性 能 测 试 报 告

AISSD5000 全闪存储
大模型推理与训练 综合性能测试报告

测试平台:AMD Instinct MI308X ×8

测试模型:Qwen2.5-32B / 7B-Instruct

对照基线:本地高端 NVMe 单盘

报告日期:2026 年 07 月 03 日

> 本报告在 AMD Instinct MI308X 平台上,系统评估 AISSD5000(全闪 NVMe-oF 存储阵列,亦称 WS5000)作为大模型推理 KV Cache 分层介质、以及推理加载/模型切换/训练权重存取后备介质的性能,统一以**本地高端 NVMe 单盘**为对照基线。全部数据为实测,采用单一变量、强制物理读盘/冷读、iostat 物理取证、带宽物理自洽校验。

1. 测试环境

组件	配置
GPU	AMD Instinct MI308X ×8,每卡 192GB HBM,ROCm 7.2(gfx942)
主机内存	约 1.5 TB
受测存储	**AISSD5000**:4×3.84TB NVMe 组 RAID0 = 14TB(`/dev/md0`),XFS,NVMe-oF/RoCEv2,经 100GbE(Intel E810/irdma)挂 `/mnt/ws5000`
对照基线	**本地单盘**:Solidigm SSDPF2KX076T1 7.68TB NVMe(`/dev/nvme1n1`,`/srv2`)
推理引擎	vLLM 0.20.1(vllm-openai-rocm 定制镜像)
KV 缓存库	LMCache(`BUILD_WITH_HIP` 源码编译,含并行读补丁)
训练栈	transformers + peft(LoRA)
模型	Qwen2.5-32B-Instruct(bf16,约 60GB);Qwen2.5-7B-Instruct

2. 术语

术语	含义
KV Cache	推理 prefill 阶段产生的键值中间结果,可复用以跳过重复计算
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)
请求并发(conc)	单实例同一时刻同时处理的请求数
实例并发	同时运行的独立 vLLM/训练进程数(每卡一个)
聚合带宽	iostat 实测块设备读/写速率(已清页缓存);多实例场景为合计带宽
冷读	清页缓存后强制从物理盘读取(排除内存缓存假象)
need to load	LMCache 日志字段,>0 表示确有物理回读

第一部分:LMCache 并行读优化(KV Cache 分层)

3.1 背景与补丁

LMCache 本地磁盘后端的批量回读走基类**串行**实现(逐个 `get_blocking`),无法利用 AISSD5000 四盘 RAID0 的并联带宽——`/dev/md0` 仅约 1GB/s、4 盘各约 250MB/s,阵列只用了约 1/5。本项目重写 `batched_get_blocking`,以**线程池(32 线程)并行**读取各文件块,打开聚合带宽(补丁见附录 A)。

> 说明:LMCache 自带的 `enable_async_loading` 异步预取在本环境会死锁,不可用;故采用"串行分配缓冲区 + 线程池并行读文件"的补丁绕过。

3.2 补丁效果(单卡 · 并发16 · 冷读盘)

指标	补丁前(串行)	补丁后(并行)	提升
TTFT p50	37.97 s	9.30 s	↓ 4.1 倍
md0 物理读带宽	0.98 GB/s	5.23 GB/s	↑ 5.3 倍
4 盘各自读带宽	250 MB/s(闲置)	1340 MB/s	↑ 5.3 倍
输出吞吐	~24 tok/s	86.5 tok/s	↑ 3.6 倍

3.3 单卡并发梯度(补丁 ON):软件单管道墙

并发	TTFT p50	md0 峰值带宽
16	9.77 s	5.18 GB/s
32	19.9 s	5.41 GB/s
48	28.4 s	5.71 GB/s
64	32.5 s	4.93 GB/s

单卡加并发,带宽稳定于约 5.5GB/s 不再上升、TTFT 线性增长——瓶颈转移至 LMCache **单引擎单管道**处理链路(读盘已并行,后续反序列化/拼装/送 GPU 仍串行)。

3.4 多卡多实例扩展(补丁 ON)

配置	md0 峰值带宽	持续带宽	总吞吐	4 盘各自
单卡 1 实例	5.2 GB/s	5.2	86 tok/s	1340 MB/s
4 卡 4 实例	8.83 GB/s	6.9	203 tok/s	2260 MB/s
8 卡 8 实例(conc16)	9.69 GB/s	7.13	228 tok/s	2480 MB/s
8 卡 8 实例(conc32)	**10.18 GB/s**	8.93	265 tok/s	2600 MB/s

多实例=多管道,把并行传导至盘与网络,聚合带宽随实例数上升;8 卡 conc32 逼近 100GbE 网络上限(10.18GB/s),而 4 盘各自仅 2600MB/s、仍有余力——此时瓶颈为网络接口,非存储介质。

3.5 严格公平对比:AISSD5000 vs 本地单盘(8卡×32会话×conc32,口径完全对齐)

指标	本地单盘	AISSD5000	AISSD5000 提升
聚合峰值带宽	6.47 GB/s	10.18 GB/s	+57%
聚合持续带宽	6.29 GB/s	8.93 GB/s	+42%
总吞吐	~179 tok/s	~265 tok/s	+48%
盘状态	单盘 util 85%(物理到顶)	4 盘各 2600MB/s(有余力)	—

KV Cache 部分结论:并行补丁是释放 AISSD5000 聚合带宽的前提;多卡并发下 AISSD5000 全面反超本地单盘(8 卡 conc32 聚合带宽 +57%、吞吐 +48%),且并发越大领先越大,当前上限为 100GbE 网络。

第二部分:存储场景三项测试(八卡多并发)

> 对照参考"存储越快 → 加载/切换/存取越快 → 有效算力越高"的思路,在本平台以本地单盘为基线,于 8 卡整机满配、多并发下测三类关键场景。核心发现:**并发把负载传导到存储聚合带宽,AISSD5000 全面反超本地单盘**。

4. 测试项① 模型并发加载(Qwen2.5-32B,冷读,8 卡同时)

指标	本地单盘	AISSD5000	AISSD5000 优势
单实例模型加载	~95.7 s	~67.0 s	**快 30%**
8 实例全就绪总墙钟	167.7 s	141.3 s	**快 16%**
盘读带宽(持续)	0.64 GB/s	0.91 GB/s	高 42%

8 实例(每卡一个)同时启动、冷读同一份模型,产生真实并发读争抢;AISSD5000 更高读带宽显现,单实例加载由本地 96s 缩至 67s。

5. 测试项② 模型切换有效词元产出率 TPS

以有效Token = R×(W − 切换次数×加载时延)、W=3600s 建模。实测聚合 R_8=11732 tok/s(8 实例 × 每实例请求并发 128),并发加载时延 AISSD5000 67s / 本地 96s:

切换频次	指标	本地单盘	AISSD5000	AISSD5000 提升
4 次/h	有效TPS	10480	10859	+3.6%
10 次/h	有效TPS	8604	9549	+11.0%
20 次/h	有效TPS	5477	7369	**+34.5%**
20 次/h	算力有效利用率	46.7%	62.8%	+16.1 个百分点

切换越频繁,AISSD5000(并发加载更快)累计节省的算力空转越多,有效产出领先越大。

6. 测试项③ 训练 + Checkpoint 并发写(8 卡,32B,每份 65.6GB 整模型快照)

8 个训练进程(每卡一个,32B LoRA 微调)同时训练、每轮各保存整份 65.6GB 模型快照(合计 ~525GB/轮):

指标	本地单盘	AISSD5000	AISSD5000 优势
每实例 ckpt 保存均值	~178 s	~94 s	**快 1.9 倍**
聚合写带宽(持续)	3.26 GB/s	6.40 GB/s	高 96%
聚合写带宽(峰值)	4.19 GB/s	7.20 GB/s	高 72%
模型加载 / 每轮训练	~34s / ~4.3s	~34.6s / ~4.4s	相当(算力,与存储无关)

Checkpoint 是**大块连续写**、且 8 个独立目录无法被页缓存抄近路——8 路并发写直接压满本地单盘,AISSD5000 四盘 RAID0 聚合写优势充分兑现,是三项中差距最大者。

7. 综合结论

7.1 规律:并发把负载传导到存储

在 8 卡整机满配、多并发下,负载被传导到存储聚合带宽:本地单盘被多路并发读/写榨干(利用率打满),而 AISSD5000 的 4 盘 RAID0 + 独立 100GbE 提供更高聚合带宽,优势全面显现——**并发越高、写越密集,AISSD5000 领先越大**。

7.2 各场景 AISSD5000 相对本地单盘的优势(八卡并发)

场景	AISSD5000 优势
KV Cache 并发回读(8卡×conc32)	聚合带宽 +57%、吞吐 +48%
模型并发加载(8卡)	快 16–30%
模型切换有效TPS(20次/h)	+34.5%
训练 checkpoint 并发写(8卡)	快 1.9 倍,聚合写带宽 +96%

7.3 关键洞察

- **写密集场景(checkpoint)差距最大**:写必须落盘、无法缓存规避,最能体现聚合写带宽差异。
- **KV Cache 需配合并行读补丁**才能发挥 AISSD5000 带宽;当前多卡上限为 **100GbE 网络**,升级 200G/400G 可进一步提升。
- **平台/基线口径**:AMD MI308X 平台、本地高端 NVMe 基线;因基线本身很快,绝对差距小于慢基线(如 NFS)场景,但趋势一致——**规模越大、并发越高、写越密集,AISSD5000 优势越明显**。

附录 A:LMCache 并行读补丁(`local_disk_backend.py`)

重写 `batched_get_blocking`:先串行分配缓冲区(CPU 分配器不假定线程安全),再用线程池(32 线程)并发 `read_file`,最后统一更新缓存策略。

```
def batched_get_blocking(self, keys):
    # [PAR-READ PATCH] 串行分配缓冲区,线程池并发 read_file
    if not keys: return []
    if len(keys) == 1: return [self.get_blocking(keys[0])]
    paths, mobjs = [], []
    for key in keys:
        with self.disk_lock:
            meta = self.dict.get(key)
            path  = meta.path  if meta else None
            dtype = meta.dtype if meta else None
            shape = meta.shape if meta else None
            fmt   = meta.fmt   if meta else None
        mo = self.local_cpu_backend.allocate(shape, dtype, fmt) if meta else None
        if mo is not None:
            _m = self.dict.get(key)
            mo.metadata.cached_positions = _m.cached_positions if _m else None
        paths.append(path); mobjs.append(mo)
    pool = getattr(self, "_par_read_pool", None)
    if pool is None:
        from concurrent.futures import ThreadPoolExecutor
        pool = ThreadPoolExecutor(max_workers=32, thread_name_prefix="lmc_par_read")
        self._par_read_pool = pool
    def _rd(i):
        if paths[i] is not None and mobjs[i] is not None:
            self.read_file(keys[i], mobjs[i].byte_array, paths[i])
    list(pool.map(_rd, range(len(keys))))
    with self.disk_lock:
        for key, mo in zip(keys, mobjs):
            if mo is not None and key in self.dict:
                self.cache_policy.update_on_hit(key, self.dict)
    return mobjs
```

附录 B:方法学要点

1. **强制冷读/物理读**:测量前 `sync; echo 3 > /proc/sys/vm/drop_caches`;KV Cache 场景另需关 vLLM prefix caching + 压小 LMCache CPU 层 + 读全新会话,并以 `need to load>0` 与 iostat 交叉验证(上报带宽 > 网络/盘物理上限即判为缓存命中、作废)。
2. **iostat 物理取证**:全程记录 md0(及 4 成员盘)与本地 nvme1n1 的读/写带宽、%util。
3. **物理自洽校验**:有效带宽 ≤ 链路上限(AISSD5000 100GbE≈12.5GB/s,本地单盘≈7GB/s)。
4. **单一变量**:每项仅切换"数据/模型/输出所在存储",其余(模型、并发、负载)一致;加载/存取类各测代表性次数。
5. **训练项**:32B 全量微调单卡显存不足,采用 LoRA 训练 + 每轮保存整份 65.6GB 模型快照(只留最新)以制造真实写负载;8 实例各绑一卡、各写独立目录(8 路并发写)。

附录 C:关键实测数据出处

- KV Cache:补丁前后、并发梯度、1/4/8 卡扩展、8卡×conc32 公平对比(iostat md0 + LMCache 日志)。
- 三项:①vLLM `Model loading took` + 就绪墙钟;②实测 R/R_8 + 建模;③训练脚本记录每轮训练/ckpt 保存耗时 + iostat 写带宽。

本报告全部数值均来自实测，方法、命令与脚本完整留档，可独立复现。