CS 301
High-Performance Computing

Assignment 1

Lukhi Krunalkumar (201901449)
Dhwanil Shah (201901450)

February 12, 2022

Contents

1 Introduction

2 Hardware Details

2.1 System 1: Lab-PC

- CPU - 4
- Socket - 1
- Cores per Socket - 4
- Size of L1d cache - 32K
- Size of L1i cache - 32K
- Size of L2 cache - 256K
- Size of L3 cache - 6144K

2.2 System 2: Cluster (gics1)

- CPU - 16
- Socket - 2
- Cores per Socket - 8
- Size of L1d cache - 32K
- Size of L1i cache - 32K
- Size of L2 cache - 256K
- Size of L3 cache - 20480K

2.3 System 3: Local PC

- CPU - 4
- Socket - 1
- Cores per Socket - 2
- Size of L1d cache - 64K
- Size of L1i cache - 64K
- Size of L2 cache - 512K
- Size of L3 cache - 3000K

3 Graphical Analysis of Throughput

3.1 Compute Throughput

1. Copy ($a[i] = b[i]$)

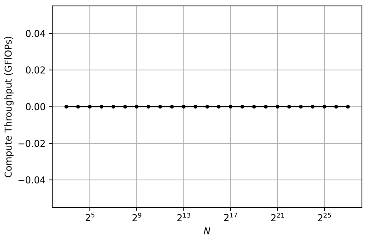

Figure 1: Local PC

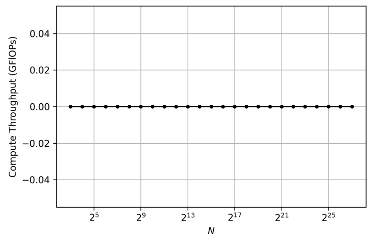

Figure 2: Lab PC

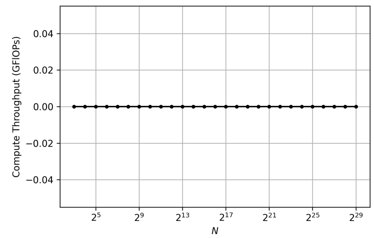

Figure 3: Cluster

2. Scale ($a[i] = q * b[i]$)

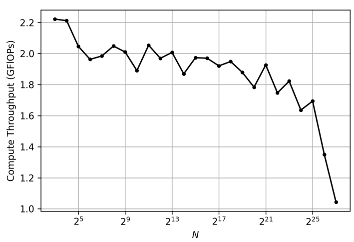

Figure 4: Local PC

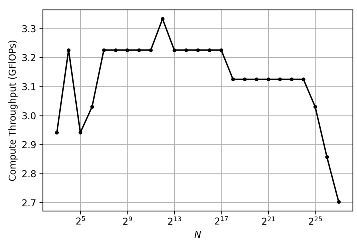

Figure 5: Lab PC

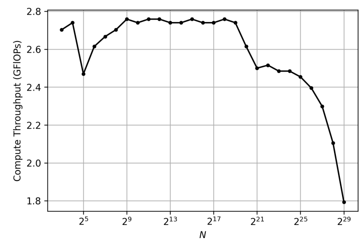

Figure 6: Cluster

3. Sum ($a[i] = b[i] + c[i]$)

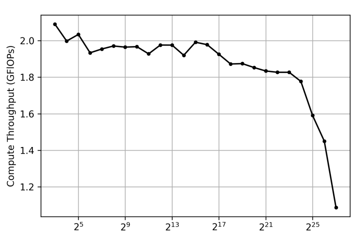

Figure 7: Local PC

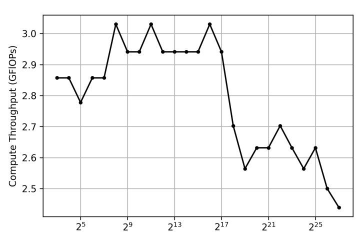

Figure 8: Lab PC

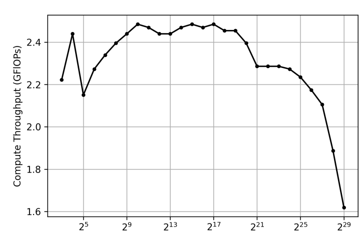

Figure 9: Cluster

4. Triad $(a[i] = b[i] + q * c[i])$

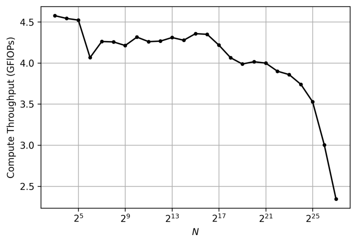

Figure 10: Local PC

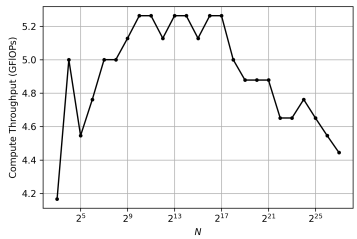

Figure 11: Lab PC

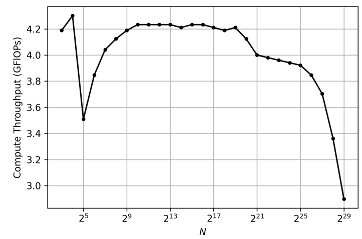

Figure 12: Cluster

5. Vector Triad $(a[i] = b[i] + c[i] * d[i])$

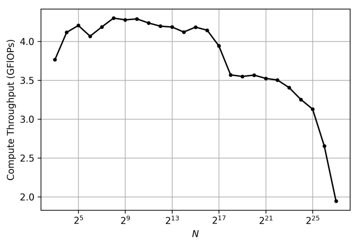

Figure 13: Local PC

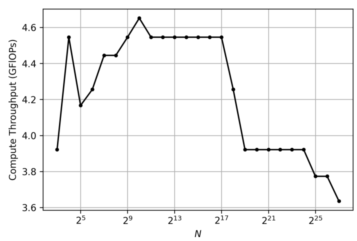

Figure 14: Lab PC

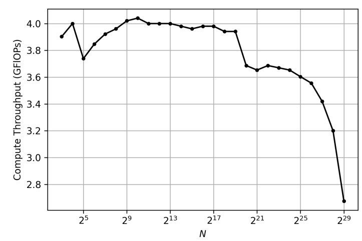

Figure 15: Cluster

3.2 Memory Throughput

1. Copy $(a[i] = b[i])$

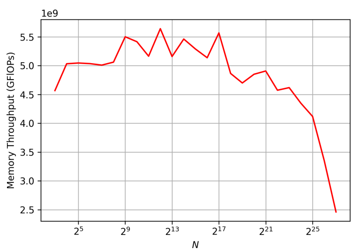

Figure 16: Local PC

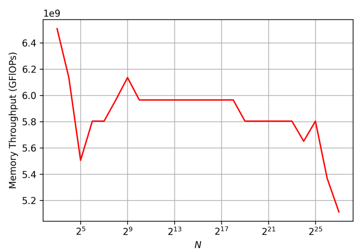

Figure 17: Lab PC

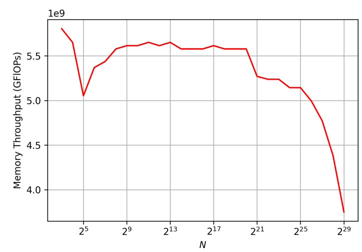

Figure 18: Cluster

2. Scale $(a[i] = q * b[i])$

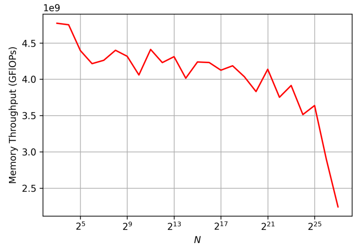

Figure 19: Local PC

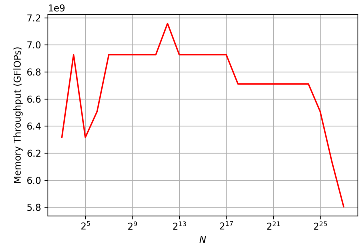

Figure 20: Lab PC

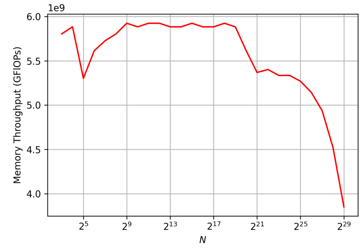

Figure 21: Cluster

3. Sum $(a[i] = b[i] + c[i])$

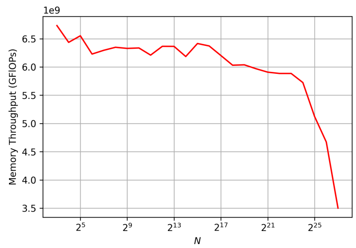

Figure 22: Local PC

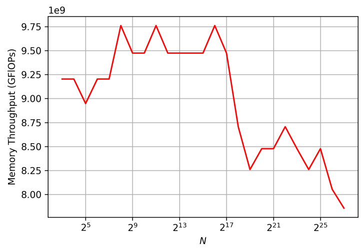

Figure 23: Lab PC

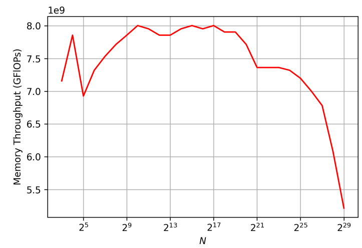

Figure 24: Cluster

4. Triad $(a[i] = b[i] + q * c[i])$

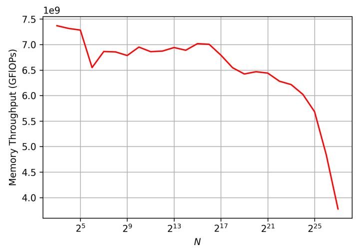

Figure 25: Local PC

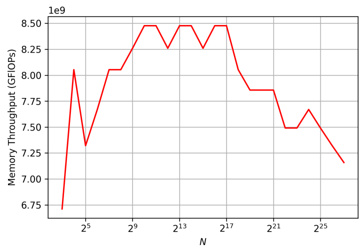

Figure 26: Lab PC

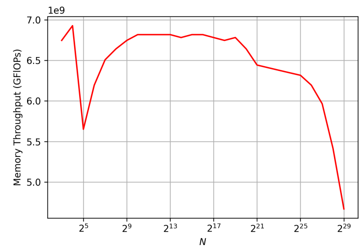

Figure 27: Cluster

5. Vector Triad ($a[i] = b[i] + c[i] * d[i]$)

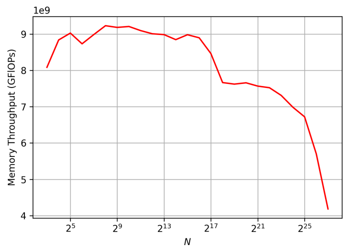

Figure 28: Local PC

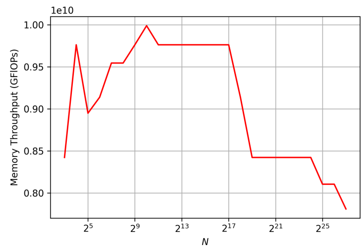

Figure 29: Lab PC

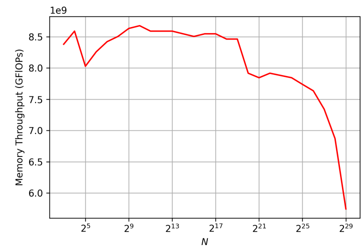

Figure 30: Cluster

3.3 Explanation

The graph of computation throughput is expected to be linearly decreasing if we consider an ideal situation because as the number of iterations increase the throughput should decrease. But practically, it is a different picture. The trends in graph can be understood in following parts:

1. Initial increasing nature of graph is mainly due to use pipelining function.
2. Now, the amount of the computation is large enough such that memory access stage of pipeline starts affecting the throughput. Here, size of L1 cache matters. Since the capacity of L1 cache becomes full, the memory access takes place from the L2 cache and this leads to a decrease in throughput.
3. If we increase the number of computation further($\approx 2^{16}$), L2 cache gets filled completely. Hence, time required to access data from here increases which leads to decrease in throughput.
4. Final drop is observed when number of computations are nearly 2^{21}. This happen when L3 cache is full. As there is no cache memory after L3, we observe sudden drop in throughput. The L3 cache of cluster is largest of them all. So, drop in cluster happen later than others.

In memory access throughput graph, we can see similar behaviour to compute throughput graph, as number of memory access are few time more than compute throughput. Hence, memory access graphs are scaled version of compute throughput graph.

Also, throughput of code involving multiplication operation is found to be lesser than addition. This is because, multiplication is expensive operation while addition is basic operation.

3.4 Peak Performance Metrics

The peak performance for a system can be calculated using the following formula
Peak Performance (GFLOPS) = (CPU speed (GHz)) $\times$ (number of CPU's per node) $\times$ (number of cores per CPU) $\times$ (number of instructions per cycle)

1. Cluster Peak Performance $= 1.4 \times 16 \times 8 \times 4 = 716.8$ GFLOPS.
2. Local PC Peak Performance $= 2.2 \times 2 \times 4 \times 4 = 70.4$ GFLOPS.
3. Lab PC Peak Performance $= 2.4 \times 4 \times 4 \times 4 = 153.6$ GFLOPS.

4 Vtune Profiling

```
ysis will be limited to kernel symbol tables. See the Enabling Linux Kernel Analysis topic in the product online help for instructions.
vtune: Executing actions 75 % Generating a report                              Elapsed Time: 13.994s
    CPI Rate: 0.383
    Average CPU Frequency: 3.690 GHz
    Total Thread Count: 1
Effective CPU Utilization: 23.8%
 | The metric value is low, which may signal a poor logical CPU cores
 | utilization caused by load imbalance, threading runtime overhead, contended
 | synchronization, or thread/process underutilization. Explore sub-metrics to
 | estimate the efficiency of MPI and OpenMP parallelism or run the Locks and
 | Waits analysis to identify parallel bottlenecks for other parallel runtimes.
 |
    Average Effective CPU Utilization: 0.951 out of 4
Memory Bound: 41.9% of Pipeline Slots
 | The metric value is high. This can indicate that the significant fraction of
 | execution pipeline slots could be stalled due to demand memory load and
 | stores. Use Memory Access analysis to have the metric breakdown by memory
 | hierarchy, memory bandwidth information, correlation by memory objects.
 |
    Cache Bound: 41.3% of Clockticks
     | A significant proportion of cycles are being spent on data fetches from
     | caches. Check Memory Access analysis to see if accesses to L2 or L3
     | caches are problematic and consider applying the same performance tuning
     | as you would for a cache-missing workload. This may include reducing the
     | data working set size, improving data access locality, blocking or
     | partitioning the working set to fit in the lower cache levels, or
     | exploiting hardware prefetchers. Consider using software prefetchers, but
     | note that they can interfere with normal loads, increase latency, and
     | increase pressure on the memory system. This metric includes coherence
     | penalties for shared data. Check Microarchitecture Exploration analysis
     | to see if contested accesses or data sharing are indicated as likely
     | issues.
     |
    DRAM Bound: 9.1% of Clockticks
Collection and Platform Info
    Application Command Line: ./v5
    Operating System: 3.10.0-693.21.1.el7.x86_64 NAME="CentOS Linux" VERSION="7 (Core)" ID="centos" ID_LIKE="rhel fedora" VERSION_ID="7"
PRETTY_NAME="CentOS Linux 7 (Core)" ANSI_COLOR="0;31" CPE_NAME="cpe:/o:centos:centos:7" HOME_URL="https://www.centos.org/" BUG_REPORT_URL
```

Figure 31: Hpc performance using Vtune for vector triad

```
vtune: Executing actions 75 % Generating a report                              Elapsed Time: 13.066s
    CPU Time: 12.960s
        Effective Time: 12.960s
        Spin Time: 0s
        Overhead Time: 0s
    Total Thread Count: 1
    Paused Time: 0s

Top Hotspots
Function                    Module     CPU Time  % of CPU Time(%)
--------------------------  ---------  --------  ----------------
main                        v5          12.950s             99.9%
[Outside any known module]  [Unknown]    0.010s              0.1%
Effective CPU Utilization: 24.9%
 | The metric value is low, which may signal a poor logical CPU cores
 | utilization caused by load imbalance, threading runtime overhead, contended
 | synchronization, or thread/process underutilization. Explore sub-metrics to
 | estimate the efficiency of MPI and OpenMP parallelism or run the Locks and
 | Waits analysis to identify parallel bottlenecks for other parallel runtimes.
 |
    Average Effective CPU Utilization: 0.997 out of 4
Collection and Platform Info
    Application Command Line: ./v5
    Operating System: 3.10.0-693.21.1.el7.x86_64 NAME="CentOS Linux" VERSION="7 (Core)" ID="centos" ID_LIKE="rhel fedora" VERSION_ID="7"
PRETTY_NAME="CentOS Linux 7 (Core)" ANSI_COLOR="0;31" CPE_NAME="cpe:/o:centos:centos:7" HOME_URL="https://www.centos.org/" BUG_REPORT_URL
="https://bugs.centos.org/"  CENTOS_MANTISBT_PROJECT="CentOS-7" CENTOS_MANTISBT_PROJECT_VERSION="7" REDHAT_SUPPORT_PRODUCT="centos" REDHA
T_SUPPORT_PRODUCT_VERSION="7"
```

Figure 32: Hotspot Vtune profiling for vector triad