# 知识回顾

## Hive定义

Hive是一个基于Hadoop的数据仓库工具。通过HiveQL来操作，实现分布式的海量离线数据处理。

## hive优点

使用类SQL（HQL）开发，避免了MR复杂的开发调试流程，适用于要求短时间内出结果的工作场景。对任何一种编程语言的使用者都更加友好（每个程序员都会SQL），不用对Hadoop底层原理有很深的了解，不用熟记大量的PAI就能开发出比较优秀的海量离线数据分析程序。对于非常复杂的业务，还可以用java代码编写UDF来丰富函数库。

## 数据仓库的概念

数据仓库是一个面向主题的，稳定的，反映历史数据的数据存储系统，为管理者提供决策分析。用空间换时间。

数据仓库和关系型数据库的区别：  
 数据库 数据仓库

面向事务 面向主题

实时数据 历史数据

避免冗余 有意冗余

捕获数据 分析数据

## 存储原理

Hive中数据库就是在HDFS中/user/hive/warehouse下的以.db结尾的一级目录。

表在HDFS中是其对应数据库目录下的子目录。

数据是HDFS中存在于对应表下的文件

默认的default数据库没有自己单独的目录，直接将表放在warehouse下。

## Hive的安装

Hive安装非常简单，只需要解压即可运行，前提是要有JAVA\_HOME和HADOOP\_HOME并且Hadoop要全量启动。

替换元数据库：默认的Derby数据库更换启动目录丢失数据，所以更换为mysql。

替换元数据库要修改hive-site.xml、导入mysql驱动包、开放mysql远程访问权限，注意，mysql中的hive表字符集必须为latin1

## 内部表外部表

创建：

内部表：先有表后有数据。外部表：先有数据，后创建表去管理数据。

删除（drop）：

内部表：元数据信息和HDFS中的数据都会被删除。

外部表：元数据信息会被删除，HDFS中的数据不会被删除。

直接向HDFS中加载数据能够被hive正确管理。

## 分区表

Hive中可以创建单级分区或多级分区表，分区表是hive实现面向主题存储的主要手段。

分区表在直接向HDFS中加载数据时不能被直接管理。需要我们手动增加alter表来增加分区信息。

分区的意义在于将经常查询的数据集放在一起，提高查询效率。

但是分区不是随意创建的不是越多越好的。分区的创建都是经过前期对整个业务的分析后得出的最佳结果。虽然存储空间在HDFS中不是稀缺资源但是仍然不能浪费。

## 分桶表

分桶表主要用于测试阶段，将整个数据集分为若干个小块，通过hash的散列保证每个桶中的数据都有一定的代表性。这样在测试时取若干桶的数据进行测试而不是全量数据参与测试，节省测试时间，提高工作效率。

分桶表创建之后不能直接载入数据，而是需要从原数据所在库中查询出结果后载入。

分桶表在载入数据之前必须开启分桶模式。需要多个reduce同时输出。默认只有一个reduce执行。

## 问题

所有的离线数据处理场景都适用于hive吗？

并不是所有场景都适合，逻辑简单又要求快速出结果的场景Hive优势更大。但是在业务逻辑非常复杂的情况下还是需要开发MapReduce程序更加直接有效。

Hive能作为业务系统的数据库使用吗？

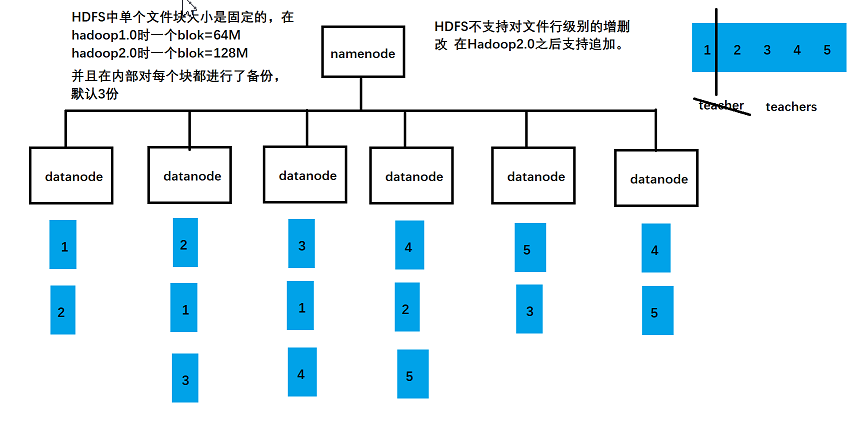
不能。传统数据库要求能够为系统提供实时的增删改查，而Hive不支持行级别的增删改，查询的速度也不比传统关系型数据库，而是胜在吞吐量高，所以不能作为关系型数据库来使用。

Hive与传统MR方式处理数据相比能够提高运行效率吗？能够提高工作效率吗？

Hive的使用中需要将HQL编译为MR来运行，所以在执行效率上要低于直接运行MR程序。但是对于我们来说，由于只需要编写调试HQL，而不用开发调试复杂的MR程序，所以工作效率能够大大提高。

Hive为什么不支持行级别的增删改？

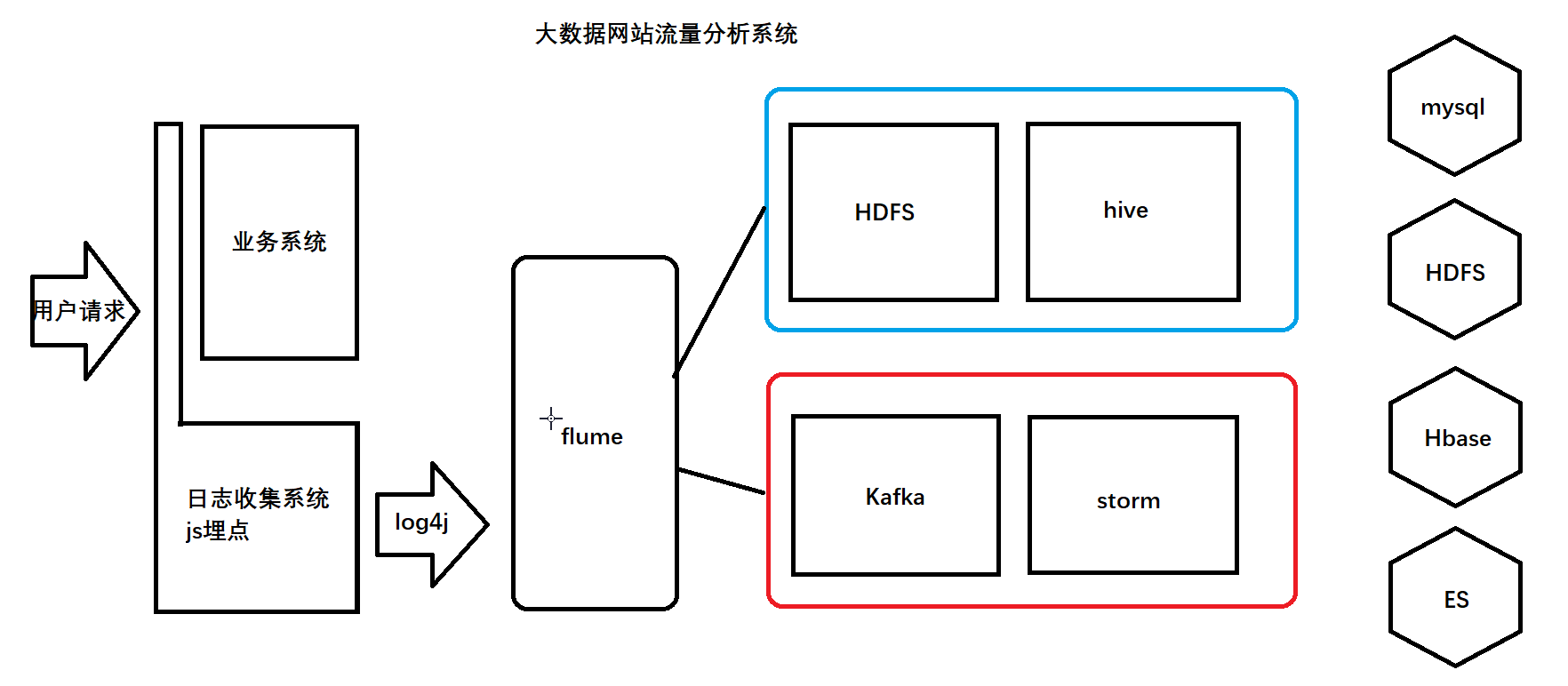
Hive不支持行级别的增删改的根本原因在于他的底层HDFS本身不支持。在HDFS中如果对整个文件的某一段或某一行内容进行增删改，势必会影响整个文件在集群中的存放布局。需要对整个集群中的数据进行汇总，重新切块，重新发送数据到每个节点，并备份，这样的情况是得不偿失的。所以HDFS的设计模式使他天生不适合做这个事，但是在Hadoop2.0之后增加了追加功能，也就是在文件的末尾追加信息是被允许的，这种情况对集群的影响相对较小，而Hive中额insert into table table\_name values(…,…);就是利用2.0之后的追加的原理。但是insert的执行过程还是较长的。



# Hive离线分析

## 准备

### 回顾业务流程



### 搭建环境

1.启动Hadoop

start-all.sh

2.修改flume配置文件

flume-jt.properties

a1.sources.r1.interceptors = t1

a1.sources.r1.interceptors.t1.type = timestamp

a1.sinks.k1.type = hdfs

a1.sinks.k1.hdfs.path = hdfs://hadoop01:9000/flux/reportTime=%Y-%m-%d

a1.sinks.k1.hdfs.fileType=DataStream

3.启动flume

cd /usr/local/src/flume/apache-flume-1.6.0-bin/conf

[root@hadoop01 conf]# ../bin/flume-ng agent -c ./ -f ./flume-jt.properties -n a1 -Dflume.root.logger=INFO,console

4.启动jt-logserver

5.测试

访问locahost/a.jsp和locahost/b.jsp

### 准备数据

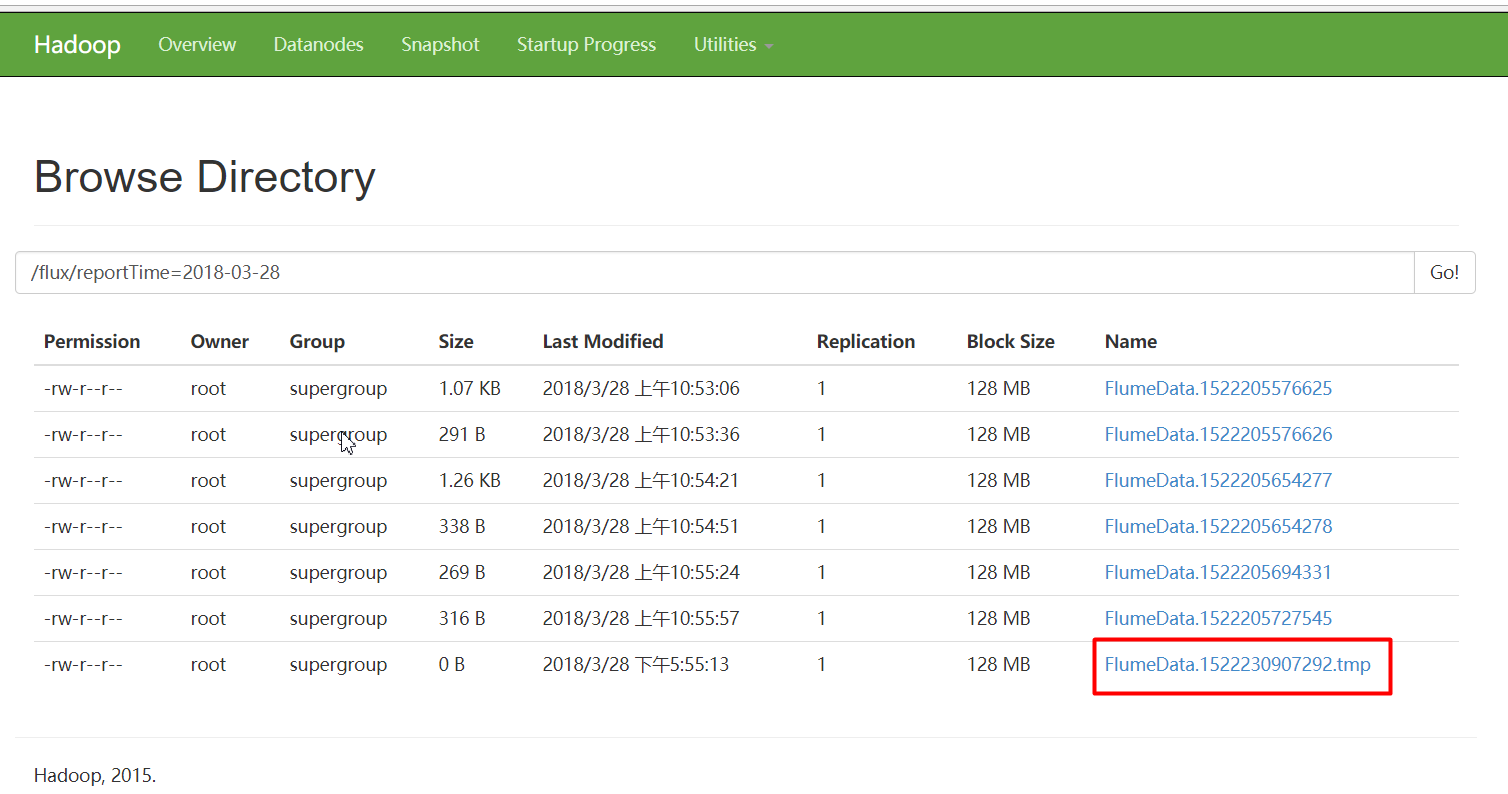
浏览器A：访问3次a.jsp，2次b.jsp关闭浏览器

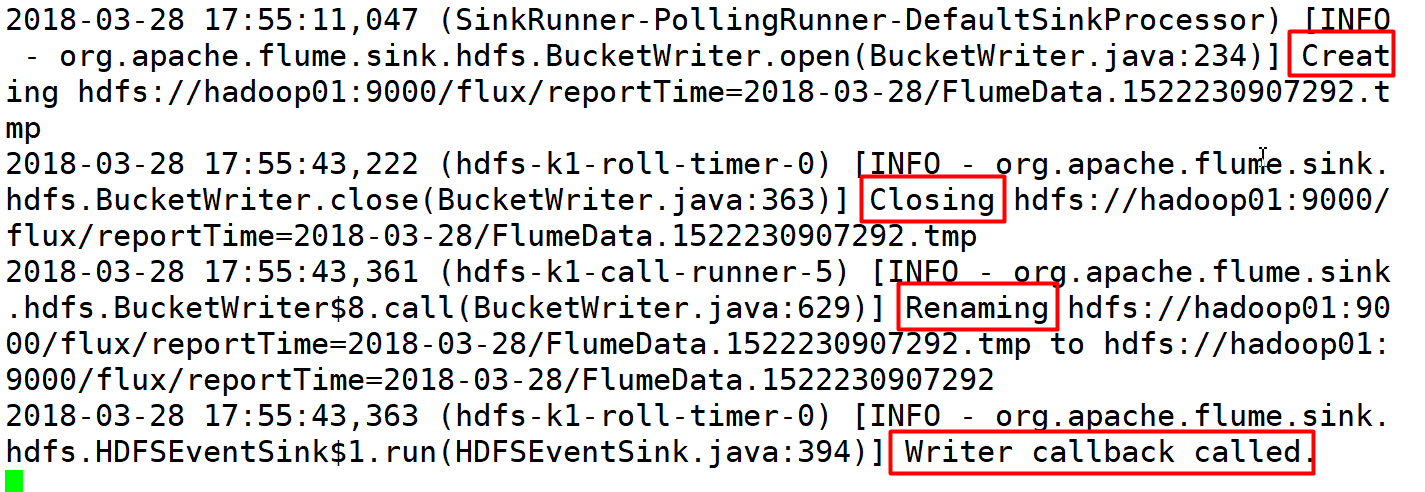
浏览器B：访问3次a.jsp，2次b.jsp关闭浏览器

浏览器A：访问1次a.jps

浏览器B：访问1次b.jps

注意，flume输出的数据不是一条一个单独文件，而是根据我们的配置及自身的策略来决定何时生成一个完整的文件。





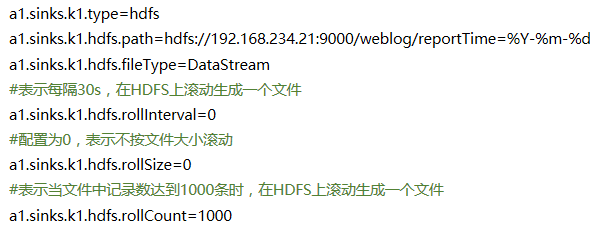
Create：开始创建一个tmp零时文件并写入数据

Closing：关闭写入链接，停止对该文件的操作

Renaming：重命名tmp文件为最终文件

Writer callback called：重置写入状态

如何配置单个文件的大小？



以上配置都有默认值，所以我们不配也没有问题。但是即使配置了固定值，flume也不一定会按照我们的想法来执行，通过翻阅源码可以发现，flume除了安装配置数据执行外还额外增加了自己的判定逻辑，当长时间没有活动时，也会关闭本次链接，生成一个完整的文件。

## 离线数据处理

### Hive管理数据

创建flux外部表，管理HDFS中的日志信息。

hive> create database jtlogdb;

hive> use jtlogdb;

hive> create external table flux (url string,urlname string,title string,chset string,src string,col string,lg string, je string,ec string,fv string,cn string,ref string,uagent string,stat\_uv string,stat\_ss string,cip string) partitioned by (reportTime string) row format delimited fields terminated by '|' location '/flux';

create external table flux：创建外部表

partitioned by (reportTime string)：根据日期分区

row format delimited fields terminated by '|'：通过 | 分割数据

location '/flux'：管理HDFS中/flux文件夹

url string

urlname string

title string

chset string

src string

col string

lg string

je string

ec string

fv string

cn string

ref string

uagent string

stat\_uv string

stat\_ss string

cip string

原始数据很多，但并不是所有的数据都跟我们的业务有关。所以。在正式处理之前我们还会对flux表做一次清洗。去除不相干的数据。

查询flux表

Select \* from flux;

发现并没有数据，这是为什么？---没有添加分区信息。

添加分区信息：

alter table flux add partition (reportTime='2018-07-07) location '/flux/reportTime=2018-07-07;

再次查看整表，发现数据已经被正确管理了。

### 数据清洗

需要的数据字段

reportTime 产生日期

url 访问路径

urlname 页面名称

uvid 访客id

ssid 会话id

sscount 会话编号

sstime 会话时间戳

cip 访客ip

创建数据清洗表：

create table dataclear(reportTime string,url string,urlname string,uvid string,ssid string,sscount string,sstime string,cip string) row format delimited fields terminated by '|';

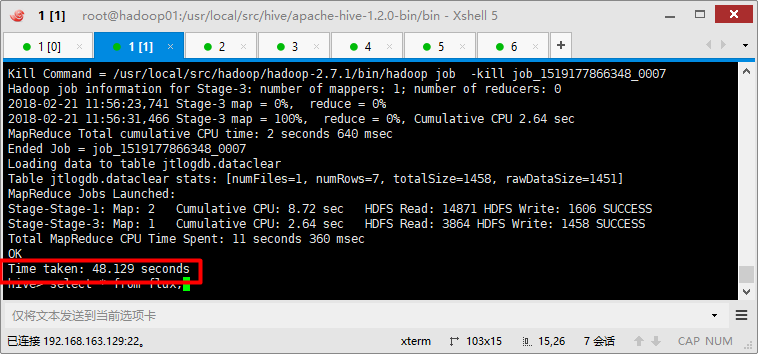
需要注意的是，在hive中将一个表内的数据导入另一个表中时，两个表的创建结构必须相同，包括分隔符！否则可能会发生数据错乱。

清洗并导入数据：

insert overwrite table dataclear select reportTime,url,urlname,

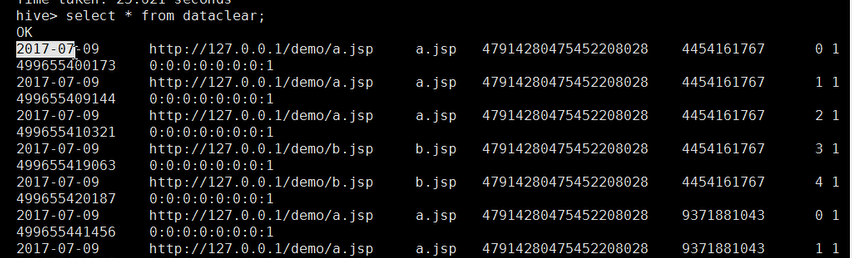
stat\_uv,split(stat\_ss,"\_")[0],split(stat\_ss,"\_")[1],split(stat\_ss,"\_")[2],cip from flux;

这个过程执行较慢。

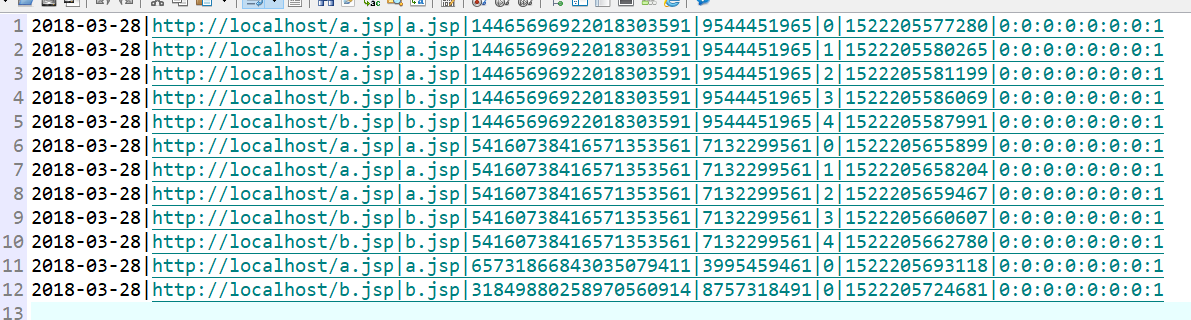


导入数据成功之后查询该表：

select \* from dataclear;



HDFS中下载查看数据：



## 数据处理

### PV：访问量

select count(\*) as pv from dataclear

where reportTime='2018-07-07';

实际就是有效日志条数

### UV：独立访客数

select count(distinct uvid) as uv from dataclear

where reportTime='2018-03-01';

关闭一次浏览器或更换一次浏览器就被认为是一次独立的访问，不同的访客

### VV：独立会话数

select count(distinct ssid) as vv from dataclear

where reportTime='2018-03-01';

session即会话，浏览器用cookie存储sessionid所以不同的cookie就代表不同的会话，其中我们使用了两个浏览器，清除了两次cookie，来模拟不同的会话。

### BR：跳出率

select br\_taba.a/br\_tabb.b as br from

(

select count(\*) as a from

(

select ssid from dataclear

where reportTime='2018-03-01'

group by ssid having count(ssid)=1

) as br\_tab

) as br\_taba,

(

select count(distinct ssid) as b from dataclear

where reportTime='2018-03-01'

) as br\_tabb;

select br\_taba.a/br\_tabb.b as br from (select count(\*) as a from (select ssid from dataclear where reportTime='2018-07-07' group by ssid having count(ssid)=1) as br\_tab) as br\_taba,(select count(distinct ssid) as b from dataclear where reportTime='2018-07-07') as br\_tabb;

跳出率就是，只访问了一个页面就走了的会话/会话总数。

为了控制结果的精确度，我们应用round函数来对结果进行处理，取小数点后四位（四舍五入）

select round(br\_taba.a/br\_tabb.b,4) as br from (select count(\*) as a from (select ssid from dataclear where reportTime='2018-03-01' group by ssid having count(ssid)=1) as br\_tab) as br\_taba,(select count(distinct ssid) as b from dataclear where reportTime='2018-03-01') as br\_tabb;

### NewIP：新增IP数

select count(distinct dataclear.cip) from dataclear

where dataclear.reportTime='2018-07-07'

and cip not in

(select dc2.cip from dataclear as dc2

where dc2.reportTime<'2018-07-07');

select count(distinct dataclear.cip) from dataclear where dataclear.reportTime='2018-07-07' and cip not in (select dc2.cip from dataclear as dc2 where dc2.reportTime<'2018-07-07');

新增ip数就是当天来访的所有ip中之前从来没有访问过的ip数量。

比如：我们的系统昨天上线，昨天访客有：韩少云，王春梅，陈子枢

今天的访客有：陈子枢，刘昱江，董长春。那么新增访客就是刘昱江和董长春，对应的新增ip数就是2。

### NewCust：新增访客数

select count(distinct dataclear.uvid) from dataclear

where dataclear.reportTime='2018-03-01'

and uvid not in

(select dc2.uvid from dataclear as dc2 where

dc2.reportTime < '2018-03-01');

select count(distinct dataclear.uvid) from dataclear where dataclear.reportTime='2018-07-07' and uvid not in (select dc2.uvid from dataclear as dc2 where dc2.reportTime < '2018-07-07');

原理与NewIP一样。只不过指标变为uvid

### AvgTime：平均访问时长

select round(avg(atTab.usetime),4) as avgtime from

(

select max(sstime) - min(sstime) as usetime from dataclear

where reportTime='2018-03-01'

group by ssid

) as atTab;

select round(avg(atTab.usetime),4) as avgtime from (select max(sstime) -min(sstime) as usetime from dataclear where reportTime='2018-07-07' group by ssid) as atTab;

平均访问时长指的是所有会话的时长的平均数。

### AvgDeep：平均访问深度

select round(avg(deep),2) as viewdeep from

(

select count(distinct urlname) as deep from flux

where reportTime='2018-03-28'

group by split(stat\_ss,'\_')[0]

) as tviewdeep;

select round(avg(deep),2) as viewdeep from (select count(distinct urlname) as deep from flux where reportTime='2018-06-05' group by split(stat\_ss,'\_')[0]) as tviewdeep;

访问深度，指一个会话中浏览的页面个数。

## 分析结果表

创建业务表并插入数据

hive> create table tongji(reportTime string,pv int,uv int,vv int, br double,newip int, newcust int, avgtime double,avgdeep double) row format delimited fields terminated by '|';

hive>insert overwrite table tongji select '2018-07-07',tab1.pv,tab2.uv,tab3.vv,tab4.br,tab5.newip,tab6.newcust,tab7.avgtime,tab8.avgdeep from

(select count(\*) as pv from dataclear where reportTime = '2018-07-07') as tab1,

(select count(distinct uvid) as uv from dataclear where reportTime = '2018-07-07') as tab2,

(select count(distinct ssid) as vv from dataclear where reportTime = '2018-07-07') as tab3,

(select round(br\_taba.a/br\_tabb.b,4)as br from (select count(\*) as a from (select ssid from dataclear where reportTime='2018-07-07' group by ssid

having count(ssid) = 1) as br\_tab) as br\_taba,

(select count(distinct ssid) as b from dataclear where reportTime='2018-07-07') as br\_tabb) as tab4,

(select count(distinct dataclear.cip) as newip from dataclear where dataclear.reportTime = '2018-07-07' and cip not in (select dc2.cip from dataclear

as dc2 where dc2.reportTime < '2018-07-07')) as tab5,

(select count(distinct dataclear.uvid) as newcust from dataclear where dataclear.reportTime='2018-07-07' and uvid not in (select dc2.uvid from

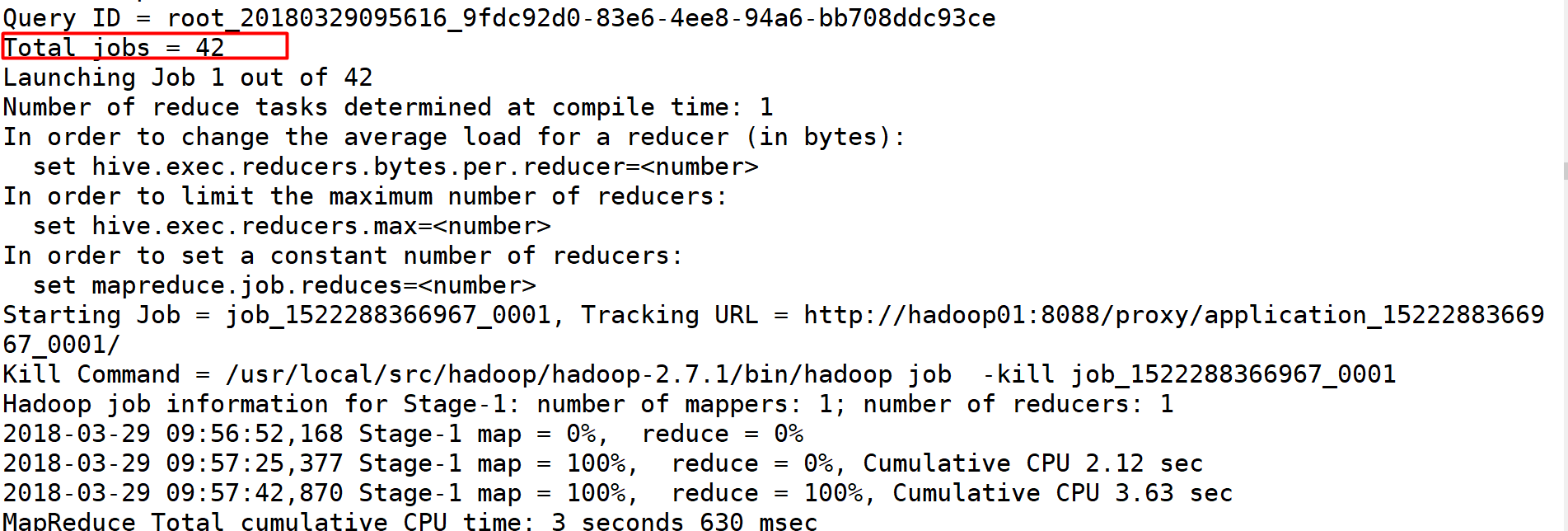
dataclear as dc2 where dc2.reportTime < '2018-07-07')) as tab6,

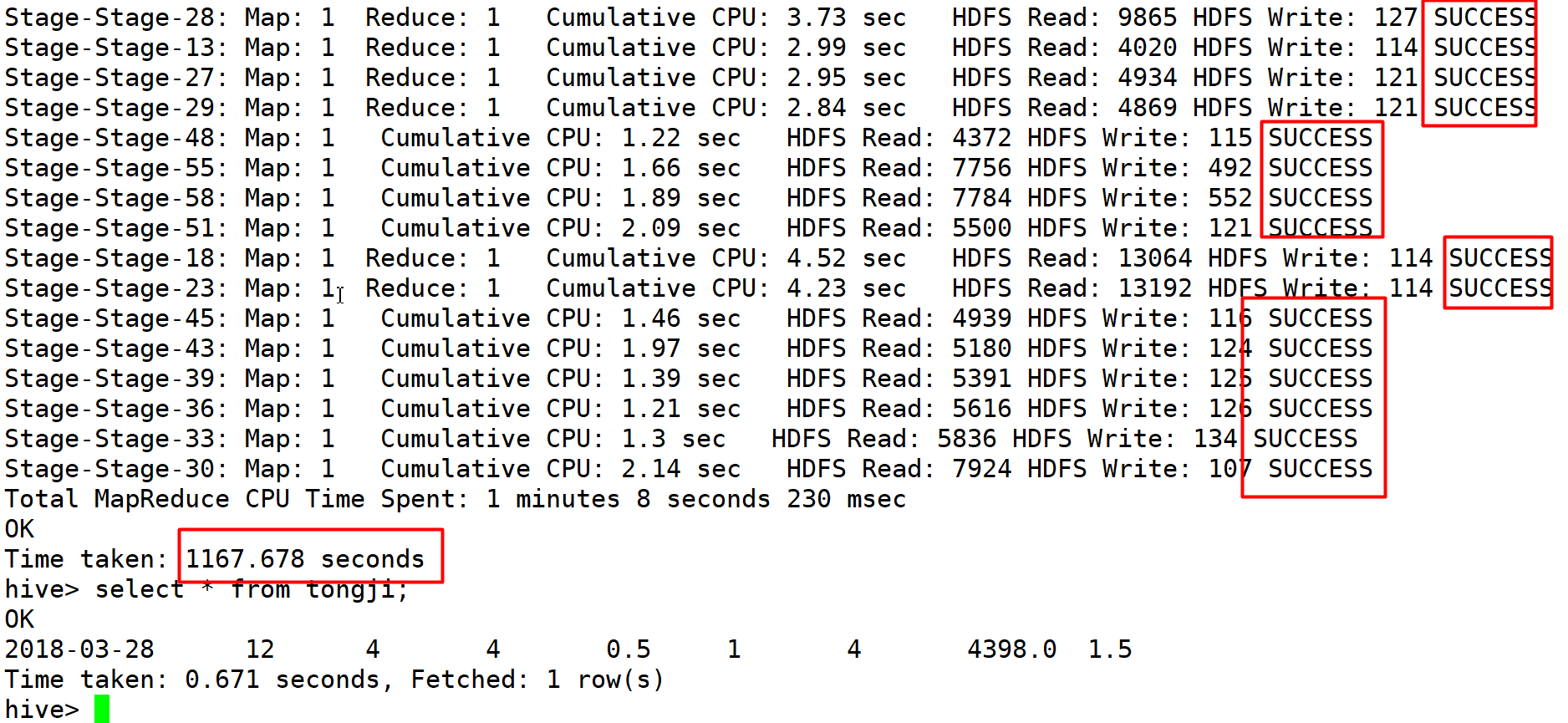
(select round(avg(atTab.usetime),4) as avgtime from (select max(sstime) - min(sstime) as usetime from dataclear where reportTime='2018-07-07'

group by ssid) as atTab) as tab7,

(select round(avg(deep),4) as avgdeep from (select count(distinct urlname) as deep from dataclear where reportTime='2018-07-07' group by ssid) as

adTab) as tab8;





## 通过sqoop将数据导入mysql

### 概念

sqoop 沟通hdfs和关系型数据库的桥梁，可以从hdfs导出数据到关系型数据库，也可以从关系型数据库导入数据到hdfs

### 下载

Apache 提供的工具

### 安装

要求必须有jdk 和 hadoop的支持，并且有版本要求。

上传到linux中，进行解压

sqoop可以通过JAVA\_HOME找到jdk 可以通过HADOOP\_HOME找到hadoop所以不需要做任何配置就可以工作。

需要将要连接的数据库的驱动包加入sqoop的lib目录下

### 使用

在mysql中创建jtlog数据库

CREATE TABLE jtdata (

reportTime varchar(100),

pv bigint(20),

uv bigint(20),

vv bigint(20),

br double,

newip bigint(20),

newcust bigint(20),

avgdeep double,

avgtime double

)

从关系型数据库导入数据到hdfs:

./sqoop import --connect jdbc:mysql://192.168.65.1:3306/jtlog --username root --password root --table jtdata -m 1 --target-dir '/sqoop/jtlog' --fields-terminated-by '|';

从hdfs导出数据到关系型数据库:

./sqoop export --connect jdbc:mysql://192.168.65.1:3306/jtlog --username root --password root --export-dir '/user/hive/warehouse/jtlogdb.db/tongji' --table jtdata -m 1 --fields-terminated-by '|';