第 34 卷 第 11 期 计 算 机 工 程 2008 年 6 月
Vol.34 No.11 Computer Engineering June 2008
·博士论文· 文章编号：1000—3428(2008)11—0001—03 文献标识码：A 中图分类号：TP316

基于服务体/执行流模型的操作系统

陈香兰，龚育昌，张 晔
(中国科学技术大学计算机科学与技术系，合肥 230026)

摘 要：介绍一种新的操作系统抽象模型——服务体/执行流模型(SEFM)。在该模型中，数据存储抽象与数据运算抽象相互分离，计算模型直接对应于物理 CPU 执行程序的过程。给出一个基于 SEFM 的操作系统——MiniCore 的功能结构。该系统与其他主流操作系统进行比较测试的结果表明，MiniCore 的同步消息通信与 Linux 管道通信相比效率高出 3 倍多，比 Win98 管道高出 62 倍多，MiniCore 的网络通信效率与 Linux 相当。通过不同负载下视频解码播放的时延分布可以看出，MiniCore 比 Linux 具有更好的实时性。
关键词：服务体；执行流；消息推动通信；引流机制；存储地址空间

Operating System Based on Servant/Exe-Flow Model

CHEN Xiang-lan, GONG Yu-chang, ZHANG Ye
(Department of Computer Science and Technology, University of Science and Technology of China, Hefei 230026)

【Abstract】 Servant/Exe-Flow Model(SEFM) is a novel abstraction of operating system. In SEFM, data storage and data computing are separated from each other, and the abstraction of computing directly corresponds to the execution locus of program on physical CPU. The organization and the comparison testing results of SEFM based operating system MiniCore are given. Testing results show that the efficiency of synchronized communication in MiniCore is 3 times higher than pipes in Linux, and 62 times higher than pipes in Win98, and the delay distributions of video decoder under different load conditions show that MiniCore has better real-time performance than Linux.
【Key words】 servant; exe-flow; message-pushing communication; flow-lead-in mechanism; storage addressing space

1 概述

传统的进程/线程模型将数据存储抽象与数据计算抽象融为一体，导致进程间通信的低效率和线程维护与调度开销过大。当前的主流操作系统大多采用宏内核结构，内核中放弃了进程/线程抽象，内核模块间采用函数调用方式进行交互，效率和实时性很高。但由于将所有系统服务都以紧耦合的函数调用方式集中置于内核中，因此严重损害了系统的可剪可扩性和可维护性。微内核结构将大部分系统服务置于内核之外的用户态，采用进程/线程抽象，可扩可剪性和可维护性良好。但进程/线程模型固有的低效率性阻碍了微内核结构的实际应用。

为克服上述缺陷，学术界开展了多方面研究。文献[1]提出利用跨地址空间调用实现快速的函数调用IPC技术；文献[2]提出了Container/Loci概念，以求摆脱进程/线程模型的制约，但尚未形成完整性的成果；L4 为C/S模型中频繁的同步通信提供专门的系统调用Call和reply&receive，将direct control transfer和fast message mapping技术相结合以实现高效的IPC[3]。上述工作均未彻底摆脱进程/线程模型的制约，在试图改进效率的同时破坏了进程与线程间的层次语义，从而导致了各自的新问题。在分析、总结上述研究成果的基础上，本文对操作系统做了新的抽象，提出了服务体/执行流模型(Servant/Exe-Flow Model, SEFM)[4-6]及基于该模型的操作系统构造模型，进而开发了基于SEFM的原型系统并验证了所提模型的正确有效性。

2 SEFM 的基本抽象

SEFM 以服务体为数据存储抽象，以执行流为数据计算抽象，存储模型与运算模型相互分离。

定义 1 执行流(exe-flow)是物理 CPU 连续执行机器指令的轨迹。从机器上电复位到关机，执行流不会被阻塞、挂起和中止，也不与某地址空间绑定。

由定义 1 可知，执行流与机器的运行有直接映射关系。以执行流为运行模型时，系统运行只存在执行流引导问题，而不像采用线程作为虚拟 CPU 那样，需要进行线程创建、维护、撤销和线程切换等复杂操作。

定义 2 服务体(servant)是具有一定功能的代码与数据的集合。服务体是静态和持久的，其生命周期不依赖于执行流，并拥有自己的地址空间。服务体中包含若干处理例程以实现某些服务，通过规定的通信接口(称为端口)与外界交互。

定义 3 服务(service)是对服务体某功能的引用。

由定义 2 和定义 3 可知，服务体是服务的载体。服务的请求和应答只能通过端口进行。执行流也只能通过端口流入服务体，并推动服务的运行。

推论 1 由于服务体间只能通过规定的通信接口交互，因此系统具有良好的可剪可扩性和可维护性。

推论 2 由于服务体是静态的，执行流只能通过端口从一个服务体直接流入另一服务体，因此服务体间的通信是主动的、直接的，可以实现同步通信，从而避免进程通信所需的

基金项目：高校博士点基金资助项目(20050358040)；安徽省自然科学基金资助项目(070412030)
作者简介：陈香兰(1977－)，女，博士研究生，主研方向：操作系统；龚育昌，教授、博士生导师；张 晔，博士研究生
收稿日期：2007-06-10 **E-mail：**xlanchen@ustc.edu.cn

巨大开销。

推论 3 由于服务体是静态和持久性的，因此基于 SEFM 构造的操作系统的运行过程就是执行流在相关服务体中“流动”的过程，不存在进程创建、撤销和进程状态转换等复杂的操作。

3 基于SEFM的操作系统构造模型[4]

在 SEFM 中，服务体是系统的基本组成单位，各种系统功能组件以及用户程序都以服务体的形式存在。服务体间通过明确定义的通信接口——端口间的消息推动通信进行服务请求和应答。基于 SEFM 的操作系统构造模型如图 1 所示。

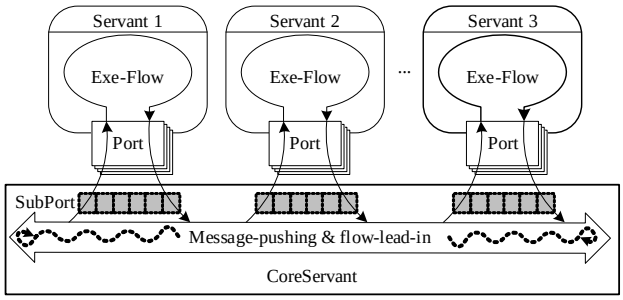

图 1 基于SEFM的操作系统的基本结构

3.1 核心服务体(CoreServant)

核心服务体是基于 SEFM 的操作系统的中枢组件，提供服务体间通信机制、并发引流机制、中断和异常处理等基础服务和多服务体并发控制，与其他类型服务体统一采用服务体构造模式和通信方式。

3.2 端口(Port)

定义 4 端口是服务体间进行服务请求和应答的通信接口，一个服务体可提供多个端口以响应不同的服务请求。

服务体通过端口局部名对端口进行授权访问。对一个端口的权限(Capability)包括：(1)消息发送权(C_S)：向一个端口发送服务请求的权限。任何一个服务体在请求一个端口提供的服务前，必须获得对该端口的C_S。(2)消息接收权(C_R)：接收发往某个端口的服务请求并执行服务的权限。C_R只属于端口发布者。(3)发送权授权权(C_A)：将一个端口的C_S授权给其他服务体的权限。对象管理器持有所有端口的C_A。

核心服务体在消息通信过程中和局部名授权操作时负责权限的检查。一个服务体在注册该服务体所属的端口时获得对该端口的所有权利。对象管理器在上述过程中获得对该端口的发送权授权权，但不具有消息发送权。当且仅当对象管理器主动打开一个端口时，才能获得对该端口的消息发送权。

3.3 并发引流机制

SEFM 采用并发引流机制处理服务体并发，根据一定的规则(如优先级、时间片)选择一个服务体，将执行流引入其中，并用引流端口保存执行流的状态。

定义 5 引流端口(简称小端口，Subport)是并发引流机制中的重要组成部分，用来记录执行流的当前上下文。小端口至少包括：(1)一个堆栈；(2)一个寄存器上下文结构；(3)小端口所处的优先级和时间片信息(包括基本时间片和当前剩余时间片)。

服务体的并发实际上就是服务的并发，包括：(1)不同端口的不同服务并发；(2)同一端口的不同服务并发；(3)同一端口的同一服务的并发。核心服务体在每个服务请求到达时为相应的服务分配一小端口以记录执行流的当前上下文和状态，从而实现端口的重入和并发。

3.4 服务体地址空间

服务体地址空间采用二级多地址空间技术，如图 2 所示。每个服务体地址空间都被划分为基本空间和扩展空间。每个服务体可以拥有多个扩展空间，从而使服务体的数据容量突破硬件对虚存空间的限制。其中，①代表服务体私有数据代码；②代表所有扩展空间共享的代码数据；③代表系统范围内共享的数据和代码。

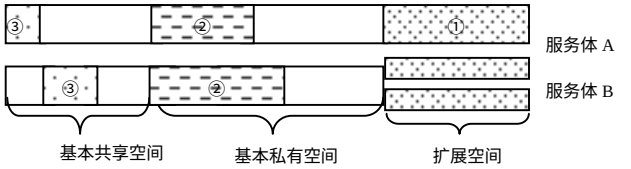

图 2 服务体地址空间结构

基本空间又进一步划分为基本共享空间和基本私有空间。每个服务体基本私有空间只能供本服务体使用，而基本共享空间则可供所有服务体共享。

基本共享空间使用单地址空间技术管理并通过capability机制[7]授权共享。不同服务体对这段空间有不同的视图。基本共享空间可以用来安全地共享复杂的数据结构。

3.5 消息推动通信机制

本模型采用主动、同步的消息推动通信作为服务体间统一的通信方式。

定义 6 消息推动通信指将消息从源信息主体的地址空间直接推入目标通信主体的地址空间的通信方式。消息推动通信仅提供一个统一的消息推动函数为编程原语：

msg_body_t* push_msg (msg_body_t* hdr, msg_option_t option)；

在基于 SEFM 的系统中无消息接收函数，同步和异步消息都被直接推入目标端口并作为参数传递给目标服务例程。消息推动通信提供 3 种通信方式：同步连续，同步分离和异步。由原语中的 option 参数指定。

同步连续式通信实现受保护的跨地址空间调用的语义，如图 3(a)所示。同步分离相当于一次滞后跳转，包括指定应答端口和直接跳转两种情况，如图 3(b)和图 3(c)所示。

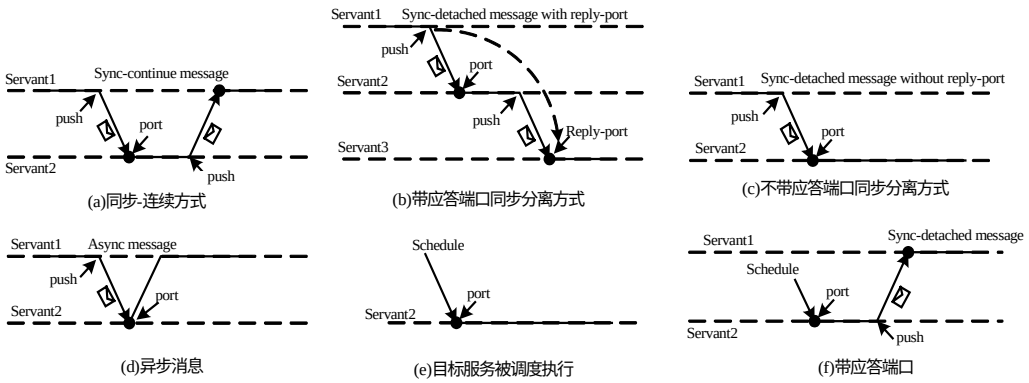

图 3 消息推动通信

异步通信方式是通过同步消息通信模拟实现的，其语义与传统的异步通信相似。消息送达目标服务体后，执行流立即返回服务请求者，如图 3 (d)所示；而目标服务由系统的引流机制调度执行。异步消息也可以指定目标服务结束后的应答端口，如图 3(f)所示。通过主动的消息推动通信机制，执行流能够将具有同步通信关系的服务体有序连贯起来，实现即时高效的跨地址空间通信，避免不必要的调度开销和延时。

4 基于 SEFM 的操作系统

本文设计实现了基于 SEFM 的操作系统原型 MiniCore，其组成按功能类别划分为基本机制层、服务层和运行环境层 3 个层次。

基本机制层为系统运行提供基础机制，包括：(1)核心服务体，提供 SEFM 的抽象和端口调度、服务体管理、消息推动通信等机制；(2)内存管理服务体，实现服务体存储模型、内存分配等功能；(3)体系相关服务体，实现底层硬件抽象。

服务层提供基本的系统服务功能，包括对象管理服务体、IO 管理服务体、文件系统服务体以及包括网络协议在内的多种驱动服务体。其中，IO 管理服务体实现系统的驱动程序和设备管理模型，对 IO 设备和文件系统进行管理；对象管理服务体实现对系统中各种对象包括端口在内的名字到局部名的映射。

运行环境层为用户提供各种编程模型和程序运行环境。其中，Linux 运行环境和其他运行环境用于实现对现有操作系统二进制应用程序的兼容；本地运行环境则提供面向服务体/执行流模型的用户编程环境。

5 性能测试与比较

实验 1 Echo 服务器

Echo 服务器提供最简单的数据传送服务：接收客户发送的 2 KB 原始数据并将该数据作为结果发送回去。测试方法为，客户端在发送消息前记录时间 $t1$，在接收到服务端的 Echo 消息后记录时间 $t2$，计算 $t2$-$t1$，统计 1 000 次，计算平均时间。测试结果如表 1 所示。

表 1 不同操作系统的 IPC 性能对比

操作系统	测试内容	平均运行时间/s
MiniCore	同步消息推动	9.48
	异步消息推动	36.09
VxWorks	Pipe	23.37
Linux	Pipe	39.01
Win 98	Pipe	595.89

测试是在相同的硬件平台上进行的。由表 1 可看出，MiniCore 的同步消息推动通信的效率比 VxWorks 高 1.5 倍，比 Linux 高出 3 倍，而比 Win98 高出 62 倍。这是由于同步消息推动避免了采用 IPC 时的时间延迟和调度开销。由于 VxWorks 使用了单地址空间技术，因此效率比 MiniCore 采用异步通信时高。

实验 2 路由器

将 MiniCore 配置成 MiniCore-for-router 路由系统，路由软件采用 NATxp。路由器采用 PCM6892 单板机。测试结果见表 2。可见，当 MiniCore 配置成路由系统时，其性能与基于 Linux 的路由系统性能相近。

表 2 不同操作系统支持下的路由性能对比

路由系统	传输速度峰值/(Kb·s^{-1})
MiniCore-for-router	10.13
Linux+iptables	10.16
Win 98+Wingate	7.50

实验 3 视频解码播放

实验采用 PCM6892 单板机，在不同负载条件下统计 1 000 帧视频解码播放的时延分布，测试结果见表 3，其中，时间延迟为任务完成时间与任务截止期之差，负数表示在任务截止期之前完成，正数表示在任务截止期之后才完成。

表 3 两种 OS 下 1 000 帧视频解码播放的时延分布比较

时间延迟/ms	硬实时发生帧数		软实时发生帧数	
	Linux	MiniCore	Linux	MiniCore
－35 以下	0	0	0	0
－35～－30	17	40	14	11
－30～－25	435	523	282	181
－25～－20	336	413	211	243
－20～－15	161	24	101	337
－15～－10	47		50	133
－10～－5	4		94	89
0～5			128	6
5～10			102	
10～15			17	

可以看出，在硬实时条件下(系统负载为 20%)，Linux 和 MiniCore 中所有帧的解码播放都在截止期前完成，但 MiniCore 的完成时间在整体上比 Linux 早。

在软实时条件下(系统负载为 60%)，Linux 出现截止期错失的情况，占总数的 12%，若使用肉眼观察则会看到某些帧出现迟滞现象。而在 MiniCore 下未出现这种情况，这表明 MiniCore 的实时性更好。

6 结束语

服务体/执行流模型(SEFM)将数据存储的抽象与数据计算的抽象相分离，采用直接对应于物理 CPU 执行过程的计算模型和具有静态、持久性的存储模型，从而使基于 SEFM 的操作系统能够采用高效率的同步消息推动通信方式和灵活的存储地址空间管理方式，使基于 SEFM 的操作系统能够在效率和实时性等方面的性能与宏内核结构的主流操作系统相当，而在可扩展、可剪裁性以及可维护性方面与微内核结构操作系统相当。对比实验结果验证了上述结论。

参考文献

[1] Bershad B N, Anderson T E, Lazowska E D, et al. Lightweight Remote Procedure Call[J]. ACM Transactions on Computer Systems, 1990, 8(1): 37-55.

[2] Rosenberg J, Dearle A, Hulse D, et al. Operating System Support for Persistant and Recoverable Computations[J]. Communications of the ACM, 1996, 39(9): 62-69.

[3] Liedtke J. Improving IPC by Kernel Design[J]. Operating System Review, 1993, 27(5): 175-187.

[4] 吴明桥, 陈香兰, 张 晔, 等. 一种基于服务体/执行流的新型操作系统构造模型[J]. 中国科学技术大学学报, 2006, 36(2): 230-236.

[5] 龚育昌, 陈香兰, 李 曦, 等. 基于服务体/执行流模型的操作系统[J]. 中华人民共和国发明专利公报, 2005, 21(37): 209-215.

[6] 李 宏, 陈香兰, 吴明桥, 等. 服务体模型与操作系统内核设计技术[J]. 计算机研究与发展, 2005, 42(7): 1272-1276.

[7] Shapiro S. EROS: A Fast Capability System[J]. Operating System Review, 1999, 33(5): 166-176.