# HPC Experiment 6 Report

Vector Dot Product

**Name:** Reuben Skariah Mathew
**Roll No:** CED18I042
**Programming Environment:** OpenMP
**Problem:** Vector Dot Product
**Date:** 26th August 2021

**Hardware Configuration:**
**Processor:** Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz
**Sockets:** 1
**Cores per Socket:** 4
**Threads per Core:** 2
**L1 Cache:** 32 kB
**L2 Cache:** 256 kB
**L3 Cache:** 6 MB
**RAM:** 8 GB

**Serial Code:**

```
#include <stdio.h>
#include <omp.h>

#define n 50000
#define delay 50000

int main()
{
    double a[n], b[n], c[n], runtime;
    float startTime,endTime;
    int i;
    double dot;
    dot=0.0;
    startTime = omp_get_wtime();
    for(i=0;i<n;i++)
    {
        a[i] = (float) i * 5.52;
        b[i] = (float) i * 3.23;
        c[i] = 0.0;
            for(int j=0;j<delay;j++)
                c[i] += a[i] * b[i];
        dot += c[i];
    }
    endTime = omp_get_wtime();
    runtime = endTime - startTime;
    printf("\n\nRun Time: %f", runtime);
    return 0;
}
```

**Parallel Code:**

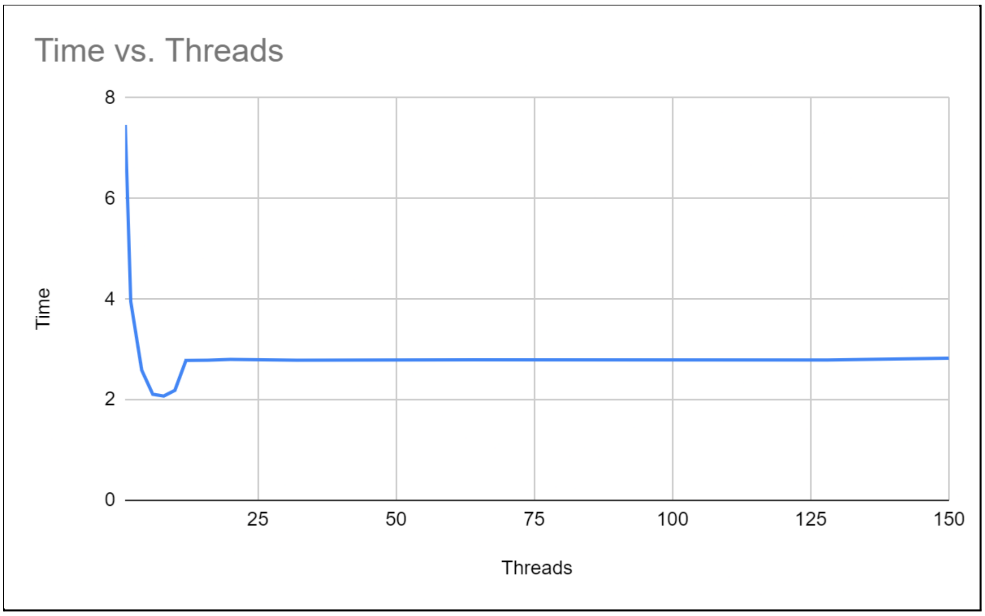
1. **Reduction**

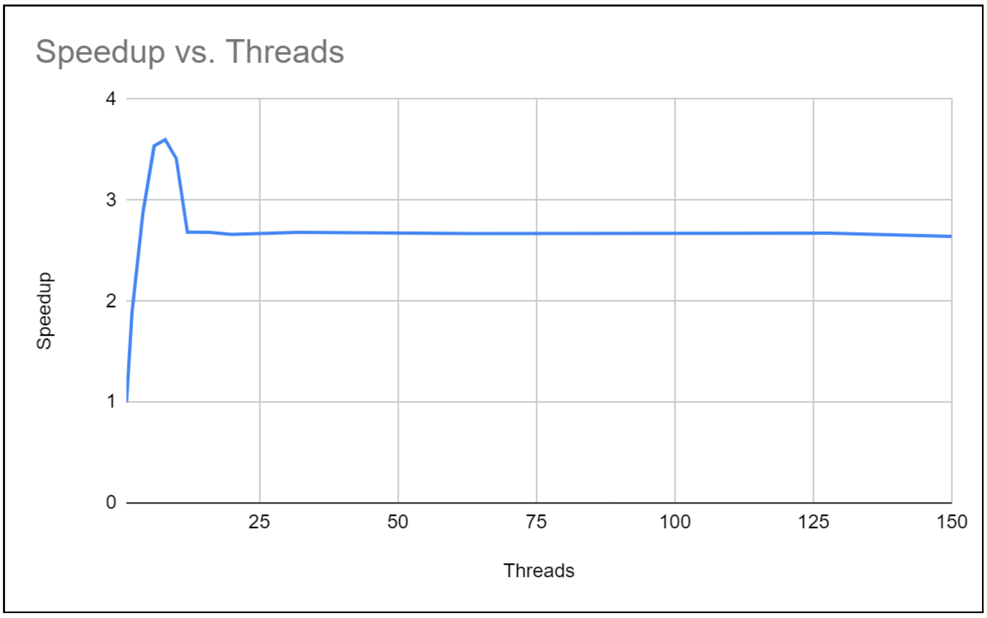
```
#include <stdio.h>
#include <omp.h>

#define n 50000
#define delay 50000

int main()
{
    double a[n], b[n], c[n], runtime[13];
    float startTime,endTime;
    int i,k,omp_rank;
    double dot;
    int threads[]={1, 2, 4, 6, 8, 10, 12, 16, 20, 32, 64, 128, 150};
    for(k=0;k<13;k++)
    {
        dot=0.0;
        omp_set_num_threads(threads[k]);
        startTime = omp_get_wtime();
        #pragma omp parallel private(i)
        {
            #pragma omp for reduction (+:dot)
            for(i=0;i<n;i++)
            {
                omp_rank = omp_get_thread_num();
                a[i] = (float) i * 5.52;
                b[i] = (float) i * 3.23;
                c[i] = 0.0;
                    for(int j=0;j<delay;j++)
                        c[i] += a[i] * b[i];
                dot += c[i];
            }
        }
        endTime = omp_get_wtime();
        runtime[k] = endTime - startTime;
    }
    for(k=0;k<13;k++)
    printf("\n\nThread Count: %d      Run Time: %f",threads[k], runtime[k]);
    return 0;
}
```

| **Threads** | **Time** | **Speedup** |
|---:|---:|---:|
| 1 | 7.463867 | 1 |
| 2 | 3.953125 | 1.888092838 |
| 4 | 2.590088 | 2.881704019 |
| 6 | 2.109863 | 3.537607418 |
| **8** | **2.073242** | **3.600094441** |
| 10 | 2.187256 | 3.412434118 |
| 12 | 2.781982 | 2.68293145 |
| 16 | 2.78418 | 2.680813381 |
| 20 | 2.803711 | 2.662138501 |
| 32 | 2.78418 | 2.680813381 |
| 64 | 2.795654 | 2.669810713 |
| 128 | 2.791016 | 2.674247299 |
| 150 | 2.824951 | 2.642122642 |





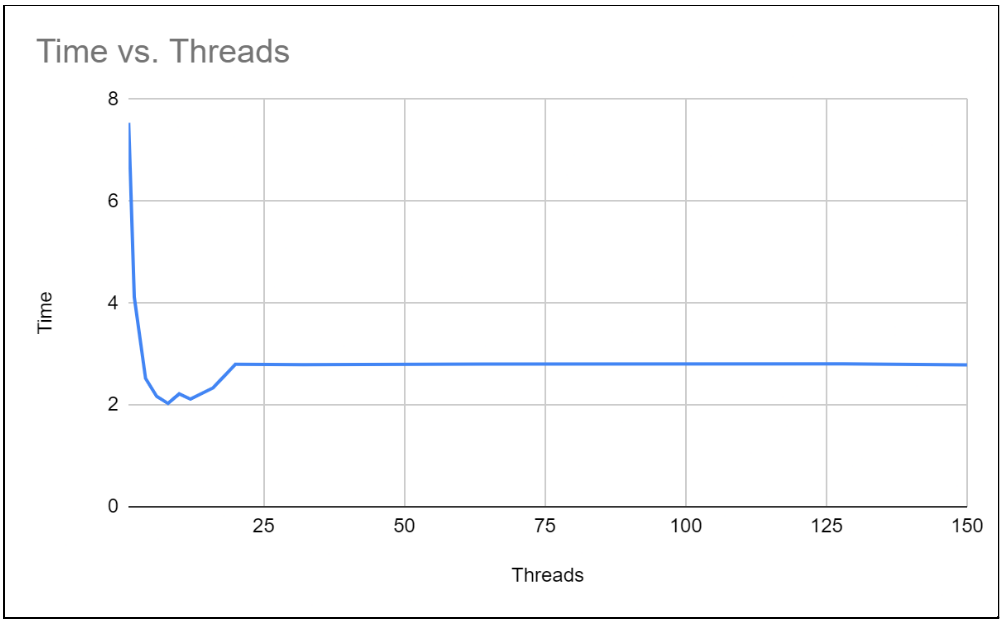
## 2. Critical Selection

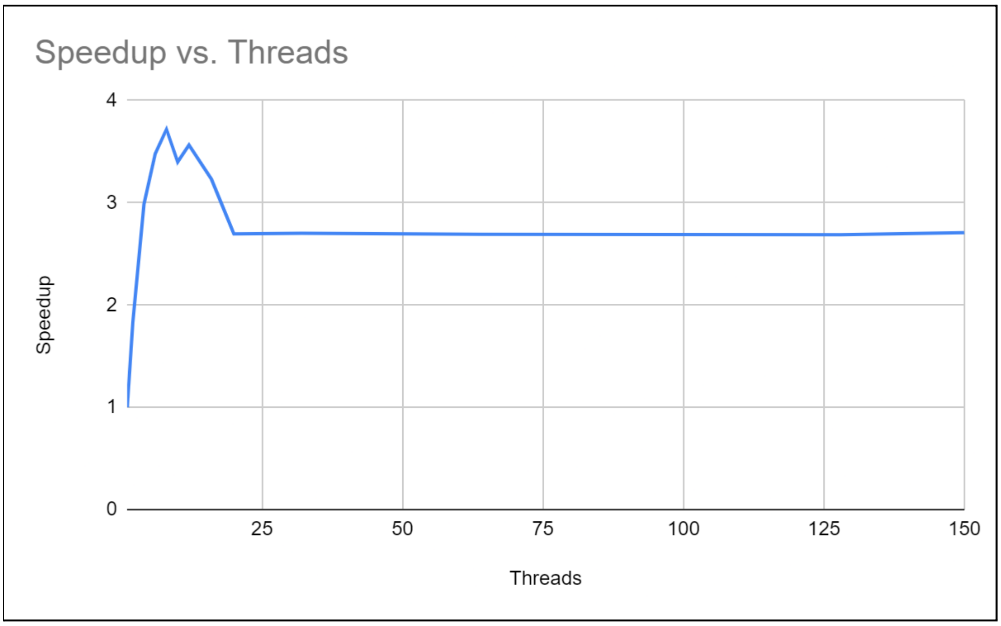
```
#include <stdio.h>
#include <omp.h>

#define n 50000
#define delay 50000

int main()
{
    double a[n], b[n], c[n], runtime[13];
    float startTime,endTime;
    int i,k,omp_rank;
    double dot, fdot;
    int threads[]={1, 2, 4, 6, 8, 10, 12, 16, 20, 32, 64, 128, 150};
    for(k=0;k<13;k++)
    {
        dot=0.0;
        omp_set_num_threads(threads[k]);
        startTime = omp_get_wtime();
        #pragma omp parallel private(i)
        {
            #pragma omp for
            for(i=0;i<n;i++)
            {
                omp_rank = omp_get_thread_num();
                a[i] = (float) i * 5.52;
                b[i] = (float) i * 3.23;
                c[i] = 0.0;
                    for(int j=0;j<delay;j++)
                        c[i] += a[i] * b[i];
                dot += c[i];
            }
            #pragma omp critical(finaldot)
            fdot += dot;
        }
        endTime = omp_get_wtime();
        runtime[k] = endTime - startTime;
    }
    for(k=0;k<13;k++)
    printf("\n\nThread Count: %d      Run Time: %f",threads[k], runtime[k]);
    return 0;
}
```

| Threads | Time | Speedup |
|---|---|---|
| 1 | 7.541016 | 1 |
| 2 | 4.117188 | 1.831593797 |
| 4 | 2.522949 | 2.988968861 |
| 6 | 2.166992 | 3.479946396 |
| **8** | **2.02832** | **3.717863059** |
| 10 | 2.218262 | 3.399515477 |
| 12 | 2.114746 | 3.565920446 |
| 16 | 2.333984 | 3.230963023 |
| 20 | 2.797852 | 2.695287671 |
| 32 | 2.791016 | 2.701889205 |
| 64 | 2.802246 | 2.691061384 |
| 128 | 2.804688 | 2.688718317 |
| 150 | 2.785645 | 2.707098715 |





**Inference:**

- Maximum speedup was observed at thread count equal to 8 in both reduction and critical selection method.

- The speedup increased from thread count 1 to 8 then tapered off as it increased further.

- Overall the run time for critical selection method and reduction method were similar for n and delay both equal to 50000.