一个基于层次结构的DSM模型*)

A Hierarchical DSM Model

李 翼 郭建新 陈贵海 谢 立

(软件新技术国家重点实验室 南京大学计算机系 南京210093)

Abstract We present a hierarchical DSM scheme named GDSM. In our solution, sub-tasks are to be mapped to groups, which apply different techniques based on the distinct characteristics of inter-group and intra-group data sharing. A serial of alternatives are used with respect to the intra-group and inter-group property of this mechanism: (1)Consistency model: release consistency vs scope consistency; (2) Coherence protocol: multiple-writer and write-update protocol vs. single-writer and write-invalidate protocol; (3)Granularity: fine-grain vs. coarse-grain. Our novel strategy to combine grouping with consistency, data conversion and granularity switch cuts down the overhead of consistency maintenance, increases the parallelism between groups and thus promotes system performance. We also put forward the architecture of Protocol Engine implementing the GDSM scheme, the engine that achieves message-forwarding transparency, clear hierarchical structure and encapsulation of group.

Keywords DSM, Group, Consistency model, Granularity, Protocol Engine

一、引言

分布式共享内存(Distributed Shared Memory, DSM)是并行处理中的一种关键技术。它为程序员提供了一个逻辑上统一的虚拟地址空间,任何一个处理机都可以对这一地址空间直接进行读写访问。其中一致性模型、一致性协议、粒度是影响DSM性能的重要因素。

DSM中物理内存分布在多个节点上,数据一般采用复制的方式来实现共享内存的抽象,因此如何在保证内存一致性前提下尽量提高效率成为DSM的一个重要研究内容。一致性模型[5]本质上是内存系统与并行程序之间关于内存访问的协议。严格一致性模型包括原子一致性、顺序一致性等,严格的一致性模型不利于开发程序的并行性,因此出现了很多松懈一致性模型,包括弱一致性、释放一致性、域一致性(Scope Consistency)和单项一致性(Entry Consistency)等。

一致性模型可以用不同程度的惰性协议来实现一致性维护操作和数据修改的传播,这样就需要区分单写和多写协议、写更新和写无效。单写协议允许多个读者同时存取某个页面,但只有一个写者拥有修改权,单写协议易于实现,但通信代价较大,多写协议同时允许多个进程对共享数据拥有写权限。这种方式减少了假共享和对全局带宽的要求,但需要额外的处理代价。写更新和写无效的区别在于系统出现不一致时的处理方式。写更新要求在某处被修改的数据拷贝立即在其他节点得到更新,而写无效仅仅使其他节点的数据拷贝无效,因此,在写更新协议中,如果该节点在数据修改前拥有该数据,则修改后该数据依然有效。在写无效中,远处的节点必须从修改节点处获得更新过的数据。

共享数据的粒度也是一个重要的问题。共享粒度指一致性维护的单位,即系统在数据存取故障时复制的数据大小。它不仅包括一次读数据要传输多少数据,还包括一次写数据会影响多少数据的有效性。粒度是

*)本文得到国家自然科学基金(NSF#69803005)的资助。

参考文献

1 Reenskaug T. Working with Objects (The OOram Software Engineering). Manning Greenwich, 1996
2 邵维忠,杨芙清. 面向对象的系统分析. 清华大学出版社,广西科学技术出版社,1998年12月
3 姜鸿飞,全炳哲,金淳兆. 面向对象开发方法的最新进展. 计算机科学,1998,25(2)
4 周之英,王淳. IDEF4与面向对象设计方法. 计算机科学,1995,22(2)
5 李师贤,李文军,周晓聪. 面向对象程序设计基础. 高等教育出版社,1998
6 袁晓东,陈家骏,郑国梁. 基于角色分类的子类型关系. 计算机研究与发展,1997(11)

影响应用程序运行效率的重要因素。大粒度减少了通信的次数，但会导致假共享和产生内存碎片；小粒度减少了假共享，却导致经常性的访问失败、映射表过大和网络通讯负载重等问题，最优的粒度取决于维护内存致性的代价和应用的存取特点。在以往的系统中，如果用硬件来处理共享存取冲突，往往以缓冲区的一行作为共享粒度；如果用软件方式处理，则多以页为单位。有的系统采用了可变粒度，但往往在很大程度上依赖于应用程序的信息，增加了编程和编译的难度。

随着网络的迅速发展，存在着在异构环境下实现DSM的可能性。统一异构环境于DSM不仅可以使多种机器协作工作，而且可以增强对单个问题的计算能力。

一致性模型、协议、共享粒度是DSM中的重要问题，以往的工作往往单纯集中在其中的一个或两个方面，缺乏系统的分析和综合。本文提出了一个基于层次结构的DSM设计，充分考虑了大规模并行计算的特点，通过分组的方式，采用不同的一致性模型开发分任务间的并行性，同时在组内和组间采用不同的共享粒度，具有很高的灵活性。

本文在大规模并行处理中引入组的概念，分析其特点并由此得出在组内与组间的不同的一致性模型、协议、粒度策略及异构的可能性，接着以协议引擎为中心给出了GDSM的详细设计方案，最后讨论相关工作并总结全文。

二、GDSM模型分析

2.1 大规模并行计算与组

2.1.1 组的提出 在大规模并行计算中，特别是在一个大规模网络计算环境下，如气象预报等，一个大任务常常被划分为若干个分任务在一个平台上运行，每个分任务又分成多个小任务运行于各自的节点上，这正适合于分组的概念。

组，是指在物理上的邻近范围内运行，具有共同特征或联合性质、关系紧密的进程的集合。

整个任务运行于通过DSM构造的虚拟全局内存的抽象层。将关系密切的小任务以组的形式组织，就是将小任务与分任务在概念和结构上进行了划分，这些分任务构成不同的逻辑组，运行于位置靠近的机器组。因此，我们把物理上的机器组与分任务的层次结构相结合，同组机器执行关系密切的各个子任务组成的分任务，从而将逻辑上的任务组与物理上的机器组有机地结合起来。

2.1.2 组特性 任务被分成多个大粒度的分任务。在程序意义上它们具有不同的功能，因此分任务之间的数据共享不密切，往往是阶段性使用数据。即对于同一数据，组之间的数据使用是间隔的，或者在同一时间段较少使用，往往是某一个阶段主要由一个分任务访问某些数据，而其它分任务较少访问；这个阶段过后再由其它分任务去访问该数据。因此，我们将一个分任务对应于一个组，组之间通常只要阶段性地维护组间的一致性，同时保证一个阶段内组内的数据访问的独立性，尽量避免组外的数据使用，充分发挥并行性。而且，在组内节点完成对数据的阶段性访问后，所有修改的数据可以一起更新，经过数据转换（如果必要）传到要访问该数据的组，减少系统和网络的开销。

在组内，分任务又被分成多个子任务，由于程序在时间和空间上的局部性，组内各节点上的子任务往往是处理相关的数据而密切协作以完成某种功能。因此它们在资源使用上有相似性，数据使用模式相关，耦合度高，数据共享紧密，存取频繁。

组这一概念的提出，为一致性模型、一致性协议和粒度在应用的实现上提供了较好的灵活性，同时为在异构环境下的DSM提供了可能性，但是，层次结构也带来了新的问题，如组的管理，组内组间的粒度转换等。这些问题将在第三部分详细讨论。

2.2 GDSM中的关键问题分析

2.2.1 一致性模型分析 一致性模型描述了应用程序如何使用共享内存。松懈一致性模型（Relaxed Consistency Model）定义了一组充分的条件或规则，如果应用程序遵守这些规则，系统就可以保证顺序的一致性。松懈一致性的目标是为系统在内存操作上提供一定程度的灵活性，减少数据访问的时延，提高系统性能。

在一个大规模的系统下实现DSM，仅采用单一的一致性模型有诸多不足。首先，系统中结点数量很多，不仅导致了页面管理的复杂，而且也延长了对共享页面的存取时间，如页面的定位、页面的状态修改等。其次，由于程序运行和分布的局部性，一些结点经常访问的页面通常也会集簇在一起。最后，不同的一致性模型维护一致性的方式不同，单一的一致性模型只考虑了数据一致性在程序运行的什么阶段得以保证，而没有考虑在什么范围内的结点的数据是一致的，这样无法刻画一个大系统中不同部分的存取模式，多一致性模型可以有效地解决这一不足，带来了很大的灵活性。而利用组，采用多重一致性模型，局部的一致性管理方法只影响管理的一部分，而全局的管理方法则负责维护整个系统的一致性。

一致性模型是DSM中决定应用性能的最重要的因素，考察组内与组外数据共享的特点，我们可以确定各自适合的一致性模型。

在组内，各节点上的子任务密切协作，处理相关的数据，因此它们在资源使用上有相似性。在这种情况下，一方面较严格的一致性模型比较符合子任务之间的特点，并且考虑到易编程性；另一方面较松懈一致性

模型有利于提高性能，折衷考虑，在此采用了惰性释放一致性模型(LRC)，在惰性释放一致性中，共享数据的存取操作被分成获取、释放和非同步操作。在释放操作中，不执行任何真正的数据一致性维护操作，而是延迟到其他处理机执行下一个获取操作时。

在组间，数据共享下密切。组在程序意义上具有不同的功能，组之间通常只需阶段性地维护组间的一致性。数据更新通常是阶段性进行，因此，松懈一致性模型更适合组间的数据共享方式。考虑到不同组之间对不同共享数据的并发使用情况，兼顾到编程的复杂性问题，域一致性模型[5]是一种较合适的模型，每一共享数据都与同步变量相联系，以增加组间的并行性，它既保证了单项一致性(EC)所具有的并行性能，又避免了显式地将数据与同步变量的绑定，从而降低了程序的复杂度。

2.2.2 一致性协议 组一致性将组内与组外的一致性分离。由于组内与组外的差别，因此组内与组外所采用的一致性协议上也有所不同。在一致性协议的选择中我们参考了其他文章关于一致性模型和协议的关系的论述[5,6]。

在组内，由于节点数目有限，管理和通信开销小，且各个节点之间数据共享密切，往往是小粒度的协作；同时考虑到组内采用的是惰性释放一致性模型，因此采取多写、写更新协议，有效地减少了假共享和数据请求时间；而在组外，各个组之间耦合度低，数据共享往往是阶段性的；而且组之间可能是异构的，数据转换的工作量大；组间通信代价较高，因此采用单写、写无效协议，同时将组内对共享数据的修改集中在一起发送，减少消息发送次数并提高通信的效率。

2.2.3 粒度策略 以往的 DSM 往往整个系统使用相同的粒度，没有考虑粒度与一致性模型的关系。这种做法虽然可以利用系统自身的软硬件，但缺少灵活性，也没有充分利用一致性模型的特点。最近的工作[3,4,9]已经开始讨论多粒度 DSM，并开始重视一致性模型与粒度的关系。

在异构下用于大规模并行任务的 DSM 中的粒度选择要考虑到组内与组间的特点和所采用的一致性模型，又要考虑到可能存在的异构性。

小粒度可以灵活地实现数据共享，减少数据冲突，防止假共享、数据颠簸和内存碎片，因此似乎组间之间的粒度小一些较好。但这种分析只考虑了一般情况，而忽视在任务划分的前提下，组内组间数据使用的特点。此时主要问题不是减少假共享，而是减少通信数据量和通信频率。考虑到组间在不同计算阶段之间数据交换量较大的特点，采用小粒度，将增大组间的通信频率。同时，由于组间采用的是较弱的一致性模型，大粒度更有利。因此，我们在组间采用相对大的共享粒度。在组内，由于数据耦合度大，数据粒度不宜大，而且网络性能较好，相比之下，虽然相对增加了访问下中的概率和数据传送次数，但对于小粒度减少了假共享，提高了于任务的效率

总之，当应用的存取模式表现出局部性(即在组内)时，使用小粒度；当表现出全局性(即在组间)时，使用大粒度。这样对应用程序的信息下必过多了解，减低了编程人员的难度，同时保证了灵活性和高效性。

2.2.4 异构性 异构机器间的高转换代价往往使得异构 DSM 难以实用化。我们利用大规模并行计算的特点，采用层次结构并结合粒度选择策略可以有效地减少异构带来的负载。

基本思想是开发组内数据使用的局部性，利用组间的数据交换的特点，尽量使得异构组间的数据转换集中进行，减少不必要的系统开销，由于分任务间往往阶段性地使用数据，数据共享下密切，因此使得数据转换的工作可以尽量集中在每个阶段结束时进行，有效地提高了转换效率，减少了转换代价。

2.3 GDSM 中的策略结合

2.3.1 GDSM 概述 根据以上分析，我们设计了一个基于层次结构的 DSM 模型 GDSM。它是一个支持多粒度和多一致性模型的 DSM 模型。其中一致性模型、一致性协议、粒度的选择之间是相互作用的，并且最终由大规模并行计算中组的特点决定。在 GDSM 中，数据的一致性定义在两个层次上。一层是组间，一层是组内。在这种层次结构中，如果数据在两个层次中都维护好，在整个系统中数据就可以保持一致性。一致性协议是由组内和组间数据共享的特点和所采用的一致性模型决定的。同时，此模型不把异构机器分在相同的组中，为异构环境的分布式共享内存提供了一种可能的机制。组内采用惰性释放一致性模型，采用多写、写更新协议，数据共享采用小粒度；组间采用域一致性模型，采用单写、写无效协议，采用大粒度，组间的数据转换等工作只是阶段性地集中进行以减少异构带来的开销。

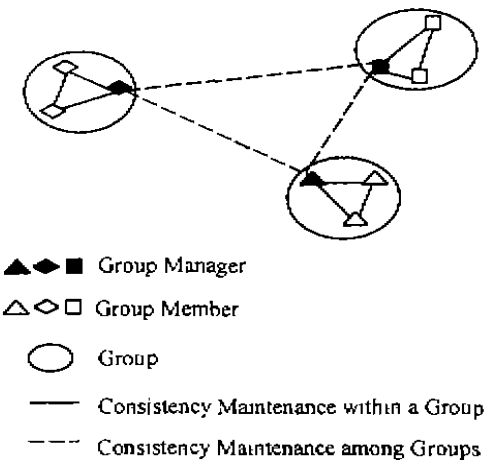

图1 GDSM 系统示意图

图1描述了GDSM的示意图。整个系统被分成多个组，在图中用椭圆表示，每个组中包含若干个同构的节点，在图中用椭圆内的相同形状的小几何体表示。其中带阴影的小几何体表示该组的组长节点（这将在下一部分详细论述），椭圆之间的虚线表示组间的松懈一致性维护，这些虚线的两端连接着两个组的组长节点，而椭圆内部的小几何体之间的实线表示组内节点之间的较强的一致性维护。

2.3.2 GDSM的优点 灵活性：组间与组内依据各自的不同特征可选用不同的一致性模型、协议和粒度；可扩展性：通过两层结构和以组的方式组织系统，易于扩充；高效性：采用组方式管理，对于组内运行的子任务，联系紧密而且数据共享比较集中，数据大都可以在组内得到，避免了在整个系统内进行数据共享的代价。组内的数据局部性得到充分开发，同时保证了组间的并行性，从而获得高效性。

三、GDSM的设计

3.1 GDSM的系统流程

组概念的提出为一致性模型、协议和粒度的选择提供了高度的灵活性，从而提高了系统的性能，但层次结构也带来了新的问题，即组内与组间在模型、协议和粒度上的转换的复杂性。我们通过组长和组员来实现它。

系统中每个组有两类节点：组员和组长，组员只负责执行本地的子任务，与组内其他成员共享数据并维护与本地数据相关的一致性，凡涉及组外的数据共享都向组长请求。组长除了完成与普通组员一样的功能外，还负责组内成员管理，组内任务分配、组间数据转换和粒度转换、同步处理等。

3.1.1 一致性模型的维护与一致性协议 系统开始时，各组的组长节点执行组间的同步操作，从其他组得到数据，将任务分配到组内成员，并在执行过程中维护组内的一致性。当组内的所有节点将小任务执行完毕后，通知组长节点，并执行一个由组长节点管理的隐式的栅（barrier）操作，用以同步组内所有节点。各节点对数据的修改都传到组长节点处，在整个过程中，不论任务的处理还是数据共享的控制，组长都起到发散和回收的作用。对外界看来，一个组似乎只有组长一个节点，屏蔽了组内其他成员，并与其他组长进行数据交换，维护组间一致性。组长可以执行组间的同步操作，更新其他组的数据，或从其他组得到数据，然后进入下一阶段的任务处理。

3.1.2 粒度转换 当某个组完成阶段性计算并由组长进行组间一致性维护时，各个节点将数据修改都传到组长处，由组长汇总，进行从小粒度到大粒度的转换，而当接收到其他组长的数据时，由本地组长进行从大粒度到小粒度的转换。

3.1.3 异构性 当一个组的分任务阶段性完成，组长与其他组长进行一致性维护时，仅负责将相应数据转换成中间格式并发给其他组长，接收者收到数据后根据发送者的类型将中间格式的数据转换成本地格式。此处由主要接收方负责数据转换的原因是发送者可能同时向多个组发送数据，由接收方进行数据转换可以并行进行，同时防止发送方成为系统的瓶颈。

3.2 GDSM的设计

3.2.1 体系结构 DSM模型分为三层（图2）。APPLICATION层是程序员与管理层联接的桥梁，为应用程序提供友好的DSM界面；MANAGEMENT/PROTOCOL层负责一致性模型和协议的维护及组管理，并为上层数据请求提供服务；COMMUNICATION层完成底层的通信，实现存储结构布局和粒度的变化，并根据上层的要求进行内存数据定位和访问。

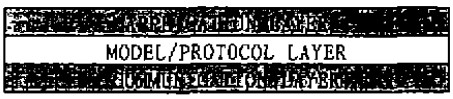

图2 DSM一般的体系结构

3.2.2 协议引擎 为了实现组一致性、共享的多粒度和数据转换，我们设计了基于组的协议引擎。它跨越Model/Protocol层和Communication层。协议引擎的层次结构与GDSM的要求一致。GDSM的结构见图3。根据节点在组内的地位，我们设计的协议引擎分为两种：组长引擎和组员引擎。这样分类的根据是基于组的一致性、共享粒度和数据转换的特点。

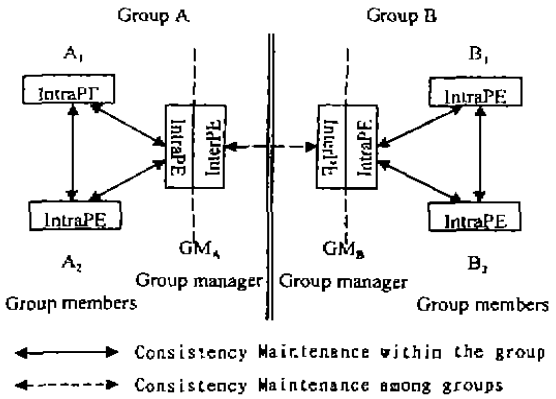

图3 GDSM系统结构图

①组员结点的协议引擎，组员结点方运行的是Intra-group Protocol Engine（IntraPE）。IntraPE位于体

系结构的 Model/Protocol Layer 和 Communication Layer(见图4),由本地客户(Local Client,LC),远程客户(Remote Clinet,RC)、服务方(Server)和 Intra-group Message Agent(IntraMA)组成。

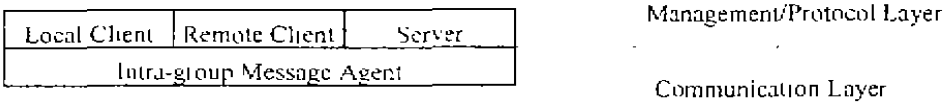

图4 组内协议引擎

本地客户(LC)负责维护复本的一致性,并实现客户方的数据请求协议。LC 运行在发生存取失败的机器上。LC 在发现缺块错误时向 IntraMA 发送数据请求,获得数据副本后修改相应状态。在此(处理缺块错误)期间组内的互斥通过一个共享锁实现。

远程客户(RC)负责在客户方执行块无效操作,它运行在存放有某块数据的拷贝的机器上。当收到块无效的请求(从本组的其他成员或从组外发到组长处的)时,RC 将本地块置为无效。RC 还负责执行更新操作。当本地拥有读权限的拷贝块被更改后,它必须被更新,RC 复制此块,对它进行更新并通知该块的属主节点。

服务方运行于某块的属主页所在机器上,负责服务器方的块拷贝和释放处理。在块的释放过程中,所有复制请求将被放在队列中,直到释放结束。

Message Agent(MA)是实现基于组的协议引擎的核心,有两种类型:Intra-group Message Agent(IntraMA)和 Inter-group Message Agent(InterMA)。前者运行在组员节点,两者都运行在组长结点,MA 实现从源到目的地的透明消息发送。协议引擎发出和接收的消息都要经过 IntraMA 的处理。Intra-group Message Agent 接到上层的消息时,它首先分析消息的目的地址,如果消息的方向是组内则直接将消息发送至目的地址的 IntraMA;否则,将消息转向组长结点上 InterPE 的 IntraMA。这种设计方便了一致性维护、粒度转换和数据转换。消息传递的过程见图5。

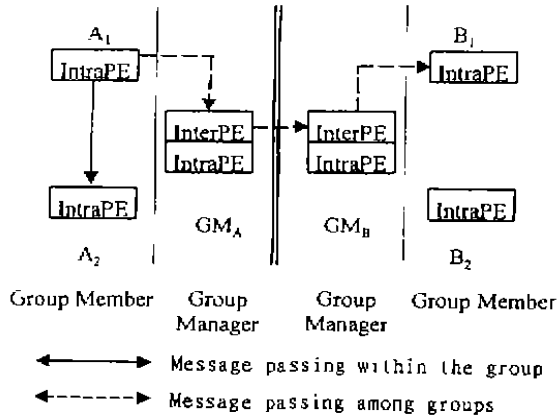

图5 消息传递过程

②组长结点的协议引擎。它是 IntraPE 和 InterPE 的有机组合,组长结点的 IntraPE 与组员结点的相同。所不同的是,组长结点还运行有 InterPE,它的结构见图6。

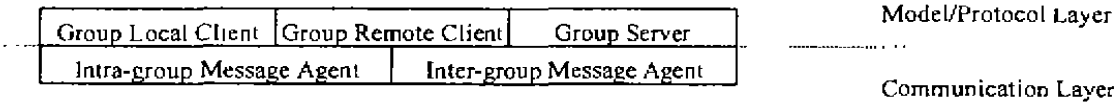

图6 组间协议引擎

组长结点是组在系统中的代表,对于整个系统来说某个组的面貌就是该组组长结点表现出来的特性的总和。InterPE 用于维护组间的一致性协议和组内与组间的粒度转换,由 Group Local Client、Group Remote Client、Group Server、IntraMA 和 InterMA 组成。Group Local Clinet(GLC)在子任务的阶段性时刻根据全局的组信息表与相关组的组长进行一致性协议维护。Group Remote Client(GRC)负责接收组外发来的命令,在本组内执行块无效操作,当收到外部发来的页无效的请求时,GRC 将请求分解成对块的无效请求,并搜索本组的局部块表,查找到相应的本组节点,命令它们将本地块置为无效,GRC 还负责执行更新操作、过程与前者类似,同时 GRC 告知发送该消息的组长。Group Server 机器上,负责将服务器方的块拷贝和释放处理向相应的其他组传播。

组长存放有全局组信息表,用于与系统中的其它组长进行通信。在每个阶段结束时,组长将在组内进行数据的收集和转换,并与相关组长进行同步。组长还存有组信息表,用于组内通信,需要考虑某些小概率事件,即在某个阶段中间,组中的成员可能需要访问组外

的共享数据，这时我们的处理是满足该成员的要求：组内结点发往组外的 Message，会被自己的 IntraMA 转发至组长结点上 InterPE 的 IntraMA，组长结点再将它向 InterMA 转发，由 InterMA 与目的结点的组长联系，由对方的 InterMA 接收消息，进行粒度转换和数据转换后传送给自己的 IntraMA，再转发给目的结点的 IntraMA。这样，一个某一阶段中间的组外的消息传递过程完成，见图5中虚线表示的消息传递。

3.2.3 优点 消息转发的透明性：Message Agent 的设计实现了引擎构件协议消息转发的透明性。有了 MA 以后上层构件的设计不必关心通信的细节。

清晰的层次结构：两组级协议引擎的设计使整个系统成为良好的层次结构，各层的协议引擎负责本层的一致性维护，具有良好的可扩展性。

组的封装性：组间一致性协议的实现只是组内一致性协议的实现在更高级别上的表现。对外部而言，组的特征仅由组长表现出来。

四、相关工作

已有的 DSM 系统包括：Dash[3][7]，TreadMark[10]，Munin[11]，HDSM[4]，MGS[3]，Millipage[14] 等，各有特点。

在组方面，PVM 提供了组的概念，可以以组的方式对进程进行组织，使用组播通信，但不具备组的数据共享功能。DASH 的实现体现了群的概念，但它不是以任务为单位，不能充分体现计算任务的特点，也没有提供较弱的一致性。

Munin 提供了多种一致性模型的选择，但不限定在多大范围内数据是一致的。MGS 在 SMP 之间采用了 RC，在 SSMP 之内采用了更严格的一致性模型。

在粒度方面，Tamir[9]提出了基于页和块的多层映射和一致性管理；MGS 将硬件缓冲一致性和软件 DSM 相结合，努力开发数据的局部性，但需要硬件支持。Millipage 实现了基于页的可变粒度和顺序一致性协议，但受缓冲区大小等因素的影响较大，并行度不高。

结论 今后的工作主要有：模型的实现。其中组长结点的协议引擎的实现是关键；组管理的改进。现有模型的组管理是集中进行的，组长在其中担当了桥梁的角色，可能成为系统的瓶颈。改成分布式管理可能会有助于提高系统的性能和可靠性；通用化。适用于本模型的计算任务必须能够分解为关系松散的分任务，因此不是所有任务都适于本模型。如何使本模型通用化是今后的工作之一。

参 考 文 献

1 Iftode L, Singh J P. Shared Virtual Memory Progress and Challenges. In Proc. of the IEEE Spring 1999

2 Li Q, Ji H, Xie L. Group Consistency Model Which Separates the Intra-group Consistency Maintenance from the Inter-group Consistency Maintenance in Large Scale DSM Systems. ACM Operation Systems Review, 1997, 31(2)

3 Yeung D, Kubiatowicz J, Agarwal A. MGS A Multigrain Shared Memory System. In Proc. of the 23rd Annual Symposium on Computer Architecture. May 1996

4 Zhou S, et al. Heterogeneous Distributed Shared Memory: An Exprimental Study IEEE Trans. on Parallel and Distributed Computing, 1992(May)

5 Zhou Y, et al. Relaxed Consistency and Coherence Granularity in DSM Systems: A Performance Evaluation. In Proc of the 6th ACM Symposium on Principles and Practice of Parallel Programming

6 Shi W, Hu W, Tang Z. An Interaction of Coherence Protocols and Memory Consistency Models in DSM Systems. ACM Operating Systems Review, 1997(Oct.)

7 Mosberger D. Memory Consistency Models. ACM Operating Systems Review, 1993(Jan.)

8 Iftode L, Singh J P, Li K. Scope Consistency: A Bridge between Release Consistency and Entry Consistency. Theory of Computing Systems, 1998, 31: 451～473

9 Tamir Y, Janakiraman G. Hierarchical Coherency Management for Shared Virtual Memory Multicomputers. Journal of Parallel and Distributed Computing, 1992, 15(8), 408～419

10 Amza C, et al. TreadMarks: Shared Memory Computing on Networks of Workstations. IEEE Computer, 1996 (Feb.)

11 Lenoski D, et al. The DASH Prototype: Implementation and Performance. ICPP'92II, pp92～103

12 Mitzberg B, Lo V. Distributed Shared Memory, A Survey of Issues and Algorithms. IEEE Computer, 1991(Aug.)

13 Bennett J K, Carter J B, Zwaenepoel W. MUNIN: Distributed Shared Memory Based on Type-Specific Memory Coherence 2nd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPOPP), March 1990. 168～176

14 Itzkovitz A, Schuster A. MultiView and Millipage-Fine-Grain Sharing in Page-Based DSMs. 3rd OSDI, Feb. 1999

15 Wilson Jr A W., LaRowe Jr R P. Hiding Shared Memory Reference Latency on the Galactica Net Distributed Shared Memory Architecture. Journal of Parallel and Distributed Computing, 1992, 15(4): 351～367