ClickHouse

在苏宁用户画像场景的实践

苏宁科技集团.大数据中心.杨兆辉

二〇一九年十月

1 关于我

- 苏宁科技集团大数据中心架构师
- 曾就职于中兴通讯10+years ，从事大规模分布式系统研发
- 10+years C++、Java、Go编程经验，熟悉大数据架构、解决方案
- ClickHouse Contributor
- Github: https://github.com/andyyzh

2 Contents

苏宁易购 suning.com

- 苏宁如何使用ClickHouse
- ClickHouse集成Bitmap
- 用户画像场景实践

3 选择ClickHouse的原因

1. 速度快
2. 特性发布快
3. 软件质量高
4. 物化视图
5. 高基数查询
6. 精确去重计数（count distinct）

4 精确去重计数性能测试

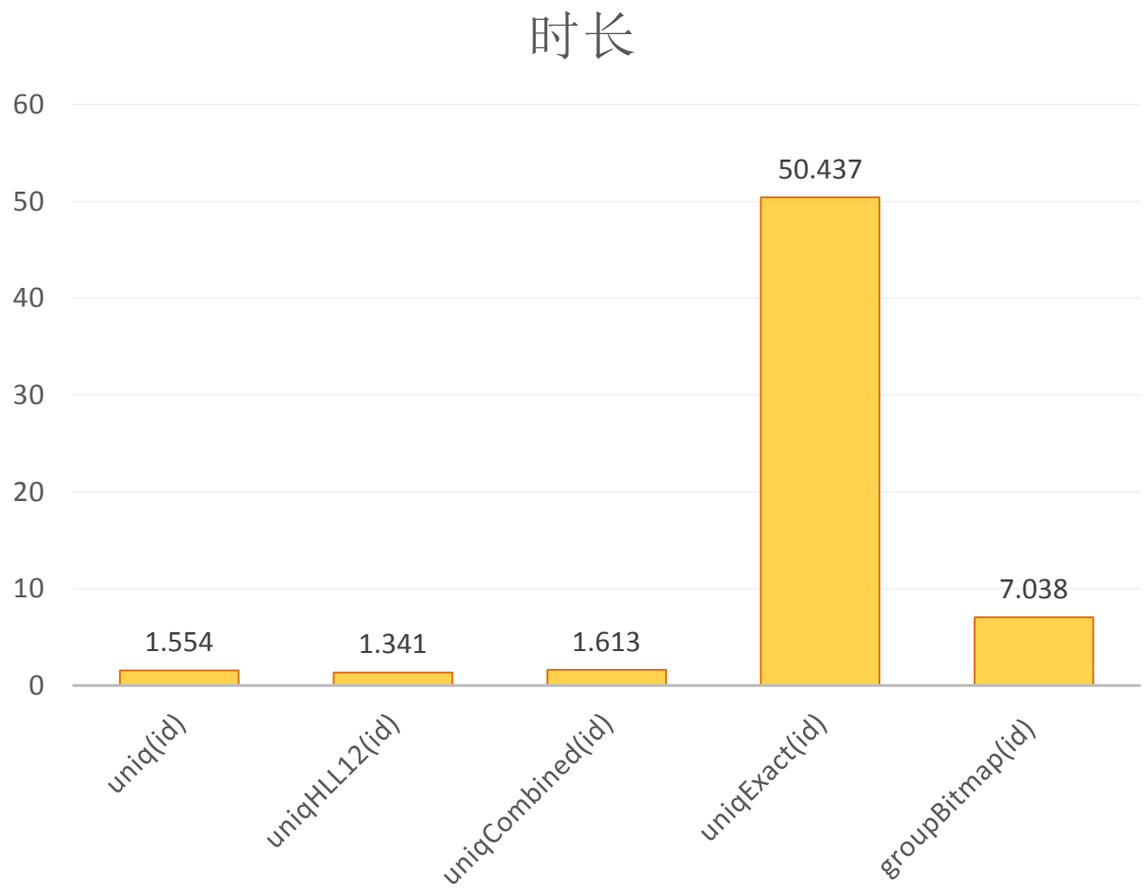
4亿多的数据集上，去重计算出6千万整形数值，

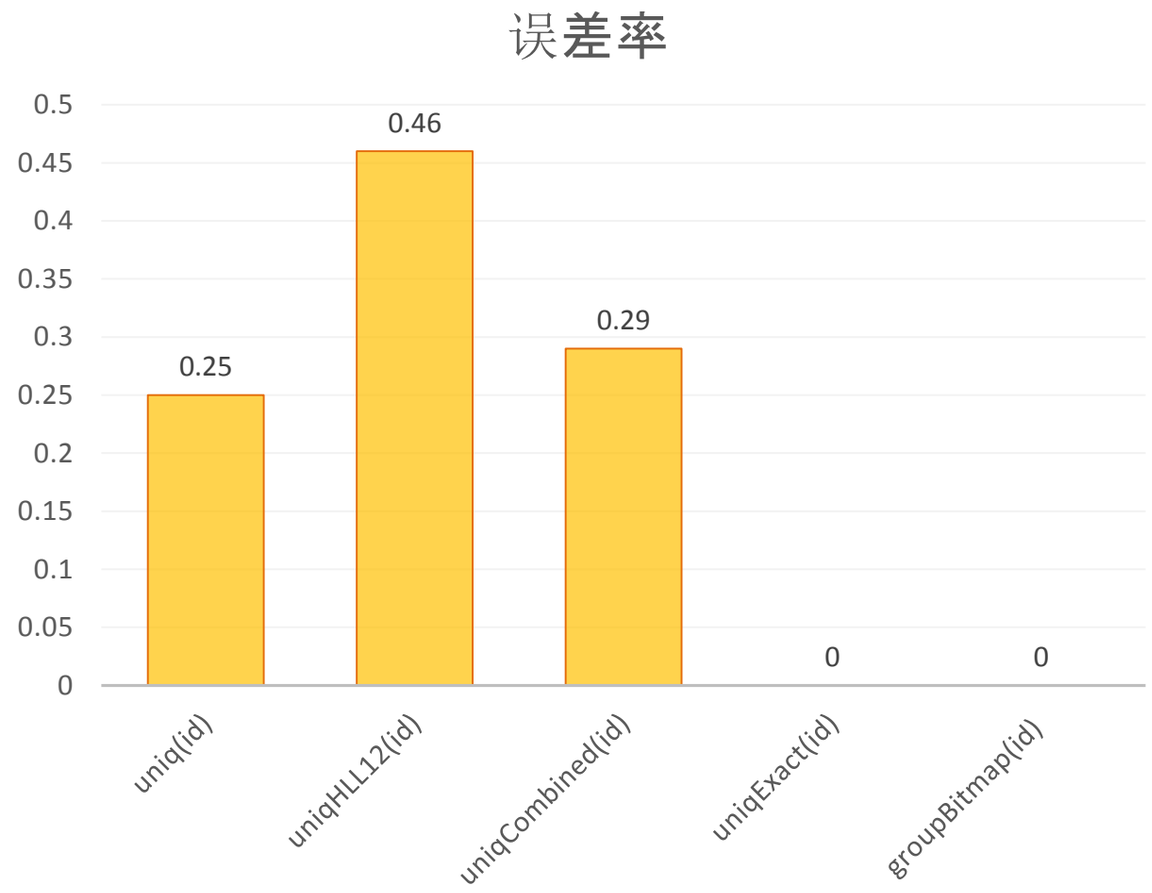
非精确去重函数：uniq、uniqHLL12、uniqCombined

精确去重函数：uniqExact、groupBitmap

函数	时长（秒）	去重后个数	误差个数	误差率
uniq(id)	1.554	63195280	155973	0.25%
uniqHLL12(id)	1.341	63331662	292355	0.46%
uniqCombined(id)	1.613	62859215	-180092	-0.29%
uniqExact(id)	50.437	63039307	0	0%
groupBitmap(id)	7.038	63039307	0	0%

5 精确去重计数性能测试





结论：

- 整形值精确去重场景，groupBitmap 比 uniqExact至少快 2x+
- groupBitmap仅支持整形值去重， uniqExact支持任意类型去重。
- 非精确去重场景，uniq在精准度上有优势。

6 精确去重计数性能测试

```
↖ Progress: 415.88 million rows, 3.33 GB (312.68 million rows/s., 2.50 GB/s.)
┌─ uniq(id)─┐
│  63195280 │
└─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ┘
↑  Progress: 415.88 million rows, 3.33 GB (267.63 million rows/s., 2.14 GB/s.)

1 rows in set. Elapsed: 1.554 sec. Processed 415.88 million rows, 3.33 GB (267.53 million rows/s., 2.14 GB/s.)
```

```
←  Progress: 400.04 million rows, 3.20 GB (299.17 million rows/s., 2.39 GB/s.)
┌─ uniqHLL12(id)─┐
│       63331662 │
└─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ┘
↖ Progress: 400.04 million rows, 3.20 GB (298.35 million rows/s., 2.39 GB/s.)
↑  Progress: 415.88 million rows, 3.33 GB (310.13 million rows/s., 2.48 GB/s.)

1 rows in set. Elapsed: 1.341 sec. Processed 415.88 million rows, 3.33 GB (310.06 million rows/s., 2.48 GB/s.)
```

```
↖ Progress: 409.82 million rows, 3.28 GB (269.44 million rows/s., 2.16 GB/s.)
┌─ uniqCombined(id)─┐
│          62859215 │
└─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ┘
↑  Progress: 409.82 million rows, 3.28 GB (254.16 million rows/s., 2.03 GB/s.)
↗ Progress: 415.88 million rows, 3.33 GB (257.91 million rows/s., 2.06 GB/s.)

1 rows in set. Elapsed: 1.613 sec. Processed 415.88 million rows, 3.33 GB (257.83 million rows/s., 2.06 GB/s.)
```

```
↑  Progress: 415.88 million rows, 3.33 GB (152.23 million rows/s., 1.22 GB/s.)
┌─ uniqExact(id)─┐
│       63039307 │
└─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ┘
↗ Progress: 415.88 million rows, 3.33 GB (8.25 million rows/s., 65.96 MB/s.)

1 rows in set. Elapsed: 50.437 sec. Processed 415.88 million rows, 3.33 GB (8.25 million rows/s., 65.96 MB/s.)
```

```
→  Progress: 415.88 million rows, 3.33 GB (103.18 million rows/s., 825.44 MB/s.)
┌─ groupBitmap(id)─┐
│          63039307 │
└─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ┘
↘ Progress: 415.88 million rows, 3.33 GB (59.09 million rows/s., 472.76 MB/s.)

1 rows in set. Elapsed: 7.038 sec. Processed 415.88 million rows, 3.33 GB (59.09 million rows/s., 472.72 MB/s.)
```

7 ClickHouse在苏宁使用场景

- OLAP平台存储引擎

-- 存储时序数据、cube加速数据，应用于高基数查询、精确去重场景。

- 运维监控

-- 实时聚合分析监控数据，主要使用物化视图技术。

- 用户画像场景

-- 标签数据的存储、用户画像查询引擎。

Contents

9 Bitmap位存储和位计算

每个bit位表示一个数字id，对于40亿个的用户id，只需要40亿bit位，
约477m大小 = （4 * 10^9 / 8 / 1024 / 1024）

通过单个bitmap可以完成精确去重操作，通过多个bitmap的and、or、xor、andnot等位操作完成留存分析、漏斗分析、用户画像分析等场景的计算。

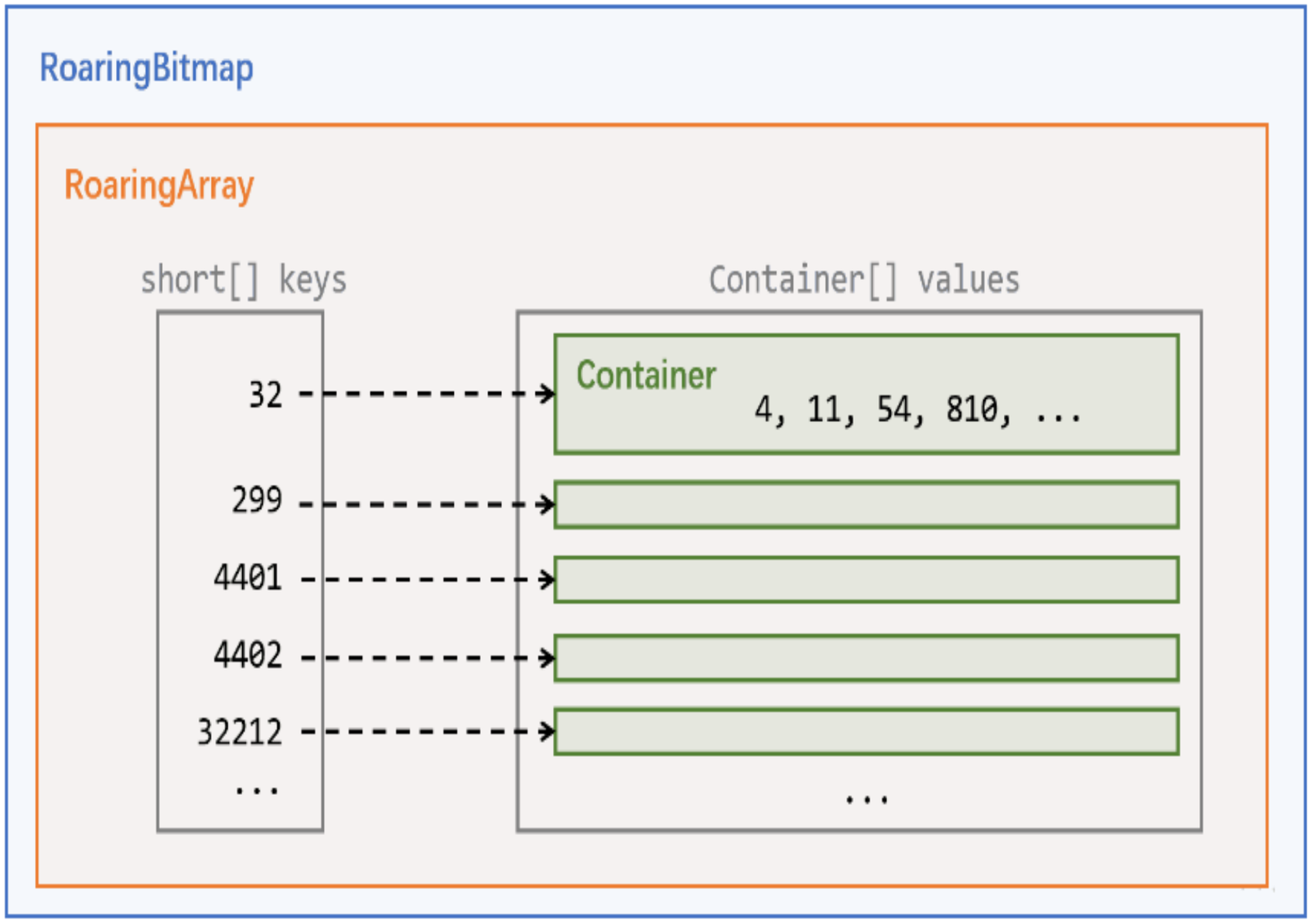
但是如果使用上述的数据结构存储单独一个较大数值的数字id，会造成空间上的浪费，例如仅存储40亿一个数值也需要477m的空间。也就是说稀疏的Bitmap和稠密的占用空间相同。通常会使用一种bitmap压缩算法进行优化。

RoaringBitmap是一种已被业界广泛使用的高效的bitmap压缩算法，使用者包括Spark、Hive、ElasticSearch、Kylin、Druid、InfluxDB等，
详见：http://roaringbitmap.org/

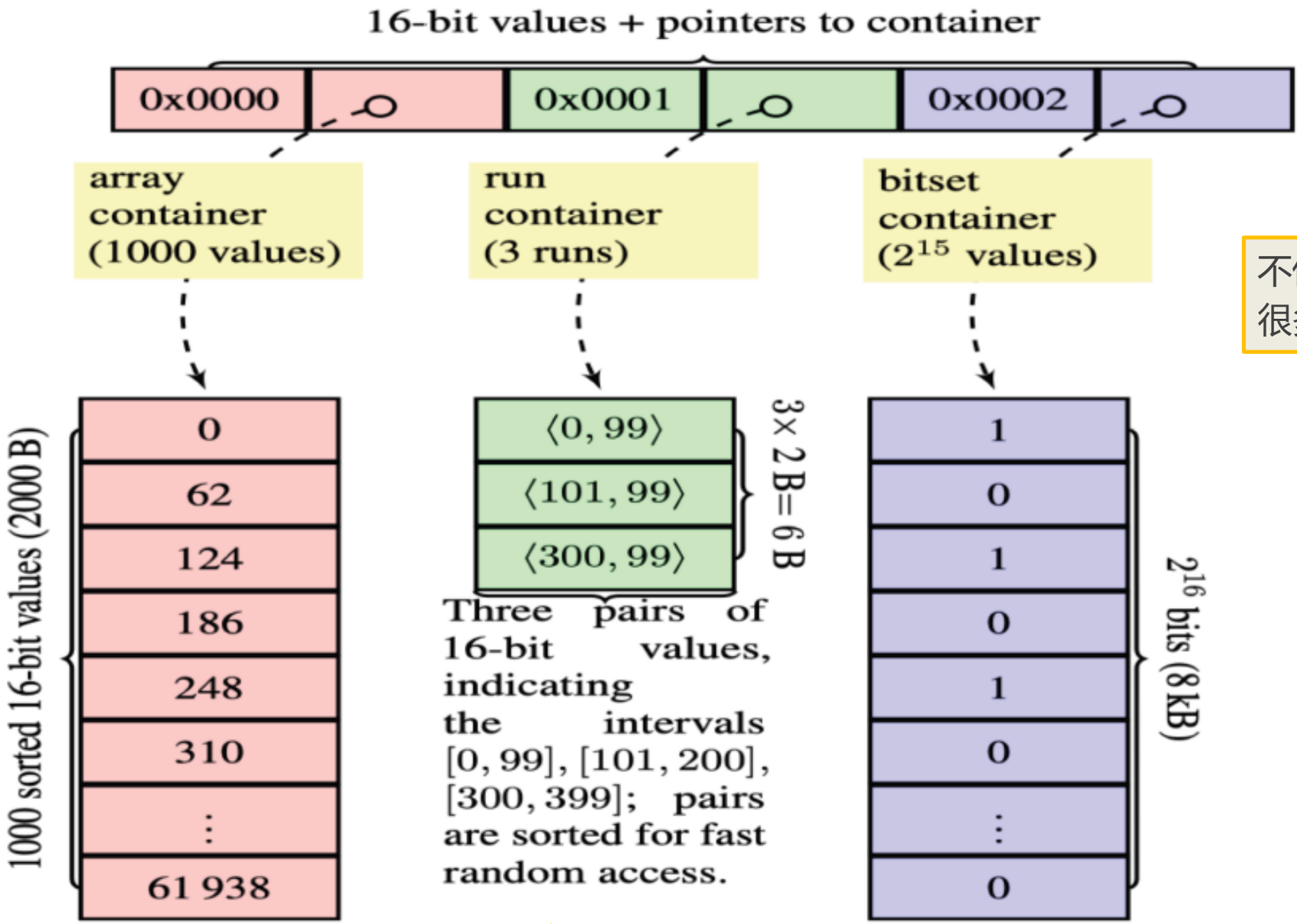
RoaringBitmap原理介绍

主要原理：将32bit的Integer划分为高16位和低16位(两个short int)，两者之间是Key-Value的关系。高16位存到short[] keys，通过高16位（Key）找到所对应Container，然后把剩余的低16位（Value）放入该Container中，RoaringBitmap有三类Container：

- Array Container
- Run Container
- Bitmap Container



RoaringBitmap原理介绍



不仅数据结构设计精巧，而且还有很多高效的Bitmap计算函数。

稀疏数据，动态分配
最大存储：4096元素
最大空间：8KB

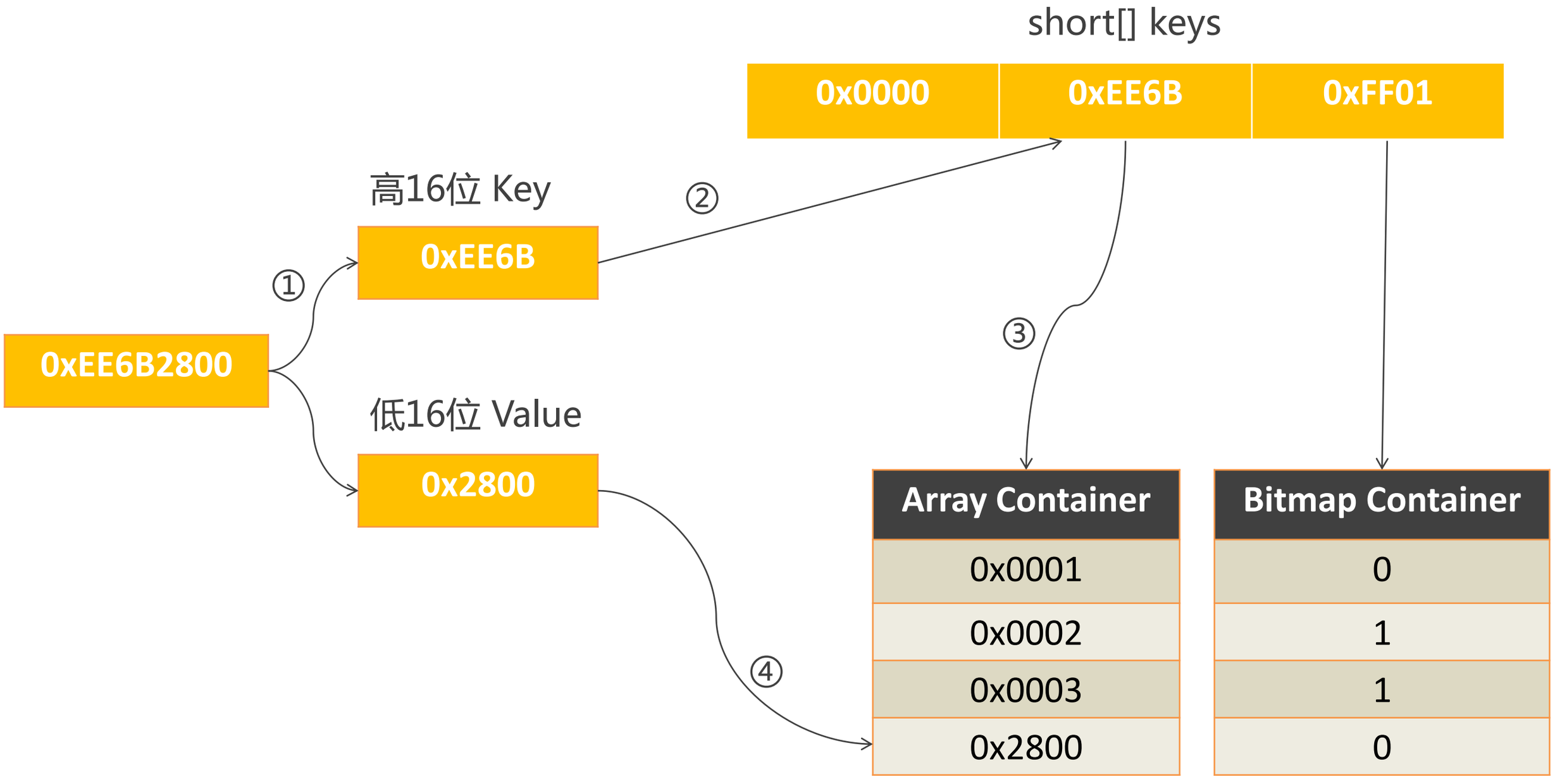
连续数据，动态分配
最大存储：65536元素
最大空间：128KB

稠密数据，固定大小
最大存储：65536元素
最大空间：8KB

RoaringBitmap原理介绍

举个栗子：

40亿（0xEE6B2800）这个值如何存入RoaringBitmap，以存入Array Container来说明：



13 ClickHouse集成RoaringBitmap

Bitmap字段类型，该类型扩展自AggregateFunction类型，字段类型定义：

```
AggregateFunction( groupBitmap, UInt(8|16|32|64))
```

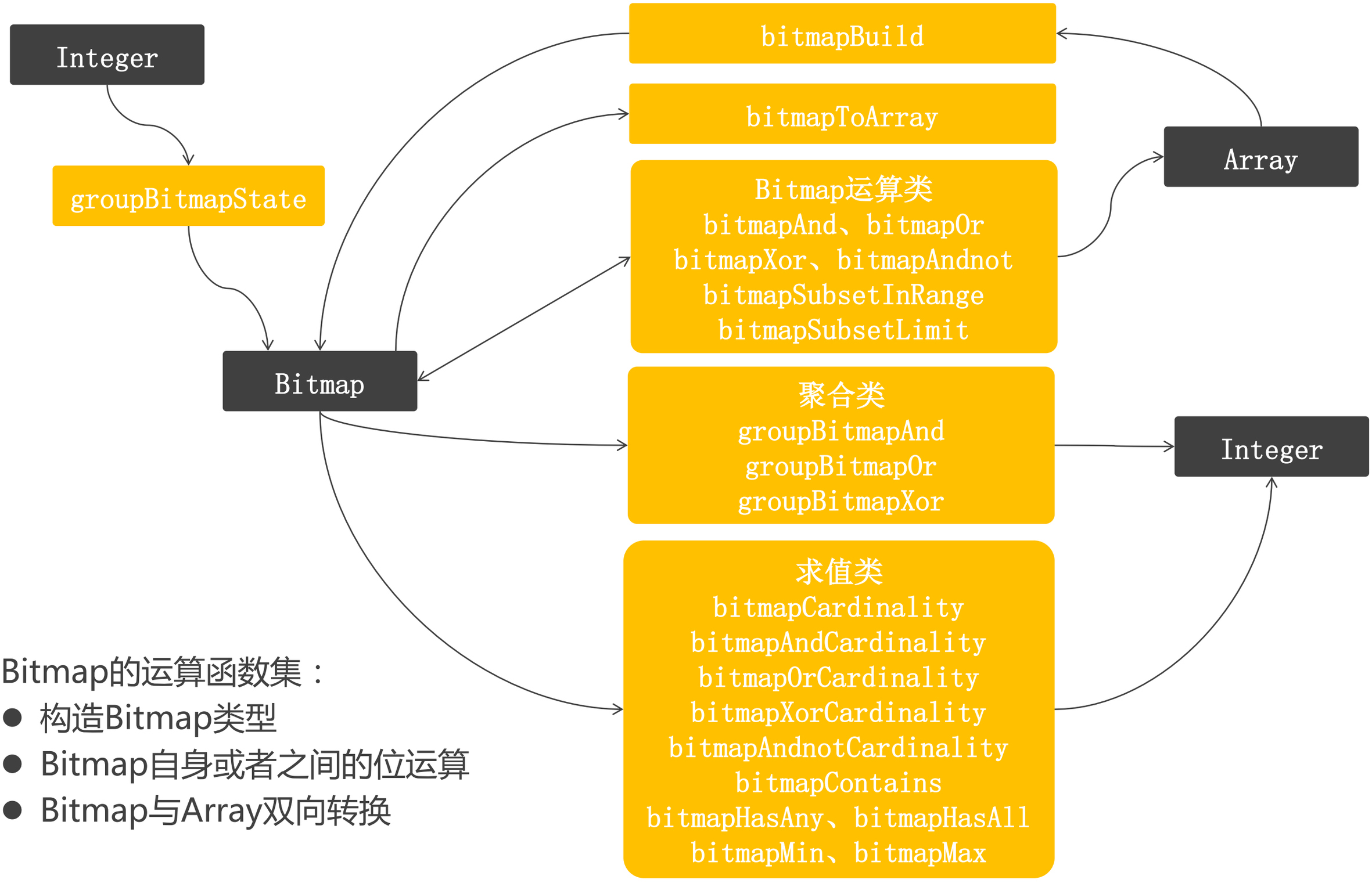
表示groupBitmap聚合函数的中间状态。
可以通过groupBitmapState创建。

```
SELECT labelname, labelvalue,
groupBitmapState(id) AS uv
FROM label_table
GROUP BY labelname, labelvalue
```

注：ClickHouse聚合函数有一些函数后缀可以使用：
-State：获取聚合的中间计算结果
-Merge：将中间计算结果进行合并计算，返回最终结果
-MergeState：将中间计算结果进行合并计算，返回合并后的中间结果

参考：
https://clickhouse.yandex/docs/en/data_types/nested_data_structures/aggregatefunction/
https://clickhouse.yandex/docs/en/query_language/agg_functions/reference/#groupbitmap

ClickHouse集成RoaringBitmap



Bitmap的运算函数集：

- 构造Bitmap类型
- Bitmap自身或者之间的位运算
- Bitmap与Array双向转换

Bitmap应用示例

一张简单的订单明细表 **detail_order**，如何计算用户的日留存？

order_id	order_date	user_id	product_id
1	2019-10-01	1	p1
2	2019-10-01	1	p2
3	2019-10-01	2	p1
4	2019-10-01	3	p1
5	2019-10-02	3	p2
6	2019-10-02	4	p1
7	2019-10-02	5	p1
8	2019-10-02	5	p2

大表join，count distinct都比较慢，而且容易OOM!

标签 SQL

```sql
SELECT count( distinct first_day.user_id ) AS retain_user
FROM
(
    SELECT distinct user_id FROM detail_order WHERE order_date = '2019-10-01'
) AS first_day
INNER JOIN
(
    SELECT distinct user_id FROM detail_order WHERE order_date = '2019-10-02'
) AS second_day ON first_day.user_id = second_day.user_id;
```

Bitmap应用示例

detail_order 聚合为天维度表

order_date	uv_bitmap
2019-10-01	{1,2,3}
2019-10-02	{3,4,5}

留存用户的SQL

```
WITH
    (
        SELECT uv_bitmap FROM uv_order WHERE order_date = '2019-10-01'
    ) AS day1,
    (
        SELECT uv_bitmap FROM uv_order WHERE order_date = '2019-10-02'
    ) AS day2
SELECT bitmapAndCardinality(day1, day2) AS retain_user;
```

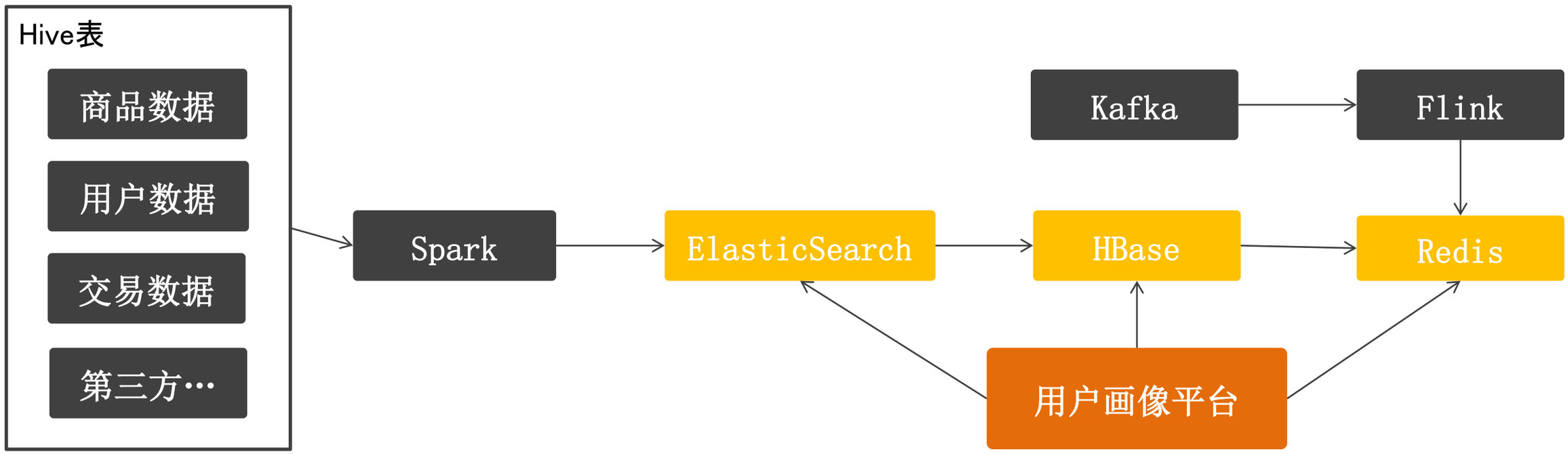
千万级用户，秒级出结果！

Bitmap函数

- 留存用户：day1 AND day2 = [3]
- 全部用户：day1 OR day2 = [1,2,3,4,5]
- 新用户： day2 ANDNOT day1 = [4,5]
- 流失用户：day1 ANDNOT day2 = [1,2]

Contents

用户画像原有的流程及痛点



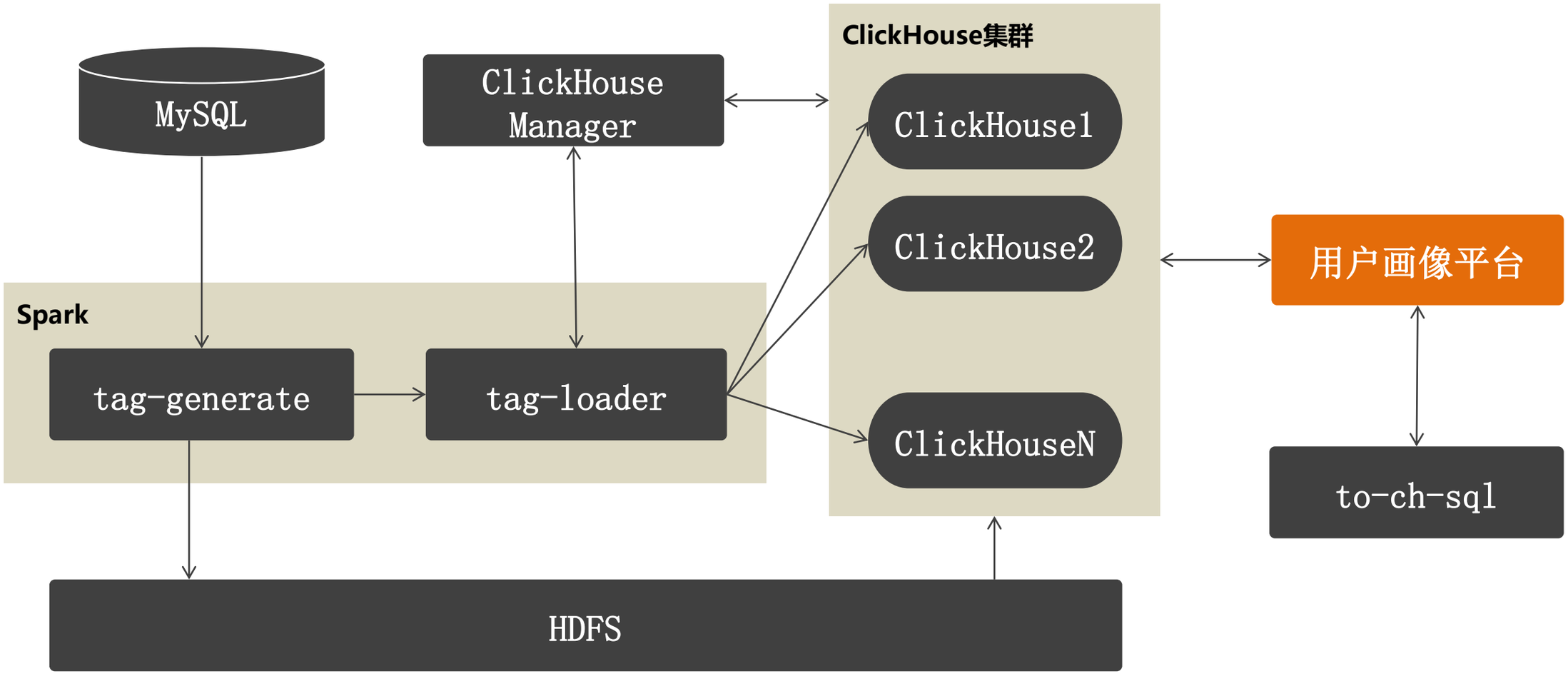
现有的流程：

- ES中定义标签的大宽表
- 通过Spark关联各种业务数据，插入到ES大宽表。
- 高频查询的画像数据通过后台任务保存到加速层：Hbase 或者 Redis
- 实时标签通过Flink计算，然后写入Redis
- 用户画像平台可以从ES、Hbase、Redis查询数据

痛点：

- 标签导入到ES的时间过长，需要等待各种业务数据准备就绪，才能进行关联查询。
- 新增或者修改标签，不能实时进行，涉及到ES文档结构的变化。
- ES对资源消耗比较大，属于豪华型配置。
- ES的DSL语法对用户不太友好，用户学习成本高。

ClickHouse替换ES存储标签数据



- ClickHouse Manager负责ClickHouse集群管理、元数据管理以及节点负载协调
- tag-generate负责标签数据构建，保存到HDFS（MySQL中存储标签配置信息）
- tag-loader向ClickHouse发送从HDFS导入标签数据的sql
- to-ch-sql模块，将用户画像查询条件转换为ClickHouse sql语句
- 用户画像平台通过Proxy从ClickHouse集群查询标签数据

标签数据表定义

String

```
CREATE TABLE ch_label_string
(
 labelname String,
 labelvalue String,
 uv AggregateFunction( groupBitmap, UInt64 )
)
ENGINE = AggregatingMergeTree()
PARTITION BY labelname
ORDER BY (labelname, labelvalue)
SETTINGS index_granularity = 8192;
```

Integer

```
CREATE TABLE ch_label_int
(
 labelname String,
 labelvalue UInt64,
 uv AggregateFunction( groupBitmap, UInt64 )
)
ENGINE = AggregatingMergeTree()
PARTITION BY labelname
ORDER BY (labelname, labelvalue)
SETTINGS index_granularity = 8192;
```

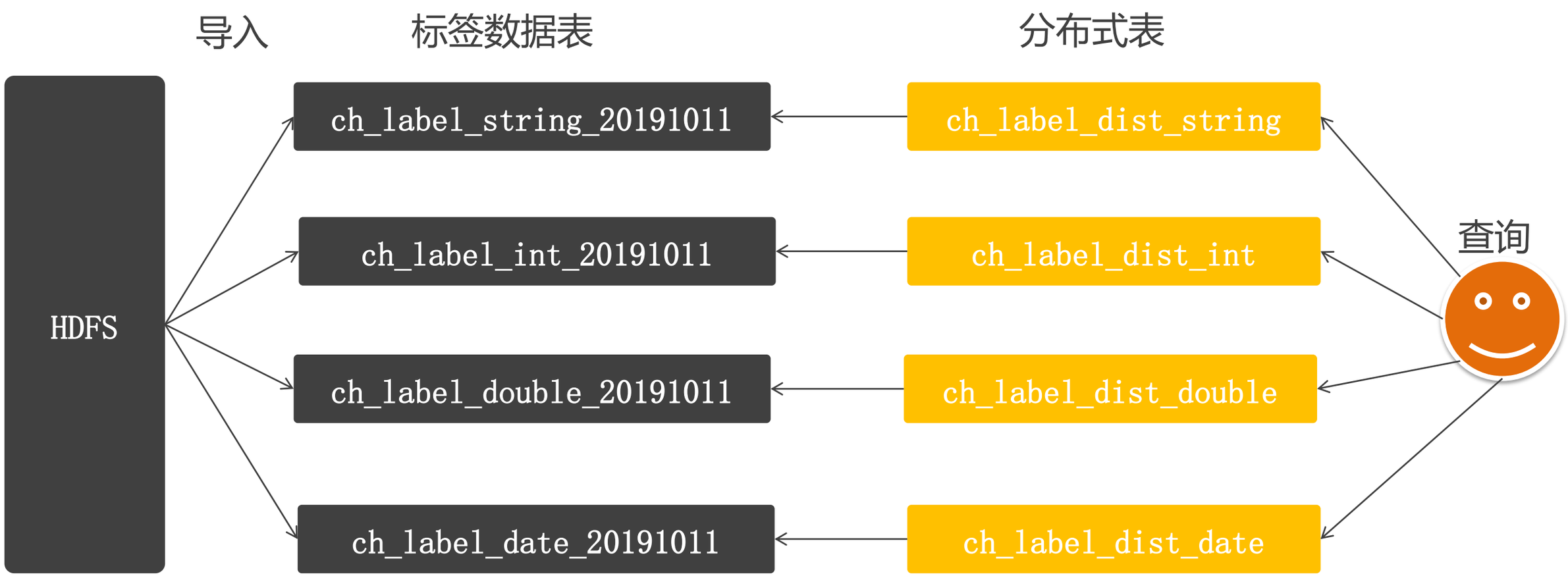
Double

```
CREATE TABLE ch_label_double
(
 labelname String,
 labelvalue double,
 uv AggregateFunction( groupBitmap, UInt64 )
)
ENGINE = AggregatingMergeTree()
PARTITION BY labelname
ORDER BY (labelname, labelvalue)
SETTINGS index_granularity = 8192;
```

Date

```
CREATE TABLE ch_label_date
(
 labelname String,
 labelvalue Date,
 uv AggregateFunction( groupBitmap, UInt64 )
)
ENGINE = AggregatingMergeTree()
PARTITION BY labelname
ORDER BY (labelname, labelvalue)
SETTINGS index_granularity = 8192;
```

数据模型定义



- HDFS上采用snappy.parquet格式存储数据。
- 采用AB表切换方式，避免查询和写入的冲突，标签数据表以日期结尾命名。
- 通过重建分布式表进行AB表切换，指向不同日期的标签数据表。
- 通过增加标签数据表的副本数，提升并发性能。

用户画像系统常见应用场景

举个栗子：

“双11” 就要到了，需要发放10万张家电类优惠券进行促销：

用户画像场景1—预估人数

场景描述

场景：限量发放10万张家电类优惠券，先预估出符合条件的用户数。
操作：用户指定标签及标签间的逻辑关系，统计出符合标签逻辑的人数。

输入条件

标签表达式，包含标签、算术运算符、逻辑运算符、括号。

返回结果

整形值，表示符合标签表达式的用户人数
例如：

user_number
100000

24 用户画像场景1—预估人数—示例

画像条件

```
(gender = 'M' AND member_level < 5) OR age IN (20,25,30)
```

查询SQL

```
SELECT bitmapOrCardinality(bitmapAnd(users0, users1), users2) AS user_number
FROM
(
    SELECT 1 AS join_id, groupBitmapMergeState(uv) AS users0
    FROM ch_label_dist_string
    WHERE (labelname = 'gender') AND (labelvalue = 'M')
)
INNER JOIN
(
    SELECT  1 AS join_id, groupBitmapMergeState(uv) AS users1
    FROM ch_label_dist_int
    WHERE (labelname = 'member_level') AND (labelvalue < 5)
) ON join_id = join_id
INNER JOIN
(
    SELECT   1 AS join_id, groupBitmapMergeState(uv) AS users2
    FROM ch_label_dist_int
    WHERE (labelname = 'age') AND (labelvalue IN (20, 25, 30))
) ON join_id = join_id
```

用户画像场景2—人群圈选画像

场景描述

场景：对选出符合发优惠券条件的用户进行画像分析，人群特征分析。
操作：用户指定标签及标签间的逻辑关系，查询出符合标签逻辑的用户ID数据集，然后对数据集进行用户画像分析。一条SQL完成人群圈选、用户画像两个动作。

输入条件

标签逻辑表达式，包含标签、算术运算符、逻辑运算符、括号。

返回结果

查询出符合标签表达式的用户ID Bitmap对象，
然后将Bitmap对象与画像表进行与（AND）操作，返回用户画像信息。
例如：

label_name	label_value	user_number
gender	M	12
gender	F	15
age	25	11
age	30	16

用户画像场景2—人群圈选画像—示例

画像条件

```
(gender = 'M' AND member_level < 5) OR age IN (20,25,30)
```

查询SQL

```
SELECT labelname, labelvalue,
    bitmapAndCardinality(users, checkout_users) AS user_number
FROM(
    SELECT 1 AS join_id, labelname, labelvalue, groupBitmapMergeState(uv) AS users
    FROM ch_profile_dist GROUP BY labelname, labelvalue
)
INNER JOIN(
    SELECT join_id, bitmapOr(bitmapAnd(users0, users1), users2) AS checkout_users
    FROM(
        SELECT 1 AS join_id, groupBitmapMergeState(uv) AS users0
        FROM ch_label_single_dist_string
        WHERE (labelname = 'gender') AND (labelvalue = 'M')
    )
    INNER JOIN(
        SELECT 1 AS join_id, groupBitmapMergeState(uv) AS users1
        FROM ch_label_single_dist_int
        WHERE (labelname = 'member_level') AND (labelvalue < 5)
    ) ON join_id = join_id
    INNER JOIN(
        SELECT 1 AS join_id, groupBitmapMergeState(uv) AS users2
        FROM ch_label_single_dist_int
        WHERE (labelname = 'age') AND (labelvalue IN (20, 25, 30))
    ) ON join_id = join_id
) ON join_id = join_id
```

用户画像场景3—用户ID清单

场景描述

场景：在筛选出符合条件的用户数后，导出用户ID明细，这样好给他们发优惠券。
操作：用户指定标签及标签间的逻辑关系，查询出符合标签逻辑的用户ID数据集。

输入条件

标签逻辑表达式，包含标签、算术运算符、逻辑运算符、括号。

返回结果

用户ID字段，表示符合标签表达式的用户ID集合。
例如：

user_list
8
10
11
12

用户画像场景3—用户ID清单—示例

画像条件

```
(gender = 'M' AND member_level < 5) OR age IN (20,25,30)
```

查询SQL

```
SELECT arrayJoin(bitmapToArray(bitmapOr(bitmapAnd(users0, users1), users2))) AS user_list
FROM
(
    SELECT 1 AS join_id, groupBitmapMergeState(uv) AS users0
    FROM ch_label_dist_string
    WHERE (labelname = 'gender') AND (labelvalue = 'M')
)
INNER JOIN
(
    SELECT 1 AS join_id, groupBitmapMergeState(uv) AS users1
    FROM ch_label_dist_int
    WHERE (labelname = 'member_level') AND (labelvalue < 5)
) ON join_id = join_id
INNER JOIN
(
    SELECT 1 AS join_id, groupBitmapMergeState(uv) AS users2
    FROM ch_label_dist_int
    WHERE (labelname = 'age') AND (labelvalue IN (20, 25, 30))
) ON join_id = join_id
```

用户画像新架构的优势

速度快　实时　友好　省钱

- 每个标签的数据可以并行构建，加快标签数据生产速度。
- HDFS文件并发导入ClickHouse，加快标签数据的就绪速度。
- 查询请求平均响应时长在2秒以下，复杂查询在10秒内。
- 支持标签数据实时更新，增加标签、删除标签、修改标签。
- 标签表达式和查询SQL对用户来说比较友好。
- 相对于ElasticSearch的配置，可以节约一半硬件资源。

社区贡献

- Bitmap功能

https://github.com/ClickHouse/ClickHouse/pull/4207

Add some bitmap functions by yuzhichang and svladykin

- 日期的周计算函数，支持十种周计算模式，兼容MySQL周计算函数

https://github.com/ClickHouse/ClickHouse/pull/5212

- Bug Fix

THANKS !

扫一扫上面的二维码图案，加我微信