Assignment 1

Shashvat Shah (sss9772)

HPML Spring 2023

Coding questions Results

Submit job used run on greene compute node with 16GB RAM

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=1
#SBATCH --time=3:00:00
#SBATCH --mem=16GB
#SBATCH --job-name=hw1
#SBATCH --output=hw1_%j

module purge

SRCDIR=$SCRATCH/hw1
module load gcc/10.2.0

$SRCDIR/dp1 1000000 1000 > hw1_$SLURM_JOB_ID.dp1.out
$SRCDIR/dp1 300000000 20 >> hw1_$SLURM_JOB_ID.dp1.out

$SRCDIR/dp2 1000000 1000 > hw1_$SLURM_JOB_ID.dp2.out
$SRCDIR/dp2 300000000 20 >> hw1_$SLURM_JOB_ID.dp2.out

module load python/intel/3.8.6
$SRCDIR/dp3 1000000 1000 > hw1_$SLURM_JOB_ID.dp3.out
$SRCDIR/dp3 300000000 20 >> hw1_$SLURM_JOB_ID.dp3.out

module load python/intel/3.8.6
python3 $SRCDIR/dp4.py 1000000 1000 > hw1_$SLURM_JOB_ID.dp4.out
python3 $SRCDIR/dp4.py 300000000 20 >> hw1_$SLURM_JOB_ID.dp4.out

module load python/intel/3.8.6
python3 $SRCDIR/dp5.py 1000000 1000 > hw1_$SLURM_JOB_ID.dp5.out
python3 $SRCDIR/dp5.py 300000000 20 >> hw1_$SLURM_JOB_ID.dp5.out
```

c1.

```
n:1000000 reps:1000
N: 1000000 <T>: 0.002614 sec B: 0.383 GB/sec F: 0.765 GFLOP/sec
n:300000000 reps:20
N: 300000000 <T>: 0.803314 sec B: 0.373 GB/sec F: 0.747 GFLOP/sec
```

c2.

```
n:1000000 reps:1000
N: 1000000 <T>: 0.001306 sec B: 0.766 GB/sec F: 5.360 GFLOP/sec
n:300000000 reps:20
```

```
N: 300000000 <T>: 0.416456 sec B: 0.720 GB/sec F: 5.043 GFLOP/sec
```

c3.

```
n:1000000 reps:1000
N: 1000000 <T>: 0.000330 sec B: 3.028 GB/sec F: 6.056 GFLOP/sec
n:300000000 reps:20
N: 300000000 <T>: 0.195640 sec B: 1.533 GB/sec F: 3.067 GFLOP/sec
```

c4.

```
[sss9772@log-1 hw1]$ gcc -I /share/apps/intel/19.1.2/mkl/include/ -L
/share/apps/intel/19.1.2/mkl/lib/intel64/ -o dp3 dp3.c -lmkl_intel_lp64 -lmkl_sequential -
lmkl_core -lpthread -lm

N 1000000 reps 1000
N 1000000 <T>: 0.570085 sec B: 0.002 GB/sec F:0.004 GFLOPS/sec
N 300000000 reps 20
N 300000000 <T>: 172.960330 sec B: 0.002 GB/sec F:0.003 GFLOPS/sec
```

c5.

```
N 1000000 reps 1000
N 1000000 <T>: 0.000329 sec B: 3.044 GB/sec F:6.087 GFLOPS/sec
N 300000000 reps 20
N 300000000 <T>: 0.190156 sec B: 1.578 GB/sec F:3.155 GFLOPS/sec
```

Q1.

When computing the mean, the first half of the measurements may experience greater fluctuation and require more time to complete. This is because, initially, there is no cache memory, and the system must load each data point. However, after the first half, the system will have cache memory, resulting in less time needed to complete the calculations, and less fluctuation in the time required for each result. Therefore, using the second half of the measurements to compute the mean could yield more accurate data, based on the experimentation already conducted in the first half.

Q2.

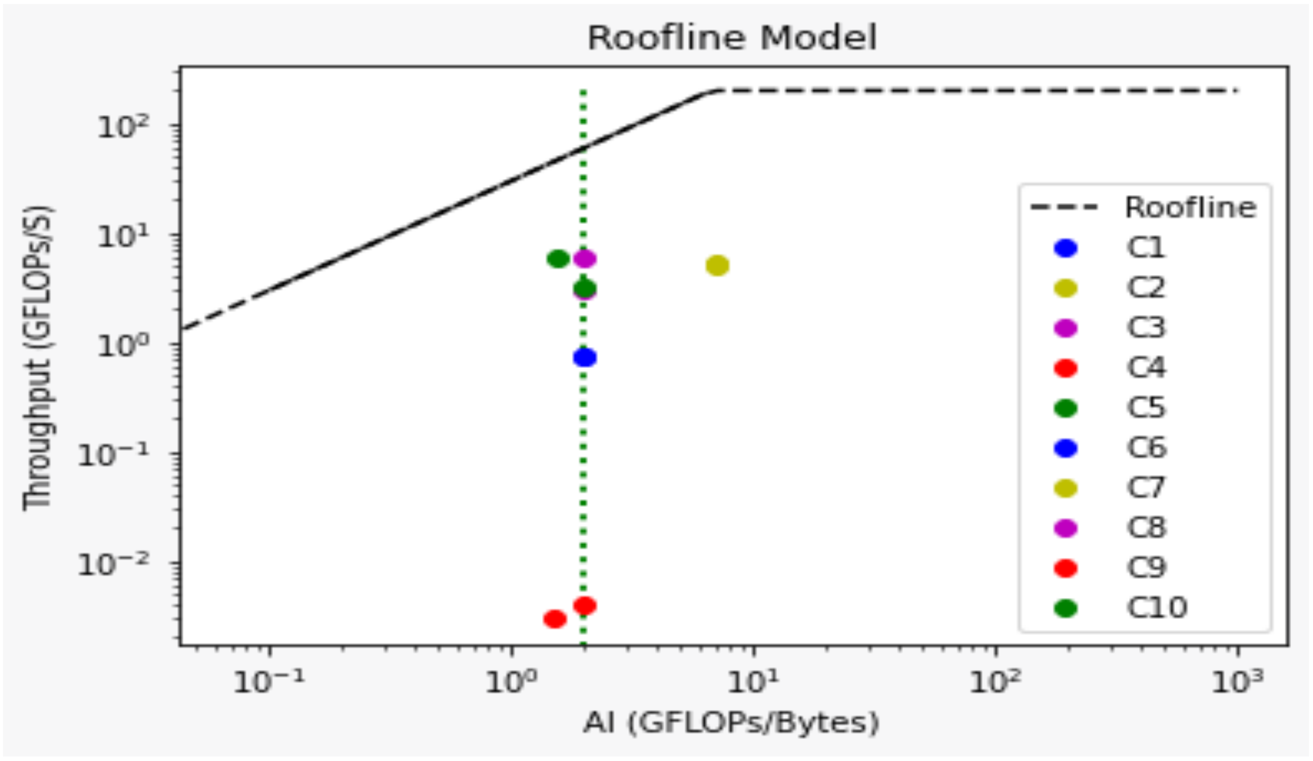
Roofline model for the given data and greene compute node with CPU configuration as

- *200 GFLOPS and 30 GB/s.*
- *Intel(R) Xeon(R) CPU E5-2690 v2 @3.00GHz*
- *Allocated RAM 16GB*

N	iterations	Meantime	bandwidth	throughput	program	Arithmetic intensity
1000000	1000	0.002614	0.383	0.765	dp1	1.997389034
1000000	1000	0.001306	0.766	5.36	dp2	6.997389034
1000000	1000	0.00033	3.028	6.056	dp3	2
1000000	1000	0.570085	0.002	0.004	dp4	2
1000000	1000	0.000329	3.044	6.087	dp5	1.999671485

300000000	20	0.803314	0.373	0.747	dp1	2.002680965
300000000	20	0.416456	0.72	5.043	dp2	7.004166667
300000000	20	0.19564	1.533	3.067	dp3	2.000652316
300000000	20	172.96033	0.002	0.003	dp4	1.5
300000000	20	0.190156	1.578	3.155	dp5	1.999366286

Plotted graph



Q3.

Comparison of DP1 and DP2

When comparing the DP1 and DP2 codes, it becomes apparent that DP1 runs N*Reps times, while DP2 only runs N*Reps/4 times, resulting in a smaller number of loops for DP2. As a result, DP2 takes significantly less time to complete while maintaining the same total number of floating-point operations as DP1. This leads to an increase in both FLOPS/sec and bandwidth, ultimately improving the performance of DP2 in comparison to DP1.

```
dp1
n:300000000 reps:20
N: 300000000 <T>: 0.803314 sec B: 0.373 GB/sec F: 0.747 GFLOP/sec
```

```
dp2
n:300000000 reps:20
N: 300000000 <T>: 0.416456 sec B: 0.720 GB/sec F: 5.043 GFLOP/sec
```

- Comparison for DP1 and DP3

While comparing the DP1 and Dp3 codes, we can see that DP1 runs for N*Reps times while the DP3 runs with optimizations and kernel acceleration as it uses MKL Library. We get improvement in Flops and Bandwidth due to MKL Library which uses BLAS optimization internally

```
dp1
n:300000000 reps:20
N: 300000000 <T>: 0.803314 sec B: 0.373 GB/sec F: 0.747 GFLOP/sec
```

```
dp3
n:300000000 reps:20
N: 300000000 <T>: 0.195640 sec B: 1.533 GB/sec F: 3.067 GFLOP/sec
```

Q3.

The absence of binary representations for floating points can negatively impact CPU calculations, leading to imprecise and unexpected results with fluctuations. However, using the numpy.dot() function can optimize the code and increase precision, resulting in the expected output without any fluctuations. With the numpy.dot() function, calculations are performed element-wise, which improves the output compared to using loops that may produce unexpected results. While these effects may not be noticeable when working with smaller digits, using larger numbers and big datasets can result in significant impacts.

As per numpy, it uses BLAS library internally to optimize. The BLAS (Basic Linear Algebra Subprograms) are routines that provide standard building blocks for performing basic vector and matrix operations. The Level 1 BLAS perform scalar, vector and vector-vector operations.

Ref.

- https://netlib.org/blas/
- https://numpy.org/doc/stable/reference/generated/numpy.dot.html