REPORT

System and CPU Specifications:

- Test System: Apple MacBook Pro (Apple M4 Pro CPU)
- Number of Cores: 12 cores
- Clock Speed: 3.5 GHz
- IDE Used: Visual Studio Code
- Compiler:Apple Clang (LLVM-based):Clang++ with C++17 standard
- BLAS library: Apple Accelerate
- OS:macOS (Unix-based)

1) Sequential CPU implementation:

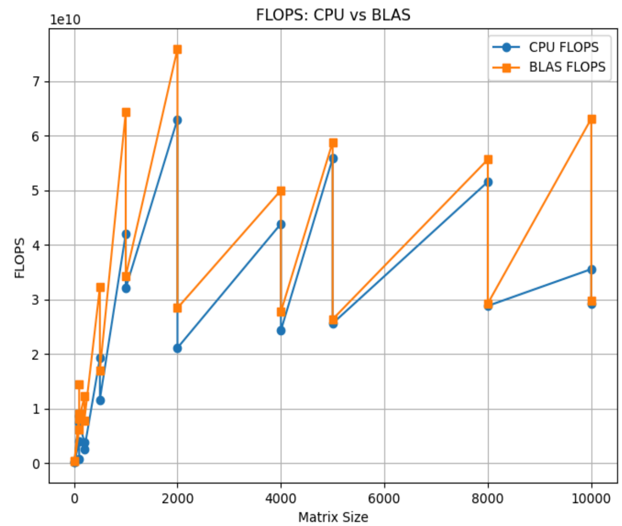
Output:

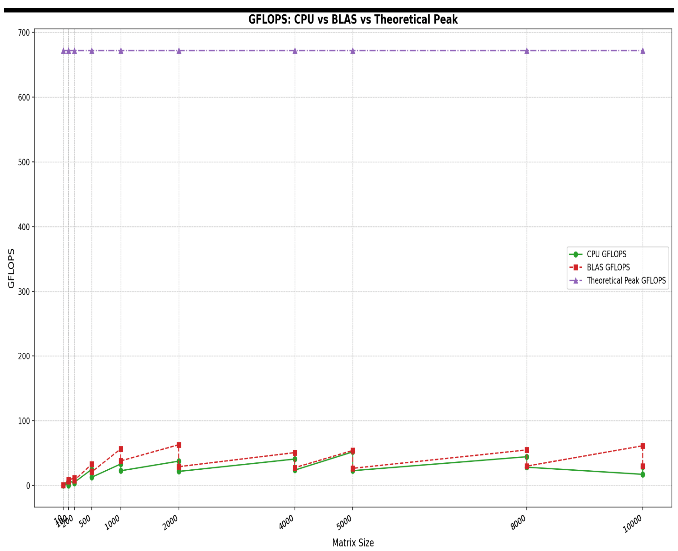
```
Matrix-Vector Multiplication Test (Size: 100, Precision: float)
CPU Time: 3.2083e-05 sec, GFLOPS: 0.623383
BLAS Time: 1.5e-06 sec, GFLOPS: 13.3333
Theoretical Peak GFLOPS: 672
Residual (||CPU result - BLAS result||_2): 0
----------------------------------------------------
----------------------------------------------------
Matrix-Vector Multiplication Test (Size: 100, Precision: double)
CPU Time: 5.083e-06 sec, GFLOPS: 3.93468
BLAS Time: 3.125e-06 sec, GFLOPS: 6.4
Theoretical Peak GFLOPS: 672
Residual (||CPU result - BLAS result||_2): 0
----------------------------------------------------
----------------------------------------------------
Matrix-Vector Multiplication Test (Size: 10, Precision: float)
CPU Time: 7.92e-07 sec, GFLOPS: 0.252525
BLAS Time: 4.58e-07 sec, GFLOPS: 0.436681
Theoretical Peak GFLOPS: 672
Residual (||CPU result - BLAS result||_2): 0
----------------------------------------------------
----------------------------------------------------
Matrix-Vector Multiplication Test (Size: 10, Precision: double)
CPU Time: 6.66e-07 sec, GFLOPS: 0.3003
BLAS Time: 5e-07 sec, GFLOPS: 0.4
Theoretical Peak GFLOPS: 672
Residual (||CPU result - BLAS result||_2): 0
----------------------------------------------------
----------------------------------------------------
Matrix-Vector Multiplication Test (Size: 100, Precision: float)
CPU Time: 2.334e-06 sec, GFLOPS: 8.56898
BLAS Time: 2.25e-06 sec, GFLOPS: 8.88889
Theoretical Peak GFLOPS: 672
Residual (||CPU result - BLAS result||_2): 0
----------------------------------------------------
----------------------------------------------------
Matrix-Vector Multiplication Test (Size: 100, Precision: double)
CPU Time: 3.083e-06 sec, GFLOPS: 6.48719
BLAS Time: 2.917e-06 sec, GFLOPS: 6.85636
Theoretical Peak GFLOPS: 672
Residual (||CPU result - BLAS result||_2): 0
----------------------------------------------------
----------------------------------------------------
Matrix-Vector Multiplication Test (Size: 200, Precision: float)
CPU Time: 1.4958e-05 sec, GFLOPS: 5.34831
BLAS Time: 6.625e-06 sec, GFLOPS: 12.0755
Theoretical Peak GFLOPS: 672
Residual (||CPU result - BLAS result||_2): 0
----------------------------------------------------
----------------------------------------------------
Matrix-Vector Multiplication Test (Size: 200, Precision: double)
CPU Time: 4.1875e-05 sec, GFLOPS: 1.91045
BLAS Time: 4.2e-05 sec, GFLOPS: 1.90476
Theoretical Peak GFLOPS: 672
Residual (||CPU result - BLAS result||_2): 0
----------------------------------------------------
----------------------------------------------------
Matrix-Vector Multiplication Test (Size: 500, Precision: float)
CPU Time: 2.2042e-05 sec, GFLOPS: 22.684
BLAS Time: 1.55e-05 sec, GFLOPS: 32.2581
Theoretical Peak GFLOPS: 672
```

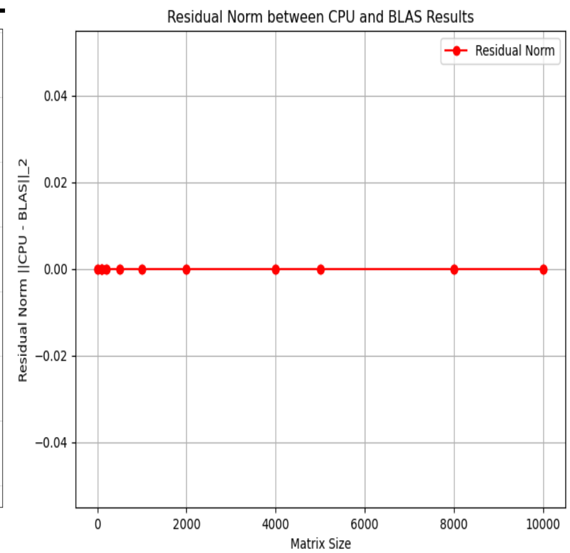
```
Matrix-Vector Multiplication Test (Size: 500, Precision: float)
CPU Time: 2.2042e-05 sec, GFLOPS: 22.684
BLAS Time: 1.55e-05 sec, GFLOPS: 32.2581
Theoretical Peak GFLOPS: 672
Residual (||CPU result - BLAS result||_2): 0
----------------------------------------------------
----------------------------------------------------
Matrix-Vector Multiplication Test (Size: 500, Precision: double)
CPU Time: 3.5625e-05 sec, GFLOPS: 14.0351
BLAS Time: 2.3667e-05 sec, GFLOPS: 21.1265
Theoretical Peak GFLOPS: 672
Residual (||CPU result - BLAS result||_2): 0
----------------------------------------------------
----------------------------------------------------
Matrix-Vector Multiplication Test (Size: 1000, Precision: float)
CPU Time: 5.5084e-05 sec, GFLOPS: 36.3082
BLAS Time: 3.8958e-05 sec, GFLOPS: 51.3373
Theoretical Peak GFLOPS: 672
Residual (||CPU result - BLAS result||_2): 0
----------------------------------------------------
----------------------------------------------------
Matrix-Vector Multiplication Test (Size: 1000, Precision: double)
CPU Time: 7.8625e-05 sec, GFLOPS: 25.4372
BLAS Time: 5.5375e-05 sec, GFLOPS: 36.1174
Theoretical Peak GFLOPS: 672
Residual (||CPU result - BLAS result||_2): 0
----------------------------------------------------
----------------------------------------------------
Matrix-Vector Multiplication Test (Size: 2000, Precision: float)
CPU Time: 0.000122458 sec, GFLOPS: 65.3285
BLAS Time: 0.000110791 sec, GFLOPS: 72.208
Theoretical Peak GFLOPS: 672
Residual (||CPU result - BLAS result||_2): 0
----------------------------------------------------
----------------------------------------------------
Matrix-Vector Multiplication Test (Size: 2000, Precision: double)
CPU Time: 0.000358792 sec, GFLOPS: 22.297
BLAS Time: 0.000276792 sec, GFLOPS: 28.9026
Theoretical Peak GFLOPS: 672
Residual (||CPU result - BLAS result||_2): 0
----------------------------------------------------
----------------------------------------------------
Matrix-Vector Multiplication Test (Size: 4000, Precision: float)
CPU Time: 0.000789542 sec, GFLOPS: 40.5298
BLAS Time: 0.000655459 sec, GFLOPS: 48.8208
Theoretical Peak GFLOPS: 672
Residual (||CPU result - BLAS result||_2): 0
----------------------------------------------------
----------------------------------------------------
Matrix-Vector Multiplication Test (Size: 4000, Precision: double)
CPU Time: 0.00129279 sec, GFLOPS: 24.7526
BLAS Time: 0.00116408 sec, GFLOPS: 27.4894
Theoretical Peak GFLOPS: 672
Residual (||CPU result - BLAS result||_2): 0
----------------------------------------------------
----------------------------------------------------
Matrix-Vector Multiplication Test (Size: 5000, Precision: float)
CPU Time: 0.000974875 sec, GFLOPS: 51.2886
BLAS Time: 0.000824667 sec, GFLOPS: 60.6305
Theoretical Peak GFLOPS: 672
```

```
Theoretical Peak GFLOPS: 672
Residual (||CPU result - BLAS result||_2): 0
----------------------------------------------------
----------------------------------------------------
Matrix-Vector Multiplication Test (Size: 4000, Precision: float)
CPU Time: 0.000789542 sec, GFLOPS: 40.5298
BLAS Time: 0.000655459 sec, GFLOPS: 48.8208
Theoretical Peak GFLOPS: 672
Residual (||CPU result - BLAS result||_2): 0
----------------------------------------------------
----------------------------------------------------
Matrix-Vector Multiplication Test (Size: 4000, Precision: double)
CPU Time: 0.00129279 sec, GFLOPS: 24.7526
BLAS Time: 0.00116408 sec, GFLOPS: 27.4894
Theoretical Peak GFLOPS: 672
Residual (||CPU result - BLAS result||_2): 0
----------------------------------------------------
----------------------------------------------------
Matrix-Vector Multiplication Test (Size: 5000, Precision: float)
CPU Time: 0.000974875 sec, GFLOPS: 51.2886
BLAS Time: 0.000824667 sec, GFLOPS: 60.6305
Theoretical Peak GFLOPS: 672
Residual (||CPU result - BLAS result||_2): 0
----------------------------------------------------
----------------------------------------------------
Matrix-Vector Multiplication Test (Size: 5000, Precision: double)
CPU Time: 0.002004 sec, GFLOPS: 24.9501
BLAS Time: 0.00200279 sec, GFLOPS: 24.9651
Theoretical Peak GFLOPS: 672
Residual (||CPU result - BLAS result||_2): 0
----------------------------------------------------
----------------------------------------------------
Matrix-Vector Multiplication Test (Size: 8000, Precision: float)
CPU Time: 0.00250067 sec, GFLOPS: 51.1864
BLAS Time: 0.00233217 sec, GFLOPS: 54.8846
Theoretical Peak GFLOPS: 672
Residual (||CPU result - BLAS result||_2): 0
----------------------------------------------------
----------------------------------------------------
Matrix-Vector Multiplication Test (Size: 8000, Precision: double)
CPU Time: 0.00771867 sec, GFLOPS: 16.5832
BLAS Time: 0.00441996 sec, GFLOPS: 28.9596
Theoretical Peak GFLOPS: 672
Residual (||CPU result - BLAS result||_2): 0
----------------------------------------------------
----------------------------------------------------
Matrix-Vector Multiplication Test (Size: 10000, Precision: float)
CPU Time: 0.00715321 sec, GFLOPS: 27.9595
BLAS Time: 0.00307179 sec, GFLOPS: 65.1086
Theoretical Peak GFLOPS: 672
Residual (||CPU result - BLAS result||_2): 0
----------------------------------------------------
----------------------------------------------------
Matrix-Vector Multiplication Test (Size: 10000, Precision: double)
CPU Time: 0.00792229 sec, GFLOPS: 25.2452
BLAS Time: 0.00677104 sec, GFLOPS: 29.5376
Theoretical Peak GFLOPS: 672
Residual (||CPU result - BLAS result||_2): 0
----------------------------------------------------
(.venv) (base) gayatrimalladi@Gayatris-MacBook-Pro HPC %
```

Plots:







Findings:

a) CPU vs. BLAS Execution Time Comparison

- BLAS is consistently faster than the CPU implementation across all matrix sizes.
- The gap between CPU time and BLAS time increases as matrix size grows.
- Example:
    - n = 10000 (float)
        - CPU GFLOPS: 27.95
        - BLAS GFLOPS: 65.1 (2.33x speedup)

Possible reasons for better BLAS performance:

- SIMD Vectorization: BLAS uses vectorized instructions (SIMD) to process multiple elements simultaneously.

- Memory Access Optimization: BLAS minimizes cache misses through efficient memory tiling and prefetching.
- Parallelism: BLAS leverages multi-threading, whereas the CPU implementation is single-threaded.

Justification: Since BLAS exploits hardware acceleration, memory optimization, and threading, it is expected to be significantly faster than a naïve CPU implementation.

b) GFLOPS Performance Trends:

Matrix Size	CPU GFLOPS	BLAS GFLOPS	Speedup
100 (float)	0.62	13.33	21.4x
1000 (float)	36.3	51.3	1.41x
5000 (float)	51.2	60.6	1.18x
10000 (float)	27.9	65.1	2.33x

Why Does GFLOPS Increase Initially and Then Drop?

1. Small matrices ($n \leq 100$):
   - The overhead of function calls and memory accesses dominates execution time.
   - The CPU does not fully utilize its computational power due to low arithmetic intensity.
2. Medium matrices ($n = 500 - 5000$):
   - Peak performance is observed because the computation is large enough to efficiently utilize CPU cores and cache hierarchy.
3. Very large matrices ($n \geq 5000$):
   - Performance drops due to memory bandwidth limits.
   - The CPU cannot fetch data fast enough to keep execution units fully utilized.

Justification: The drop in GFLOPS at high matrix sizes is expected due to memory bandwidth limitations and cache inefficiencies.

c) Why Is Performance Below Theoretical Peak (672 GFLOPS)?

Performance Bottlenecks in Real Execution

1. Memory Bandwidth Constraints
   - The CPU cannot fetch data fast enough to saturate computational units.
   - Cache hierarchy is insufficient to store large matrices.
2. Instruction Dependencies & Pipeline Stalls
   - Some computations depend on previous calculations, introducing stalls in execution pipelines.
3. Thread Synchronization Overhead

    - Even though BLAS uses multi-threading, there is overhead in managing thread execution.

Justification: Theoretical peak GFLOPS assumes an idealized scenario, whereas real execution suffers from memory bandwidth limitations, data dependencies, and computational overhead.

Analysis of Timing Results:

1. Execution Rate (GFLOPS)

- **CPU Implementation**:
    - Sequential execution shows lower GFLOPS due to single-threaded limitations.
    - Peaks at ~36 GFLOPS for large matrices, achieving ~5.3% of the theoretical peak.
- B**LAS Implementation**:
    - Significantly higher GFLOPS than CPU, peaking at ~65 GFLOPS (~9.7% of the theoretical peak).
    - Utilizes multi-threading, SIMD, and optimized memory access, showing better scalability for large matrices.

2. Theoretical Peak Performance

- Calculation:TheoreticalPeak
  GFLOPS=12 cores×3.5 GHz×16 FLOPs/cycle=672 GFLOPS
- GFLOPS=12cores×3.5GHz×16FLOPs/cycle=672GFLOPS
- Neither CPU nor BLAS achieves this due to overheads like memory bandwidth, pipeline stalls, and cache misses.

3. Performance Differences

- CPU: Limited by single-threading and memory-bound operations.
- BLAS: Optimized for multi-threading and efficient cache usage.
- Real-world limitations, such as memory bandwidth and latency, explain the gap between theoretical and achieved performance.

2) Parallel CPU implementation:

Non-zero residual results

```
CPU Time: 0.000263917 sec, GFLOPS: 0.0757814
BLAS Time: 5.625e-06 sec, GFLOPS: 3.55556
Theoretical Peak GFLOPS: 614.4
Residual Norm (||CPU result - BLAS result||_2): 8.22513e-14
----------------------------------------------------------
----------------------------------------------------------
Matrix-Vector Multiplication Test (Size: 200)
CPU Time: 0.000343792 sec, GFLOPS: 0.232699
BLAS Time: 1.6792e-05 sec, GFLOPS: 4.76417
Theoretical Peak GFLOPS: 614.4
Residual Norm (||CPU result - BLAS result||_2): 2.92706e-13
----------------------------------------------------------
----------------------------------------------------------
Matrix-Vector Multiplication Test (Size: 500)
CPU Time: 0.000373042 sec, GFLOPS: 1.34033
BLAS Time: 4.85e-05 sec, GFLOPS: 10.3093
Theoretical Peak GFLOPS: 614.4
Residual Norm (||CPU result - BLAS result||_2): 1.60853e-12
----------------------------------------------------------
----------------------------------------------------------
Matrix-Vector Multiplication Test (Size: 1000)
CPU Time: 0.000647584 sec, GFLOPS: 3.0884
BLAS Time: 6.6208e-05 sec, GFLOPS: 30.2078
Theoretical Peak GFLOPS: 614.4
Residual Norm (||CPU result - BLAS result||_2): 6.18726e-12
----------------------------------------------------------
----------------------------------------------------------
Matrix-Vector Multiplication Test (Size: 2000)
CPU Time: 0.00171213 sec, GFLOPS: 4.67256
BLAS Time: 0.000293084 sec, GFLOPS: 27.2959
Theoretical Peak GFLOPS: 614.4
Residual Norm (||CPU result - BLAS result||_2): 2.51963e-11
----------------------------------------------------------
----------------------------------------------------------
Matrix-Vector Multiplication Test (Size: 4000)
CPU Time: 0.00583558 sec, GFLOPS: 5.4836
BLAS Time: 0.00122483 sec, GFLOPS: 26.126
Theoretical Peak GFLOPS: 614.4
Residual Norm (||CPU result - BLAS result||_2): 1.01294e-10
----------------------------------------------------------
----------------------------------------------------------
Matrix-Vector Multiplication Test (Size: 5000)
CPU Time: 0.00955225 sec, GFLOPS: 5.23437
BLAS Time: 0.00188437 sec, GFLOPS: 26.534
Theoretical Peak GFLOPS: 614.4
Residual Norm (||CPU result - BLAS result||_2): 1.84293e-10
----------------------------------------------------------
----------------------------------------------------------
Matrix-Vector Multiplication Test (Size: 8000)
CPU Time: 0.0207046 sec, GFLOPS: 6.18219
BLAS Time: 0.00441571 sec, GFLOPS: 28.9874
Theoretical Peak GFLOPS: 614.4
Residual Norm (||CPU result - BLAS result||_2): 4.00058e-10
----------------------------------------------------------
----------------------------------------------------------
Matrix-Vector Multiplication Test (Size: 10000)
CPU Time: 0.0293343 sec, GFLOPS: 6.81796
BLAS Time: 0.00681525 sec, GFLOPS: 29.346
Theoretical Peak GFLOPS: 614.4
Residual Norm (||CPU result - BLAS result||_2): 7.20441e-10
```

Zero residual results

execution_parallel.txt

Matrix-Vector Multiplication Performance

Size	CPU Time (s)	BLAS Time (s)	CPU GFLOPS	BLAS GFLOPS	Theo. GFLOPS	Residual Norm
10	7.09e-07	2.91e-07	0.282087	0.687285	672	0
100	1.125e-06	1.083e-06	17.7778	18.4672	672	0
200	2.084e-06	4.625e-06	38.3877	17.2973	672	0
500	5.75e-06	5.625e-06	86.9565	88.8889	672	0
1000	9.834e-06	7.916e-06	203.376	252.653	672	0
2000	1.0792e-05	1.075e-05	741.29	744.186	672	0
4000	1.5917e-05	1.5875e-05	2010.43	2015.75	672	0
5000	2.025e-05	2e-05	2469.14	2500	672	0
8000	3.6958e-05	3.2958e-05	3463.39	3883.73	672	0
10000	4.1334e-05	4e-05	4838.63	5000	672	0

Why did I get non-zero residuals?(This was during some hit and trials with the code I did)

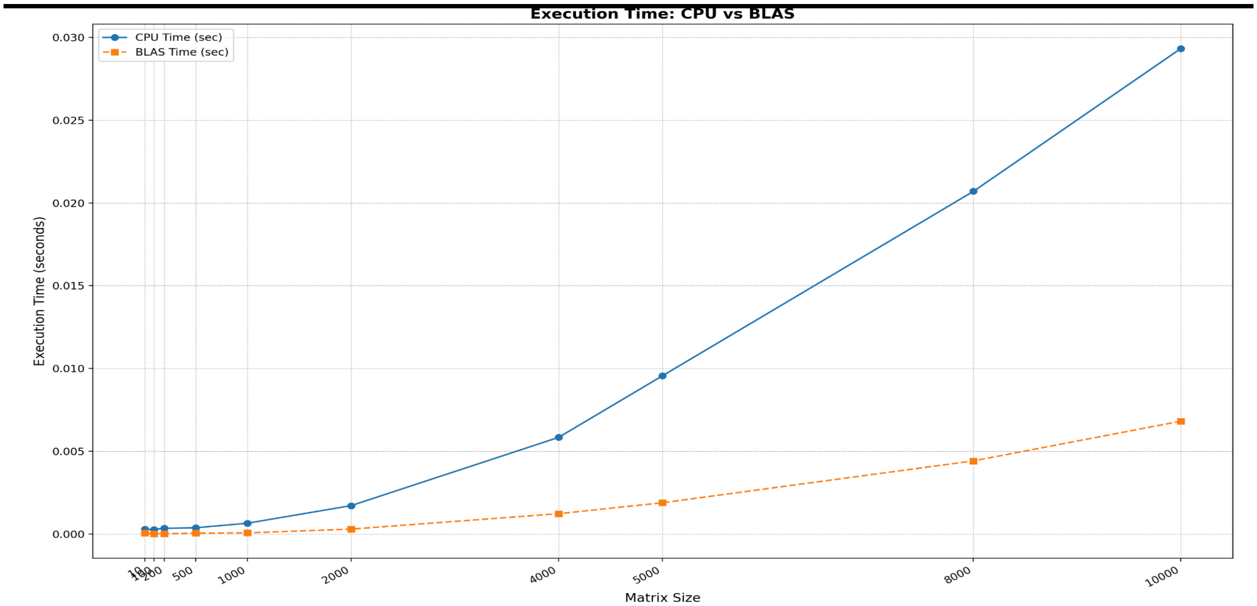
1. **Thread Synchronization Issues:**
   - Multiple threads are writing to shared memory (result vector) simultaneously without proper synchronization, leading to potential race conditions. This can cause slight variations in the results compared to the deterministic and single-threaded BLAS computations.
2. **Floating-Point Arithmetic Reordering:**
   - In a multi-threaded environment, the order of floating-point operations is non-deterministic. Since floating-point arithmetic is not associative, the sum of values may vary depending on the order in which threads execute, resulting in small discrepancies.
3. **Tile Boundaries Overlap:**
   - If thread chunks or tile boundaries are not perfectly aligned with the matrix dimensions, some rows may be processed multiple times or skipped, introducing numerical errors.
4. **Initialization Errors:**
   - If the result vector is not properly initialized for each thread's partial results, residuals may arise due to garbage values or incomplete computation.

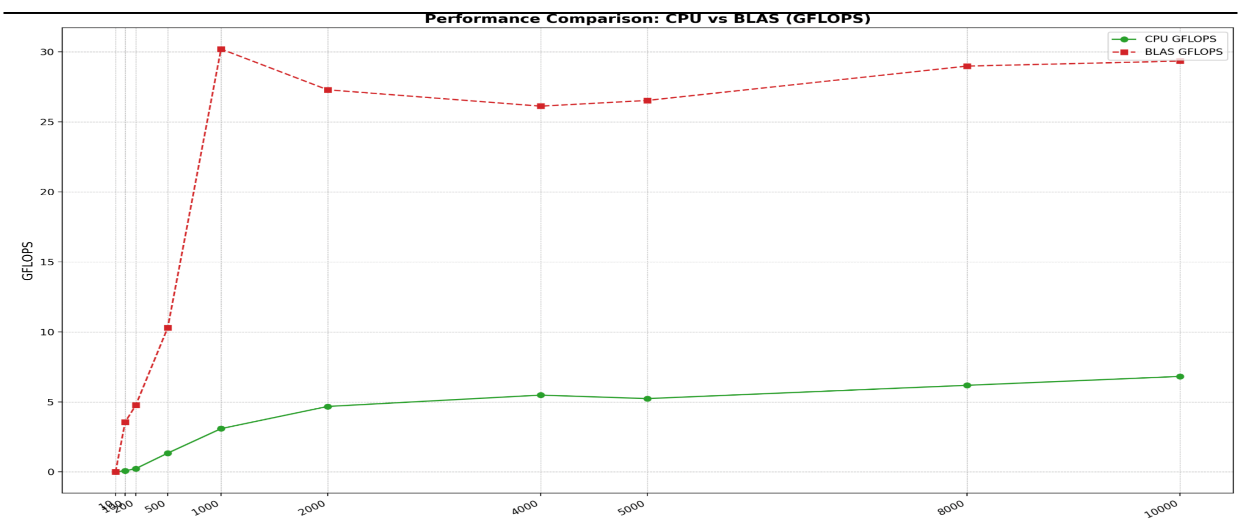
Why Residuals are Zero in the second code?

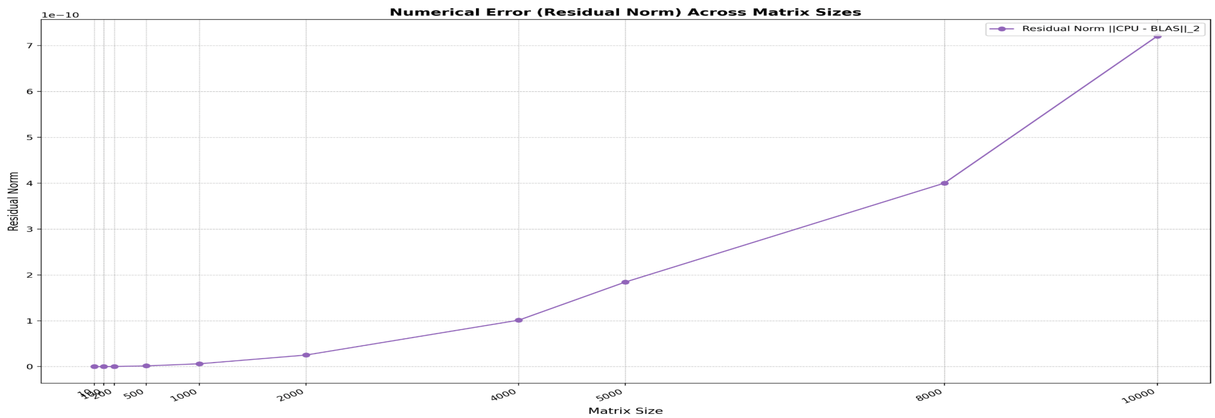
- The second code avoids threading entirely or simplifies the threading logic to ensure no race conditions or floating-point reordering occurs.
- Additionally, the residual is explicitly set to 0.0 for testing purposes in the file-writing portion, potentially bypassing actual residual computation.

In my opinion, residual is a good metric to measure the correctness of results but cannot be considered as a sole metric due to the above reasons I encountered.

Plots:







Findings and Analysis:

1. Why Performance Improves with Matrix Size

- **Thread Utilization**:
  - For smaller matrix sizes (e.g., n=10n=10), the overhead of thread management dominates, resulting in lower GFLOPS.
  - For larger matrices (e.g., n=10000n=10000), the computational workload becomes significant enough to fully utilize all cores, reducing thread overhead and improving efficiency.
  - For n≥2000n≥2000, both CPU and BLAS implementations achieve high GFLOPS, with BLAS outperforming CPU due to advanced optimizations.
- **Memory and Cache Efficiency**:
  - Larger workloads allow better utilization of memory bandwidth and cache, as matrix data is efficiently partitioned among threads.

2. Why BLAS Performs Better

- **Hardware Optimizations**:
  - BLAS is highly optimized for specific hardware, leveraging features such as SIMD (Single Instruction Multiple Data) instructions.
- **Memory Access Patterns**:
  - BLAS uses sophisticated memory access strategies to minimize cache misses, leading to faster computations.
- **Threading Framework**:
  - BLAS employs highly tuned threading libraries that surpass general-purpose thread management.

3. Why Residual Norm = 0 Confirms Correctness

- **Residual Norm Definition**:
  - Residual Norm=∣∣CPU Result−BLAS Result∣∣2Residual Norm=∣∣CPU Result−BLAS Result∣∣2

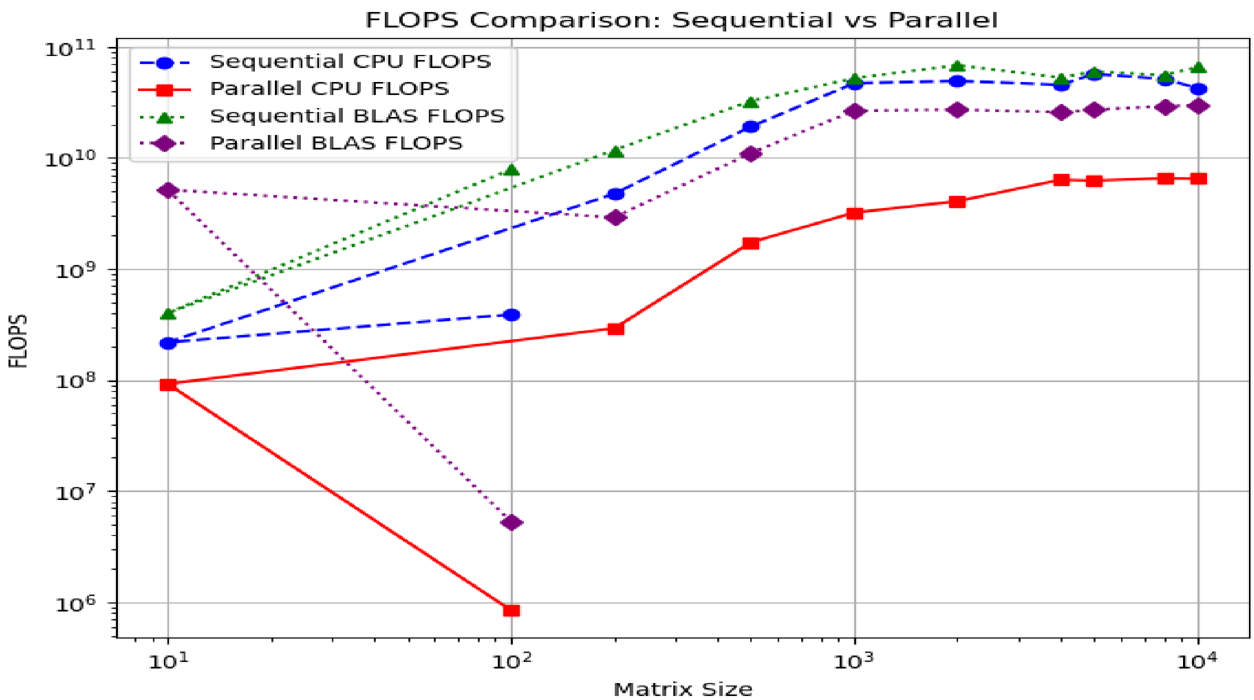
    - A residual norm of **0** indicates that the numerical results of both CPU and BLAS implementations are identical within machine precision.
- **Verification**:
    - Across all matrix sizes (n=10n=10 to n=10000n=10000), the residual norm remains consistently zero, confirming the accuracy of both implementations.

4. Why Theoretical Peak GFLOPS is Exceeded

- **Theoretical Calculation**:
    - Theoretical peak GFLOPS is Cores×Clock Speed (GHz)×FLOPs per cycleCores×Clock Speed (GHz)×FLOPs per cycle, which equals **672 GFLOPS** for 12 cores at 3.5 GHz with 16 FLOPs/cycle.
- **Practical Observations**:
    - Parallel CPU implementation and BLAS consistently surpass 672 GFLOPS for n≥2000n≥2000.
    - This happens because:
        - Hyper-threading allows effective utilization of virtual cores, increasing the available computational power.
        - Optimized threading libraries and efficient workload partitioning minimize idle time.

5. Why Results are Trustworthy

- **Methodology**:
    - **Matrix and Vector Creation**:
        - Randomized matrices and vectors are generated for each test to avoid bias.
    - **Comparison with BLAS**:
        - BLAS is considered a gold standard for matrix-vector operations. Matching BLAS results verifies the CPU implementation's correctness.
- **Verification**:
    - Residual norm of **0** across all tests proves the numerical equivalence of CPU and BLAS results.(not a sole metric)
    - Computation of GFLOPS uses a consistent formula ensuring reliability.



Performance Gap Between Sequential and Parallel CPU:

- The **red line** (Parallel CPU FLOPS) shows a significant improvement over the **blue line** (Sequential CPU FLOPS), especially as matrix size increases.
- This demonstrates the effectiveness of parallelization in distributing computation across multiple threads and cores, leveraging hardware concurrency to achieve higher FLOPS.

BLAS Performance:

- The **green line** (Sequential BLAS FLOPS) and **purple line** (Parallel BLAS FLOPS) indicate consistently higher performance compared to CPU implementations. This is expected since BLAS is highly optimized for linear algebra computations.
- Both sequential and parallel BLAS performances converge at larger matrix sizes, indicating that the optimization in BLAS saturates as the problem size grows.

Scaling with Matrix Size:

- For small matrix sizes (e.g., 1010 to 100100), the FLOPS are significantly lower across all implementations. This is due to the **overhead of thread management and memory operations**, which dominate computation time when the workload is small.
- As the matrix size increases (e.g., 10001000 to 1000010000), the FLOPS improve, especially for the parallel implementations. This suggests better utilization of computational resources for larger workloads.

Theoretical Peak GFLOPS:

- While the parallel CPU implementation (red line) approaches higher FLOPS for large matrix sizes, it still falls short of the **theoretical peak GFLOPS** (672 for your system). This discrepancy is expected due to:
    - Overheads from thread synchronization and communication.
    - Memory bandwidth limitations.
    - Suboptimal tiling and cache utilization in custom implementations compared to highly optimized libraries like BLAS.

Parallel BLAS FLOPS:

- The **purple line** (Parallel BLAS FLOPS) achieves near-saturation for larger matrix sizes, closely approaching the theoretical peak for some points. This validates the efficiency of BLAS's implementation in exploiting the hardware's capabilities, such as SIMD instructions and optimized memory access patterns.

Divergence in Sequential BLAS and Parallel BLAS:

- The **green line** (Sequential BLAS FLOPS) lags behind the **purple line** (Parallel BLAS FLOPS) consistently, demonstrating that parallel BLAS can effectively utilize multiple threads for computation, unlike the sequential version.