GLake：大模型时代显存+传输管理与优化

蚂蚁集团-基础智能-AI Infra

赵军平

2024.5

提纲

蚂蚁集团 ANT GROUP

- 背景与技术挑战
  - 显存墙、传输墙
- GLake介绍：显存与传输优化
  - 训练场景
  - 推理场景
  - 其他：混部、serverless
- 总结与展望
  - 开源地址

自我介绍

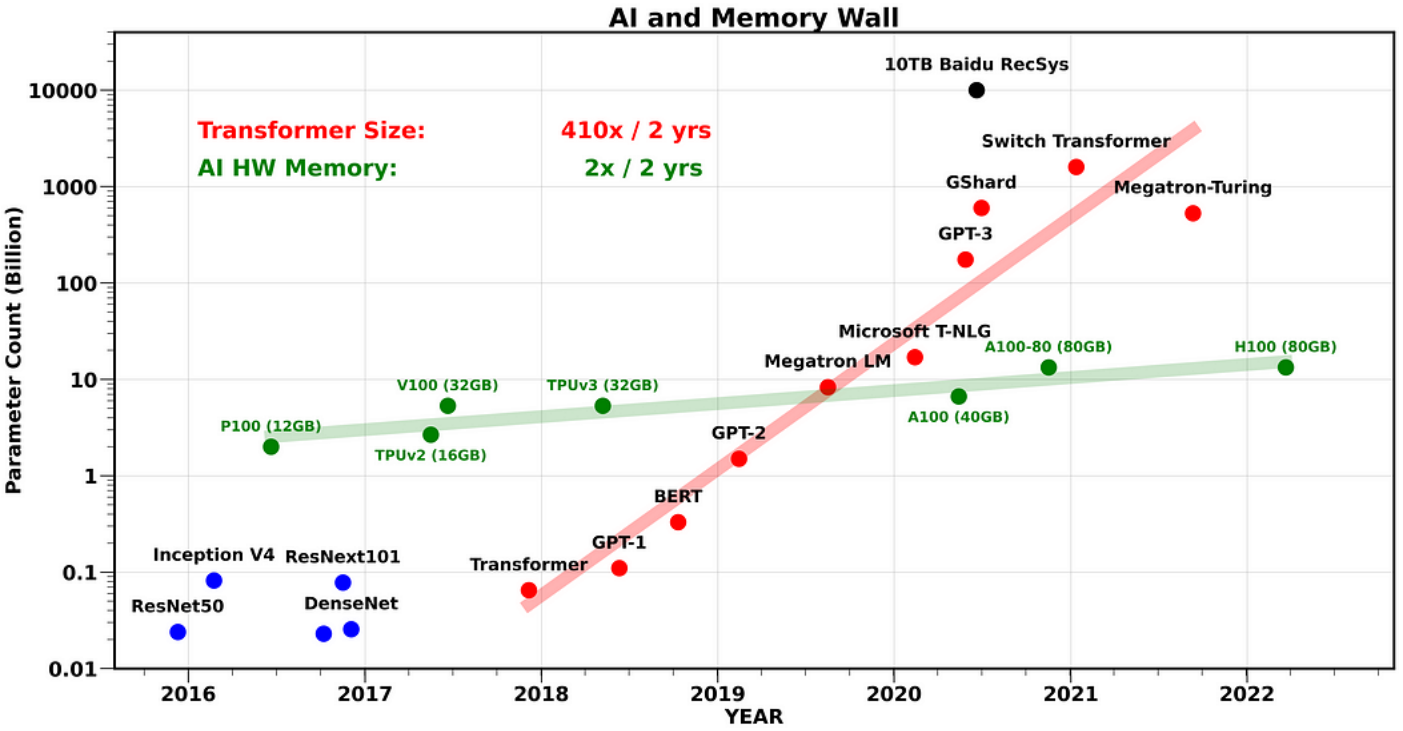
蚂蚁集团 ANT GROUP

- **赵军平，**蚂蚁**异构计算与推理引擎**负责人

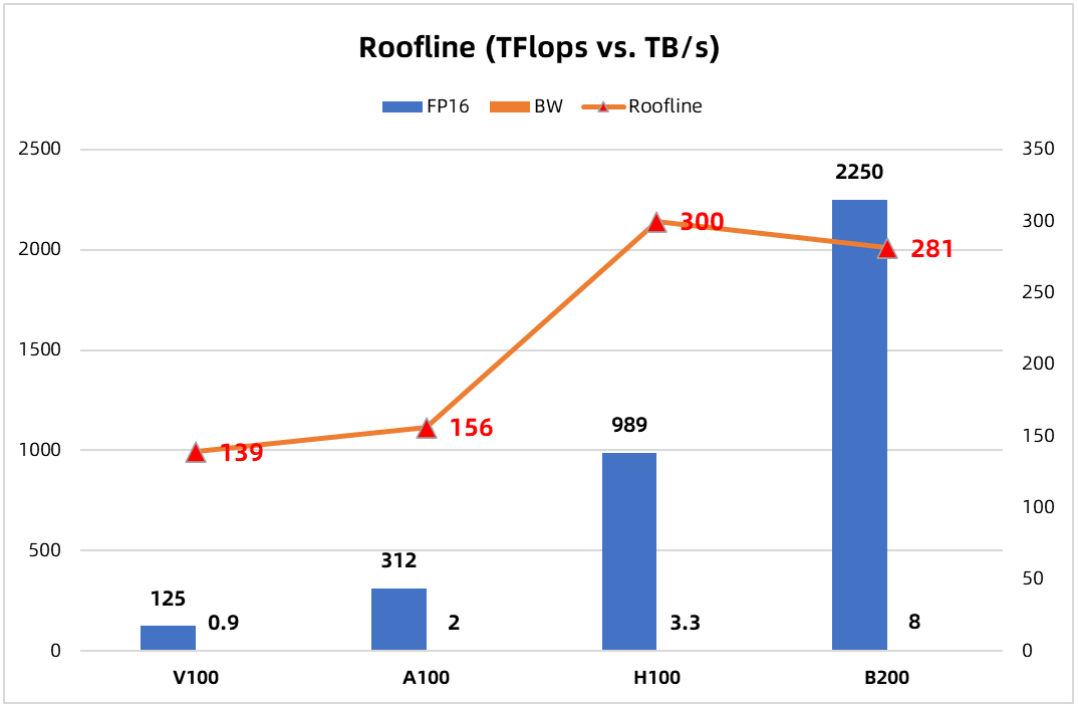
- CCF HPC和存储专委委员，~200 中/美技术专利
  - 异构加速，虚拟化，K8S，推理优化，文件系统，企业级存储-保护等
  - “数据密集型应用系统设计” 译者
  - 2007~2019：EMC、Dell EMC，Start-up

LLM技术挑战1：显存墙

- 显存**容量**、访存**带宽**（特别是推理小batch场景）



模型参数量 vs. 单卡显存容量

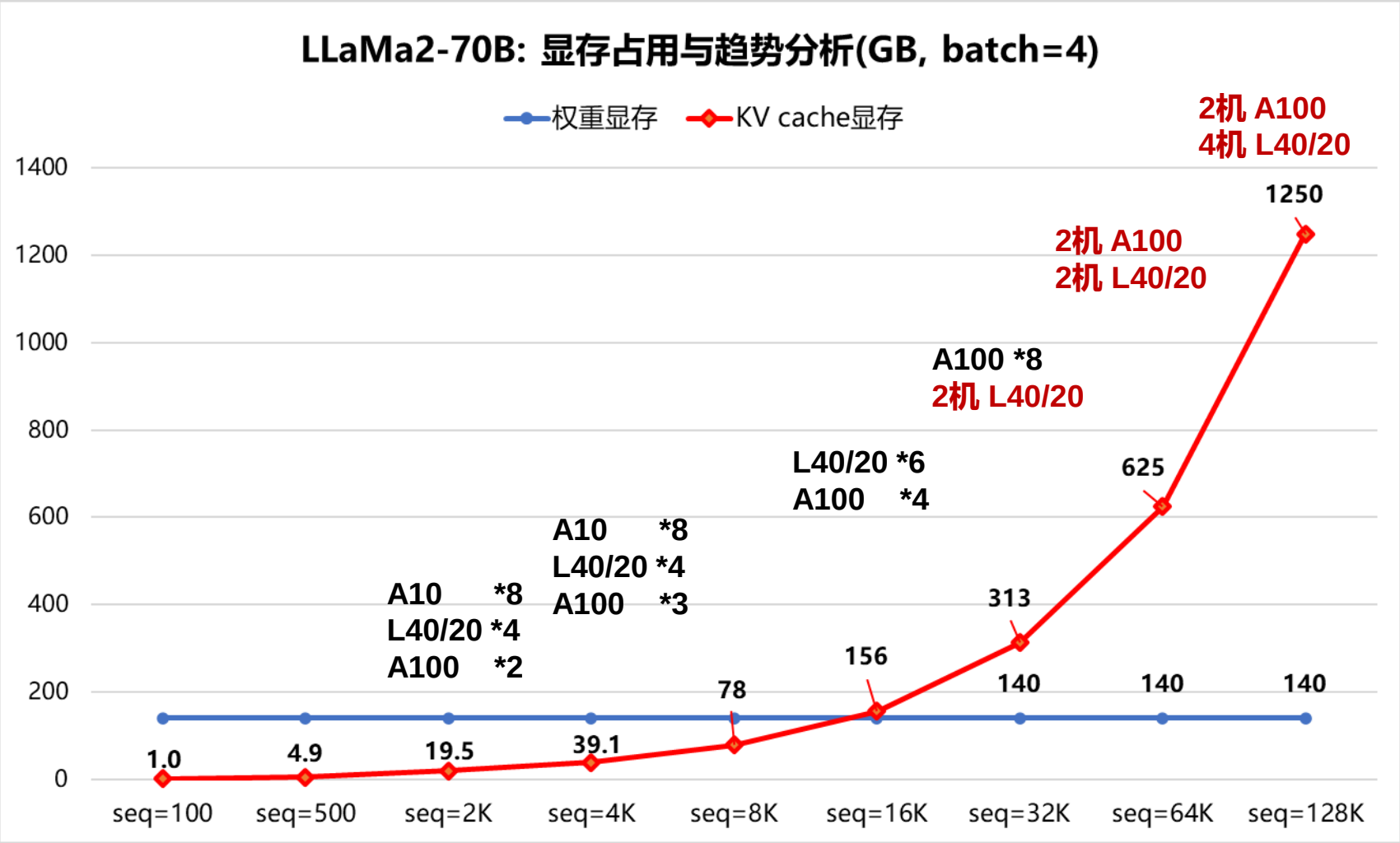


单卡算力 vs. 访存带宽

Llama2-70B推理为例

蚂蚁集团 ANT GROUP

- 权重与KV cache的显存消耗分析 （假定FP16 未优化）

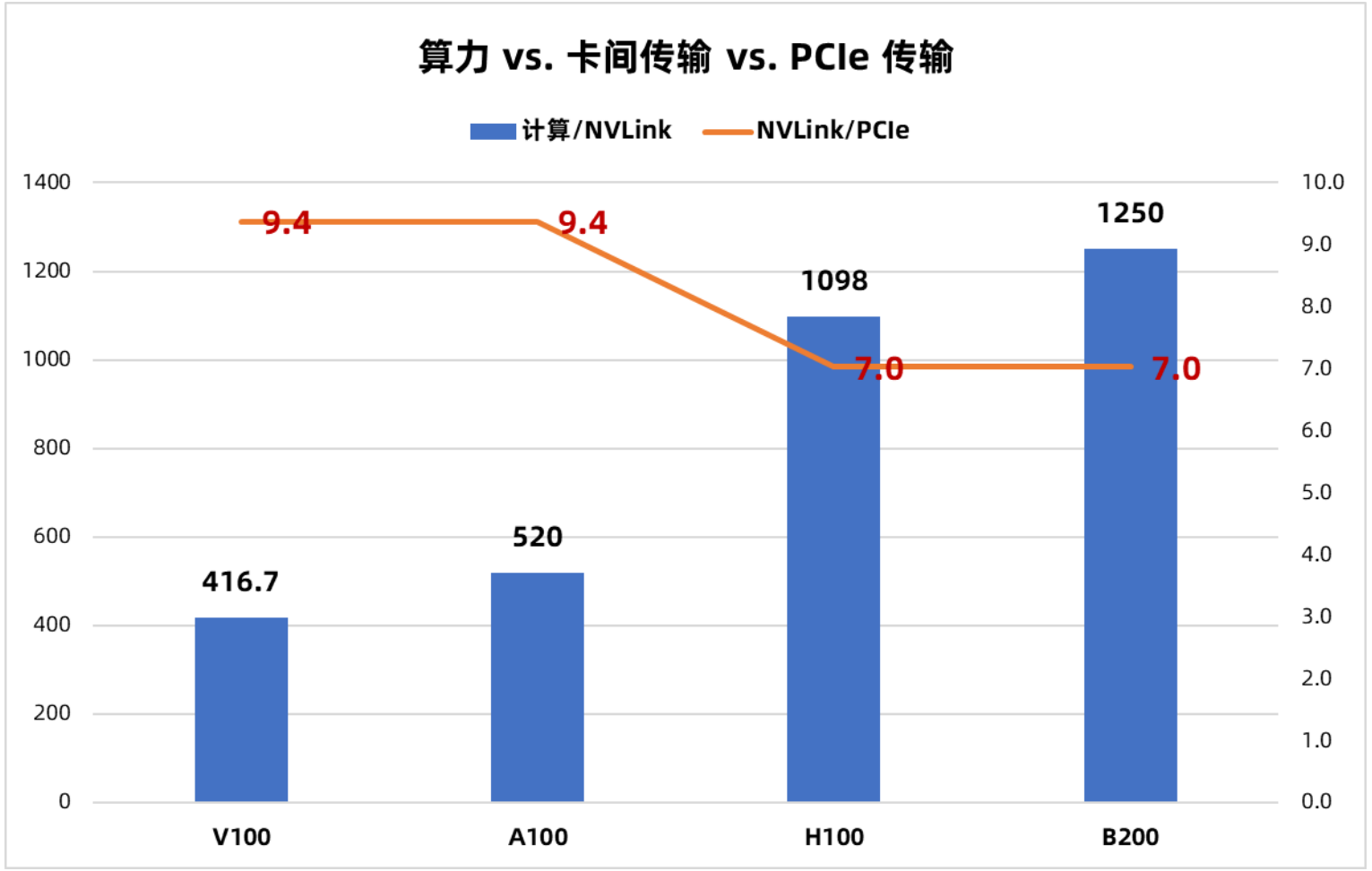


KV cache显存占用 = D# * L# * 2 * 2B * seq# * b#

LLM技术挑战2：传输墙

蚂蚁集团 ANT GROUP

- 访存、卡-卡传输、PCIe传输等发展 << 算力发展



业界方案

蚂蚁集团 ANT GROUP

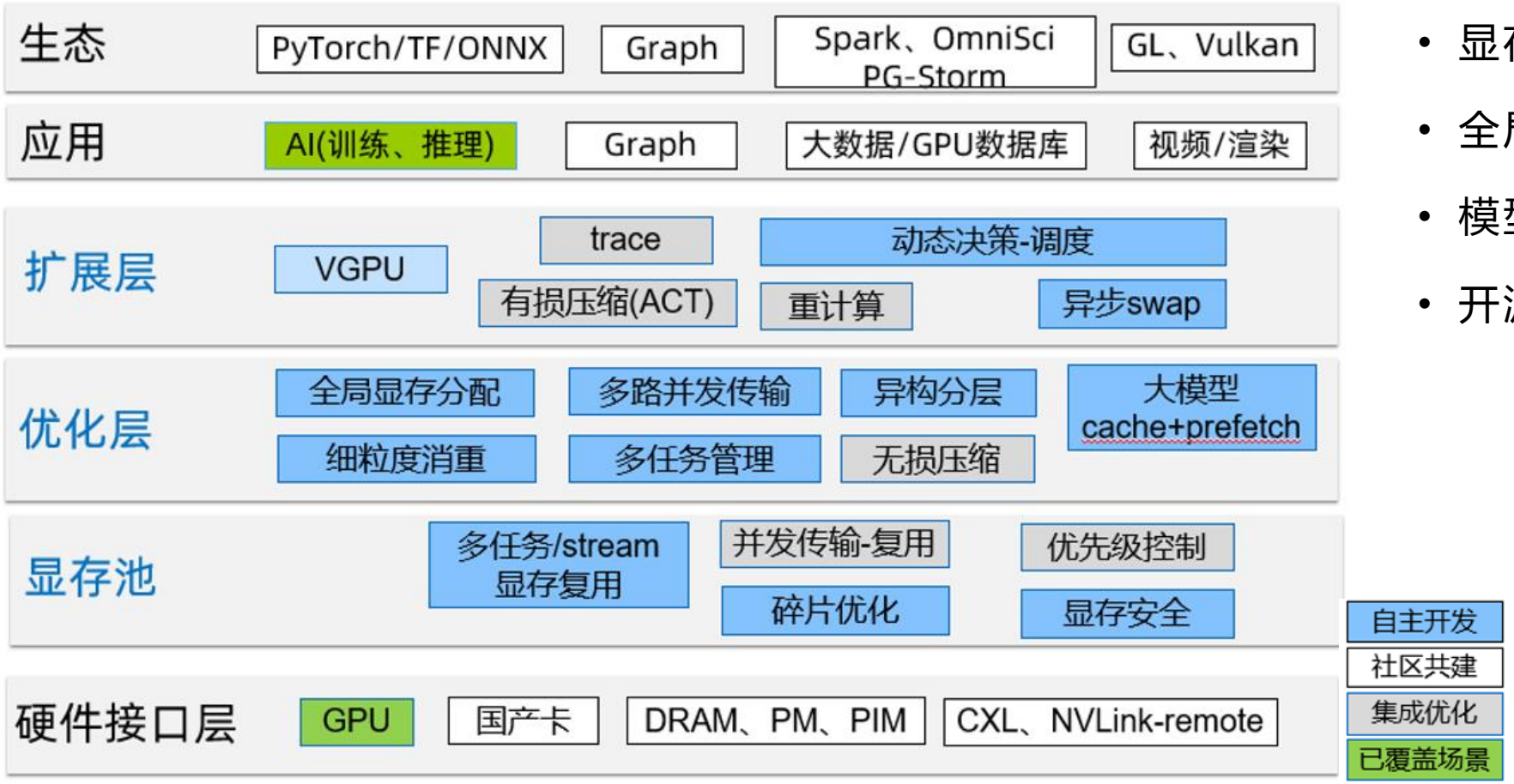
思路	精度	性能	范围
模型/算法/实现改造 • DeepSeek • YOCO • MoD, MoE • LoRA	Case by case	Case by case	case by case
梯度累计	depends	中~高	训练过程适配
重计算	无影响	中	通用 计算换显存
offloading/swapping	无影响	低(同步) ~ 高(异步)	通用 包括UMA
混合精度、小型化	可能影响 - 无损压缩不影响	中-高	Dependents QAT, AMP, INT8
多卡并行:PP, TP, EP, SP	-	-	Depends 模型改造
以上组合			

观察

蚂蚁集团 ANT GROUP

- 显存碎片化
  - 训练阶段管理开销，特别是recompute、offloading、并行等叠加使用
  - 推理阶段LLM KV cache: 动态生成
- 计算-传输-显存需一体优化
  - 多通道、细粒度融合、overlapping
- 从单任务 -> 多任务、弹性服务（例如共享混部、serverless）
  - → 系统层、全局的管理与优化；模型透明 → **GLake**

蚂蚁集团-GLake总体架构



- 显存、传输一体系统优化
- 全局：跨卡、跨进程、跨容器
- 模型基本透明
- 开源、开放、共建

训练场景：显存碎片问题

蚂蚁集团 ANT GROUP

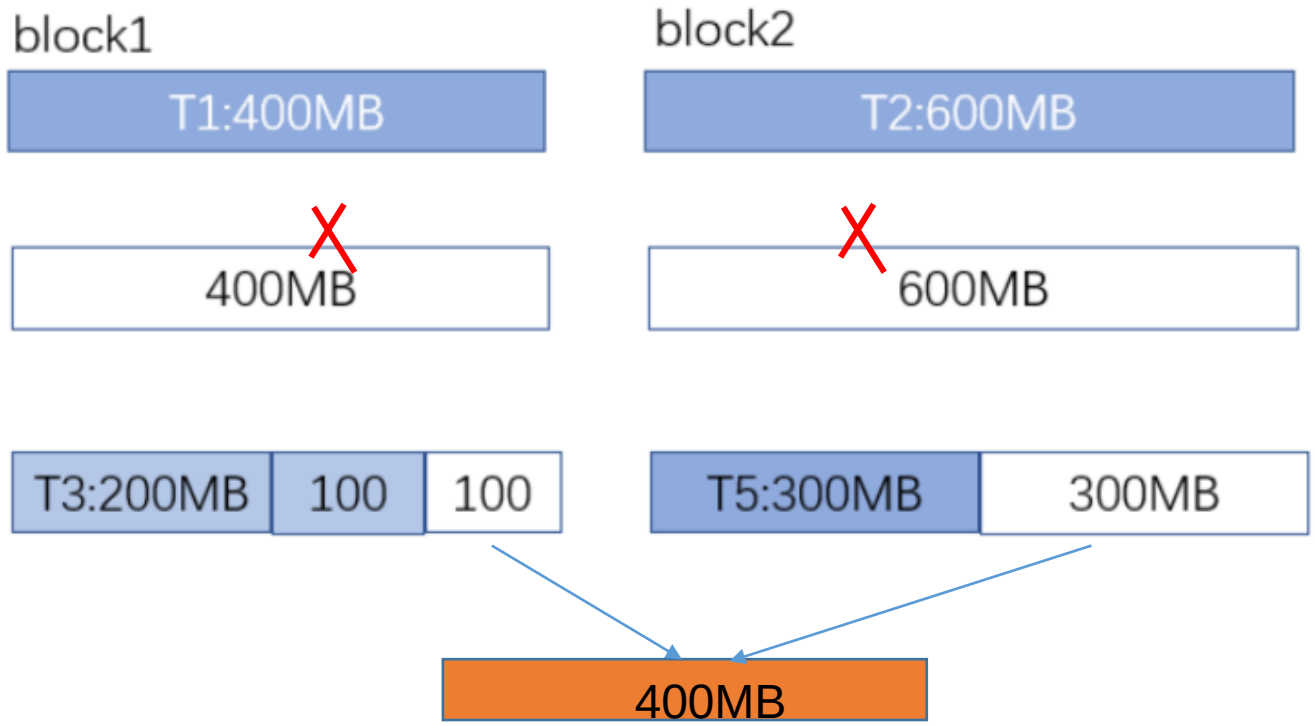
- 例子

先设置: torch.cuda.set_per_process_memory_fraction(0.065): 这样限制上限使用 ~1GB

1: new 2个GPU Tensor1/2
400MB, 600MB

2: 删除T1, T2

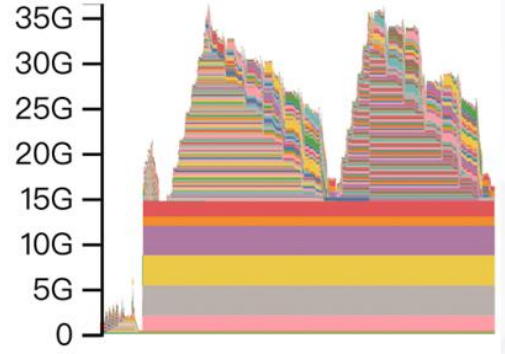
3: new Tensor3/4/5
200MB, 100MB, 300MB



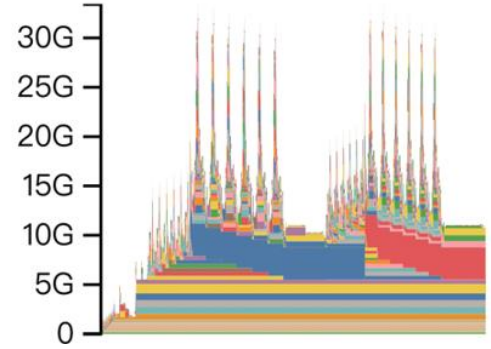
显存碎片原因分析

蚂蚁集团 ANT GROUP

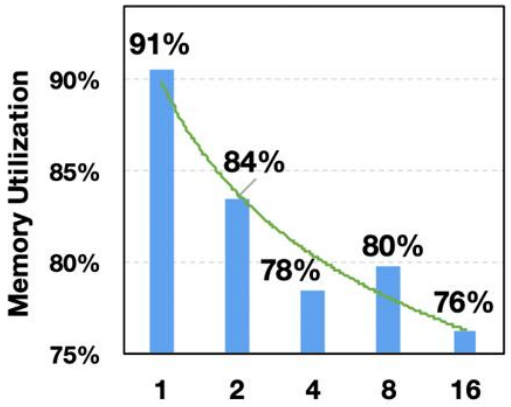
- CUDA不支持释放部分 (空闲)显存区域
- 访存特征动态变化，LLM更加复杂
  - 数据集长短不一、训练->评估、cuDNN workspace等
  - 大模型：
    - Recompute
    - 分布式：多卡、多stream并发
    - FSDP/DeepSpeed offloading
    - LoRA



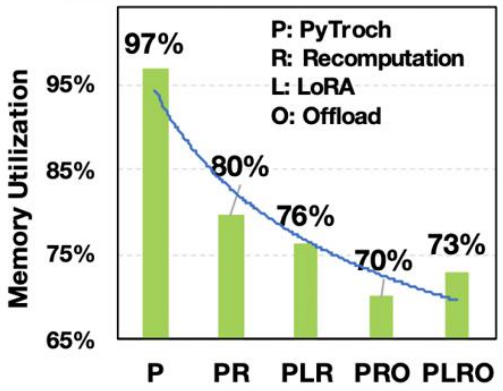
原始pattern



激活recompute后



不同卡数



不同优化策略

$$\text{Memory Utilization} = \frac{Active\ \text{GB}}{Reserved\ GB}$$

如何显存碎片优化

蚂蚁集团 ANT GROUP

挑战

- **释放部分使用**的block?
  - CUDA没有直接提供该能力API
- ”碎片整理”的**性能**影响?
  - 搬移-拷贝数据需同步计算，影响性能
- **复杂性**?
  - 对用户/模型透明，尽量无感知

思路

减少碎片产生

- Tensor及时释放-> 工具
- BFC(first-best-match-sz) -> least frag impact
- 结合Tensor类型: 同类型(权重/中间值/ws)临近分配
- 提取tensor访问特征：生命周期相近的临近分配

优化解决碎片★

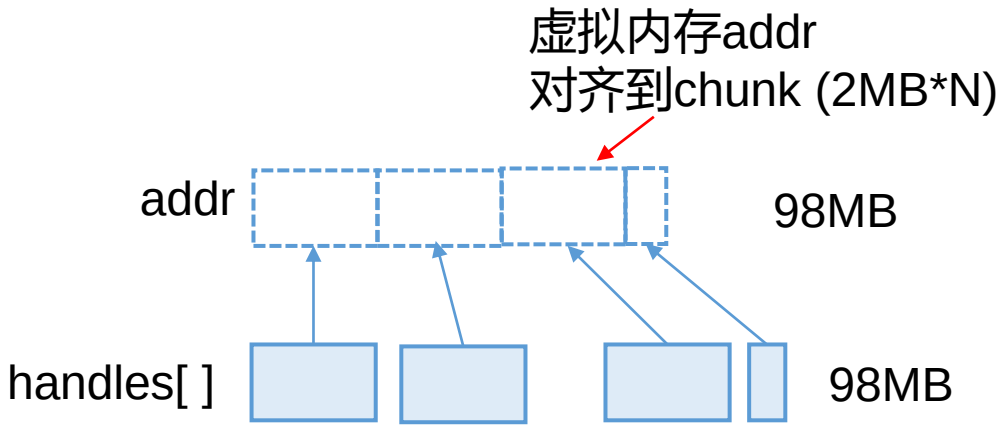
- 基于**细粒度**chunk分配和释放，
- **异步**分配、释放：减少性能影响
- **无数据搬移**: 对应用透明, 降低复杂度

基本思路

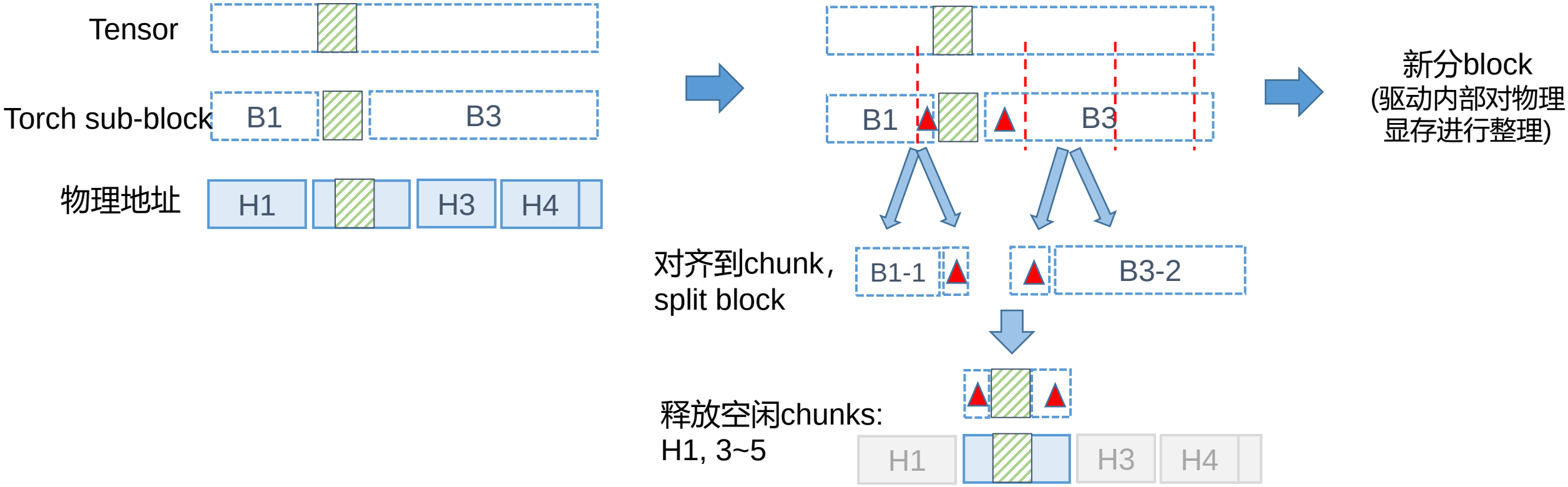
蚂蚁集团 ANT GROUP

2层指针与动态remapping （基于CUDA VMM）

- **虚拟**：对用户可见，保证连续
- **物理**：CHUNK_SIZE（2MB）粒度分配，不要求连续
- **Remapping**：动态回收chunk与重映射



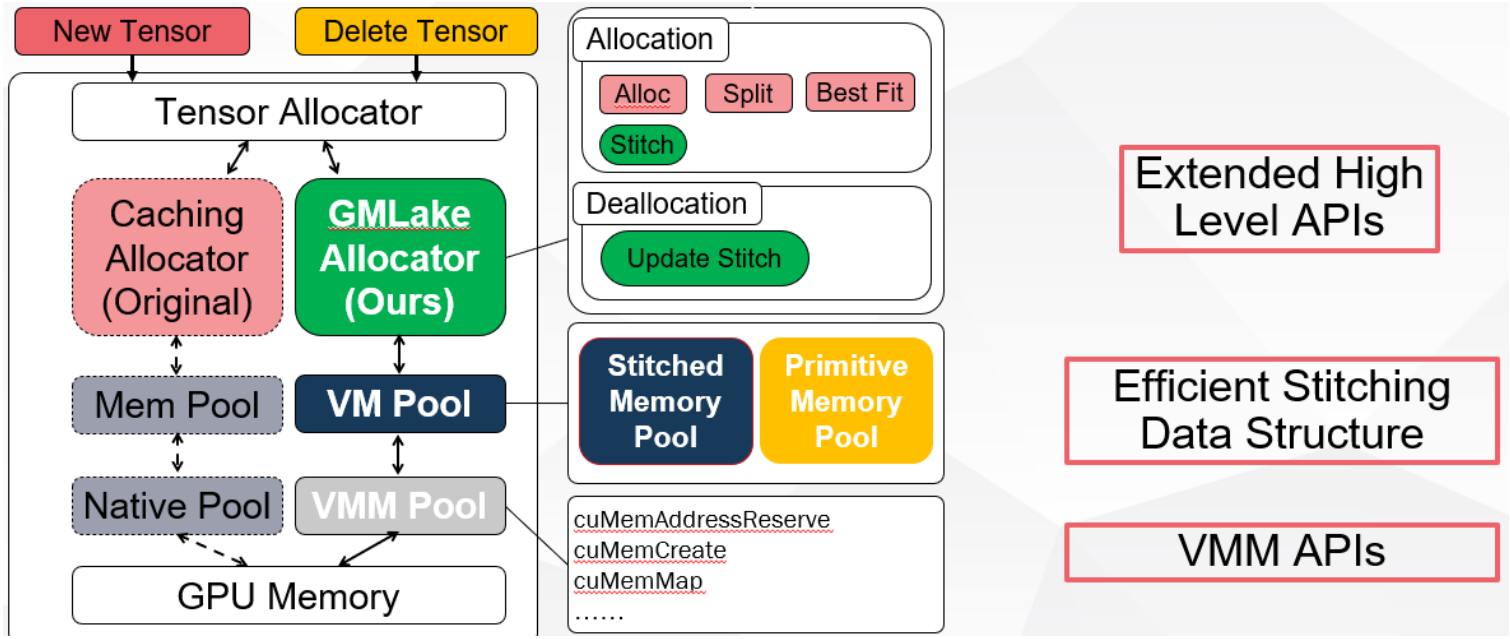
Remap例子


Tensor
Torch sub-block
B1
B3
物理地址
H1
H3
H4
B1
B3
新分block
(驱动内部对物理显存进行整理)
对齐到chunk，
split block
B1-1
B3-2
释放空闲chunks:
H1, 3~5
H1
H3
H4

GLake：优化大模型训练显存碎片

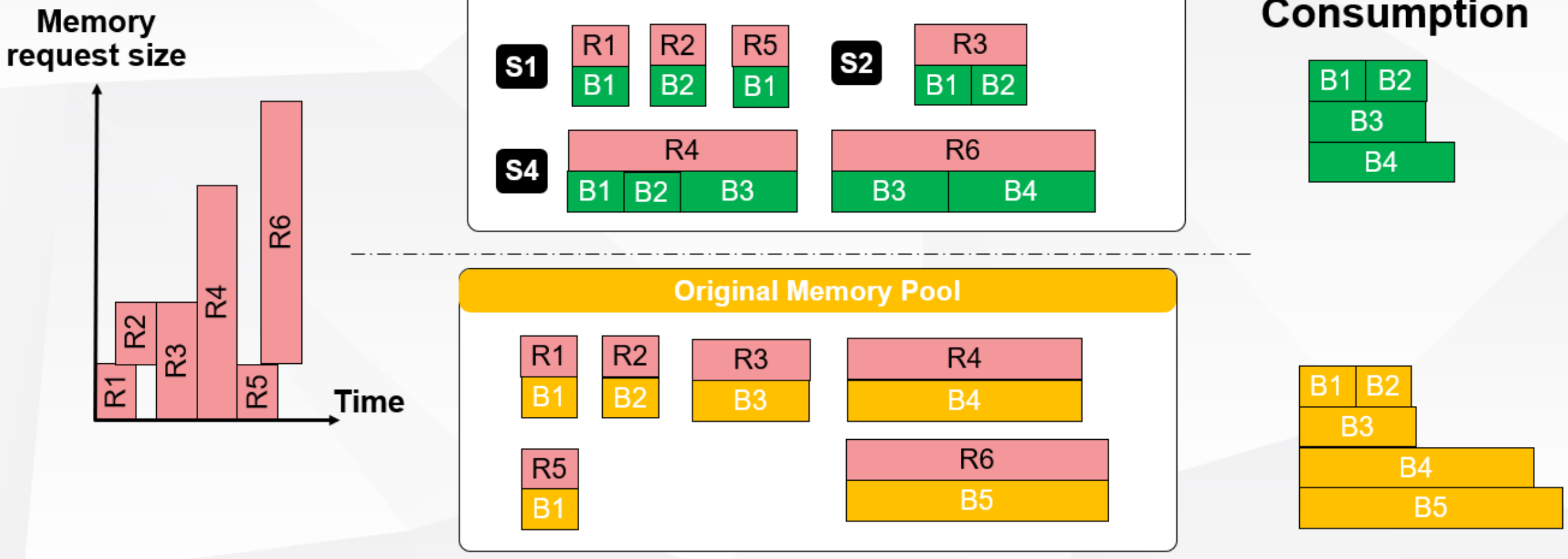
蚂蚁集团 ANT GROUP

- PyTorch caching allocator, pluggable; 对模型透明
- 重点：映射元数据管理（无数据copy），策略控制

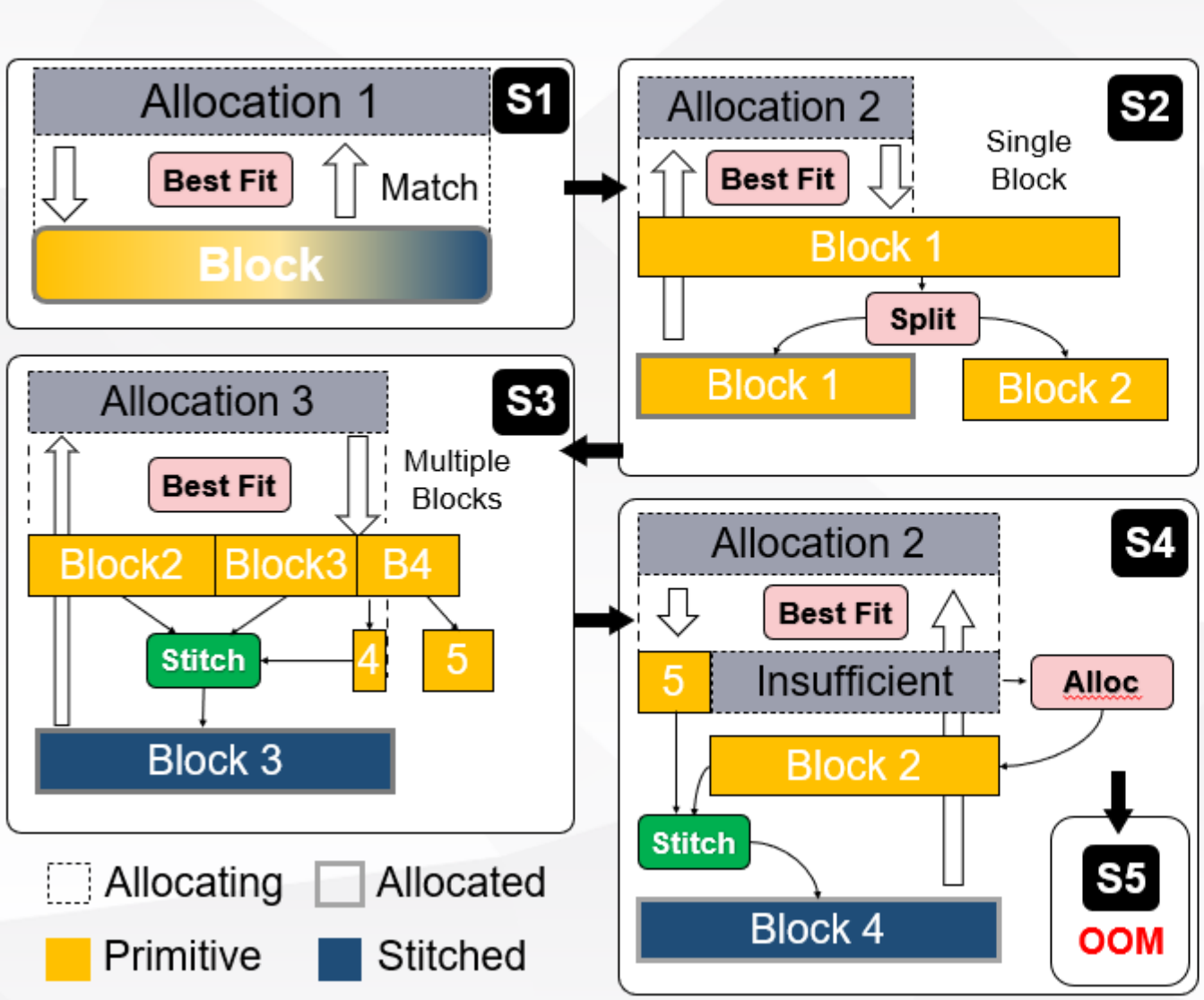


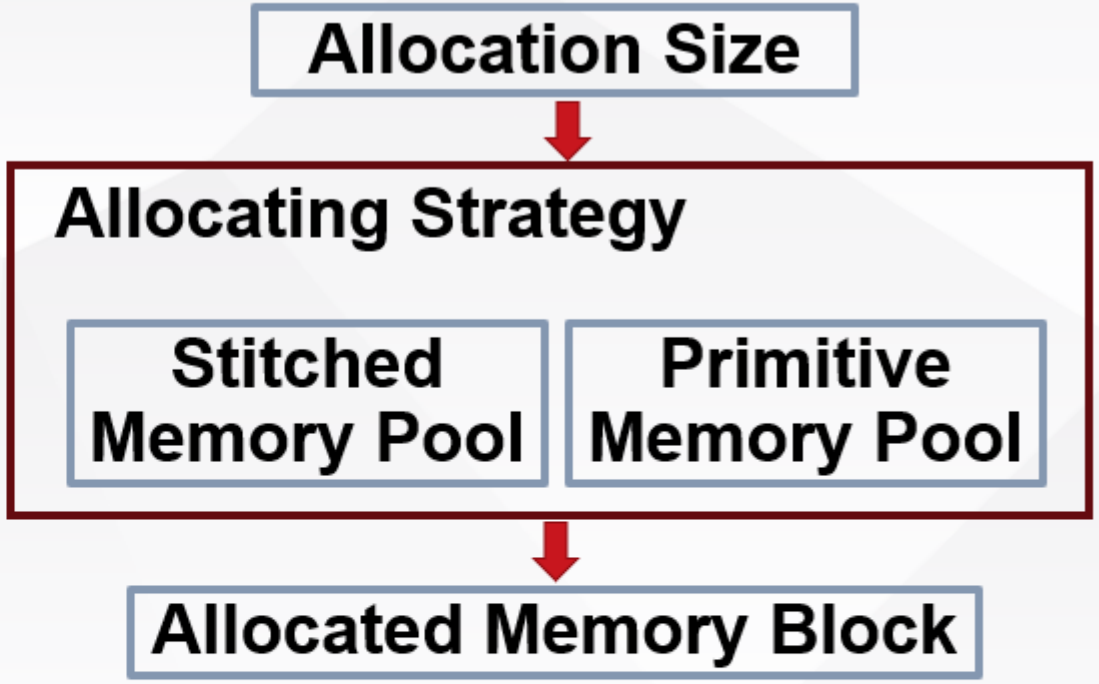
GMLake: Efficient and Transparent GPU Memory Defragmentation for Large-scale DNN Training with Virtual Memory Stitching, ASPLOS24, 蚂蚁集团、上海交大等

Case Study


Memory request size
R1
R2
R3
R4
R5
R6
Time
GMLake Memory Pool
S1
R1
B1
R2
B2
R5
B1
S2
R3
B1
B2
S4
R4
B1
B2
B3
R6
B3
B4
Original Memory Pool
R1
B1
R2
B2
R3
B3
R4
B4
R5
B1
R6
B5
Memory Consumption
B1
B2
B3
B4
B1
B2
B3
B4
B5

GMLake分配策略设计

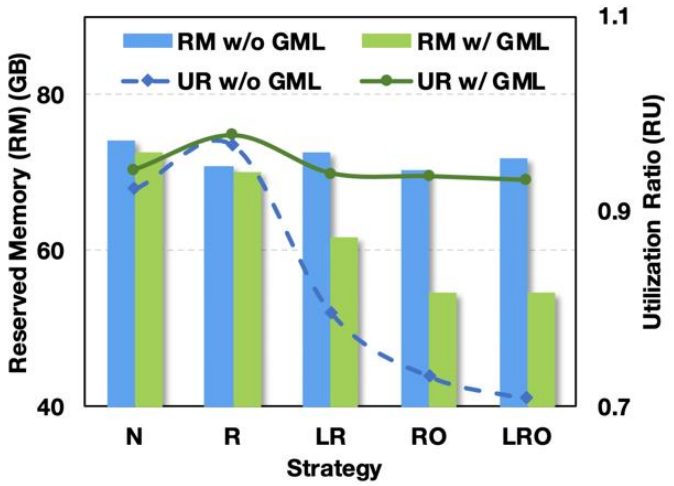




效果评测

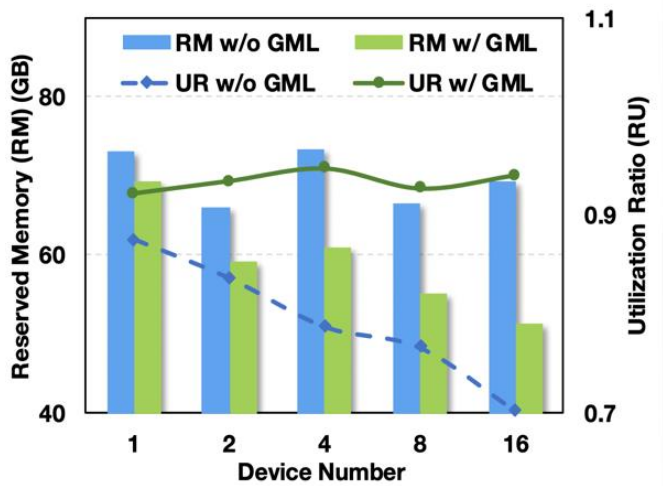
硬件环境	16 × NVIDIA A100 SXM （80GB）
训练框架	FSDP, DeepSpeed ZeRO, Colossal-AI Gemini
模型	8 LLMs including OPT, Vicuna, GPT-2, GPT-NeoX and so on
不同对比	Batch-size, Strategies, Distributed Training Devices and Different Training Engines
负载	76 fine-tune workloads

Avg 15%
Upto 33%



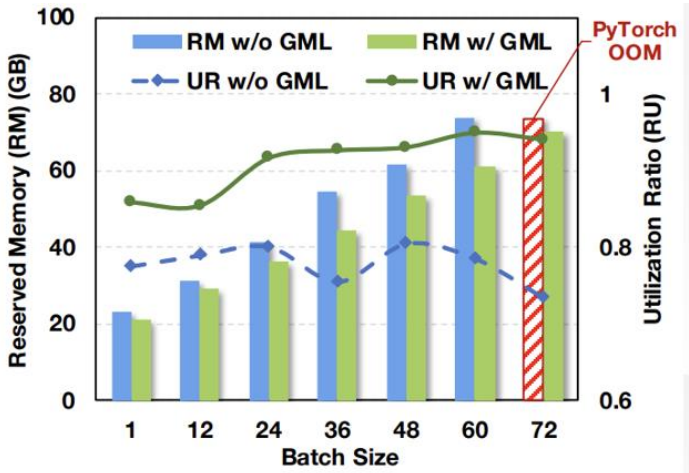
不同优化策略下

10 GB on average & **17** GB at most



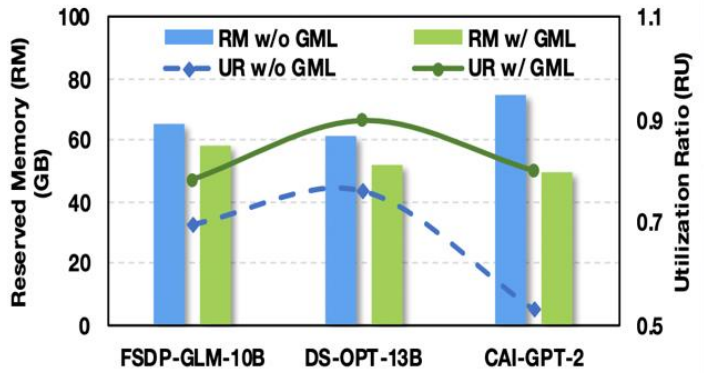
不同卡数

9 GB on average & **17** GB at most



不同batch size

7 GB on average & **12** GB at most



不同的训练框架

14 GB on average & **25** GB at most

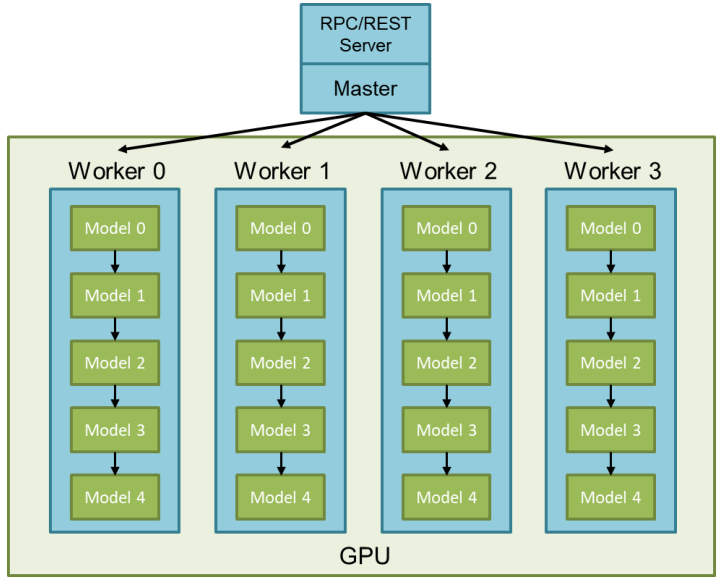
其他

蚂蚁集团 ANT GROUP

- 与PyTorch ExpandSegment对比
  - 相同：都借助VMM接口
  - 不同：GLake的分配、粘合策略不同。内部实测效果优于ExpandSegment。二者可互补（实现中）
- 扩展成为PyTorch pluggable allocator，方便lib替换方式使用
- 扩展支持跨stream：机会复用
  - 开源、复用策略可灵活配置

GLake: 推理场景

- **case1**: 单模型**多进程实例（python）**
  - 挑战：跨进程的权重、中间结果的显存如何复用
- **case2**：LLM推理**KV cache显存管理**
  - 长短不一动态生成，reserve低效；动态分配容易碎片
  - vLLM：PagedAttention，额外做特殊管理，修改kernel

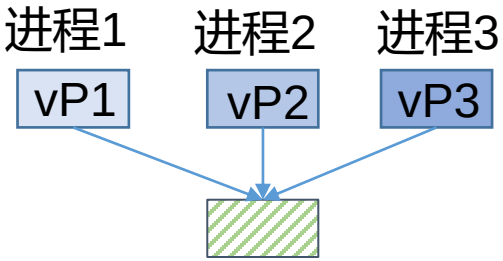


- GLake思路: **坚持系统层面全局优化入手，对模型、 kernel透明**

case1：多进程推理“显存dedup”

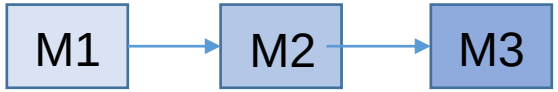
蚂蚁集团 ANT GROUP

- **权重(RO)：全局、细粒度、透明**共享，类似存储 “重删”
  - **全局**：支持跨线程、**跨进程、跨容器共**享
  - **透明**：无需用户参与、**无需配置**
  - **细粒度**：层或者block级共享，基于hash和字节比对



权重显存块

- **中间结果**：多模型（DAG）、跨进程中间结果的**时分复用**



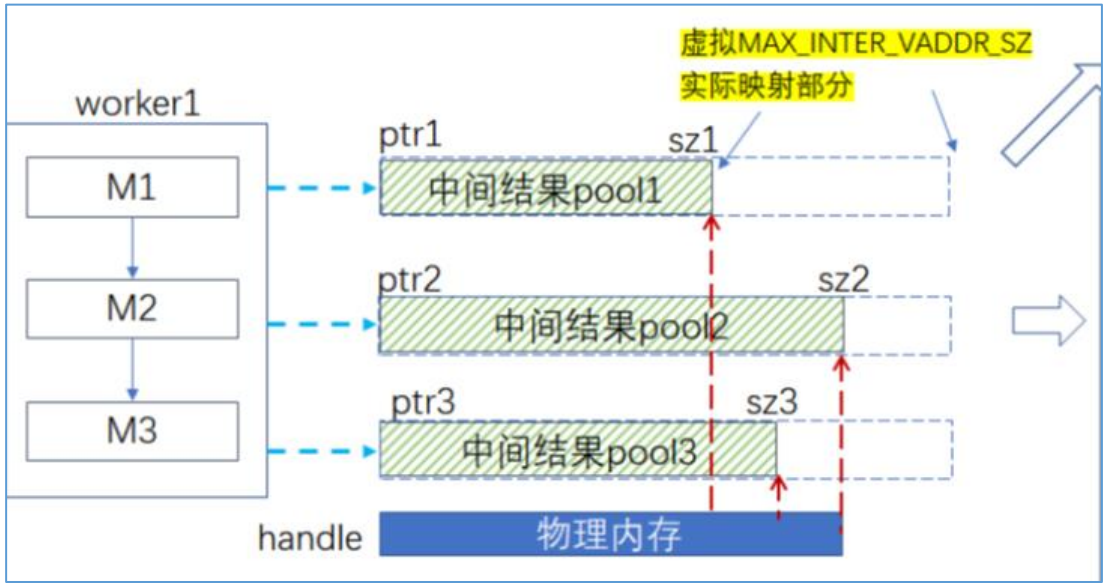
多模型DAG

不同于 Triton Server/ONNXRT:

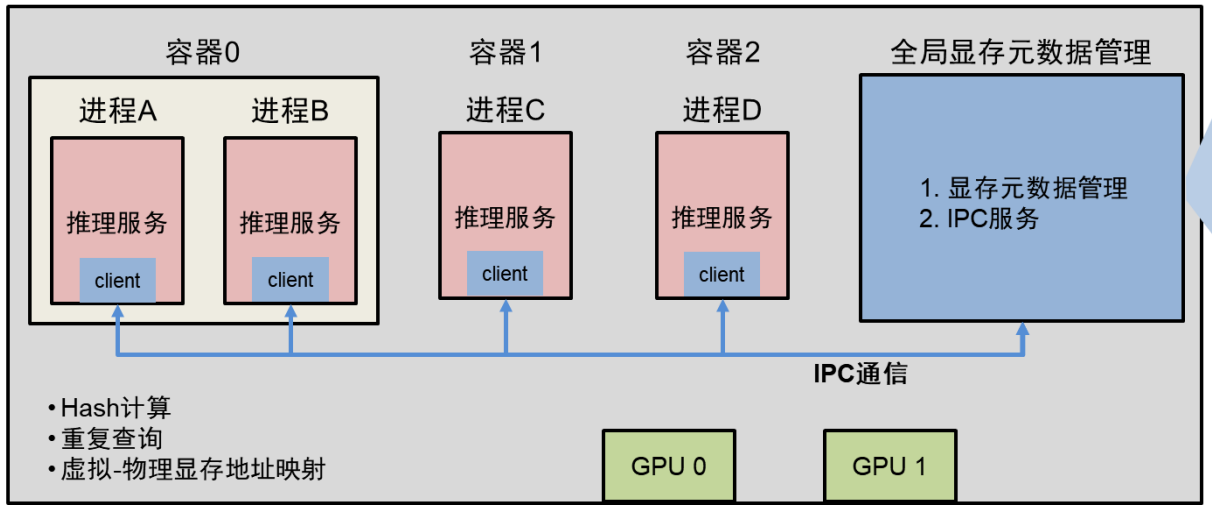
- **粒度**：模型所有W[]完全相同 vs. 显存块级（通常比tensor粒度更小）
- **范围**：单进程 vs. 跨进程、跨容器
- **使用**：手工配置 vs. 自动发现，透明使用

设计：跨进程、跨模型

- 核心：
  - 虚拟-显存指针的动态管理与映射
  - 跨进程export、引用计数等



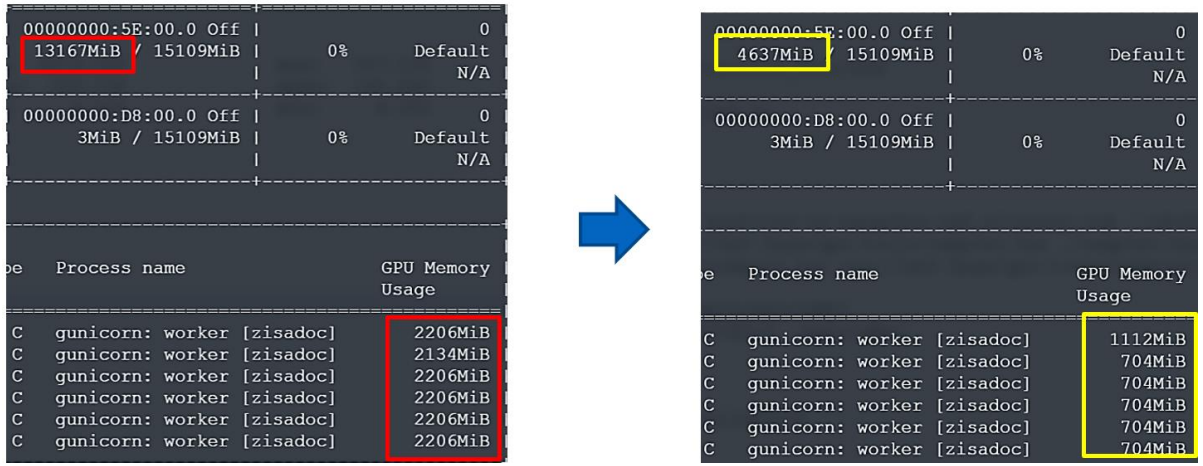
节点 1



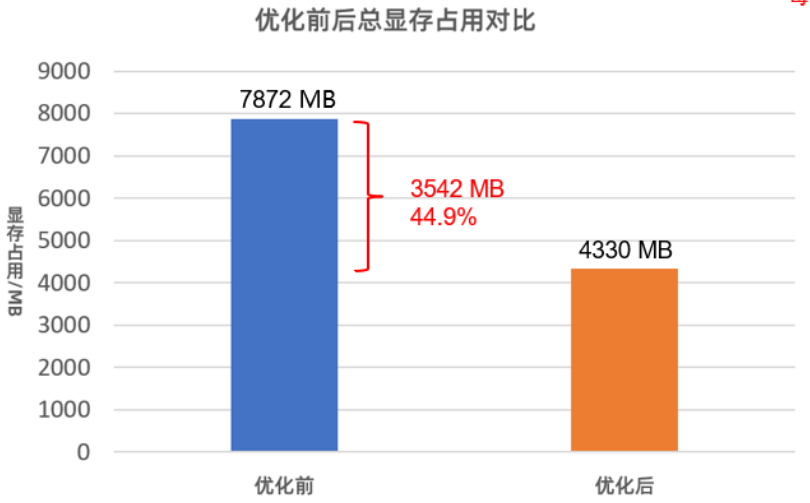
GPU#	权重大小	内容hash	物理显存地址
0	50MB	e260b31ad2	0xE0710000
0	24MB	b4ebe3387c	0xA3454120
1	12MB	b85ccdefc9s	0x78AE2390
...	...	...	...

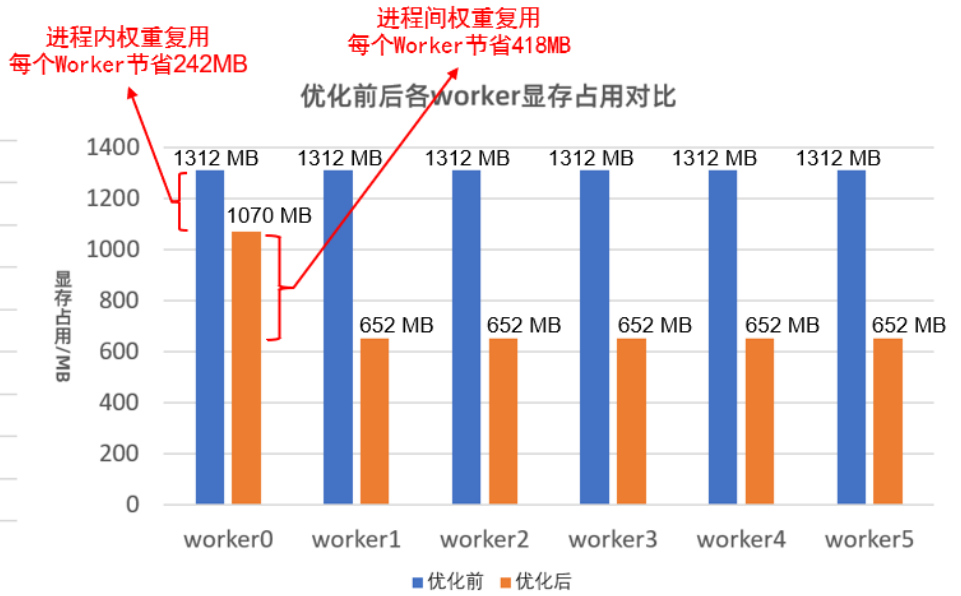
效果评测

- 6个进程总显存**13.2->4.6GB(-65%)**
  - 权重优化: - 3.3GB, 25%
  - 中间结果优化: - 5.3GB, 40%



- ✓ **精度 不影响**
- ✓ **性能 不下降**
- ✓ **模型 不感知**
- ✓ **用户 不用配**





Case2：LLM KV cache管理

蚂蚁集团 ANT GROUP

- 问题：动态生成，***GB：与模型Hd#、Layer#以及运行时Batch#、Seq#成正比
- 现有方案：PagedAttention(vLLM): 显式管理block index，显著降低碎片&提高batch
  - **不足1**：现有attention kernel需要改造，比如FlashAtt, FlashDecoding
  - **不足2**：~10%~20+% kernel性能开销

- GLake：继续系统层优化思路。模型看到**大而连续的虚地址**，物理显存动态映射
  - **好处1**：所有atten kernel几乎**不需改造**，虚拟地址连续
  - **好处2**：特定场景下，**20%~350%**的kernel性能优势（特别是GQA decoder）
    - GLake+FlashAtten/FlashDecoding/FlashInfer vs. vLLM PagedAtten kernel
    - 其他：业界最新类似的思路，vAttention（未开源）

VMM不足

蚂蚁集团 ANT GROUP

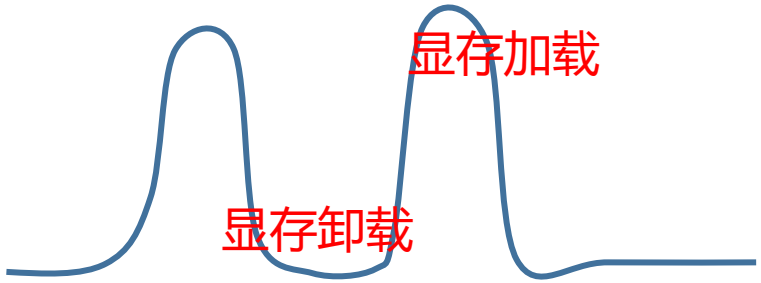
- 部分API调用耗时波动严重
  - cuMemSetAccess：10~100+X
  - 已反馈至NVIDIA：确认问题并初步分析了原因（lock attention）
- 部分硬件上kernel有影响
  - A100：~10%
  - 其他如A10、L40s、L20、H**等可持平
- 其他：API优化建议反馈至NV

其他场景：混部、serverless

流量波动、性能SLA需求差异

多任务共享、混部
- **显存动态切分**
- 算力动态压制

容器/进程保活下，显存自动保存和恢复



Serverless
- **显存(persistent)自动卸载 按需加载**

显存+传输的**全局、动态管理**
→ 提高GPU利用率、降低成本

总结

蚂蚁集团 ANT GROUP

- 显存-传输的巨大挑战
  - 硬件、互联演进：HBM4, NVLinkSwitch, CXL …
  - 软件的深层优化
- GLake：系统层全局、动态管理与优化，对模型、用户尽量透明
  - 进行中：
    - 混部+Serverless显存动态管理
    - 推理显存全局统一管理、精细分配
    - KV cache SmartRebuild
    - L2 cache, DSM, …

参考

- [1]**GMLake**: Efficient and Transparent GPU Memory Defragmentation for Large-scale DNN Training with Virtual Memory Stitching, https://arxiv.org/abs/2401.08156
- [2]Efficient memory management for large language model serving with **pagedattention**, https://arxiv.org/abs/2309.06180
- [3]**vAttention**: Dynamic Memory Management for Serving LLMs without PagedAttention, https://arxiv.org/abs/2405.04437
- [4] **LLM Inference Unveiled**: Survey and Roofline Model Insights, https://arxiv.org/abs/2402.16363
- [5] **SKVQ**: Sliding-window Key and Value Cache Quantization for Large Language Models, https://arxiv.org/abs/2405.06219
- [6] **AttentionStore**: Cost-effective Attention Reuse across Multi-turn Conversations in Large Language Model Serving, https://arxiv.org/abs/2403.19708
- [7] **You Only Cache Once**: Decoder-Decoder Architectures for Language Models, https://arxiv.org/abs/2405.05254
- [8] **DeepSeek-V2**: A Strong, Economical, and Efficient Mixture-of-Experts Language Model, https://arxiv.org/abs/2405.04434
- [9] **LoongServe**: Efficiently Serving Long-context Large Language Models with Elastic Sequence Parallelism, https://arxiv.org/abs/2404.09526
- [10] **QServe**: W4A8KV4 Quantization and System Co-design for Efficient LLM Serving, https://arxiv.org/abs/2405.04532

谢谢！

蚂蚁集团
ANT GROUP

开源： https://github.com/intelligent-machine-learning/glake

欢迎star、欢迎交流、欢迎共建