第 29 卷 第 7 期
2006 年 7 月
计 算 机 学 报
CHINESE JOURNAL OF COMPUTERS
Vol. 29 No. 7
July 2006

一种基于内存服务的内存共享网格系统

褚 瑞 肖 侬 卢锡城
(国防科学技术大学并行与分布处理国家重点实验室 长沙 410073)

摘 要 内存密集型应用对运行环境的物理内存要求严格,在物理内存不足时将会引发大量磁盘 IO,降低系统性能.传统的网络内存致力于在集群内部通过共享空闲节点的物理内存解决该问题,但受集群负载和内部网络影响较大.通过结合网络内存和服务计算、网格计算等技术,提出一种基于内存服务的内存共享网格系统——内存网格,并分析和讨论了实现内存服务的关键技术和算法.内存网格弥补了网络内存的不足,扩展了网格计算的应用范围.通过基于真实应用运行状态的模拟,证明了内存网格与网络内存相比具有性能的提高.

关键词 内存密集型;内存服务;内存网格;页帧服务;代理服务
中图法分类号 TP393

A Memory Service Based Memory Sharing Grid System

CHU Rui XIAO Nong LU Xi-Cheng
(*National Key Laboratory for Parallel and Distributed Processing, National University of Defense Technology, Changsha* 410073)

Abstract Many scientific computing applications demand a great amount of memory, which are usually run on supercomputers with large physical memory. The applications whose complete time is affected by memory capacity, is usually called memory-intensive applications. Many disk IO operations will be caused by memory-intensive applications due to insufficient physical memory of computing environment, which depress the system performance dramatically. Traditional network memory or memory server schemes try to share free memory of idle nodes in the cluster, while it is often influenced by heavy cluster loading or interior network congestion. Combining the network memory with current service computing and grid computing technology, a memory service based memory sharing grid system named RAM Grid, is proposed. RAM grid improve traditional network memory scheme, and extend the application area of grid computing. As one of the major computing resources, memory can be shared on the Internet through RAM grid, such as computing power and disk storage being shared through computational grid and data grid. The system criteria and properties are analyzed. One of the system schemes is also proposed in detail. Through the real trace based simulation, the performance of RAM Grid is evaluated and proved better than traditional network memory or disk IO obviously.

Keywords memory-intensive; memory service; RAM grid; paging service; deputy service

1 引 言

科学计算中存在一种应用程序,对运行环境(内存)有严格的要求.根据我们在气象领域的实践经验,一个普通的气象模拟程序在运行时需要耗用大约 2G 的内存,如果提高模拟精度,对内存的需求量还会增加.相对于传统的计算密集型和数据密集

收稿日期:2006-02-17;修改稿收到日期:2006-05-14.本课题得到国家"九七三"重点基础研究发展规划项目基金(2003CB317008,2005CB321801)和国家自然科学基金(60573135,90412011)资助.褚 瑞,男,1979 年生,博士研究生,主要研究方向为网格计算与高性能并行分布处理技术.E-mail:rchu@nudt.edu.cn.肖 侬,男,1969 年生,教授,博士生导师,CCF 高级会员,研究领域为网格计算和数据网格、高性能并行分布处理技术.卢锡城,男,1946 年生,教授,博士生导师,CCF 高级会员,中国工程院院士,主要研究领域为 MPP 实现技术、分布处理技术和网络技术.

型应用来说,这类应用通常可称为“内存密集型”应用.除了上述科学计算应用之外,我们在研究工作中经常使用的网络模拟器等程序也可以看作是内存密集型应用的常见范例之一.

对内存密集型应用来说,如果运行环境的物理内存不能满足需求,其性能将会出现严重的下降.实践表明,当我们把工作站的物理内存从 4G 降到 1G 之后,气象模拟程序的运行时间将从大约 30s 增加到 10min 以上.性能下降的原因主要来自于两个方面,一方面由于物理内存不足,操作系统通过虚存机制把一部分物理页帧换入磁盘,在需要的时候再次换出,频繁的磁盘操作影响了系统性能;另一方面,当应用本身具有密集的磁盘 IO,物理内存的匮乏也令操作系统无法为磁盘访问提供足够的缓存,增加了磁盘 IO 的开销.Burger 等人的研究结果表明[1],当运行并行程序的计算节点中出现物理内存不足时,由于同步的影响,性能还会进一步下降.鉴于上述原因,科学计算程序必须根据计算环境的内存配置而舍弃一定的计算精度,以免出现大幅度的性能下降.

随着网络技术的发展,20 世纪 90 年代,出现了很多关于“内存服务器”和“网络内存”的研究.其主要思想是设立一台专门提供内存的服务器,或在一个集群内部选择空闲节点暂时充当内存服务器,当某个节点的物理内存不足时,可以把部分内存页帧通过高速网络换入内存服务器,从而代替相对低速的磁盘.但是,网络内存的研究工作遇到了两个主要问题:首先,由于网络内存通常是在集群内部实现,大量的换页操作可能会导致内部网络拥塞,使性能大幅下降;其次,专用的内存服务器成本高,效率低,而如果采用空闲节点作为内存服务器,又可能出现整个集群的内存都不够用的问题.相关研究表明[2],由于网络拥塞和集群负载的影响,在某些情况下,网络内存的性能甚至比传统的磁盘换页机制更差.

近年来面向服务计算技术的蓬勃兴起,使我们考虑把网络内存的思想通过面向服务和网格计算的技术[3~5]应用到广域网络,甚至是 Internet 上,在具有空闲内存的节点上构建一种临时存储内存页帧的服务,从而解决过去的网络内存遇到的问题.从上述思想出发,我们提出了“内存服务”(memory service)和提供内存服务的“内存网格”(RAM grid)的概念.内存网格试图构建一种针对广域网、以“内存服务”为核心的内存共享网格系统,从而缓解计算节点中的物理内存匮乏问题.传统意义上把网格分为面向计算资源共享的“计算网格”和面向数据资源和海量存储资源共享的“数据网格”两类[6],内存网格可看作是这两类传统网格的补充.

从另一个角度来看,我们致力于在开放的网络基础设施之上,以网络资源的聚合与协同为核心,建立一个虚拟计算环境(iVCE).iVCE 通过对分布自治资源的集成和综合利用,为终端用户或应用系统提供和谐、安全、透明的一体化服务环境.内存网格的研究基于 iVCE 的体系结构和核心机理,也可以看作 iVCE 的实例之一.

内存网格的应用范围主要体现于两个方面,首先,对于内存密集型应用,使用内存服务代替传统的磁盘换页机制,将有效减少由于换页导致的巨大性能开销;其次,对于大量非连续的磁盘 IO 操作,如基于大型数据库系统的应用等,内存网格也能够在物理内存不足的情况下提供大容量、高速的磁盘缓存,从而提高系统的整体性能.

和传统的网络内存技术相比较,内存网格从背景、适用范围到体系结构和关键技术都有很大的区别,如表 1 所示.

由表 1 可见,内存网格具有资源丰富等优势,但也存在动态性、异构性等不利因素.要构造一个支持内存服务的网格系统还面临很多挑战.我们的贡献不仅在于首次提出了内存服务和内存网格的概念,

表 1 网络内存和内存网格的主要区别

	网络内存	内存网格
适用范围	集群内部	广域网络,特别是园区网、城域网内部
节点数量	较少(通常不超过 128 个)	多,可用内存资源丰富
网络条件	好,但存在局部拥塞现象	差,但广域网络的路由机制能够避免拥塞
体系结构	集中式管理的结构	分布式结构
资源信息	有全局的资源信息	无全局的资源信息,信息获取开销较大
异构性	所有集群节点完全同构	节点具有很强的异构性
动态性	节点状态稳定、可预测	节点随时可能加入或退出,网络条件、节点负载等随时可能发生变化
自治性	所有节点接受统一管理	节点自治性很强,无集中控制
可扩展性	弱	强
安全或可信性	基本不需要考虑	需要重点考虑
标准化	采用内部协议,非标准化	实现标准化内存服务

还针对上述挑战性问题，借鉴面向服务的网络计算的思想，提出了内存网格的设计原则和实现机制，并在实际的气象模拟应用中采集相关运行状态，根据采集到的实际数据进行模拟和评估.

本文第 2 节介绍相关研究工作；第 3 节分析一些关键问题；第 4 节具体提出一种基于内存服务的内存网格设计方案；第 5 节给出模拟实验及分析结果；第 6 节小结当前工作.

2　相关工作

为了减少物理内存不足带来的性能开销，通常可以采取两方面的手段：(1)考虑如何避免物理内存不足的现象发生；(2)考虑在物理内存不足时减少性能开销.

根据第 1 种思路，Burger 等提出[1]，对于并行程序来说，内存资源和 CPU 资源都应该实现“组调度”；Batat 等具体提出了基于内存分配的组调度方法[7]. 他们的研究工作都试图通过合理的任务调度，使得每个节点的内存基本能够满足任务的要求，尽量避免物理内存不足的情况发生. Batat 等甚至提出，当某个计算节点的物理内存不足的时候，即使 CPU 资源空闲，任务宁可暂时等待也不应该被调度到该节点上[7]. 然而，这种方法面临两方面的问题：(1)多数应用都很难按照 CPU 和内存两方面的需求而随意分割，而且很难实现充分考虑了 CPU 和内存两种需求的调度方法. (2)应用程序对内存的需求量是无法准确估计的. 在 Batat 等人的工作中，提出了使用以前的执行情况和执行文件中包含的信息进行内存需求量的估计[7]，但这种估计方法不能应对动态分配内存的情况.

网络内存的方法主要采用了前面提出的第 2 种思路，即在物理内存不足时尽量减少性能开销. Feeley 等提出了经典的全局内存管理系统[8]，称为 GMS(Global Memory Service). 他们不仅完整地提出一种网络内存的方案，并且比较全面地考虑了相关问题及其解决方法，如页帧替换算法问题、可靠性问题等等. GMS 在操作系统层面上实现，它维护了一个分布式目录，对所有内存页帧进行统一管理，每个页帧都具有全局唯一的 ID. 应用程序根据分布式目录的指导，把换下的页帧优先存入网络内存中，从而获得比本地磁盘更优的性能. Feeley 等人在其后续工作中还对 GMS 进行了深入的分析和改进，如采用页共享、小页帧传输、预取等方法提高了 GMS 的性能，并针对 GMS 中的内存服务负载问题进行了分析[9,10]. Acharya 等人的工作通过大量采集运行数据，分析了不同节点上空闲内存的出现概率和维持时间等重要参数，并在实验基础上，实现了一种名为 Dodo 的内存资源管理系统[11]. 和 GMS 类似的是，Dodo 也采用了全局管理的机制，统一管理内存资源并进行分配. 和 GMS 不同的是，Dodo 没有在操作系统层实现，应用程序对网络内存的申请和使用通过相应的 API 进行，不仅需要选择把哪些数据保存在网络内存中，还需要选择使用哪个空闲节点提供的网络内存，没有实现对应用的透明性. Dodo 的管理策略相对简单，比如随机选择空闲节点、在不能获取网络内存的时候反复重试等等，在某些情况下会严重影响系统效率. 在 Markatos 等人的工作中[12]，也实现了一种类似于 GMS 的网络内存，称为 RMP(Remote Memory Pager)，他们的研究重点在于系统的可靠性，通过设立校验服务器，实现了对网络内存的校验，在保证系统可靠性的前提下，提高了系统利用率. Xiao 等人的工作[13,14]则结合了任务迁移和网络内存这两种方法，试图首先减少物理内存不足的出现频率，再降低物理内存不足时带来的开销.

以上各种网络内存的方法都是在集群内部实现的，Oleszkiewicz 等人的研究结果表明[2]，传统的网络内存对整个集群的负载和内部网络状况非常敏感，在整个集群都处于重负载的情况下，或者是出现内部网络拥塞的时候，网络内存的性能急剧下降. Oleszkiewicz 等人的贡献还在于提出了并行网络内存的概念，他们借鉴 P2P 等网络计算的思想，设计了多种不同的设计方案以协调集群内部的内存分配，避免了内存分配的不均衡，但协调的效果仍然非常有限.

3　系统分析

网络内存对整个集群的负载和内部网络状况非常敏感，内存网格也具有类似的特性. 但是，由于在广域网上可用的资源非常丰富，整个网格环境都出现高负载的概率很小；而众多的可用资源和广域网的路由机制也令我们能够避开网络拥塞的状况. 因此，我们认为内存网格克服了网络内存的不足之处，扩展了使用范围.

然而，广域网的诸多特点也成为内存网格构造的难点所在. 与集群内部的局域网络相比，广域网的

带宽小、延迟大，这意味着并不是所有空闲的网络资源都适合提供内存服务. 我们的实践过程表明，无论是网络还是磁盘传输，延迟带来的影响都占主导地位. 根据 Patterson 等人的研究[15]，目前硬盘的延迟大约在 5～35ms，内存的延迟大约在 52～225ns，两者之间存在巨大的差距. 而广域网的延迟通常有很大的差异，从数百微秒到数秒不等. 如果能够对某个计算节点选择恰当的服务节点，使得其延迟处于硬盘和内存之间，这样的内存服务就有可能在计算节点的物理内存不足时提高其性能. 目前我们经常使用的校园网、园区网或城域网往往都具有较低的延迟，根据实际测试，校园网内部节点间的网络延迟均低于 1ms，而在同一城市的节点间，网络延迟也大都不超过 3ms. 随着互联网技术的发展，Gilder 定律说明了网络的性能还会保持高速增长. 因此，我们认为构造内存服务是有意义的，关键是需要找到网络条件好的空闲资源.

与集群中的节点相比，广域网上的资源耦合度更低，它们往往分布在不同的地理位置，且数量众多，这一特点使经典的网络内存方案采用的全局资源信息很难获取和维护，必须借鉴现有的网络计算技术，实现有效的分布式资源信息管理；广域网上的资源通常是异构的，如硬件支持的页帧大小可能不一致、使用的网络协议可能不一致等，需要我们借鉴面向服务的计算技术，抽象出统一的内存服务，屏蔽底层资源的异构性；此外，广域网上的资源具有很强的动态性和自治性，每个节点都可能具有不同的本地管理策略，且随时有可能因为资源占用的变化或其它原因撤销其提供的内存服务，我们无法对每个节点都实施全局统一的控制和管理，必须采用面向服务的协同工作等技术.

为简化问题，本文遵循下面两条基本假设：

(1) 可靠和安全

内存服务的可靠性和安全性至关重要，一旦提供内存服务的节点突然崩溃，存储在该节点的页帧可能无法找回；如果提供内存服务的节点具有恶意，还可能非法获取其它节点的私有数据，甚至篡改数据等等，造成严重的后果. 但是，可靠和安全的系统设计可以看作是独立的子问题，作为一种初步的设计，本文暂时不考虑这方面的问题. 目前，只需要计算节点把交给内存服务的页帧在本地硬盘中保留一个备份，并使用简单的校验技术，就可以回避大部分可靠性和安全性的问题.

(2) 无内部磁盘缓存

目前所有的硬盘都具有少量的内部缓存，其延迟和带宽等参数和内存相近. 内部缓存使我们对磁盘性能的分析变得复杂化. 但是，如果我们假设某节点物理内存容量是 C_{RAM}，而内部缓存容量是 C_{CACHE}，由于内部缓存和物理内存相比，是对软件完全透明的，无法由软件控制，可以认为该节点在物理内存不足时，其性能低于物理内存容量为 $C_{RAM}+C_{CACHE}$、且无内部磁盘缓存的节点. 进而，由于 C_{RAM} 通常远大于 C_{CACHE}，我们可以暂时忽略内部磁盘缓存带来的影响.

此外，本文提出的内存服务基于下面两条基本设计原则：

(1) 单一服务原则

这条原则是指，内存服务在同一时刻只能被一个节点使用. 假如一个节点 A 正在使用内存服务 S，那么即使 S 还有空闲的服务能力，也不能在 A 释放服务前给另一个节点 B 提供服务. 单一服务原则可能会造成内存服务的浪费，但也有两个明显的优点：其一是简化了资源信息管理，由于 S 只给 A 提供服务，那么 A 可以随时计算出 S 剩余的内存资源，不需要进行额外的查询. 其二是由于内存服务的性能是非常敏感的因素，单一服务原则保证了通过内存服务存取页帧的时候不会出现多个节点排队的情况，最大程度地保证了服务性能.

(2) 单一身份原则

这条原则是指，如果节点 A 正在使用内存服务 S，那么节点 A 不应该同时为其它节点提供内存服务. 反之，如果 A 同时为另一个节点 B 提供内存服务，那么对于 A 来说，由于本地内存的性能大大优于网络内存，如果 A 撤销为 B 提供的内存服务，同时减少甚至完全取消使用内存服务 S，将对其自身的性能有所提高. 考虑到 A 是广域网上的自治节点，我们认为单一身份原则是合理的，并且能够有效简化问题.

4 系统设计

4.1 内存服务

内存网格是一种基于服务的网格系统，通过把分布、异构的内存资源封装成为统一的内存服务，并采用目前的服务发布、发现和管理技术对内存服务实施组织，将有利于屏蔽底层的异构性和地理位置. 比如，不同体系结构的计算机支持的页帧大小可能不同，而内存服务将指定一个基本的页帧大小，内存服务中所有的操作都以此作为基本单元. 需要指出的是，由于对性能要求较高，内存服务无法基于目前

的 Web 服务技术构建，而必须有独立的、重点对性能优化的协议，该协议的制订是我们将来的工作之一.

内存服务的基本功能是为物理内存匮乏的计算节点存储一部分内存页帧，并在需要的时候，将这些页帧以超过本地磁盘的速度传回计算节点. 由于目前广域网范围的资源发现机制是比较耗费时间的操作，如果为此浪费过多的时间，内存服务也就失去了存在的价值. 为此，我们考虑把内存服务分为两类：一类专门用来存储和返回内存页帧，称为“页帧服务”；另一类作为计算节点的代理，专门用于查找合适的页帧服务并移交计算节点使用，称为“代理服务”. 把内存服务进一步区分为页帧服务和代理服务，使计算节点只需要查找一个代理服务，而在其执行繁重的科学计算任务的同时，由代理服务进一步查找页帧服务；代理服务将维护一个可用的页帧服务的列表，在计算节点需要使用网络内存的时候向其提供合适的页帧服务，从而实现了时间的重叠，提高了效率.

为了兼顾系统性能和设计的简洁，页帧服务和代理服务都遵循前文中的“单一服务原则”，即每个页帧服务和代理服务在同一时刻只能为一个计算节点提供服务. 与此同时，我们规定每个计算节点可以同时使用多个页帧服务，但最多只使用一个代理服务. 换言之，页帧服务和服务的使用者之间形成多对一的关系，而代理服务和服务的使用者之间则是一对一的关系. 如图 1 所示.

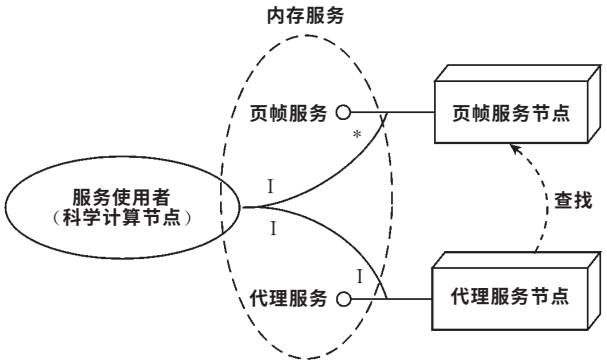

图 1　内存服务的分类及其关系

下面将详细分析这两种内存服务的基本操作和机制.

4.2　页帧服务

页帧服务的主要功能是提供网络内存，使用服务的节点可以把暂时不需要的内存页发送给页帧服务，并释放这部分内存空间；待需要这些内存页的时候再回收. 由于代理服务将会收集若干个页帧服务，但不一定都会使用，因此，我们规定页帧服务在使用前需要经过预约.

代理服务将根据用户的需要，找到若干页帧服务，并给每个页帧服务发送一个预约请求. 页帧服务响应预约请求后，代理服务将在其维护的可用页帧服务列表中增加一条记录，而页帧服务也会记录下已提出预约的代理服务. 一个页帧服务在未使用前可以同时响应多个代理服务的预约，当它开始被使用的时候，需要给每个已提出预约的代理服务发送一个取消预约的消息，令代理服务将其从可用页帧服务列表中删除，如图 2 所示.

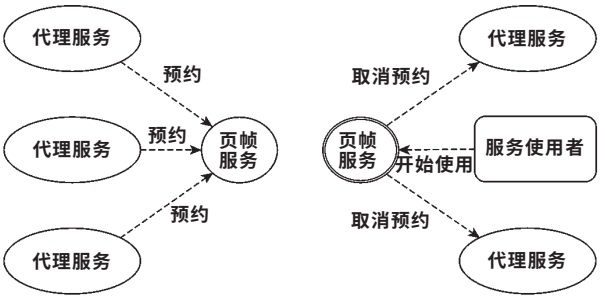

图 2　页帧服务的预约和取消预约过程

页帧服务在使用过程中，可能被使用者显式地释放，也可能由于自身的资源占用率上升、不适合继续提供服务等原因引起服务的撤销. 当页帧服务需要撤销的时候，必须把所有已经保存的用户内存页移交给另外一个页帧服务，以免造成用户内存页的丢失. 页帧服务撤销的流程如下所示：

1. 把用户的内存页暂时换入本地磁盘，释放所有被共享的网络内存空间，从而保证本地的内存需求能够优先得到满足；

2. 为用户查找若干个备用的页帧服务；

3. 根据找到的备用页帧服务能够提供的网络内存大小、到用户节点的网络延迟和带宽等参数，选择合适的备用页帧服务；

4. 把用户的内存页移交到备用页帧服务；

5. 通知用户节点页帧服务已经转移.

每个页帧服务能够提供的网络内存的大小，取决于提供该服务的节点本身的内存容量和部分历史信息. 我们令 M_{total} 为提供页帧服务的节点所拥有的全部物理内存页数，M_{local} 为该节点被本地的系统和应用软件所占用的页帧数，M_{share} 为页帧服务能够共享的网络内存页数. 那么 M_{share} 可以用下列方法计算：

(1) 当页帧服务第一次被使用的时候，令 $M_{\text{share}}=M_{\text{total}}-M_{\text{local}}\times r$，式中的含义是，由于该页帧服务第一次使用，尝试着把节点空闲内存的一部分作为网

络内存，通过页帧服务给其它节点共享. 其中 r 称为松弛因子，$r \geq 1$，通常我们取 $r=2$. r 的作用是保留适当的空闲内存，以备本地节点将来使用. 其取值需要系统管理员根据本地应用的实际情况进行设置，r 取值越大，保留的空闲内存越多. 当 $r=2$ 时，保留的空闲内存和目前已使用的内存容量相同；

(2) 如果页帧服务在使用的过程中发生了服务撤销，说明被共享的网络内存页数太多，造成了本地物理内存不足的情况，这时需要减少 M_{share}，令 $M_{\text{share}}=M_{\text{share}}/2$；

(3) 如果页帧服务被使用者显式地释放，说明共享的网络内存页还不够多，本地剩余物理内存尚充足，这时通过下式尝试增加共享的网络内存：

若 $(M_{\text{total}}-M_{\text{local}}\times r)>M_{\text{share}}$；

则 $M_{\text{share}}=M_{\text{share}}+((M_{\text{total}}-M_{\text{local}}\times r)-M_{\text{share}})/2$；

否则 M_{share} 保持不变.

在上述过程中，需要持续检测页帧服务对本节点的内存占用，以便确定 M_{local}. 同时，一旦在服务被使用的过程出现了 $(M_{\text{total}}-M_{\text{local}})<M_{\text{share}}$ 的情况，则该页帧服务需要撤销，以保证本节点使用内存的需求.

4.3 代理服务

代理服务的主要功能是为用户查找适当的页帧服务，并在用户需要的时候，发送可用的页帧服务列表. 如果用户对网络内存的需求一再增加，还有可能需要多次发送页帧服务列表. 代理服务和用户节点之间是一对一的关系，也就是说，当用户节点需要使用网络内存的时候，有且仅有一个代理服务为其查找合适的页帧服务，直到用户节点不再需要网络内存为止.

为了确保效率，我们规定代理服务不需要进行预约. 当用户节点开始准备使用网络内存的时候，系统的主要机制如下：

1. 用户节点查找到若干个可用的代理服务，并从中选择网络延迟最小的一个；

2. 用户节点申请使用代理服务，并得到回应；

3. 用户节点把当前所有需要存储在网络内存中的页帧全部发送到代理服务，由后者暂存，并释放这些页帧耗用的内存空间. 之所以由代理服务临时暂存，而不是等待找到页帧服务后再保存这些内存页，是为了优先保证用户节点的性能；

4. 代理服务开始查找可用的页帧服务，并要求每个找到的页帧服务测试其对用户节点的网络延迟，按照延迟从小到大的顺序排序，假设排序后的页帧服务集合为 $\{P_n\}$，$n=0,1,2,3\cdots$；

5. 代理服务选择最小的正整数 S，令 $\sum_{n=0}^{S} M_{\text{share}\,n} \geq M_{\text{U}}$ 成立. 其中 $M_{\text{share}\,n}$ 为页帧服务 P_n 可以共享的网络内存页数；M_{U} 为用户节点的内存总量(以页帧数为单位)；

6. 代理服务对页帧服务 $P_0, P_1, P_2, \cdots, P_S$ 发出预约请求；

7. 代理服务把步 3 中暂存的用户节点的内存页全部发送给页帧服务 P_0(如果 P_0 共享的网络内存不够，则继续使用 $P_1, P_2, \cdots$). 把 $P_0, P_1, P_2, \cdots, P_S$ 的列表全部发送给用户节点并清空该列表. 告知用户节点，其内存页已转移到页帧服务；

8. 代理服务继续查找更多的页帧服务，并按照对用户的网络延迟排序后选择最小的 S，令再次找到的页帧服务 $P_0, P_1, P_2, \cdots, P_S$ 满足 $\sum_{n=0}^{S} M_{\text{share}\,n} \geq (M_P+M_{\text{U}})\times b$，其中 M_P 为用户节点所使用以及置入列表准备使用的页帧服务所提供的网络内存的总和，b 称为备份因子，设置备份因子的原因是页帧服务 $P_0, P_1, P_2, \cdots, P_S$ 可能会被其他用户抢先使用，从而取消预约，导致剩余的页帧服务不够. b 也需要根据实际情况进行设置，通常取 $b=1.5$；

9. 代理服务把 $P_0, P_1, P_2, \cdots, P_S$ 加入当前维护的页帧服务列表，并发出预约请求；

10. 如果用户节点耗尽了所有的网络内存，则依次执行步 3、步 7～9.

在理想情况下，每次选择的页帧服务都恰好满足 $\sum_{n=0}^{S} M_{\text{share}\,n}=(M_P+M_{\text{U}})\times b$，如果忽略备份因子的影响，即令 $b=1$，则第 n 次选择的页帧服务所能够提供的网络内存的总量为 $2^{n-1}M_{\text{U}}$. 其中 M_{U} 可看作常数，故上式是指数函数. 由于用户需要的网络内存无法预知，按照上式查找页帧服务，能够保证在有限的查找的次数内迅速达到用户需要的网络内存总量.

前文中曾提及，如果一个页帧服务被多个代理服务所预约，那么当这个页帧服务开始使用的时候，需要向所有预约过的代理服务发出取消预约的消息，表示以前的预约已经失效. 对于一个代理服务来说，可能随着时间的推移，失效的预约越来越多，在必要的时候需要重新预约一些页帧服务备用. 我们设计的判断条件为 $\sum_{n=0}^{S} M_{\text{share}\,n}<(M_P+M_{\text{U}})$，当满足该条件时，需要重新执行上面的步 8 和步 9，预约足够的页帧服务.

和页帧服务一样，代理服务也可能由于自身资源占用过高等原因造成服务撤销. 由于代理服务的特殊性质，我们规定，代理服务撤销的判断条件是 CPU 占用率持续高于某个阈值，或者剩余的内存太

少，已经不够暂存用户换下的页帧. 当代理服务撤销的时候，也需要找另一个备用的服务，把当前的工作状态移交. 其过程和页帧服务的撤销类似，不再赘述.

5　验证和评价

我们采用离散事件模拟的方式，模拟了一个内存网格环境并进行性能验证. 为了真实反映实际系统的特点，我们利用实际使用中的气象模拟程序，收集了部分运行状态，并将其作为模拟的参数.

我们选择气象模拟应用程序在运行过程中，由操作系统进行的内存页换入和换出作为运行状态. 具体的信息包括每次换页的时间(精确到微秒)、换入还是换出、对应的磁盘块号. 我们使用的硬件配置是具有两个 Xeon 3.6GHz CPU、2GB 物理内存的 Linux 工作站. 首先把气象模拟程序成功地移植到 Linux 操作系统上，然后通过修改 Linux 2.4.22 版本内核，把程序运行过程中每次换页信息记录下来.

利用这些信息，我们模拟了一个具有 1000 个网格节点的环境，包括 890 个 PC、100 个工作站和 10 个集群，每个集群包含 32 个子节点. 每个节点或子节点的物理内存从 128MB 到 4GB 不等，根据节点类型随机生成. 网络拓扑采用 BRITE 软件的 ASWaxman 模式生成一个层叠网，其数学模型是一个含边权重的无向图，如果两个节点在图中没有直接相连，则采用 Dijkstra 算法模拟一个最短路由. 工作站和集群定期运行气象模拟程序，为了体现集群中应用性能受到同步操作的影响，我们在集群中每隔 1 秒钟模拟一次障碍同步，每次参与同步的节点数为 4. 内存服务由资源中暂时空闲的节点提供.

我们采取下式评估网络传输的开销：

$$O_n = T_U + T_{\mathrm{RTT}} + n \times \frac{S_P}{B_N},$$

其中，n 和 S_P 分明代表传输的页帧数和页帧大小；T_U 是启动传输操作的时间，令 $T_U = 5\mu s$；T_{RTT} 是网络传输的双向延迟时间，在集群内部，我们取 $T_{\mathrm{RTT}} = 150\mu s$，在广域网上，令 T_{RTT} 的平均值为 3ms(即单向延迟平均为 1.5ms)；B_N 是网络带宽，在集群内部，取 $B_N = 125\mathrm{MB/s}$；在广域网上，令 B_N 的平均值为 10MB/s. 上述参数取值是根据我们在校园网上实测数据设定的，具有一定的代表意义.

磁盘传输的开销则由下式评估：

$$O_d = T_S + T_L + (n-1) \times T_W + n \times \frac{S_p}{B_d},$$

其中 T_S 是寻道时间；T_L 是磁盘旋转的等待时间；T_S 和 T_L 的和代表了一次磁盘访问的延迟；T_W 是访问两个连续的磁盘块时需要额外等待的时间；B_d 是磁盘的外部传输带宽. 我们采用 Seagate 公司企业级硬盘产品捷豹系列的典型参数，令 $T_S = 4.9\mathrm{ms}$，$T_L = 3.0\mathrm{ms}$，$T_W = 0.2\mathrm{ms}$，$B_d = 80\mathrm{MB/s}$.

网络传输和磁盘传输的开销将直接影响到应用运行的平均响应时间. 我们把平均响应时间作为模拟过程中的主要评估目标.

内存网格的性能很大程度上受广域网状况的影响，我们首先评估其性能随广域网条件变差而变化的情况. 从图 3 可见，内存网格的性能总体上随着网络带宽的减少和延迟的增加而下降. 但是，即使平均带宽下降到 2MB，延迟增加到 4 倍(平均 6ms)，平均响应时间也未超过 200s；而如果不使用内存网格，磁盘 IO 操作的平均响应时间约为 291s，可见内存网格对系统性能有显著的提升. 另一方面，从图 3 可以看出，在带宽不同的内存网格环境中，几条折线之间的距离差别不大，而每一条折线都随着延迟的增加有明显的增长，这说明内存网格的性能受延迟的影响比带宽更大，这是因为每个内存页帧只有 4KB，即使每次传输的页帧不止一个，传输时间也主要消耗在网络延迟和磁盘寻道上.

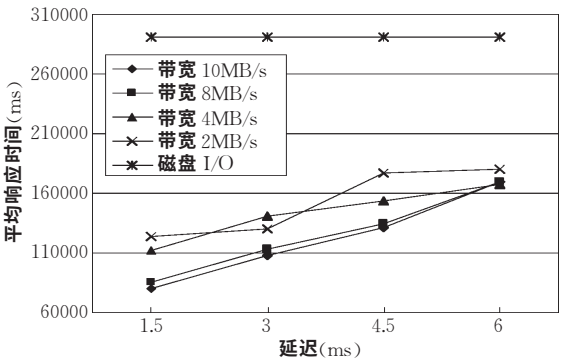

图 3　内存网格的性能和网络延迟、带宽的关系

下面我们评估内存网格的性能受负载等参数影响的情况. 作为对比，我们还模拟了传统的网络内存的性能(网络内存的作用范围局限于集群内部)，此外还包括不使用内存网格或者传统网络内存时的情况，此时的系统性能受磁盘传输速率影响非常严重.

在基本模拟条件下，我们假设工作站和集群都会隔 10～120min 运行一次内存密集型应用. 如果改变运行应用的时间间隔，就相当于系统负载发生了变化，间隔越短，平均负载越重. 结果如图 4 所示. 如果不使用内存网格和网络内存，则每个节点的运行相对独立，也就不受应用程序运行间隔的影响，在

图中表现出一条直线. 而内存网格和网络内存都需要利用空闲节点存储页帧,所以当应用程序的运行间隔增大的时候,在同一时间内出现的空闲节点增多了,我们看到,无论是内存网格还是网络内存,其曲线都大体呈下降趋势,且内存网格全面优于传统的网络内存.

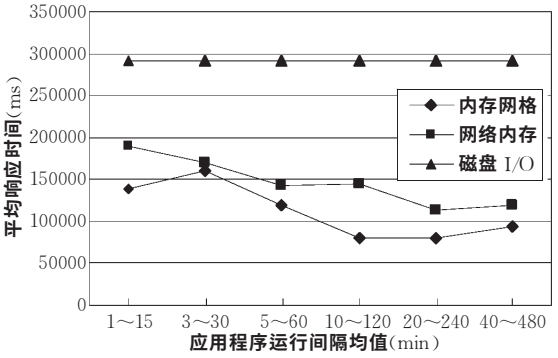

图 4 应用程序运行间隔时间变化对性能的影响

节点运行的应用程序分为三种,分别是轻负载型应用(内存需求不超过本机物理内存)、中负载应用(内存需求略高于本机物理内存)、重负载应用(内存需求超过本机物理内存一倍以上). 在基本模拟条件下,这三类应用是按照 20∶60∶20 的比例随机产生的. 我们尝试着改变该比例,结果如图 5 所示. 随着重负载应用的比例上升,系统性能急剧下降,在不使用内存网格和网络内存的情况下,下降尤其明显. 由于内存网格通过内存服务共享了更大范围的资源,其性能总体上优于传统的网络内存,在系统负载较重的时候差异更为明显.

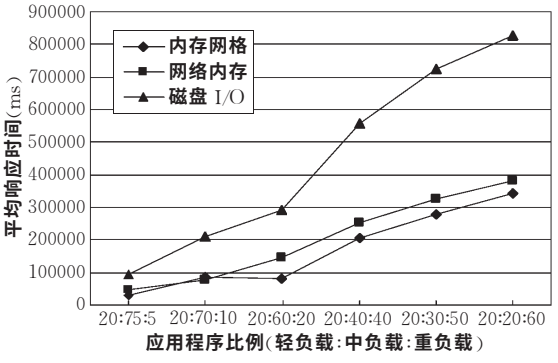

图 5 应用程序比例变化对性能的影响

另一方面,我们也能看出无论是在图 4 还是图 5 中,内存网格的性能都随着系统负载的增加而变差,即使有足够多的可用内存服务也不能避免这种情况. 这是因为无论是传统的网络内存还是内存网格,本质上都没有减少物理内存不足时发生换页的次数. 在我们的下一步工作中将针对这个问题进行研究.

在基本模拟条件下,PC、工作站和集群这三类节点的比例分别是 89∶10∶1. 我们尝试改变这一比例,结果如图 6 所示. 随着横坐标的增加,工作站与集群的比值逐渐减少,这时如果不采用内存网格或者网络内存,性能会有所下降. 因为集群中运行的应用程序要受到同步操作的影响,在发生换页的时候性能下降更多;传统的网络内存表现不稳定,但总体而言,集群所占比例较大时,其性能也较好;内存网格的性能最为稳定,且总体优于传统的网络内存.

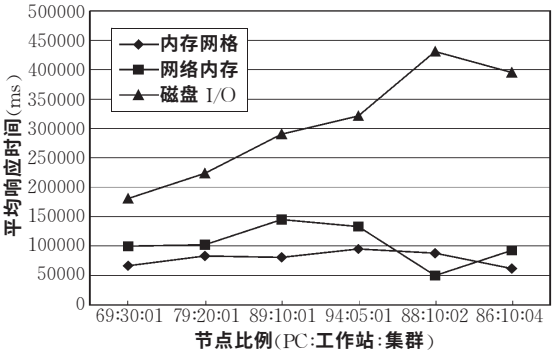

图 6 节点比例变化对性能的影响

综上所述,在校园网等较好的广域网环境下,内存网格对于内存密集型应用具有较大的性能提升;和传统的网络内存相比,由于应用范围更为广泛,也具有一定优势.

6 结束语

内存密集型应用在物理内存不足的运行环境中会出现性能大幅度的下降,其根本原因是出现了大量的磁盘 IO 操作,影响了性能. 传统的网络内存技术能够有效缓解这一问题,但由于网络内存技术限于集群内部,对集群的负载和集群内部网络拥塞比较敏感.

我们借助服务计算和网格计算的思想,提出了基于内存服务的内存网格,不仅弥补了传统的网络内存技术的不足,也扩展了网格技术的应用范围. 内存网格试图把广域网上空闲节点的内存进行共享,从而降低磁盘 IO 的开销,提高系统性能;需要指出,和传统的网格计算类似,使用内存网格的代价是增加空闲节点的负载,并需要额外地占用网络带宽进行数据传输.

通过对内存网格的分析,我们提出了一种可行的内存网格实现方案,把内存服务分为页帧服务和代理服务,有效地保证了系统性能. 通过基于真实应

用运行状态的模拟，证明了内存网格能够有效提高内存密集型应用的性能.

参 考 文 献

1 Burger D. C. *et al*. Paging tradeoffs in distributed-shared-memory multiprocessors. In: Proceedings of the Conference on Supercomputing, Washington D. C., USA, 1994, 590～599

2 Oleszkiewicz J., Xiao L., Liu Y.. Parallel network RAM: Effectively utilizing global cluster memory for large data-intensive parallel programs. In: Proceedings of the 2004 International Conference on Parallel Processing, Montreal, Quebec, Canada, 2004, 353～360

3 Foster I. *et al*. Modeling stateful resources with Web services. Argonne National Laboratory, Globus Alliance: Technical Report, 2004. http://www.globus.org/alliance/publications/papers.php

4 Foster I. *et al*. Grid services for distributed system integration. IEEE Computer, 2002, 35(6): 37～46

5 Foster I., Kesselman C.. The Grid 2: Blueprint for a New Computing Infrastructure. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc, 2003

6 Foster I., Kesselman C., Tuecke S.. The anatomy of the grid: Enabling scalable virtual organizations. International Journal of High Performance Computing Applications, 2001, 15(3): 200～222

7 Batat A., Feitelson D. G.. Gang scheduling with memory considerations. In: Proceedings of the 14th International Symposium on Parallel and Distributed Processing, Cancun, Mexico, 2000, 109

8 Feeley M. J. *et al*. Implementing global memory management in a workstation cluster. In: Proceedings of the Symposium on Operating Systems Principles, Copper Mountain Resort, Colorado, 1995, 201～212

9 Jamrozik H. A. *et al*. Reducing network latency using subpages in a global memory environment. In: Proceedings of the 7th International Conference on Architectural Support for Programming Languages and Operating Systems, Cambridge, Massachusetts, 1996, 258～267

10 Voelker G. M. *et al*. Implementing cooperative prefetching and caching in a globally-managed memory system. In: Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems, Madison, Wisconsin, 1998, 33～43

11 Acharya A., Setia S.. The utility of exploiting idle memory for data-intensive computations. University of California at Santa Barbara: Technical Report TRCS98-02, 1998

12 Markatos E. P., Dramitinos G.. Implementation of a reliable remote memory pager. In: Proceedings of the USENIX Annual Technical Conference, San Diego, CA, 1996, 177～190

13 Xiao L., Zhang X., Kubricht S. A.. Incorporating job migration and network RAM to share cluster memory resources. In: Proceedings of the 9th IEEE International Symposium on High Performance Distributed Computing, Pittsburgh, Pennsylvania, 2000, 71～78

14 Zhang X., Xiao L., Qu Y.. Improving distributed workload performance by sharing both CPU and memory resources. In: Proceedings of the 20th International Conference on Distributed Computing Systems, Taipei, Taiwan, 2000, 233～241

15 Patterson D. A.. Latency lags bandwith. Communications of the ACM, 2004, 47(10): 71～75

CHU Rui, born in 1979, Ph. D. candidate. His research interests include the grid computing and high-performance parallel and distributed processing.

XIAO Nong, born in 1969, professor, Ph. D. supervisor. His research interests include the grid computing, data grid and high-performance parallel and distributed processing.

LU Xi-Cheng, born in 1946, professor, Ph. D. supervisor, member of Chinese Academy of Engineering. His research interests include the MPP implementation, distributed processing and network technology.

Background

This work is supported by the National Basic Research Program of China (973 Program) Project "Aggregation and Coordination Mechanisms in Virtual Computing Environment". The Virtual Computing Environment (VCE) will be built on the open infrastructure of the Internet, through integrating distributed autonomous resources connected to the Internet, both of sharing and cooperating among resources become more effective and efficient. It can also exert the tremendous power of the resources on the Internet.

The core mechanisms of the VCE are aggregation and coordination of network resources. The Internet resources have clearly demonstrated three primary characteristics called "growing", "autonomous" and "diverse", which contradict with the traditional method of centralized control in resource management. Therefore, we believe that the new way of aggregation and coordination could be natural to construct VCE. In this paper, the authors extend the traditional network memory scheme using service oriented computing and grid computing technologies, and design the RAM Grid system based on memory services. The approach implements the aggregation and coordination of memory resources in wide-area network; it will be an important part of the VCE.