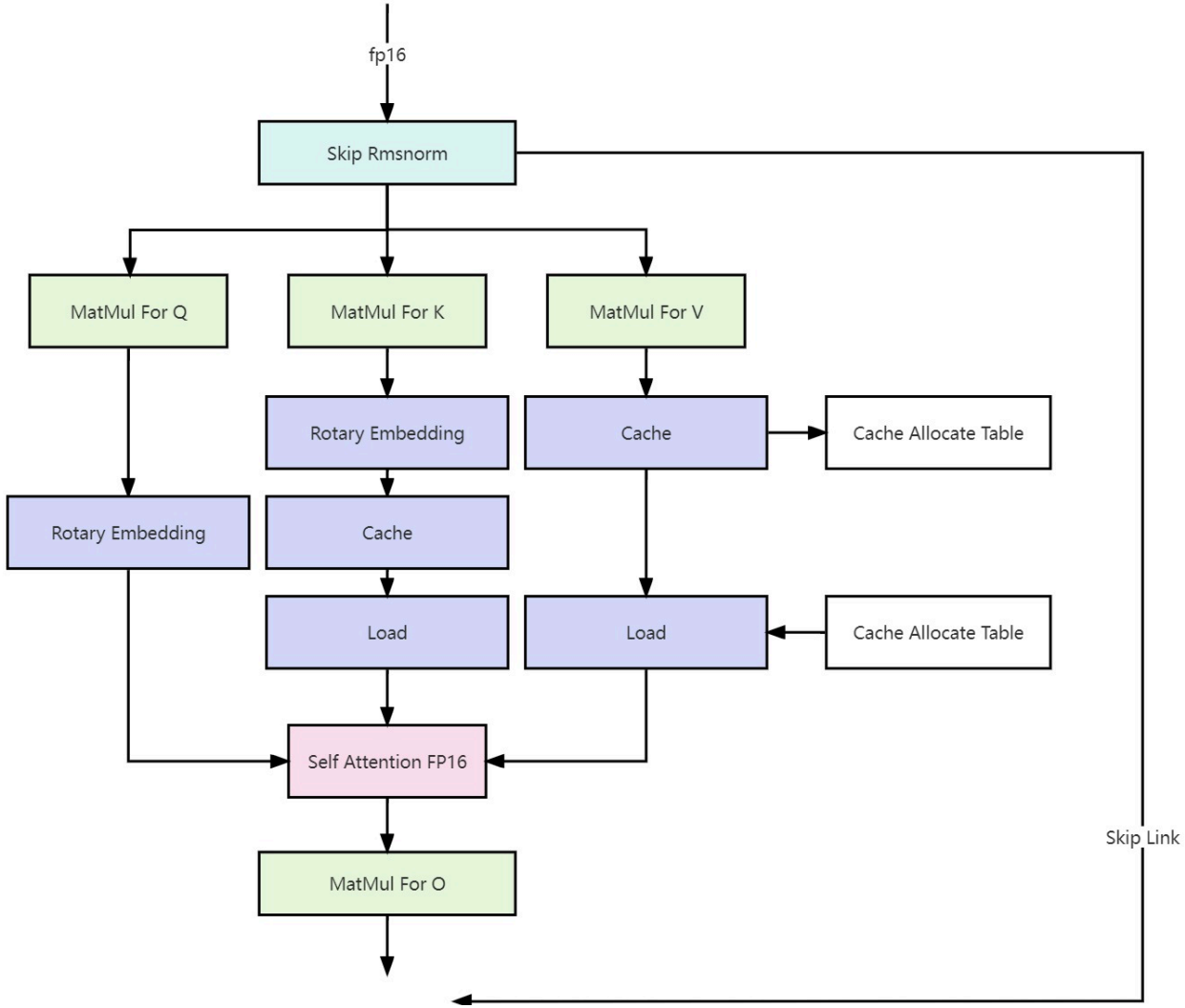
Language Model Inference

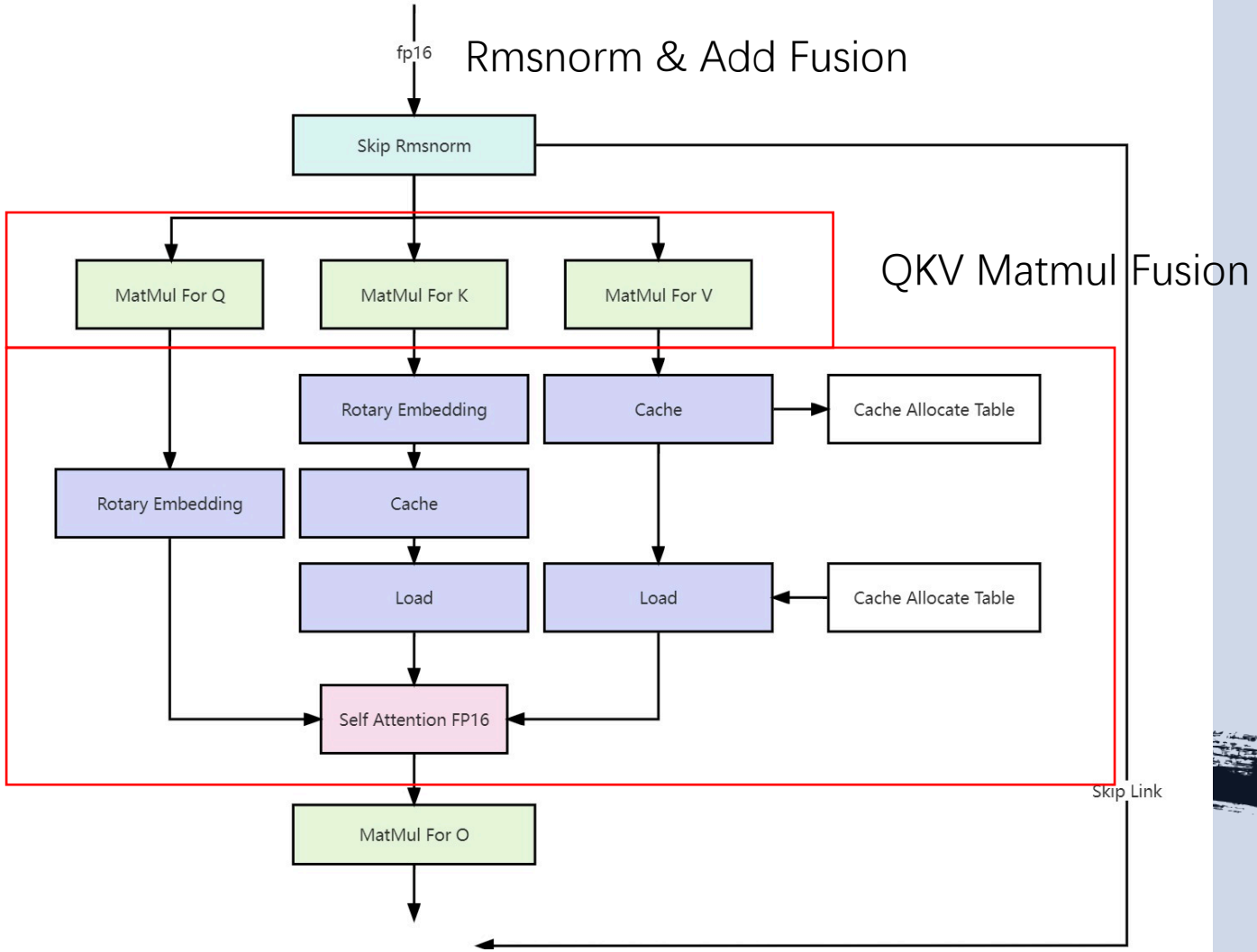
张志 商汤科技 高性能计算研发工程师

Transformer

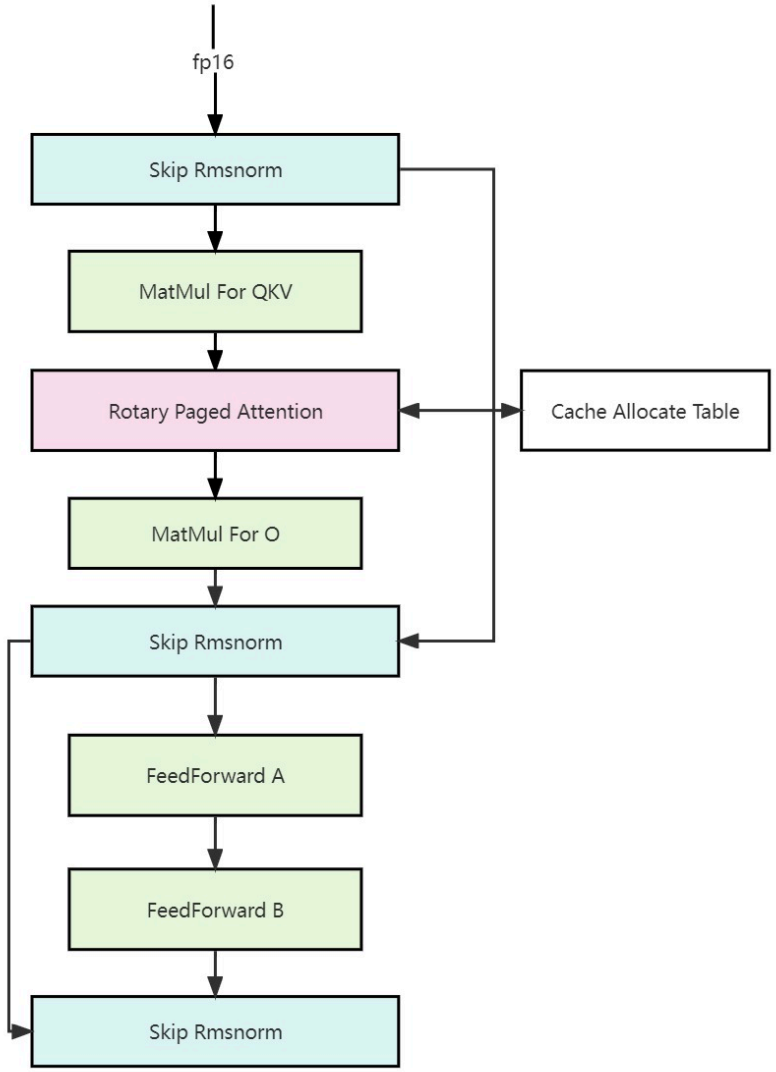


图融合

Paged Attention
Rotary Embedding
Cache Load & Write Fusion



图融合


fp16
Skip Rmsnorm
MatMul For QKV
Rotary Paged Attention
Cache Allocate Table
MatMul For O
Skip Rmsnorm
FeedForward A
FeedForward B
Skip Rmsnorm

万事第一步：测算算子开销 (Batchsize=1)

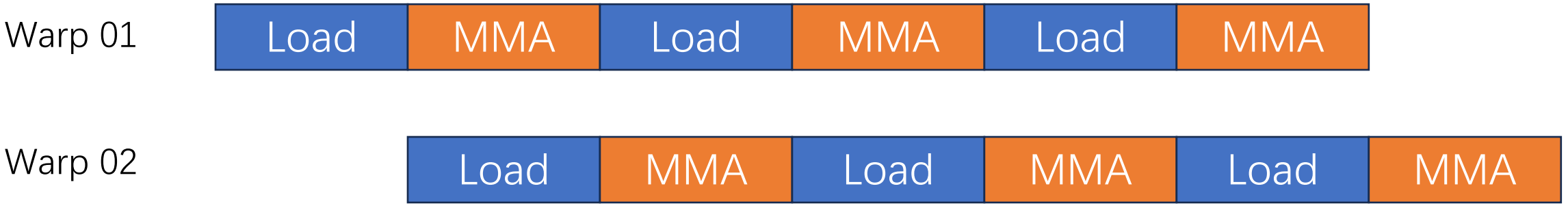
Opname	Type	Gpu time
Matmul For Q	Gemm	30
Matmul For K	Gemm	31
Matmul For V	Gemm	30
MatMul For O	Gemm	31
SkipRMSNorm A	Normalization	5
FeedForward Gemm A	Gemm	70
FeedForward Gemm B	Gemm	71
FeedForward Gemm C	Gemm	73
SkipRMSNorm B	Normalization	5
Attention	Attention	9

算子开销 (Batchsize=1)

当batchsize=1时，运算皆为矩阵向量乘，这是访存密集型算子

Opname	Type	Gpu time
Matmul For Q $[1,4096]*[4096,4096]$	Gemm	30
Matmul For K	Gemm	31
Matmul For V	Gemm	30
MatMul For O	Gemm	31
SkipRMSNorm A	Normalization	5
FeedForward Gemm A	Gemm	70
FeedForward Gemm B	Gemm	71
FeedForward Gemm C	Gemm	73
SkipRMSNorm B	Normalization	5
Attention	Attention	9

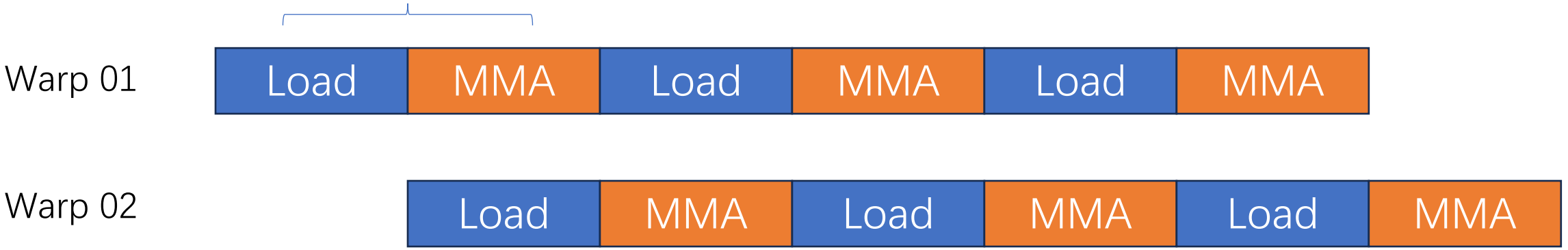
运算访存比



GPU 可以通过 Warp(线程束) 的上下文切换来掩盖访存延迟

运算访存比

我们希望计算和访存所需的时间必须尽可能相近



GPU 可以通过 Warp(线程束) 的上下文切换来掩盖访存延迟

运算访存比

测试平台

- 系统：Arch Linux
- 驱动：520.56.06
- CUDA：11.8
- GPU：Nvidia RTX 2080

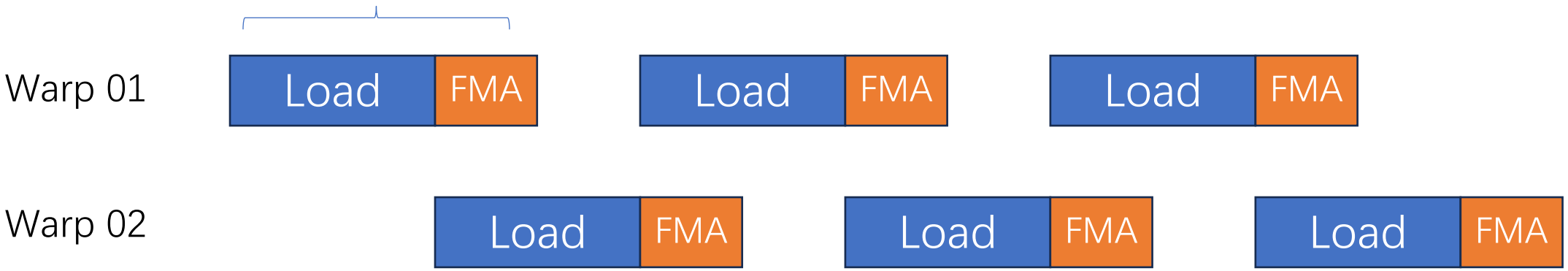
项目	RTX 2080
DRAM latency	423 cycles
DRAM bandwidth	383.4GB/s (read) 334.5 GB/s (write)
L1 cache latency	32 cycles
L2 cache latency	217 cycles
L2 cache bandwidth	1730.75 GB/s
shared memory latency	22 cycles
shared memory bandwidth	111.73 bytes/cycle 10686.72 GB/s
FP32 Tflops	10.1 Tflops

从内存中读写一个数的时间

GPU可以做近百次运算

运算访存比

访存延迟无法被掩盖，显卡利用率低



矩阵向量乘不可以使用 Tensor Core，并且访存延迟远高于计算延迟

运算访存比是优化大模型推理的核心

提升计算量的方法：

增大 Batchsize

- 量化 KV Cache
- Weight Only 量化
- Paged Attention
- …

降低访存量的方法：

Weight Only 量化

Int8 常规量化

减小计算量的方法：

Int8 常规量化

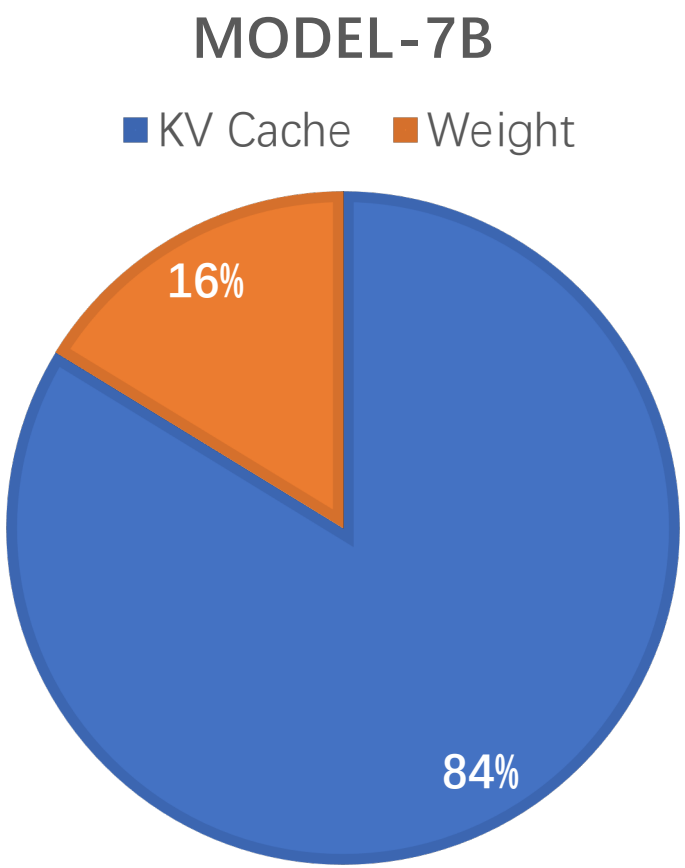
KV Cache(Decoding)

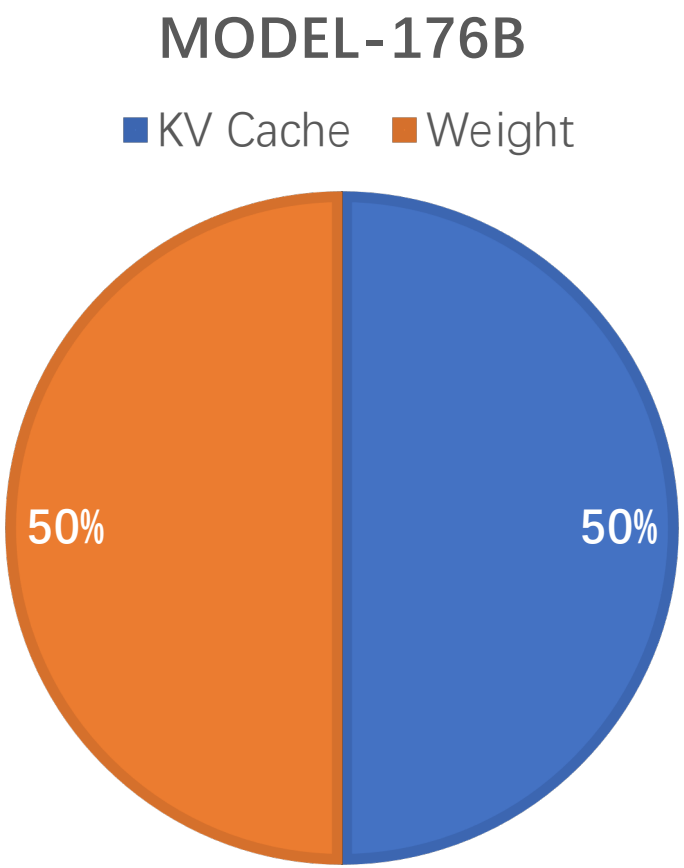

```
attn = query @ key.transpose(-2, -1)
attn = attn.softmax(-1)
return attn @ value
```

Memory Plan

Llama-7b KV-Cache FP16 (GB)							Llama-7b Weight FP16 (GB)	
Seqlen \ Batchsize	1	2	4	8	16	32	Q	1.00
128	0.06	0.13	0.25	0.50	1.00	2.00	K	1.00
256	0.13	0.25	0.50	1.00	2.00	4.00	V	1.00
512	0.25	0.50	1.00	2.00	4.00	8.00	O	1.00
1024	0.50	1.00	2.00	4.00	8.00	16.00	Feed Forward	8.00
2048	1.00	2.00	4.00	8.00	16.00	32.00	Sum	12.00
4096	2.00	4.00	8.00	16.00	32.00	64.00		

KV Cache is the Key

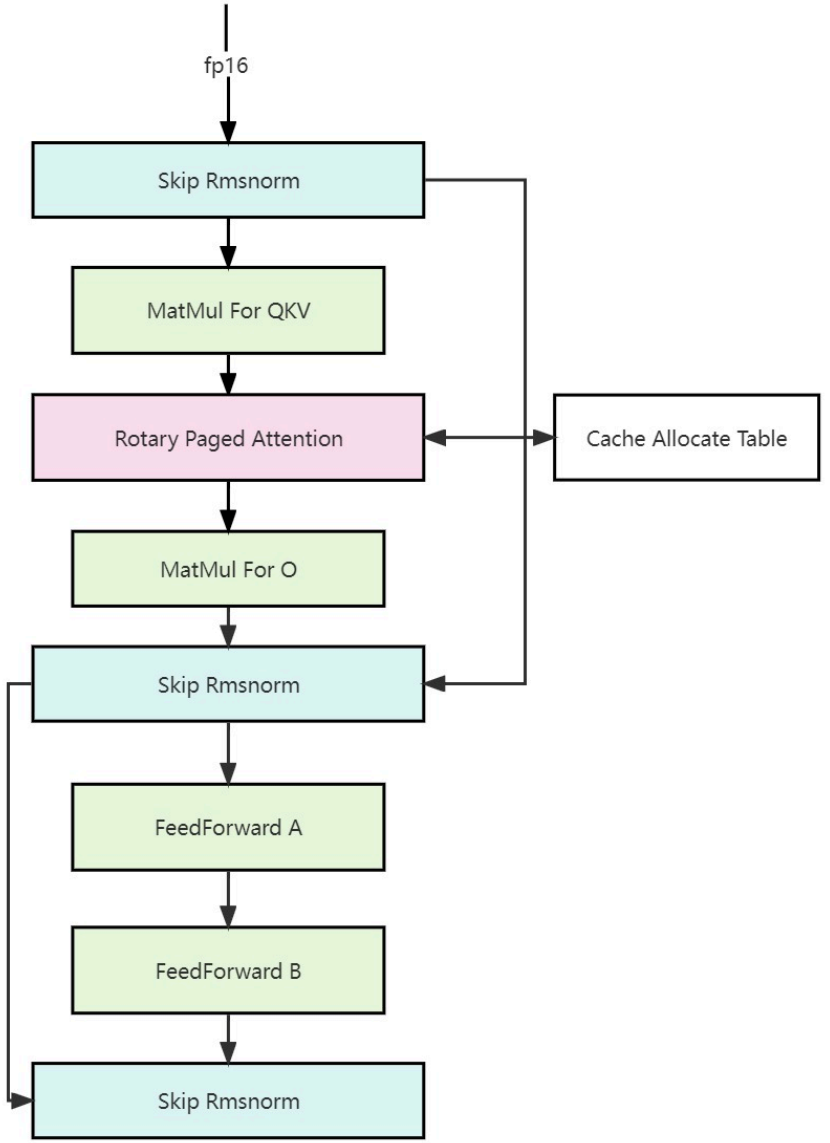




模型显存使用情况：A100 80G

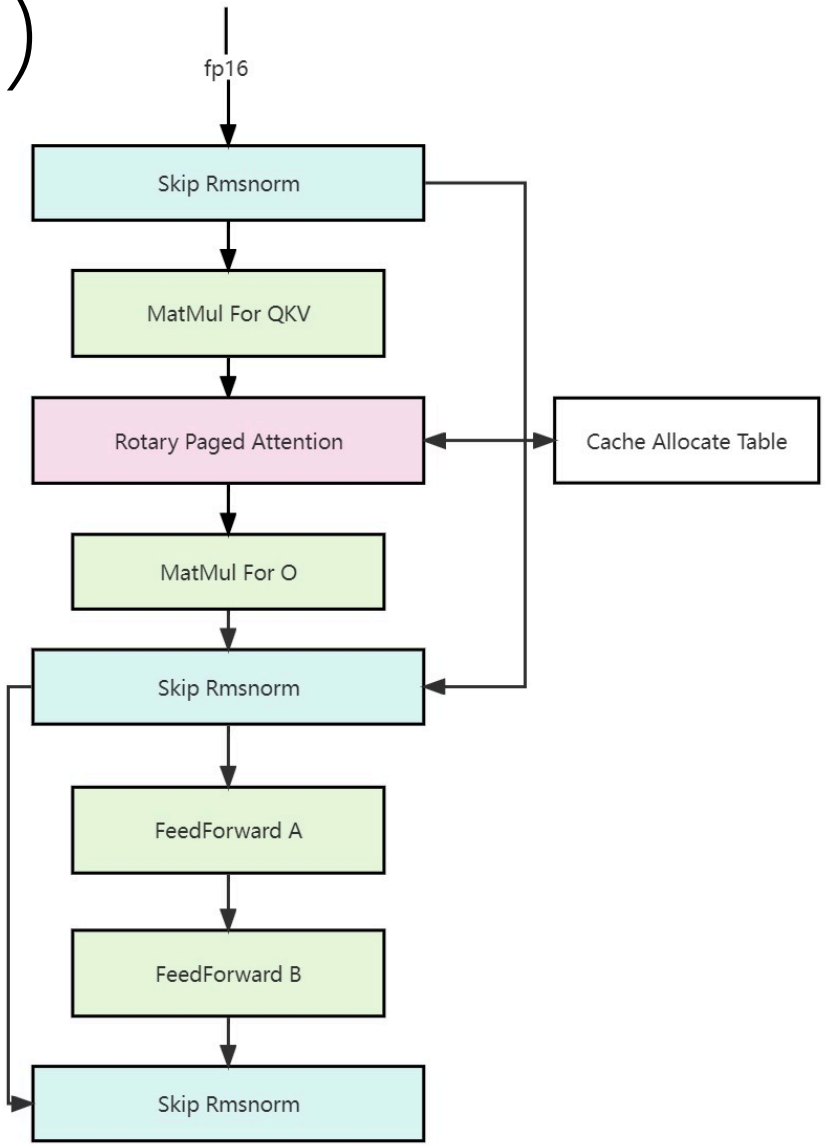
Operator Cost(Batchsize=1)

Opname	Type	Gpu time
Matmul For Q	Gemm	30
Matmul For K	Gemm	31
Matmul For V	Gemm	30
MatMul For O	Gemm	31
SkipRMSNorm A	Normalization	5
FeedForward Gemm A	Gemm	70
FeedForward Gemm B	Gemm	71
FeedForward Gemm C	Gemm	73
SkipRMSNorm B	Normalization	5
Attention	Attention	9



Operator Cost(Batchsize=512)

Opname	Type	Gpu time
Matmul For Q	Gemm	84
Matmul For K	Gemm	82
Matmul For V	Gemm	82
MatMul For O	Gemm	82
SkipRMSNorm A	Normalization	12
FeedForward Gemm A	Gemm	213
FeedForward Gemm B	Gemm	213
FeedForward Gemm C	Gemm	216
SkipRMSNorm B	Normalization	12
Attention	Attention	26



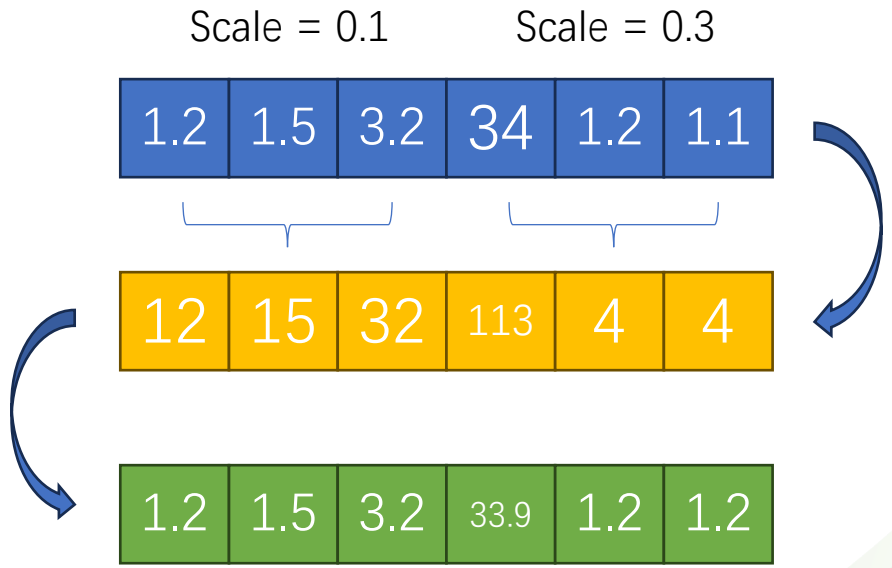
Groupwise KV Cache Quantization

- 一种针对用户上下文的量化方案

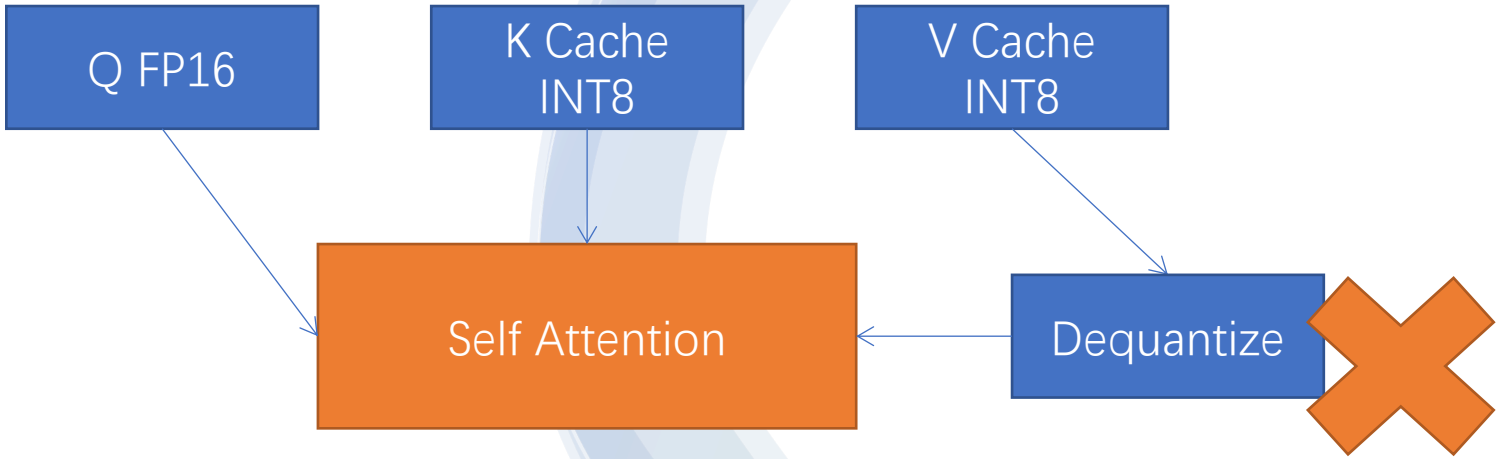
```
Def Quantized_Attention(
    FP16_Q, INT8_K, INT8_V, scale_K, scale_V):
    // Load into local
    Local_Q = Tile(FP16_Q)
    Local_K = Dequantize(Tile(INT8_K), scale_K)
    Local_V = Dequantize(Tile(INT8_V), scale_V)

    // Warp Dot
    Logits_fp32 = Warp_dot(Local_Q, Local_K)

    // Softmax…
```

Groupwise Quantization

- 一种针对访存密集型算子的量化方案
  - 被量化的 Tensor 按 k 个元素为一组，每 k 个元素使用一个 scale 作为量化参数。
  - scale 等于这 k 个元素中最大值的 1 / 127
  - 这种量化方案只压缩显存，不能加速运算
  - 这种量化方案的精度与分组大小直接相关，k 取得越小精度越高。



Groupwise Quantization



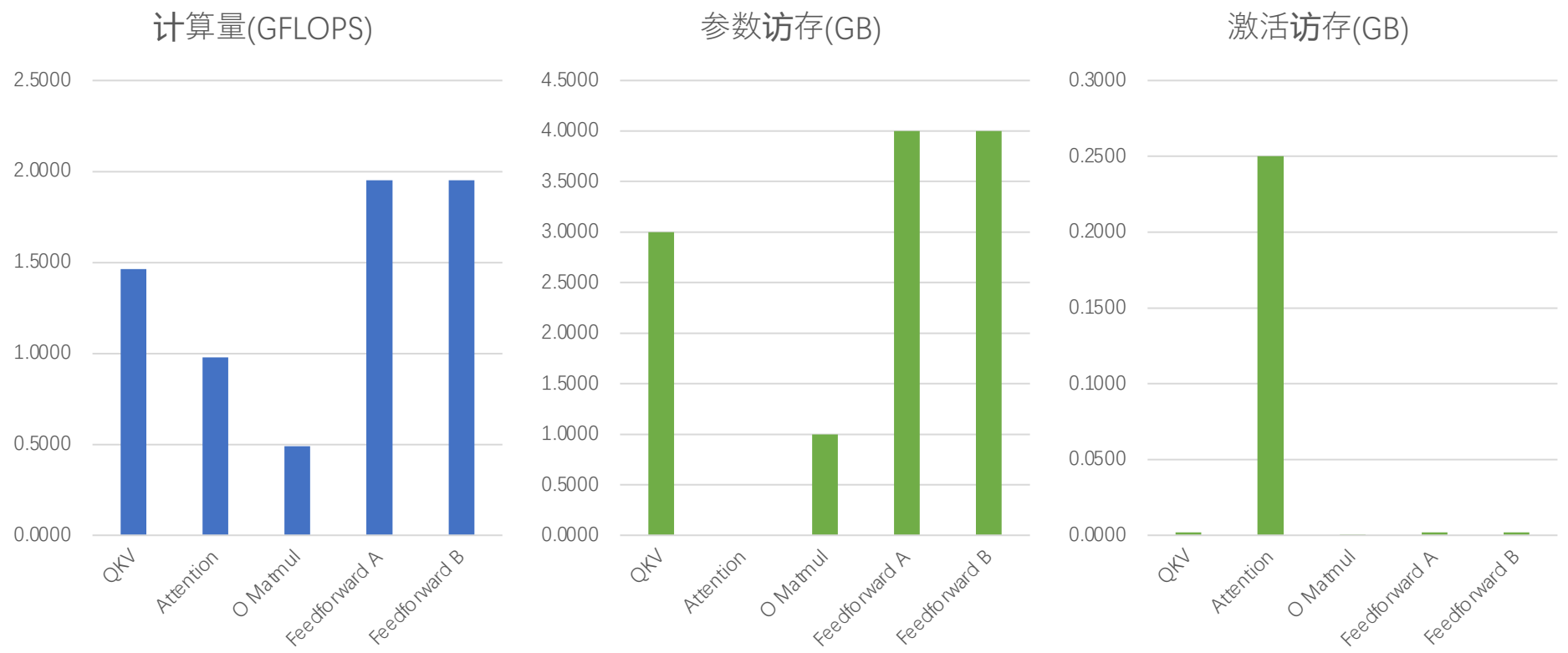
** 解压缩过程必须写入 Self Attention Kernel，必须在寄存器或Shared memeory上完成

Matmul in LLM(Decoding)

权重的显存 IO 大约是激活的 数百倍

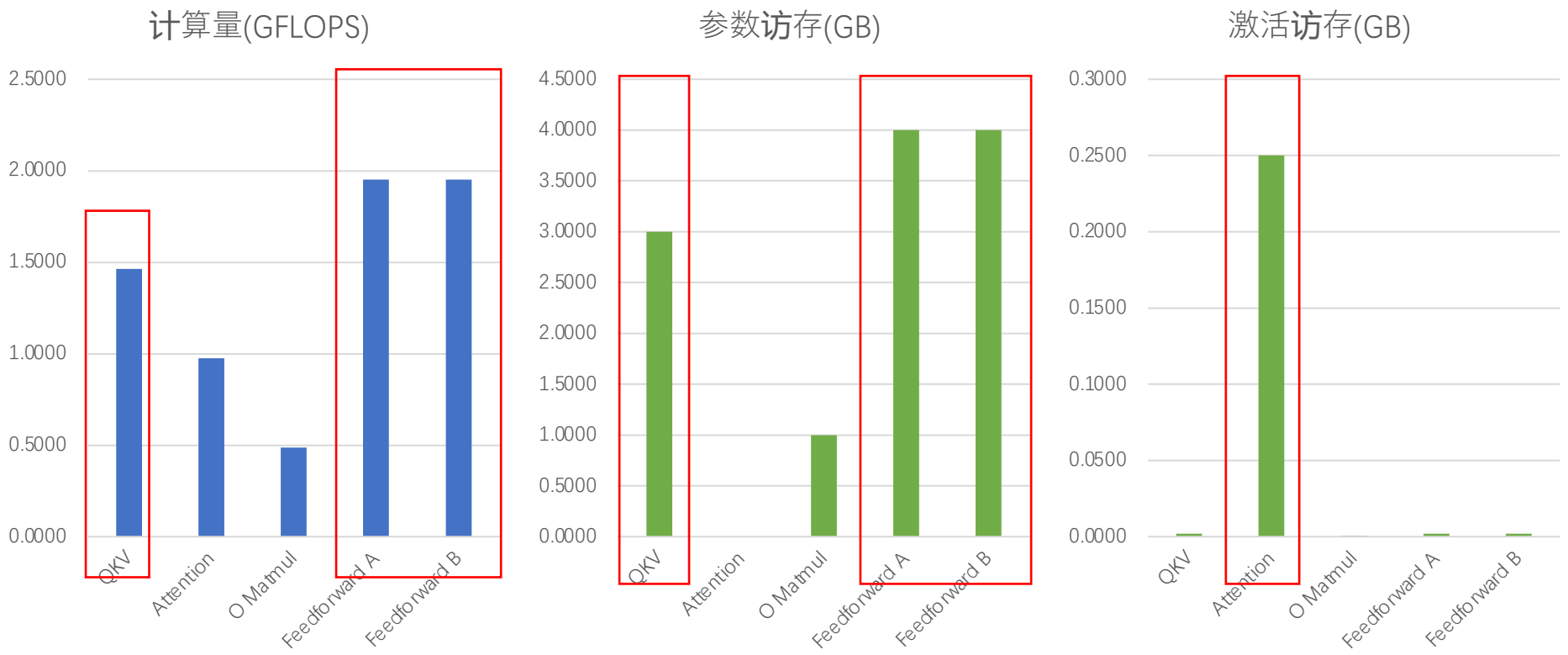
计算量与 batch size 成正比

Inference = IO + Computation



联表：LLM-7B 模型的静态分析, Hidden dim=4096, Layer=32, Seqlen=512

Inference = IO + Computation



联表：LLM-7B 模型的静态分析, Hidden dim=4096, Layer=32, Seqlen=512

第二种量化：INT8 MMA 量化

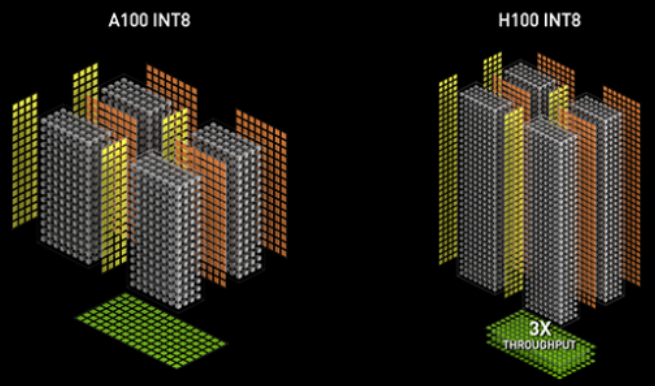
NVIDIA H100 Tensor Core

第四代

自推出 Tensor Core 技术以来，NVIDIA GPU 的峰值性能提高了 60 倍，推动了 AI 和 HPC 计算的普及化。NVIDIA Hopper™ 架构利用 Transformer 引擎改进第四代 Tensor Core，该引擎使用新的 8 位浮点精度 (FP8)，可为万亿参数模型训练提供比 FP16 高 6 倍的性能。Hopper Tensor Core 使用 TF32、FP64、FP16 和 INT8 精度，将性能提升 3 倍，能够加速处理各种工作负载。

详细了解 NVIDIA Hopper 架构 ›

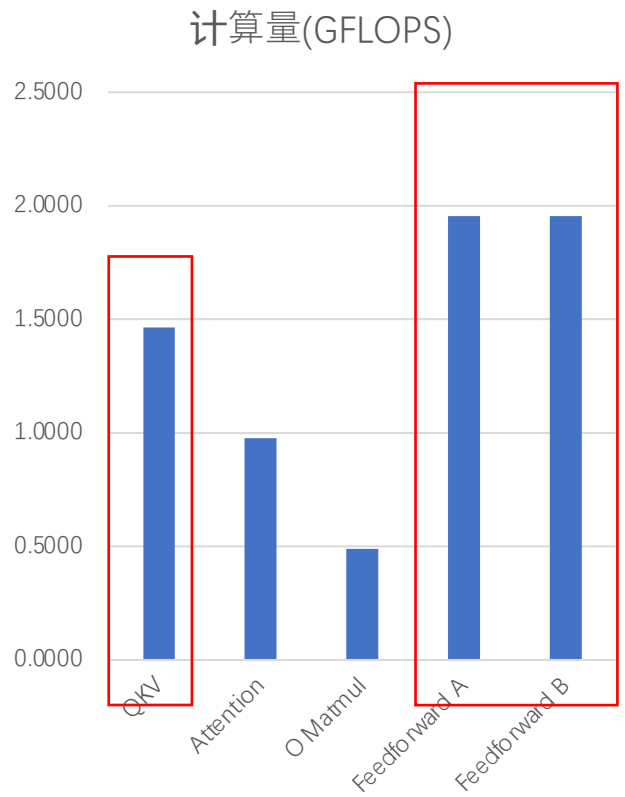
FP8 TF32 FP64 FP16 INT8



INT8

INT8 Tensor Core 在 NVIDIA Turing™ 中首次引入，可显著加速推理吞吐量，并大幅提升效率。对于生产部署，NVIDIA Hopper 架构中的 INT8 提供了比上一代 Tensor Core 高 3 倍的类似吞吐量。这种通用性为核心和边缘数据中心的大批量实时工作负载提供行业领先的性能。

第二种量化：INT8 MMA 量化



Def INT8_SGEMM(INT8_A, INT8_B):　　访存开销缩减50%

// Load into local

Local_A = Tile(INT8_A)

Local_B = Tile(INT8_B)

// Tile MMA

Local_C_int32 = TENSORCORE::MMA_INT8(A, B)　　计算开销缩减50%

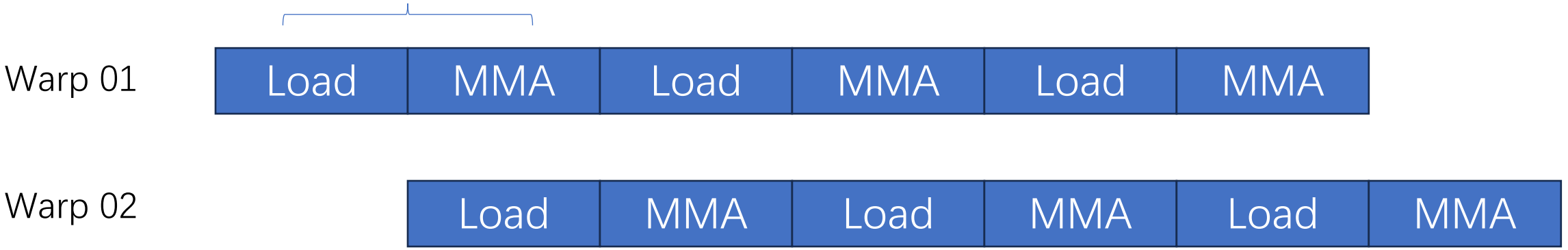
MMA 是硬件指令，要求 A, B 必须为 int8

形状必须为 [8, 16] 的整数倍

// Reduce Sum and Write …

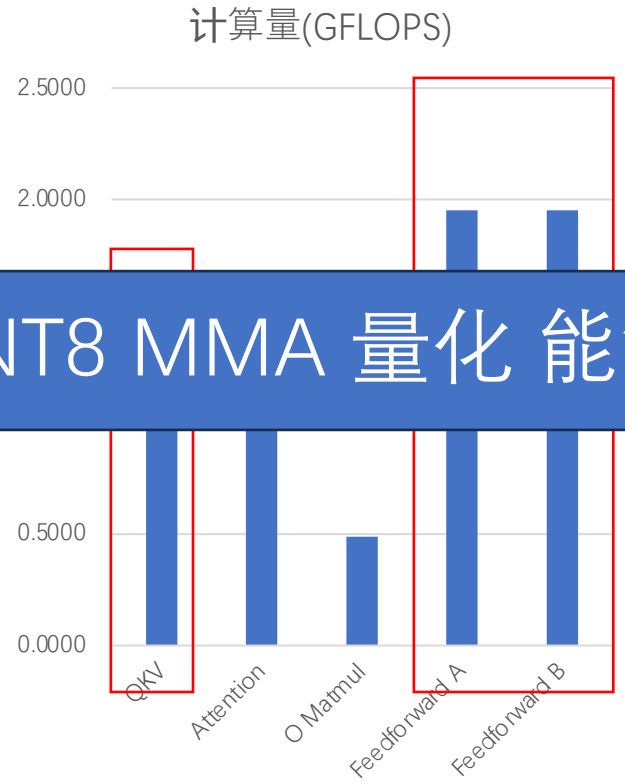
第二种量化：INT8 MMA 量化

我们希望计算和访存所需的时间必须尽可能相近



GPU将通过 Warp 的切换来掩盖访存延迟

第二种量化：INT8 MMA 量化



Def INT8_SGEMM(INT8_A, INT8_B):

// Load into local

Local_A = Tile(INT8_A)

Local_B = Tile(INT8_B)

INT8 MMA 量化 能够使得运算与访存开销全部减少 50%

MMA 是硬件指令，要求 A, B 必须为 int8

形状必须为 [8, 16] 的整数倍

// Reduce Sum and Write …

第三种量化：Weight only 量化矩阵乘

Modelsize-7B			
	计算量	参数访存	激活访存
QKV	0.001464844 * BSZ	3	0.001953 * BSZ
Attention	0.000976563 * BSZ * (SEQLEN / 512)		0.25 * BSZ * (SEQLEN / 512)
O Matmul	0.000488281 * BSZ	1	0.000488 * BSZ
Normalization			
Feedforward A	0.001953125 * BSZ	4	0.001953 * BSZ
Feedforward B	0.001953125 * BSZ	4	0.001953 * BSZ
Sum	0.006835938 * BSZ	12	0.256348 * BSZ

大语言模型参数量极大，参数访存量极高

第三种量化：Weight only 量化矩阵乘

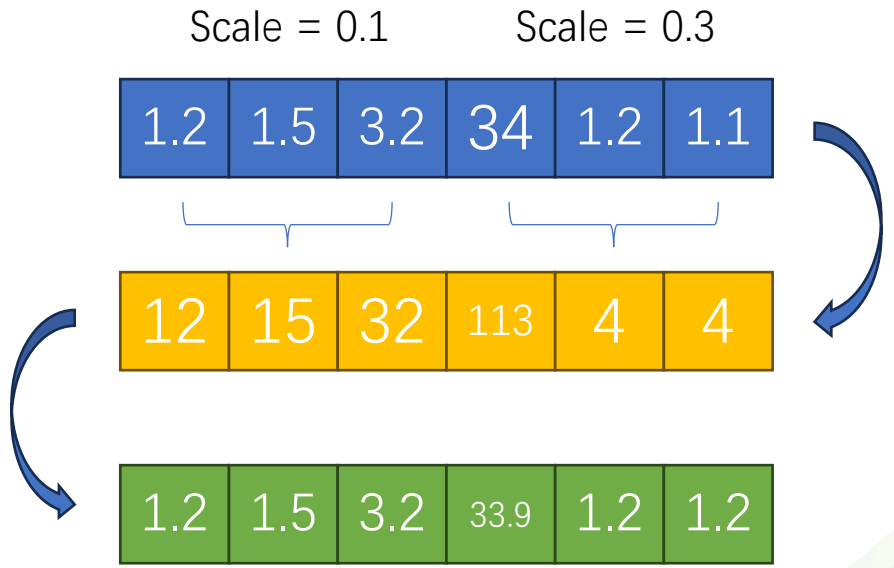
Modelsize-7B			
	计算量	参数访存	激活访存
QKV	0.001464844 * BSZ	3	0.001953 * BSZ
Attention	0.000976563 * BSZ * (SEQLEN / 512)		0.25 * BSZ * (SEQLEN / 512)
O Matmul	0.000488281 * BSZ	1	0.000488 * BSZ
Normalization			
Feedforward A	0.001953125 * BSZ	4	0.001953 * BSZ
Feedforward B	0.001953125 * BSZ	4	0.001953 * BSZ
Sum	0.006835938 * BSZ	12	0.256348 * BSZ

** 参数访存量与 Batchsize 无关

Groupwise Weight Only Quantization

- 一种针对大语言模型矩阵乘的量化方案

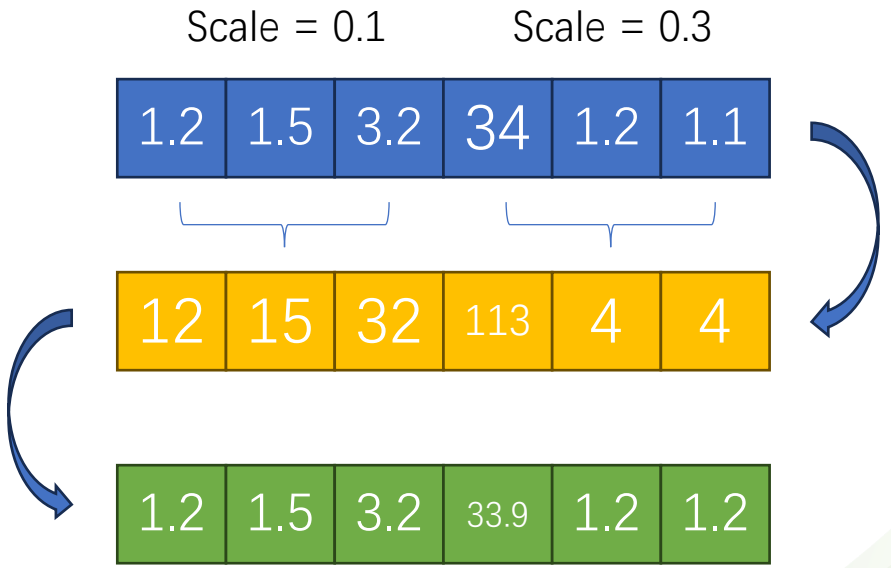
```
Def WEIGHT_ONLY_SGEMM(FP16_A, INT8_B):
    // Load into local
    Local_A = Tile(FP16_A)
    Local_B = Dequantize(Tile(INT8_B))

    // Tile MMA
    Local_C_fp32 = CUDA_CORE::FMA_FP16(A, B)
```



Groupwise Weight Only Quantization

- 一种针对大语言模型矩阵乘的量化方案

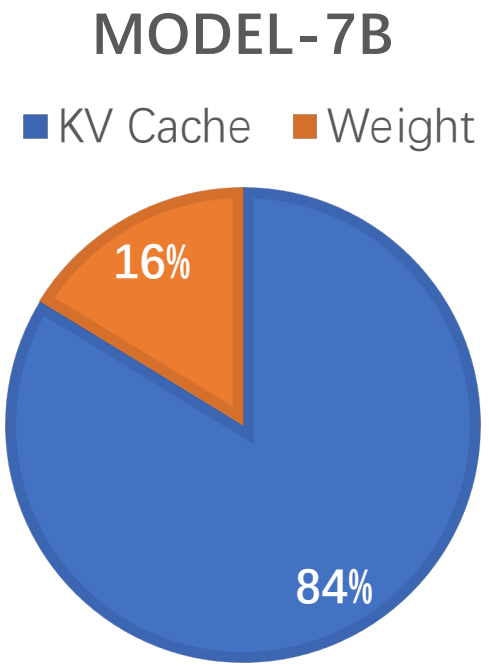
- 大语言模型在 batchsize < 8 时，矩阵乘运算完全受限于访存带宽
- Weight Only 量化实际上是以计算换空间
- 访存需求减少75%，但计算量增加。
- 当batchsize > 8时，该方案基本没有加速效果。
- 当batchsize = 1时，该方案提供3~4倍加速。



关于 Weight only 量化

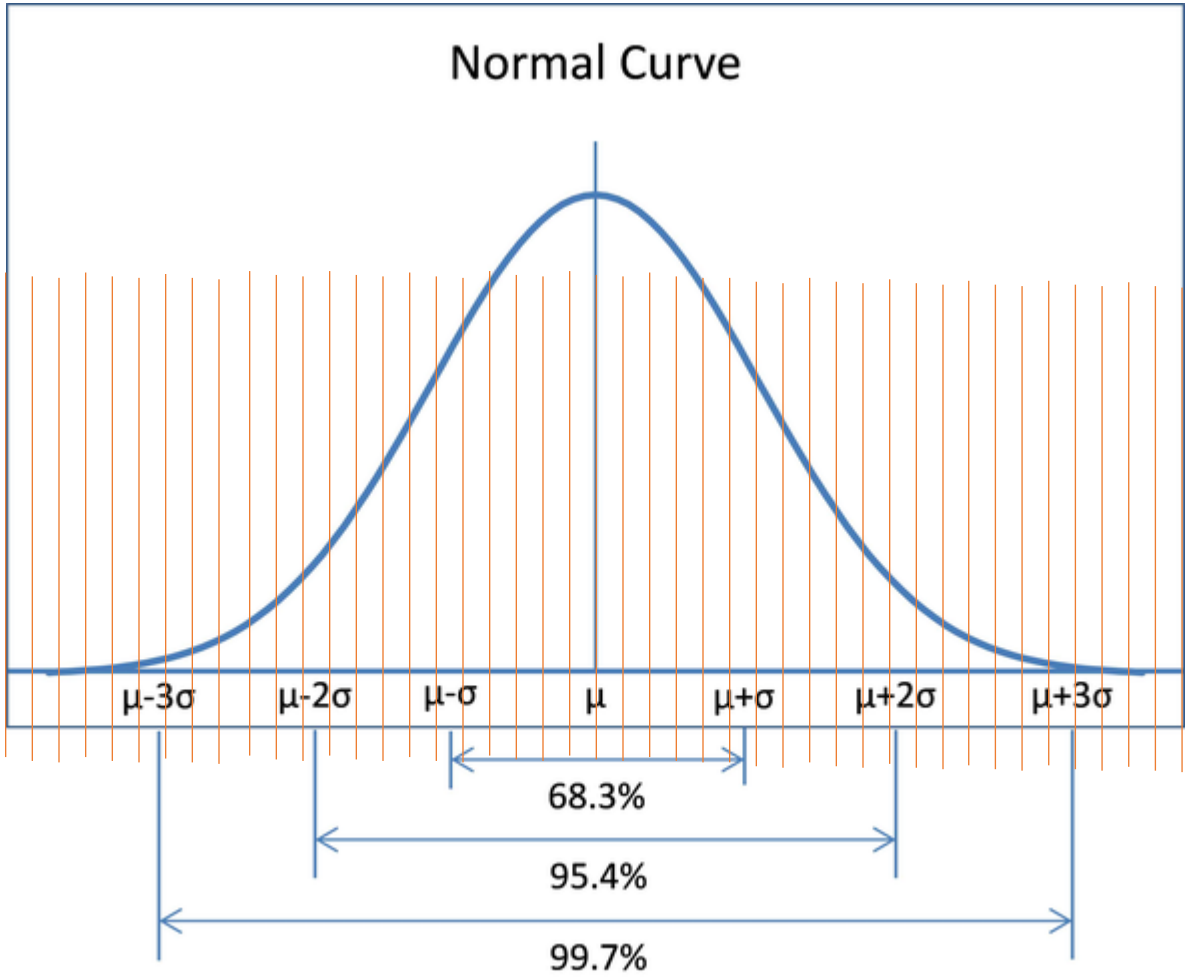
- 这是一种针对大语言模型的量化方案，包含缓存量化与权重量化两个内容。
- 如果我们的应用 Batchsize != 1，则缓存量化所带来的性能收益更大。
- 如果我们的应用 Batchsize == 1，权重量化是必须的，并将带来成倍性能提升。
- 权重量化在较大 Batchsize 时几乎没有收益。

- Weight Only 量化的精度由数制和分组大小决定，并不一定需要线性量化。
- Weight Only 量化的难度主要在于实现 Kernel。



关于 Weight only 量化

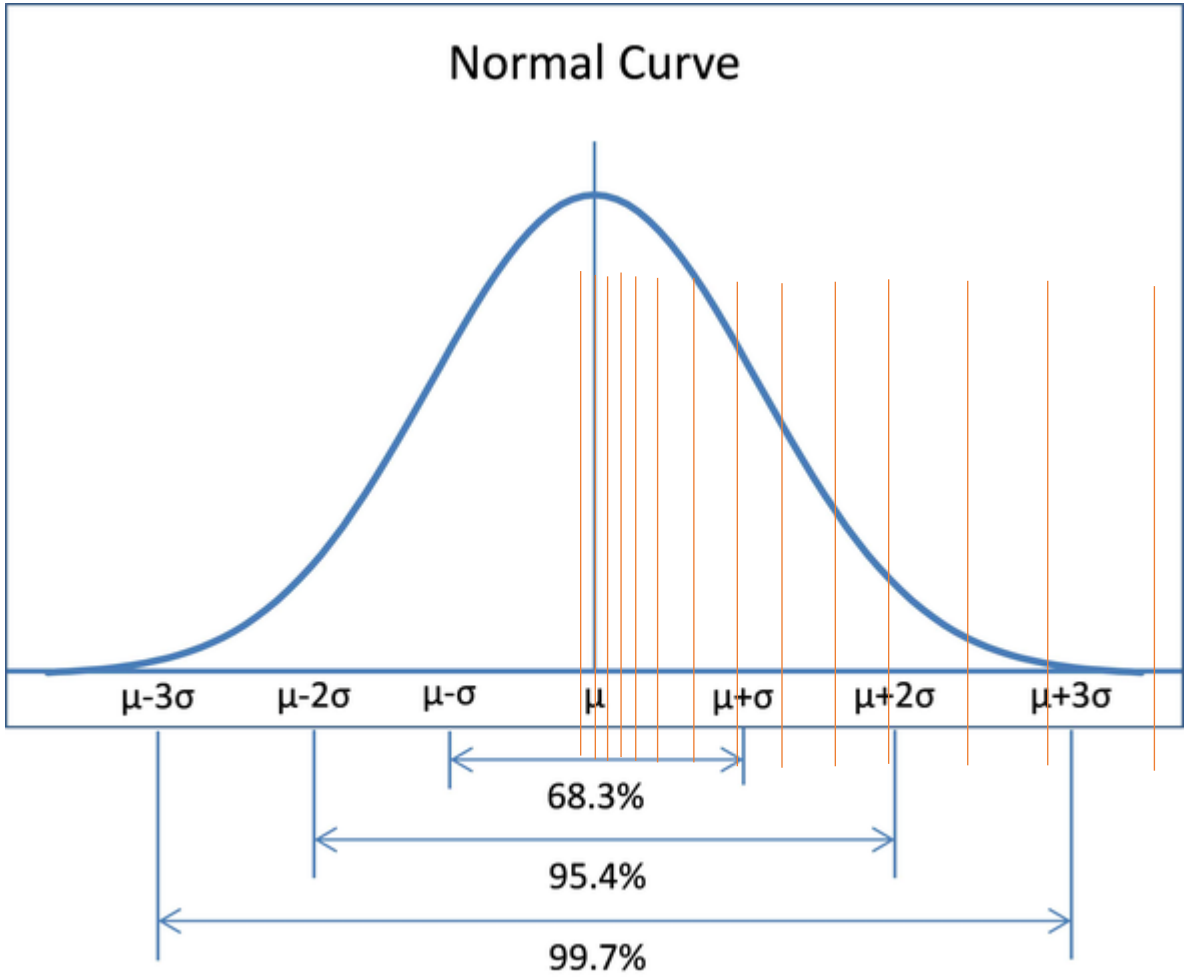
```
__device__ half dhDequantizeNF4(unsigned char val)
{
  // the values for this tree was generated by test_normal_map_tree
  // in the file tests/test_functional.py
  if((val & 0b1000) == 8)
    if((val & 0b0100) == 4) // 1
      if((val & 0b0010) == 2) // 11
        if((val & 0b0001) == 1) // 111
          return 1.0f;
        else
          return 0.7229568362236023f;
      else
        if((val & 0b0001) == 1) // 110
          return 0.5626170039176941f;
        else
          return 0.44070982933044434f;
    else
      if((val & 0b0010) == 2) //10
        if((val & 0b0001) == 1) // 101
          return 0.33791524171829224f;
        else
          return 0.24611230194568634f;
      else
        if((val & 0b0001) == 1) // 100
          return 0.16093020141124725f;
        else
          return 0.07958029955625534f;
```



线性量化器对正态分布的量化效果不是最优的！

关于 Weight only 量化

```
__device__ half dhDequantizeNF4(unsigned char val)
{
  // the values for this tree was generated by test_normal_map_tree
  // in the file tests/test_functional.py
  if((val & 0b1000) == 8)
    if((val & 0b0100) == 4) // 1
      if((val & 0b0010) == 2) // 11
        if((val & 0b0001) == 1) // 111
          return 1.0f;
        else
          return 0.7229568362236023f;
      else
        if((val & 0b0001) == 1) // 110
          return 0.5626170039176941f;
        else
          return 0.44070982933044434f;
    else
      if((val & 0b0010) == 2) //10
        if((val & 0b0001) == 1) // 101
          return 0.33791524171829224f;
        else
          return 0.24611230194568634f;
      else
        if((val & 0b0001) == 1) // 100
          return 0.16093020141124725f;
        else
          return 0.07958029955625534f;
```



线性量化器对正态分布的量化效果不是最优的！

量化是一种有损压缩

我们目前缺乏有效的手段度量大语言模型的生成质量

量化是一种有损压缩

我们目前缺乏有效的手段度量大语言模型的生成质量

量化是一种有损压缩

Table 3: Experiments comparing 16-bit BrainFloat (BF16), 8-bit Integer (Int8), 4-bit Float (FP4), and 4-bit NormalFloat (NF4) on GLUE and Super-NaturalInstructions. QLoRA replicates 16-bit LoRA and full-finetuning.

Dataset	GLUE (Acc.)	Super-NaturalInstructions (RougeL)				
Model	RoBERTa-large	T5-80M	T5-250M	T5-780M	T5-3B	T5-11B
BF16	88.6	40.1	42.1	48.0	54.3	62.0
BF16 replication	88.6	40.0	42.2	47.3	54.9	-
LoRA BF16	88.8	40.5	42.6	47.1	55.4	60.7
QLoRA Int8	88.8	40.4	42.9	45.4	56.5	60.7
QLoRA FP4	88.6	40.3	42.4	47.5	55.6	60.9
QLoRA NF4 + DQ	-	40.4	42.7	47.7	55.3	60.9

我们目前缺乏有效的手段度量大语言模型的生成质量

其他模型压缩技术

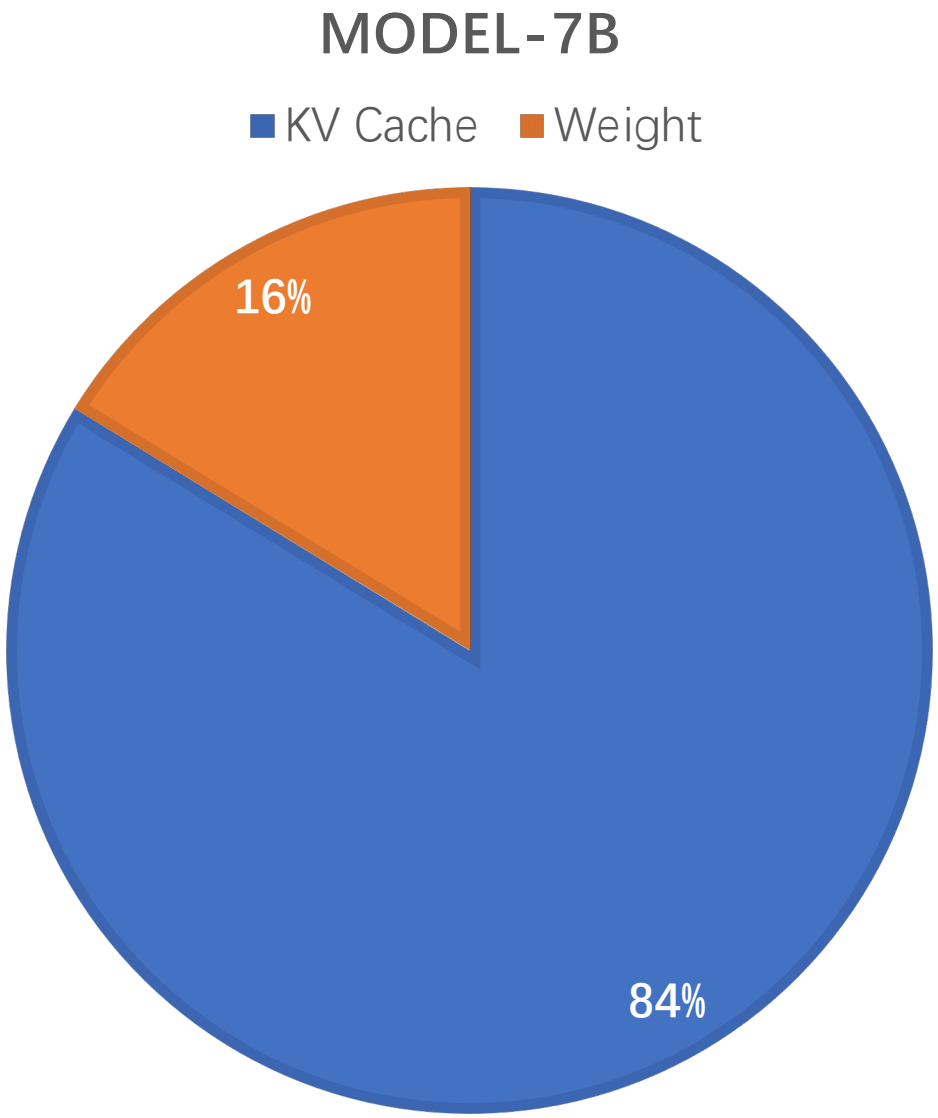
- Paged Attention(大幅压缩缓存空间占用)
- Multi Query Attention(https://arxiv.org/abs/1911.02150)

Scenario

- Mobile(Small model + batchsize 1)
  - 没计算量，算子均为访存密集型
  - 没Batchsize，KV 缓存占很少的空间
  - 设备内存十分有限

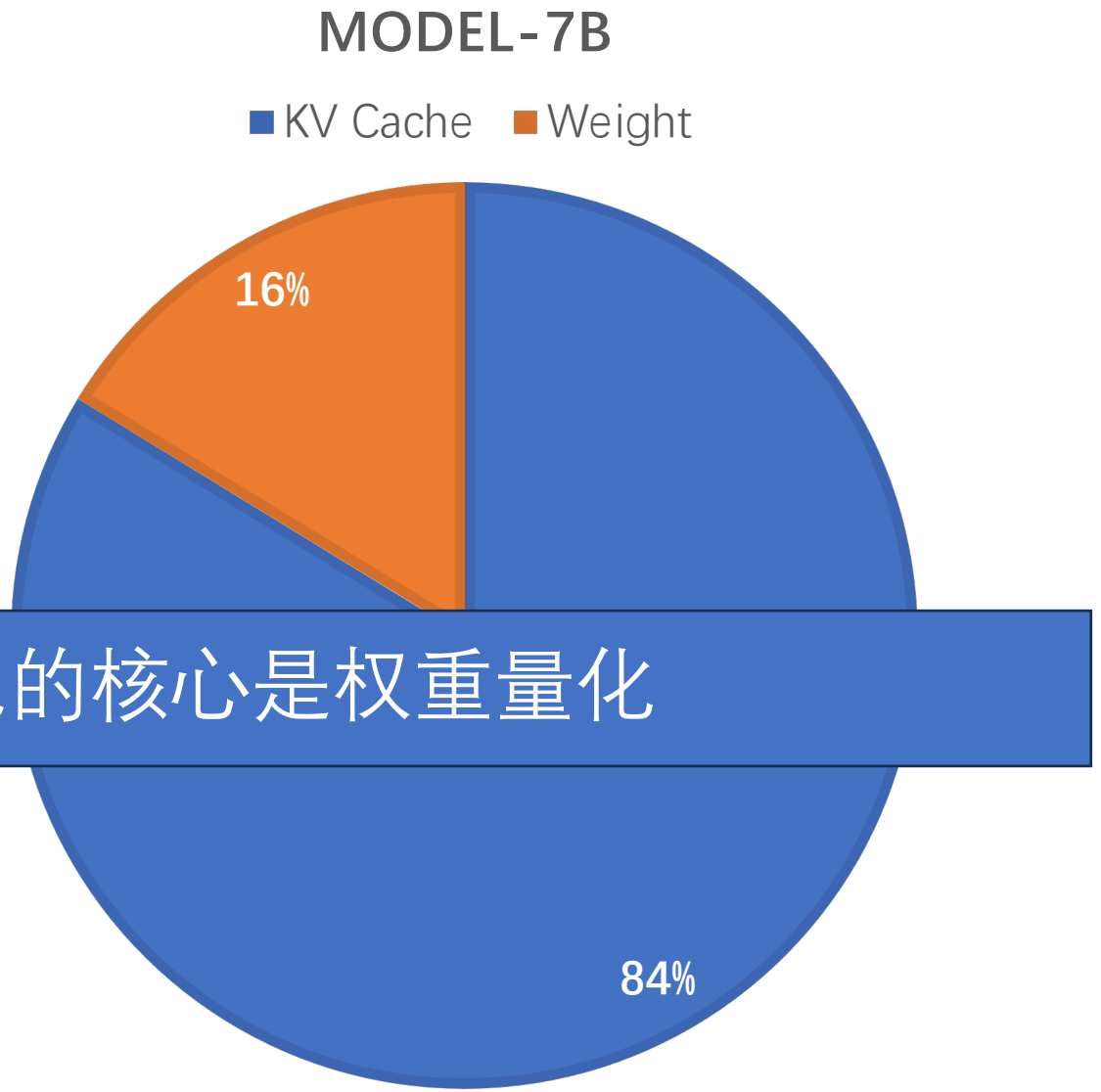
Scenario

- Server(Small model + batchsize N)
  - 计算压力最大
  - Batchsize最大
  - KV 缓存占大部分空间
  - 成本最低



Scenario

- Server(Small model + batchsize N)
  - 计算压力最大
  - KV 缓存占大部分空间
  - 成本最低



制约移动端推理性能的核心是权重量化

Performance Estimation(7B Model FP16)

型号	显存容量 (GB)	显存带宽 (GB/s)	算力 (TFLOPS)	Batchsize	最大吞吐量 (Tokens/sec)	计算延迟 (ms)	访存延迟 (ms)
RTX 4090	24	1008	165.2	47	1572.9	9.7	29.76
L40	48	864	181.05	140	2022.3	26.5	69.44
L4	24	300	121	47	468.1	13.2	100.00
A100-80G-SXM4	80	2039	312	265	5408.7	29.1	49.04
A30	24	933	165	47	1455.8	9.7	32.15
A40	48	696	149.7	140	1629.0	32.1	86.21
A10	24	600.2	125	47	936.5	12.8	49.98
A2	16	200	18	16	156.0	29.6	100.00

Performance Estimation(7B Model FP16)

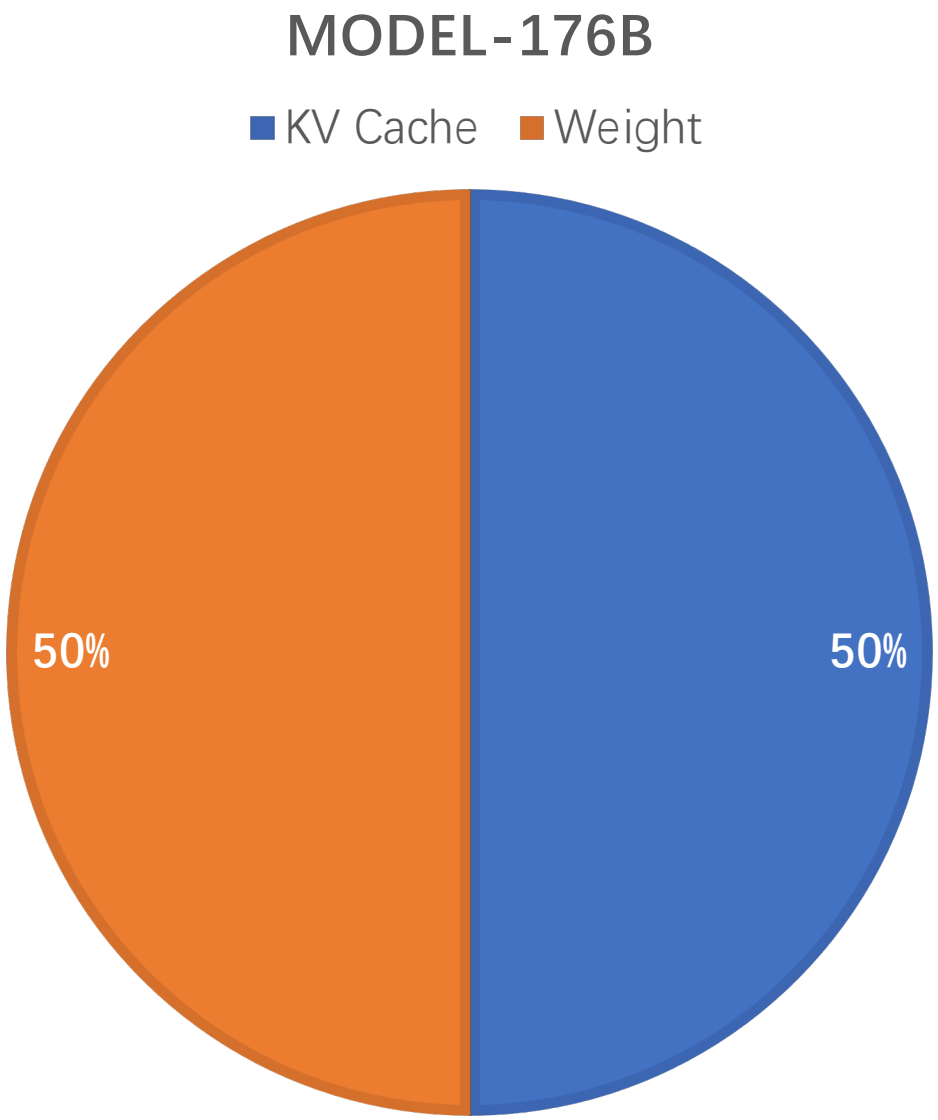
模型	Data精度	平台	Batch	最大推理速度(token / s)
ChatGLM-6b-int4	float32	RTX 4090	1	176
ChatGLM-6b-int8	float32	RTX 4090	1	121
ChatGLM-6b-fp16	float32	RTX 4090	64	2919
ChatGLM-6b-fp16	float32	RTX 4090	256	7871
ChatGLM-6b-fp16	float32	RTX 4090	512	10209
ChatGLM-6b-int4	float32	Xiaomi 10 Pro - 4 Threads	1	4 ~ 5

Performance Estimation(7B Model FP16)

制约服务端推理性能的核心是最大并发数量(Batchsize)

Scenario

- Server(Large model + batchsize N)
  - 有一定计算压力
  - Batchsize适中，KV 缓存占75%以上空间
  - 有通信压力



Performance Estimation (176B Model INT4)

型号	架构代号	显存容量 (GB)	显存带宽 (GB/S)	算力 (TFLOPS)	BATCHSIZE	最大吞吐量 (TOKENS/SEC)	计算延迟(MS)	访存延迟(MS)	通信延迟(MS)
L40	Ada lovelace	48	864	181.05	83	398.4	15.4	53.76	156.87
A100-80G-SXM4	Ampere	80	2039	312	154	3805.8	16.4	36.79	3.86
RTX 4090	Ada lovelace	24	1008	165.2	30	374.0	6.2	24.82	57.67
A30	Ampere	24	933	165	30	365.2	6.2	26.81	57.67
A40	Ampere	48	696	149.7	83	375.3	18.6	66.73	156.87
A10	Ampere	24	600.2	125	30	310.5	8.2	41.68	57.67
A2	Ampere	16	200	18	13	115.5	24.3	89.37	24.60

总结

- 在大模型时代，我们有多种多样的压缩技术，选择正确的方向尤为重要。
- 在不同的场景下，我们需要使用不同的技术组合对模型进行优化。
- 学会写 Cuda，学会推理优化