用于自主数据库管理系统调优的统一高效协调框架

张欣怡*†‡北京大学 & 阿里巴巴集团 zhang_xinyi@pku.edu.cn

常卓*†‡北京大学 & 阿里巴巴集团 z.chang@pku.edu.cn

Hong Wu‡阿里巴巴集团 hong.wu@alibaba-inc.com

Yang Li†北京大学 liyang.cs@pku.edu.cn

Jia Chen†北京大学 MaCasK@stu.pku.edu.cn

Jian Tan‡阿里巴巴集团 j.tan@alibaba-inc.com

Feifei Li‡阿里巴巴集团 lifeifei@alibaba-inc.com

Bin Cui†§ 北京大学 bin.cui@pku.edu.cn

arXiv:2303.05710v1 [cs.DB] 10 Mar 2023

摘要

近年来，使用基于机器学习（ML）的技术优化现代数据库管理系统（DBMS）的性能，受到了工业界和学术界的广泛关注。基于机器学习的调优代理旨在调优数据库管理系统的特定组件（例如索引选择、参数调优），并已证明能够找到比经验丰富的数据库管理员更优的配置。然而，一个关键且极具挑战性的问题仍未得到探索：如何让这些基于机器学习的调优代理协同工作。现有方法没有考虑多个智能体之间的依赖关系，而且每个智能体所用的模型只研究单个组件中配置变更的影响。要调优数据库管理系统的不同组件，需要一种协调机制，使多个智能体能够相互了解。此外，我们还需要决定如何在各个智能体之间分配有限的调优预算（例如时间和资源），以最大限度地提升性能。由于每个智能体所对应的奖励（即性能提升）的分布未知且非平稳，因此这一决策难以做出。本文研究了上述问题，并提出一个统一的协调框架，以高效利用现有的基于机器学习的智能体。首先，我们提出一种消息传播协议，用于规定智能体之间的协作行为，并将全局调优消息封装到每个智能体的模型中。其次，我们将研究成熟的强化学习算法汤普森采样与记忆缓冲区相结合，使框架能够在非平稳环境中合理分配调优预算。我们的框架定义了适用于广泛类别的基于机器学习的调优智能体，同时又足够简单，便于集成到现有实现中并支持未来扩展。通过广泛的评估，我们表明，该框架能够有效利用不同的基于机器学习的智能体，并找到更优的配置；与基线方法相比，工作负载执行时间加快了 1.4~14.1× 。

关键词

性能调优，数据管理中的机器学习

ACM 引用格式：

张欣怡、常卓、Hong Wu、Yang Li、Jia Chen、Jian Tan、Feifei Li 和 Bin Cui。2022。用于自主数据库管理系统调优的统一高效协调框架。载于 *SIGMOD '23* 会议论文集：*ACM SIGMOD/PODS* 会议（*SIGMOD '23*）。ACM，纽约，纽约州，美国，15 页。https://doi.org/XXXXXXX.XXXXXXX

1 引言

数据库管理系统（DBMS）在广泛的数据密集型应用中发挥着关键作用。数据库管理系统的性能取决于多个可配置组件，例如配置调优参数、索引设置和物化视图。调优数据库管理系统对于获得高性能至关重要，但这一过程并不简单。通常，数据库管理员（DBA）会投入大量精力，为数据库管理系统寻找合适的设置。然而，由于调优问题具有NP难的性质 [41] ，人工操作难以轻松应对如此复杂的调优任务。随着云环境的转变，云数据库中出现了大规模的实例部署，工作负载也日益多样化。人工调优难以有效扩展。

幸运的是，机器学习（ML）技术能够建模复杂函数并自动化性能调优过程，因此有助于克服这一挑战。研究人员一直在积极探索利用机器学习技术优化数据库系统的方法（即基于机器学习的调优代理）。图 1a 展示了过去五年中聚焦于 DBMS 基于机器学习的调优代理的已发表研究数量。这些工作包括索引选择 [15, 22, 23, 35, 38, 47, 52, 54–56, 63, 63, 64,73], 参数调优 [6, 10, 11, 20, 24, 31–33, 42, 57, 68, 70, 71],视图生成 [27, 46, 65, 67], [13, 28, 36, 36, 50, 60, 查 66, 72], [17 19, 26, 29, 74] 询优化和数据分区 – 。给定一个工作负载，基于机器学习的调优代理旨在依据特定目标函数（例如提高吞吐量和降低延迟）来提升数据库的性能。例如，为了调优

*张欣怡和常卓对本文贡献相同

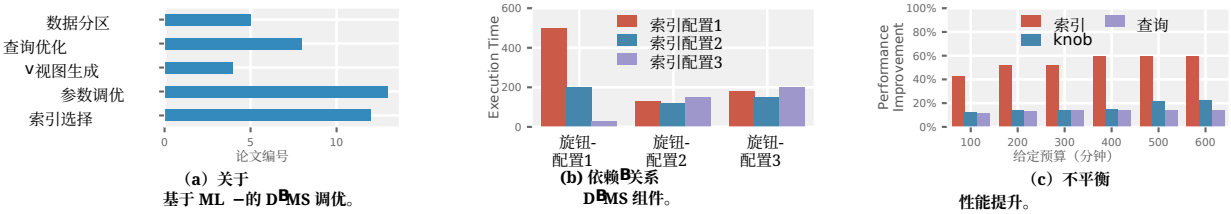
†北京大学计算机科学学院与高可信软件技术重点实验室

‡阿里巴巴集团达摩院数据库与存储实验室 §北京大学计算社会科学研究所（青岛），中国

允许免费制作本作品全部或部分内容的数字或纸质副本，供个人或课堂使用，但不得为营利或商业利益而制作或分发副本，且副本必须包含本声明及首页上的完整引文。对于本作品中由非 ACM 所有的组件，必须尊重其版权。注明出处后允许摘录。若要以其他方式复制、重新出版、发布到服务器或分发至列表，须事先获得特定许可和/或支付费用。请向 permissions@acm.org 申请许可。

SIGMOD '23，*2023*年*6*月*18*日至*23*日，美国华盛顿州西雅图© 2022 计算机协会。ACM ISBN 978-1-4503-XXXX-X/18/06...$15.00

https://doi.org/XXXXXXX.XXXXXXX



（a）关于基于 ML 一的 DBMS 调优。

(b) 依赖关系 DBMS 组件。

（c）不平衡性能提升。

图1：动机示例。图1a展示了2017年至2022年间关于基于机器学习的调优代理的已发表论文数量。图1b展示了旋钮组件和索引组件在不同配置设置下的查询执行时间。图1c展示了三个调优智能体在TPC-H工作负载上，采用不同调优预算时的性能提升。

DBMS 中的配置旋钮，参数调优代理会利用机器学习模型建议一个有希望的配置，并将其应用于 DBMS。随后，该智能体会根据配置的性能更新模型，以学习潜在的目标函数并改进其调优策略。

尽管基于机器学习的调优代理已展现出比人工调优更优异的性能和效率 [30, 39]，但它们始终只专注于调优单个可配置组件，忽视了智能体之间的依赖关系。然而，数据库管理系统的性能受多个组件影响。一个组件的最优配置选择取决于其他组件的配置 [51]。例如，在未构建索引时，最好设置较大的查询缓存和较小的缓冲区大小；而在构建了合适的索引后，则最好设置较小的查询缓存和较大的缓冲池 [51]。因此，运行独立的智能体很可能会推荐次优配置。图1b展示了一个具体示例：在TPC-H工作负载上运行Q19。我们发现，最优配置是旋钮配置1与索引配置3的组合。然而，当数据库管理系统采用旋钮配置2或3时，索引代理会建议次优配置（即索引配置2）。此外，运行某个调优智能体时，其他组件的配置通常并非理想配置（例如该示例中的旋钮配置1）。因此，与独立调优相比，同时调优多个组件更具优势。

多组件调优本质上是一个优化问题，需要在由各个组件子空间构成的联合配置空间中进行搜索。这个复合配置空间庞大而复杂，因此会产生与维度灾难相关的可扩展性问题 [45]。此外，由于不同组件的配置空间各不相同（例如，索引选择的配置空间是二元的，参数调优的是连续的，而查询重写的是树状结构），因此无法直接对它们进行联合优化。与此同时，空间分解在解决高维优化问题时已展现出良好的性能 [43–45, 48]。基于上述事实，我们按照相应的组件对联合空间进行分解，并采用现有调优智能体的求解器来调优相应组件。因此，我们可以充分利用针对特定组件定制的基于机器学习的调优智能体的最新研究成果。为此，我们试图回答这样一个问题——“如何协调现有的基于机器学习的调优智能体，为数据库管理系统中的多个组件进行配置？”

这一问题背后有三个主要挑战。首先，针对不同数据库管理系统组件，出现了具备高级功能的新型基于机器学习的调优代理，从系统角度来看，如果缺乏对智能体的高层抽象和通用解决方案，我们就无法将它们统一起来进行多组件调优。我们能否设计一种统一的解决方案，以便支持现有的基于机器学习的调优代理，并方便与未来的扩展集成（*C1*）？其次，鉴于不同组件之间存在依赖关系，调优智能体应能够根据其他智能体的调优决策调整自身的调优策略。然而，现有研究以独立方式调优数据库管理系统——调优智能体的机器学习模型假定其他组件的配置保持不变。如果其他智能体修改了其所控制的配置，智能体便无法准确预测 [51]，，因而无法找到全局有希望的配置。为了利用现有智能体调优多个组件，我们需要一种机制，让智能体彼此通信并协调（*C2*）。第三，运行基于机器学习的智能体成本高昂，因为它需要投入时间和资源来观察配置所产生的影响。然而，每个智能体对应的奖励分布（即性能提升）都是未知的。而且，在给定的调优预算（例如时间）下，智能体获得的奖励并不均衡，且呈非平稳变化（大多数情况下会递减，即边际效用递减），如图1c所示。鉴于这些事实，我们需要探索一种审慎的策略，以便将预算高效地分配给有希望的调优智能体（*C3*）。

为应对上述挑战，我们提出了一个统一、高效的协调框架 UniTune，用于编排基于机器学习的调优智能体。我们首先分析了现有的数据库管理系统调优智能体，并将其分为三类：基于贝叶斯优化（BO）的方法、基于强化学习（RL）的方法，以及基于强化学习估计器的方法。我们抽象出不同调优智能体的逻辑，并定义了接口，以支持各种现有智能体并便于未来扩展（应对 *C1*）。其次，为突破独立调优的局限，我们提出了一种消息传播协议，使智能体能够协同工作。该协议规定了一个智能体如何将其调优决策作为消息广播，以及其他智能体如何接收和处理该消息。为了根据调优消息更新调优策略，我们将消息封装为各智能体模型中的上下文特征（应对 *C2*）。第三，我们将预算分配问题建模为多臂老虎机问题，其中每个智能体对应一条臂，并提出一种基于汤普森采样的策略来选择有望取得良好效果的智能体。该策略通过审慎权衡利用与探索，根据历史观测最大化调优性能。为避免在非平稳环境中因奖励变化而导致选择失当，我们提出了一种

策略，该策略利用记忆缓冲区丢弃过时的观测（应对*C3*）。

据我们所知，UniTune 是首个利用机器学习技术调优数据库管理系统中多个组件的统一框架。总而言之，我们的贡献如下。

- 为了高效调优数据库管理系统中的多个组件，我们探讨了如何利用现有的基于机器学习的调优代理，并设计了一个具有高层抽象和灵活接口的统一协调框架。
- 为了实现智能体之间的协作，我们提出了一种消息传播协议，规定其消息共享行为，并将调优消息封装在智能体模型中。
- 为了在智能体之间分配有限的调优预算，我们提出了一种基于汤普森采样的策略，根据历史观测选择有前景的智能体。该策略通过记忆缓冲区设计应对奖励的非平稳性。
- 我们在合成工作负载和真实工作负载上进行了大量评估。结果表明，与基线方法相比，UniTune 能够支持不同的基于机器学习的调优智能体，并推荐更优的配置，使工作负载执行时间加快 1.4~14.1× 。

本文其余部分安排如下。我们在第2节讨论相关工作。在第3节中，我们对多组件调优问题进行形式化定义，并回顾现有的基于机器学习的调优智能体。接着，我们在第4节为基于机器学习的调优智能体提供一种高层抽象，并概述其整体情况。在第5节中，我们介绍一种消息传播协议，以实现调优智能体之间的协作。在第6节中，我们提出一种策略，在各智能体之间分配有限的调优预算。最后，我们在第7节报告评估结果，并以总结结束本文。

2 相关工作

近年来，自动配置数据库管理系统在工业界和学术界都引起了广泛关注。其目标是实现数据库调优任务的自动化，并以数据驱动的方式搜索更优配置 [58]。相关方法采用强化学习（RL）和贝叶斯优化（BO）算法，通过试错的方式对数据库管理系统进行调优。以往研究通常聚焦于特定的调优任务，例如索引选择、参数调优、查询重写、视图生成和数据分区。

索引选择。 在适当的列上建立索引对于加快查询执行至关重要 [49]。机器学习技术用于从大量可能的索引组合中选出合适的索引。例如，SmartIX [47], MANTIS [56] 和 AutoIndex [73] 采用强化学习，而 DBA-bandits [52] 则在贝叶斯优化框架中采用 C^2UCB 算法。**参数调优**。 数据库管理系统有数百个可配置参数，这些参数会影响其性能 [69]。为取代依赖数据库管理员经验的手动调优，数据库社区已开发出大量基于机器学习的方法来自动化这一过程，例如基于贝叶斯优化的方法 [6, 11, 16, 20, 33, 37, 70, 71] 和基于强化学习的方法 [10, 42, 68] 。**查询重写** 旨在将查询转换为等价但性能更高的查询。这是一个 NP 难问题 [21]，存在大量重写顺序（例如，不同的算子和规则）。Learne- dRewrite [72] 使用回归模型来估算重写节点的收益。此外，一种轻量级的基于强化学习的智能体——蒙特卡洛树搜索（蒙特卡洛树搜索，MCTS）被用于与回归模型交互，以寻找重写效果更好的查询。

视图生成。 基于以空间换时间原则，物化视图可以减少共享等效子查询的查询之间的冗余计算。两种方法 [27, 67]使用回归模型估计不同视图候选项及查询的收益，并通过采用深度 Q 学习算法的基于强化学习的智能体，为给定工作负载建议视图—查询对。

数据分区。 对数据库进行分区可以显著提升分析型工作负载的性能，因为数据密集型查询可以分配到多台机器上。许多方法 [17–19, 29] 利用强化学习智能体探索不同的分区键。Li 等人 [29] 则实现了一个回归模型来估计分区收益。

我们专注于设计一个统一的协调框架，协调基于机器学习的调优代理来调优多个数据库管理系统组件。与我们最接近的是近期的一项研究 UDO [59]。该研究提出使用强化学习智能体优化更多数据库管理系统组件。它将配置分为重型和轻型两类——重型参数的重新配置开销较高（例如索引），而轻型参数的开销可以忽略（例如调优参数）。它采用双层循环来降低重新配置开销。在外层，强化学习智能体建议并应用一个重型参数。随后，在内层初始化另一个强化学习智能体，并经过若干次迭代，为轻型参数找到合适的设置。然而，我们应该根据调优智能体的预期效用分配调优预算，而不是依据重新配置开销。双层调度会导致固定的预算分配模式：用于外层代理的调优预算较少。正如我们在第 7.2 节的评估中所示，这种固定调度会导致整体性能不佳。与我们的通用框架不同，UDO 并非旨在协调现有的调优智能体，而是专注于用于调优数据库管理系统的定制算法。例如，它提出了蒙特卡洛树搜索变体 delayed-HOO 来建议重型参数，并提出了一种规划器，通过谨慎安排索引配置的评估顺序来减少重新配置开销。这些定制算法与我们的工作正交，可视为另一种调优智能体并集成到我们的框架中。

3 预备知识

本节将对问题进行形式化表述，并回顾现有的基于机器学习的数据库管理系统调优代理。

3.1 术语和问题陈述

我们首先介绍相关术语，然后定义多组件调优问题。

调优智能体。 智能体是一个外部系统，专注于调优数据库管理系统中的特定组件，以改善预先定义的指标。

示例 *1.* 现代数据库管理系统有多个可配置组件，会影响其运行时执行和性能，包括：（1）物理设计（例如索引、视图和数据分区）；（2）配置旋钮；（3）查询设计（例如在外部重写查询）。智能体通常负责配置其中一个组件。例如，为了尽量缩短给定工作负载的执行时间，旋钮调优智能体会搜索合适的

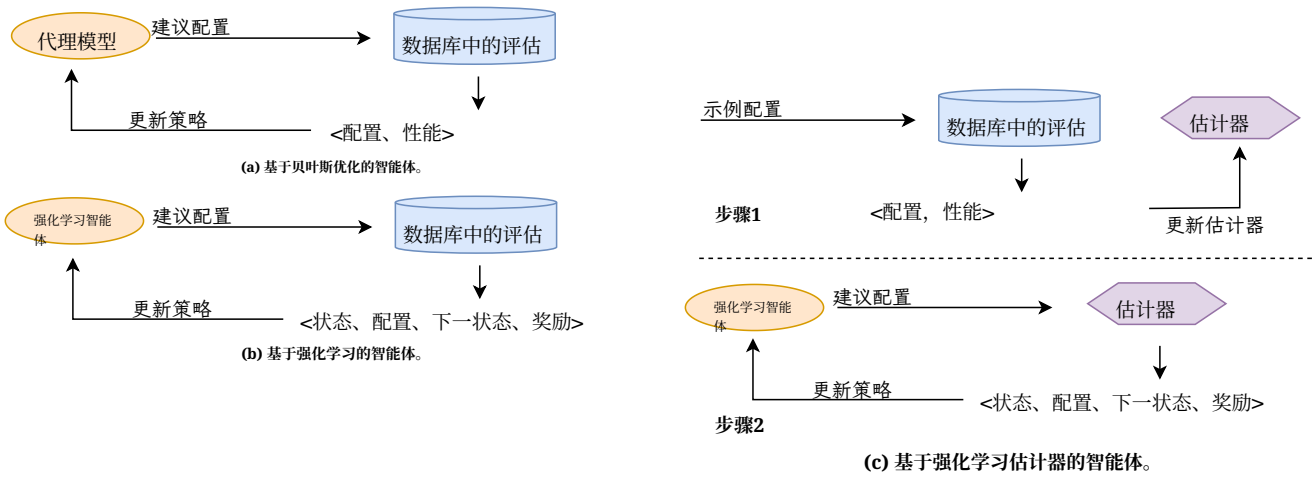


图 2：基于机器学习的调优智能体的三种类别。

调优参数的取值，而视图生成智能体会为工作负载中的查询选择合适的物化视图。我们将组件 i 对应的智能体记为 A_i。

配置子空间。 配置子空间 Θ_i 是组件 i 的所有可能配置组成的集合。我们将 Θ_i 中的配置值记为 θ_i。配置子空间的取值范围取决于相应组件的性质以及智能体采用的形式化方法。

示例 *2*。 对于参数调优，其取值范围可以由连续变量（例如 *innodb_buffer_pool_size*）和类别变量（例如 *innodb_stats_method*）混合构成。对于索引选择，其取值范围可以是一组二元值，用于表示是否创建相应的索引。取值范围甚至可以是一组元组。例如，查询重写代理会做出原子调优决策 (o, r)，表示将重写规则 r 应用于算子 o。

配置空间。 配置空间 Θ 由 m 可配置组件的配置子空间组成，即 $\Theta = \Theta_1 \times \Theta_2 \times ... \times \Theta_m$。

调优预算。 调优预算是允许用于调优数据库管理系统的总成本。成本可以从不同角度衡量，例如时间、资源或金钱，这些因素之间呈正相关。为简单起见，本文以时间衡量成本。我们使用子预算作为超参数，控制执行智能体所花费的预算单位。

问题陈述。 考虑 m 个数据库管理系统组件，多组件调优的目标是在有限的调优预算下运行基于机器学习的调优代理，以优化用户预先定义的指标：

$$\arg\max_{\theta_1,...,\theta_m} f(\theta_1, ..., \theta_m; W),$$

其中，$\theta_1, ..., \theta_m \in \Theta_1 \times \Theta_2 \times ... \times \Theta_m$ 是相应智能体建议的配置值。W 对应给定的工作负载。f 是数据库性能指标，可以是任何预定义的目标，例如吞吐量和第 99 百分位延迟。请注意，$f(\cdot)$ 的结果可以在数据库管理系统中进行评估（例如压力测试）后观测到。

3.2 现有的基于机器学习的智能体

我们将现有的基于机器学习的智能体分为三类：基于贝叶斯优化（BO）的、基于强化学习（RL）的，以及基于强化学习估计器的。

基于贝叶斯优化的智能体 将调优任务建模为黑盒优化问题，并通过与数据库系统交互来更新其调优策略，如图 2a 所示。为求解黑盒函数，智能体会迭代执行以下步骤：(1) 通过计算采集函数值（该值衡量候选点的效用）来建议下一个待评估的配置；(2) 与数据库管理系统交互，评估所建议的配置；(3) 更新概率代理模型，该模型描述配置与其性能之间的关系。

基于强化学习的智能体 将调优任务建模为马尔可夫决策过程（MDP），并通过与数据库系统交互来更新其调优策略，如图 2b 所示。MDP 将问题建模为一个多步过程。在每一步中，智能体观察当前状态 s 并根据策略选择一个动作 $a \in A(s)$。该动作会带来奖励 $r(s, a)$，并使状态变为 $s\prime$。基于强化学习的智能体旨在找到一种能够最大化累积观测奖励的决策策略。

基于强化学习估计器的智能体 同样将调优任务建模为 MDP，但强化学习智能体通过与预训练估计器交互来更新其调优策略。估计器可以估算给定动作的效用。现有工作采用两步法训练估计器和强化学习智能体，如图 2c 所示。步骤1中，估计器以离线方式训练。训练数据通过随机采样配置，并与数据库管理系统（DBMS）交互以观察其性能而获得。步骤2中，强化学习智能体通过与估计器交互来更新策略，而不是在 DBMS 中进行评估。

范围说明。 我们聚焦于通过统一框架，利用不同的基于机器学习的调优代理来配置数据库管理系统中的多个组件。为清晰、通用地阐述该框架，我们说明一些必要的约束。(1) 我们将框架的适用范围限定为支持外部智能体：它们通过数据库管理系统的应用程序编程接口配置数据库

管理系统公开的组件，而无需修改其内部实现。外部方案易于应用于现有数据库管理系统，无需投入软件工程工作来改造其架构。(2) 在算法层面，我们聚焦于离线调优。在线调优可通过克隆和并行化方案实现：在多个克隆实例上对工作负载进行压力测试，并将安全的方案应用于在线数据库 [10]。(3) 目前，我们让用户选择一种算法来调优一个组件（我们为每个组件提供默认选项）。未来，我们可以支持为每个组件自动选择算法，例如，在 OtterTune [6], 、CDBTune [68] 和 CGPTuner [11] 等候选方案中，决定为调优参数组件选用哪种算法。

4 概述

本节重点介绍 UniTune 的核心抽象，并讨论其用户界面和工作流程。

4.1 核心抽象

我们的框架为基于机器学习的调优代理提供了高层抽象。这些智能体共享一种迭代式工作流程，通过试错来调优数据库管理系统。它们利用机器学习模型预测有前景的配置（例如，构建索引），并评估建议配置的性能。根据评估结果，它们更新模型，以提升后续决策的有效性。遵循上述范式，我们的框架为基于机器学习的调优代理定义了一种抽象。具体而言，我们根据调优智能体采用的算法以及评估建议配置的方式对其进行分类，详见第 3.2 节。此外，为支持现有的基于机器学习的调优代理，我们定义了以下三个基类（贝叶斯优化、强化学习和强化学习估计器），并提供相应的接口，如表 1 所示。接口总结如下。

- 初始化模型：初始化调优智能体使用的模型。
- 建议：推断已定义的模型，以预测有潜力的配置。
- 更新策略：使用输入的增强观测更新已定义的模型。
- 更新估计器：使用输入的增强观测更新估计器。

这种抽象为不同的调优算法提供了灵活支持，因为它消除了现有基于机器学习的调优智能体中的异构性问题。对于每个主类，调优智能体在如何实现其逻辑对应的这些接口方面有所不同（例如，在初始化模型中采用哪种神经网络，以及如何在建议中基于其模型预测有望取得良好效果的配置）。借助这些接口，可以轻松将智能体与 UniTune 中现有的多组件调优实现集成。用户只需继承相应的基类并重写其函数即可。例如，要添加一个强化学习调优智能体——CDBTune [68], ，用户需要在初始化模型中为该智能体定义一个演员-评论家网络，并在建议中实现如何通过推断演员网络来提出配置，以及如何在策略更新中基于输入观测更新模型。

表 1：基于机器学习的调优智能体抽象。

Type	算法	接口
BO	贝叶斯优化	建议，模型初始化，更新策略
RL	强化学习	建议，模型初始化，更新策略
强化学习估计器	强化学习学习	建议、模型初始化，更新策略、更新估计器

```
[调优设置]
组件 = {'index': 'DBA-Bandit', 'knob':'OtterTune',
'query':'LearnedRewrite'} 调优_预算 = 108000性能_指标 ='
执行时间'
```

图3：用户界面示例。

有了用户实现的接口，我们的框架便能自动协调不同组件的智能体，为用户隐藏底层细节。首先，对于单个智能体，我们的框架会使用这些函数自动执行其流水线。对于贝叶斯优化和强化学习，它会通过与数据库管理系统交互，评估“建议”输出的配置，并将由此得到的观测输入更新策略。对于强化学习估计器，它会采样一个配置，在数据库管理系统中对其进行评估，并将由此得到的观测输入更新估计器。其次，我们的框架通过维护每个智能体的运行记录，合理调度各智能体，以调优数据库管理系统中的多个组件。它会自动决定如何在各智能体之间分配有限的调优预算（详见第6节），并按照消息传播协议管理智能体，从而实现智能体之间的协作（详见第5节）。

4.2 用户界面

要启动多组件调优任务，用户只需指定数据库设置和调优设置。数据库设置是指连接信息和工作负载信息。调优设置描述了要调优的数据库管理系统组件类型、相应的智能体、性能指标以及总体调优预算。为方便使用，我们采用配置文件来定义任务。图 3 中的代码给出了一个示例；受篇幅限制，我们省略了数据库设置。该示例定义了一个任务，分别通过基于机器学习的调优代理 DBA Bandit [52], OtterTune [6] 和 LearnedRewrite [72], 配置索引、调优参数和查询组件。调优预算设为 30 小时，调优目标为执行时间。

4.3 工作流程

图4展示了 UniTune 的概览。为说明其内部工作流程，图中着重标出了关键组件。根据用户定义的调优智能体和任务信息，两个模块（预算分配和消息传播）会迭代执行。首先，预算分配模块会选出一个有前景的调优智能体，并将选定的智能体传递给消息传播模块。随后，消息传播模块会安排输入的智能体

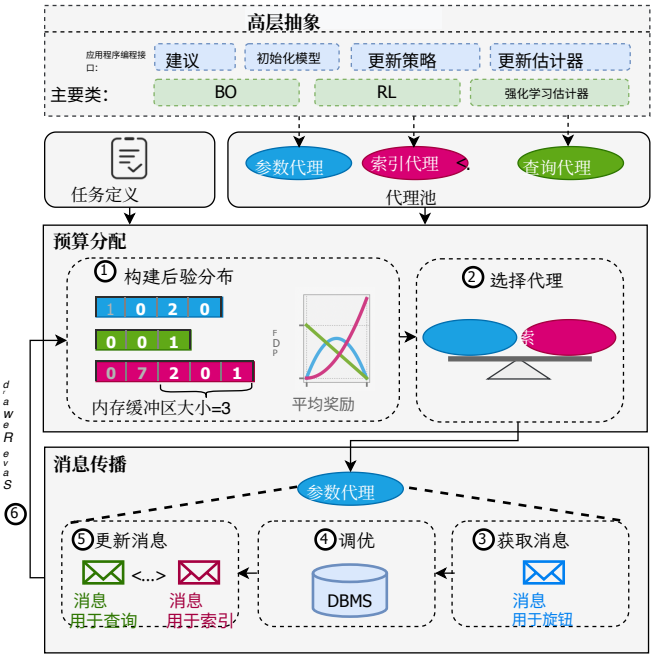


图4：UniTune 的工作流程。UniTune 具有“高层抽象”，支持不同的基于机器学习的调优智能体；具有“预算分配”模块，用于选择有前景的调优智能体；还具有“消息传播”协议，使智能体能够协作解决彼此间的依赖关系。

调优数据库管理系统，同时与其他智能体协作；最后，将获得的调优奖励返回给预算分配模块，以更新资源分配策略。当调优预算耗尽或满足终止准则时，工作流程结束。下面将分别介绍这两个模块。

预算分配 每一步都会选择一个有前景的智能体来调优数据库管理系统。预算分配问题可以建模为具有非平稳奖励的多臂老虎机问题（见第6.1节）。其目标是在给定预算内最大化累积期望奖励（即性能指标的提升）。挑战在于，只有在使用一定预算评估智能体的动作后，才能观测到该智能体的奖励；而且奖励是非平稳的——会随着优化进程而变化。为了做出良好决策，该模块通过汤普森采样这一经典的在线决策问题算法，审慎地平衡探索与利用（见第6.2节）。具体而言，它根据奖励观测（步骤1）为每个智能体构建奖励的后验分布，并根据从后验分布中采样得到的概率选择一个智能体（步骤2）。为适应非平稳奖励，我们使用记忆缓冲区限制构建后验分布时所考虑的奖励观测数量。

消息传播使调优智能体能够协作。选定的智能体按照消息传播协议执行流程，该协议规定了它应如何处理

算法1 顶层设计。

参数： 智能体 $\{A_i\}_{i=1}^m$，调优预算 K，子预算 k。
输出： 建议采用性能最佳的配置。

1: **函数** 框架($\{A_i\}_{i=1}^m, K$)
2: // 将全局性能初始化为默认值
3: $perf_{glocal} \leftarrow$ 初始化()。
4: **当**调优预算仍有剩余**时**
5: $A_j \leftarrow$ 选择智能体($\{A_i\}_{i=1}^m$)。
6: $context \leftarrow$ 获取消息(A_j)。
7: // 使用子预算运行选定的智能体
8: $config_{inc}, f_{inc} \leftarrow A_j.Run(context, k)$.
9: // 保存性能提升
10: $A_j.history.append(max(0, \quad f_{inc} - f_{global}))$。
11: **如果** $f_{inc} \leq$fglobal **then**
12: $f_{global} \leftarrow f_{inc}$.
13: $A_j.apply(config_{inc})$.
14: // 向其他智能体发送更新消息
15: 更新消息($\{A_i\}_{i=1}^m - A_j$).
16: **结束 if**
17: **结束 while**
18: **结束函数**

收到的消息并向其他智能体广播其调优消息（详见第5节）。智能体首先获取自身的消息，其中包含其他智能体做出的调优决策（步骤3）。然后，智能体将该消息封装到其模型中（详见第5.2.2节），并使用子预算调优其负责的组件（步骤4）。如果发现了更好的配置，就将该配置应用于数据库管理系统，并以上下文特征的形式表达其调优决策（详见第5.2.1节），再将其传播给其他智能体（步骤5）。最后，我们将观测到的奖励追加到相应记录中，以便后续进行预算分配（步骤6）。

算法1给出了伪代码，展示了我们框架的主要优化循环。子预算是一个超参数，用于控制运行一个智能体所分配的预算量。它应大于数据库管理系统评估（例如压力测试）所需的最长时间，同时又应足够小，以便进行细粒度的预算分配。在我们的实现中，我们将其设为压力测试超时时间的两倍。UniTune 首先将当前最佳性能初始化为默认性能（第 3 行）。然后，它选择一个智能体（第 5 行）。选定的智能体会接收其他智能体发送的消息（即上下文特征）（第 6 行），使用子预算配置数据库管理系统，并返回调优过程中找到的最佳配置及其性能（第 8 行）。UniTune 会保存奖励（即性能提升）（第 10 行）。如果选定的智能体找到了更好的配置，它就会应用该配置，并更新发送给其他智能体的消息（第 11–16 行）。

5 消息传播协议

为实现调优智能体之间的协作，我们提出了一种消息传播协议来规定其行为。在该协议中，调优智能体交替执行，并将各自的调优决策（即消息）广播给其他智能体。接下来，我们将在第5.1节讨论设计方案，并介绍如何表示消息，以及如何让智能体从这些消息中学习。

在第5.1节中。为了支持基于强化学习估计器的智能体采用该协议，我们在第5.3节提出了一种新的训练范式。

5.1 设计选项

正如我们所分析的，基于机器学习的调优智能体通过在数据库管理系统中进行评估来获取配置的性能。因此，为确保观测结果有效，在评估期间不能更改配置（包括其他组件的配置），所以无法同时运行两个智能体。为此，在数据库系统中调优多个组件有两种选择：(1) 联合优化，将数据库管理系统各组件的子空间合并，并运行单个智能体在联合空间中进行优化；(2) 交替优化，交替运行多个调优智能体，在相应的子空间中进行优化。

我们的方案采用交替优化。联合优化存在严重的可扩展性问题，而且依赖单个调优智能体，而该智能体难以适配不同的数据库管理系统组件。首先，随着搜索空间维度的增长，达到全局最优所需的评估次数呈指数级增长。某些后验不确定性较大的区域可能导致过度探索，无法利用有前景的区域。在某些高维空间中，标准贝叶斯优化的表现可能不如随机搜索 [61]。此外，强化学习面临稀疏奖励的问题，可能无法学到成功的策略 [62]。其次，底层目标函数很复杂，因此难以在庞大的空间中拟合一个全局模型。例如，用于索引选择的强化学习智能体 SmartI [47], 对离散子空间采用深度 Q 学习；用于参数调优的强化学习智能体 CDBTune [68], 对连续子空间采用深度确定性策略梯度。由单个智能体在两个子空间中进行联合优化并非易事。相比之下，交替优化通过将完整空间分解为多个子空间，高效地解决优化问题；也就是说，它只通过相应的调优智能体配置选定的组件。因此，它能够充分利用针对特定组件调优的新兴技术。例如，我们可以交替运行两个智能体。首先，我们可以运行 SmartI 等智能体，消耗一部分预算，并将迄今为止找到的最佳索引固定到数据库管理系统中。其次，我们切换到 CDBTune 等智能体，用另一部分预算找出该智能体能够找到的最佳旋钮。在 UniTune 中，预算分配模块会逐步决定执行哪个智能体，详见第6节。在每个步骤中，交替优化实质上是固定联合空间的某些维度，并在一个远小于完整空间的子空间中进行优化，因此可以高效优化。该过程通过交替探索不同子空间来搜索最优解。

5.2 通过上下文广播消息

现有调优智能体假设其他组件的配置固定不变。在 UniTune 中，如果某个智能体发现了更好的配置，就会将该配置应用于其控制的组件，并在其他智能体调优数据库管理系统时固定该配置。但是，如果一个智能体修改了自己的组件，由于环境变化，其他智能体的建模就会变得不准确。因此，我们需要向其他智能体广播其调优决策，这样其他智能体就能及时获知环境变化。接下来的问题是，如何让受到影响的智能体适当地应对环境变化。一种可能的应对方式是初始化其机器学习模型，并在新环境中从头开始搜索。然而，这样会遗忘先前的调优历史记录，而从头构建新模型又需要大量观测。因此，我们转而采取另一种思路——让智能体适应环境变化。直观而言，不同环境中的调优策略共享某些知识。我们将环境特征称为上下文。上下文和配置到性能指标之间存在映射 [71]。我们期望同一配置在相关上下文中具有相近的性能指标。利用上下文之间的相关性，可以显著加快调优过程。为此，我们决定将智能体之间广播的消息表示为上下文特征，并探讨两个子问题：(1) 如何刻画上下文；(2) 如何让智能体跨上下文学习。

5.2.1 上下文表征。 最直接的方法是将其他组件的所有配置串联起来作为上下文特征，因为这些配置能够直接反映其他智能体的调优行为所引起的环境变化。然而，配置可能是非结构化的。例如，负责查询重写的智能体会对工作负载中的查询应用若干（算子、规则）转换，而转换的数量会不断变化。要支持不同智能体的扩展，还需要额外的特征工程。受 OtterTune [6], 启发，我们转而采用数据库管理系统的内部运行时指标来表征配置的影响。所有现代数据库管理系统都会公开大量系统信息，例如数据页读写数量、查询缓存利用率和锁开销等统计数据。这些信息会受到配置设置的影响。为了观测某个智能体所处的上下文，我们固定其他组件迄今为止的最佳配置，将其负责组件的配置保持为默认值，然后对数据库管理系统进行压力测试，以收集内部指标的统计数据。

5.2.2 上下文封装。 为了利用调优消息，我们需要将上下文特征封装到调优智能体的模型中。对于基于贝叶斯优化的智能体，我们将上下文特征添加到代理模型的输入中。随后，代理模型会对上下文和配置到性能的映射进行建模。对于基于强化学习的智能体，我们将上下文特征与观测到的状态拼接起来。这样，神经网络学习到的调优策略（即从状态到动作的映射）在相关上下文之间会相似。对于基于强化学习估计器的智能体，我们将上下文特征添加到估计器的输入中。这样，估计器就能估计不同上下文中给定动作的效用。总而言之，强化学习智能体的训练数据格式为 状态, 配置, 下一状态, 奖励，其中此处的“状态”和“下一状态”均包含拼接后的上下文特征。贝叶斯优化智能体和强化学习估计器智能体的训练数据格式则为 上下文特征, 配置, 性能。

5.3 基于强化学习估计器的智能体的不确定性感知训练

在交替优化中，如果某个调优智能体找到更优的配置，我们就将其应用于数据库管理系统，并向其他智能体广播其调优决策

。随后，其他智能体可以根据收到的消息更新建议，从而逐步形成更优的配置组合。对于基于贝叶斯优化和基于强化学习的智能体而言，这一过程很直接，因为它们会分配子预算来更新调优策略，并在被调用时广播调优消息。然而，对于基于强化学习估计器的智能体，按照现有的两步方式（如图 2c 所示），它们首先使用分配的子预算，通过在数据库管理系统中进行高成本评估来训练估计器（步骤1）。估计器训练完成后，强化学习智能体会更新其调优策略，并且只推荐最终配置（步骤2）。因此，在步骤1期间（占据大部分时间），其他智能体无法根据基于强化学习估计器的智能体所做的调优决策来优化自己的建议。此外，步骤1中随机采样的配置分布可能与强化学习智能体在步骤2中推断出的配置不一致，从而导致估计不准确。

为了解决这些问题，我们为基于强化学习估计器的智能体设计了一种不确定性感知训练方案。我们在一个步骤中训练强化学习智能体和估计器，使强化学习智能体每次被调用时都能建议有前景的配置。具体而言，我们采用不确定性感知代理模型来估计配置的不确定性，例如高斯引用。调用基于强化学习估计器的智能体时，我们首先训练强化学习智能体，并推断估计器。随后，我们将推断出的配置放入优先队列，并以估计的不确定性作为优先级。强化学习智能体收敛后，我们通过与数据库管理系统交互来评估其推荐的配置。如果发现更优的配置，就应用该配置并将其广播给其他智能体，以促进智能体之间的协作。然后，我们利用剩余子预算评估优先队列中各配置的性能，并根据扩充后的观测更新估计器。不确定性感知方案以需求为导向来指导配置采样，从而提高估计器预测的准确性。

6 预算分配

可以采用多种策略在智能体之间分配调优预算。一种简单的策略是轮询，依次平均分配调优预算。然而，不同智能体所获得的奖励通常差异很大。例如，有些工作负载对索引配置非常敏感，而查询重写几乎没有改进效果，甚至毫无改进。因此，我们应该将更多预算用于学习优质索引，而不是查询重写。轮询会将预算浪费在最终奖励很少的智能体上。为了合理分配预算，我们发现，可以将智能体选择定义为统计机器学习中的一个多臂老虎机（MAB）问题 [9]，其中不同的臂对应不同的调优智能体。接下来，我们将讨论该问题的形式化定义，并介绍一种基于汤普森采样、使用记忆缓冲区来解决多臂老虎机问题的策略。

6.1 智能体选择的多臂老虎机视角

多臂老虎机问题数十年来一直是统计学领域深入研究的课题。这个名称源于这样一种设想：一名赌徒面对一台有多个臂的老虎机，必须决定拉动哪个臂 [9]。每次拉动一个臂，赌徒都会获得一笔收益。由于每个臂对应的收益分布并未给出，赌徒只能通过试验来了解它。智能体选择问题与多臂老虎机问题有相似的本质，因为二者都需要依次在线做出决策，以最大化来自未知分布的总收益。

用于智能体选择的多臂老虎机。 我们正式定义智能体选择场景中的随机多臂老虎机问题。给定一组调优智能体 $\mathbf{A} = \{A_i\}_{i=1}^{m}$，每个智能体都是一条“臂”。令 $\mathbf{T} = \{1, 2, ..., T\}$ 表示一系列决策轮次（T可根据调优预算和子预算的设置推导得出）。在每个决策轮次，我们必须选择一个智能体来执行。于轮次 $t \in \mathbf{T}$选择智能体 $A_j \in \mathbf{A}$后，将观测到一个实数值奖励 $X_t^j \in \mathbb{R}$，其中 X_t^j是一个期望值为 $\mu_t^j = \mathbb{E}[X_t^j]$的随机变量。目标是最大化轮次$\mathbf{T}$内的期望累积奖励：

$$\underset{j(1),j(2),...,j(T)}{\arg\max} \mathbb{E}\left[\sum_{t=1}^{T} \mu_t^{j(t)}\right],$$

其中，$j(t)$ 是在步骤 t 中选取的臂 .

奖励函数。 由于我们的目标是找到能够优化给定性能指标的配置组合，因此奖励的语义必须与调优目标一致。因此，我们将运行智能体时性能指标的提升定义为奖励。具体而言，奖励为

$$X_t^j = max\left(0, f_{t,inc}^j - f_{t,global}\right),$$

其中，$f_t^j{}_{,inc}$ 表示智能体 A_j 在轮次 t达到的最佳性能，而 $f_{t,global}$ 表示在 t 之前达到的最佳性能。假设我们要最大化性能指标 f。

技术挑战： 为了最大化累积奖励，我们面临两项挑战。第一项是多臂老虎机问题中常见的两难抉择——利用与探索之间的权衡：

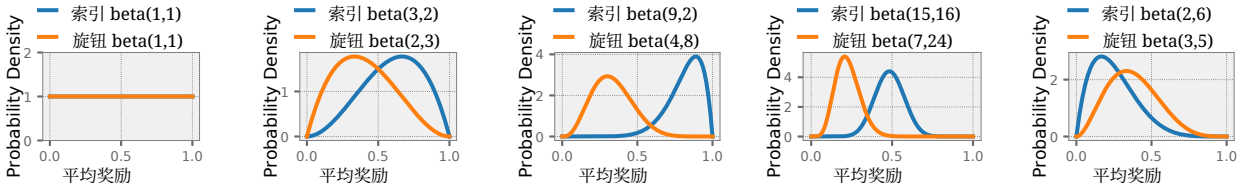
- 利用：可以将预算分配给过去已取得较高奖励的智能体。
- 探索：也可以将预算分配给未来可能获得更高奖励的智能体。

第二个是在智能体选择场景中，由非平稳奖励引发的“记住”与“遗忘”之间的权衡。在传统的多臂老虎机中，奖励分布不会随时间变化。然而，正如我们所分析的，智能体的奖励分布并非平稳，而是会随着优化的进行而变化。通常，智能体的奖励会逐渐降低，因为随着预算消耗，智能体带来的性能提升趋于饱和（即边际收益递减）。因此，旧经验可能不再适用于估计未来奖励。为了选择有潜力的智能体，我们应谨慎权衡记住与遗忘：

- 记住：可以跟踪更多观测，以降低奖励估计的方差。
- 遗忘：也可以舍弃“旧”信息，因为底层奖励可能发生变化，使这些信息不再那么相关。

6.2 带记忆缓冲区的汤普森采样

为了解决上述挑战，我们提出了一种基于汤普森采样且带有记忆缓冲区的策略。汤普森采样 [53]是一种用于多臂老虎机问题的经典强化学习算法。概括而言，它会根据某个臂成为



（a）第0轮。 (b) 第4轮。 (c) 第10轮。 (d) 第30轮。 (e) 带缓冲区的第30轮。

图 5：不同轮次的后验分布。我们展示了平均奖励的概率密度函数。

算法2智能体选择。

参数： 智能体 $\{A_i\}_{i=1}^m$，缓冲区大小 s，缩放因子 $rfactor$。
输出： 一个选定的智能体。

函数 智能体选择($\{A_i\}_{i=1}^m$)
  对于 $i \leftarrow 1, 2, \cdots, m$ **do**
    $S \leftarrow 0, F \leftarrow 0$
    对于 $reward \in A_i.history[-s:]$ **执行**
      如果 $reward > 0$ **则**
        $S \leftarrow S + round\left(\frac{reward}{rfactor}\right)$。
      else
        $F \leftarrow F + 1$。
      结束如果
    结束循环
    从 $Beta(S+1, F+1)$ 分布中采样 w_i。
  结束循环
  返回 $\arg\max_{i=1}^m w_i$。
结束函数

最佳臂的概率来选择要尝试的臂。它会积累经验（即对选定智能体及其奖励的观测），以构建奖励的后验分布。每轮中，算法都会根据抽样得到的概率，选择最有可能获得最高奖励的臂。

利用与探索。汤普森采样通过从后验分布中抽取臂，而非贪心地选择期望奖励最高的臂，来应对第一个挑战。考虑运行两个智能体的例子——索引代理和参数代理。图 5 展示了它们的后验分布如何随轮次变化。这些分布表示根据观测到的历史记录对各臂形成的信念。在第0轮（图 5a），我们还没有任何奖励观测。因此，选择这两个智能体的概率相同。在第4轮（图 5b），索引代理获得了更多奖励。贪心策略会选择索引代理，因为它的期望奖励更高。但在汤普森采样中，仍有可能选中参数代理，因为所选的臂是从后验分布中抽取的。在第10轮（图 5c），索引代理更有可能被选中，因为后验分布的差异更加明显，不确定性也更小（但汤普森采样仍有可能选中参数代理）。直观而言，若要最大限度地探索，就可以完全随机地选择智能体；若要最大限度地利用，就可以贪心地选择期望奖励最高的智能体。从后验分布中抽样，能够在这两个目标之间取得平衡。

记忆与遗忘。然而，考虑到第二个挑战——在我们的场景中，奖励分布是非平稳的，传统的汤普森采样效果会较差。继续以图 5 中的例子为例，假设索引代理在第10轮后奖励已趋于饱和（即索引代理连续无法找到更好的配置）。但由于索引代理在前10轮获得了大量奖励，因此在第30轮（图 5d），其期望奖励仍高于不确定性相对较小的参数代理。因此，尽管索引代理无法带来未来奖励，它仍更有可能被选中。为应对第二个挑战，我们采用一种简单而有效的策略——使用记忆缓冲区控制用于构建后验分布的观测数量。给定智能体 A_j 获得的奖励观测 $\{X_t^j\}_{t=0}^{T_j}$，我们仅使用 $\{X_t^j\}_{t=T_j-s+1}^{T_j}$（即之前的 s 次观测）来构建其后验分布，其中 s 是记忆缓冲区大小。缓冲区大小表示纳入考虑的观测数量。缓冲区较小时，算法受过去观测偏差影响的风险较低，也能更好地适应未来奖励的变化。但与此同时，算法可能会遗忘有用的知识，从而错失利用的机会。缓冲区大小控制着遗忘与记忆之间的权衡。理论上已经证明，在非平稳环境中，只要遗忘与记忆之间的权衡适当，就能实现次线性遗憾 [25]。图 5e 展示了缓冲区大小为 7 时的后验分布。参数代理的期望奖励高于索引代理，因此更有可能被选中，因为在纳入考虑的前七轮中，它获得了更多奖励。

如前所述，不同的调优智能体之间存在依赖关系。分配策略通过不同的组件调优顺序，探索不同的依赖方向。它会在每次迭代中选择一个有潜力的智能体，从而逐步自主决定调优顺序。例如，UniTune 可能会先评估一种调优顺序；如果改进幅度较小，就会探索其他方向。因此，系统会根据历史反馈自适应地探索不同的调优顺序。算法2对 UniTune 中的预算分配策略进行了形式化描述。我们采用贝塔分布来描述奖励的后验分布，这种方法在汤普森采样中得到了广泛应用 [5]。贝塔分布有两个参数，S和 F，用于控制对期望奖励的估计。如图 5 所示，选择一个臂后，只需调整这两个参数，即可构造期望奖励的后验分布。并且，S和 F越大

则 $Beta(S,\ F)$的分布越集中于均值附近。为了在记住和遗忘之间取得平衡，我们利用大小为 s 的记忆缓冲区中观测到的奖励来更新这两个参数（第 2–8 行）。随后，我们从后验分布中采样（第 11 行），并根据某个智能体的均值为最大值的概率来选择智能体（第 13 行）。传统的汤普森采样使用二元奖励，因此并不直接适用于连续奖励的情况。为使汤普森采样适用于连续奖励场景，我们采用"概率奖励" [5]：将连续奖励缩放至 $[0, 1]$ 的取值范围，并以缩放后的奖励作为成功概率，从伯努利分布中采样一个二元奖励。然而，"概率奖励"可能导致观测到正奖励的智能体采样得到零奖励。在我们的情形中，信息损失会进一步削弱本就稀疏的奖励。根据我们的实证经验，我们按一个常数因子缩放观测到的奖励，并使用四舍五入后的值来更新 S（第 6 行）。

7 实验评估

我们通过实验评估 UniTune。我们在第 7.1 节介绍实验设置，在第 6 节与基线方法进行端到端比较，在第 7.3 节分析 UniTune，并在第 7.4 节介绍案例研究。

7.1 实验设置

智能体。在大多数实验中，我们对数据库管理系统中的三个组件进行调优——索引、调优参数和 SQL 查询。除非另有说明，我们采用基于贝叶斯优化的智能体 OtterTune [6] 进行参数调优，采用基于贝叶斯优化的智能体 DBA-Bandit [52] 进行索引选择，并采用基于强化学习估计器的智能体 LearnedRewrite [72]进行查询重写。为展示 UniTune 的可扩展性，我们用其他参数调优智能体替换 OtterTune：CDBTune [68] 和 MysqlTuner [4]。我们还添加了基于强化学习估计器的智能体 AutoView [27] 来调优视图组件。**工作负载**。我们考虑三个分析型基准测试：JOB [40]、缩放因子为 10 的 TPC-H [1] 和缩放因子为 50 的 TPC-DS [2]，因为它们广泛用于调优智能体的评估 [27, 34, 42, 52, 71]。此外，大多数用于查询重写或视图生成的智能体都面向联机分析处理场景。JOB 数据集代表一个数据分布不均衡的典型真实数据库。我们使用其工作负载中提供的 113 条查询。TPC-H 和 TPC-DS 都是决策支持基准测试，用于模拟真实的数据仓库环境。对于 TPC-H，部分查询的开销比平均值高出数个数量级。将这些查询加入工作负载会使索引选择问题简单得多 [34]，因为能降低其中至少一条查询开销的索引，其效果总会比针对其他查询的索引高出数个数量级。因此，我们采用 12 条 TPC-H 查询，其中包含多种常见执行模式（即流水线、包含多个连接和筛选条件的作业，以及分组聚合）。我们还将 LearnedRewrite [72] 所用合成查询集中的三条编写质量较差的查询加入 TPC-H 工作负载。对于 TPC-DS，我们使用其工作负载中提供的 99 条查询。

性能指标和配置空间。我们优化给定工作负载的查询执行总时间。对于每个工作负载，我们将所有可能有用的索引（例如，出现在谓词中的列）组成索引候选集[12]，并将其作为索引选择工具的起点，该工具会从中选取一部分索引。我们只考虑单列索引，为 JOB 生成 59 个索引候选项，为 TPC-H 生成 54 个，为 TPC-DS 生成 237 个。对于 JOB 和 TPC-H，我们为已构建的索引设置 1500 MB 的存储预算；对于 TPC-DS，由于其数据集更大且模式复杂得多，我们将预算设为 20000 MB。对于这三个工作负载，我们调优 50 个旋钮。我们采用 Calcite 中的查询重写规则 [8], , 如 LearnedRewrite[72] 所述。至于视图生成，我们为 JOB 考虑 29 个视图候选项，并遵循 AutoView [27]中的视图生成模块。

基线方法。端到端评估中使用的基线方法如下：

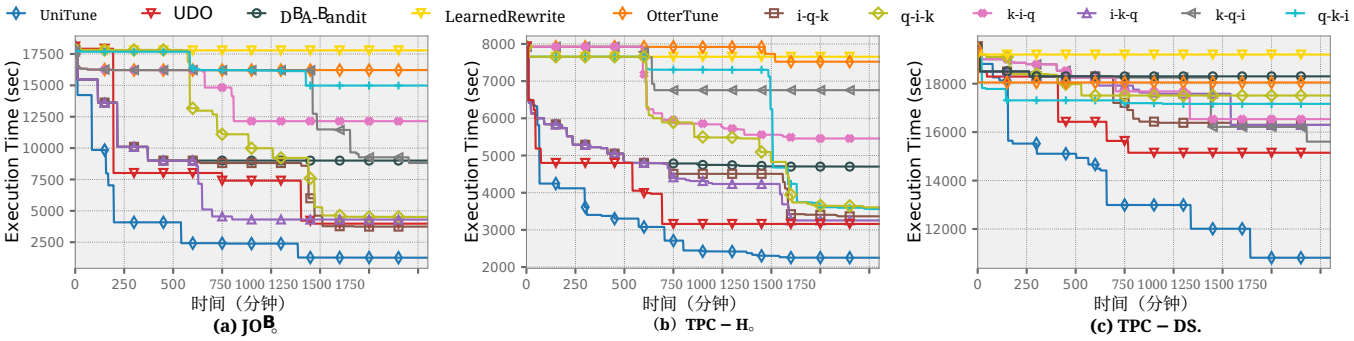
- **独立调优**使用单个智能体调优单个组件，与以往研究的做法相同。我们比较了三种独立调优基线方法：DBA-Bandit [52], LearnedRewrite [72] 和 OtterTune [6]。
- **顺序调优**是调优数据库管理系统中多个组件的一种原生方式。给定调优预算，数据库管理员可以将其平均分配给各个调优智能体，并让它们依次运行。我们比较执行顺序的所有排列：i-q-k、i-k-q、q-i-k、q-k-i、k-i-q 和 k-q-i。（i-k-q 表示索引-调优参数-查询，其他顺序以此类推。）

- **UDO [59]** 采用双层架构，将重型参数和轻型参数的调优分开。在外层，一个负责重型参数的智能体会建议并应用一种重型配置。随后，在内层，负责轻型参数的智能体会进行固定次数的内层迭代——在已应用的重型配置下搜索合适的轻型配置并对其进行评估。内循环中取得的最佳评估性能被视为该重型配置的性能，并用于更新外层智能体。我们采用作者发布的实现 [3]。为保证比较公平，我们添加 LearnedRewrite，作为查询重写的智能体，因为 UDO 目前尚不支持查询重写。按照论文 [59]，我们将查询代理视为内层智能体，因为它不会配置数据库管理系统的物理结构。查询代理会在调优参数之前重写查询，以缩短其执行时间。根据实验，我们将其内层迭代次数设为三次。因此，对于每个索引配置，查询代理的内层迭代次数为三次，参数代理的内层迭代次数为五次（与已发布的实现 [3]相同）。

设置。我们优化了部署在配备16个虚拟CPU和32GB内存的云ECS实例上的MySQL数据库。初始配置采用MySQL官方默认配置，且未建立索引和视图。我们将调优预算设为30小时，并将运行单个查询的超时时间设为10分钟。对于UniTune，我们将子预算设为20分钟，默认使用大小为7的记忆缓冲区。我们还改变缓冲区大小，以测试其鲁棒性。在分配前9个子预算（即初始化阶段）时，UniTune以轮询方式运行智能体，以启动汤普森采样。为了设置算法2中的 $rfactor$，我们观察初始化阶段内智能体执行 r_{max}时获得的最大奖励，并将 $rfactor$设为 $\frac{r_{max}}{20}$，使缩放后的奖励约为0至20。

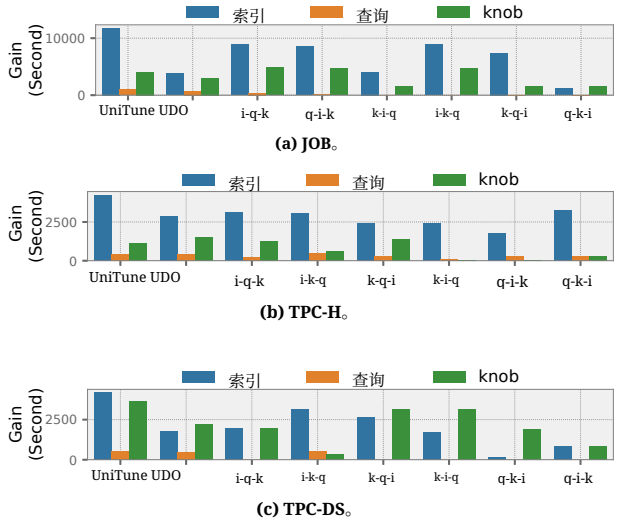
7.2 端到端评估

我们将 UniTune 与基线方法进行比较。图6展示了它们随时间取得的最佳性能。图7则进一步分解了



(a) JOB。 (b) TPC－H。 (c) TPC－DS.

Fi图6：不同基线方法随时间取得的最佳性能（左下方更优）。



(a) JOB。

(b) TPC-H。

(c) TPC-DS。

图7：按调优智能体划分的性能细目。

三个智能体在多个组件调优中的性能提升。我们发现，UniTune 在两个工作负载上都找到了最优配置；与基线方法相比，其最优配置的执行时间加速了 1.4×~14.1× 。无论调优预算（横轴）是多少，UniTune 的表现都优于基线方法。总体而言，独立调优的效果不如多组件调优，因为它只能在子空间中找到次优配置。采用不同调优顺序的六种顺序方法收敛情况各不相同。如图7所示，不同顺序会使三个智能体取得不同的性能提升，这表明它们之间存在依赖关系。考虑三个工作负载后，各种调优顺序并无明显的优胜者。不过，先调优索引再调优旋钮通常能带来更好的性能。

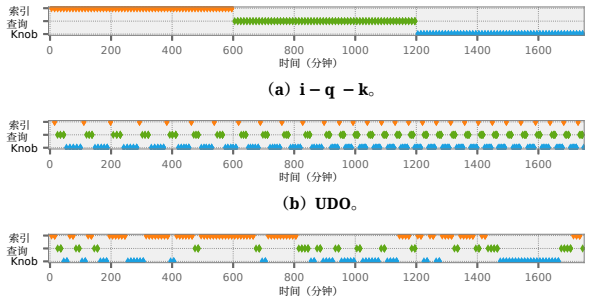
为了深入分析，我们研究了基线方法对多个组件进行调优的情况，如图8所示。我们观察到，UniTune 利用历史观测，同时探索运行次数较少的智能体，从而学会了合适的分配策略。它将更多预算分配给索引和调优参数，而非查询，因为与查询重写相比，调优索引和调优参数带来的性能提升更大，适用于该基准工作负载。UniTune 在索引、查询和调优参数上探索的配置数量约为 4:2:3。在调优的前半阶段，UniTune 学会了优先优化有望取得更好效果的组件（即索引），因为索引能带来更大的调优收益，如图7所示。随后，由于调优索引带来的性能提升趋于饱和，它在后半阶段探索其他智能体。在大多数情况下，UDO 的表现优于顺序方法，尤其是在数据规模较大时（即 TPC-H 和 TPC-DS）。UDO 中的索引代理会重新排列索引配置，大幅降低重新配置开销。然而，UDO 的表现不如 UniTune。其双层架构无法灵活地将调优预算投入到有前景的智能体上，导致分配给外层代理（即索引代理）的预算不足，如图8(b)所示。UDO 在外层采用一种索引配置，并在内层迭代中执行查询代理三次、参数代理五次。调优期间，UDO 在索引、查询和调优参数上探索的配置比例固定为 1:3:5，这与增加索引调优预算有益的实际情况不符。

7.3 UniTune 分析

我们精心设计了 UniTune，其中包含消息传播协议和预算分配策略。本节将分析 UniTune 的执行时间，通过消融研究和变体比较评估相应设计，然后验证 UniTune 在内存缓冲区大小设置方面的鲁棒性。

7.3.1 执行时间分解。 UniTune 的总执行时间包括智能体选择时间和智能体执行时间。后者由子预算控制，在此期间，选定的智能体会对数据库管理系统进行调优并更新其调优策略。前者可以忽略不计，因为它依据算法2中的闭式方程执行，无需训练机器学习模型。在我们的实验中，这一过程始终耗时不到 2 毫秒。

7.3.2 上下文特征消融研究。我们将上下文特征封装在智能体模型中，以学习调优策略。为验证上下文特征的作用，我们将 UniTune 与其他不使用上下文特征的方案进行比较，包括：(1) UniTune-w/o-C(reinit)，当背景环境发生变化时（即其他智能体改变其组件的配置），该方案会重新初始化智能体的模型，如第5.2节所述。(2) UniTune-w/o-C，该方案

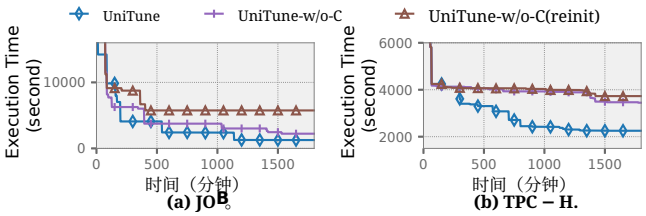


（a）i－q－k。

（b）UDO。

（c）UniTune。

图8：一种顺序调优方法（i-q-k）、UDO 和 UniTune 在 JOB 上随时间对索引、查询和旋钮智能体进行预算分配的模式。（色块表示在相应时间段内向相应智能体分配预算。）

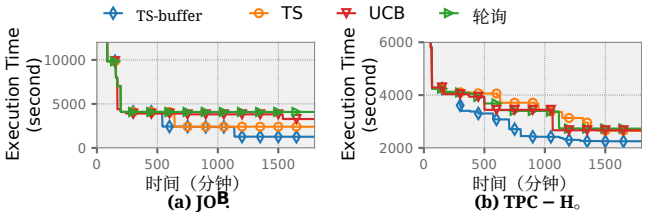


(a) JOB。 (b) TPC－H.

图9：上下文特征消融研究。

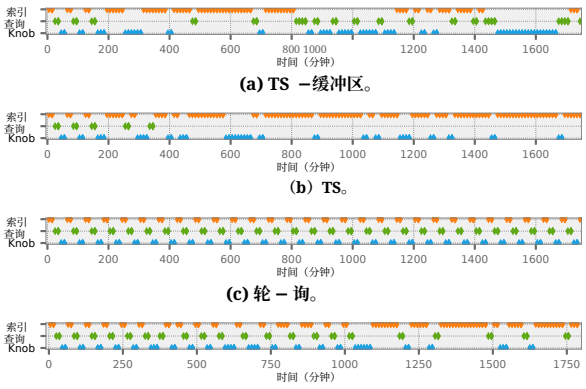
忽略环境变化。如图9所示，UniTune 的表现优于不使用上下文特征的方案。UniTune-w/o-C(reinit) 的表现最差，因为其重新学习方法会丢失先前的知识。与 UniTune-w/o-C 相比，UniTune 更具优势，这表明不同环境下配置与性能之间的关系会发生变化。

7.3.3 不同预算分配策略的比较。 我们提出了一种基于汤普森采样、使用记忆缓冲区在各智能体之间分配调优预算的策略。我们比较了四种分配策略：(1) TS-buffer，即基于汤普森采样并使用记忆缓冲区的策略（UniTune采用的策略）；(2) TS，即传统的汤普森采样；(3) 轮询，即按照索引-查询-旋钮的顺序平均分配预算，如第6节所述；(4) UCB，即采用上下文 UCB算法 [14]。该算法学习从上下文到智能体奖励的映射，并选择奖励的上置信界最大的智能体。图10展示了比较结果，图11展示了它们的预算分配模式。TS-buffer优于其他策略，表明它能够合理分配调优预算。在调优的前半阶段，它向索引代理分配更多预算，因为配置索引能带来更大的性能提升，尤其是在引导阶段。到了后半阶段，分配给索引代理的预算有所减少，因为调优索引带来的性能提升趋于饱和。TS不会丢弃过时信息，因此在后半阶段向索引代理分配了过多预算。轮询平均分配预算，导致预算浪费在前景较差的智能体上（即查询



(a) JOB. (b) TPC－H。

图10：不同预算分配策略的比较。TS-buffer 是 UniTune 采用的策略。



(a) TS －缓冲区。

（b）TS。

(c) 轮－询。

(d) UCB.

图11：JOB 上索引、旋钮和查询智能体随时间变化的不同策略下的预算分配。

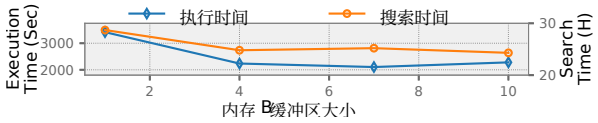


图 12：内存缓冲区大小的影响。我们报告最佳性能（即最短执行时间）以及达到最佳性能所需的搜索时间。

智能体）。UCB 对给定上下文中选择某个智能体所获得的奖励进行建模。与 TS-buffer 在时间窗口内根据观测到的奖励直接选择智能体不同，学习该模型需要更多观测。在观测极其有限的情况下（如我们的情况），由于输入上下文不太可能被先前的观测涵盖，UCB 往往会探索各个智能体。

7.3.4 内存缓冲区大小的影响。UniTune 限制了用于评估选择智能体奖励的观测数量。较小的缓冲区大小能够更及时地响应奖励变化，但也可能导致有用信息的丢失。我们评估了采用不同内存缓冲区大小的 UniTune，并在图 12 中展示其性能。缓冲区大小为 1 时性能较差，因为未能利用历史观测。总体而言，缓冲区大小设为 4 到 10 时性能良好（在我们的实验中，默认设置为 7）。

7.4 案例研究

我们通过案例研究展示 UniTune 的优势。

表2：不同方法取得的最佳性能及其对应的搜索时间。

	默认	网格搜索	UniTune	TS	轮询	UDO	i-q-k
搜索时间（小时）	/	934.4	**2.6**	3.2	3.8	3.1	4.6
执行时间（秒）	581.41	**27.11**	27.32	27.32	27.32	29.87	29.76

7.4.1 与网格搜索的比较。 为了构建真实基准，我们定义了一个小型配置空间，可通过网格搜索进行枚举。我们对 TPC-H 中的一个查询 Q19 进行优化。在 1500 MB 的存储预算下，它有 10 个索引候选项和 81 种可行的索引配置。我们根据基尼系数选出三个重要调优参数[1]，并规定每个参数只能取三个值，因此得到 63 种可行的旋钮配置[2]。此外，Q19 有四种重写方式，不考虑循环重写。由此，配置空间中共有 20412（$81 \times 63 \times 4$）种可能的配置，我们可以通过网格搜索评估其性能，从而获得最优配置。表2展示了不同基线方法的性能；受篇幅限制，我们仅展示最佳的顺序调优方法。网格搜索能够找到最优配置，但搜索时间极长。UniTune 则以最短的搜索时间找到接近最优的配置。TS 和轮询也能找到相同的配置，但搜索时间更长。

7.4.2 联合优化与交替优化的比较。UniTune 交替执行各智能体。要调优数据库管理系统中的多个组件，另一种方案是在联合配置空间上采用中心智能体，如第5.1节所述。由于合并子空间会导致配置空间呈指数级增长，联合优化存在可扩展性问题。我们通过实证验证了这一分析。我们采用这两种方法，对 JOB 中的调优参数和索引组件进行调优。对于交替优化，我们采用两个基于贝叶斯优化的智能体。对于联合优化，我们采用一个基于贝叶斯优化的智能体，在联合配置空间上进行优化。如图13所示，与联合优化方法相比，交替优化方法找到的配置更优。以调优 50 个调优参数的参数代理和拥有 59 个索引候选项的索引代理为例。为便于分析，我们假设每个调优参数有 10 个可能取值。于是，联合优化方法要在一个配置空间中进行优化，该空间包含

大小为 $10^{50} \times 2^{59}$，是两个原始子空间总和的 10_{17} 倍。交替优化通过按照原始智能体的假设分解配置空间来解决这一可扩展性问题，并且收敛速度更快。

7.4.3 可扩展性。 我们在 UniTune 中集成了不同的调优智能体，以展示其可扩展性。

采用不同的参数代理。我们仍在 TPC-H 中调优三个组件：索引、旋钮和查询。但在 UniTune 中，我们分别采用不同的参数调优智能体：OtterTune [6]、、CDBTune [68], 和 MySQLTuner [4]。MySQLTuner 是一种基于规则的调优智能体。它会检查数据库管理系统指标，并使用启发式规则建议旋钮配置。图 14 展示了结果。采用这两种基于机器学习的调优器时，UniTune 的性能都收敛到相近水平。

[1]它们分别是 innodb_buffer_pool_size、innodb_log_file_size、innodb_thread_concurrency 和 table_definition_cache。[2]左侧的 18 种不可行配置违反了以下规则：ib_logfiles 的组合大小应大于 200 kB × innodb_thread_concurrency，导致 MySQL 数据库关闭。

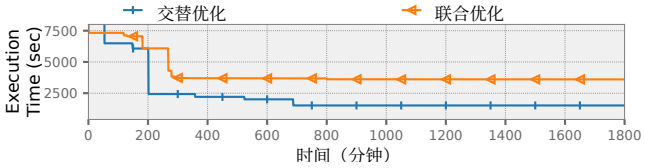
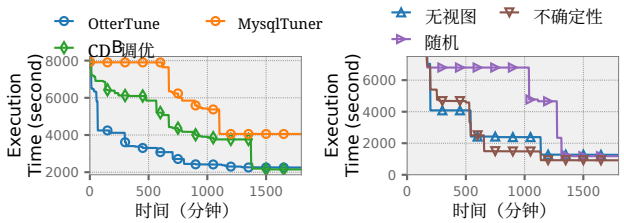


图13：联合优化与交替优化的比较。



（a）不同的参数代理。 **（b）添加视图智能体。**

图14：UniTune 的可扩展性。在图14a中，我们分别采用了三个参数代理。在图14b中，我们添加了一个用于视图生成的智能体，并比较第5节.3中讨论的不确定性感知采样与随机采样。

但采用 CDBTune 时收敛较慢，因为其强化学习智能体需要更多调优预算来学习大量神经网络参数，这与现有研究一致 [7, 69]。采用 MysqlTuner 时，性能较差，因为其有限的启发式规则无法找到良好的旋钮配置。不过，这种做法表明，UniTune 可以扩展到非基于机器学习的智能体，例如基于规则和基于成本的智能体。当非基于机器学习的智能体由 UniTune 协调时，预算分配模块会像对待基于机器学习的智能体一样对待它：UniTune 为其分配调优预算后，该智能体便会执行。主要区别在于，非基于机器学习的智能体不需要上下文特征来建议配置，因为它们是依据固定规则或数据库优化器的成本估算来建议配置，无需训练模型。

添加视图智能体。 我们添加一个基于强化学习估计器的智能体 AutoView [27]，用于视图生成，并在 JOB 中调优四个组件，将物化视图的存储预算设为 500 MB。我们比较了训练估计器时采用的两种策略，如第5.3节所述。如图 14b 所示，与调优三个组件（即不包含视图）相比，调优四个组件可获得更好的性能，因为物化视图能够减少查询之间的冗余计算。此外，不确定性感知采样也能带来更好的性能。

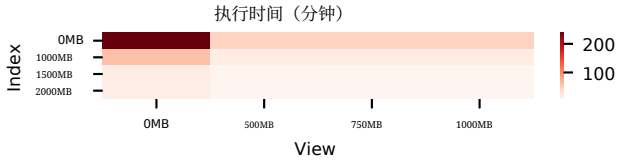


图 15：索引和视图的存储预算对最佳调优性能（即最短执行时间）的影响。

7.4.4 调整存储预算。 在调优会话开始时，用户会根据自身可承受的资源确定存储预算。在此实验中，我们改变 JOB 工作负载下索引和视图的存储预算，并在图 15 中展示 UniTune 调优后的最佳性能。我们观察到，随着给定存储预算的增加，调优性能逐步提升，随后趋于饱和。对于视图组件，存储预算达到 500 MB 后，其调优收益便趋于饱和，这与 AutoView 的报告 [27]一致。对于索引组件，其调优收益从零存储预算增加到 1500 MB 时显著提升，之后增幅趋缓。在实验中，我们将 JOB 工作负载下索引的存储预算设为 1500 MB，视图的存储预算设为 500 MB。

8 结论

本文回顾了数据库领域中关于基于机器学习的调优代理的新兴研究，并提出了一个问题：“如何让它们协同工作，以配置数据库管理系统中的多个组件？”为回答这一问题，我们提出了面向基于机器学习的调优代理的统一高效协调框架 UniTune。我们设计了一种协调协议，以实现调优智能体之间的协作，并提出了一种在智能体之间分配调优预算的策略。我们还定义了适用于广泛类型的基于机器学习的调优代理的接口，便于与现有实现集成并支持未来扩展。我们证明，UniTune 能够支持不同的基于机器学习的智能体，且性能显著优于基线方法。

致谢

本研究得到国家自然科学基金委员会（NSFC）（项目编号：61832001、U22B2037）、阿里巴巴集团通过阿里巴巴创新研究计划以及国家重点研发计划的支持。Bin Cui 和 Yang Li 为通讯作者。

参考文献

[1] 2015. TPC-H 基准测试。http://www.tpc.org/tpch/. [2] 2015. TPC-H 基准测试。https://www.tpc.org/tpcds/. [3] 2015. UDO 实现。https://github.com/jxiw/UDO. [4] 2019. MySQL 调优入门脚本。https://github.com/major/MySQLTuner- perl. [5] Shipra Agrawal 和 Navin Goyal。2012。多臂老虎机问题的汤普森采样分析。载于 *COLT*（*JMLR* 会议论文集，第 *23* 卷）。JMLR.org，39.1–39.26。[6] Dana Van Aken、Andrew Pavlo、Geoffrey J. Gordon 和 Bohan Zhang。2017。通过大规模机器学习实现数据库管理系统自动调优。载于 *SIGMOD* 会议。ACM，1009–1024。[7] Dana Van Aken、Dongsheng Yang、Sebastien Brillard、Ari Fiorino、Bohan Zhang、Christian Billian 和 Andrew Pavlo。2021。探究真实世界数据库管理系统中的机器学习自动配置调优服务。*Proc. VLDB Endow.* 14，7（2021），1241–1253。[8] Edmon Begoli、Jesús Camacho-Rodríguez、Julian Hyde、Michael J. Mior 和 Daniel Lemire。2018。Apache Calcite：一个用于优化的基础框架异构数据源上的查询处理。在 SIGMOD 会议上。ACM，221–230。[9] Sébastien Bubeck 和 Nicolò Cesa-Bianchi。2012。随机与非随机多臂老虎机问题的遗憾分析。Found. Trends Mach. Learn. 5, 1 2012 1 122( ), – .[10] Baoqing Cai、Yu Liu、Ce Zhang、Guangyu Zhang、Ke Zhou、Li Liu、Chunhua Li、Bin Cheng、Jie Yang 和 Jiashu Xing。2022。HUNTER：面向个性化需求的在线云数据库混合调优系统。在 SIGMOD 会议上。ACM，646–659。[11] Stefano Cereda、Stefano Valladares、Paolo Cremonesi 和 Stefano Doni。2021。CGPTuner：一种用于在工作负载条件变化时自动调优 IT 配置的上下文高斯过程老虎机方法。Proc. VLDB Endow. 14, 8 (2021), 1401–1413。[12] Surajit Chaudhuri 和 Vivek R. Narasayya。1997。用于 Microsoft SQL Server 的高效成本驱动索引选择工具。在 VLDB'97，《第 23 届超大型数据库国际会议论文集》，1997 年 8 月 25–29 日，希腊雅典，Matthias Jarke、Michael J. Carey、Klaus R. Dittrich、Frederick H. Lochovsky、Pericles Loucopoulos 和 Manfred A. Jeusfeld（编）。Morgan Kaufmann，146–155。http://www.vldb.org/conf/1997/P146.PDF[13] Jin Chen、Guanyu Ye、Yan Zhao、Shuncheng Liu、Liwei Deng、Xu Chen、Rui Zhou 和 Kai Zheng。2022。利用基于图的表示，高效学习连接顺序选择。在 KDD 上。ACM，97–107。[14] Wei Chu、Lihong Li、Lev Reyzin 和 Robert E. Schapire。2011。具有线性收益函数的上下文老虎机。在 AISTATS（JMLR 会议论文集，第 15 卷）上。JMLR.org，208–214。[15] Bailu Ding、Sudipto Das、Ryan Marcus、Wentao Wu、Surajit Chaudhuri 和 Vivek R. Narasayya。2019。AI遇见AI：利用查询执行改进索引推荐。在 SIGMOD 会议上。ACM，1241–1258。[16] Songyun Duan、Vamsidhar Thummala 和 Shivnath Babu。2009。使用 iTuned 调优数据库配置参数。Proc. VLDB Endow. 2, 1 (2009), 1246–1257。[17] Gabriel Campero Durand、Marcus Pinnecke、Rufat Piriyev、Mahmoud Mohsen、David Broneske、Gunter Saake、Maya S. Sekeran、Fabián Rodriguez 和 Laxmi Balami。2018。GridFormation：探索利用强化学习实现自主在线数据分区。在 aiDM@SIGMOD 上。ACM，1:1–1:7。[18] Gabriel Campero Durand、Rufat Piriyev、Marcus Pinnecke、David Broneske、Balasubramanian Gurumurthy 和 Gunter Saake。2019。利用深度强化学习实现自动垂直分区。在 ADBIS（短文与研讨会）（计算机与信息科学通信，第 1064 卷）上。Springer，126–134。[19] Tamer Eldeeb、Zhengneng Chen、Asaf Cidon 和 Junfeng Yang。2022。Neuroshard：探索利用深度强化学习实现自动多目标分片。在 aiDM@SIGMOD 上。ACM，1:1–1:12。[20] Ayat Fekry、Lucian Carata、Thomas F. J.-M. Pasquier、Andrew Rice 和 Andy Hopper。2020。要不要调优？寻找数据分析的最优配置。在 KDD 上。ACM，2494–2504。[21] Béatrice Finance 和 Georges Gardarin。1991。可扩展数据库管理系统中的基于规则的查询重写器。在 ICDE 上。IEEE计算机学会，248–256。[22] Jianling Gao、Nan Zhao、Ning Wang、Shuang Hao 和 Haoyan Wu。2022。利用学习型成本估算器自动选择索引。Information Sciences (2022)。[23] Jia-Ke Ge、Yanfeng Chai 和 Yunpeng Chai。2021。WATuning：采用基于注意力的深度强化学习的工作负载感知调优系统。J. Comput. Sci. Technol. 36, 4 (2021), 741–761。[24] Yaniv Gur、Dongsheng Yang、Frederik Stalschus 和 Berthold Reinwald。2021。面向在线数据库调优的自适应多模型强化学习。在 EDBT 上。OpenProceedings.org，439–444。[25] Yonatan Gur、Assaf Zeevi 和 Omar Besbes。2014。具有非平稳奖励的随机多臂老虎机问题。在 NIPS 上。199–207。[26] Shuai Han、Mingxia Liu 和 Jian-Zhong Li。2022。用于优化数组数据压缩的高效分区方法。J. Comput. Sci. Technol. 37, 5 (2022), 1049–1067。[27] Yue Han、Guoliang Li、Haitao Yuan 和 Ji Sun。2021。采用深度强化学习的自主物化视图管理系统。在 ICDE 上。IEEE，2159–2164。[28] Jonas Heitz 和 Kurt Stockinger。2019。采用深度强化学习算法的连接查询优化。CoRR abs/1911.11689 (2019)。[29] Benjamin Hilprecht、Carsten Binnig 和 Uwe Röhm。2020。为云数据库学习分区顾问。在 SIGMOD 会议上。ACM，143–157。[30] Shiyue Huang、Yanzhao Qin、Xinyi Zhang、Yaofeng Tu、Zhongliang Li 和 Bin Cui。2023。数据库系统性能优化综述。Sci. China Inf. Sci. 66, 2 (2023)。[31] Yoshiteru Ishihara 和 Masahito Shiba。2020。运行中的数据库管理系统的动态配置调优。在 LifeTech 上。IEEE，393–397。[32] Konstantinos Kanellis、Ramnatthan Alagappan 和 Shivaram Venkataraman。2020。调优参数太多？通过预先选择重要调优参数加快数据库调优。在 HotStorage 上。USENIX协会。[33] Konstantinos Kanellis、Cong Ding、Brian Kroth、Andreas Müller、Carlo Curino 和 Shivaram Venkataraman。2022。LlamaTune：样本高效的数据库管理系统配置调优。Proc. VLDB Endow. 15, 11 (2022), 2953–2965。

[34] Jan Kossmann、Stefan Halfpap、Marcel Jankrift 和 Rainer Schlosser。2020。魔镜魔镜，谁是天下第一？索引选择算法的实验评估。Proc. VLDB Endow. 13, 11 (2020), 2382–2395。[35] Jan Kossmann、Alexander Kastius 和 Rainer Schlosser。2022。SWIRL：利用强化学习选择工作负载感知索引。在 EDBT 上。OpenProceedings.org，2:155–2:168。[36] Sanjay Krishnan、Zongheng Yang、Ken Goldberg、Joseph M. Hellerstein 和 Ion Stoica。2018。利用深度强化学习学习优化连接查询。CoRR abs/1808.03196 (2018)。[37] Mayuresh Kunjir 和 Shivnath Babu。2020。黑还是白？如何开发用于基于内存分析的自动调优器。在 SIGMOD 会议上。ACM，1667–1683。[38] Hai Lan、Zhifeng Bao 和 Yuwei Peng。2020。基于深度强化学习的索引顾问。在 CIKM 上。ACM，2105–2108。[39] Hai Lan、Zhifeng Bao 和 Yuwei Peng。2021。推进数据库管理系统查询优化器综述：基数估计、成本模型与计划枚举。Data Sci. Eng. 6, 1 (2021), 86–101。[40] Viktor Leis、Andrey Gubichev、Atanas Mirchev、Peter A. Boncz、Alfons Kemper 和 Thomas Neumann。2015。查询优化器究竟有多出色？Proc. VLDB Endow. 9, 3 (2015), 204–215。[41] Guoliang Li、Xuanhe Zhou 和 Lei Cao。2021。数据库机器学习。Proc. VLDB Endow. 14, 12 (2021), 3190–3193。[42] Guoliang Li、Xuanhe Zhou、Shifu Li 和 Bo Gao。2019。QTune：采用深度强化学习的查询感知数据库调优系统。Proc. VLDB Endow. 12, 12 (2019), 2118–2130。[43] Yang Li、Jiawei Jiang、Jinyang Gao、Yingxia Shao、Ce Zhang 和 Bin Cui。2020。利用渐升式多臂老虎机实现高效的自动化 CASH。在 AAAI 上。AAAI Press，4763–4771。[44] Yang Li、Yu Shen、Huaijun Jiang、Tianyi Bai、Wentao Zhang、Ce Zhang 和 Bin Cui。2022。基于迁移学习的超参数调优搜索空间设计。在 KDD 上。ACM，967–977。[45] Yang Li、Yu Shen、Wentao Zhang、Jiawei Jiang、Yaliang Li、Bolin Ding、Jingren Zhou、Zhi Yang、Wentao Wu、Ce Zhang 和 Bin Cui。2021。VolcanoML：通过可扩展的搜索空间分解加速端到端自动化机器学习。Proc. VLDB Endow. 14, 11 (2021), 2167–2176。[46] Xi Liang、Aaron J. Elmore 和 Sanjay Krishnan。2019。利用深度强化学习机会式地实现视图实体化。CoRR abs/1903.01363 (2019)。[47] Gabriel Paludo Licks、Júlia Mara Colleoni Couto、Priscilla de Fátima Miehe、Renata De Paris、Duncan Dubugras A. Ruiz 和 Felipe Meneguzzi。2020。SmartIX：一种基于强化学习的数据库索引代理。Appl. Intell. 50, 8 (2020), 2575–2588。[48] Sijia Liu、Parikshit Ram、Deepak Vijaykeerthy、Djallel Bouneffouf、Gregory Bramble、Horst Samulowitz、Dakuo Wang、Andrew Conn 和 Alexander G. Gray。2020。用于自动化机器学习流水线配置的基于交替方向乘子法的框架。在 AAAI 上。AAAI Press，4892–4899。[49] Wei Lu、Xinyi Zhang、Zhiyu Shui、Zhe Peng、Xiao Zhang、Xiaoyong Du、Hao Huang、Xiaoyu Wang、AnqunPan 和 Haixiang Li。2018。MSQL+：用于分布式关系数据库系统中度量空间相似性搜索的插件工具包。Proc. VLDB Endow. 11, 12 (2018), 1970–1973。[50] Ryan Marcus 和 Olga Papaemmanouil。2018。利用深度强化学习进行连接顺序枚举。在 aiDM@SIGMOD 上。ACM，3:1–3:4。[51] Andrew Pavlo、Matthew Butrovich、Ananya Joshi、Lin Ma、Prashanth Menon、Dana Van Aken、Lisa Lee 和 Ruslan Salakhutdinov。2019。外部与内部：关于自主数据库管理系统机器学习代理的论文。IEEE Data Eng. Bull. 42, 2 (2019), 32–46。[52] R. Malinga Perera、Bastian Oetomo、Benjamin I. P. Rubinstein 和 Renata Borovica-Gajic。2021。DBA多臂老虎机：在即席分析型工作负载下实现带有安全保证的自动驾驶式索引调优。在 ICDE 上。IEEE，600–611。[53] Daniel Russo、Benjamin Van Roy、Abbas Kazerouni、Ian Osband 和 Zheng Wen。2018。汤普森采样教程。Found. Trends Mach. Learn. 11, 1 (2018),

1 96 10 1561 2200000070 – 。https://doi.org/ 。/ Zahra Sadri、Le [54] Gruenwald 和 Eleazar Leal。2020。利用深度强化学习在集群数据库中进行在线索引选择。在 ICDE研讨会上。IEEE，

158–161.[55] Vishal Sharma 和 Curtis E. Dyreson。2022。Indexer++：结合 Transformer 和强化学习的工作负载感知在线索引调优。载于 SAC。ACM，372–380。[56] Vishal Sharma、Curtis E. Dyreson 和 Nicholas Flann。2021。MANTIS：使用深度强化学习进行多类型和多属性索引选择。载于 IDEAS。ACM，56–64。[57] Jian Tan、Tieying Zhang、Feifei Li、Jie Chen、Qixing Zheng、Ping Zhang、Honglin Qiao、Yue Shi、Wei Cao 和 Rui Zhang。2019。i BTune：大规模云数据库的个性化缓冲区调优。Proc. VLDB Endow. 12，10（2019），1221–1234。[58] Danshi Wang、Chunyu Zhang、Wenbin Chen、Hui Yang、Min Zhang 和 Alan Pak Tao Lau。2022。基于机器学习的光网络故障管理综述。Sci. China Inf. Sci. 65，11（2022）。[59] Junxiong Wang、Immanuel Trummer 和 Debabrota Basu。2021。UDO：使用强化学习进行通用数据库优化。Proc. VLDB Endow. 14，13（2021），3402–3414。[60] Zhaoguo Wang、Zhou Zhou、Yicun Yang、Haoran Ding、Gansen Hu、Ding Ding、Chuzhe Tang、Haibo Chen 和 Jinyang Li。2022。WeTune：查询重写规则的自动发现与验证。载于 SIGMOD 会议。ACM，94–107。[61] Ziyu Wang、Masrour Zoghi、Frank Hutter、David Matheson 和 Nando de Freitas。2013。通过随机嵌入实现高维贝叶斯优化。载于 IJCAI。IJCAI/AAAI，1778–1784。[62] Garrett Warnell、Nicholas R. Waytowich、Vernon Lawhern 和 Peter Stone。2018。Deep TAMER：高维状态空间中的交互式智能体塑造。载于 AAAI。AAAI Press，1545–1554。[63] Sai Wu、Ying Li、Haoqi Zhu、Junbo Zhao 和 Gang Chen。2022。使用深度强化学习动态构建索引。Data Sci. Eng. 7，2（2022），87–101。[64] Wentao Wu、Chi Wang、Tarique Siddiqui、Junxiong Wang、Vivek R. Narasayya、Surajit Chaudhuri 和 Philip A. Bernstein。2022。基于强化学习的预算感知索引调优。载于 SIGMOD 会议。ACM，1528–1541。[65] Qiufen Xia、Lizhen Zhou、Wenhao Ren 和 Yi Wang。2022。在边缘云中利用物化视图主动智能地评估大数据查询。Comput. Networks 203（2022），108664。[66] Xiang Yu、Guoliang Li、Chengliang Chai 和 Nan Tang。2020。使用 Tree-LSTM 和强化学习进行连接顺序选择。载于 ICDE。IEEE，1297–1308。[67] Haitao Yuan、Guoliang Li、Ling Feng、Ji Sun 和 Yue Han。2020。使用深度学习和强化学习自动生成视图。载于 ICDE。IEEE，1501–1512。[68] Ji Zhang、Yu Liu、Ke Zhou、Guoliang Li、Zhili Xiao、Bin Cheng、Jiashu Xing、Yangtao Wang、Tianheng Cheng、Li Liu、Minwei Ran 和 Zekang Li。2019。使用深度强化学习的端到端自动云数据库调优系统。载于 SIGMOD 会议。ACM，415–432。[69] 张欣怡、常卓、Yang Li、Hong Wu、Jian Tan、Feifei Li 和 Bin Cui。2022。利用超参数优化促进数据库调优：全面的实验评估。Proc. VLDB Endow. 15，9（2022），1808–1821。[70] 张欣怡、Hong Wu、常卓、Shuowei Jin、Jian Tan、Feifei Li、Tieying Zhang 和 Bin Cui。2021。ResTune：通过元学习提升的面向资源的云数据库调优。载于 SIGMOD 会议。ACM，2102–2114。[71] 张欣怡、Hong Wu、Yang Li、Jian Tan、Feifei Li 和 Bin Cui。2022。迈向云数据库的动态安全配置调优。载于 SIGMOD 会议。ACM，631–645。[72] Xuanhe Zhou、Guoliang Li、Chengliang Chai 和 Jianhua Feng。2021。使用蒙特卡洛树搜索的学习式查询重写系统。Proc. VLDB Endow. 15，1（2021），46–58。[73] Xuanhe Zhou、Luyang Liu、Wenbo Li、Lianyuan Jin、Shifu Li、Tianqing Wang 和 Jianhua Feng。2022。AutoIndex：面向动态工作负载的增量式索引管理系统。载于 ICDE。IEEE，2196–2208。[74] Jia Zou、Amitabh Das、Pratik Barhate、Arun Iyengar、Binhang Yuan、Dimitrije Jankov 和 Chris Jermaine。2021。Lachesis：面向 UDF 分析的自动分区。Proc. VLDB Endow. 14，8（2021），1262–1275。