性　能　测　试　报　告

PERFORMANCE TEST REPORT

AISSD5000 全闪存储用于大模型推理 KV Cache 分层的性能测试报告

（480B 模型·八卡单实例与四卡双实例·长上下文冷恢复）

报　告　编　号	AISSD5000-KVC-PERF-2026-006
版　　本	V1.0
委　托　单　位	深圳中科航星科技有限公司
被　测　设　备	AISSD5000 全闪 NVMe-oF 存储阵列
测　试　类　别	性能基准测试
签　发　日　期	2026 年 07 月 06 日

编制		**日期**	
审核		日期	
批准		日期	

（此处加盖检测机构公章）

声　明

1. 本报告无检测机构公章及骑缝章无效。
2. 本报告无编制人、审核人、批准人签字无效。
3. 本报告涂改、增删无效。
4. 本报告所列结果仅对本次测试所述的被测设备、软件版本、配置及负载条件有效。
5. 本测试为单台服务器环境下的性能基准测试；跨物理节点共享 KV 池、容量弹性扩展等架构能力不在本次测试范围内。
6. 对本报告若有异议，应于收到报告之日起十五日内向本机构提出。

检测机构	
委托单位	深圳中科航星科技有限公司
报告编号	AISSD5000-KVC-PERF-2026-006
签发日期	2026 年 07 月 06 日

报告基本信息

项目	内容
报告标题	AISSD5000 全闪存储用于 LLM 推理 KV Cache 分层的性能测试报告（480B 模型·八卡单实例与四卡双实例·长上下文冷恢复）
报告编号	AISSD5000-KVC-PERF-2026-006
版本	V1.0
检测机构	
委托单位	深圳中科航星科技有限公司
被测设备（DUT）	AISSD5000 全闪 NVMe-oF 存储阵列
测试类别	性能基准测试
密级	内部

修订记录

版本	日期	说明
V1.0	2026-07-06	首次发布：基于 Qwen3-Coder-480B-FP8 超大模型（单会话 KV 约 7.15 GB、上下文约 30K token），在同一台 8 卡服务器上分别以八卡单实例（TP8×1）与四卡双实例（TP4×2）两种生产部署形态，量化 AISSD5000 相对本地 NVMe 与无外存重算的长上下文冷恢复性能，并验证基于 fs:// 共享池的跨实例 KV 热共享能力

1. 执行摘要

本报告在受控的单机（8 卡）环境下，以 **480B 参数量超大模型（Qwen3-Coder-480B-FP8，MoE 架构，权重约 450 GB）** 为负载，量化评估 AISSD5000 全闪存储作为大语言模型（LLM）推理 KV Cache（键值缓存）分层后备介质的性能价值。相比此前 7B / 14B 模型的测试，480B 模型的单会话 KV 体量增至约 **7.15 GB**（上下文约 30K token），是生产环境中长历史会话、超大模型的真实负载压力代表。

本轮测试的核心特色是在同一台 8×AMD Instinct MI308X 服务器上，覆盖两种主流生产部署形态：**八卡单实例（TP8×1）**——单个大实例、单请求由 8 卡并行读取分片，追求单请求最低延迟；**四卡双实例（TP4×2）**——一机切两个独立中实例、独立容错与弹性伸缩，是生产上更常见的多实例切分。负载统一为"长上下文会话冷恢复"（模拟服务重启、会话迁移、不活跃会话唤醒的恢复风暴），在完全一致的负载与测量口径下，对比三方后备介质（AISSD5000 / 本地 NVMe / 无外存重算）并扫描并发梯度；四卡双实例形态额外验证了基于 LMCache 内置 `fs://` 连接器的**跨实例 KV 热共享**。

主要结论（各组请求全部成功、磁盘组逐请求物理冷读取证成立）：

1. **相比本地 NVMe（核心结论）：AISSD5000 首字延迟（TTFT）降低 25%–32%、输出吞吐提升 35%–40%，且该收益跨部署形态稳定成立。**八卡单实例形态并发 16 档，TTFT p50 由 17.31 s 降至 **11.85 s（降 32%）**、输出吞吐由 53.6 提升至 **74.9 tok/s（升 40%）**；四卡双实例形态全机并发 16 档，TTFT p50 由 19.03 s 降至 **13.38 s（降 30%）**、聚合吞吐由 49.7 提升至 **66.9 tok/s（升 35%）**。机理由 `iostat` 物理读实测钉死：同一负载下本地盘所有档位带宽全部钉死在其物理上限 **6.78 GB/s**，AISSD5000 各档峰值均达 **10.2–10.4 GB/s**（单口 100 GbE 线速的 90%+）并在高并发时持续供给 8.6–9.6 GB/s——首字延迟差距即介质供给能力差距。收益不依赖"一机切几个实例"，是介质本身的能力。
1. **相比无外存重算：首字延迟快 8.6–20 倍、输出吞吐高 17–21 倍、解码速度好 75 倍以上。**八卡单实例并发 16 档重算 TTFT p50 149.5 s、吞吐 4.1 tok/s；四卡双实例并发 16 档重算 TTFT p50 114.8 s、每 token 解码间隔（TPOT）达 1.8 s（正常仅 24 ms，解码被持续预填充挤占、输出流近乎冻结）。480B 级模型对 30K 长前缀的重复预填充在工程上不可用——**长上下文超大模型的会话恢复没有"重算"这个选项，KV 存储层是刚需**；且实例算力切得越小（TP4 为 TP8 之半），重算越不可用，KV 外置层越是多实例部署的前提设施。
1. **并发扩展性：AISSD5000 与本地盘的饱和拐点相差约 4 倍。**本地盘在并发 8 之前即饱和（此后加并发吞吐零增长、延迟翻倍）；AISSD5000 供给随并发持续上行，至并发 32 才接近饱和。八卡单实例形态下 AISSD5000 最优吞吐工作点（并发 16、74.9 tok/s）较本地盘最优值（约 54 tok/s）高 40%。会话越长、并发越高，本地盘排队越深，AISSD5000 的领先越稳固。
1. **跨实例 KV 热共享成立且零性能代价（四卡双实例专项验证）。**采用 LMCache 内置 `fs://` 共享池后端，两个独立实例挂同一阵列目录：实例 B 交叉冷读实例 A 灌入的会话，**32/32 全量物理命中、无一退化重算**，交叉读相对读自身仅慢 3%–8%，且仍比本地盘读自身快 22%。这使"一次预填充、全机复用""会话跨实例自由迁移""实例重启不丢缓存"成为现实——建立在共享介质之上的能力，本地盘方案在语义上无法提供。
1. **供给能力被真实负载充分利用且可线性扩展。**本轮已将现有配置（4 盘 RAID0 + 单口 100 GbE）压至上限的 90%+（瓶颈在网络口而非盘）；AISSD5000 具备 6 个 100 G 端口与 24 盘位，增配端口/盘

位即可线性扩展（满配标称 72 GB/s）；本地盘方案带宽（6.78 GB/s 到顶）与容量（2 TB，无法同时容纳 480B 权重与 KV 池）均无扩展余地。

适用边界说明： 本测试为单台 8 卡服务器环境下的性能基准测试；跨物理节点共享同一 KV 池的直接验证（需第二台接入存储网络的主机）列为后续工作。

2. 测试目的与范围

2.1 测试目的

回答以下决策问题：

1. 在 480B 超大模型、长上下文（30K token）会话冷恢复负载下，AISSD5000 承接 KV Cache 相比本地 NVMe 与无外存重算，能带来多大的首字延迟与吞吐收益？
2. 该收益在"八卡单实例"与"四卡双实例"两种生产部署形态下是否同样成立？部署形态如何影响存储的价值兑现？
3. 随并发升高，AISSD5000 与本地盘的供给能力与饱和行为如何演化？
4. 多实例部署下，能否让多个独立实例共享同一份外置 KV 池（跨实例热共享），从而"一次预填充、全机复用"？

2.2 测试范围

- **范围内：** 单台 8 卡服务器环境下，480B 模型长上下文冷恢复负载的三方介质对照（AISSD5000 / 本地 NVMe / 无外存重算）；八卡单实例（TP8×1）与四卡双实例（TP4×2）两种部署形态；并发梯度扫描（8 / 16 / 32）；基于 `fs://` 共享池的跨实例 KV 热共享验证；逐请求 TTFT（p50/p90/p99）、TPOT、聚合吞吐与物理读带宽的量化。
- **范围外：** 跨物理节点共享 KV 池的物理验证、GPUDirect 直读路径（本轮采用 host-staging 异步加载路径）、真实业务偏斜流量——列为后续工作（§9）。

3. 术语与指标定义

指标	定义	优劣方向
KV Cache	LLM 推理 prefill 阶段产生的键值中间结果，可复用以跳过重复计算	—
部署形态	TP8×1：8 卡张量并行组成 1 个实例；TP4×2：两组 4 卡各成 1 个独立实例	—
长上下文冷恢复	会话 KV 已被逐出显存与内存、须从存储层物理读回的恢复场景	—
TTFT（首字延迟）	从提交请求到收到首个输出 token 的时间	越低越好
TPOT（每 token 解码间隔）	逐请求 (端到端时延 − TTFT) ÷ (输出 token 数 − 1)	越低越好

聚合输出吞吐（tok/s）	单位时间产出的输出 token 数；双实例口径为全机输出 token 总数 ÷ 较慢实例整批耗时	越高越好
p50 / p90 / p99	延迟的第 50/90/99 百分位（尾延迟衡量最差体验）	越低越好
全机并发	同时在飞（in-flight）的请求总数；双实例 = 2 × 每实例并发	—
need-to-load	LMCache 日志中需从下层（磁盘）加载的 token 数；>0 表示非 GPU/内存命中	—
物理读带宽	`iostat` 实测块设备读取速率（测量前已清页缓存，反映真实磁盘读）；峰值 / 忙窗均值（>0.5 GB/s 窗口）	越高越好
跨实例热共享	一个实例灌入的 KV，另一独立实例可直接命中复用	—
重算（recompute）	无外部 KV 存储时，对历史前缀重新执行 prefill 计算	性能下限

> 说明：延迟类指标统一采用 OpenAI 兼容流式接口、temperature=0；新版 LMCache 默认异步加载，其日志 `Retrieved ... throughput` 为暂存拷贝速度，**不能**用于判断物理读盘，物理读一律以 `iostat` 为准。

4. 被测系统（System Under Test）

4.1 SUT 边界

被测对象为"推理服务 + KV 分层存储"的端到端系统；受控变量为 **KV Cache 后备介质与部署形态**。推理引擎、模型、采样参数、会话构造、并发档位在各组间保持一致。

4.2 硬件与被测设备

组件	配置
GPU	8 × AMD Instinct MI308X（每卡 192 GB HBM，gfx942）
CPU / 内存	2 × AMD EPYC 9654（384 线程），约 1.5 TB 内存
被测设备（DUT）	**AISSD5000 全闪存储，4 盘组 RAID0（`/dev/md0`，xfs，14 TB），经 NVMe-oF / RoCEv2 接入，单口 100 GbE，挂载于 `/mnt/ws5000`**
对照存储	本地 NVMe 单盘（`/dev/nvme1n1`，PCIe Gen4，挂 `/srv2`，2 TB）；无后备（重算）
网络	100 GbE，RoCEv2

4.3 软件栈与版本

组件	版本
操作系统	Ubuntu 22.04，内核 6.8.0-124-generic
GPU 栈	ROCm 7.2（gfx942）

推理引擎	vLLM 0.20.1+rocm721
KV 缓存库	LMCache（上游主线 2026-06-29 源码编译，默认异步磁盘加载 + 磁盘并行读优化）
模型	Qwen3-Coder-480B-FP8（MoE，权重约 450 GB，各轮均从 AISSD5000 阵列加载）
关键运行参数	`--tensor-parallel-size {8 或 4} --enable-expert-parallel`、`--max-model-len 32768`、`--gpu-memory-utilization 0.9`、`--no-enable-prefix-caching`（强制冷读取证）、`LMCACHE_CHUNK_SIZE=256`、`PYTHONHASHSEED=0`

4.4 工作负载与容量基准

- 模型为 Qwen3-Coder-480B-FP8（MoE 架构，专家并行 EP）；单会话系统提示前缀约 **29.8K token**（reps=530），**单会话 KV 约 7.15 GB**——为 14B 模型（1.55 GB/8.4K token 会话）的约 4.6 倍，负载压力贴近生产超大模型长历史会话场景。
- 八卡单实例形态每轮灌入 **34 个互异会话**（实测落盘约 243 GB）；四卡双实例形态由两实例并行灌入 **48 个互异会话**（A 灌 0–23、B 灌 24–47，落盘约 343 GB）。工作集远超显存驻留与 CPU 中转层容量，保证测量会话必然溢出至存储层。
- 测量：单波 N=并发数，每会话读取一次，decode=64，temperature=0。八卡单实例并发档 8/16/32；四卡双实例全机并发档 16（8+8）/ 32（16+16），与单实例并发口径对齐。

4.5 部署形态说明

- **八卡单实例（TP8×1）**：8 卡张量并行 + 专家并行组成单个 480B 服务实例，单请求的 KV 由 8 卡各持 1/8 分片并行读写。追求单请求最低恢复延迟，代表"单一大服务"部署。
- **四卡双实例（TP4×2）**：卡 0–3 组成实例 A（端口 8000）、卡 4–7 组成实例 B（端口 8001），各为独立 480B 服务，各自 4 卡 TP + EP。代表生产上更常见的"一机多服务"切分（更高实例级容错、独立扩缩容、单实例故障不整机停服）。两实例错峰启动、并行从阵列加载权重（合计约 900 GB），4.5 分钟双双就绪。

5. 测试方法学

5.1 测试设计

实验一（八卡单实例 TP8×1）——三方 × 并发梯度：

组	KV 后备介质	并发档位	iostat 监控
① AISSD5000	`/mnt/ws5000/lmcache480`（md0，RAID0）	8 / 16 / 32	`/dev/md0`
② 本地 NVMe	`/srv2/lmcache480_local`（nvme1n1，单盘）	8 / 16 / 32	`/dev/nvme1n1`
③ 重算（无外存）	无	16	`/dev/md0`（应≈0）

实验二（四卡双实例 TP4×2）——四组 × 两档全机并发：

组	KV 后备配置	全机并发	说明

④ AISSD5000 • 独立池	`LMCACHE_LOCAL_DISK`（md0）	16 / 32	LocalDiskBackend，每实例进程内索引
⑤ AISSD5000 • fs 共享池	`LMCACHE_REMOTE_URL=fs://…`（md0）	16 / 32（含交叉读）	FSConnector，索引即文件系统，双实例共享一个池
⑥ 本地 NVMe	`LMCACHE_LOCAL_DISK`（nvme1n1）	16 / 32	本地单盘
⑦ 重算（无外存）	不挂 LMCache	16 / 32	纯 GPU 重算基线

> 各组仅 KV 后备介质/配置不同；部署形态、模型、权重来源、引擎参数、会话构造、并发档位完全一致。
> fs:// 共享池 URL 需写 `fs://local:0/path` 占位形式——该版本 `parse_remote_url()` 强制校验 host:port，按官方文档写 `fs:///path` 会导致远端后端建连失败、KV 静默丢弃（本轮已发现并规避）。

5.2 物理冷读保证（多重控制）

1. `--no-enable-prefix-caching`：显存不保留任何前缀 KV；
2. LMCache CPU 中转层压至 4 GB（远低于单会话 7.15 GB）：内存层无法容纳任何会话；
3. **每档测量前宿主执行** `sync; echo 3 > /proc/sys/vm/drop_caches`：清除 1.5 TB 主机内存页缓存；
4. 每会话仅读取一次（双实例形态下两实例读取集合不相交，无跨实例页缓存搭车）；
5. 双重取证：磁盘组逐请求满额命中（`hit tokens ≈ 30208`）且 `need to load > 0`，`iostat` 物理读体量与工作集吻合；重算组进程无 LMCache 且盘读≈0。

5.3 测量与数据采集

- 客户端经 OpenAI 兼容流式接口测量逐请求 TTFT（p50/p90/p99/均值）与 TPOT；聚合输出吞吐 = 输出 token 总数 ÷ 整批耗时（双实例取较慢实例整批耗时）；
- 每档并行采集 `iostat -x 1`（峰值、忙窗均值、忙窗时长）；
- 每组切换彻底清理推理进程（含 EngineCore/Worker 残留）并确认 8 卡显存归零。

5.4 方法学控制（防数据污染）

1. **强制物理读取证**：磁盘组全部请求满额命中且 need>0，无一 GPU/内存命中；重算组 need=0 且盘读≈0，反证纯重算；
2. **单一变量校验**：每次启动核验环境变量（磁盘目录 / 共享 URL / TP 并行度）与 LMCache 配置日志回显；
3. **复现性**：四卡双实例全指标数据经当日两次独立完整测量，各档偏差 ≤5%；
4. **正确性**：各组请求全部成功返回，同一提示词、temperature=0、相同 decode 上限，输出 token 计数一致。

6. 测试结果

6.1 实验一：八卡单实例（TP8×1）三方对照总表

测量条件：480B 模型，每会话约 30K token / KV 7.15 GB，单波 N=并发数，decode=64，磁盘组已清页缓存、逐请求物理命中。

档位	TTFT p50（s）	TTFT p90（s）	输出吞吐（tok/s）	盘峰值（GB/s）	盘忙窗均值（GB/s）	冷读取证
AISSD5000·并发8	**7.53**	7.54	**56.6**	10.26	7.49	8/8
AISSD5000·并发16	**11.85**	11.87	**74.9**	10.20	8.12	16/16
AISSD5000·并发32	**26.35**	26.37	**71.6**	10.19	9.29（持续25s）	32/32
本地NVMe·并发8	10.17	10.18	43.9	6.78	5.99	8/8
本地NVMe·并发16	17.31	17.32	53.6	6.78	6.17	16/16
本地NVMe·并发32	35.73	35.75	53.9	6.78	6.42	32/32
重算·并发16	149.48	237.34	4.1	≈0	—	need=0（纯重算）

6.2 实验二：四卡双实例（TP4×2）全指标总表

测量条件：480B 模型，每会话约 30K token / KV 7.15 GB，双实例同时发起，全机口径（A+B 合并样本），decode=64，磁盘组已清页缓存、逐请求物理命中。

档位	全机并发	TTFT p50（s）	TTFT p90（s）	TTFT p99（s）	TPOT p50（ms）	聚合吞吐（tok/s）	盘峰值（GB/s）	盘忙窗均值（GB/s）
AISSD5000·独立池·16	16	**13.38**	13.78	13.79	24.2	**66.9**	10.32	8.56
AISSD5000·独立池·32	32	**25.62**	26.24	26.25	32.0	**72.6**	10.41	9.59
AISSD5000·fs共享·读自身·32	32	24.83	26.03	26.04	30.7	73.4	10.36	9.36

AISSD5000 • fs 共享 • 交叉读 • 32	32	26.81	26.83	26.84	31.6	71.1	10.38	9.22
本地NVMe • 16	16	19.03	19.04	19.04	24.1	49.7	6.79	6.31
本地NVMe • 32	32	34.48	36.44	36.44	30.9	53.3	6.78	6.45
重算 • 16	16	114.78	210.41	268.52	1817.4	3.8	≈0	—
重算 • 32	32	251.69	466.95	608.52	5244.0	3.4	≈0	—

> 交叉读 = 实例 A 读实例 B 灌入的会话、实例 B 读实例 A 灌入的会话（32/32 全量物理命中）；读自身 = 各读自己灌入的会话。

6.3 灌入阶段数据

形态 / 组	灌入会话数	耗时（s）	落盘量
八卡单实例 • AISSD5000	34	约 200	AISSD5000 约 243 GB
八卡单实例 • 本地盘	34	约 200	本地盘约 243 GB
四卡双实例 • AISSD5000（并行灌）	48（A24+B24）	361 – 365	AISSD5000 约 343 GB
四卡双实例 • 本地盘（并行灌）	48	358 – 363	本地盘约 343 GB
四卡双实例 • fs 共享池（并行灌）	48	363	AISSD5000 约 344 GB / 22656 文件
重算组	—（无需灌入）	—	0

> 双实例并行灌入 343 GB 仅 363 s（≈0.95 GB/s 持续写 + 双引擎 prefill 读写混合），WS 轮与本地轮灌入耗时几乎相同——写入需求远低于两种介质上限，灌入为算力瓶颈，不影响对照公平性。

6.4 核心对比一：AISSD5000 vs 本地 NVMe（两种部署形态）

八卡单实例（TP8×1）：

并发	本地 TTFT p50	AISSD5000 TTFT p50	TTFT 降低	本地吞吐	AISSD5000 吞吐	吞吐提升
8	10.17 s	7.53 s	−26%	43.9	56.6	+29%
16	17.31 s	11.85 s	−32%	53.6	74.9	+40%
32	35.73 s	26.35 s	−26%	53.9	71.6	+33%

四卡双实例（TP4×2，全机口径）：

全机并发	本地 TTFT p50	AISSD5000 TTFT p50	TTFT 降低	本地聚合吞吐	AISSD5000 聚合吞吐	吞吐提升
16	19.03 s	13.38 s	−30%	49.7	66.9	+35%

32	34.48 s	25.62 s	−26%	53.3	72.6	+36%

> 解读：两种部署形态下结论高度一致（TTFT 降 26%–32%、吞吐升 29%–40%）——**AISSD5000 的收益不依赖"一机切几个实例"，是介质供给能力的差异**。同一负载需求下，本地盘所有档位物理读带宽全部钉死在 6.78 GB/s 物理顶（忙窗均值 5.99–6.64 GB/s），AISSD5000 以 8.6–9.6 GB/s 持续供给、峰值 10.2–10.4 GB/s（单口 100 GbE 线速的 90%+）。会话越长、并发越高，本地盘排队越深，AISSD5000 的领先越稳固。

6.5 核心对比二：AISSD5000 vs 无外存重算

形态 / 并发	指标	重算	AISSD5000	收益
八卡单实例·16	TTFT p50	149.48 s	**11.85 s**	**快 12.6 倍**
八卡单实例·16	TTFT p90	237.34 s	**11.87 s**	**快 20.0 倍**
八卡单实例·16	输出吞吐	4.1 tok/s	**74.9 tok/s**	**高 18.3 倍**
四卡双实例·16	TTFT p50	114.78 s	**13.38 s**	**快 8.6 倍**
四卡双实例·16	TTFT p99	268.52 s	**13.79 s**	**快 19.5 倍**
四卡双实例·16	TPOT p50	1817 ms	**24.2 ms**	**好 75 倍**
四卡双实例·16	聚合吞吐	3.8 tok/s	**66.9 tok/s**	**高 17.6 倍**
四卡双实例·32	TPOT p50	5244 ms	**32.0 ms**	**好 164 倍**

> 解读：480B 模型重算 30K 前缀，重算不仅首字慢一个数量级，**解码也被拖垮**——持续的预填充挤占解码批次，每 token 解码间隔（TPOT）达 1.8–5.2 秒（正常 24–32 ms），首 token 之后的输出流同样不可用。四卡双实例形态下 TP4 实例算力为 TP8 之半，重算劣化更深——**实例切得越细，KV 外置层越是刚需**。长上下文超大模型的会话恢复没有"重算"这个选项。

6.6 核心对比三：并发扩展性（饱和拐点）

盘供给能力随并发的变化（八卡单实例，忙窗均值）：

并发	AISSD5000	本地 NVMe
8	7.49 GB/s	5.99 GB/s
16	8.12 GB/s	6.17 GB/s
32	**9.29 GB/s（仍在上行）**	6.42 GB/s（钉死于物理顶）

三点结论：

1. **本地盘的饱和拐点在并发 8 之前即到达**：三档带宽 5.99→6.17→6.42 GB/s 始终贴着 6.78 GB/s 物理顶；此后加并发全部转化为排队——并发 16→32 吞吐零增长（53.6→53.9），TTFT 却翻倍（17.3→35.7 s）。
2. **AISSD5000 的饱和拐点在并发 32 附近才出现**：供给随并发持续上行，至并发 32 达峰值的 91%，吞吐才轻微回落。**两种介质的饱和拐点相差约 4 倍。**

3. **最优工作点**：本地盘方案最佳吞吐约 54 tok/s（并发 8 即到顶）；AISSD5000 方案最优工作点为并发 16 / **74.9 tok/s（高 40%）**。生产建议：本配置下长上下文恢复的并发预算按 16–24 配置，AISSD5000 可稳定承接；本地盘方案在并发 8 以上即应限流。

6.7 核心对比四：跨实例 KV 热共享（四卡双实例 fs:// 共享池专项）

两实例挂同一阵列共享池，交叉冷读对方灌入的会话，与读自身同口径对比（全机并发 32 档）：

对比	TTFT p50	TTFT p99	TPOT p50	聚合吞吐	盘忙窗均值	命中取证
独立池·读自身	25.62 s	26.25 s	32.0 ms	72.6 tok/s	9.59 GB/s	32/32 满命中
fs 共享池·读自身	24.83 s	26.04 s	30.7 ms	73.4 tok/s	9.36 GB/s	32/32 满命中
fs 共享池·交叉读对方	**26.81 s**	**26.84 s**	**31.6 ms**	**71.1 tok/s**	9.22 GB/s	**32/32 满命中、零未命中**

> 解读：三点结论。其一，**共享后端零性能代价**——fs 共享池读自身与独立后端各项指标在噪声内一致。其二，**跨实例共享近零代价**——交叉读比读自身仅慢 8%（p50）/ 3%（p99），32 个交叉请求全部满额物理命中（每请求 hit tokens = 30208），无一退化重算。其三，**跨实例读别人灌入的，仍比本地盘读自己的快 22%**（26.81 vs 34.48 s）——共享语义叠加介质优势。对照 LocalDiskBackend 的跨实例尝试（进程内索引，交叉读 hit=0、退化重算、TTFT 67 s），共享能力由后端选择决定，`fs://` 一步打通。

> 附带工程说明：`fs://` 远端路径的 CPU 中转层需 ≥ 并发在途 KV 体量。本轮 4 GB 中转层在低并发下曾触发"分配失败→阻塞超时取消重试"导致个别请求解码发散（经 5 轮复测定位为配置问题而非偶然）；将 `LMCACHE_MAX_LOCAL_CPU_SIZE` 提至 64 GB 后告警归零、连续多轮真冷读全部干净。生产配置建议中转层 ≥ 并发在途 KV 体量（本负载即 ≥57 GB）。

7. 分析与讨论

7.1 收益的稳健性：跨部署形态一致

两种部署形态（TP8×1、TP4×2）在同一台机器、同一负载规格下独立完成，AISSD5000 对本地 NVMe 的收益幅度高度一致（TTFT −26%~−32%、吞吐 +29%~+40%）。机理相同：30K 长上下文冷恢复的瞬时带宽需求越过本地盘物理顶（6.78 GB/s），差距即两介质供给能力之差。**该结论对部署形态、实例粒度不敏感，可直接外推到"一机 N 实例"的生产切分。**

7.2 部署形态对存储供给的影响

两种形态下 AISSD5000 峰值均为 10.2–10.4 GB/s（单口线速的 90%+）、本地盘均钉死 6.78 GB/s——阵列既能被一个 TP8 实例的 8 路分片并行读打满，也能被两个独立 TP4 实例的并发读打满，且双实例供给均衡（两实例 TTFT p50 差 ≤1.8 s）。同并发下 TP8 单实例单请求恢复略快（每请求由 8 卡并行读分片），并发 32 时两形态几乎重合。**形态选择可完全基于业务需要（容错/弹性 vs 单请求延迟），存储不构成约束。**

7.3 重算的形态敏感性

KV 恢复从 TP8 换到 TP4×2 只慢约 2%–14%，而重算并发 16 档从 TP8 的 149.5 s 到 TP4×2 单实例的 114.8 s（并发 32 档达 251.7 s、TPOT 5.2 s）——重算耗时随实例算力切分显著劣化。生产趋势恰是往多实例/小实例切（容错与弹性），这使 KV 外置层从"优化项"变为"部署前提"。

7.4 跨实例热共享的架构价值

`fs://` 共享池的核心优势在于其索引即文件系统本身（`exists()` 直接 stat 文件，无进程内状态），天然跨实例、跨重启可见，且写入走"临时文件+原子改名"，读方永不读到半个文件。由此带来三项本地盘方案在语义上无法提供的能力：**预填充成本全机只付一次**（任何实例处理过的长前缀/历史会话，其余实例即时命中）；**会话可跨实例自由迁移**（负载均衡无需会话亲和约束，任一实例都能以 13–27 s 而非重算的 115–252 s 恢复）；**实例滚动重启不清空缓存**（池在阵列上、索引即文件系统）。

7.5 供给能力的扩展路径

本轮已把"4 盘 RAID0 + 单口 100 GbE"配置压到网络口上限（10.2–10.4 GB/s，瓶颈在网络口而非盘，4 盘各自仍有余量）。AISSD5000 具备 6 个 100 G 端口与 24 盘位，增配第二端口即可把上限提升至约 20 GB/s，满配标称 72 GB/s——供给能力随业务增长线性扩展；本地盘方案带宽（6.78 GB/s 到顶）与容量（2 TB，测试期间已无法同时容纳 480B 权重与 KV 池，权重实际由 AISSD5000 承载）均无扩展余地。

8. 结论

1. **相比本地 NVMe（核心结论）**：480B 长上下文冷恢复负载下，AISSD5000 使**首字延迟降低 26%–32%、输出吞吐提升 29%–40%**，且该收益在八卡单实例与四卡双实例两种部署形态下同样成立；机理为本地盘钉死于 6.78 GB/s 物理上限，AISSD5000 以 8.6–9.6 GB/s 持续供给（单口线速 90%+）。
2. **相比无外存重算**：首字延迟快 8.6–20 倍、输出吞吐高 17–21 倍、解码速度好 75 倍以上；重算在首字与解码两端同时不可用，KV 外置层是长上下文超大模型多实例部署的前提设施，且实例切得越细该结论越强。
3. **并发扩展性**：本地盘并发 8 前即饱和（此后加并发零吞吐收益、延迟翻倍），AISSD5000 至并发 32 才接近饱和——**饱和拐点相差约 4 倍**；最优吞吐工作点高 40%。
4. **跨实例热共享**：基于 `fs://` 共享池，两独立实例交叉冷读 32/32 全量命中、零性能代价（交叉读仅比读自身慢 3%–8%，仍比本地盘读自身快 22%），实现一次预填充全机复用、会话自由迁移、重启不丢缓存。
5. **供给能力被充分利用且可扩展**：真实推理负载已将现有配置压至上限的 90%+（瓶颈为单网络口而非盘），增配端口/盘位可线性扩展（满配标称 72 GB/s）；本地盘带宽与容量均无扩展余地。

9. 局限与后续工作

1. **单机范围**：跨物理节点共享同一 KV 池（多机聚合、一次预填充全集群复用）的直接物理验证需第二台接入存储网络的主机，列为后续工作；`fs://` 共享池的跨节点使用需共享文件系统层（NFS/集群文件系统）承载，块设备直挂多主机不可行。
2. **访问路径**：本轮采用 host-staging 异步加载路径（新版 LMCache 默认），未覆盖 GPUDirect 直读路径在本平台的表现。

3. **合成负载与统计重复**：均匀访问、固定长度合成数据相对真实偏斜流量为保守估计；关键点已做两次独立完整测量（偏差 ≤5%），建议进一步扩大样本给出置信区间。
4. **未覆盖能力**：双 100 G 口聚合、更长上下文（>32K）、跨重启 KV 持久化复用的稳态吞吐，列为后续测试项。

附录 A：复现命令（同事服务器，容器 `vllm` 内）

A.1 八卡单实例（TP8×1）三方对照（本地轮改 `LMCACHE_LOCAL_DISK` 指向 `/srv2/...`；重算轮去掉 LMCACHE 变量与 `--kv-transfer-config`）

```
MODEL=/mnt/ws5000/models/Qwen3-Coder-480B-FP8
docker exec -d vllm bash -c "export HIP_VISIBLE_DEVICES=0,1,2,3,4,5,6,7 VLLM_ROCM_USE_AITER=1 PYTHONHASHSE
ED=0 \
 LMCACHE_CHUNK_SIZE=256 LMCACHE_LOCAL_CPU=True LMCACHE_MAX_LOCAL_CPU_SIZE=4 \
 LMCACHE_LOCAL_DISK=file:///mnt/ws5000/lmcache480 LMCACHE_MAX_LOCAL_DISK_SIZE=1000; \
 vllm serve \$MODEL --served-model-name qwen \
 --tensor-parallel-size 8 --enable-expert-parallel --trust-remote-code \
 --max-model-len 32768 --gpu-memory-utilization 0.9 --no-enable-prefix-caching \
 --kv-transfer-config '{\"kv_connector\":\"LMCacheConnectorV1\",\"kv_role\":\"kv_both\"}' \
 --port 8000 > /mnt/ws5000/vllm_ws.log 2>&1"
# 灌入 34 会话（reps=530≈30K token）→ 清页缓存 → 逐档 conc 8/16/32 测量（见 A.3）
```

A.2 四卡双实例（TP4×2）启动（独立池 / fs 共享池 / 本地盘 / 重算，仅存储环境变量不同）

```
# 独立池:  ENVS="LMCACHE_CHUNK_SIZE=256 LMCACHE_LOCAL_CPU=True LMCACHE_MAX_LOCAL_CPU_SIZE=4 \
#                LMCACHE_LOCAL_DISK=file:///mnt/ws5000/lmcache480tp4 LMCACHE_MAX_LOCAL_DISK_SIZE=1000"
# fs共享池: ENVS="LMCACHE_CHUNK_SIZE=256 LMCACHE_LOCAL_CPU=True LMCACHE_MAX_LOCAL_CPU_SIZE=64 \
#                LMCACHE_REMOTE_URL=fs://local:0/mnt/ws5000/kvpool_fs LMCACHE_REMOTE_SERDE=naive"
# 本地盘:  同独立池但 LMCACHE_LOCAL_DISK=file:///srv2/lmcache480tp4_local
# 重算:    ENVS 为空且去掉 --kv-transfer-config
for I in 0 1; do
  DEVS=$([ $I -eq 0 ] && echo "0,1,2,3" || echo "4,5,6,7"); PORT=$((8000+I))
  docker exec -d vllm bash -c "export HIP_VISIBLE_DEVICES=$DEVS VLLM_ROCM_USE_AITER=1 PYTHONHASHSEED=0 $EN
VS; \
 vllm serve $MODEL --served-model-name qwen \
 --tensor-parallel-size 4 --enable-expert-parallel --trust-remote-code \
 --max-model-len 32768 --gpu-memory-utilization 0.9 --no-enable-prefix-caching \
 --kv-transfer-config '{\"kv_connector\":\"LMCacheConnectorV1\",\"kv_role\":\"kv_both\"}' \
 --port $PORT > /mnt/ws5000/log_i$I.log 2>&1"
  sleep 30
done
```

A.3 灌入、测量与冷读取证

```
# 灌入（双实例并行；单实例则只对 8000 灌 34 会话）
docker exec -d vllm bash -c "python3 /mnt/ws5000/bench_mp.py 8000 populate 530 24 0  > /mnt/ws5000/ppA.log
2>&1"
docker exec -d vllm bash -c "python3 /mnt/ws5000/bench_mp.py 8001 populate 530 24 24 > /mnt/ws5000/ppB.log
2>&1"
# 每档测量（示例：全机并发32=16+16；交叉读档 A/B 起点互换为 24 / 0）
sync; echo 3 | sudo tee /proc/sys/vm/drop_caches
iostat -x 1 400 /dev/md0 > /tmp/io.log &        # 本地盘组监控 /dev/nvme1n1
docker exec -d vllm bash -c "python3 /mnt/ws5000/bench_mp.py 8000 measure 530 16 0  64 16 > /mnt/ws5000/A.
log 2>&1"
docker exec -d vllm bash -c "python3 /mnt/ws5000/bench_mp.py 8001 measure 530 16 24 64 16 > /mnt/ws5000/B.
log 2>&1"
# 冷读取证（每实例测量请求应全部满额磁盘命中）
grep -acE 'hit tokens: 30[0-9]{3}' /mnt/ws5000/log_i0.log
awk '$1=="md0"{if($3>m)m=$3} $1=="md0" && $3>500000{c++;s+=$3} END{printf "峰值%.2f 忙均%.2f GB/s(%ds)\n",
m/1e6, s/c/1e6, c}' /tmp/io.log
```

附录 B：负载客户端 `bench_mp.py`（端口参数化，逐请求 TTFT/TPOT）

用法：`python3 bench_mp.py <port> <mode:populate|measure> <reps> <N> <off> [decode] [conc]`；`reps=530` → 每会话约 30K token。四卡双实例形态下两实例分别对 8000/8001 端口发起，全机指标由两实例逐请求样本合并计算。

```
import urllib.request, json, time, sys
from concurrent.futures import ThreadPoolExecutor
port=sys.argv[1]; mode=sys.argv[2]; reps=int(sys.argv[3]); N=int(sys.argv[4]); off=int(sys.argv[5])
decode=int(sys.argv[6]) if len(sys.argv)>6 else 64
conc=int(sys.argv[7]) if len(sys.argv)>7 else 16
BASE='http://127.0.0.1:%s/v1/chat/completions'%port
basep='背景知识：AISSD5000是国产高性能全闪NVMe-oF存储，可作为大模型推理KV缓存的分层后备介质，配合vLLM与LMCache在显存/内存/磁盘之间分层存取KV。'
def make_prefix(i): return '[sess-%05d] '%i + basep*reps
def req(sid, maxtok):
    body=json.dumps({'model':'qwen','stream':True,'messages':[{'role':'system','content':make_prefix(sid)},{'role':'user','content':'回答%d'%sid}],'max_tokens':maxtok,'temperature':0}).encode()
    st=time.time(); ttft=None; n=0
    try:
        r=urllib.request.urlopen(urllib.request.Request(BASE,data=body,headers={'Content-Type':'application/json'}),timeout=1800)
        for line in r:
            sx=line.decode('utf-8','ignore')
            if sx.startswith('data:') and '"content"' in sx:
                if ttft is None: ttft=time.time()-st
                n+=1
        return (sid, ttft, time.time()-st, n)
    except Exception as e:
        sys.stderr.write('ERR sid=%d %s\n'%(sid,e)); return (sid,None,None,0)
def pct(a,q):
    import math
    return a[min(len(a)-1, max(0, math.ceil(q*len(a))-1))] if a else 0.0
if mode=='populate':
    t0=time.time()
    for i in range(N): req(off+i,1)
    print('[p%s] populate N=%d off=%d wall=%.1fs'%(port,N,off,time.time()-t0),flush=True); sys.exit(0)
res=[]; t0=time.time()
with ThreadPoolExecutor(max_workers=conc) as ex:
    futs=[ex.submit(req,off+i,decode) for i in range(N)]
    for f in futs:
        x=f.result()
        if x[1] is not None: res.append(x)
wall=time.time()-t0
for sid,ttft,e2e,n in sorted(res):
    tp=(e2e-ttft)/(n-1)*1000 if n>1 else 0
    print('REQ sid=%d ttft=%.3f e2e=%.3f ntok=%d tpot_ms=%.1f'%(sid,ttft,e2e,n,tp),flush=True)
tt=sorted(r[1] for r in res); tpots=sorted((r[2]-r[1])/(r[3]-1)*1000 for r in res if r[3]>1); tot=sum(r[3] for r in res)
print('[p%s] n=%d wall=%.1fs TTFT p50=%.2f p90=%.2f p99=%.2f mean=%.2f | TPOT_ms p50=%.1f mean=%.1f | outtok/s=%.1f'%(
  port,len(res),wall,pct(tt,.5),pct(tt,.9),pct(tt,.99),sum(tt)/len(tt),pct(tpots,.5),sum(tpots)/len(tpots),tot/wall),flush=True)
```

附录 C：原始数据文件（同事服务器）

文件	内容
`/tmp/kv32k.out`	八卡单实例（TP8×1）三方对照全过程日志
`/tmp/tp4x2b.out`、`/tmp/tp4x2c.out`、`/tmp/full2.out`	四卡双实例（TP4×2）全指标测量编排日志
`/tmp/fsshare.out`、`/tmp/fsrep*.out`	fs 共享池跨实例热共享验证与发散复测日志
`/mnt/ws5000/results/*_{A,B}.log`	各档两实例逐请求原始数据（REQ 行）与汇总行

`/mnt/ws5000/*_i{0,1}.log`、`/mnt/ws5000/vllm_ws.log`	各服务实例完整日志（含逐请求 hit 取证行、FS connector 初始化行）
`/tmp/io*_*.log`	各档 iostat 物理读带宽秒级时间序列
`/mnt/ws5000/kvpool_fs/`	fs 共享池（344 GB / 22656 文件，保留可复查）

本报告全部数据均来自实测，测试方法、命令与脚本完整留档，可独立复现。