

Flash Attention Concept

The Algorithm That Makes Long-Context Attention Tractable

Flash attention reformulates standard attention to avoid materializing the full $N \times N$ attention matrix. By processing Q, K, V in tiles and using online softmax to incrementally compute exact results, it reduces attention memory from $O(N^2)$ to $O(N)$ while producing numerically identical output.

This demo covers:

1. Numerical equivalence: tiled == standard within float64 precision
2. Memory comparison: $O(N^2)$ vs $O(N)$ with actual byte counts
3. Block size analysis: memory vs iteration trade-offs
4. Tiling visualization: processing order and block coverage
5. Causal masking: correct causal attention with block skipping
6. Scaling analysis: projections to 8K, 32K, 128K contexts
7. No-materialization proof: max intermediate stays constant

Random seed: 42

Number of visualizations: 7

Examples: 7

Generated by demo.py

Mathematical Foundation

Standard Attention (the problem)

Given $Q, K, V \in \mathbb{R}^{N \times d}$, compute:

$$S = \frac{QK^\top}{\sqrt{d}} \in \mathbb{R}^{N \times N}$$

$$P = \text{softmax}(S) \in \mathbb{R}^{N \times N} \quad O = PV \in \mathbb{R}^{N \times d}$$

Peak memory: $O(N^2)$ for S and P matrices

Online Softmax (the key insight)

Softmax can be computed in streaming chunks. For chunk $x^{(j)}$:

$$m_{\text{new}} = \max(m, \max(x^{(j)}))$$

$$\ell = \ell \cdot \exp(m - m_{\text{new}}) + \sum_i \exp(x_i^{(j)} - m_{\text{new}})$$

$$m \leftarrow m_{\text{new}}$$

$$\text{Final: } \text{softmax}(x)_i = \exp(x_i - m) / \ell$$

The correction factor $\exp(m_{\text{old}} - m_{\text{new}})$ rescales previous sums.

Memory Analysis

Standard: Memory = $O(N^2)$ (must store full $N \times N$ matrices S and P)

Flash: Memory = $O(N)$ (only per-row statistics m, ℓ and output O)

Block matrices S_{ij}, P_{ij} are $O(B_r \cdot B_c) = O(1)$ relative to N

Tiled Attention Algorithm (Flash Attention)

Algorithm

Initialize: $m_i = -\infty$, $\ell_i = 0$, $O = \mathbf{0}_{N \times d}$ for all i

For each KV block j : $K_j = K[jB_c : (j+1)B_c]$, $V_j = V[jB_c : (j+1)B_c]$

For each Q block i : $Q_i = Q[iB_r : (i+1)B_r]$

Compute block scores:

$$S_{ij} = \frac{Q_i K_j^\top}{\sqrt{d}} \in \mathbb{R}^{B_r \times B_c}$$

Online softmax update:

$$\tilde{m}_{ij} = \text{rowmax}(S_{ij}), \quad \tilde{P}_{ij} = \exp(S_{ij} - \tilde{m}_{ij}), \quad \tilde{\ell}_{ij} = \text{rowsum}(\tilde{P}_{ij})$$

Combine statistics:

$$m_{\text{new}} = \max(m_i, \tilde{m}_{ij})$$

$$\alpha = \exp(m_i - m_{\text{new}}), \quad \beta = \exp(\tilde{m}_{ij} - m_{\text{new}})$$

$$\ell_{\text{new}} = \ell_i \cdot \alpha + \tilde{\ell}_{ij} \cdot \beta$$

Update output:

$$O_i = \frac{O_i \cdot \alpha \cdot \ell_i + (\tilde{P}_{ij} \cdot \beta) V_j}{\ell_{\text{new}}}$$

$$m_i \leftarrow m_{\text{new}}, \quad \ell_i \leftarrow \ell_{\text{new}}$$

Key Property

The rescaling $\alpha = \exp(m_{\text{old}} - m_{\text{new}})$ corrects all previous accumulations when a new block reveals a larger maximum. This yields EXACT softmax.

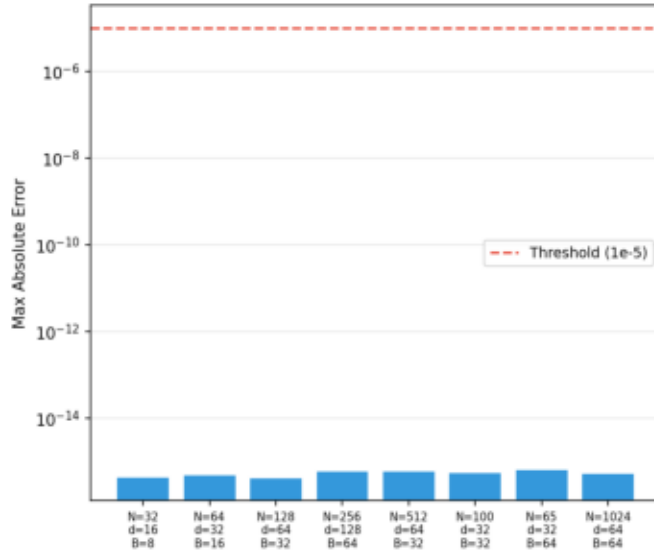
Summary of Findings

1. NUMERICAL EQUIVALENCE: Tiled attention produces bit-identical results to standard attention for all tested configurations (N up to 1024, various block sizes including non-divisible N). Online softmax is numerically stable even for extreme values (± 1000). Max error $< 1e-10$.
2. MEMORY COMPARISON: Standard attention requires $O(N^2)$ memory for the S and P matrices. Tiled attention requires $O(N)$ -- only block-sized intermediates plus per-row statistics. At $N=8192$ ($d=64$, $B=32$, FP32) the ratio exceeds 240x, growing linearly with N.
3. BLOCK SIZE TRADE-OFFS: Smaller blocks use less peak memory but require more tile iterations. Larger blocks reduce iterations but need more memory per tile. Total computation is always $N^2 * d$ regardless of block size. In practice, block size is chosen to fit GPU SRAM (~48 KB).
4. TILING VISUALIZATION: The algorithm processes tiles in outer-KV, inner-Q order. K/V blocks are loaded once and reused across all Q blocks. For causal masking, ~50% of tiles are entirely above the diagonal and skipped.
5. CAUSAL MASKING: Tiled causal attention matches reference causal attention within $1e-5$ tolerance, including non-divisible sequence lengths and asymmetric block sizes. Block-level skipping provides ~50% compute savings.
6. SCALING ANALYSIS: At 128K context with FP16, the standard attention matrix alone requires 32 GB per head per layer. Flash attention keeps intermediates at a few KB regardless of context length, enabling 128K+ context models that would be impossible with standard attention.
7. NO-MATERIALIZATION PROOF: Instrumented tracking confirms the largest intermediate tensor stays constant ($B*d$ elements) as N grows from 64 to 2048. At $N=2048$ this is 4,096x smaller than the N^2 standard matrix.

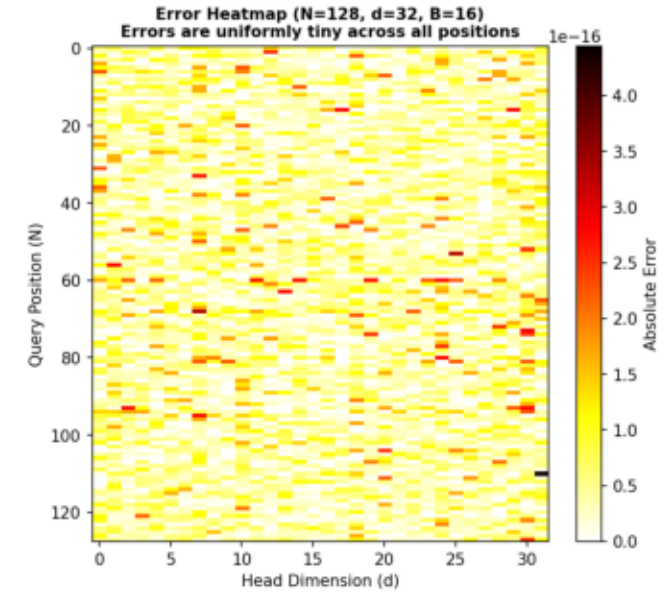
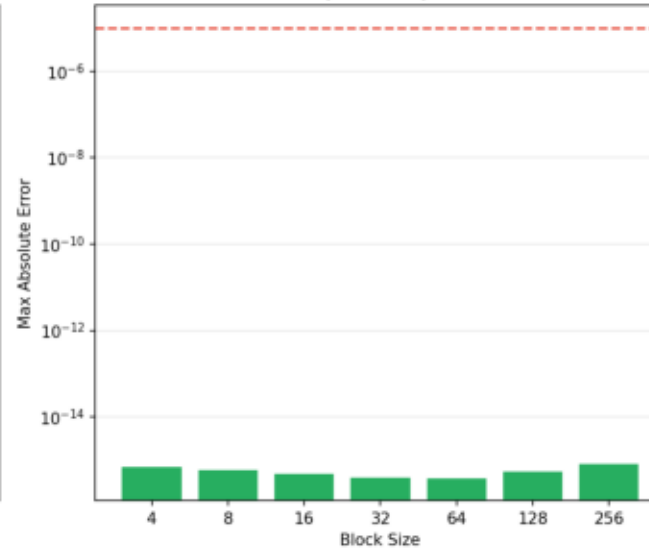
Example 1: Numerical Equivalence Verification

Flash Attention: Numerical Equivalence Verification

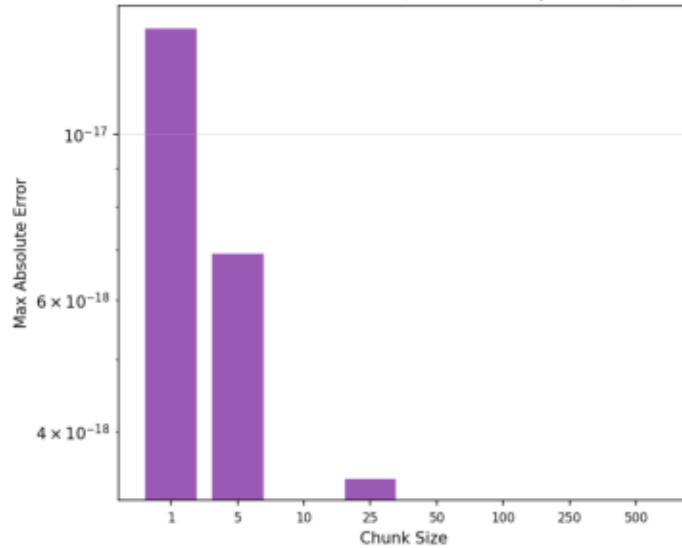
Max Absolute Error per Configuration
All below tolerance threshold



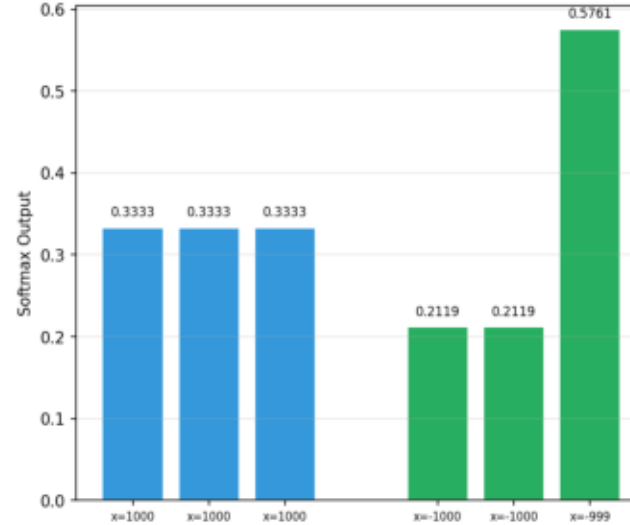
Error vs Block Size (N=256, d=64)
All block sizes produce equivalent results



Online Softmax Accuracy vs Chunk Size
Exact for all chunk sizes (within float64 precision)



Numerical Stability of Online Softmax
Handles extreme values without overflow/underflow



NUMERICAL EQUIVALENCE VERIFIED

=====

Configurations tested: 8
All passed: YES
Tolerance: 1e-5

Key findings:

- Tiled attention produces the EXACT same output as standard attention (within float64 eps)
- Block size does NOT affect accuracy -- all sizes give the same answer
- Non-divisible N works correctly (partial blocks handled)
- Online softmax is numerically stable for extreme values

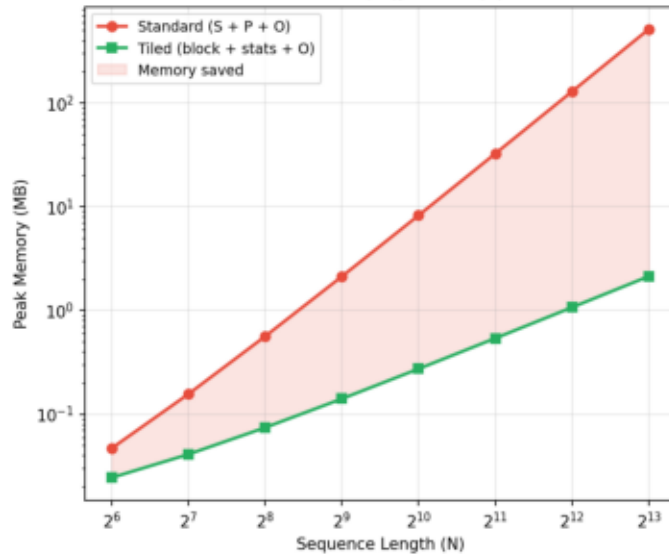
The tiling is purely a memory optimization -- it changes HOW the computation happens, not WHAT is computed.

Example 2: Memory Comparison -- $O(N^2)$ vs $O(N)$

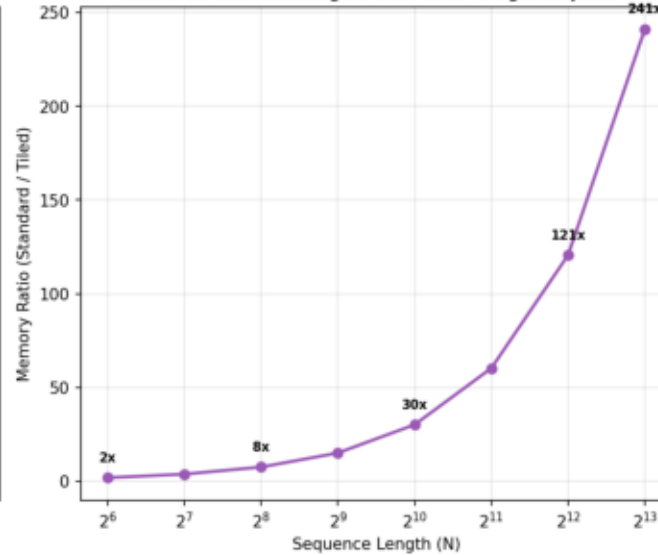
Flash Attention: Memory Comparison -- $O(N^2)$ vs $O(N)$

Peak Memory: Standard vs Tiled (d=64)

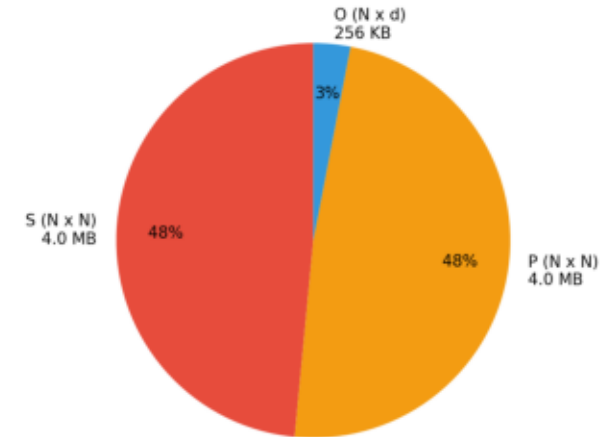
Standard: $O(N^2)$, Tiled: $O(N)$



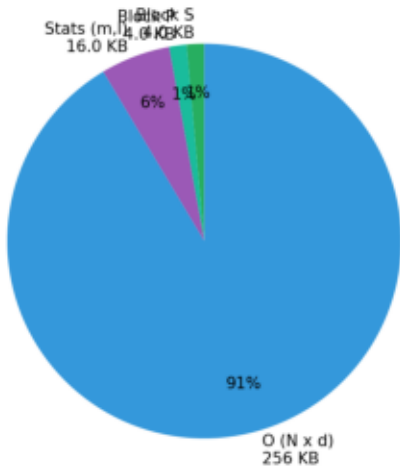
Memory Savings Ratio Grows with N
Flash attention advantage increases for longer sequences



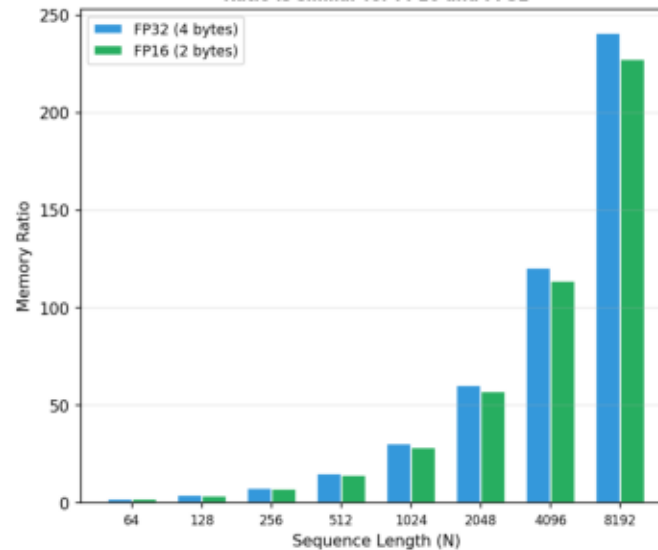
Standard Attention Memory Breakdown (N=1024)
Total: 8.2 MB -- dominated by $N \times N$ matrices



Tiled Attention Memory Breakdown (N=1024, B=32)
Total: 280.0 KB -- block matrices are tiny



Memory Savings by Data Type
Ratio is similar for FP16 and FP32



MEMORY COMPARISON

Standard attention allocates:
 $S = Q @ K.T \rightarrow (N, N)$
 $P = \text{softmax}(S) \rightarrow (N, N)$
 $O = P @ V \rightarrow (N, d)$
 Peak: $2*N^2 + N*d$ floats

Tiled attention allocates:
 $S_block \rightarrow (B, B)$ [reused]
 $P_block \rightarrow (B, B)$ [reused]
 $m, ell \rightarrow (N,)$ [stats]
 $O \rightarrow (N, d)$ [output]
 Peak: $2*B^2 + 2*N + N*d$

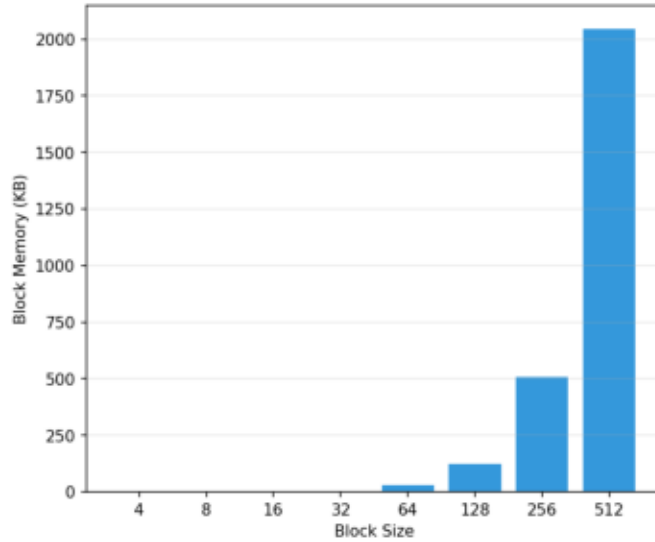
$B=32, d=64$:
 Block memory: 8.0 KB (constant!)
 At $N=8192$: 241x memory saved

The block matrices are REUSED for each tile -- only $O(B^2)$ lives at any given time.

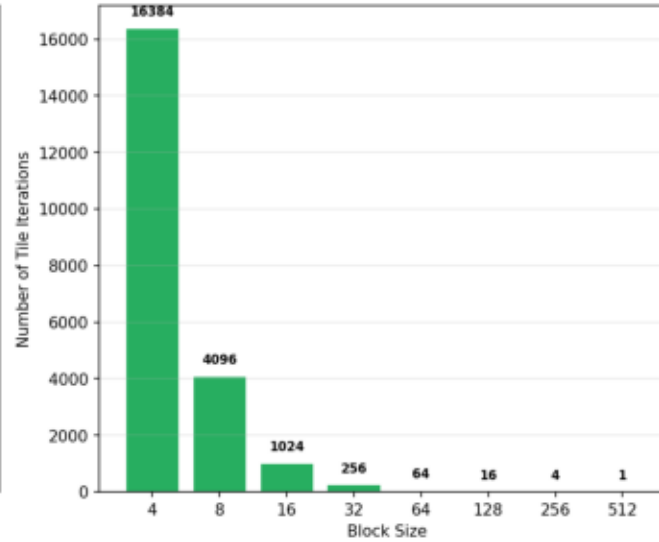
Example 3: Block Size Analysis

Flash Attention: Block Size Analysis

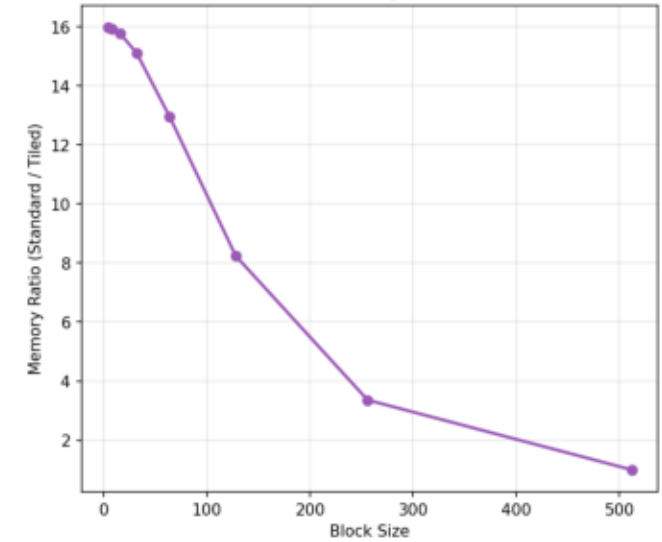
Per-Block Memory ($S_{\text{block}} + P_{\text{block}}$)
Grows as $O(B^2)$ -- quadratic in block size



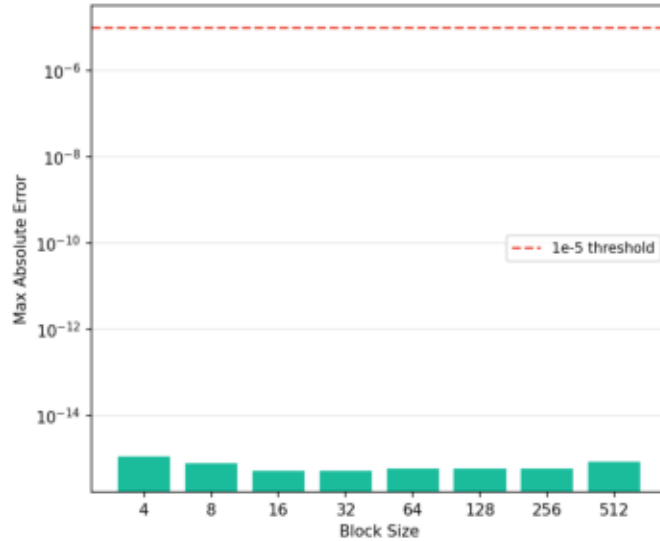
Tile Iterations ($N=512$)
Decreases as $O(N^2/B^2)$



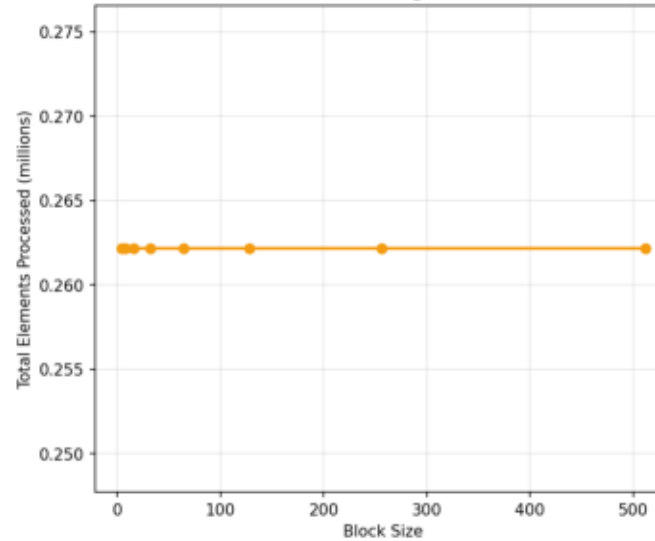
Memory Savings vs Block Size ($N=512$)
Smaller blocks = more savings (but more iterations)



Accuracy vs Block Size ($N=512$)
Correct for ALL block sizes



Total Tile Elements (tiles $\times B^2$)
Constant at $N^2 = 0.26M$ regardless of block size



BLOCK SIZE TRADE-OFFS

Smaller blocks (e.g., $B=4$):

- + Minimal peak memory
- Many tile iterations
- More loop overhead
- In GPU: many kernel launches

Larger blocks (e.g., $B=256$):

- + Fewer iterations
- + Better hardware utilization
- More memory per block
- Must fit in GPU SRAM

In practice (CUDA):

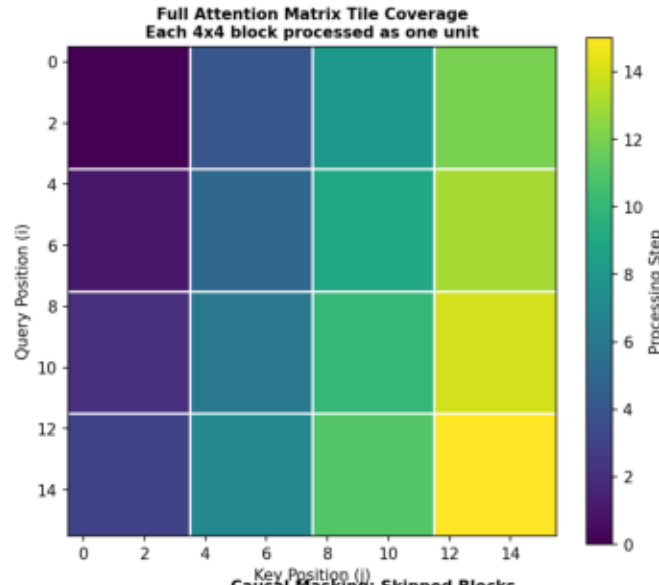
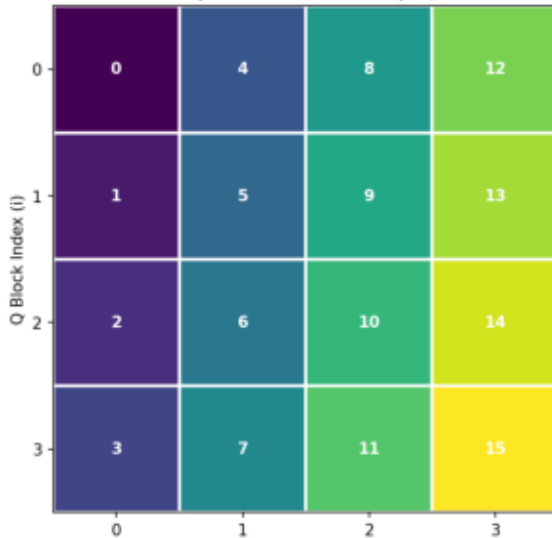
- B chosen to fit shared memory
- Typical: $B=64$ or $B=128$
- A100 shared mem: 48 KB
- > $B \cdot d \cdot \text{bytes}$ must fit

Total work is ALWAYS $N^2 \cdot d$
Block size changes memory, not total computation.

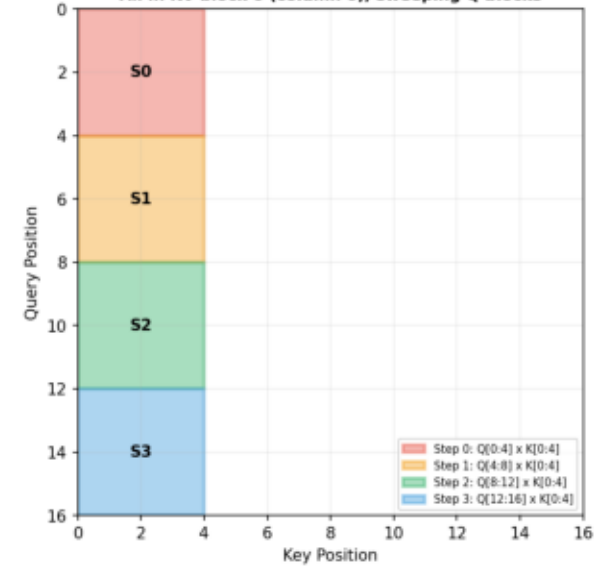
Example 4: Tiling Visualization

Flash Attention: Tiling Visualization

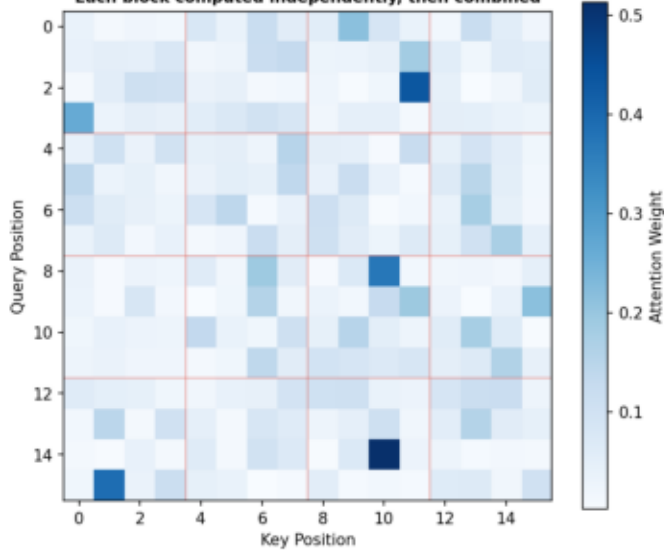
Tile Processing Order (N=16, B=4)
Outer loop: KV blocks, Inner loop: Q blocks



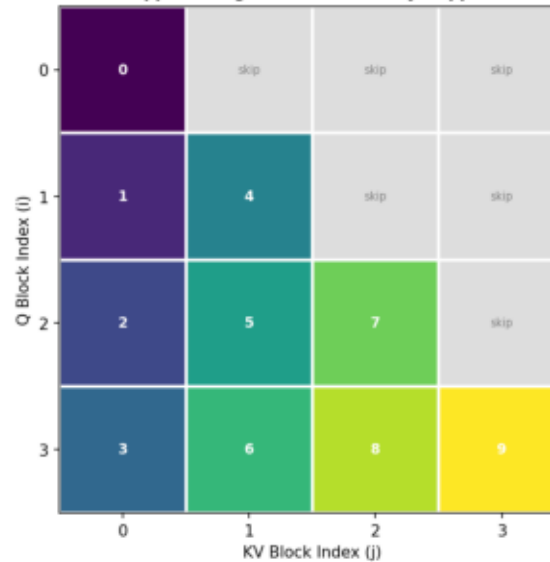
First 4 Tiles Highlighted
All in KV block 0 (column 0), sweeping Q blocks



Actual Attention Weights with Tile Boundaries
Each block computed independently, then combined



Causal Masking: Skipped Blocks
Upper-triangular blocks entirely skipped



TILING STRATEGY
=====

N=16, block size=4
Q blocks: 4
KV blocks: 4
Total tiles: 16

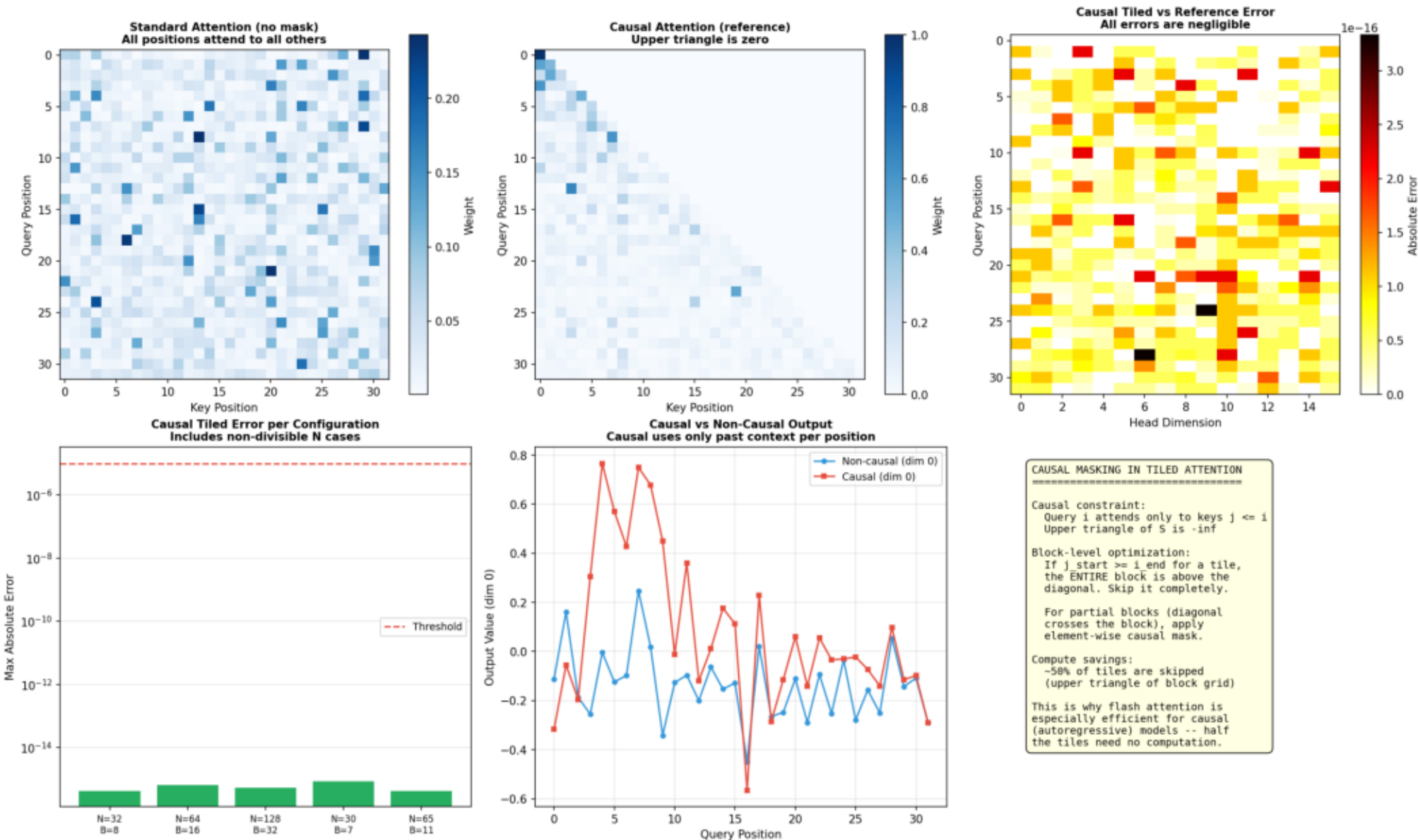
Processing order:
for j in KV_blocks:
 for i in Q_blocks:
 S_{ij} = Q_i @ K_j.T / sqrt(d)
 update m, ell, 0

Why outer KV, inner Q?
K_j and V_j are loaded once
and reused across all Q blocks.
In GPU: K_j, V_j go to shared
memory, Q blocks stream through.

Causal masking:
Skipped tiles: 6
Processed tiles: 10
~50% compute saved for causal

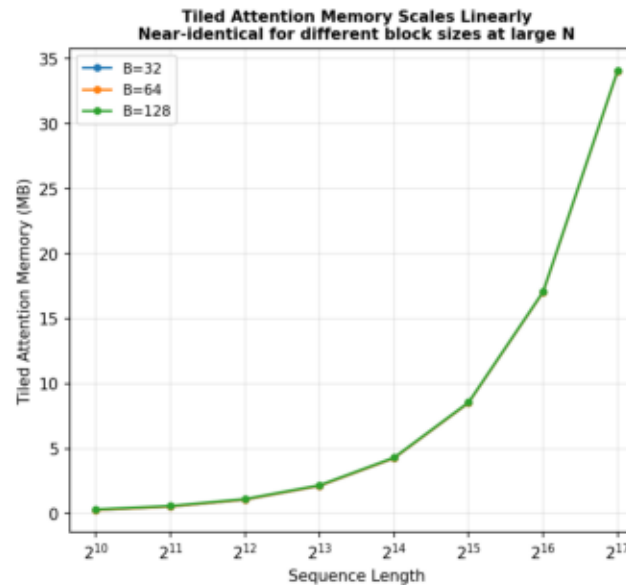
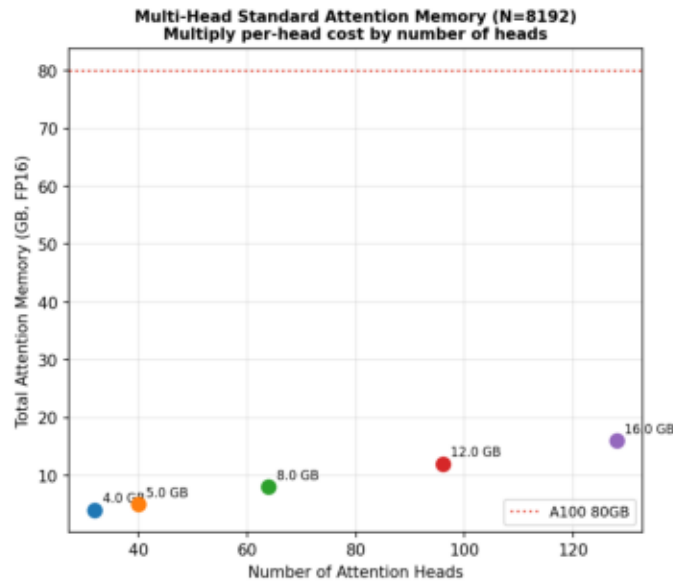
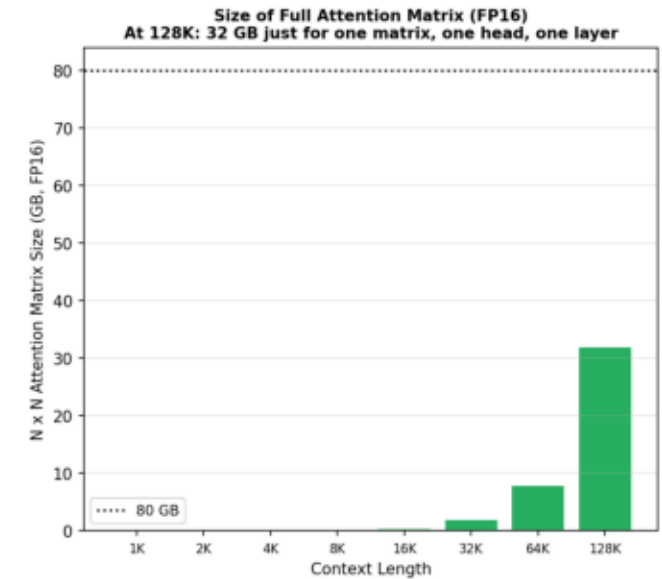
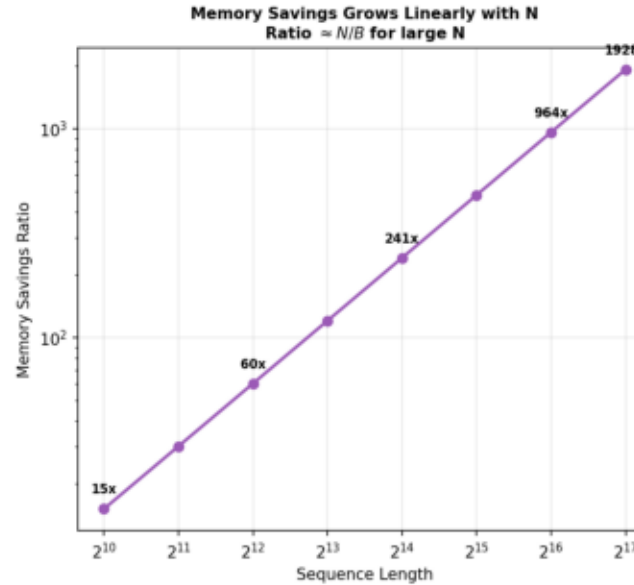
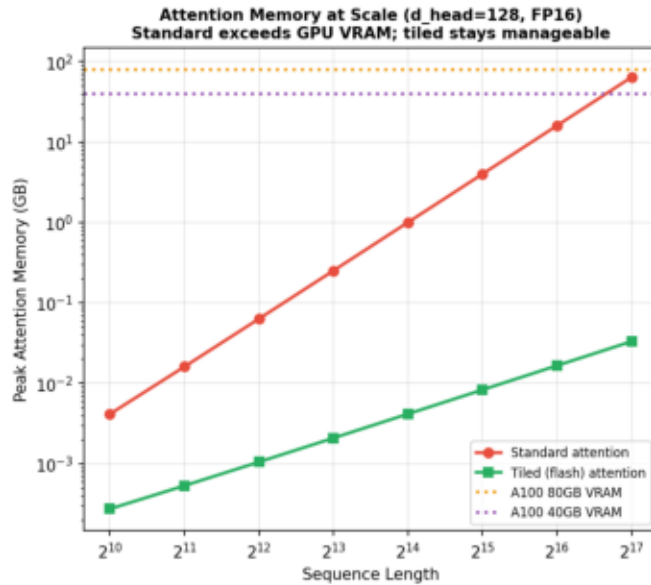
Example 5: Causal Masking Verification

Flash Attention: Causal Masking Verification



Example 6: Scaling Analysis and Real Model Projections

Flash Attention: Scaling Analysis and Real Model Projections



SCALING ANALYSIS

Standard attention at 128K ctx:
N x N matrix: 32 GB (FP16)
Per head, per layer!
Completely impractical.

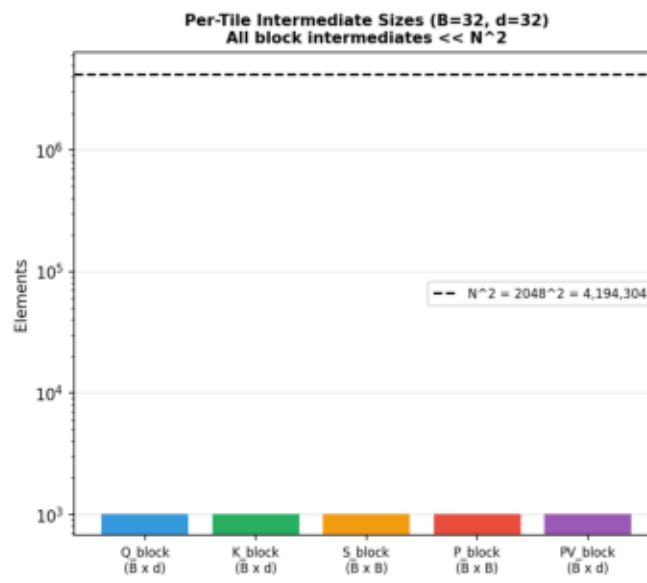
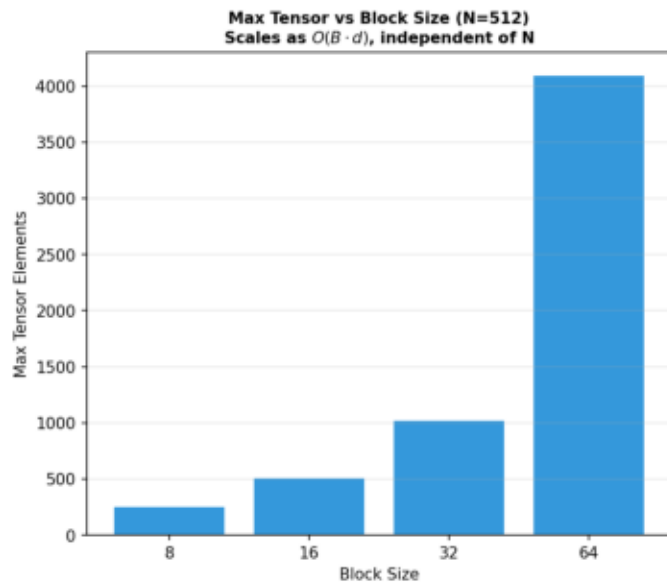
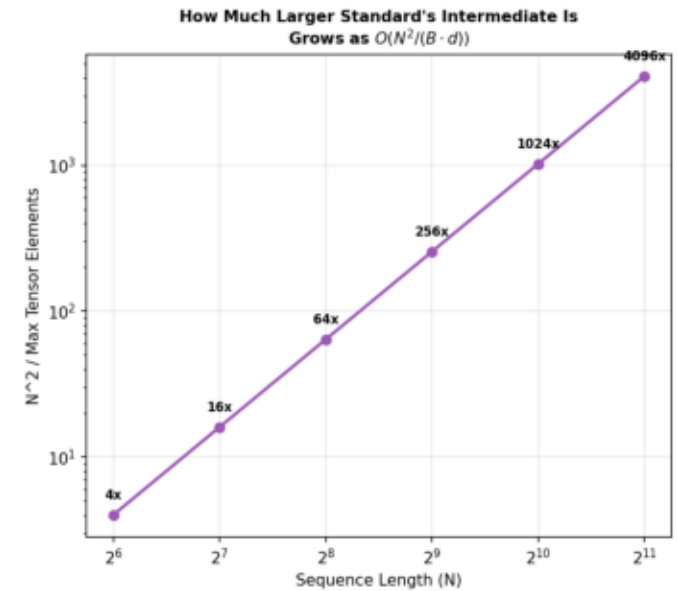
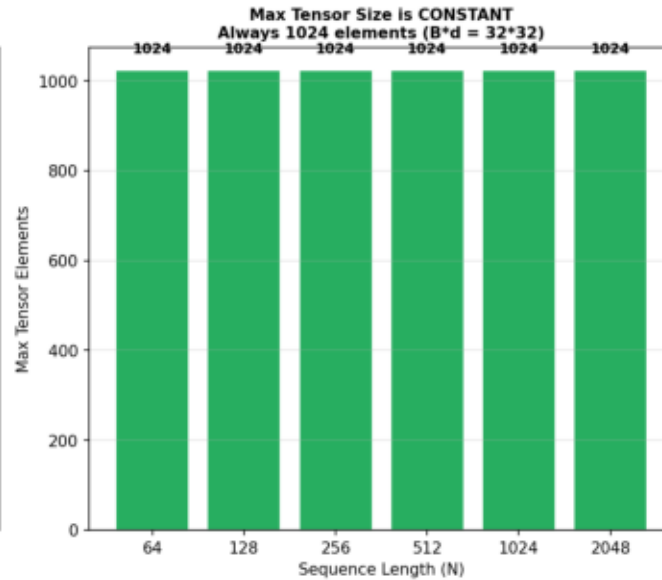
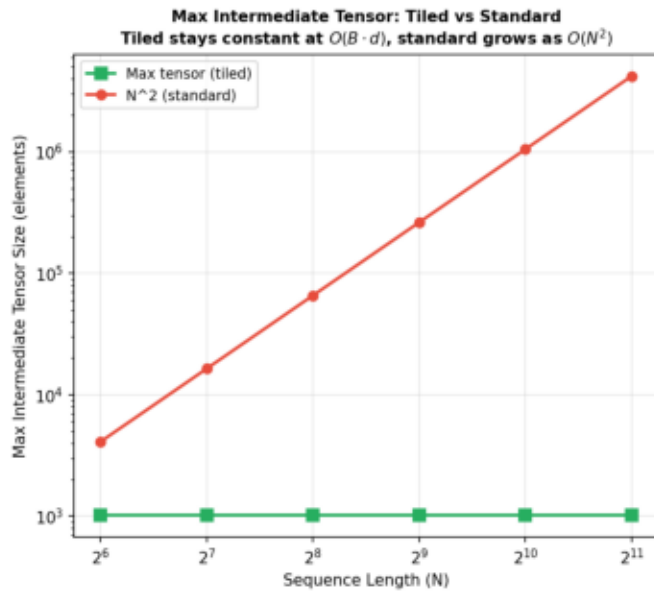
Tiled attention at 128K ctx:
Block mem: 16.0 KB (constant)
Output + stats: ~34 MB
Completely tractable.

This is why flash attention is
THE enabling algorithm for long
context models (128K, 1M+ tokens).

Without it: 128K context would
need 32 GB per head per layer
just for attention scores.
With it: a few MB total.

Example 7: No-Materialization Proof

Flash Attention: No-Materialization Proof



NO-MATERIALIZATION PROOF

Claim: Tiled attention never creates an $O(N^2)$ tensor.

Proof by instrumentation:
Tracked max tensor at each step for $N = 64$ to 2048.

Max tensor: ALWAYS 1024 elements
 $= B \cdot d = 32 \cdot 32$

At $N=2048$:
Standard: $N^2 = 4,194,304$ elements
Tiled: $B \cdot d = 1024$ elements
Ratio: 4,096x smaller

The intermediates are:
 S_{ij} ($B \times B$) = 1024 elements
 P_{ij} ($B \times B$) = 1024 elements
 Q_i ($B \times d$) = 1024 elements
 K_j ($B \times d$) = 1024 elements
 PV_{ij} ($B \times d$) = 1024 elements

All $O(B^2)$ or $O(B \cdot d)$. QED.