【FlashAttention-V4，非官方】FlashDecoding++

目录

建议先阅读专栏： LLM加速

null

null

null

1. Introdcution

为了提高softmax并行性，之前方法（FlashAttention、FlashDecoding）将计算过程拆分，各自计算partial softmax结果，最后需要通过同步操作来更新partial softmax结果。例如FlashAttention每次计算partial softmax结果都会更新之前的结果，而FlashDecoding是在最后统一更新所有partial softmax结果。

本文在A100 GPU上分析了输入长度为1024的情况，这种同步partial softmax更新操作占Llama2-7B推理的注意力计算的18.8%。**（本文没说是FlashAttention还是FlashDecoding的结果，个人认为FlashDecoding的同步更新代价并不大，应该远小于18.8%）**

这是LLM推理加速的第一个挑战。此外，本文还提出了两个挑战：

1. 在解码阶段，Flat GEMM操作的计算资源未得到充分利用。这是由于解码阶段是按顺序生成token（一次只生成一个token），GEMM操作趋于flat-shape，甚至batch size等1时变成了GEMV（General Matrix-Vector Multiplication），具体看论文Figure 2。当batch size较小时（e.g., 8），cublas和cutlass会将矩阵填充zeros以执行更大batchsize（e.g., 64）的GEMM，导致计算利用率不足50%。

2. 动态输入和固定硬件配置影响了LLM推理的性能。例如，当batch size较小时，LLM推理的解码过程是memory-bounded，而当batch size较大时是compute-bounded。

针对这3个问题，本文分别提出了对应优化方法：

1. **Asynchronized softmax with unified max value.** FlashDecoding++为分块softmax计算设置了一个共享的最大值。这样可以独立计算partial softmax，无需同步更新。
2. **Flat GEMM optimization with double buffering.** FlashDecoding++只将矩阵大小填充到8，对比之前针对flat-shaped GEMM设计的为64，提高了计算利用率。论文指出，具有不同shape的flat GEMMs面临的瓶颈也不同，于是进一步利用双缓冲等技术提高kernel性能。
3. **Heuristic dataflow with hardware resource adaption.** FlashDecoding++同时考虑了动态输入和硬件配置，针对LLM推理时数据流进行动态kernel优化。

下图展示了以上3种方法的示意图：

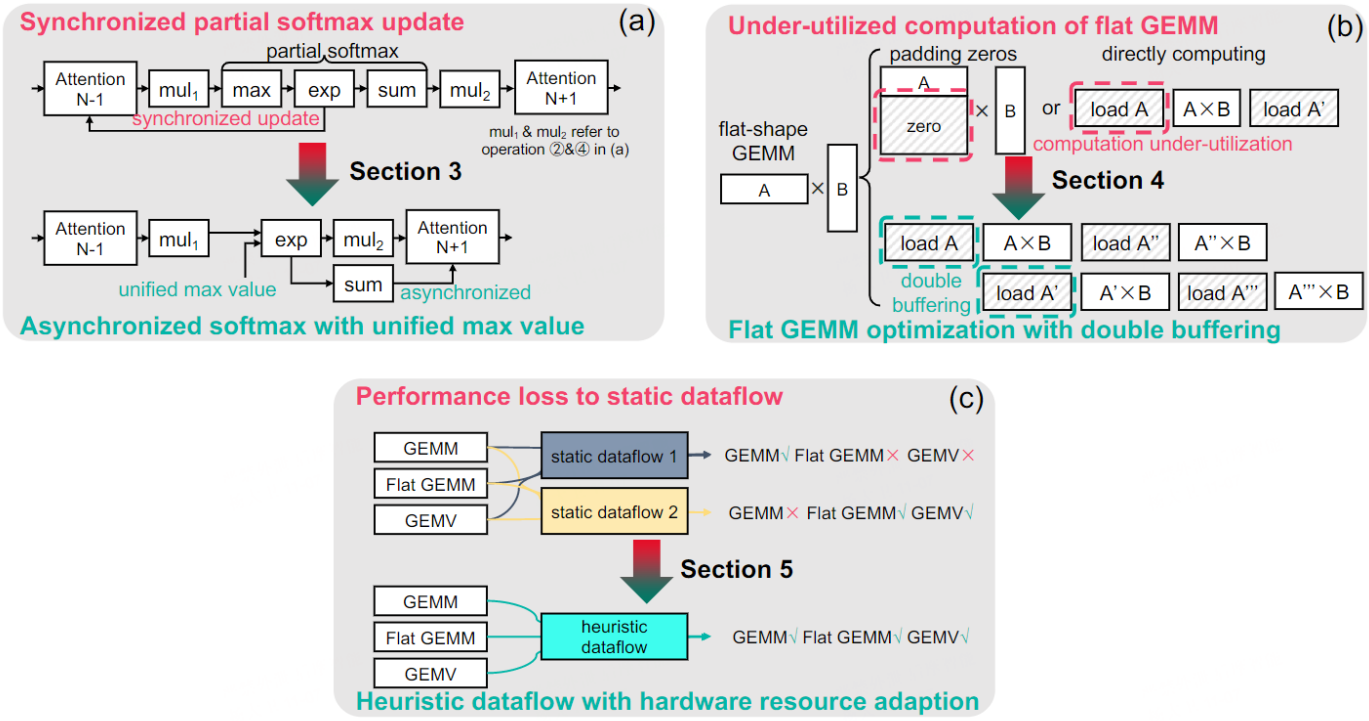


Figure 3: *FlashDecoding++* proposes three solutions for corresponding challenges in Large Language Model inference. (a) *FlashDecoding++* proposes the asynchronized softmax with unified max value technique, avoiding synchronized update to previous partial attention results. (b) *FlashDecoding++* optimizes flat GEMM by improving computation utilization. (c) *FlashDecoding++* heuristically optimizes dataflow.

Untitled

2. Backgrounds

LLM推理中的主要操作如下图所示：linear projection(①和⑤)、attention(②、③和④)和feedforward network(⑥)。为简单起见，这里忽略了position embedding、non-linear activation、mask等操作。本文将LLM推理时对Prompt的处理过程称为*prefill*phase，第二阶段预测过程称为*decode*phase。**这两个阶段的算子基本一致，主要是输入数据的shape是不同的。** 由于*decode*phase一次只处理一个令牌（batch size=1，或batch size很小），因此输入矩阵是flat-shape matrices（甚至是vectors），参见下图Decode phase部分中和KV Cache拼接的红色向量。

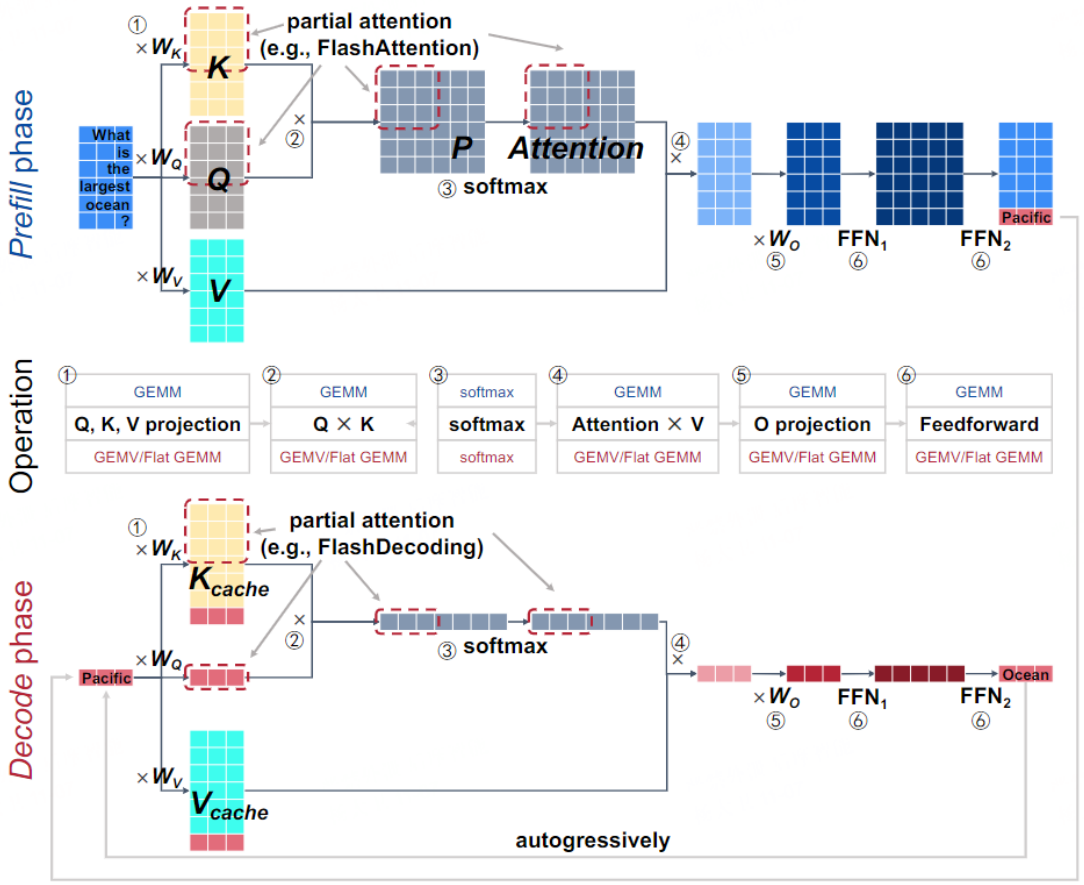


Figure 2: Overview of Large Language Model inference dataflow. We show the dataflow comparison between the *prefill* phase and the *decode* phase. The *prefill* phase mainly involves the GEMM operation, while the *decode* phase mainly involves the GEMV/Flat GEMM operation.

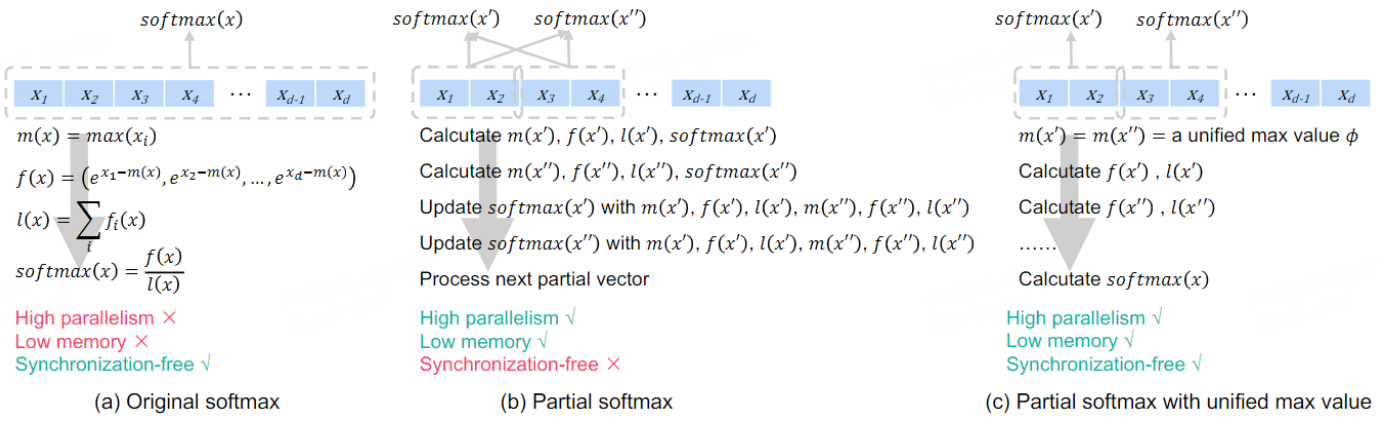
Untitled

LLM推理中的另一个问题就是Softmax算子，其需要计算并存储所有全局数据，并且数据量随着数据长度成平方增长，存在内存消耗高和低并行性等问题。一般计算流程如下：

$$
\begin{aligned}
m(x) &= \max\left(m\left(x'\right), m\left(x''\right)\right) \\
f\left(x'\right) &= e^{m(x')-m(x)} f\left(x'\right) \\
f\left(x''\right) &= e^{m(x'')-m(x)} f\left(x''\right) \\
l(x) &= f\left(x'\right) + f\left(x''\right) \\
\operatorname{softmax}\left(\left[x', x''\right]\right) &= \left[f\left(x'\right), f\left(x''\right)\right] \nabla \cdot l(x)
\end{aligned} \tag{1}
$$

3. Asynchronized Softmax with Unified Maximum Value

如下图b所示，FlashAttention和FlashDecoding对softmax操作进行了分块处理，但是块与块之间需要进行同步（主要是局部最大值）。本文发现这种同步操作的开销约为20%。因此，作者希望去除同步操作，也就是独立计算出partial softmax结果。



Untitled

其实方案也很简单，就是找到一个合适的公共最大值ϕ。然而，如果ϕ太大，会造成$e^{x_i-\phi}$溢出；如果ϕ太小，会造成$e^{x_i-\phi}$精度损失。于是作者进行了统计，如下图所示。例如，对于Llama2-7B， >超过99.99%的值在[-16.8, 6.5]之间。

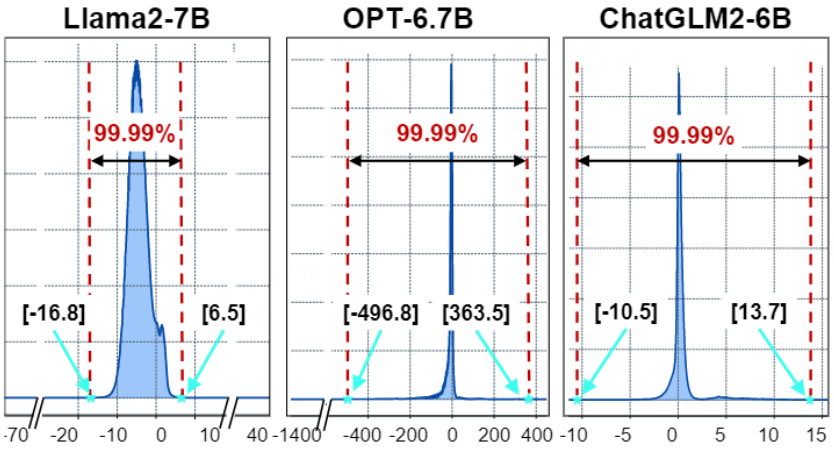


Figure 5: The statistical distribution of x_i (elements in the input vectors of softmax) in typical LLMs with different inputs.

Untitled

但是对于OPT-6.7B来说，其范围较大，于是作者采用动态调整策略，如果在推理过程中发现设置的ϕ不合理，那么就终止当前操作，然后采用FlashAttention和FlashDecoding的方法计算softmax。下图b中展示当e^{9-6}超过阈值e^3时的recomputation过程。

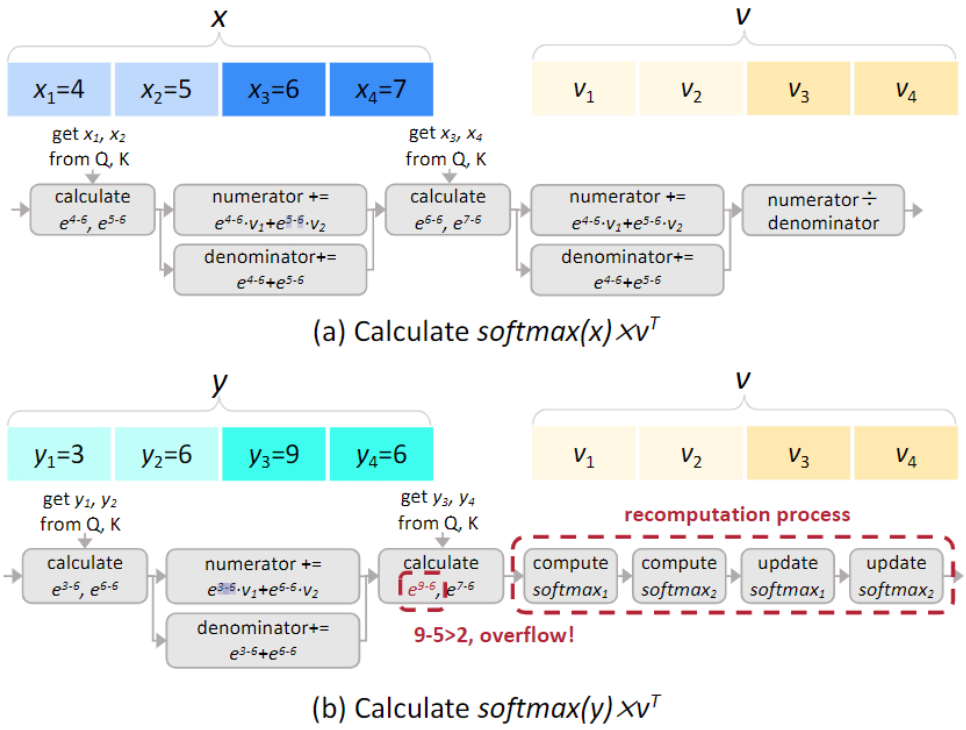


Figure 6: Example of asynchronized partial softmax computation. (a) Each partial softmax result is process individually without the synchronized update. (b) The recomputation process for all parital softmax computation is required when overflow happens.

Untitled

4. Flat GEMM Optimization with Double Buffering

Decoding阶段的过程主要由GEMV（batch size=1）或flat GEMM（batch size>1）。GEMV/GEMM运算可以用M、N、K来表示，其中两个相乘矩阵的大小分别为M × K和K × N。一般LLM推理引擎利用Tensor Core使用cuBLAS和CUTLASS等库来加速。尽管Tensor Core适合处理M = 8的GEMM，但这些库为了隐藏memory latency，通常将M维度平铺到64。然而，*decode*phase的GEMV或flat GEMM的M通远小于64，于是填充0到64，导致计算利用率低下。

若假设N维度上和K维度上的tiling size分别为B_N和B_K，那么每个GEMM tile的计算量为$2 \times M \times B_N \times B_K$（这里的2表示乘加2次），总共有$B = \frac{N \times K}{BN \times BK}$个GEMM tiles。总内存访问量为$\left(M \times B_K + B_N \times B_K\right) \times B + M \times N$。因此，计算和内存比为：

$$\begin{aligned} & \frac{2 \times M \times B_N \times B_K \times B}{\left(M \times B_K + B_N \times B_K\right) \times B + M \times N} \\ = & \frac{2 \times M \times K}{K + \frac{M \times K}{B_N} + M} \end{aligned} \tag{2}$$

另一方面，tiling后的并行度是N/B_N。**于是发现了GEMV或falt GEMM两者矛盾之处：计算和内存比与 B_N正相关，而并行度与 B_N负相关。** 下图展示了GEMM在不同B_N和N下的性能（归一化后）。本文总结了两个关键结论：

1. 当N较小时，flat GEMM是parallelism-bounded。NVIDIA Tesla A100中有108个Streaming Multiprocessors (SMs)，于是应该将N/B_N设置为一个相关的数（128或256）。
2. 当N较大时，flat GEMM是memory-bounded。通过隐藏memory access latency可以提高性能。

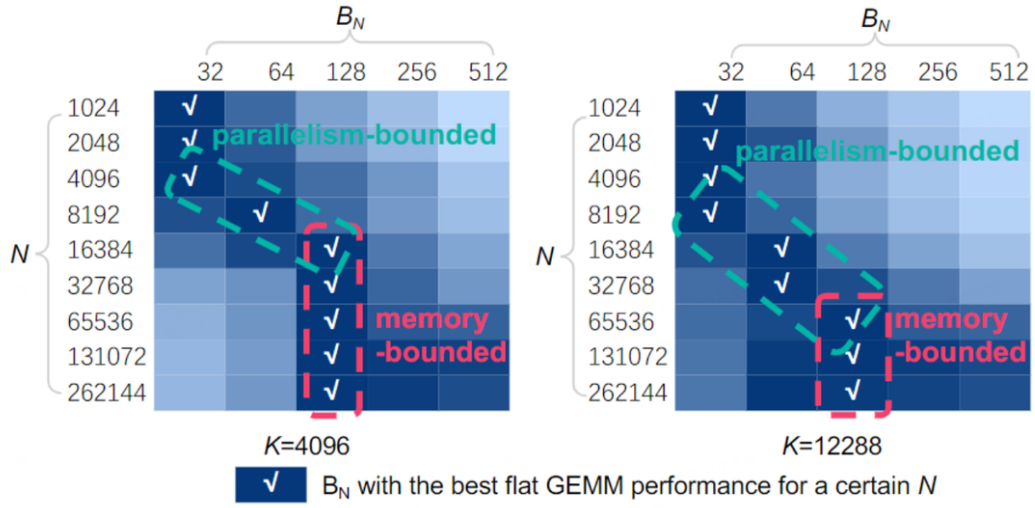


Figure 7: Normalized flat GEMM performance under different $N-$dimension sizes and $N-$dimension tiling sizes. We set $M = 8$ and execute GEMM on the NVIDIA Tesla A100 GPU.

Untitled

为了隐藏memory access latency，本文引入了double buffering技术。具体来说就是在共享内存中分配两个buffer，一个buffer用于执行当前tile的GEMM计算，同时另一个buffer则加载下一个tile GEMM所需的数据。这样计算和内存访问是重叠的，本文在N较大时采取这种策略，下图为示意图。

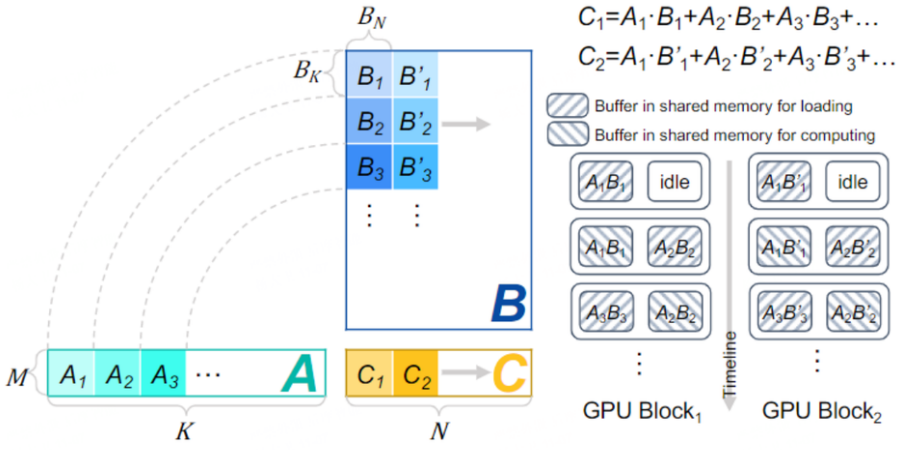


Figure 8: Double buffering for flat GEMM when $N-$dimension is large. The $M-$ dimension is padded to 8 and not tiled.

Untitled

5. Heuristic Dataflow with Hardware Resource Adaption

影响LLM推理性能的因素有很多：（a）动态输入。batch size和输入序列长度的变化造成了工作负载变化。（b）模型多样性。主要指模型结构和模型大小。（c）GPU能力不同。例如内存带宽、缓存大小和计算能力。（d）工程优化。

虽然这些因素构建了一个很大的搜索空间，但LLM中不同layer的同质性大大减少了算子优化的搜索空间。例如，*prefill*phase和*decode*phase中有4个GEMV/GEMM操作（K、Q、V投影、O投影、2个FFN），都可以表示为[M, K]和N x K，对应了四种[N, K]组合，如下图所示。此外，*prefill*phase的M与输入序列长度和batch size有关，*decode*phase的M只与batch size有关。

	Operation	M	N	K
***Prefill* phase**	K, Q, V projection	SeqLen*B	HD*3	HD
	O projection	SeqLen*B	HD	HD
	FFN1	SeqLen*B	FD	HD
	FFN2	SeqLen*B	HD	FD
***Decode* phase**	K, Q, V projection	B	HD*3	HD
	O projection	B	HD	HD
	FFN1	B	FD	HD
	FFN2	B	HD	FD

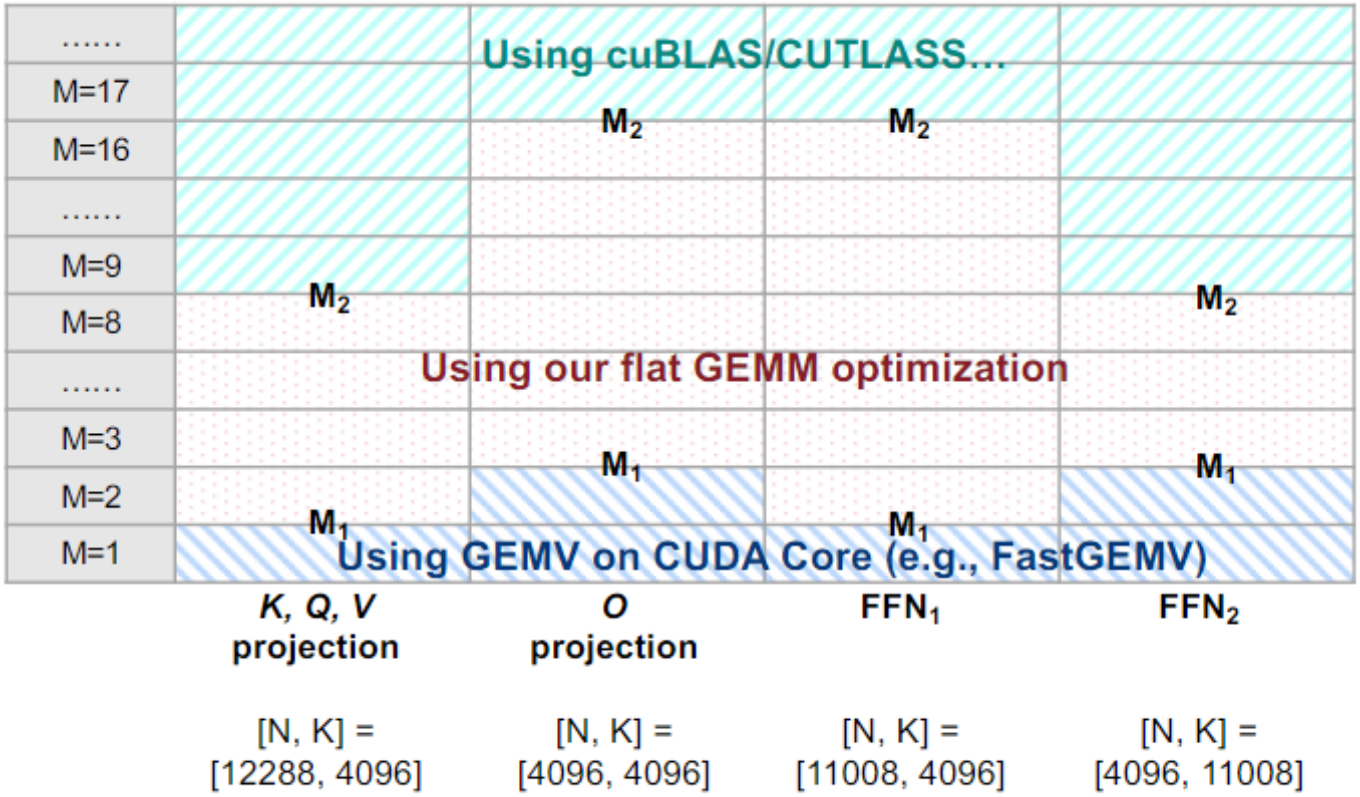
HD: Hidden dimension size
FD: Dimension size after the first FFN
B: Batch size
SeqLen: Input sequence length

Only 4 shapes!

(a) Different shapes of GEMMs in LLM

Untitled

本文根据不同的M, K, N选取FastGEMV、flat GEMM（本文方法）、CUTLASS。



(c) Example of heuristic dataflow with hardware resource adaption

Untitled

个人总结

这篇文章没有FlashAttention和FlashDecoding惊艳，个人觉得FlashDecoding的同步处理代价不大，而且本文中动态调整softmax方法也引入了判断、终止和分支跳转等操作。另一个Double Buffering就是内存优化常用的乒乓buffer，也没什么新东西。

不过话说回来，如今在tranformer架构不变的情况，LLM加速只能靠这些工程手段去优化，的确也有不错效果。还是很有价值的。

最后欢迎感兴趣的小伙伴加入我们组！简历发我邮箱381082014@qq.com，实习和校招我一轮面试即可决定，社招也是我安排面试（hr邮箱有可能过滤）。

深度学习算法实习生招聘

联系方式和地点

hr02@houmo.ai 13813371526（微信同号）

北京/南京/上海/远程

研究方向（Mentor提供论文指导）

- 自动驾驶算法研究（目标检测、BEV、点云、Occupancy、DriveGPT等）
- 大模型及多模态算法研究（开放场景的2D/3D感知、模型轻量化设计等）
- 模型加速优化研究（PTQ、QAT、混合精度量化、模型压缩等）
- 软硬件协同设计（AI模型加速、算子硬件化、指令集开发等）

开发方向（Mentor提供工程指导）

- AI工具链开发（模型解析、图优化等）
- AI算子设计和开发（如投影变换、超越函数、LayerNorm、Grid-sample等）
- 模型部署优化（性能优化、Benchmark验证等）

部分研究成果（近1年）

- A 22nm 64kb Lightning-like Hybrid Computing-in-Memory Macro with Compressor-based Adder-tree and Analog-storage Quantizer for Transformer and CNNs, ISSCC 2024
- MIM4DD: Mutual Information Maximization for Dataset Distillation, NeuIPS 2023.
- RPTQ: Reorder-based Post-training Quantization for Large Language Models. arXiv preprint 2023.
- Post-training Quantization on Diffusion Models. CVPR 2023
- PD-Quant: Post-Training Quantization based on Prediction Difference Metric. CVPR 2023.
- Latency-aware Spatial-wise Dynamic Networks, NeurIPS 2022.

- Flatfish: a Reinforcement Learning Approach for Application-Aware Address Mapping. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), 2022.
- PTQ4ViT: Post-Training Quantization Framework for Vision Transformers. European Conference on Computer Vision (ECCV), 2022.
- 3DPEE: 3D Point Positional Encoding for Multi-Camera 3D Object Detection Transformers. ICCV 2023.

👏 后摩智能于2020年在南京成立，是国内首家基于 “存算一体” 技术的智能驾驶芯片高新技术企业，在北京、上海、深圳等地方建有研发中心。后摩智能致力于突破智能计算芯片性能及功耗瓶颈，加速人工智能普惠落地。其提供的大算力、低功耗的高能效比芯片及解决方案，可应用于智能驾驶、泛机器人等边缘端，以及云端推理场景。2023年5月，发布了首款基于SRAM的存算一体大算力AI芯片产品，算力高达256Tops。