# 云计算环境中面向 OLTP 应用的数据分布研究

王晓燕[1),2),3)] 陈晋川[1)] 杜小勇[1),3),4)]

[1)](中国人民大学数据工程与知识工程教育部重点实验室 北京 100872)

[2)](最高人民法院信息中心 北京 100745)

[3)](中国人民大学信息学院 北京 100872)

[4)](北京航空航天大学软件开发环境国家重点实验室 北京 100191)

**摘 要** 云计算为大型 OLTP 应用中分布式数据的高效存储和管理带来了新的机遇,大数据则对分布式数据的存储与管理提出了新的挑战,自动数据分布逐渐成为分布式系统中的研究重点和难点.该文对影响数据分布问题的三要素数据、负载和节点进行分析,将该问题抽象为数据分片、数据分配和负载执行 3 个相互关联的子问题,提出了数据分布问题的三角架构 DaWN.由于不同的系统有不同的应用需求,DaWN 架构以代价模型为枢纽,对特定应用需要达到的效能目标和资源限制进行调配,并提出了数据分布问题所面临的技术挑战.该文对 DaWN 架构中以顶点为代表的 3 个基本要素进行详细分析,着重对以边为代表的 3 条关联关系进行阐释,并据此对云环境中大规模 OLTP 应用的数据分片、数据分配和负载执行 3 个数据分布子问题的研究成果和进展进行归纳和总结.基于以上分析,该文以数据分片、数据分片和负载执行为变量,使用真值表覆盖数据分布问题中的 8 种类型,并采用三维立体坐标系的方式对相关工作的分布进行归纳总结和呈现.最后,该文从代价模型研究、测试基准研究、自动化数据分布技术研究、特定应用研究等 4 个角度,对数据分布问题的未来发展方向进行展望.

**关键词** 数据分布;三角架构;数据分片;数据分配;OLTP;大数据

**中图法分类号** TP392 **DOI 号** 10.11897/SP.J.1016.2016.00253

# Survey on OLTP Application Oriented Data Distribution in Cloud Computing

WANG Xiao-Yan[1),2),3)] CHEN Jin-Chuan[1)] DU Xiao-Yong[1),3),4)]

[1)](*Key Laboratory of Data Engineering and Knowledge Engineering, Renmin University of China, MOE, Beijing* 100872)

[2)](*Information Center, The Supreme People's Court, Beijing* 100745)

[3)](*School of Information, Renmin University of China, Beijing* 100872)

[4)](*State Key Laboratory of Software Development Environment, Beihang University, Beijing* 100191)

**Abstract** Cloud computing raises new opportunities and challenges in efficient distributed data storage and management for large scale OLTP applications. In the fields of Data Management, data distribution is one of the most famous technologies for platform scalability. With the dramatic increase of data volume, automatic data distribution has been one of the key techniques and intractable problem for distributed systems. Focusing on the problem of data distribution in cloud environment, this work first studies the three essential elements in this field, which are data, workload and node. Based on these analyzing, it summarizes their relationships with each other as data fragmentation, data allocation and workload processing, and abstracts the problem of data distribution as a triangle model called DaWN, which uses the three essential elements as the triangle's vertexes separately, and encodes their relationships as the edges. As different systems

收稿日期:2014-07-20;在线出版日期:2015-12-18.本课题得到软件开发环境国家重点实验室开放基金(SKLSDE-2012KF-09)、国家自然科学基金(61003086,61170010)资助.**王晓燕**,女,1981 年生,博士研究生,中国计算机学会(CCF)会员,主要研究方向为大数据管理与分析.E-mail:wxy@ruc.edu.cn.**陈晋川**,男,1978 年生,博士,副教授,中国计算机学会(CCF)会员,主要研究方向为不确定性数据管理、大数据管理等.**杜小勇**(通信作者),男,1963 年生,博士,教授,博士生导师,中国计算机学会(CCF)会士,主要研究领域为数据库系统、智能信息检索等.E-mail:duyong@ruc.edu.cn.

may have different requirements, DaWN utilizes cost model as the core to obtain the performance goals under certain resource limitations for any specific application, and presents the main challenges in data distribution. This work analyzes the various characters of the three key elements in DaWN, elaborates their relationships separately including data fragmentation, data allocation and workload processing, and provides a taxonomy and peroration of the latest research for large scale OLTP applications in cloud environment. Based on these analyses, this work uses data fragmentation, data allocation and workload processing as parameters, provides a truth table to cover all 8 kinds of possibilities in data distribution, and presents these with related works in a cube like three-dimensional coordinate system. Meanwhile, this work also prospects the future work in the problem of data distribution in cloud environment, including direction studies on cost models, benchmarks, automatic technologies and specific applications.

**Keywords** data distribution; triangle model; data fragmentation; data allocation; OLTP; big data

## 1 引 言

作为一种动态演进的复杂架构,云计算通过有效整合、共享和利用分布式数据环境中的软硬件资源,构建应对多种服务需求的应用平台,是当前大规模应用研究的重点.随着信息技术的普及,云环境中各类应用系统平台的数据管理量级正从 TB 级向 PB 级甚至 EB 级规模迈进.根据 IDC 的预测①,到 2020 年全球数据存储量极可能达到 35 ZB.很多大型在线事务级应用处理平台都面临着迅速膨胀的数据规模和日益沉重的事务级计算的双重压力.举例来说,在 2013 年淘宝双十一狂欢节中,淘宝网②在短短一天内便完成了约 1.88 亿笔在线交易,最繁忙时须完成的订单量达 79 万笔/min,峰值流量 1.3 万笔/s;国内火车票预定官网 12306③ 每次出票都需要计算该车次在全线每个站点上的各类车票数量,每年春节前夕会车票预定进入高峰期,以春节预定高峰期 2014 年 1 月 6 日为例,全天网络出票 400 万张,峰值时刻出票量超过 1700 张/s.大型联机事务处理(On-Line Transaction Processing,OLTP)应用的业务需求对分布式数据管理提出了新的挑战,如何有效地存储和管理业务系统中迅猛增加的海量数据是很多应用平台上亟待解决的重要问题,其中一项关键任务就是如何进行数据分布,即如何合理地将不断增加的数据分配到不同的数据节点上,保证系统的高吞吐量、高可用性和高可扩展性.

根据数据管理与分析领域的发展历史和研究趋势,我们从数据规模和数据处理复杂度两个维度上进行分解,将云计算时代的数据管理体系划分为如图 1 所示的 4 类领域.

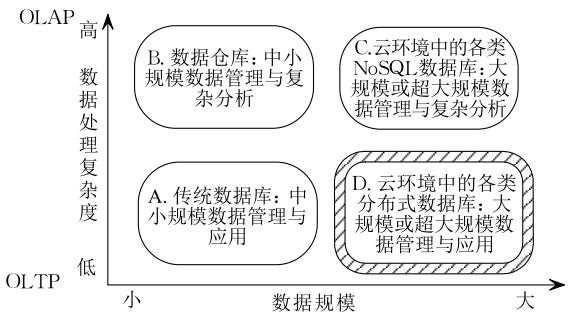


图 1 云计算时代的数据管理体系

A. 传统数据库.应用于数据规模较小、数据处理复杂程度不高的应用中,可采用传统的数据库系统进行数据管理,如图 1 中 A 域所示.

B. 数据仓库.应用于数据规模较小、数据分析处理复杂程度较高的应用中,可采用传统的数据仓库系统进行数据管理,如图 1 中 B 域所示.

C. 云环境中的各类 NoSQL 数据库.应用于大规模或超大规模数据的非关系型分布式存储与管理,主要提供复杂分析操作,可采用诸如基于 Hadoop 的 NoSQL 数据库进行数据管理,如图 1 中 C 域所示.

D. 云环境中的各类分布式数据库.应用于大规模或超大规模数据的关系型分布式存储与管理,主要提供相对简单的事务或查询操作,通常需要在毫秒甚至亚秒级完成,需要采用面向云的分布式数据管理方式,如图 1 中 D 域所示.

本文重点关注 D 类应用中的数据分布问题.随

① Digital data created in 2020 forecasted at 35 zettabytes; cloud computing will manage data growth. http://searchstorage.techtarget.com/news/1511342/Digital-data-created-in-2020-forecasted-at-35-zettabytes-cloud-computing-will-manage-data-growth

② 淘宝网. http://www.taobao.com

③ 中国铁路客户服务中心. http://12306.cn

着数据量的增长，对大数据进行管理的基本策略是将计算推向数据而不是移动大量的数据来提高计算性能[1]. 在这类应用中，单纯依靠增加高性能计算存储节点的向上扩展(Scale-Up)方式或者升级分布式数据库系统的方法已经不能适用于当前的应用需求，将大量的廉价存储和处理设备通过分布式应用系统进行整合的向外扩展(Scale-Out)方式因其高性价比和高可扩展性逐渐受到推崇. 有效管理跨越数千台服务器的大规模分布式数据不是一项简单的任务，它需要基于系统工程观点的有效技术来识别和修复在系统的实际实施和运行中可能不断出现的各种问题. 根据 CAP 原理[2-3]，分布式系统中的一致性(Consistency)、可用性(Availability)和分区容忍性(Partition Tolerance)三者不可兼得. 对于分布式系统而言，分区容忍性是业务应用的基本要求和实施策略. 如果片面追求系统并行中的高度一致性和分区容忍性(比如分布式事务机制)，很难获得良好的系统扩展性和可用性，而这正是云计算环境中大规模 OLTP 应用的重要前提. 因此，当前分布式数据存储与管理系统的设计趋势之一是转向支持符合 ACID(Atomicity 原子性、Consistency 一致性、Isolation 隔离性、Durability 持久性)约束的事务级应用.

令人遗憾的是，虽然分布式并行计算及其执行引擎的迅速发展(如 MapReduce[4]、Hadoop[5] 和 Dryad[6])和高层次的语言支持(Pig[7]、Hive[8] 和 DryadLINQ[9])极大地简化了大规模分布式数据密集型应用的发展，但这是以牺牲数据的强一致性为代价. 在每时每刻都要处理大量订单的淘宝网和 12306 等在线交易系统中，数据管理必须服从强一致性，而目前大规模分布式数据管理系统大多数仅提供非常有限的事务处理功能. 举例来说，Dynamo[10]、MongoDB[11]、CouchDB[12] 和 Cassandra[13] 等尚未提供事务级支持；Bigtable[14]、Spanner[15] 和 PNUTS[16] 等只提供单排的事务或事务更新，不能处理多表事务查询，难以推广到 OLTP 系统中，像 Azure[17]、Megastore[18]、Oracle NoSQL 数据库[19] 等这类较小规模的子集数据库也因为不完全支持传统的事务级应用而无法提供线性的向外扩展性能，而基于 H-Store[20] 的 VoltDB[21-22] 虽然支持完全的 ACID，但是它在执行并发操作时需要停止(或限制)所处理的事务来访问跨越多个分区的数据. 减少事务性支持大大简化了线性可扩展的分布式存储解决方案，然而对于不易分割事务的应用程序，保证原子性和隔离性的要求可能产生代码复杂性增加、应用程序开发速度慢、客户端事务调度性能低等弊端.

数据分布的目的是将数据分片为一系列不相交的数据片段，并按照一定的数据分配策略分散放置到各个数据节点上. 对于分布式应用而言，数据分布是决定大型 OLTP 系统性能的关键因素. 由于事务锁的存在，完成跨节点事务所需的时间开销往往远大于单节点事务. 因此，数据的并行处理能力很大程度上取决于数据分布的状况，也就是说，数据分布的优劣直接影响到大规模分布式 OLTP 系统的运行效率[23-24]. 例如，文献[25-26]的验证结果显示，在相同软硬件环境设置下，不同的数据分布策略可以带来 10 倍以上的性能差异.

对于大多数 OLTP 应用而言，数据分布实施的主要压力来自于系统运行中的动态变化：数据量迅猛增长，数据间的关联关系随着工作负载的访问频次和使用模式不断变化，物理节点的存储和处理能力也随着系统运行起伏不定. 因此，一个数据分布策略的设计目标是不仅要在系统运行的初始状态下找到一个合适的数据分布策略，而且要在数据、负载、节点等因素不断发生变化的时候进行动态调整，从而全力提升运行环境中的系统性能，并尽可能地保证全局范围内的负载均衡.

本文的第 2 节将详细阐述数据分布问题的三角架构和当前面临的键挑战；第 3 节对影响数据分布的数据分片、数据分配和负载执行 3 个关键问题的研究进展进行分类、对比和总结；第 4 节主要展望未来的研究工作；最后，在第 5 节对全文进行总结.

## 2　数据分布

数据分布(Data Distribution)是指在分布式系统中，按照一定的策略将数据分片成逻辑片段，并将这些片段分配存储到不同的物理节点上，使分布式系统对数据的并行处理能力得以充分发挥. 数据分布通常包括 3 个步骤：即数据分片、数据分配和负载执行. 其中，数据分片(Data Fragmentation)从逻辑上将全体数据按照其相互关系划分为逻辑片段，即子关系. 数据分配(Data Allocation)从物理上将划分好的逻辑片段分配存储到不同节点上. 而负载执行(Workload Processing)则是将需要执行的工作负载按照节点的实际执行能力及所存储的数据片段进行调度和运行. 数据分布就是通过数据分片、数据分布和负载执行来统一管理分布式系统中的数据和节点，并处理工作负载的综合过程，如图 2 所示.

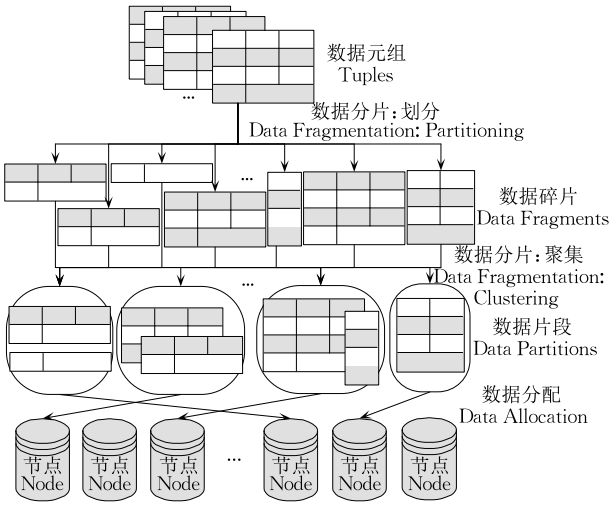


图 2 数据分布问题示意图

## 2.1 问题描述

为了进一步阐述面向大规模 OLTP 应用的数据分布问题,我们考虑以下应用场景:一个具有大规模数据量和工作负载的数据中心需要将系统中不断产生的数据和需要执行的事务配置到多个存储节点上.针对大规模 OLTP 应用,需要解决的关键问题是如何分布存储所有的数据,使得单个事务尽可能在单一存储节点上完成以避免跨节点的通信代价,并尽可能保证各节点能够相对均衡的执行不同事务请求并能充分发挥每一个存储节点的运算能力.

根据研究和分析,我们提出了称为 DaWN(表示数据 data、负载 workload 和节点 nodes)的三角架构来刻画数据分布问题,如图 3 所示.通过数据、负载和节点 3 个基本要素来回答数据分布过程中"分什么"、"怎么分"和"分到哪"这 3 个基本问题,并以代价模型为枢纽,将三要素间的相互关联关系分别抽象为数据分片、数据分配和负载执行这 3 条纽带.

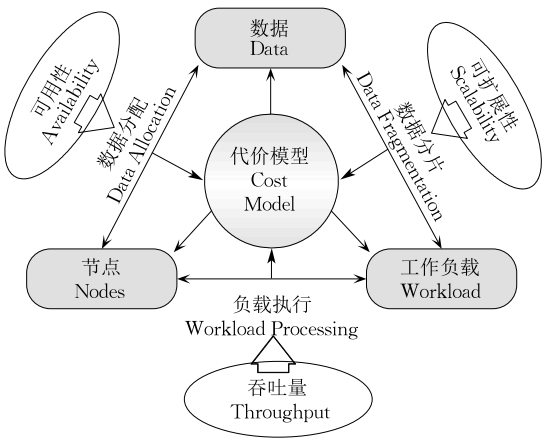


图 3 数据分布问题的三角架构

(1)数据(Data).是指系统中运行的所有数据,包括元数据、实例数据、日志数据、中间数据等.假设所有数据均以关系形式存储,且关系及关系中的属性有明确的说明,对于每一个关系,其元数据包括关系的属性集、平均元组的大小、元组的数量.对于每个属性,其元数据包括该属性是否唯一、每个不同取值的估计数目、数据类型、存储属性值所需的字节数、主外键设置等信息.

(2)负载(Workload).是指使用数据运行于节点上的各类应用操作,通常由一组事务或者查询构成,其元数据包括所使用的事务或者查询的基本模式,及其在负载执行中所占的比重.

(3)节点(Nodes).是指数据存放和负载运行的物理节点,其元数据包括节点的组织方式、节点的物理信息,如磁盘容量、CPU 频率、CPU 个数、内存大小、网络带宽等特征.

根据要素间的相互关联与制约关系,数据分布问题需考虑以下 3 个关键问题:

(1)数据分片(Data Fragmentation).又称数据划分(Data Partitioning),是指将全局概念上的数据关系逻辑地划分为若干个大小相同或者不同的局部逻辑片段(又称子关系或数据子集),分片后的每一个部分称为一个数据碎片(Data Fragment),它是数据存放的基本单位.它介于数据与负载之间,侧重于逻辑层面,主要影响系统的可扩展性(Scalability).

(2)数据分配(Data Allocation).数据分片将数据集逻辑地划分为数据片段,之后需要通过一定的数据分配策略将数据物理地存放到存储节点上.它介于数据与节点之间,侧重于物理层面,主要影响系统的可用性(Availability).

(3)负载执行(Workload Processing).它是指工作负载通过访问数据节点上的数据完成负载的执行过程.它介于节点与负载之间,侧重于实施层面,主要影响系统的吞吐量(Throughput).

此外,从静态的角度来看,数据分布需要按照从数据分片到数据分配再到负载执行的顺序来完成;从动态的角度来看,在数据生成、负载访问和节点存储的系统运行过程中,3 条纽带间存在着因三要素的变化而带来的动态制衡关系.因此,在 DaWN 架构中,我们使用代价模型来表示系统运行的基本目标和限制,基于数据、节点和工作负载的信息进行数据分片、数据分配和负载执行,处于架构的枢纽位置.

尽管我们对数据分布问题的研究主要针对云环

境中的大型 OLTP 应用，但是所提出的 DaWN 架构对于各种类型的数据分布问题研究具有普遍意义上的适用性. 在此基础上，我们定义数据分布问题如下.

**定义 1.**　数据分布(Data Distribution). 在给定的数据集 $D$、负载集 $W$ 和节点集 $N$ 的约束下，根据数据分片策略 $st_{\text{frag}}$、数据分配策略 $st_{\text{allo}}$ 及负载执行策略 $st_{\text{proc}}$，基于代价模型 $CM$ 找到合适的数据分布解决方案 $sol$.

在此，我们进一步形式化定义数据集 $D=\{F_1, F_2, \cdots, F_{|D|}\}$，其中，$F_i$ 表示按照数据分片策略 $st_{\text{frag}}$ 得到的数据片段；负载集 $W=\{T_1, T_2, \cdots, T_{|W|}\}$，其中，$T_j$ 表示负载执行中第 $j$ 种事务模式；节点集 $N=\{N_1, N_2, \cdots, N_{|N|}\}$，其中，$N_k$ 表示系统中第 $k$ 个存储节点；代价模型 $CM=CM_{\text{Cost}}-CM_{\text{Throughout}}$(详细形式化表示见 3.4 节)，其中 $CM_{\text{Cost}}$ 表示依照数据分配策略 $st_{\text{allo}}$ 得到的系统执行成本，$CM_{\text{Throughout}}$ 表示依照负载执行策略 $st_{\text{proc}}$ 得到的系统吞吐量；对于数据分布问题解空间 $SOL\langle D, W, N, st_{\text{frag}}, st_{\text{allo}}, st_{\text{proc}}\rangle$ 中的解决方案 $sol$，其最优解的目标函数可以表示为 $sol_{\text{Best}}=\min CM$.

一些早期研究[27-29]已经证明，数据分布问题及其子问题均为 NP-Hard 问题，因此，很难找到一个最优的数据分布解决方案. 同时，由于不同的应用系统有不同的实施目标，在设计不同的解决方案时，会在目标的达成中存在一定的妥协和折中. 任何系统都不是完美的，所有解决方案必然会有相应的侧重点及需要权衡的地方，如何进行取舍是通过代价模型来决定的，这也正是其作为枢纽的意义所在.

### 2.2　关键挑战

数据分布的主要目的是通过数据的合理分布，使尽可能多的数据就地存放，减少跨越逻辑分区或物理节点的数据访问，即提高访问的局部性. 如何处理 DaWN 架构中 3 条边上数据分片、数据分配和负载执行之间的相互依存和制约，是解决数据分布问题的重点和难点.

(1) 不同的数据分片策略对应不同的数据分配方案，数据分布需要考虑有关数据、场地、应用及它们之间的关联信息，与服务器能提供的支持有关，所以，在设计系统解决方案之前对用户与系统需求进行总体预测和估计，用以确定什么样的分布策略最为合适.

(2) 实际应用中，根据选定的数据分布策略，数据按照相互关系分布在不同的节点上，负载根据应用需求千变万化，节点的存储和处理能力也随行就市，需要随系统运行动态描述和维护数据分布特性及其相关信息.

(3) 用户的应用需求千变万化，节点服务器可能出现升级或宕机，且每个节点处理能力都有一定的承受限度，应用的业务负载和数据量可能面临瞬时激增等问题.

作为一个 NP-Hard 问题，需要在条件限定下得到数据分布问题的可行解，并从中选择一个相对较优的解决方案，需要遵守的策略有两个：一是最大限度地提高并行；二是尽可能减少通信. 以代价模型为核心的数据分布策略强调以实际应用需求为基础进行数据分片、数据分配和负载执行，从容应对数据、负载和节点间的制衡和挑战.

## 3　研究进展

根据 DaWN 架构，以下主要从数据分片、数据分配和负载规划这 3 个方面，对数据分布问题的相关研究和进展进行归纳和总结.

### 3.1　数据分片

数据分片[30-33]的研究始于 20 世纪 70 年代，是目前提供持久数据管理可扩展性的最优方法之一. 在逻辑上，每一个数据片段都具有一定的独立性，可以在一个或多个数据分区中进行数据存取操作. 采用合适的数据分片策略，可以大大提高数据查询速度，简化大型关系数据管理，提高系统整体性能、事务吞吐量和可扩展性[34].

文献[27,35]认为，无论采用什么方法进行数据分片，都需要参考以下 3 个准则以保证其有效性和合理性.

(1) 完整性原则(Completeness). 全局关系中的所有数据项必须通过数据分片划分到对应的数据片段中，不允许出现某个数据项属于全局关系却不属于任何一个数据片段，即对于关系 $DS$ 分片后形成的所有 $|D|$ 个数据片段 $F_1, F_2, \cdots, F_{|D|}$ 应满足 $F_1 \cup F_2 \cup \cdots \cup F_{|D|}=D$，或者说，对于每一个数据关系元组 $t$ 有 $t\in D$，$\exists F_i\in D$ 有 $t\in F_i$.

(2) 可重构原则(Reconstruction). 被划分的数据片段必须可以通过某种方式重新构成分片前的全局关系，即存在重构函数 $g$ 使得 $D=g(F_1, F_2, \cdots, F_{|D|})$，也即对于每一个数据关系元组 $t$ 有 $\forall t\in D$，$t\in g(F_1, F_2, \cdots, F_{|D|})$.

(3) 不相交原则(Disjointness). 划分全局关系

$D$ 后所得的各数据片 $F_i$ 互不重叠，即各个分片都应该是不相交的 $F_{i1} \cap F_{i2} = \varnothing (i_1 \neq i_2, 0 \leq i_1, i_2 \leq |D|)$，或者说，对于每一个数据关系元组 $t$ 有 $t \in F_{i1}$ 且 $!\exists t \in F_{i2} (i_1 \neq i_2, 0 \leq i_1, i_2 \leq |D|)$.

分片的基本逻辑形式有两种：水平分片(Horizontal Partitioning)[32]和垂直分片(Vertical Partitioning)[33]. 他们分别对具有相同性质的元组(行关系)或属性(列关系)进行划分，使具有相同划分特性的数据划分到一组，每组都构成一个数据片段. 由于数据分片的逻辑特性，不相交原则并不适用于垂直分片.

对于大规模 OLTP 应用而言，主要采用水平分片以保证数据访问的完整性和独立性，减少节点间的连接操作，从而提高执行性能，降低通信代价. 通常，水平分片方案的设计可归纳为两步：一是分片键选择策略；二是分片实施策略.

3.1.1 分片键选择策略

在实现分片之前，首先需要指定分片键作为系统索引依据，系统将根据分片键将数据划分为数据碎片并聚集到对应的数据片段中，而后这些数据片段将根据该索引的大致顺序分布存储到物理节点上. 由于数据的物理存储位置依赖于此，所以合理的选择分片键非常重要. 分片键选择策略是指如何选择合适的分片键来实施数据划分. 理想的解决方案是在无需人工干预和配置的情况下对数据进行分片，同时，能够使得分片后的系统性能尽可能达到最优.

随着分布式技术在数据库领域的应用和研究，数据分片技术有了非常多的进展[25-27,32-33,35-36]. 虽然不当的分片键选择会大大影响查询性能[26]，然而大多数研究通常假定认为适当的分片键已被选定而专注于分片实施策略[37]，对于分片键的选择问题却鲜有涉及. 在实际应用中，虽然许多商业数据库系统(如 Oracle[38]、DB2[39]、SQL Server[40]、SAP HANA[41]等)都提到分片键选择的重要性，并内置相应建议机制，但是其策略通常与系统底层深刻的结合在一起且不开源，因此很难在其他应用中直接使用. 在新近出现或者开源的大多数数据管理体系结构中，数据分片键的选择主要采用无差别的系统默认方式或者进行人工设置[27,42].

根据对已有研究和商业系统的应用分析，现有的分片键选择策略可分为以下 4 类：

(1) 主键选择法(Primary Key Selection). 这是工业界最广泛采用的分片键选择策略，其变体有诸如使用外键、关系表中的第 1 列等作为分片键.

(2) 随机选择法(Random Selection). 没有足够的数据库管理经验或没有任何约束条件的指定分片键时，可能采用这种方式. 这可能导致系统性能受到极大影响.

(3) 遍历选择法(Exhaustive Selection). 尝试遍历所有可能的分片键组合以获得分片键(组)，使得分片后的系统性能最佳. 为了获取有效信息以降低遍历空间复杂度，它需要使用样本数据集和负载模拟真实世界中的系统运行状态，然而现实应用并非一成不变，因此这种方法很难正确预测系统实时运行时的系统成本，仅适用于数据及访问模式稳定的应用类型.

(4) 分析选择法(Analytical Selection). 这种方法基于对数据和工作负载的模式分析，通常采用人工或自动优化辅助工具进行设置. 比如文献[43]提出 ASAWA 方法进行自动分片键的选择，不仅考虑了数据和工作负载模式，同时结合了设计和执行信息，从而使得经常共同出现在同一个查询中的数据元组更有可能被划分到相同的数据分片中，从而减少跨节点连接操作，提升系统性能.

随着数据量的迅速增加以及负载中各式各样的用户需求，实际应用中究竟采用哪种分片键选择策略，要根据具体业务场景决定. 这一部分的工作将集中于以自上而下的视角，根据具体的数据和负载要求自动选择.

3.1.2 分片实施策略

分片实施策略是指采用什么样的策略将数据划分到不同的分区中. 文献[26,44]的研究表明：对于分布式事务，系统吞吐量受制于节点发送和接收两阶段提交消息的速度，同时，由于系统必须等待来自多个节点的消息，其性能与分区数量是相关的. 因此，如果采用的数据分片策略能够最大限度地减少分布式事务数量和每笔事务访问的分区数量，可以减少事务执行的通信和协调成本，提高系统性能[25,35].

根据对已有应用和研究的归纳和总结，相关研究中的水平分片实施策略可以抽象为图 4 所示的层次架构.

(1) 简单分片. 这类方法以数据为中心，对表结构进行划分，是较为成熟的数据分片类型，采用的分片策略[27]主要有 Hash、Range、Round-Robin 等. 其中，只使用一种方法划分数据一次的模式是一次分片模式，采用一种或多种基本划分方法进行两次

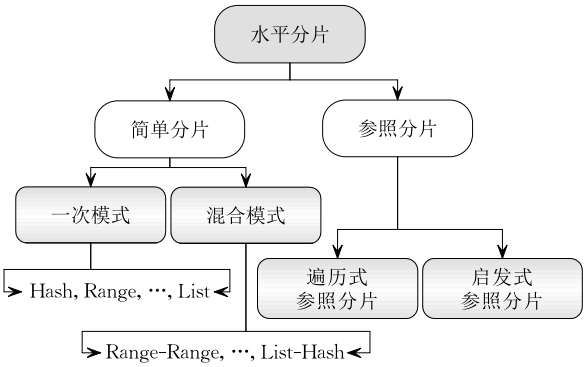


图 4　水平分片模式的层次分类

及以上划分的方式是混合分片模式，如 List-Hash、Range-Range 等. 已有的混合分片优化研究[45-47]能够在单表划分时取得较好的效果，但是随着采用不同分片方法的顺序不同，得到的结果会各有差异，而且多表环境中的优化效果不甚理想[26]. 目前，大多数商业数据库系统和工业界的应用中(如 Oracle[38]、DB2[39]、SQL Server[40,48] 等)所支持的是单表分片模式，它们通常与自身的查询优化器紧密结合[48]，根据输入数据的属性和索引策略，充分推理并集成到统一的优化框架中，完成数据分片. 简单分片策略不依赖于数据、负载和节点间的相互关系，也不依赖于数据分片、分配和负载执行的先验知识，操作简单，执行方便. 但是，由于没有考虑到数据访问模式(特别是该模式不均匀时)，可能会造成严重的节点过载和数据倾斜.

(2) 参照分片. 这类方法通常依赖于数据、负载和节点间的相互关系，或者数据分片、分配和负载执行的先验知识，根据数据与负载间的相关性及具体应用特点进行数据分片，寻求获得最优解的可能性. 与工业界应用多采用单表分片的方式不同，学术界研究大多集中于此，主要分为两类.

① 遍历式参照分片. 根据给定的数据、负载和节点特性，对所有可能的划分方案进行蛮力式遍历搜索，以求找到最合适的分片方案. 然而，如果有问题的搜索空间非常大，可能需要过度的运行时间. 为了避免检查搜索空间中的所有可行点，可采用分支定界等方法进行剪枝从而减少复杂性，也可以通过动态规划方法避免一些冗余计算. 常用的算法有枚举法、图划分法、邻域搜索算法等.

举例来说，文献[49]推出一个综合的分区调整设计方法，该方法先通过调用一个“建议”模式来评价所有的工作负载报表生成候选配置. 通过比较和评估所有的备选方案，在枚举过程中评估它们的代价模型排名顺序，直到达到停止条件的配置，并在随后的分区扩展中，进一步进行额外候选方案的比较，找到最佳解决方案.

Schism[23]是一种基于图划分的面向事务查询的数据分片方法，它将每一个元组视为一个节点，将在工作负载中被同时访问的元组节点之间用边相连，采用图划分算法 Metis① 进行数据分片，平衡分区界限，其划分结果能够保证良好的本地语义特征，同时有效地减少系统中分布式事务的数量，适合于数据和负载相对固定的 OLTP 应用. 文献[50]对 Schism 进行了改进，依据时间窗口模型聚类元组，并构建簇节点图，利用分区感知策略对图进行删减来降低算法的复杂度，提高了分区事务查询速度. 文献[51]对 Schism 进行了扩展，以 Granola 为基础，改进了划分图的边权值机制，协调事务的两阶段提交，采用基于机器学习的路由机制，完成自动数据分片，同时实现了对运行时事务的跟踪采集和基于静态程序分析的自动事务选择.

随机邻域搜索(Random Neighborhood Search，RNS)[40]通过产生中等质量的初步解决方案，根据预定义的代价模型，使用概率选择和测试等方式在附近搜索空间的解决方案中探寻更好的解决方案，是一种有效的近似方法. 文献[26]在保证事务处理本地性的前提下，对无共享数据库中数据分片可能产生的数据倾斜问题进行了研究，并采用大规模邻域搜索(Large Neighborhood Search，LNS)[52-53]进行数据分片，最小化分布式事务数量. 然而，该算法在停止指定的搜索步骤或已过固定数量的步骤后没有改善的解决方案，如果有新增数据，仍然需要重新进行计算和分配.

上述几项研究工作适用于数据、负载和节点均相对稳定的应用. 然而，这样的系统相对较少. 而且，在实际的大规模 OLTP 应用中，特别是当前的大数据环境里，基本不可能采取一次性划分且无后续调整的策略. 因此，这类方法难以得到广泛的应用. 最近，文献[54-56]提出处理增量数据的解决方案. 从 one-size-fits-all 的视角，文献[54]提出了一种单维划分算法，采用或水平或垂直划分的方式，将两种策略混合起来，完成在线的数据划分请求，可以用于 OLTP，但主要面向 OLAP 应用. 文献[55]将数据划分技术应用到一个特定的科学计算环境中，面向具

① Metis. http://glaros.dtc.umn.edu/gkhome/views/metis/index.html

有工作流特征的数据进行数据划分.文献[56]应用Table-as-a-Column理念,采取表内Range分片、表间列式聚集的方式,提出了TDPS策略,将水平划分的元组完整特性和垂直划分的元组聚集特性有效的结合起来,根据数据表的依赖关系和工作负载的均衡要求对增量数据进行动态划分,从而为各类OLTP应用提供一个合适粒度的分片方案.

②启发式参照分片.根据具体的应用场景,结合给定的数据、负载和节点的模式与特性,以具体应用需求驱动智能启发算法的使用,缩小解空间的搜索维度,以求快速找到合适的分片方案.目前,常用的启发式方法主要有遗传算法、模拟进化算法、模拟退火算法等.

遗传算法(Genetic Algorithm,GA)是模拟达尔文生物进化论的自然选择和遗传学机理的生物进化过程的计算模型[57-59].文献[58]通过将分片聚类问题形式化为旅行商问题(Traveling Salesman Problem,TSP)结构,使用优化的遗传算法进行聚类,解决数据分片问题.由于遗传算法的整体搜索策略和优化搜索方法在计算时不依赖于梯度信息或其他辅助知识,而只需要影响搜索方向的目标函数和相应的适应度函数.

模拟进化算法(Simulated Evolution Algorithm,SE)通过模拟自然进化过程来搜索最优解[60-61],它主要使用概率搜索技术,对减少或增加一个客观的代价函数的机制进行实施,并为解决这类问题找到合适的解决方案和替代方法.

模拟退火算法(Simulate Anneal Algorithm,SAA)来源于固体退火原理,即将固体充分升温再徐徐冷却,升温固体内部粒子呈无序状态且内能增大,冷却时粒子渐趋有序且在每个温度都达到平衡态,最后在常温时达到内能最小的基态.文献[62-63]将模拟退火计算与著名的Hopfield神经网络相结合,用于数据划分的组合优化问题,寻找最优的解决方案.

虽然使用启发式方法能够结合具体的应用场景,有效地降低解空间的搜索维度,然而,如果每个片段的访问模式的变化频率高,这种方法需要花费大量的时间用于不同节点间的数据通信和传输.因此,响应时间和延迟将会增加.如何能够在提高分片效果的前提下,高效地执行数据分片及分片后的工作负载是亟待解决的研究问题.

### 3.2 数据分配

对于大规模OLTP应用而言,单个节点的存储和处理能力是不能满足业务需求的,需要将系统中的数据及相关负载分散到不同的物理节点上存储、管理和执行.一个事务是否可以在单一数据分片内执行须依赖于数据分配,因此,文献[30]引入数据分片的同时也引入了数据分配的研究.数据分配(Data Allocation),又称数据放置(Data allocation),是指按照一定的方法或策略,将使用分片策略划分全局关系得到的逻辑片段合理地存放到物理数据节点上.简单的数据分配容易做到,但是要让系统高效而稳健的运行则需要按照应用目标进行优化设计.设计不当的数据分配会导致计算低效、接入成本高和网络负载重[27].在分布式系统设计中,数据分配的基本原则是:以系统全局性能优化和节点负载平衡为优化目标,尽可能地将数据放置在执行或者靠近访问它的负载所在的节点上.因此,在数据分配中需要注意做到以下两点:

(1)局部访问原则(Local Accessing).进行数据分配时,需要尽量地将可能会同时访问或处理的数据放在相同或相邻的节点上,使查询或者事务的处理尽可能地在相同或相邻的节点上完成,降低因跨节点的负载访问所产生的节点I/O、通信等代价,提高数据访问或处理的局部性,保证系统性能.文献[64]指出设计良好的应用应使得数据的本地访问程度达到90%以上.

(2)负载均衡原则(Load Balancing).每一个节点的存储和处理能力不尽相同,所存储的数据内容和处理方式也必各有千秋.在系统运行中,由于数据分配的透明性和不确定性,每个节点所需承担的局部数据处理和全局数据访问的任务和强度都有差异,而且用户需求和数据生产千变万化,需要在数据分配时,使各个节点的数据存储、访问、处理和通信的负载尽量均衡,提高系统并行处理能力.

在一般情况下,要完全做到上述准则是非常困难的,在实际的应用中需要根据具体的应用场景和用户需求选择主要的数据分配目标,结合相关约束条件构建代价模型权衡利弊,优化实际分配策略.总体上,实际采用的策略可以分为两大类:在计算过程中无须改变的分配称为静态分配,在计算执行前可预先确定;在并行计算中需要对新增数据进行分配或者对已有数据重新分配的称为动态分配或重分配.

(1)静态分配策略

早期有很多静态分配策略研究[65-68].其中,静态

Hash 方法是传统并行存储系统中方法，无需查询元数据服务器，计算从逻辑数据标识符到物理位置的映射. Round-Robin 可以根据最大并行度的定义实现最大并行，很多应用存储系统采用此机制[69]. 但是一旦有节点增删，为了保持模函数的一致性和整个系统的负载均衡，模数范围发生变化，需要根据新的范围改变存储映射或者移动数据，这种代价是很大的. 因此，在静态环境中，由于节点上的分片接入概率不变，能够为数据分配提供最佳解决方案. 然而，在动态环境中，这些概率随着时间的推移变化，原分配方案不能做出对应的调整，会影响应用系统的性能和稳定.

(2) 动态分配策略

在早期的动态数据分配研究中，文献[70-71]给出了初步的数据动态分配框架，以及进行重分配的过程. 在近期的研究中，文献[72] 定义了可扩展分布式数据库系统中的数据分配问题，提出了一个利用时间序列模型进行短期负载预测的高效算法，提前进行节点数目调整和片段重新分配，避免节点过载和性能退化. 文献[73]提出了一种改进的遗传算法作为数据分配策略，以多目标优化理论为指导，采用自适应的负载均衡策略对各个节点所要执行的工作负载进行调优，以获得更好的系统性能. 文献[74]基于一致性哈希算法的基本思想，通过引入虚拟节点提高负载均衡，提出一种高效动态数据分配策略，同时采用一种新颖的可用存储容量感知和存储容量利用率感知的方法增强云存储系统的性能. 文献[75]着眼于提高新的数据放置机制来减少因节点增删造成的数据移动，为衡量再平衡过程的性能给出了无效移动率的概念，通过使用次序选择和改进的次序选择机制的机制，对可扩展并行存储系统的数据放置进行了研究. 我们[76]提出了自动数据分片问题解决的基本架构，并对其进行了具体的实施，对各功能模块的协动关系进行探讨，采用 Nash-Pareto 优化均衡策略使得前述各机制相得益彰，协同支持自动数据分布的执行，实验结果体现了解决方案的有效性. 文献[77]将市场理念融入到云环境下负载均衡的挑战，提出一个称为 MBA 的以市场为基础的控制方法，将节点视为交易市场，通过目标市场规则来智能地决定数据的分配和迁移，从而实现云数据库节点之间数据的合理分布，平衡工作负载，并提高云数据库的整体运行性能.

此外，在科学计算及工作流领域对数据分布问题的研究尤为深入. 文献[78]提出了能够自适应存储规模变化、公平有效的数据布局算法 CCHDP，将聚类算法与一致性 hash 方法相结合，通过引入少量的虚拟设备实现按照设备权重分配数据，并在存储规模发生变化时进行自适应的调节. 文献[79]设计了基于聚类矩阵的数据放置策略，在工作流建立阶段，使用 BEA 算法[80]对按照建立的全局数据关系相关矩阵进行变换和聚类，进行高内聚低耦合的分配；在工作流执行阶段，在满足存储限制的前提下，新产生的数据集被放置在相关度最大的数据中心上. 文献[81]针对大规模数据去重查询的挑战，提出了一种自适应的散列和直方图相结合的数据分布策略，动态调整各节点之间的数据均衡，适用于数据动态变化的工作流系统. 文献[82]分别针对云计算环境数据密集型应用中的跨数据中心传输、数据依赖和全局负载均衡 3 个目标对数据布局方案进行求解、评价、调整和优化，提出一种三阶段数据布局策略，具有良好的综合性能. 然而，由于研究领域和系统特性不同，这些研究很难实施“拿来主义”直接将其应用于实践，但是可以在大规模 OLTP 应用的数据分配中作为参考和借鉴.

综合前述相关研究工作和进展，根据不同的应用需求，我们可以构建出解决数据分配问题的核心策略组件，如图 5 所示.

根据图 5，从系统架构的角度上，数据分配策略可以分为独立模式、逻辑分布模式和混合模式. 不同的模式中，模块间的耦合性不同. 从算法的角度上，数据分配策略可以分为集中式策略和非集中式策略. 从影响算法性能的因素，主要有数据加载、通讯延迟、带宽限制、机器资源限制、操作权限等. 从操作级别来看，有运算重用和复制等. 从影响数据重分布的因素，主要有网络变化、数据变化和业务流变化等. 对于数据分配的重新设置也分为静态和动态两种类型，静态方式是一次设置之后不再变化的，动态设置则根据资源控制和迁移进行相应的策略变化. 正是这些不同的组件假设展示了这些研究在核心部件的选择上带来的解决方案的多样性，极大的影响了这些数据分配策略的设计和实施. 在实际应用中，可以对核心组件进行相应更替以满足实际的系统和用户需求.

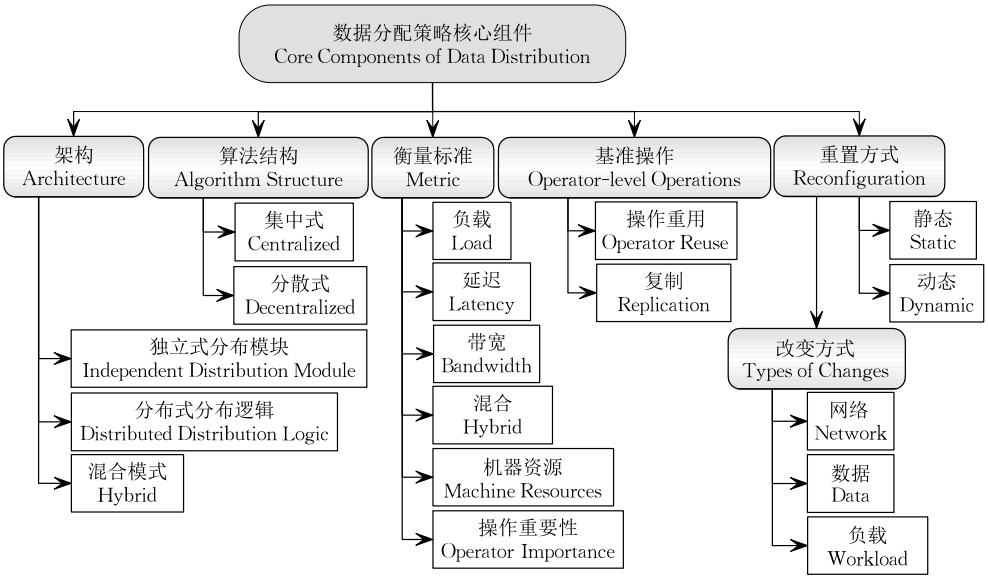


图 5 数据分配算法核心组件

### 3.3 负载执行

在云计算环境中,高效的资源管理和负载均衡是缩短事务平均响应时间和提高系统资源利用率的必然要求.负载执行要求综合数据量、工作负载量和节点存储处理能力等信息,使得每个节点的数据存储量和工作负载量能够根据其存储处理能力达到相对平衡的状态,在保证系统性能的前提下,以期达到负载均衡的运行效果.负载执行会受到许多因素的影响,如应用需求、系统架构、资源调度等.用户访问请求的不可预测性可能导致某些节点的访问压力过重,而其他节点比较空闲,在热点数据分布[83]、负载分布不均[84]的情况下,持续的并发访问压力将影响系统的整体性能.一般来说,对于动态运行的云计算系统,无论采用哪种数据分布方法,都不可避免地会出现节点负载执行不均衡的情况,这主要是 DaWN 架构里基本元素间的相互变化和制衡造成的.

(1)数据特性差异.相同任务在不同的节点可能产生不同数量的中间结果,带来计算量和存取开销的差异.

(2)负载执行差异.多任务同时运行时,由于各节点执行顺序的差异,导致每个任务在各节点的完成时间的差异.

(3)节点性能差异.由于硬件缺陷或者不稳定导致部分节点的执行性能低下,在分布式多节点环境中,不可避免地会出现个别节点的硬件发生老化或者缺陷的情况.

对负载执行的处理策略主要是通过负载预测来完成的,即根据系统的数据关系、负载运行特性、节点增容决策等影响因素,使用统计、数据挖掘、模式分析等方式,确定未来负载执行的趋势,以期达到负载均衡的目标.动态数据的重新分配必须有效地适应于访问模式的变化.采用机器学习技术对负载建模[85]是提取数据库系统信息的重要方法.文献[86]的研究工作首创通过产生内部事务执行模型,即建模事务执行过程中执行了哪些查询而不是简单地完成事务执行,来优化在分布式数据库环境中各事务的执行.已有的方法通常是基于单个查询或者一组事务来建模工作负载,从而管理资源分配或者估算其他事务未来的活动.在前者的类别中,文献[87-88]采用马尔可夫模型动态地确定何时应用程序的工作负载属性已经改变及数据库的物理设计需要更新.文献[89]通过鉴定样本数据库的工作负载判断其应用程序类型(是 OLTP 还是 OLAP 应用),并据此对调谐系统配置.文献[90]利用决策树方法对长时间运行的 OLAP 查询进行资源调度和分配.文献[91]基于当前正在执行的查询采用马尔可夫模型来估计应用程序下一步将要执行的查询,然后在有足够可用资源的前提下对下一个查询进行预取.同样,文献[92]基于当前 DBMS 正在执行的事务,采用马尔可夫模型来估计用户将要执行的下一个事务.文献[93]使用基于查询的马尔可夫模型(Markov models),但他们的模型主要以离线分析为目的,用于识别跨事务边界的用户会话和提取额外的使用模式.文献[94]通过使用 Markov 模型描述 OLTP 应

用中的存储过程，采用 4 种不同的优化设置来预测未来事务的执行以提高其可扩展性和准确性.

### 3.4 代价模型

在数据管理系统中，数据分布方案采用代价模型来估计 DBMS 将有多少资源用于执行特定的查询或事务. 根据 DaWN 架构，解决数据分布问题需要综合考虑三要素的各种特性（比如数据片段的大小、负载的执行需求、节点的存储限制以及网络通信代价等），根据应用需求构建多个数据分布方案，并通过代价模型估算和对比各方案的执行成本，得到最佳的解决策略.

作为分布式系统中的一个关键问题，数据分配有许多执行方法和实施措施. 从应用性能和用户需求的角度考虑，数据分配方案应尽可能地强调处理本地性以减少全局事务并网络通信代价；同时，必须保证数据分布方案的计算可行性，即算法须具有较小的时间和空间复杂度. 不同的优化模型采用不同的优化度量标准，这些标准通常可以归为两类.

（1）性能. 最大化整体系统性能的基本测度方法是最大限度的提高单位时间内的系统吞吐量，即单位时间内完成的事务或者查询处理的数量. 对于一个事务来说，如果能够在一个节点运算完成是最理想的状态，如果需要在多个节点运算完成，则需要增加传输成本和系统访问成本. 因此，通常这一目标的实现是通过尽量最小化全局范围内跨节点事务处理的数量.

（2）成本. 最小化整体系统运行成本的基本测度是最小化网络通信数据量和事务响应时间. 根据负载特性进行分片的数据可以通过数据分配存放到数据节点上，同时需要兼顾节点上的数据存储与工作负载均衡.

已有研究和应用系统根据各自的侧重点对代价模型的“最优”加以区别，其衡量应同时包含性能和成本这两个因素，即用最小的处理代价给用户最快最准的响应. 但由于问题过于复杂，至今还没有研究出这样的模型[27]. 根据已有研究，我们对数据分布问题中所使用的代价模型可以归为两类：一类是分析型模型，即基于启发式方法估计资源消耗[95-96]；另一类是“真实世界”模型，即利用数据库管理系统内部的查询优化器来计算和估计运行代价[35,97]. 很多文献对主内存 DBMS 的代价估算的都是面向单节点系统[98]或不考虑负载倾斜[99-100]的.

总体来讲，数据分布的设计期望是尽可能优化整体系统性能（如响应时间、吞吐量等）. 虽然理想的解决方案应该完全杜绝分布式事务的发生，但实际系统中通常只能是尽可能减少分布式事务数量. 如果还要考虑诸如数据在不断增长、工作负载的内容可能发生较大的变化等情形，上述问题将变得更为复杂. 在最快响应每一个事物、最大化每一个节点的吞吐量的同时最小化处理成本是非常复杂的参考模型. 因此，数据分布问题的解决方案的基本思想是将数据按照一定的数据划分策略分割成数据片段，并按照片段间的相互关系和工作负载的访问情况进行聚集，并分别放置到对应的数据节点上，其设计目标是使存取访问相关度高的数据分配在同一节点上，并兼顾全部节点上的数据存储和负载执行的均衡，以避免出现严重的数据倾斜.

### 3.5 分析归纳

对于大规模 OLTP 应用而言，解决数据分布问题需要从各个子问题的可行解中选择一个解决方案，并在最小化跨节点数据访问量和最大化整个应用的负载吞吐量这两个全局优化目标中进行权衡和取舍. 在国内国外已经出现的大量研究中，基本的数据分布策略通常是基于磁盘容量、访问频率或者网络流量，在最大化系统吞吐量的前提下，尽量减少分布式事务的数量，同时兼顾物理节点上的数据和负载均衡. 目前，工业界的实现偏重于忽略三要素特性的无差别简单处理，学术界的研究文献偏重于子问题的算法设计，根据 2.2 节，在当前新时期云计算环境中的大规模 OLTP 应用迫切需要对数据分布问题进行整体性和创新性的研究、设计与实现.

根据 DaWN 架构，可以按照三要素的变化特征，对不同的数据分布解决方案进行归纳和总结. 由此，我们认为，在数据分布问题中：数据的变化主要来自数据量的增加和数据更新；负载的变化主要来自负载访问模式和频次的变化；节点的变化主要来自节点的同质性和异质性设计. 因此，对于大规模分布式应用而言，重新定义数据模式、负载模式及频繁的增减更换数据节点会导致严重的系统不稳定. 根据数据、负载和节点的变化特性，按照数据分片、数据分配和负载执行的变化特征，可以得到 8 类数据分布解决方案，如表 1 所示. 其中，S 代表静态 Static 类型，F、A、P 分别代表动态变化的数据分片 Fragmentation、数据分配 Allocation 和负载执行 Processing 分量.

**表 1 解决方案分类指示表**

| 编号 | 类型 | 动态执行 Processing | 动态分配 Allocation | 动态分片 Fragmentation | 状态 |
|---|---|---|---|---|---|
| (1) | S 型 | 0 | 0 | 0 | 静态 |
| (2) | F 型 | 0 | 0 | 1 | 动态 |
| (3) | A 型 | 0 | 1 | 0 | 动态 |
| (4) | P 型 | 1 | 0 | 0 | 动态 |
| (5) | FA 型 | 0 | 1 | 1 | 动态 |
| (6) | FP 型 | 1 | 0 | 1 | 动态 |
| (7) | AP 型 | 1 | 1 | 0 | 动态 |
| (8) | FAP 型 | 1 | 1 | 1 | 动态 |

以数据变化、负载变化和节点变化分别作为 $D$、$W$、$N$ 轴的演进方向，对前述所有相关参考文献进行应用分类，如图 6 所示.

(1) S 型

静态数据分布，这类方案[23,26,30-33,58-59]主要处理数据量不增加、工作负载对数据的访问是稳定的或者具有一定的时序规则的、节点是均质的数据分布问题.

(2) F 型

动态分片型数据分布，这类方案[45-47,49-50,56-57,62]主要处理数据量增加、工作负载对数据的访问是稳定的或者具有一定的时序规则的、节点是均质的数据分布问题.

(3) A 型

动态分配型数据分布，这类方案[64-67]主要处理数据量不增加、工作负载对数据的访问是时序不稳定的或者不规则的、节点是均质的数据分布问题.

(4) P 型

动态负载调整型数据分布，这类方案[85-94,101]主要处理数据量不增加、工作负载对数据的访问是稳定的或者具有一定的时序规则的、节点是异质的数据分布问题，即节点设备在存储容量、处理速度、网络带宽等方面具有巨大差异，不能按照均质的方式来处理所有节点.

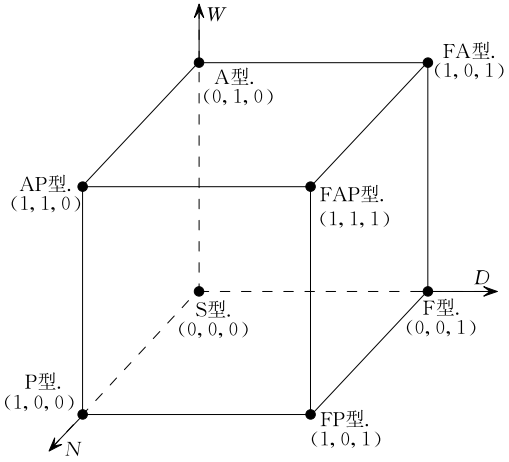


图 6 数据分布解决方案分类

(5) FA 型

节点均质型数据分布，这类方案[38-40,48,51,54-55,60-61,75,95-97]主要处理数据量增加、工作负载对数据的访问是时序不稳定的或者不规则的、节点是均质的数据分布问题.

(6) FP 型

负载稳定型数据分布，这类方案[70-71,74,76,78-79,98-100]主要处理数据量增加、工作负载对数据的访问是稳定的或者具有一定的时序规则的、节点是异质的数据分布问题.

(7) AP 型

数据稳定型数据分布，这类方案[102-106]主要处理数据量不增加、工作负载对数据的访问是时序不稳定的或者不规则的、节点是异质的数据分布问题.

(8) FAP 型

全动态型数据分布，这类方案[72-73,77,82]主要处理数据量增加、工作负载对数据的访问是时序不稳定的或者不规则的、节点是异质的数据分布问题. 这类综合性研究最难，但是由于其最符合实际的应用场景，代表着数据分布问题解决方案的发展方向. 其他 7 种类型的数据分布可以视为其某种类型的特殊应用.

## 4 未来研究方向

在面向大数据的云计算环境中进行数据分布是一个复杂的系统性问题，需要我们持续不断的努力. 在未来的研究工作中，可以在以下发展方向中进行进一步的研究.

(1) 代价模型研究. 每一个数据分布及其子问题的研究都会根据应用特性和优化目标构建相应的代价模型. 然而，由于系统目标千差万别，这些代价模型从定义到实现都很难达到互通互融的要求. 如何能够将数据分布问题所涉及的方方面面的要素统一组织起来，并在具体实施的过程中各有侧重，使得相互的研究和应用有更多的分享和借鉴是一个非常值得深思的问题.

(2) 测试基准研究. 在云计算时代，面向 OLTP 的数据分布及其各个子问题的研究仍局限于特定应用或者传统的测试基准. 然而，由于数据关系简单、事务类型较少且工作负载及数据特性稳定，如 TPCC① 的传统应用和测试基准不能反映大数据环

① http://www.tpc.org/tpcc/

境中的数据 3V 特性. 而且，应用研究各自为政，所提出的方法的普适性和适用范围难以验证和普及. 这一现状的存在迫切要求有一个反映云计算环境中 OLTP 应用特性的测试基准的诞生，实现面向数据分布问题的可对比测试，量化测试结果，实现相关研究之间的可测量性、可重复性和可对比性.

(3) 自动化数据分布技术研究. 计算的根本目的是将那些看似非常困难的问题逐步实现自动化，提高生产效率. 要显著提高数据管理系统的性能，最重要的是去除人工操作. 这是因为，硬件资源及软件运行环境非常昂贵的情况已是昨日黄花，今天最昂贵的是人员成本. 如果要在 IT 技术的演进中取胜，“一切自动”的系统(包括自我修复、自我维持、自我调整等)才是终极答案[21]. 对于数据分布问题，要实现全面的自动化解决方案需要着重解决 3 个相互关联的关键问题的挑战，即：一个全局性的关系应当如何划分，划分后的数据片段应当如何分配给通信网络中的数据节点以及如何执行数据分片和数据分配任务以实现预定的目标.

(4) 特定应用研究. 对于特定领域的数据分布问题进行优化，例如在科学计算中其优化目标是设计尽可能地使用最少时间和空间的索引，空间科学数据库的应用则要求尽可能地采用更多的优化技术在同等的约束环境中产生理想选择，但是启发式方法通常导致次优行为的产生. 根据特定应用领域进行小范围里的普适方案设计和实施将极大地提高该领域的数据分布针对性和整体系统性能.

## 5　结　论

云计算时代里大数据波谲云诡来袭，为面向大规模 OLTP 应用的数据分布问题带来了新的发展机会和挑战. 本文提出了以数据、负载和节点为要素的数据分布三角架构 DaWN，并以此为纲，归纳总结了数据分布及其 3 个子问题(即数据分片、数据分配和负载执行)的相关研究进展. 最后，提出了一些面向大规模 OLTP 应用的数据分布问题未来研究和发展的课题和方向.

## 参　考　文　献


[1] James K G, Evelson B, Karel R. In-database analytics: The heart of the predictive enterprise. Forrester Whitepaper, USA: Forrester Research, 2009

[2] Brewer E. Towards robust distributed systems//Proceedings of the 19th Annual ACM Symposium on Principles of Distributed Computing. Portland, USA, 2004: 7

[3] Brewer E. CAP twelve years later: How the “rules” have changed. Computer, 2012, 45(2): 23-29

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

[5] White T. Hadoop: The Definitive Guide. USA: Yahoo! Press, 2010

[6] Isard M, Budiu M, Yu Y, et al. Dryad: Distributed data-parallel programs from sequential building blocks//Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems. Lisbon, Portugal, 2007: 59-72

[7] Olston B, Reed U, Srivastava R, et al. Pig latin: A not-so-foreign language for data processing//Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data. Vancouver, Canada, 2008: 1099-1110

[8] Capriolo E, Wampler D, Rutherglen J. Programming Hive. USA: O'Reilly Media, 2012

[9] Yu Y, Isard M, Fetterly D, et al. DryadLINQ: A system for general-purpose distributed data-parallel computing using a high-level language//Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation. Berkeley, USA, 2008: 1-14

[10] DeCandia G, Hastorun D, Jampani M, et al. Dynamo: Amazon's highly available key-value store//Proceedings of the 21st ACM SIGOPS Symposium on Operating Systems Principles. Stevenson, USA, 2007: 205-220

[11] Plugge E, Hawkins T, Membrey P. The Definitive Guide to MongoDB: The NoSQL Database for Cloud and Desktop Computing. Berkely, USA: Apress, 2010

[12] Anderson J C, Lehnardt J, Slater N. CouchDB: The Definitive Guide. USA: O'Reilly Media, 2010

[13] Lakshman A, Malik P. Cassandra: Structured storage system on a P2P network//Proceedings of the 21st Annual Symposium on Parallelism in Algorithms and Architectures. Calgary, Canada, 2009: 47

[14] Chang F, Dean J, Ghemawat S, et al. Bigtable: A distributed storage system for structured data. ACM Transactions on Computer Systems, 2008, 26(2): 1-26

[15] Corbett J C, Dean J, Epstein M, et al. Spanner: Google's globally-distributed database//Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation. Berkeley, USA, 2012: 251-264

[16] Cooper B F, Ramakrishnan R, Srivastava U, et al. PNUTS: Yahoo!'s hosted data serving platform//Proceedings of the 34th International Conference on Very Large Data Bases. Auckland, New Zealand, 2008, 1(2): 1277-1288

[17] Campbell D, Kakivaya G, Ellis N. Extreme scale with full SQL language support in Microsoft SQL Azure//Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data. Indianapolis, USA, 2010: 1021-1024

[18] Baker J, Bond C, Corbett J, et al. Megastore: Providing scalable, highly available storage for interactive services//Proceedings of 5th Biennial Conference on Innovative Data Systems Research. Asilomar, USA, 2011: 223-234

[19] Seltzer M. Oracle NoSQL Database. Oracle White Paper. USA: Oracle, 2011

[20] Kallman R, Kimura H, Natkins J, et al. H-Store: A high-performance, distributed main memory transaction processing system//Proceedings of the 34th International Conference on Very Large Data Bases. Auckland, New Zealand, 2008, 2(1): 1496-1499

[21] Stonebraker M, Madden S R, Abadi D J, et al. The end of an architectural era (it's time for a complete rewrite)// Proceedings of the 33rd International Conference on Very Large Data Bases. Vienna, Austria, 2007: 1150-1160

[22] Jones P C, Abadi D J, Madden S R. Concurrency control for partitioned databases//Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data. Indianapolis, USA, 2010: 603-614

[23] Ke Q, Prabhakaran V, Xie Y, et al. Optimizing data partitioning for data-parallel computing//Proceedings of the 13th USENIX Conference on Hot Topics in Operating Systems. Berkeley, USA, 2011: 13

[24] Zhao Y, Xie Y, Yu F, et al. BotGraph: Large scale spamming botnet detection//Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation. Berkeley, USA, 2009: 321-334

[25] Curino C, Jones E, Zhang Y, Madden S. Schism: a workload-driven approach to database replication and partitioning// Proceedings of the 36th International Conference on Very Large Data Bases. Singapore, 2010, 3: 48-57

[26] Pavlo A, Curino C, Zdonik S. Skew-aware automatic database partitioning in shared-nothing parallel OLTP systems// Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data. Scottsdale, USA, 2012: 61-72

[27] Özsu T, Valduriez P. Principles of distributed database systems. 3rd Edition. USA: Prentice-Hall, 2011

[28] Sacca A, Wiederhold G. Database partitioning in a cluster of processors. ACM Transactions on Database Systems, 1985, 10(1): 29-56

[29] Garey M R, Johnson D S. Computers and Intractability: A Guide to the Theory of NP-Completeness. USA: Freeman W H and Company, 1979

[30] Eswaran K P. Placement of records in a file and file allocation in a computer network//Proceedings of the IFIP Congress on Information Processing. Stockholm, Sweden, 1974: 304-307

[31] Hammer M, Niamir B. A heuristic approach to attribute partitioning//Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data. Boston, USA, 1979: 93-101

[32] Ceri S A, Negri M, Pelagatti G. Horizontal data partitioning in database design//Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data. Boston, USA, 1982: 128-136

[33] Navathe S B, Ceri S, Wiederhold G, Dou J. Vertical partitioning algorithms for database design. ACM Transactions on Database Systems, 1984, 9(4): 680-710

[34] Malarvannan M, Ramaswamy S. Rapid scalability of complex and dynamic Web-based systems: Challenges and recent approaches to mitigation//Proceedings of the 5th International Conference on System of Systems Engineering. Beijing, China, 2010: 1-6

[35] Zilio D C. Physical Database Design Fecision Algorithms and Concurrent Reorganization for Parallel Database Systems [Ph. D. dissertation]. University of Toronto, Toronto, Canada, 1998

[36] Agrawal S, Narasayya V, Yang B. Integrating vertical and horizontal partitioning into automated physical database design// Proceedings of the ACM SIGMOD International Conference on Management of Data. Paris, France, 2004: 359-370

[37] Zilio D C, Jhingran A, Padmanabhan S. Partition key selection for a shared-nothing parallel database system. IBM Watson Research Center, USA: Techniqual Report IBM Research Report RC 19820(87739) 11/10/94, 1994

[38] Eadon G, Chong E I, Shankar S, et al. Supporting table partitioning by reference in Oracle//Proceedings of the ACM SIGMOD International Conference on Management of Data. Vancouver, Canada, 2008: 1111-1122

[39] Zilio D C, Rao J, Lightstone S, et al. DB2 design advisor: Integrated automatic physical database design//Proceedings of the 30th International Conference on Very Large Data Bases. Toronto, Canada, 2004: 1087-1097

[40] Nehme R, Bruno N. Automated partitioning design in parallel database systems//Proceedings of the ACM SIGMOD International Conference on Management of Data. Athens, Greece, 2011: 1137-1148

[41] Lee J, Muehle M, May N, et al. High-performance transaction processing in SAP HANA. IEEE Data Engineering Bulletin, 2013, 36(2): 28-33

[42] Rahimi S, Haug F S. Distributed Satabase Management Systems: A Practical Approach. Hoboken, USA: Wiley, 2010

[43] Wang Xiao-Yan, Chen Jin-Chuan, Du Xiao-Yong. ASAWA: An automatic partition key selection strategy//Proceedings of the 15th Asia-Pacific Conference. Sydney, Australia, 2013: 609-620

[44] Harizopoulos S, Abadi D J, Madden S, Stonebraker M. OLTP through the looking glass, and what we found there// Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data. Vancouver, Canada, 2008: 981-992

[45] Ghandeharizadeh S, Dewitt J D. Hybrid range partitioning strategy: A new declustering strategy for multi processor database machines//Proceedings of the 16th International Conference on Very Large Data Bases. Brisbane, Australia, 1990: 481-492

[46] Dewitt D, Ghandeharizadeh S, Schneider D, et al. The Gamma database machine project. IEEE Transactions on Knowledge and Data Engineering, 1990, 2(1): 44-62

[47] Nguyen K, Thompson T, Bryan G. An enhanced hybrid range partitioning strategy for parallel database//Proceedings of the 8th International Workshop on Database and Expert Systems Applications. Toulouse, France, 1997: 289-294

[48] Zhou J, Bruno N, Lin W. Advanced partitioning techniques for massively distributed computation//Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data. Scottsdale, USA, 2012: 13-24

[49] Rao J, Zhang C, Megiddo N, Lohman G M. Automating physical database design in a parallel database//Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data. Madison, USA, 2002: 558-569

[50] Lv Chen, Fang Jun, Han Yan-Bo. Incremental data partitioning approach based on tuple clustering. Journal of Frontiers of Computer Science and Technology, 2011, 5(8): 719-729(in Chinese)
(吕晨，房俊，韩燕波. 采用元组聚类的增量式数据分区方法. 计算机科学与探索，2011，5(8)：719-729)

[51] Turcu A, Palmieri R, Ravindran B. Automated data partitioning for highly scalable and strongly consistent transactions//Proceedings of the 7th ACM SIGOPS International Systems and Storage Conference. Haifa, Israel, 2014: 1-11

[52] Papadimitriou C H, Steiglitz K. Combinatorial Optimization: Algorithms and Complexity. USA: Prentice-Hall, 1982

[53] Danna E, Perron L. Structured vs. unstructured large neighborhood search: A case study on job-shop scheduling problems with earliness and tardiness costs. Principles and Practice of Constraint Programming, 2003, 2833: 817-821

[54] Jindal A, Dittrich J. Relax and let the database do the partitioning online//Proceedings of the 5th International Workshop on Enabling Real-Time Business Intelligence. Seattle, USA, 2011: 65-80

[55] Liroz-Gistau M, Akbarinia R, Pacitti E, et al. Dynamic workload-based partitioning for large-scale databases//Proceedings of the 23rd International Workshop on Database and Expert Systems Applications. Vienna, Austria, 2012: 183-190

[56] Wang Xiao-Yan, Chen Jin-Chuan, Du Xiao-Yong, Fan Xu. Research on automatic partitioning of appended data in parallel OLTP systems. Journal of Frontiers of Computer Science and Technology, 2013, 7(9): 800-810(in Chinese)
(王晓燕，陈晋川，杜小勇，范旭. 并行 OLTP 系统中增量数据的自动分片技术研究. 计算机科学与探索，2013，7(9)：800-810)

[57] Ulus T, Yysal M. Heuristic approach to dynamic data allocation in distributed database systems. Pakistan Journal, 2003, 2(3): 231-239

[58] Cheng C, Lee W, Wong K. A genetic algorithm based clustering approach for database partitioning. IEEE Transactions on Systems, Man, and Cybernetics-PartC: Applications and Reviews, 2002, 32(3): 215-230

[59] Corcoran A, Hale J. A genetic algorithm for fragment allocation//Proceedings of the 1994 ACM Symposium on Applied Computing. Phoenix, AZ, USA, 1994: 247-250

[60] Karlapalem K, Ng M. Query-driven data allocation for distributed database systems//Proceedings of the 8th International Conference on Database and Expert Systems Applications. London, UK, 1997: 347-356

[61] Kwok Y, Karlapalem K, Ahmad I, Ng M. Design and evaluation of data allocation algorithms for distributed multimedia database systems. IEEE Journal on Selected Areas in Communications, 2002, 14(7): 1332-1348

[62] Oommen B J, Ma D C Y. Deterministic Learning Automata Solutions to the equipartitioning problem. IEEE Transactions on Computers, 1998, 37: 2-13

[63] Van den Bout D E, Miller T K. Graph partitioning using annealed neural networks. IEEE Transactions on Neural Networks, 1990, 1(2): 192-203

[64] Ceri S, Pernici B, Wiederhold G. Distributed database design methodologies. Proceedings of the IEEE, 1987, 75(5): 533-546

[65] So S K, Ahmad I, Karlapalem K. Response time driven multimedia data objects allocation for browsing documents in distributed environments. IEEE Transactions on Knowledge and Data Engineering, 1999, 11: 386-405

[66] Bakker J A. Semantic partitioning as a basis for parallel I/O in database management systems. Parallel Computing, 2000, 26(11): 1491-1513

[67] Ahmad I, Karlapalem K, Kwok Y K, et al. Evolutionary algorithms for allocating data in distributed database systems. Distributed and Parallel Databases, 2002, 11(1): 5-32

[68] Chang C T. Optimization approach for data allocation in multidisk database. European Journal of Operational Research, 2002, 143(1): 210-217

[69] Holland M, Gibson G. Parity declustering for continuous operation on redundant disk arrays//Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems. Boston, USA, 1992: 23-35

[70] Wilson B, Navathe S B. An analytical framework for the redesign of distributed databases//Proceedings of the 6th Advanced Database Symposium. Tokyo, Japan, 1986: 77-83

[71] Brunstrom A, Leutenegger S T, Simha R. Experimental evaluation of dynamic data allocation strategies in a distributed database with changing workloads//Proceedings of the 1995 International Conference on Information and Knowledge Management. Baltimore, USA, 1995: 395-402

[72] Li Shun-Pun, Wong Man-Hon. Data allocation in scalable distributed database systems based on time series forecasting //Proceedings of the 2013 IEEE International Congress on Big Data. Santa Clara Marriott, USA, 2013: 17-24

[73] Wang Xiao-Yan, Fan Xu, Chen Jin-Chuan, Du Xiao-Yong. Automatic data distribution in large-scale OLTP applications. International Journal of Database Theory and Application, 2014, 7(4): 37-46

[74] Hong Tao, Wu Ya-Ting, Cao Bing-Yao Yanke, Yu Fei. A dynamic data allocation method with improved load-balancing for cloud storage system//Proceedings of the 31st AIAA International Communications Satellite Systems Conference. Florence, Italy, 2013: 220-225

[75] Zhang Hu, Wu Wei-Guo, Dong Xiao-She, et al. A study on data placement of extensible parallel storage system//Proceedings of the 6th IEEE/ACIS International Conference on Computer and Information Science. Melbourne, Australia, 2007: 610-615

[76] Wang Xiao-Yan, Chen Jin-Chuan, Guo Xiao-Yan, Du Xiao-Yong. A Nash-Pareto strategy based butomatic data distribution method and its supporting tool. Journal of Computer Research and Development, 2015 , 52(9): 1965-1975(in Chinese)
(王晓燕, 陈晋川, 郭小燕, 杜小勇. 基于 Nash-Pareto 策略的自动数据分布方法及支持工具. 计算机研究与发展, 2015, 52(9): 1965-1975)

[77] Wang Teng-Jiao, Lin Zi-Yu, Yang Bi-Shan, et al. MBA: A market-based approach to data allocation and dynamic migration for cloud database. Science China, 2013, 53(1): 1-18

[78] Chen Tao, Xiao Nong, Liu Fang, Fu Chang-Sheng. Clustering-based and consistent hashing-aware data placement algorithm. Journal of Software, 2010, 21(12): 3175-3185(in Chinese)
(陈涛, 肖侬, 刘芳, 付长胜. 基于聚类和一致 Hash 的数据布局算法. 软件学报, 2010, 21(12): 3175-3185)

[79] Yuan Dong, Yang Yun, Liu Xiao, Chen Jin-Jun. A data placement strategy in scientific cloud workflows. Future Generation Computing Systems, 2010, 26(8): 1200-1214

[80] McCormick W, Schweitzer P, White T. Problem decomposition and data reorganization by a clustering technique. Operations Research, 1972, 20(5): 993-1009

[81] Song Huai-Ming, An Ming-Yuan, Wang Yang, et al. Duplication elimination in large scale data intensive systems. Journal of Computer Research and Development, 2010, 47(4): 581-588 (in Chinese)
(宋怀明, 安明远, 王洋等. 大规模数据密集型系统中的去重查询优化. 计算机研究与发展, 2010, 47(4): 581-588)

[82] Zheng Pai, Cui Li-Zhen, Wang Hai-Yang, Xu Meng. A data placement strategy for data intensive applications in cloud. Chinese Journal of Computers, 2010, 33(8): 1472-1480(in Chinese)
(郑湃, 崔立真, 王海洋, 徐猛. 云计算环境下面向数据密集型应用的数据布局策略与方法. 计算机学报, 2010, 33(8): 1472-1480)

[83] Wu E, Curino C, Madden S. No bits left behind//Proceedings of the 5th Biennial Conference on Innovative Data Systems Research. Asilomar, USA, 2011: 187-190

[84] Rahm E, Marek R. Analysis of dynamic load balancing strategies for parallel shared nothing database systems//Proceedings of the International Conference on Very Large Databases. Dublin, Ireland, 1993: 182-193

[85] Yu P S, Chen M S, Heiss H U, Lee S. On workload characterization of relational database environments. IEEE Transactions on Software Engineering, 1992, 18(4): 347-355

[86] Pavlo A. On Scalable Transaction Execution in Partitioned Main Memory Database Management Systems [Ph. D. dissertation]. Brown University, Providence, USA, 2014

[87] Holze M, Ritter N. Towards workload shift detection and prediction for autonomic databases//Proceedings of the 1st Ph. D. Workshop in the 16th ACM Conference on Information and Knowledge Management, 2007: 109-116

[88] Holze M, Ritter N. Autonomic databases: Detection of workload shifts with n-gram-models//Proceedings of the ADBIS. Chicago, USA, 2008: 127-142

[89] Elnaffar S S. A methodology for auto-recognizing DBMS workloads//Proceedings of the 2002 Conference on the Centre for Advanced Studies on Collaborative Research. Toronto, Canada, 2002: 2

[90] Gupta C, Mehta A, Dayal U. PQR: Predicting query execution times for autonomous workload management//Proceedings of the 2008 International Conference on Autonomic Computing. Chicago, USA, 2008: 13-22

[91] Sapia C. PROMISE: Predicting query behavior to enable predictive caching strategies for OLAP systems//Proceedings of the 2nd International Conference on Data Warehousing and Knowledge Discovery. London, UK, 2000: 224-233

[92] Du N, Ye X, Wang J. Towards workflow-driven database system workload modeling//Proceedings of the 2nd International Workshop on Testing Database Systems. Providence, USA, 2009: 1-6

[93] Yao Q, An A, Huang X. Mining and modeling database user access patterns//Proceedings of the 16th International Symposium on Methodologies for Intelligent Systems. Bari, Italy, 2006: 493-503

[94] Pavlo A, Jones E P, Zdonik S. On predictive modeling for optimizing transaction execution in parallel OLTP systems//Proceedings of the 37th International Conference on Very Large Data Bases. Seattle, USA, 2011, 5: 85-96

[95] Ceri S, Navathe S, Wiederhold G. Distribution design of logical database schemas. IEEE Transactions on Software Engineering, 1983, 9(4): 487-504

[96] Ganguly S, Goel A, Silberschatz A. Efficient and accurate cost models for parallel query optimization//Proceedings of the 15th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems. Montreal, Canada, 1996: 172-181

[97] Chaudhuri S, Narasayya V. Autoadmin "what-if" index analysis utility. SIGMOD Record, 1998, 27(2): 367-378

[98] Cheng Y C, Gruenwald L, Ingels G, Thakkar M T. Evaluating partitioning techniques for main memory database: Horizontal and single vertical//Proceedings of the 5th International Conference on Computing and Information. Changsha, China, 1993: 570-574

[99] Manegold S. Understanding, Modeling and Improving Main-Memory Database Performance [Ph. D. dissertation]. University of Amsterdam, Amsterdam, Holland, 2002

[100] Manegold S, Boncz P, Kersten M L. Generic database cost models for hierarchical memory systems//Proceedings of the 28th International Conference on Very Large Data Bases, 2002: 191-202

[101] Richard B B, Derek L E, Gregory O M, et al. Achieving load balancing and effective caching in clustered web servers //Proceedings of the 4th International Web Caching Workshop. San Diego, CA, 1999: 159-169

[102] Lin F C H, Keller R M. The gradient model load balancing method. IEEE Transactions on Software Engineering, 1987, 13(1): 32-38

[103] Willebeek-LeMair M H, Reeves A P. Strategies for dynamic load balancing on highly parallel computers. IEEE Transactions on Parallel Distributed Systems, 1993, 4(9): 979-993

[104] Loh P K K, Wen J H, Cai W, et al. How network topology affects dynamic load balancing. IEEE Parallel and Distributed Technology, 1996, 4(3): 25-35

[105] Qin Xiao, Jiang Hong, Zhu Yi-Feng, et al. Boosting performance for I/O-intensive workload by preemptive job migration in a cluster system//Proceedings of the 15th Symposium on Computer Architecture and High Performance Computing. Los Alamitos, USA, 2003: 235-243

[106] Lakshmanan G, Li Ying, Strom R. Placement strategies for Internet-scale data stream systems. IEEE Internet Computing, 2008, 12(6): 50-60

**WANG Xiao-Yan**, born in 1981, Ph. D. candidate. Her research interests include big data management and analysis.

**CHEN Jin-Chuan**, born in 1978, Ph. D., associate professor. His research interests include uncertainty data management, big data management, etc.

**DU Xiao-Yong**, born in 1963, Ph. D., professor, Ph. D. supervisor. His research interests include database system, intelligent information retrieval, etc.

**Background**

In the fields of Data Management, data distribution is one of the most famous technologies for platform scalability. It is used to partition a series of data into disjoint data partitions, and allocate them into different data nodes. From the view of the whole system, the reasonableness of data distribution not only affects local access efficiency, but also restricts global queries and transaction processing efficiency. As data amount increasing rapidly and workload requests variously, it is a big issue to distribute data carefully in order to obtain high performance in terms of performance (including throughput, availability and scalability). Based on the analysis of data distribution problem in cloud environment, this work abstracts it as a triangle model called DaWN and conducts a survey study on the research progress of data distribution in the field of large scale OLTP applications, including data fragmentation, data allocation and workload processing. This paper also discusses the challenging problems and future research directions, which would like to help researchers pay attention to the interesting issues which are needed to address.

There have been many results and efforts which have been devoted to this area on a specific problem and obtained lots of achievements in the world. To our knowledge, we are the first to combine data, workload and node together and use data fragmentation, data allocation and workload processing to present the problem of data distribution. We expect to summarize the research results and efforts to help researchers to understand the key issues and technologies, and keep the study on this problem up with the main trends in the related research areas.

This work is supported by the State Key Laboratory of Software Development Environment Open Fund under Grant No. SKLSDE-2012KF-09, and the National Natural Science Foundation of China under Grant No. 61003086 and No. 61170010. This project aims to make advances to the architecture of data management and storage, especially in the era of big data in cloud environment. Our team has been studying and working on this research field for year, and has published a series if papers in various international conferences and journals. This paper summarizes the research progress of the data distribution in recent years.