CS 301
High-Performance Computing

Lab 01 - CPU architecture, Triad and Measuring Performance

February 3, 2026

Contents

1 Introduction

In this experiment, our goal is to evaluate the CPU performance of the lab computer and the cluster system. To do this, we use the vector triad program along with its related operations—copy, scale, and sum. These operations were introduced earlier in theory, and this lab allows us to apply them in a practical setting. By running these programs, we can observe how the system architecture affects overall performance.

2 Hardware Details

2.1 Hardware Details for LAB207 PCs

- Architecture: x86_64
- CPU op-mode(s): 32-bit, 64-bit
- Byte Order: Little Endian
- CPU(s): 12
- On-line CPU(s) list: 0-11
- Thread(s) per core: 2
- Core(s) per socket: 6
- Socket(s): 1
- NUMA node(s): 1
- Vendor ID: GenuineIntel
- CPU family: 6
- Model: 151
- Model name: 12th Gen Intel(R) Core(TM) i5-12500
- Stepping: 5
- CPU max MHz: 4600.0000
- CPU min MHz: 800.0000
- BogoMIPS: 5990.40
- Virtualization: VT-x
- L1d cache: 288K
- L1i cache: 192K
- L2 cache: 7.5M

- L3 cache: 18M
- NUMA node0 CPU(s): 0-11
- Flags: fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc aperfmperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 fma cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm epb invpcid_single tpr_shadow vnmi flexpriority ept vpid fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms invpcid xsaveopt dtherm ida arat pln pts

2.2 Hardware Details for HPC Cluster (Node gics1)

- Architecture: x86_64
- CPU op-mode(s): 32-bit, 64-bit
- Byte Order: Little Endian
- CPU(s): 16
- On-line CPU(s) list: 0-15
- Thread(s) per core: 1
- Core(s) per socket: 8
- Socket(s): 2
- NUMA node(s): 2
- Vendor ID: GenuineIntel
- CPU family: 6
- Model: 63
- Model name: Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60GHz
- Stepping: 2
- CPU MHz: 1288.726
- BogoMIPS: 5204.73
- Virtualization: VT-x
- L1d cache: 32K
- L1i cache: 32K
- L2 cache: 256K

- L3 cache: 20480K
- NUMA node0 CPU(s): 0-7
- NUMA node1 CPU(s): 8-15

3 Problem Description

In this lab, we aim to measure the CPU performance of both the Lab PC and the Cluster by running the Stream Benchmark. This benchmark evaluates memory bandwidth using four operations: Copy, Scale, Sum, and Triad. These operations involve reading and writing large arrays, and their performance is limited by memory bandwidth rather than computational power.

To analyze performance, we measured throughput (GB/s) vs. problem size and FLOPs vs. problem size for each operation. The results were plotted separately for the Lab PC and the Cluster, producing a total of 16 graphs.

4 Benchmarking Methodology

We executed the operations on both the Lab PC and the Cluster, measuring:

- Execution time for varying problem sizes.
- Throughput (GB/s) and FLOPs for each operation.
- Differences in performance between the Lab PC and the Cluster.

For each case, we generated plots of:

- Throughput (Bytes/s) vs. Problem Size

$$Throughput = \frac{sizeof(double) * N * Total}{alg\_time}$$

  where, N is the number of vectors used in algorithm,

- FLOPs vs. Problem Size

$$FLOPs = \frac{M * Total}{alg\_time}$$

  where, M is the number of floating point operations in single operation.

5 Graphical Results

5.1 Vector Copy Operation : $a[i] = b[i]$

5.1.1 Throughput and FLOPs using Lab PC

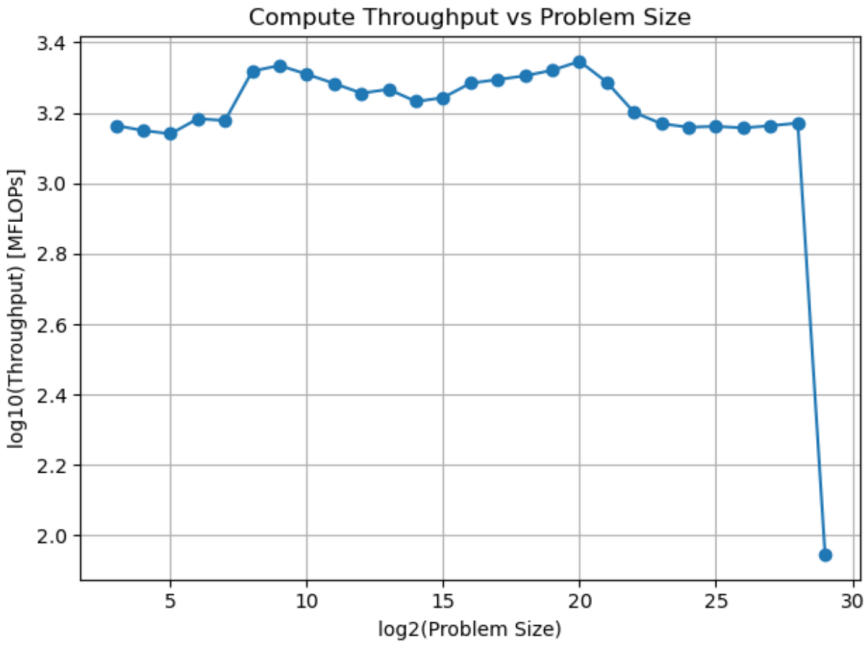

Figure 1: Throughput vs. Problem Size for the vector copy operation using Lab PC.

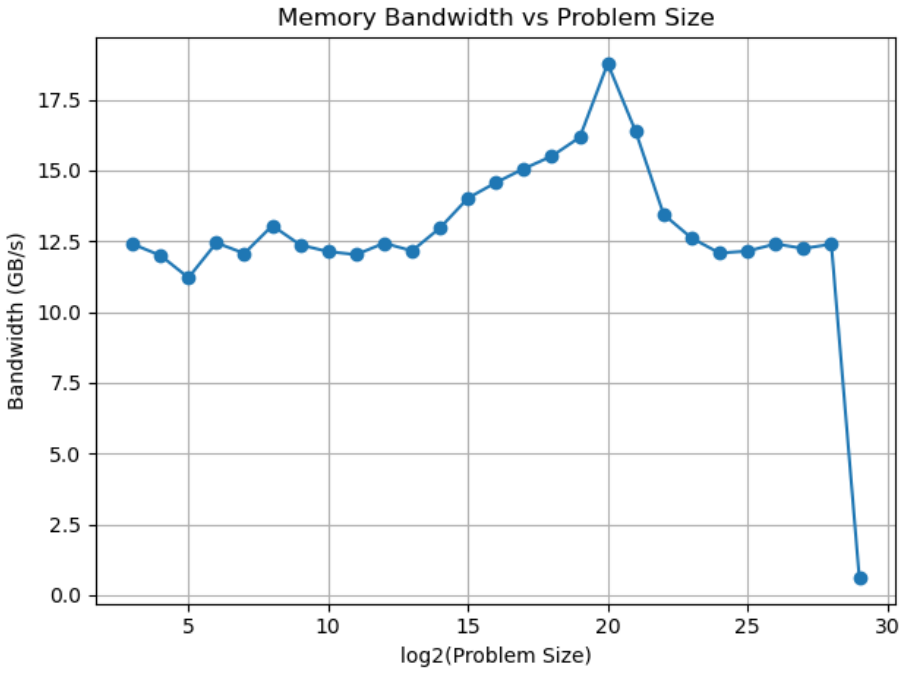

Figure 2: FLOPs vs. Problem Size for the vector copy operation using Lab PC.

5.1.2 Throughput and FLOPs using HPC Cluster

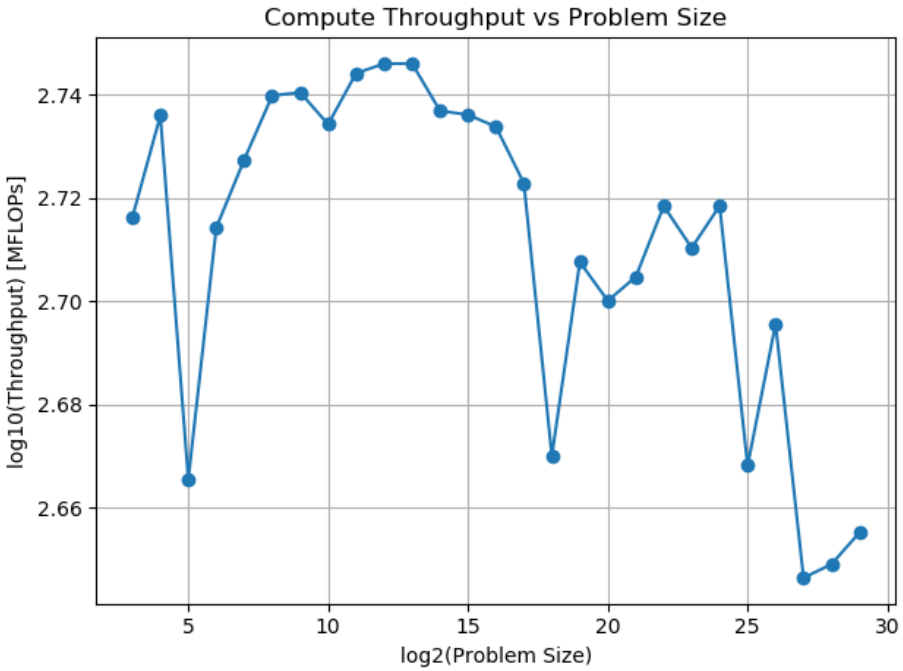

Figure 3: Throughput vs. Problem Size for the vector copy operation using HPC Cluster.

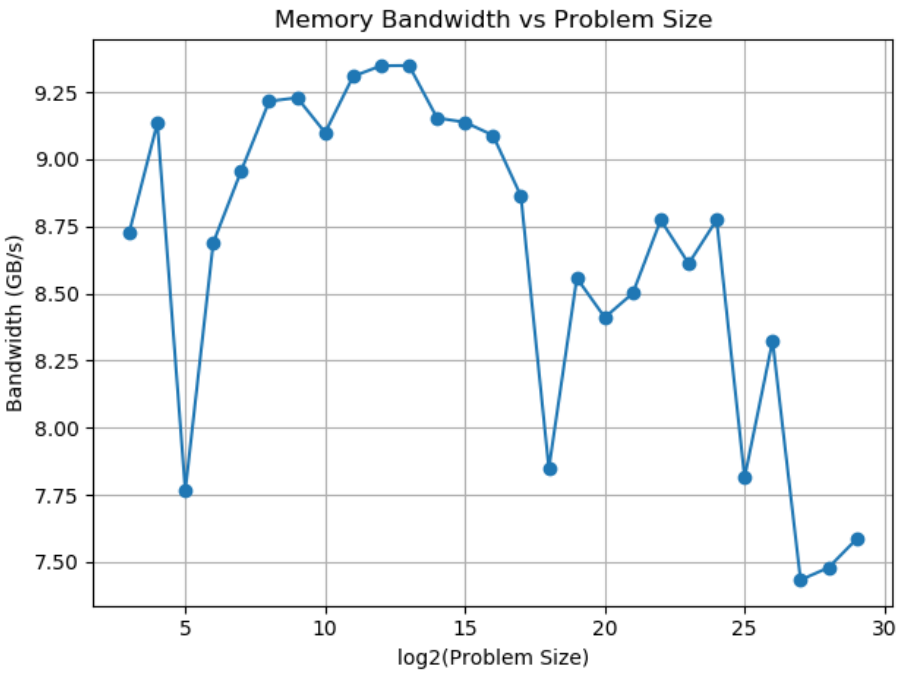

Figure 4: FLOPs vs. Problem Size for the vector copy operation using HPC Cluster.

5.2 Vector Scaling Operation : $a[i] = k * b[i]$

5.2.1 Throughput and FLOPs using Lab PC

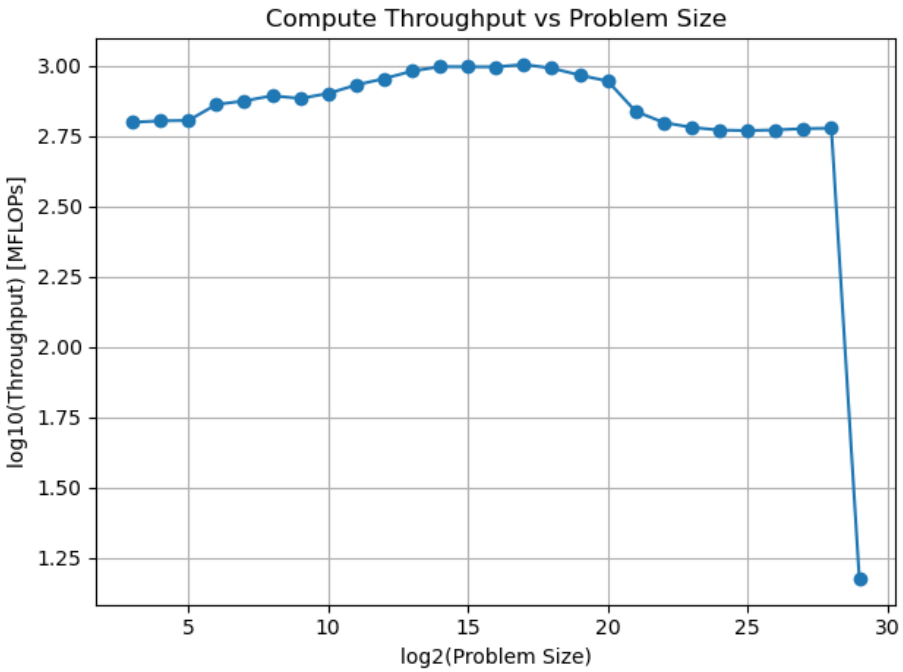

Figure 5: Throughput vs. Problem Size for the vector scaling operation using Lab PC.

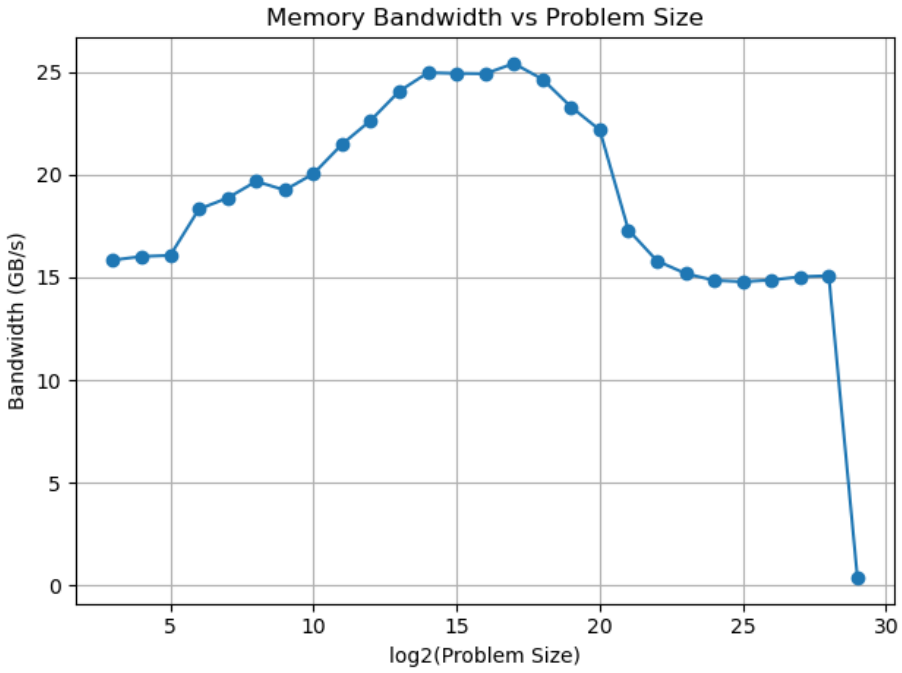

Figure 6: FLOPs vs. Problem Size for the vector scaling operation using Lab PC.

5.2.2 Throughput and FLOPs using HPC Cluster

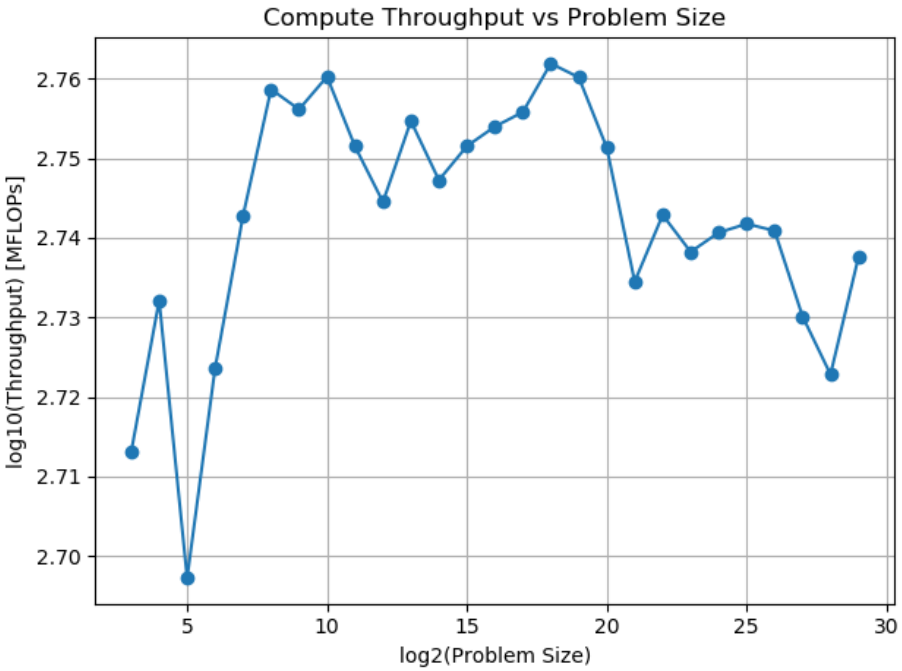

Figure 7: Throughput vs. Problem Size for the vector scaling operation using HPC Cluster

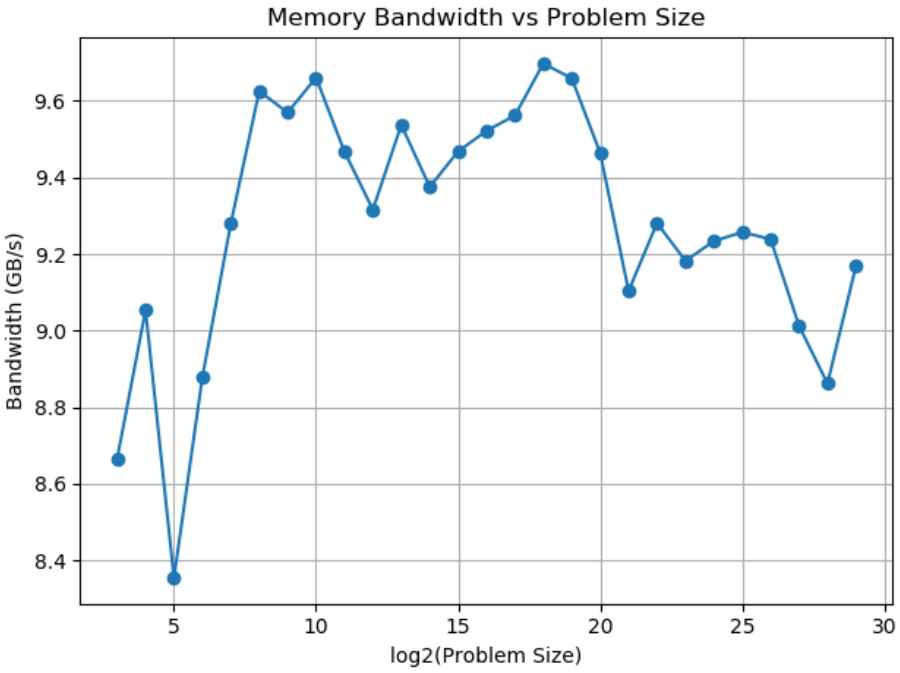

Figure 8: FLOPs vs. Problem Size for the vector scaling operation using HPC Cluster.

5.3 Vector Sum Operation : $a[i] = b[i] + c[i]$

5.3.1 Throughput and FLOPs using Lab PC

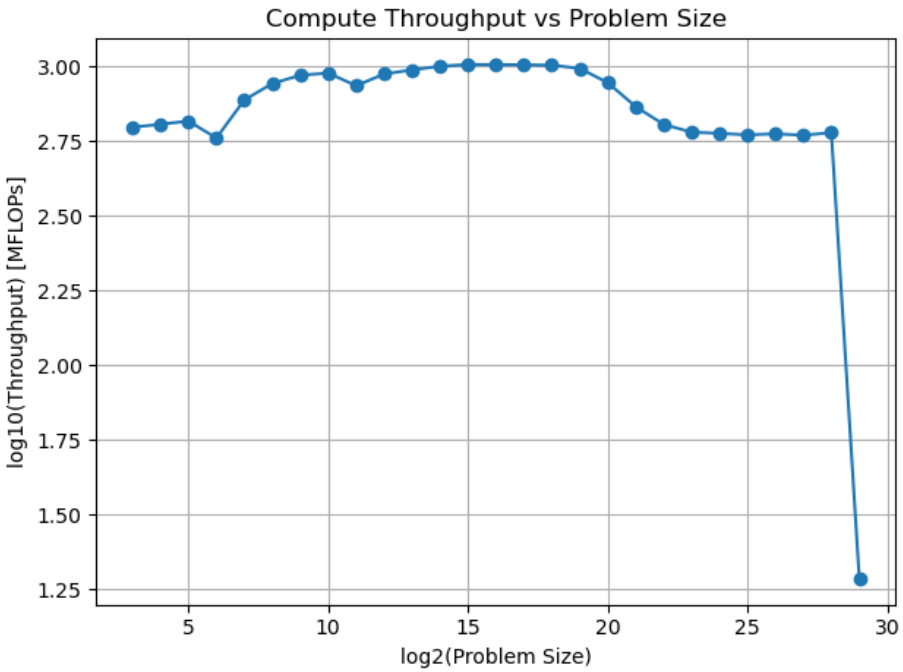

Figure 9: Throughput vs. Problem Size for the vector sum operation using Lab PC.

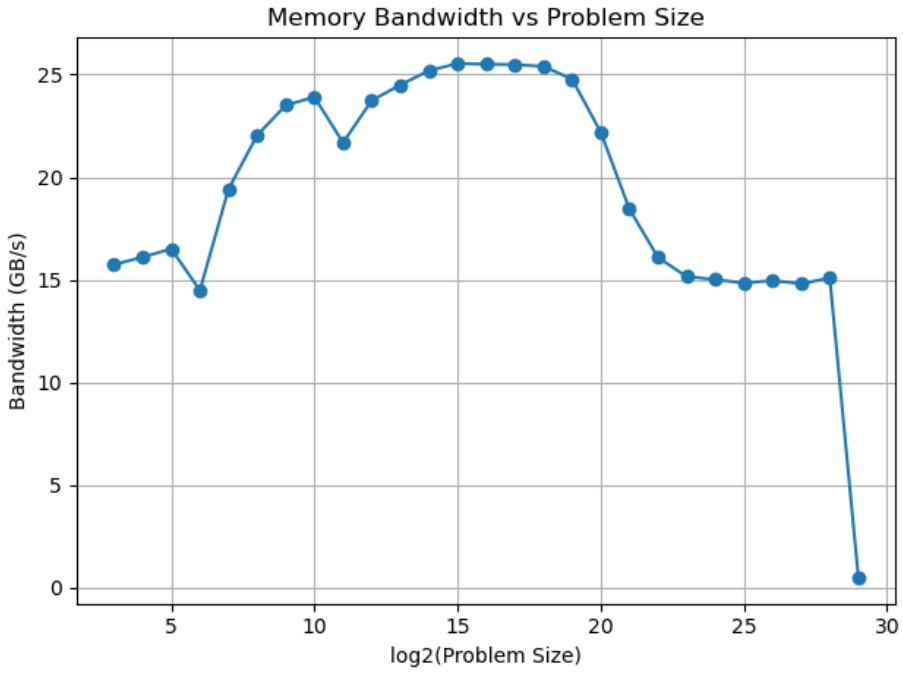

Figure 10: FLOPs vs. Problem Size for the vector sum operation using Lab PC.

5.3.2 Throughput and FLOPs using HPC Cluster

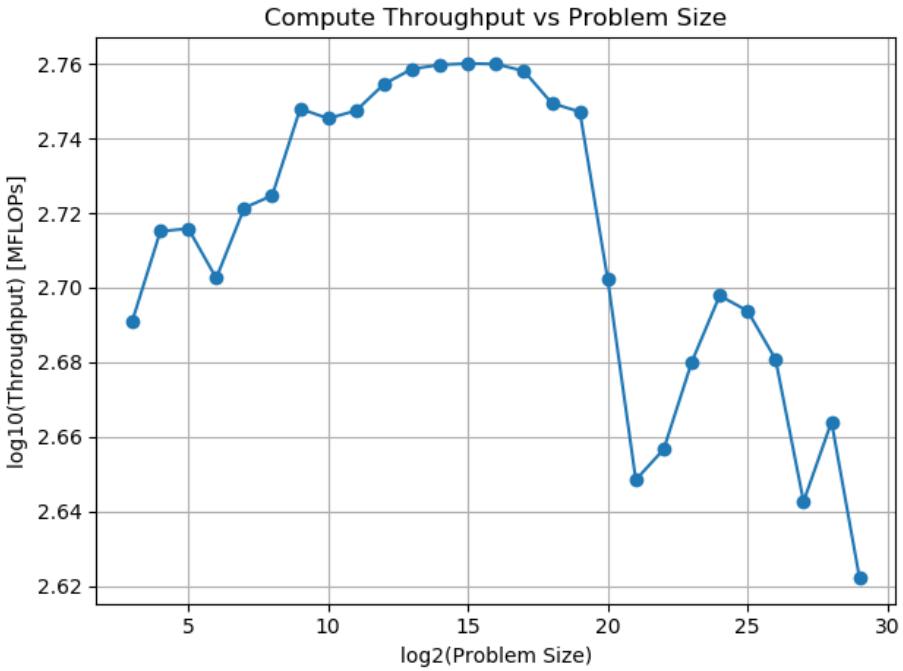

Figure 11: Throughput vs. Problem Size for the vector sum operation using HPC Cluster

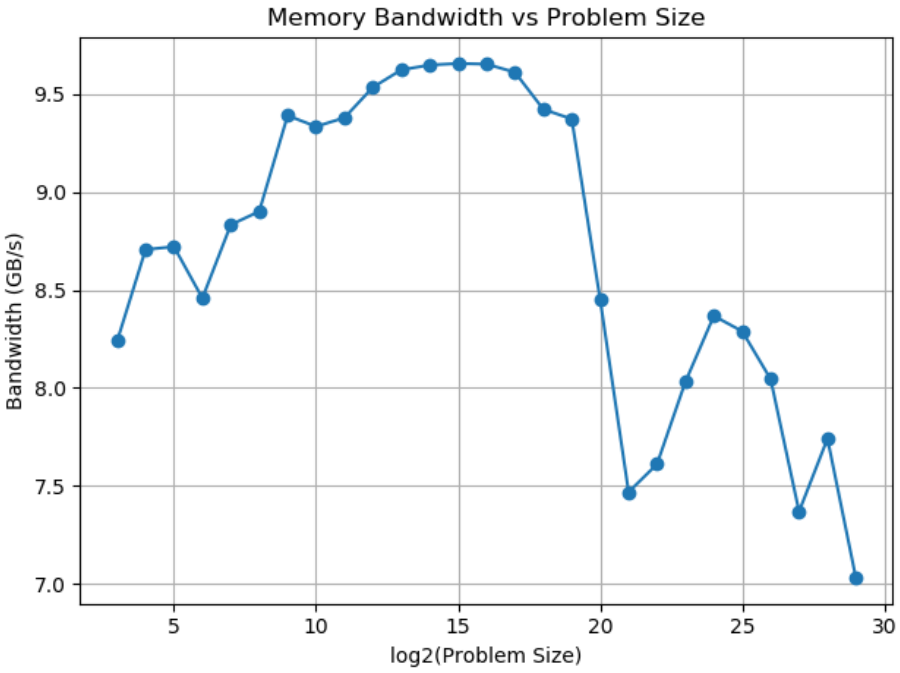

Figure 12: FLOPs vs. Problem Size for the vector sum operation using HPC Cluster.

5.4 Vector Triad Operation : $a[i] = b[i] + c[i] * d[i]$

5.4.1 Throughput and FLOPs using Lab PC

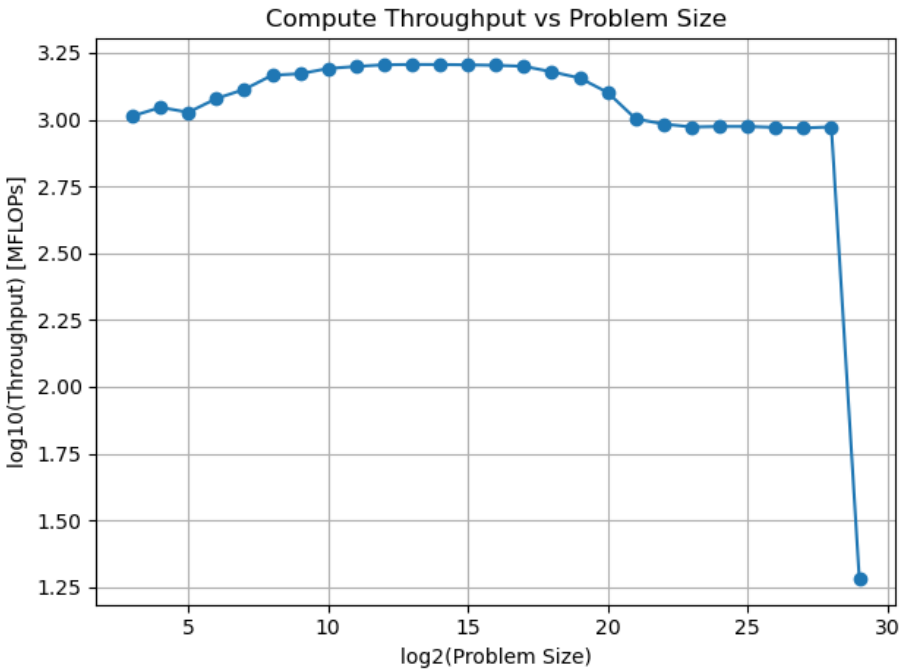

Figure 13: Throughput vs. Problem Size for the vector triad operation using Lab PC.

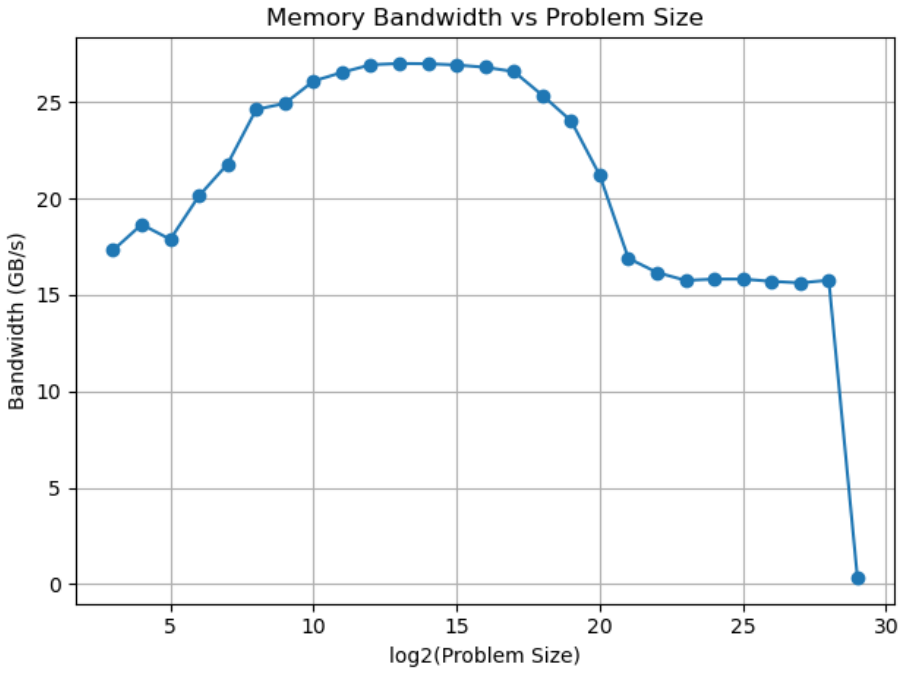

Figure 14: FLOPs vs. Problem Size for the vector triad operation using Lab PC.

5.4.2 Throughput and FLOPs using HPC Cluster

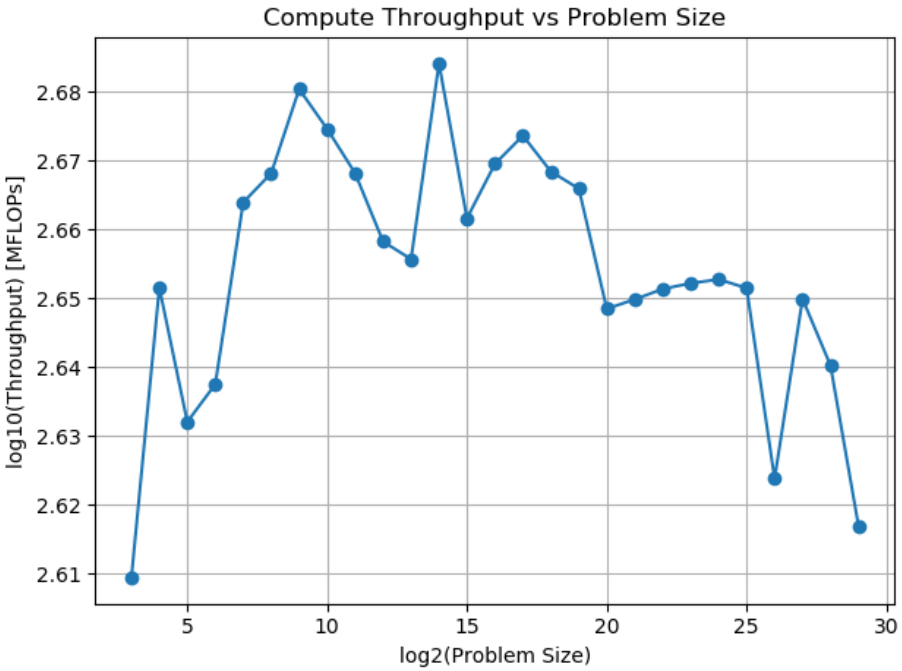

Figure 15: Throughput vs. Problem Size for the vector triad operation using HPC Cluster.

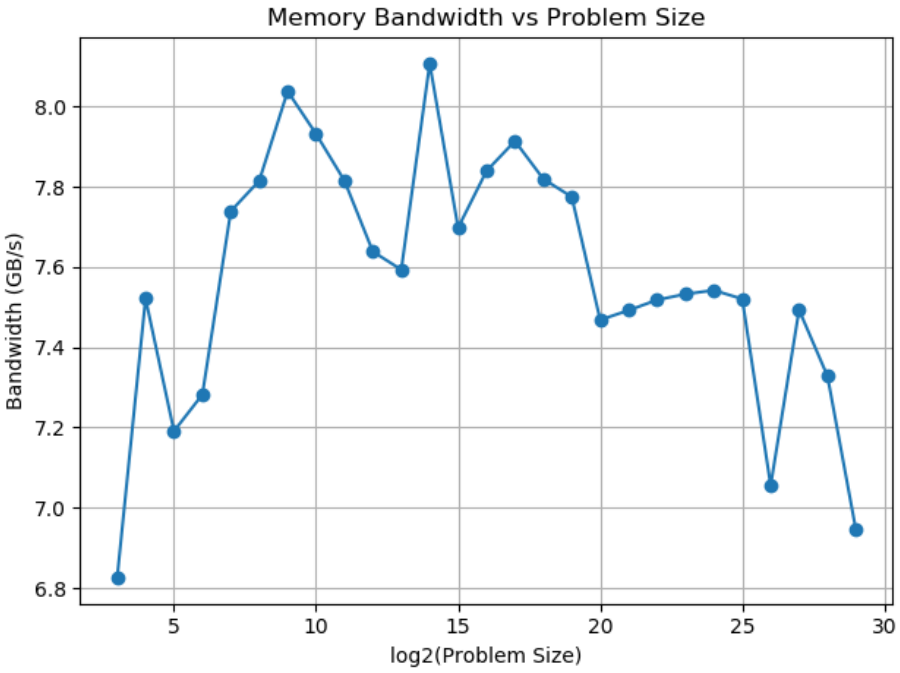

Figure 16: FLOPs vs. Problem Size for the vector triad operation using HPC Cluster.

6 Conclusion

During our experiments, we observed that system performance depended strongly on the size of the problem. When the workload was small, both the lab PC and the HPC cluster handled the operations smoothly. As the data size increased, execution became slower. The main reason for this slowdown is not the processing speed but the time required to move data in and out of memory. Memory access cannot keep up with computation, which limits overall performance. Because of this limitation, the measured results are lower than the theoretical peak values. A sudden and steep performance drop usually appears once the data no longer fits inside the cache and must be fetched from main memory.