学习笔记

**——部 门：CRM产品部**

**——导 师：丰景军**

**——汇报人：郑宇航**

## 2016年7月

### 第一周（7.26-7.29）

#### 周二（7月26日）

1. **学习目的：签相关合同文件、配置公司办公环境**
2. **学习方式：查看相关资料文档、门户网站、请教导师和助理**
3. **学习内容：配置路由、安装易信PC端与移动端、安装Foxmail、安装VPN，由于本人使用的OS是win10-64位版,所以一开始安装连接失败，需要在门户网站**[**http://oa.si-tech.com.cn/**](http://oa.si-tech.com.cn/)**下载兼容版本、安装notes，新人入职引导http://eip.teamshub.com/t/3124200，参加会议了解实时营销相关业务**
4. **存在疑惑：刚入职公司，对办公环境比较陌生，对门户网站不了解**
5. **收获体会：对常使用的软件、实时营销业务有初步了解**
6. **导师回复：**

#### 周三（7月27日）

1. **学习目的：熟悉软件使用、熟悉门户网站常用功能**
2. **学习方式：查看相关资料文档、门户网站、请教导师和助理**
3. **学习内容：了解每日签到签退注意事项、考勤与请假相关事项、每日学习笔记的填写与发帖、参加会议学习实时营销项目的结构与业务流程、了解项目的部署环境**
4. **存在疑惑：Linux相关知识需要学习、暂时还没涉及到开发**
5. **收获体会：熟悉每天必做事项、优化了上班路线**
6. **导师回复：**

#### 周四（7月28日）

1. **学习目的：熟悉开发环境、熟悉实时营销业务逻辑**
2. **学习方式：请教导师和助理**
3. **学习内容：从SVN检查出项目源码，建立本地数据库与导入数据，本地部署项目，由于编码不统一做了一些调试与配置**

**1）：mysql->my.ini 文件添加配置:**

**[client]**

**default-character-set=utf8**

**[mysql]**

**default-character-set=utf8**

**[mysqld]**

**character-set-server=utf8**

**2）：Eclipse编码配置：**

**General->Workspace->Text file encoding下设置成：GBK**

**3）：导出的项目设置成UTF-8：**

**右键->Resource-> Text file encoding下设置成：UTF-8**

**结论：数据库可以不设置编码格式，但是Eclipse如果设置不正确查询出来的数据会出现乱码，导致无法转换Json格式的数据而抛出异常。**

**成功部署后查看源码，熟悉项目结构与实现细节以准备开发工作。**

1. **存在疑惑：对项目结构还需要深入了解**
2. **收获体会：编码是常见的问题耐心调试**
3. **导师回复：**

#### 周五（7月29日）

1. **学习目的：熟悉服务器部署结构、熟悉实时营销项目**
2. **学习方式：导师讲解、查看项目文档**
3. **学习内容：听导师讲解项目部署环境、分布式服务器各自职责、实时营销项目流程与各个模块的功能、学习相关shell命令，**

**查看项目文档，熟悉项目结构**

[**http://training.si-tech.com.cn/portal/courseLibrary.jsp**](http://training.si-tech.com.cn/portal/courseLibrary.jsp)**学习课程（**市场营销产品文档、员工职业道德培训资料、养成好习惯、目标管理**）**

**学习项目源码、熟悉webmarket、gdlt\_appmarket、realtimemarket在整个项目中各自的功能、熟悉各层的调用关系**

**1）：流数据处理结构，从kafka获得输入数据，由ESP执行相应的处理再输出数据到kafka，kafka负责消息的汇总与推送**

**2）：shell调用realtimemarket中的Java程序，realtimemarket为ESP提供服务，realtimemarket连接了前端交互；**

1. **存在疑惑：不太熟悉ESP、kafka各自的实际作用**
2. **收获体会：对项目结构、各个组件、服务器环境有初步了解**
3. **导师回复：**

## 2016年8月

### 第二周（8.1-8.5）

#### 周一（8月1日）

1. **学习目的：编写测试添加Demo,** **学习kafka读写信息类**
2. **学习方式：同事讨论、请教导师**
3. **学习内容：编写Demo:**
4. **修改webmarket/com.siteh.market的jsp页面**
5. **webmarket/xwork下对应的xml配置，这里遇到一些问题：修改xml时不会编译到WebRoot\WEB-INF\classes目录下，导致修改不会发生作用，原因可能是项目的红叉太多，编译时无法忽略错误，消灭所有的红叉可能可以解决问题，但是不靠谱，临时解决方案：右键->**build path → Source → Ignore optional compile problem →ok
6. **Action关键代码**

MBean bean = **new** MBean();

bean.setBody("name", name);

String bodybean = bean.toString();

String url = Const.***ROOT\_PATH*** + SvcConst.***AtomCommonSvc\_testAdd***;

String info = SitechClient.*callService*(url, bodybean);

**其中**MBean**作为数据载体，再度向后台传递参数,枚举类**SvcConst**设置后台路径映射，内容格式：com\_sitech\_market\_atom\_common\_inter\_ITestAddSvc\_testAdd**

1. **编写TestAdd类，进行对应的数据库操作实体类参数**InDTO**转换为**Map: Map<Object,Object> inMap=atomGetLoginMsgInParam.parseInParam(inDTO);

**Kafka：是一种高吞吐量的分布式发布订阅消息系统，**

**ParseFlowFileToKafka：负责读取配置文件内容，转换为Kafka流数据数据**

**ReadFromKafkaToEsp：从Kafka读入数据，然后由Esp进行发布**

1. **存在疑惑：Eclipse自动编译问题没有完美解决，对kafka、Esp的具体实现还不够了解**
2. **收获体会：熟悉项目的调用流程、初步了解Kafka与Esp的协调关系**
3. **导师回复：**

#### 周二（8月2日）

1. **学习目的：编写流处理平台操作手册，学习流处理项目源码**
2. **学习方式：同事讨论、请教导师**
3. **学习内容：**

**①编写流处理用户手册流量累计部分**

**②查看dbo层项目源码，了解实体类的配置，由于在家里办公果然是有些不方便，到第二天才能向同事请教，并且完成添加demo的编写。**

1. **存在疑惑：basedao、Mbean没有源码，dbo层的配置稍微有点多**
2. **收获体会：初步了解项目数据访问层结构**
3. **导师回复：**

#### 周三（8月3日）

1. **学习目的：完成添加demo，学习流处理项目源码**
2. **学习方式：同事讨论、请教导师**
3. **学习内容：**

**1)添加demo服务层：**

**①com.sitech.market.atom.common.impl，创建服务类：**

@ParamTypes({

@ParamType(c = StandardInDTO.**class**, m = "testAdd",oc=StandardOutDTO.**class**)})

**public** **class** TestAdd **implements** ITestAdd {

**private** ITestAddEntity testAddEntity;

@Override

**public** OutDTO testAdd(InDTO inDTO) {

System.***out***.println("run-to:---testAdd");

StandardOutDTO ot = **new** StandardOutDTO();

MBean bean = inDTO.getMbean();

MBean resultBean = **new** MBean();

Object o = testAddEntity.insertMkTestAddInfo(bean);

resultBean.setBody("obj", o);

ot.setBean(resultBean);

**return** ot;

}

**（get/set方法）**

}

**②com.sitech.market.busiobject.entity.busi.impl，Entity类：**

**public** **class** TestAddEntity **implements** ITestAddEntity {

**private** IMKTestAddInfo mkTestAddInfo;

@Override

**public** Boolean insertMkTestAddInfo(MBean bean) {

Map<String,Object> map = **new** HashMap<String, Object>();

map.put("name", bean.getBodyStr("name"));

map.put("status", bean.getBodyStr("status"));

map.put("from\_", bean.getBodyStr("from\_"));

map.put("to\_", bean.getBodyStr("to\_"));

**return** mkTestAddInfo.add(map);

}

@Override

**public** MBean deleteEvent(MBean bean) {

MBean rbean = **new** MBean();

**try** {

mkTestAddInfo.deleteById(bean.getBodyStr("id"));

rbean.setBody("message", "测试删除成功");

} **catch** (Exception e) {

e.printStackTrace();

rbean.setBody("message", "测试删除失败");

}

**return** rbean;

}

**（get/set方法）**}

**③com.sitech.market.dbo.destribution.imp,新建dbo:**

**public** **class** MKTestAddInfo **extends** DBMysqlClass **implements** IMKTestAddInfo {

@Override

**public** **boolean** add(Map<String, Object> mkTestAddInfo) {

**try** {

baseDao.insert("MkTestAddInfo.insertMkTestAddInfo", mkTestAddInfo);

**return** **true**;

} **catch** (RuntimeException e) {

**throw** **new** BusiException("1000",e.getMessage());

}

}

}

**④/gdlt\_appmarket/src/ibatismap，配置MkTestAddInfo.xml**

**内容为格式化的sql语句**

**⑤/gdlt\_appmarket/src/springConfig，配置xml:**

dbo.xml: **配置映射dbo路径**

<bean id=*"mkTestAddInfo"* class=*"com.sitech.market.dbo.destribution.impl.MKTestAddInfo"* parent=*"dbBase"* />

entity.xml:**配置实体，引用**dbo.xml**中的***mkTestAddInfo*

<bean id=*"testAddEntity"* class=*"com.sitech.market.busiobject.entity.busi.impl.TestAddEntity"*>

<property name=*"mkTestAddInfo"* ref=*"mkTestAddInfo"*/>

</bean>

service.xml:**配置服务，引用**entity.xml**中的***testAddEntity*

<bean id=*"testAddSvc"* class=*"com.sitech.market.atom.common.impl.TestAdd"* >

<property name=*"testAddEntity"* ref=*"testAddEntity"*></property></bean>

**注意id名称加后缀+**Svc

1. **学习Esp+Kafka：**
   1. **ReadFromKafkaToEsp：**

*ini* = **new** Ini(Thread.*c*urrentThread()

.getContextClassLoader()

.getResourceAsStream("esp\_stream.cfg"));

读取Esp流配置文件，然后根据主题和标签匹配需要的数据->

**new** ReadFromKafkaToEsp(topicName).run();

主要执行以下：

Project project = EspHelper.*getFlowProject*(***s\_sdk***, espUrl);

KafkaStream<**byte**[], **byte**[]> kafkaStream = consumerMap.get(consumeTopic).get(0);

//根据消费主题生成kafka流

publisher.connect();//发布器连接

**对kafka流进行一些操作转换成可发布的message**

publisher.publish(message);

//发布消息，关闭资源

* 1. **ParsePreFlowFileToKafka：**

ConfigUtil.*getConfig*("pre.flow.file.dir");

//从路径下读取文件

BufferedReader buffered=**new** BufferedReader(reader);

//转换为字节流

message = buffered.readLine();

messageArrList = dealLineMessage(message);

//逐行读取，根据规则处理后，发送record

ProducerRecord<String, String> record =

**new** ProducerRecord<String, String>(messageArr[0], partition, line + "", messageArr[1]);

***producer***.send(record);

1. **存在疑惑：对一些方法的处理作用还不太了解**
2. **收获体会：熟悉kafka、esp数据流程**
3. **导师回复：**

#### 周四（8月4日）

1. **学习目的：编写Kafka测试类、移植代码到mvn**
2. **学习方式：同事讨论、请教导师**
3. **学习内容：** 
   1. **编写和运行测试类，只是能成功运行，还需要继续熟悉**
   2. **搭建mvn环境，基本没什么困扰**

**小技巧：设置svn忽略、（项目里设置不起作用）**

**Widows->preference->team->Ignore->Add (.settings/\*、\*/target/\*)…等等**

**时间策划、系统监控模块：**

**从jsp页面查看有哪些访问路径->**

**对于的fxxxx.xml配置文件的方法->**

**在对应路径下的Action里查找被访问的方法->**

**Appmarket项目里查找对应的服务->**

**修改类头部注解-** **为标准StandardInDTO**

@ParamType(c = StandardInDTO.**class**,

m = "method-name",oc=StandardOutDTO.**class**)

**修改方法—模板：**

**public** OutDTO saveLaccis(InDTO inparam) {

StandardOutDTO ot = **new** StandardOutDTO();

MBean resultBean = **new** MBean();

Boolean o = eventEntity.insertMkMylacciInfo(inparam.getMbean());

resultBean.setBody("returnCode", o);

ot.setBean(resultBean);

**return** ot;

}

**重复步骤，补全了一些遗漏的文件**

1. **存在疑惑：Kafka类还有些不明白的流程**
2. **收获体会：基本掌握实时营销项目的数据流程**
3. **导师回复：**

#### 周五（8月5日）

1. **学习目的：移植代码到mvn、处理乱码文件**
2. **学习方式：同事讨论**
3. **学习内容：**

**①将复杂的数据操作流程合并到Service层，直接在service层调用dbo，主要是跟着代码走流程，检查有什么遗漏的添加上**

**一开始找不到实现类：shift+ctrl+R检索资源**

**②中文乱码，从旧项目逐个方法复制到新项目；**

**总的来说并没有什么复杂度，就是项目结构比较乱，文件比较难找，注意decode方法会在get的时候处理参数：**

**public** **void** decode(MBean bean){

setBeginNum(bean.getStr(getPathByProperName("beginNum")));

setEndNum(bean.getStr(getPathByProperName("endNum")));

}

**注意**paramBean.getBodyStr("loginNo")**与以上get方法的对应**

1. **存在疑惑：项目结构有点乱，各个包放的类功能不清楚**
2. **收获体会：Service层设计变得更加精简、引用的属性变得更少**
3. **导师回复：**

### 第三周（8.8-8.12）

#### 周一（8月8日）

1. **学习目的：移植文件修改编码**
2. **学习方式：同事讨论**
3. **学习内容：**

**修改文件编码为：utf-8，测试发现了几个Bug:**

* 1. **日期选择无效：原因是在mvn环境下路径发生改变，设置正确导入**
  2. **点击系统监控模块经常会重启tomcat：内存分配不够，**

**Run-> run configurations->arguments下添加一行：**

**-Xms512m -Xmx1024m -XX:MaxNewSize=1024m -XX:MaxPermSize=1024m**

* 1. **参数乱码：url路径传参，跟分页有关，需避免二次转码**

**new** String(eventName.getBytes("iso-8859-1"),"UTF-8");

1. **存在疑惑：分页中参数乱码仍未完美解决，分页功能需要学习**
2. **收获体会：编码要统一，编码解码要对应**
3. **导师回复：**

#### 周二（8月9日）

1. **学习目的：URL编码与解码**
2. **学习方式：论坛博客、同事讨论**
3. **学习内容：**

**项目出现的问题：分页使用的是get请求，默认的编码格式是**

iso-8859-1**在页面需要用一下语句解码**

**new** String(eventName.getBytes("iso-8859-1"),"UTF-8");

**然而这种形式其实只是对取出来的参数进行了一次解码，request中的参数仍然是未解码前的字符，在第二次提request请求时，服务器又对request中的参数进行了一次**iso-8859-1**编码，第二次编码，第二次编码，第二次编码（重要的事说三遍），所以再取出时要进行两次解码才能装换成中文，也就是：**

**new** String((**new** String(eventName.getBytes("iso-8859-1"),"UTF-8")).getBytes("iso-8859-1"),"UTF-8"))

**所以问题是---->不能继续使用这种解码方式**

**想到的解决方案：**

* 1. **取出中文后再次设置request参数**

**但是request没有setParameter☺**

* 1. **改造pagelist分页标签**

**我们使用的是编译后的文件，没有源码☺**

* 1. **修改服务器编码**

<Connector connectionTimeout=*"20000"* port=*"8080"* protocol=*"HTTP/1.1"* redirectPort=*"8443"* URIEncoding=*"utf-8"*/>

**这种解决方式，配合filter设置**httpRequest.setCharacterEncoding("UTF-8");

**确实可以轻松解决乱码，但是问题又来了，换服务器，那里的服务器需要修改编码方式，因为tomcat默认的是**iso-8859-1☺

**总结：希望问题可以得到完美解决，或许可以对pagelist做一些修改，使用js对参数进行重新赋值**

**学习了\_**

URLEncoder.*encode*(str, "UTF-8");

URLDecoder.*decode*(str, "UTF-8");

浏览器(ie firefox)【get/post】------------>Servlet服务器----------->浏览器显示  
                编码                 解码成unicode，然后将显示的内容编码        解码

**链接：**

**关于GET和POST请求的编解码问题：**

[**http://blog.csdn.net/miderph84/article/details/3420668**](http://blog.csdn.net/miderph84/article/details/3420668)

**使用encode()、decode()解决中文乱码问题：**

[**http://blog.csdn.net/xiaoyu714543065/article/details/13995369**](http://blog.csdn.net/xiaoyu714543065/article/details/13995369)

**GET和POST的真正区别：**

[**http://www.nowamagic.net/librarys/veda/detail/1919**](http://www.nowamagic.net/librarys/veda/detail/1919)

**URL编码与解码：**

[**http://www.cnblogs.com/greatverve/archive/2011/12/12/URL-Encoding-Decoding.html**](http://www.cnblogs.com/greatverve/archive/2011/12/12/URL-Encoding-Decoding.html)

1. **存在疑惑：**
2. **收获体会：越来越不喜欢中文了☺**
3. **导师回复：**

#### 周三（8月10日）

1. **学习目的：熟悉Kafka**
2. **学习方式：论坛博客、同事讨论**
3. **学习内容：**

**学习Kafka教程，**

**Kafka将消息以topic为单位进行归纳。将向Kafka topic发布消息的程序成为producers。将预订topics并消费消息的程序成为consumer。Kafka以集群的方式运行，可以由一个或多个服务组成，每个服务叫做一个broker.**

* 1. **搭建Kafka运行环境：**

**Step 1: 下载Kafka**

> tar -xzf kafka\_2.9.2-0.8.1.1.tgz

> cd kafka\_2.9.2-0.8.1.1

**Step 2: 启动服务**

> **bin/zookeeper-server-start.sh config/zookeeper.properties &**

> **bin/kafka-server-start.sh config/server.properties**

**Step 3: 创建 topic**

> **bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic t**

> **bin/kafka-topics.sh --list --zookeeper localhost:2181**

test

**Step 4:发送消息**

> **bin/kafka-console-producer.sh --broker-list localhost:9092 --topic test**

**This is a messageThis is another message**

ctrl+c可以退出发送。

**Step 5: 启动consumer**

> **bin/kafka-console-consumer.sh --zookeeper localhost:2181 --topic test --from-beginning**

This is a message

This is another message

**Step 6: 搭建一个多个broker的集群**

> cp config/server.properties config/server-1.properties

> cp config/server.properties config/server-2.properties

**搭建Kafka开发环境->数据持久化->消息的事物定义->性能优化->Producer和Consumer->主从同步->客户端API->消息和日志**

**学习了Kafka的入门教程，熟悉Kafka的流程**

**了解Linux系统的安装，下一步安装ubunto，并搭建Kafka服务器与客户端开发环境**

**Kafka使用入门教程链接：**

[**http://www.linuxidc.com/Linux/2014-07/104470.htm**](http://www.linuxidc.com/Linux/2014-07/104470.htm)

1. **存在疑惑：Kafka如何与Esp合体**
2. **收获体会：系统的了解Kafka的流程**
3. **导师回复：**

#### 周四（8月11日）

1. **学习目的：事件规则配置界面修改**
2. **学习方式：同事讨论、请教导师**
3. **学习内容：**
   1. **添加事件各个界面样式风格修改，**

**涉及的文件包括：**

**webapp\n\_ui下的js文件、css、images图标文件，直接覆盖，出现了一些样式丢失的问题**

**webapp\n\_page\market\destribution下的jsp文件，**

**主要add\_basic.jsp，更改样式显示风格，js脚本导入与修改，整合rule\_add.jsp、ajax\_custgroup.jsp、ajax\_dataalert.jsp、ajax\_location.jsp、ajax\_realdpi.jsp、ajax\_realmessage.jsp、ajax\_sendmsg.jsp，由于之前内容是分散的，需要整合在同一个jsp下，参数的格式与函数的调用关系略微复杂，仍需要些时间整理**

* 1. **在虚拟机中安装了ubuntu**

1. **存在疑惑：参数的拼接规则，登录样式丢失**
2. **收获体会：熟悉添加事件配置流程**
3. **导师回复：**

#### 周五（8月12日）

1. **学习目的：事件规则配置界面修改**
2. **学习方式：同事讨论、请教导师**
3. **学习内容：**
   1. **后台整合saveInit+saveImageGraph，统一参数，涉及到的层级有Action/Service/Entity**
   2. **反馈与整理样式，add\_basic.jsp、rule\_add.jsp，移植营销元素异步加载显示**
   3. $(".element ").delegate("a","click",**function**(){}

**Delegate监听事件，监听元素点击触发，可用于新增的元素**

1. **存在疑惑：页面的营销元素比较多、参数规则略微复杂**
2. **收获体会：熟悉事件配置流程**
3. **导师回复：**

### 第四周（8.15-8.19）

#### 周一（8月15日）

1. **学习目的：整合修改事件定义页面**
2. **学习方式：同事讨论、请教导师**
3. **学习内容：**
   1. **主要页面导入，新脚本/样式文件缺少旧文件的一些样式：**

<%@ include file=*"/n\_page/include/new\_header.jsp"*%>

<%@ include file=*"/n\_page/market/destribution/add\_include.jsp"*%>

<%@ include file=*"/n\_page/include/new\_footer.jsp"*%>

图1，保存成功是的提示样式

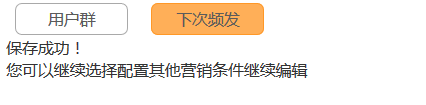


图2，上网内容选择框和编辑图标



图3，预览内容超出时按钮异常

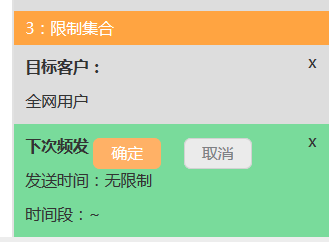
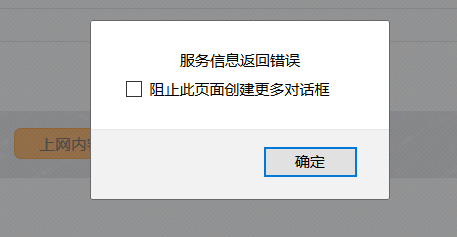


图4，遗留bug点击保存后再点击元素按钮弹出信息



* 1. **修改了ajax获取页面的，修改了函数存在问题如：**

**判断条件没有覆盖undefined的情况，url == null 没有return**

**今天主要就是调整样式与理清楚规则元素保存时参数的去向，并且把参数显示到右侧预览区，还剩下保存后初始化元素、from表单提交、一些样式问题**

1. **存在疑惑：表单参数定义规则，js+java+自定义taglib都写在jsp里理解比较吃力**
2. **收获体会：trigger要放在相关加载函数的最下面**
3. **导师回复：**

#### 周二（8月16日）

1. **学习目的：编辑初始化**
2. **学习方式：请教导师**
3. **学习内容：**
   1. **保存后再次进入时初始化页面，由于是异步加载，传参如下：**

**var** packet = **new** AJAXPacket(url,"wait...");

par = window.encodeURI(window.encodeURI(par));

packet.data.add("par" ,par);

core.ajax.sendPacketHtml(packet,ajax\_callback\_func,**true**);

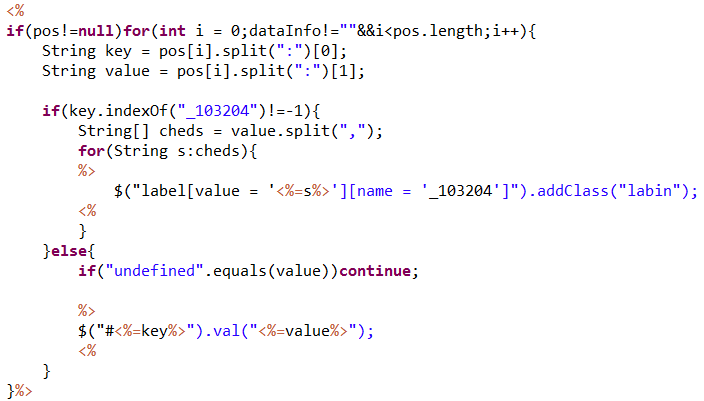
**参数par经过编码，页面获取参数时需要二次解码：**

java.net.URLDecoder

.decode(java.net.URLDecoder.decode(dataInfo, "utf-8"),"utf-8");

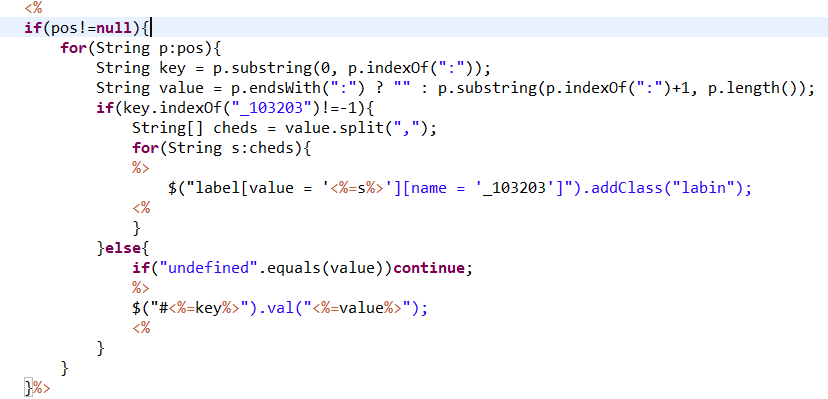
**由于格式是字符串，处理后才获得各个参数，然后初始化页面：**

代码格式不好看，上截图好了：



**若是参数有空字符，这里会出现一些bug，**

**String value = temp[1];这句会数组越界，做了些改进：**



* 1. **调整了一些样式，以及弄清楚各个参数的定义规则、初始化，整合baseInfo格式**

1. **存在疑惑：异步加载如何让页面“刷新”效果消失**
2. **收获体会：在java和js之间轮换编码是不是一种收获0.0**
3. **导师回复：**

#### 周三（8月17日）

1. **学习目的：事件定义初始化修改**
2. **学习方式：请教导师**
3. **学习内容：**
   1. **点击更新事件配置按钮，初始化数据赋值，样式调整**
   2. **继续处理了一些参数不完整数组越界的问题，改善了checkbox全选时参数与显示存在的问题**
   3. **点击保存，由于参数遗漏，后台拆分参数时会数据越界报错无法保存，正在修改中**
4. **存在疑惑：为什么参数不一个个传非要拼接**
5. **收获体会：页面基本修改完成**
6. **导师回复：**

#### 周四（8月18日）

1. **学习目的：事件定义后台拆分参数修改**
2. **学习方式：请教导师**
3. **学习内容：**

**问题：由于保存时得到的参数格式可能不完整，导致数组越界等报错，无法正常保存数据，**

* 1. **优化代码，改变遍历方式，整理参数处理流程，统一参数命名，跳过‘空参数’等，使条理更清晰易于维护**

1. **存在疑惑：有些元素没有被保存TODO?**
2. **收获体会：注释是好习惯，便于理解维护**
3. **导师回复：**

#### 周五（8月19日）

1. **学习目的：反馈与修改样式**
2. **学习方式：请教导师**
3. **学习内容：**
   1. **反馈与整理事件定义预览样式，删除元素时初始化化数据设置，整理导出静态html样式demo，**
   2. **点击更新按钮时，页面跳转修改，尝试初始化页面，由于添加时部分参数并没有添加到数据库，所以只能用旧数据测试，格式不统一，仍需要调整如何获取拆分参数并初始化页面**
4. **存在问题：数据参数格式拆分与保存时格式并不完全一样**
5. **收获体会：导出静态页面时，样式不包含图片（/images）**
6. **导师回复：**

### 第五周（8.22-8.26）

#### 周一（8月22日）

1. **学习目的：测试版本更新部署，整理新样式静态页面**
2. **学习方式：请教导师、请教同事**
3. **学习内容：**
   1. **导出war包，提交到服务器，更新修改配置并启动，使用到shell脚本，一般常用命令：**
      1. **ls [选项] 文件或目录  
         常用[选项]如下：  
         -a:显示所有的文件，包括以“.”开头的隐含文件。  
         -l:长格式输出  
         -m:宽行输出  
         -F:以各种符号表示不同的文件类型  
         --color:彩色输出  
         -R:递归输出**
      2. **rm [选项] 文件…  
         功能：删除文件或目录   
         - f 强制删除  
         - r 将参数中列出的全部目录和子目录均递归地删除**  
         **- i 进行交互式删除**
      3. **mkdir [选项] dir-name  
         功能：创建一个目录  
         - m 对新建目录设置存取权限  
         - p 可以是一个路径名称。此时若路径中的某些目录尚不存在， 加上此选项后， 系统将自动建立好那些尚不存在的目录，即一次可以建立多个目录。  
         #mkdir –m a=rwx test  
         #mkdir –p test3/test4(test3和test4均为新目录）**
      4. **rmdir [选项] dir-name  
         功能：删除空目录。  
         - p 递归删除目录dirname，当子目录删除后其父目录为空时，也一同被删除。**
      5. **tar [-c][-r][-t][-x][-v][-z][f 文件名] 文件和目录名  
         功能：可以归档多个文件和目录到一个.tar文件下，还可以从一个归档文件中抽取一个文件和目录。  
         -c:创建归档文件  
         -r:增加文件到归档文件中  
         -t:查看归档文件中的文件  
         -x：解开归档文件  
         -v:显示冗长信息  
         -z：进行压缩和解压**
   2. **导出静态html，浏览器导出是时不能导出生成在iframe下的内容，如要导出可以F12，然后选择复制内容，导出的样式脚本可以统一在一个文件夹，直接覆盖，然后各个htm文件修改导入的路劲即可，然后重命名旧文件夹才能删除**
   3. **更新事件定义拆分参数初始化，可以在获取到参数后判断是否非空，然后对html元素操作，因为加载执行的顺序，注意要在js里嵌入java而不是在Java里嵌入js**
4. **存在问题：shell脚本很强大，但是不常用就难记住**
5. **收获体会：开发工具操作习惯，复习shell脚本**
6. **导师回复：**

#### 周二（8月23日）

1. **学习目的：Storm入门demo**
2. **学习方式：请教导师、同事讨论、网上教程**
3. **学习内容：**
   1. **导入项目，1、添加遗漏的jar包，2、移除重复jar包，然后main并不能顺利运行出结果**
   2. **学习Storm教程，了解Storm在项目中的功能作用，重新搭建**

**maven环境下的demo,storm依赖如下：**

**<dependency>**

**<groupId>org.apache.storm</groupId>**

**<artifactId>storm-core</artifactId>**

**<version>1.0.0</version>**

**</dependency>**

**1.创建一个Spout作为数据源**

**Spout作为数据源，它实现了IRichSpout接口，功能是读取一个文本文件并把它的每一行内容发送给bolt。**

**2.创建bolt来处理Spout发射出的数据**

**Spout已经成功读取文件并把每一行作为一个tuple(在Storm数据以tuple的形式传递)发射过来，Bolt中最重要的是execute方法，每当一个tuple传过来时它便会被调用。**

**3.在main函数中创建一个Topology**

**在这里我们要创建一个Topology和一个LocalCluster对象，还有一个Config对象做一些配置。**

**主要就是这三个步骤，Storm可以本地运行，用作开发和测试**

* 1. **初步了解esp宕机恢复流程**
  2. **整理Excel数据**

**1、十进制转十六进制：DEC2HEX（number,place）,place参数可选**

**2、字符拼接：CONCATENATE(V1,V2,V3...)**

1. **存在问题：Storm demo运行没输出结果**
2. **收获体会：学习流处理Storm**
3. **导师回复：**

#### 周三（8月24日）

1. **学习目的：Storm测试demo**
2. **学习方式：请教导师、同事讨论、网上教程**
3. **学习内容：**

**由于原项目运行storm时没有输出，所以重建以测试**

* 1. **新建maven工程**

**1、导入storm依赖包**

**2、创建Spout**

**public** **class** WordReader **implements** IRichSpout{

**private** **static** **final** **long** ***serialVersionUID*** = -6773456320028125762L;

**private** SpoutOutputCollector collector;

**private** FileReader fileReader;

**private** **boolean** completed = **false**;

**public** **void** nextTuple() {

String str;

BufferedReader reader = **new** BufferedReader(fileReader);

**try** {

**while** ((str = reader.readLine()) != **null**) {

**this**.collector.emit(**new** Values(str), str);

}

} **catch** (Exception e) {

**throw** **new** RuntimeException("Error reading tuple", e);

} **finally** {

completed = **true**;

}

}

**public** **void** open(Map conf, TopologyContext context, SpoutOutputCollector collector) {

**try** {

**this**.fileReader = **new** FileReader(conf.get("wordsFile").toString());

} **catch** (FileNotFoundException e) {

**throw** **new** RuntimeException("Error reading file ["

+ conf.get("wordFile") + "]");

}

**this**.collector = collector;

}

**public** **void** declareOutputFields(OutputFieldsDeclarer declarer) {

declarer.declare(**new** Fields("line"));

}}

**说明：open()会根据conf配置读取文件数据源，nextTuple()会不停的向Bolt发送信息，Bolt每处理完就会再次触发该方法**

* 1. **创建Bolt**

**public** **class** WordNormalizer **implements** IRichBolt{

**private** **static** **final** **long** ***serialVersionUID*** = 4721613608468891421L;

**private** OutputCollector collector;

**public** **void** execute(Tuple input) {

String sentence = input.getString(0);

String[] words = sentence.split(" ");

**for** (String word : words) {

word = word.trim();

**if** (!word.isEmpty()) {

word = word.toLowerCase();

List a = **new** ArrayList();

a.add(input);

collector.emit(a, **new** Values(word));

}

}

collector.ack(input);

}

**public** **void** prepare(Map arg0, TopologyContext arg1, OutputCollector collector) {

**this**.collector = collector;

}

**public** **void** declareOutputFields(OutputFieldsDeclarer declarer) {

declarer.declare(**new** Fields("word"));

}}

**说明：每收到一个tuple数据，execute()方法就会对数据解析处理，然后交给另一个Bolt(如果有)**

* 1. **可以创建无限个Bolt,同理②，同样实现execute()，执行不同的业务逻辑，数据可以一个个Bolt传递下去**
  2. **创建Topology，main方法提供程序入口**

**public** **class** WordCountTopologyMain {

**public** **static** **void** main(String[] args) **throws** InterruptedException {

//定义一个Topology

TopologyBuilder builder = **new** TopologyBuilder();

builder.setSpout("word-reader",**new** WordReader());

builder.setBolt("word-normalizer", **new** WordNormalizer(),1)

.shuffleGrouping("word-reader");

builder.setBolt("word-counter", **new** WordCounter(),3)

.fieldsGrouping("word-normalizer", **new** Fields("word"));

Config conf = **new** Config();

conf.put("wordsFile", "d:/text.txt");

conf.setDebug(**false**);

conf.put(Config.***TOPOLOGY\_MAX\_SPOUT\_PENDING***, 1);

//创建一个本地模式cluster

LocalCluster cluster = **new** LocalCluster();

cluster.submitTopology("Getting-Started-Toplogie", conf,

builder.createTopology());

Thread.*sleep*(6000);

cluster.shutdown();

} }

**说明：TopologyBuilder将Spout和Bolt连接成为一个topo，不带参数的main在本机模式执行，带参数的需要搭建集群环境**

1. **存在问题：从Kafka中读数据**
2. **收获体会：Storm demo**
3. **导师回复：**

#### 周四（8月25日）

1. **学习目的：Strom从Kafka读数据**
2. **学习方式：请教导师、同事讨论、上网教程**
3. **学习内容：**
   1. **编写Storm：**

**1、Spout关键代码：**

**private** ConsumerConfig createConsumerConfig() {

Properties props = **new** Properties();

props.put("zookeeper.connect", "132.121.86.57:2181,132.121.86.103:2181,132.121.86.104:2181");

props.put("group.id", "sub\_flowmsg");

props.put("zookeeper.session.timeout.ms", "6000");

props.put("rebalance.max.retries", "5");

props.put("rebalance.backoff.ms", "2000");

props.put("zookeeper.sync.time.ms", "2000");

props.put("auto.commit.interval.ms", "5000");

**return** **new** ConsumerConfig(props);

}

**public** **void** open(Map conf, TopologyContext context, SpoutOutputCollector collector) {

**this**.collector = collector;

**this**.consumer = kafka.consumer.Consumer.*createJavaConsumerConnector*(createConsumerConfig());}

**初始化zookeeper连接**

**public** **void** activate() {

String consumeTopic = "Tflowmsg\_0";

Map<String, Integer> topicCountMap = **new** HashMap<String, Integer>();

topicCountMap.put(consumeTopic, **new** Integer(1));

Map<String, List<KafkaStream<**byte**[], **byte**[]>>> consumerMap = consumer.createMessageStreams(topicCountMap);

KafkaStream<**byte**[], **byte**[]> kafkaStream = consumerMap.get(consumeTopic).get(0);

ConsumerIterator<**byte**[], **byte**[]> it = kafkaStream.iterator();

**while** (it.hasNext()) {

String msg = **new** String(it.next().message());

collector.emit(**new** Values(msg));

}}

**读取Kafka数据并发射到Bolt**

**2、Bolt同上一节；**

**3、Topology 程序入口main：**

TopologyBuilder builder = **new** TopologyBuilder();

builder.setSpout("spout", **new** KSpout(),1);

builder.setBolt("bolt", **new** KBolt(),2).shuffleGrouping("spout");

builder.setBolt("boltA", **new** KBoltA()).fieldsGrouping("bolt", **new** Fields("av"));

builder.setBolt("boltB", **new** KBoltB()).fieldsGrouping("bolt", **new** Fields("bv"));

Config conf = **new** Config();

conf.setDebug(**false**);

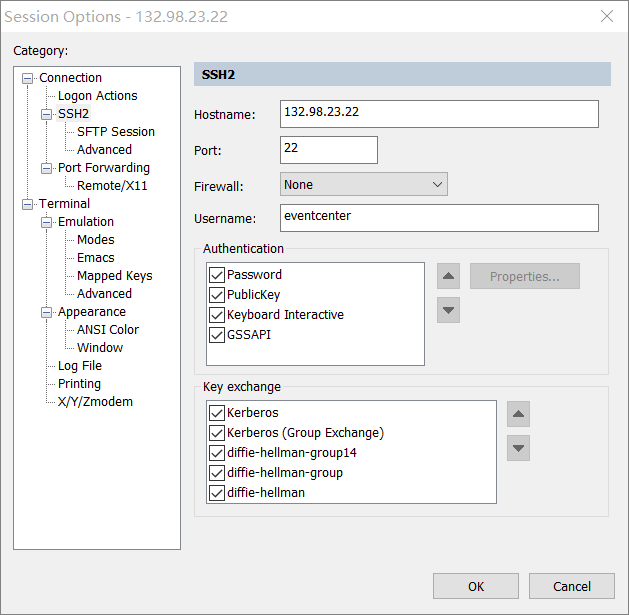
//集群环境运行

conf.setNumWorkers(5);

StormSubmitter.*submitTopology*(args[0], conf, builder.createTopology());

**以上两句是集群服务器与本机模式的代码区别，当main有参数时运行在集群上，注意zoo连接配置、确保Kafka主题对应**

* 1. **连接到Storm服务器，连接配置如图：**



**1、项目package上export出jar文件，上传至22主机上，**

**2、cd /eventapp1/apache-storm-0.9.6/**

**storm jar [jar文件路径] [类包路径] [Topo名称]**

**./：管理员权限运行**

**③宕机恢复操作：**

**导师指导下再依照文档一步步执行操作，**

**要注意脚本命令与主机用户对应（文档有说明）**

**下表是主机部署情况，主要涉及到的主机有**

**ESP--132.121.86.54-eventcenter**

**ESP--132.121.86.56 sapesp**

**ESP--132.121.86.56\_eventcenter**

**ESP--132.121.86.103sapesp //预付费**

**ESP--132.121.86.104sapesp //后付费**

|  |  |
| --- | --- |
| 54 | 信令/流量文件处理 |
| 56 | JAVA大部分进程、ESP数据备份 |
| 57 | ZK、KAFKA、ESP |
| 58 | 短信类事件处理、dpi文件处理、java进程、客户群文件、MYSQL数据、WEB应用 |
| 100 | KAFKA、部署了整个测试环境 |
| 102 | KAFKA、ESP、REDIS |
| 103 | ZK、KAFKA、ESP、REDIS |
| 104 | ZK、KAFKA、ESP、REDIS |
| 132.98.23.22 | Strom |
| 132.98.23.23 | strom |

1. **存在疑惑：Storm Bolt之间的参数传递**
2. **收获体会：成功从Kafka读到数据并部署在服务器运行**
3. **导师回复：**

#### 周五（8月26日）

1. **学习目的：学习Strom常见模式、名词概念，熟悉宕机恢复**
2. **学习方式：网上教程、同事讨论、请教导师**
3. **学习内容：**
   1. **Storm常见模式：**[**http://www.cnblogs.com/panfeng412/tag/Storm**](http://www.cnblogs.com/panfeng412/tag/Storm)

**下载example，导入到项目，对照拓扑模式查看源码学习**

* 1. **熟悉宕机恢复操作，理解各个主机的处理功能，整理操作笔记**
  2. **修改整理重新报工、试用期总结、新员工培养计划表**

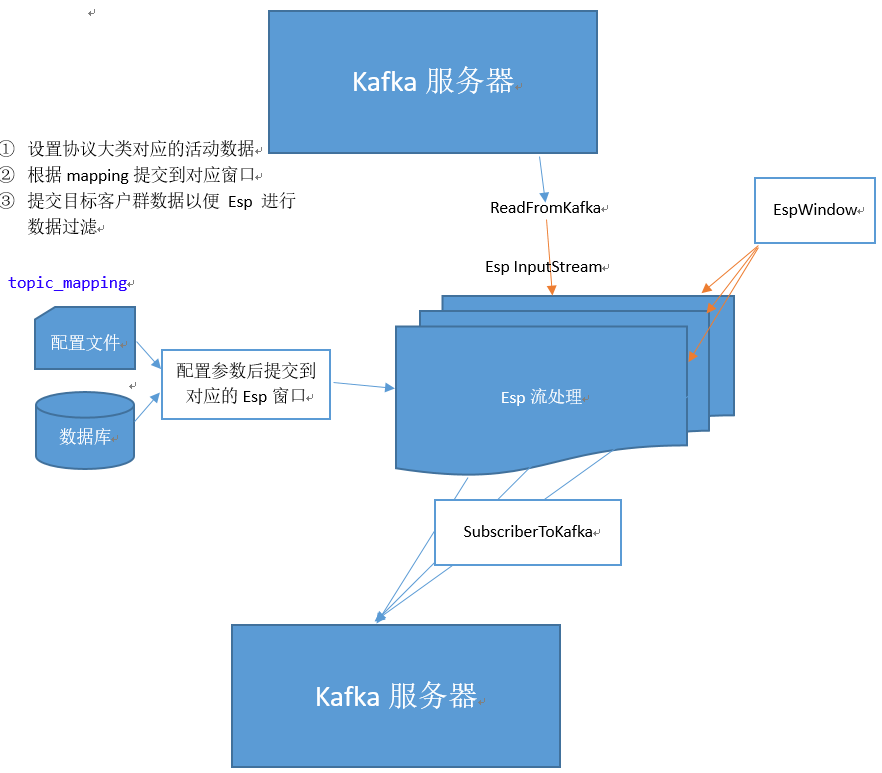
1. **存在问题：模式还未理解透彻，实现细节有待整理记录**
2. **收获体会：熟悉Strom、熟悉宕机维护**
3. **导师回复：**

### 第六周（8.29-9.2）

#### 周一（8月29日）

1. **学习目的：熟悉Esp流程**
2. **学习方式：参加培训、请教导师、同事讨论**
3. **学习内容：**
   1. **培训内容：Esp流程功能讲解，**

**个人理解：**



Esp处理流程：

\*.ccl文件：类似Sql，

1、Input Window相当于输入的Table，参数从RelativeRowWriter 设置对应windowName传入；

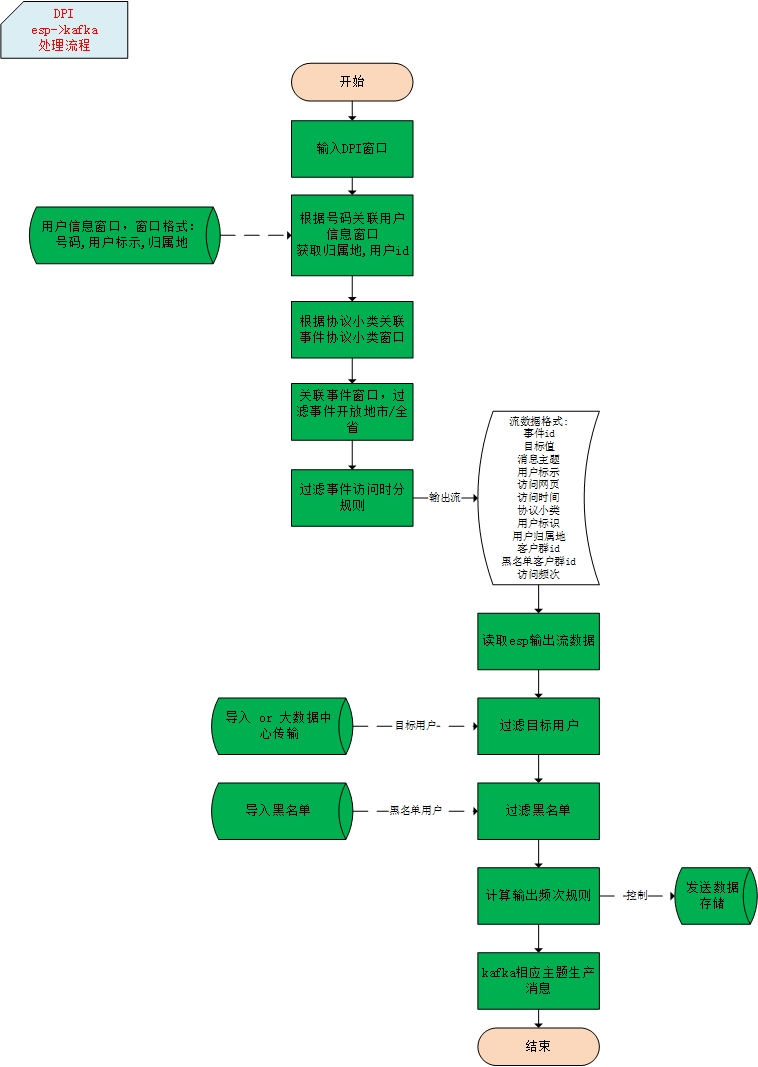
2、Output Stream作为输出Result，

Subscriber subscriber = project.createSubscriber();

subscriber.subscribeStream(*windowsName*); 获取对应的Result

再经由Producer 把信息流交给Kafka

**正解流程图：**

****

1. **存在问题：由于购买的Esp，文档说明不够详细理解起来稍微复杂**
2. **收获体会：熟悉Dpi:Kafka🡪Esp🡪Kafka**
3. **导师回复：**

#### 周二（8月30日）

1. **学习目的：Storm并行度**
2. **学习方式：网上教程**
3. **学习内容：**

**Storm允许计算水平扩展到多台机器,将计算划分为多个独立的任务在集群上并行执行。**

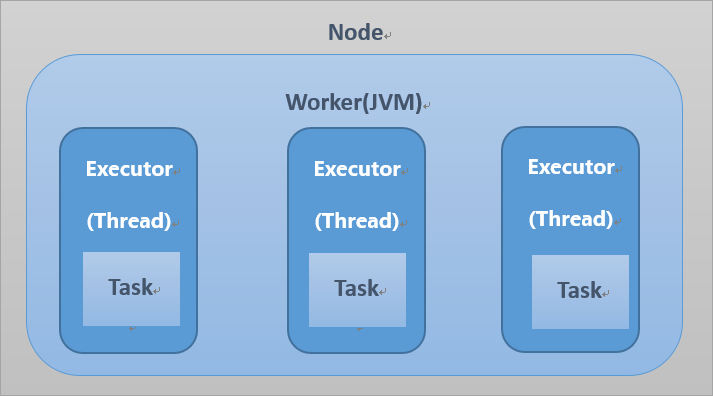
**在storm中,任务只是在集群中运行的一个Spout的bolt实例**

**四个主要组件：**

* + 1. **Nodes(机器节点)：**为Storm集群参与执行拓扑的部分机器。Storm集群包含一个或多个节点来完成工作。
    2. **Workers(JVM)：**在一个节点上运行独立的JVM进程。每个节点配置一个或更多运行的worker。一个拓扑可以请求一个或更多的worker分配给它。
    3. **Executors(线程)：**worker运行在JVM进程一个Java线程。多个任务可以分配给一个Executor。除非显式重写，Storm将分配一个任务给一个Executor。
    4. **Tasks(Spout/Bolt实例)：**任务是Spout和bolt的实例，在executor线程中运行nextTuple()和executre()方法。

**一个Topology在主机上的运行状况如下图：**

所以并行性只有线程级别，每个任务运行在一个JVM的一个单独的线程内，那么如何提高并行性?



**提高Topology并行性：**

* 1. 分配额外的worker

Config conf = **new** Config();

conf.setNumWorkers(2);

* 1. 配置executor数和task数

builder.setSpout(***SENTENCE\_SPOUT\_ID***, spout, 2); //executor为2

**注意：增加woker的数量并不会影响一个拓扑在本地模式下运行。一个拓扑在本地模式下运行总是运行在一个单独的JVM进程,所以只有任务和executro并行设置才会有影响。**

1. **存在问题：Storm还有其他知识点未熟悉，下一步Stream分组**
2. **收获体会：熟悉Strom并行运行原理**
3. **导师回复：**

#### 周三（8月31日）

1. **学习目的：编写测试用例**
2. **学习方式：同事讨论、请教导师**
3. **学习内容：**
   1. **编写测试用例**

**业务规则配置、流数据处理(分位置、流量、dpi)，**

**目标客户群管理**

**编写流程：1、登陆测试项目，截取页面附操作说明，**

**2、远程主机，按步骤输入命令，查看log**

* 1. **项目的Storm改造可能停止，整理了一下代码上传至github，方便后续学习**
  2. **配置Esp监控事件，修改了mk\_event\_info和mk\_eventconcent表格，重启了dpi窗口加载**

1. **存在问题：Esp执行顺序不明白**
2. **收获体会：复习git操作、熟悉Esp命令功能**
3. **导师回复：**

#### 周四（9月1日）

1. **学习目的：Esp流程理解**
2. **学习方式：同事讨论、请教导师**
3. **学习内容：**
   1. **后付费宕机恢复，**
      1. **停止正式拷贝、处理积压**

**Kill:cpflowdata.py**

**Restart:** **ReadKafkaToEsp #重启读数据**

**重启Esp工程**

* + 1. **Esp数据恢复：**

**/eventapp3/useddatawindow/ #每个小时数据**

**/data/proc #程序扫描的正式目录**

**/data/flowdata\_backup/ #备份每个小时**

**/data/tmpflowdata/ #临时目录**

**nohup python cpflowdata.py #正式拷贝**

* + 1. **加载窗户：**

**加载用户套餐数据**

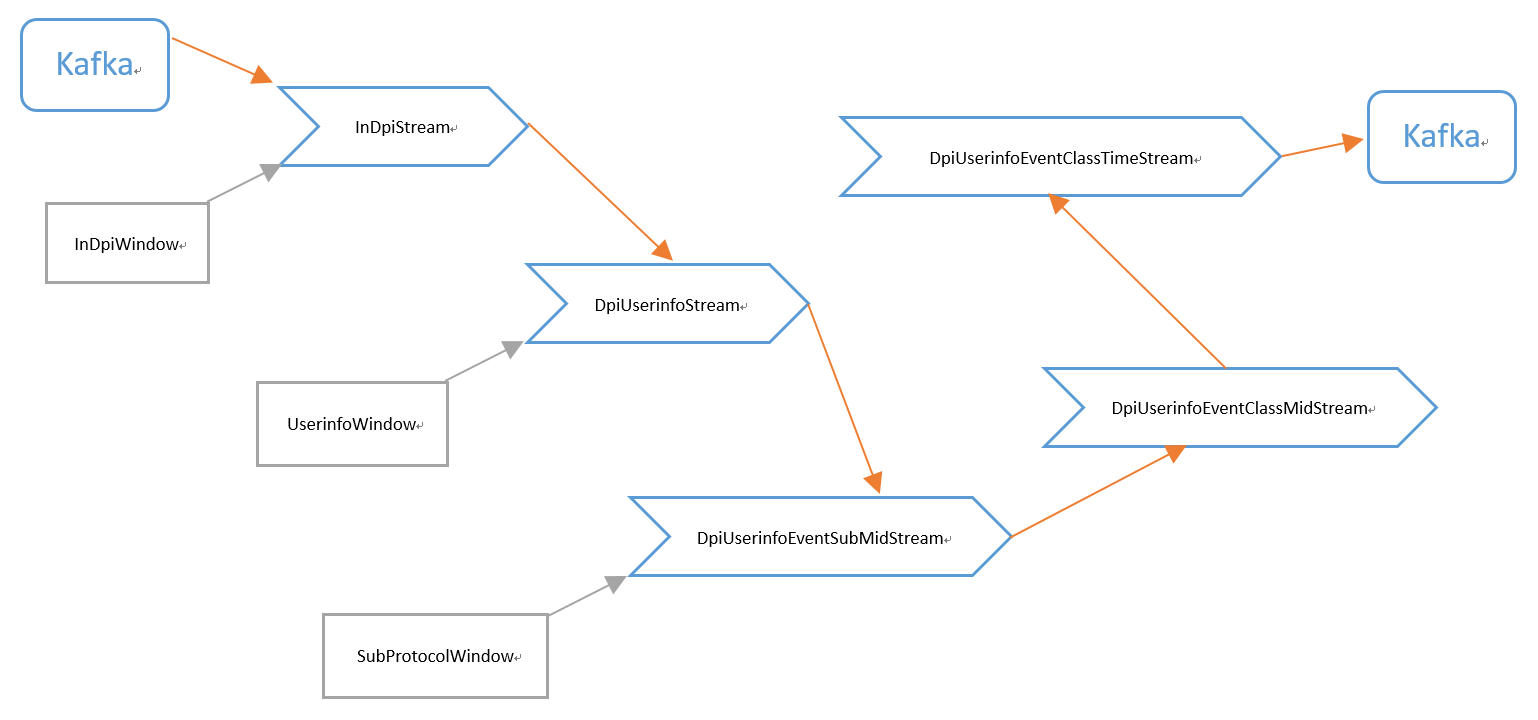
**加载阀值数据**

* + 1. **启动subscriber程序**

**Kill:SubscriberFromEspToKafka|grep PostPaidMonthData**

**Start:** **sh start\_FromEspToKafka.sh PostPaidMonthData**

* 1. **查看文档与代码，理解Esp中Dpi流程**

****

**ESP处理流程说明：**

* + 1. **DPI数据写入InDpiWindow窗口中，该窗口数据保存30分钟，窗口主键为手机号码+访问网站**
    2. **从窗口引出一条本地流InDpiStream, InDpiWindow窗口数据变化 ，InDpiStream就会产生一条数据**
    3. **用InDpiStream与用户信息窗口数据（备份流量累计文件）关联 ，生成一个新的数据流 DpiUserinfoStream**

**类似：AxxStream流与Ax数据关联, 生成一个新的数据流 BxxStream,窗口的作用就是提供配置数据。**

1. **存在问题：Esp各个名词概念**
2. **收获体会：熟悉Esp流程**
3. **导师回复：**

#### 周五（9月2日）

1. **学习目的：学习Esp、导入校园基站**
2. **学习方式：网上教程、同事讨论、请教导师**
3. **学习内容：**
   1. **查看源码，理解后付费Esp流程**
   2. **导入校园基站信息**
      1. **从Excel表格把对应数据转为16进制，**
      2. **将数据复制到文本，Java拼接参数再输出到文本**
      3. **编写Sql导入表格**
   3. **编写Poi读写Excel，为了下次更快的操作，编写Java直接处理Excel，未完待续**
4. **存在问题：暂无**
5. **收获体会：熟悉Poi**
6. **导师回复：**

### 第七周（9.5-9.9）

#### 周一（9月5日）

1. **学习目的：统计信令文件处理延时**
2. **学习方式：同事讨论、请教导师**
3. **学习内容：**
   1. **主机：132.121.86.54 sapesp**

**cd /home/sapesp/sitech/**

**cat sftp.cfg 查看连接信息**

**根据配置新建站点：**



**cd /data/uploads/sdtp/data\_succ\_backup/cs**

**下载数据源，作为统计样本**

* 1. **编写Java统计数据：**

List<Integer> allTime = **new** ArrayList<Integer>();

**this**.fileReader = **new** FileReader(filePath);

BufferedReader reader = **new** BufferedReader(fileReader);

String osTimeStr = filePath.substring(filePath.lastIndexOf("/"));

**int** hour = Integer.*parseInt*(osTimeStr.substring(17, 19));

**int** min = Integer.*parseInt*(osTimeStr.substring(19, 21));

**int** sec = Integer.*parseInt*(osTimeStr.substring(21, 23));

**int** osTime = hour\*60\*60+min\*60 + sec;

String line;

String[] temps;

String temp;

String[] times;

**int** second;

**while** ((line = reader.readLine()) != **null**) {

times = line.split(",");

temp = times[5];

temps = temp.split("-");

second = Integer.*parseInt*(temps[0])\*60\*60+Integer.*parseInt*(temps[1])\*60 + Integer.*parseInt*(temps[2]);

allTime.add(second);

}

**this**.timeAv(allTime, osTime);

**public** **double** timeAv(List<Integer> times,**int** osTime){

**byte** [] buff =**new** **byte**[]{};

StringBuilder sb = **new** StringBuilder();

**double** all = 0;

**double** max = 0;

**int** abs = 0;

//times.clear();

**for**(Integer t:times){

abs = Math.*abs*(t - osTime);

**if**(abs>max) max = abs;

all += abs;

}

sb.append(*inFile*.substring(*inFile*.lastIndexOf("/")+1))

.append("\n").append("Max:"+(**double**)(max)/3600)

.append("\n").append("Av:"+(**double**)all/times.size())

.append("\n").append(times.size());

buff = sb.toString().getBytes();

output =**new** FileOutputStream("d:/avtime.txt");

output.write(buff, 0, buff.length);

**finally**{output.close();fileReader.close();} }

* 1. **整理出Excel文档**

1. **存在问题：暂无**
2. **收获体会：Sftp over SSH**
3. **导师回复：**

#### 周二（9月6日）

1. **学习目的：宕机维护、学习Poi、熟悉Esp优化方案**
2. **学习方式：查看文档、请教导师**
3. **学习内容：**
   1. **预付费宕机，**

**宕机后的数据实际上是脏数据，第四步之前的目的清空它们**

* 1. **大体了解Esp优化方案**
  2. **Poi读取、输出Excel:**

**1、pom.xml:**

<dependency>

<groupId>commons-io</groupId>

<artifactId>commons-io</artifactId>

<version>2.4</version>

</dependency>

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi</artifactId>

<version>3.14</version>

</dependency>

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi-ooxml</artifactId>

<version>3.14</version>

</dependency>

**注意：import** org.apache.poi.xssf.streaming.SXSSFWorkbook;

导入依赖：poi-ooxml

**2、输出Excel:**

**public** **static** **void** main(String[] args) **throws** Throwable {

// keep 100 rows in memory, exceeding rows will be flushed to disk

SXSSFWorkbook wb = **new** SXSSFWorkbook(100);

Sheet sh = wb.createSheet();

**for**(**int** rownum = 0; rownum < 60; rownum++){

Row row = sh.createRow(rownum);

**for**(**int** cellnum = 0; cellnum < 10; cellnum++){

Cell cell = row.createCell(cellnum);

String address = **new** CellReference(cell).formatAsString();

cell.setCellValue(address);

}

}

FileOutputStream out = **new** FileOutputStream("d:test.xlsx");

wb.write(out);

out.close();

// dispose of temporary files backing this workbook on disk

wb.dispose();

wb.close();

}

**3、读取Excel:**

**public** **static** String *File* = "d:/test.xlsx";

**public** **void** read() **throws** IOException {

InputStream stream = **new** FileInputStream(*File*);

Workbook wb = **null**;

String fileType = *File*.substring(*File*.lastIndexOf(".")+1);

**if** (fileType.equals("xls")) {

wb = **new** HSSFWorkbook(stream);

}

**else** **if** (fileType.equals("xlsx")) {

wb = **new** XSSFWorkbook(stream);

}

**else** {

System.***out***.println("File Format Error");

}

Sheet sheet1 = wb.getSheetAt(0);

String ret = "";

**for** (Row row : sheet1) {

**for** (Cell cell : row) {

**switch** (cell.getCellType()) {

**case** Cell.***CELL\_TYPE\_BLANK***:

ret = "";

**break**;

**case** Cell.***CELL\_TYPE\_BOOLEAN***:

ret = String.*valueOf*(cell.getBooleanCellValue());

**break**;

**case** Cell.***CELL\_TYPE\_ERROR***:

ret = **null**;

**break**;

**case** Cell.***CELL\_TYPE\_FORMULA***:

System.***out***.println("CELL\_TYPE\_FORMULA:"+cell);

**break**;

**case** Cell.***CELL\_TYPE\_NUMERIC***:

**if** (DateUtil.*isCellDateFormatted*(cell)) {

Date theDate = cell.getDateCellValue();

System.***out***.println("CELL\_TYPE\_FORMULA:"+theDate);

} **else** {

ret = NumberToTextConverter.*toText*(cell.getNumericCellValue());

}

**break**;

**case** Cell.***CELL\_TYPE\_STRING***:

ret = cell.getRichStringCellValue().getString();

**break**;

**default**:

ret = **null**;

}

System.***out***.print(ret+"\t");

}

System.***out***.println("\n");

}

}

**public** **static** **void** main(String[] args) **throws** IOException {

**new** ReadExcel().read(); }

**4、根据以上，编写了格式转换**

/\*\*

\* 从Excel读取两列，将之转化为16进制，拼接为长字符输出

\* 拼接后+","接下一个

\* 例：C1E,10F,1D6

\* **@author** Jon

\*

\*/

**代码：git@github.com:Jonzheng/opExcel.git**

1. **存在问题：暂无**
2. **收获体会：妈妈再也不用担心我的格式转换了**
3. **导师回复：**

#### 周三（9月7日）

1. **学习目的：Esp技术培训**
2. **学习方式：参加培训**
3. **学习内容：**
   1. **Esp是Eclipse的插件，CCL支持图形化开发**

**Esp流的出入口均是Adapter**

**LOCAL STREAM在外部无法访问**

**Window都有keep 所以它的作用是保存数据流**

**FLEX支持更复杂的业务逻辑**

1. **存在问题：暂无**
2. **收获体会：熟悉Esp**
3. **导师回复：**

#### 周四（9月8日）

1. **学习目的：尝试实现批量Publish到Esp**
2. **学习方式：请教导师，查看代码**
3. **学习内容：**
   1. **1、复制Esp工程作为测试**

**主机：[sapesp@132.121.86.57]**

**cd /espapp/workspace 拷贝dpievent0到**

**[sapesp@132.98.23.23]/espapp/workspace**

**Cd /espapp/workspace/shell**

**sh compile.sh dpievent0 #编译项目**

**sh start\_project.sh dpievent0 #启动Esp工程**

**2、从Kafka读取数据Publish到Esp InStream**

**[eventtest@132.121.86.100]**

**cd locationevent/bin**

**sh start\_ReadKafkaToEsp.sh ReadFromDpiKafka Tdpimsg\_0**

**注意配置：**

**Cd /eventcenter1/locationevent/config**

**dpi\_topic\_mapping.properties**

**将对应主题修改：**

**\_0=esp://132.98.23.23:19011/mywork/dpievent0**

**3、查询Esp流入窗口**

**[sapesp@132.98.23.23]**

**esp\_query -p 132.98.23.23:19011/mywork/dpievent0 -c sap:sap123 -Q " select \* from InDpiWindow"**

**或者查看[eventtest@132.121.86.100]**

**Cd /eventcenter1/locationevent/log**

**cat ReadFromKafkaToEsp\_Tdpimsg\_0.console**

* 1. **修改ReadFromKafkaToEsp，从原来的单条Publish改成一堆堆地Publish，无法本地测试，仍存在一些问题**

1. **存在问题：需要Esp Publisher等API文档**
2. **收获体会：复制部署Esp工程**
3. **导师回复：**