

# 1. 项目背景

现在是互联网的时代, 每个人的生活中都会使用到互联网的各种应用, 我们会进行网络购物, 会进行新闻浏览, 视频浏览, 微信聊天等等, 当我们在使用互联网的时候, 我们的所有的数据都需要通过运行商(电信, 移动, 联通)进行数据的发送和接收, 对于每一个访问, 运营商都可以获取到对应的请求信息, 我们可以通过对网络请求的信息分析, 及时掌握互联网的动态和行业前沿, 并且根据用户的请求访问数据, 我们可以分析互联网行业的发展现状和每个城市的互联网的发展程度等等. 通过对于互联网的发展的相关指标分析, 可以为政府部门, 商业公司提供一些决策分析的数据



1. 互联网数字经济时代, 大数据已经成为生产要素之一
2. 通过分析脱敏后的用户行为数据, 让数据产生价值, 让数据为企业发展, 为行业建设提供一些决策的建设性的依据和支撑
3. 通过用户行为的分析, 增强网络安全和信息数据安全的建设

## 2. 项目的价值

### 2.1. 企业价值

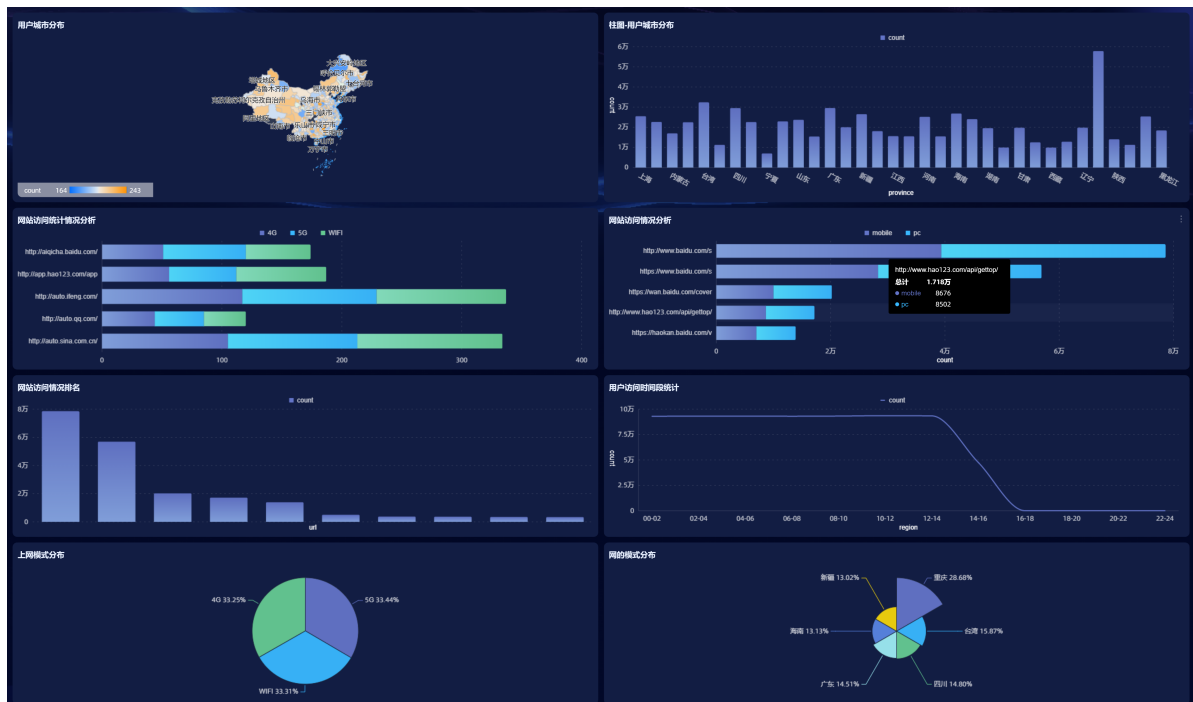
1. 为相关统计部门提供数据支持, 比如上网的用户时长, 上网的用户人数, 上网模式的占比等等
2. 为行业发展的预测提供数据支撑
3. 为企业发展提供数据支撑, 分析每个网站的访问情况

### 2.2. 学习价值

1. 完整的大数据项目实战流程

2. 最新的大数据生态技术 Hadoop3.1.3, Spark3, Hive3.1.2
3. 数据仓库的设计和建设

## 3. 项目的最终效果



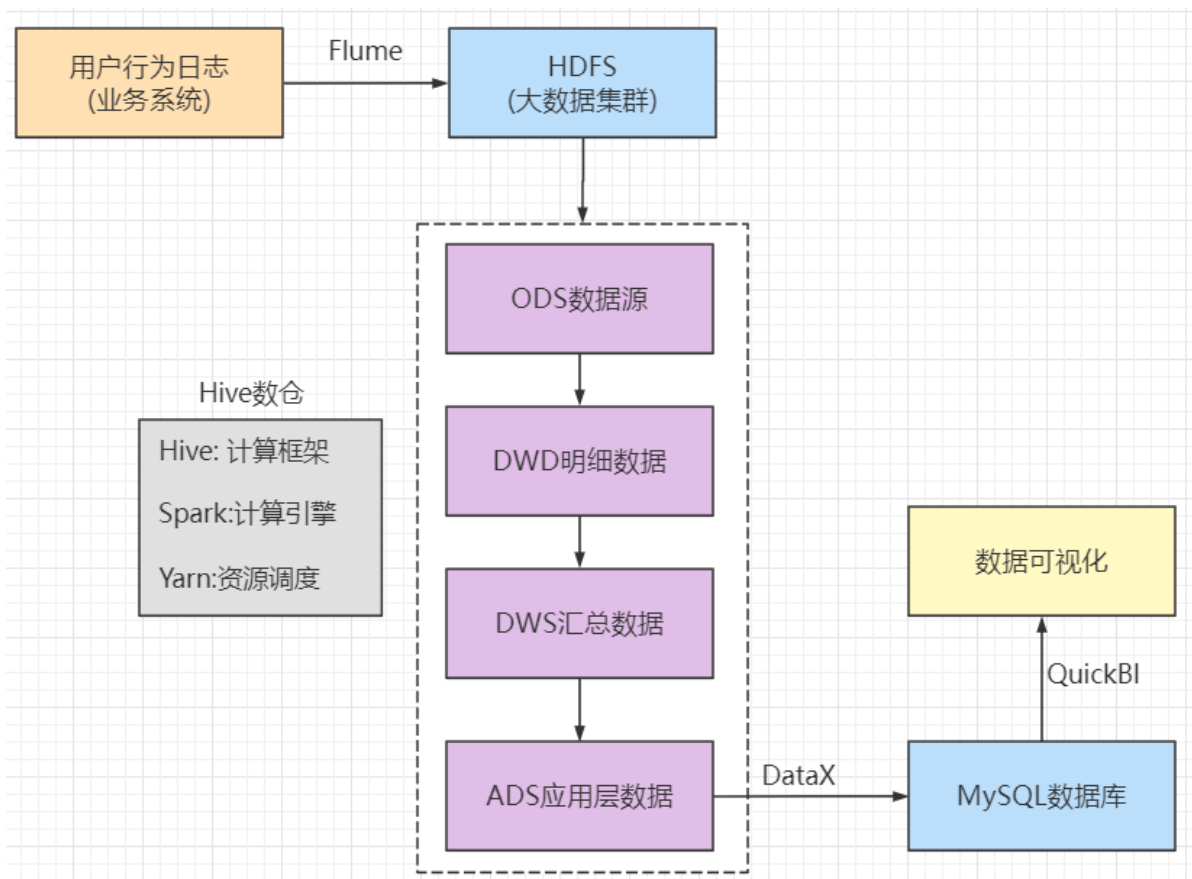
## 4. 学习收获

- Hadoop基本环境搭建 大数据环境的统一配置, HDFS分布式存储环境搭建, Yarn分布式任务调度
- MySQL数据库的安装和应用
- SpringBoot项目的日志生成生成程序的流程分析
- Flume日志采集到HDFS, 并且完成shell脚本的编写
- Hive-on-Spark环境的搭建
- Hive的基本操作, 外部表, 分区表的创建, 表数据的导入
- 数据仓库的分层架构 ODS --> DWD --> DWS --> ADS 数仓结构设计
- DataX技术从ADS同步数据到MySQL数据库
- QuickBI完成数据可视化处理

## 5. 学习要求

- 使用过Linux操作系统, 会熟练的使用常见的Linux操作系统命令
- 具有大数据的基本知识, 对于Hadoop, Hive有基本的了解
- 比较熟悉SQL语句, 特别是查询的SQL语句的基本语法
- 项目是完全从0开始, 不用担心学不会

## 6. 项目的系统流程分析



## 7. 数据格式说明

字段英文名称	字段说明	示例值
client_ip	请求的客户端IP地址	139.212.175.95
device_type	请求的设备类型, 手机, mobile 或者 电脑 pc	mobile
time	请求的时间戳 毫秒数	1659956155000
type	上网的数据格式 4G, 5G 和 WiFi	4G
device	上网的设备的唯一标记	cf287649b5b1443c903b7405fa08cb07
url	请求的目标地址	<a href="https://www.yy.com/travel">https://www.yy.com/travel</a>

完整一条日志记录格式

```
{
  "client_ip": "139.212.175.95",
  "device_type": "mobile",
  "time": 1659959785000,
  "type": "5G",
  "device": "cf287649b5b1443c903b7405fa08cb07",
  "url": "https://www.yy.com/travel"
}
```



# 1. 集群规划

	biz01	hadoop01	hadoop02	hadoop03
用途	业务系统	集群服务器	集群服务器	集群服务器
jdk1.8	√	√	√	√
MySQL5.7	√			
hdfs	客户端	Namenode, Datanode	Datanode	Datanode
yarn	客户端	ResouceManager NodeManager	NodeManager	NodeManager
历史服务器		√		
Flume	√			
Spark				√
Hive				√
DataX				√

# 2. 统一环境配置

## 2.1. [所有节点] IP地址设置

- 修改ip地址

```
vi /etc/sysconfig/network-scripts/ifcfg-ens33
```

```
TYPE="Ethernet"
PROXY_METHOD="none"
BROWSER_ONLY="no"
BOOTPROTO="static" # 设置为静态ip static
DEFROUTE="yes"
IPV4_FAILURE_FATAL="no"
NAME="ens33" # 网卡的名称
DEVICE="ens33" # 设备的名称
ONBOOT="yes" # 设置为yes，表示开机启动
IPADDR="192.168.46.150" # ip地址
PREFIX="24" # 子网掩码
GATEWAY="192.168.46.2" # 网关
DNS1="114.114.115.115" # DNS
```

- 重启网络服务

```
systemctl restart network
```

- 测试网络

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

如果可以ping通公网: 说明 ip地址和网关都配置正确

如果通过 ip addr 不能查看到ip地址, 说明配置有错误

如果可以ping通内网 192.168.46.1 但是不能ping通外网的话, 则说明网关配置有错误

## 2.2. [所有节点] 设置主机名

- 编辑主机名配置文件

```
vi /etc/hostname
```

```
hadoop01
```

## 2.3. [所有节点] 设置域名映射解析

- 编辑hosts文件

```
vi /etc/hosts
```

```
192.168.46.145 hadoop01 hadoop01
```

```
192.168.46.146 hadoop02 hadoop02
```

```
192.168.46.147 hadoop03 hadoop03
```

```
192.168.46.148 biz01 biz01
```

## 2.4. [所有节点] 关闭防火墙和Selinux

- 关闭防火墙

```
systemctl stop firewalld.service
systemctl disable firewalld.service
```

- 关闭Selinux

```
vi /etc/selinux/config
```

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=disabled
# SELINUXTYPE= can take one of three values:
#     targeted - Targeted processes are protected,
```

```
# minimum - Modification of targeted policy. Only selected processes are
protected.
# mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

## 2.5. [所有节点] 配置免密登录

1. 在所有节点生成公钥和私钥

```
ssh-keygen -t rsa
```

后面直接所有的交互都敲回车 即可

2. 拷贝公钥到每台服务器

```
ssh-copy-id hadoop01
ssh-copy-id hadoop02
ssh-copy-id hadoop03
```

3. 验证ssh登录

```
ssh hadoop01
exit # 退出ssh登录
```

## 2.6. [所有节点] 配置服务器节点时钟同步

1. 在所有节点安装ntpd

```
yum install -y ntpdate
```

2. 增加定时任务

```
crontab -e
*/1 * * * * /usr/sbin/ntpdate -u ntp4.aliyun.com > /dev/null 2>&1
```

## 2.7. [所有节点] 安装常用软件

```
yum install -y vim
yum install -y net-tools
yum install -y lrzsz
yum install -y rsync
yum install -y wget
```

## 2.8. [所有节点] 创建统一目录

```
mkdir -p /bigdata/{soft,server}
```

/bigdata/soft          安装文件的存放目录

/bigdata/server      软件安装的目录

## 3. 定义同步数据脚本

### 3.1. [所有节点] 安装软件rsync

```
yum install -y rsync
```

### 3.2. [hadoop01] 配置同步脚本

```
mkdir /root/bin
cd /root/bin
vim xsync

#!/bin/bash
#1 获取命令输入参数的个数，如果个数为0，直接退出命令
paramnum=$#
echo "paramnum:$paramnum"
if (( paramnum == 0 )); then
    echo no params;
    exit;
fi
# 2 根据传入参数获取文件名称
p1=$1
file_name=`basename $p1`
echo fname=$file_name
#3 获取输入参数的绝对路径
pdir=`cd -P $(dirname $p1); pwd`
echo pdir=$pdir
#4 获取用户名称
user=`whoami`
#5 循环执行rsync
current=`hostname`
nodes=$(cat /root/bin/works)
for host in $nodes; do
    echo ----- $host -----
    if [ "$host" != "$current" ];then
        rsync -rvl $pdir/$file_name $user@$host:$pdir
    fi
done
```

### 3.3. [hadoop01] 创建works文件



```
cd /root/bin
```

```
vi workers
```

```
hadoop01
```

```
hadoop02
```

```
hadoop03
```

### 3.4. 添加环境变量

```
vi /etc/profile.d/custom_env.sh
```

```
#!/bin/bash
```

```
# root/bin
```

```
export PATH=$PATH:/root/bin
```

```
source /etc/profile
```

设置文件执行权限

```
chmod u+x /root/bin/xsync
```

### 3.5. 测试同步脚本

## 4. jdk环境安装

1. 把安装的软件上传到/bigdata/soft 目录

 image-20220619154309897

2. 解压到指定目录

-C :指定解压到指定目录

```
tar -zxvf /bigdata/soft/jdk-8u241-linux-x64.tar.gz -C /bigdata/server/
```

3. 创建一个软链接

```
cd /bigdata/server
```

```
ln -s jdk1.8.0_241/ jdk1.8
```

4. 配置环境变量

```
vi /etc/profile.d/custom_env.sh
```

```
export JAVA_HOME=/bigdata/server/jdk1.8
```

```
export PATH=$JAVA_HOME/bin:$PATH
```

重新加载配置文件

```
source /etc/profile
```

#### 5. 测试验证

```
[root@node01 server]# java -version
java version "1.8.0_241"
Java(TM) SE Runtime Environment (build 1.8.0_241-b07)
Java HotSpot(TM) 64-Bit Server VM (build 25.241-b07, mixed mode)
```

#### 6. 同步至所有节点

```
# 同步到biz01, hadoop01, hadoop02, hadoop03
xsync /etc/profile/custom_env.sh
xsync /bigdata/server/jdk1.8.0_241
xsync /bigdata/server/jdk1.8

scp -r /etc/profile/custom_env.sh biz01:/etc/profile/custom_env.sh
scp -r /bigdata/server/jdk1.8.0_241 biz01:/bigdata/server/jdk1.8.0_241
在biz01 创建软链接
ln -s jdk1.8.0_241/ jdk1.8
```

## 5. MySQL数据库安装

### 5.1. 卸载已经安装的MySQL数据库

```
## 查询MySQL相关的依赖
rpm -qa |grep mysql
## 如果存在，则通过rpm -e --nodeps 进行卸载
```

### 5.2. 获取rpm在线安装仓库文件

```
wget https://dev.mysql.com/get/mysql80-community-release-el7-6.noarch.rpm
```

### 5.3. 安装mysql的仓库文件

```
rpm -ivh mysql80-community-release-el7-6.noarch.rpm
```

### 5.4. 修改mysql仓库的配置文件

```
cd /etc/yum.repos.d/
mysql-community.repo: 用于指定下载哪个版本的安装包
mysql-community-source.repo: 用于指定下载哪个版本的源码

`禁用8.0的版本，启用5.7的版本`
```

### 5.5. 安装MySQL5.7

```
## 导入签名的信息key
rpm --import https://repo.mysql.com/RPM-GPG-KEY-mysql-2022
## 安装5.7
yum install -y mysql-community-server
```

## 5.6. 启动数据库

---

```
systemctl start mysqld
systemctl status mysqld
systemctl enable mysqld
```

## 5.7. 登录数据库

---

```
## 查看初始密码
less /var/log/mysqld.log |grep pass

## 登录数据库
mysql -uroot -p'XRY0460efV<7'
```

## 5.8. 修改MySQL数据库密码策略

---

```
set global validate_password_length=4;
set global validate_password_policy=0;
```

## 5.9. 创建远程登录用户

---

```
ALTER USER 'root'@'localhost' IDENTIFIED BY '123456';
#创建一个远程登录用户
create user 'root'@'%' identified by '123456';
## 设置远程登录权限
grant all privileges on *.* to 'root'@'%';
```

## 5.10. 设置服务器编码为utf8

---

```
vi /etc/my.cnf
## 在mysqld下面设置
character_set_server=utf8

## 重启服务
systemctl restart mysqld
```

# 6. Hadoop集群安装

## 6.1. 集群规划

	hadoop01	hadoop02	hadoop03
角色	主节点	从节点	从节点
NameNode	√		
DataNode	√	√	√
ResourceManager	√		
NodeManager	√	√	√
SecondaryNameNode		√	
Historyserver	√		

## 6.2. 上传安装包到hadoop01

## 6.3. 解压到指定目录

```
tar -zxvf /bigdata/soft/hadoop-3.3.3.tar.gz -C /bigdata/server/
```

## 6.4. 创建软链接

```
cd /bigdata/server
ln -s hadoop-3.3.3/ hadoop
```

## 6.5. 常见的Hadoop软件目录说明

目录	作用	说明
bin/	Hadoop最基本的管理脚本和使用脚本	hdfs: 文件上传命令 hadoop文件管理基础命令 yarn: 资源调度相关 mapred: 程序运行, 启动历史服务器
etc/	Hadoop配置文件的目录	core-site.xml hdfs-site.xml mapred-site.xml yarn-site.xml
include/	对外提供的编程库头文件	对外提供的编程库头文件（具体动态库和静态库在lib目录中）， 这些头文件均是用C++定义的，通常用于C++程序访问HDFS或者编写MapReduce程序
lib/	动态库和静态库	该目录包含了Hadoop对外提供的编程动态库和静态库，与include目录中的头文件结合使用。
libexec/	shell配置文件	各个服务对用的shell配置文件所在的目录， 可用于配置日志输出、启动参数（比如JVM参数）等基本信息。
sbin/	Hadoop管理命令	主要包含HDFS和YARN中各类服务的启动/关闭脚本
share/	官方自带示例	Hadoop各个模块编译后的jar包所在的目录

## 6.6. Hadoop配置文件修改

Hadoop安装主要就是配置文件的修改，一般在 **主节点进行修改**，完毕后scp分发给其他各个 **从节点机器**。

### 6.6.1. hadoop-env.sh

文件中设置的是Hadoop运行时需要的环境变量。**JAVA\_HOME** 是必须设置的，即使我们当前的系统中设置了JAVA\_HOME，它也是不认识的，因为Hadoop即使是在本机上执行，它也是把当前的执行环境当成 **远程服务器**。

```
vim hadoop-env.sh

54行的JAVA_HOME的设置

export JAVA_HOME=/bigdata/server/jdk1.8

在文件末尾添加如下内容

export HDFS_NAMENODE_USER=root
export HDFS_DATANODE_USER=root
export HDFS_SECONDARYNAMENODE_USER=root
export YARN_RESOURCEMANAGER_USER=root
export YARN_NODEMANAGER_USER=root
```

## 6.6.2. core-site.xml

hadoop的核心配置文件，有默认的配置项 `core-default.xml`。

core-default.xml与core-site.xml的功能是一样的，如果在core-site.xml里没有配置的属性，则会自动会获取core-default.xml里的相同属性的值。

```
cd /bigdata/server/hadoop/etc/hadoop/  
vim core-site.xml
```

在文件的configuration的标签中添加以下内容:

```
<property>  
  <name>fs.defaultFS</name>  
  <value>hdfs://hadoop01:8020</value>  
</property>  
  
<property>  
  <name>hadoop.tmp.dir</name>  
  <value>/bigdata/data/hadoop</value>  
</property>  
  
<!-- 设置HDFS web UI用户身份 -->  
<property>  
  <name>hadoop.http.staticuser.user</name>  
  <value>root</value>  
</property>  
  
<!-- 整合hive -->  
<property>  
  <name>hadoop.proxyuser.root.hosts</name>  
  <value>*</value>  
</property>  
  
<property>  
  <name>hadoop.proxyuser.root.groups</name>  
  <value>*</value>  
</property>
```

## 6.6.3. hdfs-site.xml

HDFS的核心配置文件，有默认的配置项 `hdfs-default.xml`。

hdfs-default.xml与hdfs-site.xml的功能是一样的，如果在hdfs-site.xml里没有配置的属性，则会自动会获取hdfs-default.xml里的相同属性的值。

```
cd /bigdata/server/hadoop/etc/hadoop/  
vim hdfs-site.xml
```

```
<!-- 指定secondarynamenode运行位置 -->  
<property>  
  <name>dfs.namenode.secondary.http-address</name>  
  <value>hadoop02:50090</value>  
</property>
```

## 6.6.4. mapred-site.xml

MapReduce的核心配置文件，有默认的配置项 `mapred-default.xml`。

mapred-default.xml与mapred-site.xml的功能是一样的，如果在mapred-site.xml里没有配置的属性，则会自动会获取mapred-default.xml里的相同属性的值。

```
cd /bigdata/server/hadoop/etc/hadoop/
```

```
vim mapred-site.xml
```

```
<property>
  <name>mapreduce.framework.name</name>
  <value>yarn</value>
</property>
<property>
  <name>yarn.app.mapreduce.am.env</name>
  <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
</property>
<property>
  <name>mapreduce.map.env</name>
  <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
</property>
<property>
  <name>mapreduce.reduce.env</name>
  <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
</property>
```

## 6.6.5. yarn-site.xml

YARN的核心配置文件，有默认的配置项 `yarn-default.xml`。

yarn-default.xml与yarn-site.xml的功能是一样的，如果在yarn-site.xml里没有配置的属性，则会自动会获取yarn-default.xml里的相同属性的值。

```
cd /bigdata/server/hadoop/etc/hadoop/
```

```
vim yarn-site.xml
```

```
<!-- 指定YARN的主角色（ResourceManager）的地址 -->
<property>
  <name>yarn.resourcemanager.hostname</name>
  <value>hadoop01</value>
</property>

<!-- NodeManager上运行的附属服务。需配置成mapreduce_shuffle，才可运行MapReduce程序默认
值："" -->
<property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
</property>

<!-- 是否将对容器实施物理内存限制 -->
<property>
  <name>yarn.nodemanager.pmem-check-enabled</name>
```

```

    <value>false</value>
  </property>

  <!-- 是否将对容器实施虚拟内存限制。 -->
  <property>
    <name>yarn.nodemanager.vmem-check-enabled</name>
    <value>false</value>
  </property>
  <!-- 开启日志聚集 -->
  <property>
    <name>yarn.log-aggregation-enable</name>
    <value>true</value>
  </property>

  <!-- 设置yarn历史服务器地址 -->
  <property>
    <name>yarn.log.server.url</name>
    <value>http://hadoop01:19888/jobhistory/logs</value>
  </property>

  <!-- 保存的时间7天 -->
  <property>
    <name>yarn.log-aggregation.retain-seconds</name>
    <value>604800</value>
  </property>

```

### 6.6.6. workers

workers文件里面记录的是集群主机名。主要作用是配合一键启动脚本如start-dfs.sh、stop-yarn.sh用来进行集群启动。这时候workers文件里面的主机标记的就是从节点角色所在的机器。

```

cd /bigdata/server/hadoop/etc/hadoop/

vim workers

```

```

hadoop01
hadoop02
hadoop03

```

## 6.7. 同步hadoop软件包到hadoop02和hadoop03

```

scp -r hadoop-3.3.3/ hadoop02:$PWD

```

```

scp -r hadoop-3.3.3/ hadoop03:$PWD

```

在hadoop02节点配置软链接

```

ln -s hadoop-3.3.3/ hadoop

```

在hadoop03节点配置软链接

```

ln -s hadoop-3.3.3/ hadoop

```



## 6.8. [所有节点] 配置环境变量

```
vim /etc/profile.d/custom_env.sh

export HADOOP_HOME=/bigdata/server/hadoop

export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin

source /etc/profile
```

## 6.9. Hadoop集群启动

### 6.9.1. 启动方式

要启动Hadoop集群，需要启动 **HDFS** 和 **YARN** 两个集群。

注意：首次启动HDFS时，必须在主节点 **hadoop01** 对其进行格式化操作。本质上是一些清理和准备工作，因为此时的HDFS在物理上还是不存在的。

```
hadoop namenode -format
```

### 6.9.2. 手动单个节点启动

在 **主节点hadoop01** 启动namenode

```
cd /bigdata/server/hadoop/bin

./hdfs --daemon start namenode
```

在 **hadoop02** 启动secondarynamenode

```
cd /bigdata/server/hadoop/bin

./hdfs --daemon start secondarynamenode
```

在 **所有节点** 启动datanode

```
cd /bigdata/server/hadoop/bin

./hdfs --daemon start datanode
```

查看进程情况

```
jpg

netstat -ntlp

其中hdfs的web端口: hadoop01:9870已经可以正常访问
```

在 **主节点hadoop01** 启动ResourceManager

```
cd /bigdata/server/hadoop/bin
```

```
./yarn --daemon start resourcemanager
```

在 **所有节点** 启动Nodemanager

```
cd /bigdata/server/hadoop/bin
```

```
./yarn --daemon start nodemanager
```

如果想要停止某个节点上某个角色，只需要把命令中的 **start** 改为 **stop** 即可。

### 6.9.3. 一键脚本启动

如果配置了 **etc/hadoop/workers** 和 **ssh免密登录**，则可以使用程序脚本启动所有Hadoop两个集群的相关进程，在主节点所设定的机器上执行。

**hdfs: /bigdata/server/hadoop/sbin/start-dfs.sh**

**yarn: /bigdata/server/hadoop/sbin/start-yarn.sh**

停止脚本

```
hdfs: /bigdata/server/hadoop/sbin/stop-dfs.sh
```

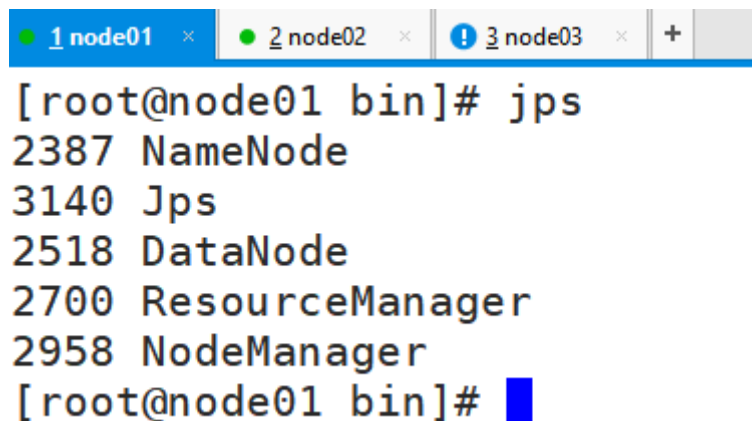
```
yarn: /bigdata/server/hadoop/sbin/stop-yarn.sh
```

完整的一键启动hdfs和yarn脚本

start-all.sh: 启动所有的hdfs和yarn的脚本

stop-all.sh: 停止所有的hdfs和yarn的脚本

## 6.10. 启动后的效果



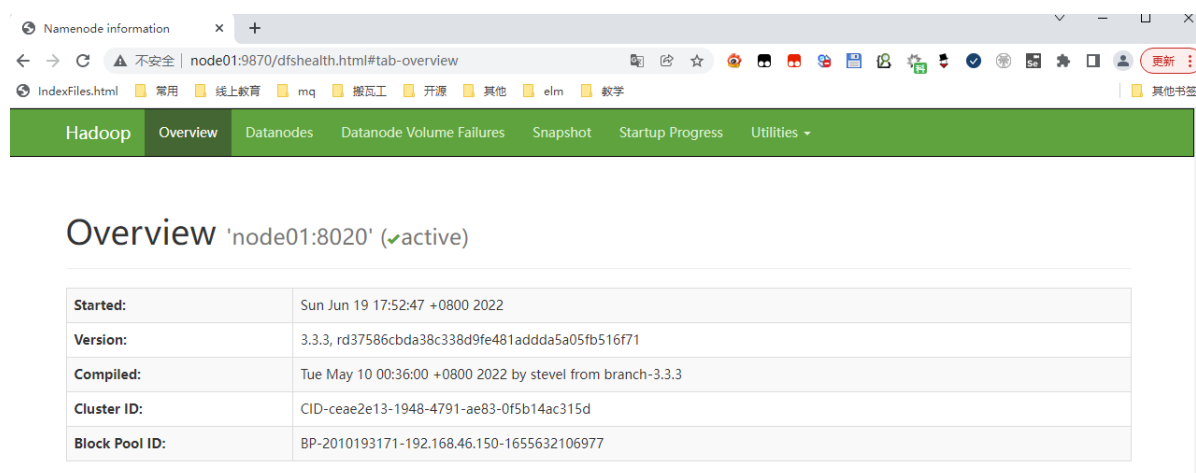
```
[root@node01 bin]# jps
2387 NameNode
3140 Jps
2518 DataNode
2700 ResourceManager
2958 NodeManager
[root@node01 bin]#
```

```
1 node01 x 2 node02 x ! 3 node03 x +
[root@node02 bin]# jps
2352 NodeManager
2608 SecondaryNameNode
2692 Jps
2183 DataNode
[root@node02 bin]#
```

```
1 node01 x 2 node02 x 3 node03 x +
[root@node03 bin]# jps
2184 DataNode
2345 NodeManager
2527 Jps
[root@node03 bin]#
```

## 6.11. 集群Web访问UI

hdfs: <http://hadoop01:9870>




Overview 'node01:8020' (✓active)

Started:	Sun Jun 19 17:52:47 +0800 2022
Version:	3.3.3, rd37586cbda38c338d9fe481addda5a05fb516f71
Compiled:	Tue May 10 00:36:00 +0800 2022 by stevel from branch-3.3.3
Cluster ID:	CID-ceae2e13-1948-4791-ae83-0f5b14ac315d
Block Pool ID:	BP-2010193171-192.168.46.150-1655632106977

yarn:<http://hadoop01:8088>

node01:8088/cluster/apps

IndexFiles.html 常用 线上教育 mq 搬瓦工 开源 其他 elm 教学 其他书签



Cluster

About

Nodes

Node Labels

Applications

NEW

NEW SAVING

SUBMITTED

ACCEPTED

RUNNING

FINISHED

FAILED

KILLED

Scheduler

Tools

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running
0	0	0	0	<memory:0 f

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes
3	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:1024, vCores:1>

Show 20 entries

ID	User	Name	Application Type	Application Tags	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State
Showing 0 to 0 of 0 entries										

## 6.12. MapReduce JobHistory

JobHistory用来记录已经finished的mapreduce运行日志，日志信息存放于HDFS目录中，默认情况下没有开启此功能，需要在 `mapred-site.xml` 中配置并手动启动。

### 6.12.1. 修改mapred-site.xml

```
cd /bigdata/server/hadoop/etc/hadoop/
vim mapred-site.xml
```

```
<property>
<name>mapreduce.jobhistory.address</name>
<value>hadoop02:10020</value>
</property>
<property>
<name>mapreduce.jobhistory.webapp.address</name>
<value>hadoop02:19888</value>
</property>
```

```
scp mapred-site.xml hadoop02:$PWD
scp mapred-site.xml hadoop03:$PWD
```

## 6.13. 在hadoop02节点启动JobHistory

```
cd /bigdata/server/hadoop/bin
./mapred --daemon start historyserver
```

<http://hadoop02:19888/jobhistory>

在hdfs创建一个目录:

```
hdfs dfs -mkdir /input
```

## 上传文件到hdfs的/input目录

```
hdfs dfs -put start-all.sh /input
```

## 运行示例程序

```
hadoop jar /bigdata/server/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.3.jar  
wordcount /input /output
```

Show  entries

Search:


<input type="checkbox"/>		Permission		Owner		Group		Size		Last Modified		Replication		Block Size		Name	
<input type="checkbox"/>		-rw-r--r--		root		supergroup		2.17 KB		Jun 19 18:47		3		128 MB		start-all.sh	

Showing 1 to 1 of 1 entries

Previous

1

Next



Cluster

[About](#)
[Nodes](#)
[Node Labels](#)
[Applications](#)

[NEW](#)
[NEW SAVING](#)
[SUBMITTED](#)
[ACCEPTED](#)
[RUNNING](#)
[FINISHED](#)
[FAILED](#)
[KILLED](#)

[Scheduler](#)

Tools

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running
1	0	0	1	0

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned
3	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:1024, vCores:1>

Show 20 entries

ID	User	Name	Application Type	Application Tags	Queue	Application Priority	StartTime	LaunchTime
<a href="#">application_1655635432951_0002</a>	root	word count	MAPREDUCE		default	0	Sun Jun 19 18:48:19 +0800 2022	Sun Jun 19 18:48:20 +0800 2022

Showing 1 to 1 of 1 entries



# JobHistory

Application

AboutJobs

Tools

Retired Jobs

Show20entries

Sea

Submit Time	Start Time	Finish Time	Job ID	Name	User	Queue	State	Maps Total	Maps Completed	Reduces Total
2022.06.19 18:48:19 CST	2022.06.19 18:48:27 CST	2022.06.19 18:48:41 CST	job_1655635432951_0002	word count	root	default	SUCCEEDED	1	1	1

Submit Tir

Start Time

Finish Tim

Job ID

Name

User

Queue

State

Maps 1

Maps Comple

Reduces 1

Showing 1 to 1 of 1 entries

FirstPrev

# 1. 集群规划

	node01	node02	node03
角色	主节点	从节点	从节点
NameNode	√		
DataNode	√	√	√
ResourceManager	√		
NodeManager	√	√	√
SecondaryNameNode		√	
Historyserver		√	

# 2. 上传安装包到node01

# 3. 解压到指定目录

```
tar -zxvf /bigdata/soft/hadoop-3.3.3.tar.gz -C /bigdata/server/
```

# 4. 创建软链接

```
cd /bigdata/server  
  
ln -s hadoop-3.3.3/ hadoop
```

# 5. 常见的Hadoop软件目录说明

目录	作用	说明
bin/	Hadoop最基本的管理脚本和使用脚本	hdfs: 文件上传命令 hadoop文件管理基础命令 yarn: 资源调度相关 mapred: 程序运行, 启动历史服务器
etc/	Hadoop配置文件的目录	core-site.xml hdfs-site.xml mapred-site.xml yarn-site.xml
include/	对外提供的编程库头文件	对外提供的编程库头文件（具体动态库和静态库在lib目录中）， 这些头文件均是用C++定义的，通常用于C++程序访问HDFS或者编写MapReduce程序
lib/	动态库和静态库	该目录包含了Hadoop对外提供的编程动态库和静态库，与include目录中的头文件结合使用。
libexec/	shell配置文件	各个服务对用的shell配置文件所在的目录， 可用于配置日志输出、启动参数（比如JVM参数）等基本信息。
sbin/	Hadoop管理命令	主要包含HDFS和YARN中各类服务的启动/关闭脚本
share/	官方自带示例	Hadoop各个模块编译后的jar包所在的目录

## 6. Hadoop配置文件修改

Hadoop安装主要就是配置文件的修改，一般在 **主节点进行修改**，完毕后scp分发给其他各个 **从节点机器**。

### 6.1. hadoop-env.sh

文件中设置的是Hadoop运行时需要的环境变量。**JAVA\_HOME** 是必须设置的，即使我们当前的系统中设置了JAVA\_HOME，它也是不认识的，因为Hadoop即使是在本机上执行，它也是把当前的执行环境当成 **远程服务器**。

```
vim hadoop-env.sh

54行的JAVA_HOME的设置

export JAVA_HOME=/bigdata/server/jdk1.8

在文件末尾添加如下内容

export HDFS_NAMENODE_USER=root
export HDFS_DATANODE_USER=root
export HDFS_SECONDARYNAMENODE_USER=root
export YARN_RESOURCEMANAGER_USER=root
export YARN_NODEMANAGER_USER=root
```



## 6.2. core-site.xml

hadoop的核心配置文件，有默认的配置项 `core-default.xml`。

core-default.xml与core-site.xml的功能是一样的，如果在core-site.xml里没有配置的属性，则会自动会获取core-default.xml里的相同属性的值。

```
cd /bigdata/server/hadoop/etc/hadoop/  
vim core-site.xml
```

在文件的configuration的标签中添加以下内容:

```
<property>  
  <name>fs.defaultFS</name>  
  <value>hdfs://node01:8020</value>  
</property>  
  
<property>  
  <name>hadoop.tmp.dir</name>  
  <value>/bigdata/data/hadoop</value>  
</property>  
  
<!-- 设置HDFS web UI用户身份 -->  
<property>  
  <name>hadoop.http.staticuser.user</name>  
  <value>root</value>  
</property>  
  
<!-- 整合hive -->  
<property>  
  <name>hadoop.proxyuser.root.hosts</name>  
  <value>*</value>  
</property>  
  
<property>  
  <name>hadoop.proxyuser.root.groups</name>  
  <value>*</value>  
</property>
```

## 6.3. hdfs-site.xml

HDFS的核心配置文件，有默认的配置项 `hdfs-default.xml`。

hdfs-default.xml与hdfs-site.xml的功能是一样的，如果在hdfs-site.xml里没有配置的属性，则会自动会获取hdfs-default.xml里的相同属性的值。

```
cd /bigdata/server/hadoop/etc/hadoop/  
vim hdfs-site.xml
```

```
<!-- 指定secondarynamenode运行位置 -->
<property>
  <name>dfs.namenode.secondary.http-address</name>
  <value>node02:50090</value>
</property>
```

## 6.4. mapred-site.xml

MapReduce的核心配置文件，有默认的配置项 `mapred-default.xml`。

mapred-default.xml与mapred-site.xml的功能是一样的，如果在mapred-site.xml里没有配置的属性，则会自动会获取mapred-default.xml里的相同属性的值。

```
cd /bigdata/server/hadoop/etc/hadoop/
```

```
vim mapred-site.xml
```

```
<property>
  <name>mapreduce.framework.name</name>
  <value>yarn</value>
</property>
<property>
  <name>yarn.app.mapreduce.am.env</name>
  <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
</property>
<property>
  <name>mapreduce.map.env</name>
  <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
</property>
<property>
  <name>mapreduce.reduce.env</name>
  <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
</property>
```

## 6.5. yarn-site.xml

YARN的核心配置文件，有默认的配置项 `yarn-default.xml`。

yarn-default.xml与yarn-site.xml的功能是一样的，如果在yarn-site.xml里没有配置的属性，则会自动会获取yarn-default.xml里的相同属性的值。

```
cd /bigdata/server/hadoop/etc/hadoop/
```

```
vim yarn-default.xml
```

```
<!-- 指定YARN的主角色（ResourceManager）的地址 -->
<property>
  <name>yarn.resourcemanager.hostname</name>
  <value>node01</value>
```

```

</property>

<!-- NodeManager上运行的附属服务。需配置成mapreduce_shuffle，才可运行MapReduce程序默认
值："" -->
<property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
</property>

<!-- 是否将对容器实施物理内存限制 -->
<property>
  <name>yarn.nodemanager.pmem-check-enabled</name>
  <value>>false</value>
</property>

<!-- 是否将对容器实施虚拟内存限制。 -->
<property>
  <name>yarn.nodemanager.vmem-check-enabled</name>
  <value>>false</value>
</property>
<!-- 开启日志聚集 -->
<property>
  <name>yarn.log-aggregation-enable</name>
  <value>true</value>
</property>

<!-- 设置yarn历史服务器地址 -->
<property>
  <name>yarn.log.server.url</name>
  <value>http://node02:19888/jobhistory/logs</value>
</property>

<!-- 保存的时间7天 -->
<property>
  <name>yarn.log-aggregation.retain-seconds</name>
  <value>604800</value>
</property>

```

## 6.6. workers

workers文件里面记录的是集群主机名。主要作用是配合一键启动脚本如start-dfs.sh、stop-yarn.sh用来进行集群启动。这时候workers文件里面的主机标记的就是从节点角色所在的机器。

```
cd /bigdata/server/hadoop/etc/hadoop/
```

```
vim workers
```

```

node01
node02
node03

```

## 7. 同步hadoop软件包到node02和node03

```
scp -r hadoop-3.3.3/ node02:$PWD
```

```
scp -r hadoop-3.3.3/ node03:$PWD
```

在node02节点配置软链接

```
ln -s hadoop-3.3.3/ hadoop
```

在node03节点配置软链接

```
ln -s hadoop-3.3.3/ hadoop
```

## 8. 【所有节点】配置环境变量

```
vim /etc/profile
```

```
export HADOOP_HOME=/bigdata/server/hadoop
```

```
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
```

```
source /etc/profile
```

## 9. Hadoop集群启动

### 9.1. 启动方式

要启动Hadoop集群，需要启动 **HDFS** 和 **YARN** 两个集群。

注意：首次启动HDFS时，必须在主节点 **node01** 对其进行格式化操作。本质上是一些清理和准备工作，因为此时的HDFS在物理上还是不存在的。

```
hadoop namenode -format
```

#### 9.1.1. 手动单个节点启动

在 **主节点node01** 启动namenode

```
cd /bigdata/server/hadoop/bin
```

```
./hdfs --daemon start namenode
```

在 **node02** 启动secondarynamenode

```
cd /bigdata/server/hadoop/bin
```

```
./hdfs --daemon start secondarynamenode
```

在 **所有节点** 启动datanode

```
cd /bigdata/server/hadoop/bin  
./hdfs --daemon start datanode
```

查看进程情况

```
jpg  
netstat -ntlp  
其中hdfs的web端口: node01:9870已经可以正常访问
```

在 **主节点node01** 启动ResourceManager

```
cd /bigdata/server/hadoop/bin  
./yarn --daemon start resourcemanager
```

在 **所有节点** 启动Nodemanager

```
cd /bigdata/server/hadoop/bin  
./yarn --daemon start nodemanager
```

如果想要停止某个节点上某个角色，只需要把命令中的 **start** 改为 **stop** 即可。

### 9.1.2. 一键脚本启动

如果配置了 **etc/hadoop/workers** 和 **ssh免密登录**，则可以使用程序脚本启动所有Hadoop两个集群的相关进程，在主节点所设定的机器上执行。

**hdfs: /bigdata/server/hadoop/sbin/start-dfs.sh**

**yarn: /bigdata/server/hadoop/sbin/start-yarn.sh**

```
停止脚本  
hdfs: /bigdata/server/hadoop/sbin/stop-dfs.sh  
yarn: /bigdata/server/hadoop/sbin/stop-yarn.sh
```

完整的一键启动hdfs和yarn脚本  
start-all.sh: 启动所有的hdfs和yarn的脚本  
stop-all.sh: 停止所有的hdfs和yarn的脚本

## 10. 启动后的效果

```
1 node01 x 2 node02 x ! 3 node03 x +
[root@node01 bin]# jps
2387 NameNode
3140 Jps
2518 DataNode
2700 ResourceManager
2958 NodeManager
[root@node01 bin]#
```

```
1 node01 x 2 node02 x ! 3 node03 x +
[root@node02 bin]# jps
2352 NodeManager
2608 SecondaryNameNode
2692 Jps
2183 DataNode
[root@node02 bin]#
```

```
1 node01 x 2 node02 x 3 node03 x +
[root@node03 bin]# jps
2184 DataNode
2345 NodeManager
2527 Jps
[root@node03 bin]#
```

## 11. 集群Web访问UI


hdfs: <http://node01:9870>

The screenshot shows a web browser window with the URL `node01:9870/dfshealth.html#tab-overview`. The page title is "NameNode information". The navigation bar includes "Hadoop", "Overview", "Datanodes", "Datanode Volume Failures", "Snapshot", "Startup Progress", and "Utilities". The main content area is titled "Overview 'node01:8020' (active)". Below this is a table with the following information:

Started:	Sun Jun 19 17:52:47 +0800 2022
Version:	3.3.3, rd37586cbda38c338d9fe481addda5a05fb516f71
Compiled:	Tue May 10 00:36:00 +0800 2022 by stevel from branch-3.3.3
Cluster ID:	CID-ceae2e13-1948-4791-ae83-0f5b14ac315d
Block Pool ID:	BP-2010193171-192.168.46.150-1655632106977

yarn:http://node01:8088

IndexFiles.html 常用 线上教育 mq 搬瓦工 开源 其他 elm 教学 其他书签



Cluster

- About
- Nodes
- Node Labels
- Applications
  - NEW
  - NEW SAVING
  - SUBMITTED
  - ACCEPTED
  - RUNNING
  - FINISHED
  - FAILED
  - KILLED
- Scheduler

Tools

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	...
0	0	0	0	0	<memory:0 f

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes
0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:1024, vCores:1>

Show 20 entries

ID	User	Name	Application Type	Application Tags	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State
----	------	------	------------------	------------------	-------	----------------------	-----------	------------	------------	-------

Showing 0 to 0 of 0 entries

## 12. MapReduce JobHistory

JobHistory用来记录已经finished的mapreduce运行日志，日志信息存放于HDFS目录中，默认情况下没有开启此功能，需要在 `mapred-site.xml` 中配置并手动启动。

### 12.1. 修改mapred-site.xml

```
cd /bigdata/server/hadoop/etc/hadoop/
```

```
vim mapred-site.xml
```

```
<property>
  <name>mapreduce.jobhistory.address</name>
  <value>node02:10020</value>
</property>
<property>
  <name>mapreduce.jobhistory.webapp.address</name>
  <value>node02:19888</value>
</property>
```

```
scp mapred-site.xml node02:$PWD
```

```
scp mapred-site.xml node03:$PWD
```

### 12.2. 在node02节点启动JobHistory

```
cd /bigdata/server/hadoop/bin
```

```
./mapred --daemon start historyserver
```

## 12.3. 访问web管理界面

`http://node02:19888/jobhistory`

## 13. 运行演示程序

在hdfs创建一个目录:

```
hdfs dfs -mkdir /input
```

上传文件到hdfs的/input目录

```
hdfs dfs -put start-all.sh /input
```

运行示例程序

```
hadoop jar /bigdata/server/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.3.jar  
wordcount /input /output
```

### Browse Directory

Show  entries

Search:

<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	
<input type="checkbox"/>	-rw-r--r--	root	supergroup	2.17 KB	Jun 19 18:47	3	128 MB	start-all.sh	

Showing 1 to 1 of 1 entries

Browsing HDFS

All Applications

JobHistory

node01:8088/cluster/apps

IndexFiles.html 常用 线上教育 mq 搬瓦工 开源 其他 elm 教学 其他书签

Cluster

About

Nodes

Node Labels

Applications

NEW

NEW SAVING

SUBMITTED

ACCEPTED

RUNNING

FINISHED

FAILED

KILLED

Scheduler

Tools

Cluster Metrics

Apps Submitted		Apps Pending		Apps Running		Apps Completed		Containers Running	
1		0		0		1		0	

Cluster Nodes Metrics

Active Nodes		Decommissioning Nodes		Decommissioned	
3		0		0	

Scheduler Metrics

Scheduler Type		Scheduling Resource Type		Minimum Allo	
Capacity Scheduler		[memory-mb (unit=Mi), vcores]		<memory:1024, vCores:1>	

Show  entries

ID	User	Name	Application Type	Application Tags	Queue	Application Priority	StartTime	LaunchTime
application_1655635432951_0002	root	word count	MAPREDUCE		default	0	Sun Jun 19 18:48:19 +0800 2022	Sun Jun 19 18:48:20 +0800 2022

Showing 1 to 1 of 1 entries





# JobHistory

Application

AboutJobs

Tools

Retired Jobs

Show20entries

Sea

Submit Time	Start Time	Finish Time	Job ID	Name	User	Queue	State	Maps Total	Maps Completed	Reduces Total
2022.06.19 18:48:19 CST	2022.06.19 18:48:27 CST	2022.06.19 18:48:41 CST	job_1655635432951_0002	word count	root	default	SUCCEEDED	1	1	1

Submit Tir

Start Time

Finish Tim

Job ID

Name

User

Queue

State

Maps 1

Maps Comple

Reduces 1

Showing 1 to 1 of 1 entries

FirstPrev

# 1. jdk安装

## 1. 创建软件安装的目录

```
mkdir -p /bigdata/{soft,server}

/bigdata/soft  安装文件的存放目录

/bigdata/server 软件安装的目录
```

## 2. 把安装的软件上传到/bigdata/soft 目录

```
[root@node01 soft]# pwd
/bigdata/soft
[root@node01 soft]# ls
jdk-8u241-linux-x64.tar.gz
[root@node01 soft]#
```

## 3. 解压到指定目录

```
-C :指定解压到指定目录

tar -zxvf /bigdata/soft/jdk-8u241-linux-x64.tar.gz -C /bigdata/server/
```

## 4. 创建一个软链接

```
cd /bigdata/server

ln -s jdk1.8.0_241/ jdk1.8
```

## 5. 配置环境变量

```
vi /etc/profile

export JAVA_HOME=/bigdata/server/jdk1.8
export PATH=$JAVA_HOME/bin:$PATH

重新加载配置文件

source /etc/profile
```

## 6. 测试验证

```
[root@node01 server]# java -version
java version "1.8.0_241"
Java(TM) SE Runtime Environment (build 1.8.0_241-b07)
Java HotSpot(TM) 64-Bit Server VM (build 25.241-b07, mixed mode)
```



## 1. 卸载已经安装的MySQL数据库

```
# 查询MySQL相关的依赖
rpm -qa |grep mysql
# 如果存在, 则通过rpm -e --nodeps 进行卸载
```

## 2. 获取rpm在线安装仓库文件

```
wget https://dev.mysql.com/get/mysql80-community-release-el7-6.noarch.rpm
```

## 3. 安装mysql的仓库文件

```
rpm -ivh mysql80-community-release-el7-6.noarch.rpm
```

## 4. 修改mysql仓库的配置文件

```
cd /etc/yum.repos.d/
mysql-community.repo: 用于指定下载哪个版本的安装包
mysql-community-source.repo: 用于指定下载哪个版本的源码

`禁用8.0的版本, 启用5.7的版本`
```

## 5. 安装MySQL5.7

```
# 导入签名的信息key
rpm --import https://repo.mysql.com/RPM-GPG-KEY-mysql-2022
# 安装5.7
yum install -y mysql-community-server
```

## 6. 启动数据库

```
systemctl start mysqld
systemctl status mysqld
systemctl enable mysqld
```

## 7. 登录数据库

```
# 查看初始密码
less /var/log/mysqld.log |grep pass

# 登录数据库
mysql -uroot -p'XRY0460efV<7'
```

## 8. 修改MySQL数据库密码策略

```
set global validate_password_length=4;  
set global validate_password_policy=0;
```

## 9. 创建远程登录用户

```
ALTER USER 'root'@'localhost' IDENTIFIED BY '123456';  
#创建一个远程登录用户  
create user 'root'@'%' identified by '123456';  
# 设置远程登录权限  
grant all privileges on *.* to 'root'@'%';
```

## 10. 设置服务器编码为utf8

```
vi /etc/my.cnf  
# 在mysqld下面设置  
character_set_server=utf8  
  
# 重启服务  
systemctl restart mysqld
```

# 1. IP地址设置

- 修改ip地址

```
vi /etc/sysconfig/network-scripts/ifcfg-ens33
```

```
TYPE="Ethernet"
PROXY_METHOD="none"
BROWSER_ONLY="no"
BOOTPROTO="static" # 设置为静态ip static
DEFROUTE="yes"
IPV4_FAILURE_FATAL="no"
NAME="ens33" # 网卡的名称
DEVICE="ens33" # 设备的名称
ONBOOT="yes" # 设置为yes, 表示开机启动
IPADDR="192.168.46.150" # ip地址
PREFIX="24" # 子网掩码
GATEWAY="192.168.46.2" # 网关
DNS1="114.114.115.115" # DNS
```

- 重启网络服务

```
systemctl restart network
```

- 测试网络

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

如果可以ping通公网: 说明 ip地址和网关都配置正确

如果通过 ip addr 不能查看到ip地址, 说明配置有错误

如果可以ping通内网 192.168.46.1 但是不能ping通外网的话, 则说明网关配置有错误

# 2. 设置主机名

- 编辑主机名配置文件

```
vi /etc/hostname
```

```
hadoop01
```

# 3. 设置域名映射解析

- 编辑hosts文件

```
vi /etc/hosts
```

```
192.168.46.145 hadoop01 hadoop01
192.168.46.146 hadoop02 hadoop02
192.168.46.147 hadoop03 hadoop03
192.168.46.148 biz01 biz01
```

## 4. 关闭防火墙和Selinux

- 关闭防火墙

```
systemctl stop firewalld.service
systemctl disable firewalld.service
```

- 关闭Selinux

```
vi /etc/selinux/config

# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=disabled
# SELINUXTYPE= can take one of three values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are
protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

## 5. 配置免密登录

1. 在所有节点生成公钥和私钥

```
ssh-keygen -t rsa
```

后面直接所有的交互都敲回车 即可

2. 拷贝公钥到每台服务器

```
ssh-copy-id hadoop01
ssh-copy-id hadoop02
ssh-copy-id hadoop03
```

3. 验证ssh登录

```
ssh hadoop01
exit # 退出ssh登录
```

## 6. 配置服务器节点时钟同步

1. 在所有节点安装ntpd

```
yum install -y ntpdate
```

## 2. 增加定时任务

```
crontab -e
```

```
*/1 * * * * /usr/sbin/ntpdate -u ntp4.aliyun.com > /dev/null 2>&1
```

# 7. 安装常用软件

```
yum install -y vim  
yum install -y net-tools  
yum install -y lrzsz  
yum install -y rsync  
yum install -y wget
```



# 1. 安装Redis数据库

## 1. 下载redis

```
wget https://download.redis.io/redis-stable.tar.gz
```

## 2. 解压到指定目录

```
tar -zxvf redis-stable.tar.gz
```

## 3. 安装一些gcc编译库

```
yum install -y gcc g++ gcc-c++ make
```

## 4. 编译并且安装

```
进入到redis的源码目录  
make MALLOC=libc  
make PREFIX=/bigdata/server/redis install
```

## 5. 拷贝配置文件redis.conf并且修改

```
cd /bigdata/server/redis  
mkdir {conf,data} # conf 配置文件目录 data 数据存放目录 log日志文件目录  
  
cd /bigdata/soft/redis-stable  
cp redis.conf /bigdata/server/redis/conf/
```

```
bind 0.0.0.0 # 配置可以所有的地址都可以访问redis  
protected-mode no # 关闭保护模式  
daemonize yes # 后台启动运行  
dir ./data # 相关的数据和日志文件的存放目录  
dbfilename dump.rdb # 数据文件存放  
logfile "redis.log" # 指定logfile的文件名 默认没有日志文件
```

## 6. 启动和关闭redis

```
# 启动redis
bin/redis-server conf/redis.conf

# 关闭redis
bin/redis-cli shutdown

# 登录到redis
bin/redis-cli
```

```
[root@localhost redis]# netstat -ntlp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:6379             0.0.0.0:*               LISTEN      10151/bin/redis-ser
tcp        0      0 0.0.0.0:22               0.0.0.0:*               LISTEN      1014/sshd
tcp        0      0 127.0.0.1:25              0.0.0.0:*               LISTEN      1096/master
tcp6       0      0 :::22                    :::*                     LISTEN      1014/sshd
tcp6       0      0 :::1:25                   :::*                     LISTEN      1096/master
```



## 2. 配置系统环境

1. 导入redis的数据库备份dump文件
2. 修改启动web应用的application.properties配置文件

```
# 生成日志的存放路径
logPath=d:/log/behavior
# redis数据库的连接地址 端口 密码
spring.redis.host=localhost
spring.redis.port=6379
spring.redis.password=
# web应用服务器的启动端口
server.port=8080
```

## 3. 启动日志系统

需要可执行的jar包和配置文件在同一目录

名称	修改日期
 application.properties	2022/8/25 16:17
 behavior-0.0.1-SNAPSHOT.jar	2022/8/25 16:19

访问接口,生成日志














```
# date: 具体生成哪一天的数据 count: 具体生成的数据量
http://localhost:8080/create/log?date=2030-09-09&count=1000
```

注意: 为了上课统一, 这里生成2022-08-08 至 2022-08-14 共7天的数据

此电脑 > 本地磁盘 (D:) > log > behavior

名称

^

-   behavior2022-08-08.log
-   behavior2022-08-09.log
-   behavior2022-08-10.log
-   behavior2022-08-11.log
-   behavior2022-08-12.log
-   behavior2022-08-13.log
-  behavior2022-08-14.log

# 1. 安装数据采集软件Flume

前提条件：

- 业务系统需要有hadoop的客户端

## 1.1. 安装hadoop集群客户端

直接从hadoop01节点通过scp拷贝客户端到biz01

```
# 在biz01上执行
cd /bigdata/server
scp root@hadoop01:/bigdata/server/hadoop-3.3.3 .
# 创建软链接
ln -s hadoop-3.3.3 hadoop

# 设置好主机名
vi /etc/hosts

192.168.113.145 hadoop01 hadoop01
192.168.113.146 hadoop02 hadoop02
192.168.113.147 hadoop03 hadoop03
192.168.113.148 biz01 biz01
```

## 1.2. 安装flume数据采集软件

在biz01上安装flume数据采集软件

```
# 1 上传apache-flume-1.10.1-bin.tar.gz 到 /bigdata/soft 目录
# 2 解压到指定目录
tar -zxvf apache-flume-1.10.1-bin.tar.gz -C /bigdata/server/
# 3 创建软链接
cd /bigdata/server
ln -s apache-flume-1.10.1-bin/ flume
```

## 1.3. 配置环境变量

```
vi /etc/profile.d/my_env.sh

#!/bin/bash
#JAVA_HOME
export JAVA_HOME=/bigdata/server/jdk1.8
export PATH=$PATH:$JAVA_HOME/bin

#FLUME_HOME
export FLUME_HOME=/bigdata/server/flume
export PATH=$PATH:$FLUME_HOME/bin
```

```
#HADOOP_HOME
export HADOOP_HOME=/bigdata/server/hadoop
export PATH=$PATH:$HADOOP_HOME/bin

# 加载配置文件
source /etc/profile
```

## 1.4. 测试环境

```
# 测试hadoop环境
hdfs dfs -ls /
```

## 1.5. 配置Flume采集数据

### 1.5.1. 在lib目录添加一个ETL拦截器

- 处理标准的json格式的数据, 如果格式不符合条件, 则会过滤掉该信息
- 处理时间漂移的问题, 把对应的日志存放到具体的分区数据中

在业务服务器的Flume的lib目录添加iterceptor-etl.jar

### 1.5.2. 配置采集数据到hdfs文件的配置

在flume的家目录创建文件 jobs/log\_file\_to\_hdfs.conf

```
#为各组件命名
a1.sources = r1
a1.channels = c1
a1.sinks = k1

#描述source
a1.sources.r1.type = TAILDIR
a1.sources.r1.filegroups = f1
a1.sources.r1.filegroups.f1 = /bigdata/soft/app/log/behavior/*.
a1.sources.r1.positionFile =
/bigdata/server/flume/position/behavior/taildir_position.json
a1.sources.r1.interceptors = i1
a1.sources.r1.interceptors.i1.type =
cn.wolfcode.flume.interceptor.ETLInterceptor$Builder
a1.sources.r1.interceptors = i2
a1.sources.r1.interceptors.i2.type =
cn.wolfcode.flume.interceptor.TimeStampInterceptor$Builder

## channel1
a1.channels.c1.type = memory
a1.channels.c1.capacity = 1000
a1.channels.c1.transactionCapacity = 100

## sink1
a1.sinks.k1.type = hdfs
a1.sinks.k1.hdfs.path = /behavior/origin/log/%Y-%m-%d
a1.sinks.k1.hdfs.filePrefix = log-
```

```

a1.sinks.k1.hdfs.round = false
a1.sinks.k1.hdfs.rollInterval = 10
a1.sinks.k1.hdfs.rollSize = 134217728
a1.sinks.k1.hdfs.rollCount = 0

## 控制输出文件是原生文件。
a1.sinks.k1.hdfs.fileType = DataStream

## 拼装
a1.sources.r1.channels = c1
a1.sinks.k1.channel=c1

```

### 1.5.3. 运行数据采集命令

```

# 进入到Flume的目录
cd /bigdata/server/flume

```

```

bin/flume-ng agent --conf conf/ --name a1 --conf-file jobs/log_file_to_hdfs.conf -
Dflume.root.logger=INFO,console

```

```

# 后台启动运行
nohup bin/flume-ng agent --conf conf/ --name a1 --conf-file jobs/log_file_to_hdfs.conf
-Dflume.root.logger=INFO,console >/bigdata/server/flume/logs/log_file_to_hdfs.log 2>&1
&

```

### 1.5.4. 日志采集效果

#### Browse Directory

Go!

Show 25 entries
Search:

<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	
<input type="checkbox"/>	<a href="#">drwxr-xr-x</a>	<a href="#">root</a>	<a href="#">supergroup</a>	0 B	Aug 29 10:12	<a href="#">0</a>	0 B	<a href="#">2022-08-08</a>	
<input type="checkbox"/>	<a href="#">drwxr-xr-x</a>	<a href="#">root</a>	<a href="#">supergroup</a>	0 B	Aug 29 10:12	<a href="#">0</a>	0 B	<a href="#">2022-08-09</a>	
<input type="checkbox"/>	<a href="#">drwxr-xr-x</a>	<a href="#">root</a>	<a href="#">supergroup</a>	0 B	Aug 29 10:12	<a href="#">0</a>	0 B	<a href="#">2022-08-10</a>	
<input type="checkbox"/>	<a href="#">drwxr-xr-x</a>	<a href="#">root</a>	<a href="#">supergroup</a>	0 B	Aug 29 10:12	<a href="#">0</a>	0 B	<a href="#">2022-08-11</a>	
<input type="checkbox"/>	<a href="#">drwxr-xr-x</a>	<a href="#">root</a>	<a href="#">supergroup</a>	0 B	Aug 29 10:12	<a href="#">0</a>	0 B	<a href="#">2022-08-12</a>	
<input type="checkbox"/>	<a href="#">drwxr-xr-x</a>	<a href="#">root</a>	<a href="#">supergroup</a>	0 B	Aug 29 10:12	<a href="#">0</a>	0 B	<a href="#">2022-08-13</a>	
<input type="checkbox"/>	<a href="#">drwxr-xr-x</a>	<a href="#">root</a>	<a href="#">supergroup</a>	0 B	Aug 29 10:12	<a href="#">0</a>	0 B	<a href="#">2022-08-14</a>	

# 1. OLTP和OLAP

On-Line Transaction Processing联机事务处理过程(OLTP)，也称为面向交易的处理过程，其基本特征是前台接收的用户数据可以立即传送到计算中心进行处理，并在很短的时间内给出处理结果，是对用户操作快速响应的方式之一。

具有较强的 数据一致性和 事务操作

联机分析处理OLAP是一种软件技术，它使分析人员能够迅速、一致、交互地从各个方面观察信息，以达到深入理解数据的目的。它具有FASMI(Fast Analysis of Shared Multidimensional Information)，即共享多维信息的快速分析的特征。其中F是快速性(Fast)，指系统能在数秒内对用户的多数分析要求做出反应；A是可分析性(Analysis)，指用户无需编程就可以定义新的专门计算，将其作为分析的一部分，并以用户所希望的方式给出报告；M是多维性(Multi-dimensional)，指提供对数据分析的多维视图和分析；I是信息性(Information)，指能及时获得信息，并且管理大容量信息。主要是一个 数据分析系统，要求有 较快的时间响应

对于我们常用的关系型数据库，对于数据一致性要求比较高，基本都是我们的OLTP系统

对于我们常见的数据分析系统，主要是根据已有的业务数据进行统计分析，比如管理驾驶舱 数据统计分析，比如做BI报表，做机器学习等，这些我们会专门在一个数据分析系统OLAP系统进行统计分析

问题：为什么不在业务系统做数据分析呢

- 1 数据分析会影响业务系统的效率，降低业务系统的处理性能
- 2 分析系统的数据有可能来自多个业务系统和业务日志文件，比如来自财务系统，客户管理系统，订单管理系统，商城系统等
- 3 分析系统的设计模型和业务系统的设计模型是不一致的，针对于分析系统，我们会使用维度建模，对于业务系统，我们会使用ER建模

## 2. 什么是数据库建模

在设计数据库时，对现实世界进行分析、抽象、并从中找出内在联系，进而确定数据库的结构，这一过程就称为数据库建模

常见的数据库建模有 关系建模 和 维度建模

### 2.1. 关系建模

关系建模将复杂的数据抽象为两个概念——实体和关系，并使用规范化的方式表示出来。关系模型如图所示，从图中可以看出，较为松散、零碎，物理表数量多。

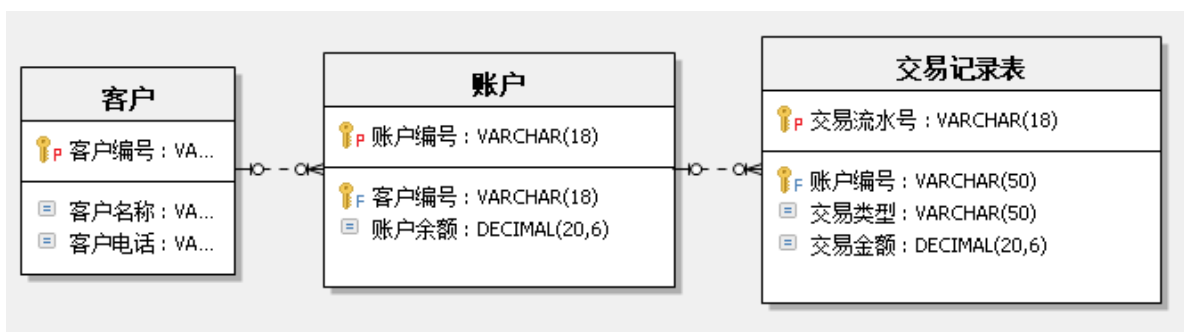
关系模型严格遵循第三范式（3NF），数据冗余程度低，数据的一致性容易得到保证。由于数据分布于众多的表中，查询会相对复杂，在大数据的场景下，查询效率相对较低。

关系建模是通过准确的 业务规则 来描述业务运作的过程

业务规则:

- 1 一个客户拥有多个银行账户

- 2 一个银行账户只属于一个客户
- 3 一个银行账户会有多个银行业务交易
- 4 一个银行业务交易记录只属于一个银行账户



## 2.2. 维度建模

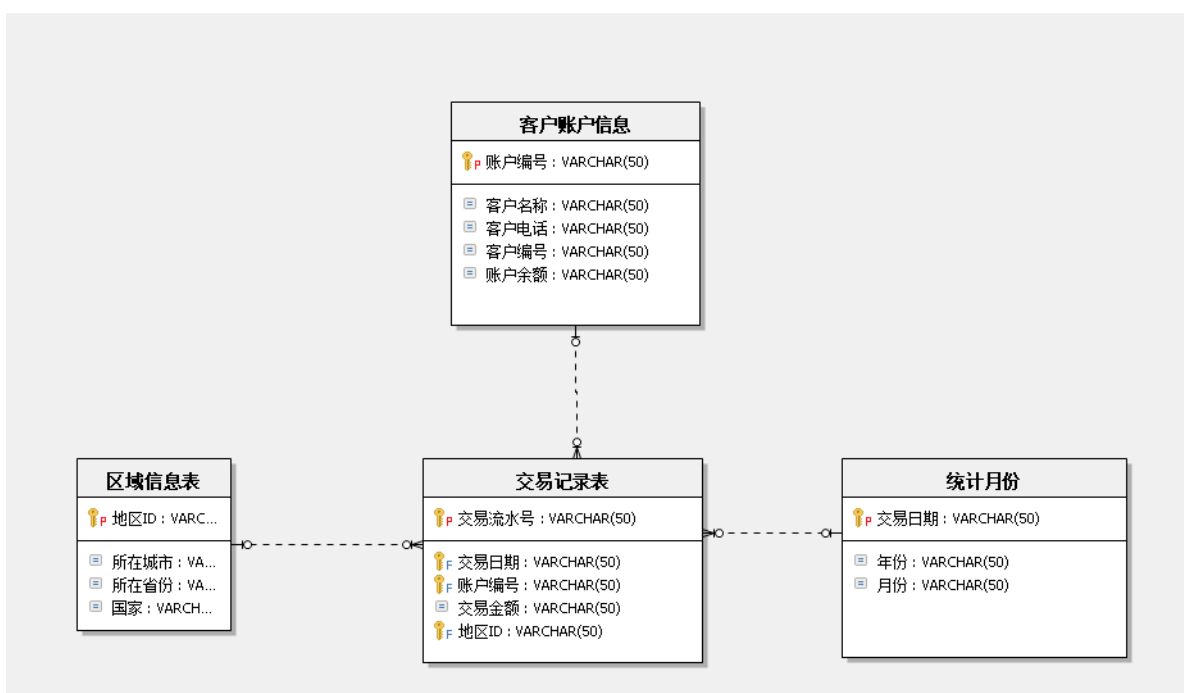
维度模型以数据分析作为出发点，不遵循三范式，故数据存在一定的冗余。维度模型面向业务，将业务用事实表和维度表呈现出来。表结构简单，故查询简单，查询效率较高。

维度建模是面向查询主题分析

“维度建模是根据不同的维度进行统计分析的维度模型”

统计需求分析:

- 1 围绕账户交易记录表进行统计
- 2 根据时间(月份, 季度, 年份)进行时间维度的统计
- 3 根据地区(城市, 省份, 国家)进行对应的地区维度进行统计
- 4 根据账户, 客户的信息进行对应的



## 3. 维度建模



在OLAP的数据仓库的设计中, 为了方便我们的查询效率, 通常采用的是维度建模, 在维度建模的设计中, 我们使用的最多的

### 3.1. 事实表

---

事实表是指存储有事实记录的表, 如系统日志、销售记录等; 事实表的记录在不断地动态增长, 所以它的体积通常远大于其他表。

事实表作为数据仓库建模的核心, 需要根据业务过程来设计, 包含了引用的维度和业务过程有关的度量。

### 3.2. 维度表

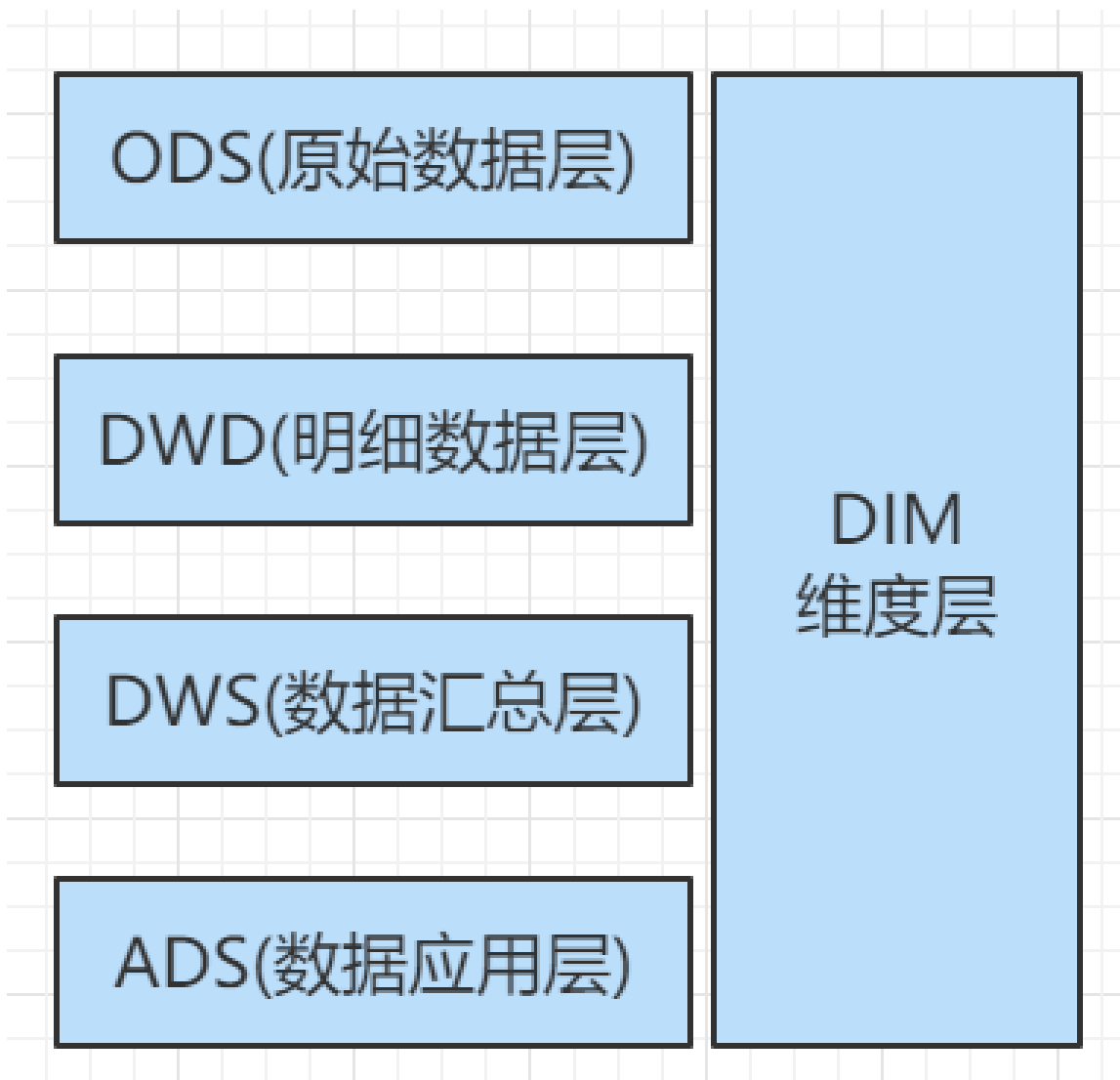
---

维度表或维表, 有时也称查找表, 是与事实表相对应的一种表; 它保存了维度的属性值, 可以跟事实表做关联; 相当于将事实表上经常重复出现的属性抽取、规范出来用一张表进行管理。常见的维度表有: **日期表** (存储与日期对应的周、月、季度等的属性)、**地点表** (包含国家、省 / 州、城市等属性) 等。 **维度是维度建模的基础和灵魂** ,

## 4. 数据仓库分层

### 4.1. 数仓分层结构

---



ODS层: 存放业务系统采集过来的原始数据, 直接加载的业务数据, 不做处理

DWD层: 对于ODS层的数据做基本的处理, 并且进行业务事实的分析和定位(不合法的数据处理, 空值的处理), 一行数据代表的是一个业务行为

DWS层: 对于DWD层的业务数据进行按天或者按照一定的周期进行统计分析, 是一个轻度聚合的结果

DIM层, 维度统计层, 对于需要统计分析(group by)的相关的条件进行统一的设计和规范, 比如时间, 地区, 用户等

ADS(数据应用层): 需要的业务统计分析结果, 一般会把ADS层的数据抽取到业务数据库MySQL中

## 4.2. 为什么需要对数据仓库分层

1. 把复杂问题简单化 不同的层次负责不同的功能定位
2. 减少重复开发 对于DIM, DWS可以根据主题进行自上而下的设计, 抽取通用的功能
3. 隔离原始数据 ODS层原始数据, 可以对统计结果进行回溯, 方便问题的定位

# 1. 数据可视化

## 1.1. Datalog的基本安装

1. 下载软件

<https://datalog-opensource.oss-cn-hangzhou.aliyuncs.com/20220530/datalog.tar.gz>

2. 上传的服务器的指定目录(hadoop02)
3. 解压到指定目录

```
tar -zxvf datalog.tar.gz -C /bigdata/server/
```

4. 运行示例程序

```
python bin/datalog.py job/job.json
```

## 1.2. Datalog的基本使用

## 1.3. 在MySQL中创建对应的表结构

用户城市分布

```
CREATE TABLE `ads_user_city` (  
  city varchar(80) DEFAULT NULL COMMENT '城市',  
  province varchar(80) DEFAULT NULL COMMENT '省份',  
  area varchar(80) DEFAULT NULL COMMENT '区域',  
  dt varchar(80) DEFAULT NULL COMMENT '日期',  
  count bigint DEFAULT NULL COMMENT '统计数量'  
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='用户城市分布'
```

网站访问的上网模式分布

```
create table ads_visit_type(  
  url VARCHAR(80) COMMENT '访问地址',  
  type VARCHAR(80) COMMENT '访问模式',  
  dt VARCHAR(80) COMMENT '日期',  
  month VARCHAR(80) COMMENT '月度',  
  quarter VARCHAR(80) COMMENT '季度',  
  count bigint COMMENT '统计数量'  
) COMMENT '网站访问的上网模式分布'
```

## 1.4. 导出数据脚本

- ads\_user\_city.json

```
{
```

```
"job": {
  "setting": {
    "speed": {
      "channel": 1
    }
  },
  "content": [
    {
      "reader": {
        "name": "hdfsreader",
        "parameter": {
          "path": "/behavior/ads/ads_user_city/*",
          "defaultFS": "hdfs://hadoop01:8020",
          "column": [
            {
              "index": 0,
              "type": "string"
            },
            {
              "index": 1,
              "type": "string"
            },
            {
              "index": 2,
              "type": "string"
            },
            {
              "index": 3,
              "type": "string"
            },
            {
              "index": 4,
              "type": "long"
            }
          ],
          "fileType": "text",
          "encoding": "UTF-8",
          "fieldDelimiter": "\t"
        }
      },
      "writer": {
        "name": "mysqlwriter",
        "parameter": {
          "writeMode": "insert",
          "username": "root",
          "password": "wolfcode2048",
          "column": [
            "city",
            "province",
            "area",
            "dt",
            "count"
          ],
          "session": [
            "set session sql_mode='ANSI'"
          ],
          "preSql": [
```



```
        "type": "long"
    },
    ],
    "fileType": "text",
    "encoding": "UTF-8",
    "fieldDelimiter": "\t"
}

},
"writer": {
    "name": "mysqlwriter",
    "parameter": {
        "writeMode": "replace",
        "username": "root",
        "password": "wolfcode2048",
        "column": [
            "url",
            "type",
            "dt",
            "month",
            "quarter",
            "count"
        ],
        "session": [
            "set session sql_mode='ANSI'"
        ],
        "preSql": [
            "delete from ads_visit_type"
        ],
        "connection": [
            {
                "jdbcUrl":
"jdbc:mysql://192.168.113.144:3306/behavior?useUnicode=true&characterEncoding=utf-8&useSSL=false",
                "table": [
                    "ads_visit_type"
                ]
            }
        ]
    }
}
}
```

## 2. Quick BI的可视化报表配置(参考官方文档即可)

[https://www.aliyun.com/product/bigdata/bi?spm=5176.19720258.J\\_3207526240.31.e93976f4NZzkjx](https://www.aliyun.com/product/bigdata/bi?spm=5176.19720258.J_3207526240.31.e93976f4NZzkjx)

