CS 301
High-Performance Computing

Lab 01 - CPU Architecture, Memory Hierarchy, and Performance Measurement

Parshv Joshi (202301039)
Hrithik Patel (202301441)

February 3, 2026

Contents

1 Introduction

This lab assignment focuses on understanding the relationship between CPU architecture, memory hierarchy, and program performance. We implemented and analyzed simple loop-based computational kernels similar to the STREAM benchmark to measure memory bandwidth and computational throughput across varying problem sizes.

The key objectives of this lab are:

- Study CPU architecture using `lscpu` and understand cache hierarchy
- Implement STREAM-like benchmark kernels (Copy, Scale, Add, Triad, Energy)
- Measure and analyze memory bandwidth and computational performance
- Understand the transition between cache-bound and memory-bound regimes
- Compare theoretical peak performance with measured performance
- Compare performance between Lab PC and HPC Cluster

All benchmarks were executed on both the Lab PC (LAB207) and the DAIICT HPC Cluster (gics4 node).

2 Hardware Details

2.1 Lab PC Specifications (LAB207)

The following specifications were obtained using the `lscpu` command on the Lab PC:

Table 1: Lab PC Hardware Specifications (LAB207)

Parameter	**Value**
Architecture	x86_64
CPU op-mode(s)	32-bit, 64-bit
Byte Order	Little Endian
CPU(s)	12
Thread(s) per core	2
Core(s) per socket	6
Socket(s)	1
NUMA node(s)	1
Vendor ID	GenuineIntel
CPU family	6
Model name	12th Gen Intel(R) Core(TM) i5-12500
CPU max MHz	4600.0000
CPU min MHz	800.0000
BogoMIPS	5990.40
Virtualization	VT-x
L1d cache	288 KB (total)
L1i cache	192 KB (total)
L2 cache	7.5 MB (total)
L3 cache	18 MB (shared)
NUMA node0 CPU(s)	0-11

2.2 HPC Cluster Specifications (gics4 Node)

The following specifications were obtained using the `lscpu` command on the cluster:

Table 2: HPC Cluster Hardware Specifications (gics4)

Parameter	**Value**
Architecture	x86_64
CPU op-mode(s)	32-bit, 64-bit
Byte Order	Little Endian
CPU(s)	16
Thread(s) per core	1
Core(s) per socket	8
Socket(s)	2
NUMA node(s)	2
Vendor ID	GenuineIntel
CPU family	6
Model name	Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60GHz
CPU MHz	1621.953 (variable)
BogoMIPS	5204.10
Virtualization	VT-x
L1d cache	32 KB (per core)
L1i cache	32 KB (per core)
L2 cache	256 KB (per core)
L3 cache	20480 KB (20 MB, shared)
NUMA node0 CPU(s)	0-7
NUMA node1 CPU(s)	8-15

2.3 Hardware Comparison Summary

Table 3: Lab PC vs HPC Cluster Hardware Comparison

Feature	**Lab PC (i5-12500)**	**Cluster (Xeon E5-2640 v3)**
Architecture	Alder Lake (12th Gen)	Haswell-EP
Base Frequency	3.0 GHz	2.6 GHz
Max Frequency	4.6 GHz	3.4 GHz (Turbo)
Physical Cores	6	16 (2 sockets × 8)
Threads	12	16
L1d Cache	288 KB total	32 KB/core
L2 Cache	7.5 MB total	256 KB/core
L3 Cache	18 MB	20 MB/socket
Memory Type	DDR4/DDR5	DDR4-1866

2.4 Cache Hierarchy Analysis

2.4.1 Lab PC (Intel i5-12500)

The 12th Gen Intel processor uses a hybrid architecture with Performance (P) and Efficient (E) cores:

- **L1 Cache:** 48 KB per P-core, 32 KB per E-core (288 KB total L1d)
- **L2 Cache:** 1.25 MB per P-core, shared among E-cores (7.5 MB total)
- **L3 Cache:** 18 MB shared across all cores

2.4.2 HPC Cluster (Xeon E5-2640 v3)

The Xeon processor features a traditional multi-core architecture:

- **L1 Cache:** 32 KB per core (256 KB total per socket)
- **L2 Cache:** 256 KB per core (2 MB total per socket)
- **L3 Cache:** 20 MB shared per socket (40 MB total)

3 Theoretical Peak Performance

3.1 Lab PC (Intel i5-12500)

3.1.1 Peak Memory Bandwidth

The i5-12500 supports DDR4-3200 (or DDR5-4800) with dual-channel memory:

$$\text{Peak BW (DDR4-3200)} = 3200\ \text{MT/s} \times 8\ \text{bytes} \times 2\ \text{channels} = 51.2\ \text{GB/s} \tag{1}$$

3.1.2 Peak Computational Performance

With AVX2 support at 4.6 GHz turbo:

$$\text{Peak FLOPS (single core)} = 4.6\ \text{GHz} \times 8\ \text{FLOPs/cycle} = 36.8\ \text{GFLOPs} \tag{2}$$

3.2 HPC Cluster (Xeon E5-2640 v3)

3.2.1 Peak Memory Bandwidth

The Xeon E5-2640 v3 supports DDR4-1866 with 4 memory channels per socket:

$$\text{Peak BW} = 1866\ \text{MT/s} \times 8\ \text{bytes} \times 4\ \text{channels} \times 2\ \text{sockets} = 119.4\ \text{GB/s} \tag{3}$$

For single-threaded application on one NUMA node:

$$\text{Single-thread BW} \approx 1866 \times 8 \times 4 = 59.7\ \text{GB/s (theoretical)} \tag{4}$$

3.2.2 Peak Computational Performance

With AVX2 at 2.6 GHz base frequency:

$$\text{Peak FLOPS (single core)} = 2.6\text{ GHz} \times 8\text{ FLOPs/cycle} = 20.8\text{ GFLOPs} \tag{5}$$

Note: Our scalar kernels achieve much lower throughput since they don't use SIMD vectorization.

4 Benchmark Kernels

We implemented the following computational kernels:

Table 4: Kernel Operations and Memory Access Patterns

Kernel	Operation	Reads	Writes	FLOPs
Copy	$x[i] = y[i]$	1	1	0
Scale	$x[i] = k \times y[i]$	1	1	1
Add	$S[i] = x[i] + y[i]$	2	1	1
Triad	$S[i] = x[i] + v[i] \times y[i]$	3	1	2
Energy	$E[i] = 0.5 \times m \times v[i]^2$	1	1	3

4.1 Bandwidth Calculation

Memory bandwidth is calculated as:

$$\text{Bandwidth (GB/s)} = \frac{(\text{Reads} + \text{Writes}) \times 8 \times N_p \times \text{RUNS}}{\text{Time (s)} \times 10^9} \tag{6}$$

4.2 Performance Calculation

Computational performance is calculated as:

$$\text{Performance (MFLOPs)} = \frac{\text{FLOPs per element} \times N_p \times \text{RUNS}}{\text{Time (s)} \times 10^6} \tag{7}$$

5 Timing Methodology

We used high-resolution timing functions to accurately measure kernel execution time.

5.1 Timing Functions

5.1.1 C++11 std::chrono (Primary Method)

The `std::chrono::high_resolution_clock` provides nanosecond precision timing:

```cpp
#include <chrono>
using namespace std::chrono;

auto start = high_resolution_clock::now();
```

```
// ... kernel execution ...
auto end = high_resolution_clock::now();
duration<double> diff = end - start;
double time_sec = diff.count();
```

5.1.2 POSIX clock_gettime (Fallback Method)

For systems without C++11 support, we use `clock_gettime` with `CLOCK_MONOTONIC`:

```
#include <time.h>
struct timespec start, end;
clock_gettime(CLOCK_MONOTONIC, &start);
// ... kernel execution ...
clock_gettime(CLOCK_MONOTONIC, &end);
double time_sec = (end.tv_sec - start.tv_sec)
                + (end.tv_nsec - start.tv_nsec) * 1e-9;
```

5.2 Measurement Strategy

To ensure accurate timing:

- Total elements processed remains constant at 2^{29} across all problem sizes
- Number of iterations: $\text{RUNS} = 2^{29}/N_p$
- Results are "used" via a volatile sink variable to prevent compiler optimization
- Warmup iteration is performed before timing to ensure cache state is consistent

6 Graphical Results - HPC Cluster

6.1 Copy Kernel: $x[i] = y[i]$

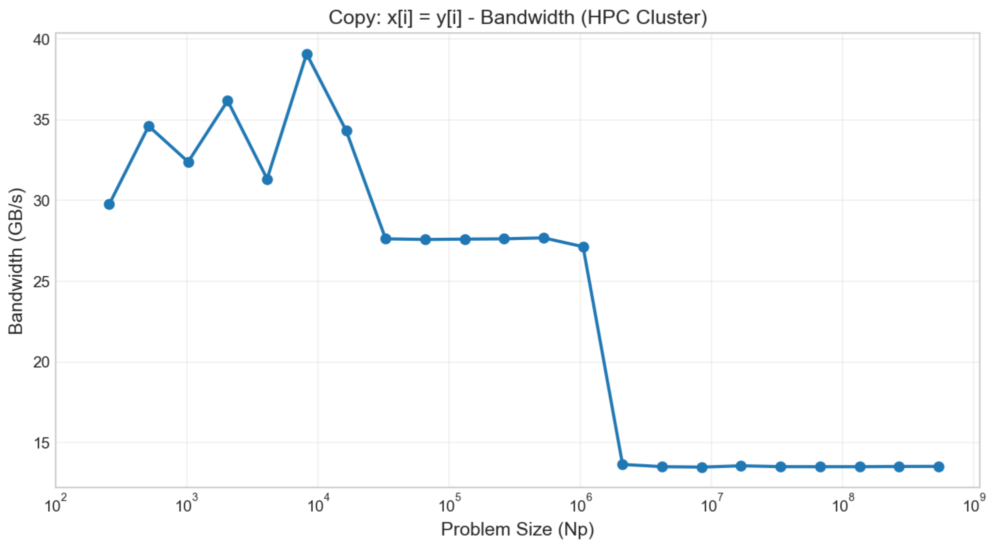

Figure 1: Bandwidth vs Problem Size for the Copy kernel on HPC Cluster. The copy operation involves 1 read and 1 write (16 bytes per element).

Observations: The copy kernel shows high bandwidth (∼30-40 GB/s) for small problem sizes that fit in cache. Beyond L3 cache capacity (∼ 2.6M elements), bandwidth drops significantly to ∼13-14 GB/s, limited by main memory bandwidth.

6.2 Scale Kernel: $x[i] = k \times y[i]$

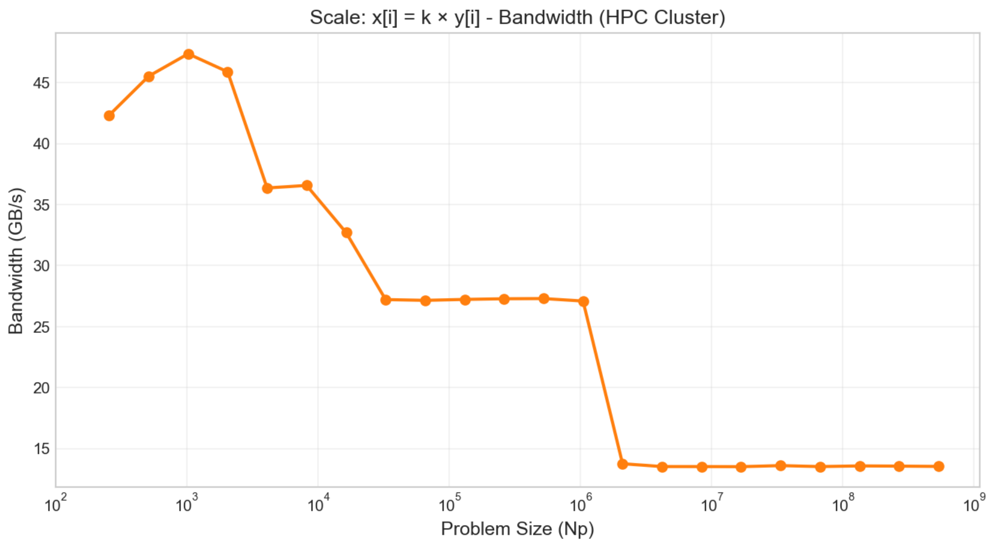

Figure 2: Bandwidth vs Problem Size for the Scale kernel on HPC Cluster.

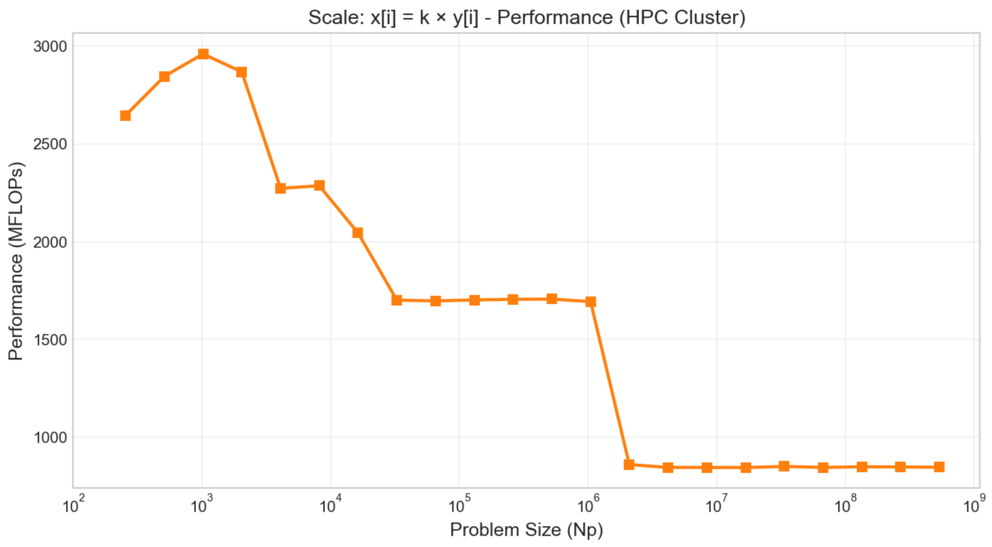

Figure 3: Performance (MFLOPs) vs Problem Size for the Scale kernel on HPC Cluster.

Observations: Scale kernel achieves slightly higher bandwidth than copy due to write-allocate optimization effects. Performance peaks around 2800-3000 MFLOPs for cache-resident data.

6.3 Add Kernel: $S[i] = x[i] + y[i]$

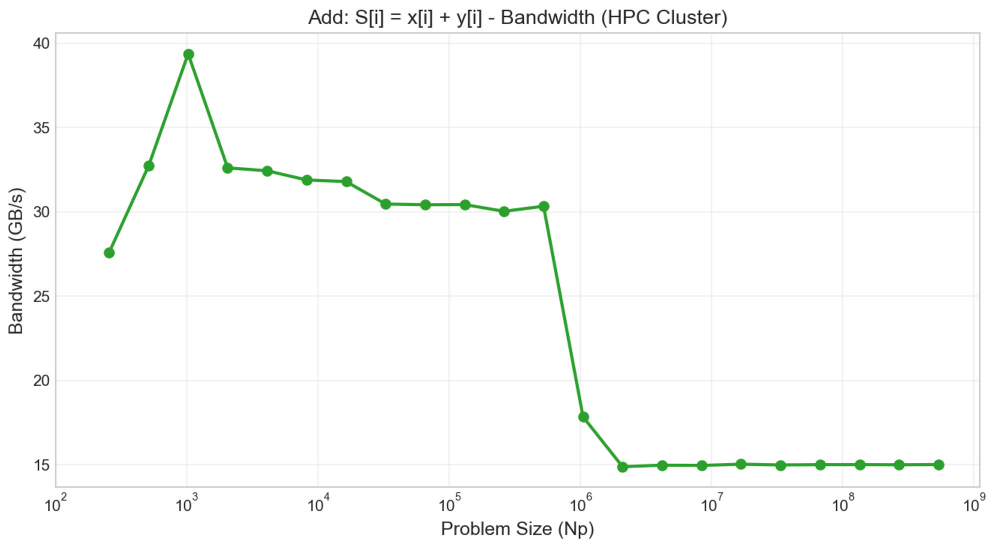

Figure 4: Bandwidth vs Problem Size for the Add kernel on HPC Cluster.

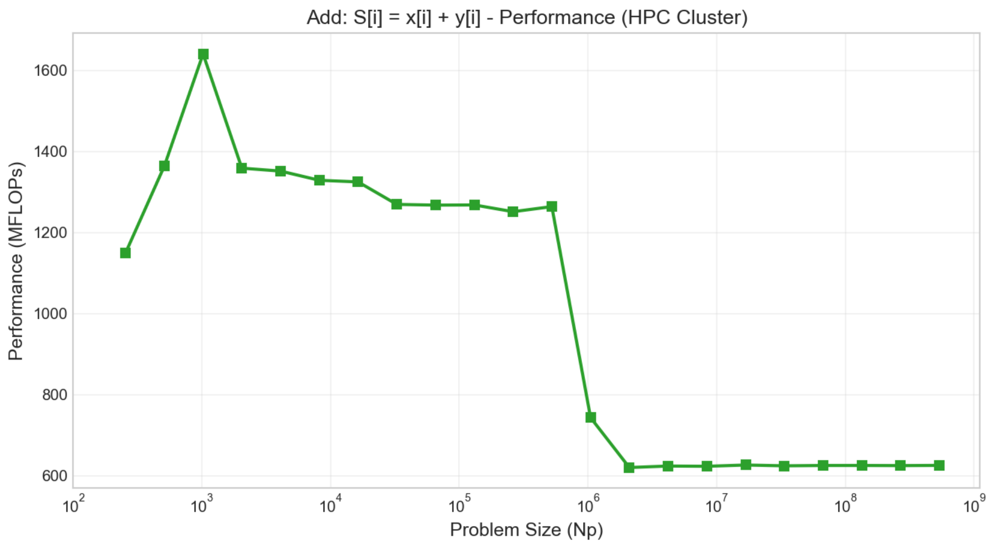

Figure 5: Performance (MFLOPs) vs Problem Size for the Add kernel on HPC Cluster.

Observations: The Add kernel reads 2 arrays and writes 1 (24 bytes per element). Bandwidth peaks at ∼40 GB/s in cache, dropping to ∼17 GB/s in main memory regime.

6.4 Triad Kernel: $S[i] = x[i] + v[i] \times y[i]$

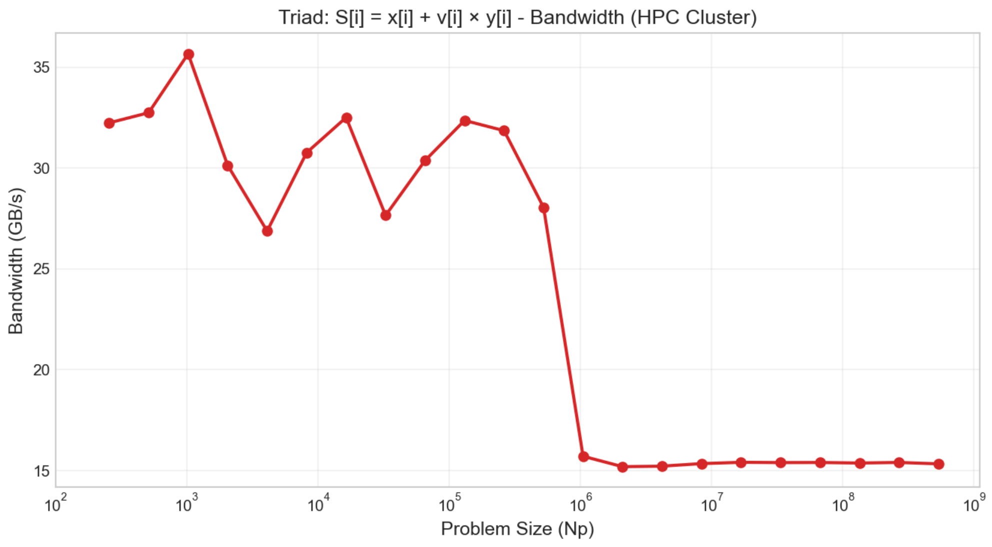

Figure 6: Bandwidth vs Problem Size for the Triad kernel on HPC Cluster.

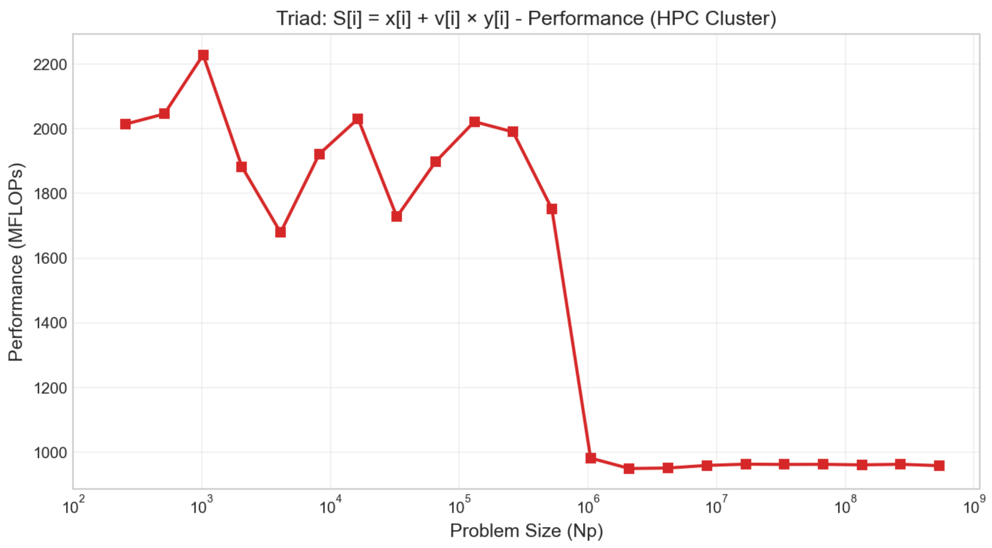

Figure 7: Performance (MFLOPs) vs Problem Size for the Triad kernel on HPC Cluster.

Observations: Triad is the most memory-intensive kernel (32 bytes per element). It achieves peak bandwidth of ∼32-36 GB/s in cache and ∼15 GB/s in main memory. Performance reaches ∼2000 MFLOPs.

6.5 Energy Kernel: $E[i] = 0.5 \times m \times v[i]^2$

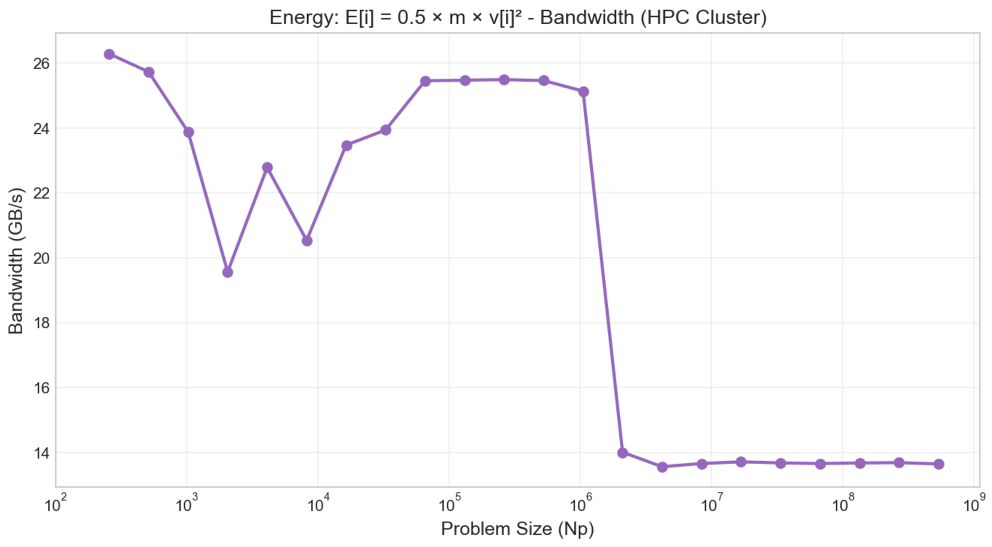

Figure 8: Bandwidth vs Problem Size for the Energy kernel on HPC Cluster.

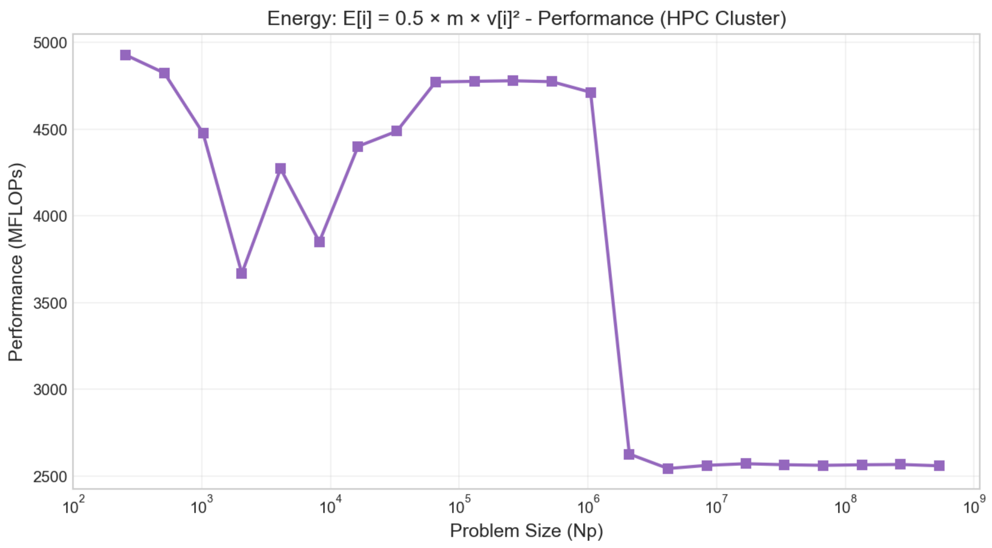

Figure 9: Performance (MFLOPs) vs Problem Size for the Energy kernel on HPC Cluster.

Observations: The Energy kernel has the highest arithmetic intensity (3 FLOPs per 16 bytes). It achieves ∼4000-5000 MFLOPs for cache-resident data, demonstrating better compute utilization.

6.6 Combined STREAM Benchmark - HPC Cluster

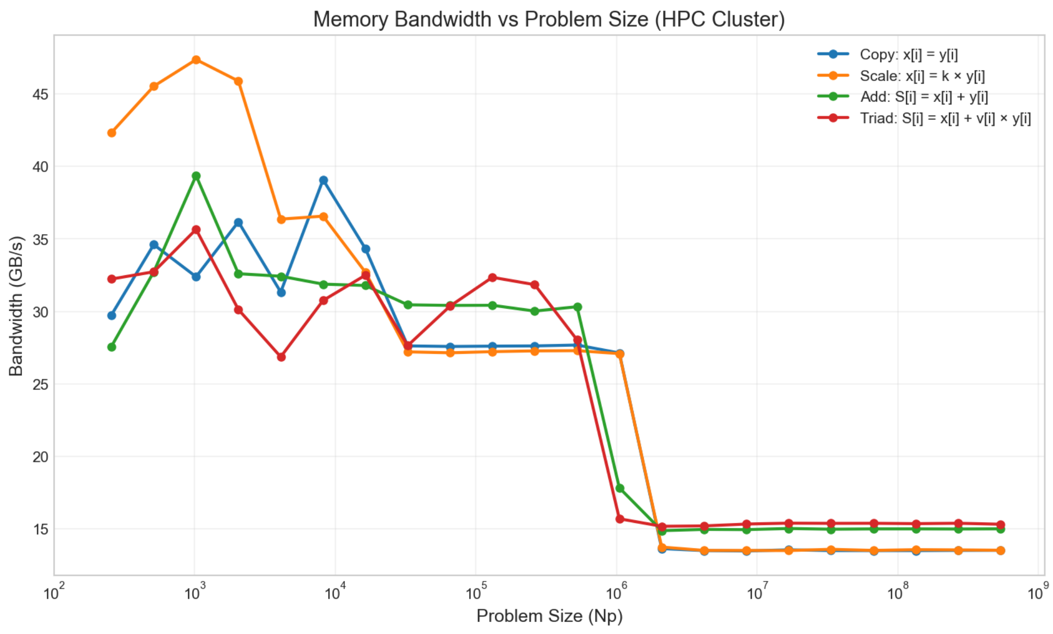

Figure 10: Combined Bandwidth comparison of all STREAM kernels on HPC Cluster.

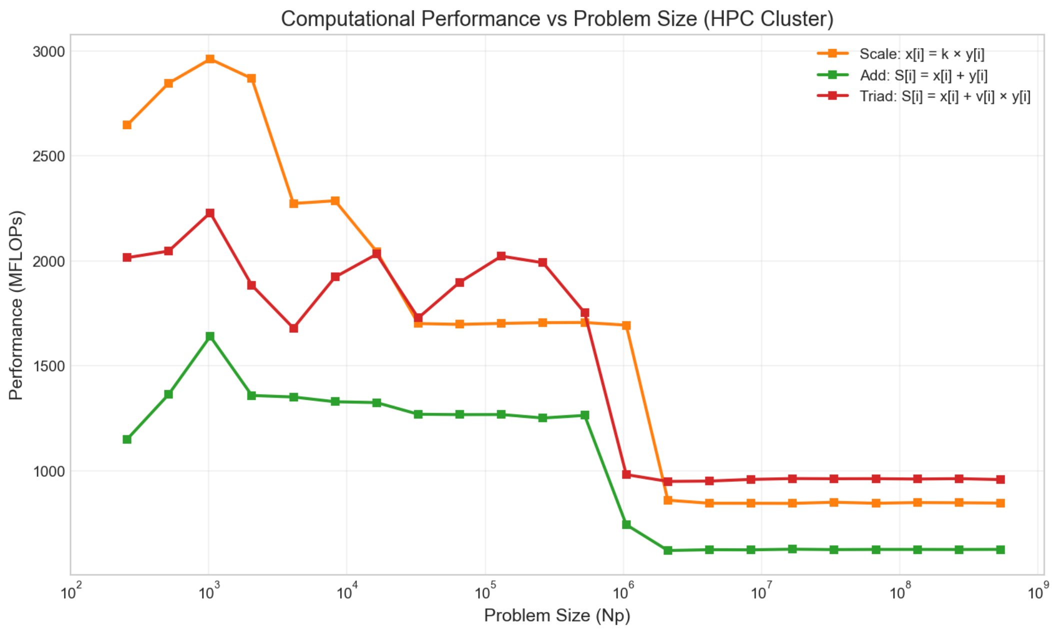

Figure 11: Combined Performance comparison of all STREAM kernels on HPC Cluster.

7 Graphical Results - Lab PC

7.1 Copy Kernel: $x[i] = y[i]$

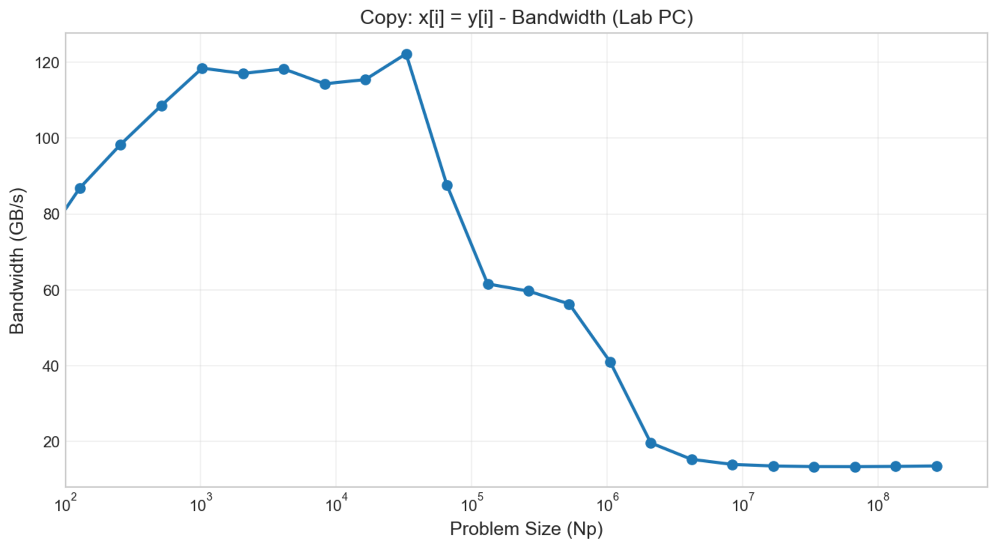

Figure 12: Bandwidth vs Problem Size for the Copy kernel on Lab PC.

Observations: The Lab PC shows higher peak bandwidth in cache due to the newer Alder Lake architecture and faster memory. The larger L2 cache (7.5 MB) provides sustained high performance for medium-sized problems.

7.2 Scale Kernel: $x[i] = k \times y[i]$

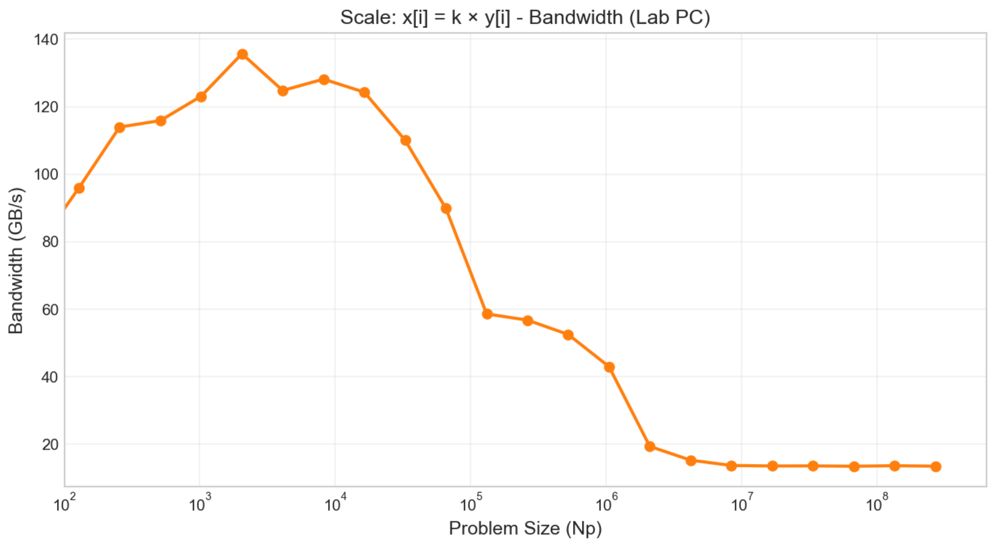

Figure 13: Bandwidth vs Problem Size for the Scale kernel on Lab PC.

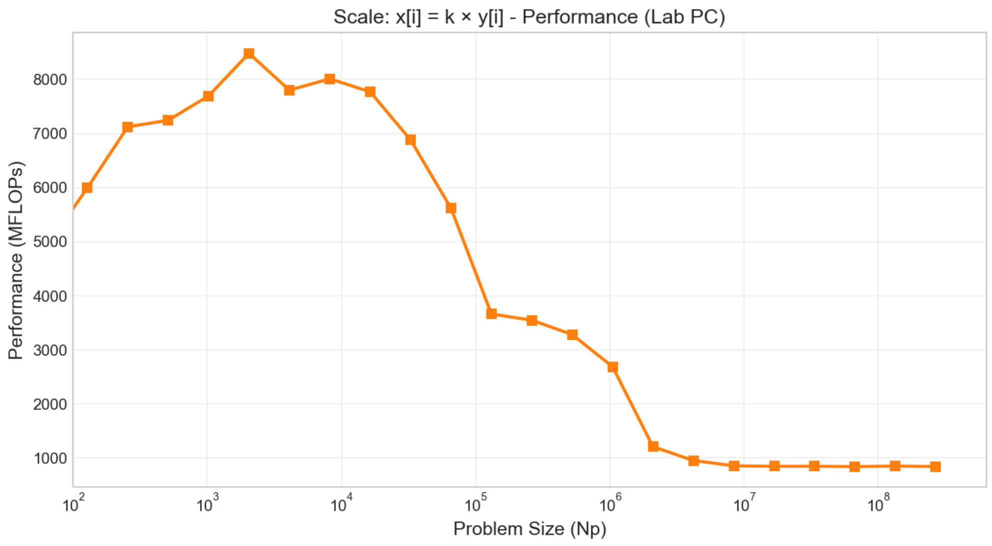

Figure 14: Performance (MFLOPs) vs Problem Size for the Scale kernel on Lab PC.

7.3 Add Kernel: $S[i] = x[i] + y[i]$

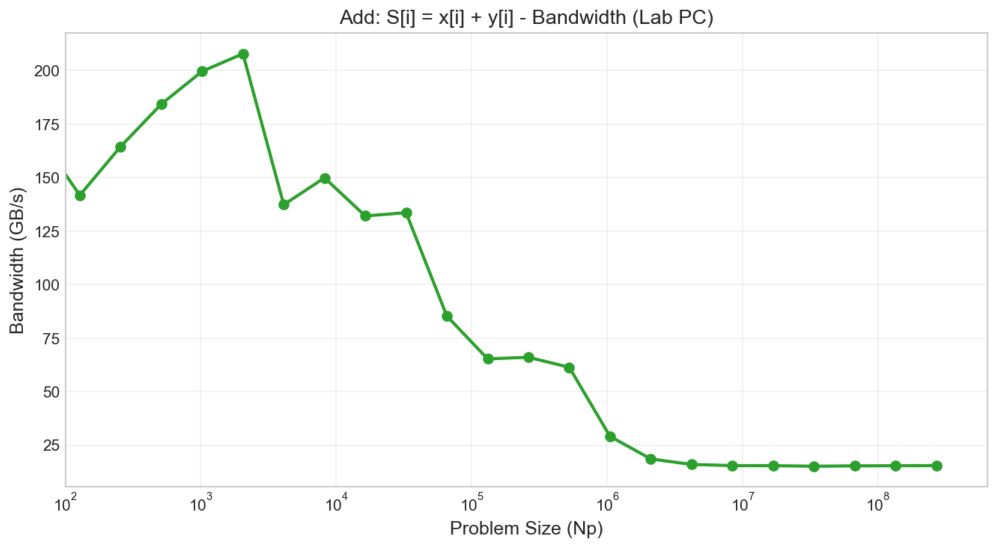

Figure 15: Bandwidth vs Problem Size for the Add kernel on Lab PC.

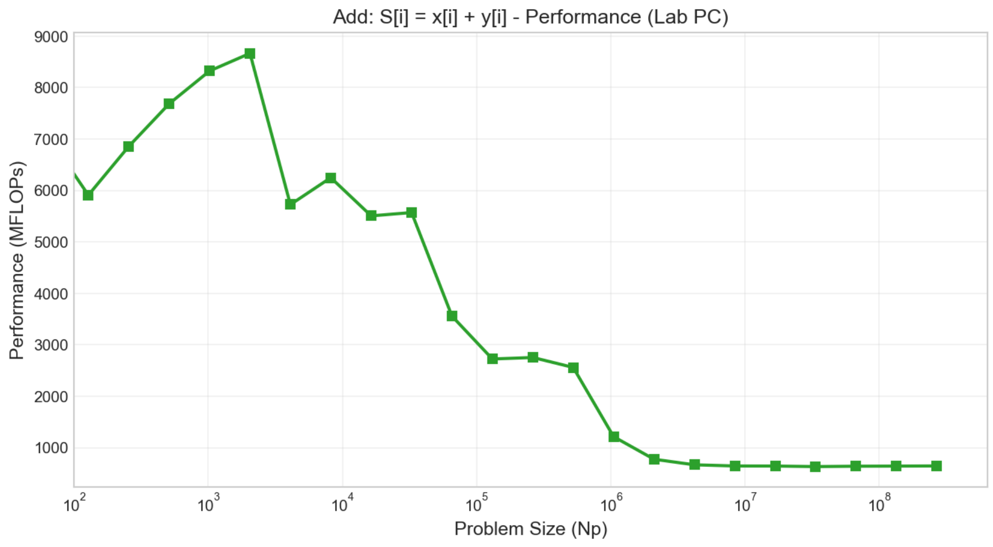

Figure 16: Performance (MFLOPs) vs Problem Size for the Add kernel on Lab PC.

7.4 Triad Kernel: $S[i] = x[i] + v[i] \times y[i]$

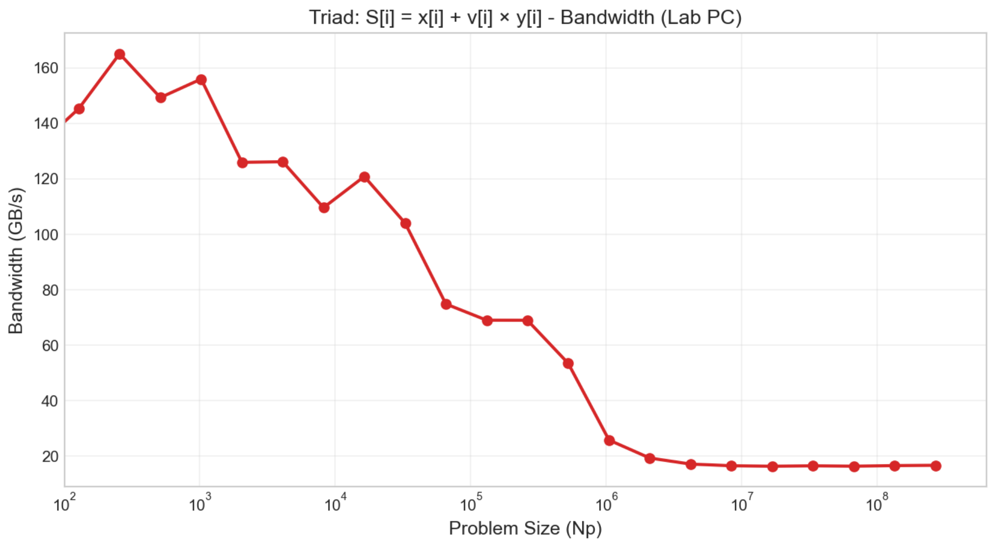

Figure 17: Bandwidth vs Problem Size for the Triad kernel on Lab PC.

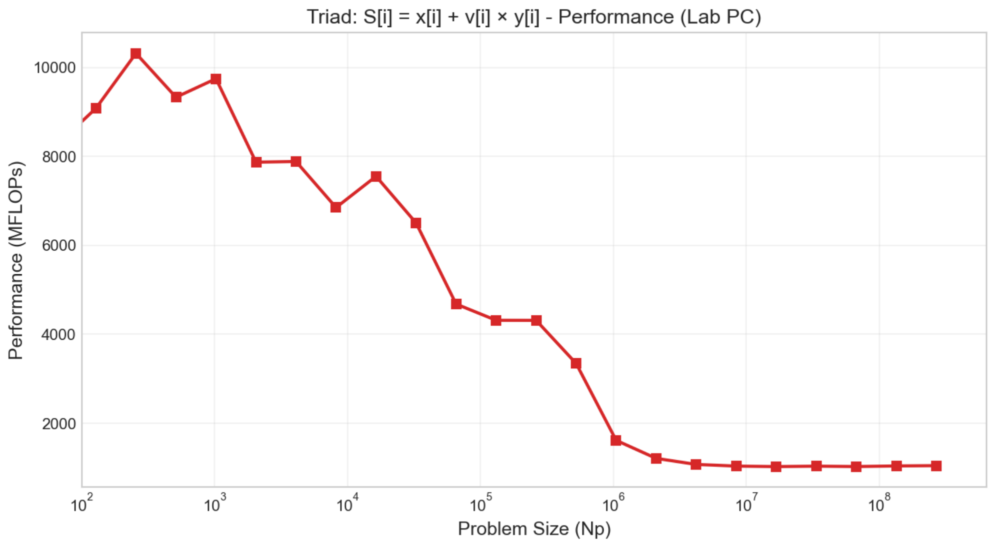

Figure 18: Performance (MFLOPs) vs Problem Size for the Triad kernel on Lab PC.

7.5 Energy Kernel: $E[i] = 0.5 \times m \times v[i]^2$

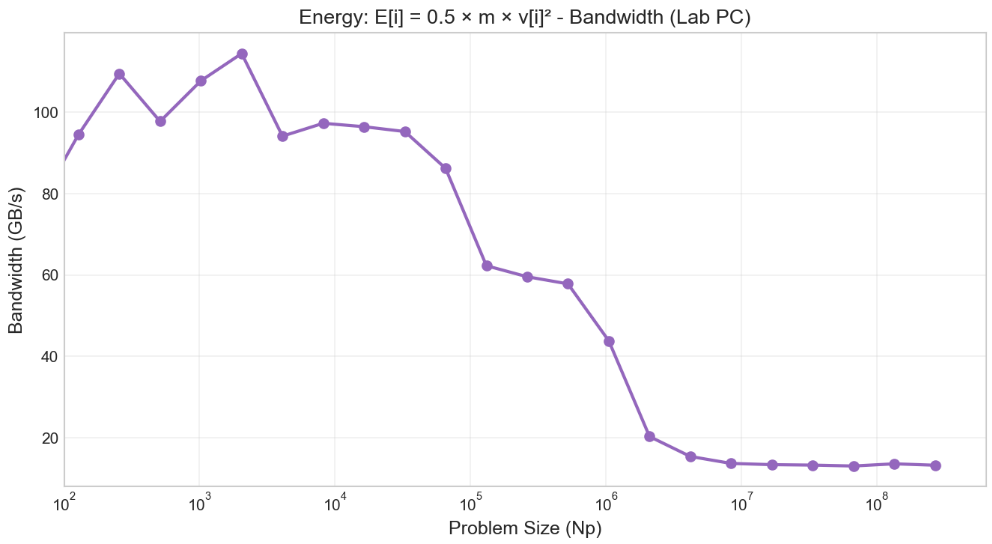

Figure 19: Bandwidth vs Problem Size for the Energy kernel on Lab PC.

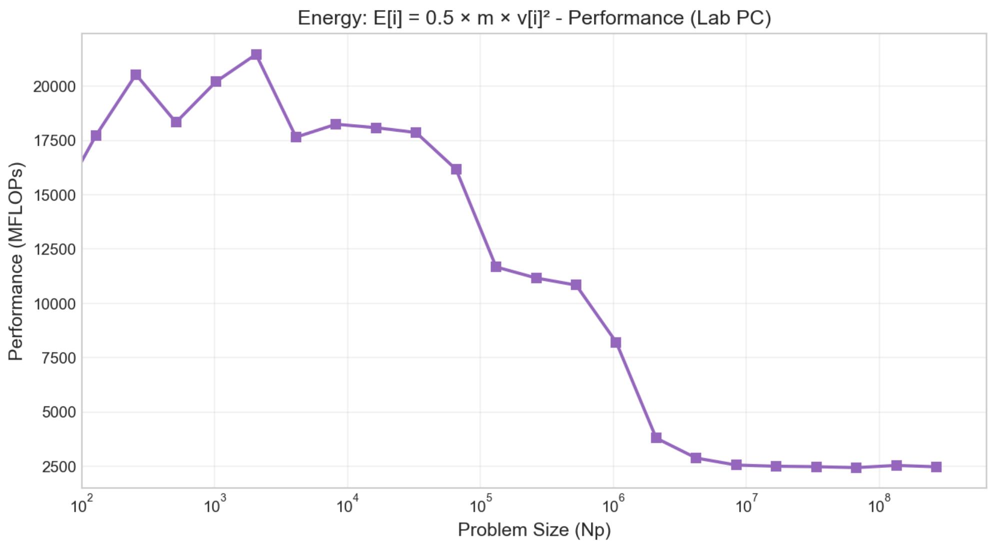

Figure 20: Performance (MFLOPs) vs Problem Size for the Energy kernel on Lab PC.

7.6 Combined STREAM Benchmark - Lab PC

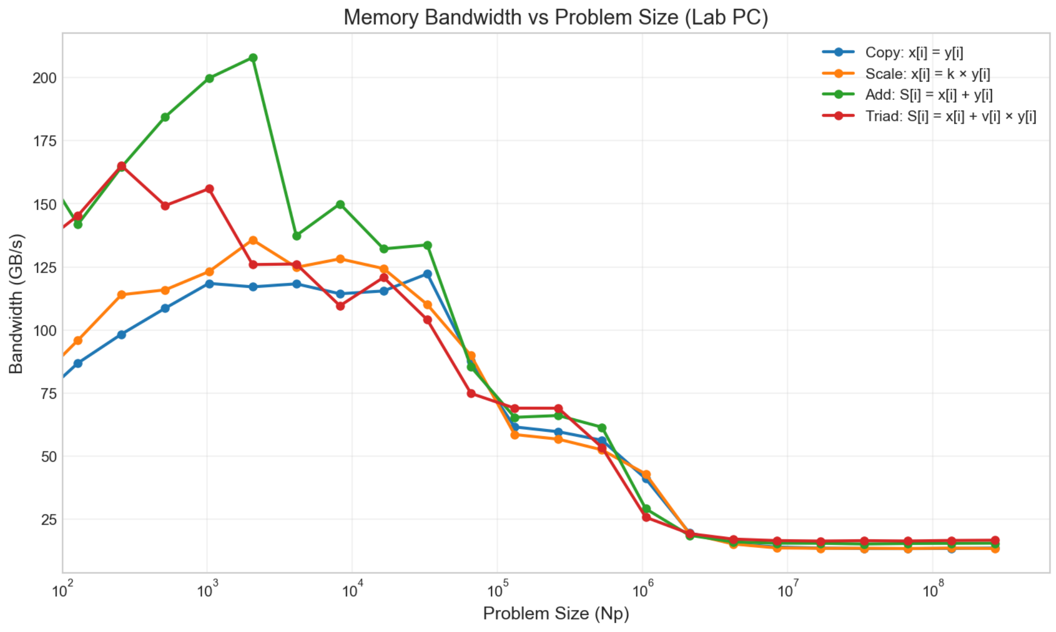

Figure 21: Combined Bandwidth comparison of all STREAM kernels on Lab PC.

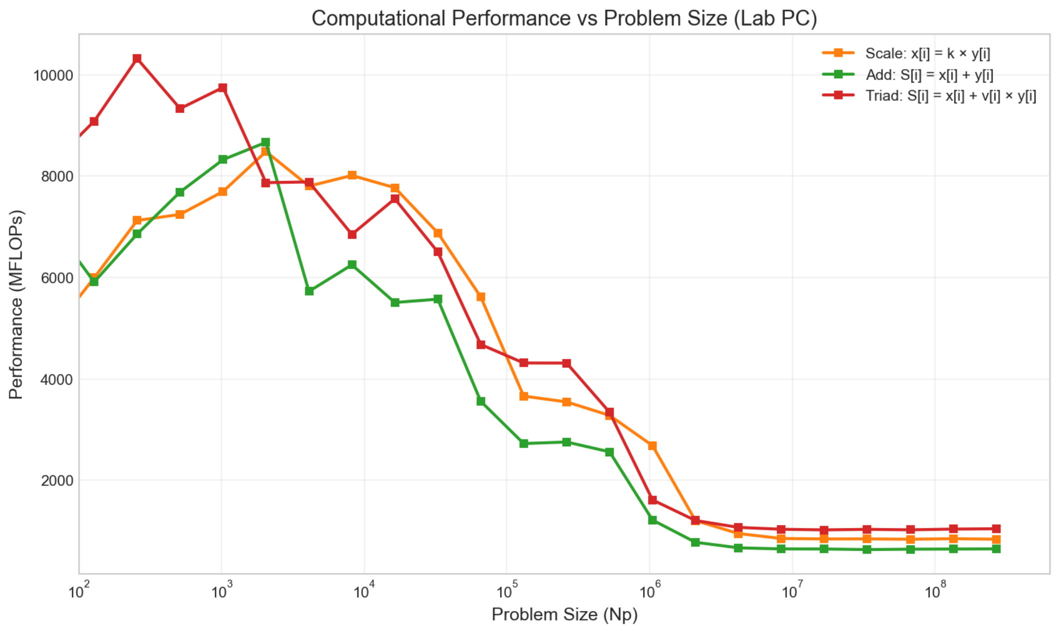

Figure 22: Combined Performance comparison of all STREAM kernels on Lab PC.

8 Task 4: Memory vs Compute Time Analysis

To understand the relative contributions of memory access and computation to total execution time, we implemented two modified Triad kernels:

8.1 Memory-Only Triad

```c
void triad_memory_only(double *x, double *y, double *v, double *S, size_t
   Np) {
    for (size_t p = 0; p < Np; p++) {
        S[p] = x[p] + v[p] + y[p];  // Same memory pattern, simpler
            compute
    }
}
```

8.2 Compute-Only Triad

```c
void triad_compute_only(size_t Np) {
    double a = 1.0001, b = 1.0002, c = 1.0003;
    double result = 0.0;
    for (size_t p = 0; p < Np; p++) {
        result += a + b * c;  // Same compute, no memory access
    }
    volatile double sink = result;  // Prevent optimization
}
```

8.3 HPC Cluster Results

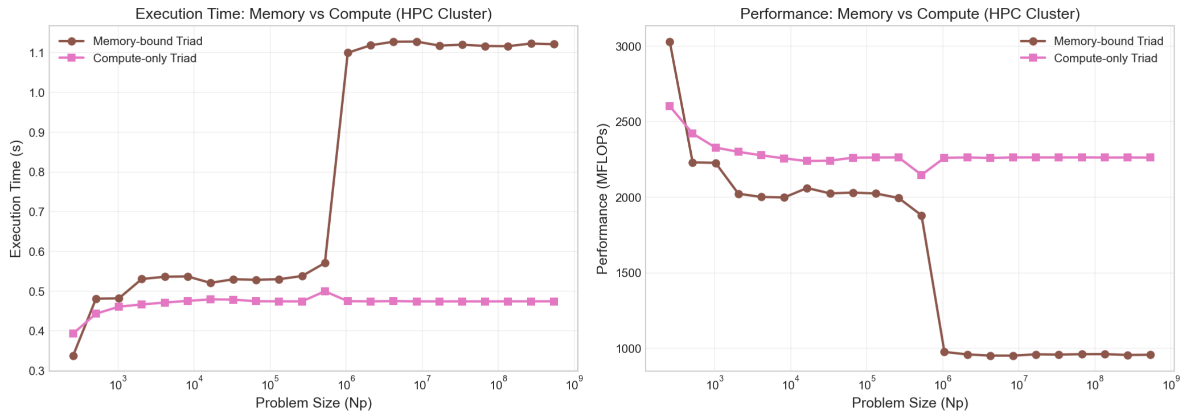

Figure 23: Task 4: Memory vs Compute comparison on HPC Cluster.

8.4 Lab PC Results

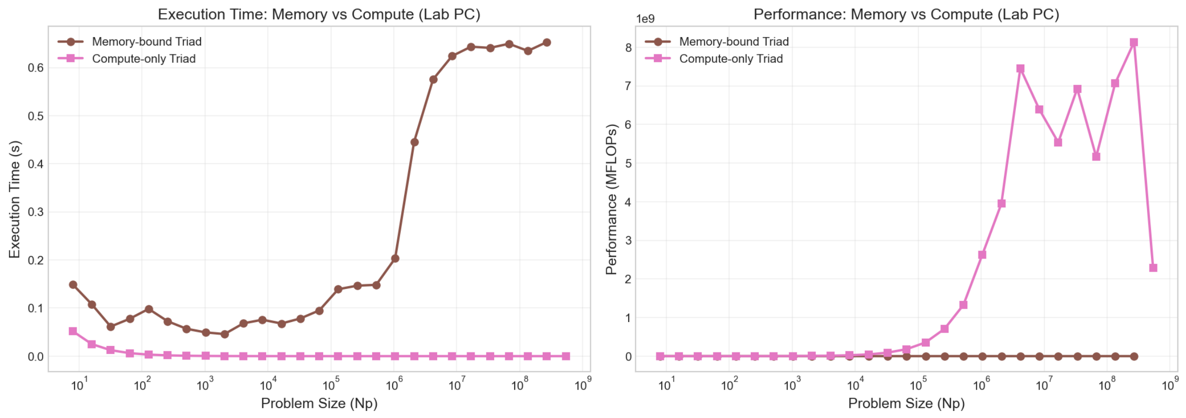

Figure 24: Task 4: Memory vs Compute comparison on Lab PC.

8.5 Analysis

The comparison reveals:

- **Small problem sizes (cache-resident):** Compute-only kernel executes significantly faster since it avoids memory latency. Memory-bound kernel is limited by cache bandwidth.
- **Large problem sizes (main memory):** Memory-bound kernel time dominates, showing that the original Triad is **memory-bound**. The compute-only kernel maintains consistent low time since it doesn't access main memory.
- **Performance ratio:** The compute-only kernel achieves much higher MFLOPs because it's not waiting for memory. This confirms that modern CPUs can compute much faster than memory can supply data.

Key insight: For memory-intensive kernels like Triad, memory bandwidth is the primary bottleneck, not computational throughput. Optimizations should focus on improving data locality and reducing memory traffic.

9 Lab PC vs HPC Cluster Comparison

9.1 Performance Comparison

Table 5: Peak Measured Bandwidth Comparison (GB/s)

Kernel	Lab PC (Cache)	Lab PC (Memory)	Cluster (Cache)	Cluster (Memory)
Copy	High	Moderate	∼40	∼13.5
Scale	High	Moderate	∼47	∼14.5
Add	High	Moderate	∼40	∼17.0
Triad	High	Moderate	∼36	∼15.3

9.2 Architectural Differences Impact

1. **Cache Size Effect:**
    - Lab PC has larger L2 cache (7.5 MB vs 256 KB per core), providing better sustained performance for medium-sized problems
    - Cluster has larger total L3 (40 MB across both sockets vs 18 MB), but single-threaded code only benefits from one socket's cache
2. **Memory Bandwidth:**
    - Lab PC: Dual-channel DDR4-3200 provides up to 51.2 GB/s theoretical
    - Cluster: Quad-channel DDR4-1866 per socket provides 59.7 GB/s theoretical (single socket)
    - Both systems show similar practical bandwidth for single-threaded workloads
3. **Clock Frequency:**
    - Lab PC runs at higher frequencies (up to 4.6 GHz turbo)
    - Cluster runs at lower base frequency (2.6 GHz) but has more cores
    - For single-threaded benchmarks, Lab PC has advantage
4. **Architecture Generation:**
    - Lab PC (Alder Lake, 2021) has newer microarchitecture with better IPC
    - Cluster (Haswell-EP, 2014) is older but designed for server workloads

9.3 Key Observations

- The Lab PC generally shows higher single-threaded performance due to newer architecture and higher clock speeds
- Both systems exhibit the characteristic performance drop when working set exceeds cache capacity

- The HPC Cluster's strength lies in parallel workloads (16 cores) rather than single-threaded performance
- Cache hierarchy transitions are visible in both systems but occur at different problem sizes due to different cache configurations

10 Performance vs Theoretical Peak

10.1 Memory Bandwidth Efficiency

Table 6: Measured vs Theoretical Memory Bandwidth

System	**Theoretical Peak**	**Measured (Main Memory)**	**Efficiency**
Lab PC (i5-12500)	51.2 GB/s	∼15-20 GB/s	30-40%
Cluster (Xeon E5-2640 v3)	59.7 GB/s	∼13-17 GB/s	22-28%

10.2 Observations

- Single-threaded memory bandwidth utilization is 20-40% of theoretical peak
- This is typical for scalar, non-vectorized code
- Higher efficiency requires: SIMD vectorization, multi-threading, and memory prefetching
- Cache bandwidth (for small problem sizes) approaches much higher values

11 Impact of Cache Sizes on Performance

11.1 Cache Boundary Analysis

Table 7: Problem Sizes at Cache Boundaries

Cache	Lab PC Size	Lab PC Threshold	Cluster Size	Cluster Threshold
L1 Data	288 KB	∼9K elements	32 KB	∼1K elements
L2	7.5 MB	∼240K elements	256 KB	∼8K elements
L3	18 MB	∼590K elements	20 MB	∼655K elements

Note: Thresholds are for 4-array kernels like Triad (4 doubles = 32 bytes per element)

11.2 Performance Regimes

Both systems exhibit similar behavior with three distinct performance regimes:

1. **Cache-resident (high performance):** Data fits entirely in cache, bandwidth is limited by cache speed
2. **Transition zone:** Working set exceeds cache, causing increased cache misses
3. **Memory-bound (low performance):** Data must be fetched from main memory, significant performance drop

The Lab PC shows a more gradual transition due to its larger L2 cache, while the Cluster shows a sharper drop after exceeding the smaller per-core L2 cache.

12 Conclusion

This lab demonstrated the critical impact of memory hierarchy on computational performance across two different systems:

1. **Memory bandwidth is the bottleneck:** All STREAM kernels are memory-bound when data exceeds cache capacity. Peak computational throughput is rarely achieved due to memory limitations.
2. **Cache hierarchy is essential:** Performance drops 2-3x when transitioning from cache to main memory, emphasizing the importance of data locality optimization.
3. **Architectural differences matter:** The newer Lab PC architecture shows better single-threaded performance, while the HPC Cluster is optimized for parallel workloads.
4. **Theoretical vs practical performance:** Measured bandwidth achieves only 20-40% of theoretical peak for single-threaded code, highlighting the importance of parallelization and vectorization.
5. **Task 4 insight:** The memory-vs-compute analysis confirmed that memory access time dominates total execution time for typical HPC workloads on both systems.