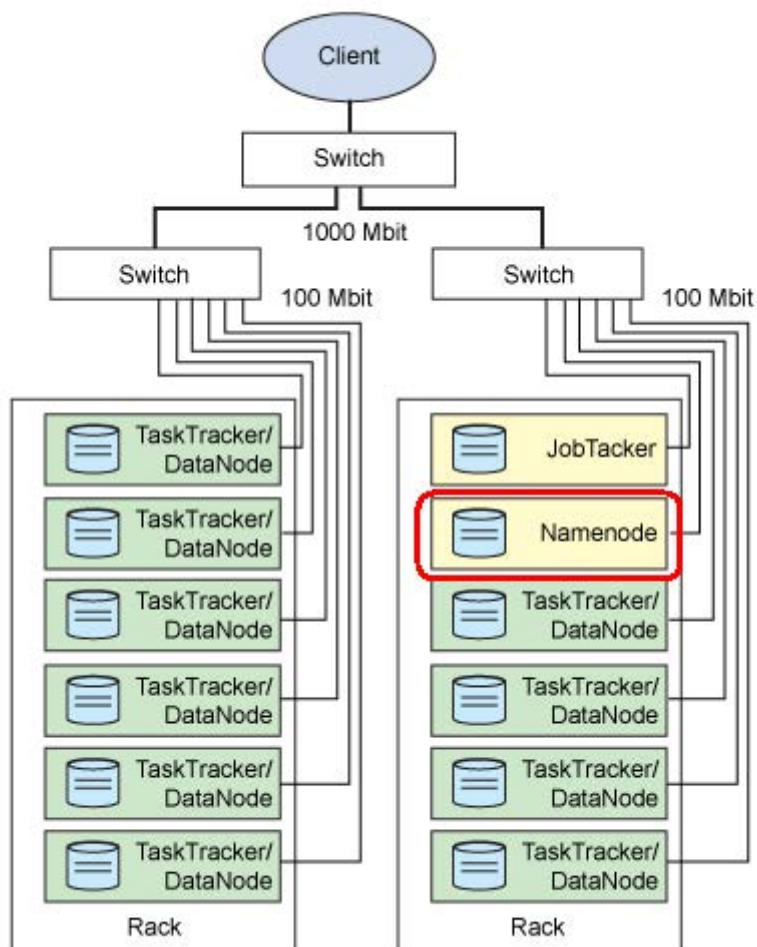
{#fig:intro17}

## Namenode

Namenode 的主要工作包括：

1. HDFS 的守护程序
2. 记录文件是如何分割成数据块的，以及这些数据块被存储到哪些节点上
3. 对内存和 I/O 进行集中管理
4. 是个单点，发生故障将使集群崩溃

Namenode 的部署位置见下图 [fig:intro18]：



{#fig:intro18}

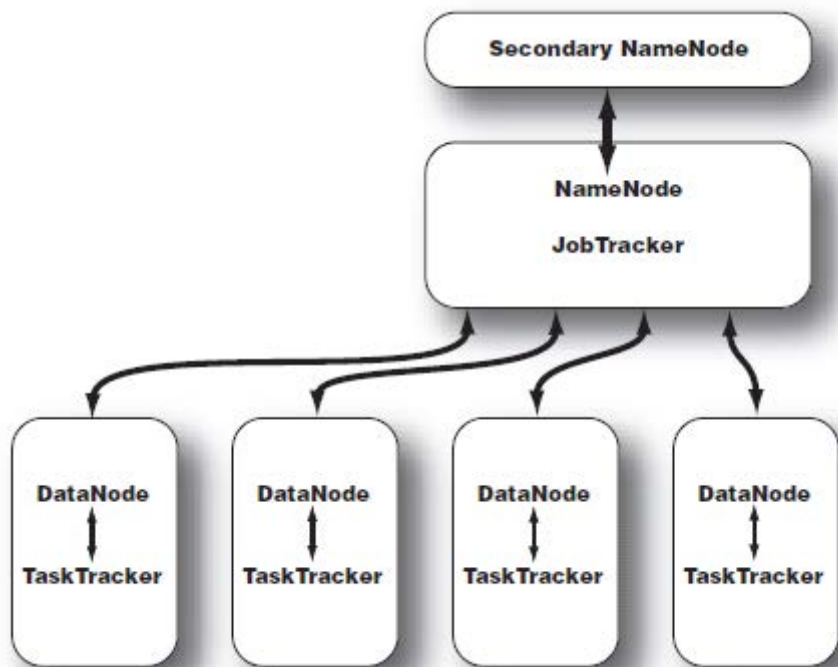
## Secondary Namenode

Secondary Namenode 的主要工作包括：

1. 监控 HDFS 状态的辅助后台程序
2. 每个集群都有一个
3. 与 NameNode 进行通讯，定期保存 HDFS 元数据快照
4. 当 NameNode 故障可以作为备用 NameNode 使用

Secondary Namenode 的部署架构见下图 [ @fig:intro19 ]:





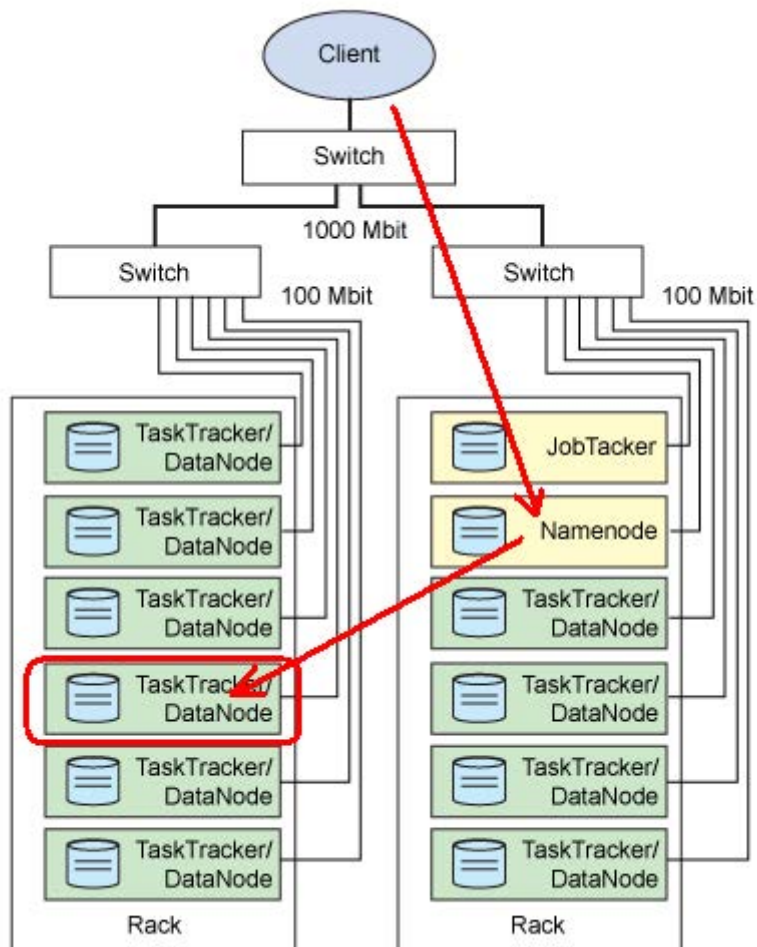
{#fig:intro19}

## DataNode

DataNode 的主要工作包括：

1. 每台从服务器都运行一个
2. 负责把 HDFS 数据块读写到本地文件系统

DataNode 的工作架构见下图 [ @fig:intro20 ]：



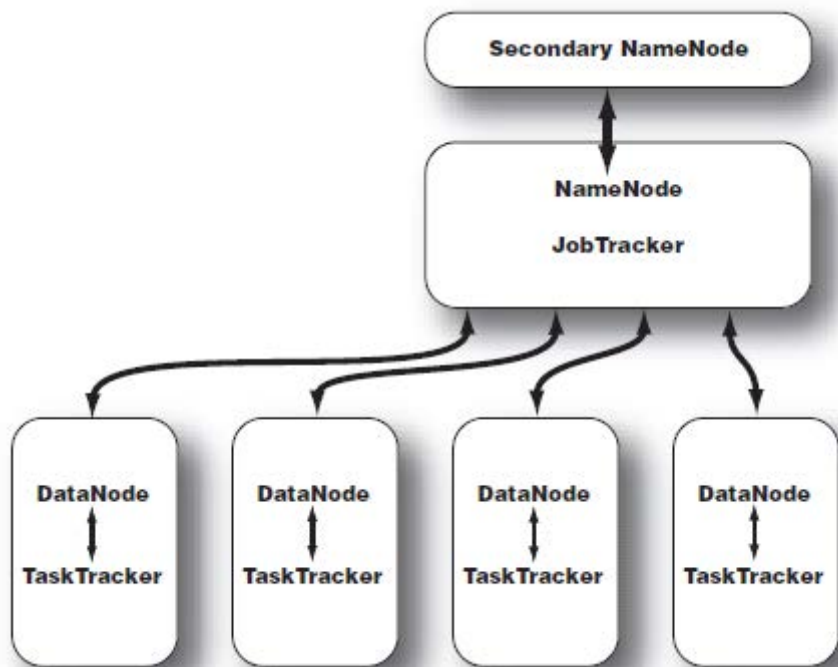
{#fig:intro20}

## JobTracker

JobTracker 的主要工作包括：

1. 用于处理作业（用户提交代码）的后台程序
2. 决定有哪些文件参与处理，然后切割 task 并分配节点
3. 监控 task，重启失败的 task（于不同的节点）
4. 每个集群只有唯一一个 JobTracker，位于 Master 节点

JobTracker 的部署架构见下图 [fig:intro21]:



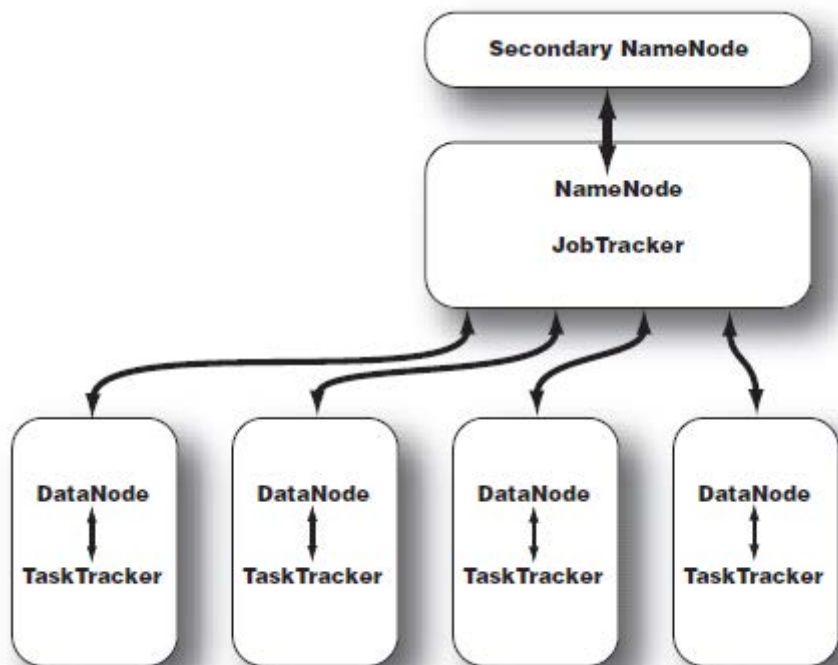
{#fig:intro21}

## TaskTracker

TaskTracker 的主要工作包括：

1. 位于 slave 节点上，与 datanode 结合（代码与数据一起的原则）
2. 管理各自节点上的 task（由 jobtracker 分配）
3. 每个节点只有一个 tasktracker，但一个 tasktracker 可以启动多个 JVM，用于并行执行 map 或 reduce 任务
4. 与 jobtracker 交互

TaskTracker 的部署架构见下图 [fig:intro22]：



{#fig:intro22}

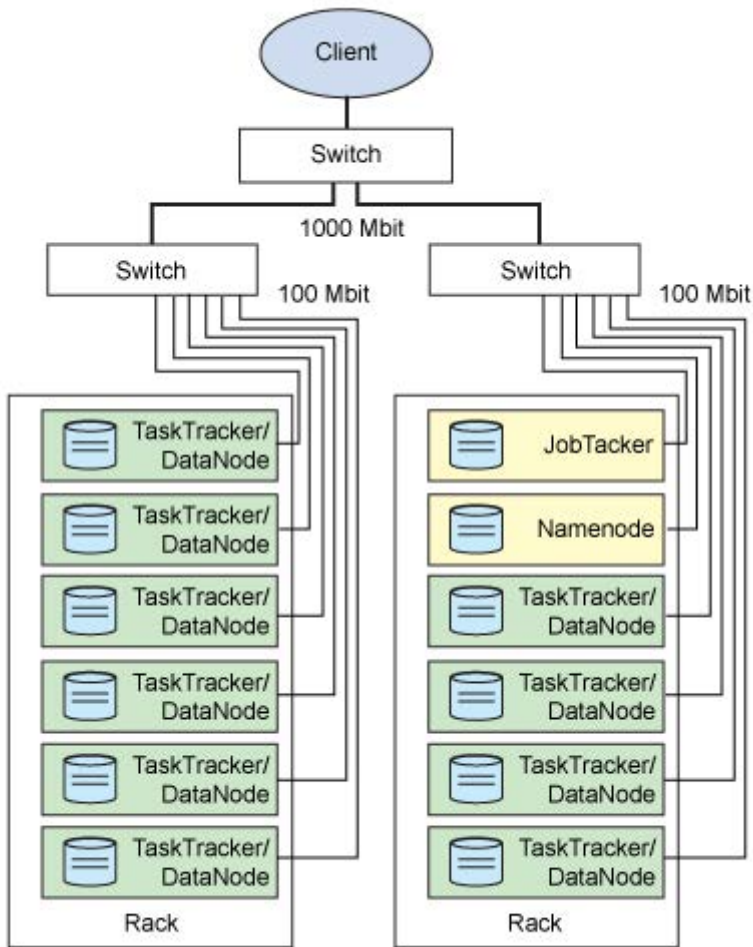
## Master 与 Slave

Master 和 Slave 的具体分工包括:

1. Master: Namenode、Secondary Namenode、Jobtracker。浏览器（用于观看管理界面），其它 Hadoop 工具
2. Slave: Tasktracker、Datanode
3. Master 不是唯一的

Master 和 Slave 的结构分工见下图 [ @fig:intro23 ]:

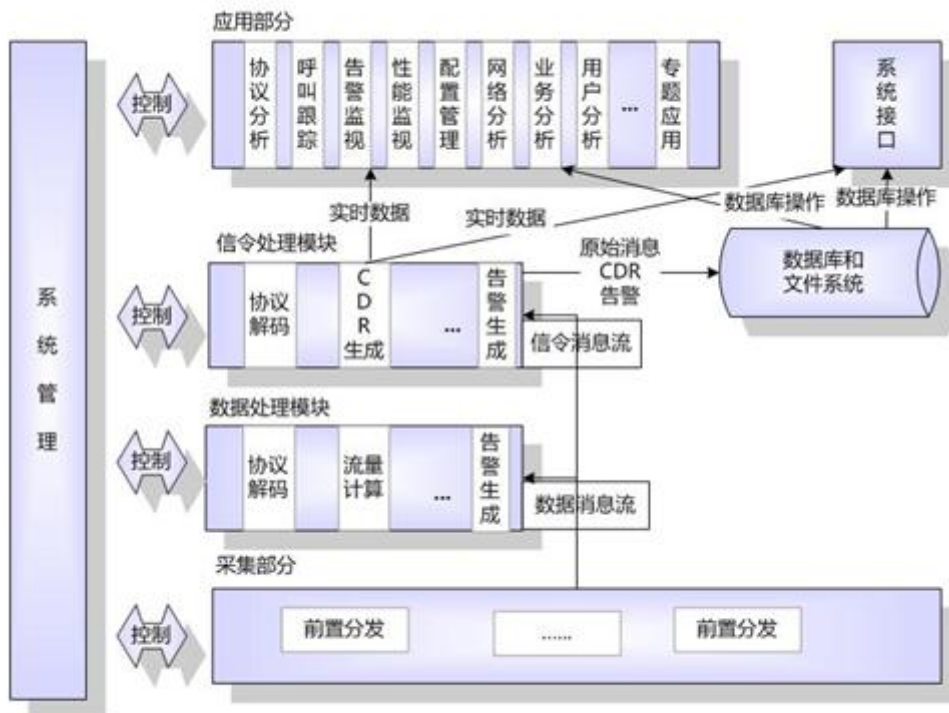




{#fig:intro23}

## Why hadoop?

下面是一个电信运营商信令分析与监测系统的系统结构图 [ @fig:intro24 ]:



{#fig:intro24}

系统架构实现存在的问题：

1. 原数据库服务器配置：HP 小型机，128G 内存，48 颗 CPU，2 节点 RAC，其中一个节点用于入库，另外一个节点用于查询
2. 存储：HP 虚拟化存储，>1000 个盘
3. 数据库架构采用 Oracle 双节点 RAC
4. 问题：1 入库瓶颈，2 查询瓶颈

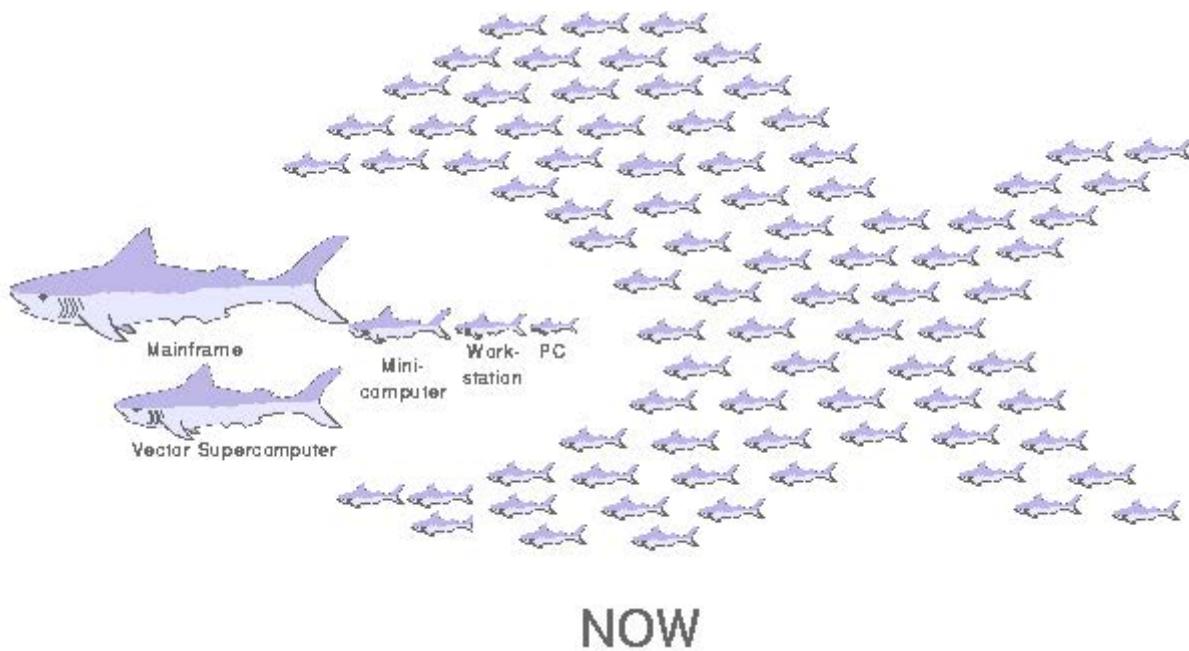
数据分析者面临的问题：

1. 数据日趋庞大，无论是入库和查询，都出现性能瓶颈
2. 用户的应用和分析结果呈整合趋势，对实时性和响应时间要求越来越高
3. 使用的模型越来越复杂，计算量指数级上升

数据分析者期待的解决方案：

1. 完美解决性能瓶颈，在可见未来不容易出现新瓶颈
2. 过去所拥有的技能可以平稳过渡。比如 SQL、R
3. 转移平台的成本有多高？平台软硬件成本，再开发成本，技能再培养成本，维护成本

Hadoop 的思想引入 [ @fig:intro25 ]：



{#fig:intro25}

## Why not Hadoop?

为什么不全面部署 Hadoop 呢？

1. Java?
2. 难以驾驭？
3. 数据集成困难？
4. Hadoop vs Oracle

Hadoop 体系下的分析手段包括：

1. 主流：Java 程序
2. 轻量级的脚本语言：Pig
3. SQL 技巧平稳过渡：Hive
4. NoSQL：HBase

## Windows 实验环境

大数据的存储和计算能力给我带来了许多机会，大数据技术值得学习和拥有。本课程假设你是一个 Java 语言的使用者，我们通过 Java 语言编写和运行用于大数据存储和分析的各种程序。具体包括 Map/Reduce 程序，HDFS 文件访问程序，Pig 程序，Hive 程序和 HBASE 程序等，并将数据分析的结果才 R 语言或 Web 的方式进行展示发表。

本课程需要 3~4 台 Ubuntu Server 作为 Hadoop 集群的节点，一台 Windows 计算机用于程序开发。你可以采用任何方法准备在同一网段内的 Ubuntu Server 和 Windows，如准备几台真是的物理机，或者在一台安装了 ESXi 的服务上，部署这些 Ubuntu Server 和 Windows。或者 Windows 是一台笔记本，其他 Ubuntu Server 部署在 ESXi 上。

本课程做两种假设

1. 你只有一台 PC 机或笔记本电脑操作系统恰巧是 Windows，硬件拥有 i5 以上的 CPU，8G 以上的内存，200G 以上的硬盘存储，你可以把那些 Ubuntu Server 部署到 VMWare Workstations 或 Virtual Box 上。本文选择 VMWare Workstations。
2. 你有一台 PC 机或笔记本电脑正在运行 Windows 10 1204 以上版本的操作系统，i5 以上的 CPU，4G 以上的内存，100G 以上的硬盘，可以考虑在 Windows 10 上安装 WSL 2，部署 Ubuntu Server 安装 Docker，然后通过 Docker 部署 Hadoop 集群。

我们先按第一种方案准备你的个人计算机。

## 准备工作目录

工作目录列表如 [@lst:windows\_work\_folder]



```
1 | D:\STUDENTS
2 |   └─20211202
3 |     └─bd1
4 |       └─idea
5 |         └─eclipse
6 |           └─plugins
7 |             └─hadoop-eclipse-plugin-2.6.0.jar
8 |               └─org.eclipse.wildwebdeveloper_0.5.18.202111090852
9 |                 └─readme
10 |                   └─readme_eclipse.html
11 |                 └─hadoop-3.3.1
12 |                   └─bin
13 |                     └─hadoop.dll
14 |                     └─winutils.exe
15 |                   └─share
16 |                 └─repos
17 |                   └─git
18 |                 └─tool
19 |                   └─eclipse-jee-2021-12-R-win32-x86_64.zip
20 |                   └─hadoop-3.3.1.tar.gz
21 |                   └─hadoop.dll
22 |                   └─jdk-11.0.13_windows-x64_bin.exe
23 |                   └─npp.8.1.9.Installer.exe
24 |                   └─winrar-x64-591scp.exe
25 |                   └─winutils.exe
26 |                 └─vms
27 |                 └─works
```

## Java

官网 [\[1\]](#) 下载 [\[2\]](#) Java 安装文件 jdk-11.0.13\_windows-x64\_bin.exe 到 D:/students/20211202/bd1/tool 文件夹下，鼠标左键双击 jdk-11.0.13\_windows-x64\_bin.exe 开始安装。依次出现如下安装向导对话框：

1. 你要允许此应用对你的设备进行更改吗？
  - 是
2. 欢迎使用 Java SE 开发工具包 11.0.13 的安装向导
  - 下一步
3. 选择 Java 安装到的文件夹
  - 保持默认 C:/Program Files/Java/jdk-11.0.13/
  - 下一步
4. Java(TM) SE Development Kit 11.0.13 (64-bit) 已成功安装
  - 关闭

Java 安装完成后需要配置正确的系统环境变量以后才可方便使用，具体配置过程如下：

1. Win+d 返回桌面：此电脑 > 鼠标右键 > 属性
2. 设置：关于 > 高级系统设置
3. 系统属性：高级 > 环境变量
4. 环境变量：系统变量 > 新建
5. 新建系统变量：
  - 变量名：JAVA\_HOME
  - 浏览目录
  - 变量值：C:/Program Files/Java/jdk-11.0.13
  - 确定
6. 环境变量：系统变量 > Path
  - 编辑
7. 编辑环境变量：新建
  - %JAVA\_HOME%
  - %JAVA\_HOME%/bin
  - 确定
8. 环境变量：确定
9. 系统属性：确定
10. 设置：关闭

方便 Java 使用的系统环境变量配置完成以后，需要打开新的命令终端验证 Java 的安装和配置情况。

1. Win+x > 运行
2. 运行：
  - 打开：cmd
  - 确定
3. 查看 path 环境变量命令：path
4. 查看全部环境变量命令：set
5. 检查 Java 安装配置是否正确命令：java -version

## Hadoop

官网 [\[3\]](#) 下载 [\[4\]](#) Hadoop 发行包 hadoop-3.3.1.tar.gz 到 D:/students/20211202/bd1/tool 文件夹下。

以 **管理员身份** 运行 WinRAR 解压软件，将 hadoop-3.3.1.tar.gz 解压到 D:/students/20211202/bd1/idea 下。

1. Win+d 返回桌面：此电脑 > 鼠标右键 > 属性
2. 设置：关于 > 高级系统设置
3. 系统属性：高级 > 环境变量

4. 环境变量：系统变量 > 新建
5. 新建系统变量：
  - 变量名：HADOOP\_HOME
  - 浏览目录
  - 变量值：D:/students/20211202/bd1/idea/hadoop-3.3.1
  - 确定
6. 环境变量：系统变量 > Path
  - 编辑
7. 编辑环境变量：新建
  - %HADOOP\_HOME%
  - %HADOOP\_HOME%/bin
  - 确定
8. 环境变量：确定
9. 系统属性：确定
10. 设置：关闭
11. 修改 D:/students/20211202/bd1/idea/hadoop-3.3.1/etc/hadoop/hadoop-env.cmd

```
set JAVA_HOME=C:/Progra~1/Java/jdk-11.0.13
```
12. Win+x > 运行
13. 运行：
  - 打开：cmd
  - 确定
14. 查看 path 环境变量命令：path
15. 查看全部环境变量命令：set
16. 检查 Hadoop 安装配置是否正确命令：

```
hadoop
hadoop -version
```

## Eclipse

官网 [\[5\]](#) 下载 [\[6\]](#) Eclipse IDE for Enterprise Java and Web Developers 发行包 eclipse-jee-2021-12-R-win32-x86\_64.zip 到 D:/students/20211202/bd1/tool 文件夹下。

1. 鼠标右键 eclipse-jee-2021-12-R-win32-x86\_64.zip 解压到当前文件夹下。
2. 移动解压好的文件夹 eclipse 到 D:/students/20211202/bd1/idea 下。
3. 进入文件夹 D:/students/20211202/bd1/idea/eclipse，鼠标左键双击 eclipse.exe，等待。..
4. Select a directory as workspace

- Workspace: D:/students/20211202/bd1/works
- Launch

5. 关闭 Welcome

6. 关闭 Eclipse

在 Eclipse 下管理 Hadoop 文件和进行 Map/Reduce 项目开发需要安装 Eclipse 插件 Hadoop-eclipse-plugin。可以从 [7] 下载该插件的源码后根据 Hadoop 的不同版本动态编译需要的 Hadoop-eclipse-plugin 插件版本。具体的编译过程见附录 [sec:C1]。

直接从 [8] 下载编译后的插件 hadoop-eclipse-plugin-2.6.0.jar, 从 [9] 下载 [10] winutils.exe 和 hadoop.dll 到文件夹 D:/students/20211202/bd1/tool 下。

1. 复制 winutils.exe 和 hadoop.dll 到 D:/students/20211202/bd1/idea/hadoop-3.3.1/bin 下
2. 复制 hadoop.dll 到 C:\Windows\System32
3. 复制 hadoop-eclipse-plugin-2.6.0.jar 到 D:/students/20211202/bd1/idea/eclipse/plugins
4. 双击 eclipse.exe 启动 Eclipse
5. Project Explorer 出现 DFS Locations
6. Window > Preferences
7. Preferences > Hadoop Map/Reduce
  - Hadoop installation directory: D:/students/20211202/bd1/idea/hadoop-3.3.1
  - Apply and Close
8. Window > Perspective > Open Perspective > Other...
9. Open Perspective > Map/Reduce
  - Open
10. Eclipse 右下方出现 Map/Reduce Locations 视图

## 过滤

1. Project Explorer > View Menu > Filters and Customization...
  - [ ] .\* resources
  - OK

## Git

1. Window > Preferences > Version Control (Team) > Git
  - Default repository folder: D:\students\202222022222\bd1\repos\git
  - [x] Use SSH agent for SSH connections
  - Default SSH agent: Pageant
  - Apply and Close



# Windows 下编写 Map/Reduce 测试程序-统计最高气温

1. 修改 etc/hadoop/hdfs-site.xml

```
<configuration>
  <property>
    <name>dfs.permissions</name>
    <value>false</value>
  </property>
</configuration>
```

2. 以管理员身份打开 Eclipse。

## 创建项目

1. File > New > Other...
2. Select a wizard > Map/Reduce Project
  - Next
3. MapReduce Project
  - Project name: temp.max.client
  - [x] Use default location
  - [x] Use default Hadoop
  - Next
4. Java Settings
  - Source > temp.max.client/src(new)
  - Default output folder: temp.max.client/bin
  - Finish

## 检查或增加 jar 包

1. Project Explorer > temp.max.client > JRE System Library
2. Project Explorer > temp.max.client > Reference Libraries

## 如果 Hadoop 的 jar 包 没有导入

1. Package Explorer > temp.max.client > 鼠标右键 > Build Path > Configure Build Path...
2. Properties for temp.max.client > Java Build Path > Libraries > Classpath
3. Add Library > User Library
  - Next
4. User Library
  - User Libraries...

## 5. Preferences > Java > Build Path > User Libraries

- New...
- New User Library
  - User library name: hadoop
  - OK
- Add External JARS...
  - D:/students/20211202/bd1/idea/hadoop-3.3.1/share/hadoop/\*\*/\*.\*jar
- Apply and Close

## 6. User Library

- User libraries: [x] hadoop
- Finish

# 编写测试代码

## 1. Project Explorer > temp.max.client > src > 鼠标右键 > New > Package

## 2. New Java Package

- Source folder: temp.max.client/src
- Name: temp.max
- Finish

## 3. Project Explorer > temp.max.client > src > temp.max > 鼠标右键 > New > Class

- Name: MaxTemperature
- [x] public static void main(String[] args)
- Finish

## 4. Project Explorer > temp.max.client > src > temp.max > 鼠标右键 > New > Class

- Name: MaxTemperatureMapper
- Finish

## 5. Project Explorer > temp.max.client > src > temp.max > 鼠标右键 > New > Class

- Name: MaxTemperatureReducer
- Finish

## 6. 在 MaxTemperature 中增加如下代码

```
static {
    try {
        System.load("D:/students/202222022222/bd1/idea/hadoop-3.2.2/bin/hadoop.dll");
    } catch (UnsatisfiedLinkError e) {
        System.err.println("Native code library failed to load.\n" + e);
        System.exit(1);
    }
}
```

## 7. MaxTemperature.java

```

1 // cc MaxTemperature Application to find the maximum temperature in the weather dataset
2 // vv MaxTemperature
3 package main.java;
4
5 import org.apache.hadoop.fs.Path;
6 import org.apache.hadoop.io.IntWritable;
7 import org.apache.hadoop.io.Text;
8 import org.apache.hadoop.mapreduce.Job;
9 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
10 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
11
12 public class MaxTemperature {
13
14     public static void main(String[] args) throws Exception {
15         if (args.length != 2) {
16             System.err.println("Usage: MaxTemperature <input path> <output path>");
17             System.exit(-1);
18         }
19
20         Job job = new Job();
21         job.setJarByClass(MaxTemperature.class);
22         job.setJobName("Max temperature");
23
24         FileInputFormat.addInputPath(job, new Path(args[0]));
25         FileOutputFormat.setOutputPath(job, new Path(args[1]));
26
27         job.setMapperClass(MaxTemperatureMapper.class);
28         job.setReducerClass(MaxTemperatureReducer.class);
29
30         job.setOutputKeyClass(Text.class);
31         job.setOutputValueClass(IntWritable.class);
32
33         System.exit(job.waitForCompletion(true) ? 0 : 1);
34     }
35 }
36 // ^^ MaxTemperature

```

## 8. MaxTemperatureMapper.java

```

1 // cc MaxTemperatureMapper Mapper for maximum temperature example
2 // vv MaxTemperatureMapper
3 package main.java;
4
5 import java.io.IOException;
6
7 import org.apache.hadoop.io.IntWritable;
8 import org.apache.hadoop.io.LongWritable;
9 import org.apache.hadoop.io.Text;
10 import org.apache.hadoop.mapreduce.Mapper;
11
12 public class MaxTemperatureMapper extends Mapper<LongWritable, Text, Text, IntWritable>
13
14     private static final int MISSING = 9999;
15
16     @Override
17     public void map(LongWritable key, Text value, Context context) throws IOException,
18         String line = value.toString();
19         String year = line.substring(15, 19);
20         int airTemperature;
21         if (line.charAt(87) == '+') { // parseInt doesn't like leading plus signs
22             airTemperature = Integer.parseInt(line.substring(88, 92));
23         } else {
24             airTemperature = Integer.parseInt(line.substring(87, 92));
25         }
26         String quality = line.substring(92, 93);
27         if (airTemperature != MISSING && quality.matches("[01459]")) {
28             context.write(new Text(year), new IntWritable(airTemperature));
29         }
30     }
31 }
32 // ^^ MaxTemperatureMapper

```

## 9. MaxTemperatureReducer.java

```

1  package temp.cluster.max;
2
3  import java.io.IOException;
4
5  import org.apache.hadoop.io.IntWritable;
6  import org.apache.hadoop.io.Text;
7  import org.apache.hadoop.mapreduce.Reducer;
8
9  public class MaxTemperatureReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
10
11      @Override
12      public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException {
13          int maxValue = Integer.MIN_VALUE;
14          for (IntWritable value : values) {
15              maxValue = Math.max(maxValue, value.get());
16          }
17          context.write(key, new IntWritable(maxValue));
18      }
19  }

```

## 执行测试程序

1. Project Explorer > temp.max.client > src > temp.max > MaxTemperature.java > 鼠标右键 > Run  
As > Run on Hadoop
2. Console
  - Usage: MaxTemperature <input path> <output path>
3. Project Explorer > temp.max.client > 鼠标右键 > New > Folder
  - Folder name: input
  - Finish
4. Project Explorer > temp.max.client > input > 鼠标右键 > New > File
  - File name: temp.txt
  - Finish
5. Project Explorer > temp.max.client > src > temp.max > MaxTemperature.java > 鼠标右键 > Run  
As > Run Configurations...
6. Run Configurations > Java Application > MaxTemperature
  - Name: MaxTemperature
  - Main
    - Project: temp.max.client
    - Main class: main.max.MaxTemperature
  - Arguments
    - input output
  - Run



7. MapTemperature.java > Run As > Run on Hadoop
8. temp.max.client > F5
9. Project Explorer > temp.max.client > output > part-r-00000

1949 111

1950 22

# Ubuntu

## 安装 VMware WorkStation

官网 [\[11\]](#) 下载 [\[12\]](#) VMware Workstation Pro 试用版 VMware-workstation-full-16.2.1-18811642.exe

鼠标双击 VMware-workstation-full-16.2.1-18811642.exe 安装开始安装。

1. 你要允许此应用对你的设备进行更改吗？
  - 是
2. 欢迎使用 VMware Workstation Pro 安装向导
  - 下一步
3. 最终用户许可协议
  - ☒ 我接受许可协议中的条款
  - 下一步
4. 自定义安装
  - 安装位置： C:/Program Files (x86)/VMware/VMware Workstation/
  - ☒ 增强型键盘驱动程序
  - ☒ 将 VMware Workstation 控制台工具添加到系统 PATH
  - 下一步
5. 用户体验设置
  - ☒ 启动时检查产品更新
  - ☐ 加入 VMware 客户体验提升计划
  - 下一步
6. 快捷方式
  - ☒ 桌面
  - ☒ 开始菜单程序文件夹
  - 下一步
7. 已准备好安装 VMware Workstation Pro
  - 安装

8. VMware Workstation Pro 安装向导已完成
  - 许可证
9. 输入许可证密钥
  - ZF3R0-FHED2-M80TY-8QYGC-NPKYF
  - 输入
10. VMware Workstation Pro 安装向导已完成
  - 完成
11. 您必须重新启动系统，对 VMware Workstation 进行的配置更改才能生效。
  - 是

## 支持嵌套虚拟化

此平台不支持虚拟化的 intel vt-x/ept 问题的解决。

1. 取消 Hyper-V 服务
2. 打开服务管理，禁止 HV 主机服务 (HvHost)
3. 以管理员身份启动 PowerShell
  - bcdedit /set hypervisorlaunchtype off
4. 重新启动电脑

Esxi 服务器虚拟化嵌套 (? 待实验)

1. esxi 启动 ssh server
2. ssh 链接 esxi
3. vi /etc/vmware/config
  - vhv.enable = "TRUE"
  - vhv.allow = "TRUE"
4. 重启 Esxi

## VMware Workstation 桥接网络配置

启动 VMware Workstation 应用程序

1. 编辑 > 虚拟网络编辑器
2. 更改设置
3. 选择“VMnet0”
4. 选择“桥接模式”
  - 已桥接至: Realtek PCIe GbE Family Controller
5. 确定

# 创建 Ubuntu 虚拟机

启动 VMware Workstation 应用程序

1. 选项卡 > 转到“主页”选项卡
2. 创建新的虚拟机
3. 欢迎使用新建虚拟机向导
  - (x) 自定义
  - 下一步
4. 选择虚拟机硬件兼容性
  - 硬件兼容性: Workstation 16.2.x
  - 下一步
5. 安装客户机操作系统
  - (x) 稍后安装操作系统
  - 下一步
6. 选择客户机操作系统
  - (x) Linux
  - 版本: Ubuntu 64 位
  - 下一步
7. 命名虚拟机
  - 虚拟机名称: node
  - 位置: D:/students/20211202/bd1/vms/node
  - 下一步
8. 处理器配置
  - 处理器数量: 1
  - 每个处理器的内核数量: 2
  - 下一步
9. 此虚拟机的内存
  - 5120MB
  - 下一步
10. 网络类型
  - (x) 使用桥接网络
  - 下一步
11. 选择 I/O 控制类型
  - (x) LSI Logic
  - 下一步
12. 选择磁盘类型
  - (x) SCSI

- 下一步
- 13. 选择磁盘
  - (x) 创建新虚拟磁盘
  - 下一步
- 14. 指定磁盘容量
  - 最大磁盘大小 (GB): 120
  - (x) 将虚拟磁盘存储为单个文件
  - 下一步
- 15. 指定磁盘文件
  - node.vmdk
  - 下一步
- 16. 已准备好创建虚拟机
  - 完成

## Ubuntu Server

官网 [\[13\]](#) 下载 [\[14\]](#) ubuntu-20.04.3-live-server-amd64.iso 到 D:/students/20211202/bd1/tool

启动 VMware Workstation 应用程序

1. 库 > 我的计算机 > node
2. node > 编辑虚拟机设置
3. 虚拟机设置 > 硬件 > CD/DVD (SATA)
  - [x] 启动时连接
  - (x) 使用 ISO 映像文件: D:/students/20211202/bd1/tool/ubuntu-20.04.3-live-server-amd64.iso
4. 虚拟机设置 > 硬件 > 网络适配器
  - [x] 启动时连接
  - (x) 桥接模式 (B): 直接连接到物理网络
5. 确定
6. node > 开启此虚拟机

安装 Ubuntu Server 的过程如下: 上下箭头移动选项, 空格选择, 回车确定

1. select your language
  - English
2. Installer update available
  - Update to the new installer
3. Keyboard configuration
  - Done

4. Network connections
  - Done
5. Configure proxy
  - Done
6. Configure Ubuntu archive mirror
  - Done
7. Guided storage configuration
  - (x) Use an entire disk
  - [ ] Set up this disk as an LVM group
  - Done
8. Storage configuration
  - Done
9. Confirm destructive action
  - Continue
10. Profile setup
  - Your name: Jack Ma
  - Your server's name: node
  - Pick a username: jack
  - Choose a password: 123456
  - Done
11. SSH Setup
  - [x] Install OpenSSH server
  - Done
12. Featured Server Snaps
  - Done
13. Installing system
14. Install complete
  - Reboot Now
15. Please remove the installation medium, then press ENTER:
16. 登录 Ubuntu
  - node login: jack
  - Password: 123456

首次登录后的基本操作：

#### 1. 安装网络工具

```
$ sudo apt install net-tools
```



## 2. 查看网卡信息

```
$ ifconfig
```

## 3. 修改 IP 地址

```
$ sudo vim /etc/netplan/00-installer-config.yaml
```

按 i 键，进入编辑模式

```
1 | network:
2 |   ethernets:
3 |     ens33:
4 |       addresses: [192.168.1.81/24]
5 |       gateway4: 192.168.1.254
6 |       dhcp4: true
7 |       optional: true
8 |   version: 2
```

按 Esc 键退出编辑模式

按: 键等待输入命令

输入 wq 命令存盘退出

## 4. 使修改后的 IP 地址生效

```
$ sudo netplan apply
$ ifconfig
```

## 5. 关闭 Ubuntu

```
$ sudo shutdown -h now
```

# Putty

官网 [\[15\]](#) 下载 [\[16\]](#)putty-64bit-0.76-installer.msi，双击安装：

### 1. Welcome to the PuTTY

- Next

### 2. Destination Folder

- C:/Program Files/PuTTY/
- Next

### 3. Product Features

- Add shortcut to PuTTY on the Desktop
  - Entire feature will be installed on local hard drive

- Install
- 4. 你要允许此应用对你的设备进行更改吗?
  - 是
- 5. Completed the PuTTY
  - [ ] View README file
  - Finish

开启 Ubuntu Server 虚拟机 node，然后运行 PuTTY

1. PuTTY Configuration
  - Host Name (or IP address): 192.168.1.81
  - Port: 22
  - Open
2. PuTTY Security Alert
  - Accept
3. 192.168.1.81 - PuTTY
  - login as: jack
  - [jack@192.168.1.83](#)'s password: 123456
4. jack@node:~

选择粘贴方法

1. 鼠标选中文字即为选择并复制到剪贴板了
2. 光标处鼠标右键即为粘贴

使用 putty 上传文件的方法

1. 进入到 PuTTY 的安装目录 C:\Program Files\PuTTY
2. 地址栏输入 cmd 后按回车，打开 Windows 的终端窗口
3. 通过 pscp.exe 上传文件

```
C:\Program Files\PuTTY>pscp readme.txt jack@192.168.1.81:/home/jack
jack@192.168.1.81's password:
readme.txt | 1 kB | 1.5 kB/s | ETA: 00:00:00 | 100%
```

免密码链接

1. 复制 SSH 服务器私钥 id\_rsa 的内容到 C:\Users\Dawei.ssh\81.rsa
2. 添加 SSH 服务器公钥 id\_rsa.pub 的内容到 authorized\_keys
3. 打开 C:\Program Files\PuTTY>puttygen.exe
  - load : C:\Users\Dawei.ssh\81.rsa
4. Save private key

- Are you sure you want to save this key without a passphrase to protect it? > yes
  - C:\Users\Dawei.ssh\81.ppk
5. 彻底删除 C:\Users\Dawei.ssh\81.rsa
  6. 启动 putty > Session
    - Host Name: 192.168.1.81
    - Port: 22
    - Connection type: SSH
    - Saved sessions: 81
  7. Connection > SSH > Auth
    - Private key file for authentication: C:\Users\Dawei.ssh\81.ppk
  8. Session > save
  9. open

## PowerShell

1. win+x > Windows PowerShell (管理员)
2. ssh 链接
 

```
ssh jack@192.168.1.81 -p 22
```
3. Are you sure you want to continue connecting (yes/no[fingerprint])?
  - yes
4. jack@192.168.1.81's password: 123456

退出 PowerShell 的 ssh 链接状态

1. 从 Windows 到 Linux

```
PS C:/Windows/system32> scp D:/students/20211202/bd1/works/temp/input/temp.txt jack@192.168.1.81:/home/jack/
jack@192.168.1.81's password:
temp.txt
100% 535 19
PS C:/Windows/system32>
```

2. 从 Linux 到 Windows

```
PS D:/tmp> scp jack@192.168.1.81:/home/jack/temp.txt D:/students/20211202/bd1/works/temp/input/
jack@192.168.1.81's password:
temp.txt
100% 535 263..
```

选择粘贴方法

1. 鼠标左键选中，鼠标右键复制

2. 光标处鼠标右键即为粘贴

# 版本控制系统

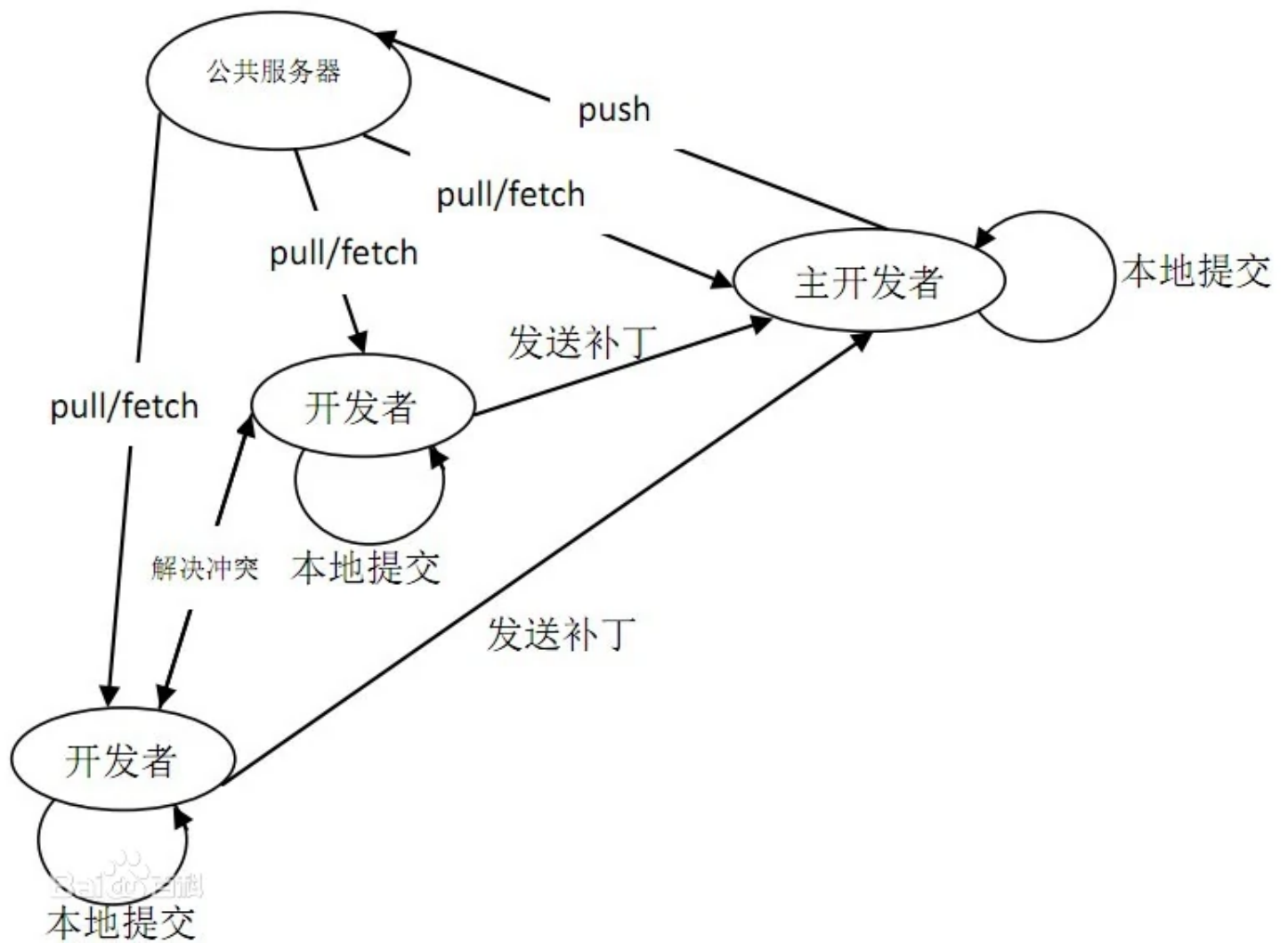
Git 与 Git 服务器通过 SSH key 建立免密码的信任链接。

通过 ssh-keygen 生成 SSH, 可以在 Git Bash 中执行, 或在 PowerShell 中执行:

1. win+r > cmd
2. ssh-keygen
  - Enter file in which to save the key (C:\Users\Dawei\.ssh/id\_rsa): Enter
  - Enter passphrase (empty for no passphrase): Enter
  - Enter same passphrase again: Enter
3. 查看 C:\用户\用户名\.ssh 是否存在
  - id\_rsa 是否存在
  - id\_rsa.pub 是否存在

## Git

## 状态图



1. git clone: 克隆远程仓库项目
2. git fetch: 拉取远程仓库版本到本地远程分支
3. git merge: 将本地远程分支 (remote) 与提交分支 (head) 合并
4. git pull: 拉取远程仓库版本到本地远程分支 (remote) 和本地分支 (head)
5. git checkout: 创建分支或切换分支
6. git rebase: 合并其他分支并形成新基线
7. git diff: 生成补丁
8. patch: 打补丁
9. git init: 初始化本地资源库
10. git remote -v: 查看远程资源库

## Git 安装

1. 官网 [\[17\]](#) 下载 [^git\_down] Git-2.35.1.2-64-bit.exe , 鼠标左键双击安装
2. 你要允许此应用对你的设备进行更改吗?
  - 。 是
3. Information

- Next
- 4. Select Destination Location
  - C:\Program Files\Git
  - Next
- 5. Select Components
  - [x] Windows Explorer integration
    - [x] Git Bash Here
    - [x] Git GUI Here
  - [x] Git LFS (Large File Support)
  - [x] Associate .git\* configuration files with the default text editor
  - [x] Associate .sh files to be run with Bash
  - [x] (NEW!) Add a Git Bash Profile to Windows Terminal
  - Next
- 6. Select Start Menu Folder
  - Default: Git
  - Next
- 7. Choosing the default editor used by Git
  - Use Notepad++ as Git's default editor
  - Next
- 8. Adjusting the name of the initial branch in new repositories
  - Let Git decide (currently: master)
  - Next
- 9. Adjusting your PATH environment
  - [x] Git from the command line and also from 3rd-party software
  - Next
- 10. Choosing the SSH executable
  - (x) Use bundled OpenSSH
  - Next
- 11. Choosing HTTPS transport backend
  - (x) Use the OpenSSL library
  - Next
- 12. Configuring the line ending conversions
  - (x) Checkout Windows-style, commit Unix-style line endings
  - Next
- 13. Configuring the terminal emulator to use with Git Bash
  - (x) Use Windows's default console window
  - Next
- 14. Choose the default behavior of 'git pull'



- (x) Default (fast-forward or merge)
  - Next
15. Choose a credential helper
- (x) Git Credential Manager
  - Next
16. Configuring extra options
- [x] Enable file system caching
  - Next
17. Configuring experimental options
- [x] Enable experimental support for pseudo consoles.
  - [x] Enable experimental built-in file system monitor
  - Install
18. Installing
19. Completing the Git Setup Wizard
- [x] Launch Git Bash
  - [ ] View Release Notes
  - Next
20. 在 Git Bash 终端输入 `git --version`

```
git version 2.35.1.windows.2
```

21. 在 Git Bash 终端输入 `exit` 退出

22. “win+r”组合件，出现“运行”页

- 打开: cmd
- 确定

23. 在终端窗口输入如下命令

- `git --version`

24. 全局配置

- `git config --global user.name "ASxx"`
- `git config --global user.email "123456789@qq.com"`

25. 查看全局配置

```
git config --list
```

## Git Server

### Gitee

1. 注册 gitee 账户并登录

2. 复制 C:\Users\Dawei.ssh\id\_rsa.pub 秘钥到剪贴板
3. 登录 gitee > 账户 > 设置 > SSH 公钥
  - 粘贴
  - 添加
4. 组织 > +
  - 名称: xxx-2022-1
  - 组织空间地址: xxx-2022-1
  - 简介: 张三的 2022 年第 1 学期
  - 创建
5. 访问组织: <https://gitee.com/xxx-2022-1>

## Gitlab

## Github

# temp.windows 项目的版本管理

## Gitee

1. 登录到 <https://gitee.com/>
2. 进入组织空间 xxx-2022-1
3. 新建仓库
  - 仓库名称: temp.windows
  - 路径: temp.windows
  - 仓库介绍: 最高气温统计 MapReduce 程序 Windows 运行
  - 初始化仓库
    - 语言: Java
    - 添加 .gitignore: Eclipse
    - 添加开源许可证: MIT
  - 设置模板
    - Readme 文件
    - Issue 模板文件
    - Pull Request 模板文件
  - 创建

## Git

1. 官员身份运行 Eclipse

2. 选择工作空间 D:\students\202222022222\bd1\weido

- Launch

3. temp.windows 鼠标右键 > Team > Share Project...

4. Configure Git Repository

- [x] Use or create repository in parent folder of project
- 点击 Project > temp.windows
- Create Repository
- Finish

5. temp.windwos 鼠标右键 > Team > Add to Index

6. temp.windwos 鼠标右键 > Team > Commit

- Commit Message: First
- Commit

7. 修改文件 D:\students\202222022222\bd1\works\temp.windows.git\config

```
[core]
  repositoryformatversion = 0
  filemode = false
  bare = false
  logallrefupdates = true
  symlinks = false
  ignorecase = true
[remote "origin"]
  url = git@gitee.com:ls-2022-1/temp.windows.git
  fetch = +refs/heads/*:refs/remotes/origin/*
[branch "master"]
  remote = origin
  merge = refs/heads/master
[user]
  name = Dawei Zhang
  email = dawewiz@lnnu.edu.cn
```

8. 鼠标右键 temp.pseudo Show in Local Terminal > Terminal

- git pull --allow-unrelated-histories

9. 修改 .gitignore 冲突

```
/bin/
```

10. temp.windows 鼠标右键 > Team > Add to Index

11. temp.windwos 鼠标右键 > Team > Commit

- Commit and push

12. Pushed to temp.windows refs/heads/master -origin

- Close

# Hadoop 本地部署

## 修改镜像源：

1. `sudo cp /etc/apt/sources.list /etc/apt/sources.list.back`
2. `sudo vim /etc/apt/sources.list`

# 更换为阿里源

```
deb http://mirrors.aliyun.com/ubuntu/ bionic main restricted universe multiverse
deb http://mirrors.aliyun.com/ubuntu/ bionic-security main restricted universe multiverse
deb http://mirrors.aliyun.com/ubuntu/ bionic-updates main restricted universe multiverse
deb http://mirrors.aliyun.com/ubuntu/ bionic-proposed main restricted universe multiverse
deb http://mirrors.aliyun.com/ubuntu/ bionic-backports main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-security main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-updates main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-proposed main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-backports main restricted universe multiverse
```

# 中科大源

```
deb https://mirrors.ustc.edu.cn/ubuntu/ bionic main restricted universe multiverse
deb-src https://mirrors.ustc.edu.cn/ubuntu/ bionic main restricted universe multiverse
deb https://mirrors.ustc.edu.cn/ubuntu/ bionic-updates main restricted universe multiverse
deb-src https://mirrors.ustc.edu.cn/ubuntu/ bionic-updates main restricted universe multiverse
deb https://mirrors.ustc.edu.cn/ubuntu/ bionic-backports main restricted universe multiverse
deb-src https://mirrors.ustc.edu.cn/ubuntu/ bionic-backports main restricted universe multiverse
deb https://mirrors.ustc.edu.cn/ubuntu/ bionic-security main restricted universe multiverse
deb-src https://mirrors.ustc.edu.cn/ubuntu/ bionic-security main restricted universe multiverse
deb https://mirrors.ustc.edu.cn/ubuntu/ bionic-proposed main restricted universe multiverse
deb-src https://mirrors.ustc.edu.cn/ubuntu/ bionic-proposed main restricted universe multiverse
```

# 清华源

```
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic main restricted universe multiverse
deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic main restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic-updates main restricted universe multiverse
deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic-updates main restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic-backports main restricted universe multiverse
deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic-backports main restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic-security main restricted universe multiverse
deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic-security main restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic-proposed main restricted universe multiverse
deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic-proposed main restricted universe multiverse
```

# 163 源

```
deb http://mirrors.163.com/ubuntu/ bionic main restricted universe multiverse
deb http://mirrors.163.com/ubuntu/ bionic-security main restricted universe multiverse
deb http://mirrors.163.com/ubuntu/ bionic-updates main restricted universe multiverse
deb http://mirrors.163.com/ubuntu/ bionic-proposed main restricted universe multiverse
deb http://mirrors.163.com/ubuntu/ bionic-backports main restricted universe multiverse
deb-src http://mirrors.163.com/ubuntu/ bionic main restricted universe multiverse
deb-src http://mirrors.163.com/ubuntu/ bionic-security main restricted universe multiverse
deb-src http://mirrors.163.com/ubuntu/ bionic-updates main restricted universe multiverse
deb-src http://mirrors.163.com/ubuntu/ bionic-proposed main restricted universe multiverse
deb-src http://mirrors.163.com/ubuntu/ bionic-backports main restricted universe multiverse
```

## 修改 DNS

1. `sudo vim /etc/systemd/resolved.conf`

```
[Resolve]
DNS=210.47.208.6 210.47.208.8 114.114.114.114
```

2. `systemctl restart systemd-resolved`

3. `systemctl enable systemd-resolved`

## 系统环境准备

1. 系统查新操作

```
$ sudo apt-get update
$ sudo apt-get upgrade
```

2. 安装或更新 ssh 客户端程序。SSH(Secure Shell) 安全外壳协议。

```
$ sudo apt-get install ssh
```

3. rsync 是 linux 系统下的数据镜像备份工具。Remote Sync 可以远程同步，支持本地复制，或者与其他 SSH、rsync 主机同步。

```
$ sudo apt-get install rsync
```

4. wget 是一个从网络上自动下载文件的自由工具，支持通过 HTTP、HTTPS、FTP 三个最常见的 TCP/IP 协议下载，并可以使用 HTTP 代理。"wget"这个名称来源于“World Wide Web”与“get”的结合。

所谓自动下载是指 wget 可以在用户退出系统之后在后台继续执行，直到下载任务完成。

```
$ sudo apt-get install wget
```

5. tree 生成目录树

```
$ sudo apt-get install tree
```

6. vim 编辑器

```
$ sudo apt-get install vim
```

### vim 的简单使用方法

1. vim 启动后一般会停留在“浏览”状态，此时只能看，不能改

2. 在浏览状态按 `i` 键进入编辑状态，屏幕下端出现 `---INSERT---` 提示信息



3. 退出编辑状态按 `Esc` 键
4. 在浏览状态按 `:` 键, 就是 `shift+;` 组合键, 进入命令状态
5. 在命令状态键入 `wq` 两个字母表示存盘并退出
6. 在命令状态键入 `q!` 表示强制退出

## 工作环境准备

我们将规划部署的内容命名为 `bd`, 其目录结构如下:

```
/usr/bd
├── dfs
│   ├── data
│   └── name
├── hadoop-3.2.2      注: 软件安装时创建 (版本可能不同)
├── jdk-11.0.10       注: 软件安装时创建 (版本可能不同)
├── logs
└── tmp
```

创建上述目录结构:

### 1. 创建目录

```
$ sudo mkdir /usr/bd
$ cd /usr/bd
$ sudo mkdir dfs logs tmp dfs/name dfs/data
```

### 2. 修改目录权限

```
$ sudo chown -R jack:jack dfs logs tmp
```

### 3. 查看目录权限

```
$ ls -ls
```

1		4	drwxr-xr-x	4	jack	jack	4096	Mar	19	08:26	dfs
2		4	drwxr-xr-x	2	jack	jack	4096	Mar	19	08:27	logs
3		4	drwxr-xr-x	2	jack	jack	4096	Mar	19	08:27	tmp

```
$ ls -ls dfs
```

1		4	drwxr-xr-x	2	jack	jack	4096	Mar	19	08:26	data
2		4	drwxr-xr-x	2	jack	jack	4096	Mar	19	08:26	name

# 准备 Java

在 Windows 环境下从官网 [\[18\]](#) 下载 [\[19\]](#) Linux 版本的 Java 部署文件到 D:/students/20211202/bd1/tool/jdk-11.0.13\_linux-x64\_bin.tar.gz。

打开 PowerShell 终端，上传 Java 部署文件到 Ubuntu Server。

```
scp D:\tools\java\jdk-11.0.13_linux-x64_bin.tar.gz jack@192.168.1.81:/home/jack
```

1. 登录到 Ubuntu Server
2. 查看家目录

```
cd ~  
ls
```

3. 解压 Java

```
sudo tar -zxvf jdk-11.0.13_linux-x64_bin.tar.gz
```

4. 移动 Java 部署文件夹到/usr/bd

```
sudo mv jdk-11.0.13 /usr/bd
```

5. 进入/usr/bd 文件夹

```
$ cd /usr/bd  
ls
```

6. 配置 Java

```
$ sudo vim /etc/profile
```

在文件末尾处增加如下两行

```
1 | export JAVA_HOME=/usr/bd/jdk-11.0.13  
2 | export PATH=$PATH:$JAVA_HOME/bin
```

7. 使/etc/profile 配置生效

```
$ source /etc/profile
```

8. 检查系统环境变量

```
$ env
```

## 9. 验证 Java 配置

```
$ java -version
```

# 准备 Hadoop

在 Windows 环境去 Apache [\[20\]](#) 的官网 [\[3:1\]](#) 下载 [\[4:1\]](#) Hadoop 部署文件到 D:/students/20211202/bd1/tool/hadoop-3.3.1.tar.gz。

打开 PowerShell

```
scp D:/students/20211202/bd1/tool/hadoop-3.3.1.tar.gz jack@192.168.1.81:/home/jack
```

### 1. 登录 Ubuntu Server

### 2. 进入家目录

```
cd ~  
ls
```

### 3. 解压 Hadoop

```
$ sudo tar -zxvf hadoop-3.3.1.tar.gz
```

### 4. 部署 Hadoop

```
sudo mv hadoop-3.2.2 /usr/bd  
ls /usr/bd
```

### 5. 配置 Path

```
$ sudo vim /etc/profile
```

```
1 | export JAVA_HOME=/usr/bd/jdk-11.0.11  
2 | export PATH=$PATH:$JAVA_HOME/bin  
3 |  
4 | export HADOOP_HOME=/usr/bd/hadoop-3.2.2  
5 | export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
```

### 6. 执行/etc/profile 使配置生效

```
$ source /etc/profile
```

### 7. 检查系统环境变量

```
$ env
hadoop
hadoop version
```

## 8. 配置 `hadoop-env.sh`

```
$ cd /usr/bd/hadoop-3.2.2/
$ sudo vim etc/hadoop/hadoop-env.sh
```

```
1 | # The java implementation to use. By default, this environment
2 | # variable is REQUIRED on ALL platforms except OS X!
3 | export JAVA_HOME=/usr/bd/jdk-11.0.13
4 | ... ..
5 | # Where (primarily) daemon log files are stored.
6 | # ${HADOOP_HOME}/logs by default.
7 | # Java property: hadoop.log.dir
8 | # export HADOOP_LOG_DIR=${HADOOP_HOME}/logs
9 | export HADOOP_LOG_DIR=/usr/bd/logs
```

## 9. 测试 Hadoop 命令

```
cd ~
$ hadoop
$ hadoop version
```

# Hadoop 本地模式测试

### 1. 进入家目录

```
$ cd ~
```

### 2. 准备测试数据

```
$ sudo mkdir input
$ sudo cp /usr/bd/hadoop-3.3.1/etc/hadoop/*.xml input
```

### 3. Hadoop 本地模式测试

```
$ hadoop jar /usr/bd/hadoop-3.2.2/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.2.jar
```

### 4. 查看测试结果

```
$ cat output/*
```

## 5. 查看目录结构

```
$ tree -L 2 /usr/bd
```

```
1  /usr/bd
2  ├── dfs
3      ├── data
4      └── name
5  ├── hadoop-3.3.1
6      ├── bin
7      ├── etc
8      ├── include
9      ├── lib
10     ├── libexec
11     ├── LICENSE-binary
12     ├── licenses-binary
13     ├── LICENSE.txt
14     ├── NOTICE-binary
15     ├── NOTICE.txt
16     ├── README.txt
17     ├── sbin
18     └── share
19 ├── jdk-11.0.13
20     ├── bin
21     ├── conf
22     ├── include
23     ├── jmods
24     ├── legal
25     ├── lib
26     ├── man
27     ├── README.html
28     └── release
29 ├── logs
30 └── tmp
31
32 22 directories, 7 files
```

# Temperature

1. 官员身份运行 Eclipse
2. 选择工作空间 D:\students\202222022222\bd1\weido
  - Launch
3. File > New > Other ...

#### 4. Select a wizard > Map/Reduce Project

- Next

#### 5. MapReduce Project

- Project name: temp.local
- Next

#### 6. Java Settings

- Finish

### 编写测试代码

#### 1. Project Explorer > temp.local > src > 鼠标右键 > New > Package

#### 2. New Java Package

- Source folder: temp.local/src
- Name: temp.local.max
- Finish

#### 3. Project Explorer > temp.local > src > temp.local.max > 鼠标右键 > New > Class

- Name: MaxTemperature
- ☒ public static void main(String[] args)
- Finish

#### 4. Project Explorer > temp.local > src > temp.local.max > 鼠标右键 > New > Class

- Name: MaxTemperatureMapper
- Finish

#### 5. Project Explorer > temp.local > src > temp.local.max > 鼠标右键 > New > Class

- Name: MaxTemperatureReducer
- Finish

### 设置 Java 的编译级别

#### 1. Project Explorer > temp > 鼠标右键 > Properties

#### 2. Properties for temp > Java Compiler

- ☒ Enable project specific settings
- Compiler compliance level: 11
- Apply and Close

#### 3. Compiler Settings Changed

- Yes

### 编译并打包部署文件

#### 1. Project Explorer > temp > 鼠标右键 > New > Folder

#### 2. New Folder

- Folder name: dist
- 3. Project Explorer > temp > Export...
- 4. Export > Java > Runnable JAR file
  - Next
- 5. Runnable JAR File Specification
  - Launch configuration: MaxTemperature - temp
  - Export destination: temp/dist/temp.jar
  - (x) Package required libraries into generated JAR
  - Finish

## PowerShell 上传 temp.jar

1. win+x > Windows PowerShell
2. 准备

```
d:
cd .\students\20211202\bd1\works\temp\dist\
scp .\temp.jar jack@192.168.1.81:/home/jack/
cd ../input
scp .\temp.txt jack@192.168.1.81:/home/jack/
```

## putty 登录 Ubuntu Server

1. 进入家目录

```
cd ~
ls
```

2. Ubuntu Server Hadoop 本地执行

```
hadoop jar temp.jar temp.txt temp
cat temp/*
```

# temp.local 项目的版本管理

## Gitee

1. 登录到 <https://gitee.com/>
2. 进入组织空间 xxx-2022-1
3. 新建仓库
  - 仓库名称: temp.local
  - 路径: temp.local



- 仓库介绍：最高气温统计 MapReduce 程序 Local 运行
- 初始化仓库
  - 语言：Java
  - 添加 .gitignore: Eclipse
  - 添加开源许可证：MIT
- 设置模板
  - Readme 文件
  - Issue 模板文件
  - Pull Request 模板文件
- 创建

## Git

1. 官员身份运行 Eclipse
2. 选择工作空间 D:\students\202222022222\bd1\weido
  - Launch
3. temp.local 鼠标右键 > Team > Share Project...
4. Configure Git Repository
  - [x] Use or create repository in parent folder of project
  - 点击 Project > temp.local
  - Create Repository
  - Finish
5. temp.local 鼠标右键 > Team > Add to Index
6. temp.local 鼠标右键 > Team > Commit
  - Commit Message: First
  - Commit
7. 修改文件 D:\students\202222022222\bd1\works\temp.local.git\config

```
[core]
  repositoryformatversion = 0
  filemode = false
  bare = false
  logallrefupdates = true
  symlinks = false
  ignorecase = true
[remote "origin"]
  url = git@gitee.com:ls-2022-1/temp.local.git
  fetch = +refs/heads/*:refs/remotes/origin/*
[branch "master"]
  remote = origin
  merge = refs/heads/master
[user]
  name = Dawei Zhang
  email = daweiz@lnnu.edu.cn
```

8. 鼠标右键 temp.local > Show in Local Terminal > Terminal

- git pull --allow-unrelated-histories

9. 修改 .gitignore 冲突

```
/bin/
```

10. temp.local 鼠标右键 > Team > Add to Index

11. temp.local 鼠标右键 > Team > Commit

- Commit and push

12. Pushed to temp.local refs/heads/master -origin

- Close

# Hadoop 伪分布部署

## ssh 免密码登录

1. 验证 ssh 免密码

```
$ ssh localhost
Are you sure you want to continue connecting (yes/no/[fingerprint])?yes
```

提示输入密码表明不能免密码。

```
$ exit
```

## 2. 生成免密码公钥

```
$ ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
```

## 3. 查看生成的密钥文件

```
$ ls ~/.ssh
```

```
1 | id_rsa id_rsa.pub known_hosts
```

## 4. 复制公钥文件到 authorized\_keys

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

```
1 | authorized_keys id_rsa id_rsa.pub known_hosts
```

## 5. 修改 authorized\_keys 的权限

```
$ sudo chmod 0600 ~/.ssh/authorized_keys  
$ ls -ls ~/.ssh
```

```
1 | total 16  
2 | 4 -rw----- 1 jack jack 563 Jan 4 02:10 authorized_keys  
3 | 4 -rw----- 1 jack jack 2590 Jan 4 02:08 id_rsa  
4 | 4 -rw-r--r-- 1 jack jack 563 Jan 4 02:08 id_rsa.pub  
5 | 4 -rw-r--r-- 1 jack jack 222 Jan 4 02:06 known_hosts
```

## 6. 重新验证 ssh 免密码登录

```
$ ssh localhost
```

没有出现输入密码提示

```
$ exit
```

# 伪分布模式最简配置

## 1. 修改 hadoop 下 etc/hadoop/core-site.xml 文件

```
$ sudo vim /usr/bd/hadoop-3.3.1/etc/hadoop/core-site.xml
```

```
1 <configuration>
2   <property>
3     <name>fs.defaultFS</name>
4     <value>hdfs://localhost:9000</value>
5   </property>
6   <property>
7     <name>hadoop.tmp.dir</name>
8     <value>/usr/bd/tmp</value>
9     <description>for hadoop temporary directories.</description>
10  </property>
11 </configuration>
```

## 2. 修改 hadoop 下 etc/hadoop/hdfs-site.xml 文件

```
$ sudo vim /usr/bd/hadoop-3.3.1/etc/hadoop/hdfs-site.xml
```

```
1 <configuration>
2   <property>
3     <name>dfs.replication</name>
4     <value>1</value>
5   </property>
6   <property>
7     <name>dfs.namenode.name.dir</name>
8     <value>file:/usr/bd/dfs/name</value>
9   </property>
10  <property>
11    <name>dfs.datanode.data.dir</name>
12    <value>file:/usr/bd/dfs/data</value>
13  </property>
14  <property>
15    <name>dfs.permissions</name>
16    <value>false</value>
17  </property>
18 </configuration>
```

## 3. 格式化文件系统

```
$ hdfs namenode -format
```

## 4. 启动 dfs

```
$ start-dfs.sh
```

## 5. 查看启动进程

```
$ jps
```

```
2690 Jps  
2306 DataNode  
2132 NameNode  
2554 SecondaryNameNode
```

## 6. 查看日志信息

```
$ ls /usr/bd/logs/
```

## 7. 查看 Namenode 网站

- <http://192.168.1.90:9870>

## 8. 在 dhfs 上创建文件夹

```
$ cd ~  
$ hdfs dfs -mkdir /user  
$ hdfs dfs -mkdir /user/jack  
$ hdfs dfs -mkdir /user/jack/input
```

## 9. 复制本地文件到 hdfs

```
$ hdfs dfs -put /usr/bd/hadoop-3.2.2/etc/hadoop/*.xml input  
$ hdfs dfs -ls input
```

## 10. 查看 Namenode 网站

- <http://192.168.1.90:9870>
- Utilities -> Browse the file system
- Browse Directory: /user/jack/input
- Go!

## 11. 离开安全模式

```
hadoop dfsadmin -safemode leave
```

## 12. 执行伪分布测试

```
$ hadoop jar /usr/bd/hadoop-3.2.2/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.2.jar
```

## 13. 查看 hdfs 文件信息

```
$ hdfs dfs -cat output/*
```

1	1	dfsadmin
2	1	dfs.replication
3	1	dfs.permissions
4	1	dfs.namenode.name.dir
5	1	dfs.datanode.data.dir

#### 14. 复制 hdfs 文件到本地查看

```
cd ~
sudo rm -rf output
$ hdfs dfs -get output output
$ cat output/*
```

1	1	dfsadmin
2	1	dfs.replication
3	1	dfs.permissions
4	1	dfs.namenode.name.dir
5	1	dfs.datanode.data.dir

#### 15. 查看 Namenode 网站

- <http://192.168.1.90:9870>
- Utilities -> Browse the file system
- Browse Directory: /user/jack/output
- Go!

#### 16. 查看 datanode 网站接口

- <http://192.168.1.81:9864>

#### 17. 停止 dfs

```
$ stop-dfs.sh
```

#### 18. 查看启动进程

```
$ jps
```

由于网络和 Hadoop 配置并不完善, Eclipse 下的 DFS 文件浏览仍然不能正常使用, MapReduce 程序仍然在本地的 Windows 上运行。

可以通过 SCP 上传打包的 jar 文件到 Ubuntu, 然后在 Ubuntu 上本地运行 MapReduce 程序。

## 伪分布模式基本配置

### 检查网络配置

## 1. 配置主机名

```
$ sudo vim /etc/hostname
```

```
1 | name
```

## 2. 查看网卡信息

```
$ ifconfig
```

```
1 | ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
2 |         inet 192.168.1.81 netmask 255.255.255.0 broadcast 192.168.1.255
```

## 3. 配置 IP 地址

```
$ sudo vim /etc/netplan/00-installer-config.yaml
```

```
1 | network:
2 |     ethernets:
3 |         ens33:
4 |             addresses: [192.168.1.81/24]
5 |             gateway4: 192.168.1.254
6 |             dhcp4: true
7 |             optional: true
8 |     version: 2
```

## 4. 执行网卡配置

```
$ sudo netplan apply
```

## 5. 查看网卡信息

```
$ ifconfig
```

```
1 | ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
2 |         inet 192.168.1.81 netmask 255.255.255.0 broadcast 192.168.1.255
```

## 6. 配置域名

```
$ sudo vim /etc/hosts
```



```

1 | 127.0.0.1 localhost
2 | #127.0.1.1 name
3 |
4 | 192.168.1.91 name
5 |
6 | # The following lines are desirable for IPv6 capable hosts
7 | ::1 ip6-localhost ip6-loopback
8 | fe00::0 ip6-localnet
9 | ff00::0 ip6-mcastprefix
10 | ff02::1 ip6-allnodes
11 | ff02::2 ip6-allrouters

```

## 7. 配置 DNS 服务器

```
$ sudo vim /etc/systemd/resolved.conf
```

```

1 | [Resolve]
2 | DNS=114.114.114.114 8.8.8.8

```

## 8. 测试 DNS

```

ping www.baidu.com
ctrl + c

```

# Hadoop 配置

## 1. 配置 core-site.xml

```
$ sudo vim /usr/bd/hadoop-3.3.1/etc/hadoop/core-site.xml
```

```

1 | <configuration>
2 |   <property>
3 |     <name>fs.defaultFS</name>
4 |     <value>hdfs://name:9000</value>
5 |   </property>
6 |   <property>
7 |     <name>hadoop.tmp.dir</name>
8 |     <value>/usr/bd/tmp</value>
9 |     <description>for hadoop temporary directories.</description>
10 |   </property>
11 | </configuration>

```

## 2. 配置 mapred-site.xml

```
$ sudo vim /usr/bd/hadoop-3.3.1/etc/hadoop/mapred-site.xml
```

```
1 | <configuration>
2 |     <property>
3 |         <name>mapreduce.jobtracker.address</name>
4 |         <value>name:9001</value>
5 |     </property>
6 | </configuration>
```

### 3. 配置 hdfs-site.xml

```
$ sudo vim /usr/bd/hadoop-3.3.1/etc/hadoop/hdfs-site.xml
```

```
1 | <configuration>
2 |     <property>
3 |         <name>dfs.replication</name>
4 |         <value>1</value>
5 |     </property>
6 |     <property>
7 |         <name>dfs.namenode.http-address</name>
8 |         <value>name:50070</value>
9 |     </property>
10 |    <property>
11 |        <name>dfs.namenode.name.dir</name>
12 |        <value>file:/usr/bd/dfs/name</value>
13 |    </property>
14 |    <property>
15 |        <name>dfs.datanode.data.dir</name>
16 |        <value>file:/usr/bd/dfs/data</value>
17 |    </property>
18 |    <property>
19 |        <name>dfs.permissions</name>
20 |        <value>false</value>
21 |    </property>
22 | </configuration>
```

### 4. 配置 workers

```
$ sudo vim /usr/bd/hadoop-3.3.1/etc/hadoop/workers
```

```
1 | name
```

## 首次启动 Hadoop 系统

### 1. 清理残留文件

```
sudo rm -rf /usr/bd/dfs/data/current /usr/bd/dfs/name/current
```

## 2. 格式化文件系统

```
$ hdfs namenode -format
```

## 3. 启动 hadoop

```
$ start-dfs.sh
```

## 4. 查看启动进程

```
$ jps
11424 Jps
11304 SecondaryNameNode
10875 NameNode
11052 DataNode
```

## 5. 上传数据

```
$ hdfs dfs -mkdir /user
$ hdfs dfs -mkdir /user/jack
$ hdfs dfs -mkdir /user/jack/input
$ hdfs dfs -put /usr/bd/hadoop-3.2.2/etc/hadoop/*.xml /user/jack/input
```

## 6. 查看上传数据

```
hdfs dfs -ls input/*
```

或者打开网站

- <http://192.168.1.81:50070>
- Utilities -> Browse the file system
- Browse Directory: /user/jack/input
- Go!

# Yarn 伪分布部署

Apache Hadoop YARN (Yet Another Resource Negotiator, 另一种资源协调者)

是一种新的 Hadoop 资源管理器，它是一个通用资源管理系统，可为上层应用提供统一的资源管理和调度，它的引入为集群在利用率、资源统一管理和数据共享等方面带来了巨大好处。

## 1. 停止 hdfs

```
stop-dfs.sh
```

## 2. 修改 mapred-site.xml 文件

```
$ sudo vim /usr/bd/hadoop-3.3.1/etc/hadoop/mapred-site.xml
```

```
1  <configuration>
2    <property>
3      <name>mapreduce.jobtracker.address</name>
4      <value>name:9001</value>
5    </property>
6    <property>
7      <name>mapreduce.framework.name</name>
8      <value>yarn</value>
9    </property>
10   <property>
11     <name>yarn.app.mapreduce.am.env</name>
12     <value>HADOOP_MAPRED_HOME=/usr/bd/hadoop-3.3.1</value>
13   </property>
14   <property>
15     <name>mapreduce.map.env</name>
16     <value>HADOOP_MAPRED_HOME=/usr/bd/hadoop-3.3.1</value>
17   </property>
18   <property>
19     <name>mapreduce.reduce.env</name>
20     <value>HADOOP_MAPRED_HOME=/usr/bd/hadoop-3.3.1</value>
21   </property>
22 </configuration>
```

### 3. 修改 hadoop 下 etc/hadoop/yarn-site.xml 文件

```
$ sudo vim /usr/bd/hadoop-3.2.2/etc/hadoop/yarn-site.xml
```

```
1  <configuration>
2    <!-- Site specific YARN configuration properties -->
3    <property>
4      <name>yarn.nodemanager.aux-services</name>
5      <value>mapreduce_shuffle</value>
6    </property>
7    <property>
8      <name>yarn.nodemanager.vmem-check-enabled</name>
9      <value>>false</value>
10   </property>
11 </configuration>
```

### 4. 启动 YARN

```
$ start-yarn.sh
```

### 5. 查看启动进程

```
$ jps
13442 Jps
12502 NodeManager
12333 ResourceManager
```

6. <http://192.168.1.81:8088>

7. 启动 DFS

```
$ start-dfs.sh
```

8. 查看启动进程

```
$ jps
13920 DataNode
13745 NameNode
12502 NodeManager
14295 Jps
12333 ResourceManager
14173 SecondaryNameNode
```

9. 删除输出目录

```
$ hdfs dfs -rm -r -f /user/jack/output
```

10. 查看 dfs 文件列表

```
$ hdfs dfs -ls /user/jack
```

11. 执行 MapReduce

```
$ hadoop jar /usr/bd/hadoop-3.2.2/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.2.jar
```

12. 查看 Yarn 的 Web 页面

- <http://192.168.1.90:8088>

13. 安全模式问题: Name node is in safe mode. 稍等一会就可以了。

14. 查看执行结果

```
$ hdfs dfs -cat /user/jack/output/*
```

1	1	dfsadmin
2	1	dfs.replication
3	1	dfs.permissions
4	1	dfs.namenode.name.dir
5	1	dfs.namenode.http
6	1	dfs.datanode.data.dir

## 15. 停止 YARN

```
$ stop-yarn.sh
```

## 16. 停止 dfs

```
$ stop-dfs.sh
```

# 配置作业历史服务器

## 1. 修改 mapred-site.xml 文件

```
$ sudo vim /usr/bd/hadoop-3.3.1/etc/hadoop/mapred-site.xml
```

```
1  <configuration>
2      <property>
3          <name>mapreduce.jobtracker.address</name>
4          <value>name:9001</value>
5      </property>
6      <property>
7          <name>yarn.app.mapreduce.am.env</name>
8          <value>HADOOP_MAPRED_HOME=/usr/bd/hadoop-3.3.1</value>
9      </property>
10     <property>
11         <name>mapreduce.map.env</name>
12         <value>HADOOP_MAPRED_HOME=/usr/bd/hadoop-3.3.1</value>
13     </property>
14     <property>
15         <name>mapreduce.reduce.env</name>
16         <value>HADOOP_MAPRED_HOME=/usr/bd/hadoop-3.3.1</value>
17     </property>
18     <property>
19         <name>mapreduce.framework.name</name>
20         <value>yarn</value>
21     </property>
22     <property>
23         <name>mapreduce.jobhistory.address</name>
24         <value>name:10020</value>
25     </property>
26 </configuration>
```

## 2. 启动 YARN

```
$ start-yarn.sh
```

## 3. 查看启动进程

```
$ jps
```

#### 4. 启动 DFS

```
$ start-dfs.sh
```

#### 5. 查看启动进程

```
$ jps
```

#### 6. 启动 historyserver

```
$ mr-jobhistory-daemon.sh start historyserver
```

#### 7. 查看启动进程

```
$ jps
17714 NameNode
18148 SecondaryNameNode
16972 ResourceManager
17916 DataNode
17148 NodeManager
18973 JobHistoryServer
19038 Jps
```

#### 8. 进入 history 管理页面

- <http://192.168.1.81:19888>

#### 9. 删除输出目录

```
$ hdfs dfs -rm -r -f /user/jack/output
```

#### 10. DFS 文件列表

```
$ hdfs dfs -ls /user/jack
```

#### 11. 执行 MapReduce

```
$ hadoop jar /usr/bd/hadoop-3.3.1/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.1.jar
```

#### 12. HDFS 命令查看测试结果

```
$ hdfs dfs -cat /user/jack/output/*
```

1	1	dfsadmin
2	1	dfs.replication
3	1	dfs.permissions
4	1	dfs.namenode.name.dir
5	1	dfs.namenode.http
6	1	dfs.datanode.data.dir

### 13. 停止 historyserver

```
$ mr-jobhistory-daemon.sh stop historyserver
```

### 14. 停止 YARN

```
$ stop-yarn.sh
```

### 15. 停止 dfs

```
$ stop-dfs.sh
```

## 制作启动脚本

#### 1. 查看系统 [start-all.sh](#) 脚本

```
$ vim /usr/bd/hadoop-3.3.1/sbin/start-all.sh
```

#### 2. 查看系统 [stop-all.sh](#)

```
$ sudo vim /usr/bd/hadoop-3.3.1/sbin/stop-all.sh
```

#### 3. 创建自己的脚本 [startall.sh](#)

```
$ sudo vim /usr/bd/hadoop-3.3.1/sbin/startall.sh
```



```
1  #!/usr/bin/env bash
2
3  # Start all hadoop daemons.  Run this on master node.
4
5  echo "This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh"
6
7  echo "Starting yarn..."
8  start-yarn.sh
9
10 echo "Starting dfs..."
11 start-dfs.sh
12
13 echo "Starting historyserver..."
14 mr-jobhistory-daemon.sh start historyserver
15
16 jps
```

#### 4. 修改权限

```
$ sudo chmod 755 /usr/bd/hadoop-3.3.1/sbin/startall.sh
```

#### 5. 修改所属

```
$ sudo chown 1001:1002 /usr/bd/hadoop-3.3.1/sbin/startall.sh
```

#### 6. 执行脚本

```
$ startall.sh
```

#### 7. 创建自己的脚本 `stopall.sh`

```
$ sudo vim /usr/bd/hadoop-3.2.2/sbin/stopall.sh
```

```
1  #!/usr/bin/env bash
2
3  # Stop all hadoop daemons.  Run this on master node.
4
5  echo "This script is Deprecated. Instead use stop-dfs.sh and stop-yarn.sh"
6
7  echo "Stoping historyserver..."
8  mr-jobhistory-daemon.sh stop historyserver
9
10 echo "Stoping dfs..."
11 stop-dfs.sh
12
13 echo "Stoping yarn..."
14 stop-yarn.sh
```

## 8. 修改权限

```
$ sudo chmod 755 /usr/bd/hadoop-3.2.2/sbin/stopall.sh
```

## 9. 修改所属权

```
$ sudo chown 1001:1002 /usr/bd/hadoop-3.2.2/sbin/stopall.sh
```

## 10. 执行脚本

```
$ stopall.sh
```

# Eclipse 浏览 Hadoop 文件系统 DFS

## 启动 Hadoop 文件系统

```
start-dfs.sh  
jps
```

## 修改 hosts

1. 修改 C:\Windows\System32\drivers\etc\hosts，添加域名
  - 210.47.218.170 namenode

## 启动 Eclipse

1. Window > Show View > Other...
2. Show View > MapReduce Tools > Map/Reduce Locations
  - Open
3. Map/Reduce Locations
  - 点击 大象 图标
4. Define Hadoop location > General
  - Location name: willow
  - Map/Reduce(V2) Master
    - Host: namenode
    - Port: 9001
  - DFS Master
    - [x] Use M/R Master host
    - Host: namenode
    - Port: 9000

- User name: jack

- Finish

## 5. Project Explorer

- DFS Locations > willow > (1) > user > jack

## 6. Project Explorer > DFS Locations > willow > (1) > user > jack > 鼠标右键

- Download from DFS...

- Create new directory...

- Upload files to DFS...

- Upload directory to DFS...

- Refresh

- Delete

# Eclipse 远程数据本地执行

## 启动 Hadoop

```
start-yarn.sh
```

```
start-dfs.sh
```

## 1. 官员身份运行 Eclipse

## 2. 选择工作空间 D:\students\202222022222\bd1\weido

- Launch

## 3. File > New > Other ...

## 4. Select a wizard > Map/Reduce Project

- Next

## 5. MapReduce Project

- Project name: temp.pseudo

- Next

## 6. Java Settings

- Finish

## 编写测试代码

## 1. Project Explorer > temp.windows > src > 鼠标右键 > New > Package

## 2. New Java Package

- Source folder: temp.pseudo/src

- Name: temp.pseudo.max

- Finish

## 3. Project Explorer > temp.windows > src > temp.windows.max > 鼠标右键 > New > Class

- Name: MaxTemperature
  - [x] public static void main(String[] args)
  - Finish
4. Project Explorer > temp.windows > src > temp.windows.max > 鼠标右键 > New > Class
- Name: MaxTemperatureMapper
  - Finish
5. Project Explorer > temp.windows > src > temp.windows.max > 鼠标右键 > New > Class
- Name: MaxTemperatureReducer
  - Finish
6. Define Hadoop location > Advanced parameters (需要先 Run As on Hadoop 一次)
- dfs.client.use.datanode.hostname=true
  - Finish
7. Project Explorer > DFS Locations > willow > (1) > user > jack > 鼠标右键
- Refresh
  - Create new directory...
    - Enter the name of the subfolder = temp
    - OK
  - Refresh
8. Project Explorer > DFS Locations > willow > (1) > user > jack > temp > 鼠标右键
- Upload files to DFS...
  - Refresh
9. Project Explorer > temp > 鼠标右键 > Run As > Run Configurations...
10. Run Configurations > Java Application > MaxTemperature > Arguments
- Program arguments = hdfs://namenode:9000/user/jack/temp  
hdfs://namenode:9000/user/jack/tempout
  - Close
11. src > hdfs-site.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <property>
    <name>hadoop.job.user</name>
    <value>jack</value>
  </property>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://namenode:9000</value>
  </property>
  <property>
    <name>mapred.job.tracker</name>
    <value>namenode:9001</value>
  </property>
  <property>
    <name>dfs.client.use.datanode.hostname</name>
    <value>true</value>
  </property>
  <property>
    <name>dfs.permissions</name>
    <value>false</value>
  </property>
</configuration>
```

12. src > log4j.properties

```

hadoop.root.logger=INFO,console
hadoop.log.dir=.
hadoop.log.file=hadoop.log

# Define the root logger to the system property "hadoop.root.logger".
log4j.rootLogger=${hadoop.root.logger}, EventCounter

# Logging Threshold
log4j.threshold=ALL

# Null Appender
log4j.appender.NullAppender=org.apache.log4j.varia.NullAppender

#
# Rolling File Appender - cap space usage at 5gb.
#
hadoop.log.maxfilesize=256MB
hadoop.log.maxbackupindex=20
log4j.appender.RFA=org.apache.log4j.RollingFileAppender
log4j.appender.RFA.File=${hadoop.log.dir}/${hadoop.log.file}

log4j.appender.RFA.MaxFileSize=${hadoop.log.maxfilesize}
log4j.appender.RFA.MaxBackupIndex=${hadoop.log.maxbackupindex}

log4j.appender.RFA.layout=org.apache.log4j.PatternLayout

# Pattern format: Date LogLevel LoggerName LogMessage
log4j.appender.RFA.layout.ConversionPattern=%d{ISO8601} %p %c: %m%n
# Debugging Pattern format
#log4j.appender.RFA.layout.ConversionPattern=%d{ISO8601} %-5p %c{2} (%F:%M(%L)) - %m%n

#
# Daily Rolling File Appender
#

log4j.appender.DRFA=org.apache.log4j.DailyRollingFileAppender
log4j.appender.DRFA.File=${hadoop.log.dir}/${hadoop.log.file}

# Rollover at midnight
log4j.appender.DRFA.DatePattern=.yyyy-MM-dd

log4j.appender.DRFA.layout=org.apache.log4j.PatternLayout

# Pattern format: Date LogLevel LoggerName LogMessage
log4j.appender.DRFA.layout.ConversionPattern=%d{ISO8601} %p %c: %m%n
# Debugging Pattern format
#log4j.appender.DRFA.layout.ConversionPattern=%d{ISO8601} %-5p %c{2} (%F:%M(%L)) - %m%n

#
# console

```

```

# Add "console" to rootlogger above if you want to use this
#

log4j.appender.console=org.apache.log4j.ConsoleAppender
log4j.appender.console.target=System.err
log4j.appender.console.layout=org.apache.log4j.PatternLayout
log4j.appender.console.layout.ConversionPattern=%d{ISO8601} %p %c{2}: %m%n

#
# TaskLog Appender
#
log4j.appender.TLA=org.apache.hadoop.mapred.TaskLogAppender

log4j.appender.TLA.layout=org.apache.log4j.PatternLayout
log4j.appender.TLA.layout.ConversionPattern=%d{ISO8601} %p %c: %m%n

#
# HDFS block state change log from block manager
#
# Uncomment the following to log normal block state change
# messages from BlockManager in NameNode.
#log4j.logger.BlockStateChange=DEBUG

#
#Security appender
#
hadoop.security.logger=INFO,NullAppender
hadoop.security.log.maxfilesize=256MB
hadoop.security.log.maxbackupindex=20
log4j.category.SecurityLogger=${hadoop.security.logger}
hadoop.security.log.file=SecurityAuth-${user.name}.audit
log4j.appender.RFAS=org.apache.log4j.RollingFileAppender
log4j.appender.RFAS.File=${hadoop.log.dir}/${hadoop.security.log.file}
log4j.appender.RFAS.layout=org.apache.log4j.PatternLayout
log4j.appender.RFAS.layout.ConversionPattern=%d{ISO8601} %p %c: %m%n
log4j.appender.RFAS.MaxFileSize=${hadoop.security.log.maxfilesize}
log4j.appender.RFAS.MaxBackupIndex=${hadoop.security.log.maxbackupindex}

#
# Daily Rolling Security appender
#
log4j.appender.DRFAS=org.apache.log4j.DailyRollingFileAppender
log4j.appender.DRFAS.File=${hadoop.log.dir}/${hadoop.security.log.file}
log4j.appender.DRFAS.layout=org.apache.log4j.PatternLayout
log4j.appender.DRFAS.layout.ConversionPattern=%d{ISO8601} %p %c: %m%n
log4j.appender.DRFAS.DatePattern=.yyyy-MM-dd

#
# hadoop configuration logging
#

```

```

# Uncomment the following line to turn off configuration deprecation warnings.
# log4j.logger.org.apache.hadoop.conf.Configuration.deprecation=WARN

#
# hdfs audit logging
#
hdfs.audit.logger=INFO,NullAppender
hdfs.audit.log.maxfilesize=256MB
hdfs.audit.log.maxbackupindex=20
log4j.logger.org.apache.hadoop.hdfs.server.namenode.FSNamesystem.audit=${hdfs.audit.logger}
log4j.additivity.org.apache.hadoop.hdfs.server.namenode.FSNamesystem.audit=false
log4j.appender.RFAAUDIT=org.apache.log4j.RollingFileAppender
log4j.appender.RFAAUDIT.File=${hadoop.log.dir}/hdfs-audit.log
log4j.appender.RFAAUDIT.layout=org.apache.log4j.PatternLayout
log4j.appender.RFAAUDIT.layout.ConversionPattern=%d{ISO8601} %p %c{2}: %m%n
log4j.appender.RFAAUDIT.MaxFileSize=${hdfs.audit.log.maxfilesize}
log4j.appender.RFAAUDIT.MaxBackupIndex=${hdfs.audit.log.maxbackupindex}

#
# NameNode metrics logging.
# The default is to retain two namenode-metrics.log files up to 64MB each.
#
namenode.metrics.logger=INFO,NullAppender
log4j.logger.NameNodeMetricsLog=${namenode.metrics.logger}
log4j.additivity.NameNodeMetricsLog=false
log4j.appender.NNMETRICSRFA=org.apache.log4j.RollingFileAppender
log4j.appender.NNMETRICSRFA.File=${hadoop.log.dir}/namenode-metrics.log
log4j.appender.NNMETRICSRFA.layout=org.apache.log4j.PatternLayout
log4j.appender.NNMETRICSRFA.layout.ConversionPattern=%d{ISO8601} %m%n
log4j.appender.NNMETRICSRFA.MaxBackupIndex=1
log4j.appender.NNMETRICSRFA.MaxFileSize=64MB

#
# DataNode metrics logging.
# The default is to retain two datanode-metrics.log files up to 64MB each.
#
datanode.metrics.logger=INFO,NullAppender
log4j.logger.DataNodeMetricsLog=${datanode.metrics.logger}
log4j.additivity.DataNodeMetricsLog=false
log4j.appender.DNMETRICSRFA=org.apache.log4j.RollingFileAppender
log4j.appender.DNMETRICSRFA.File=${hadoop.log.dir}/datanode-metrics.log
log4j.appender.DNMETRICSRFA.layout=org.apache.log4j.PatternLayout
log4j.appender.DNMETRICSRFA.layout.ConversionPattern=%d{ISO8601} %m%n
log4j.appender.DNMETRICSRFA.MaxBackupIndex=1
log4j.appender.DNMETRICSRFA.MaxFileSize=64MB

# Custom Logging levels

#log4j.logger.org.apache.hadoop.mapred.JobTracker=DEBUG
#log4j.logger.org.apache.hadoop.mapred.TaskTracker=DEBUG
#log4j.logger.org.apache.hadoop.hdfs.server.namenode.FSNamesystem.audit=DEBUG

```



```

# AWS SDK & S3A FileSystem
#log4j.logger.com.amazonaws=ERROR
log4j.logger.com.amazonaws.http.AmazonHttpClient=ERROR
#log4j.logger.org.apache.hadoop.fs.s3a.S3AFileSystem=WARN

#
# Event Counter Appender
# Sends counts of logging messages at different severity levels to Hadoop Metrics.
#
log4j.appender.EventCounter=org.apache.hadoop.log.metrics.EventCounter

#
# Job Summary Appender
#
# Use following logger to send summary to separate file defined by
# hadoop.mapreduce.jobsummary.log.file :
# hadoop.mapreduce.jobsummary.logger=INFO,JSA
#
hadoop.mapreduce.jobsummary.logger=${hadoop.root.logger}
hadoop.mapreduce.jobsummary.log.file=hadoop-mapreduce.jobsummary.log
hadoop.mapreduce.jobsummary.log.maxfilesize=256MB
hadoop.mapreduce.jobsummary.log.maxbackupindex=20
log4j.appender.JSA=org.apache.log4j.RollingFileAppender
log4j.appender.JSA.File=${hadoop.log.dir}/${hadoop.mapreduce.jobsummary.log.file}
log4j.appender.JSA.MaxFileSize=${hadoop.mapreduce.jobsummary.log.maxfilesize}
log4j.appender.JSA.MaxBackupIndex=${hadoop.mapreduce.jobsummary.log.maxbackupindex}
log4j.appender.JSA.layout=org.apache.log4j.PatternLayout
log4j.appender.JSA.layout.ConversionPattern=%d{ISO8601} %p %c{2}: %m%n
log4j.logger.org.apache.hadoop.mapred.JobInProgress$JobSummary=${hadoop.mapreduce.jobsummary.logger}
log4j.additivity.org.apache.hadoop.mapred.JobInProgress$JobSummary=false

#
# shuffle connection log from shuffleHandler
# Uncomment the following line to enable logging of shuffle connections
# log4j.logger.org.apache.hadoop.mapred.ShuffleHandler.audit=DEBUG

#
# Yarn ResourceManager Application Summary Log
#
# Set the ResourceManager summary log filename
yarn.server.resourcemanager.appsummary.log.file=rm-appsummary.log
# Set the ResourceManager summary log level and appender
yarn.server.resourcemanager.appsummary.logger=${hadoop.root.logger}
#yarn.server.resourcemanager.appsummary.logger=INFO,RMSUMMARY

# To enable AppSummaryLogging for the RM,
# set yarn.server.resourcemanager.appsummary.logger to
# <LEVEL>,RMSUMMARY in hadoop-env.sh

# Appender for ResourceManager Application Summary Log

```

```
# Requires the following properties to be set
#   - hadoop.log.dir (Hadoop Log directory)
#   - yarn.server.resourcemanager.appsummary.log.file (resource manager app summary log filename)
#   - yarn.server.resourcemanager.appsummary.logger (resource manager app summary log level and appender)

log4j.logger.org.apache.hadoop.yarn.server.resourcemanager.RMAppManager$ApplicationSummary=${yarn.server.resourcemanager.appsummary.log.level}
log4j.additivity.org.apache.hadoop.yarn.server.resourcemanager.RMAppManager$ApplicationSummary=false
log4j.appender.RMSUMMARY=org.apache.log4j.RollingFileAppender
log4j.appender.RMSUMMARY.File=${hadoop.log.dir}/${yarn.server.resourcemanager.appsummary.log.file}
log4j.appender.RMSUMMARY.MaxFileSize=256MB
log4j.appender.RMSUMMARY.MaxBackupIndex=20
log4j.appender.RMSUMMARY.layout=org.apache.log4j.PatternLayout
log4j.appender.RMSUMMARY.layout.ConversionPattern=%d{ISO8601} %p %c{2}: %m%n

# Appender for viewing information for errors and warnings
yarn.ewma.cleanupInterval=300
yarn.ewma.messageAgeLimitSeconds=86400
yarn.ewma.maxUniqueMessages=250
log4j.appender.EWMA=org.apache.hadoop.yarn.util.Log4jWarningErrorMetricsAppender
log4j.appender.EWMA.cleanupInterval=${yarn.ewma.cleanupInterval}
log4j.appender.EWMA.messageAgeLimitSeconds=${yarn.ewma.messageAgeLimitSeconds}
log4j.appender.EWMA.maxUniqueMessages=${yarn.ewma.maxUniqueMessages}

# Log levels of third-party libraries
log4j.logger.org.apache.commons.beanutils=WARN
```

13. Project Explorer > temp > src > [temp.mr](#) > MaxTemperature.java > 鼠标右键
  - Run As > Run on Hadoop

## Gitee

1. 登录到 <https://gitee.com/>
2. 进入组织空间 xxx-2022-1
3. 新建仓库
  - 仓库名称: temp.pseudo
  - 路径: temp.pseudo
  - 仓库介绍: 最高气温统计 MapReduce 程序 Pseudo 运行
  - 初始化仓库
    - 语言: Java
    - 添加 .gitignore: Eclipse
    - 添加开源许可证: MIT
  - 设置模板
    - Readme 文件
    - Issue 模板文件
    - Pull Request 模板文件

- 创建

## Git

1. 官员身份运行 Eclipse
2. 选择工作空间 D:\students\202222022222\bd1\weido
  - Launch
3. temp.pseudo 鼠标右键 > Team > Share Project...
4. Configure Git Repository
  - [x] Use or create repository in parent folder of project
  - 点击 Project > temp.pseudo
  - Create Repository
  - Finish
5. temp.pseudo 鼠标右键 > Team > Add to Index
6. temp.pseudo 鼠标右键 > Team > Commit
  - Commit Message: First
  - Commit
7. 修改文件 D:\students\202222022222\bd1\works\temp.windows.git\config

```
[core]
    repositoryformatversion = 0
    filemode = false
    bare = false
    logallrefupdates = true
    symlinks = false
    ignorecase = true
[remote "origin"]
    url = git@gitee.com:ls-2022-1/temp.pseudo.git
    fetch = +refs/heads/*:refs/remotes/origin/*
[branch "master"]
    remote = origin
    merge = refs/heads/master
[user]
    name = Dawei Zhang
    email = dawewiz@lnnu.edu.cn
```

8. 鼠标右键 temp.pseudo Show in Local Terminal > Terminal
  - git pull --allow-unrelated-histories
9. 修改 .gitignore 冲突

/bin/

10. temp.pseudo 鼠标右键 > Team > Add to Index
11. temp.pseudo 鼠标右键 > Team > Commit

- Commit and push

12. Pushed to temp.pseudo refs/heads/master -origin

- Close

# Hadoop 集群部署

复制或安装 Ubuntu Server 虚拟机节点 node1, node2, node3。

进入文件夹 D:/students/20211202/bd1/vms，复制文件夹 node 到当前文件夹，命名为 node1, node2, node3。

打开 VMware Workstation

1. 库 > 我的计算机 > node > 鼠标右键 > 重命名
  - name
2. 选项卡 > 主页 > 打开虚拟机
  - D:/students/20211202/bd1/vms/node1/node.vmx
3. 库 > 我的计算机 > node > 鼠标右键 > 重命名
  - node1
  - 开启此虚拟机
  - 我已复制该虚拟机
4. 选项卡 > 主页 > 打开虚拟机
  - D:/students/20211202/bd1/vms/node2/node.vmx
5. 库 > 我的计算机 > node > 鼠标右键 > 重命名
  - node2
  - 开启此虚拟机
  - 我已复制该虚拟机
6. 选项卡 > 主页 > 打开虚拟机
  - D:/students/20211202/bd1/vms/node3/node.vmx
7. 库 > 我的计算机 > node > 鼠标右键 > 重命名
  - node3
  - 开启此虚拟机
  - 我已复制该虚拟机

输入用户名 jack 和密码 123456 可以登录到虚拟机 node1,node2,node3。

## 网络配置

### 1. 对每个节点复查或配置 cloud.cfg

```
$ sudo vim /etc/cloud/cloud.cfg
```

```
1 | preserve_hostname: true
```

### 2. 分别修改每个节点的主机名 name,node1,node2,node3,...

```
$ sudo vim /etc/hostname
```

### 3. 查看每个节点的网卡信息

```
$ ifconfig
```

### 4. 配置每个节点的 IP 地址 192.168.1.81,91,92,93,...

```
$ sudo vim /etc/netplan/00-installer-config.yaml
```

```
1 | # This is the network config written by 'subiquity'
2 | network:
3 |   ethernets:
4 |     ens33:
5 |       addresses: [192.168.1.9x/24]
6 |       gateway4: 192.168.1.254
7 |       dhcp4: true
8 |       optional: true
9 |   version: 2
```

### 5. 应用每个节点的网卡配置

```
$ sudo netplan apply
```

### 6. 查看每个节点的网络配置是否生效

```
$ ifconfig
```

### 7. 为每个节点配置本地域名

```
$ sudo vim /etc/hosts
```

```
1 | 127.0.0.1 localhost
2 | #127.0.1.1 node0
3 |
4 | 192.168.1.81 name
5 | 192.168.1.91 node1
6 | 192.168.1.92 node2
7 | 192.168.1.93 node3
8 |
9 | # The following lines are desirable for IPv6 capable hosts
10 | ::1 ip6-localhost ip6-loopback
11 | fe00::0 ip6-localnet
12 | ff00::0 ip6-mcastprefix
13 | ff02::1 ip6-allnodes
14 | ff02::2 ip6-allrouters
```

## 8. 为每个节点配置 DNS 解析

```
$ sudo vim /etc/systemd/resolved.conf
```

```
1 | [Resolve]
2 | DNS=210.47.208.8 210.47.208.6 114.114.114.114
3 | #FallbackDNS=
4 | #Domains=
5 | #LLMNR=no
6 | #MulticastDNS=no
7 | #DNSSEC=no
8 | #Cache=yes
9 | #DNSStubListener=yes
```

## 9. 检查域名解析

```
$ ping www.baidu.com
```

按 Ctrl+c 结束

## 10. 重新启动系统

```
$ sudo reboot
```

# SSH 免密码设置

## 1. 重新生成每个节点的 ssh 秘钥

```
$ ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
```

## 2. 查看每个节点生成的公钥

```
$ ssh-keygen -y -f ~/.ssh/id_rsa
```

### 3. 生成 name 的公共公钥文件

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

### 4. 追加其他节点公钥到 name 公共公钥文件

```
$ cat ~/.ssh/id_rsa.pub | ssh name 'cat ->> /home/jack/.ssh/authorized_keys'
```

### 5. 查看 name 公共公钥文件

```
$ cat ~/.ssh/authorized_keys
```

### 6. 复制 name 的公共公钥文件到其他节点 node1,node2,node3,...

```
$ scp /home/jack/.ssh/authorized_keys node1:/home/jack/.ssh/
```

### 7. name 登录到其他节点

```
jack@node0:~$ ssh node1
```

```
jack@node1:~$ exit
```

### 8. 其他节点登录到 name

```
jack@node1:~$ ssh name
```

```
jack@node0:~$ exit
```

## 复查每个节点的基础环境

### 1. 复查每个节点的工作目录

```
$ tree -L 2 /usr/bd
```

```
1 | /usr/bd
2 |   | dfs
3 |   |   | data
4 |   |   | name
5 |   |   |
6 |   |   |
7 |   |   |
8 |   |   |
```

## 2. 复查每个节点的/etc/profile

```
sudo vim /etc/profile
```

## 3. 使每个节点的 profile 生效

```
source /etc/profile
```

## 4. 复查每个节点的 java 路径配置

```
$ java -version
```

## 5. 设置节点 name 的 hadoop-env.sh

```
sudo vim /usr/bd/hadoop-3.3.1/etc/hadoop/hadoop-env.sh
```

```
export JAVA_HOME=/usr/bd/jdk-11.0.13
export HADOOP_LOG_DIR=/usr/bd/logs
```

## 6. 复制 name 的 [hadoop-env.sh](#) 到其他节点

```
scp /usr/bd/hadoop-3.3.1/etc/hadoop/hadoop-env.sh node1:/usr/bd/hadoop-3.3.1/etc/hadoop/
```

## 7. 复查 hadoop 路径配置

```
$ hadoop
$ hadoop version
```

# 修改 name 的 hadoop 配置文件

## 1. 修改 node0 的 workers

```
$ sudo vim /usr/bd/hadoop-3.3.1/etc/hadoop/workers
```

```
Error: ENOENT: no such file or directory, open 'F:\words\handout\big-data-1\config\c
```



## 2. 复制 name 的 worker 到其他节点

```
scp /usr/bd/hadoop-3.3.1/etc/hadoop/workers node1:/usr/bd/hadoop-3.3.1/etc/hadoop/
```

## 3. 修改 name 的 core-site.xml

```
$ sudo vim /usr/bd/hadoop-3.3.1/etc/hadoop/core-site.xml
```

```
Error: ENOENT: no such file or directory, open 'F:\words\handout\big-data-1\config\c
```

## 4. 复制 node0 的 core-site.xml 到其他节点

```
scp /usr/bd/hadoop-3.3.1/etc/hadoop/core-site.xml node1:/usr/bd/hadoop-3.3.1/etc/hadoop/
```

## 5. 修改 name 的 hdfs-site.xml, 注意复制份数为 2

```
$ sudo vim /usr/bd/hadoop-3.3.1/etc/hadoop/hdfs-site.xml
```

```
Error: ENOENT: no such file or directory, open 'F:\words\handout\big-data-1\config\c
```

## 6. 复制 name 的 hdfs-site.xml 到其他节点

```
scp /usr/bd/hadoop-3.3.1/etc/hadoop/hdfs-site.xml node1:/usr/bd/hadoop-3.3.1/etc/hadoop/
```

## 7. 修改 name 的 mapred-site.xml

```
$ sudo vim /usr/bd/hadoop-3.3.1/etc/hadoop/mapred-site.xml
```

```
Error: ENOENT: no such file or directory, open 'F:\words\handout\big-data-1\config\c
```

## 8. 复制 node0 的 mapred-site.xml 到其他节点

```
scp /usr/bd/hadoop-3.3.1/etc/hadoop/mapred-site.xml node1:/usr/bd/hadoop-3.3.1/etc/hadoop/
```

## 9. 修改 name 的 yarn-site.xml

```
$ sudo vim /usr/bd/hadoop-3.3.1/etc/hadoop/yarn-site.xml
```

```
Error: ENOENT: no such file or directory, open 'F:\words\handout\big-data-1\config\c
```

## 10. 复制 name 的 yarn-site.xml 到其他节点

```
scp /usr/bd/hadoop-3.3.1/etc/hadoop/yarn-site.xml node1:/usr/bd/hadoop-3.3.1/etc/hadoop/
```

## 11. 清除遗留数据

```
$ sudo rm -rf /usr/bd/dfs/data/current/
$ ls /usr/bd/dfs/data

$ sudo rm -rf /usr/bd/dfs/name/current/
$ ls /usr/bd/dfs/name
```

# Cluster 测试

以下操作均在 name 上完成：

### 1. 格式化 HDFS

```
$ hdfs namenode -format
```

### 2. 启动 hdfs

```
jack@name:~$ start-dfs.sh
jps
```

### 3. Web 查看各节点的启动情况

- <http://192.168.1.81:9870>
- <http://192.168.1.91:9864>
- <http://192.168.1.92:9864>
- <http://192.168.1.93:9864>

### 4. jps 查看各节点的启动情况

```
jack@name:~$ jps
jack@name:~$ ssh node1
jack@node1:~$ jps
jack@node1:~$ exit
jack@name:~$ ssh node2
jack@node2:~$ jps
jack@node2:~$ exit
jack@name:~$ ssh node3
jack@node3:~$ jps
jack@node3:~$ exit
```

### 5. 启动 yarn

```
jack@name:~$ start-yarn.sh
```

### 6. Web 查看 yarn 的启动情况

- <http://192.168.1.81:8088>

## 7. jps 查看各节点的启动情况

```
jack@name:~$ jps
jack@name:~$ ssh node1
jack@node1:~$ jps
jack@node1:~$ exit
jack@name:~$ ssh node2
jack@node2:~$ jps
jack@node2:~$ exit
jack@name:~$ ssh node3
jack@node3:~$ jps
jack@node3:~$ exit
```

## 8. 启动 historyserver

```
jack@name:~$ mr-jobhistory-daemon.sh start historyserver
```

## 9. Web 查看 historyserver 的启动情况

- <http://192.168.1.81:19888>

## 10. 复查 stopall.sh

```
$ vim /usr/bd/hadoop-3.3.1/sbin/stopall.sh
```

## 11. 停止全部 Hadoop 进程

```
jack@name:~$ stopall.sh
jps
```

## 12. 复查 startall.sh

```
$ vim /usr/bd/hadoop-3.3.1/sbin/startall.sh
```

## 13. 全部启动测试

```
jack@node0:~$ startall.sh
jps
```

## 14. Web 查看启动情况

- <http://192.168.1.81:9870>
- <http://192.168.1.91:9864>
- <http://192.168.1.92:9864>
- <http://192.168.1.93:9864>
- <http://192.168.1.81:8088>
- <http://192.168.1.81:19888>

## 15. 全部停止测试

```
jack@node0:~$ stopall.sh
```

# Mapreduce

## 1. 启动集群

```
$ startall.sh
```

## 2. 在集群上创建输入文件夹

```
$ hdfs dfs -mkdir /user
$ hdfs dfs -mkdir /user/jack
$ hdfs dfs -mkdir /user/jack/input
```

## 3. 复制文件到输入文件夹

```
$ hdfs dfs -put /usr/bd/hadoop-3.3.1/etc/hadoop/*.xml /user/jack/input
```

## 4. 查看输入文件情况

```
$ hdfs dfs -ls /user/jack/input
```

## 5. 在集群上做 MapReduce 任务测试

```
$ hadoop jar /usr/bd/hadoop-3.3.1/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.1.jar
```

## 6. 查看执行结果

```
$ hdfs dfs -cat /user/jack/output/*
```

## 7. Web 查看 MapReduce 执行情况

- <http://192.168.1.81:9870>
- <http://192.168.1.81:8088>
- <http://192.168.1.81:19888>

## 8. 其他测试练习

```
$ hadoop jar /usr/bd/hadoop-3.3.1/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.1.jar
```

```
$ hadoop jar /usr/bd/hadoop-3.3.1/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.1.jar
```

```
$ hadoop jar /usr/bd/hadoop-3.3.1/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.1.jar
```

# 关机过程

## 1. 关闭集群

```
$ stopall.sh
```

## 2. 查看集群状态

- <http://192.168.1.81:9870>
- <http://192.168.1.81:8088>
- <http://192.168.1.81:19888>

## 3. 关闭虚拟机

```
$ ssh -t node1 "sudo shutdown -h now"
$ ssh -t node2 "sudo shutdown -h now"
$ ssh -t node3 "sudo shutdown -h now"
```

## 4. 关闭 VMware

## 5. 关闭计算机

# Cluster Run

关于 Hadoop 集群的网络所有节点都在一个网段，万兆网，编程客户端最好也在相同的网段，一般来讲是内部集群。

另外一种场景是，Hadoop 集群在内部万兆网上，向外部网络用户提供客户端编程。

一般的组织方式采用每个节点安装两块网卡，一块网卡是内网，一块网卡是外网。

Hadoop 集群的内外网访问需要每个节点机上安装两块网卡，还需要对 Hadoop 集群的网络进行额外设置。

我们在虚拟机条件下无法实现上述方案。我给大家的解决方案如下：

## 硬件方案

1. 针对只有热点网络的情况，买一个可以接入热点作为外网，带 Wifi 和网络的路由器，然后所有计算机包括 windows 和 Ubuntu 通过 VMware 的桥接模式都接入路由器。
2. 针对有网线接入外网的情况，可以买一个带 WiFi 的普通路由器，然后所有计算机都接入路由器提供的网络
3. 针对只有 Wifi 接入外网的情况，买一个带 Wifi 接入外网的路由器，然后所有计算机都接入路由器提供的网络

购买设备需要认真选择，花钱，费时间。  
我们可以考虑采用软件方案。

## 软件方案

1. 设置 VMware 的网络为 Nat 或主机模式均可，建议采用 Nat 模式，保证 VMware 内的虚拟机能够接入外网
2. 在 VMware 内创建虚拟机 Ubuntu1,2,3（作为 Hadoop 集群）
3. 在 VMware 内创建虚拟机 Windows10（作为 Window 客户端）
4. 上述操作系统在相同网段，可以无限制互通
5. 在 VMware 宿主机上可以通过远程桌面访问 VMWare 内的 Windows，进行软件开发和集群使用

## Eclipse 在 Hadoop 集群上运行 MapReduce 程序

有两种方式：

1. 打包成 Jar 复制到 Hadoop 集群用 `hadoop jar` 指令来执行
2. 以开发环境作为客户端与 Hadoop 集群进行交互（不仅是数据交互，包括 Map 和 reduce 交互）

## Eclipse 作为客户端与 Hadoop 交互

1. 在 windows 下设置系统环境变量 `HADOOP_USER_NAME=jack`
2. 修改 `C:/windows/system32/drivers/etc/hosts`，增加主机解析

```
192.168.1.202 namenode
192.168.1.203 datanode1
192.168.1.204 datanode2
192.168.1.205 datanode3
```

1. 官员身份运行 Eclipse
2. 选择工作空间 `D:\students\202222022222\bd1\weido`
  - Launch
3. `File > New > Other ...`
4. `Select a wizard > Map/Reduce Project`
  - Next
5. `MapReduce Project`
  - Project name: `temp.cluster`
  - Next
6. `Java Settings`
  - Finish

## 编写测试代码

1. Project Explorer > temp.cluster > src > 鼠标右键 > New > Package
2. New Java Package
  - Source folder: temp.cluster/src
  - Name: temp.cluster.max
  - Finish
3. Project Explorer > temp.cluster > src > temp.cluster.max > 鼠标右键 > New > Class
  - Name: MaxTemperature
  - [x] public static void main(String[] args)
  - Finish
4. Project Explorer > temp.cluster > src > temp.cluster.max > 鼠标右键 > New > Class
  - Name: MaxTemperatureMapper
  - Finish
5. Project Explorer > temp.cluster > src > temp.cluster.max > 鼠标右键 > New > Class
  - Name: MaxTemperatureReducer
  - Finish
6. 在项目的 src 下创建 hadoop-cluster.xml 集群配置文件

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <property>
    <name>hadoop.job.user</name>
    <value>jack</value>
  </property>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://namenode:9000</value>
  </property>
  <property>
    <name>mapred.job.tracker</name>
    <value>namenode:9001</value>
  </property>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
  <property>
    <name>yarn.resourcemanager.hostname</name>
    <value>namenode</value>
  </property>
  <property>
    <name>mapreduce.app-submission.cross-platform</name>
    <value>true</value>
  </property>
  <property>
    <name>yarn.app.mapreduce.am.env</name>
    <value>HADOOP_MAPRED_HOME=/usr/bd/hadoop-3.2.2</value>
  </property>
  <property>
    <name>mapreduce.map.env</name>
    <value>HADOOP_MAPRED_HOME=/usr/bd/hadoop-3.2.2</value>
  </property>
  <property>
    <name>mapreduce.reduce.env</name>
    <value>HADOOP_MAPRED_HOME=/usr/bd/hadoop-3.2.2</value>
  </property>
  <property>
    <name>mapred.jar</name>
    <value>E:/student/202222022222/bd1/wedo/mapred.temp.max/dist/temperature.jar</value>
  </property>
</configuration>
```

## 7. 修改 MaxTemperature.java



```

package mapred.temp.max;

import java.net.URI;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class MaxTemperature {

    static {
        try {
            System.load("D:/students/202222022222/bd1/idea/hadoop-3.2.2/bin/hadoop.dll");
        } catch (UnsatisfiedLinkError e) {
            System.err.println("Native code library failed to load.\n" + e);
            System.exit(1);
        }
    }

    public static void main(String[] args) throws Exception {
        Configuration conf=new Configuration();
        conf.addResource("hadoop-cluster.xml");

        String[] otherArgs=new String[]{"temp","tempout"};
        if(otherArgs.length!=2){
            System.err.println("Usage:Temperature <in> <out>");
            System.exit(2);
        }

        Path in = new Path(otherArgs[0]);
        Path out = new Path(otherArgs[1]);
        FileSystem fileSystem = FileSystem.get(new URI(in.toString()), conf);
        if (fileSystem.exists(out)) {
            fileSystem.delete(out, true);
        }

        Job job = Job.getInstance(conf,"Temperature");
        job.setJarByClass(MaxTemperature.class);

        job.setMapperClass(MaxTemperatureMapper.class);
        job.setReducerClass(MaxTemperatureReducer.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
    }
}

```

```
FileInputFormat.addInputPath(job,in);
FileOutputFormat.setOutputPath(job,out);

System.exit(job.waitForCompletion(true) ? 0 : 1);
}
}
```

6. 在项目空间 temp 下建立文件夹 dist

7. 鼠标右键项目名 temp > properties

8. Java Compiler

- Enable project specific settings
- Compiler compliance level: 1.8
- Apply and Close

9. 鼠标右键项目名 temp > Export...

10. Select > Java > Runnable JAR file

- Next

11. Runnable JAR file Specification

- Launch configurations: MaxTemperature - temp
- Export destination: temp\dist\temp.jar
- Package required libraries into generated JAR
- Finish

12. 鼠标右键项目名 temp > Run As > Run on Hadoop

13. Select Java Application

- Matching items: MaxTemperature - mapred.temp.max
- OK

14. 等待结果或调试

## Gitee

1. 登录到 <https://gitee.com/>
2. 进入组织空间 xxx-2022-1
3. 新建仓库

- 仓库名称: temp.cluster
- 路径: temp.cluster
- 仓库介绍: 最高气温统计 MapReduce 程序 Cluster 运行
- 初始化仓库
  - 语言: Java
  - 添加 .gitignore: Eclipse
  - 添加开源许可证: MIT
- 设置模板
  - Readme 文件
  - Issue 模板文件
  - Pull Request 模板文件
- 创建

## Git

1. 官员身份运行 Eclipse
2. 选择工作空间 D:\students\202222022222\bd1\weido
  - Launch
3. temp.cluster 鼠标右键 > Team > Share Project...
4. Configure Git Repository
  - [x] Use or create repository in parent folder of project
  - 点击 Project > temp.cluster
  - Create Repository
  - Finish
5. temp.cluster 鼠标右键 > Team > Add to Index
6. temp.cluster 鼠标右键 > Team > Commit
  - Commit Message: First
  - Commit
7. 修改文件 D:\students\202222022222\bd1\works\temp.cluster.git\config

```
[core]
    repositoryformatversion = 0
    filemode = false
    bare = false
    logallrefupdates = true
    symlinks = false
    ignorecase = true
[remote "origin"]
    url = git@gitee.com:ls-2022-1/temp.cluster.git
    fetch = +refs/heads/*:refs/remotes/origin/*
[branch "master"]
    remote = origin
    merge = refs/heads/master
[user]
    name = Dawei Zhang
    email = daweiz@lnnu.edu.cn
```

8. 鼠标右键 temp.cluster > Show in Local Terminal > Terminal

- git pull --allow-unrelated-histories

9. 修改 .gitignore 冲突

```
/bin/
```

10. temp.cluster 鼠标右键 > Team > Add to Index

11. temp.cluster 鼠标右键 > Team > Commit

- Commit and push

12. Pushed to temp.cluster refs/heads/master -origin

- Close

## Zookeeper

1. 官网下载 apache-zookeeper-3.8.0-bin.tar.gz

- sudo wget <https://dlcdn.apache.org/zookeeper/zookeeper-3.8.0/apache-zookeeper-3.8.0-bin.tar.gz>

2. sudo tar -zxvf apache-zookeeper-3.8.0-bin.tar.gz

3. sudo mv apache-zookeeper-3.8.0-bin /usr/bd

4. sudo mkdir /usr/bd/zoo

## 单机模式

1. cd /usr/bd/apache-zookeeper-3.8.0-bin/conf/

2. sudo cp zoo\_sample.cfg zoo.cfg

3. sudo vim zoo.cfg

```
tickTime=2000
initLimit=10
syncLimit=5
dataDir=/usr/bd/data
clientPort=2181
```

4. cd ..

5. sudo ./bin/zkServer.sh start

6. sudo ./bin/zkServer.sh status

## 伪分布模式

zookeeper01

sudo tar -zxvf apache-zookeeper-3.8.0-bin.tar.gz

sudo mv apache-zookeeper-3.8.0-bin /usr/bd/zookeeper01

cd /usr/bd/zookeeper01

sudo mkdir data

sudo vim data/myid

0

cd /usr/bd/zookeeper01/conf

sudo cp zoo\_sample.cfg zoo.cfg

sudo vim zoo.cfg

```
tickTime=2000
initLimit=10
syncLimit=5
dataDir=/usr/bd/zookeeper01/data
clientPort=2181
server.0=namenode:2888:3888
server.1=namenode:4888:5888
server.2=namenode:6888:7888
4lw.commands.whitelist=*
```

cd /usr/bd/zookeeper01/bin

sudo ./zkCli.sh

- create /appconfig "My Config"

- create /appconfig/hosts 192.168.1.81

- Ctrl+c

```
sudo ./zkServer.sh start
```

zookeeper02

```
sudo tar -zxvf apache-zookeeper-3.8.0-bin.tar.gz
```

```
sudo mv apache-zookeeper-3.8.0-bin /usr/bd/zookeeper02
```

```
cd /usr/bd/zookeeper02
```

```
sudo mkdir data
```

```
sudo vim data/myid
```

1

```
cd /usr/bd/zookeeper02/conf/
```

```
sudo cp zoo_sample.cfg zoo.cfg
```

```
sudo vim zoo.cfg
```

```
tickTime=2000
```

```
initLimit=10
```

```
syncLimit=5
```

```
dataDir=/usr/bd/zookeeper02/data
```

```
clientPort=2182
```

```
server.0=namenode:2888:3888
```

```
server.1=namenode:4888:5888
```

```
server.2=namenode:6888:7888
```

```
4lw.commands.whitelist=*
```

```
cd /usr/bd/zookeeper01/bin
```

```
sudo ./zkCli.sh
```

```
- create /appconfig "My Config"
```

```
- create /appconfig/hosts 192.168.1.81
```

- Ctrl+c

```
sudo ./zkServer.sh start
```

zookeeper03

```
sudo tar -zxvf apache-zookeeper-3.8.0-bin.tar.gz
```

```
sudo mv apache-zookeeper-3.8.0-bin /usr/bd/zookeeper03
```

```
cd /usr/bd/zookeeper03
```

```
sudo mkdir data
```

```
sudo vim data/myid
```

```
cd /usr/bd/zookeeper03/conf/
sudo cp zoo_sample.cfg zoo.cfg
sudo vim zoo.cfg
```

```
tickTime=2000
initLimit=10
syncLimit=5
dataDir=/usr/bd/zookeeper03/data
clientPort=2183
server.0=namenode:2888:3888
server.1=namenode:4888:5888
server.2=namenode:6888:7888
4lw.commands.whitelist=*
```

```
cd /usr/bd/zookeeper01/bin
sudo ./zkCli.sh
- create /appconfig "My Config"
- create /appconfig/hosts 192.168.1.81
- Ctrl+c
sudo ./zkServer.sh start
```

## 集群模式

### zkui

1. cd ~
2. git clone <https://github.com/DeemOpen/zkui.git>
3. cd zkui
4. mvn clean install
 

```
sudo mkdir /usr/bd/zkui
sudo cp config.cfg /usr/bd/zkui
sudo cp target/zkui-2.0-SNAPSHOT-jar-with-dependencies.jar /usr/bd/zkui/zkui.jar
cd /usr/bd/zkui
sudo vim config.cfg
```

```
zkServer=namenode:2181,namenode:2182,namenode:2183
```

```
cd /usr/bd/zkui
sudo java -jar zkui.jar
http://192.168.1.81:9090/
- username: admin
- password: manager
- username: appconfig
- password: appconfig
```

## zabbix

1. Java 官网, <https://www.oracle.com/java/technologies/> ↩
2. Java 下载, <https://www.oracle.com/java/technologies/downloads/#java11> ↩
3. Hadoop 官网, <http://hadoop.apache.org/> ↩ ↩
4. Hadoop 下载地址, <https://hadoop.apache.org/releases.html> ↩ ↩
5. Eclipse 官网, <https://www.eclipse.org/> ↩
6. Eclipse 下载, <https://www.eclipse.org/downloads/packages/> ↩
7. eclipse-hadoop3x 官网, <https://github.com/Wooooosz/eclipse-hadoop3x> ↩
8. eclipse-hadoop3x 下载, <https://github.com/Wooooosz/eclipse-hadoop3x/tree/master/release> ↩
9. winutils 官网, <https://github.com/cdarlint/winutils> ↩
10. winutils 下载, <https://github.com/cdarlint/winutils/tree/master/hadoop-3.2.2/bin> ↩
11. VMware Workstation Pro 官网, <https://www.vmware.com/cn/products/workstation-pro.html> ↩
12. VMware Workstation Pro 下载, <https://www.vmware.com/cn/products/workstation-pro/workstation-pro-evaluation.html> ↩
13. Ubuntu 官网, <https://ubuntu.com/> ↩
14. Ubuntu 下载, <https://ubuntu.com/download/server> ↩
15. PuTTY 官网, <https://www.chiark.greenend.org.uk/~sgtatham/putty/> ↩
16. PuTTY 下载, <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html> ↩
17. <https://git-scm.com/> ↩
18. Java 官网, <https://www.oracle.com/java/technologies/> ↩
19. Java SE 11 (LTS) 下载, <https://www.oracle.com/java/technologies/javase-jdk11-downloads.html> ↩
20. Apache 官网, <https://www.apache.org/> ↩