arXiv:2601.06520v1 [cs.DC] 10 Jan 2026

SkyNomad：关于使用多区域竞价实例降低人工智能批处理作业成本

李志飞*†，夏天*†，毛子明†，周子涵¶，Ethan J. Jackson†，Jamison Kerney†，Zhanghao Wu†，Pratik Mishra§，徐毅†，Yifan Qiao†，Scott Shenker†,⋄，Ion Stoica†

†加州大学伯克利分校 ¶上海交通大学 §*AMD* ⋄国际计算机科学研究所

摘要

模型训练、推理流水线和数据分析等人工智能批处理作业需要大量 GPU 资源，而且通常必须在截止时间前完成。竞价型实例的成本比按需实例低 3–10×，但其可用性难以预测，因此难以确保按时完成。现有系统要么只依赖竞价型实例，面临违反截止期限的风险，要么仅在简化的单区域设置中运行。这些方法忽视了竞价实例可用性、生命周期和价格在空间与时间上的显著异质性。我们表明，利用这种异质性来获取更多竞价型实例容量，是降低作业执行成本的关键。

我们提出了 SkyNomad，这是一种多区域调度系统，能够在确保满足截止期限的同时，最大限度地使用竞价型实例并降低成本。SkyNomad 通过轻量级探测估算可用性、预测竞价实例生命周期、计入迁移成本，并将区域特征和截止期限压力统一纳入货币成本模型，以指导调度决策。评估结果显示，在真实云部署中，SkyNomad 可节省 1.25–3.96× 的成本；在仿真中，其表现与最优策略的成本差异始终在 10% 以内，同时始终满足截止期限。

1 引言

新兴的生成式人工智能工作负载需要前所未有的计算容量。大规模模型训练、离线推理和数据分析都需要大量 GPU 资源，通常需要数小时甚至数天才能完成 [9, 11, 20, 25, 44]。本文将这类长时间运行的作业称为人工智能批处理作业，它们通常需要在指定的截止时间前完成。这类截止时间自然产生，并取决于工作负载。例如，周期性数据处理必须在固定的时间窗口内（如几分钟到几小时）完成，以确保下游分析或仪表板反映实时数据。

*贡献相同。

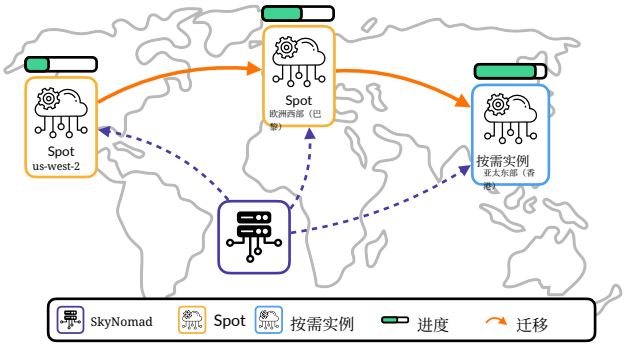


图 1：SkyNomad 概览。SkyNomad 会在多个区域之间迁移人工智能批处理作业，以利用可用的竞价型实例并降低成本。它会监控作业进度，并在竞价实例容量不足时切换到按需实例，确保作业在截止时间前完成。

随着模型规模和系统复杂度不断增长，运行这些作业的成本也越来越高。本文重点探讨如何在满足截止时间的同时，尽可能降低运行人工智能批处理作业的成本。许多此类作业假定自身可中断，并且必须能够容忍故障（§3.1），因为它们通常需要长时间运行，而 GPU 实例也可能以不可忽略的概率发生故障 [16, 39]，执行期间发生故障的可能性相当高。为避免故障发生时浪费计算资源，这些作业通常采用基于检查点的恢复 [18, 36–38, 46, 47, 49]。

为降低成本，我们可以利用作业的松弛时间，即作业在面临截止期限风险之前所能容忍的延迟时长，以及这些作业可被抢占的特性，使用竞价实例。云服务提供商以这些实例作为按需实例的低成本替代方案，其成本通常低10× 3– [3]。不过，它们存在两项限制：会被云服务提供商抢占，并且在竞价市场中可能暂时不可用。因此，仅依靠竞价实例容量无法保证按时完成截止期限，必须与按需实例结合使用。为了在满足截止期限的同时最大限度地节省成本，我们需要考虑

在哪里以及何时使用竞价实例，因为它们在空间上跨区域、在时间上跨时间表现出异质性。选择得当后，我们可以以更低的价格获取更多竞价实例容量，进一步降低成本，尤其是对于使用 GPU 且频繁遭遇抢占的人工智能批处理作业 [32]。现有研究（§2, [4, 8, 13, 23, 43, 50]）探讨了这一机会，但没有同时考虑竞价异质性的空间和时间维度。接下来，我们将详细介绍这两种异质性。

区域间的空间异质性。 在公有云提供商中，竞价实例在不同云区域的可用性可能存在很大差异（§3.2.1）。例如，`eu-central-1`可能仍有竞价实例容量，而`us-east-1`则完全没有。通过利用不同区域的竞价实例，即使某个区域的容量耗尽，作业仍可继续在竞价实例上运行并取得进度，同时节省更多成本。不同区域的定价策略也会根据供需模式而变化，差异最高可达 5× [5, 14]（§3.2.3），从而创造主动迁移到更便宜区域以节省成本的机会。然而，将这些机会结合起来颇具挑战。区域可用性事先是未知的，当不同区域的可用性和价格出现分歧时，调度器必须在多维权衡中做出决策，例如在可用性高但成本高的区域与容量有限但成本低的区域之间进行选择。此外，在检查点较大的情况下跨区域迁移作业会产生不可忽略的出口成本 [1, 12, 34]，因此，迁移必须通过提升可用性或降低价格来证明其合理性 [26, 42]。

跨时间的时间异质性。 竞价实例生命周期（即从资源配置到被抢占之间的时长）会随时间显著变化（§3.2.2）。我们观察到，由频繁抢占导致的较短竞价实例生命周期，往往出现在短暂的波动期。避开这些时期，让作业使用生命周期更长的竞价实例，有助于摊销冷启动成本（即从上一个检查点恢复所需的成本）并提高有效吞吐量。然而，预测竞价实例生命周期仍然很困难。同时，竞价实例可用性也会随时间显著变化（§3.2.1），进一步加大了调度的复杂性。在调度初期，策略可以等待未来可用性更高的时段，在这些时段运行，并在资源不可用时暂停，以提高竞价实例使用率。随着时间推移、截止时间日益临近，策略必须变得更加谨慎，优先利用当前可用资源稳步推进；这意味着要在未来可用性可能更高的机会与确保按时完成之间权衡。

总之，真实云环境十分复杂，同时呈现空间和时间异质性。以截止时间为考量的先前研究 [50] 探讨了这一问题，但仅考虑单一区域，因此许多此类挑战并不会出现。扩展到多个区域则从根本上更为困难，因为必须利用空间异质性，同时考虑迁移成本；还必须跨区域评估时间异质性。在这些条件下运行的调度器必须随时决定：(i) 使用当前区域的竞价实例，或等待其出现；(ii) 迁移到有可用竞价实例容量的其他区域；或 (iii) 使用始终可用的按需实例，以推进任务并赶上截止时间。

我们设计了一种多区域竞价调度策略，以应对上述挑战。首先，我们以货币成本量化剩余进度的价值（§4.5）。我们根据当前的截止期限压力，估算按时完成作业的预期成本，从而判断作业何时可以安全暂停，以等待更好的机会。其次，竞价实例仅在其有效时间内（即冷启动后的运行时间）取得进度。为此，我们采用在线区域可用性探测（§4.3）来预测竞价实例生命周期（§4.4），并且只在冷启动后计入进度价值，以避免选择可用性波动大且频繁遭遇抢占的区域。第三，我们将这一价值与各区域的运营成本进行比较，以指导策略决策（§4.6），使系统能够持续评估各区域，并在出现成本效益更高的区域时择机迁移（§4.7）。

我们在 `SkyNomad`（图 1）中实现了这一策略。SkyNomad 是一个 AI 批处理作业执行系统，它从多个区域获取竞价实例容量，以在截止时间内完成作业。用户指定作业并启用定期检查点，`SkyNomad` 则根据策略决策，在各区域之间迁移计算任务和检查点，以最大限度地降低成本。

为了评估 `SkyNomad`，我们将其部署在公有云上运行真实的批处理作业，以体验实时抢占和迁移。如 §6.1 所示，与先前的研究系统（例如，Uniform Progress [50]）和生产系统（例如，Amazon SageMaker [4]）相比，`SkyNomad`可节省 1.25– 3.96× 成本，同时满足截止期限。我们还通过重放真实的抢占轨迹并与现有策略进行比较，开展了全面评估。结果表明，在广泛的配置范围内，`SkyNomad` 的表现与全知策略相差不超过 10%。这些结果表明，通过利用截止时间裕度并从多个区域获取竞价实例容量，大幅降低人工智能批处理作业的成本是可行的。

总之，本文做出了三项贡献：

- 发现了利用跨区域竞价实例降低人工智能批处理作业成本的机会。
- 设计了一种调度策略，能够从多个区域获取竞价实例容量，同时保证满足截止期限。
- SkyNomad，一款可跨区域运行作业并显著节省成本的 AI批处理作业执行系统。

2 个现有的人工智能批处理作业系统

本节回顾了若干现有的云端批处理作业运行系统，并讨论了它们的局限性以及降低成本的机会缺失问题（表1）。

	截止时间保证	Spot 实例	多区域区域
SageMaker [4]	✓	×	×
仅竞价实例 [4, 13, 43]	×	✓	×
SkyServe [32]	×	✓	✓
均匀进度 [50]	✓	✓	×
SkyNomad	✓	✓	✓

表1：系统比较。仅竞价实例系统：Sage——由服务商管理的竞价型实例 [4], Parcae [13], 和 Bamboo [43]。

2.1 仅按需实例系统

最常见的方法是在云端启动按需实例，运行作业直至完成，然后终止这些实例。AWS SageMaker [4],就是一个例子，它是用于人工智能批处理作业的托管执行系统（我们将在后文讨论其对竞价型实例的托管支持）。这种方法能够保证在截止时间前及时执行，并避免大多数故障，从而简化系统设计。然而，这种方法成本很高。由于按需执行忽略了空闲时间，作业往往会远早于截止时间完成，造成不必要的成本。

2.2 竞价型系统

先前的工作 [4,13,32,43,53,56] 使用竞价型实例来降低成本。这些系统分为两类：(1) 面向批处理的仅竞价实例系统，以及 (2) 在线服务多区域系统。

仅竞价实例系统无法满足截止时间要求。一些现有系统（如 SageMaker 托管竞价型实例 [4],Parcae [13] 和 Bamboo [43] ）完全依赖竞价型实例。当竞价实例可用性较高时，这种方式运行良好；但当竞价实例容量变得稀缺时，执行可能会无限期延迟。先前的工作 [32] 表明，在最坏情况下，一个区域可能会耗尽竞价型实例资源长达 72 小时，在此期间作业不会取得任何进度。**在线系统面向不同的工作负载**。SkyServe [32] 等系统在多个区域中的竞价型实例上运行在线大语言模型服务，并优先保证响应能力；当竞价型实例不足以维持延迟服务级别目标时，则配置按需实例。由于这些系统会立即处理请求，无法利用空闲时间，因此错失了节省成本的机会，也不适合批处理作业。

2.3 截止时间感知系统

均匀进度（UP） [50]是一种截止时间感知策略，可交替使用按需实例和竞价实例。竞价实例可用时便使用竞价实例；当容量稀缺或松弛时间较少时，则改用按需实例，并将作业进度均匀分布在整个时间段内，以便在截止时间前完成。尽管 UP 简单易用，但它忽略了竞价实例可用性的时间变化，且仅在单一区域运行；当该区域没有竞价实例容量时，它只能依赖按需实例，导致成本接近始终使用按需实例的情况（§6.1）。

UP 假设单一区域内的竞价实例可用性足够稳定，能够确保进度稳步推进。我们的分析见 §3，结果表明，真实世界的可用性和生命周期在不同区域和时间段内都高度不均，因此单区域策略本质上效率低下。因此，更有效的策略可以利用多个区域，获取额外的竞价实例容量，并进一步降低成本 [6]。

3 背景与动机

3.1 人工智能批处理作业的特征

截止期限与松弛时间。 人工智能批处理作业通常是内部工作负载，不会直接面向终端用户。与优先考虑响应能力且必须满足严格服务级别目标（SLO）的在线服务不同，人工智能批处理作业通常运行时间较长，无需立即响应。例如，使用人工智能模型进行模型训练和大规模数据分析。虽然这类作业对延迟不敏感，但通常有截止期限，不能无限期延迟。例如，模型训练必须在预定发布之前完成，每日数据管道必须在下一个周期开始之前完成。本文假设作业具有预先指定的明确截止时间，并且必须在该时间前完成。我们还假设作业的执行时间短于截止时间前的可用时间，因此会产生松弛时间，即作业在面临截止期限违约风险之前可以容忍的延迟时长。

可中断。 由于运行时间较长，批处理作业通常设计为可中断。它们必须能够承受意外故障，而不会损失大量进度。训练作业通常会定期创建检查点，将模型参数和优化器状态保存为恢复点 [36, 57]。如果发生故障，作业会从最近的检查点恢复。批量推理和数据处理工作流通常可以分解为彼此独立的单元，并逐步保存各单元的输出；已处理的数据索引则可充当轻量级检查点。这样，发生故障时只需从尚未处理的单元重新开始，无须重新执行整个数据集。

冷启动延迟。 随着模型规模增大，批处理作业在启动时以及每次恢复时都会产生不可忽略的冷启动延迟。这一延迟包括实例配置、依赖项安装，以及将模型权重下载并加载到 GPU 上。对于模型训练等有状态作业，在新位置恢复时还包括传输作业状态。例如，在 8 个 GPU 上启动一个针对 14B 参数模型的监督微调，可能需要 5–10 分钟。这些冷启动延迟会显著影响整体进度，因此设计调度策略时必须将其纳入考量。

3.2 竞价实例的异质性

云服务提供商提供一类特殊的实例——竞价实例，作为一种节省成本的选择。它们的价格通常比常规按需实例便宜 3–10× 倍 [28, 32, 50] ，但

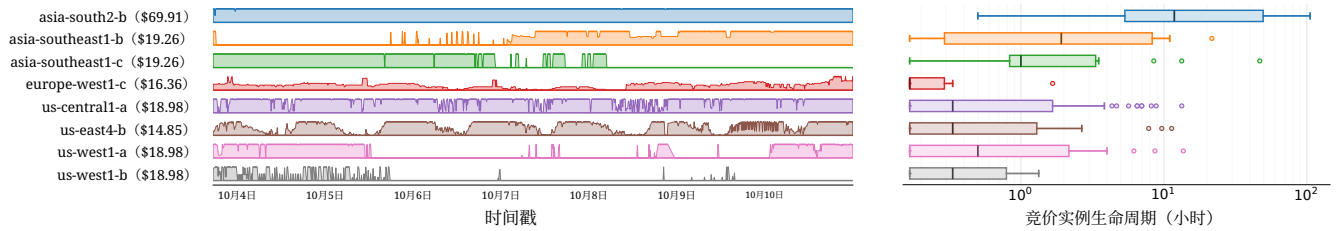


图2：不同区域的竞价实例可用性和生命周期差异很大。**左侧**：展示了 GCP 上 16 个 a3-highgpu-1g（1×H100）实例在 7 天内的可用性，涵盖我们采集数据的 13 个可用区中具有代表性的 8 个。纵轴表示每个时刻可用的实例数量。**右侧**：生命周期分布的箱线图，采用对数刻度。我们考察一个以组调度方式使用 16 个实例的作业：如果任意一个实例被抢占，整个作业都会被抢占；只有当这 16 个实例全部再次可用时，作业才会恢复。图中展示了这一情形下得到的生命周期。

也存在随时被云服务提供商抢占 的风险。竞价实例在多个维度上都表现出空间异质性（不同区域之间）和时间异质性（随时间变化）。接下来，我们将介绍这些维度，以及利用它们降低成本所面临的挑战。

3.2.1 非均匀抢占式实例可用性模式

竞价实例可用性指的是某个区域能够提供竞价实例容量的时段。如果预言机调度器能够完美预知未来的可用性，作业几乎总是可以完全 在竞价实例上运行，尽管不一定是在同一个区域，从而大幅降低成本。然而在实践中，各区域的可用性模式各不相同，且会随时间波动。如果事先不知道可用性，就很难描述各个区域的情况，也难以判断任一时刻哪个区域最合适。

我们收集了 13 个 GCP 区域中 16 个 `a3-highgpu-1g`实例（1×H100）的竞价实例可用性轨迹。图 2展示了 8 个具有代表性的示例。不同区域呈现出不同的可用性模式：有些区域通常都有可用实例（`asia-south2-b`），有些区域大多可用，但经常遭遇抢占（`us-central1-a`），还有些区域基本不可用（`us-west1-b`）。我们发现，不同区域同时发生抢占的情况很少见，这与 [27, 32]中的观察结果一致。这使各区域能够相互补充：几乎在任何时刻，至少有一个区域拥有可用的竞价实例容量。先前的研究 [32] 表明，纳入超过四个区域后，竞价实例可用时段所占比例会迅速接近 99%；我们在轨迹中观察到，纳入 6 个区域时也有类似表现。

除了跨区域异质性之外，单一区域内的可用性也存在很大差异。例如，`asia-southeast1-c` 在跟踪记录的前半段可用性很高，但在后半段则完全不可用。`us-east4-b` 呈现出明显的昼夜模式，夜间可用，白天不可用。这些观察结果表明，没有哪个完美的区域始终可用（我们将在 §`asia-south2-b` 中讨论其局限性，见 §3.2.3）。因此，单区域策略从根本上受到限制，而应对非均匀抢占式实例可用性模式则成为核心挑战。

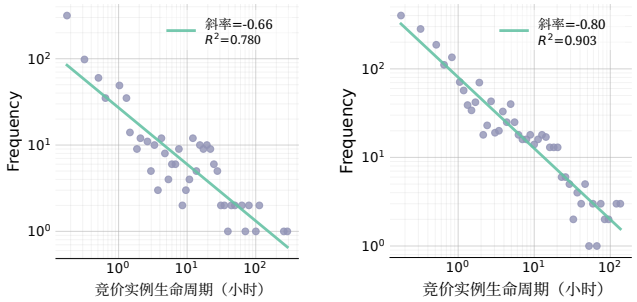
3.2.2 变化很大的竞价实例生命周期

除了可用性之外，竞价实例生命周期也是另一项关键指标。它指竞价实例在被抢占之前可使用的时长。由于冷启动时间不可忽略，且不会推进作业进度，如果竞价实例的生命周期接近或短于冷启动延迟，那么短生命周期的竞价实例几乎没有价值。令d 表示冷启动时间。如果竞价实例的生命周期t_0,，那么作业实际推进的有效时间为$t_0 - d$。当t_0 接近d时，由于大部分时间都花在重新启动而非计算上，作业的有效吞吐量会非常低。

然而，生命周期分布在不同区域和不同时间都存在很大差异。例如，图2 展示了 8 个区域的竞价实例生命周期分布：中位生命周期从不到 1 小时（`europe-west1-c`）到超过 20 小时（`asia-south2-b`），跨度超过一个数量级。选择竞价实例生命周期较长的区域至关重要，因为这样可以减少恢复操作的次数，并将冷启动开销分摊到更长的执行时间内。

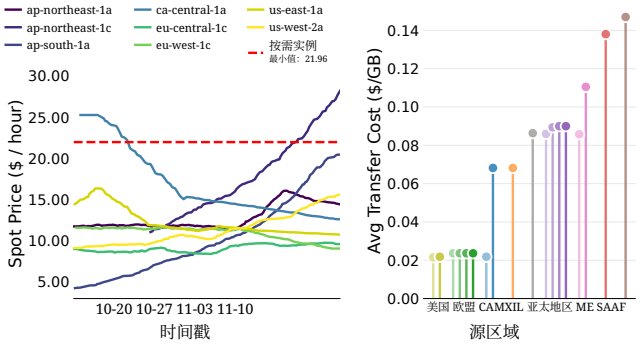
我们还观察到，许多竞价实例抢占发生在很短的时间跨度内，我们将其称为波动期。这些时期会产生许多短命的竞价实例，它们仅短暂可用，几乎会立即被抢占。例如，在区域 us-central1-a 中，15 天期间内有 90% 的竞价实例抢占发生在 85 小时以内。

相反，我们观察到竞价实例生命周期呈现重尾分布 [24,30]：实例存活的时间越长，其预期剩余生命周期往往也越长。如图3所示，双对数空间中的近线性衰减（$R^2 \approx 0.78$–0.90）证实了这一模式：大多数实例会在几小时内被抢占，但仍有相当一部分实例能够存活数十小时甚至更久。



(a) 16×GCP 上的 H100。 (b) 1×AWS 上的 V100。

图 3：不同加速器上竞价型实例的生命周期分布。散点表示等宽对数10分箱中的频数，实线是在双对数空间中的最小二乘拟合，显示出明显的重尾模式 [10, 31]。



(a) 竞价价格。 (b) 出口成本。

图4：不同区域和时间的成本差异。(a) 相同实例类型的**竞价价格**。虚线表示最低按需价格。(b) **出口成本** 由源区域决定，范围从 0.02 美元/GB（美国、欧盟）到 0.14 美元/GB（南美洲、非洲）。缩写遵循 AWS 区域命名惯例（例如，AF 表示非洲，SA 表示南美洲）。

3.2.3 定价差异与主动迁移

异质性的另一种形式体现在定价策略上。如图4a所示，同一实例类型的价格在不同区域之间可能存在显著差异，最高可达 5×。在大多数云（包括 AWS [5] 和 GCP [15]）中，价格还会随时间波动，并且仅在 12 天内就可能变化达 1。7× 以往的研究 [32, 50] 往往忽视了这些差异，并假设采用静态定价，从而过度简化了这一问题。

除了实例价格外，云服务提供商还会对跨区域数据传输收取额外费用。对于需要保留大型检查点的作业（例如训练工作负载），每当作业迁移到新区域时，都必须移动检查点。这种移动会产生不可忽视的出口流量成本 [1, 12, 34]，具体金额因源区域而异。如图4b所示，不同区域的出口成本最高可相差 7× 。例如，一个拥有 140 亿参数的微调作业可能需要移动约 500GB 的复制检查点；从美国到欧洲的费用约为 10 美元，而从亚洲到美国则约为 40 美元。

迁移到竞价实例价格更低的区域可能颇具吸引力，即使当前竞价实例尚未被抢占。然而，只有当更低价格区域的竞价实例生命周期足够长，足以抵偿迁移成本时，这种价格感知主动迁移才有益。设区域i的价格为p_i，区域i和j之间的迁移成本为$C_{i,\ j}$，区域i的预期竞价实例生命周期为t_i，而d为冷启动延迟。假设作业当前正在区域a的竞价实例上运行。当价格更低的区域b变得可用时，只有满足$C_{a,\ b} \leq p_a - p_b t_b - d$时，迁移才有益。

然而，单一区域很少能同时提供最佳可用性、最长生命周期和最低价格。例如，在 §2.3 中，尽管 `asia-south2-b` 的可用性较高，4×但其价格高于最便宜的区域，且接近按需价格。相比之下，`us-east4-b` 的成本最低，但会频繁遭遇抢占和波动期。加上迁移成本，在这四项指标之间权衡会形成庞大的组合决策空间。

4 设计

现在，我们介绍 `SkyNomad` （图 5，算法 1）的设计。通过考虑可用性、生命周期和定价方面的时空异质性（§3.2），充分利用竞价型实例运行人工智能批处理作业。为及时掌握这种异质性，`SkyNomad` 持续收集各区域的实时信息（§4.3），并结合生存分析 [2] 来预测竞价实例生命周期（§4.4）。为了判断当前的竞价实例可用性是否值得把握，`SkyNomad` 会考虑当前的截止期限压力，并估算剩余进度的价值（即预期货币成本）（§4.5），以此作为选择立即使用还是等待更好机会的依据。最后，`SkyNomad` 采用统一成本模型（§4.6），在当前截止期限压力下计算各区域的效用，并在有更优选择时主动迁移；若松弛时间足以等待未来更好的机会，则暂停作业（§4.7）。

4.1 问题形式化

我们考虑一个单独的批处理作业，例如模型训练。该作业在任何时刻都运行于一组固定的实例上，且组内所有实例均位于同一区域，以利用高带宽网络。这与自动扩缩设计不同；后者会同时将工作分配到动态数量的实例和多个区域（我们将在第8节进一步讨论）。我们假设组内所有实例均采用成组调度，并将任何实例的抢占视为整个组被抢占；我们不假设任何特定的部分恢复策略，以最大限度地提高兼容性。不失一般性，本文余下部分聚焦于单实例情况，因为成组调度允许将实例组视为原子单元。

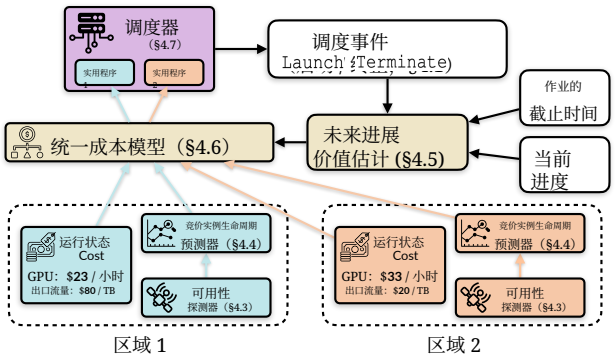


图5：SkyNomad 的设计概览。

该作业会定期创建检查点，使其在发生中断后能够暂停，并在另一个区域恢复运行。我们假设可以预先估算作业的执行时间，并将其表示为P个工作单位。在任何时刻，作业都可以保持空闲，利用松弛时间等待更好的机会，也可以在某个实例上运行。作业在新实例上启动或恢复运行后，需要经历d个时间单位的冷启动延迟（第3.1节），之后才能开始执行。令$p(t)$表示截至时间t所取得的总进度。作业取得进度的速率取决于系统状态：

$$\frac{\mathrm{d}p}{\mathrm{d}t} = \begin{cases} 1 & \text{Ready} \\ 0 & \text{Otherwise (Cold Start or Idle)} \end{cases} \tag{1}$$

作业必须在截止时间T之前完成，也就是说，必须满足$p(T) \geq P$。我们的目标是在满足截止时间的前提下，最小化总成本C，即 $\operatorname{argmin} C \text{ s.t. } p(T) \geq P$。

令$\mathcal{R}$表示所有区域的集合，每个区域均提供竞价型实例和按需实例。作业可以在$\mathcal{R}$中的任意区域运行。在任意时间t，系统处于某个状态$s = (r, m)$，其中$r \in \mathcal{R}$是当前作业所在区域，$m \in \{\text{idle}, \text{spot}, \text{od}\}$是当前模式。idle 模式表示作业正在等待，且没有实例运行；spot 和 od 则分别表示作业正在竞价实例或按需实例上运行。竞价实例价格较低，但可能被抢占；按需实例始终可用，但价格较高。时间t处于状态$s = (r, m)$时的价格记为$C^{(r,m)}(t)$。

状态变化通过以下三个事件发生：

- 启动（r，m）：尝试在区域r中以模式$m \in$ {竞价型实例，订单}启动一个实例。对于m =订单，此操作始终成功；对于m = 竞价型实例，只有在竞价实例容量可用时才会成功，而容量是否可用要到尝试时才能确定。启动成功后，如果检查点尚未位于区域r，则会将其迁移到该区域，并将当前状态设置为（r，m）。

- 终止：停止当前实例，将$m \leftarrow$设置为空闲，同时保持r不变，表示检查点仍保留在同一区域。

- 抢占：云回收竞价实例的一种外部事件，其效果与终止相同。

鉴于上述事件，系统会沿着一条轨迹运行，产生总成本C，由两部分组成：作业在区域之间迁移时产生的计算成本和迁移成本。计算成本在作业运行期间按当前区域的价格累积：$C^{\text{compute}} = \int_{m \neq \text{空闲}} C^{(r,m)}(t)\,\mathrm{d}t$。

每当 r 发生变化时，即在新区域中启动实例时，都会产生迁移成本。每次此类移动都需要传输检查点，成本为 $E_{r_i \to}$ $\mathrm{rj} = e_{r_i \to}\mathrm{rj} \cdot S^{\text{ckpt}}$，其中 $e_{r_i \to}\mathrm{rj}$ 是每字节出口成本，S^{ckpt} 是检查点大小（我们假设 $e_{r_i \to}\mathrm{ri} = 0$）。如果作业依次经过区域 r $_0 \to \mathrm{r}_1 \to \cdots \to \mathrm{r}_k$进行迁移，则总迁移成本为 $C^{\text{migrate}} = \sum_{i=1}^{k} E$ $_{r_{i-1} \to}\mathrm{ri}$。因此，总成本为 C $= C_{\text{compute}} + C^{\text{migrate}}$。

4.2 截止期限感知规则

首先，我们制定若干规则，以确保作业在截止时间前完成且不造成浪费。这些规则改编自 [50]，并扩展为多区域版本。

节俭规则。 所有工作完成后，作业应保持空闲。也就是说，当 $p(t) \geq P$ 时，它应处于$s = (r, m)$ 状态，其中 m = 为空闲。

安全网规则。 当作业处于空闲状态且剩余松弛时间不足时，应切换到按需实例，并保持使用该实例直到完成。具体而言，在时间 t，若 $T - t < P - p(t) + 2d$，调度器就会切换到按需实例，以保证赶上截止时间。术语 $2d$ 表示当前启动的冷启动时间，以及抢占后可能发生的回退所需时间。

多区域回退。 在多区域环境中，各区域的按需实例价格可能不同，因此在可行时，回退应选择价格最低的按需实例选项。假设触发安全网时，作业当前位于区域 r_0，我们选择回退区域 r 为

$$\arg\min_{r \in \mathcal{R}} \left[C_{(r,\text{od})}(t) \cdot (P - p(t) + d) + E_{r_0 \to r} \right] \tag{2}$$

其中，第一项表示在区域 r中完成剩余工作的按需实例成本，第二项表示将检查点从 r_0 迁移到 r的迁移成本。

4.3 竞价实例可用性探测

鉴于竞价实例可用性的异质性（§3.2.1），我们使用在线可用性探测来刻画每个候选区域，并指导调度决策。一次探测是一个启动请求：它以相同的 GPU 配置启动一个实例，并在启动成功后立即终止该实例，从而以经济高效的方式测量区域可用性。我们定期向候选区域发送探测，间隔设为两小时；观测结果表明，可用性模式在此时间窗口内保持稳定（图2）。与可能过时的静态历史轨迹不同，探测可在作业执行期间提供实时信息。

虚拟实例。根据探测结果，我们维护一个虚拟实例视图，仿佛每个区域中都有一个实例持续运行并接收实时抢占

。这一做法源于一项观测：分配失败（无法启动新的竞价型实例）与抢占事件高度相关 [50]。对于每个区域 r，我们根据 (t_i, o_i) 形式的观测构建此类虚拟实例视图，其中 t_i 是时间戳，$o_i \in 0$ 为 1 表示可用。观测有四种来源：(1) 探测（成功 $o = 1$，失败 $o = 0$）；(2) 启动操作（成功 $o = 1$，失败 $o = 0$）；(3) 抢占事件（$o = 0$）；以及 (4) 当我们主动迁离某个区域时的终止操作（$o = 0$）。借助这一逻辑构造，我们无需实际运行实例，就能推断实例何时会被抢占：每当可用性从 1 变为 0，即 $o_{i-1} = 1$ 且 $o_i = 0$ 时，我们便将时间 t_i 发生的此类事件视为虚拟实例被抢占。同样，当可用性从 0 变为 1 时，我们将其视为虚拟实例的资源配置。

4.4 竞价实例生命周期预测

如 §3.2.2所述，了解各区域未来的竞价实例生命周期，有助于调度器优先选择存续时间更长的竞价实例，从而减少抢占次数，并摊销冷启动开销和迁移成本。我们利用各区域的虚拟实例轨迹预测生命周期；为纳入重尾分布的影响，我们会根据虚拟实例当前的存续时间进行条件预测，该时间定义为自上次被抢占以来经过的时间。我们采用生存分析来估算存续时间分布，并进一步根据波动期的观测结果改进模型，以考虑抢占突然激增的情况。

4.4.1 生存分析

我们首先提取虚拟实例的历史存续时间，将其定义为每段连续可用期。每段存续时间始于探测或启动成功时（§4.3中的 (1) 和 (2)），结束于观测到抢占时（3），或者在我们主动迁出该区域时被右删失（4）。由此得到一组虚拟实例存续时间。我们用 $e(l)$ 表示在存续时间 l 时发生的抢占次数（以 (3) 结束），用 $c(l)$ 表示在存续时间l 时发生的主动迁移次数（以 (4) 结束）。

根据近期对虚拟实例的观测，我们首先计算其年龄，记为$a(t)$。例如，若在时间t，最近三次探测均成功，而倒数第四次探测失败，且探测间隔为两小时，则$a(t) = 6$ 小时。根据这一年龄，我们的目标是预测该区域还能保持可用多长时间。这需要估算条件期望剩余生命周期 $\mathbb{E}[L - a(t) \mid L > a(t)]$,，其中$L$是表示生命周期的随机变量。由于删失数据使直接估算变得复杂，我们采用生存分析。

生存分析中的关键量是风险率 $h(l)$，即在存活至l 时刻的条件下，在该时刻终止的瞬时风险。我们使用以下方法估计 $h(l)$：

Nelson-Aalen估计量 [2]：

$$h(l) = \frac{e(l)}{n(l)} \text{ with } n(l) = \sum_{x \geq l} (e(x) + c(x)) \tag{3}$$

其中$e(l)$是在生命周期l时观测到的抢占次数，$c(l)$是在生命周期l时右删失观测（主动迁移）的数量，而$n(l)$是风险集，即生命周期至少为l的实例数量。主动迁移的实例属于右删失：它们计入$n(l)$，但不计入$e(l)$。该估计量是非参数的，也就是说，它不作任何分布假设，并且能自然捕捉重尾模式：当生命周期较长的实例抢占风险较低时，在较大的l处发生的抢占次数较少，因此$e(l)$会下降，$h(l)$也会随l增大而下降。

对$h(l)$求和可得累积风险 $H(l) = \sum_{l_i \leq l} h(l_i)$，由此可推导出生存函数 $S(l) = \exp(-H(l)) = \Pr(L > l)$。对于已存活至年龄$a(t)$的实例，其预期剩余生命周期为：

$$\bar{L}(a(t)) = \mathbb{E}[L - a(t) \mid L > a(t)] = \frac{1}{S(a(t))} \sum_{l_i > a(t)} S(l_i) \tag{4}$$

4.4.2 波动性与风险评估

为了确保进度平稳，调度器必须及时检测波动期（§3.2.2），并避免将作业放置在频繁发生抢占的区域。我们通过将波动性感知融入寿命估计来实现这一点，并推导出调整后的生存函数 $\tilde{S}(l)$。计算 $\tilde{L}(a(t))$ 时，我们使用 $\tilde{S}$ 替代S，从而对当前处于此类波动期的区域施加惩罚。核心思路是将观测到的抢占次数与长期风险率$h(l)$所预测的次数进行比较：如果某个区域发生的抢占远超预期，那么它很可能正处于波动期。对于一个截至当前时间的时间窗口W，我们将波动率 定义为 $\gamma_W = e_W \Sigma_{t \in W} h(a(t))$，其中$e_W$是$W$内观测到的抢占次数，分母则是预期次数，计算方法为对该窗口内每个观测时刻的风险率h 求和。当 $\gamma_W > 1$ 时，实际发生的抢占次数超过预期，表明当前处于波动期。

我们取 $\gamma^* = \max_{W=(t_0 \text{来} t),\ t_0 \in [0,t)} \gamma_W$，遍历所有以当前时间为终点的窗口长度，以便保守地捕捉任何时间尺度下最严重的抢占密度。然后通过缩放累积风险来调整生存曲线：使用 $\tilde{S}(l) = \exp(\gamma^* \cdot -H(l))$ 替换公式（$S(l)$ 中的4）。

4.5 未来进展价值估计

在了解每个区域的竞价型实例行为后，现在的问题是：当前的竞价型实例机会是否值得把握，还是暂停并等待未来的机会。为评估这一点，我们确定未来进度的货币价值：通过比较未来完成同等进度的预期成本与当前观察到的竞价型实例（或

按需实例）的运行成本，我们可以决定系统应当把握当前机会还是等待。这项评估应取决于时间t时的截止期限压力，其定义为剩余待完成进度除以截止时间前的剩余时间：

$$\theta(t) = \frac{P - p(t)}{T - t} \tag{5}$$

以及目前为止的平均进度：

$$\tilde{\theta}(t) = \frac{p(t)}{t} \tag{6}$$

设计原则。我们首先提出若干用于评估任何进度价值的一般原则：

- **均衡锚定**：当作业进度符合预期时（$\theta(t) = \tilde{\theta}(t) = \frac{P}{T}$），策略应接受所有可用的竞价实例，但避免使用按需实例。在这种情况下，历史竞价供给足以满足截止时间的要求，因此，如果此刻只有按需实例可用，等待比额外花费更好。

- **单调性**：截止期限压力 $\theta(t)$ 越高，价值就应越高，因为空闲时间越少，策略就越需要取得进度，以避免发生截止期限违约。

- **尺度不变性**：无论 P 和 T 的绝对值如何，评估都应保持一致，并且只取决于两者的比值。

进度价值。根据这些原则，我们通过直接比较截止期限压力与目前为止的平均进度来估算这一价值：

$$V(t) = C_{\text{od}} \cdot \frac{\theta(t)}{\tilde{\theta}(t)} \tag{7}$$

其中，$C_{\text{od}} = \min_r C_{(r,\text{od})}$ 是所有区域中最低的按需价格（我们假设按需价格不会随时间变化）。它满足以下三项原则：

- **均衡锚定**：当 $\theta(t) = \tilde{\theta}(t)$ 时，进度按$C_{\text{订单}}$评估。为简单起见，我们假设所有竞价型实例的价格都低于按需实例，因此策略会接受任何可用的竞价型实例，因为它们能以较低成本推进进度；同时会避开所有按需实例，因为$C_{\text{订单}}$ 不高于任何按需价格，因此使用按需实例并不划算。
- **单调性**：固定t。那么，$\theta(t)$ 随$p(t)$ 增大而减小，且$\tilde{\theta}(t)$ 随$p(t)$增大而增大。因此，较高的 $\theta(t)$会带来更高的比值 $\frac{\theta(t)}{\tilde{\theta}(t)}$，因此 $V(t)$ 更高。
- **尺度不变性**：$V(t)$ 仅取决于进度与时间的比值，该比值通过 $\theta(t)$ 和 $\tilde{\theta}(t)$ 表示，而与P 或T的绝对值无关。

4.6 统一成本模型

确定进度价值 $V(t)$ （§4.5）后，我们通过计算每个候选状态 $s = (r,m) \in \mathcal{R} \times \{\text{竞价型实例}\}$ 的——预期净效用来评估它（下一段将讨论 订单 和 空闲的情况）。设当前状态为$s_0 = (r_0, m_0)$，并设 $\bar{L}_s$ 表示预期剩余生命周期（§4.4）。在实例生命周期内，我们定义

$$\bar{L}_s \cdot U_s = V(t) \cdot \max(0, \bar{L}_s - d) - C_{(r,m)}(t) \cdot \bar{L}_s - E_{r_0 \to r} \tag{8}$$

其中U_s 表示生命周期内的单位时间效用。右侧三项分别表示：（1）有效运行时间内取得进度的价值，不包括冷启动时间d；（2）计算总成本$C_{(r,m)}(t)$（为简单起见，我们假设实例生命周期内价格不变）；以及（3）如果目标区域不同于当前检查点位置，则产生的一次性迁移成本$E_{r_0 \to r}$。

为了比较不同候选方案，我们将效用除以预期生命周期，得到效用U_s：

$$U_s = \underbrace{V(t) \cdot \eta_s}_{\substack{\text{Expected Value of} \\ \text{Effective Progress}}} - \underbrace{C_{(r,m)}(t)}_{\substack{\text{Operational} \\ \text{Compute Cost}}} - \underbrace{\frac{E_{r_0 \to r}}{\bar{L}_s}}_{\substack{\text{Amortized} \\ \text{Migration Cost}}} \tag{9}$$

其中 $\eta_s = \max(0, \bar{L}_s - d)/\bar{L}_s$ 表示候选状态s的有效性，即其生命周期中用于有效工作的时间占比。因此，$V(t) \cdot \eta_s$ 表示进度的有效价值。短生命周期的竞价实例有效性较低，因为冷启动开销占其运行时间的比重较大。第二项和第三项表示在s下运行的运营成本，即计算成本与摊销后的出口成本之和。迁移成本$E_{r_0 \to r}$ 按生命周期进行摊销，因此生命周期较长的实例能够更有效地摊销这笔固定成本。

特殊情况。对于按需实例，我们假设$L_{(r,\text{od})} \to \infty$（不会发生抢占），因此 $\eta_{(r,\text{od})} \to 1$，迁移成本可以完全摊销，从而得到 $U_{(r,\text{od})} = V(t) - C_{(r,\text{od})}$。对于空闲状态，不会产生成本，但也没有进度，因此$U_{(r,\text{idle})} = 0$。这意味着，只有在$U > 0$时，在$s$ 上运行才值得，也就是说，进度的有效价值超过其成本时才值得运行。

4.7 综合运用

算法 1 概述了调度策略。其核心思想很简单：在每个调度步骤中评估所有候选项，并迁移到任何优于当前状态的选项。必要时，调度器会先应用安全网（§4.2，第 4–5 行）。然后定期向所有区域发送轻量级探测（§4.3，第 6 行）。根据当前的截止期限压力计算未来进度的价值后（§4.5，第 7 行），策略会遍历所有候选状态 $s = (r,m)$，利用虚拟实例视图预测其生命周期（§4.4，第 9 行），并计算其效用$U_{(r,m)}$ （§4.6，第 10 行）。接着，它会将每个候选项与当前状态的效用$U_{(r_0,m_0)}$进行比较。[2] 候选项按效用从高到低依次尝试，调度器会迁移到第一个成功的候选项

[2]在实践中，我们会加入滞回阈值 Δ 以防止抖动：只有当$U_{(r,m)} > U_{(r_0,m_0)} + \Delta$时，调度器才会切换。

算法1 SkyNomad 调度策略

输入： 需要总进度为 P、截止时间为 T 的作业，以及候选区域 $\mathcal{R}$
$(r_0, m_0) \leftarrow$ （初始区域，空闲）
当 $p(t) < P$ **时循环** ▷每个调度步骤
  如果SAFETYNET(t)**那么** ▷ §4.2
    启动订单如果 $m \neq$ 订单；**继续**
  PROBEREGIONS($\mathcal{R}$) 定期 ▷ §4.3
  价值 $\leftarrow$进度条价值估算估计$(t, T, p(t), P)$ ▷§4.5
  对于每个状态 $s \in \mathcal{R} \times$ {竞价型实例，订单、空闲}**执行**
    $\bar{L}_s \leftarrow$预测生命周期(s) ▷ §4.4
    $U_s \leftarrow$计算效用$(s, V, \bar{L}_s)$ ▷ §4.6
  按 U_s 降序排列候选状态
  对于每个 $s = (r, m)$ 且 $U_s > U_{(r_0, m_0)}$ **执行** ▷ §4.7
    成功后 $\leftarrow$ **启动** s **或** m =空闲
    如果 成功**则**
      终止 (r_0, m_0) 如果 $m_0 \neq$ 空闲
      $(r_0, m_0) \leftarrow (r, m)$ ； **跳出**
终止 (r_0, m_0) ▷ 作业完成后终止

（空闲总能成功；竞价型实例可能因不可用而失败）。这种方法确保系统能够随着条件变化，持续转向更优的选项。

5 SkyNomad 实现

我们实现了 SkyNomad，这是一个通用的、感知截止时间的批处理作业执行系统，可结合多个区域的竞价型实例和按需实例，在保证按时完成作业的同时最大限度地降低成本。SkyNomad 与工作负载无关，支持任何定期将检查点保存到持久化存储的工作负载，并能在跨区域迁移时无缝管理检查点迁移。 SkyNomad 构建于 SkyPilot 之上 [54], 这是一个开源多云系统，并在其之上增加了一个调度器，代码量为 ~6,000 行。该实现支持常见的训练框架，包括 PyTorch [57]和 Hugging Face Transformers [48]。

调度器。 调度器负责管理作业的整个生命周期。它首先定期执行探测，以收集用于（§4.3）的数据。探测间隔可配置，默认设置为 2 小时。我们发现，这足以捕捉竞价市场的关键特征，同时将探测开销控制在合理范围内（§6.1）。随后，调度器会将探测结果传递给策略，并根据策略的决策配置实例。

检查点迁移。 SkyNomad 假设工作负载会定期将检查点保存到持久化存储（例如云存储桶），并在作业迁移到其他区域时处理检查点迁移。由于公有云上的数据必须在实例完成资源配置后加载到实例上，SkyNomad 实现了两阶段流水线以加快迁移速度。在目标实例进行资源配置期间，SkyNomad 首先将检查点复制到目标区域的对象存储中。实例可用后，SkyNomad 会在运行时设置期间下载检查点，以便作业尽快恢复执行。

6 评估

我们通过两种方式全面评估 SkyNomad：一是在会发生实际抢占的真实云部署中进行测试，二是利用重放的竞价实例轨迹进行模拟。我们旨在回答以下问题：

- SkyNomad 能否 在真实云环境中降低成本，同时仍能在截止时间内完成作业？（§6.1）
- SkyNomad 能否 适用于不同的加速器类型及其抢占模式？（§6.1和 §6.2.2）
- SkyNomad 能否 在截止期限紧迫度、区域数量和检查点大小各不相同的情况下，始终实现低成本？（§6.2.3至 §6.2.5）
- SkyNomad 能否 适用于不同的地理部署和数据主权约束？（§6.2.6）

6.1 端到端实验

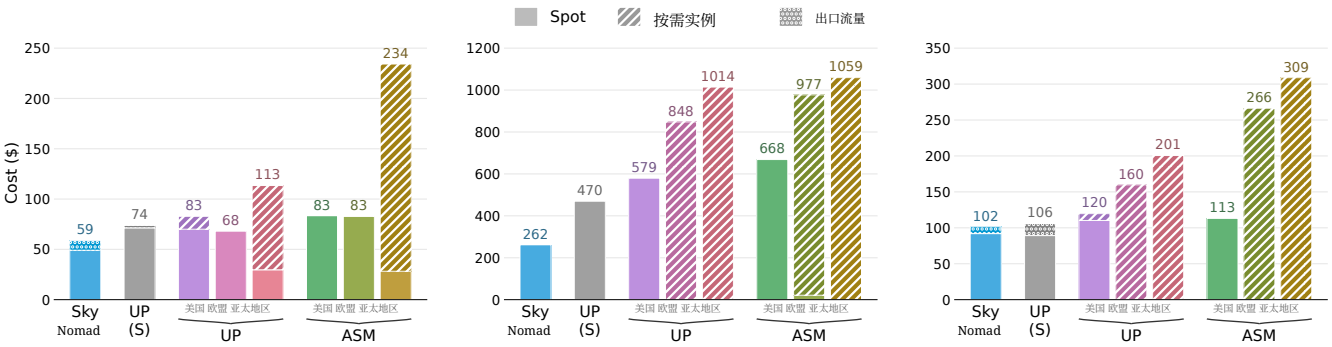
我们在 AWS 上使用三种不同的加速器配置对大语言模型进行微调，开展端到端实验：4×L4、8×A100 和 4×A10G。我们同时启动所有基线系统和 SkyNomad，使它们经历相同的实时竞价实例抢占事件。这些实验的总成本约为 8,000 美元。

基线。 我们将 SkyNomad 与以下系统进行比较：

- **均匀进度（UP）** [50]：一种单一区域的截止时间感知策略，会随时间均匀分配进度，并交替使用按需实例和竞价实例，以保证按时完成。
- **AWS SageMaker 托管竞价实例（ASM）** [4]：一种生产级批处理作业执行系统，使用竞价实例来降低成本。ASM 只使用竞价实例，并可从单一区域内的多个可用区获取竞价实例容量，但仅在发生抢占时才切换可用区，且不支持多区域。
- **UP(S)**：我们实现的 UP 多区域扩展，作为基线，代表 SkyPilot 的生产级故障转移策略 [41]。发生抢占时，它会将作业切换到其他区域以避开波动期 [32], , 尝试候选区域时按成本从低到高依次进行，直到成功为止。在两次抢占之间，它遵循 UP 的进度分配策略，以保证按时完成，并利用规则 [50], : 只要当前区域仍然可用，就留在该区域。[3]

我们在三个区域（us-west-2、eu-central-1 和 ap-northeast-1）中分别运行 UP 和 ASM，以研究区域性竞价实例可用性对其性能的影响。

[3]我们还设计了额外的启发式方法（UP(A)、UP(AP)），用于消融 SkyNomad在 §6.2中的。



(a) 4×L4 (g6.12xlarge) (b) 8×A100 (p4d.24xlarge) (c) 4×A10G（g5.12xlarge）

图6：三种加速器类型的端到端成本比较。SkyNomad 在所有配置下都实现了最低成本。单区域基线（UP 和 ASM）的成本会因区域性竞价实例可用性而出现较大波动。只有多区域方案会产生出口成本（以柱顶的虚线区域表示）。A100 的出口成本极低，因为较大的检查点大小（500GB）使跨区域迁移成本高昂，因此系统倾向于在区域内切换可用区。所有作业均在截止期限内完成。

ASM 本身并不具备截止时间感知能力；为确保比较公平，我们手动触发安全网（§4.2），在必要时切换到按需实例，以满足截止时间。

工作负载。我们使用 Hugging Face Transformers [48]对以下模型进行微调：Qwen3 [52]，使用 Orca-Math 数据集 [35]：Qwen3-4B 运行于 4×L4（`g6.12xlarge`），以及4×A10G（`g5.12xlarge`）；Qwen3-14B 运行于 8×A100（`p4d.24xlarge`）。每个作业都在单个实例上运行，需要约 30 小时的计算时间，截止时间为 45 小时。检查点大小分别为 100 GB 和 500 GB。冷启动时间（包括实例配置、环境设置和检查点加载）约为 6 分钟。

结果。 图 6 展示了三种加速器配置下各系统的总成本。`SkyNomad` 在所有情况下的成本都是最低的：相比最佳单区域基线节省 10–55%，相比平均单区域基线节省 47–69%，相比多区域基线 UP(S) 节省 4–44%。这些节省已计入`SkyNomad` 的开销：跨区域迁移的出口成本最高可占总成本的 16%（图6中的点状部分），而探测成本可以忽略不计，每个作业仅为 1–3 美元。尽管存在出口开销，跨区域迁移仍使`SkyNomad` 能够利用原本无法获取的低价竞价实例容量。

单一区域解决方案本质上存在局限。单一区域基线有两项局限。首先，竞价市场瞬息万变，没有任何单一区域能够始终提供最佳可用性或价格。某个区域在作业开始时表现良好，但在执行过程中可能变得不可用，而静态的区域选择无法适应此类变化。在我们针对三个具有代表性的区域所做的实验中，UP 的总成本最高相差 1。7× （L4 上；图 6a ）和 1。8× （ A100 上；图 6b ）。虽然表现最佳的区域 `eu-central-1` 在 L4 上（图 6a）凭借充足的竞价实例容量实现了较低成本，但表现最差的区域（`ap-northeast-1`）超过 70% 的时间没有可用的竞价型实例，因此不得不转而使用昂贵的按需实例，使其总成本中有 74% 来自这部分开销。ASM 的成本差异更大（A10G 上最高相差 2。8×），因为其区域级故障转移可能会切换到更昂贵的区域（图 6c）。其次，不同加速器类型的竞价实例可用性模式各不相同：`eu-central-1` 在 L4 上表现良好，但在 A100 上表现不佳；而 `us-west-2` 则恰好相反。因此，根据一种 GPU 类型确定的区域选择未必适用于另一种类型。鉴于这些局限，即使是表现最佳的单一区域基线（UP 在 `eu-central-1`）的成本也比 `SkyNomad`高 10%。

朴素的多区域方案还不够。 通过从多个区域获取竞价实例容量，多区域方案能够实现更低且更稳定的成本；`SkyNomad` 和 UP(S) 都受益于跨区域汇聚的竞价资源，并且无需使用任何按需实例即可完成作业（图 6a）。然而，UP(S) 只会在被抢占时迁移。如果它落在一个价格昂贵但可用性恰好稳定的区域，就会一直留在那里。因此，尽管能够访问全部三个区域，UP(S) 在 L4 上的表现仍不如最佳单区域基线（图6a）。 `SkyNomad`则不同，它通过探测持续监控所有候选可用区，并在预期节省超过迁移成本时主动迁移。这使得 `SkyNomad` 在 A100 上的成本比 UP(S) 低 44%（图 6b）。例如，在图 7中，持续探测价格较低的可用区`us-east-2b/c` （1.8 美元/小时）检测到第 23–25 小时出现可用性窗口，并预测其持续时间为 3.5 小时。根据统一成本模型，该可用区的效用高于 `eu-central-1a`，因此触发了价格感知主动迁移（□ 标记）。这种实时可见性使`SkyNomad` 区别于 UP(S) 等反应式策略。

`SkyNomad` 纵观全局。与仅考虑单一指标（例如价格或可用性）的启发式方法不同，SkyNomad 会综合考虑价格、预测的生命周期、出口流量

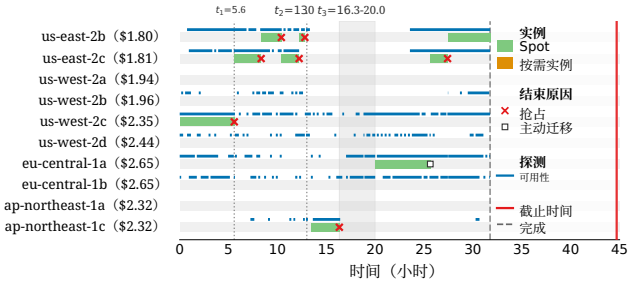
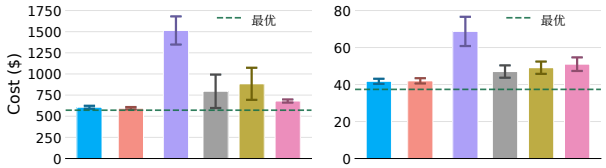


图 7：L4 实验中 SkyNomad 的迁移轨迹。绿色条表示竞价实例的使用情况；蓝色区段表示探测到的可用性；□ 标记表示主动迁移决策。



(a) H100 (b) V100

图 8：两条独立竞价实例轨迹上的成本：(a) GCP 上的 H100 和 (b) AWS 上的 V100。绿色水平线表示最优策略的成本。SkyNomad 可推广至不同的云服务提供商和 GPU 类型。我们模拟了 20 个开始时间各异的作业；误差线表示标准误。

成本和截止期限压力。我们重点介绍 L4 跟踪记录中的三个决策（图 7），以作说明。

(1) $t_1 = 5.6$**小时：** 在 us-west-2c 被抢占后，SkyNomad 会评估所有可用区域。探测结果显示，us-east-2c（$1.81/小时）可用，预测生命周期为 4 小时。此时，$V(t) \approx$ $2.6/小时；计入冷启动开销（6 分钟）后，每小时有效进度的价值为 $2.5。$2 的出口成本分摊到 4 小时的生命周期中，相当于每小时 $0.50。每小时综合成本为 $1.81 + $0.50 =$2.31/小时，低于进度的货币价值，因此效用为正，SkyNomad 遂进行迁移。

(2) $t_2 = 13.0$**小时：** 在 us-east-2 经历 4 次抢占后（第 9–13 小时），作业已落后于计划，并承受截止期限压力$\theta(t) = 0.73$（相较于名义值 $P/T = 0.67$；参见 §4.5），使$V(t)$ 升至 $\approx$$4/小时。在进度价值升高后，即便成本更高的 ap-northeast-1c （$2.32/小时）也能产生正效用。同时，系统检测到 us-east-2 正处于波动期（$\gamma^* > 1$），其预测生命周期因此缩短至不足 1 小时。SkyNomad 于是选择 ap-northeast-1c 作为唯一具有正效用的区域，并接受 $2 的出口成本。

(3) $t_3 = 16.3$–20.0**小时：** 在 ap-northeast-1c 发生竞价实例抢占后，SkyNomad 首先尝试在价格低廉的 us-east-2 区域（$1.80/小时）启动（效用最高），但这些区域均不可用。其他可用区域，例如 us-west-2c（$2.35/小时），似乎处于波动期；考虑到$V(t) \approx$ $2.6/小时，这些区域的效用接近于零或为负，而空闲的效用为 $U = 0$。SkyNomad 选择等待；随着截止期限压力增加，$V(t)$不断上升，直到第 20 小时，eu-central-1a（$2.65/小时）终于产生正效用，此时 SkyNomad 便在那里启动作业。

总结。 端到端实验证实，SkyNomad 能够满足所有截止期限，并通过其统一成本模型，在成本方面平均比单区域基线节省 55%（最高可达 70%）。该模型会权衡价格、可用性、迁移成本和截止期限压力。

6.2 基于竞价实例可用性轨迹的仿真

为了了解 SkyNomad 在各种条件下能否始终保持良好表现，我们在仿真中回放了真实的竞价实例可用性轨迹。这样，我们便能大规模、系统地研究截止期限紧迫度、区域数量和检查点大小等参数对策略的影响。

6.2.1 实验设置

轨迹。我们收集了以下实例的竞价实例可用性轨迹： 16 1× H100 实例（a3-highgpu-1g），在为期 14 天的时间里，每 10 分钟探测 13 个 GCP 可用区，总成本约为 9,000 美元。我们还使用公开可用的 1×AWS 上 V100 实例的轨迹 [50] 以验证其泛化能力。

基线。我们将 SkyNomad 与以下方案进行比较：

- **最优**：一种全知策略，完全了解未来的竞价实例可用性。我们使用动态规划计算其最优决策，这些决策为完成作业所能达到的成本提供了下界。
- **SkyNomad (o)**：SkyNomad 的一个变体，它使用预言机获取每个区域中下一个竞价实例生命周期的信息。
- **UP**：单区域均匀进度 [50], 与 §6.1中的方案相同。我们报告所有区域的平均成本。
- **UP(S)**：与 §6.1中的方案相同。
- **UP(A)**：UP 的多区域扩展，采用与 SkyNomad （§4.3）相同的探测机制，并根据观测到的可用性选择区域。观测到的可用性定义为最近 5 个样本的滑动窗口中成功探测所占的比例。
- **UP(AP)**：UP 的多区域扩展，根据可用性与竞价型实例价格的比值选择区域，在平衡可用性与成本的同时不进行生命周期预测。

默认参数。除非另有说明，我们将作业时长设为 100 小时，截止时间设为 150 小时，检查点大小设为 50GB，冷启动开销设为 6 分钟。默认使用 8 个区域。

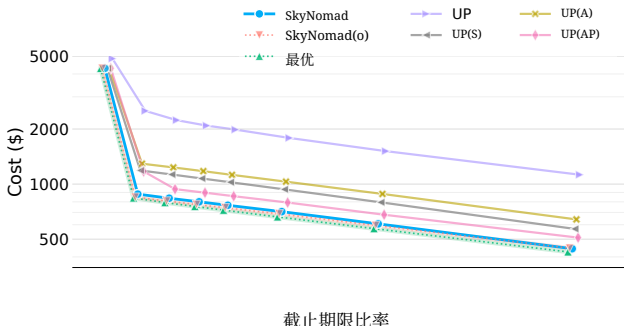


图 9：不同截止时间比率下的成本。y 轴采用对数刻度。比率为 1.0 时即为理论下界。[4]

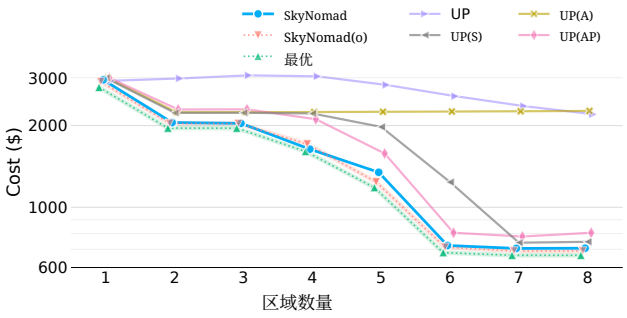


图 10：不同可用区域数量下的成本。对于所有系统，可用区域越多，成本就越单调下降。在所有情况下，SkyNomad 都能实现接近最优的成本。

6.2.2 跨加速器的泛化能力

图 8 比较了两条相互独立的竞价实例轨迹上的策略性能：来自 GCP 的 ourH100 轨迹，以及来自 AWS、公开可用的 V100 轨迹 [50]。

SkyNomad 在两组跟踪数据上均实现了接近最优的成本：H100 上为 610 美元（比最优值高出 11%），V100 上为 42 美元（比最优值高出 12%）。这表明我们的方法能够泛化到不同的云服务提供商和 GPU 类型。

我们将选择准确率定义为策略使用可用性最高区域的时间占比，用以衡量其在整个执行过程中识别并利用最佳选项的能力。SkyNomad在两组跟踪数据上的选择准确率均达到 83–100%。相比之下，UP(A) 从未选择价格最低的区域，因此在 V100 和 H100 上的选择准确率均为 0%：无论价格如何，它总是选择可用性最高的区域。这解释了成本差异：统一成本模型的价格项（§4.6）会引导选择转向成本效益高的区域，而仅考虑可用性的启发式策略则完全忽略价格。

SkyNomad 与SkyNomad 的差距SkyNomad (o) 在两组跟踪数据上均低于 5%，区域选择重合率为 95–99%，这表明SkyNomad的生命周期预测所产生的迁移决策与预言机几乎完全相同。由于SkyNomad (o) 使用预言机来获取竞价实例生命周期，这一微小差距证实，我们基于生存分析的生命周期预测（§4.4）对生命周期的估算足够准确，能够做出近乎最优的决策。在 V100 上，SkyNomad 的表现略优于SkyNomad(o)；这处于仿真的方差范围内。

6.2.3 截止期限紧迫度的影响

图 9 展示了成本如何随截止期限紧迫度而变化。截止期限比率定义为 T/P，其中 T 是截止时间，P 是作业时长。比率越大，等待竞价实例容量的灵活性就越高。

当截止期限比率为 1.0 时，所有策略都会立即触发安全网（§4.2），并完全使用按需实例运行，成本收敛至 $4,000–$5,000。随着空闲时间增加，SkyNomad

[4]在实践中，冷启动开销要求比率略高一些 tio.

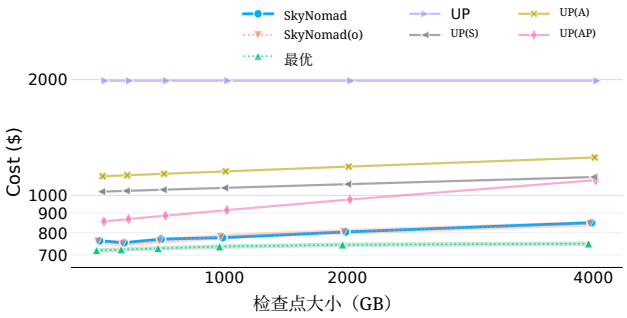


图 11：不同检查点大小下的成本。检查点越大，迁移成本越高，从而降低跨区域调度的收益。SkyNomad 通过将迁移成本分摊到预测生命周期内，实现了平稳扩展；而 UP(AP) 等启发式方法则因频繁迁移而表现不佳。

与最优方案紧密吻合，在 2.0× 时将成本降至 $420，并且在所有测试比率下都与最优方案相差不超过 10%。

单区域 UP 的扩展性较差，即使在 2.0× 的空闲时间下，成本仍为 1,100 美元（2.6×SkyNomad）。多区域策略通过从多个区域调取容量，扩展性更好，但成本仍比 SkyNomad 高 15–30%。这种差距并非源于按需实例回退，因为多区域策略很少触发安全网。相反，它们选择了价格昂贵的区域：UP(A) 贪心地选择可用性高的区域，而不考虑价格（在亚洲的成本最高达 3.6× ）；UP(AP) 和 UP(S) 则会对抢占作出响应，却不考虑预测生命周期。

6.2.4 区域数量的影响

图10 展示了候选区域数量从 1 增至 8 时成本的变化情况。我们按照竞价实例平均可用性的高低顺序逐步添加区域。

只有一个区域时，由于无需优化区域选择，所有策略的表现都相近（$2,800–$3,000）。随着区域数量增加，SkyNomad 的表现逐渐接近最优值：在 8 个区域时，SkyNomad 的成本为 $707，比最优值（$666）高出不到 6%，并

SkyNomad (o) 的 2%（$691）。

简单的启发式方法则表现停滞或不佳。SkyNomad 与最优策略的区域选择重合率达到 95%，而 UP(A) 仅为 0.2%。UP(A) 会停留在可用但价格昂贵的区域，且从不迁移离开：在 8 个区域的场景中，UP(A) 有 94% 的时间选择 asia-south2-b（可用性为 95%，但 4× 价格最便宜），导致成本比最优策略高出 4.8× 。UP(S) 只考虑价格而忽略可用性，整体表现通常不如 UP(AP)。UP(AP) 在平衡可用性与价格方面更接近 SkyNomad（8 个区域时成本为 $ 806），但仍无法匹敌我们的统一成本模型。

超过 6 个区域后，收益递减：`SkyNomad` 和 Opti-mal 的表现持续提升，直至达到 6 个区域；之后，汇总可用性趋于饱和，新增区域的可用性窗口会彼此重叠，而非提供互补容量。对于受地理限制的用户（§6.2.6），即使只选择 2–3 个精心挑选的区域，也足以显著节省成本。

6.2.5 检查点大小的影响

检查点大小会影响迁移成本，因为出口成本与传输的数据量成正比。图 11将检查点大小从 0 调整到 4TB，以涵盖从索引状态较小的批量推理到大规模训练（完整模型和优化器状态）等不同工作负载。

单区域 UP 不受检查点大小影响，因为它从不跨区域迁移。SkyNomad 可平稳扩展：在 4 TB 时，SkyNomad 的成本为 $850，比成本为 $2,000 的 UP 低 57%；与 UP(AP) 相比，它将迁移频率降低了 70%，优先等待竞价型实例恢复，而不是承担昂贵的跨区域传输成本。SkyNomad 的成本从 200 GB 到 500 GB 略有下降，这处于模拟方差范围内。

对频繁变化的信号做出反应的启发式方法，在检查点间隔较长时表现不佳。UP(AP)的成本增长最为显著：随着新探测结果到来，可用性会发生变化；在廉价区域中，除以价格会放大这些波动，因此可用性的微小变化就可能颠倒比率排名并触发区域切换。UP(A)更稳定，因为高可用性区域往往会继续保持高可用性。UP(S)最为稳定，因为价格很少变化，所以它会回到同一批廉价区域。

6.2.6 数据主权要求的影响

数据主权要求（例如 GDPR [22]）可能会将作业限制在特定地理区域内，以防止数据离开指定区域。我们通过以下方式模拟此类约束：
将候选区域限制在同一洲内。图12评估了 `SkyNomad` 在四种配置下的表现：仅美国（3 个区域）、仅欧洲（2 个区域）、[5] 仅亚洲（3 个区域）和全球（全部 13 个区域）。`SkyNomad` 即使受到地理限制也表现出色：在美国（3 个区域）的成本为 700 美元，接近

[5]由于配额限制，我们只能访问欧洲的两个区域。

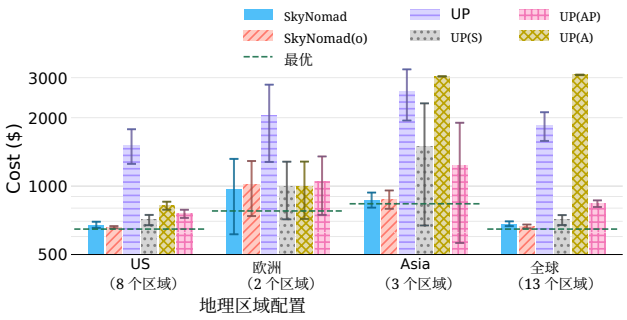


图12：地理限制下的成本（仅美国、仅欧洲、仅亚洲和全球）。SkyNomad 能够有效适应所有配置，即使区域多样性有限，也能大幅节省成本。误差线表示 20 个作业的 95% 置信区间。

全局最优值 650 美元；在欧洲（2 个区域）的成本为 950 美元，仍比 UP 便宜 52%；在亚洲（3 个区域）的成本为 870 美元，比 UP 便宜 68%。即使只有 2–3 个区域，`SkyNomad`也能显著节省成本，这证实只需数量不多但经过精心选择的区域即可（§6.2.4）。由于可用性模式差异较大，亚洲的方差最大。UP(A) 的表现尤其糟糕（3,038 美元，3。6× 最优），因为它 100% 的时间都选择了成本最高的区域（`asia-south2-b`），选择准确率为 0%。相比之下，`SkyNomad` 始终选择成本最低的区域（`asia-southeast1-b`），选择准确率达到 100%，成本仅比最优值高 0.1%。

7 相关工作

用于训练和批处理作业的竞价型实例。人们已广泛研究如何利用竞价型实例降低批处理计算的成本，包括高性能计算 [45, 55]，分析 [40, 51]，和机器学习训练 [8, 13, 23, 29, 43]。例如，Bamboo [43] 通过冗余计算提供弹性，Varuna [8] 动态调整训练配置，而 Parcae [13]则预测竞价实例可用性并实时迁移任务。然而，这些系统旨在最大化吞吐量，却不保证能够按时完成作业。它们完全依赖竞价型实例，因此，如果竞价实例容量长时间消失（可能持续数天 [32]），进度就会停滞，也无法保证按时完成。

截止期限感知的竞价实例调度。均匀进度（UP） [50] 在竞价型实例可用时使用它们，并随着截止时间临近而转用按需实例，相比仅使用按需实例可节省 27–84% 的成本。然而，UP 仅在单一区域内运行，无法利用多区域竞价实例容量：当一个区域没有可用的竞价型实例时，另一个区域可能仍有容量。Proteus [19] 提出了采用临时服务器的分层可靠性方案，但也仅在单一区域内运行。Snape [53] 使用强化学习在竞价型实例和按需虚拟机之间分配容量，但面向的是具有服务级别目标要求的长时间运行服务，而非具有截止期限的批处理作业——

而非具有截止期限的批处理作业。 SkyNomad 首次将截止期限保证与多区域竞价型实例调度结合起来，利用时空异质性来最大限度地降低成本。**用于服务的竞价实例。** MArk 等系统 [56], Cocktail [17], 和 Tributary [21] 使用竞价型实例提供具有响应时间服务级别目标的推理服务。SpotServe [33]将大语言模型推理并行分布于竞价型 GPU 上，并适应抢占；而 SkyServe [32] 则在多个区域复制实例，以维持可用性。然而，服务系统无法利用空闲时间：为维持延迟服务级别目标，一旦竞价型实例消失，就必须立即配置按需实例。相比之下，批处理作业可以等待竞价实例容量恢复，从而在不牺牲截止期限要求的前提下进一步降低成本。

8 讨论

多云扩展。 SkyNomad 目前仅在单一云服务提供商内运行。扩展至多云环境可进一步扩大竞价型实例容量池，但也会带来一些挑战，包括 API 异构、较高的跨云迁移成本，以及特定提供商的探测语义。我们将此留作未来工作。

可用性信号。 我们使用轻量级探测作为可用性信号，每个作业仅产生少量成本（1–3 美元）。云服务提供商开始提供原生可用性信号（例如 AWS 竞价型实例放置评分 [7]），这有望降低探测成本，但会牺牲可移植性。

自动扩缩集成。 为确保兼容性，SkyNomad 使用固定数量的实例，并采用成组调度抢占。将多区域调度与弹性自动扩缩容 [8, 23, 33, 43] 相结合，或可进一步节省成本，但这需要针对特定工作负载支持动态并行处理。

抢占后的进度损失。 发生抢占时，自上一个检查点以来的计算进度会丢失。SkyNomad 的生命周期预测可用于指导自适应检查点机制 [18, 36],：当预测剩余生命周期较短时，系统将更频繁地创建检查点。我们将此留作未来工作。

任务即服务定价模型。 单区域调度的成本波动较大。多区域调度借助地理多样性，能够提供更稳定的成本，或可实现任务即服务定价模型，从而消除竞价市场波动带来的影响。

9 结论

SkyNomad 表明，利用竞价型实例在空间和时间上的异质性，可以大幅降低人工智能批处理作业的成本，同时满足截止期限要求。借助实时探测、生命周期预测，以及纳入截止期限压力和运营成本的统一成本模型，SkyNomad 会持续选择最具成本效益的区域，并在有利时进行迁移。在实际部署中，与现有研究和生产系统相比，SkyNomad 可将成本降低 1.25–3.96× 。

参考资料

[1] 网络服务层级定价，2025年。访问日期：2026年1月10日。

[2] Odd Aalen。计数过程族的非参数推断。《统计年刊》，6(4):701–726，1978年。

[3] Orna Agmon Ben-Yehuda、Muli Ben-Yehuda、Assaf Schuster 和 Dan Tsafrir。剖析亚马逊 EC2 竞价实例定价。*ACM*经济学与计算汇刊（*TEAC*），1(3):1–20，2013年。[4] 亚马逊。Amazon SageMaker。访问地址：https://aws.amazon.com/sagemaker/，2025年。访问日期：2026年1月。[5] 亚马逊云科技。Amazon EC2 竞价型实例定价。https://aws.amazon.com/ec2/spot/pricing/，2025年。访问日期：2026年1月。[6] 亚马逊云科技。EC2 竞价型实例最佳实践。https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/spot-best-practices.html，2025年。访问日期：2026年1月。[7]亚马逊云科技。竞价型实例放置评分。https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/spot-placement-score.html，2025年。访问日期：2026年1月。[8] Sanjith Athlur、Nitika Saran、Muthian Sivathanu、Ramachandran Ramjee 和 Nipun Kwatra。Varuna：大规模深度学习模型的可扩展、低成本训练。载于第十七届欧洲计算机系统会议论文集（*EuroSys '22*），第472–487页。ACM，2022年。[9] Tom B. Brown、Benjamin Mann、Nick Ryder、Melanie Subbiah、Jared 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。语言模型是少样本学习器，2020年。

[10] Tanujit Chakraborty、Swarup Chattopadhyay、Suchis- mitaDas、UttamKumar 和 J.Senthilnath。用于建模的重尾概率分布搜索

现实世界复杂网络。*IEEE Access*，10:115092– 115107，2022年。

[11] Aakanksha Chowdhery、Sharan Narang、Jacob Devlin、Maarten Bosma、Gaurav Mishra、Adam Roberts、Paul Barham、Hyung Won Chung、Charles Sutton、Sebastian Gehrmann 等。Palm：利用路径扩展语言建模。机器学习研究杂志，24(240):1–113，2023年。

[12] CloudZero。AWS 出站成本解析：如何降低支出，2024年7月。访问日期：2026年1月10日。

[13] Jiangfei Duan、Ziang Song、Xupeng Miao、Xiaoli Xi、Dahua Lin、Harry Xu、Minjia Zhang 和 Zhihao Jia。《 Parcae：在可抢占实例上进行主动式、实时迁移优化的深度神经网络训练》。载于 第 *21* 届 *USENIX* 网络系统设计与实现研讨会论文集，NSDI' 24，美国，2024年。USENIX 协会。[14] Nnamdi Ekwe-Ekwe 和 Adam Barker。《地点，地点，还是地点：探索亚马逊 EC2 竞价实例在不同地理区域的定价》。载于 第 *18* 届 *IEEE/ACM* 集群、云与网格计算国际研讨会论文集（*CCGrid '18*），第 370–373 页。IEEE，2018年。[15] Google Cloud。Spot 虚拟机定价。https://cloud.google.com/spot-vms/pricing，2025年。访问日期：2026年1月。[16] Aaron Grattafiori 等。《Llama 3 模型家族》，2024年。[17] Jashwant Raj Gunasekaran、Prashanth Thinakaran、Deepak Narayanan、Ramanathan Kannan、Gabriel Parmer、Prakash Mysore 和 Anand Sivasubramaniam。《Cocktail：云端模型服务的多维优化》。载于 第 *19* 届 *USENIX* 网络系统设计与实现研讨会（*NSDI 22*），第 1041–1057 页。USENIX 协会，2022年。[18] Tanmaey Gupta、Sanjeev Krishnan、Rituraj Kumar、Abhishek Vijeev、Bhargav Gulavani、Nipun Kwatra、Ramachandran Ramjee 和 Muthian Sivathanu。《即时检查点：以低成本从深度学习训练故障中恢复》。载于 第十九届欧洲计算机系统会议论文集，第 1110–1125 页，2024年。[19] Aaron Harlap、Alexey Tumanov、Andrew Chung、Gregory R. Ganger 和 Phillip B. Gibbons。《Proteus：在动态资源市场中通过分层可靠性实现灵活的机器学习弹性》。载于 第十二届欧洲计算机系统会议论文集（*EuroSys '17*），第 589–604 页。ACM，2017年。

[20] Jordan Hoffmann、Sebastian Borgeaud、Arthur Mensch、Elena Buchatskaya、Trevor Cai、Eliza Rutherford、Diego de Las Casas、Lisa Anne Hendricks、Johannes Welbl、Aidan Clark、Tom Hennigan、Eric Noland、Katie Millikan、George van den Driessche、Bogdan Damoc、Aurelia Guy、Simon Osindero、Karen Simonyan、Erich Elsen、Jack W. Rae、Oriol Vinyals 和 Laurent Sifre。《训练计算最优大型语言模型》，2022年。

[21] Jiawei Hu、Sayan Ghosh、Wuxinlin Jiang、Wenqi Zheng、Ling Zhuo、Narayanan Natarajan、Abhishek Chandra 和 Ramesh Kannan。Tributary：弹性云服务的竞价实例舞蹈。载于第十八届欧洲计算机系统会议（*EuroSys '23*）论文集，第410–426页。ACM，2023年。

[22] Ahmed Issaoui、Jonas Örtensjö 和 M. Sirajul Islam。《探索云服务中的通用数据保护条例（GDPR）合规性：瑞典公共机构对隐私合规的见解》。《未来商业期刊》，9(1):107，2023年。[23] Insu Jang、Zhenning Yang、Zhen Zhang、Xin Jin 和 Mosharaf Chowdhury。《Oobleck：利用流水线模板实现大型模型的弹性分布式训练》。载于第*29*届 *ACM*操作系统原理研讨会（*SOSP '23*）论文集，第382–395页。ACM，2023年。[24] JCS Kadupitiya、Vikram Jadhao 和 Prateek Sharma。《临时云虚拟机时间约束抢占建模》。载于第*29*届高性能并行与分布式计算国际研讨会（*HPDC*）论文集，第41–52页，2020年。[25] Jared Kaplan、Sam McCandlish、Tom Henighan、Tom B. Brown、Benjamin Chess、Rewon Child、Scott Gray、Alec Radford、Jeffrey Wu 和 Dario Amodei。《神经语言模型的缩放定律》，2020年。[26] Ali Khajeh-Hosseini、David Greenwood 和 Ian Sommerville。《云迁移：将企业IT系统迁移至基础设施即服务的案例研究》。载于 *2010*年 *IEEE* 第三届国际云计算会议，第450–457页。IEEE，2010年。[27] KyungHwan Kim 和 Kyungyong Lee。《让云竞价实例中断事件变得可见》。载于*ACM*网络会议*2024*（*WWW '24*）论文集，第3490–3501页。ACM，2024年。[28] Dinesh Ku mar、Gaurav Baranwal、Zahid Raza 和 Deo Prakash Vidyarthi。《云计算中竞价定价综述》。《网络与系统管理期刊》，26(4):809–856，2018年。

[29] Kyungyong Lee 和 Myungjun Son。《DeepSpotCloud：利用跨区域 GPU 竞价实例进行深度学习》。载于*2017*年 *IEEE* 第十届国际云计算会议（*CLOUD*），第98–105页。IEEE，2017年。

[30] Sungjae Lee、Jaeil Hwang 和 Kyungyong Lee。Spot-lake：多样化竞价实例数据集归档服务。载于*2022*年 *IEEE* 国际工作负载特征研讨会（*IISWC*），第 242–255 页。IEEE，2022年。

[31] Will E. Leland、Murad S. Taqqu、Walter Willinger 和 Daniel V. Wilson。以太网流量的自相似性。载于通信架构、协议与应用会议论文集，SIGCOMM '93，第 183–193 页，美国纽约州纽约市，1993年。计算机协会。

[32] Ziming Mao、Tian Xia、Zhanghao Wu、Wei-Lin Chi- ang、Tyler Griggs、Romil Bhard waj、Zongheng Yang、Scott Shenker 和 Ion Stoica。Skyserve：利用竞价实例跨区域和云提供人工智能模型服务。载于第二十届欧洲计算机系统会议论文集，EuroSys '25，第 159–175 页，美国纽约，2025年。计算机协会。[33] Xupeng Miao、Chunan Shi、Jiangfei Duan、Xiaoli Xi、Dahua Lin、Bin Cui 和 Zhihao Jia。Spotserve：在可抢占实例上提供生成式大语言模型服务。载于第 *29* 届 *ACM* 编程语言与操作系统体系结构支持国际会议论文集，第 *2* 卷，ASPLOS '24，第 1112–1127 页，美国纽约，2024年。计算机协会。[34] Microsoft Azure 网络团队。Azure 数据传输定价指南，2025年2月。访问日期：2026年1月10日。[35] Arindam Mitra、Hamed Khanpour、Corby Rosset 和 Ahmed Awadallah。Orca-math：释放小型语言模型在小学数学中的潜力，2024年。[36] Jayashree Mohan、Amar Phanishayee 和 Vijay Chi- dambaram。CheckFreq：频繁、细粒度的深度神经网络检查点保存。载于第 *19* 届 *USENIX* 文件与存储技术会议（*FAST 21*），第 203–216 页。USENIX 协会，2021年。[37] Deepak Narayanan、Mohammad Shoeybi、Jare d Casper、Patrick LeGresley、Mostofa Patwary、Vijay Korthikanti、Dmitri Vainbrand、Prethvi Kashinkunti、Julie Bernauer、Bryan Catanzaro 等。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、存储与分析会议论文集，第 1–15 页，2021年。

[38] Samyam Rajbhandari、Jeff Rasley、Olatunji Ruwase 和 Yuxiong He。Zero：面向万亿参数模型训练的内存优化。载于*SC20*：高性能计算、网络、存储与分析国际会议，第 1–16 页。IEEE，2020年。

[39] Charles Reiss、Alexey Tumanov、Gregory R Ganger、Randy H Katz 和 Michael A Kozuch。大规模云的异质性与动态性：谷歌跟踪数据分析。载于第三届 *ACM* 云计算研讨会论文集，第 1–13 页，2012年。

[40] Prateek Sharma、Stephen Lee、Tian Guo、David Irwin 和 Prashant Shenoy。Spotcheck：设计一种基于竞价市场的衍生型基础设施即服务云。发表于《第十届欧洲计算机系统会议（*EuroSys '15*）论文集》，第 1–15 页。ACM，2015年。

[41] SkyPilot。SkyPilot 任务 YAML 规范：恢复策略。https://docs.skypilot.co/en/latest/reference/yaml-spec.html，2025年。访问日期：2026年1月。

[42] Byung Chul Tak、Bhuvan Urgaonkar 和 Anand Sivasubramaniam。迁移还是不迁移：云计算的经济学。发表于第三届 *USENIX* 云计算热点研讨会（*HotCloud 11*），2011年。

[43] John Thorpe、Pengzhan Zhao、Jonathan Eyolfson、Yifan Qiao、Zhihao Jia、Minjia Zhang、Ravi Netravali 和 Guoqing Harry Xu。Bamboo：让可抢占实例更具弹性，以经济实惠地训练大型深度神经网络。发表于第二十届 *USENIX* 网络系统设计与实现研讨会（*NSDI 23*），第 497–513 页，马萨诸塞州波士顿，2023年4月。USENIX 协会。

[44] Hugo Touvron、Louis Martin、Kevin Stone、PeterAlbert、Amjad Almahairi、Yasmine Babaei、Nikolay Bashlykov、Soumya Batra、Prajjwal Bhargava、Shruti Bhosale 等人。Llama 2：开放式基础模型与微调聊天模型。*arXiv* 预印本 *arXiv:2307.09288*，2023年。

[45] William Voorsluys 和 Rajkumar Buyya。为计算密集型应用可靠地配置竞价型实例。发表于第*26*届 *IEEE* 高级信息网络与应用国际会议（*AINA*）论文集，第 542–549 页。IEEE，2012年。

[46] Borui Wan、Mingji Han、Yiyao Sheng、Yanghua Peng、Haibin Lin、Mofan Zhang、Zhichao Lai、Menghan Yu、Junda Zhang、Zuquan Song 等。 {ByteCheckpoint}：面向大型基础模型开发的统一检查点系统。发表于第*22*届 *USENIX* 网络系统设计与实现研讨会（*NSDI 25*），第559–578页，2025年。

[47] Zhuang Wang、Zhen Jia、Shuai Zheng、Zhen Zhang、Xin- wei Fu、TS Eugene Ng 和 Yida Wang。Gemini：利用内存检查点实现分布式训练中的快速故障恢复。发表于第*29*届操作系统原理研讨会论文集，第364–381页，2023年。

[48] Thomas Wolf、Lysandre Debut、Victor Sanh、Julien Chaumond、Clement Delangue、Anthony Moi、Pierric Cistac、Tim Rault、Remi Louf、Morgan Funtowicz、Joe Davison、Sam Shleifer、Patrick von Platen、Clara Ma、Yacine Jernite、Julien Plu、Canwen Xu、Teven Le Scao、Sylvain Gugger、Mariama Drame、Quentin Lhoest 和 Alexander Rush。《Transformers：自然语言处理领域的最新进展》。载于*2020*年自然语言处理实证方法会议论文集：系统演示，第38–45页。计算语言学协会，2020年。

[49] Yongji Wu、Wenjie Qu、Xueshen Liu、Tianyang Tao、Yifan Qiao、Zhuang Wang、Wei Bai、Yuan Tian、Ji aheng Zhang、Z. Morley Mao、Matthew Lentz、Danyang Zhuo 和 Ion Stoica。《Lazarus：混合专家模型的弹性、抗故障训练》，2025年。

[50] Zhanghao Wu、Wei-Lin Chiang、Ziming Mao、Zongheng Yang、Eric Friedman、Scott Shenker 和 Ion Stoica。《不能迟到：在截止时间约束下优化竞价实例节省成本》。载于第 *21* 届 *USENIX* 网络系统设计与实现研讨会（*NSDI 24*），第 185–203 页，加利福尼亚州圣克拉拉，2024年4月。USENIX 协会。

[51] Yan Yan、Yongqiang Gao、Yang Chen、Zonghua Guo、Binjie Chen 和 Thomas Moscibroda。《TR-Spark：面向大数据分析的瞬态计算》。载于第七届 *ACM* 云计算研讨会论文集（*SoCC '16*），第 484–496 页。ACM，2016年。

[52] An Yang 等。《Qwen3 技术报告》，2025年。

[53] Fangkai Yang、Lu Wang、Zhenyu Xu、Jue Zhang、Liqun Li、Bo Qiao、Camille Couturier、Chetan Bansal、Soumya Ram、Si Qin、Zhen Ma、Inigo Goiri、Eli Cortez、Terry Yang、Victor Ruhle、Saravan Rajmohan、Qingwei Lin 和 Dongmei Zhang。《Snape：结合竞价实例与按需实例的可靠低成本计算》。载于第 *28* 届 *ACM* 编程语言与操作系统架构支持国际会议论文集（*ASPLOS '23*），第 631–643 页。ACM，2023年。

[54] Zongheng Yang、Zhanghao Wu、Michael Luo、Wei-Lin Chiang、Romil Bhardwaj、Woosuk Kwon、Siyuan Zhuang、Frank Sifei Luan、Gautam Mittal、Scott Shenker 和 Ion Stoica。《SkyPilot：天空计算的云间经纪服务》。发表于第*20*届 *USENIX* 网络系统设计与实现研讨会 *(NSDI 23)*，第437–455页，马萨诸塞州波士顿，2023年4月。USENIX 协会。

[55] Sangho Yi、Derrick Kondo 和 Artur Andrzejak。在亚马逊弹性计算云中通过检查点降低竞价实例成本。载于*2012 IEEE* 第三届国际云计算会议（*CLOUD*），第 236–243 页。IEEE，2010年。

[56] Chengliang Zhang、Minchen Yu、Wei Wang 和 Feng Yan。Mark：利用云服务实现成本效益高、感知服务级别目标的机器学习推理服务。载于*2019 USENIX* 年度技术会议（*USENIX ATC 19*），第 1049–1062 页。USENIX 协会，2019年。

[57] 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 和 Shen Li。PyTorch FSDP：全分片数据并行扩展实践。*VLDB Endowment* 论文集，16(12):3848–3860，2023年。