CSCE 654 HW2 Report

Aditya Kovilur

September 26 2025

1 Introduction

This report analyzes the performance of **DGEMM** and attention matrix computation (with and without DGEMM) on a GPU. The observations have been made through testing on the **Perlmutter** supercomputer with an interactive node.

2 DGEMM Naive

This is a naive implementation for dense matrix multiplication with floating-point operations. The observations were made with block dimensions of [1x1, 4x4, 16x16]. The results are shown in Table 1 and the plot is shown in Figure 1.

3 DGEMM Tiled

This is an optimized implementation for dense matrix multiplication with floating-point operations. Observations were made with tile sizes of [1, 4, 8, 16, 32]. The results are shown in Table 2 and the plot is shown in Figure 1.

Comparison with DGEMM Naive at block_dim = tile = 16 shows that the reduced memory accesses give a significant performance boost. The added loop, however, to create the shared memory matrix adds some overhead in DGEMM Tiled, therefore the growth is not sharp.

Table 1: DGEMM Naive Results

Block Dim	Time (ms)	Rate (GF/s)	Diff
16	1052.639	2040.10	0.000×10^0
4	4145.356	518.05	0.000×10^0
1	55553.379	38.66	0.000×10^0

Table 2: DGEMM Tiled Results

Tile	Time (ms)	Rate (GF/s)	Diff
32	750.849	2860.07	0.000×10^0
16	797.934	2691.31	0.000×10^0
8	1263.939	1699.04	0.000×10^0
4	3576.855	600.38	0.000×10^0
1	132451.750	16.21	0.000×10^0

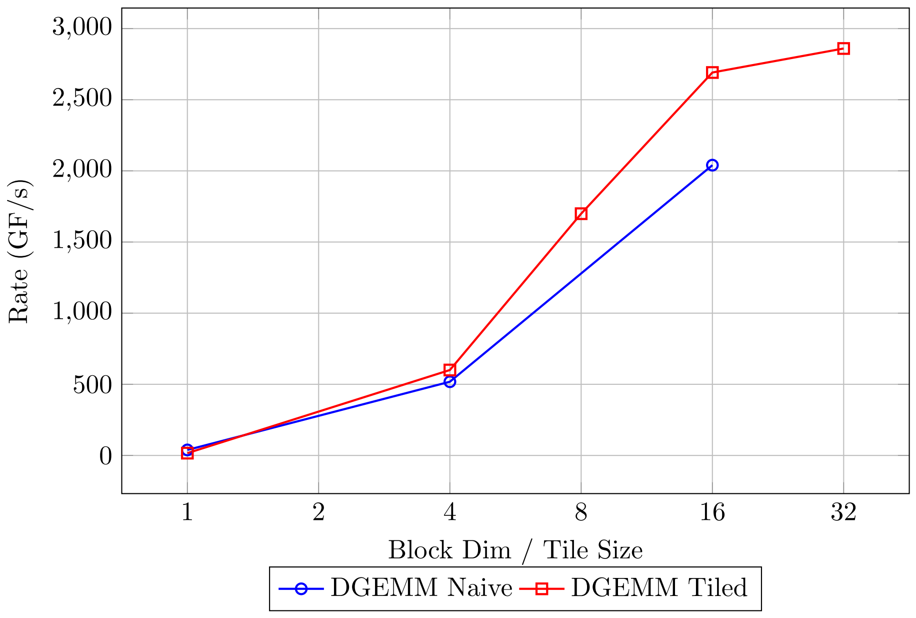

Figure 1: Performance comparison of DGEMM Naive vs Tiled

4 Attention

The computation of the attention matrix involves multiplying elements of 3 matrices. This can be done by abstracting out the matrix multiplication part with DGEMM (Naive or Tiled). The observations are shown in Table 3. The comparison plot is shown in Figure 2. The number of FLOPs is taken as 4 * L * L * D.

As can be observed from Figure 2, the bottleneck does not seem to be the tile/block dimensions since there is not much variance between the FLOP rate. This behavior could be attributed to the softmax function, since the exponential function, though optimized, takes the main chunk of time.

Table 3: Attention Results

Type	Tile/Block	Time (ms)	Rate (GF/s)	Diff
Naive	16	0.401	669.59	1.139×10^{-2}
Tiled	32	0.504	532.85	1.139×10^{-2}
Tiled	16	0.443	605.98	1.139×10^{-2}
Tiled	8	0.516	520.45	1.139×10^{-2}
Tiled	4	0.451	595.44	1.139×10^{-2}
Tiled	1	0.523	513.54	1.139×10^{-2}

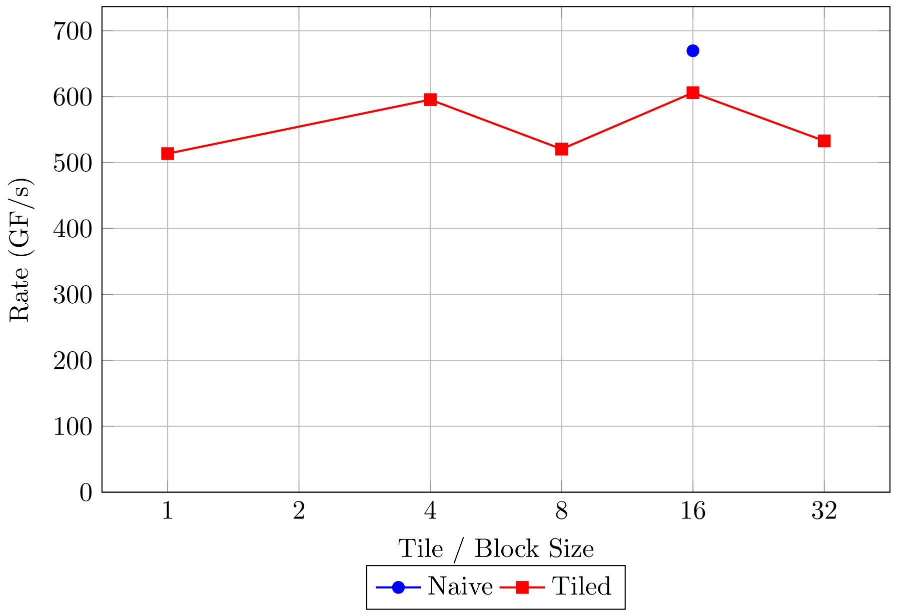

Figure 2: Performance of Attention: Naive vs Tiled