arXiv:2301.09674v1 [cs.AR] 23 Jan 2023

非聚合系统中高效数据移动的架构支持

Christina Giannoula[§†] *Kailong Huang[§] *Jonathan Tang[§]
NectariosKoziris[†] Georgios Goumas[†] ZeshanChishti[‡] Nandita Vijaykumar[§§多伦多大学 †雅典国立理工大学 ‡英特尔公司

1. 非聚合系统中的数据移动

传统数据中心包含单体服务器，它们将中央处理器、内存和磁盘紧密集成在一起（图 1a）。相反，非聚合系统（**分解存储系统**）[1]–[3]通过高带宽网络将多个计算（**缓存计算**）、内存（**内存控制器**）和存储设备组织为相互独立且故障隔离的 组件（图 1b）。在现代数据中心中，分解存储系统能够通过提高资源利用率、实现资源扩展、故障处理和弹性，大幅降低数据中心成本 [1]–[5]。

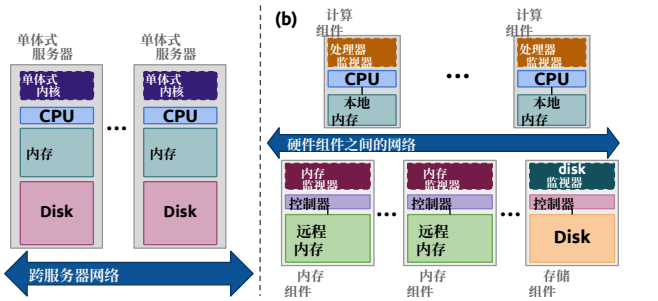

图 1. (a) 传统系统与 (b) 分解存储系统。

内存控制器提供大量主存（远程内存），而 CC 包含本地内存中的片上缓存和数 GB 动态随机存取存储器，这些资源充当远程内存的缓存。在此背景下，应用程序的大部分数据（～ 80%）[1], [3], [6] 位于远程内存中，通过网络远程访问这些数据会导致较大的性能损失（见图2）。

在分解存储系统中，降低数据访问开销颇具挑战，原因如下。首先，分解存储系统并非单体式，而是由独立管理的实体组成：每个组件都有自己的硬件控制器，专用的内核监控器则利用自身功能管理其所在的组件（只有在需要访问远程资源时，才通过网络消息与其他监控器通信）。这一特性要求采用分布式、解耦的解决方案，以便扩展到系统中的大量独立组件。其次，远程内存访问延迟存在很大差异，因为它取决于内存控制器的位置、共享同一网络和内存控制器的其他作业所带来的争用，以及运行时或多次执行之间可能发生变化的数据放置。这就要求解决方案能够应对网络带宽和远程内存带宽及延迟的波动。第三，导致性能下降的一个主要因素，是分解存储系统中常用的按页粒度移动数据的做法 [1], [3], [4], [6] 。这种做法有效地实现了软件透明性，降低了内存管理中的元数据开销，并在许多应用程序中提供了较高的空间局部性。然而，这会导致较高的带宽消耗和网络拥塞，并且往往会显著拖慢对其他并发访问页中关键路径缓存行的访问。

图2展示了使用分解存储系统对一组具有代表性的异构工作负载采用不同数据移动方案的性能分析。我们评估了内存控制器和缓存计算，其本地内存 可容纳 ～20% 的应用数据，并将网络带宽设为总线带宽的 1/2-1/8× [1],[6]。所有方案的性能均以单体式方案为基准进行归一化；在该方案中，所有 数据都能放入缓存计算的本地内存。我们有三项观察。

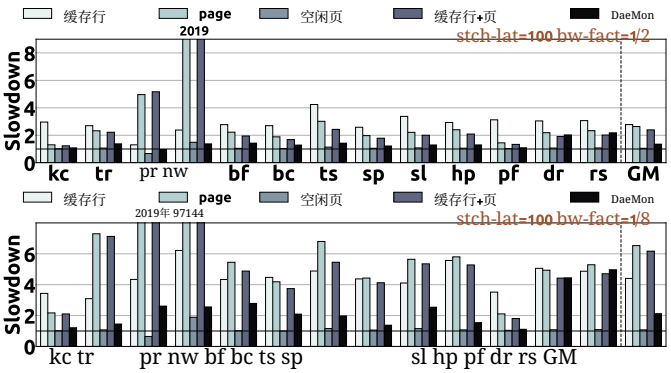

图2. 分解存储系统中的数据移动开销。

首先，通常采用的页方案以页粒度移动数据，由于通过网络传输大量数据会拖慢对其他页的访问，因此相比单体本地配置会造成较大的性能下降。相反，当页可以免费移动，即采用空闲页方案时，得益于页内的空间局部性，性能会显著提升。其次，始终以固定粒度移动数据（通过末级缓存按缓存行粒度，或通过页粒度的本地内存）无法适应不同应用程序和网络配置：某些应用程序适合按缓存行粒度（例如*pr*、*nw*）或页粒度（例如*pf*、*dr*）移动数据，而性能最佳的粒度也取决于网络特性（例如*bf*、*ts*）。第三，简单地同时按两种粒度移动数据（缓存行+页），以便用最先到达的数据包的延迟为缓存计算（CC）提供数据请求服务，仍然相当低效，因为关键的缓存行请求仍会排在同时访问的大页之后。相比之下，图2表明，*DaeMon* （见§3）能够针对各种网络和应用程序特性，显著降低分解存储系统中的数据移动开销。

2. 先前工作

先前工作 [1]–[4], [6]–[26] 提出了面向分解存储系统的操作系统内核、系统级解决方案、软件管理系统和体系结构

*** Kailong Huang 和 Jonathan Tang 贡献相同。**

这些工作没有解决分解存储系统中的数据移动难题，因此我们的工作与它们相互正交。

混合系统的先前工作 [27]–[39] 将堆叠式动态随机存取存储器 [40] 集成为大型主内存的动态随机存取存储器缓存 [27], [29], [32]，用于单体服务器，并通过页面放置／热点页面选择方案，或以更小的粒度（例如缓存行）移动数据，来应对双层物理内存中高昂的页面迁移成本。然而，分解存储系统中的数据移动带来了截然不同的挑战。首先，跨网络访问的速度远低于服务器内部访问，因此，仅靠智能页面放置／迁移无法解决这些高昂的成本。其次，分解存储系统的访问延迟会因当前网络架构以及共享内存控制器／网络的并发作业而显著变化，因此需要一种主要针对这种变化而设计、具备鲁棒性的解决方案。最后，分解存储系统包含由独立缓存计算分别管理的内存控制器和网络，这些缓存计算运行着未知作业。因此，与混合系统不同，该解决方案不能假设内存控制器侧的内存管理完全由中央处理器侧控制。我们的工作首次研究了完全分解存储系统中的数据移动问题，并为分解存储系统设计了一种有效的解决方案。

3. *DaeMon* 的核心思想

DaeMon （图 3）是一种自适应且可扩展的机制，由三项技术组成，旨在降低分解存储系统中的数据开销。

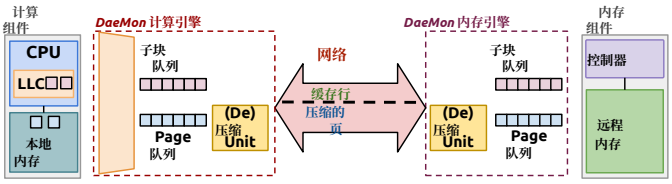

图 3. *DaeMon* 的高层概览。

(I) 解耦式多粒度数据迁移。我们集成两个独立的硬件队列，以两种粒度处理来自远程内存 的数据请求：缓存行粒度（通过子块队列 传输至末级缓存）和页粒度（通过页队列 传输至本地内存）。此外，我们采用带宽划分方法，有效地优先移动缓存行而非页面：队列控制器以高于页面请求的预定义固定速率 处理缓存行请求，确保在任意时刻，始终有一定比例的带宽资源分配给缓存行移动，以便快速完成移动。这项关键技术能够：（i）通过保留页面迁移，降低元数据开销；（ii）利用页内数据局部性，提升性能；（iii）减少此前可能已触发的高开销页面移动对关键路径上缓存行数据移动造成的减速。

（II）选择粒度数据迁移。为提供自适应的数据移动解决方案，我们在每个缓存计算（CC）中都配备两个独立的硬件缓冲区，用于跟踪待处理的数据迁移，分别以缓存行粒度（通过在途子块缓冲区）和页粒度（通过在途页面缓冲区）进行跟踪；此外，还配备一个粒度选择单元，根据在途缓冲区的利用率，决定数据请求应由缓存行、页还是两者处理。鉴于*DaeMon* 优先进行缓存行迁移，而非页面迁移，这些在途缓冲区的使用方式各不相同，使我们能够在运行时捕捉应用程序行为和系统负载。例如：(a) 如果页之间的局部性较低页缓冲区的利用率会高于子块缓冲区（缓存行具有优先级），因此选择单元会倾向于迁移缓存行，并限制页迁移（反之亦然）。(b) 在带宽利用率较低的情况下，页缓冲区利用率较低，因此选择单元会安排更多页面迁移（或同时采用两种粒度），以获得数据局部性方面的收益（反之亦然）。

（III）页面迁移中的链路压缩。我们在 CC 和内存控制器两端都采用硬件压缩单元，对通过网络迁移的页进行高效压缩。链路压缩可减少页面迁移过程中的网络带宽消耗，并缓解网络瓶颈。

三种技术的协同作用。*DaeMon*通过协同整合这三种技术，显著降低数据移动开销，并能适应网络、架构和应用特征：

(1) 优先处理请求的缓存行，有助于*DaeMon* 容忍通过网络进行页面迁移时较高的（解）压缩延迟，同时发挥页面迁移的优势（元数据开销低，页具有空间局部性）。

(2) 压缩页占用的网络带宽较少，使*DaeMon* 能够预留部分带宽，以有效优先处理关键路径上的缓存行访问。

(3) 页面迁移时进行压缩，有助于*DaeMon* 根据数据的可压缩性进行调整：如果页高度可压缩，在途页面缓冲区就会更快清空，此时*DaeMon*会倾向于发送更多页（反之亦然）。

4. *DaeMon* 的关键成果

我们扩展 Sniper [41]以模拟分解存储系统，并评估图处理、高性能计算、数据分析、生物信息学和机器学习领域的容量密集型工作负载。与广泛采用的按页粒度移动数据的方法相比，*DaeMon*分别降低了数据访问成本并提升了性能，幅度为 3.06× 和 2.39×。

DaeMon 结合了这三种技术的优势，在保持页粒度移动所带来的稳健性、空间局部性、透明性和元数据管理优势的同时。我们展示了，与广泛采用的页粒度数据移动方法相比，*DaeMon* 在各种网络/架构配置和应用程序（图4上部）以及数据中心中多个并发运行的作业（图4下部）中，均实现了出色的系统性能。

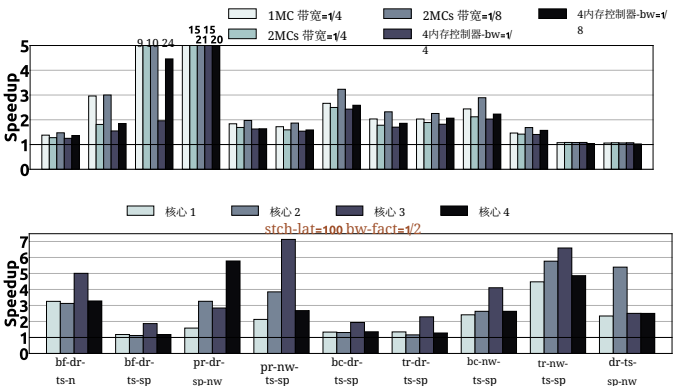

图 4. *DaeMon*相较于页 方案的优势：（i）改变内存控制器、网络带宽和应用程序；（ii）在 4核 CPU CC 和内存控制器中运行多个应用程序。

参考文献

[1] Yizhou Shan、Yutong Huang、Yilun Chen 和 Yiying Zhang。《Lego OS：一种用于硬件资源解聚的分布式操作系统》。发表于 OSDI，2018年。[2] Zhiyuan Guo、Yizhou Shan、Xuhao Luo、Yutong Huang 和 Yiying Zhang。《Clio：软硬件协同设计的解聚内存系统》。发表于 ASPLOS，2022年。[3] Seung-seob Lee、Yanpeng Yu、Yupeng Tang、Anurag Khandelwal、Lin Zhong 和 Abhishek Bhattacharjee。《MIND：面向解聚数据中心的网络内存管理》。发表于 SOSP，2021年。[4] Irina Calciu、M. Talha Imran、Ivan Puddu、Sanidhya Kashyap、Hasan Al Maruf、Onur Mutlu 和 Aasheesh Kolli。《重新思考解聚内存的软件运行时》。发表于 ASPLOS，2021年。[5] Christina Giannoula。《通过高效同步与数据访问技术加速非规则应用程序》。发表于 arXiv，2022年。[6] Peter X. Gao、Akshay Narayan、Sagar Karandikar、Joao Carreira、Sangjin Han、Rachit Agarwal、Sylvia Ratnasamy 和 Scott Shenker。《资源解聚的网络需求》。发表于 OSDI，2016年。[7] Sebastian Angel、Mihir Nanavati 和 Siddhartha Sen。《解聚与应用程序》。发表于 HotCloud，2020年。[8] Vlad Nitu、Boris Teabe、Alain Tchana、Canturk Isci 和 Daniel Hagimont。《欢迎来到 Zombieland：数据中心中实用且节能的内存解聚》。发表于 EuroSys，2018年。[9] Sangjin Han、Norbert Egi、Aurojit Panda、Sylvia Ratnasamy、Guangyu Shi 和 Scott Shenker。《面向下一代数据中心的资源解聚网络支持》。发表于 HotNets，2013年。[10] Marcos K. Aguilera、Nadav Amit、Irina Calciu、Xavier Deguillard、Jayneel Gandhi、Pratap Subrahmanyam、Lalith Suresh、Kiran Tati、Rajesh Venkatasubramanian 和 Michael Wei。《高速网络时代的远程内存》。发表于 SoCC，2017年。[11] Qizhen Zhang、Yifan Cai、Sebastian G. Angel、Vincent Liu、Ang Chen 和 B. T. Loo。《重新思考面向解聚数据中心的数据管理系统》。发表于 CIDR，2020年。[12] Kevin Lim、Yoshio Turner、Jose Renato Santos、Alvin AuYoung、Jichuan Chang、Parthasarathy Ranganathan 和 Thomas F. Wenisch。《解聚内存的系统级影响》。发表于 HPCA，2012年。[13] Chenxi Wang、Haoran Ma、Shi Liu、Yuanqi Li、Zhenyuan Ruan、Khanh Nguyen、Michael D. Bond、Ravi Netravali、Miryung Kim 和 Guoqing Harry Xu。《Semeru：内存解聚的托管运行时》。发表于 OSDI，2020年。[14] Pengfei Zuo、Jiazhao Sun、Liu Yang、Shuangwu Zhang 和 Yu Hua。《面向解聚内存的单边 RDMA 感知型可扩展哈希》。发表于 ATC，2021年。[15] Laurent Bindschaedler、Ashvin Goel 和 Willy Zwaenepoel。《Hailstorm：面向分布式基于LSM的数据库的解聚计算与存储》。发表于 ASPLOS，2020年。[16] Ivy Peng、Roger Pearce 和 Maya Gokhale。《论内存利用不足：探索高性能计算系统中的解聚内存》。发表于 SBAC-PAD，2020年。[17] 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 和 Parthasarathy Ranganathan。《仓库级计算机中的软件定义远程内存》。发表于 ASPLOS，2019年。[18] Juncheng Gu、Youngmoon Lee、Yiwen Zhang、Mosharaf Chowdhury 和 Kang G. Shin。《借助 Infiniswap 实现高效内存解聚》。发表于 NSDI，2017年。[19] Marcos K. Aguilera、Nadav Amit、Irina Calciu、Xavier Deguillard、Jayneel Gandhi、Stanko Novakovi´c、Arun Ramanathan、Pratap Subrahmanyam、Lalith Suresh、Kiran Tati、Rajesh Venkatasubramanian 和 Michael Wei。《Remote Regions：远程内存的简单抽象》。发表于 ATC，2018年。[20] Christian Pinto、Dimitris Syrivelis、Michele Gazzetti、Panos Koutsovasilis、Andrea Reale、Kostas Katrinis 和 H. Peter Hofstee。《Thymes-isFlow：面向机架级内存解聚的软件定义、软硬件协同设计互连栈》。发表于 MICRO，2020年。[21] K. Katrinis、D. Syrivelis、D. Pnevmatikatos、G. Zervas、D. Theodoropoulos、I. Koutsopoulos、K. Hasharoni、D. Raho、C. Pinto、F. Espina、S. Lopez-Buedo、Q. Chen、M. Nemirovsky、D. Roca、H. Klos 和 T. Berends。《机架级解聚云数据中心：dReD-Box 项目愿景》。发表于 DATE，2016年。[22] Dhantu Buragohain、Abhishek Ghogare、Trishal Patel、Mythili Vutukuru 和 Purushottam Kulkarni。《DiME：解聚内存体系结构的性能模拟器》。发表于 APSys，2017年。[23] Pramod Subba Rao 和 George Porter。《内存解聚可行吗？以 Spark SQL 为例的案例研究》。发表于 ANCS，2016年。[24] Georgios Zervas、Hui Yuan、Arsalan Saljoghei、Qianqiao Chen 和 Vaibhawa Mishra。《采用最少远程内存延迟：技术、体系结构与资源分配。*JOCN*，2018年。[25] Donghyun Gouk、Sangwon Lee、Miryeong Kwon 和 Myoungsoo Jung。直接访问：借助 DirectCXL 实现高性能内存解聚。发表于*ATC*，2022年。[26] Yang Zhou、Hassan M. G. Wassel、Sihang Liu、Jiaqi Gao、James Mickens、Minlan Yu、Chris Kennelly、Paul Turner、David E. Culler、Henry M. Levy 和 Amin Vahdat。Carbink：容错远端内存。发表于*OSDI*，2022年。[27] Xiangyu Dong、Yuan Xie、Naveen Muralimanohar 和 Norman P. Jouppi。简单而有效的异构主存：支持片上内存控制器。发表于*SC*，2010年。[28] Haikun Liu、Yujie Chen、Xiaofei Liao、Hai Jin、Bingsheng He、Long Zheng 和 Rentong Guo。面向混合 DRAM/NVM 内存架构的软硬件协同缓存。发表于*ICS*，2017年。[29] Xiaowei Jiang、Niti Madan、Li Zhao、Mike Upton、Ravishankar Iyer、Srihari Makineni、Donald Newell、Yan Solihin 和 Rajeev Balasubramonian。CHOP：面向 CMP服务器平台的自适应滤波器式DRAM缓存。发表于*HPCA*，2010年。[30] Jagadish B. Kotra、Haibo Zhang、Alaa R. Alameldeen、Chris Wilkerson 和 Mahmut T. Kandemir。CHAMELEON：动态可重构的异构内存系统。发表于*MICRO*，2018年。[31] Andreas Prodromou、Mitesh Meswani、Nuwan Jayasena、Gabriel Loh 和 Dean M. Tullsen。MemPod：用于平坦地址空间多级内存中高效、可扩展迁移的集群式架构。发表于*HPCA*，2017年。[32] Neha Agarwal 和 Thomas F. Wenisch。Thermostat：应用透明的双层主存页管理。发表于*ASPLOS*，2017年。[33] Mitesh R. Meswani、Sergey Blagodurov、David Roberts、John Slice、Mike Ignatowski 和 Gabriel H. Loh。异构内存体系结构：混合堆叠式 DRAM 与封装外内存的软硬件协同方法。发表于*HPCA*，2015年。[34] Kai Wu、Yingchao Huang 和 Dong Li。Unimem：基于非易失性内存的异构主存运行时数据管理。发表于*SC*，2017年。[35] Gabriel H. Loh 和 Mark D. Hill。为超大堆叠式 DRAM 缓存高效支持传统块大小。发表于*MICRO*，2011年。[36] Gabriel Loh 和 Mark D. Hill。通过复合访问调度和 MissMap 支持超大 DRAM 缓存。*IEEE Micro*，2012年。[37] Chia Chen Chou、Aamer Jaleel 和 Moinuddin K. Qureshi。CAMEO：兼具主存容量与硬件管理缓存灵活性的两级内存组织。发表于*MICRO*，2014年。[38] Jee Ho Ryoo、Mitesh R. Meswani、Andreas Prodromou 和 Lizy K. John。Silc-fm：子块交错式类缓存平坦内存组织。发表于*HPCA*，2017年。[39] Djordje Jevdjic、Stavros Volos 和 Babak Falsafi。面向服务器的芯片堆叠DRAM缓存：命中率、延迟还是带宽？Footprint Cache 助您兼得。发表于*ISCA*，2013年。[40] Hongshin Jun、Jinhee Cho、Kangseol Lee、Ho-Young Son、Kwiwook Kim、Hanho Jin 和 Keith Kim。HBM DRAM技术与架构。发表于*IMW*，2017年。[41] Trevor E. Carlson、Wim Heirman 和 Lieven Eeckhout。Sniper：探索可扩展且精确的并行多核模拟的抽象级别。发表于*SC*，2011年。