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实践介绍

目录

- 容错设计
- 指标监控
- 运维实践
- 案例分享

容错设计

- 基本角色
  - Coordintor
  - Historical
  - Broker
- Standalone Realtime Node
- Tranquility+Indexing-service
- Kafka-indexing-service
- 第三方依赖

容错设计-基本角色

Coordinator

- 角色职能
  - 管理Segment的生命周期（发出加载/卸载的指令）
- 失败现象
  - 新Segment无法加载 – 实时写入任务无法完成handoff
  - 应当过期的数据一致存在
  - CoordinatorUI
- 容错设计
  - 支持主备

容错设计-基本角色

Historical

- 角色职能
  - 负责加载/卸载Segment（执行加载/卸载的指令）
  - 提供历史数据窗口内数据的查询
- 失败现象
  - 集群数据视图将不完整
- 容错设计
  - Segment默认2副本存储，保证一台Historical节点挂掉对数据视图无影响
  - Historical节点挂掉，其上丢失的Segment将由其他Historical加载

容错设计-基本角色

Broker

- 角色职能
  - 负责接收查询
- 失败现象
  - 无法响应查询
- 容错设计
  - Broker多实例部署，前置VIP

容错设计-Standalone Realtime Node

- 角色职能
  - 消费数据，构建Segment
  - 提供实时窗口的数据查询
- 失败现象
  - 数据停止写入
  - 实时窗口内数据缺失
- 容错设计
  - 如果是Topic是单partition，可以多副本启动

容错设计-Tranquility+Indexing-service

Tranquility进程

•角色职能

- 消费数据，推送到Druid

•失败现象

- 数据停止写入

•容错设计

- 可以多个Tranquility实例消费数据
- 进程重启后，接着上次提交的offset继续消费

容错设计-Tranquility+Indexing-service

Overlord进程

•角色职能

- 接受实时任务的提交
- 接受来自Peon进程的元数据修改请求

•失败现象

- 新实时任务无法提交
- 原有实时任务元数据修改请求失败，导致任务失败

•容错设计

- 支持主备

容错设计-Tranquility+Indexing-service

MiddleManager进程

- 角色职能
  - 领取实时任务，启动实时任务
- 失败现象
  - 该MiddleManager上的实时任务将会全部失败
- 容错设计
  - 多实例部署

容错设计-Tranquility+Indexing-service

Peon

•角色职能

- 接受数据，生成Segment
- 提供实时窗口内数据的查询服务

•失败现象

- 实时任务写入停止
- 实时窗口内数据缺失

•容错设计

- 启动多副本

容错设计-Kafka-Indexing-service

Overlord进程

- 角色职责
  - 运行每个数据源的Supervisor
    - 管理实时任务生命周期（启动、停止）
    - 提供实时任务需要的API服务（如SegmentAllocate、LockAcquire）
  - 接受实时任务的提交
  - 接受来自Peon进程的元数据修改请求
- 失败现象
  - 新实时任务无法提交
  - 原有实时任务无法结束，直到调用Overlord请求失败，导致任务失败
- 容错设计
  - 支持主备
  - Supervisor持久化，重启可恢复状态

容错设计-Kafka-Indexing-service

Peon

•角色职能

- 消费Kafka数据，生成Segment
- 提供实时窗口内数据的查询服务

•失败现象

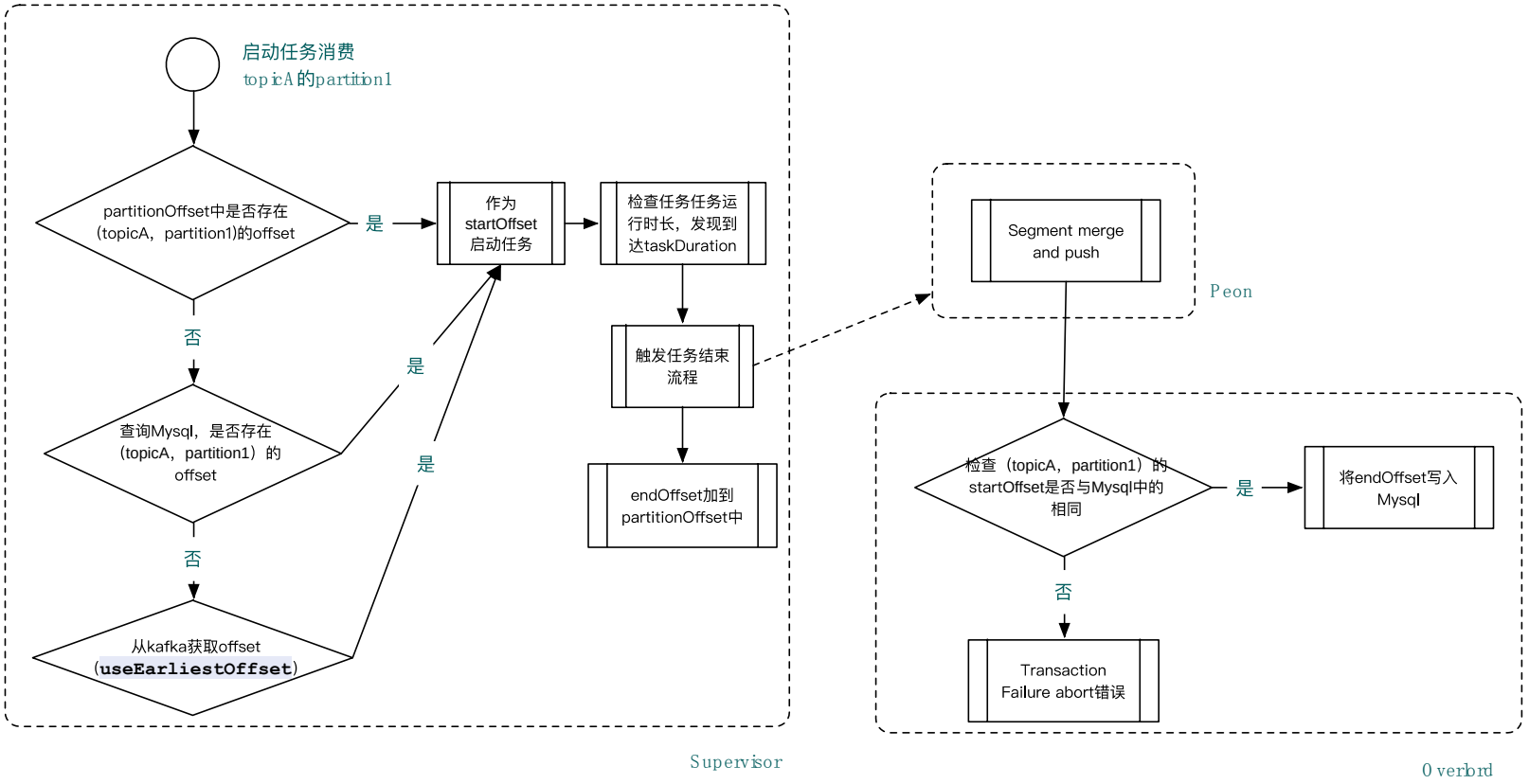
- 实时任务写入停止
- 实时窗口内数据缺失

•容错设计

- 启动多副本
- 任务失败或会在其他MiddleManager重新启动
- 任务重新启动后，会从该任务原始startOffset开始消费

容错设计-Kafka-Indexing-service

Offset如何管理？



容错设计-第三方依赖

Zookeeper

- 角色职能
  - Segment Change Request发布
  - 实时任务发布、状态发布
  - 集群角色服务发现
- 失败现象
  - 集群数据视图冻结
  - Indexing-service服务不可用
  - 不会影响查询已经加载好的数据
- 高可用

容错设计-第三方依赖

Mysql

- 角色职能
  - Segment元数据存储
- 失败现象
  - 集群数据视图冻结
  - 不会影响查询当前已经加载的数据
- 高可用
  - 主从

容错设计-第三方依赖

HDFS

•角色职能

- 参与Segment的发布流程，写入Segment、读出Segment

•失败现象

- 集群数据视图冻结
- 不会影响查询当前已经加载的数据

•高可用

指标监控

- 指标上报方式
  - Logging Emitter
  - Http Emitter
  - Composing Emitter
  - 其他 Kafka/Graphite/… Emitter
- 常用指标
  - 查询相关
  - 数据写入相关
  - Segment管理相关
  - JVM相关
- Graphite简介

指标监控-指标上报方式

Kafka-Emitter、Graphite-Emitter

- 自建指标系统，如利用Druid收集指标，可以用是Kafka-Emitter
- 使用第三方指标系统，使用对应系统的Emitter，如Graphite-Emitter

指标监控-常用指标

指标基本模式

使用MetricsMonitor添加指标

- SysMonitor
- JvmMonitor
- RealtimeMetricsMonitor

指标非常丰富，需要有所取舍

指标监控-常用指标

查询相关指标

•Broker、Historical、Realtime

- query/time – datasource、queryType （统计分位数）
- query/bytes – datasource、queryType
- query/scan/pending

指标监控-常用指标

数据写入相关

•Realtime

- ingest/events/throwAway
- ingest/events/unparseable
- ingest/events/processed
- ingest/events/messageGap
- ingest/kafka/lag
- task/run/time

指标监控-常用指标

Segment管理相关

•Coordinator

- segment/assigned/count
- segment/moved/count
- segment/dropped/count
- segment/unneeded/count
- segment/overShadowed/count
- segment/loadQueue/count
- segment/dropQueue/count
- segment/unavailable/count
- segment/underReplicated/count

指标监控-常用指标

Segment管理相关

•Historical

- segment/used
- segment/usedPercent
- segment/count
- segment/pendingDelete

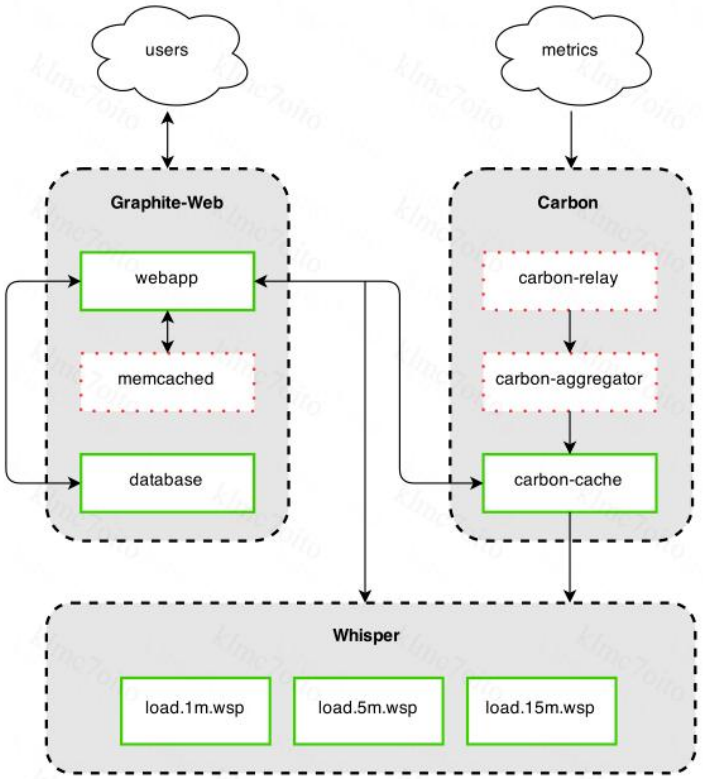
指标监控-常用指标

JVM相关

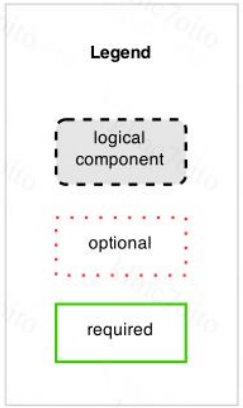
- jvm/mem/max
- jvm/mem/used
- jvm/gc/count
- jvm/gc/time

指标监控-Graphite简介

Graphite



- Carbon - 接收时间序列数据
- whisper - 用来存储时间序列数据
- graphite-web –从whisper数据库获取时间序列数据并且进行展示



指标监控-Graphite简介

Graphite

- 消息格式
  - <namespace>.<dim1>.<dim2>...<dimn> value timestamp
- 生命周期

```
[druid]
pattern = ^druid\.
retentions = 1m:30d,10m:300d
```

- 下采样规则

```
[sum]
pattern = \.sum$
xFilesFactor = 0
aggregationMethod = sum
```

指标监控-Graphite简介

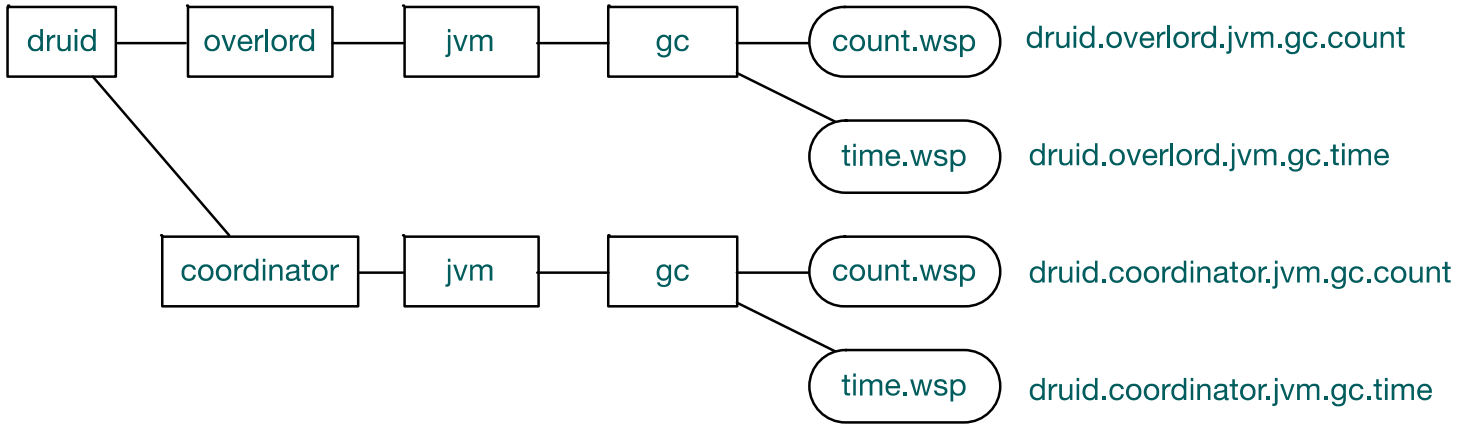
Graphite

•容量规划

- 单个文件大小计算

  计算规则：每一个whisper数据源（一个指标）有16-byte的元数据信息，数据源中的每一个档案（archive）有12-byte的档案信息，每一个数据点占12-bytes的空间

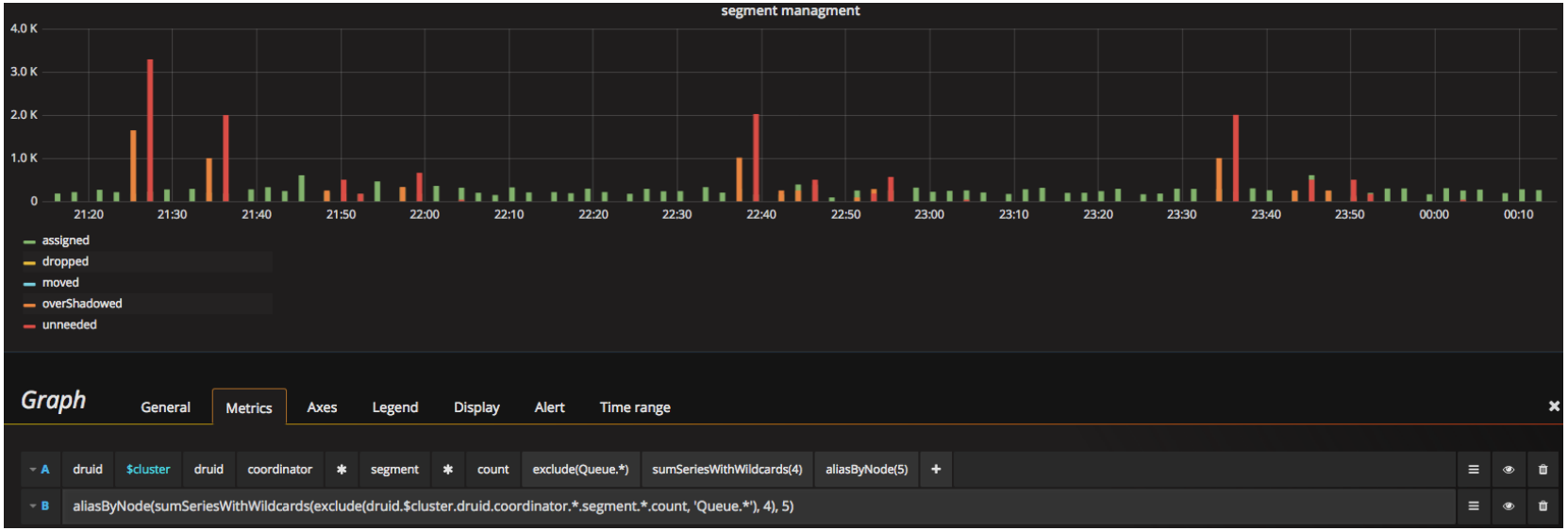
- 指标数量-对应whisper文件数



指标监控-Graphite简介

Graphite

- 查询Function



基于Grafana的查询配置

运维实践

- 集群升级
- 资源隔离

运维实践-集群升级

官方推荐升级顺序

- Historical
- Overlord
- MiddleManager
- Standalone Real-time
- Broker
- Coordinator（or merged Coordinator+Overlord）

本质上并没有严格的顺序问题

TIPS：先Historical，后Coordinator

运维实践-集群升级

- Historical滚动升级，因为数据双副本，所以不会影响查询

  GET http://<historical IP>:8083/druid/historical/v1/loadstatus 返回

  {"cacheInitialized": true}

- Overlord、Coordinator支持主备，滚动升级就好
- Broker升级

  •GET http://<brokerIP>:8082/druid/broker/v1/loadstatus 返回

  {“inventoryInitialized”:true}

  •但是对于SQL schema的构建仍然可能为完成，需要等待一段时间

运维实践-集群升级

- MiddleManager升级
  - Restore-based Rolling restart
    - 需要配置druid.indexer.task.restoreTasksOnRestart=true
    - MiddleManager停止是会让peon保存运行状态，启动后接着消费
  - Graceful-termination-based Rolling restart
    - POST <MiddleManager_IP:PORT>/druid/worker/v1/disable
    - 防止新任务调度到该MM，待旧任务跑完后，直接重启

运维实践-资源隔离

资源隔离级别

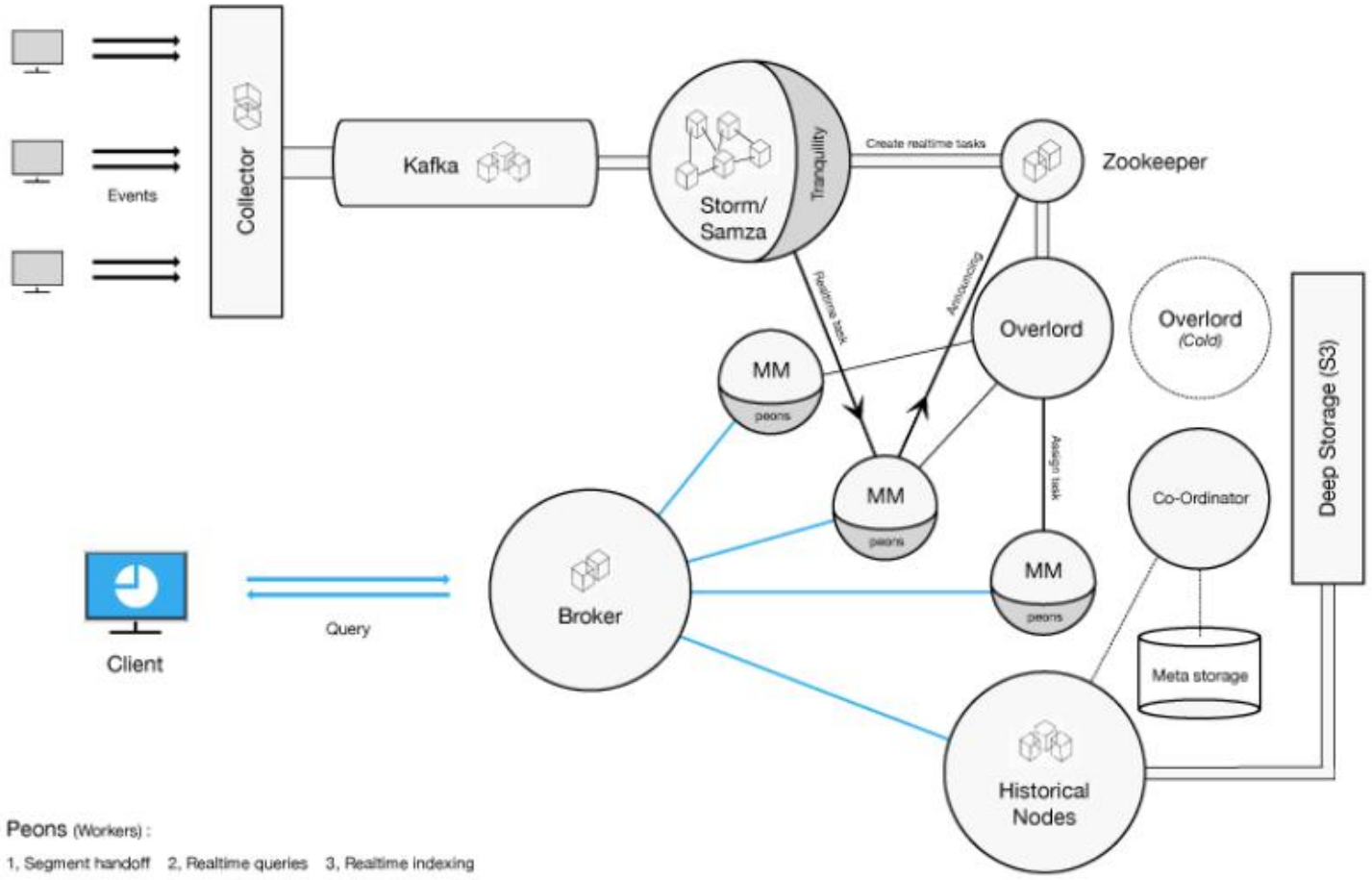
- 查询
  - Broker隔离（网关）
  - 实时节点隔离（天然隔离）
  - Historical节点隔离（分Tier）
- 数据写入
  - 实时节点隔离（天然隔离）
  - Overlord服务隔离（Overlord Sharding方案）

案例分享

- 案例分享
  - 指标监控
  - 大盘展示
  - 对比Flink聚合方案

案例分享

通用集成架构



案例分享-指标监控系统

常用指标类型

- count – flush单元内的累计值 – sum
- gauge – flush单元内最后一个值 – longLast, doubleLast
- timer – flush单元内计算均值、分位数等
    - 均值： sum + count + postAggregator
    - 分位数：approxHistogram

案例分享-指标监控系统

数据生命周期管理

- 支持数据自动过期
- 通过tier分冷热数据
- 不同queryGranularity的Segment可以共存，可以模拟下采样操作

案例分享-指标监控系统

监控GPU状态

字段含义	是否维度	指标类型	Druid聚合算子
服务器地址	是		
GPU编号	是		
校验错误计数	否	count	longSum
风扇转速	否	gauge	longLast
内存利用率	否	gauge	longLast

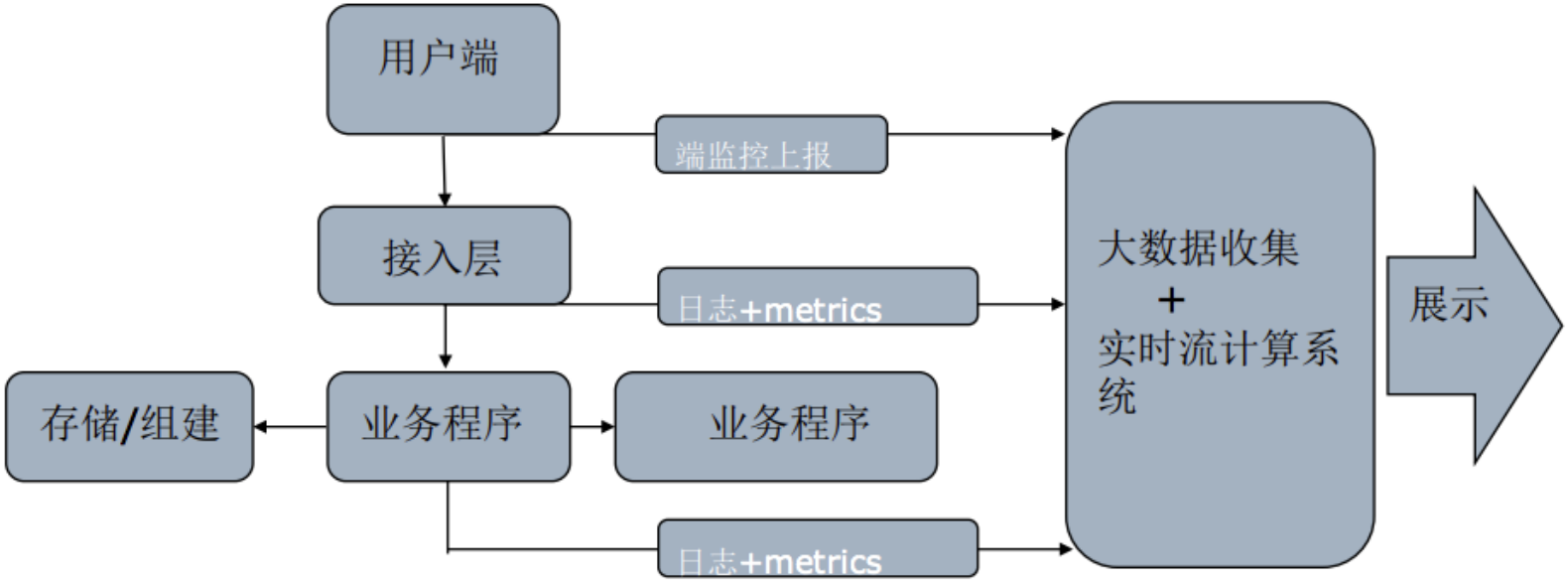
案例分享-大盘展示

客服服务状态

查询需求	Druid实现
机器人UV、人工UV	Distinct count 1.HyperUnique aggregator 2.Cardinality Aggregator 3.精确nest groupBy查询
点赞率、使用率	PostAggregator直接计算
FAQ Top5、关键字Top5	TopN查询

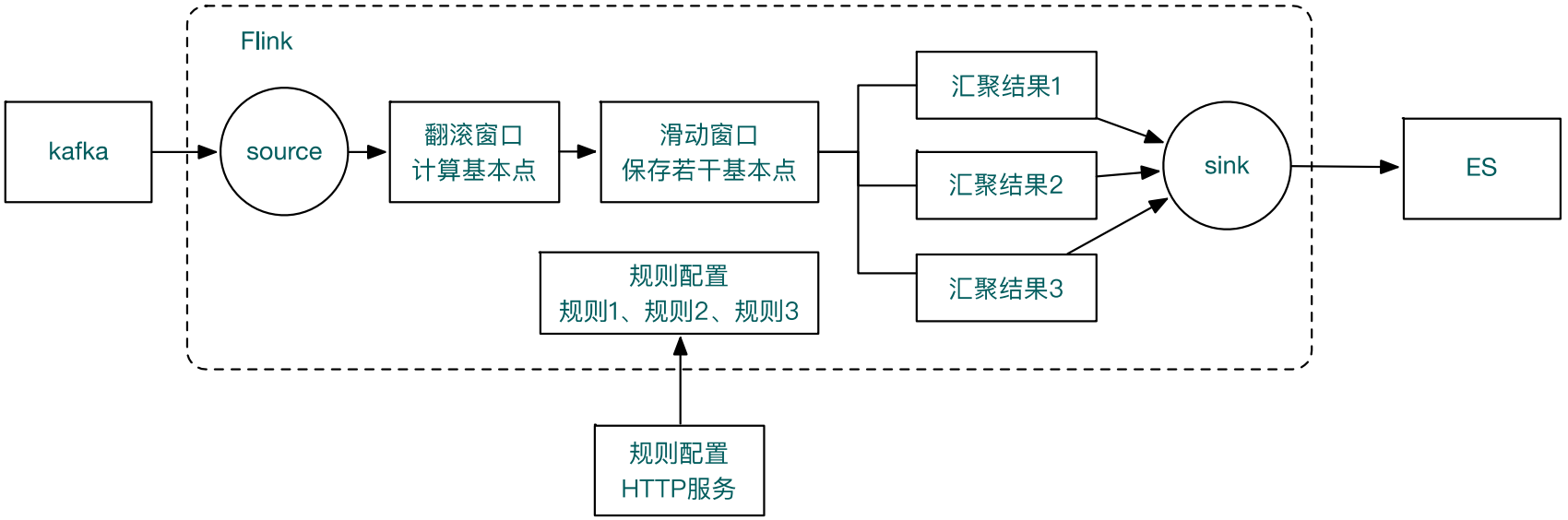
案例分享-与Flink聚合方式的比较

全链路服务监控



案例分享-与Flink聚合方式的比较

全链路服务监控



- 基本点计算 ->queryGranularity
- 滑动窗口
  - 窗口长度 – 决定聚合规则可聚合的时间范围
  - 滑动频率 – 数据输出频率
- 汇聚结果 – 类似中间结果

案例分享-与Flink聚合方式的比较

全链路服务监控

与Druid实现的区别

- 基本点计算 ->queryGranularity 相同
- 滑动窗口
  - 窗口长度 – 决定聚合规则可聚合的时间范围 Druid是任意范围
  - 滑动频率 – 数据输出频率，决定了数据实时可见性 Druid是数据逐条可见
  - 滑动窗口的聚合值 – Druid是滚动窗口的聚合值（看业务需求）
- 汇聚结果 – 类似中间结果
  - 规则越多，数据膨胀越厉害 Druid是同一份数据
- 数据消费与后续处理可设置不同并行度提升，数据处理能力可扩展
  - Druid数据消费与索引构建在同一线程中，数据处理能力依赖有限

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**