# AISSD5000 KV Cache 性能测试汇总报告 —— 480B 模型 · TP4×2 双实例 · 长上下文冷恢复（全指标）

**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）

**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，权重约 450 GB）

**部署形态**：双实例 TP4×2（实例 A：卡 0–3 / 端口 8000；实例 B：卡 4–7 / 端口 8001，各 + expert parallel）

**负载**：长上下文冷恢复——每会话约 29.8K token，KV ≈ 7.15 GB/会话，输出 64 token

**日期**：2026-07-06（本报告汇总当日全部 TP4×2 实验；总表数据为当日下午以增强客户端统一重测所得，与当日上午两轮独立结果一致，偏差 ≤5%）

## 一、主要结论

**四组配置（AISSD5000 独立池 / AISSD5000 fs:// 共享池 / 本地 NVMe / 无外存重算）× 两档全机并发（16 / 32），共九档测量，全部为清页缓存后的物理冷读，逐请求记录 TTFT 与 TPOT。**

1. **相比本地 NVMe（核心对比）：AISSD5000 使首字延迟 p50 降低 26%–30%、p99 降低 28%、聚合输出吞吐提升 35%–36%**。全机并发 16 档：TTFT p50 由 19.0 s 降至 **13.4 s**、p99 由 19.0 s 降至 **13.8 s**、吞吐由 49.7 升至 **66.9 tok/s**；并发 32 档：TTFT p50 由 34.5 s 降至 **25.6 s**、p99 由 36.4 s 降至 **26.3 s**、吞吐由 53.3 升至 **72.6 tok/s**。机理：本地盘两档均钉死在 6.78 GB/s 物理上限，AISSD5000 以 8.6–9.6 GB/s 忙窗均值、10.3–10.4 GB/s 峰值（单口 100GbE 线速 90%+）持续供给。
2. **相比无外存重算：TTFT p50 快 8.6–9.8 倍，p99 快 19–23 倍，吞吐高 18–21 倍，TPOT p50 好 75–164 倍**。重算档 TPOT p50 达 1.8–5.2 秒/token（解码被持续 prefill 挤占，输出流近乎冻结），TTFT p99 达 4.5–10 分钟——480B 长上下文的会话恢复没有"重算"这个选项。
3. **fs:// 共享池与独立盘后端性能完全打平，且跨实例热共享零代价**：共享池读自己（OWN）与独立后端（WS）各项指标一致（TTFT p50 24.8 vs 25.6 s、吞吐 73.4 vs 72.6 tok/s）；**交叉读对方灌入的会话（CROSS）仅比读自己慢 8%（p50）/ 3%（p99），且仍比本地盘读自己的快 22%**。一份 KV 全机复用、会话跨实例自由迁移成立。
4. **解码质量（TPOT）四组磁盘档一致健康**：TPOT p50 全部落在 24–32 ms 区间，介质不影响稳态解码。
5. **测量自洽、可复现**：各档 iostat 读盘体量与工作集吻合（如 WS 并发 32 档：9.59 GB/s × 25 s ≈ 240 GB ≈ 32 × 7.15 GB）；当日上午与下午两次独立完整测量，各档指标偏差 ≤5%。

# 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） |
| 部署形态 | 实例 A：卡 0-3（TP=4，端口 8000）；实例 B：卡 4-7（TP=4，端口 8001），均 + 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 软件与关键参数

| 组件 | 版本/取值 |
| --- | --- |
| 操作系统 / GPU 栈 | Ubuntu 22.04（内核 6.8.0-124） / ROCm 7.2 |
| 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） |
| 引擎参数 | `--tensor-parallel-size 4 --enable-expert-parallel`、`--max-model-len 32768`、`--gpu-memory-utilization 0.9`、`--no-enable-prefix-caching`、`PYTHONHASHSEED=0` |
| LMCache 公共参数 | `LMCACHE_CHUNK_SIZE=256`、`LMCACHE_LOCAL_CPU=True`、`LMCACHE_MAX_LOCAL_CPU_SIZE=4` |

## 2.3 四组存储配置

| 组 | KV 后备配置 | 说明 |
| --- | --- | --- |
| ① WS（独立池） | `LMCACHE_LOCAL_DISK=file:///mnt/ws5000/lmcache480tp4`（md0） | LocalDiskBackend，每实例进程内索引 |
| ② FS（共享池） | `LMCACHE_REMOTE_URL=fs://local:0/mnt/ws5000/kvpool_fs`（md0） | FSConnector，索引即文件系统，双实例共享一个池 |
| ③ LOC（本地盘） | `LMCACHE_LOCAL_DISK=file:///srv2/lmcache480tp4_local`（nvme1n1） | LocalDiskBackend，本地单盘 |

| | | |
|---|---|---|
| ④ RC（重算） | 不挂 LMCache | 纯 GPU 重算基线 |

> 注：fs:// 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 RemoteBackend 建连失败、KV 静默丢弃（详见《AISSD5000-KVCache跨实例热共享验证报告》§2）。

# 三、测试方法学

## 3.1 指标定义

| 指标 | 定义 |
|---|---|
| TTFT | 请求发出到收到首个内容 token 的时间（流式接口逐请求测量）；p50/p90/p99/均值按全机 A+B 合并样本计算（并发 16 档 n=16，并发 32 档 n=32，p99 为名义秩即最差请求） |
| TPOT | 逐请求 (E2E − TTFT) ÷ (输出 token 数 − 1)，即首 token 后的平均每 token 解码间隔 |
| 聚合输出吞吐 | 全机输出 token 总数（2×N×64）÷ 两实例中较慢者的整批耗时 |
| 盘峰值 / 忙窗均值 | `iostat -x 1` 秒级采样的读带宽最大值 / >0.5 GB/s 窗口内均值 |

## 3.2 负载与冷读保证

- 双实例并行灌入 48 个互异会话（A 灌 0–23、B 灌 24–47，落盘 343–344 GB），灌入耗时约 363 s；
- 测量档位：全机并发 16（每实例 8 会话 × 并发 8）与 32（每实例 16 × 16）；A 读 0 起、B 读 24 起（FS-CROSS 档交换：A 读 24 起、B 读 0 起，即互读对方灌入的会话）；
- 每档测量前宿主执行 `sync; echo 3 > /proc/sys/vm/drop_caches` 清除 1.5 TB 页缓存；每会话仅读一次；两实例读取集合不相交（无页缓存搭车）；
- 冷读取证：磁盘组逐请求满额命中（hit tokens = 30208，当日两轮独立验证 96/96 与 96/96）；iostat 读盘体量与工作集吻合；重算组盘读 ≈0 且进程无 LMCache；
- 组间切换彻底清理推理进程（含 EngineCore/Worker 残留）并确认 8 卡显存归零。

# 四、九档全指标总表（全机口径，A+B 合并样本）

| 档位 | 全机并发 | TTFT p50（s） | TTFT p90（s） | TTFT p99（s） | TTFT 均值（s） | TPOT p50（ms） | 聚合吞吐（tok/s） | 盘峰值（GB/s） | 盘忙窗均值（GB/s） |
|---|---|---|---|---|---|---|---|---|---|
| WS·16 | 16 | **13.38** | 13.78 | **13.79** | 12.15 | 24.2 | **66.9** | 10.32 | 8.56 |
| WS·32 | 32 | **25.62** | 26.24 | **26.25** | 23.09 | 32.0 | **72.6** | 10.41 | **9.59** |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| FS共享·OWN·16 | 16 | **10.57** | 14.41 | 14.42 | 11.39 | 82.5 | 64.0 | 10.38 | 9.21 |
| FS共享·OWN·32 | 32 | **24.83** | 26.03 | **26.04** | 23.97 | 30.7 | **73.4** | 10.36 | 9.36 |
| FS共享·CROSS·32 | 32 | **26.81** | 26.83 | **26.84** | 24.57 | 31.6 | **71.1** | 10.38 | 9.22 |
| 本地NVMe·16 | 16 | 19.03 | 19.04 | 19.04 | 16.11 | 24.1 | 49.7 | 6.79 | 6.31 |
| 本地NVMe·32 | 32 | 34.48 | 36.44 | 36.44 | 31.60 | 30.9 | 53.3 | 6.78 | 6.45 |
| 重算·16 | 16 | 114.78 | 210.41 | 268.52 | 122.80 | 1817.4 | 3.8 | ≈0 | — |
| 重算·32 | 32 | 251.69 | 466.95 | 608.52 | 255.70 | 5244.0 | 3.4 | ≈0 | — |

> CROSS = 交叉读（A 读 B 灌入的会话、B 读 A 灌入的会话，32/32 全量磁盘命中）；OWN = 各读自己灌入的。FS·OWN·16 档 TTFT p50 偏低但 p90 抬升，为 fs 异步加载在低并发下的请求间发散（详见 §5.4），高并发档该现象消失。

## 五、对比分析

### 5.1 AISSD5000 vs 本地 NVMe（同为 LocalDiskBackend，仅介质不同）

| 全机并发 | 指标 | 本地 NVMe | AISSD5000 | 收益 |
|---|---|---|---|---|
| 16 | TTFT p50 | 19.03 s | **13.38 s** | **−30%** |
| 16 | TTFT p99 | 19.04 s | **13.79 s** | **−28%** |
| 16 | 聚合吞吐 | 49.7 tok/s | **66.9 tok/s** | **+35%** |
| 32 | TTFT p50 | 34.48 s | **25.62 s** | **−26%** |
| 32 | TTFT p99 | 36.44 s | **26.25 s** | **−28%** |
| 32 | 聚合吞吐 | 53.3 tok/s | **72.6 tok/s** | **+36%** |

- 差距即介质供给能力之差：本地盘两档忙窗均值 6.31 / 6.45 GB/s，贴死其 6.78 GB/s 物理顶；AISSD5000 为 8.56 / 9.59 GB/s，峰值 10.3–10.4 GB/s（单口线速 90%+）；
- 带宽账目自洽：并发 32 档需搬运 32 × 7.15 = 229 GB，AISSD5000 按 9.59 GB/s 需约 23.9 s（实测 TTFT p50 25.6 s），本地盘按 6.45 GB/s 需约 35.5 s（实测 34.5 s）——**TTFT 差距 ≈ 介质带宽差距**；
- p99 与 p50 几乎相等（磁盘四档均如此）：LMCache 异步加载使同档各请求充分重叠、同步完成，无长尾——**低延迟是全体请求的，不是中位数的。**

### 5.2 AISSD5000 vs 无外存重算

| 全机并发 | 指标 | 重算 | AISSD5000 | 收益 |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| 16 | TTFT p50 | 114.8 s | **13.4 s** | **快 8.6 倍** |
| 16 | TTFT p99 | 268.5 s | **13.8 s** | **快 19.5 倍** |
| 16 | TPOT p50 | 1817 ms | **24.2 ms** | **好 75 倍** |
| 16 | 聚合吞吐 | 3.8 tok/s | **66.9 tok/s** | **高 17.6 倍** |
| 32 | TTFT p50 | 251.7 s | **25.6 s** | **快 9.8 倍** |
| 32 | TTFT p99 | 608.5 s | **26.3 s** | **快 23.2 倍** |
| 32 | TPOT p50 | 5244 ms | **32.0 ms** | **好 164 倍** |
| 32 | 聚合吞吐 | 3.4 tok/s | **72.6 tok/s** | **高 21.4 倍** |

重算不仅首字慢一个数量级，**解码也被拖垮**：持续的 30K prefill 挤占解码批次，TPOT p50 达 1.8–5.2 秒/token（正常 24–32 ms），首 token 之后的输出流同样不可用；p99 口径下差距扩大到 19–23 倍（重算最差请求等 4.5–10 分钟）。TP4 实例算力为 TP8 之半，重算劣化比 TP8 形态（12.6–20 倍）更深——**实例切得越细，KV 外置层越是刚需**。

## 5.3 fs:// 共享池 vs 独立盘后端 & 跨实例共享代价

| 对比（并发 32 档） | TTFT p50 | TTFT p99 | TPOT p50 | 聚合吞吐 | 盘忙窗均值 |
|---|---|---|---|---|---|
| WS 独立池（读自己） | 25.62 s | 26.25 s | 32.0 ms | 72.6 tok/s | 9.59 GB/s |
| FS 共享池·读自己 | 24.83 s | 26.04 s | 30.7 ms | 73.4 tok/s | 9.36 GB/s |
| FS 共享池·交叉读对方的 | **26.81 s** | **26.84 s** | **31.6 ms** | **71.1 tok/s** | 9.22 GB/s |

- **共享后端零性能代价**：FS·OWN 与 WS 各项指标在噪声内一致；
- **跨实例共享近零代价**：CROSS 比 OWN 慢 8%（p50）/ 3%（p99），32/32 全量磁盘命中；
- **跨实例读别人的，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（hit=0、TTFT 67 s 退化重算）：共享能力由后端选择决定，fs:// 一步打通。

## 5.4 TPOT（解码质量）解读

- 四个磁盘档 TPOT p50 全部在 24–32 ms：**KV 介质不影响稳态解码速度**，差异只在恢复阶段（TTFT）；
- FS·OWN·16 档的 TPOT p50 82.5 ms 偏高（伴随 TTFT 请求间发散）：**经 5 轮复测 + 2 轮修复验证定位为配置问题而非偶然**。根因是 4 GB CPU 中转层对 fs remote 取回路径不足——8 个并发请求在途 KV（约 57 GB）远超中转池，触发"分配失败（单轮最高 1.9 万次告警）→ 无逐出候选 → 阻塞超时取消重试"，轻则各请求完成时刻发散（TPOT p50 升至 80–92 ms），重则整波停摆（复测 5 轮中 1 轮：单实例 8 个请求 TTFT 同为 61.8 s）。**将 `LMCACHE_MAX_LOCAL_CPU_SIZE` 提至 64 GB 后，分配失败与超时告警归零，连续 3 轮真冷读全部干净**（TTFT p50 13.1–15.4 s、TPOT p50 23–27 ms、无停摆），仅实例启动后的首轮测量存在轻微预热发散。生产配置建议：fs remote 路径的 CPU 中转层应 ≥ 并发在途 KV 体量（本负载即 ≥57 GB）；

- 附带发现：64 GB 中转层使近期读过的会话在 CPU 层驻留，二次恢复 TTFT 仅 1.8 s（内存层命中、零盘读）——显存/内存/阵列三级分层的中间层价值在大内存主机上可直接兑现；
- 重算档 TPOT p50 1.8–5.2 s：解码与持续 prefill 争抢引擎，输出流冻结——重算的不可用是全链路的。

### 5.5 与 TP8 单实例形态（07-05）的横向印证

同一台机器、同一负载规格下，TP8×1 轮 AISSD5000 对本地盘为 TTFT −26%~−32%、吞吐 +29%~+40%；本轮 TP4×2 为 TTFT −26%~−30%、吞吐 +35%~+36%——**收益幅度跨部署形态稳定**，两种形态下阵列峰值均为 10.2–10.4 GB/s、本地盘均钉死 6.78 GB/s。

## 六、结论

1. **AISSD5000 对本地 NVMe：TTFT p50 降 26%–30%、p99 降 28%、吞吐升 35%–36%**（480B TP4×2 长上下文冷恢复，全档物理冷读取证）；p99≈p50 表明收益覆盖全体请求无长尾。机理为本地盘钉死 6.78 GB/s 物理顶、AISSD5000 以 8.6–9.6 GB/s 持续供给（单口线速 90%+）。
2. **对无外存重算：TTFT p50 快 8.6–9.8 倍、p99 快 19–23 倍、吞吐高 18–21 倍、TPOT 好 75–164 倍**——重算在首字与解码两端同时不可用，KV 外置层是长上下文大模型多实例部署的前提设施。
3. **fs:// 共享池实现跨实例热共享且零性能代价**：交叉读全量命中，代价仅 3%–8%；跨实例读别人的仍比本地盘读自己的快 22%。一份 KV 全机复用、会话自由迁移、重启不丢缓存成立。
4. **解码质量与介质无关**（TPOT p50 24–32 ms 四组一致），存储的价值集中兑现在恢复延迟与吞吐上。
5. **数据可信**：两次独立完整测量偏差 ≤5%；带宽账目与工作集自洽；全部命中/盘读双重取证。

## 七、局限

1. 并发 16 档的 FS 共享池 TPOT 发散已复测定界并修复（§5.4：4 GB CPU 中转层不足所致，64 GB 后消失）；总表数据为 4 GB 配置下所测，FS 两档在充足中转层下的指标只会更优；
2. p99 为名义秩（n=16/32 下即最差请求），更严格的尾延迟需更大样本；
3. 跨节点共享、多机聚合供给未做物理验证（需第二台接入存储网络的主机）；
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。

# 附录 A：复现命令（同事服务器，容器 `vllm` 内）

## A.1 双实例启动（四组仅存储环境变量不同）

```
MODEL=/mnt/ws5000/models/Qwen3-Coder-480B-FP8
# ① WS 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"
# ② FS 共享池: ENVS="LMCACHE_CHUNK_SIZE=256 LMCACHE_LOCAL_CPU=True LMCACHE_MAX_LOCAL_CPU_SIZE=4 \
#               LMCACHE_REMOTE_URL=fs://local:0/mnt/ws5000/kvpool_fs LMCACHE_REMOTE_SERDE=naive"
# ③ 本地NVMe:  ENVS 同① 但 LMCACHE_LOCAL_DISK=file:///srv2/lmcache480tp4_local
# ④ 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.2 灌入与测量（bench_mp2.py 见附录 B）

```
# 并行灌入 48 会话
docker exec -d vllm bash -c "python3 /mnt/ws5000/bench_mp2.py 8000 populate 530 24 0  > /mnt/ws5000/result
s/m2_ppA.log 2>&1"
docker exec -d vllm bash -c "python3 /mnt/ws5000/bench_mp2.py 8001 populate 530 24 24 > /mnt/ws5000/result
s/m2_ppB.log 2>&1"
# 每档测量（示例: 全机并发32; CROSS 档 A/B 起点互换）
sync; echo 3 | sudo tee /proc/sys/vm/drop_caches
iostat -x 1 900 /dev/md0 > /tmp/io.log &      # 本地盘组监控 /dev/nvme1n1
docker exec -d vllm bash -c "python3 /mnt/ws5000/bench_mp2.py 8000 measure 530 16 0  64 16 > /mnt/ws5000/r
esults/m2_WS16_A.log 2>&1"
docker exec -d vllm bash -c "python3 /mnt/ws5000/bench_mp2.py 8001 measure 530 16 24 64 16 > /mnt/ws5000/r
esults/m2_WS16_B.log 2>&1"
# 全机分位聚合: 合并 A/B 两文件的 REQ 行后按 3.1 节定义计算
```

## 附录 B：增强负载客户端 `bench_mp2.py`（逐请求 TTFT/E2E/TPOT）

```
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 p99=%.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),pct(tpots,.99),sum(tpots)/len(tpots),tot/wall),flush=True)
```

## 附录 C：原始数据存档（同事服务器）

| 文件 | 内容 |
|---|---|
| `/tmp/full2.out` | 全指标重测编排日志（九档全输出） |
| `/mnt/ws5000/results/m2_*_{A,B}.log` | 各档两实例逐请求原始数据（REQ 行）与汇总行 |

| | |
|---|---|
| `/mnt/ws5000/{m2fs,m2ws,m2loc,m2rc}_i{0,1}.log` | 八个服务实例完整日志（磁盘组含逐请求 hit 取证行） |
| `/tmp/io_m2_*.log` | 各档 iostat 秒级原始记录 |
| `/tmp/tp4x2b.out`、`/tmp/tp4x2c.out`、`/tmp/fsshare.out` | 当日上午/下午两轮独立测量日志（复现性对照） |
| `/mnt/ws5000/kvpool_fs/` | fs 共享池（344 GB / 22656 文件，保留可复查） |