智能体原生GPU架构设计
Agent-Ready GPU Design And Implementation

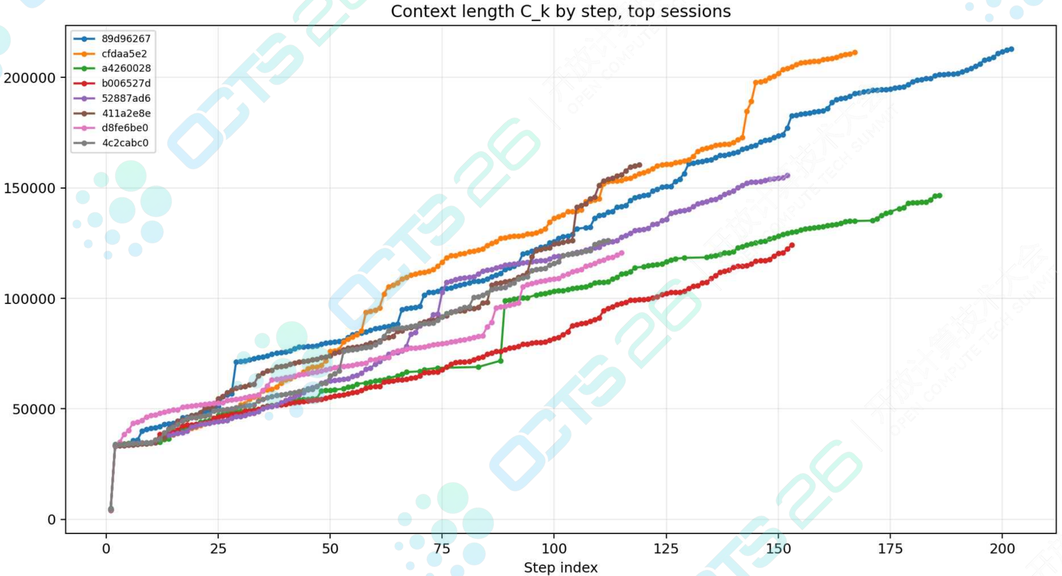
李兆石 沐曦股份

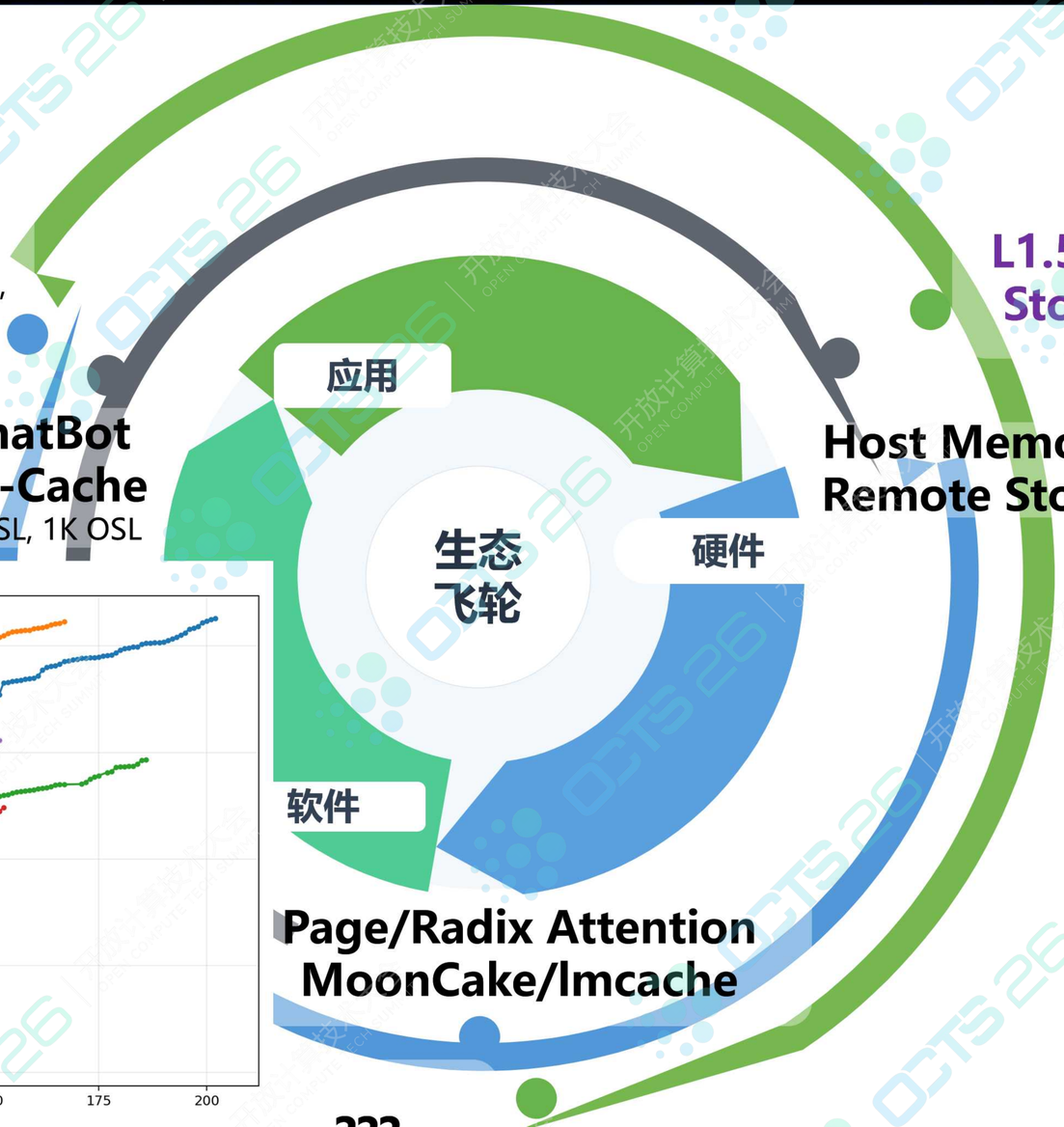
Zhaoshi Li, MetaX

智能体时代的生态飞轮

Agentic Context
512K Prefix-Hit, 8K ISL, 256 OSL
Periodic Compaction

ChatBot KV-Cache
8K ISL, 1K OSL





Host Memory + Remote Storage

Page/Radix Attention
MoonCake/lmcache

*** Emerging**
L1.5 GPU-Initiated Storage/Memory

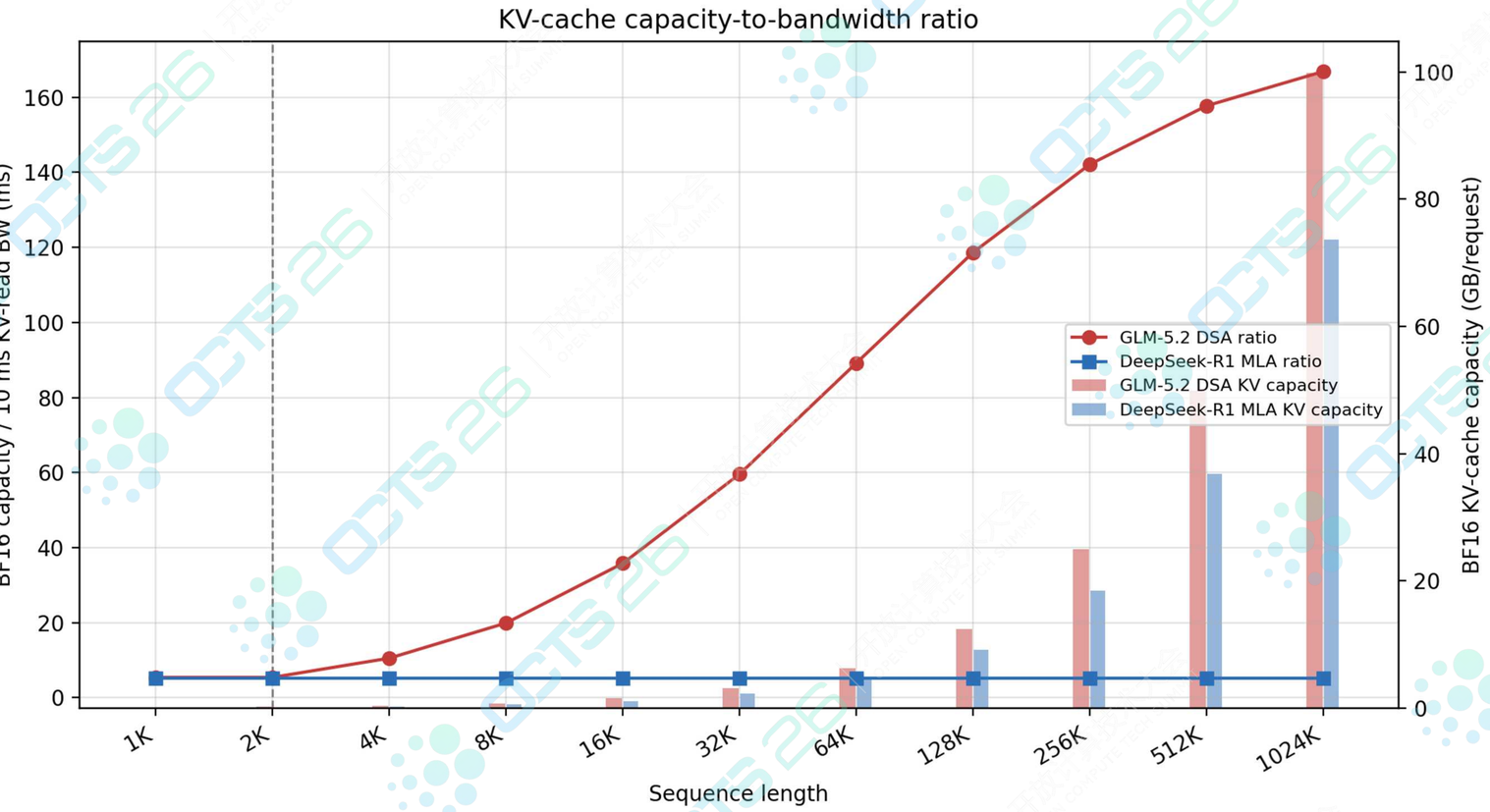
L1 GPU 显存: hot KV compute tier

L2 Host Memory / CPU DRAM: capacity expansion

L3 External / Distributed Storage Backend: external shared tier

智能体时代的挑战

- 计算与存储分离：计算在GPU上，存储中除了L1 GPU 显存外都要通过CPU
  - **当前vLLM/SGLang的标准三层抽象：L1 / L2 / L3**
  - 当前L2及以上的访问都需要CPU参与，CPU成为系统瓶颈
- Agent下1M以上的上下文长度成为刚需，需要频繁访问L2及以上存储



L1 GPU 显存: hot KV compute tier

L2 Host Memory / CPU DRAM: capacity expansion

L3 External / Distributed Storage Backend: external shared tier

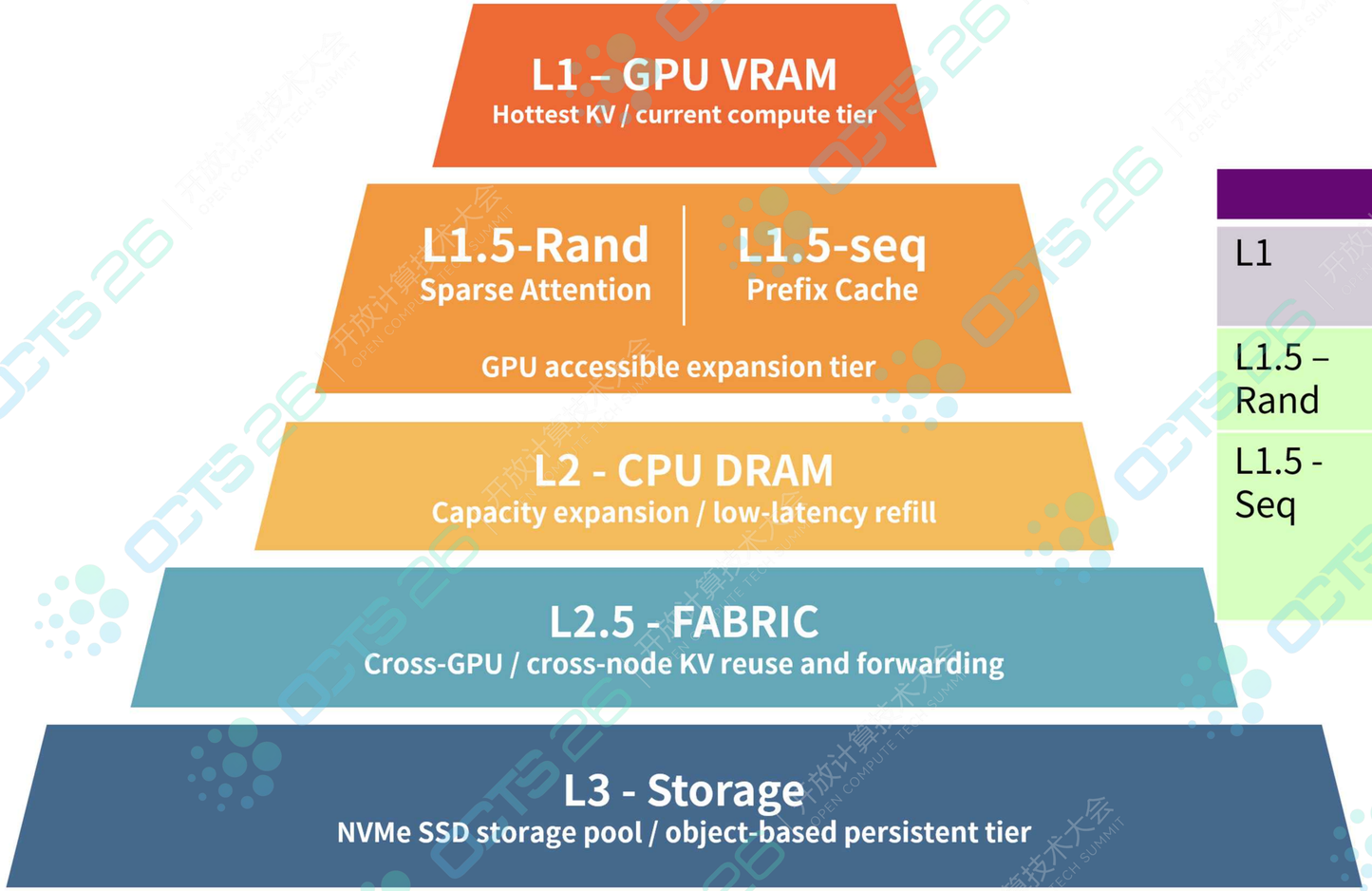
KV-Cache缓存命中折扣率

模型版本		DeepSeek-V4-Flash	DeepSeek-V4-Pro
思考模式		支持非思考与思考模式（默认） 切换方式详见思考模式	
上下文长度		1M	
输出长度		最大 384K	
价格	百万tokens输入（缓存命中）	0.02元	0.025元
	百万tokens输入（缓存未命中）	1元	3元
	百万tokens输出	2元	6元
并发限制[2]		2500	500

价格单位：元/M Token	Deepseek V4 - Flash/Pro	GLM – 5.2	GPT5.5
缓存未命中	1/12	8	34
缓存命中	0.02/0.1	2	3.4
输出	2/24	28	204
命中折扣率	**0.02/0.0083**	**0.25**	**0.1**

- Summary
  - 都针对输入输出，及是否缓存命中做了分别定价
  - 输入的缓存命中，在价格上都有折扣
  - **实际Agent使用中缓存命中费用占比最高**

Solution: 以GPU为中心的存储金字塔



分层定位

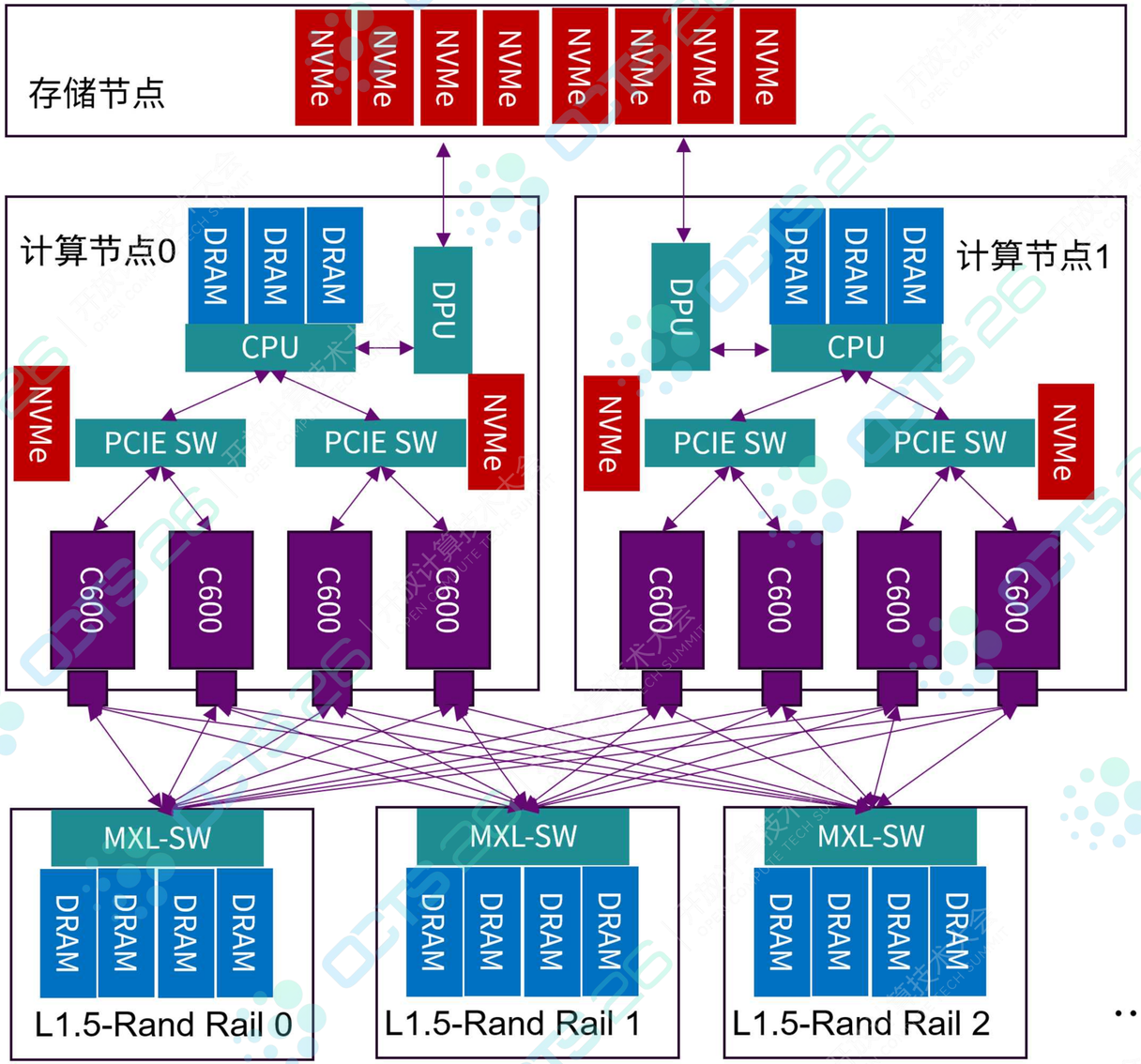
L1 / L1.5：贴近 GPU，服务高频命中

	硬件	软件框架	语义	应用
L1	VRAM	通用	内存语义 128 B 粒度	通用
L1.5 – Rand	CXL-MXME (DRAM)	自研DSA算子	内存语义 1KB粒度	DSA/CSA 显存扩展
L1.5 - Seq	北向PCIe NVME	GPU-Initiated Direct Storage	轻量级消息语义 => 异步内存语义	prefix kvcache

L2 / L2.5：承接容量与跨节点复用，避免重复 Prefill

L3：以对象化 KV + GPU 直接 IO 提升 SSD 回填效率

Solution: 以GPU为中心的存储金字塔



L3.5总容量：弹性可配

L2总容量：2~4TB（单机8卡共享 DDR5）

L1.5 SEQ总容量： N*4TB NVMe

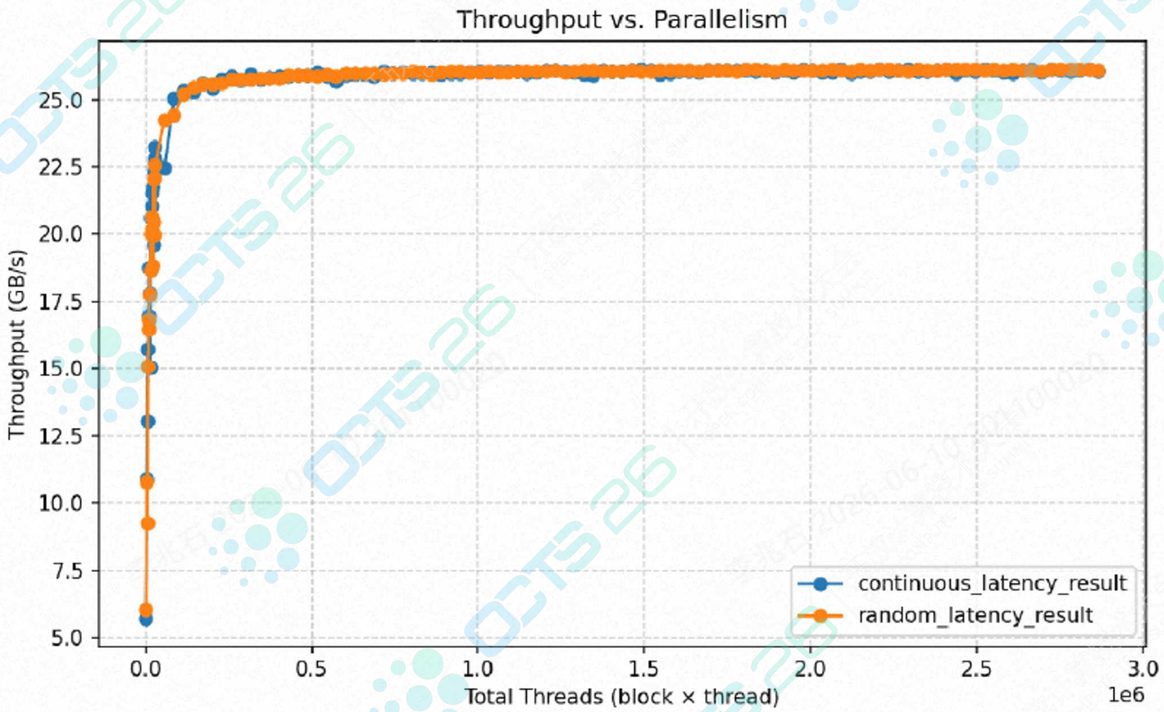
L1总容量：< 300GB 显存/GPU

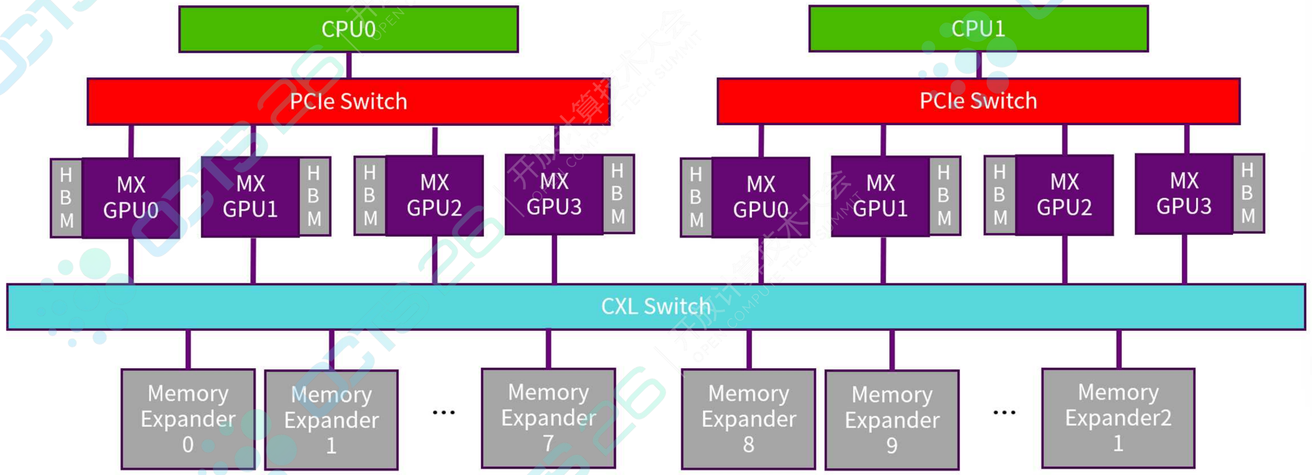
L1.5 RAND总容量：N*256GB DRAM

L1.5 RAND: MetaX Memory Expander (MXME)

- C600实测CXL Switch + 三星/澜起 DRAM Expander
  - GPU thread 读 round-trip latency 0.9 us
- Gen5x8 1KB随机访问，跟连续访存性能一致。

- 单机八卡MXME
- 模型：7B ~ 34B
- 总容量：22 x 0.512 = 11.264TB
- 总带宽：8 x 256Gbps





Sparse Attention: Cap/BW Shift

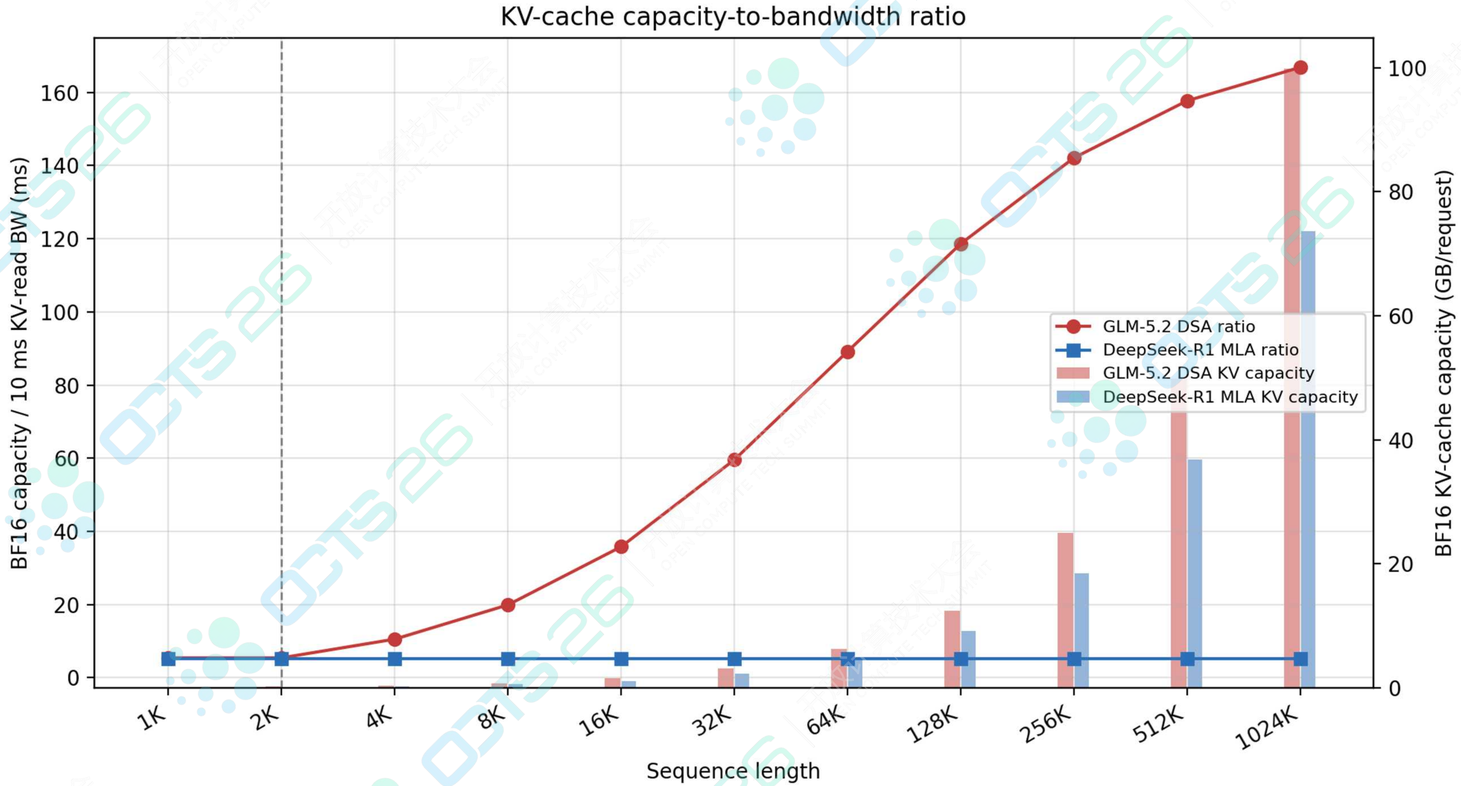
After top-k (2048), read BW saturates;
KV capacity and Indexer_QK keep growing

=> Cap/BW grows linearly with seq. length

GPU	Memory	Cap/BW
H200	HBM3E	29.38
5090	GDDR7	17.86
Thor	LPDDR5X	540.08

H200 cap/BW = 141 GB / 4.8 TB/s = 29.38 ms

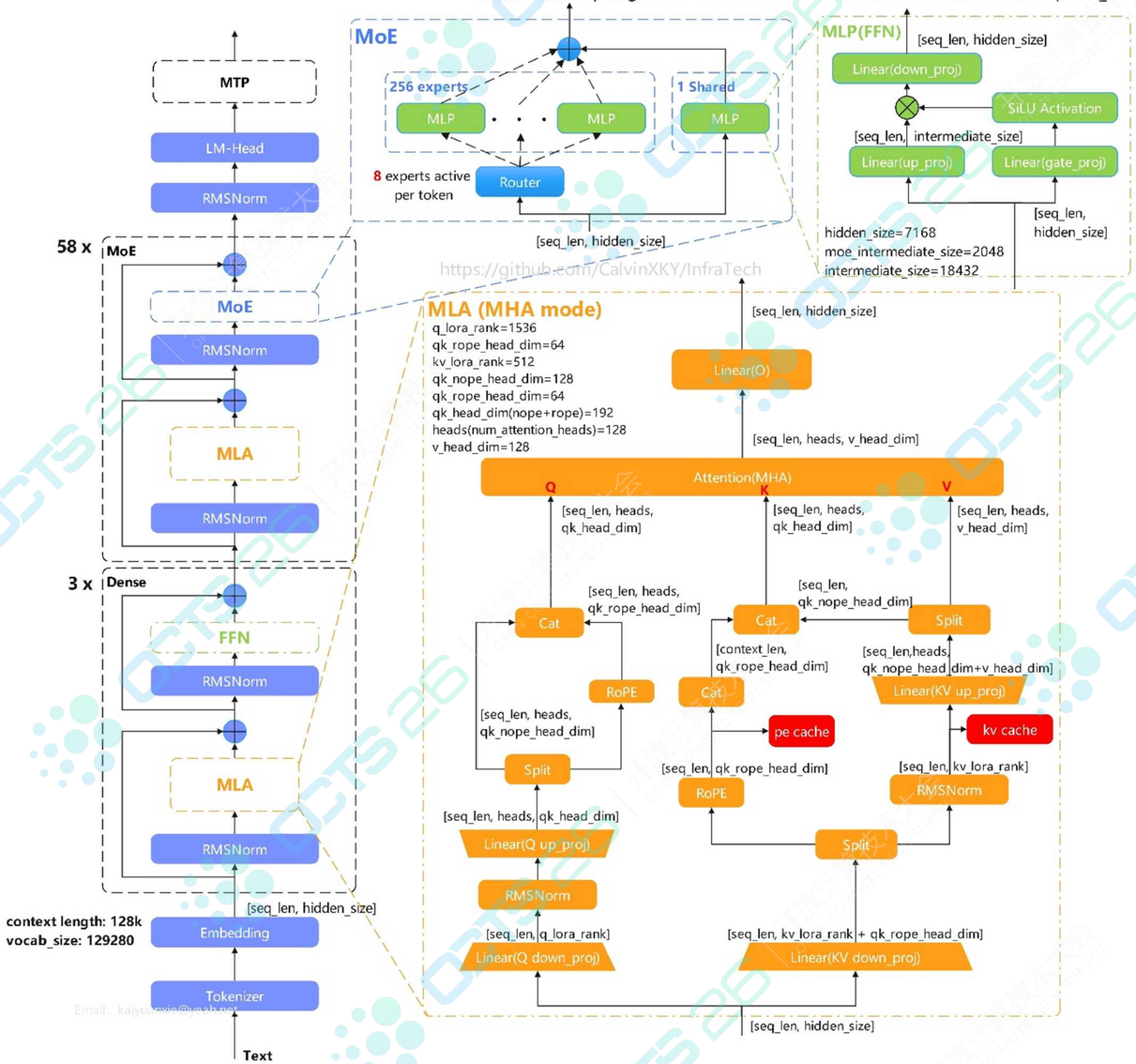
How much time does it to iterate the full capacity?



From MLA to DSA (Sparse Attention)

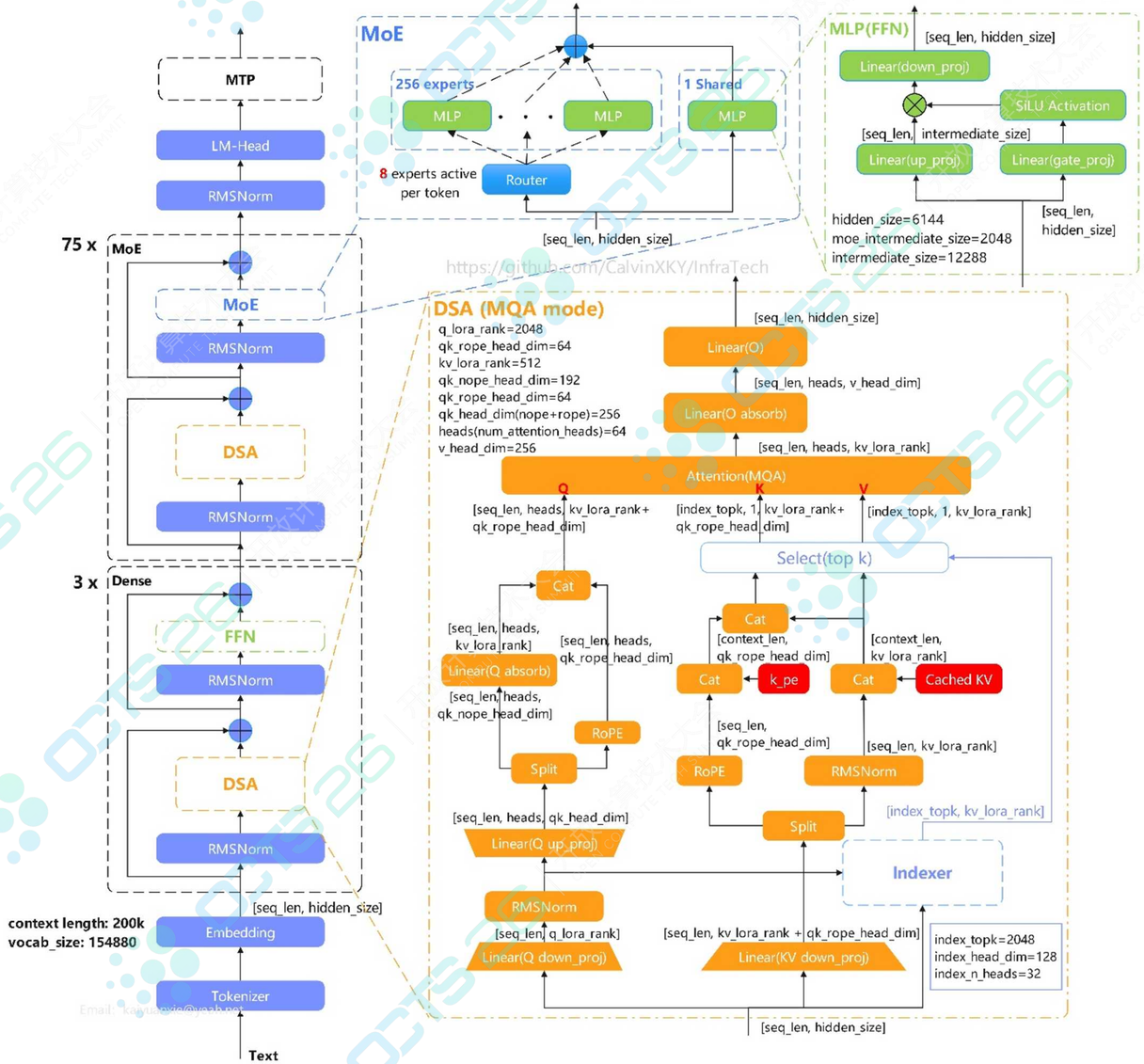
DeepSeek V3 (671B A37B)

高清图地址：https://github.com/CalvinXKY/InfraTech/tree/main/models/deepseek_v3



GLM-5 (744B A40B)

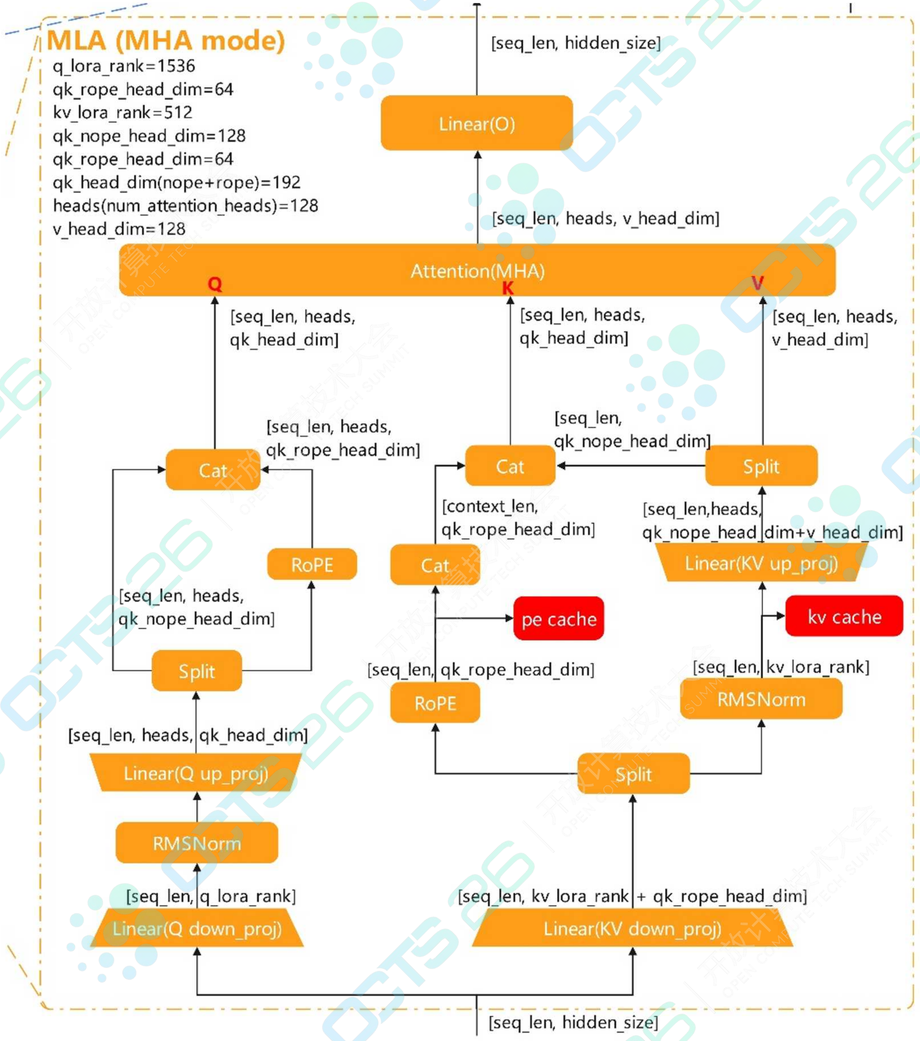
高清图地址：https://github.com/CalvinXKY/InfraTech/tree/main/models/glm_5

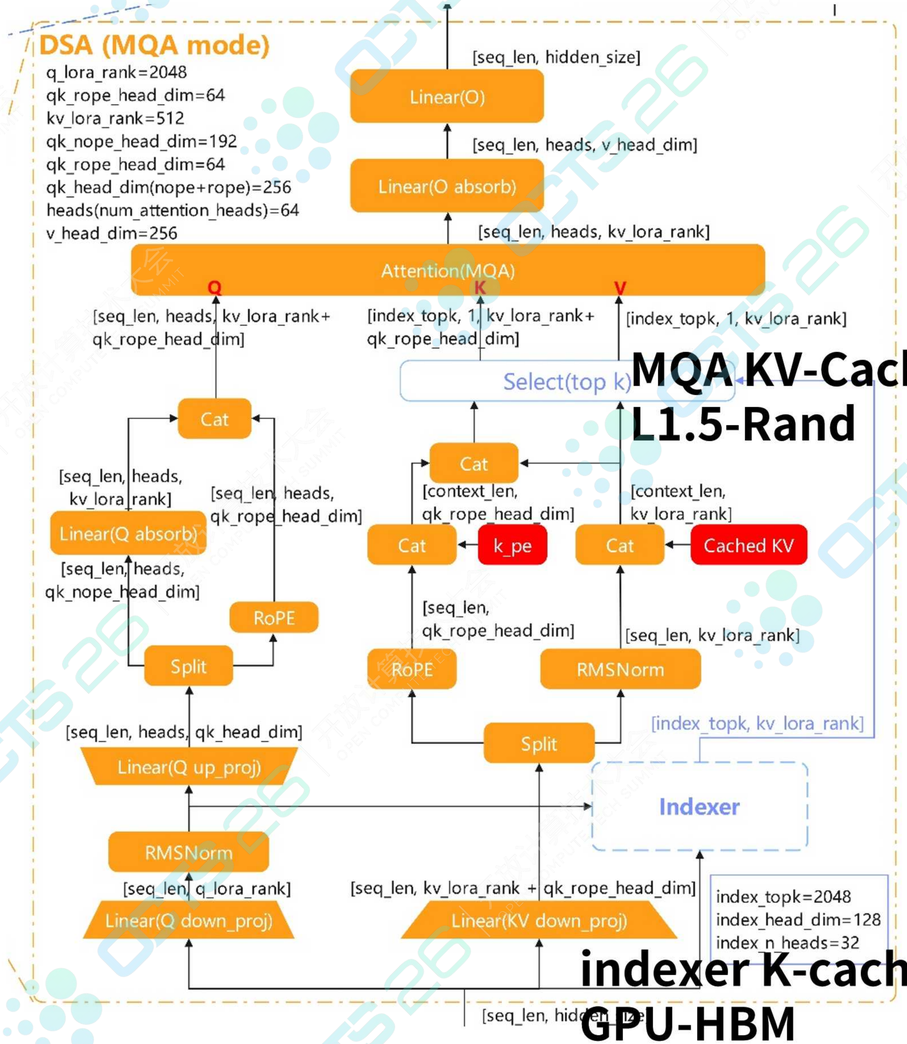


From MLA to DSA (Sparse Attention)

KV_Cache Capacity = S * C * b
KV_Cache Read/token ~= S * (C + kv_lora_rank) * b

KV_Cache Capacity = S * (C + I)
KV_Cache Read/token ~= **2048** * (C + kv_lora_rank) * b

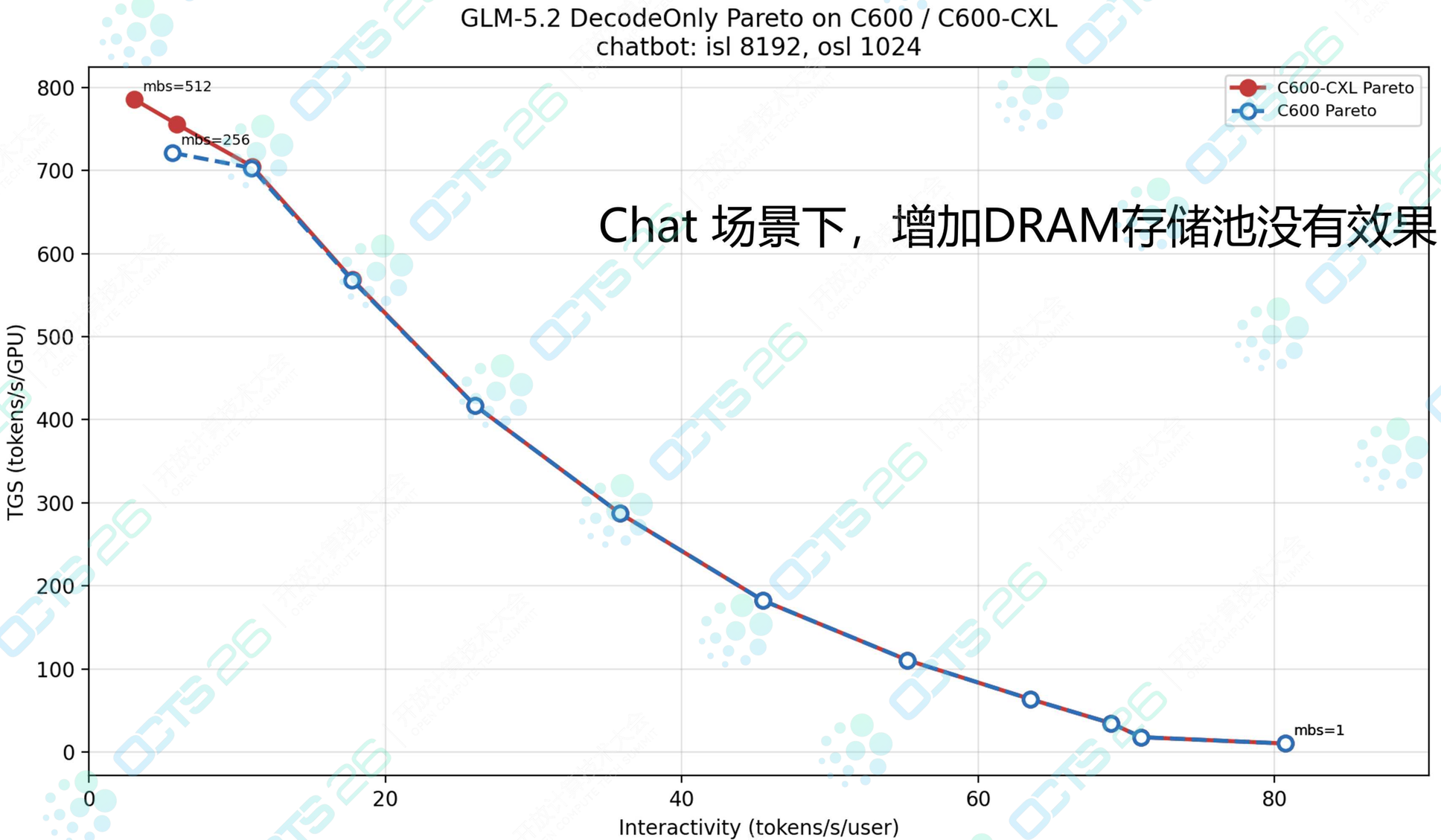




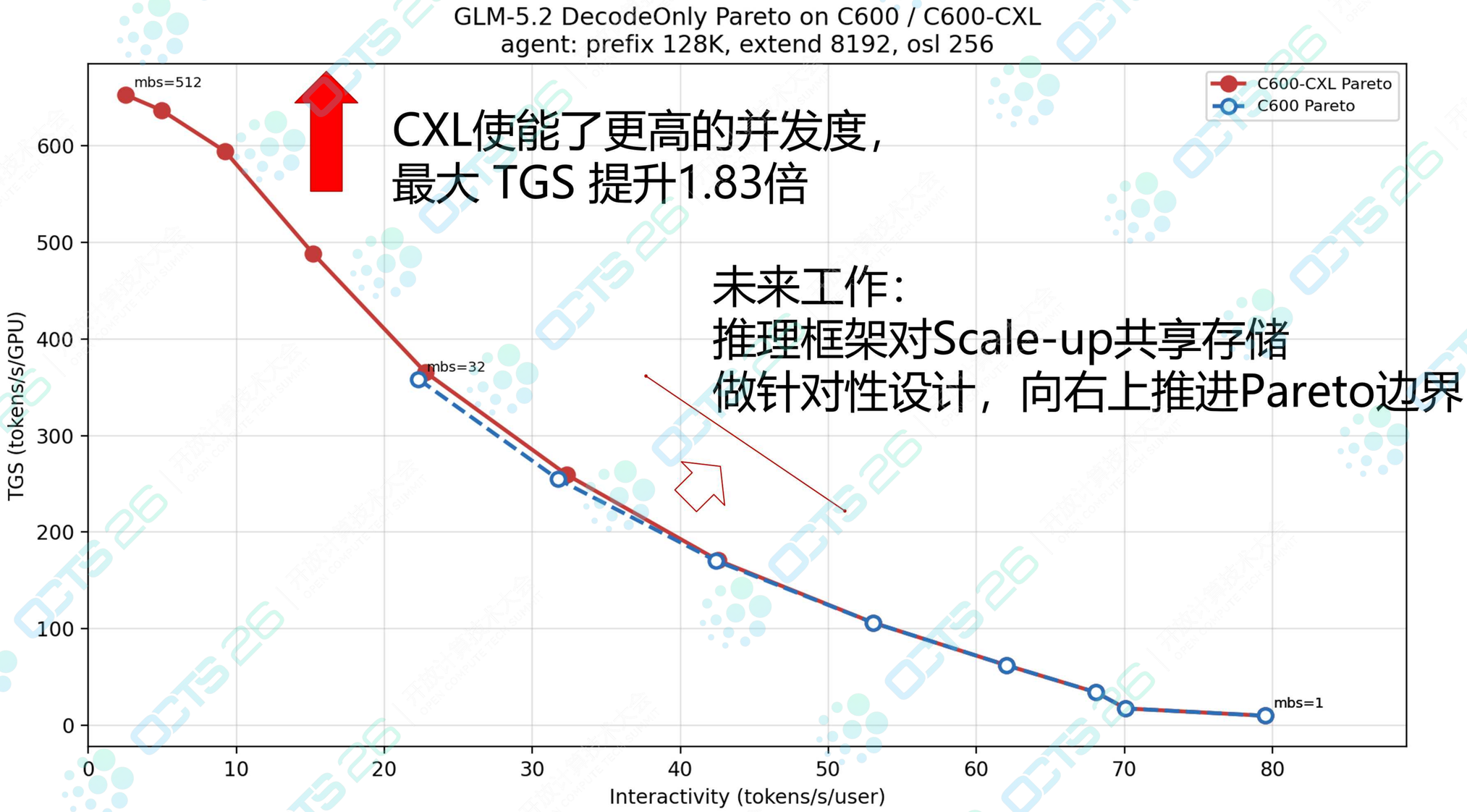
MQA KV-Cache => L1.5-Rand

indexer K-cache => L1 GPU-HBM

POC：Chatbot场景（8192 + 1024 token）



POC：Agent场景 (128K + 8192 + 256 token)



L1.5 Seq-GiDS: (GPU-Initiated Direct Storage)

- 类似 IBGDA，GPU thread直接操作SSD Queue
- 增加interleaved SSD，可以提高带宽
- 小包（64KB），少量GPU Core即可打满PCIe Gen5 x8带宽
- **1TB DDR5 vs. 1TB NAND Flash： 20万元 vs. 1千元**

单机八卡SSD
总容量：4*8T= 32TB
总带宽：4 x 128Gbps

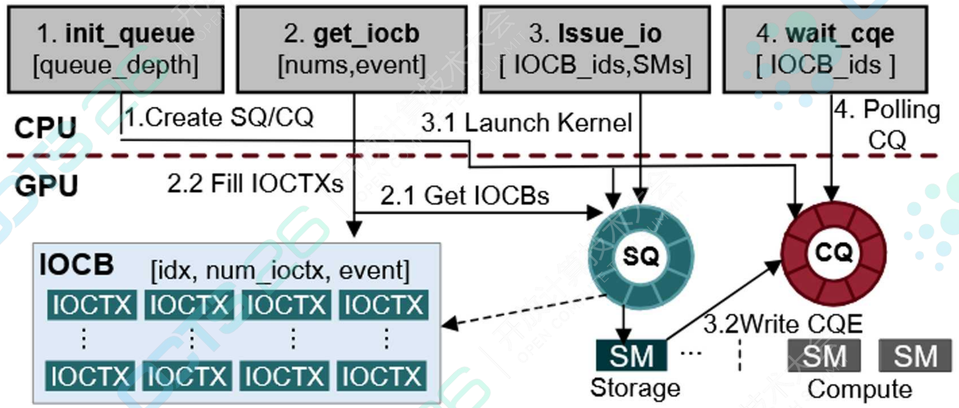
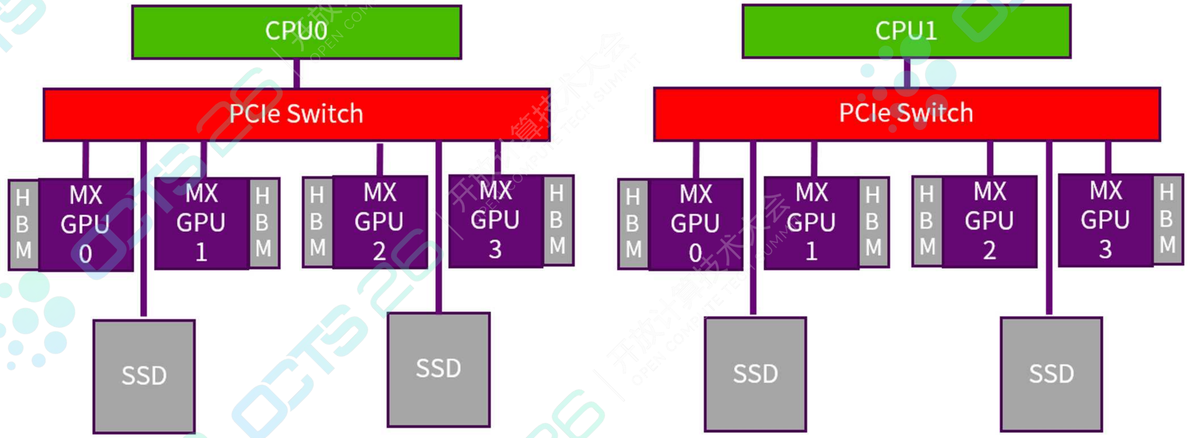
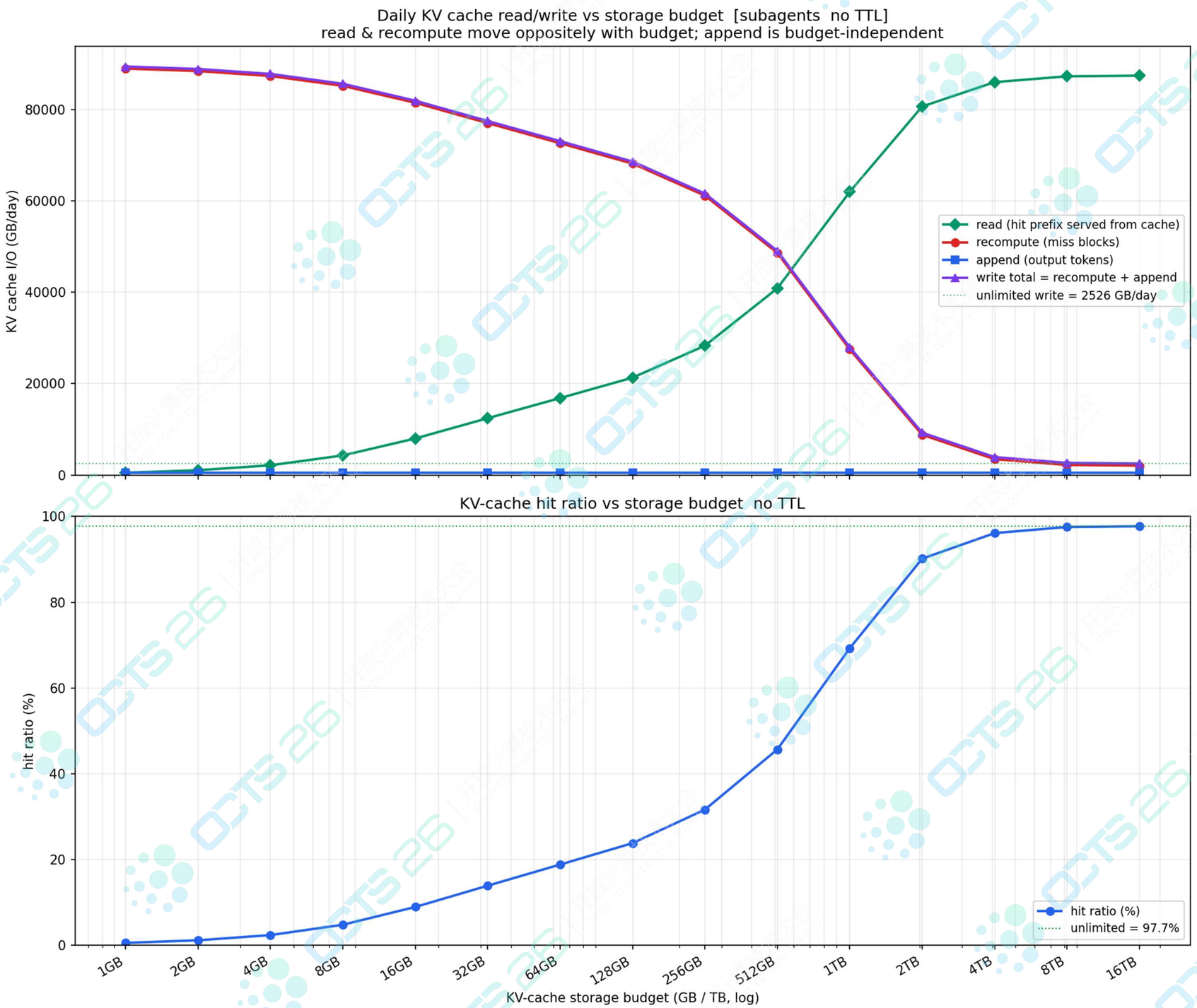


Figure 5. Architecture and I/O Process of GPU io_uring.



[https://arxiv.org/pdf/2605.03375] Tutti: Making SSD-Backed KV Cache Practical for Long-Context LLM Serving



- 使用Weka trace 数据，模拟Claude Code Agent执行过程
  - https://huggingface.co/datasets/semianalysisai/cc-traces-weka-with-subagents-052726-256k
- DSV4-Pro，单节点 4TB 容量，依然有很好的 hit rate 收益
  - 4TB 盘，每日写入2.5TB
- GLM5.2的话，对容量需求会更大

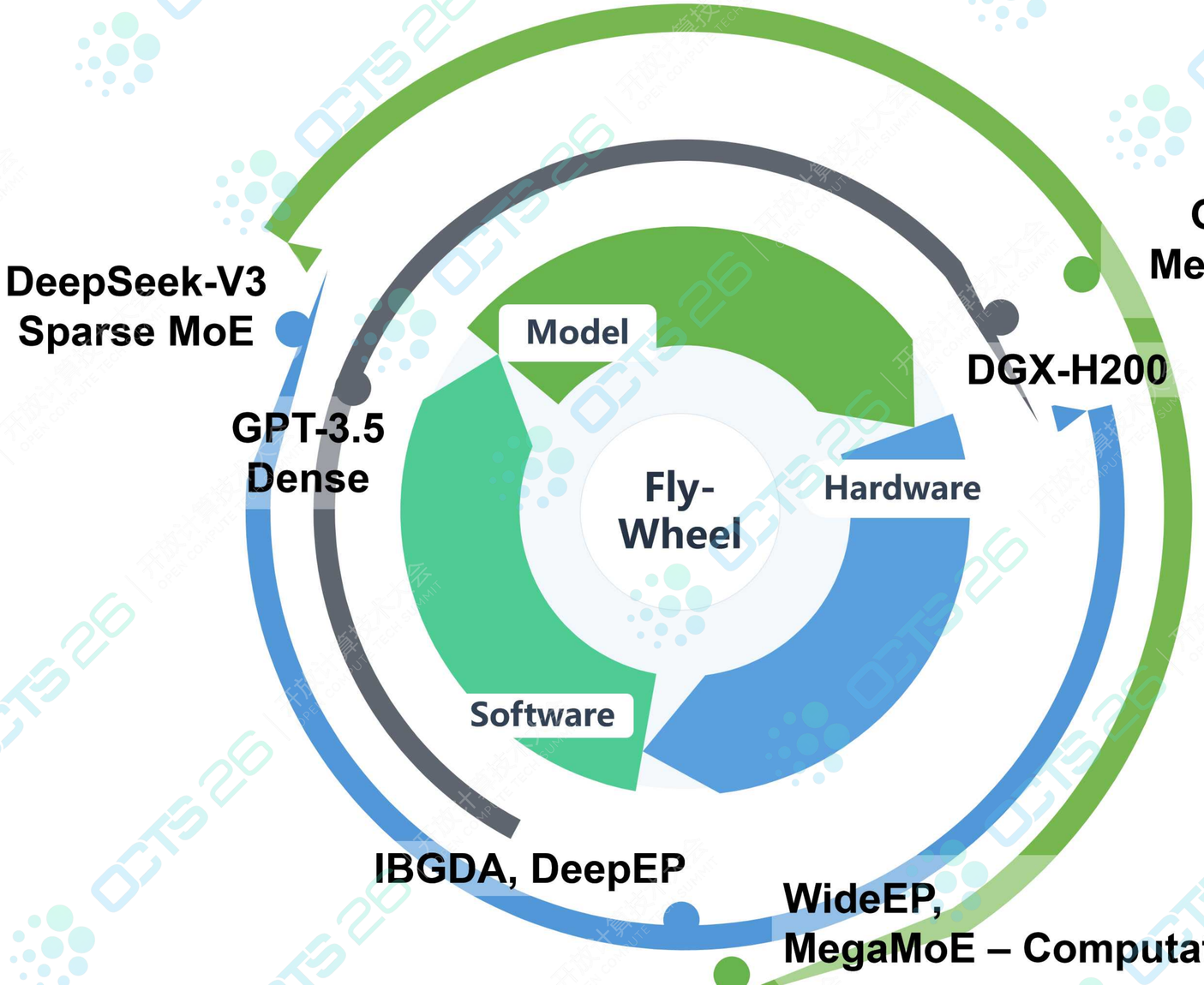
Prefix Hit Rate与$/Token成本的关系

95% -> 97.5% 的hit rate 看起来提升很小，但5%-> 2.5% 的 miss rate提升就非常大了。

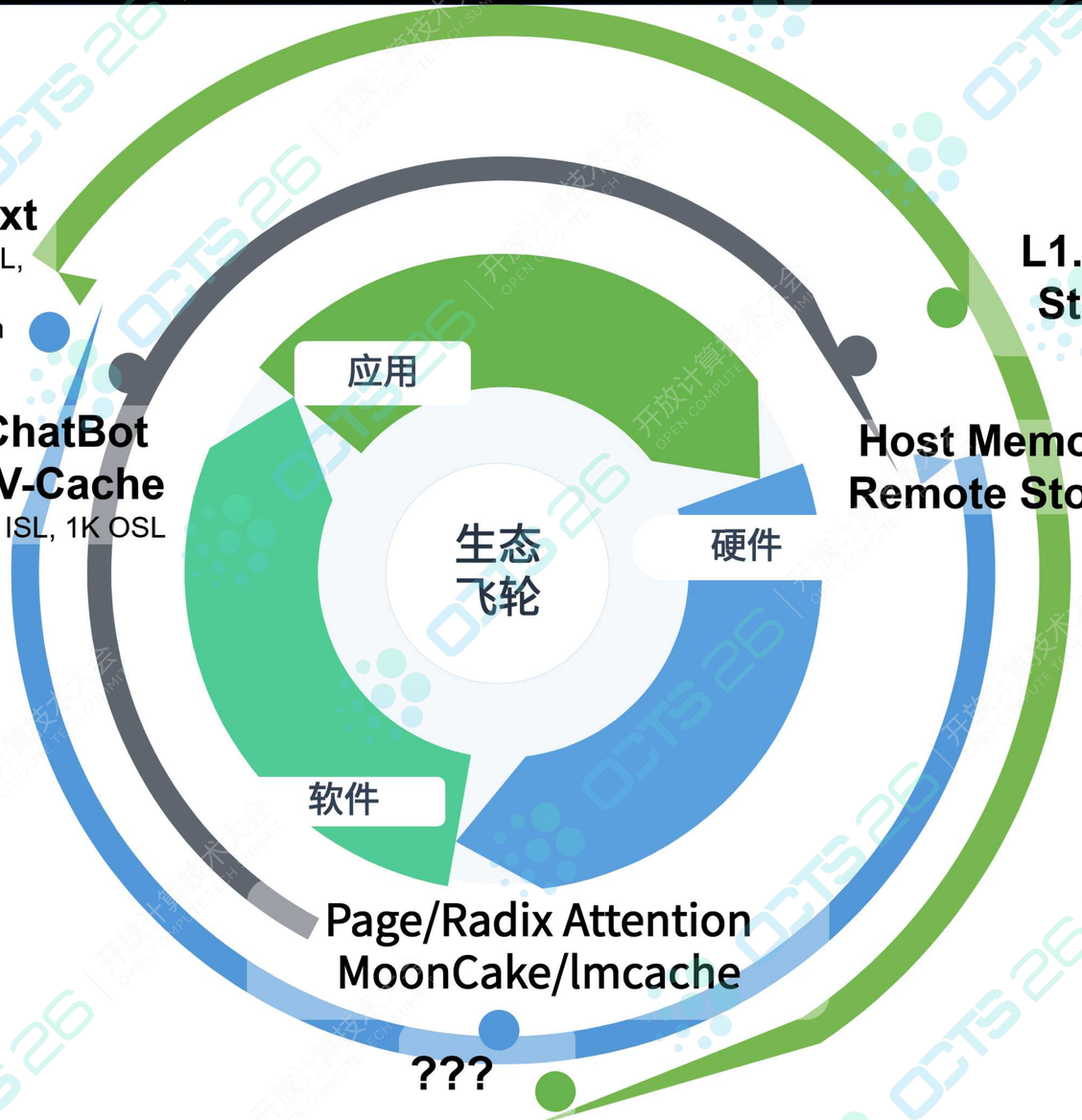
- Miss 的 block，都要prefill做计算。例如1M token，5% miss，就是50K个token做prefill;
- 2.5% miss，只需要25K token 做prefill
- 算力需求减半，大幅度降低请求的成本

当token计费从API付费转向订约付费时，每请求的成本降低，都是净利率的提高。

2025: MoE-超节点时代的生态飞轮



2026: 智能体时代存储架构的生态飞轮



应用场景的演化方向，
软件框架的优化潜力，
决定了硬件方案的价值。

单纯堆砌硬件参数没有价值，
飞轮的目标是降低每Token成本