LLM模型压缩与推理加速实践

小红书中台技术部–推理加速团队负责人 / 陈 磊

LLM 模型压缩与推理加速实践

小红书

- 领域背景
- 大语言模型压缩
- 推理框架与计算优化
- 总结与展望

领域背景 – LLM推理难点

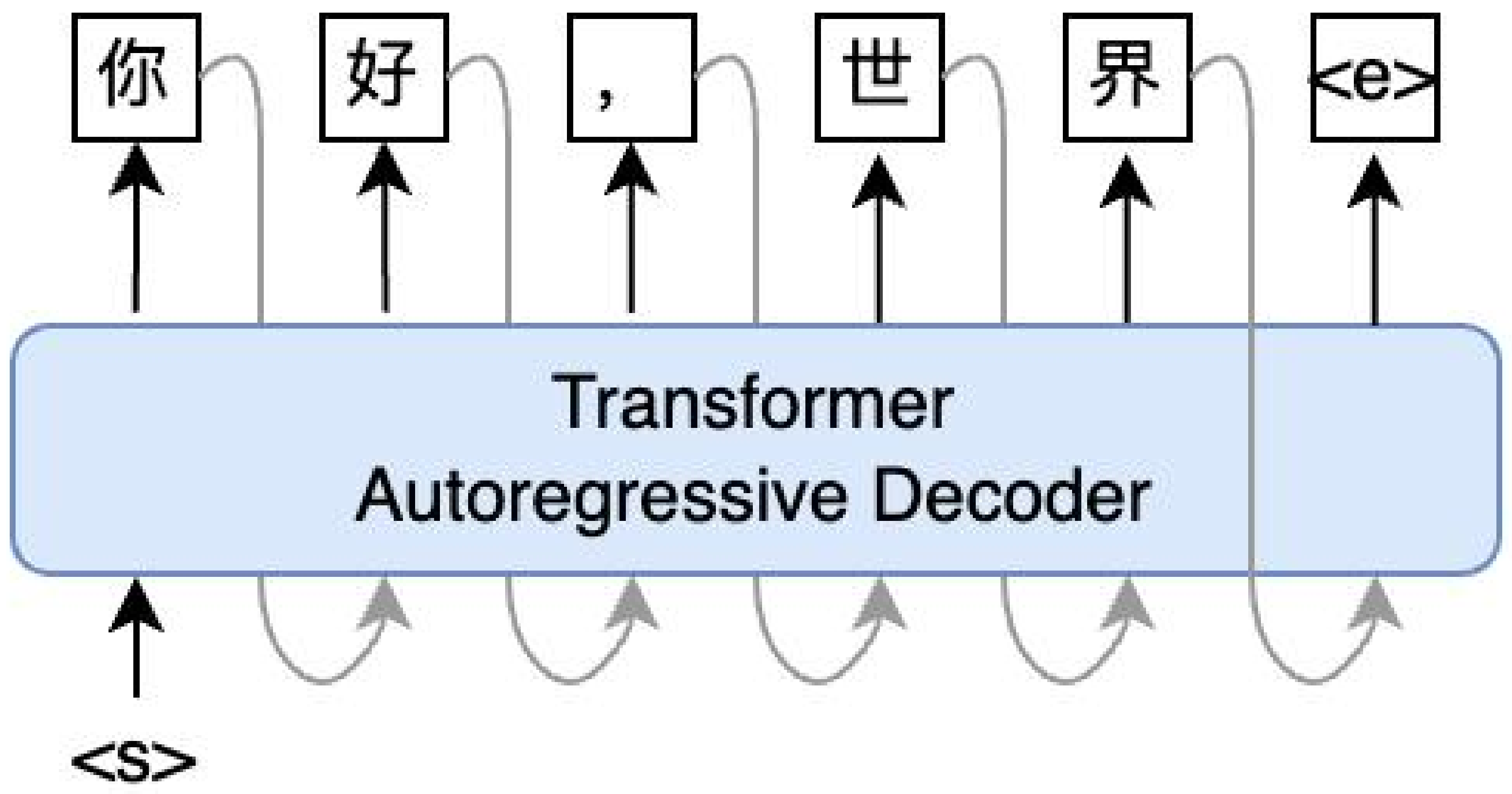
- 巨大的内存/显存需求量

✓对于如下模型和场景：
- ✓Llama 65B 模型
- ✓max_batchsize=64
- ✓max_input_seq_length = 1024
- ✓max_output_seq_length = 512

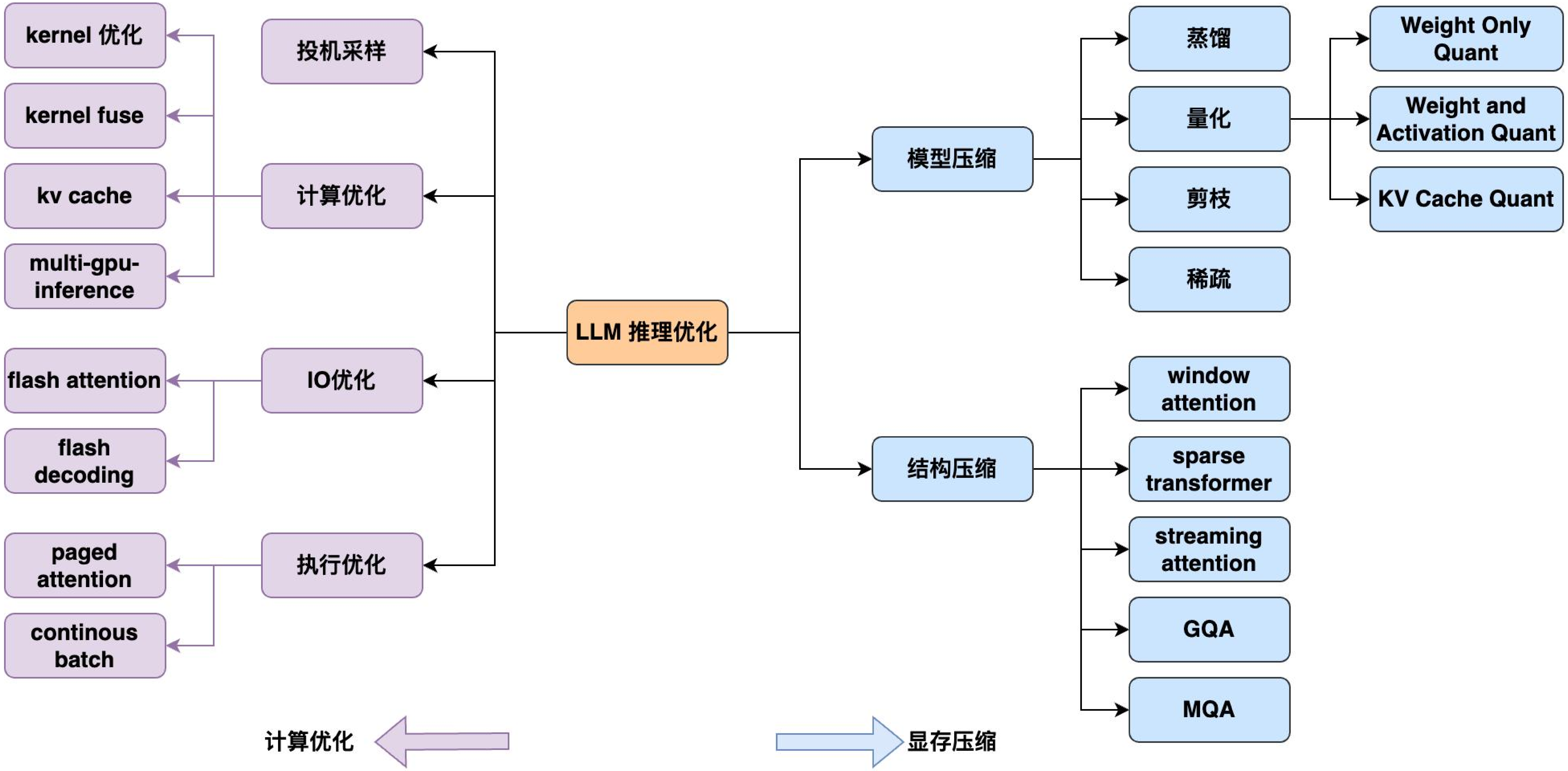
类别	参数量	显存用/GB
Weights	$12lh^2$	120
Key/Value cache	$4blh(s+n)$	240

巨大的部署代价（高延迟、低吞吐、昂贵的高性能GPU），是 LLM 模型能力在产品中真正落地的拦路虎

- 自回归生成过程无法充分并行



领域背景 – LLM推理难点



模型压缩 – 量化原理

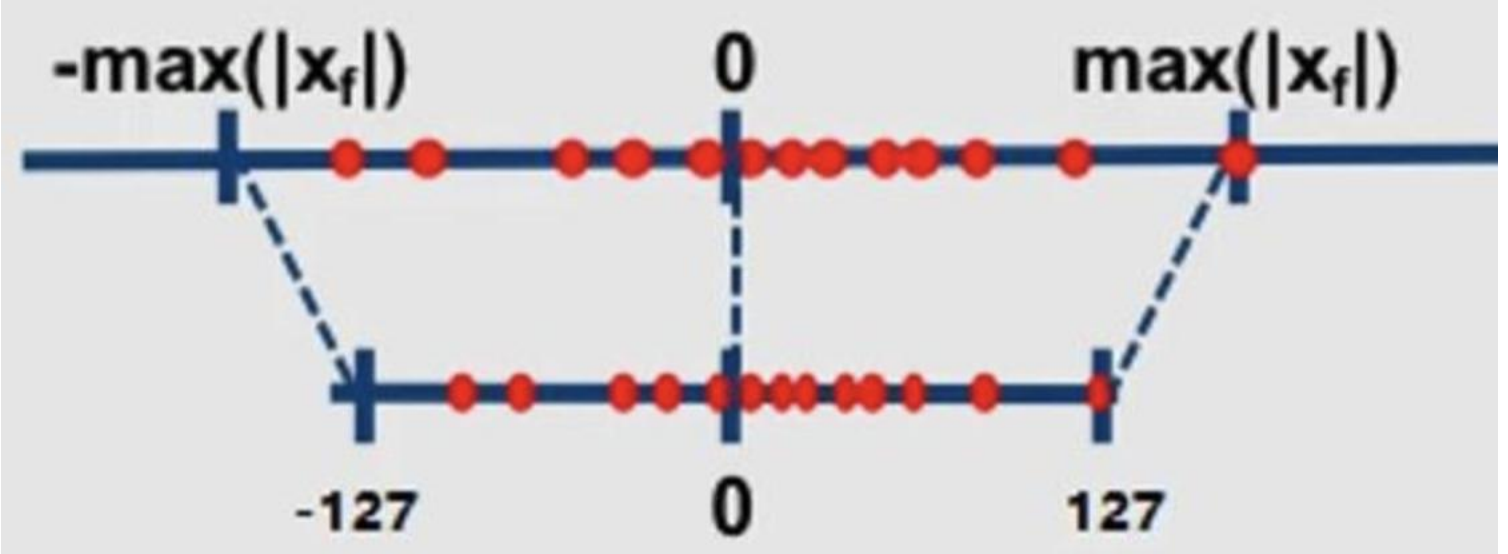
- 对称量化&反量化：

$$Quant(X) = \textcolor{red}{clip}(\textcolor{red}{round}(X * scale), -2^{b-1}+1, -2^{b-1}-1)$$

$$\textcolor{red}{scale} = \left|\frac{2^{N-1}-1}{alpha}\right|, \ N = 8 \ or \ 4$$

$$Dequant(X) = (Quant(X))/\textcolor{red}{scale}$$

$$MSE(X) = |X - Dequant(X)|$$



- Example:

$$X = [10, 2, 3, 4, \textcolor{red}{127}] \xrightarrow{scale = 1} Xq = [10, 2, 3, 4, 127]$$

MSE(X) = [0,0,0,0,0]

$$X = [10, 2, 3, 4, \textcolor{red}{1270}] \xrightarrow{scale = 1/10} Xq = [0, 0, 0, 0, 127]$$

MSE(X) = [10,2,3,4,0]

<u>异常值（outliers）是影响量化误差的重要因素</u>

模型压缩 – W8A8量化

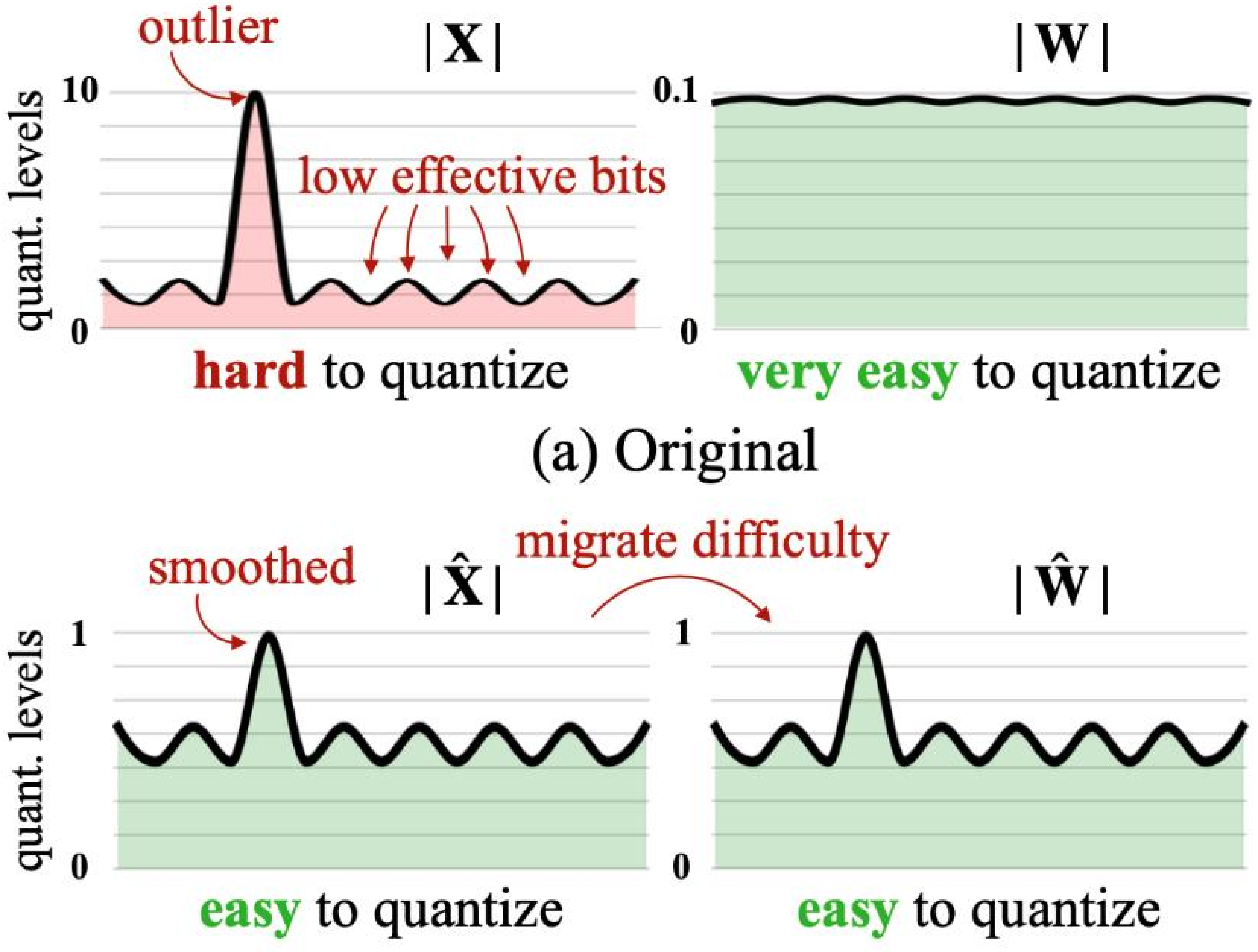
- LLM量化难点：
  - Activation异常值能达到其他值的100x以上；
  - Weight数值分布均衡，容易量化；
  - Activation异常值分布基本集中在特定的若干通道；
- SmoothQuant：

$$Y = \left(\frac{X}{param}\right) * (W * param) = X_p * W_p,$$

$$param = \frac{\max(|X|)^{alpha}}{\max(|W|)^{1-alpha}}$$

- *alpha = 0.5~0.75 is good enough for most models*
- *Apply per-tensor-quant on Weight*
- *Apply per-tensor/per-token-quant on Activation*

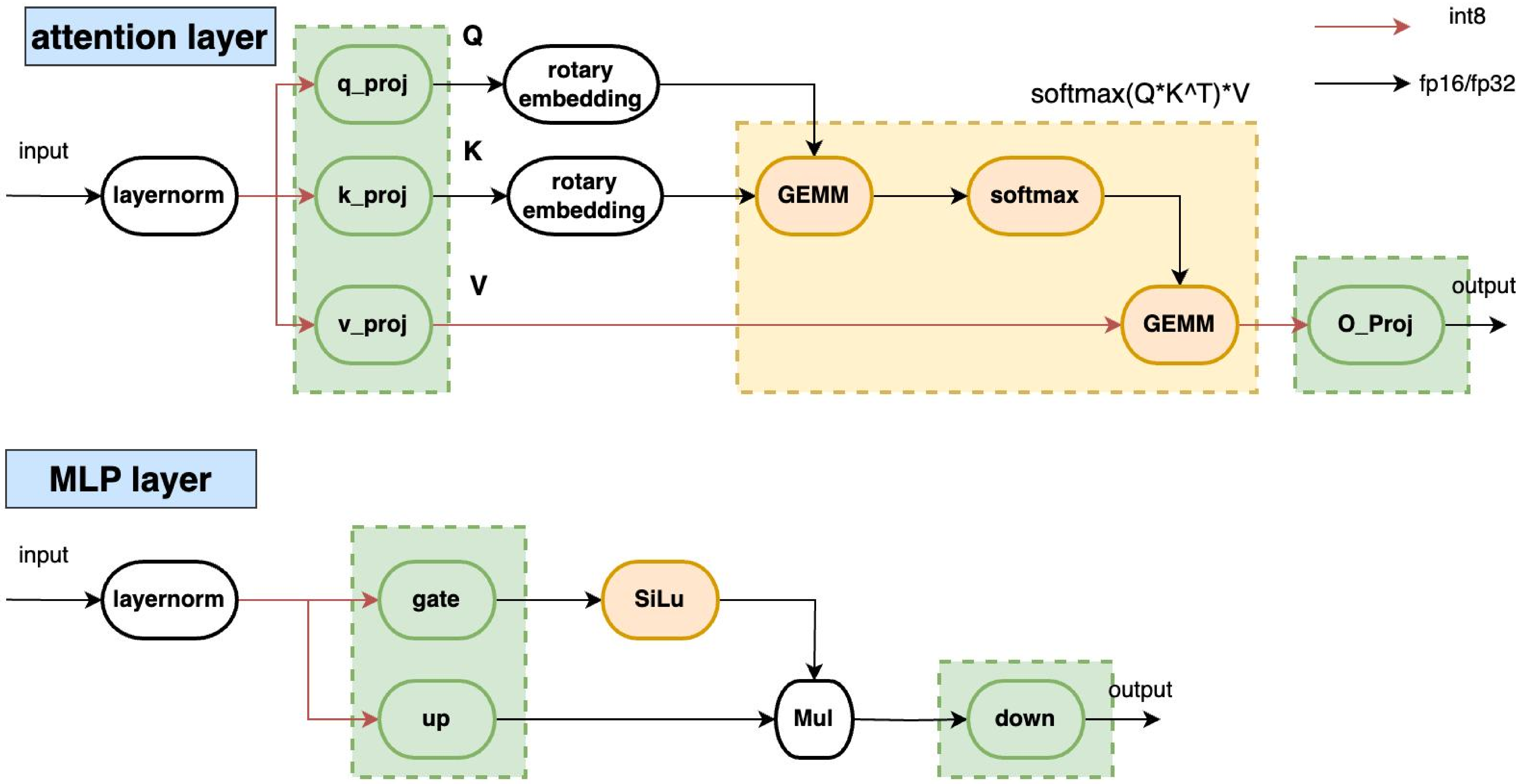
将激活的量化困难“部分转移”给权重



图片来自smoothquant paper

模型压缩 – W8A8量化

- 量化方案：
  - 以 LLAMA 为例，W8A8 量化算法设计如图；
- 实际落地碰到问题及分析：
  - Decoder-only 模型架构，误差逐级累计放大，最终模型精度影响较大；
  - 不同位置的量化带来的误差对精度损失不同，比如mlp.fc2；
  - alpha=0.5 ~ 0.8并非对所有层都合适；



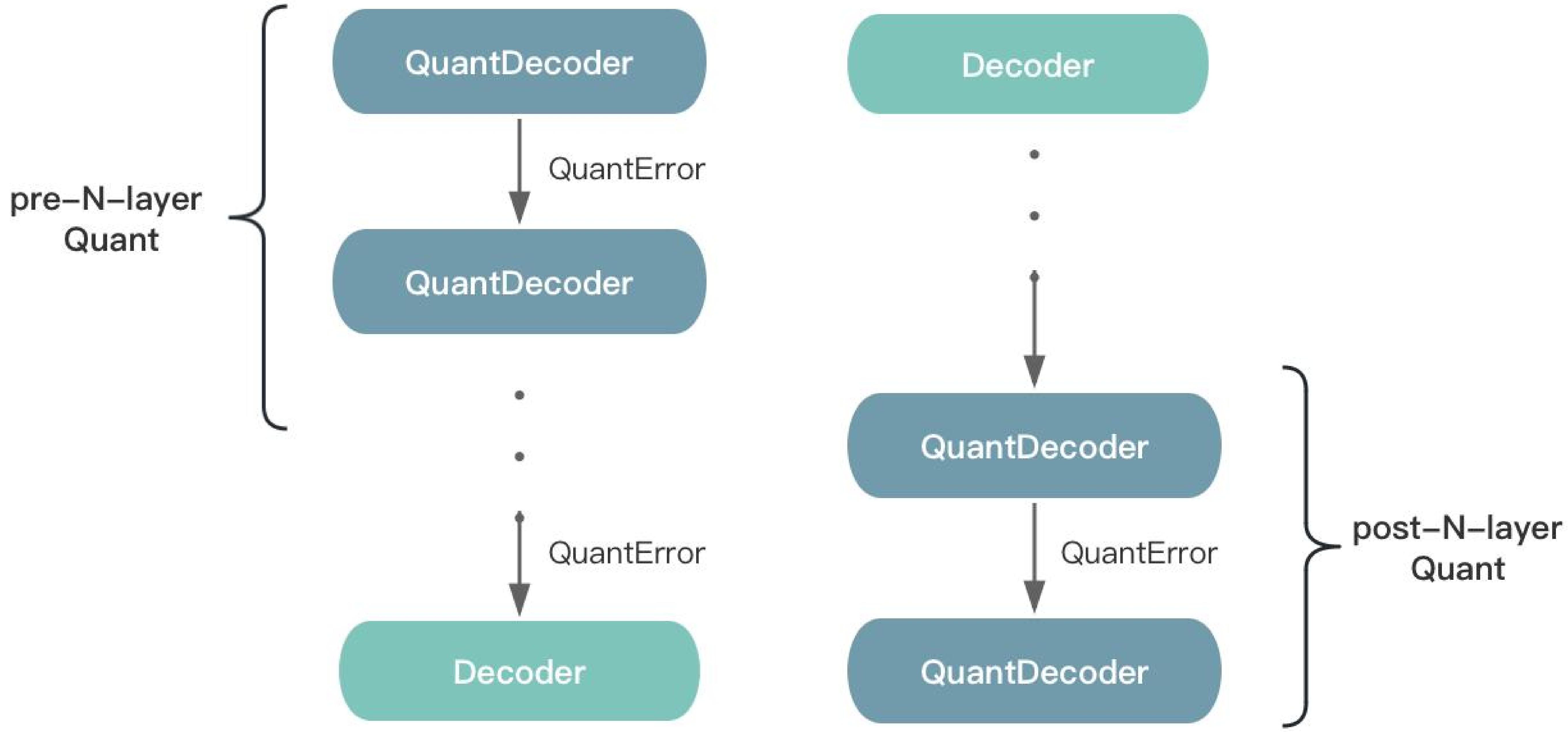
模型压缩 – W8A8量化

改进1：

- 只对部分decode layer 进行量化；

Type	MMCU-Accuracy
FP16	50.04%
ALL-Layer	46.53%
Pre-N-Layer	48.15%
Post-N-Layer	49.17%

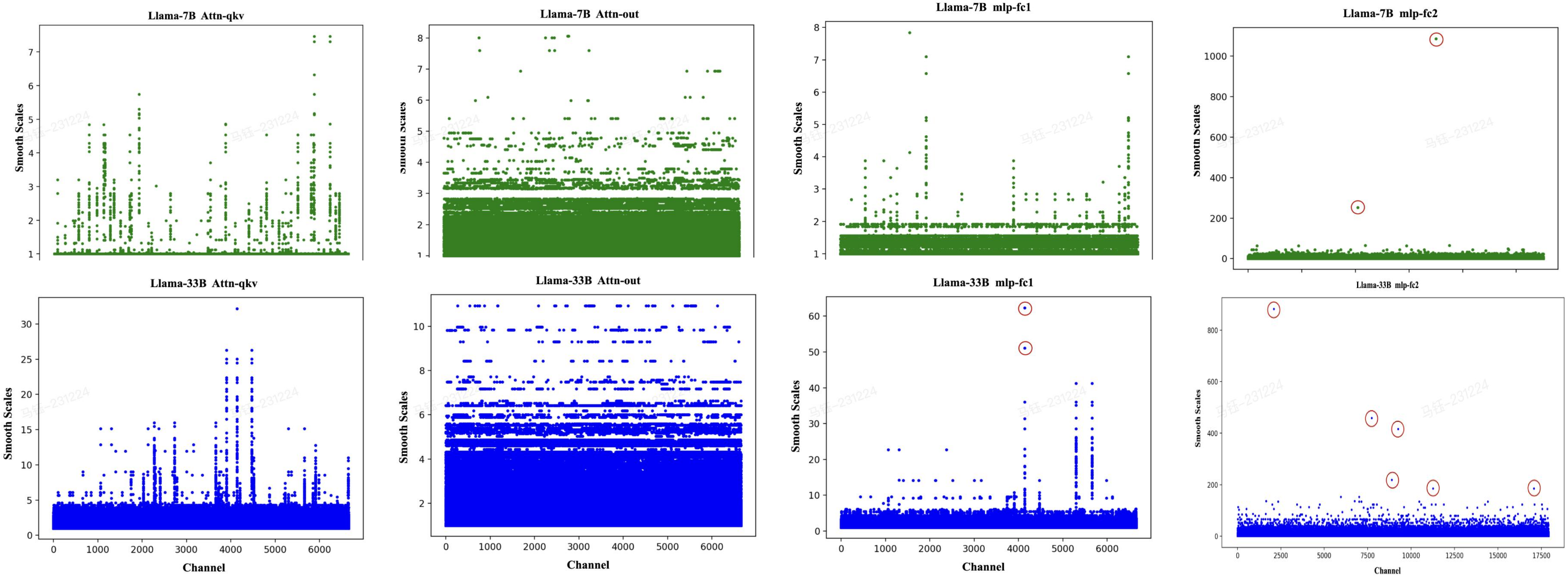
N通过搜索得到



模型压缩 – W8A8量化

改进2：

- Weight采用per-channel

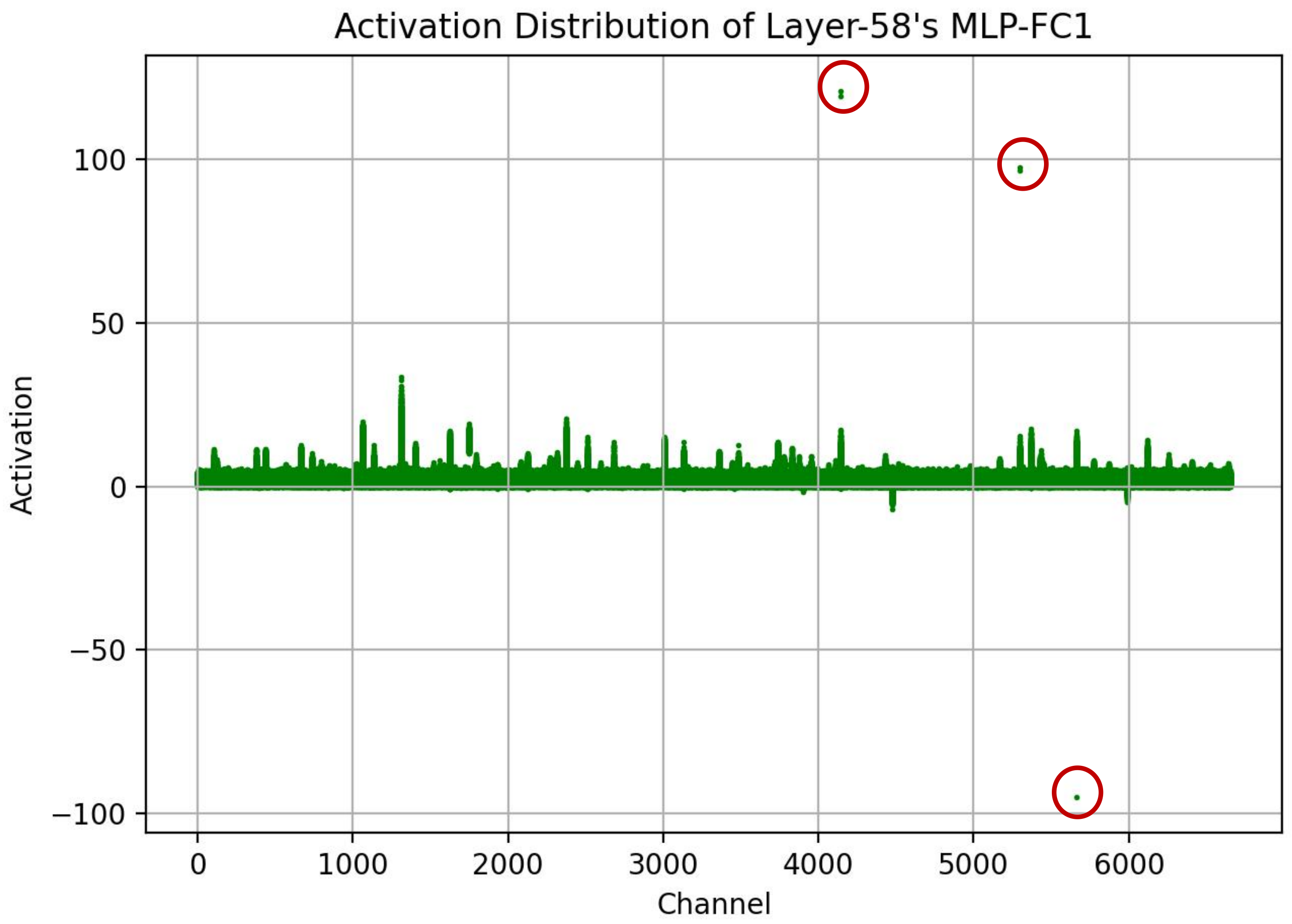


模型压缩 – W8A8量化

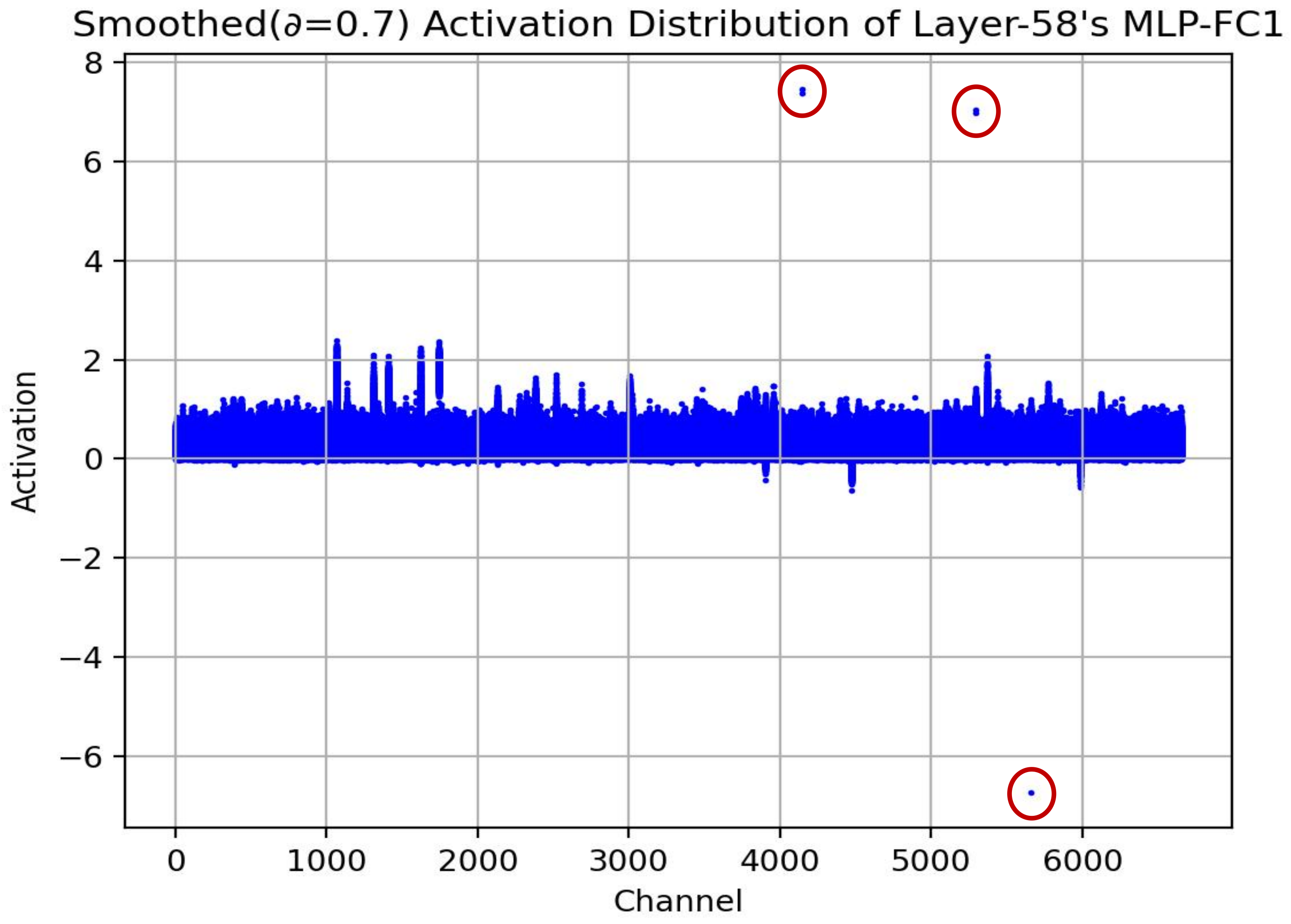
改进3：

- 分层制定不同的 smooth 超参数（alpha）

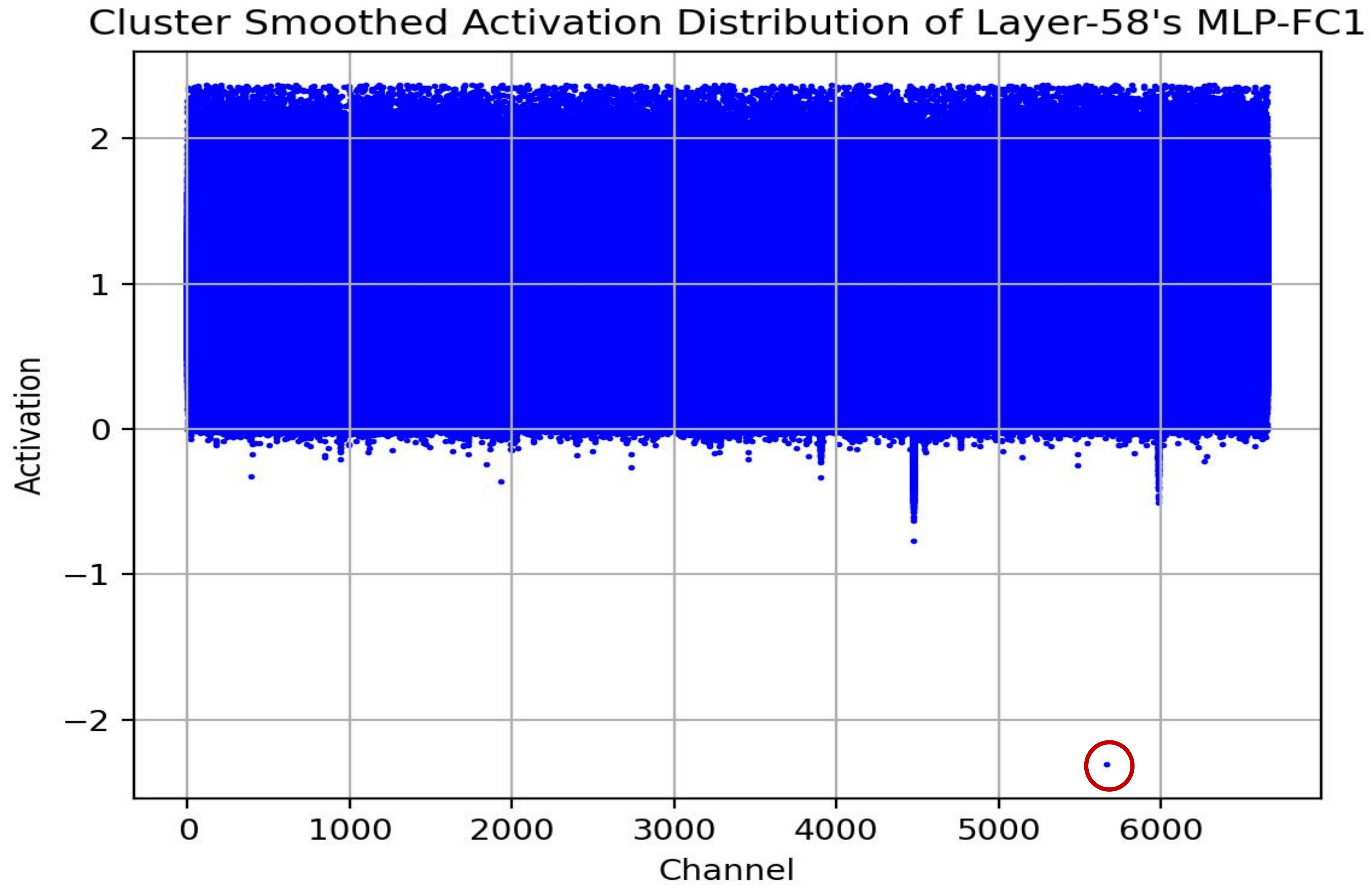
原数据分布	使用 ∂=0.7 smooth 后数据分布	使用分层搜参 smooth 后数据分布



$MSE_{Qerror} = 0.1275$



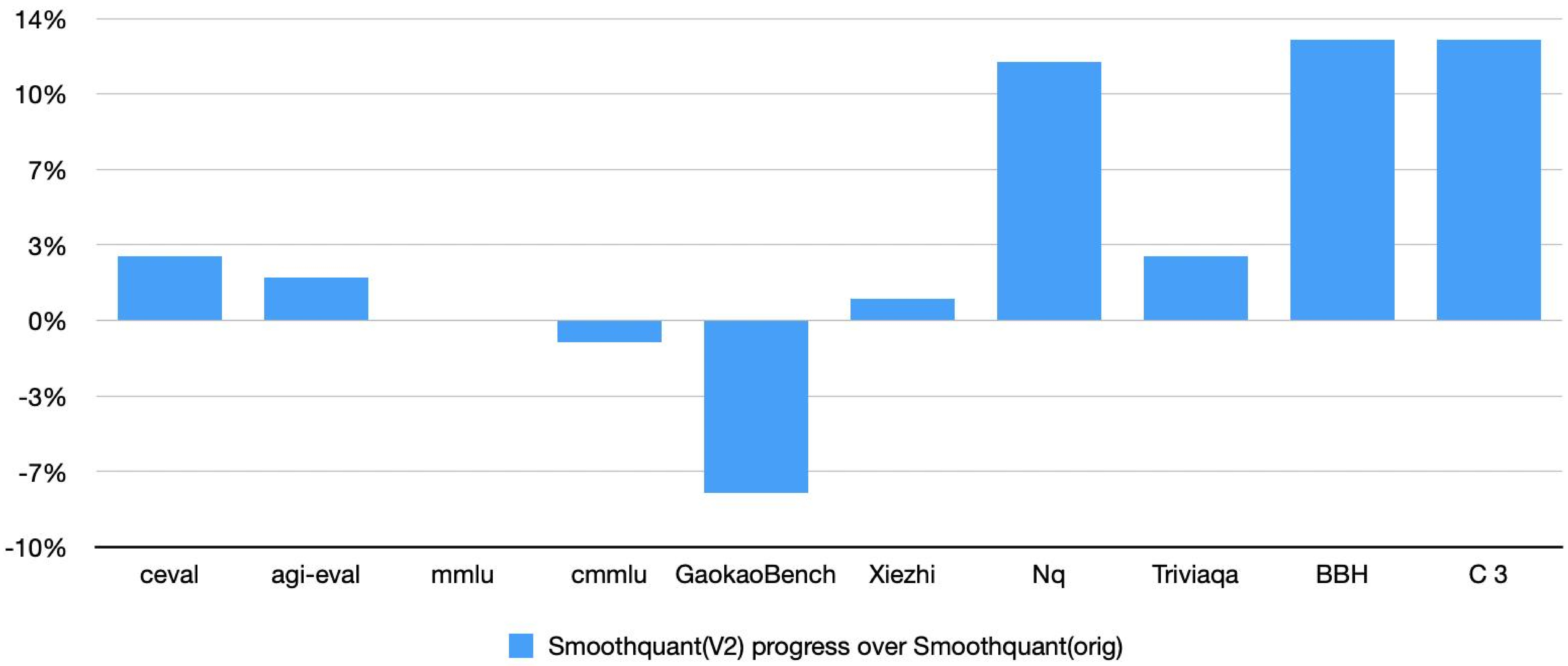
$MSE_{Qerror} = 0.0228$



$MSE_{Qerror} = \mathbf{0.00789}$

模型压缩 – W8A8量化

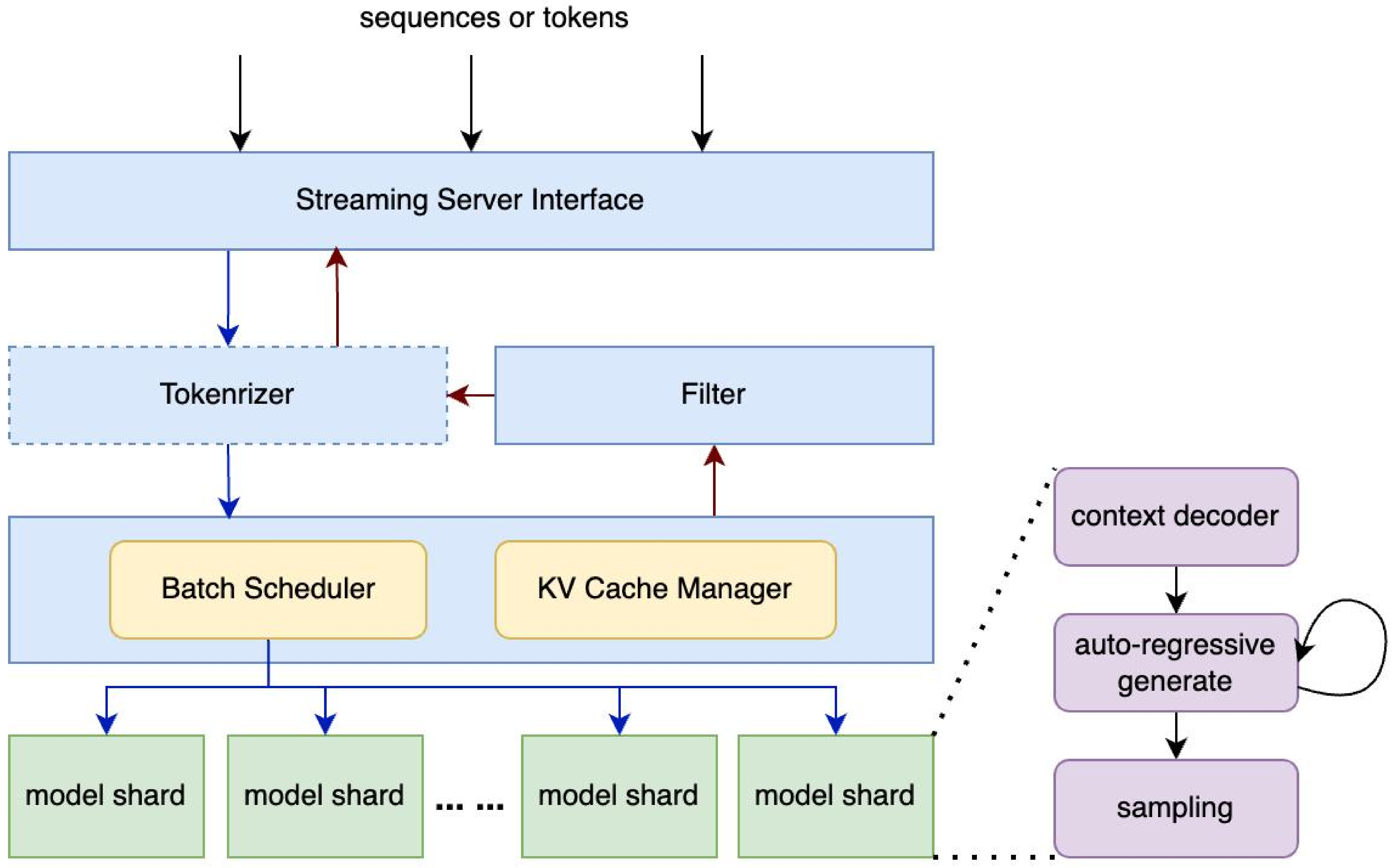
- **基于opencompass 的评测实验**
  - 在大部分评测集上，SQ（V2）相比于原算法有明显的精度提升；



框架与计算优化

- **框架架构及功能总览**

模块	功能
http server	stream / non-stream 访问模式
Tokenrizer	可插拔多款tokenrizer
Filter	对output tokens进行规则过滤
Batch Scheduler	实现continuous batch推理机制
KV Cache Manager	实现 paged attention等cache 管理机制
Model shards	多卡分布式推理

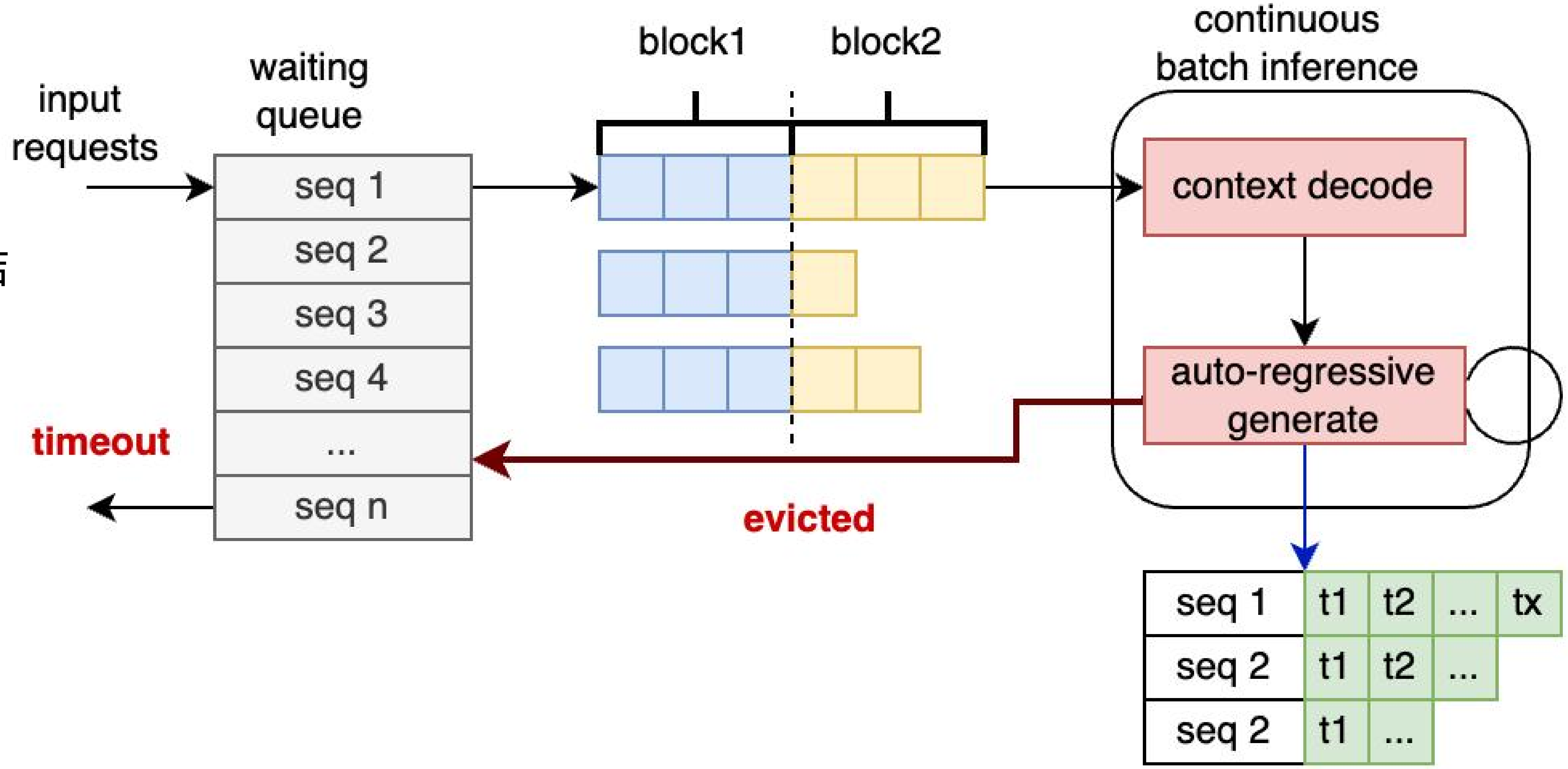


框架与计算优化

- **服务性能观测指标：**
  - First-token-latency；
  - Avg-token-package-cost；
  - Avg-tokens-per-second；
  - … …

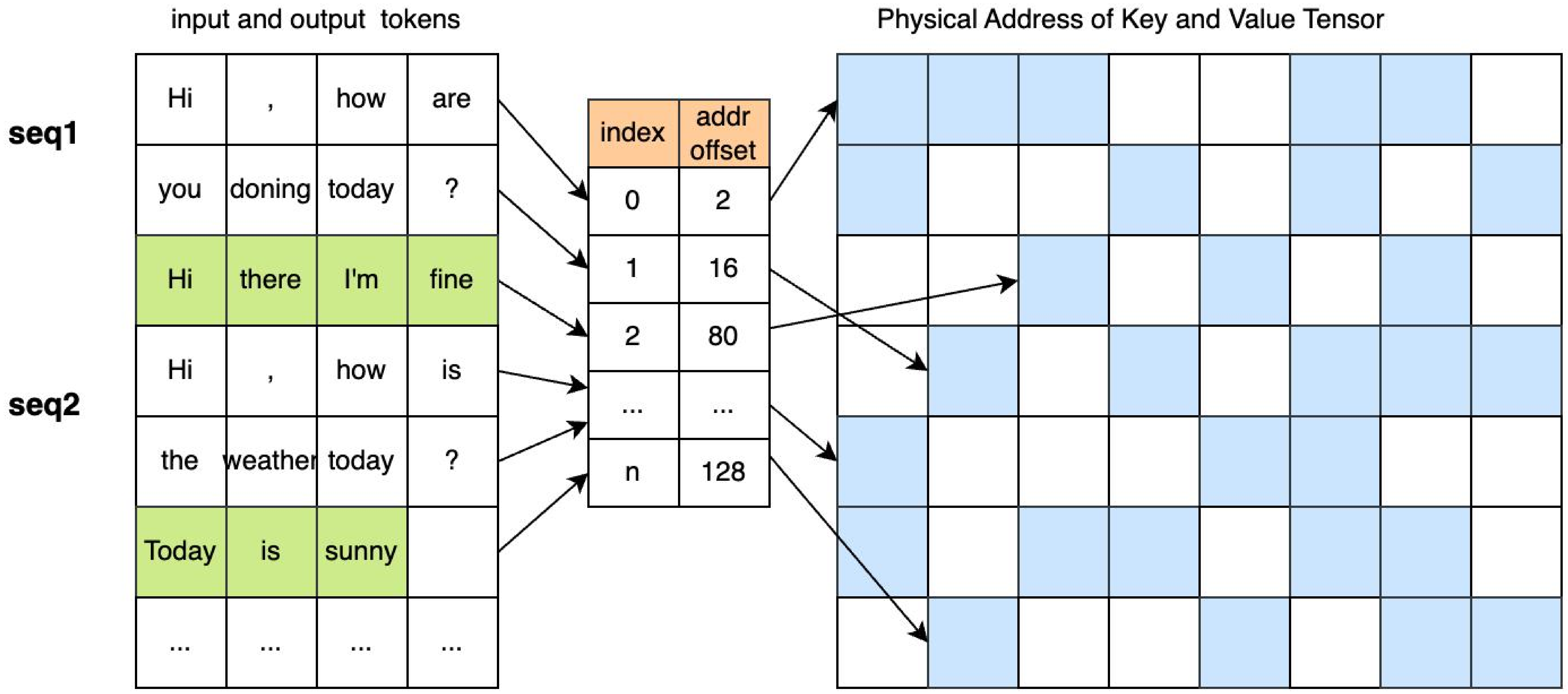
- **Continuous Batch Inference策略：**
  - Decoding阶段可被周期性打断，清理已结束请求，穿插执行新请求的prefill；
  - 当batch队列有空位时，waiting请求直接加入；
  - 显存不足时，按优先级回退进行中的请求，并回收cache，下轮加入重新计算；

- **改良策略：**
- decoding batchsize 动态自适应；
- decoding 中断interval 动态自适应；
- input batch 在一定范围内更加总长度自动做循序调整；



框架与计算优化

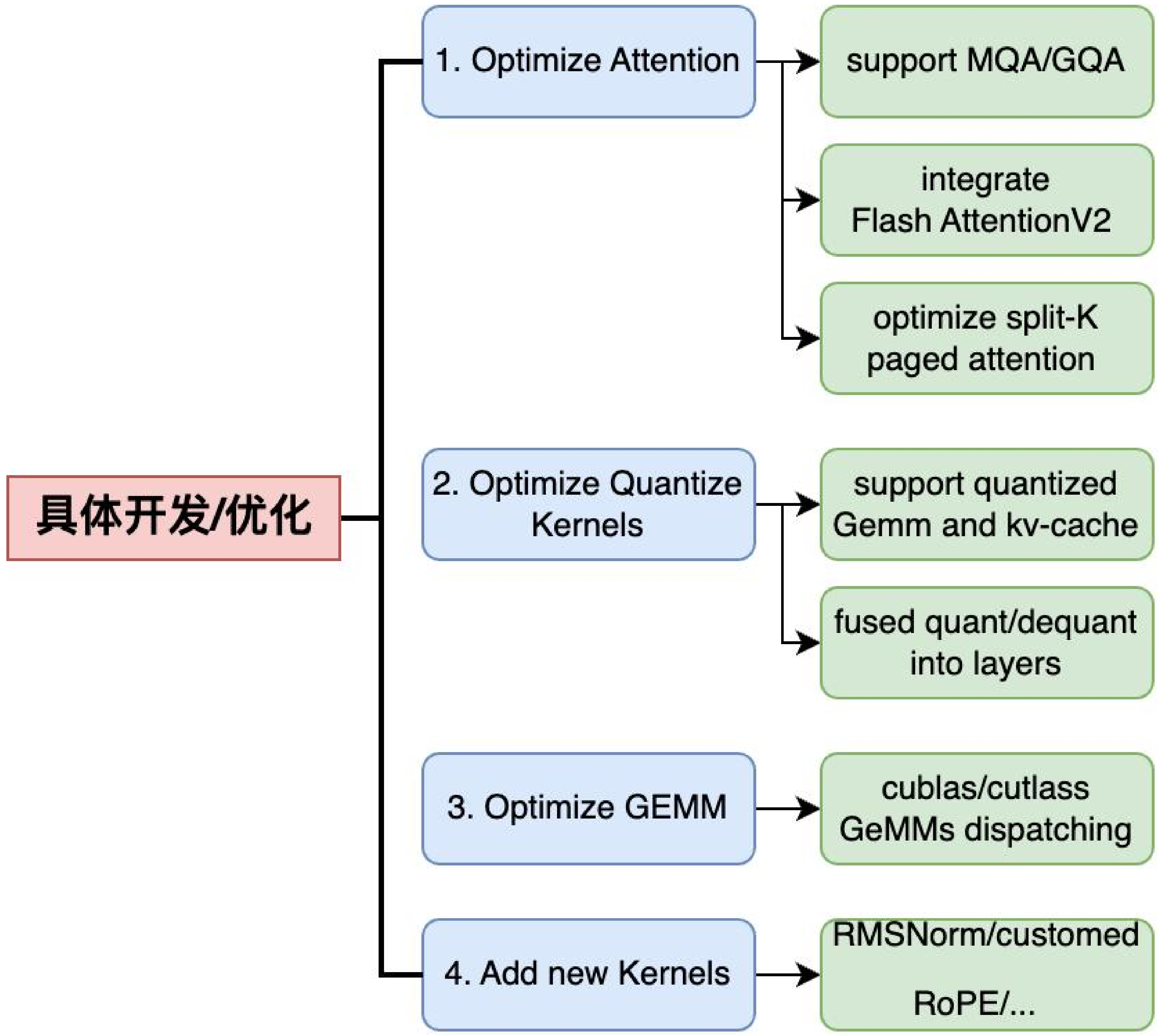
- paged attention 机制与实现
  - 将原来连续的k/v cache 改成离散分块存储
  - 根据decoding实际需求按需分配block，降低显存浪费

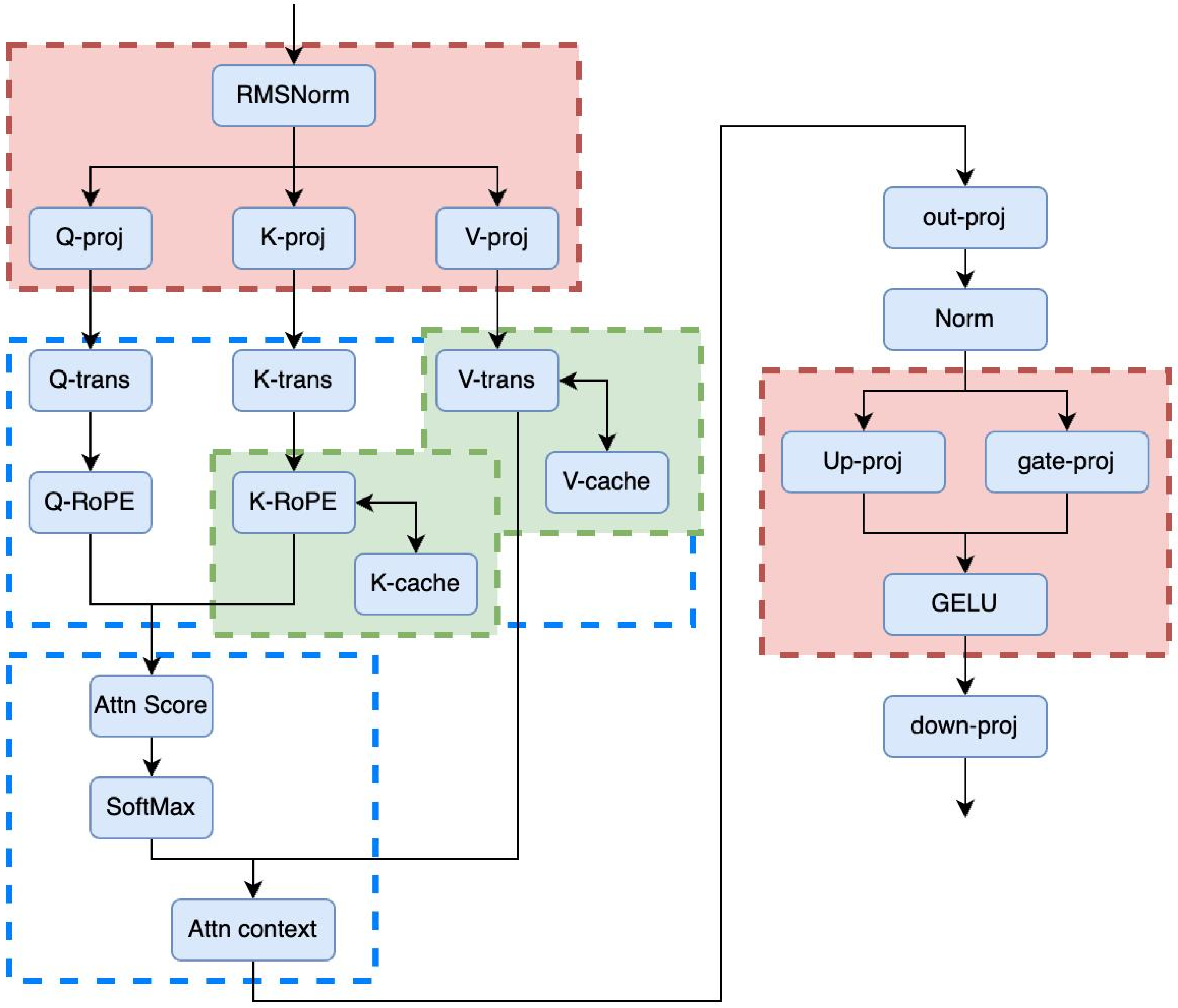


小红书

框架与计算优化

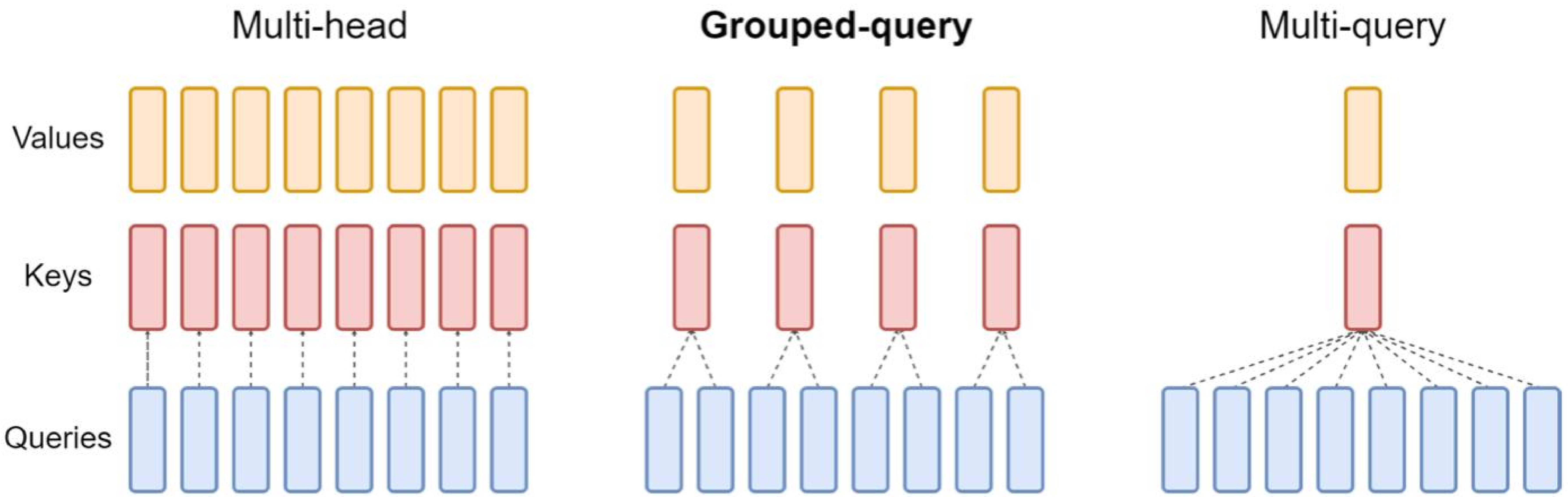
- 优化 generate decoding 环节 attention 实现





框架与计算优化

- 新增实现 MQA/GQA



Regular multi-head self-attention with *h* query, key, and value heads

Grouped-query attention shares key and value heads among multiple query heads

Multi-query attention shares key and value heads among *all* query heads

框架与计算优化

- 优化 generate decoding 环节 attention 实现
  - Tiling 优化
  - KV Cache排布优化
- 伪码

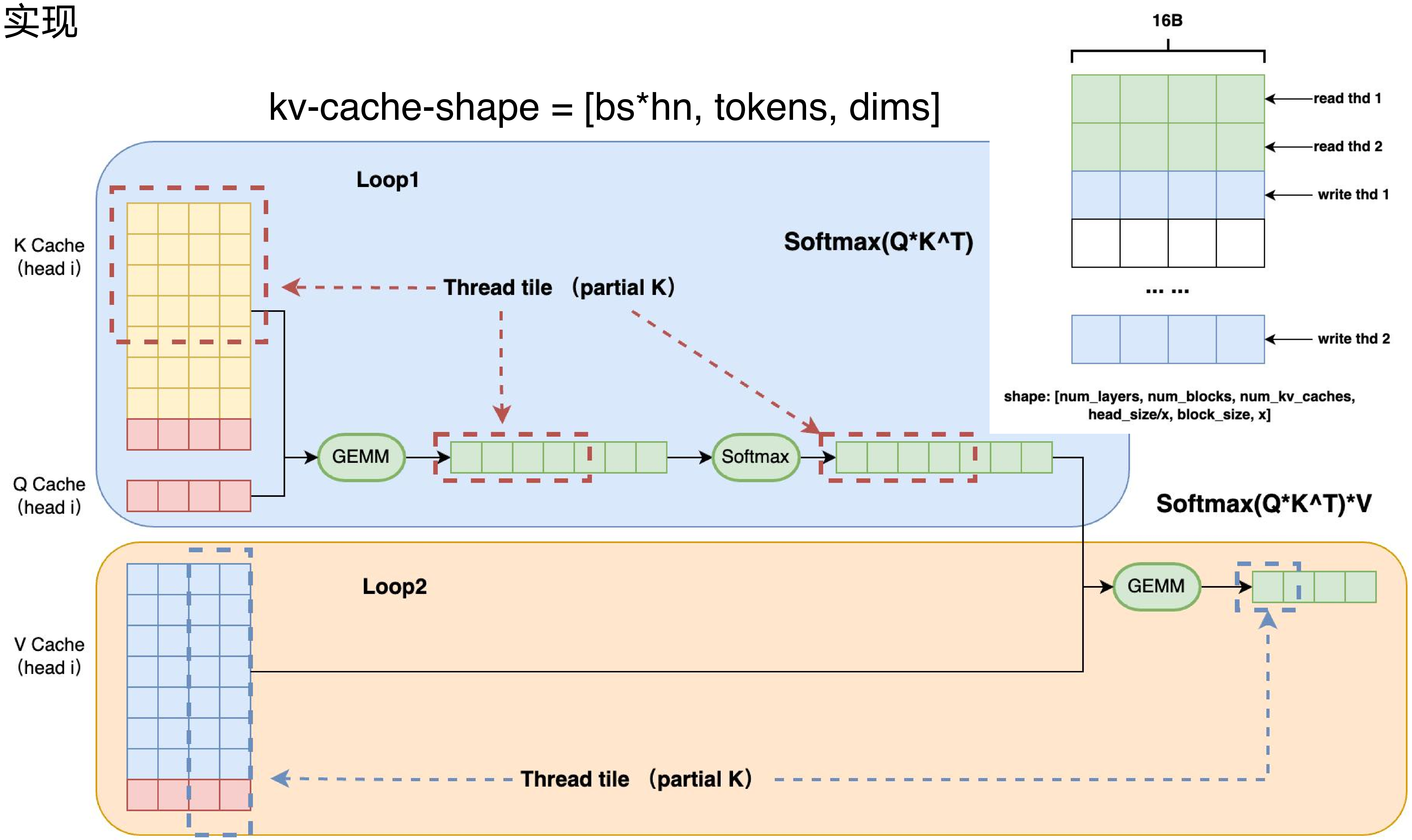
Step1: Tiling Key&Value around [bs*hn]

Step2: Tiling Key around [*,tokens,*]

Step3: Tiling Value around [*,*,dims]

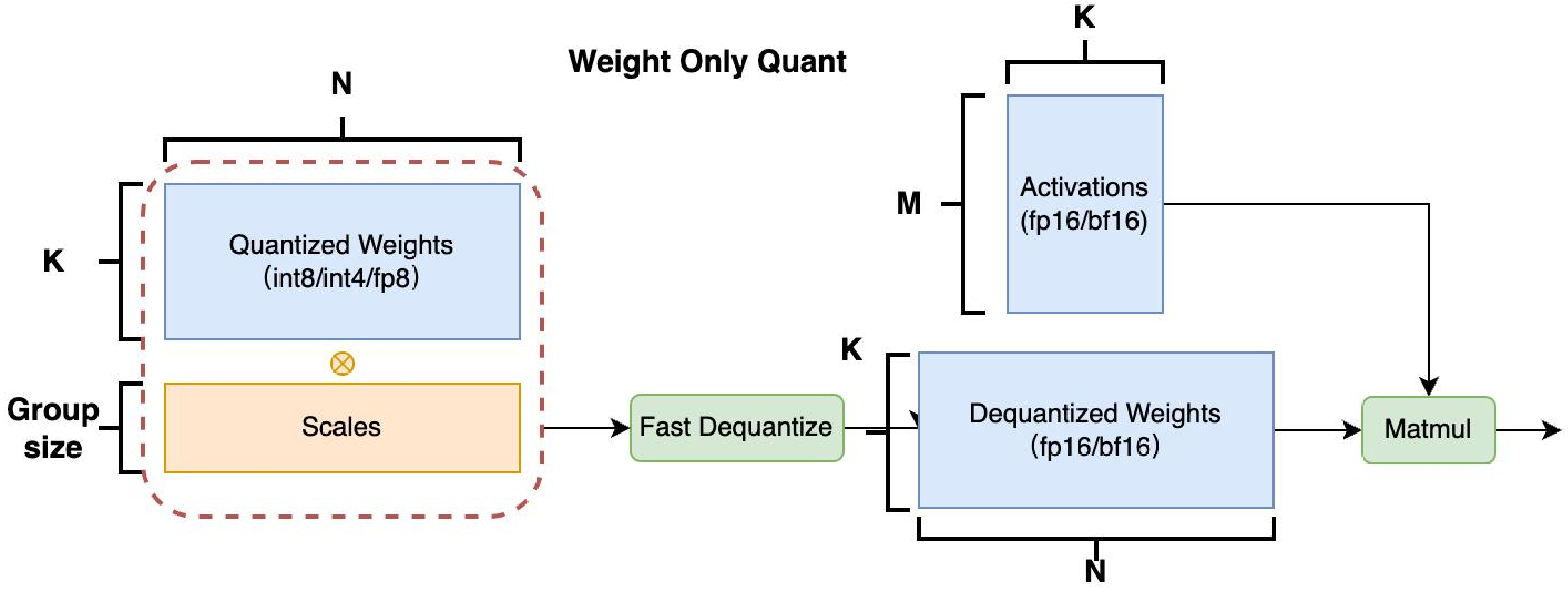
Step4: Get P=softmax(Q*K^T) done in Loop1

Step5: Get O=P*V done in Loop2

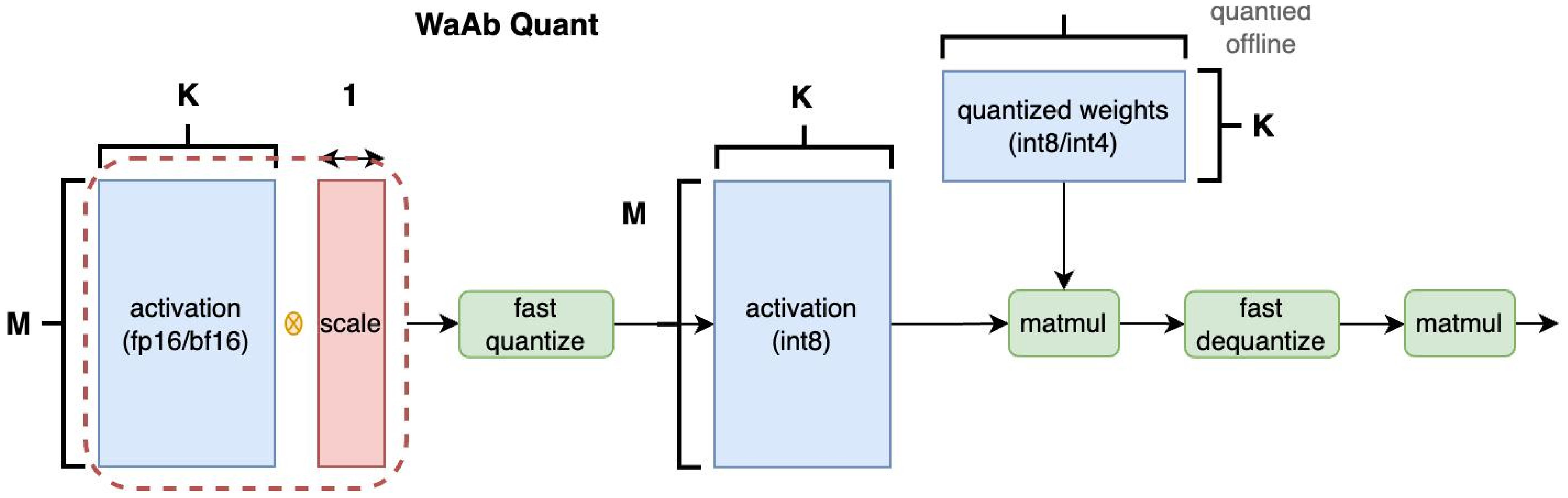


框架与计算优化

- 优化 generate decoding 环节 attention 实现
  - **Weigh-only-Quant:**
    - Per-channel
    - Per-group



  - **WaAb Quant:**
    - Weight： Per-channel
    - Activation: Per-tensor

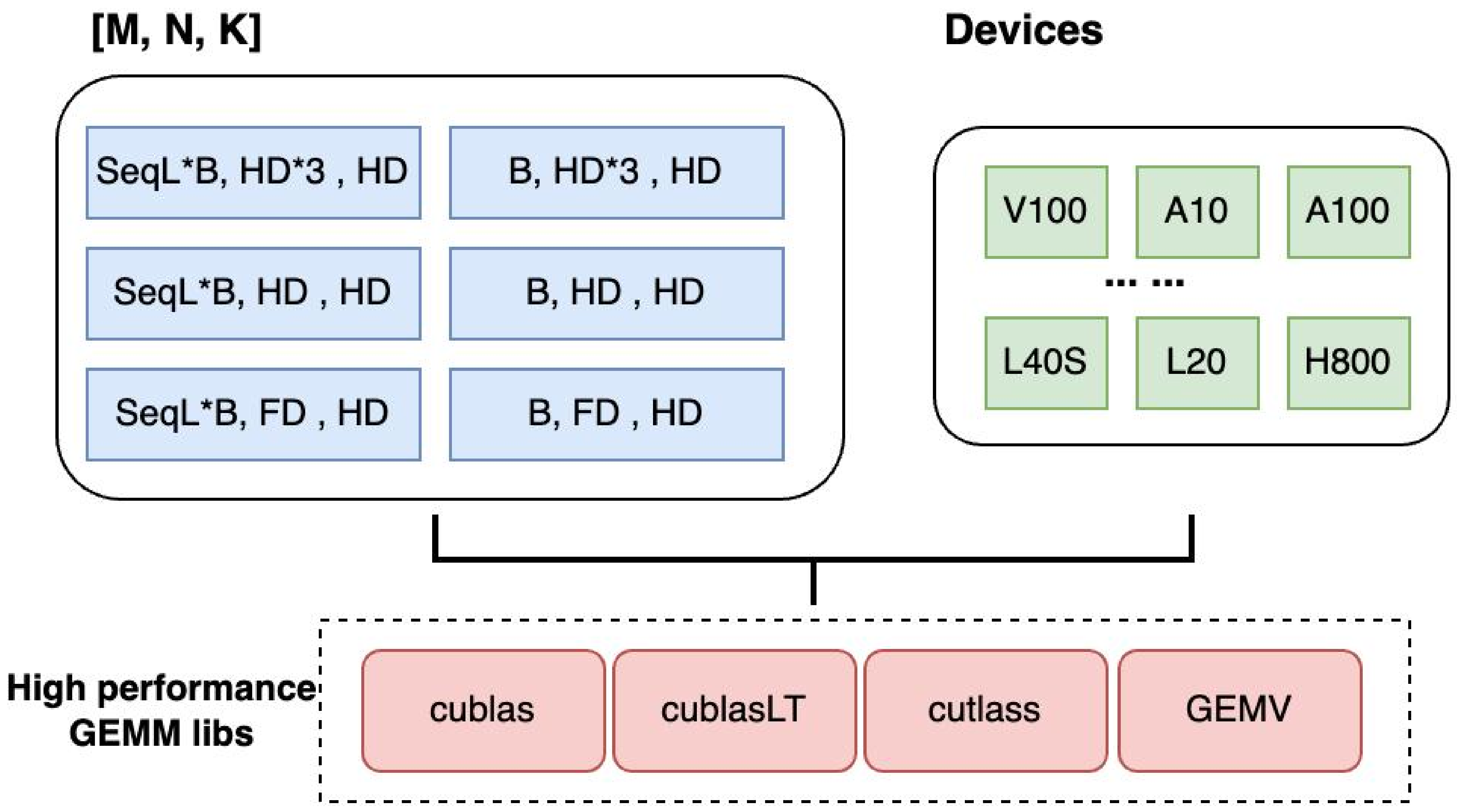


框架与计算优化

- GEMM优化
  - 针对不同 Device & TensorShape， 搜索最高效的GEMM实现

phase	op	GEMM-M	GEMM-N	GEMM-K
Prefill	QKV	SeqLen*B	HD * 3	HD
	Out-proj	SeqLen*B	HD	HD
	FFN1	SeqLen*B	FD	HD
	FFN2	SeqLen*B	HD	FD
Decode	QKV	B	HD * 3	HD
	Out-proj	B	HD	HD
	FFN1	B	FD	HD
	FFN2	B	HD	FD

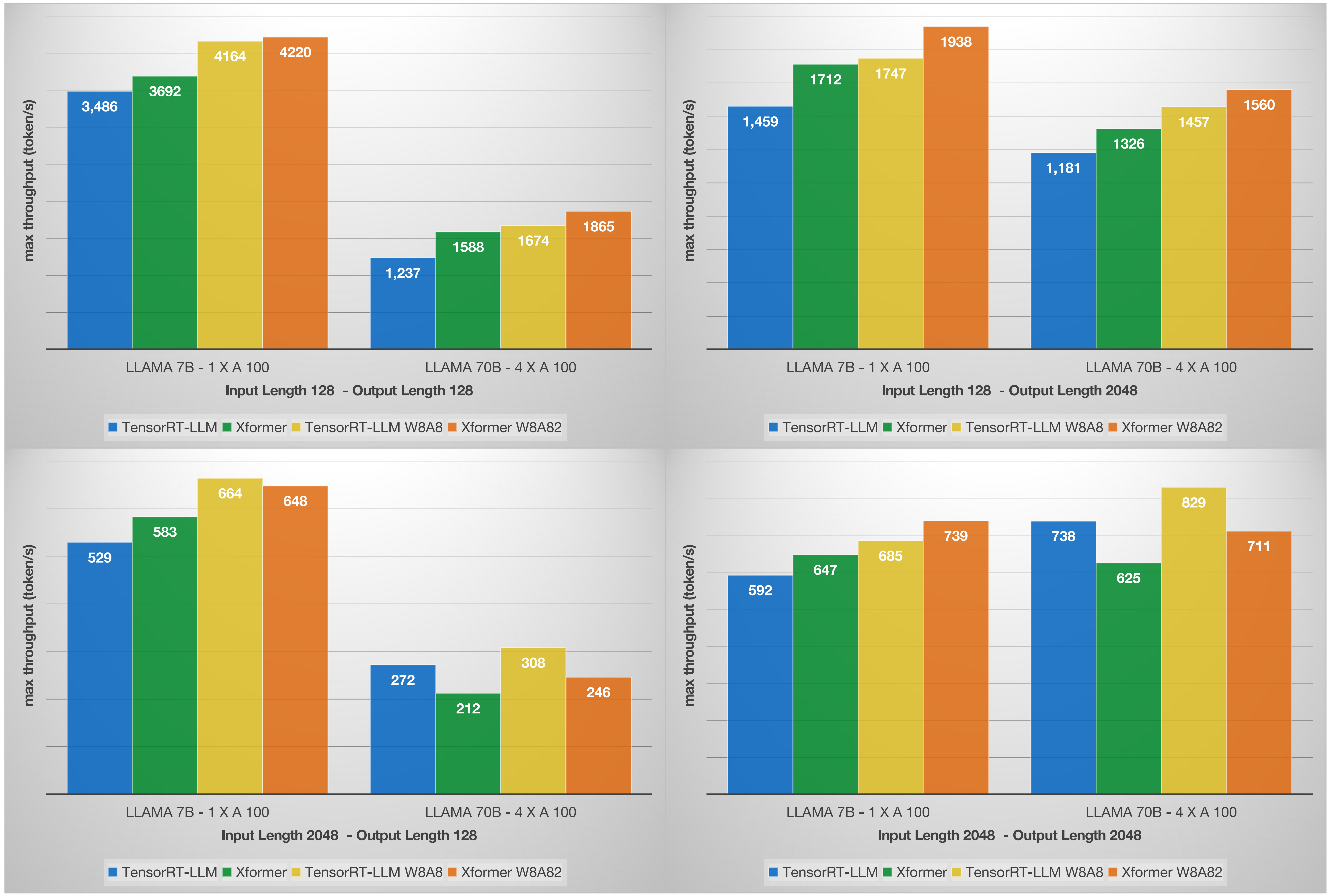
llama模型中所有的gemm input tensor shape



针对不同的Gemm shape 和 不同的device,
找到不同的最佳实现

框架与计算优化

- Compare with TensorRT-LLM
  - 测试硬件：A100 （80G）
  - 测试对象：
  - TensorRT-LLM
  - Xformer（小红书自研框架）



展望与挑战

- 更新的模型架构
- 更大的模型参数量
- 更有效的模型压缩方案
- 更长的输入/输出 token 长度
- 更多的应用场景：对话、推荐、安全等
- 多模态大模型
- 适配不同异构加速计算平台

欢迎关注
小红书技术 REDtech

THANKS

软件正在重新定义世界

Software Is Redefining The World