计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2019.20190446
56(9): 1803-1820, 2019

机器学习化数据库系统研究综述

孟小峰 马超红 杨 晨
(中国人民大学信息学院 北京 100872)
(xfmeng@ruc.edu.cn)

Survey on Machine Learning for Database Systems

Meng Xiaofeng, Ma Chaohong, and Yang Chen
(*School of Information*, *Renmin University of China*, *Beijing* 100872)

Abstract As one of the most popular technologies, database systems have been developed for more than 50 years, and are mature enough to support many real scenarios. Although many researches still focus on the traditional database optimization tasks, the performance improvement is little. Actually, with the advent of big data, we have met the new gap obstructing the further performance improvement of database systems. The database systems face challenges in two aspects. Firstly, the increase of data volume requires the database system to process tasks more quickly. Secondly, the rapid change of query workload and its diversity make database systems impossible to adjust the system knobs to the optimal configuration in real time. Fortunately, machine learning may be the dawn bringing an unprecedented opportunity for the traditional database systems to lead us to the new optimization direction. In this paper, we introduce how to combine machine learning into the further development of database management systems. We focus on the current research work of machine learning for database systems, mainly including the machine learning for storage management and query optimization, as well as automatic database management systems. This area has also opened various challenges and problems to be solved. Thus, based on the analysis of existing technologies, the future challenges, which may be encountered in machine learning for database systems, are pointed out.

Key words database systems; machine learning; learned index; automatic database systems

摘 要 数据库系统经过近50年的发展,虽然已经普遍商用,但随着大数据时代的到来,数据库系统在2个方面面临挑战.首先数据量持续增大期望单个查询任务具有更快的处理速度;其次查询负载的快速变化及其多样性使得基于DBA经验的数据库配置和查询优化偏好不能实时地调整为最佳运行时状态.而数据库系统的性能优化进入瓶颈期,优化空间收窄,进一步优化只能依托新的硬件加速器来实现,传统的数据库系统不能够有效利用现代的硬件加速器;数据库系统具有成百个可调参数,面对工作负载频繁变化,大量繁琐的参数配置已经超出DBA的能力,这使得数据库系统面对快速而又多样性的变化缺乏实时响应能力.当下机器学习技术恰好同时符合这2个条件:应用现代加速器以及从众多参数调节经验中学习.机器学习化数据库系统将机器学习技术引入到数据库系统设计中.一方面将顺序扫描转化为计算模型,从而能够利用现代硬件加速平台;另一方面将DBA的经验转化为预测模型,从而使得数据

收稿日期:2019-06-21;**修回日期**:2019-08-22
基金项目:国家自然科学基金项目(61532016,61532010,91846204,91646203,61762082);国家重点研发计划项目(2016YFB1000602,2016YFB1000603)
This work was supported by the National Natural Science Foundation of China (61532016, 61532010, 91846204, 91646203, 61762082) and the National Key Research and Development Program of China (2016YFB1000602, 2016YFB1000603).

库系统更加智能地动态适应工作负载的快速多样性变化.将对机器学习化数据库系统当前的研究工作进行总结与归纳,主要包括存储管理、查询优化的机器学习化研究以及自动化的数据库管理系统.在对已有技术分析的基础上,指出了机器学习化数据库系统的未来研究方向及可能面临的问题与挑战.

关键词 数据库系统;机器学习;学习化索引;自动化数据库系统

中图法分类号 TP311

数据库系统已经在商业中成熟应用,表现为产品级数据库系统如 Oracle,PostgreSQL 等持续稳定地运行,说明经典的数据库算法已经趋于完善,且基于这些算法的性能优化也已经达到很好的程度.但大数据时代下,数据库系统需要处理的数据量不断增加,工作负载也面临着快速而多变的特性,对数据库系统提出了更高的要求.一方面数据量持续增大,期望数据库系统具有更快的处理速度;另一方面查询负载的快速变化及其多样性要求数据库系统能够动态调整系统参数以达到最佳运行时状态.传统的数据库系统优化技术和静态地依靠数据库管理员(database administrator, DBA)对系统进行参数配置的方式,已不能处理新应用场景下对系统性能的优化.在人工智能蓬勃发展的推动下,将机器学习与数据库系统有机结合来改进系统性能成为本领域的研究热点,亦称为机器学习化的数据库系统研究.

在 2015 年 ACM SIGMOD 会议上,Ré 等人[1]最早明确提出关于机器学习与数据库系统结合的思考,并展开对数据库领域结合机器学习技术的激烈讨论.机器学习以其优异的特性迅速在大数据驱动的应用领域普及并逐渐成为主流.在机器学习的浪潮下,数据库系统应处于一个什么样的位置? 数据库系统是否会在这股浪潮下被冲击出历史的舞台? 由此,数据库界开始了构建机器学习化数据库系统的探索.

归纳起来,我们认为机器学习化的数据库系统研究得到大家的普遍关注有 3 方面原因:

1)数据库系统性能优化进入瓶颈期,由传统的 I/O 性能瓶颈转为 CPU 性能瓶颈.为揭示传统优化技术对数据库系统性能提升空间逐渐收窄的问题,本文采用作者提出的资源解耦方法[2]对现有的 PostgreSQL 系统进行性能测试.在 Linux Ubuntu 16.04 系统下 PostgreSQL 系统中,运行随机生成的 50 个 TPCH 查询,得到如下数据:在 PostgreSQL 系统执行查询的时间中,CPU 约占 68%,磁盘约占 19%,内存约占 13%,图 1 为 PostgreSQL 系统执行查询的时间占比(资源影响度).另外我们通过 perf 工具,在 Linux 系统下测得 PostgreSQL 系统上查询的 L1 缓存命中率为 97.74%.

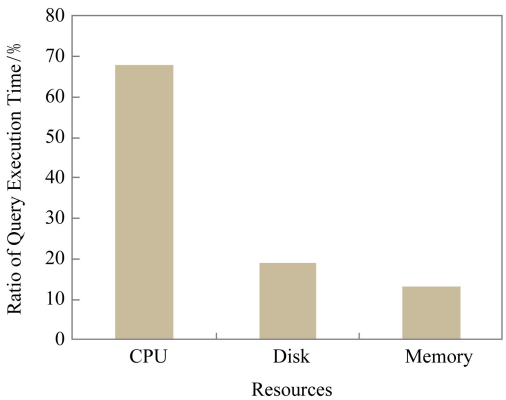

Fig. 1 Ratio of query execution time in PostgreSQL

图 1 PostgreSQL 数据库系统查询执行时间占比

由此看来,CPU 已成为数据库系统的主要瓶颈.可见,以优化 I/O 为主的传统数据库系统优化方法,难以优化当前的性能瓶颈,因此需要寻求新的途径.

2)大量面向机器学习的现代硬件加速器为解决数据库系统的 CPU 性能瓶颈提供了机会.机器学习技术的发展,新的加速算法、新硬件的迭代速度越来越快,但其更多集中在计算部件,如众核处理器、高性能处理器、GPU 处理器以及智能化处理器等,预计到 2025 年 GPU 的性能可以再提高 1 000 倍[3-4].然而这些成果和数据库系统的发展关系较远.对于数据库系统而言,传统的单线程算法和顺序访问特性不适合在现代加速器上运行.如 B 树索引,基于二分查找的法则对数据进行扫描.而现代加速硬件具有高并发、多线程、速度快的众核处理能力,这些加速器主要用于加速机器学习的并行和迭代算法,例如英伟达发布的 GeForce RTX 2080 Ti 拥有 4 352 个 CUDA 单元[5].显然,数据库系统要依托新的硬件加速平台实现其性能优化,必须对数据库系统的算法进行重构和改写.因此,机器学习化数据库系统的研究重点之一是将传统数据库系统的顺序操作转化为并行和迭代计算,例如采用机器学习模型替换

数据库系统内部的传统索引结构[3]、基数估计模型[6]等组件.

3) 数据库系统还面临着参数配置的问题,机器学习方法是解决这一问题的有效手段.数据库系统具有成百个参数,控制并影响着系统的性能,并且默认的参数配置较差,例如 2016 年 MySQL 系统默认其部署在仅有 160MB RAM 的机器上[7-8].同时,工作负载频繁而又多样性地快速变化已远超出 DBA 的能力,导致数据库系统在参数配置、系统管理方面缺乏快速响应和智能适配的能力.鉴于机器学习善于从训练数据中学习经验,并用于预测新的未知数据[9],同时快速给出预测结果.因此可借助机器学习从数据库系统长久以来的参数配置、查询优化等经验数据中,学习出不同工作负载下动态参数配置的能力,即将 DBA 的调优经验转化为机器学习模型,从而更好地适应快速变化的工作负载,动态地为数据库系统推荐最佳的系统配置.

综上,为借助机器学习算法的性能优势,同时利用现代加速硬件的特性,探索在数据库系统实现中引入机器学习技术,主要体现在 2 方面:一方面,借助机器学习的加速算法、现代加速器,提高数据库系统的处理速度,改善数据库系统的性能瓶颈;另一方面,借助机器学习的智能来提高数据库系统的易用性,使得数据库系统面对多样化的工作负载能够更加智能地动态调整数据库系统的配置,即借助机器学习技术实现数据库系统的自动化.

1 问题定义与研究框架

将机器学习的思想应用于数据库系统设计与实现中,已经取得较好的研究成果.数据库系统中引入机器学习技术已成为当下的研究热点.本节首先分析数据库系统目前面临的挑战;接着阐述机器学习化数据库系统研究问题的定义;最后给出机器学习化数据库系统的研究框架,并对当前的研究工作进行梳理.

1.1 数据库系统面临的挑战

数据库系统已经在行业中成熟而稳定地运行,而大数据时代为数据库系统提出了更高的要求.数据量增大,要求数据库系统能够具备更快的查询速度、更高的系统吞吐量.数据的类型模式不断增多,使得查询工作负载具有快速而又多样化的特点,要求数据库系统能够具备快速、准确地响应工作负载动态变化的能力.接下来从索引结构、查询优化、参数配置来举例说明数据库系统目前面临的挑战.

索引结构作为一种存取模式,对于数据的高效访问至关重要[10],并且索引结构能够满足不同访问的需求.例如 B 树索引最适合范围查询,Hash 索引适合单个关键字的查找.在过去的几十年中,索引被广泛优化,但大多集中在提高内存、缓存或 CPU 执行效率.而传统的数据库系统已进入瓶颈期,优化空间收窄.查询最基本的操作是顺序扫描,其不能够充分利用现代加速器的高并行处理能力.另一方面,传统的索引结构是通用的数据结构,并没有利用数据的分布,现实数据库系统中的数据通常具有常见的模式,因此若已知数据的分布则几乎可以优化所有的数据结构[3].

当下数据库系统的规模不断增大,复杂性增加,处理的数据量越来越大,类型也不断丰富.由此带来了查询工作负载快速而又多样性的变化.查询优化和系统参数配置一直是数据库系统需要解决的问题[7,11].传统的数据库系统,需要大量的时间对特定工作负载进行调优.这些调优基于有经验的 DBA,而当下工作负载快速而又多样性的变化使得数据库系统的适应能力降低.面对不同工作负载,数据库系统的性能差异较大,不能动态地调整为最佳运行时状态.

机器学习为解决这些问题带来了机会.一方面,传统的数据库系统问题,例如索引结构用于建立查找键和数据记录之间的关联[3],对应关键字与位置之间的映射.这些数据之间关键字与存储位置的映射关系,可以作为机器学习的训练集,为机器学习模型的训练提供了先决条件,因此机器学习在数据库系统中的应用具有天然的数据优势.同时,将索引结构转化为机器学习模型后,可以进一步借助硬件加速器来提高数据库系统的处理速度.

另一方面,数据库管理系统[11]存在着参数调优、工作负载预测等问题,而机器学习善于利用已有的数据进行预测,这为机器学习在数据库系统中的应用带来了切入点.查询优化、参数配置长久以来积累了大量 DBA 的调优经验.机器学习能够从查询优化、参数配置等的调优数据中学习到模式,进行快速预测.从而面对快速多变的工作负载,动态地为数据库系统提供最佳的运行配置,使得数据库系统变得更加自动化.机器学习为数据库系统的性能优化提供了机会.

综上,设计并实现机器学习化数据库系统具有重要研究意义.学术界如 CMU,MIT,与工业界如

Oracle,IBM,HUAWEI 等,都在不断探索机器学习化数据库系统的研究.

1.2 问题定义

机器学习化数据库系统泛指结合机器学习的方法、模型来替换数据库系统的内部组件,或将机器学习的算法引入到数据库系统设计中.借助机器学习的加速算法、硬件加速平台来帮助数据库系统实现性能优化,从而突破数据库系统性能瓶颈.

机器学习,特别是深度学习在传统上被认为计算开销巨大、耗费时间,但是随着下一代硬件的不断发展,例如 GPU,TPU 等加速硬件的出现,这些看似是障碍的限制正在逐步地消除,深度学习用于数据库系统不仅在理论上,而且在实践上也变得越来越有可能.数据库系统和机器学习都是面向数据驱动的应用[11],由此看来,2 个领域的结合将极大地推动大数据驱动应用领域的发展.

机器学习化的数据库系统研究带来了机遇,同时也带来了挑战.例如机器学习化的 B 树索引查询速度比传统 B 树索引快 1.5～3 倍,机器学习化的布隆过滤器比传统布隆过滤器的误报率更低[3].但机器学习模型的计算需要时间,当插入新数据时模型的重训练问题需要考虑.再者,基于深度学习的查询优化模型在 10 000 次训练后可以超过传统查询优化器的优化性能[12].但深度学习的模型需要大量训练数据,为达到模型收敛,需要进行大量多次的训练,因此在前期的优化效果较差.

当前的机器学习化研究工作主要针对数据库系统的某一组件,采用机器学习化的模型来替换或辅助数据库系统组件的操作.而面对庞大的数据库管理系统,如果每一组件都附着一个机器学习化的模型,对于数据库系统来说是极大的累赘.众所周知,机器学习的训练需要时间和资源,必然会增加数据库系统的能耗.并且这些模型的重训练、模型更新、模型的统一管理等都值得进一步研究和探讨.在数据库系统中结合机器学习,需要探索这 2 种技术领域的差异,并且必然需要付出极大的努力来解决在机器学习化数据库系统研究进程中的这些差异[1,13].这些问题都促使着数据库领域思考数据库系统的发展以何种方式来促进机器学习与数据库系统的交叉.

由此看来,迎接机器学习化数据库系统发展机遇的同时,也需要面临新的研究问题和挑战.数据库管理系统经过几十年的发展,已经具备自己的独有特点,因此机器学习算法在数据库系统中的应用不是简单的算法搬移,需要针对数据库系统的特点和不同的应用场景进行设计.机器学习化的数据库系统研究需要探索并不断解决数据库系统和机器学习领域的差异,即解决机器学习技术应用于数据库系统的适配性问题.这对数据库领域的研究者提出了新的要求,不仅需要深入了解数据库系统,同时需要掌握机器学习技术的灵活应用,这些都需要数据库研究者进一步探索.

现阶段,在数据库领域已经开始探索利用机器学习来改善性能,机器学习与数据库系统的结合作为一个新的研究领域,受到大家的广泛关注.

1.3 研究框架

数据库系统与机器学习结合的概念在文献[13]中被进一步提出后,机器学习化数据库系统的研究逐渐成为数据库领域广受关注的研究方向.

数据库系统主要包含底层组件、核心组件以及应用层[10,14-15].底层组件主要包括存储在底层的数据库数据、数据库管理系统的配置以及针对数据库系统的硬件加速方案等.核心层组件主要包括查询处理、事务处理、存储管理等,其中存储管理涉及缓冲区管理、内外存交换、索引等存取模式的管理等.应用层用于用户应用程序与数据库之间的交互.

图 2 为数据库系统的整体架构,同时表明目前机器学习化数据库系统研究工作的分布和研究热度.其中白色框表示目前尚未研究基于该组件的机器学习化.本文针对搜集到的文献资料,在分析归纳的基础上得出 2017 年至 2019 年 2 月机器学习化数据库系统研究现状的分布.在近 2 年的 36 篇文献中,针对数据库系统参数配置的研究占到 36%(13 篇);机器学习化的存储管理占 28%(10 篇);针对查询优化机器学习化研究占 19%(7 篇);其他组件的机器学习化研究占 17%(6 篇),主要为应用层和硬件加速解决方案.

从图 2 可以看出,机器学习化数据库系统虽然是目前广受关注的研究领域,但传统数据库系统作为一个发展成熟的领域,其包含的组件数目庞大.目前的机器学习化研究只针对数据库系统中的部分模块,数据库领域仍然有很多方面尚未被涉及.研究最多的领域为机器学习化的参数配置,结合机器学习技术对数据库系统参数进行配置要实现的目标是自动化数据库管理系统.本文主要从存储管理、查询优化、自动化数据库管理系统 3 个方面对数据库系统的机器学习化研究进行阐述.

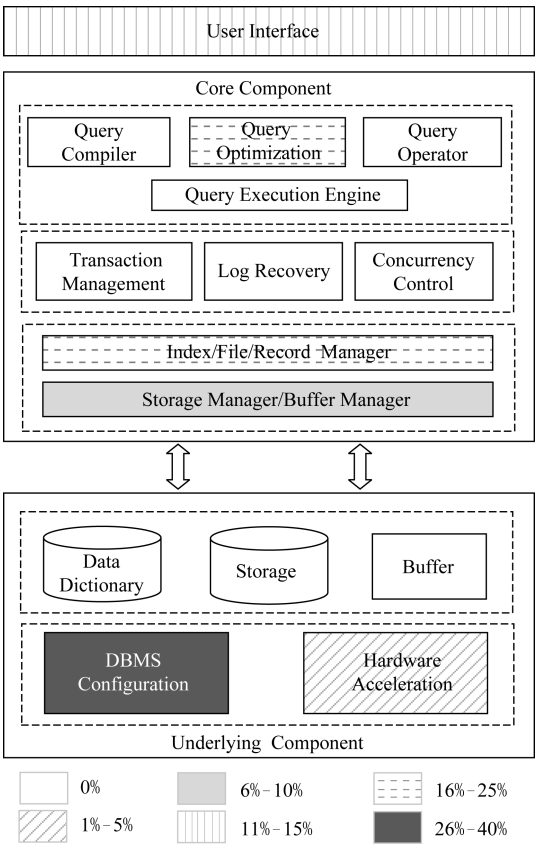

Fig. 2 Heat distribution of research on learned database systems

图 2 机器学习化数据库系统研究热度分布

1) 存储管理的机器学习化研究.存储管理[10]主要实现的功能有:缓冲区管理、内外存交换、外存管理,同时包含对存取模式、存取路径的管理.主存和缓冲区管理器作为数据库系统不可缺少的组成部件,页面置换策略、地址转换等已开始机器学习化研究的探索[16-17].作为数据库系统的重要存取模式,在索引/文件/记录管理方面研究最多的是索引结构,索引作为数据库系统中重要的支持对数据进行高效存取的数据结构,建立了查找关键字与数据记录的关联.Kraska 等人[3]提出机器学习化的索引结构是这一领域的开创性研究.本文第 2 节将介绍存储管理的机器学习化研究.

2) 查询优化的机器学习化研究.查询处理是数据库管理系统(database management system, DBMS)中一个重要的部件集合,能够将用户的查询和数据修改等操作命令通过语法分析,并进行物理优化、逻辑优化等操作,转换为数据库系统的操作序列——查询计划,最终提交给数据库系统执行引擎来执行这些操作[10,15].查询处理目前的机器学习化研究主要集中在查询优化方面.查询编译、查询算子构造、查询执行等目前尚未进行机器学习化研究.根据查询优化器架构可以将查询优化的机器学习化研究分为 3 个方面,分别为连接次序枚举、基数估计、代价模型.本文第 3 节将对查询优化的机器学习化研究进行详细介绍.

3) 自动化数据库管理系统的研究.数据库系统的参数配置对于数据库系统的性能十分重要[8,18],参数优化是任何数据密集型应用程序必须考虑的方面之一.数据库领域的参数调优是长久以来的研究问题[8,18-19],因此该研究方向借助机器学习的方法开展较早,数据库领域无论是学术界还是工业界都在探索实现数据库系统的自动化管理.数据库系统真正地自动化管理应当是系统的所有方面都由一个集成的规划组件来控制,而不是系统中的每一个组件配备一个管理工具.机器学习的发展、硬件的改善以及自适应数据库架构的发展使得自动化管理变得更加可能.Pavlo 等人提出自主数据库管理系统(self-driving database management systems)的概念[20],并构建了第 1 个自主数据库系统 Peloton[21].本文第 4 节将对自动化数据库管理系统的研究进行阐述.

4) 其他组件的机器学习化研究.数据库系统应用层也已经开始机器学习化的探索,例如近似查询处理(approximate query processing, AQP)[22]、数据探索[23-26]等领域.Park 等人[22]提出数据库学习(database learning)的概念,即如果存在极其准确的关于底层数据的统计模型,那么不需要访问底层数据,只需要模型即可.并将 database learning 与 AQP 结合,从过去的近似查询结果中学习,用于改善对底层数据的后验知识.NNCubes[23]将神经网络作为数据立方体技术的替代模型.Cumin 等人[24]采用基于决策树的机器学习方法来自动地构建查询重写.Guilly 等人[25]基于聚类和二叉决策树的机器学习算法提出针对数据探索的 SQL 查询完成算法(SQL query completion).Zolaktaf[26]采用特定领域的推荐系统来帮助关系数据库环境下的数据探索和查询构建.Martins[27]提出一种半自动化的方法——智能符号机器(intelligent semiotic machine, iSM),该方法考虑了用户的角度和目的,从而交互地生成定制查询,促进个性化的数据探索.应用层位于数据库系统的外层,本文不做过多介绍.

用硬件来加速数据库系统也是比较热的研究方向,基于 GPU 加速的数据库系统层出不穷,例如

OmniSci Core[28], Kinetica[29], BlazingSQL[30], Blazegraph[31], Pg-strom[32]等.同时基于可编程门阵列(field programmable gate arrays, FPGA)的硬件加速解决方案也有很多研究成果,如 Centaur[33], LINQits[34]等.上述基于 FPGA 的研究工作大多集中在加速 DBMS 的操作上,Mahajan 等人[35]迈出了应用 FPGA 来加速数据库内高级分析解决方案的第 1 步.

机器学习算法不断改进、新硬件设计不断出现,这些领域的进步都推动了数据革命.可编程加速器、现代加速器等正在进入数据驱动的应用领域[36-37],同时人工智能芯片的研究[38-39]也不断深入.在数据库领域实现更好的硬件加速解决方案,促进机器学习化数据库系统的研究,以搭上现代硬件发展的快车,是值得未来研究的课题之一.

任何一个查询或修改动作都可视为事务处理,事务处理包括并发控制、日志恢复、事务管理等,对数据库系统至关重要.作为数据库系统核心组件之一的事务处理,目前尚未进行机器学习化研究.事务是组成一组的若干个查询和其他动作[10],是独立的原子单位,事务的执行需满足 4 个特性:原子性、一致性、隔离性、持久性.事务不涉及对数据的直接存取,同时事务的执行过程要求可追溯、可解释.机器学习化的模型建立在数据基础之上;某些机器学习模型尤其深度学习模型的可解释性较差.因此事务处理模块的研究不具备直接应用机器学习技术的天然条件——数据.针对事务处理模块中的并发控制、故障恢复等,是否可以从整体考虑,提前预测超时、死锁等影响系统执行性能的瓶颈,从而进行优化?对事务处理、并发控制、故障恢复等进行基于机器学习模型的优化研究,有待进一步思考.同时针对机器学习化数据库系统的可解释性研究也值得探讨.

数据库管理系统作为成熟的商业化产品,是一个庞大的研究领域.现代数据库系统已经超越了传统的关系型 DBMS 研究领域,包括在数据库系统之上构建搜索引擎,进行数据集成、数据挖掘等复杂应用程序.机器学习化的数据库系统同样也面临着庞大的研究领域,就目前的研究工作来看,仅仅是整个数据库领域中极小的部分,未来需要探索的方面还有很多.

本文针对当前机器学习化数据库系统已有的研究工作进行归纳总结,从 3 个角度对机器学习化数据库系统的研究工作进行划分:1)存储管理的机器学习化研究,主要从索引结构和缓冲区管理 2 方面来介绍;2)查询优化的机器学习化研究;3)自动化数据库管理系统的研究.

2 机器学习化的存储管理

存储管理主要实现的功能:缓冲区管理、内外存交换、外存管理,同时包含对存取模式、存取路径的管理[17].索引结构作为数据库系统中重要的存取模式,支持对数据进行高效存取,建立了查找关键字与数据记录之间的关联.基于索引结构的机器学习化研究工作层出不穷.主存和缓冲区管理器作为数据库系统不可缺少的组成部件,页面置换策略、地址转换等已经开始机器学习化的探索[16-17].本节从索引结构和缓冲区管理 2 个方面对机器学习化的存储管理进行介绍.

2.1 机器学习化的索引结构

在数据库系统中,有效的数据访问查询离不开索引结构.数据库系统中存在多种索引结构来满足不同的访问模式需求[10,14].B 树索引最适用于范围请求;对于单个关键字的查找,Hash 索引的效果最好;布隆过滤器适用于判断某个记录是否存在.

索引作为一种存取模式,对数据库系统具有重要作用,能够加速一个或多个属性上特定值的查询[10,40].图 3 为传统数据库系统的索引结构[10,40].索引是一种数据结构,能够以一个或多个字段的值作为输入,并"迅速地"找到其对应的数据项——具有该值的记录.

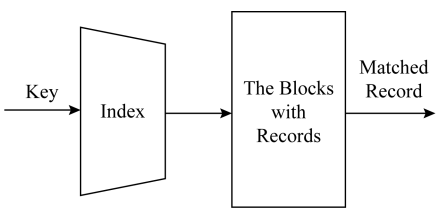

Fig. 3 The traditional index structure of database

图 3 传统数据库索引结构

索引结构对于数据库系统至关重要,因此对于索引结构的优化一直是数据库领域的研究重点.传统数据库系统的索引结构是通用型的数据结构,对底层数据的分布不采取任何假设,而现实中很多场景下数据的分布具有规律性,因此可以使用机器学习方法对索引结构进行建模[3].在索引结构中,数据表中所有的〈*key*, *position*〉对,自然地构成了机器学习中的训练数据,如果我们可以知道数据的准确分布,那么几乎可以优化所有的索引结构.Kraska 等人[3]为该领域的研究做出开创性工作,机器学习

方法通过学习来反映数据的模式,因此使用学习到的模型来拟合索引结构,提出了学习化的索引结构(learned index structures).

Kraska 等人[3]将索引结构视为模型,B 树索引视为查找关键字在有序数组中的位置;Hash 索引模型是在一组无序数据中查找关键字的位置;位图索引模型返回在一组数据中是否存在某查找关键字.学习化索引的关键在于训练一个模型,能够学习到关键字的排序位置或结构,基于模型来有效地预测数据记录的位置和存在性.

B 树模型和 Hash 模型都可归纳为:针对给定关键字,模型返回该关键字位置的问题,即 $position=F(key)$,同时 B 树模型还适用于范围索引.机器学习化的布隆过滤器(learned Bloom filters)将关键字是否存在视为一个分类问题,为减少假负类样本的出现,Kraska 等人提出在传统布隆过滤器前使用一个预滤器(pre-filter)——神经网络,用于评估某元素出现在集合中的概率,即训练模型 $f(k)$ 满足:$f(k)=\begin{cases}1, & k\in K\\ 0, & k\notin K\end{cases}$,从而在预滤器之后可以使用一个较小的布隆过滤器;另一种机器学习化的布隆过滤器是用机器学习化的 Hash 函数来替换传统布隆过滤器中的 Hash 函数.Reichenbach 在他的学位论文[41]中评估和比较了学习化索引与传统索引的性能.

Mitzenmacher 在学习化索引[3]的基础上,针对学习化的布隆过滤器在理论上构建了更加形式化的模型[42].在正类实例集合 P 和负类实例集合 N 上的学习化布隆过滤器包含:1)带有阈值 τ 的函数 f:$U\to[0,1]$;2)一个相关的标准布隆过滤器 B,称之为后备过滤器(backup filter).其中 U 表示全体可能的查询关键字集合,理想状态下,$U=P\cup N$.后备过滤器用于处理集合 $\{z\mid f(z)<\tau\}$ 中的关键字.对于一个查询 y,学习化布隆过滤器返回“真”的条件为:1)$f(y)\geqslant\tau$ 或 2)$f(y)<\tau$ 但后备过滤器返回 $y\in K$,否则学习化的布隆过滤器返回 $y\notin K$.该研究工作指出学习化布隆过滤器模型的优势以及该模型不适用的场景.在后续的研究工作[43]中,为同时减少假正类和假负类,提出采用“夹心布隆过滤器”,即在学习化函数 f 的前后分别配有一个传统的布隆过滤器.

另外内存索引的创建或修改是十分耗时的过程,通常情况下,当参数更改时支持索引的数据结构通常需要从头创建.Darshana 等人[44]提出自适应索引(self-adapting index, SAI)的方法,在后台不断地进行动态小规模增量重组,是一种“即时”索引重组方式,避免在重组索引时,由索引不可用带来的数据库性能的下降问题.目前也已经开始探索将学习化的索引用于倒排索引.Oosterhuis[45]探索将学习化索引结构用于索引压缩的潜力,以及应用学习化索引来支持基于布尔交叉的搜索;Pavo[46]是基于 RNN 的倒排索引,利用分层神经网络来模拟 Hash 函数,该工作还得出一个有趣的结论:同时构建有监督和无监督的学习策略,实验结果证明无监督的学习策略更好.在目前大数据时代,如何设计有效灵活的数据组织方式改善传统倒排表的空间利用率一直是需要解决的问题.学习化的模型应用于倒排索引方面具有巨大的潜力,考虑不同的数据分布、探索更智能的倒排索引,是未来有前途的研究领域[46].

依据目前研究机器学习化索引结构的文献,表 1 针对学习化的索引与传统的索引结构进行比较.

Table 1 Comparison Between Learned Index and Traditional Index

表 1 学习化索引与传统索引比较及研究工作

Items	Comparations with Traditional Index	References	Applicable Scenario
Learned B-tree Index	Average speed is 1.5~3x faster than traditional indexes; space usage is 2 orders of magnitude less.	Ref [3,42]	Model the distribution of specific data; read-only workloads; the model will retrain if a write operation occurs.
Learned Hash Index	The conflict reduction is up to 77%; but the average lookup time is 1.5x.	Ref [3,43-44]	
Learned Bloom Filter	Space usage reduces about 30%; need negative samples; the prediction includes false negative classes.	Ref [3,43-44]	

学习化索引使用简单的神经网络来替代数据库系统中传统的索引结构.开始了通过机器学习方法来构建学习化模型,并用来替换数据库管理系统核心组件的研究工作.但目前学习化索引结构的研究工作通常假设数据库系统中底层数据分布是静态的,即针对只读型(read-only)数据库系统,不支持数据的更新.当底层数据分布发生变化时,学习化的模型需要重新训练.随着硬件资源的不断发展、计算能力的不断提升,未来这一思想将会对数据库系统的设计产生深远影响.同时该研究工作也带来了新的

研究问题:1)针对 B 树索引结构中频繁插入、删除或更新场景下的模型重训练问题,需要进一步解决;2)针对布隆过滤器在训练时数据库中不存在负类数据的情形,需要进一步探讨;3)针对 Hash 索引模型中使用模型预测的计算开销相对传统的 Hash 索引开销较大,如何设计既避免冲突,同时又快速预测的学习化 Hash 函数,值得进一步研究.

2.2 机器学习化的缓冲区管理

缓冲区管理,是当出现缺页中断时,如果有空闲页,则直接读入要访问的页面;如果不存在空闲页,则采用缓冲区淘汰策略[40],进行页的换出和换入操作.常用的页面置换算法有:FIFO,LRU,ClOCK 等.

传统的在线学习算法将决策制定封装在不确定性下,为应对所有可能的未来事件提供了方法,并保证提供近似最优的方案.而在机器学习中,使用从数据中发现的模式预测未来,通常是最小化误差.因此 Lykouris 等人[16]提出将在线算法与机器学习预测相结合的通用框架——提供机器学习建议的在线模型(online with learned advice model, OLAM).采用机器学习模型来增强在线学习的能力,并将该方法用于传统的缓存问题,为缓冲区制定淘汰策略,即具有机器学习建议的竞争性缓存.实验证明,即便是使用简单的预测器,Lykouris 提出的算法相比 LRU 策略都有所改进.

在缓冲区管理中,如何结合页面的历史使用情况以及预测到的工作负载,不断收集新的页面使用信息,同时反馈页面置换的性能,预测哪些页需要换入、哪些页需要换出,从而进一步减少缺页中断,是未来值得研究的问题.

为改进用户体验,许多现代数据中心的服务都在争取更大的内存容量,这给虚拟内存子系统带来了很大的压力.大量的内存数据集是许多服务器应用程序的主要特性之一,包括数据库系统、键值存储和数据分析框架.传统的软件和硬件虚拟内存机制已经受到限制,Margaritov 等人在机器学习的推动下探索新的方法.受机器学习化索引[3]的启发,提出学习化的页表索引(learned page table indexes)[17],是一种基于学习模型的神经网络地址转换机制,同时该论文指出要进一步构建神经页表索引器.

在存储管理方面的研究工作还有学习化的内存存取模式[47],提出了基于长短期记忆网络(long-short term memory, LSTM)的预取模型,将预取问题视为基于序列的预测问题,为计算机体系结构的研究中嵌入机器学习方法开辟了新方向.目前对于计算机硬件体系结构的机器学习化研究还较少.

3 机器学习化的查询优化

传统的数据库领域在查询优化方面已经做了很多努力[47-48],查询优化是为计算关系表达式选择最有效且代价最小的查询计划的过程.

图 4 为传统的查询优化器架构[48],首先枚举所有可能的查询计划集合的部分子集,采用启发式的规则进行基数估计,并输入到代价模型中,选择语义上相等但代价最小的执行计划.

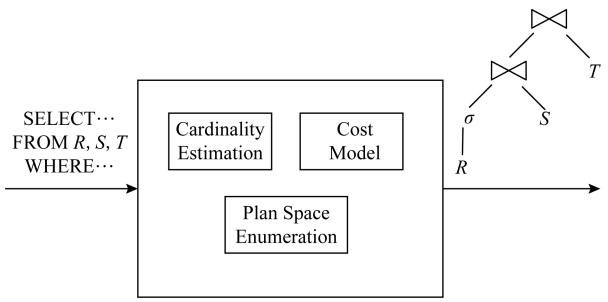

Fig. 4 The architecture of traditional query optimizer

图 4 传统查询优化器架构

根据查询优化器架构可以将查询优化的机器学习化研究分为 3 个方面,分别为连接次序枚举、基数估计、代价模型.因此本节从这 3 个方面阐述机器学习化查询优化的现有研究工作.

3.1 机器学习化的连接次序枚举

在关系数据库中,连接次序的选择对于查询性能具有重要影响[48],并且一直是数据库系统中广泛研究的问题之一.传统的查询优化器通常使用静态连接次序枚举算法,这些算法不包含关于产生的查询计划质量好与坏的反馈,因此优化器通常会重复选择同样差的查询计划.连接次序选择的主要挑战是通过枚举候选空间找到代价最小的连接次序,因此枚举器需要在最小化枚举候选空间的同时,能够找到代价最小的连接次序.大多数数据库系统,由数据库管理员来控制候选计划空间的大小[47],例如只在左深连接子树中选择,或者在一段时间之后终止枚举,在枚举时一般采用动态规划[47]或贪心算法[49].

深度强化学习(deep reinforcement learning, DRL)[50]正在迅速改变着人工智能领域.深度强化学习能够对模型和环境有更高层次的理解,使模型能够学习更复杂的动态任务.Ortiz 等人[51]探索将 DRL 用于解决查询优化的问题,通过训练深度学习模型来预测查询计划的基数估计,主要解决查询优化领域

的状态表示和状态转换函数的构造问题. Ortiz 等人训练模型来学习并逐步生成每一个子查询中间结果的简洁表示,模型以当前子查询和新的操作符作为输入,预测产生下一个子查询的特性,该特性用于计算该子查询的基数.同时基于学习化的表示(learned representation),使用强化学习改进查询计划枚举,将其视为一个马尔可夫过程来逐步构建查询计划.针对传统查询优化器不能从先前经验中学习这一问题,Marcus 等人[12]认为深度强化学习可用于改善此问题,基于人工神经网络的深度强化学习能够整合反馈来自动改进查询优化器的决策.文献[12]将深度强化学习用于连接次序枚举,提出 ReJOIN. ReJOIN 只考虑连接次序问题,不执行索引选择、连接操作符选择,这些都交给 DBMS 传统优化器的其他部分去执行,因此 ReJOIN 不是端到端的查询优化器.ReJOIN 将连接次序枚举视为一个强化学习过程,通过神经网络来不断地从反馈中学习,在 ReJOIN 中使用基于传统优化器的代价模型作为强化学习中的奖励信号(reward signal).

Marcus 等人[52]在 ReJOIN 的基础上,提出免调节(hand-free)的查询优化器,基于深度学习的方法,实现端到端的查询优化器.包含 3 种可行的深度学习查询优化框架:1)示范学习[53],训练模型来模仿传统的已经调节好的查询优化器;2)引导式代价模型,初始使用传统代价模型作为奖励信号,并记录查询的执行时间,当模型收敛后使用查询时间作为奖励信号;3)渐进式学习[54-55],将查询优化任务分解为较小的多个任务,通过划分任务使得搜索空间变得可管理.由此看来 Marcus 等人提出的方法并非真正的"免调节",示范学习、引导式代价模型、渐进式学习在一定程度上都依赖于已经调节好的查询优化器或代价模型.类似的研究工作还有 Krishnan 等人[56]基于深度强化学习提出的查询优化器 DQ.

3.2 机器学习化的基数估计

基数估计一般基于对数据的均匀性、一致性、独立性等假设,实际的数据集中这些假设大多不能成立.在大数据系统中,基数估计对于云共享基础设施极为重要,例如谷歌的 BigQuery、亚马逊的 Athena、微软的 Azure Data Lake 等[57].云共享系统中,基数估计的不正确性不仅导致优化性能差,还决定现代共享云基础设施中所消耗的资源.基于机器学习的 CARDLEARNER[57]比默认的基数估计器可精确高达 5 个数量级.CARDLEARNER 从先前的作业执行中学习基数估计模型,并使用模型来预测未来作业的基数.该研究工作已经集成到 SCOPE 查询优化器中.针对传统数据库的基数估计问题,Kiefer 等人采用传统的机器学习方法,构建基于核密度估计(kernel density estimation, KDE)的基数估计模型[58-59],实现比现有方法更好的性能.Kipf 等人[6]还提出学习化基数(learned cardinalities)的概念.采用神经网络,将基数估计视为一个有监督的问题,将用户查询和该查询的基数进行特征化作为模型输入,模型输出为基数的估计值.QuickSel[60]针对基于代价模型的查询优化器中估计查询的选择性问题,提出了选择学习框架(selectivity learning framework).以上研究工作大多只考虑查询优化器的基数估计部分,其他任务留给传统的优化器执行.目前基于机器学习的基数估计研究,已经对数据库的性能产生良好的影响,同时提供了新思路,未来值得探究更加合适的基于机器学习的新方法,具有较大的研究空间.

3.3 机器学习化的代价模型

在数据库系统性能调优过程,特别是查询优化过程中,代价预测模型对于优化效果极为重要. 现有的研究大多依赖于对数据的简单假设,利用系统的统计信息,并没有真正预测查询的执行时间.

Ganapathi 等人[61]提出使用机器学习来精确预测数据库查询的性能指标.在一组查询的训练集上发现查询属性和查询性能指标之间的多元关联,并使用这些关联的统计关系来预测新查询的性能.该方法丢失了查询计划的结构信息.毕里缘等人[62]针对查询优化器的查询开销预测问题,使用 LSTM 来预测查询开销,首先采用后序编码将查询语法树转换为一个长度为 5 的操作序列,将其输入到 LSTM 网络中,进行查询开销预测. 该方法以查询计划中的操作行为和实际运行的时间作为神经网络提取特征的来源.该模型最后输出该查询计划的预测开销序列,并且在实际执行查询计划前,能够产生对该计划实际运行时间的预测.

表 2 为针对现有机器学习化的查询优化研究工作的总结,主要针对查询优化器中的 3 个部分:连接次序枚举、基数估计、代价模型,同时也有一些工作是针对整体查询优化器进行优化.

机器学习化的查询优化器,基于深度强化学习,能够从先前的错误中学习,自动改进查询优化器的决策,对数据库系统的查询优化研究具有开创性的进步.在针对查询优化中代价预测的问题,以查询计划中的执行时间为代价预测标准,值得进一步探讨,后续依然有很多工作值得做.

Table 2 Query Optimization with Machine Learning

表 2 机器学习化查询优化器

Items	Methods Based on Traditional Machine Learning	Methods Based on Deep Learning
Cardinality Estimation	KDE[58]; QuickSel[60]; Bandwidth-Optimized KDE[59]; CARDLEARER[57]	Learned Cardinalities[60]
Cost Model	Ganapathi's cost predition[61]	Bi's Cost Model[62]
Plan Space Enumeration		Deep RL-based DQ[56]; ReJOIN[12]
Query Optimizer		Ortiz's Model Using DRL[51]; Hands-Free Query Optimizer[52]

未来机器学习化的查询优化对于数据库系统的设计、性能提升都会产生深远影响.同时该研究工作也带来新的挑战:1)深度强化学习需要大量的训练数据,在训练初期数据库查询优化的性能很差;2)深度强化学习通常假设奖励信号很容易获得,而查询优化最本质的性能指标是查询延时,实际中计算代价太大;3)传统的代价模型是建立在不实际的数据集特性假设之上的,不能够真正反映查询优化的性能.以查询计划中的操作行为和实际运行的时间作为特征的来源,并以此来预测查询代价,构建以此为代价模型的学习化查询优化器值得进一步探讨.因此针对数据库系统的查询优化,结合深度强化学习等其他机器学习方法仍然有很多值得研究的工作.

4 自动化数据库管理系统

在数据库领域,参数的调优是长久以来的研究问题,同时也是难题[18-19].参数配置主要面临着 3 个难题:1)参数名称不标准,不同的数据库系统在同一个参数上可能具有不同的名称;2)参数功能不独立,改变一个参数会影响其他参数的配置性能;3)参数调优适用范围不具通用性,在一个应用上优化好的参数,在另一个系统上一般不适用[7].

优化 DBMS 的参数对系统性能至关重要.DBMS 的默认配置通常差强人意,随着数据库系统的发展,现代 DBMS 拥有大量参数[63-65].数据库系统和应用程序的规模不断增大,复杂性不断上升,良好的 DBMS 配置很多时候依赖于 DBA 所无法知道和预料的因素,优化 DBMS 的参数已经超过人类能力[7].

实现数据库系统的自动化管理是一个长期的研究过程.数据库系统借鉴自动化、机器学习等方法来实现参数配置,已有很多探索.表 3 为自动化数据库管理系统研究的 3 个阶段及其主要特征,主要依据为这 3 个阶段的研究工作以及文献[66]的总结.

依据自动化数据库系统研究的侧重点不同,可以分为 3 个阶段:自适应数据库系统,自调节数据库系统,以及云计算出现后的自主数据库系统[66].

本节介绍自动化数据库管理系统(automatic database management system)近 50 年来的研究历程.主要从云计算出现前后介绍自动化数据库管理系统的研究.云计算出现后,由于云平台的大规模和复杂性,使得数据库系统的自动化研究更加迫切.

Table 3 The Development History on Automatic Database Management System

表 3 自动化数据库管理系统研究历程

Items	Age	Main Topics
Self-adaptive Database	1970s~1990s	Index Selection, Database Partition
Self-tuning Database	1990s~2000s	DBA Advisor Tools, Knobs Tuning
Self-driving Database	2010s~now	Workload Forecast, System Configuration(Physical Design, Data Organization, Optimization)

4.1 自适应和自调节数据系统

1970~1999 期间,自适应数据库(self-adaptive database)系统主要集中于数据库系统的物理设计[66],尤其在索引选择[67-68]、数据分割[69-70]方面.这些早期的工作奠定了现代数据库系统调优工作的基础:1)系统收集关于应用程序如何访问数据的评价指标;2)根据代价模型搜索需要进行哪些更改来提高系统性能. Hammer 等人提出在自适应数据库系统中进行自动化索引选择[67].数据库自动化分区和数据放置的需求在 20 世纪 90 年代更加突出,出现

分布式和并行数据库,代表性研究工作有文献[71],提出根据数据模式和工作负载特性生成最优分区策略的集成数据放置算法.

针对自调节数据库(self-tuning database)系统的研究集中在1990~2009年,是自动化数据库研究的第2个浪潮.Self-Tuning数据库系统[72],构建用于帮助数据库管理员的咨询工具,针对不同的工作负载,选择最优的索引、物化视图、分区模式.代表性的系统有微软的AutoAdmin[73]、IBM的DB2 Designer[74]等.

21世纪初同时也开始了针对自动化配置参数调优的研究.几乎所有的数据库供应商都拥有自己的参数调优工具[46,75].这些都依赖于数据库管理员来决定这些优化是否正确并且何时部署这些配置.代表性的研究工作有IBM为DB2构建自动化组件(autonomic components),例如LEO[76],SASH[77]等.

在自动化调优DBMS参数方面已有较多研究工作.Narayanan等人[78]提出资源顾问(resource advisor),该方法针对于SQL Server,基于细粒度、低开销的数据库性能追踪,能够自动回答资源的"假设"(what-if)问题,同时准确地预测联机事务处理(online transaction processing, OLTP)工作负载性能的变化,能够更好地理解系统性能. Dias等人[63]提供一种执行自动化性能诊断和调优的方法,设计了自动数据库监视器(automatic database diagnostic monitor, ADDM),来自动诊断影响数据库总吞吐量的瓶颈,提供可操作的建议来缓解瓶颈,并应用于Oracle 10g[79]中.Sullivan等人[80]使用被称为影响图(influence diagram)的概率推理模型来实现有效的自动化软件调优方法.Tran等人[81]提出一种基于缓冲区丢失方程的缓冲区优化方法,将可用数据与丢失方程拟合进行优化计算. Duan等人[82]提出自动化推荐数据库配置参数的工具iTuned,结合自适应采样技术使用有计划的实验来发现影响较大和性能高的参数进行设置.

以上这些自动化参数调优工具大都存在缺陷,导致其不能够适用于一般用途的数据库系统应用程序.大多是由数据库供应商提供,只针对特定的数据库系统,少部分工具适用于多个数据库系统但依然需要人工手动配置的步骤.例如需要映射配置参数之间的关系[80]、需要引导训练的过程[81]、需要部署数据库的第2个副本[82]等.并且这些工具都独立地检查每个DBMS的部署,不能从先前的参数调优工作中获取经验,因此每次参数配置工作都耗费大量的时间和资源.

2010~2019年的早期,伴随着云计算的出现,由于云平台的大规模和复杂性[83],针对数据库系统的自动化研究更加迫切.在此期间,云服务提供商开发自定义工具来部署数据库系统.例如,微软的Azure服务模型利用DBMS容器的资源从而自动调整资源的分配[84],以满足服务质量和资源预算约束.

4.2 自主数据库系统

卡耐基梅隆大学的Pavlo教授指出,现在是自主数据库(self-driving database)的时代,并指出一个真正的自主数据库管理系统[66]应当具备3方面能力:1)自主地决定采用何种动作来对系统进行优化;2)自主决定何时部署优化动作;3)自主地从优化动作与性能改善中学习.所有的能力应该都不需要DBA的交互,自主数据库可称为自动化数据库系统发展的第3个阶段.

自适应和自调节数据库系统的研究工作不能称之为完整的自动化数据库管理系统[21],原因有3方面:1)先前的自动化工具都是作为数据库系统的外部组件使用;2)这些组件在数据库系统发生问题时给予DBA调优建议,最终需要DBA来决定是否采纳建议,并决定何时部署调优动作;3)这些工具在调节时,对数据库系统的各个方面没有整体的考虑,往往仅考虑数据库系统性能的某一方面.即便是这些系统工具是自动化的,可以自行部署优化,现有的DBMS体系结构也不能支持重大的更改,如果不进一步改进系统,这些自动化的工具也无法适应未来的瓶颈.

Peloton是第1个自主数据库管理系统[20-21],该系统集成了关于工作负载预测和数据库配置动作部署的深度学习框架.并指出self-driving database能够支持的3类优化:1)数据库的物理设计(索引、物化视图、存储布局等);2)数据组织的改变(位置、数据分割等);3)影响数据库系统执行时间的配置(资源、配置调优、查询优化). Peloton作为第1个自动化的内存数据库系统,能够逐步地对数据库系统进行优化,在部署期间不会对应用程序造成明显的影响.Ma等人在文献[85]中,针对"self-driving" DBMS——Peloton的工作负载预测框架进行了详细阐述,称之为QueryBot5000.

Aken等人[7]提出OtterTune,采用自动化的方法来调节数据库系统的配置.结合了有监督和无监督的机器学习方法,利用过去的经验,首先识别主要具有影响力的配置参数(knobs),为借鉴先前工作负

载调优的经验,OtterTune 将未知的或将要处理的工作负载与已有的工作负载相匹配,基于保存的调优经验数据,为提高 DBMS 的某一特定目标(例如 latency, throughput 等),推荐优化的 DBMS 参数. OtterTune 假设拥有数据库系统的各种权限,包括重启数据库系统.

针对云托管的多租户数据库服务(cloud-hosted multi-tenant database services),Jain 等人提出与数据库系统无关的工作负载管理和分析服务架构 Querc[86]. Querc 通过挖掘和管理大规模、异构的工作负载,将工作负载管理和分析视为一组查询标记任务的模型,因此需要大量的训练数据才能有效.随着工作负载复杂性增加、负载规模增大、负载管理需求多样化,云托管的数据库服务十分需要自动化的方法来对用户行为模式、工作负载进行分析,从而进行资源配置、路由查询等.因此该领域具有理论和实际的研究价值.

SageDB[87]指出现在的数据库系统大多是为处理不同的数据模式、数据类型、数据分布等而设计的通用型系统.这类系统没有利用特定应用场景和特定数据分布特征的优势.事实上,如果能够知道数据的特定分布,那么几乎可以优化所有系统[3].因此 Kraska 提出一种新数据库系统——SageDB,旨在通过代码合成与机器学习的优越性设计并优化特定应用场景下的数据库系统.主要使用机器学习对数据分布、工作负载、硬件等进行建模,确定特定数据库系统的数据结构、最佳存取路径及查询计划等优化目标.同时指出,学习化的组件能够完全替代数据库系统的核心部分.

表 4 对自主数据库系统的部分研究工作进行简要比较和总结,包括 Peloton[20-21], OtterTune[7], Querc[86], SageDB[87],主要从这些研究工作所采用的机器学习方法、涉及的数据库系统组件或功能及这些研究工作的特点出发进行总结.

Table 4 Research Work on Self-Driving Database Management Systems

表 4 自主数据库管理系统研究工作总结

Items	Machine Learning Methods	Database Components & Functions	Characteristic
Peloton	Unsupervised learning methods (DBSCAN); deep learning methods (RNNs)	Workload classification & forecasting; action planning & execution	Operate without DBA guidance; the main optimization objective is latency; in-memory database; optimization for HTAP
OtterTune	Supervised and unsupervised machine learning methods (factor analysis, K-means clustering, Lasso regression)	DBMS configuration	Modeling configuration data; operate without DBA guidance; assume the controller has administrative privileges of DBMS (including restarting the DBMS)
Querc	Learned vector representations (context prediction models, LSTM AutoEncoders)	Query workloads management	Database-agnostic workload management system; obviate feature engineering; for cloud services, multi-tenant and multi-database platforms
SageDB (vision paper)	CDF model; deep neural nets et al	Argue that learned components can fully replace core components of a database	A learned database; for the specific workload and data characteristics of users; presents radical departure from developed way of currently database

在自动化数据库领域的研究工作还包括:Deep Tune DB[88],主要探索强化学习在数据库物理设计中的应用;DASlab 实验室开发的自动设计数据库系统(self-designing data systems)[89-91],快速生成给定应用程序的最优解决方案,self-designing data systems 将减轻系统设计人员和最终用户在数据管理方面的烦恼,最终提高生产率;Stratos 等人提出了自动化数据结构设计的概念[92];DBSeer[93]记录表示数据库系统性能的时间序列数据,并进行比较来帮助数据库管理员诊断系统中运行缓慢的区域和正常运行的区域.Kossmann 指出未来的自动化数据库系统将利用基于工作负载驱动的优化和机器学习技术来生成对未来工作负载的预测,并决定如何选择操作来最优地处理工作负载,同时能够从过去的经验中学习[94].

工业界在自动化数据库系统方面也在不断地探索.Oracle 在 2017 年 9 月发布了 Oracle 自治数据库[95-97],基于云环境的自治数据库,利用机器学习来实现自动化,能够自动地执行查询优化和参数调优等.但针对如何预测未来工作负载,并基于预测对数据库系统进行调优,Oracle 没有给出方案[66].华为在 2019 年 5 月发布 AI-Native 数据库 GaussDB[98-99],在数据库系统中引入人工智能技术,将 AI 嵌入到数据库系统的生命周期,实现数据库系统的自运维、自管理、自调优和故障自诊断.GaussDB 的另一大性能优势是支持异构计算.

同时,Pavlo[66]认为“是否能够拥有一个完全自动化的DBMS——能够在所有可能的工作负载中达到与人工维护的DBMS相同或更好的性能?答案是肯定的,但短时间内还肯定不会.”

因此自动化数据库系统依然有很多值得研究的机遇和挑战,任重而道远,十分具有研究前景.

5 未来研究问题与挑战

如前所述,机器学习化数据库系统泛指结合机器学习的方法、模型来替换数据库系统的内部组件,采用机器学习的方法来帮助数据库系统实现性能优化,并提高数据库系统的易用性,实现数据库系统的动态、智能配置.因此未来的机器学习化数据库系统不单单是对系统内部组件的替换,在数据库系统机器学习化的发展中,将会更多使数据库系统向着更加自动化、智能化的方向发展.

据此提出自动化数据库管理系统(autonomous database management system)框架,如图5所示.不仅包含机器学习化数据库系统,同时包含采用机器学习的方法对数据库系统进行智能管理.

首先在数据库系统内部将会存在各种学习化的模型来替换传统数据库系统组件,例如机器学习化的查询优化器、索引结构等.针对机器学习化的数据库系统,必须有统一的模型管理器(unified model manager),负责模型的更新、维护等工作.其次作为自动化数据库管理系统,工作负载预测模型(workload prediction model)要具备对未来工作负载预测的能力,从而对数据库系统进行动态地配置调优.对数据库系统工作负载预测能力的评估需要采集数据库系统的性能,因此性能监控器(performance monitor)用来监控数据库系统的性能,并将性能数据传递给工作负载预测模型作为训练数据.

Pavlo[66]指出,将来的“self-driving” DBMS,同样需要人类能够部署一些优化,就像数据库系统的助理一样.因此我们认为在自动化数据库管理系统中,依然需要部署针对特定应用领域的启发式规则和约束条件.

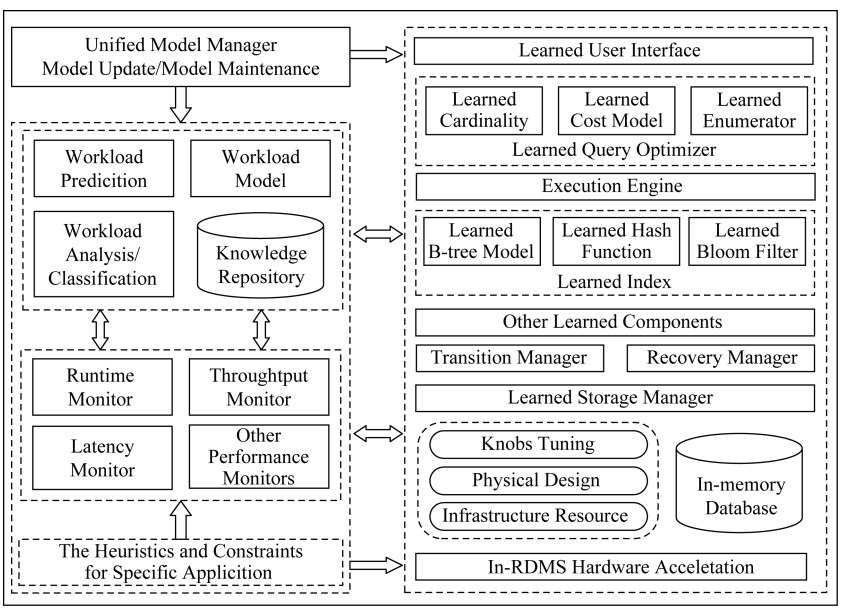

Fig. 5 The architecture of autonomous database management system

图5 自动化数据库管理系统框架

在数据库系统中结合机器学习等新技术,我们认为以下7个方面是机器学习化数据库系统未来的主要研究问题与挑战:

1) 机器学习化数据库系统的模型管理问题.从当前的研究工作可以看出,大多是针对数据库系统的某一方面,采用机器学习化的模型来替换数据库系统的内部组件或采用机器学习化的模型来辅助数据库系统工作.而面对庞大的数据库管理系统,如果每一组件都附着一个机器学习模型,对数据库系统的访问将变成对模型的访问,而不需要访问底层数据.那么数据库管理系统将变成“模型库管理系统”(model base management system, MBMS).模型的统一管理中,模型的训练、更新、维护等是否会增加数据库系统的能耗?另一方面,深度学习模型具有众多

参数需要调节,是否会增加数据库系统需要配置的参数量,从而带来系统配置的“参数爆炸”问题.模型之间的通信、模型之间如何配合、模型的参数配置以及模型的更新维护等都是亟待解决的问题.

2) 系统化的机器学习化数据库系统研究.现有的数据库系统机器学习化研究主要集中在替换数据库系统中的单个组件,这些附着到数据库系统的每一个模型都需要考虑其模型构建、重新训练等问题.对于庞大的数据库系统,如果每一个组件都带有一个机器学习化模型,势必会增加系统的负担.数据库系统组件之间不是单纯的线性叠加,因此学习化后的组件与其他组件之间非线性的影响关系需要考虑.因此,我们提出要以系统的观念来构建机器学习化的数据库系统. 从系统的角度考虑机器学习技术与数据库系统的结合,构建整体的机器学习化数据库系统.

3) 机器学习化数据库系统的性能瓶颈.现有数据库系统的性能瓶颈集中在 CPU 上,那么当数据库系统被机器学习化后,性能瓶颈是否会发生变化?答案是肯定的.针对学习化的索引,相对传统的索引结构性能提升 3 倍[2],假设所有的性能提升都来自 CPU,那么学习化后的数据库系统执行查询所占用的时间比为:CPU 占 42%,硬盘占 34%,内存占 24%.由此可见机器学习化的数据库系统,如果要继续优化,瓶颈不单单是 CPU,硬盘和内存也会成为瓶颈.只有了解机器学习化数据库系统的性能瓶颈,才能够进一步对系统实施优化.未来数据库系统除了从算法、模型层面改进,新硬件是不能忽略的重要因素.

4) 基于新硬件的机器学习化数据库系统.机器学习化的数据库系统,面临着从数据密集型到计算密集型的转变.随着在数据库系统中引入机器学习技术的不断深入,未来机器学习化数据库系统的部署环境必定需要有配套的新硬件环境来支持.同时将现有的数据库系统算法迁移到现代加速器中,需要对系统算法进行重构和改写.如何有效利用硬件资源对数据库系统提出了新的挑战,利用新的硬件特性及技术对大数据进行高效管理具有重要意义.伴随着计算机硬件及人工智能芯片[39-40]的发展,单纯在软件层面引入机器学习所带来的性能提升,与同时在系统底层引入智能处理器、机器学习加速设备等新硬件带来的性能提升,必然有差别,但该方向上的研究也面临着更大挑战.

5) 特定应用领域的机器学习化数据库系统.目前的机器学习化数据库系统的研究工作主要集中在利用机器学习等方法对底层数据进行建模.设计通用型系统,需要处理不同的数据模式、数据类型、数据分布等通用目标,不具有灵活性.面向特定应用领域,利用特定应用场景和该应用领域所独有的数据特征、数据分布等优势,能够训练出更好的模型.因此设计面向特定应用领域的机器学习化数据库系统更具有发展前景.并且机器学习化数据库系统在只读型数据库系统中有较好的应用,面对更新频繁的数据库,底层数据变化、模型失效、更新模型的代价以及模型的维护都是有待考虑的问题.因此更新频繁的数据库应用领域的机器学习化研究面临着更大的挑战.

6) 云数据库系统的机器学习化研究.随着数据库系统更多地部署在云服务器中,工作负载的规模不断增大,工作负载的类型也不断丰富,异构性突出.工作负载快速变化,对数据库系统的性能提出更高的要求.工作负载的快速变化及其多样性,要求数据库系统针对工作负载的变化能够智能且动态地适配数据库系统的配置,从而保持数据库系统的最佳运行时状态.云环境下数据库系统的工作负载异构性更加突出[88].云环境下部署的数据库系统,带来了数据库系统研究工作中很多不需要考虑的执行环境,在云环境下计算和存储资源可以视为是“无限的”,理论上 DBMS 可以立即执行对资源的配置,进行灵活地缩减或扩张.为机器学习的应用提供了更广阔空间,推动云环境下数据库管理系统的动态配置,值得进一步研究.

7) 新兴数据库系统的机器学习化研究.大数据时代的发展,数据产生方式千变万化,数据之间存在错综复杂的关系[100].当前数据时代呈现大规模数据关联、交叉、融合的局面,数据的形态也产生巨大变化,因此数据的存储、管理、查询也随之而变,并超越传统数据库模式.例如云环境下的分布式数据库管理系统、非关系型数据库系统、XML 数据库系统、XML/RDBMS 混合的数据库系统、内存/闪存数据库系统等新兴的数据库管理系统.推动新兴数据库管理系统与机器学习技术的结合同样具有理论和实践的研究价值,该领域具备广阔的发展空间.

数据库领域经过几十年的研究,已经有了很多成熟的优化系统性能的技术,机器学习的发展也有目共睹.机器学习与数据库系统这 2 个领域,都致力于数据驱动的应用程序,有很多可以共享的通用技术.无论数据库系统化的机器学习研究,还是机器学习化的数据库系统研究,作为 2 个领域进一步结合

的研究方向,具有发展潜力.现在是软件 2.0(Software 2.0)的时代[101-102],即基于数据与新硬件的机器学习模型,将机器学习模型视为新一代软件系统:Software 2.0=Model+Data+Hardware.机器学习化的数据库系统迎合了 Software 2.0 时代的发展.

近来机器学习化数据库系统的提出和研究,有望突破数据库系统发展的瓶颈,未来值得我们进一步探讨在数据库系统中引入机器学习的相关技术.

参 考 文 献

[1] Ré C, Agrawal D, Balazinska M, et al. Machine learning and databases: The sound of things to come or a cacophony of hype? [C] //Proc of the 2015 ACM SIGMOD Int Conf on Management of Data. New York: ACM, 2015: 283-284

[2] Yang Chen, Du Zhihui, Meng Xiaofeng, et.al. A frequency scaling based performance indicator framework for big data systems [C] //Proc of the 24th Int Conf on DSFAA. Berlin: Springer, 2019: 19-35

[3] Kraska T, Beutel A, Chi E H, et al. The case for learned index structures [C] //Proc of the 2018 Int Conf on Management of Data. New York: ACM, 2018: 489-504

[4] Huang J. Moore law is dead but GPU will get 1000X faster by 2025 [OL]. [2018-10-15]. https://www.nextbigfuture.com/2017/06/moore-law-is-dead-but-gpu-will-get-1000x-faster-by-2025.html

[5] Nvidia. GEFORCE RTX 2080 Ti [OL]. [2019-02-20]. https://www.nvidia.cn /geforce/graphics-cards/rtx-2080-ti/

[6] Kipf A, Kipf T, Radke B, et al. Learned cardinalities: Estimating correlated joins with deep learning [C/OL] //Proc of the 9th Biennial Conf on Innovative Data Systems Research (CIDR). 2019 [2019-01-20]. http://cidrdb.org/

[7] Van Aken D, Pavlo A, Gordon G J, et al. Automatic database management system tuning through large-scale machine learning [C] //Proc of the 2017 ACM Int Conf on Management of Data. New York: ACM, 2017: 1009-1024

[8] Oracle Corporation and/or Its Affiliates. MySQL-InnoDB startup options and system variables [OL]. [2019-03-20]. http://dev. mysql. com/doc/refman /5. 6/en/innodb-parameters.html

[9] Zhou Zhihua. Machine Learning [M]. Beijing: Tsinghua University Press, 2016 (in chinese)
(周志华. 机器学习[M]. 北京: 清华大学出版社, 2016)

[10] Garcia-Molina H, Ullman J D, Widom J. Database System Implementation [M]. 2nd ed. Upper Saddle River, NJ: Prentice Hall, 2000

[11] Kumar A, Boehm M, Yang Jun. Data management in machine learning: Challenges, techniques, and systems [C] //Proc of the 2017 ACM Int Conf on Management of Data. New York: ACM, 2017: 1717-1722

[12] Marcus R, Papaemmanouil O. Deep reinforcement learning for join order enumeration [C] //Proc of the 1st Int Workshop on Exploiting Artificial Intelligence Techniques for Data Management co-located with SIGMOD. New York: ACM, 2018: 3:1-3:4

[13] Wang Wei, Zhang Meihui, Chen Gang, et al. Database meets deep learning: Challenges and opportunities [J]. ACM SIGMOD Record, 2016, 45(2): 17-22

[14] Silberschatz A, Korth H F, Sudarshan S. Database System Concepts [M]. 6th ed. New York: McGraw-Hill, 2010

[15] Wang Shan, Sa Shixuan. An Introduction to Database System [M]. 5th ed. Beijing: Advanced Education Press, 2000 (in Chinese)
(王珊, 萨师煊. 数据库系统概论[M]. 第 5 版. 北京: 高等教育出版社, 2000)

[16] Lykouris T, Vassilvitskii S. Competitive caching with machine learned advice [C] //Proc of the 35th Int Conf on Machine Learning. Stockholm, Sweden: PMLR 80, 2018: 3302-3311

[17] Margaritov A, Ustiugov D, Bugnion et al. Virtual address translation via learned page table indexes [C/OL] //Proc of Workshop on ML for Systems at NeurIPS co-located with 32nd Conf on NIPS. 2018 [2019-02-20]. http://www-use rs.cselabs.umn.edu/classes/Spring2019/csci8980/papers/virt_addr_trans.pdf

[18] Cao Wei. Survey on automatic physical database design [J]. Application Research of Computers, 2012, 29(5): 1606-1612 (in Chinese)
(曹巍. 数据库物理自调优研究技术综述[J]. 计算机应用研究, 2012, 29(5): 1606-1612)

[19] Shasha D, Bonnet P. Database Tuning: Principles, Experiments, and Troubleshooting Techniques [M]. Netherlands: Elsevier, 2002

[20] Pavlo A, Angulo G, Arulraj J, et al. Self-driving database management systems [C/OL] //Proc of the 2017 Biennial Conf on Innovative Data Systems Research (CIDR). 2018 [2018-12-15]. http://cidrdb.org/

[21] Carnegie Mellon University Database Group. Peloton database management system [OL]. [2018-11-20]. http://pelotondb.org

[22] Park Y, Tajik A S, Cafarella M, et al. Database learning: Toward a database that becomes smarter every time [C] //Proc of the 2017 ACM Int Conf on Management of Data. New York: ACM, 2017: 587-602

[23] Wang Zhe, Cashman D, Li Mingwei, et al. NNCubes: Learned structures for visual data exploration [J]. arXiv preprint arXiv:1808.08983, 2018

[24] Cumin J, Petit J M, Scuturici V M, et al. Data exploration with SQL using machine learning techniques [C/OL] //Proc of the Int Conf on EDBT 2017. [2018-11-15]. https://openproceedings.org/2017/conf/edbt/paper-155.pdf

[25] Guilly M L, Petit J M, Scuturici V M. SQL query completion for data exploration [J]. arXiv preprint, arXiv:1802.02872, 2018

[26] Zolaktaf Z. Facilitating user interaction with data [C/OL] // Proc of the 2017 VLDB PhD Workshop. [2018-12-23]. http://ceur-ws.org/Vol-1882/paper11.pdf

[27] Martins D M L, Vossen G, de Lima Neto F B. Learning database queries via intelligent semiotic machines [C] //Proc of 2017 IEEE Latin American Conf on Computational Intelligence (LA-CCI). Piscataway, NJ: IEEE, 2017: 1-6

[28] OmniSci. OmniSci core_open source analytical database for the GPU [OL]. [2019-03-10]. https://www.omnisci.com/platform/core/

[29] Kinetica D B. Kinetica_GPU-accelerated database [OL]. [2019-03-10]. https://www.kinetica.com/products/gpu-accelerated-database/

[30] BlazingDB. BlazingDB_Distributed GPU SQL Engine on RAPIDS [OL]. [2019-03-10]. http://blazingdb.com/#/

[31] Blazegraph Company. Blazegraph Whitepapers [OL]. [2019-03-10]. https://www.blazegraph.com/

[32] PostgreSQL. Pg-strom [OL]. [2019-03-10]. https://github.com/heterodb/pg-strom

[33] Owaida M, Sidler D, Kara K, et al. Centaur: A framework for hybrid CPU-FPGA databases [C] //Proc of 2017 IEEE 25th Annual Int Symp on Field-Programmable Custom Computing Machines (FCCM). Piscataway, NJ: IEEE, 2017: 211-218

[34] Chung E S, Davis J D, Lee J. LINQits: Big data on little clients [C] //Proc of ACM SIGARCH Computer Architecture News. New York: ACM, 2013: 261-272

[35] Mahajan D, Kim J K, Sacks J, et al. In-RDBMS hardware acceleration of advanced analytics [J]. VLDB Endowment, 2018, 11(11): 1317-1331

[36] Sidler D, István Z, Owaida M, et al. doppioDB: A hardware accelerated database [C] //Proc of the 2017 ACM Int Conf on Management of Data. New York: ACM, 2017: 1659-1662

[37] Mueller R, Teubner J, Alonso G. Data processing on FPGAs [J]. VLDB Endowment, 2009, 2(1): 910-921

[38] Du Zidong, Rubin D D B D, Chen Yunji, et al. Neuromorphic accelerators: A comparison between neuroscience and machine-learning approaches [C] //Proc of 2015 48th Annual IEEE Int Symp on Microarchitecture. Piscataway, NJ: IEEE, 2015: 494-507

[39] Liu Shaoli, Du Zidong, Tao Jinhua, et al. Cambricon: An instruction set architecture for neural networks [C] //Proc of ACM SIGARCH Computer Architecture News. Piscataway, NJ: IEEE, 2016: 393-405

[40] Ullman J D, Widom J. A First Course in Database Systems [M]. Englewood Cliffs, NJ: Prentice Hall, 1997

[41] Reichenbach L. Learned index structures: An evalution of their performance in key-value storage solutions [D]. Hamburg, DE: University of Hamburg, 2018

[42] Mitzenmacher M. A model for learned Bloom filters and related structures [J]. arXiv preprint, arXiv: 1802.00884, 2018

[43] Mitzenmacher M. A model for learned Bloom filters and optimizing by sandwiching [C/OL] //Proc of NeurIPS 2018. [2019-02-20]. http://papers.nips.cc/book/advances-in-neural-information-processing-systems-31-2018

[44] Darshana B, Lukasz Z, Oliver K. Self-adapting index compilation [OL]. [2018-11-01]. https://cse.buffalo.edu/tech-reports/2018-03.pdf

[45] Oosterhuis H, Culpepper J S, de Rijke M. The potential of learned index structures for index compression [C] //Proc of the 23rd Australasian Document Computing Symp. New York: ACM, 2018: 7:1-7:4

[46] Xiang Wenkun, Zhang Hao, Cui Rui, et al. Pavo: A RNN-based learned inverted index, supervised or unsupervised? [J]. IEEE Access, 2019 (7): 293-303

[47] Babcock B, Chaudhuri S. Towards a robust query optimizer: A principled and practical approach [C] //Proc of the 2005 ACM SIGMOD Int Conf on Management of Data. New York: ACM, 2005: 119-130

[48] Leis V, Gubichev A, Mirchev A, et al. How good are query optimizers, really? [J]. VLDB Endowment, 2015, 9(3): 204-215

[49] Moerkotte G, Neumann T, Steidl G. Preventing bad plans by bounding the impact of cardinality estimation errors [J]. VLDB Endowment, 2009, 2(1): 982-993

[50] Arulkumaran K, Deisenroth M P, Brundage M, et al. A brief survey of deep reinforcement learning [J]. arXiv preprint, arXiv:1708.05866, 2017

[51] Ortiz J, Balazinska M, Gehrke J, et al. Learning state representations for query optimization with deep reinforcement learning [C] //Proc of the 2nd Workshop on Data Management for End-To-End Machine Learning co-located with SIGMOD. New York: ACM, 2018: 4:1-4:4

[52] Marcus R, Papaemmanouil O. Towards a hands-free query optimizer through deep learning [C/OL] //Proc of the 9th Biennial Conf on Innovative Data Systems Research (CIDR 2019). [2019-01-20]. http://cidrdb.org/

[53] Hester T, Vecerik M, Pietquin O, et al. Deep Q-learning from demonstrations [J]. arXiv preprint, arXiv:1704.03732, 2017

[54] Erickson N, Zhao Qi. Dex: Incremental learning for complex environments in deep reinforcement learning [J]. arXiv preprint, arXiv:1706.05749, 2017

[55] Buffet O, Dutech A, Charpillet F. Incremental reinforcement learning for designing multi-agent systems [C] //Proc of the 5th Int Conf on Autonomous Agents. New York: ACM, 2001: 31-32

[56] Krishnan S, Yang Zongheng, Goldberg K, et al. Learning to optimize join queries with deep reinforcement learning [J]. arXiv preprint, arXiv:1808.03196, 2018

[57] Wu Chenggang, Jindal A, Amizadeh S, et al. Towards a learning optimizer for shared clouds [J]. VLDB Endowment, 2018, 12(3): 210-222

[58] Heimel M, Kiefer M, Markl V. Self-tuning, GPU-accelerated kernel density models for multidimensional selectivity estimation [C] //Proc of the 2015 ACM SIGMOD Int Conf on Management of Data. New York: ACM, 2015: 1477-1492

[59] Kiefer M, Heimel M, Breß S, et al. Estimating join selectivities using bandwidth-optimized kernel density models [J]. VLDB Endowment, 2017, 10(13): 2085-2096

[60] Park Y, Zhong S, Mozafari B. QuickSel: Quick selectivity learning with mixture models [J]. arXiv preprint, arXiv: 1812.10568, 2018

[61] Ganapathi A, Kuno H, Dayal U, et al. Predicting multiple metrics for queries: Better decisions enabled by machine learning [C] //Proc of IEEE 25th Int Conf on Data Engineering. Piscataway, NJ: IEEE, 2009: 592-603

[62] Bi Liyuan, Wu Sai, Chen Gang, et al. Database query cost prediction using recurrent neural network [J]. Journal of Software, 2018, 29(3): 799-810 (in Chinese)
(毕里缘, 伍赛, 陈刚, 等. 基于循环神经网络的数据库查询开销预测[J]. 软件学报, 2018, 29(3): 799-810)

[63] Dias K, Ramacher M, Shaft U, et al. Automatic performance diagnosis and tuning in Oracle [C/OL] //Proc of the 2005 Biennial Conf on Innovative Data Systems Research (CIDR 2005). [2018-10-22]. http://cidrdb.org/

[64] Storm A J, Garcia-Arellano C, Lightstone S S, et al. Adaptive self-tuning memory in DB2 [C] //Proc of the 32nd Int Conf on Very Large Data Bases. New York: ACM, 2006: 1081-1092

[65] Linster M. Best practices for becoming an exceptional Postgres DBA [OL]. 2014 [2018-10-20]. http://www.enterprisedb.com/best-practices-becoming-exceptional-postgres-dba

[66] Pavlo A. What is a self-driving database management system? Carnegie Mellon University [OL]. [2018-10-25]. http://www.cs.cmu.edu/~pavlo/blog/2018/04/what-is-a-self-driving-database-management-system.html#footnote-shareddisk

[67] Hammer M, Chan A. Index selection in a self-adaptive data base management system [C] //Proc of the 1976 ACM SIGMOD Int Conf on Management of Data. New York: ACM, 1976: 1-8

[68] Finkelstein S, Schkolnick M, Tiberio P. Physical database design for relational databases [J]. ACM Transactions on Database Systems, 1988, 13(1): 91-128

[69] Hammer M, Niamir B. A heuristic approach to attribute partitioning [C] //Proc of the 1979 ACM SIGMOD Int Conf on Management of Data. New York: ACM, 1979: 93-101

[70] DeWitt D J, Ghandeharizadeh S. Hybrid-range partitioning strategy: A new declustering strategy for multiprocessor database machine [C] //Proc of the 16th Int Conf on VLDB. New York: ACM, 1990: 481-492

[71] Zilio D C, Jhingran A, Padmanabhan S. Partitioning key selection for a shared-nothing parallel database system [OL]. 1994 [2018-12-15]. https://www.researchgate.net/publication/2623565_Partitioning_Key_Selection_for_a_Shared-Nothing_Parallel_Database_System

[72] Agarawal S, Chaudhuri S, Narasayya V. Automated selection of materialized views and indexes for SQL databses [C] //Proc of the 26th Int Conf on Very Large Databases. New York: ACM, 2000: 191-207

[73] Agrawal S, Chaudhuri S, Kollar L, et al. Database tuning advisor for microsoft SQL server 2005 [C] //Proc of the 2005 ACM SIGMOD Int Conf on Management of Data. New York: ACM, 2005: 930-932

[74] Zilio D C, Rao J, Lightstone S, et al. DB2 design advisor: Integrated automatic physical database design [C] //Proc of the 30th Int Conf on Very Large Data Bases-Volume 30. New York: ACM, 2004: 1087-1097

[75] Yagoub K, Belknap P, Dageville B, et al. Oracle's SQL performance analyzer [J]. IEEE Data Engineering Bulletin, 2008, 31(1): 51-58

[76] Markl V, Lohman G M, Raman V. LEO: An autonomic query optimizer for DB2 [J]. IBM Systems Journal, 2003, 42(1): 98-106

[77] Lim L, Wang Min, Vitter J S. SASH: A self-adaptive histogram set for dynamically changing workloads [C] //Proc of the 29th Int Conf on Very Large Data Bases-Volume 29. New York: ACM, 2003: 369-380

[78] Narayanan D, Thereska E, Ailamaki A. Continuous resource monitoring for self-predicting DBMS [C] //Proc of the 13th Int Symp on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems. Piscataway, NJ: IEEE, 2005: 239-248

[79] Wiseth K. Oracle database 10g: The world's first self managing, grid-ready database arrives [J]. Oracle Magazine, 2003, 17(5): 28-37

[80] Sullivan D G, Seltzer M I, Pfeffer A. Using probabilistic reasoning to automate software tuning [C] //Proc of the Int Conf on SIGMETRICS 2004. New York: ACM, 2004: 404-405

[81] Tran D N, Huynh P C, Tay Y C, et al. A new approach to dynamic self-tuning of database buffers [J]. ACM Transactions on Storage, 2008, 4(1): 3:1-3:25

[82] Duan S, Thummala V, Babu S. Tuning database configuration parameters with iTuned [J]. VLDB Endowment, 2009, 2(1): 1246-1257

[83] Meng Xiaofeng, Ci Xiang. Big data management: Concepts, techniques and challenges [J]. Journal of Computer Research and Development, 2013, 50(1): 146-169 (in Chinese)
(孟小峰, 慈祥. 大数据管理: 概念, 技术与挑战[J]. 计算机研究与发展, 2013, 50(1): 146-169)

[84] Das S, Li Feng, Narasayya V R, et al. Automated demand-driven resource scaling in relational database-as-a-service [C] //Proc of the 2016 Int Conf on Management of Data. New York: ACM, 2016: 1923-1934

[85] Ma Lin, Van Aken D, Hefny A, et al. Query-based workload forecasting for self-driving database management systems [C] //Proc of the 2018 Int Conf on Management of Data. New York: ACM, 2018: 631-645

[86] Jain S, Yan Jiaqi, Cruanes T, et al. Database-agnostic workload management [C/OL] //Proc of the 9th Biennial Conf on Innovative Data Systems Research (CIDR). 2019 [2019-01-20]. http://cidrdb.org/

[87] Kraska T, Alizadeh M, Beutel A, et al. SageDB: A learned database system [C/OL] //Proc of the 9th Biennial Conf on Innovative Data Systems Research (CIDR). 2019 [2019-01-20]. http://cidrdb.org/

[88] Sharma A, Schuhknecht F M, Dittrich J. The case for automatic database administration using deep reinforcement learning [J]. arXiv preprint, arXiv:1801.05643, 2018

[89] Kester M S, Athanassoulis M, Idreos S. Access path selection in main-memory optimized data systems: Should I scan or should I probe? [C] //Proc of the 2017 Int Conf on Management of Data. New York: ACM, 2017: 715-730

[90] Stratos I, Niv D, Wilson Q, et al. Design continuums and the path toward self-designing key-value stores that know and learn [C/OL] //Proc of the 9th Biennial Conf on Innovative Data Systems Research (CIDR). 2019 [2019-01-20]. http://cidrdb.org/

[91] Harvard University-Stratos Idreos. Self-designing data systems [OL]. [2019-01-14]. https://stratos.seas.harvard.edu/

[92] Stratos I, Kostas Z, Manos A, et al. The periodic table of data structures [J]. IEEE Data Engineering Bulletin, 2018, 41(3): 64-75

[93] Tian Boyu, Huang Jiamin, Mozafari B, et al. Contention-aware lock scheduling for transactional databases [J]. VLDB Endowment, 2018, 11(5): 648-662

[94] Kossmann J. Self-driving: From general purpose to specialized DBMSs [C/OL] //Proc of the PhD Workshop co-located with 44th Int Conf on Very Large Database. [2019-02-15]. http://ceur-ws.org/Vol-2175/paper15.pdf

[95] Oracle. Oracle's autonomous database: AI-based automation for database management and operations [OL]. [2018-10-15]. http://www.oracle.com/us/products/database/idc-oracles-autonomous-database-4497146.pdf, 2018.

[96] Oracle. Oracle autonomous database: The industry's first self-driving database [OL]. [2018-09-18]. https://www.oracle.com/cn/database/autonomous-database.html

[97] Oracle integrated cloud. Oracle autonomous database. [2018-10-20]. https://www.oracle.com/cn/database/autonomous-database.html

[98] Huawei. Huawei database & storage product launch [OL]. [2019-05-16]. http://e.huawei.com/topic/database-storage-launch2019/cn/

[99] Chen Jianjun, Chen Yu, Chen Zhibiao, et al. Data management at huawei: Recent accomplishments and future challenges [C] //Proc of the 35th IEEE Int Conf on Data Engineering (ICDE). Piscataway, NJ: IEEE, 2019: 13-24

[100] Meng Xiaofeng, Du Zhijuan. Research on the big data fusion: Issues and challenges [J]. Journal of Computer Research and Development, 2016, 53(2): 231-246 (in Chinese)
(孟小峰, 杜治娟. 大数据融合研究:问题与挑战[J]. 计算机研究与发展, 2016, 53(2):231-246)

[101] Andrej K. Software 2.0 [OL]. [2017-11-12]. https://medium.com/@karpathy/software-2-0-a64152b37c35

[102] Cedric R, Bojan K, Bolin D, et al. Continuous integration of machine learning models with ease.ml/ci: towards a rigorous yet practical treatment [C/OL] //Proc of the 2nd SysML Conf. 2019 [2019-04-10]. http://www.sysml.cc/

Meng Xiaofeng, born in 1964. Professor, PhD supervisor at Renmin University of China. Fellow of CCF. His main research interests include cloud data management, Web data management, flash-based databases and privacy protection.

Ma Chaohong, born in 1994. PhD candidate at Renmin University of China. Student member of CCF. Her main research interests include learned database systems, autonomous database systems and science data management. (chaohma@ruc.edu.cn)

Yang Chen, born in 1987. PhD candidate at Renmin University of China. His main research interests include science data management and performance bottleneck quantification. (yang_chen@ruc.edu.cn)