在没有 C2C 遥测的情况下测量 NVIDIA GB10 上的 GPU 内存流量

Deepak Panigrahy*、Sirisha Kota*、Aakash Tyagi、David Kebo 和 Rabi Mahapat ra

arXiv:2611.29915v1 [cs.AR] 23 Nov 2026

摘要——**NVIDIA GB10 应用于 DGX Spark 以及多家厂商的桌面 AI 系统，它通过 NVLink C2C 将 Arm CPU芯粒与 Blackwell GPU芯粒连接起来。GPU 没有本地内存，因此每次未命中其 L2缓存的 GPU 访问都会通过 C2C 到达连接至 CPU 裸片的 LPDDR5X。然而，在我们测试的系统中，软件没有提供用于统计这类流量的计数器。我们展示了这样一种方法：通过 NVIDIA的CUPTI采样接口，由一个不附加到工作负载的进程每 10 毫秒对 GPU 的五个 L2 扇区计数器进行采样，即可实时测量 GPU 内存流量，并将复制引擎流量与 GPU内核的读取未命中和写入流量区分开来。对于大小已知的大规模流式传输，这些计数器在 Acer GN100 上的结果相差不超过 0.28%，在 DGX Spark 上相差不超过 0.005%；无论传输量如何拆分为多次复制，它们都能准确追踪传输总量，并排除在 L2 中重复使用的数据。GPU 的 SMMU计数器看似是自然的替代方案，但它统计的是地址转换，而非字节数。将此方法应用于未作修改的服务器后，我们发现，Mistral 7B 解码每个 token 会读取 13.47 吉比字节的数据，即所用权重及其键值（KV）缓存的数据量，达到 NVIDIA 标称内存带宽的 83%。我们将此方法作为开源工具发布。**

索引词—**性能计数器、统一内存、NVLink C2C、L2缓存、GPU 内存流量、NVIDIA GB10**。

I. 引言

GB10 上的统一内存。 NVIDIA GB10 用于 DGX Spark，以及 Acer、ASUS、Dell、GIGABYTE、HP、Lenovo 和 MSI 等多家厂商推出的桌面AI系统。它通过 NVLink芯片间互连（C2C）将 Arm CPU芯粒与 Blackwell GPU芯粒连接起来，二者均使用连接到 CPU裸片的 128GB LPDDR5X [8]。GPU 没有自己的内存，因此，GPU 的访问若到达内存而不是由 GPU 的 L2 缓存提供服务，就会通过 C2C （图 1 ）。工作负载移动的字节数决定了它是否受带宽限制、移动数据需要消耗多少能量，以及它与 CPU 争用同一内存的程度。在独立GPU上，这些问题可以清晰分开：设备内存流量留在板卡上，只有显式传输才会经过 PCIe。而在 GB10 上，只有一块内存和一条通往该内存的路径，因此每个未命中 GPU 的 L2缓存的内核加载也都是 C2C 流量。

互连遥测。我们测试的 GB10 系统没有为此流量提供任何计数器。Grace服务器具有用于统计跨越 C2C [9]，的字节数的 Linux 计数器，而数据中心 GPU 则通过 DataCenter GPU 管理器（DCGM）报告互连流量，但 GB10 上两者都不可用：在我们测试的两套系统上，DCGM 均无法加载（表 I）。

现有工具为何不足。 现有工具回答的是相邻的问题。Nsight Compute 会报告单个内核的内存流量，但它会对每个内核重放数次以收集其计数器 [2]。重放会将程序串行化，大幅拖慢程序，而且会单独运行每个内核，而不是让它与通常会和它并行执行的工作同时运行，因此无法观察处于正常状态的运行中服务器。Nsight Compute 也看不到驱动程序在没有内核参与的情况下执行的复制或内存设置。Nsight Systems 会记录内核和复制的运行时间，以及每次显式复制的大小——该大小由程序在调用中声明。相比之下，内核不会声明大小：它运行时读取和写入多少字节（图 1 中的紫色和绿色路径），只有硬件计数器才能得知；而在大语言模型推理中，这些内核读取和写入几乎构成了全部流量，因此对于正在运行的模型，Nsight Systems 能显示内核何时运行，却几乎看不到它们传输了多少字节。在 GB10 上，其采样的 GPU指标会报告时钟频率，以及流式多处理器 (SM)、张量核心和复制引擎的忙碌程度，但不报告内存或链路流量。`nvidia-smi`报告的内存利用率是内存处于忙碌状态的时间占比，而不是流量大小。缺少的是一个在正常执行期间采样、涵盖整个 GPU 且无需内核重放的字节计数。

方法与贡献。我们发现，可以从 GPU 端缩小这一差距。GPU 发往内存的每个请求都会经过 GPU 的 L2 缓存，而 L2 会按发出请求的单元统计其处理的 32B 扇区。本文有三项贡献。第一，我们确定了五个 L2 计数器，可用于统计 GPU 内存流量，并将其拆分为复制以及内核读写；我们还表明，可以从独立进程实时采样这些计数器。第二，我们在两台 GB10 系统上用已知流量对这些计数器进行验证，并说明为什么

GPU 的 SMMU计数器（一个自然的首选方案）无法测量字节数。第三，我们展示这些计数器如何揭示真实工作负载的情况：在不重新构建或插桩的情况下，观测大语言模型服务器的预填充和解码过程。全文将测量量称为在 *L2* 缓存处观测到的 *GPU* 内存流量；在 GB10 的拓扑结构下，它可用于估算通过 C2C 传输的有效载荷，而非线路上的字节数（第 V 节）。

II. 计数器可用性与选择

直接遥测。 表 I 列出了我们在两套系统上检查过的接口。DCGM 性能分析无法加载，英伟达管理库 (NVML) 列出了 C2C 链路，但无法查询其流量；Nsight Compute 的 C2C 部分没有收集到任何指标，

*D. Panigrahy 和 S. Kota 对本研究的贡献相同。D. Panigrahy、A. Tyagi、D. Kebo 和 R. Mahapatra 任职于得克萨斯农工大学计算机科学与工程系，美国得克萨斯州大学城，邮编 77843（电子邮箱：deepakp@tamu.edu；tyagi@cse.tamu.edu；davidkebo@tamu.edu；rabi@tamu.edu）。S. Kota 任职于喜马拉雅基金会研究部，美国阿肯色州本顿维尔，邮编 72712（电子邮箱：skota@thehimalayanfoundation.in）。

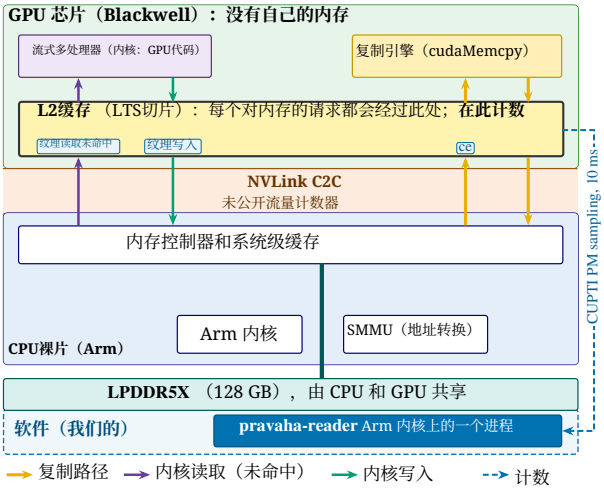


图 1. GB10：未命中 L2 缓存的 GPU 访问会经过 NVLink C2C，但没有针对此路径公开计数器。每个请求都会经过 GPU 的 L2 缓存，其计数器（蓝色标签）会统计内核读取未命中的扇区数、内核写入量以及复制引擎流量。我们的软件在 Arm 内核上作为独立进程运行（虚线区域），每 10 毫秒对这些计数器采样一次；其上方的所有部分均为硬件。芯片布局改编自 [8]。

表 I 两个系统中检查的 GPU 内存流量接口。

接口	结果
DCGM 性能分析	加载失败
NVML	已列出 C2C 链路；无流量查询
Nsight Compute C2C, NVLink	无指标；0 条链路
Nsight Systems GPU 指标	仅活动百分比；无流量
Linux 英伟达_cspmu, APMT	没有 C2C 计数器单位；表格缺失
GPU SMMU PMCG（perf）	统计地址转换次数，而非字节数
L2 计数器、CUPTI 性能监控采样	**扇区、活动（已使用）**

而 Linux 驱动程序找不到 C2C 计数器，因为固件缺少用于定位这些计数器的 Arm 性能监控表 (APMT)。在 Grace 上，同一驱动程序会将 C2C 和内存计数器作为`perf` 单元公开，用于统计每个链路方向上的字节数 [9]；在 GB10 上，`perflist`则未显示任何此类单元。来自两家厂商的这两套系统均运行 NVIDIA 驱动程序 580.159.03，得出的结果相同。因此，我们的结论是，受测软件未公开 C2C 流量计数器，而不是芯片本身不具备此类计数器。

SMMU 计数器。 系统内存管理单元（SMMU）会将 GPU 使用的地址转换为物理内存地址，并通过性能监控计数组（PMCG）报告其活动情况；Linux 将其公开为 `perf` [1]。GB10 有 17 个 PMCG；其中 7 个属于 GPU 的 SMMU，其余属于芯片上其他设备的 SMMU。Arm SMMU 由转换缓冲单元（TBU）和一个转换控制单元（TCU）组成：TBU 位于设备旁，用于缓存最近的地址转换；TCU 则处理 TBU 未命中的地址转换 [1]。响应复制操作的 PMCG 是 GPU 的 TCU，其 `transaction` 事件等于`pcie_ats_trans_rq`：它会统计地址转换请求，而且这些请求仅针对主机页发出。每个请求对应的字节数并不是复制操作本身的属性：固定内存为 2,055，注册内存为 3,690

表 II GPU 内存流量的候选计数器及决定每个计数器的测试。

候选者	结论	确定测试
SMMU PMCG 事务	被拒绝	统计地址转换次数；每次 2,055 至 3,690 B，具体取决于缓冲区
L2 缓存读取，无查找_未命中	已拒绝	包含命中；重新读取 16 mebibyte 缓冲区再次计数
L2地址空间变体	已拒绝	被 PM 采样拒绝
Nsight Compute 动态随机存取存储器、系统内存复制	已拒绝	仅支持内核重放；无法捕获驱动程序
五个 L2 计数器（第 III 节）	**kept**	已知容量，误差在 0.28% 以内，适用于两个系统

缓冲区， 10× 使用 2MiB 页大小时请求数更少，而当一个缓冲区被重复使用时，最多可达 2,969。GPU 到 GPU 的复制会跨越 C2C 两次，因此不会产生任何请求。使用一次缓冲区分配获得的字节校准值无法迁移到另一次分配，因为计数取决于页大小以及地址转换缓存的状态。该计数器仍可用于研究地址转换，但不能用于研究流量。

L2 计数器。 GPU 的 L2 缓存由英伟达称为 LTS 的多个切片构成，拥有数百个硬件计数器，而英伟达为这些计数器提供了三种软件接口。Nsight Compute 通过重放按内核读取这些计数器，并记录了它们的名称 [2]；Nsight Systems 会随时间对固定的一组计数器进行采样，但在 GB10 上不包括内存流量；CUDA 性能分析工具接口（CUPTI）则提供性能监控（PM）采样，在程序正常运行时以固定间隔读取选定的计数器子集，无需重放 [3]。我们采用第三种方式。由于 GPU 没有自己的内存，它移动的字节数就是其 L2 缓存读取未命中时提取的数据，以及它向内存方向发送的写入数据。因此，我们需要能够统计扇区而非请求的 L2 计数器；只统计未命中的读取，因为命中不会离开 GPU；统计写入；将复制引擎与内核区分开；并且能够由 PM 采样在一次运行中接受。不同于重放——重放会在内核的多次重复运行中收集不同的计数器——PM 采样只配置一次计数器并持续读取，因此它只接受硬件能够在单次运行中收集的计数器；英伟达用后缀 `_realtime` 标记 L2 计数器的此类变体，而 CUPTI 在配置采样器时会拒绝其他变体。表 II 列出了我们测试过的候选项，以及保留或拒绝每一项的原因。两个看似正确的选择实际上是错误的（表 II，第 2 行和第 3 行）。所有 L2 读取的计数器 lts__t_sectors_op_read 包含命中读取，而这些读取由 L2 缓存提供且永远不会到达内存，因此只要数据被重复使用，它就会高估流量；lts__t_sectors_op_read_lookup_miss 只统计离开 GPU 的读取。地址空间计数器（例如 lts__t_sectors_aperture_device 和 lts__t_sectors_aperture_sysmem）会将流量拆分为 GPU 缓冲区和 CPU 缓冲区；Nsight Compute 可以通过重放读取这些计数器，但 PM 采样会拒绝它们，因此无法进行实时采样。

III. 测量方法

五个计数器。三个计数器按来源划分流量，另外两个统计所有来源的总量，因此总量已经包含前三项，不应再与它们相加

。复制通过 lts__t_sectors_srcunit_ce 计数。内核读取通过 lts__t_sectors_srcunit_tex_op_read_lookup_miss_realtime 计数，内核写入则通过 lts__t_sectors_srcunit_tex_op_write_realtime 计数。总量是 lts__t_sectors_op_read_lookup_miss_realtime 与 lts__t_sectors_op_write_realtime 之和。每项都是对所有 L2 切片求和。

换算。在某个时间间隔内，计数器 x 统计的扇区数为 S_x 时，

$$B_{\text{kread}} = 32\,S_{\text{tex,rd miss}}, \qquad B_{\text{kwrite}} = 32\,S_{\text{tex,wr}}, \tag{1}$$

$$B_{\text{copy}} = 32\,S_{\text{ce}}/2, \qquad B_{\text{L2}} = 32\,(S_{\text{rd miss}} + S_{\text{wr}}). \tag{2}$$

B_{L2} 是 L2 缓存观测到的 GPU 内存流量总量，也是我们对经由 C2C 传输的有效载荷的估算。复制引擎位于 GPU 芯片上，而复制的源和目的地都位于 LPDDR5X 中，因此复制引擎会通过 C2C 读取每个字节，再通过 C2C 将其写回。复制计数器会统计这两次传输，因此 B_{copy} 要除以二。从主机向 GPU 内存复制 2 吉比字节，复制计数器因此会报告 2 吉比字节，而内存流量约为 4 吉比字节，这才是该复制操作实际产生的流量。总量还包括复制引擎和流式多处理器以外的单元产生的流量，例如扫描输出桌面画面的显示引擎。

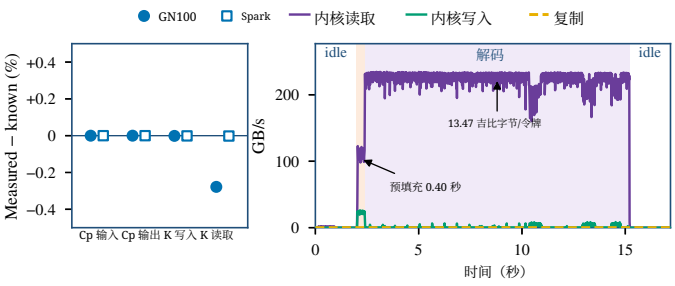
从外部采样。一个小型读取进程只需在设备上配置一次 CUPTI 性能监控 PM 采样，随后便会每 10 毫秒传输一个采样结果，其中包含对应的时间间隔和五个数值。它不运行任何 GPU 任务，也不附加到被测程序，因此无需重新构建或插桩，就能观测程序。它每秒记录一次 GPU 采样时钟和主机时钟；在 GB10 上，两者的偏差在 528 纳秒以内（跨度为 648 纳秒），因此采样结果可以与主机侧功率和能量日志对应起来。当 GPU 大部分时间处于空闲状态时，采样持续时间会超过请求值（中位数为 22 毫秒）；每个采样都涵盖其自身的时间间隔，因此总量不受影响。要统计两个主机时间戳之间某个阶段的总量，若采样跨越了边界，就按其在边界两侧各自持续的时间比例拆分。此方法假定该采样内的流量均匀，因此会一并报告各项总量中来自拆分采样的比例；对于采样数量较少的短阶段，这一比例可以反映结果在多大程度上依赖该假设。这些计数器是设备范围的：会涵盖 GPU 上的每个进程。为去除桌面等产生的后台流量，我们会在执行任务前测量空闲 GPU 的流量速率，并按任务持续时间扣除相应流量，方法与能量测量中扣除空闲功率 [6]相同：

$$B_{\text{net}} = B_{\text{run}} - (B_{\text{idle}}/t_{\text{idle}})\,t_{\text{run}}. \tag{3}$$

这假设后台活动保持在空闲速率，因此 B_{net} 是估算值，并且始终保留原始数值。

系统。我们使用了一台 Acer Veriton GN100（Linux 6.17.0- 1021-nvidia）和一台 NVIDIA DGX Spark（Linux 6.17.0- 1026-nvidia），两者均运行驱动程序 580.159.03、CUDA 13.0 和 Nsight Compute2025.3.1.0。GN100专用于测量，且桌面环境处于运行状态；DGX Spark 则是一台共享设备，没有桌面环境。

验证设计。测试程序使用 CUDA 自带的操作移动精确的数据量：`cudaMemcpy` 从主机到



(a) 已知体积 **(b) 实时观测的一次大语言模型请求**

图 2。（a）两个系统上测量值与已知数据量之间的误差：复制到 GPU 内存（Cp in）和从 GPU 内存复制出（Cp out），各 2 吉比字节；内核写入（K wr）和读取（K rd），各 4吉比字节。（b）向 vLLM 服务器发送一个请求（Mistral 7B，1,510 个提示令牌，生成 200 个令牌）：预填充在 0.40 秒内读取权重约 2.9 次；解码每个令牌读取 13.47 GiB，而其写入和复制量均接近于零。

GPU 再返回（复制引擎）、`cudaMemset`（内核写入），以及一次 GPU 到 GPU 的复制，后者由驱动程序通过一个读取并写入的内核执行。对于每种操作，读取器测量工作运行的时间窗口，减去背景值，并与已知数据量进行比较；每项检查运行三次，并报告中间值。

IV. 结果

已知容量。在 GN100 上，测得容量与已知容量的差异在 0.28% 以内；在 DGX Spark 上则在 0.005% 以内，这也是我们比值的舍入极限（图 2a）。DGX Spark 未运行桌面环境，因此检查期间，除了测试程序外没有其他程序移动内存；GN100 的误差较大则源于其桌面环境，读取检查说明了这一点。在 GN100 上，双向复制和内核写入测得的容量分别为已知容量的 1.0000、1.0000 和 1.0000；在 DGX Spark 上也分别为 1.0000、1.0000 和 1.0000；内核读取测得的结果为 0.9972 和 1.0000。由于桌面环境的屏幕更新会读取内存，读取检查对 GN100 的要求最高，其 0.28% 的误差也是所有检查中最大的；如果不进行后台流量扣除，误差会更大，而剩余的小幅负误差表明，桌面环境突发性的后台流量只能得到近似扣除。还有三个特性支持将这些计数解读为流量。第一，它们反映的是移动的数据量，而不是复制次数：以 8×256、4×512 或16×128 MiB 为单位复制 2 GiB，可得到 2.0141 至 2.0158 GiB，超出的部分是程序产生的 15 MiB CUDA 启动复制。第二，它们排除了重复利用：重新读取一个适合放入 L2 缓存的 16 MiB 缓冲区时，读取未命中仅占所读取字节数的 0.0012。第三，总量遵循数据路径：在 GN100 上，一次完整运行将 4 GiB 从一个 GPU 缓冲区复制到另一个 GPU 缓冲区，会产生 8.8 GiB 的内存流量，其中复制操作读取 4 GiB、写入 4 GiB，程序清空其缓冲区时写入 0.5 GiB，其余流量来自桌面环境和 CUDA 启动复制；在独立 GPU上，这些流量都不会经过主机链路。从外部观察 PyTorch 发出的复制操作时，复制 4 GiB 测得的结果为 3.9999 GiB。

使用 Nsight Compute 获取的 L2 缓存计数还有两个特性也会影响其用途。它们统计的是不同的 32B 扇区，而不是请求的字节数：在所有测试的访问模式中，该计数与实际触及的扇区数之差都在 0.19% 以内。对可分页

表 III 解码流量／每个 token：根据模型规模推算的预期值，以及从服务器外部测得的值。

Case	预期	实测值
Qwen3 4B，短提示	全部权重：7.49	7.52
Mistral 7B，短提示	除未使用的行之外的所有权重 嵌入表中的行：13.25	13.26
Mistral 7B，1,510 提示令牌	上方行加上以下 KV 缓存 1,510 个令牌，每个 128 KiB：13.45	13.47

所有数值均为每生成一个令牌对应的吉比字节数。预期值未计入激活值和小型张量。

内存的写入产生的写入扇区数比预期多 15.2%；在两个系统上，每次处理 256 MiB 时还多出约 110 MB 的读取扇区。我们尚未确定这是额外流量还是计数效应，因此只能比较使用相同分配类型的工作负载之间的流量。单独的读取器不会对程序造成可测量的减速：在有无读取器的情况下分别计时 20 次，在两个系统上，一个传输 8 GiB 的 GPU 作业其中位耗时变化最多为 0.53%，处于重复运行之间的波动范围内（至少 2%）。发布的工具在两个系统上均通过了全部 19 项测试，涵盖安装、采样、已知容量、阶段、大语言模型观测和干净移除；每项测试的命令和结果都列在制品的测试矩阵中 [5]。

从外部观测到的大语言模型服务器。图 2b 展示了向运行 Mistral 7B（权重占 13.50 吉比字节）的 vLLM 服务器发送的一个请求，测量过程未改变服务器。在解码期间，GPU 每个令牌读取 13.47 吉比字节，以 226 GB/s 的速率执行读写，并生成 15.5 个令牌/秒：每个令牌会读取除嵌入表中未使用行之外的全部权重，以及此前 1,510 个令牌的 KV 缓存，每个令牌占用 128 KiB（表 III）。预填充在 0.40 秒内读取 38.2 吉比字节；对于包含 1,510 个令牌的提示词，这相当于对权重执行约 2.9 次遍历，与分块处理提示词的情况一致，同时写入 7.9 吉比字节。整个过程中复制量均为零：加载完成后，推理仅通过内核传输数据。解码速率约为英伟达为 GB10 内存指定的 273 GB/s 的 83%， [4],这与在该系统上解码受内存带宽限制的情况一致。表 III 还加入了第二个模型 Qwen3 4B，并将每项测量值与模型规模的预测值进行对比。预期值未计入激活值和小型张量，因此我们将这种一致性视为解码会流式读取权重和 KV 缓存的证据，而非精确测试。预填充的表现有所不同：其读写总速率为 125GB/s，约为解码速率的一半，但它会多次读取权重，因为提示词的每个分块都会在多个令牌之间复用权重，并将更多时间用于计算。其 7.9 吉比字节的写入主要是提示词的 KV 缓存和激活值；解码的写入量很小。在请求之间，服务器并非完全无活动；其空闲流量是在请求前测得的，并通过基线予以扣除。

V. 讨论、局限性与结论

研究发现概要。在 GB10 上，GPU 内存流量只能从 GPU 侧的 L2 缓存观察到。若没有这些计数器，就无法实时测量内核搬运的字节数，而这些字节几乎构成了大语言模型推理的全部流量；有了这些计数器，未经修改的服务器显示，解码会在每个 token 中各读取一次权重和 KV 缓存，速度接近内存标称上限。关于 Grace Hopper [7] 和 MI300A [10] 统一 CPU 与 GPU 内存的研究依赖于 GB10 未公开的计数器，而 GB10 的研究则使用 Nsight Compute [11]按内核测量流量。CUPTI PM 采样的设计初衷是在被分析的程序内部运行 [3],，SMMU 计数器则用于监控地址转换 [1]；我们的贡献是确定哪些计数器能够追踪流量，并从独立进程对其进行采样。

局限性。这些计数器测量的是 L2 缓存处观察到的流量，而不是 C2C 链路上的字节数；后者还包括协议开销和地址转换流量。将这些计数读作 C2C 有效载荷，取决于英伟达对内存控制器的布置 [8],，这一布置与我们已知的流量核查结果一致。L2 缓存是写回缓存，因此，数据在被逐出前若被反复写入，计数可能会高于其实际到达内存的次数；我们的核查并未测试这种情况。这些计数覆盖整个设备，不包括 CPU 内存流量，且一次只允许一个读取者；此外，我们仅在使用一个驱动程序的两套系统上进行了验证。

结论。GB10 未提供 C2C 流量计数器，但从独立进程采样的五个 L2 计数器可以实时测量其 GPU 内存流量，并区分复制、内核读取和写入。将这些数据与基于同一主机时钟的功耗采样相结合，便可测量 GPU 内存流量的每字节能耗 [6], ，并研究推理如何拖慢共享内存中的 CPU 工作。该工具和数据已作为开源项目发布 [5]。

参考文献

[1] Arm Limited。《Arm 系统内存管理单元架构规范，SMMU 架构第 3 版》（IHI 0070）。https://deve loper.arm.com/documentation/ihi0070/latest。访问日期：2026-11-04。[2] 英伟达。《Nsight Compute：内核分析指南》。https://docs.nvidia. com/nsight-compute/ProfilingGuide/index.html。访问日期：2026-11-04。[3] 英伟达。CUPTI：PM 采样。CUDA Toolkit 文档，CUPTI 用户指南，2025。[4] 英伟达。DGX Spark 用户指南：硬件概览，2025。[5] Deepak Panigrahy 和 Sirisha Kota。pravaha-mem：通过 NVLink C2C 在 NVIDIA GB10 上实时获取 GPU 内存流量，2026。[6] Deepak Panigrahy、Aakash Tyagi、Stephen Abkin、David Kebo Houngni- nou 和 Rabi Mahapatra。每个成功目标的能耗：智能体 AI 系统的目标级能耗核算。arXiv:2605.22883，2026。[7] Gabin Schieffer、Jacob Wahlgren、Jie Ren、Jennifer Faj 和 Ivy Peng。利用集成式 CPU-GPU 系统内存实现高性能计算：初探 Grace Hopper。载于 第 *53* 届国际并行处理会议（*ICPP*）论文集，第 199–209 页。ACM，2024。[8] Andi Skende。NVIDIA GB10 SoC：桌面上的 AI 超级计算机。载于 *2025 IEEE Hot Chips* 第 *37* 届研讨会（*HCS*），美国加利福尼亚州斯坦福，2025。幻灯片：https://www.hc2025.hotchips.org/assets/program/confere nce/day2/21_nvidia_skende_final.pdf。[9] Linux 内核文档。NVIDIA Tegra SoC 非核心性能监控单元（PMU）。https://www.kernel.org/doc/html/latest/admin-guide/perf/nvidia-pmu.html。访问日期：2026-11-04。[10] Jacob Wahlgren、Gabin Schieffer、Ruimin Shi、Edgar A. León、Roger Pearce、Maya Gokhale 和 Ivy Peng。剖析 AMD MI300A 加速处理器上的 CPU-GPU 统一物理内存。arXiv:2508.12743，2025。[11] Yifan Zhu、Yekai Pan 和 Chen Ding。锯齿波前重排序：在 NVIDIA GB10 16032 上增强 CuTile FlashAttention，2026 2601。arXiv：。。