基于 CUTE DSL 的通信计算重叠算子实践

GEMM + AR 经验分享

周沛源

NVIDIA 高级解决方案架构师

2026/01/17

蚂蚁开源 ANT OPEN SOURCE

SGL

周沛源

NVIDIA 高级解决方案架构师

专注于大模型训推，深耕算子，框架及通信。

蚂蚁开源 ANT OPEN SOURCE

SGL

CONTENT
目录

不同方案比较

Intra SM vs Inter SM

硬件粒度	缺点	实现方式	备注
Inter-SM	部分Tensor core 始终空置	2 kernel + green context	利用 fine-grain GTX / green context 让 2 kernel 并行调度，相比fused kernel, 实现更简单. 灵活配对通信-计算 kernel，仅需要通信+计算个 kernel. 但并不能保证kernel同时启动.
		2 kernel + priority stream (不推荐)	通过高/低优先级 stream 抢占调度，不稳定.
		1 kernel, 按 CTA 划分	同一 kernel 内部分SM 做通信，部分做计算, 比如ParallelKitten, Comet。 相比green context实现，实现复杂，并且需要通信*计算个kernel, 但能保证调度.
Intra-SM	通信占用过多资源(reg…), 影响计算效率	1 kernel, 按 Warp划分	同一kernel内相同SM中部分 warp 做通信，部分做计算
		2 kernel 共享同一 SM	两个 kernel 共用 reg/smem，调度和竞争问题多

不同方案比较

通信粒度

按照通信粒度，一般分为tile和chunk

粒度	描述	适合场景示例
tile	tile为粒度进行通信，通常是一个tile的计算结果	GEMM, GroupGEMM
chunk	chunk为粒度通信，粒度相比tile更大,一般使用2 kernel，一个负责计算，一个负责通信. 方便使用CE节省SM资源	All-Gather + GEMM, Ring Attention

不同方案比较

GEMM+AR

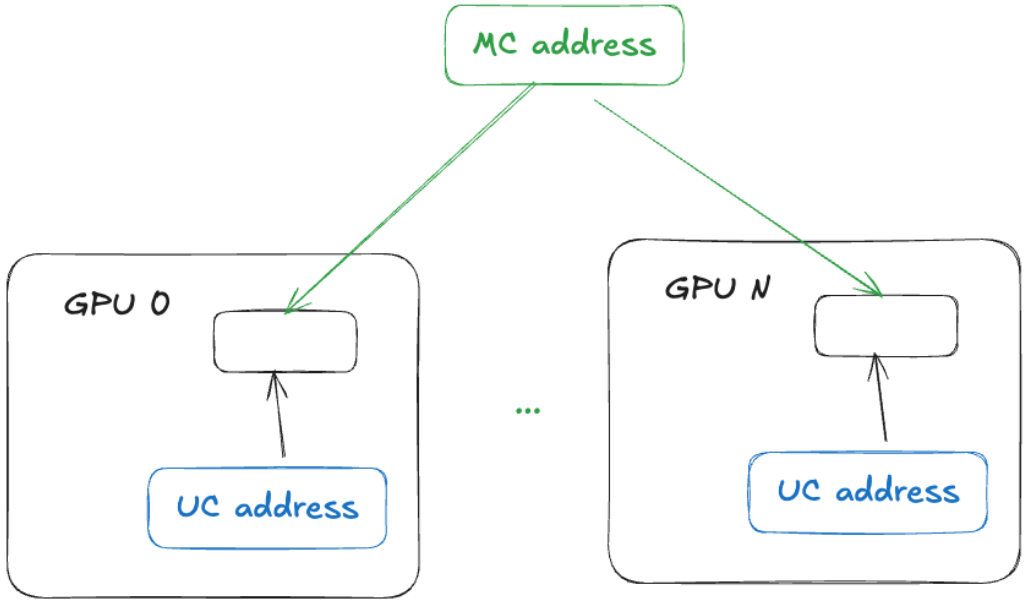
GEMM + AR适合fused kernel的tile粒度方式，尤其是SM100，理由如下：

1. GEMM是tensor core heavy的计算，如果使用Inter SM方式，会影响计算效率，增加计算时长.
2. CTA tile天然匹配GEMM计算.
3. 如果使用2 kernel，一旦计算 kernel 先被调度就会占满 SM，导致通信 kernel 需要等计算结束后才能被调度，**没有实现真正的重叠.**
4. 切分GEMM可能会影响计算效率，并引入额外的启动开销.
5. SM100引入了TMEM，寄存器资源充足，对计算影响较小.

Multicast

介绍

Multimem address是一个虚拟地址抽象，用于指向分布在**多个设备**上的多个不同内存位置.
仅multimem.* 系列操作对 multimem address 有效，在其他任何内存操作中对multimem address的访问行为为未定义.
配合 NVSwitch 的内置计算功能，可以实现在网计算.



相关指令	在GEMM +AR中的作用
multimem.ld_reduce	Reduce不同rank上的数据，返回给调用者
multimem.st	广播reduce结果到所有rank
multimem.red	对multimem flag执行原子加，用于在多rank间同步

Multicast

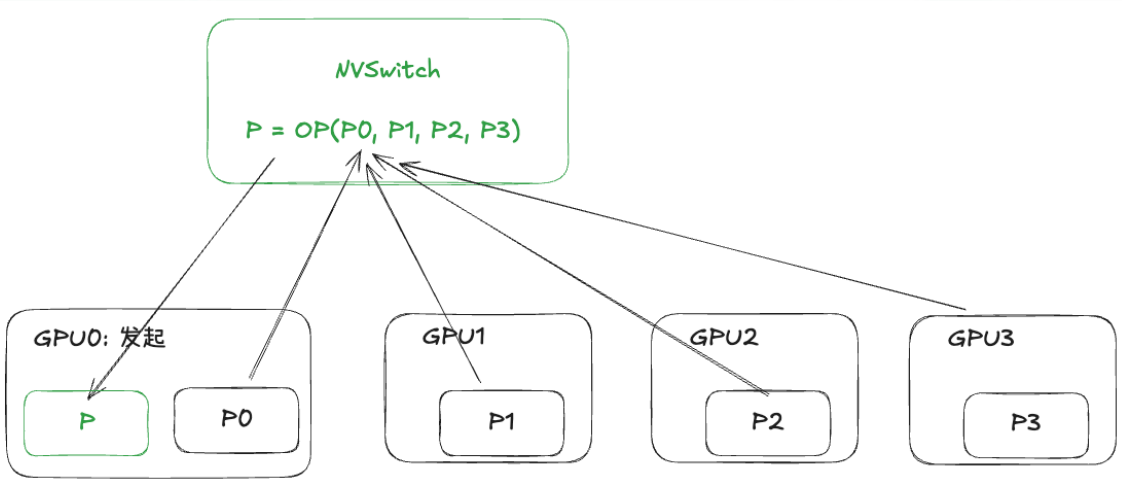
multimem.ld_reduce

multimem.ld_reduce **执行以下操作：**

1. Rank i发起操作, NVSwitch从所有参与rank的对等地址读取数据.
2. **在 NVSwitch 内部执行reduce操作.**
3. 由 NVSwitch 将结果发送回rank i

PTX: multimem.ld_reduce{.ldsem}{.scope}{.ss}.op.type d, [a]

- d **在发起者的寄存器上**.
- **地址操作数** a **必须是**multimem address.
- scope = { .cta, .cluster, .gpu, .sys }.
- ldsem = { .relaxed, .acquire }, **不同的** memory order **对性能影响较大**.



Multicast

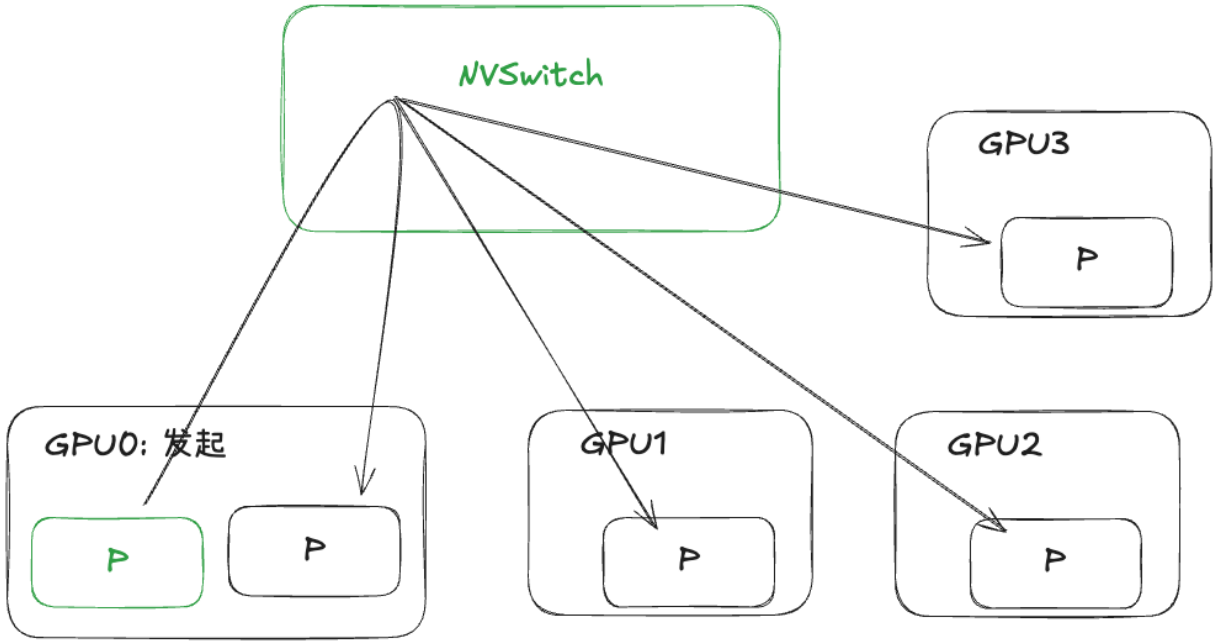
multimem.st

multimem.st **执行以下操作：**

- Rank i发起操作，将数据发送给NVSwitch.
- NVSwitch将该数据多播给所有参与rank.

PTX: multimem.st{.stsem}{.scope}{.ss}.type [a], b;

- **地址操作数** a **必须是**multimem address.
- scope = { .cta, .cluster, .gpu, .sys }.
- .stsem = { .relaxed, .release }，**不同的** memory order**对性能影响较大**.



Multicast

multimem.red

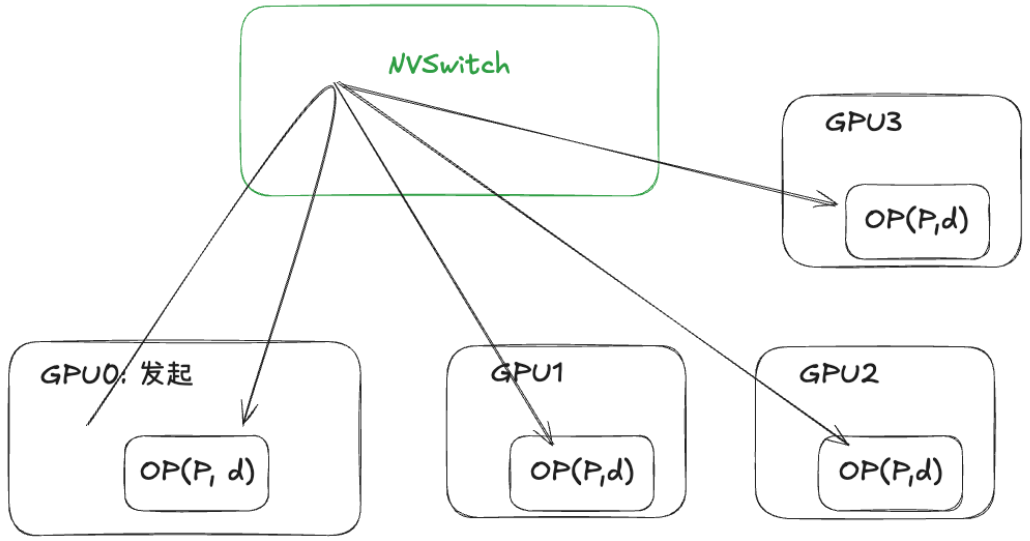
multimem.red **执行以下操作：**

1. Rank i发起操作, 数据从第 i 个 rank 发送到NVSwitch.
2. NVSwitch将数据多播给所有参与的 rank.
3. 在每个目标 rank **本地**执行reduce操作.

PTX: multimem.red{.redsem}{.scope}{.ss}.op.type [a], b;

- **地址操作数 a 必须是**multimem address.
- scope = { .cta, .cluster, .gpu, .sys }.
- .redsem = { .relaxed, .release }.

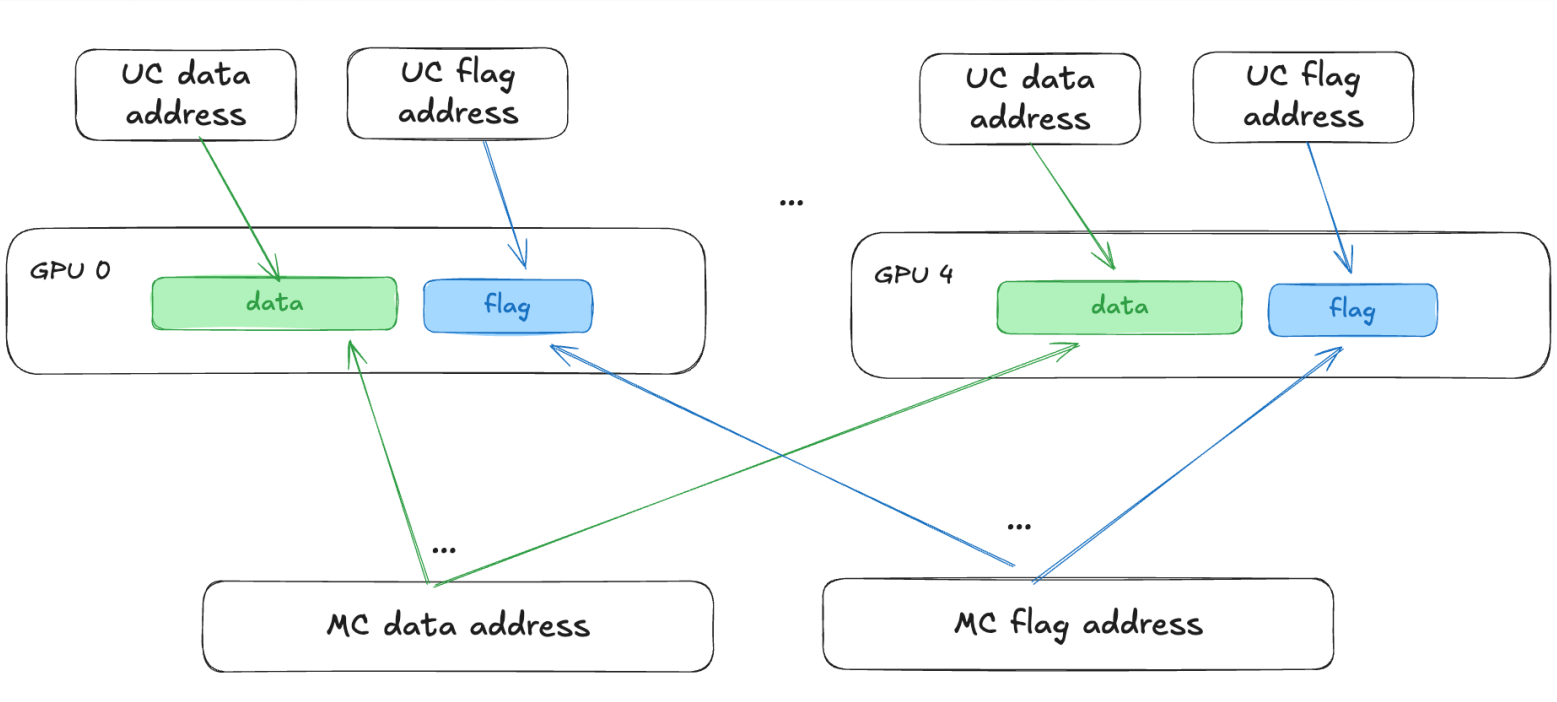
不同的 memory order **对性能影响较大**.



GEMM-AR fused kernel

初始化

使用 nvshmem4py 或者 torch symmetric memory, 生成multimem address.

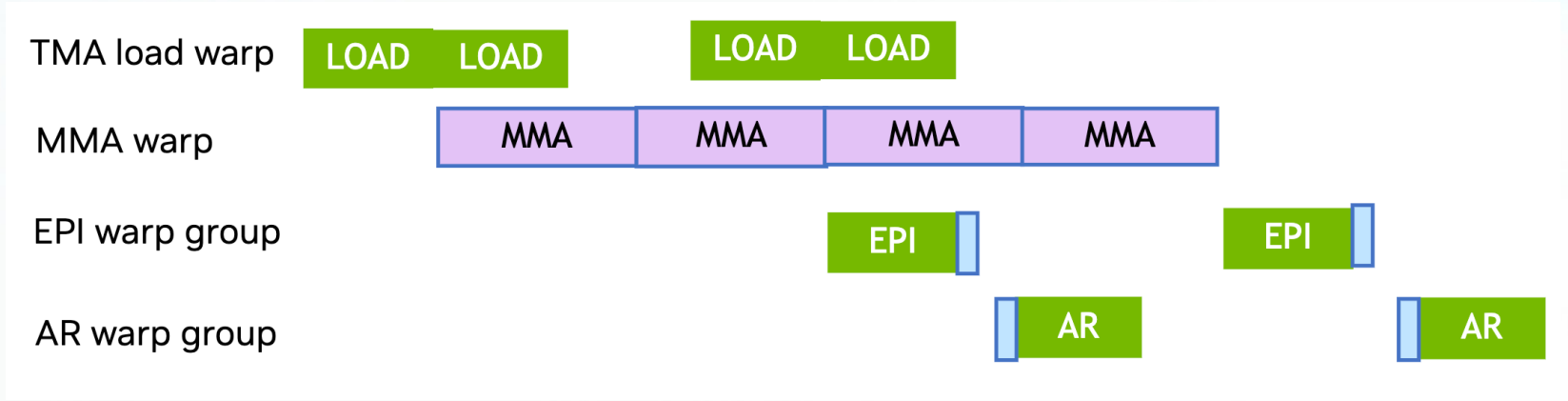


UC 和 MC 指向同一片物理内存，底层通过 memory map API 映射。

GEMM-AR fused kernel

workflow

Idea: 增加一个warp group用于all reduce通信.

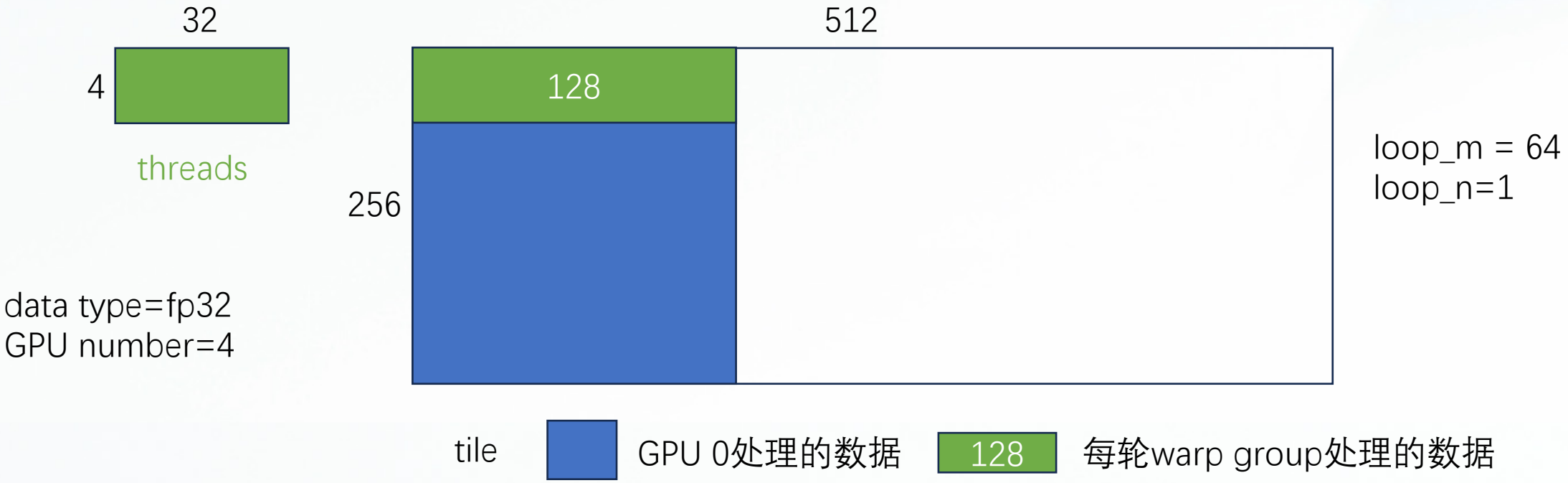


指同步，EPI warp会对MC地址指向的flag执行原子+1. AR warp需要等flag == num_ranks.

GEMM-AR fused kernel

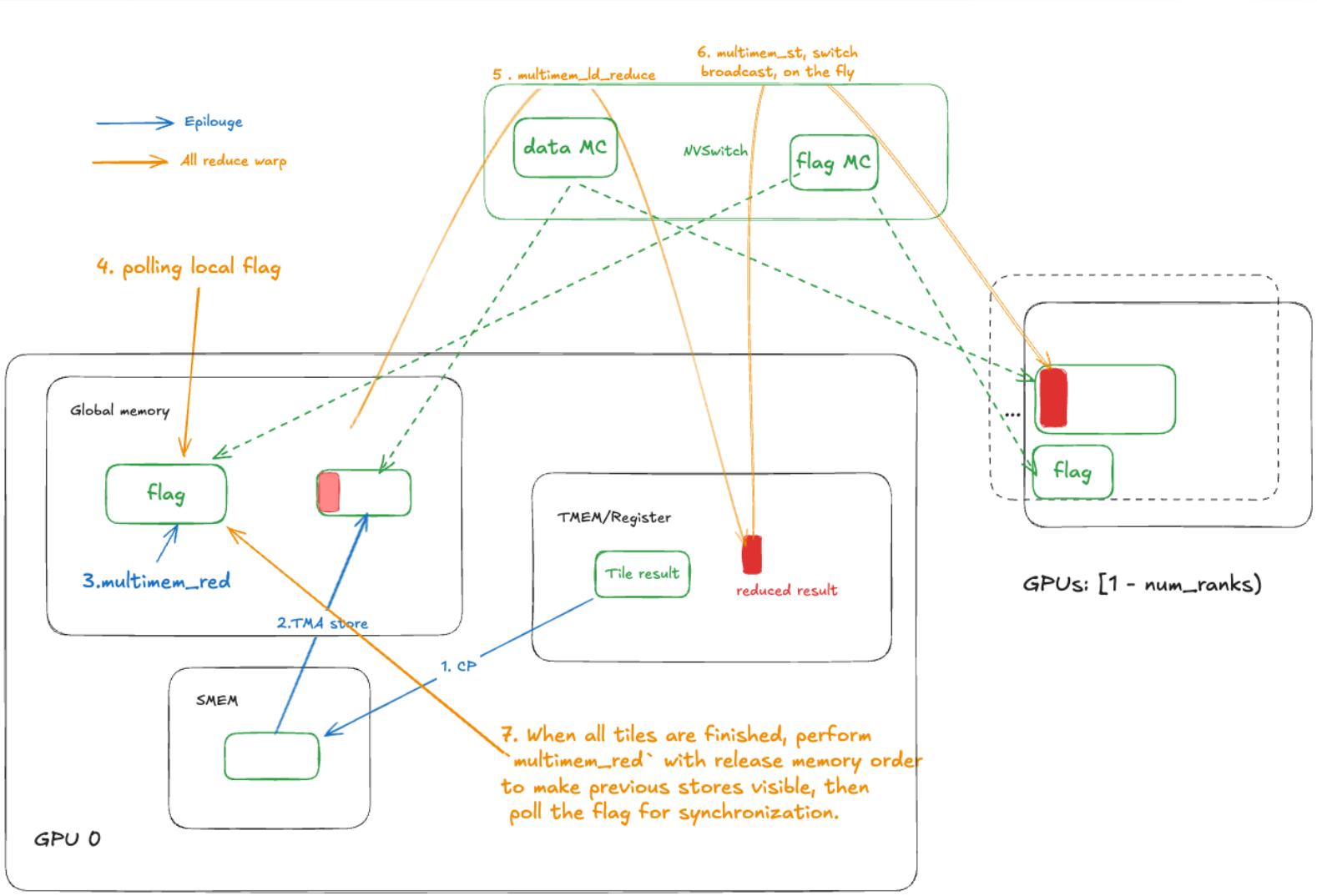
任务划分

1. 使用persistent kernel, 每个SM处理多个不同的tile. 不同GPU上的同一SM处理同一块tile.
2. 每个tile均分给所有GPU, 每个SM每次处理tile/N大小的数据.
3. 2D TV layout， 每个线程均分数据， 线程之间互相无依赖. 线程每条指令处理128 bits数据.



GEMM-AR fused kernel

workflow



1. Tile all reduce 前同步，确保计算结果写入对应 MC 地址.
2. 所有 tile 完成后同步，确保所有rank都已完成通信.
3. 每个线程处理自身数据，reduce 和 store 操作于同一数据块，粒度128bits。
4. spin lock机制等待flag.

GEMM-AR fused kernel

memory order

不同的 memory order 对性能影响很大，在保证正确的情况下使用最小限制的 memory order.
实践中，*ld_reduce* 和 *store* 都使用了 relaxed order， 原因如下:

multimem.ld_reduce.sys.relaxed.global.add
multimem.st.sys.relaxed.global

1. 不同线程并行发送 multimem 指令，相互之间无任何依赖。
2. 同一线程的 ld_reduce 和 store 操作的是同样的 register, 自身具有可见性。
3. 对 flag 的 red 操作采用 release order，保证前面所有的 store 都已经完成并可见。
4. 最后的red使用 sys scope 保证所有GPU的可见性。

GEMM-AR fused kernel

memory order

不同的 memory order 对性能影响很大，在保证正确的情况下使用最小限制的 memory order.
实践中，*ld_reduce* 和 *store* 都使用了 relaxed order， 原因如下:

multimem.ld_reduce.sys.relaxed.global.add
multimem.st.sys.relaxed.global

1. 不同线程并行发送 multimem 指令，相互之间没有顺序要求.
2. 同一线程的 ld_reduce 和 store 操作的是同样的 register, 自身具有可见性.
3. 对 flag 的 red 操作采用 release order，保证前面所有的 store 都已经完成并可见。
4. 最后的red使用 sys scope 保证所有GPU的可见性。

GEMM-AR fused kernel

performance

根据实验结果，大部分情况下 **60%** 左右的通信时间都会被隐藏掉.
SGLang PR bench_serving E2E throughput **+6.42%**:

model	TP	in/out	latency/ms	throuput	sglang version
Qwen2.5-72B-Instruct	4	4096/1	5195.9	21947	origin
Qwen2.5-72B-Instruct	4	4096/1	4902.7	23356	origin+overlap

model	TP	in/out	latency/ms	throuput	sglang version
Qwen2.5-72B-Instruct	4	4096/128	52.8	685	origin
Qwen2.5-72B-Instruct	4	4096/126	50.5	729	origin+overlap

蚂蚁开源
ANT OPEN SOURCE

SGL

谢谢大家，欢迎指正！