软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2013,24(8):1786−1803 [doi: 10.3724/SP.J.1001.2013.04416]

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

支持大数据管理的 NoSQL 系统研究综述*

申德荣, 于 戈, 王习特, 聂铁铮, 寇 月

(东北大学 信息科学与工程学院,辽宁 沈阳 110004)

通讯作者: 申德荣, E-mail: shenderong@ise.neu.edu.cn, http://www.neu.edu.cn

摘 要: 针对大数据管理的新需求,呈现出了许多面向特定应用的 NoSQL 数据库系统.针对基于 key-value 数据模型的 NoSQL 数据库的相关研究进行综述.首先,介绍了大数据的特点以及支持大数据管理系统面临的关键技术问题;然后,介绍了相关前沿研究和研究挑战,其中典型的包括系统体系结构、数据模型、访问方式、索引技术、事务特性、系统弹性、动态负载均衡、副本策略、数据一致性策略、基于 flash 的多级缓存机制、基于 MapReduce 的数据处理策略和新一代数据管理系统等;最后给出了研究展望.

关键词: NoSQL;key-value 存储;大数据管理

中图法分类号: TP311 **文献标识码**: A

中文引用格式: 申德荣,于戈,王习特,聂铁铮,寇月.支持大数据管理的 NoSQL 系统研究综述.软件学报,2013,24(8):1786−1803. http://www.jos.org.cn/1000-9825/4416.htm

英文引用格式: Shen DR, Yu G, Wang XT, Nie TZ, Kou Y. Survey on NoSQL for management of big data. Ruan Jian Xue Bao/ Journal of Software, 2013,24(8):1786−1803 (in Chinese). http://www.jos.org.cn/1000-9825/4416.htm

Survey on NoSQL for Management of Big Data

SHEN De-Rong, YU Ge, WANG Xi-Te, NIE Tie-Zheng, KOU Yue

(College of Information Science and Engineering, Northeastern University, Shenyang 110004, China)

Corresponding author: SHEN De-Rong, E-mail: shenderong@ise.neu.edu.cn, http://www.neu.edu.cn

Abstract: Many specific application oriented NoSQL database systems are developed for satisfying the new requirement of big data management. This paper surveys researches on typical NoSQL database based on key-value data model. First, the characteristics of big data, and the key technique issues supporting big data management are introduced. Then frontier efforts and research challenges are given, including system architecture, data model, access mode, index, transaction, system elasticity, load balance, replica strategy, data consistency, flash cache, MapReduce based data process and new generation data management system etc. Finally, research prospects are given.

Key words: NoSQL; key-value storage; big data management

大数据[1,2]通常被认为是 PB(1 024 terabytes)或 EB(1EB=100 万 TB)或更高数量级的数据,包括结构化的、半结构化的和非结构化的数据,其规模或复杂程度超出了常用传统数据库和软件技术所能管理和处理的数据集范围.随着技术的发展,大数据广泛存在,如企业数据、统计数据、科学数据、医疗数据、互联网数据、移动数据、物联网数据,等等,并且各行各业都可得益于大数据的应用[3,4].按其应用类型,可将大数据分为海量交易数据(企业 OLTP 应用)、海量交互数据(社网、传感器、GPS、Web 信息)和海量处理数据(企业 OLAP 应用)这 3 类[5].海量交易数据的应用特点是多为简单的读写操作,访问频繁,数据增长快,一次交易的数据量不大,但要求

* 基金项目: 国家重点基础研究发展计划(973)(2012CB316201); 国家自然科学基金(61033007, 61003060)
收稿时间: 2012-03-28; 修改时间: 2012-10-19; 定稿时间: 2013-03-29; jos 在线出版时间: 2013-05-23
CNKI 网络优先出版: 2013-05-23 15:17, http://www.cnki.net/kcms/detail/11.2560.TP.20130523.1517.001.html

支持事务特性.其数据的特点是完整性好、实效性强,有强一致性要求.海量处理数据的应用特点是面向海量数据分析,操作复杂,往往涉及多次迭代完成,追求数据分析的高效率,但不要求支持事务特性,典型的是采用并行与分布处理框架实现.其数据的典型特点是同构性(如关系数据或文本数据或列模式数据)和较好的稳定性(不存在频繁的写操作).海量交互数据的应用特点是实时交互性强,但不要求支持事务特性.其数据的典型特点是结构异构、不完备、数据增长快,不要求具有强一致性.

大数据带来了大机遇[6],同时也为有效管理和利用大数据提出了挑战.尽管不同种类的海量数据存在一定的差异,但总体而言,支持海量数据管理的系统应具有如下特性[7]:高可扩展性(满足数据量增长的需要)、高性能(满足数据读写的实时性和查询处理的高性能)、容错性(保证分布系统的可用性)、可伸缩性(按需分配资源)和尽可能低的运营成本等.然而,由于传统的关系数据库所固有的局限性,如峰值性能、伸缩性、容错性、可扩展性差等特性,很难满足海量数据的柔性管理需求.为此,提出了云环境下面向海量数据管理的新模式,如采用 NoSQL 存储系统[8,9]或可扩展的数据管理系统[10−12](或称关系云系统)支持海量数据的存储和柔性管理.目前,它们是云环境下所采用的典型的云存储系统.

NoSQL 是指那些非关系型的、分布式的、不保证遵循 ACID 原则的数据存储系统[13],并分为 key-value 存储、文档数据库和图数据库这 3 类[14].其中,key-value 存储备受关注,已成为 NoSQL 的代名词.典型的 NoSQL 产品有 Google 的 BigTable[15]、基于 Hadoop HDFS[16]的 HBase[17]、Amazon 的 Dynamo[18]、Apache 的 Cassandra[19]、Tokyo Cabinet[20]、CouchDB[21]、MongoDB[22]和 Redis[23]等.针对 key-value 数据存储的细微不同,研究者又进一步将 key-value 存储细分为 key-document 存储(MongoDB,CouchDB)、key-column 存储(Cassandra,Voldemort, Hbase)和 key-value 存储(Redis,Tokyo Cabinet).

NoSQL 典型地遵循 CAP 理论[24]和 BASE 原则[25].CAP 理论可简单描述为:一个分布式系统不能同时满足一致性(consistency)、可用性(availability)和分区容错性(partition tolerance)这 3 个需求,最多只能同时满足两个.因此,大部分 key-value 数据库系统都会根据自己的设计目的进行相应的选择,如 Cassandra,Dynamo 满足 AP;BigTable,MongoDB 满足 CP;而关系数据库,如 Mysql 和 Postgres 满足 AC.BASE 即 Basically Available(基本可用)、Soft state(柔性状态)和 Eventually consistent(最终一致)的缩写.Basically Available 是指可以容忍系统的短期不可用,并不强调全天候服务;Soft state 是指状态可以有一段时间不同步,存在异步的情况;Eventually consistent 是指最终数据一致,而不是严格的时时一致.因此,目前 NoSQL 数据库大多是针对其应用场景的特点,遵循 BASE 设计原则,更加强调读写效率、数据容量以及系统可扩展性.

在性能上,NoSQL 数据存储系统都具有传统关系数据库所不能满足的特性,是面向应用需求而提出的各具特色的产品.在设计上,它们都关注对数据高并发地读写和对海量数据的存储等,并具有很好的灵活性和性能[26].它们都支持自由的模式定义方式,可实现海量数据的快速访问,灵活的分布式体系结构支持横向可伸缩性和可用性,且对硬件的需求较低.

可扩展的数据管理系统(关系云)是侧重扩展数据库系统到云环境下,使关系云支持海量数据管理.典型的系统如微软的 SQL Azure[10]和 MIT 的 Relational Cloud[12]等.主要针对事务特性、系统弹性性能、多租户负载均衡技术等进行研究.

本文以 key-value 数据模型的 NoSQL 数据库系统相关研究为核心进行综述,同时也介绍了与 NoSQL 相关的关系云系统和 MapReduce 框架[27]的相关研究和研究挑战.下文将不区分 NoSQL 数据存储系统和 key-value 存储系统.

1 关键技术问题

尽管大多数 NoSQL 数据存储系统都已被部署于实际应用中,但归纳其研究现状,还有许多挑战性问题.

研究现状:1) 已有 key-value 数据库产品[15,17,19]大多是面向特定应用自治构建的,缺乏通用性;2) 已有产品支持的功能有限(不支持事务特性),导致其应用具有一定的局限性[28];3) 已有一些研究成果和改进的 NoSQL 数据存储系统[29−31],但它们都是针对不同应用需求而提出的相应解决方案,如支持组内事务特性、弹性事务等,

很少从全局考虑系统的通用性,也没有形成系列化的研究成果;4) 缺乏类似关系数据库所具有的强有力的理论(如 armstrong 公理系统)[32]、技术(如成熟的基于启发式的优化策略、两段封锁协议等)、标准规范(如 SQL 语言)的支持.

研究挑战:随着云计算、互联网等技术的发展,大数据广泛存在,同时也呈现出了许多云环境下的新型应用,如社网、移动服务、协作编辑等.这些新型应用对海量数据管理或称云数据管理系统也提出了新的需求,如事务的支持、系统的弹性等.同时,业界专家也指出,云计算时代海量数据管理系统的设计目标为可扩展性、弹性、容错性、自管理性和"强一致性".目前,已有系统通过支持可随意增减节点来满足可扩展性;通过副本策略保证系统的容错性;基于监测的状态消息协调实现系统的自管理性."弹性"的目标是满足 Pay-per-use 模型,以提高系统资源的利用率.该特性是已有典型 NoSQL 数据库系统所不完善的,但却是云系统应具有的典型特点;"强一致性"主要是新应用的需求.因此,为有效支持云环境下的海量数据管理,还存在许多挑战性问题[28].典型的关键问题如下:

- 海量数据分布存储与局部性

海量数据均匀分布存储在各节点上,典型的策略是采用 Hash 分布存储或连续存储.Hash 存储可实现均匀分布,但弱化了数据间的联系,不能很好地支持范围查询;连续存储支持范围查询但影响数据分布存储的均衡性.目前,简单的 key-value 数据模型为提升可扩展性,弱化了数据间的关联关系,这对于 key 间不具有关联性的事务是合适的.然而,产生的大数据不是孤立存在的,它们之间存在必然联系,如社网中某一主题的数据、同类产品的销售数据等.另外,为增值海量数据的价值,进行海量数据分析也是必然趋势.因此,在弱化数据关系实现可扩展性的同时,需要考虑数据的局部性,以有效提升海量数据查询与分析的 I/O 性能.为此,数据模型应兼顾可扩展性和数据间的关联性,基于此研究相应的索引和查询以及相应的支持理论.

- 分布式事务特性

为了支持分布环境下的事务特性,典型的策略是采用两段提交协议实现.然而,集群环境下节点的动态性很难保证支持事务的节点的及时提交,导致事务代价大.为此,需要研究如何避免采用 2PC[33]来支持事务特性的 key-value 数据库.典型的研究是将事务操作数据分布到一个节点上或动态重组到一个节点上来避开分布环境,但这种策略局限性较大.为此,需要考虑如何避开 2PC 协议,即回避因单点失败导致整个事务废弃的处理过程,如采用乐观的并发控制方法,并结合副本数据支持云环境下的事务特性.

- 负载均衡的自适应性

在面向海量数据管理的云环境下,用户访问量、数据变化量或执行任务量都是事先无法精确预见的.随着数据量的增加或应用范围的扩大或应用用户的增加,都可能出现存储热点或应用热点.因此,弹性动态平衡是云环境下分布系统的典型特点.弹性(elasticity),就是随着负载的增加和减少可动态调节,并最小化操作代价.为此,需要有自适应协调方法,保证系统的弹性性能,有效地提高系统资源利用率.目前,典型的研究是:1) 侧重多租户的事务迁移达到动态平衡;2) 事先选择均衡的执行方案,没有考虑自适应性.

- 灵活支持复杂查询

若按需满足不同用户的查询需求,云环境必须具有更大的灵活性和柔性.目前,已有 key-value 数据库均支持简单查询,而将复杂查询交由应用层完成.MapReduce 是被广泛接受的分布式处理框架,用于实现海量数据的并行处理,通常,应用层基于该框架定制相应的查询视图.尽管 MapReduce 为海量数据处理提供了灵活的并行处理框架,但如何最优地实现各种复杂查询和数据分析还需要深入研究.希望能提供一种更为灵活的、可优化的复杂查询定义模型,可按需满足不同用户的查询和数据分析需求.

- 灵活的副本一致性策略

在云环境下,典型的是基于副本策略提高系统的可用性,同时也带来了维护副本一致性的代价.已有系统采用最终一致性策略[34,35]或强一致性策略[36]实现副本同步.然而,无论何种策略都具有一定的局限性,限制了 NoSQL 数据存储系统的应用范围.如强一致性适用于同时访问数据量不是很大的 OLTP 应用或在线交易系统,而最终一致性适用于不要求具有实时一致需求的 Web 查询系统.因此,需要提供一个灵活的、自适应的副本一

致性策略模型,按需配置副本一致性策略,并最小代价地满足各种应用需求.

针对上述关键问题,目前研究者们侧重研究的内容主要有:针对事务和系统弹性的研究有支持多关键字查询的事务语义研究[10,29,30]、弹性事务的研究[31,37−43]、负载均衡策略的研究[10,44−47]、自适应副本策略的研究[48−51]等;提升数据访问效率的研究有基于 flash 扩展缓存的研究[52−56];支持新应用需求的新一代数据存储系统的研究[29−31];还有采用 MapReduce 框架支持海量数据分析的研究[57−59]和 NoSQL 数据库与 MapReduce 优势结合的研究[60,61]等.下面简述相关研究和研究挑战.

2 系统体系结构

尽管目前流行的 NoSQL 数据存储系统的设计与实现方式各有不同,但是总结起来大体上有两种架构:master-slave 结构和 P2P 环形结构.两者各具特色.

2.1 Master-Slave结构与P2P环形结构

在采用 master-slave 结构[15,17]的系统中,master 节点负责管理整个系统,监视 slave 节点的运行状态,同时为其下的每一个 slave 节点分配存储的范围,是查询和写入的入口.master 节点一般全局只有 1 个,该节点的状态将严重影响整个系统的性能,当 master 节点宕机时,会引起整个系统的瘫痪.实践中,经常设置多个副本 master 节点,通过联机热备的方式提高系统的容错性.slave 节点是数据存储节点,通常也维护一张本地数据的索引表.系统通过添加 slave 节点来实现系统的水平扩展.在 master-slave 框架下,master 节点一直处于监听状态,而 slave 节点之间尽量避免直接通信以减少通信代价.在运行过程中,salve 节点不断地向 master 节点报告自身的健康状况和负载情况,当某个节点宕机或负载过高时,由 master 节点统一调度,或者将此节点的数据重新分摊给其他节点,或者通过加入新节点的方式来调节.BigTable,Hbase 是典型的 master-slave 结构的 key-value 存储系统.

在 P2P 环形结构[18,19]中,系统节点通过分布式哈希算法在逻辑上组成一个环形结构,其中的每个 node 节点不但存储数据,而且管理自己负责的区域.P2P 环形结构没有 master 节点,可以灵活地添加节点来实现系统扩充,节点加入时只需与相邻的节点进行数据交换,不会给整个系统带来较大的性能抖动.P2P 环形结构没有中心点,每个节点必须向全局广播自己的状态信息.例如,目前流行的采用 P2P 环形结构的 Cassandra 和 Dynamo 系统采用 Gossip 机制[24]来进行高效的消息同步.

可见,NoSQL 数据存储系统的两种流行的体系结构的框架存在很大的不同,各自所需维护网络运行的协议差别也很大,二者典型的特点如下:

1) Master-Slave 结构的系统设计简单,可控性好,但 master 中心节点易成为瓶颈;P2P 环形结构的系统无中心节点,自协调性好,扩展方便,但可控性较差,且系统设计比 master-slave 结构的系统要复杂.
2) Master-Slave 结构的系统需要维护 master 服务节点,由 master 节点维护其管理的 slave 节点,维护简单、方便;P2P 环形结构的系统自协调维护网络,扩展方便,可扩展性好.
3) Master-Slave 结构的系统将 master 节点和 slave 节点的功能分开,可减轻节点的功能负载;P2P 环形结构的系统,各节点平等,没有起到功能分布的作用.
4) Master-Slave 结构的系统通常基于水平分区实现数据分布,方便支持范围查询;P2P 环形结构的系统适于基于 Hash 分布数据,负载均衡性好,但不利于支持范围查询.

2.2 两种体系结合的研究

由于 key-value 数据存储系统的两种体系结构差别很大,它们所采用的支持技术存在很大差别,导致了不同体系结构的系统所支持的功能的局限性.Cloudy[62]为用户提供了一个可配置采用 master-slave 或 DHT 体系结构的 Demo 系统,但两种体系结构结合的研究还很少.未来支持 key-value 数据存储系统的体系结构应结合 P2P 分布式结构和 master-slave 集中式结构两者的优势,如 Chord 和 master-slave 的结合、CAN 与 master-slave 的结合等,侧重研究面向组件的灵活可配置的体系结构,这样,可以灵活地结合两者的优势,并可以综合考虑数据存储的全局性和局部性.

3 数据存储的研究

Key-Value 数据库的目标是支持简单的查询操作,将复杂操作留给应用层实现.在 NoSQL 数据存储领域,为了提高存储能力和并发读写能力,采用了弱关系的数据模型,典型的是 key-value 数据模型.本节将介绍 key-value 数据模型、数据读写方式、索引机制、支持的查询操作以及研究挑战.

3.1 Key-Value数据模型

Key-Value 存储可细分为 key-value 型、key-document 型和 key-column 型.key-column 型是 key-value 键值对的典型扩充,也是目前业界推崇的数据模型.由于 key-document 型适于文档型数据,本部分不予介绍.

1) Key-Value 型

Key-Value 键值对数据模型实际上是一个映射,即 key 是查找每条数据地址的唯一关键字,value 是该数据实际存储的内容.例如键值对:(“20091234”,“张三”),其 key:“20091234”是该数据的唯一入口,而 value:“张三”是该数据实际存储的内容.Key-Value 数据模型典型的是采用哈希函数实现关键字到值的映射,查询时,基于 key 的 hash 值直接定位到数据所在的点,实现快速查询,并支持大数据量和高并发查询.

2) Key-Column 型

Key-Column 型数据模型主要来自 Google 的 BigTable.目前流行的开源项目 Hbase 和 Cassandra 也采用了该种模型.Column 型数据模型可以理解成一个多维度的映射,主要包含 column,row 和 columnfamily 等概念.图 1 描述了一个 column 型数据模型的实例.

Key	Columns		
	Name	Value	Timestamp
k_1	c_1	v_1	123456
	c_2	v_2	123456
k_2	c_1	v_3	123456
	c_2	v_4	123456

Fig.1 A columnfamily instance

图 1 一个 columnfamily 的实例

如图 1 所示,在 key-column 型数据模型中,column 是数据库中最小的存储单元,它是一个三元组,包括 name(如 c_1,c_2),value(如 v_1,v_2)和 timestamp(如 123456),即一个带有时间戳的 key-value 键值对.每一个 row 也是一个 key-value 对,对于任意一个 row,其 key 是该 row 下数据的唯一入口(如 k_1),value 是一个 column 的集合(如 column: c_1,c_2).Columnfamily 是一个包含了多个 row 的结构,相当于关系库中表的概念.

简单来说,key-column 型数据模型是通过多层的映射模拟了传统表的存储格式,实际上类似于 key-value 数据模型,需要通过 key 进行查找.因此,key-column 型数据模型是 key-value 数据模型的一种扩展.

3) 支持 key-value 数据模型的技术研究

数据模型是数据管理所关注的核心问题.Key-Value 数据模型因其简单以及具有灵活的可扩展性而广泛被云系统所采用.目前,已有一些 key-value 数据库产品都是面向特定应用构建的,支持的功能以及采用的关键技术都存在很大差别,并没有形成一套系统化的规范准则.因此,需要规范 key-value 数据模型及其支持理论,主要包括:1) 研究 key-value 数据模型的规范定义和所支持的基本操作;2) 研究面向应用设计 key-value 数据组织所遵循的准则,如代价最小化的 key-value 数据物理组织模型、代价最小化的数据可扩展的启发式准则,为数据最优组织提供遵循准则;3) 研究 key-value 数据对间的关联关系以及正确性验证规则,为数据组织的合理性和正确性提供一定的依据.

3.2 读写方式

分析已有 key-value 数据库,其读写方式可分为面向磁盘的读写方式和面向内存的读写方式两种.后者适合于不要求存储海量的数据但需要对特定的数据进行高速并发访问的场景.采用哪一种读写方式,通常由数据量的大小和对访问速度的要求决定的.本部分只介绍面向磁盘的读写方式.

1) 面向磁盘的读写方式

通常情况下,NoSQL 系统中都存储着海量的数据,且无法全部维持在内存中,所以一般都采用面向磁盘的读写方式,图 2 描述了 NoSQL 系统中采用的典型的面向磁盘读写的一般过程.

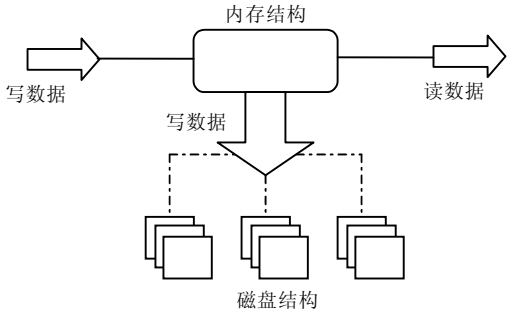


Fig.2 Disk-Oriented read and write process

图 2 面向磁盘的读写过程

如图 2 所示,通常,当写入数据时,数据首先会被写到一个内存结构中,系统返回写入成功.当内存中的数据达到指定大小或存放超过指定时限时,会被批量写入磁盘.当需要读取数据时,首先访问内存结构,如果未命中则需要访问磁盘上的实例化文件.当系统发生意外宕机时,内存结构中的数据将丢失,因此,一般采用日志的方式来帮助进行数据恢复.为了进一步提高写入效率和并发能力,许多系统都采用了 Append 的方式,即将修改和删除操作都追加写到文件末尾,而读数据时利用时间戳过滤掉旧信息,返回给用户最新版本的数据.因此,数据库需要进行定期的数据合并,将过期的冗余数据删除.

此外,在一些面向文档的 NoSQL 数据库中(例如 MongoDB),其主要采用内存文件映射的机制(MMAP)来实现对文档的读写操作,即把磁盘文件的一部分或全部内容直接映射到内存当中,避免了频繁的磁盘 IO,通过简单的指针来实现对文件的读写操作,极大地提高了读写效率.

2) 基于 flash 内存扩展缓存的研究

为了提高数据的读写速度,提出了基于 flash 内存扩展缓存的研究,以提高持久化的 key-value 存储系统的吞吐率,进而提高系统应用性能.由于 flash 内存在性能和费用上介于 DRAM 和 disk 之间,如目前 flash 内存比硬盘高 100 倍~1 000 倍的访问时间,比 DRAM 访问时间约低 100 倍;相反地,flash 内存的费用是 DRAM 的 1/10 左右,而比 disk 贵 20 倍左右,可见,flash 内存是自然的选择.为此,为了提高系统的数据处理性能,进行了相关应用 flash 内存[52−56]的 key-value 存储系统的研究,它们混合使用 RAM 和 flash 内存,将所有的 key-value 对存于 flash 内存中,并将少量的 key-value 对的元信息存在 RAM 中支持快速插入和查询.flash 内存的容量可远远大于 RAM,因此需要减少存在于 flash 内存中的 key-value 对所需要的 RAM 字节数.目前,这方面的研究侧重于如何利用最小 RAM 存储最多的 flash 中的 key-value 对以及恰当的多级存储策略,提供高吞吐率和低延迟的服务.

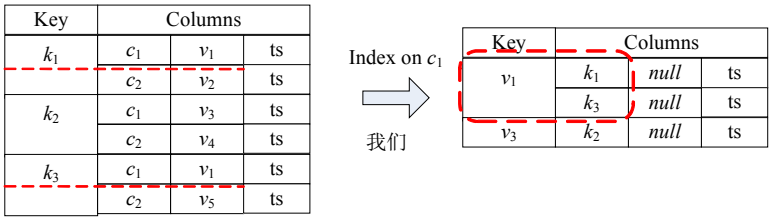
3.3 索引技术

目前,大多数云框架基于分布式文件系统(如 DFS),通常采用 key-value 存储模型存储数据,即云系统中的数据组织为 key-value 对.因此,当前的云系统(如 Google's GFS 和 Hadoop's HDFS)只支持 keyword 查询,即用户只能通过点查询满足用户查询需求.Key-Value 数据库典型的是以 key 索引为主,常见的有 hash 索引、B-tree 索引等.为了提供丰富的查询能力,一些 key-value 数据库还建有二级索引或称辅助索引(secondary index)[63],同时,为了提高对海量数据的查询效率,一些系统采用了 BloomFilter 技术.但已有的这些索引都是局部索引.

1) 二级索引或辅助索引

在 key-value 数据库中,数据的 key 是数据的检索入口.为了实现对值的查询,需要对值建立一个有效的索引,称为列值索引或二级索引或辅助索引.下面以 Cassandra 为实例,介绍二级索引的运作原理.

在基于 column 数据模型的 Cassandra 中,column.value 上的二级索引实际上是一个新的 columnfamily 结构.图 3(a)为原有的数据表 columnfamily cf_1,图 3(b)为新建的 columnfamily,即二级索引 cf_1-c_1.在 cf_1-c_1 中,新 key 是原数据表中 column 的 value 值,对应行下的 column.name 为原 columnfamily 中的 key,column value 为空.利用构建的辅助索引,可以实现对 column.value 的条件查询.例如,对于图 3 的 columnfamily:cf_1,若查询满足条件:$c_1.value=v_1$ 的行的数据内容,则其过程是:首先查找 cf_1 中 c_1 的索引 cf_1-c_1,在索引中按 $key=v_1$ 查找到该行数据,读取该行下每一列的列名,即 k_1,k_3;接下来在原数据表 cf_1 中按 $key=k_1$,$key=k_3$ 分别进行查询,得到 k_1,k_3 行下的数据.



(a) Columnfamily cf_1 (b) 二级索引 cf_1-c_1

Fig.3 Index logic structure sketch

图 3 索引的逻辑结构示意图

2) BloomFilter

在 key-value 数据库中经常会遇到如下情况:一个表由数十亿甚至更多行(每行对应一个 key-value 对)组成,这些数据被实例化到数千个磁盘文件当中.若建立一个统一的索引,则维护代价很大.通常,需要分别对一个或是一组实例化文件建立独立的索引,当检索一条数据时,首先需要快速地判断这条数据是不是在这个或是这组文件当中.key-value 数据库中普遍采用 BloomFilter 技术来解决这个问题.BloomFilter 是一种空间效率很高的随机数据结构,它利用位数组简洁地表示一个集合,能够快速地判断一个元素是否属于这个集合.应用 BloomFilter 的过滤示意如图 4 所示,设 BFV_1,BFV_2,…,BFV_n 是关键字 k_1,k_2,…,k_n 插入时基于 hash 函数生成的 BFV 数组,若查询关键字为 k,则应用 hash 映射数组,基于 BFV_i 确定 DataBlock 中是否有要读取的 k.若在,进一步查找,否则跳过该块.这样,可有效提高查找效率.

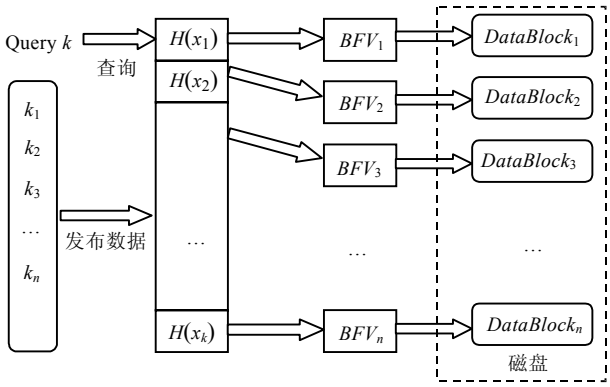


Fig.4 Filtering sketch using Bloom Filter

图 4 应用 Bloom Filter 过滤示意图

3) 分布索引研究

目前,大多已有 key-value 存储系统采用局部索引,其“全局索引”典型的是采用 Hash 直接定位数据所在的节点.目前,有关云环境下用于数据管理的索引结构的典型研究有支持多属性查询或范围查询或 K-NN 查询而建立的索引结构[64−66]和适用于集群结构的索引结构[67,68].

文献[64]建立了多维索引机制 RT-CAN.该索引集成了 CAN 路由协议和基于 R-tree 的索引模式来支持云系统中的多维查询处理,其思想是:将服务器组织为 CAN 覆盖网络,实现全局索引分布存储;R-tree 是构建的局部索引,全局索引由 R-tree 点组成,可有效减少全局索引的大小和全局索引的维护代价.文献[65]针对云计算环境下的大规模数据查询处理,提出了二级索引技术 CG-index.它首先在一个数据分片上建立本地 B+tree 形成索引分片,然后将计算节点组织成 Overlay 结构,接下来,基于 Overlay 的路由协议把各计算节点上 B+tree 分片发布到 Overlay 上,建立全局索引 CG-index.文献[66]是在分布式 KD-tree 基础上提出多维索引 MIDAS,用于支持多维查询、范围查询和 k 最近邻查询等应用.系统中一个节点对应多个虚拟节点,一个虚拟节点对应 KD-tree 的一个叶子点,用于存储和索引存储在叶子矩形区的元组.HyperDex[67]使分布式 key-value 存储系统支持基于辅助属性的查询.核心思想是提出了超空间 Hash 概念,将多属性映射为有一个多维超空间,使其支持关键字、辅助属性和范围查询.分片位图索引[68]的核心思想是在字段值上进行全局排序和位图索引局部存储.全局排序能够为各数据节点提供一定的全局信息,即局部数据在全局值域中的分布情况;而位图索引的局部存储使得索引结构局部化,从而使不同数据节点上的检索任务充分独立,以便并发执行.

文献[69]提出了一个具有可扩展性的分布式的 B-tree 索引结构.它将分布式 B-tree 的点分散存在多个服务器上.其典型特点如下:1) 可支持在一个或多个 B-tree 上执行多个操作,即实现事务操作,简化了由应用层实现事务特性;2) 树节点可在线迁移,可将树节点透明地从一个服务器迁移到其他服务器或新加入的服务器.文献[70]将 R-tree 和 KD-tree 结合起来组织数据记录,提出了用于云数据管理的多维索引 EMINC.该索引可以提供快速的查询处理和有效的索引维护服务,Master 节点维护 R-tree,slave 节点维护 KD-tree,R-tree 的每个叶节点包括点 cube 和指向相应 slave 点的指针.

在云环境下,为了提高海量数据的查询性能,多种索引结构共存是必然趋势.其中,辅助索引和 BloomFilter 过滤技术已被业界广泛认可,而构建全局索引技术已成为云环境下高效管理数据的研究热点.

3.4 支持的查询

目前,各 key-value 数据库都支持典型的 key-value 简单查询功能,这里将简单 key-value 查询称为简单查询,而其他查询统称为复杂查询.key-value 数据库除了支持简单查询以外,对复杂查询的支持通常分为两种方式:动态查询和视图查询.所谓动态查询是指类似于关系数据库的查询,即直接在已有数据上进行任何限定条件的查询,为了提高查询效率,可以设计合理的索引机制.视图查询是指预先为查询条件创建相应的视图(如基于 MapReduce 函数定义视图),然后在视图上进行相应的查询.

目前,MongoDB 基于其提供的类似于关系数据库的索引机制实现复杂查询;Tokyo Cabinet/Tokyo Tyrant 支持类似于单表的基本查询;Hbase,Hypertable,CouchDB 支持基于 MapReduce 的并行机制实现复杂查询.Cassandra 已开发了与 Hadoop 集成代码,可以采用 MapReduce 机制与 Cassandra 存储的数据进行交互.

可见,key-value 数据存储系统都支持简单的查询功能以及面向 key 的索引,通常把复杂查询留给应用层实现,主要采用 MapReduce 框架实现各种复杂查询.然而,该方法完全是基于 MapReduce 模型组件定义相应的查询视图,导致部分复杂查询的视图定义复杂和查询处理代价大.

4 支持事务的研究

推动在云环境中支持事务特性的动机[37]有:1) 已有商用的 NoSQL 系统[15,18,71]不支持事务特性;2) 新应用需求事务的支持,如 Web2.0 应用、社网、协作编辑[72,48]等;3) 传统数据库应用希望采用云平台框架,需要支持事务.为此,提出了针对云存储系统(NoSQL 和关系云)支持事务的研究.

已有 key-value 云存储系统的典型特点是容错性、可扩展性和有效性.依据 CAP 理论,分布式系统只能满足 CAP 中的两个特性,为此,呈现出了许多摒弃事务特性的分布式系统,如保证 AP 特性的 key-value 数据库系统(典型的有 Dynamo,Cassandra 等).这些系统都通过松散一致性来保证其有效性,主要体现为支持单数据项或数据项集合的原子性或最终一致性.当然,弱一致性对某些应用是可行的,如 Web 关键字搜索、库存查询等.然而,许多应用在应用弱一致性的云存储系统时却带来许多不便,如社网、Web 2.0 应用、在线拍卖、合作编辑等应用,

需要由应用层保证一致性.

我们知道,传统的分布式数据库系统为保证事务特性而不支持可扩展性和有效性,主要原因是采用两段提交协议保证分布式数据库的事务特性时,只要有一点失败,将废弃整个事务.而key-value数据库的可扩展性如数据分布存储、节点动态加入、数据量急剧增加等,可能会使一个事务涉及较多的分布节点,且聚簇环境下允许节点失效,导致惩罚代价太大.因此,在云环境下实现事务的思想是尽量避免采用两段提交协议.

为了支持 ACID 事务特性,在 NoSQL 的云存储系统中,典型的有以下 3 种研究[37]:1) 支持弱一致性的存储系统的实现,将一致性检查交给应用层处理.这种方法增加了开发者的负担,不是一种有效的方法[11].2) 扩展单key 事务以支持多 key[10,29,30].通常是为了避免采用 2PC,将事务限制在一定范围内的多 key,即组内多 key 特性,但不支持不同组中多 key 事务.3) 基于 NoSQL 支持弹性事务的研究[42,43].在扩展的关系云系统中,典型的研究有:1) 在支持随负载可弹性扩展的数据库或数据存储系统[31,38−40]的基础上,支持有限的可扩展性和事务语义;2) 面向 OLTP 可扩展的事务支持[37],即基于分离事务组件和数据组件思想,根据应用负载和事务负载配置事务组件(TC)和数据管理组件(DC).

目前,为了支持相应的场景下的事务特性,通常采用如下的典型策略来实现:

1) 应用层保证事务一致性

已有key-value存储系统只提供单key访问保证,因此,应用key-value存储系统的协作应用需要通过应用层工作流处理多 key 访问和原子性与一致性的正确保证.也就是说,key-value 存储系统只支持简化的一致性保证和单 key 数据访问粒度,而将大工作负载留给了应用程序员,由应用层处理节点故障、并发控制和数据的不一致来保证系统的正确性.这种方法需要多次与服务器交互才能保证一致性[9],影响系统性能,因此,该系统适于需要强一致性较少的应用.但当需要强一致性要求的操作时,影响是不可避免的.

2) 本地事务支持

采用回避分布式数据库系统中的 2PC 协议的思想,将事务操作限制于一个节点上.相关研究主要是将事务内的多关键字组织在一个节点上,典型的 key-value 存储系统有 Google Megastore[29],典型的关系云有微软的SQL Azure[10,73]和 ElasTras[39,40].Google Megastore 和微软的 SQL Azure 支持多记录事务,但它们要求记录以某种方式共存.Megastore 将数据划分为多个实体组,在每一个数据分区内基于传统数据库的 ACID 思想支持ACID 特性,分区间的数据副本的一致性应用改进的 Paxos[74]实现同步复制.而 SQL Azure 需要将数据大小限制在一个节点上.ElasTras 根据事务负载可支持弹性事务,但事务语义限制在一个分区内支持小事务(minitransaction)[75].

3) 有限范围内的事务支持

针对已有key-value数据库系统如Dynamo,PNUTS,HBase和Cassandra都不支持多关键字查询的事务语义,不能完全满足新一代 Web2.0 的需求,如网络游戏、社网、协作应用等,ecStore[31],G-Store[30]等分别针对相应的应用提出了相应的解决方案.ecStore 采用 3 层结构(存储层、副本层、事务层),采用多版本(multi-version)和乐观的并发控制方法相结合的模式实现隔离和一致性保证.例如,应用多版本数据支持只读事务,如 OLAP 业务;应用乐观并发控制方法保证系统不被修改事务封锁,如 OLTP 应用.但事务需要知道数据节点上所有数据部署的知识.G-Store 基于 key-value 存储底层提出了组抽象概念,定义了一组 key 的关联关系,按需基于组粒度实现事务访问,基于 key 组协议协调控制组中的 key,并有效地访问这组 key.G-store 支持动态分组内的事务,但不支持跨组数据的事务特性.

4) 弹性的事务支持

典型的研究是关系云 Deuteronomy[37]和 CloudTPS[41],以及基于 NoSQL 支持弹性事务的 Partiqle[42]和TIRAMOLA[43].Deuteronomy 将数据库存储引擎功能分为事务组件(TC)和数据组件(DC).TC 提供逻辑上的并发控制和 undo/redo 恢复,不需要知道数据的物理位置;DC 负责缓存数据,并知道数据的物理组织,包括访问方法、面向记录的原子操作接口,但不必知道事务特性.其主要特性是支持跨多 DC 的事务特性,由 TC 负责事务的提交和保证 DC 上数据的修改.CloudTPS 通过分裂事务管理器为多个子事务管理器,保证每个子事务管理器访问单

数据项,即采用两级的事务管理器.CloudTPS 只支持短事务并只访问组内数据的事务.

针对 key-value 存储系统支持的查询和数据管理功能有限,Partiqle 提出了在 key-value 存储系统上构建弹性的 SQL 引擎.其核心思想是提出了事务类(transaction classes)概念,将一个 SQL 查询语句所需要访问的数据组成一个事务类,事务类间由访问路径隔离,而不是由记录隔离.文献[43]针对已有 NoSQL 数据库的弹性特性需要用户参与实现的特点,结合云管理平台提出了一种可自动实现 NoSQL 数据库的弹性特性的框架 TIRAMOLA.该框架包括监控模块(monitoring module)、云管理模块(cloud management)、集群协调模块(cluster coordinator)和决策支持模块(decision making).

5) 面向分区数据支持分布式事务的研究

Calvin[76]是在分布式数据存储系统上构建的一个事务调度和数据副本层,可将非支持事务的存储系统转换为接近于线性的、可扩展的非共享的数据库系统,且支持系统的高有效性、数据的强一致性和事务特性.其核心思想是将事务中各操作按全局定义其调度序列,尽量避免单点失败和分布式事务的冲突操作,一旦出现单点失败,则由副本所在节点代替执行.可以说,Calvin 是依赖于构建的中间件层支持可跨数据区的事务特性,并且不影响系统的事务吞吐率.

目前,相关事务研究的局限性体现在:将一致性检查交给应用层处理的方法增加了开发者的负担,显然不是一种有效的方法;本地事务只限于支持一个区内的事务,无法实现跨区事务,因此只限于相关数据存储于一个分区上的事务应用,局限性较大;G-Store 虽然支持动态分组,但分组代价大;关系云主要针对事务处理的弹性进行研究,如 CloudTPS 采用两级的事务管理器,保证子事务管理器访问单数据项,但全局上还是采用两阶段提交协议;Deuteronomy 虽然可以灵活配置 TC 和 DC,并通过分离可灵活扩展事务,但事务管理器的实现还是采用类似于两阶段提交的两级实现,导致效率不高.Partiqle 和 TIRAMOLA 也是从上层或称中间件层实现弹性事务特性.

5 自适应的动态负载平衡技术的研究

随着数据量的增加或应用范围的扩大或应用用户的增加,都可能出现存储热点或应用热点,而且是不可预测的.弹性动态平衡是云环境下分布系统的典型特点.弹性(elasticity)就是随着负载的增加和减少可动态调节,并最小化操作代价.

通常,负载可分为用户负载、事务负载、操作负载和数据负载等.动态负载均衡是分布式系统中普遍关注的热点问题.针对业务侧重点的不同,采用的动态负载解决方案也存在着一定的差别.目前,已有相关研究主要分为如下两个部分:1) 多租户负载动态迁移技术的研究[12,46,47].目前,集中数据库或 key-value 数据存储系统主要采用重量级的技术实现数据迁移;而对于 OLTP 业务,由于需要访问的数据量不大而侧重于多租户的业务处理性能的研究,如事务特性和响应时间特性.2) 面向查询处理的负载均衡技术研究[12,44,45],强调均匀处理负载和数据负载的并行执行,以最小化查询处理响应时间.

1) 面向多租户的动态迁移技术

传统企业数据库具有静态特性,并不关注弹性和动态迁移(live migration)[77].然而,弹性负载均衡是云环境下大数据管理系统的典型特点,要求低代价实现用户在主机间迁移.目前,典型的研究是面向多租户的动态迁移技术的研究,key-value 存储[17−19]和关系云[10,12]是支持多租户动态迁移技术的典型的云存储系统.

key-value 存储如 Bigtable,PNUTS,Dynamo 等已被用于多租户应用部署,并且大多数 key-value 存储系统支持系统容错或负载均衡下的数据迁移.它们在集群环境上提供统一的名字空间,多个租户存储其数据库于一个共享的名字空间中,由数据存储系统管理多租户的数据存放与协同操作.key-value 存储系统大多采用重量级的(stop-and-copy)方法实现静态迁移,主要过程是采用重量级技术如停掉部分数据库,把它们的状态信息迁移到新节点上,然后重启动.像这样停掉再迁移技术或通过简单的优化会导致事务中断的高代价的性能惩罚(事务延迟和吞吐率降低(因为存在事务重启动的冷缓存问题)).

目前,针对关系云的动态迁移技术研究较多.为使数据库系统扩展为适应于云环境下的数据管理系统[11,12,37],其典型特点是采用动态迁移技术[78,79]实现轻量级弹性特性.其核心思想是:依赖共享的持续的存储概

要,采用混合 stop-and-copy,pull 和 push 方法在原点和目的点间迁移内存页和处理状态,但长时间的 stop-and-copy 方法会导致系统长时间不可用.为此,进一步提出了迭代状态复制方法(iterative state replication,简称 ISR)[46].其思想是:建立检查点并迭代地拷贝,即目的点装载检查点,而原节点维护不同的变化,并迭代地拷贝,直到需要传输的变化量足够小或达到最大的迭代次数,再执行最后的 stop 和 copy.但在最后的 stop 阶段,也会出现租户数据库不可用的情况.Albatross[46]是最早面向共享存储数据库云而提出的应用动态数据迁移(live data migration)技术的轻量级弹性框架.其思想是最小操作代价地实现 OLTP 类型功能的动态负载均衡,通过迁移数据库缓存和事务状态,保证最小影响事务的执行.Albatross 因共享存储不需要迁移持久映像,它不仅强调最小化服务中断,也侧重在迁移过程中因复制数据库缓存而影响的事务延迟.Zephyr[47]是支持非共享的数据库弹性云平台,侧重数据库层的动态迁移问题.在 Zephyr 中,针对节点都具有本地磁盘,需要迁移持久映像(persistent image),侧重于最小化服务中断.其思想是:采用同步的双模式(dual mode)方式(源和目标节点同时执行事务),允许原点和目的点同时执行一个租户的事务,达到最小化服务中断.主要策略为:1) 迁移开始,传输元数据到目的点,目的点即可以开始执行新事务;2) Read/write 访问的数据页也分为两部分,原点拥有所有的数据页,目的点拥有事务按需访问的数据页;3) 索引结构直接复制,迁移过程中不可改变.Curino 等人的方法[12]也支持面向非共享体系结构的数据库系统的动态迁移,基于类似 cache 的方法,建议在目的点启动事务,并由目的点按需取页.

2) 面向查询处理的负载均衡技术

在已有有关分布环境下的分布式查询处理[80−84]研究中,都假定最初已实现了最优分区,并在此条件下考虑如何最优实现查询处理.实际上,在执行过程中可能需要大量的数据传输操作,导致系统性能下降.已有 key-value 数据库采用的数据分布策略比较简单,如 Cassandra 依赖 hash 函数、Hbase 依据各节点的数据量实现.它们实现简单,且有一定的局限性,如没有面向查询处理考虑数据偏斜问题和查询热点问题等.在 Curino 等人[12]提出的关系云中,针对 OLTP 负载提出 Schism 分区系统[85].其主要思想是:将一个数据库和负载表示为图(元组或元组组表示点,事务表示边,连接事务边的两个点表示该事务涉及的元组),并应用图分区算法[86]找到平衡分区,最小化分割边事务.该方法侧重于 OLTP 事务的细粒度的最优划分,适于短事务且事务中涉及数据量少的情况,具有接近线性的弹性可扩展性.文献[45]面向复杂的长事务并访问大量数据的环境,典型的如数据仓库系统.其思想是:利用并行优化器的代价模型以及内部数据结构(不包括实例化数据)来找到最好的可能分区配置,可采用遗传搜索(genetic search)、模拟退火(simulated annealing)、爬山搜索(hill-climbing search)或几种方法结合的算法找到最优的分区配置.Ghandeharizadeh 等人[87]面向查询分析的 hash 分区和 range 分区提出了混合-范围分区策略.其思想是:将长查询事务划分,并在服务节点上并行执行,而本地化小范围查询.该方法只支持单表的范围查询,没有涉及副本.文献[44]讨论如何在 MapReduce 框架下将 Join 数据单元均衡分布到 reduce 上,实现最小化时间处理代价.MRShare[57]为了提高查询效率,将一批查询的任务进行重新合并而形成新的组.Gufler 等人[88]面向具有偏斜的科学数据的动态平衡进行研究.通过聚集由 mapper 检测的局部数据来近似评估全局数据分布,进而恰当地平衡分布任务的执行,达到提高数据处理的目的.SkewTune[89]的思想是:当集群中一节点空闲时,SkewTune 预测一个任务的最大期望处理时间,并提前对输入数据进行重分区,以便有效地利用集群众节点资源,同时保证输入数据的顺序,可方便实现输出结果的重构.

当前,大多数已有动态迁移技术重点面向关系云数据管理系统的研究,分别面向数据迁移、事务迁移和两者结合的研究.然而,针对 key-value 数据库的动态迁移技术的研究很少.基于已有相关工作,需要研究适合于 key-value 数据库的动态迁移技术.已有面向查询处理负载均衡技术的研究成果典型的是将数据资源均匀分布到各处理节点上(如 Hash 分布方法和考虑数据偏斜的数据分布方法)以及将处理任务均衡分布到处理节点上,并且大多独立地针对某一侧面进行研究,没有考虑相互影响的作用,也没有利用副本资源的作用.

6 副本管理研究

副本策略是分布式数据管理系统的典型特点.云环境下存在着大量为支持系统的可扩展性和可用性的副本,并分布存储于不同的节点上.在传统的分布式数据库系统中,通常是在设计阶段依据数据特点和应用特点确

定副本分布策略,目的是增加局部处理能力、提高系统可靠性和系统性能.但副本增加了其维护代价,并且在特定的应用需求下,要达到完全全局一致性是不可行的.本节主要介绍自适应的副本策略及其一致性维护策略.

6.1 自适应副本策略研究

云环境下数据管理系统中的数据副本策略与已有传统的分布式数据库系统中的副本策略目标一致,但策略不完全相同.云环境下数据管理系统的集群环境以及易扩展性需求弱化了数据间联系和特定的应用需求,其数据副本分布策略并不完全依据数据特点和应用特点,而是在有助于提高系统性能的基础上,更侧重于系统的可靠性,目前有存储于数据中心的副本分布策略、机架感知的副本分布策略和非机架感知的副本分布策略.基于相应的副本分布策略,再依据统一可配置的副本协议实现副本分布存储.在已有 key-value 存储中,副本迁移主要针对预先配置进行数据迁移,如当节点数据量达到阈值时进行迁移,而没有涉及动态自适应迁移过程.目前,只有 epiC[90]系统提出了根据访问负载动态构建副本的策略.epiC 是面向海量数据分析(OLAP)提供服务的支持平台.它采用自适应负载的副本策略.其思想是,除了主副本以外,还存在两类副本:从副本和辅助副本.对于频繁访问的主副本或从副本,需要再建立辅助副本,以解决热点查询的负载均衡问题.

虽然副本策略是分布式数据管理系统的典型特点,但目前主要还是利用已有副本来提高系统的可用性,其作用有限.为此,需要针对副本对云环境下的弹性、动态负载均衡和提高系统性能等方面的作用进行深入研究,平衡副本策略的利益和代价,提高副本策略对系统的贡献度.

6.2 数据一致性策略研究

Brewer 提出的 CAP 理论[34,35]证明了分布式系统不可能同时保证强一致性、可用性和容错性,为此,最终一致性[40]和弱一致性(如因果一致性(causal consistency)[91])等被引用到云环境中.然而,弱一致性语义只能适应特定的应用,如合作编辑,而对于银行转账等业务必须保证强一致性.可见,一致性需求与应用需求密切相关,并且数据一致性策略的性能对系统性能起到重要作用,因为其与读写访问密切相关.在云环境下,强一致性意味着事务执行的高代价、低可用性,但避免了惩罚代价;低一致性导致操作的低代价,但可能导致高昂的惩罚代价.但很难找到低执行代价和高惩罚代价的平衡点,因为它与应用语义密切相关.已有 key-value 数据库系统的一致性主要有 NRW 最终一致性[35]和应用 Paxos 算法[92](如 Chubby 锁服务[93])的强一致性.虽然 NRW 最终一致性可配置,但需要事先设置,缺乏灵活性.为此,文献[48]提出了一种动态一致性策略,给出了用于评估惩罚代价的评价模型,根据惩罚代价来确定一致性级别,即在代价、一致性和可用性间找到恰当的平衡点.相关的研究还有文献[49−51].文献[50]提出了基于检测的可适应的一致性保证框架(IDEA),当检测到不能满足一致性需求时进行调整.文献[51]提出了一组覆盖一致性的度量尺度,侧重用最大偏离值来保证一致性.文献[49]将数据分为几类,为每类提供不同的副本一致性策略.

在已有研究中,事先定义一致性策略不灵活,而实时检测代价大,因此,多种一致性策略共存并研究轻量级的一致性调整方案应是这方面的研究方向[71].比如,强一致性和弱一致性如何融合在一个系统中,两者如何相互作用来平衡系统的性能和可扩展性等.

7 基于 Map Reduce 框架的数据分析研究

NoSQL 数据库系统的典型特点是支持简单查询操作,而将复杂查询交给应用层处理,基于 MapReduce 框架高效实现海量数据分析是典型的代表.目前最新研究可分为基于 MapReduce 框架优化海量数据并行处理策略的研究和 NoSQL 数据库与 MapReduce 结合的研究.

1) 采用 MapReduce 框架并行处理海量数据的研究

MapReduce 模型[26]是云计算环境下的产物,并受到了业界的广泛关注和认可.MapReduce 模型基础框架简单,适合于海量数据的聚集计算,但支持的功能有限,具有一定的局限性.为了改进其对数据处理的支持能力,近期进行的典型改进研究有:MRShare[57]为了提高查询效率,将一批查询任务进行重新合并而形成新的组,在 MapReduce 模型下,针对各组定义优化模型,从而提高批处理查询的总体效率;Hadoop++[58]是在 Hadoop 基础上

提出的,其通过用户定义函数(UDF)规划数据的水平划分,呈现类似于数据库的物理查询执行计划,并基于索引优化数据的读入,从而提高 MapReduce 的数据处理性能;Llama[59]是一个基于聚簇的数据仓库系统,它采用行-列感知的混合数据管理系统,底层应用分布式文件系统(DFS)分布存储数据,上层基于 MapReduce 查询引擎实现快速的 Join 处理.

目前,这方面的研究挑战主要是改善 MapReduce 框架的支持能力,进一步提高数据处理性能,并灵活地适应各种数据分析任务.

2) NoSQL 数据库和 MapReduce 结合的研究

将 NoSQL 数据库的高并发读写能力和 MapReduce 的高效并行处理能力有效结合,是当前实现海量数据管理的必然趋势.即针对应用需求,将面向批处理的系统(基于 MapReduce 框架)和面向服务的系统(NoSQL 数据库)相结合的应用研究,如 Hadoop 和 Hbase 相结合的应用[60]、Hadoop 和 PNUTS 相结合的应用[61].其思想是:利用批处理系统优化大量请求的读写吞吐率,服务系统使用索引提供低延迟的记录访问,即通过两者结合支持海量数据管理与查询处理.

应用 MapReduce 机制实现海量数据处理,实现简单但也具有一定的局限性,如复杂查询和数据分析的实现代价较大.为此,需要有更优化的海量数据分布处理模型来支持更丰富的数据分析功能.

8 其他研究

规范标准、测试基准的制定的研究:提出新技术之后,往往接着推出相应的标准.就云计算而言,目前已有许多云协同联盟和组织,如 Distributed Management Task Force(DMTF),National Institute of Standards and Technology(NIST),Open Cloud Consortium(OCC),Open Grid Forum(OGF),Open Cloud Computing Interface Working Group 等,都在为云定义相应的标准,但还缺乏开放性和协作性.为此,需要有相应的规范标准的制定,使云服务开放,方便更多的研究者进入云研究市场,方便用户选择云服务提供者.而对于云环境下的海量数据管理系统,不同的云提供者已提供相似的 API 接口,如 MapReduce 和 Hadoop 提供了相似的 API;Amazon's Dynamo, Yahoo!'s PNUTS 等也提供了相似的 key-value 查询服务;其他组件如 Google's Chubby,Yahoo!'s Zoo-keeper[94] 等也提供了不同的 API.但以何种方式提供组件和接口、如何比较和对比相似组件的实现、如何评价 key-value 存储系统的存储性能等,都还没有相应的遵循标准.同时,云平台还需要有类似于用于评价数据库的 TPC benchmark,等等.

数据的安全性研究[95,96]:云环境下存储个人数据,如何保证个人数据的安全性至关重要.有针对个人数据的私密性和安全性的研究,如在云内定义安全性、数据加密、在加密数据上进行相应的操作、面向特定应用的数据加密/解密操作等.侧重于研究如何平衡数据的安全与计算的复杂度,希望提出可行的、可信的商业模型.

还有云环境下能耗的研究[95,97],如 Solid-state disks(SSDs)的快速访问和低能耗特性、采用 SSDs 来代替硬盘存储读密集型的数据、数据中心中商用机的最小能量消耗、设计更好的改进能效的硬件和框架结构、云节点上数据和计算的组织以及在不降低性能的情况下降低软件的能耗等.

9 结束语及研究展望

Key-Value 数据库是应用于云环境下的典型云存储系统,并被业界广泛关注,为此,各商家面向特定应用推出了许多 key-value 数据库.随着技术的发展和应用需求的不断变化,也出现了一些新一代 key-value 数据库和相关研究成果.虽然各具特色,但也都有一定的局限性,还有许多挑战性工作进行深入研究.结合已有相关研究成果及其发展趋势,我们认为,为了有效地支持大数据管理,以下有关 NoSQL 及其相关研究将是研究者近年来所关注的问题:

(1) 扩展 key-value 数据模型与多种数据模型共存的研究

Key-Value 数据模型的出现,满足了海量数据管理的可扩展性、简单查询的高效率等应用需求.目前,key-value 数据存储系统典型的是基于 key 哈希存储或基于范围存储,并只支持简单的查询操作,限制了 key-value 存

储系统的应用.为此,出现了支持类 SQL 语言的 key-value 存储系统,如 Hive 和目前广泛流行的 MapReducec 并行处理模型.然而,它们只是在上层构建了一个支持复杂查询的视图定义模型,并没有改变底层的结构.为了扩大 key-value 数据存储系统的应用范围,需要增强 key-value 存储系统的支持功能,如复杂查询、事务能力等.但如果不从根本上完善核心的数据模型,很难获得较好的性能.因此,针对 key-value 数据存储系统,研究者将关注:1) 如何改进基于 key 哈希的数据分布模式,使其能够更有效地支持灵活的复杂查询和具有低惩罚代价的事务特性等.2) 扩展 key-value 数据模型的研究,如基于 key-value 数据模型构建多样的索引,尤其是全局索引,以扩展其查询能力和查询效率.3) 多种数据模型无缝转换的研究,以支持多种数据模型共存来满足复杂数据的管理需求等.

(2) 支持分布式事务的研究

支持事务特性的数据管理系统可以有效地支持以 OLTP 类业务为主的应用需求.针对 key-value 数据存储系统典型的只支持单 key 的事务特性,尽管提出了一些基于多 key 的事务特性或将事务限制在同组内而避开 2PC 协议的解决方案,但还是不能灵活而低代价地支持真正的分布式事务.因此,如何基于 key-value 数据模型提出有效而灵活的分布式事务模型应是研究者关注的热点.重点研究轻量级的、灵活的事务模型,可满足各类应用需求.要易于配置,并可提供简单且健壮的解决方案.

(3) 提高系统查询处理性能的研究

基于 key-value 数据模型的数据存储系统只支持简单的数据查询操作,为此,典型的是采用 MapReduce 处理模型实现查询处理操作.MapReduce 模型简单,非常适合于可将一个事务可划分为多个独立子事务的聚集查询处理业务.而对于大数据的复杂查询处理需求,所有查询处理操作都需要转换为一轮或多轮 MapReduce 模型实现,导致传输代价大,不利于改善数据的查询处理性能.因此,提出面向数据查询处理的高性能处理模型将会引起研究者所关注.同时,为提高系统资源利用率、节约能源、提高系统吞吐率、提高系统整体性能,面向资源负载均衡、数据负载均衡、任务负载均衡的动态负载平衡模型的研究也将是研究者关注的热点.

(4) 有效利用副本资源的研究

副本是分布式数据管理系统的典型特点,有助于提高系统可用性、可靠性、系统查询处理效率等.然而,已有成果如动态负载平衡、事务特性、高效的查询处理等,都未能有效地利用副本数据.已有系统的索引模型、事务模型、动态负载平衡模型、优化的查询处理模型等也都对副本考虑得很少,未能有效地利用好副本的作用.因此在未来的研究中,如何有效地利用副本数据将是热点研究问题.

(5) 支持理论的研究和规范标准的制定

自 key-value 数据模型提出以后,其简单的数据模型结构受到了大数据管理者的青睐.虽然 key-value 数据模型广泛被 Google,Facebook 等大公司所采用,但还没有可遵循的规范标注和理论支持.30 年前,关系模型因其模型简单、易于管理和应用等特点,尤其是关系理论的提出,造就了关系数据库在应用了 30 多年后的今天,仍占据数据管理的主导地位.为了支持云环境下的大数据管理,key-value 数据模型有其固有的优点和应用优势,但目前有关 key-value 数据模型的支持理论还是空白,需要深入研究.同时,随着相关研究成果的出现,还需要形成一定的技术规范标准.

除此之外,云环境下数据管理的数据安全至关重要,有关云数据安全的研究也是研究热点,还有支持数据管理的支持系统的能耗问题等.

大数据带来了大机遇,key-value 数据存储系统也成为研究者关注的热点问题.为了有效地支持云环境下的数据管理,有关 key-value 数据存储系统及其相关研究还有许多挑战性问题,需要研究者去研究和探讨.

References:

[1] Big data. 2011. http://en.wikipedia.org/wiki/Big_data

[2] Zhou XF, Lu JH, Li CP, Du XY. The challenges of big data from the perspective of data management. Communications of the China Computer Federation, 2012,8(9):16−21 (in Chinese).

[3] Li GJ. The scientific value of big data research. Communications of the China Computer Federation, 2012,8(9):8−15 (in Chinese).

[4] The Internet Analysis Salon. Big data is coming. 2011 (in Chinese). http://www.techxue.com/portal.php?mod=view&aid=55
[5] Informatica. Big data unleased. 2011. http://www.informatica.com/downloads/1601_big_data_wp.pdf
[6] Ma S, Li JX, Hu CM. The challenge and thinking of big data science and engineering. Communications of the China Computer Federation, 2012,8(9):22−28 (in Chinese).
[7] Rys M. scalable SQL. Communications of the ACM, 2011,54(6):48−53. [doi: 10.1145/1953122.1953141]
[8] NoSQL. 2011. http://zh.wikipedia.org/wiki/NoSQL
[9] NoSQL. 2011. http://nosql-databases.org/
[10] Campbell DG, Kakivaya G, Ellis N. Extreme scale with full SQL language support in microsoft SQL azure. In: Proc. of the SIGMOD. New York: ACM Press, 2010. 1021−1024. [doi: 10.1145/1807167.1807280]
[11] Brantner M, Florescu D, Graf D, Kossmann D, Kraska T. Building a database on S3. In: Proc. of the SIGMOD. New York: ACM Press, 2008. 251−264. [doi: 10.1145/1376616.1376645]
[12] Curino C, Jones EPC, Popa RA, Malviya N, Wu E, Madden S, Balakrishnan H, Zeldovich N. Relational cloud: A database as a service for the cloud. In: Proc. of the CIDR. 2011. 235−240.
[13] NOSQL. 2009. http://nosql.eventbrite.com/
[14] Nosqleast conference. 2010. https://nosqleast.com/2009/
[15] Chang F, Dean J, Ghemawat S, Hsieh WC, Wallach DA, Burrows M, Chandra T, Fikes A, Grubwr RE. Bigtable: A distributed storage system for structured data. In: Proc. of the OSDI. New York: ACM Press, 2006.
[16] Borthaku D. The hadoop distributed file system: Architecture and design. 2009. http://hadoop.apache.org/common/docs/r0.18.0/hdfs_design.pdf
[17] Hbase Development Team. Hbase: Bigtable-like structured storage for hadoop HDFS. 2009. http://wiki.apache.org/hadoop/Hbase
[18] DeCandia G, Hastorun D, Jampani M, Kakulapati G, Lakshman A, Pilchin A, Sivasubramanian S, Vosshall P, Vogels W. Dynamo: Amazon's highly available key-value store. In: Proc. of the SOSP. New York: ACM Press, 2001. 205−220. [doi: 10.1145/1294261.1294281]
[19] Lakshman A, Malik P. Cassandra—A decentralized structured storage system. 2009. http://www.cs.cornell.edu/projects/ladis2009/papers/lakshman-ladis2009.pdf
[20] The Document Collection of Tokyocabinet/Tokyotyrant. 2010 (in Chinese). http://www.162cm.com/p/tokyotyrant.html#toc5
[21] Apache CouchDB: The apache CouchDB project. 2010. http://couchdb.apache.org/
[22] MongoDB. 2010. http://www.mongodb.org
[23] Redis. 2010. http://redis.io/
[24] Gilbert S, Lynch N. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant Web services. ACM SIGACT News, 2002,33(2):51−59. [doi: 10.1145/564585.564601]
[25] Pritchett D. BASE: An acid alternative. 2008. http://queue.acm.org/detail.cfm?id=1394128
[26] Agrawal R, Ailamaki A, Bernstein P A, Brewer E A, Carey M J, Chaudhuri S, Doan A, Florescu D, Franklin M J, Garcia-Molina H, Gehrke J, Gruenwald L, Hass L M, Halevy A, Hellerstein J M, Ioannidis Y E, Korth H K, Kossmann D, Madden S, Magoulas R, Ooi B C, O'Reilly T, Ramakrishnan R, Sarawagi S, Stonebraker M, Szalay A S, Weikum G. The claremont report on database research. Communications of the ACM, 2009,52(6):56−65. [doi: 10.1145/1516046.1516062]
[27] Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters. In: Proc. of the OSDI. New York: ACM Press, 2004. 1−13.
[28] Stonebraker M, Cattell R. 10 rules for scalable performance in 'simple operation' datastores. Communications of the ACM, 2011, 54(6):72−80. [doi: 10.1145/1953122.1953144]
[29] Baker J, Bond C, Corbett JC, Furman JJ, Khorlin A, Larson J, Léon JM, Li YW, Lloyd A, Yushprakh V. Megastore: Providing scalable, highly available storage for interactive services. In: Proc. of the CIDR. 2011. 223−234.
[30] Das S, Agrawal D, Abbadi AE. G-Store: A scalable data store for transactional multi-key access in the cloud. In: Proc. of the SoCC. New York: ACM Press, 2010. [doi: 10.1145/1807128.1807157]
[31] Vo HT, Chen C, Ooi BC. Towards elastic transactional cloud storage with range query support. Proc. of the VLDB Endowment, 2010,3(1-2):506−514.
[32] Wang S, Sa SX. Introduction to Database System. 4th ed., Beijing: Higher Education Press, 2006 (in Chinese).
[33] Yang F, Shanmugasundaram J, Yerneni R. A scalable data platform for a large number of small applications. In: Proc. of the CIDR. 2009.
[34] Petersen K, Spreitzer MJ, Terry DB, Theimer MM, Demers AJ. Flexible update propagation for weakly consistent replication. In Proc. of the SOSP. New York: ACM Press, 1997. 288−301. [doi: 10.1145/268998.266711]
[35] Vogels W. Eventually consistent. Communications of the ACM, 2009,52(1):40−44. [doi: 10.1145/1435417.1435432]
[36] Lamport L. The part-time parliament. ACM Trans. on Computer Systems, 1998,16(2):133−169. [doi: 10.1145/279227.279229]

[37] Levandoski JJ, Lomet D, Mokbel MF, Zhao KK. Deuteronomy: Transaction support for cloud data. In: Proc. of the CIDR. 2011. 123−133.

[38] Curino C, Jones E, Zhang Y, Wu E, Madden S. Relational cloud: The case for a database service. Technical Report, MIT-CSAIL-TR-2010-014, Massachusetts Institute of Technology, 2010.

[39] Das S, Agarwal D, Abbadi AE. ElasTraS: An elastic, scalable, and self managing transactional database for the cloud. Technical Report, 2010-4, Santa Barbara: University of California, 2010.

[40] Das S, Agrawal D, Abbadi AE. ElasTraS: An elastic transactional data store in the cloud. In: Proc. of the USENIX HotCloud Workshop. 2009.

[41] Wei Z, Pierre G, Chi CH. CloudTPS: Scalable transactions for Web applications in the cloud. Technical Report, IR-CS-053, Amsterdam: Vrije Universiteit, 2010.

[42] Tatemura J, Po O, Hsiung WP, Hacigümüş H. Partiqle: An elastic SQL engine over key-value stores. In: Proc. of the SIGMOD. New York: ACM Press, 2012. 629−632.

[43] Konstantinou I, Angelou E, Tsoumakos D. TIRAMOLA: Elastic NoSQL provisioning through a cloud management platform. In: Proc. of the SIGMOD. New York: ACM Press, 2012. 725−728. [doi: 10.1145/2213836.2213943]

[44] Okcan A, Riedewald M. Processing Theta-joins using MapReduce. In: Proc. of the SIGMOD. New York: ACM Press, 2011. 949−960. [doi: 10.1145/1989323.1989423]

[45] Nehme R, Bruno N. Automated partitioning design in parallel database systems. In: Proc. of the SIGMOD. New York: ACM Press, 2011. 1137−1148. [doi: 10.1145/1989323.1989444]

[46] Das S, Nishimura S, Agrawal D, Abbadi AE. Live database migration for elasticity in a multitenant database for cloud platforms. Technical Report, 2010-09, Department of Computer Science, University of California at Santa Barbara, 2010.

[47] Elmore AJ, Das S, Agrawal D, Abbadi AE. Zephyr: Live migration in shared nothing databases for elastic cloud platforms. In: Proc. of the SIGMOD. New York: ACM Press, 2011. 301−312. [doi: 10.1145/1989323.1989356]

[48] Kraska T, Hentschel M, Alonso G, Kossmann D. Consistency rationing in the cloud: Pay only when it matters. Proc. of the VLDB Endowment, 2009,2(1):253−264.

[49] Gao L, Dahlin M, Nayate A, Zheng JD, Lyengar A. Application specific data replication for edge services. In: Proc. of the WWW. New York: ACM, 2003. 449−460. [doi: 10.1145/775152.775217]

[50] Lu Y, Lu Y, Jiang H. Adaptive consistency guarantees for large-scale replicated services. In: Proc. of the NAS. IEEE, 2008. 89−96. [doi: 10.1109/NAS.2008.64]

[51] Yu H, Vahdat A. Design and evaluation of a continuous consistency model for replicated services. In: Proc. of the OSDI. New York: ACM Press, 2000. 305−318.

[52] Debnath B, Sengupta S, Li J. SkimpyStash: RAM space skimpy key-value store on flash-based storage. In: Proc. of the SIGMOD. New York: ACM, 2011. 25−36. [doi: 10.1145/1989323.1989327]

[53] Anand A, Muthukrishnan C, Kappes S, Akella A, Nath S. Cheap and large CAMs for high performance data-intensive networked systems. In: Proc. of the NSDI. 2010. 29.

[54] Andersen1 DG, Franklin J, Kaminsky M, Phanishayee A, Tan L, Vasudevan V. FAWN: A fast array of wimpy nodes. In: Proc. of the SOSP. New York: ACM Press, 2009. [doi: 10.1145/1629575.1629577]

[55] Debnath B, Sengupta S, Li J. ChunkStash: Speeding up inline storage deduplication using flash memory. In: Proc. of the USENIX. 2010.

[56] Debnath B, Sengupta S, Li J. FlashStore: High throughput persistent key-value store. In: Proc. of the VLDB. Morgan Kaufmann, ACM, 2010. 1414−1425.

[57] Nykiel T, Potamias M, Mishra C, Kollios G, Koudas N. MRShare: Sharing across multiple queries in MapReduce. In: Proc. of the VLDB. Morgan Kaufmann, ACM, 2010. 494−505.

[58] Dittrich J, Quiané-Ruiz JA, Jindal A, Kargin Y, Setty V, Schad J. Hadoop++: Making a yellow elephant run like a cheetah (without it even noticing). In: Proc. of the VLDB. Morgan Kaufmann, ACM, 2010. 518−527.

[59] Lin Y, Agrawal D, Chen C. Llama: Leveraging columnar storage for scalable join processing in the MapReduce framework. In: Proc. of the SIGMOD. New York: ACM Press, 2011. 961−972. [doi: 10.1145/1989323.1989424]

[60] Borthakur D, Sarma JS, Gray J. Apache hadoop goes realtime at facebook. In: Proc. of the SIGMOD. New York: ACM Press, 2011. 1071−1080. [doi: 10.1145/1989323.1989438]

[61] Silberstein A, Sears R, Zhou WC, Cooper BF. A batch of PNUTS: Experiences connecting cloud batch and serving systems. In: Proc. of the SIGMOD. New York: ACM Press, 2011. 1101−1112. [doi: 10.1145/1989323.1989441]

[62] Kossmann D, Kraska T, Loesing S, Merkli S, Mittal R, Pfaffhauser F. Cloudy: A modular cloud storage system. In: Proc. of the VLDB. Morgan Kaufmann, ACM, 2010. 1533−1536.

[63] O'reilly. Cassandra: The definitive Guide. 2010. http://ishare.iask.sina.com.cn/f/17348687

[64] Wu S, Jiang DW, Ooi BC, Wu KL. Efficient B-tree based indexing for cloud data processing. In: Proc. of the VLDB. Morgan Kaufmann, ACM, 2010.

[65] Wang JB, Wu S, Gao H, Li JZ, Ooi BC. Indexing multi-dimensional data in a cloud system. In: Proc. of the SIGMOD. New York: ACM Press, 2010. 591−602. [doi: 10.1145/1807167.1807232]

[66] Tsatsanifos G, Sacharidis D, Sellis T. MIDAS: Multi-Attribute indexing for distributed architecture systems. In: Proc. of the SSTD. Heidelberg: Springer-Verlag, 2011. 168−185. [doi: 10.1007/978-3-642-22922-0_11]

[67] Escriva R, Wong B, Gün Sirer EG. HyperDex: A distributed, searchable key-value store. In: Proc. of the SIGCOMM. 2012. 1−12. [doi: 10.1145/2377677.2377681]

[68] Meng BP, Wang TJ, Li HY, Yang DQ. Regional bitmap index: A secondary index for data management in cloud computing environment. Chinese Journal of Computers, 2012,35(11): 2306−2316 (in Chinese with English abstract).

[69] Aguilera MK, Golab W, Shah MA. A practical scalable distributed B-tree. In: Proc. of the VLDB. Morgan Kaufmann, ACM, 2008.

[70] Zhang XY, Ai J, Wang ZY, Lu JH, Meng XF. An efficient multi-dimensional index for cloud data management. In: Proc. of the CloudDB. New York: ACM Press, 2009. 17−24. [doi: 10.1145/1651263.1651267]

[71] Cooper BF, Ramakrishnan R, Srivastava U, Silberstein A,Bohannon P, Jacobsen HA, Puz N, Weaver D, Yerneni R. PNUTS: Yahoo!'s hosted data serving platform. Proc. of the VLDB Endowment, 2008,1(2):1277−1288.

[72] Amer-Yahia S, Halevy A, Alonso G, Kossmann D, Markl V, Doan AH, Weikum G. Databases and Web 2.0 panel at VLDB 2007. SIGMOD Record, 2008,37(1):49−52. [doi: 10.1145/1374780.1374794]

[73] Bernstein PA, Cseri I, Dani N, Ellis N, Kalhan A, Kakivaya G, Lomet DB, Manne R, Novik L, Talius T. Adapting Microsoft SQL server for cloud computing. In: Proc. of the ICDE. IEEE, 2011. 1255−1263. [doi: 10.1109/ICDE.2011.5767935]

[74] Gray J, Lamport L. Consensus on transaction commit. ACM Trans. on Database System, 2006,31(1):133−160. [doi: 10.1145/1132863.1132867]

[75] Aguilera MK, Merchant A, Shah M, Veitch A, Karamanolis C. Sinfonia: A new paradigm for building scalable distributed systems. In: Proc. of the SOSP. New York: ACM Press, 2007. [doi: 10.1145/1294261.1294278]

[76] Thomson A, Diamond T, Weng SC. Calvin: Fast distributed transactions for partitioned database systems. In: Proc. of the SIGMOD. New York: ACM Press, 2012. [doi: 10.1145/2213836.2213838]

[77] Das S, Nishimura S, Agrawal D, Abbadi AE. Albatross: Lightweight elasticity in shared storage databases for the cloud using live data migration. In: Proc. of the VLDB. Morgan Kaufmann, ACM, 2011. 494−505.

[78] Clark C, Fraser K, Hand S, Hansen JG, Jul E, Limpach C, Pratt I, Warfield A. Live migration of virtual machines. In: Proc. of the NSDI. 2005. 273−286.

[79] Liu HK, Jin H, Liao XF, Hu LT, Yu C. Live migration of virtual machine based on full system trace and replay. In: Proc. of the HPDC. 2009. 101−110. [doi: 10.1145/1551609.1551630]

[80] De Witt D, Gray J. Parallel database systems: The future of high performance database systems. Communications of the ACM, 1992,35(6):85−98. [doi: 10.1145/129888.129894]

[81] Ganguly S, Goel A, Silberschatz A. Efficient and accurate cost models for parallel query optimization. In: Proc. of the PODS. New York: ACM Press, 1996. 172−181. [doi: 10.1145/237661.237707]

[82] Isard M, Budiu M, Yu Y. Dryad: Distributed data-parallel programs from sequential building blocks. In: Proc. of the EuroSys. New York: ACM Press, 2007. 59−72. [doi: 10.1145/1272996.1273005]

[83] Yang HC, Dasdn A, Hsiao RL, Parker DS. Map-Reduce-Merge: Simplified relational data processing on large clusters. In: Proc. of the SIGMOD. New York: ACM Press, 2007. [doi: 10.1145/1247480.1247602]

[84] Chaiken R, Jenkins B, Larson PÅ, Ramsey B, Shakib D, Weaver S, Zhou JR. Scope: Easy and efficient parallel processing of massive data sets. In: Proc. of the VLDB. New York: ACM Press, 2008. 1265−1276.

[85] Curino C, Jones E, Zhang Y, Madden S. Schism: A workload-driven approach to database replication and partitioning. Proc. of the VLDB Endowment, 2010,3(1-2):48−57.

[86] Karypis G, Kumar V. A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM Journal on Scientific Computing, 1998,20(1):359−392. [doi: 10.1137/S1064827595287997]

[87] Ghandeharizadeh S, De Witt DJ. Hybrid-Range partitioning strategy: A new declustering strategy for multiprocessor database machines. In: Proc. of the VLDB. 1990. 481−492.

[88] Gufler B, Augsten N, Reiser A, Kemper A. Load balancing in MapReduce based on scalable cardinality estimates. In: Proc. of the ICDE. IEEE, 2012. [doi: 10.1109/ICDE.2012.58]

[89] Kwon YC, Balazinska M, Howe B, Rolia J. SkewTune: Mitigating skew in MapReduce applications. In: Proc. of the SIGMOD. New York: ACM Press, 2012. 25−36. [doi: 10.1145/2213836.2213840]

[90] Chen C, Chen G, Jiang DW, Ooi BC, Vo HT, Wu S, Xu QQ. Providing scalable database services on the cloud. In: Proc. of the WISE. Heidelberg: Springer-Verlag, 2010. 1−19.

[91] Vogels W. Eventually consistent. 2008. http://www.allthingsdistributed.com/2008/12/eventually_consistent.html

[92] Rao J, Shekita EJ, Tata S. Using Paxos to build a scalable, consistent, and highly available datastore. In: Proc. of the VLDB. 2011. 243−354.

[93] Burrows M. The chubby lock service for loosely-coupled distributed system. In: Proc. of the OSDI. 2006. 335−350.

[94] Reed B, Junqueira FP. A simple totally ordered broadcast protocol. In: Proc. of the Workshop on LADIS. 2008. 1−6. [doi: 10.1145/1529974.1529978]

[95] Vigfusson Y, Chockler G. Clouds at the crossroads research perspectives. Spring, 2010,16(3):10−13. [doi: 10.1145/1734160.1734165]

[96] Lin ZY, Lai YX, Lin C, Xie Y, Zou Q. Research on Cloud Databases. Ruan Jian Xue Bao/Journal of Software, 2012,23(5): 1148−1166 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4195.htm [doi: 10.3724/SP.J.1001.2012.04195]

[97] Wang YJ, Sun WD, Zhou S, Pei XQ, Li XY. Key Technologies of distributed storage for cloud computing. Ruan Jian Xue Bao/Journal of Software, 2012,23(4):962−986 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4175.htm [doi: 10.3724/SP.J.1001.2012.04175]

附中文参考文献:

[2] 周晓方,陆嘉恒,李翠平,杜小勇.从数据管理视角看大数据挑战.中国计算机学会通讯,2012,8(9):16−21.

[3] 李国杰.大数据研究的科学价值.中国计算机学会通讯,2012,8(9):8−15.

[4] 互联网分析沙龙.海量数据来了.2011. http://www.techxue.com/portal.php?mod=view&aid=55

[6] 马帅,李建欣,胡春明.大数据科学与工程的挑战与思考.中国计算机学会通讯,2012,8(9):22−28.

[20] Tokyocabinet/Tokyotyrant 文档大合集.2010. http://www.162cm.com/p/tokyotyrant.html#toc5

[32] 王珊,萨师煊.数据库系统概论.第 4 版,北京:高等教育出版社,2006.

[68] 孟必平,王腾蛟,李红燕,杨冬青.分片位图索引:一种适用于云数据管理的辅助索引机制.计算机学报,2012,35(11):2306−2316.

[96] 林子雨,赖永炫,林琛,谢怡,邹权.云数据库研究.软件学报,2012,23(5):1148−1166. http://www.jos.org.cn/1000-9825/4195.htm [doi: 10.3724/SP.J.1001.2012.04195]

[97] 王意洁,孙伟东,周松,裴晓强,李小勇.云计算环境下的分布存储关键技术.软件学报,2012,23(4):962−986. http://www.jos.org.cn/1000-9825/4175.htm [doi: 10.3724/SP.J.1001.2012.04175]

申德荣(1964－),女,辽宁铁岭人,博士,教授,博士生导师,CCF 高级会员,主要研究领域为分布式数据库,Web 数据管理,数据集成.
E-mail: shenderong@ise.neu.edu.cn

于戈(1962－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为数据库,大数据管理,图数据,数据挖掘.
E-mail: yuge@mail.neu.edu.cn

王习特(1987－),男,博士生,CCF 学生会员,主要研究领域为密集型数据管理.
E-mail: wangxite@research.neu.edu.cn

聂铁铮(1980－),男,博士,副教授,CCF 会员,主要研究领域为数据质量管理.
E-mail: nietiezheng@ise.neu.edu.cn

寇月(1980－),女,博士,副教授,CCF 会员,主要研究领域为实体识别.
E-mail: kouyue@ise.neu.edu.cn