CPU Benchmarking

Design Decisions

- Strong scaling is performed by having the problem size in our case 10^9 * 24 fixed and varying the number of threads to execute(1, 2, 4 ,8)
- 24 operations include addition, multiplication and division operations on integer / double precision floating point values are performed
- We have utilized - O2 optimization and ensured in assembly code that there are 24 operations being performed (can be verified in CPU/test.s assembly)
- Linpack bench mark being used is intel optimized binary and having a specific input given in CPU/linpack/lininput_xeon64

Improvements

- Incorporation of AVX instructions would be the future enhancement to the cpu benchmark
- Increasing the complexity of operations with maintaining data in cache would be the immediate improvement scope

Performance Results

Below tables represent the IOPS and FLOPS obtained by our benchmark and represent data over 3 consecutive runs of the tests.

1st run

	IOPS	FLOPS
Threads - 1	2.3586	0.9548
Threads - 2	4.5063	1.8192
Threads - 4	4.4828	1.8218
Threads - 8	4.4278	1.9086

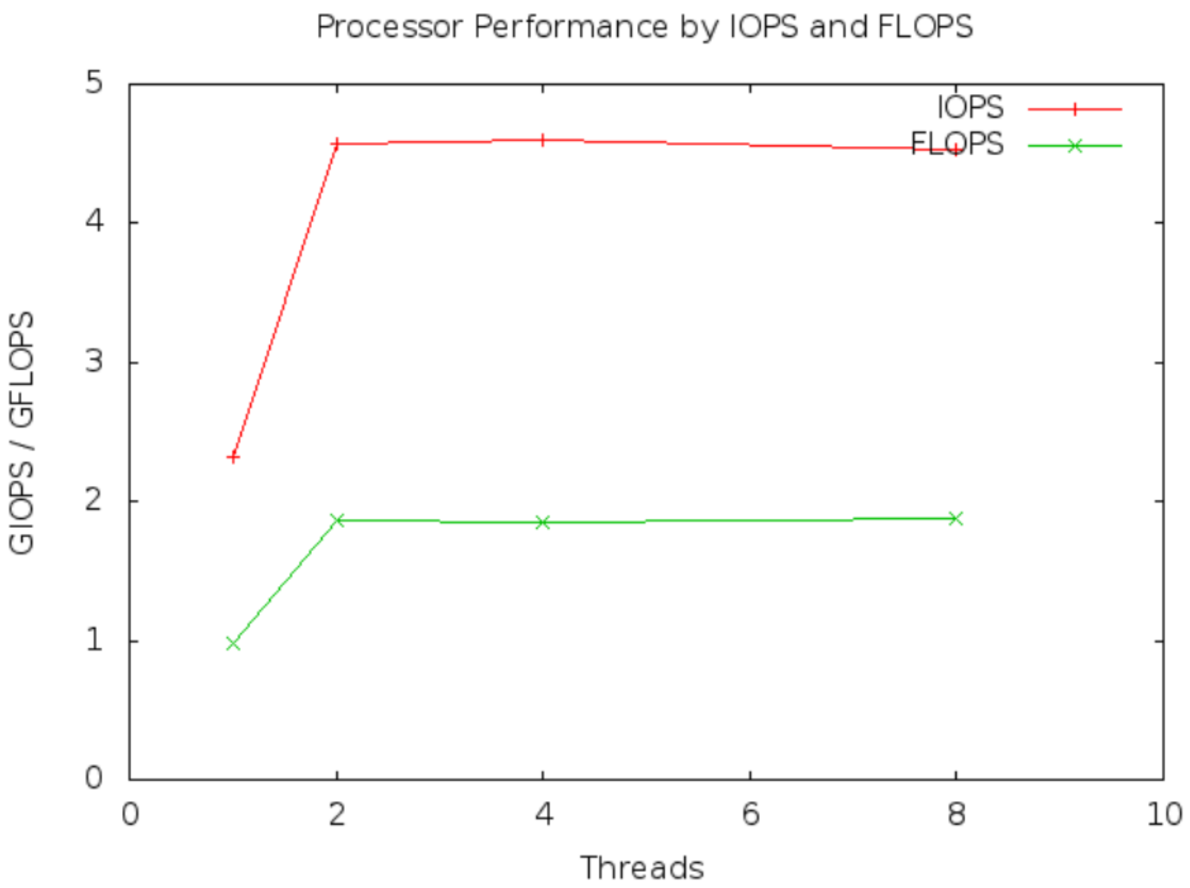
2nd run

	IOPS	FLOPS
Threads - 1	2.3182	0.9402
Threads - 2	4.5381	1.8644
Threads - 4	4.5645	1.8104
Threads - 8	4.4102	1.8044

3rd run

	IOPS	FLOPS
Threads - 1	2.2881	1.0423
Threads - 2	4.6758	1.9178
Threads - 4	4.7510	1.9064
Threads - 8	4.7415	1.9160

Performance Graph (average of three execution results)



Performance is seen to increase with the increased number of concurrency from 1 to 2 but as the concurrency is increased further it is seen that the performance impact has reduced. Here we can observe that the current hardware supports 2 CPU's with 1 core each because of which from the graphs we are able to conclude that the CPU performance tend to increase with varying number of concurrency till the number of thread count has reached equivalent the number of available CPU's

Theoretical Peak Performance

a) Intel Xeon E312xx (sandy bridge)
   CPU speed: 2.3 GHz
   Number of Cores: 1
   Number of CPU's: 2
   Instructions per cycle: 8 CPI

Theoretical Peak Performance = (CPU Speed * No. of cores * Instructions per cycle * No. CPU's)

= 2.3 * 1 * 8 * 2
= 36.8

Efficiency compared to theoretical performance = (GFLOPS / Theoretical Peak Performance) * 100
= (1.9 / 36.8) * 100
= 5.16 %

b) Intel(r) Xeon (r) CPU E5-2670 v3 (formerly Haswell) (baremetal)
   CPU speed: 2.3 GHz
   Number of Cores: 12
   Number of CPU's: 2
   Instructions per cycle: 16 CPI

Theoretical Peak Performance = (CPU Speed * No. of cores * Instructions per cycle * No. CPU's)

= 2.3 * 12 * 16 * 2
= 883.2

Linpack Bechmark Report (Intel Optimized Binary)

```
Fri Oct  6 17:01:15 UTC 2017
Intel(R) LINPACK data

Current date/time: Fri Oct  6 17:01:15 2017

CPU frequency:    2.234 GHz
Number of CPUs: 2
Number of cores: 2
Number of threads: 2

Parameters are set to:

Number of tests                             : 1
Number of equations to solve (problem size) : 20000
Leading dimension of array                  : 20000
Number of trials to run                     : 2
Data alignment value (in Kbytes)            : 4

Maximum memory requested that can be used = 3200404096, at the size = 20000

============= Timing linear equation system solver =================

Size   LDA    Align. Time(s)    GFlops   Residual     Residual(norm)
20000  20000  4      143.920    37.0632  4.097986e-10 3.627616e-02
20000  20000  4      141.807    37.6155  4.097986e-10 3.627616e-02

Performance Summary (GFlops)

Size   LDA    Align.  Average  Maximal
20000  20000  4       37.3394  37.6155

End of tests

Fri Oct  6 17:06:47 UTC 2017
```

a) Efficiency compared to theoretical performance

Efficiency = (GFLOPS / Theoretical Peak Performance) * 100
= (37.3 / 36.8) * 100
= 101 %

b) Efficiency of my code with respect to linpack

Efficiency = (GFLOPS / linpack GFlops) * 100
= (1.9 / 37.3) * 100
= 5.09 %