基于 WIFI 探针的数据分析系统

体系结构文档

2017-6-24

南京大学软件学院
Coding Fairy

目录

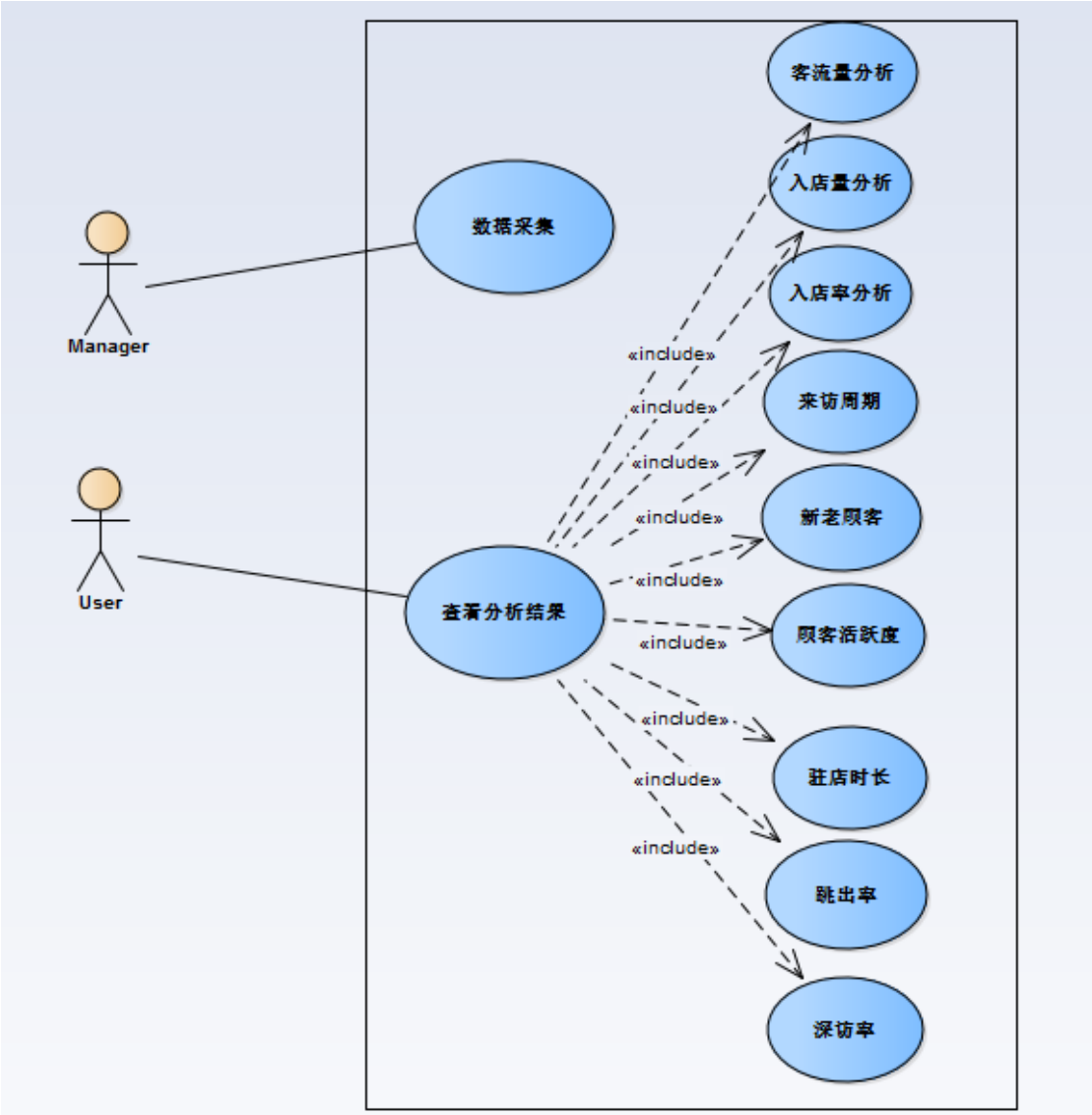
1. 需求

1.1. 需求定义

具体需求描述见需求文档。

需求	需求描述	需求优先级
WIFI 探针	硬件设备	高
数据采集	收集用户数据	高
客流量分析	收集数据后处理	中
入店量分析	收集数据后处理	中
入店率分析	收集数据后处理	中
来访周期分析	收集数据后处理	中
新老顾客分析	收集数据后处理	中
顾客活跃度	收集数据后处理	中
驻店时长分析	收集数据后处理	中
跳出率分析	收集数据后处理	中
深访率分析	收集数据后处理	中
安全性	防 XSS，防注入，防 Ddos	高
可维护性	项目架构与代码风格	中
易用性	前端交互	中
可靠性	扛并发能力，数据吞量	中

1.2. 用例图



2. 架构功能简述

2.1. 主要模块

系统分为四部分：statDisplay 前端展示页面，web 服务器，接收服务器以及存储与计算平台。

2.1.1. statDisplay 前端展示页面

- 调用 web 服务器，展示实时数据与分析结论

2.1.2. Server:web 服务器

- 提供 RESTful 的接口，可获取实时数据与分析结果

2.1.3. Receiver: wifi 探针接收服务器

- 提供多服务器的、可靠的、高并发的、大存储量的 wifi 探针数据接收接口。

2.1.4. mapreduce:计算与存储平台

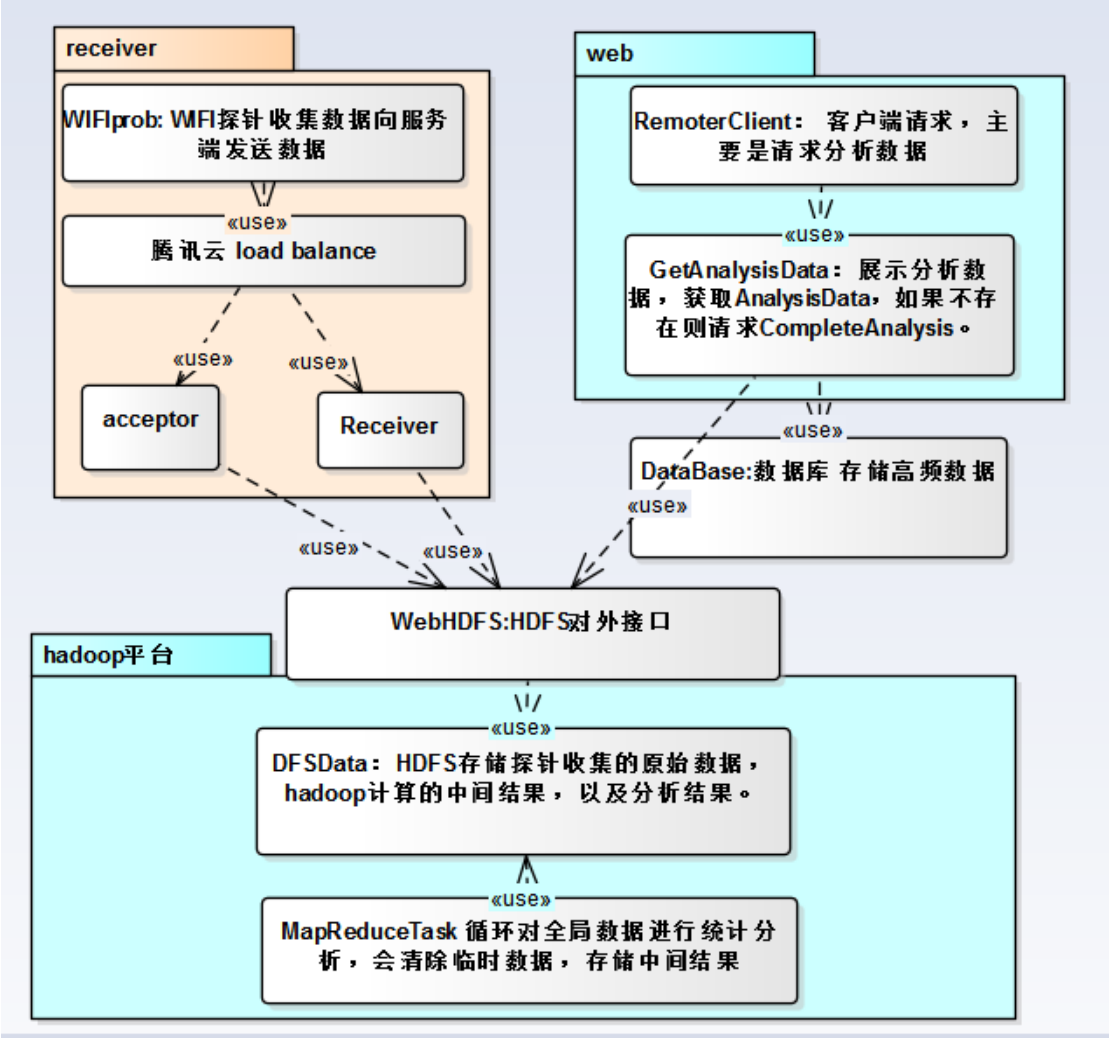
- hadoop 计算集群　对 Receiver 收集的数据进行全局的统计与分析
- hdfs 文件系统　按块存储所有的数据
- 关系型数据库　存储高频度使用的数据，主要是对探针数据的统计结果

2.2. 开发模块目录

子项目目录图如下：

2.3. 模块交互图

下图刻画了各个子项目之间的逻辑关系与交互方式。



2.4. 主要功能执行流程

2.4.1. wifiprob 数据接收流程描述

- 探针发送数据

我们选用的探针是选购的设备，可配置网络，可配置接收服务器以及发送频率等等。探针上传方式是 HTTP post 方式。

- 负载均衡部件选择单个请求的探针接收服务器

在采购腾讯的负载均衡前，我们使用 NGINX 作为单节点转发，转发策略是轮询方式（因为发送数据频率与数据大小在概率上开始均衡的）

目前我们采用腾讯的负载均衡服务，探针设备设置的服务器为负载均衡的浮动 IP，处理会被，根据设置的权重转发至任意一台 receiver 服务器。

具体的压力测试过程详见*测试文档*

- **被选择的探针接收服务器接收数据**

探针接收服务器会对每条数据进行初步处理。

这里会做一些统计分析，尤其是比较好进行增量分析的、不需要所有数据即可分析的统计。（比如总客户统计是很好进行增量分析的，但是回访量分析却没有很好的算法进行增量分析）

处理后会将数据存入队列。

这个缓冲队列初次采用了 redis 的内置队列实现，但是考虑到队列功能上的单一性，我们后来直接采用了 java 内存实现这个多线程的队列。

队列中数据达到一定量时会异步的写入 HDFS 文件系统（这里是为了尽可能的减少与 HDFS 的交互次数，也是为了较少 HDFS 的小文件数量，这个大小被设置为 127M，如果离上一次提交时间超过 20 分钟，不论缓冲大小，都会提交。这样理论上仍有可能会造成 HDFS 小文件，但是这会在 HADOOP 循环任务中处理。）

2.4.2. 用户请求分析数据流程描述

1. 用户发起请求
2. 界面将请求转发的 web 服务器
3. web 服务器收到请求
4. web 服务器查询本地缓存是否有用户所需数据，有则返回

5. web 服务器查询数据库是否有用户所需数据，有则返回
6. web 服务器读取 HDFS 文件系统中的数据，并根据需要存入数据库与本地缓存中，都到后即返回。这里 HDFS 通过 WEBHDFS 以及 Kerberos 授权的方式提供远程调用。

2.4.3. hadoop 平台 mapreduce 任务

hadoop 集群会循环执行以下任务：

1. 合并 hdfs 中的小文件
2. 执行 mapreduce 任务
3. 清除临时文件
4. 清除过期统计数据

2.5. 模块接口描述

2.5.1. Hadoop(HDFS)提供的接口

1.	接口	readStatistic(Path)
	接口描述	通过 WEBHDFS,根据路径读取 HDFS 文件中的统计数据
	前置条件	Path 对应文件已存在，hadoop 平台已启动
2.	接口	saveData(Data,Path)
	接口描述	通过 WEBHDFS 将数据存储至 HDFS 平台
	前置条件	Hadoop 平台已启动，path 指定的父目录存在

3. 详细设计

3.1. MapReduce

3.1.1. MapReduce 算法描述

系统使用 map-reduce 算法统计当前为止的客户流量，新老客户，回访周期，入店时长。算法使用两趟 map-reduce 来得出统计结果。

- **运行前提**

Hadoop 平台已经搭建好，HDFS 文件系统可以访问。

- **执行过程**

(a) 运行 map-reduce 程序前会清空 HDFS 临时文件夹。

(b) 第一次 map-reduce，我们会把 HDFS 文件系统的所有探针数据按照用户 MAC 地址分类，并按照时间排序，存入 HDFS 临时文件夹中。

所有的用户行踪均根据探测到的指针来判断。

(c) 第二次 map-reduce，我们会对每个用户(mac)已排序好的探针数据，统计探针数据在时间轴上的出现密度，由此得出用用户在店内的时间段，继而得出根据单个用户数据统计出来的客户流量，新老客户，回访周期，入店时长这几个统计数据。

统计单个用户的数据后，然后需要进行归并。

1.对于客户流量与新老客户，是分时间段归并的。

2.对于回访周期与入店时长，是分时间长短归并的。

从语言形式上与实际处理上来说，这两者的归并方式都差不多。

- **算法运行示例**

(对应下表第二个 map 的结果)对单个用户数据统计结果如下：

用户 A:　9：30 点入店 1 次，作为新客户入店 1 次，作为老客户入店 0 次

11：20 点入店 1 次，作为新客户入店 0 次，作为老客户入店 1 次

11：30 点入店 1 次，作为新客户入店 0 次，作为老客户入店 1 次

用户 B:　9:40 点入店 1 次，作为新客户入店 1 次，作为老客户入店 0 次

11:40 点入店 1 次，作为新客户入店 0 次，作为老客户入店 1 次

那么

(对应第二个 combiner 的结果)对多个用户数据在单台（mapper）机器上统计结果如下：

9-10 点入店 2 次，作为新客户入店 2 次，作为老客户入店 1 次

10-11 点入店 3 次，作为新客户入店 0 次，作为老客户入店 3 次

(对应第二个 reduce)当用户 A,B 的数据被不同的 mapper 读取时，以上 combiner 的工作应当由 reducer 做。

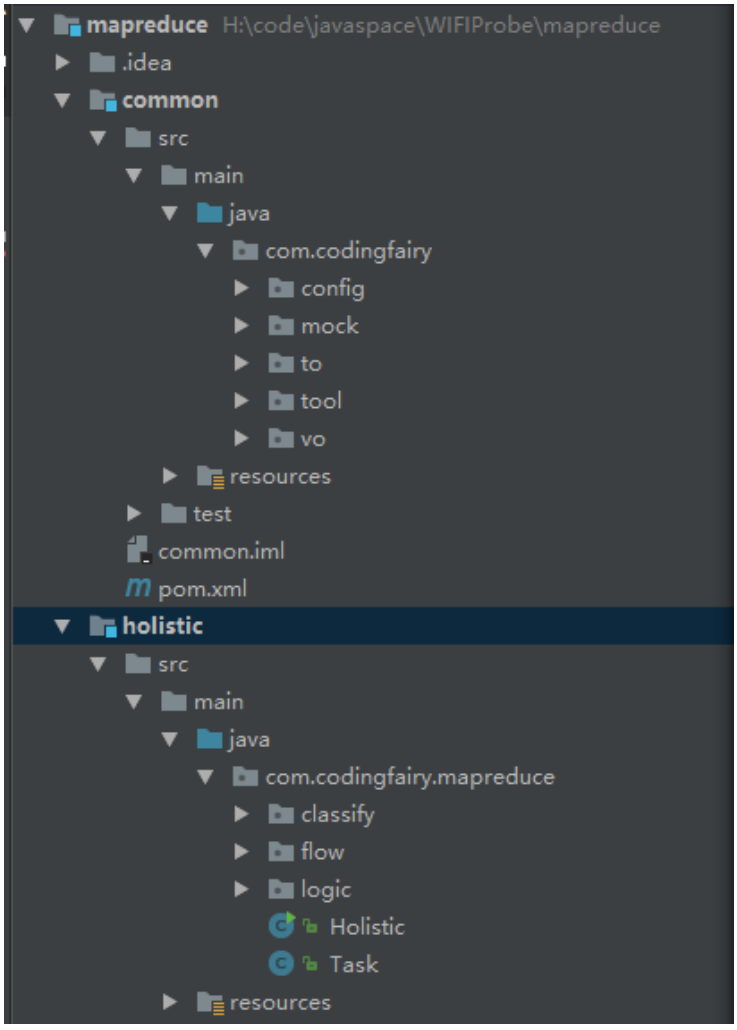
Mapreduce 算法可以如下表所示：

	key	value	备注
第一次读入数据	行号	探针数据包	
第一个 map 输出	用户的 mac 地址	探针数据包	探针数据包中含有 mac 地址,我们把这个 mac 地址拿出来作为 key，数据包作为 value
combiner 输出	用户的 mac 地址	探针数据包	
第二个 reducer 输出	用户的 mac 地址	探针数据包	这里把每个用户的发送的数据包都归一后按照数据包的时间戳排序后存储，存储格式为：*用户 mac 地址【空格】用户所有数据包 json 列表*
第二次读入数据	行号	*用户 mac 地址【空格】*	这里的所有数据包都是时间上排过序的

			用户所有数据包	
第二个 map 输出	客户流量	“Flow”+Hour	(Hour,Hour+1)期间入店数，出店数，深访率，跳出率	都是统计每一个时间区间内的响应统计项的数量。如：*2016.5.30 日 4 到 5 点的入店数为 12，出点数为 45，深访率 0.34，跳出率 0.12，新客户 7，老客户 5。*
	新老客户	“newOld”+Hour	(Hour,Hour+1)期间新客户数，老客户数	
	回访周期	“cycle”+时间区间（t,t+1）	回访周期时间长度居于（t,t+1）区间的客户数	都是统计落在某个区间的的客户数。比如：*回访周期为（6-12 小时）的客户有 122 个，入店时长为（3-5 分钟）的客户有 1241 个*
	入店时长	“inStore”+时间区间（t,t+1）	入店时长时间长度居于（t,t+1）区间的客户数	
combiner 输出	客户流量 新老客户 回访周期 入店时长	同 map 输出	同 map 输出	对来自同一个 map 同一个 KEY 的数据累加,比如在 1:05 跟 1:45 的分别有一人入店，那么输入为{(1,2)=>1},{(1,2)=>1}（表示两个人都是在 1-2 点进入了店铺，但是没有合并），那么 combiner 会合并为{(1,2)=>2}
第二个 reducer 输出	客户流量 新老客户 回访周期 入店时长	同 map 输出	内容同 map 输出 但会转换成 json 格式的 Text	同上，对来自不同的 map 输出同一个 KEY 的数据累加

Map reduce 的统计结果会被存入 HDFS 文件系统中。

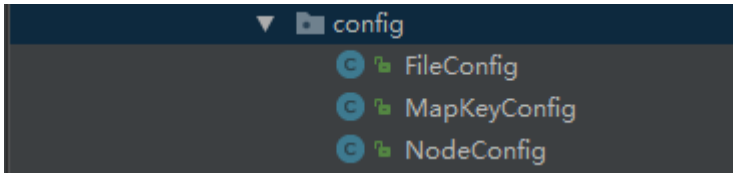
3.1.2. mapreduce 包设计



- **common 公共依赖的代码**

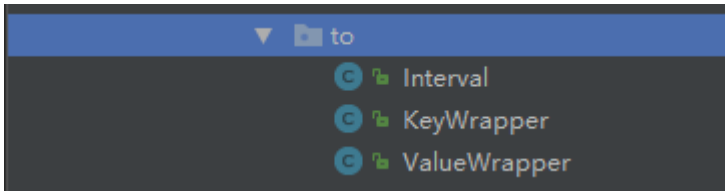
- config 配置类

HDFS 各个文件夹路径配置，字符串常量，服务器节点配置。



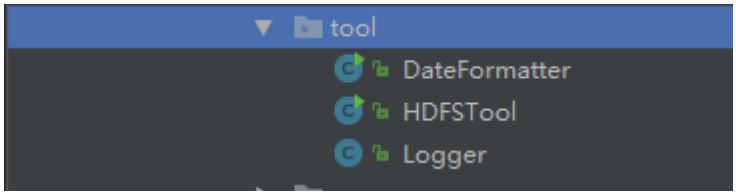
- TO 传输对象封装

to 实现了 Writable 与 cloneable 的 value 包装类，实现了 ComparableWritable 的 key 包装类。方便的自定义包装各种可作为 KEYVALUE 使用的类。



- tool 常用工具类

如文件读写，日志打印，日期格式转换等。

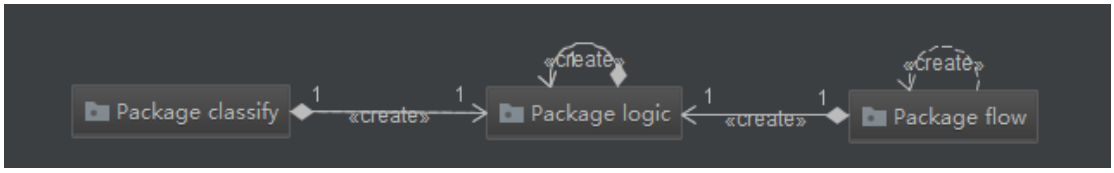


- VO value object

主要是对应 wifi 探针上传的数据格式对象，以及分析结果的数据对象。

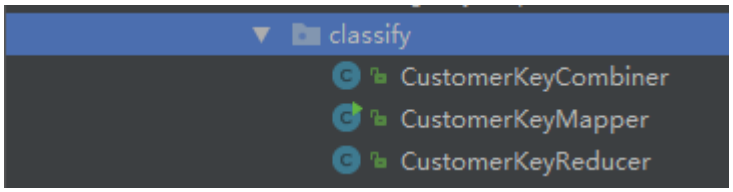
- **整体分析与计算的包**

主要有三个包，依赖关系如下



- classify 安装用户 mac 地址分类数据--第一个 mapreduce

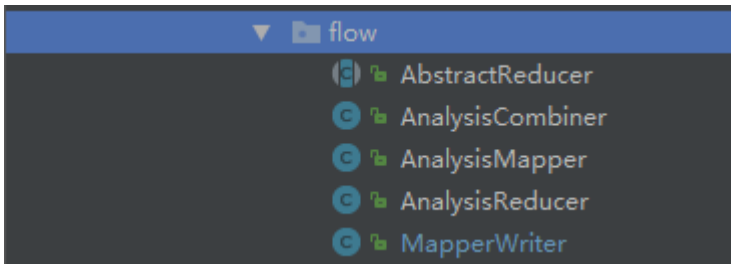
也是第一个 mapreduce 的主要代码 ,用于根据 mac 地址分类的 map-reduce 程序，其中包括了 Mapper、Reducer,Combiner 三个类。



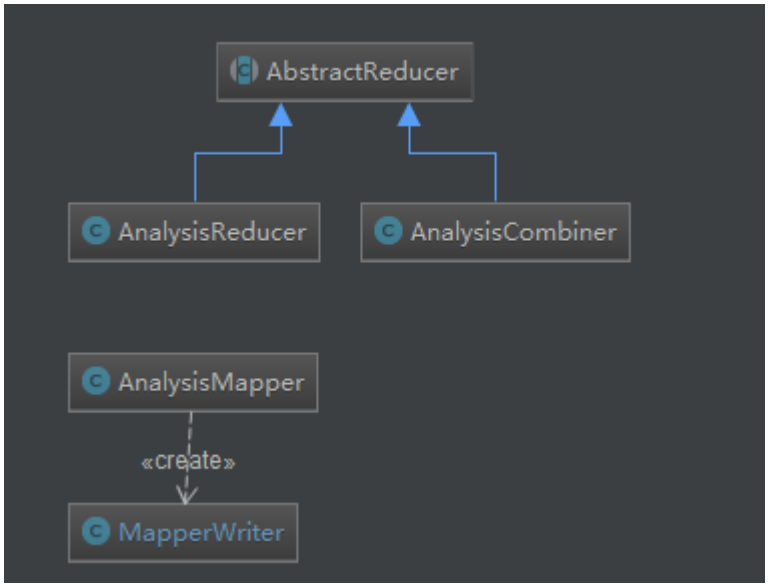
- flow 流式有序的扫描并分析数据--第二个 mapreduce

也就是第二个 mapreduce 的主要代码，用于统计主要分析项。包括了

Mapper、 Reducer,Combiner 三个部分以及辅助类 MapperWriter

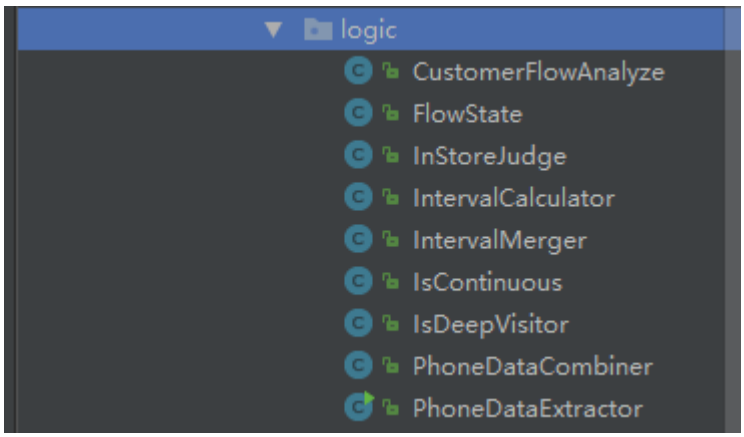


其依赖关系如下：



- logic 复杂逻辑实现

主要是实现一些复杂的逻辑，是 flow 的辅助包

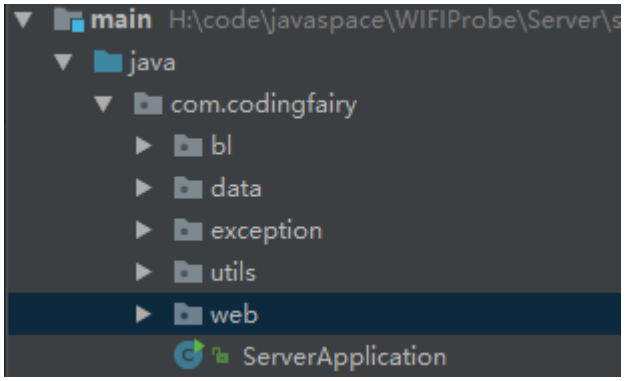


3.2. Web

3.2.1. 处理用户请求

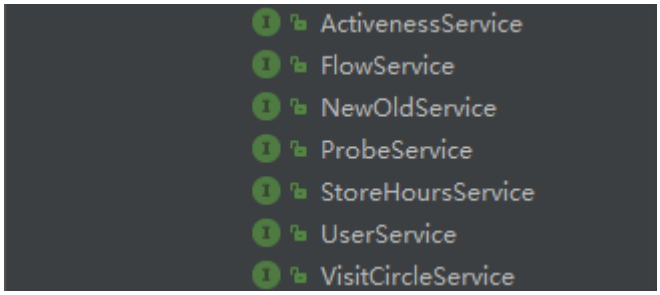
- 用户发起请求
- 界面将请求转发的 web 服务器
- web 服务器收到请求
- web 服务器查询本地缓存是否有用户所需数据，有则返回
- web 服务器查询数据库是否有用户所需数据，有则返回
- web 服务器读取 HDFS 文件系统中的数据，并根据需要存入数据库与本地缓存中，都到后即返回。这里 HDFS 通过 WEBHDFS 以及 Kerberos 授权的方式提供远程调用。

3.2.2. 主要实现分包



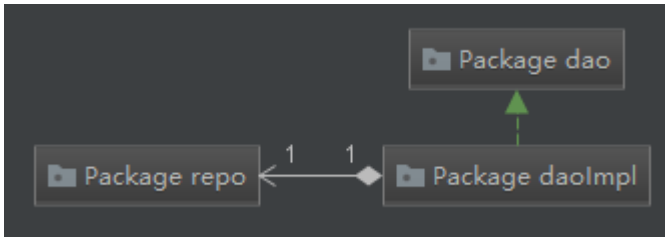
- **Bl：业务逻辑**

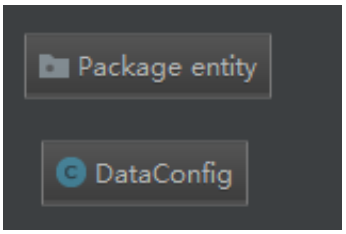
这些业务逻辑基本都对应着前面提到的需求：如活跃度，用户量，新老客户，入店时长，访问周期等等。



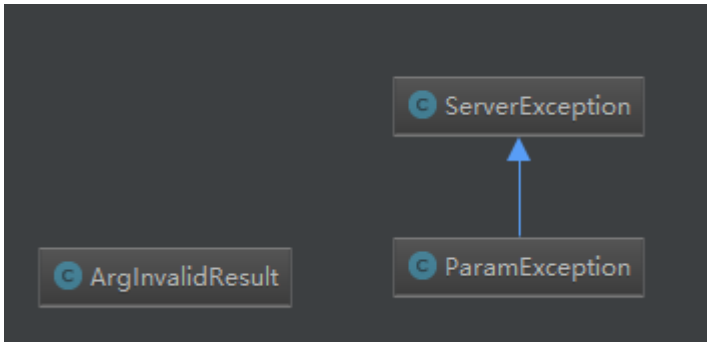
- **data：数据访问层以及持久化对象**

包括数据库映射实体，数据库访问层，数据库配置对象以及协助类。



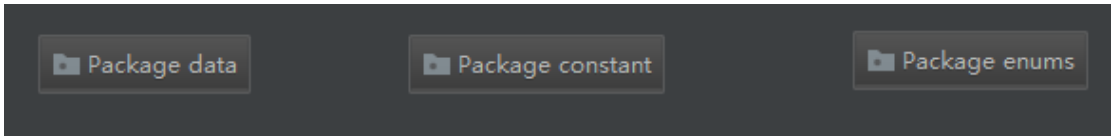


- Exception: 所有的自定义异常封装。



- Util: 工具类封装

  公共工具类，常量，枚举类



- Web: web controller

  主要是路由控制，安全处理等。

3.3. Receiver

探针接收服务器以分布式部署，HTTP post 的方式提交数据。

3.3.1. 负载均衡

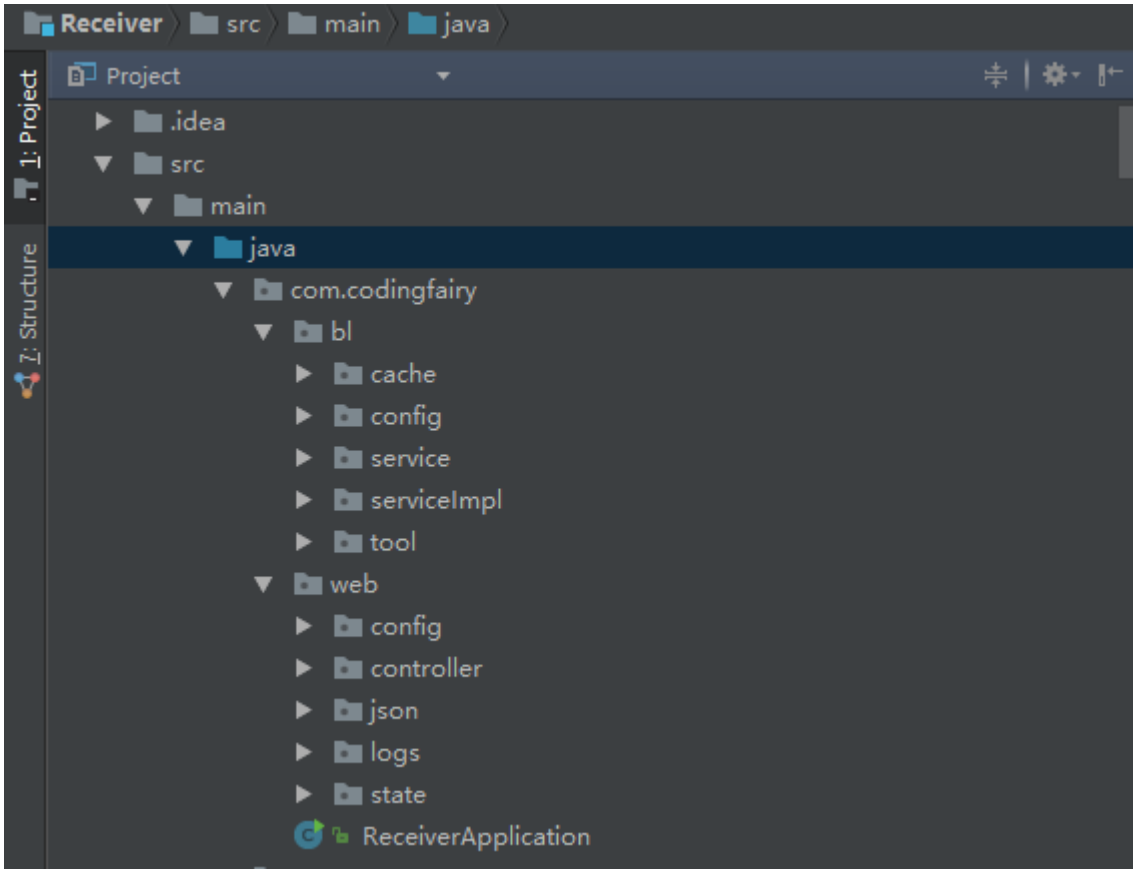
使用腾讯云负载均衡，每个探针向服务器发送的数据都会均匀的负载到各台服务器上。

3.3.2. Session 一致性

搭建分布式 redis 集群存储 session 信息，由于 receiver 服务器目前只接受 wifi 探针发送的数据，所以对 session 的一致性要求比较低。

3.3.3. 数据存储

由于 HDFS 不适合频繁写、小文件存储，所以每个节点都会对本地数据缓存，达到一定规模后才会写入 HDFS 文件。



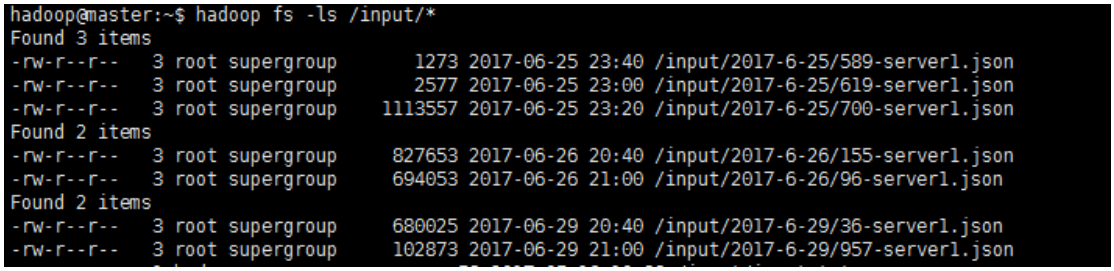
- bl

主要的业务逻辑，包括

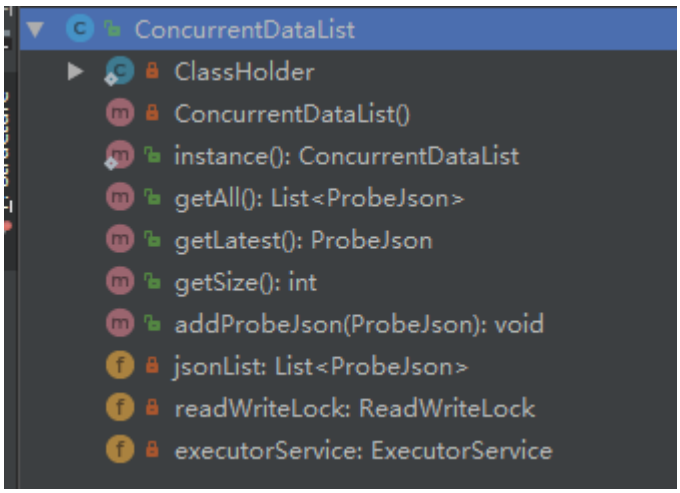
Cache: 对每个 wifi 探针提交到该服务器上的数据都会缓存到一个线程安全的队列中。当这个队列足够大或者离上一次提交的时间超过 20 分钟时，则会被提交

到 Config 包配置的 HDFS 文件系统中。目录与文件格式为 *日期/时间-receiver 服务器名.json*

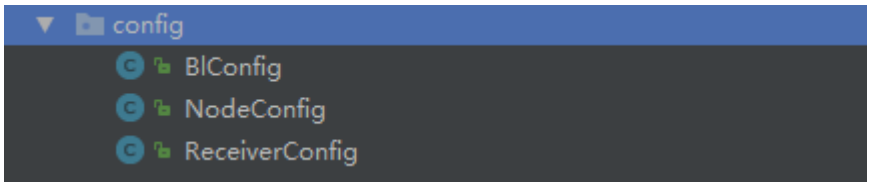
如图：



队列主要接口：

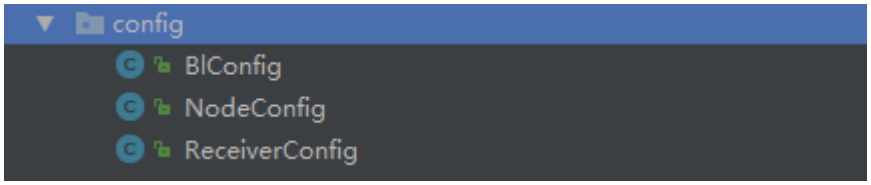


Config：依赖的其他模块配置，主要是 hdfs webHTTP 接口地址。



Service: bl 模块提供的接口，主要是对 wifi 探针提交到该服务器上的数据的具体处理，以及提交本地缓存数据至 HDFS 的接口。

ServiceImpl:service 的具体实现。



- web

http post 数据的接收逻辑：

Config: 字符串常量的配置。

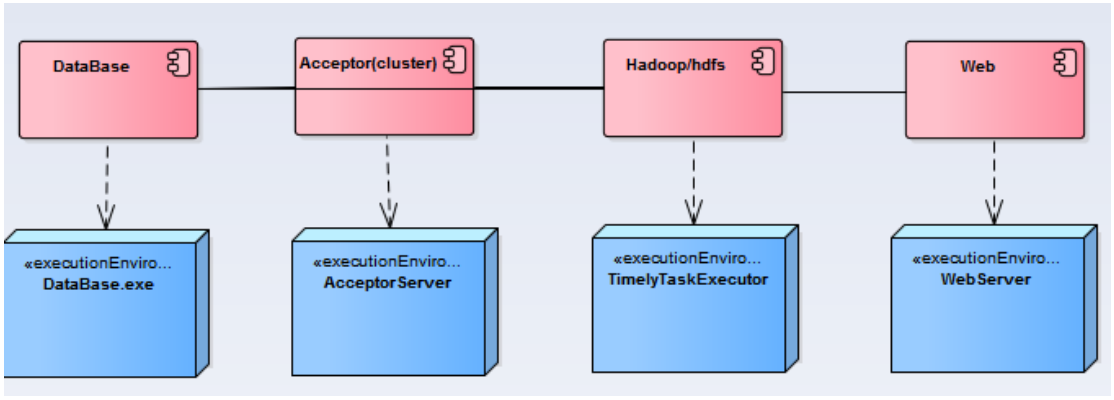
Controller :http 访问路径的配置管理与转发 ,这里主要是 wifi 探针的提交路径，以及将请求转发到业务逻辑中。

Json：java value object。主要是接收 wifi 探针数据字段。

Logs：日志工具类，对业务逻辑的主要方法切入 log

State：服务器的心跳包发送，监控服务器健康

4. 部署视图



4.1. 部署方式

DataBase:mysql 数据库单台

Receiver（Acceptor）:wifi 探针数据接收服务器多台

Hadoop:多节点集群

WebServer：单台

4.2. 节点交互

4.2.1. 用户与 web 服务器

标准的 HTTP 通信协议:对用用户提供查看与监控的接口

4.2.2. wifi 探针与接收服务器

标准的 HTTP 通信协议：对 wifi 探针提供上传数据的接口

4.2.3. Web 服务器与 HDFS

基于 WEB-HDFS 的 HTTP：对 web 服务器提供读文件接口

4.2.4. 接收服务器 Receiver 与 HDFS

基于 WEB-HDFS 的 HTTP：对接收服务器提供读、写文件接口