|  |  |
| --- | --- |
| 成绩 |  |



**课程设计报告**



**题 目 基于Flink风电实时数据采集**

**课 程 名 称 毕业实习**

**院 部 名 称**  软件工程学院

**专 业** 软件工程（本）

**班 级** 15软件工程（Z）

**学 生 姓 名**  吴跟强

**学 号** 1512001066

**课程设计地点 南京天数科技**

**指 导 教 师** 陶玉婷

金陵科技学院教务处制

**课程设计报告书写要求**

课程设计报告原则上要求学生手写，要求书写工整。若因课程特点需打印的，要遵照以下字体、字号、间距等的具体要求。纸张一律采用A4的纸张。

**课程设计报告书写说明**

课程设计报告应包含以下七部分内容：1、摘要  2、目录  3、前言/引言  4.正文 5. 结论   6. 参考文献    7. 附录，每部分的书写要求参见具体条目要求。

**填写注意事项**

（1）准确说明，层次清晰。

（2）尽量采用专用术语来说明事物。

（3）外文、符号、公式要准确，应使用统一规定的名词和符号。

（4）应独立完成课程设计报告的书写，严禁抄袭、复印，一经发现，以零分论处。

**课程设计报告批改说明**

课程设计报告的批改要及时、认真、仔细，一律用红色笔批改。课程设计报告的批改成绩采用五级计分制或百分制，具体评分标准由各院部自行制定。

**课程设计报告装订要求**

报告批改完毕后，任课老师应将课程设计报告以自然班为单位、按学号升序排列，并附上一份该课程设计的教学大纲。

实验项目名称： **基于Flink风电实时数据采集**

实验学时： 4周

同组学生姓名： 实验地点： 天数科技

实验日期： 2018.08.06~09.02 实验成绩：

批改教师： 批改时间：

1. 实验目的

传统的用户行为分析系统通常以离线批处理模式根据既定规则对用户数据进行分析。规则相对简单，且更新规则需要重启系统。风机在运行中产生的数据种类多样，数量巨大，有测风数据、电量指标，以及箱变系统、变桨系统、偏航系统和齿轮箱各个设备部件的运行状态和故障信息等等。由于风电机组中的环境、工况、状态参数具有一定的复杂、多变的特性，且各参数之间相互关联。我们可以利用大数据手段，从海量的数据中挖掘出影响风机正常运行的关键因素，提升风机发电量，并对频繁发生故障的部件进行预警，降低运维成本。以提高发电量一项来说。提高发电量是复杂的系统工程，因为它会受到影响的因素非常多。比如主控参数优化，参数设置、偏航优化和变桨优化，需要整体考虑才能得到理想的结果，得到更高的产能和利润。这就需要对风机产生的数据进行整体性、联动性分析，而这就是大数据技术的用武之地了。

风机在运行中产生的数据种类多样，数量巨大，有测风数据、电量指标，以及箱变系统、变桨系统、偏航系统和齿轮箱各个设备部件的运行状态和故障信息等等。由于风电机组中的环境、工况、状态参数具有一定的复杂、多变的特性，且各参数之间相互关联，我们可以利用大数据手段，从海量的数据中挖掘出影响风机正常运行的关键因素，提升风机发电量，并对频繁发生故障的部件进行预警，降低运维成本。

本项目利用大数据技术，对风电数据采集服务器的访问日志进行可视化展示。当访问此程序时，可以看到风电数据采集日志中统计出来的全天分时网站点击量等信息的图表展示。能够支撑海量数据存储。

1. 实验要求

完成项目：

利用大数据技术，对风电数据采集服务器的访问日志进行可视化展示。当访问此程序时，可以看到风电数据采集日志中统计出来的全天分时网站点击量等信息的图表展示。能够支撑海量数据存储。

1. **实验环境、设备**
2. 采用三台虚拟机来模拟真实环境下的Hadoop高可用集群服务。主机名和ip地址分别为master-192.168.75.213、slave1-192.168.75.214、slave2-192.168.75.215。
3. 在master、slave1和salve2上安装Flume数据导入工具、Kafka、Flink
4. Master上安装OpenTSDB、Grafana。

实验流程：

1. JDK版本使用1.8。
2. 搭建hadoop集群以及yarn集群。
3. 安装并配置flume 。
4. 引入Kafka进行缓存。
5. 安装flink，使用flink流处理kafka的数据。
6. 处理完数据存储进OpenTSDB （还需配置HBase、Gnuplot）。

读取数据使用Grafana可视化工具显示在web端进行展示。

1. 实验代码

### 项目架构图

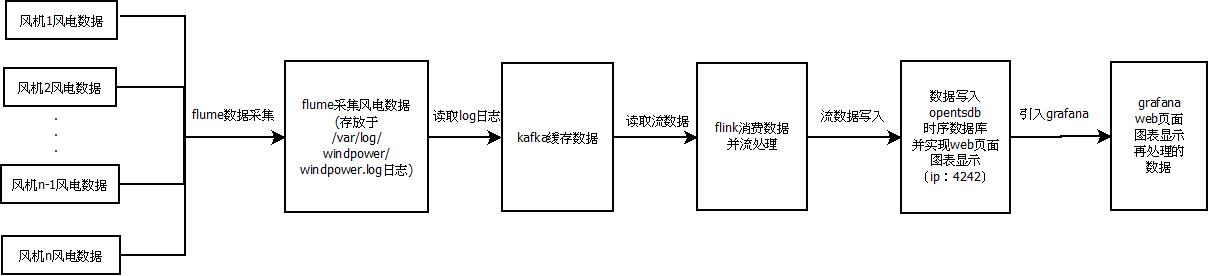


图1 项目架构图

### 流程图

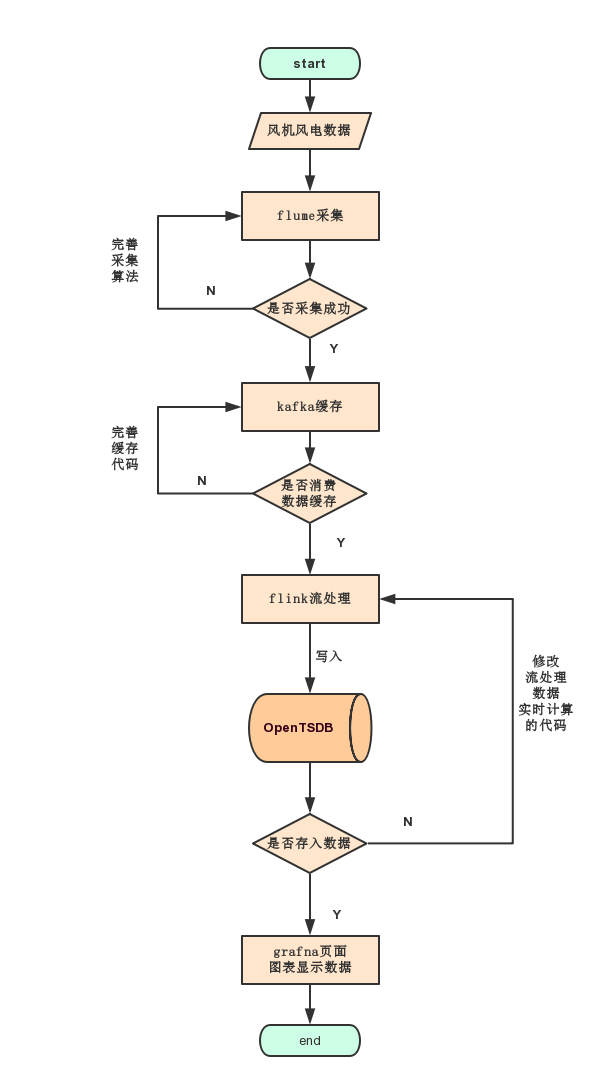


图2 流程图

### 时序图

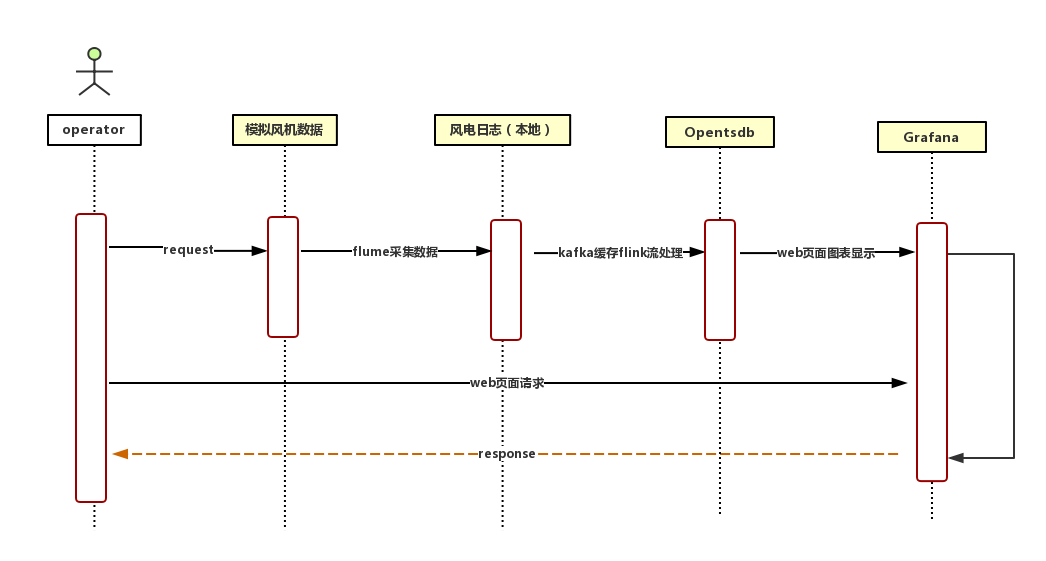


图3 时序图

## 功能点设计

### 4.1模拟风电数据

日志设计：

参数：

1. 系统时间: systemTime= `date +%Y-%m-%d\ %H:%M:%S`
2. 环境温度: ambientTemperature
3. 机身温度: fuselageTemperature
4. 风速: windSpeed
5. 发电量: powerGeneration
6. 风机编号:fanNumber
7. 机型:model

|  |  |  |
| --- | --- | --- |
| **字段名** | **字段含义** | **样例** |
| systemTime | 系统时间 | 2018-08-30 12:25:00 |
| ambientTemperature | 环境温度 | 26.43 |
| fuselageTemperature | 机身温度 | 68.52 |
| windSpeed | 风速 | 20.63 |
| powerGeneration | 发电量 | 100.54 |
| fanNumber | 风机编号 | 001 |
| model | 机型 | X |

自动化脚本运行获取模拟数据：

|  |
| --- |
| #!/bin/bash  step=1  RANDOM\_MAX\_INT=32768  function random\_double() {  fanNumber\_list=("001" "002" "003")  model\_list=("X" "Y" "Z")  local environmentmin=23  local environmentmax=36  local fuselageTempmin=68  local fuselageTempmax=76  local windSpeedmin=20  local windSpeedmax=23  local powerGeneramin=100  local powerGeneramax=102  #风机编号  fanNumber\_num=$(((RANDOM%3)))  fanNumber=${fanNumber\_list[$fanNumber\_num]}  #机型  model\_num=$(((RANDOM%3)))  model=${model\_list[$model\_num]}  #环境温度  local random1=$( echo "scale=2;${environmentmin}+${RANDOM}/${RANDOM\_MAX\_INT}\*(${environmentmax}-${environmentmin}+1)" | bc -l )  #机身温度  local random2=$( echo "scale=2;${fuselageTempmin}+${RANDOM}/${RANDOM\_MAX\_INT}\*(${fuselageTempmax}-${fuselageTempmin}+1)" | bc -l )  #风速  local random3=$( echo "scale=2;${windSpeedmin}+${RANDOM}/${RANDOM\_MAX\_INT}\*(${windSpeedmax}-${windSpeedmin}+1)" | bc -l )  #发电量  local random4=$( echo "scale=2;${powerGeneramin}+${RANDOM}/${RANDOM\_MAX\_INT}\*(${powerGeneramax}-${powerGeneramin}+1)" | bc -l )  mylist[0]=$fanNumber  mylist[1]=$model  mylist[2]=$random1  mylist[3]=$random2  mylist[4]=$random3  mylist[5]=$random4  current=`date "+%Y-%m-%d %H:%M:%S"`  timeStamp=`date -d "$current" +%s`  #currentTimeStamp=$((timeStamp\*1000+`date "+%N"`/1000000))  echo "${timeStamp} ${mylist[\*]}" >> /var/log/windpower/windpower.log  }  while [ 1 ]  do  random\_double  sleep $step  done |

### 4.2 Flume采集数据

**基本配置：**（配置路径/etc/flume/conf/flume\_kafka.conf）

|  |  |
| --- | --- |
| 数据源 | exec |
| 输入项 | /var/log/windpower/windpower.log |
| 输出项 | kafka.topic |

**配置文件：**

|  |
| --- |
| a1.sources = r1  a1.sinks = k1  a1.channels = c1  # Describe/configure the source  a1.sources.r1.type = exec  a1.sources.r1.command = tail -F /var/log/windpower/windpower.log  # Describe the sink  #a1.sinks.k1.type = logger  a1.sinks.k1.type = org.apache.flume.sink.kafka.KafkaSink  a1.sinks.k1.topic = mastertest  a1.sinks.k1.brokerList = master:9092,slave1:9092,slave2:9092  a1.sinks.k1.requiredAcks = 1  a1.sinks.k1.batchSize = 20  # Use a channel which buffers events in memory  a1.channels.c1.type = memory  a1.channels.c1.capacity = 1000  a1.channels.c1.transactionCapacity = 100  # Bind the source and sink to the channel  a1.sources.r1.channels = c1  a1.sinks.k1.channel = c1 |

**启动命令：**

|  |
| --- |
| flume-ng agent -Dflume.root.logger=DEBUG.console \  -Dorg.apache.flume.log.printconfig=true \  --conf /etc/flume/conf/ \  --conf-file /etc/flume/conf/flume\_kafka.conf \  --name a1 |

### 4.3 Kafka缓存数据

**基本配置：**

|  |  |
| --- | --- |
| 数据源 | flume |
| 输入项 | kafka.topic |
| 输出项 | kafka.topic |

**创建topic并进行分区实现缓存**

|  |
| --- |
| bin/kafka-topics.sh --create --zookeeper master:2181,slave1:2181,slave2:2181 --replication-factor 1 --partitions 4 --topic mastertest |

### 4.4 Flink处理数据

**基本配置：**

|  |  |
| --- | --- |
| 数据源 | Kafka.topic |
| 输入项 | kafka.topic |
| 输出项 | opentsdb |

**处理代码：**

|  |
| --- |
| */\*\*  \** ***@author*** *createdBy liuxiaojun editBy wangzhijian and wugenqiang  \** ***@Title:*** *ReadingFromKafka  \** ***@ProjectName*** *flink\_opentsdb  \** ***@Description:*** *flink读取kafka数据  \** ***@date*** *2018/8/29 10:29  \*/* **public class** ReadingFromKafka {  */\*\*  　　\* @Description: flink读取kafka数据  　　\* @param []  　　\* @return void  　　\* @throws  　　\* @author wugenqiang  　　\* @date 2018/8/29 10:33  　　\*/* **public void** readingFromKafka(){  **final** String ZOOKEEPER\_HOST = **"master:2181,slave1:2181,slave2:2181"**;  **final** String KAFKA\_HOST = **"master:9092,slave1:9092,slave2:9092"**;  **final** StreamExecutionEnvironment env = StreamExecutionEnvironment.*getExecutionEnvironment*();  env.enableCheckpointing(1000); *// 非常关键，一定要设置启动检查点！！* env.setStreamTimeCharacteristic(TimeCharacteristic.***EventTime***);  Properties props = **new** Properties();  props.setProperty(**"zookeeper.connect"**, ZOOKEEPER\_HOST);  props.setProperty(**"bootstrap.servers"**, KAFKA\_HOST);  props.setProperty(**"group.id"**, **"transaction"**);  DataStream<String> transction = env.addSource(**new** FlinkKafkaConsumer010<String>(**"mastertest"**, **new** SimpleStringSchema(), props));  transction.rebalance().map(**new** MapFunction<String, Object>() {  **public** String map(String value)**throws** IOException {  */\*//此处处理一条数据仅有一个数据项  WriteIntoOpentsdb writeIntoOpentsdb = new WriteIntoOpentsdb();  writeIntoOpentsdb.writeIntoOpentsdb(value);\*/  //此处处理一条数据中的多个数据项* String result[]=value.split(**" "**);  WriteIntoOpentsdb writeIntoOpentsdb = **new** WriteIntoOpentsdb();  writeIntoOpentsdb.writeIntoOpentsdb(result);  **return** value;  }  }).print();  **try** {  env.execute();  } **catch** (Exception ex) { Logger.*getLogger*(ReadingFromKafka.**class**.getName()).log(Level.***SEVERE***, **null**, ex);  ex.printStackTrace();  }  } } |
| **public class** Main {  **public static void** main(String[] args) {  ReadingFromKafka readingFromKafka = **new** ReadingFromKafka();  *//调用readingFromKafka()实现读取kafka数据* readingFromKafka.readingFromKafka();  } } |

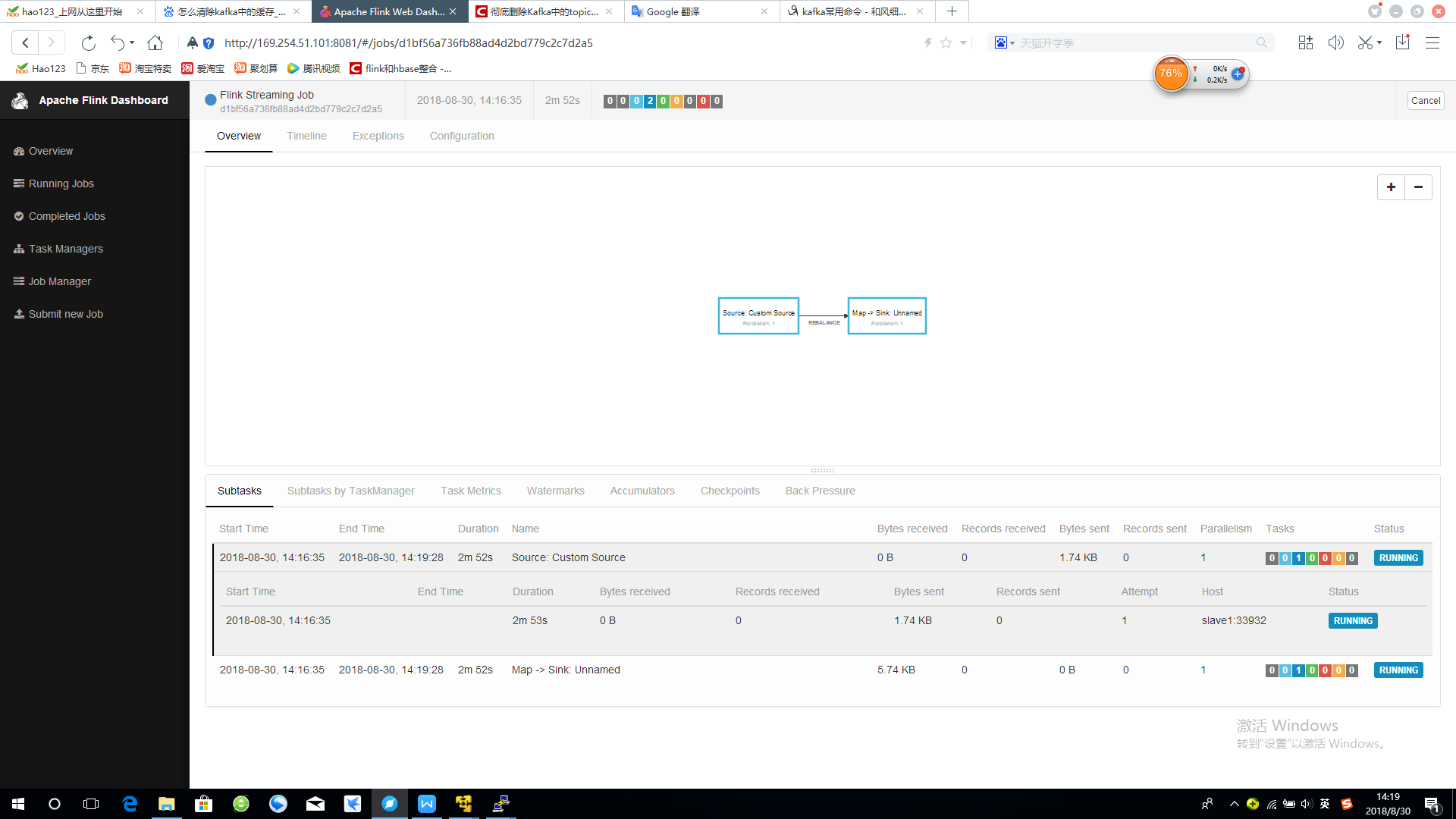
### 

### 4.5 OpenTSDB存储数据

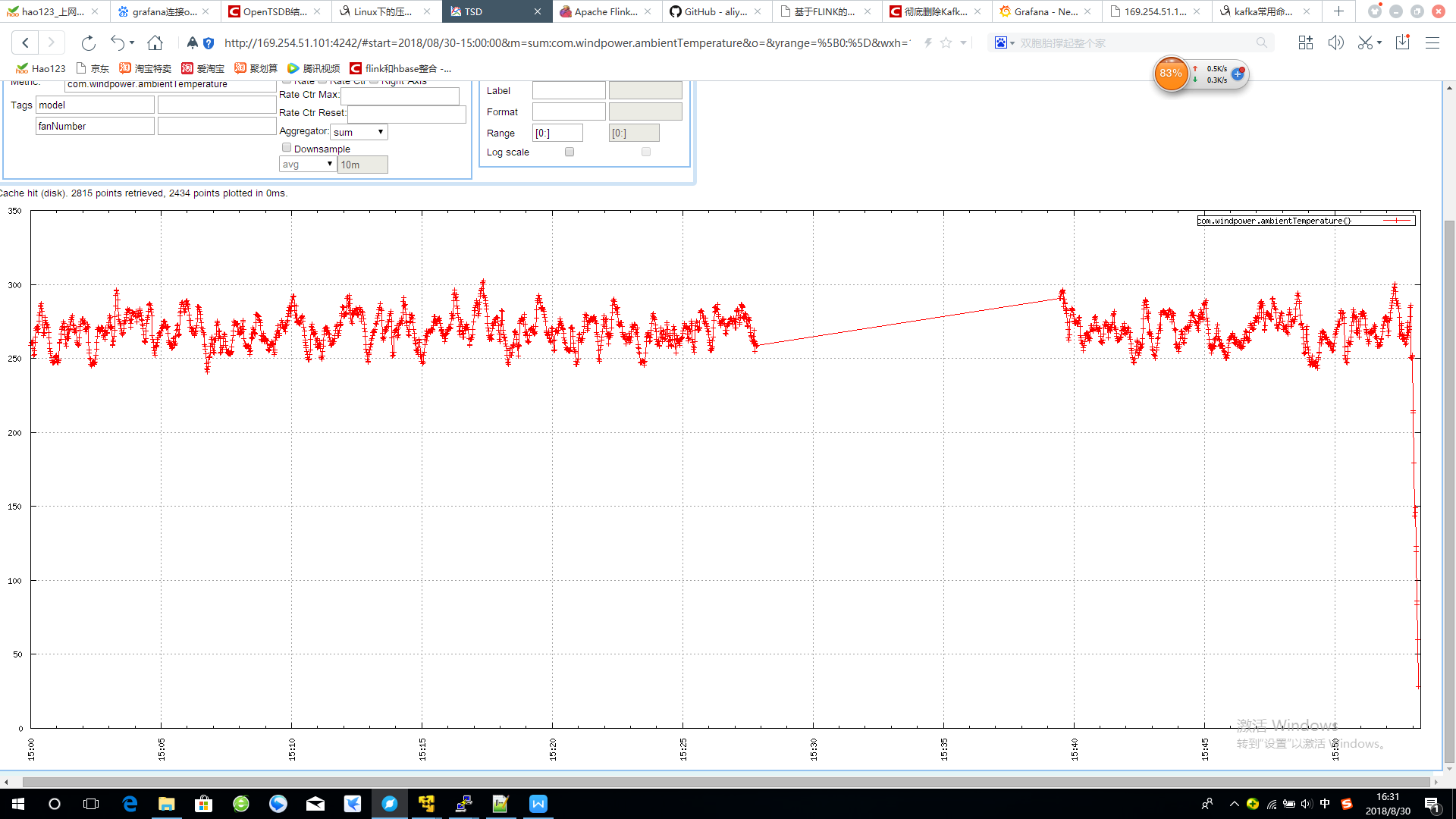
|  |
| --- |
| */\*\*  \** ***@author*** *wugenqiang  \** ***@Title:*** *WriteIntoOpentsdb  \** ***@ProjectName*** *flink\_opentsdb  \** ***@Description:*** *监听数据写入opentsdb  \** ***@date*** *2018/8/29 10:35  \*/* **public class** WriteIntoOpentsdb { */\*\*  　　\* @Description: flink读取kafka数据，此处处理一条数据中的多个数据项  　　\* @param [value]  　　\* @return void  　　\* @throws  　　\* @author createdBy liuxiaojun editBy wugenqiang  　　\* @date 2018/8/29 12:42  　　\*/* **public void** writeIntoOpentsdb(String[] value) {  HttpClientImpl client = **new** HttpClientImpl(**"http://master:4242"**);  MetricBuilder builder = MetricBuilder.*getInstance*();  *//时间戳* **long** timeStramp = Long.*parseLong*(value[0]);  *//风机编号* String fanNumber = value[1];  *//机型编号* String model = value[2];  *//环境温度* Double ambientTemperature =Double.*valueOf*(value[3]);  *//机身温度* Double fuselageTemperature =Double.*valueOf*(value[4]);  *//风速* Double windSpeed =Double.*valueOf*(value[5]);  *//发电量* Double powerGeneration =Double.*valueOf*(value[6]);  builder.addMetric(**"com.windpower.ambientTemperature"**).setDataPoint(timeStramp,ambientTemperature)  .addTag(**"fanNumber"**, fanNumber).addTag(**"model"**, model);  builder.addMetric(**"com.windpower.fuselageTemperature"**).setDataPoint(timeStramp,fuselageTemperature)  .addTag(**"fanNumber"**, fanNumber).addTag(**"model"**, model);  builder.addMetric(**"com.windpower.windSpeed"**).setDataPoint(timeStramp,windSpeed)  .addTag(**"fanNumber"**, fanNumber).addTag(**"model"**, model);  builder.addMetric(**"com.windpower.powerGeneration"**).setDataPoint(timeStramp,powerGeneration)  .addTag(**"fanNumber"**, fanNumber).addTag(**"model"**, model);  **try** {  Response response = client.pushMetrics(builder,  ExpectResponse.***SUMMARY***);  System.***out***.println(response);  } **catch** (IOException e) {  e.printStackTrace();  }  } } |

1. 实验结果与分析

Flink处理数据图：



Opentsdb接收到数据图：



访问Grafana WEB页面结合OpenTSDB进行视图展现

默认端口：3000

假设是在本机部署，默认是安装在3000端口，在浏览器打开3000端口。



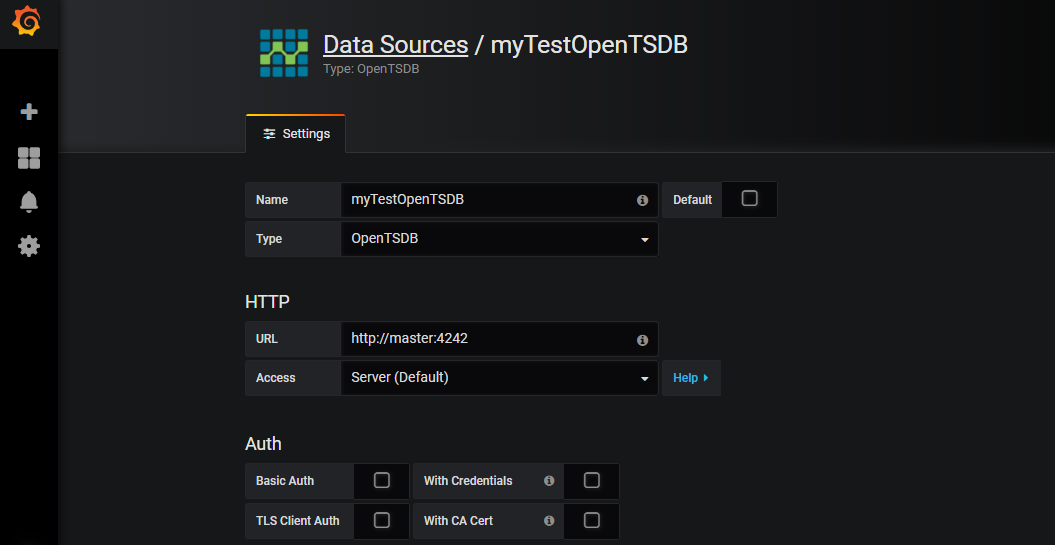
登录名密码默认admin

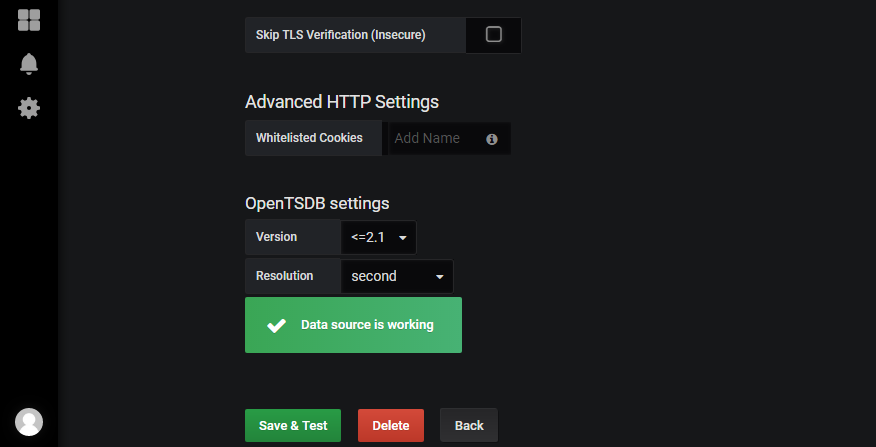
我修改的自己密码123456

1. 为grafana配置数据源为opentsdb

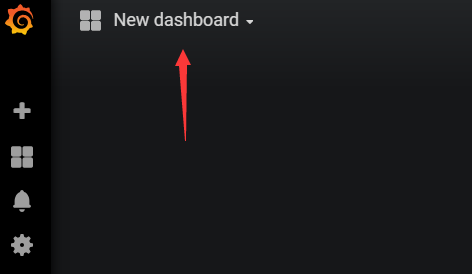
grafana默认admin账号

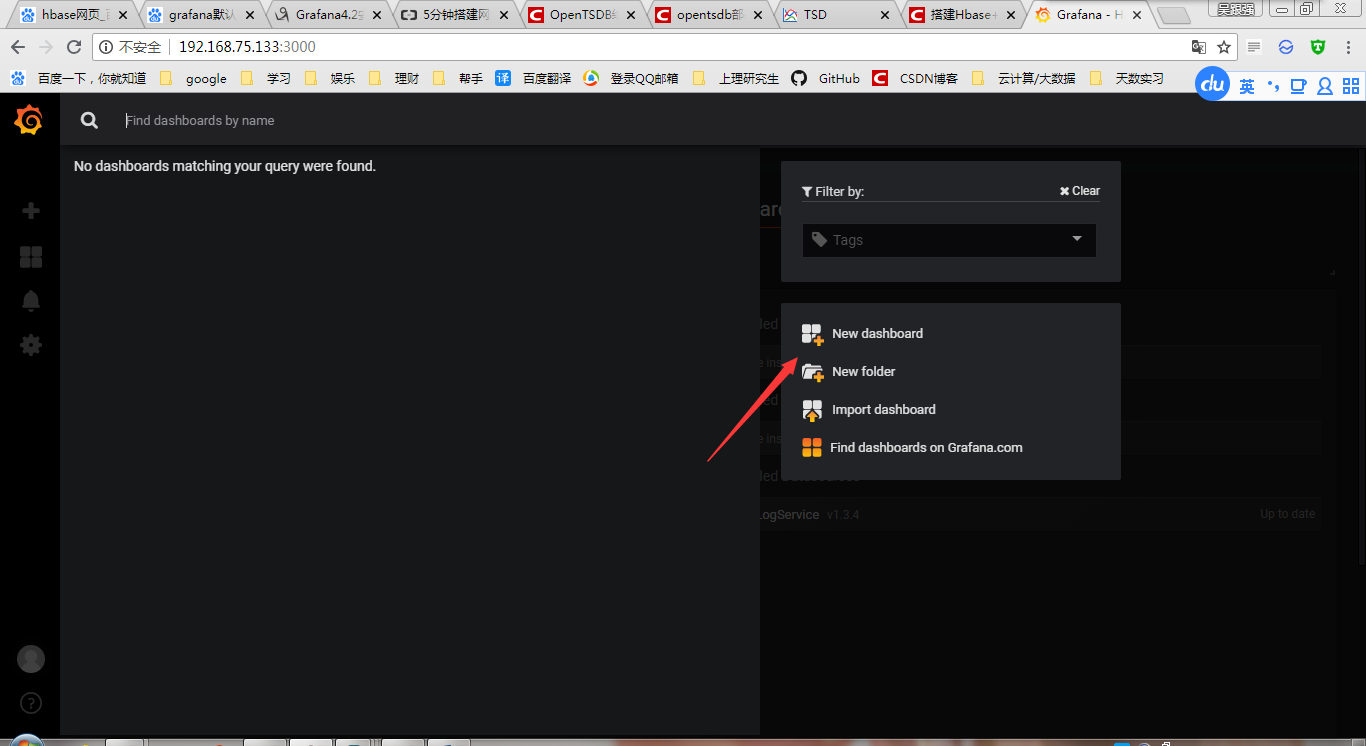
点击DataSource > add DataSource.填写相关的信息

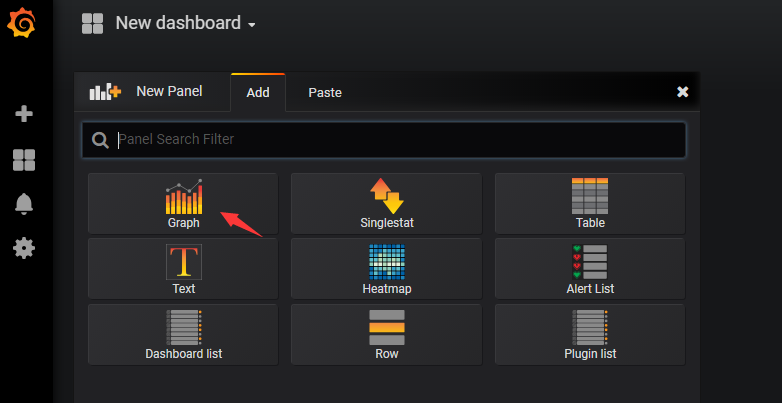




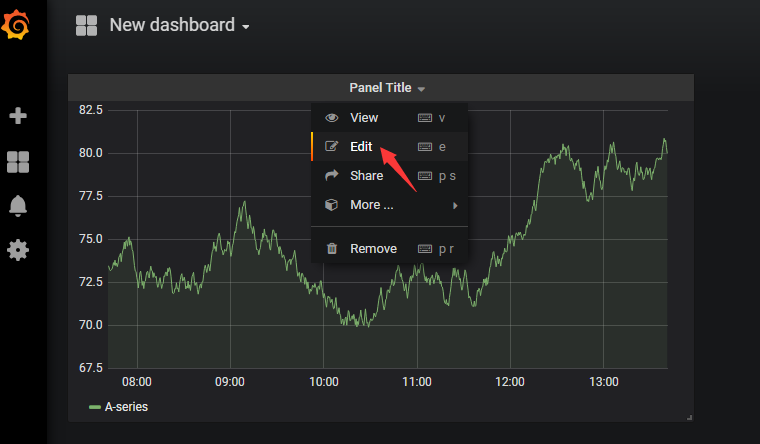
1. 创建个图表,并配置告警



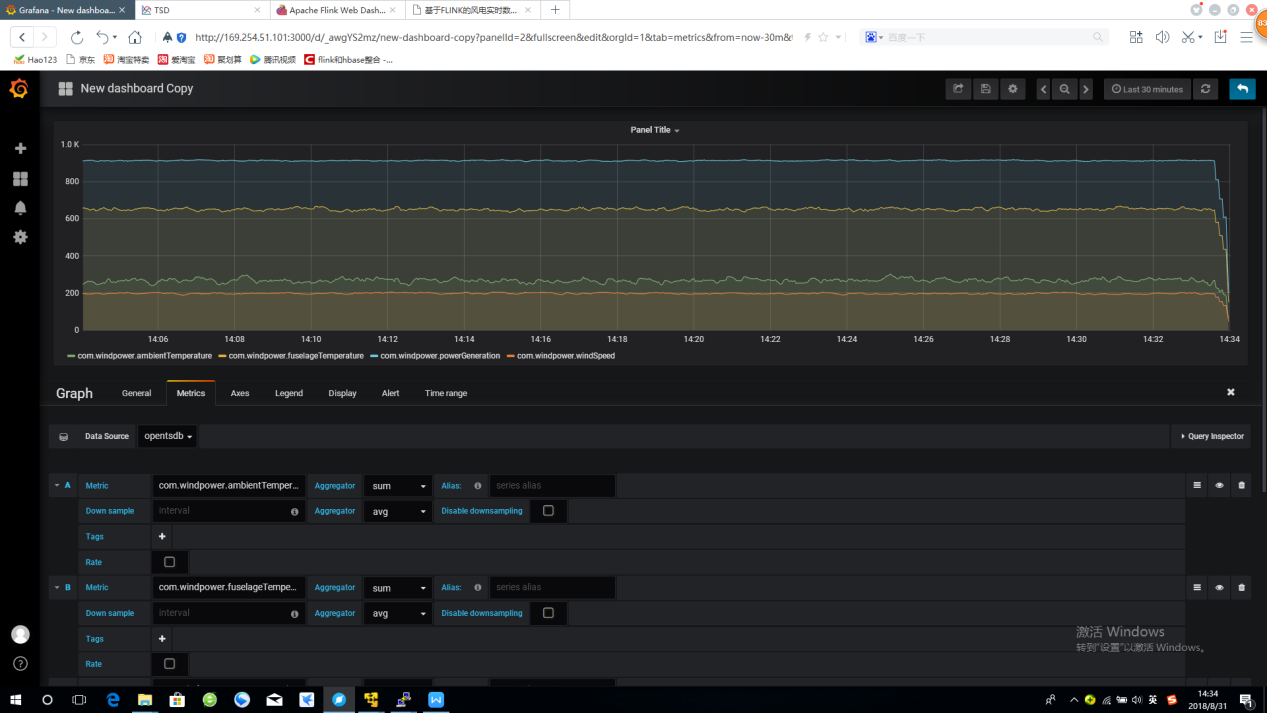




Grafana显示数据图：



Grafana显示数据整合图：



六、实验心得

通过这段时间的毕业实习，我们组终于完成了项目的设计，系统在设计方面具有可优化功能，这个系统设计不是很复杂，但是要优化其实并不容易，在我们的项目中，我被分到的模块是数据进入opentsdb并在grafana显示，在这个过程中，我学习到了很多东西，同时也在老师身边学习到了开发系统的一些技巧和方法，对我以后的学习给予了很大帮助，以前只是喜欢程序，学习了不少语言，但是，我还真没有系统的团队协作做过项目，这个过程则是理论和实践的有机结合，为我以后的发展提供了很大的帮助。

希望以后有更多这样的机会来提升自己，加油！