[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
H04L 12/24 (2006.01)
H04L 29/08 (2006.01)
G06F 9/46 (2006.01)

[12] 发明专利说明书

专利号 ZL 200610164850.0

[45] 授权公告日 2009 年 5 月 6 日

[11] 授权公告号 CN 100486178C

[22] 申请日 2006.12.6

[21] 申请号 200610164850.0

[73] 专利权人 中国科学院计算技术研究所

地址 100080 北京市海淀区中关村科学院南路 6 号

[72] 发明人 王 楠 刘旭辉 韩冀中 贺 劲 章立生

[56] 参考文献

CN1391176A 2003.1.15

CN1553716A 2004.12.8

CN2569238Y 2003.8.27

CN1347034A 2002.5.1

US6205528B1 2001.3.20

US5592625A 1997.1.7

CN1447257A 2003.10.8

CN1447255A 2003.10.8

动态共享内存缓冲池技术. 余翔湛，殷丽华. 哈尔滨工业大学学报，第 36 卷第 3 期. 2004

基于内存映射文件的数据共享技术研究与应用. 孙文庆，刘秉权，肖镜辉. 微计算机应用，第 26 卷第 2 期. 2005

内存映射文件及其在大数据量文件快速存取中的应用. 杨宁学，诸昌钤，聂爱丽. 计算机应用研究，第 8 期. 2004

审查员 王伦杰

[74] 专利代理机构 北京泛华伟业知识产权代理有限公司

代理人 高存秀

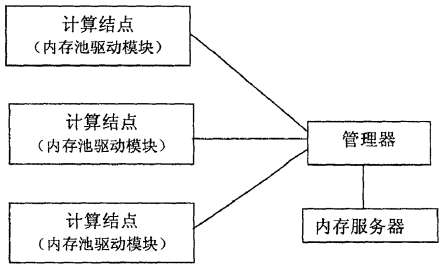
权利要求书 3 页 说明书 14 页 附图 7 页

[54] 发明名称

一种远程内存共享系统及其实现方法

[57] 摘要

本发明公开了一种远程内存共享系统，包括管理器和计算结点，还包括内存服务器；其中，在计算结点上安装有为远程内存共享系统提供并管理物理内存块的内存池驱动模块，带有内存池驱动模块的计算结点通过连接器连接到网络上，计算结点和内存服务器都通过网络与管理器连接；所有计算结点的内存池驱动模块为远程内存共享系统提供的物理内存和内存服务器所提供的内存组成一个内存池，内存池中的内存供各个计算结点中的应用共享；在计算结点上还包括提供内存映射的映射器和提供块设备的交换器。 本发明还公开了一种远程内存共享实现方法。 本发明可解决内存资源浪费的缺陷，并可兼顾存储类应用和计算类应用，还可同时应用于用户态应用和内核态应用。



1、一种远程内存共享系统，包括管理器和至少两个计算结点，其特征在于，还包括内存服务器；其中，在所述的计算结点上安装有为所述的远程内存共享系统提供并管理物理内存块的内存池驱动模块，所述的带有内存池驱动模块的计算结点通过连接器连接到网络上，所述的计算结点和所述的内存服务器都通过网络与所述的管理器连接；所有计算结点的内存池驱动模块为远程内存共享系统提供的物理内存和内存服务器所提供的内存组成一个内存池，内存池中的内存供各个计算结点中的应用共享；在所述的计算结点上还包括提供内存映射的映射器和提供块设备的交换器；所述内存池驱动模块还包括向内核态应用提供接口；

所述的内存池驱动模块通过空闲链表 free_list、用于链接本地应用所使用的本地结点的物理内存块的块描述的 LLL 链表、用于链接远程应用所使用的本地结点的物理内存块的块描述的 RLL 链表、用于链接远程应用所使用的远程结点的物理内存块的块描述的 RRL 链表、用于链接本地应用所使用的远程结点的物理内存块的块描述的 LRL 链表对整个远程内存共享系统中的物理内存块进行管理；

在所述的远程内存共享系统中，应用将一个物理内存块存入该远程内存共享系统，作为交换，所述远程内存共享系统提供给所述应用一个物理内存块使用；所述应用对已经分配给它的物理内存块有完全的控制权，而对于所述应用提供给所述远程内存共享系统的物理内存块不能再直接使用，需要时所述应用只能通过所述物理内存块的编号用另外一个物理内存块换回，所述远程内存共享系统保证换回的块内容与先前存入的内容一致，但不保证换回的是同一个物理内存块。

2、根据权利要求 1 所述的远程内存共享系统，其特征在于，所述的映射器是一个实现 mmap 接口的字符设备。

3、根据权利要求 1 所述的远程内存共享系统，其特征在于，所述的交换器是一个块设备。

4、根据权利要求 1 所述的远程内存共享系统，其特征在于，所述的内存服务器使用文件系统，通过操作系统的页面缓存技术，将本结点的内存资源作为缓存，为远程内存共享系统中的计算结点提供内存。

5、一种应用于权利要求 1 所述的远程内存共享系统实现远程内存共享的方法，包括：

步骤 10、一个计算结点上的应用申请内存块，首先从本地的空闲链表 free_list 中划拨空闲块的块描述到用于链接本地应用所使用的本地结点的物理内存块的块描

述的 LLL 链表中，若在 free_list 中没有空闲内存块的块描述或空闲内存块的块描述不够，则执行下一步；

步骤 20、本地的空闲内存块不能满足本地应用的需求，内存池驱动模块向管理器发出请求，要求管理器为其寻找空闲内存块，管理器将查找结果返回给发出请求的结点，发出请求的结点与空闲内存块所在的结点间建立通信连接；

步骤 30、本地结点与空闲内存块所在的结点建立通信连接后，内存池驱动模块将用于链接远程应用所使用的本地结点的物理内存块的块描述的 RLL 中链表尾部的块描述所对应的物理块中的数据传输到远程结点，并将这些块描述链接到用于链接远程应用所使用的远程结点的物理内存块的块描述的链表 RRL 上，所空出的物理内存块供本地应用使用，若所得到的空闲内存块仍然不能满足应用的需求，则执行下一步；

步骤 40、将链表 LLL 中尾部的块描述连接到用于链接本地应用所使用的远程结点的物理内存块的块描述的链表 LRL 上，该块描述所对应的物理内存块中的数据传送到远程结点的物理内存块上，所空出的物理内存块供本地应用使用；

步骤 50、本地结点上的应用访问内存块，包括：

步骤 51、判断本地应用所要访问的内存块的块描述的位置，若内存块的块描述在链表 LLL 上，执行下一步，若内存块的块描述在 LRL 上，执行步骤 53；

步骤 52、所要访问的内存块的块描述在 LLL 上，将所要访问内存块的块描述提到 LLL 链表头，用传入的物理块地址替换原先的物理块地址，将原先保存的物理块地址交给应用；

步骤 53、所要访问的内存块的数据在远程结点上，本地结点与远程结点作远程通信，将 LLL 上链表尾部的块描述所对应的物理内存块与远程结点上所要访问的数据所在的物理内存块之间进行数据交换，把数据被交换到远程结点的内存块的块描述链入 LRL 的头部， LLL 尾部的块描述所对应的物理内存块交换到所要访问的数据后，将该块描述置于 LLL 的头部；

步骤 60、远程结点上的应用访问内存块，包括远程结点上的应用对保存在本地 RRL 链表中的块描述所对应的物理内存块进行访问，将该块描述所对应的物理内存块的数据在远程结点上的存储位置返回给远程结点的应用，并将该块描述从 RRL 链表中删除；

步骤 70、本地应用对内存块的使用结束，释放内存块，优先释放块描述位于

LRL 上的物理内存块；

步骤 80、释放块描述位于链表 LLL 上的物理内存块，当内存块释放完毕后，检查在当前的链表 LRL 上，是否还有块描述，若还有块描述，则利用已释放的块描述对应的物理内存块，将数据保存在远程结点上的内存块的数据读回本地结点。

一种远程内存共享系统及其实现方法

技术领域

本发明涉及计算机中内存共享的实现，特别涉及一种用于实现远程内存共享的系统及方法。

背景技术

随着网络技术的不断发展，网络的速度不断提高。近年来出现了 InfiniBand，10G 以太网等新技术。这些技术的出现，为集群系统的应用创造了良好的条件，当前，集群系统普遍应用在并行计算、Web、科学计算、数据库等方面。

集群系统可以为用户提供大量的 CPU 和内存资源，但由于集群中每一个结点依然是一个自治的个体，其内存资源并不能得到有效的利用，使得集群的内存资源存在着严重的浪费：在桌面应用中，随着为集群配置的内存的增加，集群中空闲内存也随之增加。从 32MB 中的 12--14MB 到 256MB 中的 180--192MB；在内存容量超过 64MB 的机器上，一半内存空闲时间超过 12 分钟，四分之一的内存空闲超过 30 分钟；在集群的全部结点中，空闲内存在集群总内存中所占比例为 60-68%；仅在空闲结点中就有占集群中总内存的 53%的内存处于空闲状态（请见参考文献 4：A. Acharya and S. Setia. Availability and utility of idle memory in workstation clusters[J]. *ACM SIGMETRICS Performance Evaluation Review*, 1999.）。

而另一方面，在实际应用中还可以观察到，由于结点间负载不平衡，繁忙的结点由于物理内存不足而被迫使用磁盘交换空间。磁盘交换空间虽然将应用逻辑上可用的内存数量提高到只受磁盘容量的限制，且磁盘价格远远低于内存，但磁盘的速度也远远慢于内存。这就造成了在集群系统中，空闲结点有大量的内存空闲不能得到应用，而繁忙结点却由于物理内存不足导致结点工作效率低的不均衡现象。

要克服上述集群系统中内存空间利用的不均衡现象，可以通过对集群中的内存空间的共享来实现。关于利用集群中空闲内存空间的研究，从上世纪 90 年代就已开始。Michael J. Franklin 等人在 1992 年就发现，在 C/S 的数据库系统中，一个 Server 为多个 Client 服务时，一个 Client 上的 Cache 对其它 Client 也有意义，因此有必

要进行全局的内存管理(具体情况可见参考文献1:M. J. Frankling, M. J. Carey, and M. Livny. Globla memory management in client-server dbms architectures[A]. In *Proceeding of the 18th VLDB Conference*[C]. 1992.)。1993年Liviu Iftode等人提出了“内存服务器”的概念，将集群中的结点分为“计算结点”和“内存服务结点”，后者在前者产生页面错误时为它提供内存（请见参考文献2：L. Iftode, K. Li, and K. Petersen. Memory servers for multicomputers[A]. In *Proceeding of the IEEE Spring COMPCON 93*[C]. 1993:538-547.)。1995年Michael J. Feeley在参考文献1研究的基础上提出了一个GMS系统，后来的GMS系统多是对这个系统的改进（请见参考文献3：M. J. Feeley, W. E. Morgan, F. H. Pighin, A. R. Karlin, H. M. Levy, and C. A. Thekkath. Implementing global memory management in a workstation cluster[J]. *ACM SIGOPS Operating Systems Review*, 1995, pages 201 - 212.)。这种GMS系统有以下特点：

- 全局内存管理，结点动态增减；
- 结点之间对等，没有Server和Client的区别；
- 使用磁盘增大存储容量；
- 没有管理结点，从计算结点中选举出master进行管理。

1999年Anurag Acharya等人系统地对集群中空闲内存的可用性进行了分析，得出了上文提到的集群中存在严重内存浪费的结论，对使用集群中空闲内存具有指导意义。

2003年Michael R. Hines等人在Anemone中使用网络交换空间的技术，提出了使用远程内存的一种新方法（请见参考文献5：M. R. Hines, M. Lewandowski, and K. Gopalan. Anemone: Adaptive network memory engine[D]. Master’s thesis, Florida State University, 2003.)。之后，Tia Newhall（参考文献6：T. Newhall, S. Finney, K. Ganchevm, and M. Spiegel. Nswap:a network swapping module for linux clusters[A]. In *Proceeding of Euro-Par’03 International Conference on Parallel and Distributed Computing*[C]. Klagenfurt, Austria, 2003.)、Guozhong Sun（参考文献7：H. Tang Sun, M. Chen, and J. Fan. A scalable dynamic network memory service system[A]. In *Proceeding of High-Performance Computing in Asia-Pacific Region*[C]. 2005.）分别用Nswap和NBD实现了网络交换空间。2005年Shuang Liang在InfiniBand上实现了网络交换空间（参考文献8：S. Liang, R. Notonha, and D. K. Panda. Swapping to

remote memory over infiniband: An approach using a high performance network block device[J]. *IEEE Cluster Computing*, September 2005.)。

总结前人的研究成果，在利用远程结点内存提高集群性能的实现技术方面，主要分为以下几种：

- 核心态捕获远程分页(page fault)；
- 提供 memcpy 式接口；
- 块设备与交换空间；
- 在用户态捕获 SIGSEGV。

Eric A. Anderson 等人对 6 种使用网络内存的实现方法进行了总结：

显式的修改应用程序：在应用程序中直接写入将数据存入远程和从远程找回数据的代码。这种技术需要对应用程序的代码作重大的修改。但是这种实现方案可以最大限度的提升性能，降低开销，因为这种方案处理了整个内存使用结构且可以针对应用程序访问模式进行设计；

新的 malloc 函数：这种解决方案需要用户修改代码使用新的 malloc 函数从网络分配内存。这种方案对现有代码的修改比上一种少，但依然需要修改代码。用户可以限制应用程序占用的本机内存，使得其它的应用速度提高。相对于后面几种方案而言，这种方案通用性最好：它对操作系统所作的假设最少，但处理 page fault 中断和通过系统调用处理页表会带来额外的开销；

设备驱动程序：这种方案将交换设备替换成向网络内存发送页面的设备。这种方案的优点在于不需要对应用程序和系统内核做任何修改。缺点在于系统的页面存放在远程，对可靠性提出了很高的要求。它的通用性低于上一种方案，但强于修改内核。本方案的开销包括 VM 子系统的开销、内核态－用户态切换的开销。如果使用了额外的进程，则要加上进程上下文切换的开销；

修改内核：不修改应用程序的话，这个方案可以提供最佳的性能，因为它不需要额外的中断或上下文切换。但是，在不同的体系结构之间，修改内核的方案不能通用。因为 VM 的开销下降了，这个方案更可取，该方案降低了许多其它的开销；

网络接口：这个方案需要将内存控制器用某种网络内存芯片替换，因此这个方案是通用性最差的。但是它已经被应用于 Alewife、Flash、Shrimp 等多处理器体系结构中，用于处理共享内存。这个方案的开销低于修改内核，因为它可以以缓存行(Cache-line)为单位而不必以整个页面为单位，因此传输的数据量小。另外这种方案

可以避免 VM 的开销。多处理器会引入维护内存一致性的开销，但是如果只有一个处理器使用这个数据的话，这个开销就可以忽略。

关于对现有网络内存共享实现方法的详细内容可见参考文献 9：E. A. Anderson and J. M. Neefe. An exploration of network ram[R]. Technical Report CSD-98-1000,UC Berkley, December 1994。

与上述的内存共享实现方法相对应的，现有内存共享系统的体系结构可分为 client - server 式和对等式。client - server 式严格的说并不是一种应用远程结点空闲内存的方法，因为内存服务器不承担计算任务。client - server 结构的设计中大部分都用磁盘作为后备存储介质，可以为应用程序提供巨大的存储空间。对等式的设计可以有效的使用结点的剩余内存，但因为没有后备存储空间，对等式的结构能够提供的空间存在限制。

虽然到目前为止，前人已经做了很多的相关研究，但一些问题依然有待解决。

应用的需求各种各样，计算类的应用需要直接用内存进行计算，而存储类的应用倾向于使用块设备的接口将数据缓存。现有的技术不能兼顾这两种应用。

随着网络存储技术（例如 iSCSI） 的兴起，内核应用对内存的需求越来越大。Xubin He 等人提出在 iSCSI 的结构中加入 Cache 以提升 iSCSI 的性能，Jizhong Han 等人分析了 iSCSI 的 I/O 特性，论证了通过充分利用 iSCSI Target 内存以提高 iSCSI 性能的可行性。Xuhui Liu 等人通过 NBD 完成了 Remote iCache。而任何 iCache 都必须在内核空间实现。现有的技术无法为内核空间的应用提供支持。

另外，Client - Server 式的互连结构必须使用专门的内存服务结点，这些结点的 CPU 和内存资源一定程度上被浪费；而对等式的结构又无法像用磁盘做后备存储的 Memory Server 一样提供 TB 甚至 PB 级的空间。

发明内容

本发明的目的是克服现有的内存共享系统无法支持大规模内存请求，以及内存负载不平衡，造成资源浪费的缺陷，从而提高一种高效的远程内存共享系统。

本发明的另一个目的是克服现有的内存共享系统不能同时兼顾计算类应用和存储类应用的缺陷，从而提供一种能兼顾两种类型的应用的远程内存共享系统。

本发明的第三个目的是克服现有的内存共享系统不能同时兼顾内核态应用和用户态应用的缺陷，从而提供一种能兼顾两种类型的应用的远程内存共享系统。

本发明的又一个目的是提供一种实现远程内存共享的方法。

为了实现上述目的，本发明提供了一种远程内存共享系统，包括管理器和至少两个计算结点，还包括内存服务器；其中，在所述的计算结点上安装有为所述的远程内存共享系统提供并管理物理内存块的内存池驱动模块，所述的带有内存池驱动模块的计算结点通过连接器连接到网络上，所述的计算结点和所述的内存服务器都通过网络与所述的管理器连接；所有计算结点的内存池驱动模块为远程内存共享系统提供的物理内存和内存服务器所提供的内存组成一个内存池，内存池中的内存供各个计算结点中的应用共享；在所述的计算结点上还包括提供内存映射的映射器和提供块设备的交换器；所述内存池驱动模块还包括向内核态应用提供接口；

所述的内存池驱动模块通过空闲链表 free_list、用于链接本地应用所使用的本地结点的物理内存块的块描述的 LLL 链表、用于链接远程应用所使用的本地结点的物理内存块的块描述的 RLL 链表、用于链接远程应用所使用的远程结点的物理内存块的块描述的 RRL 链表、用于链接本地应用所使用的远程结点的物理内存块的块描述的 LRL 链表对整个远程内存共享系统中的物理内存块进行管理；

在所述的远程内存共享系统中，应用将一个物理内存块存入该远程内存共享系统，作为交换，所述远程内存共享系统提供给所述应用一个物理内存块使用；所述应用对已经分配给它的物理内存块有完全的控制权，而对于所述应用提供给所述远程内存共享系统的物理内存块不能再直接使用，需要时所述应用只能通过所述物理内存块的编号用另外一个物理内存块换回，所述远程内存共享系统保证换回的块内容与先前存入的内容一致，但不保证换回的是同一个物理内存块。

上述技术方案中，所述的映射器是一个实现 mmap 接口的字符设备。

上述技术方案中，所述的交换器是一个块设备的接口。

上述技术方案中，所述的内存服务器使用文件系统，通过操作系统的页面缓存技术，将本结点的内存资源作为缓存，对远程内存共享系统中的计算结点提供内存。

本发明还提供了一种实现远程内存共享的方法，包括：

步骤 10、一个计算结点上的应用申请内存块，首先从本地的空闲链表 free_list 中划拨空闲块的块描述到用于链接本地应用所使用的本地结点的物理内存块的块描述的 LLL 链表中，若在 free_list 中没有空闲内存块的块描述或空闲内存块的块描述不够，则执行下一步；

步骤 20、本地的空闲内存块不能满足本地应用的需求，内存池驱动模块向管理器发出请求，要求管理器为其寻找空闲内存块，管理器将查找结果返回给发出请求的结点，发出请求的结点与空闲内存块所在的结点间建立通信连接；

步骤 30、本地结点与空闲内存块所在的结点建立通信连接后，内存池驱动模块将用于链接远程应用所使用的本地结点的物理内存块的块描述的 RLL 中链表尾部的块描述所对应的物理块中的数据传输到远程结点，并将这些块描述链接到用于链接远程应用所使用的远程结点的物理内存块的块描述的链表 RRL 上，所空出的物理内存块供本地应用使用，若所得到的空闲内存块仍然不能满足应用的需求，则执行下一步；

步骤 40、将链表 LLL 中尾部的块描述连接到用于链接本地应用所使用的远程结点的物理内存块的块描述的链表 LRL 上，该块描述所对应的物理内存块中的数据传送到远程结点的物理内存块上，所空出的物理内存块供本地应用使用；

步骤 50、本地结点上的应用访问内存块，包括：

步骤 51、判断本地应用所要访问的内存块的块描述的位置，若内存块的块描述在链表 LLL 上，执行下一步，若内存块的块描述在 LRL 上，执行步骤 53；

步骤 52、所要访问的内存块的块描述在 LLL 上，将所要访问内存块的块描述提到 LLL 链表头，用传入的物理块地址替换原先的物理块地址，将原先保存的物理块地址交给应用；

步骤 53、所要访问的内存块的数据在远程结点上，本地结点与远程结点作远程通信，将 LLL 上链表尾部的块描述所对应的物理内存块与远程结点上所要访问的数据所在的物理内存块之间进行数据交换，把数据被交换到远程结点的内存块的块描述链入 LRL 的头部， LLL 尾部的块描述所对应的物理内存块交换到所要访问的数据后，将该块描述置于 LLL 的头部；

步骤 60、远程结点上的应用访问内存块，包括远程结点上的应用对保存在本地 RRL 链表中的块描述所对应的物理内存块进行访问，将该块描述所对应的物理内存块的数据在远程结点上的存储位置返回给远程结点的应用，并将该块描述从 RRL 链表中删除

步骤 70、本地应用对内存块的使用结束，释放内存块，优先释放块描述位于 LRL 上的物理内存块；

步骤 80、释放块描述位于链表 LLL 上的物理内存块，当内存块释放完毕后，检查在当前的链表 LRL 上，是否还有块描述，若还有块描述，则利用已释放的块描述对应的物理内存块，将数据保存在远程结点上的内存块的数据读回本地结点。

本发明的优点在于：

1、本发明的远程内存共享系统可以实现系统中各个结点内存资源的均衡，克服计算资源分配不合理所造成的资源浪费现象。

2、本发明的远程内存共享系统通过 mapper 提供内存映射，可以用于内存类的应用；通过 swapper 提供块设备，可以用于存储类的应用，克服了现有技术不能同时应用于内存类应用和存储类应用的缺陷。

3、本发明通过 mapper 和 swapper 为用户态应用提供接口，通过内核接口函数为内核态应用提供接口，克服了现有技术不能同时应用于用户态应用和内核态应用的缺陷。

附图说明

图 1 为本发明的远程内存共享系统的结构图；

图 2 为计算结点的一个应用向远程内存共享系统申请、使用和释放块的过程图；

图 3 为一个计算结点上的 5 个空闲内存块未占用前 5 个链表的示意图；

图 4 为一个计算结点上的本地应用占用一个物理内存块后 5 个链表的示意图；

图 5 为计算结点上的本地应用读取数据后 5 个链表的示意图；

图 6 为远程结点从本地结点调度内存块后 5 个链表的示意图；

图 7 为计算节点的本地应用再次申请 6 个内存块后 5 个链表的示意图；

图 8 为计算节点的本地应用为新申请的 6 个内存块添加相应数据后 5 个链表的示意图；

图 9 为计算结点中的本地应用访问内存块 1 时链表的变化情况示意图；

图 10 为计算结点中的本地应用访问内存块 1 后 5 个链表的示意图；

图 11 为远程结点上的应用要访问本地结点的内存块 2 时，链表的变化情况示意图；

图 12 为本地结点上的应用释放内存块 1、9 和 5 时，链表的变化情况示意图；

图 13 为一个计算结点申请内存块的示意图。

具体实施方式

下面结合附图和具体实施方式对本发明作进一步说明。

如图 1 所示，为本发明的远程内存共享系统的结构图，该远程内存共享系统由多个计算结点通过网络连接而成。在整个系统中，各个计算结点可以分为三种角色，

分别为内存池驱动模块（rmp_pool），管理器（rmp_mamager）和内存服务器（rmp_memserver）。

内存池驱动模块（rmp_pool）安装在计算结点上，它将本结点的一部分物理内存提供给远程内存共享系统（RMP），远程内存共享系统从计算结点上所得到的所有共享物理内存形成一个内存池。内存池驱动模块对外提供一接口组，远程内存共享系统（RMP）的应用通过该接口组可访问各个计算结点提供给 RMP 的物理内存。rmp_pool 是 RMP 的核心，它使用连接器（connector）和其它结点(计算结点或内存服务器)互连。

在计算结点的内存池驱动模块（rmp_pool）上还有两个特殊的应用，分别为映射器（mapper）和交换器（swapper）。其中，映射器（mapper）是在内存池驱动模块（rmp_pool）上构建的一个应用。它是一个实现 mmap 接口的字符设备。用户态程序可以通过 mmap 系统调用将 RMP 内存映射到用户进程的地址空间。交换器（swapper）是在内存池驱动模块（rmp_pool）上构建的一个应用。swapper 是一个块设备的接口，它将 RMP 内存映射成块设备。可以在此之上构建文件系统以满足存储类应用，也可以使用它作交换空间以提升系统性能。本发明通过 mapper 提供内存映射，可以用于内存类的应用；通过 swapper 提供块设备，可以用于存储类的应用，克服了现有技术不能同时应用于内存类应用和存储类应用的缺陷。同时，rmp_pool 向内核态应用提供编程接口，克服了现有技术不能同时应用于内核态应用和用户态应用的缺陷。

管理器（manager）负责为提出使用远程结点内存请求的计算结点寻找和分配内存，在远程内存共享系统中，每一个计算结点都与作为管理器的计算结点相连接，并将本结点为 RMP 提供的内存数量向管理器注册。管理器保存所有计算结点和内存服务器的连接信息。在计算结点申请内存时，manager 通过轮询的方式为它发现空闲内存，计算结点可以通过 manager 发回的信息与其它结点建立连接。manager 承担的任务很少，不需要复杂的计算和大量的传输，因此不需要专门的结点，可以部署在计算结点或内存服务器上。

内存服务器按照 RMP 的协议，利用自身大容量的存储空间，对外提供“内存”。内存服务器并非将本身的内存资源贡献给 RMP，而是通过操作系统的文件系统，将自身的存储资源提供给 RMP。虽然在其它结点看来，内存服务器结点就是一个有大容量内存资源，和它们对等的计算结点，但实际上它是一个专用于服务的结点，本

身不承担计算任务。内存服务器使用文件系统，可通过操作系统的页面缓存等技术，将本机的内存资源作为缓存，加快访问速度。内存服务器通常部署在一个专用的结点上，不部署在计算结点上。

为了将带有内存池驱动模块（rmp_pool）的计算结点连接到网络上，需要在计算结点上加载连接器（connector）。连接器是一个用于实现连接的类，通过连接器可以将 RMP 网络部署在各种类型的网络上。

在本发明的远程内存共享系统中，对各个计算结点上的内存管理是实现远程内存共享的关键。在每个计算结点上，分出一部分的空闲物理内存，这些空闲内存作为内存池的一部分注册到管理器上。远程内存共享系统为各个计算结点上的应用提供了访问内存池的接口，计算结点在管理器的调度下，将本结点的空闲物理内存提供给其他结点使用，或者根据本地应用的请求，向其他结点请求空闲物理内存使用。

内存池驱动模块（rmp_pool）将计算结点上的内存分成等大的块，以“块”为内存管理的最小单位。一个“块”的大小是连续的一个或几个内存页面。由于采取了这种管理模式，远程内存共享系统（RMP）对外提供的最底层的接口(面向内核应用的接口)是基于“交换”的，即： 应用将一个物理内存块存入 RMP，作为交换，RMP 提供给它一个物理内存块使用。对已经分配给应用的块，称之为“属于应用”的，应用对它有完全的控制，可以进行任意的读、写甚至释放；RMP 不对它进行传输等操作。而对于应用提供给 RMP 的物理块，RMP 可能进行读写、传输，应用不能再直接使用，需要时应用只能通过块的编号用另一个物理块换回。RMP 保证换回的块内容和先前存入的一致，但不保证换回的是同一个物理块。

在图 2 中，描述了计算结点的一个应用如何向远程内存共享系统申请、使用和释放块的过程。

应用首先从 RMP 处获取页面。RMP 将块的编号返回给应用。之后应用根据编号从 RMP 处找回新的或先前存入的块，而存入的块顶替这个编号。使用完毕后通过 RMP 声明将块释放。应用并不关心 rmp_pool 是如何处理存入的页面的。

计算结点上的内存池驱动模块（rmp_pool）对物理内存块进行管理时，采用了 5 种链表，内存池驱动模块把对内存块的描述（简称块描述）链接在 5 个链表上，以实现对物理内存块的调度和分配。所述的 5 个链表分别为 free_list、LLL、LRL、RLL、RRL。其中 free_list 是空闲链表，所有既不被本地应用使用也不被远程结点使用的物理内存块的块描述连接在这条链表上，当需要分配时从 free_list 中找一个

空闲的物理内存块的块描述，并将该块描述与其所对应的物理内存块分配给应用。对其余 4 个链表的含义在表 1 中作了详细说明。

名称	使用者	提供者
LLL	本地应用	本结点
LRL	本地应用	远程结点
RLL	远程应用	本结点
RRL	远程应用	远程结点

表 1

在上述链表中，LRL 和 RLL 不可能都非空，因为这两个链表都非空代表远程应用使用本地物理内存块的同时，本地应用也在使用远程结点上的物理内存块，这种情况会大大降低远程内存共享系统的工作效率，RMP 在分配和释放内存时能够避免这种情况。在上述 5 种链表中的块描述包含了物理内存块的相应信息，如编号、位置、在远程结点上的编号等。链表 LLL 和链表 RLL 上的一个块描述对应了本地结点中的一个物理内存块。

本发明的远程内存共享系统实现远程内存共享的方法包括内存块的申请、内存块的访问和内存块的释放三大步骤，下面分别进行说明。

一、内存块的申请。

步骤 10、一个计算结点上的应用申请内存块，首先从本地的 free_list 中划拨空闲块的块描述到 LLL 链表中，若在 free_list 中没有空闲内存块的块描述或空闲内存块的块描述不够，则执行下一步。

如图 3 所示，在一个实施例中，空闲链表 free_list 上有 5 个空闲内存块，分别用 1、2、3、4、5 来标记内存块。各个内存块的本地物理地址分别为 0x0、0x1、0x2、0x3 和 0x4。空闲内存块标记为 clean，表示目前这些块中还没有存入有意义的信息。当应用采用一个新块替换它后，称这个块为 dirty 的。在远程传输时，如果一个块是 clean 的可以不进行传输，以节省网络带宽。为方便描述，以后的示例中假设这个结点上的应用只持有 1 个物理内存块，最初的物理地址为 0x5。

在图 4 中，一个计算结点上的本地应用占有一个物理内存块，该应用向 RMP 申请一个空闲内存块，把空闲内存块 1 的块描述由空闲链表 free_list 转移到链表 LLL 中。

在前面的描述中已经提到，应用在对 RMP 中的内存块做访问时，是基于“交换”的。应用在访问内存块 1 之前有自己的物理内存块，该物理内存块的地址为 0x5，

基于交换的原理，经应用访问过的内存块 1 的物理地址不再是原来的 0x0，而是 0x5。如图 5 所示，交换后的物理内存块中保存着数据 AAA，记录在块描述中。

步骤 20、本地的空闲内存块不能满足本地应用的需求，内存池驱动模块（rmp_pool）向管理器（rmp_mamager）发出请求，要求管理器为其寻找空闲内存块，管理器将查找结果返回给发出请求的结点，发出请求的结点与空闲内存块所在的结点间建立通信连接。

在本步骤中，还涉及到本地结点为远程计算结点分配内存块。当本地结点接收到管理器要求为远程结点分配内存块的请求时，从空闲链表中将相应数量的空闲内存块移到链表 RLL 上，并将实际提供的内存块的数量返回给管理器，由管理器将信息通知给相应的远程结点。

图 13 描述了计算结点 1 申请内存的过程。在申请内存之前，计算结点 2 通过 swapper 使用了计算结点 1、计算结点 4 的内存，计算结点 1 通过 mapper 向应用程序提供服务。在图示的时间，计算结点 1 上的应用通过 mapper 向 RMP 申请内存。计算结点 1 向管理器申请内存，管理器分别向计算结点 4 和内存服务器（memserver）询问剩余内存数量，并将这两个结点的信息返回计算结点 1。计算结点 1 根据返回的结果分别与内存服务器和计算结点 4 建立连接，并将计算结点 2 使用的一部分内存转移到计算结点 4 上。

如图 6 所示，本地结点收到管理器所发出的通知，为远程结点上的其他应用预留了标号为 2、3、4 的空闲内存块，其中，远程结点上的应用使用了其中的内存块 2、3。对上述被远程结点申请的内存块，在链表 RLL 中有对它们的块描述。

步骤 30、本地结点与空闲内存块所在的结点建立通信连接后，内存池驱动模块（rmp_pool）将链表 RLL 中的靠后的块描述所对应的物理块（不经常被访问的块）中的数据传输到远程结点，并将这些块描述连接到链表 RRL 上，所空出的物理内存块供本地应用使用，若所得到的空闲内存块仍然不能满足应用的需求，则执行下一步。

步骤 40、将链表 LLL 中尾部的块描述连接到链表 LRL 上，该块描述所对应的物理内存块中的数据传送到远程结点的物理内存块上，所空出的物理内存块供本地应用使用。

如图 7 所示，在本实施例中，计算结点上的本地应用又申请了 6 个内存块，如前所述，在空闲链表上还保存有一个空闲内存块 5，将该内存块供本地应用使用，

并在链表 LLL 上添加该内存块的块描述。本地应用还需要 5 个内存块，但在本地结点上已经无法满足其需要，因此向管理器发出申请内存的请求，管理器对远程内存共享系统上的其他结点进行查找，发现标记为 N2 的计算结点上有空余的内存块，于是本地结点（可以记为 N1）和 N2 建立通信连接。然后将链表 RLL 上的内存块 2、3、4 中的数据传输到结点 N2 上的空闲内存块中，并在链表 RRL 添加传输到远程结点上的内存块的块描述，链表 RRL 上的块描述除了记录物理内存块在本地的编号外，还记录了物理内存块所传输数据所在结点的编号，在远程结点上所传输数据所在内存块的编号。在本地结点上，数据传输后所得到的空闲物理内存块供本地应用使用。在图 7 中可以看到，标记为 6、7、8 的内存块分别占用了内存块 4、3、2 原先占有的内存块（它们的物理地址相同）。经过此操作后，链表 LLL 从链表头到链表尾部的内存块顺序为内存块 8、7、6、5、1。由于本地应用一共申请了 6 个内存块，在前述的操作中完成了对 4 个内存块的请求，因此，还需要 2 个内存块。此时，把链表 LLL 上链表尾部块描述所对应的物理内存块的数据传输到远程结点上，并在链表 LRL 添加相应的块描述。物理内存块将自身的信息传送到远程结点后，可供本地结点上的应用使用。链表 LLL 末端为内存块 1 的块描述，对应的物理内存块地址为 0x5，所记录的数据为 AAA，将数据传输到远程结点上，并在链表 LRL 中添加内存块 1 的块描述，然后将得到的空闲物理内存块供本地应用使用，得到新的内存块标记 9，其地址不变，仍为 0x5。然后对内存块 5 做类似的操作，得到内存块 10。

在图 7 的描述中，应用完成了对物理内存块的申请，在图 8 中，应用为所申请到的各个内存块添加了相应的数据。

二、内存块的访问。

对于内存块的访问，可以分为本地结点上的应用对内存块的访问，和远程结点上的应用对内存块的访问，而在两种访问操作的实现时，又会因为所要访问的内存块所在位置的不同而有不同的实现。下面分别就本地结点应用和远程结点应用对内存块的访问分别进行说明。

本地应用通过“交换”方式访问内存块中的信息，根据内存块在链表中的不同位置，对内存块的访问有不同的实现。

步骤 50、本地应用访问内存块。

步骤 51、判断本地应用所要访问的内存块的块描述的位置，若内存块的块描述在链表 LLL 上，执行下一步，若内存块的块描述在链表 LRL 上，执行步骤 53。

步骤 52、所要访问的内存块的块描述在链表 LLL 上，将所要访问内存块的块描述提到 LLL 链表头，用传入的物理块地址替换原先的物理块地址，将原先保存的物理块地址交给应用。

步骤 53、所要访问的内存块的数据在远程结点上（即在该内存块的块描述链表 LRL 上），本地结点与远程结点作远程通信，将链表 LLL 上链表尾部的块描述所对应的物理内存块与远程结点上所要访问的数据所在的物理内存块之间进行数据交换。把数据被交换到远程结点的内存块的块描述链入链表 LRL 的头部，链表 LLL 尾部的块描述所对应的物理内存块交换到所要访问的数据后，将该块描述置于链表 LLL 的头部。

如图 9 和图 10 所示，本地结点 N1 中的本地应用访问内存块 1。根据步骤 40 中所列举的实施例可知，内存块 1 的块描述在链表 LRL 上，即该内存块的数据保存在远程结点 N2 上。因此，从链表 LLL 的尾部取出内存块 6 的块描述，将该块描述所对应的物理内存块中保存的数据 GGG 传送到远程结点 N2 上，远程结点 N2 则把原本由内存块 1 所保存的数据 AAA 传送到本地结点 N1 上，并保存在内存块 6 所在的物理内存块上，将该物理内存块的标记对应于块描述 1，并将块描述置于链表 LLL 的头部。同时，还要将内存块 6 的块描述链接到链表 LRL 的头部。应用在访问内存块 1 时，做了交换操作，交换后的物理内存块的地址为 0x0，保存的数据为 ZZZ。

步骤 60、远程结点上的应用对保存在本地 RRL 链表中的块描述所对应的物理内存块进行访问。块描述在 RRL 链表上，表明该块描述所对应的物理内存块中的数据被传送到一个远程结点上，将该块描述所对应的物理内存块的数据在远程结点上的存储位置返回给远程结点的应用，并将该块描述从 RRL 链表中删除。

如图 11 所示，一个远程结点上的应用要访问本地结点的内存块 2。但该内存块中的数据已经在前面的操作中传送到远程结点 N2 上了，并在链表 RRL 中保留有相应的块描述，该块描述表明内存块 2 中的信息被传送到远程结点 N2 上，且保存在该结点的内存块 3 中。上述信息通过网络传送到要访问内存块 2 的应用后，该应用对远程结点 N2 作进一步的访问操作，在链表 RRL 中删除内存块 2 的块描述。

三、内存块的释放。

应用对内存块的访问操作结束后，释放不再访问的内存块。

步骤 70、在释放内存块时，优先释放块描述位于链表 LRL 上的物理内存块。

步骤 80、释放块描述位于链表 LLL 上的物理内存块，当内存块释放完毕后，

检查在当前的链表 LRL 上，是否还有块描述，若还有块描述，则利用已释放的块描述对应的物理内存块，将数据保存在远程结点上的内存块（即块描述在链表 LRL 上的内存块）的数据读回本地结点。

如图 12 所示，本地结点 N1 上的应用释放内存块 1、9 和 5。内存块 5 在远程结点 N2 上，本地结点与远程结点 N2 进行通信，通知该结点释放内存块 5，远程结点将释放得到的内存块的块描述置于该结点的空闲链表上。内存块 1 的块描述位于本地结点的 LLL 链表上，释放该内存块时，直接将该内存块的块描述从链表中删除，并将该块描述标记为 clean，然后安放在空闲链表中。同样的，对链表 LLL 上的内存块 9 可以做类似的释放操作。在内存块 1 和 9 释放结束时，检查当前的 LRL 链表上是否还有块描述，通过检查可知，内存块 6 的数据依然保存在远程结点 N2 上，因此利用内存块 1 或内存块 9 释放后得到的物理内存空间，将内存块 6 的数据从远程结点 N2 上读回，并将内存块 6 的块描述置于 LLL 链表中。

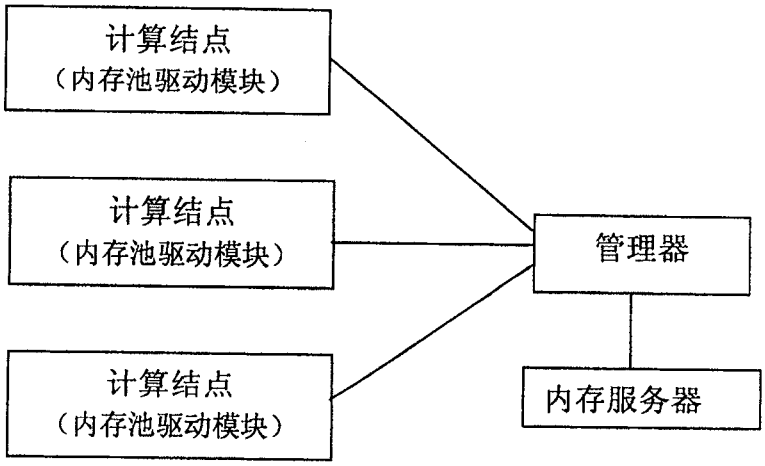

计算结点
（内存池驱动模块）
计算结点
（内存池驱动模块）
计算结点
（内存池驱动模块）
管理器
内存服务器

图 1

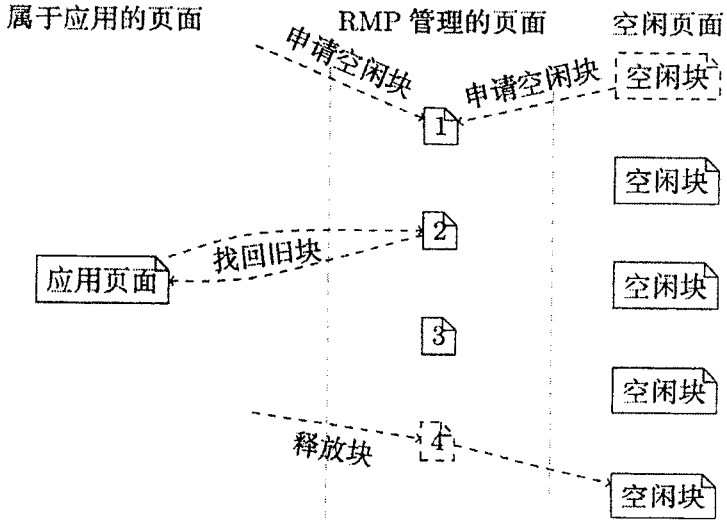

属于应用的页面
RMP 管理的页面
空闲页面
申请空闲块
申请空闲块
空闲块
1
空闲块
2
找回旧块
应用页面
空闲块
3
空闲块
释放块
4
空闲块

图 2

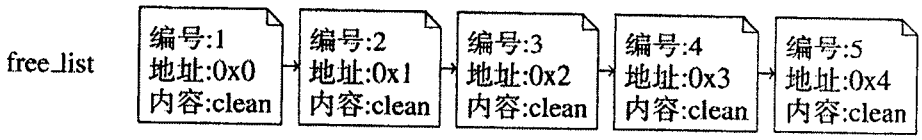

free_list
编号:1
地址:0x0
内容:clean
编号:2
地址:0x1
内容:clean
编号:3
地址:0x2
内容:clean
编号:4
地址:0x3
内容:clean
编号:5
地址:0x4
内容:clean
LLL
LRL
RLL
RRL

图 3

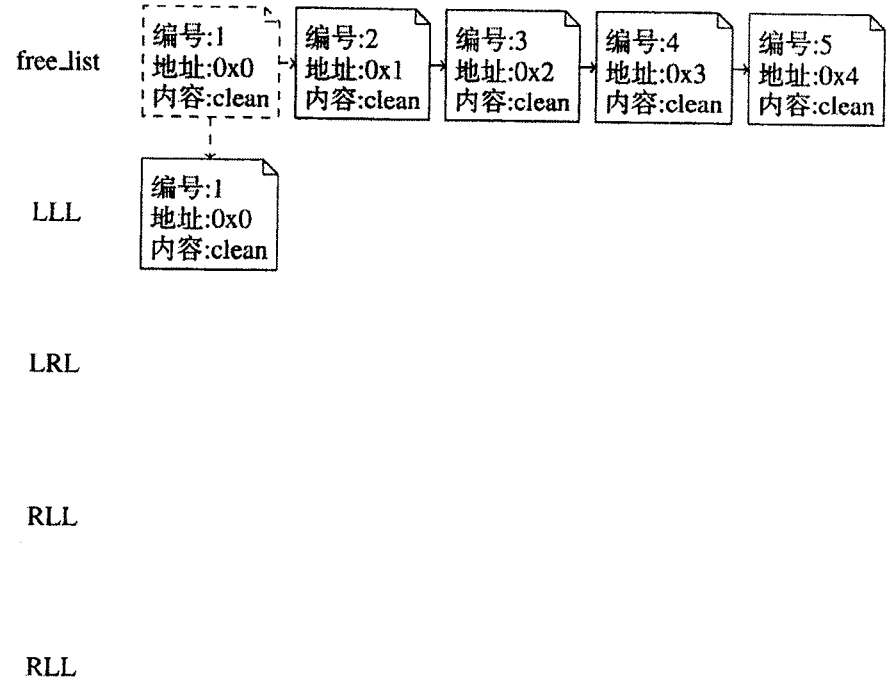

free_list
编号:1
地址:0x0
内容:clean
编号:2
地址:0x1
内容:clean
编号:3
地址:0x2
内容:clean
编号:4
地址:0x3
内容:clean
编号:5
地址:0x4
内容:clean
编号:1
地址:0x0
内容:clean
LLL
LRL
RLL
RLL

图 4

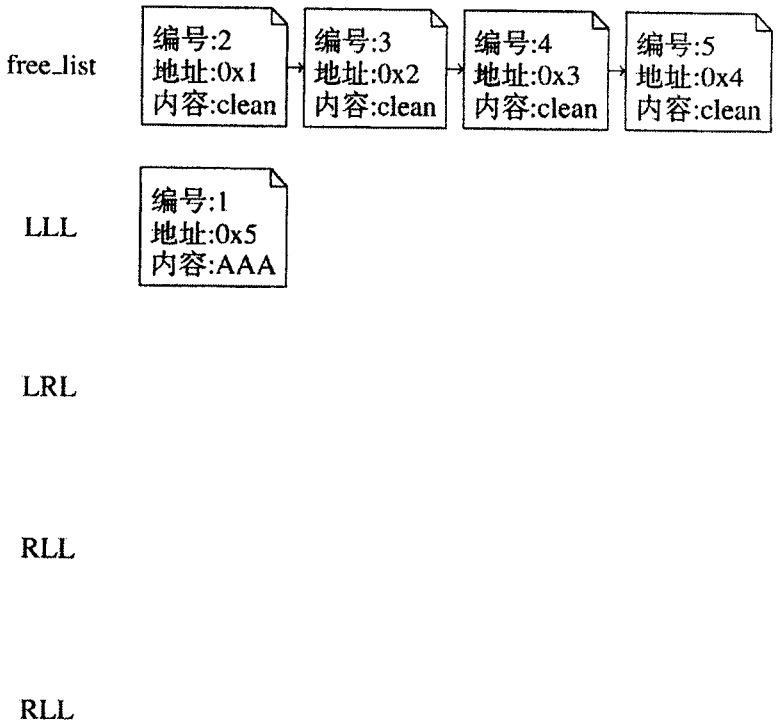

free_list
编号:2
地址:0x1
内容:clean
编号:3
地址:0x2
内容:clean
编号:4
地址:0x3
内容:clean
编号:5
地址:0x4
内容:clean
LLL
编号:1
地址:0x5
内容:AAA
LRL
RLL
RLL

图 5

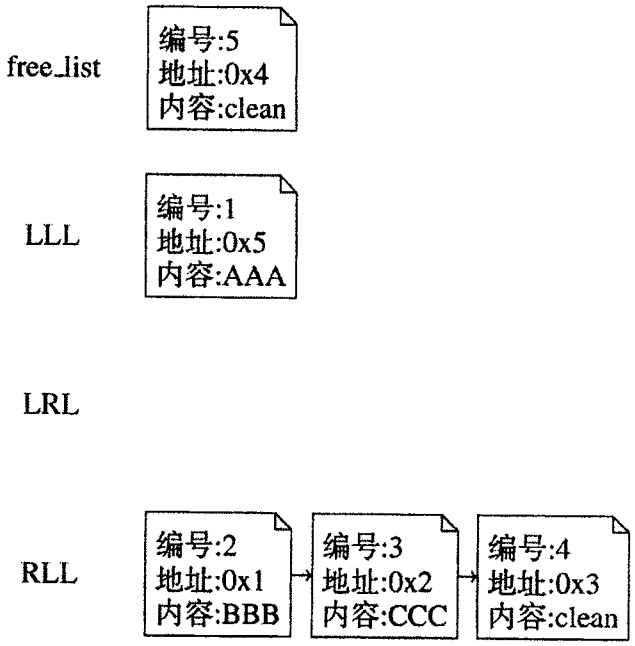

free_list
编号:5
地址:0x4
内容:clean
LLL
编号:1
地址:0x5
内容:AAA
LRL
RLL
编号:2
地址:0x1
内容:BBB
编号:3
地址:0x2
内容:CCC
编号:4
地址:0x3
内容:clean

图 6

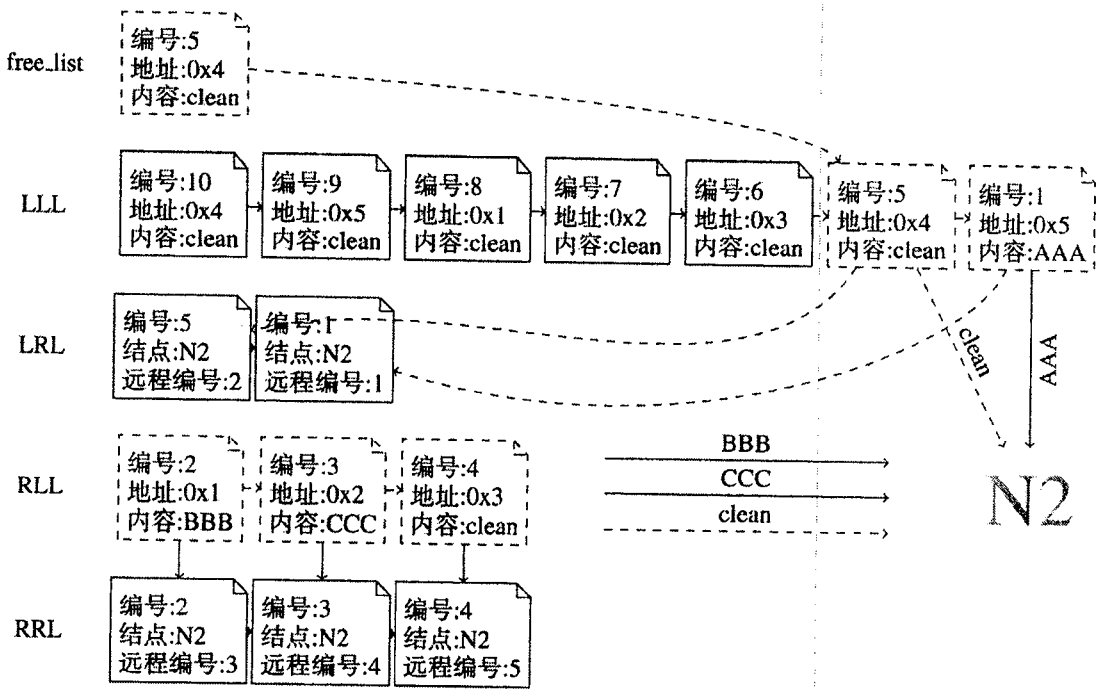

free_list
编号:5
地址:0x4
内容:clean
LLL
编号:10
地址:0x4
内容:clean
编号:9
地址:0x5
内容:clean
编号:8
地址:0x1
内容:clean
编号:7
地址:0x2
内容:clean
编号:6
地址:0x3
内容:clean
编号:5
地址:0x4
内容:clean
编号:1
地址:0x5
内容:AAA
LRL
编号:5
结点:N2
远程编号:2
编号:1
结点:N2
远程编号:1
clean
AAA
RLL
编号:2
地址:0x1
内容:BBB
编号:3
地址:0x2
内容:CCC
编号:4
地址:0x3
内容:clean
BBB
CCC
clean
N2
RRL
编号:2
结点:N2
远程编号:3
编号:3
结点:N2
远程编号:4
编号:4
结点:N2
远程编号:5

图 7

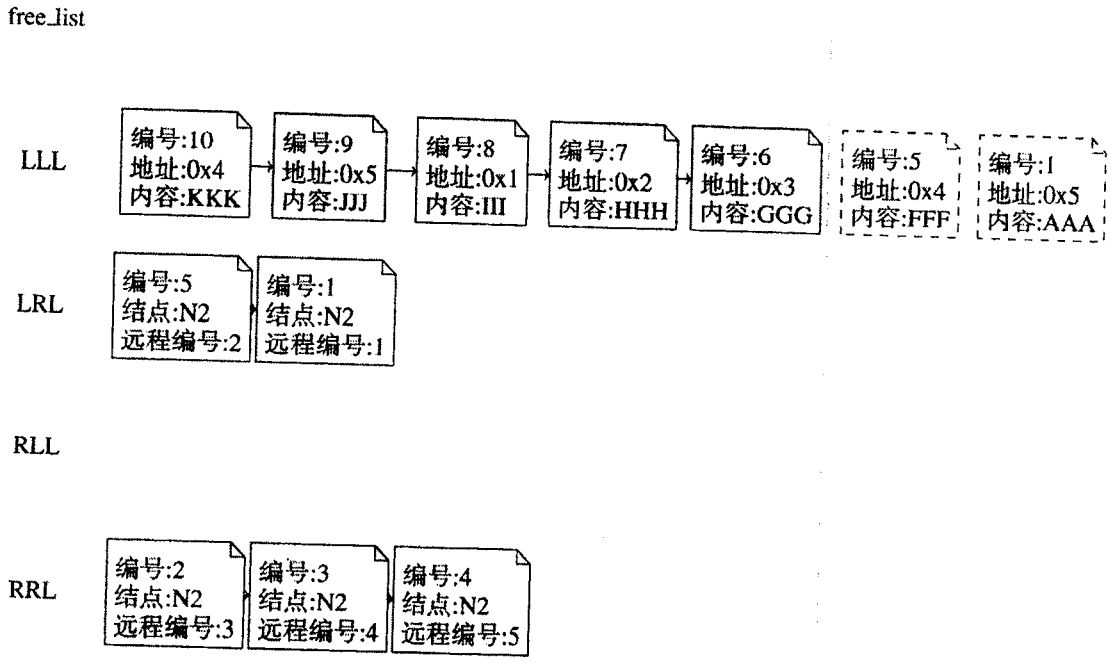

free_list
LLL
编号:10
地址:0x4
内容:KKK
编号:9
地址:0x5
内容:JJJ
编号:8
地址:0x1
内容:III
编号:7
地址:0x2
内容:HHH
编号:6
地址:0x3
内容:GGG
编号:5
地址:0x4
内容:FFF
编号:1
地址:0x5
内容:AAA
LRL
编号:5
结点:N2
远程编号:2
编号:1
结点:N2
远程编号:1
RLL
RRL
编号:2
结点:N2
远程编号:3
编号:3
结点:N2
远程编号:4
编号:4
结点:N2
远程编号:5

图 8

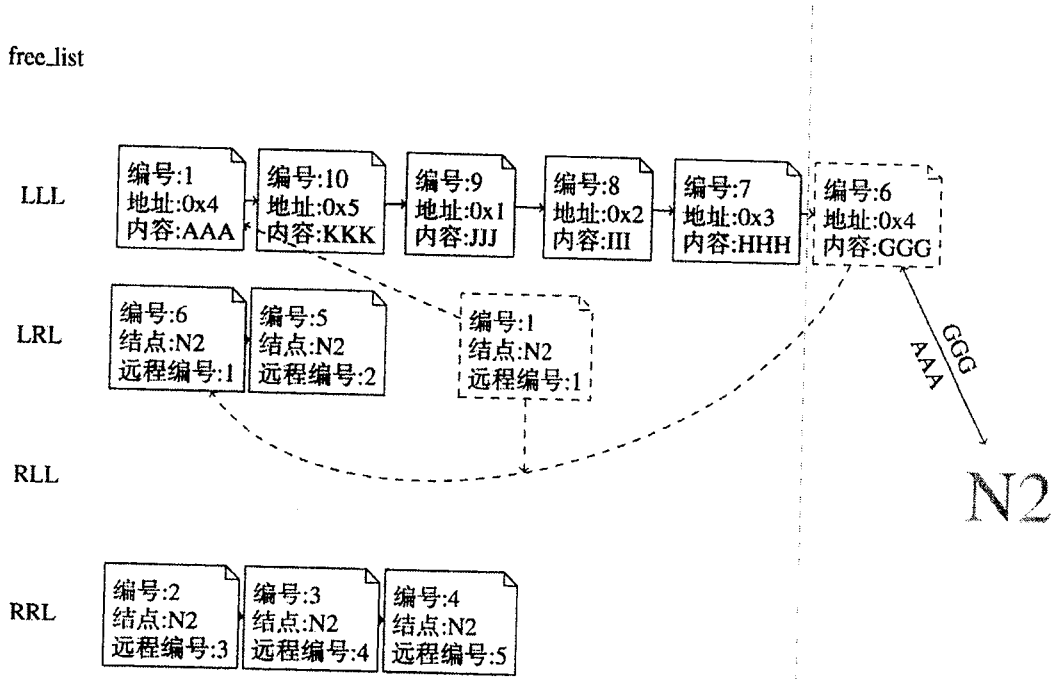

free_list
LLL
编号:1
地址:0x4
内容:AAA
编号:10
地址:0x5
内容:KKK
编号:9
地址:0x1
内容:JJJ
编号:8
地址:0x2
内容:III
编号:7
地址:0x3
内容:HHH
编号:6
地址:0x4
内容:GGG
LRL
编号:6
结点:N2
远程编号:1
编号:5
结点:N2
远程编号:2
编号:1
结点:N2
远程编号:1
GGG
AAA
N2
RLL
RRL
编号:2
结点:N2
远程编号:3
编号:3
结点:N2
远程编号:4
编号:4
结点:N2
远程编号:5

图 9

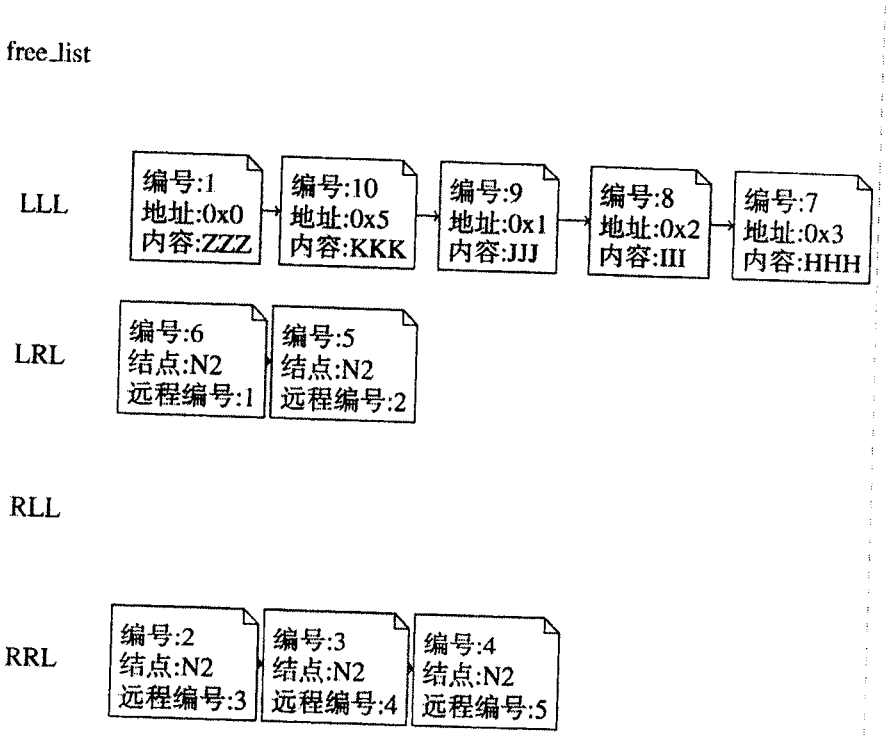

free_list
LLL
编号:1
地址:0x0
内容:ZZZ
编号:10
地址:0x5
内容:KKK
编号:9
地址:0x1
内容:JJJ
编号:8
地址:0x2
内容:III
编号:7
地址:0x3
内容:HHH
LRL
编号:6
结点:N2
远程编号:1
编号:5
结点:N2
远程编号:2
RLL
RRL
编号:2
结点:N2
远程编号:3
编号:3
结点:N2
远程编号:4
编号:4
结点:N2
远程编号:5

图 10

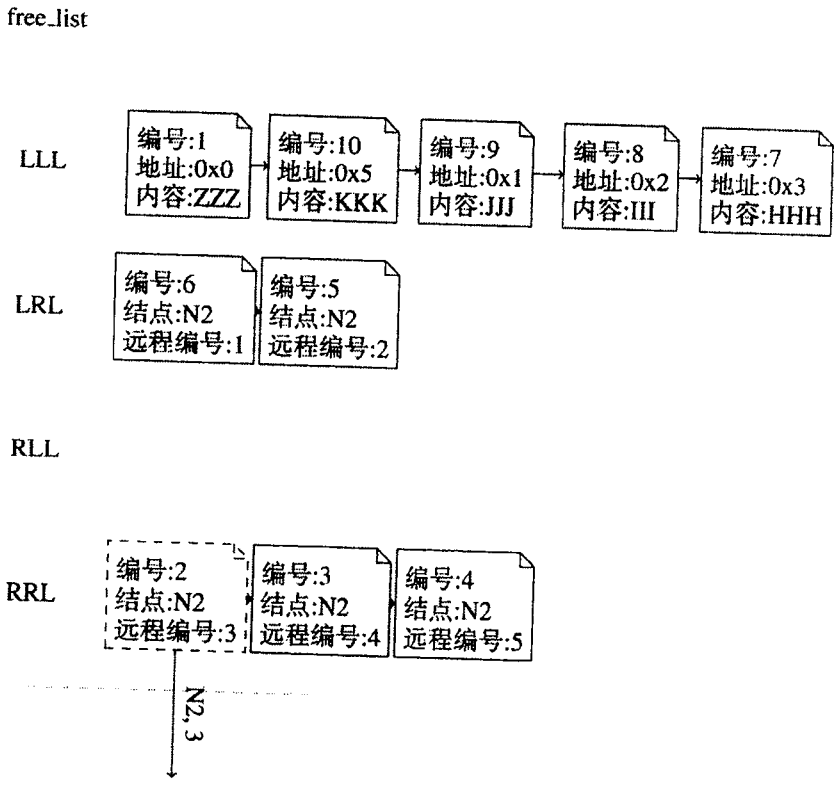

free_list
LLL
编号:1
地址:0x0
内容:ZZZ
编号:10
地址:0x5
内容:KKK
编号:9
地址:0x1
内容:JJJ
编号:8
地址:0x2
内容:III
编号:7
地址:0x3
内容:HHH
LRL
编号:6
结点:N2
远程编号:1
编号:5
结点:N2
远程编号:2
RLL
RRL
编号:2
结点:N2
远程编号:3
编号:3
结点:N2
远程编号:4
编号:4
结点:N2
远程编号:5
N2, 3

图 11

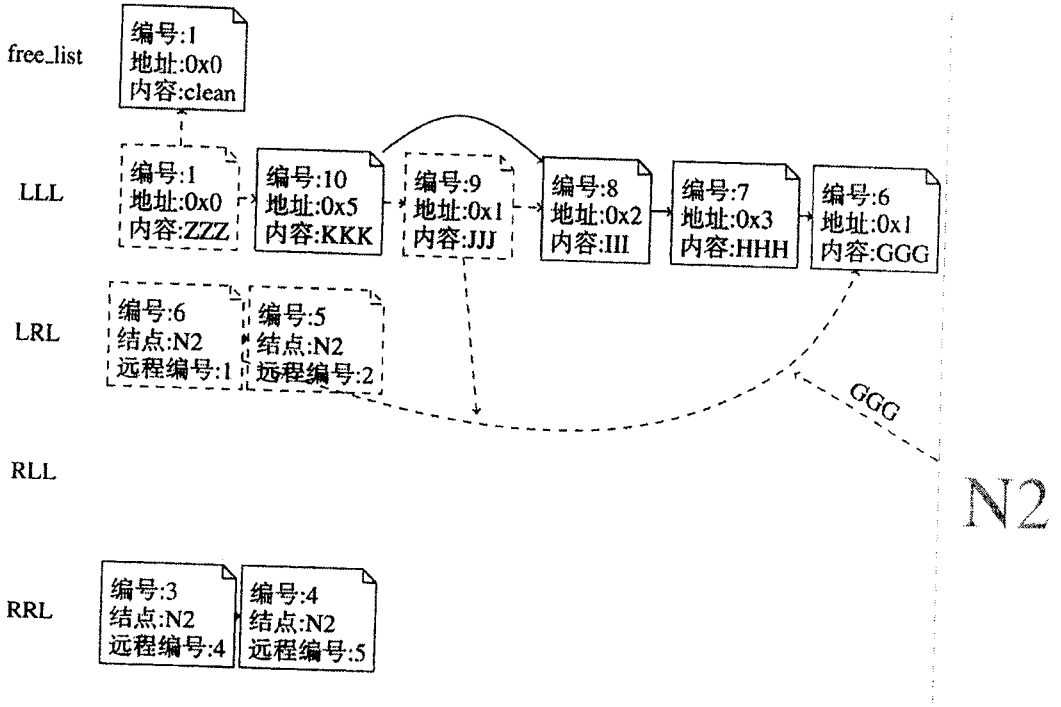

free_list
编号:1
地址:0x0
内容:clean
LLL
编号:1
地址:0x0
内容:ZZZ
编号:10
地址:0x5
内容:KKK
编号:9
地址:0x1
内容:JJJ
编号:8
地址:0x2
内容:III
编号:7
地址:0x3
内容:HHH
编号:6
地址:0x1
内容:GGG
LRL
编号:6
结点:N2
远程编号:1
编号:5
结点:N2
远程编号:2
GGG
RLL
N2
RRL
编号:3
结点:N2
远程编号:4
编号:4
结点:N2
远程编号:5

图 12

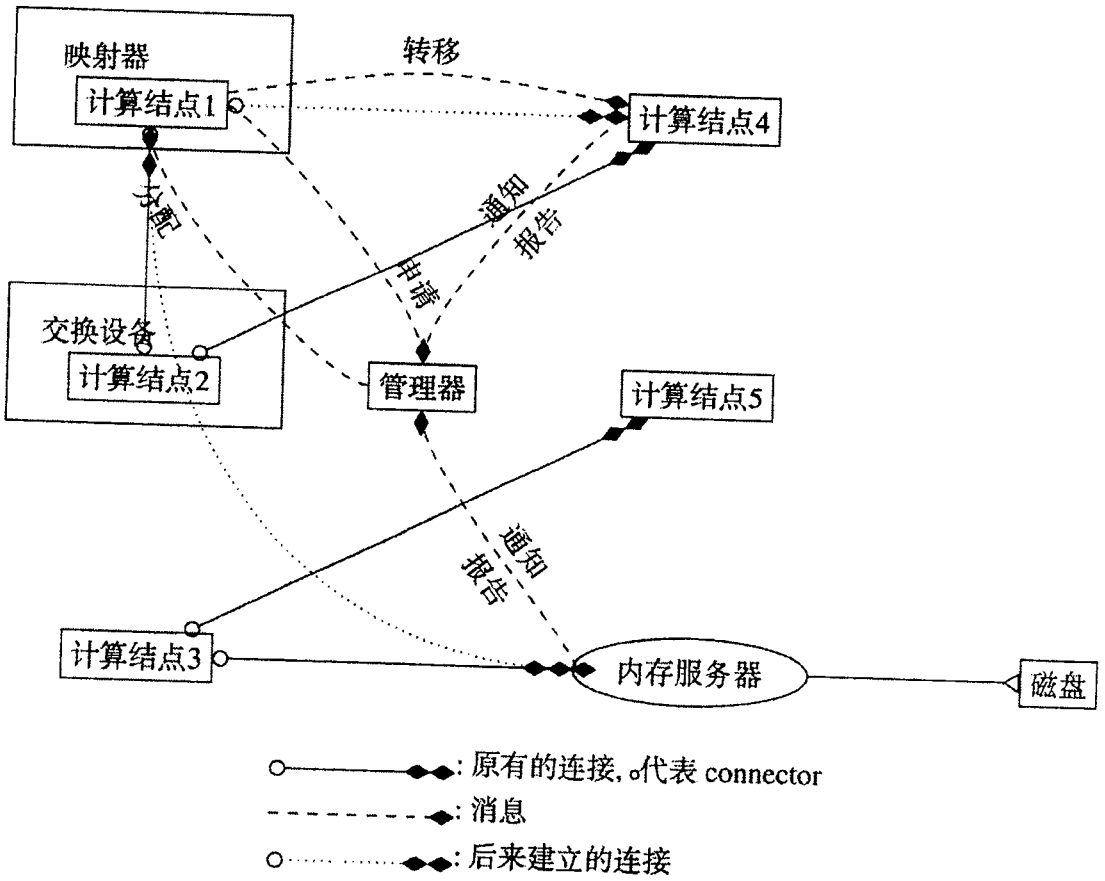

映射器
计算结点1
转移
计算结点4
申请
通知
报告
交换设备
计算结点2
管理器
计算结点5
通知
报告
计算结点3
内存服务器
磁盘
: 原有的连接, o代表 connector
: 消息
: 后来建立的连接

图 13