MORE EXPLORATION TO COMPOSABLE INFRASTRUCTURE THE APPLICATION AND ANALYSIS OF COMPOSABLE MEMORY

Wo-Hao Ruan
Cheng-Yueh Liu
Kong-Yu Shiu
Shih-Hao Hung

Department of Computer Science and Information Engineering
National Taiwan University
No.1, Sec. 4, Roosevelt Road
Taipei, Taiwan
{d06922037,d03922010}@ntu.edu.tw
{r04944031,hungsh}@csie.ntu.edu.tw

Matt Hsiao
Andy Liang
Keng-Yu Lin

Hewlett Packard Enterprise (Taiwan)
No. 66 Jing Mao 2 Road
Taipei, Taiwan
{matt.hsiao,andy.liang,keng-yu.lin}@hpe.com

ABSTRACT

Memory-Driven Computing facilitates real-world applications like in-memory database to benefit from substantial memory resources; however, these workloads perform dramatically downgraded if the memory capacity cannot reach to a threshold. In this paper, we present a novel remote memory system, which can handle memory-intensive workloads up to the scale of 1 Terabyte, by leveraging sophisticated tmpfs, virtual memory system, and Remote Direct Memory Access (RDMA) over Ethernet. To the best of our knowledge, this is the first design that provides (1) the most significant memory extension without any software modification and (2) the large-scale performance evaluations comparing to other state-of-the-art works about network-based remote memory. Based on the evaluation results, we believe the proposed composable memory is valuable for readers to re-visit wildly developed computing models (e.g., Apache Spark) and facilitate research work which can potentially reshape the data center architecture/deployment further.

Keywords: composable infrastructure, virtual memory, memory-intensive, resource utilization, big data

1 INTRODUCTION

The large-scale and data-centric analytic workloads for scientific and enterprise computing, which involve Big Data and Machine Learning are becoming the fastest growing demands to public/private cloud environments (Li et al. 2017, Lim et al. 2012); meanwhile, the virtualized deployment models like IaaS (Infrastructures as a Service) are widely adopted. Both of them point to the same demands and challenges: (1) greater memory capacity in volume server systems, (2) fast-changing system configuration requirements due to highly dynamic workload constraints, (3) varying innovation cycles of system components, and (4) the need for maximal sharing of system resources (Lim et al. 2012, Reiss et al. 2012, Samih et al. 2011, Zhang et al. 2011). Therefore, the concept of disaggregating traditional server resources and then placing in one or multiple shared pools was proposed (Lim et al. 2009, Lim et al. 2012). All of these drive the IT environments to evolve from *Traditional IT* to the emerging *Composable Infrastructure* during the past decade, with the purpose to fit the goals of maximizing resource utilization and minimize configuration complexity.

Today, no matter operating on public clouds (e.g., AWS, Azure, or GCP) or private clouds (e.g., On-Promise Infrastructure), the resource utilization under constraints mixed by performance, flexibility, and the cost is always the top concern to run a large-scale data center. Considering data gathered from related researches show the memory utilization of typical data centers is as low as 50% (Meng et al. 2010, Reiss et al. 2012, Samih et al. 2011, Zhang et al. 2011), it's convincing that the disaggregating DRAM pool or "Composable Memory Subsystem" will improve the utilization, alleviate related concerns and benefit to contemporary data centers (Abali et al. 2015).

Meanwhile, memory-intensive workloads like Big Data analytics and In-Memory database (Graefe et al. 2014) intend to occupying a massive trunk of memory at once, and avoiding performance penalties introduced by moving data among the memory hierarchies (Gu et al. 2017). That kind of tendency will further restrict these applications to run on the machines with a huge direct attached memory; where the traditional virtual memory cannot help since the costs of swapping memory pages to/from external spin disks are simply unacceptable (Mel Gorman 2010). However, in general cases of data center deployment, only limited nodes can support large-scale physical memory; which brings severe limitations: (1) only a fraction of "big nodes" can support memory-intensive workloads, even though other nodes can offer enough memory resources in a summation viewpoint, and (2) these "big nodes" will intend to be reserved for memory-intensive workloads only and avoid the resource provisioning and deployment latency; which leads to the further resource under-utilization.

Unlike the cases of storage and network fabric, memory component (e.g., CPU DIMM slots) is still a "physically limited" resource for contemporary infrastructure; even the most popular public cloud provider AWS does not provide the pricing for on-demand memory capacity. The main reasons are: (1) the physical constraints of CPU memory channel and pin count limit the maximum number of DIMM slots (Lim et al. 2009), (2) the latency and signal integrity requirements for DDR/DIMM are more restricted compared with other components (Abali et al. 2015, Li et al. 2017), and (3) CPU and DIMMs are tightly coupled; the interconnecting technology which can meet the cache coherence requirements over rack-level or even PoD scale is prohibitively expensive and proprietary (Friedrich et al. 2014).

The mature network technology - RDMA over Converged Ethernet (RoCE) and related studies (Guo et al. 2016, Aguilera et al. 2017) make us believe they have tremendous potential to become an alternative solution for maximizing the memory sharing and utilization within the rack-level (within 3-meters distance) or even the PoD (within 50-meters distance) scale. Also, recent work continually exploit the possibilities to use local memory as a cache of remote memory by utilizing the virtual memory management subsystem (Han et al. 2013, Rao and Porter 2016). Therefore, we propose a method of leveraging RAM-based file system (tmpfs), SCSI/iSER storage protocol, RoCE link protocol, and OS VMM to enable Composable Memory.

The rest of this paper is organized as follows. Section 2 introduces the evolution of IT infrastructures and related work. Section 3 mentions related work about Composable Memory solutions. Then, the details of the proposed framework will be illustrated in section 4. Section 5 will depict the experimental results and the efficiency of the proposed framework. Finally, conclusion will be made in section 6.

2 BACKGROUND

In the era of big data, to efficiently store and process fast-growing data, software stacks have been reshaped accordingly (e.g., in-memory database or distributed computing framework); the IT infrastructure of the data center also has been evolving to meet the service-level agreement.

Traditional IT relies on manual setups to assembly every subsystem like compute, storage and networking separately; all configurations work under a static/fixed mode for all kinds of workload. **Converged System** moves some of the setup complexities to software and expose them in a predefined and unified way. For example, SAN (Storage Attached Network) can support the storage deployment atop of a high-speed network like FoE to extend application scenarios (storage and network are "converged"). However, the knowledge about network setup is still necessary to configure well two separated subsystems when the use cases/requirements change. **Hyper-Converged System** (Koziris 2015) takes one more step by providing a software-defined layer atop H/W subsystem. For example, applications can ask for a 1TB storage space with certain QoS constraint; system can automatically select HDD/SSD from different nodes and virtually aggregate them through software layers with proper configurations which removes the complexities encountered by Converged System. However, having virtualized workloads is the core prerequisite and bare-metal mode is not supported. **Composable Infrastructure** (Li et al. 2017) takes a more aggressive move about how to utilize resources through more interconnecting innovations like SAS (Serial Attached SCSI), PCIe Switch, image streamer and their software composition layers. All subsystems (compute, storage, and network fabric) are treated as a component within a common "fluid resource pool" and "(un)wired physically" when necessary. The most significant difference is that Composable Infrastructure can provide a more fine-grained resource partition with a lower system overhead: all resources (any combination of single compute unit, storage device, and network fabric) operate *through real wires* just like the Traditional IT instead of software-defined layers, and fully support bare-metal workloads.

3 RELATED WORK

Fiber Switch/Interconnect - Since the optical-based interconnecting technology and high-speed/low-latency network technologies are getting mature, they can facilitate CPU LOAD/STORE memory access from cross-CPU to cross-nodes, and maximize memory utilization (Kachris et al. 2013, Zervas et al. 2018, Kachris and Tomkos 2012). However, when it comes to the rack-level (within 3-meters distance) or even the PoD scale (within 50-meters distance), the potential NUMA problems will rise again, and the complexity will exceed the state-of-the-art implementation of NUMA balancing (Lepers 2014, Chiang et al. 2018) because of the scale and the latency requirement for DIMMs is more restricted than other components. Even though the optical-based technology can be commercialized, its latency still has a huge impact (Calient , Li et al. 2017): (1) Speed of the light is close to 300K km/s or 300M m/s; travels 3m distance (within a rack) will take around 1/100 ms or 10 ns. (2) With around 30 ns latency of state-of-the-art optical circuit switch, the total sum (10 + 10 + 30 = 50ns) can reach double or more compared to the latency CPU load/store from/to DDR4 memory. (3) Additional latency through the copper to optical transceivers adds hundreds of nanoseconds.

Software-Based VMM Extension:InfiniSwap - Since a block device and system daemon are present in almost every machine and can work together without any central coordination. InfinitSwap leverages both of them and brings a novel, software-based solution to extend VMM and and provide an efficient paging system atop RDMA network. However, the new-introduced "InfiniSwap block device" requires plenty of patches binding to a specific Linux kernel version (Gu et al. 2017), and, inevitably, tightly couples with specific version of device drivers and associated NICs. Furthermore, it will take a long time for every system software to get mature, validated and then wildly adopted for deployment in fields.

In brief, memory-intensive applications (e.g. Memcached, Redis, TPC-C) are widely adopted today for low-latency services and data intensive analytics. However, when their working sets do not fully fit in memory, these applications will experience rapid performance deterioration (Gu et al. 2017) or severe memory thrashing; where system spends more time in retrieving and storing swapped pages than performing actual computation (Goichon et al. 2013, Moltó et al. 2013). Therefore, we need an affordable infrastructure design, which can rely on COTS components without expensive proprietary hardware, to build up efficient platforms with scale-up memory capacity to support these widely adopted workloads.

4 METHODOLOGY

We propose a method of leveraging RAM-based file system (tmpfs), SCSI/iSER storage protocol, RoCE link protocol, and OS VMM to enable Composable Memory; we want to maximize the utilization of memory subsystem at rack-level or PoD scale without problems mentioned: (1) The physical pin count issue of memory subsystem. (Lim et al. 2009) (2) The latency and signal integrity requirements of direct/fabric attached DIMMs. (Abali et al. 2015, Li et al. 2017) (3) The impacts of interconnect latency to CPU's LOAD/STORE instructions; VMM operates under a page granularity (e.g. 4KB) instead of a cache line (e.g. 64Bytes) with coherence concern. (Abali et al. 2015, Han et al. 2013)

4.1 RoCE/tmpfs-based Composable Memory

To build up a Composable Memory subsystem, there are two key components; one is for the physical interconnection, and another is for enabling memory access across nodes. The combinations of technologies will bring a specific cost and performance matrix, and benefit workloads in different ways. In this paper, we use two techniques to construct these two key components: (1) RoCE enables the interconnection among clients and the remote memory pools; clients rely on SCSI/iSER over RDMA NIC to mount one or more remote block devices as their VMM swapping partitions. (2) RAM-based tmpfs reserves a dedicated memory region on a remote node, which is exported as a block device and acts as the memory pool.

Therefore, we integrate all related components in Figure 1 to construct the RoCE/tmpfs-based Composable Memory and build up below three test benches atop that subsystem.

4.2 Latency Measurement: LMbench

LMbench (Staelin 2005) provides a built-in utility to simulate the cost of page swapping:lat_pagefault(). It leverages several POSIX calls to trigger continual 4K page swapping, measures their durations, and simulates the cost of OS VMM. With its capacity, we can separately put three identical files into local HDD-based block device, local RAMDisk, and RoCE/tmpfs-based block device created as the Figure 1; then use their results to simulate the VMM page fault cost atop these swapping devices. However, we have to recognize that the implementation of MS_INVALIDATE flag is not well defined in POSIX standard, and Linux will not flush out memory through the msync() system call; which may bring cache hits and under-report the latency. So using patches of force kernel to flush swap cache, excluding the overhead of drop_caches, and accessing pages in serialized sequences are necessary steps to get the accurate results and evaluate essential characteristics on Linux: (1) **Average page fault latency**: the basic block of performance measurement; (2) **Overhead of CPU**: monitor the CPU usages of remote (acts as the memory pool) and client nodes

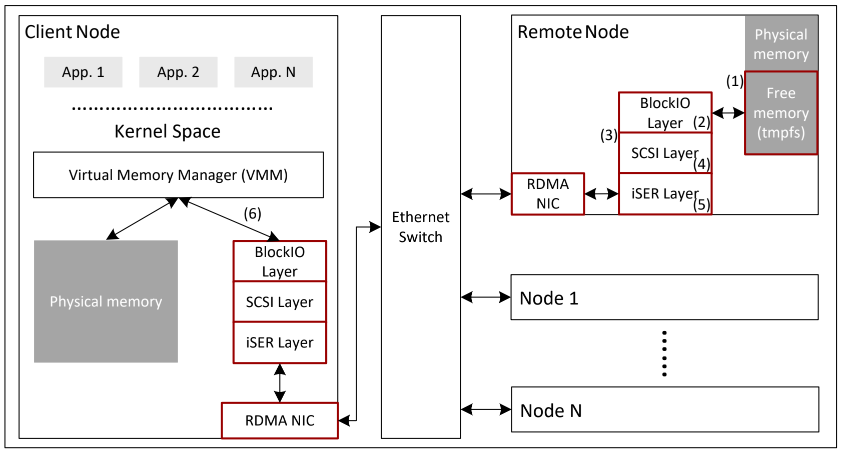

Figure 1: The architecture of RoCE/tmpfs-based composable memory.

during the execution of lat_pagefault() benchmark; (3) **Impact of parallelism**: understand the impact of parallelism on throughput, average page fault latency and CPU usage.

4.3 Micro-benchmark: Intensive-write

To further evaluate the potential gain, we implement a simple micro-benchmark 'paging_w' which will allocate memory and then do a continual WRITE (8 bytes) till total allocated memory is touched. There are three system configurations designed for evaluating this micro-benchmark: full memory without swap device (i.e. the baseline), limited memory with RoCE/tmpfs-based swap device, and limited memory with HDD-based local swap device. We implement a testbed of this micro-benchmark atop these three system configurations and evaluate two key characteristics: **(1) Duration of using different swap devices**: run "paging_w" on client node with 307GB, 358GB, 512GB, 768GB and 1TB data size to monitor their total execution time. **(2) Limitation of scale**: evaluate whether the ratio (duration difference) of full memory and swap device will converge to a certain level when the over-commit ratio is increasing.

Furthermore, considering the scale limitation, we propose a generalize model to predict the degradation rate by the following Equation (1):

$$\begin{array}{rcc}
RoCE/tmpfs & v.s. & FullMemory \\
1+C_1 & v.s. & 1 \\
D_{paging}+C_2 & v.s. & 1 \\
 & \cdots & \\
+)\quad D_{paging}+C_n & v.s. & 1 \\
\hline
1+(n-1)\times(D_{paging})+\sum_{k=1}^{n} C_k & v.s. & n
\end{array} \Bigg\downarrow n \tag{1}$$

Each row represents the duration of running a fix mount of workload size; which is normalized to 1 where the workload can be resident in physical memory. D_{paging} represents the multiple of paging latency between

Table 1: System configuration of VoltDB/TPC-C.

Master node	HPE ProLiant DL360 Gen10 with 24 * DDR4 32GB DIMMS
Client node	HPE ProLiant DL360 Gen10 with 6 * DDR4 16GB DIMMS
Standalone node	HPE ProLiant DL360 Gen10 with 12 * DDR4 64GB DIMMS
NIC Adaptor	HPE Bradford2 100G(Server) HPE 640FLR SFP28 25G (Client)
Operating System	Red Hat Enterprise Linux 7.4
HDD Drive	HPE 1TB 6G SATA 7.2K rpm
RDMA Drive	Mellonex OFED Driver 4.2-1.2.0.0 for RHEL 7.4
Script on Master	Create ramdisk as a block device shared through iSER protocol
Script on Client	Mount a remote block device through iSER protocol
VoltDB	Version 4.9.10
JDK	OpenJDK 1.7.0

Table 2: Simulated page fault latency.

File size	**RDMA**	**HDD**	**RAM**
128MB	4us	20.2us	0.95us
2GB	3.8us	20us	0.94us
20GB	3.8us	20us	0.94us

RoCE/tmpfs and LocalRAMDisk:

$$\frac{Latency_{RoCE/tmpfs}}{Latency_{LocalRAMDisk}}$$

We use this value to model the difference of execution time between using RoCE/tmpfs-based swap device and full memory configurations. C_k represents the cost of OS VMM itself, which is assumed as a relatively small number and can be ignored before reaching the memory thrashing state (Denning 1968).

Considering the "RoCE/tmpfs" configuration with 64GB physical memory: client node runs a 256GB workload (over-commit multiple = 4), and only the first 64GB can be resident in physical memory with normalized duration = 1; other three 64GB sections will be atop RoCE/tmpfs-based swap device with duration = (S1+S2). In sum, "RoCE/tmpfs-based" needs total execution time = 1 + 3(S1+S2), compared to 4 of "Full Memory"; which represents the peformance degradation rate where n = 4:

$$\frac{n}{1+(n-1)D_{paging}+\sum_{k=1}^{n} C_k} \rightarrow \frac{4}{1+3(D_{paging})+\sum_{k=1}^{4} C_k}$$

Furthermore, $\sum_{k=1}^{n} C_k$ can be ignored and the degradation rate will approximate to $\frac{1}{D_{paging}}$ before reaching the memory thrashing state where OS VMM overhead becomes significant (Denning 1968).

4.4 Industrial Benchmark: VoltDB/TPC-C

Beyond the latency analysis and micro-benchmark, we evaluate a commercial-grade, open-source In-Memory Database - VoltDB through the "transactions per sec" of TPC-C benchmark.

For full memory configuration, it is deployed on Standalone node with 12 * 64GB = 768GB physical memory; which can support "In-Memory Processing" for complete TPC-C benchmark data (around 400GB in our testbed). For two swap device configurations, it is deployed on Client Server with 6 * 16GB = 96GB physical memory and leverages RoCE/tmpfs-based or HDD-based swap device to extend limited physical memory and support "In-Memory Processing" for whole TPC-C benchmark data (around 400GB in our testbed). Also, run TPC-C benchmark in every 30 mins incremental period, and log its total database size and average throughput (transaction/sec) information. Since TPC-C can only have one set of output (database size and throughput) for every single run with 30 mins granularity, we use logarithmic approach to calculate the throughput number (**BOLD** in Table 4) under specific database size which sits between two closest sets of valid output.

5 EVALUATION

5.1 Latency Measurement: LMbench

Average page fault latency: The result in Table 2 shows the relationship between the mapped file size and average page fault latency of local RAMDisk and RoCE/tmpfs-based block device (RDMA). On local RAMDisk, the latency keeps the same as around 0.94 microseconds regardless of its file size; on the RoCE/tmpfs-based block device (RDMA), a larger file size makes the latency converge to a stable 3.83 microseconds; therefore we choose 2GB as target file size for following experiments since there's no significant difference after that. Comparing to the 0.94us of full memory configuration (RAM), the simulated page fault latency of RoCE/tmpfs-based block device (RDMA) is about 3.8us, and lead to the multiple of page fault latency to 4 as (equation (1)) stated.

Overhead of CPU: The main reason of choosing RoCE as the interconnection is that RDMA can offload the network protocol processing from CPU to dedicated blocks within NICs, achieve the low latency requirement, and facilitate our Composable Memory design. In Figure 2, client node (blue) shows approximately linear increment of CPU usage along with the additional processes of running lat_pagefault(). During that time remote node (red; the memory pool) keeps consuming around 3% of CPU cycles when the total throughput of 4KB page swapping (supported by the tmpfs on remote node) grows to 8 times; which demonstrates the scalability of RoCE/tmpfs-based Composable Memory.

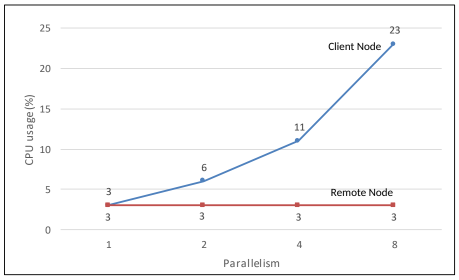

Figure 2: Overhead of CPU: LMbench with parallelism.

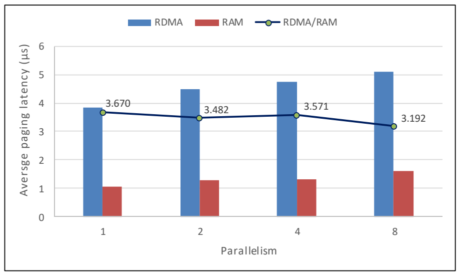

Figure 3: Average paging latency: LMbench with parallelism.

Impact of parallelism: Figure 3 shows that when the number of parallel processes increased from 1 to 2, 4 and 8, the average page fault latency increased 16.6%, 23.5%, 32.9% for RoCE/tmpfs-based block device (RDMA) and 22.9%, 26.9%, 52.8% for local RAMDisk (RAM). We can find the latency ratio of RDMA over RAM decreases from 4 to 3 times; which demonstrates an important finding: before reaching limitations of I/O bound, we can leverage parallelism over RoCE/tmpfs-based Composable Memory to deliver a higher throughput if its latency can meet application's requirement. In this case, we can increase the data swapped to 800% with 32.9% latency degradation and reach 537% gain in total throughput.

5.2 Micro-benchmark: Intensive-Write

Duration of using different swap devices: Figure 4 shows the measurement results of Intensive-Write on three configurations and using full memory as the base line (100%). The vertical axis represents the related performance compared to full memory:

$$\frac{Duration_{FullMemory}}{Duration_{RoCE/tmpfs}}$$

. 50% means it takes double duration time, compared to full memory, to complete the task. The horizontal axis represents the five over-commit ratios (data size/physical RAM size). At 120% over-commit ratio, RoCE/tmpfs-based swap device drops to 72% and dominates the HDD-based local swap device which drops to 14%. when we double the data size to over-commit ratio 200%), RoCE/tmpfs-based performance drops to around 50% and HDD-based drops to 10%. Furthermore, when the over-commit ratio reaches 400%, it will lead to a drop to 36% for RoCE/tmpfs-based and almost unpractical 6% for HDD-based. Comparing to HDD, using RoCE/tmpfs-based swap device shows more promising scalability to enable Composable Memory and support memory-intensive workloads.

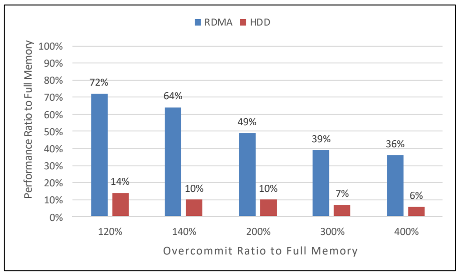

Figure 4: Performance ratio: compared to full memory.

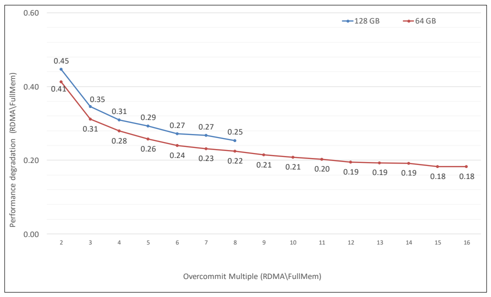

Figure 5: Limitation of Scale: Intensive-Write

Limitation of scale: Figure 5 shows the results of further scaling. For the 128GB configuration (blue), performance degradation are 45%, 35%, 31% with over-commit multiple 2, 3, 4; which perform similarly as over-commit ratio 200%, 300% and 400% in Figure 4. When we further extend the over-commit multiple from 5 to 8 (800%; or execute 1TB data atop 128GB physical memory), their performance drops from 29% to 25%. For 64GB configuration (red), its performance keeps decreasing with a more flatter rate from over-commit multiple 9 to 16, and drops to the minimal number we can measure: 18%. We can notice that there is a fix 3-4% difference between 64GB (red) and 128GB (blue); which is caused from OS which can occupy a fraction of memory (available physical memory is smaller) and lead the over-commit multiple of 64GB (red) is a little bit larger than 128GB (blue).

Based on Table 2, the multiple of page fault latency between RoCE/tmpfs-based block device and full memory is 4; which leads to the result of Equation (1) as:

$$\frac{n}{1+(n-1)(4)+\sum_{k=1}^{4} C_k}$$

If temporarily ignore the OS VMM overhead $\sum_{k=1}^{4} C_k$ and put n = 2, 4, 8, and 16, we can derive the theoretical degradations as 2/5 = 0.4, 4/13 = 0.31, 8/29 = 0.28 and 16/61 = 0.26; below Table 3 shows the comparison of theoretical and experimental results (for 64GB, there is a fixed addition to compensate the OS effects).

We can see, except the multiple 16 (difference is around 20%), all experimental values are very close to the theoretical results (difference is 0-10%) where OS VMM overhead is ignored. Atop the experiment results we can develop a hypothesis: *before the multiple reaches 8*, degradation ratio follows the prediction of Equation (1) where paging latency dominates, and OS VMM overhead $\sum_{k=1}^{n} C_k$ can be ignored; which means, beyond this multiple 8, OS VMM may need revisiting to further optimize RoCE/tmpfs-based Composable Memory.

Table 3: Comparison of degradations.

Multiple	**Theoretical**	**128GB**	**64GB (+3%)**
2	0.4	0.45	0.41 (0.44)
4	0.31	0.31	0.28 (0.31)
8	0.28	0.27	0.22 (0.25)
16	0.26	N/A	0.18 (0.21)

Table 4: VoltDB/TPC-C performance comparison.

Trend line	**Transactions per second**			**Performance diff**	
DB Size (GB)	**RDMA**	**HD**	**MEM**	**RDMA**	**HD**
90	15661	15579	**16508**	5.1%	5.6%
120	14232	6534	**15694**	9.3%	5.6%
140	**13529**	4805	**15067**	10.2%	58.4%
180	**12558**	N/A	**14247**	11.9%	N/A
270	11173	N/A	12653	11.7%	N/A
360	**9879**	N/A	**11985**	17.6%	N/A

5.3 Industrial benchmark: VoltDB/TPC-C

Figure 6 shows the relationship between TPC-C performance (transaction/sec) and VoltDB size of three configurations:

Orange represents full memory (RAM). Since the standalone node equips 768GB physical memory, it can support "In-Memory Processing" for complete working set with the best performance as the baseline. Performance indicator (transaction/sec) will decline along with the growth of database size because of the overhead from operating increasing tables and rows (e.g., select a tabular from 10K rows vs. 100K rows).

Blue represents RoCE/tmpfs-based swap device (RDMA). Since the client node equips only 96GB physical memory, it leverages its swap device to extend physical memory limitation and support "In-Memory Processing". Before the "breaking point" around 96GB, its performance is identical to Orange (baseline); after that, the system will start to consume memory supported by RoCE/tmpfs-based swap device, and its tendency to decline is more notable compared to Orange (full memory; the baseline).

Red represents HDD-based local swap device (HDD). Since the client node equips only 96GB physical memory, it leverages its swap device to extend physical memory limitation and support "In-Memory Processing". Before the "breaking point" around 96GB, its performance is identical to Orange (baseline); after that, the system will start to consume memory supported by local HDD-based swap device, and its tendency to decline is more significant and severe compared to Blue (RDMA) and Orange (RAM; the baseline); moreover, before its database size can grow to double, TCP-C benchmark execution will go into an extremely slow state or even crash, because of the memory thrashing of Linux VMM.

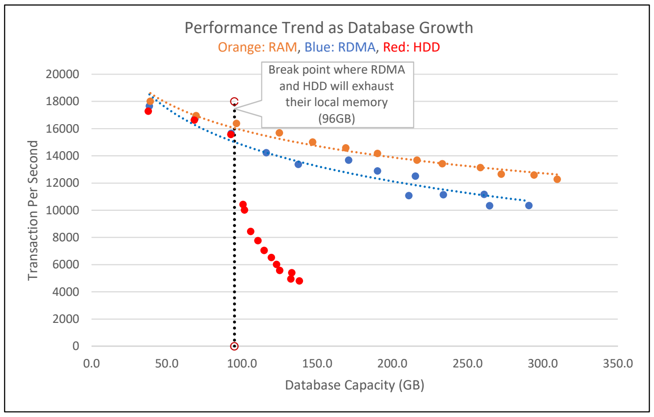

Figure 6: VoltDB/TPC-C Performance Trend.

The results in Table 4 shows that, atop RoCE/tmpfs-based Composible Memory, VoltDB/TPC-C can extend its capacity (database size) from 90GB to 360GB (over-commit ratio 400%) with only 17.6% throughput degradation; which brings a 330% synergy gain in total.

6 CONCLUSIONS AND FUTURE WORK

This paper rethinks the well-known remote memory paging problem in the context of leveraging COTS HW and SW components without any modification to applications, libraries, OSes, or hardware. With the performance/scalability models based on LMbench results and evaluations atop unmodified VoltDB/TPC-C, we understand that:

- For memory-intensive workloads, the proposed system is worth more comprehensive studies. Comparing to local HDD, its 4KB swapping latency is much lower (Table 2) and therefore it can deliver a significant enhancement both for micro-benchmark (Figure 4) and industrial system-benchmark (Figure 6) we have demonstrated.
- Through alleviating the memory thrashing problem stated in Sect. 4.3 and Figure 6, the proposed system provides a new direction to deploy memory-intensive workloads on nodes without sufficient memory; which is impractical and never considered as an option. E.g., the official documentation of VoltDB suggests to turn-off virtual memory for maximizing its performance.
- The performance and usability of proposed system can benefit new emerging researches like advanced memory management, application-centric memory over-commit, Memory-Driven Computing, and TensorFlow through alleviating the limitations of memory over-commit; the key is to figure out the degradation sensibility and cost-performance matrix of applying Composable Memory on specific workload.

RoCE/tmpfs-based Composable Memory can further extend the concept of Composable Infrastructure. We believe combining OS VMM and high speed/low latency network has great potential for related researches and we can develop in many directions. First, we can analyze the performance bottleneck inside Linux Kernel and figure out the optimization opportunities like the VMM overhead when reaching memory thrashing state. Second, we can develop the cost-performance matrix for widely adopted memory-intensive workloads, which can impact the contemporary data center design and public cloud pricing model (e.g., AWS).

REFERENCES

Abali, B., R. J. Eickemeyer, H. Franke, C.-S. Li, and M. A. Taubenblatt. 2015. “Disaggregated and optically interconnected memory: when will it be cost effective?”. *arXiv preprint arXiv:1503.01416*.

Aguilera, M. K., N. Amit, I. Calciu, X. Deguillard, J. Gandhi, P. Subrahmanyam, L. Suresh, K. Tati, R. Venkatasubramanian, and M. Wei. 2017. “Remote memory in the age of fast networks”. In *Proceedings of the 2017 Symposium on Cloud Computing*, pp. 121–127. ACM.

Calient. “Calient S320 Datasheet”. http://www.calient.net/members-area/?redirect-to=/download/s320-optical-circuit-switch-datasheet/. accessed on 23 September 2015.

Chiang, M.-L., S.-W. Tu, W.-L. Su, and C.-W. Lin. 2018. “Enhancing Inter-Node Process Migration for Load Balancing on Linux-Based NUMA Multicore Systems”. In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, pp. 394–399. IEEE.

Denning, P. J. 1968. “Thrashing: Its causes and prevention”. In *Proceedings of the December 9-11, 1968, fall joint computer conference, part I*, pp. 915–922. ACM.

Friedrich, J., H. Le, W. Starke, J. Stuechli, B. Sinharoy, E. J. Fluhr, D. Dreps, V. Zyuban, G. Still, C. Gonzalez et al. 2014. "The POWER8 tm processor: Designed for big data, analytics, and cloud environments". In *2014 IEEE International Conference on IC Design & Technology*.

Goichon, F., G. Salagnac, and S. Frénot. 2013. "Swap fairness for thrashing mitigation". In *European Conference on Software Architecture*, pp. 311–315. Springer.

Graefe, G., H. Volos, H. Kimura, H. Kuno, J. Tucek, M. Lillibridge, and A. Veitch. 2014. "In-memory performance for big data". *Proceedings of the VLDB Endowment* vol. 8 (1), pp. 37–48.

Gu, J., Y. Lee, Y. Zhang, M. Chowdhury, and K. G. Shin. 2017. "Efficient memory disaggregation with INFINISWAP". In *Proceedings of the 14th USENIX Conference on Networked Systems Design and Implementation*, pp. 649–667. USENIX Association.

Guo, C., H. Wu, Z. Deng, G. Soni, J. Ye, J. Padhye, and M. Lipshteyn. 2016. "RDMA over commodity ethernet at scale". In *Proceedings of the 2016 ACM SIGCOMM Conference*, pp. 202–215. ACM.

Han, S., N. Egi, A. Panda, S. Ratnasamy, G. Shi, and S. Shenker. 2013. "Network support for resource disaggregation in next-generation datacenters". In *Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks*, pp. 10. ACM.

Kachris, C., K. Kanonakis, and I. Tomkos. 2013. "Optical interconnection networks in data centers: recent trends and future challenges". *IEEE Communications Magazine* vol. 51 (9), pp. 39–45.

Kachris, C., and I. Tomkos. 2012. "A survey on optical interconnects for data centers". *IEEE Communications Surveys & Tutorials* vol. 14 (4), pp. 1021–1036.

Koziris, N. 2015. "Fifty years of evolution in virtualization technologies: from the first IBM machines to modern hyperconverged infrastructures". In *Proceedings of the 19th Panhellenic Conference on Informatics*, pp. 3–4. ACM.

Lepers, B. 2014. *Improving performance on NUMA systems*. Ph. D. thesis, Université de Grenoble.

Li, C.-S., H. Franke, C. Parris, B. Abali, M. Kesavan, and V. Chang. 2017. "Composable architecture for rack scale big data computing". *Future Generation Computer Systems* vol. 67, pp. 180–193.

Lim, K., J. Chang, T. Mudge, P. Ranganathan, S. K. Reinhardt, and T. F. Wenisch. 2009. "Disaggregated memory for expansion and sharing in blade servers". In *ACM SIGARCH Computer Architecture News*, Volume 37, pp. 267–278. ACM.

Lim, K., Y. Turner, J. R. Santos, A. AuYoung, J. Chang, P. Ranganathan, and T. F. Wenisch. 2012. "System-level implications of disaggregated memory". In *High Performance Computer Architecture (HPCA), 2012 IEEE 18th International Symposium on*, pp. 1–12. IEEE.

Mel Gorman 2010. "Huge Pages/libhugetlbfs". http://lwn.net/Articles/374424/.

Meng, X., C. Isci, J. Kephart, L. Zhang, E. Bouillet, and D. Pendarakis. 2010. "Efficient resource provisioning in compute clouds via vm multiplexing". In *Proceedings of the 7th international conference on Autonomic computing*, pp. 11–20. ACM.

Moltó, G., M. Caballer, E. Romero, and C. de Alfonso. 2013. "Elastic memory management of virtualized infrastructures for applications with dynamic memory requirements". *Procedia Computer Science* vol. 18, pp. 159–168.

Rao, P. S., and G. Porter. 2016. "Is memory disaggregation feasible?: A case study with spark sql". In *Proceedings of the 2016 Symposium on Architectures for Networking and Communications Systems*, pp. 75–80. ACM.

Reiss, C., A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch. 2012. "Heterogeneity and dynamicity of clouds at scale: Google trace analysis". In *Proceedings of the Third ACM Symposium on Cloud Computing*, pp. 7. ACM.

Samih, A., R. Wang, C. Maciocco, T.-Y. C. Tai, and Y. Solihin. 2011. "A collaborative memory system for high-performance and cost-effective clustered architectures". In *Proceedings of the 1st Workshop on Architectures and Systems for Big Data*, pp. 4–12. ACM.

Staelin, C. 2005. "lmbench: an extensible micro-benchmark suite". *Software: Practice and Experience* vol. 35 (11), pp. 1079–1105.

Zervas, G., H. Yuan, A. Saljoghei, Q. Chen, and V. Mishra. 2018. "Optically disaggregated data centers with minimal remote memory latency: technologies, architectures, and resource allocation". *Journal of Optical Communications and Networking* vol. 10 (2), pp. A270–A285.

Zhang, Q., J. L. Hellerstein, R. Boutaba et al. 2011. "Characterizing task usage shapes in Google's compute clusters". In *Proceedings of the 5th international workshop on large scale distributed systems and middleware*, pp. 1–6. sn.

AUTHOR BIOGRAPHIES

WO-HAO RUAN is an engineering manager in HPE HybridIT and enrolled as PhD student of NTU PASLab; He keeps driving the industrial-academic collaboration.

CHENG-YUEH LIU is currently a Ph.D. candidate of NTU PASLab under Prof. Shih-Hao Hung.

MATT HSIAO is a Sr. System Software Engineer, HPE HybridIT.

ANDY LIANG is a System Software Engineer, HPE HybridIT.

KENG-YU LIN is a Sr. System Software Engineer, HPE HybridIT.

KONG-YU SHIU is a graduate student of NTU PASLab under Shih-Hao Hung.

SHIH-HAO HUNG is currently a professor in Department of Computer Science and Information Engineering at National Taiwan University.