CS 495 Assignment 2

Parker Joncus A20350578

September 15, 2018

Exercise 1: HPL on GPU

(a) Calculate the theoretical flops of a Nvidia P100 GPU or Nvidia Tesla V100. I used chameleon with the P100's since the matrix cluster does not seem to be completely finished yet.

$$Texture\ Units * Raster\ Operators * Core\ Clock$$

I was not able to find the hardware specifications to calculate this myself, but according to `https://images.nvidia.com/content/tesla/pdf/nvidia-tesla-p100-PCIe-pdf`, the p100 has a double-precision theoretical rate of 4.7 Teraflops.
Similarly for the V100, I found that the theoretical rate is 7-7.8 Teraflops from `https://www.nvidia.com/en-us/data-center/tesla-v100/`.

(b) Compile Linpack on targeting the GPU
I was able to compile HPL targeting the GPU. I followed the two guides on the websites: `http://hpl-calculator.sourceforge.net/Howto-HPL-GPU.pdf` and `http://www.advancedclustering.com/act_kb/installing-nvidia-drivers-`
I compiled HPL using icc, mpiicc, and mkl, like how I did in the last assignment.

(c) Run Linpack with double precision using at least 3/4 of memory.
Even though I was able to compile HPL, When I tried to run it, it says that I have 0 GPUs and 1 is required. I am not sure why I have this error since I obviously do have GPUs. See screenshots below:

```
[cc@parker-gpu ~]$ nvidia-smi
Sat Sep 15 13:30:46 2018
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 390.87                 Driver Version: 390.87                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla P100-PCIE...  Off  | 00000000:03:00.0 Off |                    0 |
| N/A   24C    P0    29W / 250W |      0MiB / 16280MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Tesla P100-PCIE...  Off  | 00000000:82:00.0 Off |                    0 |
| N/A   18C    P0    26W / 250W |      0MiB / 16280MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
[cc@parker-gpu ~]$
```

```
[cc@parker-gpu CUDA]$ mpiexec -n 1 ./xhpl
================================================================================
HPLinpack 2.0  --  High-Performance Linpack benchmark  --   September 10, 2008
Written by A. Petitet and R. Clint Whaley,  Innovative Computing Laboratory, UTK
Modified by Piotr Luszczek, Innovative Computing Laboratory, UTK
Modified by Julien Langou, University of Colorado Denver
================================================================================

An explanation of the input/output parameters follows:
T/V    : Wall time / encoded variant.
N      : The order of the coefficient matrix A.
NB     : The partitioning blocking factor.
P      : The number of process rows.
Q      : The number of process columns.
Time   : Time in seconds to solve the linear system.
Gflops : Rate of execution for solving the linear system.

The following parameter values will be used:

N      :   25000
NB     :     768
PMAP   : Row-major process mapping
P      :       1
Q      :       1
PFACT  :   Crout
NBMIN  :       2
NDIV   :       2
RFACT  :   Crout
BCAST  :   1ring
DEPTH  :       1
SWAP   : Mix (threshold = 192)
L1     : no-transposed form
U      : no-transposed form
EQUIL  : yes
ALIGN  : 8 double precision words

--------------------------------------------------------------------------------

- The matrix A is randomly generated for each test.
- The following scaled residual check will be computed:
      ||Ax-b||_oo / ( eps * ( || x ||_oo * || A ||_oo + || b ||_oo ) * N )
- The relative machine precision (eps) is taken to be               1.110223e-16
- Computational tests pass if scaled residuals are less than                16.0

!!! ERROR: Not enough GPUs on node parker-gpu.novalocal, 0 GPUs found, 1 GPUs required !!!
[cc@parker-gpu CUDA]$
```

(d) Could not record power information since HPL would not recognize the GPUs.

(e) Could not get eficiency since HPL would not recognize the GPUs.

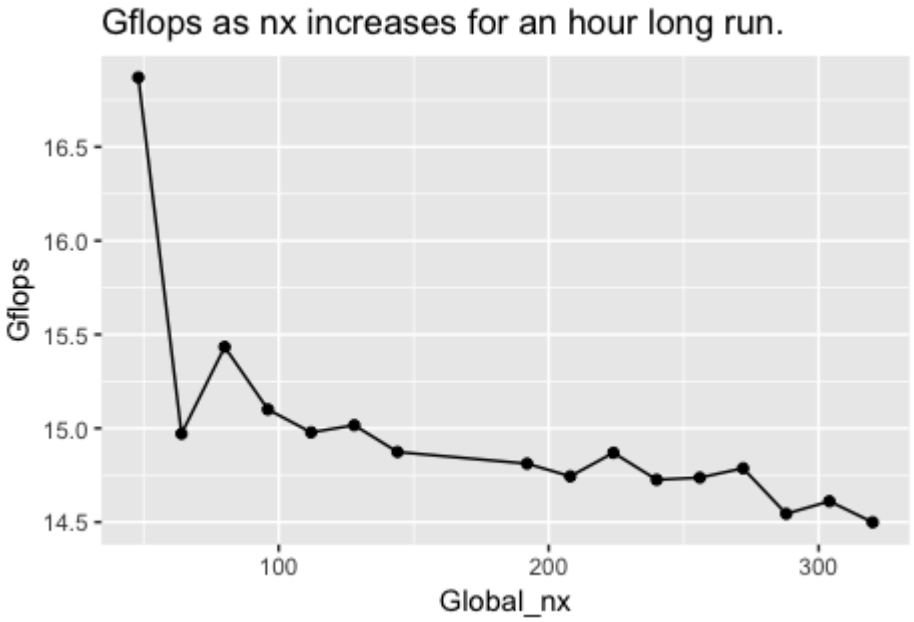
Exercise 2: HPCG on Skylake

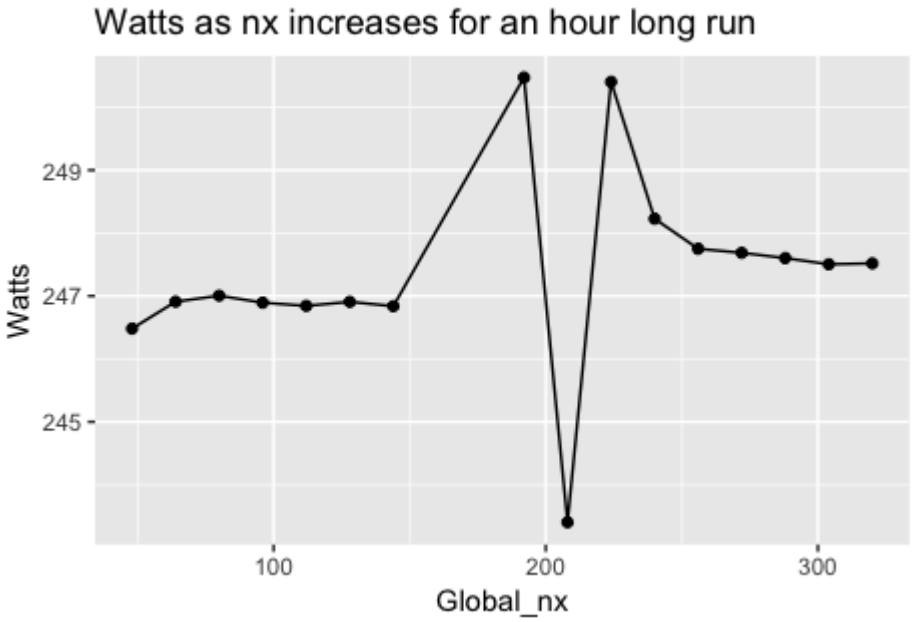
(a) Compiling hpcg was similar to hpl, the main difference is that hpcg does not require a BLAS library. The way I compiled hpcg can be found in the compileICC file. This file is the same as assignment1, I just added the hpcg part at the bottom.

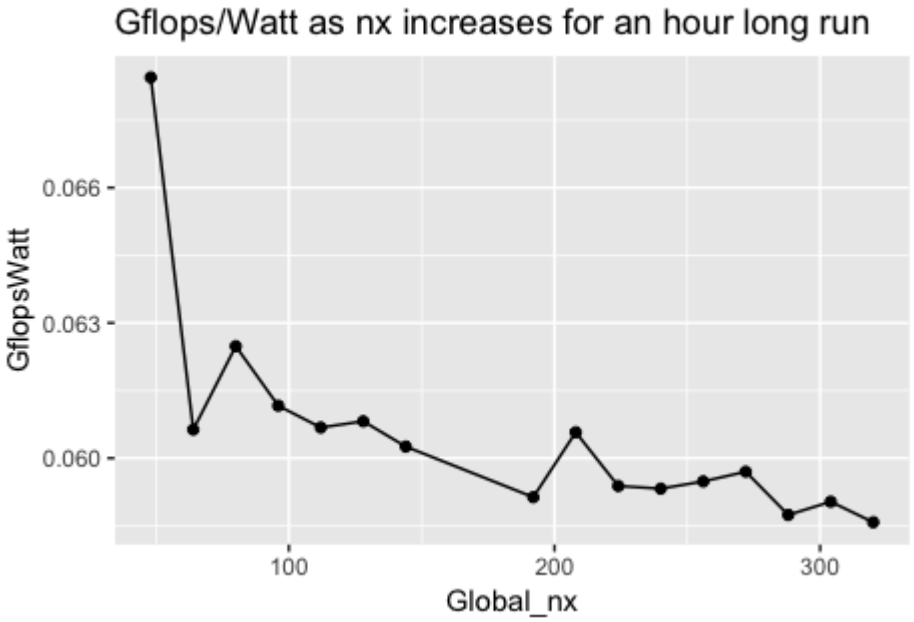
(b) I ran hpcg by running the run file. This file calls on helper files to create the hpcg.dat files. These files are a lot easier to tune since there are only two things that can be changed. The first is what mainly tunes hpcg. The second is how long to run the benchmark. In my tests, I ran the bencharks for an hour each or 3600 seconds. I wrote the helper file writeHPCG to write files with the first line have many different options from 8 to 160. It also kept the time the same. Other helper files include the powerMoniter file that will start the powerstat and will pipe the data into the powerdata file. I am also piping the current date before and after each run so that I can join the powerdata with the corresponding run. I also have the file movey which moves all of the yaml files in the directory output. Last is removetext, which just cleans up and removes all the .txt files from hpcg. I can then take all the files in output and scrape them for the desired data. Once everything is done, I clean up all of the data files and turn them into csv files. I can then upload the csv files into R where I run the setupPowerData.R script to create the plots.

(c) Covered in item b)

(d) Results:







It can be seen in these results that we seem to get better results with nx is smaller. The wattage for all of the nx values is about the same (within 2 watts), but there is a weird dip at nx=208. As far as the Gflops and Gflops per watt plots, it can be seen that the best performance is at nx=48 and it decreases as nx gets larger. This relationship almost looks negative logarithmic. All of the tests were ran with 24 processes on the node.
Another weird thing about hpcg is that even though I specified that I wanted the benchmark to run for an hour, it didnt actually take an hour to run. I think this is because there are 24 processes so in total they ran an hour if they all run for a couple of minutes, but this jsut seems a little weird to me.

Exercise 3: HPCG on GPU

(a) I was not able to find the HPCG GPU compatible download to compile, so I downloaded the HPCG GPU binary from `http://www.hpcg-benchmark.org`. This binary should run and give me hpcg results, however it kept giving me an error saying that I am missing libcudart.so.9.0, however I instead have libcudart.so.9.2.18.