云数据库应用研究

青 欣,胥光辉,戢 瑶,郭 霄
(解放军理工大学 指挥自动化学院,江苏 南京 210007)

摘 要:在信息大爆炸时代,每天产生着海量的数据,并且由这些数据产生更大规模的管理操作记录文件和系统日志文件,传统数据库已经不能满足如此庞大的数据管理要求。文中从实际出发分析了传统数据库面临的问题,提出用云数据库来解决问题。对云数据库技术的发展和现状进行了阐述,分析了当前主流云数据库的特点和优势,并采用 HBase 作为云数据库平台进行了实验研究。实验表明云数据库具有很好的可扩展性和容灾性,并且具有较高的读写性能,能满足海量数据的分布式应用需求。

关键词:数据库;云数据库;HBase
中图分类号:TP392 **文献标识码**:A **文章编号**:1673-629X(2013)05-0037-05
doi:10.3969/j.issn.1673-629X.2013.05.010

Research on Application of Cloud Database

QING Xin, XU Guang-hui, JI Yao, GUO Xiao
(Institute of Command Automation, PLAUST, Nanjing 210007, China)

Abstract: In the age of information explosion, huge amounts of data are produced every day. And these data generate more large-scale management operations log file and system log file. Traditional database can no longer meet such a large data management requirements. Analyze the traditional database problems, describe the development and current status of cloud database technology and analyze the characteristics and advantages of the current mainstream cloud database. Use HBase as a cloud database experiment platform and the experiments show that the cloud database has good scalability and tolerance, and has higher concurrent reading and writing performance. It meets the requirement in distributed application.

Key words: database; cloud database; HBase

0 引 言

在企业、学校和各类服务提供商的计算中心建设中,数据库的搭建具有重要的地位。而为了满足应用的需求,需要不断地提高和更新硬件设施,这是一笔巨大的开销。并且随着数据量的增加和服务请求的增长,传统数据库将会面临诸多问题[1]:

1)可扩展性差:传统数据不是为大规模可伸缩的分布式处理设计的,虽然也提供复制和分区的解决方案,但不能从根本上解决问题,并且非常难以安装和维护,甚至要牺牲一些传统 RDBMS(Relational DataBase Management System:关系型数据库)的重要特性,不满足弹性需求的要求;

2)海量数据条件下读写性能低下:当数据或并发用户超过某个数量级后,性能上会有明显下降,不能满足高并发读写的服务请求;

3)管理复杂困难:传统数据库的维护要求人员专业性强,管理人员要进行严格的培训,对数据的管理和维护复杂;

4)运行维护成本高:传统数据库很难进行升级和更新,当现有数据库不能满足应用需求的时候一般是全部采用新的更强大的硬件和新版本的软件,这样不仅需要巨大的开销,还会使数据库暂停服务,在很多场合这是不能容忍的。

1 云数据库技术的发展和优点

传统数据库在一定程度上满足了目前传统的应用需求,但是由于其自身的缺陷和信息技术的发展,特别是在云计算平台上海量数据的管理和应用的背景之下,云数据库成为新一代数据库的发展方向,研究云数据库具有重大的意义[2]。

云计算[3,4]按照服务类型大致可以分为三类[5]:IaaS(Infrastructure as a Service:基础设施即服务)、

收稿日期:2012-08-15;**修回日期**:2012-11-20
基金项目:国家重大专项项目(GFZX0404010503)
作者简介:青 欣(1986-),男,硕士研究生,CCF 会员,研究方向为云计算;胥光辉,副教授,硕士生导师,研究方向为软件工程、云计算。

PaaS(Platform as a Service:平台即服务)和 SaaS(Software as a Service:软件即服务)。云数据库是在 SaaS 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,消除了资源的重复配置,让软、硬件升级变得更加容易。云数据库具有高可扩展性、高可用性,采用多租户形式和支持资源有效分发等特点。可以说,云数据库代表着数据库技术未来发展的一种主流方向。目前,对于云数据库的概念定义不尽相同,文中云数据库定义是:云数据库是部署在云计算环境中的数据库。

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,所有的底层硬件和实现对客户端而言是透明的,它就像在使用一个运行在本地的数据库一样,非常方便简单,同时又可以获得理论上近乎无限的存储和处理能力。具有如下优点:

动态可扩展:理论上,云数据库具有无限可扩展性,可满足不断增加的数据存储需求。在面对不断变化的条件时,云数据库可表现出很好的弹性。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,其过程只需几分钟。一旦需求过去以后,就可立即释放这些资源。

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。

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,这也大大降低了用户开销。

易用性:使用云数据库的用户不用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的链接字符串就可以开始使用云数据库。

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。

2 主流的云数据库产品和比较

经过近几年的发展,各企业根据自身的业务需求和数据特征设计了各自的云数据库,通过对当前云数据库市场的调查,结果如表 1 所示。

表 1 主流云数据库

企业	数据库产品
Amazon	SimpleDB
Google	BigTable
Microsoft	SQL Azure
Oracle	Oracle Cloud
Yahoo!	PNUTS
EnerpriseDB	Postgres Plus in the Cloud
开源产品	Hbase,Hypertable
其他	EnerpriseDB,FathomDB,ScaleDB

使用云数据库平台可以直接采用 Amazon、Microsoft、Oracle 的云存储解决方案,但是建设这样的平台代价很大,对经费要求很高。同时也可以采用 HBase、Hypertable 等开源的解决方案,这虽然免费并且可以根据自身应用做相应的优化,但是对技术要求很高,后期开发具有一定难度。为了选取合适的云数据库,对各个产品进行了对比。

2.1 Amazon 的 SimpleDB

SimpleDB[6]是 Amazon 提供的简单数据库服务,主要用于存储结构化数据,并为数据提供查找、删除等基本的服务,其具体的实现细节 Amazon 没有公开。

由于 Amazon 主要是提供商业性的服务,使用其服务需要一个 Amazon 的帐户,那么一个用户帐户就相当于全集,而具体的数据则相当于子集。由于 SimpleDB 简单的数据存储方式,其所有的数据都是以字符串形式存储,导致其采取词典顺序进行查询,数据操作很不方便。考虑到其技术封闭性,不能在实验环境中使用其平台。

2.2 Google 的 BigTable

BigTable[7]是 Google 基于 GFS[8](Google File System)和 Chubby 开发的分布式存储系统。BigTable 是非关系型数据库,是一个稀疏的、分布式、持久化存储的多维度排序表。它采用行键(row key)、列键(column key)和时间戳(timestamp)对表进行索引。表中的每个值都是未经解释的字节数组。BigTable 在行键上根据字典顺序对数据进行维护,并且一张表的行键也是其划分行区间,进行 Split 和负载均衡的依据。其设计目的是可靠地处理 PB 级的数据,并且能够部署到上千台机器上。BigTable 已经实现了下面的几个目标:适用性广泛、可扩展、高性能和高可用性。BigTable 已经在多个 Google 的新产品和项目中得到了应用,如 Google Analytics 和 Google Earth 等。

根据对 BigTable 的研究,了解到 BigTable 主要是

Google 针对自身各种应用设计的存储系统,并且没有开源,在实验中无法使用。但是 BigTable 的主要技术和实现都对外开放,并且 BigTable 的研究资料非常丰富,在研究云存储的过程中具有很高的借鉴意义和参考价值。

2.3　Microsoft 的 SQL Azure

SQL Azure 是微软的云关系型数据库,是基于 SQL Server 技术构建的,主要为用户提供数据应用服务。SQL 简化了数据库的部署,用户无需安装和配置数据库,也不需进行维护和管理。并且,SQL Azure 还为用户提供高可用性和容错能力。SQL Azure 做为一个商用数据库,其提供一个云端的 DBMS(DataBase Manager System),这使得本地应用和云应用可以在微软的数据中心的服务器上存储数据。用户是按需付费,其中主要的费用是操作费用,而不是磁盘和 DBMS 软件投入的费用。

SQL Azure 做为一款商业软件,其采用按需服务、按需收费的模式。由于其不开源,所以无法在实验环境中搭建,但是做为云数据库中采用关系型数据模型的典型代表,依然具有重要的研究意义。

2.4　Apache 的 HBase

HBase[9] 数据库是基于 Hadoop 的 Apache 顶层项目,它是 BigTable 的开源实现,但存在很多不同之处。HBase 是一个在 HDFS 上开发的面向列的分布式数据库,主要支持实时的随机读写超大规模数据集。HBase 是自底向上地进行构建,能够简单地通过增加节点来达到线性扩展。HBase 是非关系型数据库,不支持 SQL 查询,但其具备了 RDBMS 无法比拟的特性:在廉价硬件构成的集群上管理超大规模的稀疏表。HBase 系统结构如图 1 所示。

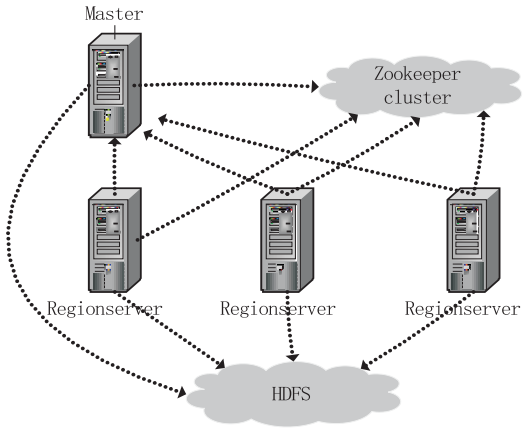

图 1　HBase 系统结构

HBase 采用一个 Master 节点协调管理一个或多个 RegionServer 从属机(HBase 把表水平划分成 Region "区域")。HBase 主控机(master)负责启动和注册 RegionServer,把 Region 分配给 RegionServer 并负责 RegionServer 的故障恢复。RegionServer 负责 Region 的管理和响应用户的读写请求,当有新的 Region 产生时,RegionServer 将通知 Master 节点。

HBase 依赖于 Zookeeper(Zookeeper 提供分布式锁服务,类似 Google 的 Chubby),它管理一个 Zookeeper 实例,作为集群的"权威"(authority),并负责根目录表的位置、当前集群主控机地址等重要信息的管理,并负责维护整个集群的工作状态和灾难恢复。

HBase 是开源实现,可以方便地从互联网上下载安装包和源代码,非常适合企业、科研单位和学校进行使用学习和再开发。但其操作和管理界面比较简单不够友好,需要进一步提高。

2.5　云数据产品比较结果

通过对几种具有代表性的云数据库进行研究,比较结果如表 2 所示。

表 2　具有代表性数据库比较结果

数据库	优缺点
SimpleDB	商业服务简单易用,不利于开发研究
BigTable	Google 自用,体系设计公开,原始资料丰富有利参考研究
SQL Azure	提供弹性的商业服务,具有参考价值
HBase	BigTable 开源实现,具有很多成功案例可以供参考,可做为深入学习和研究云数据库的平台

3　实　验

综上所述,决定采用 HBase 做为研究云数据库的实验平台。

HBase 的安装可以分为三种模式:单机模式、伪分布式模式和完全分布式模式。文中采用完全分布式模式,这样可以模拟实际网络环境,能够体现云数据库的特性,使实验结果更具有说服力。

3.1　实验环境的搭建

在实验环境中共有 6 台服务器,搭建完全分布式 HDFS 与 HBase 环境,采用 hadoop0.20.0 与 HBase0.92.0 版本,其中二台节点做为 Namenode 和 Master 节点,另外四台做 Datanode 和 RegionServer,并且在其上运行 Zookeeper 服务。整个实验环境[10] 如图 2 所示。

在安装 Hadoop 和 HBase 之前要在系统中安装 JDK 并配置好环境变量。在用户目录下安装好 Hadoop 和 HBase 之后,进入 Hadoop 目录下输入命令:

```
rm /tmp/ *
bin/hadoop namenode - format
bin/start-all.sh
```

来启动 HDFS,可以使用 bin/hadoop dfsadmin - report 来查看 HDFS 的可用资源、实际使用百分比和 Datanode 的运行情况。

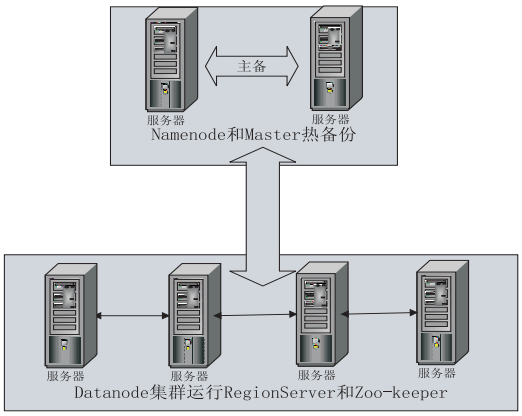

图 2 HBase 集群结构

HBase 是运行在 HDFS 之上的,所以必须确保 HDFS 处于正常运行状态。同时因为存在版本兼容性问题,在启动 HBase 之前必须让 HBase 确定使用 Hadoop 的版本,需要把 Hadoop 目录下的 hadoop-0.20.2-core.jar 替换掉 HBase/lib 目录下的 hadoop-core-1.0.0.jar。最后确保集群中每台的时间保持相对一致(误差小于 30 秒),进入 HBase 目录输入命令 bin/start-hbase.sh 启动 HBase。用命令 bin/hbase shell 进入外壳程序,用命令 status 查看 HBase 的状态。

3.2 实验详解

在完成实验环境的搭建之后,为了对 HBase 官方描述的 HBase 系统具备的特性进行验证,设计了一些实验对 HBase 集群进行初步的测试:

1)海量数据的存储[11,12]。

HBase 提供 JAVA 编程 API,在 Eclipse 环境下编写数据库写入程序,并且递增数据写入量,查看所需的时间,对性能进行简单的分析。

根据 HBase 存储的特点,因为 HBase 是对 Rowkey 进行排序的,随机 Rowkey 将被分配到不同的 region 上,这样能发挥出分布式数据库的优点。而 Value 对于 HBase 来说不会进行任何解析,其数据是否变化,对性能是不应该有任何影响的。同时为了简单起见,所有的数据都将只插入到一个表格的同一个列中。

数据插入性能测试的设计场景是这样的:取随机值的 Rowkey 长度为 2000 字节,值的 Value 长度为 4000 字节,每次插入 10000 条数据,直到 1000 万条结束。实验建立的表名为 TestData,实验开始后可以从 http://master:60010 查看进展。

结果表明从数据刚开始写入时,只存在 1 个 Region,随着数据达到一定的规模之后 TestData 开始分裂,并实现负载均衡,把 Region 平均存放在 Datanode 中。

HBase 不仅支持自动的负载均衡和副本容灾机制,并且读写性能也很优秀,通过对上述实验数据写入时间的统计,数据写入速度达到 0.47ms/条(2127 条/秒),并且在理论上还有很大的提高可能,因为实验用的数据库写入程序为单线程,没有采用 MapReduce 并行编程模型。如果采用 MapReduce 模型,写入速度将会有很大程度的提高。这将在以后的研究工作中得到实验验证。

2)数据库的动态扩展。

相对传统的数据库,云数据库具有近乎无限的可扩展性,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。为证明云数据库具有的这个优势设计了如图 3 所示的实验。

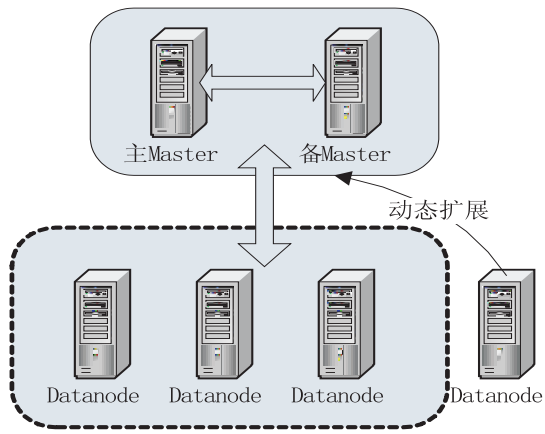

图 3 Datanode 动态扩展

首先用 5 台服务器建立一个云数据库平台,其中 2 台 Namenode 和 Master,另外 3 台为 Datanode 和 RegionServer。启动 HDFS 和 HBase,待系统正常运行之后查看 HDFS 和 HBase 状态可以得到系统中正常的 Datanode 和 RegionServer 都是 3 个。然后,在 Namenode 上注册想要加入的节点 IP 地址,于 Hadoop 的 conf 目录下的 slaves 文件和 HBase 的 conf 目录下的 regionservers 文件中添加新增节点描述,并把配置文件复制到整个集群的每台服务器上。用 start-all.sh 命令对整个 HDFS 系统进行启动,这时系统会对注册 Slave 的节点进行检测。如果已经挂载那么则跳过,否则将在该节点上启动 HDFS 并挂载到系统中来。再运行 start-hbase.sh 把新添加的 RegionServer 挂载到集群中。最后,可以看到整个 HDFS 和 HBase 的可用节点都动态地增加了。这种动态扩展性能能够满足动态的需求并在节能方面具有很大的优势。

3)数据库集群的抗毁性。

云数据库的抗毁性是其中一个很重要的性能指标,主要体现在数据节点的容错性和 Master 节点的动态备份容错。针对这两点设计实现了如图 4 所示的实验。

数据节点的容错性实验:在搭建好数据库集群中

创建表并写入一定量的数据,手动进行负载均衡让数据平均存放在每个节点中。然后,再在数据节点集群中 kill 一定数量的节点,并检测数据是否可用。结果是:当 down 掉的数据节点小于集群配置 dfs. replication 的数据副本数时,该集群的数据始终保持可用。

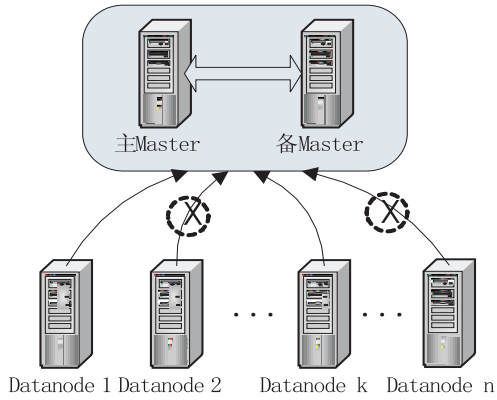

图 4　Datanode 容错

Master 节点的动态切换:Master 节点是管理数据的元数据表,表的查询都要经过元数据表的索引,所以 Master 节点具有唯一性,并且 Master 必须要高可靠。设计如图 5 所示的实验。

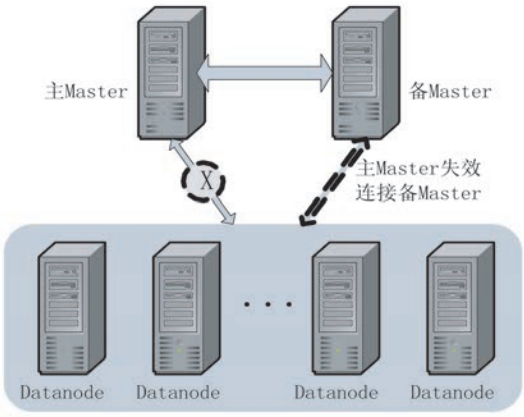

图 5　Master 容错

两个节点同时运行 HMaster 服务,其中一个为主 Master,另一个为备份 Master 并与主 Master 保持同步。在数据库集群正常运行的情况下,在主 Master 节点中 kill 掉 HMaster 进程,集群在经过 zookeeper. session. timeout 定义的时间之后,检测到主 Master 节点不可用,这时集群将进行 Master 节点的切换,并且之前数据库的数据不会丢失,整个集群依然可用。所以,备 Master 在主 Master 出异常不可用后将接替其管理 HBase 数据库的功能。

4　实验结果

通过实验可以看出 HBase 在可扩展性、海量数据存储、高可用性以及管理和运行维护方面具有很大的改进和提高。

在动态可扩展性方面:实验表明 HBase 在添加存储节点对数据库进行扩容的过程中,数据库没有停止服务,并且也不要求物理机具有相同的架构。说明 HBase 的扩展是在线、动态具有弹性的。从而使得 HBase 可用于提供弹性服务,在服务峰值动态加入大量节点用以满足服务请求,提高服务质量。而在低谷时期则关闭大量节点,这样可减少能量消耗降低成本。

海量数据的读写性能方面:由于 HBase 是基于 HDFS 之上建立的,所以也继承了其 Map/Reduce 的计算模型,这使其具有优秀的读写性能。淘宝网通过优化 HBase 使得在数亿条商品数据中查询指定商品的时间可以达到毫秒级。

高可用性方面:传统数据库设计时是考虑如何避免故障的发生,而 HBase 则是以大的集群为出发点,把故障做为一个常态来进行对待。HBase 采用多 Master 节点同时运行的策略:其中一个 Master 为主 Master 提供服务,而其它的 Master 节点则保持与主 Master 的同步,当主 Master 出错之后由 Zookeeper 选举产生新的主 Master 继续提供服务。对 RegionServer 而言只要不同时有大于副本数目的节点失效则可保证数据一定可用。

另外,在低成本和易用性方面 HBase 对传统数据库也有很大的优势。硬件方面 HBase 只要求使用普通的商用服务器即可,软件则是开源,节约了大量成本。并且,HBase 屏蔽了物理层,对于客户端而言就像使用一个安装在本地的数据库,操作简单,维护方便。

5　结束语

文中提出了可以使用云数据库来解决当前传统数据库面临的诸多问题,并用 HBase 做了验证性的实验。通过实验发现云数据库除了可以提供传统数据库一样的数据存储服务以外,还解决了目前传统数据库面临的问题,特别是在可扩展性方面具有无与伦比的优势。云数据库是未来数据库发展的主流方向,可以做为企业单位、科研院所和学校数据库的首选产品。

下一步实验和研究工作主要集中在云数据库性能的详细测试和分析,并根据实际的应用场景进行性能的优化。

参考文献:

[1]　林子雨,赖永炫,林　琛,等. 云数据库研究[J]. 软件学报,2012,23(5):1148-1166.

[2]　Chen C,Chen G,Jiang D W,et al. Providing scalable database services on the cloud[C]//Proc. of the 11th Int'l Conf. on Web Information Systems Engineering (WISE 2010). Berlin:

(下转第 46 页)

4 结束语

文中提出了基于 ROS 和扩散机制的平均时间同步算法(IATS)。利用 ROS 模式,监听范围的节点可以通过一次成对报文交换过程,同时与这对节点进行时间平均而不失精度,提高了算法效率。同时,通过改进 ATSP 的扩散机制,使尽量多的节点通过 ROS 模式达到同步,这节省了大量的报文开销。仿真结果证明,新算法更加适合于能量有限、拓扑结构动态变化的 WSN 中。今后的工作主要集中在以下几个方面:一、研究自适应调整时钟的方法,加速平均时间同步的收敛;二、包含移动节点的 WSN 时间同步问题将是未来的研究方向。

参考文献:

[1] Elson J, Romer K. Wireless Sensor Networks: A New Regime for Time Synchronization[C]//Proceeding of the First Workshop on Hot Topics in Networks (HotNets-I). [s. l.]: [s. n.], 2002: 28-29.

[2] Elson J, Girod L, Estrin D. Fine-grained time synchronization using reference broadcasts[C]//The 5th Symp on Operation System Design and Implementation. Boston: [s. n.], 2002.

[3] Ganeriwal S, Kumar R, Srivastava M. Timing-sync protocol for sensor networks[C]//The 1st ACM Conf on Embedded Networked Sensor Systems. Los Angeles: [s. n.], 2003.

[4] Miklos M, Branislav K, Gyula S, et al. The flooding time synchronization protocol[C]//The 2nd ACM Conf on Embedded Networked Sensor Systems. Baltimore, USA: [s. n.], 2004.

[5] Greunen J V, Rabaey J. Lightweight time synchronization for sensor networks[C]//The 2nd ACM Int'l Workshop on Wireless Sensor Networks and Applications. San Diego: [s. n.], 2003.

[6] Mihail L S, Chanchai V. Simple, Accurate Time Synchronization for Wireless Sensor Networks[C]//Proceedings of the IEEE Wireless Communications and Networking Conference. New Orleans, LA: [s. n.], 2003: 1266-1273.

[7] Su Ping. Delay measurement time synchronization for wireless sensor networks[R]. Berkeley: Berkeley Lab, 2003.

[8] Xu C N, Zhao L, Xu Y J, et al. Broadcast time synchronization algorithm for wireless sensor networks[C]//Proceedings of the 1th International Conference on Sensing. China: [s. n.], 2006: 2366-2371.

[9] Li Q, Rus D. Global clock synchronization in sensor networks[J]. IEEE Trans on Computers, 2006, 55(2): 214-226.

[10] Li L, Liu Y, Yang H, et al. A Precision Adaptive Average Time Synchronization Protocol in Wireless Sensor Networks[C]//Proceedings of the IEEE International Conference on Information and Automation. Zhangjiajie, China: [s. n.], 2008.

[11] Wu Jianshe, Jiao Licheng, Ding Ranran. Average time synchronization in wireless sensor networks by pairwise messages[J]. Computer Communications, 2012, 35(2): 221-233.

[12] Ren F, Lin C, Liu F. Self-correcting time synchronization using reference broadcast in wireless sensor network[J]. IEEE Wireless Communications, 2008, 15(4): 79-85.

[13] Rentel C H, Kunz T. A mutual network synchronization method for wireless adhoc and sensor networks[J]. IEEE Transactions on Mobile Computing, 2008, 7(5): 633-646.

[14] Noh K L, Serpedin E. Pairwise broadcast clock synchronization for wireless sensor networks[C]//Proceedings of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks. Helsinki: [s. n.], 2007: 1-6.

[15] Estrin D. Tutorial "Wireless Sensor Networks" Part IV: Sensor Network Protocols[C]//Proc of MobiCom. USA: [s. n.], 2002.

[16] Cheng K, Lui K, Wu Y, et al. A Distributed Multihop Time Synchronization Protocol for Wireless Sensor Networks Using Pairwise Broadcast Synchronization[J]. IEEE Transaction on Wireless Communications, 2009, 8(4): 1764-1772.

(上接第 41 页)

Springer-Verlag, 2010: 1-19.

[3] Chen K, Zheng W M. Cloud computing: System instances and current research[J]. Journal of Software, 2009, 20(5): 1337-1348.

[4] 宋丽华,姜家轩,张建成,等. 黄河三角洲云计算平台关键技术研究[J]. 计算机技术与发展,2011,21(6):40-43.

[5] 刘 鹏. 云计算[M]. 北京:电子工业出版社,2010.

[6] Amazon Group. Amazon elastic compute cloud (Amazon EC2)[EB/OL]. 2010-01-01[2010-12-15]. http://aws.amazon.com/ec2.

[7] Chang F, Dean J, Ghemawat S, et al. Bigtable: A distributed storage system for structured data[C]//Proc. of the 7th Symp. on Operating Systems Design and Implementation (OSDI 2006). Seattle: USENIX Association, 2006: 205-218.

[8] Cryans J D, April A, Abran A. Criteria to compare cloud computing with current database technology[C]//Proc. of the Int'l Conf. on Software Process and Product Measurement (IWSM/ Metrikon/Mensura 2008). Berlin: Springer-Verlag, 2008: 114-126.

[9] Ghemawat S, Gobioff H, Leung S T. The Google file system[C]//Proc. of the 19th ACM Symp. on Operating Systems Principles (SOSP 2003). New York: ACM Press, 2003: 29-43.

[10] 刘 鹏. 实战 Hadoop[M]. 北京:电子工业出版社,2011.

[11] 李 华,赵建平. 分布式数据库数据查询的优化处理方法[J]. 长春理工大学学报,2005,28(4):85-87.

[12] 刘 星. HBase 性能深度分析[EB/OL]. 2012-08-01. http://www.programmer.com.cn/7246/#more-7246.

云数据库应用研究

作者： 青欣， 胥光辉， 戢瑶， 郭霄
作者单位： 解放军理工大学 指挥自动化学院,江苏 南京 210007
刊名： 计算机技术与发展
英文刊名： Computer Technology and Development
年，卷(期)： 2013(5)