if reboot, don't need publich-key again.
just login. A>ssh_ B
>exit

Spring 2019 CIS 620 Homework 2

(Due Mar. 27)

The purpose of this assignment is to introduce you the CUDA GPU and the MPI programming environments. Use the following command

tar xvfz ~cis620s/pub/gpu620.tar.gz

to extract the necessary files under the directory `NVIDIA_CUDA_620_S19`.

Part I: deviceQuery

Use `make` to build the executable file `deviceQuery` under the subdirectory `deviceQuery` and then run it. Take a screen shot of the result. What is the GPU clock rate? How many CUDA cores?

Part II: Euclidean Distance

You are asked to use several GPUs over MPI to calculate the average Euclidean distance to the origin in parallel. First, follow the instructions in ~cis620s/pub/MPI_setup. Next, you need to modify the code under the subdirectory simpleMPI. Assume that for each point i, its coordinate is $(A[i], B[i])$. Therefore, your root node has to initialize the array A and the array B. Then, use the MPI_Scatter to dispatch the data to each node specified in the file `machinefile` for calculation on GPU. Before calling MPI_Reduce to collect the results, each node should print its local summation result along with its hostname. Take a screen shot of the result.

Turning it in

Each groups needs to submit this homework using the following turnin command on grail:

```
turnin -c cis620s -p hw2 NVIDIA_CUDA_620_S19
```

Each group also needs to hand in a hard-copy document which includes the description of your code, experiences in testing/debugging, experimental results, etc. The document should be typed. The cover page should contain your photo(s), name(s) and the login-id you used to turnin. Start on time and good luck. If you have any questions, send e-mail to `sang@eecs.csuohio.edu`.

after setup MPI first ↳ check MPI_hello is working → check (handout)
return the program

Part1: cd NVIDIA_CUDA_620_S19/DeviceQuery/ $ make → Pic2
$ > ./deviceQuery → This will show this above
and take one screenshot
of the clock rate/core etc
regis

Part2 modify the simple_mpi.cu -

MPI & CPU

Details can be found in
https://help.ubuntu.com/community/MpichCluster

Below is just for CIS620 students :

1. Pick up a machine (e.g. arthur) from which you want to run the MPI root node. Login to the machine.

   Use
   arthur> ssh-keygen -t rsa
   and type a passphrase to generate an RSA key pair under the default directory ~/.ssh/id_rsa.

2. Add this key to authorized keys:
   arthur> cd .ssh
   arthur> cat id_rsa.pub >> authorized_keys (if authorized_keys exists)
   or
   cp id_rsa.pub authorized_keys (if authorized_keys not exists)

(You may repeat Steps 1 and 2 by choosing another machines)

3. Edit the .cshrc file under your home directory and put the following :

cat ~cisbros/pub/MPI_setup -5

```
# keychain for mpi app
if (-e /usr/bin/keychain) then
        keychain --nogui -q id_rsa
        set host=`uname -n`
        if (-f $HOME/.keychain/$host-csh) then
             source $HOME/.keychain/$host-csh
        endif
        if (-f $HOME/.keychain/$host-csh-gpg) then
             source $HOME/.keychain/$host-csh-gpg
        endif
endif
```

4. To test passwordless SSH login (from arthur),
   arthur> ssh bach
   and type the passphrase. Exit the machine bach and login again. You won't be asked for the passphrase the second time.

5. To test MPI programs, use an editor to build the program mpi_hello.c (see below).

   Compile it:
   mpicc -o mpi_hello mpi_hello.c

   Use an editor to type the machine names into the file machinefile:
   arthur:1
   bach:1
   chopin:1
   degas:1

   Run it on a single machine:
   mpirun -n 2 ./mpi_hello
   Note that the parameter next to -n specifies the number of processes to spawn and distribute among nodes

   Run it among several machines specified in the file machinefile:
   mpirun -n 2 -f machinefile ./mpi_hello
   mpirun -n 8 -f machinefile ./mpi_hello

---------- mpi_hello.c ----------

```
#include <stdio.h>
#include <mpi.h>
#include <unistd.h>

int main(int argc, char** argv) {
  int myrank, nprocs;

  char hostname[256];
  gethostname(hostname, 256);

  MPI_Init(&argc, &argv);
  MPI_Comm_size(MPI_COMM_WORLD, &nprocs);
  MPI_Comm_rank(MPI_COMM_WORLD, &myrank);

  printf("Hello from %s processor %d of %d\n", hostname, myrank, nprocs);

  MPI_Finalize();
  return 0;
}
```