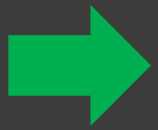
基于Metis-Druid平台的移动数据分析服务

团队：手机卫士服务端
讲师：蒋冬临

提纲



1. Metis-druid 平台介绍
2. 移动端日志解析与存储
3. 多维实时查询与接入

Metis-druid 介绍 主要模块

每日处理的日志种类150多种，，处理比较繁琐，日志自动化接入处理

日志处理

日志存储

面对T级别大小的客户端复杂日志的存储

数据分析的方式多样化，支持多维图表展示，支持A|B 数据对比，支持top排名等功能分析

数据统计

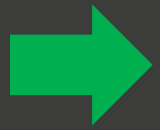
数据分析

根据用户行为，挖掘用户兴趣，对用户进行归类，实现用户画像等内容

可以根据用户的阈值配置，对监控数据预警，同时可以对后续的一些行为进行预测

预警预测

提纲



1. Metis-druid 平台介绍
2. 移动端日志解析与存储
3. 多维实时查询与接入

移动端日志解析与存储

移动端日志示例

……

移动端日志解析与存储

花椒相机测试数据

```
{
  "用户属性1": "aaa",
  "用户属性2": "aaa",
  "用户属性3": "aaa",
  "用户属性4": "aaa",
  "用户属性5": "aaa",
  "用户属性6": "aaa",
  "用户行为": [
    {
      "用户行为1": "abc",
      "用户时间": "dd",
      "用户行偏好": "abc"
    },
    {
      "用户行为1": "zxc",
      "用户时间": "qq",
      "用户行偏好": "fdg"
    },
    {
      "用户行为1": "gjh",
      "用户时间": "ew",
      "用户行偏好": "yu"
    }
  ]
}
```

扁平化格式的日志解析

按照字段将原有的复杂日志格式解析为**扁平化、格式化**数据。

公共部分：

```
"用户属性1": "aaa",
"用户属性2": "aaa",
"用户属性3": "aaa",
"用户属性4": "aaa",
"用户属性5": "aaa",
"用户属性6": "aaa",
```

展开部分：

```
"用户行为1": "abc",
"用户时间": "dd",
"用户行偏好": "abc"
```

移动端日志解析与存储

"用户属性1": "aaa",
"用户属性2": "aaa",
"用户属性3": "aaa",
"用户属性4": "aaa",
"用户属性5": "aaa",
"用户属性6": "aaa",
"用户行为1": "abc",
"用户时间": "dd",
"用户行偏好": "abc"

1

"用户属性1": "aaa",
"用户属性2": "aaa",
"用户属性3": "aaa",
"用户属性4": "aaa",
"用户属性5": "aaa",
"用户属性6": "aaa",
"用户行为1": "zxc",
"用户时间": "qq",
"用户行偏好": "fdg"

2

"用户属性1": "aaa",
"用户属性2": "aaa",
"用户属性3": "aaa",
"用户属性4": "aaa",
"用户属性5": "aaa",
"用户属性6": "aaa",
"用户行为1": "gjh",
"用户时间": "ew",
"用户行偏好": "yu"

3

一条日志 展开为 多条日志

缺点：

1、日志量膨胀严重，公共部分信息冗余

2、解析后原本格式丢失

Parquet 优势

1

*压缩编码可以降低磁盘存储空间。由于同一列的数据类型是一样的，可以使用更高效的压缩编码进一步节约存储空间。

2

*可以跳过不符合条件的数据，只读取需要的数据，降低IO数据量；
*只读取需要的列，能够获取更好的扫描性能。

3

*支持多层嵌套的数据结构。既可以保持数据原有的层次，又可以满足多样化的查询需求。

数据模型schema

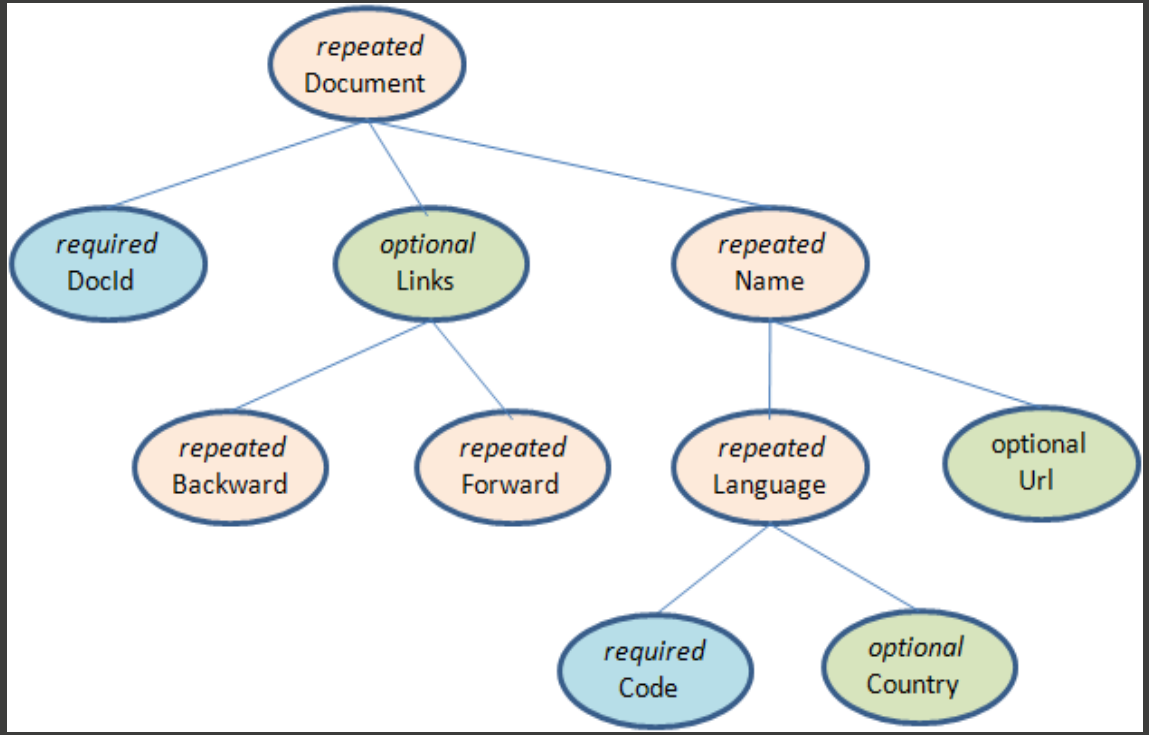
```
message Document {
  required int64 DocId;              [1,1]
  optional group Links {
    repeated int64 Backward;         [0,*]
    repeated int64 Forward;
  }
  repeated group Name {
    repeated group Language {
      required string Code;
      optional string Country;       [0,1]
    }
    optional string Url;
  }
}
```

```
DocId: 10
Links
  Forward: 20
  Forward: 40
  Forward: 60
Name
  Language
    Code: 'en-us'
    Country: 'us'
  Language
    Code: 'en'
  Url: 'http://A'
Name
  Url: 'http://B'
Name
  Language
    Code: 'en-gb'
    Country: 'gb'
```

树状结构

1. 叶子节点存放基本类型数据

2. 没有Map、Array复杂数据结构



数据模型schema


```
{  "name": "xxxx",
   "age": 30,
   "company": {
      "title": “yyyy",
      "departName": “zzzzz"
   },
   "jobs": [
      {
         "projectName": “xxx develop",
         "projectType": "inner",
         "periods": {
            "startTime": "2016-10-01",
            "endTime": "2017-10-01"
         }
      },
      {
         "projectName": "sms interception",
         "projectType": "to general customers",
         "periods": {
            "startTime": "2015-10-01",
            "endTime": "2022-10-01"
         }}]}
```

```
message Document {
  required string name;
  optional int32 age;
  required group company {
    required string title;
    required string departName;
  }
  repeated group jobs {
    optional group periods {
      required string startTime;
      required string endTime;
    }
    required string projectName;
    required string projectType;
  }
}
```

数据模型schema

```
message Document {
  required string name;
  optional int32 age;
  required group company {
    required string title;
    required string departName;
  }
  repeated group jobs {
    optional group periods {
      required string startTime;
      required string endTime;
    }
    required string projectName;
    required string projectType;
  }
}
```

Striping/Assembly算法

在该算法中列的每一个值都包含三部分

1. Value

2. repetition level
   - 在写入的时候该值等于它和前面的值在哪一层节点是不共享的
   - 在读取的时候推导出哪一层上需要创建一个新的节点

3. definition level:对于NULL来说，路径p上有多少字段可以是不存在(例如在文档定义中是optional或repeated，而不是required)，然而实际却存在的

Repetition and definition levels

```
DocId: 10
Links
  Forward: 20
  Forward: 40
  Forward: 60
Name
  Language
    Code: 'en-us'
    Country: 'us'
  Language
    Code: 'en'
  Url: 'http://A'
Name
  Url: 'http://B'
Name
  Language
    Code: 'en-gb'
    Country: 'gb'
```

: common prefix

r: repetition level
d: definition level

$doc_1.Name_1.Language_1.Code$: **'en-us'**
$doc_1.Name_1.Language_2.Code$: **'en'**
$doc_1.Name_2$
$doc_1.Name_3.Language_1.Code$: **'en-gb'**

Name.Language.Code

value	r	d
en-us	**0**	2
en	**2**	2
NULL	**1**	1
en-gb	**1**	2

最终数据格式

DocId

value	r	d
10	0	0
20	0	0

Name.Url

value	r	d
http://A	0	2
http://B	1	2
NULL	1	1
http://C	0	2

Links.Forward

value	r	d
20	0	2
40	1	2
60	1	2
80	0	2

Links.Backward

value	r	d
NULL	0	1
10	0	2
30	1	2

Name.Language.Code

value	r	d
en-us	0	2
en	2	2
NULL	1	1
en-gb	1	2
NULL	0	1

Name.Language.Country

value	r	d
us	0	3
NULL	2	2
NULL	1	1
gb	1	3
NULL	0	1

列式存储格式

应用数据对比

数据量大小	业务1数据	业务2数据
原始日志数据	42.62G	688.16G
原始数据Gz压缩存储	5.17G	99.49G
原始数据 Parquet+gz	3.02G	16.56G
原始日志压缩比例	14.11倍	42.21倍
相比Gzip 压缩比例	1.7倍	6.00倍

应用数据对比

打点数据大小	数据量	统计PV	某一字段去重统计	投影某两列字段
扁平化行式	548.46M	13.917 s	31.016 s	50.177 s
parquet	**26.39M**	**7.053 s**	**10.498 s**	**8.448 s**
比例	**20.78倍**	**1.97倍**	**3.10倍**	**5.9倍**

提纲

1 Metis BI 介绍

2 移动端日志解析与存储

3 多维实时查询与接入

什么是OLAP？

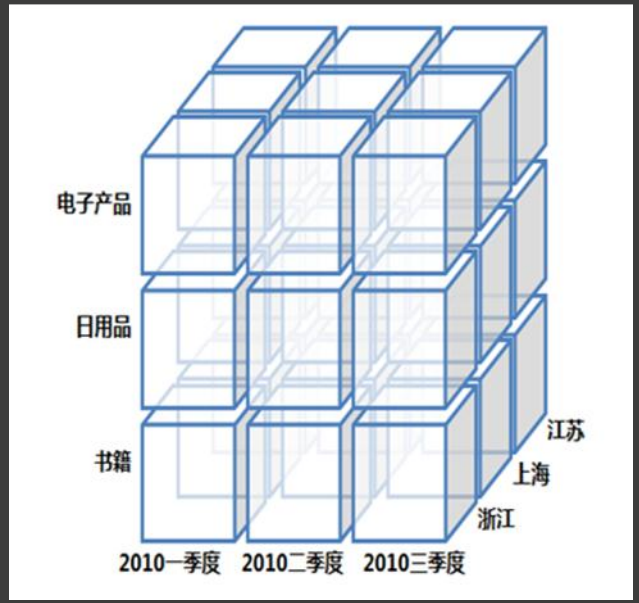
OLAP(Online Analytical Processing)

- 联机分析处理
- 快速、灵活地进行大数据量的**多维度**
  实时查询。

druid

kylin

数据立方体



Druid主要解决问题场景

- 百亿千亿级别日志，独立访客(UV)的统计
- 多维度多样化的留存率分析
- top-k elements，统计top APP访问排行
- Range Query，统计一段时间范围的网站访问量（PV，UV）
- Group by dimensions 分析

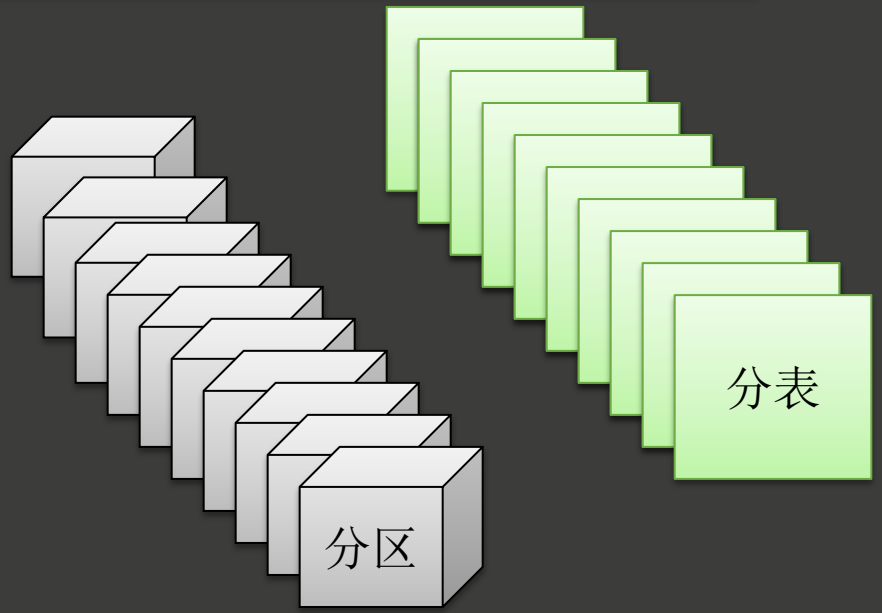
Druid主要解决问题场景

- 多维查询下，如果多达20维以上。。

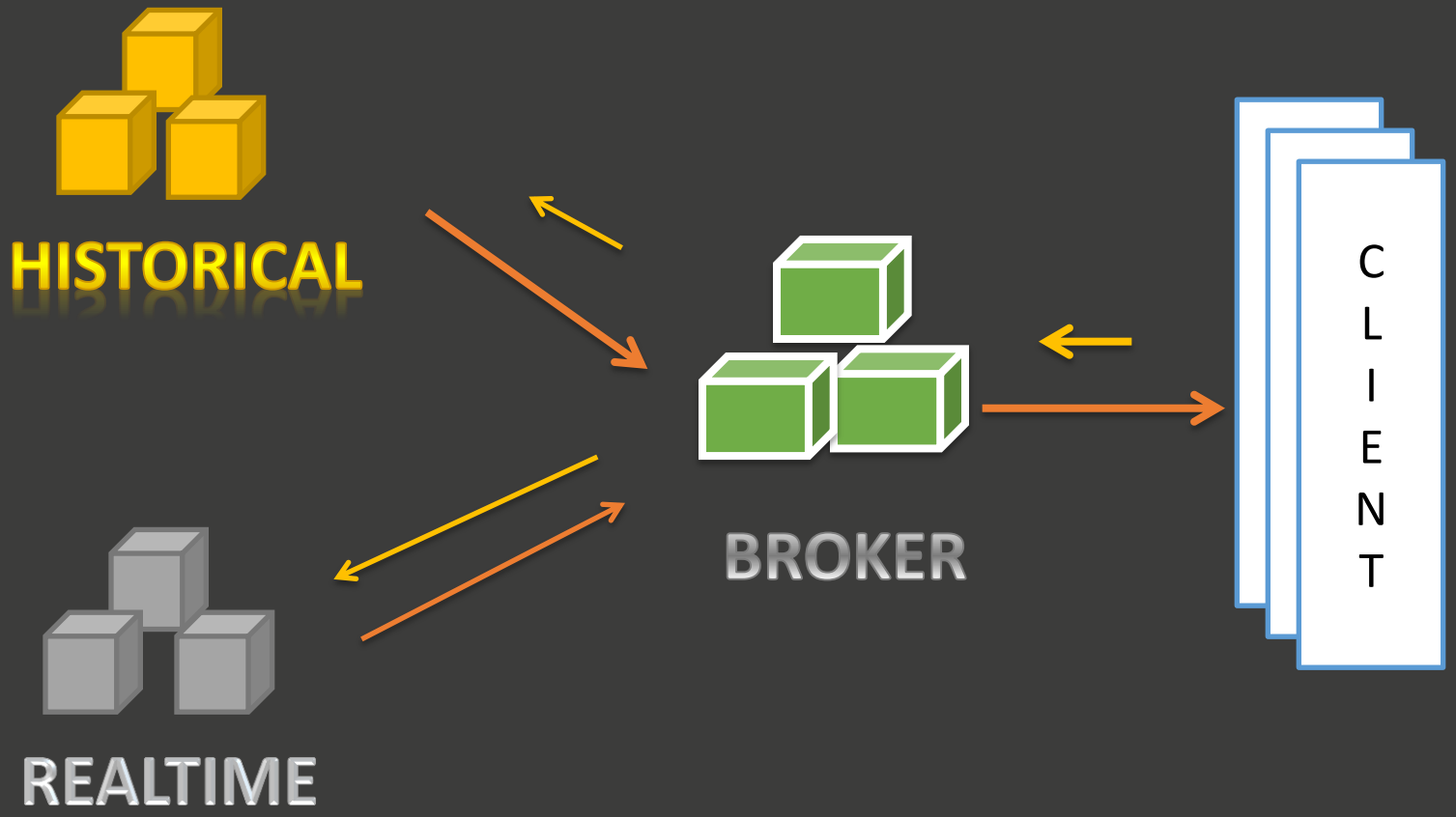
查询慢、查不了

怎么办？

维护麻烦查询效率一般

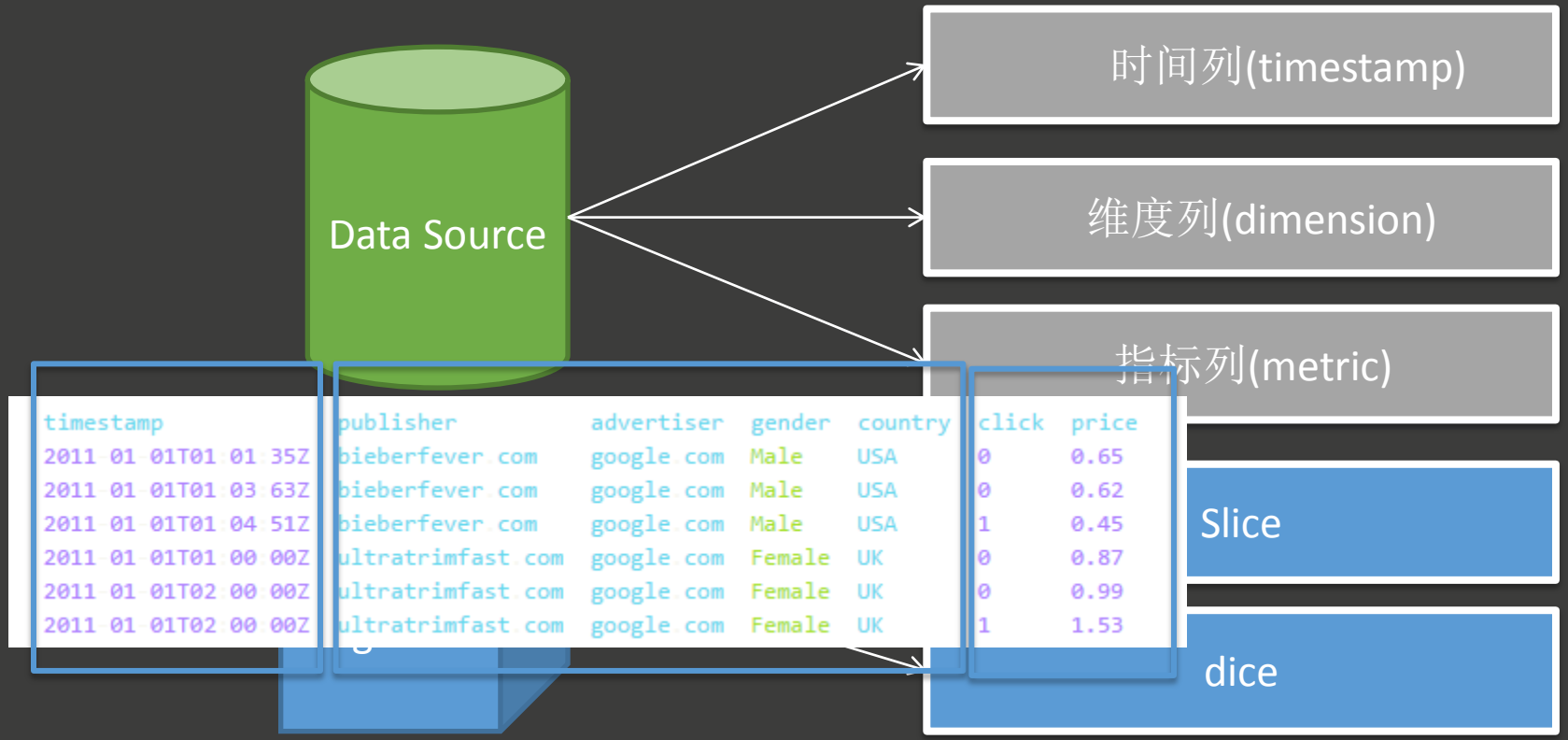


Druid主要解决问题场景



- 使用Druid，查询一个月的趋势数据，秒级以内可以返回；
- 查询多个维度的top排序，秒级内可以返回；

Druid数据结构


Data Source
时间列(timestamp)
维度列(dimension)
指标列(metric)
Slice
dice
timestamp publisher advertiser gender country click price
2011 01 01T01 01 35Z bieberfever com google com Male USA 0 0.65
2011 01 01T01 03 63Z bieberfever com google com Male USA 0 0.62
2011 01 01T01 04 51Z bieberfever com google com Male USA 1 0.45
2011 01 01T01 00 00Z ultratrimfast com google com Female UK 0 0.87
2011 01 01T02 00 00Z ultratrimfast com google com Female UK 0 0.99
2011 01 01T02 00 00Z ultratrimfast com google com Female UK 1 1.53

Druid roll-up

timestamp	publisher	advertiser	gender	country	click	price
2011-01-01T01:01:35Z	bieberfever.com	google.com	Male	USA	0	0.65
2011-01-01T01:03:63Z	bieberfever.com	google.com	Male	USA	0	0.62
2011-01-01T01:04:51Z	bieberfever.com	google.com	Male	USA	1	0.45
2011-01-01T01:00:00Z	ultratrimfast.com	google.com	Female	UK	0	0.87
2011-01-01T02:00:00Z	ultratrimfast.com	google.com	Female	UK	0	0.99
2011-01-01T02:00:00Z	ultratrimfast.com	google.com	Female	UK	1	1.53

timestamp	publisher	advertiser	gender	country	impressions	clicks	revenue	
2011-01-01T01:00:00Z	ultratrimfast.com	google.com	Male	USA	1800	25	15.70	segment1
2011-01-01T01:00:00Z	bieberfever.com	google.com	Male	USA	2912	42	29.18	
2011-01-01T02:00:00Z	ultratrimfast.com	google.com	Male	UK	1953	17	17.31	segment2
2011-01-01T02:00:00Z	bieberfever.com	google.com	Male	UK	3194	170	34.01	

Druid 数据查询

- Druid的查询使用HTTP REST风格的请求，发送给查询节点(Broker -> Historical, Realtime)
- Druid查询主要通过JSON文件指定维度和聚合

Druid的聚合查询主要有三种形式:

1. Timeseries
2. TopN
3. GroupBy

Druid 数据查询

```
{
  "queryType":"timeseries",
  "dataSource": "dap_yy",
  "granularity": {"type": "period", "period": "P1D", "timeZone":
"Asia/Shanghai"},
  "threshold": 20,
   "aggregations": [
    {
      "type": "longSum","name": "active_user","fieldName":
"active_user"
    },
    {
      "type": "longSum","name": "active_user1","fieldName":
"active_user1"
    }
  ],
  "intervals": [
       "2017-01-01/2017-01-03"
       ]
}
```

Timeseries输出每个时间粒度内指定条件的统计信息，通过filter来指定过滤条件，通过aggregations和PostAggregations指定聚合方式。

Druid 数据查询

```
{
  "queryType":"topN",
  "dataSource": "dap_xx",
  "granularity": {"type": "period", "period": "P1D", "timeZone": "Asia/Shanghai"},
  "dimension": "channel",
  "threshold": 20,
  "metric": "active_user",
  "aggregations": [
    {
      "type": "longSum","name":"active_user","fieldName":"active_user"
    },
    {
      "type": "longSum","name": "active_user1","fieldName": "active_user1"
    }
  ],
  "intervals": [
        "2017-01-01/2017-01-03"
       ]
}
```

1. TopN查询返回的是根据某一维度进行group by后再排序，返回结果集
2. TopN的查询是近似查询，每个节点会返回topK(max(1000, threshold))，*实际使用上，以top1000为例，前900个的结果项是精确的，而后100个则不能保证精确性*

Druid 数据查询

```
{
  "queryType": "groupBy",
  "dataSource": "dap_xx",
  "granularity": {"type": "period", "period": "P1D", "timeZone": "Asia/Shanghai"},
  "dimensions": ["channel","province","brand"],
  "limitSpec": { "type": "default", "limit": 10, "columns": [{"dimension": "active",
"direction": "descending", "dimensionOrder":"numeric"}]},
  "aggregations": [
    {"type": "longSum","name": "active","fieldName": "active_user"},
    {"type": "longSum","name": "active1","fieldName": "active_user1"},
    {"type": "longSum","name": "active3","fieldName": "active_user3"},
    {"type": "longSum","name": "active7","fieldName": "active_user7"},
    {"type": "longSum","name": "active30","fieldName": "active_user30"}
  ],
  "filter": {
    "type": "not",
    "field": {"type": "in", "dimension": "channel", "values": ["\"other\"",9999]}
  },
  "intervals": ["2017-01-09/2017-01-10"]
}
```

GroupBy类似于SQL中的group by操作，
能对指定的多维度进行分组。

谢 谢!