Greenplum在TalkingData的应用实践

8/24/17 周明

TalkingData

Pivotal | IT大咖说 知识分享平台

目 录

1

TalkingData公司介绍

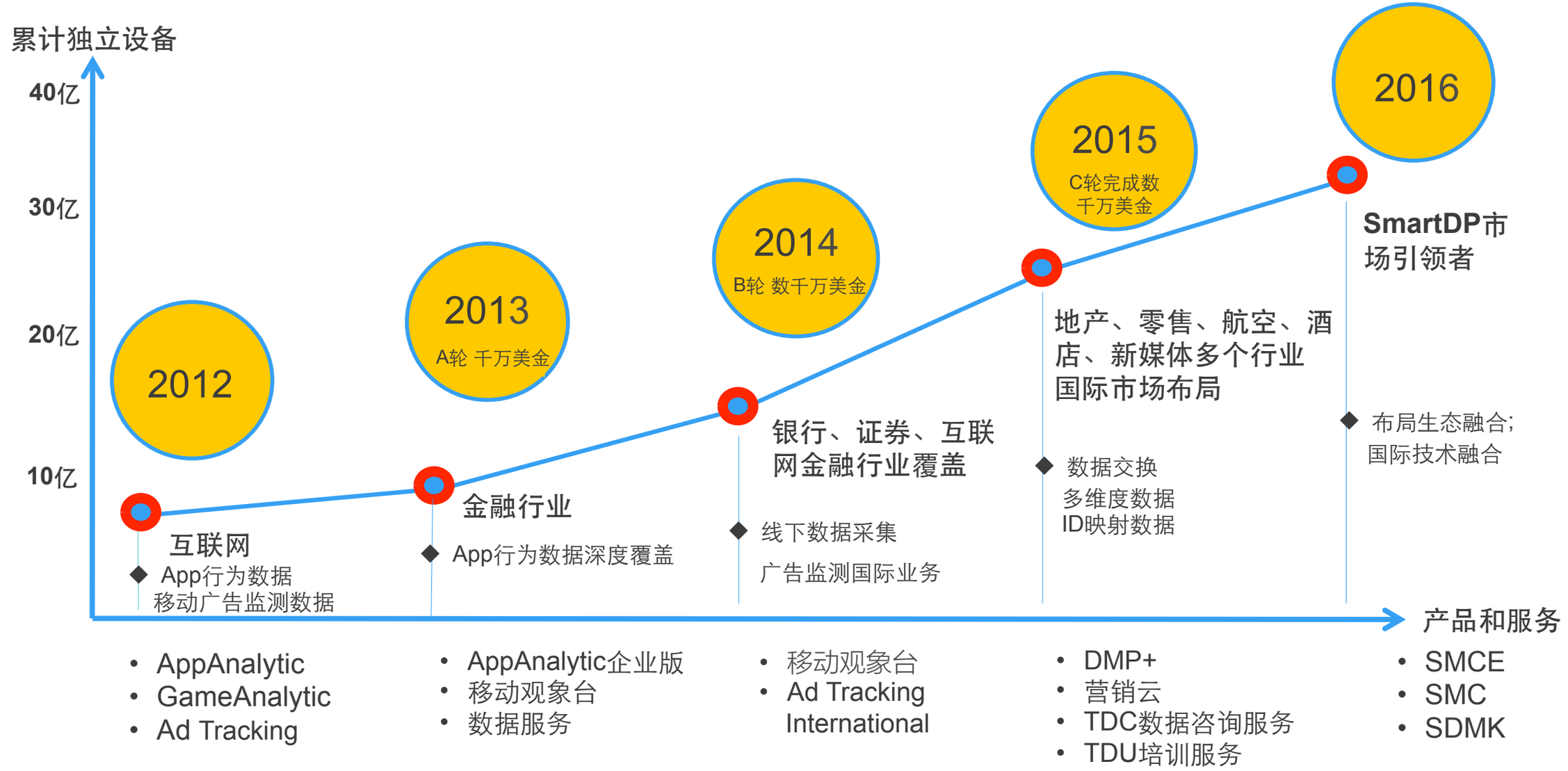

TalkingData
成立于2011年9月
国内领先的独立第三方数据服务提供商
帮助传统企业转型升级为数据驱动型企业
通过数据改善人类自身和环境

TalkingData主要客户列表

Pivotal | IT大咖说 知识分享平台

Mobile Apps	Financial Services	Real Estate	Retail	Auto	Airlines	Gov't	Hospitality	Media

五年高速发展


累计独立设备
40亿
30亿
20亿
10亿
2012
2013
A轮 千万美金
2014
B轮 数千万美金
2015
C轮完成数
千万美金
2016
SmartDP市场引领者
互联网
App行为数据
移动广告监测数据
金融行业
App行为数据深度覆盖
银行、证券、互联网金融行业覆盖
线下数据采集
广告监测国际业务
地产、零售、航空、酒店、新媒体多个行业国际市场布局
数据交换
多维度数据
ID映射数据
布局生态融合;
国际技术融合
产品和服务
AppAnalytic
GameAnalytic
Ad Tracking
AppAnalytic企业版
移动观象台
数据服务
移动观象台
Ad Tracking International
DMP+
营销云
TDC数据咨询服务
TDU培训服务
SMCE
SMC
SDMK

以SMC和SDMK平台为核心的数据生态体系

2

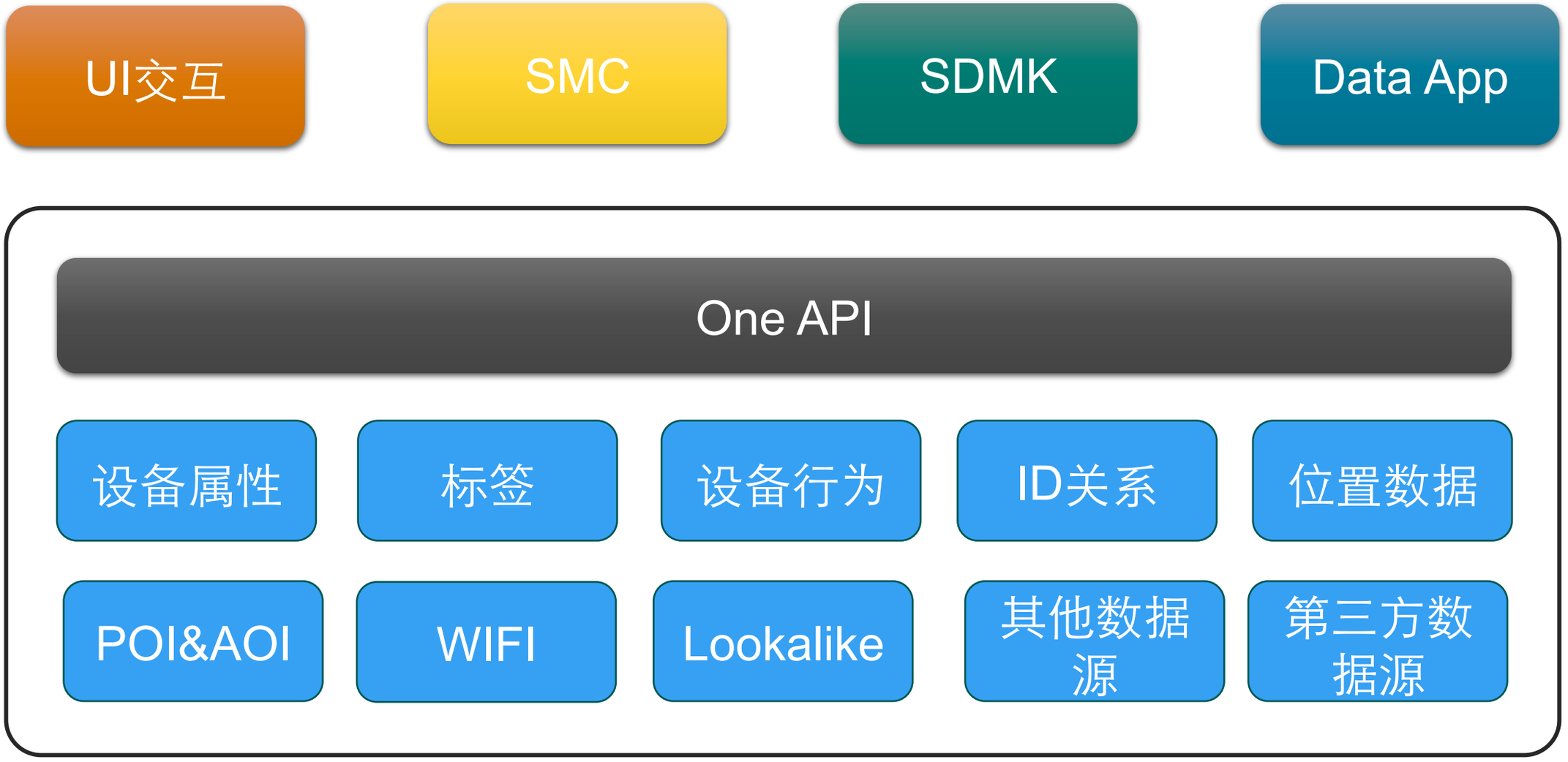
Data ATM

TalkingData

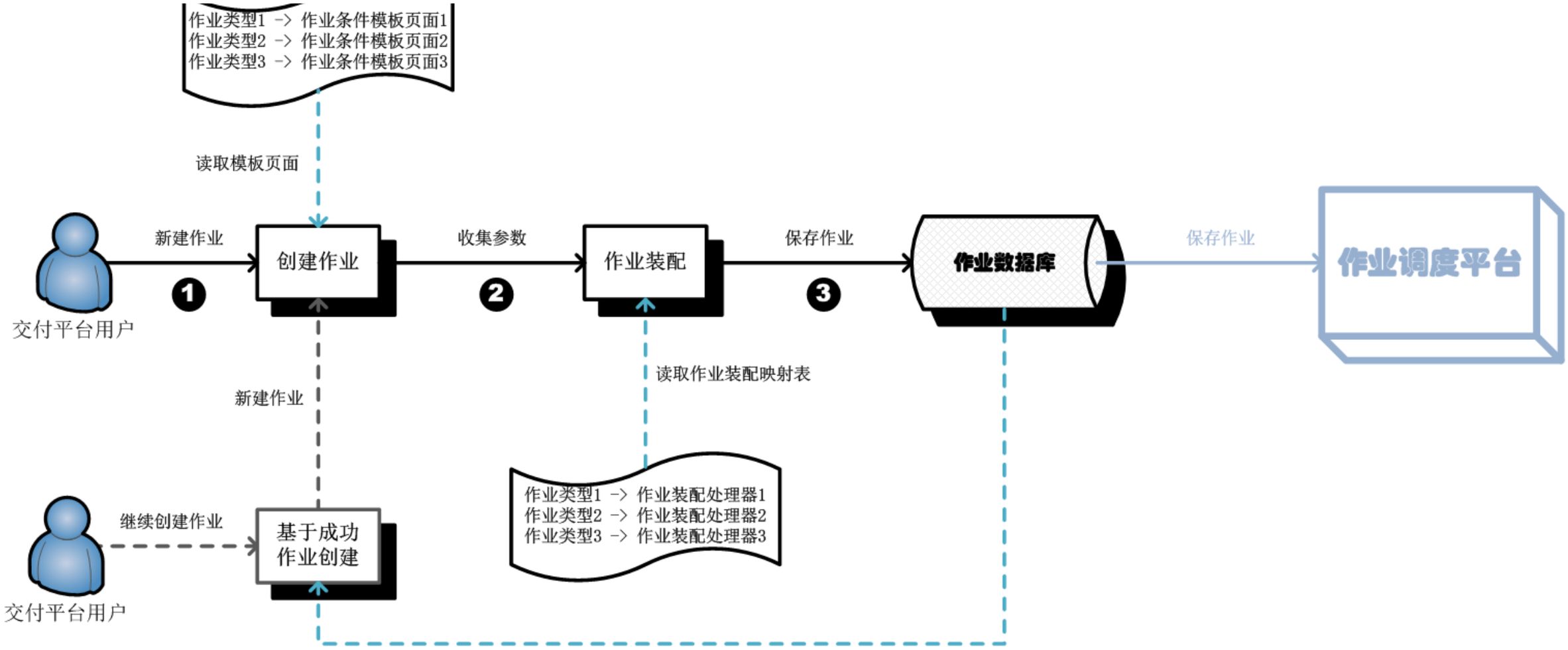
系统简介

- 人群定位与分析平台
- 基于多个数据源的不同条件，找出特定人群
- 对特定人群进行分析画像
- 为前端数据应用提供数据支撑

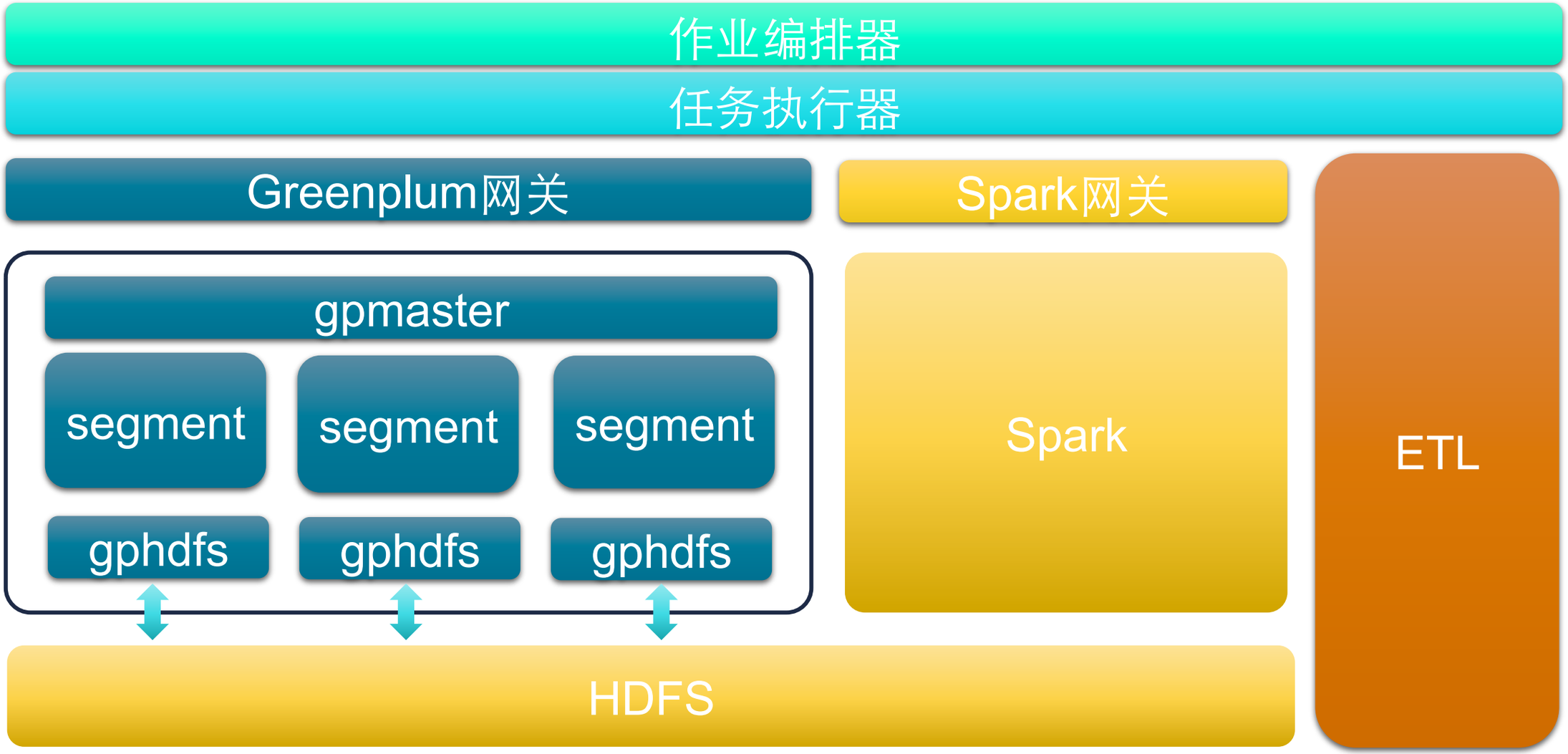
业务架构


UI交互
SMC
SDMK
Data App
One API
设备属性
标签
设备行为
ID关系
位置数据
POI&AOI
WIFI
Lookalike
其他数据源
第三方数据源

流程图


作业类型1 -> 作业条件模板页面1
作业类型2 -> 作业条件模板页面2
作业类型3 -> 作业条件模板页面3
读取模板页面
新建作业
创建作业
收集参数
作业装配
保存作业
作业数据库
保存作业
作业调度平台
交付平台用户
1
2
3
新建作业
读取作业装配映射表
作业类型1 -> 作业装配处理器1
作业类型2 -> 作业装配处理器2
作业类型3 -> 作业装配处理器3
继续创建作业
基于成功
作业创建
交付平台用户

技术架构


作业编排器
任务执行器
Greenplum网关
Spark网关
gpmaster
segment
segment
segment
gphdfs
gphdfs
gphdfs
Spark
ETL
HDFS

Pivotal | IT大咖说 知识分享平台

3

Greenplum

TalkingData

数据容量

- 总数据容量400TB
- 唯一设备数50亿+
- 日活跃设备数2.5亿
- 月活跃设备数6.5亿
- 月位置数据（聚集后）600亿
- 最大单个任务参与计算数据超过6TB
- 任务耗时1秒-5分钟

目前Greenplum集群规模

- 21个数据节点；
- 84个Primary Segment，84个Mirror Segment；
- 单节点配置：
  - 24Core CPU
  - 128GB RAM
  - 14 * SAS 7200 Disk Raid 5 = 45TB
- 目前存储数据量（纯文本）
  - 400TB+

Why use Greenplum?

- 得益于可控的数据分布，执行效率较高；
- 支持列式存储和压缩；
- 支持多样的数据类型：Array XML Json等；
- 支持GEO空间计算引擎PostGIS；
- 扩展性强，Function Operator DataType Aggregation Index都可以由用户来自定定义；
- 支持多种主流语言：PGSQL C Python Perl Java R；
- 使用标准的SQL语言，降低前端开发成本和时间

4

技术点分享

TalkingData

技术点

- Array & Json
- PostGIS
- Bitmap

Array & Json

- 通过使用数组缩减记录量，加快查询；
- 部分数据使用到了多维数组，原生支持较差；
- 多维数组解析通过UDF来实现；
- Greenplum5.0 支持Json格式；

Array 在标签数据上的使用

```
              tdid                |                                                                         tags
----------------------------------+-----------------------------------------------------------------------------------------------------------------------------------------------------
 2b1ed6b6c2ad95d216f8055ea2ded8eb8 | {{020107,0.50},{02,-1},{02011305,0.50},{02011310,0.50},{02011302,0.50},{02010601,0.50},{030210,100.00},{0302,-
 32d90db2cb8db4c356f9bd75fe88b0ea5 | {{020107,0.50},{02,-1},{030211,100.00},{0304,-1},{0302,-1},{020105,0.50},{02010503,0.50},{0201,-1},{03,-1},{030
(2 rows)
```

```
#筛选标签
SELECT
	*
FROM
	taginfo
WHERE
	tags[ARRAY_LOWER(tags,1),ARRAY_UPPER(tags,1)][1] @> '{020107}';

	
#将标签数据转换成多条结构化的记录
SELECT
	tdid,
	UNNEST_2D_1D(tags)
FROM
	taginfo
WHERE
	tdid = '32d90db2cb8db4c356f9bd75fe88b0ea5';
```

```
             tdid                 | unnest_2d_1d
----------------------------------+----------------
 3f614d2f092f6b745ea3be08afa6353a2 | {02011119,0.50}
 3f614d2f092f6b745ea3be08afa6353a2 | {02011110,0.50}
 3f614d2f092f6b745ea3be08afa6353a2 | {02011403,0.30}
 3f614d2f092f6b745ea3be08afa6353a2 | {080216,1.20}
 3f614d2f092f6b745ea3be08afa6353a2 | {02011101,0.30}
 3f614d2f092f6b745ea3be08afa6353a2 | {020108,0.50}
```

PostGIS

- 支持所有的空间数据类型，这些类型包括:点、线、多边形、多点、多线、多多边形和集合对象集等;
- 支持复杂的空间和地理位置计算;
- 数据库坐标变换;
- 球体长度运算;
- 三维的几何类型;
- 空间聚集函数;

聚集点数据，以下位置数据为了保护个人隐私都进行了聚集，误差+-76米

tdid	dateid	hourid	st_astext
39eec4560f0789ab3ee606dc18d3a0d9a	2017-07-01	6	MULTIPOINT(113.313675 22.973099,113.312302 22.973099)
30236edd8c9ec8997fff9443f5179a7eb	2017-07-01	5	MULTIPOINT(115.141525 35.873795,115.137405 35.872421,115.137405 35.873795,115.1
hd288059358af032358e281f391bb85a9	2017-07-01	10	MULTIPOINT(121.531448 31.214218)
a31dedbaff5e42ccb7726812b2e32408	2017-07-01	3	MULTIPOINT(123.842697 42.280197)
3fa5d560d9d8a74c93770334b78825e9e	2017-07-01	16	MULTIPOINT(120.443802 41.555099,120.438309 41.561966,120.443802 41.563339)
3fc48c75a4489664c6fc492df5f22759a	2017-07-01	11	MULTIPOINT(120.979385 31.416092)
3aea701571509ace732f28cbd2d380270	2017-07-01	19	MULTIPOINT(104.690781 29.247665)
3c9102fc1c4f5cff894e0de16212ef9c9	2017-07-01	13	MULTIPOINT(118.532181 25.044022)
h08edec82b54082da23813112576ad356	2017-07-01	1	MULTIPOINT(117.888107 34.184647)
3519d7d67bd6f878c664a2b00a5dbda5f	2017-07-01	16	MULTIPOINT(114.949265 25.841904)

(10 rows)

```
#中心点加半径筛选
SELECT
    *
FROM
    geoinfo
WHERE
    dateid between '2017-01-01' and '2017-01-31'
    AND ST_Intersects(geo,
        ST_GeomFromEWKB(
            ST_Buffer(
                ST_GeographyFromText('SRIC=4326;POINT(112.588419 29817109)',1000)
        );

#围栏筛选
SELECT
    *
FROM
    geoinfo
WHERE
    dateid between '2017-01-01' and '2017-01-31'
    AND ST_Intersects(geo,
        ST_GeometryFromTEXT(
            'POLYGON((104.517868 30.853043,104.742966 31.454544,111.20018 30.472641,104.517868 30.853043))'
            ,4326)
        );
```

Bitmap

- 在Greenplum内实现了Bitmap相关计算功能；
- 基于RoaringBitmap算法；
- 通过标准的SQL语句来执行计算；
- 10亿级别的位运算达到了毫秒级别；

```
fedw=# create table test_bitmap(id int, bitmap roaringbitmap);
NOTICE:  Table doesn't have 'DISTRIBUTED BY' clause -- Using column named 'id' as the Greenplum Database data dis
tribution key for this table.
HINT:  The 'DISTRIBUTED BY' clause determines the distribution of data. Make sure column(s) chosen are the optima
l data distribution key to minimize skew.
CREATE TABLE
fedw=# insert into test_bitmap select 1,rb_build(array[1,2]);
INSERT 0 1
fedw=# insert into test_bitmap select  2,rb_build_agg(e) from generate_series(1,100) e;
INSERT 0 1
fedw=# select rb_iterate(rb_and((select bitmap from test_bitmap where id =1),(select bitmap from test_bitmap wher
e id =2 ))) ;
 rb_iterate
------------
          1
          2
(2 rows)
fedw=# select rb_cardinality(rb_and_agg(bitmap)) from test_bitmap;
 rb_cardinality
----------------
              2
(1 row)
```

通过bitmap筛选标签

- 将每个设备映射到一个INT或者BIGINT数字上，通过sequence或者hash来实现映射；
- 设备的INT值对应BITMAP的位来标记该设备在此BITMAP中是否存在；
- 将同一个标签的所有的设备都集合到一个或者一组BITMAP内；

通过bitmap筛选标签

```
   Column    |     Type       | Modifiers
-------------+----------------+-----------
 tag         | text           |
 partitionid | integer        |
 bitmap      | roaringbitmap  |
 cnt         | integer        |
```

```
fedw=# select sum(rb_cardinality(bitmap)) as cnt from dw_tag.v_bitmap_tag where tag in ('01');
    cnt
------------
 1258038337
(1 row)

Time: 31.828 ms
fedw=# select sum(rb_cardinality(bitmap)) as cnt from dw_tag.v_bitmap_tag where tag in ('02');
    cnt
------------
 2143585396
(1 row)

Time: 35.180 ms
fedw=# select sum(cnt) from (select rb_cardinality(rb_and_agg(bitmap)) as cnt,partitionid from dw_tag.v_bitmap_tag where tag in ('01','02') group by partitionid) t ;
    sum
------------
 1099668763
(1 row)

Time: 94.168 ms
```

通过BITMAP+GEOHash实现毫秒级筛选ID

- 将位置信息转换为GEOHash，比如这里为了保护个人隐私，将geohash只保留7位，误差范围在+-76米。
- 将每一个GEOHASH的每个时间切片上生成ID对应的BITMAP

```
   Column    |         Type          | Modifiers
-------------+-----------------------+-----------
 geohash     | character varying(7)  |
 hourid      | smallint              |
 geo         | geometry              |
 dateid      | date                  |
 partitionid | integer               |
 bitmap      | roaringbitmap         |
```

```
fedw=# select rb_cardinality(rb_or_agg(bitmap)) from dw_geo.bitmap_geo_daily where dateid = '2017-01-01' and geoh
ash in ('wx4sp0y','wwkr9hr','wws8yt3','wm7bdb2','tz3447x','wkr5t7c','w7qt961','wwgq3wu','wwgq3y7','wk3n9tt','wm7c
7kb','wtw0zn1') ;
 rb_cardinality
----------------
           3086
(1 row)

Time: 70.662 ms
fedw=# select rb_cardinality(rb_or_agg(bitmap)) from dw_geo.bitmap_geo_daily where dateid = '2017-01-01' and geoh
ash in ('wx4sp0y','wwkr9hr','wws8yt3','wm7bdb2','tz3447x','wkr5t7c','w7qt961','wwgq3wu','wwgq3y7','wk3n9tt','wm7c
7kb','wtw0zn1') and hourid in (8,9,10,12,13,15);
 rb_cardinality
----------------
           1425
(1 row)

Time: 64.047 ms
fedw=# select rb_cardinality(rb_or_agg(bitmap)) from dw_geo.bitmap_geo_daily where dateid = '2017-01-01' and geoh
ash in ('wx4sp0y','wwkr9hr','wws8yt3','wm7bdb2','tz3447x','wkr5t7c','w7qt961','wwgq3wu') and hourid in (8,9,10,12
);
 rb_cardinality
----------------
            586
(1 row)

Time: 40.379 ms
```

通过BITMAP实现设备的时间序列

- 实现在一条记录内保存一个设备所有时间周期（粒度到分钟）的活跃情况记录

  - 每个设备一条记录，建立活跃BITMAP字段
  - 将活跃的具体活跃的时间转换为从1970年01月01日开始的分钟级偏移量组成一个BITMAP；

```
    Column    |     Type      | Modifiers
--------------+---------------+-----------
 offsetid     | bigint        |
 tdid         | text          |
 activebitmap | roaringbitmap |
```

```
fedw=# insert into device_active_bitmap(offsetid,tdid,activebitmap) select 2543323103,'3e509b2307bca32a3e3bba081d
8f2acca',rb_build_agg(e) from generate_series((EXTRACT(EPOCH FROM TIMESTAMP '2017-07-01 00:00:00')/60)::INT, (EXT
RACT(EPOCH FROM TIMESTAMP '2017-07-01 00:10:00')/60)::INT) e;
INSERT 0 1
Time: 45.763 ms
fedw=# update device_active_bitmap set activebitmap = rb_or(activebitmap,(select rb_build_agg(e) from generate_se
ries((EXTRACT(EPOCH FROM TIMESTAMP '2017-06-01 00:00:00')/60)::INT, (EXTRACT(EPOCH FROM TIMESTAMP '2017-06-01 00:
10:00')/60)::INT) e)) where offsetid= 2543323103;
UPDATE 1
Time: 40.925 ms
fedw=# select rb_cardinality(activebitmap) from device_active_bitmap  where offsetid= 2543323103;
 rb_cardinality
----------------
             22
(1 row)

Time: 33.897 ms
fedw=# select tdid from device_active_bitmap where rb_cardinality(rb_and(activebitmap, (select rb_build_agg(e) fr
om generate_series((EXTRACT(EPOCH FROM TIMESTAMP '2017-06-30 00:00:00')/60)::INT, (EXTRACT(EPOCH FROM TIMESTAMP '
2017-07-10 00:10:00')/60)::INT ) e )  )) >=1 limit 1;
               tdid
----------------------------------
 3e509b2307bca32a3e3bba081d8f2acca
(1 row)

Time: 20.377 ms
```

Bitmap功能开源

- https://github.com/zeromax007/gpdb-roaringbitmap
- 欢迎提交BUG和改进意见
- Email：zeromax@live.cn

Pivotal

IT大咖说 知识分享平台

谢谢!
Q&A

TalkingData