1000-9825/2001/12(10)1562-07 ©2001 Journal of Software 软件学报 Vol.12, No.10

基于多虚空间多重映射技术的并行操作系统*

陈左宁, 金怡濂

(国家并行计算机工程技术研究中心,北京 100080)

E-mail: czn@jnc.wx.net.cn

摘要: 高性能计算机系统的可扩展性是系统设计的一大难题,NUMA(non_uniform memory architecture)结构正是为了解决共享存储体系的可扩展性问题而提出来的.研究和实践表明,整机系统的可扩展性与操作系统的结构有着密切的关系.典型的多处理机操作系统通常采用两种结构,基于共享的单一核心结构以及基于消息的多核心结构.通过分析得出结论认为,这两种结构都不能很好地适应可扩展并行机尤其是NUMA结构并行机的需求.针对存在的问题,提出了新的结构设计思想:多虚空间多重映射与主动消息相结合.测试和运行结果显示,该结构成功地解决了系统的可扩展问题.

关键词: 操作系统;可扩展性;可编程性;NUMA(non_uniform memory architecture)体系结构;大规模并行处理

中图法分类号: TP316 **文献标识码**: A

在NUMA(non_uniform memory architecture)体系结构发展的过程中,我们注意到这样一个事实:体系结构、互连以及工程实现方面的技术进步往往超前于相关的系统软件的进步.其中的一个主要原因是NUMA结构的操作系统不成熟.主要表现在结构各异;可裁减性或模块化程度差;未形成主流标准;可移植性、平台独立性和健壮性较之SMP(symmetric multi_processor)的操作系统有较大的差距.就NUMA结构而言,一方面,操作系统为机器性能的发挥承担了比SMP机器的操作系统更大的责任,操作系统自身的性能变成了整机系统达到无缝可扩展的一个关键因素.另一方面,提供单一系统映像也是NUMA机器的一个主要设计目标.设计可扩展的并具有单一系统映像的操作系统,会遇到相当多的矛盾.

目前,在解决操作系统的可扩展问题上,常见的方法或是采用多核心的系统辅以一些分布式的算法,或是采用单一全共享核心,同时将操作系统核心数据结构作适当的物理上的分布.前一种方法增加了实现单一系统映像的难度,后一种方法很难扩展到更大的规模.本文试图分析这些问题并提出有效的解决途径.针对一个NUMA结构的MPP(massively parallel processing)系统,我们提出并实现了称作多虚空间多重映射与主动消息相结合的操作系统结构设计方法,从测试结果和系统实际投入运行的效果来看,该方法成功地解决了操作系统的可扩展问题,进而支持整机系统向更大规模扩展.本文将介绍这种设计思想及实现方法.

1 NUMA结构操作系统的可扩展问题

NUMA结构所要解决的最大问题是单一映像下的系统可扩展性,这也正是造成NUMA与SMP体系结构的操作系统在结构和算法上需要采取不同对策的原因.

* 收稿日期: 2000-03-02; 修改日期: 2000-06-12

作者简介: 陈左宁(1957-),女,北京人,研究员,主要研究领域为计算机系统软件和体系结构;金怡濂(1929-),男,天津人,研究员,博士生导师,中国工程院院士,主要研究领域为计算机体系结构,高速线路工程.

1.1 分层可扩展性

可扩展性问题可分为几个层次(如图1所示):硬件结构的可扩展性,操作系统的可扩展性,并行语言、并行库、运行时系统(run-time system)的可扩展性,应用程序的可扩展性.在这4个层次中,每两个层次之间存在一个可扩展层面.每层基于其支撑层面,其可扩展性主要取决于为解决自身面向的问题所需要的算法设计、逻辑设计和结构设计.每个层面的可扩展性依赖于该层面以下所有层同时的可扩展.最终,全系统的可扩展性要求这4个层次必须都是可扩展的,即平衡的扩展[5].

Scalable APP①
Scalable language, Lib, Run-Time system②
Scalable OS③
Scalable hardware④

①可扩展的应用程序,②可扩展的并行语言、并行库、运行时系统,③可扩展的操作系统,④可扩展的硬件结构.

Fig. 1 Hierarchical relationship of scalable system

图1 可扩展系统的层次关系

近年来,围绕着系统可扩展问题,硬软件各层次都展开了开拓性的研究,取得了相应的成果.首先,由于GB级带宽网络的出现(如GSN,10Gbit/s Ethernet,4Gbit/s Fiber Channel),支持高速通信的标准网络协议的制定(如VIA:virtual interface architecture协议支持用户级直接通信)以及分布式共享主存一致性处理技术(如基于目录的一致性算法)的逐渐成熟,使得影响系统可扩展性的主要环节——处理机间通信的实际带宽和延迟有了很大的改进.在语言方面,OpenMP将数据分布的描述引入共享型并行模式,增强了共享型并行语言的可扩展性.MPI-2加入了单边通信及并行I/O功能,有利于提高大规模并行的效率.

并行机操作系统作为一个高度并发的系统软件,其可扩展性可从两个方面来理解.首先,它对外提供服务的能力应是可扩展的.比如当共享的内存空间、外部网络接口以及磁盘系统随CPU个数的增大而按一定比例增多时,操作系统应能提供所谓"无缝"增强的共享主存、联网以及文件系统服务.可扩展性的另一方面反映在操作系统内部结构与算法上,这主要是由操作系统自身的并发性所决定的.操作系统是一个十分复杂的并行系统,它的无规律的数据空间布局、非规则化的数据访问模式、大量的异步事件以及需要面对NP完全类问题等特点,使得设计一个高效、可扩展的并行操作系统变得非常困难.

1.2 操作系统结构与可扩展的关系

操作系统的可扩展性与操作系统的结构有着密切的关系.在过去的十几年里,多处理机操作系统基本上采用的或者是基于共享机制的单一(monolithic)核心结构,或者是基于消息传递机制的多核心结构[2].

单一核心操作系统的主要代表是UNIX类系统,目前,高端计算机大多数使用这类系统.这类系统采用的是进程分层结构以及共享的操作系统空间.从程序设计方法学的角度来看,集中式操作系统的功能可按半序结构划分,进程分层结构正好对应于纵向分层的结构.操作系统共享空间布局可与物理的共享存储体系结构相对应.所以,这类系统适合于SMP结构的并行机.

消息型多核心的操作系统从结构上分解了单一核心操作系统,这类操作系统的最大优点是容易实现模块化设计,进而使系统物理上容易扩展,以适应纯分布式的体系结构.

理论上讲,NUMA结构的系统既可以用单一共享核心的操作系统,也可以用多核心消息型操

作系统.首先,NUMA结构提供了与传统的SMP结构近乎相同的存储语义,无须作太多改动即可直接运行单一共享核心的操作系统,不少中小规模的并行系统正是这样做的.但是,大量课题实验表明[3~5],分布式共享结构用集中式的软件来管理,不仅限制了硬件效率的发挥,而且也不利于系统的扩展.这是因为,当规模扩大到一定程度时,共享内核容易产生软件热点.NUMA结构既有分布的属性又有共享的属性,而可扩展性恰恰来自于前一属性,因而要求操作系统从结构、算法上都要与之相适应.

为了解决共享引起的软件热点问题,操作系统采用基于消息机制的分布式结构是最直接的方法,但实际情况并非这样简单.这里,最大的问题是消息机制本身限制了系统性能的发挥.多机操作系统所承担的任务决定了多处理机间有大量离散的控制信件,与批量数据移动相比,由少量数据构成的控制信件对消息的传送延迟更加敏感,其影响也更大.因此,这类操作系统通常只适应于中粗粒度的并行模型,而对于强相关的并行应用模型,尤其是当其基于分布式共享结构时,往往会由于系统开销过大而不能取得满意的效果.

1.3 可扩展性与单一系统映像(single system image)的关系

单一系统映像(SSI)可以在不同层次上实现:中间件(middleware)、基础软件(underware)和硬件[6].在某一层上实现了SSI,可以简化高层的开发运行环境.显然,除了完全紧耦合的硬件平台,可扩展的硬件结构是不可能提供SSI的.因而可扩展并行机的SSI均在软件层次上实现.

操作系统的单一映像有两个含义:一是操作系统映射到一维的运行空间,即操作系统自身在一个统一的空间上设计;另一个是操作系统提供用户单一映像的API(application programming interface).

基于单一共享核心的对称式操作系统利用共享数据结构实现并行资源的管理调度.这种结构的最大优点是在描述系统内各类随机并行事件时自然、方便,因而易于实现单一系统映像的目标.

基于多核心的消息型分布式操作系统虽然适应了系统物理上的可扩展性,但同时也带来了难以实现单一系统映像的问题.如同分布式共享的硬件结构要解决可扩展性与共享存储这一对矛盾一样,操作系统设计需要解决自身可扩展性与实现SSI的矛盾.

2 多虚空间多重映射

多虚空间多重映射与主动消息相结合是我们为一台NUMA结构的大规模并行机设计操作系统时提出并实现的技术,它的目标是在实现系统单一映像的同时,使操作系统具有良好的可扩展性.

我们设计的操作系统由微内核MKM(microkernel based mach)及各种服务器组成,采用了基于微核的两级并行化结构.第1级并行在3类执行不同功能的节点——运算节点、I/O节点和前端节点间实现.第2级分布在由运算节点构成的作业分区内,采用了多虚空间多重映射和主动消息机制,形成了并行程序的运行环境.最终形成了一个物理上功能分担、具有单一操作和运算环境、结构上可扩展(规模由几个节点到几百个节点)的分布式并行操作系统.

2.1 定 义

图2表明了多虚空间多重映射的思想.首先,我们将操作系统虚空间V划分为全局与局部两个域G和L,G和L分别包含各自的子域:

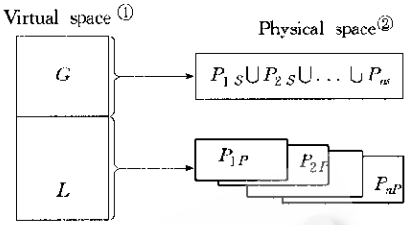
$$G=\{G_1,G_2,\ldots,G_m\};\qquad L=\{L_1,L_2,\ldots,L_m\}. \tag{1}$$

G,L 与 V 的对应关系是惟一的,且 G_i 和 L_i 交叉分布到 V 上.m 为子域总数.

"多虚空间"是指每个 PE(processing element)的微核心 MKM 具有独立的虚空间 Vk($k=1,\ldots,n$,n 为总的 PE 数),形成独立的操作系统.这里,每个 Vk 与 G,L 的对应关系是相同的.表示为

$$V \Leftrightarrow V_1 \Leftrightarrow V_2 \Leftrightarrow \ldots V_n \Leftrightarrow G \bigcup L, \tag{2}$$

其中"⇔"表示各 PE 的操作系统具有相同的虚空间.



①虚空间,②物理空间.

Fig. 2 Logical diagram of multi-virtual-space and multi-mapping

图2 多虚空间多重映射射逻辑图

"多重映射"是指构成每个 Vk 的 G,L 映射到的物理空间 Pk($k=1,\ldots,n$,n 为总的 PE 数)是不同的,既有 PE 私有空间 Pkp,又有分布式共享空间 Pks.表示为

$$G \rightarrow P1s \bigcup P2s \ldots \bigcup Pns, \tag{3}$$

$$\left.\begin{array}{l} L \rightarrow P1p \\ L \rightarrow P2p \\ \vdots \\ L \rightarrow Pnp \end{array}\right\}. \tag{4}$$

其中"→"表示虚实空间映射关系.式(3)表示全局域分布映射到各 PE 的共享空间.式(4)表示局部域完整地映射到各 PE 的私有空间.

使用该技术构成的操作系统具有以下特征:

• 整个系统具有连续统一编址的核心空间,从这点来看,所有 PE 的 MKM 好象共享单一核心 V.

• 每个 PE 的 MKM 具有独立的核心虚空间映射,系统中存在与 PE 个数相对应的多个核心虚空间 Vk,从这点来看,又与消息型操作系统相似.

这里,我们利用局部域将操作系统的管理和服务功能局部化,提高了整个系统的并行处理能力.不同 PE 局部域的联系通过消息机制,利用全局域有效地处理系统内强相关事件,同时支持系统更方便地实现单一映像.

2.2 多虚空间多重映射的实现

实现多虚空间多重映射首先要完成对资源对象的描述.依据确定的 MKM 的功能,将系统管理的资源分为静态对象、动态对象两种.然后依据各对象的特性,将其数据结构及功能操作进行划分和分布.这些特性包括并发处理特性(分离并行处理与串行处理)、局部特性(分离全局域与局部域)以及策略与机制特性(分离微核与服务器).这里,我们主要关心的是前两个特性.

系统采用动态划分分区(partition)的资源分配策略.静态对象是指当分区形成时所确定的资源,如分布式存储空间、活动 PE 族等.动态对象是指在并行程序运行过程中动态派生或消亡的资源,如并行任务族、各类虚拟存储对象(包括虚存空间、文件空间等)、通信端口等.

各类对象的并发特性可分为完全并行、弱相关并行和强相关并行 3 种.全局域与局部域的划分与对象的并发特性关系密切.完全并发的对象安排在局部域;强相关并行的对象一般安排在全局域.弱相关并行的对象既可以安排在局部域(采用主动消息进行相关性操作),也可以分布映射到不同 PE 空间(避免"假共享")的全局域(采用局部或全局共享锁进行相关操作).

比如,对分布式存储空间以及并行任务私有空间的管理,系统设计了完全并行的管理算法,各PE独立完成空间的分配、调度和映射到I/O节点的对换空间管理.为此,系统在每个PE的MKM空间上设置了远程空间服务器,PE物理空间、并行任务私有空间的数据结构安排在局部域,相关的操作在局部域上进行.其他PE对这些空间的服务请求通过远程空间服务器实现.

又如,对活动PE的管理采用了弱相关的方式.为此,首先将描述活动PE的数据结构在全局空间进行分布,同时在局部域设立局部调度队列,然后设计分布与集中相结合的管理算法.在创建并行任务时,统一分配调度处理器.在并行程序运行时,各PE自行调度管理本PE的运行.

再如,对并行任务的控制,分布式共享虚拟存储空间的管理属于强相关范围,为此,在全局域设置了统一的进程、任务和线程名字空间,每个PE设有远程进程服务器,与其他PE无关的事件在本地完成,涉及到全局的事件或是通过核心共享数据临界区或是通过主动消息激活远程进程服务器来协作完成.

2.3 多虚空间多重映射的优点

采用多虚空间多重映射带来的好处是:

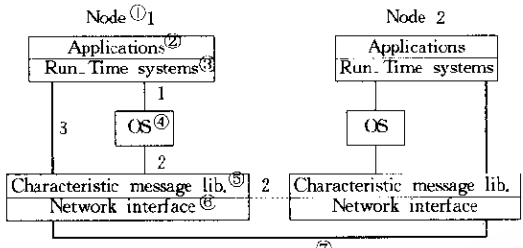
- 提高了系统的并行处理能力.PE本地功能由于相关数据结构局部化而彻底并行化.
- 减少了系统开销,提高了系统效率.分区中各PE协作处理全局事件,视事件本身的粒度,选择最优的协作方式.如状态变化可通过共享数据传递,而像Fork这类复制操作或者复杂的服务请求可使用消息.
- 减轻了网上压力,避开了网上热点.
- 统一编址的MKM空间,方便了在多机环境下操作系统高层功能的开发.
- 多虚空间的设置使得每个PE的MKM虚空间总量压缩,有利于大型课题.
- 避免了共享单一核心的"假共享"问题.
- 有利于故障隔离,改善了系统的健壮性.

3 基于主动消息和zero_copy的通信机制

解决系统可扩展性的另一种手段是提高系统实际通信速度.我们设计了一种由网络接口固件和MKM共同实现的主动消息机制,用以解决多PE的MKM高效协作的问题.主动消息[7]是一种高效灵活的异步通信机制.它的核心是避开常规通信模式中的多层次协议转换和数据拷贝,利用消息本身直接激活相应的处理程序,对消息所带来的数据直接进行处理.

我们利用系统中断信件通信协议实现主动消息.该协议分为两层,底层由网络接口实现,主要功能是对网上中断信件进行流量控制以及信件上网的发送与接收.高层由MKM实现,主要功能是实现统一的中断信息管理接口以及主动消息的调度队列和激活机制.在发送方,采用延迟缓冲管理技术实现网络拥塞时的流控,理论上发送中断信件个数不限.在接收方,网络接口通过中断激活PE,PE按信件头指示进行信件的调度,直接激活消息处理者.从网络接口传过来的消息,一次定位到目的空间,无须再拷贝.

为了支持不同PE上不同任务空间的直接消息传送,系统设计了一套支撑机制(如图3所示).通过在网络接口设置一张由操作系统管理的任务-空间映射表,实现了以任务为对象的傍路操作系统的零拷贝(zero copy)通信模式,从而大大提高了通信效率.MPI/PVM等高层消息库都是基于这种机制而实现的.



①节点,②应用,③运行时库,④操作系统,⑤特征消息库,⑥网络接口,⑦任务建立/终止,⑧建立/撤销任务/空间映射,⑨节点间消息传送.

Fig. 3 Message passing between nodes
图3 节点间消息传送

4 测试结果分析

为了测试系统的可扩展性,我们选择了对可扩展性敏感的操作系统的基本功能进行测试,包括进程控制管理和存储管理典型操作的执行时间.测试规模由 4 个 PE 到 256 个 PE.测试基于 SPMD(single program over multiple data streams)与 MPMD(multiple programs over multiple data streams)两种并行模型.表 1 是测试结果.

在表 1 中,A 是在多 PE 上创建一批进程,它们执行同一程序且继承了父进程的所有属性(FORK).这里,系统最大的开销是在多 PE 上同时进行动态空间复制.B 也是在多 PE 上创建一批进程,这些进程执行与父进程不同的程序文件(FORK+EXEC).主要的系统动作是多 PE 同时安装同一个可执行文件.C 是父进程杀死所有子进程并等待它们的中止(WAIT+EXIT).主要开销是从多 PE 上回收进程状态.D 是父进程向所有子进程发信号(SIGNAL),子进程接收信号(调用信号处理函数通知父进程).E 是共享空间的动态申请(MMAP)、分布(SETPATT)和初始化.设定申请的总的共享空间为 30MB * PE 数.

Table 1 Test results of representative OS functions (μs)
表 1 操作系统典型功能测试结果 (μs)

Number①	Tested functions②	4PE	8PE	16PE	32PE	64PE	128PE	256PE
A	FORK	169	219	250	344	370	840	1428
B	FORK+EXEC	139	146	160	200	370	463	872
C	WAIT+EXIT		167	166	167	500	517	
D	SIGNAL+call FUN		8	16	57	127	230	
E	MMAP+SETPATT	2 065	2 174	2 431	2 760	3 675	3 726	6 478

①编号,②测试功能.

表 1 表明,系统开销未随规模的扩大成比例地增加.这主要是由于批量进程控制与空间管理分布到各 PE 完成,多 PE 间离散的同步控制事件采用共享的数据结构,而成批的空间复制采用消息.C 和 D 由于存在突发性的多对一的通信(主进程从其他 PE 上的子进程回收状态),系统开销的增加比例要大一些.在系统设计过程中,我们对比了全部用共享临界区方式或只用消息方式实现这些操作系统功能.结果全部用共享临界区实现时,当系统规模超过 32 个 PE,处理时间发生突变,当超过 64 个 PE,网上出现严重拥塞.而当仅用消息方式实现时,与本文所介绍的方案相比,同等系统规模下的处理时间呈几个数量级的增加.对比结果表明,新的操作系统结构的设计思想很好地实现了系统的可扩展性.

5 总 结

高性能计算机的设计给操作系统提出了挑战,实现SSI和可扩展性是其中的关键问题.针对现有的多处理机操作系统所存在的问题,我们在一台NUMA结构的MPP机操作系统的设计中,提出并实现了新的结构设计思想.经过上百道并行课题的实际使用,验证了该系统的高效性和可扩展性.国外有关NUMA结构操作系统的研究工作,以SGI Origin2000的Cellular IRIX为主要代表,目前仍在进展中.

References:

[1] Erik, Hager, Greg, Papadopoulos. Parallel computing in the commercial marketplace: research and innovation at work. Proceedings of the IEEE, 1999,87(3):405~411.

[2] Nukherjee, B. A survey of multiprocessor operating system kernels. Technical Report, GITCC-92/05, Georgia Institute of Technology, 1993.

[3] Chapin, J., Mendel, Rosenblum, Scott, Devine, *et al*. Hive: fault containment for shared-memory multiprocessors. In: Proceedings of the 15th ACM Symposium on Operating Systems Principles. New York: ACM Press, 1995. http://sdg.lcs.mit.edu/~jchapin/ publications.html.

[4] Kaushik, Ghosh, Christie, A. J. Communication across fault-containment firewalls on the SGI origin. In: Proceedings of the International Symposium on High-Performance Computer Architecture. Los Alamitos, CA: IEEE Computer Society Press, 1998.

[5] Ben, Verghese, Scott, Devine, Anoop, Gupta, *et al*. Operating system support for improving data locality on CC-NUMA compute servers. In: Berman, A. M., ed. Proceedings of the 7th International Conference on Architectural Support for Programming Languages and Operating Systems. New York: ACM Press, 1996. 279~289.

[6] Bruce, Walker, Douglas, Steel. Implementing a full single image Unixware cluster: Middleware vs Underware. In: Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications. Los Alamitos, CA: IEEE Computer Society Press, 1999. http://www.dgs.monsh.edu.au/~rajkumar/pdpta99.

[7] Von Eicken, T., Culler, D. E., Goldstein, S. C., *et al*. Active message: a mechanism for integrated communication and computation. In: Proceedings of the 19th Annual International Symposium on Computer Architecture. Los Alamitos, CA: IEEE Computer Society Press, 1992.

A Parallel Operating System Based on Multi-Virtual-Space and Multi-Mapping Technology*

CHEN Zuo-ning, JIN Yi-lian

(*National Research Center of Parallel Computer Engineering and Technology*, *Beijing* 100080, *China*)

E-mail: czn@jnc.wx.net.cn

Abstract: The scalability of system is a puzzle to design high performance computers. NUMA (non-uniform memory architecture) architecture is proposed for scalable system underlying a shared memory pattern. Research and practice results show that the scalability of system tightly correlates with that of the operating system. Usually, two structures of operating system are employed in multiprocessor systems, monolithic based on shared memory and multi-kernel based on message passing. However, both of them cannot fully fit scalable parallel computers, especially MPP systems with NUMA architecture. In this paper, the problems in two structures are analyzed, a new structure, multi-virtual-space and multi-mapping with active message, is proposed. The test and practice results show that the new structure is successful in the scalability of system. This paper introduces detailedly the implementation of the structure in a MPP system with NUMA architecture.

Key words: operating system; scalability; programmability; NUMA (non-uniform memory architecture) architecture; massively parallel processing

* Received March 2, 2000; accepted June 12, 2000