十年架构 成长之路

SACC 第十届中国系统架构师大会

SYSTEM ARCHITECT CONFERENCE CHINA 2018

2018年10月17-10月21日 北京海淀永泰福朋喜来登酒店

海量实时用户标签存储引擎Uindex架构与实践

周建平@广东数果

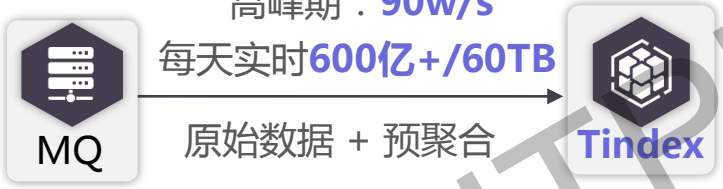
公司介绍

SugoIO 数果智能

广东数果科技有限公司　专业的智能大数据分析服务公司

实时用户行为分析

分析加速引擎(Tindex)



16台物理主机
(32Core，128GB，1TB * 10)

实时用户画像

标签引擎(Uindex)

7千万用户，上千维度

准实时标签

5台物理主机
(32Core，128GB，1TB * 10)

实时日志分析

分析加速引擎(Tindex)

300多台服务器日志
每天实时75亿/10TB+
日志数据高峰期17 万/秒
单条日志可达到数百K
无丢失，无延时。

15台Ucloud云主机
(16Core，32GB，1TB * 4)

感性认知用户画像

- 用户属性
  - 男
  - 30~40岁
  - 广东省
  - 深圳
  - iphone
  - 身份证件
- 浏览偏好
  - 倾向产品PN000005
  - 起购金额10-20万
  - 风险等级R3
- 用户价值
  - 平均理财金额1500元
  - 投资者等级C2
  - 半年下了8单
- 购买偏好
  - 倾向产品PN000005
  - 起购金额10-20万
  - 风险等级R3
- 用户行为
  - 注册到现在有120天
  - 列表页→详情页（≥2分钟）
  - ……
- 来源渠道
  - 世界杯营销活动
  - 广告联盟推广

为什么要建立用户画像

- 查询高价值用户
- 查询所推商品目标客户

- 重点维护高价值用户
- 提高客户忠诚度
- 提升重复购买率
- 提升客户ARPU值

- 流失会员站外购买偏好分析
- 站内站外购买偏好对比
- 流失预警

实时用户画像

- 基于实时行为通过算法识别出各级别**用户群特征**，然后对用户进行**360度特征刻画**，且打上**客户兴趣偏好标签**，通过合适的渠道给潜在目标客户群**个性化推荐合适的产品及服务**

用户360度画像

识别潜在高价值客户

推荐个性化产品及服务

行为风险预测与控制

通过精准推荐，使得产品推广及服务效果得到有效提升

场景验证

业务一：资讯App，推送资讯内容

场景人群：最近20分钟～1小时，需要学车人群、需要买车人群等，推送相关内容，最高点击率超过20%。

标题	内容	用户类型	推送时间	目标数	有效数	推送数	接收数	展示数	点击数	点击率
钻石、星耀、王者各个段位	都应该禁什么英雄?	IMEI	2017-10-25 15:54:05	4154	3929	3929	3797	3518	220	6.25%
日本产妇分享医院月子餐	不用担心坐月子了	IMEI	2017-10-25 15:51:57	511	485	485	460	434	30	6.91%
健身入门：	学习使用健身房的器械	IMEI	2017-10-25 15:50:51	466	423	423	411	383	43	11.23%
10月份最严驾考新规很难受?	好消息，未来可能不用考驾驶证了	IMEI	2017-10-25 15:49:11	509	488	488	469	445	103	23.15%
为何说买车要买低配	原来厂商真实的造车成本是这样算的	IMEI	2017-10-25 15:48:50	223	214	214	208	197	34	17.26%
最强王者算什么	有本事你找个玩王者的女朋友	IMEI	2017-10-25 14:30:19	9233	8782	8782	8386	7637	534	6.99%
父母经常和孩子这样对话	会让孩子变成一个情商低的人	IMEI	2017-10-25 14:27:16	1226	1171	1171	1117	993	69	6.95%
20种你不常见的微胖体	菜鸟到大师，你能来几种	IMEI	2017-10-25 14:26:36	735	671	671	651	605	44	7.27%
10月份最严驾考新规很难受?	好消息，未来可能不用考驾驶证了	IMEI	2017-10-25 14:25:45	810	766	766	754	703	147	20.91%
为何说买车要买低配	原来厂商真实的造车成本是这样算的	IMEI	2017-10-25 14:24:39	343	311	311	307	291	60	20.62%

业务二：钱包App

场景人群：最近20分钟～1小时，需要贷款人群等，推送相关内容，点击率超过18%，无精准策略的对照组点击率为2%。

标题	内容	推送时间	目标数	展示数	点击数	点击率	备注
朋友在这成功借到5万	利息比同行低2~3倍，可借1年	2017-10-26 17:38:09	1026	871	145	16.65%	近20分钟
		2017-10-26 15:38:32	1099	951	159	16.72%	近20分钟
		2017-10-26 11:28:02	3082	2671	482	18.05%	近1个小时
		2017-10-25 17:34:18	50000	27077	582	2.15%	对照数据
		2017-10-25 17:34:18	3200	2736	494	18.06%	近1个小时

2017-10-25 17:34:18
基于实时用户画像：18.06%
传统无精准策略：2.15%

使用方式

微观画像	根据ID查询用户信息
用户筛选	单个标签过滤筛选用户
用户圈选	多个组合标签圈选用户
宏观画像	对标签数据的统计分析
关联用户群	基于已圈选用户群进行分析

标签数据特点

- 成千上万，数量会不断增加
- 不同的标签更新频率不同，每月，每周、每天、每小时、每十分钟
- 大量空缺的标签导致数据非常稀疏

传统数据库方案

根据业务垂直分割：

用户基本属性				
用户ID	性别	年龄	学历	职业
001	男	28	本科	程序员
002	女	35	硕士	产品经理
003	不详	31	博士	研究员

用户价值			
用户ID	有车	有房	房估值
001	否	否	0
002	是	是	三百万
003	是	是	五百万

……

适用场景：
数据量少，业务场景固定

优点：
方案简单，所有数据库都支持

缺点：
跨表的查询慢
每列建索引，更新慢
无法满足动态增删标签

传统数据库方案

竖表&水平分割：

用户ID	标签名	标签值
001	sex	男
001	age	25
001	has_car	N
002	sex	女
002	age	32
002	has_car	Y

适用场景：
数据量少，支撑标签动态增删

优点：
支持稀疏数据
更新方便

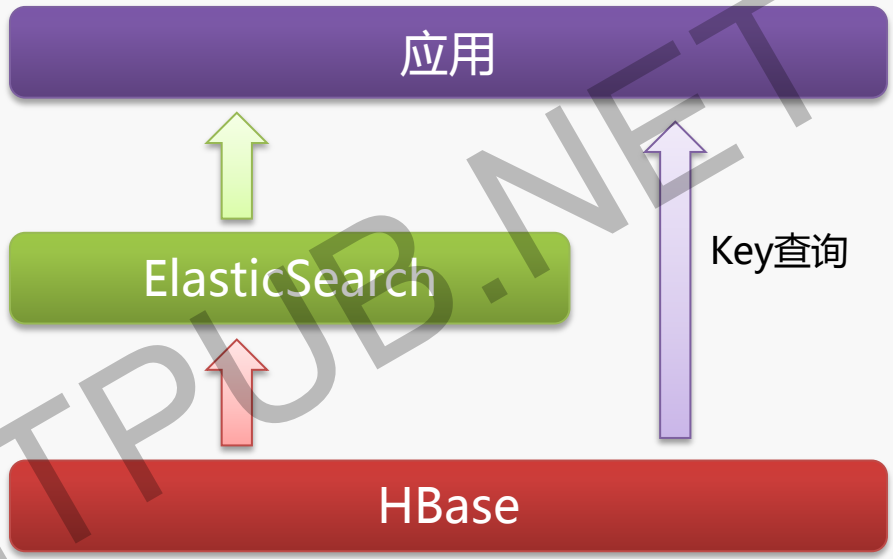
缺点：
查询复杂，尤其是组合查询
数据管理与维护麻烦
无法对用户群进行分析

HBase+ElasticSearch

标签查询
组合标签查询
宏观画像

标签筛选
组合标签

存储标签数据



HBase+ElasticSearch

宽表

ElasticSearch

- ✓ 使用Lucene，具有高效的查询能力
- ✓ 支持组合查询和条件过滤
- ✓ 提供聚合函数，支持统计分析功能
- ✓ 面向文档，可不断动态增加标签

- x 写入性能差，无法满足实时更新需求
- x 大数据量更新时容易丢失数据
- x OOM和脑裂问题困恼

+

竖表

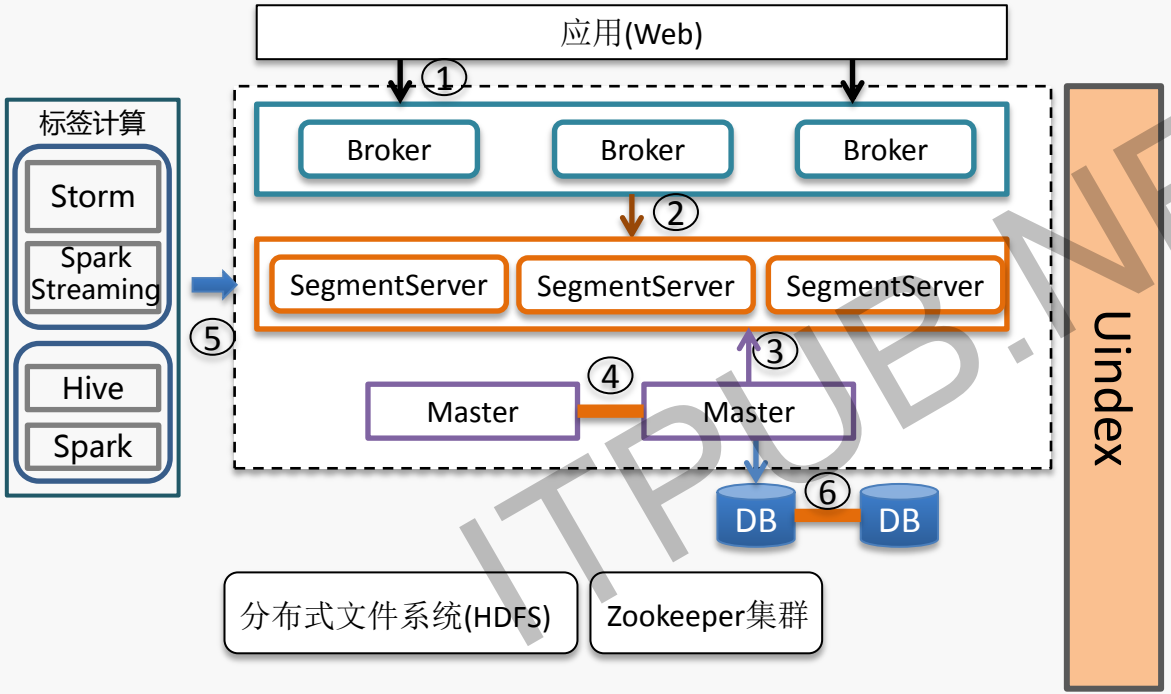
Hbase

- ✓ 实时读写、随机访问超大规模数据量
- ✓ 根据业务设计RowKey，可动态增加标签
- ✓ 适于存储稀疏的标签数据
- ✓ 良好的系统伸缩性、高容错性

- x 不支持数据类型
- x 按RowKey查询，不能支持条件查询
- x 难以支持群体画像分析和定向
- x 支持的标签数量通常不超过一千个

- x 两者写入性能相差太远，实时写入时数据同步存在很大问题
- x 不同场景使用不同的平台，增加了系统复杂度与维护难度

实时标签存储引擎Uindex



实时更新

能够支持标签数据的列级别实时更新

高性能

能够支持千万级用户的高效写入和查询导出

高可用

Uindex中所有的管理节点均有HA

水平扩展

采用无共享的设计和实现，随数据的增加可以无限制的水平扩展

实时标签存储引擎Uindex

高效强大检索能力

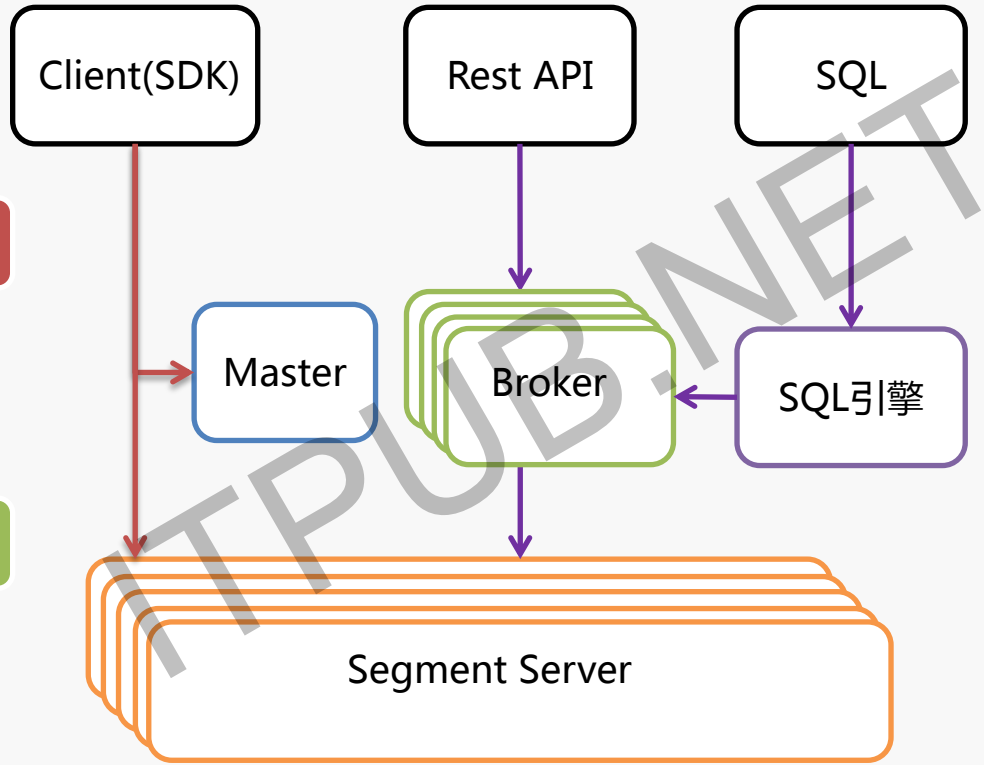
+

列级别实时更新

Uindex整合了“搜索引擎”对数据的检索能力，以及列级别数据更新能力，能够有效地支持实时精准投放需求

大幅提升数据写入和查询性能的同时，有效支持实时精准投放

读请求



Get

- 根据主键查询
- 高并发、低延迟
- 实时

Scan

- 批量导出

多种查询类型

- Select
- groupby
- timeseries

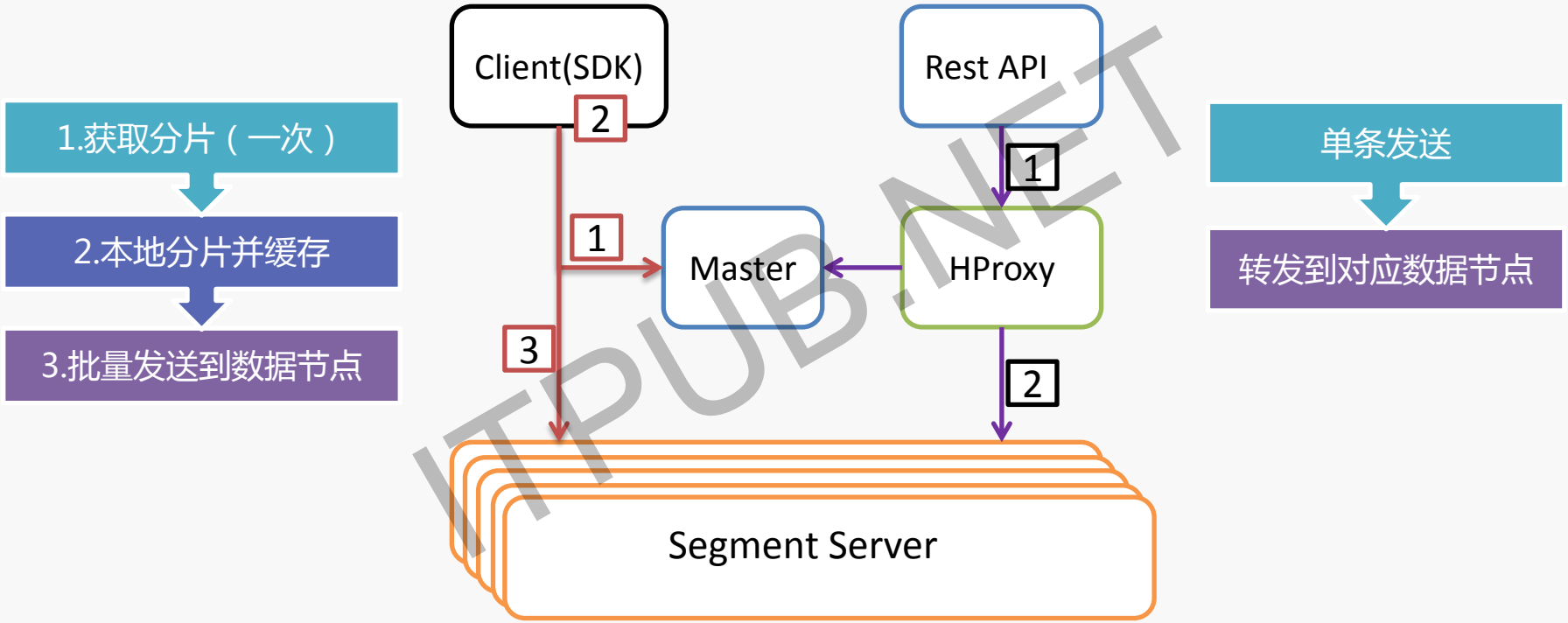
多种条件过滤

- 日期/数字/坐标范围
- 精确/正则/模糊匹配
- 空值/非空/非等匹配

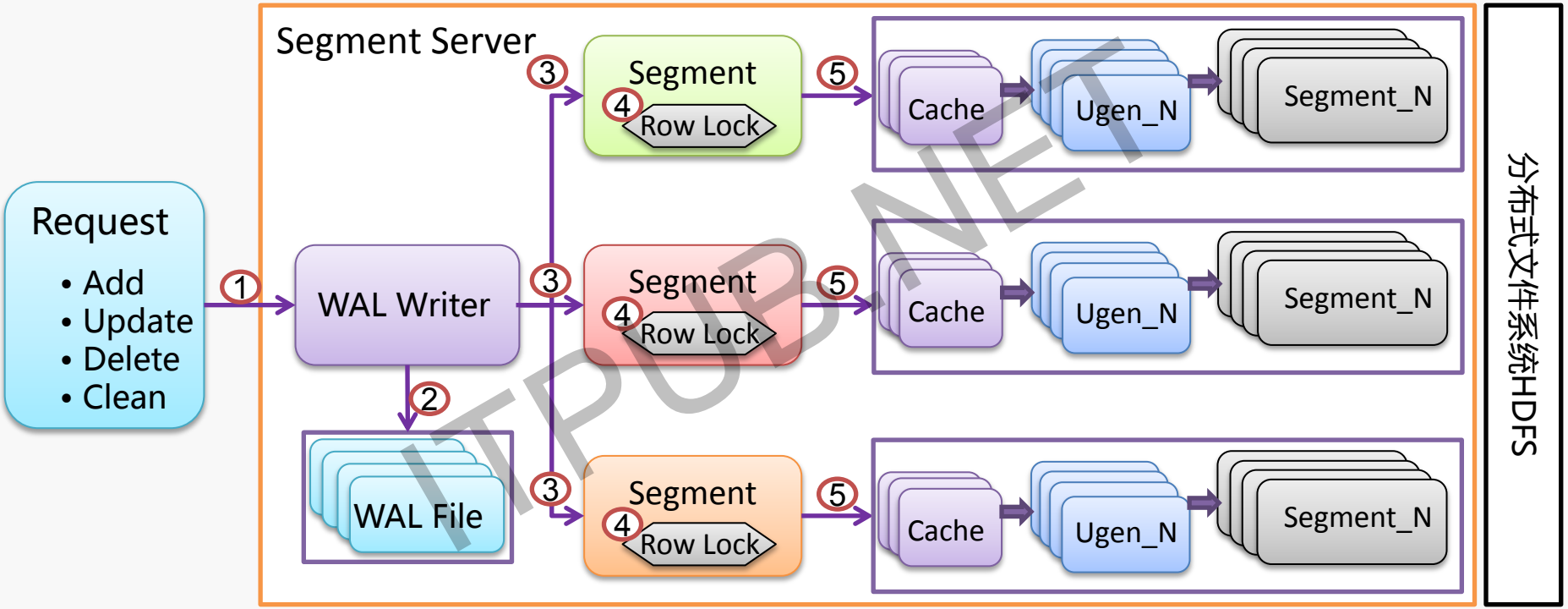
多种聚合

- count、sum、min、max、cardinality等

写请求



写处理



列级更新

段数据 Segment_1

docId	Id	Name	Score
11	1011	Tom	88
12	1012	Jack	70
13	1013	Son	90

更新代 Ugen_1

docId	Name	Score
11	John	75
12		80
13	Smith	

更新代 Ugen_2

docId	Name	Score
11	Bob	
12		90
13		85

读取过程：

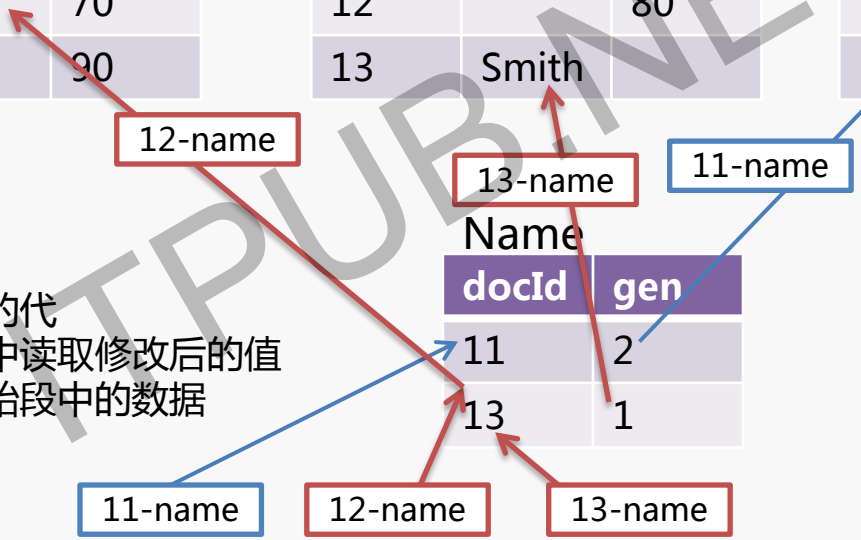
1 判断是否存在更新

2 如果存在

2.1 首选读取最新更新的代

2.2 然后到相应更新代中读取修改后的值

3 如果不存在，则读取原始段中的数据

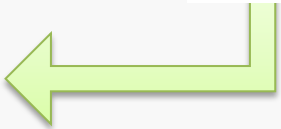


列级更新

Id	Name	Score
1001	Tom	88
1002	Jack	70
1003	Son	90

Id	Name	Score
1001	Tom	88
	Ugen_1	86
1002	Jack	70
1003	Son	90

Id	Name	Score
1001	Tom	88
	Ugen_2	92
	Ugen_1	86
1002	Jack	70
1003	Son	90



Id	Name	Score
1001	Tom	88
1002	Jack	92
1003	Son	90

性能优化 – JVM参数

堆内存分配少于32g

- $ java -Xms32768m -Xmx32768m -Xmn50m Memory
compressedOops : false

32g * 1024 = 32768m

- $ java -Xms32760m -Xmx32760m -Xmn50m Memory
compressedOops : true

对象头OBJECT_HEADER　　OBJECT_REF + 8;
对象引用 OBJECT_REF　　compressedOops ? 4 : 8
数组头ARRAY_HEADER　　OBJECT_HEADER + NUM_BYTES_INT

```
private static class Entity {
   public String uid;
   public String name;
   public Double score;
   public Integer age;
 }
```

⇒ LinkedList

-Xms & -Xmx	compressed	Result
31g	true	588,555,700
32g	false	385,833,800
48g	false	579,446,830
31g	false	373,734,440

-XX:-UseCompressedOops

性能优化 – JVM参数

新生代堆内存调整到相应大小，避免对象进入老年代

-XX:NewSize=12G -XX:MaxNewSize=12G

在数据查询过程中使用DirectByteBuffer Pool，提高性能，避免mirror GC

-XX:MaxDirectMemorySize=6G

性能优化 – 向量化

```
int sum = 0;
for (int i = 0; i < CNT; i++) {
    sum += i;
}
```

```
for (int i = 0; i < CNT; i+=4) {
    sum0 += i;
    sum1 += i + 1;
    sum2 += i + 2;
    sum3 += i + 3;
}
int sum = sum0 + sum1 + sum2 + sum3;
```

测试结果-数据导入

参考指标

按照2小时内导入6000万行数据的参考标准，希望导入速度达到10000行/秒；

```
-- 全量（6000 万行）数据，上千个维度
insert into t_userprofile
values (imei, tag1, tag2, tag3, ..., tagN)
```

5台机器：32cpu，128G内存，1TB * 10

全量写入		
	ElasticSearch	Uindex-a
机器数	15	5
数据量	6000万+	6000万+
维度数	1000+	1000+
耗时	六七个小时	70分钟
TPS	2778	14285

测试结果-实时更新

参考指标

使用实时流计算标签，实时更新标签，更新的字段数比较少，看性能能否比全量导入有明显的提升，目标大于20000行/秒

```
-- 根据 primary key 对少量字段（标签）更新
insert into t_userprofile (imei, tag)
values ('imei_xxxxxxx', 'value')
on uplicate update set tag = values(tag)
```

5台机器：32cpu，128G内存，1TB * 10

更新性能		
	更新10维度	更新20维度
数据量	1000万	1000万
更新耗时	5分钟	5分钟
资源消耗	12%cpu，30G内存	15%cpu，30G内存
TPS	3.3万	3.3万

业务场景

- 基于实时用户行为对用户打标签
- 使用实时流计算引擎计算并更新用户标签

测试结果-人群导出

参考指标: 目前使用用 ES 导出，导出 1000 万记录大大概 5-10分钟
希望 Uindex 起码要在同级别，甚至更优

1.有些导出场景下，除了输出主键(imei)，还需要输出指定的若干个维度
2.输出的维度值可能需要转码，例如“性别”标签需要把0,1转换成男，女

5台机器：32cpu，128G内存，1TB * 10

个数	维度	100万耗时(s)	1000万耗时(s)
1	umid	4.7	45
2	umid、imei	9.8	69
3	umid、imei、sn	13.6	115
4	umid、imei、sn、uid	14.1	122
6	umid、imei、sn、uid、recharge_7d、wallpaper	15.6	126

测试结果-根据id查询用户

1台机器：32cpu，128G内存，1TB*10

标签数	并发数	请求次数	耗时（毫秒）	QPS
2	20	2万	3012	6640
6	20	2万	3180	6289

测试结果-SQL查询

5台机器：32cpu，128G内存，1TB*10

说明	Sql语句	耗时
统计记录数	select count(*) from update_test;	84(ms)
统计记录数 （一个过滤条件）	select count(*) from update_test where startup_themes='start_1';	280(ms)
分组统计行数并排序	select themes_set_ring, count(*) from update_test group by themes_set_ring order by themes_set_ring;	560(ms)
分组统计行数并排序 （一个过滤条件）	select themes_set_ring, count(*) from update_test where push_sub_apps='sub_8' group by themes_set_ring order by themes_set_ring;	68(ms)

测试结果-用户群关联

5台机器：32cpu，128G内存，1TB*10

说明	Sql语句	耗时
关联200w用户群 统计行数	select count(*) from update_test where umid in umid_200w_lookup;	3842(ms)
关联200w用户群 分组统计行数并排序	select themes_set_ring, count(*) from update_test where umid in umid_200w_lookup group by themes_set_ring order by themes_set_ring;	3715(ms)

业务场景

查看高价值用户群的风险等级偏好

高价值用户列表可以是通过系统圈选出来或外部导入

THANKS

数果大数据

sugo.io