

## HPC LAB - Vector Addition

**Name:** Paleti Krishnasai

**Roll No:** CED18I039

**Programming Environment:** OpenMP

**Problem:** Vector Addition

**Date:** 19th August 2021

### Hardware Configuration:

PU NAME: Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz

Number of Sockets: 1

Cores per Socket : 4

Threads per core: 2

L1d cache: 128 KiB

L1i cache: 128 KiB

L2 cache: 1 MiB


L3 cache: 8 MiB

CPU-Z - ID : hzkq5i

CPU | Caches | Mainboard | Memory | SPD | Graphics | Bench | About

Processor

Name	Intel Core i5 8300H		
Code Name	Coffee Lake	Max TDP	45 W
Package	Socket 1440 FCBGA		
Technology	14 nm	Core Voltage	1.08 V



Specification Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz

Family	6	Model	E	Stepping	A
Ext. Family	6	Ext. Model	9E	Revision	U0

Instructions MMX, SSE, SSE2, SSE3, SSE4.1, SSE4.2, EM64T, VT-x, AES, AVX, AVX2, FMA3

Clocks (Core #0)

Core Speed	3886.66 MHz
Multiplier	x 39.0 ( 8 - 40 )
Bus Speed	99.66 MHz
Rated FSB	

Cache

L1 Data	4 x 32 KBytes	8-way
L1 Inst.	4 x 32 KBytes	8-way
Level 2	4 x 256 KBytes	4-way
Level 3	8192 KBytes	16-way

Selection Processor #1 Cores 4 Threads 8

CPU-Z Ver. 1.94.0.x64 Tools Validate Close

```
paleti@paleti-Lenovo-ideapad-330-15ICH:~$ lscpu
Architecture:                x86_64
CPU op-mode(s):              32-bit, 64-bit
Byte Order:                  Little Endian
Address sizes:               39 bits physical, 48 bits virtual
CPU(s):                      8
On-line CPU(s) list:        0-7
Thread(s) per core:         2
Core(s) per socket:         4
Socket(s):                   1
NUMA node(s):               1
Vendor ID:                   GenuineIntel
CPU family:                  6
Model:                       158
Model name:                  Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz
Stepping:                    10
CPU MHz:                     900.021
CPU max MHz:                 4000.0000
CPU min MHz:                 800.0000
BogoMIPS:                    4599.93
Virtualization:              VT-x
L1d cache:                   128 KiB
L1i cache:                   128 KiB
L2 cache:                    1 MiB
L3 cache:                    8 MiB
NUMA node0 CPU(s):          0-7
Vulnerability Itlb multihit: KVM: Mitigation: VMX disabled
Vulnerability L1tf:          Mitigation; PTE Inversion; VMX conditional cache flushes, SMT vulnerable
Vulnerability Mds:           Mitigation; Clear CPU buffers; SMT vulnerable
Vulnerability Meltdown:      Mitigation; PTI
Vulnerability Spec store bypass: Mitigation; Speculative Store Bypass disabled via prctl and seccomp
Vulnerability Spectre v1:    Mitigation; usercopy/swapgs barriers and __user pointer sanitization
Vulnerability Spectre v2:    Mitigation; Full generic retpoline, IBPB conditional, IBRS_FW, STIBP conditional, RSB filling
Vulnerability Srbds:         Mitigation; Microcode
Vulnerability Tsx async abort: Not affected
```

Serial Code :

```
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>
#include <time.h>
#define n 100000

int main()
{
    double a[n],b[n], c[n], random_a, random_b;
    float startTime, endTime,execTime;
    int i;
    srand(time(0));

    startTime = omp_get_wtime();

    for(i=0;i<n;i++)
    {
        random_a = rand() , random_b = rand();
        a[i] = i * random_a;
        b[i] = i * random_b;
        for(int j=1;j<n;j++)
            c[i] = a[i] + b[i];
    }

    endTime = omp_get_wtime();

    execTime = endTime - startTime;
    printf("%f \n",execTime);
    return(0);
}
```

```
paleti@paleti-Lenovo-ideapad-330-15ICH:~/Documents/SEM 7/HPC_Linux/Week1$ gcc -fopenmp vector_add_serial.c
paleti@paleti-Lenovo-ideapad-330-15ICH:~/Documents/SEM 7/HPC_Linux/Week1$ ./a.out
31.302734
paleti@paleti-Lenovo-ideapad-330-15ICH:~/Documents/SEM 7/HPC_Linux/Week1$
```

Parallel Code :

```
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>
#include <time.h>
#define n 100000

int main()
{
    double a[n], b[n], c[n], random_a, random_b;
    float startTime, endTime, execTime;
    int i;
    int omp_rank;
    srand(time(0));

    startTime = omp_get_wtime();
    #pragma omp parallel private (i) shared (a,b,c)
    {
        #pragma omp for
        for(i=0; i<n; i++)
        {
            random_a = rand() , random_b = rand();
            omp_rank = omp_get_thread_num();
            a[i] = i * random_a;
            b[i] = i * random_b;
            for(int j=1; j<n; j++)
            {
                c[i] = a[i] + b[i];
                //printf("The value of a[%d] = %lf and b[%d] = %lf and result
c[%d] = %lf Thread rank = %d\n", i, a[i], i, b[i], i, c[i], omp_rank);
            }
        }

        endTime = omp_get_wtime();

        execTime = endTime - startTime;
        printf("%f \n", execTime);
        return(0);
    }
}
```

### Compilation and Execution:

To enable OpenMP environment, use -fopenmp flag while compiling using gcc.

```
gcc -O0 -fopenmp vector_add_mp.c
```

Then,

```
export OMP_NUM_THREADS= no of threads for parallel execution.
```

```
./a.out
```

### Observations :

Threads ( n )	Runtime	Speedup ( s )	Parallelization Fraction
1	33.208984	1	
2	17.189453	1.931939545	0.9647709186
4	9.169922	3.621512157	0.9651630013
6	10.953125	3.031918653	0.8042110171
8	10.042969	3.306689884	0.7972374498
10	9.253906	3.588645054	0.8014925519
12	10.353516	3.207507865	0.7507979714
16	9.394531	3.534927289	0.7649159998
20	11.507812	2.885777418	0.6878662397
32	10.826172	3.067472418	0.6957405922
64	10.410156	3.190056326	0.6974231479
128	10.410156	3.190056326	0.691931627

Speed up can be found using the following formula,

$$S(n)=T(1)/T(n)$$

where, **S(n)** = Speedup for thread count 'n'

**T(1)** = Execution Time for Thread count '1' (serial code)

**T(n)** = Execution Time for Thread count 'n' (serial code)

Parallelization Fraction can be found using the following formula,

$$S(n) = 1 / ((1 - p) + p/n)$$

where,  $S(n)$  = Speedup for thread count 'n'

$n$  = Number of threads

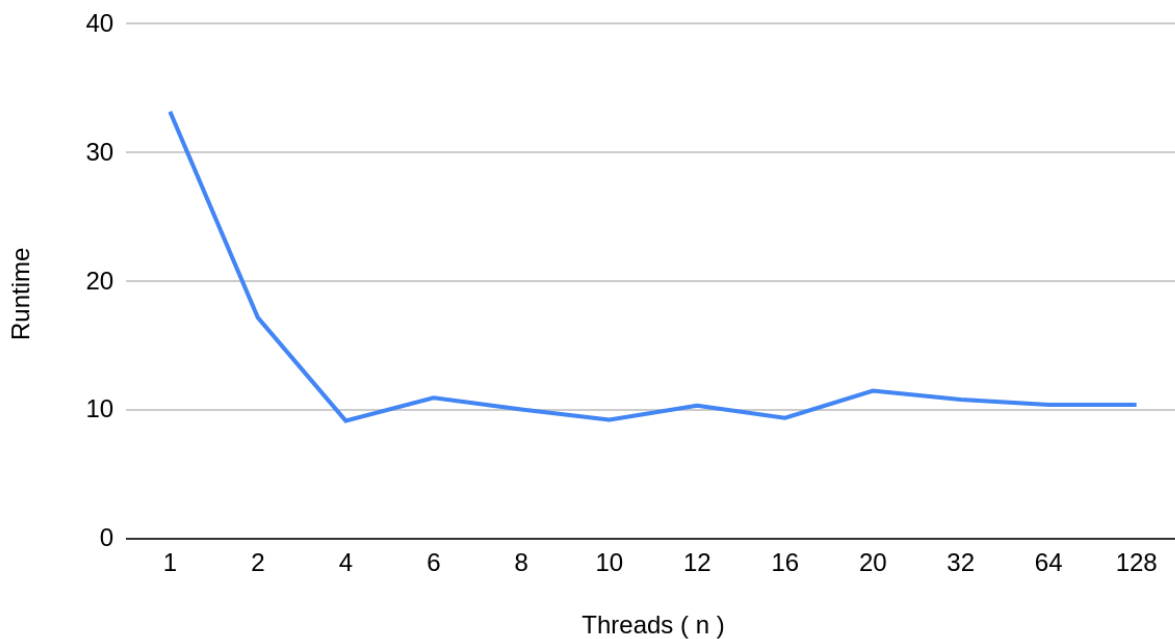
$p$  = Parallelization fraction

### Assumption:

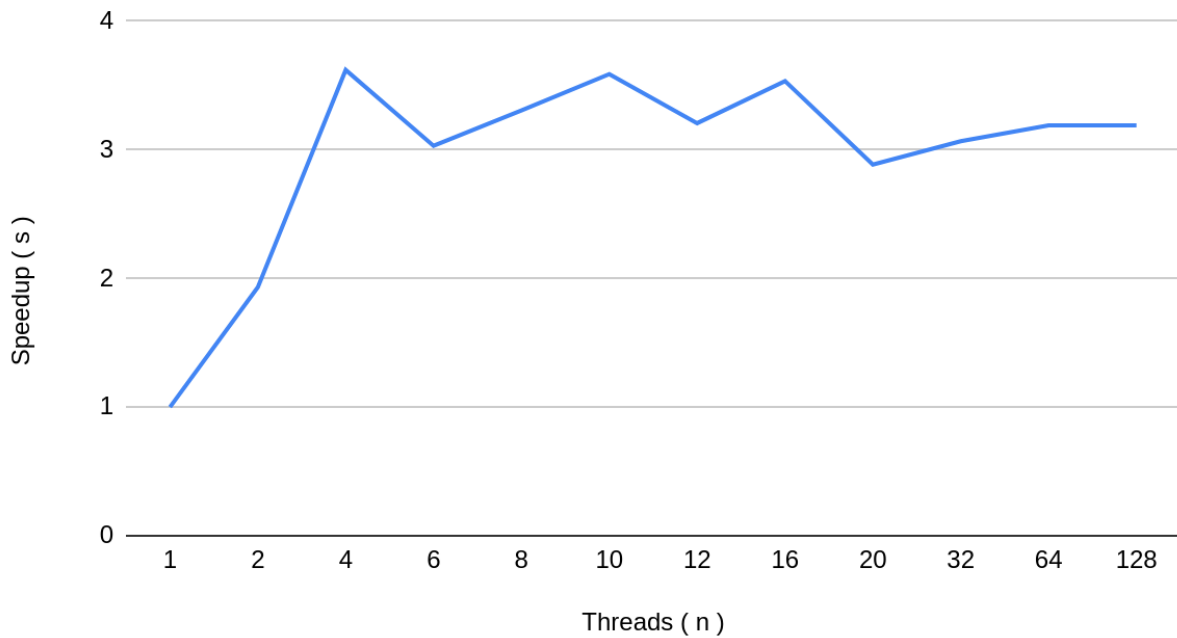
Following extra for loop is added to increase the number of operations in the parallel region to visualize the effect of multi-threading in vector addition.

```
for(int j=0;j<m;j++)  
    c[i] = a[i] + b[i];
```

Runtime vs. Threads ( n )



Speedup ( s ) vs. Threads ( n )



**Inference:** (**Note:** Execution time, graph and inference will be based on hardware configuration)

- At thread count 4,10,16 maximum speedup is observed
- The Runtime is least at thread count is 4, but it fluctuates from there on rising slightly. This may be due to the fact that there are 4 cores and only 1 thread on each core is active at a moment for this task.