#!/bin/bash

set -e

sudo apt-get update

echo "Y" | sudo apt-get upgrade

git clone https://github.com/microsoft/DiskANN.git --branch 0.6.0

cd DiskANN/

sudo apt install make cmake g++ libaio-dev libgoogle-perftools-dev clang-format libboost-all-dev

echo "Y"

wget https://registrationcenter-download.intel.com/akdlm/irc\_nas/18487/l\_BaseKit\_p\_2022.1.2.146.sh

sudo sh l\_BaseKit\_p\_2022.1.2.146.sh -a --components intel.oneapi.lin.mkl.devel --action install --eula accept -s

echo "Y"

mkdir build && cd build && cmake -DCMAKE\_BUILD\_TYPE=Release .. && make -j

mkdir -p data && cd data/

wget ftp://ftp.irisa.fr/local/texmex/corpus/sift.tar.gz

tar -xf sift.tar.gz

cd ..

./apps/utils/fvecs\_to\_bin float data/sift/sift\_learn.fvecs data/sift/sift\_learn.fbin

./apps/utils/fvecs\_to\_bin float data/sift/sift\_query.fvecs data/sift/sift\_query.fbin

./apps/utils/compute\_groundtruth --data\_type float --dist\_fn l2 --base\_file data/sift/sift\_learn.fbin --query\_file data/sift/sift\_query.fbin --gt\_file data/sift/sift\_query\_learn\_gt100 --K 100

./apps/build\_disk\_index --data\_type float --dist\_fn l2 --data\_path data/sift/sift\_learn.fbin --index\_path\_prefix data/sift/disk\_index\_sift\_learn\_R32\_L50\_A1.2 -R 32 -L50 -B 0.003 -M 1

./apps/search\_disk\_index --data\_type float --dist\_fn l2 --index\_path\_prefix data/sift/disk\_index\_sift\_learn\_R32\_L50\_A1.2 --query\_file data/sift/sift\_query.fbin --gt\_file data/sift/sift\_query\_learn\_gt100 -K 10 -L 10 20 30 40 50 100 --result\_path data/sift/res --num\_nodes\_to\_cache 10000 -T 8

sudo apt install libcpprest-dev

cmake -DRESTAPI=True -DCMAKE\_BUILD\_TYPE=Release ..

make -j

./apps/restapi/ssd\_server --address <http://localhost:3001/> --data\_type float --index\_path\_prefix data/sift/disk\_index\_sift\_learn\_R32\_L50\_A1.2 --num\_nodes\_to\_cache 10000 --num\_threads 8

client localhost:3000 data\_type float query\_file num\_queries Ls"

cd DiskANN/apps/utils/

vim compute\_groundtruth.cpp

cd

cd DiskANN/build/

INSERT INTO mysql.user (Host, User, Password) VALUES ('%', 'root', password('Aremaniak1\_'));

GRANT ALL ON \*.\* TO 'root'@'%' WITH GRANT OPTION;

import requests

jsonquery = {"Ls": 256,

"query\_id": 1234,

"query": [0.00407, 0.01534, 0.02498, ...],

"k": 10}

response = requests.post('http://ip\_addr:port', json=jsonquery)

print(response.text)

import requests

import time

# Fungsi untuk mengukur latency

def measure\_latency():

jsonquery = {

"Ls": 256,

"query\_id": 1234,

"query": [0.00407, 0.01534, 0.02498],

"k": 10

}

start\_time = time.time() # Catat waktu mulai

response = requests.post('http://localhost:3000', json=jsonquery)

end\_time = time.time() # Catat waktu selesai

latency = end\_time - start\_time

return latency

# Fungsi untuk mengukur throughput

def measure\_throughput(num\_requests, duration):

jsonquery = {

"Ls": 256,

"query\_id": 1234,

"query": [0.00407, 0.01534, 0.02498],

"k": 10

}

start\_time = time.time() # Catat waktu mulai

for \_ in range(num\_requests):

requests.post('http://localhost:3000', json=jsonquery)

end\_time = time.time() # Catat waktu selesai

elapsed\_time = end\_time - start\_time

throughput = num\_requests / elapsed\_time

return throughput

# Mengukur dan mencetak latency

latency = measure\_latency()

print(f"Latency: {latency:.6f} seconds")

# Mengukur dan mencetak throughput (misalnya, 50 permintaan dalam 10 detik)

num\_requests = 50

duration = 10

throughput = measure\_throughput(num\_requests, duration)

print(f"Throughput: {throughput:.2f} queries per second")

import requests

import time

import threading

# Fungsi untuk mengukur latency

def measure\_latency(num\_requests):

jsonquery = {

"Ls": 256,

"query\_id": 1234,

"query": [0.00407, 0.01534, 0.02498],

"k": 10

}

latencies = [] # List untuk menyimpan waktu latency dari setiap permintaan

def send\_requests():

for \_ in range(num\_requests):

start\_time = time.time() # Catat waktu mulai

response = requests.post('http://localhost:3000', json=jsonquery)

end\_time = time.time() # Catat waktu selesai

latency = end\_time - start\_time

latencies.append(latency)

threads = []

for \_ in range(num\_threads):

thread = threading.Thread(target=send\_requests)

threads.append(thread)

thread.start()

for thread in threads:

thread.join()

# Hitung rata-rata latency dari semua permintaan

average\_latency = sum(latencies) / len(latencies)

return average\_latency

# Mengukur dan mencetak latency untuk 50 permintaan dalam 4 thread

num\_requests\_per\_thread = 50

num\_threads = 4

total\_requests = num\_requests\_per\_thread \* num\_threads

latency = measure\_latency(total\_requests)

print(f"Latency: {latency:.6f} seconds")