使用现成大语言模型和编译器产物估算 GPU Roofline 算术强度

Gregory Bolet [1] **Giorgis Georgakoudis** [2] **Konstantinos Parasyris** [2] **Harshitha Menon** [3] **Niranjan Hasabnis** [4] **Kirk W. Cameron** [1] **Gal Oren** [5 6]

arXiv:2611.32975v1 [cs.DC] 24 Nov 2026

摘要

GPU 开发者使用性能分析来了解计算和数据移动，但访问目标硬件可能会延误优化。我们研究现成大语言模型（LLM）能否在不执行目标 GPU 工作负载的情况下估算 Roofline 算术强度（即浮点运算次数（FLOPs）与设备内存字节数之比）。我们针对 179 种组合进行评估，涵盖五个内核族、四款 NVIDIA 和 AMD GPU、四种并行运行时（CUDA、HIP、OpenMP 和 OpenCL）以及经过修改的代码变体，并比较仅提供源代码的查询、附带编译器产物的查询，以及沙箱化智能体分析。编译器产物使算术强度估计值落在剖面化数值 1.20 倍范围内的比例从 8–14% 提升至 26–32%，相应的带宽受限或计算受限分类在 78–92% 的运行中正确。使用编译器产物时，浮点运算次数估计在 59–88% 的运行中满足 $1.20\times$ 容差，而设备内存流量的这一比例为 36–40%。对于 gpt-5.6-sol，智能体分析使算术强度估计值落在 $1.20\times$ 容差范围内的比例提高了 6.5 个百分点， $3.6\times$成本中位数和 $2.5\times$延迟中位数与编译器辅助查询相当；其他模型则成本更高，且未显示出统计上明确的改进。这些发现表明，在 GPU 访问受限时，LLM 估计有助于确定优化优先级，同时也指出依赖缓存的设备内存流量是精准预测算术强度的主要障碍。

1 引言

开发者应该首先优化什么：GPU 内核的计算，还是其数据移动？在目标图形处理器（GPU）上对内核进行性能分析有助于作出这一决策，但由于共享集群分配或优化候选项队列的影响，访问可能会延迟。大语言模型（LLM）优化系统也会生成需要进行硬件评估的候选项（Zhang etal.，2025；Nichols et al.，2025）。为避免对设备显存流量测量造成干扰，在单独进行性能分析时，这些候选项必须轮流使用目标 GPU

(NVIDIA，2026年）。大语言模型助手日益融入软件开发工作流（Nichols 等人，2024a），因此静态估算有望成为开发者已在使用的工具的一项额外用途

[1]弗吉尼亚理工大学，布莱克斯堡，弗吉尼亚州，美国 [2]英特尔公司，美国 [3]劳伦斯利弗莫尔国家实验室，利弗莫尔，加利福尼亚州，美国 [4]Codemetal，波士顿，马萨诸塞州，美国 [5]斯坦福大学，斯坦福，加利福尼亚州，美国 [6]以色列理工学院，海法，以色列。通信联系人：Gregory Bolet <gbolet@vt.edu>、Giorgis Georgakoudis <giorgis.georgakoudis@intel.com>、Konstantinos Parasyris <konstantinos.parasyris@intel.com>、Harshitha Menon <gopalakrishn1@llnl.gov>、Niranjan Hasabnis <niran-jan@codemetal.ai>、KirkW. Cameron <cameron@vt.edu>、Gal Oren <galoren@stanford.edu>。

预印本。

开发者已经会向这些工具询问相关问题。图1 展示了在图形处理器访问受限时，估算结果如何帮助确定优化候选项的优先级。

Roofline 算术强度（RAI）是浮点运算次数（FLOPs）与传输的设备内存字节数之比（Williams 等人，2009）。将其与目标 GPU 的平衡点进行比较，可以确定 Roofline 模型所处的带宽受限区间或计算受限区间。数值本身还有助于界定可达到的吞吐量，并显示与平衡点的接近程度；仅用区间标签无法提供这些信息。预测 RAI 需要考虑编译器转换对实际执行的算术运算的影响，以及访问模式、缓存复用和测量显存流量所依据的边界。这些估算有助于确定优化优先级；执行测量则可验证性能提升。

分析型和静态 GPU 方法根据程序和架构信息估算性能（Hong &Kim，2009；Emondset al.，2023）。先前的研究还探索了基于大语言模型的 Roofline 分类（Bolet et al.，2025a）、浮点运算次数预测（Boletet al.，2025b）以及经训练的数值型 GPU 性能预测（Nguyen-Nhat et al.，2024；Wang et al.，2025）。除了工作区间分类之外，我们还评估现成大语言模型恢复数值

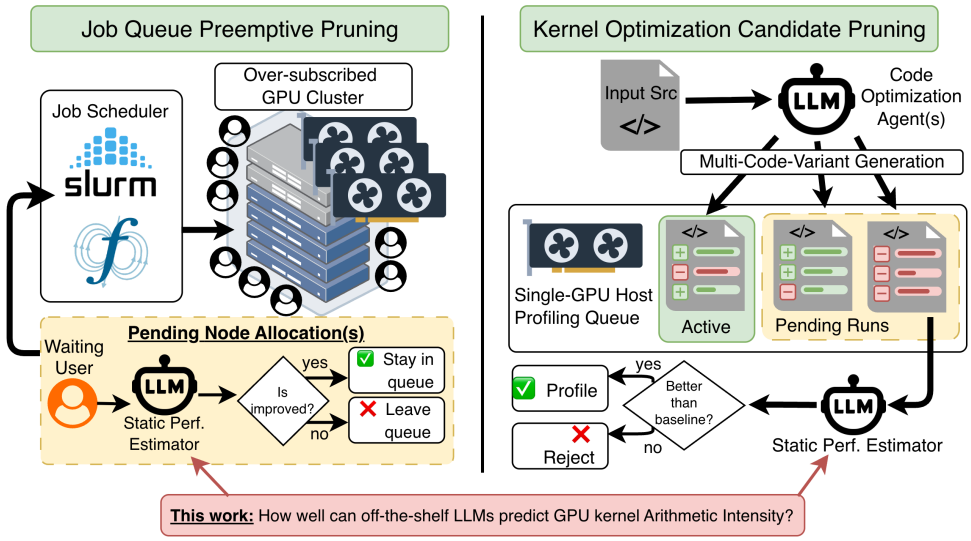


图 1：GPU 访问受限时静态估算的潜在用途：筛选等待集群分配的作业（左），以及确定生成的优化候选项的性能分析优先级（右）。我们评估其底层算术强度预测。

RAI 及其浮点运算次数和设备内存流量组成部分的准确性，这些评估覆盖不同目标平台；同时考察编译器证据和沙盒分析在无需特定任务训练的情况下能带来哪些帮助。

我们比较了三种无需执行目标 GPU 工作负载的策略：仅使用源代码进行零样本查询、辅以编译后指令集架构（ISA）产物的零样本查询，以及沙箱化智能体分析。本研究评估了三种模型（`deepseek-v4-flash`、`gemini-3.7-flash`和`gpt-5.6-sol`），涵盖内核、变体、GPU 和运行时的 179 种组合：五个`HeCBench`内核系列（Jin & Vetter，2023）、四款 NVIDIA/AMD GPU，以及受支持的 CUDA、HIP、OpenMP 和 OpenCL。四种变体用于探测模型对名称、编译器设置、输入和数据类型的敏感性，从而尽可能检验其是否依赖可识别的代码，或是否回忆起原始源代码中的测量结果。模型会获得执行输入和目标规格；用于评分的性能分析计数器则不会提供，失败或不完整的预测均计为未命中。本研究区分数值准确率与性能区间准确率，诊断浮点运算次数和设备内存流量估算中的误差，并测量额外分析的成本与延迟。我们围绕四个研究问题组织研究结果：

RQ1. 大语言模型能否根据静态证据预测 GPU 内核的 RAI? 在提供 ISA 时，大语言模型在 26–32% 的运行中对 RAI 的估算与参考值相差不超过 1.20×，高于仅提供源代码时的 8–14%；在 2× 时，ISA 准确率达到 48–60%。提供 ISA 时，浮点运算次数准确率在 1.20× 范围内的比例为 59–88%，而设备内存流量的这一比例为 36–40%，这是反复出现的数值短板。

RQ2. 预测的 RAI 能否正确识别 GPU 内核的 Roofline 区域? 尽管数值估计通常无法达到 1.20×目标，但源代码加上指令集架构能够正确分类 78–92% 的运行，超过了针对每个 GPU–精度组合选择更常见参考类别的预测器（73.2%，在相同案例上拟合并评分）。对于根据剖析得到的 RAI 判定、处于平衡点 2×以内的 40 个案例，指令集架构将 gpt-5.6-sol 的分类准确率从 42.5% 提升至 79.5%。

RQ3. 数值 RAI 准确率是否会因工作负载、目标 GPU 和代码变体而异? 准确率因工作负载–GPU 组合而异，这与访问模式和有效复用的差异一致；指令集架构带来的增益适用于全部四个变体。例如，对于 1.20×中的 lavaMD，在 A100/H100 上，使用指令集架构的 gpt-5.6-sol 可达到 90–100% 的准确率，但对于 particlefilter，在每个 GPU 上的准确率均为 0%。

研究问题4：沙箱分析能否在零样本 ISA 提示的基础上进一步提升数值 RAI 准确度? 沙箱分析使 gpt-5.6-sol 的准确率在 1.20× 内提高了 6.5 个百分点， 3.6× 成本中位数与源码加 ISA 查询相当；对于其他模型，增益仍不确定，成本中位数高达 6.7–17.7× 倍，完成状态最低降至 64%。

2 背景与相关工作

Roofline 背景。The Roofline 模型以峰值算力与

峰值吞吐量和峰值内存带宽乘以算术强度两者中的较小值，来界定可实现吞吐量（Williams 等人，2009）。将算术强度与平衡点进行比较，可判断模型处于带宽受限还是计算受限区间，并据此确定优化优先级。缓存感知 Roofline 建模将此分析扩展到 GPU 内存层次结构（Lopes 等人，2017）。

静态、分析和仿真方法。分析型 GPU 模型将执行行为与程序结构及体系结构资源联系起来（Hong 与 Kim，2009 年；Baghsorkhi 等，2010 年）。静态 PTX 分析可用于 CUDA 执行时间预测（Alavani 等，2018 年），而静态分析和预测分析则无需运行程序即可指导自动调优（Lim 等，2017 年）。学习型预测变量根据静态 PTX 特征（Guerreiro 等，2019 年；Tran 等，2025 年）或通过插桩执行收集的指令统计信息（Braun 等，2021 年），估算执行时间或功耗。CUDAsap 静态获取基本块执行频率和指令统计信息，作为性能分析的替代方案（Emonds 等，2023 年）。TileSight 显式对分块复用、缓存行为和内存流量进行建模（Mo 等，2026 年）。其他方法则使用指令跟踪进行分析型 GPU 核心建模，例如 GCoM（Lee 等，2022 年），或进行详细的 GPU 仿真，例如 PPT-GPU 和 Accel-Sim（Arafa 等，2019 年；Khairy 等，2020 年）。在这些方法之外，我们探究使用大语言模型，根据静态证据估算执行的算术运算量和设备内存流量。

用于 GPU 性能预测的大语言模型。 与此密切相关的研究由 (2025a;b) 评估现成大语言模型对 HeCBench CUDA 内核进行 Roofline 区间分类和 FLOP 计数预测的能力。本研究将该设定扩展至设备内存流量以及由浮点运算次数和流量得出的数值 RAI 的估算。经过训练的预测器则针对相关数值：LLMPerf 预测 OpenCL 内核执行时间，而《代码回归语言模型》涵盖 Triton 内核延迟（Nguyen-Nhat 等，2024；Akhauri 等，2026）。Omniwise 在 HIP 内核上微调模型，用于预测 AMD MI250 和 MI300X GPU 上的算术强度、带宽、缓存命中率及其他性能指标（Wang 等，2025）。GPU Forecasters 评估现成大语言模型和强化学习调优的大语言模型，这些模型可预测相对内核加速比，以便在内核搜索期间确定 GPU 测量的优先级（Khan 等，2026）。我们使用未经特定任务训练的现成模型，在 NVIDIA 和 AMD 目标平台上比较源代码证据、编译器产物和沙盒分析。我们在预测期间不执行目标 GPU 工作负载，并测量准确性、成本、延迟和完成状态。

用于并行代码和工具辅助优化的大语言模型。相关工作还开发了高性能计算代码模型（Nichols 等，2024b；Kadosh 等，2023；2024）、面向中间表示和汇编语言的编译器模型（Cummins 等，2024），以及用于并行代码和加速器内核生成的基准测试（Nichols 等，2024a；Ouyang 等，2025；Wen 等，2025）。CudaForge 和 AccelOpt 等优化系统利用执行反馈或性能分析反馈来引导大语言模型生成的修改（Zhang 等，2025；Nichols 等，2025；Zhang 等，2026）。CUDA-LLM 使用目标 GPU 的延迟反馈，而 SwizzlePerf 提供硬件规格和性能分析日志（Chen 等，2025；Tschand 等，2025）。AutoKernel 对模型进行性能分析，以识别瓶颈，并迭代测试 Triton 或 CUDA 内核优化（Jaber 和 Jaber，2026）。Kevin 通过多轮强化学习和执行反馈训练 CUDA 内核优化模型（Baronio 等，2025）。更广泛地说，PerfCodeGen 利用运行时反馈优化通用代码（Peng 等，2025），而 AlphaEvolve 将进化式代码搜索与评估器反馈相结合（Novikov 等，2025）。Mapo 训练一种内存性能预测器，该预测器在推理时使用静态特征，并将预测指标作为奖励提供给微调大语言模型优化器（Liu 等，2026）。此类静态估算可以通过在加速器访问受限时优先筛选候选项，来补充基于硬件反馈的优化。

3 方法

我们通过一系列实验，评估现成大语言模型静态预测 GPU 内核 Roofline 算术强度（RAI）的能力，实验旨在：**(1)** 探查对所选评估内核潜在记忆效应的敏感性，**(2)** 检验模型是否偏向某些 GPU 架构和并行运行时，以及**(3)** 通过及早筛选目标 GPU 内核和待查询的大语言模型，避免耗费高昂成本的大语言模型实验。这些考量使我们能够节省实验时间和经费，同时深入了解现成大语言模型在目标任务中的表现。下文将详细介绍实验步骤和设计决策。

3.1 目标内核选择

图 2概述了我们选择内核的方法。我们的目标内核取自 `HeCBench` 套件（Jin & Vetter，2023），我们从其中包含的成熟套件（例如：NAS 并行基准测试、Parboil、Rodinia、NVIDIA CUDA Samples）中，对 254 个内核进行了性能分析，涵盖三款 NVIDIA GPU：A10、A100 和 H100。 `HeCBench` 为所选程序提供 CUDA、HIP 和 OpenMP 实现；OpenCL 实现则取自 Rodinia

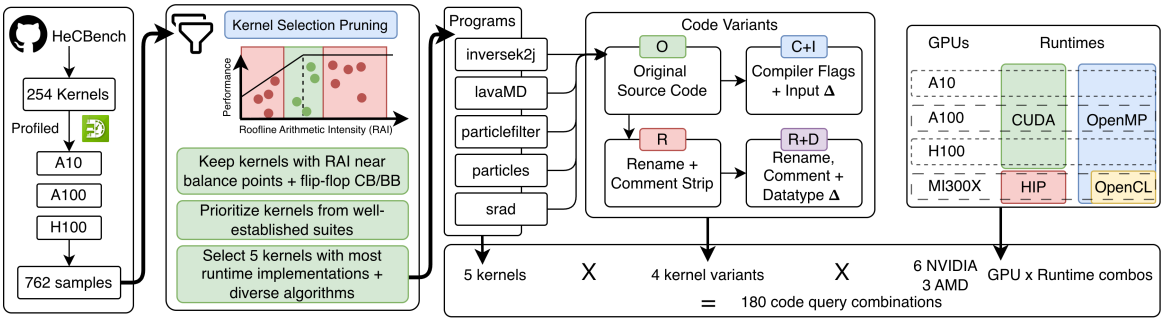


图 2：内核选择、代码变体以及用于构建预测查询的 GPU–运行时组合概览。五个内核族、四种变体和九种 GPU–运行时组合共构成 180 个名义案例；由于粒子计算内核的执行端 OpenCL 变体不可用，因此实际评估了 179 个案例。

或为本研究移植，详情如下。

我们依据与 GPU平衡点的接近程度，以及不同硬件和并行运行时下带宽受限/计算受限（BB/CB）行为的变化来指导内核选择。接近平衡点的案例会使分类对数值 RAI 误差敏感，而运行区间发生变化时则需要针对目标进行预测。选择以内核系列为单位进行，因此，单个变体–GPU–运行时案例仍可能远离其平衡点。

尽管我们最初使用 NVIDIA GPU 进行内核选择，但也在 AMD MI300X GPU 上考察了目标内核的 HIP、OpenMP 和 OpenCL 变体。鉴于 CUDA/HIP 之间存在很强的词汇相似性，且 OpenMP 卸载内核具有 GPU 无关性，这使我们能够检查预测准确度中的跨厂商偏差。遗憾的是，NVIDIA 工具链已不再支持 OpenCL 性能分析，因此我们在 MI300X 上对内核的 OpenCL 版本进行采样，作为一项额外的泛化能力研究。

每增加一个内核族，就会增加 36 个案例，相当于每个模型、策略和重复实验的查询量约增加 20%，因此 第 3.4 小节中所述的 2,000 美元实验预算促使我们将研究范围限制在五个族。符合条件的五族集合有很多，因此我们采用分阶段流程从中进行选择，详情见 附录 A.3 节。我们在全部四款 GPU 上对八个候选族进行了剖面分析；每个族都在 `HeCBench`中提供了 CUDA、HIP 和 OpenMP 卸载实现，以及可获取源代码或移植的 OpenCL 实现。对于同一源代码和输入在 NVIDIA GPU 上主导精度的 FLOP 计数存在差异的族，我们予以排除：包括 `threadfence`，其依赖争用的原子重试使 FLOP 计数相差 37%；以及仅使用整数运算的 `bfs`，其少量计入的浮点运算次数也会因 GPU 而异。随后，我们要求每个集合都包含一个达到饱和状态的计算受限锚点和一个达到饱和状态的带宽受限锚点，并根据案例中距离平衡点不超过 1.5 个倍频程（2.8×）的占比，对剩余四个有效集合进行排名——该集合排名第一（其中 31% 的案例接近平衡点），不包含任何两个工作形态和内存访问模式相同的族，并且在八种替代排名方案中的六种下仍位居第一。

3.2 所选图形处理器内核及其变体

五个选定程序及其目标内核总结于附录中的 表 A.3 。它们的上游来源涵盖 AxBench、Rodinia 和 NVIDIA CUDA Samples：`inversek2j` 来自 AxBench，`lavaMD`、`particlefilter`和`srad`来自 Rodinia，而 `particles` 来自 NVIDIA SDK 样本（Cheet al., 2009; Yazdanbakhsh et al., 2017; NVIDIA Corporation, 2018）。CUDA、HIP 和 OpenMP 目标卸载实现取自 `HeCBench`，而匹配的 OpenCL 实现则在可用时采用 Rodinia提供的实现，并为 `inversek2j`和 `particles`采用本研究创建的移植版本。它们测得的运行状态会随目标 GPU、运行时和数据类型而变化。

为测试对源代码和执行条件变化的稳健性，我们研究每个程序的四个变体：（**O**）每个运行时的**Original** 基线源代码；（**CI**）在支持的情况下，保持基线源代码不变，但更改**Compiler** 标志和/或程序**Inputs**；（**R**）删除注释，并将所有符号/函数**Renamed** 为中性名称的版本；（**RD**）在相同的去注释/**Renamed** 版本基础上更改 **Datatype**（从 FP32 →FP64 更改为 FP64 → FP32）。这些变体通过使用未修改和已修改的源代码，提供互补的比较。**O** 和**CI** 共享完全相同的源代码，而**R**和**RD** 则移除了可识别的名称和注释。**R**保留了**O**的预期计算，而 **CI**更改编译设置或输入，**RD** 则更改浮点数据类型。这些比较用于测试预测是否会对表层变化以及影响数值目标的变化做出响应，并在不证明存在训练数据暴露的情况下，尽可能检查模型是否依赖记忆化答案。

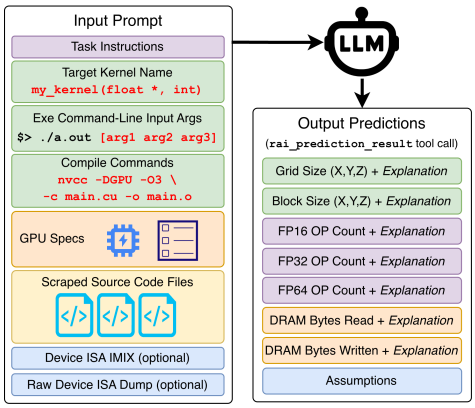


图 3：零样本提示及预期的结构化工具调用响应。仅源代码查询提供源代码，而源码加 ISA 查询还会提供汇编指令和指令混合（IMIX）证据。

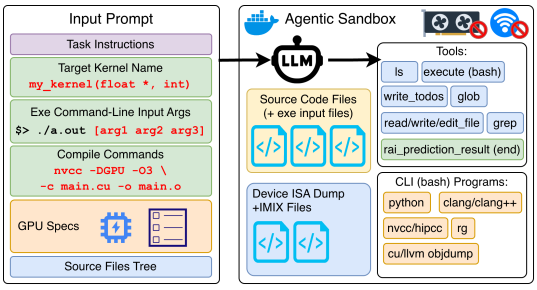


图 4：智能体沙盒设置。我们采用了与 ZS-Code 实验类似的输入提示，但将冗长的源代码和 ISA 信息移入沙盒，供智能体探索。我们向大语言模型提供了各种辅助工具，用于管理待办事项列表、读写文件，以及与环境中预安装的命令行界面程序交互。

这些内核、变体、图形处理器和运行时组合定义了 180 个名义案例，其中我们评估了 179 个。OpenCL particles`CI`案例被排除，因为如果不修改源代码，就无法更改其工作负载维度或设备编译选项，而这将违反 `CI` 约束。

3.3 大语言模型查询方法

我们的评估考察了 **(1)** 直接零样本（`ZS`）提示方法，以及 **(2)**智能体引导（`AG`）提示方法。智能体设置通常是我们预期成果将应用于瓶颈系统的场景，这类系统往往会循环执行智能体内核优化。我们有意将 ZS 与 AG 提示一并研究，以对比二者在成本与时间方面的权衡。

图 3 和 图 4 分别描述了 `ZS` 和 智能体引导 提示方法。对于两种案例，我们都使用相似的输入/指令提示，引导大语言模型得出答案。我们提供目标内核的名称、命令行参数、编译命令和图形处理器规格（即：模型名称、缓存层次结构），使大语言模型能够了解我们要求其预测的内核，以及其运行所在的系统。 `ZS`查询会在提示中直接接收固定证据，而 智能体引导运行则可以交互式地检查文件，并在沙盒中执行计算。 `ZS` 提示包含目标程序的每个源文件，其中许多文件并不影响目标内核；智能体引导 运行会在沙盒中接收相同的文件以及文件树，并自行选择要检查的内容。沙盒无法访问网络，也没有图形处理器，因此智能体可以编译代码并运行自己的分析脚本，但无法在图形处理器上运行目标工作负载；每次运行最多可获得 100 个模型响应。

我们还研究了将与目标内核对应的 SASS（NVIDIA）和 GCN（AMD）指令形式的编译后信息纳入其中（下文称为**指令集架构**）。除原始指令集架构转储外，我们还将静态指令混合（**指令混合**）信息添加到 ZS 提示中，并为智能体引导提供静态 IMIX 文件。加入指令混合信息后，大语言模型无需计算精确的静态指令集架构指令计数，可以专注于推理动态执行。我们将仅提供源代码证据的 ZS 实验称为`ZS-Code`，将提供指令集架构的实验称为 `ZS-ISA`。

大语言模型预期输出的是一个名为**RAI 预测结果** (见图 3）的工具调用，其中包含大语言模型给出的所有与 RAI 相关的预测。此工具调用包括**(1)**目标内核的启动边界（即网格/线程块大小），以确认线程数估计正确；**(2)**用于 RAI 分子的 FP16/32/64 浮点运算次数估计；以及**(3)**用于 RAI 分母的指定设备内存计数器边界上读写的字节数。为便于事后分析，每项预测都应提供说明，并列出大语言模型计算时所作的假设。这些工具调用使预期输出规范化，从而能够将真实测得的 RAI 性能计数器与大语言模型 RAI 估计值进行比较。

NVIDIA 参考计数器测量首次目标内核调度期间 L2 与 DRAM 之间的传输；MI300X 计数器测量从 TCC/L2 向 MALL 发出的请求，包括由 MALL 提供服务的请求。提示中明确规定了这些边界，因此目标是在调度期间跨越所述边界的流量，而不是 ISA 加载和存储的字节量。因此，内核可以执行存储操作，但如果在测量时间段内没有写入流量跨越该边界，记录的写入量仍可能为零。

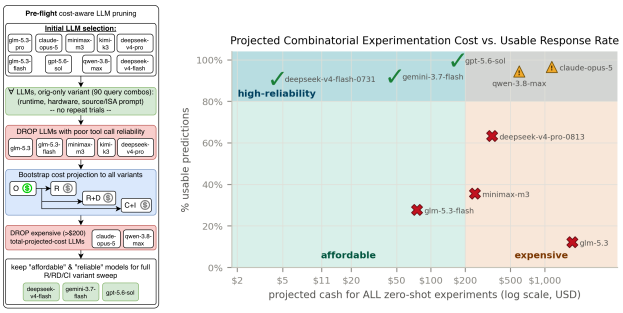
3.4 大语言模型筛选阶段

拥有 179 个已评估案例和 3 种不同的提示策略（ZS-Code、ZS-ISA、智能体引导），并希望在多个大语言模型上快速测试这些策略，很快就会变得非常昂贵，尤其是当我们希望对每个查询进行重复试验，以考虑预测结果的变化时。我们考虑了开源大语言模型和前沿大语言模型，发布日期截止至 2026年8月18日。附录 表 A.2 列出了我们最初考虑用于本研究的大语言模型。为确保可复现性，我们将实验总支出上限设为 2000 美元，并预期单个大语言模型的所有零样本提示运行总成本应控制在合理的 200 美元以内，剩余总预算则用于智能体实验。根据这一支出上限，我们筛选掉了成本高昂且不可靠的大语言模型，如图 5a所示。我们仅使用 O 代码变体，以及 ZS-Code 和 ZS-ISA提示，对每个大语言模型进行采样（共 90 个查询组合），并记录每个查询成功完成的 RAI 预测结果 工具调用次数（即可用预测数量）和金额支出。每个大语言模型均采用其制造商默认的思考（推理）设置进行查询，因为我们的实验重点是这些大语言模型的开箱即用行为。基于这些初始成本数据，我们通过自助法预测查询每个大语言模型其他变体的成本，以估算每个模型进行零样本实验的费用；其中每个查询重复 5 次，并剔除了claude-opus-5和 qwen-3.8-max ，因为预计它们将分别花费 1151 美元和 597 美元。随后，我们剔除了可用预测低于 80% 的大语言模型（glm-5.3、glm-5.3-flash、minimax-m3、deepseek-v4-pro-0813），因为它们会在未完成的查询上浪费资金。图 5b 展示了这一成本感知的大语言模型筛选阶段中预计的支出，以及达到的可用预测百分比。

经过这些筛选步骤后，我们最终保留了三个可靠且经济高效的 大语言模型（deepseek-v4-flash、gemini-3.7-flash 和 gpt-5.6-sol），并在后续实验中使用它们。在评估中，我们分别将这些模型称为 deepseek、gemini 和 gpt-5.6。

4 评估

我们在三种模型和三种策略下评估了 179 个内核–变体–图形处理器–运行时案例，每个案例重复 5 次：每种模型–策略配置运行 895 次，总计运行 8,055 次。一次运行 即为一次预测尝试，包括失败和不完整的预测。我们用总 FP16、FP32 和 FP64 浮点运算次数除以读写的设备内存字节数总量来计算 RAI。流量统计的是内核调度期间跨越目标性能分析器内存边界的字节数。参考真实硬件性



(a) (b)

图 5：（a）预先成本感知的大语言模型筛选工作流；（b）预计全扫描成本与工具调用完成率。

能计数器值是每个案例十次性能分析试验的均值；参考值RAI 根据这些计数器均值计算。对于经过性能分析的参考值 x（RAI、浮点运算次数或字节数）以及大语言模型的估计值 $\hat{x}$， $f\times$ 容差要求$x/f \le \hat{x} \le fx$，并对按比例计算的高估和低估采用对称处理。我们选择 $1.20\times$ 作为近似数值一致性的主要报告阈值，同时报告更宽松的 $2\times$容差。数值准确率是满足该容差的运行所占比例，失败或不完整的预测计为未命中。附录图 A.21 显示，编译器证据带来的提升和 gpt-5.6的智能体引导增益在 $1.10\times$和 $1.50\times$均持续存在。

我们还评估了每个数值估计所隐含的 Roofline 分类机制。平衡点 B 是图形处理器在相应精度下的计算峰值，该精度的参考浮点运算次数除以内存带宽峰值所得值最大。RAI 值低于 B 时属于带宽受限（BB）；RAI 值大于或等于 B 时属于计算受限（CB）。从每个预测的主导精度中选择B 不会改变所评估运行的分类。没有可用预测或 RAI 为非有限值的运行将被归类为未分类，并计为分类未命中。

RQ1 评估数值准确率，RQ2 评估由此得出的 Roofline 分类，RQ3 考察不同工作负载、GPU 和代码变体之间的差异，RQ4 则评估沙箱分析在准确率、成本和延迟方面的权衡。

4.1 RQ1：大语言模型能否根据静态证据预测 GPU 内核的 RAI?

借助编译器证据，大语言模型在 $1.20\times$的运行中，有 26–32% 能够预测出数值 RAI 的参考值。 图 6 比较了三个模型和策略下 RAI 及其浮点运算次数和流量组成部分的数值准确率。与三个模型在 ZS-Code 下达到的 8–14% 相比，编译器证据使数值 RAI 准确率提高了 18–20 个百分点。在 $2\times$ 容

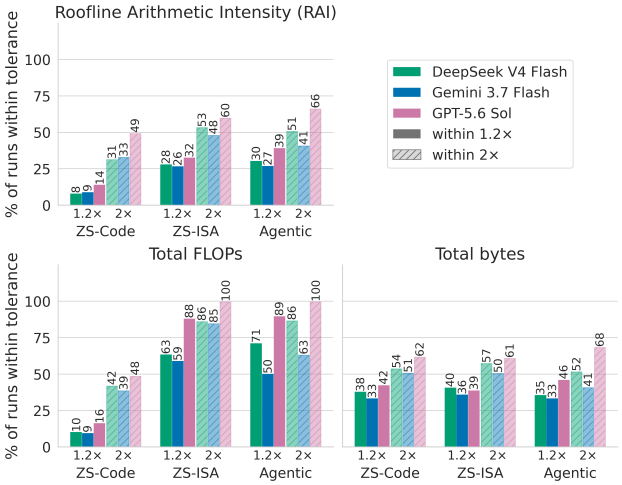


图 6：编译器证据提升了数值 RAI 预测的准确性，对 FLOP 计数的提升大于对设备内存流量的提升。各面板展示了不同模型和策略下的 RAI（左上）、总浮点运算次数（左下；FP16+FP32+FP64）以及设备内存总流量（右下；读写字节数）。实心柱表示运行结果在 1.20× 容差范围内的百分比；斜线柱表示采用 2× 容差时的结果。失败和不完整的预测均计为未命中。附录中的图 A.18 按精度细分了浮点运算次数，并将流量拆分为读和写。

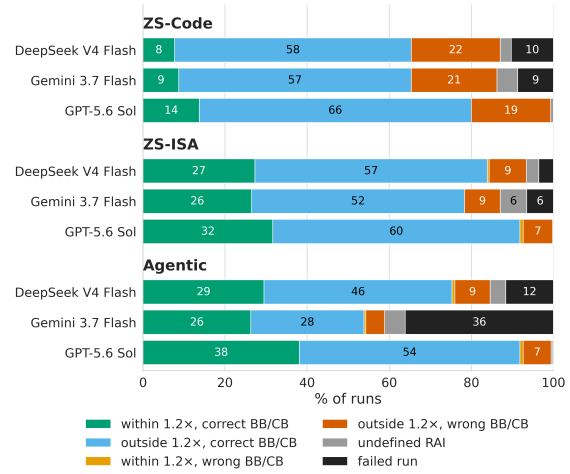


图 7：正确的 Roofline 分类通常能够经受数值 RAI 误差。各段分别展示了 1.20×中不同模型和策略的数值准确率与 BB/CB 分类结果。绿色和蓝色段分别表示估计值在容差范围内和超出容差范围、但分类仍然正确的情况；失败的运行和未定义的 RAI 估计值则单独显示。各段标签均四舍五入为整数百分比。

度下，ZS-ISA的准确率达到 48–60%。附录表 A.14显示，在 1.20× 处，三个编译器证据带来的提升的 95% 区间均为正值；这里采用配对案例自助法，将每个案例及其在两种策略下的五次运行作为一个整体进行重采样。

与内存流量估计相比，浮点运算次数估计更能从编译器证据中获益。在 ZS-ISA 下，总浮点运算次数准确率达到 59–88%；其中 gpt-5.6 从 16% 提升至 88%。在 ZS-ISA 下，设备内存总流量的准确率较低，为 36–40%，仍是数值 RAI 误差的常见来源。与 ISA 加载/存储字节总量不同，测得的流量取决于调度期间的有效缓存复用、驱逐和回写。

原始 CUDAlavaMD案例体现了这一难点：在 A10 上测得的流量为 290MB，而在 A100/H100 上为 149–152 MB。对于 gpt-5.6 在ZS-ISA下，预测流量中位数为 0.92–0.93× A100/H100 上参考值的 1.20× 容差范围内，但 0.67×在 A10 上则位于区间之外；附录表 A.13（a）给出了匹配的测量结果。

> **RQ1 总结：**编译器证据显著提升了数值 *RAI* 预测，使其在测试模型中的一致性提高了 1.20× *18–20* 个百分点。在 *ZS-ISA* 下，总浮点运算次数准确度为 *59–88%*，而设备显存流量准确度为 *36–40%*，因此流量估计仍是一个反复出现的薄弱环节。

4.2 RQ2：预测的 RAI 能否正确识别 GPU 内核的 Roofline 区域?

尽管存在数值误差，预测的 RAI 通常仍能识别正确的 Roofline 区域。图 7 分别展示了每个模型和策略的数值 RAI 一致性与 BB/CB 分类结果。对于 gpt-5.6 在 ZS-ISA下的运行，容差范围内和范围外的正确分类（绿色加蓝色）合计占运行次数的 91.8%，相比之下，RAI 落在 1.20×参考值容差范围内的运行仅占 32.4%。**在所有 gpt-5.6ZS-ISA 运行中，有 60.2% 的运行其数值估计值超出该容差范围，但 Roofline 分类仍然正确。**

分类准确率从ZS-Code下的 65–80% 提升至ZS-ISA 下的 78–92%，适用于三个模型。179 个评估案例中，依据剖析得到的 RAI，其中 93 个属于带宽受限（BB），86 个属于 CB，因此始终预测为 BB 的准确率为 52%。一个简单的基线是针对每种 GPU–精度组合选择更常见的参考类别；无需读取代码，准确率即可达到 73.2%。附录中的表 A.8给出了各组的计数。八种类别选择均基于同一批 179 个案例进行拟合和评估，因此这是一次

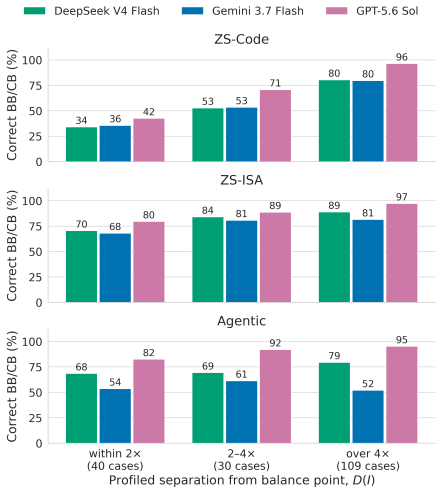


图 8：编译器证据能提升平衡点附近的 Roofline 分类准确率。各面板比较了 ZS-Code、ZS-ISA 和 AG 下的分类准确率，并按剖面化间隔 $D(I) \leq 2$、$2 < D(I) \leq 4$ 和 $D(I) > 4$对案例分组。各组分别包含 40、30 和 109 个案例；每个模型和策略对每个案例运行五次。颜色区分不同模型；柱状图标签四舍五入为整数百分比，失败或不完整的预测均计为未命中。

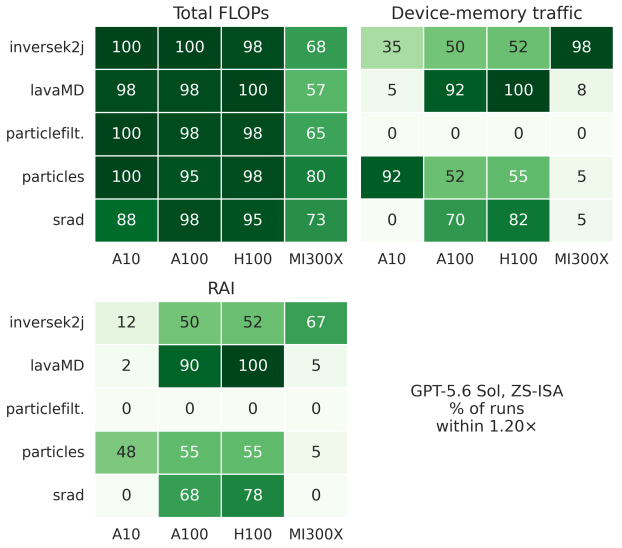


图 9：gpt-5.6 的数值准确率因工作负载–GPU 组合而异。各面板展示了 ZS-ISA 在剖面化总浮点运算次数、总设备内存流量和 RAI 的 1.20×范围内的准确率。变体和运行时合并统计；每个 NVIDIA 单元格包含 40 次运行，每个 MI300X 单元格包含 60 次，而 MI300X 上的粒子包含 55 次。失败的预测计为未命中。单元格数值四舍五入为整数百分比。不同厂商使用的运行时集合不同（NVIDIA 上为 CUDA 和 OpenMP；MI300X 上为 HIP、OpenMP 和 OpenCL），因此单元格比较的是经过评估的目标。

样本内比较。在`ZS-ISA`下，三个模型对单个计算内核的分类都比这一粗略规则更准确，尽管该规则是根据评估标签拟合的；而在`ZS-Code`下，只有`gpt-5.6` 做到了这一点。附录中的表 A.7证实，`gpt-5.6`的`ZS-ISA` 准确率在两种模式下均成立，对 BB 和 CB 的召回率也相近。

在平衡点附近，编译器证据很有帮助。在平衡点 B 附近，分类对数值误差更加敏感：较小的误差就可能改变预测的运行区间，而在距离更远处，较大的误差也可能不会使其发生变化。对于正 RAI I，我们将距 B 的乘法距离量化为 $D(I) = \max(I/B, B/I)$。该因子具有对称性：在平衡点处等于 1，在 $B/4$ 或 $4B$ 处等于 4。图 8 按剖面化距离 $D(I)$ 对分类结果进行分组，而不是按数值容差分组。对于距平衡点在 $2\times$ 以内的案例，gpt-5.6 的分类准确率从 ZS-Code 下的 42.5% 提升至 ZS-ISA 下的 79.5%，增益为 37 个百分点。该增益的配对案例自助法95%区间为 $[19.5, 54.0]$ 个百分点，不包含零，支持通过案例重采样获得正向改进。对于距离超过 $4\times$ 的案例，gpt-5.6 在每种策略下的准确率都超过 95%。

RQ2 总结： 编译器证据使 *gpt-5.6* 在距平衡点 $2\times$ 以内的 *40* 个案例中的分类准确率从 *42.5%* 提升至 *79.5%*。在所有案例中，*ZS-ISA* 分类在 *78–92%* 的运行中均为正确分类，并且通常能够在存在数值误差的情况下保持运行区间不变。

4.3 研究问题3：数值 RAI 准确度是否因工作负载、目标 GPU 和代码变体而异?

数值 RAI 准确度因工作负载、目标 GPU 和代码变体而异。图 9展示了`ZS-ISA`在不同工作负载和 GPU 上针对`gpt-5.6`的 RAI、浮点运算次数和流量的准确率。我们重点研究整体数值准确率最高的模型，以考察其成功案例和仍存在的局限性。`lavaMD`在 1.20× 上达到 90% 的准确率，在 H100 上达到 100%，而在 A10 上为 2.5%，在 MI300X 上为 5%。相比之下，`particlefilter` 在任何 GPU 上都没有准确的 RAI 估计值。GPU 的排名也取决于工作负载：`inversek2j` 在 MI300X 上的准确率最高。附录中的图 A.19将分析扩展到所有模型以及各个运行时–变体组合；不同模型的确切 GPU 排名有所不同。

NVIDIA 单元格的总浮点运算次数准确度达到 88–100%；

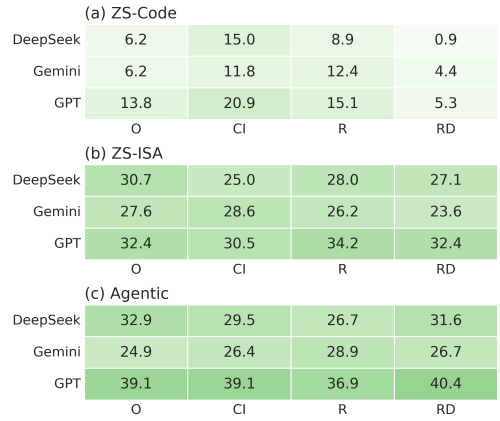


图 10：编译器证据带来的提升适用于所有四种代码变体。单元格显示每种模型和策略的数值 RAI 准确度（%），并汇总了不同工作负载、GPU 和运行时的数据。各列依次为原始版本（**O**）、修改后的标志/输入（**CI**）、重命名版本（**R**）以及更改数据类型后的重命名版本（**RD**）。除**CI** 有 220 次外，每个单元格均包含 225 次运行；失败和不完整的预测均计为未命中。

流量仍是常见限制。在 MI300X 上，FLOP 计数错误也会造成影响：对于`inversek2j`，流量准确率达到 98%，但 FLOP 准确率为 68%，RAI 准确率为 67%。FLOP 准确率还要求与目标性能分析器的计数约定相匹配。附录表 A.5 在已核对的GPT-5.6 ZS-ISA inversek2j 预测中说明了这一点：对打包算术和除法辅助函数的计数不匹配，导致浮点运算次数估计值与 RAI 估计值均未落入容差。分量误差可能叠加：估计值若处于 1.20× 以内（对浮点运算次数和流量两者而言），只能保证 1.44×的 RAI 界限。对于`gpt-5.6`在粒子/A10 上，40 次中有 18 次`ZS-ISA`运行同时满足两项分量容差，却未达到 RAI 目标。

对相邻方框的重复访问可在 `lavaMD` 中实现缓存复用。**O**CUDA `particlefilter` 会写入并重新读取一个 110MB 的临时数组，相邻线程访问的条目间隔 276 字节。这些跨步访问会使流量对事务粒度和有效复用情况十分敏感，即使浮点运算次数估计准确也不例外。附录 表 A.13(a) 显示，26× 对于全部四种 GPU 上的 **O** CUDA/HIP`particlefilter` 案例，流量预测中位数比参考值低 4–。

编译器证据提升了所有变体的准确率。图 10 比较了不同模型和策略在名称、编译器标志、输入或数据类型有所变化的各个变体中的 RAI 准确率。`ZS-ISA` 在 所有模型—变体组合中均优于 ZS-Code。**CI** 提升了`ZS-Code`相对于 **O**forall 三种模型的准确率，但效果不一

表 1：在 `ZS-ISA` 和 智能体引导下的准确率、完成状态、记录成本和预测延迟（每种配置运行 895 次）。成本和延迟为中位数，比例在四舍五入前计算；RAI 准确率采用 1.20× 容差。完成状态表示预测可用。附录图 A.14 和 A.15展示了相应的逐次运行分布。未记录的费用不计入。

策略	每次运行成本（美元）中位数	延迟（分钟）中位数	RAI 准确率 (%)	完整 (%)
DeepSeek				
ZS-ISA	0.0060	7.6	27.7	96.4
AG	0.0405	22.0	30.2	88.4
Gemini				
ZS-ISA	0.0400	0.62	26.5	93.5
AG	0.709	7.6	26.7	63.9
gpt-5.6				
ZS-ISA	0.223	1.3	32.4	100.0
AG	0.800	3.4	38.9	99.8

ZS-ISA 的影响`ZS-ISA`。与 **O** 相比，**R** 对不同模型和策略的准确率没有一致的方向性影响。 **RD** 会降低 `ZS-Code` 的准确率，相较于 **O**，其中对 DeepSeek和 `gpt-5.6`的影响最为明显；`gpt-5.6`的**O** 和 **RD**准确率在`ZS-Code`下分别为 13.8% 和 5.3%，而在 `ZS-ISA` 下均升至 32.4%。附录中的表 A.12 显示，`gpt-5.6`的各个变体几乎都能完成预测，而其他模型的完成情况则因变体而异。

研究问题3总结： 数值 *RAI* 准确度因工作负载和目标设备而异：对于 *lavaMD*，在 *A100/H100* 上，*gpt-5.6* 使用 *ZS-ISA* 时的准确率达到 *90–100%*（ 1.20×内），但对于 *particlefilter*，在所有图形处理器上均为 *0%*。编译器证据带来的提升适用于全部四种代码变体，而流量估算仍是一个反复出现的薄弱环节。

4.4 研究问题4：沙箱分析能否在零样本ISA提示之外提升数值 RAI 准确度?

智能体提示可提升 `gpt-5.6` 的端到端数值RAI准确率，但其他模型的增益仍不确定。 表 1 比较了三种模型采用这两种策略时的准确率、完成率、成本和延迟。对于`gpt-5.6`，在 1.20× 中，准确率从 32.4% 提升至 38.9 %，增加了 6.5 个百分点，配对案例自助法95%区间为 $[3.4, 9.6]$ 个百分点。该提升的代价约为 3.6×记录成本中位数，以及 2.5× 延迟中位数（相较于`ZS-ISA`）。

观察到的智能体引导RAI 准确率分别提升了 2.5 个百分点（ DeepSeek ）和 0.2 个百分点（Gemini）。附录A.15显示，这些差异的案例自助法区间同时涵盖了下降和上升，因此无法确定其增益。

。**完成状态较低也影响了端到端结果：智能体引导的完成状态分别降至 88% 和 64%**。分类准确率也分别降至DeepSeek 的 75% 和Gemini的 54%，而`gpt-5.6` 仍为 91.8%。附录图 A.13 显示，耗尽 100 次响应预算是导致智能体引导运行失败的主要原因，涉及DeepSeek和Gemini。

沙盒支持超越固定提示词的分析：经检`gpt-5.6` `lavaMD` 查的轨迹构建了特定于 GPU 的 LRU缓存模拟器代码。附录 A.14 记录了不同 GPU 之间准确率关联的差异；该观察性比较并未将模拟收益单独分离出来。

在 `ZS-ISA` 下，`gpt-5.6`的数值 RAI 准确率最高，`deepseek`的记录成本中位数最低，`gemini` 的延迟中位数最低。**智能体引导使每次运行的成本中位数增加** 3.6–17.7×**倍，但只有 gpt-5.6 的数值准确率获得了有明确支持的 6.5 个百分点增益**。

> **研究问题4总结：**智能体引导使 *gpt-5.6* 在 1.20× 内的数值 *RAI* 准确率相较于 *ZS-ISA* 提高了 *6.5* 个百分点，而成本为 3.6× 的中位数。对于 *deepseek* 和 *gemini*，智能体引导的成本是 *ZS-ISA* 的 6.7–17.7× 倍，增益并不确定，且完成状态更低。

4.5 范围与有效性

在 2,000 美元的实验预算内，我们将研究范围限定为五个内核族和三种大语言模型，以优先比较不同硬件、运行时、代码变体和策略。重复预测反映的是这些内核族内部的变异性，而非额外的工作负载多样性。在 179 个案例中，有 109 个与剖面化平衡点的距离超过 4×，因此应结合距离细分来解读总体分类准确率。

对每个案例进行十次性能分析试验后，RAI 的逐案例变异系数中位数为 0.08%，最大值为 10.3%；附录 A.8 给出了 GPU细分。我们为每种策略采用固定的提示设计，不评估对其他措辞的敏感性。我们不将数值估算结果与专用的静态FLOP计数或内存流量模型进行比较。

5 结论

在这项工作中，我们展示了大语言模型能够根据静态证据，估算不同硬件目标、并行运行环境及修改后代码变体中的 GPU 内核的 Roofline 算术强度。通过实证，我们表明，与仅使用源代码提示相比，使用编译后汇编指令产物提示大语言模型，可显著提高算术强度估算的准确性

。尽管估算结果往往存在误差，但根据所用预测大语言模型的不同，最终的 Roofline 带宽受限或计算受限分类在 78–92% 的运行中仍然正确。我们的结果表明，特定于硬件的浮点运算计数约定可能导致浮点运算次数估计误差，而设备显存流量估算仍是这种方法的主要局限，因为这需要在目标性能分析器的计数器边界上核算有效缓存复用。我们还强调，与零样本提示相比，智能体式大语言模型提示有时会妨碍任务完成，而在其他情况下，它对算术强度预测准确率的提升微乎其微，成本却高达原来的 17.7× 倍。因此，大语言模型可以根据静态证据帮助识别内核可能受 Roofline 带宽限制还是计算限制；而缓存模拟和内存流量工具能否改善算术强度数值估算，仍有待进一步研究。

致谢

本研究由 Code Metal Inc. 和 Pazy Foundation 资助。本研究部分是在美国能源部支持下，由劳伦斯利弗莫尔国家实验室根据合同 DE-AC52-07NA27344（LLNL-JRNL-2026894）完成。作者的观点和意见不一定代表美国政府或劳伦斯利弗莫尔国家安全有限责任公司的观点和意见；上述机构及其任何雇员均不作任何明示或暗示的认可、保证或陈述，也不对本文所含信息的准确性、完整性或有用性承担任何法律责任或义务。美国政府保留相关权利；出版方接受本文发表，即承认美国政府保留非独占、已付费、不可撤销、全球范围的许可，可为美国政府之目的发表或复制本手稿的已发表版本，或允许他人这样做。

参考文献

Advanced Micro Devices, Inc. ROCm 计算分析器：流水线指标——浮点运算计数约定，2026年。网址 https://rocm.docs.amd.com/projects/rocprofiler-compute/en/docs-7.2.0/conceptual/pipeline-metrics.html#flop-counting-conventions。ROCm 7.2.0 文档。

Akhauri, Y.、Song, X.、Wongpanich, A.、Lewandowski,B. 和 Abdelfattah, M. S. 代码回归语言模型，2026年。网址https://arxiv.org/abs/2509.26476。

Alavani, G.、Varma, K. 和 Sarkar, S. 使用静态分析预测 CUDA 内核的执行时间。载于*2018 IEEE* 国际并行与分布式处理及其应用会议（*ISPA/IUCC/BDCloud/Social-Com/SustainCom*），第 948–955 页，2018 年。doi：10.1109/ BDCloud.2018.00139。

Arafa, Y.、Badawy, A.-H. A.、Chennupati, G.、Santhi, N. 和 Eidenbenz, S. PPT-GPU：可扩展的图形处理器性能建模。《*IEEE*计算机体系结构快报》，18(1)：55–58，2019 年。doi：10.1109/LCA.2019.2904497。

Baghsorkhi, S. S.、Delahaye, M.、Patel, S. J.、Gropp, W. D. 和 Hwu, W.-m. W. 面向图形处理器架构的自适应性能建模工具。载于第 *15* 届 *ACM SIGPLAN* 并行编程原理与实践研讨会（*PPoPP*）论文集，第 105–114 页，2010 年。doi：10.1145/1693453.1693470。

Baronio, C.、Marsella, P.、Pan, B.、Guo, S. 和 Alberti, S. Kevin：用于生成 CUDA 内核的多轮强化学习，2025 年。arXiv：2507.11948。

Bolet, G.、Georgakoudis, G.、Menon, H.、Parasyris, K.、Hasabnis, N.、Estes, H.、Cameron, K. 和 Oren, G. 大语言模型能否预测并行代码性能？载于第 *34* 届国际高性能并行与分布式计算研讨会（*HPDC*）论文集，第 1–6 页，2025a 年。doi：10.1145/ 3731545.3743645。

Bolet, G., Georgakoudis, G., Parasyris, K., Menon, H., Hasabnis, N., Cameron, K. W., and Oren, G. 无需运行即可计数：评估大语言模型对代码复杂度的推理，2025b。网址 https://arxiv.org/abs/2512.04355。

Braun, L.、Nikas, S.、Song, C.、Heuveline, V. 和 Fr¨oning, H. 一种用于便携式且快速预测 GPU 内核执行时间和功耗的简单模型。*ACM Transactions onArchitecture and Code Optimiza-tion*，18(1)：7:1–7:25，2021年。doi：10.1145/3431731。

Che, S., B$_{oyer,}$ M., Meng, J., Tarjan, D., Sheaffer, J. W。，Lee, S.-H., and Skadron, K. Rodinia：异构计算基准测试套件。发表于*2009 IEEE*国际工作负载特征研讨会（*IISWC*），第44–54页，2009年。doi：10.1109/IISWC.2009. 5306797。Chen, W., Zhu, J., Fan, Q., Ma, Y., and Zou, A. CUDA-大语言模型：大语言模型能够编写高效的CUDA内核，2025年。arXiv:2506.09092。

Cummins, C., Seeker, V., Grubisic, D., Roziere, B., Gehring, J., Synnaeve, G., 和 Leather, H. Meta 大语言模型编译器：用于编译器优化的基础模型，2024。arXiv:2407.02524。

Emonds, Y., Braun, L., 和 Fr¨oning, H. CUDAsap：将静态确定的执行统计数据作为基于执行的性能分析的替代方案。发表于*2023 IEEE/ACM* 第 *23* 届国际集群、云计算与互联网计算研讨会（*CCGrid*），第 119–130 页，2023。doi: 10.1109/CCGrid57682.2023.00021。

Guerreiro, J., Ilic, A., Roma, N., 和 Tom´as, P. 使用 PTX 和深度结构化学习进行图形处理器静态建模。*IEEEAccess*，7:159150–159161，2019。doi: 10.1109/ ACCESS.2019.2951218。

Hong, S. 和 Kim, H. 一种兼顾显存级并行性与线程级并行性的 GPU 架构分析模型。*SIGARCH*计算机体系结构新闻，37（3）：152–163，2009。doi：10.1145/1555815.1555775。

Jaber, J. 和 Jaber, O. AutoKernel：通过迭代式智能体驱动搜索自主优化图形处理器内核，2026年。网址 https://arxiv.org/abs/2603.21331。

Jin, Z. 和 Vetter, J. S. 用于提升 SYCL编程模型性能可移植性的基准测试套件。载于*2023*年*IEEE*国际系统与软件性能分析研讨会（*ISPASS*），第325–327页，2023年。doi：10.1109/ISPASS57527.2023. 00041。

Kadosh, T.、Hasabnis, N.、Vo, V. A.、Schneider, N.、Krien, N.、Wasay, A.、Ahmed, N.、Willke, T.、Tamir, G.、Pinter, Y.、Mattson, T. 和 Oren, G. 范围即一切：面向高性能计算代码的大语言模型转型，2023年。网址 https://arxiv.org/abs/2308.09440。

Kadosh, T., Hasabnis, N., Vo, V. A., Schneider, N., Krien, N., Capot˘a, M., Wasay, A., Tamir, G., Willke, T., Ahmed, N., Pinter, Y., Mattson, T., 和 Oren, G. MonoCoder：面向高性能计算代码和任务的领域专用代码语言模型。载于*2024 IEEE* 高性能极限计算会议 *(HPEC)*，第 1–7 页，2024 年。doi：10.1109/HPEC62836.2024.10938441。

Khairy, M., Shen, Z., Aamodt, T. M., 和 Rogers, T. G. Accel-sim：用于经验证的图形处理器建模的可扩展仿真框架。载于*2020 ACM/IEEE 第 47 届年度国际计算机体系结构研讨会 (ISCA)*，第 473–486 页，2020 年。doi: 10.1109/ISCA45697. 2020.00047。

Khan, Z., Chen, J. C.-Y., Cho, J., Stengel-Eskin, E., and Bansal, M. GPU Forecasters：将语言模型用作内核运行时优化的选择性代理，2026年。网址 https://arxiv.org/abs/2605.31464。

Lee, J., Ha, Y., Lee, S., Woo, J., Lee, J., Jang, H., and Kim, Y. GCoM：用于现代 GPU 准确解析建模的详细 GPU 核心模型。载于第*49*届年度国际计算机体系结构研讨会（*ISCA*）论文集，第424–436页，2022年。doi: 10.1145/ 3470496.3527384。

Lim, R., Norris, B., and Malony, A. 通过静态分析和预测分析对 GPU 内核进行自动调优。载于*2017*年第*46*届并行处理国际会议（*ICPP*），第523–532页，2017年。doi: 10.1109/ICPP.2017.61。

Liu, S., Dong, X., Cao, J., Chu, R., Wang, Z., Wang, Q., 和 Bai, X. Mapo：由性能模型驱动的 GPU内存访问代码优化。*FutureGeneration ComputerSystems*，184:108606，2026年。ISSN 0167-739X。doi: 10.1016/j.future.2026.108606。网址: https://www.sciencedirect.com/science/article/pii/S0167739X26002402。

Lopes, A., Pratas, F., Sousa, L., 和 Ilic, A. 使用缓存感知 Roofline 建模探索 GPU性能、功耗与能效上界。载于*2017* 年 *IEEE* 系统与软件性能分析国际研讨会（*ISPASS*），第 259–268 页，2017 年。doi: 10.1109/ISPASS.2017.7975297。

Microsoft。由 Azure 直接销售的模型。在线文档，2026 年。访问日期：2026-08-24。

Mo, Z., Cheng, Y., Wang, L., Tang, Z., Xu, L., Li, G., Dong, Y., Ma, L., Xia, Y., Xue, J., Yang, F., Mai, L., Yang, Z., Luk, W., and Fan, H. TileSight：一种从核心到集群、基于第一性原理且以瓦片为中心的图形处理器性能解析模型，2026年。网址 https://arxiv.org/abs/2607.22432。

Nguyen-Nhat, M.-K., Do, H. D. N., Le, H. T., 和 Dao, T.T. LLMPerf：GPU性能建模与大语言模型相结合。载于*2024* 年第 *32* 届计算机与电信系统建模、分析与仿真国际会议（*MASCOTS*），第 1–8 页，2024 年。doi: 10.1109/MASCOTS64422.2024.10786558。

Nichols, D., Davis, J. H., Xie, Z., Rajaram, A., 和 Bhatele, A. 大语言模型能编写并行代码吗？载于第 *33* 届高性能并行与分布式计算国际研讨会（*HPDC*）论文集，第 281–294 页，2024a。doi: 10.1145/3625549. 3658689。Nichols, D., Marathe, A., Menon, H., Gamblin, T., 和 Bhatele, A. HPC-Coder：使用大语言模型对并行程序建模。载于*ISC* 高性能计算 *2024* 研究论文集，第 1–12 页，2024b。doi: 10.23919/ISC.2024.10528929。Nichols, D., Parasyris, K., Jekel, C., Bhatele, A., 和 Menon, H. 将性能工具集成到模型推理中，以实现 GPU 内核优化，2025 年。arXiv:2510.17158。

Novikov, A., V˜u, N., Eisenberger, M., Dupont, E., Huang, P.-S., Wagner, A. Z., Shirobokov, S., Kozlovskii, B., Ruiz, F. J. R., Mehrabian, A., Kumar, M. P., See, A., Chaudhuri, S., Holland, G., Davies, A., Nowozin, S., Kohli, P., and Balog, M. AlphaEvolve：用于科学与算法发现的代码编写智能体，2025年。arXiv:2506.13131。

NVIDIA。Nsight Compute：已知问题。在线文档，2026年。性能分析与指标。访问日期：2026-10-30。

NVIDIA Corporation。CUDA 示例：粒子——粒子，2018。网址 https://docs.nvidia.com/cuda/archive/10.0/cuda-samples/index.html#particles。CUDA 工具包 10.0 文档。

OpenAI。GPT-5.5 模型文档。在线文档，2026年。访问日期：2026-08-24。

OpenRouter。模型 API 与提供商路由。在线文档，2026a。访问日期：2026-08-24。

OpenRouter。推理令牌。在线文档，2026b。访问日期：2026-08-24。

Ouyang, A.、Guo, S.、Arora, S.、Zhang, A. L.、Hu, W.、R´e, C. 和 Mirhoseini, A. KernelBench：大语言模型能编写高效的图形处理器内核吗？，2025。arXiv:2502.10517。

Peng, Y.、Gotmare, A. D.、Lyu, M. R.、Xiong, C.、Savarese, S. 和 Sahoo, D. PerfCodeGen：通过执行反馈改进大语言模型生成的代码的性能。载于*2025 IEEE/ACM* 第二届人工智能基础模型与软件工程国际会议（*Forge*），第 1–13 页，2025 年。doi: 10.1109/Forge66646.2025.00008。

Tran, H.-Q.、Pham, V.-S.、Pham, T.-S.、Chu, T.-D.、Mai, A.-T.、Nguyen, X. T. 和 Dao, T.-T. 基于 PTX 特征和 XGBoost 的基于 Triton 的图形处理器内核精确延迟预测器。载于 *2025 RIVF* 计算与通信技术国际会议（*RIVF*），第 980–985 页，2025 年。doi：10.1109/RIVF68649.2025.11365185。Tschand, A.、Awad, M.、Swann, R.、Ramakrishnan, K.、Ma, J.、Lowery, K.、Dasika, G. 和 Reddi, V. J. SwizzlePerf：用于图形处理器内核性能优化的硬件感知大语言模型，2025 年。arXiv：2508.20258。Wang, Z.、Ramos, C.、Awad, M. A. 和 Lowery, K. Omniwise：利用大语言模型预测图形处理器内核性能，2025 年。arXiv：2506.20886。Wen, Z.、Zhang, Y.、Li, Z.、Liu, Z.、Xie, L. 和 Zhang, T. MultiKernelBench：用于内核生成的多平台基准测试，2025 年。URL https://arxiv.org/abs/2507.17773。Williams, S.、Waterman, A. 和 Patterson, D. A. Roofline：一种富有洞察力的多核架构可视化性能模型。*Communications ofthe ACM*，52(4)：65–76，2009 年。doi：10.1145/1498765.1498785。Yazdanbakhsh, A.、Mahajan, D.、Esmaeilzadeh, H. 和 Lotfi-Kamran, P. AxBench：用于近似计算的多平台基准测试套件。*IEEE Design & Test*，34(2)：60–68，2017 年。doi：10.1109/MDAT.2016.2630270。Zhang, G.、Zhu, S.、Wei, A.、Song, Z.、Nie, A.、Jia, Z.、Vijaykumar, N.、Wang, Y. 和 Olukotun, K. AccelOpt：用于人工智能加速器内核优化的自我改进型大语言模型智能体系统。载于机器学习与系统论文集（*MLSys*），2026 年。URLhttps://arxiv.org/abs/2511.15915。Zhang, Z.、Wang, R.、Li, S.、Luo, Y.、Hong, M. 和 Ding, C. CudaForge：具有硬件反馈的 CUDA 内核优化智能体框架，2025 年。URLhttps://arxiv.org/abs/2511.01884。

人工智能使用披露

我们既将生成式人工智能作为实验的研究对象，也利用它协助实施研究和准备手稿。方法部分介绍了作为预测器进行评估的模型、它们的输入以及它们在生成实验输出中的作用。以下内容介绍了用于协助完成本研究的人工智能工具。在实施研究和准备手稿时，我们使用了 OpenAI 的 Codex 助手，以及采用 Anthropic Claude Opus 模型的智能体。

研究规划与实现。在作者的指导下，大语言模型智能体编写了实验实现和数据收集软件的大部分内容。人工智能助手还协助开发分析脚本、生成图表和表格，并将报告的结果与分析输出进行核对。它们提出了改进指标、统计分析、可视化和研究结果呈现方式的建议；作者决定采纳哪些建议。

基准测试和数据。参考浮点运算次数和设备显存流量值是使用供应商分析器收集的硬件性能计数器测量结果。大语言模型生成的浮点运算次数和流量估算值，是与这些参考值进行对比评估的预测结果。智能体还协助完成了本研究编写的 OpenCL 移植版本。作者通过将这些移植版本的真实性能剖面与原始实现进行比较，检查了移植版本，力求在保留源代码语义的同时，尽可能贴近原始实现的性能。

写作和文献检索。作者撰写了论文所有章节的初稿，并使用人工智能助手修改、编辑和评议手稿。助手还协助拟定提纲、在作者指导下起草并大幅改写部分段落，以及完善图表说明。作者反复审阅建议、提出修改意见，并作出最终编辑决定。人工智能助手还帮助查找遗漏的相关论文、检索书目记录，并根据可获取的一手来源核对引文。

作者对论文、软件及其报告结果的正确性、原创性和完整性承担全部责任，其中包括借助 AI 生成的内容。

A 附加详情

A.1 模型和端点配置

表A.2记录了实验规划所采用的模型与端点配置快照。价格采用研究注册表中冻结的、未缓存的标准费率，以每百万输入和输出令牌计价；这不一定是网关当前公布的最便宜路由。资源比较采用记录的实验成本，包括根据报告用量估算的费用。提供商目录和端点限制变化频繁，因此我们报告快照日期，并区分模型级限制与依赖路由的限制（OpenRouter，2026a；b；OpenAI，2026；Microsoft，2026）。

未单独设置推理令牌限制，并不意味着推理没有限制。在这些行中，隐藏推理会与可见输出争用端点的补全限制。OpenRouter 的努力程度标签可能会影响提供商的分配，但无法确立可移植的硬性令牌限制。因此，对于取决于路由的模型，实测成本和延迟结果必须同时注明确切的提供商路由以及响应报告的用量。尤其是，尽管 GLM-5.3 宣称最大输出为 131,072 个令牌，但我们发现，在未由客户端施加输出限制的请求中，实际停止点为 65,536 个令牌；我们将其视为端点行为，而非普遍适用的模型限制。

A.2 选定的目标内核和硬件

表A.3 总结了选定的五种内核族、其实现来源、不同变体下测得的 Roofline 区域，以及剖析所用图形处理器的规格。

A.3 内核族选择

我们分四个阶段选出了五个内核族，仅使用性能分析计数器，不使用大语言模型预测。这里的一个案例是一个目标内核单元：某个变体、运行时和图形处理器下的某个内核族的目标内核，不包括**R**变体，因为重命名不会改变其计数器。

候选项。很少有程序同时提供 CUDA、HIP、Open MP 卸载和 OpenCL 实现：HeCBench 提供前三种实现，但不提供 OpenCL，因此 OpenCL 实现来自 Rodinia，或是在本研究中移植而来。在这一约束下，我们在全部四个 GPU 上对八个候选族进行了性能分析：bfs、inversek2j、lavaMD、lud、particlefilter、particles、srad 和 threadfence。

准入条件。 每个候选项都必须具备<样式 id='3'>O 变体的全部九种 GPU–运行时组合，以及一个不依赖于运行相同源代码和输入的 NVIDIA GPU 的主导精度浮点运算次数计数；任何高于的跨运行时算术强度差异都必须由有文档记录的机制解释，并且测量结果必须干净。

有两个系列未通过 FLOP 稳定性要求。threadfence 的 FLOP 计数包括依赖争用的原子重试，因此其主导精度浮点运算次数计数在 A10、A100 和 H100 之间相差 37 %，无法得到可用于预测的机器无关值；其他每个系列的差异最多为 2×10^{-5}。bfs 仅使用整数运算，因此没有有意义的算术强度，而且其少量 FLOP 计数也会因 GPU 而异。跨运行时差异并不是排除理由：inversek2j 在 MI300X 运行时之间的 $10.9\times$ 算术强度差异，源于 OpenCL 中更为复杂的 acos 和 sqrt 库实现；我们已在反汇编中确认了这一点，并将其视为一个特意设置的困难案例。

结构约束。有效集合必须包含一个饱和的计算受限锚点和一个饱和的带宽受限锚点，并且至少包含一个其状态沿图形处理器、运行时和变体这三个轴中的两个或更多轴发生变化的成员。只有 lavaMD 符合计算受限锚点的条件，只有 particlefilter 符合带宽受限锚点的条件，因此剩下四个有效的五族集合。

排名。 我们根据各有效集合中距离相关平衡点不超过 1.5 个倍频程的案例占比对其进行排名（表 A.4），因为远离平衡点时的准确率反映的是较容易的分类问题。我们还记录了计算受限/带宽受限平衡，以及同时具有相同工作形态（固定迭代、数据依赖或单遍）和相同内存访问模式（流式访问、通过共享内存进行分块访问或间接访问）的成员对数量。选定的集合排名第一，并且它是唯一没有此类重复成员对的有效集合（`lavaMD` 和 `lud` 同时具备这两种属性）。在采用八种替代选择重新进行排名时（使用平均距离而非中位数距离、将近脊阈值设为 1.0 和 2.0 个倍频程、纳入 **R** 变体，以及依次去除每个图形处理器），该集合在其中六种情况下仍排名第一；2.0 个倍频程的阈值和去除 H100 则倾向于用 `srad`替换 `lud`。计数器噪声不会影响排名：没有任何案例的状态处于其平衡点三个标准差以内。没有符合条件的族会执行原子操作，因此本研究不涵盖原子操作密集型内核。

A.4 智能体工具使用与未完成的智能体分析

图 A.11 详细列出了通过`execute` 发出的命令，以及所有智能体引导运行和重复中的记录时长。Python/算术命令分别占`deepseek-v4-flash`、`gemini-3.7-flash`和`gpt-5.6-sol`的 execute 调用次数的 38%、62% 和 52%，而在各模型的 execute 调用总时长中约占 50–57%。

表 A.2：模型与端点对比（配置快照：2026-08-24）。上下文通常指输入输出窗口总和，而非可独立使用的输入额度。标注为 [C] 的价格是保守的路由规划费率；[S] 表示以项目额度而非研究经费支付的影子/标价估值。

模型	发布日期	每 100 万令牌的价格（输入/输出）	上下文限制	输出限制	推理令牌限制
GPT-5.6 Sol	2026-07-09	$5.50 / $33.00[C]	1,050,000	128,000	未单独公布上限；推理和可见输出共用补全预算。
Claude Opus 5	2026年-07-24	$5.00 / $25.00	1,000,000	128,000	OpenRouter 将显式推理分配额度限制为 128,000；最终答案也必须符合限制。
Kimi K3	2026年-07-16	$3.00 / $15.00[C]	1,048,576	取决于路由；目录最大值约974,842	没有单独公布的限制；受服务路由的补全预算限制。
GPT-5.5（Azure）	2026年-04-24	$5.00 / $30.00[S]	1,050,000 个模型；约 922,000 Azure 有效输入预算	128,000	没有单独的限制；推理和可见输出共用 128,000 个令牌生成预算。
Gemini 3.7 Flash	2026年-08-13	$0.375 / $1.875	1,048,576	65,536	没有公开的按推理力度划分的上限；提供商在内部决定推理的使用方式。
Qwen 3.8 Max	2026年8月3日	$2.00 / $6.00	1,000,000	131,072	未单独公布限制；受补全预算限制。
GLM-5.3	2026年8月18日	$1.40 / $4.40	1,048,576	宣传为 131,072；65,536 有效默认值观测值	未单独公布上限；受以下因素限制补全预算。
DeepSeek V4 Pro 0813	2026-08-12	$1.32 / $3.96[C]	1,048,576	384,000 目录最大值；取决于路由	未单独公布限制；受服务路由限制。
DeepSeek V4 Flash 0731	2026-07-31	$0.44 / $1.32[C]	最高 1,310,720；取决于路由	262,144 目录最大值；取决于路由	没有单独公布的上限；受服务路由限制。
MiniMax M3	2026-05-31	$0.75 / $3.00[C]	最多 1,048,576；取决于路由	262,144 目录最大值；取决于路由	没有单独公布的限制；受服务路由约束。

该类别包括解析并打印反汇编、统计已执行指令，以及计算候选组件计数器的脚本。编译和构建命令占 gpt-5.6-sol 的 execute 调用总时长的 32%，相比之下，deepseek-v4-flash 为 10%，gemini-3.7-flash 为 12%。deepseek-v4-flash 每次运行的 execute 调用总时长平均为 0.6 分钟，gemini-3.7-flash 为 1.1 分钟，gpt-5.6-sol 为 0.2 分钟。这些时长不包括模型推理和其他工具调用。

图 A.12对比了 95 次`deepseek-v4-flash`运行和 321 次`gemini-3.7-flash`运行中尚未完成的任务，这些运行均耗尽了 100 条响应预算。百分比按每个模型受预算限制的运行次数分别归一化。人工审查根据每次运行最后一次实质性工具操作，为其指定一项主要任务，并参考此前四次实质性操作和更早生成的代码来确定上下文。识别该操作时，不计入空响应和任务列表更新。对于`gemini-3.7-flash`，指令分析和浮点运算次数计算涉及 163 次运行（51%），其次是 76 次运行（24%）中的内存/缓存分析和设备内存流量计算。内存/缓存类别包括占用量计算、访问枚举、缓存驻留分析，以及缓存模型开发或执行。对于 deepseek-v4-flash，OpenMP 运行时和启动几何检查涉及 52 次运行（55%），其次是 29 次运行（31%）中的指令分析和浮点运算次数计算。其余类别包括计数器定义、最终计算和结果准备，以及输入/预处理重建。这些标签描述的是分析结束的位置；它们既不衡量整个运行的时间分配，也无法确定是哪项任务导致未能完成。

A.5 完成可靠性与资源分布

图 A.13区分了所有提示策略中的可用预测与未成功的运行结果。可用预测是指有效的结构化结果，无论其预测计数器或推导出的 RAI 是否准确。对于`deepseek-v4-flash`和`gemini-3.7-flash`而言，响应预算耗尽是最常见的未成功智能体结果，而大多数未成功的零样本运行会返回架构无效结果，或遗漏必需的结果调用。

图 A.14展示了每次计划运行的记录成本分布，以及各模型–策略组合的成本总和。成本包括与

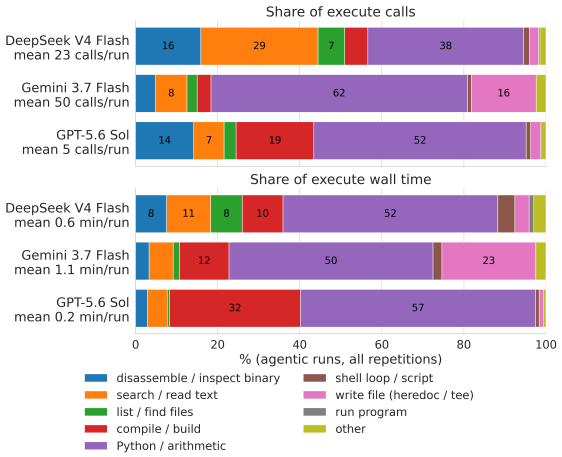


图 A.11：通过 execute 发出的命令：调用占比（上）和调用总时长（下），汇总所有运行和重复。每次调用均根据其第一个实质性命令进行分类，并将该调用的完整时长计入相应类别。时长不包括模型推理和其他工具的耗时。

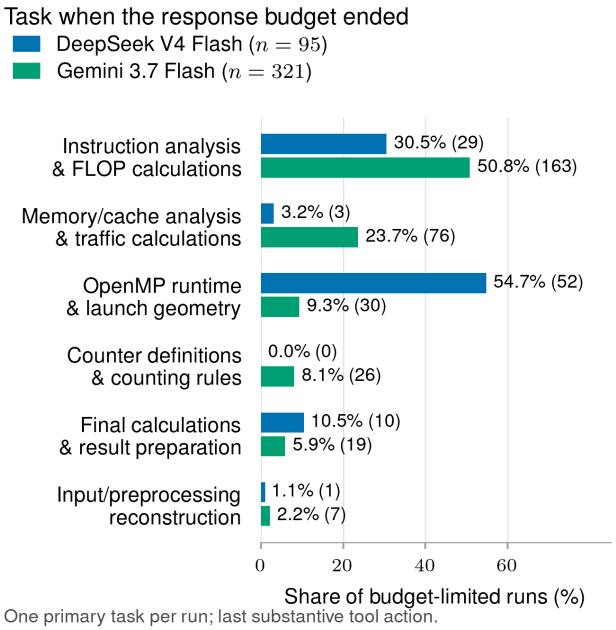


图 A.12：预算耗尽时的主要任务：95 次 deepseek-v4-flash 运行和 321 次 gemini-3.7-flash 运行，分别按各自模型归一化。根据每次运行最后一次实质性工具操作，为其分配一个定性标签，并结合此前的操作和生成的代码作为上下文。标签显示百分比和运行次数。

同一次计划运行相关的未成功提交和更早执行所产生的记录费用。未报告的费用不会补入，因此受影响的单次运行数值和成本总和均为下限。对于 Forgemini-3.7-flash，323 次未成功的智能体运行产生了 294.51 美元的记录成本，占 638.24 美元总记录成本的 46 %。

图 A.15 展示了三种提示策略的每次运行预测延迟分布。对于零样本运行，预测延迟是与计划运行相关的、记录在案的请求链时长之和。对于智能体运行，延迟涵盖从沙盒创建和证据准备，到模型请求、工具调用及重试等待的整个选定执行过程，直至提交预测或执行终止。该时长不包括后续的归档和沙箱清理。一个 `gemini-3.7-flash`智能体运行没有记录时长，因此未计入延迟分布，剩余 894 个观测值；其他配置各有 895 个观测值。运行总小时数是这些单次运行时长之和，并不代表活动实际经过的时间，因为各运行可能并发执行。

图 A.16显示了每次智能体运行中实际提供的模型响应数量，其中 `deepseek-v4-flash`的中位数为 30，`gemini-3.7-flash`为 85，`gpt-5.6-sol`为 17。这些计数包括分析轮次和摘要响应，它们共享每次运行 100 次响应的预算。它们与工具调用计数不同，因为一次模型响应可能请求多个工具，也可能不请求任何工具。

A.6 不同计算内核和目标上的纯源代码准确率

图 A.17提供了仅源代码版本，与内核、变体和 GPU–运行时细分相对应。每个已填充单元格代表一个包含五次重复的案例，其中包括将未成功的预测计为未命中。对于 `gpt-5.6`，在 1.20× 容差范围内，RAI 准确率在 **O**上为 14%，在 **CI**上为 21%，在 **R**上为 15%，在 **RD**上为 5%。相应的 `ZS-ISA`准确率分别为 32%、30%、34% 和 32%。这些比较表明结果对所提供的证据和代码变体具有敏感性，但无法确定模型是否在训练期间遇到过原始内核。

A.7 操作计数约定

预测目标是分析器派生计数器，其操作计数可能与源代码中的数学运算计数不同。我们的 AMD 参考值遵循 VALU 指令计数器加权：加法、乘法和超越函数指令对每个工作项贡献一次操作，而融合乘加贡献两次，且不对打包组件施加额外因子（Advanced MicroDevices, Inc., 2026年）。NVIDIA FP16 参考值计数可以

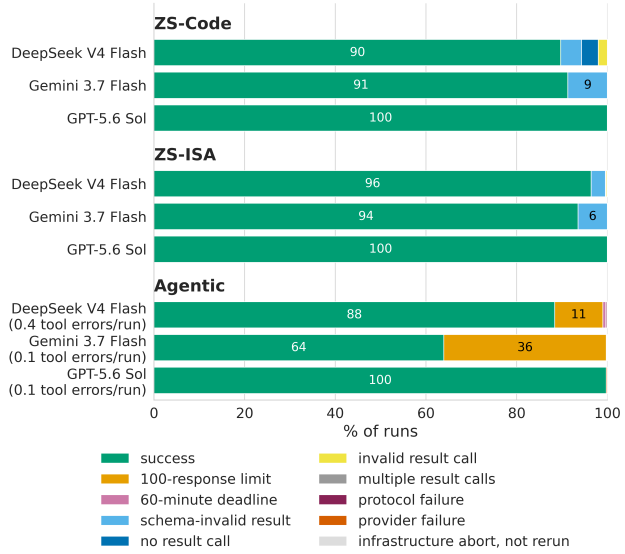


图 A.13：按模型和提示策略划分的全部 895 次运行结果。成功表示获得了可用的结构化预测，不论其数值准确率如何；其余类别则区分终止失败和结果格式失败。

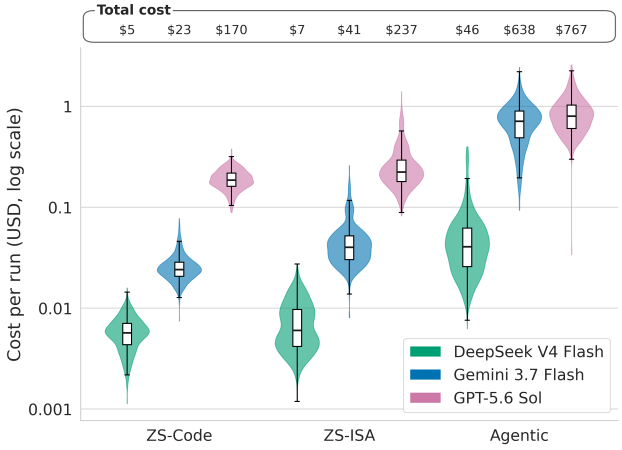


图 A.14：每次计划运行的记录成本（包括未成功的运行），采用对数坐标轴。箱线图显示中位数和四分位距；标题显示每种配置下 895 次运行的记录成本总和。未报告的费用不予计入，因此受影响的数值是下限。

也包括编译器生成的、用于物化常量的 HFMA2 指令，即使源代码中没有 FP16 算术。这些贡献项会保留在参考目标中。

表 A.5协调了 `gpt-5.6`的 `ZS-ISA` FLOP 预**测结果，针对 O** 和 inversek2j 案例，在 MI300X/OpenMP 上进行了核对。参考值中的每个工作项总计 173 个操作，其中计算主体包含 171 个，另有两个设置操作：一个倒数和一个乘法。模型则统计了九个打包 FMA、五个打包乘法和四个打包加法的两个组成部分，相对于参考值的权重增加了 $9\times2+5+4=27$ 个操作。此外，它还将 12 条 `v div scale` 和六条`vdiv fixup` 指令计为参考算术类别之外的18 个额外操作。因此，其计算主体估算为 $171+27+18=216$，加上设置操作后，每个工作项共计 218 个操作，恰好与其对 32,768 个工作项预测的 7,143,424 浮点运算次数总数一致。

表 A.5：gpt-5.6 在 MI300X/OpenMP 上对 inversek2j 的 ZS-ISA **O** 和 **R**预测的操作计数对账，按工作项计。

贡献项	操作
参考总数（包括设置）	173
额外打包权重	27
除法辅助指令	18
预测总数（包括设置开销）	218

这种核算不匹配解释了这些案例中对 $218/173\approx1.26\times$浮点运算次数的高估。预测总流量约为 $0.933\times$ 参考值，因此合并后的 RAI 高估约为 $1.35\times$。**RD** FP64 计算主体缺少相应的打包 FP32 算术；其较小的浮点运算次数差异和更准确的流量估算，使 RAI 处于 $1.20\times$容差范围内。这一协调结果仅适用于经过核查的模型、策略和变体，并不意味着 MI300X 误差都源于同一约定。

A.8 性能分析参考值的变异性

参考浮点运算次数和字节计数器的数值，是针对 179 个案例分别进行 10 次性能分析试验所得的未舍入均值。参考 RAI 为浮点运算计数器均值之和除以读计数器均值与写计数器均值之和。表 A.6汇总了参考表征中记录的每个案例 10 次单独试验所得 RAI 值的变异系数（CoV）。每次试验的 RAI 均使用该次试验的总浮点运算次数和字节总量；CoV 为样本标准差除以均值，以百分比表示。这描述的是性能分析试验之间的变异，而数值一致性容差比较的是大语言模型估计值与计数器均值的参考比率。单个组件计数器的变异性不一定等同于总流量或 RAI 的变异性。

A.9 组件与分类准确率 acy

图 A.18 区分了各精度浮点运算计数器以及设备内存读写。计数器准确度采用对称相对误差 $2|\hat{x}-x|/(\hat{x}+x)$，当两个量均为零时误差为零；

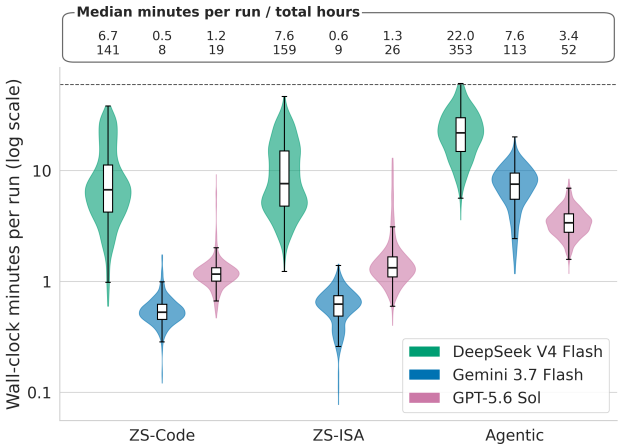


图 A.15：具有记录计时的各次运行中的单次运行预测延迟。小提琴图分布中的箱线图显示中位数和四分位距；标题栏报告中位数和运行时长总和。虚线表示 60 分钟。时长包括智能体沙盒设置和分析，不包括归档与清理。

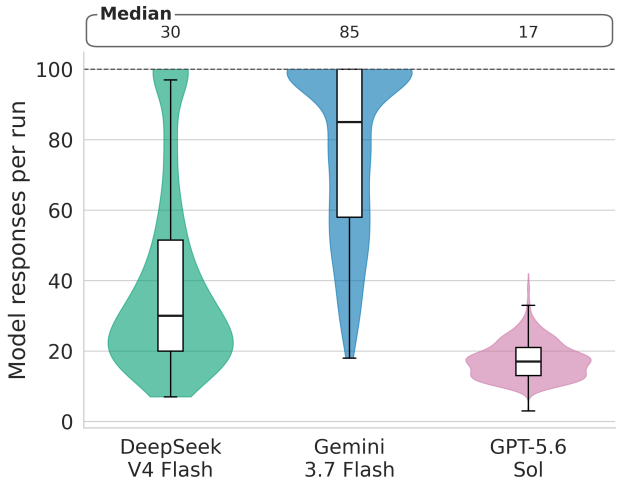


图 A.16：每个模型在全部 895 次运行中，每次智能体运行所交付的模型响应数。箱线图显示中位数和四分位距；虚线表示由分析和摘要共用的 100 条响应预算。

否则，$f\times$ 容差对应于不超过 $2(f-1)/(f+1)$的误差。预测不成功均计为漏检，包括参考计数器为零的情况。表 A.7 报告了两个参考类别，并将未分类运行计入每个真实类别的召回率分母。参考集包含 93 个带宽受限案例和 86 个计算受限案例，每个案例重复 5 次。

表 A.8 展示了仅使用以下信息进行的多数类查找：

图形处理器和占主导地位的参考精度。我们根据最大的参考浮点运算计数器确定精度，因此在四种 GPU 上分别得到 FP32 和 FP64 两组。在每组中，查找会预测出现次数更多的参考类别；若两类持平，则默认预测为带宽受限，这不会改变其样本内准确率。179 个案例中有 131 个分类正确，准确率为 73.2%；相比之下，始终预测为带宽受限的准确率为 52.0%。由于拟合和评分使用相同的标签，因此这只是类别构成的乐观样本内参考值，而非留出预测结果。

作为精度敏感性检查，我们使用每个预测自身最大的浮点运算次数计数器来选择其平衡点，同时保留相同的硬件平衡点、参考标签以及对未分类运行的处理方式。全部 7,122 个可分类预测所选择的主导精度均与其参考值相同；根据预测精度选择平衡点后，全部 8,055 次运行的分类均未发生变化。

A.10 已完成预测中的数值准确率

表 A.9 将所有运行的准确率与每种配置已完成预测中的准确率区分开来。完成意味着获得可用的结构化结果，包括那些其派生 RAI 为非有限值、因而在数值上不准确的预测。对于Gemini，837 个已完成的`ZS-ISA`预测中有 237 个满足 $1.20\times$容差，而 572 个已完成的智能体引导 预测中有 239 个满足该条件：两者分别为 28.3% 和 41.8%。不同策略的已完成子集并不相同，且可能以不同方式加权各个案例，因此这些比率描述的是筛选后返回的预测，而不是在同一批案例上进行配对比较所得的改进。所有运行的准确率仍是主要的端到端指标，未成功运行计为未命中。

A.11 探索性可靠性信号

我们考察预测衍生信号与评估案例本身准确率之间的关联。这些分析属于探索性分析，未在留出案例上进行验证。表 A.10 总结了预测分离度和重复运行间的一致性的分类准确率与覆盖率。对于正的预测强度 I 和已知硬件平衡点 B，预测分离度为 $D(I) = \max(I/B, B/I)$，且无需进行目标内核性能分析。在`ZS-ISA`下，大于 $4\times$ 的预测分离度会在三个模型中选择所有运行的 46–53%，分类准确率为 94–98%。当全部五个`gpt-5.6 ZS-ISA`预测给出相同的有效类别时，其正确率为 98.6%，覆盖 82% 的案例。一致性信号要求每个案例运行五次预测，而预测分离度只需运行一次。

数值一致性提供的保证较弱。对于

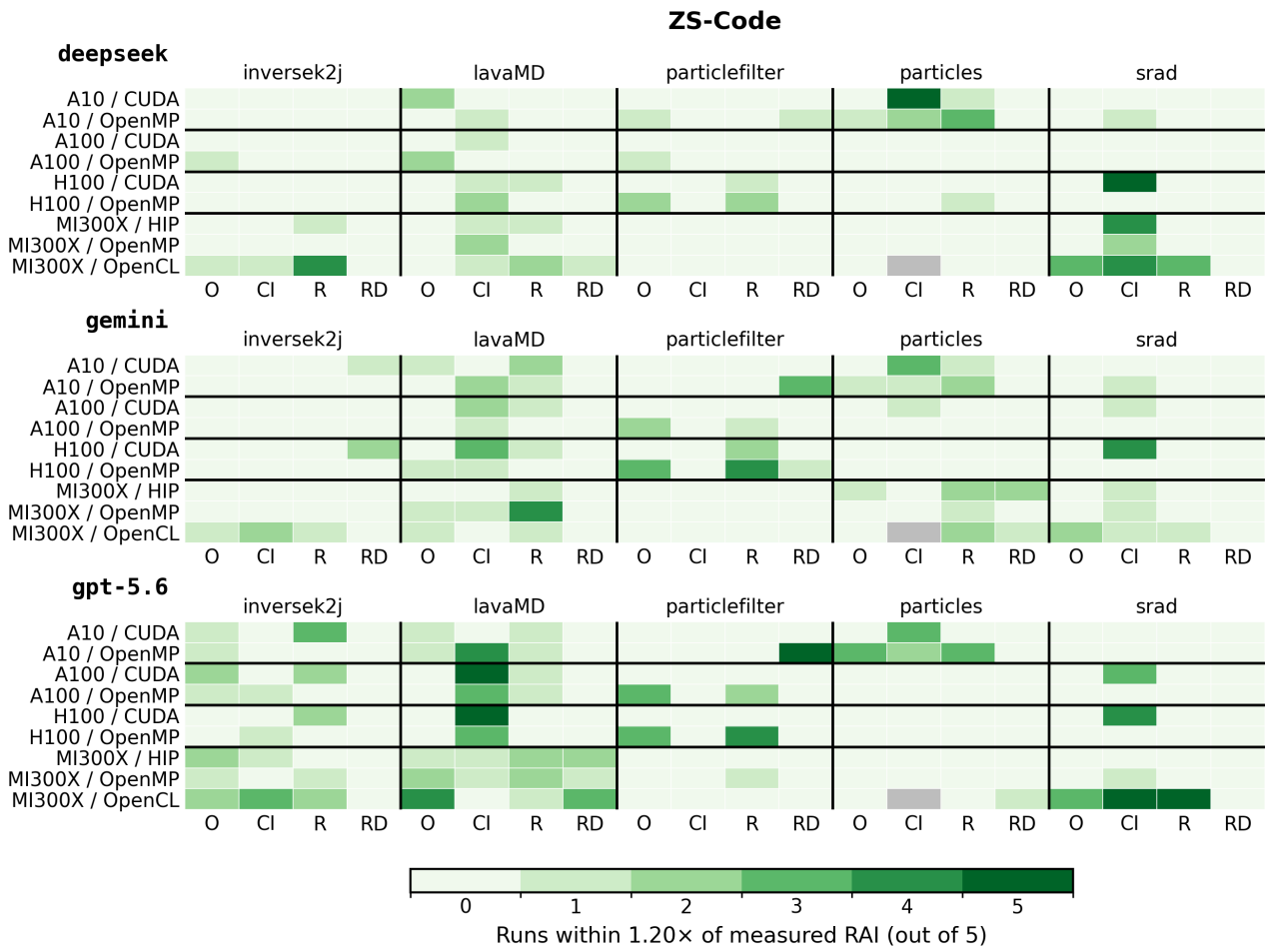


图 A.17：按内核、变体和 GPU–运行时对划分的 ZS-Code 下 RAI 准确率。每个单元格统计五次重复中满足 1.20× 容差的次数；预测失败计为未命中。每个内核中的列依次对应 **O**、**CI**、**R** 和 **RD**。灰色表示 MI300X/OpenCL 上不可用的 particles 的 **CI** 变体。

`gpt-5.6` 在`ZS-ISA`下，179 个案例中有 51 个（28%）包含五个有限且为正的 RAI 估计值，彼此之间的差异均在 1.20×以内。在这些案例的预测中，55% 的结果满足相对于参考值的数值容差，而其余案例中这一比例为 23%。一致性与更高比例的准确估计值相关，但也可能重复同一个错误。表 A.11将五次运行的一致性结果扩展到了全部九种模型—策略配置。数值准确率是指在选定案例中，满足参考值容差的单个预测所占的比例，而不是聚合预测的准确率。对于`gpt-5.6`在 `ZS-ISA`下，51 个数值上一致的案例分为两类：28 个案例的五次预测全部准确，23 个案例的五次预测全部不准确。根据图注中的定义，其他配置中也存在同时包含准确和不准确预测的一致案例。在仅源代码提示下，数值一致性关联性较弱或不存在，因此不应将编译器信息辅助结果视为被解释为一条通用的校准规则。

A.12 工作负载、目标和变体细分

图 A.19保留了每种内核、变体以及图形处理器与运行时的组合，涵盖 `ZS-ISA` 和智能体引导；仅使用源代码的对应结果见 图 A.17。 表 A.12汇总了全部四种变体的数值准确率和可用预测率。对于`gpt-5.6`，在`ZS-Code`下，原始版本与数据类型变体的准确率分别为 13.8% 和 5.3 %；在 `ZS-ISA`下，两者均为 32.4%。DeepSeek的相应差距缩小，而Gemini的差距略有扩大。重命名在不同模型和策略下并未导致准确率出现一致的变化方向。这些比较衡量的是模型对所提供程序的变换的敏感度，并不能确定模型是否在训练期间接触过某个内核。

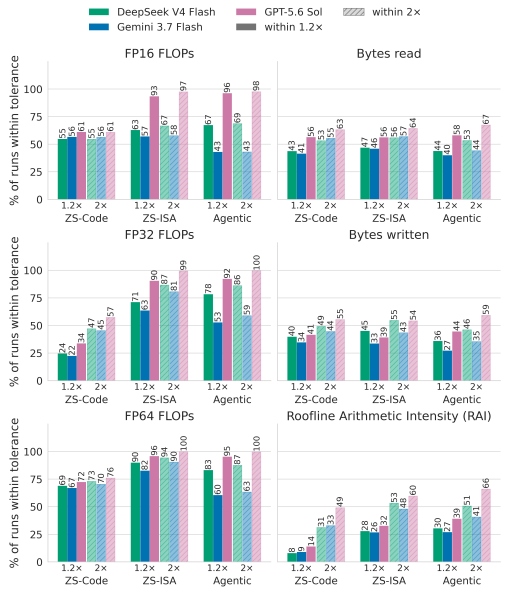


图 A.18：预测的组件计数器（FP16/32/64 浮点运算次数以及读写的设备内存字节数）和推导出的 RAI 的准确率。实心柱表示在 1.20× 容差范围内（为剖面化数值的 1/1.20–1.20）的运行百分比；斜线填充柱使用 2× 容差（0.5–2.0）。预测失败或未完成均计为未命中。

A.13 代表性设备内存流量案例

表 A.13(a) 对比了 NVIDIA 图形处理器上的原始变体 CUDA 实现与 MI300X 上的 HIP 实现。每一行都报告参考总流量、五次重复预测结果的中位数，以及落在容差范围内的单个 RAI 估计值的数量。流量中位数比接近 1，并不意味着每个单独的 RAI 估计值都准确，因为每次运行都有各自的浮点运算次数和流量估计值。

对于 lavaMD，对相邻方框的重复访问可带来复用，但跨越测量边界的流量在不同 GPU 之间差异显著。对于原始 CUDA 案例，剖面化的总流量在 A100/H100 上为 149–152MB，在 A10 上为 290MB。预测值与剖面化数值之比的中位数分别为 0.92–0.93 和 0.67。这种差异与 A100/H100 上更高的有效复用率相符，但无法据此将原因单独归结为缓存容量、调度或替换策略。

对于原始 CUDA particlefilter，40 万个粒子各自生成 69 个整数索引，形成一个 110.4 MB 的临时数组。在固定的循环迭代中，相邻线程访问的条目相距 276 字节。内核先写入、随后读取该数组，因此流量会受到事务粒度和有效复用的影响。预测总流量的中位数约为剖面化结果的 4–26×

在四个 CUDA/HIP 案例中，预测总数均低于剖面化结果，且这些 RAI 估计值无一达到容差要求。粒子案例进一步说明了基于数据占用进行估计的局限：MI300X 的流量为 4.8 MB，而预测中位数仅为 0.8 MB，尽管数据占用很小。仅凭标称容量无法确定哪些请求会越过计数器的内存边界。

A.14 智能体轨迹中的缓存模拟器代码

检查的 `gpt-5.6 AGlavaMD` 轨迹包含针对相邻方框的 Python LRU 缓存模拟器，其中包括一种模拟 MI300X 独立 XCD 本地 L2 缓存的实现。为生成可复现的代码模式摘要，我们标记包含以下内容的工具调用参数：移至末尾、`OrderedDict()`或`popitem(last=False)`。这些模式出现在 180 次`lavaMD` 运行中的 56 次，其中 A10 上 40 次运行中有 15 次，MI300X 上 60 次运行中有 26 次（表 A.13(b)）。模式匹配只能识别候选模拟器代码，不能确认其成功执行；其他实现也可能无法被检测到。该表比较了包含和不包含这些模式的运行中的准确率，但智能体会自行决定何时进行模拟，因此这种比较既不能证明存在因果收益，也不能证明不存在因果收益。 图 A.20 展示了整体工具调用分布。

A.15 准确率差异的不确定性

表 A.14报告了编译器证据带来的数值提升（RQ1）、`gpt-5.6`近边界分类的提升（RQ2），以及沙箱分析带来的数值差异（研究问题4）的配对案例自助法区间。RQ1 和研究问题4 使用全部 179 个案例；RQ2 使用剖析得到的 RAI 位于 2×平衡点范围内的 40 个案例。每次 20,000 次重抽样均采用有放回抽样，抽取案例，并将每个案例在两种对比策略下的五次重复作为整体保留，未成功运行则仍计为未命中。我们计算每次重抽样的平均配对准确率差异，并报告第 2.5 和第 97.5 百分位数（以百分点表示）；随机种子在论文 figures.py中设为零。配对单位是案例，因此重复次数不作为相互独立的案例处理。案例还共享变体、运行时和内核族；这些区间描述的是所评估案例集中的变化，并不能证明结果可泛化到这五个内核族之外。

A.16 对数值容差的敏感性

在更严格的容差下，编译器证据带来的提升以及`gpt-5.6`的智能体优势不只体现在主要 1.20×**目标上**。图A.21 展示了从 1× 到 5× 之间，满足数值容差要求的运行所占比例，涉及 RAI、总浮点运算次数和设备内存流量。表 A.15列出了每个模型在六个选定因子下的 RAI 准确率，以及

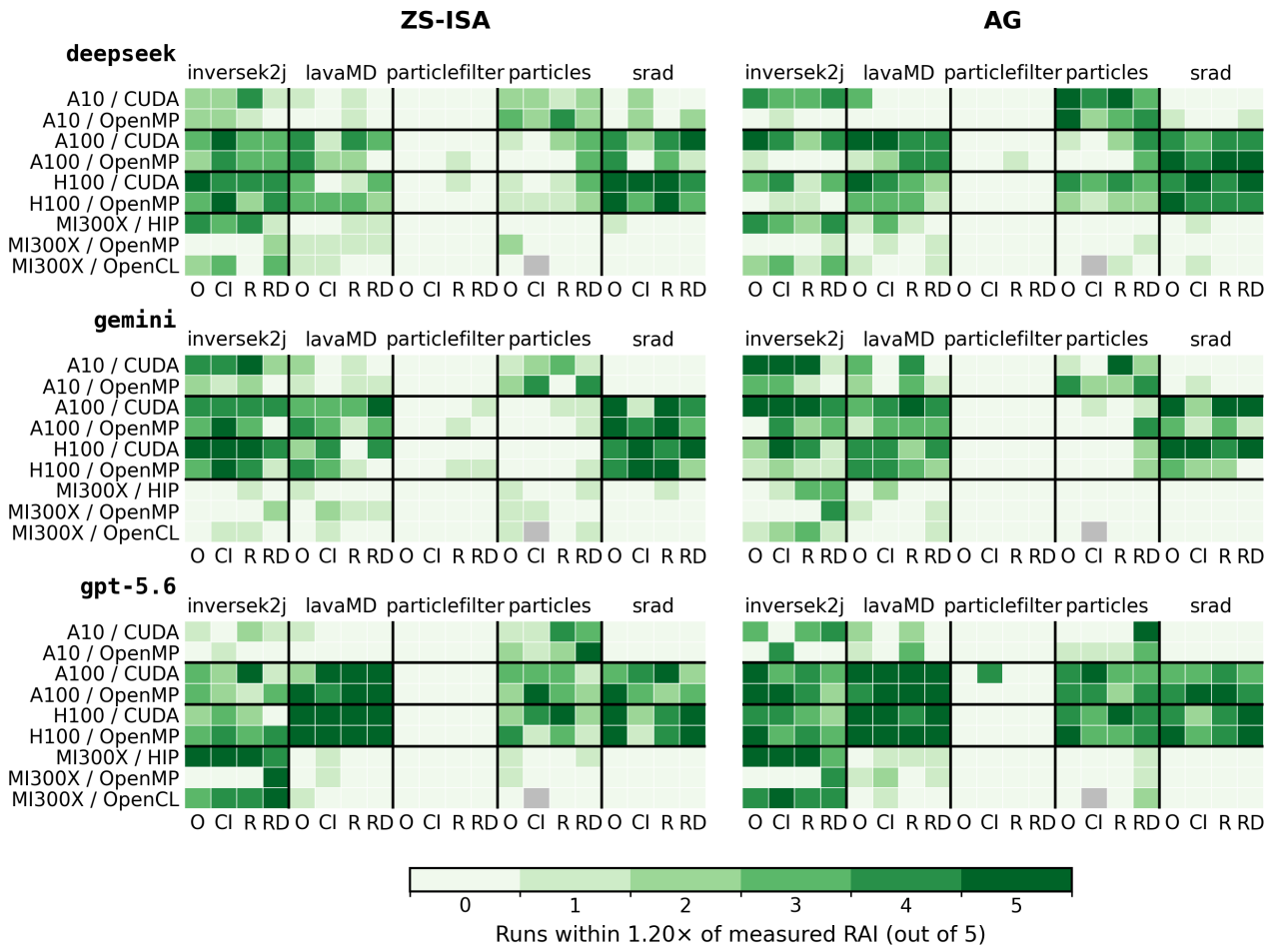


图 A.19：在 ZS-ISA 和智能体引导（AG）下，按内核、变体和 GPU–运行时对划分的 RAI 准确率。每个单元格统计五次重复中有多少次满足 1.20× 容差；未成功的预测计为未命中。每个内核内的列依次为**O**、**CI**、**R**和**RD**。灰色表示 MI300X/OpenCL 上不可用的粒子**CI**变体。

策略。

在 1.10×、1.20×、1.50×和 2×下，对于全部三个模型，ZS-ISA 的数值 RAI 准确率均优于 ZS-Code。对于 gpt-5.6，在整个 1.10×–5× 范围内，观测到的智能体引导（AG）准确率始终高于 ZS-ISA。较宽松的容差缩小了 gpt-5.6 仅基于源代码与结合编译器信息两种方法之间的差异：在 4×下，ZS-Code 的准确率为 77.0%，ZS-ISA 为 76.5%；在 5×下，两者均达到 80.6%。

对于 DeepSeek 和 Gemini，1.20× 处较小的智能体引导差异导致其准确率低于 ZS-ISA 在1.50×、2×、4×和 5×处的准确率，这进一步说明沙箱隔离的益处取决于模型。这里，容差系数用于比较预测值与其经性能分析的数值参考值；而 RQ2 中的距离区间则用于比较剖析得到的 RAI 与硬件平衡点。

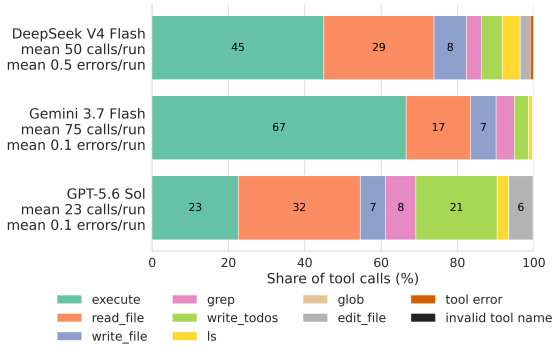


图 A.20：所有运行和重复中记录的智能体工具调用分布，包括未成功的运行。标签显示每次运行的平均调用次数和错误数；工具错误与无效工具名称分别归为不同类别。

表 A.3：所选目标内核、实现来源、测得的 Roofline 行为，以及进行这些测量所用的 GPU。

(a) 标识、来源和操作。

程序	按应用程序接口划分的目标内核	上游套件	实现源代码 CUDA/HIP/OMP	实现源代码 OpenCL	操作
inversek2j	CUDA/HIP/OpenCL：逆运动学内核	AxBench	HeCBench	研究移植	两关节逆运动学
lavaMD	CUDA/HIP：分子动力学 OpenCL：GPU OpenCL 内核	Rodinia	HeCBench	Rodinia 供体	邻域盒分子力
particlefilter	CUDA/HIP：核似然 OpenCL：似然核	Rodinia	HeCBench	Rodinia 供体	粒子似然更新
粒子	CUDA/HIP/OpenCL：collideK	NVIDIA CUDA 样本	HeCBench	研究移植	粒子碰撞响应
srad	CUDA/HIP：srad OpenCL：srad 内核	Rodinia	HeCBench	Rodinia 供体	各向异性扩散系数

（b）测得的受限类别特征：计算受限，■ 带宽受限，并且–未收集。

GPU	API	
A10	CUDA	○○■○ ■■■■ ○○○○ ○○■○ ○○■○
	OMP	○○■○ ■■■■ ○○■○ ○○■○ ○○■○
A100	CUDA	■■■■ ■■■■ ○○○○ ■■■■ ■■■○
	OMP	■■■■ ■■■■ ○○○○ ■■■■ ■■■■
H100	CUDA	■■■○ ■■■■ ○○○○ ■■■○ ■■■○
	OMP	■■■○ ■■■■ ○○○○ ○○■○ ■■■○
MI300X	OMP	○○○○ ■■■■ ○○○○ ○○○○ ○○○○
	HIP	○○○○ ■■■■ ○○○○ ○○○○ ○○○○
	OpenCL	■■■■ ■■■■

单元格内顺序为原始版本、仅重命名（R）、重命名加数据类型（RD）以及编译器加输入（CI）。每个分支均根据其主导精度对应的目标 GPU 脊点进行分类。

（c）已分析的 GPU 及用于带宽受限/CB 分类的 DRAM 级平衡点。峰值吞吐量以 TFLOP/s 为单位；平衡点（BP）以 FLOP/字节为单位。

	A10	A100	H100	MI300X
架构	GA102	GA100	GH100	CDNA3
计算能力上限。	8.6	8.0	9.0	gfx942
显存（GB）	24	40	80	192
带宽（GB/s）	600.1	1,555.2	3,352.3	4,208.7
末级缓存（MiB）	6	40	50	256
FP16 峰值	16.31	60.55	107.46	39.09
FP32 峰值	16.31	15.14	53.73	128.15
FP64 峰值	0.25	7.57	26.86	104.97
FP16 带宽-性能	27.18	38.93	32.06	9.29
FP32 带宽-性能	27.18	9.73	16.03	30.45
FP64 带宽-性能	0.42	4.87	8.01	24.94

性能上限数据来自供应商分析器：NVIDIA 设备属性由 Nsight Compute 提供，MI300X 则使用 ROCm Compute Profiler 独立的 Roofline 微基准测试。MI300X 列出的是其 ISA 目标，而非计算能力。

表 A.4：按近脊线占比排序的结构有效五族集合（距平衡点 1.5 倍频程以内的案例）。中位倍频程是中位 $|\log_2(RAI/\text{平衡点})|$；平衡度为 $\min(CB, BB)$ 除以案例数量；冲突统计工作形态和内存访问模式相同的成员对。选定集合以粗体显示。

五族集合	单元格	近脊线占比（%）	中位倍频程	平衡	冲突
inversek2j、lavaMD、particlefilter、粒子、srad	134	31	3.12	0.485	0
inversek2j、lavaMD、lud、particlefilter、粒子	134	27	3.11	0.418	1
inversek2j、lavaMD、lud、particlefilter、srad	135	25	3.12	0.415	1
lavaMD、lud、particlefilter、粒子、srad	134	25	3.14	0.388	1

表 A.6：每个案例进行十次试验时剖析得到的 RAI 的变化。Median 和 Maximum 汇总了每个图形处理器组内各案例的 CoV 值；All 汇总了全部 179 个案例。这些数据反映的是性能分析的变异性，而不是大语言模型准确率的置信区间。

		RAI CoV（%）	
GPU	案例	中位数	最大值
A10	40	0.278	1.944
A100	40	0.018	0.229
H100	40	0.075	10.257
MI300X	59	0.056	0.579
All	179	0.076	10.257

表 A.7：每个模型和策略的 BB/CB 分类指标（%；每项 895 次运行；93 个带宽受限案例和 86 个 CB 案例）。未分类运行（Uncl.：没有可用预测或 RAI 为非有限值）在准确率以及真实类别的召回率计算中均计为误分类；精度根据预测为相应类别的运行计算。Bal.：平衡准确率，即两类召回率的平均值。

			BB		CB	
配置	准确率	平衡准确率	精确率	召回率	精确率	召回率	未分类率
DeepSeekZS-Code	65.4	65.3	74.3	65.8	76.0	64.9	13.0
Gemini ZS-Code	65.4	65.3	73.4	66.0	78.5	64.7	13.7
gpt-5.6 ZS-Code	80.0	79.5	75.4	91.2	89.3	67.9	0.7
DeepSeek ZS-ISA	83.9	84.0	91.0	82.6	88.6	85.3	6.6
Gemini ZS-ISA	78.3	78.5	88.0	74.4	92.0	82.6	13.0
GPT-5.6 ZS-ISA	91.8	91.8	92.2	92.0	91.8	91.6	0.2
DeepSeek AG	75.3	75.6	90.6	68.2	87.7	83.0	15.4
Gemini AG	53.9	54.3	91.9	43.9	91.1	64.7	41.1
GPT-5.6 AG	91.8	92.0	96.0	88.0	89.0	96.0	0.6

表 A.8：在相同的 179 个案例上拟合并评估的 GPU–精度多数查找表。BB 和 CB 列给出参考案例计数；Guess 选择该行的多数类。准确率对每个案例赋予相同权重，相当于每个案例重复五次；All 汇总八组数据。

GPU	精度	BB	CB	猜测	准确率（%）
A10	FP32	24	6	BB	80.0
A10	FP64	1	9	CB	90.0
A100	FP32	7	23	CB	76.7
A100	FP64	2	8	CB	80.0
H100	FP32	14	16	CB	53.3
H100	FP64	2	8	CB	80.0
MI300X	FP32	30	12	带宽受限	71.4
MI300X	FP64	13	4	BB	76.5
All		93	86	按组	73.2

表 A.9：数值 RAI 准确度，涵盖 1.20× 所有运行以及已完成的预测（每种配置运行 895 次）。Complete 和 Accurate 分别给出可用结果的数量，以及达到容差要求的估计数量。在“准确率”下，All 使用 895 作为分母，Complete 使用可用结果数量作为分母；已完成的子集因策略而异。

				准确率 (%)
模型	策略	完成 (n)	准确 (n)	全部完成
DeepSeek ZS-ISA		863	248	27.7 28.7
DeepSeek AG		791	270	30.2 34.1
Gemini	ZS-ISA	837	237	26.5 28.3
Gemini	AG	572	239	26.7 41.8
gpt-5.6	ZS-ISA	895	290	32.4 32.4
gpt-5.6	AG	893	348	38.9 39.0

表 A.10：探索性可靠性信号：按预测分离度与平衡点的距离（上图）及五次运行一致性（下图）划分的单次运行分类准确率。

单次运行：与平衡点的预测距离

配置	$\leq 2\times$	$(2,\ 4]\times$	$> 4\times$	All
DeepSeek ZS-ISA	80 (26)	93 (16)	94(52)	84
Gemini ZS-ISA	77 (26)	89 (15)	98(46)	78
GPT-5.6 ZS-ISA	79 (28)	95 (19)	98(53)	92
GPT-5.6 AG	81 (28)	96 (19)	97(52)	92

五次运行：重复运行间的一致性

配置	同一类别正确案例	1.20× 内的 RAI 案例	准确
DeepSeek ZS-ISA	110/179	97.3 12/179	60 / 25
Gemini ZS-ISA	105/179	93.3 27/179	52 / 22
GPT-5.6 ZS-ISA	147/179	98.6 51/179	55 / 23
GPT-5.6 AG	149/179	98.0 59/179	56 / 30

顶部：分类准确率（%），括号内为该区间占全部 895 次运行的比例。区间依据预测的 RAI 划分；失败或非有限预测值不属于任何区间，因此各比例之和不一定为 100%。全部：所有运行的准确率，将未成功运行计为错误。底部：五次预测一致的案例（共 179 个）：五次均给出相同的有效类别（同一类别），或五次 RAI 估计值均为有限正数，且最大值/最小值 ≤ 1.20（*RAI* 在 1.20×以内）。正确：一致案例中的分类准确率。准确：在一致案例/其余案例中，预测值与参考值相差在 1.20×以内的比例。

表 A.11：每个模型和策略的五次运行一致性；案例数均为 179 个。相同类别：五次预测均给出相同的有效 BB/CB类别；正确：这些预测的分类准确率。RAI 在 1.20×以内：五次 RAI 估计值均为有限正数，且最大值/最小值≤ 1.20；准确率：在一致案例/其余案例中，预测值与参考值相差在 1.20× 以内的比例。区间内/区间外：一致的 RAI 案例中，五次预测均在参考值1.20×以内，或均在其外；其余一致案例包含同时处于容差范围内和范围外的预测值。

配置	同一类别 案例	正确案例	1.20×内的 RAI 准确率	输入	输出
DeepSeek ZS-Code	104	77.9	20	9 / 8	1 18
Gemini ZS-Code	95	77.9	10	2 / 9	0 9
gpt-5.6 ZS-Code	144	84.0	43	15 / 13	5 34
DeepSeek ZS-ISA	110	97.3	12	60/25	7 4
Gemini ZS-ISA	105	93.3	27	52/22	12 11
GPT-5.6 ZS-ISA	147	98.6	51	55 / 23	28 23
DeepSeek AG	92	98.9	13	91 / 25	11 1
Gemini AG	26	100.0	21	71 / 21	15 6
GPT-5.6 AG	149	98.0	59	56 / 30	30 24

表 A.12：对于每个模型和策略，按变体列出的 1.20× 内 RAI 准确率和完成状态（可用预测）。失败的运行在准确率统计中计为未命中；**CI** 的案例较少，因为在 MI300X/OpenCL 上无法使用粒子**CI**。

配置	1.20× 内（%） O	CI	R	RD	完成状态（%） O	CI	R	RD
DeepSeek ZS-Code	6.2	15.0	8.9	0.9	97	94	95	73
Gemini ZS-Code	6.2	11.8	12.4	4.4	90	94	92	89
gpt-5.6 ZS-Code	13.8	20.9	15.1	5.3	100	100	100	100
DeepSeek ZS-ISA	30.7	25.0	28.0	27.1	99	100	99	88
Gemini ZS-ISA	27.6	28.6	26.2	23.6	96	94	93	92
GPT-5.6 ZS-ISA	32.4	30.5	34.2	32.4	100	100	100	100
DeepSeek AG	32.9	29.5	26.7	31.6	92	90	87	85
Gemini AG	24.9	26.4	28.9	26.7	54	75	65	62
gpt-5.6 智能体引导	39.1	39.1	36.9	40.4	100	99	100	100

表 A.13：代表性的 gpt-5.6 流量案例。(a) O变体上的 gpt-5.6 ZS-ISA（NVIDIA 上的 CUDA、MI300X 上的 HIP）：剖面化的设备内存流量、五次运行的预测流量中位数、二者的比值，以及 RAI 准确率在 1.20× 以内的运行次数。MI300X 配备八个 XCD 本地 4 MiB L2 缓存和一个 256 MB Infinity Cache。(b) gpt-5.6 智能体引导 lava MD 运行（所有变体和运行时），其工具调用参数包含 LRU 缓存模拟器代码模式（已检测）；模式匹配并不能确认模拟器确实运行，其他实现也可能无法被检测到。对于检测到/未检测到模式的运行，RAI 准确率在 1.20× 以内属于观察结果，因为智能体会自行决定何时进行模拟。

(*a*) *ZS-ISA*，O 变体：总流量

GPU	L2（MiB）性能 ·	（MB）预测值（MB）	比率	RAI 中
lavaMD				
A10	6	290	195	0.67 1/5
A100	40	152	141	0.93 2/5
H100	50	149	137	0.92 5/5
MI300X	8 × 4	814	547	0.67 0/5
particlefilter				
A10	6	2,198	117	0.05 0/5
A100	40	323	82	0.26 0/5
H100	50	370	71	0.19 0/5
MI300X	8 × 4	3,231	123	0.04 0/5
粒子				
A10	6	0.9	0.8	0.84 1/5
A100	40	0.8	0.8	0.98 3/5
H100	50	0.8	1.0	1.32 2/5
MI300X	8 × 4	4.8	0.8	0.16 1/5

(b) AG lavaMD：*LRU* 模拟器代码

GPU	检测到的 RAI 准确率（%）	检测到/未检测到
A10	15/40	33 / 12
A100	10/40	100 / 97
H100	5/40	100 / 97
MI300X	26/60	12 / 15

表 A.14：评估主要比较结果的准确率差异及配对案例自助法 95% 区间。增益和区间均以百分点表示，正值表示有利于 Difference 中的第一种策略。数值准确率使用 1.20×容差；区间基于 20,000 次重抽样计算，每个案例和策略均保留全部五次重复，包括未成功运行。

模型	差异	案例	增益	95% 区间
RQ1：1.20×内的数值 *RAI*				
DeepSeek ZS-ISA –ZS-Code		179	20.0	$[14.5, 25.5]$
Gemini	ZS-ISA − ZS-Code	179	17.8	$[12.2, 23.6]$
gpt-5.6	ZS-ISA − ZS-Code	179	18.7	$[12.0, 25.4]$
RQ2：分类，剖面化 $D(I) \leq 2$				
gpt-5.6	ZS-ISA − ZS-Code	40	37.0	$[19.5, 54.0]$
研究问题*4*：数值 *RAI within* 1.20×				
DeepSeek AG –ZS-ISA		179	2.5	$[-1.8, 6.6]$
Gemini	智能体引导 − ZS-ISA	179	0.2	$[-3.4, 3.7]$
gpt-5.6	智能体引导 − ZS-ISA	179	6.5	$[3.4, 9.6]$

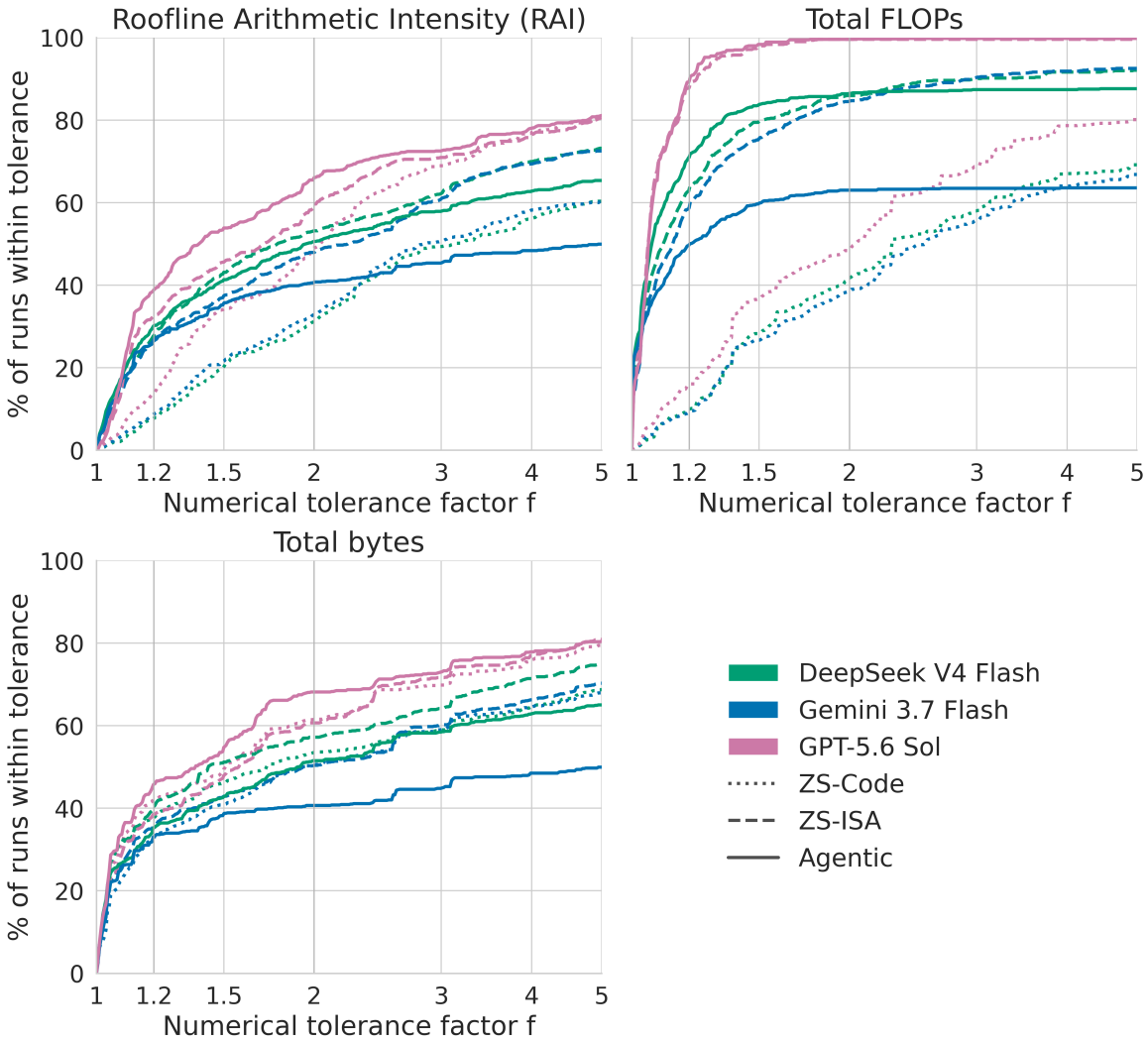


图 A.21：编译器证据可提高较严格容差下的数值准确率，而较宽松的容差则会缩小部分差异。各面板展示了数值容差因子从 1 变化到 5 时，RAI、总浮点运算次数和总设备内存流量的变化，横轴采用对数坐标；曲线越高，表示在该容差下预测越准确。颜色表示模型，线型表示策略；垂直参考线标示 1.20×和 2×。每条曲线均包含每种配置的全部 895 次运行，失败和未完成的预测均计为未命中。

表 A.15：全部九种模型–策略配置在所选数值容差因子下的 RAI 准确率（%）。每项均包含全部 895 次运行，失败和未完成的预测均计为未命中；百分比四舍五入到小数点后一位。

配置	1.10×	1.20×	1.50×	2×	4×	5×
DeepSeek ZS-Code	3.4	7.7	20.2	31.2	56.1	60.3
Gemini ZS-Code	4.5	8.7	21.7	32.8	58.2	60.3
gpt-5.6 ZS-Code	7.5	13.7	34.2	49.1	77.0	80.6
DeepSeek ZS-ISA	18.2	27.7	43.1	53.2	69.9	73.2
Gemini ZS-ISA	17.0	26.5	37.4	47.9	69.4	72.5
GPT-5.6 ZS-ISA	20.7	32.4	45.6	59.7	76.5	80.6
DeepSeek AG	20.3	30.2	41.3	50.6	62.8	65.4
Gemini AG	18.2	26.7	35.6	40.7	48.4	49.9
GPT-5.6 AG	23.9	38.9	53.9	66.0	78.0	81.1