文章编号:1671-6833(2024)01-0001-11

特约述评:数据库系统参数优化

【特约专家】石磊:河南省教育厅学术技术带头人

【按 语】数据库是承载数字经济发展的重要基础软件,是网络强国和数字中国建设的关键支撑,具有举足轻重的战略地位。数字经济时代,数据迎来快速增长,数据利用价值越来越大,数据使用频率越来越高,对数据库高可靠、高性能的要求水涨船高。《"十四五"软件和信息技术服务业发展规划》指出,要突破高可靠数据存储引擎等关键技术,推动高性能数据库在重点行业关键系统中的应用。数据库参数优化是提升数据库性能的关键技术,参数优化通过调整数据库的各项配置,如内存分配、线程数量、I/O 管理和查询优化等来提高系统的性能。合理的数据库配置可避免系统过度负荷、延长硬件使用寿命、提高吞吐量和访问速度,有助于在关键时刻做出快速响应,为国家战略决策提供实时和准确的数据支撑。例如,防范网络威胁、保护关键基础设施和确保公共安全等,都依赖于高效运行的数据库。然而,数据库中可调节的配置一般高达上百项,且配置种类多,配置之间关联性强,为数据库参数优化带来挑战。《基于机器学习的数据库系统参数优化方法综述》系统回顾了数据库参数优化方法的发展历程,总结了国内外相关方法,包括专家决策、静态规则、启发式算法、传统机器学习方法和深度强化学习方法,剖析了各类方法的优缺点,综述了基于深度强化学习的参数优化方法,描述了模型的架构和工作原理,并针对数据库参数优化方法的未来发展趋势进行了展望。

基于机器学习的数据库系统参数优化方法综述

石 磊[1,2,3],李 天[2],高宇飞[1,3],卫 琳[1],李翠霞[1],陶永才[2]

(1.郑州大学 网络空间安全学院,河南 郑州 450002;2.郑州大学 计算机与人工智能学院,河南 郑州 450001;3.嵩山实验室,河南 郑州 450046)

摘 要: 参数优化是影响数据库性能和适应性的关键技术,合理的参数配置对于保障数据库系统的高效运行至关重要,但由于参数较多且参数间具有强关联性,传统参数优化方法难以在高维连续的参数空间中寻找最优配置,机器学习的发展为解决这一难题带来新的机遇。通过总结和分析相关工作,将已有工作按照发展时间和特性分为专家决策、静态规则、启发式算法、传统机器学习方法和深度强化学习方法。对数据库参数优化问题进行定义,并说明启发式算法在参数优化问题上的局限性。介绍基于传统机器学习的参数优化方法,包括随机森林、支持向量机、决策树等,描述机器学习方法解决参数优化问题的一般流程并给出一般实现。由于需要大量带标注的数据,传统机器学习模型在适应性和调优能力等方面存在不足。侧重介绍深度强化学习模型的工作原理,定义参数优化问题与深度强化学习模型的映射关系,比较基于深度强化学习的相关工作对数据库性能提升、模型训练时间和涉及的技术,描述基于深度神经网络构建和训练智能体的具体流程。最后,总结已有工作的特点,对当前机器学习在数据库参数优化方面的研究热点和发展方向进行展望,指出多粒度调优、自适应算法和自运维是未来的研究趋势。

关键词: 数据库系统; 参数优化; 性能优化; 机器学习; 强化学习; 数据库运维

中图分类号: TP392; TP319; N945.15 **文献标志码:** A **doi:** 10.13705/j.issn.1671-6833.2024.01.008

收稿日期: 2023-08-20;**修订日期:** 2023-09-19

基金项目: 国家重点研发计划(2022YFC3800057, 2020YFB1712401)

作者简介: 石磊(1967—),男,河南郑州人,郑州大学教授,博士,博士生导师,主要从事数据科学与智能计算、网络与分布式计算、信息安全等研究,E-mail:shilei@zzu.edu.cn。

引用本文: 石磊,李天,高宇飞,等. 基于机器学习的数据库系统参数优化方法综述[J]. 郑州大学学报(工学版),2024,45(1):1-11,28. (SHI L, LI T, GAO Y F, et al. A review of machine learning-based methods for database tuning[J]. Journal of Zhengzhou University (Engineering Science),2024,45(1):1-11,28.)

数据库是信息系统用于存储、分析和管理数据的基础设施,直接影响信息系统的可靠性。数据库中有上百项可调整的参数,控制着各组件的表现,如openGauss的参数max_process_memory控制数据库节点上的最大可用内存,通过调整该参数可为各类缓存分配更大内存,减少磁盘读写次数,同时避免操作系统内存溢出(out of memory,OOM)。

数据库参数优化面临以下挑战。

(1)参数空间大。假设有 n 个参数,每个参数有 k 种取值,则可选择的配置有 k^n 个,若参数取值范围为一个连续区间,则配置数目无穷。因此,在高维参数空间内寻找最优配置是NP-Hard问题[1]。

(2)训练样本少。为确定一组配置对应的数据库表现,需要在数据库上运行较长时间的压力测试,以获取吞吐量等性能指标的变化。但是,为保障数据库的稳定性,调优时间有限,因此获取训练样本的时间短、数量少。

(3)环境变化多。数据库的性能与软硬件环境和工作负载等因素密切相关,在生产环境中,这些因素随时可能发生变化,导致之前的最优配置失效,需要针对新的环境重新优化。

参数优化方法通过自动选择合适的配置来快速提升数据库的性能,现有数据库参数优化方法可分为专家决策、启发式算法、传统机器学习方法和深度强化学习方法。专家决策基于人工经验调整数据库,是早期数据库参数优化的主要手段;启发式算法将参数空间离散化,通过抽样来搜索最优解,难以在多项式时间内搜索到合适的配置;传统机器学习算法基于训练样本找到数据库状态和参数之间的映射关系,效率较高,但是需要大量高质量的训练样本;深度强化学习结合神经网络和强化学习的优势,在高维参数空间中以试错的方式进行探索,减少了对训练样本的依赖性。

传统参数优化方法依赖于人工调整或启发式算法,随着数据库实例越来越多,场景越来越复杂,传统的参数优化方法难以取得理想的结果。随机森林、支持向量机和决策树等方法能够通过历史数据积累"经验",提升解决问题的能力。但是传统机器学习模型难以解决像数据库参数优化这样具有高维连续空间的优化问题。随着算力的发展,深度学习和强化学习[2-3](reinforcement learning,RL)为此类问题带来更好的解决方案。深度学习利用神经网络的学习能力,通过调整神经元之间的连接来模拟数据的映射关系,适用于传统方法难以解决的问题或存在大量训练样本的场景。数据库有着复杂的参数和大量的数据,为深入学习解决数据库参数优化问题提供了很好的先验条件。强化学习旨在训练一个智能体与环境交互,智能体观测环境后根据策略生成配置,然后根据环境的反馈来调整策略,以提升配置的质量。强化学习以"试错"的方式进行学习,因此不需要预先准备大量的样本,深度强化学习结合了深度学习和强化学习的优点,在工业界复杂的场景下有更好的表现。

本文首先对数据库参数优化问题进行定义,并对相关方法进行分类;其次,分析现有数据库参数优化方法的关键特征;最后,结合现阶段存在的问题,对该领域未来的研究方向进行展望。

1 数据库参数优化方法

为避免混淆,使用"配置参数"表示所有可配置的参数构成的集合;"配置"表示配置参数的一组完整取值;"参数"表示配置参数集合中的某个参数;"参数值"表示参数的具体取值。

1.1 数据库参数优化问题定义

数据库通常涉及上百项可调节的参数,运行环境和性能指标也较复杂。对数据库参数优化的定义如下。

(1)配置。假设数据库系统中共有 n 个可调整的参数,记为

$$\boldsymbol{x} = < x_1, x_2, \cdots, x_n > 。 \tag{1}$$

式中:x_i 为第 i 个参数的取值;$\boldsymbol{x}$ 为一个配置,包含了所有可能的参数值的组合,每个配置表示 n 维参数空间的一个点。

(2)运行环境。数据库的表现不仅受参数影响,同时也和工作负载和硬件配置等运行环境密切相关。将运行环境集合记为 E,每一种独特的环境记为 e,其中 $e \subseteq E$。

(3)系统性能。性能指标有吞吐量、响应时间等,设 P 表示数据库的整体性能,P 的计算方式通常不唯一。P 受配置 $\boldsymbol{x}$ 和环境 e 的影响,P 与 $\boldsymbol{x}$ 的映射关系定义为

$$P = f(\boldsymbol{x}, e)。 \tag{2}$$

式中:$f(\boldsymbol{x}, e)$ 为数据库系统在环境 e 下,部署配置 $\boldsymbol{x}$ 的性能,参数优化的方向即为最大化函数 $f(\boldsymbol{x}, \mathrm{e})$。

(4)约束条件。约束条件指变量 x_i 必须满足某些条件,如数据库中每个参数的值都必须在规定的取值范围内。此外参数优化消耗的时间、资源等也构成了约束条件的一部分。

数据库参数优化问题可定义为

$$\max f(\boldsymbol{x},e); \tag{3}$$

$$\text{s.t.}\quad g_i(\boldsymbol{x}) \leqslant 0,\quad i=1,2,\cdots,p; \tag{4}$$

$$h_i(\boldsymbol{x}) = 0,\quad i=1,2,\cdots,q。 \tag{5}$$

式中:$\boldsymbol{x}$ 为数据库配置;e 为环境;求解目标为最大化 $f(\boldsymbol{x},e)$,$g_i(\boldsymbol{x}) \leqslant 0$ 和 $h_j(\boldsymbol{x})=0$ 为求解目标函数需要满足的第 i 或 j 个约束条件。参数优化问题可概括为在满足约束条件的同时,为数据库寻找合适的配置,使得数据库在特定环境达到最优的性能。

1.2　数据库参数优化方法介绍

多年来,学者通过各种方式尝试解决这一问题[4-11],参数优化方法分类如表 1 所示。

表 1　数据库参数优化方法比较

Table 1　Comparison of database tuning methods

方法	优点	缺点	优化结果	泛化能力
专家决策	满足基本需求	人工配置耗时耗力,适应性差	低	弱
基于静态规则[4]	基于规则,原理简单易实现	适应性差	低	弱
启发式方法[7]	自动执行,速度快	不能利用历史经验	低	弱
传统机器学习[9]	可利用历史经验	需要高质量样本	中	中
深度强化学习[11]	结果好,不依赖样本	波动性大,不稳定	高	强

1.2.1　专家决策

对数据库参数的调整依赖于数据库管理员(database administrator,DBA),管理员通过手工多次实验得到统计结果,再根据经验调整数据库参数。但是数据库参数较多,多个参数组成的高维参数空间较大,通过手工抽样的方式难以获得最优解。此外,手工测试每个样本的质量费时费力,调优结果难以适应变化的生产环境,因此,有学者尝试使用经典算法来解决该问题。

1.2.2　基于规则的参数优化方法

为提高调优效率,DBA 使用工具来辅助决策,使用某种方法,分析出模糊或确切的规则,然后辅助 DBA 调整数据库参数。如 PostgreSQLTune 能够收集并分析数据库的状态(如当前缓存命中率等),分析出影响数据库性能的关键参数,根据分析的结果和固定的规则来推荐用户修改某些参数。Trummer[12] 提出了一种自动从文本中提取出规则的工具 DB-BERT。DB-BERT 使用预先训练好的自然语言处理模型 BERT[13] 从数据库手册中识别出和参数相关的文本,将文本转换成规则,根据规则推荐数据库配置。这类工具虽然能够减轻 DBA 的负担,但其核心是基于规则来调整参数,应对的场景有限,面对复杂的数据库环境和硬件,难以保持良好的表现。MySQLTuner[14] 是一款基于规则调整 MySQL 配置的工具,可在不停机的情况下实时获取数据库的状态,给出调整参数的建议。Xu 等[15] 针对如何选择系统性能优化最相关的参数开展研究,结果表明,大量的数据库参数没有调整的价值,因此有必要进行参数筛选;简化枚举类型的参数可有效降低计算量,且对调优结果的影响几乎可以忽略。

1.2.3　基于启发式算法的参数优化方法

启发式算法调整参数的主要思路是“搜索”,例如 Zhu 等[16] 提出了一个基于搜索的参数优化系统 BestConfig。BestConfig 使用网格抽样(grid sampling)的方式将连续的参数取值范围离散化:参数取值范围被划分为 k 段,然后在每段上随机取 1 个值,这样 n 个参数可以组合出的配置为 k^n 个。为减少搜索时间,BestConfig 并不测试每一个配置,而是使用有界递归搜索算法随机地选取 k 个样本进行测试,然后以表现好的样本为中心,重新开始搜索,如此迭代可在有限的资源下找到表现较好的配置。

此外,还有针对特定类型和版本的数据库调优指南,例如 Oracle 数据库调优指南[17]、DB2 数据库调优手册[18]、Azure SQL 数据库的性能优化指南[19]。以上方法主要依赖于固定的参数优化规则或使用搜索算法找到合适的配置,在效率上高于 DBA,但无法完全解决参数优化这类 NP-Hard 问题,也不能在历史经验的基础上更新模型,每次调优都要重新开始,模型的优化能力有限。

1.2.4　基于机器学习的参数优化方法

不同于启发式算法,机器学习方法可以从训练样本中学习输入与输出之间的关系,因此能够对输出给出更为准确的预测。现有的研究成果中,主要使用了传统机器学习和深度强化学习两种方法。

2　基于传统机器学习的参数优化方法

2.1　方法概述

传统机器学习方法包括决策树、线性回归和神经网络等多种经典模型,利用训练数据进行回归分析和模式识别等,一般包括以下几个步骤。

步骤 1　性能采样。数据库的主要性能指标为

吞吐量和延迟,使用采样算法或随机采样,得到若干配置,通过实验或者仿真等方式对配置进行评估,记录下每种配置部署到数据库后对数据库带来的提升,得到训练样本。

步骤2 构建性能预测模型。如图1所示,基于步骤1得到的训练样本,利用机器学习方法,如回归模型等,预测数据库与某一配置下的性能之间的隐含关系。

步骤3 寻找最优配置。利用遗传算法和梯度上升等方法,尝试不同的配置,找到使系统性能达到最优的配置。

基于传统机器学习的参数优化方法如表2所示。

2.2 经典模型

目前较常用的方法是根据专家的经验筛选出对数据库性能影响最大的参数,同时也有一些研究人员使用机器学习技术对相关性较强的参数进行排序,例如Lasso和方差分析。Lima等[27]使用递归特征消除(recursive feature elimination,RFE)和其他有监督学习方法评估配置的表现,迭代地减少相关性较弱的参数。算法分为3步:①根据预测的性能指标来给配置的表现打分;②利用线性回归和决策树等方法对参数排序;③基于随机森林、梯度提升机

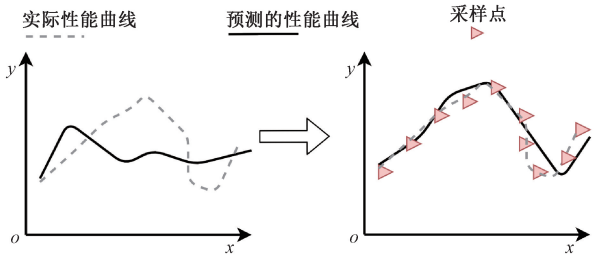

图1 构建性能预测模型

Figure 1 Constructing performance prediction model

表2 基于传统机器学习的数据库参数优化方法

Table 2 Traditional machine learning-based database tuning methods

方法	使用的技术	吞吐量提升/%	延迟降低/%	运行时间/h
iTuned[20]	高斯过程、拉丁超立方抽样	200~500	200~500	1~8
OtterTune[21]	因子分析、k-均值、Lasso回归	12~35	57~60	<1
iBTune[22]	神经网络	—	—	—
Rafiki[23]	遗传算法、神经网络	14.2~41.1	—	0.2
LlamaTune[24]	高斯过程、贝叶斯优化	1.08~21	—	0.7~0.45
CGPTuner[25]	高斯过程、贝叶斯优化	—	—	<1
OPTIMUSCLOD[26]	随机森林、遗传算法	40	450	—

(gradient boosting machine,GBM)和决策树构建性能预测模型。实验结果表明,在准确性上,使用参数子集训练的模型优于使用整个参数集训练的模型,说明了选择参数的重要性。

iTuned[20]是一款基于机器学习的自动数据库参数优化系统,其优化过程主要分为两步。首先,使用拉丁超立方体采样(Latin hypercube sampling,LHS)选择初始样本。然后,利用这些样本构建基于高斯过程[28](Gauss processes,GP)的性能曲面,并基于观测结果改进函数,在参数空间中继续选择采样点并在采样点上再次进行实验,得到对应的性能值后,更新性能曲面,重复这个过程,直到达到最优配置。在这个过程中,基于高斯过程的性能曲面逐渐逼近实际性能曲面,因此可以更快地找到最佳配置。但预测实际的性能曲面计算量较大,预测不够精确,尽管iTuned使用参数选择和工作负载压缩等方法提升调优速度,但在性能曲面上寻找最优解仍需要消耗大量时间,例如在MySQL数据库上选择7个参数进行调优,每次可消耗3.2~7.0 h,这使得iTuned难以应用在生产环境。

OtterTune[21]使用了与iTuned类似的策略,创新之处在于实验过程中使用之前学习到的知识来指导参数采样,通过利用历史数据,OtterTune可以更快地找到全局最优解,实现更好的性能优化效果。OtterTune优化参数的过程主要分为以下3步。

(1)特征抽取。从原始数据中选择与之相关的一部分特征,使用部分特征来代替原始数据参与计算。OtterTune根据数据库工作负载(workload)的特征来调整参数,不同的工作负载对参数的敏感性不同,因此,提取工作负载的特征并找到工作负载与参数的关联非常重要。工作负载的特征包含两方面:一方面是工作负载本身的特征,如负载中包含的查询语句的类型、关联的视图、存储过程和索引等;另一方面是数据库在执行负载时的内部状态,如内存使用率和缓存命中率等。

(2)降维。OtterTune使用因子分析(factor analysis,FA)对上一步中抽取出的特征进行筛选以去除无关特征,使用机器学习方法 k-means 选择与参数

相关的特征,筛选后的特征将作为模型输入的一部分。为进一步减少机器学习算法的搜索空间,OtterTune 利用 Lasso 算法来选取与数据库性能相关性最强的参数,并将参数按照重要性排序。

(3)搜索。OtterTune 根据特征将当前的工作负载 W 映射到一个相似的负载 W',W'是在以前的参数优化任务中出现过的负载,所以针对 W'调优的经验可以复用,从而提升模型的效率。但由于 W 和 W'并非完全相同,所以在调优时,系统综合考量 W 和 W'。

与 iTuned 类似,OtterTune 使用高斯过程估计出一个函数 $f(\boldsymbol{x})$,$\boldsymbol{x}$ 为配置,$f(\boldsymbol{x})$代表将 $\boldsymbol{x}$ 部署在数据库上执行负载 W 的性能。在 $f(\boldsymbol{x})$上进行梯度下降(gradient descent)可找到局部最优的点。整个梯度下降的过程需要 10~20 s,每次调整参数的结果都将被保存起来,以便后续利用历史经验,结果明显优于启发式方法。然而,在高斯过程中,需要估计每个数据点的均值和协方差,从而得到整个函数的概率分布。这需要求解一个 $n\times n$ 的协方差矩阵 $\boldsymbol{K}$,其中 n 为数据点的数量,$K_{i,j}=k(x_i,x_j)$表示数据点 x_i 和 x_j 的协方差,整个高斯过程的训练时间复杂度为 $O(n^3)$,因此难以处理高维或大规模的数据。

Gunasekaran 等[29]对 OtterTune 进行了改进,使用高斯混合模型(Gaussian mixture models,GMM)来修剪指标,并将集成模型(如随机森林)与非线性模型(如神经网络)结合起来进行建模。其中的一个关键步骤是利用历史调优数据来训练机器学习模型以提高预测的性能。此外,使用 GMM 聚类替代 k-means 聚类进行指标修剪,通过将每个簇视为不同的高斯分布,并考虑簇的均值和方差来克服 k-means 聚类的缺点。通过使用轮廓系数和贝叶斯信息准则(Bayesian information criterion,BIC)来确定 GMM 的最佳簇数。Gunasekaran 等[29]建议进一步研究 GMM 聚类作为 k-means 聚类的代替,并收集更多数据以深入探索不同方法。实验结果表明,基于 GPR(Gaussian process regression)的基准模型表现最好,平均百分比误差 *MAPE* 达到 69%。

沈忱等[30]基于 OtterTune 提出了一种改进的方法 D-OtterTune。D-OtterTune 利用动态负载匹配方法解决 OtterTune 等模型中对静态负载描述不够准确的问题,基于动态时间规整算法将数据对齐来解决负载匹配中序列不规则的问题。

iBTune[22]专注于对数据库缓冲池(buffer pool)进行优化,通过对缓存模型进行偏差分析(deviation analysis),在确保缓存命中率和响应时间的同时降低内存占用。Rafiki[23]是一种针对静态工作负载的自动调优工具,当工作负载发生变化时,需要重启调优过程并重启数据库,会对数据库的可用性和稳定性造成较大影响。LlamaTune[24]利用领域知识(domain knowledge)提升调优过程中的采样效率,利用随机投影对参数空间降维,并对参数的取值范围离散化,以进一步降低计算量。CGPTuner[25]综合考虑操作系统和物理硬件等因素,基于上下文高斯过程(contextual Gaussian process)进行优化。OPTIMUSCLOUD[26]通过分析虚拟机参数和数据库之间的依赖关系,基于随机森林预测给定软硬件环境下配置的表现。根据工作负载的变化情况决定何时更新配置,然后通过评估不同配置的适用性来确定更新哪些数据库实例。Ishihara 等[31]提出的调优模型采用高斯过程来搜索较优的数据库配置,能够应用于多种软件系统。Siegmund 等[32]综合利用抽样启发方法和机器学习方法为系统生成性能影响模型。Nair 等[33]提出了一种基于排序的性能预测模型,以降低构建性能模型的成本。

2.3　神经网络模型

深度神经网络可以通过多层神经元进行高维特征提取,对高维数据具有较好的处理能力,同时神经网络可以通过并行计算和 GPU 加速等方式高效处理大规模数据。Zheng 等[4]提出了一种基于神经网络的调优方法。从自动工作负载存储库(automatic workload repository,AWR)收集数据,包括系统的统计信息、会话日志和查询语句的执行记录,然后构建神经网络模型以获取数据库参数和性能指标(如吞吐量)之间的关系,最后采用调优算法优化这些关键参数。Rodd 等[34]也提出了一种基于神经网络的调优方法,通过收集数据库的关键性能指标为数据库的缓存推荐合适的值。Taft 等[35]针对 OLTP 类型的工作负载构建负载预测模型,在负载增加之前对计算资源进行预测性分配,结果优于被动式调优系统。ACTGAN[36]是基于生成式对抗网络(generative adversarial network,GAN)的软件系统优化模型,也可用于数据库参数优化。ACTGAN 利用随机抽样得到初始样本,从中选取表现较好的配置来训练 GAN 网络,不同于“搜索”的策略,ACTGAN 使用生成的方式直接推荐配置,这与强化学习的思路较为相似。Ha 等[37]结合深度前馈神经网络(deep feed-forward neural network,DFNN)和 $L1$ 正则化的系统性能预测方法,提出了一种可以对系统性能进行预测的模型 DeepPerf。该工作研究对象为软件系统,同时也适用于数据库管理系统。

Fang 等[38]提出了一种基于机器学习的查询级分布式数据库调优模型。首先,利用查询向量嵌入网络(query vector embedding network,QVEN)压缩原始输入,生成表示查询特征的向量。QVEN 的训练目标是使输出尽可能接近实际查询特征向量的压缩表示,可用于查询语句向量化。对输入的查询语句和对应的执行计划进行解析和降维生成表示查询特征的压缩查询向量。在此基础上,利用查询性能预测网络(query performance predict network,QPPN)来预测查询的执行时间。QPPN 模型由 3 层神经网络组成,包括线性层、BatchNorm 层和 ReLU 激活函数。在训练模型之前,输入的查询向量和配置向量会进行归一化处理,该模型的训练数据是由查询向量、配置向量和延迟时间组成的集合。通过运行工作负载并获取查询语句的执行计划,可以生成训练数据。在 CockroachDB 上的实验结果表明,该系统在典型的 OLAP 工作负载下实现了更高的性能,平均降低 9.2%的延迟,并且对于某些查询类型,延迟降低超过 60%。

3 基于深度强化学习的参数优化方法

传统机器学习方法利用过去的调优数据能够加速调参任务,但仍然存在一些局限性。首先,传统机器学习方法依赖大规模高质量的训练样本;其次,传统机器学习方法通常采用链式结构,在每个阶段获得的最优解并不能保证在下一阶段仍然是最优解;最后,数据库系统具有高维连续的参数空间,传统的机器学习模型拟合出的性能曲面不够精确,而且难以在多项式时间内寻找该曲面上的最优解。针对上述问题,一些研究人员开始尝试使用强化学习来解决数据库参数优化问题。

3.1 方法概述

强化学习不像传统机器学习那样需要大量带标注的样本,相反,智能体(agent)在与数据库交互的过程中,通过试错来学习如何生成高质量配置。强化学习的核心是利用智能体与环境(environment)进行交互,并通过在环境中执行一系列动作来学习如何最大化奖励(reward)。DQN[39]等方法可用于处理离散动作,而数据库中大部分参数的取值范围为一个连续区间,多个参数值构成的配置有无穷多个。Mnih 等[40]将深度神经网络与强化学习结合,提出了一种深度强化学习(deep reinforcement learning,DRL)模型,利用深度神经网络来处理强化学习中具有连续取值空间的参数。

将 DRL 应用到参数优化问题的最大挑战之一是基于强化学习模型对参数优化问题建模。强化学习的数学基础是马尔科夫决策过程(Markov decision process,MDP),如图 2 所示,在一个 MDP 中,智能体在观测环境的状态后,选择一个动作并执行,执行动作后,智能体根据执行结果获得一次奖励,随后,环境的状态可能发生变化,智能体再次观测环境状态和选择动作,这个过程不断重复,直至达到终止条件。智能体的目标是选择一个最优的策略,使得回报(累计奖励)最大化。

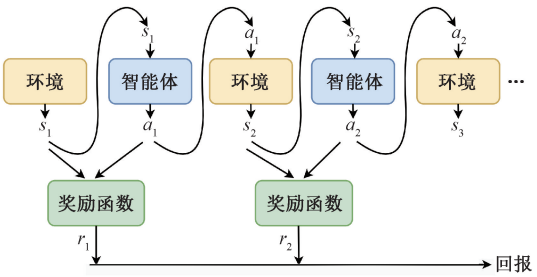

图 2 强化学习流程

Figure 2 Process of reinforcement learning

图 2 中 s_i 为第 i 步环境的状态,a_i 为第 i 步智能体的动作,r_i 为第 i 步智能体获得的奖励。强化学习与参数优化问题的映射如表 3 所示。

表 3 强化学习与参数优化问题的映射

Table 3 Mapping from RL to database tuning problem

强化学习	参数优化问题
智能体	深度神经网络
环境	数据库
动作	生成配置
动作空间	所有配置的集合
奖励	对数据库性能的提升
状态	数据库的状态

(1)智能体。智能体是做出决策的主体,智能体可以基于多种模型构建。

(2)环境。数据库被视为一个环境,智能体在与环境的交互中学习。

(3)动作。动作是智能体基于当前状态做出的决策。在参数优化问题中,动作是生成配置。

(4)动作空间。指所有可能动作的集合。在参数优化问题中,指所有可选择的数据库配置的集合。

(5)奖励。奖励是环境对于智能体动作的反馈,智能体生成的配置对数据库带来的提升越大,奖励越大,反之亦然。奖励的计算方式由奖励函数(reward function)定义。

(6)状态。对当前时刻数据库运行状态的概括,如当前死锁数量(lock_deads)和当前打开文件数量(file_num_open_files)等。

其中,智能体是整个参数优化模型的核心,智能

体的构建和训练直接影响模型的表现。

3.2　智能体的构建

智能体的形式并不唯一,通常需要根据强化学习的目标和环境的特点来设计。由于智能体需要在连续高维的参数空间中探索合适的配置,所以需要使用能够处理连续动作空间的算法来构建智能体。以 soft actor-cricit[41] 为例,智能体的结构如图 3 所示。其主要包含 actor 和 critic 两个神经网络,通过更新 actor 和 critic 网络的参数来学习生成高质量配置。

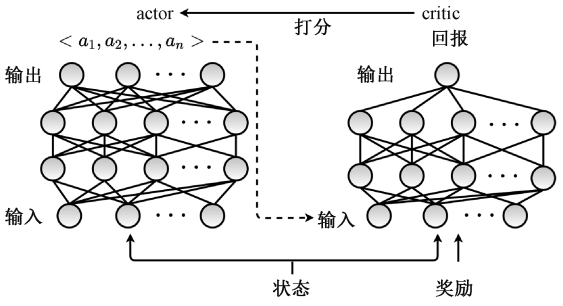

图 3　智能体架构图

Figure 3　Architecture of agent

给定任意随机变量 x,假设 x 的概率密度为 $P(x)$,则 x 的熵的定义如下:

$$H(P) = \mathop{\mathbb{E}}_{x \sim P}[-\log P(x)]。 \tag{6}$$

熵反映了一个系统的混乱程度,在参数优化中,该项表示智能体输出配置的多样化程度,该项可引导智能体尽可能地输出多样的配置,从而增加智能体在参数空间的探索能力。

智能体学习的过程是最大化目标函数的过程,目标函数不唯一,其中一种定义如下:

$$V^{\pi}(s) = \mathop{\mathbb{E}}_{\pi}\left[\sum_{t=0}^{\infty}\gamma^t(R(s_t,a_t) + \alpha H(\pi(\cdot \mid s_t)))\right]。 \tag{7}$$

式中:π 为智能体的策略;$V^{\pi}(s)$ 为在状态 s 下策略 π 表现的好坏程度;$R(s_t,a_t)$ 为奖励函数,表示智能体在环境的状态为 s_t 下做出动作 a_t 能够获得的奖励;α 为系数;$H(\pi(\cdot \mid s_t))$ 为智能体输出的动作的熵。

智能体对应的动作价值函数(action-value function)定义为

$$Q^{\pi}(s,a) = \mathop{\mathbb{E}}_{\pi}[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t) + \alpha\sum_{t=1}^{\infty}\gamma^t H(\pi(\cdot \mid s_t))]。 \tag{8}$$

动作价值函数代表在策略 π 和状态 s_t 下采取动作 a_t 可获得的回报的期望。回报(累计奖励)为多次奖励的加权和,根据式(7)和式(8),$V^{\pi}(s)$ 可改写为

$$V^{\pi}(s) = \mathop{\mathbb{E}}_{a \sim \pi}[Q^{\pi}(s,a)] + \alpha H(\pi(\cdot \mid s))。 \tag{9}$$

智能体中 actor 的输入是状态 s,输出为数据库配置;critic 的输入为状态 s、配置和奖励,输出为 1 个累计奖励的期望。智能体使用梯度上升更新 actor 和 critic 网络的权重来最大化目标函数。训练智能体的流程如下。

步骤 1　使用随机参数初始化 actor 和 critic 网络。

步骤 2　在 t 时刻,智能体观测环境状态 s_t 并由 actor 选择动作,即生成数据库配置。

步骤 3　从环境获得奖励 r_t 和新的状态 s_{t+1}。

步骤 4　critic 为 actor 输出的配置生成 1 个打分,打分代表智能体未来可获得的回报。打分越高,表示 actor 输出的配置的质量越好。

步骤 5　critic 根据 r_t 和式(8)更新网络参数,以获得更精确的打分。

步骤 6　actor 使用梯度上升最大化式(7),获得更高的打分。

步骤 1 至步骤 6 重复进行,直到智能体收敛或达到终止条件。以上步骤与实际情况略有出入,在实际中,还会使用经验回放和自动调整熵正则项优化等技术来提升智能体的稳定性和效率。

3.3　相关工作

深度强化学习相关参数优化系统如表 4 所示。

表 4　深度强化学习相关参数优化系统

Table 4　Database tuning system with reinforcement learning

相关工作	特点	吞吐量提升率/%	延迟降低率/%	运行时间/h
CDBTune+[11]	针对云数据库优化	800	400	50
Qtune[42]	提供 3 种不同细粒度的优化方案	400	400	20
Hunter[43]	基于强化学习综合使用多种机器学习算法	200	440	3
ResTune[44]	使用元学习加速优化过程	175	170	—

深度强化学习方法在吞吐量和延迟两方面表现更出色,强化学习能将吞吐量提升到 8 倍,而大部分传统机器学习方法仅能提升不到 1 倍。然而,由于智能体和数据库之间频繁交互,深度强化学习方法的运行时间更长。例如,CDBTune+[11] 运行时间长达 50 h,Hunter[43] 采用分布式训练可将运行时间缩

短至 3 h,传统机器学习方法则只需要 1 h 左右。

3.3.1 CDBTune

CDBTune 首次将深度强化学习应用到数据库参数优化问题上,基于深度确定性策略梯度算法(deep deterministic policy gradient,DDPG)[45-46]构建智能体。CDBTune 优化的指标为数据库的吞吐量和延迟,在 t 时刻,配置对两个指标的提升的定义如下:

$$\Delta T=\begin{cases}\Delta T_{t\to 0}=\dfrac{T_t-T_0}{T_0};\\ \Delta T_{t\to t-1}=\dfrac{T_t-T_{t-1}}{T_{t-1}}。\end{cases}\tag{10}$$

$$\Delta L=\begin{cases}\Delta L_{t\to 0}=\dfrac{-L_t+L_0}{L_0};\\ \Delta L_{t\to t-1}=\dfrac{-L_t+L_{t-1}}{L_{t-1}}。\end{cases}\tag{11}$$

式中:T_t 为 t 时刻的吞吐量;T_0 为数据库初始状态的吞吐量;$\Delta T_{t\to 0}$ 和 $\Delta T_{t\to t-1}$ 分别为 t 时刻与初始状态和上一状态相比较对吞吐量的影响;ΔT 为 t 时刻配置对吞吐量的总体影响;ΔL 为 t 时刻配置对延迟的总体影响。基于式(10)和式(11),奖励函数定义如下:

$$r=\begin{cases}((1+\Delta_{t\to 0})^2-1)\mid 1+\Delta_{t\to t-1}\mid,\Delta_{t\to 0}>0;\\ -((1-\Delta_{t\to 0})^2-1)\mid 1-\Delta_{t\to t-1}\mid,\Delta_{t\to 0}\leqslant 0。\end{cases}\tag{12}$$

奖励函数引导着智能体生成能够提升吞吐量和降低延迟的配置,若配置未能给数据库带来提升,则奖励函数返回 0 或负值。相较于传统机器学习方法,CDBTune 有以下优势。

(1)在奖励函数的引导下,智能体以试错的方式寻找最优配置,仅需少量样本即可实现较好的结果。

(2)在高维连续空间中表现优于 OtterTune 等基于传统回归方法的模型。

(3)利用端到端的方法减少出错概率,避免陷入局部最优。

CDBTune 的调优粒度较粗,仅能对 3 种工作负载调优:只读(read-only)型工作负载、只写(write-only)型工作负载、读写混合(read-write)型工作负载,无法对特定的工作负载优化。

3.3.2 Qtune

Qtune 同样基于 DDPG 构建智能体,但是 Qtune 可提供不同粒度的优化方案。Qtune 从 SQL 语句中提取特征,如查询类型、查询代价和查询关联的数据库表等,将这些特征提供给深度强化学习模型,以动态选择合适的参数配置。Qtune 的调优粒度从高到低分别为查询级调优、负载级调优、簇级调优。查询级调优针对单个查询生成合适的配置。这种方法延迟较低,但不能并行运行 SQL 语句,导致吞吐量较低。工作负载级调优面向工作负载推荐配置,吞吐量较高,但不能针对 SQL 语句推荐配置,因此具有高延迟。簇级调优兼顾了高吞吐量和低延迟,将 SQL 语句分为若干个组,以组为单位进行调优。Qtune 提出了一种基于深度学习的查询聚类方法,能够根据 SQL 语句的特征进行聚类。这种方法可以实现高吞吐量和低时延。用户可根据需求权衡吞吐量和延迟,选择不同的调优粒度。

3.3.3 Hunter

Qtune 和 CDBTune 能有效地生成高质量数据库配置,但是在训练智能体过程中,每回合都需要利用吞吐量和延迟计算奖励值,为获取准确的吞吐量和延迟,每次需要对数据库进行至少 1 min 的压力测试,这消耗了大量的时间。为解决该问题,Hunter 同样以 DRL 为核心,利用主成分分析(principal component analysis,PCA)和随机森林(random forest,RF)减少 DRL 模型的搜索空间,提升搜索效率,利用遗传算法为 DRL 模型生成训练样本,提升模型训练速度。为了进一步加速模型的训练,Hunter 克隆多个数据库实例并行训练,使用 10 个克隆的数据库实例可将训练速度提升约 3 倍。

3.3.4 ResTune

ResTune 使用元学习(meta-learning)从历史调优中抽取经验,加速带约束的贝叶斯优化问题,保证性能满足用户需求的同时,尽可能减少资源消耗。

DRL 融合 MDP 和梯度上升调整数据库参数,降低了对样本的需求,然而,智能体的动作具有一定的不确定性,尤其在初始阶段,智能体的策略近似于随机,因此可能生成危险的配置,进而导致数据库性能大幅下降甚至停机。此外,模型的表现受软硬件环境的影响。因此需要权衡多个因素,才能使总体效果达到最优。

4 趋势与展望

(1)参数优化的细粒度。已有的工作大多是针对数据库的吞吐量和延迟进行优化,这些指标可根据业务类型进一步进行细分,如写吞吐量,读吞吐量、写延迟、读延迟和缓存大小等。优化的对象也可以是单条查询、单个类型查询或某一用户的行为习惯。根据场景的变化,动态调整参数优化的细粒度能够使模型有更好的适应性[47]。

(2)基于变化环境的数据库参数优化。数据库在不同软硬件环境下表现有一定的差异,如何在环

境发生变化时能够用已有的经验快速生成针对新环境的优化结果,是一个重要的问题。迁移学习通过利用源域中学到的知识和经验来改善目标域中的学习性能,减少标注数据的需求,提升模型的泛化能力,并帮助解决样本不平衡问题。Jamshidi 等[48]利用迁移学习将部分历史经验从源域迁移至目标域以适应新的环境。然而,如何在环境剧烈变化时仍能充分利用已有系统的知识,是未来重要的研究方向。

(3)自运维的数据库。已有的数据库参数优化系统由人工设置优化指标和各种超参数,然后在接收到优化请求后,启动优化流程[49]。Pavlo 等[50]认为自运维的数据库应该满足:①自动选择进行何时进行何种操作以改进某项指标;②从历史经验中自动学习,并在没有人工干预的情况下根据历史经验做出决策。目前几乎没有能完全解决这类问题的工作。除了数据库参数调优外,未来的数据库也将朝着自主监控、自主诊断、自主修复等方向发展[51-56]。

5　结束语

数据库系统是一个由存储介质、处理对象和管理系统集合体构成的复杂系统,数据库自身的复杂性和外部环境的多样性为参数优化带来了多种挑战。相较于经典机器学习模型,神经网络能够更好地处理复杂数据,但是对训练样本的依赖性限制了其泛化能力。而强化学习基于马尔科夫决策过程,以试错的方式在连续高维空间内搜索最优解,减少了对训练数据的依赖。以强化学习为核心,并结合多种方法的数据库参数优化模型取得了较好的结果。此外,如何在自运维等场景下进行参数优化,已成为当前研究热点和难点。

参考文献:

[1] 李国良, 周煊赫, 孙佶, 等. 基于机器学习的数据库技术综述[J]. 计算机学报, 2020, 43(11): 2019-2049.
LI G L, ZHOU X H, SUN J, et al. A survey of machine learning based database techniques[J]. Chinese Journal of Computers, 2020, 43(11): 2019-2049.

[2] FRANÇOIS-LAVET V, HENDERSON P, ISLAM R, et al. An introduction to deep reinforcement learning[J]. Foundations and Trends in Machine Learning, 2018, 11(3/4): 219-354.

[3] 黄万伟, 郑向雨, 张超钦, 等. 基于深度强化学习的智能路由技术研究[J]. 郑州大学学报(工学版), 2023, 44(1): 44-51.
HUANG W W, ZHENG X Y, ZHANG C Q, et al. Research on intelligent routing technology based on deep reinforcement learning[J]. Journal of Zhengzhou University (Engineering Science), 2023, 44(1): 44-51.

[4] ZHENG C H, DING Z H, HU J L. Self-tuning performance of database systems with neural network[C]// 10th International Conference on Intelligent Computing. Piscataway: IEEE, 2014: 1-12.

[5] ZHANG X Y, WU H, LI Y, et al. Towards dynamic and safe configuration tuning for cloud databases[C]// Proceedings of the 2022 International Conference on Management of Data. New York: ACM, 2022: 631-645.

[6] KUNJIR M, BABU S. Black or white? how to develop an AutoTuner for memory-based analytics[C]//Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data. New York: ACM, 2020: 1667-1683.

[7] SCHNAITTER K, POLYZOTIS N. Semi-automatic index tuning: keeping DBAs in the loop[J]. Proceedings of the VLDB Endowment, 2012, 5(5): 478-489.

[8] FEKRY A, CARATA L, PASQUIER T, et al. To tune or not to tune? in search of optimal configurations for data analytics[C]//Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. New York: ACM, 2020: 2494-2504.

[9] FEKRY A, CARATA L, PASQUIER T, et al. Tuneful: an online significance-aware configuration tuner for big data analytics[EB/OL]. (2020-01-22)[2023-08-01]. https://arxiv.org/abs/2001.08002.

[10] MARCO A, BERKENKAMP F, HENNIG P, et al. Virtual vs. real: trading off simulations and physical experiments in reinforcement learning with Bayesian optimization[C]//2017 IEEE International Conference on Robotics and Automation (ICRA). Piscataway: IEEE, 2017: 1557-1563.

[11] ZHANG J, ZHOU K, LI G L, et al. CDBTune$^+$: an efficient deep reinforcement learning-based automatic cloud database tuning system[J]. The VLDB Journal, 2021, 30(6): 959-987.

[12] TRUMMER I. DB-BERT: a database tuning tool that "reads the manual"[C]//Proceedings of the 2022 International Conference on Management of Data. New York: ACM, 2022: 190-203.

[13] DEVLIN J, CHANG M W, LEE K, et al. BERT: pre-training of deep bidirectional transformers for language understanding[EB/OL]. (2019-05-24)[2023-08-01]. https://arxiv.org/abs/1810.04805.

[14] HAYDEN M. MySQLTuner needs you[EB/OL]. [2023-08-01]. https://github.com/major/MySQLTun-

er-perl.

[15] XU T Y, JIN L, FAN X P, et al. Hey, you have given me too many knobs!: understanding and dealing with over-designed configuration in system software[C]//Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering. New York: ACM, 2015: 307-319.

[16] ZHU Y Q, LIU J X, GUO M Y, et al. BestConfig: tapping the performance potential of systems via automatic configuration tuning[C]//Proceedings of the 2017 Symposium on Cloud Computing. New York: ACM, 2017: 338-350.

[17] Oracle. Oracle database online documentation 11g release2(11.2)[EB/OL]. [2023-08-01]. https://docs.oracle.com/cd/E11882_01/index.html.

[18] IBM. DB2 tuning overview[EB/OL]. [2023-08-01]. https://www.ibm.com/docs/en/sdse/6.4.0?topic=overview-db2-tuning.

[19] Microsoft. Tune applications and databases for performance in Azure SQL Database and Azure SQL Managed Instance[EB/OL]. [2023-08-01]. https://learn.microsoft.com/en-us/azure/azure-sql/database/performance-guidance?view=azuresql-mi.

[20] DUAN S Y, THUMMALA V, BABU S. Tuning database configuration parameters with iTuned[J]. Proceedings of the VLDB Endowment, 2009, 2(1): 1246-1257.

[21] VAN AKEN D, PAVLO A, GORDON G J, et al. Automatic database management system tuning through large-scale machine learning[C]//Proceedings of the 2017 ACM International Conference on Management of Data. New York: ACM, 2017: 1009-1024.

[22] TAN J, ZHANG T Y, LI F F, et al. iBTune: individualized buffer tuning for large-scale cloud databases[J]. Proceedings of the VLDB Endowment, 2019, 12(10): 1221-1234.

[23] MAHGOUB A, WOOD P, GANESH S, et al. Rafiki: a middleware for parameter tuning of NoSQL datastores for dynamic metagenomics workloads[C]//Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference. New York: ACM, 2017: 28-40.

[24] KANELLIS K, DING C, KROTH B, et al. LlamaTune: sample-efficient DBMS configuration tuning[EB/OL]. (2022-05-10)[2023-08-01]. https://arxiv.org/abs/2203.05128v1.

[25] CEREDA S, VALLADARES S, CREMONESI P, et al. CGPTuner[J]. Proceedings of the VLDB Endowment, 2021, 14(8): 1401-1413.

[26] MAHGOUB A, MEDOFF A, KUMAR R, et al. OPTIMUSCLOUD: heterogeneous configuration optimization for distributed databases in the cloud[C]// 2020 USENIX Annual Technical Conference. Berkeley: USENIX Association, 2020: 189-203.

[27] LIMA M I V, DE FARIAS V A E, PRACIANO F D B S, et al. Workload-aware parameter selection and performance prediction for in-memory databases[C]// Brazilian Symposium on Bioinformatics. Brazil: SBC, 2018: 169-180.

[28] SUI Y N, GOTOVOS A, BURDICK J, et al. Safe exploration for optimization with Gaussian processes[J] Proceedings of Machine Learning Research, 2015, 37: 997-1005.

[29] GUNASEKARAN K P, TIWARI K, ACHARYA R. Deep learning based auto tuning for database management system[EB/OL]. (2023-05-24)[2023-08-01]. https://arxiv.org/abs/2304.12747.

[30] 沈忱, 邰凌翔, 彭煜玮. 面向自动参数调优的动态负载匹配方法[J]. 计算机应用, 2021, 41(3): 657-661.
SHEN C, TAI L X, PENG Y W. Dynamic workload matching method for automatic parameter tuning[J]. Journal of Computer Applications, 2021, 41(3): 657-661.

[31] ISHIHARA Y, SHIBA M. Dynamic configuration tuning of working database management systems[C]//2020 IEEE 2nd Global Conference on Life Sciences and Technologies (LifeTech). Piscataway: IEEE, 2020: 393-397.

[32] SIEGMUND N, GREBHAHN A, APEL S, et al. Performance-influence models for highly configurable systems[C]// Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering. New York: ACM, 2015: 284-294.

[33] NAIR V, MENZIES T, SIEGMUND N, et al. Using bad learners to find good configurations[C]//Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering. New York: ACM, 2017: 257-267.

[34] RODD S F, KULKARNI U P. Adaptive self-tuning techniques for performance tuning of database systems: a fuzzy-based approach[C]//2013 2nd International Conference on Advanced Computing, Networking and Security. Piscataway: IEEE, 2013: 124-129.

[35] TAFT R, EL-SAYED N, SERAFINI M, et al. P-store: an elastic database system with predictive provisioning[C]// Proceedings of the 2018 International Conference on Management of Data. New York: ACM, 2018: 205-219.

[36] BAO L, LIU X, WANG F Z, et al. ACTGAN: automatic configuration tuning for software systems with generative adversarial networks[C]//2019 34th IEEE/ACM Interna-

tional Conference on Automated Software Engineering (ASE). Piscataway: IEEE, 2019: 465-476.

[37] HA H, ZHANG H Y. DeepPerf: performance prediction for configurable software with deep sparse neural network [C]//2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE). Piscataway: IEEE, 2019: 1095-1106.

[38] FANG X, ZOU Y, FANG Y G, et al. A query-level distributed database tuning system with machine learning [C]//2022 IEEE International Conference on Joint Cloud Computing (JCC). Piscataway: IEEE, 2022: 29-36.

[39] MNIH V, KAVUKCUOGLU K, SILVER D, et al. Playing Atari with deep reinforcement learning [EB/OL]. (2013-12-19) [2023-08-01]. https://arxiv.org/abs/1312.5602.

[40] MNIH V, KAVUKCUOGLU K, SILVER D, et al. Human-level control through deep reinforcement learning [J]. Nature, 2015, 518(7540): 529-533.

[41] HAARNOJA T, ZHOU A, HARTIKAINEN K, et al. Soft actor-critic algorithms and applications[EB/OL]. (2018-12-13) [2023-08-01]. https://arxiv.org/abs/1812.05905.

[42] LI G L, ZHOU X H, LI S F, et al. QTune: a query-aware database tuning system with deep reinforcement learning [J]. Proceedings of the VLDB Endowment, 2019, 12: 2118-2130.

[43] CAI B Q, LIU Y, ZHANG C, et al. HUNTER: an online cloud database hybrid tuning system for personalized requirements[C]//Proceedings of the 2022 International Conference on Management of Data. New York: ACM, 2022: 646-659.

[44] ZHANG X Y, WU H, CHANG Z, et al. ResTune: resource oriented tuning boosted by meta-learning for cloud databases [C]//Proceedings of the 2021 International Conference on Management of Data. New York: ACM, 2021: 2102-2114.

[45] 李琳, 李玉泽, 张钰嘉, 等. 基于多估计器平均值的深度确定性策略梯度算法[J]. 郑州大学学报(工学版), 2022, 43(2): 15-21.
LI L, LI Y Z, ZHANG Y J, et al. Deep deterministic policy gradient algorithm based on mean of multiple estimators[J]. Journal of Zhengzhou University (Engineering Science), 2022, 43(2): 15-21.

[46] SILVER D, LEVER G, HEESS N, et al. Deterministic policy gradient algorithms[C]//Proceedings of the 31st International Conference on International Conference on Machine Learning. New York: ACM, 2014: 1-9.

[47] LEE J, CHOI J, SEO S, et al. K2vTune: automatic database tuning with knob vector representation[EB/OL]. (2022-09-21)[2023-08-01]. https://ssrn.com/abstract=4225456.

[48] JAMSHIDI P, SIEGMUND N, VELEZ M, et al. Transfer learning for performance modeling of configurable systems: an exploratory analysis [C]//2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE). Piscataway: IEEE, 2017: 497-508.

[49] ZHANG X Y, CHANG Z, LI Y, et al. Facilitating database tuning with hyper-parameter optimization[J]. Proceedings of the VLDB Endowment, 2022, 15(9): 1808-1821.

[50] PAVLO A, ANGULO G, ARULRAJ J, et al. Self-driving database management systems[C]// Conference on Innovative Data Systems Research. Chaminade: CIDR, 2017: 1-6.

[51] LI G L, ZHOU X H, SUN J, et al. openGauss[J]. Proceedings of the VLDB Endowment, 2021, 14(12): 3028-3042.

[52] 李国良, 周煊赫. 轩辕: AI 原生数据库系统[J]. 软件学报, 2020, 31(3): 831-844.
LI G L, ZHOU X H. XuanYuan: an AI-native database systems [J]. Journal of Software, 2020, 31(3): 831-844.

[53] LI G L, ZHOU X H, CAO L. Machine learning for databases[C]//Proceedings of the First International Conference on AI-ML Systems. New York: ACM, 2021: 1-2.

[54] LI G L, ZHOU X H, CAO L. AI meets database: AI4DB and DB4AI[C]//Proceedings of the 2021 International Conference on Management of Data. New York: ACM, 2021: 2859-2866.

[55] SHI J C, CONG G, LI X L. Learned index benefits: machine learning based index performance estimation[J]. Proceedings of the VLDB Endowment, 2022, 15: 3950-3962.

[56] ZHAO X Y, ZHOU X H, LI G L. Automatic database knob tuning: a survey[J]. IEEE Transactions on Knowledge and Data Engineering, 2023, 35(12): 12470-12490.

(下转第 28 页)

Coefficient Optimization of Grinding Force Model Based on Genetic Algorithm

WANG Dong, ZHANG Zhipeng, ZHAO Rui, ZHANG Junyu, QIAO Ruiyong, SUN Shaozheng

(School of Mechanical and Power Engineering, Zhengzhou University, Zhengzhou 450001, China)

Abstract: When solving problems in the grinding force model, most of the methods of segmental calculation or column equations were used to calculate each coefficient directly, which not only demanded a large amount of calculation but also could not guarantee its accuracy. In addition the traditional regression model was easy to fall into local optimal, difficult to describe the nonlinear relationship. Therefore, the genetic algorithm was introduced into the parameter optimization of the nonlinear fitting function, and the coefficient optimization method of the theoretical model of grinding force was studied based on the existing model data such as the model of cylindrical transverse grinding, the model of plane grinding and the model of cylindrical longitudinal grinding. Correlation analysis results showed that the predicted accuracy of grinding force of the three models was increased by 14.69%-42.54%. The average error of normal grinding force predicted by the three models was 5.9%, 9.13% and 3.23%, respectively. The mean error of tangential force was 6.78%, 8.36% and 3.69%, respectively. Through comparison, it could be concluded that the optimized model had a better fitting degree, and the prediction accuracy of the model was significantly improved. The nonlinear fitting function GA-LSQ algorithm optimized by genetic algorithm was more suitable for solving grinding force model and could provide reference for predicting grinding force and parameter optimization in actual production.

Keywords: grinding force model; cylindrical grinding; surface grinding; empirical formula; model coefficient optimization; model prediction; genetic algorithm; nonlinear optimization function

---

(上接第11页)

A Review of Machine Learning-Based Methods for Database Tuning

SHI Lei[1,2,3], LI Tian[2], GAO Yufei[1,3], WEI Lin[1], LI Cuixia[1], TAO Yongcai[2]

(1. School of Cyber Science and Engineering, Zhengzhou University, Zhengzhou 450002, China; 2. School of Computer and Artificial Intelligence, Zhengzhou University, Zhengzhou 450001, China; 3. Songshan Laboratory, Zhengzhou 450046, China)

Abstract: Knobs tuning is a key technology that affects the performance and adaptability of databases. However, traditional tuning methods have difficulty in finding the optimal configuration in high-dimensional continuous parameter spaces. The development of machine learning could bring new opportunities to solve this problem. By summarizing and analyzing relevant work, existing work was classified according to development time and characteristics, including expert decision-making, static rules, heuristic algorithms, traditional machine learning methods, and deep reinforcement learning methods. The database tuning problem was defined, and the limitations of heuristic algorithms in tuning problems were discussed. Traditional machine learning-based tuning methods were introduced, including random forest, support vector machine, decision tree, etc. The general process of using machine learning methods to solve tuning problems was described, and specific implementations were provided. The shortcomings of traditional machine learning models in adaptability and tuning capabilities were also discussed. The principles of deep reinforcement learning models were emphasized, and the mapping relationship between tuning problems and deep reinforcement learning models was defined. Recent relevant work on improving database performance, time consumption and model characteristics was introduced, and the process of building and training agents based on deep neural networks was described. Finally, the characteristics of existing work were summarized, and the research hotspots and development directions of machine learning in database tuning were outlined. Distributed scenarios, multi-granularity tuning, adaptive algorithms and self-maintenance capabilities were identified as future research trends.

Keywords: database system; knobs tuning; performance optimization; machine learning; reinforcement learning; database maintenance