ECE 6130 Big Data and Cloud Computing Spring 2019

Homework #4 Report

Name: Tianyu Yang

GW ID: G38878678

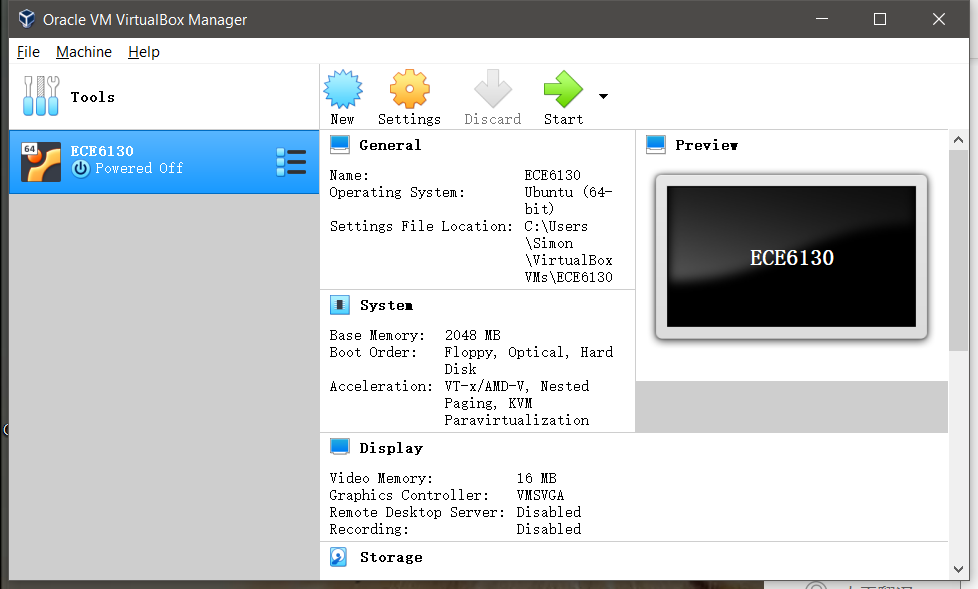
Date: 3/19/2019

Objective:

1. This homework is to use both virtual machine and physical machine to run BFS code
2. Record the configuration of virtual machine and physical machine.
3. Record the running time for both machine

Procedure 1: Computer configuration

1. In this project, I use Oracle VM VirtualBox Manager as the virtual machine. The configuration is shown as below:



The Operation System is ubuntu (64-bit);

The Base Memory is 2048 MB;

The Video Memory is 16 MB with the Graphics Controller VMSVGA;

The Storage is Normal 20.00GB;

1. To settle the physical machine, I use USB to boot my computer in a Linux system. The reason is that I use Linux system as OS for my virtual machine. As for comparation with the same OS, I also need to use Linux as OS for my physical machine. However, I do not own a physical computer with Linux system. Therefore, I use USB to store the Ubuntu system as boot my computer in the USB. Therefore, I can use any of the computer as physical machine with Ubuntu OS. The configuration of the computer is shown as below:



The CPU is Inter(R) Core (TM) i7-7700HQ with frequency 2.80GHz;

The Memory is SK Hynix 16.0 GB;

The GPU is NVIDIA GeForce GTX 1060 6GB;

The Hard Disk is 1TB with HDD and 256GB SDD;

Procedure 2: BFS code review

The BFS code used to run on the machine is the simple code on Graph 500. <http://graph500.org/?page_id=47>. The related code for the algorithm of BFS is in bfs\_reference.c

void make\_graph\_data\_structure(const tuple\_graph\* const tg) {

int i,j,k;

convert\_graph\_to\_oned\_csr(tg, &g);

column=g.column;

rowstarts=g.rowstarts;

visited\_size = (g.nlocalverts + ulong\_bits - 1) / ulong\_bits;

aml\_register\_handler(visithndl,1);

q1 = xmalloc(g.nlocalverts\*sizeof(int)); //100% of vertexes

q2 = xmalloc(g.nlocalverts\*sizeof(int));

for(i=0;i<g.nlocalverts;i++) q1[i]=0,q2[i]=0; //touch memory

visited = xmalloc(visited\_size\*sizeof(unsigned long));

}

void run\_bfs(int64\_t root, int64\_t\* pred) {

int64\_t nvisited;

long sum;

unsigned int i,j,k,lvl=1;

pred\_glob=pred;

aml\_register\_handler(visithndl,1);

CLEAN\_VISITED();

qc=0; sum=1; q2c=0;

nvisited=1;

if(VERTEX\_OWNER(root) == rank) {

pred[VERTEX\_LOCAL(root)]=root;

SET\_VISITED(root);

q1[0]=VERTEX\_LOCAL(root);

qc=1;

}

// While there are vertices in current level

while(sum) {

#ifdef DEBUGSTATS

double t0=aml\_time();

nbytes\_sent=0; nbytes\_rcvd=0;

#endif

//for all vertices in current level send visit AMs to all neighbours

for(i=0;i<qc;i++)

for(j=rowstarts[q1[i]];j<rowstarts[q1[i]+1];j++)

send\_visit(COLUMN(j),q1[i]);

aml\_barrier();

qc=q2c;int \*tmp=q1;q1=q2;q2=tmp;

sum=qc;

aml\_long\_allsum(&sum);

nvisited+=sum;

q2c=0;

#ifdef DEBUGSTATS

aml\_long\_allsum(&nbytes\_sent);

t0-=aml\_time();

if(!my\_pe()) printf (" --lvl%d : %lld(%lld,%3.2f) visited in %5.2fs, network aggr %5.2fGb/s\n",lvl++,sum,nvisited,((double)nvisited/(double)g.notisolated)\*100.0,-t0,-(double)nbytes\_sent\*8.0/(1.e9\*t0));

#endif

}

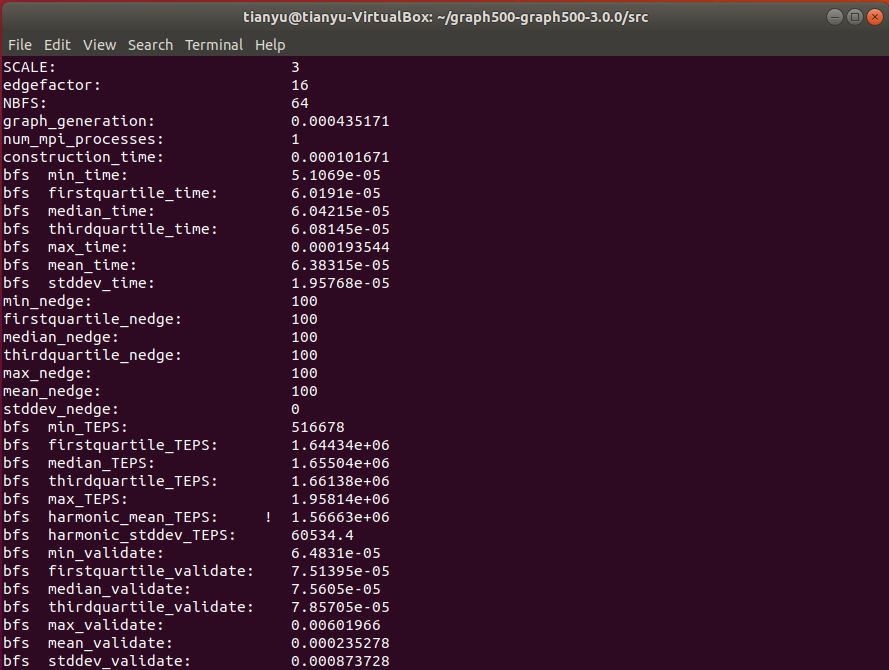
aml\_barrier();

}

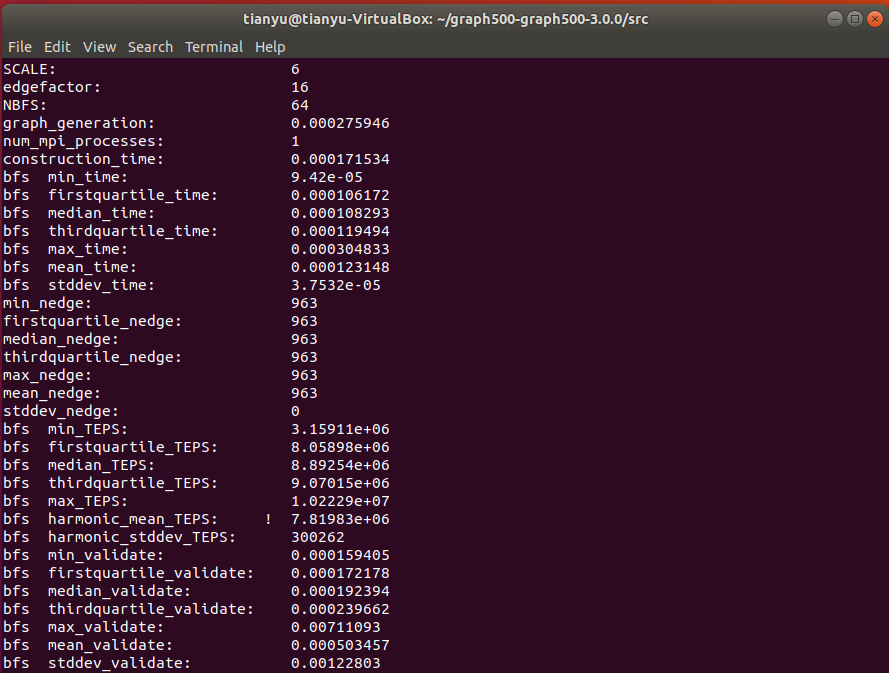
Procedure 3: Running time

**The running time for the virtual machine**

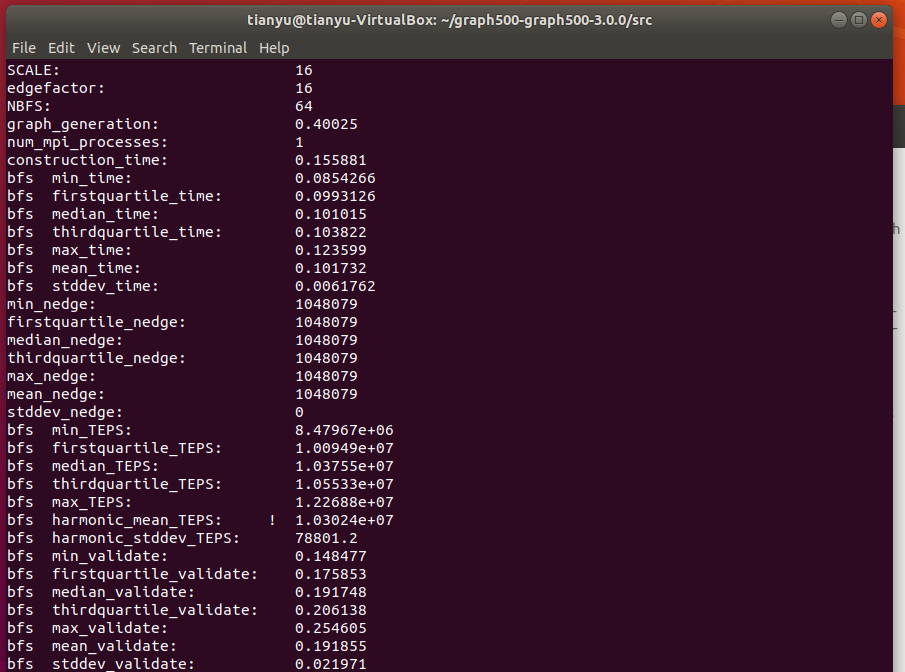
When the scale = 3 and the edge value is 100:



When the scale = 6 and the edge value is 1K:

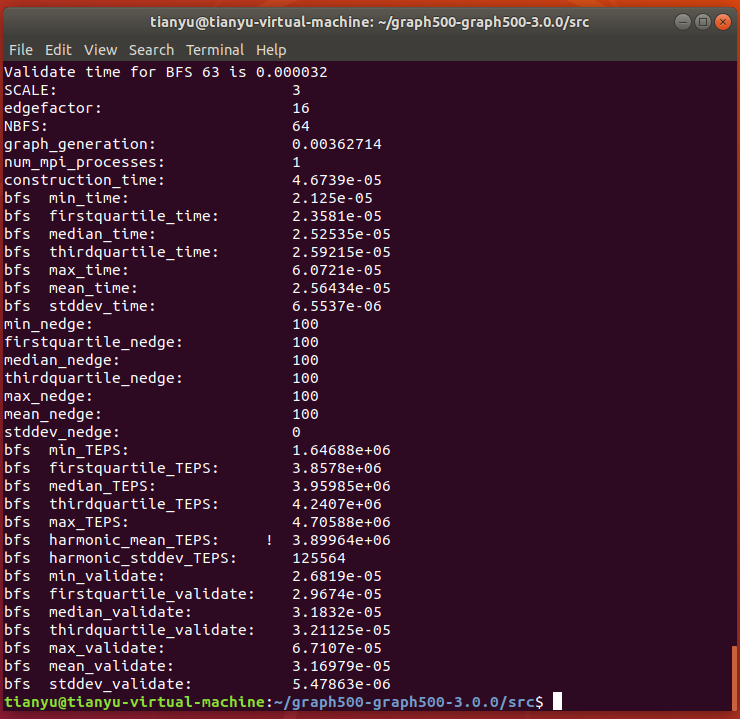


When the scale = 16 and the edge value is 1M:

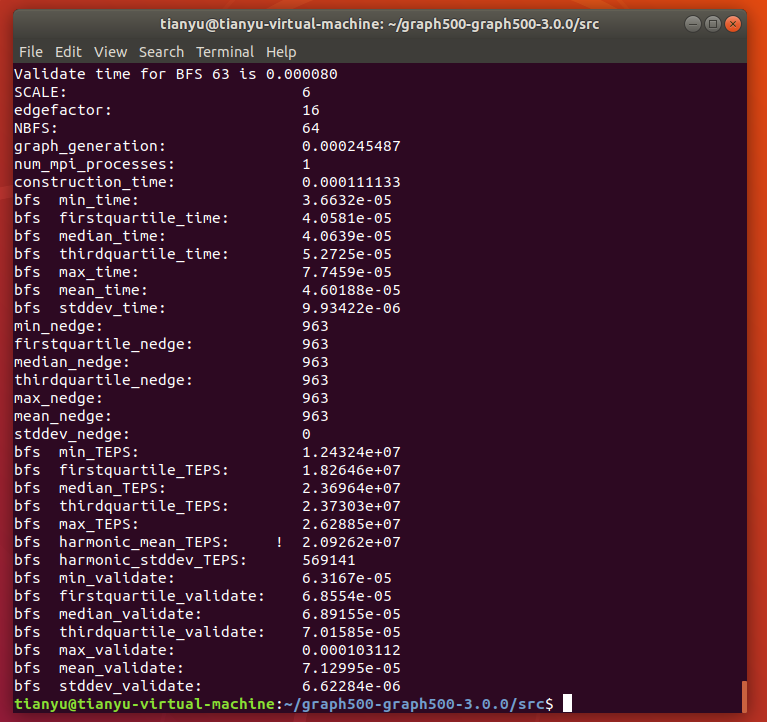


**The running time for the physical machine**

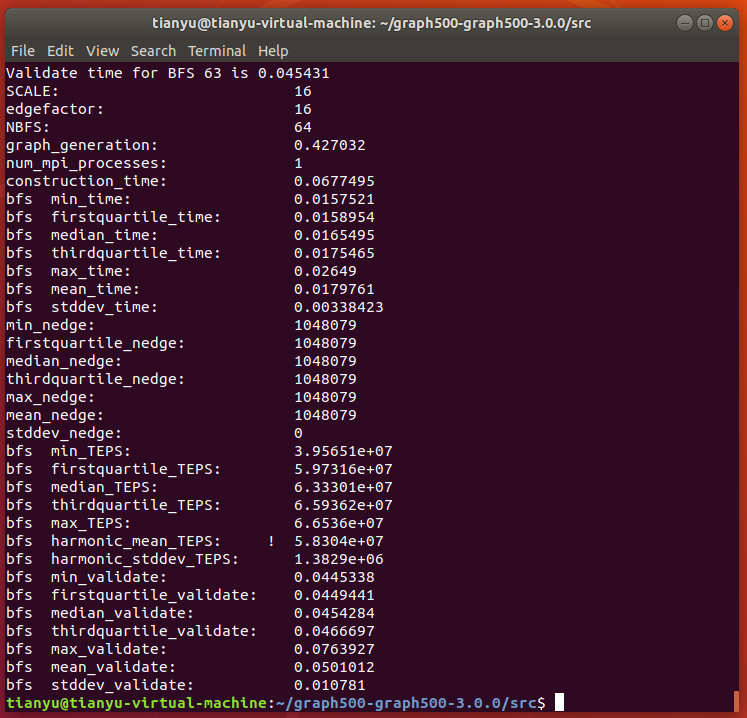
When the scale = 3 and the edge value is 100:



When the scale = 6 and the edge value is 1K:



When the scale = 16 and the edge value is 1M:



Procedure 4: Result analysis

To compare with the result of the running time in virtual machine and physical, I make a table below for the comparation.

|  |  |  |  |
| --- | --- | --- | --- |
| **Edge=100** | **graph\_\_generation** | **construction\_time** | **bfs mean\_time** |
| **Virtual Machine** | 0.000435171 | 0.000101671 | 0.000063815 |
| **Physical Machine** | 0.00362714 | 4.6739E-05 | 2.56434E-5 |
| **Comparation(less)** | V | P | P |
|  |  |  |  |
| **Edge=1K** | **graph\_\_generation** | **construction\_time** | **bfs mean\_time** |
| **Virtual Machine** | 0.000275946 | 0.000171534 | 0.000123148 |
| **Physical Machine** | 0.000245487 | 0.000111133 | 4.60188E-5 |
| **Comparation** | P | P | P |
|  |  |  |  |
| **Edge=1M** | **graph\_\_generation** | **construction\_time** | **bfs mean\_time** |
| **Virtual Machine** | 0.40025 | 0.155881 | 0.101732 |
| **Physical Machine** | 0.427032 | 0.0677495 | 0.0179761 |
| **Comparation** | V | P | P |
|  |  |  |  |

Therefore, with the comparation, we can conclude that the physical machine has the faster running time for BFS algorithm than the virtual machine. Though, there is natural error due to the limited number of edges (too small). With the comparation of computer configuration, we can get that when the machine has better performance CPU and Memory, the BFS algorithm can run faster on the machine.