

Deep Learning Systems: Algorithms and Implementation

Transformers implementation

Tim Dettmers (this time) and Tianqi Chen, Carnegie Mellon University

Transformer block

In more detail, the Transformer block has the following form:

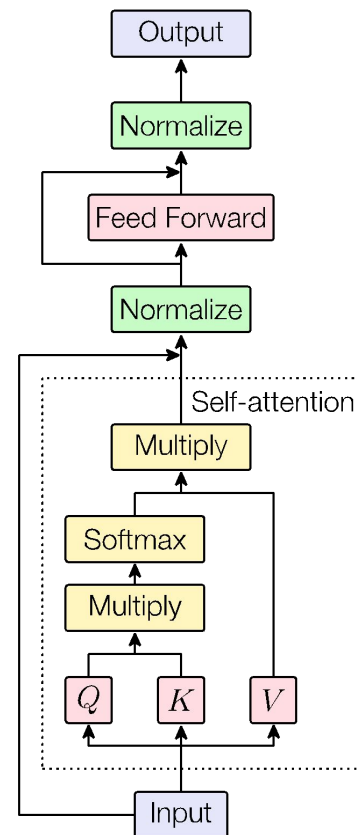
$$\tilde{Z} := \text{SelfAttention}(Z^{(i)}W_K, Z^{(i)}W_Q, Z^{(i)}W_V)$$

$$= \text{softmax} \left(\frac{Z^{(i)}W_KW_V^T Z^{(i)T}}{d^{1/2}} \right) Z^{(i)}W_V$$

$$\tilde{Z} := \text{LayerNorm}(Z^{(i)} + \tilde{Z})$$

$$Z^{(i+1)} := \text{LayerNorm}(\text{ReLU}(\tilde{Z}W_1)W_2 + \tilde{Z})$$

A bit complex, but really just self-attention, followed by a linear layer + ReLU, with residual connections and normalization thrown in somewhat arbitrarily (and precisely where these go are often adjusted)



Self-attention with einsum and einops

```
def self_attention(x, W_qkv, W_o, n_heads):
    # b=batch, s=sequence, h=hidden
    b, s, h = x.shape
    # d=head_dim
    d = h // n_heads
    # o=output

    qkv = einsum('b s h, h o -> b s o', x, W_qkv)
    # kvq axis is labeled in [q|k|v] layout, so unpacking yields q, k, v in that order
    q, k, v = rearrange(qkv, 'b s (kvq heads d) -> kvq b heads s d', kvq=3, heads=n_heads)
    scores = einsum('b heads s d, b heads z d -> b heads s z', q, k) / d**0.5
    attn = scores.softmax(dim=-1)
    out = einsum('b heads s z, b heads z d -> b heads s d', attn, v)
    out = rearrange(out, 'b heads s d -> b s (heads d)')
    out = einsum('b s h, h o -> b s o', out, W_o)
    return out
```

NVIDIA Blackwell Ultra GPU

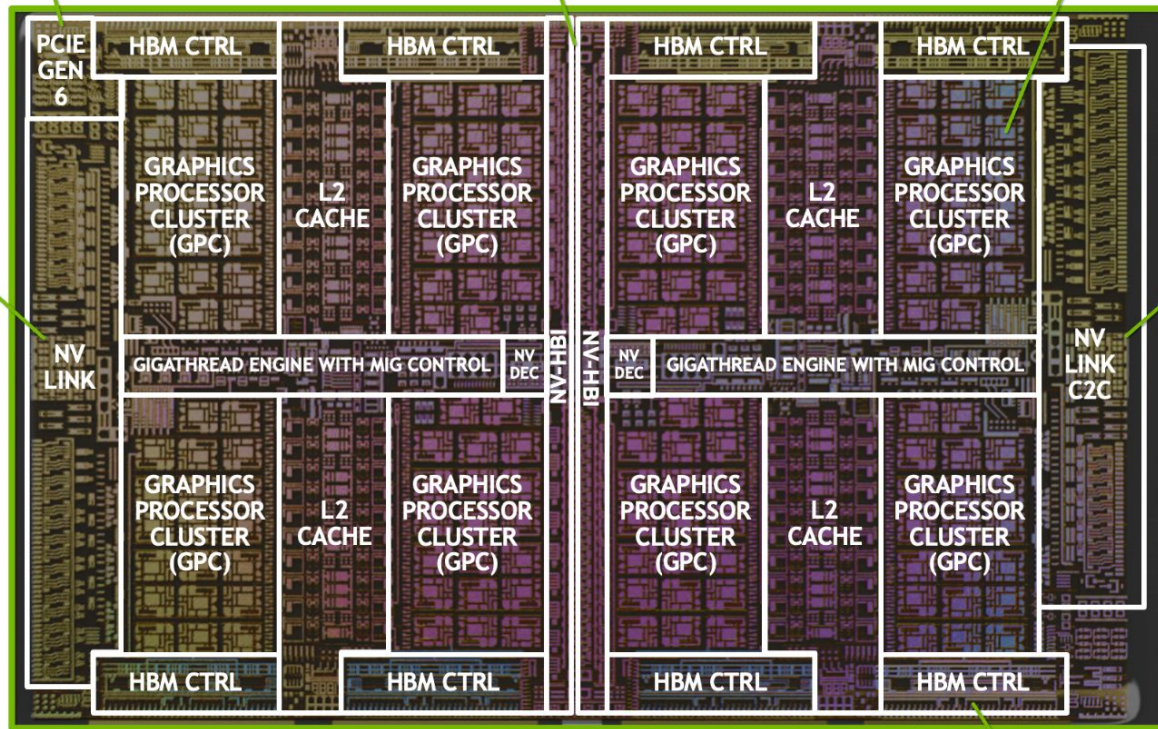
x16 PCIe Gen 6
256 GB/s CPU Host Interface

High Bandwidth Interface
10TB/s Die-to-Die

160 SMs per GPU: 640 Tensor Cores*
15 PetaFLOPS Dense NVFP4

NVLink v5
1,800 GB/s to
NVLink Switch

NVLink-C2C
900GB/s Coherent
CPU-GPU Interface



Confidential Computing
TEE-I/O Capable

288GB HBM3E Memory*
(12-Hi Stacks, Up to 8 TB/s)

Streaming Multiprocessor (SM)



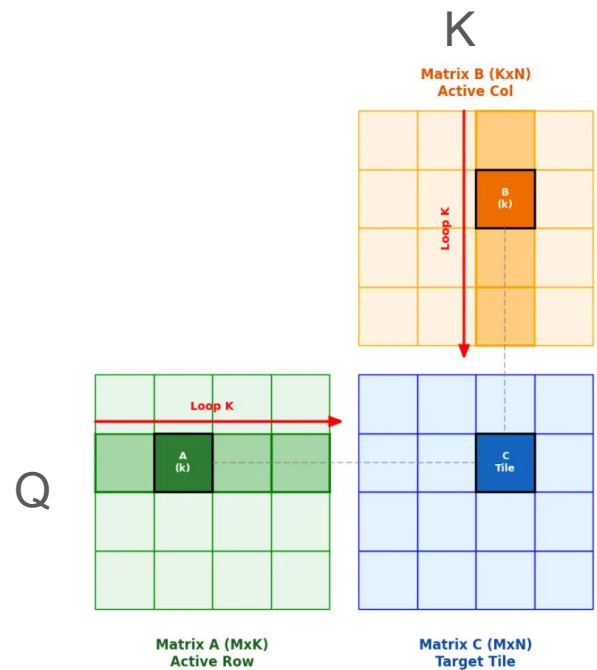
HBM / global memory
~700 cycles

L2 cache / shared memory
~200 to ~400 cycles

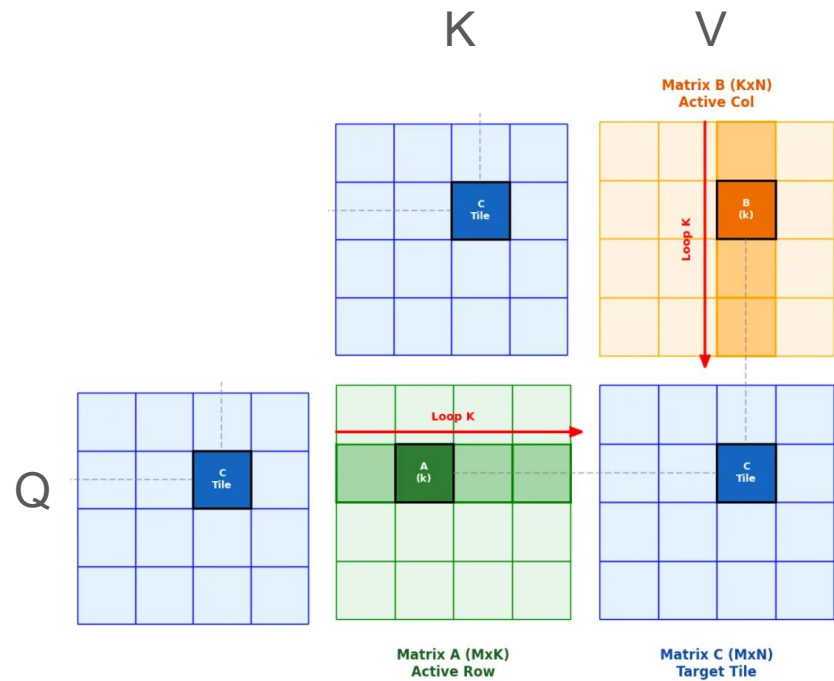
L1 cache / shared memory
~25 cycles

Registers / tensor cores
1 cycle

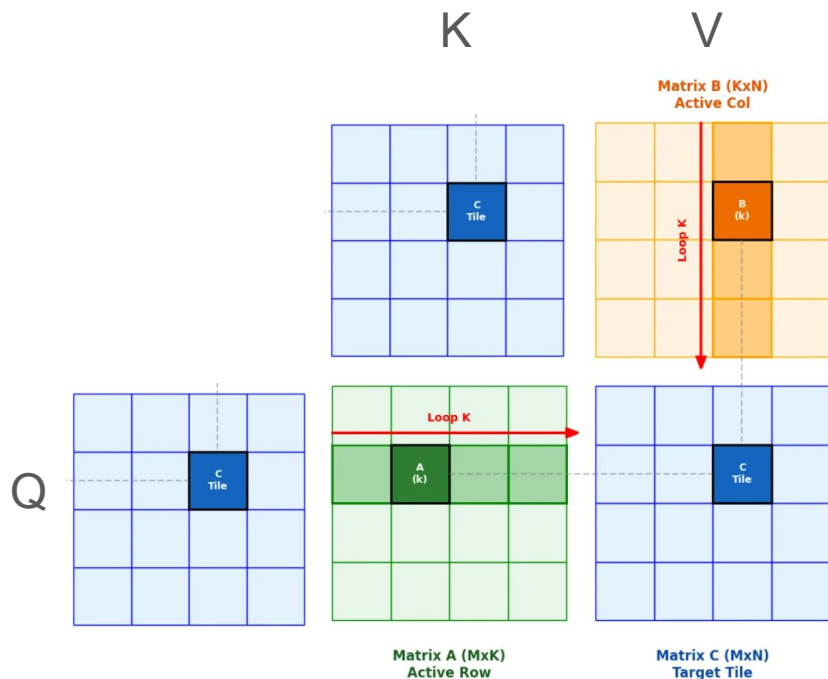
FlashAttention



FlashAttention



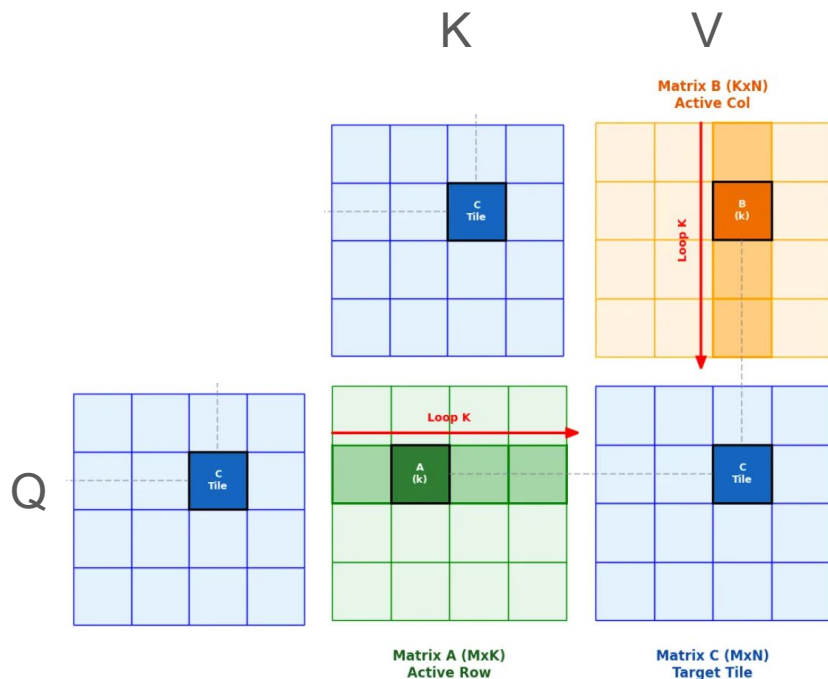
FlashAttention



Sum Z

$$\text{softmax}(x)_i = \frac{e^{x_i}}{Z}, \quad Z = \sum_{j=1}^N e^{x_j}.$$

FlashAttention



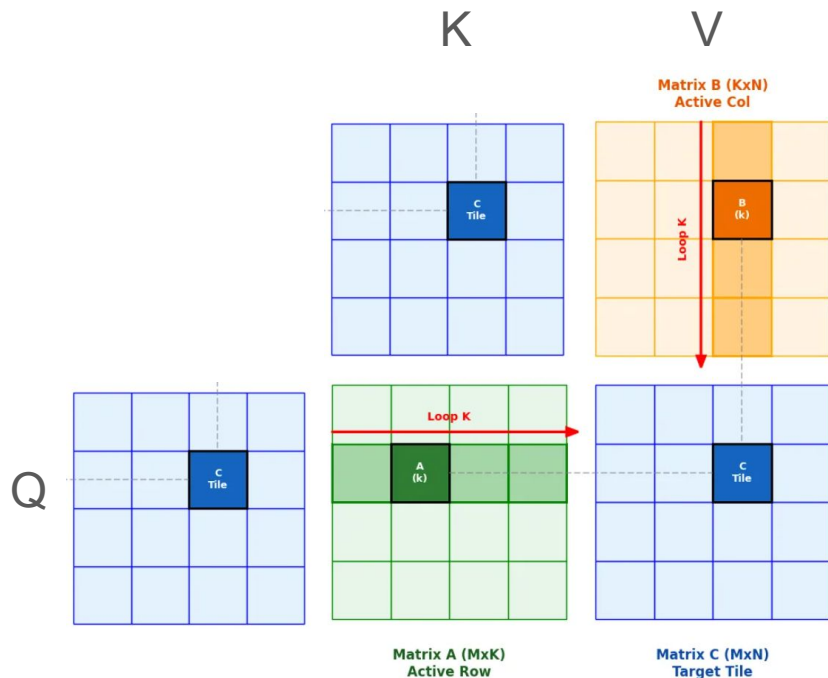
Sum Z

$$\text{softmax}(x)_i = \frac{e^{x_i}}{Z}, \quad Z = \sum_{j=1}^N e^{x_j}.$$

Simple softmax

$$Z = \sum_j e^{x_j}$$

FlashAttention



Sum Z

$$\text{softmax}(x)_i = \frac{e^{x_i}}{Z}, \quad Z = \sum_{j=1}^N e^{x_j}.$$

Simple softmax

$$Z = \sum_j e^{x_j}$$

Stable softmax

$$m = \max_j x_j, \quad Z = \sum_j e^{x_j - m},$$

Q



Online softmax

$$m = \max_j x_j, \quad Z = \sum_j e^{x_j - m},$$

$$\underbrace{Z_k^{[m_k]}}_{\text{new } Z, \text{ new unit}} = \underbrace{Z_{k-1}^{[m_{k-1}]} \cdot e^{m_{k-1}-m_k}}_{\text{old } Z \text{ converted to the new unit}} + \underbrace{e^{x_k-m_k}}_{\text{new term, already in the new unit}}.$$