ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.8, August 2007, pp.1858−1869
DOI: 10.1360/jos181858

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

一种基于聚类的虚拟计算环境资源聚合方法*

褚 瑞+, 卢锡城, 肖 侬

(并行与分布处理国家重点实验室(国防科学技术大学),湖南 长沙 410073)

A Clustering Based Resource Aggregation Method for Virtual Computing Environment

CHU Rui+, LU Xi-Cheng, XIAO Nong

(National Laboratory for Parallel and Distributed Processing (National University of Defense Technology), Changsha 410073, China)

+ Corresponding author: Phn: +86-731-4512695, Fax: +86-731-4575835, E-mail: rchu@nudt.edu.cn, http://www.nudt.edu.cn

Chu R, Lu XC, Xiao N. A clustering based resource aggregation method for virtual computing environment. *Journal of Software*, 2007,18(8):1858−1869. http://www.jos.org.cn/1000-9825/18/1858.htm

Abstract: As an instance of Internet resource sharing based virtual computing environment, iVCE (Internet based virtual computing environment) for Memory try to solve the problem of memory resource sharing and utilization. Due to the special properties of memory resource, traditional resource management approaches can not be adapted easily. A clustering based resource aggregation scheme is proposed under the background of iVCE for Memory, which can reduce the problem scale efficiently. With analogy to the force field and potential energy theory in physics, the basic model, force field-potential energy model and corresponding distributed algorithms are proposed respectively. The models and algorithms are also evaluated by real network topology based simulation.

Key words: virtual computing environment; iVCE (Internet based virtual computing environment) for Memory; resource aggregation; clustering; force field-potential energy model

摘 要: 作为面向互联网资源共享的虚拟计算环境的实例,iVCE(Internet based virtual computing environment) for Memory 致力于解决广域分布的内存资源的共享与综合利用问题.由于内存资源的特殊性,传统的资源管理方法很难适用.以 iVCE for Memory 作为背景,提出一种基于聚类的虚拟计算环境资源聚合方法,有效降低了资源聚合的问题规模;借鉴物理学中的力场和势能理论,建立了实现资源聚合的基本模型和力场-势能模型以及相应的分布式算法;通过基于真实网络拓扑的模拟,对两种模型和算法分别进行了评估和验证.

关键词: 虚拟计算环境;iVCE(Internet based virtual computing environment) for Memory;资源聚合;聚类;力场-势能模型

中图法分类号: TP393 **文献标识码**: A

随着网络和分布式计算技术的不断发展和深入应用,两种技术逐渐融合所形成的网络计算技术已成为学术研究的热点,并正在致力于解决互联网范围内的资源共享和综合利用的问题.为了在资源共享的基础理论和机理上取得新的突破,我们已经在过去的研究工作中提出了面向互联网资源共享的虚拟计算环境(下文简称

* Supported by the National Natural Science Foundation of China under Grant Nos.60673167, 90412011 (国家自然科学基金); the National Basic Research Program of China under Grant No.2005CB321801 (国家重点基础研究发展计划(973))

Received 2007-03-01; Accepted 2007-04-26

iVCE(Internet based virtual computing environment)或虚拟计算环境)的概念[1],并根据广域分布、异构的内存资源共享这一实际问题,利用iVCE的体系结构和核心机理,提出了iVCE for Memory,或称为内存网格的概念[2].iVCE for Memory试图有效聚合目前广域网上大量分布的空闲内存资源,并利用远程内存资源在小粒度访问时性能优于本地磁盘或远程磁盘的特点,为虚拟计算环境中具有大量磁盘操作的节点提供缓存服务.与传统的集群计算技术中的网络内存[3−9]方案不同,iVCE for Memory所关注的资源分布范围更广,远远超越了集群的边界限制,且具备虚拟计算环境成长性、自治性、多样性等重要特点,因此,必须使用虚拟计算环境的机制和方法构建iVCE for Memory.事实上,目前iVCE for Memory是虚拟计算环境的主要实例之一[1],对iVCE for Memory的实例分析已成为我们研究虚拟计算环境的重要手段.

虚拟计算环境的核心机制是对网络资源的聚合与协同.聚合是获取、汇聚、组织资源特征信息的过程;协同是多个资源为完成共同任务而进行交互、同步和计算的过程[1].作为虚拟计算环境的实例,iVCE for Memory同样需要对内存资源进行有效的聚合与协同.具体来说,iVCE for Memory需要把具有内存共享这一共同兴趣的节点进行有效的组织和管理,并加入资源信息发布、组织和发现的基础设施,此过程可看作是内存资源的聚合;还需要描述内存资源共享的任务流程,并指导各节点的任务状态的推进,此过程可看作是内存资源的协同.

iVCE for Memory 的资源聚合与协同机制与一般的分布式计算技术相比有一定的区别.以聚合为例,在传统的网络内存研究中,由于资源范围有限,多使用集中管理的方法,由 1 个或数个管理节点对资源进行全局控制;而对于虚拟计算环境所面向的问题域来说,这种方法的扩展性和可靠性较差.另一方面,在目前大量的网格、P2P等系统的研究和实现中,也提出了很多分布式的资源管理方法,如非结构化与结构化的 P2P 方法等.但非结构化P2P 的资源管理往往耗费较高;结构化的 P2P 资源管理,如 DHT 等方法,通常需要利用文件名等作为资源的唯一标识.而在 iVCE for Memory 中,通常关注的资源信息是网络延迟和内存容量,两者都不适合作为内存资源的标识.总之,过去研究的资源管理方法难以用于 iVCE for Memory.

在本文中,我们基于虚拟计算环境的基本思路,以 iVCE for Memory 为背景,提出一种新的虚拟计算环境资源聚合方法.该方法主要基于下述思想:iVCE for Memory 的本质是利用通过高速网络互联的内存资源,对本地或远程磁盘访问进行加速,而由于网络条件的限制,对于一个特定节点来说,可用的远程内存仅仅是 iVCE for Memory 中所有内存资源的一个较小的子集,而其他内存资源与该节点的网络通信速度可能会比本地磁盘低,因此对该节点并不适用.如果我们能用聚类的方法把 iVCE for Memory 中的所有节点分为若干组,令一个节点在使用远程内存资源时,只需在所处的组内进行查找即可得到大多数可用资源的信息,而其他各组的资源对其适用的概率很低,则该聚类方法对于降低问题的规模是有意义的.特别是当聚类形成的每个组中所包含的节点数量较小时,即使是简单的集中管理方法也能够有效地实现 iVCE for Memory 的资源聚合.

为了研究内存资源的聚类方法,我们首先利用目前已有的网络嵌入(network embedding)方法构建一个网络坐标系统,用于估计任意两节点之间的网络延迟;由于 iVCE for Memory 中的节点及其关系与物理学中质点之间的弹力和万有引力具有类似的特点,我们采取类比的建模方法对内存资源的聚类过程进行建模,并在模型中通过概率分析考虑了虚拟计算环境中成长性、自治性、多样性等特点及相应的对策.基于上述模型,我们提出了两种不同的资源聚类算法,并通过几种真实的物理网络拓扑进行了系统的模拟验证.

本文的贡献主要在于:

- 我们分析了 iVCE for Memory 中内存资源聚合的基本问题,并提出了以聚类为基础的资源聚合机制.该机制不仅适用于 iVCE for Memory,对于其他关注网络延迟及访问开销的资源共享系统来说也同样有意义,是一种重要的虚拟计算环境资源聚合方法.
- 我们提出了内存资源聚合的基本模型和算法,并通过分析其不足,提出了基于物理学中力场和势能的改进模型.改进模型及相应算法能够有效地反映系统的实际特点,并进一步构成了虚拟计算环境中资源聚合的基础设施.
- 通过基于真实物理网络拓扑的模拟,我们从几个方面比较了基本模型和改进模型及其算法的性能,验证了基于聚类的资源聚合方法的有效性.

本文第 1 节介绍相关研究工作.第 2 节提出内存资源的聚类问题、基本模型和算法并进行分析.第 3 节提出基于物理学中力场和势能的模型和算法.第 4 节进行模拟验证并分析其结果.第 5 节总结全文.

1 相关工作

从 20 世纪 90 年代开始流行的网络内存技术[3−9]与iVCE for Memory有着重要的关系.网络内存的发展源于高性能通信网络技术的进步,统计表明[10],磁盘的延迟性能每年提高 10%,而带宽则每年提高 20%;网络的延迟性能每年提高 20%,带宽则每年提高 45%.由于磁盘的机械部件对性能的影响,使得通过高性能网络连接的内存资源的性能在 10 年前已经超过了磁盘,两者之间的差距还在不断扩大.传统的网络内存技术主要有 3 方面的应用:用于虚存换页设备、用于文件系统或数据库的磁盘缓存、用于构建高性能的临时文件系统.Feeley等人提出的GMS[8]是最著名的网络内存系统之一,该系统主要面向虚存换页的内存密集型应用.GMS所管理的内存资源在一个集群内部,它对每个内存页帧都分配一个全局唯一标识,维护每个页帧的年龄信息,并确保全局范围内较老的页帧总是被先换入磁盘.所有的标识分配、年龄维护等全局算法都是在一个管理节点的统一控制下进行的.因此,GMS的资源管理属于典型的集中式方法.其他网络内存系统,如Markatos等人提出的Remote Memory Pager[7]和Hines等人提出的Anemone[4]等,也都采用了类似于GMS的资源管理方法,只是在具体细节上有所变化,如设立一个备份的管理节点等.在Oleszkiewicz等人提出的Parallel Network RAM[5]中,提出了 4 种不同的资源管理策略:集中策略、客户策略、局部策略和骨干策略,并对 4 种策略的性能进行了比较.比较的结果表明,在多数情况下,集中式资源管理方法的性能都不是最优的.

协作缓存技术也是网络内存的一个分支,其研究工作从 20 世纪 90 年代一直持续至今.其中最典型的成果是Dahlin等人的工作[9],他们也提出了 4 种内存资源管理策略,并以一种多节点协作的分布式策略为主,其他策略为辅.Sarkar等人的工作中提出了一种基于线索(hints)的资源管理策略[6],通过在多个节点之间维护一条内存页帧移动链来进行管理.Song等人则提出了一种基于局部感知协议的资源管理方法[3],其性能优于前两者.上述工作都是基于一个集群内部的内存资源管理问题而研究的,在这种情况下,网络通信开销往往可以看作是一个较小的常数.而我们的iVCE for Memory主要关注广域网范围内的资源聚合问题,由于广域网的通信开销大且具有较大波动,因此,对网络延迟的有效估计是一个重要的研究基础.

我们主要借助于目前的网络嵌入方法构建一个网络坐标系统,并根据各个节点在坐标系统中的欧几里德距离估计其网络延迟.网络嵌入方面最具有影响力的研究工作是GNP[11],很多后续工作,如PIC[12],ICS[13],Vivaldi[14]和BBS[15]等都在GNP的基础上进行了不同的改进.上述方案都能把网络中的节点置入一个高维的平面直角坐标系中,并通过节点的坐标进行网络延迟估计,其误差往往较小.与上述方案不同,Meridian[16]并不进行网络延迟估计而是进行实际测试,虽然其开销较大,但精度也较高.无论是基于延迟估计的方案,还是基于延迟测试的方案,都在我们的工作中有具体的应用.

2 系统分析

2.1 概 述

iVCE for Memory 的关键是使用高性能通信网络连接的内存资源对本地或远程磁盘访问进行加速.网络的性能对 iVCE for Memory 的使用效果有重要的影响.我们在表 1 中列出了分别使用具有 2MB 带宽和 2ms 延迟的广域网络连接的内存、典型的企业级磁盘和通过千兆以太网进行连接的磁盘,访问一个 8KB 大小的数据块的时间开销.

由表 1 可以看出,对于一个猝发的磁盘块访问来说,使用本地磁盘和网络磁盘访问的主要开销来源于磁盘和网络的延迟.虽然对于远程内存访问来说,受带宽限制的网络传输占据了大部分时间,但由于网络传输的过程可以通过时间重叠等常见技巧进行隐藏,而延迟带来的影响则很难消除.所以,为了简化问题,我们在后文的模型中只考虑网络延迟和磁盘延迟这两个基本参数.值得注意的是,虽然磁盘在连续访问的时候,其延迟带来的开销所占比例有所降低,从而导致性能大幅度提高,但我们主要考虑的是大型数据库、Web 服务器等具有猝发、

非连续等数据访问特点的应用,从而忽略了磁盘连续访问的情况.

Table 1 Analysis and comparison of data access overhead in different ways (ms)

表 1 几种数据访问方式的时间开销分析和对比 (ms)

	Remote memory	Local disk	Disk in LAN
Memory access	<0.01		
Network latency	2		0.68
Network transmission (depend on bandwidth)	4		0.06
Disk latency		7.9	7.9
Disk transmission		0.1	0.1
Total	≈6	8	8.74

对于网络延迟来说,我们在建模之前进行如下假设:

- 任意两个节点都能通过底层的物理网络互相通信,忽略由于 NAT(network address translation)或者防火墙导致的无法建立网络连接的问题.
- 任意两个节点之间的网络延迟都具有对称性.我们用 $ND(A,B)$代表节点 A 和 B 之间的网络延迟,则有 $ND(A,B)=ND(B,A)$.
- 网络延迟总是符合三角形法则,即 $ND(A,B)<ND(A,C)+ND(C,B)$.

我们对磁盘延迟也进行如下的假设:

- 任意节点 A 的磁盘延迟,用 $DL(A)$表示,都可以看作是一个常数,并且 $DL(A)$能够很容易地被测量出来.
- 任意节点的磁盘容量均为无穷大.

忽略磁盘内部缓存的作用.这条假设的合理性在我们之前的工作中[2]已有解释,这里不再赘述.

我们认为,在 iVCE for Memory 中,如果两个节点 A 与 B 满足关系 $ND(A,B)\geq DL(A)$,那么当 B 提供内存资源时,A 可以使用 B 的内存资源为其提供缓存服务,因为这时 A 从 B 的内存中取一个数据块的速度高于使用本地磁盘.但是对于 A 来说,并非所有节点和 A 之间都满足上述关系,因此,A 只能使用一个较小的节点集合中的内存资源.

虽然传统的集中式资源管理由于扩展性和可靠性等问题而不适用于虚拟计算环境,结构化和非结构化 P2P 中的资源管理方法对于 iVCE for Memory 来说也不完全适用.但是我们注意到,对任意特定节点来说,能使用的内存资源总是 iVCE for Memory 中很小的一部分.因此可以采用集中式与分布式混合的方法,把所有节点通过聚类划分为若干个组,而在各组内部使用集中式的管理方法,由一个组内的节点承担管理任务.这样,如果我们同时限制各组中最多能包含的节点数量,则有效降低了问题规模.需要说明的是,各组内部的管理节点选择可以有很多技巧,受篇幅所限,本文暂时采用最基本的方法,即令每个组中第 1 个加入的节点作为管理节点,更优的管理节点选取办法是我们今后的工作之一.

虚拟计算环境是由各个节点逐步加入而构成的系统.在第 1 个节点加入之前,我们需要围绕这个系统,确立一个公共的资源共享的兴趣范围.根据该兴趣范围,我们把具有共同兴趣、遵从共同原则的资源集合称为虚拟共同体[1],如iVCE for Memory中即为“内存共享虚拟共同体”.虚拟共同体中包含了资源聚合与协同的基础设施和相应部件,如果一个节点具有内存共享的兴趣,即可申请加入内存共享虚拟共同体,并同时下载和安装相关部件.虚拟共同体中的基础设施将为该节点分配唯一标识,通过网络嵌入等方法确定该节点在网络坐标系统中的位置,通知其邻近节点信息等.上述过程完成后,将根据聚类算法引导该节点归入相应的组中.此后,当该节点需要使用其他资源时,也要通过本组的管理节点进行资源查找和匹配.

由于每个节点所能使用的内存资源都被局限于组内,所以我们的聚类算法需要确保以下两点:

- 组内任意两个节点之间的网络延迟需要满足内存共享的基本条件,即至少需要有其中一个节点能够使用另一个节点提供的内存资源;
- 如果 A 能够使用 B 提供的内存资源,则应尽量降低 A 与 B 被划分到不同组内的概率.

我们将对上述问题进行建模.

2.2 基本模型

我们首先尽量简化聚类问题,建立一个基本模型.基本模型中的聚类问题可以如下描述:对于节点集合$\{N_i\}$,$i\in[1,n]$,找出其子集构成的集合$\{G_k\},k\in[1,m]$,满足$\bigcup_{i=1}^{m}G_k=\{N_i\}$且$\forall i,j\in[1,m],i\neq j\Rightarrow G_i\cap G_j=\varnothing$.

限制 1. $\forall i\in[1,m],DIA(G_i)\leq D$.其中,$DIA(G_i)$是$G_i$的直径,即$G_i$中欧几里德距离最远的两个节点之间的距离,$D$是$DIA(G_i)$的上限.

限制 2. $\forall i\in[1,m],S(G_i)\leq M$.其中,$S(G_i)$是$G_i$中节点的数量,也称为$G_i$的大小;$M$是$S(G_i)$的上限.

目标:最小化下述情况的概率:$(A\in G_i)\wedge(B\in G_k)\wedge(i\neq k)\wedge ND(A,B)\leq D$.

在上述模型中,D可以看作是一个典型的磁盘访问延迟值.或者说,我们简单地认为,$\forall i\in[1,n],DL(N_i)=D$.$S(G_i)$的上限$M$可以根据实际情况选取,原则是在分组内部便于实施集中式资源管理.

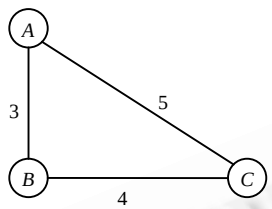


Fig.1 An example of basic model

图 1 基本模型的一个例子

上述模型的解不是唯一的.如图 1 所示,假设D=4.5,则模型的解可以是$G_1=\{A,B\},G_2=\{C\}$,也可以是$G_1=\{A\},G_2=\{B,C\}$.显然,这里$G_1=\{A,B\}$,$G_2=\{C\}$比$G_1=\{A\},G_2=\{B,C\}$更优,因为对于B来说,它与A的网络延迟更低,更应该被划分到一个组中.

在该模型中,$(A\in G_i)\wedge(B\in G_k)\wedge(i\neq k)\wedge ND(A,B)\leq D$的含义是,$A$和$B$的网络延迟较低,令它们能够互相使用对方提供的内存资源,但却不在同一个组内.我们称其为A和B的遗漏,并用谓词$Miss(A,B)$来表示.那么,考虑到图 1 中两组解的优劣之分,我们修改基本模型,增加一个目标:

目标 1. 最小化 $Miss(A,B)$的发生概率.

目标 2. 最小化 $\sum_{\forall A,B(A\in G\wedge B\in G)}ND(A,B)$.

虽然只是基本模型,但这个聚类问题却具有一定的复杂度,它不仅是非线性的,而且有多个目标.显然,对于这样的问题很难在线性时间内找到最优解.特别是解上述问题的算法必须是分布式的,没有集中控制,且需要满足各个节点先后加入 iVCE for Memory 的情况.我们提出一种贪婪算法,虽然不能保证最优,但简单易于实现,因此具有实际意义.

2.3 基本模型算法

如上文所述,iVCE for Memory是由各个节点逐步加入而构成的.每个节点在加入的时候,都能够通过基础设施获知在网络坐标系统中的位置并发现周围邻近的节点.网络嵌入方法能够有效支持上述过程的实现,网络坐标的确定可以通过GNP[11]的方法实现,邻近节点的发现则可以通过Meridian[16]来实现.

不妨用$COORD(A),A\in\{N_i\}$标识各个节点的网络坐标.同时,由于各个节点是先后加入到iVCE for Memory中的,可以把节点集合$\{N_i\}$表示为有序集$\langle N_i\rangle$,N_1代表第 1 个加入的节点,N_2代表第 2 个,依此类推.算法如下:

算法 1. 针对基本模型的贪婪算法.

输入:有序的节点集$\langle N_i\rangle$和每个节点的网络坐标$COORD(A),A\in\langle N_i\rangle$.各组直径上限$D$,各组大小上限$M$.

输出:节点子集的集合$\{G_i\},i\in[1,m]$,满足基本模型的条件.

过程:

对每个 i

节点N_i找到k个最接近的节点,其与N_i之间的欧几里德距离需要小于D.找到的节点集合用$\langle P_j\rangle$表示,其中$j\in[1,k]$,按其欧几里德距离排序.

如果k=0,则新建一个组,仅包含 1 个节点N_i.

否则

令 j=1 to k

如果P_j所在的组中的所有节点与N_i的欧几里德距离都小于D,且P_j所在的组的大小不到上

限 M,则

把N_i加入到P_j所在的组中;

跳出对 j 的循环.

否则

新建一个组,仅包含 1 个节点N_i.

算法 1 是符合分布式要求的算法,因为该算法可以在各个节点上分别执行.虽然不能保证最优,但其简单、有效.事实上,上述算法虽然存在不足,但大都是由模型的不合理性导致的,如下所述:

- 算法的解取决于各个节点加入的顺序,即受$\langle N_i\rangle$的顺序影响.举例来说,在图 1 中,如果有$\langle N_i\rangle=\langle A,B,C\rangle$,则结果是最优的$G_1=\{A,B\},G_2=\{C\}$;而如果$\langle N_i\rangle=\langle B,C,A\rangle$,结果就是$G_1=\{A\},G_2=\{B,C\}$.
- 各组的直径上限D是一个常数,D通常取一个典型的磁盘访问延迟.但对于实际系统来说,$\forall i\in[1,n]$,$DL(N_i)=D$这条假设显然是不成立的.如企业级磁盘的访问延迟远低于普通PC的磁盘延迟,可能出现A可以用B的远程内存,但B不能用A的远程内存的情况.如$DL(A)=10,DL(B)=5,ND(A,B)=8$,显然,$DL(A)>ND(A,B)$,而$DL(B)<ND(A,B)$.
- 基本模型中没有体现出节点的内存容量所带来的影响.假设 $DL(A)=5,ND(A,B)=ND(A,C)=3$,但节点 B 的内存容量大于 C,且 B 与 C 不在同一个组内,则 A 更应该加入 B 所在的组,因为 B 具有更强的潜在服务能力.而在基本模型中,B 与 C 的区别无法体现.
- 基本模型完全用网络坐标系统中的欧几里德距离估算节点之间的延迟,但事实上,任何网络嵌入方法都不能完全避免误差.而且两个节点之间的网络延迟随时可能发生变化,这一变化在基本模型中也没有体现出来.
- 基本模型中对系统的成长性考虑不够,特别是节点的离开,可能会影响到聚类的结果.如图 1 所示,假设最初的聚类结果是$G_1=\{A,B\},G_2=\{C\}$,如果节点A离开了,则结果自然变为$G_1=\{B\},G_2=\{C\}$.显然,这时把两个组进行合并形成$G_1=\{B,C\}$属于更优的结果.

考虑到基本模型具有不足,我们将提出改进的聚类模型来解决上述问题.

3 聚类模型与算法

3.1 力场-势能模型

为了更精确地建模,我们首先分析两个节点之间的关系.如上文所述,当 $ND(A,B)<DL(A)$时,A 能使用 B 的内存,或者说此时 $DL(A)–ND(A,B)>0$.那么,$DL(A)$和 $ND(A,B)$之间的差决定了它们的关系.当 $DL(A)–ND(A,B)>0$ 时,应该把它们归入一个组中,$DL(A)–ND(A,B)$越大,则把它们归入一个组的需要越强烈.这容易让我们联想到物理学中通过弹簧连接的两个质点,弹簧被拉得越长,产生的弹力就越大;如果弹簧被压缩,则产生相反方向的弹力.

我们从这个类比开始建模.对于 iVCE for Memory 中的两个节点,它们的关系可以用两个对应的质点之间的力来表示.如果 A 和 B 之间互相能使用对方提供的远程内存,则他们的质点之间产生拉力,否则产生推力.拉力主要来自于 A 和 B 之间的两根弹簧,在后文中我们还会引入万有引力.为阐述方便,我们在后文中不区分节点和相应的物理系统中的质点.

对于A和B来说,假设它们已经被固定在网络坐标系统的空间中,之间通过两根弹簧连接,如图 2 所示.A上连接的那一根弹簧被称为S_{AB},B上连接的那一根被称为S_{BA}.

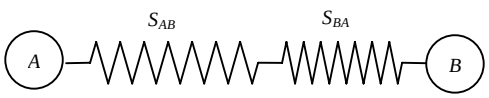


Fig.2 Two nodes (particles) and the springs between them

图 2 两个节点(质点)和它们之间的弹簧

每根弹簧都有同样的弹性系数s,弹簧的自然长度即为两个节点的磁盘访问延迟$DL(A)$和$DL(B)$.那么,从A来看,两根弹簧都会对它产生弹力,这个力表示为F_{AB},可以根据胡克定律进行计算:

$$F_{AB}=s(DL(B)-ND(A,B))+s(DL(A)-ND(A,B))=s(DL(A)+DL(B)-2\cdot ND(A,B)).$$

与物理系统稍有区别的是,当F_{AB}为正值时,它代表的是拉力;当F_{AB}为为负值时代表推力.

为了考虑A和B的内存容量带来的影响,我们引入万有引力的概念:当A和B足够接近时,它们之间将会产生万有引力U_{AB}.这里,“足够接近”的含义是

$$ND(A,B)<\min(DL(A),DL(B)).$$

于是,根据万有引力定律,U_{AB}的计算方法如下:

$$U_{AB}=\begin{cases} g\cdot\dfrac{M(A)\cdot M(B)}{(ND(A,B))^2}, & \text{当}ND(A,B)<\min(DL(A),DL(B))\text{时}\\ 0, & \text{否则}\end{cases},$$

其中,$M(A)$在物理学中是指质点 A 的质量,而在我们的模型中是指节点 A 的内存容量.g 是万有引力常数.

下面我们考虑由多个节点组成的系统.在这样的系统中,每两个节点之间都存在作用力,多个节点之间构成了一个力场.由于力是用矢量表示的,多个力可能会相互抵消,这显然与我们考虑的实际情况不符.因此,我们引入物理学中势能的概念,因为势能是标量,可以进行叠加.在弹力相互抵消的情况下,势能仍可能存在.

质点A和B之间具有的势能用$E(A,B)$表示,该势能是由A和B之间的弹性势能和万有引力势能组成的,分别用$E_S(A,B)$和$E_U(A,B)$表示.有时,我们还需要考虑一个组G内部的势能,用$E(G)$表示,它代表了组G内部的所有节点对之间的势能的叠加:

$$E(G)=\sum_{\forall A,B\in G}E(A,B)=\sum_{\forall A,B\in G}(E_S(A,B)+E_U(A,B)),$$

其中:

$$E_S(A,B)=\int_0^{|DL(A)-ND(A,B)|}s(DL(A)-ND(A,B))\,dx+\int_0^{|DL(B)-ND(A,B)|}s(DL(B)-ND(A,B))\,dx$$

$$=\frac{s}{2}[|DL(A)-ND(A,B)|\cdot(DL(A)-ND(A,B))+|DL(B)-ND(A,B)|\cdot(DL(B)-ND(A,B))],$$

$$E_U(A,B)=U_{AB}\cdot ND(A,B)=\begin{cases} g\cdot\dfrac{M(A)\cdot M(B)}{(ND(A,B))}, & \text{当}ND(A,B)<\min(DL(A),DL(B))\text{时}\\ 0, & \text{否则}\end{cases}.$$

显然,对于势能$E(A,B)$来说,有$E(A,B)=E(B,A)$.另外,对于一个单独的节点,势能是不存在的,因此有$E(A,A)=0$.上述势能的计算方法与物理学中基本类似,但有一点差别:物理学中的势能总是正值;而我们的模型中,弹性势能$E_S(A,B)$有可能是负值,因此,$E(A,B)$也有可能取负值.$E(A,B)$为正,代表A和B应该被分入一个组中;反之亦然.因此,谓词$Miss(A,B)$的定义变为

$$(A\in G_i)\wedge(B\in G_k)\wedge(i\neq k)\wedge E(A,B)>0.$$

相应地,基本模型中的限制和目标也需要变为:

限制 1. $\forall A,B(A\in G\wedge B\in G),E(A,B)>0$.

限制 2. $\forall i\in[1,m],S(G_i)\leq M$.

目标 1. 最小化 $Miss(A,B)$的发生概率.

目标 2. 最小化 $\sum_{i=1}^{m}E(G_i)$.

与基本模型相比,上述力场-势能模型更为精确地反映了实际系统的情况.事实上,虽然力场-势能模型是针对内存共享的问题而建立的,但也同样适用于对传输延迟较为敏感的分布式系统,如基于对等模式的视频流服务系统,为了保证服务质量,可以把相互之间具有较低延迟的节点进行聚类,并在此基础上对传输开销进行优化,等等.

当然,力场-势能模型也存在不足之处,如不能很好地反映系统的动态性.因此,我们提出了一种对力场-势能模型进行求解的聚类算法,并在该算法中对系统的动态性进行相应的处理.

3.2 聚类算法

首先,我们分析和改进节点加入的过程.假设节点A希望加入组G,而在G中存在节点B,符合$E(A,B)\leq 0$.我们称B这样的节点构成了A在G中的冲突集,用$Col(A,G)$表示.由于模型的限制 1,需要把G分成两个组G_1和G_2.G_1包含了A在G中的冲突集,G_2则包含A.对于所有其他满足$C\in G\wedge C\notin Col(A,G)$的节点,需要判断:

当 $E(A,C)<\sum_{\forall B\in Col(A,G)}E(B,C)$ 时,留在G_1中,否则在G_2中.如果有$Col(A,G)=\varnothing$,则$G_1=\varnothing$且$G_2=G\cup\{A\}$.

这时,只要满足 $S(G)<M$,A 都能够成功加入.

当G被分为G_1和G_2后,也可能出现$G_1=G$且$G_2=\{A\}$的情况,这时,我们称A无法被组G接受;反之,如果A被G接受,则必有$G_1\subset G$,在这种情况下,由于一部分原本在G中的节点没有留在G_1中,根据目标 1 和目标 2,可以尝试把G_1或G_2与其他组进行合并.另外,当某个节点主动离开iVCE for Memory时,也需要尝试把它所在的组进行合并.

我们还要考虑网络延迟的估计误差和波动的问题.为了使用较准确的网络延迟计算力场和势能,需要尽可能地对实际网络延迟进行测试,但测试又不能过多地影响网络的正常通信.我们设定两个节点在下面几种情况下需要进行实际延迟测试:

- 当一个节点加入 iVCE for Memory 系统时,它需要在内存共享虚拟共同体的基础设施的帮助下查找 k 个最邻近的节点.在这个查找的过程中需要进行实际的延迟测试.
- 当一个节点准备加入到一个组的时候,它需要向该组的管理者发出一个加入请求,由后者进行势能计算,从而决定是否允许它的加入.这个加入请求及其应答的过程可以看作是一次网络延迟测试.
- 当一个节点需要使用组内的内存资源时,根据我们在iVCE for Memory的前期工作[2]中的规定,用户节点需要事先向资源节点提出预约.这个预约消息及其应答可以看作是在实际使用资源前对网络延迟的测试和确认.

在测试的过程中可能会出现两个节点间的网络延迟无穷大的情况.由于我们在前文中已经假设了任意两个节点之间总是能直接通信,因此,无穷大的网络延迟即可认为是对方节点已经离开了 iVCE for Memory,从而需要进行合并组的尝试.

如果测试的节点间网络延迟不是无穷大,记为$ND_P(A,B)$,由于网络延迟估计的误差或延迟的波动可能会导致估计的网络延迟$ND_E(A,B)$与之不符.在这种情况下,我们考虑到远程内存的作用是为本地磁盘提供缓存.而如果缓存速度太慢,仍可以从本地磁盘获得数据,而并不会对系统性能造成太大的影响.因此,我们可以利用多次测试得到的$ND_P(A,B)$对A与B之间的网络延迟进行预测,以概率方式找到在某个置信度下的置信区间,取其网络延迟上限值$ND_{upper}(A,B)$,以此进行力场和势能计算.具体的算法如下:

算法 2. 力场-势能模型算法.

输入:有序的节点集$\langle N_i\rangle$和每个节点的网络坐标$COORD(A)$,$A\in\langle N_i\rangle$.各组直径上限D,各组大小上限M,网络延迟预测的置信度.

输出:节点子集的集合$\{G_i\}$,$i\in[1,m]$,满足力场-势能模型的条件.

过程:

对每个 i

循环

节点N_i找到第j个最邻近的节点P_j,记录它和P_j之间网络延迟的估计值$ND_E(N_i,P_j)$和实际测试值$ND_P(N_i,P_j)$.

节点N_i根据多次积累的$ND_P(N_i,P_j)$预测$ND(N_i,P_j)$的概率分布,并计算置信度C下的置信区间$(0,ND_{upper})$.

节点N_i计算N_i和P_j之间的势能$E(N_i,P_j)$.

当$E(N_i,P_j)>0$时保持循环.

对所有的邻近节点P_j按势能由高到低排序.

对于满足$P_j \in G_j$的每个组G_j

节点N_i向G_j的管理节点发送一个加入请求,后者计算组内各节点到N_i的势能.

如果节点N_i能够被组G_j所接受,则

根据需要把组G_j分为组G_{j1}和G_{j2};

跳出当前循环.

如果节点N_i无法加入到任何一个组内

则新建一个组,仅包含 1 个节点N_i.

如果在测试网络延迟的过程中发现组G_L中有节点离开

如果离开的节点是G_L的管理节点

则重新选择G_L的管理节点.

如果G_L发现满足情况$\exists G_S(\forall A,B \in G_L \cup G_S(E(A,B)>0) \wedge (S(G_L)+S(G_S) \le M))$

则合并组G_L和G_S.

上述算法也是分布式的,支持多个节点先后加入系统,并分别进行计算.值得说明的是,为了对网络延迟进行预测,我们假设网络延迟服从正态分布$N(\mu,\sigma^2)$.当两个节点之间的网络延迟还没有经过实际测试时,需要采用估计值$ND_E(A,B)$计算势能;否则,可以针对一系列测试的结果$ND_P(A,B)$,计算正态分布的参数μ和σ,然后采用区间估计的方式,计算出在置信度C下的置信区间$(0,ND_{upper})$,并使用$ND_{upper}(A,B)$对节点间的势能进行计算.

4 模拟和结果

4.1 模拟设置

为了对聚类的过程进行模拟,我们首先需要选取合适的网络拓扑和基于该拓扑的瞬时网络延迟.我们在模拟中使用了 3 种不同的网络拓扑,分别简称为 PlanetLab,DIMES 和 BRITE.

- PlanetLab 是 MIT 的 Jeremy Stribling 等人开展的一个名为 All Pairs Pings Data for PlanetLabs 的项目,用于测试 PlanetLabs 中多个节点之间的网络延迟.测试从 2003 年开始,几乎每天都在进行,持续至今.其中,时间上较晚的数据一般包括了更多遍及世界的节点的测试结果,但由于节点分布较广,网络延迟通常也比较高.为了获取更多对 iVCE for Memory 较有意义的网络延迟,我们选取了 2003 年 12 月 5 日的测试结果,其中包括了 270 个节点.
- DIMES也是一个对Internet进行实际测试的项目[17].测试从 2004 年 10 月至今都在进行.测试结果中包含了自治系统级的节点、自治系统级的网络连接、IP级的节点和网络连接的相关数据,我们主要利用了其公布的IP级网络连接的数据.DIMES的原始数据中包括了大量的节点,但由于模拟规模的限制,我们从中选取了 1 000 个经常出现的节点,并记录了节点间测量的网络延迟.
- BRITE是一个著名的网络拓扑生成器[18],它支持单层和双层等多种网络模型.我们主要采取BRITE中的自顶向下的生成方法,产生了一个双层的网络拓扑,该拓扑中包含了 10 个自治系统,其中各包括 300 个节点.为了贴近Internet的实际情况,我们选用了厚尾(heavy-tailed)分布的方式,并采用了Waxman模型.在最终的结果中,共计含 3 000 个节点和 6 020 条网络连接.

为使网络延迟符合第 3 节中假设的情况,我们采用了两种方法对其进行处理:1) 如果两个节点间无法直接联通,或没有延迟数据,则采用Dijstra算法生成一条最短路径,并以最短路径上的延迟之和作为这两个节点的实际延迟;2) 如果两个节点的延迟不对称,即$ND(A,B) \ne ND(B,A)$,则采用其平均值.为了模拟网络延迟的波动情况,我们假设网络延迟服从正态分布$N(\mu,\sigma^2)$,其中,μ取网络延迟的瞬时平均值,σ取μ的 1/10.正态分布的随机数采用Box-Muller转换公式产生.

我们还模拟了各个节点的内存容量和磁盘访问延迟等重要参数.内存容量总是 2 的整数次幂,服从

[256,4096]上的均匀分布,单位是 MB;磁盘访问延迟则服从(8,16)上的均匀分布,单位是 ms,则分别根据实际情况取值,受篇幅所限,这里不再赘述.

4.2 结果和分析

前文中已经定义了聚类的遗漏,即 $Miss(A,B)$的情况;这里再定义一种聚类错误的情况,即$(A\in G)\wedge(B\in G)\wedge E(A,B)\le 0$,其直观意义是指由于网络延迟的估计误差等原因造成的两个延迟过大的节点被归入一组的情况.显然,聚类的遗漏和错误都应该尽量避免.我们用聚类结果中发生的遗漏或错误的节点对的个数除以系统中节点的总数作为遗漏或错误的评估值,统计结果如图 3 所示.

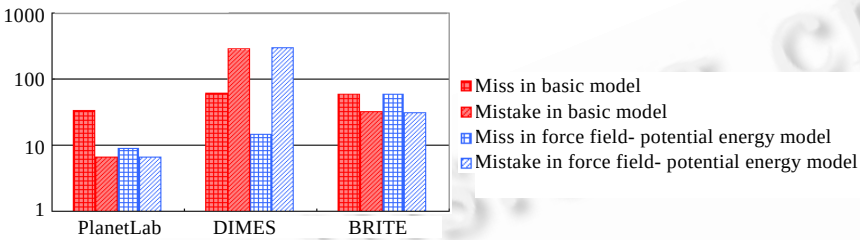


Fig.3 Miss and mistake in two clustering models

图 3 两种聚类模型的遗漏和错误

为使令差异较大的数值便于在一张图中进行比较,我们采用对数方式表示纵轴.由图 3 可以看出,力场-势能模型与基本模型相比,由于受网络延迟波动的影响,在聚类错误方面的改进不明显;但在聚类遗漏方面有一定的改进,特别是在 PlanetLab 和 DIMES 两组网络拓扑的模拟中,改进效果非常明显.由于纵轴是以对数方式增长的,故上述两种情况中聚类的遗漏几乎降低了一个数量级.

下面我们统计对每个节点而言,同处于一个组内且网络延迟较低的内存资源的总量.这一数值代表了该节点在理论上有可能使用的远程内存的最大值,虽然其他节点对内存也有占用,但理论最大值仍然具有参考意义.统计结果如图 4 所示.

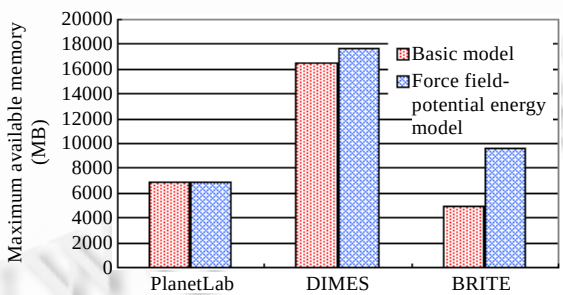


Fig.4 Maximum available memory in two clustering models

图 4 两种模型的最大可用内存

不难发现,力场-势能模型在 3 种网络拓扑中均优于基本模型,但在 BRITE 中的优势最为明显.联系到图 3 中的结论,我们认为,聚类的遗漏和错误虽然应该尽量避免,但与节点可用的远程内存关系并不大.另外,DIMES 拓扑的节点总数比 BRITE 少,但各节点的最大可用内存却比前者高,这说明 DIMES 拓扑中的网络延迟较低,更适合 iVCE for Memory 的应用.

为了发现力场-势能模型与基本模型相比的具体优势,我们对 3 种网络拓扑的聚类结果分别进行了统计,并画出各种大小的组的个数.如在 DIMES 拓扑中,力场-势能模型的聚类结果包括了 68 个仅由 1 个节点构成的组,则在横坐标 1 处画一个高度为 68 的条形.统计结果如图 5 所示.

由图 5 可以看出,力场-势能模型与基本模型相比,其聚类结果里中等大小的组相对较多,而过大的组和过小

的组通常都比较少.如在PlanetLab拓扑中,力场-势能模型生成了更多大小在8~18之间的中等组.不难理解,过大的组容易造成聚类错误,而过小的组容易造成聚类遗漏,中等大小的组往往是比较合适的.我们在力场-势能模型的算法中,不断地把过大的组进行分裂,又把过小的组尽量合并,形成了较多中等大小的组,从而使结果较优.

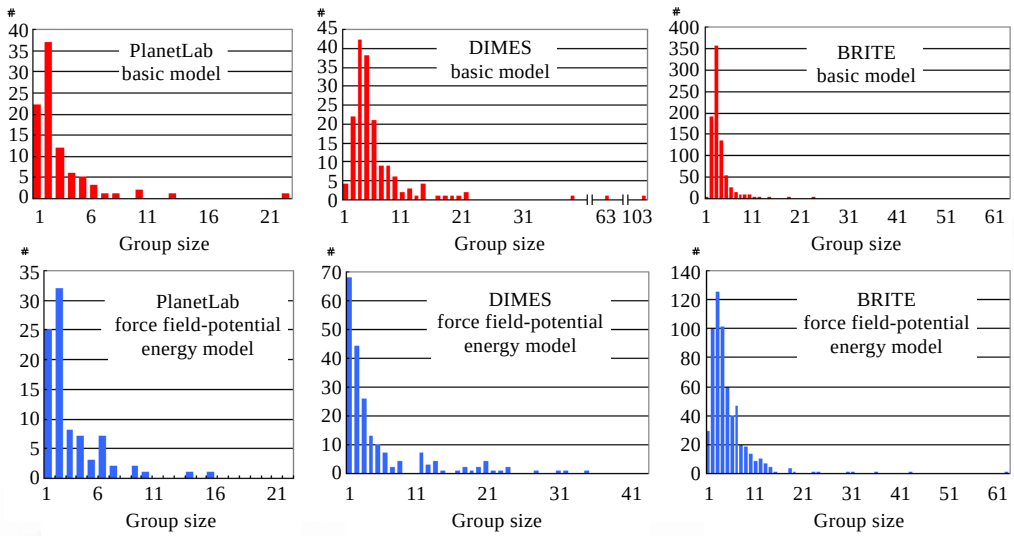


Fig.5 Relationship of group size/count in two clustering models

图 5 两种模型的组大小/个数关系

5 结束语

面向互联网的虚拟计算环境(iVCE)致力于在资源共享和综合利用的基础理论和根本机理上取得新突破,并提出了资源聚合与协同的核心机制.iVCE for Memory 作为 iVCE 的实例,试图利用 iVCE 的体系结构与核心机制,有效共享目前广域网上大量分布的空闲内存资源.本文以 iVCE for Memory 为背景,分析了基于聚类的资源聚合方式,并参考物理学中的力场和势能理论,采用两种模型和算法分别实现了 iVCE for Memory 的资源聚合机制.模拟结果表明,基于聚类的资源聚合能够有效降低问题规模,对虚拟计算环境的研究具有重要的价值.

References:

[1] Lu XC, Wang HM, Wang J. Internet-Based virtual computing environment (iVCE): Concepts and architecture. Science in China (Series E), 2006,36(10):1081−1099 (in Chinese with English abstract).

[2] Chu R, Xiao N, Zhuang YZ, Liu YH, Lu XC. A distributed paging RAM grid system for wide-area memory sharing. In: Proc. of the 20th Int'l Parallel and Distributed Processing Symp. Toronto: X-CD Technologies Inc., 2006. 88.

[3] Jiang S, Davis K, Petrini F, Ding XN, Zhang XD. A locality-aware cooperative cache management protocol to improve network file system performance. In: Proc. of the 26th IEEE Int'l Conf. on Distributed Computing Systems (ICDCS 2006). Washington: IEEE Computer Society, 2006. 42. http://dblp.uni-trier.de/rec/bibtex/conf/icdcs/2006

[4] Hines MR, Lewandowski M, Wang J, Gopalan K. Anemone: Transparently harnessing cluster-wide memory. In: Proc. of the Int'l Symp. on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2006). 2006. http://www.cs.binghamton.edu/~jianwang/resume.html

[5] Oleszkiewicz J, Xiao L, Liu YH. Parallel network RAM: Effectively utilizing global cluster memory for large data-intensive parallel programs. In: Proc. of the 2004 Int'l Conf. on Parallel Processing. Los Alamitos: IEEE Computer Society, 2004. 353−360. http://dblp.uni-trier.de/rec/bibtex/conf/icpp/2004

[6] Sarkar P, Hartman J. Efficient cooperative caching using hints. In: Proc. of the ACM Symp. on Operating Systems Design and Implementation (OSDI). New York: ACM Press, 1996. 35−46. http://portal.acm.org/citation.cfm?id=238721.238741

[7] Markatos EP, Dramitinos G. Implementation of a reliable remote memory pager. In: Proc. of the USENIX Annual Technical Conf. Monterey: USENIX Association, 1996. 177−190. http://citeseer.ist.psu.edu/19174.html

[8] Feeley MJ, Morgan WE, Pighin FH, Karlin AR, Levy HM, Thekkath CA. Implementing global memory management in a workstation cluster. In: Proc. of the Symp. on Operating Systems Principles. New York: ACM Press, 1995. 201−212. http://citeseer.ist.psu.edu/feeley95implementing.html

[9] Dahlin MD, Wang RY, Anderson TE, Patterson DA. Cooperative caching: Using remote client memory to improve file system performance. In: Proc. of the 1st Symp. on Operating Systems Design and Implementation. 1994. 267−280. http://www.cs.utah.edu/~lepreau/osdi94/index.html

[10] Buyya R. High Performance Cluster Computing: Architecture and Systems, Vol.1. English Reprint Edition, Beijing: Posts & Telecommunications Press, 2002. 383−385.

[11] Ng TSE, Zhang H. Predicting Internet network distance with coordinates-based approaches. In: Lee D, Orda A, eds. Proc. of the INFOCOM 2002, 21st Annual Joint Conf. of the IEEE Computer and Communications Societies. New York: IEEE Communication Society, 2002. 170−179.

[12] Costa M, Castro M, Rowstron A, Key P. PIC: Practical Internet coordinates for distance estimation. In: Proc. of the 24th Int'l Conf. on Distributed Computing Systems (ICDCS 2004). Washington: IEEE Computer Society, 2004. 178−187. http://dblp.uni-trier.de/rec/bibtex/conf/icdcs/2004

[13] Lim H, Hou JC, Choi CH. Constructing Internet coordinate system based on delay measurement In: Proc. of the 3rd ACM SIGCOMM Conf. on Internet Measurement. New York: ACM Press, 2003. 129−142. http://portal.acm.org/citation.cfm?id=948205.948222

[14] Dabek F, Cox R, Kaashoek F, Morris R. Vivaldi: A decentralized network coordinate system. SIGCOMM Computer Communication Review, 2004,34(4):15−26.

[15] Shavitt Y, Tankel T. Big-Bang simulation for embedding network distances in Euclidean space. IEEE/ACM Trans. on Network, 2004,12(6):993−1006.

[16] Wong B, Slivkins A, Sirer EG. Meridian: A lightweight network location service without virtual coordinates. In: Proc. of the 2005 Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications. New York: ACM Press, 2005. 85−96. http://portal.acm.org/citation.cfm?id=1080091.1080103

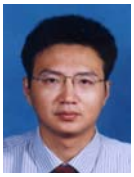
[17] Shavitt Y, Shir E. DIMES: Let the Internet measure itself. SIGCOMM Computer Communication Review, 2005,35(5):71−74.

[18] Medina A, Lakhina A, Matta I, Byers J. BRITE: An approach to universal topology generation. In: Proc. of the Int'l Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems—MASCOTS 2001. Washington: IEEE Computer Society, 2001. http://portal.acm.org/citation.cfm?id=882563

附中文参考文献:

[1] 卢锡城,王怀民,王戟.虚拟计算环境 iVCE:概念与体系结构.中国科学(E 辑),2006,36(10):1081−1099.

褚瑞(1979－),男,陕西西安人,博士生,主要研究领域为网格计算,高性能并行分布处理技术.



肖侬(1969－),男,博士,教授,博士生导师,主要研究领域为网格计算,数据网格,高性能并行分布处理技术.

卢锡城(1946－),男,教授,博士生导师,中国工程院院士,CCF 高级会员,主要研究领域为 MPP 实现技术,分布处理技术,网络技术.