

# Kimi K2



技术报告的核心亮点：

- MuonClip 优化器：在万亿参数规模下实现稳定且token-efficient的预训练
- 2w+ 工具（含真实与模拟工具）：释放可规模化的agent数据潜力
- 结合可验证与自我评估标准奖励的联合强化学习：具备自适应能力的对齐机制
- 超稀疏 1 万亿参数混合专家模型（MoE）：在agent任务上达到开源领域的当前sota水平

Trending on 😊 this week

## Models

moonshotai/Kimi-K2-Instruct
Updated about 2 hours ago · ± 171k · 1.6k
Chain-GPT/Solidity-LLM
Updated 28 days ago · ± 149 · 463
mistralai/Voxtral-Mini-3B-2507
Updated 3 days ago · ± 20.5k · 369
mistralai/Voxtral-Small-24B-2507
Updated 3 days ago · ± 876 · 310
LGAI-EXAONE/EXAONE-4.0-32B
Updated 3 days ago · ± 273k · 179

## Spaces

DeepSite v2 🚀
Generate any application with DeepSeek
FLUX.1 Kontext ✨
Kontext image editing on FLUX[dev]
Anycoder 🎖️
Generate HTML/CSS/Javascript code for web applications
Wan2.1 Fast 🎥
Transform images into dynamic videos
Realistic Text To Speech Unlimited 🔥
Free Text-To-Speech generator with Emotion control (OpenAI)

## Datasets

NousResearch/Hermes-3-Dataset
Updated 10 days ago · ± 2.59k · 182
common-pile/caselaw_access_project
Updated Jun 6 · ± 2.58k · 143
microsoft/rStar-Coder
Updated 1 day ago · ± 4.65k · 112
fka/awesome-chatgpt-prompts
Updated Jan 6 · ± 31k · 8.41k
snorkelai/agent-finance-reasoning
Updated 20 days ago · ± 1.89k · 37

[Browse 1M+ models](#)

[Browse 400k+ applications](#)

[Browse 250k+ datasets](#)

Hugging Face 首页的 Trending 榜完全按**真实下载量 & 点赞数**实时排序，Kimi K2 能冲到周榜第一，等于被全球 1000 万开发者“用手投票”评为本周最火的开源模型。

一起来读技术报告吧~

## Ｋ KIMI K2: OPEN AGENTIC INTELLIGENCE

TECHNICAL REPORT OF KIMI K2

Kimi Team

# 1 预训练

Kimi K2 是一个万亿参数的超稀疏 MoE Transformer，在 15.5 万亿高质量 tokens 上完成预训练。由于高质量人类数据有限，K2 重点优化 token efficiency，采用 Muon 优化器并引入 QK-Clip 技术稳定训练，同时利用合成数据生成提升数据利用率。其架构延续 DeepSeek-V3 风格的 MLA + MoE 设计，并通过底层基础设施优化训练和研究效率。

## 1.1 MuonClip 优化器与 QK-Clip

Kimi K2 使用 **Muon 优化器** 完成训练，该优化器结合了权重衰减与一致更新的 RMS scaling，在月之暗面此前的 Moonlight 项目中，相比 AdamW 在相同计算预算与模型规模下表现更优，显著提升了 token efficiency。

### 训练不稳定性问题

Muon 在大规模训练中易因 attention logits 爆炸导致训练不稳定。传统的缓解方法存在不足，例如，对数软截断 (logit soft - cap) 方法直接对注意力对数进行裁剪，但在执行裁剪操作前，queries 和 keys 之间的点积仍可能过度增长。另一个方法，Query - Key Normalization (QK - Norm) 无法应用于 MLA 机制，因为在推理过程中，其 Key 矩阵无法完全实例化。

### QK - Clip 方案

于是提出了 QK - Clip，在参数更新后对 query 和 key 的投影权重进行重新缩放，以此限制 attention logits 的增长。

设 Transformer 层的输入表示为  $\mathbf{X}$ 。对于每个注意力头  $h$ ，其 query、key 和 value 的投影计算方式如下：

$$\mathbf{Q}^h = \mathbf{X}\mathbf{W}_q^h, \quad \mathbf{K}^h = \mathbf{X}\mathbf{W}_k^h, \quad \mathbf{V}^h = \mathbf{X}\mathbf{W}_v^h$$

其中  $\mathbf{W}_q$ 、 $\mathbf{W}_k$ 、 $\mathbf{W}_v$  为模型参数。注意力输出为：

$$\mathbf{O}^h = \text{softmax} \left( \frac{1}{\sqrt{d}} \mathbf{Q}^h \mathbf{K}^{h\top} \right) \mathbf{V}^h$$

我们将每个注意力头的最大对数 (max logit, 一种标量) 定义为当前批次  $B$  中 softmax 输入的最大值，即：

$$S_{\max}^h = \frac{1}{\sqrt{d}} \max_{\mathbf{X} \in B} \max_{i,j} \mathbf{Q}_i^h \mathbf{K}_j^{h\top}$$

其中  $i$  和  $j$  是训练样本  $\mathbf{X}$  中不同 token 的索引。

**QK - Clip** 的核心思想是，每当  $S_{\max}^h$  超过目标阈值  $\tau$  时，对  $\mathbf{W}_k$  和  $\mathbf{W}_q$  进行重新缩放。重要的是，此操作不会改变当前步骤中的前向/反向计算，这里仅将 max logit 作为指导信号，以确定控制权重增长的强度。

一种简单的实现方式是同时对所有注意力头进行裁剪：

$$\mathbf{W}_q^h \leftarrow \gamma^\alpha \mathbf{W}_q^h, \quad \mathbf{W}_k^h \leftarrow \gamma^{1-\alpha} \mathbf{W}_k^h$$

其中  $\gamma = \min(1, \tau / S_{\max})$ ， $S_{\max} = \max_h S_{\max}^h$ ， $\alpha$  是一个平衡参数，通常设为 0.5，用于对 queries 和 keys 进行均等缩放。

然而，我们发现在实际情况中，仅有小部分注意力头会出现 logits 爆炸的情况。为尽量减少对模型训练的干预，为每个注意力头确定一个缩放因子  $\gamma_h = \min(1, \tau / S_{\max}^h)$ ，并选择逐头应用 QK - Clip。对于常规的多头注意力 (MHA)，这种裁剪操作直接明了。对于 MLA，仅对注意力头的非共享组件进行裁剪：

- $\mathbf{q}^C$  和  $\mathbf{k}^C$  (特定于注意力头的组件)：分别按  $\sqrt{\gamma_h}$  进行缩放；
- $\mathbf{q}^R$  (特定于注意力头的旋转组件)：按  $\gamma_h$  进行缩放；
- $\mathbf{k}^R$  (共享的旋转组件)：保持不变，以避免对不同注意力头产生跨头影响。

简单来说，QK-Clip 就是给 Query/Key 投影权重加动态限幅，从源头防止注意力值爆炸，保证训练稳定性，同时减少对其他注意力头的干扰。

---

#### Algorithm 1 MuonClip Optimizer

---

```

1: for each training step  $t$  do          先用 Muon 优化器做参数更新
2:   // 1. Muon optimizer step
3:   for each weight  $\mathbf{W} \in \mathbb{R}^{n \times m}$  do
4:      $\mathbf{M}_t = \mu \mathbf{M}_{t-1} + \mathbf{G}_t$            ▷  $\mathbf{M}_0 = \mathbf{0}$ ,  $\mathbf{G}_t$  is the grad of  $\mathbf{W}_t$ ,  $\mu$  is momentum
5:      $\mathbf{O}_t = \text{Newton-Schulz}(\mathbf{M}_t) \cdot \sqrt{\max(n, m)} \cdot 0.2$     ▷ Match Adam RMS
6:      $\mathbf{W}_t = \mathbf{W}_{t-1} - \eta (\mathbf{O}_t + \lambda \mathbf{W}_{t-1})$            ▷ learning rate  $\eta$ , weight decay  $\lambda$ 
7:   end for
8:   // 2. QK-Clip      再针对注意力头做权重裁剪，以解决 Attention logits 爆炸问题
9:   for each attention head  $h$  in every attention layer of the model do
10:    Obtain  $S_{\max}^h$  already computed during forward
11:    if  $S_{\max}^h > \tau$  then
12:       $\gamma \leftarrow \tau / S_{\max}^h$ 
13:       $\mathbf{W}_{qc}^h \leftarrow \mathbf{W}_{qc}^h \cdot \sqrt{\gamma}$ 
14:       $\mathbf{W}_{kc}^h \leftarrow \mathbf{W}_{kc}^h \cdot \sqrt{\gamma}$ 
15:       $\mathbf{W}_{qr}^h \leftarrow \mathbf{W}_{qr}^h \cdot \gamma$ 
16:    end if
17:  end for
18: end for

```

---

## 1.2 预训练数据：通过重写提高 Token 利用效率

Kimi K2 在15.5 万亿高质量 Token 上预训练，覆盖网页文本、代码、数学和知识四大领域。与 Kimi K1.5 相比，K2 的关键改进是引入数据重写（rewriting）策略，**以在高质量 Token 供应有限的情况下最大化 Token 的学习效用。**

下面介绍两个面向知识和数学领域的重写技术，它们实现了这种数据增强：

### 知识数据重写

- **多样化风格和视角的Prompt**: 为了在保持事实完整性的同时增强语言多样性，应用了一系列精心设计的Prompt，引导大语言模型以多种风格和不同视角对原始文本进行忠于原文的重写。
- **分块自回归生成**: 为了保持长文档的全局连贯性并避免信息丢失，采用了基于分块的自回归重写策略。文本被分成小段，单独重写，然后再拼接成完整的段落。这种方法减轻了LLMs常见的输出长度限制。该流程概述见图4。
- **保真度验证**: 为确保原文与重写内容的一致性，进行了保真度检查，比较每个重写段落与原文的语义对齐情况。这作为训练前的初始质量控制步骤。

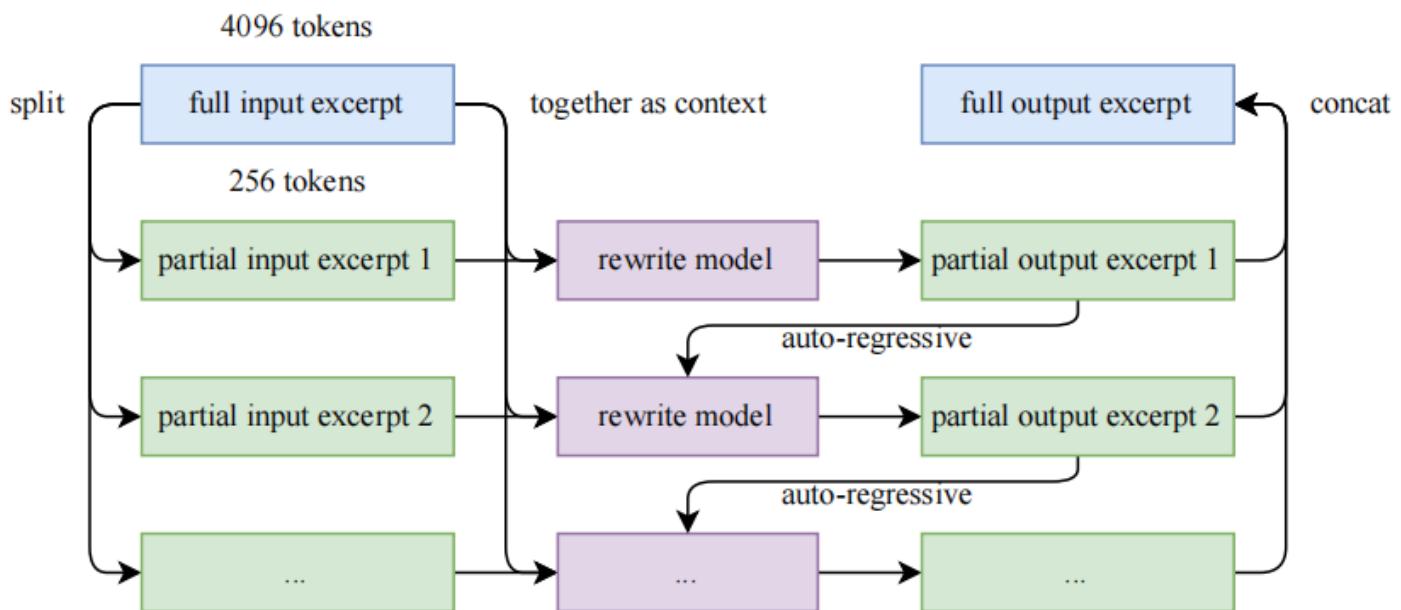


图4：长文本片段的自回归分块重写pipeline。输入被分割成小块，在保留上下文的情况下顺序重写，然后拼接成完整的重写段落。

### 数学数据重写

将高质量数学文档改写为“学习笔记”风格（参考 SwallowMath），并通过跨语言翻译扩展数据多样性，进一步提升模型的数学推理能力。

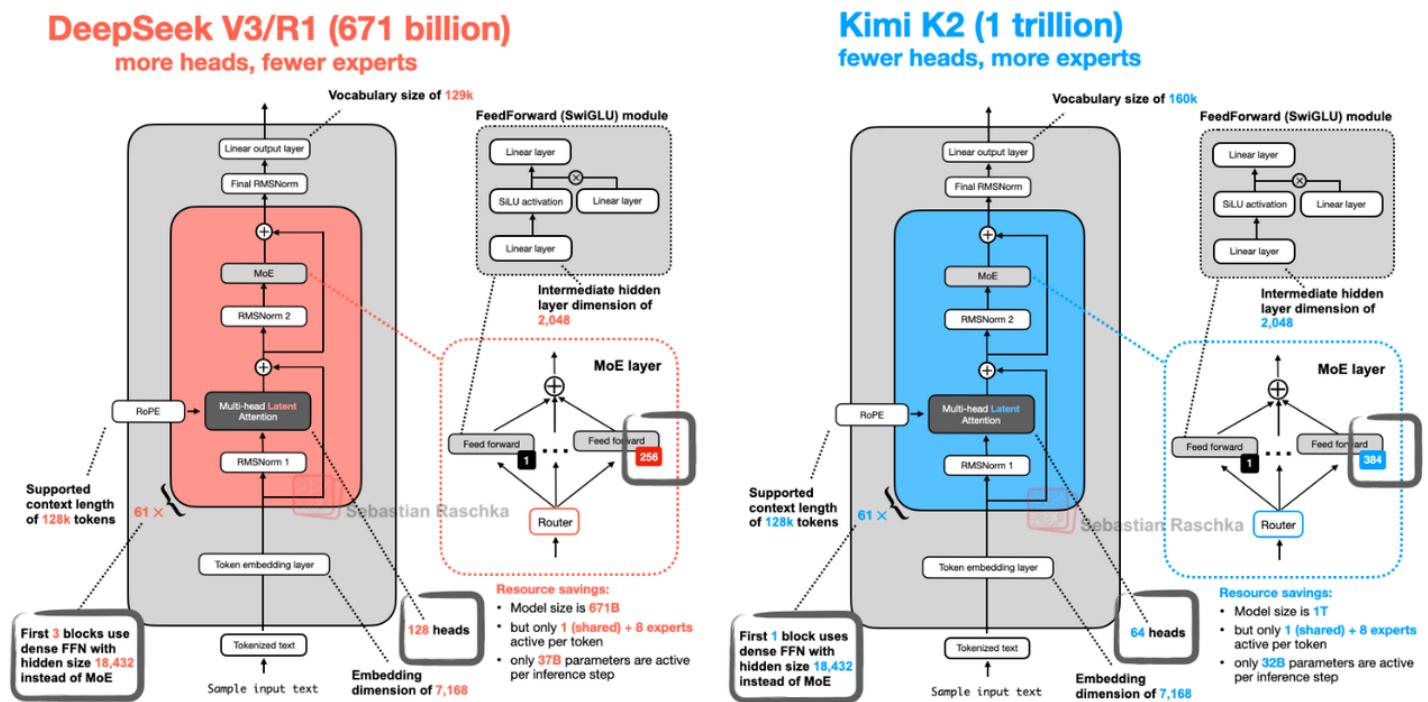
## 2 模型架构

从模型架构上看，Kimi K2 和 DeepSeek-V3 的设计思路一脉相承，两者都采用Mixture-of-Experts (MoE) 架构，并使用多头潜在注意力机制 (MLA)。

不过，Kimi K2 在两个关键点上做了调整：

**1. MoE模块：增加专家数量** —— 稀疏度指的是专家总数与激活专家数的比值。实验表明，增加专家总数（提高稀疏度）能够持续降低训练和验证损失，从而提升整体模型性能。Kimi K2 采用48 的稀疏度，即每次前向计算激活384 个专家中的 8 个。

**2. MLA模块：减少注意力头数** —— DeepSeek-V3 的注意力头数量设置为模型层数的两倍（128 个），但考虑到稀疏度 48 已经带来了较强性能，翻倍注意力头数带来的边际收益不足以抵消额外的推理开销，因此 Kimi K2 最终选择64 个注意力头。



### 3 训练基础设施

Kimi K2 的训练采用**多维并行策略**（流水线并行PP + 专家并行EP + ZeRO-1数据并行DP）来充分利用 GPU 资源。训练过程中，模型的计算、通信和数据卸载被交错并行安排，从而避免资源闲置。

为降低内存压力，K2 将部分优化器状态卸载到 CPU，并通过**选择性重计算**和**FP8 存储**减少激活内存；剩余的激活值也通过 CPU Offload 流式传输，与计算和通信并行执行。

图7展示了这种交错：在一个 micro-batch 的前向计算还在进行时，另一个 micro-batch 的反向传播、专家通信以及激活卸载可以同时进行。实现了**1F1B交错调度**。这种机制让训练大模型时，GPU 几乎无闲置状态，大幅提升整体效率。

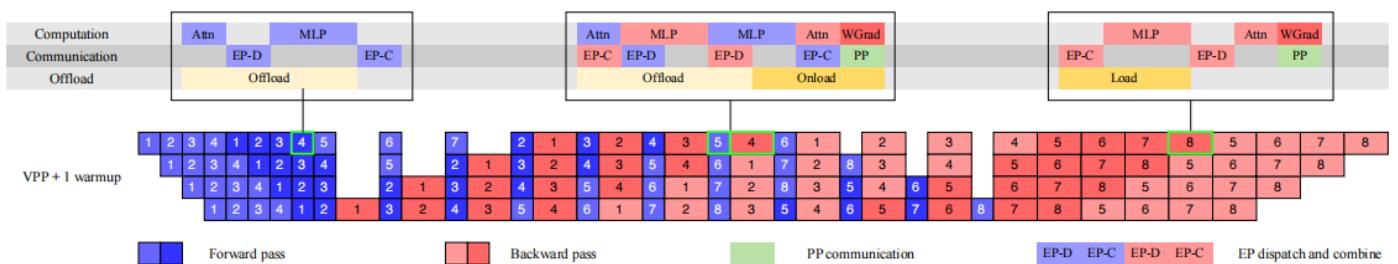


图 7 展示了在不同 PP 阶段中，计算-通信-卸载的交错模式

## 4 后训练

### 4.1 SFT

Kimi K2 的后训练阶段使用 Muon 优化器进行监督微调。SFT 数据集覆盖多种任务领域，核心目标是最大化 Prompt 多样性和确保响应质量。为此，kimi 团队开发了数据生成 pipeline，结合了人工标注、提示工程和质量验证，并通过专家模型与 LLM 生成候选回答，再由自动化与人工评审进行筛选。

在工具使用（Tool Use）学习上，K2 通过大规模 agentic 数据合成教会模型多步骤交互和工具调用能力：

- 图 8 展示了数据合成流程：从工具库中生成工具规范（包括真实和合成工具），再创建 Agent 和任务，最后生成并过滤执行轨迹。
- 图 9 通过 t-SNE 可视化展示了工具库的多样性：真实 MCP 工具和合成工具分别覆盖了互补的功能空间，实现系统化覆盖。

该策略让 K2 在多领域指令跟随和复杂工具使用方面显著增强，确保模型能高效应对实际应用任务。

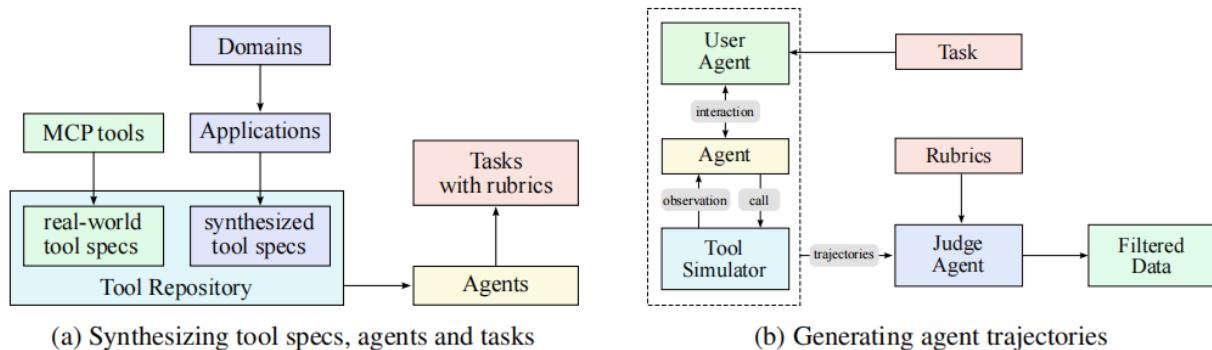
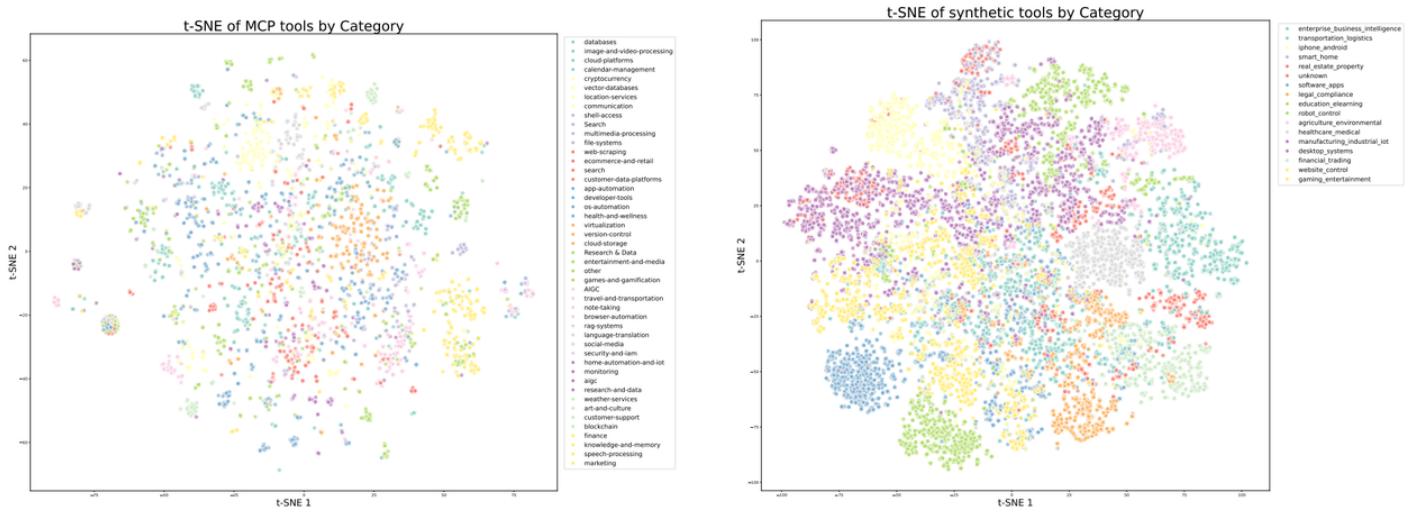


图 8：工具使用的数据合成 pipeline

- (a) 工具规范（tool specs）来自真实工具和 LLM 合成工具；Agent 和任务则从工具库中生成。  
(b) Multi-Agent pipeline 通过调用工具生成并筛选执行轨迹。



(a) t-SNE visualization of real MCP tools, colored by their original source categories

(b) t-SNE visualization of synthetic tools, colored by pre-defined domain categories

图 9：工具嵌入的 t-SNE 可视化。

- (a) 真实 MCP 工具按其原始来源类别进行着色并展示自然聚类。
- (b) 合成工具按预定义的领域类别进行组织，系统化覆盖工具空间。两者结合后，实现了对不同工具功能的全面表示。

## 4.2 强化学习

Kimi K2 在强化学习 (RL) 阶段，目标是通过更高效的 token 使用和更强泛化能力来超越 SFT。基于 K1.5 的策略优化算法，K2 扩展了 RL 训练的任务多样性和 FLOPs，并构建了一个可扩展的 Gym 框架，涵盖数学、STEM、逻辑推理、复杂指令跟随、代码与软件工程及安全性任务。在可验证任务（如 QA、逻辑推理）中，采用基于规则或单元测试的可验证奖励 (Verifiable Rewards)；而在主观任务（如创意写作和开放式问答）中，引入自我批评奖励 (Self-Critique Rubric Reward)，模型通过生成多条候选响应并成对比较，给出自己的偏好信号。

**策略优化** 遵循下列目标函数：

$$L_{RL}(\theta) = \mathbb{E}_{x \sim \mathcal{D}} \left[ \frac{1}{K} \sum_{i=1}^K \left( r(x, y_i) - \bar{r}(x) - \tau \log \frac{\pi_\theta(y_i|x)}{\pi_{\text{old}}(y_i|x)} \right)^2 \right]$$

其中  $\bar{r}(x) = \frac{1}{K} \sum_{i=1}^K r(x, y_i)$  是  $K$  个样本响应的平均奖励， $\tau > 0$  是正则化参数，用于平衡策略更新的稳定性与探索。

**数据与奖励体系：** K2 收集了大量高质量 QA 对（专家标注 + 内部 QA pipeline + 开放数据集），并对逻辑任务覆盖了多种格式（如多跳表格推理、逻辑谜题）。复杂指令跟随采用混合验证框架：代码解释器用于规则性验证，LLM 判别器评估满足隐式约束，外加对抗检测与课程生成流水线。代码和软件工程任务基于 GitHub Pull Requests 和真实开发环境构建奖励，利用 Kubernetes 支撑 10,000+ 并发沙盒实例，通过单元测试给出可验证奖励。安全性方面，K2 构建了攻击模型（生成对抗 prompt）—目标模型—评审模型的三阶段 pipeline，检测并防御越狱攻击。

**优化技巧：**K2 使用 Muon 优化器并增加了多项工程改进：

- **Budget Control**: 限制每条响应的最大 token 数，对超长响应截断并施加惩罚，以提升 token 效率。
- **PTX Loss**: 混入人工高质量样本作为额外损失项，防止灾难性遗忘并提高泛化。
- **Temperature Decay**: 训练初期保持高温度促进探索，后期逐步降温以收敛到稳定高质量的输出。
- **闭环 Critic 对齐**: Critic 使用可验证奖励不断更新自己的评估标准，确保策略优化与偏好信号始终保持一致。

通过**可验证奖励 + 自我批评奖励**双轨机制和上述工程优化，Kimi K2 的 RL 显著提升了复杂任务的对齐性、推理深度与生成质量。

### 4.3 强化学习基础设施（RL Infrastructure）

Kimi K2 的 RL 基础设施采用**同位架构（Colocated Architecture）**，训练引擎与推理引擎在同一节点上交替工作，以便高效共享 GPU 资源。整个流程由**中央控制器协调**：推理引擎生成新数据后，参数通过**checkpoint 引擎**广播给训练引擎；训练完成后，更新的参数再次通过 checkpoint 引擎推送给推理引擎（图 10 所示）。

为解决 1T 级参数规模下的高效参数切换，K2 设计了**分布式 checkpoint 引擎**：每个 worker 会先获取本地参数副本，然后广播给全体 checkpoint 节点，推理引擎再按需拉取对应分片。虽然这种方式会传输更多数据，但避免了复杂的分片同步，仅需 30 秒即可完成一次全量参数更新，大幅降低了维护成本。

此外，K2 优化了**系统启动**：通过 checkpoint 引擎集中加载参数，再将状态分发给推理引擎副本，减少磁盘 IO 并消除单点故障。

在**agenetic rollout** 中，K2 使用**并行多环境 + partial rollout 技术**，避免长尾任务阻塞，并通过类似 OpenAI Gym 的统一接口简化新环境接入。

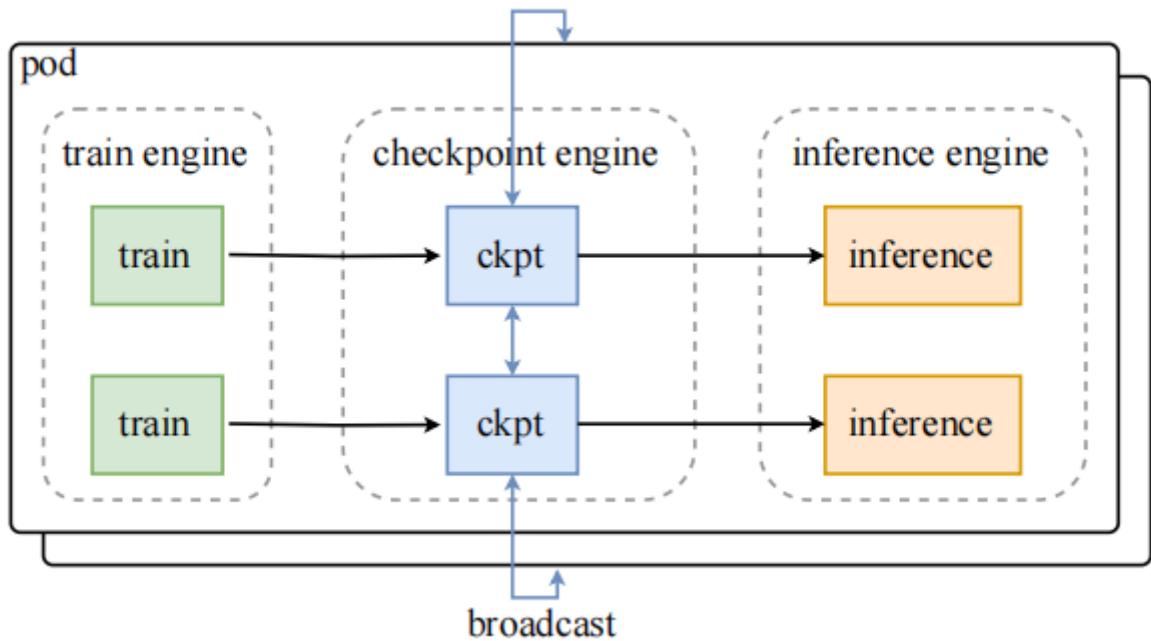


图 10：使用 checkpoint 引擎进行参数更新的流程：训练引擎的参数先传输至 checkpoint 引擎，再广播至推理引擎。