Architectural Support for Efficient Data Movement in Fully Disaggregated Systems

Christina Giannoula
University of Toronto & National Technical University of Athens
christina.giann@gmail.com

Kailong Huang*
University of Toronto
kailong9@gmail.com

Jonathan Tang*
University of Toronto
jonathanj.tang@alum.utoronto.ca

Nectarios Koziris
National Technical University of Athens
nkoziris@cslab.ece.ntua.gr

Georgios Goumas
National Technical University of Athens
goumas@cslab.ece.ntua.gr

Zeshan Chishti
Intel Corporation
zeshan.a.chishti@intel.com

Nandita Vijaykumar
University of Toronto
nandita@cs.toronto.edu

1 DATA MOVEMENT IN DISAGGREGATED SYSTEMS

Traditional data centers include monolithic servers that tightly integrate CPU, memory and disk (Figure 1a). Instead, *Disaggregated Systems* (**DSs**) [8, 13, 18, 27] organize multiple compute (**CC**), memory (**MC**) and storage devices as *independent, failure-isolated* components interconnected over a high-bandwidth network (Figure 1b). DSs can greatly reduce data center costs by providing improved resource utilization, resource scaling, failure-handling and elasticity in modern data centers [5, 8–10, 10, 11, 13, 18, 27].

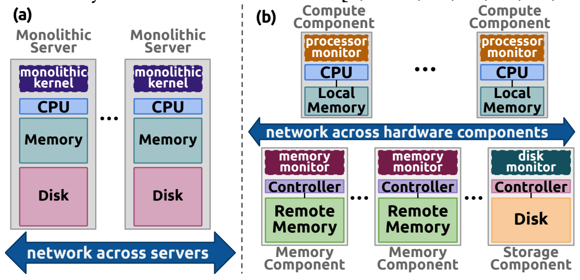

Figure 1: (a) Traditional systems vs (b) DSs.

The MCs provide large pools of main memory (***remote memory***), while the CCs include the on-chip caches and a few GBs of DRAM (***local memory***) that acts as a cache of *remote memory*. In this context, a large fraction of the application's data (~ 80%) [8, 18, 27] is located in *remote memory*, and can cause large performance penalties from remotely accessing data over the network.

Alleviating data access overheads is challenging in DSs for the following reasons. First, DSs are not monolithic and comprise independently managed entities: each component has its own hardware controller, and a specialized kernel monitor uses its own functionality to manage the component it runs on (only communicates with other monitors via network messaging if there is a need to access remote resources). This characteristic necessitates a distributed and disaggregated solution that can scale to a large number of independent components in the system. Second, there is high variability in remote memory access latencies since they depend on the locations of the MCs, contention from other jobs that share the same network and MCs, and data placements that can vary during runtime or between multiple executions. This necessitates a solution that is robust towards fluctuations in the network/remote memory bandwidth and latencies. Third, a major factor behind the performance slowdowns is the commonly-used approach in DSs [5, 8, 18, 27] of moving data at page granularity. This approach effectively provides software transparency, low metadata costs in memory management, and high spatial locality in many applications. However, it can cause high bandwidth consumption and network congestion, and often significantly slows down accesses to critical path cache lines in other concurrently accessed pages.

* **Kailong Huang and Jonathan Tang have equal contribution.**

SIGMETRICS '23 Abstracts, June 19–23, 2023, Orlando, FL, USA
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0074-3/23/06.
https://doi.org/10.1145/3578338.3593533

2 PRIOR WORK

Prior works [2–5, 8, 12–14, 18, 19, 24, 27, 28, 30] propose OS kernels, system-level solutions, software management systems, architectures for DSs. These works do not tackle the data movement challenge in DSs, and thus our work is orthogonal to them.

Prior works on hybrid systems [1, 6, 7, 15, 17, 20–23, 25, 26, 29] integrate die-stacked DRAM [16] as DRAM cache of a large main memory [1, 7, 15] in a monolithic server, and tackle high page movement costs in two-tiered physical memory via page placement/hot page selection schemes or by moving data at smaller granularity, e.g., cache line. However, data movement in DSs poses fundamentally different challenges. First, accesses across the network are significantly slower than within the server, thus intelligent page placement cannot by itself address these high costs. Second, DSs incur significant variations in access latencies based on the current network architecture and concurrent jobs sharing the MCs/network, thus necessitating an solution primarily designed for robustness to this variability. Finally, DSs include independently managed MCs and networks shared by independent CCs running unknown jobs. Thus, unlike hybrid systems, the solution cannot assume that the memory management at the MCs can be fully controlled by the CPU side. Our work is the first to examine the data movement problem in fully DSs, and design an effective solution for DSs.

3 DAEMON'S KEY IDEAS

DaeMon (Figure 2) is an adaptive and scalable mechanism to alleviate data costs in DSs, consisting of three techniques.

(I) Decoupled Multiple Granularity Data Movement. We integrate two separate hardware queues to serve data requests from *remote memory* at two granularities, i.e., cache line (via the *sub-block queue* to LLC) and page (via the *page queue* to *local memory*)

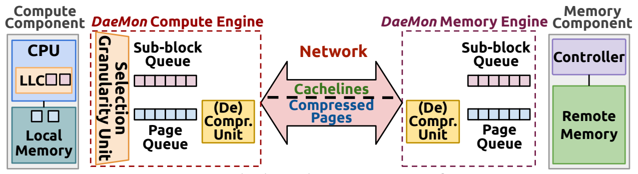

Figure 2: High-level overview of *DaeMon*.

granularity, and effectively *prioritize* moving cache lines over moving pages via a *bandwidth partitioning* approach: a queue controller serves cache line requests with a higher *predefined fixed rate* than page requests to ensure that any given time a certain fraction of the bandwidth resources is always allocated to serve cache line moves fast. This key technique enables (i) low metadata overheads by retaining page migrations, (ii) high performance by leveraging data locality within pages, and (iii) fewer slowdowns in cache line data movements that are on the critical path, from expensive page moves that may have been previously triggered.

(II) Selection Granularity Data Movement. To provide an adaptive data movement solution, we include in each CC two separate hardware buffers to track pending data migrations for both cache line (via the *inflight sub-block buffer*) and page (via the *inflight page buffer*) granularity, and a selection granularity unit to decide if a data request should be served by cache line, page or *both* based on the utilization of the *inflight* buffers. Given that *DaeMon* prioritizes cache line over page moves, the inflight buffers are utilized in different ways, allowing us to capture the application behavior and the system load during runtime. For example: (a) If there is *low locality* within pages, the page buffer has higher utilization than the sub-block buffer (cache lines are prioritized), thus the selection unit favors moving cache lines and throttles pages (and vice-versa). (b) Under low *bandwidth utilization* scenarios, the page buffer utilization is low, thus the selection unit schedules more page movements (or both granularities) to obtain data locality benefits (and vice-versa).

(III) Link Compression on Page Movements. We employ hardware compression units at both the CCs and MCs to highly compress pages migrated over the network. *Link compression* on page moves reduces the network bandwidth consumption and alleviates network bottlenecks.

Synergy of Three Techniques. *DaeMon* cooperatively integrates all three techniques to significantly alleviate data movement overheads, and provide robustness towards network, architecture and application characteristics:

(1) Prioritizing requested cache lines helps *DaeMon* to tolerate high (de)compression latencies in page migrations over the network, while also leveraging benefits of page migrations (low metadata costs, spatial locality in pages).

(2) Compressed pages consume less network bandwidth, enabling *DaeMon* to reserve part of the bandwidth to effectively prioritize critical path cache line accesses.

(3) Compression on page moves helps *DaeMon* to adapt to the data compressibility: if the pages are highly compressible, the inflight page buffer empties at a faster rate, and *DaeMon* favors sending more pages (and vice-versa).

4 DAEMON'S KEY RESULTS

We evaluate *DaeMon* using a range of capacity intensive workloads with different memory access patterns from machine learning, high-performance-computing, graph processing, and bioinformatics domains. Over the widely-adopted approach of moving data at page granularity, *DaeMon* decreases memory access latencies by 3.06× on average, and improves system performance by 2.39× on average. We demonstrate that *DaeMon* provides (i) robustness and significant performance benefits on various network/architecture configurations and application behavior, (ii) scalability to *multiple* hardware components and networks, and (iii) adaptivity to dynamic workload demands, even when multiple heterogeneous jobs are concurrently executed in the DS.

ACM Reference Format:
Christina Giannoula, Kailong Huang, Jonathan Tang, Nectarios Koziris, Georgios Goumas, Zeshan Chishti, and Nandita Vijaykumar. 2023. Architectural Support for Efficient Data Movement in Fully Disaggregated Systems. In *Abstract Proceedings of the 2023 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '23 Abstracts), June 19–23, 2023, Orlando, FL, USA.* 2 pages. https://doi.org/10.1145/3578338.3593533

References

[1] Neha Agarwal Thomas Wenisch. 2017. Thermostat: Application-Transparent Page Management for Two-Tiered Main Memory. In *ASPLOS*.
[2] Marcos K. Aguilera, et. al. 2018. Remote Regions: A Simple Abstraction for Remote Memory. In *ATC*.
[3] Marcos K. Aguilera, et. al. 2017. Remote Memory in the Age of Fast Networks. In *SoCC*.
[4] Sebastian Angel, et. al. 2020. Disaggregation and the Application. In *HotCloud*.
[5] Irina Calciu, et. al. 2021. Rethinking Software Runtimes for Disaggregated Memory. In *ASPLOS*.
[6] Chia Chen Chou, et. al. 2014. CAMEO: A Two-Level Memory Organization with Capacity of Main Memory and Flexibility of Hardware-Managed Cache. In *MICRO*.
[7] Xiangyu Dong, et. al. 2010. Simple but Effective Heterogeneous Main Memory with On-Chip Memory Controller Support. In *SC*.
[8] Peter X. Gao, et. al. 2016. Network Requirements for Resource Disaggregation. In *OSDI*.
[9] Christina Giannoula. 2022. Accelerating Irregular Applications via Efficient Synchronization and Data Access Techniques. In *arXiv*. https://arxiv.org/abs/2211.05908
[10] Christina Giannoula, et. al. 2023. DaeMon: Architectural Support for Efficient Data Movement in Disaggregated Systems. In *arXiv*.
[11] Christina Giannoula, et. al. 2023. DaeMon: Architectural Support for Efficient Data Movement in Fully Disaggregated Systems. In *Proc. ACM Meas. Anal. Comput. Syst.*
[12] Juncheng Gu, et. al. 2017. Efficient Memory Disaggregation with Infiniswap. In *NSDI*.
[13] Zhiyuan Guo, et. al. 2022. Clio: A Hardware-Software Co-Designed Disaggregated Memory System. In *ASPLOS*.
[14] Sangjin Han, et. al. 2013. Network Support for Resource Disaggregation in Next-Generation Datacenters. In *HotNets*.
[15] Xiaowei Jiang, et. al. 2010. CHOP: Adaptive Filter-Based DRAM Caching for CMP Server Platforms. In *HPCA*.
[16] Hongshin Jun, et. al. 2017. HBM DRAM Technology and Architecture. In *IMW*.
[17] Jagadish B. Kotra, et. al. 2018. CHAMELEON: A Dynamically Reconfigurable Heterogeneous Memory System. In *MICRO*.
[18] Seung-seob Lee, et. al. 2021. MIND: In-Network Memory Management for Disaggregated Data Centers. In *SOSP*.
[19] Kevin Lim, et. al. 2012. System-Level Implications of Disaggregated Memory. In *HPCA*.
[20] Haikun Liu, et. al. 2017. Hardware/Software Cooperative Caching for Hybrid DRAM/NVM Memory Architectures. In *ICS*.
[21] Gabriel Loh Mark D. Hill. 2012. Supporting Very Large DRAM Caches with Compound-Access Scheduling and MissMap. In *IEEE Micro*.
[22] Gabriel H. Loh Mark D. Hill. 2011. Efficiently Enabling Conventional Block Sizes for Very Large Die-Stacked DRAM Caches. In *MICRO*.
[23] Mitesh R. Meswani, et. al. 2015. Heterogeneous Memory Architectures: A HW/SW Approach for Mixing Die-Stacked and Off-Package Memories. In *HPCA*.
[24] Christian Pinto, et. al. 2020. ThymesisFlow: A Software-Defined, HW/SW co-Designed Interconnect Stack for Rack-Scale Memory Disaggregation. In *MICRO*.
[25] Andreas Prodromou, et. al. 2017. MemPod: A Clustered Architecture for Efficient and Scalable Migration in Flat Address Space Multi-level Memories. In *HPCA*.
[26] Jee Ho Ryoo, et. al. 2017. SILC-FM: Subblocked InterLeaved Cache-Like Flat Memory Organization. In *HPCA*.
[27] Yizhou Shan, et. al. 2018. LegoOS: A Disseminated, Distributed OS for Hardware Resource Disaggregation. In *OSDI*.
[28] Chenxi Wang, et. al. 2020. Semeru: A Memory-Disaggregated Managed Runtime. In *OSDI*.
[29] Kai Wu, et. al. 2017. Unimem: Runtime Data Managementon Non-Volatile Memory-Based Heterogeneous Main Memory. In *SC*.
[30] Qizhen Zhang, et. al. 2020. Rethinking Data Management Systems for Disaggregated Data Centers. In *CIDR*.