arXiv:2602.04181v1 [cs.DB] 4 Feb 2026

Piece of CAKE：通过微秒级学习实现自适应执行引擎

Zijie Zhao 宾夕法尼亚大学
bytez@seas.upenn.edu

Ryan Marcus 宾夕法尼亚大学
rcmarcus@seas.upenn.edu

摘要

底层数据库算子通常有多种物理实现（“内核”），这些实现语义等价，但其性能特征会因输入数据分布不同而大相径庭。现有数据库系统通常依赖静态启发式方法或最坏情况最优的默认方案来选择内核，因此常常错失显著的性能提升机会。本文提出 CAKE（反事实自适应内核执行），这是一种利用微秒级上下文多臂老虎机，为每个数据“数据块”学习选择最优内核的系统。CAKE 利用反事实成本低廉这一特点——有选择地运行多个内核以获取完整反馈——并将策略编译为低延迟遗憾树，从而避免传统强化学习的高延迟。实验结果表明，与最先进的静态启发式方法相比，CAKE 可将端到端工作负载延迟最多降低 2× 。

PVLDB 参考文献格式：
Zijie Zhao 与 Ryan Marcus。Piece of CAKE：通过微秒级学习实现自适应执行引擎。PVLDB，19(1)：XXX-XXX，2026。doi:XX.XX/XXX.XX

PVLDB 工件可用性：
源代码、数据和/或其他工件已发布于 https://github.com/bytezzz/CAKE。

1 引言

底层数据库算子通常有多种物理实现。例如，排序算子可以使用快速排序或堆排序对小批量数据进行排序。为了在这些方案中做出选择，有些数据库管理系统会直接采用最坏情况下最优的默认方案（例如，PostgreSQL [2] 始终使用快速排序进行内存排序），或者依赖简单的启发式方法（例如，DataFusion [3] 根据密度在切片和位迭代器之间切换）。

这两种策略都有明显的缺点。采用单一实现会忽视输入数据的特征，因此无法针对给定工作负载选择最合适的算法。相反，依赖简单的启发式方法需要手动调优，可能需要开发者和最终用户针对每个数据集分别进行调优，并且还需要持续维护。

以下文中 [24],将这些底层实现称为内核。几乎每个内核都有多种实现，

本作品采用知识共享署名-非商业性使用-禁止演绎 4.0 国际许可协议授权。请访问 https://creativecommons.org/licenses/by-nc-nd/4.0/ 查看此许可协议的副本。对于超出本许可协议授权范围的任何用途，请发送电子邮件至 info@vldb.org 获取许可。版权归所有者/作者所有。出版权授权给 VLDB基金会。VLDB 学报，第 19 卷，第 1 期，ISSN 2150-8097。doi:XX.XX/XXX.XX

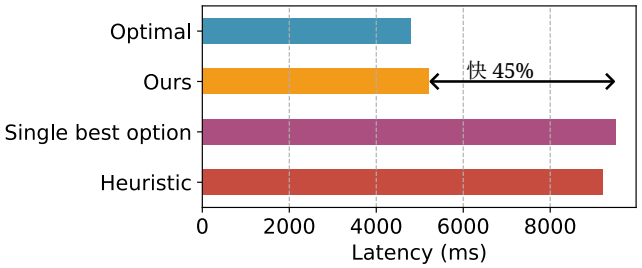


图1：使用不同内核选择算法时，IMDb 工作负载的延迟。为每个数据块选择合适的内核会对性能产生重大影响。详见第4节。

其中每种实现都可能适用于某种特定类型的输入数据。例如，有些内核在处理近乎有序数据时效率很高，而另一些则更适合稀疏数据。由于最优内核取决于输入数据的属性，因此对于查询中的每个数据块 [36]，正确的选择都可能不同。

选择正确的内核对性能至关重要。图1展示了在我们的 IMDb 工作负载上使用不同内核选择方法时的性能（详情见第4节）。值得注意的是，选择单一最佳静态选项与为每个数据块选择最优内核之间的性能差距接近 2×。

这引出了一个关键问题：我们如何构建数据库管理系统，使其能够为每个数据块选择正确的内核，同时又不会显著增加数据库管理系统开发人员的负担，也不要求用户针对自己的数据集手动调优系统？理想情况下，开发人员无需设计启发式方法，数据库管理系统也应能开箱即用地高效运行。

我们提出了一种新的强化学习（RL）技术，具体而言，是一种上下文多臂老虎机技术（CMAB） [75]（与标准设定略有不同，因为我们允许模型付出额外代价来观察未选中臂的奖励），以在线方式自动学习数据块与最优内核之间的映射。我们的系统 CAKE（反事实自适应内核执行）采用一种基于条件采样的简单而强大的算法来解决这一问题。与以往的学习型查询优化工作（例如 [41, 67, 78]）不同，CAKE 在算子粒度上运行，会针对查询中的每个数据块分别做出决策。要将强化学习应用于内核选择这类问题，我们面临两项关键挑战。

挑战 1：开销 大多数内核调用耗时数十到数百微秒。这意味着，要有效选择内核，强化学习智能体必须在微秒级运行；推理时间每增加一微秒，都会侵蚀优秀内核选择带来的收益。因此，绝大多数强化学习技术都无法胜任，例如使用神经网络的方法 [18, 44, 58]。CAKE 通过一种微秒级上下文老虎机解决了这一问题，该方法使用局部性加权

重采样来估算每个内核的延迟分布，并通过经验置信度检验确认最优选择。

挑战 2：探索/利用与样本效率所有强化学习技术都必须在探索新策略与利用现有知识之间取得平衡 [60]。强化学习的一个核心挑战是缺乏反事实：一旦采取某个动作并观察到奖励，我们就永远无法得知如果做出不同选择会得到何种奖励。在我们的场景中并非如此。由于单个查询由数千个数据块组成，且每次内核调用通常在几微秒内完成，反事实评估往往可以以很低的成本获得。例如，如果某个算子的两种内核实现分别需要 12 µs 和 6 µs，则同时执行两者即可获得完整反馈，总成本仅为 18µs。由于端到端查询执行时间处于毫秒量级，有选择地计算此类反事实所引入的开销可以忽略不计。尽管仍然需要探索，CAKE 可以策略性地决定何时收集反事实数据，从而使 CAKE 的样本效率显著高于其他方法。CAKE 的样本效率如此之高，以至于即使仅在少数几个查询的执行过程中，CAKE 也能学习到有效的内核选择策略。

我们将 CAKE 设计为可轻松集成到高性能数据库管理系统中。首先，我们的开发者接口（见第 3.1 节）仅要求数据库工程师定义每个内核实现，并提供一个特征提取函数。其次，由于 CAKE 有可能从系统处理的每个数据块中学习，其决策模型能够快速收敛。尽管这些决策模型的推理延迟仅为几微秒（通常在 1µs 以内），CAKE 仍可通过将已收敛的模型编译为**遗憾树** 来几乎完全消除这一开销；遗憾树是一种特殊的决策树，可最小化每次划分的遗憾。CAKE 之所以能够训练这类模型，是因为它收集反事实数据。遗憾树是小型结构（小于一个缓存行），其推理路径仅包含少量分支，在已缓存时通常不到 20 ns。

实验表明，CAKE 能够以较低开销学习准确的内核选择算法，并可将真实数据集上的端到端工作负载延迟最多降低 2 倍。我们还研究了 CAKE 的超参数敏感性、样本效率以及遗憾树优化。

本文作出以下贡献：

- 我们介绍了 CAKE 的核心算法，该算法旨在通过对经验加权样本使用经验置信区间或近似置信区间，在微秒级时间尺度上运行；同时，我们还证明了该算法的收敛性。
- 我们展示了如何将学习模型有效地编译为高效的静态遗憾树。
- 我们对 CAKE 进行了实验评估，结果表明，CAKE 最多可将总计延迟和尾延迟降低至原来的 1/2。

本文其余部分安排如下。我们在第2节介绍系统模型。在第3节概述 CAKE 的实现，包括 CAKE 收敛性的证明。在第4节给出实验结果。最后，我们在第5节讨论相关工作，并在第6节总结全文。

2 系统模型

本节将说明 CAKE 所采用的关键假设。

Morsel驱动并行 我们假设使用一种基于 Morsel驱动并行 [36] 的执行引擎。Morsel驱动并行是一种专为众核架构设计的查询执行框架。Morsel驱动系统将输入数据划分为较小的固定大小片段（例如，每个数据块包含 10,000 个元组）。固定在硬件核心上的工作线程从队列中获取一个数据块，并使该数据块经过完整的流水线。这种执行策略对缓存和寄存器十分友好，并支持弹性等高级特征。

内核选择由于每条流水线都由多个算子组成，数据块驱动型数据库系统必须为每个算子选择实现。按照 [24], , 我们将这些实现称为内核。特定算子的最优内核取决于数据块的数据分布。例如，平均情况下最优的内核，对于相当一部分数据块而言可能并非最优：向量化内核可能需要支付固定的设置成本，只有处理较重的数据块时才能摊销；而分支型标量内核可能擅长处理较轻的数据块，却会在处理较重的数据块时停滞。有些数据库管理系统对每个算子使用单一内核，另一些则采用简单的启发式方法，根据数据选择内核 [6, 53]。遗憾的是，这些简单的启发式方法有几个缺点：(1) 必须针对每个算子手动设计，并且在开发出新内核时需要更新；(2) 必须针对每个数据集进行调优，才能获得最优性能。我们认为，与其手动开发启发式方法，不如让**执行引擎学习以数据块为粒度选择内核**，根据每个数据块的局部特征调整实现，从而最大限度地缩短端到端查询时间。

学习型内核选择的要求学习如何在语义等价的内核中进行选择很困难，因为选择器必须比其所控制的内核开销更低、可靠性更高。首先，许多内核在微秒内即可完成，因此特征提取和推理必须保持在微秒级（最好低于1µs），否则预测器的开销将占据主导，抵消所有收益。其次，训练开销必须可以忽略不计：系统不能为了批量再训练而暂停查询处理，而必须在严格的 CPU 预算下在线学习，同时继续执行数据块。第三，决策质量必须足以改善端到端延迟：学习策略必须在延迟差距显著的区域内准确，而不能只是达到较高的总体分类准确率——换言之，模型必须在关键之处做出正确判断。

遗憾的是，现成的强化学习算法无法满足这些要求。例如，使用计算开销高的深度神经网络来建模策略 [66], 价值函数 [44], 或两者 [25,58], 的技术，固然能够达到较高的准确率，但开销实在过大 [61]。另一方面，多臂老虎机技术 [75]虽然足够轻量 [24, 54], , 却对特征与奖励之间的关系作出了过多假设 [38]。

机遇我们利用两项结构优势，使学习在这一场景中切实可行。首先，系统会产生大量数据：单个查询可能由数千个数据块组成，这意味着单个查询可以生成数千个训练实例。这与先前将机器学习应用于查询

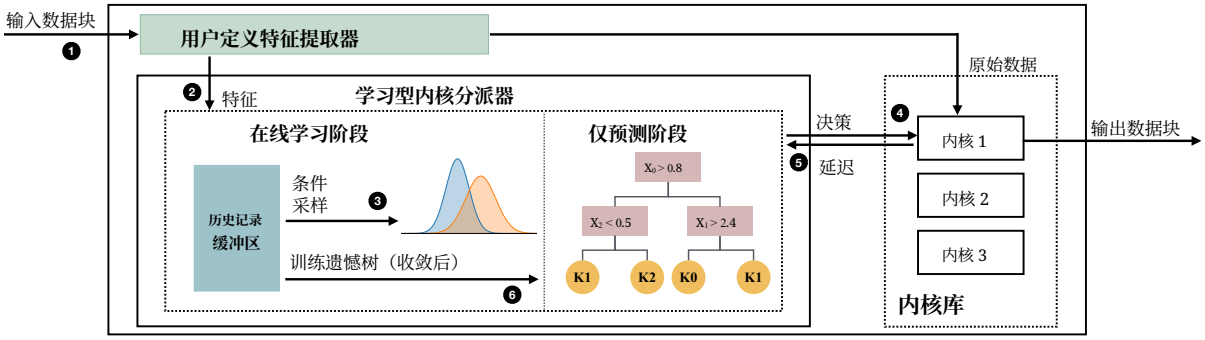


图 2：系统图

优化的工作形成对比，其中反馈稀疏（每个查询一次）[30, 41, 69]。其次，反事实反馈成本低廉且通常可得：通过偶尔在同一数据块上运行多个候选内核，我们可以同时观察到“正确”与“错误”的结果（例如，1毫秒对2毫秒），总成本与二者之和相当（例如，3毫秒），从而能够直接度量遗憾——而这正是大多数强化学习算法需要费力近似的东西 [60]。这与查询优化场景形成鲜明对比：在后者中，探索一个糟糕的计划可能浪费数分钟乃至数小时，使得反事实实际上无法获得。这两项结构优势也为设计新算法的方向提供了指导。

2.1 架构

我们的系统 CAKE 可以自动学习针对不同数据块应选择哪些内核。数据库开发人员只需提供可用内核列表并计算一些特征即可。CAKE 的架构如图 2 所示，具体说明如下。

特征提取 当一个❶输入数据块 到达时，会将其传入用户定义特征提取器。该用户定义组件会提取一个表示数据块数据分布的上下文向量（例如选择率、长度），并将其传递给学习型内核分派器。开发者需要确保特征提取的开销极低——我们将展示，利用采样等技术可以轻松实现这一点。如果已经存在启发式方法，开发者可以重新使用该启发式方法的输入作为特征。

在线学习阶段 最初，分派器在图 2 左侧所示的在线学习阶段运行。它维护一个历史缓冲区，其中包含过去的上下文及其观测到的延迟。对于❷提取出的特征，CAKE❸会使用精心设计的条件采样算法对这些历史证据进行加权，以决定运行哪个内核。它可以选择利用已知表现最佳的内核，也可以通过收集反事实数据（运行替代内核）进行探索。学习型内核分派器将决定执行哪个内核（或哪些内核）❹，然后在输入数据块上执行相应内核。由此产生的延迟❺会反馈到历史缓冲区中，以改进未来的决策。

仅预测阶段 学习模型收敛后，CAKE 会进入仅预测阶段，如图 2 右侧所示。复杂的概率模型❻会被编译为轻量级的遗憾树。该树根据输入特征进行简单的条件检查，将数据块路由至最优实现（K1、K2、等等）。此阶段完全消除了学习带来的计算开销，确保稳态执行期间的延迟可以忽略不计。

这些阶段共同使 CAKE 能够在局部证据稀缺时进行在线学习（在线学习阶段），并在决策稳定后以接近零的开销运行（仅预测阶段），从而实现逐数据块内核选择，在不改变面向开发者的执行接口的情况下改善端到端延迟。

3 CAKE

本节介绍 CAKE 的细节。第 3.1 节描述 CAKE 提供给数据库开发人员的接口，以及一个示例实现。第 3.2 节描述核心在线学习算法，以及收敛性证明和优化。最后，第 3.3 节解释 CAKE 如何将已收敛的模型“冻结”为推理开销几乎为零的遗憾树。

3.1 接口

代码清单 1 展示了使用 CAKE 的最小接口。代码清单 1 使用 Python 语法以提高可读性和简洁性；我们的原型系统用 Rust 实现。用户（数据库管理系统开发人员）需要提供候选内核实现列表 $K = \{k_1, \ldots, k_m\}$，以及一个特征提取函数，用于从输入数据块中提取特征 $q = \{q_1, \ldots, q_n\}$。此外，为确保稳健性，用户可以选择性地指定备用内核和超时阈值。如果在在线学习阶段任何内核的执行时间超过该阈值，CAKE 将终止学习过程，并在后续所有执行中默认使用备用内核。我们的设计目标是不给用户增加额外的使用负担，使其能够从基于启发式的内核选择无缝过渡到 CAKE，同时透明地受益于自适应切换带来的性能提升。

示例 为了实现一个内核，使用位图从数据块中提取特定行，并将选中的条目复制到输出缓冲区，开发者在 Arrow [1] Rust 库（DataFusion [35] 使用）中实现了两种备选复制策略：

(1) **索引迭代器**：枚举真位位置，并将对应元素收集到新的连续缓冲区中。(2) **切片迭代器**：将布尔掩码合并为连续真值段，然后逐段将这些范围从源数据批量复制到输出中。

直观来看，当位图较密集且包含较长的连续真值段时，切片迭代器更合适，因为它减少了代价高昂的 memcpy 调用次数。相反，当位图较稀疏时，索引迭代器效率更高，因为它避免了压缩位的开销。要将 CAKE 应用于这个问题，开发者可以定义一个特征提取器来计算位图选择性（即真位所占的比例），并提供这两种迭代器实现；随后，CAKE 会根据运行时信息动态进行选择。

Arrow库包含一种手工设计的启发式方法，根据位图 [3] 的稀疏度，在两者之间进行选择。遗憾的是，这种启发式方法只考虑位图的稀疏度，而不考虑游程的密度。因此，对于像 $[(110)_n]$（模式 110 重复 n 次）这样的位图，可能会选择切片迭代器；但对于像 $[0_n 1_k]$ when $k < n$ 这样的位图则不会。使用与该启发式方法相同的信息时，CAKE 会针对每个数据集学习稀疏度与最优内核之间的关系。我们在第4节通过实验展示，CAKE 学到的启发式方法明显更优。CAKE 甚至可以加入其他特征，例如样本中的平均游程长度，从而学习到更稳健的策略。

```
# 不同的内核实现
定义 kern_a (x: 数据块 , alpha : 浮点数 , beta: 整数) ->  数据
块 : 通过

定义 kern_b (x: 数据块 , alpha : 浮点数 , beta: 整数) ->  数据
块 : 通过

c 类 过滤器（AdaptiveKernel）：内核  = [内核_a,
  内核_b ]定义 提取_特征（cls，x：数据块）：返回  [
  x.选择率，x.长度 ]


# 执行自适应内核选择 out  =过滤器.自适应_执行（数据，
0.1，3）
```

列表 1：最小接口，使用 Python 语法以提高可读性和简洁性。

3.2 核心学习算法

在确立开发者接口和系统架构之后，我们现在转向驱动 CAKE 的决策逻辑。挑战在于如何在严格的延迟约束下，为每个数据块选择最优内核。以下小节将详细介绍我们的形式化模型、学习算法，以及支撑该方法的理论保证。

3.2.1 问题定义。形式上，一次查询执行被分解为一系列数据块 $\{u_t\}_{t=1}^{T}$。开发者提供一个特征（上下文）提取器 $x(\cdot)$，生成上下文向量 $\boldsymbol{q}_u = x(u)$。设 K 表示候选内核实现的集合。K 中的所有内核在语义上都是等价的（它们产生相同的逻辑结果），但其运行时间可能不同。对于上下文 $\boldsymbol{q}$ 和内核 $k \in \mathrm{K}$，我们将运行时间建模为随机变量 $Y_k(\boldsymbol{q})$。

系统按顺序观察上下文，即 $\boldsymbol{q}_t = x\ u_t \quad t = 1 \ldots T \quad t$ ( )，针对 „。在时刻做出决策之前，策略会观察一个历史数据集 D，其中包含先前见过的上下文、选定内核以及观测到的运行时间。在时刻 t，策略选择一个非空内核序列 $(k_{t,1}, k_{t,2}, \ldots, k_{t,m_t})$，其中 $k_{t,j} \in \mathrm{K}$ 且 $m_t \geq 1$。第一个内核 $k_{t,1}$ 被执行以生成该数据块的输出，并观测其运行时间。任何额外内核 $\{k_{t,j}\}_{j\geq 2}$均为可选的反事实测量：它们会带来额外的运行时开销，但仅提供计时观测（没有额外的语义收益）。

步骤 t 所产生的时间开销为

$$C_t \;=\; \sum_{j=1}^{m_t} Y_{k_{t,j}}(\boldsymbol{q}_t),$$

策略 π 在整个查询过程中的总成本为

$$C(\pi) \;=\; \sum_{t=1}^{T} C_t.$$

我们的目标是找到一个策略 π，使预期总成本最小（即最优策略）：

$$\pi = \arg\min_{\pi}\ \mathbb{E}[C(\pi)] \quad \text{s.t.} \quad m_t \geq 1,\ \forall t \in [T].$$

3.2.2 通过自举反事实进行学习。CAKE 的核心算法如算法 1 所示。该核心算法使用局部性加权证据和基于自举的检验，来决定何时利用单个内核，何时付出额外成本进行探索（即执行反事实）。其基本直觉如下：在特征空间中，**与当前数据块特征更接近的历史观测，对当前数据块的内核选择更具预测性**；而是否探索，则取决于**历史点能否对当前点的性能做出一致的评估**。因此，如果历史缓冲区中与当前数据块相关的样本不足，我们就应当探索。如果与该数据块相关的样本存在歧义，我们也应当探索。否则，我们就利用已有结果。

接下来，我们将精确描述该算法。对于当前到达的数据块，其上下文向量为 $\boldsymbol{q}$，我们首先根据每条记录到 $\boldsymbol{q}$ 的距离（经径向基函数核衰减）为其赋予权重，从而识别出最相关的历史执行记录（$\boldsymbol{x}_i$，$\boldsymbol{y}_i$）$\in$ D（第 1–2 行）。随后，我们对这些权重进行归一化，构成针对过往观测的抽样分布（第 3 行）。我们计算有效样本量 [32] N_{eff}，以概括历史记录在 $\boldsymbol{q}$ 附近提供了多少独立的局部支持；如果 N_{eff} 低于阈值 $N_{\min}$，算法便将该邻域视为支持不足，并立即进行探索（第 5 行）。接着，我们从上述局部分布中反复抽取自助法样本，构成多个样本“陪审团”集合（第 7 行），并利用这些样本中各内核的平均延迟，估算查询点处每个内核的延迟（第 9 行）。为达成共识，我们计算内核两两运行时间差的经验分布（第 14 行），并根据经验置信区间作出最终决策（第 16–17 行）。如果无法在给定置信水平下确定最优内核，我们就执行一次探索步骤：在当前数据块上运行所有内核，记录完整的延迟向量，并将其追加到 D，从而提升未来上下文的局部覆盖（第 20–22 行）。

这一设计确保 CAKE 能无缝地在执行低开销利用（当存储历史

算法 1： CAKE^{Bootstrap}

输入： 查询特征向量 $\boldsymbol{q}$；历史数据集 $\mathrm{D} = \{(\boldsymbol{x}_i,\ \boldsymbol{y}_i)\}_{i=1}^{n}$ 其中 $\boldsymbol{y}_i = \{y_{i,\ k}\}_{k\in\mathcal{K}}$ 是每个内核的延迟（越小越好）；显著性水平 α；带宽 h；最小有效样本量 $N_{\min}$；自助法重复次数 B。

输出： 选定内核 k_{sel}。

对于 $i \leftarrow 1$ **到** n，**执行**

  $w_i \leftarrow \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{q}\|_2^2}{h^2}\right)$

$\tilde{w}_i \leftarrow \frac{w_i}{\sum_{j=1}^{n} w_j}$　　// 离散分布 $P(i) = \tilde{w}_i$

$N_{\mathrm{eff}} \leftarrow \frac{1}{\sum_{i=1}^{n} \tilde{w}_i^2}$　　// 有效样本量

如果 $N_{\mathrm{eff}} > N_{\min}$，**则**

  // 在 $P(i) = \tilde{w}_i$ 下的自助法样本均值

  对于 $b \leftarrow 1$ **到** B，**执行**

    样本计数 $\boldsymbol{c}^{(b)} \sim$ 多项分布$(n, \tilde{\boldsymbol{w}})$

    对于每个 $k \in \mathcal{K}$，**执行**

      $\bar{y}_k^{(b)} \leftarrow \frac{1}{n}\sum_{i=1}^{n} c_i^{(b)}\, y_{i,k}$　　// 样本均值

  $k^\star \leftarrow$ 参数最小化$_{k\in\mathcal{K}}\ \sum_{i=1}^{n} \tilde{w}_i\, y_{i,k}$

  $\alpha_{\mathrm{adj}} \leftarrow \alpha/\max(1, |\mathrm{K}| - 1)$　　// 邦费罗尼校正

  $q_{\mathrm{lo}} \leftarrow \alpha_{\mathrm{adj}}/2$，置信度为 $\leftarrow$ **真**

  对每个 $k \in \mathcal{K} \setminus \{k^\star\}$ **执行**

    $\Delta^{(b)} \leftarrow \bar{y}_k^{(b)} - \bar{y}_k^{(b)}{}^{\star}$　　// $b = 1, \ldots, B$

    $L \leftarrow$ 分位数$(\{\Delta^{(b)}\}_{b=1}^{B}, q_{\mathrm{lo}})$

    如果 $L \le 0$**则**

      确信 $\leftarrow$ **假** // 置信区间与*0*重叠或偏向 k

  如果 确信**则**

    返回 $k^\star$　　// 利用

探索：

在当前查询上运行所有内核，以获得 $\boldsymbol{y} = \{y_k\}_{k\in\mathcal{K}}$

$\mathrm{D} \leftarrow \mathrm{D} \cup \{(\boldsymbol{q}, \boldsymbol{y})\}$

具有决定性时）与执行有针对性的探索（当局部证据不足或统计上存在歧义时）之间切换。

接下来，我们概述 CAKE$^{\mathrm{bootstrap}}$ 收敛性的证明。完整证明见在线附录 [74]。

定理 1. 选择一致性。随着观测历史 n 规模的增大，在方差有界噪声的假设下，*CAKE* 为查询小块 q 选择次优内核的概率线性趋近于零；在次高斯噪声的假设下，该概率呈指数级趋近于零。

证明概要。 选择内核的核心挑战在于理解即将到来的数据块的特征与先前观测（其中含有噪声）之间的关系。为了估算内核 k 在特定查询点 $\boldsymbol{q}$ 处的延迟，记作 $E[\boldsymbol{q}, k]$，CAKE 使用历史数据 D 的加权平均，其中权重会随与 $\boldsymbol{q}$ 的距离增大而衰减（径向基函数核）。为简洁起见，我们用数据块 $\boldsymbol{a}$ 表示其特征（上下文）向量为 $\boldsymbol{a}$ 的数据块。令 $\tilde{w}(\boldsymbol{a},\ \boldsymbol{b})$ 表示从 $\boldsymbol{a}$ 到 $\boldsymbol{b}$ 的归一化权重距离，并令 $f_k(\boldsymbol{a})$ 表示使用内核 k 执行数据块 $\boldsymbol{a}$ 的真实延迟（不含噪声），CAKE 计算如下：

$$E[\boldsymbol{q}, k] = \sum_{\boldsymbol{d}\in\mathcal{D}} \tilde{w}(\boldsymbol{q}, \boldsymbol{d}) \times L(\boldsymbol{d}, k) \tag{1}$$

其中，$L(\boldsymbol{d},\ k)$ 表示内核 k 处理数据块 $\boldsymbol{d}$ 时历史观测到的延迟（含噪声）。该估计的误差有两个来源：**偏差** 来自对附近但不完全等同于 $\boldsymbol{q}$ 的点进行平均，以及**方差** 来自延迟的噪声测量。

为了证明定理 1，我们需要以下假设：(1) 光滑性（Lipschitz）：我们假设 $f_k(\boldsymbol{x})$ 是光滑的。如果两个数据块 $\boldsymbol{a}$ 和 $\boldsymbol{b}$ 相似，我们假设存在一个 L，使得 $|f_k(a) - f_k(b)| \le L||a - b||$。(2) 分离性（间隔）：我们假设在点 $\boldsymbol{x}$ 处存在唯一的最佳内核 k^*，其性能以 $\Delta > 0$ 的间隔优于次优内核。这未必总是成立，但当两个内核并列时，我们并不在意 CAKE 选择哪一个。(3) 噪声：我们假设执行时间测量存在误差，但该误差的均值为 0，且方差有界 σ^2。

由于我们只需识别最优内核（即，CAKE 应像依赖真实延迟函数的预言机选择器一样，做出一致的选择 f），因此只需证明估计间隔中的总估计误差 $E[\boldsymbol{q}, k] - E[\boldsymbol{q}, k^\star]$（即来自 $k^\star$ 和 k 的偏差与方差之和）严格小于真实间隔 $\Delta(\boldsymbol{q})$，从而使间隔保持正确的符号，并保留最小值自变量。接下来，我们界定 CAKE 估计值的偏差和方差。

请注意，由于 CAKE 使用相邻数据块的加权平均来估计延迟，该估计量纳入了上下文不同于 $\boldsymbol{q}$ 的样本，因此会引入偏差。在我们的平滑性假设下，这种偏差取决于查询与从局部分布 [74], 中抽取的点之间的典型距离，形式上为 $\mathbb{E}_{X\sim P_h(\cdot|\boldsymbol{q})}[||X - \boldsymbol{q}||]$，其中 P_h 是局部抽样分布，h 是带宽。这与直觉相符：对过多相距较远的点进行平滑会增大偏差。我们在在线附录 [74] 中证明，$\mathbb{E}_{X\sim P_h(\cdot|\boldsymbol{q})}[||X - \boldsymbol{q}||]$ 随着 n 增大而收敛至与 h 成正比的常数；因此，估计量的偏差为 $O(h)$。直观而言，当我们将关注范围限制在足够小的邻域内时，邻近样本的加权平均能够在 $\boldsymbol{q}$ 处提供良好的近似。

遗憾的是，即使我们在 $\boldsymbol{q}$ 附近的一个小邻域内拥有数据，这些测量值仍然存在噪声。我们依赖有效样本量 N_{eff}，它统计了我们找到的相关历史数据点数量。由于我们假设 σ^2 的方差有界，因此加权平均的方差为 $\frac{\sigma^2}{N_{eff}(\boldsymbol{q})}$。

最后，要让 CAKE 选错内核，噪声必须足够大，才能克服真实间隔 Δ（减去邻居与 $\boldsymbol{q}$ 并非完全相同所带来的微小几何偏差）。

$$P(\text{wrong choice}) \approx P(\varepsilon_{i,k} - \varepsilon_{i,k^\star} > \Delta - \text{bias})$$

根据切比雪夫不等式，我们有

$$Prob(\text{wrong choice}) \le \frac{4\sigma^2}{(\Delta - bias)^2} \cdot \frac{1}{N_{eff}(\boldsymbol{q})}$$

算法 2： CAKE^{CLT}

输入： 查询特征向量 $\boldsymbol{q}$；历史数据集 $\mathrm{D} = \{(\boldsymbol{x}_i,\ \boldsymbol{y}_i)\}_{i=1}^{n}$ 其中 $\boldsymbol{y}_i = \{y_{i,\ k}\}_{k\in\mathcal{K}}$ 是每个内核的延迟；显著性水平 α；带宽 h；最小有效样本量 $N_{\min}$。

输出： 选定内核 k_{sel}。

计算 $\tilde{w}_i$ 和 N_{eff}，与算法 *1* 第 *1-4* 行相同
如果 $N_{\mathrm{eff}} > N_{\min}$ **则**
  对每个 $k \in \mathcal{K}$ **执行**
    $\hat{\mu}_k \leftarrow \sum_{i=1}^{n} \tilde{w}_i\, y_{i,\ k}$
    $\hat{\sigma}_k^2 \leftarrow \frac{1}{n}\left(\sum_{i=1}^{n} \tilde{w}_i\, y_{i,k}^2 - \hat{\mu}_k^2\right)$ // $\bar{y}_k \approx \mathrm{N}(\hat{\mu}_k, \hat{\sigma}_k^2)$
  $k^\star \leftarrow \arg\min_{k\in\mathcal{K}} \hat{\mu}_k$，置信 $\leftarrow$ **真**
  $\alpha_{\mathrm{adj}} \leftarrow \alpha / \max(1, |\mathrm{K}| - 1)$ // 邦费罗尼校正
  对每个 $k \in \mathcal{K} \setminus \{k^\star\}$ **执行**
    $z \leftarrow \frac{\hat{\mu}_k - \hat{\mu}_{k^\star}}{\sqrt{\hat{\sigma}_k^2 + \hat{\sigma}_{k^\star}^2}}$
    如果 $z \le z_{1-\alpha_{adj}}$ **则**
      有把握 $\leftarrow$ **错误** // 未能拒绝 $\mu_{k^\star} \ge \mu_k$
  如果 有把握 **则**
    返回 $k^\star$ // 利用
探索：与算法 *1* 第 *20–22* 行相同

随着历史记录大小的增长，N_{eff} 呈线性增长（严谨证明见 [74]），因此 CAKE 选错内核的概率会随着历史记录的增加而降低。在附录中，我们证明，假设噪声为次高斯噪声，这种收敛是指数级的。 □

优化：基于中心极限定理的置信界先前描述的算法CAKE$^{\mathrm{Bootstrap}}$通过抽取加权自助法重复样本并构建平均延迟差的经验置信界，显式近似局部抽样分布。遗憾的是，该采样过程可能代价高昂（我们在第4节中通过实验加以证明）。在此，我们提出一种优化算法CAKE$^{\mathrm{CLT}}$，它用闭式正态近似取代整个重采样阶段。对于每个内核 k，CAKE$^{\mathrm{CLT}}$ 以相同的加权均值 $\hat{\mu}_k = \sum_i \tilde{w}_i y_{i,k}$ 估计局部平均延迟；在我们的情形中，这恰好是称为Nadaraya-Watson核回归的点估计方法 [47]。但CAKE$^{\mathrm{CLT}}$ 并不生成自助法样本均值 $\{\bar{y}_k^{(b)}\}_{b=1}^{B}$，而是通过插入式二阶矩计算来估计（加权）样本均值的方差：

$$\hat{\sigma}_k^2 = \frac{1}{n}\left(\sum_i \tilde{w}_i y_{i,k}^2 - \hat{\mu}_k^2\right)$$

根据中心极限定理（CLT），我们可以将样本均值近似视为高斯分布，$\bar{y}_k \approx \mathrm{N}(\hat{\mu}_k,\ \hat{\sigma}_k^2)$。随后，算法选择暂定赢家 $k^\star = \arg\min_k \hat{\mu}_k$，并使用参数 z 检验对其进行认证：对于每个竞争者 $k \neq k^\star$，计算标准化差值 $z = (\hat{\mu}_k - \hat{\mu}_{k^\star}) / \sqrt{\hat{\sigma}_k^2 + \hat{\sigma}_{k^\star}^2}$，并将其与经过 Bonferroni 校正的临界值 $z_{1-\alpha_{\mathrm{adj}}}$ 进行比较，以检验 $k^\star$ 是否显著快于 k。如果所有成对检验均通过，算法则通过返回 $k^\star$ 进行利用；否则，它会回退到与 CAKE$^{\mathrm{Bootstrap}}$ 相同的探索步骤，在当前上下文中执行所有内核，并将 $(\boldsymbol{q},\ \boldsymbol{y})$ 追加到 D。

因此，实际差异体现在计算与统计两方面：CAKE$^{\mathrm{Bootstrap}}$ 是非参数的，其决策源自加权重采样所诱导的经验分布；而 CAKE$^{\mathrm{CLT}}$ 是参数的，它用高斯近似以及对每个竞争者的一次z检验取代重采样（从而去除自助法循环，但依赖于局部加权数据下中心极限定理近似的充分性）。在实践中，我们依靠 CAKE$^{\mathrm{CLT}}$进行决策，因为CAKE$^{\mathrm{Bootstrap}}$ 需要从离散分布中进行大量采样，这会显著增加推理延迟。令人惊讶的是，在我们的实验（第4节）中，我们发现CAKE^{CLT} 在准确率上往往优于 CAKE^{Bootstrap}（而不仅仅是推理速度）。这可能意味着高斯近似实际上是一个正则化项，并且内核在特征空间局部邻域中的行为事实上近似于高斯分布。

与 MDP 集成CAKE 与数据块驱动的并行机制（MDP）[36]能够自然集成。每个工作线程都维护自己的本地经验缓冲区，并且仅根据该本地状态做出内核选择决策，因此处理数据块时无需增加引擎现有调度之外的任何同步操作——也就是说，CAKE 不会在关键路径上增加同步开销。查询完成后（或系统空闲时），再同步各线程的经验缓冲区。此次合并在关键路径之外进行，实践中应能有效摊销，因为新记录主要在探索期间产生，而决策稳定后探索并不频繁（见下一节第 3.3 节）。

3.3 编译遗憾树

在执行初期，为了学习而承担一些推理开销（特征提取、模型打分、探索逻辑）是合理的，因为这些开销可以通过弥补较大的性能差距以及应对快速变化的判断来摊销。然而，一旦策略收敛且相关特征空间得到充分覆盖，每处理一个数据块都反复调用完整的推理流程就显得过于冗余：收敛后，CAKE 的内核选择实际上是确定性的。

一种诱人的简化方案是训练一个标准的多类分类器，让它在历史数据上以较高准确率预测最佳内核，从而“冻结”策略。然而，这一目标与我们的目标并不一致。准确率将所有错误一视同仁：将延迟相差 0.1 毫秒的两个内核混淆，与将延迟相差 10 毫秒的内核混淆，受到的惩罚相同。对于内核选择，重要的并非预测标签是否完全正确，而是预测错误时会额外耗费多少时间。因此，冻结策略应优先确保那些犯错代价高昂区域的预测正确；而当多个内核的表现几乎不相上下时，则可以容许一定的模糊性。

我们通过遗憾树来解决这一问题。这是一种代价感知决策树，[12]经过训练后可最小化经验遗憾，而非误分类率。这种方法借助反事实测量得以实现：对于一部分数据块中的每个训练数据块，其特征向量为 $\boldsymbol{x}_i$，观测到的延迟为 $y_{i,\ k}{}_{k\in\mathcal{K}}$，定义单臂遗憾

$$r_{i,k} = y_{i,k} - \min_{k'\in\mathcal{K}} y_{i,k'}$$

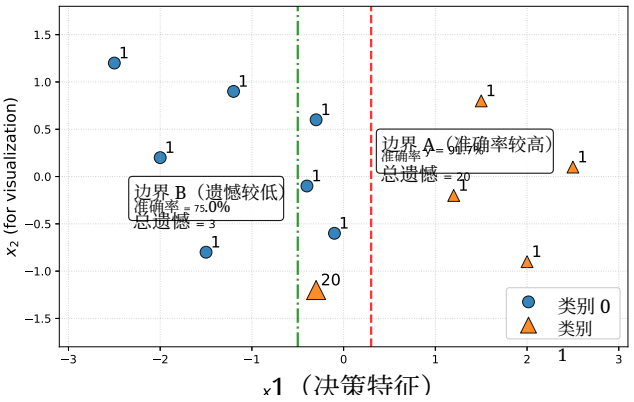


图 3：最小化遗憾与最大化准确率。每个点都标注了误分类的惩罚。

它量化了选择内核 k而非观测到的最佳替代方案所带来的时间惩罚。遗憾树通过最小化总遗憾 $\sum_i r_{i,\ T(x_i)}$，学习映射 T: $x \mapsto k$。具体而言，对于包含训练点子集 S 的任意节点，最优叶动作是该子集上累积遗憾最小的内核，$k_S = \arg\min_k \sum_{i \in S} r_{i,\ k}$，而该节点的损失为

$$\mathcal{L}(S) = \min_k \sum_{i \in S} r_{i,k}$$

选择划分时以最大化遗憾缩减为目标，即最小化 $\mathrm{L}(S_{\text{left}}) + \mathrm{L}(S_{\text{right}})$，从而得到由遗憾较大的位置而非标签边界明显的位置所决定的分区。

示例图 3 展示了标准决策树的边界选择与遗憾树的边界选择之间的区别。每个点旁边的数字表示该点被误分类时产生的遗憾。尽管边界 A 的分类准确率较高，但它会误分类一个在不同内核之间存在较大性能差距的点，导致端到端性能较差。相比之下，边界 B 的准确率较低，但产生的遗憾显著更小，因此整体性能更好。这种“编译后”的策略几乎没有运行时开销（只需少量比较，并直接跳转到选定内核），同时保留了端到端目标。

何时应该构建遗憾树？ 理想情况下，应在模型收敛、不再需要学习时构建遗憾树。遗憾的是，检测模型是否收敛并非易事，尽管目前已有多种实用技术可用 [4, 28, 50, 79]。

因此，我们将寻找最优收敛检测方法留待未来工作。我们在第4节通过实验表明，在处理完最初十几个查询后构建遗憾树通常就足够了。另一种可行的解决方案是监控推理时间。开发者可以设置推理时长的上限，然后在推理时间接近该上限时，或在较长时间内没有新的点被判定为“探索”时，构建遗憾树。

4 实验

为了全面了解 CAKE 提升现有执行引擎的能力，我们开展了一系列实验，以回答以下研究问题：

- CAKE 能否改善端到端查询时间（即，将学习开销纳入考虑）？
- CAKE 的开销如何？
- 中心极限定理近似的效果如何？
- CAKE 对超参数取值有多敏感？
- CAKE 的样本效率如何？
- 遗憾树优化能否在不影响性能的情况下降低推理成本？

接下来，我们将介绍实验设置以及 CAKE 原型的实现。

数据库集成与任务我们基于 arrow-rs [1], 这一广泛使用的列式数据处理库实现 CAKE。我们将 CAKE 集成到三种具有代表性的任务中（表 1）：过滤器迭代器选择（基于切片的迭代与基于点的迭代）、排序算法选择（快速排序与堆排序），以及嵌套谓词求值（部分求值与独立求值）。

基线 我们将 CAKE 与若干基线进行比较。首先，我们纳入每项任务的最佳单一算法（“SingleBest”）（即，如果只能选择一个内核时应选用的最佳内核）。其次，我们纳入一个先知式、零延迟的内核选择器（“Oracle”），它通过预先枚举所有选项得出（在实际中并不可行）。请注意，由于方差的影响，如果其他方法恰好遇到有利的方差，“Oracle”的表现可能略逊一筹。第三，我们纳入人工编写的基线（“启发式”），并针对我们的硬件平台进行了调优。第四，我们纳入一个非上下文多臂老虎机（MAB）算法（UCB），其实现与 Excalibur [24] 中的实现相同（见第 5 节）。我们还尝试了最先进的强化学习算法（例如，[58]）；然而，由于其推理开销超过内核执行时间，我们无法获得合理的结果。

超参数调优 我们通过树结构 Parzen 估计器 [64]（TPE）进行贝叶斯优化，并使用 Optuna [5] 实现，对 CAKE 和所有基线方法的超参数进行调优，目标是最小化总体遗憾。此优化在离线进行，不会带来运行时开销。

平台与实验方法所有实验均在配备 AMD Ryzen 7 Pro 8700GE 中央处理器和 128 GB 内存的平台上进行。除非另有说明，每项实验均重复五次。对于确定性算法（启发式和 SingleBest），我们绘制延迟中位数；对于随机方法，则展示平均延迟及自助法 95% 置信区间。

工作负载与数据集 我们在三个数据集上评估 CAKE：

- **IMDb：** 该数据集于 [37],引入，包含电影的元数据。其不均匀的数据分布非常适合在真实、倾斜的工作负载下评估 CAKE。
- **Stack：** 这是 Bao [41], 中 StackOverflow 基准测试的一个子集，包含来自 StackExchange 的问题和答案。
- **DSB**：DSB 数据集 [20], 是对 TPC- DS [48] 的修改版本，具有复杂的数据分布，用于对学习型和传统数据库系统进行基准测试。

由于每个数据集对应的原始基准查询并非都能测试我们的三项任务（表 1），我们采用了以下技术：

表 1：使用 CAKE 评估的任务

任务	选项	注释
过滤迭代器选择	索引迭代器	枚举真位位置，并将对应元素收集到新的连续缓冲区中。Oracle 适用于逐索引收集优于范围复制的稀疏谓词。
	切片迭代器	将布尔掩码合并为连续真值段，并对每一段，从源中批量复制这些区段到输出中。适用于包含较长真值段的稠密掩码，或大多数行都被保留的情况。
排序算法选择	快速排序	效率在很大程度上取决于输入数据的分布情况。当数据能够形成平衡分区时（例如随机排列或充分打乱的输入），表现良好。
	堆排序	运行时间基本不受输入数据分布的影响，因此无法从有利的输入分布中获益。
嵌套预测值求值（NPE）	并行	分别运行两个谓词以得到两个位图，再将两个位图求交以获得最终结果。它执行当许多行都通过内层谓词时，效果很好。
	顺序	先计算内层谓词，然后仅对有效行检查另一个谓词，以避免扫描未选中的数据——当基础选择率较低或外层谓词的计算成本较高时，这种方法效果很好。

SQLStorm [57] 使用大语言模型（LLMs）生成具有代表性的查询。对于每个数据集，我们持续生成查询，直到获得一组 500 条查询，其中每条都至少触发一项所考虑的任务。我们的数据集和查询可在第 1 页链接的 GitHub代码仓库中找到。

4.1 性能提升

在这里，我们衡量 CAKE 提升工作负载端到端性能的能力。

降低端到端延迟。图4展示了在我们的测试工作负载中使用不同内核选择算法时，查询延迟的中位数（P50）、P90 和总延迟。累积分布函数（CDF）见图5。

在 **IMDb** 数据集上，CAKE 在尾延迟和总延迟方面取得了最大幅度的改善，整体性能接近 Oracle。与启发式方法相比，CAKE 将 P90 从 44.20 ms 降至 21.77 ms（-50.7%）。CAKE 还将最坏情况下的查询延迟从 187.29 ms（启发式方法）降至 53.59 ms（-71.4%），同时总延迟仅比 Oracle 高 2.57%。在 **Stack** 数据集上，CAKE 持续改善中位数、尾延迟和总延迟，且表现与 Oracle 策略十分接近。CAKE 将 P90 从 144.61 ms 降至 103.18 ms（-28.7%）。相较于 Oracle，CAKE 在 P50/P90/总延迟上分别相差 5.27%/5.09%/4.35% 以内。在 **DSB** 数据集上，CAKE 主要优于 SingleBest 和 UCB，与启发式方法相比则结果相当。与 UCB 相比，CAKE 将总延迟降低了 20.3%。不过，与启发式方法相比，CAKE 将 P50 从 17.81 ms 略微提高至 19.99 ms，并使总延迟增加了 4.95%。尽管如此，CAKE 的总延迟仍仅比 Oracle 高 10.3%。

总体而言，我们观察到相较于 SingleBest 的显著改进，这有力地证明了动态切换内核能够实现比使用整体性能最佳的固定内核更好的性能。与人工定义的启发式方法相比，CAKE 在两个数据集上仍展现出显著优势，并在剩余的数据集上取得了相当的性能，同时无需开发者手动编写或维护启发式方法。

中心极限定理近似 在第 3 节中，我们介绍了 CAKE 的自助法变体（$\text{CAKE}^{\text{bootstrap}}$），以及经过优化的近似版本（$\text{CAKE}^{\text{CLT}}$）。在此，我们评估中心极限定理优化的效果。表 2 展示了 $\text{CAKE}^{\text{bootstrap}}$与 CAKE^{CLT}之间在性能、准确率和开销方面的差异。虽然 $\text{CAKE}^{\text{bootstrap}}$实现了较低的总内核执行时间（9.26 毫秒），但其基于自助法的推断会导致每次查询产生高达 140.22 毫秒的额外开销（这是由于生成每个自助法样本的成本所致）。将自助法抽样替换为中心极限定理近似后，总推理开销降至 0.24 毫秒，同时内核执行时间略有改善，降至 8.12 毫秒，最终端到端延迟为 8.55 毫秒。与 UCB（16.98 毫秒）和启发式基线（16.57 毫秒）相比，这大约实现了 2× 的加速。我们推测 CAKE^{CLT}相比 $\text{CAKE}^{\text{bootstrap}}$性能更好的原因可能有两个。首先，$\text{CAKE}^{\text{CLT}}$强制使每个样本的方差服从高斯分布（而 $\text{CAKE}^{\text{bootstrap}}$只有在经过多次迭代后才会收敛到高斯分布），因此起到了一种正则化的作用，降低了模型的方差。其次，从多项分布中进行采样需要构建并访问额外的数据结构，这可能会干扰预取机制并降低性能。

遗憾树由于 CAKE 可以从单个查询中获取许多样本，CAKE 的模型能够快速收敛。模型收敛后，我们可以将其编译为高效的遗憾树，从而进一步缩短推理时间。为了测试这一能力，我们循环运行了 IMDb 工作负载，并将每次循环称为一个轮次。表 3展示了遗憾树编译策略对延迟的影响。无编译时，推理延迟会持续增加，直到历史缓冲区在第 2 轮达到最大容量，随后稳定在每个查询 0.28 毫秒。相比之下，使用遗憾树可将推理时间降至每个查询 0.01 毫秒，同时内核执行时间略有缩短（这可能是因为 CAKE 的遗憾树使用了更少的寄存器/缓存）。这些结果表明，遗憾树能够显著降低推理开销，同时不损害整体性能，甚至可能提升整体性能。

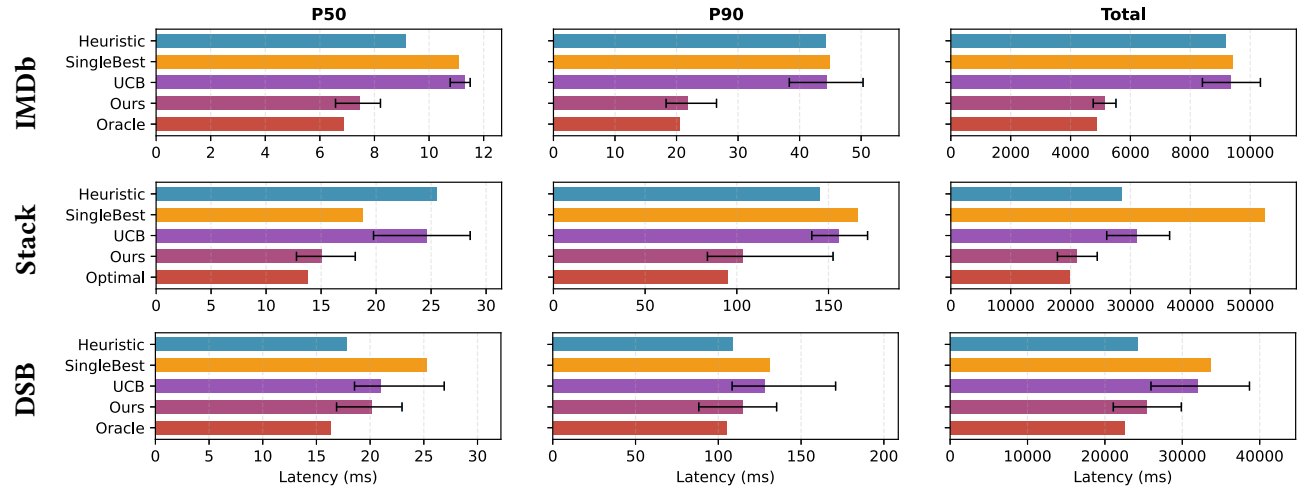


图4：各数据集的延迟对比

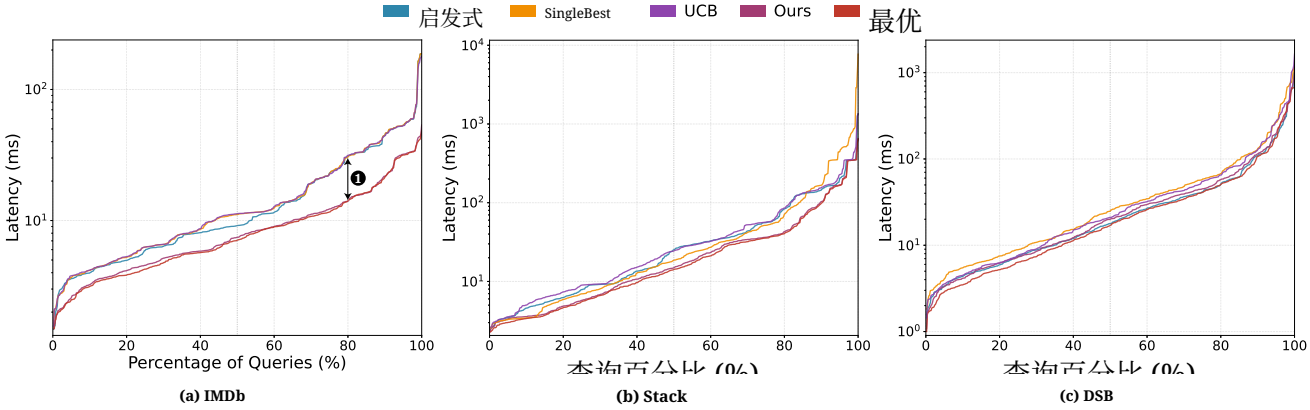


图 5：查询延迟的累积分布函数（CDF）。曲线之间的间距表示较低曲线在所有分位数上都具有严格更优的性能。❶ 展示了我们的方法相较于启发式基线所获得的性能提升，在 P80 处的表现几乎与最优预言机无法区分。

表 2：IMDb数据集上不同学习算法的平均延迟（每次查询）明细。括号中的数值表示 95%置信区间。

	CAKE$^{\text{bootstrap}}$	CAKE$^{\text{CLT}}$	UCB	启发式
反事实	**0.12** (0.09, 0.15)	**0.11** (0.08, 0.14)	**0**	**0**
推理	**140.22** (124.34, 157.30)	**0.24** (0.22, 0.27)	**0.04**(0.03, 0.04)	**0.01**
内核执行	**9.26** (8.58, 10.02)	**8.12**(7.55, 8.76)	**16.87**(15.04, 18.92)	**16.56**
端到端	**149.68**(133.19, 167.37)	**8.55**(7.95, 9.21)	**16.98**(15.15, 19.04)	**16.57**

4.2 量化开销

CAKE 在在线学习阶段产生的开销由三个部分构成：特征提取、条件采样（推理）以及生成反事实。CAKE 转入仅预测阶段（遗憾树）后，剩余的开销仅来自特征提取。

表4展示了 CAKE 在在线学习阶段产生的端到端延迟细分情况。在所有三个数据集上，总开销仅占端到端延迟的 5%。与优化内核带来的显著性能提升相比，这种权衡带来了明显的收益（见图4）。此外，相对较小的反事实开销表明，只需生成少量

表3：平均延迟（每次查询 ）在 IMDb 数据集上有编译和无编译时的分解。括号中的数值表示 95% 置信区间 。

	无编译				有编译（遗憾树）			
	反事实	推理	内核	端到端	反事实	推理	内核	端到端
第 1 轮	**0.11** (0.08, 0.14)	**0.22** (0.21, 0.25)	**8.05** (7.48, 8.69)	**8.37** (7.86, 9.11)	**0.11** (0.08, 0.14)	**0.22**(0.20, 0.24)	**8.07** (7.49, 8.71)	**8.4**(7.87, 9.14)
轮次 2	**0.01** (0.00, 0.01)	**0.28** (0.26, 0.31)	**8.11** (7.53, 8.76)	**8.42** (7.86, 9.15)	-	**0.01** (0.01, 0.04)	**7.94** (7.37, 8.57)	**7.95** (7.45, 8.68)
第3轮	**0.00**(0.00, 0.00)	**0.29** (0.27, 0.32)	**8.11** (7.53, 8.76)	**8.53** (7.87, 9.15)	-	**0.01** (0.01, 0.01)	**7.94** (7.37, 8.57)	**7.95** (7.44, 8.66)
第4轮	**0.00**(0.00, 0.00)	**0.29** (0.27, 0.32)	**8.12** (7.53, 8.76)	**8.53** (7.87, 9.16)	-	**0.01** (0.01, 0.01)	**7.94** (7.37, 8.58)	**7.95** (7.44, 8.67)
轮次 5	**0.00**(0.00, 0.00)	**0.29** (0.27, 0.32)	**8.11** (7.52, 8.75)	**8.53** (7.86, 9.15)	-	**0.01** (0.01, 0.01)	**7.94** (7.37, 8.58)	**7.95** (7.45, 8.67)

表4：平均延迟（每次查询）分解。对于每个数据集，CAKE 将约 95% 的时间用于执行内核。反事实执行和模型相关开销约占 5%（不包括遗憾树编译）。

	模型相关	内核执行	反事实
IMDb	3.76%（0.32 ms/q）	95.00%（8.12 ms/q）	1.24%（0.11 ms/q）
Stack	4.93%（1.05 ms/q）	94.23%（20.09 ms/q）	0.84%（0.18 毫秒/查询）
DSB	5.45%（1.76 毫秒/查询）	94.12%（30.48 毫秒/查询）	0.42%（0.14 毫秒/查询）

反事实即可提升模型性能，从而验证了我们算法所依据的假设。

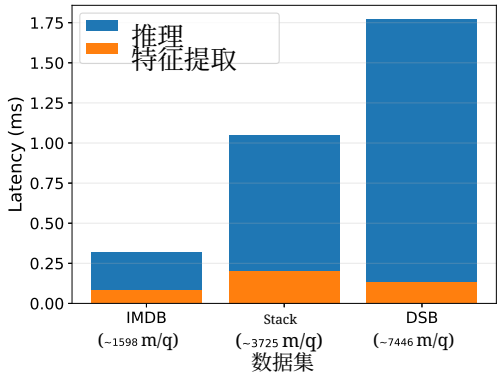


图6：按查询计的模型相关开销，按推理时间和特征提取时间进行分解。每个工作负载名称下列出了每条查询的数据块数量（m/q）。每条查询包含更多数据块的数据集开销更高，因为CAKE被调用的次数更多。模型推理占模型相关开销的绝大部分。

图6展示了每个数据集的平均模型相关开销分解。从结果中可以得出两点观察。首先，CAKE的总体开销在按查询计算时可以忽略不计。对于IMDb数据集，模型平均每条查询被调用来处理1,598个数据块，但由此产生的开销仅为每条查询0.28 ms。这相当于每个数据块平均175 ns的模型相关延迟，对端到端延迟的影响微乎其微。其次，推理时间

每个查询的开销随着每个查询的数据块数量增加而几乎线性增长，而非呈二次增长。简单的最坏情况分析可能会得出二次时间复杂度：推理时间随历史缓冲区中保留的数据量线性增长，并且在采样率 p 恒定的假设下，可得复杂度为 $O(p \cdot n \cdot n) = O(n^2)$，其中 n 表示待处理的数据块数量。然而，我们的实证结果表明，实际生成的反事实数量非常少，恰如图4所示的模式。因此，实践中观察到的推理开销更接近 $O(n)$。

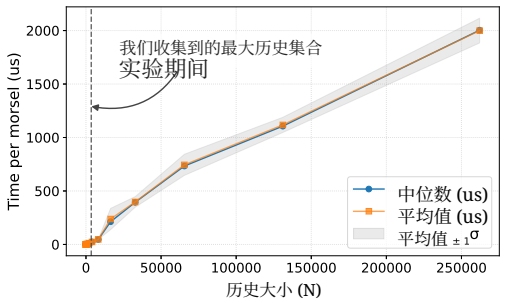


图7：随着历史大小增加，推理开销的增长情况。如果历史缓冲区无限增长，推理延迟将会增加。垂直线表示我们在实验中观察到的最大历史缓冲区大小。

在图7中，我们分析了 CAKE 的推理时间如何随历史缓冲区大小变化。当存储在历史缓冲区中的数据量增加时，执行一次条件采样操作所需的时间也会线性增长。然而，对于大小合理的历史缓冲区，推理延迟仍然较低。因此，与其他流行的强化学习算法通常达到毫秒级的推理延迟相比，CAKE 特别适合时间受限的场景 [61]。在强化学习应用于数据库系统的其他场景中，维持有界的历史缓冲区同样很常见 [41]。

4.3 样本效率

为评估样本效率，我们在将 CAKE 的学习限制为初始预算 n 个查询的同时测量遗憾，随后立即构建遗憾树，并报告在完整数据集上产生的遗憾。结果绘制于图 8。整个数据集上的总遗憾在第五个与第六个查询之间急剧下降，此后基本保持稳定。当 CAKE 仅从六个查询中学习时，它在此窗口内大约处理 8,000-10,000 个数据块。因此，CAKE 不会遭遇其他基于强化学习的数据库优化技术（例如 [42]）中普遍存在的同样样本效率问题。

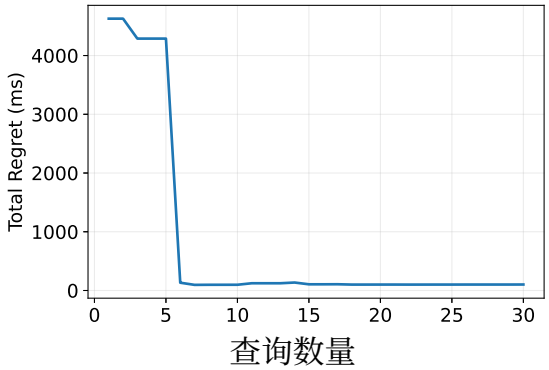


图 8：在完整的 IMDb数据集上，将学习限制在前 n 个查询内时产生的遗憾

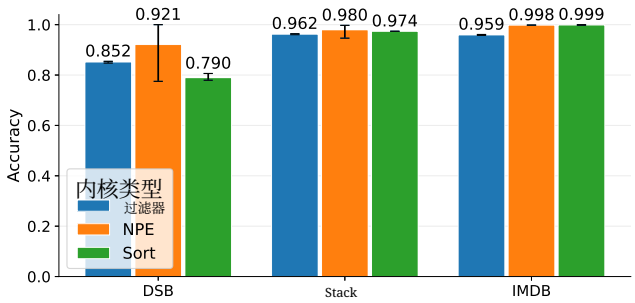


图 9：CAKE 在不同内核上的准确率

4.4 内核分析

为理解 CAKE 的端到端收益源自何处，我们在表 1 中三项任务的单个内核选择粒度上考察决策质量。图 9 汇总了初始运行期间各数据集上的内核选择准确率（与准确率为 100% 的 Oracle 相比）。在 IMDb 上，CAKE 的过滤器选择准确率达到 95.9%，NPE 与排序选择则近乎完美（分别为 99.8% 和 99.9%）。在 Stack 上，准确率同样很高（过滤器/NPE/排序分别为 96.2%/98.0%/97.4%）。DSB 更具挑战性：尽管 NPE 仍保持准确（92.1%），过滤器与排序选择的准确率分别降至 85.2% 和 79.0%，这与在 DSB 上相对于手工编码启发式所观察到的较小端到端收益相一致。这可能归因于 DSB 被设计为对学习模型具有对抗性 [20]。

表 5 进一步刻画了错误分布。对于 NPE，误预测很少：在 120,553 次决策中，仅有 85 个并行数据块被判定为顺序数据块，446 个顺序数据块被判定为并行数据块。对于过滤器选择，错误分布不对称：切片被误分类为索引的次数为 57,322 次，而反向错误仅发生 9,957 次；这与明显的类别不平衡相吻合，因为索引在 589,414 个实例中是最优选择，而切片在 149,577 个实例中是最优选择。对于排序，堆的情况并不常见（43,898 个实例中有 2,574 个），大多数错误都对应于将这些少见的堆实例与占主导地位的快速排序情况混淆。

最后需要指出，虽然准确率提供了一种便捷且与任务无关的概括指标，但它并非 CAKE 的优化目标：CAKE 旨在特征空间中遗憾较高的区域实现正确预测，而不是将所有误分类一视同仁。

4.5 超参数

我们通过实验评估了 CAKE 对各种超参数的敏感性。表6展示了在不同优化目标下调整超参数所得到的平均查询延迟。除了两个确定性基线（启发式和 SingleBest）外，我们还报告了 CAKE 在两种超参数优化方案下的表现：（1）逐数据集贝叶斯优化，即针对每个数据集独立调整超参数；（2）全数据集贝叶斯优化，即在 IMDb 、 Stack 和 DSB 上进行一次联合优化。我们观察到，在所有数据集上联合优化超参数所得的表现与在各个数据集上分别优化的结果相当，这表明针对每个数据集调整 CAKE 的超参数益处不大（也就是说，不同数据集可以使用相同的超参数）。

接下来，我们通过在保持其他参数不变的同时，对各个超参数在其调优值附近进行扫描，评估模型的敏感性。图 10 展示了三个关键超参数（alpha、带宽和 $n_{\min}$）的代表性扫描下的总遗憾。在这些局部范围内，CAKE 的遗憾曲线基本平坦，没有明显的性能下降，这表明 CAKE 无需依赖经过精细调优的超参数，也能做出良好的决策。相比之下，两种确定性基线的遗憾始终较高，因为它们不会根据当前数据集调整核选择。

总体而言，这些结果表明，CAKE 的收益在合理范围内的超参数变化下保持稳定，而且经过联合优化的单一配置能够很好地迁移到不同工作负载。

5 相关工作

CAKE 专注于适应不同的数据集，同时也有研究致力于使执行引擎能够适应硬件变化。据我们所知，与本文最相近的工作是 Excalibur [24] （它建立在 Vectorwise [54] 的微适应性基础上）。该工作的作者针对以 VOILA [23] 编写的内核，在广泛的编译选项中进行搜索，并采用两种策略：一种通过启发式方法生成候选变体；另一种先执行蒙特卡洛树搜索来生成变体，再使用多臂老虎机从生成的变体中进行选择。我们在第4节中将 CAKE 与 Excalibur 的多臂老虎机算法进行比较，但由于 CAKE 的问题具有依赖上下文的特性，因此无法进行直接比较。在 HAWK [13], 中，作者提出为同一个输入内核生成许多机器码变体，然后学习哪种变体在用户的硬件上表现最佳。在 [11], 中，作者提出了一种基于 MLIR 和编译的方法。所有这些方法都专注于将同一种逻辑内核实现编译为不同的机器码，而 CAKE 则专注于在不同的逻辑内核实现之间进行选择。关键区别在于，CAKE 的决策需要上下文，而基于硬件的优化决策不需要（因为硬件不会随查询而变化）。

自适应查询处理（例如， [7, 29, 62, 63, 68, 72]）也与此相关，但往往着重于修复查询计划中的结构性问题（例如，糟糕的连接顺序），而非底层内核实现。查询重新优化 [8, 19, 40, 52] 同样旨在通过在执行过程中重新优化计划来修复结构性计划问题。

数据库内部机制中，强化学习最广泛的应用很可能是用于查询优化（例如， [9, 17, 30, 39, 41, 42, 45, 65, 67, 69, 70, 76– 78]）。由于大多数学习型查询优化器都侧重于连接排序，

表 5：IMDb数据集上各任务的混淆矩阵。

(a) 查询计划

实际值 \ Pred	并行	Seq.	总计
并行	30,187	85	30,272
Seq.	446	89,835	90,281
总计	30,633	89,920	120,553

(b) 过滤器

实际值 \ Pred	切片	索引	总计
切片	92,255	57,322	149,577
索引	9,957	579,457	589,414
总计	102,212	636,779	738,991

(c) 排序

实际值 \ Pred	快速排序	Heap	总计
快速排序	39,350	1,974	41,324
Heap	1,716	858	2,574
总计	41,066	2,832	43,898

表6：不同超参数优化程度下的平均查询延迟。括号中的数值表示95%置信区间。为所有数据集调优同一组超参数（“全数据集贝叶斯优化”）与针对每个数据集分别调优超参数（“逐数据集贝叶斯优化”）的性能相近。

	IMDb	Stack	DSB
启发式	16.64 毫秒	36.22 毫秒	30.31 毫秒
SingleBest	17.15 毫秒	84.36 毫秒	49.22 毫秒
全数据集贝叶斯优化	8.47 毫秒（7.86，9.13）	21.51 毫秒（17.97，25.95）	29.85 毫秒（24.18，36.27）
逐数据集贝叶斯优化	8.56 毫秒 (7.55, 9.22)	21.32 毫秒 (17.80, 25.76)	32.38 毫秒 (26.38, 39.18)

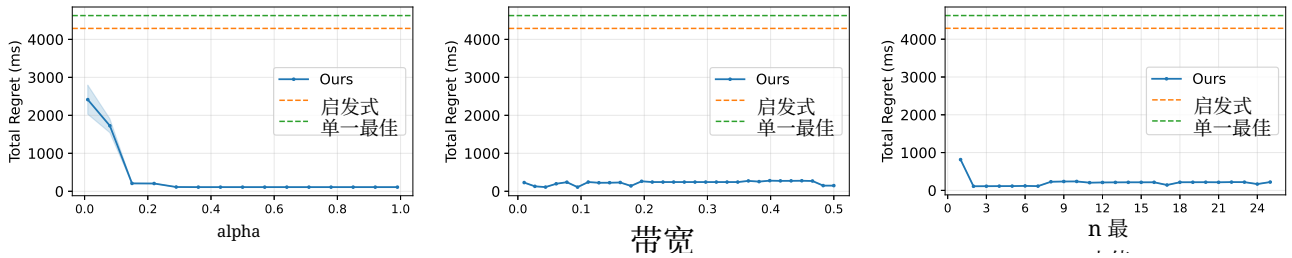


图 10：超参数敏感性

算子选择和访问路径选择，因此 CAKE 与这些工作大多是正交的。学习型查询优化器通常会在每个查询开始执行前做出所有决策，而 CAKE 则会在查询执行过程中主动学习。CAKE 有两个关键特性，使其比大多数学习型查询优化器更容易使用：（1）CAKE 不会做出后果极其严重的决策，因为最佳内核与最差内核之间的差距通常只有几微秒（而最佳连接顺序与最差连接顺序之间的差异可能意味着数天的计算量 [37]）；（2）CAKE 获取训练数据的速度远快于学习型查询优化器（也就是说，学习型查询优化器每个查询只能获取一个样本，而 CAKE 每个数据块就能获取一个样本），从而弥补样本效率方面的问题。将学习型查询优化器与 CAKE 相结合的研究留待未来开展。

围绕构建自适应系统已有大量研究。MLOS [34] 为开发者提供工具，使其能够轻松地对程序进行基准测试并搜索其配置空间，但它侧重于编译时调优（类似于基于性能剖析的优化 [16]）。与 CAKE 类似，SmartChoices [15], 也允许开发者轻松地将机器学习驱动的决策集成到应用程序中，但其目标延迟范围为 10–100 毫秒，并使用深度学习、网络附加推理和历史缓冲区。

CAKE 遵循一种通用理念，即构建名为“机器编程”的自调优、适应性系统 [22]。在数据管理领域，索引结构 [33], 、基数估计 [31], 、查询调度 [55, 71]、工作负载管理 [43, 56, 59], 、实体匹配 [46], 和事务调度 [51] 也已有类似研究。

6 结论

本文介绍了 CAKE，这是一种利用亚毫秒级强化学习技术构建自适应查询执行引擎的方法。与此前将强化学习应用于数据库系统的工作不同，CAKE 会在一次查询中 进行学习，有可能从一次执行中收集数千个样本。实验结果表明，CAKE 能够快速学得近乎最优的策略，在我们的基准测试中将尾延迟降低了最多 2 倍。未来，我们计划研究如何将 CAKE 应用于数据库管理系统的其他组成部分，例如自适应查询规划 [7, 68],横向信息传递 [10, 14, 73], 渐进式索引内核 [26], 和数据布局 [21, 27, 49]。

致谢

本研究得到了 Merly AI 的资助以及 Google 机器学习与系统青年教师奖的支持。

参考文献

[1] [n.d.]。Apache Arrow。https://arrow.apache.org/ [2] 2024。PostgreSQL 数据库, http://www.postgresql.org/。http://www.postgresql. org/ [3] 2026。Arrow 过滤逻辑。https://github.com/apache/arrow-rs/blob/5695bb33746735dde01b80ee9e7e6bcccbd7bdd4/arrow-select/src/filter.rs#L356[4] Rishabh Agarwal、Max Schwarzer、Pablo Samuel Castro、Aaron Courville 和 Marc G Bellemare。2021。统计悬崖边缘的深度强化学习（NeurIPS '21）。在线。[5] Takuya Akiba、Shotaro Sano、Toshihiko Yanase、Takeru Ohta 和 Masanori Koyama。2019。Optuna：新一代超参数优化框架。载于第 25 届 ACM SIGKDD 知识发现与数据挖掘国际会议论文集（KDD '19）。美国计算机协会，纽约，美国，2623 2631 10 1145 3292500 3330701 – 。[6] [ ] https://doi.org/ . / . Apache。n.d.。arrow::compute - Rust。https://docs.rs/arrow/latest/arrow/ compute/index.html https://docs.rs/arrow/latest/arrow/compute/index.html。[7] Ron Avnur 和 Joseph M. Hellerstein。2000。Eddies：持续自适应查询处理。载于 2000 年 ACM SIGMOD 数据管理国际会议论文集（SIGMOD '00）。ACM，纽约，美国，261–272。https://doi.org/10.1145/342009.335420 活动地点：达拉斯，德克萨斯州，美国。[8] Shivnath Babu、Pedro Bizarro 和 David DeWitt。2005。借助 Rio 主动重新优化。载于 2005 年 ACM SIGMOD 数据管理国际会议论文集（SIGMOD '05）。美国计算机协会，纽约，美国，936 938 10 1145 1066157 1066294 – 。https://doi.org/ . / . Henriette Behr 、 Volker Markl 和 Zoi Kaoudi。. 学习真正重要的内容：一种用于基于机器学习的查询优化的学习排序方法。载于《商业、技术与网络数据库系统 2023》（BTW '23），Birgitta König-Ries、Stefanie Scherzinger、Wolfgang Lehner 和 Gottfried Vossen（编）。Gesellschaft für Informatik e.V. https://doi.org/ 10 18420 2023 25 . /BTW - Altan Birler、Alfons [9] [10] 2023 2024 Kemper 和 Thomas Neumann。. 采用 Diamond Hardened Joins 实现稳健连接处理。VLDB基金会会刊 17, 11（2024 年 7 月），3215–3228。https://doi.org/ 10 14778 3681954 3681995 . / . Paul Blockhaus、Gabriel Campero [11] Durand、David Broneske 和 Gunter Saake。2023。迈向完全自优化查询引擎的未来。载于第 34 届 GI 数据库基础研讨会（GI '23）。CEUR 研讨会论文集，德国希尔绍。[12] Leo Breiman、Jerome Friedman、Richard Olshen 和 Charles Stone。1998。分类与回归树。Chapman & Hall，博卡拉顿。[13] Sebastian Breß、Bastian Köcher、Henning Funke、Steffen Zeuch、Tilmann Rabl 和 Volker Markl。2018。为异构处理器上的高效查询执行生成定制代码。VLDB期刊 27, 6（2018 年 12 月），797–822。https://doi.org/10.1007/s00778-018-0512-y[14] Daniela Böhm、Georg Gottlob、Matthias Lanzinger、Davide Longo、Cem Okulmus、Reinhard Pichler 和 Alexander Selzer。2025。选择性使用 Yannakakis 算法提升查询性能：机器学习来助力。https://doi.org/10.48550/arXiv.2502.20233 arXiv:2502.20233 [cs].[15] Victor Carbune、Thierry Coppey、Alexander Daryin、Thomas Deselaers、Nikhil Sarda 和 Jay Yagnik。2019。SmartChoices：融合编程与机器学习。https://doi.org/ 10 48550 1810 00619 1810 00619 . /arXiv. . arXiv: . [ ] cs . [16] Pohua P. Chang、Scott A. Mahlke 和 Wen-Mei W. Hwu。1991。利用性能剖析信息辅助经典代码优化。Software: Practice and Experience 21, 12（1991），1301–1321。https://doi.org/10.1002/spe.4380211204 _电子印本：https://onlinelibrary.wiley.com/doi/pdf/10.1002/spe.4380211204。[17] Xu Chen、Haitian Chen、Zibo Liang、Shuncheng Liu、Jinghong Wang、Kai Zeng、Han Su 和 Kai Zheng。2023。LEON：机器学习辅助查询优化的新框架。VLDB基金会会刊 16, 9（2023 年 5 月），2261–2273。https://doi.org/ 10.14778/3598581.3598597[18] Mark Collier 和 Hector Urdiales Llorens。2018。深度上下文多臂老虎机。arXiv:1807.09809 [cs, stat]（2018 年 7 月）。http://arxiv.org/abs/1807.09809[19] Guilherme Damasio、Vincent Corvinelli、Parke Godfrey、Piotr Mierzejewski、Alex Mihaylov、Jaroslaw Szlichta 和 Calisto Zuzarte。2019。面向查询工作负载重新优化的引导式自动学习。VLDB 学报 12, 12 2019 2010 2021 10 14778 3352063 3352120（8 月），– 。[20] https://doi.org/ . / . Bailu Ding、Surajit Chaudhuri、Johannes Gehrke 和 Vivek Narasayya。2021。DSB：面向工作负载驱动型和传统数据库系统的决策支持基准。VLDB 学报 14, 13（2021 年 9 月），3376–3388。https://doi.org/ 10 14778 3484224 3484234 . / . Jialin Ding、Vikram Nathan、[21] Mohammad Alizadeh 和 Tim Kraska。2020。Tsunami：面向相关数据和倾斜工作负载的学习型多维索引。VLDB 学报 14, 2（2020 年 10 月），74–86。https://doi.org/ 10 14778 3425879 3425880 . / . Justin Gottschlich、Armando [22] Solar-Lezama、Nesime Tatbul、Michael Carbin、Martin Rinard、Regina Barzilay、Saman Amarasinghe、Joshua B. Tenenbaum 和 Tim Mattson。2018。机器编程的三大支柱。载于第 2 届 ACM SIGPLAN 机器学习与编程语言国际研讨会论文集（MAPL 2018）。美国计算机协会，美国宾夕法尼亚州费城，69 – 80 10 1145 3211346 3211355 . [23] 2021 https://doi.org/ . / . Tim Gubner 与 Peter Boncz. . 使用 VOILA 绘制查询执行的设计空间. VLDB基金会会刊 14, 6 (2021年2月), 1067–1079. https://doi.org/10.14778/3447689.3447709[24] Tim Gubner 与 Peter Boncz. 2022. Excalibur：基于 VOILA 的自适应细粒度即时编译查询执行虚拟机. VLDB 学报 16, 4 (2022年12月), 829–841. https://doi.org/10.14778/3574245. 3574266[25] Runsheng Benson Guo 与 Khuzaima Daudjee. 2020. 基于深度强化学习的连接查询优化中的研究挑战. 见第三届人工智能数据管理技术应用国际研讨会论文集 (aiDM '20). 美国计算机协会, 波特兰, 俄勒冈, 1 6 10 1145 3401071 3401657 – . https://doi.org/ . / . [26] Stratos Idreos、Martin L. Kersten 与 Stefan Manegold. 2007. 数据库裂分 (CIDR '07). 阿西洛马, 加利福尼亚. [27] M. Ivanova、M. L. Kersten 与 N. Nes. 2008. 面向科学数据库的自适应分段. 见第24届国际数据工程会议论文集 (ICDE '08). 1412–1414. https://doi.org/10.1109/ICDE.2008.4497573[28] Scott M Jordan、Yash Chandak、Daniel Cohen、Mengxue Zhang 与 Philip S Thomas. 2020. 评估强化学习算法的性能. 见第 37 届国际机器学习会议论文集 (ICML '20). 维也纳 , 奥地利 . [29] Tomer Kaftan 、Magdalena Balazinska、Alvin Cheung 与 Johannes Gehrke. 2018. Cuttlefish：一种用于自适应查询处理的轻量级原语. arXiv 预印本 (2月 2018 1802 09180). https://arxiv.org/abs/ . [30] Amin Kamali、Verena Kantere、Calisto Zuzarte 与 Vincent Corvinelli. 2024. Roq：基于风险感知学习成本模型的稳健查询优化. (2024). https://doi.org/10.48550/ARXIV.2401.15210[31] Andreas Kipf、Thomas Kipf、Bernhard Radke、Viktor Leis、Peter Boncz 与 Alfons Kemper. 2019. 学习型基数：利用深度学习估计相关连接. 见第9届创新数据系统研究双年会 (CIDR '*19* 1809 00677). http://arxiv.org/abs/ . Leslie Kish. . 抽样调查. [32] 1965 John Wiley & Sons, Inc., 纽约 . [33] Tim Kraska 、 Alex Beutel 、 Ed H. Chi 、Jeffrey Dean 与 Neoklis Polyzotis. 2018. 学习型索引结构的论证. 见2018年国际数据管理会议论文集 (SIGMOD '18). ACM, 纽约, NY, 美国. https://doi.org/10.1145/3183713.3196909[34] Brian Kroth、Sergiy Matusevych、Rana Alotaibi、Yiwen Zhu、Anja Gruenheid 与 Yuanyuan Tian. 2024. MLOS 实践：弥合云中实验与自动调优之间的差距. VLDB基金会会刊 17, 12 (2024年8月), 4269–4272. https://doi.org/10.14778/3685800.3685852[35] Andrew Lamb、Yijie Shen、Daniël Heres、Jayjeet Chakraborty、Mehmet Ozan Kabak、Liang-Chi Hsieh 与 Chao Sun. 2024. Apache Arrow DataFusion：快速、可嵌入、模块化的分析型查询引擎. 见2024年国际数据管理会议配套论文集 (SIGMOD '24). 美国计算机协会, 纽约, NY, 美国, 5–17. https://doi.org/10.1145/ 3626246.3653368[36] Viktor Leis、Peter Boncz、Alfons Kemper 与 Thomas Neumann. 2014. 数据块驱动的并行：面向众核时代的 NUMA感知查询评估框架. 见2014年ACM SIGMOD 数据管理国际会议论文集 (SIGMOD '14). ACM, 斯诺伯德 犹他 美国, 743–754. https: // doi.org/10.1145/2588555.2610507[37] Viktor Leis、Andrey Gubichev、Atanas Mirchev、Peter Boncz、Alfons Kemper 与 Thomas Neumann. 2015. 查询优化器究竟有多好？ PVLDB 9, 3 (2015 204 215 10 14778 2850583 2850594), – . https://doi.org/ . / . Lihong Li 、 Wei Chu 、 John Langford 与 Robert E. [38] Schapire. 2010. 一种面向个性化新闻文章推荐的上下文赌博机方法. 见第19届万维网国际会议论文集. 661–670. https://doi.org/ 10.1145/1772690.1772758 arXiv:1003.0146 [cs]. [39] Hanwen Liu、Qihan Zhang、Ryan Marcus 与 Ibrahim Sabek. 2026. SEFRQO：一种自演进微调的基于RAG的查询优化器. ACM数据管理会刊 3, 6 2026 10 1145 3769826 (5月). [40] 2016 https://doi.org/ . / Mengmeng Liu、Zachary G. Ives 与 Boon Thau Loo. . 实现增量式查询再优化. 见2016年国际数据管理会议论文集 (SIGMOD '16). 美国计算机协会, 纽约, NY, 美国, 1705 1720 10 1145 2882903 2915212 – . [41] https://doi.org/ . / . Ryan Marcus、Parimarjan Negi、Hongzi Mao、Nesime Tatbul、Mohammad Alizadeh 与 Tim Kraska. 2021. Bao：使学习型查询优化切实可行. 见2021年国际数据管理会议论文集 (SIGMOD '21). 中国. https://doi.org/10.1145/3448016.3452838 奖项：'最佳论文奖'.[42] Ryan Marcus、Parimarjan Negi、Hongzi Mao、Chi Zhang、Mohammad Alizadeh、Tim Kraska、Olga Papaemmanouil 与 Nesime Tatbul. 2019. Neo：一种学习型查询优化器. PVLDB 12, 11 (2019), 1705–1718. https://doi.org/10.14778/ 3342263.3342644[43] Ryan Marcus 与 Olga Papaemmanouil. 2016. WiSeDB：面向云数据库的基于学习的工作负载管理顾问. PVLDB 9, 10 (2016), 780–791. https://doi.org/10.14778/2977797.2977804 tex.acmid= 2977804 tex.issue_date=

2016年6月 tex.numpages= 12. [44] Volodymyr Mnih、Koray Kavukcuoglu、David Silver、Andrei A. Rusu、Joel Veness、Marc G. Bellemare、Alex Graves、Martin Riedmiller、Andreas K. Fidjeland 与 Georg Ostrovski. 2015. 通过深度强化学习实现人类水平控制. 《自然》 518, 7540 (2015), 529–533. [45] Songsong Mo、Yile Chen、Hao Wang、Gao Cong 与 Zhifeng Bao. 2023. Lemo：面向并发查询的缓存增强型学习优化器. ACM 数据管理会议录 1, 4 (2023年12月), 247:1–247:26. https://doi.org/10.1145/3626734[46] Sidharth Mudgal、Han Li、Theodoros Rekatsinas、AnHai Doan、Youngchoon Park、Ganesh Krishnan、Rohit Deep、Esteban Arcaute 与 Vijay Raghavendra. 2018. 用于实体匹配的深度学习：设计空间探索. 见2018年国际数据管理会议论文集 (SIGMOD '18). ACM, 纽约, NY, 美国, 19 34 10 1145 3183713 3196926 – . [47] 1964 https://doi.org/ . / . E. A. Nadaraya. . 论回归估计. 《概率论及其应用》 9, 1 (1964), 141–142. https://doi.org/10.1137/1109020 _eprint: https://doi.org/10.1137/1109020. [48] Raghunath Othayoth Nambiar 与 Meikel Poess. 2006. TPC-DS 的构建. 见 VLDB (VLDB '06). VLDB基金会, 首尔, 韩国, 1049–1058. http: //dl.acm.org/citation.cfm?id =1182635 1164217 . Vikram Nathan、Jialin Ding、[49] Mohammad Alizadeh 与 Tim Kraska. 2019. 学习多维索引. 见 NeurIPS 系统机器学习研讨会 (MLForSystems @ NeurIPS '19). [50] Andrew Patterson、Samuel Neumann、Martha White 与 Adam White. 2024. 强化学习中的经验设计. 《机器学习研究期刊》 25 (2024), 1–63. [51] Andrew Pavlo、Evan P. C. Jones 与 Stan Zdonik. 2011. 论用于优化并行 OLTP 系统中事务执行的预测建模. PVLDB 5, 2 (2011 86 96 10 14778 2078324 2078325), – . https://doi.org/ . / . [52] Matthew Perron、Zeyuan Shang、Tim Kraska 与 Michael Stonebraker. 2019. 我如何学会不再担忧并爱上重新优化. 2019年IEEE第35届国际数据工程会议 (ICDE) (2019年4月), 1758–1761. https://doi.org/10.1109/ICDE.2019.00191 会议名称：2019年IEEE第35届国际数据工程会议 (ICDE) ISBN: 9781538674741 地点：澳门, 澳门 出版方: IEEE. [53] Mark Raasveldt 与 Hannes Mühleisen. 2019. DuckDB：一种可嵌入的分析型数据库. 见2019年国际数据管理会议论文集 (SIGMOD '19). 美国计算机协会, 纽约, NY, 美国, 1981 1984 10 1145 3299869 3320212 – . https://doi.org/ . / . [54] Bogdan Răducanu、Peter Boncz 与 Marcin Zukowski. 2013. Vectorwise 中的微自适应. 见2013年ACM SIGMOD 数据管理国际会议论文集 (SIGMOD '13). 美国计算机协会, 纽约 , NY, 美国 , 1231 1242 10 1145 2463676 2465292 – . [55] 2022 https://doi.org/ . / . Ibrahim Sabek、Tenzin Samten Ukyab 与 Tim Kraska. . LSched：面向分析型数据库系统的工作负载感知学习型查询调度器. 见2022年国际数据管理会议论文集 (SIGMOD '22). 美国计算机协会, 纽约, NY, 美国, 1228–1242. https://doi.org/10.1145/3514221.3526158[56] Gaurav Saxena、Mohammad Rahman、Naresh Chainani、Chunbin Lin、George Caragea、Fahim Chowdhury、Ryan Marcus、Tim Kraska、Ippokratis Pandis 与 Balakrishnan (Murali) Narayanaswamy. 2023. Auto-WLM：Amazon Redshift 中的机器学习增强型工作负载管理. 见2023年国际数据管理会议配套论文集 (SIGMOD '23). https: // doi.org/ 10 1145 3555041 3589677 . / . Tobias Schmidt、Viktor Leis、[57] Peter Boncz 与 Thomas Neumann. 2025. SQLStorm：将数据库基准测试带入大语言模型时代. PVLDB 18, 11 (2025), 4144– 4157 10 14778 3749646 3749683 . https://doi.org/ . / . John [58] Schulman、Filip Wolski、Prafulla Dhariwal、Alec Radford 与 Oleg Klimov. 2017. 近端策略优化算法. arXiv:1707.*06347 [*cs*]* (2017年7月). http://arxiv.org/abs/1707.06347 arXiv: 1707.06347. [59] Shrainik Jain、Jiaqi Yan、Thiery Cruanes 与 Bill Howe. 2019. 数据库无关的工作负载管理. 见第9届创新数据系统研究双年会 (CIDR '19). [60] Richard S. Sutton 与 Andrew G. Barto. 1998. 强化学习导论 (第1版). 麻省理工学院出版社, 剑桥, MA, 美国. [61] Pierre Thodoroff、Wenyu Li 与 Neil D. Lawrence. 2022. 对实时强化学习进行基准测试. 见 NeurIPS 2021 机器学习预注册研讨会. PMLR, 26–41. https://proceedings.mlr.press/ v181/thodoroff22a.html[62] Immanuel Trummer、Samuel Moseley、Deepak Maram、Saehan Jo 与 Joseph Antonakakis. 2018. SkinnerDB：通过强化学习实现遗憾有界查询评估. PVLDB 11, 12 (2018), 2074–2077. https://doi.org/10.14778/ 3229863.3236263[63] Kostas Tzoumas、Timos Sellis 与 Christian Jensen. 2008. 一种用于自适应查询处理的强化学习方法. 技术报告 (2008年6月). [64] Shuhei Watanabe. 2025. 树结构 Parzen 估计器：理解其算法组件及其对提升经验性能的作用. https: // doi.org/10.48550/arXiv.2304.11127 arXiv:2304.11127 [cs].

[65] Lianggui Weng、Rong Zhu、Di Wu、Bolin Ding、Bolong Zheng 和 Jingren Zhou。2024。Eraser：消除学习型查询优化器中的性能退化。*PVLDB* 17, 5 2024 926 938 10 14778 3641204 3641205 ( ), –。[66] 1992 https://doi.org/ 。/ 。Ronald J. Williams。 。用于连接主义强化学习的简单统计梯度跟随算法。载于*Machine Learning*（*Machine Learning '92*）。229–256。[67] Lucas Woltmann、Jerome Thiessat、Claudio Hartmann、Dirk Habich 和 Wolf- gang Lehner。2023。FASTgres：让学习型查询优化器提示发挥作用。*VLDB* 学报 16，11（2023年8月），3310–3322。https://doi.org/10.14778/3611479.3611528[68] Maryann Xue、Yingyi Bu、Abhishek Somani、Wenchen Fan、Ziqi Liu、Steven Chen、Herman Van Hovell、Bart Samwel、Mostafa Mokhtar、Rk Korlapati、Andy Lam、Yunxiao Ma、Vuk Ercegovac、Jiexing Li、Alexander Behm、Yuanjian Li、Xiao Li、Sriram Krishnamurthy、Amit Shukla、Michalis Petropoulos、Sameer Paranjpye、Reynold Xin 和 Matei Zaharia。2024。面向大规模湖仓的自适应且稳健的查询执行。*VLDB* 学报 17，12（2024年8月），3947 3959 10 14778 3685800 3685818 –。https://doi.org/ 。/ 。[69] Zongheng Yang、Wei-Lin Chiang、Sifei Luan、Gautam Mittal、Michael Luo 和 Ion Stoica。2022。Balsa：无需专家演示即可学习查询优化器。载于*2022*年美国计算机协会数据管理国际会议论文集（*SIGMOD '22*）。美国计算机协会，纽约，纽约州，美国，931 944 10 1145 3514221 3517885 –。https://doi.org/ 。/ 。[70] Xiang Yu、Guoliang Li、Chengliang Chai 和 Nan Tang。2020。使用树形长短期记忆网络进行强化学习，实现连接顺序选择。载于*2020*年 *IEEE* 第*36*届国际数据工程会议（*ICDE '20*）。1297–1308。https://doi.org/10.1109/ ICDE48307.2020.00116 ISSN：2375-026X。[71] Chi Zhang、Ryan Marcus、Anat Kleiman 和 Olga Papaemmanouil。2020。通过深度强化学习实现缓冲池感知的查询调度。载于第二届数据库系统与应用人工智能应用国际研讨会（*AIDB@VLDB '20*），Bingsheng He、Berthold Reinwald 和 Yingjun Wu（编）。日本东京。https://drive.google.com/file/d/1trNYAcQ3S71SHu5dbtkBR2hjcK-VWFSx/view?usp=sharing [72] Yunjia Zhang、Yannis Chronis、Jignesh M. Patel 和 Theodoros Rekatsinas。2023。简单自适应查询处理与学习型查询优化器：观察与分析。*VLDB* 学报 16，11（2023年7月），2962–2975。https://doi.org/10.14778/3611479.3611501[73] Junyi Zhao、Kai Su、Yifei Yang、Xiangyao Yu、Paraschos Koutris 和 Huanchen Zhang。2025。揭穿连接排序的迷思：迈向稳健的 SQL 分析。*ACM* 数据管理会议录 3，3（2025年6月），146:1–146:28。https: //doi.org/ 10 1145 3725283 。/ Zijie Zhao 和 Ryan Marcus。2026。*Piece of CAKE*：通过微秒级学习实现自适应执行引擎的附录。附录。宾夕法尼亚大学。[74] https://rm.cab/cake_附录 [75] Li Zhou。2016。上下文多臂老虎机综述。http://arxiv.org/abs/1508.03326[76] Rong Zhu、Wei Chen、Bolin Ding、Xingguang Chen、Andreas Pfadler、Ziniu Wu 和 Jingren Zhou。2023。Lero：一种学习排序查询优化器。*VLDB* 学报 16，6（2023年2月），1466–1479。https://doi.org/10.14778/3583140.3583160[77] Rong Zhu、Lianggui Weng、Wenqing Wei、Di Wu、Jiazhen Peng、Yifan Wang、Bolin Ding、Defu Lian、Bolong Zheng 和 Jingren Zhou。2024。PilotScope：利用机器学习驱动器来引导数据库。*PVLDB* 17，5（2024），980–993。https://doi.org/ 10 14778 3641204 3641209 。/ 。Sergey Zinchenko 和 Sergey Iazov。[78] 2024。HERO：基于提示的高效可靠查询优化器。https://doi.org/10.48550/arXiv.2412.02372 arXiv:2412.02372 [cs]。[79] Joshua P Zitovsky。2023。重新审视用于离线模型选择的贝尔曼误差。载于第*40*届国际机器学习会议论文集（*ICML '23*）。美国夏威夷。