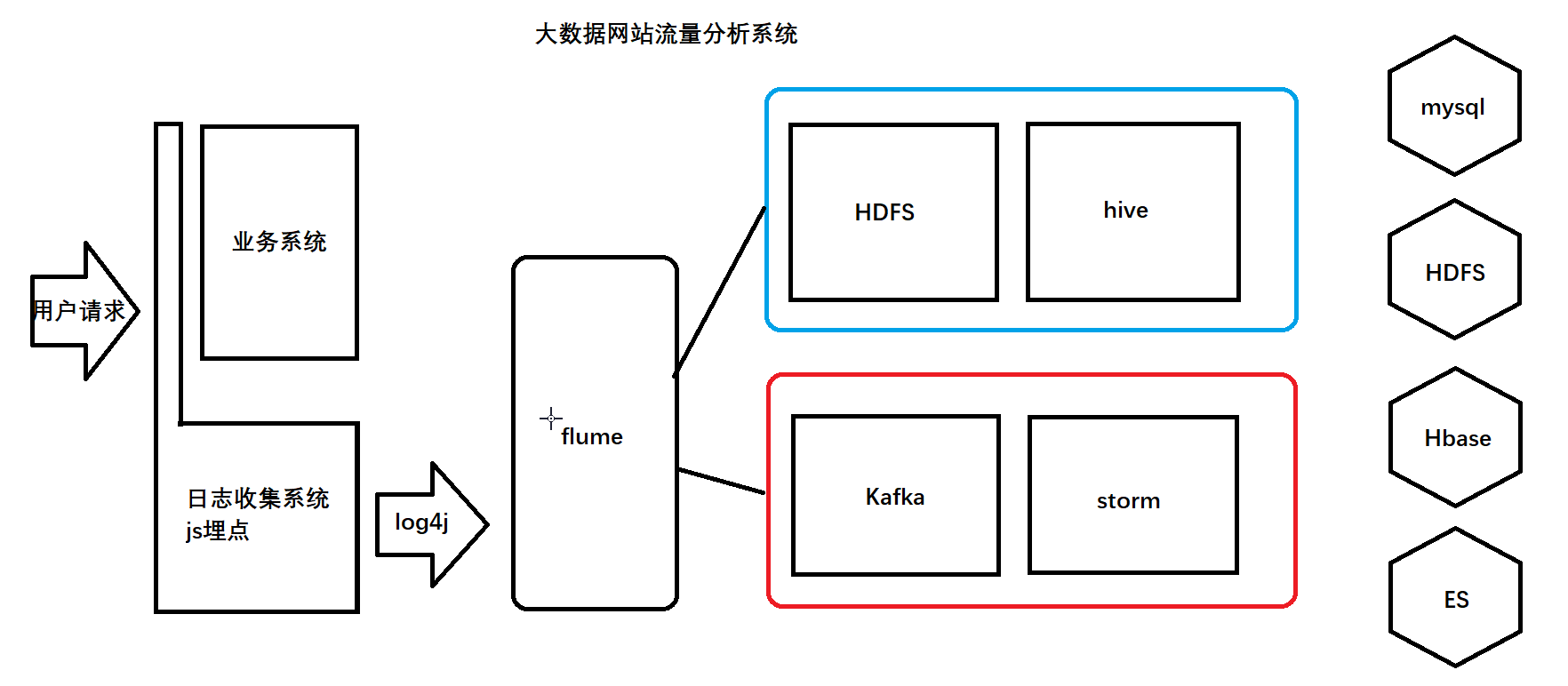
# Hive离线分析

## 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.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 reportTime=2018-11-19

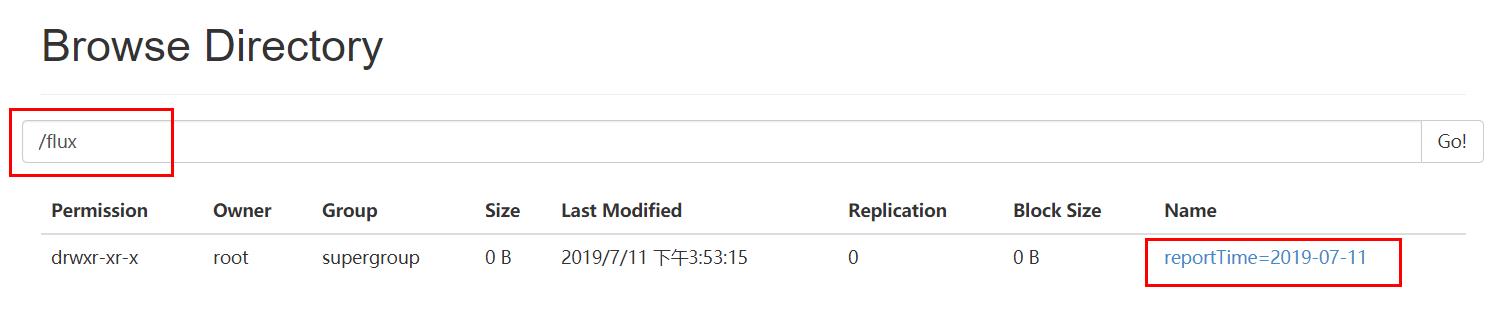
cd /usr/local/src/flume/apache-flume-1.6.0-bin/conf

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

4.启动jt-logserver

5.测试

访问localhost/a.jsp和localhost/b.jsp查考hadoop客户端是否有目录flux;



若测试失败没有flux目录，请修改配置文件。

### 准备数据(\*删除hadoop里面的flux目录并准备俩个浏览器都清楚cookie\*)

hadoop fs -rm -R /flux

浏览器A：访问3次a.jsp，2次b.jsp关闭浏览器

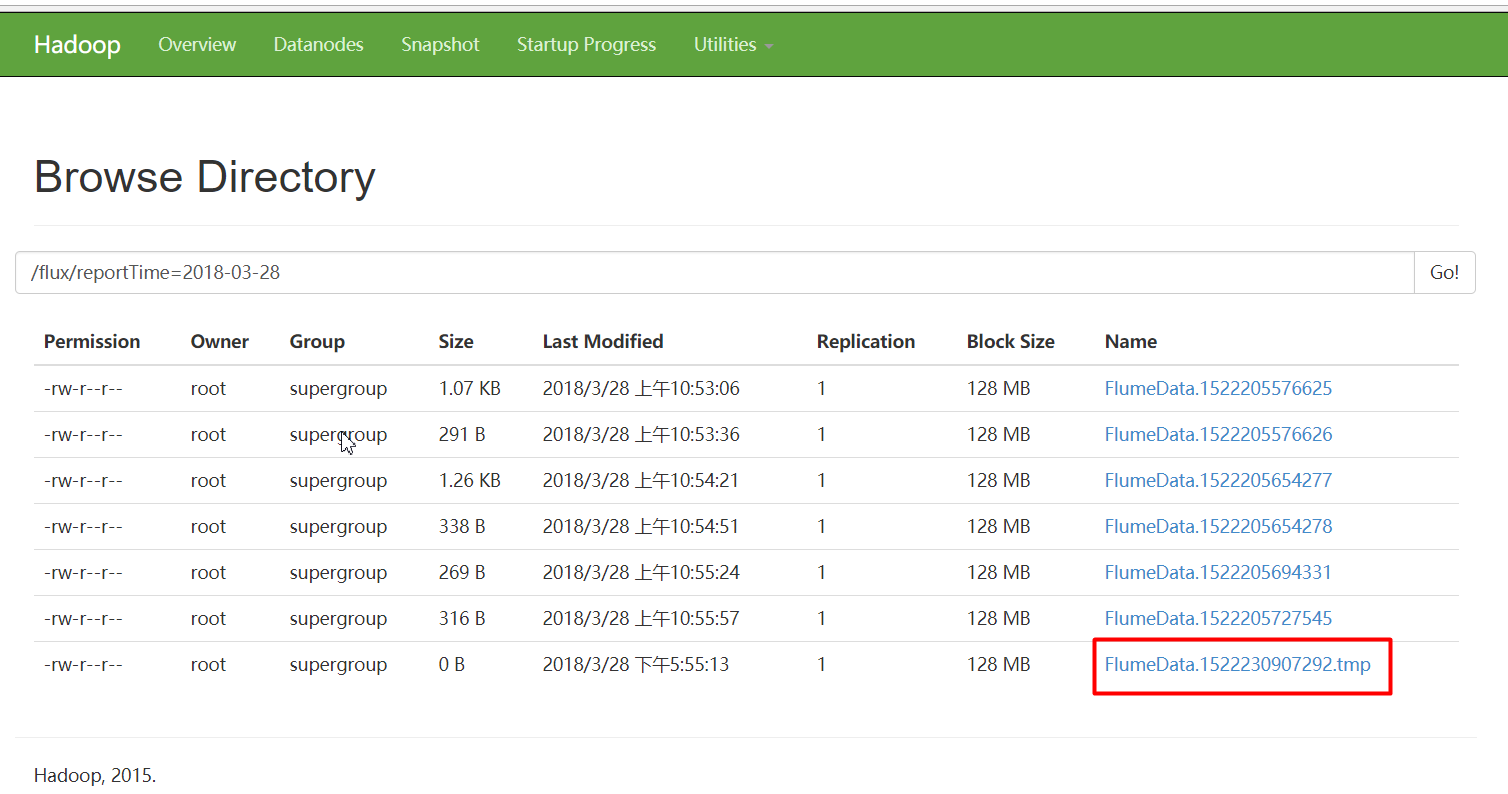
浏览器B：访问3次a.jsp，2次b.jsp关闭浏览器

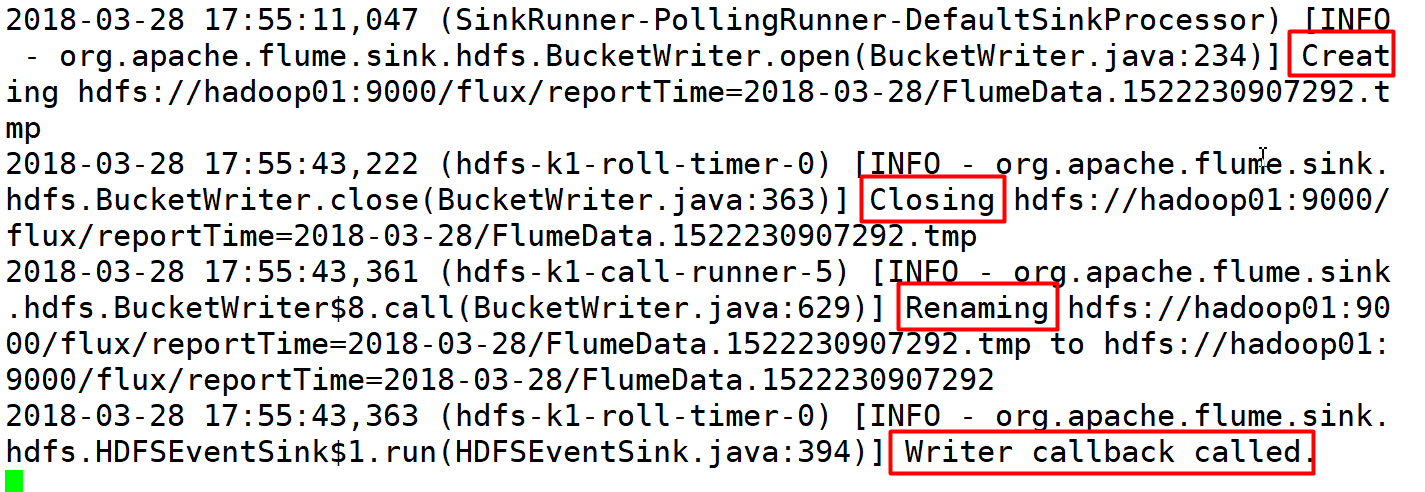
浏览器A：访问1次a.jps

浏览器B：访问1次b.jps

浏览器访问完后，停止jt\_logserver服务，防止在产生数据。

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。





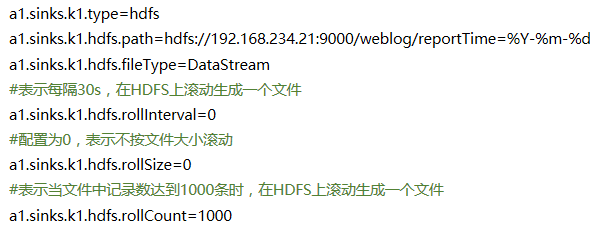
Create：开始创建一个tmp零时文件并写入数据

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

此时可以把jt\_logserver服务停止，flume进程停止；

### Hive管理数据

创建flux外部表，管理HDFS中的日志信息。

hive> create database xzlogdb;

hive> use xzlogdb;

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：（\*日期给今天的\*）

alter table flux add partition (reportTime='2019-07-11') location '/flux/reportTime=2019-07-11';

再次查看整表，发现数据已经被正确管理了（但是数据太多看起来很乱）。

### 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 '|';

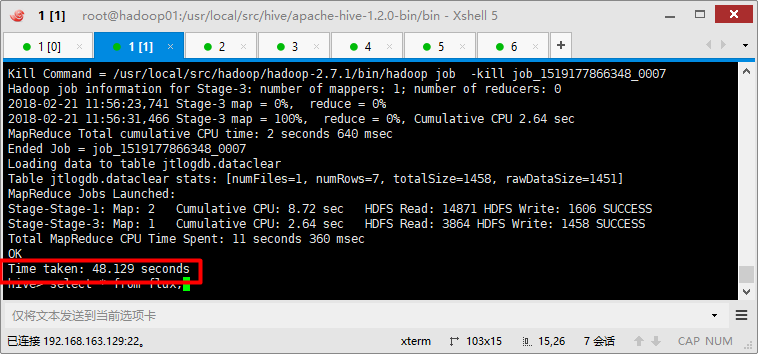
需要注意的是，在hive中将一个表内的数据导入另一个表中时，两个表的创建结构必须相同，包括分隔符！否则可能会发生数据错乱。

清洗并导入数据：

insert overwrite table dataclear select reportTime,url,urlname,

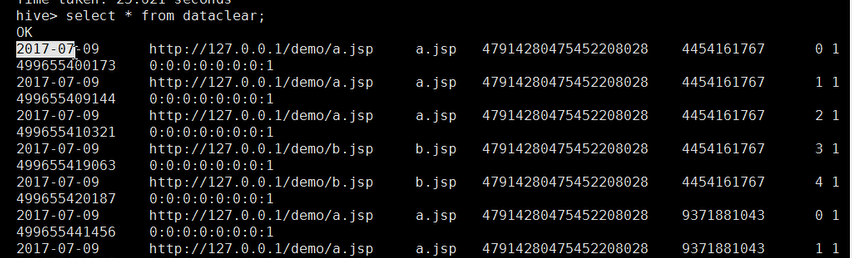
stat\_uv,split(stat\_ss,"\_")[0],split(stat\_ss,"\_")[1],split(stat\_ss,"\_")[2],cip from flux;

这个过程执行较慢。

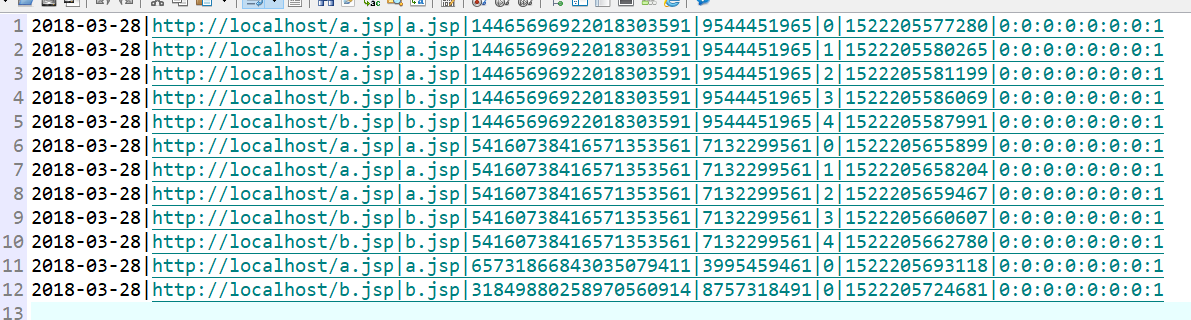


导入数据成功之后查询该表：

select \* from dataclear;



HDFS中下载查看数据：网页/user/hive/warehouse/xzlogdb.db/dataclear下



## 数据处理

### PV：访问量

select count(\*) as pv from dataclear

where reportTime='2019-07-11';

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';

记录不同用户的20位随机数（uvid），去重后进行计数。

### 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='2019-01-22' group by ssid having count(ssid)=1) as br\_tab) as br\_taba,(select count(distinct ssid) as b from dataclear where reportTime='2019-01-22') 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='2019-01-22' group by ssid having count(ssid)=1) as br\_tab) as br\_taba,(select count(distinct ssid) as b from dataclear where reportTime='2018-11-19') as br\_tabb;

### 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='2019-01-22' and uvid not in (select dc2.uvid from dataclear as dc2 where dc2.reportTime < '2019-01-22');

2。指标为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='2018-12-19' 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='2019-03-19' 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。