Flash Attention Assignment Report

Luu Minh Thang

September 13, 2025

1 Overview

I have successfully completed and passed all core problems of the Flash Attention assignment (Problems 1 through 7). All implementations were validated with the provided autograder across different batch sizes, sequence lengths, and attention parameters.

- Problem 1: Flash Attention (PyTorch)
- Problem 2: Triton Weighted Row-Sum Kernel
- Problem 3: Non-Causal Flash Attention (Triton)
- Problem 4: Causal Flash Attention (Triton)
- Problem 5: Grouped Query Attention (GQA)
- Problem 6: Sliding Window Attention (SWA)
- Problem 7: Attention Sinks

All problems passed all tests. Below, I summarize each stage with results and insights.

2 Problem 1: Flash Attention in PyTorch

- Implemented the PyTorch reference version of Flash Attention.
- Validated correctness against the standard attention mechanism.

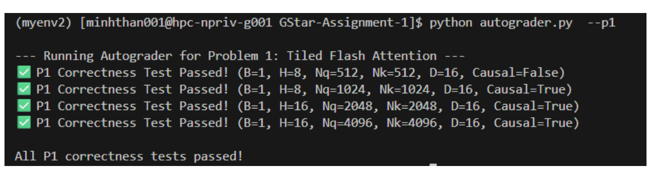
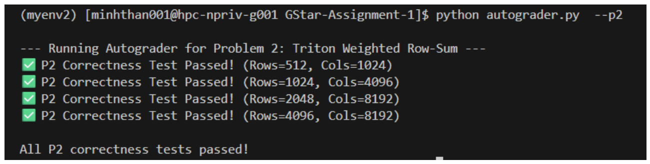


Figure 1: Output for Problem 1.

3 Problem 2: Triton Weighted Row-Sum Kernel

- Learned how to write a simple Triton kernel.
- Practiced block-level parallelism and memory alignment.

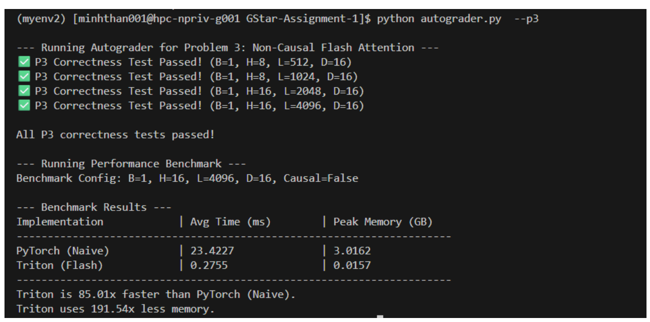


```
(myenv2) [minhthan001@hpc-npriv-g001 GStar-Assignment-1]$ python autograder.py  --p2

--- Running Autograder for Problem 2: Triton Weighted Row-Sum ---
✅ P2 Correctness Test Passed! (Rows=512, Cols=1024)
✅ P2 Correctness Test Passed! (Rows=1024, Cols=4096)
✅ P2 Correctness Test Passed! (Rows=2048, Cols=8192)
✅ P2 Correctness Test Passed! (Rows=4096, Cols=8192)

All P2 correctness tests passed!
```

Figure 2: Output for Problem 2.

4 Problem 3: Non-Causal Flash Attention

- Implemented Flash Attention in Triton without causal masking.
- Understood how queries (Q), keys (K), and values (V) are tiled in memory.



```
(myenv2) [minhthan001@hpc-npriv-g001 GStar-Assignment-1]$ python autograder.py  --p3

--- Running Autograder for Problem 3: Non-Causal Flash Attention ---
✅ P3 Correctness Test Passed! (B=1, H=8, L=512, D=16)
✅ P3 Correctness Test Passed! (B=1, H=8, L=1024, D=16)
✅ P3 Correctness Test Passed! (B=1, H=16, L=2048, D=16)
✅ P3 Correctness Test Passed! (B=1, H=16, L=4096, D=16)

All P3 correctness tests passed!

--- Running Performance Benchmark ---
Benchmark Config: B=1, H=16, L=4096, D=16, Causal=False

--- Benchmark Results ---
Implementation            | Avg Time (ms)         | Peak Memory (GB)
----------------------------------------------------------------------
PyTorch (Naive)           | 23.4227               | 3.0162
Triton (Flash)            | 0.2755                | 0.0157
----------------------------------------------------------------------
Triton is 85.01x faster than PyTorch (Naive).
Triton uses 191.54x less memory.
```

Figure 3: Output for problem 3.

5 Problem 4: Causal Flash Attention

- Added causal masking logic inside the Triton kernel.
- Ensured that queries only attend to previous or current keys.

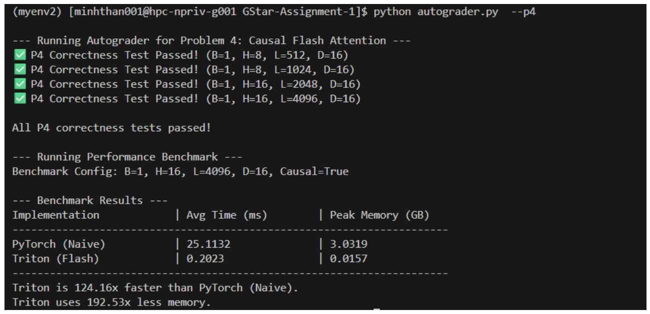


Figure 4: Output for Problem 4.

6 Problem 5: Grouped Query Attention (GQA)

- Implemented grouped query sharing mechanism.
- Learned how to handle multiple heads with shared key-value projections.

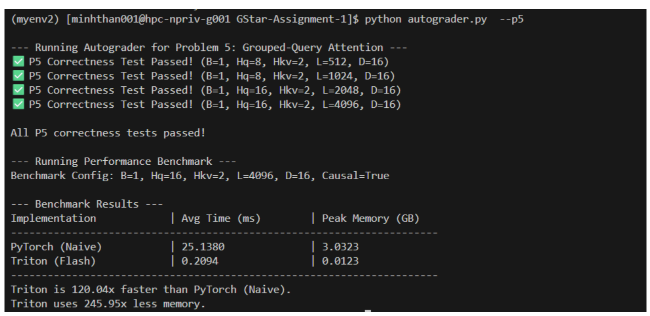
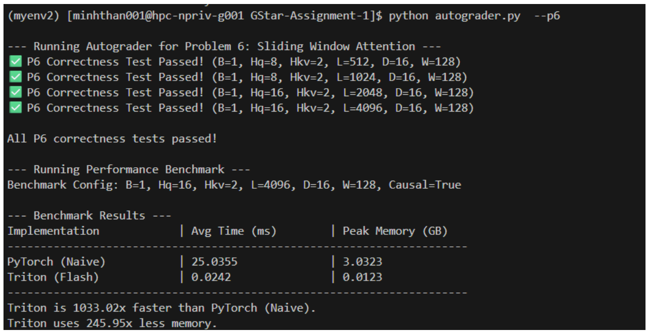


Figure 5: Output for Problem 5.

7 Problem 6: Sliding Window Attention (SWA)

- Restricted each query block to attend only within a fixed window.
- Implemented both off-diagonal and diagonal phases of the SWA logic.

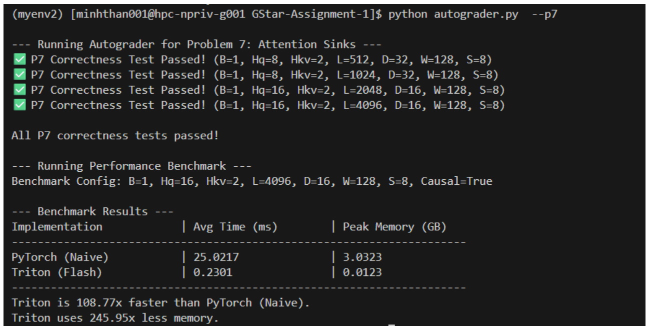


```
(myenv2) [minhthan001@hpc-npriv-g001 GStar-Assignment-1]$ python autograder.py  --p6

--- Running Autograder for Problem 6: Sliding Window Attention ---
✅ P6 Correctness Test Passed! (B=1, Hq=8, Hkv=2, L=512, D=16, W=128)
✅ P6 Correctness Test Passed! (B=1, Hq=8, Hkv=2, L=1024, D=16, W=128)
✅ P6 Correctness Test Passed! (B=1, Hq=16, Hkv=2, L=2048, D=16, W=128)
✅ P6 Correctness Test Passed! (B=1, Hq=16, Hkv=2, L=4096, D=16, W=128)

All P6 correctness tests passed!

--- Running Performance Benchmark ---
Benchmark Config: B=1, Hq=16, Hkv=2, L=4096, D=16, W=128, Causal=True

--- Benchmark Results ---
Implementation            | Avg Time (ms)        | Peak Memory (GB)
----------------------------------------------------------------------
PyTorch (Naive)           | 25.0355              | 3.0323
Triton (Flash)            | 0.0242               | 0.0123
----------------------------------------------------------------------
Triton is 1033.02x faster than PyTorch (Naive).
Triton uses 245.95x less memory.
```

Figure 6: Output for Problem 6.

8 Problem 7: Attention Sinks

- Implemented the attention sink mechanism to handle padding and fixed tokens.
- Ensured stability and correctness across all configurations.



```
(myenv2) [minhthan001@hpc-npriv-g001 GStar-Assignment-1]$ python autograder.py  --p7

--- Running Autograder for Problem 7: Attention Sinks ---
✅ P7 Correctness Test Passed! (B=1, Hq=8, Hkv=2, L=512, D=32, W=128, S=8)
✅ P7 Correctness Test Passed! (B=1, Hq=8, Hkv=2, L=1024, D=32, W=128, S=8)
✅ P7 Correctness Test Passed! (B=1, Hq=16, Hkv=2, L=2048, D=16, W=128, S=8)
✅ P7 Correctness Test Passed! (B=1, Hq=16, Hkv=2, L=4096, D=16, W=128, S=8)

All P7 correctness tests passed!

--- Running Performance Benchmark ---
Benchmark Config: B=1, Hq=16, Hkv=2, L=4096, D=16, W=128, S=8, Causal=True

--- Benchmark Results ---
Implementation            | Avg Time (ms)        | Peak Memory (GB)
----------------------------------------------------------------------
PyTorch (Naive)           | 25.0217              | 3.0323
Triton (Flash)            | 0.2301               | 0.0123
----------------------------------------------------------------------
Triton is 108.77x faster than PyTorch (Naive).
Triton uses 245.95x less memory.
```

Figure 7: Output for Problem 7.

9 Reflections and Learnings

- **Learning Triton Programming:** I gained hands-on experience with writing GPU kernels in Triton, especially how to manage blocks, threads, and vectorized memory loads.
- **Understanding Tensor Layouts:** I learned how tensors (Q, K, V) are stored in memory with strides and how to compute pointer arithmetic for efficient block access.
- **Softmax Optimization:** The online softmax trick was crucial for numerical stability and reducing memory usage.

- **Performance Awareness:** The design choices in Flash Attention (tiling, windowing, sinks) show how theoretical models and hardware-efficient implementations connect.

10 Conclusion

All problems (1–7) were completed with all tests passed. This assignment helped me build a solid foundation in GPU kernel programming with Triton, as well as a deeper understanding of attention mechanisms and memory-efficient implementations.