向量数据库性能测试教程

概述

本教程介绍如何使用提供的C++代码对向量数据库进行QPS和召回率性能测试。测试基于SIFT数据集，支持IVF索引的多线程性能评估。

环境准备

参考向量数据加速卡安装指南中的软件部署，docker部署方案，启动docker客户端。

```
cd docker
sudo docker compose run --rm client

# 进入性能测试目录
cd /root/hilbert/c++

# 执行qps recall测试
./bazel-bin/test_qps_recall config.ini
```

配置文件说明

config.ini 参数详解

```
[hdf5]
file = /mnt/soft/data_set/SIFT_1M.hdf5  # SIFT数据集路径

[search]
nq = 10000              # 查询向量数量
topk = 10               # 返回最相似的K个结果
thread_num = 32         # 并发线程数
index_name = sift_1M_s10  # 索引名称前缀
nlist = 512             # IVF聚类中心数量
nprob = 16              # 搜索时探测的聚类数量
batch_size = 200        # 批处理大小
thread_repeat = 10      # 每线程重复次数
need_ivf_recall = false # 是否计算IVF召回率对比
```

关键性能参数

参数	作用	性能影响
`thread_num`	并发线程数	影响QPS，过高可能导致资源竞争
`batch_size`	批处理大小	影响吞吐量和延迟平衡
`nlist`	聚类中心数	影响索引质量和搜索速度
`nprob`	探测聚类数	召回率与速度的权衡

快速开始

1. 基础性能测试

```
# 使用默认配置运行
./bazel-bin/test_qps_recall config.ini
```

2. 修改配置进行对比测试

> **注意，以下是E5服务器上的典型配置，其他配置的服务器需要针对性的调优**

高QPS配置：

```
nlist = 4096
nprob = 48
thread_num = 128
batch_size = 500
```

高召回率配置：

```
nlist = 4096
nprob = 64
thread_num = 128
batch_size = 100
```

性能调优指南

线程数优化

```
# 测试不同线程数的影响
for threads in 8 16 32 64; do
    sed -i "s/thread_num = .*/thread_num = $threads/" config.ini
```

```
    echo "Testing with $threads threads:"
    ./test_qps_recall config.ini | grep QPS
done
```

批处理大小优化

```
# 测试不同批处理大小
for batch in 50 100 200 500 1000; do
    sed -i "s/batch_size = .*/batch_size = $batch/" config.ini
    echo "Testing batch_size=$batch:"
    ./test_qps_recall config.ini | grep QPS
done
```

nprob参数扫描

```
# 在配置文件中设置多个nprob值进行对比
nprob = 1,2,8,16,24,32,48,64
```

输出结果解析

典型输出格式

```
[NlistPerf-MT] nlist=512 batch_size=200 nprob=16 threads=32 QPS=15420.5
Recall=0.892 IVF Recall=0.945 Latency(ms)=1.24
```

指标含义

- **QPS**: 每秒查询数量，衡量系统吞吐能力
- **Recall**: 召回率，与ground truth的匹配度
- **IVF Recall**: 与Faiss IVF实现的召回率对比
- **Latency**: 平均延迟（毫秒）

性能基准对比

推荐测试场景

场景	nlist	nprob	期望QPS	期望Recall
高性能搜索	4096	48	>40000	>=0.95

高精度搜索	4096	128	>7000	>=0.99

性能瓶颈分析

CPU瓶颈识别：

```
# 运行测试时监控CPU使用率
top -p $(pgrep test_qps_recall)
```

内存使用监控：

```
# 监控内存占用
ps aux | grep test_qps_recall
```

常见问题排查

1. QPS过低

- 检查线程数是否合理（建议为CPU核心数的1-2倍）
- 调整batch_size增加批处理效率
- 降低nprob值减少计算量

2. 召回率不理想

- 增加nprob值
- 调整nlist数量
- 检查数据质量和分布

3. 延迟过高

- 减少batch_size
- 降低nprob
- 检查网络连接（如果是分布式部署）

多索引对比测试

```
# 配置文件支持多个nlist值
nlist = 512,1024,2048,4096
```

最佳实践

1. **渐进式调优**：从默认参数开始，逐步调整单个参数
2. **多轮测试**：每个配置运行多次取平均值
3. **资源监控**：关注CPU、内存、网络使用情况
4. **记录基准**：建立性能基线用于回归测试