RapidSwap: A Hierarchical Far Memory

Hyunik Kim[0000−0001−7740−8344], Changyeon Jo[0000−0002−9707−1256], Jörn Altmann[0000−0002−8880−9546], and Bernhard Egger ✉[0000−0002−6645−6161]

Seoul National University, Seoul, Korea
hyunik@csap.snu.ac.kr, changyeon@csap.snu.ac.kr,
jorn.altman@acm.org, bernhard@csap.snu.ac.kr

Abstract. As more and more memory-intensive applications are moved into the cloud, data center operators face the challenge of providing sufficient main memory resources while achieving high resource utilization. Solutions to overcome the unsatisfying performance degradation of traditional on-demand paging include memory disaggregation that allows applications to access remote memory or compressing memory pages in local DRAM; however, the former's extended failure domain and the latter's low efficacy limit their broad applicability. This paper presents RapidSwap, a hierarchical far memory manager that exploits the wide availability of phase-change memory (Intel Optane memory) in data centers to achieve quasi-DRAM performance at a significantly lower total cost of ownership (TCO). RapidSwap migrates infrequently accessed data to slower and cheaper devices in a hierarchy of storage devices by tracking applications' memory accesses. Evaluated with several real-world cloud benchmarks, RapidSwap achieves a reduction of 20% in operating cost at minimal performance degradation and is 30% more cost-effective than pure DRAM solutions. The results demonstrate that proper management of new memory technologies can yield significant TCO savings in cloud data centers.

Keywords: Memory hierarchy · Far memory · Cloud Datacenter.

1 Introduction

Over the past two decades, big data and artificial intelligence techniques have been adopted by numerous application domains. A common characteristic of such workloads is their need for large amounts of main memory to process the big data sets [20]. As these workloads are moved into the cloud, cloud service providers have started to offer virtual machine (VM) instances optimized for such memory-intensive workloads. Amazon, Google Cloud, and Microsoft Azure, for example, support VM instances with up to 24 terabytes of main memory [5,6,17]. Ideally, data center operators would equip their machines with sufficient DRAM to store all data; however, this approach negatively impacts the TCO of a warehouse.

One way to lower memory pressure is to impose a price penalty on using DRAM and induce the use of cheaper low-tier storage devices. According to the pricing policy of different Amazon EC2 instances as of June 2021 [4], DRAM storage is 45 times more expensive than solid state drive (SSD) storage at the

same capacity. A better approach than offloading the burden of using less DRAM to the customer is to provide the required performance through an optimized storage hierarchy that can offer the same service (performance) at a lower price. Common techniques involve demand paging to local storage [14] and memory disaggregation. Based on the principle of locality [2], both techniques keep frequently accessed pages (also called *hot pages*) in the fast and expensive DRAM and relegate infrequently accessed parts (the *cold pages*) to slower and cheaper storage tiers. Both techniques suffer from a significant performance slowdown the more memory is paged out. This is caused by (1) the large access latency of far storage tiers and (2) inflexible and slow system software that fails to exploit new and fast storage technologies such as phase change memory (PCM) [13].

In this paper, we present RapidSwap, a framework built for modern storage hierarchies to achieve a lower TCO at near-DRAM performance. RapidSwap classifies pages into different temperatures based on their access history. Hot pages are kept in DRAM and gradually downgraded to slower devices as they cool down. RapidSwap's awareness of the storage hierarchy and its optimized software stack minimize the page reclaim overhead and achieve a significantly lower TCO and cost effectiveness than existing solutions.

2 Background and Motivation

2.1 Tiered Storage and Novel Storage Devices

Tiered storage, also known as hierarchical storage, is a widely adopted technique in computing devices [7]. Faster and more expensive devices are placed at the upper side of the storage hierarchy, while slower and cheaper media are located below. Placing the data of all workloads in high-performance devices yields the best performance at the expense of larger operating costs. One possibility to decrease the cost while maintaining performance is to monitor and classify memory pages by their access frequency into different temperatures from *hot* to *cold*. The principle of locality dictates that, in general, the colder a page gets, the less likely it is to be accessed and can thus be migrated to slower storage devices without causing a large performance drop.

Recently, new storage technologies with dramatically improved performance characteristics have entered the market. Non-volatile memory (NVM) devices such as phase change memory (PCM) used in Intel's Optane product line have a read/write theoretical latency of 10 µs [10]; three orders of magnitude below that of conventional Hard Disk Drives (HDD). The NVDIMM interface allows direct load/store accesses by the CPU and is thus able to benefit from caches.

2.2 Techniques Proposed to Lower Memory Pressure

Transparent Memory Compression compresses cold pages *in memory* with a lightweight algorithm [3]. Support in the Linux kernel is provided by *zswap* [21]. Pages that do not benefit from compression are sent to local storage devices.

Table 1: Comparison of existing techniques and RapidSwap.

Type	Granularity	Failure Domain	Overhead
Software-defined Far Memory [15]	Page	Local	CPU
Hydra [16]	Page	Remote	CPU & Network
RackMem [13]	Page	Remote	Network
RapidSwap	Page	Local	minimal CPU
Optane PMEM Memory Mode [8]	Cache-line	Local	No SW Overhead

Zswap is expected to work well if the read latency from DRAM is significantly shorter than that of the backing store, however, its efficiency depends on the compressibility of the data in memory. A practical implementation of zswap is provided by Google's Far Memory [15]. Applied in their data centers, Google Far Memory classifies around 20% of all pages as cold, and among those, about 70% achieve 3x compression and are kept in DRAM. The remaining 30% are stored on traditional storage devices. Google reports a 4-5% reduction of their TCO.

Memory Disaggregation pools memory resources from different physical nodes over a low latency and high throughput network to overcome the limitations of the node-centric computation model. A significant disadvantage of memory disaggregation is the extension of the failure domain from a single local to multiple remote machines. Replication, erasure coding [16], or hybrid approaches [19] are used to achieve fault tolerance, however, these approaches requires additional storage or computational resources.

2.3 Tiered storage as a promising alternative

Current approaches from Linux demand paging, over transparent memory compression, to memory disaggregation all have shortcomings. On the other hand, RapidSwap eliminates the deficiencies of existing approaches and implements a high-performance demand paging system to a local storage hierarchy consisting of various types of devices with different characteristics. Data is stored in one of the local storage tiers according to RapidSwap's page classification. Pages are assigned a temperature ranging from hot to cold, representing how recently the page has been accessed. Pages are migrated between the different storage tiers depending on their temperature to store each page in the most beneficial device. Table 1 summarizes state-of-the-art techniques and RapidSwap.

3 RapidSwap

This section discusses the design and implementation of RapidSwap. RapidSwap is composed of three main components: an optimized *swap handler*, a *storage frontend*, and a *storage backend*. Figure 1 shows the overall architecture.

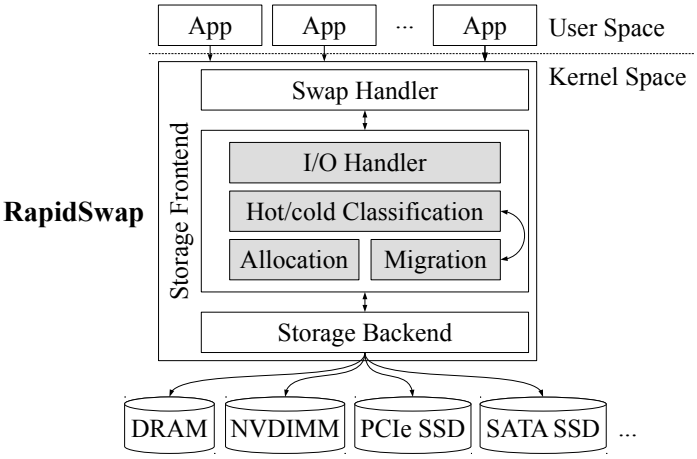


Fig. 1: Overall architecture of RapidSwap.

3.1 Swap Handler

Linux's virtual memory management is too slow for modern storage devices [12,13]. RapidSwap's optimized swap handler follows the design of RackMem [13] and manages pages with two quasi-ordered lists: the *active* and the *inactive* page list. To quickly react to page allocation requests, the inactive list is kept populated by pro-actively paging data out. If the *inactive* list becomes empty under high load, victim pages are taken from the head of the *active* page list.

3.2 Storage Frontend

RapidSwap's *storage frontend* manages the different tiered storage devices and exposes a uniform paging device to the *swap handler*. Consecutive pages are grouped into *slabs* to minimize metadata and I/O overhead. The storage frontend swaps in/out slabs from/to different storage devices and maintains a mapping of virtual pages to their locations in the storage hierarchy.

Slabs are classified into *hot*, *warm*, or *cold*. A newly allocated slab is considered hot, then transitions over warm to cold if it is not accessed for a certain period of time. Cold slabs are periodically migrated to the next lower level in the storage hierarchy. A page fault causes the associated slab to be immediately migrated up to the fastest storage below DRAM.

3.3 Storage Backend

The *storage backend* provides a uniform abstraction for physical storage devices. When the *storage backend* registers a device, RapidSwap's *storage frontend* gathers information about the storage device including its capacity and allocation, 4 KiB read/write, and deallocation latencies.

Table 2: Yahoo! Cloud Serving Benchmark (YCSB) workloads [1].

Workload Type	Distribution	Details
A: Update Heavy	Zipfian	50% Reads, 50% Writes
B: Read Mostly	Zipfian	95% Reads, 5% Writes
C: Read Only	Zipfian	100% Reads
D: Read Latest	Latest	Read from the fresh data
E: Short Ranges	Zipfian/Uniform	95% Scans, 5% Writes
F: Read-Modify-Write	Zipfian	50% Reads, 50% Read-Modify-Writes

Table 3: Maximum number of PMEM slabs allocated by local memory size.

Workload Type	Local: 50%	Local: 60%	Local: 70%	Local: 80%
A	2388	1403	330	222
B	2374	1438	402	215
C	2345	1433	273	225
D	1459	795	405	292
E	2549	2184	396	332
F	2340	1456	366	309

4 Results

4.1 Experimental Setup

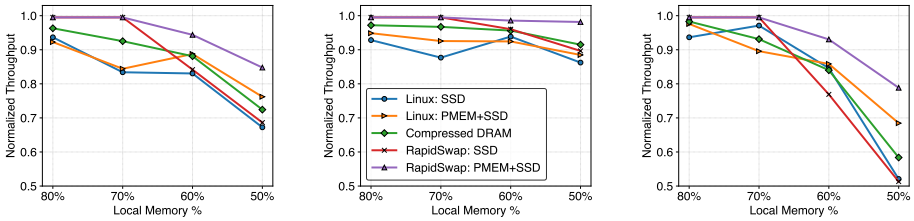
RapidSwap is evaluated on a data center server node with an Intel Xeon Silver 4215R processor and 64 GiB of DRAM. The node contains a two-tiered storage hierarchy consisting of a 960 GB Intel 905P Optane NVMe PCIe (SSD) and an 128 GB Intel Optane Persistent Memory 200 Series (PMEM). The base operating system is Ubuntu Server 20.04. The slab size is set to 1 MB (256 pages per slab). Slabs are demoted to the next colder level after a threshold of 5 seconds.

We use six different workloads from the Yahoo! Cloud Serving Benchmark (YCSB) suite [1] to measure the performance of RapidSwap. Benchmarks `A`, `B`, `C`, `D`, and `F` follow a *zipfian* access pattern where 80% of the total accesses go to 20% of the data. Workload D predominantly reads from just inserted data that is not physically contiguous. Workload E selects the key with a zipfian distribution, then scans a uniformly distributed number of records.

The performance of RapidSwap is evaluated by measuring the query response latency as reported by YCSB. Memory scarcity is simulated by artificially limiting the available DRAM to a certain percentage of the benchmark's overall maximum memory requirements (resident set size, RSS). RapidSwap is compared against a Linux baseline with PMEM and SSD paging devices where the former is prioritized, i.e., the SSD is only used when the PMEM paging device is completely full. The size of the available PMEM device is identical to that in RapidSwap. We also compare RapidSwap against compressed DRAM. Other than *zswap* we store all pages in DRAM whether compressible or not.

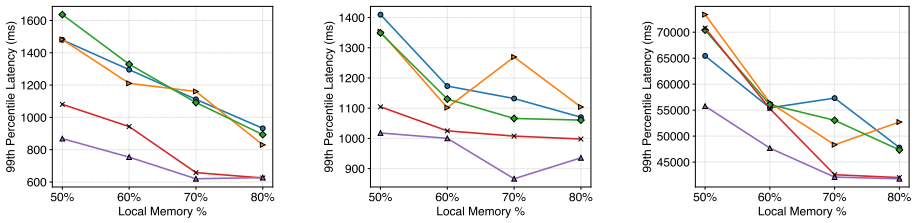
4.2 RapidSwap Performance

Degradation over DRAM Figure 2 plots the normalized throughput of the different implementations and with local memory limits set to 80, 70, 60, and 50 percent of the benchmark's RSS, which is around 3 GiB for all workloads. As



(a) YCSB C workload (b) YCSB D workload (c) YCSB E workload

Fig. 2: Normalized throughput over DRAM of RapidSwap and prior work.



(a) YCSB C workload (b) YCSB D workload (c) YCSB E workload

Fig. 3: 99th percentile latency of RapidSwap and prior work.

the amount of DRAM is reduced, all implementations experience performance degradations. Workloads exhibit three different patterns with minor (workloads `A`, `B`, `C`, and `F`; `C` is shown as a representative), average (workload `D`), and high sensitivity (workload `E`) to the available local memory. Table 3 shows the maximal number of allocated slab in PMEM for the different benchmarks. RapidSwap outperform the other approaches in all configurations.

CPU Overhead and I/O latency The CPU overhead is gathered by calculating the average system level utilization throughout the workload execution. CPU overhead caused by all three methods is less than 1% compared to the ideal case with 100% local memory. The average 99th percentile latency reported by RapidSwap is presented in Figure 3. Compared to the Linux baseline and compressed DRAM, RapidSwap exhibits significantly lower latencies thanks to its optimized page fault handler and pro-active page reclamation.

4.3 Cost of Storage Tier

To analyze the benefits of RapidSwap on the cost of the entire storage tier, we surveyed the current market prices of the different storage backends [18,9,11]. The cost is obtained by multiplying the peak utilization in all storage tiers by the cost of the respective device. The total cost is obtained by adding the

cost of the allocated DRAM. We consider only the fractional cost of a storage device (as opposed to the cost of the entire device) to reflect the pricing models of cloud data centers. RapidSwap achieves cost savings for all workloads and all configurations. At 70%, a 18-20% cost reduction in the storage hierarchy is achieved. As the amount of local memory is reduced, more data gets paged out to secondary storage which, in turn, leads to a higher cost in the storage hierarchy.

4.4 Cost Effectiveness

The total cost of the storage tier does not consider the cost incurred by performance degradation. A more sensible metric is the *cost effectiveness*, i.e., performance per cost. We compared the cost effectiveness of compressed DRAM and RapidSwap relative to a DRAM-only solution. Our first observation is that RapidSwap achieves a significantly better cost effectiveness than other solutions for all workloads and all configurations. Compared to DRAM, RapidSwap achieves an up to 40% higher cost effectiveness with 70% of the data kept in DRAM and 30% paged out. As the amount of DRAM is reduced, workloads experience a higher performance degradation and require larger amounts of storage.

5 Conclusion and Future Work

Motivated by the broad availability of novel storage technologies and the shortcomings of existing approaches to resource overcommitment, we have presented RapidSwap, a hierarchical far memory implementation that is built for diverse storage tiers composed of faster and slower devices. Paging only to local devices, RapidSwap does not extend the failure domain, and its awareness of the storage hierarchy allows it to significantly outperform other techniques that swap out data locally. The results demonstrate that proper management of new memory technologies can yield significant cost savings in data centers.

One direction of future work is application-specific resource management. As shown in previous work [15], RapidSwap can also benefit from a machine learning based approach to adjust the amount of local memory and the policies to degrade slabs to colder storage. Also, we did not yet compare RapidSwap against the memory mode configuration of PMEM, which offers similar functionality as RapidSwap, implemented in hardware at cache-line granularity.

Acknowledgments

We thank our shepherd Carl Waldspurger for his guidance and helpful feedback. This work was supported by the Korean government (MSIT) through the National Research Foundation by grants 0536-20210093 and 21A20151113068 (BK21 Plus for Pioneers in Innovative Computing - Dept. of Computer Science and Engineering, SNU). ICT at Seoul National University provided research facilities for this study.

References

1. Cooper, B.F., Silberstein, A., Tam, E., Ramakrishnan, R., Sears, R.: Benchmarking cloud serving systems with YCSB. In: Proceedings of the 1st ACM Symposium on Cloud Computing. pp. 143–154. SoCC '10, Association for Computing Machinery, New York, NY, USA (2010). https://doi.org/10.1145/1807128.1807152
2. Denning, P.J.: Thrashing: Its causes and prevention. In: Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I. pp. 915–922. AFIPS '68 (Fall, part I), Association for Computing Machinery, New York, NY, USA (1968). https://doi.org/10.1145/1476589.1476705
3. Douglis, F.: The compression cache: Using on-line compression to extend physical memory. In: USENIX Winter 1993 Conference (USENIX Winter 1993 Conference). USENIX Association, San Diego, CA (Jan 1993), https://www.usenix.org/conference/usenix-winter-1993-conference/compression-cache-using-line-compression-extend-physical
4. EC2 On-Demand Instance Pricing - Amazon Web Services. https://aws.amazon.com/ko/ec2/pricing/on-demand/
5. Amazon EC2 high memory instance types. https://aws.amazon.com/ec2/instance-types/
6. SAP HANA planning guide | Google Cloud. https://cloud.google.com/solutions/sap/docs/sap-hana-planning-guide
7. Herodotou, H., Kakoulli, E.: Automating distributed tiered storage management in cluster computing. Proc. VLDB Endow. **13**(1), 43–56 (Sep 2019). https://doi.org/10.14778/3357377.3357381
8. Why is the Intel Optane persistent memory in memory mode. https://www.intel.com/content/www/us/en/support/articles/000055895/memory-and-storage/intel-optane-persistent-memory.html, (Accessed on 08/11/2021)
9. Intel NMB1XXD128GPSU4 Intel Optane 200 128GB DDR-T Persistent Memory Module. https://www.itosolutions.net/Intel-Optane-200-128GB-DDR-T-Persistent-Memory-p/nmb1xxd128gpsu4.htm
10. Intel Optane SSD 905P Series. https://www.intel.com/content/www/us/en/products/memory-storage/solid-state-drives/consumer-ssds/optane-ssd-9-series/optane-ssd-905p-series.html
11. Intel Optane 905P 1.50 Tb Solid State Drive. https://www.newegg.com/intel-optane-ssd-905p-series-1-5tb/p/0D9-002V-003X1
12. Jo, C., Kim, H., Egger, B.: Instant virtual machine live migration. In: Economics of Grids, Clouds, Systems, and Services. GECON '20, Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-63058-4_14
13. Jo, C., Kim, H., Geng, H., Egger, B.: RackMem: A tailored caching layer for rack scale computing. In: Proceedings of the ACM International Conference on Parallel Architectures and Compilation Techniques. pp. 467–480. PACT '20, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3410463.3414643
14. Kilburn, T., Edwards, D.B.G., Lanigan, M.J., Sumner, F.H.: One-level storage system. IRE Transactions on Electronic Computers **EC-11**(2), 223–235 (1962). https://doi.org/10.1109/TEC.1962.5219356
15. Lagar-Cavilla, A., Ahn, J., Souhlal, S., Agarwal, N., Burny, R., Butt, S., Chang, J., Chaugule, A., Deng, N., Shahid, J., Thelen, G., Yurtsever, K.A., Zhao, Y., Ranganathan, P.: Software-defined far memory in warehouse-scale computers. In: Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems. pp. 317–330. ASPLOS '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3297858.3304053
16. Lee, Y., Maruf, H.A., Chowdhury, M., Shin, K.G.: Mitigating the performance-efficiency tradeoff in resilient memory disaggregation. CoRR **abs/1910.09727** (2019), http://arxiv.org/abs/1910.09727
17. Linux Virtual Machines | Microsoft Azure. https://azure.microsoft.com/en-us/pricing/details/virtual-machines/linux/
18. Samsung M386A8K40BM1-CPB 64GB DDR4-2133 4Rx4 LP ECC LRDIMM Server Memory. https://www.amazon.com/Samsung-M386A8K40BM1-CPB-DDR4-2133-LRDIMM-Server/dp/B017A8FJEG
19. Wang, Z., Li, T., Wang, H., Shao, A., Bai, Y., Cai, S., Xu, Z., Wang, D.: Craft: An erasure-coding-supported version of raft for reducing storage cost and network cost. In: 18th USENIX Conference on File and Storage Technologies (FAST 20). pp. 297–308. USENIX Association, Santa Clara, CA (Feb 2020), https://www.usenix.org/conference/fast20/presentation/wang-zizhong
20. Zaharia, M., Xin, R.S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M.J., Ghodsi, A., Gonzalez, J., Shenker, S., Stoica, I.: Apache spark: A unified engine for big data processing. Commun. ACM **59**(11), 56–65 (Oct 2016). https://doi.org/10.1145/2934664
21. zswap - The Linux Kernel documentation. https://www.kernel.org/doc/html/latest/vm/zswap.html