STAT_624 Homework-02

Compare performance of Intel FORTRAN (v2020) to GNU FORTRAN (v10.2.0).

Nano Script (Copy-pasted):

```
#!/bin/bash

#Compare performance of Intel FORTRAN (v2020) to GNU FORTRAN (v10.2.0).


#Intel compiler: How do the options:-no o flag, -00, -01...etc.


#How do the Intel compiler options (individually) impact performance?

# Which one of these two gives the biggest performance boost?


#GNU compiler: How do the options: impact performance?


module load intel/2020b # GCC v10.2.0


INTEL=("-O0" "-O1" "-O2" "-O3" "-fast")

FGNU=("-O0" "-O1" "-O2" "-O3" "-Ofast")
```

NOTE: I found these below descriptions for different options on stack overflow. These are not my original comments.

```
# O0 - No optimization

# O1 - Some speed

# O2 - All for speed

# O3 - Aggressive

# -fast and -Ofast for GNU


#The following for loop runs each compiler with different options

#and also records the times for each of these processes.
```

STAT_624 Homework-02

```
echo "Optimizer comparison between Intel Fortran vs GNU Fortran"


for i in {0..4}
     do
          ifort ${INTEL[$i]} -o brem brem.f gammln.f
          echo "Time for" ${INTEL[$i]} "optimization of Intel compiler is"
          time ./brem < thik.inp $> /dev/null
          gfortran ${FGNU[$i]} -o brem brem.f gammln.f
          echo "Time for" ${FGNU[$i]} "optimization of GNU compiler is"
          time ./brem < thik.inp $> /dev/null
done
```

OUTPUT:

```
Time for -O0 optimization of Intel compiler is


real    4m53.543s
user    2m28.818s
sys     0m0.209s
Time for -O0 optimization of GNU compiler is


real    8m15.449s
user    4m7.308s
sys     0m0.240s


Time for -O1 optimization of Intel compiler is


real    3m7.316s
user    1m22.392s
sys     0m0.071s
```

STAT_624 Homework-02

Time for -O1 optimization of GNU compiler is

```
real    7m19.707s
user    3m34.248s
sys     0m0.247s
```

Time for -O2 optimization of Intel compiler is

```
real    2m36.960s
user    1m17.064s
sys     0m0.094s
```

Time for -O2 optimization of GNU compiler is

```
real    7m49.657s
user    3m28.288s
sys     0m0.130s
```

Time for -O3 optimization of Intel compiler is

```
real    2m38.972s
user    1m19.798s
sys     0m0.114s
```

Time for -O3 optimization of GNU compiler is

```
real    6m47.678s
user    3m18.690s
sys     0m0.150s
```

Time for -fast optimization of Intel compiler is

```
real    1m1.429s
user    0m27.912s
sys     0m0.031s
```

Time for -Ofast optimization of GNU compiler is

```
real    6m55.829s
user    3m19.169s
sys     0m0.203s
```

CONCLUSIONS:

The intel compiler is much faster than the GNU compiler. The fastest real execution time for the GNU compiler was 6 minutes, 47.678 seconds, which is still larger compared to the slowest real execution time for the intel compiler (4minutes, 53.543 seconds). The -fast option gave the fastest execution time (1minute, 1.429 seconds) for the intel compiler, while the -O3 optimizer was the fastest (6minutes, 47.678 seconds) for the GNU option. However, the difference between the -O3 option and the -Ofast option was minor for the GNU compiler.

• Intel compiler: How do the options: no -O flag, -O0,-O1,-O2,-O3,-fast impact performance?

Nano Script (Copy-pasted):

```
#!/bin/bash

module load intel/2020b # GCC v10.2.0


INTEL=(" " "-O0" "-O1" "-O2" "-O3" "-fast")




#The following for loop runs the intel compiler with different options

#and also records the times for each of these processes.
```

STAT_624 Homework-02

```
echo "Optimizer comparison for  Intel Fortran"

for i in {0..5}
        do
                ifort ${INTEL[$i]} -o brem brem.f gammln.f
                echo "Time for" ${INTEL[$i]} "optimization of Intel compiler is"
                time ./brem < thik.inp $> /dev/null
done
```

OUTPUT:

#The following result is for a no -O flag case.

```
Time for optimization of Intel compiler is

real    2m31.647s
user    1m17.143s
sys     0m0.144s

Time for -O0 optimization of Intel compiler is

real    4m58.040s
user    2m29.091s
sys     0m0.216s

Time for -O1 optimization of Intel compiler is

real    2m50.073s
user    1m23.655s
sys     0m0.099s
```

Time for -O2 optimization of Intel compiler is

```
real    2m35.997s
user    1m16.932s
sys     0m0.121s
```

Time for -O3 optimization of Intel compiler is

```
real    2m41.499s
user    1m21.313s
sys     0m0.098s
```

Time for -fast optimization of Intel compiler is

```
real    0m55.717s
user    0m27.113s
sys     0m0.026s
```

CONCLUSIONS:

From the output above, we see that for the intel compiler, -fast optimizer gives the fastest processing time. The -no flag option defaults to -O2 and they both give similar results (2m31.647s, 2m35.997s). After the -fast option, -O2, -O3 give the next best results, in that order. However, the differences in processing time for optimizers -O0, O1,…O3 are not too large.

• How do the Intel compiler options (individually)

-xHost, -ipo

impact performance? Which one of these two gives the biggest

performance boost?

STAT_624 Homework-02

Nano Script (Copy-pasted):

```
#!/bin/bash


#How do the Intel compiler options (individually) impact performance?

# Which one of these two gives the biggest performance boost?


module load intel/2020b # GCC v10.2.0


INTEL=("-xHost" "-ipo")



#The following for loop runs the intel Fortran compiler with the -O2

#optimizer, and -xHost, -ipo options. This exercise checks the impact

#on compiler performance by these two options.


echo "Investigating the -xHost and -ipo options for Intel Fortran Compiler"


for i in {0..1}

    do

        ifort ${INTEL[$i]} -O2 -o brem brem.f gammln.f

        echo "Time for" ${INTEL[$i]} "option with default -O2 optimizer is"

        time ./brem < thik.inp $> /dev/null

done
```

OUTPUT:

```
Time for -xHost option with default -O2 optimizer is


real    3m4.568s
```

```
user    1m19.302s
sys     0m0.048s
```

Time for -ipo option with default -O2 optimizer is

```
real    0m56.317s
user    0m28.771s
sys     0m0.046s
```

CONCLUSIONS:

-ipo option gives the biggest performance boost. It is more than three times as fast as the -xHost option. -ipo is also approximately 2.5 times faster than the default -O2 optimizer without any other options. The -xHost option, on the other hand performs worse than the default -O2 optimizer by itself.

• GNU compiler: How do the options: no -O flag, -O2,-Ofast

impact performance?

Nano Script (Copy-pasted):

```
#!/bin/bash

#GNU compiler: How do the options: impact performance?

module load intel/2020b # GCC v10.2.0

FGNU=(" " "-O2" "-Ofast")

#The following for loop runs the GNU compiler with different options
#and also records the times for each of these processes.
```

STAT_624 Homework-02

```
echo "Optimizer comparison For GNU Fortran"

for i in {0..2}
    do

        gfortran ${FGNU[$i]} -o brem brem.f gammln.f
        echo "Time for" ${FGNU[$i]} "optimization of GNU compiler is"
        time ./brem < thik.inp $> /dev/null
done
```

OUTPUT:

#The following result is for a no -O flag case.

```
Time for optimization of GNU compiler is

real    8m26.925s
user    4m7.428s
sys     0m0.207s

Time for -O2 optimization of GNU compiler is

real    7m2.654s
user    3m28.952s
sys     0m0.281s

Time for -Ofast optimization of GNU compiler is

real    6m46.528s
user    3m19.130s
```

```
sys     0m0.238s
```

CONCLUSIONS:

From the output above, we see that for the GNU compiler, -Ofast optimizer gives the fastest processing time (of the three options investigated here). However, as we noted in the first part of this assignment, the -O3 optimizer performed the best for the GNU compiler. The -no flag option is slightly slower (8m26.925s) than the -O2 optimizer (7m2.654s). The difference in execution time between -Ofast and the other two options is noticeable.