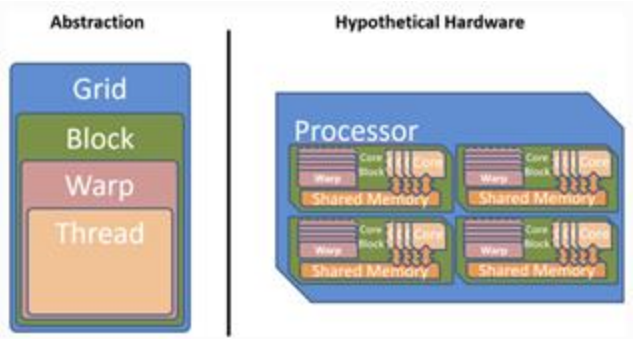
FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning

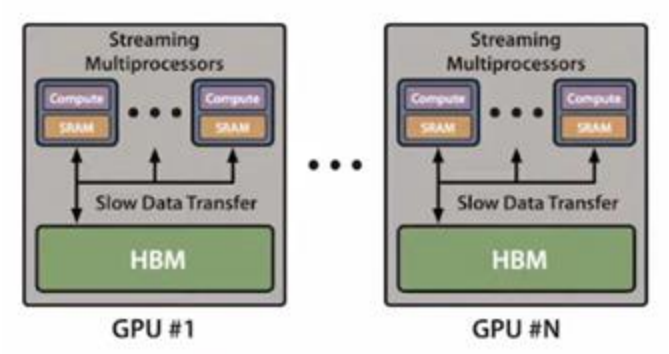
Author: Tri Dao
11/13 2025
Presented by Shuang Ma

GPU Execution Model

A massive number of threads to execute an operation (kernel)

- Threads are grouped into **warps**.
- Threads within a warp can communicate by fast shuffle instructions or cooperate to perform matrix multiply **(quick)**.
- Warps within a thread block can communicate by reading from / writing to shared memory **(slow)**.



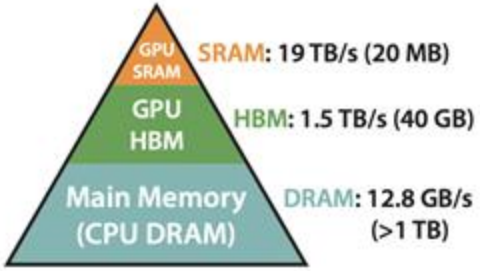


GPU Memory Hierarchy

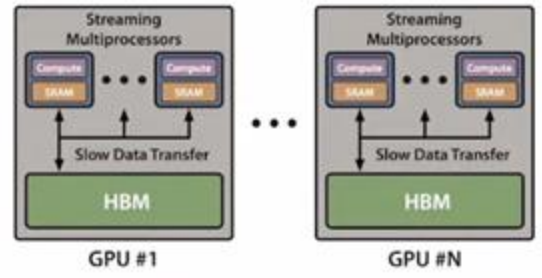
Comprise of high bandwidth memory (**HBM**), and on-chip **SRAM** (aka shared memory).

Before execute an operation:

- **Load input from** HBM to registers and SRAM
- Computes
- **Load output to HBM**



Memory Hierarchy with Bandwidth & Memory Size



Why Flash Attention v2 is proposed?

Language models with much longer context:

- GPT-4 [12] with context length 32k, MosaicML's MPT with context length 65k, and Anthropic's Claude with context length 100k.

FlashAttention (v1) is still not as efficient as other primitives (GEMM):

- Modern GPUs have specialized compute units (e.g., Tensor Cores on Nvidia GPUs) that **makes matmul (matrix multiply) much faster.**
- The forward pass only reaches 30-50% of the maximum throughput
- Optimized GEMM can reach up to 80-90% of the theoretical maximum device throughput

What does FlashAttention v2 do?

- Algorithm: Tweak the algorithm from FlashAttention to reduce the number of non-matmul FLOPs:

  Non-matmul FLOPs is relatively more inefficient.

- Parallelism: Additionally parallelize over the sequence length:

  Especially useful for long context.

- Work Partitioning: Decide how to partition the work between different warps.

    Avoid communication between warp.

Recall: What does FlashAttention v1 do?

Standard softmax:

$$\mathbf{S} = \mathbf{Q}\mathbf{K}^\top \in \mathbb{R}^{N\times N},$$

$$\mathbf{P} = \mathrm{softmax}(\mathbf{S}) \in \mathbb{R}^{N\times N},$$

$$\mathbf{O} = \mathbf{P}\mathbf{V} \in \mathbb{R}^{N\times d},$$

$$m = \max(\mathrm{rowmax}(\mathbf{S}^{(1)}), \mathrm{rowmax}(\mathbf{S}^{(2)})) \in \mathbb{R}^{B_r}$$

$$\ell = \mathrm{rowsum}(e^{\mathbf{S}^{(1)}-m}) + \mathrm{rowsum}(e^{\mathbf{S}^{(2)}-m}) \in \mathbb{R}^{B_r}$$

$$\mathbf{P} = \begin{bmatrix}\mathbf{P}^{(1)} & \mathbf{P}^{(2)}\end{bmatrix} = \mathrm{diag}(\ell)^{-1}\begin{bmatrix}e^{\mathbf{S}^{(1)}-m} & e^{\mathbf{S}^{(2)}-m}\end{bmatrix} \in \mathbb{R}^{B_r\times 2B_c}$$

$$\mathbf{O} = \begin{bmatrix}\mathbf{P}^{(1)} & \mathbf{P}^{(2)}\end{bmatrix}\begin{bmatrix}\mathbf{V}^{(1)}\\ \mathbf{V}^{(2)}\end{bmatrix} = \mathrm{diag}(\ell)^{-1}e^{\mathbf{S}^{(1)}-m}\mathbf{V}^{(1)} + e^{\mathbf{S}^{(2)}-m}\mathbf{V}^{(2)} \in \mathbb{R}^{B_r\times d}.$$

Online softmax:

Instead computes “local” softmax with respect to each block and rescale to get the right output at the end:

$$m^{(1)} = \mathrm{rowmax}(\mathbf{S}^{(1)}) \in \mathbb{R}^{B_r}$$

$$\ell^{(1)} = \mathrm{rowsum}(e^{\mathbf{S}^{(1)}-m^{(1)}}) \in \mathbb{R}^{B_r}$$

$$\tilde{\mathbf{P}}^{(1)} = \mathrm{diag}(\ell^{(1)})^{-1}e^{\mathbf{S}^{(1)}-m^{(1)}} \in \mathbb{R}^{B_r\times B_c}$$

$$\mathbf{O}^{(1)} = \tilde{\mathbf{P}}^{(1)}\mathbf{V}^{(1)} = \mathrm{diag}(\ell^{(1)})^{-1}e^{\mathbf{S}^{(1)}-m^{(1)}}\mathbf{V}^{(1)} \in \mathbb{R}^{B_r\times d}$$

“local” softmax

$$m^{(2)} = \max(m^{(1)}, \mathrm{rowmax}(\mathbf{S}^{(2)})) = m$$

$$\ell^{(2)} = e^{m^{(1)}-m^{(2)}}\ell^{(1)} + \mathrm{rowsum}(e^{\mathbf{S}^{(2)}-m^{(2)}}) = \mathrm{rowsum}(e^{\mathbf{S}^{(1)}-m}) + \mathrm{rowsum}(e^{\mathbf{S}^{(2)}-m}) = \ell$$

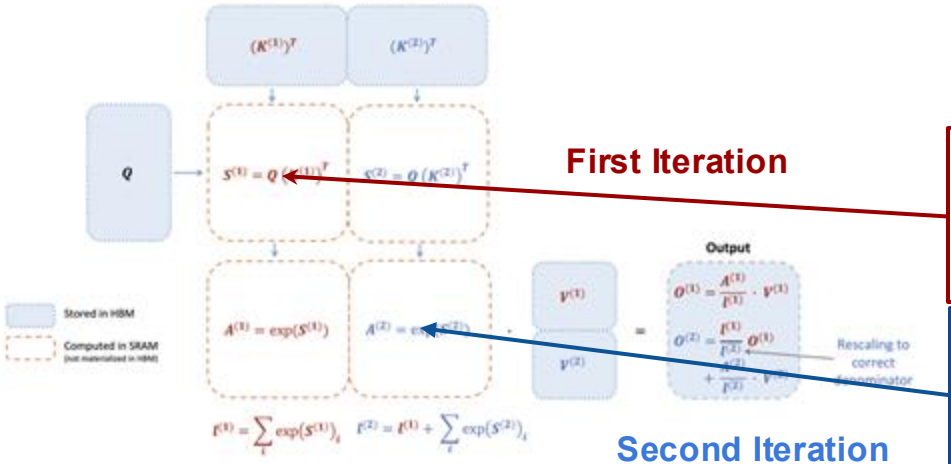
$$\tilde{\mathbf{P}}^{(2)} = \mathrm{diag}(\ell^{(2)})^{-1}e^{\mathbf{S}^{(2)}-m^{(2)}}$$

$$\mathbf{O}^{(2)} = \mathrm{diag}(\ell^{(1)}/\ell^{(2)})^{-1}\mathbf{O}^{(1)} + \tilde{\mathbf{P}}^{(2)}\mathbf{V}^{(2)} = \mathrm{diag}(\ell^{(2)})^{-1}e^{s^{(1)}-m}\mathbf{V}^{(1)} + \mathrm{diag}(\ell^{(2)})^{-1}e^{s^{(2)}-m}\mathbf{V}^{(2)} = \mathbf{O}.$$

rescale

Recall: What does FlashAttention v1 do?

Online softmax:



First Iteration

Second Iteration

By computing attention with respect to each block and rescaling the output, we get the right answer at the end, while **avoiding expensive memory reads/writes of the intermediate matrices S and P.**

$$m^{(1)} = \text{rowmax}(\mathbf{S}^{(1)}) \in \mathbb{R}^{B_r}$$
$$\ell^{(1)} = \text{rowsum}(e^{\mathbf{S}^{(1)} - m^{(1)}}) \in \mathbb{R}^{B_r}$$
$$\tilde{\mathbf{P}}^{(1)} = \text{diag}(\ell^{(1)})^{-1} e^{\mathbf{S}^{(1)} - m^{(1)}} \in \mathbb{R}^{B_r \times B_c}$$
$$\mathbf{O}^{(1)} = \tilde{\mathbf{P}}^{(1)} \mathbf{V}^{(1)} = \text{diag}(\ell^{(1)})^{-1} e^{\mathbf{S}^{(1)} - m^{(1)}} \mathbf{V}^{(1)} \in \mathbb{R}^{B_r \times d}$$

$$m^{(2)} = \max(m^{(1)}, \text{rowmax}(\mathbf{S}^{(2)})) = m$$
$$\ell^{(2)} = e^{m^{(1)} - m^{(2)}} \ell^{(1)} + \text{rowsum}(e^{\mathbf{S}^{(2)} - m^{(2)}}) = \text{rowsum}(e^{\mathbf{S}^{(1)} - m}) + \text{rowsum}(e^{\mathbf{S}^{(2)} - m}) = \ell$$
$$\tilde{\mathbf{P}}^{(2)} = \text{diag}(\ell^{(2)})^{-1} e^{\mathbf{S}^{(2)} - m^{(2)}}$$
$$\mathbf{O}^{(2)} = \text{diag}(\ell^{(1)} / \ell^{(2)})^{-1} \mathbf{O}^{(1)} + \tilde{\mathbf{P}}^{(2)} \mathbf{V}^{(2)} = \text{diag}(\ell^{(2)})^{-1} e^{s^{(1)} - m} \mathbf{V}^{(1)} + \text{diag}(\ell^{(2)})^{-1} e^{s^{(2)} - m} \mathbf{V}^{(2)} = \mathbf{O}.$$

FlashAttention2: Minor tweak to reduce non-matmul FLOPs

V1

$$m^{(1)} = \text{rowmax}(\mathbf{S}^{(1)}) \in \mathbb{R}^{B_r}$$

$$\ell^{(1)} = \text{rowsum}(e^{\mathbf{S}^{(1)} - m^{(1)}}) \in \mathbb{R}^{B_r}$$

$$\tilde{\mathbf{P}}^{(1)} = \text{diag}(\ell^{(1)})^{-1} e^{\mathbf{S}^{(1)} - m^{(1)}} \in \mathbb{R}^{B_r \times B_c}$$

$$\mathbf{O}^{(1)} = \tilde{\mathbf{P}}^{(1)} \mathbf{V}^{(1)} = \text{diag}(\ell^{(1)})^{-1} e^{\mathbf{S}^{(1)} - m^{(1)}} \mathbf{V}^{(1)} \in \mathbb{R}^{B_r \times d}$$

V2: Maintain un-scaled Version

$$\tilde{\mathbf{O}}^{(1)} = e^{\mathbf{S}^{(1)} - m^{(1)}} \mathbf{V}^{(1)} \in \mathbb{R}^{B_r \times d}$$

$$m^{(2)} = \max(m^{(1)}, \text{rowmax}(\mathbf{S}^{(2)})) = m$$

$$\ell^{(2)} = e^{m^{(1)} - m^{(2)}} \ell^{(1)} + \text{rowsum}(e^{\mathbf{S}^{(2)} - m^{(2)}}) = \text{rowsum}(e^{\mathbf{S}^{(1)} - m}) + \text{rowsum}(e^{\mathbf{S}^{(2)} - m}) = \ell$$

$$\tilde{\mathbf{P}}^{(2)} = \text{diag}(\ell^{(2)})^{-1} e^{\mathbf{S}^{(2)} - m^{(2)}}$$

$$\mathbf{O}^{(2)} = \text{diag}(\ell^{(1)} / \ell^{(2)})^{-1} \mathbf{O}^{(1)} + \tilde{\mathbf{P}}^{(2)} \mathbf{V}^{(2)} = \text{diag}(\ell^{(2)})^{-1} e^{s^{(1)} - m} \mathbf{V}^{(1)} + \text{diag}(\ell^{(2)})^{-1} e^{s^{(2)} - m} \mathbf{V}^{(2)} = \mathbf{O}.$$

V2: Maintain un-scaled Version

$$\tilde{\mathbf{O}}^{(2)} = \text{diag}(e^{m^{(1)} - m^{(2)}}) \tilde{\mathbf{O}}^{(1)} + e^{\mathbf{S}^{(2)} - m^{(2)}} \mathbf{V}^{(2)} = e^{s^{(1)} - m} \mathbf{V}^{(1)} + e^{s^{(2)} - m} \mathbf{V}^{(2)}$$

V2: Scale the final to get the right output

$$\mathbf{O}^{(2)} = \text{diag}(\ell^{(2)})^{-1} \tilde{\mathbf{O}}^{(2)} = \mathbf{O}.$$

FlashAttention2: Parallelism

FlashAttention1 parallelism over batch size and number of head.

- There are BS * #head thread blocks, each running on a Streaming multiprocessor.

V1: Parallelize over batch size and number of head.

V2: Parallelize over the b. h. and seq length dimension.

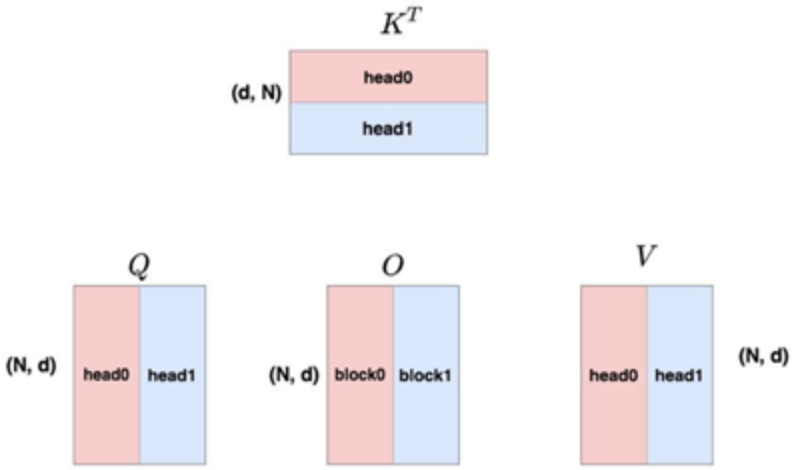
How to achieve parallelism over seq length?

V2: Swap order of loop

FlashAttention 1: How to achieve parallelism?

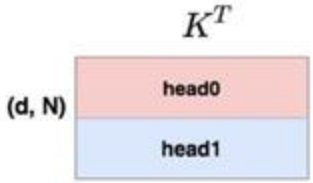
FlashAttention v1:

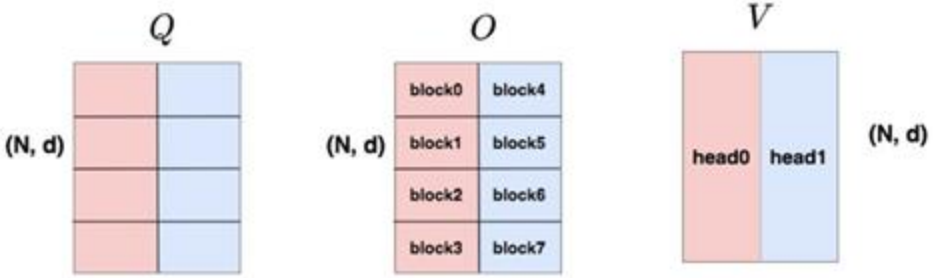
For j-th (K, V), for i-th Q, computer using Kj, Vj, Qi in SRAM, and update Oi, li, mi in HBM



batch_size = 1, num_heads = 2, 2 blocks for parallelism.

FlashAttentionv2:How to achieve parallelism over seq length?

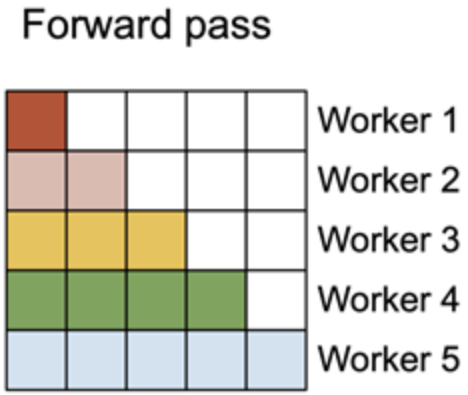




batch_size = 1, num_heads = 2
divide the seq length of Q.
8 blocks for parallelism.

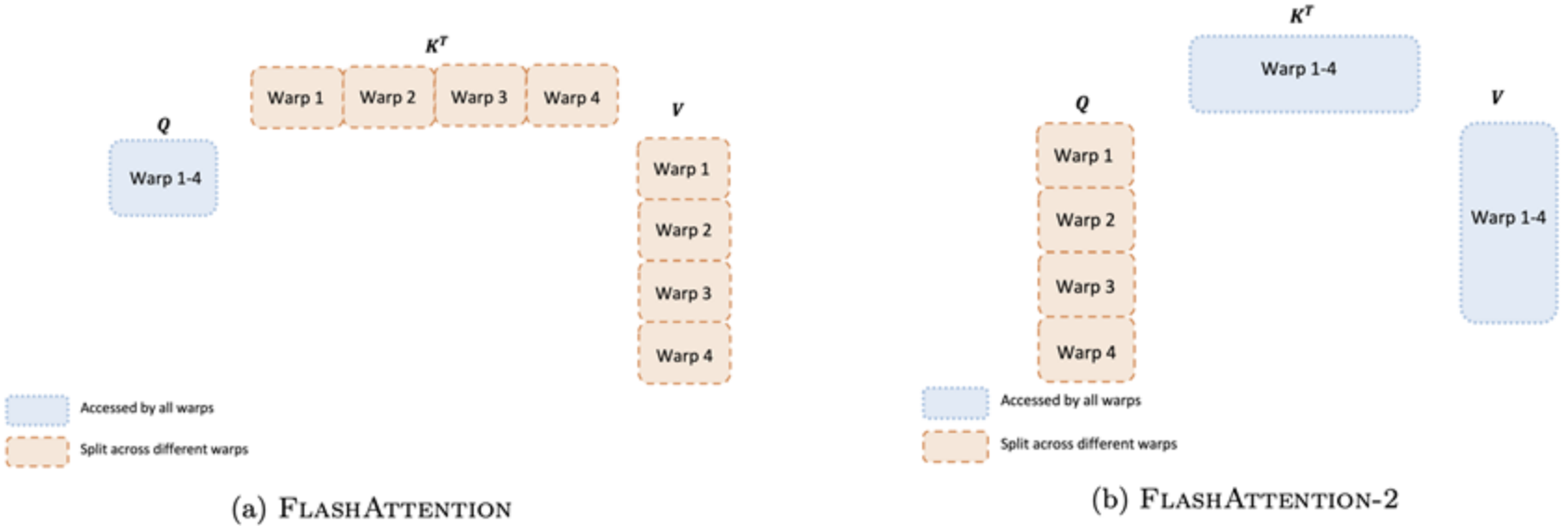
FlashAttention v2

- Swap order of loop
- Parallelize outer loop
- Leads to improved occupancy



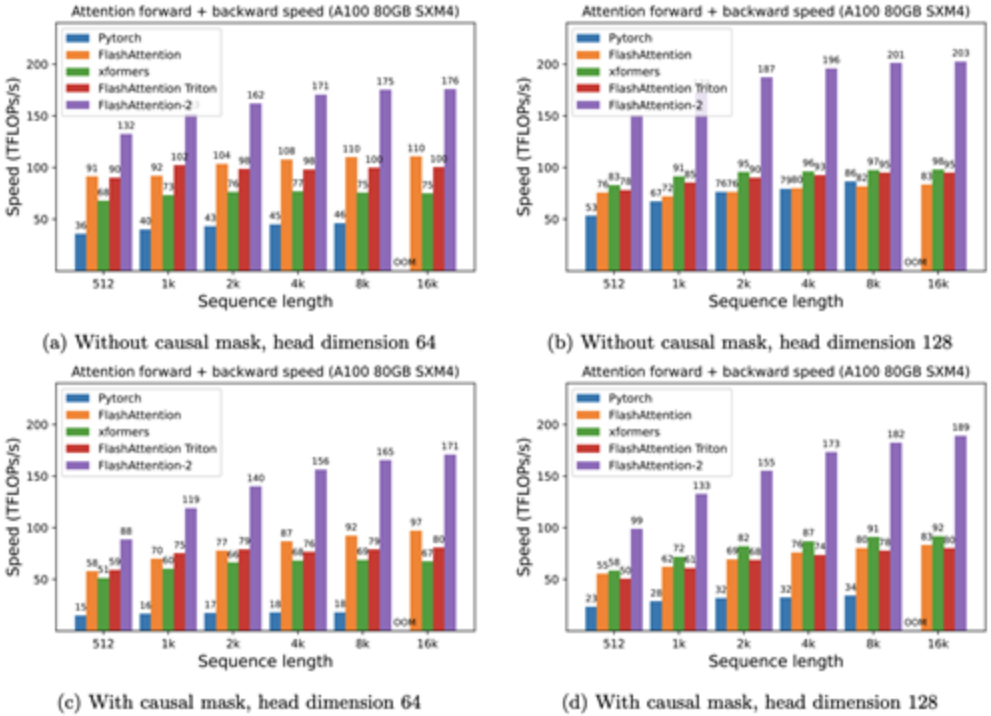
Parallelize the workers (thread blocks) where each worker takes care of a block of rows of the attention matrix.

FlashAttentionv2: Work Partitioning Between Warps



Work partitioning between different warps in the forward pass

Benchmarking attention



Attention forward speed on A100 GPU

2X faster than FlashAttention
1.3X faster than Triton
10X faster than Pytorch

Reaches up to 230 TFLOPs/s, 73% of the threoretical max on A100

End to End Performance

2.8X faster than baseline
1.3X faster than FlashAttention

Model	Without FlashAttention	FlashAttention	FlashAttention-2
GPT3-1.3B 2k context	142 TFLOPs/s	189 TFLOPs/s	196 TFLOPs/s
GPT3-1.3B 8k context	72 TFLOPS/s	170 TFLOPs/s	220 TFLOPs/s
GPT3-2.7B 2k context	149 TFLOPs/s	189 TFLOPs/s	205 TFLOPs/s
GPT3-2.7B 8k context	80 TFLOPs/s	175 TFLOPs/s	225 TFLOPs/s

Table 1: Training speed (TFLOPs/s/GPU) of GPT-style models on 8×A100 GPUs. FlashAttention-2 reaches up to 225 TFLOPs/s (72% model FLOPs utilization). We compare against a baseline running without FlashAttention.

Summary

- FlashAttention2 is 2x faster than FlashAttention
- FlashAttention2 will also speed up training, finetuning, and inference
- Future Directions
  - Device dependent
  - Hand-writing CUDA implementation, specially designed for specific attention implementation
  - How the FlashAttention2 performs on sparse attention mechanisms
  - Auto-tuning mechanisms for selecting optimal block sizes and partitioning strategies could simplify the use of FlashAttention-2