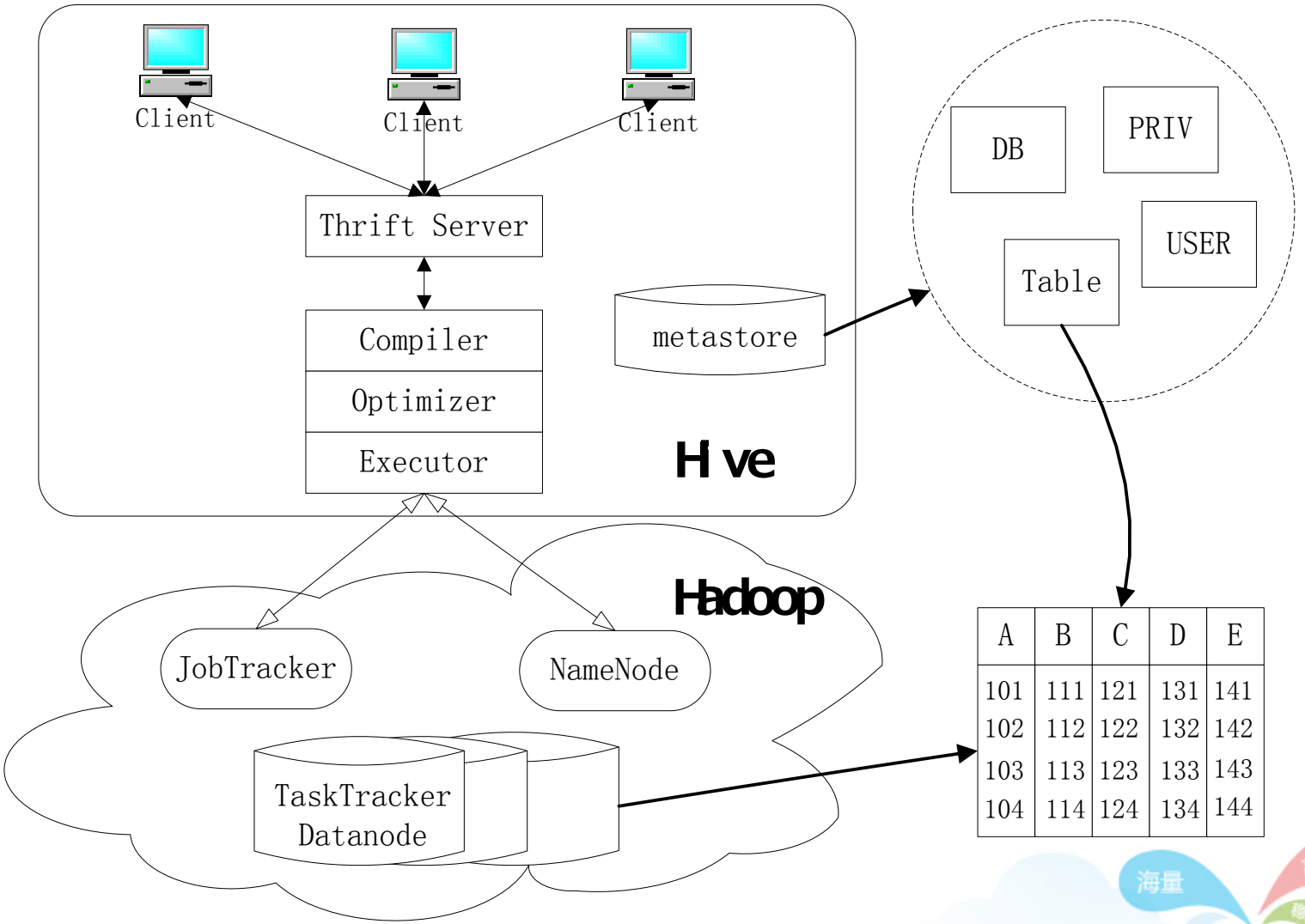
TDW protobuf存储格式功能介绍

马淑婧
cherrysjma

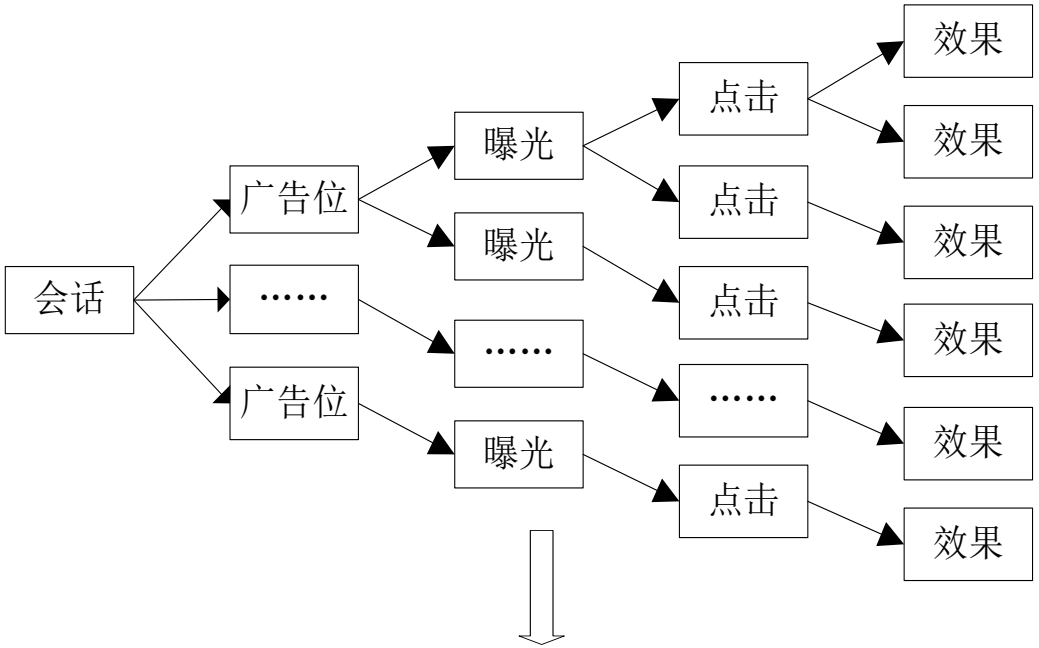
Tencent 腾讯 TDW 腾讯分布式数据仓库 TENCENT DISTRIBUTED DATA WAREHOUSE

目录

TDW查询引擎


Client
Client
Client
Thrift Server
Compiler
Optimizer
Executor
metastore
Hive
DB
PRIV
USER
Table
Hadoop
JobTracker
NameNode
TaskTracker
Datanode
A B C D E
101 111 121 131 141
102 112 122 132 142
103 113 123 133 143
104 114 124 134 144

广点通日志模型带来的问题



会话 ID	广告位 ID	曝光 ID	点击 ID	效果 ID
1	10	101	1011	10111
1	10	101	1011	10112
1	10	101	1012	10121
1	10	101	1012	10122
1	10	102	1021	10211
1	10	102	1021	10212
1	10	102	1022	10221
1	10	102	1022	10222

- **完整会话信息无法直接获取**，结构化数据存储，一个会话信息变为了多条记录。
- **存储冗余**，越上层的数据冗余越多。
- **表模式修改困难**，行式存储，只能在表的末尾增加列。

目录

Protocol Buffers

- Google定义的一种数据存储格式
  - 描述数据格式的IDL，支持结构体和嵌套
  - 对IDL描述格式进行编码的二机制编码方案
  - 代码生成器用于生成二进制编解码代码
- Protocol Buffers的优势
  - 存储占用小
  - 编解码速度快
  - 编解码代码简单易书写，而且平台及语言无关
- Protocol Buffers目前的应用情况
  - 广泛用于通信协议和数据存储领域

Protocol Buffers

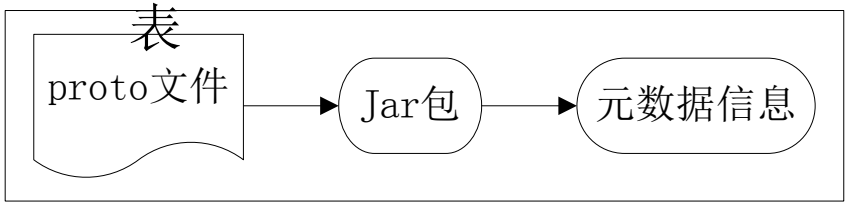

```
message Session{ //会话
    optional string id = 1;
    repeated Position positions =
    10;
}
message Position { //广告位
    optional string id = 1;
    repeated Impression imprs =
    10;
}
message Impression{ //曝光
    optional string id = 1;
    repeated Click clicks = 10;
}
```

```
message Click { //点击
    optional string id = 1;
    repeated Trace traces =
    10;
}
message Trace{ //效果
    optional string id = 1;
    repeated string effect=
    10;
}
```

目录

Protobuf表转换规则

通过proto文件生成的jar包获取元数据信息进行建表



表名

字段

字段类型

类型转换规则

1. 支持protobuf中message定义的嵌套
2. Protobuf中enum 类型转化为tdw的int类型
3. repeated类型转化为tdw的array
4. 嵌套message类型转化为tdw的struct类型

```
table session{
    id string,
    positions
     array<struct<id:string,

    imprs:array<struct<id:s
    tring,

    clicks:array<struct<id:
    string,

    traces:array<struct<id:
```

创建Protobuf表

- 上传Proto定义文件
  - 将proto文件上传到$QE_HOME/protobuf/upload/${UserName}/中

session.proto

```
import "position.proto"
message Session{ //会话
    optional string id = 1;
    repeated Position
     positions = 10;
}
```

position.proto

```
import "impression.proto"
message Position { //广告位
    optional string id = 1;
    repeated Impression
     imprs = 10;
}
```

trace.proto

```
message Trace{ //效果
    optional string id =
     1;
    repeated string
     effect= 10;
}
```

click.proto

```
import "trace.proto"
message Click { //点击
    optional string id =
     1;
    repeated Trace traces
     = 10;
}
```

impression.proto

```
import "click.proto"
message Impression{ //
    曝光
    optional string id =
     1;
    repeated Click
     clicks = 10;
}
```

创建Protobuf表

- 上传Proto定义文件
  - UserName为TDW建表用户名，若upload或用户目录不存在使用mkdir命令创建
  - 如果一张表对应有多个proto文件，也就是说主proto文件中import了其他proto文件，需要将所有的proto文件都上传到目标目录

```
$QE_HOME/protobuf/upload/test> ls
click.proto   impression.proto   position.proto   session.proto
     trace.proto
```

创建Protobuf表

- 准备Jar文件
  - $QE_HOME/bin/makejar.sh pgurl user passwd dbname tablename username filename protoversion
  - 多个proto文件会合并成为一个(表名.proto)，用来生成jar包(db名_表名_建表时间.jar)
  - proto文件和jar包均被上传到pg库中

```
$QE_HOME/bin/makejar.sh pgurl user passwd testpb
    session test session.proto 2.3.0
```

创建Protobuf表

- 准备Jar文件
  - 用makejar脚本预处理proto文件，产生并上传对应表的读写接口jar包
  - pgurl，user，passwd：保存jar报的元数据库信息，应该分别与$QE_HOME/conf中的hive.metastore.pbjar.url，hive.metastore.user，hive.metastore.passwd一致
  - dbname，tablename：要创建表在TDW中的db名和表名，注意表名要与主message名相同
  - username：建表用户名
  - filename：主proto文件名，这个文件及其包含的文件（如果有的话）应该已经拷贝到指定的目录中
  - protoversion：protobuf版本号，目前只支持2.3.0

创建Protobuf表

- 执行建表语句
  - 创建protobuf存储格式的TDW SQL语句是：
  - CREATE TABLE <tableName> [partition_def] STORED AS PB
  - 创建普通表（不包含分区）
    - create table session stored as pb
  - 创建带分区的表
    - 以session message中的id字段为分区字段建立分区表
    - create table session partition by list(id) (partition default) stored as pb

创建Protobuf表

- 执行建表语句
  - 创建protobuf格式的表前必须保证上面的makejar命令成功
  - 当前db必须与之前makejar命令指定的dbname相同，tableName也必须与之前makejar命令指定是tablename相同。
  - 在创建protobuf存储格式的表时，也可以同时定义分区，定义方法与普通表一样
  - 用describe extended + 表名来查看建表成功后表的元数据结构和分区信息
  - 增删分区、读写数据与普通表语法一致
  - 不支持通过sql增加或修改字段

创建Protobuf表

- proto文件定义注意事项
  - proto文件名一定要是小写，并且不能包含空格等特殊字符
  - proto文件中用到import其他proto文件的，不要写路径，只指明文件名即可，例如import "position.proto"
  - 主proto文件的message名字一定要与表名相同，根据proto文件生成jar包的时候会进行检查，不相同会报错
  - 自定义的类型名和变量名不能相同（支持区分大小写，即message A类型的变量名可以为a），否则生成jar包会失败
  - 不能包含空的message，否则建表的时候会出错

PB数据计算

- Lateral View + explode 实现对protobuf表中repeated字段做sql计算
- 假设广告展示表pageAds的定义中每个广告展示的记录由一个页面的id和当前页面上展示的广告几个id的list组成，其proto定义为：

```
message pageAds{
required string pageid =
    1;
repeated int adid_list
    =2;
}
```

- 当前表pageAds中有如下数据：

string pageid	Array<int> adid_list
"front_page"	[1, 2, 3]
"contact_page"	[3, 4,]

PB数据计算

- 对表做lateral view + explode 的SQL

```
SELECT pageid, adid FROM pageAds
    LATERAL VIEW
    explode(adid_list) adTable AS
    adid;
```

- 可以产生如下的输出：

string pageid	int adid
"front_page"	1
"front_page"	2
"front_page"	3
"contact_page"	3
"contact_page"	4

- 对repeated字段explode后的结果做方便的计算

```
SELECT adid, count(1)
FROM pageAds LATERAL VIEW
    explode(adid_list) adTable
    AS adid
GROUP BY adid;
```

int adid	count(1)
1	1
2	1
3	2
4	1

PB数据计算

- repeated字段的值可能为空，在TDW中为explode增加了设置缺省值的功能
  - 不Map和Struct类型，支持List类型
  - 对于基本类型可以支持int到bigint，float到double的自动类型提升

explodetest

int id	Array<int> test
2	[3, 4, 5]
3	null

```
select id, testid from
  explodetest LATERAL
  VIEW explode(test)
  testdd as testid;
```

int id	int testid
2	3
2	4
2	5
3	null

```
select id, testid from
  explodetest LATERAL VIEW
  explode(test,100) testdd
  as testid;
```

int id	int testid
2	3
2	4
2	5
3	100

目录

Protobuf表元数据存储

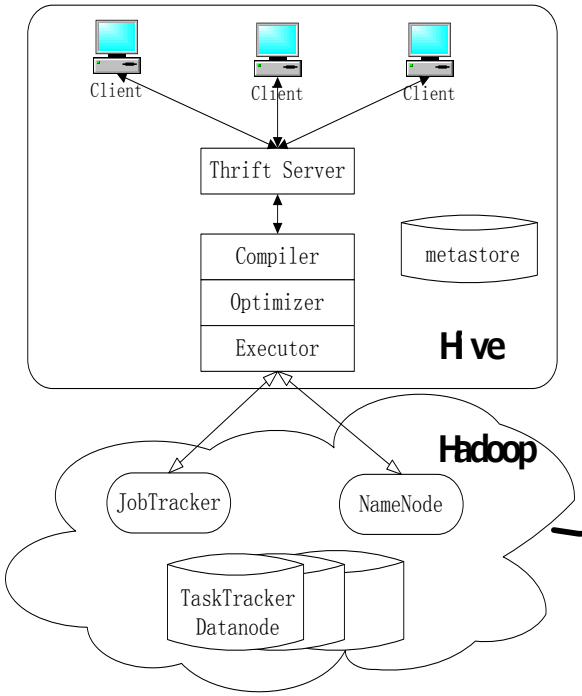
- proto和jar文件存储
  - 处理protobuf表和数据均依赖jar包，存储在pg库中
  - 处理pb表时将对应jar包下载到$QE_HOME/auxlib目录下，hive重启时清空

```
CREATE TABLE pb_proto_jar
(
  db_name character varying NOT NULL,
  tbl_name character varying NOT NULL,
  proto_name character varying,
  proto_file bytea,
  jar_name character varying,
  jar_file bytea,
  user_name character varying,
  modified_time timestamp without time zone NOT NULL,
  protobuf_version character varying DEFAULT '2.3.0'::character varying,
  CONSTRAINT pb_proto_jar_pkey PRIMARY KEY (db_name, tbl_name, modified_time)
)
```

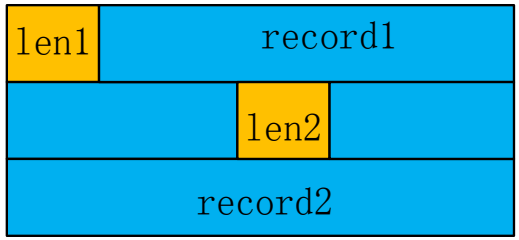
Protobuf表元数据存储

- proto和jar文件存储
  - db_name：pb表所在db名
  - tbl_name：pb表的表名
  - proto_name：由makejar合并而成的proto文件的名字(表名.proto)
  - proto_file：proto文件的内容，二进制存储
  - jar_name：由makejar生成的jar包的名字(db名_表名_建表时间.jar)
  - jar_file：jar包的内容，二进制存储
  - user_name：建表用户名
  - modified_time：建表时间
  - protobuf_version：proto版本号，暂时只支持2.3.0
  - 通过pg命令可以下载二进制文件

PB存储格式及编解码



长度+PB记录



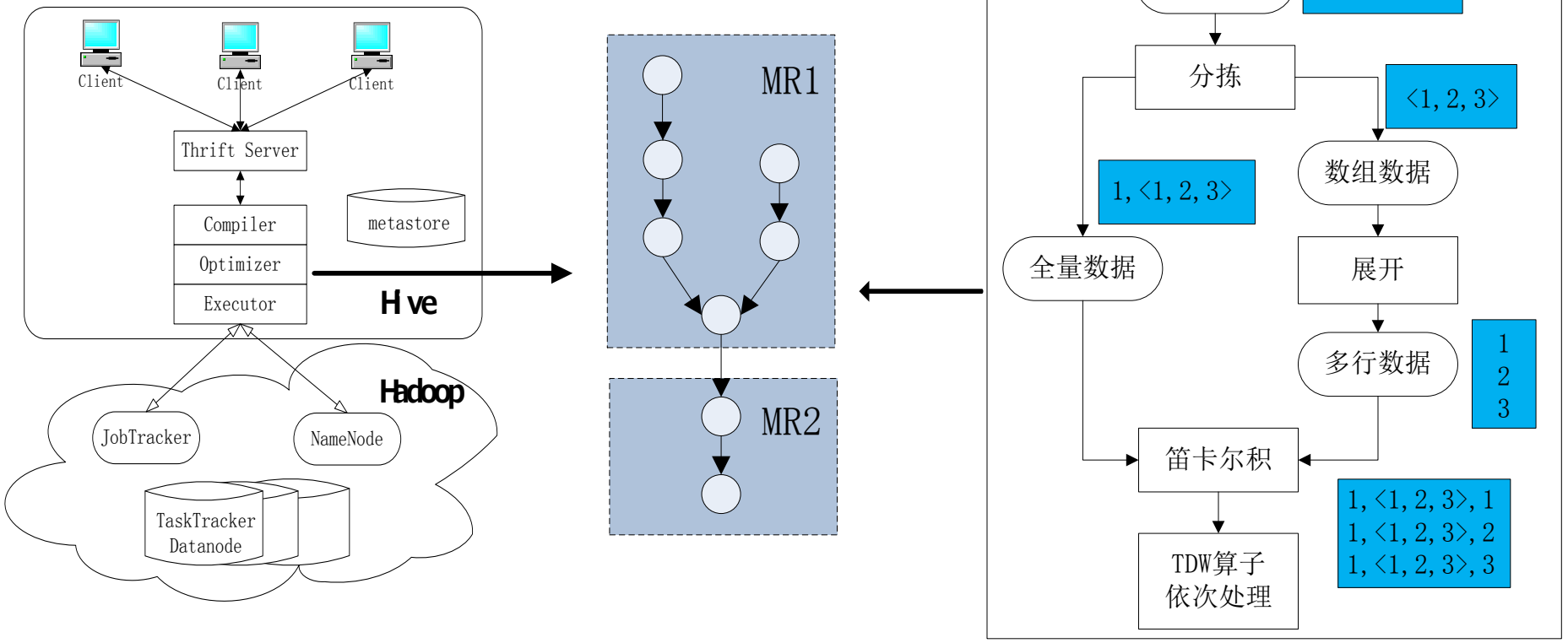
1. 一条完整的数据存储为一条PB记录
2. 记录间采用一个4字节的长度字段分隔

过滤错误Protobuf文件

- 过滤错误文件
  - PB数据容易出现错误文件，当错误文件的个数较少时，希望能够略过不进行统计，当达到一定数量时能够报警给业务进行处理
  - 增加一个counter，用于记录错误的PB文件个数，在job完成后读取该counter的数值，发现错误个数超过一定的阈值则报错
  - 把错误的文件个数记录到pg库中
  - `set hive.pb.badfile.skip = true`
  - `set hive.pb.badfile.limit = 10`

PB数据计算实现

对repeated字段进行展开以适应算子树的处理。



微信公共帐号

Q&A
Thanks！