软件学报 ISSN 1000-9825, CODEN RUXUEW
[doi: 10.13328/j.cnki.jos.007240] [CSTR: 32375.14.jos.007240]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

# 云多租数据库资源规划调度技术综述*

刘海龙[1,2], 王 硕[1,2], 侯舒峰[1,2], 徐海洋[1,2], 李战怀[1,2]

[1](西北工业大学 计算机学院, 陕西 西安 710072)

[2](大数据存储与管理工业和信息化部重点实验室 (西北工业大学), 陕西 西安 710072)

通信作者: 刘海龙, E-mail: liuhailong@nwpu.edu.cn

**摘 要**: 云多租数据库具有按需付费、按需扩展、免部署、高可用、自带运维能力、资源共享等诸多优势, 可以大大降低用户使用数据库服务的成本. 现在越来越多的企业和个人开始在数据库即服务 (DaaS) 平台托管他们的数据库服务. DaaS 平台需要按照用户服务水平协议 (SLA) 为诸多租户提供服务, 同时也需要保障平台收益. 但是, 由于租户及其负载具有动态性、异构性和竞争性等特点, 如何在遵循 SLA 的同时根据负载自适应规划调度资源同时兼顾平台收益对于 DaaS 平台来说是一件极具挑战性的工作. 针对云多租数据库中比较常见的类型, 如关系型数据库, 详细分析了当前云多租数据库资源规划调度技术所面临的挑战, 提炼了关键科学问题, 给出了技术框架, 然后从资源规划调度技术、资源预估技术、资源弹性伸缩技术以及数据库资源规划调度工具等 4 个方面对现有研究工作进行了总结和分析, 并且展望了未来的研究方向.

**关键词**: 多租户; 云数据库; 资源规划调度; 资源预估; 资源弹性伸缩; 关系数据库; 云原生; 多模数据库

**中图法分类号**: TP311

中文引用格式: 刘海龙, 王硕, 侯舒峰, 徐海洋, 李战怀. 云多租数据库资源规划调度技术综述. 软件学报. http://www.jos.org.cn/1000-9825/7240.htm

英文引用格式: Liu H, Wang S, Hou SF, Xu HY, Li ZH. Survey on Resource Planning and Scheduling Technologies for Multi-tenant Cloud Databases. Ruan Jian Xue Bao/Journal of Software (in Chinese). http://www.jos.org.cn/1000-9825/7240.htm

## Survey on Resource Planning and Scheduling Technologies for Multi-tenant Cloud Databases

LIU Hai-Long[1,2], WANG Shuo[1,2], HOU Shu-Feng[1,2], XU Hai-Yang[1,2], LI Zhan-Huai[1,2]

[1](School of Computer Science, Northwestern Polytechnical University, Xi'an 710072, China)

[2](Key Laboratory of Big Data Storage and Management (Northwestern Polytechnical University), Ministry of Industry and Information Technology, Xi'an 710072, China)

**Abstract**: Multi-tenant cloud databases offer services more cheaply and conveniently, with advantages like paying on demand, scaling on demand, automatic deployment, high availability, self-maintenance, and shared resources. Now more and more enterprises and individuals begin to host their database services on database as a service (DaaS) platforms. These DaaS platforms provide services to multiple tenants in accordance with their service-level agreements (SLAs), while improving revenue for themselves. However, due to the dynamic, heterogeneous, and competitive characteristics of multiple tenants and their loads, it is a very challenging task for DaaS platform providers to adaptively plan and schedule resources according to dynamic loads while complying with multi-tenants' SLAs. For common types of multi-tenant cloud databases, such as relational databases, this survey firstly analyzes the challenges faced by resource planning and scheduling of multi- tenant cloud databases in detail and then outlines related key scientific issues. Then, it provides a framework of related techniques and a summary of existing research in four areas: resource planning and scheduling technologies, resource forecasting technologies, resource elastic scaling technologies, and resource planning and scheduling tools for existing databases. Lastly, this survey provides suggestions for future research directions on resource planning and scheduling technologies for multi-tenant cloud databases.

**Key words**: multi-tenant; cloud database; resource planning and scheduling; resource estimation; resource elastic scaling; relational database;

---

* 基金项目: 国家重点研发计划 (2023YFB4503600); 国家自然科学基金 (62172335); CCF-华为胡杨林基金 (CCF-HuaweiDBIR0004B)
收稿时间: 2023-10-27; 修改时间: 2024-05-06; 采用时间: 2024-06-17; jos 在线出版时间: 2024-11-06

cloud-native; multi-model database

随着云计算与大数据技术的发展, 越来越多的企业和个人选择租用云服务提供商的平台来存放和处理用户或产品数据. 因为云数据库具有按需付费、按需扩展、免部署、高可用、自带运维能力等诸多优势, 数据库即服务 (database as service, DaaS) 也成为企业和学术界关注的热点.

为了提高资源利用率, 增加平台收益, 云服务提供商通常会考虑将一套硬件资源、软件资源分租给多个用户. 多个租户共享集群中的 CPU、内存、网络带宽等资源, 就不可避免地存在资源竞争关系. 不合理的资源共享会导致集群性能损耗, 真实生产环境中集群性能最坏可能会降低到原来的十分之一[1]. 为应对上述问题, 分布式集群都提供资源的规划调度机制, 但是它们资源管理的粒度基本都是实体机、虚拟机或者 Docker, 并不关注某一类特定软件系统的特性. 云多租数据库作为当下最重要的软件系统之一, 一般用于处理来自不同租户间的在线处理任务[2], 通常提供 3 种多租户模式: “不共享”, 一个租户独享一个数据库实例; “部分共享”, 多个租户共享一个数据库实例但是数据空间独立; “全共享”, 多个租户共享数据库实例及数据空间. 在“不共享”多租户模式下, 多个租户数据库实例会竞争集群资源, 其资源竞争、共享的模式和其他多租户系统类似. 而在“部分共享”和“全共享”多租户模式下, 多个租户、多种任务势必会竞争数据库实例内部的资源, 其资源竞争、共享的模式与数据库系统本身的实现机制密切相关. 云多租数据库管理者必须根据这些特性提供一套灵活的资源规划调度机制来保障在共享资源的同时提升云多租数据库的性能.

但是, 鉴于数据库系统的复杂性, 如何在满足用户性能需求, 即遵守用户服务水平协议 (SLA) 的同时根据负载自适应规划调度资源对于云多租数据库来说是一件极具挑战性的工作. 文献 [3] 关注了云环境下的资源规划调度技术, 梳理了单个机器内部和集群级别上的资源调度技术, 并关注了深度学习在资源规划调度上的应用. 文献 [4] 关注了集群资源的规划调度技术, 其中提及的方法重点解决 Hadoop、Spark 任务间粗粒度的资源共享问题. 文献 [5] 中认为云时代数据库需要具备利用不同类型云基础服务独立伸缩的特点优化弹性调度能力、提高资源利用效率的能力, 但是该文专注于云原生 OLTP、OLAP 核心技术. 文献 [6] 综述了机器学习在数据库调参、基数估计、查询计划选择、索引推荐、物化视图选择等方面的应用. 这些工作都没有涉及多租户数据库 3 种共享模式下的资源规划调度技术. 为此, 本文对云多租数据库的资源规划调度技术所面临的挑战、研究进展及未来研究趋势进行了系统地综述.

论文的核心贡献如下:

(1) 剖析了云多租数据库资源规划调度所面临的异构性、动态性及竞争性 3 个方面的挑战, 提炼了应对这些挑战需解决的关键科学问题, 并给出了对应的技术框架. 框架中资源预估技术、资源弹性伸缩技术及资源规划调度技术共同协作为多租户多模融合负载的数据库提供低成本、自适应、精细的资源规划调度服务.

(2) 根据本文提出的技术框架, 系统地梳理了资源规划调度技术、资源预估技术、资源弹性伸缩技术以及数据库资源规划调度工具等 4 个方面的研究工作进展.

(3) 分析了云原生、多模态云多租数据库由于资源池化、引擎异构在资源规划调度方面面临的新挑战, 并且展望了未来的研究方向.

本文第 1 节分析云多租数据库资源规划调度技术所面临的核心挑战、提炼拟解决的关键科学问题. 第 2 节介绍资源规划调度技术的研究框架. 第 3 节系统地综述云多租数据库资源规划调度技术的研究进展. 第 4 节展望未来的研究趋势. 第 5 节对全文工作进行总结.

## 1 核心挑战和亟需解决的关键科学问题

为便于理解, 首先对如下术语进行说明:

(1) 云多租数据库: 指云服务提供商提供的支持多租户模式的关系型数据库、NoSQL 数据库等. 数据库的类型对资源规划调度并没有本质影响, 故本文将以关系型多租户数据库为主要研究对象, 其方法也可以应用于其他类型的多租户数据库.

(2) 资源: 在本文中指代每个租户在云服务提供商所获取到的计算资源 (CPU 核数)、存储资源 (内存大小、

I/O 带宽)、网络资源 (网络带宽) 等.

(3) 作业: 租户提交的插入、查询、更新等任务, 是资源消耗的实体.

(4) 资源规划调度: 根据不同场景的实际需求分配、调整各个租户和整个系统所能使用的资源数量.

云多租数据库资源规划调度主要面临如下 3 个方面的挑战.

(1) 异构性. 现代云多租数据库系统中租户的复杂性需要更复杂的资源分配策略进行数据处理. 负载资源消费模式更加复杂多样, 而且呈明显的分布特性. 不同任务之间还可能存在一定的依赖或约束关系. 不同计算节点的硬软件资源配置 (CPU、内存、存储设备等)、数据分布也各异. 不同的租户或者同一租户不同时间也会对数据库有不同的定制化需求, 导致用户、负载性能要求、资源约束都不相同. 多种引擎、多类负载、多种资源、多种约束等因素的混杂大大增加了数据库资源规划调度问题的复杂度, 使其成为一个 NP 难组合优化问题[7].

(2) 动态性. 现代云多租数据库中负载通常是动态变化的, 负载变化导致不同租户或同一租户的资源需求也在动态变化. 另外, 系统资源和数据分布格局也都在动态变化. 在这种动态的环境下, 传统的静态资源规划机制很难发挥效用. 如何自适应动态规划调整资源以满足不同租户的性能要求、保障服务质量并同时提高资源利用率就成为一个难题.

(3) 竞争性. 现代云多租数据库为了保障不同租户的融合访问效率, 任务间通常会细粒度的共享缓存、CPU 等资源, 竞争几率大. 另外, 为了保障平台收益, 多个租户也会最大限度地共享系统资源, 资源超售[8]的情景会时常发生. 这导致不同任务之间资源冲突的几率大大增加. 而冲突又会导致系统性能下降并面临因违反 SLA 而引起的收益减少、用户满意度下降等风险. 如何保障租户最大限度地共享系统资源的同时并保障系统效益的最大化是云多租数据库资源管理面临的一个难题.

若要很好地应对上述挑战, 首先需要解决两个关键问题: (1) 多租数据库的动态变化的、多模态的负载是如何消费资源的? (2) 如何控制资源规划调度的综合收益? 即如何获知租户的资源需求与引擎、负载、SLA 之间的关联关系? 如何控制成本提升资源规划调度的收益? 鉴于此, 本文提炼出如下关键科学问题.

(1) 租户的资源需求与引擎、负载、租户 SLA 之间的关联关系机理

解决云多租数据库资源规划调度面临的异构性、动态性及竞争性挑战的关键在于能否准确地预知租户负载以及租户的资源需求, 进行精准的资源画像, 其核心是不同多租户模式下资源需求与引擎、负载、租户 SLA 之间的关联关系机理. 该机理是 AI 赋能的资源规划调度机制的核心, 相关模型的准确度是影响资源分配效率和资源伸缩效率的关键因素. 云多租数据库中负载资源消费模式较之普通本地部署数据库的负载更加复杂, 数据库在线事务处理的业务特性也决定其不一定能够提供足够的样本用于模型的训练, 所以求解资源需求与引擎、负载、租户 SLA 等参数之间关联关系是一个难题. 它是云多租数据库资源规划调度必须要解决的一个关键问题.

(2) 资源自适应调整的成本控制机制

资源的自适应调整是应对动态性挑战的核心手段. 云多租数据库资源池化的特性导致其资源在物理位置上是分散的, 使得资源自适应调整难度更大、影响更广, 成本控制更难. 首先, 负载的执行路径可能会跨越多个物理节点, 在全路径上预测资源的调整时机、调整数量难度大. 某一处不恰当的调整都会导致系统处于“震荡”性反复调整状态从而严重降低整个系统的性能. 另外, 资源需要按照一定的策略在不同租户的负载间轮转, 存在一定的调度延迟. 在资源总量有限的情况下, 调整也势必会影响系统的资源配置格局, 部分租户负载的性能可能会受到影响, 从而触发因违反 SLA 协议而引起的惩罚. 另外, 若只专注于收益, 会导致部分租户的负载永远得不到响应而影响公平性. 所以自适应调整成本控制机制是衡量资源规划调度效果的关键, 也是关系型云多租数据库资源规划调度必须要解决的一个关键问题.

## 2 研究框架

云多租数据库系统资源规划调度技术存在异构性、动态性和竞争性等方面的挑战, 所以一个正确的资源规划调度框架需要是精细的、动态的和支持租户隔离的. 鉴于此, 本文提出的关系型云多租数据库资源规划调度技术框架如图 1 所示. 研究挑战、关键科学问题及研究框架各部分内容之间的关系如图 2 所示.

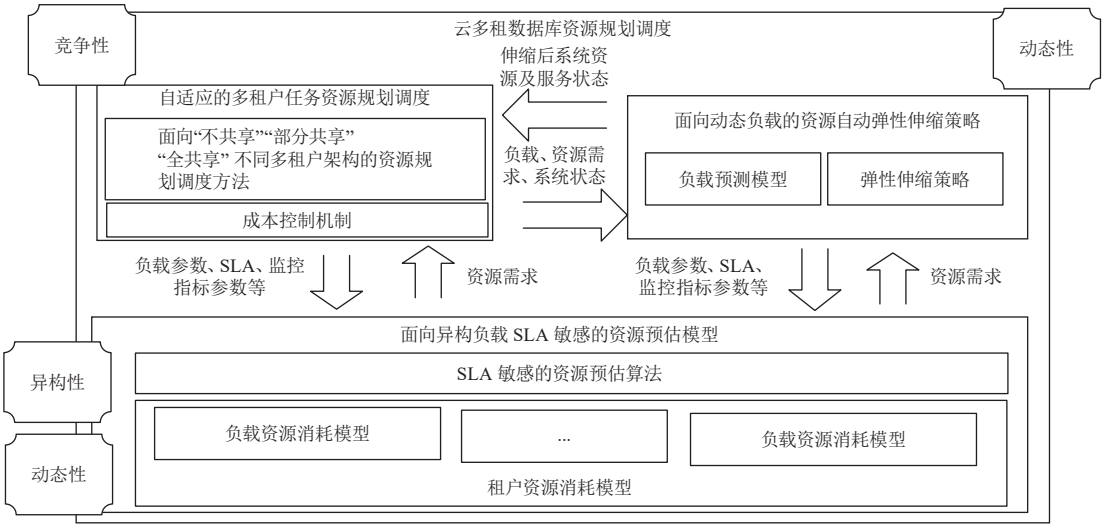


图 1　云多租数据库资源规划调度技术框架图

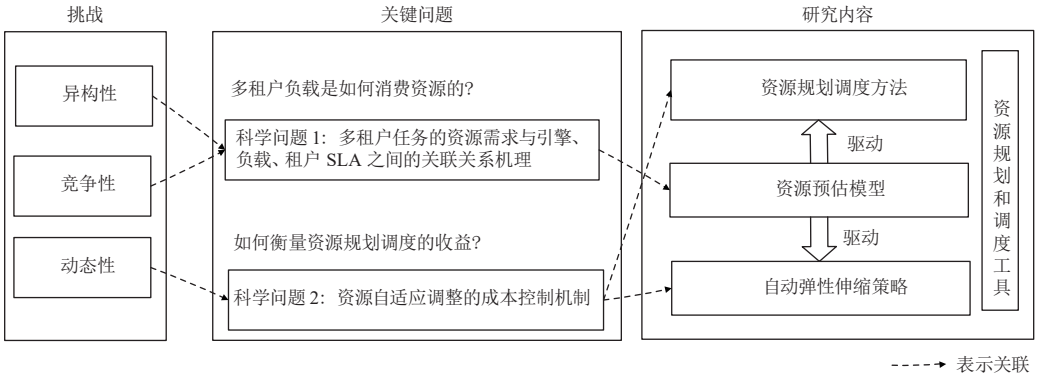


图 2　挑战、关键问题及研究内容之间的关系

云多租数据库资源规划调度技术框架包含如下 3 个部分.

(1) 资源规划调度. 为解决竞争性带来的问题, 需要对租户的资源进行精细地管控 (隔离、约束、共享). 资源规划调度技术根据负载特征、租户 SLA 等要求, 通过自适应精细调配各租户 CPU、内存、IO 等资源, 以实现提升系统性能及平台收益的目标.

(2) 资源预估. 为应对异构性、动态性方面的挑战, 需要提前预知租户未来的资源使用状况. 资源预估技术通过收集整个系统的特征化指标数据 (负载特征、租户 SLA、系统指标等) 来预测未来某一时间间隔内系统或租户资源需求. 它负责为资源弹性伸缩和资源规划调度提供精细的资源需求计算模型.

(3) 资源弹性伸缩. 为应对动态性方面挑战, 需要对系统的资源供给进行适时调整. 资源弹性伸缩技术利用资源预估技术预测系统资源消耗情况, 据此动态改变系统的资源供给 (例如: 调整资源池的大小). 它通过探究数据库系统负载特征、性能指标与资源调整的范围边界、资源调整的时机和资源量之间的数学关系, 基于异构计算资源的变更和指标 (负载、功能、性能、容错) 参数之间的关联关系自动求解系统资源动态伸缩策略.

三者的关系具体为: 资源预估主要探究资源需求与引擎、负载、租户 SLA 之间的关联关系机理, 为资源规划调度和资源弹性伸缩提供精细的资源需求计算模型; 面向动态负载的资源弹性伸缩则根据负载特征和系统资源消耗情况动态改变系统资源的供给格局; 自适应的资源规划调度则负责根据负载特征、租户 SLA 等进行租户间自

适应的资源配比. 三者协作在负载、租户 SLA、数据库系统状态等驱动下为多租户多模融合负载的数据库提供低成本、自适应、精细的资源规划调度服务.

## 3 研究现状分析

国内外在数据库和集群资源规划调度技术方面已开展了相关的研究, 针对不同类型的多租数据库在资源规划调度技术、资源预估技术、资源弹性伸缩技术和资源规划调度工具等方面取得了一系列研究成果.

### 3.1 资源规划调度

#### 3.1.1 “不共享”多租户模式下资源规划调度

资源规划调度是指通过一系列的策略使得在满足租户 SLA 的同时最大化平台收益. 在“不共享”多租户模式下, 资源规划调度更多依赖于云平台的资源规划调度能力. 这种模式下, 若把整个数据库实例看成一个大的“作业”, 资源规划调度的核心问题是为“作业”寻找合适的资源配置并根据负载的变化进行动态调整. 这方面已经有诸多可借鉴的研究成果.

Yahoo 和 Facebook 提出了 Capacity Scheduler[9]和 Fair Scheduler[10]. Capacity Scheduler 的核心是以队列为单位的资源管控机制. 每一个队列都会分配到一定额度的资源, 所有提交到队列中的作业共享该队列中的资源. 这样就保证了交互式短作业也会及时获得资源. Fair Scheduler 与 Capacity Scheduler 类似, 按资源池来组织作业. Fair Scheduler 采用 Max-Min[11]策略, 把资源公平地分配给每一个作业. 但是, 当网络带宽成为瓶颈的时候, 本地化就成为影响大数据集群性能的一个重要因素[12], 因此 Fair Scheduler 采用延迟调度策略[13]来调高 Hadoop 作业的本地化执行的概率, 其核心思想为: 在公平调度时, 若当前资源不是本地化的资源, 则跳过该作业而去考虑其他作业. Quincy[14]是一个为 Dryad[15]开发的基于流的调度器. 它把调度问题映射成一个图模型, 其中边的容量和权值分别表示公平性和本地性, 然后把公平调度方案选择问题转化成一个最小代价流问题. 相比较基于队列的公平调度方法, Quincy 网络传输的数据量减少了约 80%、系统的吞吐量提高了约 40%. 该方法的前提是任务调度前数据已经放置好, 数据量和位置都是明确的, 所以任务调度策略的数据传输代价的计算可以精确到字节. 而且该方法中假设每一个 Task 都是独立的, 杀死一个 Task 不会影响其他 Task 的运行. 事实上, 很多情况下 Task 之间有交互, 此时可能会要求某些作业调度到同一个框架上, Quincy 模型无法描述此类场景. 同时, Quincy 中资源需求的描述是单维的, 而现实中需求通常是多维的, 例如 CPU、内存和 IO 等资源的混合调整. 另外, 目前 Quincy 模型假设一台计算机只运行一个 task, 并不符合多核环境的应用场景. 此外, 很多作业的放置是有一些硬性约束的, 例如需要 GPU、需要一个公共的 IP 地址等, 不满足这些硬性约束会导致计算任务即使获取到资源仍无法运行, Max-Min 算法[11]也没有考虑到这些约束. 针对该问题, 文献 [16] 提出了一种带硬性约束的最大最小公平调度算法 CMMF. 但是 CMMF 只关注了单一约束, 并没有关注多个条件的组合约束. 文献 [16] 认为涉及到多维资源的硬性约束问题将是一个非常有挑战性的问题. 后续提出的主导资源公平算法 (dominant resource fairness, DRF)[17]将公平性的定义拓展到多维资源的组合, 针对多维资源将最大最小公平资源算法进行泛化, 在每个用户的主导资源 (所占总资源比例最多的资源) 上利用 Max-Min 算法[11]进行资源分配. DRF 算法在为每一个计算引擎分配资源时, 按照各引擎主导份额值从小到大排序, 每次给主导份额值最小的引擎分配资源, 然后重新计算该引擎的主导份额, 并重新对所有计算引擎的主导份额进行排序, 重复上述过程直到本次资源分配完毕. DRF 算法正是通过反复对拥有最小主导份额的引擎的利益最大化使得各个部分在它们主导资源类型上公平地分配到了相近的份额. Medea[18]专为放置长时间和短时间作业运行的容器而设计. 它遵循一种新颖的双调度器设计: (1) 对于长期运行的容器, 它采用基于优化的方法来考虑约束和全局目标; (2) 对于短期运行的容器, 它使用传统的基于任务的调度程序来降低放置延迟. Medea 的不足在于: 只考虑了节点的选择, 没有考虑在节点上分配资源的大小. 有一些工作在深度学习方面做出相应尝试. Optimus[19]是一种用于深度学习集群的定制作业调度程序, 使用在线拟合来预测训练期间模型的收敛性, 并建立性能模型来准确地估计训练速度. 但该资源调度模型主要针对深度学习这一类特定负载, 对数据库并不一定适用. 文献 [20] 关注了负载动态变化对于集群资源调度的影响, 提出了一种基于深度强化学习的模型来探索时

变负载在不同的时间间隔内的资源使用状态, 并且设法为负载划分等价类以提升模型的适用性. 针对不同资源需求的资源分配策略应该具备刺激共享、防欺诈、不嫉妒和帕累托最优的特征[21]. 在上述研究工作当中, 因为 DRF 算法同时具备上述 4 个特征而被 Mesos、Yarn 等系统广泛采用. 但是 DRF 只关注资源的数量而没有关注资源的质量, 会有失公平.

3.1.2 “部分共享”和“全共享”多租户模式下的资源规划调度

在“部分共享”和“全共享”多租户模式下, 资源规划调度则依赖于数据库内核本身的资源规划调度能力, 比较有代表性的工作分析如下.

微软 SQLVM 项目组研究数据库即服务 (DaaS) 中多租户间的 CPU 有效共享和资源隔离问题, 提出了一种动态优先级调度算法——最大亏损优先级调度算法 (LDF)[22]. LDF 计算所有活跃租户的 CPU 赤字, 并使用贪婪启发式策略将 CPU 分配给 CPU 赤字最大的租户. 与传统的亏损轮询算法 (DRR) 和截止日期优先算法 (EDF) 相比, LDF 可以降低 20%–30% 的查询延迟, 能够更好地对 CPU 资源进行分配. 此外, 缓冲区的大小对云多租数据库的性能也至关重要. DaaS 提供商希望最大化使用缓冲区资源, 但又不影响租户的性能. 这就要求平台允许在所有租户之间动态共享内存. 因为这种模式会导致租户的实际工作负载内存命中率比静态预留内存时的命中率要低, 所以服务提供者需要为无法履行承诺的行为对用户进行补偿. SQLVM 项目组提出了“SLA-aware”[8]框架, 针对不同的租户开发不同的补偿函数. 在多租户的情况下, 页面替换算法必须满足两个要求: (1) 需要处理租户的不对称 (租户与租户之间的 SLA 协议是不同的), 即保证对服务提供者惩罚最小的同时对租户之间的公平性进行约束; (2) 必须灵活, 即可以针对不同的服务提供商目标进行优化. 微软提出了多租户页面替换算法 (MT-LRU) 可以显著降低对服务提供商的惩罚. 该算法不同于传统的页面替换算法, 提出了“影子价格”的概念. 与传统 LRU 算法中仅通过最近使用时间更新相比, “影子价格”的更新包含了时间和惩罚两个方面的考虑, 即满足了总体惩罚最小要求的同时又对不同租户之间的公平性进行了约束.

iBTune[23]在缓存资源方面做出了尝试, 通过学习相似负载能够容忍的缓存失效比, 基于缓存失效比和已分配的内存大小之间的关系来计算数据库实例缓冲区的大小. 目前, iBTune 还是依赖 DBA 在操作前人工分析系统扩展的需求, 无法自动的根据实际需求自动扩充缓冲池. 文献 [24] 针对云多租数据库中非易失内存 (NVM) 缓冲区管理提出了一个模拟器用于预测缓冲区操作过程中的 SLA 惩罚. 文献 [24] 将 NVM 作为除 RAM 和磁盘层的第 3 层存储, 逻辑的放在两者之间. 相比较于只有 SSD 和 RAM 的传统数据库缓冲池管理 (数据页要么已经在 RAM 中, 要么不在 RAM 中并从 SSD 中复制过去), 外加 NVM 存储层有了额外的选择: 数据页已经在 NVM 层中. 因为 NVM 也是由 CPU 进行访问的, 所以在页面访问率的表现上有了显著的提升, 从而可以降低 SLA 惩罚. 该方法的缺点是由于增加了额外的存储层, 导致层与层之间的搜索和通信成本增加, 如果可以利用 NVM 和 RAM 的字节寻址特性, 将可以达到更好的效果. 另外, 外加 NVM 存储层的方法更多的是硬件层面的进步 (扩大内存), 并没有在租户资源之间的分配上做出突破.

文献 [25] 针对大数据分析平台 (AaaS) 提出了可扩展的自动准入控制和利润优化资源调度算法. 该算法能够动态进行资源伸缩并最大化 AaaS 提供商的利润, 同时满足具有服务水平协议保证的 QoS 要求. 其算法的应用分为两个场景: 一种场景是查询有充足的预算和执行时间, 允许通过处理整个数据集来执行查询. 在这种场景下, 文献 [25] 采用两阶段的策略, 首先搜索资源注册表中所有可能的资源配置来判断某个查询的 QoS 能否被满足, 随后在资源满足的情况下寻求最低的 SLA; 另一种是租户的预算有限, 在这种场景下, 通过找出近似查询任务的特点对数据进行分割和抽样来分配相对应的资源数量. 类似的, 文献 [26] 也采用了类似分类的方法对资源进行调度, 提出不同租户之间的 QoS 要求通常存在不同的特征, 基于这些特征, 可以将租户和业务分别划分到不同的租户集群和业务集群中. 同一集群中的租户对 QoS 的要求相似, 而其他集群中的租户对 QoS 的要求不同. 基于这种相似性和多样性, 文献 [26] 提出了一种 5 阶段来匹配租户和资源: (1) 根据租户的 QoS 要求计算租户之间的相似度, 并将其划分为不同的租户集群; (2) 识别每个租户集群的 QoS 需求特征, 并将其映射到相应的业务 QoS 空间中; (3) 根据租户集群映射的 QoS 特征将各种资源进行组合; (4) 对各种资源组合依据其效果进行排名; (5) 以每个集群中效果较好的服务为代表, 进行保存. 文献 [25,26] 这种分类的方法为租户和资源之间的匹配提供了很好的思路, 但

是目前提出的分类方法只能针对专门的场景人工定制, 不能自适应地对租户进行划分, 适用性较差; 另外, 每一个租户的 QoS 在不同时间也会有不同的要求, 如何在一次分类后再次分类也是一个棘手的问题. HTAP 数据库技术作为数据库领域的一个研究热点, 其针对执行 OLTP 负载的实例和执行 OLAP 负载的实例之间的资源分配也值得关注.

HTAP 数据库需要同时支持执行 OLTP 事务和 OLAP 查询, 但这两种负载之间会存在 数据的同步与 CPU 资源的共享[27]. 如何平衡 TP 事务和 AP 查询的 CPU、内存等资源对提升系统整体性能非常关键. 文献 [28] 将事务、分析混合负载处理 (HATP) 看作是一种资源调度问题, 根据负载来动态分配资源以同时满足事务处理 (TP) 和分析处理 (AP) 请求. 其中引入了一个数据交换引擎 (RDE), 用于管理系统的 CPU 和内存资源, 并将它们分布到事务处理 (TP) 和分析处理 (AP) 实例中. TP 实例和 AP 实例使用各自的调度器来根据实际负载调度这些资源. RDE 根据状态决定 TP 实例和 AP 实例何时共享 CPU 或者内存资源, 文献 [1] 在针对事务处理的 CPU 资源调度时考虑了如何在减少延迟的同时降低能耗, 其提出的 POLARIS 算法针对事务截止时间和事务预测时间的建模对多租户的资源调度具有启发意义. 但是, 针对 HTAP 数据库的资源调度算法基本都没有同时考虑负载特点和系统行为趋势的特点, 缺少对资源的预测, 所以无法提前按照趋势进行资源调度, 系统整体也可能出现抖动.

大数据时代下, 面对不断膨胀的数据信息、复杂多样的应用场景、异构的硬件架构和参差不齐的用户使用水平, 传统数据库技术很难适应这些新的场景和变化[6]. 因此在数据库资源规划调度领域, 也有一些文献尝试采用深度学习的方式来解决相关问题.

文献 [29] 提出了一种基于机器学习 QoS 敏感的资源规划调度算法. 它使用了两个多层感知器和一个强化学习模型来进行调度空间探索, 并且可以有效地避开“资源悬崖”问题 (即有一些应用程序或租户在某些资源临界值两侧性能表现差异巨大). 该流程主要包括 3 个模型: Model-A 用于寻找特定服务的最优分配区域 (OAA) 和 Rcliff (资源悬崖界限); Model-B 用于 QoS 和分配资源的处理, 以期满足 QoS 的同时降低 SLA; Model-C 是一个在线学习模型, 动态地处理发生错误预测, 环境、用户需求变化, 资源共享等不可见的情况. 与之前的研究相比, 该机制主要有两个优点: 一是通过 Model-A 限定范围, 不使用代价昂贵的试错方法, 同时有效地避开“资源悬崖”问题; 二是可以根据用户的习惯做出针对性的微调, 实时收集信息并进行在线训练, 纠正前两个模型造成的预测错误. 文献 [30] 针对上述挑战提出了一种基于元学习模型的资源调优算法 ResTune. ResTune 针对不同硬件环境中的不同负载, 利用历史任务中的调优经验并结合积累的信息以加速新任务的调优过程. 该算法属于带约束的贝叶斯优化算法, 在传统算法中只通过吞吐量等信号约束的基础上, 将 SLA 也加入约束. 但与其他的数据库资源规划调度算法不同, 该算法并不会降低 SLA, 而是在保证预定 SLA 的基础上优化资源分配情况. 文献 [31,32] 提出了一种支持超卖的多租户数据库内存资源共享方法, 通过预测租户的内存需求区间, 按照区间上限为租户动态调整内存配额, 统一“拆借”空闲内存资源以支持更多租户, 实现内存超卖. 该方法在不影响用户业务运行时间的情况下, 支持的租户数增加了 2–2.6 倍, 资源利用率峰值提升了 175%–238%. 为了解决多租户数据库多维资源规划调度问题, 文献 [33] 提出了一种基于 MADDPG 的多租户数据库多维资源规划调度算法, 将多目标优化问题建模为马尔可夫博弈问题, 通过对租户各类资源利用率、系统整体资源利用率、SQL 响应时间、资源预定量、负载类型等信息设计编码形成状态空间, 并考虑了资源之间的相互影响. 相较于恒定策略、比例分配策略、基于 LDF 算法的策略, 论文提出的算法可以提升资源利用率 3%–29%, 降低资源闲置量 11%–34%, 降低 SQL 响应时间 32%–59%, 提高平台利润 16%–87%.

#### 3.1.3 小 结

现有的“不共享”多租户模式下的资源规划调度算法主要考虑负载实例或任务间的资源配置与调度. 它主要解决如何为租户任务配置合适的计算资源的问题, 资源规划调度的粒度通常较大, 常以资源槽、docker 或者虚拟机为单位, 而且一旦分配任务运行过程中通常不再调整. 它主要关注的是公平性、本地化等因素. “部分共享”或者“全共享”多租户模式下, 多租户会共享数据库的资源池, 资源共享、竞争的粒度更小、通常是动态的, SLA 要求苛刻, 所以算法通常也更为精细并且要求能够根据负载变化动态调整资源配给. 所以“部分共享”或者“全共享”多租户模式下资源规划调度相关工作有较多是基于机器学习的方法. 总体而言, 现有云多租数据库资源规划调度的工

作更多地关注了多租户间的资源共享, 对于租户间资源竞争的影响关注不多, 相关工作较少, 例如资源超售. 表 1 总结了这些模式下多租数据库系统资源规划调度方法的特点.

表 1 资源规划调度技术总结

| 方法 | | 特点 |
|---|---|---|
| “不共享”多租户模式下资源规划调度 | Capacity Scheduler[9] | 每个队列都会分配到一定额度的资源, 所有提交到队列的作业共享该队列中的资源, 但是作业之间的差异性并未体现 |
| | Fair Scheduler[10] | 把资源公平地分配给每一个作业, 同时优先处理本地作业 |
| | Quincy[14] | 把调度问题映射成图模型, 同时把公平调度方案转换成最小代价流问题. 但该方法的前提是每一个作业都是独立的, 并未考虑实际情况中作业之间的交互 |
| | CMMF[16] | 带有硬性约束, 可以刺激资源共享, 但是该算法只关注了单一约束, 并未关注多个条件的组合约束 |
| | DRF[17] | 将公平性定义拓展到多维资源的组合, 针对多维资源将Max-Min公平资源算法进行泛化, 但是该方法只关注资源的数量而没有关注资源的质量 |
| | Optimus[19] | 采用深度学习的方式进行资源管控 |
| “部分共享”和“全共享”多租户模式下的资源规划调度 | SQLVM [8,22] | 针对CPU和缓存的调度分别提出了相应的算法, 解决了云环境中租户之间的资源分配问题, 但是其提出的算法不能细粒度地对资源进行管控 |
| | 基于强化学习的调度方法[29,30,33] | 现有基于强化学习的调度方法大多只针对某类特定的资源或负载, 尚未形成体系 |
| | iBTune[23] | 针对关系数据库实例的资源进行规划调度, 没有考虑多租户共享, 且无法自动根据实际需求自动扩充缓冲池 |
| | 外加NVM存储层[24] | 通过硬件层面的进步显著地提升了页面访问率, 但是增加了层与层之间的搜索和通信成本, 并没有在租户资源分配上做出突破 |
| | 自动准入控制和利润优化资源调度算法[25,26] | 能够动态提供资源, 最大化AaaS提供商利润并满足QoS要求, 但只能针对人工定制的场景, 适应性差, 难以解决QoS要求频繁变化的情况 |

### 3.2 资源预估

准确预估负载的时变特征、负载所需的资源量、中间结果、运行时间或者系统未来的资源使用状态对于云多租数据库进行精细的资源分配和动态配置具有重要意义. 负载特征的预测是资源预估的前提, 性能预估可以帮助云服务提供商预测负载的资源需求并据此确定集群中可运行的服务数量、调整服务器的资源配置[30]. 本节将分析负载特征预测、负载资源需求预测方面的研究进展.

#### 3.2.1 负载特征预测

相对早期的一些查询执行时间预测算法, 如文献 [34–45] 等都存在诸多的局限性问题. 比如文献 [35,37,38,40,41] 只能预测单个查询的查询时间, 文献 [34,36] 可以预测多个查询的查询时间但是假设负载中的查询是事先已知的. 现代云多租数据库中的负载通常是动态并发的, 所以上述方法都已不再适用. 文献 [39,42] 提出了针对同时发生的多个动态负载执行时间的预测方法. 文献 [39] 中使用了优化器的代价模型来预测每一个查询的 IO 和 CPU 操作, 然后使用一个联合队列来关联 IO 操作和 CPU 操作来预测查询的执行时间. 文献 [42,43] 则主要根据资源的使用情况建模来预测查询的性能. 文献 [44] 针对特定负载预测了相同查询在不同数据上的查询时间. 文献 [45] 提出了一种基于统计学模型的 SQL 查询执行进度的预测方法. 但是上述方法都限于简单负载的预测, 当负载变得复杂时, 上述的传统方法均表现不好. 通常, 数据库使用统计数据估计成本和基数来达到预测执行时间的效果. 对于过滤器操作, 传统的方法使用直方图来估计时间. 一般情况下, 利用直方图可以进行有效的估计. 但是, 由于表之间的相关性, 连接产生的错误将非常大. 通常连接越多, 错误就越大. 与传统的成本估算方法不同, 在查询执行时间算法方面, 一些文献采用机器学习的方法取得了不错的效果.

文献 [21] 提出了一种基于机器学习的端到端估计器 (predictor), 通过学习多个列和表之间的相关性, 可以同时对关系数据库查询的基数和代价进行预测, 相较于传统的算法, 可以将准确度提高 3–4 倍. 如图 3 所示, Predictor 所采用的模型由 3 层组成: 嵌入层, 表示层和估计层. 首先, 每个平面节点的特征向量是大而稀疏的,

Predictor 需要对其进行压缩, 提取特征的高维信息, 从而嵌入层对每个平面节点嵌入向量. 其次, 表示层采用树状结构模型, 每个节点都是一个表示模型, 树状结构与查询计划相同. 每个表示模型学习对应自己的两个向量 (全局向量和局部向量), 其中全局向量获取根节点计划的信息, 局部向量获取该节点的点的向量和对应节点的特征向量来学习这两个信息. 每个表示模型的节点都是基于其两个子节向量. 最后, 基于根节点的向量, 估计层对代价和基数进行估计. 同时, 文献 [21] 也提出了针对查询 SQL 语句有效的编码方式. 但是这种方法对训练数据的要求高, 且训练时间过长. 文献 [46,47] 提出一种基于深度学习网络的关系数据库负载基数预估模型. 模型本质就是通过查询的相关信息进行特征编码, 输入多集卷积网络后进行预测. 但是这种方法并不适用于查询优化, 因为在优化属性结构时, 基于查询的编码过于复杂, 并且这种方法的泛化是有限的. 文献 [48] 提出了一种基于混合模型的查询的选择性估计模型. 这些模型可以为负载资源消费模型的构建提供支撑. 相比较传统基于直方图的查询预测, 混合模型 QuickSel 可以从每个查询中学习, 不断完善其底层数据的内部模型, 生成越来越准确的选择性估计. 文献 [49] 对数据库系统未来负载的到达模式进行了预测. 相比较之前对系统资源需求或数据库的工作负载进行建模的思路, 文献 [49] 提出基于查询到达率时间的预测方法: QB5000, 使得训练模型可以独立于硬件和数据库设计, 无需重复地构建模型. QB5000 的工作流程主要由两个阶段组成. 当 DBMS 向 B5000 发送一个查询时, 它首先进入预处理器和集群组件, 将原始的 SQL 查询转换为通用模板, 随后记录每个模板的出现次数. 同时为了进一步减少计算资源的压力, QB5000 根据模板的语义将模板映射到最相似的查询组, 使用在线集群技术进一步压缩工作负载, 将具有相似出现次数的模式的模板分组在一起. 实验结果表明, 通过压缩 5 个最大的集群就可以覆盖 95% 的查询量. 最后, 预测模块针对每个集群的平均出现次数训练预测模型. 这些模型会预测应用程序将来在每个模板集群中执行多少查询, DBMS 可根据它对应用程序未来的期望决定如何优化自己. 在预测模型的选择上, QB5000 采用集成方法, 将线性自回归模型 (LR) 和长短记忆网络 (LSTM) 相结合, 以保证在短期和较长期都有较好的预测结果, 此外, 由于 LR 与 LSTM 对预测峰值的效果很差, QB5000 还集成了 KR 模型, 用于预测特定情况下的峰值数据. 实验表明, 该混合模型 (HYBRID) 在 61% 的工作场景中表现良好, 优于 LR、ARMA、FNN、RNN 等传统的预测方法. 上述相关工作都是在预测系统负载的宏观特性. 不同于之前的预测模型只针对自己的工作负载历史跟踪, 不考虑 VM 之间的相关性, 文献 [50] 提出的一种基于深度学习的虚拟机工作负载预测方法, 可以从云中的所有 VM 的工作负载数据中学习固有特性. 但是, 文献 [50] 提出的方法预测时间间隔较长, 不灵活, 且相较于传统方法开销过大.

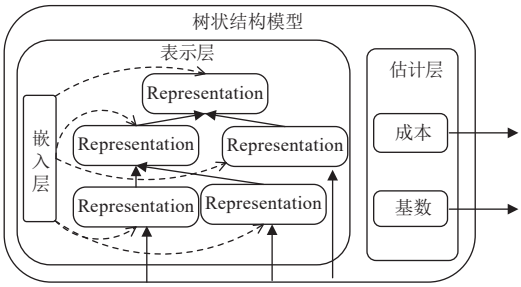


图 3 文献 [21] 所提出的树状结构模型图

### 3.2.2 负载资源需求预测

负载动态变化的特征预估是资源建模的前提. ARIMA[51]模型可以抓取负载或 CPU 利用率等时间序列特征并且结合这些特征预测未来 CPU 的使用情况. 文献 [52] 采用双重指数平滑的方法预测 CPU 和内存的利用率. DBSeer[53,54]对高并发事务处理系统不同事务的 CPU、RAM、磁盘 IO、锁等资源的消费模型进行了建模, 可以预测不同事务组合的情况下系统资源的消耗情况. DBSeer 提出了一系列资源预测模型. 对于 CPU、日志写等, 基于回归算法的黑盒模型可以有效预测, 即使在预测资源利用率偏差较大的情况下, 或者在不同的事务随机组合的情况下, 也只产生几个百分点的相对误差, 这是在可以接受的范围内. 对于其他资源, 例如 RAM 利用率、页面刷新以及锁, 在预测与训练集的偏差较大下的系统性能时, 需要对数据库建模的白盒模型. DBSeer 的白盒模型包括基

于日志旋转的迭代模型, 以及对锁争用模型一系列优化. 文献 [18] 提出了一种新的针对 CPU、I/O 资源估计统计模型. 该方法将回归树模型与缩放函数进行结合, 前者用于常规情况下的预测, 后者主要通过 SQL 查询中的“高级特征”(如某些特殊的查询条件) 来做到针对特殊极值进行预测. Liquid[55]是一个针对深度学习任务的 GPU 资源需求估计和调度管理平台, 其中提出了一种基于回归模型的负载资源需求估计方法, 以避免为用户过度分配计算 GPU 资源. 文献 [56] 为 spark SQL 查询开发了一个细粒度的 CPU 资源调度器. 它可以为 spark SQL 查询进行细粒度的 CPU 资源建模.

#### 3.2.3 小 结

上述的这些方法基本上都是针对特定数据管理引擎或者特定类型的负载进行预测, 有些模型并不是直接预测资源需求或者预测资源需求时并没有考虑租户的性能要求, 预测模型上有很大的改善空间. 目前鲜有一个可以根据负载、引擎、租户 SLA 等多方面资源需求的预估模型. 表 2 分别总结了预测查询执行时间算法和基于负载的预测算法.

表 2 资源预估技术总结

| 方法 | | 特点 |
|---|---|---|
| 负载特征预测 | 预测单个查询的查询时间[1,35,37–41] | 早期的查询执行时间的预测算法, 可以预测多个查询的查询时间但是假设负载中的查询是事先已知的, 不适用于现代云多租数据库动态并发的负载 |
| | 预测多个动态负载执行时间[39,42] | 使用统计数据估计成本和基数来预测执行时间, 但是由于表之间的相关性, 连接会产生非常大的错误 |
| | 基于机器学习的端到端估计器[21] | 可同时对关系数据库负载基数和代价进行预测, 准确度提高3–4倍, 但对训练数据要求高, 训练时间长 |
| | QB5000[49] | 基于查询到达率时间的预测方法, 训练模型可以独立于硬件和数据库设计, 无需重复地构建模型, 适应性强, 但只针对自己的工作负载历史跟踪, 没有考虑到VM之间的相关性 |
| | 基于深度学习的虚拟机工作负载预测[50] | 可以从云中的所有VM的工作负载数据中学习固有特性, 但预测时间间隔较长, 不灵活, 开销过大 |
| 负载资源需求预测 | ARIMA[51]模型 | 可以抓取负载或CPU利用率等时间序列特征并且结合这些特征预测未来CPU的使用情况 |
| | DBSeer[53,54] | 可以预测不同事务组合的情况下系统资源的消耗情况, 适应性好, 准确率高 |
| | Liquid [55] | 提出了一种基于回归模型的负载资源需求估计方法, 能避免为用户过度分配计算GPU资源 |

### 3.3 资源弹性伸缩技术

负载是动态变化的, 随着时间的推移会出现资源和负载不匹配的状态. 不匹配的状态主要有 3 类[57]: (1) 资源紧缺, 即获取的资源不足以运行现有负载; (2) 资源充足, 即获取的资源在运行现有负载的情况下有结余; (3) 震荡状态, 即资源时而不够, 时而有结余. 在一些典型的应用场合, 例如“双十一”“黑色星期五”, 数据库系统可以由专家根据经验人为调整资源配置. 但是大多数场景下负载增减无法有效预知, 所以无法根据负载人工调整资源供给. 这就要求系统必须具有根据负载变化自动调整资源供给的能力.

资源弹性伸缩主要包括 4 个过程[57]: (1) 监控: 收集资源使用情况及负载情况; (2) 分析: 分析资源和负载的匹配情况, 并且对未来资源使用情况进行预测; (3) 计划: 制定资源的调整计划; (4) 行动: 对资源进行调整. 其中监控指标的质量和获取的代价对于资源自动调整的性能至关重要[58]. 文献 [59] 将资源动态伸缩问题归结为建模、结构、粒度和决策这 4 类问题. 其中建模是指资源管理系统自我感知、自我调整的能力, 结构即资源伸缩系统处理过程、与其他组件的交互等, 粒度即资源调整的范围和边界, 决策就是决定何时调整和如何调整. 现有的资源弹性伸缩技术的研究大多围绕这 4 类问题展开. 基于此, 本节将对 5 类主流的资源弹性伸缩方案进行总结分析.

#### 3.3.1 资源弹性伸缩算法

资源弹性伸缩算法划分为 5 类[57]:

(1) 基于阈值规则的方案[60–69]. 被占用的资源到达事先设定的阈值就进行资源的自动调整, 例如如果 CPU 的

利用率持续 5 min 超过 70% 就需要增加 2 份资源实例. 此类方案的机制简单并且易实现, 比如文献 [61] 所采用的 LS 算法就以 SLA、QoS 为指标设定上下界限的方式来控制虚拟机所拥有的资源数量. 文献 [60] 提出一个基于语义的自主监控和管理工具 SAMM, 通过监控一组提前定制的高级指标来出发规则, 提醒系统管理员进行相对应的资源伸缩操作. 与文献 [60] 相类似, 文献 [58−61] 都是采用预先设定的指标 (如 SLA, QoS, 各类资源使用情况) 作为参考, 以资源利用率、资源整合度、空间开销、成本高低作为最终目标. 但是, 以资源阈值规则作为缺点明显, 即无法针对先行多租户复杂的情况进行灵活更新, 在资源震荡状态下反复伸缩, 造成不必要的操作, 增加额外开销. 另外, 上述诸多算法都是针对特定的虚拟场景, 未在实际的情况下运用, 局限性很大.

开源系统中, Kubernetes[68]作为有一个开源平台, 可以轻松打包和运行容器化的应用程序、工作负载和服务, 并且为分布式系统框架提供可扩展服务, Kubernetes 采用基于阈值规则的方法进行资源弹性伸缩. 它提供了一个控制平台来周期性地检查 CPU 或者内存的使用率, 当超出某一个阈值时或某一指标过低时, 便对 Pod 的数量 (可以抽象理解为资源实例) 进行增加或减少. 同时, 不仅仅是调控 Pod 的数量, Kubernetes 还详细解释了如何收集、计算和提取每种资源类型的指标. 文献 [69] 中关注了如何通过评估租户工作负载的资源需求来实现数据库容器的弹性伸缩功能. 它主要包括 3 个组件: 1) 数据收集模块: 为每个租户的数据库收集相关数据 (也称为信号), 包括: 延迟、资源利用率、资源等待时间、资源上升/下降趋势、资源之间的相关性等; 2) 资源预估模块: 通过利用数据库引擎内部的逻辑关系和资源如何交互来组合信号, 它提出了一个决策逻辑, 使用数据收集模块收集到的多组信号来确定租户工作负载的资源需求. 基于规则的逻辑核心是为每个信号设置一组阈值, 以单独确定每个信号的状态, 随后结合每个信号的状态来确定总体资源需求. 通过遍历规则层次结构, 就可以在逻辑上决定是否需要为某个租户扩大/缩小资源; 3) 计算下一计费间隔的预算: 在知道下一计费间隔所需的资源后, 该模块会计算相对应资源的“价格”以保证 SLA, 为分配设置限制 (以免预算过高). 通过监视延迟、资源需求和可用预算来确定下一个计费间隔的容器大小, 从而实现自动扩展. 但是, 文献 [69] 并不能进行“旋钮”级别调控, 只能设定几个层级的容器大小进行“伸缩”. 此类方法的机制较为简单并且容易实现, 但是效果主要依赖所选择的性能参数的阈值以及负载的变化幅度.

(2) 强化学习的方案[70−81]. 与之前提出的基于阈值规则来改变容器数量的方法不同, 有一些文献尝试了利用强化学习模型来探索较优的资源调整时机及资源调整的数量, 以指导系统的弹性伸缩. 文献 [70] 实现了一种非集中式的基于强化学习模型的自动缩放方法来扩大或缩小 Web 应用作业资源. 它将系统状态和应用程序状态建模为正常、警告和异常这 3 类, 并将动作空间划分为缩小和放大两类. 通过周期性地对服务器相关指标进行采集, 无监督的强化学习模型可以不断学习服务器的状态, 从而达到对服务器状态的可预测, 就可以增加或者缩减服务器数量来实行扩容或缩容. 文献 [71] 中考虑了云平台、负载及用户行为的不确定性, 针对 Web 应用场景提出了一种基于异常检测的深度强化学习资源伸缩框架——ADRL. ADRL 定期采集每个虚拟机的资源利用率指标, 监控虚拟机的性能, 将收集到的数据输入前馈神经网络进行预测. 与传统直接根据预测结果来进行资源伸缩的方法不同, ADRL 通过检测异常的预测值来触发强化学习模型做出伸缩决策, 即不会每时每刻都采取伸缩动作. 此外, 因为异常状态和正常状态之间的转换可能只是某些峰值出现的结果, 为了解决这种抖动问题, 一个异常的警报不会被视为严重的异常事件, 只有当接收到至少连续 $N$ 个警报时才会触发决策. 因此该系统会忽略最初的几个警报, 以避免对短暂更改做出不必要的反应, 降低振荡性. 但是文献 [71] 只是针对了其模拟场景进行了实验, 能否在其他类型的更复杂的应用程序运行还有待商榷. 文献 [72] 为 Kubernetes 研制了一款基于强化学习模型的开源弹性伸缩工具, 支持 Serverless Kubernetes 集群的横向弹性伸缩 (资源的横向扩展是指增减虚拟机的数量, 资源纵向扩展是指控制某一具体虚拟机可用的 CPU 等资源多少), 但在实际的应用场景中, 集群横向的资源扩展粒度还是过大, 无法很好地满足现在场景细粒度的需求. 文献 [71−74,76−78] 都认为强化学习的方法相对于基于阈值的方法以及时间序列的方法能够更有效地应对系统的动态性. 但是上述文献都是独立采用强化学习, 仅是应用场景和参数的相关调整, 在此不再赘述. 文献 [72] 针对常见的基于强化学习的方法进行了对比实验, 得到了如下的几个经验: 1) 对于控制器所采取的所有行动, 被控系统都需要有足够的时间来反馈被控系统稳定后的真实状态; 2) 确保调整系统

比稳定系统不会用更多的时间, 在这种情况下, 系统可能会在一个决策后马上触发反向决策, 从而进入一个无限循环, 使系统不稳定; 3) 强化学习的扩展性较差, 其搜索空间随状态变量呈指数级增长; 4) 由于缺乏相关的领域知识, 强化学习在初始的性能可能非常差, 它需要相当多的探索行动. 针对文献 [72] 所发现的经验 1)、2), 文献 [81] 提出可以通过动态预测负载提前对数据库扩容或者缩容, 而不是等到负载发生变化之后再弹性伸缩, 并研制了支持弹性伸缩的数据库 P-Store. P-Store 中提出了一个时间序列模型用于预测负载并且采用了一个动态规划模型用于减少数据库资源弹性伸缩的代价. 这类方法不需要任何关于应用的先验知识和模型, 而是通过试错的方法为每一种特殊状态找到合适的资源调整方案. 此类方案存在初始性能差、训练时间长、状态空间大、探索动作和适应性需要根据环境变化等缺点. 针对文献 [72] 所发现的经验 3)、4), 文献 [75] 提出了一种混合方法, 它结合了单纯形 (Simplex) 方法和强化学习方法的优点. 该方法不是在整个可配置的状态空间中进行 RL 探索, 而是首先使用 Simplex 方法将搜索空间缩小到一个更小但“有希望的”状态集合. 同时为了避免随机探索导致的性能下降, 文献 [75] 对强化学习部分进行了先验知识引导探索动作的选择. 该混合方法很好地减少了搜索空间, 提高了探索阶段的性能, 针对独立使用强化学习的方法, 可以将系统吞吐量提高 30% 以上.

(3) 基于排队论的方案[82−88]. 基于排队论的方法通常会将云数据库基础设施看成一种排队系统, 系统中服务机构的 CPU、内存等资源是可变的. 多租户的负载进入排队系统并且被处理. 该类方法的核心是根据负载到达情况及资源需求动态调整服务机构的资源配给以满足租户的 SLA(逗留时间、等待时间等) 及服务水平 (例如: 平均响应时间、资源利用率等). 基本框架如图 4 所示. 文献 [82] 提出的模型中, 假设队列是稳定的, 那么一段时间内的平均服务速率等于一段时间内的平均到达速率, 当系统由于需求的变化而带来资源不稳定的问题时, 弹性控制器考虑资源异构、请求延迟、虚拟机服务速率可变等因素以增加或删除虚拟机的数量来保持队列的稳定. 文献 [87,88] 与文献 [82] 的思路基本相同, 都是基于排队论的资源调度方法, 在此不再赘述. 文献 [85] 提出了历史排队方法, 即在排队方法的基础上加入了历史建模技术 (HYDRA), 通过采集历史数据输入提前定义的模型, 可以解决传统基于排队论需要大量 CPU 时间来进行平均响应时间计算的问题, 提高了整个系统的稳定性和实时性. 相对于文献 [82] 直接将响应时间作为评判标准, 文献 [85] 把平均到达时间抽象成一种成本, 使其可以直接融入进 SLA 的评判体系, 无需在以平均响应时间为标准的基础上再去考虑 SLA 的限制, 让整个系统资源调度更加简洁. 此外, 由于将平均响应时间抽象加入 SLA, 使得调度系统可以统一细粒度地进行定价 (从按小时计费发展成按分钟计费甚至按秒计费). 文献 [86] 针对云平台峰值工作负载资源分配的问题提出了一种基于预测的排队论方法, 其主要包含两个部分: 1) 预测性和反应式供应: 文献 [86] 提出的资源供应技术采用了两种在两个不同时间尺度上运行的方法——以小时或天为尺度的预测性供应和以分钟为尺度的反应式供应, 用以解决预测长期行为和短暂峰值出现的问题; 2) 基于排队理论的灵活分析模型: 该模型主要针对最坏情况设计的, 所以使用了工作负载分布尾部的概率分布来估计峰值需求. 文献 [86] 在一定程度上通过预测的方式解决了之前基于排队论调度方法应用场景固定的问题, 但是由于在模型参数等方面仍然需要人为干预, 故无法作为一个全自动资源调度系统进行使用. 这类方法把使用该资源的作业序列看成一个队列, 通过这些作业到达时间和离开系统时间之间的关系来动态调整资源. 但是这类模型结构一般比较固定, 适合特定的场景, 并不适用于负载和资源池动态变化的场景.

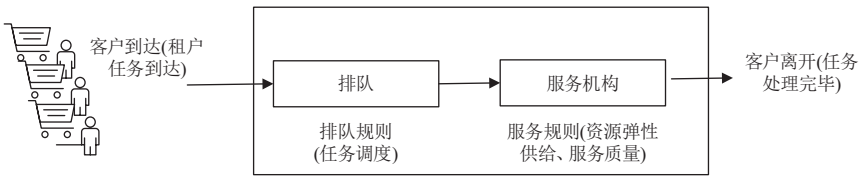


图 4 基于排队论的资源弹性伸缩方案框架

(4) 基于控制论的方案[89−93]. 此方案将学习与反馈机制引入到资源的弹性伸缩过程中, 将当前策略下的调度效果反馈到模型, 帮助输出是下一个阶段资源的调整策略. 文献 [89] 提出了一种基于服务模式的弹性调节方法, 它开发了自适应水平弹性控制器, 用于控制缩放决策和防止资源振荡. 弹性控制器的输出是应添加或删除虚拟机的

数量, 用以确保服务请求的数量等于资源所需的数量且符合 SLA 协议, 同时保持虚拟机的数量最低. 此外, 文献 [89] 还为弹性控制器增加了一个预估器, 对输出的结果加以矫正, 避免出现资源动荡的问题. 文献 [90] 设计了一套基于统计机器学习 (statistical machine learning, SML) 的建模、控制和分析技术, 该技术主要包含以下 3 个组件: 1) 性能统计模型, 可以预测未来配置和工作负载的系统性能; 2) 控制策略模拟器, 为了解决传统基于控制论只针对 SLA 限制而不能找到保持性能且最小化资源使用的控制策略, 该组件从性能统计模型中进行模拟, 以比较添加和删除资源的不同策略的效果; 3) 在线训练, 为了增强鲁棒性, 该技术加入了在线训练和变更点检测, 可以根据性能变化而及时调整模型. 文献 [90] 提出的方法可以很好地解决单纯基于控制论无法在保持性能的同时最小化资源使用的问题. 文献 [91,92] 与文献 [90] 的思路类似, 都是在采用控制论方法的基础上融入机器学习的思路, 在此不再赘述. 但是, 由于机器学习自身需要大量训练数据的原因, 该方法在场景有较大转变的时候无法提供稳定的方案. 文献 [93] 提出了一种专门针对数据库资源管理系统的自适应模糊建模方法. 模糊建模将模糊逻辑与数学方程相结合, 描述已发现的系统行为模式, 指定系统的控制策略. 模糊模型是一个规则库, 由“如果 x 是 A, 则 y 是 B”形式的模糊规则集合组成. 与清晰集相比, 模糊集允许部分集隶属度, 可以根据隶属度函数将其量化为数值. 通过引入模糊建模的方法, 文献 [91] 可以更好地针对数据库查询负载既有可能是 CPU 密集型也有可能是 I/O 密集型或者是两者混合的问题场景, 能够快速适应由于工作负载强度和组成变化而引起的资源需求动态变化. 与基于峰值负载 (传统控制论方法) 的分配相比, 模糊建模可以节省大量的资源 (约 62.6% 的 CPU 资源和 76.5% 的磁盘 I/O 带宽). 这类方法试图为应用建立一个模型, 通过定义一个控制器来为应用动态地调整资源. 此类方法的实用性主要依赖于选择的控制模型类型以及系统的动态性, 另外, 建立一个与输入输出参数相匹配的模型是一件很困难的事情.

(5) 时间序列分析方案[32, 94−100]. 此方案主要思路是通过检查数据库过去使用情况并提取相似的使用情况来推断未来数据库的系统资源使用情况, 根据预测的资源使用情况对系统资源进行扩张或者收缩, 此类方法的核心是资源使用情况的预测. 文献 [94] 通过历史数据的模式对应当前系统的使用模式, 再通过插入历史数据的方式确定当前模式之后的内容来预测系统的使用情况. 文献 [95] 与文献 [94] 的思路基本相同, 都是将工作负载分类为一组历史的工作负载类之一. 文献 [95] 提出的模型中, 将工作负载表征为两种分布: 请求到达过程和服务需求分布, 并利用线性回归和滑动窗口技术, 测量在一定时间内控制这些分布的各种参数, 并使用测量值预测下一个适应窗口的工作负载. 文献 [97] 在文献 [95] 的基础上, 将线性回归和神经网络进行比较, 并在训练和预测阶段结合了滑动窗口和交叉验证技术, 通过实验证明了神经网络模型预测优于线性回归, 且滑动窗口技术更适用于神经网络预测模型. 文献 [96] 提出采用两种互补的技术来预测短期的资源需求, 对于具有重复模式的工作负载, 模型采用信号处理技术来识别用于预测的重复模式, 称为签名, 并利用该签名进行预测; 对于没有重复模式的工作负载, 模型采用统计状态驱动的方法来捕获资源需求的短期模式, 并使用离散时间马尔可夫链来预测近期需求. 文献 [100] 与文献 [96] 的预测模型基本相同, 都采用了签名驱动和状态驱动的混合预测算法, 在此不再赘述. 文献 [98] 提出的模型首先使用布朗二次指数平滑法准确预测应用程序的未来工作负载, 并通过实验发现布朗的二次指数平滑方法以少量误差产生了良好的估计. 文献 [99]使用了二阶自回归移动平均法 (ARMA) 滤波器对未来负载进行预测, 取得了不错的预测效果. 这类方法根据时间序列上一些已知的数据点的使用情况来检测资源使用的模式、预测未来数据点的资源使用情况. 此类方法的优点在于可以提前预测未来资源需求从而提前做好准备, 但是此类方法预测的准确率受到应用的特点、输入的负载的模式、选择的指标、历史窗口、预测间隔等诸多因素的影响. 文献 [32] 中提出了基于内存负载预测的多租户数据库弹性伸缩方法, 其中的负载预测模型融合了卷积神经网络、长短期记忆网络和门控循环单元的优势以期望精确预测数据库集群内存负载需求, 然后基于需求预测结果调整虚拟机数目. 论文模型预测误差降低 8.7%–21.8%, 预测拟合度提高 4.6%. 与 Kubernetes 中应用最广泛的伸缩策略相比, 论文弹性伸缩方法响应时间降低 8.12%, 延迟降低 9.56%.

#### 3.3.2 小 结

通过分析主流的 5 类资源弹性伸缩方案, 可以发现, 上述的方案都分别适用于一些特定的场景, 而且各有不

足. 强化学习是云平台 AI 赋能的资源弹性伸缩方案中所采用的主流模型, 但是目前的工作主要考虑资源的横向拓展, 而现代云多租数据库系统中多引擎、多租户共享 CPU、缓存等模式会同时带来资源纵向的扩展需求. 另外, 比较新的方法大多没有在实际的生产系统中验证. 大数据生产系统中环境、负载及用户行为的动态性会给学习模型的适用性带来极大的挑战. 表 3 总结了上述 5 类方案.

表 3 资源弹性伸缩技术总结

| 研究方向 | 特点 | 缺点 |
|---|---|---|
| 基于阈值规则的方案 [60−69] | 实现机制简单, 被占用资源达到事先设定的阈值就自动调整 | 效果主要依赖所选择性能参数的阈值以及负载的变化 |
| 强化学习的方案[70−81] | 不需要任何关于应用的先验知识和模型, 通过试错的方法为每一种特殊状态找到一个合适的资源调整方案 | 初始性能差、训练时间长、状态空间大、探索动作和适应性需要根据环境变化 |
| 基于排队论的方案[82−88] | 把使用该资源的作业序列看成一个队列, 通过这些作业的到达时间和离开系统时间之间的关系来动态调整资源 | 模型结构一般比较固定, 不适用于负载和资源动态变化的场景 |
| 基于控制论的方案[89−93] | 为应用建立一个模型, 通过定义一个控制器动态地调整资源 | 建立相匹配的模型或控制器难度较大, 可行性较低 |
| 时间序列分析方案[32, 94−100] | 根据时间序列上一些已知数据点的情况来检测资源使用的模式, 预测未来数据库资源使用情况 | 预测的准确率受到应用的特点、输入负载的模式、指标的选择、历史窗口、预测间隔等诸多因素的影响, 对建模要求较高 |

### 3.4 资源规划调度工具

#### 3.4.1 通用云平台的资源规划调度工具研究现状

“不共享”多租户模式下数据库资源规划调度主要依赖于云平台的资源管控能力. 目前, 国内外针对通用云平台资源规划调度工具方面有 5 种主流方案, 主要研究情况如下.

(1) 集中式调度. 所有资源都由资源管理器直接统一管理, 所有系统的可用信息都聚集在资源调度节点上[4]. 因为资源管理器可以获知整个集群的资源使用状态, 所以这种方案可以获得全局最优的调度方案. 但是资源调度节点负载大, 扩展性差. 传统的关系数据库以及 Google 早期的资源管理系统 Borg[101]都采用此类模式.

(2) 完全分布式调度. 全面去中心化, 具有如下特点: 1) 逻辑比较简单、决策快速, 适合一些特定的工作负载和场景; 2) 调度过程中使用的集群信息是本地的、片面的、过时的, 难以进行全局最优决策; 3) 该方案忽视了工作负载的异质性, 所有任务被划分给了相同的时间槽, 任务不存在优先级. 完全分布式调度的典型代表是 Sparrow[102].

(3) 两层调度. 资源管理器负责将资源分配给各个数据管理引擎, 然后由这些引擎管理自己所得的资源. 因为资源管理器只负责将资源推送给各个数据管理引擎, 并不负责每一个具体任务的资源分配, 所以较之前集中式调度方案而言, 其负载大大减轻. Yarn[103]和 Mesos[104]就是典型的两层调度方案. 但是, 两层调度采用的是类似于悲观锁协议, 资源一旦分配给某一个数据管理引擎, 任务结束前其他引擎都无法使用. 另外, 由于数据管理引擎只能被动地接受或者选择资源, 所以很难获取到全局最优的资源配置方案.

(4) 混合式调度. 根据文献 [105] 中的调研结果: Google 系统中超过 80% 的负载是批处理作业 (短作业), 但是 55%–80% 的资源分配给了长服务 (长作业). 长作业迁移代价高但是对于延迟的容忍度高, 短作业迁移的代价低但对延迟容忍度也同样低. 集中式调度能够采用全局较优的资源分配策略, 适宜调度长作业. 而分布式调度并发度好、延迟小, 适宜调度短作业. 为了提高混合负载的调度效率, 混合式调度结合了两者的优点, 对长作业采用集中式调度机制, 对短作业采用分布式调度机制. Hawkp[106]和 Mercury[107]是两种典型的混合式调度工具.

(5) 共享状态调度. 最理想的资源管理与调度工具, 没有单点故障和性能瓶颈、资源分配全局最优、扩展性好. 文献 [105] 认为允许数据处理引擎自由竞争整个集群的所有资源、采用乐观锁的机制可以解决两层调度的主要问题, 并据此提出了 Google 新一代资源调度工具 Omega. Omega[105]不设置负责总协调的资源管理器, 而是由每个数据管理引擎保存全局资源的使用状态并根据各自保存的其可以使用的资源的状态来确定分配方案. 因为数据

管理引擎保存了其可以使用的所有资源的状态, 所以多数场合下可以获得全局最优的资源分配方案. 采用这种方案不用等待其他数据处理引擎的作业, 所以提高了并发度. 但是, 当冲突较多时会增加重调度的几率从而增加系统开销.

虽然针对云计算环境下的资源规划调度技术已经有了较多研究和成熟的商业软件, 但是上述的方案都不是专门针对数据库实例设计的, 资源分配的粒度是作业而不是租户, 无法深入考虑租户之间的负载信息差别, 也无法保障资源差异化供给.

各类工具的优缺点如表 4 所示.

表 4　云平台资源规划调度工具

| 工具类型 | 特点 | 缺点 | 典型代表 |
|---|---|---|---|
| 集中式调度 | 所有资源都由资源管理器直接统一管理 | 资源调度节点负载大, 扩展性差 | Borg[101] |
| 完全分布式调度 | 全面去中心化, 逻辑简单、决策快速 | 调度过程使用的信息片面, 难以进行全局最优决策, 不考虑任务的优先级 | Sparrow[102] |
| 两层调度 | 资源管理引擎只负责将资源分配给各个数据管理引擎, 并不负责每一个具体任务的资源分配, 负载大大减轻 | 资源共享不彻底, 数据管理引擎之间不能共享资源; 难获取全局最优策略 | Yarn[103], Mesos[104] |
| 混合式调度 | 结合集中式调度和分布式调度, 可以得到全局较优决策 | 实现机制较为复杂 | Hawkp[106] , Mercury[107] |
| 共享状态调度 | 数据管理器保存全局和各自的资源使用状态, 无需等待其他数据处理引擎, 提高了并发度 | 在冲突较多时会增加重调度的几率从而增加系统开销 | Omega[105] |

### 3.4.2　云数据库资源规划调度工具研究现状

数据库厂商在多租户资源规划调度技术方面提出了自己的方案, 以下是针对常见方案的总结, 见表 5.

(1) Greenplum[108]是一款广泛应用的开源 MPP 数据库的产品, 兼容 PostgreSQL 生态, 被广泛应用于大数据的存储与分析. Greenplum 采用资源组 (resource group) 进行资源管理, 能够细粒度定义限制不同数据库角色 (用户) 的资源使用. 超级用户可以在数据库内定义多个资源组, 并设置每个资源组的资源限制. 在一个数据库中, 每个资源组可以关联一个或多个数据库用户, 而每个数据库用户只能归属于单个资源组. 控制组 (control groups, Cgroups) 是 Linux 内核提供的一种可以限制、记录、隔离进程组所使用的物理资源 (如: CPU、内存、IO 等) 的机制. 如果一个进程加入了某一个控制组, 该控制组对 Linux 的系统资源都有严格的限制, 进程在使用这些资源时, 不能超过其最大限制. 资源组通过 Master 上的并发锁实现对并发的限制, 通过 Cgroup 实现对 CPU 的限制, 通过 vmtracker(内存分析工具) 和 Cgroup 两种方式对内存进行限制. 资源组对资源的管理更加细颗粒度, 并且实现了组内内存共享, 全局内存共享等多个级别的资源灵活使用.

(2) OceanBase[109] 数据库是多租户的分布式数据库, 用户可以在一个集群内部创建多个相互独立的租户, 每个租户是一个独立的数据库服务, 用户可以指定需要的硬件资源, 创建数据库服务的账号、表格、存储过程等对象. 租户的硬件资源以资源池 (resource pool) 的形式进行描述, 资源池是由资源单元 (resource unit) 组成, 资源单元是数据库服务内部分配的 CPU、内存、硬盘等硬件资源. OceanBase 数据库集群由一个或多个 Region 组成, Region 由一个或多个 Zone 组成, Zone 由一台或多台服务器组成. OceanBase 数据库会将一个租户的资源单元在集群的服务器间均衡分布. 一个租户拥有的一个或多个资源单元, 在每个 Zone 内资源单元的基本均衡策略如下: 同一个租户的若干个资源单元, 会均匀分散在不同的服务器上; 当服务器存储使用率超过一定阈值时, 通过交换或迁移资源单元, 以降低服务器资源压力; 根据资源单元的 CPU 和内存规格, 通过交换或迁移资源单元, 降低服务器 CPU 和内存的使用压力; OceanBase 实现了节点间自动负载均衡, 能进一步支持多种业务场景下负载均衡需求.

(3) openGauss[110]数据库资源负载管理的核心是资源池, 而配置资源池首先要在环境中实现控制组 Cgroup 的设置. 资源池 (resource pool) 是 openGauss 提供的一种配置机制, 用于对主机资源 (内存、IO) 进行划分并提供 SQL 的并发控制能力. 资源池通过绑定 Cgroup 对资源进行管理. 用户通过绑定资源池可以实现对其下作业

的资源负载管理. 针对 CPU 资源调度, 华为西安研究所与西北工业大学提出了两种兼顾收益、公共性及整体性能表现的资源规划调度方案. 具体包括: 1) 定额分配、比例分配两种资源分配策略, 以及最大赤字调度、收入调度、感知调度这 3 种调度算法. 实验结果表明: 收入调度在多租户资源竞争场景下, 可以较大程度提升供应商收入; 感知调度在大部分场景下可以大幅提高 CPU 利用率. 因此将两者有效结合, 是一种可以兼顾公平、收入以及整体性能表现的做法; 2) 实现了基于强化学习的 CPU 资源规划调度算法. 该方案对 CPU 利用率、租户 SLA 等信息进行编码, 形成状态, 并给予 CPU 利用率、查询延迟、动作有效性等反馈奖励, 构建用于该场景的 Q-learning 模型. 实验结果表明: 模型针对 openGauss 运行 TPC-C 负载, 可以提升整体 CPU 利用率 3.42%, 降低查询延迟 3.14%.

(4) Oracle[111]数据库利用 Oracle 资源管理器 (Oracle database resource manager, DBRM) 管理数据库资源, 为不同的会话分配不同的数据库资源. Oracle 资源管理器允许数据库控制硬件资源的分配, 在多用户并发执行不同优先级的任务时, 不同的会话请求可以按不同的优先级对待, 资源使用组由许多用户会话组成, 这些会话有相同的资源使用请求. Oracle 资源管理器允许我们根据会话请求的属性将这些请求分类到不同的分组, 然后对不同的分组分配不同的资源, 最大化提高数据库应用性能.

(5) 在微软数据库系统 SQL Server[112]资源管理器中, 资源池表示数据库引擎实例的部分物理资源. 通过资源管理器, 可以指定针对传入应用程序请求可在资源池内使用的 CPU、物理 IO 和内存的使用量的限制. 每个资源池均可包含一个或多个工作负荷组. 在某个会话启动时, 资源管理器会将此会话分配给一个特定的工作负荷组, 并且此会话必须使用分配给该工作负荷组的资源运行. 资源池表示服务器的物理资源, 它有两个部分. 一部分不与其他资源池重叠, 这使得资源预留最少. 另一部分与其他资源池共享, 支持最大可能的资源消耗.

表 5 常见数据库多租户资源管理方案

| 数据库 | 管理方案 | 特点 |
|---|---|---|
| Greenplum[108] | 采用资源组的方式进行限制 | 资源管理粒度小, 资源组内共享性好 |
| OceanBase[109] | 以资源单元的形式抽象化资源分配给资源池 | CPU和内存负载更加均衡, 便于调度 |
| openGauss[110] | 资源池绑定用户限制资源 | 实现机制简易, 针对多租户环境提出了专属的资源规划调度算法 |
| Oracle[111] | 根据负载属性进行分组, 不同组之间分配不同资源 | 开销小, 调度效率高, 但是无法考虑租户有多种负载的情况 |
| SQL Server[112] | 根据负载属性进行分组, 分配给不同的资源池, 各资源池之间互有交互 | 资源预留小, 资源池的大小设置非常困难 |

### 3.5 研究进展小结

云多租数据库资源规划调度的研究进展如表 6 所示. 综上可知, 现有针对数据库资源智能规划调度技术的研究才刚刚起步, 尚不成熟. 尤其是云原生数据库资源智能规划调度技术的研究尚少, 自适应资源规划调度、资源预估、资源弹性伸缩等方面的难题均未得到有效解决. 资源规划调度工具基本只能实现分配功能, 不能实现资源的规划调度. 相关技术尚存在如下问题: (1) 大多数模型或算法都是只针对关系数据库中某种特定类型的数据, 难以应对多租户多种类的需求变化; (2) 只是针对资源规划调度的部分特定问题 (如资源预估、弹性伸缩), 对其他问题没有涉及, 更没有提供系统的解决方案.

表 6 研究进展小结

| 研究方向 | 代表性工作 | 进展情况总结 |
|---|---|---|
| 资源规划调度 | 大数据集群[9−20]; 数据库[22−30] | 进展: 集中在单模数据库实例与租户之间的资源调度<br>不足: 尚未发现针对多模数据库的相关调度方法 |
| 资源预估 | 预测查询执行时间算法[31−48]<br>基于负载的预测算法[49−56] | 进展: 多是针对某类资源的预估预测<br>不足: 尚未发现针对租户级别的预测, 如租户资源画像 |
| 资源弹性伸缩 | 阈值规则[60−69]; 强化学习[70−81]; 排队论[82−88]<br>控制论[89−93]; 时间序列[94−100] | 进展: 主要考虑资源的横向扩展<br>不足: 对资源纵向扩展基本鲜有研究, 且大多没有在实际生产环境中验证 |

表 6 研究进展小结 (续)

| 研究方向 | 代表性工作 | 进展情况总结 |
| --- | --- | --- |
| 资源规划调度工具 | 基于作业的数据库数资源规划调度工具[101−105]<br>基于租户的数据库资源规划调度方案[108−112] | 进展: 作业可以实现资源的调度, 分配至对应的节点或任务; 当前原生的数据库系统进展大多集中在资源隔离阶段;<br>不足: 鲜有针对资源池弹性伸缩的研究, 仅有调度功能或仅有隔离功能 |

## 4 研究展望

### 4.1 面临的新挑战

合理有效的资源规划调度技术对数据库性能和用户体验的提升有着至关重要的作用. 尽管在资源规划调度方面已经有了大量研究, 但随着云原生及多模态数据库技术的兴起, 云多租数据库资源规划调度面临诸多新的需求, 具体如下.

(1) 云原生数据库要解决的核心问题是如何基于存算分离、资源池化的环境在提高资源利用率的同时进行高效地计算. 不同于先前的云多租数据库, 云原生数据库主要针对的是资源池化后的环境, 其主要特征就是资源的网络化. 这种特征导致云原生数据库在资源规划调度时需要考虑网络化引起的额外成本, 例如数据或者计算迁移的代价. 另外, 如何在资源池化的环境下进行细粒度的资源共享也是云原生数据库面临的一个难题.

(2) 智能制造、智慧城市、智能交通等领域的大数据通常具有 4V、强关联、高通量、多模态特征. 如何针对这些大数据进行有效地融合管理就成为一个难题. 多模一体化的方案是研制用于管理具有 4V、强关联、高通量、多模态特征数据的大数据管理系统的一种有效途径. 多模云多租数据库在智能制造、智能安防、智慧城市、智慧交通等领域具有广阔的应用前景, 已经成为数据库技术发展的一个新方向[7]. 多模云多租数据库中多模融合的特性导致其资源消费、竞争模式不同于云多租数据库以及单一模型的数据库系统. 其内部多个租户、多类引擎、多种任务会竞争资源. 异构性、动态性及竞争性等方面的问题导致多模大数据管理系统的资源规划调度极具挑战性, 目前鲜有相关研究成果.

### 4.2 未来研究趋势

云数据库服务 (DaaS) 在提供免部署、高性能、高可靠和扩展灵活等特性的同时, 也对各类数据库技术有了更高的要求. 在新的发展背景下, 云多租数据库尚面临如下新的问题亟待解决.

(1) 资源规划调度

通过第 3.1 节中的分析可知: 现有的多数算法都只考虑负载实例或任务间的资源调度, 并没有考虑多租户之间的资源竞争. 目前最新的研究也主要针对云多租数据库系统的虚拟机等的资源规划调度, 并没有考虑 CPU、内存等更细粒度的资源管控. 上述方法主要针对单模数据库系统, 云平台、云数据系统中多租户、多模融合特性所导致的问题并没有得到解决. 由于云多租数据库系统中资源、租户、负载及数据的动态性, 本文认为云多租数据库系统资源规划调度需要解决如下问题.

1) 云多租数据库提供商需要为租户提供最低的资源保证. 如何在不进行静态保留资源的情况下针对网络化的资源池、多类引擎、多种负载进行细粒度的、自适应的资源规划调度是未来云多租数据库系统需要重点考虑的问题.

2) 云多租数据库服务提供商希望服务能够获取更高的收益. 在资源有限的情况下, 资源超售就变得很常见. 如何在资源超售的情况下维护租户的利益 (即满足 SLA 需求)、兼顾收入与租户公平性就变得十分重要.

(2) 资源预估

在云多租数据库系统中, 负载的变化通常是动态变化的, 负载变化导致不同租户或同一租户的资源需求也在动态变化. 在这种资源不确定性强的背景下, 合理准确地预估负载的时变特征、负载所需的资源量、系统未来的

资源使用状态, 得出负载与租户的资源画像对于云多租数据库管理系统进行精细的资源调度和动态分配具有重要意义. 因此云多租数据库系统应该具备资源预估能力, 避免传统响应式资源管理所带来资源调整的滞后问题, 提高资源利用率, 降低运营成本. 但是在第 3.2 节所提到的算法中, 无论是针对查询的预测还是针对负载的预测, 都只考虑了预测准确率这一个标准, 如何使用预测结果鲜有涉及; 另外, 也没有工作针对多租数据库环境来开发预测算法 (例如考虑 SLA, QoS 等指标).

因此本文认为在资源预测方面, 对于租户级别的预估来说, 需要根据租户的使用习惯得出精确的租户资源画像; 对于负载级别的预估来说, 资源预估模块需要在微观和宏观两个方面进行预测. 在微观方面, 需要有针对 SQL 语句级别的预测方法, 确保每一个租户的性能表现; 在宏观方面, 则要根据负载特征预测未来某一时间段的资源使用状态, 方便资源弹性伸缩模块调整资源池的大小和时机. 另外, 在资源池化的环境下, 资源的分布对性能也有很大的影响, 所以需要同时预估资源的布局及数量.

(3) 资源弹性伸缩

第 3.3 节总结了 5 种当前资源弹性伸缩的主流方案, 但是上述的方案都分别适用于一些特定的场景, 而且分别存在着适应性差, 迁移性差, 稳定性差等诸多问题. 云多租数据库系统中租户、负载、系统都是动态变化的, 所以资源需求也是动态变化的. 根据负载变化、系统变化、用户需求变化自动调整资源供给是云多租数据库系统资源规划调度的基本能力. 对于大多数租户而言, 自行预估工作负载的资源需求不仅复杂而且困难. 此外, 云多租数据库中工作负载和资源需求时刻都可能发生显著变化. 如何准确针对租户工作负载动态性所带来的挑战自适应地选择资源伸缩的时机及方案是云多租数据库资源规划调度亟待解决的问题.

## 5 结束语

面对海量异构数据和云场景下多样的用户和应用需求, 当前云多租数据库面临诸多方面的挑战. 资源规划调度技术是能否充分发挥云多租数据库效能、提升平台收益及用户满意度的关键. 本文针对云多租数据库的资源规划调度机制进行了综述研究, 分析了其所面临的挑战, 系统地梳理了相关研究工作, 展望了未来研究方向并提炼出了其中的关键科学问题. 期望本文能为研究者开展相关研究提供借鉴.

**References:**


[1] Zhang X, Tune E, Hagmann R, Jnagal R, Gokhale V, Wilkes J. CPI$^2$: CPU performance isolation for shared compute clusters. In: Proc. of the 8th ACM European Conf. on Computer Systems. Prague: ACM, 2013. 379–391. [doi: 10.1145/2465351.2465388]

[2] Lu JH, Holubová I. Multi-model databases: A new journey to handle the variety of data. ACM Computing Surveys, 2020, 52(3): 55. [doi: 10.1145/3323214]

[3] Wang YZ, Yu JQ, Yu ZB. Resource scheduling techniques in cloud from a view of coordination: A holistic survey. Frontiers of Information Technology & Electronic Engineering, 2023, 24(1): 1–40. [doi: 10.1631/FITEE.2100298]

[4] Li YF, Zhou MQ, Hu HL. Survey of resource uniform management and scheduling in cluster. Journal of East China Normal University (Natural Science), 2014, (5): 17–30 (in Chinese with English abstract). [doi: 10.3969/j.issn.1000-5641.2014.05.002]

[5] Dong HW, Zhang C, Li GL, Feng JH. Survey on cloud-native databases. Ruan Jian Xue Bao/Journal of Software, 2024, 35(2): 899–926 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6952.htm [doi: 10.13328/j.cnki.jos.006952]

[6] Li GL, Zhou XH, Sun J, Yu X, Yuan HT, Liu JB, Han Y. A survey of machine learning based database techniques. Chinese Journal of Computers, 2020, 43(11): 2019–2049 (in Chinese with English abstract). [doi: 10.11897/SP.J.1016.2020.02019]

[7] Pavlo A, Butrovich M, Ma L, Menon P, Lim WS, Van Aken D, Zhang W. Make your database system dream of electric sheep: Towards self-driving operation. Proc. of the VLDB Endowment, 2021, 14(12): 3211–3221. [doi: 10.14778/3476311.3476411]

[8] Narasayya V, Menache I, Singh M, Li F, Syamala M, Chaudhuri S. Sharing buffer pool memory in multi-tenant relational database-as-a-service. Proceedings of the VLDB Endowment, 2015, 8(7): 726–737. [doi: 10.14778/2752939.2752942]

[9] CapacityScheduler Guide. 2024. http://hadoop.apache.org/docs/r1.2.1/capacity_scheduler.html

[10] Fair Scheduler. 2024. http://hadoop.apache.org/docs/r1.2.1/fair_scheduler.html

[11] 2024. https://en.wikipedia.org/wiki/Max-min_fairness

[12] Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters. Communications of the ACM, 2008, 51(1): 107–113.

[doi: 10.1145/1327452.1327492]

[13] Zaharia M, Borthakur D, Sen Sarma J, Elmeleegy K, Shenker S, Stoica I. Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling. In: Proc. of the 5th European Conf. on Computer Systems. Paris: ACM, 2010. 265–278. [doi: 10.1145/1755913.1755940]

[14] Isard M, Prabhakaran V, Currey J, Wieder U, Talwar K, Goldberg A. Quincy: Fair scheduling for distributed computing clusters. In: Proc. of the 22nd ACM SIGOPS Symp. on Operating Systems Principles. Big Sky: ACM, 2009. 261–276. [doi: 10.1145/1629575.1629601]

[15] Isard M, Budiu M, Yu Y, Birrell A, Fetterly D. Dryad: Distributed data-parallel programs from sequential building blocks. ACM SIGOPS Operating Systems Review, 2007, 41(3): 59–72. [doi: 10.1145/1272998.1273005]

[16] Ghodsi A, Zaharia M, Shenker S, Stoica I. Choosy: Max-min fair sharing for datacenter jobs with constraints. In: Proc. of the 8th ACM European Conf. on Computer Systems. Prague: ACM, 2013. 365–378. [doi: 10.1145/2465351.2465387]

[17] Ghodsi A, Zaharia M, Hindman B, Konwinski A, Shenker S, Stoica I. Dominant resource fairness: Fair allocation of multiple resource types. In: Proc. of the 8th USENIX Conf. on Networked Systems Design and Implementation. Boston: USENIX Association, 2011. 323–336.

[18] Li JX, König AC, Narasayya V, Chaudhuri S. Robust estimation of resource consumption for SQL queries using statistical techniques. Proc. of the VLDB Endowment, 2012, 5(11): 1555–1566. [doi: 10.14778/2350229.2350269]

[19] Peng YH, Bao YX, Chen YR, Wu C, Guo CX. Optimus: An efficient dynamic resource scheduler for deep learning clusters. In: Proc. of the 13th EuroSys Conf. Porto: ACM, 2018. 3. [doi: 10.1145/3190508.3190517]

[20] Mondal SS, Sheoran N, Mitra S. Scheduling of time-varying workloads using reinforcement learning. In: Proc. of the 35th AAAI Conf. on Artificial Intelligence. Virtually: AAAI, 2021. 9000–9008. [doi: 10.1609/aaai.v35i10.17088]

[21] Sun J, Li GL. An end-to-end learning-based cost estimator. Proc. of the VLDB Endowment, 2019, 13(3): 307–319. [doi: 10.14778/3368289.3368296]

[22] Das S, Narasayya VR, Li F, Syamala M. CPU sharing techniques for performance isolation in multi-tenant relational database-as-a-service. Proc. of the VLDB Endowment, 2013, 7(1): 37–48. [doi: 10.14778/2732219.2732223]

[23] Tan J, Zhang TY, Li FF, Chen J, Zheng QX, Zhang P, Qiao HL, Shi Y, Cao W, Zhang R. iBTune: Individualized buffer tuning for large-scale cloud databases. Proc. of the VLDB Endowment, 2019, 12(10): 1221–1234. [doi: 10.14778/3339490.3339503]

[24] Basiuk T, Gruenwald L, D'Orazio L, Leal E. A persistent memory-aware buffer pool manager simulator for multi-tenant cloud databases. In: Proc. of the 36th Int'l Conf. on Data Engineering Workshops (ICDEW). Dallas: IEEE, 2020. 121–126. [doi: 10.1109/ICDEW49219.2020.00011]

[25] Zhao YL, Calheiros RN, Gange G, Bailey J, Sinnott RO. SLA-based profit optimization resource scheduling for big data analytics-as-a-service platforms in cloud computing environments. IEEE Trans. on Cloud Computing, 2021, 9(3): 1236–1253. [doi: 10.1109/TCC.2018.2889956]

[26] Wang YC, He Q, Zhang XY, Ye DY, Yang Y. Efficient QoS-aware service recommendation for multi-tenant service-based systems in cloud. IEEE Trans. on Services Computing, 2020, 13(6): 1045–1058. [doi: 10.1109/TSC.2017.2761346]

[27] Zhang C, Li GL, Feng JH, Zhang JT. Survey of key techniques of HTAP databases. Ruan Jian Xue Bao/Journal of Software, 2023, 34(2): 761–785 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6713.htm [doi: 10.13328/j.cnki.jos.006713]

[28] Raza A, Chrysogelos P, Anadiotis AC, Ailamaki A. Adaptive HTAP through elastic resource scheduling. In: Proc. of the 2020 ACM SIGMOD Int'l Conf. on Management of Data. Portland: ACM, 2020. 2043–2054. [doi: 10.1145/3318464.3389783]

[29] Korkmaz M, Karsten M, Salem K, Salihoglu S. Workload-aware CPU performance scaling for transactional database systems. In: Proc. of the 2018 Int'l Conf. on Management of Data. Houston: ACM, 2018. 291–306. [doi: 10.1145/3183713.3196901]

[30] Zhang XY, Wu H, Chang Z, Jin SW, Tan J, Li FF, Zhang TY, Cui B. ResTune: Resource oriented tuning boosted by meta-learning for cloud databases. In: Proc. of the 2021 Int'l Conf. on Management of Data. Virtual Event: ACM, 2021. 2102–2114. [doi: 10.1145/3448016.3457291]

[31] Xu HY, Liu HL, Yang CY, Wang S, Li ZH. MMOS: Memory resource sharing methods to support overselling in multi-tenant databases. Computer Science, 2024, 51(2): 27–35 (in Chinese with English abstract). [doi: 10.11896/jsjkx.231000141]

[32] Xu HY. Research and implementation of key technologies in multi-tenant database resource management [MS. Thesis]. Xi'an: Northwestern Polytechnical University, 2024 (in Chinese with English abstract).

[33] Wang S. Research on AI based multi tenant resource allocation and scheduling mechanism [MS. Thesis]. Xi'an: Northwestern Polytechnical University, 2024 (in Chinese with English abstract).

[34] Ahmad M, Duan SY, Aboulnaga A, Babu S. Predicting completion times of batch query workloads using interaction-aware models and simulation. In: Proc. of the 14th Int'l Conf. on Extending Database Technology. Uppsala: ACM, 2011. 449–460. [doi: 10.1145/1951365.1951419]

[35] Akdere M, Çetintemel U, Riondato M, Upfal E, Zdonik SB. Learning-based query performance modeling and prediction. In: Proc. of the 28th Int'l Conf. on Data Engineering. Arlington: IEEE, 2012. 390–401. [doi: 10.1109/ICDE.2012.64]

[36] Duggan J, Cetintemel U, Papaemmanouil O, Upfal E. Performance prediction for concurrent database workloads. In: Proc. of the 2011 ACM SIGMOD Int'l Conf. on Management of Data. Athens: ACM, 2011. 337–348. [doi: 10.1145/1989323.1989359]

[37] Ganapathi A, Kuno H, Dayal U, Wiener JL, Fox A, Jordan M, Patterson D. Predicting multiple metrics for queries: Better decisions enabled by machine learning. In: Proc. of the 25th Int'l Conf. on Data Engineering. Shanghai: IEEE, 2009. 592–603. [doi: 10.1109/ICDE.2009.130]

[38] Wu WT, Chi Y, Zhu SH, Tatemura J, Hacigümüs H, Naughton JF. Predicting query execution time: Are optimizer cost models really unusable? In: Proc. of the 29th Int'l Conf. on Data Engineering (ICDE). Brisbane: IEEE, 2013. 1081–1092. [doi: 10.1109/ICDE.2013.6544899]

[39] Wu WT, Chi Y, Hacigümüs H, Naughton JF. Towards predicting query execution time for concurrent and dynamic database workloads. Proc. of the VLDB Endowment, 2013, 6(10): 925–936. [doi: 10.14778/2536206.2536219]

[40] Singhal R, Nambiar M. Predicting SQL query execution time for large data volume. In: Proc. of the 20th Int'l Database Engineering & Applications Symp. Montreal: ACM, 2016. 378–385. [doi: 10.1145/2938503.2938552]

[41] Popescu AD, Balmin A, Ercegovac V, Ailamaki A. PREDIcT: Towards predicting the runtime of large scale iterative analytics. Proc. of the VLDB Endowment, 2013, 6(14): 1678–1689. [doi: 10.14778/2556549.2556553]

[42] Duggan J, Papaemmanouil O, Cetintemel U, Upfal E. Contender: A resource modeling approach for concurrent query performance prediction. In: Proc. of the 17th Int'l Conf. on Extending Database Technology. Athens: OpenProceedings.org, 2014. 109–120. [doi: 10.5441/002/edbt.2014.11]

[43] Ahmad M, Duan S, Aboulnaga A, Babu S. Interaction-aware prediction of business intelligence workload completion times. In: Proc. of the 26th Int'l Conf. on Data Engineering (ICDE 2010). Long Beach: IEEE, 2010. 413–416. [doi: 10.1109/ICDE.2010.5447834]

[44] Popescu AD, Ercegovac V, Balmin A, Branco M, Ailamaki A. Same queries, different data: Can we predict runtime performance? In: Proc. of the 28th Int'l Conf. on Data Engineering Workshops. Arlington: IEEE, 2012. 275–280. [doi: 10.1109/ICDEW.2012.66]

[45] König AC, Ding BL, Chaudhuri S, Narasayya V. A statistical approach towards robust progress estimation. Proc. of the VLDB Endowment, 2011, 5(4): 382–393. [doi: 10.14778/2095686.2095696]

[46] Kipf A, Vorona D, Müller J, Kipf T, Radke B, Leis V, Boncz P, Neumann T, Kemper A. Estimating cardinalities with deep sketches. In: Proc. of the 2019 Int'l Conf. on Management of Data. Amsterdam: ACM, 2019. 1937–1940. [doi: 10.1145/3299869.3320218]

[47] Chen L, Huang HQ, Chen DH. Join cardinality estimation by combining operator-level deep neural networks. Information Sciences, 2021, 546: 1047–1062. [doi: 10.1016/j.ins.2020.09.065]

[48] Park Y, Zhong SC, Mozafari B. QuickSel: Quick selectivity learning with mixture models. In: Proc. of the 2020 ACM SIGMOD Int'l Conf. on Management of Data. Portland: ACM, 2020. 1017–1033. [doi: 10.1145/3318464.3389727]

[49] Ma L, Van Aken D, Hefny A, Mezerhane G, Pavlo A, Gordon GJ. Query-based workload forecasting for self-driving database management systems. In: Proc. of the 2018 Int'l Conf. on Management of Data. Houston: ACM, 2018. 631–645. [doi: 10.1145/3183713.3196908]

[50] Zhang QC, Yang LT, Yan Z, Chen ZK, Li P. An efficient deep learning model to predict cloud workload for industry informatics. IEEE Trans. on Industrial Informatics, 2018, 14(7): 3170–3178. [doi: 10.1109/TII.2018.2808910]

[51] Fang W, Lu ZH, Wu J, Cao ZY. RPPS: A novel resource prediction and provisioning scheme in cloud data center. In: Proc. of the 9th Int'l Conf. on Services Computing. Honolulu: IEEE, 2012. 609–616. [doi: 10.1109/SCC.2012.47]

[52] Huang JH, Li CL, Yu J. Resource prediction based on double exponential smoothing in cloud computing. In: Proc. of the 2nd Int'l Conf. on Consumer Electronics, Communications and Networks (CECNet). Yichang: IEEE, 2012. 2056–2060. [doi: 10.1109/CECNet.2012.6201461]

[53] Mozafari B, Curino C, Jindal A, Madden S. Performance and resource modeling in highly-concurrent OLTP workloads. In: Proc. of the 2013 ACM SIGMOD Int'l Conf. on Management of Data. New York: ACM, 2013. 301–312. [doi: 10.1145/2463676.2467800]

[54] Yoon DY, Mozafari B, Brown DP. DBSeer: Pain-free database administration through workload intelligence. Proc. of the VLDB Endowment, 2015, 8(12): 2036–2039. [doi: 10.14778/2824032.2824130]

[55] Gu R, Chen YQ, Liu S, Dai HP, Chen GH, Zhang K, Che Y, Huang YH. Liquid: Intelligent resource estimation and network-efficient scheduling for deep learning jobs on distributed GPU clusters. IEEE Trans. on Parallel and Distributed Systems, 2022, 33(11): 2808–2820. [doi: 10.1109/TPDS.2021.3138825]

[56] Sen R, Roy A, Jindal A. Predictive price-performance optimization for serverless query processing. In: Proc. of the 26th Int'l Conf. on Extending Database Technology. Ioannina: OpenProceedings.org, 2023. 118–130. [doi: 10.48786/EDBT.2023.10]

[57] Lorido-Botran T, Miguel-Alonso J, Lozano JA. A review of auto-scaling techniques for elastic applications in cloud environments. Journal of Grid Computing, 2014, 12(4): 559–592. [doi: 10.1007/s10723-014-9314-7]

[58] Alhamazani K, Ranjan R, Mitra K, Rabhi F, Jayaraman PP, Khan SU, Guabtni A, Bhatnagar V. An overview of the commercial cloud monitoring tools: Research dimensions, design issues, and state-of-the-art. Computing, 2015, 97(4): 357–377. [doi: 10.1007/s00607-014-0398-5]

[59] Chen T, Bahsoon R, Yao X. A survey and taxonomy of self-aware and self-adaptive cloud autoscaling systems. ACM Computing Surveys, 2019, 51(3): 61. [doi: 10.1145/3190507]

[60] Han R, Guo L, Ghanem MM, Guo YK. Lightweight resource scaling for cloud applications. In: Proc. of the 12th IEEE/ACM Int'l Symp. on Cluster, Cloud and Grid Computing (CCGRID 2012). Ottawa: IEEE, 2012. 644–651. [doi: 10.1109/CCGrid.2012.52]

[61] Koperek P, Funika W. Dynamic business metrics-driven resource provisioning in cloud environments. In: Proc. of the 9th Int'l Conf. on Parallel Processing and Applied Mathematics. Torun: Springer, 2012. 171–180. [doi: 10.1007/978-3-642-31500-8_18]

[62] Hasan MZ, Magana E, Clemm A, Tucker L, Gudreddi SLD. Integrated and autonomic cloud resource scaling. In: Proc. of the 2012 IEEE Network Operations and Management Symp. Maui: IEEE, 2012. 1327–1334. [doi: 10.1109/NOMS.2012.6212070]

[63] Casalicchio E, Silvestri L. Autonomic management of cloud-based systems: The service provider perspective. In: Proc. of the 27th Int'l Symp. on Computer and Information Sciences III. London: Springer, 2013. 39–47. [doi: 10.1007/978-1-4471-4594-3_5]

[64] Chieu TC, Mohindra A, Karve AA. Scalability and performance of web applications in a compute cloud. In: Proc. of the 8th Int'l Conf. on e-Business Engineering. Beijing: IEEE, 2011. 317–323. [doi: 10.1109/ICEBE.2011.63]

[65] Ghanbari H, Simmons B, Litoiu M, Iszlai G. Exploring alternative approaches to implement an elasticity policy. In: Proc. of the 4th Int'l Conf. on Cloud Computing. Washington: IEEE, 2011. 716–723. [doi: 10.1109/CLOUD.2011.101]

[66] Simmons B, Ghanbari H, Litoiu M, Iszlai G. Managing a SaaS application in the cloud using PaaS policy sets and a strategy-tree. In: Proc. of the 7th Int'l Conf. on Network and Services Management. Paris: International Federation for Information Processing, 2011. 343–347.

[67] Maurer M, Brandic I, Sakellariou R. Enacting SLAs in clouds using rules. In: Proc. of the 17th Int'l Euro-Par Conf. on Euro-Par 2011 Parallel Processing. Bordeaux: Springer, 2011. 455–466. [doi: 10.1007/978-3-642-23400-2_42]

[68] Nguyen TT, Yeom YJ, Kim T, Park DH, Kim S. Horizontal pod autoscaling in kubernetes for elastic container orchestration. Sensors, 2020, 20(16): 4621. [doi: 10.3390/s20164621]

[69] Das S, Li F, Narasayya VR, König AC. Automated demand-driven resource scaling in relational database-as-a-service. In: Proc. of the 2016 Int'l Conf. on Management of Data. San Francisco: ACM, 2016. 1923–1934. [doi: 10.1145/2882903.2903733]

[70] Nouri SMR, Li H, Venugopal S, Guo WX, He MY, Tian WH. Autonomic decentralized elasticity based on a reinforcement learning controller for cloud applications. Future Generation Computer Systems, 2019, 94: 765–780. [doi: 10.1016/j.future.2018.11.049]

[71] Kardani-Moghaddam S, Buyya R, Ramamohanarao K. ADRL: A hybrid anomaly-aware deep reinforcement learning-based resource scaling in clouds. IEEE Trans. on Parallel and Distributed Systems, 2021, 32(3): 514–526. [doi: 10.1109/TPDS.2020.3025914]

[72] Zafeiropoulos A, Fotopoulou E, Filinis N, Papavassiliou S. Reinforcement learning-assisted autoscaling mechanisms for serverless computing platforms. Simulation Modelling Practice and Theory, 2022, 116: 102461. [doi: 10.1016/j.simpat.2021.102461]

[73] Garí Y, Monge DA, Pacini E, Mateos C, Garino CG. Reinforcement learning-based application Autoscaling in the Cloud: A survey. Engineering Applications of Artificial Intelligence, 2021, 102: 104288. [doi: 10.1016/j.engappai.2021.104288]

[74] Dutreilh X, Moreau A, Malenfant J, Rivierre N, Truck I. From data center resource allocation to control theory and back. In: Proc. of the 3rd Int'l Conf. on Cloud Computing. Miami: IEEE, 2010. 410–417. [doi: 10.1109/CLOUD.2010.55]

[75] Dutreilh X, Kirgizov S, Melekhova O, Malenfant J, Rivierre N, Truck I. Using reinforcement learning for autonomic resource allocation in clouds: Towards a fully automated workflow. In: Proc. of the 7th Int'l Conf. on Autonomic and Autonomous Systems. Venice: IARIA, 2011. 67–74.

[76] Barrett E, Howley E, Duggan J. Applying reinforcement learning towards automating resource allocation and application scalability in the cloud. Concurrency and Computation: Practice and Experience, 2013, 25(12): 1656–1674. [doi: 10.1002/cpe.2864]

[77] Bu XP, Rao JH, Xu CZ. Coordinated self-configuration of virtual machines and appliances using a model-free learning approach. IEEE Trans. on Parallel and Distributed Systems, 2013, 24(4): 681–690. [doi: 10.1109/TPDS.2012.174]

[78] Rao J, Bu XP, Xu CZ, Wang LY, Yin G. VCONF: A reinforcement learning approach to virtual machines auto-configuration. In: Proc. of the 6th Int'l Conf. on Autonomic Computing. Barcelona: ACM, 2009. 137–146. [doi: 10.1145/1555228.1555263]

[79] Rao J, Bu X, Xu CZ, Wang K. A distributed self-learning approach for elastic provisioning of virtualized cloud resources. In: Proc. of the 19th Annual Int'l Symp. on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems. Singapore: IEEE, 2011. 45–54. [doi: 10.1109/MASCOTS.2011.47]

[80] Zhu Q, Agrawal G. Resource provisioning with budget constraints for adaptive applications in cloud environments. IEEE Trans. on Services Computing, 2012, 5(4): 497–511. [doi: 10.1109/TSC.2011.61]

[81] Taft R, El-Sayed N, Serafini M, Lu Y, Aboulnaga A, Stonebraker M, Mayerhofer R, Andrade F. P-store: An elastic database system with predictive provisioning. In: Proc. of the 2018 Int'l Conf. on Management of Data. Houston: ACM, 2018. 205–219. [doi: 10.1145/

3183713.3190650]

[82] Ali-Eldin A, Kihl M, Tordsson J, Elmroth E. Efficient provisioning of bursty scientific workloads on the cloud using adaptive elasticity control. In: Proc. of the 3rd Workshop on Scientific Cloud Computing. Delft: ACM, 2012. 31–40. [doi: 10.1145/2287036.2287044]

[83] Ali-Eldin A, Tordsson J, Elmroth E. An adaptive hybrid elasticity controller for cloud infrastructures. In: Proc. of the 2012 IEEE Network Operations and Management Symp. Maui: IEEE, 2012. 204–212. [doi: 10.1109/NOMS.2012.6211900]

[84] Bacigalupo DA, Van Hemert J, Usmani A, Dillenberger DN, Wills GB, Jarvis SA. Resource management of enterprise cloud systems using layered queuing and historical performance models. In: Proc. of the 2010 IEEE Int'l Symp. on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW). Atlanta: IEEE, 2010. 1–8. [doi: 10.1109/IPDPSW.2010.5470782]

[85] Han R, Ghanem MM, Guo L, Guo YK, Osmond M. Enabling cost-aware and adaptive elasticity of multi-tier cloud applications. Future Generation Computer Systems, 2014, 32: 82–98. [doi: 10.5555/2748143.2748357]

[86] Urgaonkar B, Shenoy P, Chandra A, Goyal P, Wood T. Agile dynamic provisioning of multi-tier internet applications. ACM Trans. on Autonomous and Adaptive Systems, 2008, 3(1): 1. [doi: 10.1145/1342171.1342172]

[87] Villela D, Pradhan P, Rubenstein D. Provisioning servers in the application tier for e-commerce systems. ACM Trans. on Internet Technology, 2007, 7(1): 57–66. [doi: 10.1145/1189740.1189747]

[88] Zhang Q, Cherkasova L, Smirni E. A regression-based analytic model for dynamic resource provisioning of multi-tier applications. In: Proc. of the 4th Int'l Conf. on Autonomic Computing (ICAC'07). Jacksonville: IEEE, 2007. 27. [doi: 10.1109/ICAC.2007.1]

[89] Bodík P, Griffith R, Sutton C, Fox A, Jordan M, Patterson D. Statistical machine learning makes automatic control practical for internet datacenters. In: Proc. of the 2009 Conf. on Hot Topics in Cloud Computing. San Diego: USENIX Association, 2009. 12.

[90] Lama P, Zhou X. Autonomic provisioning with self-adaptive neural fuzzy control for percentile-based delay guarantee. ACM Trans. on Autonomous and Adaptive Systems, 2013, 8(2): 9. [doi: 10.1145/2491465.2491468]

[91] Lim HC, Babu S, Chase JS. Automated control for elastic storage. In: Proc. of the 7th Int'l Conf. on Autonomic Computing. Washington: ACM, 2010. 1–10. [doi: 10.1145/1809049.1809051]

[92] Sellami W, Hadj Kacem H, Hadj Kacem A. Dynamic provisioning of service composition in a multi-tenant SaaS environment. Journal of Network and Systems Management, 2020, 28(2): 367–397. [doi: 10.1007/s10922-019-09510-2]

[93] Caron E, Desprez F, Muresan A. Pattern matching based forecast of non-periodic repetitive behavior for cloud clients. Journal of Grid Computing, 2011, 9(1): 49–64. [doi: 10.1007/s10723-010-9178-4]

[94] Chandra A, Gong W, Shenoy P. Dynamic resource allocation for shared data centers using online measurements. In: Proc. of the 11th Int'l Workshop on Quality of Service. Berkeley: Springer, 2003. 381–398. [doi: 10.1007/3-540-44884-5_21]

[95] Dutta S, Gera S, Verma A, Viswanathan B. SmartScale: Automatic application scaling in enterprise clouds. In: Proc. of the 5th Int'l Conf. on Cloud Computing. Honolulu: IEEE, 2012. 221–228. [doi: 10.1109/CLOUD.2012.12]

[96] Gong ZH, Gu XH, Wilkes J. PRESS: Predictive elastic resource scaling for cloud systems. In: Proc. of the 2010 Int'l Conf. on Network and Service Management. Niagara Falls: IEEE, 2010. 9–16. [doi: 10.1109/CNSM.2010.5691343]

[97] Islam S, Keung J, Lee K, Liu AN. Empirical prediction models for adaptive resource provisioning in the cloud. Future Generation Computer Systems, 2012, 28(1): 155–162. [doi: 10.1016/j.future.2011.05.027]

[98] Mi HB, Wang HM, Yin G, Zhou YF, Shi DX, Yuan L. Online self-reconfiguration with performance guarantee for energy-efficient large-scale cloud computing data centers. In: Proc. of the 2010 IEEE Int'l Conf. on Services Computing. Miami: IEEE, 2010. 514–521. [doi: 10.1109/SCC.2010.69]

[99] Roy N, Dubey A, Gokhale A. Efficient autoscaling in the cloud using predictive models for workload forecasting. In: Proc. of the 4th Int'l Conf. on Cloud Computing. Washington: IEEE, 2011. 500–507. [doi: 10.1109/CLOUD.2011.42]

[100] Shen ZM, Subbiah S, Gu XH, Wilkes J. CloudScale: Elastic resource scaling for multi-tenant cloud systems. In: Proc. of the 2nd ACM Symp. on Cloud Computing. Cascais: ACM, 2011. 5. [doi: 10.1145/2038916.2038921]

[101] Verma A, Pedrosa L, Korupolu M, Oppenheimer D, Tune E, Wilkes J. Large-scale cluster management at Google with Borg. In: Proc. of the 10th European Conf. on Computer Systems. Bordeaux: ACM, 2015. 18. [doi: 10.1145/2741948.2741964]

[102] Ousterhout K, Wendell P, Zaharia M, Stoica I. Sparrow: Distributed, low latency scheduling. In: Proc. of the 24th ACM Symp. on Operating Systems Principles. Farminton: ACM, 2013. 69–84. [doi: 10.1145/2517349.2522716]

[103] Vavilapalli VK, Murthy AC, Douglas C, Agarwal S, Konar M, Evans R, Graves T, Lowe J, Shah H, Seth S, Saha B, Curino C, O'Malley O, Radia S, Reed B, Baldeschwieler E. Apache hadoop YARN: Yet another resource negotiator. In: Proc. of the 4th Annual Symp. on Cloud Computing. Santa Clara: ACM, 2013. 5. [doi: 10.1145/2523616.2523633]

[104] Hindman B, Konwinski A, Zaharia M, Ghodsi A, Joseph AD, Katz R, Shenker S, Stoica I. Mesos: A platform for fine-grained resource sharing in the data center. In: Proc. of the 8th USENIX Conf. on Networked Systems Design and Implementation. Boston: USENIX Association, 2011. 295–308.

[105] Schwarzkopf M, Konwinski A, Abd-El-Malek M, Wilkes J. Omega: Flexible, scalable schedulers for large compute clusters. In: Proc. of

the 8th ACM European Conf. on Computer Systems. Prague: ACM, 2013. 351–364. [doi: 10.1145/2465351.2465386]
[106] Delgado P, Dinu F, Kermarrec AM, Zwaenepoel W. Hawk: Hybrid datacenter scheduling. In: Proc. of the 2015 USENIX Conf. on Usenix Annual Technical Conf. Santa Clara: USENIX Association, 2015. 499–510.
[107] Karanasos K, Rao S, Curino C, Douglas C, Chaliparambil K, Fumarola GM, Heddaya S, Ramakrishnan R, Sakalanaga S. Mercury: Hybrid centralized and distributed scheduling in large shared clusters. In: Proc. of the 2015 USENIX Annual Technical Conf. Santa Clara: USENIX Association, 2015. 485–497.
[108] Greenplum Database Resource Groups. 2022. https://greenplum.org/greenplum-database-resource-groups/
[109] https://www.oceanbase.com
[110] https://opengauss.org/zh/
[111] https://docs.oracle.com/en/database/oracle/oracle-database/19/admin/managing-resources-with-oracle-database-resource-manager.html
[112] https://learn.microsoft.com/zh-cn/sql/relational-databases/resource-governor/resource-governor-resource-pool?source=recommendations&-view=sql-server-ver16

## 附中文参考文献:

[4] 李永峰, 周敏奇, 胡华梁. 集群资源统一管理和调度技术综述. 华东师范大学学报 (自然科学版), 2014, (5): 17–30. [doi: 10.3969/j.issn.1000-5641.2014.05.002]
[5] 董昊文, 张超, 李国良, 冯建华. 云原生数据库综述. 软件学报, 2024, 35(2): 899–926. http://www.jos.org.cn/1000-9825/6952.htm [doi: 10.13328/j.cnki.jos.006952]
[6] 李国良, 周煊赫, 孙佶, 余翔, 袁海涛, 刘佳斌, 韩越. 基于机器学习的数据库技术综述. 计算机学报, 2020, 43(11): 2019–2049. [doi: 10.11897/SP.J.1016.2020.02019]
[27] 张超, 李国良, 冯建华, 张金涛. HTAP 数据库关键技术综述. 软件学报, 2023, 34(2): 761–785. http://www.jos.org.cn/1000-9825/6713.htm [doi: 10.13328/j.cnki.jos.006713]
[31] 徐海洋, 刘海龙, 杨超云, 王硕, 李战怀. MMOS: 支持超卖的多租户数据库内存资源共享方法. 计算机科学, 2024, 51(2): 27–35. [doi: 10.11896/jsjkx.231000141]
[32] 徐海洋. 多租户数据库资源管理关键技术研究与实现 [硕士学位论文]. 西安: 西北工业大学, 2024.
[33] 王硕. 基于 AI 的多租户资源分配与调度机制研究 [硕士学位论文]. 西安: 西北工业大学, 2024.

**刘海龙**(1980－), 男, 博士, 副教授, CCF 专业会员, 主要研究领域为大数据管理与分析技术, 数据库技术.

**徐海洋**(2000－), 男, 硕士, 主要研究领域为大数据管理与分析技术.

**王硕**(1999－), 男, 硕士, 主要研究领域为大数据管理与分析技术.

**李战怀**(1961－), 男, 博士, 教授, 博导, CCF 高级会员, 主要研究领域为大数据存储技术, 大数据管理技术.

**侯舒峰**(2002－), 男, 本科生, CCF 学生会员, 主要研究领域为大数据管理与分析技术.