# 离线数仓建设

### 1.1 集群命令批量执行脚本

1）在/home/test1/bin目录下创建脚本xcall

[test1@master bin]$ vim xcall

2）在脚本中编写如下内容

#! /bin/bash

for i in master slave1 slave2

do

echo --------- $i ----------

ssh $i "$\*"

done

3）修改脚本执行权限

[test1@master bin]$ chmod 777 xcall

4）启动脚本

[test1@master bin]$ xcall jps

### 1.2 Hadoop安装

**1）安装步骤**

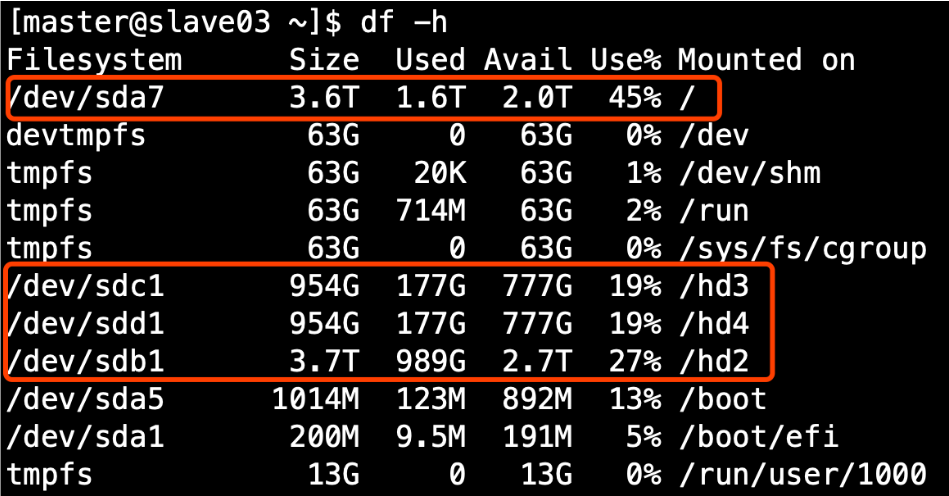
详见：Hadoop安装.docx\



**2）项目经验**

（1）项目经验之HDFS存储多目录

① 生产环境服务器磁盘情况



② 在hdfs-site.xml文件中配置多目录，注意新挂载磁盘的访问权限问题。

HDFS的DataNode节点保存数据的路径由dfs.datanode.data.dir参数决定，其默认值为file://${hadoop.tmp.dir}/dfs/data，若服务器有多个磁盘，必须对该参数进行修改。如服务器磁盘如上图所示，则该参数应修改为如下的值。

<property>

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

<value>file:///dfs/data1,file:///hd2/dfs/data2,file:///hd3/dfs/data3,file:///hd4/dfs/data4</value>

</property>

注意：每台服务器挂载的磁盘不一样，所以每个节点的多目录配置可以不一致。单独配置即可。

（2）项目经验之集群数据均衡

① 节点间数据均衡

开启数据均衡命令：

start-balancer.sh -threshold 10

对于参数10，代表的是集群中各个节点的磁盘空间利用率相差不超过10%，可根据实际情况进行调整。

停止数据均衡命令：

stop-balancer.sh

② 磁盘间数据均衡

生成均衡计划**（我们只有一块磁盘，不会生成计划）**

hdfs diskbalancer -plan slave1

执行均衡计划。

hdfs diskbalancer -execute slave1.plan.json

查看当前均衡任务的执行情况。

hdfs diskbalancer -query slave1

取消均衡任务。

hdfs diskbalancer -cancel slave1.plan.json

**（3）项目经验之Hadoop参数调优**

HDFS参数调优hdfs-site.xmls. If dfs.namenode.servicerpc-address is not configured then Namenode RPC server threads listen to requests from all nodes.

NameNode有一个工作线程池，用来处理不同DataNode的并发心跳以及客户端并发的元数据操作。

对于大集群或者有大量客户端的集群来说，通常需要增大参数dfs.namenode.handler.count的默认值10。

<property>

<name>dfs.namenode.handler.count</name>

<value>10</value>

</property>

dfs.namenode.handler.count=，比如集群规模为8台时，此参数设置为41。可通过简单的python代码计算该值，代码如下。

[test1@master ~]$ python

Python 2.7.5 (default, Apr 11 2018, 07:36:10)

[GCC 4.8.5 20150623 (Red Hat 4.8.5-28)] on linux2

Type "help", "copyright", "credits" or "license" for more information.

>>> import math

>>> print int(20\*math.log(8))

41

>>> quit()

YARN参数调优yarn-site.xml

**情景描述**：总共7台机器，每天几亿条数据，数据源->Flume->Kafka->HDFS->Hive

**面临问题**：数据统计主要用HiveSQL，没有数据倾斜，小文件已经做了合并处理，开启的JVM重用，而且IO没有阻塞，内存用了不到50%。但是还是跑的非常慢，而且数据量洪峰过来时，整个集群都会宕掉。基于这种情况有没有优化方案。

**解决办法：**

内存利用率不够。这个一般是Yarn的2个配置造成的，单个任务可以申请的最大内存大小，和Hadoop单个节点可用内存大小。调节这两个参数能提高系统内存的利用率。

（a）yarn.nodemanager.resource.memory-mb

表示该节点上YARN可使用的物理内存总量，默认是8192（MB），注意，如果你的节点内存资源不够8GB，则需要调减小这个值，而YARN不会智能的探测节点的物理内存总量。

（b）yarn.scheduler.maximum-allocation-mb

单个任务可申请的最多物理内存量，默认是8192（MB）。

### 1.3 Zookeeper安装

**1）安装步骤**

详见：Zookeeper安装



### 1.4 Kafka安装

**1）安装步骤**

详见：Kafka安装



### 1.5 kafka-ui 安装

**1）安装步骤**

详见：docker安装kafka-ui



### 1.6 flink 安装

**1）安装步骤**

详见：flink安装



### 1.7 doris 安装

**1）安装步骤**

详见：doris安装



### 1.8 dolphinscheduler安装

**1）安装步骤**

详见：dolphinscheduler安装



### 1.9 seatunnel安装

**1）安装步骤**

1. 上传seatunnel 安装包到 master 节点的/opt/software 目录
2. 解压安装包到当前目录

[test1@master software]$ tar -zxvf apache-seatunnel-incubating-2.3.0-bin.tar.gz

1. 进入到解压目录下执行脚本即可

[test1@master software]$ sh bin/install\_plugin.sh

1. 等待下载结束就好了

# 采集通道建设

### 2.1 ods建设

### 使用Doris目录下的ddl目录sql语句在doris web ui创建目的表

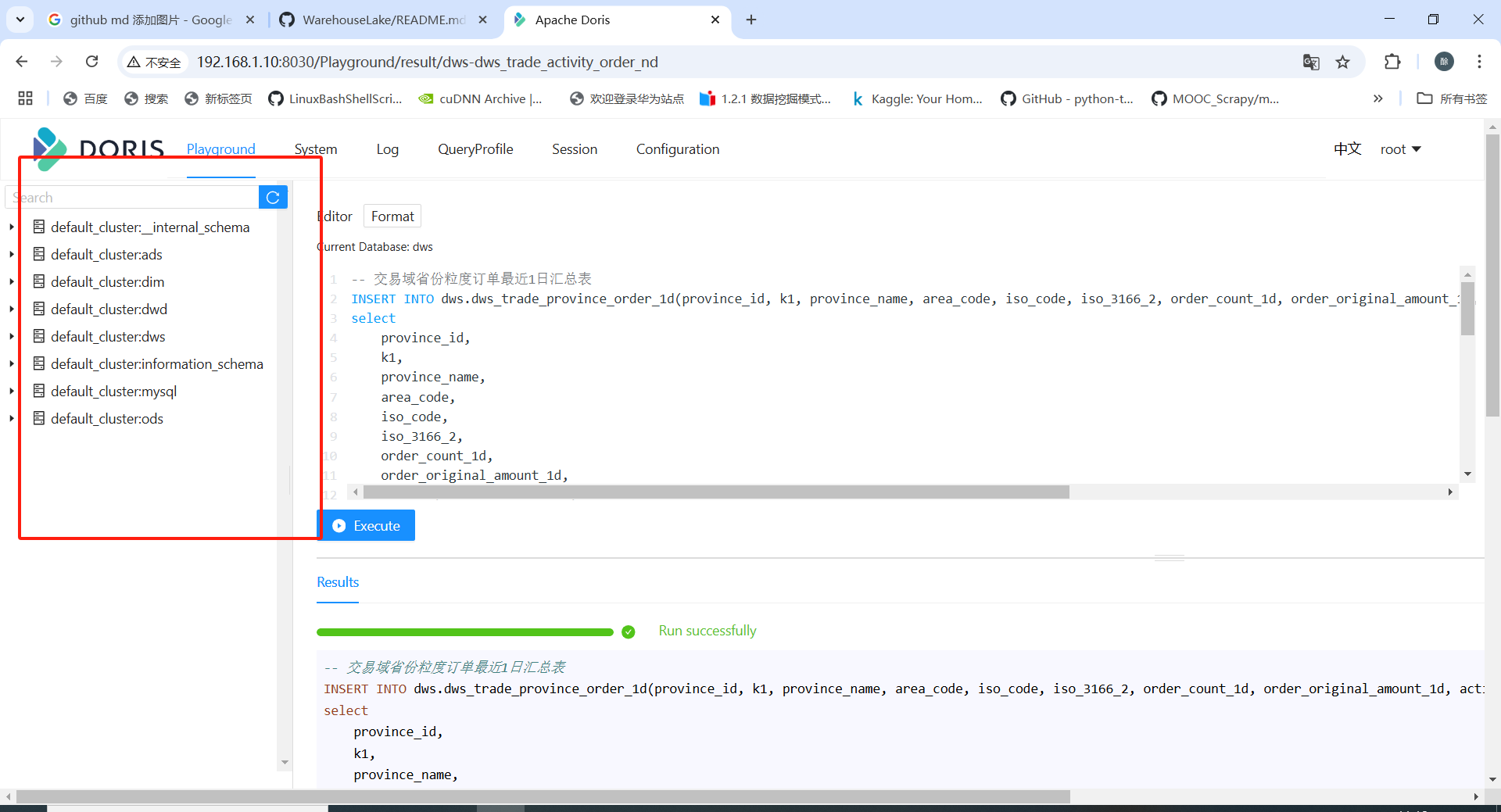
create database ods;

create database dwd;

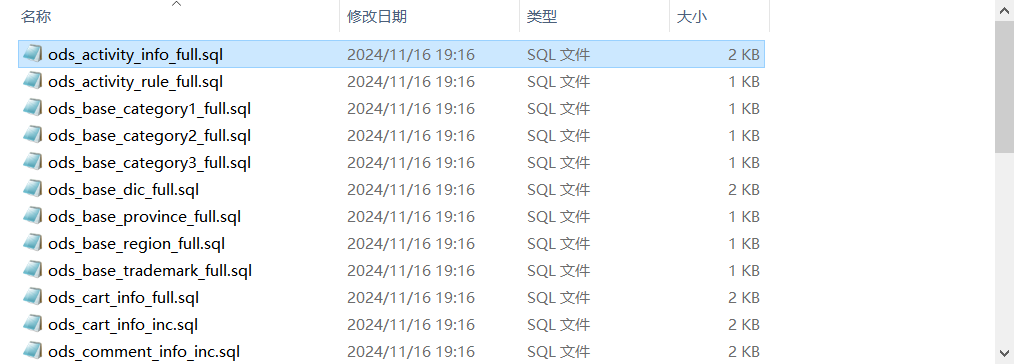
create database dim;

create database dws;

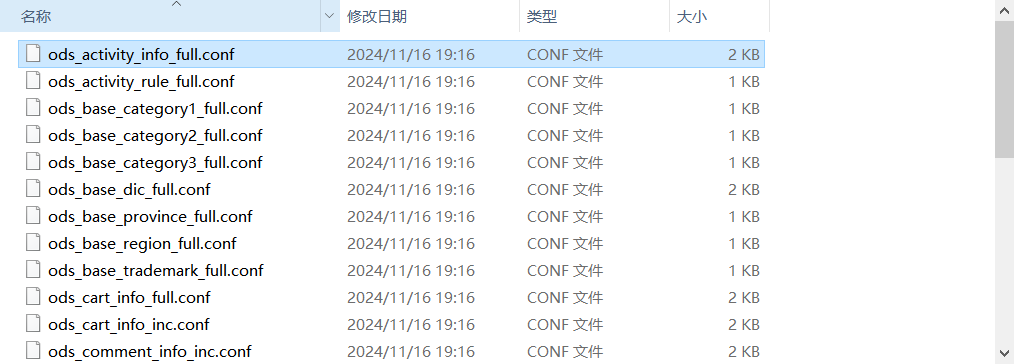
create database ads;



### 2.1.1 逐一运行这里面的sql语句



### 2.1.2 使用seatunnel配置文件将数据采集到doris上



# 同步MySQL活动规则表（全量表）到Doris ods层

env {

execution.parallelism = 2

job.mode = "BATCH"

checkpoint.interval = 10000

}

#自己更改mysql相关信息

source {

Jdbc {

url = "jdbc:mysql://slave2:3306/gmall?serverTimezone=GMT%2b8"

driver = "com.mysql.jdbc.Driver"

connection\_check\_timeout\_sec = 100

user = "root"

password = "root"

query = "select id, activity\_id, activity\_type, condition\_amount, condition\_num, benefit\_amount, benefit\_discount, benefit\_level from gmall.activity\_rule"

}

}

sink {

Doris {

fenodes = "192.168.1.10:8030"

username = "root"

password = ""

table.identifier = "ods.ods\_activity\_rule\_full"

sink.enable-2pc = "true"

sink.label-prefix = "test\_json"

doris.config = {

format="json"

read\_json\_by\_line="true"

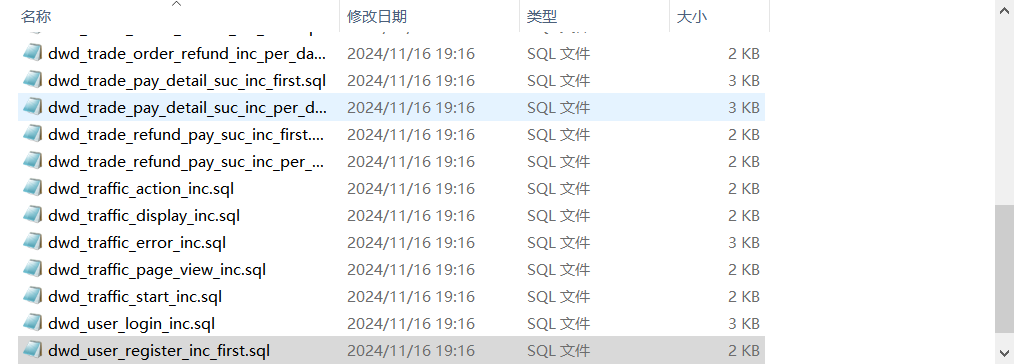
}

}

}

### 2.1 dwd建设

### 使用Doris目录下的logical目录sql语句在doris web ui查询dwd表数据



#注意有部分sql需要输入指定日期

### 2.2 Dws、dim、ads层的做法与dwd一致