大规模并行操作系统研究

邵宗有[1,2] 王昭顺[1] 许建卫[3]

(北京科技大学信息工程学院 北京 100083)[1] (曙光信息产业有限公司研发中心 北京 100094)[2]

(无锡城市云计算中心有限公司 无锡 214315)[3]

摘 要 通过对大规模并行超级计算机模拟器上运行的操作系统进行分析,发现超级计算机内部计算节点、I/O节点和服务节点对操作系统各模块的需求是不同的,一个通用的操作系统无法满足不同类型节点的需求。提出了一种轻核心结构的操作系统并在模拟器上实现了原型系统SandOS,它包括计算节点轻核心SandPOS、I/O节点操作系统FileServer和服务节点操作系统MonitorServer。对计算节点轻核心SandPOS与通用的操作系统在内存开销、调度效率、运行效率、页表管理等方面的性能进行了分析对比,结果显示出超大规模并行系统中,SandOS比通用的操作系统具有更好的执行效率。

关键词 大规模,轻核心,操作系统 SandOS

中图法分类号 TP303 **文献标识码** A

Study of the Large Scale Parallel Operating System

SHAO Zong-you[1,2] WANG Zhao-shun[1] XU Jian-wei[3]

(School of Information Engineering, University of Science and Technology Beijing, Beijing 100083, China)[1]

(Research and Development Center of Dawning Information Industry CO., LTD, Beijing 100094, China)[2]

(Wuxi Cloud Computing Center, Wuxi 214315, China)[3]

Abstract By analyzing the operating system running in a supercomputer simulator, it is found that different nodes in the supercomputer use different part of an operating system. However, a traditional operating system cannot meet the demand. In such case, a heterogeneous large scale operating system is designed and a prototype named SandOS is implemented based on the simulator. The prototype includes light weight kernel operating system SandPOS for compute nodes, FileServer for I/O nodes and MonitorServer for service nodes. By comparing SandPOS with traditional operational system in memory overhead, schedule efficiency, run efficiency and address translating efficiency, result shows that in large scale parallel system, SandOS has better efficiency than traditional operational system.

Keywords Large scale, Light weight, Operating system, SandOS

1 引言

目前大规模并行系统的发展呈现出新的特点,首先是节点个数飞速增长,比如首次突破万亿次的ASCI red[13]有4千多节点,首次突破百万亿次的BlueGene/L[10]有6万多个节点,最新的千万亿次天河-2[19]超级计算机则有312万计算核心;其次系统中不同节点间的功能区分逐渐明确,甚至不同功能的节点结构特点也不相同。一般来说,超大规模并行系统中的节点可以分为计算节点、I/O节点和服务节点3类[10,11,13],其中计算节点聚集了整个系统的计算资源,其特点是数目巨大,但I/O能力和存储能力相对较小,通过高速网络将I/O请求转发到I/O节点;I/O节点提供系统中几乎全部的I/O功能,其特点是外设丰富,但数目相对较少;服务节点则提供系统的登录管理、作业管理等功能,数目相对较少。在即将到来的超大规模万万亿次超级计算机中,上述区分会更加明显。典型的超大规模并行系统结构中,计算节点之间通过高速互联网络连接,可进行计算节点间快速通信。计算节点和服务节点、I/O节点间分别通过服务网络和I/O网络互联,可传递管理信息或I/O信息。

在这种新的形势下,如何高效地管理和使用硬件资源,尤其是如何充分发挥大量计算节点的性能将对整个系统的性能产生重大影响。传统大规模操作系统在计算节点和I/O节点上都采用标准的UNIX/Linux:比如IBM的ASCI white系统采用AIX操作系统[14],美国Sandia国家实验室的Cplant机群系统[15]、LLNL的MCR机群系统[12]和SGI公司的Altix 3000系统[16]都采用linux操作系统。标准UNIX/linux对硬件支持范围广,并且GNU提供了大量的软件和库,所以在一个新的系统上移植和维护UNIX/linux比较容易。

但在计算节点上采用标准的UNIX/linux随着系统规模的进一步增大逐渐暴露出功能和效率方面的不足,在即将到

邵宗有(1976—),男,博士生,高级工程师,主要研究方向为信息安全、计算机系统结构,E-mail:shao@sugon.com;王昭顺(1969—),男,博士,博士生导师,主要研究方向为信息安全、软件工程、计算机系统结构;许建卫(1978—),男,博士,高级工程师,主要研究方向为网络安全、计算机系统结构。

来的万万亿次计算机上这种不足会更加明显:科学计算程序具有节点任务单一、计算密集的特点,而 linux 设计中多任务支持、权限保护、支持丰富的 I/O 等功能是不需要的,它们将导致计算节点效率下降;而且,标准的 linux 并不是针对高性能计算而开发的操作系统,因此有些功能是和科学计算应用程序的需求相反的。比如 linux 的越界保护(out of memory killer)功能会在内存资源紧张时将使用内存最多的进程杀死。而在高性能计算机中,用户希望业务系统把硬件资源使用得越充分越好,因此用户希望进程可以尽可能多地使用内存,甚至是全部的内存,但这个需求和 linux 的实现恰恰相反。并且,由于 linux 内核比较庞大,因此修改 linux 内核来增加或删减一些功能需要花费较大的精力。随着 linux 内核的不断更新,之前所做的修改与 linux 的同步升级也需要较高的代价。

鉴于上述原因,近年来一些并行系统计算节点采用专门定制的轻核心操作系统,比如 ASCI red 系统上计算节点采用 cougar 操作系统[13],BlueGene/L 系统计算节点采用 BLRTS 操作系统[10],Cray XT3 计算节点采用 UNICOS/lc 操作系统[11]。这些轻核心操作系统的设计目标是让科学计算程序尽可能高效地使用系统资源,从而最大限度地发挥单节点的性能,并且单个计算节点上应用程序运行时要尽量避免噪音,使得系统的整体性能得到提高。但这些操作系统都是针对特殊硬件平台的商用操作系统,我们无法直接使用,因此,需要对超大规模操作系统进行研究。

由于超大规模并行系统价格高昂、操作复杂,因此使用真实硬件来研究超大规模操作系统代价很大。并且,受限于硬件系统固有限制,不同硬件配置下操作系统的行为分析也比较困难。因此,本文基于模拟器对超大规模并行操作系统进行研究,设计实现了一个采用轻核心结构的超大规模并行操作系统原型系统 SandOS。

本文第 2 节介绍相关工作;第 3 节对 SandFox 模拟器进行介绍;第 4 节基于 SandFox 模拟器设计并实现了轻核心操作系统原型 SandOS;第 5 节给出 SandOS 中轻核心 SandPOS 与 linux 系统在内存开销、调度效率、运行效率和内存管理方面的性能对比;最后给出总结与下一步工作。

2 相关工作

IBM 公司开发的 Blue Gene 大规模并行集群上的计算节点最初采用轻核心操作系统 CNK[1],它采用微内核结构,每个 CPU 核上仅有一个用户线程,并且采用虚拟地址和物理地址直接映射的方法来加速虚实地址转化的过程,优化了内存访问效率。后期,IBM 公司又采用精简 linux 内核的方法[2,4],将精简后的 linux 直接运行在开源工程 ZeptoOS[17] 上,在内存效率和中断处理方面取得了和 CNK 相近的结果,但又可以保持和现有标准 linux 的兼容性。但不论是 CNK 还是基于 ZeptoOS 精简的 linux 操作系统,都是针对 IBM 的 PowerPC450 CPU 定制开发的,并重点优化了 Blue Gene 系统实际运行中的 DMA 访问问题,这些定制特性降低了该系统的适用性。

PUMA/Cougar[5] 是已经投入使用的商用轻核心操作系统,用在 Intel Paragon[7] 和 ASCI Red[9] 计算节点上,它的前身是 1991 年美国 Sandia 国家实验室和新墨西哥州立大学共同开发的 SUNMOS 操作系统。PUMA/Cougar 面向超大规模并行系统,它只支持单用户进程模式,采用一次性加载程序的方法,并且为了减少换页开销,不支持按需调页功能。它采用微内核结构,主要包括 Q-kernel 和 PCT 两部分,其中 Q-kernel 负责对底层硬件封装,PCT 负责进程调度。PUMA/Cougar 系统的通信基于 portal 的概念,portal 指的是接收者进程开放自己的地址空间,发送进程可以直接将消息放入接收者开放的地址空间。由于不提供 linux 兼容 ABI,PUMA/cougar 上无法直接使用标准库,用户程序也需要做专门的移植或修改,这样带来的直接影响是 PUMA/cougar 上的系统软件更新缓慢,而过时软件的低效抵消了轻核心带来的优势[6]。

OSF/1 AD[5] 是 DEC 公司 1992 年发布的基于 MIPS 体系结构的操作系统,它基于 mach 微内核开发,对 mach 系统在实时性方面做了很大的增强。为了提供 UNIX 兼容,它开发时引入了一部分 BSD 操作系统。它使用 mach 的任务管理、进程间通信和内存管理等功能,通过一个轻核心的用户级库将 UNIX 进程映射到 mach 上,对 UNIX 系统的兼容则通过一个模拟库来实现。它的特点是支持共享内存的多处理器,为整个系统提供单一系统映像,并且提供了对线程支持。OSF 提供了较好的 UNIX 兼容性,但 OSF/1 的主要缺点是运行时占用内存较大,由于计算节点内存资源相对较少而且节点数据巨大,因此较大的内存消耗限制了 OSF 的进一步发展。

UNICOS[11] 是 Cray 超大规模并行系统上使用的操作系统。最初的 UNICOS 是由 COS 而来,后续的 UNICOS 版本并没有严格继承 UNICOS,比如 T3D 上使用的 UNICOS MAX 基于 mach 微内核,T3E 上使用的 UNICOS/mk 使用 chorus 微内核,X1 上使用的 UNICOS/mp 基于 IRIX,XT3 上使用的 UNICOS/lk 则是基于 cougar/linux。纵观 UNICOS 各版本操作系统,它们都采用了微内核结构,UNICOS/lk 则和前述的 cougar 类似,除 UNICOS/lk 外的其它版本没有区分计算节点和 I/O、服务节点,而轻核心系统用在 I/O 和服务节点上并不合适。

Palacios[3] 是美国新墨西哥州实验室开发的支持大规模机群的虚拟机系统,它在硬件层上提供最基本的封装,采用将硬件设备直接映射到宿主操作系统的方法来加速系统执行,但 Palacios 最终必须依赖宿主操作系统才可以运行用户程序。

3 SandFox 模拟器

SandFox[8] 模拟器是中国科学院计算技术研究所设计开发的并行计算机体系结构模拟器。它采用的是模块化开发的路线,将整个系统用不同层次的基本模块逐级组合而成,包括处理器级、节点极、物理系统级、全系统级 4 个层次。为了满足对目标系统的资源按需分配、资源部件管理及资源部件的可重构,SandFox 模拟器从设计到实现过程中充分考虑了功能的灵活性及模拟器自身的扩展性。因此,SandFox 模拟器的 4 个层次均可以独立运行,分别适用于处理器级、节点级、互联级、全系统级别的体系结构研究及相关软件的开发、调优工作。

SandFox 模拟器的处理器模拟器 SandUPSim[18] 实现了

MIPS32/64 全指令集，并与标准的 MIPS 处理器 ABI 兼容，因此 MIPS 指令集的应用程序可以不加修改直接在 SandFox 模拟器上运行。为了对程序执行的时间进行细粒度分析，SandUPSim 支持时钟级模拟和功能级模拟两种方式。在 SandUPSim 的基础上，增加远程内存访问模块 BMMU，形成节点模拟器 SandMP。SandMP 提供了灵活的资源配置策略，便于研究不同的 CACHE、内存配置下操作系统或应用程序的行为。基于 SandMP，增加远程节点的外设访问模拟模块，就组成了多节点物理系统模拟器 SandBoss。在 SandBoss 物理模拟器之上，增加系统配置管理相关的模块，并运行操作系统 SandOS，便实现了全系统模拟器 SandFox，如图 1 所示。

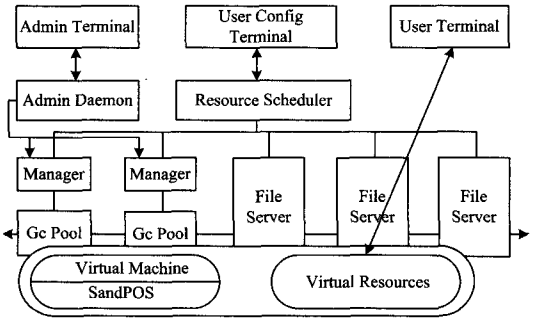

图 1　SandFox 结构图

4　SandOS 设计与实现

千万亿次超大规模系统中操作系统有 4 个方面的需求：1）计算节点操作系统高效率；2）操作系统整体高可扩展性；3）节点间高效的通信机制；4）系统的高容错性。SandOS 将 CPU 和内存资源尽可能多地用于用户程序，通过功能分解和功能简化来实现高效率的计算节点轻核心操作系统；通过分布式结构设计来满足高可扩展性需求；通过采用简化通信协议等方法来实现节点间高效通信；通过对进程的容错迁移来提高系统的健壮性。

为了提高系统性能，应该尽可能地把系统资源用来运行用户程序，所以计算节点轻核心操作系统主要追求精简和高效。SandOS 中通过功能分解和功能简化两个方面来实现上述目标，功能分解是指把不经常使用的功能都从计算节点操作系统中剪裁掉，放在 I/O 节点和服务节点操作系统上；功能简化是指对于需要在本地实现的功能，尽量简单高效地实现。

操作系统最重要的功能包括进程管理、内存管理和文件管理。对于计算节点来说，在运行大规模科学计算任务时往往每个节点上只有一个任务，因此传统操作系统中对多进程的支持可以简化为单用户进程的管理模式。在单进程支持的前提下，传统操作系统中为了支持多任务而采取的多级页表机制、按需调页机制也可以简化，可以采用单一页表并将地址空间直接连续映射的方法来管理内存，这种方式不但可以使得内存管理更为高效，还能为大页面映射带来机会。文件管理对于计算节点来说，是必要的功能但却不是常用的功能，因此我们采用功能分解的方法将文件管理功能放到 I/O 节点上进行。

功能简化后的计算节点轻核心操作系统是一个类微内核结构，核心由两个线程组成：内存管理线程和进程管理线程。其中内存管理线程用来管理物理内存，处理用户进程的内存请求；线程管理线程负责用户进程的创建、调度和销毁。

为了做到对现有应用的兼容，功能分解通过在内核层截获并分析系统调用来实现。首先，将所有系统功能分为本地服务和非本地服务，其中本地服务主要包括进程管理和内存管理相关的服务，而非本地服务主要指 I/O 服务。然后根据系统服务的类型将请求分发处理，处理流程如图 2 所示。

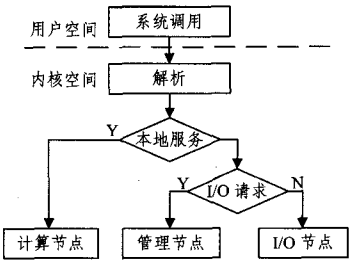

图 2　功能分解

分解后的 SandOS 结构如图 3 所示，包括计算节点轻核心操作系统 SandPOS、I/O 节点操作系统 FileServer 和服务节点操作系统 MonitorServer 3 个部分，3 个部分之间采用松耦合的连接方式，通过消息进行交互，这种情况下，SandPOS、FileServer 和 MonitorServer 都可以根据需求独立进行扩充，从而提高了整体系统的可扩展性。

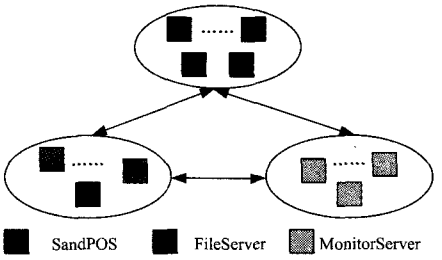

图 3　SandOS 结构图

程序运行过程中需要不断地在计算和系统调用这两个状态之间切换，因此 SandOS 中 SandPOS、FileServer 和 MonitorServer 三者之间通信非常密集，而且对于阻塞型系统调用，必须等待系统调用结果返回后程序才能继续运行，因此三者之间需要采用高效的通信协议。

SandOS 中采用自定义的三层通信协议栈，如图 4 所示。其中应用层之间传递的是用户语义的信息，包括消息包目的地址、长度以及消息包内容，数据包的长度不限。传输层将数据进行分包，并加上传输层的目的地址、本次包长以及包标识符（起始包、中间包、终止包）等。链路层提供无差错的端到端的数据传输。相比 TCP/IP 的多层通信协议，SandOS 通信更为简洁高效。

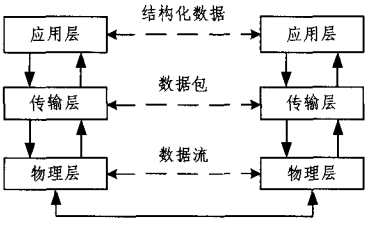

图 4　通信栈

4.1　SandPOS

计算节点效率是超大规模计算机系统的首要问题，因此 SandPOS 是本文研究的重点，基于 SandFox 模拟器来开发；FileServer 和 MonitorServer 只做关键技术验证，所以基于

linux 环境开发，作为一个普通程序运行在 linux 系统上。

由于文件管理由 FileServer 完成，SandPOS 主要负责进程管理和内存管理。通过对现有操作系统中的进程管理和内存管理的简化，SandPOS 实现了一种“精简”的轻核心。

单用户进程

Linux 中的守护进程会给程序的执行带来很大的不确定性因素，对于紧耦合的系统，某个进程的影响会扩散到整个系统，从而对整个系统的性能造成很大影响。因此计算节点只支持单用户进程，单用户进程可以简化进程的管理和调度，省去多用户进程间切换开销，并减少系统的守护进程。

直接创建用户进程

Linux 系统采用 fork/exec 的方式来创建用户进程。由于 SandPOS 是单用户进程系统，所以上述方式费时而且没有必要。在 SandPOS 的实现中，收到创建用户进程的请求就直接创建一个新的用户进程。

地址空间直接映射

考虑到多用户进程随机进入系统，并且对内存需求有很大的不确定性，传统的 Linux 系统采用按需调页的机制，而计算节点上的操作系统则没有这个需求；并且计算节点上没有磁盘，因此按需调页开销比较大。因此 SandPOS 在加载用户进程映像的时候一次性全部加载并直接建立内存映射，这样可以尽量避免缺页异常引起的开销。

地址空间连续映射

现在的处理器都支持大页面映射，因此我们在 SandFox 模拟器中将 SandUPSim 中 TLB 表项大小设置为 16MB。由于 SandPOS 中进程空间采用直接映射的方式，所以内存布局在程序开始运行时已经确定，这样 SandPOS 在建立页表影射的时候将逻辑上连续的地址空间影射到物理上连续的空间。这种方式有利于使用大页表影射，从而减少 TLB 失效次数。

单级用户页表

单级用户页表查找效率高，但占用内存空间较大。Linux 是多进程系统，如果使用单级页表，则系统中所有进程的页表将会占用较多的内存。但由于 SandPOS 是单用户系统，因此采用单级页表不会带来大内存开销，比如对于 32 位 MIPS 系统来说，即使采用 4kB 的物理页面，2GB 的用户空间建立单级用户页表的开销也只是 2MB(每个页表项 4 个字节)。将 SandUPSim 的表项调整为 16MB 时，页表仅 128 项，可以完全放入 TLB 中。单级页表带来的好处是省去了多级地址变换的开销，可以提高地址变换的效率。

必要时调度策略

SandPOS 是单进程系统，所以无需在多个用户进程之间进行调度。同时内核中没有 linux 中的守护进程，因此用户进程可以一直运行，直到进行系统调用为止。在真实硬件中仍然会存在时钟中断，强制将作业从用户态切换到内核态。通过在 SandFox 中降低时钟中断频率，甚至直接关闭时钟中断功能，可以将进程调度和切换的次数减到最少，把系统资源尽可能多地留给用户进程。

用户态网卡

通过调整 SandFox 模拟器中 BMMU 映射机制，将高速网卡映射在用户态，并在 SandPOS 中为其建立相应的页表项，就可以实现在用户态直接访问远程网卡。这样可以避免每次进程间通信时进行的系统调用，从而提高程序运行的性能。

4.2 I/O 节点操作系统 FileServer

FileServer 提供系统中并发文件访问，本文中基于标准 linux 进行剪裁来实现，实现中重点解决的问题如下：

• 系统调用号的映射和参数的转换。由于 SandPOS 和 FileServer 运行在不同的硬件体系结构上，所以相同的系统调用的系统调用号存在差异，相应的系统调用参数也不尽相同。

• 并发控制和状态维护的问题。由于 I/O 节点数目小于计算节点数目，因此 FileServer 需要同时处理来自多个 SandPOS 的请求，这样就需要解决并发控制的问题。同时由于要系统遵循 linux 兼容文件接口，而 linux 使用有状态的文件系统，所以 FileServer 端需要维护 SandOS 中各个 SandPOS 的文件状态信息。

• 数据传输问题。由于不同体系结构下对数据的表示方式不同，所以需要使用体系无关的数据表示方式来在 SandPOS 和 FileServer 之间进行数据传输。

4.3 服务节点操作系统 MonitorServer

MonitorServer 接收用户请求和 SandPOS 端的标准输入输出请求，可以使用标准 linux 稍加修改来实现。由于系统中同时可能存在多个用户，所以它需要维护多个用户和多个 SandPOS 的对应关系。

负载是否平衡也是影响超大规模系统性能的重要因素，如果某个节点负载过重而导致程序运行较慢，则会影响整个系统的性能。SandOS 提供了进程的动态负载迁移功能，MonitorServer 通过检测系统负载情况，可以通过 SandFox 发出负载迁移命令将应用迁移到更适合的计算节点上去运行，从而实现系统的负载平衡。负载迁移功能除了满足系统负载平衡的需求，对系统的容错特性也提供了很好的支持。

5 实验

5.1 实验环境

实验中 SandPOS 运行在 SandFox 模拟平台上，FileServer和 MonitorServer 运行在 X86 平台上。实验所用程序为 SPEC CPU 2000 和 NPB 2.4，测试中 SPEC CPU 2000 采用 test 数据集，NPB 采用 W 测试集。5.2 节正确性测试中给出了测试集中所有测试程序的运行结果，而 5.3 节性能测试中根据应用程序在标准 linux 上运行行为类似地取一个作为代表，共选取选 apsi、applu、twolf、mesa、wupwise、gcc、crafty 和 sixtrack 等 8 个程序为代表。

5.2 功能验证

表 1 给出了 SPEC CPU 2000 和 NPB 程序的运行结果。SPEC CPU 2000 整数程序中 perlbmk 由于 SandPOS 不支持 fork 系统调用，浮点程序中 galgel、facerec、lucas 和 fma3d 由于我们的交叉编译器不支持 Fortran-90 程序外，其它均运行通过。NPB 程序中除了 FT 需要 Fortran-90 编译器外，其它程序 W 测试集全部运行通过。上述结果表明，SandOS 可以正确执行 linux 二进制兼容的应用程序。

表 1(a) SPEC CPU 2000 整数集执行结果

应用	gzip	vpr	gcc	mcf	crafty	twolf
时间(秒)	4	1214	1072	101	2363	155
应用	parser	eon	gap	vortex	bzip2	
时间(秒)	2306	397	729	6073	4780	

表 1(b)　SPEC CPU 2000 浮点集执行结果

应用	wupwise	swim	mgrid	applu	mesa
时间(秒)	5602	274	18114	237	1665
应用	art	equake	ammp	sixtrack	apsi
时间(秒)	1022	671	2629	9316	3455

表 1(c)　NPB 执行结果(W 测试集)

应用	BT	CG	EP	IS
规模	24×24×24	7000	67108864	1048576
时间(秒)	51745	13010	46605	2990
应用	LU	MG	SP	
规模	32×32×32	64×64×64	36×36×36	
时间(秒)	134300	13801	104046	

5.3　性能测试

5.3.1　内存使用

经过功能分解和简化，SandPOS 对内存要求非常低，系统映像大小约为 440kB，因此可以考虑放在 ROM 中，从而去除传统操作系统启动时复杂的 loader 操作，加速启动进程。系统运行时除页表为 SandPOS 所占用的内存不足 2M，这个特点对于节省结算节点内存使用非常有意义。

5.3.2　调度效率

如前所述，并行系统中计算节点应该尽可能将所有资源都用来运行用户程序。我们选取 SPEC CPU 2000 作为测试程序，对在一个独占环境的 linux(2.6.11 版本)系统上的进程调度行为进行分析，运行结果如表 2 所列。

表 2　Linux 进程调度

	apsi	applu	twolf	mesa
调度总数(次)	14422	12825	9580	6908
有效调度(次)	6918	6356	4752	3398
	wupwise	gcc	crafty	sixtrack
调度总数(次)	5796	3615	2631	392
有效调度(次)	2876	1726	1253	191

表 2 第一行给出程序运行期间进程调度总次数，第二行给出被测试程序参与的调度次数。从中可以看出，虽然在一个没有任何干扰的 linux 系统下运行程序，系统中仍会有多次的进程切换，而且超过半数以上的进程切换和用户进程无关，这些切换将带来系统资源的浪费。

5.3.3　运行效率

为了客观表述操作系统的效率，我们定义运行效率为：

运行效率＝用户态运行时间/总运行时间

图 5 给出了各个测试程序在 linux 和 SandPOS 上运行的效率，对比可以看出，对于 linux 下运行效率在 99%以上的程序，SandPOS 上所占时间达到 99.999%以上。对于 linux 资源利用不太充分的程序(sixtrack，gcc)，SandPOS 下程序占用时间也在 99.99%以上，相比 linux 有 6%～9%的效率提高。

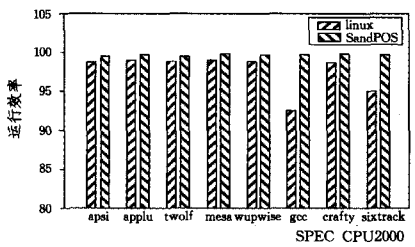

图 5　运行效率

细粒度同步系统中，只有所有节点都稳定地表现出高效率，整个系统才能有好的性能，否则会因为某个节点速度较慢而牵制其它节点，导致系统性能下降。从中可以看出，各个程序运行在 SandPOS 都表现出高效率特点，而不像在 linux 上具有较大的变化，这表明在超大规模并行系统中，SandPOS 较 linux 会有较低的系统噪音。

5.3.4　页面大小对性能的影响

SandPOS 由于对地址空间采用连续映射的方式，因此提供了大页面映射的机会，增大页面大小可以减少程序所需使用的页面个数，从而减少 TLB 失效次数。在我们的 SandFox 模拟中模拟的是 MIPS R4000 结构的计算机，共有 64 个 TLB 项，每项可以同时映射 2 个页面，因此一共可以映射 128 个 TLB 项。在 MIPS R4000 中的页面是可以配置的，因此 SandPOS 对各种大小的页面都做了支持。

表 3(a)　TLB 缺失次数

页面大小	apsi	applu	twolf	mesa
4kB	183358	160459	329	32561
64kB	1592	17954	12	753
1MB	106	36	6	13
	wupwise	gcc	crafty	sixtrack
4kB	593342	1708479	22892907	54630
64kB	25587	109	36	1428
1MB	502	7	5	29

表 3(b)　内存印迹大小

应用	apsi	applu	twolf	mesa
Footprint(MB)	191.4	3.9	0.7	9.23
应用	wupwise	gcc	crafty	sixtrack
Footprint(MB)	176.2	5.6	2.36	19.9

对选取的应用分别设置页面大小为 4k，64k 和 1M 时，程序运行中 TLB 缺失次数如表 3(a)所列。从中可以看出，随着页面大小的增大，TLB 缺失次数较大程度地降低。对于 gcc 和 crafty，页面大小从 4k 增加到 64k 时 TLB 缺失次数减少尤其明显，主要原因在于当页面大小为 4k 时，程序运行过程中所访问的页面数超过了 TLB 容量，因此需要不断进行 TLB 的替换操作，造成 TLB 缺失次数的急剧增加。而 TLB 大小改为 64k 和 1M 之后，上述现象就不再发生。当页面大小为 1M 时，applu、twolf、mesa、gcc、crafty 和 sixtrack 的 TLB 缺失次数都少于 TLB 项数目，因此所有缺失均为首次访问缺失，而 apsi 和 wupwise 则出现了 TLB 替换现象。为了对这一情况进行验证，表 3(b)给出了程序执行过程中内存印迹的大小。当页面大小为 1M 时，64 项 TLB 可以映射 128M 内存(每个 TLB 项映射两个页面)，表 4(b)中结果表明，apsi 和 wupwise 由于内存印迹超过了 TLB 映射容量，所以发生了 TLB 替换操作。

从中可以看出，SandPOS 不但对大页面做了支持，而且地址空间连续映射的方式使得大页面映射更为有效，增大 TLB 页面大小可以有效降低 TLB 失效次数，从而提高操作系统效率。

结束语　随着超级计算机规模的扩大，超大规模并行系统中所有节点都使用传统的 UNIX/linux 存在着弊端，尤其是 linux 对于计算节点存在着功能冗余、效率较低的缺点。因此，本文通过对超大规模系统操作系统需求进行分析，基于 SandFox 模拟器设计和实现了超大规模并行系统操作系统原型 SandOS。通过功能分解和功能简化，基于模拟器实现了计算节点轻核心操作系统原型 SandPOS，并在 linux 环境下对 I/O 节点操作系统 FileServer 和服务节点操作系统 MonitorServer 进行技术验证。对于计算节点轻核心 SandPOS，基于单用户进程的前提实现了用户进程的直接创建、用户进程

(下转第 40 页)

仅需接收锚节点信息即可实现定位，算法简单。仿真结果表明，本文算法具有较高的定位精度和稳定性。因此，本文算法是一种低成本、低复杂度、高精度的 WSN 自定位技术。

参 考 文 献

[1] 崔逊学，左从菊. 无线传感器网络简明教程[M]. 北京：清华大学出版社，2009：69

[2] 李牧东，熊伟，郭龙. 基于人工蜂群算法的 DV-Hop 定位改进[J]. 计算机科学，2013，40(1)：33-36

[3] 欧阳丹彤，何金胜，白洪涛. 一种约束粒子群优化的无线传感器网络节点定位算法[J]. 计算机科学，2011，38(7)：46-50

[4] Chiu Wei-yu, Chen Bor-Sen, Yang Chang-Yi. Robust Relative Location Estimation in Wireless Sensor Networks with Inexact Position Problems [J]. Mobile Computing, IEEE, 2012, 11(6): 935-946

[5] Zhang Rong-biao, Zhang Li-li, Feng You-bing. Very Low Energy Consumption Wireless Sensor Localization for Danger Environments with Single Mobile Anchor Node [J]. Wireless Personal Communications, 2008, 47(4): 497-521

[6] 刘辉，李智. 基于移动锚节点的无线传感器网络圆心定位[J]. 计算机与数字工程，2012(3)：7-9

[7] Jiang Jin-fang, Han Guang-jie, Xu Hui-hui, et al. LMAT: Localization with a Mobile Anchor Node Based on Trilateration in Wireless Sensor Networks [C] // Global Telecommunications Conference. Dec 2011: 1-6

[8] 刘辉亚，徐建波. 无线传感器网络节点定位的移动信标节点路径规划[J]. 传感技术学报，2010，23(6)：873-877

[9] 陈子琦. 基于移动锚节点的 WSN 节点定位研究[D]. 长沙：长沙理工大学，2012：5-79

[10] 鲍可进，王伟. 一种移动单锚节点的无线传感器网络定位算法[J]. 计算机应用研究，2010，27(4)：1452-1454

[11] 董振中，王恩博. 无线传感器网络中一种基于定向天线的节点定位算法[J]. 电子技术，2010，37(10)：7-9

[12] Ou Chia-ho. A Localization Scheme for Wireless Sensor Networks Using Mobile Anchors with Directional Antennas [J]. Sensors Journal, IEEE, 2011, 11(7): 1607-1616

[13] 李海涛，李燕，张建忠. 微波定向天线对准实现方法[J]. 无线电工程，2011，41(3)：44-46

(上接第 36 页)

的一次性加载并且连续映射的功能，在此基础上实现了大页面支持，减少了程序运行过程中 TLB 缺失次数。通过必要时调度策略减少了应用程序在计算节点上运行时非必要的调度，提高了计算节点运算效率，并减少了超大规模情况下的系统噪音。SandPOS 和 FileServer、MonitorServer 之间采用消息传递的通信方式，这种松耦合的结构可以带来较好的系统扩展性。为了提供不同节点间的高效通信，SandOS 中使用了简洁的通信协议，并通过将网卡映射在用户态的方法减少了通信操作时进出内核的开销。试验结果表明，SandPOS 比标准 Linux 在内存开销、调度效率、运行效率、页表管理等方面都具有更高的效率。

因不支持按需调页，故对于用户程序内存超过物理内存的情况需通过系统中增加内存服务器来作为内存和磁盘的中间层来解决，如何有效地使用内存服务器是下一步 SandOS 的研究方向。SandOS 目前的实现中还存在不足的地方，如 SandPOS 仅支持 MIPS 体系结构，FileServer 和 MonitorServer 对 linux 环境的依托等，这些将在后期开发中逐渐完善。

参 考 文 献

[1] Moreira J, Brutman M, et al. Designing a highly-scalable operating system: the blue gene/L story[C]// Proceedings of the 2006 ACM/IEEE conference on Supercomputing. Tampa, Florida, November 2006

[2] Yoshii K, Iskra K. Performance and Scalability Evaluation of 'Big Memory' on Blue Gene Linux[J]. International Journal of High Performance Computing Applications, May 2010: 148-160

[3] Lange J, Pedretti K. Palacios and Kitten: New High Performance Operating Systems For Scalable Virtualized and Native Supercomputing[C]// 2010 IEEE International Symposium on Parallel & Distributed Processing (IPDPS). Atlanta, USA, April 2010

[4] Yoshii K, Iskra K. Characterizing the performance of "big memory" on blue gene linux[C]// Proceedings of the 2009 International Conference on Parallel Processing Workshops. Vienna, Austria, September 2009

[5] Wheat S R, et al. PUMA: an operating system for massively parallel systems[C]// Proceedings of the 27th Hawaii International Conference on System Sciences. Hawaii, USA, January 1994

[6] Brightwell R, Maccabe A B. On the appropriateness of commodity operating systems for large-scale, balanced computing systems[C]// Proceedings of the 17th International Symposium on Parallel and Distributed Processing. Nice, France, April 2003

[7] 黄凯，徐志伟. 可扩展并行计算-技术、结构与编程[M]. 北京：机械工业出版社，2001：37-39

[8] 包云岗，许建卫，陈明宇，等. 一种新型计算机体系结构模拟器的研究与实现[J]. 系统仿真学报，2007

[9] Zajcew R, Roy P. An OSF/1 UNIX for Massively Parallel Multicomputers [C] // Proceedings of the Winter USENIX Conference. San Diego, USA, January 1993

[10] Adiga N R, et al. An overview of the BlueGene/L supercomputer[C]// Proceedings of the 2002 ACM/IEEE conference on Supercomputing. Baltimore, Maryland, November 2002

[11] CrayXT3Datasheet[N]. http://www.craysupercomputers.com/downloads/CrayXT3/CrayXT3_Datasheet.pdf

[12] Linux Networx Cluster System Climbs to Third Fastest Supercomputer in the World [N]. http://www.prnewswire.com/news-releases/linux-networx-cluster-system-climbs-to-third-fastest-supercomputer-in-the-world-71374077.html

[13] Mattson T G, Henry G. An overview of the Intel TFLOPS supercomputer [J]. Intel Technology Journal, 1998, 1

[14] ASCI White[N]. http://www.thocp.net/hardware/asci_white.htm, 2003

[15] Ron B, Ann F L, et al. Massively parallel computing using commodity components[M]. Elsevier Science Publishers B. V. 26: 243-266

[16] Steve N. Scaling Linux to new heights: the SGI Altix 3000 system [J]. Specialized Systems Consultants, 2003, 3

[17] The Small Linux for Big Computers[N]. http://www.mcs.anl.gov/ research/ projects/zeptoos/links, 2011

[18] 沈林峰，陈明宇，许建卫. 兼容 Linux 应用环境的多粒度全系统模拟平台-SandUPSim [J]. 计算机工程，2005(22)

[19] TOP500 Supercomputer Sites[N]. http://www.top500.org/, 2013