超大规模 MoE 与长上下文推理的性能秘籍

——从计算稀疏化到调度流水的全链路实战

徐善若

2025 年 5 月 22 日

目录

1 写在前面

本篇定位

本文不是 Dry Reference，而是面向工程实践者的「长文博客」：

- 先交代背景和基本概念，避免公式堆砌无上下文。
- 再穿插实验数据、图示与小结，让抽象概念落到可感知的度量。
- 重点选取我们在对话中多次讨论且对性能影响巨大的技术：
  稀疏激活、MLA/GQA、FP8 Tensor Core、Prefill/Decode 分离、
  CUDA Graph 捕获、多流调度、All-to-All 负载均衡、长上下文 NSA……

2 术语表 & 缩写速查

缩写	含义（博客上下文）
MoE	Mixture-of-Experts，参数级稀疏化
EP / TP / DP	Expert / Tensor / Data Parallelism（三维并行）
P 阶段	Prefill：一次吃完整段 prompt，$\mathcal{O}(L^2)$ 注意力
D 阶段	Decode：自回归生成，1 token 一循环
MLA	Multi-token Linear Attention：把 GEMV → GEMM
MQA/GQA	Multi-/Group-Query Attention，共享 K/V 减显存
NSA	Native Sparse Attention，长上下文稀疏化框架
MFU	Model FLOPs Utilisation=$\frac{\text{实际FLOPs}}{\text{峰值FLOPs}}$

3 MoE：让 FFN 参数只算 5%

3.1 路由器与 Top-k 激活

对任一 token 表示 $\boldsymbol{x} \in \mathbb{R}^H$，路由器给出

$$\boldsymbol{p} = \mathrm{softmax}(W_{\text{router}}\boldsymbol{x}) \in \mathbb{R}^{N_{\text{exp}}}, \qquad \text{Top-}k(\boldsymbol{p}) \to \{\text{Expert}_i\}$$

只前向 k 个专家，FLOPs 与显存下降 $\frac{k}{N_{\text{exp}}} \approx 5\%$.

3.2 Expert Parallel & 冗余专家

(1) 将 N_{exp} 个 FFN 切到各 GPU（EP）。

(2) 统计最热专家的 token 分配 K_i，若 $\max_i K_i / \operatorname{mean} K > 2$，复制权重到空闲 GPU，Router 哈希分流。

示例：Llama-2-MoE-64 在 256 GPU 上复制前 5 % 专家，All-to-All 尾延迟从 60 µs 降至 28 µs。

4 Attention 侧：GEMV → GEMM 的三板斧

4.1 MLA：批量拼接 ⇒ 大矩阵

Prefill 将 $B\times L$ 个 Query 合并：

$$Q\in\mathbb{R}^{(BL)\times d},\quad K^{\top}\in\mathbb{R}^{d\times L},\ \text{GEMM}\ (BL,d,d,L)$$

MFU ↑；Decode 靠 in-flight batching 做同理扩张。

4.2 MQA/GQA：共享 K/V

如果让所有 head 共用一份 K,V，显存 $\downarrow H$；计算转为 $HQ\times 1$ KV 的 GEMM，复杂度不变，带宽省。

4.3 长上下文稀疏化 NSA

将长度 $L\sim 10^5$ 的历史分三步：

$$\underbrace{\text{Compress}}_{O(L)}\ \rightarrow\ \underbrace{\text{Top-}k\ \text{Select}}_{O(L)}\ \rightarrow\ \underbrace{\text{Sliding Window}}_{O(wL)}$$

整体近似 $O(L)$，再映射为 GQA 算子；示例代码见 Listing 1。

5 FP8 Tensor Core 与分段累加

5.1 格式与动态缩放

FP8-E4M3: 4 bit exp + 3 bit mantissa FP8-E5M2: 5/2 拆分，动态范围大。

$$\text{scale}=\max(|x|)/\left(2^{7}-1\right)$$

每 128 tile 累加一次 flush 到 FP32 寄存器，误差 $\le 2^{-14}\times 128\approx 10^{-2}$.

6 Prefill / Decode (P/D) 分离架构

6.1 KV-Cache 单向转递

Prefill-GPU 把 `KV` 拷到 CPU → RDMA → Decode-GPU，一次性写入对应 slot 后立即复用显存。**优点**：长 prompt 不再拖慢短对话；首 token 延迟下降 $\sim 40\%$。

7 CUDA Graph 捕获的三次进化

graph-1.0：局部 Attention 子图

graph-2.0：整层 Graph + padding 掩码

graph-3.0：双流交错，计算/通信重叠

最终 MFU $\approx 85\%$，单 H100 解码 ≥ 8 k tok/s。

8 Scheduler Overlap：CPU & GPU 齿轮咬合

核心思路

- 两个队列：`input_queue`（待前向）、`result_queue`（待采样）。
- CPU 线程把 batch 写入前者即刻返回；GPU Worker 消费后把 logits 推到后者。
- CPU 采样、排下轮 batch 与 GPU 前向并行执行。

9 All-to-All 通信提速

9.1 D-通路（小包）

GPU-direct RDMA，ib-gdr；每包 $\sim$ 7kB，RTT 3 µs。

9.2 P-通路（大包）

RDMA 到机箱 →NVLink 二次扇叶，400 G 口利用率 85 %.

10 Chunk-Prefill 与 DP 均衡

把最长 4 k prompt 切成 8 块 512 token：CPU-launch 次数 ×8，但经 CUDA Graph 合并后 Launch 0 开销；DP 负载长尾从 4× 降到 1.3×。

11 长上下文 Example： $L = 64k$

$$\text{Dense Attention}: \ L^2 d = 64k^2 d$$

$$\text{NSA} + \text{MQA}: \ Ld + Ld/H \approx 65kd/H$$

当 $H = 8$ 时，计算下降近 $8 \times \mathbf{10^3}$ 倍；实验：GPT-3-MoE-13B 加入 64 k 上下文，BLEU 下降 0.15。

12 代码片段与实测结果

```
# 简化版 NSA Top-k 选择
scores = (q @ k.T).softmax(-1)
topv, topi = torch.topk(scores, k=64, dim=-1)
ctx = (topv[..., None] * v[topi]).sum(-2)
```

图 1: Top-k 相关性选择示例

图 2: Scheduler Overlap 前后 MFU 对比

13 结语：三句 Take-away

1. **参数稀疏 (MoE) + 上下文稀疏 (NSA)** 是扩张 1T-Param、1M-Token 的双引擎。
2. 计算与通信的 ** 重叠 ** 和 CPU/GPU 的 ** 解耦 **，比单纯算子优化能带来更大实际吞吐收益。
3. 模型-硬件协同（FP8 Tensor Core、CUDA Graph）+ 系统级流水（P/D 分离、双队列）才能真正触碰“理论极限”。

参考文献

[1] Shazeer N. *et al.* Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. arXiv 2020.

[2] NVIDIA. Hopper Architecture Whitepaper, 2022.

[3] Liu Z. *et al.* Native Sparse Attention for Long-Context LLMs. arXiv 2024.

[4] Deng B. *et al.* vLLM: Easy, Fast, and Cheap LLM Serving. MLSys 2024.