AISSD5000 KV Cache 性能测试报告 —— 480B 模型·八卡单实例·长上下文冷恢复

测试平台：AMD Instinct MI308X ×8（每卡 192 GB HBM） / ROCm 7.2 / vLLM 0.20.1+rocm721 + LMCache v1（上游主线源码编译，含磁盘并行读优化）

被测存储：AISSD5000（WS5000）全闪 NVMe-oF 阵列（4 盘 RAID0，RoCEv2 over 100GbE，XFS，14 TB）

对照存储：服务器本地 NVMe（Solidigm，PCIe Gen4，挂载分区 2 TB）；无外存重算

模型：Qwen3-Coder-480B-FP8（MoE，张量并行 TP=8 + expert parallel，权重约 450 GB）

负载：长上下文冷恢复——每会话约 29.8K token，KV ≈ 7.15 GB/会话

日期：2026-07-05

一、测试目的与结论摘要

在 **8 卡单实例（TP=8）** 这一 480B 级大模型的标准生产部署形态下，以**长上下文会话冷恢复**为负载（模拟 Agent/代码助手的长历史会话恢复风暴：服务重启、会话迁移、不活跃会话唤醒），量化 AISSD5000 作为 KV Cache 分层后备介质相对本地 NVMe 与无外存重算的性能收益，并给出并发扩展曲线。

主要结论（全部档位逐请求物理冷读取证成立）：

1. **相比本地 NVMe 硬盘：AISSD5000 首字延迟（TTFT）降低 26%–32%，输出吞吐提升 29%–40%，并发 8–32 全区间稳定成立**。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%）**。机理由 `iostat` 物理读实测钉死：同一负载下本地盘三档带宽全部钉死在其物理上限 **6.78 GB/s**，AISSD5000 三档峰值均达 **10.2 GB/s** 并在并发 32 时持续供给 **9.29 GB/s** 达 25 秒——首字延迟差距即介质供给能力差距。
2. **并发扩展性：两种介质的饱和拐点相差约 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。AISSD5000 最优吞吐工作点（并发 16、74.9 tok/s）较本地盘最优值高 40%。
3. **相比无外存重算：首字延迟快 12.6–20 倍、输出吞吐高 18 倍**（重算并发 16 下 TTFT p50 149.5 s、p90 237 s、吞吐 4.1 tok/s）——480B 模型的长上下文会话恢复，重算在工程上不可用，KV 存储层是刚需。
4. **负载越真实越重，优势越大；且供给能力可继续扩展**。此前 8K 短会话测试（需求低于本地盘天花板）两介质打平；本轮 30K 会话使需求越过该天花板后优势全面显现。本轮已将现有配置（4 盘 + 单口 100GbE）压至上限的 91%（瓶颈在网络口而非盘），增配端口/盘位可线性扩展（设备满配标称 72 GB/s）；本地盘带宽与容量（2 TB，已无法同时容纳模型权重与 KV 池）均无扩展余地。

二、被测系统与环境

2.1 硬件

组件	配置

GPU	8 × AMD Instinct MI308X（每卡 192 GB HBM，gfx942），TP=8 单实例 + expert parallel
CPU / 内存	2 × AMD EPYC 9654（384 线程），约 1.5 TB 内存
被测存储	**AISSD5000：4 盘组 RAID0（`/dev/md0`，xfs，14 TB），NVMe-oF / RoCEv2，单口 100 GbE**
对照存储	本地 NVMe 单盘（`/dev/nvme1n1`，PCIe Gen4，挂 `/srv2`）；无外存（重算）

2.2 软件

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，两轮均从 AISSD5000 阵列加载）
关键参数	`--tensor-parallel-size 8 --enable-expert-parallel`、`--max-model-len 32768`、`--gpu-memory-utilization 0.9`、`--no-enable-prefix-caching`（冷读取证）、`LMCACHE_CHUNK_SIZE=256`、`LMCACHE_MAX_LOCAL_CPU_SIZE=4`、`PYTHONHASHSEED=0`

2.3 工作负载与容量基准

- 单会话系统提示前缀约 **29.8K token**（reps=530），单会话 KV 约 **7.15 GB**——TP=8 下由 8 卡各持 1/8 分片、并行读写；
- 每轮灌入 **34 个互异会话**（实测落盘约 **243 GB**），远超显存驻留与 CPU 中转层（4 GB）容量；
- 测量：单波 N=并发数（8/16/32），读会话 0..N-1 各一次，decode=64，temperature=0。

三、测试方法学

3.1 对照设计（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）

三组仅 KV 后备介质不同；模型、权重来源、引擎参数、会话构造、并发档位完全一致。

3.2 物理冷读保证（五重控制）

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：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 主机内存的页缓存；
4. 每会话仅读取一次：清缓存后每次读取均为物理首读；
5. 双重取证：LMCache need-to-load 计数（磁盘组各档全部请求 >0；重算组 =0）+ `iostat` 物理读带宽与工作集体量吻合（重算组盘读 ≈0 反证纯重算）。

> 注意：新版 LMCache 默认异步加载，其日志中 `Retrieved ... throughput` 数字为暂存拷贝速度（可达 40+ GB/s），不能用于判断是否物理读盘；物理读一律以 `iostat` 为准。

3.3 测量与采集

- 客户端经 OpenAI 兼容流式接口测逐请求 TTFT（p50/p90/p99/均值）；输出吞吐 = N×64 ÷ 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）并确认显存归零。

四、测试结果

4.1 三方对照总表

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

重算 • 并发16	149.48	237.34	4.1	≈0	—	need=0（纯重算）

4.2 核心对比一：AISSD5000 vs 本地 NVMe（首字延迟与输出吞吐）

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%

> 解读：三档全胜、幅度稳定。差距来源不是软件而是介质供给能力——同一负载需求下，本地盘被物理上限（6.78 GB/s）卡住排队，AISSD5000 以 9.3–10.2 GB/s 持续供给。**会话越长、并发越高，本地盘排队越深，AISSD5000 的领先越稳固。**

4.3 核心对比二：并发扩展性（饱和拐点）

盘供给能力随并发的变化（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）

输出吞吐随并发的变化：

并发	AISSD5000	本地 NVMe
8	56.6 tok/s	43.9
16	**74.9（最优工作点）**	53.6
32	71.6（轻微回落）	53.9（平台：加并发零收益）

三个梯度结论：

1. **本地盘的饱和拐点在并发 8 之前即已到达**：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%）**，超载区（并发 32）仍保持 71.6 且延迟增长可控。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。

4.4 机理验证（带宽账目自洽）

- 并发 16 需搬运 16 × 7.15 = **114 GB**：AISSD5000 按忙窗均值 8.12 GB/s 需约 14.1 s（实测 TTFT p50 11.85 s、整批 13.3 s，吻合）；本地盘按 6.17 GB/s 需约 18.5 s（实测 17.31 s / 18.8 s，吻合）——**TTFT 差距 ≈ 介质带宽差距，测量自洽**；
- AISSD5000 三档峰值稳定于 10.2 GB/s，为单口 100 GbE 有效上限（约 11–12 GB/s）的 90%+——现有配置的供给能力已被真实推理负载充分利用，且瓶颈在网络口而非盘（4 盘各自仍有余量）；
- 推理软件栈未构成瓶颈：TP=8 使每请求 KV 由 8 卡并行读取各自分片，LMCache 默认异步加载使各请求加载充分重叠（同档内 TTFT p50≈p90，无串行阶梯）——胜负完全由存储介质供给能力决定。

4.5 核心对比三：AISSD5000 vs 无外存重算（并发16）

指标	重算	AISSD5000	收益
TTFT p50	149.48 s	**11.85 s**	**快 12.6 倍**
TTFT p90	237.34 s	**11.87 s**	**快 20.0 倍**
输出吞吐	4.1 tok/s	**74.9 tok/s**	**高 18.3 倍**

> 解读：480B 模型重算 30K 前缀，16 路并发下首字要等 2.5–4 分钟，工程上不可用；KV 外置复用压缩到 12 秒内。**长上下文大模型没有"重算"这个选项，KV 存储层是刚需**；在此前提下选哪种介质，由 §4.2/4.3 回答。

五、分析与讨论

5.1 AISSD5000 的优势区间与边界

此前 8K 短会话测试（每会话 KV 2 GB、并发 16、需求约 5.5 GB/s——低于本地盘上限）中两介质打平；本轮把会话拉长至 30K（KV 7.15 GB）后，冷恢复需求持续超过本地盘物理上限，AISSD5000 全面胜出。规律清晰：**需求低于本地盘天花板时两者等效；需求越过天花板后，差距即为两者供给能力之差，且随负载增长而扩大。** 480B 级模型的生产负载（长历史、多会话、恢复风暴）恰恰运行在后一区间。

5.2 容量维度的结构性差异

本测试单轮 KV 池即 243 GB；本地盘容量 2 TB 且须与模型权重、镜像共存（测试期间本地盘因空间腾挪已无法容纳 480B 权重副本，权重实际由 AISSD5000 承载）。生产级 KV 池在本地盘上没有容身之地；AISSD5000 起步 14 TB、按盘扩展，且天然可被多节点共享。

5.3 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 GB/s）。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——供给能力可随业务增长线性扩展；本地盘方案无此路径。

六、结论

1. **相比本地 NVMe 硬盘（核心结论）**：480B 八卡单实例、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%**，优势覆盖并发 8–32 全区间；机理为本地盘带宽钉死于 6.78 GB/s 物理上限，AISSD5000 以 9.3–10.2 GB/s 持续供给。
2. **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%。
3. **相比重算**：TTFT 快 12.6–20 倍、吞吐高 18 倍，长上下文大模型的会话恢复必须依赖 KV 存储层。
4. **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 91%（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。

七、局限与后续工作

1. **单机范围**：跨节点共享同一 KV 池（多机聚合、一次预填充全集群复用）未做物理验证，列为后续项；
2. **容量墙场景**：KV 池超过本地盘可用容量（迫使本地方案逐出重算）的稳态吞吐对比，是体现容量优势的下一个实验；
3. **重算组仅测一档**（并发 16）：重算耗时过长，一档已足够定性；
4. **合成负载**：均匀访问、固定长度，相对真实偏斜流量为保守估计；各点为单次测量。

附录 A：复现命令（三方完整）

> 约定：宿主直接执行；`docker exec vllm` 进容器执行。每组前彻底清理推理进程（含 EngineCore/Worker_TP 残留）并确认显存归零；populate 前台阻塞完成后再清缓存与测量。

A.1 起服（① AISSD5000；②③ 仅改注释处）

```
docker exec -d vllm bash -c "export VLLM_ROCM_USE_AITER=1 PYTHONHASHSEED=0 LMCACHE_LOG_LEVEL=INFO \
 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 /mnt/ws5000/models/Qwen3-Coder-480B-FP8 --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\"}' \
 > /mnt/ws5000/kv32k_ws.log 2>&1"
# ② 本地盘：LMCACHE_LOCAL_DISK=file:///srv2/lmcache480_local ，日志 kv32k_loc.log
# ③ 重算：去掉 LMCACHE_LOCAL_DISK / LMCACHE_MAX_LOCAL_DISK_SIZE ，日志 kv32k_rec.log
```

A.2 灌入（34 会话 × 29.8K token，重算组跳过）

```
docker exec vllm python3 /mnt/ws5000/benchcap_full.py WS32K_pp 530 34 populate 1 1   # 约 280s，落盘约 243GB
```

A.3 并发梯度测量（每档前清页缓存 + iostat）

```
for C in 8 16 32; do
  sync; echo 3 > /proc/sys/vm/drop_caches
  nohup bash -c "iostat -x 1 240 /dev/md0 > /tmp/io32k_WS32K_c$C.log 2>&1" &   # ②监控 /dev/nvme1n1
  docker exec vllm python3 /mnt/ws5000/benchcap_off.py WS32K_c$C 530 $C 64 $C 0
  awk '$1=="md0"{if($3>m)m=$3}END{printf "peak %.2f GB/s\n", m/1e6}' /tmp/io32k_WS32K_c$C.log
  awk '$1=="md0" && $3>500000{c++;s+=$3}END{if(c)printf "busy_avg %.2f GB/s (%ds)\n", s/c/1e6, c}' /tmp/io
32k_WS32K_c$C.log
done
# 取证：grep -acE 'need to load: [1-9]' <服务日志>   # 磁盘组应=N，重算组应=0
```

附录 B：测量脚本

benchcap_off.py（测量：N 会话单波冷读，offset 支持）

```
# 用法: python3 benchcap_off.py <label> <reps> <N> <decode> <conc> <offset>
import urllib.request, json, time, sys
from concurrent.futures import ThreadPoolExecutor
BASE='http://localhost:8000/v1/chat/completions'; MODEL='qwen'
label=sys.argv[1]; reps=int(sys.argv[2]); N=int(sys.argv[3]); decode=int(sys.argv[4]); conc=int(sys.argv
[5]); off=int(sys.argv[6])
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':MODEL,'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':'applicatio
n/json'}),timeout=1200)
        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 (ttft, time.time()-st, n)
    except Exception:
        return (None,None,0)
def pct(a,q): return a[min(len(a)-1,int(len(a)*q))] if a else 0.0
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[0] is not None: res.append(x)
wall=time.time()-t0
tt=sorted(r[0] for r in res)
print('[%s] n=%d wall=%.1fs TTFT p50=%.3f p90=%.3f p99=%.3f mean=%.3f'%(label,len(res),wall,pct(tt,.5),pct
(tt,.9),pct(tt,.99),(sum(tt)/len(tt) if tt else 0)),flush=True)
```

benchcap_full.py（灌入模式：逐会话 prefill、KV 落盘）用法：`python3 benchcap_full.py <label> <reps> <N> populate 1 1`；`reps=530` → 每会话约 29.8K token。

附录 C：原始数据文件

路径	内容
`/tmp/kv32k.out`	实验全程编排日志（各档 TTFT/吞吐/带宽/取证输出）
`/mnt/ws5000/kv32k_ws.log`、`kv32k_loc.log`、`kv32k_rec.log`	三方 vLLM/LMCache 服务日志（need-to-load 原文）
`/tmp/io32k_WS32K_c{8,16,32}.log`、`/tmp/io32k_LOC32K_c{8,16,32}.log`	iostat 物理读时间序列
`/mnt/ws5000/lmcache480`、`/srv2/lmcache480_local`	两轮 KV 数据（各约 243 GB）

本报告全部数据来自实测，测试方法、命令与脚本完整留档，可独立复现。