LIBRA：面向时间高效的大语言模型训练的批大小与并行度运行时协同调优

Shengkai Lyu[1,2]
lvshengkai23@mails.ucas.ac.cn

Hairui Zhao[1,*]
zhaohairui@ict.ac.cn

Daran Sun[1,2]
sundaran24s@ict.ac.cn

Yida Gu[1,2]
guyida@ict.ac.cn

Xingchen Liu[1,2]
liuxingchen232@mails.ucas.ac.cn

YueYuan Zhou[1]
zhouyueyuan@ict.ac.cn

Zheng Wei[1]
weizheng@ncic.ac.cn

Guangming Tan[1]
tgm@ict.ac.cn

Dingwen Tao[1], *
taodingwen@ict.ac.cn

[1]中国科学院计算技术研究所，中国科学院，北京，中国[2]中国科学院大学，北京，中国 *通讯作者：Hairui Zhao 和 Dingwen Tao。

arXiv:2611.32008v1 [cs.DC] 24 Nov 2026

摘要

训练大语言模型（LLM）需要同时选择批大小和并行配置。现有系统将这些决策分开处理：针对吞吐量调优并行度，针对收敛调优批大小，但单独优化任一目标都无法决定训练效率。我们提出 LIBRA，这是一种轻量级运行时框架，可协同调优批大小和并行配置，以最小化端到端时间。LIBRA 将吞吐量和收敛视为两个带噪声的目标，并使用并行噪声期望超体积改进（qNEHVI）搜索二者的联合空间。为提高运行时搜索效率，我们提出分层剪枝：首先通过系统级性能估计移除不可行和低吞吐量的并行配置，然后通过临界批大小感知的收敛监测进一步缩小批量大小候选项范围。我们还设计了采集函数引导的并行评估（AGPE），按照采集值的顺序，在不同 GPU 组上并发评估保留下来的候选项。在最多包含 256个 GPU 的集群上，与最先进方法相比，LIBRA 将收敛时间平均缩短了 31.1%，而运行时调优仅占总时间的 0.3–3.5%。

1 引言

大语言模型（LLM） [3, 11, 25, 32, 39, 40]推动了众多应用的快速发展。其训练越来越依赖大型 GPU集群。例如，LLaMA 3 405B [25] 的训练最多使用了 16K 个 H100 GPU。扩展此类工作负载需要结合数据并行（DP） [35], 、张量并行（TP） [37],、流水线并行（PP） [12], 和上下文并行（CP） [13], , 同时选择微批大小（MBS）和全局批大小（GBS），从而形成一个需要调优的庞大配置空间。

为什么需要协同调优? 训练性能通常使用步时间、令牌吞吐量 [31, 43], 或模型浮点运算利用率（模型浮点运算利用率） [25], 来报告，而不是达到收敛所需的时间。然而，最大化系统吞吐量并不一定能最小化端到端时间。表1比较了三种调优目标。仅针对吞吐量进行调优可获得最高吞吐量，但需要更多令牌才能达到目标损失。仅针对收敛进行调优所需令牌最少，但吞吐量低得多。两种目标都无法最小化端到端时间。同时针对两者进行调优时，既无法获得最高吞吐量，也无法使用最少令牌，但达到目标损失的速度比仅针对吞吐量进行调优快30.8–34.8%，比仅针对收敛进行调优快44.1–50.2%。因此，端到端训练取决于处理速率（即吞吐量）和每令牌进展（即收敛）这两方面。

表1。 在三个调优目标下选定的配置：吞吐量、收敛，以及两者兼顾（吞吐量与收敛）。对于给定模型，所有配置都能达到相同的目标损失，因此总训练时间可以直接比较。

模型（目标损失）	GPUs	调优 goal	吞吐量（千词元/秒）	令牌目标值（B）	Time (h)
Code LLaMA 34B ($L_{val} \to 2.25$)	128	吞吐量	**154.16**	70.93	127.80
		收敛	58.41	**33.28**	158.28
		吞吐量与收敛	121.02	38.55	**88.48**
LLaMA-2 70B ($L_{val} \to 2.22$)	256	吞吐量	**82.35**	88.76	299.42
		收敛	24.66	**34.82**	392.28
		吞吐量与收敛	58.8	41.32	**195.22**

然而，吞吐量与收敛通过训练配置相互耦合，并共同决定端到端训练效率。具体而言，每个数据并行副本执行 GBS/(DP × MBS) 个微批次，因此较小的 GBS 会限制 DP，也就几乎没有足够的工作量来摊销流水线气泡。MBS 控制每GPU利用率。为了让流水线持续繁忙，通常需要 GBS $\geq 2 \times$ PP $\times$DP $\times$ MBS，这可能超过有利于收敛的 GBS。对于 LLaMA-2 13B，恒定 GBS 可实现 112.00K 令牌 / 秒的吞吐量，但需要消耗 51.32B 个词元。渐增（32 → 128 → 512）将词元需求降至 34.00B，并行调优将吞吐量从 83.13K 提高至 100.46K 令牌/秒，而联合协同调优则以 28.27B 个词元在 81.37 小时内达到目标。

因此，在训练过程中，批大小必须与并行配置协同调整。

现有方法的局限性。Alpa [43], Aceso [23], FlexFlow [15], 和 Hyper- tron [21] 等并行自动调优器会优化执行吞吐量，却将批大小及其收敛效应排除在外。相反，Hyper-band [19], ASHA [20], BOHB [10], 和 DEHB [1]等多保真超参数优化（HPO）方法会淘汰前景不佳的批大小试验，同时固定并行配置。实际上，LLaMA 3 [25], DeepSeek-V3 [7], 和 MiniMax-01 [29] 使用针对特定模型手工设计的渐进式训练调度，而分支式训练 [27] 则根据部分运行估算临界批大小（CBS）。这两种策略都无法迁移到其他模型，也无法在训练过程中自适应调整。Pollux [33] 最接近这一目标，它会协同调整批大小和资源分配，但其目标是集群调度，且不会搜索并行配置。总之，现有方法都无法在线低成本地协同调优批大小和并行配置。

研究挑战。 这些局限性要求一种轻量级运行时机制，能够随着训练进程共同调优批大小和并行配置。这带来了两项挑战。首先，联合配置空间规模庞大，且会在训练过程中不断变化。在64个GPU上，它包含约150万个候选项；在128个GPU上则约有190万个候选项，而经过可行性检查后仍有超过1.1万个候选项。其次，运行时评估成本高昂。每个候选项都要占用多个GPU，并且必须观测其吞吐量和收敛情况，因此顺序评估并不可行。

我们的解决方案。 为应对这些挑战，我们提出了LIBRA——一种轻量级运行时框架，可共同调优批大小和并行配置，以优化吞吐量和收敛情况。LIBRA将两者都建模为噪声目标，并使用并行噪声期望超体积改进（qNE-HVI）[5]来选择有前景的配置。为应对庞大且不断变化的搜索空间（挑战1），分层剪枝将性能估计和吞吐量感知剪枝与收敛监测及CBS感知的GBS剪枝相结合。为控制运行时评估的成本（挑战2），采集函数引导的并行评估（AGPE）会在多个GPU组上并发评估保留的候选项，并将异步观测反馈给优化器。随后，LIBRA会在训练过程中应用选定的配置。本文的贡献如下：

- 我们将联合吞吐量–收敛调优确定为大规模大语言模型训练中缺失的一环，并将批大小和并行配置的选择 formulates 为噪声多目标优化问题，通过基于qNEHVI的贝叶斯优化求解该问题。
- 我们通过利用搜索空间中的有效区域会随训练进度而变化这一事实，使运行时协同调优更加轻量：LIBRA 通过性能估计剪枝低性能的并行配置，然后通过在线收敛监测和 CBS感知的GBS剪枝缩小批量大小候选项范围。
- 我们设计了采集函数引导的并行评估（AGPE），利用qNEHVI采集函数值、GPU需求和估计的评估时间选择并放置候选项，并将观测结果异步返回给优化器。
- 在最多使用256个GPU、涵盖多种大语言模型的实验中，与最先进方法相比，LIBRA平均将端到端收敛时间缩短了31.1%，而运行时调优仅占端到端训练时间的0.3–3.5%。

2 背景

2.1 面向吞吐量的并行配置

分布式大语言模型训练结合了多个并行维度。数据并行（DP） [35] 对批次进行分区并同步梯度。张量并行（TP） [37] 对层内算子进行分区。流水线并行（PP） [12, 30] 将层划分为多个阶段，上下文并行（CP） [13, 24]对长序列进行分区。状态分片方法，包括零冗余优化器（ZeRO） [34] 和全分片数据并行（FSDP） [42], ，用于分布模型状态。

MBS 控制每个 GPU 的内存使用量和内核效率，而 GBS 决定每个优化器步中参与的微批次和数据并行副本数量。这些选择共同决定通信、流水线气泡、GPU 利用率和吞吐量。Alpa [43]和 Aceso [23] 等系统使用性能分析或成本模型来选择可行的高吞吐量配置。

2.2 面向收敛的批次大小递增

较大的批大小会降低梯度方差并释放更多数据并行度，但最终对减少优化器步的作用会变得很小。对于逐样本梯度 g_i、均值 $G = \mathbb{E}[g_i]$ 和协方差 $\Sigma = \mathrm{Cov}(g_i)$，McCandlish 等人 [26] 将简化的梯度噪声尺度（ GNS ）定义为 $B_{\text{noise}} = \mathrm{tr}(\Sigma)/\|G\|_2^2$，其近似于临界批大小（CBS） [26, 36]。CBS 标志着优化器步与词元之间权衡发生转变的位置：在此之下，增大批大小几乎会按比例减少优化器步；而在此之上，步数节省会逐渐减少，同时消耗的词元仍在持续增加。因此，它是时间效率与词元效率之间的转折点，而不是一个严格的收敛上限 [2]。由于 GNS 通常会随着损失降低而增大，训练过程中有益的批大小也会随之增加。分支式训练 [27] 在 OLMo 1B/7B 上证实了这一趋势：在临界批次大小引导的递增策略下，OLMo-1B 的损失略有改善，同时优化器步减少了 43%。

已有工作在多个粒度上调整批大小：通过 Hyperband [19]和 ASHA [20],进行部分训练超参数优化，采用运行中规则（如 AdaBatch [8] 以及耦合的批次大小与学习率缩放 [38], ），以及大语言模型规模的分支式训练 [27]。然而在生产环境中，调度仍然

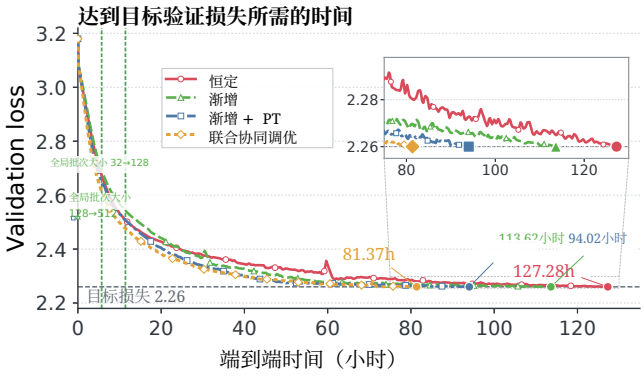


图 1. LLaMA-2 13B 的恒定 GBS、GBS 渐增、GBS 渐增并结合并行调优（PT），以及联合协同调优的比较。曲线展示了验证损失随端到端时间的变化，并标出了 GBS 的变化和达到目标所需的时间。

依赖针对每个模型的手动调优：LLaMA 3 [25] 将令牌批次从 4M→8M→16M 逐步调整，DeepSeek-V3 [7] 在 469B 个词元的训练过程中将序列批次从 3,072 增长到 15,360，而 MiniMax-01 [29]使用 16M→32M→64M→128M。

2.3 多目标优化方法

多目标优化旨在平衡相互冲突的目标。帕累托最优配置需要在各目标之间权衡，其目标向量构成帕累托前沿 [28]。

标量化方法。加权和与切比雪夫效用函数将多个目标简化为一个目标 [28], ，但依赖归一化和固定偏好。加权和可能会遗漏非凸前沿上的非支撑点。

基于帕累托的进化方法。 NSGA-II [6]采用非支配排序和拥挤距离，而 SPEA2 [44] 采用基于强度的适应度、密度和档案集。这类种群方法适用于离散的非凸空间，但通常需要大量评估。

基于代理模型的多目标贝叶斯优化。多目标贝叶斯优化利用概率代理模型和采集函数处理代价高昂的观测。ParEGO [18] 将期望改进应用于随机化切比雪夫标量化。期望超体积改进（EHVI） [9] 用于衡量期望超体积增长。qNEHVI [5] 通过整合潜在前沿中的不确定性，将 EHVI 扩展到含噪声的批量选择。

3 动机与挑战

本节说明为什么在训练过程中必须联合优化吞吐量和收敛性，随后总结促使我们提出 LIBRA 的局限性与挑战。

3.1 为什么要协同调优吞吐量和收敛性?

单看吞吐量并不能缩短收敛所需时间。表1比较了针对 Code LLaMA 34B 和 LLaMA-270B 的三种调优目标。针对吞吐量进行调优可达到最高处理速率，针对收敛进行调优则所需词元最少，但二者完成时间都晚于同时针对这两个目标进行调优；后者达到相同目标损失的时间比仅针对吞吐量进行调优缩短了 30.8–34.8%。达到目标所需时间等于所需词元数除以平均令牌吞吐量。全局批次大小会影响这两个量。因此，吞吐量较低的配置也可能因所需词元更少而更早完成，所以在整个配置搜索过程中，必须将吞吐量和收敛作为独立目标保留。

批大小与并行配置相互耦合，且这种耦合关系会在训练过程中发生变化。由于 GBS =、DP ×、MBS×（累积的微批次）之间的关系，较小的 GBS 会限制 DP，并可能导致微批次过少，无法摊销流水线气泡；而较大的 GBS 则能释放更多数据并行度。反过来，TP、PP、CP 和 MBS 决定了给定 GBS 所需的内存、通信和流水线开销。对于运行在 64个 GPU 上的 LLaMA-2 13B，将 GBS 从 16 提高到 256 会使选定的 DP 从 2 变为 32，并使活动 GPU 分配从 8个GPU 增加到 64个GPU，因此，吞吐量最优的并行配置是批大小决策的函数。这种依赖关系也会随着训练过程而演变（图 1）：恒定 GBS 可实现最高吞吐量，但需要处理 51.32B 个词元并耗时 127.28 小时；采用渐增 GBS$32 \rightarrow 128 \rightarrow 512$可减少词元需求，并在 113.62 小时内达到目标；沿同一渐增过程重新调优并行配置，则可将这一时间进一步缩短至 94.02 小时。由于这种组合仍然是分别进行选择，因此会被联合协同调优超越：后者将词元需求降至 28.27B，以较低的平均吞吐量（96.50K 对比 100.46K 令牌/秒）在 81.37 小时内完成训练——这证实了 GBS 与并行配置必须在运行时一并选择。

3.2 现有调优范式的局限性

现有方法涵盖联合搜索的不同部分。表 2 比较了它们在训练期间更新了哪些决策。

并行配置搜索。 并行配置搜索同时选择 DP、TP、PP、CP 和 MBS。Alpa [43]、Aceso [23]、FlexFlow [15],和 Hyper- tron [21] 等系统搜索并行配置，以减少迭代时间或提高吞吐量。它们的搜索假设计算任务已确定，并将 GBS 及其对收敛的影响排除在优化范围之外。在给定外部批次调度的情况下，这些系统可以针对每个指定的 GBS 找到快速的并行配置，但无法确定何时更改 GBS，也无法判断该选择是否能最小化收敛时间。

收敛优化与在线调优。 收敛优化会在选择配置时考虑训练进度。在线调优会利用已完成的评估来更新后续调优决策。Hyperband [19], ASHA [20], BOHB [10], 和 DEHB [1] 等多保真度方法会在不同超参数配置之间分配部分训练资源，并提前停止表现不佳的运行。ASHA 和 BOHB 会利用部分运行观测来更新后续调优

表 2. 现有方法与 LIBRA 的比较。

方法	并行度搜索	收敛优化	GBS-Parallel 协同调优	在线调优
Alpa [43]	✓	✗	✗	✗
Aceso [23]	✓	✗	✗	✗
ASHA [20]	✗	✓	✗	✓
BOHB [10]	✗	✓	✗	✓
手工设计的渐进式训练 [25]	✗	✓	✗	✗
分支式训练 [27]	✗	✓	✗	✗
Pollux [33]	✗	✓	✗	✓
LIBRA（我们的方法）	✓	✓	✓	✓

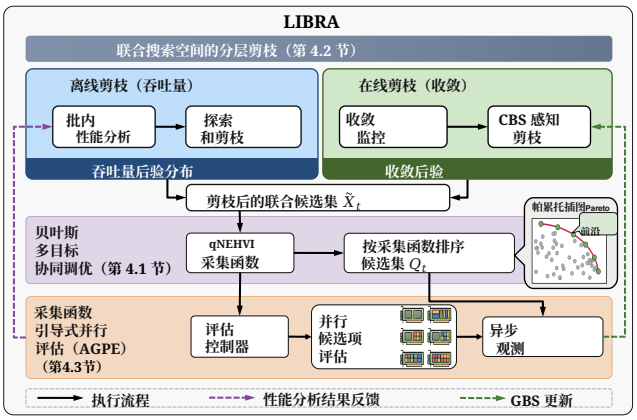


图 2. LIBRA 核心设计概览。

决策，但会保持并行配置固定。将这些方法用于并行配置搜索会引入成本高昂的集群级评估，因为每个候选项都要占用多个 GPU，且必须运行足够长的时间才能显现收敛差异。提前停止虽能缩短运行时长，却无法避免这些评估。

Pollux [33] 根据吞吐量和统计效率在线调整批大小与资源分配。然而，它既不搜索张量并行/流水线并行/上下文并行/微批次大小配置，也不将吞吐量和收敛作为独立目标加以保持。

手工与分支批次调优。 全局批次大小与并行配置协同调优会同时选择全局批次大小和并行配置。生产环境中的大语言模型训练通常会针对特定训练方案规定批大小计划，如 LLaMA 3 [25], 、DeepSeek-V3 [7], 和 MiniMax-01 [29]所示。这些计划为一个模型和训练方案选择批大小转换时机，同时保持并行配置不变，或在每次批大小变化后单独重新调优。分支式训练 [27]通过多次部分运行估算临界批次大小，但在集群规模扩大时，从共享检查点复制分支仍然代价高昂。这些方法使用固定计划或独立分支，而不是利用正在进行的训练运行中已完成的评估；两者都无法实现全局批次大小与并行配置协同调优。

总之，实现高训练效率需要一种轻量级运行时调优器，对吞吐量和收敛进行协同优化。

3.3 研究挑战

为了实现这一设计，LIBRA 必须缩小大规模联合域，并高效获取成本高昂的运行时观测数据。

并行度与批大小的庞大联合搜索空间。 GBS、MBS、DP、TP、PP 和 CP 的笛卡尔积受到约束，但仍具有组合爆炸性。在 LIBRA 采用的离散化方法下，该搜索空间在 64个 GPU 上约包含 150 万个配置，在 128个GPU 上约包含 190 万个配置。在对模型和允许范围进行完整参数扫描时，可行性筛选后，在 64个GPU 上剩余 1.1 万至 36 万个候选项，在 128个GPU 上剩余 1.5 万至 48 万个候选项。这些数量反映的是广泛扫描范围。实际评估域采用更窄的模型特定范围，包含 1.13 万至 2.46 万个可执行候选配置。此外，系统测量数据可以移除不可行或运行缓慢的配置，但吞吐量无法安全地淘汰某个批次，因为其收敛收益可能在后续发生变化。因此，系统侧和收敛侧的互补剪枝必须为贝叶斯优化器保留不确定性。

顺序配置评估的低效率。 评估一个候选项需要实测的系统行为以及用于观察收敛进展的有界训练窗口。不同评估所使用的 GPU数量和持续时间各不相同。顺序执行会将每项成本都添加到关键路径中，而固定数量的并发评估可能导致 GPU 碎片化，或运行低价值候选项。运行时调优必须优先处理信息量高的候选项，依据 GPU需求和估计评估时间对候选项进行打包，并异步返回观测。第4节通过分层剪枝应对第一个挑战，通过采集函数引导的并行评估应对第二个挑战。

4 吞吐量与收敛的迭代联合优化

LIBRA 将吞吐量和收敛保留为彼此独立的目标，分层剪枝配置空间，然后通过采集函数引导的并行评估 (AGPE) 评估候选项。图 2 展示了 LIBRA 的关键设计概览。

4.1 问题表述与贝叶斯优化

问题表述。 令 s_t 表示训练状态，x_t 表示在决策点 t选定的配置。LIBRA 按如下方式最小化达到目标验证损失 $\ell^\star$ 所需的时间：

$$T^\star = \min_{\{x_t\}} \inf\{\tau \geq 0 \mid \ell_{\text{val}}(s(\tau)) \leq \ell^\star\}, \quad x_t \in \mathcal{X}(s_t), \qquad (1)$$

其中，$s(\tau)$ 表示诱导出的训练轨迹，$\mathrm{X}(s_t)$ 表示可行集。对配置序列求解式 1 需要进行许多分支运行。LIBRA 则根据当前状态下的令牌吞吐量 f_{thr} 和每词元收敛进展 f_{conv} 做出局部决策。

候选项为 $x = (b, \pi)$，其中 b 为全局批次大小，$\pi = (DP, TP, PP, CP, MBS)$ 是其并行配置。候选项需满足设备、内存、可整除性和执行

约束条件。每个候选项都从一个共同的检查点 c_t 开始，并运行 N_x 个词元。LIBRA 在监控前后使用相同的验证数据测量验证损失，从而得到

$$\widehat{f}_{\text{conv}}(x \mid s_t) = \frac{\ell(c_t) - \ell(c_t^x)}{N_x}, \tag{2}$$

其中 c_t^x 是候选项 x 达到的检查点。每个优化器步处理 bN_{tok} 个词元，其中 N_{tok} 是序列长度。在固定 b 的情况下，改变布局会改变该步骤的耗时，同时保持更新中的样本不变。因此，LIBRA 在 c_t 处每经过一个全局批次大小就测量一次收敛情况。随着模型不断学习，后续检查点会提供新的观测。共享检查点和验证数据使观测结果具有可比性，而运行时变化和较短的时间范围会使两个目标都带有噪声。

贝叶斯多目标优化。 LIBRA 在有限集合 $\mathbf{X}_t$ 上执行多目标贝叶斯优化。已完成的评估构成 $\mathbf{D}_t = \{(x_i, \mathbf{y}_i, \Sigma_i)\}_{i=1}^{n_t}$，其中 $\mathbf{y}_i$ 包含两个目标，而 Σ_i 则是其观测噪声协方差。LIBRA 为吞吐量和收敛情况分别维护高斯过程后验。性能估计用于初始化前者，运行时监控则更新后者。有效吞吐量（吞吐量与统计效率的乘积） [33], 提供了一种固定的标量化方法，而 NSGA-II [6]和 SPEA2 [44] 等基于帕累托的进化方法需要进行大量评估。在贝叶斯方法中，ParEGO [18] 仍然优化采样得到的标量化目标，而标准 EHVI [9] 无法处理带噪声的观测或并行选择。并行带噪声期望超体积改进（qNEHVI） [5] 兼顾这两者，适用于 LIBRA 中代价高昂、带噪声且并发的运行时评估。

非支配观测构成帕累托前沿 $\mathbf{P}_t$。LIBRA 使用 qNEHVI 选择有望扩展此前沿的候选项，同时考虑观测噪声和后验不确定性。对于一批包含 X 个候选项的批次，

$$\alpha_{q\text{NEHVI}}(X) = \mathbb{E}_{\mathbf{f}_t \mid \mathcal{D}_t}\big[\text{HV}_\mathbf{r}(\mathcal{P}_t(\mathbf{f}_t) \cup \mathbf{f}_t(X)) - \text{HV}_\mathbf{r}(\mathcal{P}_t(\mathbf{f}_t))\big], \tag{3}$$

其中，$\mathbf{f}_t$ 是一个后验函数样本，$\mathbf{P}_t(\mathbf{f}_t)$ 是其非支配前沿，而 $\text{HV}_\mathbf{r}(\cdot)$ 表示相对于参考点 $\mathbf{r}$ 的超体积。LIBRA 使用蒙特卡洛样本估算公式 3，并根据边际 qNEHVI 值，从经过剪枝的~集合 $\mathbf{X}_t$ 中贪心地批量选择候选项。运行中的评估会保持待处理状态，以避免对同一区域重复采样。

qNEHVI 对评估进行排序，而不是选择实际部署的配置。设 $\Delta\ell_t$ 为当前验证损失到下一个损失里程碑之间固定的损失递减量，在 s_t 处由所有候选项共享。对于后验样本 j，LIBRA 将已评估候选项的本地时间估计为

$$\widehat{T}_t^{(j)}(x) = t_{\text{reconf}}(x) + \frac{\Delta\ell_t}{f_{\text{thr}}^{(j)}(x) f_{\text{conv}}^{(j)}(x)}, \tag{4}$$

其中，$t_{\text{reconf}}(x)$ 表示检查点和重新配置时间，对于 x_t 而言该值为零。如果 $f_{(j)\text{conv}}(x) \le 0$，LIBRA 将 $\widehat{T}_{(j)t}(x) = +\infty$，因此

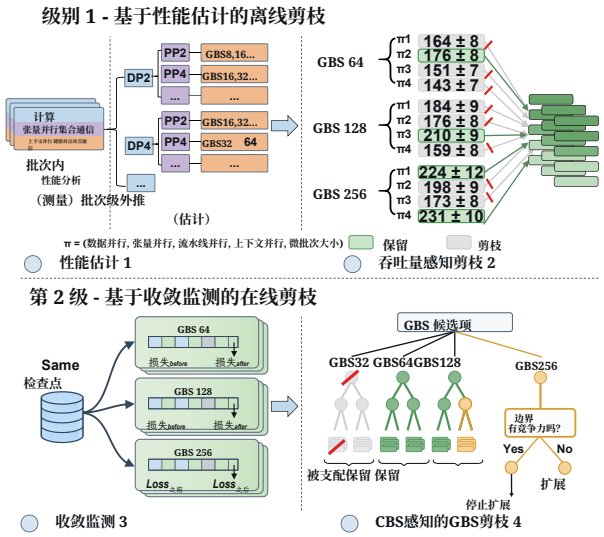


图 3. LIBRA 中分层剪枝的概览。第一层针对每种全局批次大小和 GPU 数量对布局进行排名。第二层则从一个共同的检查点开始，跟踪不同批大小的进展。

该样本无法支持切换。对于经过评估且未被支配的候选项，每个样本都会给出时间缩减 $\Delta T_{(j)t}(x) = \widehat{T}_{(j)t}(x_t) - \widehat{T}_{(j)t}(x)$。控制器仅在 $\Delta T_t(x)$ 的第五后验百分位数（即 95%下界）为正时才进行切换。否则，它会保留 x_t。公式 4 中的乘积给出了在候选项部署吞吐量下每秒预期的损失下降量。即使每秒处理的词元更多，较大的全局批次大小也可能需要更长时间才能达到下一个里程碑。在下一个决策点，LIBRA 会根据所选的训练状态更新估计值。

4.2 联合搜索空间的分层剪枝

将 qNEHVI 应用于 GBS 与并行配置的笛卡尔积并不实际。因此，LIBRA 采用图 3 所示的两个剪枝级别。级别 1 估算性能，并在每个 GBS 内对并行配置进行剪枝。第 2 级监控收敛情况，并对有序 GBS 池进行剪枝。它们会保留不确定的候选项，以供 qNEHVI 选择。

通过性能估计进行离线剪枝。 LIBRA枚举满足 $DP \cdot TP \cdot PP \cdot CP \le W$的候选项，其中$W$为GPU数量，并要求 b 可被 $DP \cdot MBS$整除。在性能分析之前，带有安全裕量的内存上限会移除容易发生内存溢出（OOM）故障的配置。设 $N_{\text{stage}}(PP)$和 $L_{\text{stage}}(PP)$分别为每个流水线阶段的最大参数数量和层数量，S 和 H 分别为序列长度和隐藏层大小，而 b_p、b_g、b_o、b_m和 b_a 分别为参数、梯度、优化器状态、主权重和激活值的存储开销。LIBRA估计

$$\widehat{M}_{\text{peak}}(x) = \frac{N_{\text{stage}}(PP)}{TP}\left(b_p + \frac{b_g}{z_g(x)} + \frac{b_o}{z_o(x)} + \frac{b_m}{z_m(x)}\right) + MBS \cdot \frac{S}{CP} \cdot H \cdot L_{\text{stage}}(PP) \cdot \frac{\kappa(TP, CP)}{TP} \cdot b_a, \tag{5}$$

表 3.不同工作负载下可行搜索空间与测量配置的比较。

工作负载	搜索空间	测量配置	覆盖率
7B@64 个 GPU	19,872	84	0.42%
13B@64 个 GPU	11,264	68	0.60%
34B@128 个 GPU	15,360	92	0.60%
700亿@256个GPU	24,576	98	0.40%

其中，$z_g(x)$、$z_o(x)$和 $z_m(x)$分别是梯度、优化器状态和主权重的数据并行分片因子。复制时各值为1，分片时为 DP。$\kappa(TP,\ CP)$涵盖激活检查点、序列并行和临时缓冲区。利用公式 5，LIBRA直接纳入满足且 $M_{\text{peak}}(x) \le 0.9M_{\text{GPU}}$的布局，其中 M_{GPU} 为内存容量。在451次启动中，配置均保持内存溢出安全。在接近设备容量时，临时缓冲区和在途微批次的数量可能决定某种布局是否适配。LIBRA可以在所提议的微批次大小下，通过完成完整的优化器步来解决这些情况：填满流水线，并记录每个等级上的峰值内存。完成这些步的布局将进入吞吐量比较。

批内分析与批次级外推。 图 3 中的模块 1 测量 $(TP,\ CP,\ MBS)$ 配置。性能分析记录最小可行 GPU 组上的微批次计算、张量并行集合通信以及上下文并行键值（KV）交换延迟。LIBRA 对每项性能分析重复 3–5 次，丢弃预热步骤，并估算均值和方差。共享 $(TP,\ CP,\ MBS)$ 元组的候选项复用此性能分析结果。

表 3 显示，稀疏分析测量了 68–98 个代表配置，占可行搜索空间的 0.40–0.60%。

LIBRA 在模块 1 中对 DP、PP 和全局批次大小分支的性能分析结果进行外推。令 t_{MB} 表示在性能分析所用的张量并行和上下文并行度下，所有层中一个微批次的前向与反向传播时间；令 $G = b/(DP \cdot MBS)$ 表示每个优化器步包含的微批次数。在均衡的一前向一反向（1F1B）调度下，一个阶段处理每个微批次需要 t_{MB}/PP。填充和排空流水线会增加 $PP-1$ 个时隙，因此得到 $T_{\text{comp}}(x) = (G + PP - 1)t_{\text{MB}}/PP$。将其与流水线通信、数据并行通信和优化器耗时相结合，可得

$$\widehat{f}_{\text{thr}}(x) = \frac{b\,N_{\text{tok}}}{\widehat{T}_{\text{comp}}(x) + \widehat{T}_{\text{pp}}(x) + \widehat{T}_{\text{dp}}(x) + \widehat{t}_{\text{opt}}(x)}, \tag{6}$$

其中，N_{tok} 表示每个序列的词元数，T_{pp} 和 T_{dp} 分别表示流水线并行和数据并行通信耗时，$\widehat{t}_{\text{opt}}$ 表示优化器耗时。近期执行会校准这些项以及吞吐量方差。对于每个 GPU数量，LIBRA 会针对具有最大可行数据并行度的布局，测量完整优化器步的耗时。这些步骤包括梯度累积、通信和优化器更新。对于固定布局，GBS 决定累积微批次的数量 G。因此，一次步耗时测量即可支持多个 GBS 估计值。

吞吐量感知剪枝。 模块 2 使用公式 6 对相同 GPU数量和 GBS 下的布局进行排序。每个框显示以 K 词元 / 秒为单位的预测吞吐量，作为均值 ± 一个标准差。LIBRA 保留最佳配置，以及区间与最佳区间重叠的配置。它会剪除上界低于最佳配置的配置，从而保留具有吞吐量竞争力的配置以供监控。如果能够使用更大的 MBS，LIBRA 会探索更高的 TP、CP 或 PP 度数。如果最大可行 MBS 保持不变，则停止扩展该度数。当 MBS 增大时，LIBRA 会将均值吞吐量与相同 GPU数量下、每个共同 GBS 对应的已找到最佳布局进行比较。如果新布局至少赢得一次比较，搜索就会沿该方向继续。最终生成的表格会为 AGPE 提供每个可行 GPU数量与 GBS 组合对应的布局。

通过收敛监控进行在线迭代剪枝。级别 1 之后，LIBRA 将剩余的全局批次大小值作为有序池维护。

收敛监测。 模块 3 会从同一检查点出发，为每个选定的全局批次大小启动一个分支。检查点包含相同的模型、优化器、调度器和已消耗令牌状态。一个监控窗口涵盖当前最优解及更大的相邻批次。每个分支使用为其 GPU 组选择的布局。评估会运行一个较短的令牌时域，或持续到达到固定的损失递减量。图 3 展示了令牌时域的情况，其中所有分支处理相同数量的令牌，并记录窗口开始前后的损失。这些测量结果会更新公式 2 的后验分布。

监控窗口能够揭示损失进展的差异，但其时长短于一个训练阶段。例如，一个序列长度为 4,096、全局批次大小为 8 的 LLaMA-2 13B 候选项，每个窗口会处理 330 万至 660 万个令牌。使用相同的训练状态和停止规则，可使不同全局批次大小和并行配置下的每令牌进展具有可比性。

*CBS*感知的*GBS*剪枝。 模块 4 在有序 GBS 池中比较级别 1 的吞吐量后验分布与模块 3 的收敛后验分布。CBS 的关键洞见是：随着 GBS 增大，每个词元的进展会减小，而有效批次区间会向上移动。LIBRA 利用这一洞见进行边界剪枝，而无需估计 CBS，并将最大的活动 GBS 视为边界。只有当某个 GBS 的收敛上界低于最近保留 GBS 的下界，且其吞吐量上界相较于该 GBS 的增益不足 3% 时，联合置信度检验才会剪除该 GBS。具有帕累托竞争力或存在不确定性的 GBS 值会被保留。在停止边界候选项或向上扩展之前，该条件必须在连续两个监控窗口内成立。如果某个较小的 GBS 在达到下一个里程碑所需的时间上输给较大的邻近 GBS，则可以将其移出池。具有更高吞吐量的较大邻近 GBS 仍有资格在稍后的检查点参与比较，因为相对于当前最优解，其每词元进展可能会有所改善。完整的 GPU 分配为该搜索设定了上限。一旦 GBS 加倍相较于前一个 GBS 仅带来边际吞吐量增益，则该前一个 GBS 就是最后一个需要监控的

在前一个全局批次大小下的布局，该前一个全局批次大小是上一个要监控的对象。

在图 3 中，为便于阅读，模块 3 展示了三个监控分支，而模块 4 也对 GBS 32 使用同一检查点观测。GBS 32 被剪枝，GBS 64 和 128 被保留，GBS 256 则是边界。在此限制以下具有竞争力的边界会接纳下一个可行的 GBS。保留的相邻项可以重新评估，而无需重新打开整个搜索空间。重复此过程即可逐步增大批次大小，无需预先设定调度计划。控制器比较完最大的批次，或 CBS 测试停止扩展后，固定训练便会使用选定的 GBS 及其性能分析得到的布局，在完整的 GPU 分配上进行。最深的兼容检查点可让选定配置恢复运行，而无需重复监控窗口。

4.3 采集函数引导的并行评估（AGPE）

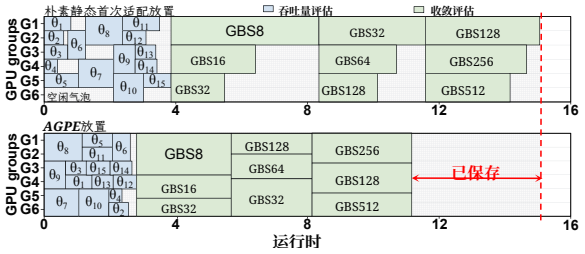
第 4.1 节中的贝叶斯协同调优器使用 qNEHVI 生成候选集 Q_t 和边际采集值 $a_t(x)$，决定评估什么，但不决定如何将评估映射到 GPU。对于每个就绪候选项 x，$w(x)$ 是其 GPU 需求，而 $d_t(x)$ 是其估计评估时间。两者的乘积 $\widehat{c}_t(x) = w(x)d_t(x)$ 是估计 GPU 时间成本。AGPE 使用 $p_t(x) = a_t(x)/\widehat{c}_t(x)$ 作为调度优先级。每次调度事件都会通过求解以下问题来选出一个可行子集

$$\begin{aligned} \max_{z_x \in \{0,1\}} \quad & \sum_{x \in Q_t} p_t(x) z_x \\ \text{s.t.} \quad & \sum_{x \in Q_t} w(x) z_x \le W_{\text{free}}, \end{aligned} \tag{7}$$

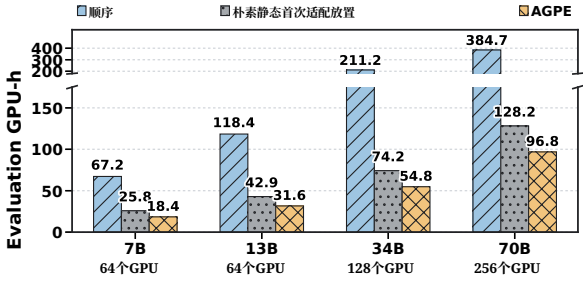
其中 z_x 是二元准入变量，Q_t 是就绪队列，W_{free} 是空闲 GPU 的数量。拓扑结构和可整除性会限制放置方案。公式 7 使用 qNEHVI 的采集函数值，同时考虑获取观测结果所需的预期 GPU 时间。

AGPE 根据性能分析步数和近期步时间测量值估算吞吐量评估时间。对于固定的 token 处理规模，它通过将该规模除以一级吞吐量估计来估算收敛评估时间。近期监控窗口会更新在损失递减量达到固定值时停止的评估的时间估计。准入后，AGPE 使用 $w(x)$ 和 $d_t(x)$ 比较可行放置方案，并选择能够平衡预测完成时间的放置方案。评估完成后会更新时间估计。

算法 1 将按采集函数排序的提案转换为异步执行。在每个调度事件中，AGPE 将运行中的评估作为待处理项报告给 qNEHVI，估算每个提案的 GPU 需求和评估时间，并将优先级最高的提案装入空闲 GPU 组（第 3–8 行）。图 4(a) 展示了与朴素静态首次适配放置的示意性比较，后者按到达顺序放置评估任务，而不使用预估的评估时间。它展示了 AGPE 如何利用 GPU 需求和估计的评估时间来减少 GPU 空闲气泡，而具体的放置则在运行时确定。



(a) 评估放置。



(b) 评估成本。

图 4. 评估放置与成本。(a) 示意 AGPE 如何相较于朴素静态首次适配放置减少 GPU空闲气泡。(b) 顺序评估、朴素静态首次适配放置和 AGPE 的总 GPU 小时成本。

算法 1： 我们提出的 AGPE 方法。

输入： 协同调优器 B_t，候选池 $\widetilde{X}_t$，数据 D_t，检查点 c_t，GPU 组 G

```
R ← ∅                                              // 运行中的评估
当 调优活动状态(s_t) 执行
    P ←候选项(R)                                    // 待处理点
    (Q, a) ← 提议(B_t, X̃_t, D_t, P)
    (w, d̂) ← 估算资源(Q, D_t)
    A ← 采集包(Q, a, w, d̂, 空闲(G))
    遍历 (x, g) ∈ A 执行
        R ← R ∪ {启动(x, g, c_t)}
    如果 A = ∅ 则
        中断
    e ←等待下一个事件(R)                            // 监控或终端事件
    (x, y Σ, 状态) ← 观测(e)
    若状态 ∉ {失败,超时} 则
        D_t ← D_t ∪ {(x, y Σ)}
        B_t ← 更新(B_t, D_t)
        X̃_t ←保留-剪枝-扩展(X̃_t, x, D_t)
        若状态 =运行中∧
          收敛性评估(x) ∧ CBS停止(x, D_t) 则
            终止(e); 状态 ← 已停止
    如果状态 ∈{完成,已停止,失败,超时} 则
        R ← R \ {e}; 释放(e, G)
返回 B_t, X_t, D_t
```

表4。 LIBRA 的运行时接口。

Role	信息
用户输入	训练任务、GPU分配、可行参数范围、目标验证损失
固定配置	模型、数据顺序、优化器、学习率调度器、序列长度
决策	全局批次大小和DP/TP/PP/CP/MBS配置
观测	吞吐量、监测到的损失进展、资源需求、完成状态
持久化状态	候选队列、后验数据、当前最优解、检查点元数据

AGPE 等待下一个监控事件或终止事件，并记录其噪声观测（第 11–20 行）。监控事件会更新后验分布和候选池，同时评估仍在进行。终止事件会释放其资源。下一次迭代会利用新数据重新计算评估时间估计（第 5 行），然后再次进行调度，而无需等待固定时长的轮次。

每个窗口结束后，CBS 感知联合置信度检验会保留、剪枝或扩展有序 GBS 池，并应用提前停止策略（第 16–20 行）。例如，当两个 GPU 组空闲时，AGPE 可以并行运行当前最优解对应的 GBS 及其下一个更大的邻近值。每个组的吞吐量决定其评估占用这些 GPU 的时长。为了比较用于训练的两个批次，LIBRA 将每个分支实测的每个词元损失下降量与其提议的部署布局的吞吐量结合起来，代入公式 4。各组的评估窗口结束后，便可用于新的评估。

图 4(b) 展示了吞吐量和收敛评估。AGPE 使用 18.4–96.8 GPU 小时，而朴素静态首次适配放置使用 25.8–128.2 GPU 小时，顺序评估则使用 67.2–384.7 GPU 小时。顺序基线会完成每项评估，不进行剪枝或提前停止。朴素静态首次适配放置采用与 AGPE 相同的更新，但按队列顺序启动候选项。它可能在观测将候选项从队列中移除之前就启动这些候选项。AGPE 会更早返回高优先级观测，在启动前移除候选项，并停止表现较差的评估。基于估计时间的放置解释了图 4(a) 中较短的完成时间，而较少的评估次数和提前停止则解释了图 4(b) 中 GPU 小时数的减少。

5 系统实现

本节介绍 LIBRA 如何将优化决策转化为可执行训练作业，并将其观测纳入正在进行的运行中。

运行时控制工作流。 LIBRA 是位于 Megatron-LM [31, 37] 等训练框架之上的外部控制平面。用户提供训练规范、集群描述、全局批次大小以及并行度范围，但不提供固定配置或渐增调度计划。LIBRA 会验证输入，并且仅在评估和重新配置边界处将决策转换为可执行作业。表 4 总结了该接口及其运行时状态。

候选构造。 每个保留点 $x = (b,\ \pi)$ 都会转换为一条记录，其中包含 GPU需求、评估类型、优先级、检查点和观测。构造过程会针对 GPU 组大小 w_x，强制满足 DP $\times$TP $\times$PP$\times$ CP $= w_x$，以及全局批次大小、分片和内存约束条件。共享同一 $(TP,\ CP,\ MBS)$ 元组的候选项会复用同一份性能分析结果。对于收敛评估，控制器会根据提议的全局批次大小和组大小查找选定布局。在所有 GPU 上部署时，相同的全局批次大小也可能采用不同的布局。

运行时控制与重配置。AGPE 使用采集值、GPU需求和估计评估时间，将候选项分配到互不重叠的 GPU 组，并通过 Megatron-LM 启动它们。任务完成后，系统会异步更新剪枝、贝叶斯和时间估计模块，然后再进行下一轮调度。在线候选项从同一检查点开始，以确保收敛观测具有可比性。候选项一旦通过公式 4 中的保守规则，控制器便会在优化器步数边界处切换。控制器从选定分支上最深的兼容检查点恢复，其中包含监控期间完成的更新。并行布局发生变化时，模型状态和优化器状态会重新分配到新的秩上。数据迭代器和学习率调度从已保存的训练进度继续。只有选定分支的词元会推进正在进行的训练。如果当前最优解胜出，训练便从其自身的终点继续。

其他细节。LIBRA 在 Megatron-LM 之上由约 7K 行 Python 代码构成。它利用后端计时器和通信统计信息，通过 Megatron-LM 入口点启动保留的配置，不触碰模型算子、集合通信操作、流水线调度、优化器和学习率调度，并将其搜索状态与检查点一同保存。评估失败或超时后会释放其 GPU，且不更新后验分布；而提前停止的评估则会保留其观测。如果没有候选项通过保守的切换规则，训练将继续使用当前配置。

6 评估

我们在不同模型和 GPU 规模下对 LIBRA 进行端到端评估，并分别评估各组件的效果与开销。

6.1 实验设置

硬件和软件。实验使用了海光 AMD GPU 集群中的 1–32 个节点，共计 8–256 个GPU。每个节点配备两颗 64 核海光 C86 CPU 和八个拥有 64 GB 内存的海光 AMD GPU，基于 gfx936 架构，包含80个计算单元。GPU 通过 PCIe Gen5 x16 和全互连的海光 hyswitch 互连网络连接。每个节点配备四个 400 Gbps InfiniBand 网卡。软件栈包括 Anolis

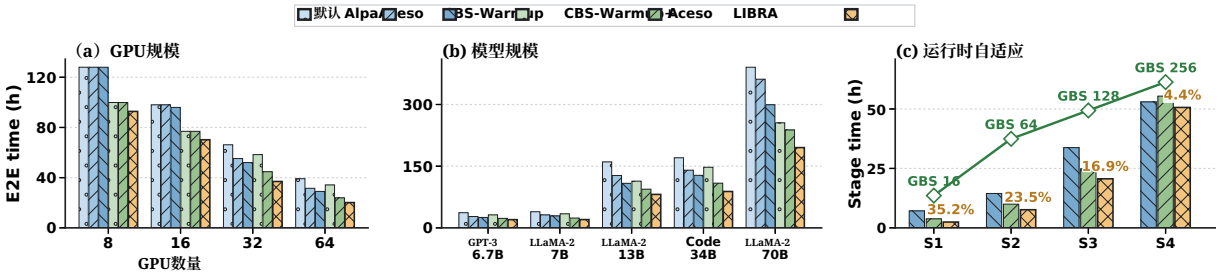


图 5. 达到目标验证损失所需的端到端时间及运行时自适应。(a) 不同 GPU数量下的 LLaMA-2 7B。(b) 在 64/64/64/128/256 个 GPU 上运行的不同模型。(c) 不同阶段的阶段时间及 GBS 的变化情况。

OS 8.9、兼容 ROCm 6.3 的 GPU 运行时、RCCL 2.22.3、PyTorch 2.4.1 和 Megatron-LM 0.12.0。所有实验均在该海光 AMD GPU 平台上运行，而本文其余部分使用 GPU 作为加速器的术语。LIBRA 并不绑定于该硬件。运行时性能分析会在部署平台上重新校准其性能估计，而不是依赖固定的硬件模型。

模型和数据集。 我们在 OpenWebText 上训练 GPT-3 6.7B、LLaMA-2 7B/130亿/700亿以及 Code LLaMA 34B。GPT-3 的序列长度为 1,024，LLaMA-2 为 4,096，Code LLaMA 为 4,096。GPT-3 的目标验证损失为 2.45，LLaMA-2 7B、13B 和 70B 的目标验证损失分别为 2.37、2.26 和 2.22，Code LLaMA 的目标验证损失为 2.25。

基线方法。 基线方法涵盖专家实践、并行调优、批次自适应和联合调优。所有方法使用相同的候选空间和 Megatron-LM [37] 后端。(i) **默认方法**采用按模型设定的批次策略和一种手动配置的并行配置。(ii) **Alpa** [43] 将该策略与算子间及算子内并行搜索相结合。(iii) **Aceso** [23] 使用瓶颈缓解进行面向吞吐量的并行搜索。(iv)**CBS-Warmup**基于 Merrill 等人 [27], 的方法，在临界批次大小超过当前批次大小的两倍时，将全局批次大小加倍，同时固定并行配置。(v) **CBS-Warmup+Aceso** 在每个全局批次大小下采用 Aceso 的吞吐量最优配置。

组合测试旨在检验将批次调优器与并行调优器结合起来是否能解释 LIBRA 的性能提升。所有方法使用相同的初始化、数据顺序、优化器、训练方案、资源分配、约束条件和目标损失。端到端时间包括在线性能分析、评估和重新配置，但不包括离线调度和配置构建，这对基线方法更有利。

对于第 6.2 节中的超参数优化比较，我们在离线剪枝后，将 ASHA [20] 和 BOHB [10] 调整到相同的联合域。每个候选项由一个全局批次大小和一个并行配置组成。监控词元数定义保真度，单位时间内的进度则作为晋级信号。为单独考察搜索策略，所有方法都采用相同的 GPU 组放置、异步反馈和保真度限制。

训练设置。所有运行都从相同的模型状态开始，并从头训练。各方法仅在同一搜索空间内采用不同的全局批次大小和并行配置，前提是
模型、内存、可整除性和资源分配约束。优化器、学习率调度、序列长度、数据

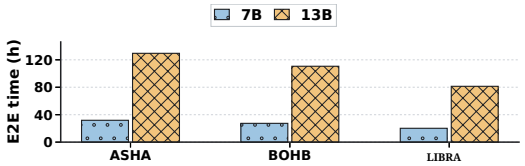


图 6。超参数优化基线方法与 LIBRA 的比较。

顺序、验证频率、目标损失和检查点状态保持不变。我们将端到端时间定义为从作业启动到验证首次达到目标所用的时间，其中包括在线分析、收敛监测、候选调度、检查点 I/O 和重新配置。吞吐量不包括预热步骤，并采用训练循环的令牌统计。

6.2 端到端收敛时间

图 5(a) 和 (b) 涵盖 8–64个GPU，以及从 GPT-3 6.7B 到 LLaMA-2 70B 的模型。在不同设置下，LIBRA 相较于 Alpa 和 Aceso 的降幅平均为 31.1%。相较于 Alpa，在 8–64个GPU 范围内，降幅从 27.5% 增至 36.0%；在不同模型规模下，降幅从 27.6% 增至 46.0%。相较于 Aceso，不同模型规模下的降幅为 21.2% 至 34.8%。对于 GPT-3 6.7B，LIBRA 使用 14.12B 个令牌，在 20.02 小时内达到目标；而 Alpa 用时 27.65 小时，消耗 24.09B 个令牌，尽管 LIBRA 的吞吐量更低（196K 对比 242K 令牌/秒）。LIBRA 相较于最佳替代方案的优势，从使用 8–16个 GPU 时的 7.1–8.7% 增至使用 32–64个GPU 时的 15.6–17.4%；在所有基线方法均可用的模型规模中，优势范围为 13.4–18.6%。因此，单靠并行调优或独立的批次自适应都无法解释这一提升。图 5(c) 考察了一次 130亿模型的运行。每个阶段都是一个验证进度区间，由 LIBRA 的重新配置点划定。柱形图比较各方法遍历同一区间所需的时间，折线则追踪 LIBRA 的全局批次大小。随着全局批次大小从 16 增至 256，LIBRA 相较于最佳替代方案的阶段时间分别缩短 35.2%、23.5%、16.9% 和 4.4%。因此，其最大收益出现在进入最终的大批量区间之前。

图 6 比较了相同联合域中的 HPO基线和 LIBRA。ASHA、BOHB 和 LIBRA 在 7B 模型上分别用时 31.83/27.37/20.17 小时达到目标，在 13 B 模型上分别用时 129.47/110.62/81.37 小时。BOHB 通过基于模型的提议改进了 ASHA，但二者都在更广泛的候选集上使用标量晋级信号。

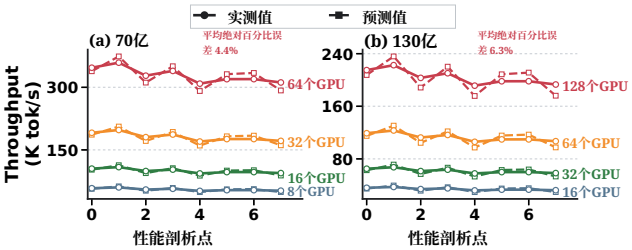


图 7. LIBRA 的性能估计 （第 4.2 节）中预测吞吐量与实测吞吐量的比较。

表 5.分层剪枝和 AGPE 后的搜索空间大小与评估成本。Full、Feasible、O1 和 O2 分别表示未筛选的空间，以及经过可行性检查、离线剪枝和在线剪枝后的空间。成本以 GPU 小时计；Reduction 表示相对于顺序评估的减少量。

模型 size	搜索空间大小 （完整/可行/O1/O2）	评估成本 序列 LIBRA		
7B	1.52M/19,872/96/7	67.2	18.4	72.6%
13B	1.52M/11,264/72/6	118.4	31.6	73.3%
34B	1.91M/15,360/88/7	211.2	54.8	74.1%
70B	2.44M/24,576/96/8	384.7	96.8	74.8%

基于模型的提议，但二者都在更广泛的候选集上使用标量晋级信号。LIBRA 将双目标采集函数与分层剪枝相结合，因此即使 HPO基线方法保留了其并行执行层，也能更快达到目标。

6.3 性能预测的准确性

图7比较了实测值与预测值的扩展吞吐量。预测曲线遵循实测的扩展趋势：对于70亿模型，在8–64个GPU范围内的平均绝对百分比误差（MAPE）为4.4%；对于130亿模型，在16–128个GPU范围内为6.3%。这种准确性使离线剪枝能够对未实测的批量级配置进行排序，同时保留不确定性与最佳估计值重叠的配置。随着GPU数量增加，误差仍保持较低水平，表明外推在评估的扩展范围内保持了准确性。

表5展示了候选项经过两个剪枝层级的过程。可行性检查将152–244万个完整元组减少至1.13–2.46万个可行候选项。性能剖析复用和性能外推在O1之后保留72–96个吞吐量竞争性配置，而收敛监测在O2之后将其减少至6–8个。LIBRA仅在7–9个并行轮次中测量68–98个性能剖析结果。分层剪枝、性能剖析复用、AGPE和提前停止共同使评估GPU小时数较顺序评估节省72.6–74.8%。这些减少是累积的：可行性检查移除无效配置，离线剪枝避免测量大多数配置，而在线剪枝将保留的联合域缩减至6–8个候选项。在不同工作负载下保持稳定的节省表明，随着模型和GPU规模增长，该层级结构能够控制评估成本。

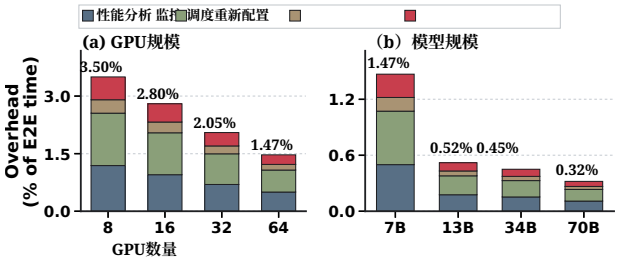


图8。LIBRA的运行时开销，测得为附加—不同模型和GPU数量下，关键路径时间占端到端训练时间的比例。条形图显示了性能分析、监控、调度和重新配置所占的时间。

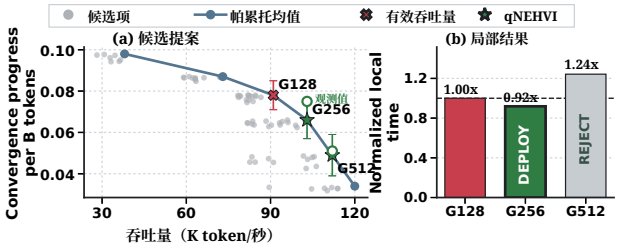


图 9. 两种多目标优化方法：qNEHVI 和有效吞吐量的比较。(a) 两种方法选出的配置均位于帕累托前沿上。(b) 检查点处观测候选项的归一化本地时间。

6.4 LIBRA 的开销

图8报告了性能分析、监控、调度和重新配置所增加的关键路径时间，占端到端训练时间的百分比。控制器会记录某个模块延迟训练或重新配置操作的时间区间，但不计入与其他GPU组上的执行过程重叠的工作。该指标不同于图4(b)和表5中的评估GPU小时数；后者衡量的是GPU使用量，而不是并发评估重叠后关键路径上剩余的时间。对于70亿模型，开销从使用8个GPU时的3.5%降至使用64个GPU时的1.47%。在各模型规模设置下，70亿/130亿/340亿/700亿模型对应的开销分别为1.47%/0.52%/0.45%/0.32%。有界窗口和并发放置使成本保持较低，并且所有端到端结果都包含该开销。更大规模下占比下降表明，LIBRA的运行时控制不会成为可扩展性瓶颈。

此外，LIBRA 的内存占用仅限于主机端搜索和检查点元数据；由于分层剪枝限制了保留候选集的规模，每次运行的内存占用均低于 51.8 MiB。评估复用 Megatron-LM 的分配器，不会增加持久化设备内存分配。

6.5 消融实验

多目标采集。 图 9 比较了在 130亿检查点处使用有效吞吐量和 qNEHVI 进行候选选择的结果。有效吞吐量选择了 GBS 128，而 qNEHVI 还提议了 GBS 256 和 512。这一差异源于两种方法利用吞吐量和收敛的方式不同。有效吞吐量通过统计效率将二者合并为

单一分数，预先确定二者之间的权衡。相比之下，qNEHVI 将这两个目标分开处理，并在选择候选项时考虑不确定性。短期监控表明，与 GBS 128 相比，GBS 256 和 512 分别需要 0.92× 和 1.24× 的本地时间（公式 4，已针对 GBS 128 进行归一化）。因此，控制器部署 GBS 256，并拒绝 GBS 512。尽管这些差异并不显著，但将两个目标分开处理，能够区分有益的候选项和有效吞吐量无法识别的较慢候选项。将 qNEHVI 替换为有效吞吐量后，7B 和 130亿的端到端时间分别增加了 8.1% 和 10.1%（图 10）。

组件移除。 图 10 对比了完整系统与五种变体，每种变体都移除了一种机制。“无 qNEHVI”用有效吞吐量替代 qNEHVI，“无离线剪枝”移除性能剪枝，“无在线剪枝”移除 CBS 感知剪枝，“无 AGPE”按顺序评估候选项，“无提前停止”则完成每个监控窗口。这些变体分别使端到端时间增加 8.1–10.1%、11.1–14.2%、20.5–23.8%、17.0–18.1% 和 12.6–14.8%。

CBS 感知的在线剪枝影响最大。若没有这一机制，每令牌进展很小的 GBS 可能会在后续检查点再次接受测试。由于离线剪枝比较的是固定 GBS 下的吞吐量，因此最快的布局仍可能通过离线剪枝。提前停止则能在当前监控窗口内释放弱分支占用的 GPU。在 70亿和 130亿工作负载上，移除任何一种机制都会使端到端时间至少增加 8%。

7 相关工作

面向吞吐量的并行调优。 分布式训练系统会自动规划并行化、放置，以及内存与通信之间的权衡。FlexFlow [15] 通过模拟器引导搜索来探索算子级并行化。Alpa [43]将算子内分片与算子间流水线相结合。Aceso [23]通过缓解性能瓶颈来搜索多维并行配置，而 AutoDDL [4] 和 Hypertron [21] 则利用解析模型探索更广阔的空间。这些系统共同提升了固定训练计算任务的执行吞吐量。然而，它们不会利用收敛反馈来决定何时应改变全局批次大小。LIBRA 通过联合调整批大小和并行配置来补充这些系统，因为吞吐量最优的配置未必能使收敛时间最短。

超参数优化与在线调工程。 多保真超参数优化通过在不同配置之间分配部分训练资源来降低调优成本。逐次减半 [14], Hyperband [19],和 ASHA [20] 会提前终止表现不佳的配置，并提升有潜力的配置。BOHB [10] 和 DEHB [1] 将多保真资源分配与基于模型的搜索或进化搜索相结合，而 FABOLAS [17], BOCA [16], 和轨迹感知优化 [41] 则利用低保真观测来指导选择。Ray Tune [22] 提供分布式执行

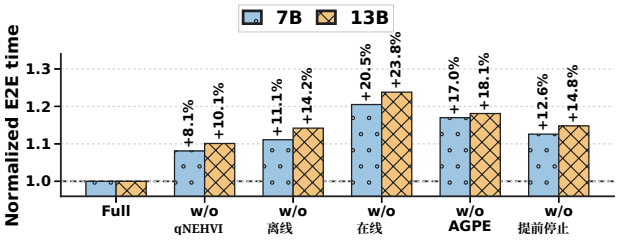


图 10。LLaMA-2 7B 和 130亿的消融实验。

以支持此类方法。这些方法将候选项作为独立的训练运行进行评估。对于大语言模型训练，每个批量大小候选项都映射到多个并行配置，使超参数优化成本高昂。低保真观测可以降低成本，但无法消除这种并行配置搜索。LIBRA使用 qNEHVI，但其优化围绕分层剪枝以及正在进行的运行内的并发评估展开，而不是启动完整运行或按轮次运行。

8 结论与未来工作

我们提出了 LIBRA，这是一种轻量级运行时框架，可联合调整批大小和并行配置，以最大限度缩短收敛时间。LIBRA 将基于 qNEHVI 的多目标协同调优、分层剪枝和 AGPE 相结合，使运行时候选搜索与评估切实可行。在涵盖多种大语言模型、集群规模最高达 256个GPU 的实验中，与最先进方法相比，LIBRA 平均将端到端收敛时间缩短了 31.1%，而其运行时调优仅占端到端训练时间的 0.3–3.5%。这些结果表明，相较于孤立优化，联合优化吞吐量与收敛效果更有优势。

仍有若干方向值得进一步探索。我们的原型基于 Megatron-LM 构建，将控制器移植到其他训练框架，可以检验这一设计在不同运行时接口中的泛化能力。更广泛的评估——涵盖更多模型系列、更大规模的集群以及混合专家架构——将进一步揭示联合协同调优在哪些场景下最有益。

参考文献

[1] Noor Awad、Neeratyoy Mallik 和 Frank Hutter。2021 年。DEHB: 面向可扩展、稳健且高效的超参数优化的进化式 Hyperband。载于第三十届国际人工智能联合会议论文集（*IJCAI*）。2147–2153。doi:10.24963/ijcai.2021/296[2] Léon Bottou、Frank E. Curtis 和 Jorge Nocedal。2018 年。大规模机器学习优化方法。arXiv:1606.04838 [stat.ML]doi:10.48550/arXiv.1606.04838[3] Tom B. Brown、Benjamin Mann、Nick Ryder、Melanie Subbiah、Jared D. Kaplan、Prafulla Dhariwal、Arvind Neelakantan、Pranav Shyam、Girish Sastry、Amanda Askell、Sandhini Agarwal、Ariel Herbert-Voss、Gretchen Krueger、Tom Henighan、Rewon Child、Aditya Ramesh、Daniel M. Ziegler、Jeffrey Wu、Clemens Winter、Christopher Hesse、Mark Chen、Eric Sigler、Mateusz Litwin、Scott Gray、Benjamin Chess、Jack Clark、Christopher Berner、Sam McCandlish、Alec Radford、Ilya Sutskever 和 Dario Amodei。2020 年。语言模型是少样本学习者。载于神经信息处理系统进展，第 33 卷。1877–1901。doi:10.48550/arXiv.2005.14165

[4] Jinfan Chen、Shigang Li、Ran Guo、Jinhui Yuan 和 Torsten Hoefler。2024 年。AutoDDL：具有近似最优带宽成本的自动分布式深度学习。*IEEE* 并行与分布式系统汇刊 35, 8 2024 1331 1344 10 1109 2024 3397800 ( ), – 。doi: 。/ 2021 [5] TPDS。。Samuel Daulton、Maximilian Balandat 和 Eytan Bakshy。。具有期望超体积改进的多个含噪目标的并行贝叶斯优化。载于神经信息处理系统进展，第 34 卷。2187–2200。[6] Kalyanmoy Deb、Amrit Pratap、Sameer Agarwal 和 T. Meyarivan。2002 年。一种快速且精英保留的多目标遗传算法：NSGA-II。*IEEE* 进化计算汇刊 6，2（2002），182–197。doi:10.1109/4235.996017[7] DeepSeek-AI。2024 年。DeepSeek-V3 技术报告。*arXiv* 预印本 *arXiv*： *2412 19437* 2024 10 48550 2412 19437 。( )。 [8] doi: 。/arXiv。。Aditya Devarakonda、Maxim Naumov 和 Michael Garland。2017 年。AdaBatch：用于训练深度神经网络的自适应批大小。*arXiv* 预印本 *arXiv*： *1712 02029* 2017 10 48550 1712 02029 。( )。doi: 。/ [9] arXiv。。Michael T. M. Emmerich、André H. Deutz 和 Jan Willem Klinkenberg。2011 年。基于超体积的期望改进：单调性性质与精确计算。载于 *IEEE* 进化计算大会论文集。2147–2154。doi:10.1109/CEC.2011.5949880[10] Stefan Falkner、Aaron Klein 和 Frank Hutter。2018 年。BOHB：大规模场景下稳健且高效的超参数优化。载于第 *35* 届国际机器学习会议（*ICML*）论文集（机器学习研究论文集，第 *80* 卷）。PMLR，1437–1446。https://proceedings.mlr.press/v80/falkner18a.html [11] Gemini 团队。2023 年。Gemini：高能力多模态模型系列。*arXiv* 预印本 *arXiv:2312.11805*（2023）。doi:10.48550/arXiv.2312.11805[12] Yanping Huang、Youlong Cheng、Ankur Bapna、Orhan Firat、Mia Chen、Dehao Chen、HyoukJoong Lee、Jiquan Ngiam、Quoc V. Le 和 Yonghui Wu。2019 年。GPipe：利用流水线并行高效训练巨型神经网络。载于神经信息处理系统进展，第 32 卷。103–112。https://dl.acm.org/doi/10.5555/3454287.3454297[13] Sam Ade Jacobs、Masahiro Tanaka、Chengming Zhang、Minjia Zhang、Shuaiwen Leon Song、Samyam Rajbhandari 和 Yuxiong He。2023 年。DeepSpeed Ulysses：用于实现超长序列 Transformer 模型训练的系统优化。*arXiv* 预印本 *arXiv*： *2309 14509* 2023 10 48550 2309 14509 。( )。doi: 。/ arXiv。 [14] 。Kevin Jamieson 和 Ameet Talwalkar 2016 。。非随机最佳臂识别与超参数优化。载于第 *19* 届人工智能与统计学国际会议（*AISTATS*）论文集（机器学习研究论文集，第 *51* 卷）。240–248。https://proceedings.mlr.press/v51/jamieson16.html [15] Zhihao Jia、Matei Zaharia 和 Alex Aiken。2019 年。超越深度神经网络的数据并行与模型并行。载于机器学习与系统论文集（*MLSys*），第 1 卷。1–13。https://proceedings.mlsys.org/paper_files/paper/2019/hash/b422680f3db0986ddd7f8f126baaf0fa-Abstract.html [16] Kirthevasan Kandasamy、Gautam Dasarathy、Jeff Schneider 和 Barnabás Póczos。2017 年。采用连续近似的多保真贝叶斯优化。载于第 *34* 届国际机器学习会议（*ICML*）论文集（机器学习研究论文集，第 *70* 卷）。PMLR，1799–1808。https://proceedings.mlr.press/v70/kandasamy17a.html[17] Aaron Klein、Stefan Falkner、Simon Bartels、Philipp Hennig 和 Frank Hutter。2017 年。大型数据集上的机器学习超参数快速贝叶斯优化。载于第 *20* 届人工智能与统计学国际会议（*AISTATS*）论文集（机器学习研究论文集，第 *54* 卷）。PMLR，528–536。https://proceedings.mlr.press/v54/klein17a.html [18] Joshua Knowles。2006年。ParEGO：一种针对昂贵多目标优化问题的在线景观近似混合算法。*IEEE* 进化计算汇刊10，1（2006年），50–66。doi:10.1109/TEVC.2005.851274[19] Lisha Li、Kevin Jamieson、Giulia DeSalvo、Afshin Rostamizadeh 和 Ameet Talwalkar。2018年。Hyperband：一种新颖的基于老虎机的方法，用于超参数优化。机器学习研究期刊18，185（2018年），1–52。https://jmlr.org/papers/v18/16-558.html[20] Liam Li、Kevin Jamieson、Afshin Rostamizadeh、Ekaterina Gonina、Jonathan Ben-tzur、Moritz Hardt、Benjamin Recht 和 Ameet Talwalkar。2020年。大规模并行超参数调优系统。载于机器学习与系统会议论文集（*MLSys*），第2卷。230–246。https://proceedings.mlsys.org/paper_files/paper/2020/hash/a06f20b349c6cf09a6b171c71b88bbfc-Abstract.html[21] Shigang Li、Jingkun Dong、Jihao Chen、Zhi Ma 和 Zhongzhe Hu。2025年。Hypertron：通过探索高维并行化空间高效扩展大型模型。载于国际高性能计算、网络、存储与分析会议（*SC*）论文集。 1755 1768 10 1145 3712285 3759783 – 。doi: 。/。 [22] Richard Liaw、Eric Liang、Robert Nishihara、Philipp Moritz、Joseph E. Gonzalez 和 Ion Stoica。2018年。Tune：用于分布式模型选择和训练的研究平台。*arXiv*预印本 *arXiv:1807.05118*（2018年）。doi:10.48550/arXiv.1807.05118[23] Guodong Liu、Youshan Miao、Zhiqi Lin、Xiaoxiang Shi、Saeed Maleki、Fan Yang、Yungang Bao 和 Sa Wang。2024年。Aceso：通过迭代式瓶颈缓解实现高效并行DNN训练。载于第十九届欧洲计算机系统会议（*EuroSys*）论文集。163–181。doi:10.1145/3627703.3629554[24] Hao Liu、Matei Zaharia 和 Pieter Abbeel。2023年。通过分块式Transformer实现环形注意力与近无限上下文。*arXiv*预印本 *arXiv:2310.01889*（2023年）。doi:10.48550/arXiv.2310.01889[25] Llama团队。2024年。Llama 3模型群。*arXiv*预印本 *arXiv:2407.21783*（2024年）。doi:10.48550/arXiv.2407.21783[26] Sam McCandlish、Jared Kaplan、Dario Amodei 和 OpenAI Dota团队。2018年。大批量训练的经验模型。*arXiv*预印本 *arXiv:1812.06162*（2018年）。doi:10.48550/arXiv.1812.06162[27] William Merrill、Shane Arora、Dirk Groeneveld 和 Hannaneh Hajishirzi。2025年。重新审视临界批大小：一种面向大批量语言模型训练的简单经验方法。载于神经信息处理系统进展，第38卷。https://proceedings.neurips.cc/paper_files/paper/2025/hash/a99f732df9b668284b449da0214a3286-Abstract-Conference.html[28] Kaisa Miettinen。1999年。非线性多目标优化。运筹学与管理科学国际丛书，第12卷。Kluwer Academic Publishers，美国波士顿。doi:10.1007/978-1-4615-5563-6[29] MiniMax、Aonian Li、Bangwei Gong、Bo Yang、Boji Shan、Chang Liu、Cheng Zhu、Chunhao Zhang、Congchao Guo、Da Chen、Dong Li、Enwei Jiao、Gengxin Li、Guojun Zhang、Haohai Sun、Houze Dong、Jiadai Zhu、Jiaqi Zhuang、Jiayuan Song、Jin Zhu、Jingtao Han、Jingyang Li、Junbin Xie、Junhao Xu、Junjie Yan、Kaishun Zhang、Kecheng Xiao、Kexi Kang、Le Han、Leyang Wang、Lianfei Yu、Liheng Feng、Lin Zheng、Linbo Chai、Long Xing、Meizhi Ju、Mingyuan Chi、Mozhi Zhang、Peikai Huang、Pengcheng Niu、Pengfei Li、Pengyu Zhao、Qi Yang、Qidi Xu、Qiexiang Wang、Qin Wang、Qiuhui Li、Ruitao Leng、Shengmin Shi、Shuqi Yu、Sichen Li、Songquan Zhu、Tao Huang、Tianrun Liang、Weigao Sun、Weixuan Sun、Weiyu Cheng、Wenkai Li、Xiangjun Song、Xiao Su、Xiaodong Han、Xinjie Zhang、Xinzhu Hou、Xu Min、Xun Zou、Xuyang Shen、Yan Gong、Yingjie Zhu、Yipeng Zhou、Yiran Zhong、Yongyi Hu、Yuanxiang Fan、Yue Yu、Yufeng Yang、Yuhao Li、Yunan Huang、Yunji Li、Yunpeng Huang、Yunzhi Xu、Yuxin Mao、Zehan Li、Zekang Li、Zewei Tao、Zewen Ying、Zhaoyang Cong、Zhen Qin、Zhenhua Fan、Zhihang Yu、Zhuo Jiang 和 Zijia Wu。2025年。MiniMax-01：使用闪电注意力扩展基础模型。*arXiv*预印本 *arXiv:2501.08313*

（2025年）。doi:10.48550/arXiv.2501.08313[30] Deepak Narayanan、Aaron Harlap、Amar Phanishayee、Vivek Seshadri、Nikhil Devanur、Gregory R. Ganger、Phillip B. Gibbons 和 Matei Zaharia。2019年。PipeDream：用于DNN训练的广义流水线并行。载于第*27*届*ACM*操作系统原理研讨会（*SOSP*）论文集。doi:10 1145 3341301 3359646。/。Deepak [31] Narayanan、Mohammad Shoeybi、Jared Casper、Patrick LeGresley、Mostofa Patwary、Vijay Anand Korthikanti、Dmitri Vainbrand、Prethvi Kashinkunti、Julie Bernauer、Bryan Catanzaro、Amar Phanishayee 和 Matei Zaharia。2021年。使用Megatron-LM在GPU集群上进行高效的大规模语言模型训练。载于国际高性能计算、网络、存储与分析会议（*SC*）论文集。doi:10 1145 3458817 3476209 。/。[32] OpenAI。2023 4 *2303 08774*。GPT技术报告。*arXiv*预印本 *arXiv:*。（2023 10 48550 2303 08774）。doi：。/arXiv。。Aurick Qiao、[33] Sang Keun Choe、Suhas Jayaram Subramanya、Willie Neiswanger、Qirong Ho、Hao Zhang、Gregory R. Ganger 和 Eric P. Xing。2021年。Pollux：面向有效吞吐量优化深度学习的协同自适应集群调度。载于第*15*届*USENIX*操作系统设计与实现研讨会（*OSDI*）。USENIX协会，1–18。https://www.usenix.org/conference/osdi21/presentation/qiao [34] Samyam Rajbhandari、Jeff Rasley、Olatunji Ruwase 和 Yuxiong He。2020年。ZeRO：面向万亿参数模型训练的内存优化。载于国际高性能计算、网络、存储与分析会议（*SC*）论文集。doi:10.1109/SC41405.2020.00024[35] Alexander Sergeev 和 Mike Del Balso。2018年。Horovod：在TensorFlow中实现快速、简便的分布式深度学习。*arXiv*预印本 *arXiv:1802 05799* 2018 10 48550 1802 05799。（）。doi：。/arXiv。。[36] Christopher J. Shallue、Jaehoon Lee、Joseph Antognini、Jascha Sohl-Dickstein、Roy Frostig 和 George E. Dahl。2019年。衡量数据并行对神经网络训练的影响。机器学习研究期刊20，112（2019年），1–49。https://www.jmlr.org/papers/v20/18-789.html[37] Mohammad Shoeybi、Mostofa Patwary、Raul Puri、Patrick LeGresley、Jared Casper 和 Bryan Catanzaro。2019年。Megatron-LM：使用模型并行训练数十亿参数的语言模型。*arXiv*预印本 *arXiv:1909 08053* 2019 10 48550 1909 08053。（）。[38] doi：。/arXiv。。Samuel L. Smith、Pieter-Jan Kindermans、Chris Ying 和 Quoc V. Le。2018年。不要衰减学习率，而要增大批大小。载于国际学习表征会议（*ICLR*）论文集。https://openreview.net/forum?id=B1Yy1BxCZ [39] Hugo Touvron、Thibaut Lavril、Gautier Izacard、Xavier Martinet、Marie-Anne Lachaux、Timothée Lacroix、Baptiste Rozière、Naman Goyal、Eric Hambro、Faisal Azhar、Aurelien Rodriguez、Armand Joulin、Edouard Grave 和 Guillaume Lample。2023年。LLaMA：开放且高效的基础语言模型。*arXiv*预印本 *arXiv:2302.13971*（2023年）。doi:10.48550/arXiv.2302.13971[40] Hugo Touvron、Louis Martin、Kevin Stone、Peter Albert、Amjad Almahiri、Yasmine Babaei、Nikolay Bashlykov、Soumya Batra、Prajjwal Bhargava、Shruti Bhosale、Dan Bikel、Lukas Blecher、Cristian Canton Ferrer、Moya Chen、Guillem Cucurull、David Esiobu、Jude Fernandes、Jeremy Fu、Wenyin Fu、Brian Fuller、Cynthia Gao、Vedanuj Goswami、Naman Goyal、Anthony Hartshorn、Saghar Hosseini、Rui Hou、Hakan Inan、Marcin Kardas、Viktor Kerkez、Madian Khabsa、Isabel Kloumann、Artem Korenev、Punit Singh Koura、Marie-Anne Lachaux、Thibaut Lavril、Jenya Lee、Diana Liskovich、Yinghai Lu、Yuning Mao、Xavier Martinet、Todor Mihaylov、Pushkar Mishra、Igor Molybog、Yixin Nie、Andrew Poulton、Jeremy Reizenstein、Rashi Rungta、Kalyan Saladi、Alan Schelten、Ruan Silva、Eric Michael Smith、Ranjan Subramanian、Xiaoqing Ellen Tan、Binh Tang、Ross Taylor、Adina Williams、Jian Xiang Kuan、Puxin Xu、Zheng Yan、Iliyan Zarov、Yuchen Zhang、Angela Fan、Melanie Kambadur、Sharan Narang、Aurelien Rodriguez、Robert Stojnic、Sergey Edunov 和 Thomas Scialom。2023年。Llama 2：开放基础模型与微调聊天模型。*arXiv*预印本 *arXiv:2307 09288* 2023 10 48550 2307 09288。()。doi：。/[41] arXiv.。Jian Wu、Saul Toscano-Palmerin、Peter I. Frazier 和 Andrew Gor- don Wilson。2020年。用于超参数调优的实用多保真贝叶斯优化。载于第*35*届人工智能不确定性会议（*UAI*）（机器学习研究论文集，第*115*卷）。PMLR，788–798。https://proceedings.mlr. press/v115/wu20a.html [42] Yanli Zhao、Andrew Gu、Rohan Varma、Liang Luo、Chien-Chin Huang、Min Xu、Less Wright、Hamid Shojanazeri、Myle Ott、Sam Shleifer、Alban Desmaison、Can Balioglu、Pritam Damania、Bernard Nguyen、Geeta Chauhan、Yuchen Hao、Ajit Mathews 和 Shen Li。2023年。PyTorch FSDP：扩展全分片数据并行的实践经验。*VLDB*基金会论文集16，12（2023年），3848–3860。doi:10.14778/3611540.3611569[43] Lianmin Zheng、Zhuohan Li、Hao Zhang、Yonghao Zhuang、Zhifeng Chen、Yanping Huang、Yida Wang、Yuanzhong Xu、Danyang Zhuo、Eric P. Xing、Joseph E. Gonzalez 和 Ion Stoica。2022年。Alpa：为分布式深度学习自动化算子间与算子内并行。载于第*16*届*USENIX*操作系统设计与实现研讨会（*OSDI*）。559–578。https://www.usenix.org/conference/osdi22/presentation/zheng-lianmin [44] Eckart Zitzler、Marco Laumanns 和 Lothar Thiele。2001年。*SPEA2*：改进强度帕累托进化算法。技术报告 TIK报告103。苏黎世联邦理工学院计算机工程与网络实验室。doi:10.3929/ethz-a-004284029