操作系统体系结构的研究分析*)

石 进 陆 音 谢 立

(南京大学计算机科学与技术系 计算机软件新技术国家重点实验室 南京 210093)

摘 要 操作系统是计算机中最基本的系统软件,它控制计算机的所有资源并提供应用程序开发的基础。本文介绍了几种主要的计算机操作系统体系结构,并分析和比较了它们的优缺点,最后还介绍了操作系统体系结构目前研究趋势。

关键词 操作系统,体系结构,内核

Research on Operating System Architectures

SHI Jin LU Yin XIE Li

(State Key Laboratory for Novel Software Technology, Department of computer science and technology, Nanjing University, Nanjing 210093)

Abstract Operating system is the most essential system software in computer, which controls all resources of computer and provides the foundation for applications development. We introduce some basic architectures of computer operating system, and analyze their relative merits. The progress direction of operating system architecture research is described in the last of the paper.

Keywords Operating system, Architecture, Kernel

1 引言

操作系统是计算机中最基本的系统软件,它控制计算机的所有资源并提供应用程序开发的基础。从不同的角度来看,操作系统所表现的形式是不同的。从用户的角度看,操作系统所体现是它所提供的各式各样的服务;从程序员的角度来看,操作系统体现的是提供给用户的界面和接口;而从设计人员的角度来看,操作系统又是一大堆模块和它们之间的相互联系,即操作系统的体系结构[1]。实际上,建造一个新的操作系统最主要的任务就是体系结构的设计。随着计算机体系结构和计算模式的发展,它从最初的简单控制循环体的形式发展到复杂的分布式操作系统,用户对操作系统的需求也呈现出了多样化的趋势,现代操作系统已经发展成了最为复杂、最为庞大的软件系统之一。然而软件工程的研究与实践告诉我们,研究软件系统的体系结构对于处理软件系统的复杂性有着重要的意义。操作系统从诞生之初到现在,已经有了几十年的历史,其间提出了多种操作系统体系结构。本文将依次介绍其中最具有代表性的操作系统结构,分析比较它们的优、缺点,并介绍了其研究趋势。

2 主要操作系统体系结构分析

2.1 简单结构系统

简单结构系统主要产生在操作系统发展初期,由于受硬件平台的性能、软件工程技术水平的限制,当时的操作系统结构体现出来的实际上是没有清晰的整体结构,整个系统呈现一种"大杂烩"的局面。操作系统内核程序和用户应用程序混杂在一起,在同一个地址空间上运行(如图1)。

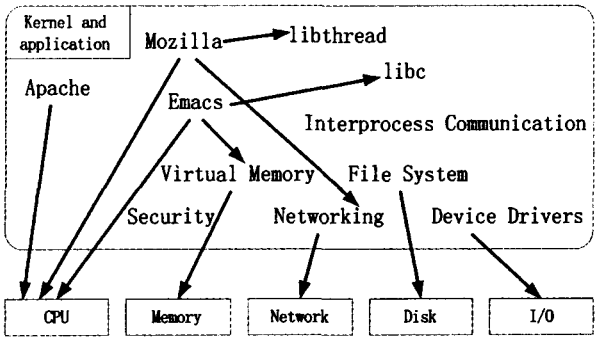


图1 简单结构操作系统示意图

这些操作系统往往是由很小的实验性的项目逐步演化而来的,因而宏观结构非常模糊,使用了早期的多入口、多出口、一个模块完成多个功能的粗模块方法,模块之间可以相互任意调用,整个系统实际上是一堆过程的集合。MS-DOS就是一个很好的例子,在设计之初,MS-DOS的设计目标是在比较有限的硬件资源上运行比较有限的应用程序,开发人员很可能都没有预料到它日后在市场上的巨大成功,因而模块之间的相对独立性几乎被忽略[2]。相似的情况也发生在UNIX家族之中。早期的UNIX因为受限于当时的硬件能力,也一直都是采用非常简单的、模糊的结构[3]。随着UNIX的不断发展这样结构也很快成为了UNIX演进的瓶颈。其他采用这种简单结构的操作系统还包括PalmOS 5及以前的PalmOS[4]、Mac OS 9及以前的Mac OS[5]、Windows ME及以前版本的Windows操作系统[2,6],以及很多其他的小型的嵌入式操作系统[7]。

2.2 单体内核结构系统

*)本课题得到江苏省科技攻关项目BE2002045资助。石 进 博士研究生,研究方向为信息系统安全、安全操作系统。陆 音 博士研究生,研究方向信息系统安全。谢 立 教授,博士生导师,研究领域为信息系统安全,分布式操作系统等。

随着硬件平台的性能逐渐提高，硬件的数量和种类也越来越多，操作系统也越来越复杂，它提供的功能也越来越多，逐渐地单体内核结构的操作系统开始出现，它通过一种称之为系统调用的 API 机制对外层的用户程序提供服务。通常情况下，单体结构内核提供的主要功能有：文件管理、设备驱动、内存管理、CPU 调度以及网络协议处理等。为了更好地控制内核的复杂度，内核的开发人员开始借助于软件工程领域中比较成熟的模块化方法，按照操作系统的功能将单体结构内核进行结构化。这样，整个内核按照功能的不同，被结构化成若干模块：文件管理模块、设备驱动模块、内存管理模块、CPU 调度模块以及网络协议处理等模块。这些模块共享内核的地址空间，它们之间定义了很好的以函数调用的形式提供的通讯接口，模块之间的通讯只能借助于这一接口进行。模块内部的变量只能在模块内部访问。模块化的方法使得某一模块的变化被局部化，只要模块间的通讯接口没有发生改变，一个模块局部的改变不会影响其它的模块，不像简单结构操作系统，一个过程发生变化，调用该过程的其它过程均要做出相应的变化。这种模块化的方法使得操作系统的内核具有易维护、易扩充的特点。

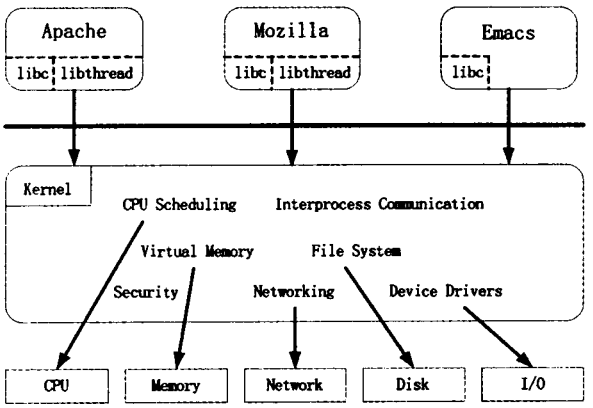


图 2　单体结构操作系统示意图

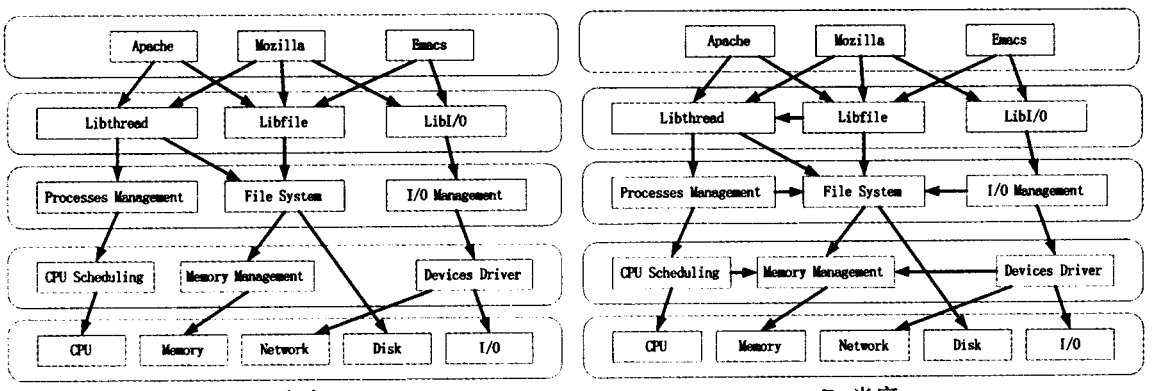
虽然内核被模块化，但所有的模块仍然运行于硬件之上、应用软件之下的操作系统核心中[1]，各个模块的活动层次仍然是相同的（如图 2）。其模块化的过程是纯粹从软件工程的角度进行的。由于单体内核操作系统的简单性，在以集中式计算为主要计算模式的主机时代，单体结构内核得到了广泛的应用。著名的 UNIX 操作系统[3]就是那个时代的典型代表。直到今天，仍然有很多著名的操作系统采用的是单体内核，如 Linux[8]，Mac OS X[9]，Windows NT/XP[10]，BSD[11]等系统。

2.3　层次式结构

为了消除简单结构系统和单体内核结构系统的许多弊病，切实实现操作系统的设计目标，必须减少各模块之间的紧密依赖、相互调用的关系，特别是消除循环调用现象，实现有序调用。层次式结构正是从这一点出发的。

在分层结构的操作系统内核中，系统由若干个层次构成，每一层都构建在其下的一层之上。最底层就是硬件裸机，最高层则是应用程序。在设计分层结构的操作系统内核时，每一层的构造采用的是类似于抽象数据类型的设计方法。每一层中包含了若干的数据和操作，所有的层次内的数据以及部分层次内的操作对其它层是不可见的，也即其它层不能对这些数据和操作进行访问。每一层均公布了一定的操作接口以供其它层调用，这些接口也是外层访问该层唯一的途径。层与层之间的调用关系是严格遵守调用规则。每一层只能够访问位于其下层所提供的服务，利用它的下层提供的服务来实现本层的功能并为其上的层提供服务，每一层不能够访问位于其上的层次所提供的服务。

理想的层次式结构不仅之间是单向依赖的，而且每一层之间也是相互独立的，即它们仅调用低层模块，各模块之间没有调用关系，这种结构称为全序的（图 3A）[12]。1968 年 Dijkstra 和他的学生在荷兰的 Eimdhoven 技术学院所开发的 THE 系统，就是一个全序的层次式结构的操作系统。但是在实际实现特别是大型操作系统，建成全序层次是很困难的，无法完全消除循环调用现象。各层之间是单向依赖，但是在某些层内，允许各模块之间有循环关系，这种层次式结构称为半序的（图 3B），多伦多大学的 SUE[13]操作系统是半序结构的。



A 全序　　B 半序

图 3　层次式结构示意图

2.4　微内核结构

微内核结构又称客户机/服务器结构。现代操作系统的一个发展趋势是，进一步发展将代码移到更高层次的思想，即尽可能多地从操作系统中去掉东西，只留下一个很小的内核。通常采用的方法是，由用户进程来实现大多数操作系统的功能。为了得到某项服务，比如读一文件块，用户进程（即客户机进程）把请求发给服务器进程，随后服务器进程完成这个操作并返送回答信息。

如图 4 所示，操作系统内核的全部工作是处理客户机与服务器之间的通信。操作系统被分成多个部分，每个部分仅仅处理一个方面的功能，如文件服务、进程服务、终端服务或存储器服务等。这样每个部分更小，更易于管理。而且所有

的服务都以用户进程的形式运行，它们不在核心态下运行，所以不直接访问硬件。这样处理的结果是：如果文件服务器中发生错误，则文件服务器有可能崩溃，但是整个系统不会导致崩溃。

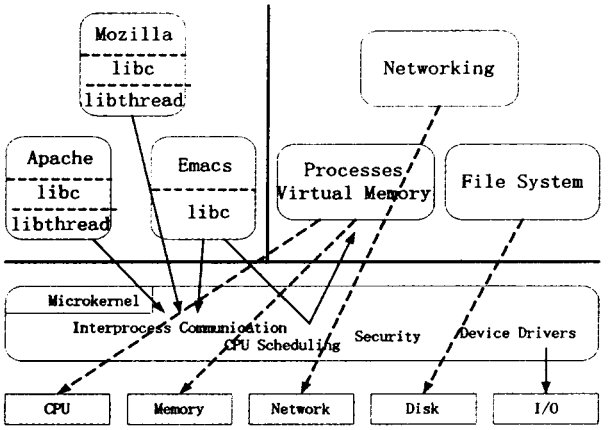


图 4　微内核结构示意图

图 4 中的内核只处理客户机与服务器之间的消息传递，实际的系统与图中的情形不完全符合。一些操作系统的功能（比如在物理 I/O 设备寄存器中写入命令字）单靠用户空间的程序是很难完成的。有两种解决这个问题的方法：一个方法是建立一些运行于核心态的关键的服务进程（例如，I/O 设备驱动程序），它们拥有访问所有硬件的绝对权力，但它们仍然使用通常的消息机制与其他的进程通信。另一个方法是，在内核建立起最小的机制，从而把策略留给在用户空间的服务进程。例如，内核可能会向某一个磁盘上的 I/O 寄存器，用来启动一个读盘操作。在此例子，内核甚至不对该消息的内容进行合法性检查，而只是把它们机械地拷贝进磁盘寄存器。微内核操作系统的代表有：Mach[14]，Chorus[15]，QNX[16]，GNU/Hurd[17]，L4[18]等。

2.5　外核结构[19]

外核结构是操作系统设计中为了获得性能和灵活性的一个极端。它试图将操作系统接口降低到硬件层，从内核中去除所有传统操作系统提供的抽象，并且将重点放在以可获得的硬件资源的复用上。在外核结构中，内核只负责简单的申请、释放并复用硬件资源，而将内存映射、I/O 和复杂的线程包等所有在传统操作系统内核中提供的抽象都转移到用户空间运行（图 5）。

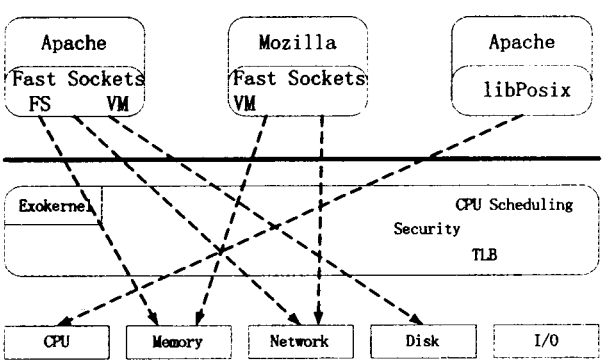


图 5　外核结构示意图

在外核结构中，内核负责三个重要的任务：(1)跟踪资源的所有权，(2)通过保护所有应用或绑定点来实现访问控制从而确保系统安全性没有受到侵犯，和(3)撤回对资源的访问。在核外，所有在传统操作系统内核中的抽象都由用户应用以库的形式提供，如应用级虚拟内存、IPC 等，用户程序通过调用库的形式实现对硬件资源的直接访问。外核结构的操作系统一个典型就是 MIT 的 Exo kernel[19]。

3　操作系统体系结构分析与比较

简单结构的操作系统的优点是结构紧凑、组合方便、对不同的用户环境和要求可以进行裁剪，而且具有较好的灵活性；而且由于各过程间可以直接调用或引用，从而系统效率高。它的主要缺点是系统的结构不清晰，由于各过程之间无规律地相互调用，相互依赖，构成了一个复杂的网络，其调用关系是一个相当复杂的有向图，由于整体性太强，人们难于对结构作出清晰的判断；它的各过程间紧密耦合，对任何情况的修改很可能需要连锁式修改有关很多模块，因此其适应性差，难于对系统进行修改和维护；另外其系统的可靠性低，一方面是因为很难保证这种复杂关系的模块设计的正确性，另一方面是因为各模块之间的网状调用关系会造成死锁；从信息隐藏的角度看，它没有任何隐藏，每一个过程对任何其它过程都可见，因此其安全性也较差。

单体内核结构操作系统的优点是结构简单，容易理解；由于其大部分操作系统模块均在内核中，因此性能较高；并且由于在应用之间的实行的保护机制，因此安全性也较高。缺点是核心组件没有保护；且核心间模块的关系复杂，也就是说实际上整体的结构仍然是复杂的，导致其可扩展性差，灵活性也不够高。

用分层的方法来构造操作系统的好处主要是易于系统调试和验证。第一层是直接构建在硬件裸机之上的，在假设硬件裸机在无误的情况下，可以调试第一层，以确保它的正确性。在得到第一层正确性的保证的情况下，可以调试第二层，以确保第二层的正确性，利用这种层层演进的方法可以验证整个系统的正确性。如果某一个错误在调试某一层的时候被发现，可以确定该错误一定是该层造成的，因为其下的层次已经被调试过，并得到其正确性的保证。通过分层的方法可以简化系统的调试与验证。层次式结构的缺点是困难的系统设计。它的每一层需要严格而仔细的设计，因为每一层只能访问其下层提供的服务来实现自己的功能。在操作系统的内核中，功能模块之间的调用关系是网状的，很难模型化为层次结构，有时为了层次化的结构不得不把更多的功能模块合为一个层次，或者强行将各个模块层次化，这又不可避免的带来了系统性能的下降。由于上述的困难，分层结构的操作系统的研究与开发主要集中于研究领域，但是商业的一些知名的操作系统在设计的时候也大量借鉴了分层的思想。

用微内核结构构造操作系统可以带来较高的灵活性、扩展性以及可靠性，同时又是控制系统的复杂度的有效方法，因此微内核结构得到了广泛的关注，很多著名的商用操作系统都是基于微内核结构构造的。在研究领域，微内核结构一直是操作系统界的研究热点之一。但微内核结构的缺点也是很明显的，因为每次应用程序对服务器的调用都要经过两次核心态和用户态的切换，因为效率比较低。微内核结构的另一个优点是，它适用于分布式系统，如果客户机能通过消息传递和服务器通信，那么客户机不需知道这条消息是在本地机处理还是通过网络送给了远地上的服务器。这两种情况下，客户机对它们的处理都是相同的：发一请求，收一应答。

外核结构一个明显的优点是较快的运行速度，因为应用

是直接访问硬件的；另外一个优点就是可扩展性，因为对于外核结构来说，增加一个新的功能仅意味着在应用层增加一个系统库而已，对内核不需要作任何修改；还有就是灵活性，也是比较明显的，因为对于不同的应用环境和应用需求，所做的修改仅是系统库的配置的不同。外核结构的缺点是安全性较低，因为大量的共享服务被放到应用层，而在应用层的共享服务的安全必须由用户来保护的，显然这就降低了操作系统的安全性。

通过上文中对操作系统主要体系结构的分析，我们可以看出，各种体系结构均有其优缺点，如表1所示。

表1 各体系结构相关特性比较表

	性能	扩展性	复杂度	灵活性	安全性
简单结构	很高	很差	很复杂	很差	很低
单体内核结构	高	差	复杂	差	高
层次式结构	很低	差	很简单	差	高
微内核结构	低	好	简单	好	高
外核结构	很快	好	复杂	很好	低

计算机操作系统要实现的任务主要对计算机硬件资源进行管理并提供给应用相应的接口。操作系统要尽可能提供很高的性能，很好的扩展性，很好的灵活性，很高的安全性并且结构要简单。可从上面的分析我们可以看出，没有一种体系结构能同时满足上述要求，而只能在某些方面能够满足，也就是说目前的体系结构都是上述五个特性的折衷。比如，层次式结构和微内核结构的操作系统的结构都是比较简单的，但是这却牺牲了系统的性能，因为层次式结构在分层调用和微内核结构在通信时会浪费一些时间。而外核结构和简单结构由于应用能够直接访问硬件设备，使得性能大大提高，可是也因此而安全性降低了，同时系统的结构也很复杂。

4 操作系统体系结构研究趋势

目前，有关操作系统体系结构的研究，主要集中在三个方面：

1)将软件方法学中的一些新方法使用到操作系统的设计中，比如使用面向对象[20]、面向 Agent[21]、面向 Aspect[22]技术而构造出的面向对象的操作系统[23]、基于 Agent 的操作系统[24]、面向 Aspect 的操作系统[25]等。实际上使用这样方法后，对操作系统的实现和结构化带来了一定的好处，而对操作系统的体系结构并没有本质的改变，其使用的还是上述的体系结构。

2)对目前已有的体系结构进行优化，试图克服各种体系结构所具有的缺点以及充分发挥各自的优势。比如使用单地址空间或通过选择内核中适当抽象集的方法[26]增加微内核结构的效率，以及对外核结构应用范围的扩展[27]与如何发挥外核结构的优势[28]，如何解决安全性与灵活性的折衷[29]，通过隔离局部错误增加系统的稳定性[30]，以及对 I/O 方式的修改来改善其效率[31]等，同样的结果是无论作了哪方面的优化，必然会造成另外方面的损失。

3)对已有的体系结构进行局部修改，以构造出更适合其硬件环境或应用环境的体系结构。比如现在常见的超微内核(Nanokernel)[32]和极微内核(Picokernel)[33]等。

结语 目前操作系统应用的两大热点，安全的操作系统[34]和嵌入式操作系统[35]。而这两大热点都需要操作系统具有很好的灵活性和可扩展性，而嵌入式则对实时性及效率又有比较高的要求。而在设计安全的或嵌入式操作系统系统的时候，最先考虑的就是所选用的体系结构是否满足需要。通过以上章节的介绍，我们知道目前主流的操作系统体系结构，均各有自己的优缺点，也就是说没有一种体系结构能同时满足所有的需要，比较而言安全的操作系统可能会倾向选择使用微内核结构或者单体内核结构，而嵌入式操作系统可能会倾向选择简单结构或外核结构。而实际上，在具体的应用中要根据具体的硬件环境、安全性要求、灵活性要求和性能要求来对各种特性进行综合的折衷，或者在已有的体系结构上做部分修改，当然如前所述这也是一种权衡；其结果是突出其最需要的一种或几种比较好的特性，而不太需要的特性则是所选体系结构的劣势，这样的体系结构才最适合我们操作系统的需求，最适合我们整个计算机系统的需求。

参 考 文 献

1 Abraham S, Peter B G, Greg G. Operating system concepts, Sixth Edition. John Wiley & Sons, Inc. 2002

2 Hudson K, Ruth A. Operating System Basics/Architecture, 29th Street Press, 1999

3 Bach M J. 陈葆珏译. UNIX 操作系统设计. 北京-机械工业出版社, 2000

4 Foster L R. Palm OS Programming Bible, John Wiley & Sons, Inc., 2002

5 Poole L, Stauffer T. Macworld MAC OS 9 Bible, Hungry Minds, Incorporated, 1999

6 Livingston B, Straub D. Windows Me Secrets, John Wiley &Sons, Inc. 2000

7 Catherine, Wang Lingxia, Yao Bo, Yang Yang, Zhu Zhengyong. A Survey of Embedded Operating System. http://www.cs.ucsd.edu/classes/fa01/cse221/projects/group2.pdf, 2003

8 毛德操，胡希明. LINUX 内核源代码情景分析. 杭州-浙江大学出版社, 2001

9 Jesse Feiler 著. 梁静，等译. Mac OS X 技术大全. 北京-机械工业出版社, 2002

10 Custer H, 程渝荣译. Windows NT 技术内幕. 北京-清华大学出版社, 1993.7

11 McKusick M K, Keith B, Michael J K, John S Q. The Desgn and Implementation of 4.4BSD Operating System Posts and Telecommunication Press, 2002

12 张凤芝，王肃静，宁禄乔. 操作系统原理教程. 北京-北京希望电子出版社, 2002

13 Sevcik K C, et al. Project SUE as a Learning Experience. FJCC 1972,41(I): 331～338

14 Mach R D. In: Proc. of the Workshop on Micro-kernels and Other Kernel Architectures, 1992. 11～30

15 Chorus M R. In: Proc. of the Workshop on Micro-kernels and Other Kernel Architectures, 1992. 39～70

16 Hildebrand D. An Architectural Overview of QNX. In: Proc. of the Workshop on Micro-kernels and Other Kernel Architectures, 1992. 113～126

17 http://www.gnu.org/software/hurd/hurd.html

18 http://os.inf.tu-dresden.de/L4/

19 Engler D R, Kaashoek M F, O'Toole Jr J. Exokernel: An Operating System Architecture for Application-Level Resource Management. In 15th ACM Symposium on Operating System Principles, 1995,12:251～266

20 Booch. Object Oriented Development. IEEE Transactions on Soft-

ware Engineering, 1986,12(2):211～221
21 Shoham Y. Agent-oriented programming. Artificial Intelligence, 1993,60 (1):51～92
22 Kiczales G, Lamping J, Mendhekar A, et al. Aspect-Oriented Programming. ACM Computing Surveys, Dec. 1996,28
23 Cordsen J, Schroder-Preikschat W. Object-Oriented Operating System Design and the Revival of Program Families. In: Proc. of the Second Intl. Workshop on Object Orientation in Operating Systems (I-WOOOS' 91), 1991. 24～28
24 Chen L T. AgentOS: the agent-based distributed operating system for mobile networks, Special issue on networks and distributed systems Winter 1998 (2):12～14
25 Coady Y, Kiczales G, Feeley M. Exploring an Aspect-Oriented Approach to Operating System Code. Workshop on Advanced Separation of Concerns at OOPSLA 2000, Minneapolis, MN, Oct. 2000
26 Liedtke J. On μ-Kernel Construction. In: Proc. of the Fifteenth ACM Symposium on Operating System Principles (Copper Mountain Resort, CO., Dec. 3-6). ACM Press, New York, NY, 1995. 237～250
27 Artiaga A E, Serra A, Gil M. Porting multithreading libraries to an exokernel system. In: Proc. of the 9th workshop on ACM SIGOPS European workshop: beyond the PC: new challenges for the operating system, 2000. 121～126
28 Ganger G, Engler D, Kaashoek M F, Briceno H, Hunt R, Pinckney T. Fast and Flexible Application-Level Networking on Exokernel Systems. ACM Transactions on Computer Systems, Feb. 2002,20(1):49～83
29 Christophe Rippert Protection in flexible operating system architectures, ACM SIGOPS Operating Systems Review, 2003,37(4): 8～18
30 Swift M M, Bershad B N, Levy H M. Improving the reliability of commodity operating systems, In: Proc. of the nineteenth ACM symposium on Operating systems principles, 2003. 207～222
31 Burnside M, Keromytis A D. HighSpeed I/O: The Operating System As A Signalling Mechanism, ACM SIGCOMM 2003 Workshops, Aug. 2003. 25～29
32 Bomberger A C, Frantz W S, Hardy A C, et al. The KeyKOS Nanokernel Architecture. In: Proc. of the USENIX Workshop on Micro-Kernels and Other Kernel Architectures, USENIX Association, April 1992. 95～112
33 Assenmacher H, Breitbach T, Buhler P, Htibsch V, Schwarz R. The PANOA System Architecture－A Pico-Kernel Approach. In: Proc. 4th Workshop on Future Trends of Distributed Computing Systems, Lisbon, Portugal, Sep. 1993. 470～476
34 Zegzhda D P, Stepanov P G, Otavin A D. Fenix Secure Operating System: Principles, Models, and Architecture. MMM-ACNS 2001, LNCS 2052, 2001. 207～218
35 Sztipanovits J, Karsai G. Generative Programming for Emedded Systems. GPCE 2002, LNCS 2487, 2002. 32～49

(上接第 207 页)

数-支持向量回归机中参数 r 与 σ 之间的近似线性反比关系的结论，并进一步得出了 r 与 σ 的关系曲线的斜率随着信噪比增加而减小、整个曲线下移的结果。上述结果印证和丰富和补充了先前的对此问题的理论研究结果，为 r 范数-支持向量回归机在输入信号存在噪声时合理的选择参数以提高其鲁棒性提供了更可信的依据。

参 考 文 献

1 Cristianini N, Shawe-Taylor J. An Introduction to Support Vector Machines[M]. Cambridge University Press, 2000
2 Vapnik V. Statistical Learning Theory[M]. New York: Wiley, 1998
3 Kwok J T, Tsang I W. Linear dependency between ε and the input noise in ε-support vector regression[J]. IEEE Transaction on Neural Networks, 2003(5)
4 Smola A J, Murata N, Schölkopf B, Müller K-R. Asymptotically optimal choice of ε-loss for support vector machines[A]. In: Proc. of the Intl. Conf. on Artificial Neural Networks[C]. 1998
5 Smola A J, Schölkopf B. A tutotial on support vector regression. Royal Holloway College, 1998, NeuroCOLT2 Technical Report NC2-TR-1998-030
6 Law M H, Kwok J T. Bayesian support vector regression[A]. In: Proc. of the English Intl. Workshop on Artificial Intelligence and Statistics[C]. Florida: Key West, 2001. 239～244
7 Gao J B, Gunn S R, Ham's C J. A probabilistic framework for SVM regression and Error Bar Estimation[J]. Machine Learning, 2002,46:71～89
8 Cherkassky V, Ma Yunqian. Practical selection of SVM parameters and noise estimation for SVM regression[J]. Neural Networks, in press, 2003
9 阎平凡，张长水. 人工神经网络和模拟进化计算[M]. 北京：清华大学出版社，2001
10 沈永欢，等. 实用数学手册[M]. 北京：科学出版社，2002
11 朱嘉钢，王士同，杨静宇. 鲁棒的 r-支持向量回归机中参数 r 的选择研究[J]. 控制与决策(已录用，文号 04105)