openLooKeng多分片管理特性

by 陈平增

openLooKeng

https://openlookeng.io

Content

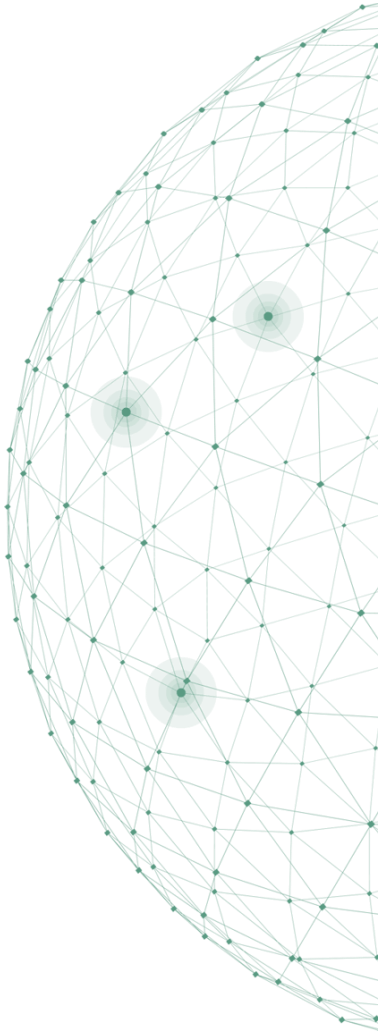
多分片特性背景介绍

多分片原理及交互流程

多分片配置使用

多分片特性的效果表现

结果分析



openLooKeng

https://openlookeng.io

多分片特性背景介绍

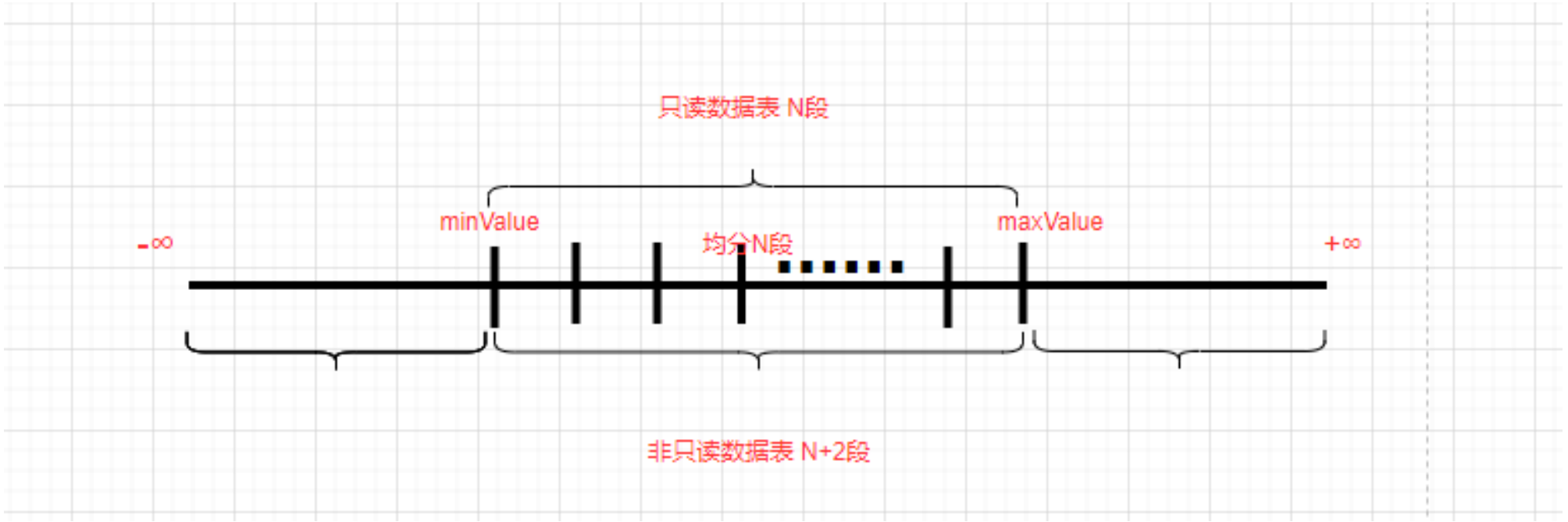
特性的价值：

数据源不具备多分片能力的情况下，扫表工作负载集中在单个worker上，无法充分利用集群优势，读取数据的效率不高。考虑在openLooKeng侧增加多分片处理能力，通过多分片并发访问以提高数据查询的处理的效率。

原理：

分片管理模块按照用户指定的列对目标表拆分为相应数量的分片，分片的分割以逻辑区段来划分。如果目标表在分割列上存在数据分布不均的情况，分片管理功能提供动态步长算法来均衡分片间的数据量。

多分片原理及交互流程(1)



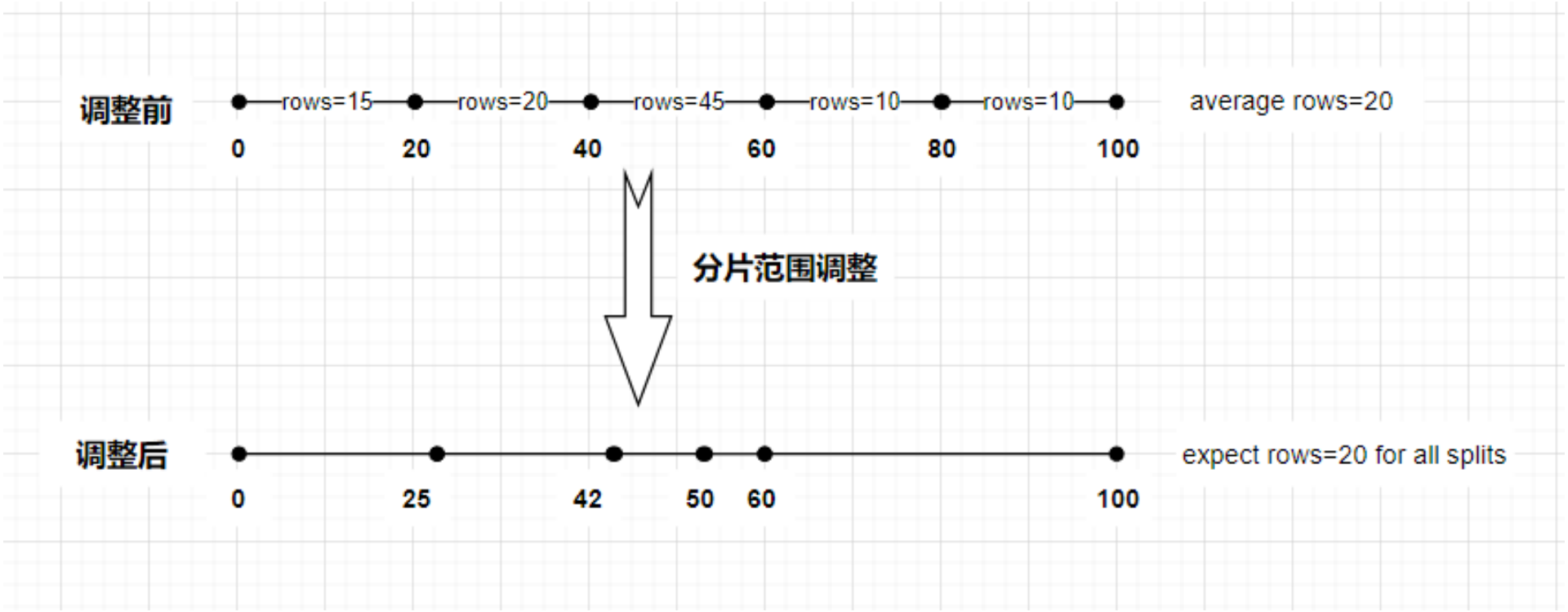
只读表按用户设置分割为N个分片

非只读表加上(-∞,minValue)和(maxValue,+ ∞)这两段，分为N+2个分片

openLooKeng

https://openlookeng.io

多分片原理及交互流程(2)-动态步长调整

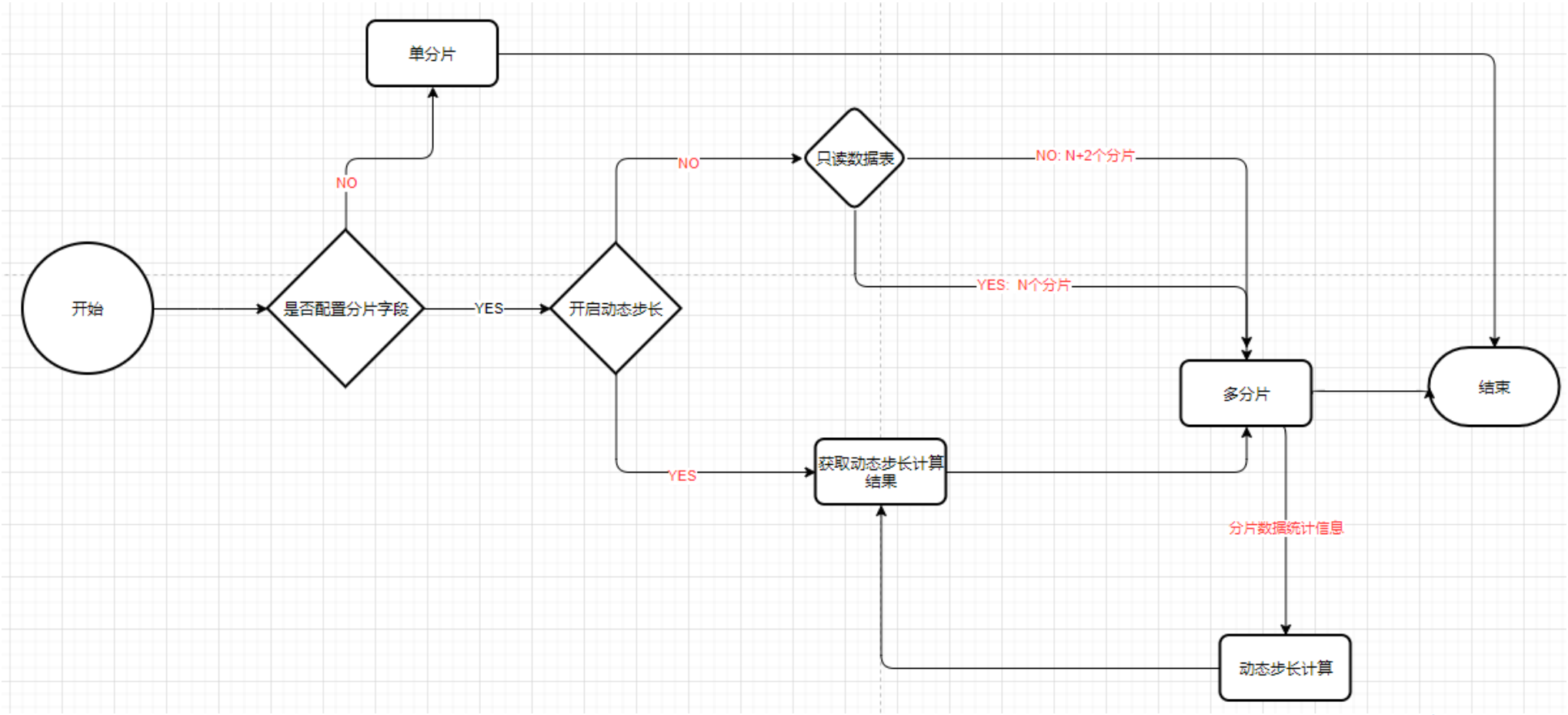


核心思想：按分片的取值范围进行排序，合并行数小于平均值的连续分片，按比例拆分行数大于平均值的分片，保持分片个数在调整前、后不变

openLooKeng

https://openlookeng.io

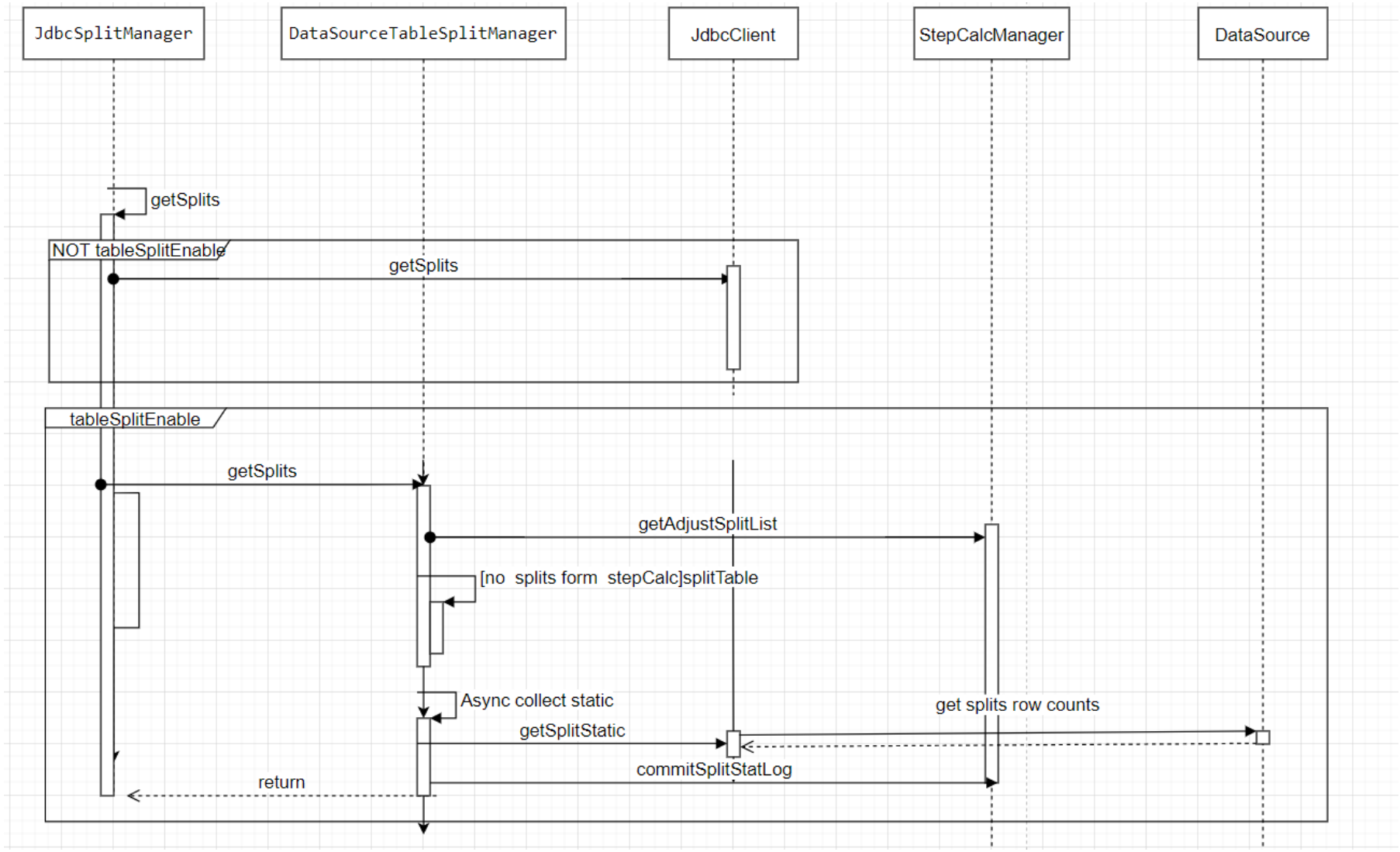
多分片原理及交互流程(3)-业务流程



openLooKeng

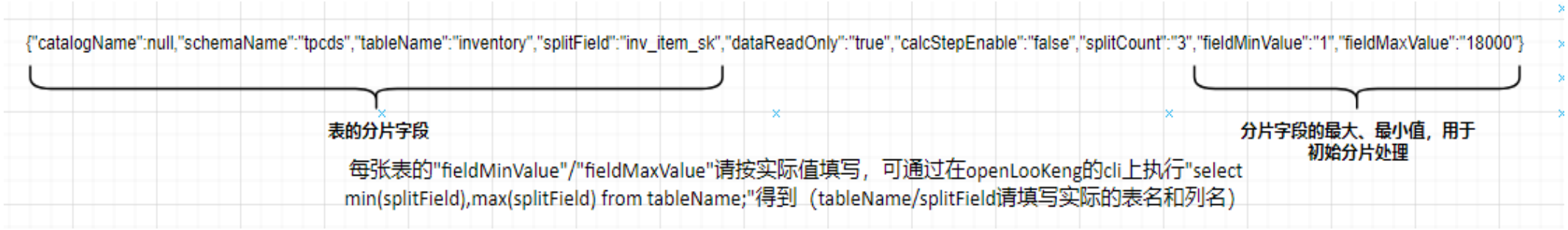
https://openlookeng.io

多分片原理及交互流程(4)



多分片配置使用

```
connector.name=opengauss
connection-url=jdbc:postgresql://76.75.67.18:25308/postgres?useSSL=false
connection-user=root
connection-password=123456
use-connection-pool=true
jdbc.table-split-enabled=true
jdbc.table-split-fields=[{"catalogName":null,"schemaName":"tpcds","tableName":"catalog_returns","split
Field":"cr_order_number","dataReadOnly":"true","calcStepEnable":"false","splitCount":"3","fieldMinValu
```



- 详细配置说明：https://openlookeng.io/zh-cn/docs/docs/admin/multi-split-for-jdbc-data-source.html

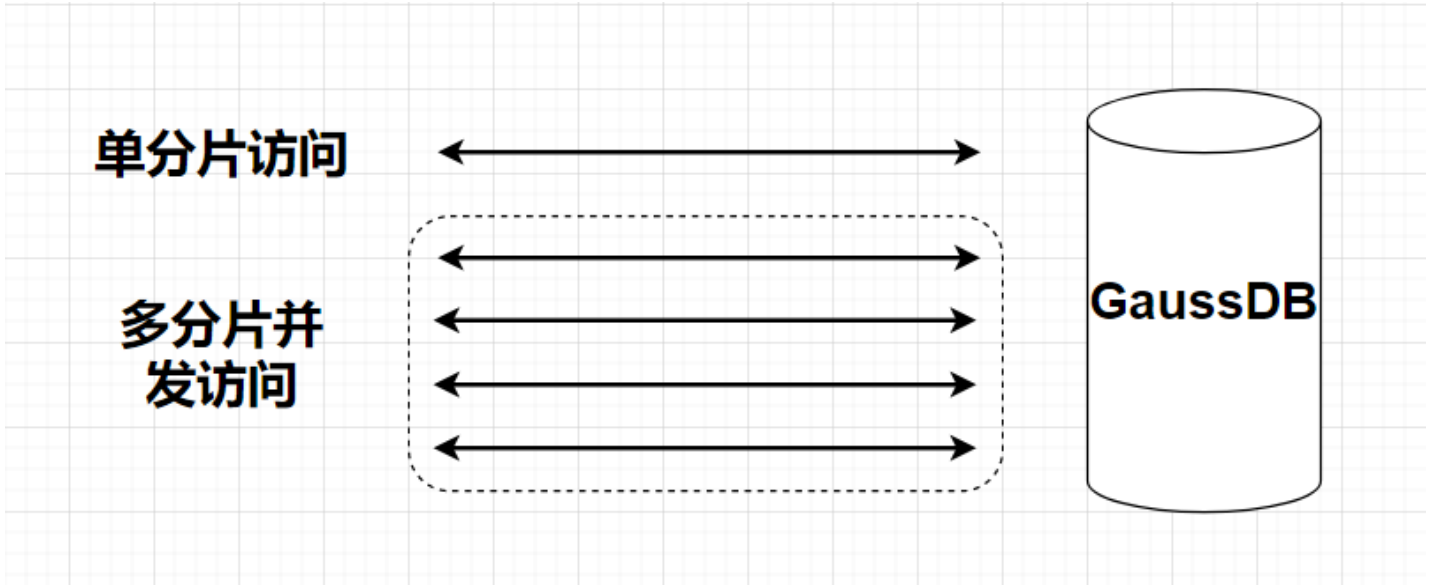
openLooKeng

https://openlookeng.io

多分片配置使用-注意事项及经验传递

- 本特性针对JDBC数据源做优化，如MySQL、PG、openGauss等；
- 需要用户选定表中的整形值的列，设置为分片列；
- 建议选取重复值少、分布均匀的列；
- 大规模结果集回显占据大量时间，避免选取此类语句进行对比测试；
- 对网络延时较大或者数据源jdbc连接延时较大的场景，推荐在connector配置”use-connection-pool=true”，性能对比效果更显著；

多分片特性的效果表现-单表测试

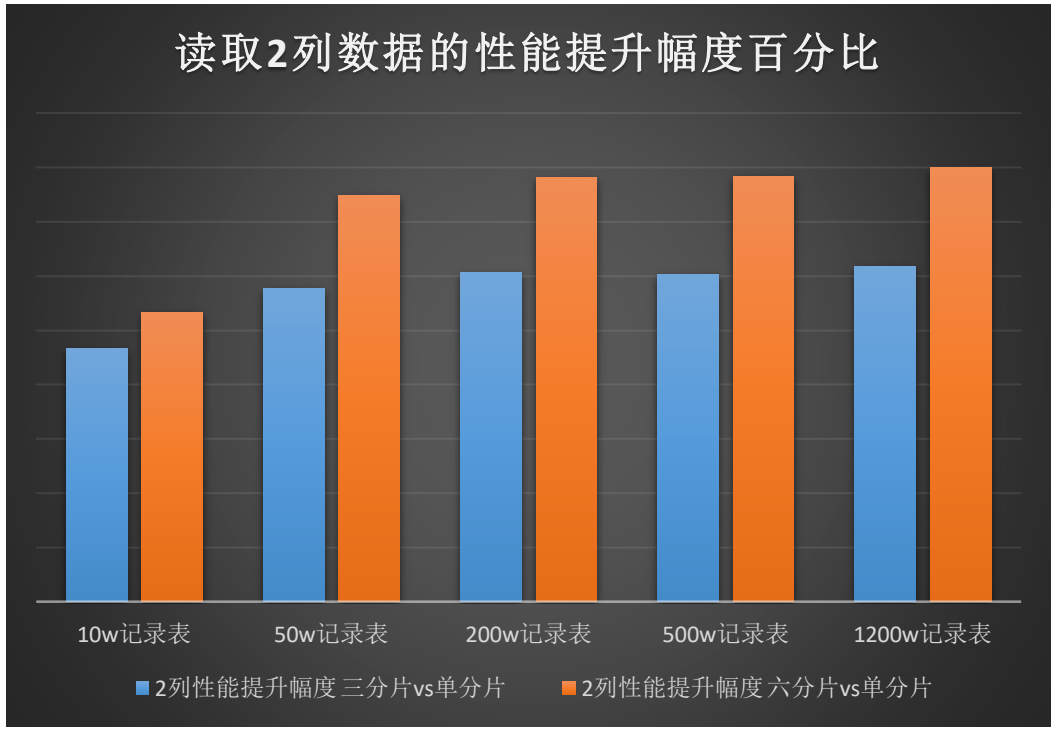


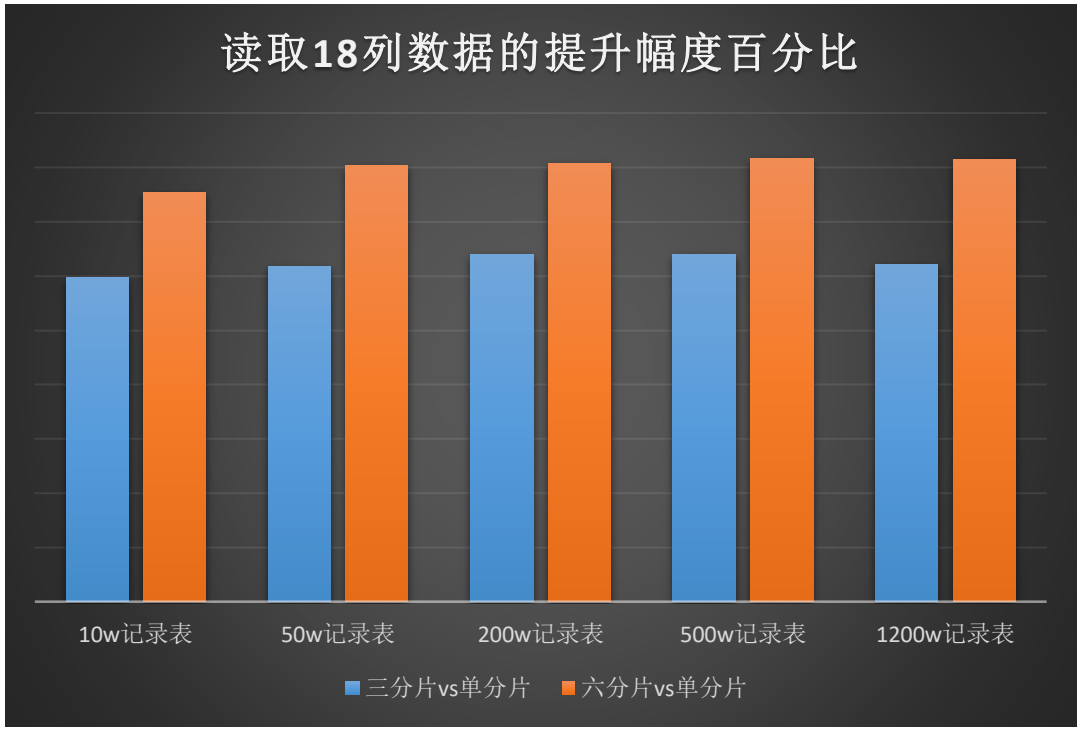
环境： 四节点（1CN + 3worker） openLooKeng 1.3.0
gaussDB 6.5.1单机
数据集：tpcds customer表，分别测试10W，50W，200W，500W，1200W五个不同规格的表在二个维度(投影列数，分片数)扫表的性能与单分片相比的提升幅度。
测试方法：分别设置gaussDB连接器使用单分片/多分片方式启动openLooKeng，从JMeter下发查询语句。所有查询语句的时延均采用端到端时延（从JMeter获取），8轮测试取平均值。

```
select * from customer100000 where c_customer_sk in (select c_customer_sk from customer10)
select * from customer500000 where c_customer_sk in (select c_customer_sk from customer10)
select * from customer2000000 where c_customer_sk in (select c_customer_sk from customer10)
select * from customer5000000 where c_customer_sk in (select c_customer_sk from customer10)
select * from customer12000000 where c_customer_sk in (select c_customer_sk from customer10)
```

openLooKeng
https://openlookeng.io

多分片特性的效果表现

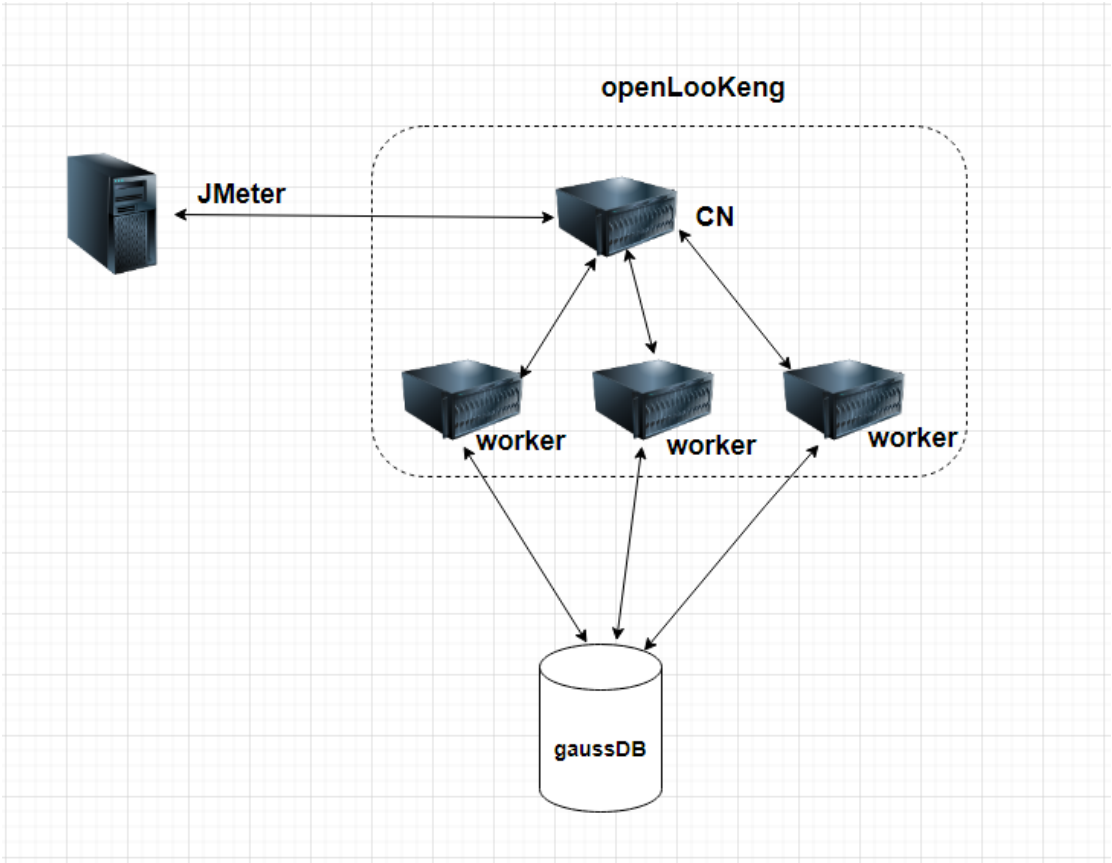




环境： 四节点（1CN + 3worker） openLooKeng 1.3.0
gaussDB 6.5.1单机
数据集：tpcds customer表，分别测试10W，50W，200W，500W，1200W五个不同规格的表在二个维度(投影列数，分片数)扫表的性能与单分片相比的提升幅度。
测试方法：分别设置gaussDB连接器使用单分片/多分片方式启动openLooKeng，从JMeter下发查询语句。所有查询语句的时延均采用端到端时延（从JMeter获取），8轮测试取平均值。

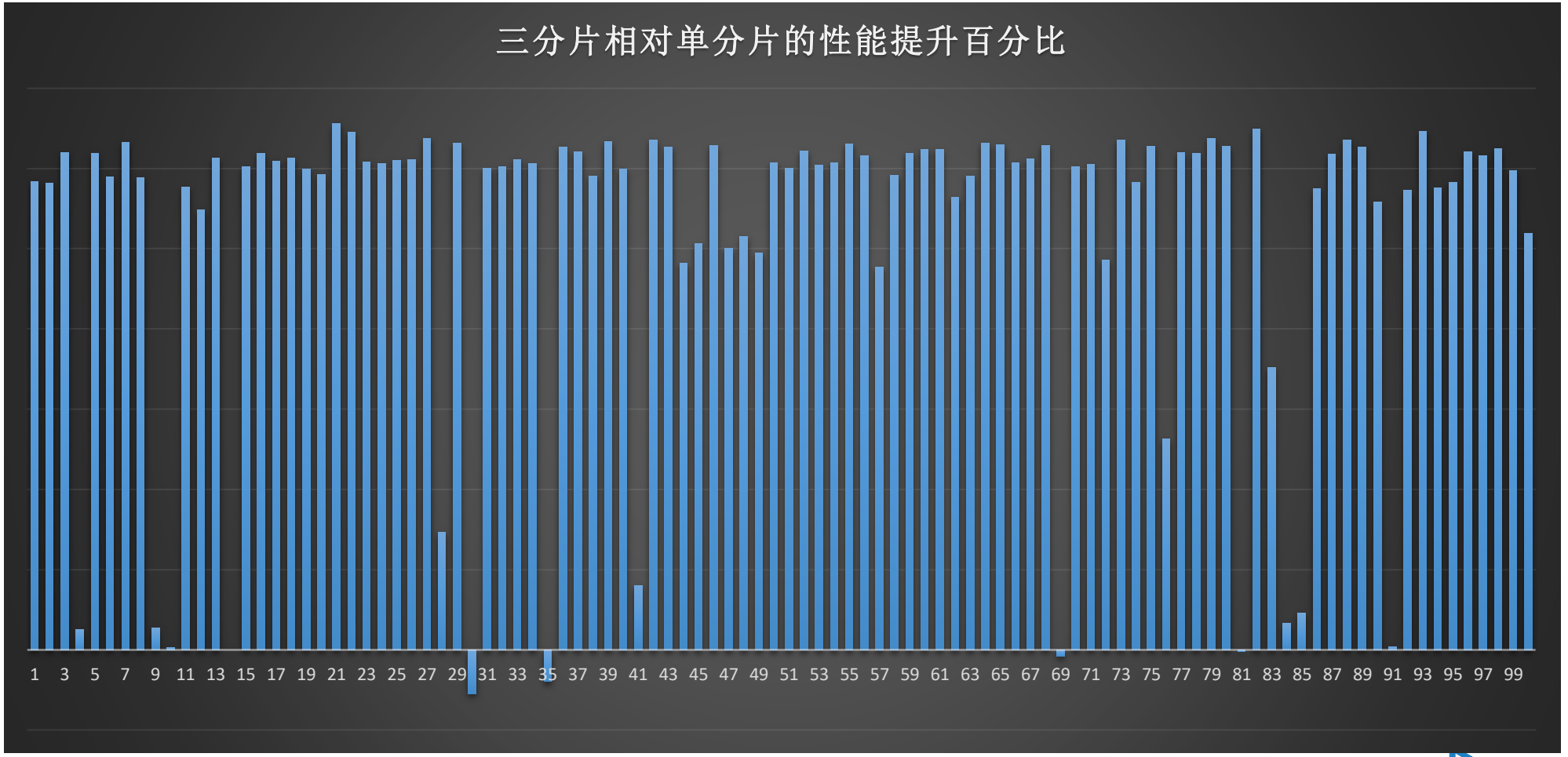
openLooKeng
https://openlookeng.io

多分片特性的效果表现-部署图



openLooKeng环境：1CN + 3worker 每台服务器 CPU：2* Intel(R) Xeon(R) CPU E5-2667 v4 @ 3.20GHz 内存：768GB
GaussDB环境：单节点 CPU：2*Intel(R) Xeon(R) Gold 6154 CPU @ 3.00GHz 内存：256GB
数据集：1GB gaussDB tpc-ds数据集
测试方法：在openLooKeng侧对TPC-DS的7张事实表，配置为3分片并发读取；gaussDB采用默认配置。采用TPC-DS 99语句，设置JMeter每条语句循环执行10遍。

多分片特性的效果表现-TPC-DS SQL99测试



openLooKeng

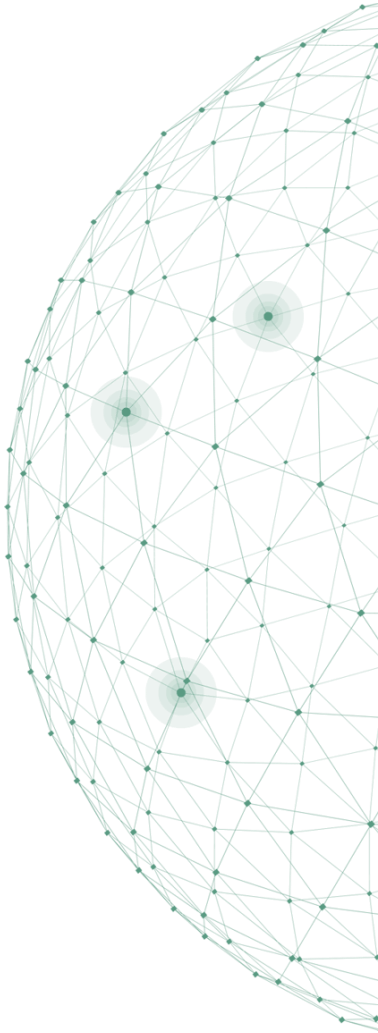
https://openlookeng.io

结果分析

TPC-DS数据集效果好的原因：

- 99语句语法复杂，大部分事实表都是扫描表操作，叠加分片效果， 扫表性能提升明显；
- 由于扫表采用多分片，上层算子由单分片处理转3并发处理，算子的并发度提高了，table-scan以上的算子效率也提升明显；
- TPC-DS 99语句输出结果比较少（很多采用limit100进行了缩减），结果集的输出对端到端的影响较小；

Thank you!



openLooKeng

https://openlookeng.io