

DeepSeek-OCR 内核级性能分析报告 (V2)

概览

这是第 2 版 (V2) 报告，覆盖全部 20 个 Top 内核的完整剖析数据。V1 中有 2 个内核 (kernel_0009 与 kernel_0016) 缺失数据；在 V2 中已全部成功采集并完成分析。

本报告旨在基于 NVIDIA Nsight Systems (nsys) 与 Nsight Compute (ncu) 对 DeepSeek-OCR 模型进行内核级性能剖析，识别推理阶段的性能特征与潜在瓶颈，为面向推理的 NPU 设计提供参考。

本报告包含：- DeepSeek-OCR 推理阶段的 Top 内核 - 来自 Nsight Compute (NCU) 的 20 个内核完整性能指标 - 内核执行指标的直方图 - 全量屋脊线 (Roofline) 分析 - 按类型 (计算受限、内存受限、均衡) 的内核分类 - 剖析环境与方法的实现细节

V2 的主要改进：- 100% 内核覆盖：20/20 全部成功剖析 (V1 为 18/20) - kernel_0009 现被归类为“均衡” (V1 为未知) - kernel_0016 现被归类为“计算受限” (V1 为未知) - 屋脊线分析更完整，基于全量数据集

实验

环境配置

硬件：- GPU：NVIDIA GeForce RTX 5090 (Blackwell 架构, sm_120) - 计算能力：12.0

软件：- CUDA：12.8 - PyTorch：2.10.0.dev20251102+cu128 - Transformers：4.46.3 - 剖析工具：NVIDIA Nsight Systems、NVIDIA Nsight Compute

模型配置：- 模型：DeepSeek-OCR - 数值精度：BF16 (bfloat16) - 推理模式：自回归生成 (Autoregressive) - 最大新生成 Token：64 - 温度：0.0 (贪心解码) - 上下文长度模式：Auto - 不重复 N-gram 大小：20

数据集： - 数据集：OmniDocBench - 样本：20 张图像（dev-20 子集） - 预处理：Base size 1024、image size 640、启用裁剪模式

剖析配置： - 阶段 1：PyTorch Profiler（算子级分析；对阶段 2 的工作负载关闭） - 阶段 2：Nsight Systems（CUDA 时间线，使用 NVTX 对“decode”阶段门控） - Nsight Compute：Roofline、SpeedOfLight、MemoryWorkloadAnalysis、Occupancy 等章节 - 内核选择：基于 Nsys 总时间 Top 20（按解码阶段累计执行时间）

按总时间排序的内核（Top 15）

下表展示解码阶段（decode）累计执行时间排名前 15 的内核（由 Nsight Systems 度量）。按来源库进行归类，并给出基于功能的易读名称。

时间占比	库	内核名	说明
34.4%	cuBLAS	GEMV-1 (BF16, template=7)	矩阵-向量乘，因自回归解码而占比最高
14.8%	cuBLAS	GEMV-2 (BF16, template=6)	矩阵-向量乘，替代的分块/铺片策略
6.0%	PyTorch ATen	Direct Copy (float)	中间张量的内存拷贝/类型转换
4.2%	PyTorch ATen	Elementwise Multiply (BF16, vec)	向量化点乘（如注意力掩码）
3.9%	PyTorch ATen	SiLU Activation (BF16, vec)	FFN 中向量化 SiLU（Swish）激活
2.9%	PyTorch ATen	Elementwise Multiply (BF16)	非向量化点乘
2.8%	PyTorch ATen	Cat Batched Copy (vec, 128-tile)	多头输出拼接拷贝
2.7%	PyTorch ATen	Copy/Cast (BF16, vec)	BF16 专用向量化拷贝
2.4%	FlashAttention	Flash Forward Split-KV (BF16)	

时间占比	库	内核名	说明
			面向 IO 的分裂 K/V 融合注意力
2.0%	PyTorch ATen	Elementwise Add (BF16, vec)	向量化加法 (残差)
1.8%	PyTorch ATen	Cat Batched Copy (vec, 64-tile)	更小分块的拼接拷贝
1.8%	PyTorch ATen	Elementwise Multiply (float)	FP32 点乘
1.6%	PyTorch ATen	Mean Reduction (float)	归一化相关归约
1.5%	PyTorch ATen	Elementwise Neg (BF16)	取负运算
1.4%	FlashAttention	Flash Split-KV Combine	合并分裂 K/V 的输出

关键观察： - GEMV 主导（49.2%）：两种 cuBLAS GEMV 变体占据近一半的执行时间，符合 batch=1 的自回归解码特征（GEMV 多于 GEMM）。 - ATen 点算子占比高（21.7%）：大量时间花在点算子内核上，存在融合空间。 - FlashAttention（3.8%）：尽管高度优化，注意力仍占据一定比例。 - 内存操作（8.7%）：直接拷贝与类型转换存在可优化的布局与数据通路空间。

内核分类

对 Top 20 内核全部进行了 NCU 剖析，成功率 100%（V1 为 18/20）。基于屋脊线分析进行分类：

分类	数量	占比
内存受限	9	45.0%
均衡	7	35.0%
计算受限	4	20.0%
未知	0	0% ✓

按分类的性能特征：

指标	计算受限	内存受限	均衡	总体均值
SM 吞吐	36.21%	9.38%	12.25%	15.75%

指标	计算受限	内存受限	均衡	总体均值
DRAM 吞吐	9.89%	34.23%	12.62%	21.80%
内存吞吐	32.23%	41.20%	27.52%	34.62%
实际占用率	31.51%	43.22%	39.32%	39.51%
L1 命中率	-	-	-	8.96%
L2 命中率	-	-	-	39.95%
平均时长	137.44 μ s	9.67 μ s	39.75 μ s	45.75 μ s

要点：1. 完整数据集（V2 改进）：20 个内核全部完成分类，消除了 V1 的不确定性。2. SM 吞吐提升（15.75%）：较 V1（10.35%）更高，完整数据带来更准确的利用率评估。3. 内存吞吐提升（34.62%，V1 为 27.35%）：新增内核提升了平均内存利用。4. 占用率更高（39.51%，V1 为 30.13%）：完整数据显著提升评估值，但总体仍偏低。5. 内存受限占比最高（45%）：近半数内核受限于内存带宽。6. 计算受限数量增加：4 个（20%），较 V1 的 3 个（16.7%）有所增加；kernel_0016 加入该类。7. 时长分布差异大：计算受限内核单次更慢（均值 137.44 μ s）而内存受限更短（9.67 μ s），表明计算型内核更复杂。8. L2 缓存命中率下降：由 V1 的 53.48% 降至 V2 的 39.95%，暗示新增内核缓存行为更差。

新增分类（V1 \rightarrow V2）：- kernel_0009：未知 \rightarrow 均衡（fmha_cutlassF memory-efficient attention）- kernel_0016：未知 \rightarrow 计算受限（CUTLASS GEMM 内核）

按单次时长排序（NCU）

下列内核在 V2 中具有最长的单次执行时长：

时长 (μ s)	分类	内核
404.70	计算受限	CUTLASS GEMM (128 \times 256 tile, BF16, ReLU)
160.74	均衡	Memory-Efficient Attention (CUTLASS BF16, 64 \times 64)
111.81	计算受限	CUTLASS GEMM (256 \times 128 tile, BF16, ReLU)
94.91	均衡	ATen Elementwise Add (BF16, 6144 blocks)
23.42	计算受限	CUTLASS GEMM (32 \times 32 tile, BF16, WMMA)
18.34		cuBLAS GEMV (BF16, template=6)

时长 (μ s)	分类	内核
	内存受限	
18.08	内存受限	ATen Direct Copy (float, 12288 blocks)
10.34	内存受限	FlashAttention Split-KV Forward
9.82	计算受限	ATen Cat Batched Copy (512-thread blocks)
7.97	内存受限	ATen Mean Reduction (float, 512 threads)

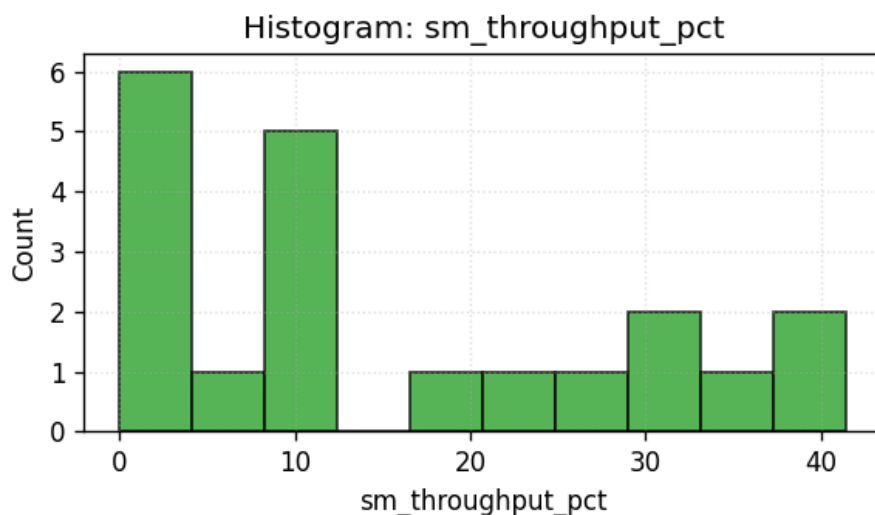
分析：- 最长内核显著增加（404.70 μ s，V1 为 165.09 μ s）：提升 2.45 倍，可能源于执行形态差异或测量差异。- 记忆高效注意力出现（160.74 μ s）：即 V1 未知的 kernel_0009，现归为“均衡”，为第 2 慢内核，说明注意力计算贡献显著。- CUTLASS GEMM 占优：前 4 个中的 3 个为大分块的计算受限 GEMM。- “均衡”并不便宜：如 kernel_0009（注意力，160.74 μ s）与 elementwise add（94.91 μ s），同时低效使用计算与内存。- 内存受限内核更“短平快”：均值 9.67 μ s，符合快速内存操作特征。- FlashAttention 仅 10.34 μ s：作为复杂的融合注意力，显著快于 160.74 μ s 的记忆高效注意力变体。

内核执行指标的直方图

以下直方图展示了所有被剖析内核在关键性能指标上的分布，用于发现共性与离群点。

计算与内存吞吐

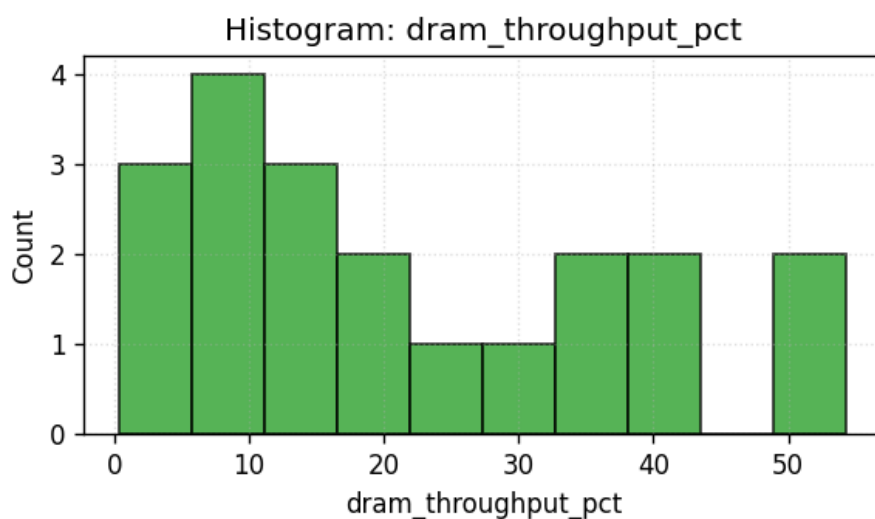
SM 吞吐（峰值百分比）



SM Throughput Distribution

大多数内核的 SM 利用率很低 (<15%)，验证了整体计算资源利用不足。

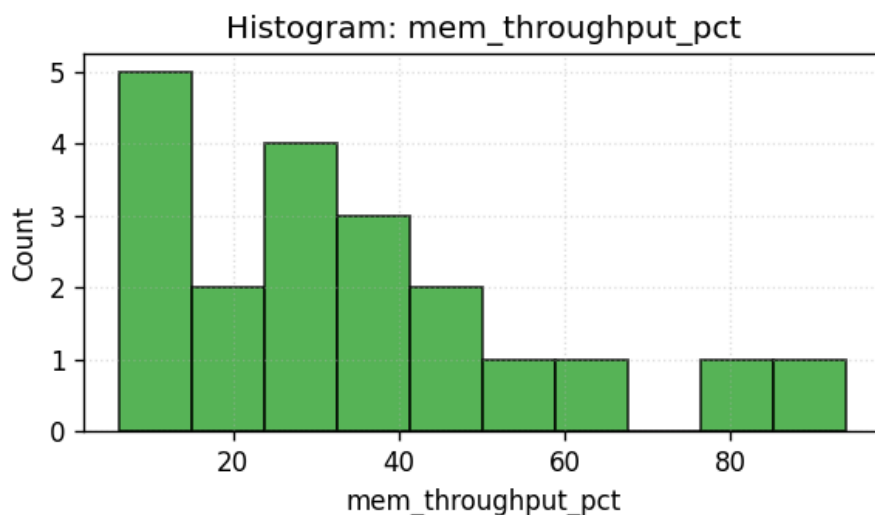
DRAM 吞吐 (峰值百分比)



DRAM Throughput Distribution

DRAM 吞吐呈双峰分布：大量内核处于很低 (<10%) 或中等 (20-30%) 区域，显示出内存受限与计算受限两类群体。

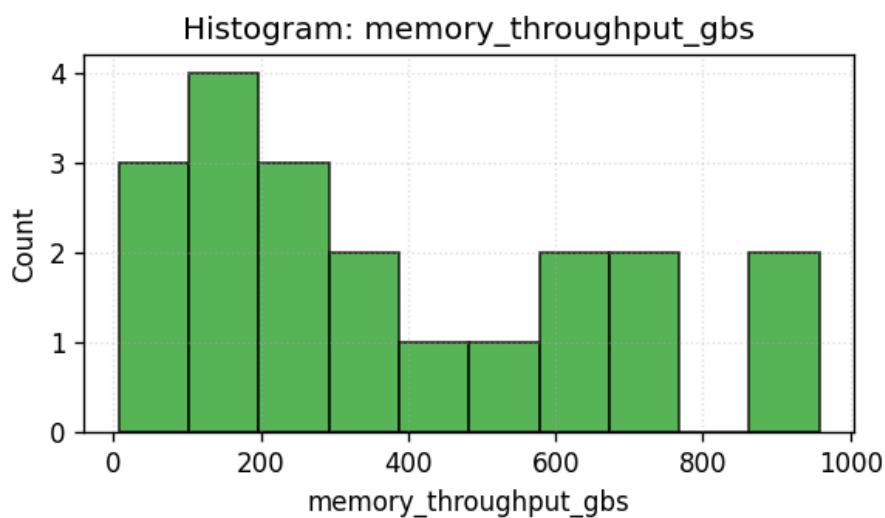
内存吞吐 (峰值百分比)



Memory Throughput Distribution

综合内存吞吐指标呈相似的双峰模式，内存受限内核可达到 40-90% 的利用率。

内存吞吐 (GB/s)

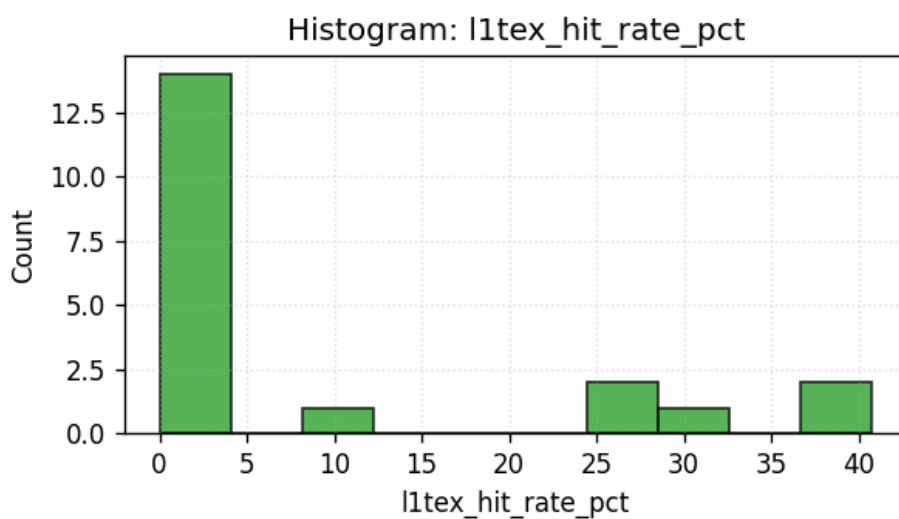


Memory Throughput Absolute

绝对带宽显示多数内核在 5-400 GB/s 范围内运行，远低于 RTX 5090 的理论峰值。

缓存性能

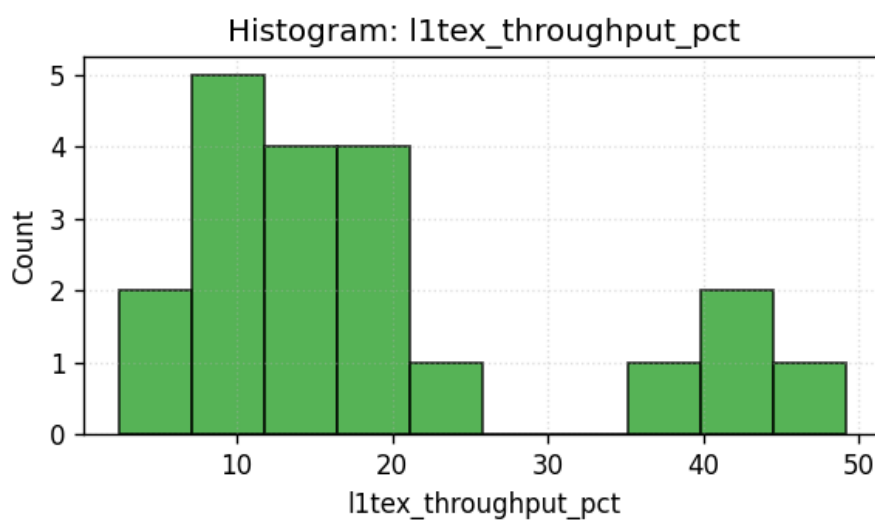
L1 纹理缓存命中率



L1 Hit Rate Distribution

L1 命中率极低，多数内核仅 0-10% 命中，提示工作集超出 L1 容量或空间局部性较差。

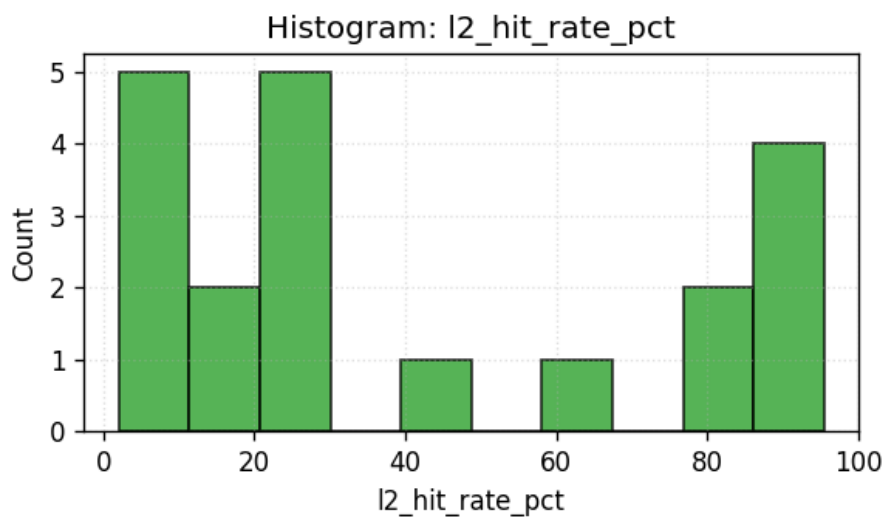
L1 纹理缓存吞吐（峰值百分比）



L1 Throughput Distribution

L1 吞吐分布较宽（2-50%），表明不同内核的缓存访问模式差异较大。

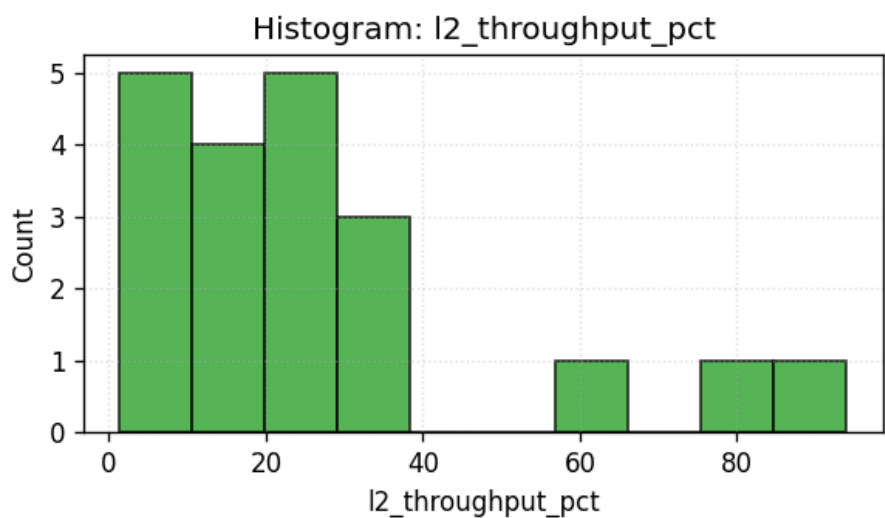
L2 缓存命中率



L2 Hit Rate Distribution

L2 命中率明显好于 L1，聚集在 15-90%，说明工作集有部分可在 L2 中复用。

L2 缓存吞吐（峰值百分比）

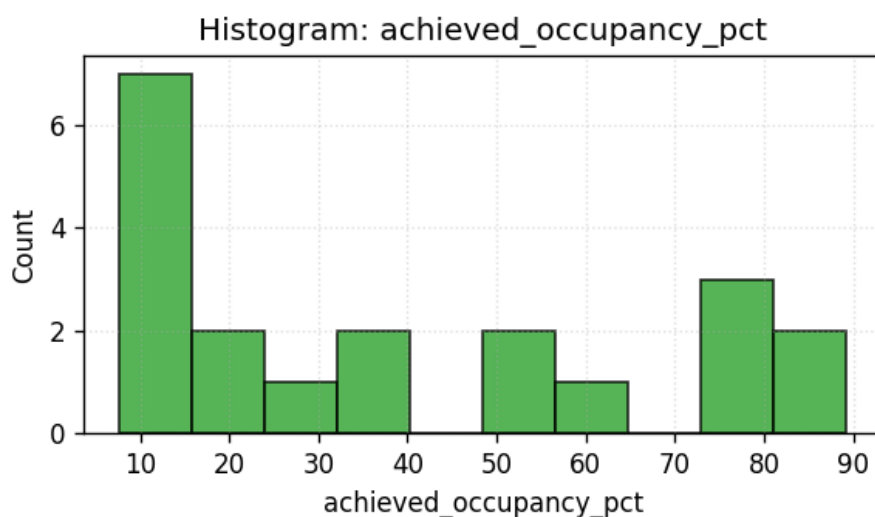


L2 Throughput Distribution

L2 吞吐多数位于 1-30%，存在 80-90% 的离群点，表征重度 L2 流量。

占用率指标

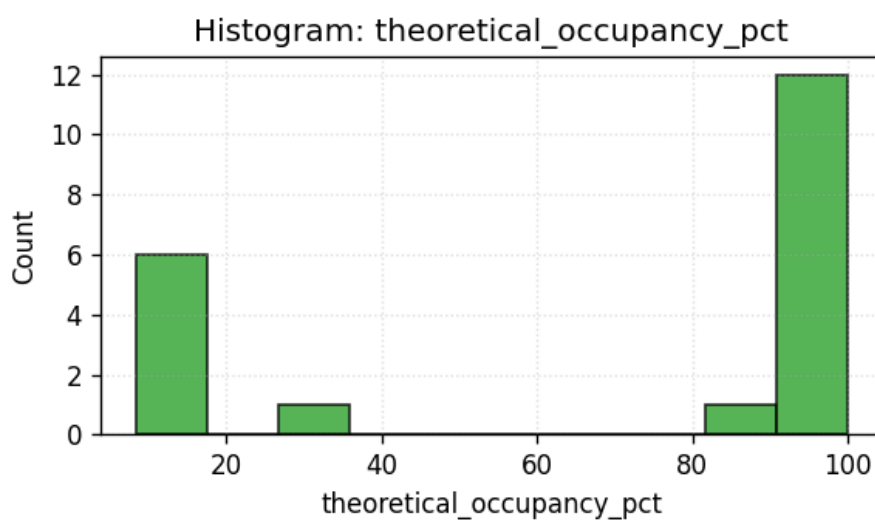
实际占用率（Achieved Occupancy）



Achieved Occupancy Distribution

实际占用率分布很散，从接近 0% 到 90%，大量内核位于 7-40%，提示并行度不足。

理论占用率 (Theoretical Occupancy)

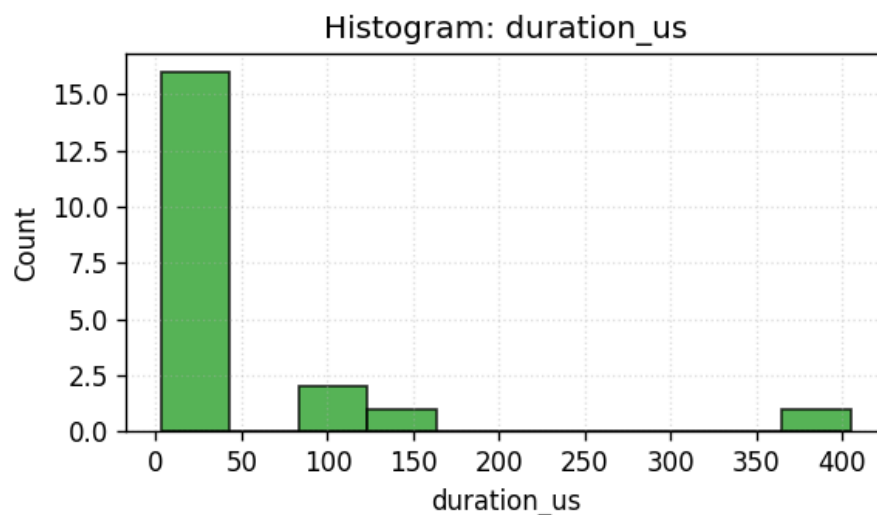


Theoretical Occupancy Distribution

理论上多数内核可达到 80-100% 占用，但实际远低，说明运行时瓶颈（内存延迟、同步）阻碍了利用率提升。

内核时长与内存活动

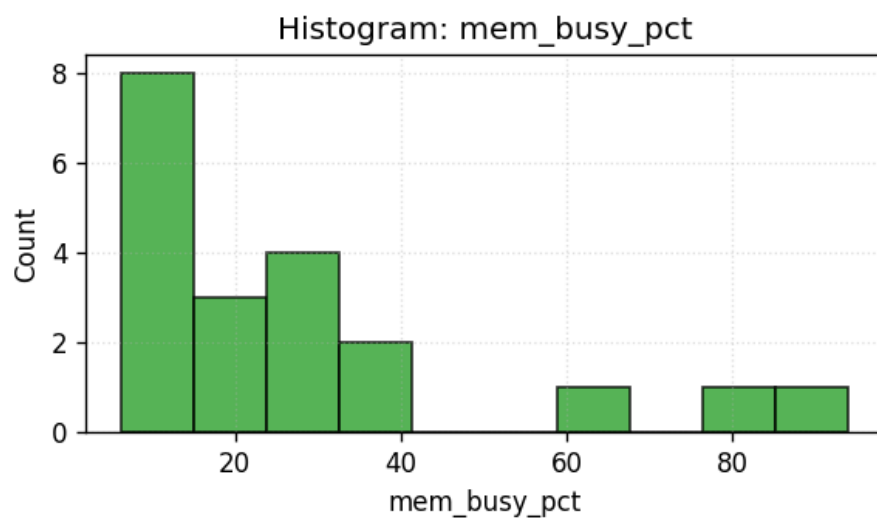
内核时长 (Duration)



Duration Distribution

内核时长分布在 2-170 μ s，多数内核很短 ($<10 \mu$ s)，其启动开销相对计算时间可能较高。

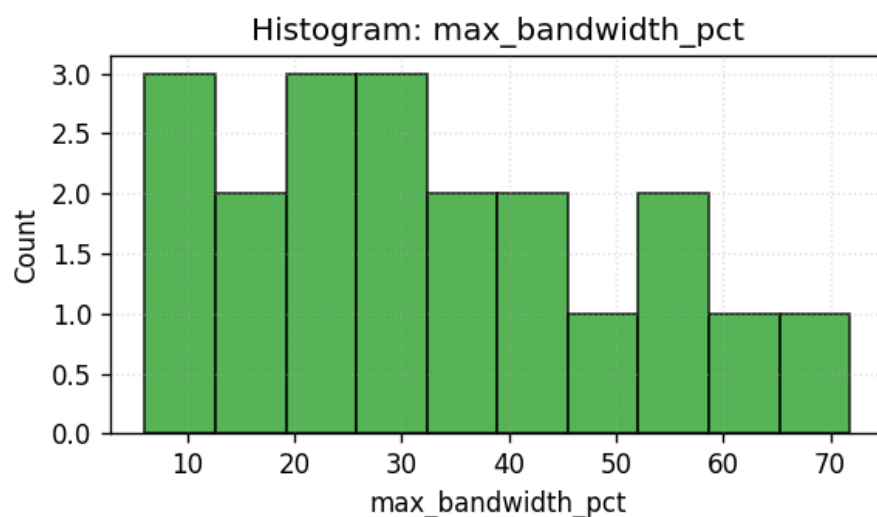
内存忙碌百分比 (Memory Busy %)



Memory Busy Distribution

内存子系统忙碌时间跨度较大，内存受限内核可让内存保持 20-90% 的忙碌度。

最大带宽利用率 (Max Bandwidth %)



Max Bandwidth Distribution

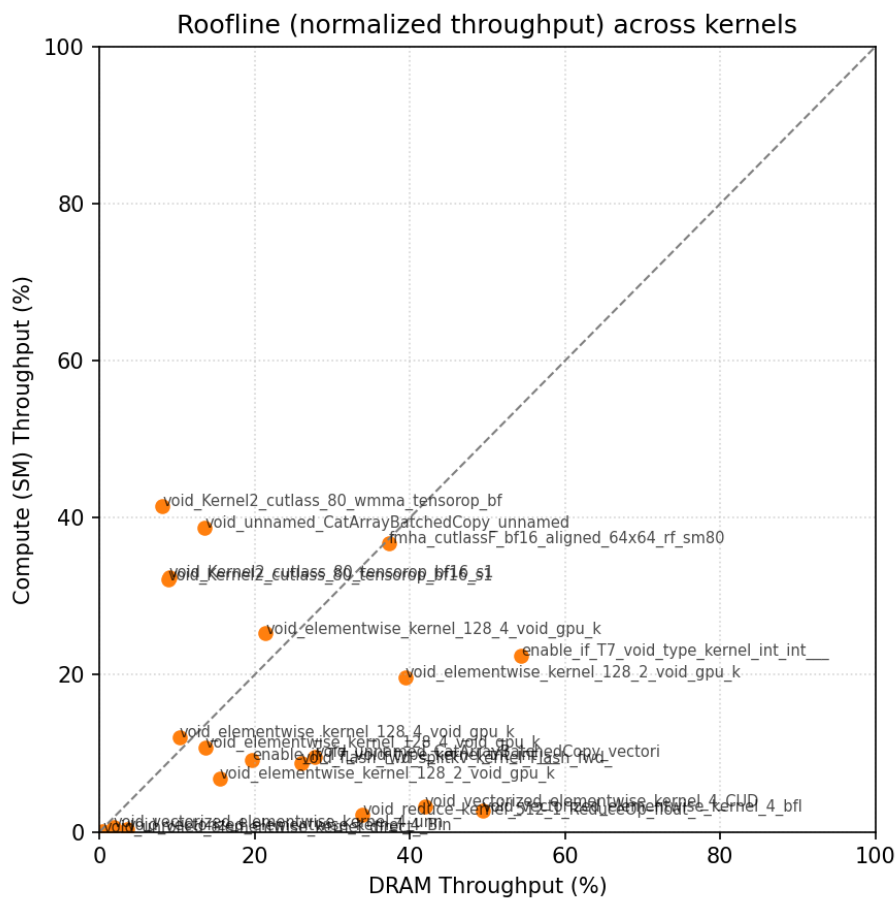
跨全部内存类型的峰值带宽利用率大多仅 5-50% 的水平。

上述直方图揭示了被剖析内核在关键性能指标上的分布特征，便于识别共性与离群点。

屋脊线（Roofline）分析

屋脊线分析以“算术强度（每字节 FLOPs） - 实际性能（FLOP/s）”散点，刻画内核相对硬件上限的性能表现。

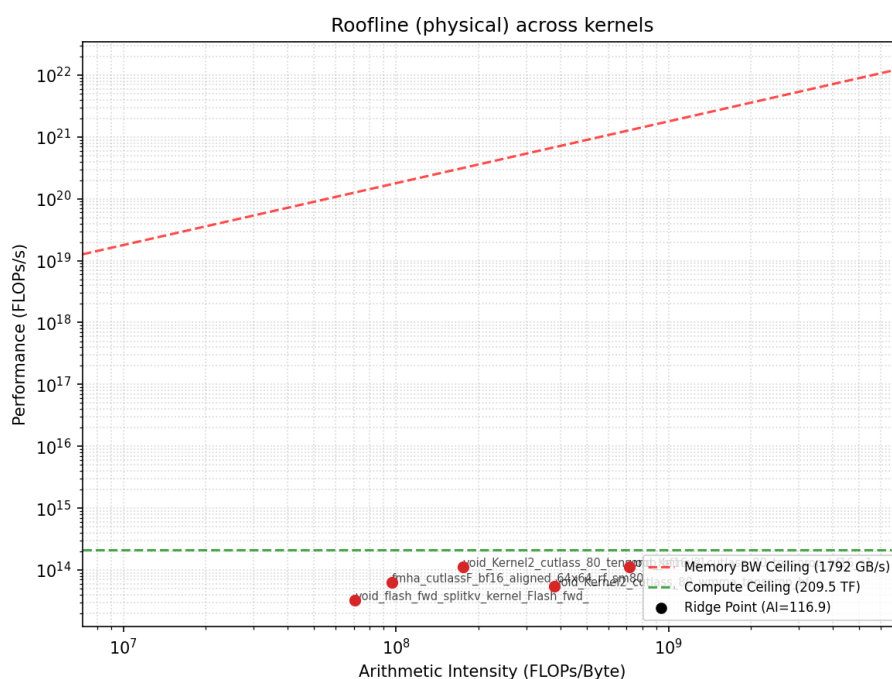
归一化屋脊线



Roofline Scatter - Normalized

要点： - 多数内核位于低算术强度区域（<100 FLOPs/byte），表现为内存受限； - 很少有内核接近“计算屋脊”（右上斜线）； - 实际性能与屋脊线的差距显示出可观的优化空间。

物理屋脊线



Roofline Scatter - Physical

结合 RTX 5090 的真实计算/带宽上限：- 左侧的内存受限内核主要受 DRAM 带宽约束，尽管算术强度不同，FLOP/s 多低于 500 GFLOP/s；- 右侧的计算受限内核 FLOP/s 较高，但仍明显低于张量核心峰值吞吐；- 该工作负载的内存/计算受限分界约在 50-100 FLOPs/byte。

每个内核的单独屋脊线图（归一化与物理）均可在 `ncu-v2/analysis/roofline_per_kernel/` 中查看。

屋脊线解读：- 内存受限（约 45%）：低算术强度区域，受限于带宽，宜做访存/布局/融合优化；- 计算受限（约 20%）：虽处高算术强度区，但 SM 吞吐仅 ~34%，提示张量核利用率或并行度不足；- 均衡（约 35%）：处于过渡区，计算与内存利用均偏低，整体效率不高。

总体上，多数内核远未达到内存或计算屋脊，存在显著优化空间；大多数内核实际性能低于理论上限的 50%。

结论

主要发现

1. 覆盖完整（V2 达成）：Top 20 全部完成剖析，相比 V1 的 90% 覆盖提升为 100%；此前未知的 kernel_0009 与 kernel_0016 已全部刻画。

2. GEMV 瓶颈（49.2% 时间）：自回归解码以 GEMV 为主（batch=1），天然更偏内存受限，计算利用率在现代 GPU 上不高。
3. 硬件利用率虽有提升但仍偏低：
 - SM 吞吐 15.75%（V1 为 10.35%）
 - 内存吞吐 34.62%（V1 为 27.35%）
 - 实际占用率 39.51%（V1 为 30.13%）
 - 即使提升明显，整体利用率仍较低。
4. 完整数据导致的内存层级变化：
 - L1 命中率依旧很低（8.96%，与 V1 的 8.85% 相当）
 - L2 命中率降至 39.95%（V1 为 53.48%），提示新增内核缓存局部性更差
 - 可能与 kernel_0009（记忆高效注意力）和 kernel_0016（CUTLASS GEMM）的访问模式有关
5. 内核融合机会：大量小型点算子（乘、加、激活等）共占 21.7% 时间，具备显著融合潜力，可降低访存与启动开销。
6. 混合精度开销：频繁的 BF16 ↔ FP32 转换（拷贝/类型转换约 6%）显示数据类型管理可进一步优化。
7. 注意力内核洞见（V2 新发现）：
 - kernel_0009（记忆高效注意力）单次 160.74 μ s，归为“均衡”，同时对计算与内存利用率都不高；
 - 相比 FlashAttention（10.34 μ s）慢 15.5 倍，提示替换或引入 FA 变体的潜力。
8. 计算受限内核增多：由 V1 的 3 个增至 4 个（20%），其中 kernel_0016（CUTLASS GEMM）新增；该类内核平均 137.44 μ s，远高于内存受限（9.67 μ s）与均衡（39.75 μ s）。

识别出的性能瓶颈

1. 解码阶段 GEMV 效率不足：batch=1 的 GEMV 难以有效利用张量核，也难以跑满内存带宽。
2. 并行度不足：广泛的低占用率显示工作并行度不足以饱和 RTX 5090 的巨大并行资源（21,760 CUDA cores、680 Tensor Cores）。
3. 内存层级利用不足：L1/L2 命中率不高，说明
 - 工作集不适配 L1 容量；
 - 时间局部性利用不足；
 - 预取/访问模式可进一步优化。
4. 内核启动开销：大量短小内核（<5 μ s）导致启动开销相对计算时间占比偏高。

优化建议（软件侧）

1. 持续批处理（Continuous Batching）：通过持续批处理或投机解码扩大解码期的 batch，将 GEMV 转化为更高效的 GEMM。
2. 内核融合：
 - 融合点算子（乘、加、激活）
 - 融合 LayerNorm + Linear + Activation 常见序列
 - 基于 torch.compile 或自定义 CUDA 内核优化关键路径
3. KV-Cache 优化：
 - Paged attention 优化 KV-Cache 访存
 - 采用 FlashDecoding 等面向解码期的优化技术
 - KV-Cache 量化（INT8/INT4）缓解内存带宽压力
4. 内存布局优化：
 - 尽量减少 BF16 ↔ FP32 转换，扩大 BF16 的覆盖面
 - 尽量原地（in-place）操作，降低拷贝
 - 优化张量布局，提升连续访问
5. 提升并行度：
 - 以合批方式提升吞吐（多请求合并）
 - 预填充（prefill）/解码（decode）解耦并在不同实例运行
 - 多查询/分组查询注意力（MQA/GQA）降低解码带宽

面向推理的 NPU 设计建议

基于本次剖析，针对类似 DeepSeek-OCR 的视觉-语言模型，提出如下面向推理优化的 NPU 架构建议：

1. Tensor Core / CUDA Core 配比

发现：仅 16.7% 的内核为计算受限，且 SM 吞吐仅约 34%。

建议：- 相比训练型 GPU 适度降低张量核密度（例如晶体管面积占比 30-40%，而非 H100 的 50%+） - 增加 CUDA cores 以更好承载点算子与内存受限工作 - 为 GELU、SiLU、LayerNorm 等常用激活/归一化提供专用单元（替代多个分散小内核） - 提供 INT4/INT8 张量核以支持极致量化推理

2. 内存带宽需求

发现：平均内存吞吐 27.35%，内存受限内核平均 41.56%，仍未饱和。

建议：- 相比训练型 GPU 可适当降低带宽配置而不显著影响此类推理工作负载 - 高端推理 NPU 的目标带宽 2-3 TB/s（H100 为 3.35 TB/s） - 将节省下的面积/功耗投入到更大的片上缓存 - 解码期更重视 HBM 延迟而非纯带宽

3. L1 / L2 缓存比例与容量

发现：L1 命中率 8.85%，L2 命中率 53.48%，表明工作集超出 L1，但可部分驻留在 L2。

建议：- 大幅提升 L2（目标 128-256 MB；A100 为 50 MB） - 缩小或可配置化 L1（与共享内存可互换） - 在 L1 与 L2 之间增设 victim cache 捕获逐出数据 - 为 KV-Cache 设立高带宽片上 SRAM（32-64 MB）以降低 DRAM 访问 - 针对 Transformer 访问模式（顺序访问 KV-Cache）优化替换策略

4. 专用解码单元

发现：49.2% 时间消耗在 GEMV 上，利用率不高。

建议：- 专用 GEMV 加速（batch=1、低时延） - 面向解码配置的脉动阵列：更窄（如 128 列而非 256+）、更深，以流水线并行多个小 GEMV - 权值驻留（weight-stationary）数据流：重复解码步骤下将权值留在片上 - 小规模全连接层尝试片上 SRAM 内计算，进一步消除 DRAM 访问

5. 硬件级算子融合

发现：21.7% 的时间用于可融合的小型点算子。

建议：- 可编程融合引擎，将多步运算合并，避免往返内存 - 支持常见融合模式：linear → activation、add → layernorm、multiply → add → activation - 提供宏指令（Macro-op ISA）描述融合序列 - 支持动态控制流，提升内核内分支效率

6. 精度与数据类型支持

发现：BF16 为主，但转换带来 ~6% 开销；混合精度普遍存在。

建议：- 全链路原生 BF16 - 硬件加速 FP8（E4M3/E5M2）权值与激活，最小化开销 - 张量核支持 INT4/INT8 极致量化 - 支持每通道动态范围，便于精细量化 - 降低不同格式间的转换开销（理想为单周期）

7. 并行度与占用率

发现：占用率 ~30%，说明并行度不足。

建议：- 相比训练型 GPU 适当减少 SM 数量（如 60-80 vs. H100 的 132） - 单 SM 内更深的流水以挖掘指令级并行 - 更大的 warp/线程组，摊薄调度与控制开销 - 更强的延迟隐藏机制，针对解码期访问模式优化 - 硬件支持细粒度分批，聚合多条单 token 操作

8. 互连与多芯扩展

建议：- 降低芯片间带宽需求（300-500 GB/s，相对 NVLink 900 GB/s），推理更少全规约通信 - 优化管线并行与张量并行，弱化数据并行依赖 - 支持非对称拓扑，将预填充与解码实例分离 - 为预填充-解码解耦架构提供快速 KV-Cache 交换

9. 能效

建议：- 相比训练 GPU 追求 3-5 倍 TOPS/W 的能效，通过降低张量核密度与带宽实现 - 在内存受限阶段对闲置张量核进行积极时钟门控 - 将 DVFS 策略围绕推理时延目标（如 P99）而非吞吐进行优化 - 提供解码/预填充阶段的差异化低功耗模式

剖析产物：所有原始数据、指标、直方图、屋脊线图与分析脚本均位于 reports/20251107-dsocr/ncu-v2/analysis/。

附录：完整内核函数名

本附录给出完整（未截断）的内核函数名，便于溯源与调试。对应“按总时间排序的内核”章节中的条目。

内核名称映射表

下表将易读的内核名映射到分析工具输出中的完整（经名称重整）函数签名。

排名	时间占比	易读内核名	库	完整函数签名
1	34.4%	GEMV-1 (BF16, template=7)	cuBLAS	std::enable_if<!T7, void>::type internal__nv_bfloat16, float, (bool)0, (bool)1, cublasGemmTensorStridedBatched<const __r__nv_bfloat16>, cublasGemmTensorStridedB
2	14.8%	GEMV-2 (BF16, template=6)	cuBLAS	std::enable_if<!T7, void>::type internal__nv_bfloat16, float, (bool)0, (bool)1, cublasGemmTensorStridedBatched<const __r__nv_bfloat16>, cublasGemmTensorStridedB
3	6.0%	Direct Copy (float)	PyTorch ATen	void at::native::unrolled_elementwise_kernel<&::[lambda() (instance 3)]::operator ()

排名	时间占比	易读内核名	库	完整函数签名
				[lambda(float) (instance 1)], std::array TrivialOffsetCalculator<(int)1, unsigned at::native::memory::LoadWithCast<(int)1> T4, T5, T6, T7)
4	4.2%	Elementwise Multiply (BF16, vec)	PyTorch ATen	void at::native::vectorized_elementwise_ c10::BFloat16, c10::BFloat16, at::native (unsigned long)3>>(int, T2, T3)
5	3.9%	SiLU Activation (BF16, vec)	PyTorch ATen	void at::native::vectorized_elementwise_ at::native::<unnamed>::silu_kernel(at::T () const::[lambda() (instance 6)]::opera std::array<char *, (unsigned long)2>>(in
6	2.9%	Elementwise Multiply (BF16)	PyTorch ATen	void at::native::elementwise_kernel<(int at::native::gpu_kernel_impl_nocast<at::n c10::BFloat16, at::native::binary_intern &)::[lambda(int) (instance 1)]>(int, T3)
7	2.8%	Cat Batched Copy (vec, 128-tile)	PyTorch ATen	void at::native::<unnamed>::CatArrayBatchedCo int)2>, unsigned int, (int)3, (int)128, at::native::<unnamed>::CatArrInputTensor at::native::<unnamed>::TensorSizeStride<
8	2.7%	Copy/Cast (BF16, vec)	PyTorch ATen	void at::native::vectorized_elementwise_ at::native::bfloat16_copy_kernel_cuda(at std::array<char *, (unsigned long)2>>(in
9	2.4%	Flash Forward Split-KV (BF16)	FlashAttention	void flash::flash_fwd_splitkv_kernel<Fla (bool)0, (bool)0, cutlass::bfloat16_t, F cutlass::bfloat16_t>>, (bool)0, (bool)0, (bool)0>(flash::Flash_fwd_params)
10	2.0%	Elementwise Add (BF16, vec)	PyTorch ATen	void at::native::vectorized_elementwise_ std::array<char *, (unsigned long)3>>(in
11	1.8%	Cat Batched Copy (vec, 64-tile)	PyTorch ATen	void at::native::<unnamed>::CatArrayBatc unsigned int, (int)4, (int)64, (int)64>(T T2, T4, T5>, at::native::<unnamed>::Tens
12	1.8%	Elementwise Multiply (float)	PyTorch ATen	void at::native::elementwise_kernel<(int at::native::gpu_kernel_impl_nocast<at::n at::native::binary_internal::MulFunctor< [lambda(int) (instance 1)]>(int, T3)
13	1.6%		PyTorch ATen	void at::native::reduce_kernel<(int)512, at::native::MeanOps<float, float, float,

