**Druid优化文档**

**段卫东**

# Druid现状：

A.Druid日志未全部开发，遇到问题，无法从日志里面快速定位问题。

B.Druid监控不全面，现有的监控，仅仅是服务器参数的监控。缺少Druid服务运行情况的监控以及业务上的监控。

C.未形成一套完备的迭代（数据为支撑，理论为依据）为基础的优化方案。

# 优化期望：

Druid高效稳定。

# 优化方面：

## 集群：

集群的优化只要体现在集群的架构组成，以及参数的优化上。集群的配置没有最好的配置，只有最适合业务场景的配置。

### A.集群的监控。

集群的监控是集群优化的基础，通过监控指标，可以反映出当前集群的特点，再有针对的去调整集群的配置，使其更好的服务于业务。

监控主要分为以下几个方面：一下监控涉及到统计的，默认是一分钟一次。druid.monitoring.emissionPeriod

**broker**：

query/time :一次查询所花费的时间。可以统计出集群的平均响应时间，慢查询，查询QPS。

query/bytes:一次查询返回的数据量。可以统计出平均查询返回的数据量，返回数据量大的查询。对于集群网络IO瓶颈，可以考虑从此参数找出优化方向，比如借助HTTP cache。

query/node/time:查询historical结点或者realtime结点的时间。可以统计出各个查询结点的各自的响应时间，作为解决单结点负载过高或者其他异常的依据。

query/node/bytes:查询historical结点或者realtime结点的返回结果字节数。可以统计出各结点的网络IO量，解决网络问题的依据，增加cache，减少网络IO的传输。

query/node/ttfb:Time to first byte.更准确的反应出各个查询结点的响应时间，抛开数据传输的时间。

query/intervalChunk/time:查询每个时间块的时间。这个默认情况下是关闭的，也是没必要统计的字段。

query/success/count: 成功查询的个数。默认不开启

query/failed/count:失败查询的个数，默认不开启。

query/interrupted/count:被打断的查询个数。因为取消或者超时。

**Historical:**

query/time:一次查询花费的时间。在historical里面，没有太多的参考价值。可以配合慢查询进行优化查询SQL。

query/segment/time:查询某个segment花费的时间，包括query/wait/time。这个对于集群内部的优化，有很大的参考价值。

query/wait/time:一个segment开始被扫描的所等待的时间。此时间包括segment从磁盘加载到内存的时间，以及segment排队等待的时间。

segment/scan/pending:等待扫描的segment数。正常值是接近于0的，如果pending的数量比较多，就说明当前查询负载已经很高了，查询速度普遍下降，可以作为报警阀值。

query/cpu/time:一次查询话费的CPU时间，单位：微秒。跟query/time做比较，可以得出此次查询话费在CPU上的时间百分比。以此来参考判断整个集群是否瓶颈在CPU上。

**Real-time**：

real-time一般不是查询的瓶颈，监控意义不大，仅有少部分监控项。

query/time:查询时间

query/wait/time:查询等待的时间。

segment/scan/pending:等待被扫描的segment队列的长度。

**Jetty**

jetty/numOpenConnections: jetty的连接数。

**Cache:**

query/cache/delta/\* 截止到上一次记录的信息

query/cache/total/\* 整个的统计信息

hitRate:cache命中率，对于优化查询cache有帮助。

集群监控可以分析出：数据冷热分布（哪个时间段，哪些segment的查询比较热），集群整体的查询QPS以及各个节点的查询QPS。

哪些维度filter比较多，哪些维度groupBy比较多。（注：这部分数据在druid中无法获得，只能通过API服务等其他途径获得。）

### B.集群的优化

集群的优化可以分为两部分，一部分是摄入，一部分是查询。

#### 摄入

1.**摄入的动态扩容**。可以根据当前datasource的数据量，机器的负载等参数，实时动态的分配worker的数量。

2.**摄入的预聚合**。实时和离线两条数据流，是否可以再进行日志ETL的过程中，提前聚合一部分数据，以达到数据成倍的减少，降低 Druid实时计算节点的负载以及hadoop任务的执行时间。

3.**摄入的负载均衡**。根据业务中，一些任务的优先级，进行任务槽分割。比如实时任务的任务槽和离线任务的 任务槽分割。当批量跑离线任务的时候，任务槽不足，可以排队执行实时任务，但是不会占用实时的任务槽。

#### 查询

1.**数据结构的优化**。根据druid最新版本的特性，dimension支持数值和String类型两种。String优势在于filter，数值优势在于groupBy。合理利用业务的特点，定义数据结构，保证数据查询的高效。

**2.数据分表。**根据业务的特点，可以对每一类查询单独订制一张表，以此来保证数据高度聚合，加快查询速度。

3.**数据分层。**数据分层可以分为：按照时间线的分层，按照业务分层。

4.**慢查询优化。**实行慢查询报警策略，针对慢查询，根据Druid提供的监控信息，找出查询过程中具体耗时的地方，进行SQL上的优化。从头到尾绑定queryId。

5.**加入查询权限验证。**保护集群接口不被盗刷。

## 业务：

业务上的优化主要体现在跟Druid对接的API服务的优化。

1.**自动识别查询类型**。根据SQL选择Druid擅长的timeseries和TOPN查询，少用groupby查询。

2.**缓存的使用**。缓存上一次的查询结果，并统计命中率，进而不断优化缓存策略。

3.**屏蔽数据结构新增字段。**对于数据结构新增的字段，可以在API层统一配置，对业务进行屏蔽。

4.**增加多个数据源**。可以根据用户查询的SQL，自动适配数据所在的数据源，保证数据组对外API的统一性。(Druid,Mysql,Redis,Hive等)

5.**查询的负载均衡。**对Druid broker结点进行负载均衡。