法律声明

- 本课件包括：演示文稿，示例，代码，题库，视频和声音等，小象学院拥有完全知识产权的权利；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或机构不得盗版、复制、仿造其中的创意，我们将保留一切通过法律手段追究违反者的权利。

关注 小象学院

Druid基本概念以及架构原理

目录

- Druid是什么？
- Druid概念与特性
- Druid架构以及原理
- 第三方依赖

Druid是什么？

Druid是针对时间序列数据提供低延时的数据写入以及快速交互式查询的分布式OLAP数据库。

Druid是什么？

针对时间序列数据 TSDB（Time-series database）

1. InfluxDB
2. Graphite
3. OpenTSDB

指标系统

1.容量可控（数据生命周期 or 下采样）

2.Schema less

3.数据可视化

分布式OLAP数据库

1. ES – 明细数据检索
2. Kylin – 预计算 + kv存储
3. Presto – 可直接读HDFS文件的查询引擎

低延时的数据写入以及快速交互式查询

Druid是什么？

Druid擅长的部分

- 对于大部分查询场景可以亚秒级响应
- 事件流实时写入与批量数据导入兼备
- 数据写入前预聚合节省存储空间，提升查询效率
- 水平扩容能力强
- 社区活跃

Druid短板的部分

- 不支持Join
- 大数据量场景下明细查询有瓶颈

Druid是什么？

我是否需要使用Druid？

- 处理时间序列事件
- 快速的聚合以及探索式分析
- 近实时分析亚秒级响应
- 存储大量（TB级、PB级）可以预先定义若干维度的事件
- 无单点问题的数据存储

Druid基本概念与特性

- 数据模型
  - 事件
  - Roll-up
  - 数据存储
- 数据写入
  - 实时数据写入
  - 离线数据写入
- 数据查询
  - 查询模式
  - DSL查询介绍
  - SQL查询介绍

Druid基本概念与特性-事件

事件

Druid将一个事件分解成三种数据类型:

Timestamp column（数据分片）；

Dimension columns（过滤数据，分组）；

Metric columns（聚合计算）；

Timestamp	city	brand	Down(下行流量)	count
2017-04-01T10:00:01Z	北京	OPPO	10	1
2017-04-01T10:00:10Z	上海	vivo	50	1
2017-04-01T10:00:22Z	北京	OPPO	20	1
2017-04-01T10:01:02Z	北京	OPPO	60	1
2017-04-01T10:01:05Z	上海	vivo	30	1
2017-04-01T10:01:30Z	北京	vivo	10	1
2017-04-01T10:01:45Z	上海	vivo	20	1

Druid基本概念与特性-Roll-up

预聚合

Druid读取数据时并不会直接存储原始数据, 而是使用Roll-up方式进行聚合计算, 最终存储的是聚合后的结果。

Timestamp	city	brand	Down(下行流量)	count
2017-04-01T10:00:00Z	北京	OPPO	30	2
2017-04-01T10:00:00Z	上海	vivo	50	1
2017-04-01T10:01:00Z	北京	OPPO	60	1
2017-04-01T10:01:00Z	北京	vivo	10	1
2017-04-01T10:01:00Z	上海	vivo	50	2

Druid基本概念与特性-Segment

Segment

- Druid将索引数据保存到Segment文件中,Segment文件根据时间进行分片。Segment中会保存维度、指标以及索引信息。
- 将行式数据转换为列式存储结构: 按需加载，提高查询速度
- 有三种类型的数据结构:
  - timestamp列, long数组
  - 指标列: int数组或float数组
  - 维度列: 支持过滤和分组。使用压缩的BitMap索引。

Druid基本概念与特性-Segment

1. 字典，将列的所有值进行编码

```
{
  “北京”：0,
  “上海”：1
}
```

2. 列的数据, 要保存的是每一行中这一列的值, 使用字典中的编码

[0, 1, 0 ，0 ，1]

3. Bitmaps – 对于列的每一个不同的值都有

“北京”：[1, 0, 1, 1, 0]

“上海”：[0, 1, 0, 0, 1]

在最坏情况下前面两种会随着数据量的大小而线性增长.

而BitMap的大小则等于数据量大小 * 列的个数.

Druid基本概念与特性-实时数据写入

接入方式

- 内置KafkaIndexingService机制，无缝对接Kafka
- 使用Realtime Node或Traquality管理写入数据，通过扩展Firehose对接多种上游数据

数据格式

- JSON, CSV, TSV

Druid基本概念与特性-离线数据写入

支持文件类型（HDFS, S3, ...）

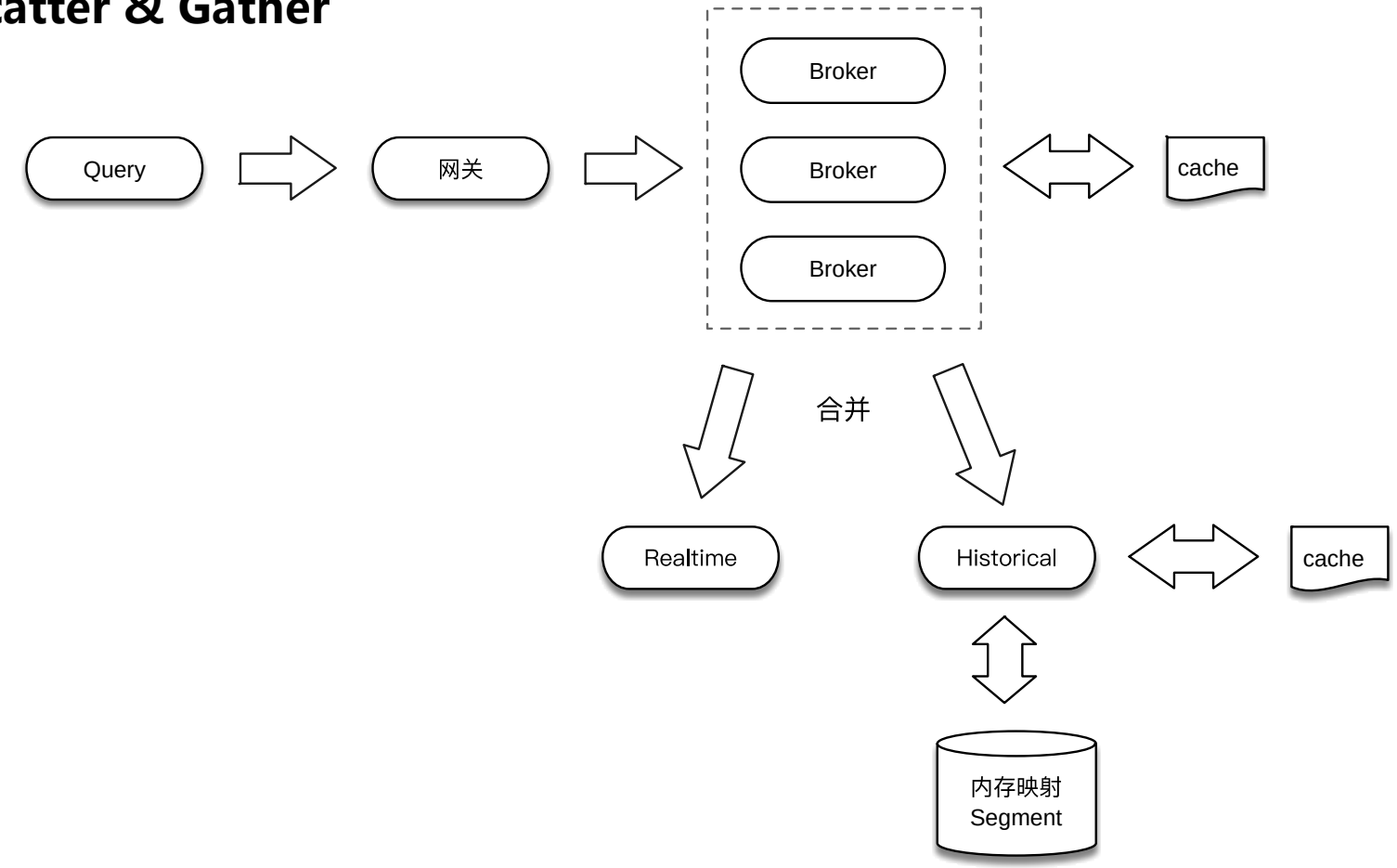
- TextFile(JSON, CSV, TSV), ORCFile, ParquetFile
- 灵活的插件机制，支持其他文件格式的解析

索引方式

- 内部 Indexer (数据集<1G, 开发测试用)
- 内置基于MapReducer任务的 Indexer

Druid基本概念与特性-查询模式

Scatter & Gather



Druid基本概念与特性-查询接口

查询方式

基于HTTP/JSON的DSL

基于HTTP/JSON的SQL

JDBC方式

```
{
  "queryType": "groupBy",
  "dataSource": "sample_datasource",
  "granularity": "day",
  "dimensions": ["country", "device"],
  "limitSpec": { "type": "default", "limit": 5000, "columns": ["country", "data_transfer"] },
  "filter": {
    "type": "and",
    "fields": [
      { "type": "selector", "dimension": "carrier", "value": "AT&T" },
      { "type": "or",
        "fields": [
          { "type": "selector", "dimension": "make", "value": "Apple" },
          { "type": "selector", "dimension": "make", "value": "Samsung" }
        ]
      }
    ]
  },
  "aggregations": [
    { "type": "longSum", "name": "total_usage", "fieldName": "user_count" },
    { "type": "doubleSum", "name": "data_transfer", "fieldName": "data_transfer" }
  ],
  "postAggregations": [
    { "type": "arithmetic",
      "name": "avg_usage",
      "fn": "/",
      "fields": [
        { "type": "fieldAccess", "fieldName": "data_transfer" },
        { "type": "fieldAccess", "fieldName": "total_usage" }
      ]
    }
  ],
  "intervals": [ "2012-01-01T00:00:00.000/2012-01-03T00:00:00.000" ],
  "having": {
    "type": "greaterThan",
    "aggregation": "total_usage",
    "value": 100
  }
}
```

Druid基本概念与特性-查询接口

查询方式

基于HTTP/JSON的DSL

基于HTTP/JSON的SQL

JDBC方式

```
{
  "query" : "SELECT COUNT(*) FROM data_source WHERE foo = 'bar' AND __time > TIMESTAMP '2000-01-01 00:00:00'",
  "context" : {
    "sqlTimeZone" : "America/Los_Angeles"
  }
}
```

Druid基本概念与特性-查询接口

查询方式

基于HTTP/JSON的DSL

基于HTTP/JSON的SQL

JDBC方式

```
String broker = PropertyUtils.getStringAppProperty("broker", "127.0.0.1:8082");
String url = "jdbc:avatica:remote:url=http://" + broker + "/druid/v2/sql/avatica/";

JDBCHelper jdbc = null;
try {
  jdbc = new JDBCHelper(url);
}
catch (SQLException e) {
  e.printStackTrace();
  System.exit(-1);
}

/*执行查询*/
logger.info("== ExecuteQuery ==");
String sql = "SELECT COUNT(*) AS cnt FROM ZL_ARCH_HOLMES_ERROR_SAVE";
try {
  ResultSet resultSet = jdbc.executeQuery(sql);
  while (resultSet.next()) {
    logger.info("SQL: " + sql + "; Res: " + resultSet.getLong("cnt"));
  }
}
catch (SQLException e) {
  e.printStackTrace();
}
```

Druid基本概念与特性-查询接口

原生查询类型

- Timeseries 返回时间序列
- Top N 根据除时间以外的一个维度列聚合
- Group by 根据除时间以外的一个或多个维度列聚合
- Select 查询Druid原始数据
- Search 查询维度列的值
- Time boundary 查询一个数据源可查数据的时间范围
- Metadata query 元数据查询
- Scan 查询

Druid基本概念与特性-小结

接入方式

- 实时数据接入
  - Kafka-index-service
  - realtimeNode&traquality

- 离线数据导入
  - MR – 高吞吐量的离线任务
  - Indexer – 通过实时任务索引离线数据

Druid基本概念与特性-小结

特有的概念

- Roll up
  - 数据预聚合，将一定时间粒度范围内、维度列相同的数据进行聚合
- Segment
  - 按照时间对数据进行shard
  - 按列存储
  - 包含数据以及索引

Druid基本概念与特性-小结

查询

- Scatter & Gather
- 查询方式
  - Http+Json
  - Http+SQL
  - Jdbc
- 基本查询类型
  - Timeseries
  - TopN
  - Groupby
  - Select
  - Search
  - Time boundary
  - Metadata

Druid基本概念与特性-小结

Druid VS ES

ES 支持全文检索 & 结构化索引

=> 结构化数据与非结构化数据，明细查询与聚合能力

=> 存储空间开销大

=> 数据的写入与聚合开销大

Druid 结构化索引 & 预聚合

=> 结构化数据 较弱的明细查询能力

=> 存储空间更小

=> 针对数据的写入与聚合进行优化

Druid基本概念与特性-小结

Druid VS KV存储（Hbase, Cassandra)

KV存储通过预计算来实现聚合，key涵盖了查询参数，值就是查询结果

=> 查询速度极快

=> 损失了查询的灵活性，复杂的场景下，预计算过程可能十分耗时

=> 只有前缀拼配一种索引方式，大数据量下负载过滤条件性能下降

=> 缺少聚合下推的能力

Druid基本概念与特性-小结

Druid VS SQL on Hadoop（ Presto, Spark SQL)

SQL on Hadoop

=> SQL支持强大

=> 无冗余数据，不需要预处理

=> 分钟级响应

=> QPS低

Druid

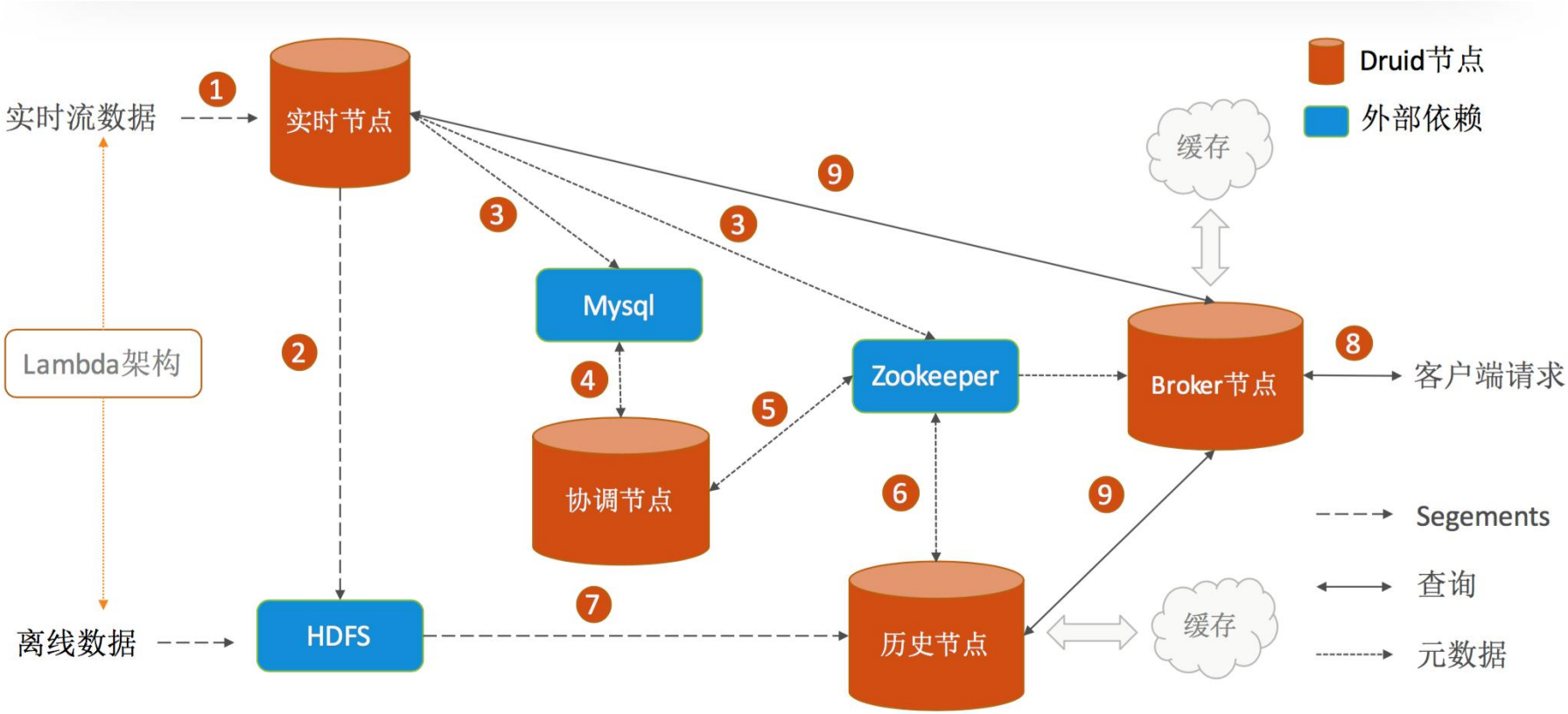
=> SQL支持有限

=> 必须预先定义维度指标

=> 亚秒级响应

=> 高并发

Druid架构原理


Druid节点
外部依赖
实时流数据
1
实时节点
缓存
9
3
3
Mysql
Lambda架构
2
4
Zookeeper
Broker节点
8
客户端请求
5
协调节点
6
9
Segements
查询
元数据
离线数据
HDFS
7
历史节点
缓存

Druid架构原理-外部依赖

- 重度依赖Zookeeper
  - Leader选举
  - 服务发现
  - 任务发布
  - 数据发布
- 元数据存储依赖Mysql、PostgreSql
  - Segments
  - Rule
  - Config
  - Tasks
- DeepStorage HDFS, S3
  - Segments
  - Indexing-log

Druid架构原理-角色节点

- Coordinator – Master节点，管理集群数据视图，数据load/drop
- Historical node – 历史节点，负责历史窗口内数据的查询
- Broker – 查询节点，查询拆分，结果汇聚
- Indexing Service – 一套实时所有任务的调度服务
  - Overlord – 调度服务的Master节点，负责接收任务，管理任务状态
  - MiddleManager – worker节点，接收任务，启动任务
  - Peon – 实际的任务进程

Druid架构原理-角色节点

Coordinator

- 从metastore中读取Segment的元数据,并决定哪些Segments需要被加载到集群中
- 使用ZooKeeper查看已经存在的历史节点都有哪些，了解集群各个节点负载情况。
- 创建一个ZK的条目告诉历史节点加载、删除、或者移动Segments

Druid架构原理-角色节点

Historical

- 提供对Segment的数据查询服务
- 与ZooKeeper通信，上报节点信息，告知ZK自己拥有哪些Segments
- 从ZooKeeper中获取执行任务:
    - 从DeepStorage加载新的Segments(从实时节点转存过来的不可变的Segments)
    - 删除自己已经保存的旧的Segments
- ShareNothing的架构：任何一个历史节点挂掉都没有关系

Druid架构原理-角色节点

Broker

- 负责将查询请求分发到历史节点和实时节点,并聚合这些节点返回的查询结果数据.
- Broker节点通过zookeeper知道Segment都存放在哪些节点上.

Druid架构原理-角色节点

Indexing Service

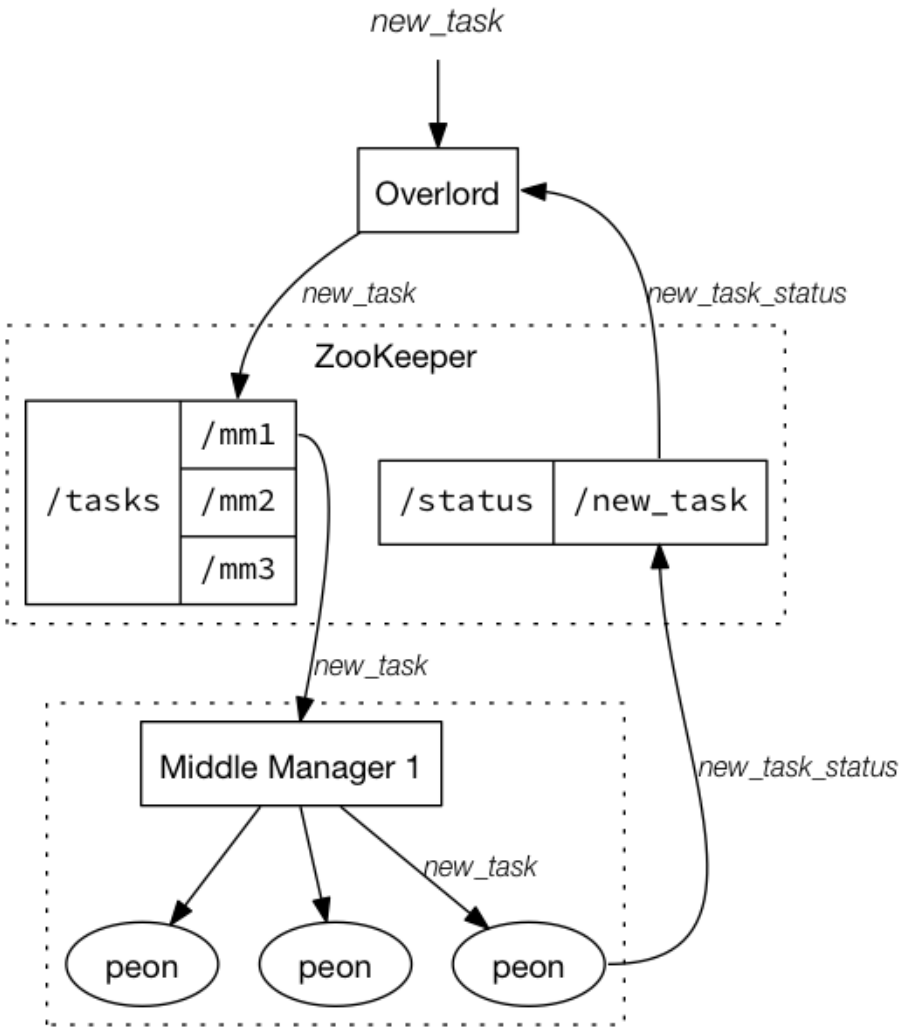
任务管理

•Overlord接受任务

•通过zk将任务信息分配给MM

•MM领取任务，创建Peon进程

•通过zk发布任务状态



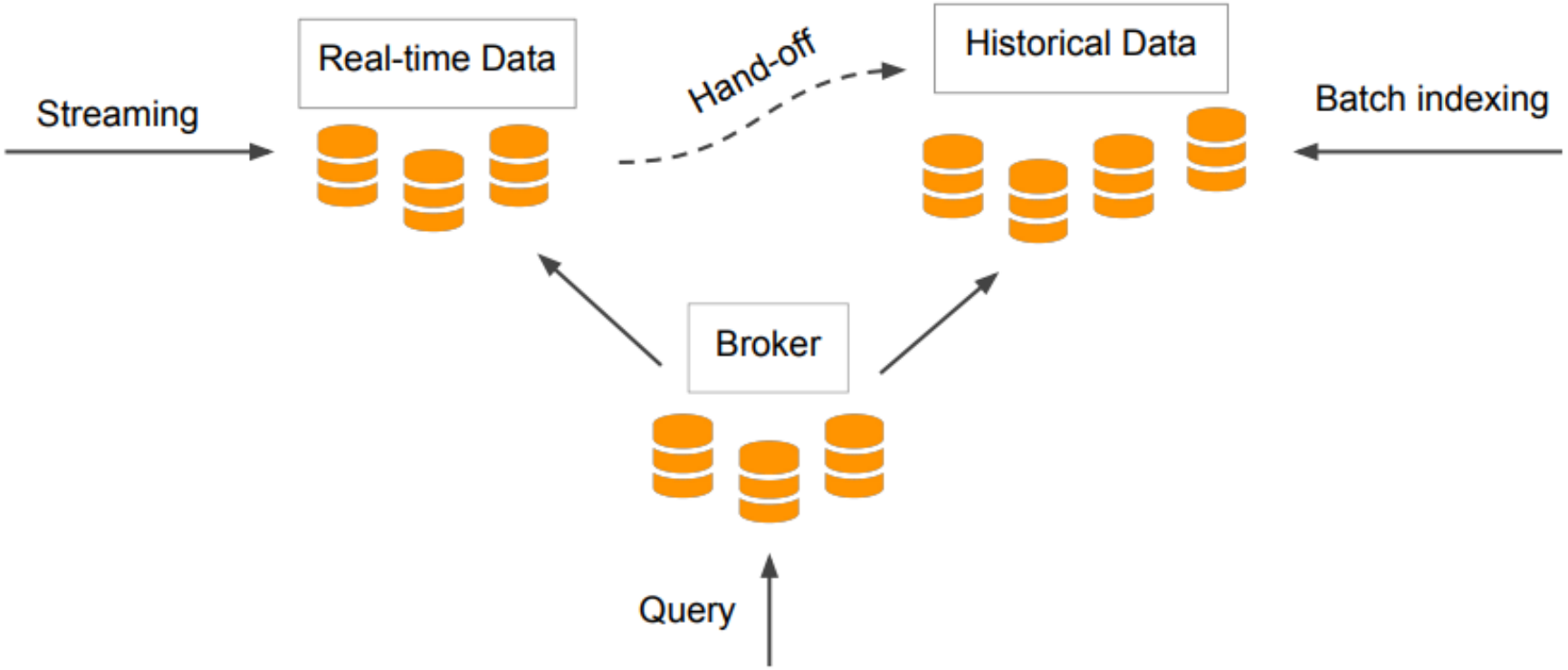
小象学院 ChinaHadoop.cn

Druid架构原理-角色节点

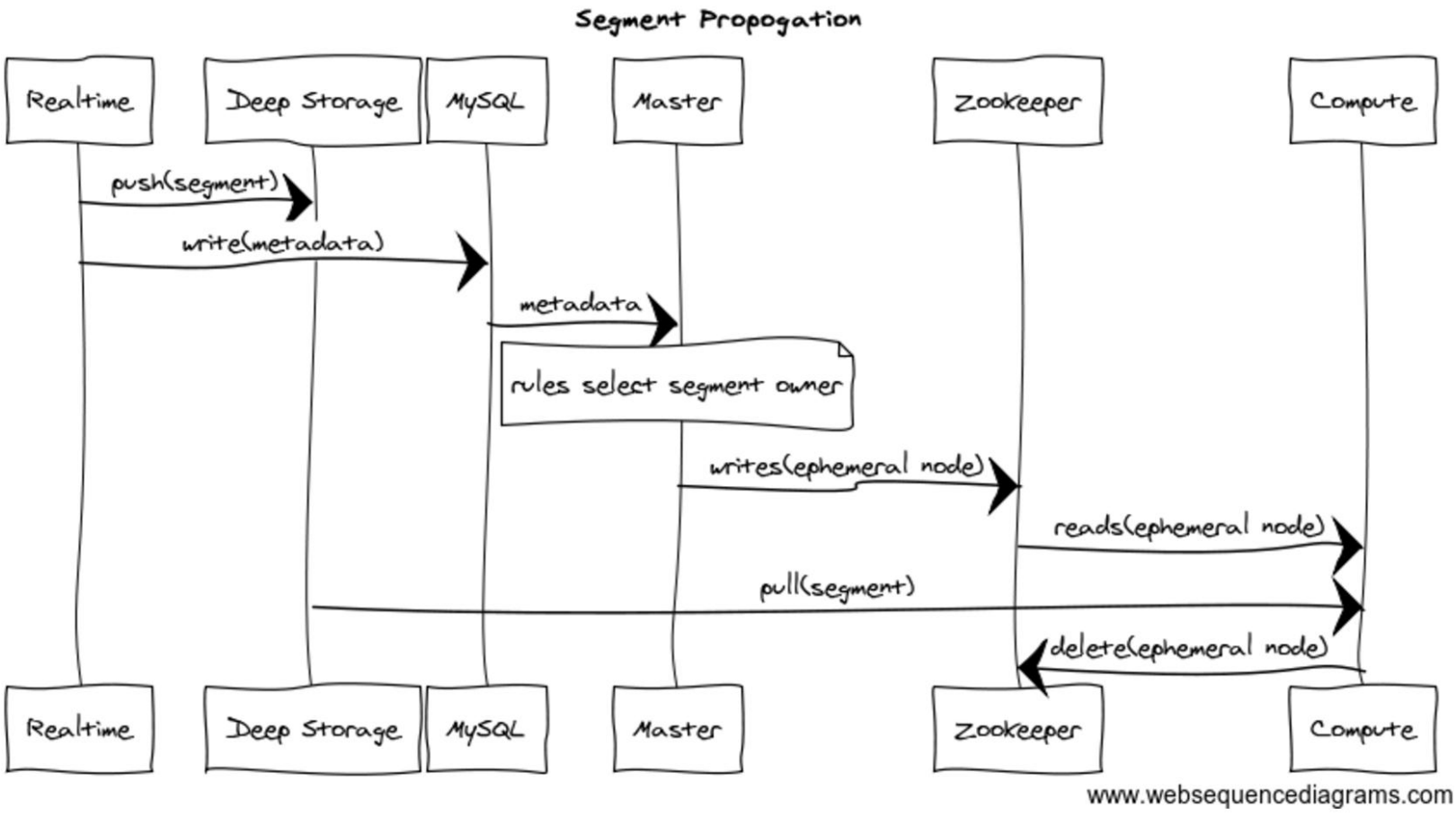
Indexing Service

- Peon 实时读取数据,实时索引数据,创建Segments.
- Peon 响应Broker的查询请求并返回结果给Broker节点.
- 数据一旦被实时节点（Peon）接收就可用于查询.
- 周期性地将Segments转存到DeepStorage, 并通知ZooKeeper
- 历史节点会接收到这个通知发现有新的Segments需要被加载/或被删除. 只有当Segments最终转存到历史节点,可以从历史节点查询时, 实时节点之前保存的Segments才会被删除。

Druid架构原理-角色节点



Druid架构原理-角色节点



Druid架构原理-第三方依赖

Zookeeper依赖

- Leader选举
    - Coordinator选举
    - Overlord选举
- 服务发现
- 任务发布 – indexing service 任务管理
- 数据发布
    - Publishing protocol
    - Load/Drop protocal

Druid架构原理-第三方依赖

元数据存储依赖Mysql、PostgreSql

- Segments
- Rule
- Config
- Tasks

```
+-----------------------+
| druid_audit           |
| druid_config          |
| druid_dataSource      |
| druid_pendingSegments |
| druid_rules           |
| druid_segments        |
| druid_supervisors     |
| druid_tasklocks       |
| druid_tasklogs        |
| druid_tasks           |
+-----------------------+
```

Druid架构原理-第三方依赖

DeepStorage HDFS, S3

- Segments

/user/druid/segments/<datasource>/<timeFrom/timeTo>/<version>/xxx_index.zip

/user/druid/segments/<datasource>/<timeFrom/timeTo>/<version>/xxx_descriptor.json

- Indexing-logs

/user/druid/indexing-logs/<taskid>

- hadoop-tmp

/user/druid/hadoop-tmp/<datasource>/xxxx

联系我们

小象学院：互联网新技术在线教育领航者

– 微信公众号：**小象学院**