软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2012,23(5):1148−1166 [doi: 10.3724/SP.J.1001.2012.04195]

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

云数据库研究*

林子雨[1], 赖永炫[2+], 林 琛[1], 谢 怡[1], 邹 权[1]

[1](厦门大学 计算机科学系,福建 厦门 361005)

[2](厦门大学 软件学院,福建 厦门 361005)

Research on Cloud Databases

LIN Zi-Yu[1], LAI Yong-Xuan[2+], LIN Chen[1], XIE Yi[1], ZOU Quan[1]

[1](Department of Computer Science, Xiamen University, Xiamen 361005, China)

[2](School of Software, Xiamen University, Xiamen 361005, China)

+ Corresponding author: E-mail: laiyx@xmu.edu.cn

Lin ZY, Lai YX, Lin C, Xie Y, Zou Q. Research on cloud databases. *Journal of Software*, 2012,23(5): 1148−1166. http://www.jos.org.cn/1000-9825/4195.htm

Abstract: With the recent development of cloud computing, the importance of cloud databases has been widely acknowledged. Here, the features, influence and related products of cloud databases are first discussed. Then, research issues of cloud databases are presented in detail, which include data model, architecture, consistency, programming model, data security, performance optimization, benchmark, and so on. Finally, some future trends in this area are discussed.

Key words: cloud computing; cloud database; key-value store; transaction consistency

摘 要: 随着云计算的发展,云数据库的重要性和价值日益显现.介绍了云数据库的特性、影响、相关产品.详细讨论了云数据库领域的研究问题,包括数据模型、系统体系架构、事务一致性、编程模型、数据安全、性能优化和测试基准等.最后讨论了云数据库未来的研究方向.

关键词: 云计算;云数据库;键值存储;事务一致性

中图法分类号: TP311 **文献标识码**: A

云计算(cloud computing)[1]是 IT 技术发展的最新趋势,正受到业界和学术界的广泛关注[2,3].云计算是在分布式处理、并行处理和网格计算等技术的基础上发展起来的,是一种新兴的共享基础架构的方法.它可以自我维护和管理庞大的虚拟计算资源(包括计算服务器、存储服务器、宽带资源等等),从而提供各种 IT 服务.用户在使用云计算提供的服务时按需付费,这不仅降低了使用门槛,也极大地节省了开销.由于云计算存在着巨大的潜在市场,Google,IBM,Microsoft,Amazon,Sun,HP,Yahoo,Oracle 等国际知名大公司都已经涉足云计算.云计算也开始在电信、金融等需要大规模并行处理的领域得到应用,比如中国移动研究院开发的云数据挖掘平台 BC-

* 基金项目: 厦门大学基础创新科研基金(中央高校基本科研业务费专项资金)(2011121049, 2010121066); 国家自然科学基金(61001013, 61102136); 福建省自然科学基金(2011J05156, 2011J05158)

收稿时间: 2011-06-21; 修改时间: 2011-09-02, 2011-10-17; 定稿时间: 2012-02-15; jos 在线出版时间: 2012-02-27

CNKI 网络优先出版: 2012-02-27 11:43, http://www.cnki.net/kcms/detail/11.2560.TP.20120227.1143.001.html

PDM[4]和云数据库产品 HugeTable[5].

随着云计算技术的不断升温,它对各个技术领域的影响开始显现,其中比较典型的包括数据库领域[6−10].截止到 2011 年 6 月,传统的数据库厂商,比如 Oracle,Teradata,IBM,Microsoft 等,都已经推出了基于云计算环境的相关数据库产品.原来没有从事数据库产品开发的知名大公司,比如 Amazon 和 Google 等,也发布了 SimpleDB 和 BigTable[11]等产品.

迅速发展的云数据库市场极大地影响着数据库技术的未来发展方向,甚至出现了关系数据库是否已经没落的争议.与此同时,许多云数据库的相关问题开始被关注,比如云数据库的体系架构、数据模型、事务一致性、数据安全和性能优化等等.由于云数据库是一个比较新的研究领域,目前还很少有相关研究对这个领域进行全面概括的介绍.因此,本文将结合我们在过去两年中已经进行的大量调研成果,对云数据库及其相关研究进行综合阐述.

本文第 1 节介绍云数据库,包括云数据库的特性、影响、云数据库与传统分布式数据库的区别.第 2 节介绍相关的云数据库产品.第 3 节阐述云数据库领域的研究问题.第 4 节是本文的结论,并给出未来研究展望.

1 云数据库概述

云数据库是在 SaaS(software-as-a-service:软件即服务)成为应用趋势的大背景下发展起来的云计算技术,它极大地增强了数据库的存储能力,消除了人员、硬件、软件的重复配置,让软、硬件升级变得更加容易,同时也虚拟化了许多后端功能.云数据库具有高可扩展性、高可用性、采用多租形式和支持资源有效分发等特点.可以说,云数据库是数据库技术的未来发展方向.目前,对于云数据库的概念界定不尽相同,本文采用的云数据库定义是:云数据库是部署和虚拟化在云计算环境中的数据库[12].

如图 1 所示,在云数据库应用中,客户端不需要了解云数据库的底层细节,所有的底层硬件都已经被虚拟化,对客户端而言是透明的.它就像在使用一个运行在单一服务器上的数据库一样,非常方便、容易,同时又可以获得理论上近乎无限的存储和处理能力.

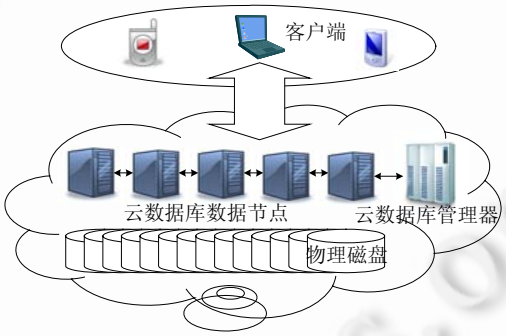

Fig.1 A diagram showing the application of cloud databases

图 1 云数据库应用示意图

1.1 云数据库的特性

云数据库具有以下特性:

(1) 动态可扩展:理论上,云数据库具有无限可扩展性,可以满足不断增加的数据存储需求.在面对不断变化的条件时,云数据库可以表现出很好的弹性.例如,对于一个从事产品零售的电子商务公司,会存在季节性或突发性的产品需求变化;或者对于类似 Animoto 的网络社区站点,可能会经历一个指数级的增长阶段.这时,就可以分配额外的数据库存储资源来处理增加的需求,这个过程只需要几分钟.一旦需求过去以后,就可以立即释放这些资源.

(2) 高可用性:不存在单点失效问题.如果一个节点失效了,剩余的节点就会接管未完成的事务.而且在云数

据库中,数据通常是复制的,在地理上也是分布的.诸如 Google,Amazon 和 IBM 等大型云计算供应商具有分布在世界范围内的数据中心,通过在不同地理区间内进行数据复制,可以提供高水平的容错能力.例如,Amazon SimpleDB 会在不同的区间内进行数据复制,因此,即使整个区域内的云设施发生失效,也能保证数据继续可用.

(3) 较低的使用代价:通常采用多租户(multi-tenancy)的形式,这种共享资源的形式对于用户而言可以节省开销;而且用户采用按需付费的方式使用云计算环境中的各种软、硬件资源,不会产生不必要的资源浪费.另外,云数据库底层存储通常采用大量廉价的商业服务器,这也大幅度降低了用户开销.

(4) 易用性:使用云数据库的用户不必控制运行原始数据库的机器,也不必了解它身在何处.用户只需要一个有效地链接字符串就可以开始使用云数据库.

(5) 大规模并行处理:支持几乎实时的面向用户的应用、科学应用和新类型的商务解决方案.

1.2 云数据库是海量存储需求的必然选择

云数据库在当前数据爆炸的时代具有广阔的应用前景.根据 IDC 的研究报告,在未来的 5 年中,企业对结构化数据的存储需求会每年增加 20%左右,而对非结构化数据的存储需求将会每年增加 60%左右.在小规模应用的情况下,系统负载的变化可以由系统空闲的多余资源来处理;但是在大规模应用的情况下,不仅存在海量的数据存储需求,而且应用对资源的需求也是动态变化的,这意味着大量虚拟机器的增加或减少.对于这种情形,传统的关系数据库已经无法满足要求,云数据库成为必然的选择.换句话说,海量存储催生了云数据库.

1.3 云数据库与传统的分布式数据库

分布式数据库是计算机网络环境中各场地或节点上的数据库的逻辑集合.逻辑上它们属于同一系统,而物理上它们分散在用计算机网络连接的多个节点,并统一由一个分布式数据库管理系统管理.

分布式数据库已经存在很多年,它可以用来管理大量的分布存储的数据,并且通常采用非共享的体系架构.云数据库和传统的分布式数据库具有相似之处,比如,都把数据存放到不同的节点上.但是,分布式数据库在可扩展性方面是无法与云数据库相比的.由于需要考虑数据同步和分区失败等开销,前者随着节点的增加会导致性能快速下降.而后者则具有很好的可扩展性,因为后者在设计时就已经避免了许多会影响到可扩展性的因素,比如采用更加简单的数据模型、对元数据和应用数据进行分离以及放松对一致性的要求等等[13].另外,在使用方式上,云数据库也不同于传统的分布式数据库.云数据库通常采用多租户模式,即多个租户共用一个实例,租户的数据既有隔离又有共享,从而解决数据存储的问题,同时也降低了用户使用数据库的成本.

1.4 云数据库的影响

云数据库的影响主要体现在以下几个方面:

(1) 极大地改变企业管理数据的方式.Forrester Research 分析师 Yuhanna 指出,18%的企业正在把目光投向云数据库.对于中小企业而言,云数据库可以允许他们在 Web 上快速搭建各类数据库应用,越来越多的本地数据和服务将逐渐被转移到云中.企业用户在任意地点通过简单的终端设备,就可以对企业数据进行全面的管理.此外,云数据库可以很好地支持企业开展一些短期项目,降低开销,而不需要企业为某个项目单独建立昂贵的数据中心.但是,云数据库的成熟仍然需要一段时间.中小企业会更多地采用云数据库产品,但是对于大企业而言,云数据库并非首选,因为大企业通常自己建造数据中心.

(2) 催生新一代的数据库技术.IDC 的数据库分析师 Olofson 认为,云模型提供了无限的处理能力以及大量的 RAM,因此,云模型将会极大地改变数据库的设计方式,将会出现第三代数据库技术.第一代是 20 世纪 70 年代的早期关系数据库,第二代是 20 世纪 80 年代~90 年代的更加先进的关系模型.第三代的数据库技术,要求数据库能够灵活处理各种类型的数据,而不是强制让数据去适应预先定制的数据结构.事实上,从目前云数据库产品中的数据模型设计方式来看,已经有些产品(比如 SimpleDB,Hbase,Dynamo,BigTable)放弃传统的行存储方式,而采用键/值存储,从而可以在分布式的云环境中获得更好的性能.可以预期的是,云数据库将会吸引越来越多的学术界的目光,该领域的相关问题也将成为未来一段时间内数据库研究的重点内容,比如云数据库的体系架构和数据模型等等.

(3) 数据库市场份额面临重新分配.在过去的几十年里,数据库市场一直被诸如 Teradata,Oracle,IBM DB2, Microsoft SQL Server,Sybase 等传统数据库厂商所垄断.随着云数据库的出现和不断发展,市场将面临重新洗牌.首先,Amazon 和 Google 等原本并不从事数据库业务的国际知名企业,也乘着云计算的东风,开发了云中的数据库产品,加入这场新兴市场的角逐.实际上,对于云数据库市场而言,Amazon SimpleDB 和 Google BigTable 这类产品扮演了引领者的角色,传统的数据库厂商已经成为跟进者;其次,一些新的云数据库厂商开始出现,并且推出了具有影响力的产品,比如 Vertica 的 Analytic Database for the Cloud 和 EnterpriseDB 的 Postgres Plus in the Cloud.因此,数据库市场份额的重新分配不可避免.

2 云数据库产品

云数据库供应商主要分为 3 类:

- 传统的数据库厂商:Teradata,Oracle,IBM DB2 和 Microsoft SQL Server;
- 涉足数据库市场的云供应商:Amazon,Google 和 Yahoo;
- 新兴小公司:Vertica,LongJump 和 EnterpriseDB.

就目前阶段而言,虽然一些云数据库产品,如 Google BigTable,SimpleDB 和 HBase,在一定程度上实现了对于海量数据的管理,但是这些系统暂时还不完善,只是云数据库的雏形.让这些系统支持更加丰富的操作以及更加完善的数据管理功能(比如复杂查询和事务处理)以满足更加丰富的应用,仍然需要研究人员的不断努力.

表 1 给出了目前市场上常见的云数据库产品,对于其中一些主要产品,下面我们会作简要介绍.

Table 1 Cloud database products

表 1 云数据库产品

企业	产品
Amazon	Dynamo, SimpleDB, RDS
Google	BigTable, FusionTable
Microsoft	Microsoft SQL Server Data Services 或 SQL Azure
Oracle	Oracle Cloud
Yahoo!	PNUTS
Vertica	Analytic Database v3.0 for the Cloud
EnerpriseDB	Postgres Plus in the Cloud
开源项目	Hbase, Hypertable
其他	EnerpriseDB, FathomDB, ScaleDB, Objectivity/DB, M/DB:X

2.1 Amazon的云数据库产品

Amazon 是云数据库市场的先行者.Amazon 除了提供著名的 S3 存储服务和 EC2 计算服务以外,还提供基于云的数据库服务 Dynamo[14].Dynamo 采用“键/值”存储,其所存储的数据是非结构化数据,不识别任何结构化数据,需要用户自己完成对值的解析.Dynamo 系统中的键(key)不是以字符串的方式进行存储,而是采用 md5_key(通过 md5 算法转换后得到)的方式进行存储,因此,它只能根据 key 去访问,不支持查询.SimpleDB 是 Amazon 公司开发的一个可供查询的分布数据存储系统,它是 Dynamo“键/值”存储的补充和丰富.顾名思义,SimpleDB 的目的是作为一个简单的数据库来使用,它的存储元素(属性和值)是由一个 id 字段来确定行的位置.这种结构可以满足用户基本的读、写和查询功能.SimpleDB 提供易用的 API 来快速地存储和访问数据.但是,SimpleDB 不是一个关系型数据库,传统的关系型数据库采用行存储,而 SimpleDB 采用了“键/值”存储,它主要是服务于那些不需要关系数据库的 Web 开发者.

Amazon RDS(Amazon relational database service)是 Amazon 开发的一种 Web 服务,它可以让用户在云环境中建立、操作关系型数据库(目前支持 MySQL 和 Oracle 数据库).用户只需要关注应用和业务层面的内容,而不需要在繁琐的数据库管理工作上耗费过多的时间.

此外,Amazon 和其他数据库厂商开展了很好的合作,Amazon EC2 应用托管服务已经可以部署很多种数据库产品,包括 SQL Server,Oracle 11g,MySQL 和 IBM DB2 等主流数据库平台,以及其他一些数据库产品,比如

EnerpriseDB.作为一种可扩展的托管环境,开发者可以在 EC2 环境中开发并托管自己的数据库应用.

2.2 Google的云数据库产品

Google BigTable 是一种满足弱一致性要求的大规模数据库系统.Google 设计 BigTable 的目的,是为了处理 Google 内部大量的格式化及半格式化数据.目前,许多 Google 应用都是建立在 BigTable 上的,比如 Web 索引、Google Earth、Google Finance、Google Maps 和 Search History.文献[15]描述了 BigTable 提供的简单数据模型,它允许客户端对数据部署和格式进行动态控制,并且描述了 BigTable 的设计和实现方法.BigTable 是构建在其他几个 Google 基础设施之上的:首先,BigTable 使用了分布式 Google 文件系统 GFS(Google file system)[16]来存储日志和数据文件;其次,BigTable 依赖一个高可用的、持久性的分布式锁服务 Chubby[17];再次,BigTable 依赖一个簇管理系统来调度作业、在共享机器上调度资源、处理机器失败和监督机器状态.

但是,与 Amazon SimpleDB 类似,目前来说,BigTable 实际上还不是真正的 DBMS(database management system),它无法提供事务一致性、数据一致性.这些产品基本上可以被看成是云环境中的表单.

Google 开发的另一款云计算数据库产品是 Fusion Tables[18].它采用了基于数据空间的技术.该技术在上世纪 90 年代就已经出现.Google 充分利用了该技术的潜力.Fusion Tables 是一个与传统数据库完全不同的数据库,可以弥补传统数据库的很多缺陷.比如通过采用数据空间技术,它能够简单地解决 RDBMS 中管理不同类型数据的麻烦,以及排序整合等常见操作的性能问题.Fusion Tables 可以上传 100MB 的表格文件,同时支持 CSV 和 XLS 格式,并且具有处理大规模数据的能力.

2.3 Microsoft的云数据库产品

2008 年 3 月,微软通过 SQL Data Service(SDS)提供 SQL Server 的 RDBMS 功能,这使得微软成为云数据库市场上的第一个大型数据库厂商.此后,微软对 SDS 功能进行了扩充,并且重新命名为 SQL Azure.微软的 Azure 平台提供了一个 WEB 服务集合,可以允许用户通过网络在云中创建、查询和使用 SQL SERVER 数据库,云中的 SQL SERVER 服务器的位置对于用户而言是透明的.对于云计算而言,这是一个重要的里程碑.SQL Azure 具有以下特性:

- 属于关系型数据库:支持使用 TSQL(transact structured query language)来管理、创建和操作云数据库;
- 支持存储过程:它的数据类型、存储过程和传统的 SQL Server 具有很大的相似性,因此,应用可以在本地进行开发,然后部署到云平台上;
- 支持大量数据类型:包含了几乎所有典型的 SQL Server 2008 的数据类型;
- 支持云中的事务:支持局部事务,但是不支持分布式事务.

2.4 开源云数据库产品

HBase[19]和 Hypertable 利用开源 MapReduce 平台 Hadoop,提供了类似于 BigTable 的可伸缩数据库实现.MapReduce[20]是 Google 开发的、用来运行大规模并行计算的框架.采用 MapReduce 的应用更像一个人提交的批处理作业,但是这个批处理作业不是在单个服务器上运行,应用和数据都是分布在多个服务器上.Hadoop 是由 Yahoo 资助的一个开源项目,是 MapReduce 的开源实现,从本质上来说,它提供了一个使用大量节点来处理大规模数据集的方式.

HBase 已经成为 Apache Hadoop 项目的重要组成部分,并且已经在生产系统中得到应用[5].与 HBase 类似的是 Hypertable.不过,HBase 的开发语言是 Java,而 Hypertable 则采用 C/C++开发.与 HBase 相比,Hypertable 具有更高的性能.但是,HBase 不支持 SQL(structual query language)类型的查询语言.

甲骨文开源数据库产品 BerkelyDB 也提供了云计算环境中的实现.

2.5 其他云数据库产品

Yahoo! PNUTS[21]是一个为网页应用开发的、大规模并行的、地理分布的数据库系统,它是 Yahoo!云计算平台重要的一部分.Vertica Systems 在 2008 年发布了云版本的数据库.10Gen 公司的 Mongo、AppJet 的 AppJet

数据库也都提供了相应的云数据库版本.M/DB:X 是一种云中的 XML 数据库,它通过 HTTP/REST 访问.FathomDB 旨在满足基于 Web 的公司提出的高传输要求,它所提供的服务更倾向于在线事务处理而不是在线分析处理.IBM 投资的 EnerpriseDB 也提供了一个运行在 Amazon EC2 上的云版本.LongJump 是一个与 Salesforce.com 竞争的新公司,它推出了基于开源数据库 PostgreSQL 的云数据库产品.Intuit QuickBase 也提供了自己的云数据库系列.麻省理工学院研制的 Relational Cloud[22]可以自动区分负载的类型,并把类型近似的负载分配到同一个数据节点上,而且采用了基于图的数据分区策略,对于复杂的事务型负载也具有很好的可扩展性.此外,它还支持在加密的数据上运行 SQL 查询.

3 云数据库领域的研究问题

对于学术界而言,要想在云数据库中提供类似于现有 DBMS 的丰富功能,比如查询、索引和事务处理,仍然有许多亟待解决的问题.云数据库领域中的研究问题主要包括:云数据库中数据模型设计、编程模型、服务器体系架构设计、事务一致性、基于云数据库的容灾和 SLA(service level agreement)监控、云数据的访问控制和授权管理、云应用数据访问体系的调优、云数据生命周期管理、云数据库与本地数据库的协同和联邦设计、测试基准等.下面将介绍一些典型问题的研究现状.

3.1 数据模型

云数据库的设计可以采用不同的数据模型,不同的数据模型可以满足不用应用类型的需求,主要包括:键/值模型和关系模型.

3.1.1 键/值模型

BigTable,Dynamo,SimpleDB,PNUTS,HBase 等产品都采用了键/值模型存储数据.

下面我们以 Google BigTable 的数据模型为例来介绍键/值模型.

Google BigTable 的数据模型:BigTable 和它的同类开源产品 HBase,提供了一个不同于以往的简单的、动态的、非关系型的数据模型.BigTable 采用了键/值数据模型.在 BigTable 中,包括行列以及相应的时间戳在内的所有数据都存放在表格的单元里.BigTable 的内容按照行来划分,多个行组成一个小表(Tablet),保存到某一个服务器节点中.这就意味着,每个 Tablet 包含了位于某个区间内的所有数据.对于 BigTable 而言,一个数据簇中存储了许多表,其中每个表都是一个 Tablet 集合.在最初阶段,每个表只包含 1 个 Tablet.随着表的增长,它会被自动分解成许多 Tablet,每个 Tablet 默认尺寸大约是 100MB~200MB.BigTable 使用一个类似于 B+树的 3 层架构来存储 Tablet 位置信息[11].由于 BigTable 采用了键/值数据模型,因此不存在表间的联接操作,这也使得数据分区操作相对简单,只需要根据键的区间来划分即可.

一个 BigTable 实际上就是一个稀疏的、分布的、永久的多维排序图,它采用行键盘(row key)、列键(column key)和时间戳(timestamp)对图进行索引.图中的每个值都是未经解释的字节数组[15]:

- 行键:BigTable 在行键上根据字典顺序对数据进行维护.对于一个表而言,行区间是根据行键的值进行动态划分的.每个行区间称为一个 Tablet,它是负载均衡和数据分发的基本单位,这些 Tablet 会被分发到不同的数据服务器上.
- 列键:被分组成许多“列家族”的集合,它是基本的访问控制单元.存储在一个列家族当中的所有数据,通常都属于同一种数据类型,这通常意味着具有更高的压缩率.数据可以被存放到列家族的某个列键下面,但是在把数据存放到这个列家族的某个列键下面之前,必须首先创建这个列家族.在创建完成一个列家族以后,就可以使用同一个家族当中的列键.
- 时间戳:在 BigTable 中的每个单元格当中都包含相同数据的多个版本,这些版本采用时间戳进行索引.BitTable 时间戳是 64 位整数.一个单元格的不同版本是根据时间戳降序的顺序进行存储的,这样,最新的版本可以被最先读取.

这里以文献[15]中的一个实例来阐释 BigTable 的数据模型.图 2 显示了存储了网页数据的 WebTable 的一个片段.行名称是反转的 URL,contents 列家族包含了网页内容,anchor 列家族包含了任何引用这个页面的

anchor 文本.CNN 的主页被 Sports Illustrated 和 MY-look 主页同时引用,因此,这里的行包含了名称为“anchor:cnnsi.com”和“anchor:my.look.ca”的列.每个 anchor 单元格都只有 1 个版本,contents 列有 3 个版本,分别对应于时间戳 t_3,t_5 和 t_6.

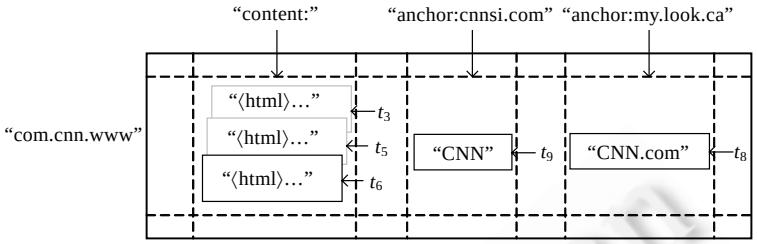

Fig.2 An example of the data model of BigTable

图 2 BigTable 数据模型的一个实例

HBase[19]和 BigTable 一样,也采用了键/值数据模型,它是一个开放源码、分布式、面向列、多维、高可用、高性能的存储技术,采用 JAVA 语言编写.作为一个使用了 Hadoop 的分布式数据库,HBase 可以实现结构化数据的可靠存储.就像 Google BigTable 充分利用 Google File System 提供的分布式数据存储功能一样,HBase 的目的就是在 HDFS(hadoop distributed file system)上提供类似 BigTable 的功能.HBase 采用多层索引表来执行键/值映射,获得了优越的主键查询性能.

Hbase,BigTable 中的数据库模式和关系型模式有很大的区别:第一,不存在表间的联接操作;第二,整个模式也只有 1 个索引——行键.与 RDBMS(relational database management system)不同的是,开发者不需要使用 WHERE 从句的等价表达形式.通过设计,HBase 中的所有访问方法或者通过行键访问,或者通过行键扫描,从而使得整个系统不会慢下来.由于 HBase 位于 Hadoop 框架之上,MapReduce 就可以用来生成索引表[19].

HBase 列家族可以被配置成支持不同类型的访问模式.一个家族也可以被设置成放入内存当中,以消耗内存为代价,从而换取更好的响应性能.

此外,Amazon Dynamo,SimpleDB 也都和 BigTable 一样采用了键/值存储.SimpleDB 中包含 3 个概念:domain,item 和 attribute.其中,domain 相当于一个 table;item 相当于一行;attribute 相当于一列,一列可以有多个值.但是,SimpleDB 和 BigTable 在数据划分的方式上存在一些差别:

- SimpleDB 的数据划分:采用静态数据划分方法,利用哈希函数把数据分发到多个数据节点,这使得 SimpleDB 更像一个哈希系统.这种方法的优点是实现难度小;但是缺点也很明显,即用户的一些 domain 存放不连续.为了降低不连续数据存放对用户查询性能带来的负面影响,SimpleDB 对用户 domain 的大小进行限制,比如一个 domain 大小不超过 10GB.
- BigTable 的数据划分:采用动态划分方法,一个用户的数据可能会被划分成多个 Tablet,分发到不同的数据节点上.这种方法的优点是可以很好地支持负载均衡,缺点是需要相关的 Tablet 分拆与合并机制.

BigTable,Dynamo,SimpleDB,PNUTS,HBase 等产品虽然都采用了键/值模型存储数据,但是这些系统在进行数据访问的时候都是以单个键作为访问粒度,这种方式对于一些网页应用而言无法取得好的性能,比如在线游戏、社区网络、合作编辑等包含合作性质的应用,这些应用需要对一个键组(key group)进行一致性的访问.由此,文献[23]设计了名为 G-Store 的键/值系统,并提出“键组协议”来对一组键进行一致的、高效的访问.也就是说,在该系统中,数据访问的粒度不再是单个的键,而是一个键组.

3.1.2 关系模型

微软的 SQL Azure 云数据库采用了关系模型,它的数据分区方式和 BigTable 有些不同.关系型云数据库的数据模型涉及行组和表组等相关概念.

一个表是一个逻辑关系,它包含一个分区键,用来对表进行分区.具有相同分区键的多个表的集合称为表组.在表组中,具有相同分区键值的多个行的集合称为行组.一个行组中包含的行总是被分配到同一个数据节点

上.每个表组会包含多个行组,这些行组会被分配到不同的数据节点上.一个数据分区包含了多个行组.因此,每个数据节点都存储了位于某个分区键值区间内的所有行.

这里以一个实例来解释关系型云数据库的数据模型.如图 3 所示,一个表组包含了两个相关的表,即图 3 中的表 1 和表 2.表 1 和表 2 分别包含了一个分区键列,每个分区键列是由多个分区键值组成的,这两个分区键列具有相同类型的分区键,都是 ID 类型的数值型数据.表 1 中的 ID 列和表 2 中的 ID 列存在主外键关联.因此,表 1 中和表 2 中的 ID 列值为 27 的 3 个行(图中用灰色底纹表示,并且用虚线框包围的部分)就属于一个行组.从图中也可以看出,表组被分割成多个分区.而且,同一行组中的内容都位于同一个数据分区内.

一个表组通常包含那些会被某个应用同时使用的多个表,因此需要把这些表存储在一起,从而加快查询效率.比如,有两个表 *STUDENT_INFO* 和 *BOOKBORROWING_INFO*,前者记录的信息包括名字、性别、年龄、电话等等,后者记录了借书的相关信息.当图书馆管理员查询一个人的借书记录时,通常需要同时查看借阅者姓名、年龄、电话、借书数量、借书时间等信息,这就需要在 *STUDENT_INFO* 和 *BOOKBORROWING_INFO* 这两个表之间进行联接操作.显然,如果把这两个表存储在同一个数据节点上,联接操作则会快很多.

类似地,把一个行组中的多个行存储在一个数据分区内也具有明显的好处.比如,图 3 中表 1 和表 2 中的 ID 为 27 的的多个行构成一个行组,它们之间存在主外键关联,把它们存放在一起可以加快查询的效率.

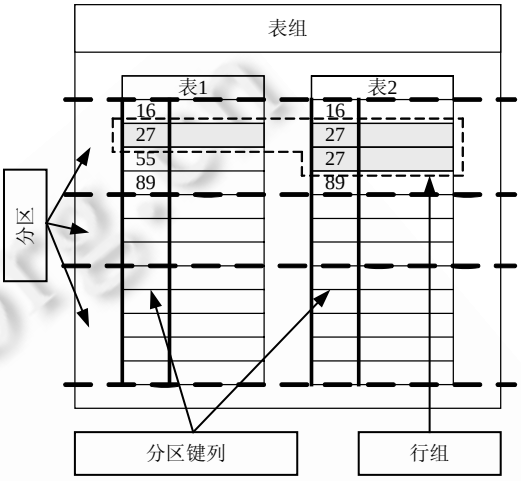

Fig.3 Data model of relational cloud databases

图 3 关系型云数据库的数据模型

3.1.3 支持多种数据模型

SQL Azure 等云数据库产品采用了关系模型,BigTable 等产品采用了键/值模型,而文献[24]中描述的 CloudDB 则可以同时支持关系模型和键/值模型,甚至可以采用列式存储(columnar store)[25].CloudDB 可以同时维护 3 种不同类型的数据存储,从而有效处理各种不同类型的负载.在 CloudDB 中,针对一个具体的应用而言,采用何种存储类型,取决于应用环境、负载类型特点和 SLA 需求等因素.

3.2 系统体系架构

这里主要讨论在存储层面采用非共享架构的云数据库.我们将阐述其体系架构和数据访问方法,并介绍几个系统实例.

3.2.1 体系架构

这里将以 HBase 和 SQL Azure 为例,分别介绍采用键/值数据模型和关系数据模型的具有代表性的云数据库体系架构.

3.2.1.1 采用键/值数据模型的云数据库体系架构

HBase[19]作为 BigTable 的一个开源实现,基本采用了和 BigTable 类似的架构.如图 4 所示,HBase 体系架构中包括 Client,Zookeeper,Hmaster,HRegionServer 和 Store,具体功能如下[26]:

- Client:访问 HBase 的接口.
- Zookeeper:存储了 HBase 的数据库模式和所有 HRegion 的寻址入口,实时监控 HRegionServer 的状态.
- HMaster:管理用户对 Table 的增、删、改、查操作,管理 HRegionServer 的负载均衡,调整 Region 分布等.
- HRegionServer:负责响应用户 I/O 请求,向 HDFS 文件系统中读写数据,是 HBase 中最核心的模块.
- Store:是 HBase 存储的核心,由 MemStore 和 StoreFiles 两部分组成.用户写入的数据首先会放入 MemStore,当 MemStore 满了以后会被存放到一个 StoreFile 中,StoreFile 将被存放在 HDFS 文件系统的

HFile 中.

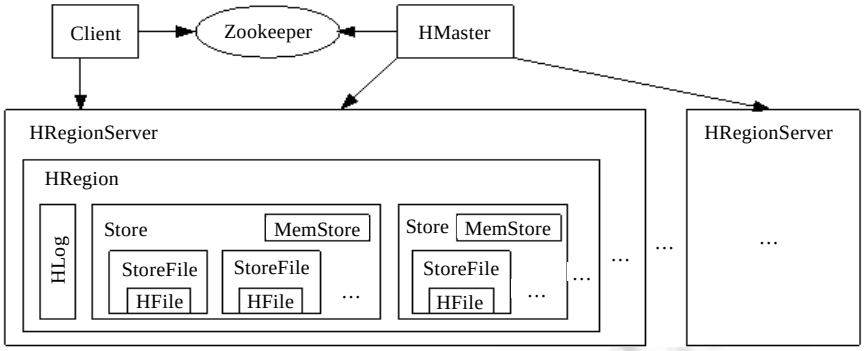

Fig.4 Architecture of HBase

图 4 HBase 的体系架构

3.2.1.2 采用关系数据模型的云数据库体系架构

如图 5 所示,SQL Azure 的体系架构[27]中包含了一个虚拟机簇,可以根据工作负载的变化,动态增加或减少虚拟机的数量.每台虚拟机 SQL Server VM(virtual machine)安装了 SQL Server 2008 数据库管理系统,以关系模型存储数据.通常,一份数据库会被散存储到 3 台~5 台 SQL Server VM 中.每台 SQL Server VM 同时安装了 SQL Azure Fabric 和 SQL Azure 管理服务,后者负责对每个数据库间的数据复写工作,以保障 SQL Azure 的基本高可用性要求.不同 SQL Server VM 内的 SQL Azure Fabric 和管理服务之间会彼此交换监控信息,以保持整体服务的可监控性.

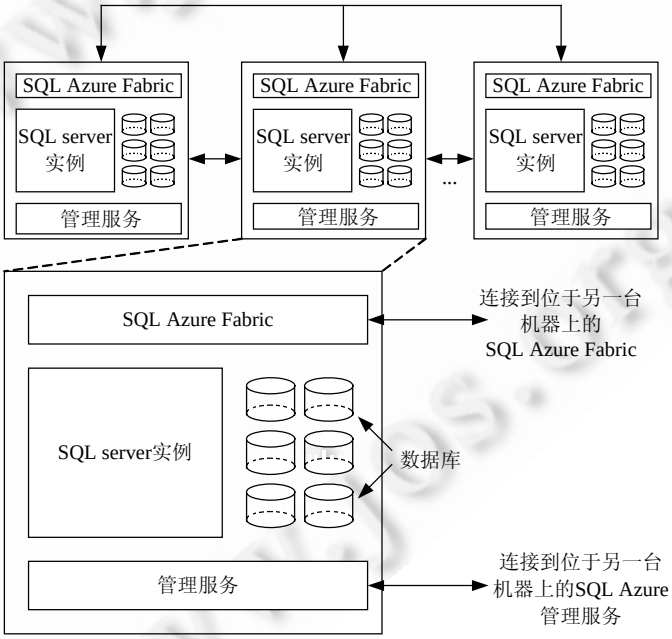

Fig.5 Architecture of SQL Azure

图 5 SQL Azure 的体系架构

3.2.2 数据访问方法

这里以一个实例来阐释云数据库的数据访问方法.如图 6 所示,当客户端请求数据时,它首先向管理器请求一份分区映射图,管理器向客户端发送分区映射图;客户端收到以后,在图中进行搜寻,根据键值找到自己所需数据的存储位置;然后,客户端到指定的数据节点请求数据;最后,由该数据节点把数据返回给客户端.实际上,为了改进性能,同时也为了避免管理器的性能瓶颈,通常会在客户端缓存常用的分区映射图.这样,客户端在很多情况下不必与管理器交互就可以直接访问相应的数据节点.

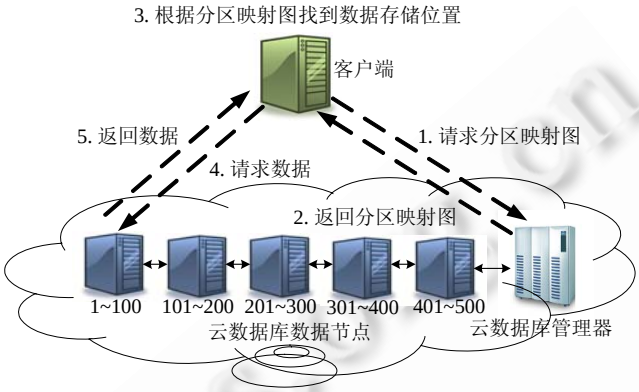

Fig.6 Data accessing method in cloud databases

图 6 云数据库中的数据访问方法

3.2.3 系统实例

在 BigTable 中,数据节点服务器称为 Tablet 服务器,管理器称为主服务器.主服务器负责把 Tablet 分配到 Tablet 服务器,探测 Tablet 服务器的信息,进行 Table 服务器的负载均衡,以及 GFS 文件系统中的垃圾收集(数据分区被存储在 GFS 文件中).此外,它还处理模式变化,比如表和列家族创建.就像许多单服务器分布式存储系统一样,客户端并不是直接从主服务器读取数据,而是直接从 Tablet 服务器上读取数据.因为客户端函数库会缓存 Tablet 位置信息,所以 BigTable 客户端并不依赖于主服务器来获得 Tablet 的位置信息,因而大多数客户端从来不和主服务器通信,这就使得在实际应用中主服务器的负载很小.

HBase[19]采用了和 BigTable 类似的实现方法,不同的是,它的数据分区存储在 HDFS 中.一个数据分区是由一个表中排序的行构成的,从而使得所有的表行都包含在一个数据分区的集合中.HBase 依赖于 Zookeeper (Zookeeper 和 BigTable 一样,依赖于 Chubby[17])来保证在任何时候都有总有一个主服务器可以用来存储 HBase 数据的位置引导信息,这些信息可以帮助发现数据分区服务器的位置.采用单个主服务器的设置很简单,但是可能会导致较差的可用性.

HugeTable[5]是一个分布式结构化数据库系统,它采用了标准的 SQL 语言查询接口,支持高性能的全局索引,并且在设计上采用了多主服务器机制、TabletServer 持续服务保证以及可靠的 Zookeeper 系统,已经被完全避免了类似 HBase 系统中的单点失效问题.HugeTable 的基本框架包含 4 个层次:应用层、ODBC/JDBC 驱动层、SQL 服务层和 Hadoop 云基础层.在应用层,HugeTable 支持朴素应用接口(它支持使用批更新的直接数据访问)和标准的 SQL 接口.ODBC/JDBC 驱动层就位于应用层的下面,它为上面的 SQL 应用提供了 JDBC/ODBC 驱动.SQL 服务层会分析所有到达的 SQL 语句,并生成相应的并行查询过程.Hadoop 云基础层包括一些基本的系统架构组件,比如 MapReduce[20],DFS,Zookeeper 和 HBase[19],HugeTable 依赖这些组件来实现自己的功能.

3.3 事务一致性

2000 年,UC Berkeley 的 Brewer 教授提出了著名的 CAP 理论[28],后来,Gilbert 和 Lynch 两人证明了 CAP 理论的正确性[29].所谓的 CAP 指的是:

- C: Consistency(一致性):任何一个读操作总是能读取到之前完成的写操作结果;
- A: Availability(可用性):每一个操作总是能够在确定的时间内返回;
- P: Tolerance of network Partition(分布性):在出现网络分区的情况下,仍然能够满足一致性和可用性.

CAP 理论告诉我们,一个分布式系统不可能同时满足一致性、可用性和分区容错性这 3 个需求,最多只能同时满足 2 个.

因此,根据 CAP 理论,我们在设计一个系统时可以有几种选择:

- 第 1 种:放弃分布性.把所有与事务相关的东西都放置到同一台机器上.显然,这种方法可扩展性很差.
- 第 2 种:放弃可用性.比如,可以把所有的数据都放置到一个节点上,当其他节点存在访问请求时,都被转向到该节点.显然,这方方案存在明显的单点性能瓶颈问题.
- 第 3 种:放弃一致性.由于不同用户对数据的一致性要求不一样,当用户的这种要求不高时,就要果断考虑放弃一致性.

传统的数据库提供强一致性保证,可以保证调用时序的一致性.在分布式的云环境中,维护事务的一致性具有很大的难度,通常代价很高.很多现有的云数据库产品在实现时,为了获得 CAP 理论中的可用性和分布性,都放松了对事务 ACID(atomicity,consistency,isolation,durability)四性的要求.比如,Google BigTable 就弱化了事务的原子性要求,只支持单行事务,可以允许对存储在某个行键下面的数据执行原子的“读-修改-写”操作.BigTable 当前不支持通用的跨行键的事务,虽然它在客户端提供了跨行键批量写入数据的接口[11].类似地,Amazon SimpleDB 等都没有实现通用的事务[30].比如,Amazon SimpleDB 放松了对事务的一致性和隔离性的要求,转而采用最终一致性(用户 *B* 虽然不能立即看到用户 *A* 修改的信息,但是用户 *B* 最终会看到用户 *A* 的修改信息),使得所有副本不必都获得数据副本的当前最新值.但是,SimpleDB 可以支持强一致读以及基于条件更新或者删除的乐观锁机制,并提供了简单的 SQL Select 子集支持.Amazon Dynamo[14]也采用了最终一致性.Yahoo! PNUTS 考虑到大部分 WEB 应用对一致性并没有非常严格的要求,因此在设计上放弃了对强一致性的追求(采用时间线一致性),转而追求更高的可用性、容错能力和更快的响应等[21].

文献[31]提出了一种基于树的一致性方法,并为云数据库引入了部分一致性和完整一致性的概念,从而有效减少不同数据副本之间的依赖性,极大地降低了事务失败的概率.

Chohan 等人[32]认为,支持对分布在不同数据库中的多行数据进行并发式地原子更新(要么全部更新,要么都不更新),可以极大推动云数据库的商业化应用.因此,该文针对键/值模型的云数据库提出了称为 DAT (database agnostic transactions)的事务处理方法.该方法采用了一个单独的、独立于数据库的软件层,并采用了 Zookeeper 服务(云计算和其他分布式环境中使用的一种开源的锁服务),只需要底层的云数据库提供行级别的原子更新支持,就可以实现针对多行数据更新的原子性、一致性、隔离性和持久性.Chohan 等人在 HBase 和 HyperTable 等云数据库产品上对 DAT 进行了事务处理性能的测试,取得了较好的效果.

实际上,分布式环境下的事务一致性并不是云数据库才面临的问题,云存储系统也同样存在该问题.文献[33]研究了在 Amazon S3 中的一种新的事务模型,它不仅允许设计者在事务级水平定义数据的一致性保障水平,而且允许在运行时自动切换一致性保障水平,并且提出了许多技巧,它允许系统通过监督数据和收集数据时间统计信息来自动地调整一致性水平.相信这方面的研究会给云数据库的相关研究带来一定的启示.

事务型数据管理系统当中存在维护分布式环境下的事务一致性的开销,在分析型数据管理系统当中就没有这个问题.对于事务型的工作负载,一个 DBMS 可以从一个失败中恢复,而不会丢失任何数据;并且在分布式数据库环境中,即使在面临节点失效的情况下,系统也可以成功地提交事务,这其中涉及各种保障机制的设计和相关开销.对于分析型负载当中的只读查询,根本不包含需要提交的写事务,也不必担心节点失效时会丢失更新操作.因此,一个分析型 DBMS 只是一个非常简单的 DBMS,当查询涉及的某个节点失效时,不需要让查询重启.这也是分析型数据库管理系统更适合部署在云环境中的原因.

文献[34]提出把云环境下数据库中的事务服务从数据管理组件中单独分离出来.传统的DBMS都存在一个事务存储管理器,它包含相互紧密结合的 4 个组件:(1) 负责并发控制的锁管理器;(2) 负责恢复的日志管理器;

(3) 进行分批 I/O 处理的数据缓冲池;(4) 负责如何在磁盘上存放数据的访问控制方法.云计算时代的到来,产生了对事务服务和数据管理进行分离的需求.该文献把存储引擎分成两个部分,即事务组件(transactional component)和数据组件(data component).事务组件只在逻辑层面工作,即它知道事务以及事务的逻辑并发控制和恢复,但是并不知道的底层存储页(page)的结构(比如可能采用 B-树);数据组件知道物理存储结构,它可以支持面向记录的原子访问操作,但是不知道事务.这种分离方法支持在云数据库中实现比较灵活的事务处理.

3.4 编程模型

云数据库存储了海量数据,涉及到大量的数据运算,如果仍然采用传统的数据处理方法,则将无法充分发挥云环境的优势.因此,采用一种机制简单而又具有高可扩展性的编程方法就显得尤为重要.查询这种大型的数据集,可以采用 Google 发明的一种新的编程模型,称为 MapReduce[20].

MapReduce 编程模型用来处理跨越成百上千个机器的大规模数据集,该软件架构隐藏了并行计算、数据分布和失败处理的细节.它背后的思想是,在分布式环境下处理数据,总是包括两个基本步骤:(1) 映射所需的数据;(2) 对这些数据进行聚集(aggregate)操作.它让用户定义一个映射函数,这个映射函数把一个“key/value”对的集合转换成一个中间临时的“key/value”对的集合,然后使用一个 reduce 函数把所有那些与同一个键相关的中间临时值都进行合并.许多现实世界的任务都可以采用这个模型来表达.MapReduce 可以运行在大规模商用服务器簇上,并且具有很高的可扩展性[20].一个典型的 MapReduce 计算会在成千上万的机器上处理许多 TB 的数据.

但是,传统的关系型数据库和 MapReduce 对数据的处理方式存在很大的区别,因此需要把关系数据库中的一些操作转换成 MapReduce 操作,这其中一类典型的操作就是联接(join)操作.

总的来说,在 MapReduce 环境下执行两个关系的联接操作的方法如下[35]:假设关系 $R(A,B)$和 $S(B,C)$都存储在一个文件中.为了联接这些关系,必须把来自每个关系的各个元组都与一个 key 关联,这个 key 就是属性 B 的值.可以使用一个 Map 进程集合把来自 R 的每个元组(a,b)转换成一个 key-value 对,其中的 key 就是 b,值就是(a,R).注意,这里把关系 R 包含到 value 中.这样做使得我们可以在 Reduce 阶段,只把那些来自 R 的元组和来自 S 的元组进行匹配.类似地,可以使用一个 Map 进程集合把来自 S 的每个元组(b,c)转换成一个 key-value 对,key 是 b,value 是(c,S).这里把关系名字包含在属性值中,可以使得在 Reduce 阶段只把那些来自不同关系的元组进行合并.Reduce 进程的任务就是把来自关系 R 和 S 的具有共同属性 B 值的元组进行合并.这样,所有具有特定 B 值的元组必须被发送到同一个 Reduce 进程.假设使用 k 个 Reduce 进程.这里选择一个哈希函数 h,它可以把属性 B 的值映射到 k 个哈希桶,每个哈希值对应一个 Reduce 进程.每个 Map 进程把 key 是 b 的 key-value 对都发送到与哈希值 $h(b)$对应的 Reduce 进程.Reduce 进程把联接后的元组(a,b,c)写到一个单独的输出文件中.

虽然采用 MapReduce 可以带来很多收益,但并不是说可以不要任何代价.实际上,它有时也会影响到数据库其他方面的性能.文献[36]开展了一系列实验,对 Hadoop 和几个性能优秀的并行数据库进行性能比较.实验结果表明,对于一些分析型的负载,MapReduce 不仅没有性能优势,而且要比并行数据库慢 3.1 倍~6.5 倍.针对这个问题,文献[13]的研究发现,当云数据库中的数据节点数量众多(几千个节点),并且是经常需要动态分发资源的分析型应用时,MapReduce 可以带来很好的性能.

此外,类似 BSP(bulk synchronous parallel)[37]和 MapReduce Online[38]等并行编程框架也将在云数据库中发挥重要作用.BSP 包含了许多通过通信网络连接起来的处理器,每个处理器都有非常快速的局部内存,可以开展多线程计算.MapReduce Online 是一种针对 MapReduce 的改进框架.MapReduce 只能支持批作业处理,每个 MapReduce 任务的输出都会首先被存储到磁盘然后才被使用;而 MapReduce Online 可以对各种操作进行流水线化处理,提高了并行性,大量降低作业完成时间,提高了系统的利用率.

3.5 数据安全

云数据库的安全性也是影响其普及与应用的关键因素.云数据库虽然提供了对海量数据的存储,但同时也对数据的保密性和访问控制等安全问题提出了新的挑战:一方面,一些敏感的数据,比如企业财务数据、医疗机构的病例档案、政府机构的文件等,必须经过加密才能放到云数据库中;另一方面,针对众多的用户,必须设置不

同的访问控制级别,保证不同层次的用户只能授权访问限定范围内的数据.因此,如何在云数据库中对数据实行加密和应用不同等级的访问控制约束规则,就成为新的研究热点.

文献[12]提出了衡量云计算环境的各种资源的私有性的有效方法,并且通过采用基于元数据和特权访问控制的方法来实现对云数据库资源的可信授权访问.该文献为每个 VMM(virtual machine memory)中的授权策略和云数据库中的资源都构建成以图表示的特权链,用来进行访问和安全控制.

文献[10]重点研究了支持多个用户在加密云数据库上执行 SQL 查询,并允许数据库管理员对不同用户设置不同级别的访问权限.为了实现只有授权用户才能对数据进行访问,文中采用 KP-ABE(key-policy attribute based encryption)加密策略,即每条策略都和用户的解密密钥关联,分配给每个用户的访问策略就被内嵌到该用户的解密密钥中.因此,只有当一个用户的解密密钥中包含的访问策略允许该用户访问某个数据时,该用户才可以访问该数据.

此外,同态加密也是云数据库安全领域未来的重点研究方向.同态加密是基于数学难题的计算复杂性理论的密码学技术.对经过同态加密的数据进行处理得到一个输出,将这一输出进行解密,其结果与用同一方法处理未加密的原始明文数据得到的输出结果是一样的.同态加密不同于传统的数据加密,它允许在没有解密算法和解密密钥的条件下对加密的数据进行运算.

3.6 性能优化

云计算是一个虚拟机器环境,在虚拟机器环境中部署软件的一个公共模型是虚拟器具(virtual appliance)模型.一个虚拟器具是一个 VM(virtual machine)镜像,具有预装和配置的应用.配置一个应用,只需把这个 VM 镜像拷贝到一个物理机器上面,启动这个 VM,然后执行配置任务即可.随着虚拟化和云计算的日益普及,我们期望可以采用一种更加普遍的方式来提供数据库服务,也就是通过部署在云当中的数据库器具来提供数据库服务,这是云数据库的一种实现方式.例如,Amazon 在 EC2 平台上提供了 MySQL,ORACLE 和 Microsoft SQL Server 数据库服务.由于云环境中的负载模式是不断变化的,这就会导致各种资源的动态分配.如何在这种动态环境下获得最好的数据库性能,是一个值得研究的问题.

文献[39]讨论了在云中部署数据库器具的一些性能优化方面的挑战,并给出了相应的解决方案;并以一个在不同的数据库器件之间分配 CPU 处理能力的例子阐述了虚拟环境调优方案.文献[7]简单讨论了在云计算的虚拟化环境下,如何根据资源使用情况对数据库实例进行迁移,从而达到资源最大化利用,优化数据库服务性能.文献[40]提出了 Albatross,可以用较小的代价处理面向 OLTP 应用的云数据库中的数据库实例动态迁移问题,从而实现有效的负载均衡;Albatross 在迁移过程中可以记录数据库缓存和活跃事务状态,迁移结束后相关事务可以继续执行,从而使得对事务的影响最小化.文献[41]设计了名为“Dolly”的原型系统,它采用 VM 克隆技术和自定义代价模型,可以根据不同应用情况调整数据库配置策略.文献[42]研究了如何在云数据库中为多个用户所使用的资源进行智能化管理和分配,从而实现整体性能优化.该文提出了一个名为 SmartSLA 的系统,它包含两个模块:系统建模模块和资源分配决策模块.前者可以利用机器学习算法为不同资源分配模式构建代价模型,进行代价和收益评估;后者可以根据代价评估结果动态决定资源分配,使得整个云数据库系统实现收益最大化.文献[43]研究了在给定查询负载和满足 QoS(quality of service)的情况下,为不同的云数据库实例分配资源,实现系统性能的最优化.文中提出了两种方法,即白盒方法和黑盒方法.前者对负载所消耗的系统资源进行精细地评估,而后者则只进行粗略评估.文中把两种方法等价变换成多维装箱(bin-packing)问题,并使用通用约束求解器(generic constraint solver)来解决该问题.

3.7 测试基准

在云数据库逐渐走向成熟时,有很多企业开始考虑是否要将自己的企业应用搬到云环境中.由于云数据库市场存在种类繁多的相关产品,企业必须决定究竟要选择哪个厂商的产品.企业做出决定需要首先考虑的一个因素是,是否可以取得更好的性能和更低的代价.因此,这里就需要一个测试不同云数据库产品性能的测试基准(benchmark).

目前,已经有一些可以使用的测试基准,比如:

- Google's BigTable 采用的性能测试方法[15]:可以对那些不支持结构化查询语言的系统进行性能评估.
- Hadoop 采用的性能评估方法:可以对一些结构化查询的性能进行评估.
- Yahoo! Cloud Serving Benchmark (YCSB)[44]:支持在测试时采用不同类型负载的组合,包括插入、读取、更新和扫描等;该测试主要针对类似 PNUTS 的系统,因为这些系统主要提供对数据的在线读写访问.
- 其他:比如文献[36]采用的方法.

但是,上述性能测试方法都没有体现当采用不同的系统实现方法时对系统整体性能的影响.实际上,系统实现方法多种多样,比如存储体系架构、数据模型、一致性和可用性水平等等,都不尽相同.因此,文献[45]对多个系统采用不同的实现方法进行了详尽的性能指标测试,可以作为评估不同系统性能的参考.

文献[46]则采用 TPC-W 测试基准对采用不同架构(比如分区、复制、分布式控制和缓存等)的多种云数据库产品的性能进行了比较和分析,比较的产品包括 AWS MySQL,AWS MySQL/R,AWS RDS,AWS SimpleDB, MS Azure 等.总体而言,该文侧重于对事务型操作(比如读和更新操作)的性能分析,而没有针对 OLAP(on-line analytical processing)操作进行分析.

3.8 其他研究

文献[13]描述了如何在云环境中提供高效可扩展的数据库服务,并介绍了作者开发的一个新系统 epiC,它可以根据负载类型的需求,同时支持 OLAP 和 OLTP(on-line transaction processing)类型的存储.在 epiC 中,OLAP 查询是通过并行扫描的方式进行处理,而 OLTP 查询则采用索引和局部查询优化.文献[47]描述了 epiC 系统所采用的多维数据索引方法 RT-CAN,它可以大大改善查询的性能.

文献[48]重点研究了数据库之间的动态数据迁移技术,可以有效支持云数据库环境下的动态负载均衡.该文结合了按需拉取(on-demand pull)和异步推给(asynchronous push),从而实现在最小的时间间隔内迁移数据库实例.通过使用轻量级的同步,不仅极大地减小了数据迁移过程中的失败操作数量,也极大地减小了迁移操作所涉及的数据量.

数据质量也是云数据库在设计和应用中需要考虑的重要问题.数据质量问题一般都是由于脏数据(dirty data)造成的.这里所谓的脏数据,包括不一致性、不准确性、错误性、冗余性和过期数据.在云数据库这种海量数据环境下,更容易产生脏数据,这里主要有两个方面的原因:一是由于在大规模数据环境下,一个错误会传播到许多数据中;另一方面,在大规模数据环境下,数据不一致性的概率进一步增大.数据清洗是传统的处理脏数据的方法,但是在云数据库的海量数据环境下,数据清洗的代价很高,会影响系统的效率.为此,文献[49]没有采用基于清洗的脏数据处理方法,而是直接在脏数据上进行查询,同时能够取得很好的查询效率和质量.文中提出了针对云数据库的脏数据库的存储架构,把相似的元组分成多个簇,脏数据被分布到多个节点上.为了保证在脏数据上进行查询也可以取得好的性能,文中采用了 3 层索引结构:(1) 代表层:同一个簇中的元组都由一个元组来统一代表,查询直接作用于这个元组代表,从而极大地缩减了扫描真实数据的开销;(2) 数据索引层:为代表元组中的列建立索引,从而使得查询能够快速找到相应的元组;(3) 节点索引层:可以快速定位那些可能包括查询相关结果的节点.

文献[8]提出了采用数据分裂(data fission)和数据融合(data fusion)的方法,在云计算环境下设计可扩展的数据库管理体系架构.该文献采用低代价的数据库迁移来支持轻量级的可伸缩性,并且设计了全自动的控制器,避免了人工操作.

4 结束语及未来研究展望

云数据库伴随着云计算技术的成熟而迅速发展,并且越来越受到业界和学术界的关注.本文阐述了云数据库的特性及其影响,并根据大量调研结果对相关的云数据库产品进行了归类和介绍;针对围绕云数据库的一些争议,我们做了简要分析,并给出了结论;同时,我们列出了云数据库领域的相关研究.

综合过去两年对云计算和云数据库的关注和研究,我们认为,在未来几年,数据模型、体系架构、编程模型、

性能优化等方面将会有更多的研究成果出现.同时,以下问题是云数据及其相关研究领域的重点:

(1) 事务一致性.目前,只有 SQL Azure 等少数关系型云数据库能够支持严格的事务四性.BigTable 和 HBase 等采用了键/值模型的云数据库,都采用放松对 ACID 四性的要求,以保证系统的性能.事务型系统是企业应用中的一个重要部分,若要使云数据库很好地支持事务处理,必须设计相应机制,既能实现事务处理,又能够不对系统性能造成严重影响.类似于文献[34]中的事务服务和数据管理进行分离的方法,是一种可行的研究方向.

(2) 企业本地数据库和云数据库中数据的有效融合机制.经过几十年的应用,企业本地数据库存储了大量数据,纵然云数据库能够带来诸多好处,但是企业全面步入云数据库仍然需要一个过程,这期间,必然有一个长期的本地数据和云数据并存的局面.这就需要设计有效的机制,实现本地数据库和云数据库的有效融合,支持对两种数据库的统一管理.一种思路是,采用视图的机制,在逻辑层面对两种数据库进行整合.对于用户而言,看到的只是一个统一的逻辑数据视图,并不知道数据的物理存储地点;但是在物理层面,这些数据分布在企业本地数据库和云数据库中.如果采用视图机制,将会涉及数据模型的统一构建和来自不同数据源的数据的快速合并机制等方面的研究.

(3) 向云数据库中迁移数据.目前,大多数企业数据库都存储在企业内部的数据库里.在云数据库不断发展的过程中,如何把这些数据迁移到云中,是一个具有挑战性的问题.尤其是一些包含海量数据的科学数据库,可能无法直接迁移到云中,需要做一些模式和设置上的修改才可以.而且,科学数据库通常需要使用一些 DBMS 提供的高级管理功能,比如用户自定义函数和存储过程等,要在云数据库中实现所有这些高级功能,从目前来看还存在一定的困难.Thakar 等人[50]在一个基于 SQL SERVER 的天文数据库上进行了大量实验,结果表明,在当前技术条件下,把如此复杂的数据库迁移到云数据库中几乎是不可能实现的.因此,需要研究向云数据库中迁移数据的有效方法.这里需要重点研究企业已采用的不同数据模型(比如面向对象或者关系模型)到云数据库产品所采用数据模型(比如键/值模型)的转换.文献[51]在这个方面做了初步的探索与尝试,总结了一些困难挑战和可能的解决方法.

(4) 云数据库的容灾.云数据库的数据存储在分布式环境中,不同的云数据库产品可能采用不同的存储平台.比如:Oracle 云数据库托管在 Amazon EC2 平台上,使用了 Amazon S3 存储服务;而 SQL Azure 则可以直接使用微软自己的数据中心.这些云环境已经设计了相应的容灾机制,但它们都是针对通用的文件存储情形,而云数据库由于涉及事务处理,对故障的处理要更加复杂.因此,云数据库的容灾是一个需要重点研究的内容.一种比较新的尝试是,把灾难处理作为云环境中的一种服务提供给各种应用(包括云数据库)使用.

(5) 云数据库性能优化.云数据库同时为多个用户提供服务,不同用户的负载类型各不相同.如何根据用户的负载类型为不同用户合理分配资源,从而既能满足用户的处理要求,又能实现云数据库整体性能最大化,是一个需要继续深入研究的问题.我们认为,可以从 3 个方面开展相关研究:

- 研究智能资源管理模型,采用代价函数有效评估不同资源分配方法的代价,发现最优的资源分配方法.比如,文献[43]把这类问题转换成多维装箱问题,并使用通用约束求解器来解决,是一个可行的思路;再比如,文献[22]采用的方法也很典型,它可以自动区分负载的类型,并把类型近似的负载分配到同一个数据节点上,实现资源的合理分配.
- 研究结合经济效益的负载均衡方法.一种好的负载均衡方法,应该使得服务器总体开销最小化.具体来说,候选服务器节点可能来自企业内部,例如内部私有云;也可能来自企业外部,例如公共云计算服务供应商.当采用企业内部服务器时,不同的服务器显然具备不同的负载处理开销.当采用企业外部服务器资源时,比如 Amazon,Google 和 Microsoft 等公司提供的公共云计算资源,同样需要考虑价格因素.这些云计算资源采取租赁的方式,按照 CPU 和带宽使用情况计费.当企业对 IT 资源需求量增加时,可以随时租赁更多的 IT 资源;而当企业需求减少时,又可以及时退租多余的 IT 资源.通过这种方式,可以使得企业最小化 IT 建设和维护开销.不同云计算服务提供商的服务水平和租赁价格通常各不相同,因此在处理负载均衡问题时,为了实现开销最小化,无论使用内部服务器资源还是外部服务器资源,都需要考虑经济因素.而现有的负载均衡方法在选择目标服务器时无法考虑经济因素.我们认为,可以考虑采用基于

市场机制和 Agent 的负载均衡方法.市场机制是经济学领域非常成熟的资源分配方法,并且已经被计算机研究人员引入解决资源分配问题,但是还没有用市场机制解决云数据库负载均衡的相关研究.

- 研究动态数据迁移方法.云数据库的负载分布是一个动态变化的过程,为了保持性能最优,需要动态进行负载均衡和资源再分配.其中将涉及数据的动态迁移问题,即如何把数据从一个服务器节点迁移到另一个节点.数据迁移必须保证服务间断时间尽可能短,对系统性能的影响尽可能地小,这就对相关的算法提出了很高的效率要求.比如,文献[48]重点研究了数据库之间的动态数据迁移技术,可以有效地支持云数据库环境下的动态负载均衡.

(6) 云数据库测试基准.众多的云数据库产品可以满足不同类型的用户的应用需求.对于用户而言,在选择具体产品时,必须基于可靠的产品性能评测结果.因此,必须继续深入研究针对不同类型负载特点的测试基准,能够清晰呈现不同产品在不同负载类型下的性能变化.

(7) 云数据库相关标准.由于现阶段包括云数据库在内的各种云计算技术都是由各大厂商独立发展的,彼此都没有遵循统一的规范,这就造成了数据共享和迁移的诸多障碍,不利于网络数据应用的实现.当这个技术领域逐渐成熟时,将会出现各种云数据库标准,例如云数据库的 SQL 语言、云数据库的应用模型、云数据库的安全标准等等.比如,AtomPub 已经成为基于云的数据存储的事实访问标准,可以让不同的存储平台之间(包括云数据库)具有统一的访问方法.

随着市场的发展和云数据库的不断成熟,一些围绕云数据库的问题将逐渐得到解决,一些争议也将会有更加明晰的结果.可以说,云数据库是数据库技术在未来一段时间内的发展趋势,该领域的相关问题也将成为数据库研究的重点内容.云数据库的发展、成熟与应用需要学术界和业界人员的共同努力.

References:

[1] Chen K, Zheng WM. Cloud computing: System instances and current research. Journal of Software, 2009,20(5):1337−1348 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3493.htm [doi: 10.3724/SP.J.1001.2009.03493]

[2] Dash D, Kantere V, Ailamaki A. An economic model for self-tuned cloud caching. In: Ioannidis YE, Lee DL, Ng RT, eds. Proc. of the 25th Int'l Conf. on Data Engineering (ICDE 2009). New York: IEEE Computer Society Press, 2009. 1687−1693. [doi: 10.1109/ICDE.2009.143]

[3] Feng DG, Zhang M, Zhang Y, Xu Z. Study on cloud computing security. Journal of Software, 2011,22(1):71−83 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3958.htm [doi: 10.3724/SP.J.1001.2011.03958]

[4] Xu M, Gao D, Deng C, Luo ZG, Sun SL. Cloud computing boosts business intelligence of telecommunication industry. In: Jaatun MG, Zhao GS, Rong CM, eds. Proc. of the 1st Int'l Conf. on Cloud Computing (CloudCom 2009). Berlin: Springer-Verlag, 2009. 224−231. [doi: 10.1007/978-3-642-10665-1_20]

[5] Qi J, Qian L, Luo ZG. Distributed structured database system HugeTable. In: Jaatun MG, Zhao GS, Rong CM, eds. Proc. of the 1st Int'l Conf. on Cloud Computing (CloudCom 2009). Berlin: Springer-Verlag, 2009. 338−346. [doi: 10.1007/978-3-642-10665-1_31]

[6] Abouzeid A, Bajda-Pawlikowski K, Abadi DJ, Silberschatz A, Rasin A. HadoopDB: An architectural hybrid of MapReduce and DBMS technologies for analytical workloads. PVLDB, 2009,2(1):922−933.

[7] Ahrens M, Alonso G. Relational databases, virtualization, and the cloud. In: Abiteboul S, Böhm K, Koch C, Tan KL, eds. Proc. of the 27th Int'l Conf. on Data Engineering (ICDE 2011). New York: IEEE Computer Society Press, 2011. 1254. [doi: 10.1109/ICDE.2011.5767966]

[8] Agrawal D, Abbadi AE, Das S, Elmore AJ. Database scalability, elasticity, and autonomy in the cloud—(extended abstract). In: Yu JX, Kim MH, Unland R, eds. Proc. of the 16th Int'l Conf. on Database Systems for Advanced Applications (DASFAA 2011). Berlin: Springer-Verlag, 2011. 2−15. [doi: 10.1007/978-3-642-20149-3_2]

[9] Soares L, Pereira J. Improving the scalability of cloud-based resilient database servers. In: Felber P, Rouvoy R, eds. Proc. of the 11th IFIP WG 6.1 Int'l Conf. on Distributed Applications and Interoperable Systems (DAIS 2011). Berlin: Springer-Verlag, 2011. 136−149. [doi: 10.1007/978-3-642-21387-8_11]

[10] Ion M, Russello G, Crispo B. Enforcing multi-user access policies to encrypted cloud databases. In: Proc. of the IEEE Int'l Symp. on Policies for Distributed Systems and Networks (POLICY 2011). New York: IEEE Computer Society Press, 2011. 175−177. [doi: 10.1109/POLICY.2011.14]

[11] Chang F, Dean J, Ghemawat S, Hsieh WC, Wallach DA, Burrows M, Chandra T, Fikes A, Gruber RE. Bigtable: A distributed storage system for structured data. ACM Trans. on Computer Systems, 2008,26(2):1−26. [doi: 10.1145/1365815.1365816]

[12] Yoon JP. Access control and trustiness for resource management in cloud databases. In: Fiore S, Aloisio G, eds. Proc. of the Int'l Conf. on Grid and Cloud Database Management. Berlin: Springer-Verlag, 2011. 109−131. [doi: 10.1007/978-3-642-20045-8_6]

[13] Chen C, Chen G, Jiang DW, Ooi BC, Vo HT, Wu S, Xu QQ. Providing scalable database services on the cloud. In: Chen L, Triantafillou P, Suel T, eds. Proc. of the 11th Int'l Conf. on Web Information Systems Engineering (WISE 2010). Berlin: Springer-Verlag, 2010. 1−19. [doi: 10.1007/978-3-642-17616-6_1]

[14] 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: Bressoud TC, Kaashoek MF, eds. Proc. of the 21st ACM Symp. on Operating Systems Principles (SOSP 2007). New York: ACM Press, 2007. 205−220. [doi: 10.1145/1294261.1294281]

[15] Chang F, Dean J, Ghemawat S, Hsieh WC, Wallach DA, Burrows M, Chandra T, Fikes A, Gruber RE. Bigtable: A distributed storage system for structured data. In: Proc. of the 7th Symp. on Operating Systems Design and Implementation (OSDI 2006). Seattle: USENIX Association, 2006. 205−218.

[16] Ghemawat S, Gobioff H, Leung ST. The Google file system. In: Scott ML, Peterson LL, eds. Proc. of the 19th ACM Symp. on Operating Systems Principles (SOSP 2003). New York: ACM Press, 2003. 29−43. [doi: 10.1145/945445.945450]

[17] Burrows M. The Chubby lock service for loosely coupled distributed systems. In: Proc. of the 7th Symp. on Operating Systems Design and Implementation (OSDI 2006). Seattle: USENIX Association, 2006. 335−350.

[18] Gonzalez H, Halevy AY, Jensen CS, Langen A, Madhavan J, Shapley R, Shen WR, Goldberg-Kidon J. Google fusion tables: Web-centered data management and collaboration. In: Elmagarmid AK, Agrawal D, eds. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2010). New York: ACM Press, 2010. 1061−1066. [doi: 10.1145/1807167.1807286]

[19] Cryans JD, April A, Abran A. Criteria to compare cloud computing with current database technology. In: Dumke RR, Braungarten R, Büren G, Abran A, Cuadrado-Gallego JJ, eds. Proc. of the Int'l Conf. on Software Process and Product Measurement (IWSM/Metrikon/Mensura 2008). Berlin: Springer-Verlag, 2008. 114−126. [doi: 10.1007/978-3-540-89403-2_11]

[20] Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters. In: Proc. of the 6th Symp. on Operating Systems Design and Implementation (OSDI 2004). Seattle: USENIX Association, 2004. 137−150.

[21] 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. [doi: 10.1145/1454159.1454167]

[22] Curino C, Jones EPC, Popa RA, Malviya N, Wu E, Madden S, Balakrishnan H, Zeldovich N. Relational cloud: A database service for the cloud. In: Proc. of the 5th Biennial Conf. on Innovative Data Systems Research (CIDR 2011). 2011. 235−240. http://www.crdrdb.org

[23] Das S, Agrawal D, Abbadi AE. G-Store: A scalable data store for transactional multi key access in the cloud. In: Hellerstein JM, Chaudhuri S, Rosenblum M, eds. Proc. of the 1st ACM Symp. on Cloud Computing (SoCC 2010). New York: ACM Press, 2010. 163−174. [doi: 10.1145/1807128.1807157]

[24] Hacigümüs H, Tatemura J, Hsiung WP, Moon HJ, Po O, Sawires A, Chi Y, Jafarpour H. CloudDB: One size fits all revived. In: Proc. of the 6th World Congress on Services (SERVICES 2010). New York: IEEE Computer Society, 2010. 148−149. [doi: 10.1109/SERVICES.2010.96]

[25] Stonebraker M. Technical perspective—One size fits all: An idea whose time has come and gone. Communications of the ACM, 2008,51(12):76. [doi: 10.1145/1409360.1409379]

[26] The architecture of HBase. http://hbase.apache.org/book.html#architecture

[27] The architecture of SQL Azure. http://zh.wikipedia.org/wiki/SQL_Azure

[28] Browne J. Brewer's CAP theorem. 2009. http://www.julianbrowne.com/article/viewer/brewers-cap-theorem

[29] Gilbert S, Lynch NA. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant Web services. SIGACT News, 2002,33(2):51−59. [doi: 10.1145/564585.564601]

[30] Abadi DJ. Data management in the cloud: limitations and opportunities. IEEE Data Engineering Bulletin, 2009,32(1):3−12.

[31] Islam MA, Vrbsky SV. Tree-Based consistency approach for cloud databases. In: Proc. of the 2nd Int'l Conf. on Cloud Computing (CloudCom 2010). New York: IEEE Computer Society, 2010. 401−404. [doi: 10.1109/CloudCom.2010.87]

[32] Chohan N, Bunch C, Krintz C, Nomura Y. Database-Agnostic transaction support for cloud infrastructures. In: Liu L, Parashar M, eds. Proc. of the IEEE Int'l Conf. on Cloud Computing (CLOUD 2011). New York: IEEE Computer Society, 2011. 692−699. [doi: 10.1109/CLOUD.2011.111]

[33] 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.

[34] Lomet DB, Fekete A, Weikum G, Zwilling MJ. Unbundling transaction services in the cloud. In: Proc. of the 4th Biennial Conf. on Innovative Data Systems Research (CIDR 2009). 2009. 1−10. http://www.crdrdb.org

[35] Afrati FN, Ullman JD. Optimizing joins in a map-reduce environment. In: Manolescu I, Spaccapietra S, Teubner J, Kitsuregawa M, Léger A, Naumann F, Ailamaki A, Özcan F, eds. Proc. of the 13th Int'l Conf. on Extending Database Technology (EDBT 2010). New York: ACM Press, 2010. 99−110. [doi: 10.1145/1739041.1739056]

[36] Pavlo A, Paulson E, Rasin A, Abadi DJ, DeWitt DJ, Madden S, Stonebraker M. A comparison of approaches to large-scale data analysis. In: Çetintemel U, Zdonik SB, Kossmann D, Tatbul N, eds. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2009). New York: ACM Press, 2009. 165−178. [doi: 10.1145/1559845.1559865]

[37] BSP Worldwide. Co-Ordinating bulk synchronous parallel computing. http://www.bsp-worldwide.org

[38] Condie T, Conway N, Alvaro P, Hellerstein JM, Gerth J, Talbot J, Elmeleegy K, Sears R. Online aggregation and continuous query support in MapReduce. In: Elmagarmid AK, Agrawal D, eds. Proc. of the 2010 ACM SIGMOD Conf. on Management of Data (SIGMOD 2010). New York: ACM Press, 2010. 1115−1118. [doi: 10.1145/1807167.1807295]

[39] Aboulnaga A, Salem K, Soror AA, Minhas UF, Kokosielis P, Kamath S. Deploying database appliances in the cloud. IEEE Data Engineering Bulletin, 2009,32(1):13−20.

[40] Das S, Nishimura S, Agrawal D, Abbadi AE. Albatross: Lightweight elasticity in shared storage databases for the cloud using live data migration. Proc. of the VLDB Endowment, 2011,4(8):494−505.

[41] Cecchet E, Singh R, Sharma U, Shenoy PJ. Dolly: Virtualization-driven database provisioning for the cloud. In: Petrank E, Lea D, eds. Proc. of the 7th Int'l Conf. on Virtual Execution Environments (VEE 2011). New York: ACM Press, 2011. 51−62. [doi: 10.1145/1952682.1952691]

[42] Xiong PC, Chi Y, Zhu SH, Moon HJ, Pu C, Hacigümüs H. Intelligent management of virtualized resources for database systems in cloud environment. In: Abiteboul S, Böhm K, Koch C, Tan KL, eds. Proc. of the 27th Int'l Conf. on Data Engineering (ICDE 2011). New York: IEEE Computer Society, 2011. 87−98. [doi: 10.1109/ICDE.2011.5767928]

[43] Rogers J, Papaemmanouil O, Çetintemel U. A generic auto-provisioning framework for cloud databases. In: Proc. of the 4th Int'l ICDE Workshop on Ranking in Databases (DBRank 2010). New York: IEEE Computer Society, 2010. 63−68. [doi: 10.1109/ICDEW.2010.5452746]

[44] Cooper BF, Silberstein A, Tam E, Ramakrishnan R, Sears R. Benchmarking cloud serving systems with YCSB. In: Hellerstein JM, Chaudhuri S, Rosenblum M, eds. Proc. of the 1st ACM Symp. on Cloud Computing (SoCC 2010). New York: ACM Press, 2010. 143−154. [doi: 10.1145/1807128.1807152]

[45] Shi YJ, Meng XF, Zhao J, Hu XM, Liu BB, Wang HP. Benchmarking cloud-based data management systems. In: Meng XF, Chen Y, Xu JL, Lu JH, eds. Proc. of the 2nd Int'l CIKM Workshop on Cloud Data Management (CloudDB 2010). New York: ACM Press, 2010. 47−54. [doi: 10.1145/1871929.1871938]

[46] Kossmann D, Kraska T, Loesing S. An evaluation of alternative architectures for transaction processing in the cloud. In: Elmagarmid AK, Agrawal D, eds. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2010). New York: ACM Press, 2010. 579−590. [doi: 10.1145/1807167.1807231]

[47] Wang JB, Wu S, Gao H, Li JZ, Ooi BC. Indexing multi-dimensional data in a cloud system. In: Elmagarmid AK, Agrawal D, eds. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2010). New York: ACM Press, 2010. 591−602. [doi: 10.1145/1807167.1807232]

[48] Elmore AJ, Das S, Agrawal D, Abbadi AE. Zephyr: Live migration in shared nothing databases for elastic cloud platforms. In: Sellis TK, Miller RJ, Kementsietsidis A, Velegrakis Y, eds. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2011). New York: ACM Press, 2011. 301−312. [doi: 10.1145/1989323.1989356]

[49] Wang HZ, Li JZ, Wang JB, Gao H. Dirty data management in cloud database. In: Fiore S, Aloisio G, eds. Proc. of the Int'l Conf. on Grid and Cloud Database Management. Berlin: Springer-Verlag, 2011. 133−150. [doi: 10.1007/978-3-642-20045-8_7]

[50] Thakar A, Szalay AS, Church K, Terzis A. Large science databases—Are cloud services ready for them? Scientific Programming, 2011,19(2-3):147−159. [doi: 10.3233/SPR-2011-0325]

[51] Thakar A, Szalay AS. Migrating a (large) science database to the cloud. In: Hariri S, Keahey K, eds. Proc. of the 19th ACM Int'l Symp. on High Performance Distributed Computing (HPDC 2010). New York: ACM Press, 2010. 430−434. [doi: 10.1145/1851476.1851539]

附中文参考文献:

[1] 陈康,郑纬民.云计算:系统实例与研究现状.软件学报,2009,20(5):1337−1348. http://www.jos.org.cn/1000-9825/3493.htm [doi: 10.3724/SP.J.1001.2009.03493]

[3] 冯登国,张敏,张妍,徐震.云计算安全研究.软件学报,2011,22(1):71−83. http://www.jos.org.cn/1000-9825/3958.htm [doi: 10.3724/SP.J.1001.2011.03958]

林子雨(1978－),男,吉林柳河人,博士,讲师,CCF 会员,主要研究领域为数据库,实时主动数据仓库,数据挖掘.

谢怡(1979－),女,博士,讲师,CCF 会员,主要研究领域为网络性能优化和安全技术,系统建模与仿真,软件开发.

赖永炫(1981－),男,博士,讲师,主要研究领域为数据库,传感器网络数据管理.

邹权(1982－),男,博士,讲师,主要研究领域为生物信息学,大规模数据挖掘.

林琛(1982－),女,博士,讲师,CCF 会员,主要研究领域为数据挖掘,信息检索,社会网络分析.