DOI: 10.13700/j.bh.1001-5965.2001.01.023
2001年 2月 北京航空航天大学学报 February 2001
第27卷 第1期 Journal of Beijing University of Aeronautics and Astronautics Vol.27 No.1

NET-RAM的设计与实现

龙 翔 聊鸿斌 高小鹏
(北京航空航天大学 计算机科学与工程系)

摘 要: 对SNOW机群系统的分布式共享存储管理机制进行适当的简化,就可实现网络虚拟存储环境.它以本地节点的主存作为cache,以网络上其它节点机的主存作为本地节点机主存的后援存储,取代本地节点机的硬盘或其它介质的存储设备.充分利用网络上节点的物理存储空间,使一些在单机上由于内存限制而无法完成的计算成为可能.这对于时间敏感性不高但需使用大内存的计算程序尤其适合.本文在介绍了网络虚存的概念之后,详细阐述了一个网络虚拟环境NET-RAM总体结构,软cache的实现方法,并对该网络虚存环境进行了测试,分析了测试结果.最后对今后的工作进行了探讨.

关 键 词: 网络; 虚拟存储; 分布型网络; 分布式共享存储; 软cache
中图分类号: TP 333.96
文献标识码: A **文 章 编 号**: 1001-5965(2001)01-0083-05

并行计算机系统已成为处理当前国民经济、国防建设和高科技发展中许多广泛而又影响深刻的重大问题,如航空航天器的设计和地面仿真,物探地震数据的分析和处理,大型数据库的查询和维护, 资源及气象数据的处理等的重要工具.NOW (Network of Workstations)系统是以商用工作站或高档微机作为处理节点,通过高速商用局域网或专用互连网络互联而构成的一种并行计算机系统,一般称之为并行机群系统.它具有可扩展性好、性能价格比高、用户投资风险小、配置灵活及软件资源丰富等优点,日益成为一种重要的科学计算平台.由国家863计划资助的、由北京航空航天大学承担研制的并行机群系统SNOW(Shared memory based NOW)是一套可同时支持共享存储和消息传递模式的、以高性能PC机为节点的、具有高带宽低延迟互连网络的规模可扩展的多计算机系统.

在SNOW系统的研制过程中发现,对系统的共享内存环境进行若干简化,如去除一致性维护等工作,就可实现网络虚存(Network Virtual Memory).网络虚存的基本概念就是以本地节点的主存作为cache,以网络上其它节点机的主存作为本地节点机主存的后援存储,取代本地节点机的硬盘或其它介质的存储设备.由于现今网络设备的I/O带宽(Bandwidth)和延迟(Latency)已显著高于磁盘,使用网络虚存将使虚存的整体性能大大提高(理想情况下接近总线上的存储器).同时,充分利用网络上节点的资源可以获得一个很大的物理存储空间,使一些在单机上由于内存限制而无法完成的计算成为可能.这对于时间敏感性不高但需使用大内存的计算程序尤其适合.理论上只要网络系统上连接的节点机足够多,所有数据均可放置系统的物理内存中.

国外对网络虚存也进行了有关研究,如UC Berkerley的NOW[1]; DEC Systems Research Center的GMS(Global Memory Service)[2]等.这些系统所采取的方法是更改交换区设备(Swap Device)和修改操作系统核心(OS Kernel).这些方法都可以透明地将用户进程的代码或数据分布到网络上其它节点上,给用户一个大容量的、高性能低成本的存储器的概念.

与这些方法不同,SNOW系统采用用户级虚存管理(User-Level Virtual Memory Management)来实现网络虚存.SNOW系统通过使用UNIX系统提供的虚存映射机制(mmap函数)实现了一个用户级的软件cache.该cache的后援存储设备是网络上的其它节点,通过标准的网络接口完成数据的装入和替换,透明地分布数据于网络上的各个节

收稿日期: 1999-05-04
基金项目: 国家863高技术计划资助项目(863-306-ZD-03)
作者简介: 龙 翔(1963-),男,广西柳州人,副教授,100083,北京.
(C)1994-2021 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

点机.

SNOW 系统采用用户级虚存管理的原因是给用户对软件 cache 最大的可控性(系统同时提供一套通用的接口以方便用户使用),根据程序的行为来调整软件 cache 的行为,如预取、调整 cache 行尺寸等,以发挥系统的最大性能.以下本文将分别介绍以网络虚存为基础的 NET-RAM 系统的设计、实现、性能测试及分析.

1 NET-RAM 系统设计

1.1 软件 cache

用户进程通过联入网络虚存函数库(NET-RAM Lib)来使用网络虚存.NET-RAM Lib 通过用户级虚存管理来分配虚存空间和维护软件 cache,截获进程对虚存空间的访问并动态地载入和替换软件 cache 行,建立虚存空间到软件 cache 之间的映射,使用户进程透明地使用网络虚存.软件 cache 的结构如图 1 所示.

针对 cache 存储层次的 4 个基本问题[3]提出了自己的软件 cache 解决方案:软件 cache 采用全互联(Full Associative)方式分配 cache 行;软件 cache 行的标识是由 CPU 的 MMU 部件完成的.在 cache 中的软件 cache 行均由页表完成虚空间到软件 cache 的映射,不在 cache 中的软件 cache 行在被访问时由于没有映射而产生失效;由于替换算法可以是多样的,本系统可以由用户指定软件 cache 的替换算法.

目前由于操作系统的限制无法简单地高效地实现 LRU 算法,采用 Random 替换算法替代.在失效率方面,LRU 和 Random 算法在 cache 尺寸较大时性能接近;软件 cache 采用写回策略(Write back strategy).

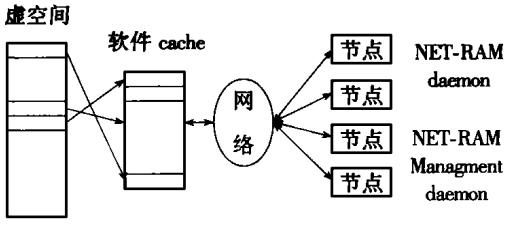


图 1 软件 cache 的结构方案

1.2 系统结构

NET-RAM 由存储节点和存储管理节点组成,存储节点上启动一个 NET-RAM daemon 响应软件 cache 对 cache 行的装载和替换请求,提供后援存储.存储管理节点运行 NET-RAM Management daemon 对软件 cache 的存储分配释放请求提供服务,完成存储分配的管理.

节点可以同时是存储节点和存储管理节点,也可以是单一功能的节点.存储管理节点可以由一个或多个节点组成,多个节点时由多个节点协同完成网络虚存的分配.

1.3 负载平衡

网络中的每个存储节点都可以被选中作为软件 cache 在替换时的回写存储后援.但当某个存储节点的负载很大时这样做会使 cache 行的装载响应变慢,影响系统的整体性能.NET-RAM daemon 周期地向软件 cache 发送自己的负载情况.软件 cache 则根据各个存储节点的负载情况决定在 cache 行回写时写回哪个节点,将负载均匀地分布到整个系统上.

1.4 可靠性

分布式系统中某个节点失效的可能性远大于单机的情况,提高可靠性自然是用户的需求之一.NET-RAM 采用与 RAID 技术中相似的机制来提高可靠性.RAID 技术中有许多级别,表 1 给出了 RAID 的级别和如何利用 8 个磁盘通过冗余提供高可靠性.

表 1 不同 RAID 级别对应的可靠性

RAID 级别		可容忍的失效数	数据磁盘数	校验磁盘数
0	无冗余	0	8	0
1	映像	1	8	8
2	ECC 校验	1	8	4
3	位间隔奇偶校验	1	8	1
4	块间隔奇偶校验	1	8	1
5	块间隔分布式奇偶校验	1	8	1
6	P+Q 冗余	2	8	2

由于 NET-RAM 使用用户级虚存管理,NET-RAM Lib 可以由用户指定冗余方式.对于高级用户,也可由其自行提供冗余策略,定制与应用相关的冗余策略.

2 系统的初步实现

目前,NET-RAM 已在 Solaris,Linux 及 FreeBSD 平台上实现了基本原型,通过 100 MB 的快速以太网和 TCP/IP 协议将 3 台 PC 机互联起来,2 台作为存储节点和存储管理节点运行 NET-RAM daemon 和 NET-RAM Management daemon,1 台运行使用软件 cache 的用户进程.

2.1 NET-RAM 库

NET-RAM Lib 实现了软件 cache,其基本数据

(C)1994-2022 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

结构是 cache 行结构,它记录进程的虚地址,后援存储的存储节点名及相应的后援存储物理信息.当页面(即软件 cache 行)失效时,系统通过失效虚地址和 HASH 函数找到相应的 cache 行结构,分析失效类型,读失效或写失效,进行页面的装载或替换.算法如下:

```
void page_fault_handler(){
    cache_line = find_the_cache_line(virtual_addr)
    if (not found) {
        cache_line = get_free_cache_line();
        if (not_enough) {
            try_replace_one_cache_line();
            cache_line = get_free_cache_line();
        }
        load_cache_line(cache_line);
    } else {
// this should only happen on cache-line without write-enable,
// and process try to write on this cache line
        change_cache_line_state_to_dirty(cache_line);
    }
    resume process running;
}
```

在对 cache 行进行替换时根据 cache 行的状态,Clean 或 Dirty,采取不同的替换步骤:对于 Dirty 的 cache 行要回写存储节点,对于 Clean 的 cache 行就简单地将其置为 Free.

2.2 NET-RAM daemon

NET-RAM daemon 响应软件 cache 的请求装载和回写 cache 行.算法如下:

```
NET-RAM daemon(){
    while (TRUE) {
        request = get_request();
        if (request is WRITE-BACK) {
            write_cache_line();
        } else if (request is READ-CACHE-LINE) {
            send_cache_line();
        }
    }
}
```

2.3 用户级虚存管理(User-Level Virtaul Memory Management)

传统 UNIX 的 mmap 函数提供给用户一定程度的虚存管理功能,但用户还不能干预许多虚存管理的活动,如页面换入、页面换出、页面与后援之间的同步、映射的建立和去除等.这使用户无法高效的实现虚存管理.

为此 NET-RAM 准备增加 mmap 函数的语义,向用户提供虚存管理所需的全部机制,使用户可以在用户级实现虚存管理.NET-RAM 在 Solaris 2.6 上通过实现虚设备来模拟软件 cache 所需的特殊 mmap 行为.

3 性能分析

基于现有的 NET-RAM Lib 进行了实例测试.测试样例是矩阵乘,尺寸为 500×500,数据量约为 6 MB.测试平台是具有 128 MB 内存的 Pentium II 266 MHz 系统,操作系统为 Solaris 2.6,并采用 100 MB快速以太网.表 2 是 500×500 的矩阵乘在软 cache 尺寸分别为 1 MB、2 MB 和 4 MB 时的测试结果.其中 NET-RAM 是以远程节点的主存为后援设备;RAW-DISK 是以本地磁盘作为后援设备,并在每次磁盘操作后进行 I/O 同步操作,这是为真实模拟当用户数据相对于 cache 尺寸很大时系统的实际工作情况.表中的数据为程序从开始执行到得到执行结果所花费的时间.图 2 则是两种方法的直观比较和趋势图.

表 2　采用两种后援方式的测试结果　　s

方法	cache 大小/MB		
	1	2	4
RAW-DISK	354.787 665	133.683 884	70.935 747
NET-RAM	33.913 282	19.422 903	14.001 939

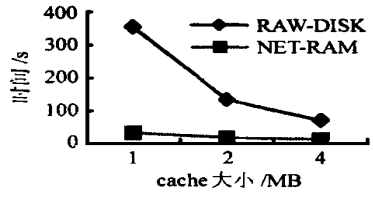


图 2　两种后援方案的直观比较

从表 2 的结果可以看出,以远程节点的主存为后援设备的方法比与以本地磁盘作为后援设备的方法相比,在软 cache 容量为数据总量的 2/3 时,速度快 5 倍;当软 cache 容量为数据总量的 1/6 时速度快 10 以上;而且随着软 cache 容量与数据

(C)1994-2022 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

总量差异的增大,速度的比值有明显的扩大趋势.测试结果证明,采用本地磁盘作为后援设备的方法效率极低,是根本不可取的.

从表2中还可以看出,随着软cache容量与数据总量差异的增大,计算程序的整体执行效率也逐步降低.为了比较单机计算与网络计算的性能,在不使用NET-RAM lib的情况下(相当与用户数据均在内存中)进行了单机测试.单机的计算结果为11.18 s.通过简单计算得知,在以远程节点的主存为后援设备的情况下,效率最高约为单机计算效率的80%(软cache大小为4 MB时),最低为32%(软cache大小为1 MB时).

为进一步探寻原因,进行了更详尽的测试.首先将整体执行时间细分为用户时间(user-time)、系统时间(system-time)和其它时间(other-time).user-time是进程处于用户态的时间(即用户的计算时间);system-time是进程处于核心态的时间;other-time是其它时间,这一部分时间主要是设备的I/O时间,如网络传输延时等.测试结果如图3所示.

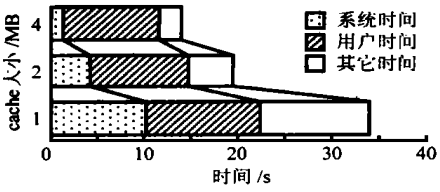


图3 对NET-RAM的直观比较

由图3可知,随着软cache大小从4 MB减小到1 MB,system-time和other-time都有较大的增长,而user-time始终保持与单机计算时间几乎相等.图3、表3中的数据还显示,在使用NET-RAM(100 MB快速以太网)时有17%～35%的时间是花费在网络传输上(使用TCP/IP协议),且从趋势上看,随着软cache的大小与程序需处理的数据总量的差距增大,网络传输时间所占比例还会进一步扩大,系统总体计算性能将进一步下降.我们知道,带宽和延迟是决定网络性能的两个关键因素.由图3数据可以粗略计算一下在测试中所需要的网络带宽,由于请页的数量远大于写回的数量,只计算请页.

在软cache为1 MB时,整个测试过程中请页共15 627次,造成数据传输量为

$$(15\,627\times 4\,\mathrm{KB})/1\,024=6.12\,\mathrm{MB}.$$

在理想状况下(即单机方式)计算时间为11.18 s.这样在主存和后援之间的传输带宽至少为

$$6.12\,\mathrm{MB}/11.18\,\mathrm{s}=0.55\,\mathrm{MB/s}.$$

0.55 MB/s的带宽在100 MB的快速以太网上是很容易达到的,这说明带宽的影响不是主要因素.据测试,在Solaris系统上TCP协议的乒乓延迟为0.833～2.667 ms(数据量由0～4 KB).可见,系统通信延迟是造成整体计算性能下降的主要原因.

表3 NET-RAM的测试结果 s

执行时间	cache大小/MB		
	1	2	4
其它时间	11.51	4.573	2.332
用户时间	12.10	10.630	10.260
系统时间	10.31	4.220	1.410

4 进一步工作

由实验数据分析,网络虚存与网络的I/O性能息息相关.同时,灵活的用户接口,高效实用的环境也必不可少.因此,今后的研究工作将主要集中在以下几个方面:

1) 采用高效和高速的通信接口将进一步提高系统的性能;

2) 提供预取和对cache行大小的更改接口,以增加用户对软件cache的控制能力;

3) 增加系统自动负载平衡机制和用户可控的高可靠性机制;

4) 增加对已有程序的支持.许多用户都有已经编译好的程序,如何支持这些程序很具挑战性,特别是对那些只使用静态数据的程序,如FORTRAN程序;

5) 用户级虚存管理.完善用户级虚存管理的用户接口,提供用户对虚存的更多的控制能力,同时提供安全和保护.这一部分工作将主要集中于Linux和FreeBSD平台.

5 总 结

由于现今网络设备的I/O带宽(Bandwidth)和延迟(Latency)已显著高于磁盘,使用网络虚存将使虚存的整体性能大大提高.同时,充分利用网络上节点的资源可以获得一个很大的物理存储空间,使一些在单机上由于内存限制而无法完成的计算成为可能.通过对SNOW机群系统的分布式共享存储管理机制进行适当的简化,实现了网络虚拟存储环境.本文介绍了NET-RAM的总体结构,软cache的实现方法,为了验证网络虚存的性

(C)1994-2022 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

能实现了 NET-RAM 的原型系统.最后利用 NET-RAM 原型系统进行了测试,比较和分析了测试结果.

通过测试结果可以看出,网络虚存较本地磁盘而言可以大大提升应用的性能,尤其上当应用所需的空间相对物理内存较大时,所获得的性能提升更加明显.性能分析表明网络延时是造成网络虚存性能不能达到理想情况的主要原因.针对这一问题,将着重于改善网络延时特性,通过提供给用户对网络虚存的控制接口,进一步隐藏网络延时对系统整体性能的影响.

参 考 文 献

[1] Mainwaring Alan, Yoshikawa Chad, Wright Kristin. Network RAM, White Paper[DB/OL]. http://now.cs.berkeley.edu/Papers/Papers/nram-wp1.ps.

[2] Michael J Feeley, William E Morgan, Frederic H Pighin, *et al*. Implementing global memory management in a workstation cluster[A]. In:Proc of the 15th ACM Symposium on Operating Systems Principles[C], 1995.201～212.

[3] Hennessy J L, Patterson D A. Computer architecture:a quantitative approach[M]. 2nd ed.San Francisco:Morgan Kaufmann Publishers Inc, 1996.

On Design and Implementation of NET-RAM

LONG Xiang　LIAO Hong-bin　GAO Xiao-peng

(Beijing University of Aeronautics and Astronautics, Dept. of Computer Science and Engineering)

Abstract: A NVM(Network Virtual Memory) system treats all the memories distributed on the network as the memory of a single node so that an application can take the advantage of much larger memory resources on the network. This is especially suitable for such applications which are not time-sensitive but need a large amount of memories. A method of implementing a NVM system by simplifying the distributed shared memory management, which is used in our SNOW clustering system, is introduced. After giving the concept of network virtual memory, we describe the architecture of a network virtual system-NET-RAM and expatiate the structure of soft-cache. The prototype of NET-RAM is tested and the result is analyzed. Finally, we discuss our feature work to improve the NET-RAM system.

Key words: electric networks; virtual store; distributed networks; distributed shared memory; software cache

(C)1994-2022 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net