计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2015.20140832
52(9):1965-1975,2015

基于 Nash-Pareto 策略的自动数据分布方法及支持工具

王晓燕[1,2,3] 陈晋川[1] 郭小燕[4] 杜小勇[1,3,5]
[1](数据工程与知识工程教育部重点实验室(中国人民大学) 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](易安信中国研究院 北京 100084)
[5](软件开发环境国家重点实验室(北京航空航天大学) 北京 100191)
(wxy@ruc.edu.cn)

A Nash-Pareto Strategy Based Automatic Data Distribution Method and Its Supporting Tool

Wang Xiaoyan[1,2,3], Chen Jinchuan[1], Guo Xiaoyan[4], and Du Xiaoyong[1,3,5]
[1](*Key Laboratory of Data Engineering and Knowledge Engineering* (*Renmin University of China*), *Ministry of Education*, *Beijing* 100872)
[2](*Information Center*, *The Supreme People's Court*, *Beijing* 100745)
[3](*School of Information*, *Renmin University of China*, *Beijing* 100872)
[4](*EMC Labs China*, *Beijing* 100084)
[5](*State Key Laboratory of Software Development Environment* (*Beihang University*), *Beijing* 100191)

Abstract The era of big data brings new challenges in the field of data storage and management. With the dramatic increase of data volume, automatic data distribution has been one of the key techniques and intractable problem for distributed systems. Based on the studies on data, workload and node in this field, this work abstracts the problem of data distribution as a triangle model called DaWN (data, workload, node), and summarizes their relationships with each other as data fragmentation, data allocation and workload processing. According to DaWN, it proposes an automatic solution for data distribution in large-scale on-line transaction processing (OLTP) applications, and discusses the details and interactions of each module in this consolidation architecture. Combined with our existing research, it applies the optimal equilibrium conduct of Nash-Pareto strategy into practice. According to the results of a series of experiments, the proposedapproach shows nice overall performance and effectiveness. Meanwhile, this work also implements a prototype tool called ADDvisor for automatic data distribution supporting in the expect of smoothly promoting more research work into real world practice and effectively coordinating automatic data distribution in large scale OLTP distributed applications.

Key words data distribution; triangle model; automatic solution; optimal equilibrium; on-line transaction processing (OLTP)

摘 要 大数据时代的来临为数据存储与管理提出了新的挑战.随着数据量的迅猛增加,自动数据分布逐渐成为分布式系统中的研究重点和难点.根据对数据分布问题中数据、负载和节点3个要素的研究和

收稿日期:2014-09-28;**修回日期**:2014-12-05
基金项目:软件开发环境国家重点实验室开放基金项目(SKLSDE-2012KF-09);中央高校基本科研业务费专项基金项目(14XNLQ06)
通信作者:杜小勇(duyong@ruc.edu.cn)

分析,将数据分布问题抽象为称为 DaWN(data,workload,node)的三角模型,并将 3 要素之间的相互关联关系抽象为数据分片、数据分配和负载执行 3 条纽带;据此,提出了解决自动数据分布问题的基本架构,对各功能模块的协动关系进行探讨;同时,结合已有的研究工作,采用 Nash-Pareto 优化均衡策略使得前述各机制相得益彰,实验结果验证了其有效性.为使研究工作更多地应用于实践,设计并实现了自动数据分布辅助原型工具 ADDvisor(automatic data distribution advisor),协同支持自动数据分布的执行,共同促进大规模分布式联机事务处理系统的并行性能和自动化管理技术的发展.

关键词 数据分布;三角模型;自动化解决方案;优化均衡;联机事务处理

中图法分类号 TP392

进入 21 世纪以来,很多大型的在线事务级系统平台,如淘宝、12306 等应用以及金融、电信等领域,都面临着迅速膨胀的数据规模以及日益沉重的事务处理压力.举例来说,在 2013 年淘宝双十一狂欢节[1]中,淘宝网一天内完成了约 1.88 亿笔在线交易,峰值流量 1.3 万笔/s;国内火车票预定官网 12306 每次出票都需要计算该车次全线各站点上各类各等次车票的数量,以春节预定高峰期 2014-01-06 为例[2],全天网络出票 400 万张,峰值时刻出票量超过 1 700 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,单纯依靠增加或者升级高性能计算存储节点、使用分布式并行数据库进行简单的传统数据分布和管理的解决方案已经不能完全适用,使用数据分布技术构建具有高吞吐量、高可用性和高可扩展性的分布式并行化数据管理方式日趋成为新的研究热点.

数据分布将数据分片为一系列不相交的数据片段,并按照一定的策略放置到各个数据节点上,其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,减少跨逻辑数据分区或者跨物理数据分配的数据访问,即提高访问的局部性.对于分布式应用而言,数据分布是决定大型 OLTP 系统性能的一个关键因素,其优劣直接影响到整个系统的运行效率[3-4].文献[5-6]的实验结果显示,在相同的软硬件环境设置下,不同的数据分布策略可以带来 10 倍以上的系统性能差异.虽然分布式并行计算及其执行引擎的迅速发展(如 MapReduce[7], Hadoop[8], Dryad[9])和高层次的语言支持(如 Pig[10], Hive[11], DryadLINQ[12])极大地简化了大规模分布式数据密集型应用的发展,但是,这些技术的实现和实施都是以牺牲数据的强一致性为代价的.因此,目前数据分布的设计趋势之一是转向支持符合 ACID(atomicity, consistency, isolation, durability)约束的事务级应用.大多数自动数据分布支持工具主要深入集成并应用于相关商业系统中[13-15]:IBM DB2[13]实现了一个基于组件的自主调优架构,Oracle 10g[14]开始实现了辅助调优模式中的系统运行,微软 SQL Server[15]则通过引入数据分布信息对查询计划进行进一步优化.近年来,随着云计算的兴起,学术界也纷纷展开自动化机制的设计,比如 AutoPart[16]和 H-Store[6],但通常侧重于数据分片或者数据分配等单一子问题的研究,而忽略了子问题间的相互作用,从而限制了解决方案的质量.因此,目前的数据分布问题研究相对较为松散,特别是在大规模 OLTP 应用中,还没有成熟的系统化数据分布模型和自动化解决方案.

根据对数据分布问题的研究和分析,本文提出称为 DaWN(data,workload,node)的三角模型来刻画数据分布问题,通过数据、负载和节点 3 个基本要素回答数据分布中“分什么”、“怎么分”和“分到哪”这 3 个基本问题,并将 3 要素间的相互关联关系抽象为数据分片、数据分配和负载执行 3 条纽带;同时,基于我们已有的研究工作,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;此外,还设计了称为 ADDvisor(automatic data distribution advisor)的面向大规模 OLTP 应用的自动数据分布辅助原型工具,用于支持不同系统的自动数据分布方案设计,协同促进大规模 OLTP 应用的并行性能和自动化的管理技术.

1 模型与方案

数据分布(data distribution)是指在分布式系统中,按照某种策略将数据分片成逻辑片段并将这

些片段存储到不同的物理节点上来处理工作负载的综合过程,如图 1 所示:

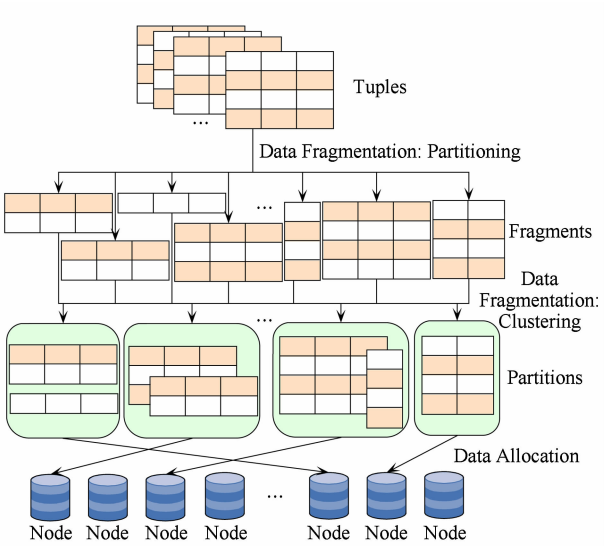


Fig. 1 Data distribution processing.

图 1 数据分布过程

为了进一步阐述 OLTP 环境中的自动数据分布问题,我们考虑以下应用场景:在一个具有大规模数据量和工作负载的数据中心,管理员需要将数据和负载配置到多个存储节点上.需要解决的一个关键问题是如何使得每个事务尽可能地在一个数据节点上执行完成,以避免跨节点的通信代价;同时,还要尽可能保证负载均衡,以充分发挥每一个节点的运算能力.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的数量.如果管理员还要考虑数据在不断增长、工作负载的内容可能发生较大变化,上述问题将变得更为复杂.总体来讲,该系统的设计期望是尽可能优化系统性能(如响应时间、吞吐量等),减少工作负载执行代价,同时,保证节点上的数据和负载均衡.

1.1 DaWN 模型

通过对数据分布问题的分析,我们提出了以代价模型为核心,以数据、负载和节点为 3 要素,以要素间的互动关系(数据分片、数据分配和负载执行)为纽带的数据分布三角模型,称为 DaWN,如图 2 所示.

在 DaWN 模型中,3 个顶点要素分别是:

1) 数据(data).指系统中运行的所有数据元组,其元数据包括所有相关的关系模式.

2) 负载(workload).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.

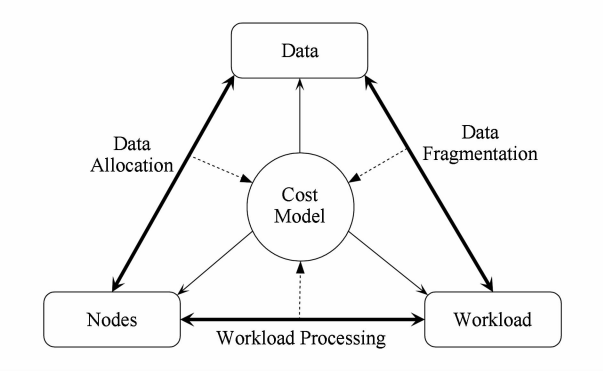


Fig. 2 DaWN model.

图 2 DaWN 模型

3) 节点(node).用于数据存放和负载运行的物理节点,其元数据包括节点组织方式、物理信息(如磁盘容量、CPU 频率、内存大小)等.

根据 3 个要素之间的相互关联与制约,数据分布问题的解决需要考虑以 3 条边上的关系为代表的 3 个方面的问题:

1) 数据分片(data fragmentation).也称为数据划分,从逻辑上将全体数据按照它们之间的相互关系划分为逻辑片段,即子关系.

2) 数据分配(data allocation).从物理上将数据分片中划分好的逻辑片段放置到存储节点上.

3) 负载执行(workload processing).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子关系进行调度和运行.

一些早期的文献[17]已经证明,数据分布问题及其上述 3 个子问题均为 NP-hard 问题.在任何一个分布式系统中,要达到所有数据访问的完全本地化是非常困难的.为了尽可能减少跨节点的分布式访问、降低远程访问开销,如何更好地提高系统的本地访问性是分布式系统成功的关键.因此,作为系统运行的基本目标,代价模型(cost model)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,用于平衡系统性能与运行代价,处于整个模型的核心位置.

1.2 自动分布架构

在文献[18]中,作者分析了传统数据库系统架构在用户需求、硬件特性等方面所面临的问题,他认为要获得性能上的显著提高,新一代系统最需要实现的是:没有旋钮(即去除人工操作),也就是说,如果要在信息技术的演进中取胜,“一切自动”(即自动修复、自动维护、自动调优等)的系统才是终极目标.

为此,我们根据 DaWN 模型提出了一个解决自动数据分布问题的系统架构,如图 3 所示. 它的输入是数据集、工作负载以及节点信息,输出是数据分布策略,其目标是在保证系统性能的前提下降低整体运行的预期成本.

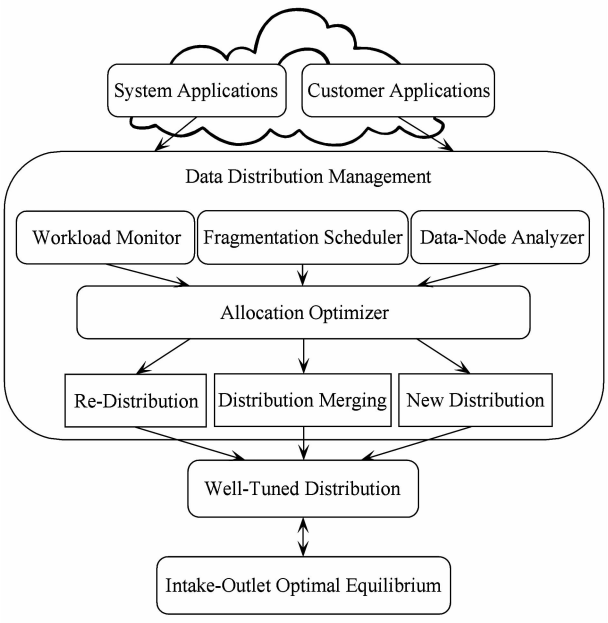


Fig. 3 Automatic distribution architecture.

图 3 自动分布架构

1) 工作负载监视器(workload monitor). 用来监控和获得运行时工作负载的变化信息,用于后续数据分布和负载运行的调整.

2) 数据节点分析器(data-node analyzer). 用于分析数据节点的运行能力,预测进一步分割部分持有和计算运行本身的需要和能力.

3) 数据分片调度器(fragmentation scheduler). 是一个可以常驻内存缓存的结构,将在系统中保存数据碎片的信息,告诉应用程序到哪里找到需要响应用户请求的数据.

4) 数据分配优化器(allocation optimizer). 将使用从工作负载监视器、数据节点分析器和全局分布索引器获得的信息执行初始数据分布,并在运行时执行其他的操作(包括新生分布、分布合并和重新分布等),从而让整个系统以一种动态的方式自适应地调整数据分布状态. 当然,这对于将要研究的数据进出平衡和迁移也是有重要影响的.

在初始配置阶段,解决方案需要收集数据、负载和节点的相关信息,通过分析应用需求和特点,通过分片设计确定数据分配单位,根据频率表和极化表将数据片段分配到物理节点上,并根据计划表构造局部运行模式. 在动态分布阶段,解决方案需要通过对数据、负载和节点的变化状态监控,定期分析关联矩阵,决定数据分布操作,同时避免逻辑和物理架构的混淆,达到自动分布、存储、管理和运行的目标.

通过这个自动分布架构,希望达成以下目标:

1) 自动化(automation). 自动地进行数据分布中各因素之间的关联和调节,尽量减少人工干预甚至不干预,完成"分布即服务"(distribution-as-a-service)的设计理念.

2) 动态化(dynamism). 根据运行的数据、负载和节点变化,动态地进行数据分布的处理和调整,达到系统整体性能上的优化,完成"按需分布"(distribution-on-demand)的设计理念.

2 策略实施

根据文献[6],当前大规模 OLTP 应用中单个事务的运行通常是迅速而短暂的,需要系统迅速响应,而且每个事务触及到的元组数量很少且具有较强的语义相关性,不需要进行全表扫描或执行大型连接操作;此外,在实际应用中,工作负载通常是预定义的事务模板或存储过程的重复执行. 对于大多数 OLTP 应用而言,数据分布实施的压力主要来自于系统运行中的动态变化:数据量不断增长,数据关联关系和负载访问模式不断变化,物理节点的存储能力和处理性能也随行就市. 因此,自动分布架构的实施不仅要在系统的初始数据、给定负载和节点状态下找到一个较好的数据分布,并要在各因素变化时进行动态调整,保持系统性能的优化均衡.

2.1 优化均衡设计

根据 DaWN 模型,自动分布架构的实施需要解决 3 个相互关联的问题:一个全局性的关系应当如何分片,划分后的数据片段应当如何分配到节点,以及如何协同数据分片和分配实现优化均衡目标. 在已有的研究工作[19-21]中,我们已经分别解决了 OLTP 应用中的自动数据分片和自动数据分配问题,本文的研究重点是将已有的工作整合到自动数据分布策略中,协同实现数据分布的优化均衡. 因此,我们以节点作为参与者(player),以节点上的数据存储和负载执行作为策略(strategy),以节点采取策略后的系统性能和运行代价作为实现 DaWN 中代价模型的效用函数(utility function),结合文献[22-23]设计

的优化均衡方法,使得整个系统在采用所选择的数据分布策略后,达到保持 Nash 均衡且保证 Pareto 最优的状态.

我们以成本和性能作为代价模型的度量标准,根据数据分布问题的特点,效用函数由执行代价和吞吐量这 2 个部分构成. 假设 t 是每一回合的持续时间,s_i 为每一回合中节点 i 需向节点 j 提供的负载请求量,其中 $s_1\in\{0,1,\cdots,m\}$,$s_2\in\{0,1,\cdots,n\}$,$m,n\in\mathbb{N}$.

定义 1. 单位执行代价 C_i. 节点 i 回应节点 j 执行一个分布式负载运行请求所需要的数据量.

定义 2. 单位吞吐量 T_i. 单位时间里,节点 i 接受的负载运行请求所需要的数据量.

由此,每一回合节点 i 的吞吐量和执行代价分别为 s_jT_i 和 $s_iC_i(i,j\in\{1,2\}$且 $i\neq j)$.

定义 3. 效用函数 u_i. 设置每一回合中的效用函数为

$$u_i = s_jT_i - s_iC_i, \tag{1}$$

其中,$i,j\in\{1,2\}$且 $i\neq j$.

效用函数反映了节点间的博弈关系,在此,我们假设参与节点均尽力服务时的效用不低于 0,即 $nT_1-mC_1\geqslant 0$,$mT_2-nC_2\geqslant 0$. 作为一个策略组合(strategy profile),Nash 均衡要求任何参与者都不可以通过单方改变策略的方式来提高效用,而 Pareto 最优则是任何一个参与者的效用在不损害其他参与者效用的前提下不能得到提高的状态. 如果自动数据分布策略中的博弈只进行有限次,博弈结束的状态是可知的,此时 Nash 均衡为(0,0). 但现实中需要考虑无限重复博弈的情况[22],因此参与节点的总效用函数为

$$U_i = \lim_{t\to\infty}\sum_{t=0} u_i. \tag{2}$$

定理 1. 数据分布策略中至少存在一个 Nash 均衡策略的效用组合大于(0,0).

证明. 根据式(1),当参与者 i 选择 $s_i=0$ 时,最小效用为 0;选择 $s_i=1$ 时,最小效用为 $-C_i$. 因此节点 i 会选择策略 $s_i=0$,同理,节点 j 会选择策略 $s_j=0$. 因此,数据分布策略的极值组合为(0,0),对应保留效用组合为(0,0). 由于无限重复博弈中至少存在一个大于保留效用的 Nash 均衡策略,即其效用组合大于(0,0). 证毕.

由此,数据分布策略的可行效用组合的集合可以表示为 $V_0=\{v\mid u(s_1,s_2)=v\}$ 的凸包. 结合保留效用组合(0,0)得到的个体理性效用组合的集合表示为 $V_1=\{v\mid v\in V_0$ 且 $s_1>0,s_2>0\}$ 的凸包.

图 4 中以 u_1 和 u_2 分别表示 X 轴和 Y 轴,任何一个策略对应的效用组合都能够以坐标系上的对应点表示,由 $OABC$ 组成的平行四边形区域就是可行效用组合,四边形 $ODBE$ 组成的区域即为个体理性效用组合集合.

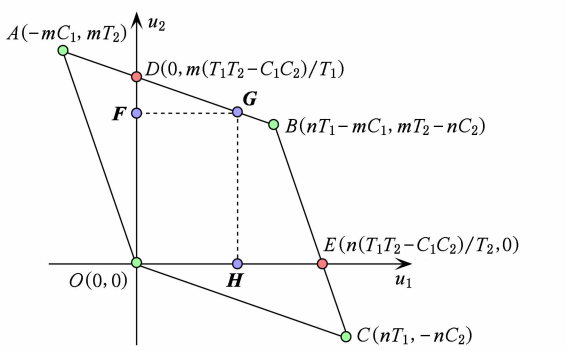


Fig. 4 Utility of strategy profiles.

图 4 策略效用组合示意图

定理 2. 数据分布策略中存在满足 Pareto 最优的 Nash 均衡策略.

对于效用组合 $v\in V_0$,当且仅当不存在任何 $v_i'\geqslant v_i$(其中,$v'\in V_0$ 且 $i\in\mathbb{N}$)时,是 Pareto 最优策略. 该策略可表述为对图 4 中线段 DB 和线段 BE 上任一点 $F(x_1,y_1)$,不存在任何点 $F'(x_2,y_2)\in V_1$,使得 $x_2\geqslant x_1$ 时的 $y_2>y_1$ 或者 $y_2\geqslant y_1$ 时的 $x_2>x_1$. 采用加辅助线的方法并使用反证法,易证得线段 DB 和 BE 上对应的策略组合满足 Pareto 最优(证明从略). 因此,满足 Pareto 最优且 Nash 均衡的数据分布策略组合在理论上是存在的. 在实际实施中,还有一些现实需求和限制,需要对 Nash 均衡进行精炼. 在此,我们选择实现 $\max(U_1\times U_2)$ 达到比例公平作为精炼标准,可通过式(2)在每个回合中实现.

假设效用组合 $(s_2T_1-s_1C_1,s_1T_2-s_2C_2)$ 对应的 Nash 均衡策略组合的集合为 $s^*=(s_1,s_2)$. 设 $F(x_1,y_1)$ 是折线段 DBE 上的任意一点,点 F 和 H 的坐标分别为 $(x_1,0)$ 和 $(0,y_1)$,将 $\max(u_1\times u_2)$ 转化为求矩形 $OFGH$ 的最大面积 $\max(S_{OFGH})$ 的几何问题:

1) 如果 $\dfrac{n}{m}\geqslant\dfrac{1}{2}\left(\dfrac{T_2}{C_2}+\dfrac{C_1}{T_1}\right)$,那么点 G 在线段 DB 上可以得到 $\max(S_{OFGH})$. 由点 A 和 B 可以得到 $C_2x_1+T_1y_1+mC_1C_2-mT_1T_2=0$,那么由此可以得到 $S_{OFGH}=x_1y_1=\dfrac{(mT_1T_2-mC_1C_2-C_2x_1)x_1}{T_1}$. 我们

对 x_1 求导 $\frac{\mathrm{d}(S_{OFGH})}{\mathrm{d}x_1}=\frac{mT_1T_2-mC_1C_2-C_2x_1}{T_1}$，令 $\frac{\mathrm{d}(S_{OFGH})}{\mathrm{d}x_1}=0$，可得 $x_1=\frac{m(T_1T_2-C_1C_2)}{2C_2}$ 和 $y_1=\frac{m(T_1T_2-C_1C_2)}{2T_1}$. 将它们代入式(1)，可以解得效用组合 (x_1,y_1) 对应的策略组合为 $s_1=m$ 和 $s_2=m\left(\frac{T_2}{2C_2}+\frac{C_1}{2T_1}\right)$.

2) 如果 $\frac{n}{m}\leqslant\frac{2C_1T_2}{C_1C_2+T_1T_2}$，那么点 G 在线段 DE 上可以得到 $\max(S_{OFGH})$，采用 1)中的方法，我们可以得到 $x_1=\frac{n(T_1T_2-C_1C_2)}{2T_2}$ 和 $y_1=\frac{n(T_1T_2-C_1C_2)}{2C_1}$，由此可以解得 (x_1,y_1) 对应的策略组合 $s_1=n\left(\frac{C_2}{2T_2}+\frac{T_1}{2C_1}\right)$ 和 $s_2=n$.

3) 如果 $\frac{2C_1T_2}{C_1C_2+T_1T_2}\leqslant\frac{n}{m}\leqslant\frac{1}{2}\left(\frac{T_2}{C_2}+\frac{C_1}{T_1}\right)$，那么在线段 BD 和 BE 的交点 B 上可以得到 $\max(S_{OFGH})$，采用 1)中的方法，我们可以得到 $x_1=nT_1-mC_1$ 和 $y_1=mT_2-nC_2$，由此可以解得 (x_1,y_1) 对应的策略组合 $s_1=m$ 和 $s_2=n$.

综合上述 3 种情况，得到优化后的策略组合 $s^*=(s_1,s_2)$ 可以达到 Nash 均衡且满足 Pareto 最优，如式(3)所示：

$$s^*=\begin{cases}\left(m,\frac{m}{2}\left(\frac{C_1C_2+T_1T_2}{C_2T_1}\right)\right),\\ \quad\frac{n}{m}\geqslant\frac{C_1C_2+T_1T_2}{C_2T_1};\\ (m,n),\ \frac{2C_1T_2}{C_1C_2+T_1T_2}<\\ \quad\frac{n}{m}<\frac{C_1C_2+T_1T_2}{C_2T_1};\\ \left(\frac{n}{2}\left(\frac{C_1C_2+T_1T_2}{C_1T_2}\right),n\right),\\ \quad\frac{n}{m}\leqslant\frac{2C_1T_2}{C_1C_2+T_1T_2}.\end{cases}\tag{3}$$

2.2 自动分布策略

作为系统化整合的初步尝试，在本文的实施中，针对 OLTP 的应用具有以下特性：数据量不断增加、负载模式按比例调和、节点状态相对稳定. 按照自动分布架构的设计，基于已有的研究工作[19-21]，我们结合 Nash-Pareto 优化均衡理论设计了自动数据分布策略 ADDS(automatic data distribution strategy)，如图 5 所示，它主要包含 2 个阶段：

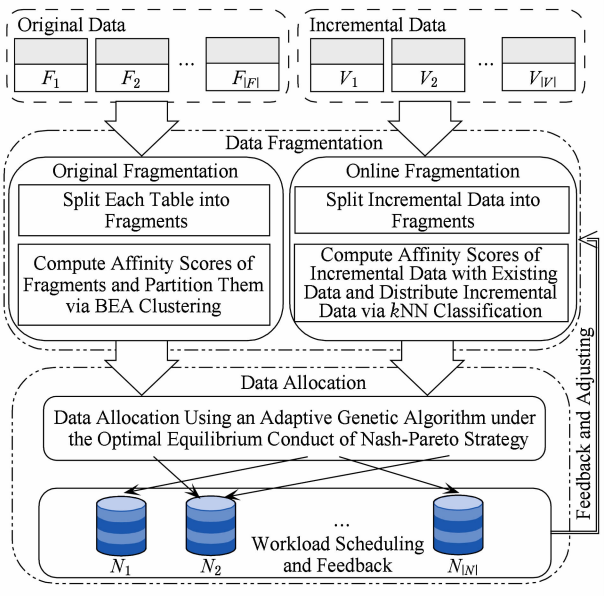


Fig. 5 Processing flow of ADDS.

图 5 ADDS 处理流程

1) 数据分片阶段. 首先使用文献[19]中的分片键自动选择策略为需要进行分片的数据表自动选择分片键，然后采用文献[20]中提出的自动数据分片方法，对于初始数据使用初始划分模块，根据给定的工作负载计算数据碎片间的亲和度，使用键能算法(bond energy algorithm, BEA)进行碎片聚合，生成数据分片. 对于新增数据，定期调用在线划分模块，根据负载执行特性和节点状态，使用 k-最邻近算法(k-nearest neighbor, kNN)进行碎片聚合，生成数据分片. 该阶段为数据分布提供合适粒度的增量数据分片方案.

2) 数据分配阶段. 采用文献[21]提出的自动数据分配方法，对于从前一阶段得到的数据分片，使用自适应的遗传算法进行训练和聚合，并根据式(3)结合负载执行特性和节点状态将数据分片自动分配到存储节点上. 同时，记录分配方案，并结合负载执行和节点特性对下一步的数据分布方案进行动态反馈与调整.

2.3 实验分析

在实验中，我们使用 8 台配备有 Intel® Xeon® 5645 (2.4 GHz)×6 核×2 处理器、48 GB 内存、300 GB SAS 硬盘的 HP 服务器，以 Redhat-release-5server-5.5.0.2 作为操作系统；所有的实验运行所需和所产生的数据都存储在 MySQL 5.4.3 中；数据集及事务的生成均根据 TPC-C 5.11[24]产生.

根据硬件条件，在初始阶段 TPC-C 数据集的比例因子设定为 100，每一次数据增加中增量数据的

比例因子为 50,直到整个数据集的比例因子达到 500. 在每一轮的负载执行中,根据 TPC-C 设计比例混合的 5 种事务数量设定为 5 000. 在实验中,我们分别使用 Hashing,Round-Robin 和 ADDS 共 3 种不同的数据分配策略进行了结果对比. 每一种策略在每一个阶段的的执行中,运行相同的查询负载集 5 次,以获得相对平稳的运行结果.

1) 分布式事务的数量及其变化趋势. 采用不同的数据分布策略在各阶段执行相同的工作负载时得到的分布式事务的比率如图 6 所示. 与 Hashing 和 Round-Robin 相比,采用 ADDS 策略所产生的分布式事务的比率平均可以减少 28.45%. 这是因为在 ADDS 策略的实施中,根据数据集和工作负载模式,考虑了相关语义信息和数据依赖,将可能被同时访问的数据更多分布到相同的数据节点上,从而减少分布式事务数量,降低数据通信代价. 根据图 7 所展示的分布式事务数量的变化趋势,使用 Hashing 和 Round-Robin 策略在执行过程中的变化趋势是相似的,而同等条件下采用 ADDS 策略比采用前述二者显示出放缓的增长趋势. 这是因为运行过程中,ADDS 的数据分布会根据新增数据和负载变化,在

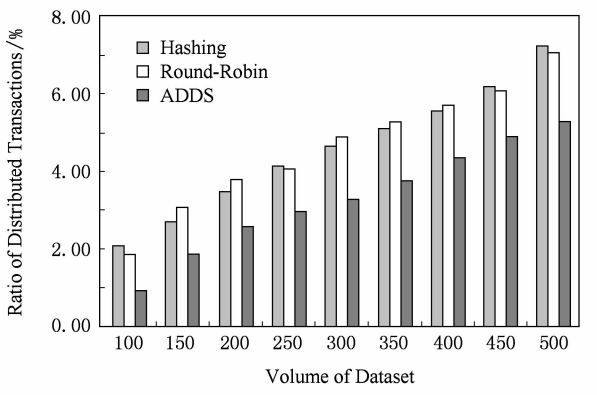


Fig. 6 Ratio of distributed transactions.

图 6 分布式事务比率

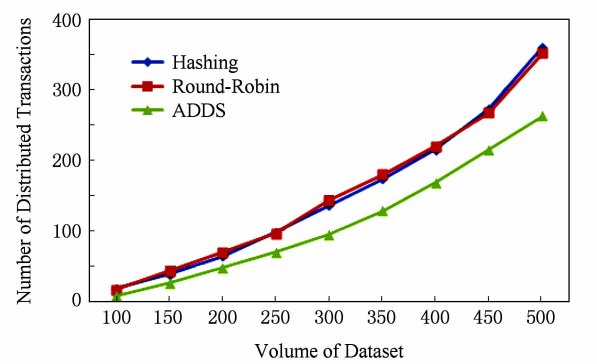


Fig. 7 Number of distributed transactions.

图 7 分布式事务数量

最优均衡的目标指引下,通过 kNN 算法聚类和自适应遗传算法的实施进行动态生成和调整,数据分布结果也会不可避免地跟初始结构有所差异. 在实际应用中,如果这个变化缓慢增大到一定程度或者对应数据节点的存储容量达到极限,将意味着我们需要调整使用数据迁移或者节点合并等方式对这些数据分布进行处理. 对这一问题的讨论超出了本文的研究范围,我们将在后续的工作中进行研究和探讨.

2) 节点上的工作负载. 采用不同的数据分布策略在各阶段所得到的数据节点上的平均工作负载量如图 8 所示. 使用 Hashing 和 Round-Robin 策略时,每一个阶段的工作节点上需要处理的工作负载都比使用 ADDS 策略多,负载增加程度平均要高 10.42%. 这是因为如果所需数据分布在多个节点上,该事务的执行不但需要访问多个工作节点,而且需要增加节点间的通信代价. Hashing 和 Round-Robin 都是均分策略,而 ADDS 则以减少跨节点事务数量为目标,考虑了事务执行可能涉及的数据子集,并尽可能将其放在相同的节点上,同时保持系统均衡,提高负载执行效率和系统吞吐量.

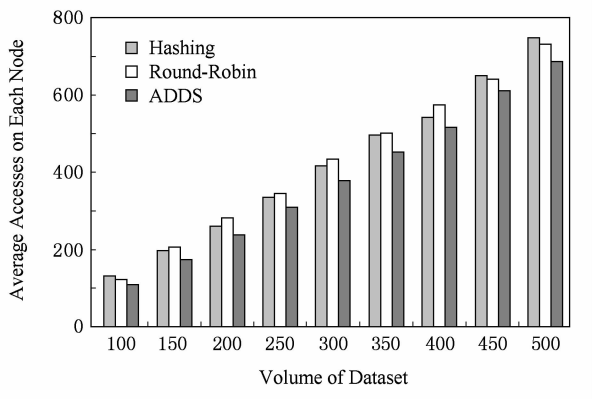


Fig. 8 Workload on nodes.

图 8 节点上的工作负载

3) 节点上的数据相关性. 采用不同的数据分布策略在各阶段所得到的节点上数据的相关性,如图 9 所示. 由于 ADDS 将更有可能同时被访问到的数据放在相同的节点上,增加了同一节点上数据片段的相关性,因此分布结果更为相关,即更有可能同时被访问到;同时,由于都是均分策略,使用 Hashing 和 Round-Robin 得到的各阶段节点上的数据相关程度是非常相近的.

4) 执行时间. 采用不同的数据分布策略所需要的执行时间,如图 10 所示. 由于在每一个阶段中数据的增长量相同,图 10 同时反映了 3 种策略的执行时间的线性增长趋势以及执行时间与数据量之间的

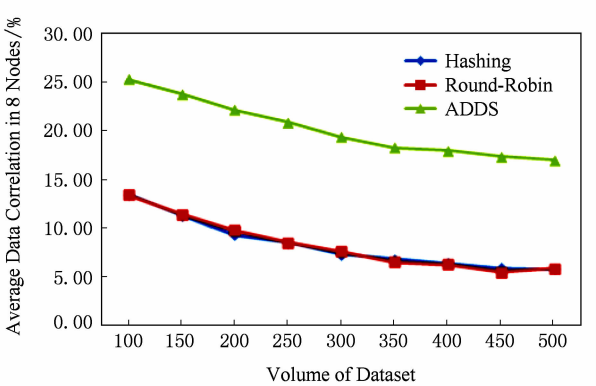


Fig. 9 Data correlation on nodes.

图 9 节点上的数据相关度

潜在关系. 根据图 10, Hashing 和 Round-Robin 的执行时间极为相近, 而 ADDS 则比二者的平均降低了约 10%, 这是因为 ADDS 不需要进行元组级别的运算; 同时, ADDS 与数据量的线性相关特征为系统的可扩展性提供了可能的支持.

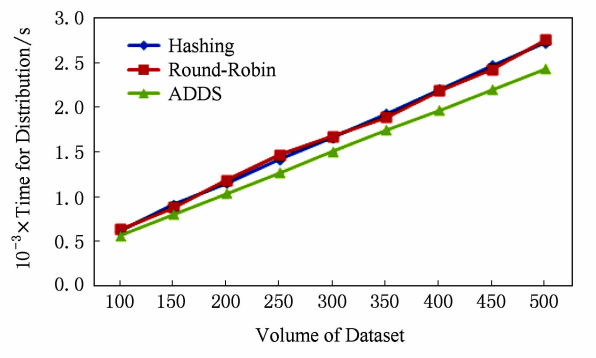


Fig. 10 Time for distribution.

图 10 执行时间

3 辅助支持工具

为了将以上研究更广泛地应用于实践, 基于 DaWN 模型和自动分布架构, 我们构建了辅助支持工具 ADDvisor 辅助实现有效的数据分布解决方案.

3.1 工具设计

为了得到自适应的数据分布策略, 最大程度地利用已有输入内容中的潜在信息是非常重要的. 对于一个设计良好的 OLTP 应用, 系统的基本信息(如数据模式、负载模式及特性、节点信息等)及初始数据、工作负载和节点状态等是可以在进行数据分布之前获得的, 我们设计将其作为 ADDvisor 的输入信息. 同时, 将目前普遍应用的 Hashing, Round-Robin 等算法, 与针对不同应用设计的各种启发式数据分片或分配算法进行了具体实现, 并放入算法选择池, 结合实际应用中的数据、负载和节点信息设定等输入信息进行计算和模拟, 对运行结果进行可视化呈现, 从而辅助支持自动数据分布方案的设计. 辅助支持工具 ADDvisor 的基本设计思想如图 11 所示. 它以 Java 作为开发语言, 使用 Eclipse 作为开发环境, 可以运行在装有 Java 虚拟机的平台上, 通过标准化的图形用户界面提供自动数据分布的辅助支持.

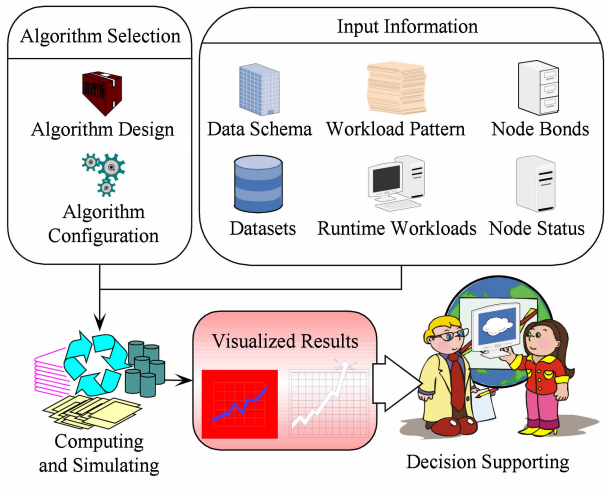


Fig. 11 Design of ADDvisor.

图 11 ADDvisor 设计

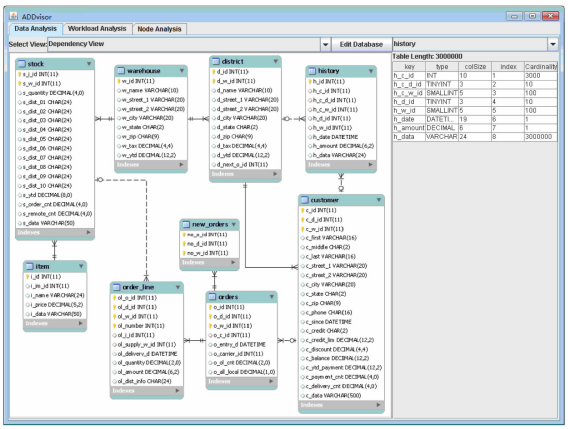
3.2 ADDvisor 实现

已经实现的 ADDvisor 原型是一个独立的具有图形用户界面的应用程序, 如图 12 所示, 它可与多种数据库系统兼容. 用户通过登陆界面选择和设置相关参数, 提供数据、负载和节点的相关信息, 并同时确定使用的数据库(如 MySQL, PostgreSQL 等)类型.

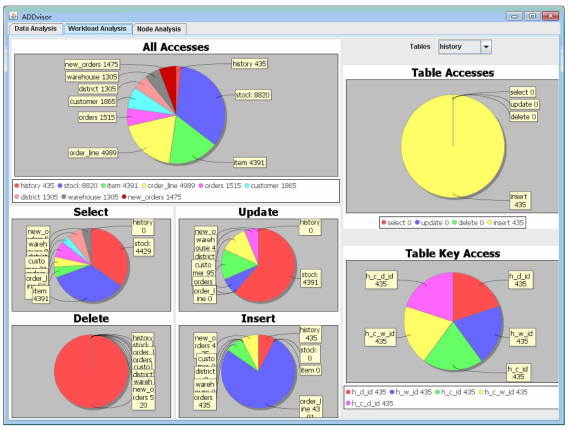
在运行实施中, ADDvisor 会根据用户需求即时呈现已有信息, 用户可以据此进行算法选择和设置、评估和比较数据分布方案. 它可以自动生成具体方案的模拟计算结果, 并在图形化界面中使用不同形式的统计图表展示给用户, 以提供有效的数据分布决策支持信息.

ADDvisor 的实际运行演示如图 12 所示, 其中: 图 12(a)展示了数据分析模块, 通过关系表之间的主外键依赖关系和比例因子设定, 可以呈现系统的数据模式, 明晰关系表之间的相关关系, 并辅助进行分片键的选择. 图 12(b)展示了负载分析模块, 通过负载对数据的访问信息统计, 呈现对应系统运行模式下工作负载对每一个关系表的访问和对整个数据的访问信息统计图表. 图 12(c)展示了节点分析模块,

通过交互式的算法选择和设定以及实际应用中的节点信息设置,对输入的数据和负载信息进行模拟,获得数据和负载的分布信息,计算可能产生的分布式事务的数量,并预测和提示可能需要进行迁移的数据元组和迁移量.



(a) Data analysis



(b) Workload analysis

(c) Node analysis

Fig. 12 Implementation of ADDvisor.

图 12 ADDvisor 实现

在 ADDvisor 的辅助下,用户可以根据实际需求,非常方便地配置各种数据分布方案. 由于采用可视化方式进行生动展现,ADDvisor 可以帮助用户更为直观和迅速地了解不同方案的执行效果,通过传递相应方案的专业领域信息帮助用户进一步提高数据分布质量.

4 相关工作

根据 DaWN 模型,以下主要从数据分片、数据分配和负载执行 3 个方面,对面向大规模 OLTP 的数据分布问题的相关研究和进展进行归纳和总结.

4.1 数据分片

数据分片的基本逻辑形式有 2 种:水平分片和垂直分片. 它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. 大规模 OLTP 应用主要采用水平分片以保证数据访问的完整性和独立性,主要策略有 2 种:

1) 简单分片. 以数据为中心,采用一次或多次简单策略对表结构进行划分,常用算法[25] 主要有 Hashing,Range,Round-Robin 等,应用于大多数商业数据库系统中,但由于没有考虑到在数据访问模式(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. 根据数据与负载间的具体应用特点进行分片,寻求获得最优方案的可能性. 其中,文献[3]提出的 Schism 将每个元组视为一个节点,将负载执行中被同时访问的元组节点之间用边相连,采用超图划分算法进行分片,有效减少分布式事务的数量. 文献[4]对分片可能产生的数据倾斜问题进行了研究,采用大规模邻域搜索方法进行改进;但是如果有新增数据,二者均需进行重新计算和划分. 本文采用的是文献[20]提出的方法,应用 Table-as-a-Column 理念,采取表内 Range 分片、表间列式聚集的方式,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地结合起来,生成一个合适粒度的分片方案.

4.2 数据分配

数据分配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. 实际采用的策略可以分为 2 类:

1) 静态分配. 这类方案在计算过程中无须改变分配策略,在计算执行前可预先确定,主要方法有 Hashing,Round-Robin 等,但因不能根据负载和节点特性进行动态调整,会影响应用系统的性能和稳定.

2) 动态分配. 这类方案需要在系统运行过程中对新增数据进行分配,在计算执行前不能预先确定. 其中,文献[26]提出了一种利用时间序列模型进行短期负载预测的算法,提前进行节点数目调整和片段重新分配,适用于时序性较强的应用. 文献[27]提出以市场理念为基础的控制方法,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分配和迁移,但不完全适用于大型 OLTP 应用. 本文采用的是文献[21]提出的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.

4.3 负载执行

负载执行要求综合数据、负载和节点的信息,在保证系统性能的前提下,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. 已有的负载执行策略主要有 2 类:

1) 静态策略. 根据系统的静态信息预先在各节点上平均分配负载,主要方法有 Random, Hashing 等. 它们简单、高效、易于实现,但由于没有充分考虑系统运行过程中节点上的数据和负载变化,非常容易导致节点间的负载不均和数据倾斜.

2) 动态策略. 根据各节点的实时负载和响应情况自适应地调整节点间的负载处理比例,避免节点过载时依然收到大量负载执行请求,减少负载执行平均响应时间,提高系统整体性能,典型方法主要有梯度模型策略[28]、集中式调度策略[29]等. 在本文中,我们采用 Nash-Pareto 策略对数据、负载和节点关系进行分析,并在各因素变化时进行动态调整,保持系统性能的优化均衡.

5 结束语

大数据时代的来临为事务型应用中数据的存储与管理带来了新的发展机遇. 根据对数据分布问题中数据、负载和节点 3 个要素的研究和分析,本文提出了数据分布问题的三角模型 DaWN,并将 3 要素之间的相互关联关系抽象为数据分片、数据分配和负载执行 3 条纽带;据此,本文提出了解决自动数据分布问题的基本架构,对各功能模块的协动关系进行探讨,协同支持自动数据分布的执行;同时,采用 Nash-Pareto 优化均衡策略对前述各机制进行了具体的实施,实验结果验证了其有效性;此外,为使研究工作更多地应用于实践,设计并实现了自动数据分布辅助原型工具 ADDvisor,协同支持自动数据分布的执行,共同促进大规模 OLTP 分布式系统的并行性能和自动化管理技术的发展.

在未来的工作中,自动数据分布技术的有效实施还需要基于现实世界里的业务需求进行优化和调整,而诸如动态变化的工作负载的均衡、数据复制的利用率等其他问题也应予以认真考量. 在后续研究中,我们将对以上问题进行进一步的研究和探讨.

参 考 文 献

[1] Incitez China. Taobao total sales reached USD 5.7 billion on one single day [EB/OL]. (2013-11-14) [2014-04-23]. http://www.chinainternetwatch.com/4691/taobao-bachelors-day

[2] Zhihu. Is it feasible to have 12306 outsourced to enterprises like Alibaba and IBM [EB/OL]. (2014-01-09) [2014-04-23]. http://www.qianzhan.com/qzdata/detail/308/140109-d3904418.html (in Chinese)
(知乎. 12306 外包给阿里巴巴、IBM 等大企业做是否可行 [EB/OL]. (2014-01-09) [2014-04-23]. http://www.qianzhan.com/qzdata/detail/308/140109-d3904418.html)

[3] Ke Q, Prabhakaran V, Xie Y, et al. Optimizing data partitioning for data-parallel computing [C] //Proc of the 13th Workshop on Hot Topics in Operating Systems (HotOS XIII). Berkeley, CA: USENIX Association, 2011: 1-5

[4] Zhao Y, Xie Y, Yu F, et al. BotGraph: Large scale spamming botnet detection [C] //Proc of the 6th USENIX Symp on Networked System Design and Implementation. Berkeley, CA: USENIX Association, 2009: 321-334

[5] Curino C, Jones E, Zhang Y, et al. Schism: A workload-driven approach to database replication and partitioning [J]. Proceedings of the VLDB Endowment, 2010, 3(1): 48-57

[6] Pavlo A, Curino C, Zdonik S. Skew-aware automatic database partitioning in shared-nothing parallel OLTP systems [C] //Proc of the 2012 ACM SIGMOD Int Conf on Management of Data. New York: ACM, 2012: 61-72

[7] Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters [C] //Proc of the 6th Symp on Operating Systems Design and Implementation. Berkeley, CA: USENIX Association, 2004: 137-150

[8] White T. Hadoop: The Definitive Guide Paperback [M]. 3rd ed. Sebastopol, CA: O'Reilly Media, 2012

[9] Isard M, Budiu M, Yu Y, et al. Dryad: Distributed data-parallel programs from sequential building blocks [C] //Proc of the 2nd European Conf on Computer Systems. New York: ACM, 2007: 59-72

[10] Olston C, Reed B, Srivastava U, et al. Pig latin: A not-so-foreign language for data processing [C] //Proc of the 2008 ACM SIGMOD Int Conf on Management of Data. New York: ACM, 2008: 1099-1110

[11] Capriolo E, Wampler D, Rutherglen J. Programming Hive Paperback [M]. Sebastopol, CA: O'Reilly Media, 2012

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

[13] Wiese D, Rabinovitch G, Reichert M, et al. ATE: Workload-oriented DB2 tuning in action [C] //Proc of 2009 Databanksystem in Business, Technology and Web. Bonn: GI, 2009: 620-623

[14] Weikum A, Mönkeberg A, Hasse C, et al. Self-tuning database technology and information services: From wishful thinking to viable engineering [C] //Proc of the 28th Int Conf on Very Large Data Bases. New York: ACM, 2002: 20-31

[15] Zhou J, Bruno N, Lin W. Advanced partitioning techniques for massively distributed computation [C] //Proc of the 2012 ACM SIGMOD Int Conf on Management of Data. New York: ACM, 2012: 13-24

[16] Papadomanolakis S, Ailamaki A. AutoPart: Automating schema design for large scientific databases using data partitioning [C] //Proc of the 16th Int Conf on Scientific and Statistical Database Management. Piscataway, NJ: IEEE, 2004: 383-392

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

[18] Stonebraker M, Madden S, Abadi D, et al. The end of an architectural era (it's time for a complete rewrite) [C] //Proc of the 33rd Int Conf on Very Large Data Bases. New York: ACM, 2007: 1150-1160

[19] Wang X, Chen J, Du X. ASAWA: An automatic partition key selection strategy [C] //Proc of the 15th Asia-Pacific Web Conf. Berlin: Springer, 2013: 609-620

[20] Wang Xiaoyan, Chen Jinchuan, Du Xiaoyong, et al. Research on automatic partitioning of appended data in parallel OLTP systems [J]. Journal of Frontiers of Computer Science and Technology, 2013, 7(9): 800-810 (in Chinese)
(王晓燕, 陈晋川, 杜小勇, 等. 并行 OLTP 系统中增量数据的自动分片技术文献[J]. 计算机科学与探索, 2013, 7(9): 800-810)

[21] Wang X, Fan X, Chen J, et al. Automatic data distribution in large-scale OLTP applications [J]. International Journal of Database Theory and Application, 2014,7(4): 37-46

[22] Lin W, Zhao H, Liu K. A game theoretic framework for incentive-based peer-to-peer live-streaming social networks [C] //Proc of 2008 IEEE Int Conf on Acoustics, Speech, and Signal Processing. Piscataway, NJ: IEEE, 2008: 2141-2144

[23] Cheng Pu, Chu Yanping, Du Ying. Analysis of cooperation model for P2P live streaming in game theoretic framework [J]. Journal of Computer Applications, 2011, 3(5): 1159-1161, 1188 (in Chinese)
(程普, 楚艳萍, 杜莹. 博弈论框架下 P2P 实时流的合作模型文献[J]. 计算机应用, 2011, 3(5): 1159-1161,1188)

[24] TPC. TPC benchmark C standard specification revision 5.11 [EB/OL]. San Francisco, CA: Transaction Processing Performance Council, 2010 [2014-04-23]. http://www.tpc.org/tpcc

[25] Özsu T, Valduriez P. Principles of Distributed Database Systems [M]. 3rd ed. Berlin: Springer, 2011

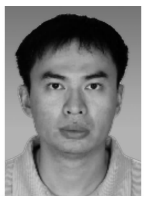
[26] Li S, Wong M. Data allocation in scalable distributed database systems based on time series forecasting [C] //Proc of 2013 IEEE Int Congress on Big Data. Piscataway, NJ: IEEE, 2013: 17-24

[27] Wang T, Lin Z, Yang B, et al. MBA: A market-based approach to data allocation and dynamic migration for cloud database [J]. Science China Information Sciences, 2012, 55 (9): 1935-1948

[28] Lin H, Keller M. The gradient model load balancing method [J]. IEEE Trans on Software Engineering, 1987, 13(1): 32-38

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

Wang Xiaoyan, born in 1981. PhD candidate at Renmin University of China. Student member of China Computer Federation. Her research interests include big data management and analysis.



Chen Jinchuan, born in 1978. Associate professor at Renmin University of China. Member of China Computer Federation. His research interests include uncertainty data management, big data management, etc.

Guo Xiaoyan, born in 1985. Senior researcher in EMC Labs China. His research interests include big data analytics, distributed computing, etc (xiaoyan.guo@emc.com).

Du Xiaoyong, born in 1963. Professor and PhD supervisor at Renmin University of China. Senior member of China Computer Federation. His research interests include big data management and analysis, etc.