|  |  |
| --- | --- |
| **分 数：** |  |
| **评卷人：** |  |

****

**研究生（数据中心技术）课程论文（报告）**

**题 目：内存分解技术综述**

**学 号 Y202102009**

**姓 名 姚沛嵚**

**专 业 计算机科学与技术**

**课程指导教师 施展 童薇**

**院（系、所） 计算机科学与技术学院**

**2022年 1月7日**

内存分解技术综述

姚沛嵚 Y202002009

摘 要 随着内存需求的增长和内存技术的缓慢进步，足够的主内存的可用性越来越成为大型计算集群的瓶颈。由于需要同时实现透明的弹性、高资源利用率和硬件异构性，内存分解技术面临挑战。本文论述了几种分布式共享内存与内存分解技术在内存管理中的应用。详细介绍了这些技术的优点、不足与改进方法。总结现有工作并提出对未来方向的展望。

关键词 透明弹性 内存分解 分布式共享内存 内存管理

Overview of Memory Decomposition Technology

Peiqin Yao

**Abstract** With the increase in memory requirements and the slow progress of memory technology, the availability of sufficient main memory has increasingly become a bottleneck for large computing clusters. Due to the need to achieve transparent elasticity, high resource utilization, and hardware heterogeneity at the same time, memory decomposition technology faces challenges. This article discusses the application of several distributed shared memory and memory decomposition technologies in memory management. The advantages, disadvantages and improvement methods of these technologies are introduced in detail. And I also summarize the existing work and put forward the outlook for the future direction.

**Key words** Transparent elasticity; Memory disaggregation; Distributed shared memory; Memory management

**1、引言**

内存中工作负载（例如机器学习应用程序和键值存储）的日益普及导致计算集群中的内存需求快速增长。同时，由于摩尔定律的约束，DRAM 制造商正在努力实现更高的存储密度和更低的每存储单位成本。综上所述，这两种发展导致主内存越来越成为运行计算集群时的瓶颈。

内存分解一直是学术研究和商业项目的主题，是解决内存瓶颈的一种方法。其中，计算和内存物理上分离为网络连接的资源刀片，与传统数据中心体系结构相比，大大提高了资源利用率、硬件异构性、资源弹性和故障处理能力。

然而，由于三个方面的要求，内存分解具有挑战性。首先，对远程内存的访问必须具有低延迟和高吞吐量：之前研究表明，延迟达到，带宽达到100Gbps才可将应用程序性能降级降至最低。其次，应用程序可用的内存和计算资源必须弹性伸缩，以符合分解的要求。不幸的是，以前的内存分解设计暴露了这两个目标之间的艰难权衡。具体来说，应用程序计算资源的透明弹性伸缩需要在分解的内存池上进行共享内存抽象，这会由于应用程序数据和内存管理元数据所需的缓存一致性而带来不小的性能开销。

（1）透明设计。研究者提出两种方法。第一种是以计算为中心的方法，每个计算刀片拥有一个地址空间分区并管理相应的元数据，但内存本身是分解的。计算刀片现在必须等待完成每个未缓存内存读取或写入的多个连续远程请求，例如到远程主计算刀片，以触发块的状态转换并使相关刀片失效，并从当前拥有该块的刀片获取内存块。另一种以内存为中心的设计（将元数据放置在相应的主内存刀片服务器上）仍然像以前一样收到多个连续的远程内存访问请求，唯一的区别是主节点访问现在被定向到内存刀片服务器。虽然可以通过在计算刀片服务器上缓存元数据来减少开销，但这也需要元数据的一致性，从而导致额外的设计复杂性和性能开销。

(2)非透明设计。将进程限制为单个计算刀片，即当计算刀片在本地缓存数据时，不同的计算刀片不共享内存以避免通过网络发送一致性消息。因此，这些方案将应用程序的透明计算弹性限制在单个计算刀片上，如果希望扩展到计算刀片之外，则需要对应用程序进行修改。

基于此，本文详细论述了多种分布式共享内存与内存分解技术，并对其优缺点进行总结。

**2. 原理与优势**

**2.1.** **使用RDMA和缓存实现高效的分布式内存管理**

DSM（分布式共享内存）系统提出将物理分布式内存结合在一起，以实施统一的全局内存模型。这些系统通常使用缓存来缓冲远程内存访问。为了在缓存数据上保持一致的视图，它们使用同步原语来传播脏写入和清除缓存读取，这会在同步点产生大量开销。此外，要求程序员手动调用同步原语以确保数据一致性，这使得使用内存模型编程和调试变得困难。

RDMA（远程直接内存访问）的出现使网络I/O成为远程内存访问，从而进一步增强了统一内存模型的吸引力。因此，研究者提出了几个基于RDMA的系统，以利用RDMA从物理分布的内存节点实现统一的内存抽象。但是，它们仍然需要用户手动调用同步原语以实现缓存一致性。避免上述问题的一种方法是放弃缓存，以便将每个操作（例如读/写）路由到请求数据所在的节点。然而，即使使用RDMA，细粒度远程内存访问仍然会产生高延迟，如表1所示。

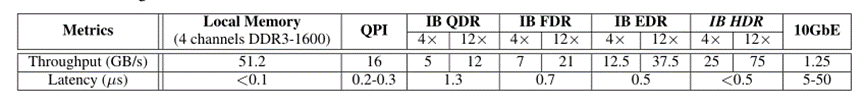


表1：远程内存访问延迟

基于此，Cai等[1]的研究提出了GAM，一种基于RDMA的分布式内存计算平台。原理和优势如下：

（1）GAM管理分布式内存以提供统一的全局内存模型。GAM为全局内存操作提供一组API。采用同步读取和异步写入的编程模型，并采用PSO内存一致性。提供了一组分布式同步原语，如锁和内存围栏，以增强一致性。

（2）GAM在全局内存之上添加了另一层分布式缓存，以利用数据局部性并隐藏远程内存访问的延迟。研究者基于RDMA设计了一个高效的分布式缓存一致性协议，以增强GAM提供的内存一致性模型。RDMA的各种特殊功能（例如，单边直接内存访问、纯报头通知、数据包内联）被用于高效的协议实现。

**2.2 Fastswap**

内存分解允许计算节点访问远程节点的内存，我们称之为远内存，使作业无需局限于本地内存。实现远内存有两个障碍。第一个在于如何设计访问远内存所需的交换机制，第二个在于在于如何决定如何在本地内存和远程内存之间划分每个作业的内存需求。

基于此，Amaro等[2]设计了基于Linux的交换系统Fastswap。对于第一个问题，该系统经过优化，可以通过RDMA使用远内存。它对应用程序和开发人员都是透明的。此外，它直接与Linux控制组交互，允许Fastswap强制执行本地内存分配。Fastswap通过在到单独队列的关键路径上引导页面获取来防止行首阻塞。此外，它还通过轮询关键页操作的完成情况和将内存回收卸载到专用CPU来减少关键路径上的延迟。

对于第二个问题，研究者设计了一个支持远内存的集群调度器，用来提高作业吞吐量。当新作业到达时，计划程序可以将作业放置在最初没有足够可用本地内存来处理分配给它的所有作业的服务器上。然后，我们的调度程序减少该服务器上某些现有作业使用的本地内存，并使用远内存来确保所有作业都可以访问足够的总内存。

实验表明，使用远内存的优势在于：（1）对于内存密集型工作负载，在两个机架配置的总内存量相同的情况下，将一个计算节点转换为一个远内存服务器可以使吞吐量比原始机架提高约10%。（2）可以对机架中的内存容量实现更精细的粒度增加。

**2.3** **MIND**

为同时支持内存分解的两个需求：高性能和透明弹性。Lee等[3]提出MIND，第一个用于机架级内存分解的内存管理系统。关键思想是将用于内存管理的逻辑和元数据放在网络结构中。MIND的分解内存体系结构中的网络结构本质上是一个CPU内存互连，集中放置在网络处理设备中的可编程网络交换机因此承担了MMU的角色，以实现高性能共享内存抽象。由于MIND以线路速率实现可编程硬件中内存管理的逻辑和元数据，因此延迟和带宽开销最小。

MIND采用了一个由所有进程共享的全局虚拟地址空间，在内存刀片上进行范围分区，以最大限度地减少需要存储在交换机ASIC片上内存中的地址转换项的数量。同时，它采用了一种物理内存分配机制，可在内存刀片服务器之间进行负载平衡分配，以实现高内存吞吐量。同时，通过将内存权限的存储与地址转换条目分离，实现细粒度和灵活的保护，并减少了开关ASIC的片上内存开销。MIND将基于目录的MSI一致性调整为网络内设置。为了减轻缓存一致性的网络开销，MIND利用以网络为中心的硬件原语，如交换机ASIC中的多播，以有效地实现其一致性协议。并且通过一种新的有界分割算法，动态调整内存区域大小，以限制交换机存储需求以及由于粗粒度跟踪内存区域导致的页面错误失效而增加的性能开销。

**3 研究进展**

**3.1 基于GAM的应用程序**

**3.1.1 事务引擎**

共享内存模型通过隐藏复杂的网络通信使分布式事务引擎更容易实现。开发人员只需关注核心的单节点事务处理逻辑。每个事务处理节点都有一个全局索引的根条目，其指针是全局地址，因此它可以通过使用读写API遍历全局索引来访问所有表。对于事务处理协议，GAM只是采用了传统的两阶段锁定（2PL），基于TryRLock/TryWLock API。GAM不依赖类似于两阶段提交（2PC）的协议来实现分布式一致性，因为在事务准备提交时，请求节点已经获取了数据。通过使用隐式发布MFence的锁同步原语，可以在GAM提供的PSO一致性模型下轻松实现可串行化。由于GAM提供的统一内存模型，事务引擎可以避免2PC开销，并且完全消除了应用层数据传输的复杂性。

**3.1.2 分布式哈希表**

基于GAM提供的共享内存模型，DHT可以实现为包裹在多个节点上的分布式存储bucket阵列。具体而言，每个GAM节点负责64位密钥空间的子集，并且密钥与其驻留节点之间的映射由最高位确定。在每个bucket中，都有多个12字节的条目和一个溢出指针，用于处理哈希冲突并提高哈希表的占用率。每个哈希表条目都包含一个从密钥的最低12位提取的12位标记，用于区分同一存储桶中的密钥，一个20位整数记录索引密钥-值对的大小，以及一个指向存储密钥-值对的全局地址的64位指针。这样，与散列表及其索引键值对在同一物理节点内并置的传统DHT不同，基于GAM的DHT实现将键值对与其索引项解耦，因此不仅能够降低DHT更新的成本（可以在本地重新分配，而不必考虑原始映射），还可以自动平衡GAM中节点的负载。

**3.2 远内存相关工作**

**3.2.1硬件资源分解**

数据中心中分解硬件资源的想法近年来越来越流行。因此，最近的工作考虑了如何调整数据中心的各种组件以支持分解，提出了新的硬件设计、操作系统、内存抽象和网络堆栈，并研究了对底层网络的要求。2018，Chung等[4]提出了可组合的系统设计，该系统提供了一个动态协同设计平台，允许在受控环境中进行实验和测量，加快了系统设计和软件开发；解耦了组件的生命周期，提供了灵活性，可以服务于各种工作负载。Shan等[5]提出了用于硬件资源分解的分布式操作系统LegoOS，在硬件级别和操作系统级别将处理器、内存和存储设备完全分离。CFM的调度策略和更快的交换机制也是对这些研究的补充。

**3.2.2 远内存访问**

过去的系统使用网络分页来利用远程内存。最近的工作，如HPBD[6]和Infiniswap[7]利用RDMA在网络上以较低的延迟实现交换。尽管Fastswap也通过RDMA实现交换，但它克服了一些限制这些现有系统交换延迟和吞吐量的挑战，如行首阻塞、RDMA操作中断、页面回收等。最近的另一种方法通过压缩冷页并将其本地存储在DRAM中来实现“远内存”[8]。通过这种方法，作者能够在DRAM中存储大约20%的压缩数据。然而，这种方法在满足日益增长的内存需求方面能力有限。结构连接内存建议在不分页的情况下使用远内存；然而，据我们所知，此技术目前还没有公开的硬件实现。

**3.2.3 集群调度**

现有的集群调度器，如Decima[9]、Tetris[10]等已经考虑了如何将作业打包到计算集群上，以便最大限度地高效利用集群资源，如内存、CPU、磁盘和网络。集群管理器，如Borg[11]、Omega[12]、Thread[13]和Mesos[14]大规模地跨机器（例如，数千台机器）调度作业，同时还解决故障和异构硬件等问题。但是，当作业的内存可以在本地和共享远程内存之间拆分时，这些方法都没有指定如何调度作业。将Fastswap的策略合并到调度器中，有望解决作业调度问题

**3.3分解数据中心的网络内存管理**

**3.3.1 可编程开关**

近年来，可编程交换机沿着两个协调一致的方向发展：灵活的网络交换机编程语言P4的开发，以及可编程交换机硬件的设计。这些交换机承载一个专用集成电路（ASIC），以及一个带有DRAM的通用CPU，如图1（左）所示。交换机ASIC包括入口管道、流量管理器和出口管道，它们按照该顺序处理分组。通过P4的可编程性通过可编程解析器和入口/出口管道中的匹配操作单元来实现，如图1（右）所示。具体来说，该程序定义解析器如何解析数据包头以提取一组字段，以及匹配动作单元的多个阶段（每个阶段都有有限的TCAM/SRAM和ALU）如何处理这些字段。通用CPU通过PCIe接口连接到交换机ASIC，并提供两个功能：（i）执行由于资源限制而无法在ASIC中执行的数据包处理，以及（ii）承载控制器功能，该控制器功能计算网络范围的策略并将其推送到交换机ASIC。

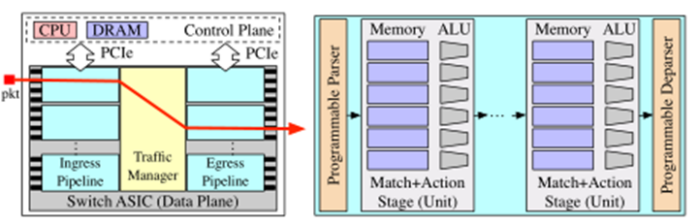


图1：（左）可编程交换机架构和（右）交换机入口/出口管道。

FPGA、定制ASIC甚至通用CPU同样可以用来实现MIND。虽然每种交换机都有不同的折衷方案（表2），但由于其性能、可用性、功耗和成本效率，Lee等[3]采用了RMT交换机。



表2：网络技术权衡

**3.3.2 分布式共享内存设计**

传统上，共享内存是在NUMA和分布式共享内存（DSM）体系结构的背景下探索的。在这种设计中，虚拟地址空间跨各个节点进行分区。每个分区都有一个主节点来管理其元数据，例如页表。每个节点还具有一个缓存，以提高频繁访问的内存块的性能。

由于块的副本可能驻留在多个节点缓存中，因此需要一致性协议，以确保每个节点在块的最新版本上运行。在流行的基于目录的失效协议，如MSI[15]中，每个内存块可以处于三种状态之一：修改（M），其中单个节点对该块具有独占读写访问权限（或“拥有”），共享（S），其中一个或多个缓存对该块具有共享只读访问权限，以及无效（I），其中块不存在于任何缓存中。目录跟踪每个块的状态，以及当前在其缓存中保存该块的节点列表（“共享列表”）。目录通常跨各个节点进行分区，每个主节点跟踪其自己的地址空间分区的目录条目。非本地块的存储器访问涉及联系该块的归属节点；它会在其他节点之间触发块的状态转换和潜在失效，然后从拥有该块的节点检索该块。

**4总结与展望**

内存需求的不断增长，使足够的主内存的可用性越来越成为大型计算集群的瓶颈，并对内存分解提出了透明的弹性、高资源利用率和硬件异构性的要求。本文主要论述了分布式共享内存、内存分解等数据中心内存管理技术，并详细论述了基于DSM的GAM、使用远内存的Linux交换系统Fastswap和网络内存管理单元MIND三种内存管理系统的原理、优势和研究进展。其中，GAM具有弹性可扩展性与高延迟低吞吐量；Fastswap高性能，但不可扩展；MIND可以在性能与最新内存分解系统相匹配的情况下，提供透明弹性，缺点是性能扩展不能保持线性。

对于内存分解研究现状，研究者们提出如下展望：

（1）在网络上计算。网络内存管理是实现网上计算的目标之一。其中，MIND可以利用NOPaxos来实现分解内存的一致复制。

（2）应用程序驱动的内存分解。主张操作系统向应用程序公开资源管理抽象，如内存放置和故障，以获得高性能和更好的容错性。

（3）新兴行业标准。高性能计算内存互连的行业标准从以服务器内设置为目标转变为以服务器间结构为目标。网络内存管理方法高性能互连技术的补充。

参 考 文 献

[1] Qingchao Cai, Wentian Guo, Hao Zhang, Divyakant Agrawal, Gang Chen, Beng Chin Ooi, Kian-Lee Tan, Yong Meng Teo, and Sheng Wang. Efficient Distributed Memory Management with RDMA and Caching. Proc. VLDB, 2018.

[2] Emmanuel Amaro, Christopher Branner-Augmon, Zhihong Luo, Amy Ousterhout, Marcos K. Aguilera, Aurojit Panda, Sylvia Ratnasamy, and Scott Shenker. Can Far Memory Improve Job Throughput? In Proc. EuroSys, 2020.

[3] Seung-seob Lee, Yanpeng Yu, Yupeng Tang, Anurag Khandelwal, Lin Zhong, and Abhishek Bhattacharjee. 2021. MIND: In-Network Memory Management for Disaggregated Data Centers. In Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles (SOSP '21). Association for Computing Machinery, New York, NY, USA, 488–504. DOI:https://doi.org/10.1145/3477132.3483561

[4] I-Hsin Chung, Bulent Abali, and Paul Crumley. 2018. Towards a Composable Computer System. In International Conference on High Performance Computing in Asia-Pacific Region (HPC Asia 2018). 137–147. https://doi.org/10.1145/3149457.3149466

[5] Yizhou Shan, Yutong Huang, Yilun Chen, and Yiying Zhang. 2018. LegoOS: A Disseminated, Distributed OS for Hardware Resource Disaggregation. In Symposium on Operating Systems Design and Implementation (OSDI’18). 69–87.

[6] Shuang Liang, Ranjit Noronha, and Dhabaleswar K. Panda. 2005. Swapping to Remote Memory over InfiniBand: An Approach using a High Performance Network Block Device. In IEEE International Conference on Cluster Computing.

[7] Juncheng Gu, Youngmoon Lee, Yiwen Zhang, Mosharaf Chowdhury, and Kang G. Shin. 2017. Efficient Memory Disaggregation with INFINISWAP. In Symposium on Networked Systems Design and Implementation (NSDI’17). 649–667.

[8] Andres Lagar-Cavilla, Junwhan Ahn, Suleiman Souhlal, Neha Agarwal, Radoslaw Burny, Shakeel Butt, Jichuan Chang, Ashwin Chaugule, Nan Deng, Junaid Shahid, Greg Thelen, Kamil Adam Yurtsever, Yu Zhao, and Parthasarathy Ranganathan. 2019. Software-defined Far Memory in Warehouse-scale Computers. In International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS’19). 317–330. https://doi.org/10.1145/3297858.3304053

[9] Hongzi Mao, Malte Schwarzkopf, Shaileshh Bojja Venkatakrishnan, Zili Meng, and Mohammad Alizadeh. 2019. Learning Scheduling Algorithms for Data Processing Clusters. In ACM Special Interest Group on Data Communications (SIGCOMM’19). 270–288. https://doi.org/10.1145/3341302.3342080

[10] Robert Grandl, Ganesh Ananthanarayanan, Srikanth Kandula, Sriram Rao, and Aditya Akella. 2014. Multi-resource Packing for Cluster Schedulers. In ACM Special Interest Group on Data Communications (SIGCOMM’14). 455–466. https://doi.org/10.1145/2619239.2626334

[11] Abhishek Verma, Luis Pedrosa, Madhukar Korupolu, David Oppenheimer, Eric Tune, and John Wilkes. 2015. Large-scale Cluster Management at Google with Borg. In European Conference on Computer Systems (EuroSys’15). https://doi.org/10.1145/2741948.2741964

[12] Yizhou Shan, Yutong Huang, Yilun Chen, and Yiying Zhang. 2018. LegoOS: A Disseminated, Distributed OS for Hardware Resource Disaggregation. In Symposium on Operating Systems Design and Implementation (OSDI’18). 69–87.

[13] MSI Protocol. https://en.wikipedia.org/wiki/MSI\_ protocol.

[[1]](#footnote-1)

1. [↑](#footnote-ref-1)