分别记录主机内存、显存与输入缓存，定位模型任务峰值

完整操作教程 · 参考固定 · 往返恢复验收

GPU空闲不代表主机RAM充足；用任务cgroup和设备日志区分加载、缓存与推理峰值。

运行边界：需要已分配GPU和对应任务cgroup；不具备这些条件时，不把监控示例当作GPU实测。

模型权重放进GPU后，输入解码、Python对象、DataLoader worker和结果缓存仍可占用主机RAM。本篇给已有、能独立运行的infer.py加一层监测，不更改模型方法。目标是回答“哪一阶段先接近哪种容量”，而不是只看一张nvidia-smi截图。

先确定监测属于这一份任务

需要Linux cgroup v2、可读的独立任务cgroup以及可用的NVIDIA监测接口。请由调度器或管理员确认CGROUP_DIR确实包含当前任务及worker，且不混入其他项目；不能随便取系统根cgroup。示例只读现有统计，不创建或修改资源限制。

记录	口径	用途
memory.current	cgroup及其后代当前内存，字节	包含多进程和相关缓存的任务口径
memory.peak	该cgroup生命周期峰值	旧cgroup的历史峰值不能当本次峰值
nvidia-smi设备内存	指定设备的使用量，通常MiB	共享卡上可能混有其他任务
阶段日志	加载、首批、稳定推理、写出	用时间戳与曲线对应

运行同一输入，保留两类曲线

在infer.py关键位置输出带时间戳的阶段名并立即flush。先确认下面的配置确实能运行；GPU_ID填管理员分配的物理UUID，不推断它与CUDA_VISIBLE_DEVICES中的逻辑序号相同。监测循环每秒采样，极短峰值可能漏掉。

```
#!/usr/bin/env bash
set -euo pipefail
: "${CGROUP_DIR:?set the verified job cgroup path}"
: "${GPU_ID:?set the assigned GPU UUID}"
test -r "$CGROUP_DIR/memory.current"
test -r "$CGROUP_DIR/memory.peak"
test -s infer.py
test -s config.json
nvidia-smi -i "$GPU_ID" -L
mkdir memory-probe-001
root="$PWD/memory-probe-001"
cat "$CGROUP_DIR/memory.peak" > "$root/peak-before.bytes"
(
  while true; do
    printf '%s\t%s\n' "$(date -u +%FT%TZ)" "$(cat "$CGROUP_DIR/memory.current")"
    sleep 1
  done
) > "$root/host.tsv" & host_pid=$!
nvidia-smi -i "$GPU_ID" --query-gpu=timestamp,uuid,memory.used,memory.total \
  --format=csv -l 1 > "$root/gpu.csv" 2> "$root/gpu-monitor.err" & gpu_pid=$!
cleanup() { kill "$host_pid" "$gpu_pid" 2>/dev/null || true; wait "$host_pid" "$gpu_
pid" 2>/dev/null || true; }
trap cleanup EXIT
status=0
/usr/bin/time -v -o "$root/process-time.txt" \
  python -u infer.py --config config.json > "$root/phases.log" 2>&1 || status=$?
cat "$CGROUP_DIR/memory.peak" > "$root/peak-after.bytes"
printf '%s\n' "$status" > "$root/exit-code.txt"
test -s "$root/host.tsv"
test -s "$root/gpu.csv"
exit "$status"
```

用小对照找到缓存来源

先读退出码、阶段日志和监测错误，GPU表只有表头或N/A不能算有效测量。将主机字节统一换算GiB，再与设备MiB分开画图。time的MaxRSS也保留，但不能把它当所有worker同时占用的物理内存总量；cgroup内存还可能含文件页缓存，需结合memory.stat分解。

对同一代表输入先把num_workers设为0，再试小的正数；正数时记录prefetch_factor、pin_memory、persistent_workers和batch size。PyTorch说明多worker可能复制或扩大父进程Python对象占用，预取和锁页也有主机代价。一次只改一项，并比较完整时间、输出数与结果一致性，不因RAM下降就认定分析正确。

加载阶段暴涨先查是否一次读完整数据集、重复构造对象或保留全部预测结果；首批后持续上涨再查队列与列表缓存。设备显存空余无法补足这些主机对象。memory.events中的oom/oom_kill需结合调度器记录确认，退出137本身不能唯一诊断内存原因。

每次用独立新任务记录输入数量、总量、配置、主机峰值、显存和耗时。若必要缓存策略已调整，真实主机峰值仍超过获配容量，再评估RAM；只有显存不足才按模型与设备需求另算。监测脚本的成功退出不替代模型输出与科学结果验收。

把监测对象限定为一个作业

首先确认运行方式：单机进程、容器、Slurm作业或其他调度器的统计口径并不相同。cgroup必须对应当前作业并包含其worker；若把父级服务的所有子任务都计入，memory.current可能包含其他项目。将作业ID、cgroup路径、GPU UUID、容器ID、软件版本、输入指纹与开始时间一起记录。容器内看到的路径可能经过命名空间映射，不能只凭名字相似就认定读取到了目标作业统计。

推荐在作业启动后由同一个调度脚本记录本次作业的cgroup路径，退出时一并保存调度器Accounting。不要预先记住一个可复用的cgroup目录：任务结束后它可能被删除或重用，memory.peak也可能是旧任务的峰值。若调度器已有可信的每作业内存峰值和GPU分配记录，优先保存原生作业记录，再用本篇采样补充阶段信息。

在推理代码中标记可比较阶段

为模型加载、数据解码、第一批推理、稳定批次和结果写出分别输出时间戳。阶段日志写stderr或单独文件，不要改变模型参数、批量大小或worker数量来“方便监测”。先跑一个固定输入，记录每阶段处理样本数和失败状态。若DataLoader有预取队列，首批之后的主机内存可能仍然上升；因此需要观察稳定阶段和写出阶段，而不是加载完成后立即结束采样。

GPU内存由驱动工具报告，但共享设备的总使用量可能包含其他账号。要根据管理员提供的GPU UUID确认任务分配，并在可能时记录进程PID与设备关联。CUDA_VISIBLE_DEVICES的逻辑编号可能被重新映射，不能直接把逻辑GPU 0写成物理卡0。若作业使用MIG或容器设备隔离，应分别记录实例标识和可见设备信息。

怎样理解两种内存曲线

现象	优先检查	不要立即推断
主机内存上升、GPU稳定	解码缓存、worker、预取和对象复制	不能仅凭此调大GPU显存
GPU内存接近上限、主机余量足	批量大小、模型激活与其他进程	不能把主机空闲RAM视为可替代显存
两者都上升	批次处理和缓存是否持续累积	不一定是单一泄漏，也可能为正常预热
曲线正常但作业被杀	cgroup限制、调度器记录及采样间隔	一秒采样可能漏掉短峰值

memory.current包含cgroup统计口径内的当前内存，并非Python进程RSS简单求和；memory.peak是对应cgroup自创建以来的峰值。部分内核或控制组环境不提供同名文件，需要据cgroup版本查看可用字段。NVIDIA工具的MiB与cgroup

字节不能直接逐数相加；两者代表不同设备与内存空间。记录单位、查询时间和来源能避免把格式差异误当资源变化。

采样结果如何变成下一次试验

先按同一输入和版本重复一到两次，确认曲线形状是否稳定。若主机峰值发生在数据解码阶段，可尝试在小输入上降低worker数或预取深度，并每次只改一个设置；若峰值发生于结果累积阶段，检查是否将所有预测结果长期保留在内存。GPU峰值发生于模型初始化或大批次时，比较官方支持的批量参数与设备可用显存。所有修改都要保存命令、配置差异和输出验收，避免以一次偶然的低峰值作配置保证。

同时记录吞吐量、失败率、读写速率与等待时间。只降低内存而让吞吐降到不可接受，不是完整优化；只提高GPU占用也不等于分析更快。若CPU内存高但任务可正常完成，可能是可预期缓存；只有接近限制、发生swap/OOM、降低可用作业数量或导致失败时，才构成需处理的容量问题。

复核常见故障并形成报告

报告至少写明作业范围、软件和驱动版本、设备UUID、cgroup路径、采样周期、每阶段时间、主机与GPU峰值、调度器退出状态、输入规模、输出检查及监测盲点。若监控进程与推理作业处于同一cgroup，它本身也会增加少量内存；可在报告中注明，或由调度器提供外部采样。权限不足时应请求管理员导出对应作业统计，不要改写cgroup控制文件或尝试绕过设备隔离。

如果差异无法解释，依次确认cgroup是否包含worker、采样文件是否来自当前任务、共享GPU是否有其他使用者、阶段标记是否落在正确位置，以及失败作业是否由调度器强制终止。下一步配置应以代表性负载峰值、并发目标和调度策略共同判断。GPU算力租赁只在模型确实需要GPU且设备内存、驱动与软件栈兼容时才值得比较；若瓶颈在CPU内存、I/O或缓存，先修正对应环节。

把曲线整理成可比较的阶段记录

每条样本记录可以包含timestamp、job_id、stage、cgroup_current_bytes、cgroup_peak_bytes、gpu_uuid、gpu_used_mib、processed_items和error_count。GPU字段允许缺失并注明原因，例如工具无权限或设备未分配；不要用0替代“未知”，否则图表会误导读者认为设备当时没有使用量。时间使用带时区的ISO 8601格式，所有曲线以同一时钟为准。样本或模型名称可用项目内部代号，不把敏感路径写进公开日志。

绘图时把主机内存和GPU内存分成上下两幅图，单位分别固定为GiB与MiB或统一换算后明确标注；以阶段标记画垂直线，叠加任务限制线而非整机物理内存线。不同任务比较时同时标出输入条数、批量大小、worker数和输出数量。图上峰值属于该次运行，不应直接写成软件固定需求；更可靠的结论是在哪个阶段、哪种配置下观察到怎样的资源曲线。

用受控小改动定位CPU内存来源

如主机内存是主要限制，可在同一小输入上分别运行原设置、较少DataLoader worker、较小预取队列或分块写出设置。一次只改一个因素，并固定模型权重、批量大小和随机种子（若适用）。比较每阶段内存峰值、样本吞吐和输出一致性。worker降低后若内存峰值下降但数据加载变慢，团队需按时限权衡；若内存仍单调上升，则检查结果列表、日志缓冲或未释放张量，而不是继续盲目扩容。

每个尝试都保存配置快照与命令，不直接修改唯一的生产配置。清理GPU缓存只能释放未引用的CUDA缓存块，不会修复仍被Python对象引用的张量，也不增加主机RAM上限。执行清理操作前，需确认应用框架支持且不会影响其他共享进程。发生OOM时保存stderr、内核或调度器事件及退出码，不能因监测脚本自身继续运行就把作业记为成功。

如果多个作业共用同一GPU，设备总显存曲线不能归因到单个任务；使用管理员提供的进程级监控或调度器隔离记录。若不能区分任务，报告应明确这一限制，不用GPU总占用推断本任务峰值。最终租赁或采购配置应参考隔离任务的测量或合同约定的分配额度，保留并发和模型变化带来的余量。

识别监测工具本身的盲点

1秒轮询适合观察分钟级阶段变化，但很短的峰值可能发生在两个采样点之间。若作业即将达到硬内存上限，调度器的MaxRSS或内核OOM记录通常比用户态轮询更重要。采样进程还可能因权限、cgroup目录消失、NVIDIA查询超时或驱动错误中断；每种异常都应写入monitor_error字段并继续保存已经收集的证据。最终图表标注实际采样间隔、缺失段和共享设备限制，读者才能判断峰值精度。监控不到数据不等于资源使用为零，也不应据此调高配置。

为比较配置，可计算每阶段峰值与本作业memory.max的比例，同时报告GPU分配额度与观测峰值；只在调度器确认该限额确实属于本作业时使用。比例接近边界时先检查采样缺口和并发变化，再进行低风险复测，不能在正式大任务上故意触发OOM来验证上限。

采样频率和监控开销也需写入报告；当作业很短或极快释放缓存时，系统级峰值统计更有代表性。

配套练习包

本PDF对应的ZIP包含本文代码块、主题练习模板或合成数据、README和SHA256校验文件。请从同一教程页面下载练习包；先核对校验值，再在独立目录复现。

参考资料

Linux cgroup v2内存接口 · 来源资料

PyTorch DataLoader参数与多进程内存说明 · 来源资料

NVIDIA nvidia-smi查询接口 · 来源资料

操作建议：先在合成小样本上验证流程，再依据项目参考、标签合同、存储策略与获配资源验收真实数据。