

基于 GemFire 的海量数据计算性能实验分析

徐翔, 邹复民*, 廖律超, 朱铨

(福建工程学院 下一代互联网技术应用研究开发中心, 福州 350108)

(*通信作者电子邮箱 fuminzou@ngi.fj.cn)

摘要:针对交通领域多源动态海量数据高性能计算的实时性、动态扩展性处理要求,提出了一种基于 GemFire 的分布式内存数据库实验平台。采用键-值数据存储结构和分布式动态成员关系,通过加载浮动车系统的真实数据在完整的云计算架构下,进行了计算性能测试与分析。实验结果表明,平台可将千万级以上大数据量的计算时间缩短至原系统的 10% 以内,满足了交通物联网云平台整合利用各子系统数据资源的应用需求。

关键词:海量数据;分布式内存数据库;云计算;动态扩展性

中图分类号: TP393.03 **文献标志码:** A

Experimental analysis for calculation performance of mass data based on GemFire

XU Xiang, ZOU Fumin*, LIAO Lyvchao, ZHU Quan

(Research Center for Next-Generation Internet Technology and Applications, Fujian University of Technology, Fuzhou Fujian 350108, China)

Abstract: With the demand of real-time and dynamic scalability processing for multi-source mass data in transportation, this paper proposed a distributed in-memory database experimental platform based on GemFire. The platform used the attributes of GemFire, such as key-value data storage structure and distributed dynamic membership. The actual data from floating car system was used to complete the performance analysis in cloud computing architecture. The experimental results show that the platform can shorten the calculation time of mass data to less than 10% of the existing system and basically satisfy the application requirements of transport data resources integration in cloud computing platform.

Key words: mass data; distributed in-memory database; cloud computing; dynamic scalability

0 引言

交通物联网的规划和建设已经成为我国交通部信息化“十二五”时期的重点工作内容之一^[1],它需要处理多源的海量数据资源,为人、车(船)、路、货和环境等各交通要素提供所需的交通信息数据^[2-3]。而传统数据库及系统平台架构不能满足海量动态数据实时处理大量化、多样化和快速化等需求^[4],因此,需要利用云平台自动化信息技术资源调度和快速部署的特性,充分发挥分布式内存数据库的动态扩展性,从而解决海量动态数据处理、分析、挖掘和利用过程中的计算性能瓶颈问题。

交通物联网云平台自身就是一个强大的“云”网络,连接了大量的并发网络设备和服务,可以将各交通要素的资源通过云计算平台结合起来,从而提供超级计算和存储能力,实现数据资源的按需服务^[5]。传统类型的基于磁盘 I/O (Input/Output) 的数据库架构是将数据统一放在磁盘上^[6],而内存数据库则将数据库所有或部分数据放置在服务器内存中运行,由于内存 I/O 访问速度通常是磁盘 I/O 访问速度的几百倍,内存数据库性能较传统磁盘数据库有明显优势^[7-8]。同时,分布式数据库则通过将海量数据基于特定算法分散在多个服务器节点上,当进行任何计算时,多个服务器节点可并发执行数据运算任务,大大缩短了运算时间^[9]。因此,利用分布式内存数据库来进行海量数据计算已得到电力、医疗、电信等领域的广泛关注和高度重视,证明了其对于改进海量数据计算性能的有效性^[10-12]。

但是,目前已有的研究及应用成果,例如文献[13]中所提出的分布式地图匹配系统和文献[14]中所提出的分布式缓存架构 OnceDC 等,尚未能良好地将云平台的虚拟化等关键技术同分布式内存数据库充分结合起来,其用于分析计算性能的数据样本也往往并不具备海量数据的显著特征,尤其对于交通领域海量数据计算动态扩展性的研究,尚缺乏实际案例背景下的有力佐证。

为此,本文将借助业界领先的虚拟化平台 VMware vSphere 构建云平台基础架构,利用 Spring Java 开发框架等技术,实现基于 GemFire 分布式内存数据库的交通物联网云平台原型系统,并通过加载“省交通运输行业物联网应用整合与服务工程”项目中的实际海量交通数据来验证、分析其计算性能的动态扩展性。

1 海量数据计算关键技术

1.1 云计算数据处理技术

海量数据包含各种格式和环境的信息,对其进行计算处理的过程,实际上是在实现数据存储和检索之后,如何将有用的数据进行整合、提炼和分析,最终形成支持企业业务发展的决策过程^[15]。为了更好地实现该过程,2006 年由 Google、Amazon 等公司提出了“云计算”的构想,它描述了一种新的基于互联网的 IT 服务增值、使用和交付模式^[16],是数据共享与服务共享计算模式的结合体^[17]。

其中:Google 公司提出的高效大规模数据处理 MapReduce 是云计算最重要的核心技术之一,能够通过这种

收稿日期:2012-07-20;修回日期:2012-09-03。 基金项目:国家自然科学基金资助项目(61101139);福建省杰出青年基金资助项目(2012J06015);福建省重大专项专题项目(2011HZ0002-1)。

作者简介:徐翔(1980-),男,福建福州人,硕士,主要研究方向:云计算在交通运输行业的应用; 邹复民(1976-),男,湖南隆回人,副教授,博士,主要研究方向:无线宽带网络、交通物联网; 廖律超(1980-),男,福建福州人,博士研究生,主要研究方向:物联网在交通运输行业的应用; 朱铨(1983-),男,湖北黄冈人,博士研究生,主要研究方向:车载 Internet 在智能交通的应用。

新型的分布式计算技术所提供的 Map 和 Reduce 两个数据处理函数简化数据计算模型,向程序员屏蔽了任务调度、数据存储和传输等细节,非常适合解决海量数据处理问题的伸缩性需求^[18-20];而 Apache 软件基金会则根据 MapReduce 模式和 Google 文件系统(Google File System, GFS)研发实现了开源的分布式并行编程框架 Hadoop,极大加速了云计算技术与服务的深入研究和普及,成为 Yahoo!、Facebook、百度和淘宝等企业所采用的核心数据处理技术。

随着云计算技术的不断发展,逐渐催生出新的大数据(Big Data)处理框架^[21],它包括加速、服务和批处理三个层次,并通过一系列开源项目及其所研发实现的内存数据库验证了其对于实时数据系统的计算性能优势,包括批量计算系统 Hadoop、序列化框架 Avro、支持任意存取的 NoSQL(Not only SQL)^[22]数据库 Cassandra 和 CouchDB 等。

这些分布式内存数据库能够为用户提供高性能、高可用、可伸缩的数据计算服务,通过将数据分布到多个计算服务节点,直接在内存中计算、管理和维护数据,对外提供统一的访问接口以及可选的冗余备份机制,通常又被称为分布式内存数据网格(Distributed, In-memory Data Grid)或分布式缓存^[23-24]。TechTarget 的专题评论^[25]认为,对于数据密集型的 Web 应用,若失去分布式缓存这一关键技术的支撑,云的潜能将是十分有限的。Forrester 的技术报告^[26]中提出了弹性缓存平台(Elastic Caching Platform)的概念,强调弹性缓存的关键特性是大容量计算下性能表现的一致性、不停机情况下的动态扩展性和由一系列容错机制所提供的高可用性,这与云平台的内在需求相吻合。

1.2 内存数据库技术

内存数据库是基于磁盘数据库技术发展而来,可将数据库中所有的数据或某几个事务所需要的数据常驻内存,在处理事务的过程中,尽可能减少或者完全消除和硬盘之间的数据交互工作^[27]。

内存数据库系统可将所管理的全部或大部分数据存于内存中,它利用内存的随机访问特点,在内存中模仿建立表结构和索引结构,从而获得非常高的存取速度的数据库系统。在传统的数据库系统 DRDB(Disk Resident Database)中,数据常驻磁盘,DRDB 将磁盘上的数据当作是数据的主拷贝,将内存中的缓存数据看作是复制的备份。在内存数据库系统(Main Memory Database, MMD)中,数据常驻在物理内存中,MMD 将内存中的数据看作主拷贝数据,硬盘上的数据看作是数据备份。由于计算机的内存和磁盘有着明显不同的特性,这决定了两者在存取时间、索引结构和查询优化等方面有很大的差异,如表 1 所示。其中:AVL 树(Adelson-Velsky and Landis Tree)为自平衡二叉查找树。

表 1 内存数据库与磁盘数据库的差异

性能	内存数据库	磁盘数据库
存取时间	10^{-8} s 量级	10^{-3} s 量级
数据存储	不需要连续存放	连续存放
缓冲管理	不需要	需要
索引结构	哈希、AVL 树、T 树、B 树	B 树、B+ 树、Hash
并发控制	大粒度锁	细粒度锁加锁、解锁、死锁检测
查询优化	基于处理器代价以及 cache 代价	基于 I/O 代价

由此可以看出,内存数据库可以提供比磁盘数据库好得多的响应时间、事务吞吐量和查询性能,这对那些要求事务在特定期限内完成的实时应用尤为重要,适用于海量数据的实时处理。

而在云计算环境下,内存数据库逐步向分布式弹性计算

方向发展^[26]。

1)本地缓存。数据存储在应用代码所在内存空间,可提供快速数据访问,但无法分布式共享数据,也不提供容错处理,例如 Cache4j。

2)分布式缓存系统。数据在固定数目的集群节点间分布存储,缓存容量可实现静态扩展,但扩展过程中需大量配置,无容错机制,例如 Memcached。

3)弹性缓存平台。数据在集群节点间分布存储,基于冗余机制实现高可用性,可实现动态扩展并提供容错机制,但复制备份会对系统性能造成一定影响,例如 Appfabric Caching。

4)弹性应用平台。代表了云计算环境下分布式内存数据库未来的发展方向,是弹性缓存与代码执行的组合体,可将业务逻辑代码转移到数据所在节点执行,从而极大地降低数据传输开销,提升系统性能,例如 GemFire。

目前,典型的分布式内存数据库产品主要包括两大类:一类通常以中间件形式出现,以改进数据访问速度为主要目的,例如 Oracle Coherence、IBM WebSphere eXtreme Scale(WXS)、GigaSpaces eXtreme Application Platform(XAP)等;另一类通常以独立数据管理系统形式出现,以实现动态扩展性和高可用性为主要目的,例如 Oracle Timesten、IBM solidDB、McObject eXtremeDB 等。

2 分布式内存数据库 GemFire

内存数据库中间件通常配置复杂,系统平台集成度要求高且商业版本采购成本昂贵。而大多基于开源项目实现的内存数据管理系统则存在着进一步开发维护难、可靠性和稳定性存疑的缺陷。另一方面,要在云平台上发挥内存数据库的优势,离不开虚拟化技术的支持。弹性内存数据管理系统 GemFire 恰好能避免上述缺陷,并能同云计算平台虚拟化技术良好结合。

GemFire 原先是 Spring Java 企业开发框架的开源项目,于 2010 年被全球虚拟化技术、云计算基础架构解决方案领先企业 VMware 收购,成为其云计算应用平台 vFabric 的一部分,并保留原来的开源部分。从而, GemFire 能够为以云为中心的应用程序,在具有弹性、高可伸缩性和分布式的框架中进行部署,提供高效的数据管理方法,使得在一个分布式云环境中可以在准确的时间将恰当的数据提供给相应的云应用^[28]。

VMware vFabric GemFire 作为一个分布式数据管理平台^[25],已经成功地应用部署在全球外汇交易系统、美国国防部、摩根大通银行、江苏邮政等。它通过融合复制、分区、数据感知型路由和连续查询等高级技术,实现了云规模的高性能数据管理,并可提供动态扩展性、高性能和类似于传统磁盘数据库的可靠性和持久性,其平台架构如图 1 所示。

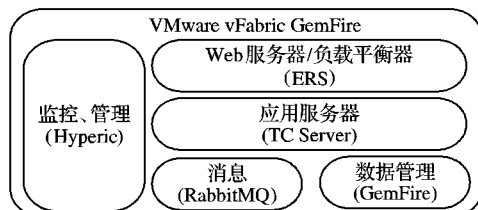


图 1 VMware vFabric GemFire 内存数据库架构

图 1 中:ERS 是 Apache Web Server 的企业版,易于安装和使用,可提供针对企业及应用场景的性能和安全增强;TC Server 是面向企业的 Tomcat 轻量级服务器,提供了适用于虚拟环境下的 Spring 应用程序容器;RabbitMQ 是基于高级消息队列协议(Advanced Message Queuing Protocol, AMQP)的消息中间件,可提供健壮、可靠的跨系统消息传递;而 Hyperic 则是应用程序性能跟踪、监控、报警和管理的工具,可提供自下而上的可见性,并可

通过与 vSphere 的集成同时监控物理和虚拟化环境。

VMware vFabric GemFire 的主要特性和功能还包括:针对 Tomcat 和 TC Server 的 HTTP 会话管理、用于 Hibernate 的 2 级缓存、增强的“无共享”并行磁盘持久性、Spring 集成和简化的 API(Application Programming Interface)提高了开发的方便性、通过分区提高了横向扩展能力、跨分区事务共存大幅提高了吞吐量、可预测的低延迟、跨多个成员节点进行动态数据分区并在服务器之间分布数据负载来实现的高可扩展性、同步或异步归档磁盘所实现的连续可用性、无需借助转换层的异构数据共享、无限和松散耦合横向扩展方式的广域数据分发等^[29]。

3 基于 GemFire 海量交通数据计算实验平台

根据上述针对海量数据计算关键技术的比较研究,本文基于 GemFire 分布式内存数据库完成了交通物联网云平台原型系统设计,使之符合交通物联网在云计算环境下处理多源海量动态数据的应用需求。

为了测试该系统在实际环境下的计算性能表现,本文选取了“省交通运输行业物联网应用整合与服务工程”实施项目中的浮动车实时定位子系统真实数据作为海量数据源。浮动车实时定位系统汇集了来自于 16 万辆(平均在线 9.6 万至 9.8 万)浮动车的实时数据,包括全球定位系统(Global Positioning System, GPS)数据、车辆状态数据等,采用 30 s 的更新采样频率和 5 min 的发布周期,其数据带有明显的海量数据特征,计算量大,且必须具备较高的时效性和可靠性。

浮动车系统的实验平台数据模型通过 GemFire 的键-值(key-value)存储引擎来实现其数据存储结构的对象模型。

定义 key 类为:

```
class OrientDataKey {
    private String Tm_id;           //GPS 终端号
    private String Time;           //时间
}
```

定义 value 类为:

```
class OrientData {
    private byte Order_id;          //指令 id
    private String Tm_id;           //GPS 终端号
    private String Time;           //发送时间
    private int IsLocate;           //是否精确定位
    private double Longitude;       //经度
    private double Latitude;        //纬度
    private float Speed;            //速度
    private int Direct;             //方向
    private String alarm;           //车辆预警
}
```

根据 GemFire 内置的基于分布式系统的动态成员关系,本文所设计的实验平台逻辑结构如图 2 所示。

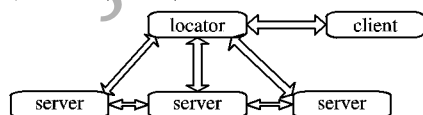


图2 实验平台逻辑架构

1) server 节点。

一个 server 通常可视为一个内存数据库的存储节点,该节点可以运行在任何能够运行 Java 虚拟机的系统内。运用云平台中普遍采用的虚拟化技术,既可以把多个 server 运行在一台物理机中,也可以分散到不同的物理主机上来运行。

一个 server 通过 locator 发现其他 server 并建立连接,server 之间的连接通常用于数据备份。同时,每个 server 中的数据一般都存在内存,在资源不足的时候可以存放在硬盘中,并支持以一定的策略归档至磁盘。同时,多个 server 之间选用了全复制分区(Replicated Region)的缓存策略。

2) locator 节点。

locator 通常可视为管理 server 存储节点的程序,可以独立存在于一个单独系统,也可以跟某个 server 放在一个系统里,该过程可通过在 server 的配置文件里配置好正确的 locator 的 IP 地址和端口来实现。

locator 的作用是帮助 server 互相建立连接,运行负载均衡策略,并帮助 client 定位到合适的 server。

3) client 节点。

client 负责发送和接收 GPS 等数据信息,将其提交到 server 中进行存储。同时,通过 client 可发起查询等操作指令,通过 server 运行相关计算逻辑任务后,将运算结果返回 client 接收。

同时,云平台基础架构也是本实验平台的重要组成部分,直接关系到能否顺利、可靠、稳定地进行海量数据计算性能实验分析。本文采用 4 台 Dell PowerEdge R710 服务器作为硬件基础设施,每台均配备了 2 块 Intel Xeon 5506 2.13 GHz CPU 和 32 GB 内存,带有 8 个千兆以太网接口和 1 张光纤 HBA 卡,并将这些服务器通过 Dell Power Connect 6248 交换机连接至 IBM DS4700 存储设备。本文通过 VMware vSphere 5.0 将硬件基础设施进行资源池虚拟化,并通过 VMware vCloud Director 构建起一个真实可用、完整的云计算平台环境,其实验平台物理结构如图 3 所示。

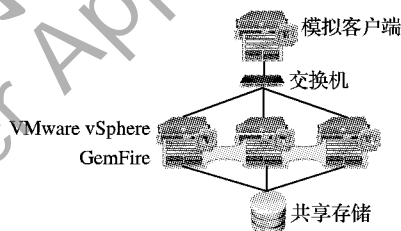


图3 实验平台物理架构

4 实验测试及结果分析

传统应用案例中,通常仅将内存数据库作为纯粹的数据缓冲层,或应用于电信业的计费系统等场景下,较少见其在海量数据计算场景下的性能表现,尤其对于交通物联网云计算环境下的海量多源动态交通数据的实时性处理方面鲜见实质性测试或应用。本文将通过以下 3 个不同的实验,测试分析 GemFire 云架构分布式内存数据库在交通物联网应用中的适用特性:

- 1) 节点数量的增加对 GemFire 数据库计算性能的影响;
- 2) GemFire 数据库每个节点分配的实例数量对性能的影响;
- 3) 随着数据总量的增加, GemFire 数据库处理性能的变化情况。

实验 1 在云平台资源池内逐步增加 3~7 台虚拟机,其上所部署的每个 GemFire 数据管理节点开启 2 个 server,且分配共计 10 GB 内存,固定加载 1000 万条数据,实验时 GemFire 处理节点数递增。图 4 所示为 GemFire 处理节点数量增加时, GemFire 内存数据库的计算性能曲线。

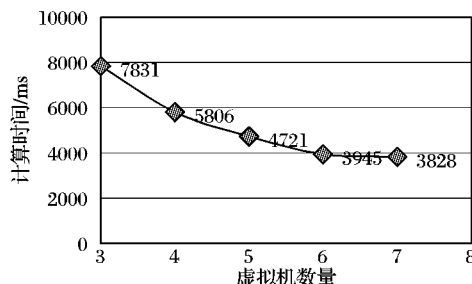


图4 计算性能曲线(GemFire 处理节点增加)

通过图4的性能曲线,可以得到以下结论:

1000万条数据的遍历排序、计算总和最短时间最短仅需3.828 s,而原系统在未采用 GemFire 的条件下需花费1 min 35 s。由于充分利用了 key-value 形式在高速内存中进行存储,极大避免了传统数据库大规模数据访问时,磁盘 I/O 所导致的性能瓶颈,可获得动态随机存取存储器(Dynamic Random Access Memory, DRAM)级别的读写性能。

实验2 在云平台资源池内逐步增加3~6台虚拟机,其上所部署的每个 GemFire 数据管理节点开启3个 server,其分配共计12 GB 内存,加载1000万数据,实验时 GemFire 处理节点数递增。图5所示为 GemFire 处理节点的实例数量增加时, GemFire 内存数据库计算性能曲线。

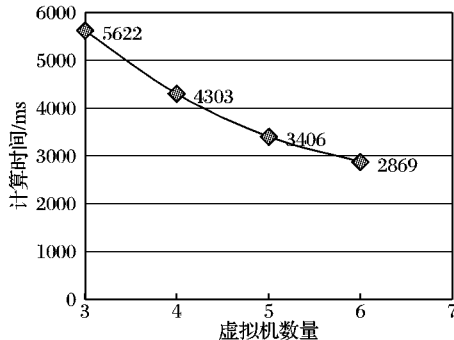


图5 计算性能曲线(GemFire 处理节点的实例增加)

通过图5的性能曲线,可以得到以下结论:

1000万条用户数据的遍历排序、计算总和最短时间仅仅只需要2.869 s时间,实验2的计算性能明显比实验1的计算性能要好,说明 GemFire 数据管理节点的实例越多(意味着进程数量越多),其计算性能越好。

同时,由实验1和2的结果还可得到以下结论:

在1000万用户数据的负载下,6个 GemFire 节点的测试值几乎相当于3个节点的一半,意味着随着节点/实例数量的增加, GemFire 数据库的计算性能也符合线性增长规律。

另外,得益于 GemFire 的弹性扩展(Elastic Scalability)支持能力,它采用一种被动边界技术(patent pending)来确保新节点或新实例加入时,可从有源节点以非阻塞的方式快速获得分布式数据快照来实施初始化,从而可通过动态增加或减少节点/实例来应对变化的数据访问负载,提供可预测的性能与扩展性,并最大限度地提高资源利用率。实验1和2是在数据总量固定,节点/实例数增加的情况下,分析 GemFire 云架构内存数据库的计算性能,接下来将进一步测试在数据总量增加的情况下 GemFire 内存数据库的计算性能。

实验3 在云平台资源池内固定分配6台虚拟机,其上所部署的每个 GemFire 数据管理节点开启3个 server,且分配共计12 GB 内存,实验时数据总量以100万递增。图6所示为数据总量增加时, GemFire 内存数据库的计算性能曲线。

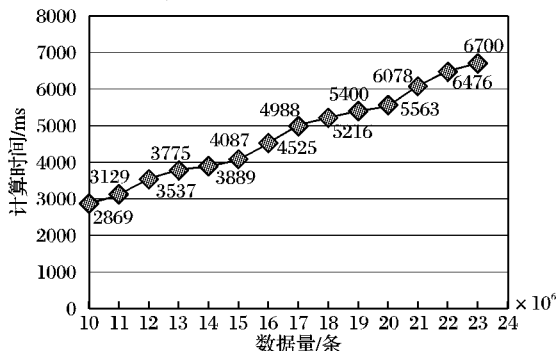


图6 计算性能曲线(GemFire 处理数据总量增加)

通过图6的性能曲线,可以得到以下结论:

在保持6个 GemFire 节点不变的条件下,数据总量从1000万增加至2300万,整个数据库遍历排序并计算总和的时间从2.869 s 增加到6.7 s,且其计算性能曲线变化过渡得相对平滑。同时,通过 vCenter 对于实验云平台 CPU、内存、磁盘资源池的性能指标监控表明,随数据量逐步增长,云平台整体性能开销尚在可接受的范围之内。

由于在 GemFire 的 server 节点之间采用了全复制的缓存策略,数据写入和更新将复制到所有缓存节点,从而使得每个节点都可直接以极低的相应延迟本地读取数据。而通过将计算代码也转移到各 server 节点并行执行后,由 client 节点聚合返回结果,可有效避免缓存数据的移动与传输。另外,若数据总量继续增长,超出了 GemFire 可承受的最大运行容量,则还可通过异步持久化的方式,将数据归档至磁盘。

5 结语

本文针对海量数据计算对于弹性资源供给、可用性和可靠性、敏捷性与自适应性等方面的性能需求,深入分析了云计算和内存数据库技术,设计和实现了基于 GemFire 分布式内存数据库的交通物联网云平台原型系统,实例验证分析了其在大数据量承载下的高速计算性能、动态扩展性、数据持久化、分布式代码执行等方面的性能指标。实验结果表明其完全满足实时性要求,适用于具有时间限制要求的实时系统和海量数据高性能事务处理平台,能够解决传统关系型数据库海量数据处理实时性的瓶颈问题。该实验原型系统为建设交通物联网提供了一个可行的云计算平台应用支撑架构,可将原有数据子系统顺利地进行迁移过渡,从而提高将现有的各交通信息资源整合到交通物联网云平台的速度,为交通物联网的建设提供了一定的理论和实践依据。

下一步的研究重点是如何通过 Hyperic 等性能监控方法来实现海量数据计算性能与云平台资源自主分配之间的良性联动机制,以及针对空间数据库的传统运算和地理信息系统(Geographic Information System, GIS)开发框架在分布式内存数据库架构下的可行性研究。

参考文献:

- [1] 十二五智能交通规划,打造物联网十倍商机[EB/OL]. (2011-10-19) [2012-05-10]. http://www.chinadaily.com.cn/mic-ro-reading/tech/2011-10-19/content_4107623.htm.
- [2] 张国伍.城市智能交通物联网建设探讨——“交通7+1论坛”第二十三次会议纪实[J].交通运输系统工程与信息,2011,11(4): 1-9.
- [3] 谢辉,董德存,欧冬秀.基于物联网的新一代智能交通[J].交通科技与经济,2011,13(1):33-36.
- [4] 薛竹颀.实时内存数据库关键技术的研究与实现[D].南京:东南大学,2006.
- [5] 石建军,李晓莉.交通信息云计算及其应用研究[J].交通运输系统工程与信息,2011,11(1):179-184.
- [6] 杨武军,张继荣,屈军锁.内存数据库技术综述[J].西安邮电学院学报,2005,10(3):95-99.
- [7] 王珊,肖艳芹,刘大为,等.内存数据库关键技术研究[J].计算机应用,2007,27(10):2353-2357.
- [8] 袁培森,皮德常.用于内存数据库的 Hash 索引的设计与实现[J].计算机工程,2007,33(18):69-71.
- [9] 李娇,刘全,傅启明,等.分布式数据库中基于局部 CON 模型的记录匹配方法[J].通信学报,2011,32(7):196-202.
- [10] 李剑.电网监控系统中分布式内存数据库系统的研究[D].南京:河海大学,2004.
- [11] 侯建,帅仁俊,侯文.基于云计算的海量数据存储模型[J].通信技术,2011,44(5):163-165.
- [12] 邵璐,费洪晓.内存数据库技术在移动实时累加系统中的应用[J].计算机系统应用,2011,20(8):169-173.

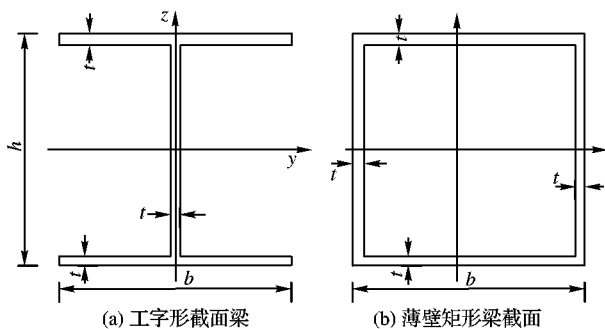


图2 工字形截面梁和薄壁型截面

4 结语

对于一类特殊的非线性方程组,给出了一个显式地计算其在零点的对偶空间的公式,在此基础上将奇异解的重数降为1,恢复了牛顿迭代算法的二次收敛性,并将此算法应用到梁截面几何性质的尺寸实现和解析函数方程组中,在传统算法不收敛的情况下,改进算法的收敛速度和计算精度都得到了提高。对于解析函数方程组,传统的算法是线性收敛,而本文算法达到了二次收敛效果,算法的不足是当奇异解的重数过高时,因引入变量会引起累计误差,在后面的文章中讨论这个问题。

参考文献:

- [1] MISHRA B. Algorithmic algebra [M]. New York: Springer-Verlag, 1993: 71–130.
- [2] 吴文俊. 初等几何判定问题与机械化证明[J]. 中国科学, 1977, 7(6): 507–516.
- [3] 杨路, 张景中, 侯晓容. 非线性代数方程组与定理机器证明[M]. 上海: 上海科技教育出版社, 1996: 95–101.
- [4] YANG L, HOU X R. Gather-and-shift: a symbolic method for solving polynomial systems [C]// Proceedings of the First Asian Technology Conference in Mathematics. Singapore: [s.n.], 1995: 771–780.
- [5] KAPUR D, SAXENA T, YANG L. Algebraic and geometric reasoning using Dixon resultants [C]// Proceedings of the 1994 International Symposium on Symbolic and Algebraic Computation. New York: ACM Press, 1994: 99–107.
- [6] LI T Y, SAVER T, YORKE J A. The random product homotopy and deficient polynomial systems [J]. Numerische Mathematik, 1987, 51(5): 482–500.
- [7] KOU J S, LI Y T, WANG X H. Efficient continuation Newton-like method for solving systems of non-linear equations [J]. Applied Mathematics and Computation, 2006, 174(2): 846–853.
- [8] NEDZHI BOV G H. An acceleration of iterative processes for solving nonlinear equations [J]. Applied Mathematics and Computation, 2005, 168(1): 320–332.
- [9] TRAUB J F. Iterative methods for the solution of equations [M]. New York: AMS Chelsea Publishing, 1982.
- [10] DECKER D W, KELLEY C T. Convergence acceleration of Newton's method at singular points [J]. SIAM Journal on Numerical Analysis, 1982, 19(1): 219–229.
- [11] LEYKIN A, VERSHELDE J, ZHAO A L. Newton's method with deflation for isolated singularities of polynomial systems [J]. Theoretical Computer Science, 2006, 359(1/2/3): 111–122.
- [12] DAYTON B H, ZENG Z G. Computing the multiplicity structure in solving polynomial systems [C]// Proceedings of the 2005 International Symposium on Symbolic and Algebraic Computation. New York: ACM Press, 2005: 116–123.
- [13] DAYTON B H, LI T Y, ZENG Z G. Multiple zeros of nonlinear systems [J]. Mathematics of Computation, 2011, 80(276): 2143–2168.
- [14] LI N, ZHI L H. Computing isolated singular solutions of polynomial systems: case of breadth one [J]. SIAM Journal on Numerical Analysis, 2011, 50(1): 354–372.
- [15] LI N, ZHI L H. Compute the multiplicity structure of an isolated singular solution: case of breadth one [J]. Journal of Symbolic Computation, 2012, 47(6): 700–710.
- [16] MOURRAIN B. Isolated points, duality and residues [J]. Journal of Pure and Applied Algebra, 1997, 117–118: 469–493.
- [17] WU X L, ZHI L H. Computing the multiplicity structure from geometric involutive form [C]// Proceedings of the Twenty-First International Symposium on Symbolic and Algebraic Computation. New York: ACM Press, 2008: 325–332.
- [18] ZENG Z G. The closedness subspace method for computing the multiplicity structure of a polynomial system [C]// Proceedings of the 2009 Interactions of Classical and Numerical Algebraic Geometry. New York: AMS, 2009: 347–362.
- [13] 沈斌, 蒋昌俊, 章昭辉, 等. 一种基于海量 GPS 数据的分布式地图匹配系统的设计与实现[J]. 小型微型计算机系统, 2007, 28(3): 479–481.
- [14] 朱鑫, 秦秀磊, 王联华, 等. 弹性分布式缓存动态扩展方法研究[J]. 计算机科学与探索, 2012, 6(2): 97–107.
- [15] 朱德新, 宋雅娟. 海量数据分析及处理算法实现[J]. 长春大学学报, 2011, 21(8): 42–45.
- [16] Wikipedia. Cloud computing [EB/OL]. [2012-05-05]. http://en.wikipedia.org/wiki/Cloud_computing.
- [17] 陈康, 郑伟民. 云计算: 系统实例与研究现状[J]. 软件学报, 2009, 20(5): 1337–1348.
- [18] DEAN J, GHEMAWAT S. MapReduce: simplified data processing on large clusters [J]. Communications of the ACM, 2008, 51(1): 107–113.
- [19] 覃雄派, 王会举, 杜小勇, 等. 大数据分析——RDBMS 与 MapReduce 的竞争与共生[J]. 软件学报, 2012, 23(1): 32–45.
- [20] 王珊, 王会举, 覃雄派, 等. 架构大数据: 挑战、现状与展望[J]. 计算机学报, 2011, 34(10): 1741–1752.
- [21] MARZ N, WARREN J. Big data: principles and best practices of scalable realtime data systems [M]. Greenwich: Manning Publications, 2012: 1–28.
- [22] NoSQL. Wikipedia [EB/OL]. [2012-04-07]. <http://en.wikipedia.org/wiki/NoSQL>.
- [23] 秦秀磊, 张文博, 魏峻, 等. 云计算环境下分布式缓存技术的现状与挑战[EB/OL]. [2012-03-16]. http://www.jos.org.cn/ch/reader/create_pdf.aspx?file_no=4276.
- [24] BAIN W L. Distributed, in-memory data grids accelerate map/reduce analysis [EB/OL]. [2012-05-08]. <http://www.codeproject.com/Articles/378117/Distributed-In-Memory-Data-Grids-Accelerate-Map-Re>.
- [25] EARLS A. Data grids: foundation for future cloud computing? [EB/OL]. [2012-02-03]. <http://searchsoa.techtarget.com/news/1518647/Data-Grids-Foundation-for-future-cloud-computing>.
- [26] GUALTIERI M, RYMER J R. The forrester wave: elastic caching platforms, Q2 2010 [EB/OL]. [2011-09-03]. [ftp://ftp.boulder.ibm.com/software/solutions/soa/pdfs/wave_elastic_caching_platforms_q2_2010.pdf](http://ftp.boulder.ibm.com/software/solutions/soa/pdfs/wave_elastic_caching_platforms_q2_2010.pdf).
- [27] GARCIA-MOLINA H, SALEM K. Main-memory database systems: an overview [J]. IEEE Transactions on Knowledge and Data Engineering, 1992, 4(6): 509–516.
- [28] VMware. VMware 和云计算手册[EB/OL]. [2012-02-08]. <http://www.vmware.com/files/pdf/cloud/VMware-Cloud-Brochure-CN.pdf>.
- [29] VMware. 弹性内存数据管理产品介绍[EB/OL]. [2012-02-08]. <http://www.vmware.com/files/cn/pdf/vfabric/VMware-vFabric-GemFire-Datasheet.pdf>.

(上接第229页)