HPML Lab 1

ar7996

HPC Node Configuration:

```
#SBATCH --job-name=hpml_5
#SBATCH --nodes=1
#SBATCH --cpus-per-task=1
#SBATCH --mem=8GB
```

C1:

```
N: 1000000      <T>: 0.001103 sec      B: 0.725 GB/sec        F: 1813466920.107 FLOP/sec
N: 300000000    <T>: 0.355160 sec      B: 0.676 GB/sec        F: 1689378678.385 FLOP/sec
```

C2:

```
N: 1000000      <T>: 0.000356 sec      B: 2.245 GB/sec        F: 5612560805.848 FLOP/sec
N: 300000000 <T>: 0.199578 sec         B: 1.203 GB/sec        F: 3006350857.469 FLOP/sec
```

C3:

```
N: 1000000      <T>: 0.000299 sec      B: 2.676 GB/sec        F: 6691171863.873 FLOP/sec
N: 300000000 <T>: 0.164592 sec         B: 1.458 GB/sec        F: 3645373707.372 FLOP/sec
```

C4:

```
N: 1000000      <T>: 0.301035 sec      B: 0.003 GB/sec        F: 6643755.677 FLOP/sec
N: 300000000 <T>: 91.053772 sec        B: 0.003 GB/sec        F: 6589512.850 FLOP/sec
```

C5:

```
N: 1000000      <T>: 0.000304 sec      B: 2.634 GB/sec        F: 6583892980.250 FLOP/sec
N: 300000000 <T>: 0.157002 sec         B: 1.529 GB/sec        F: 3821618645.048 FLOP/sec
```

Q1:

By using only, the second half of the measurements for the computation of mean, it will help in mitigating potential biases that could arise from any warm-up effects or variations in system load during initial repetitions. Moreover, second half of the repetitions is more likely to represent stabilized performance of the system, providing a more accurate estimation of performance metrics such as execution time, bandwidth, throughput, and arithmetic intensity.

Q2:

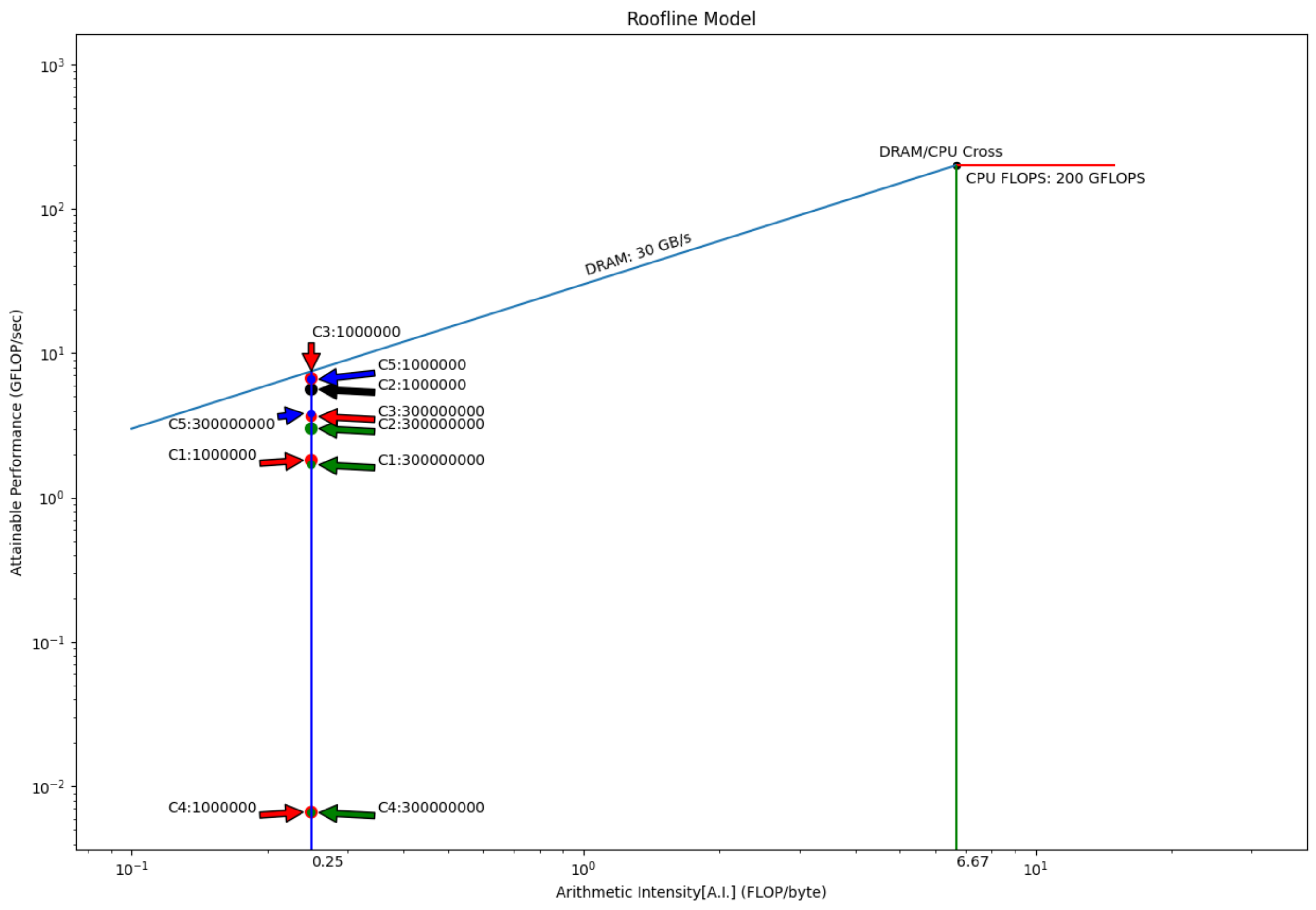
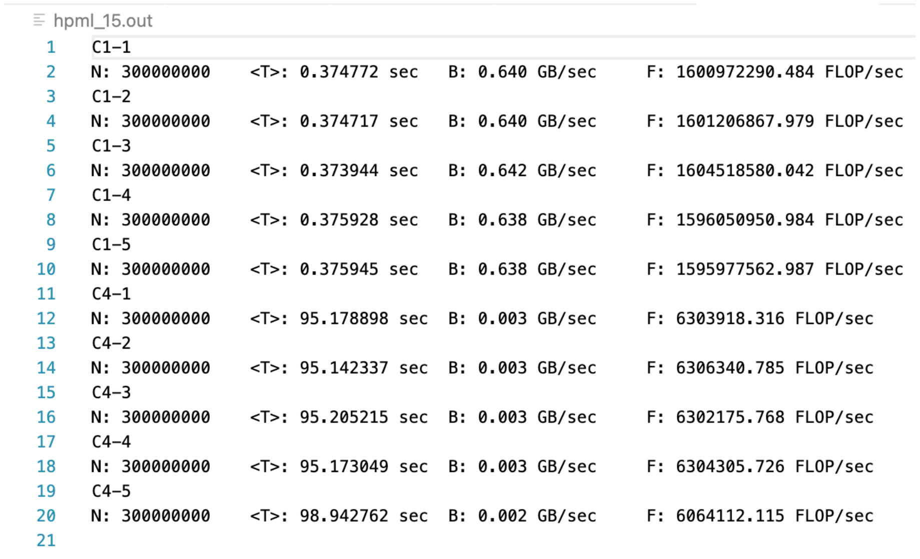


Figure 1: Roofline Model

Q3:



hpml_15.out

```
C1−1
N: 300000000    <T>: 0.374772 sec   B: 0.640 GB/sec     F: 1600972290.484 FLOP/sec
C1−2
N: 300000000    <T>: 0.374717 sec   B: 0.640 GB/sec     F: 1601206867.979 FLOP/sec
C1−3
N: 300000000    <T>: 0.373944 sec   B: 0.642 GB/sec     F: 1604518580.042 FLOP/sec
C1−4
N: 300000000    <T>: 0.375928 sec   B: 0.638 GB/sec     F: 1596050950.984 FLOP/sec
C1−5
N: 300000000    <T>: 0.375945 sec   B: 0.638 GB/sec     F: 1595977562.987 FLOP/sec
C4−1
N: 300000000    <T>: 95.178898 sec  B: 0.003 GB/sec     F: 6303918.316 FLOP/sec
C4−2
N: 300000000    <T>: 95.142337 sec  B: 0.003 GB/sec     F: 6306340.785 FLOP/sec
C4−3
N: 300000000    <T>: 95.205215 sec  B: 0.003 GB/sec     F: 6302175.768 FLOP/sec
C4−4
N: 300000000    <T>: 95.173049 sec  B: 0.003 GB/sec     F: 6304305.726 FLOP/sec
C4−5
N: 300000000    <T>: 98.942762 sec  B: 0.002 GB/sec     F: 6064112.115 FLOP/sec
```

We can see that in above results for single loop program of C(C1) and python(C4), performance of C is much better in all 5 measurements.

1. **Execution time:** Mean execution time of C variant is approx. 0.37 secs which is much less than that of python variant which is approx. 95.2 seconds.

2. **Bandwidth:** Calculated bandwidth of C variant is much higher than python variant. Bandwidth of C variant is approx. 0.64 GB/sec whereas that of python variant is approx. $0.3\text{x}10^{-2}$ GB/sec.

3. **Throughput:** Again, calculated throughput of C variant is much higher than python variant. Throughput of C variant is approx. 1.6 GFLOP/sec whereas that of python variant is approx. $0.6\text{x}10^{-2}$ GFLOP/sec.

Q4:

For two vector of size N and value "1.0", dot product is defined as

$$A \cdot B = \sum_{i=1}^{N} A[i] \times B[i], \qquad A[i], B[i] = 1.0, \forall i$$

Hence, value should be N.
But, in the C program of dot product using simple loop (or unrolled loop) we got the output as 16777216.000000 in case of simple loop and 67108864.000000 in case of unrolled loop for N = 300000000. This is because 32-bit floating points has precision up to a limit (23 bits) and hence,

when program reached to point where sum was 16777216.0 and tried to add 1.0 to it in next turn, value of the float variable remains same as 32-bit float was unable to represent 16777217.0. Similarly in the case of unrolled loop, 32-bit float was unable to represent 67108868.0 (67108864.0 + 4.0).

32-bit float has maximum value of 3.4028 … x 10^{38} but it does not mean that it can store all the values. Value of float is stored in terms of sign, exponent of 2 and significand(precision) and hence, there is gap in values that 32-bit float can store. As the value of exponent increase, gap increases.