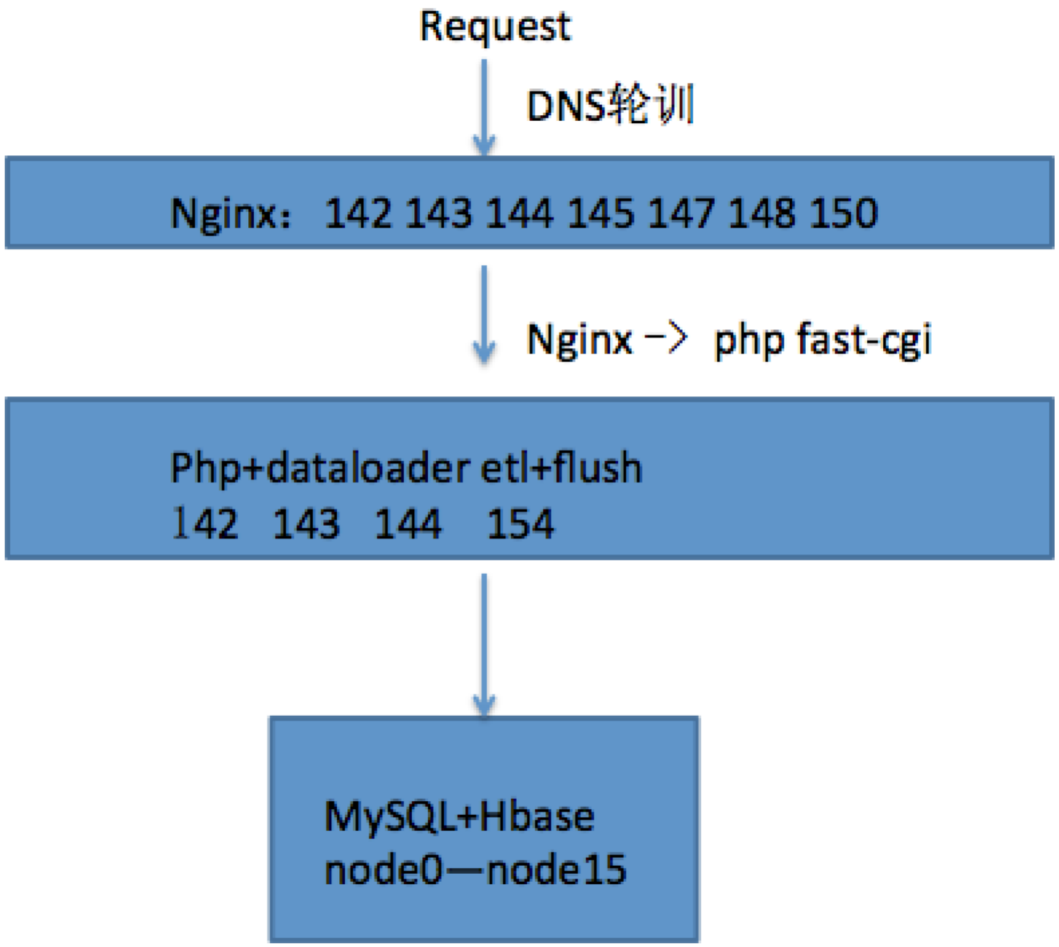
DataLoader 结构



1. 服务器分布

7 台 Nginx： 142 143 144 145 147 148 150
nginx 启动：/etc/init.d/nginx start
nginx 关闭：/etc/init.d/nginx stop
nginx 重启：/etc/init.d/nginx restart
nginx 重载配置文件：/etc/init.d/nginx reload

4 台 PHP ：142 143 144 145
php 启动：/data/php/sbin/php-fpm
php 关闭：cat /data/php/var/run/php-fpm.pid|xargs kill

4 个 dataloader 进程：142 143 144 145
主要分为两部分，etl 和 flush

2. Nginx+PHP
前端 log http 接口，目前支持 v2，v3 和 v4 的 log。
举例（v4）：
http://xa.xingcloud.com/v4/age/uid123?action=visit,0&action0=adcalc.cost,100&update0=language,us&update1=platform,androidmarket
会生成：
site_data 类型的 log
age uid123 user.visit 1379328627192
age uid123 user.update {"geoip":"2013197170"}1379328627192
age uid123 user.update
{"language":"us","platform":"androidmarket"} 1379328627
store_log 类型
{"signedParams":{"appid":"age","uid":"uid123"},"stats":[{"timestamp":1379328627192,"data":["adcalc","cost","","","","",100],"statfunction":"count"}]}

- site_data 和 store_log 区别
site_data 只包含 5 种事件，visit, pay , heartbeat, quit, update。
/data/htdocs/elex_analytics/site_data/
每一条 log 为 6 项，以\t 隔开；
格式为 appid\tuid\tref\tevent\tupdate_json\ttimestamp
其他自定义事件放入 store_log 中。
/data/htdocs/elex_analytics/store_log/
- geoip 的 update 事件
每一个 visit 事件，php 都会生成一个 geoip 的 update 事件。

3. ETL
Github：https://github.com/XingCloud/dataloader_etl；分支为 luav4log
141 的目录为：/home/hadoop/ivytest/dataloader_etl
Etl 的作用在于：
转换原始 log 为内部需要的格式，生成 stream.log 和 user.log。
stream.log 用于离线 offline 的计算和事件导入 hbase，user.log 导入 mysql 的用户属性表。
stream.log 和 user.log 的生成依赖 log4j，配置在 xingcloudlog4j.properties。
转换包括：
- 原始 uid →innerUid
- 事件过滤，验证事件的合法性。
每个项目的合法事件都存放在 144 的 mongo user_info. events_list
- 事件转换，比如 pay.complete 转为 pay.gross 和 pay.fee
- 用户属性的过滤，用户的属性分为 once，cover 和 inc。
once：只更新一次；cover：可覆盖；inc：增加的。
5min 内，一个用户的属性只会被更新一次；
etl 还维护了用户属性的 bitmap：

对于 once 的属性更新过的用户会保存在 bitmap 里，once 的属性不会重复更新；

cover 的属性：last_login_time 的 bitmap 1 个小时重置一次，platformField，versionField，identifierField，languageField4 个小时重置。

Inc 的属性不做缓存。

目录结构：

```
bin
pom.xml
README.md
src
```

bin 下存放部署脚本：

dataloader_deploy_test.sh 部署脚本，git pull、git package、scp 到 141-145 的 xa/runJar 目录。（常用）

dataloader_server_fixstart_test.sh 安全启动，启动时会执行 fillTask，0-mi 的操作，补全当天未完成的任务。（常用）

dataloader_server_print_test.sh 打印出 4 个机器上的 etl 的进程号（常用）

dataloader_server_start_test.sh 启动 etl，这个时刻之前未完成的任务不会补

dataloader_server_stop_test.sh 强制关闭 kill 进程（graceful 的关闭在下一节）

程序入口：

- 提交任务进 redis：

com.xingcloud.dataloader.server.DataLoaderMessageMaker

任务格式有：

message@ip

message：

printTask 打印出当前所有的任务

exit 强制退出

waitAndExit 安全退出，完成当前的任务后退出

nowTask 提交当前任务

ip：具体 ip 或者 all

task,date,index,project,type@ip

提交指定日期，时间段，运行项目，任务类型到 ip 消息队列上

task：

fixTask 强制提交任务，不管 mongo 里的完成情况

fillTask 查询 mongo，完成的任务就忽略

date：

today 或者 20120612

index：每五分钟一个任务，每天的任务为[0，287]，一共 288 个

project：all or 单个项目名

type；任务类型 Normal,EventOnly,UserOnly

Normal = EventOnly+UserOnly

EventOnly 只生成 stream.log，UserOnly 只生成 user.log

ip：all or 192.168.1.142，192.168.1.143...

141 上 crontab：

*/5 * * * *

java -classpath /home/hadoop/xa/runJar/dataloader_etl_testCoin.jar com.xingcloud.dataloader.server.DataLoaderMessageMaker nowTask@all

常用命令：

安全退出：

java -classpath ~/xa/runJar/dataloader_etl_testCoin.jar com.xingcloud.dataloader.server.DataLoaderMessageMaker waitAndExit@all

强制提交

java -classpath ~/xa/runJar/dataloader_etl_testCoin.jar com.xingcloud.dataloader.server.DataLoaderMessageMaker fixTask,today,0-100,all,Normal@all

- etl 进程：com.xingcloud.dataloader.server .DataLoaderETLWatcherCoin

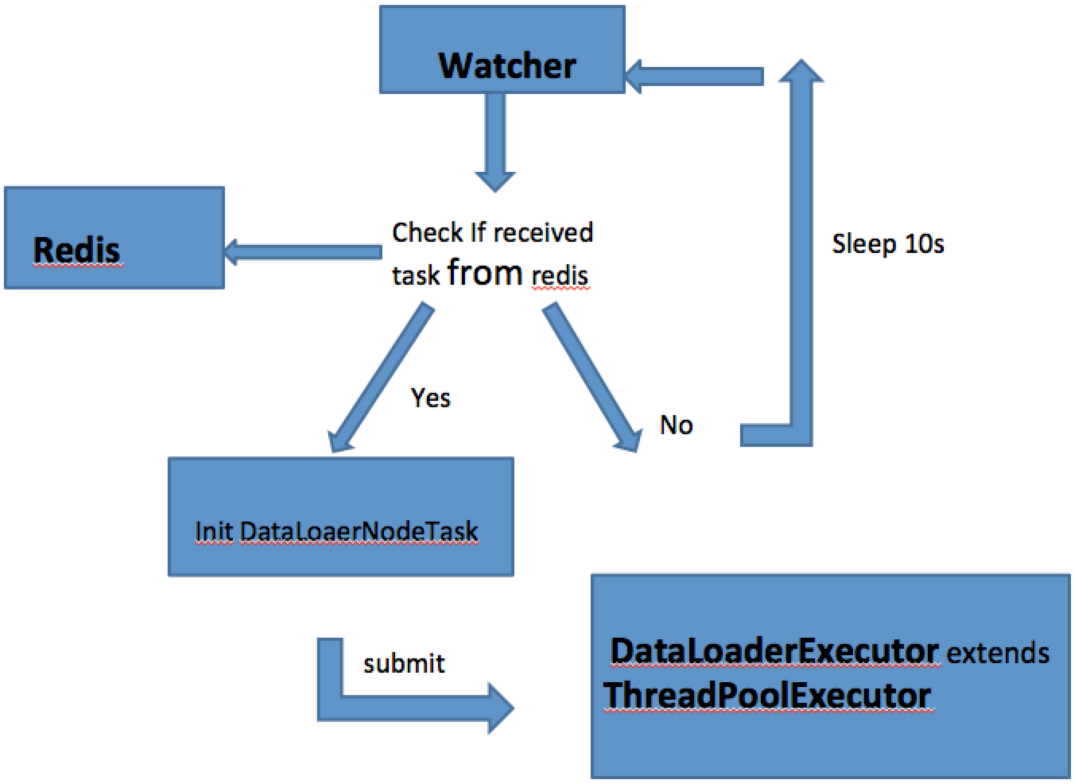
run()函数不停循环从 redis 拿的任务，任务有：

WaitAndExitMessage、ExitMessage、PrintTaskMessage 和 TaskMessage。

以下重点说明 TaskMessage：

当 DataLoaderETLWatcherCoin 从 redis 收到 TaskMessage 的命令后，生成一个 DataLoaderNodeTask。

DataLoaderNodeTask 提交到 DataLoaderExecutor，DataLoaderExecutor 是一个只有一个线程的线程池，处理所有 5min 的任务，并保证重复提交的任务只会被执行一次。



DataLoaderNodeTask 执行过程：

- 记录任务开始状态到 MongoDB；
- 扫描本地 log 目录（包括 v4log），拿到所有 appid，验证 appid 和 projectid 的合法性，分配 appid 和 projectid 的对应关系。
  Map<String, List<String>> projectAppidMatch = assignAppid();
- 针对不同更新方式属性的 bitmap 做一些重置处理：
  比如 12 的倍数次任务清空 bitmap 的 lastlogintime，48 的倍数清空 platformField， versionField，identifierField，languageField；
- 读取所有的日志并存储到中间类 TablePut 中去,并 flush 到本地
  boolean readerFinish = buildProjectTablePut(tablePutPool, projectAppidMatch, v4LogsMaps);
- 清理工作；
- 记录结束状态到 Mongo。

buildProjectTablePut(TablePutPool tablePutPool, Map<String, List<String>> projectAppidMatch, Map<String, List<String>> v4LogsMaps)说明：

对每一个 pid，生成一个 ReaderTask，submit 给 ReaderPool。
ReaderPool 维护了一个线程池（线程大小可配置）。

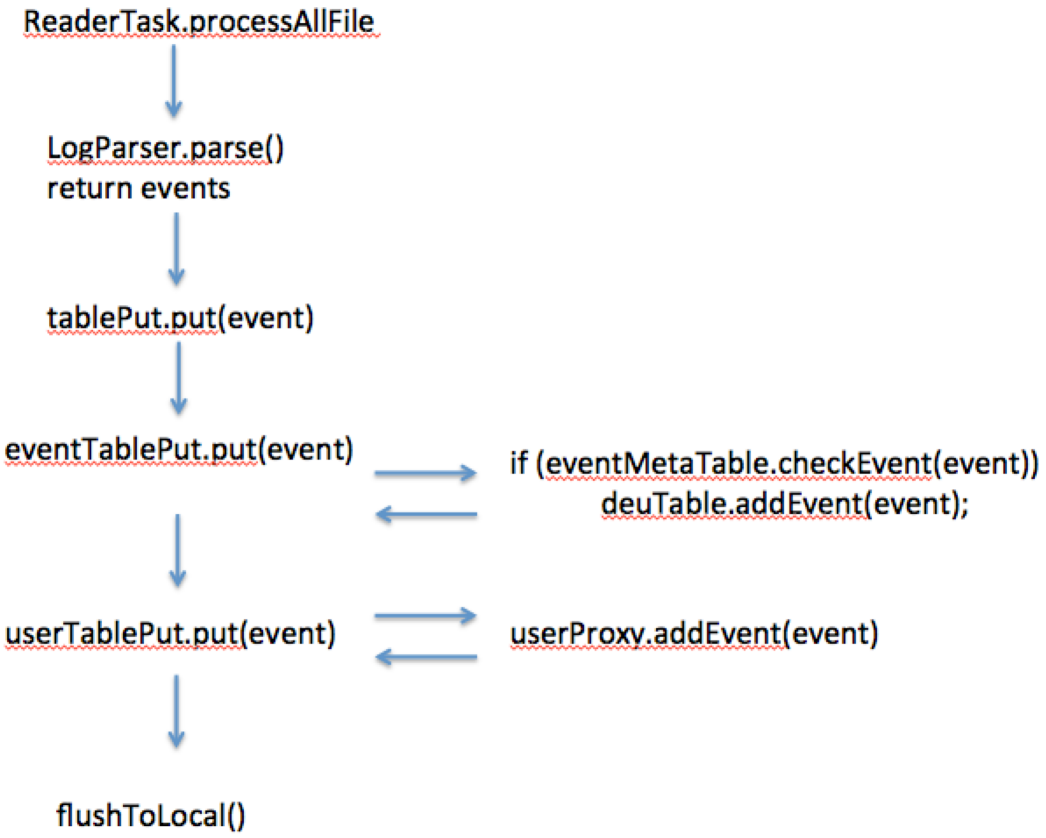
ReaderTask 执行过程：

- RefBitMapRebuild.getInstance().rebuildSixtyDays(project, date, index);
  每天的第 0 个任务，从 mysql dump 出 60 天的活跃用户到本地，重新生成 ref Bitmap。
  如果 ref bitmap 为空，说明 etl 刚刚重启过，就从 60 天的活跃用户恢复出 User.refField, User.registerField, User.geoipField, User.nationField 一

共 4 个 Bitmap。

- SeqUidCacheMap.getInstance().initCache(project);
  如果 rawuid—sequid 缓存不存在，就从本地文件恢复
- for (String appid : appidList) {
  processAllFile(audit_exec, tablePut, LocalPath.SITE_DATA, appid, date, index);
  processAllFile(audit_exec, tablePut, LocalPath.STORE_LOG, appid, date, index);
  }
  轮询所有 appid 分析 sitedata 日志和 store_log 日志
- processV4Log(audit_exec, tablePut, LocalPath.V4_LOG, date, index);
  处理 V4Log，并放入 TablePut
- processCoinTask(audit_exec);
  处理 coin 相关。
- tablePut.flushToLocal();
  flush stream.log 和 user.log
- 任务结束清理工作
  //每 4 个小时 flush cache 到本地一次，但是内存中的缓存不清空
  SeqUidCacheMap.getInstance().flushCacheToLocal(project);
  //每天检查一次该项目的缓存是否需要重置
  SeqUidCacheMap.getInstance().resetPidCache(project);

主要分析过程为：



LogParser：

LogParser 是最主要的解析原始 log 的类，解析原始 log 转化为 event，有三种类型 log 的解析，SITE_DATA, STORE_LOG, V4_LOG。

- SITE_DATA
  List<Event> parseSite()
  对特殊事件，user.visit、pay.complete、user.update、user.heartbeat 和 user.quit 做特殊的事件转换处理。
- STORE_LOG
  按照规则解析自定义事件
- V4log
  如果 nginx 采用 openresty，v4log 都采用这个格式解析

EventTablePut：

EventTablePut 是只更新 hbase event 的 tableput。

当接收到新的 Event 时，先 check event 的合法性，如果合法就存入 DeuTable。

EventMetaTable：

控制项目的事件列表，并提供查询接口判断输入的事件名符不符合要求。

DeuTable：

Hbase event 缓存类。

UserTablePut：

只更新 mysql 的 tableput。

4. Flush

Github 地址：git@github.com:XingCloud/dataloader_flush.git：分支为 for_16_nodes

141 的目录为：/home/hadoop/ivytest/dataloader_flush

Flush 维护了 2 个主要的线程，eventThread 和 userBulkLoadThread，两个线程完全独立，各自读取本地配置文件

eventThread 配置文件路径：/data/log/event16config

userBulkLoadThread：/data/log/user16config

配置文件夹里面包含两种类型的配置文件：

config.properties：log 的路径，名称和当前应该读哪个日期的 log

```
datadir=/data/log
day=20130926
datafile=user.log
```

sendlog.process:

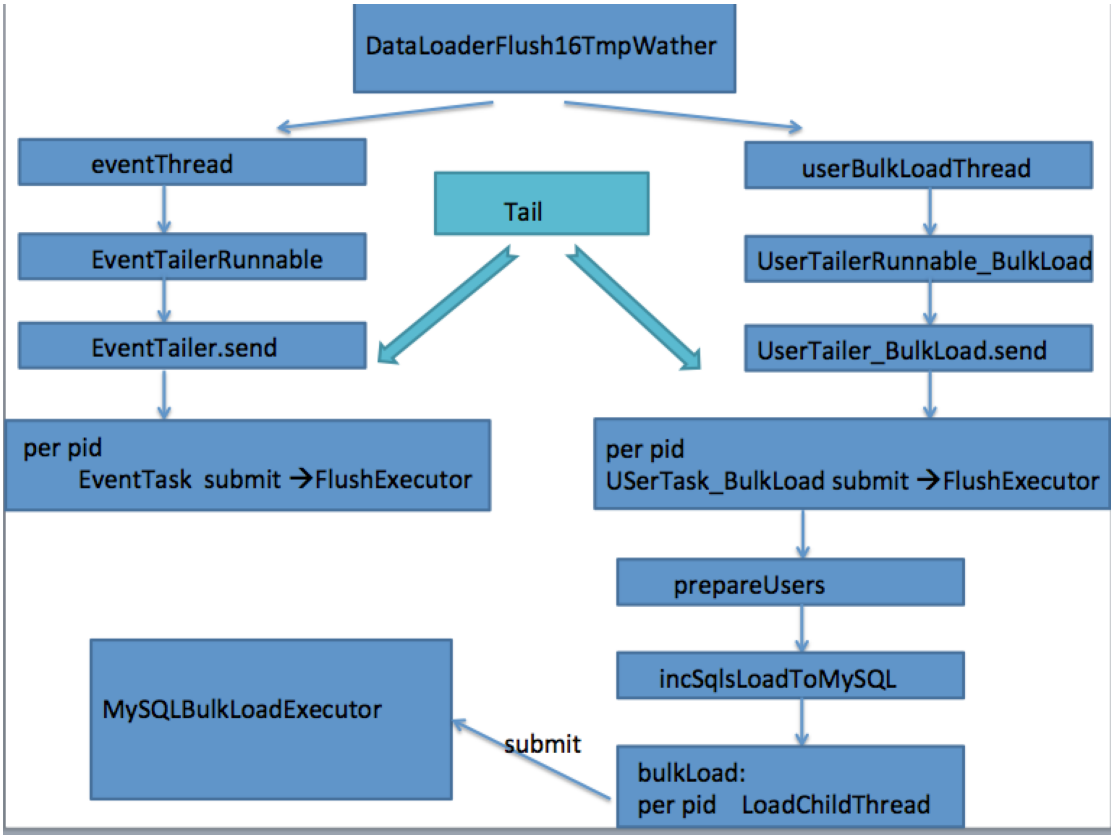
记录当前 log 的读取位置，sendlog.process 如果不存在，就从第 0 行开始读。

关键类：

- Tail：
  实现 linux 下的 tail 功能类，需要子类重写的是
  public abstract void send(List<String> logs, long day)　logs 为从 log 文件中读取的 logs，day 为 log 当前天。
- USerTask_BulkLoad(String project, List<User_BulkLoad> users)：
  bulkload 流程为 prepareUsers → incSqlsLoadToMySQL→bulkLoad

prepareUsers 生成 bulkload 需要的文本文件；
incSqlsLoadToMySQL 更新属性为 inc 的属性无法通过 bulkload 的方式更新，只能 sql；
bulkload 提交对应每张表的 mysql bulkload 到线程池。

- EventTask
  Flush event 到 hbase



5. 142-145 上本地的 cache 文件
   /home/hadoop/uidcache_etl：原始 uid—innerUid 对应，二进制文件，存储的格式为 yuanshiUid MD5 之后的 8 个字节的 long+2 个字节的'\t'+4 个字节 int 的 innerUid+2 个字节的'\n'。
   /home/hadoop/60days_active_users：60 天内的活跃用户，用于 etl 重启和每天第 0 个项目时候，ref 属性的重建。

6. mysql 和 hbase 的格式
   mysql：每个项目一个库，16_{pid}，每新建一个项目，系统会默认建 23 张

表，包括 sys_meta，冷表 cold_user_info，和 21 个系统属性表，这 21 个系统属性的相关信息也会存在 sys_meta 中。

hbase：每个项目一张表，deu_{pid}。

7. 依赖的 xingcloud lib：

dbutil：mysql 所有的建表操作，初始化项目

```
<dependency>
    <groupId>com.xingcloud.xa</groupId>
    <artifactId>dbutil</artifactId>
    <version>1.3</version>
</dependency>
```

hashutil：对所有节点的 hash 操作

```
<dependency>
    <groupId>com.xingcloud.xa</groupId>
    <artifactId>hashutil</artifactId>
    <version>1.2</version>
</dependency>
```

id_service：原始 uid 和 innerUid 对应关系

```
<dependency>
    <groupId>com.xingcloud</groupId>
    <artifactId>id_service</artifactId>
    <version>1.0</version>
</dependency>

<dependency>
    <groupId>com.xingcloud</groupId>
    <artifactId>cache</artifactId>
    <version>0.0.4</version>
</dependency>

<dependency>
    <groupId>com.xingcloud</groupId>
    <artifactId>basis</artifactId>
    <version>0.0.5</version>
</dependency>
```

forexutil：支付汇率兑换

```
<dependency>
    <groupId>com.xingcloud.forexutil</groupId>
    <artifactId>xa.forex.getter</artifactId>
    <version>1.0.0</version>
</dependency>
```

8. 143 上的 mongo

db：user_info
collection：dataloader_etl　存放了 etl 4 个机器的任务进行状态。
events_list　每个项目合法的事件，每个项目可以存放 5W 个

9. 134 上的几个服务

- idservice：
  /home/hadoop/id_service
  1080 的 thrift 接口，提供原始 uid 和 innerUid 对应关系查询和建立的接口。etl 使用 IDClient.getInstance().getCreatedId(project, rawUid)，如果 rawUid 存在则返回对应的 innerUid，不存在就返回创建的 innerUid。
  mysql 里面：
  每个项目对应一个数据库，vf_{pid}，比如 age 对应 vf_age。
  vf_{pid}里只有一张表，id_map，表结构为：

```
+---------+--------------+------+-----+---------+-------+
| Field   | Type         | Null | Key | Default | Extra |
+---------+--------------+------+-----+---------+-------+
| id      | int(11)      | NO   | PRI | NULL    |       |
| orig_id | varchar(255) | YES  | UNI | NULL    |       |
+---------+--------------+------+-----+---------+-------+
```

  两列，id 为 innerUid，orig_id 为原始 Uid。id 列自增长。
- 延迟 log 处理
  134：/home/hadoop/ivytest/DelayLogServer
  svn 地址：
  http://svn.xinggeq.com/svn/elex_analytics/branches/DelayLogServer
  basecamp 上的说明：
  https://basecamp.com/1850924/projects/463134-analytics/documents/2039897-log
  142-145 上运行了延迟 log 处理的客户端，DelayClient。DelayClient tail etl 每天生成的延迟 log 的文件，发送给 134 的 ProcessDelayLog 进程。
  延迟 log 的处理流程为：

```
[hadoop@ELEX-LA-DB2 DelayLogServer]$ crontab -l
#prepare delay log mysql

0 10 * * *      ssh 192.168.1.61 perl /home/hadoop/flush_redis.pl
30 10 * * *     java -classpath /home/hadoop/ivytest/DelayLogServer/dist/delayserver.jar com.xingcloud.dumpredis.DumpRedis dumptomysql
0 14 * * *       java -classpath /home/hadoop/ivytest/DelayLogServer/dist/delayserver.jar com.xingcloud.dumpredis.DumpRedis processdelay

0 16 * * *      ssh 192.168.1.61 perl /home/hadoop/flush_redis.pl
30 16 * * *     java -classpath /home/hadoop/ivytest/DelayLogServer/dist/delayserver.jar com.xingcloud.dumpredis.DumpRedis dumptomysql
0 20 * * *       java -classpath /home/hadoop/ivytest/DelayLogServer/dist/delayserver.jar com.xingcloud.dumpredis.DumpRedis processdelay


0 0 * * *       ssh 192.168.1.61 perl /home/hadoop/flush_redis.pl
30 0 * * *      java -classpath /home/hadoop/ivytest/DelayLogServer/dist/delayserver.jar com.xingcloud.dumpredis.DumpRedis dumptomysql
0 4 * * *       java -classpath /home/hadoop/ivytest/DelayLogServer/dist/delayserver.jar com.xingcloud.dumpredis.DumpRedis processdelay
```

  需要三步：
  - ssh 192.168.1.61 perl /home/hadoop/flush_redis.pl
    flush 61 上的 redis 到本地，生成 rdb 文件。
  - dumptomysql
    scp 61 上 flush 的 redis rdb 文件到 134，解析并将 redis 里面的 key 按照格式存进 mysql，生成 cache 和 filter 表。
  - processdelay

发送处理信号到 redis，ProcessDelayLog 从 redis 拿到处理信号后发起一轮延迟 log 的处理。

10. GEOIP 处理
    etl 中类 GeoIPCountryWhois，提供 ip→country 的转换。
    数据文件来自 maxmind http://dev.maxmind.com/geoip/legacy/geolite/
11. 监控
    etl log: /data/log/dataloader_etl.log
    flush log: /data/log/dataloader_flush.log
    检测 ERROR
    常用的查看 Log：
    cat /data/log/dataloader_etl.log|grep "end run Task"|tail　看 etl 执行到哪个任务
    tail /data/log/event16config/sendlog.process　event flush 进度
    tail /data/log/user16config/sendlog.process　user flush 进度