1. 问题描述

需求：订单服务器产生数据，要求统计每 1 分钟的交易量

1、统计总的交易额

2、审计前面的金额。

思路：

0.产生交易数据

0.1 使用 SheetGeneratorServer 类产生交易数据，每 1s 产生一个记录；

0.2 产生的数据使用 log4j 写入到日志文件中，按照每 128MB 产生一个文件；

0.3 产生的数据同时写入到 kafka 中，topic 自定义；

1.通过实时进行统计汇总，这个与前面做的内容思路类似；

1.1 使用 storm 消费 kafka 中的数据，代码单独建立一个工程去实现；

1.2 在 redis 中存储的是实时查询的数据；

1.3 在 MySQL 中存储每一分钟的数据；

```
create table trade_minutes(
    id bigint primary key auto_increment,
    value decimal(10,2), --表示每分钟的交易额
    logtime datetime  --表示每分钟的时间戳
);
```

2.通过离线进行统计汇总，与实时的数据做对照；

2.1 使用 Flume NG 监控日志文件中的数据变化，实时写入到 HDFS 中，HDFS 的路径自定义；

2.2 创建 hive 的外部表管理导入的数据；

3.审计

3.1 使用 java 代码写一个方法

```
//形参是 201612070949 格式
```

```
boolean audit(String minute){
    //1.查询MySQL
    //2.查询Hive的时候，范围是[20161207094900, 20161207095000)
    //3.判断两个结果是否相等
}
```

2. 集群启动

2.1. 关闭防火墙

```
service iptables stop
```

2.2. Redis

```
redis-server /opt/redis/etc/redis.conf
```

2.3. Mysql

```
service mysqld start
```

2.4. Zookeeper

在每个节点上执行bin/zkServer.sh start
启动之后，一定要查看文件zookeeper.out。
并且使用ps -ef |grep zookeeper查看进程是否存在

2.5. Kafka

2.5.1. 启动命令

```
nohup  bin/kafka-server-start.sh  config/server.properties>nohup.out
2>&1 &
```

2.5.2. 命令行操作

创建主题

```
bin/kafka-topics.sh --create --zookeeper
192.168.1.100:2181,192.168.1.101:2181,192.168.1.102:2181
--replication-factor 1 --partitions 1 --topic test
```

描述主题

```
bin/kafka-topics.sh --describe --zookeeper
192.168.1.100:2181,192.168.1.101:2181,192.168.1.102:2181 --topic test
```

删除主题

```
bin/kafka-topics.sh --delete --zookeeper
192.168.1.100:2181,192.168.1.101:2181,192.168.1.102:2181 --topic test
```

生产者

```
bin/kafka-console-producer.sh --broker-list
192.168.1.100:9092,192.168.1.101:9092,192.168.1.02:9092 --topic test
```

消费者

```
bin/kafka-console-consumer.sh --zookeeper
192.168.1.100:2181,192.168.1.101:2181,192.168.1.102:2181 --topic test
--from-beginning
```

查看消费进度

```
bin/kafka-run-class.sh kafka.tools.ConsumerOffsetChecker
--zookeeper 192.168.1.100:2181 --topic test -group group1
```

Pid	Offset	logSize	Lag
0	10	10	0
1	5	5	0
分区编号	消费了多少	一共有多少	还有多少没消费

2.6. Storm

2.6.1. 启动命令

启动 nimbus

```
nohup bin/storm nimbus >storm.out  2>&1  &
```

启动 supervisor

```
nohup bin/storm supervisor >storm.out  2>&1  &
```

启动 ui

```
nohup bin/storm ui >storm.out  2>&1  &
```

2.6.2. 命令行操作

提交代码到 storm 集群运行。

把代码打成 jar 包，执行 storm jar xxxx.jar xxxxxTopology

2.7. Hdfs

```
sbin/start-dfs.sh
```

2.8. Hive

在 hive 中运行以下代码，打开对 jdbc 的监听

```
nohup hive  --service  hiveserver  -p 10000 >/dev/null  2>/dev/null &
```

外部表

```
create  external  table  trade(time  string,  ordernum  string,goods
string,price  string,amount  string)  row  format  delimited  fields
terminated by '\t' location '/imxlee/out';
```

2.9. Flume

nohup flume-ng agent -n agent1 -f spoll2hdfs.agent -Dflume.root.logger=DEBUG,console &

3. 项目代码

3.1. 产生随机订单

3.2. 写 Log4j 日志

以下是 log4j 的配置文件

```
log4j.rootLogger=ERROR, stdout, R
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout

log4j.appender.stdout.layout.ConversionPattern=%m%n

log4j.appender.R=org.apache.log4j.RollingFileAppender
log4j.appender.R.File=./logs/example.log
log4j.appender.R.MaxFileSize=12KB
log4j.appender.R.layout=org.apache.log4j.PatternLayout
log4j.appender.R.layout.ConversionPattern=%m%n

log4j.logger.cn.crxy=INFO
```

这里设定的输出文件大小是 12KB，方便测试。

特别注意 ConversionPattern 的格式，目的是只输出 msg，不输出其他。

运行稳定后，这里的 stdout 可以删除。目前存在的意义是为了测试方便。

3.3. 写 Kafka

创建主题语句

/opt/kafka/bin/kafka-topics.sh --create --zookeeper 192.168.1.100:2181,192.168.1.101:2181,192.168.1.102:2181 --replication-factor 1 --partitions 1 --topic sheet_topic

3.4. 打 jar 包部署

使用 maven-assembly-plugin 插件打包

```
<plugin>
            <artifactId>maven-assembly-plugin</artifactId>
            <configuration>
                <descriptorRefs>

<descriptorRef>jar-with-dependencies</descriptorRef>
                </descriptorRefs>
                <archive>
                    <manifest>

<mainClass>xxxx.SheetGeneratorServer</mainClass>
                    </manifest>
                </archive>
            </configuration>
            <executions>
                <execution>
                    <id>make-assembly</id>
                    <phase>package</phase>
                    <goals>
                        <goal>single</goal>
                    </goals>
                </execution>
```

```
            </executions>
        </plugin>
```

执行命令 mvn clean package 就能打包，结果在 target 目录中。

打完包后，使用 winrar 打开，检查一下里面的内容是否全面。特别是检查 log4j.properties 和 price900 文件是否存在。

3.5. 运行 jar

上传到 linux 后，执行 java -jar xxxxx.jar 就可以运行。

注意观察输出的 log4j 日志内容。观察文件的变化。

3.6. 使用 flume NG 收集到 HDFS

agent 的配置文件如下所示

```
#起名字
agent1.sources=s1
agent1.sinks=k1
agent1.channels=c1

#配置 source
agent1.sources.s1.type=exec
agent1.sources.s1.command=tail -F /home/wuchao/logs/example.log
agent1.sources.s1.channels=c1

#配置 sink
agent1.sinks.k1.type = hdfs
agent1.sinks.k1.channel = c1
agent1.sinks.k1.hdfs.path = hdfs://192.168.1.100:9000/user/out
agent1.sinks.k1.hdfs.fileType = DataStream
agent1.sinks.k1.hdfs.rollInterval=0
agent1.sinks.k1.hdfs.rollSize=10240
```

```
agent1.sinks.k1.hdfs.rollCount=0
agent1.sinks.k1.hdfs.idleTimeout=0

#配置 channel
agent1.channels.c1.type = SPILLABLEMEMORY
```

注意这里 hdfs 的配置。默认 hdfs 大小是 10KB，方便测试。

执行命令

nohup flume-ng agent -n agent1 -f exec2hdfs.agent -Dflume.root.logger=DEBUG,console & 就可以启动 flume

3.7. 创建 Hive 外部表管理数据

建表语句如下：

```
create external table t11(logtime int, sid string, pid string, price double, amount int)
row format delimited fields terminated by '\t'
location '/user/out';
```

显示结构如下

```
hive> select * from t11 limit 10;
OK
NULL    000005300       6935918600209   2.8     35
NULL    000005301       20209044        0.8     94
NULL    000005302       6921160173172   5.8     78
NULL    000005303       6935918600605   3.2     27
NULL    000005304       6923721201126   3.0     35
NULL    000005305       6920907800616   3.5     98
NULL    000005306       6921734971012   0.9     69
NULL    000005307       6903148031957   8.2     28
NULL    000005308       6938382501638   2.8     47
NULL    000005309       20209079        0.65    36
Time taken: 0.133 seconds, Fetched: 10 row(s)
hive> drop table t11;
OK
```

（请忽略左侧的 NULL，着急，没调试）

3.8. 使用 Storm 消费 Kafka 数据