CS 495 Assignment 1

Parker Joncus A20350578

August 30, 2018

Exercise 1: Skylake CPU wtih gcc

(a) Calculate the theoretical flops of a Skylake node

$$Cores * Sockets * AVX * Instructions\ Per\ Cycle * Double\ Precicion$$

Cores: The skylake cpu has 12 cores per socket
Sockets: There are two sockets
AVX: The CPU runs at 1.3 MHz with AVX
Instructions Per Cycle: There are 8 instructions per cycle
Double Precision: 8 bits of double precision.

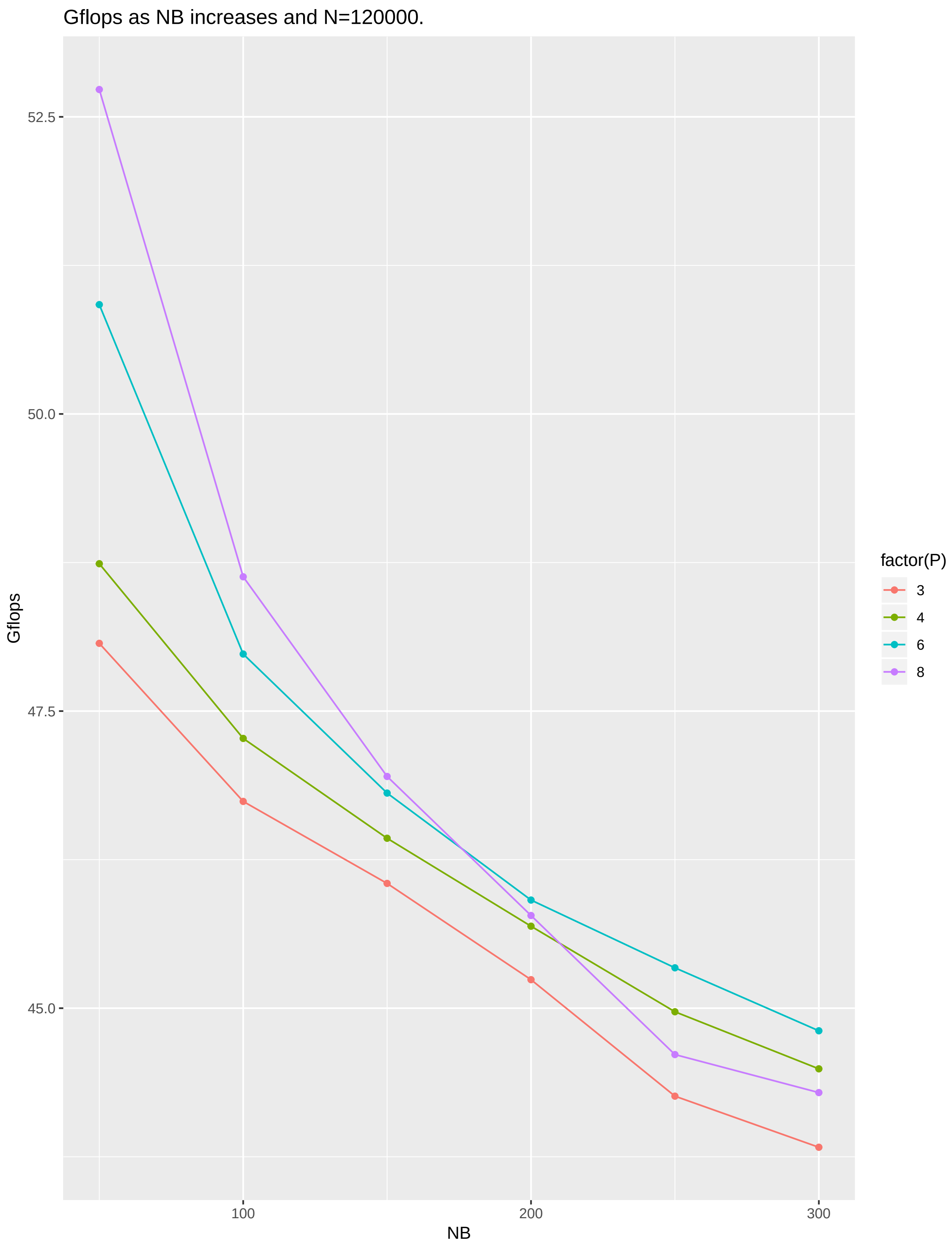
$$FLOPS = 12 * 2 * 1.3 * 8 * 8 = 1,996.8 = 1.996\ GFLOPS$$

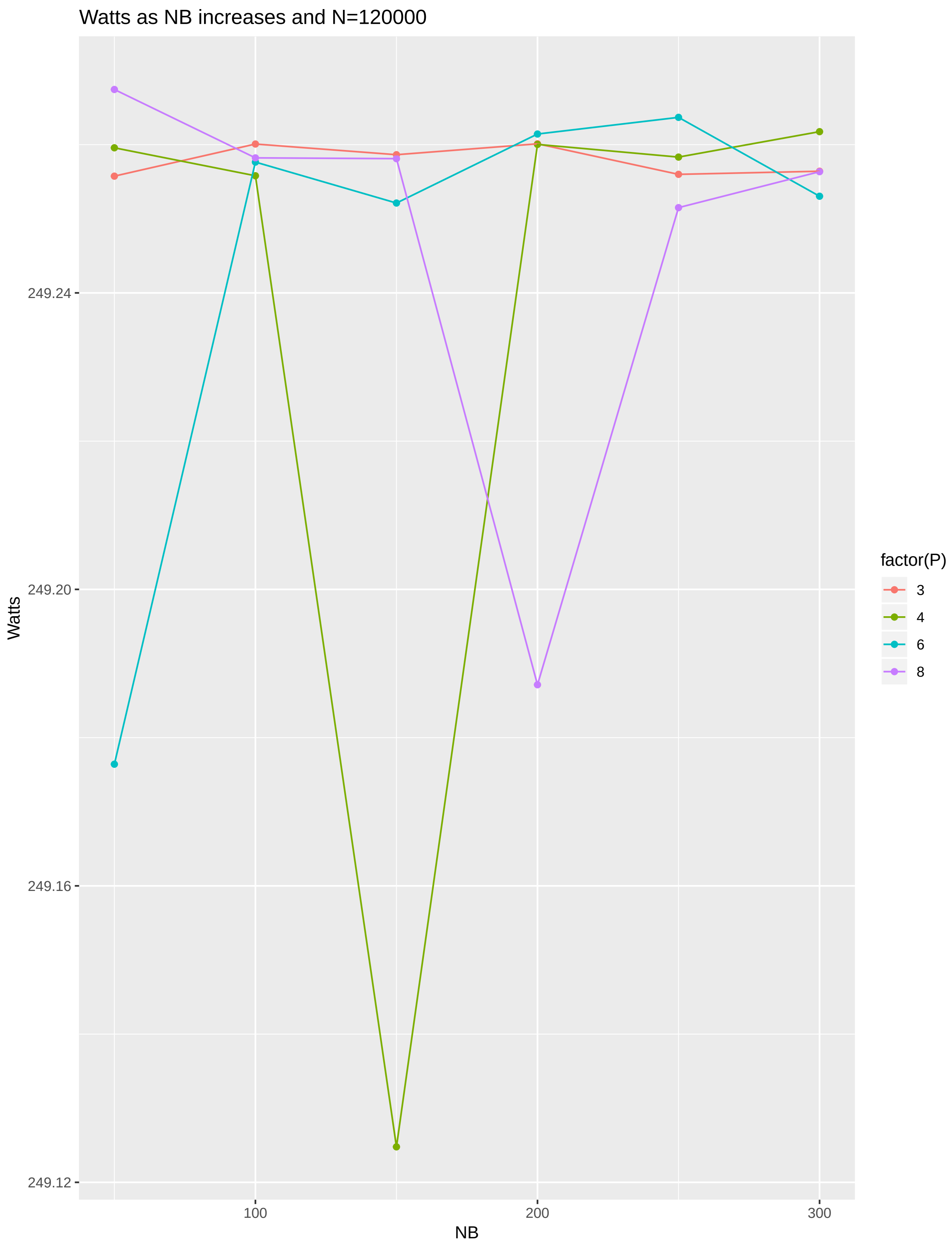
(b) Compile Linpack on the Skylake CPUs
See code compileGCC which will build mpich, install blas and compile hpl.

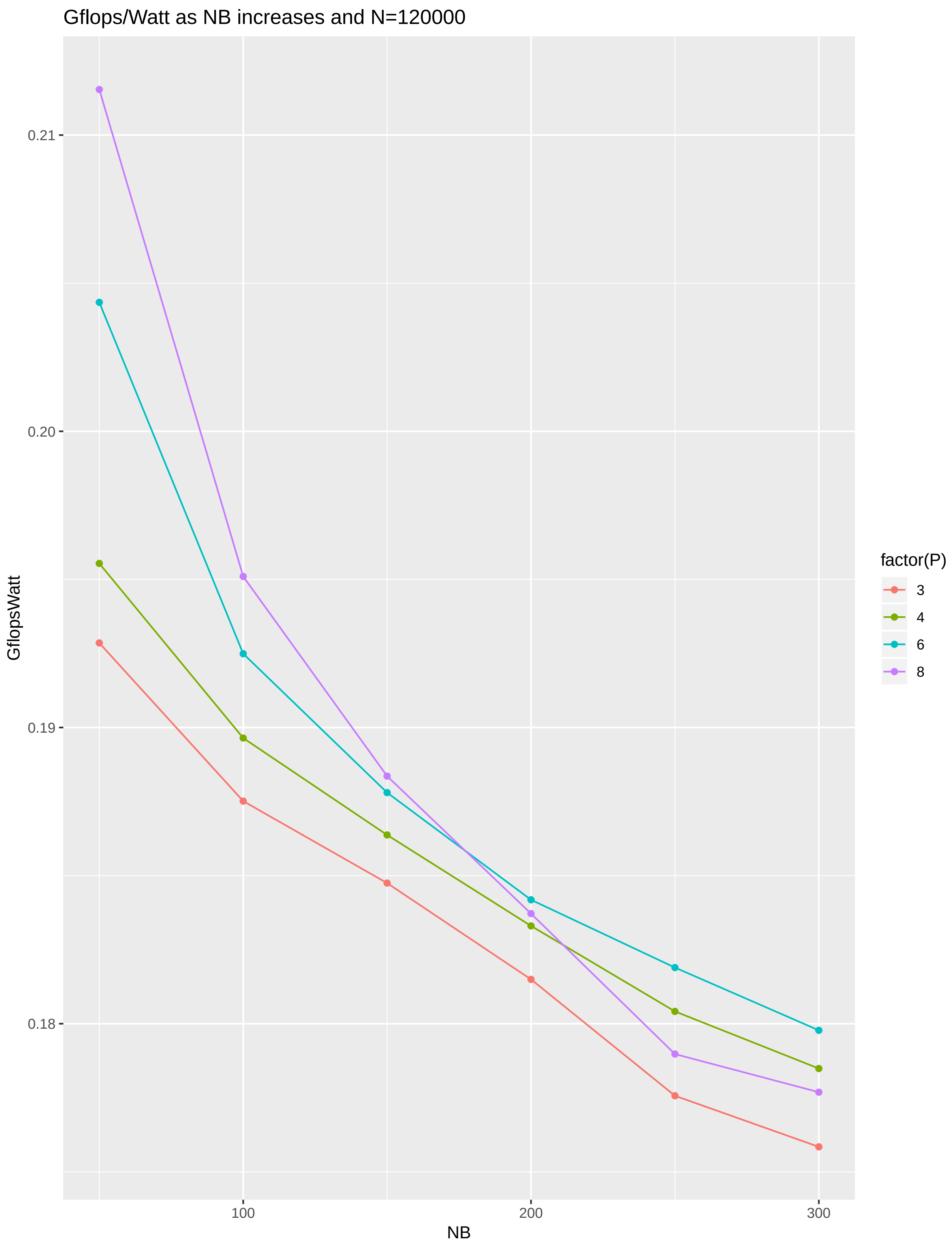
(c) Run Linpack with double precision using at least 3/4 of memory.
Run code benchmark in SkylakeGCC directory. This file calls on some other helper functions that overall will write HPL.dat files for each desired run, start data collection, run hpl, and store extract and store the data as a csv file.

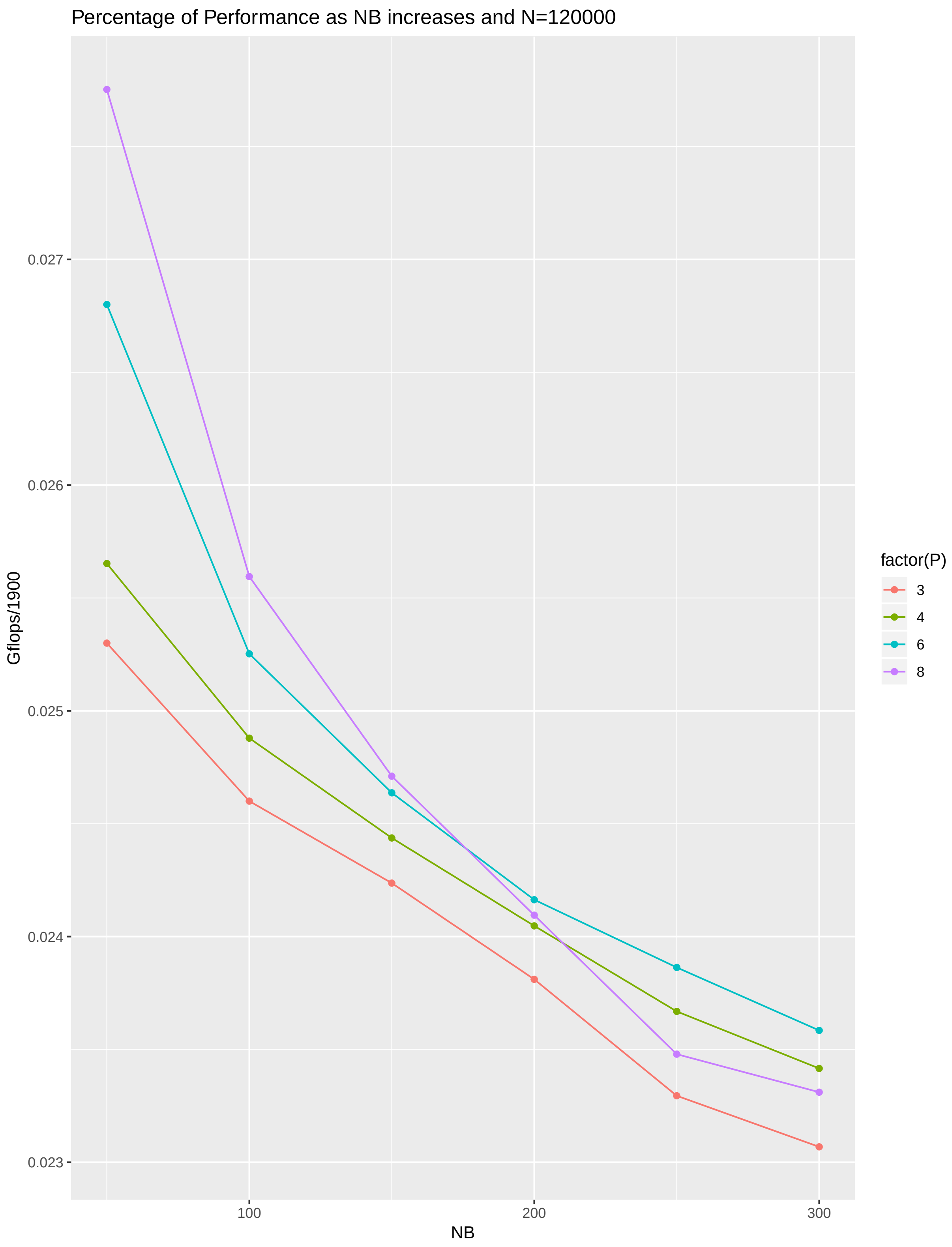
(d) Record Power information while Linpack is running.
See executable file powerMoiter in SkylakeGCC directory. This file is run within benchmark and will start the power monitoring before running hpl and will be killed when hpl is finished. The time that hpl starts and finishes will also be recorded and then compared to the power dataset to get the power consumption while hpl is run.

(e) Results:
The results were plotted by a variant of the setupPowerData.R


Gflops as NB increases and N=120000.
Gflops
52.5
50.0
47.5
45.0
NB
100
200
300
factor(P)
3
4
6
8


Watts as NB increases and N=120000
Watts
249.24
249.20
249.16
249.12
100
200
300
NB
factor(P)
3
4
6
8


Gflops/Watt as NB increases and N=120000
0.21
0.20
0.19
0.18
GflopsWatt
100
200
300
NB
factor(P)
3
4
6
8



The maximum flops obtained with GCC 4.5 was about 53 Gflops, which is really bad. This graph shows the wattage usage which should be constant after a while since it will be doing as much work as it can. It can be seen that the flops/watt is largest while NB is small. This graph shows the percentage of the theoretical flops obtained, which is very low. I believe this is due to not many optimizations, gcc being a early version, and blas not being optimized.

Exercise 2: NVIDIA GPU

(a) Calculate the theoretical flops of a Nvidia P100 GPU

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

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.

(b) - (e)
I was not able to work with a Nvidia p100 GPU. All the machines on chameleon were reserved this entire week. I was able to get a reservation for a little while and only managed to compile HPL on it, however I just compiled targeting the CPUs.

Exercise 3: Skylake CPU with icc compiler

(a) Calculate the theoretical flops of a Skylake node

$$Cores * Sockets * AVX * Instructions\ Per\ Cycle * Double\ Precicion$$

Cores: The skylake cpu has 12 cores per socket
Sockets: There are two sockets
AVX: The CPU runs at 1.3 MHz with AVX
Instructions Per Cycle: There are 8 instructions per cycle
Double Precision: 8 bits of double precision.

$$FLOPS = 12 * 2 * 1.3 * 8 * 8 = 1,996.8 = 1.996\ GFLOPS$$

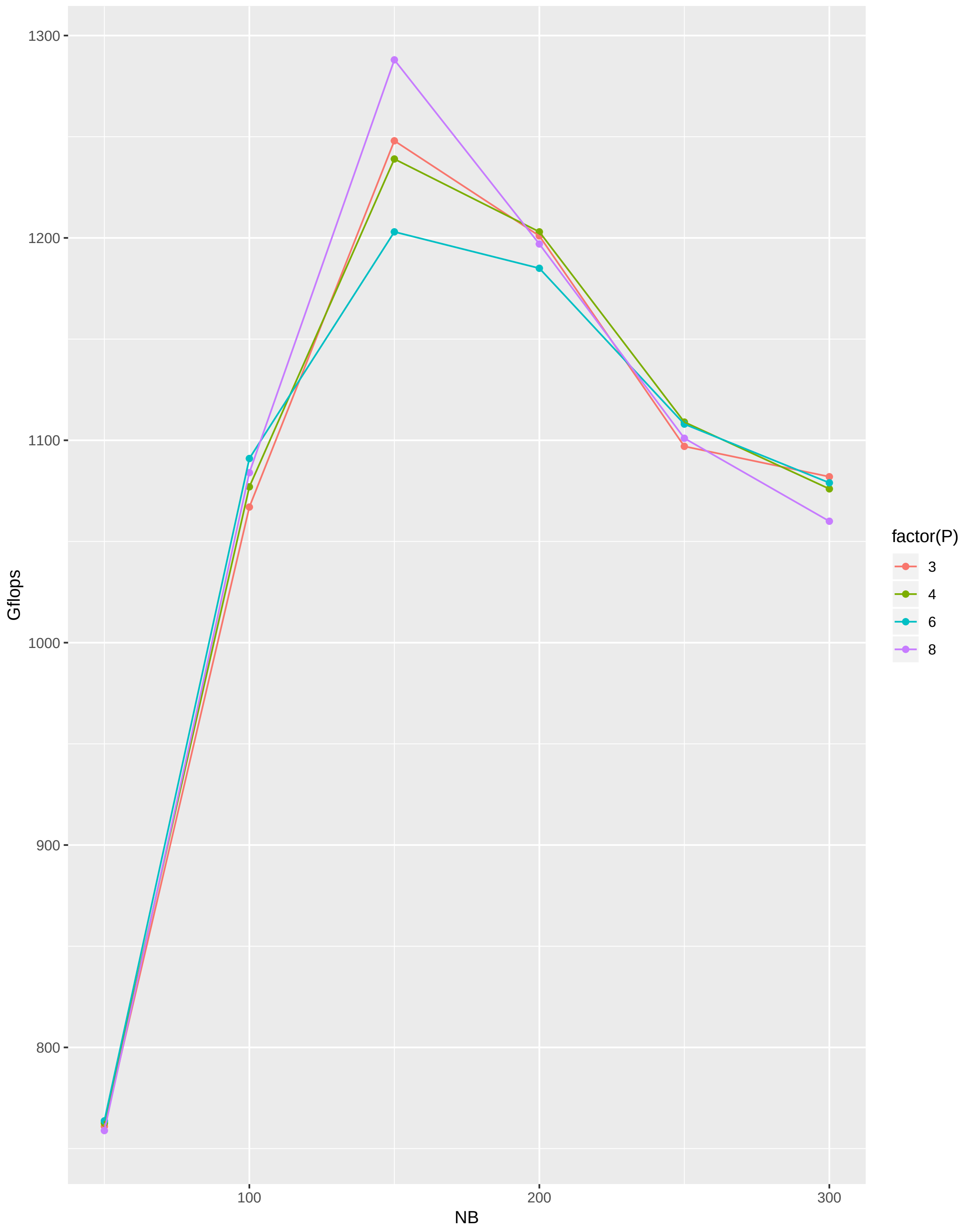
(b) Compile Linpack on the Skylake CPUs
See code compileICC. This will install the Intel compilers and compile HPL with the intel compilers.

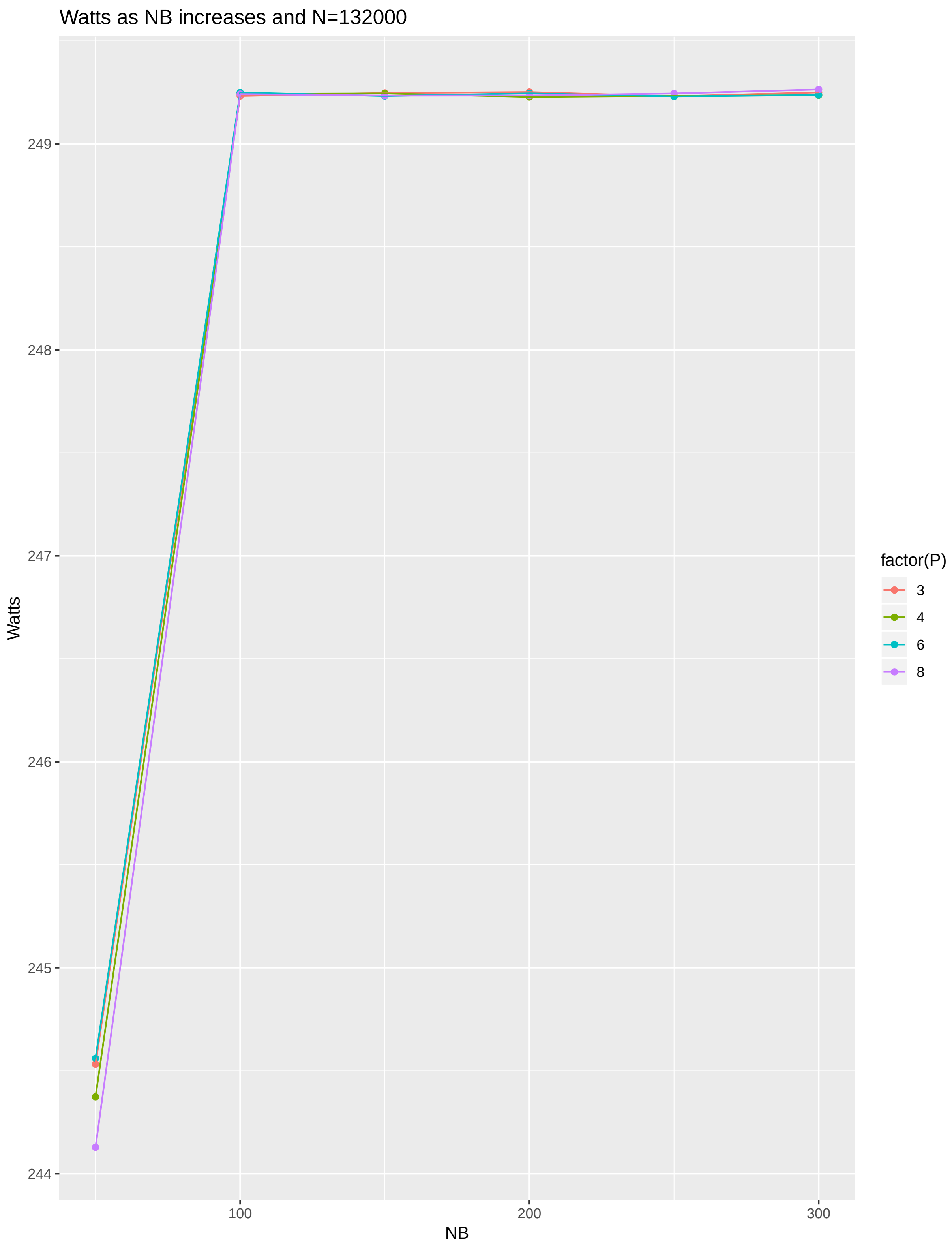
(c) Run Linpack with double precision using at least 3/4 of memory.
Run code benchmark in the SkylakeICC directory

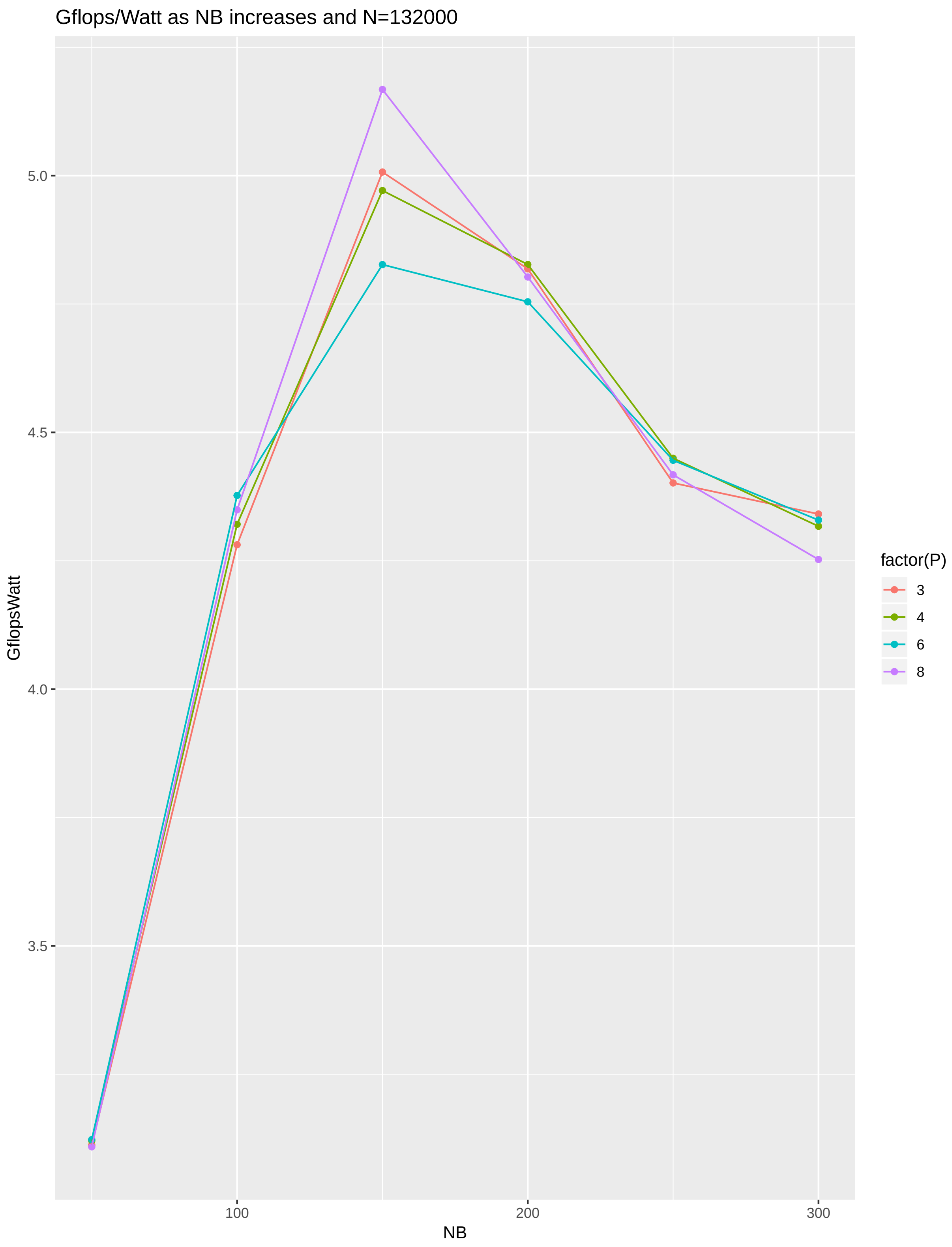
(d) Record Power information while Linpack is running.
See executable file powerMoiter in SkylakeICC directory. This file is run within benchmark and will start the power monitoring before running hpl and will be killed when hpl is finished. The time that hpl starts and finishes will also be recorded and then compared to the power dataset to get the power consumption while hpl is run.

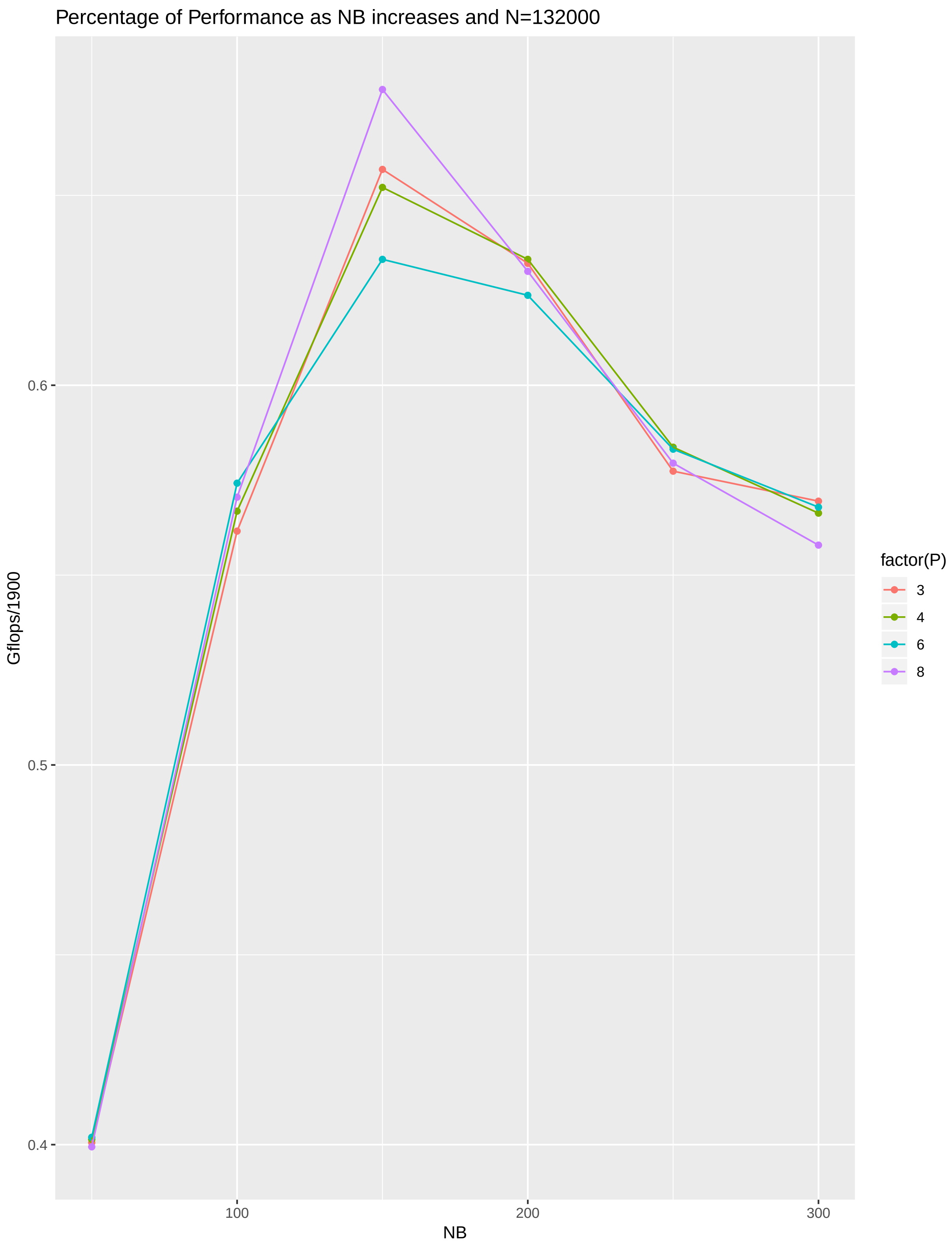
(e) Results:
The results were plotted by a variant of the setupPowerData.R

Gflops as NB increases and N=132000.




Watts as NB increases and N=132000
Watts
244
245
246
247
248
249
NB
100
200
300
factor(P)
3
4
6
8


Gflops/Watt as NB increases and N=132000
GflopsWatt
5.0
4.5
4.0
3.5
NB
100
200
300
factor(P)
3
4
6
8



The maximum flops obtained with ICC was a little under 1.3 Teraflops. This is a ton better than gcc, at about 20 times better. It can also be seen that the wattage plateaus once the cpu is fully loaded. We also get that the largest flops per watt is when NB is around 150. Last, the percentage of the theoretical performance is a little below .7 which is good, but could still be better. I will look to try to optimize this more and get more flops out of it. Since ICC performs so much better than gcc, I plan to use ICC for the rest of my experiments.

Exercise 4: Haswell CPU with icc compiler

(a) Calculate the theoretical flops of a Haswell node

$$Cores * Sockets * AVX * Instructions\ Per\ Cycle * Double\ Precicion$$

<u>Cores:</u> The haswell cpu has 12 cores per socket
<u>Sockets:</u> There are two sockets
<u>AVX:</u> The CPU runs at 1.3 MHz with AVX
<u>Instructions Per Cycle:</u> There are 8 instructions per cycle
<u>Double Precision:</u> 8 bits of double precision.

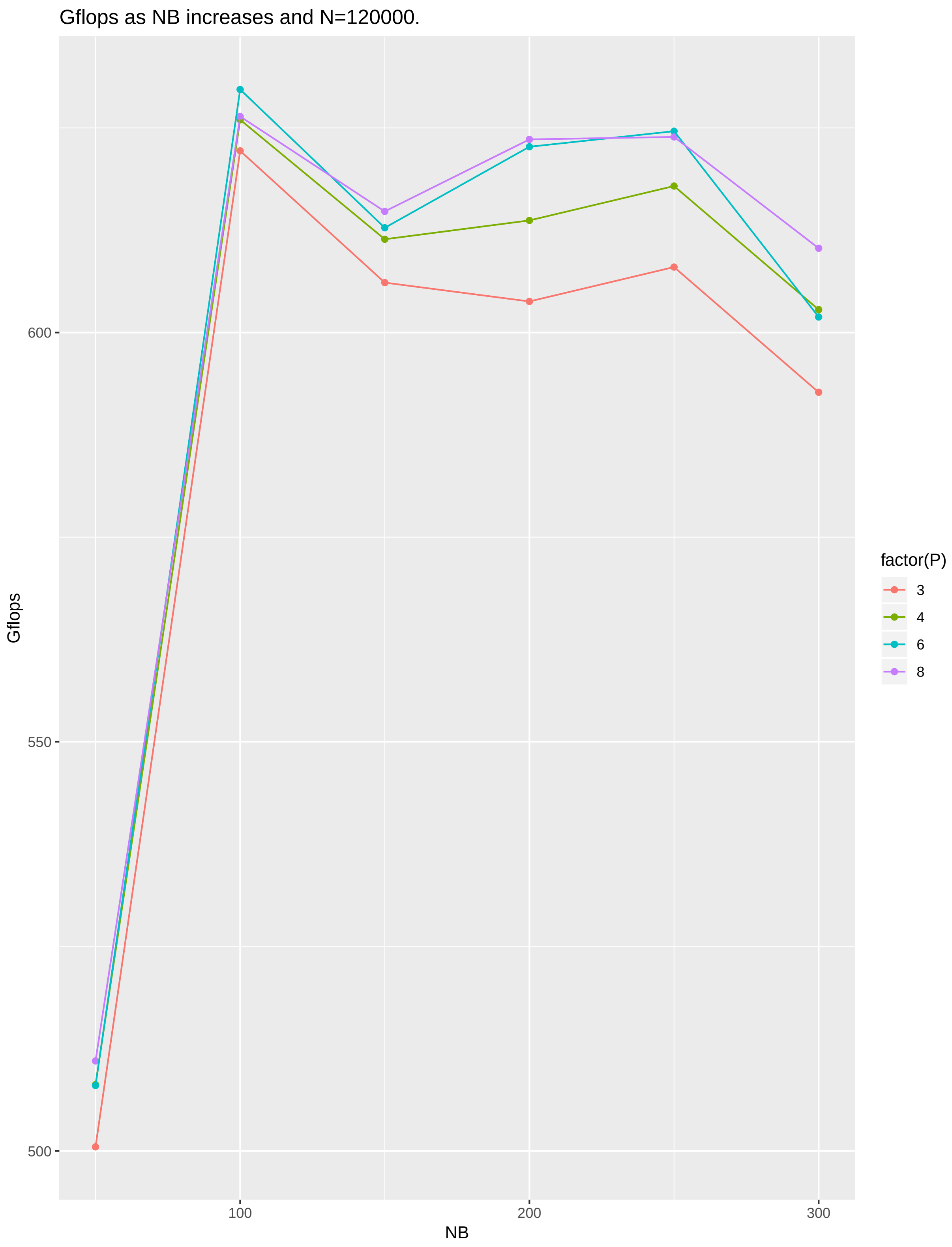
$$FLOPS = 12 * 2 * 1.3 * 8 * 8 = 1,996.8 = 1.996\ GFLOPS$$

(b) Compile Linpack on the Haswell CPUs
See code compileICC.

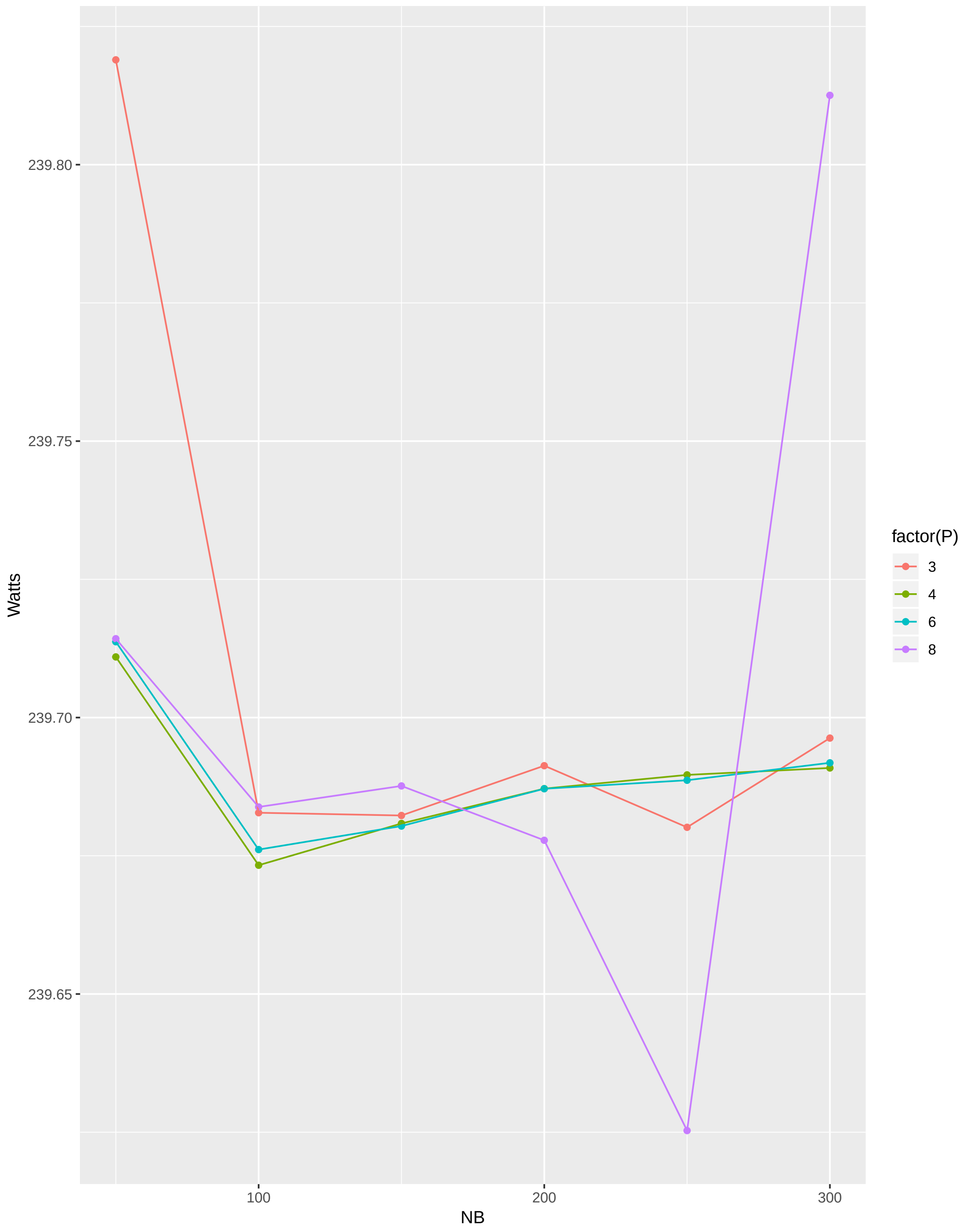
(c) Run Linpack with double precision using at least 3/4 of memory.
Run code benchmark in the HaswellICC directory.

(d) Record Power information while Linpack is running.
Same process as other exercises.

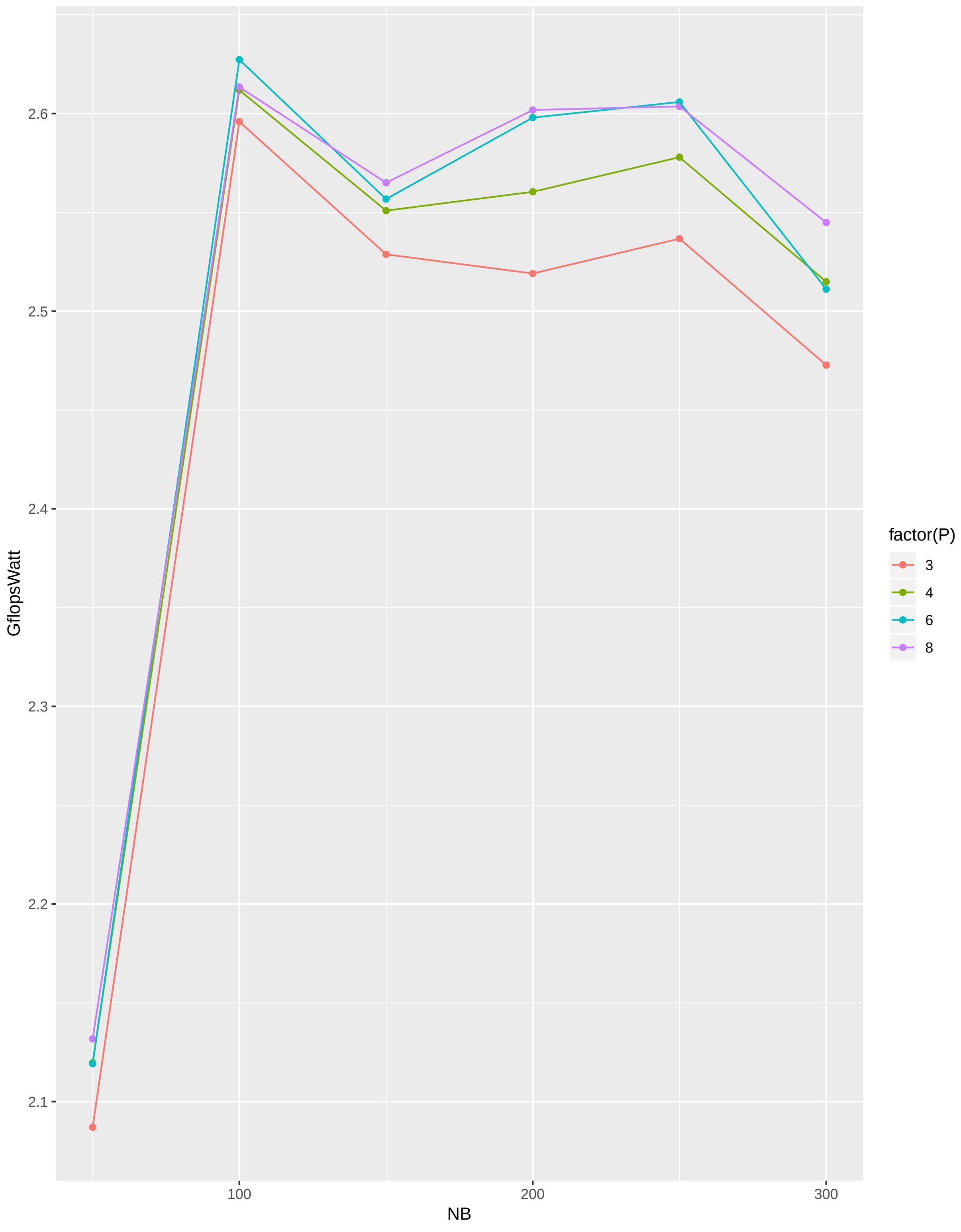
(e) Results:

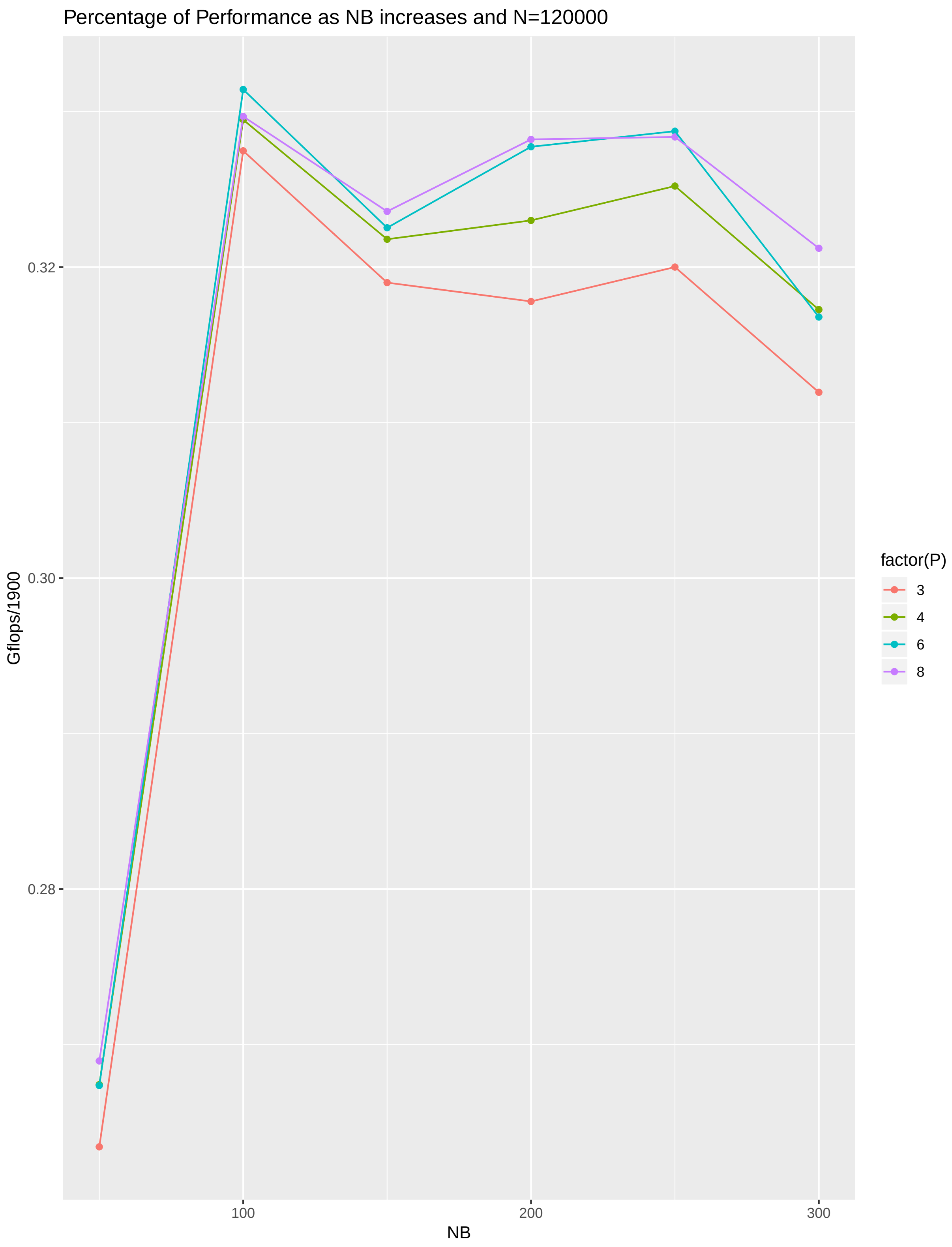

Gflops as NB increases and N=120000.
Gflops
600
550
500
NB
100
200
300
factor(P)
3
4
6
8

Watts as NB increases and N=120000



Gflops/Watt as NB increases and N=120000





From these plots it can be seen that we get a maximum performance of a little over 700 Gflops which is alright, but a lot less than the skylake nodes. The wattage also seems to be a bit all over the place with a typical usage of about 239 watts. The flops per watt is similar to the flops because all of the wattages are about the same. The percentage of performance is a maximum of .34 which is not very good. This may be because the theoretical performance was calculated wrong, since the compiling was exactly the same as the skylake with icc.

Exercise 5: Skylake CPU weak scaling changing number of cores

(a) Calculate the theoretical flops of a Skylake node

$$Cores * Sockets * AVX * Instructions\ Per\ Cycle * Double\ Precicion$$

Cores: The skylake cpu has 12 cores per socket
Sockets: There are two sockets
AVX: The CPU runs at 1.3 MHz with AVX
Instructions Per Cycle: There are 8 instructions per cycle
Double Precision: 8 bits of double precision.

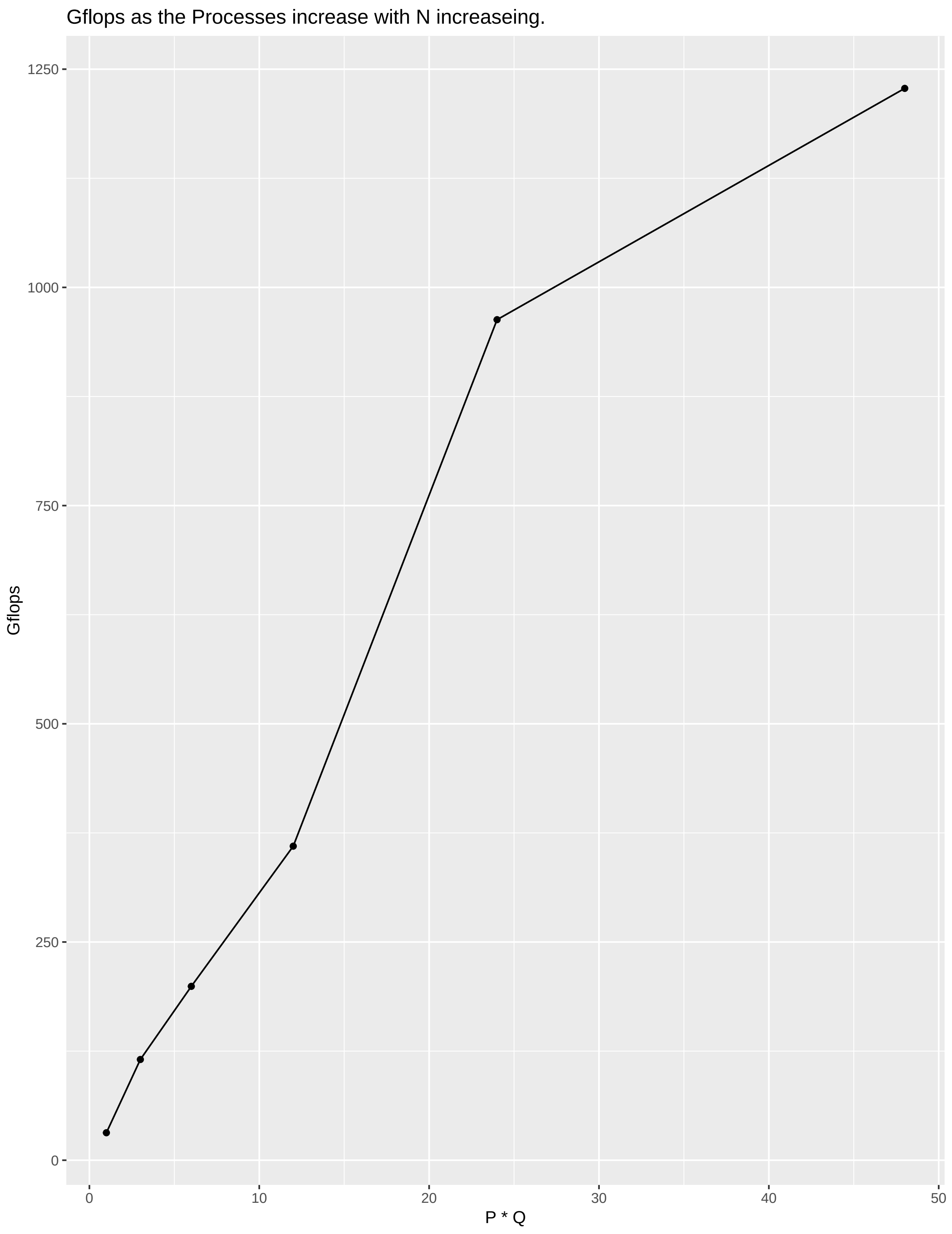
$$FLOPS = 12 * 2 * 1.3 * 8 * 8 = 1,996.8 = 1.996\ GFLOPS$$

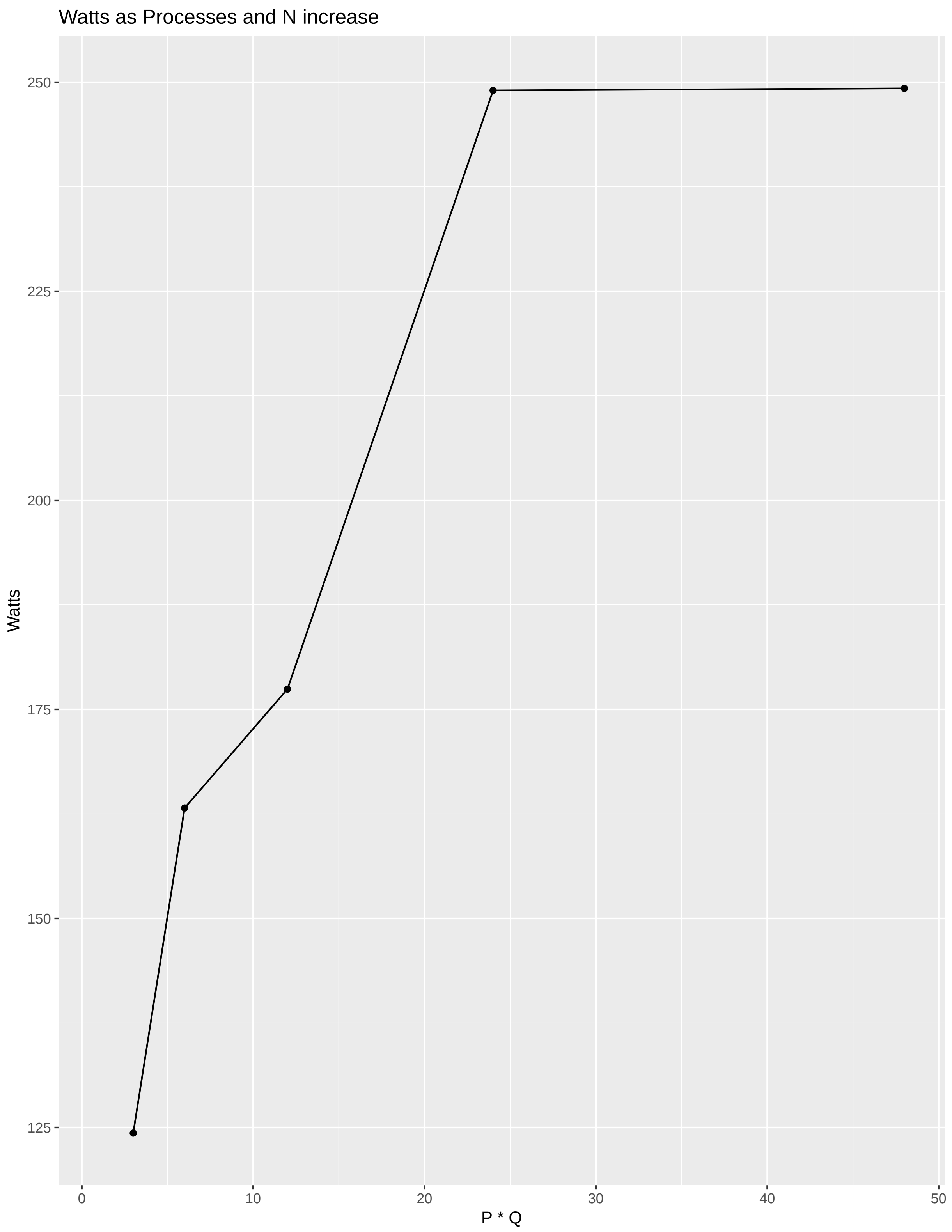
(b) Compile Linpack on the Skylake CPUs
See attached code[1]

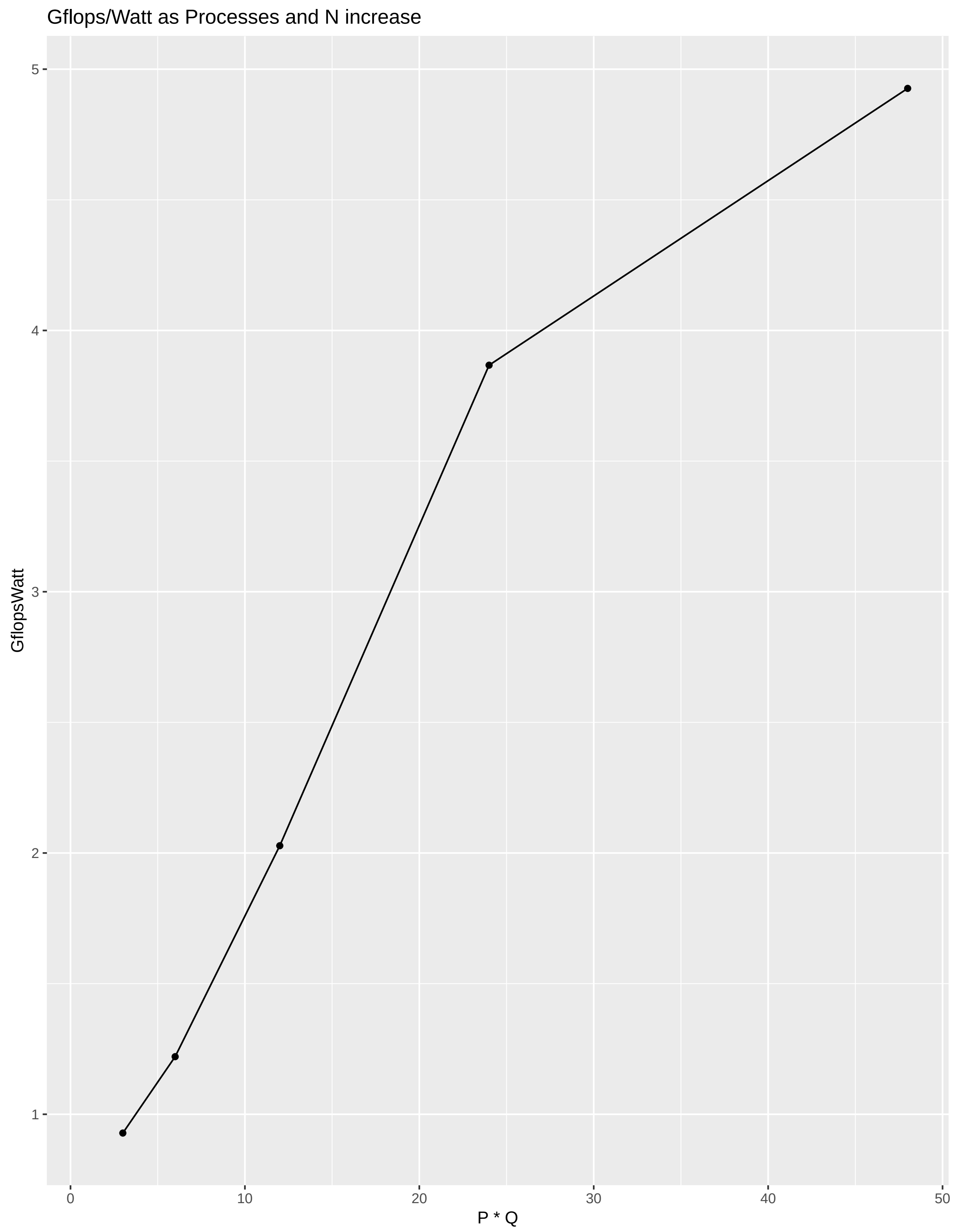
(c) Run Linpack with double precision using at least 3/4 of memory.

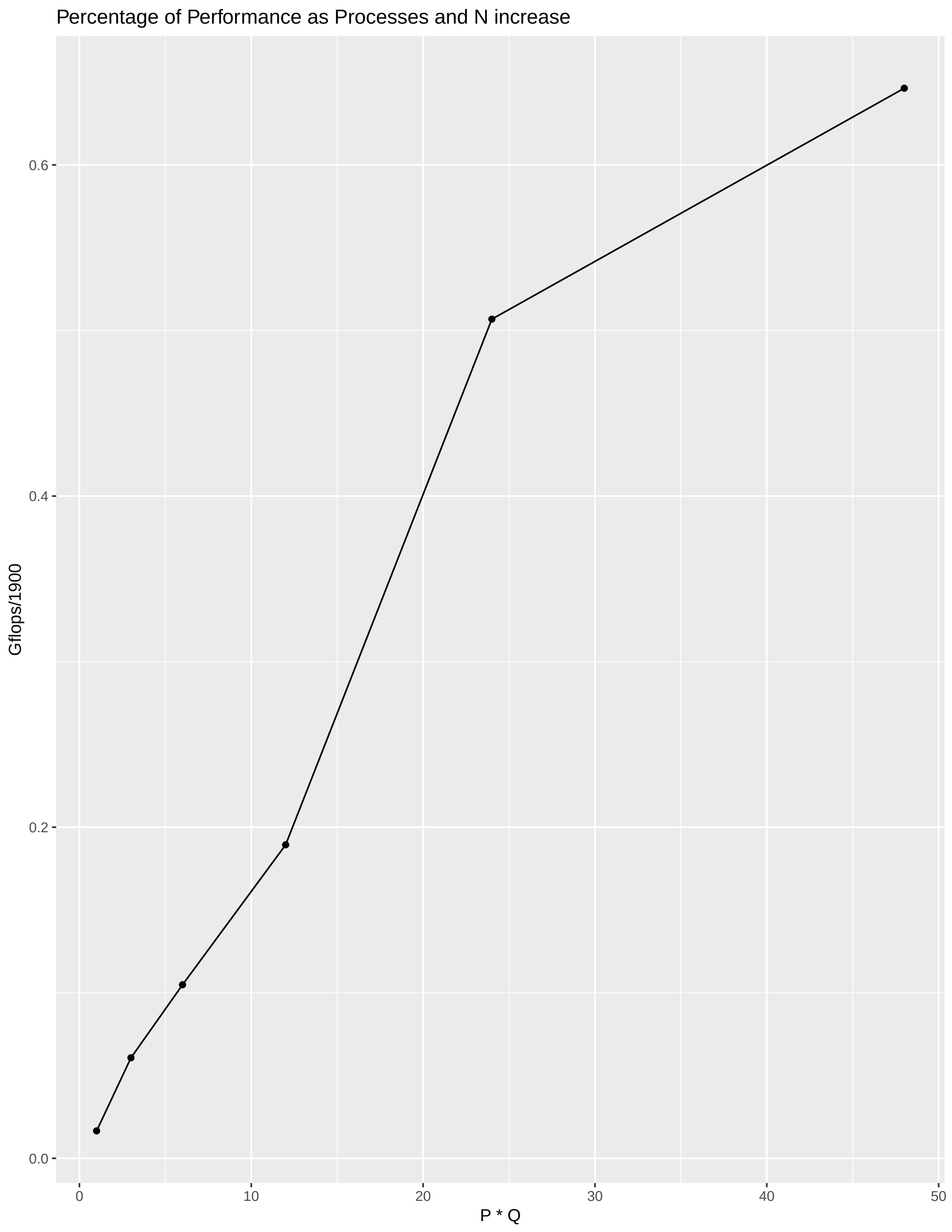
(d) Record Power information while Linpack is running.

(e) Results:


Gflops as the Processes increase with N increaseing.
Gflops
P * Q
0
250
500
750
1000
1250
0
10
20
30
40
50


Watts as Processes and N increase
Watts
P * Q
250
225
200
175
150
125
0
10
20
30
40
50


Gflops/Watt as Processes and N increase
GflopsWatt
P * Q
5
4
3
2
1
0
10
20
30
40
50



In this experiment, I wanted to test how the machines performed as the number of processes increased. To do this, I used weak scaling by keeping the increase in the number of processes similar to the increase in N. It can be seen that there is a pretty linear increase as the number of processes increase. I would be interested in testing if 24 processes is outright better than 48 processes since a classmate said they got better results with 24.

Findings:

All of the experiments had a maximum bound of 250 watts. Note, that this is just getting the CPU power usage. However, If the power usage is 250 per node, then we could use 12 nodes and stay around 3000 watts of power. This means we could theoretically get 12 times the current flops. This means the skylake with ICC could get 15.6 Teraflops, the haswell with ICC could get 8.4 teraflops and the Skylake with gcc could get 660 Gflops approximately. This is interesting since the gcc with 12 would do worse than 1 ICC node.
They key variables in hpl are P and Q, N, and NB. N should stay fairly constant. P and Q are paired together so if one changes the other needs to change. I tend to get the best performance when P and Q are close to each other, but Q is still the larger value. NB is hard to tune, since it depends on the system you are using. For example with gcc on a skylake, low NBs got the best results, whereas icc had higher NBs yield the best results. This variable should divide N and must be tested for a lot to find the best.