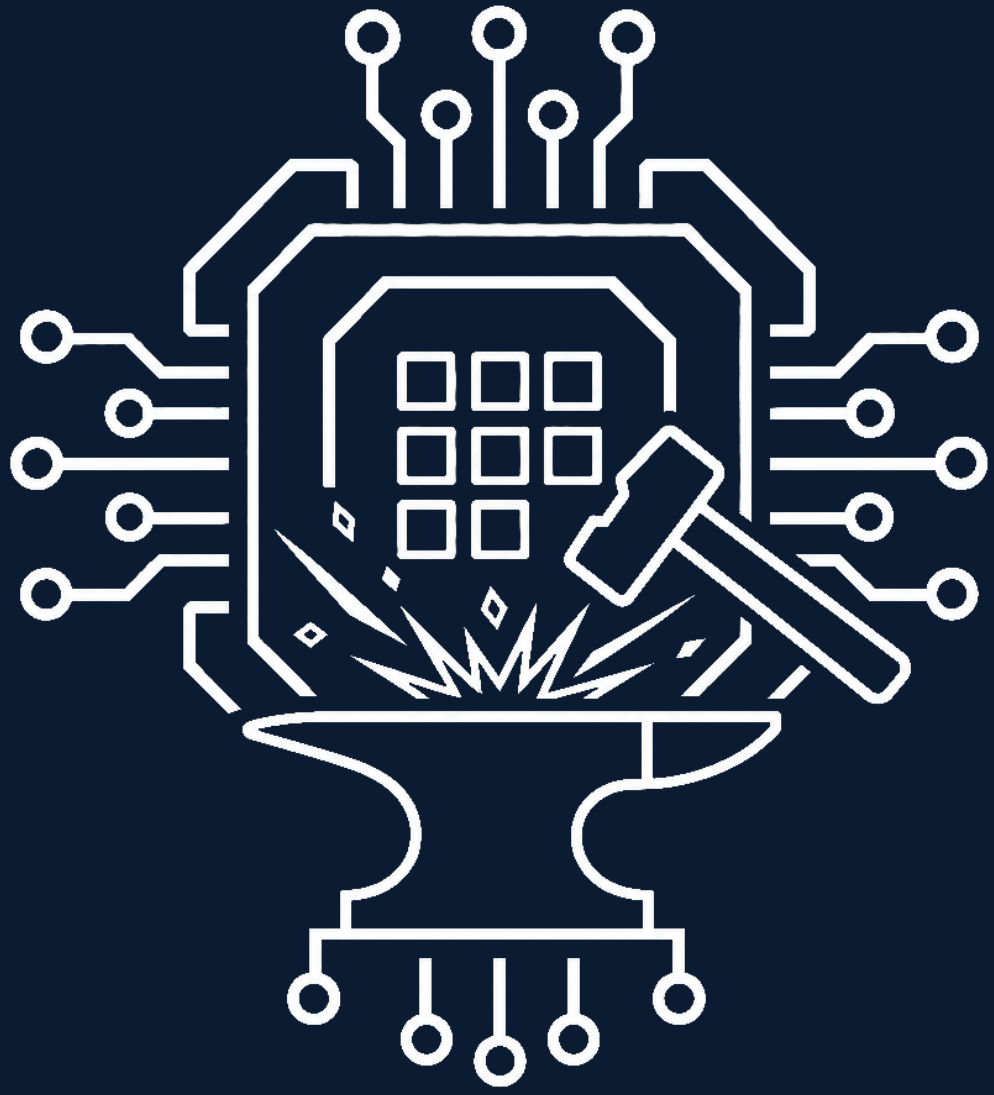
KERNELFORGE

KernelForge

让每一份算力创造更大价值

面向 GPU 算子优化的自主 LLM Agent 框架

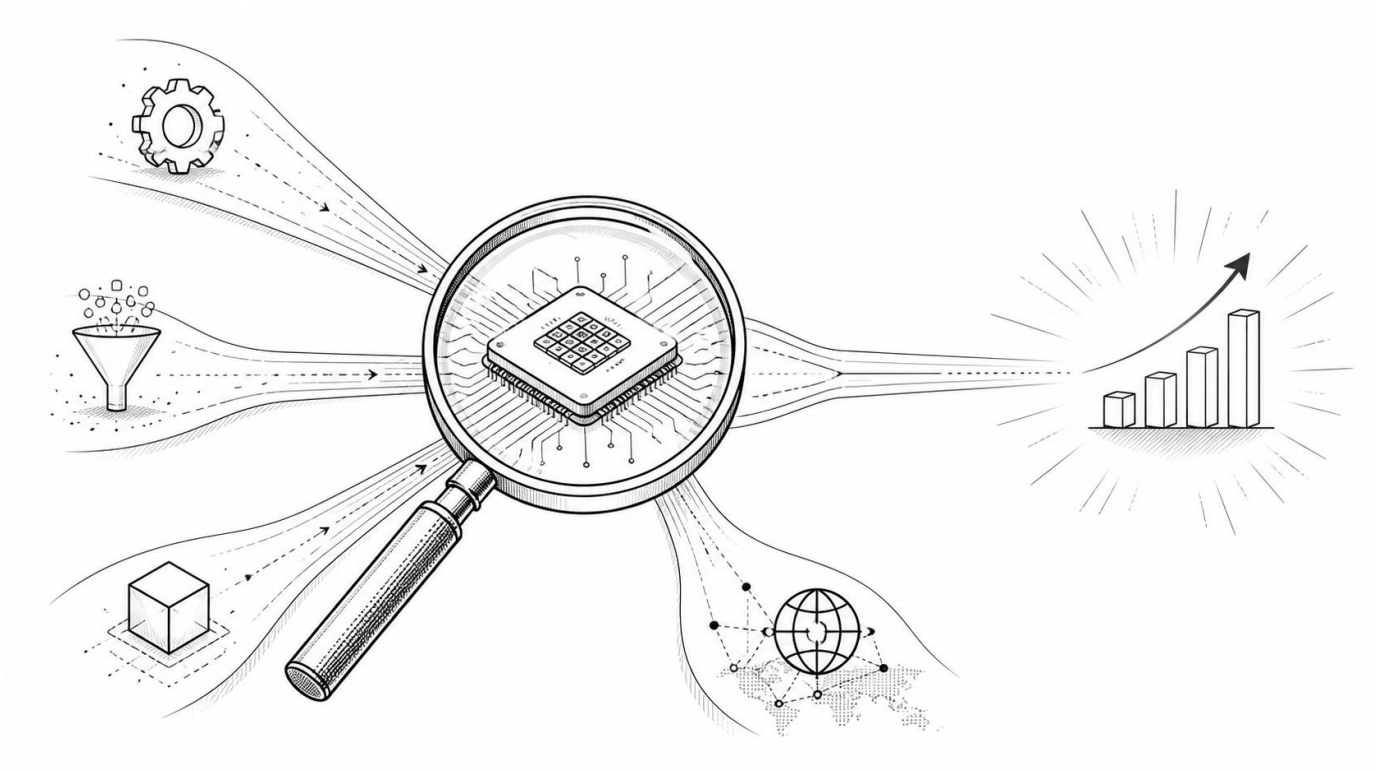
让有限 GPU 支撑更低延迟、更高并发、更低成本的 AI 服务



AI算子优化队

START HERE

今天只讲一个问题：如何让算力产生更大业务价值?



FOUR STEPS TO THE ANSWER

01 **AI 深入业务**

体验、并发与成本成为硬约束

02 **GPU 效率受限**

算力稀缺，通用实现难吃满硬件

03 **KernelForge 闭环**

分析、实现、验证与经验沉淀

04 **落地业务场景**

推理、适配、运维与团队协作

NOW

AI 正在成为各行各业的新型基础能力

从信息获取到内容生产，再到企业运营与科学探索，AI 正在重塑工作和服务的方式。

AI IS MOVING INTO PRODUCTION

智能办公

写作、检索、会议与代码协作

企业服务

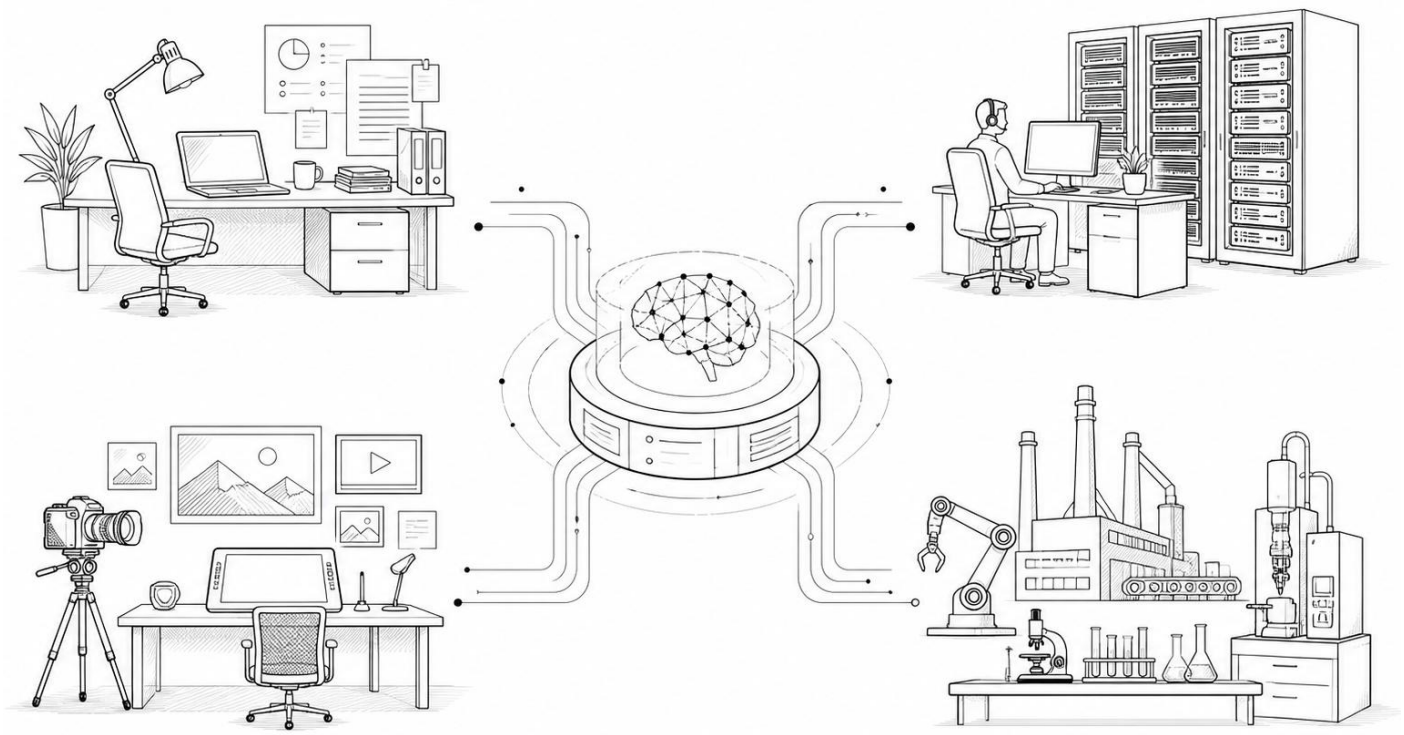
客服、知识库、运营与决策支持

内容创作

图像、视频、设计与多媒体生成

产业研发

制造、医疗、金融与科研分析

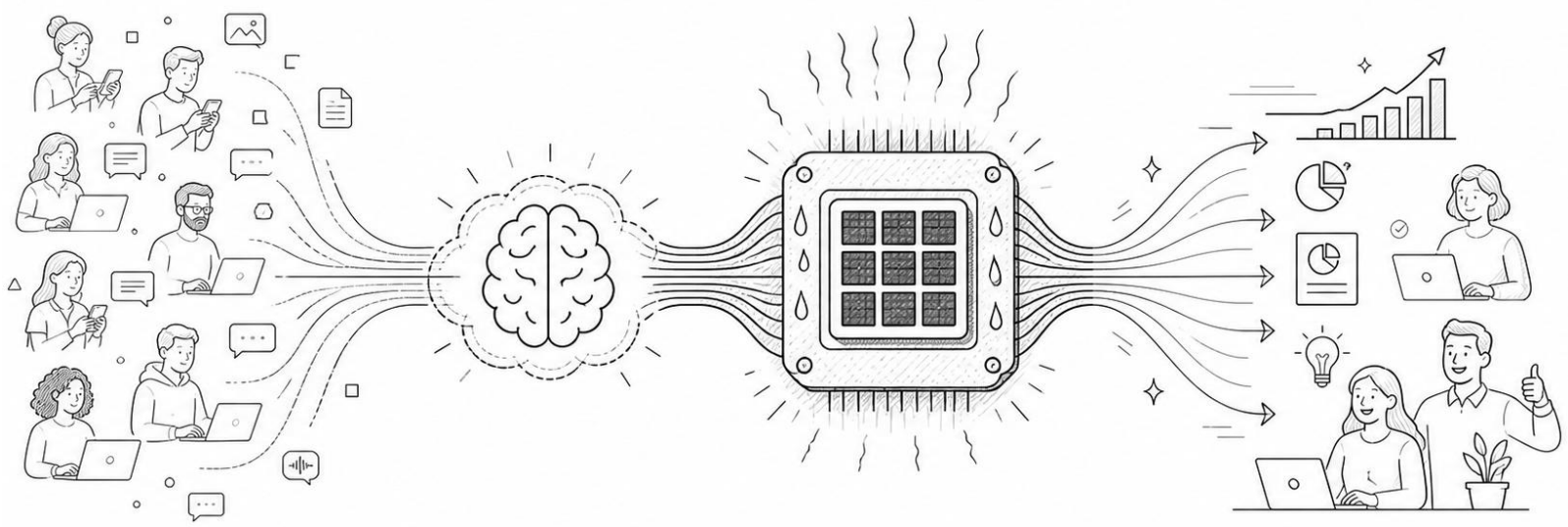


AI 越深入业务，稳定、快速、可负担的计算能力就越成为关键。

WHY IT MATTERS

一次 AI 回答的背后，是持续消耗的 GPU 算力

模型能力走向生产，必须经过“算力效率”这一关。



WHY GPU EFFICIENCY MATTERS

请求持续增长

并发会重复放大每一次底层调用

GPU 成为瓶颈

计算、访存与启动开销共同累积

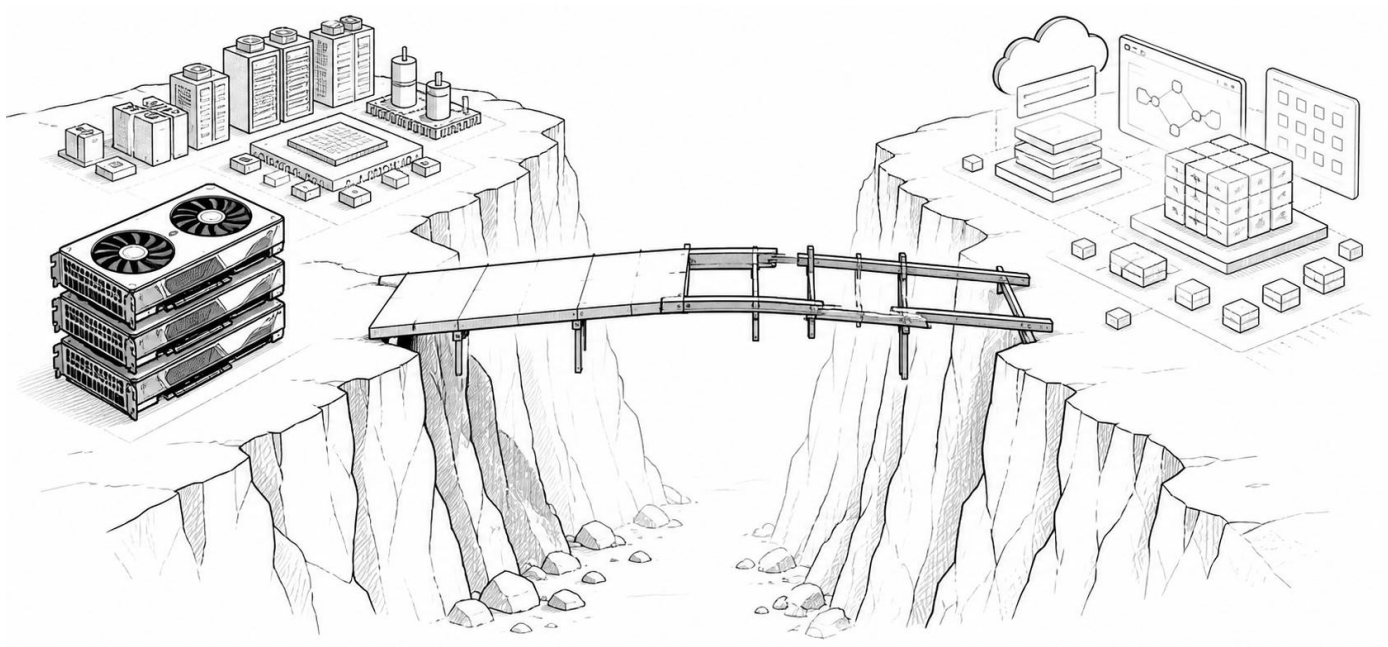
业务指标被牵引

时延、吞吐与单位成本同步变化

同一块 GPU，运行更快 = 服务更多用户 = 消耗更少资源

THE GAP

买到算力，只是第一步；用好算力，才决定实际价值



THREE CONSTRAINTS

现实 01

GPU 资源稀缺

高性能算力昂贵、供给有限，每一份资源都应被充分利用。

现实 02

平台差异显著

硬件架构、编译器、运行时和软件栈共同决定真实性能。

现实 03

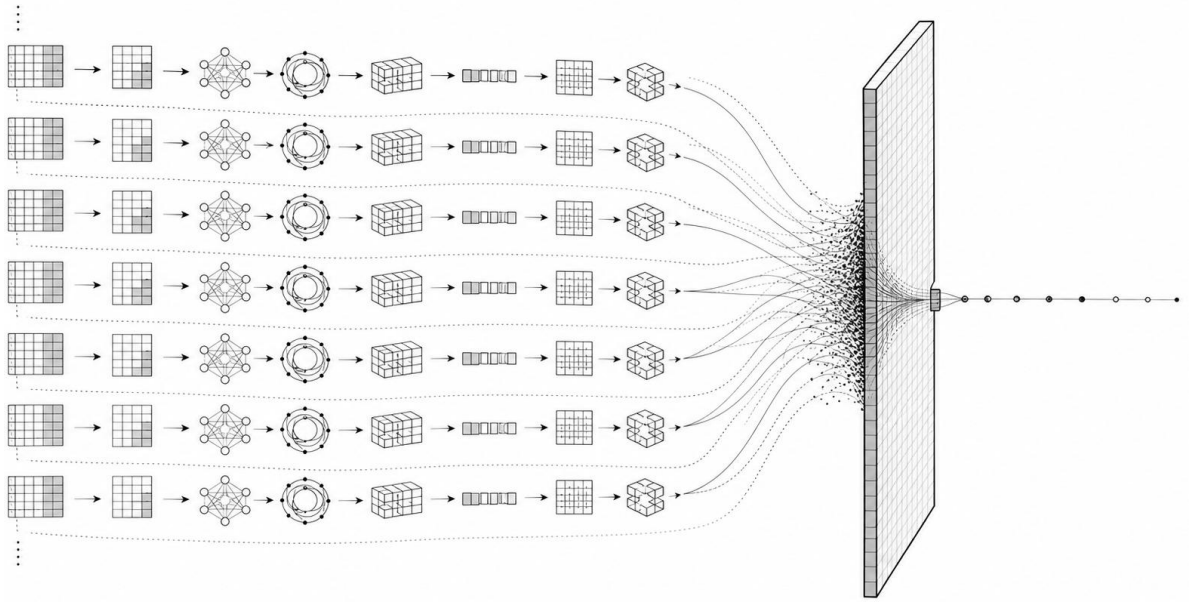
迁移不等于高效

代码能够运行，只代表兼容；持续优化才能释放硬件价值。

硬件能力 ≠ 实际业务效率

LOOK INSIDE

模型的“慢”，往往藏在成千上万个小步骤里



FROM MICRO COST TO SYSTEM BOTTLENECK

STEP 01

单步开销很小

计算、访存、转换或启动，可能只比最优实现慢一点。

STEP 02

模型中反复调用

层数、序列长度与并发请求，会持续放大每一次微小损失。

STEP 03

最终形成业务瓶颈

用户看到的是端到端时延、吞吐下降与单位请求成本。

GPU 算子 = 模型运行时反复执行的基础计算步骤

KernelForge 的任务：找出 AI 生产线上的“隐性堵点”，并持续消除它们。

THE CHALLENGE

GPU 优化不是“改一行代码”，而是一轮轮工程试验

1 找到哪里慢

2 判断为什么慢

3 设计优化方案

4 编写底层实现

5 验证正确性

6 测量性能

WHY THE TRADITIONAL WORKFLOW IS HARD

PAIN 01

依赖专家判断

瓶颈定位、优化目标与实现路径，都需要长期工程经验。

PAIN 02

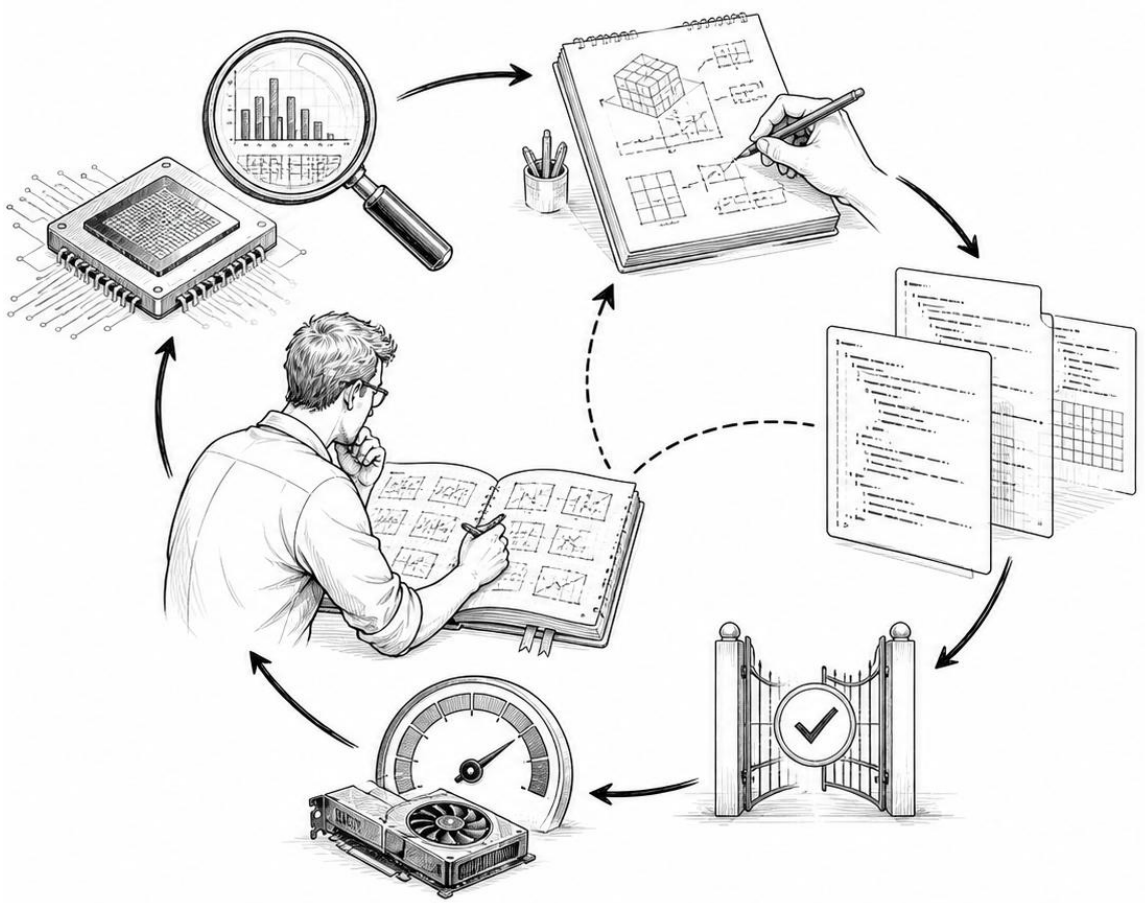
试错成本高

编译、验证、回退和复测，让每次尝试都有完整周期。

PAIN 03

经验难复用

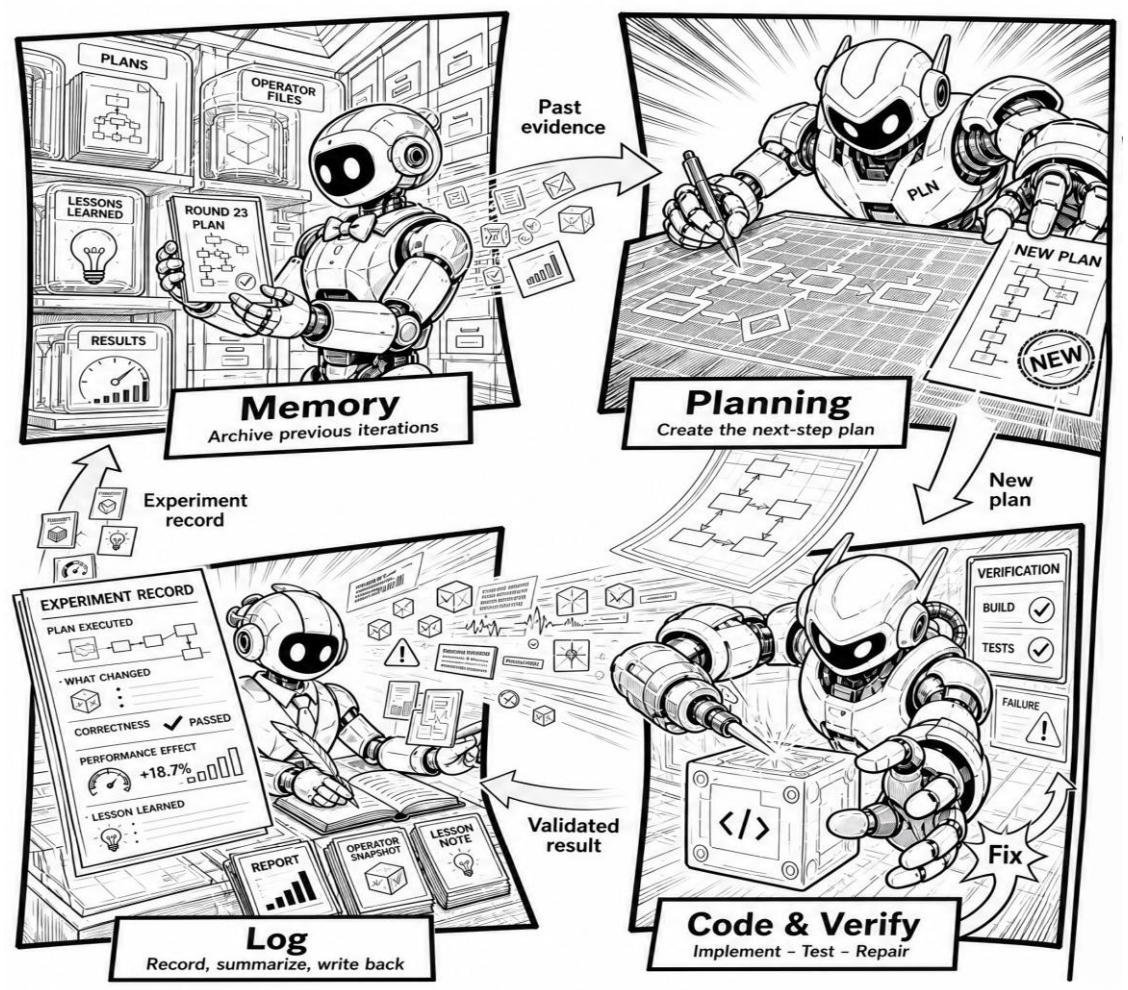
策略和失败教训留在个人记忆里，团队容易重复试错。

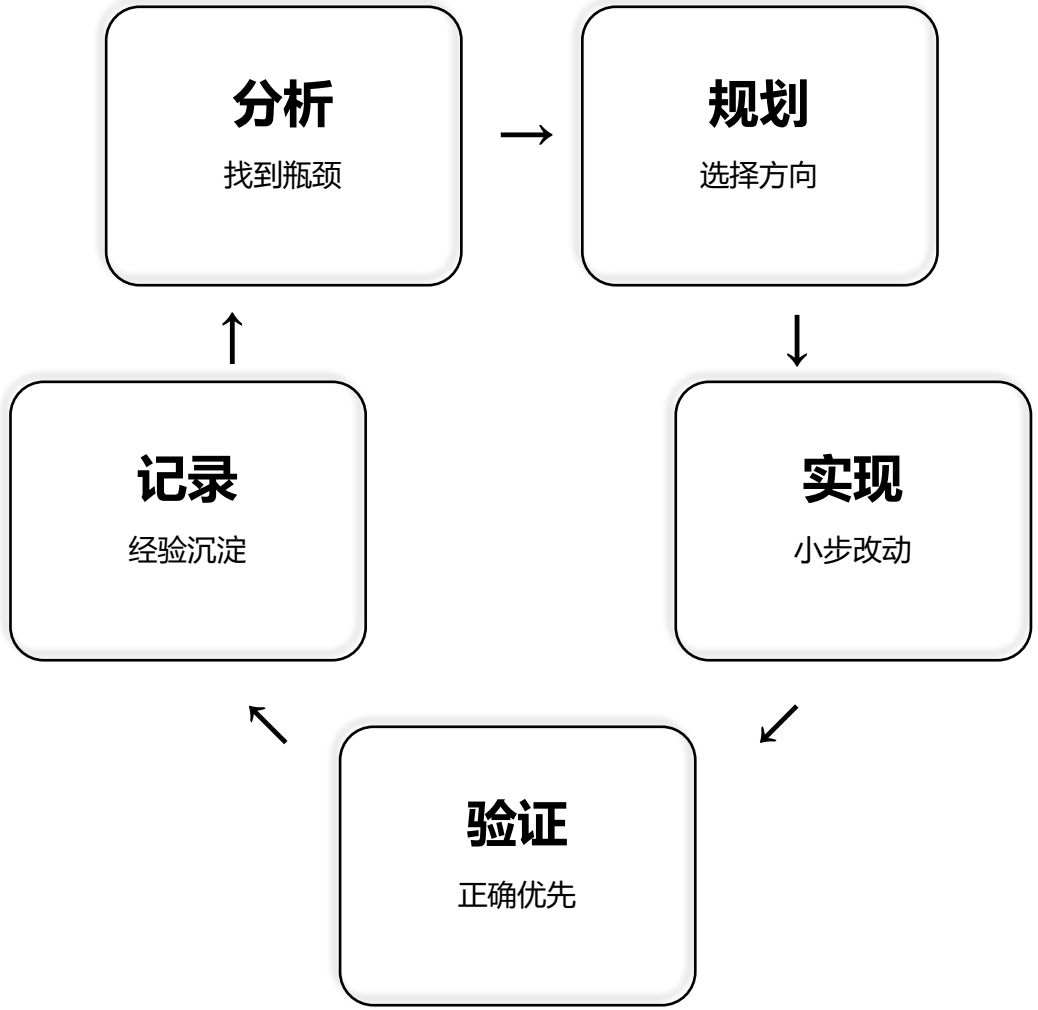


THE ANSWER

KernelForge：让优化从“手艺活”变成可复制的闭环

不是让 AI 猜一个答案，而是让它像工程师一样持续试验、验证与复盘。

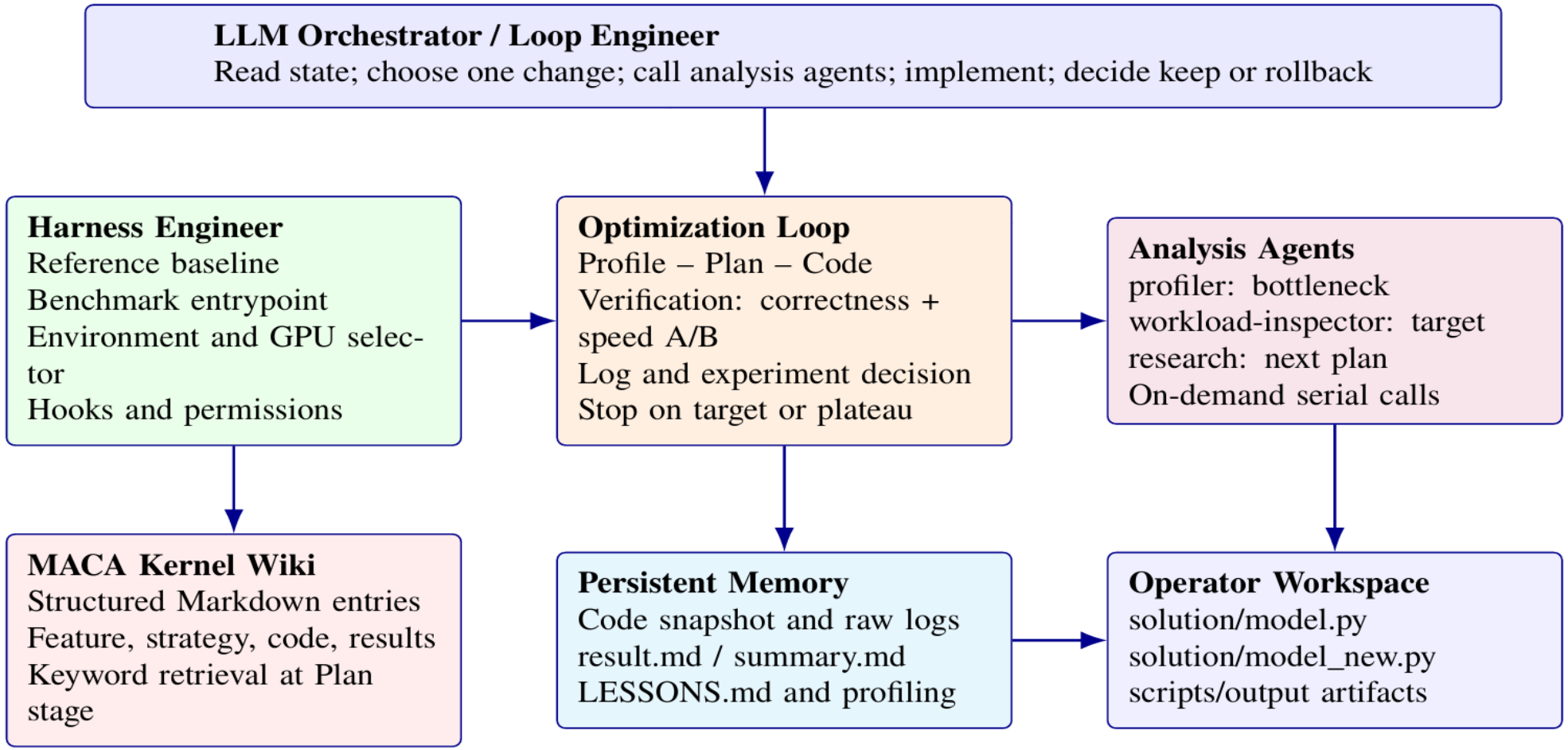




ARCHITECTURE

KernelForge 架构

把基线、执行、分析、记忆和权限解耦



TRUSTWORTHY RESULTS

更快不够：必须结果正确、比较公平、过程可追溯

KEY 01

正确性保障

先确认优化结果与原始结果一致；不正确，速度再快也没有意义

KEY 02

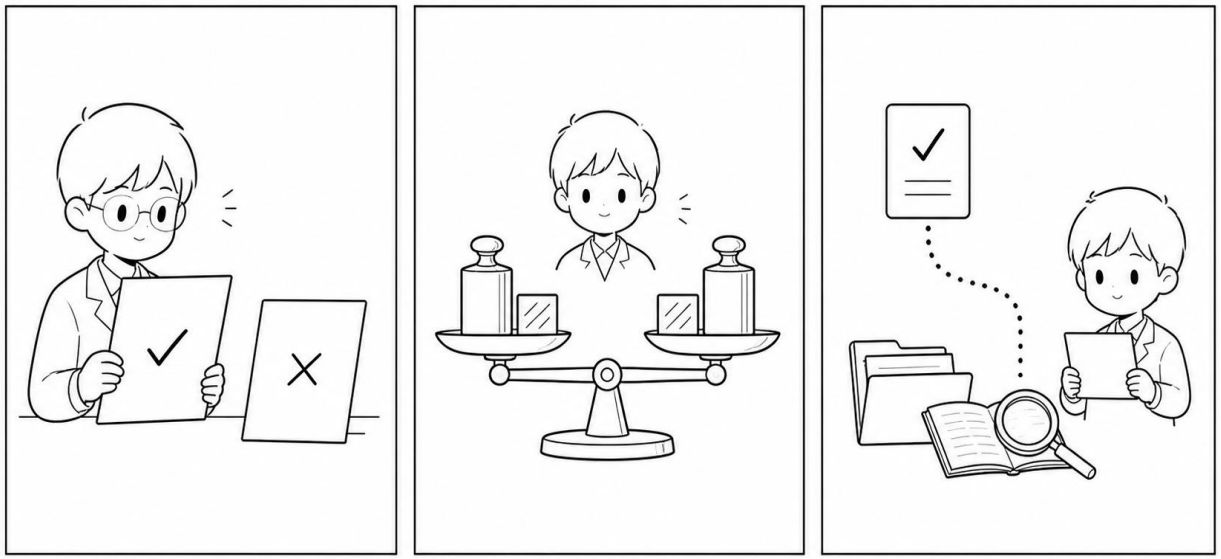
公平评测

固定基线、输入与运行环境；确保收益来自真实优化。

KEY 03

经验可追溯

代码、日志、失败原因和结论全部留档，可复盘、可复用。

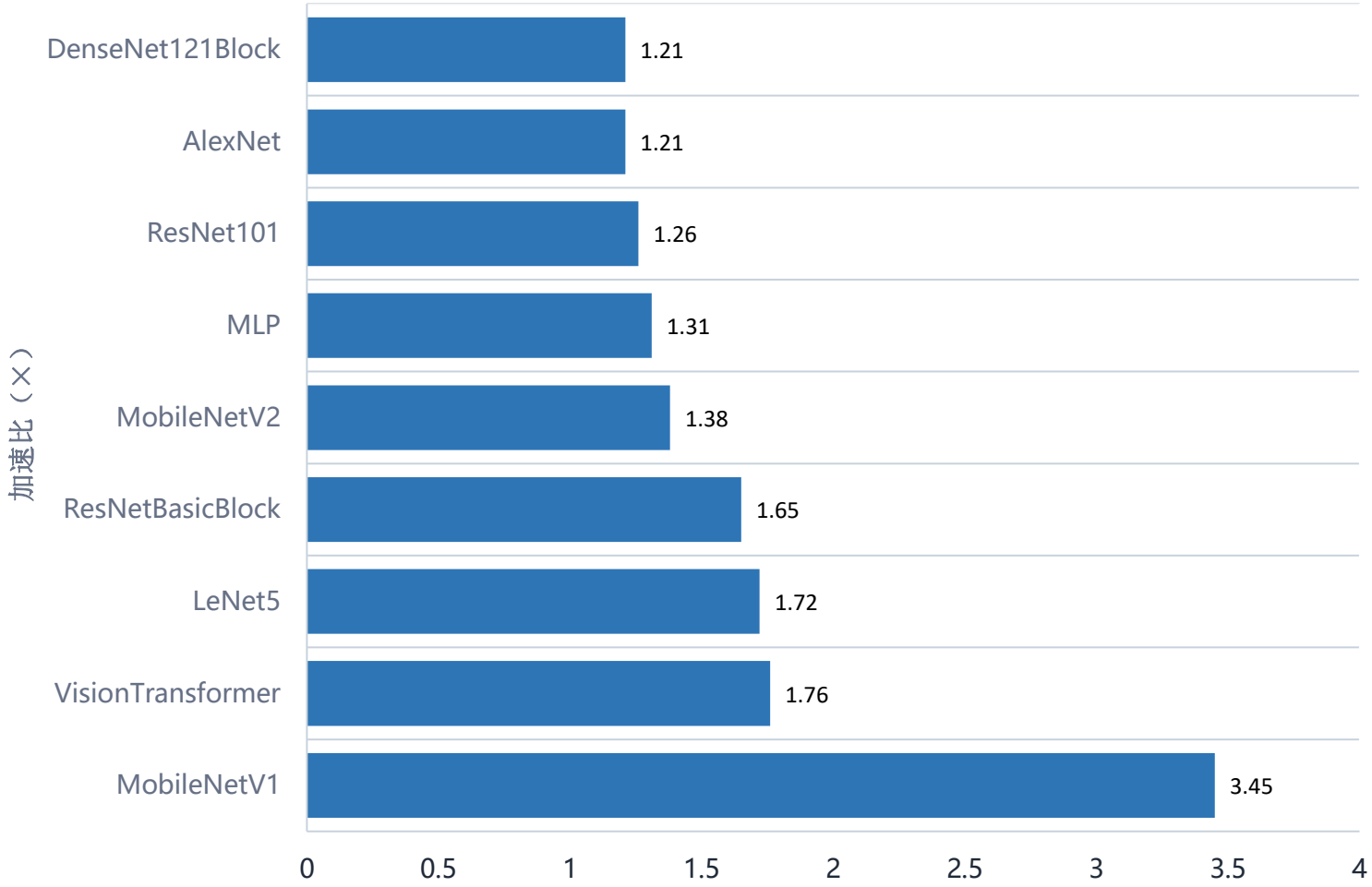


KernelForge 追求的不是“看起来更快”，而是“可信地更快”。

RESULT

KernelBench Level 3

端到端模型性能评测



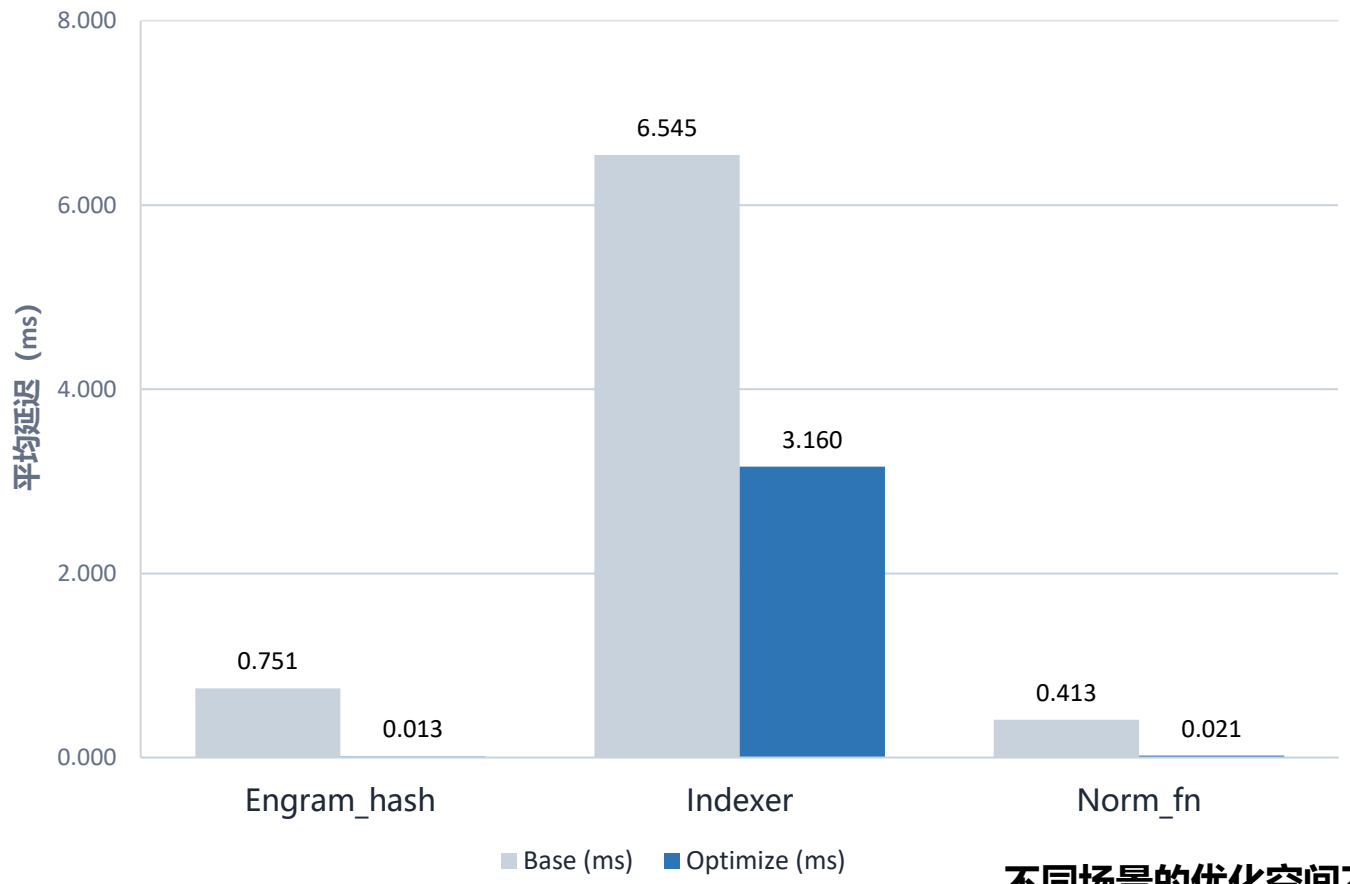
≈ 1.57×

主要观察

- 9 个任务全部超过 1.2×
- MobileNetV1 达到 3.45×
- 收益覆盖多种网络结构

RESULT

DeepSeek 推理算子



加速比

Engram_hash	56.621×
Indexer	2.071×
Norm_fn	19.273×

不同场景的优化空间不同；重要的是系统能够持续寻找、验证并复用性能收益。

APPLICATION 01

生产 LLM 推理服务：更低延迟、更高并发、更低成本

ONLINE INFERENCE

OUTCOME 01

响应更快

热点算子缩短后，智能问答、客服和代码助手更少等待。

OUTCOME 02

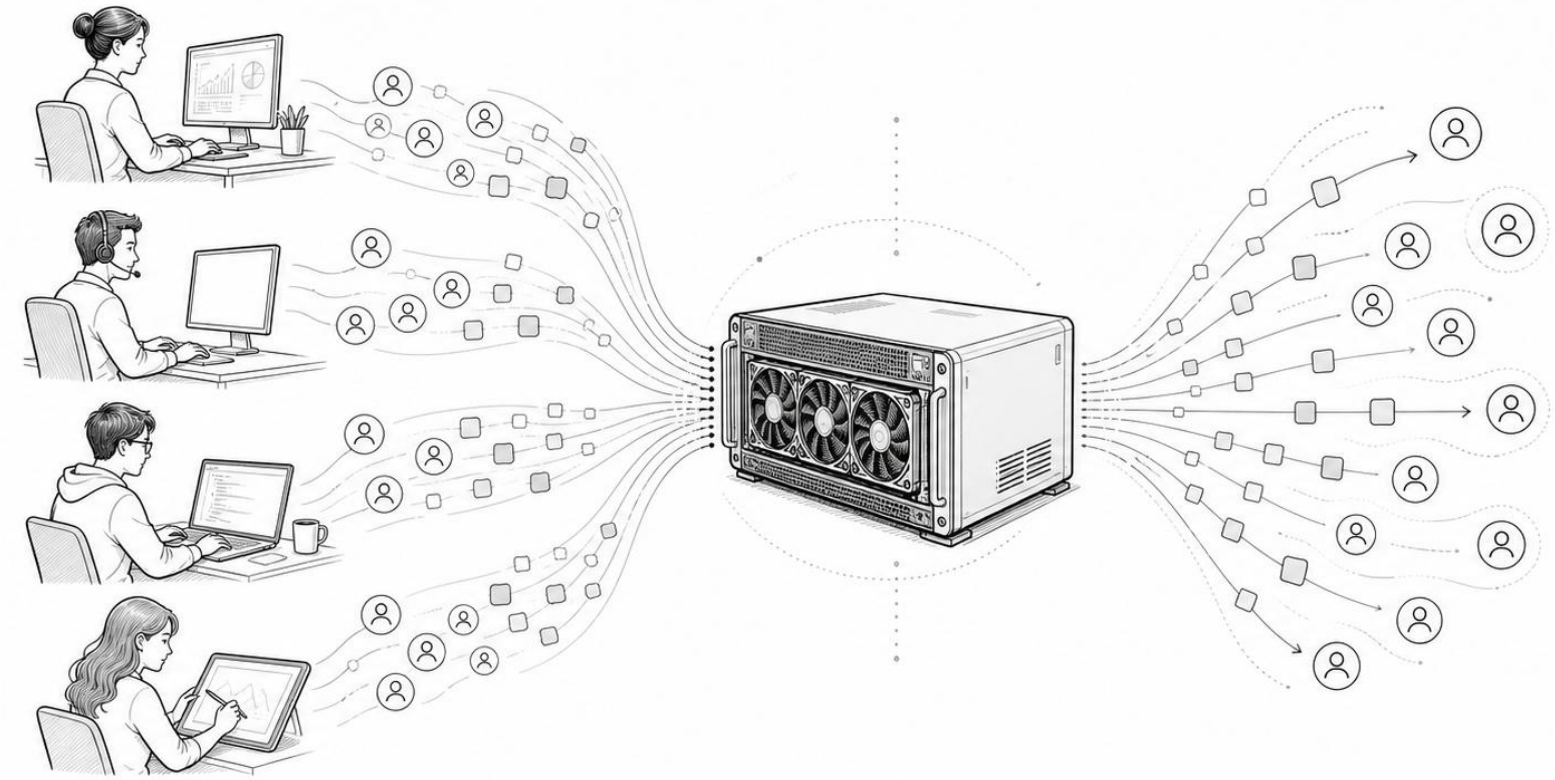
并发更高

同一套 GPU 资源可承载更多会话与实时请求。

OUTCOME 03

成本更低

减少单次推理耗时，让既有硬件完成更多有效工作。



不增加 GPU，也有机会让服务跑得更快、接得更多。

APPLICATION 02

国产 GPU 软件适配：从“能用”走向“好用”

PLATFORM ADAPTATION

STAGE 01

代码能跑

先解决语义兼容和正确执行，完成从源实现到目标平台的迁移。

STAGE 02

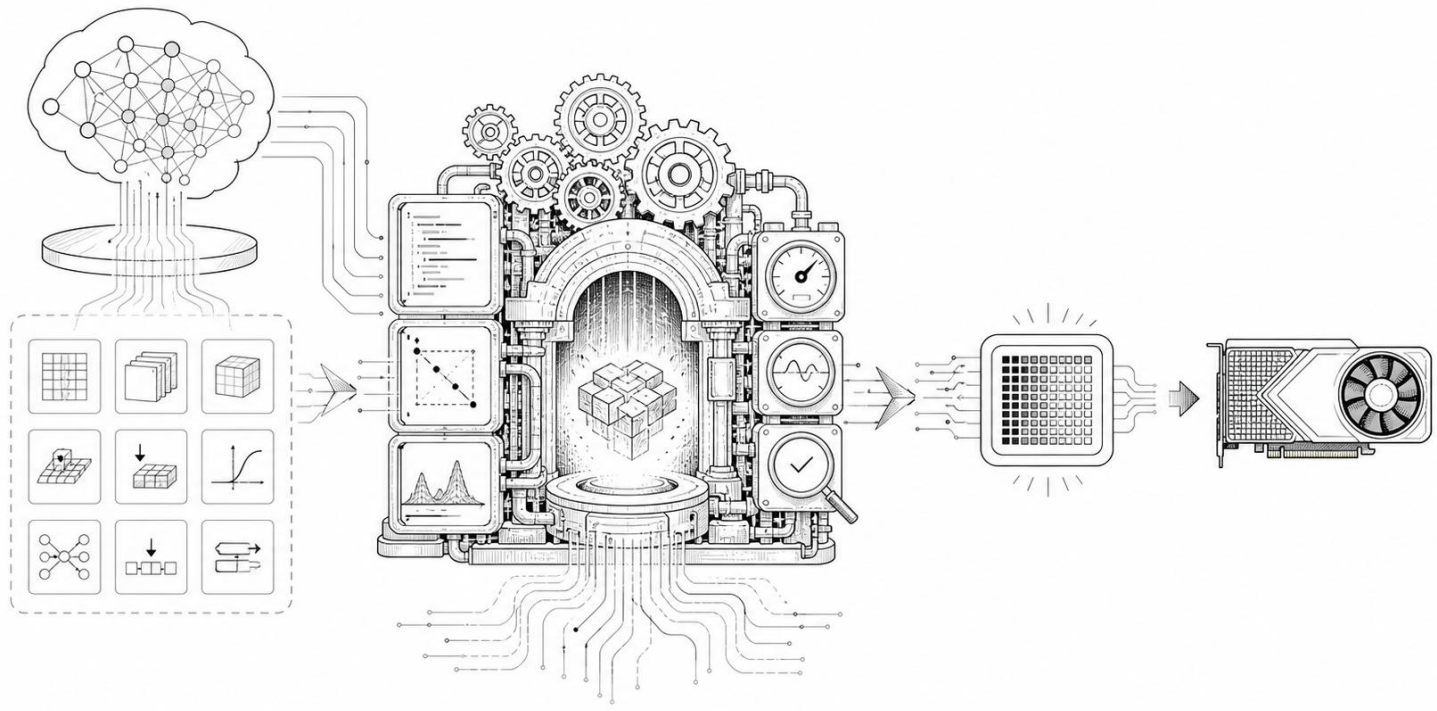
性能对齐

针对架构、编译器、运行时和数据布局持续分析与优化。

STAGE 03

释放硬件能力

让国产 GPU 从“可用算力”变成真正可承载业务的“有效算力”。



真正的竞争力不仅在芯片参数，也在软件能否把芯片性能释放出来。

APPLICATION 03

性能不是一次性项目，而是一项长期运维能力

PERFORMANCE OPERATIONS

CHANGE 01

模型更新

层结构、算子组合和形状分布发生变化。

CHANGE 02

软件升级

编译器、框架与运行时版本改变执行路径。

CHANGE 03

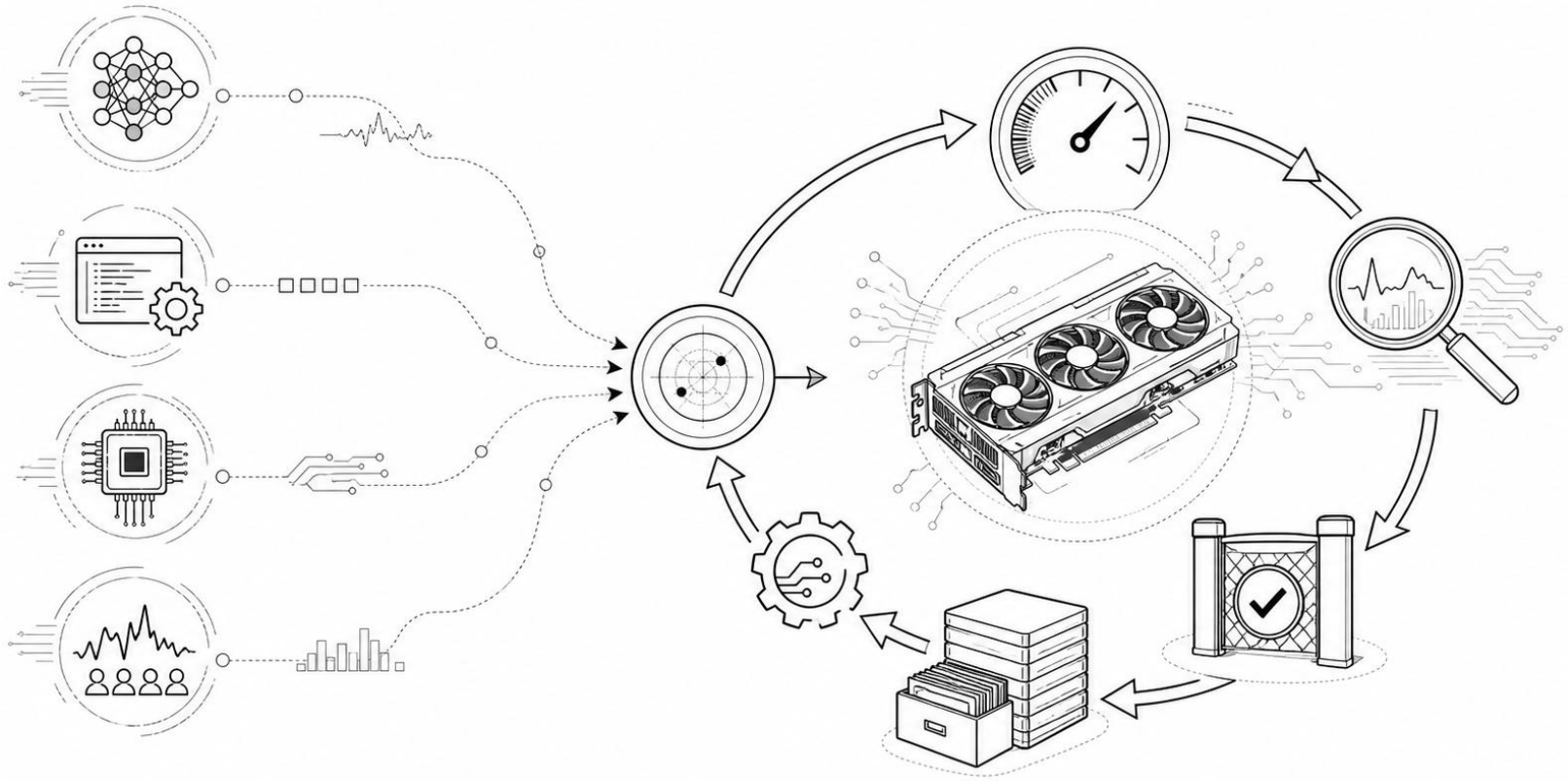
硬件迭代

新 GPU 架构带来新的并行和访存最优点。

CHANGE 04

业务变化

并发、序列长度和请求结构改变原有瓶颈。



保存历史代码、性能数据和实验经验 → 发现回归 → 重新评测 → 复用已有方案

APPLICATION 04

把个人经验转化为团队资产

KNOWLEDGE COMPOUNDING

CAPABILITY 01

过程结构化

把目标、方案、代码、评测与结论组织为统一记录。

CAPABILITY 02

结果可追溯

每次性能变化都有配置、日志与正确性证据。

CAPABILITY 03

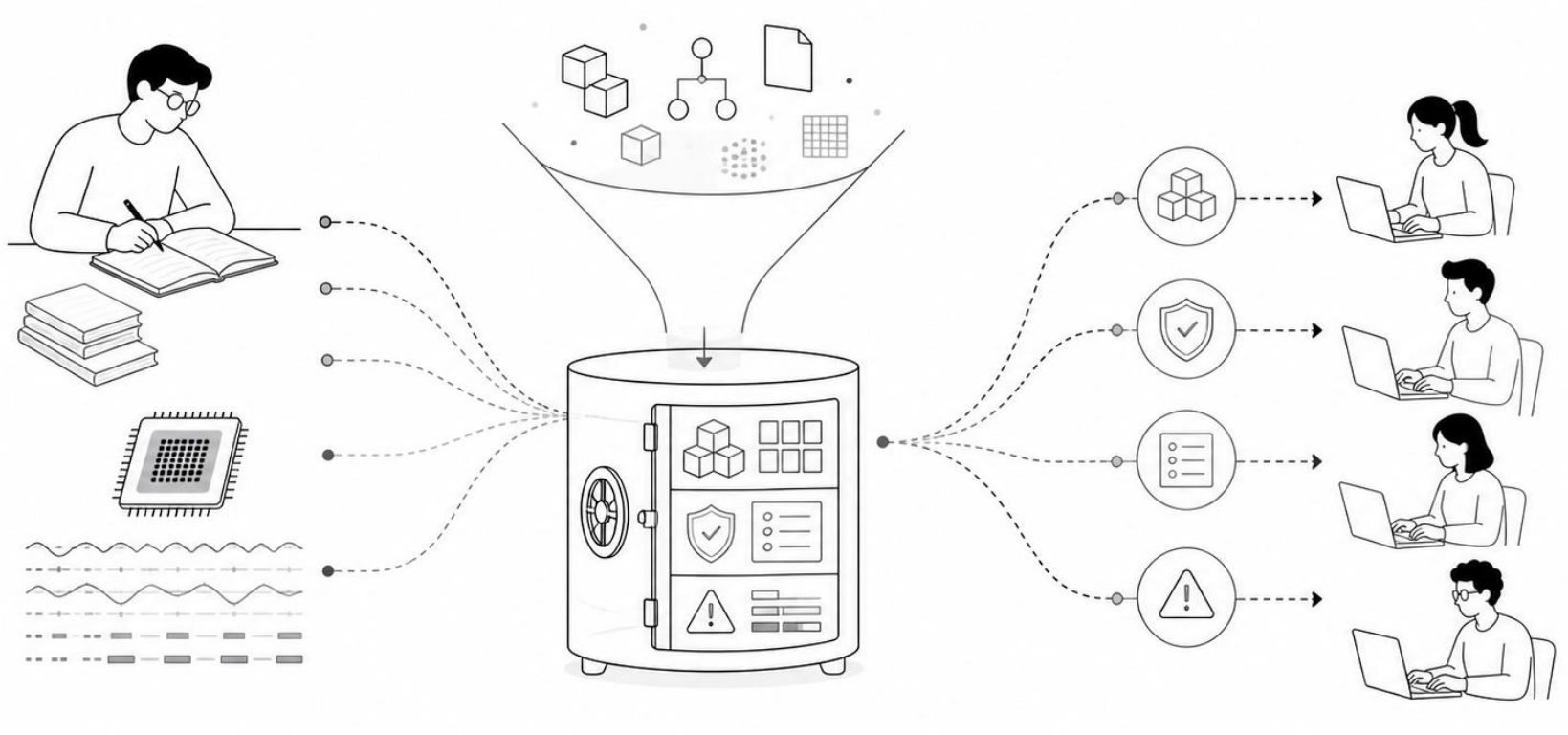
经验可检索

成功策略与失败方向可被后续任务直接发现。

CAPABILITY 04

团队可复用

新人不必从零试错，专家经验持续放大。



沉淀下来的不是一段优化代码，而是一套可持续复用的工程知识。

TAKEAWAY

KernelForge：连接 AI 应用需求与底层算力效率

ONE ENGINE • THREE LEVELS OF VALUE

FOR BUSINESS

更低延迟 • 更高并发 • 更低成本

直接改善在线 AI 服务体验与单位算力产出。

FOR PLATFORM

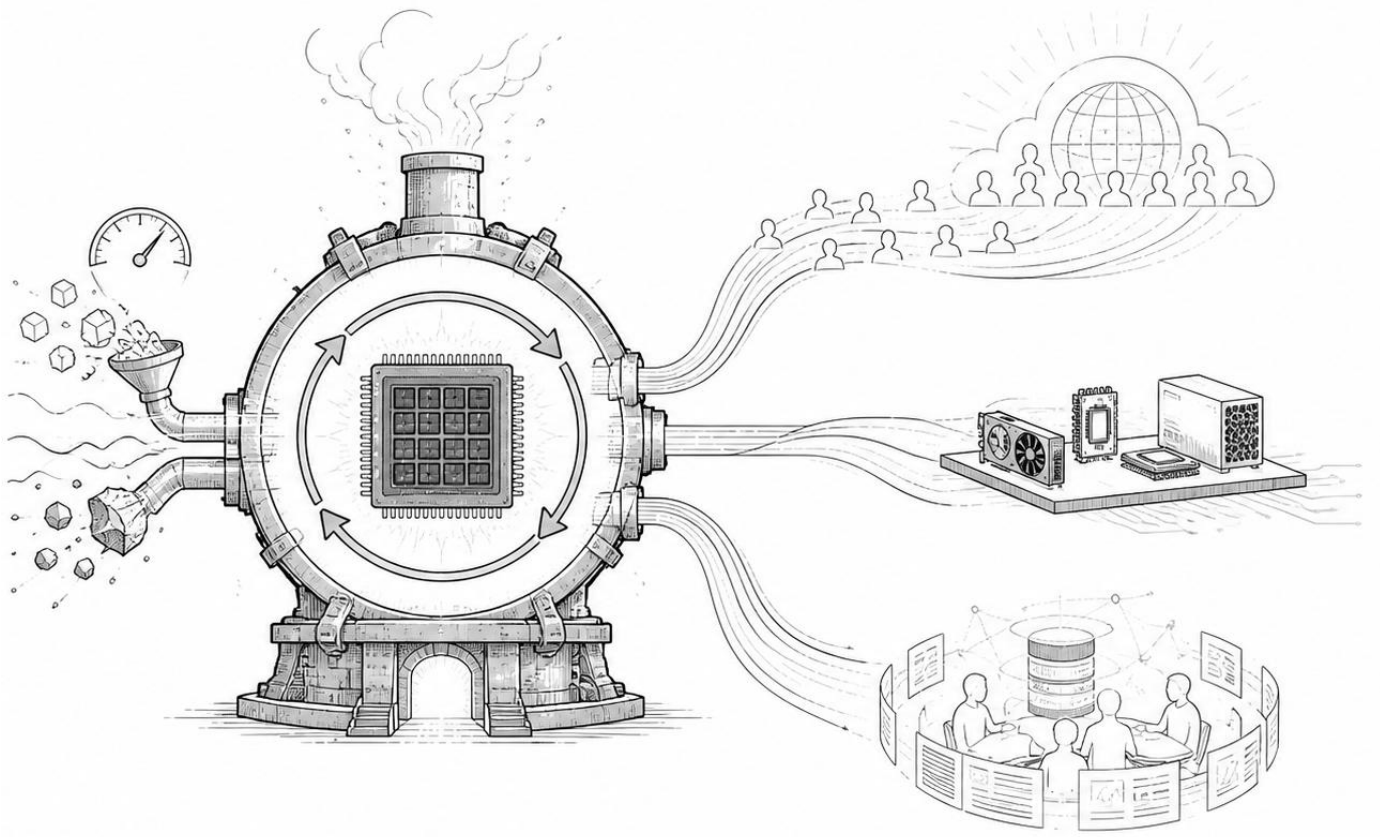
加速适配 • 释放性能 • 支撑生态

让异构与国产 GPU 更快从“能运行”走向“高效运行”。

FOR TEAM

复用经验 • 减少试错 • 沉淀能力

把一次优化转化为长期可积累、可扩展的工程资产。



让每一份宝贵算力，都能够被更聪明、更可信地使用。

APPENDIX

正确性门控和统一测量

正确性

$$E_{max} = \max_i \left| Y_{baseline,i} - Y_{optimized,i} \right|$$

默认通过条件：$E_{max} \leq 10^{-4}$

性能

$$S = \frac{T_{baseline}}{T_{optimized}}$$

20 次预热 · 1000 次正式测量 · 报告平均延迟

执行顺序

基线与候选同输入、同初始化 → 正确性 A/B → 性能 A/B → 必要时 full profiling

APPENDIX

三类 Hook 分别保护基线、评测入口与依赖环境

Hook	审核对象	典型规则	失败处理
pre_bash.sh	Shell 命令	禁止破坏性 git、联网安装依赖、绕过 run.sh	拒绝并返回原因
pre_edit.sh	文件编辑	禁止修改 model.py 与 scripts/；仅允许候选实现	拒绝写入
stop.sh	会话结束	检查代码是否晚于最新 benchmark / profiling 产物	提示补测

不可变条件：参考语义 • 评测逻辑 • 依赖环境