1. Neptune client工程概述

本工程是基于java的maven工程，集成了Neptune客户端所需要的类库。

neptune集群的版本为1.2.0.2及以上版本，Java的版本为openjdk 17，默认是部署在neptune server所在的vpc内来运行，权限设置参照官网，

采用Springboot框架，对象都采用依赖注入，写了单元测试代码。

mvn package -DskipTests #不跑单元测试

mvn package #跑单元测试【测试压力写入，慎用】

nohup java -jar neptunedemo-1.0-SNAPSHOT.jar &

配置项是application.yml：

server:  
 port: 8081   
  
aws:  
 region: eu-west-1  
 clusterId: database-1  
 intervalSeconds: 15

server.port 是本机的web端口。

aws.region是neptune服务部署的区域。

aws.clusterId是neptune服务的cluster id。

aws.intervalSeconds是neptune服务的配置信息的拉取间隔。

权限说明：

SDK上，我们使用DefaultAWSCredentialsProviderChain, 它会自动按优先级匹配权限设置，依次是：环境变量（AKSK）,Java系统变量（AKSK），Web Identity Token，credential profile(AKSK)等。SDK说明：

<https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/com/amazonaws/auth/DefaultAWSCredentialsProviderChain.html>

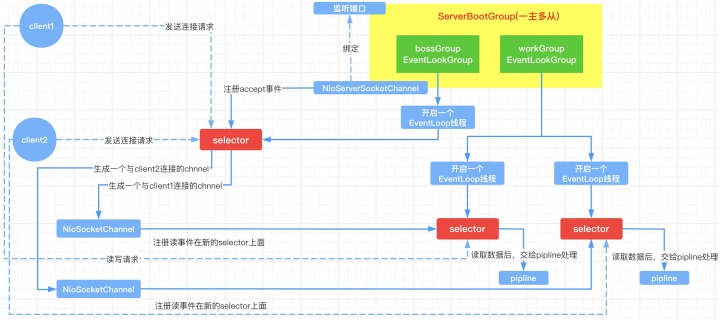
简单来说，SDK就是有AKSK就优先匹配，没有的话再去匹配机器上的IAM Role。

该工程重组了以下的官方样例代码，并参照Neptune server架构和官方的最佳实践说明，做了优化。

<https://github.com/awslabs/amazon-neptune-tools/tree/master/neptune-gremlin-client>

1. Neptune server架构概述
2. neptune是基于netty实现的网络架构

参考netty的NIO网络模型



客户WebSocket连接由服务器的bossgroup线程池接收，连接数限制：

<https://docs.aws.amazon.com/neptune/latest/userguide/limits.html>

特别注意：

如果客户端没有关闭连接，则连接可能会在 20 到 25 分钟的空闲超时后自动关闭；

启用 IAM 身份验证时，如果 WebSocket 连接在建立 10 天后仍未关闭，则连接始终会在建立 10 天后的几分钟内断开连接；

服务器的workgroup线程池 实际处理图数据库的查询和更新语句，连接数默认为：VCPU\*2。

2）集群架构

一主多从；主支持写入，主从都支持读取；支持failover。

所以，程序里的neptune的read Client和write Client都采用了单例；添加了定时查询激活、定时断开重连这两个机制。

1. 最佳实践

参考：<https://docs.aws.amazon.com/zh_cn/neptune/latest/userguide/best-practices.html>

1）客户端的库要使用最新版本

2) 在多个线程之间重用同一个Client或 GraphTraversalSource对象

3) 为读取和写入端点创建单独的 Gremlin Java 客户端对象

4) 优雅关闭 Cluster.close( )，内部调用了client.close()

5）将maxInProcessPerConnection和设置maxSimultaneousUsagePerConnection为相同的值,例如32（最大并发度）

6）不要使用client来提交，而去使用GraphTraversalSource来提交；使用 limit(X) 步骤来限制服务器生成的结果数。

7）插入大量数据时，批量添加顶点和边

8）负载平衡方案，采用拉取模式

【程序里连接的是Neptune endpoint地址，且有refresh agent负责做配置的拉取和更新】

9）设置个别查询的超时

g.with(ARGS\_EVAL\_TIMEOUT, 500L)

10)连接调优

maxParallelQueries = maxConnectionPoolSize \* Max( maxSimultaneousUsagePerConnection, maxInProcessPerConnection )

延长连接超时时间（延迟不重要的查询）；增加每个连接的最大请求数；增加连接池中连接数；

11) 故障转移后的新连接

【程序里有设置定时重连】

1. 批量写入的单元测试说明

服务器配置：r6g, 4核32G,EBS带宽是4750Mbps

客户端配置：c6i.xlarge, 4核8G

测试数据：

10个工作线程，1个写入client；

10万次循环，每次生成100个点和50条边；

总共生成1000万个节点，500万条边

测试结果：

1. 日志

单次执行时间：300-500ms；

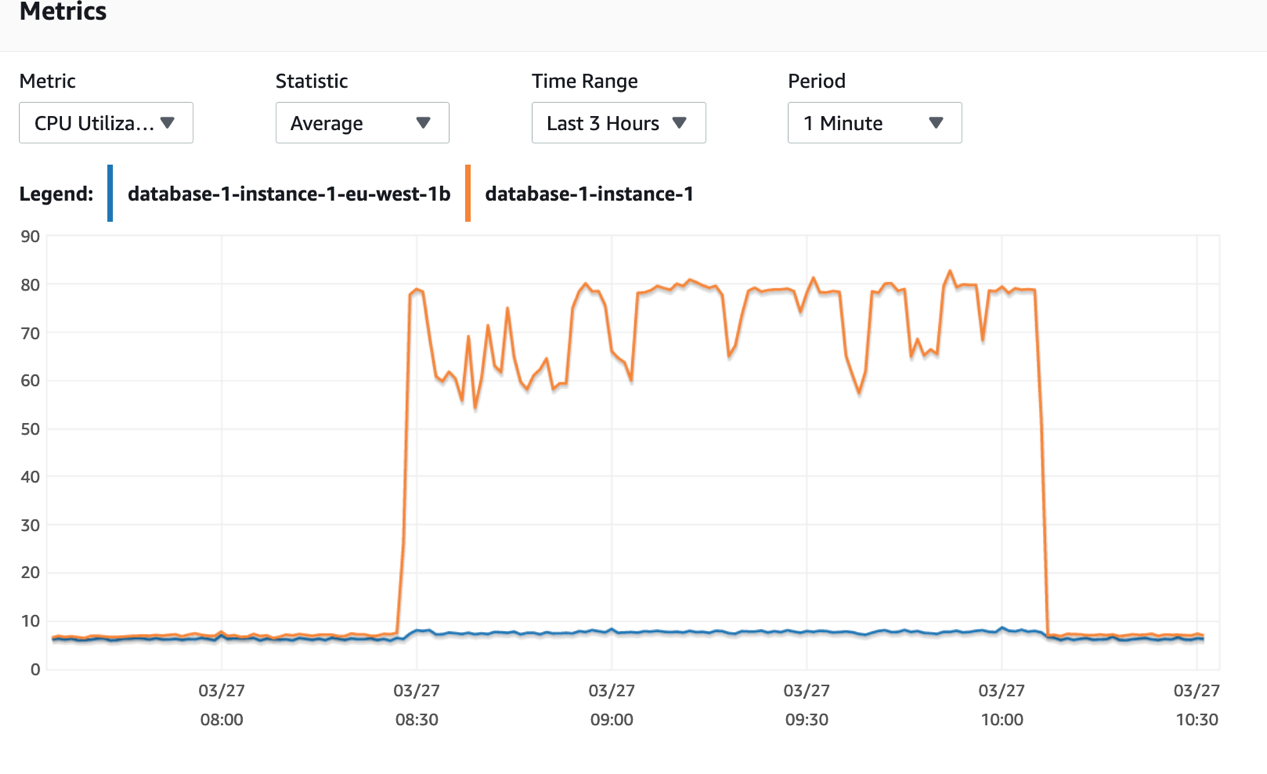
总执行时间：(1679882780542-1679876904274)/100=5876.268秒，约为1.63小时

错误率：0

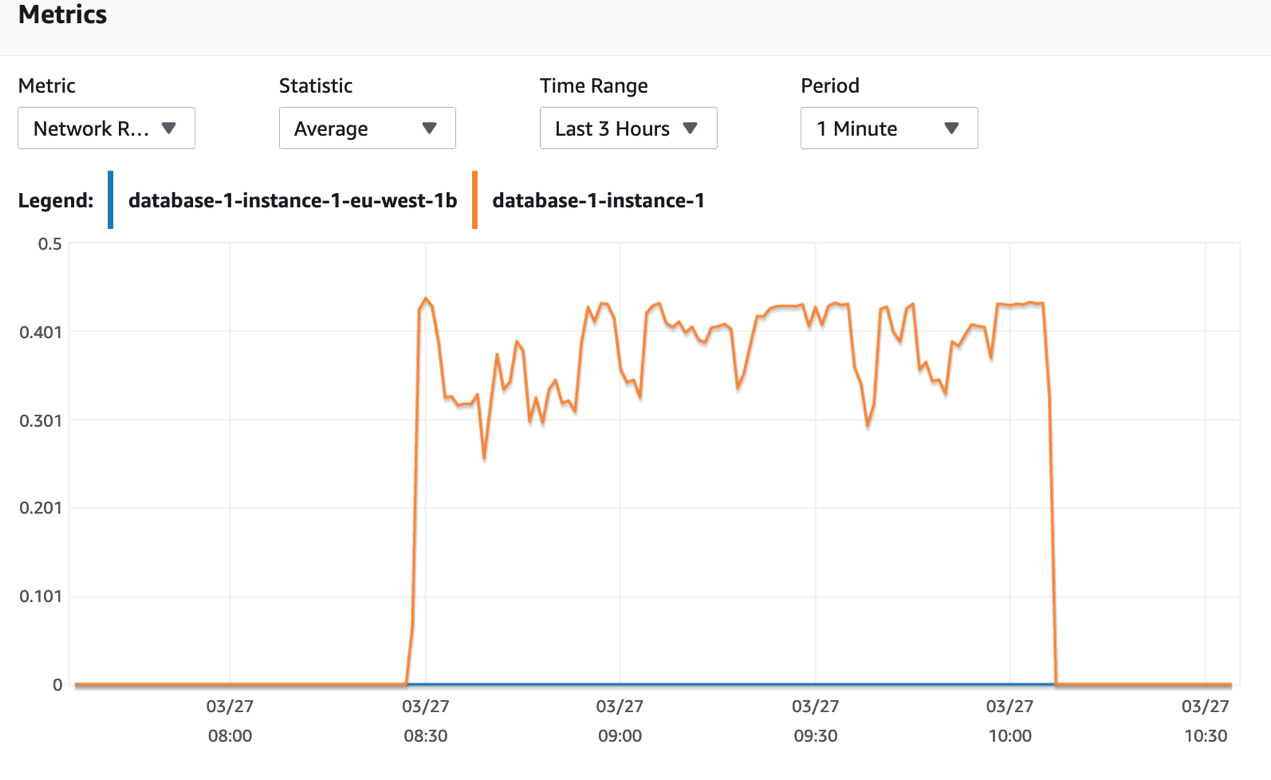
02:06:20.542 [pool-19-thread-6] INFO com.amazonaws.demo.service.DataLoadWorker - thread pool-19-thread-6 ends processing, batchNo:99999, time is 1679882780542, total is 327, errorCount: 0

2）服务器资源监控

CPU负载



网络流量负载



1. Web查询接口

说明：以下接口是为了验证gremlin的语法正确，并且能正常操作neptune的服务器资源。可以根据自己需要做调整和优化。

1. 查询

/graph/query

1. 删除

/graph/delete

1. 增加

/graph/add

1. 批量增加

/graph/batchInsert