International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 10 Issue: 09 | Sep 2023 www.irjet.n p-ISSN: 2395-0072

Study on Composable Infrastructure – Breakdown of Composable Memory

Shubham Prashant Bhambar[1], Rohit Sanjay Pawar[2], Nupur Shashikant Patil[3], Prathamesh Kailas Bachhav[4]

[1]Master of Science in Industrial Engineering, University of Minnesota, USA
[2 and 3]Bachelors in Engineering in Artificial Intelligence & Data Science Engineering, SNJB'S College of Engineering
[4]Bachelors in Engineering in Computer Engineering, SNJB'S College of Engineering, Chandwad

---

Abstract - Memory-based computing allows real-world applications, such as in-memory databases, to easily take advantage of large memory resources; however, these workloads degrade significantly if memory capacity does not reach a threshold. In this paper, we present a new remote memory system, capable of handling memory-intensive workloads up to 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 project to deliver the major memory expansion without software modifications and full-scale performance testing against other state-of-the-art work on the remote memory-based network. Based on the evaluation results, we believe that the proposed composable memory is useful for readers to examine widely developed computational models (e.g. Apache Spark) and to facilitate research work that may further reshape data center architecture implementation.

Key Words: **Composable Infrastructure, Virtual Memory, Memory-Intensive, Resource Utilization, Big Data**

1. INTRODUCTION

Large-scale, data-centric analytics workloads for scientific and business computing, involving big data and machine learning, are becoming the fastest growing demands for private/public cloud environments (Li et al. 2017, Lim et al. 2012); meanwhile, virtualized deployment models such as IaaS (Infrastructure as a Service) are being widely adopted. Both point to the same needs and challenges: increased memory capacity in volume server systems, rapidly changing system configuration requirements due to highly dynamic workload constraints, variable innovation cycles of system components, and the need for maximum 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 putting them into one or more shared pools has been proposed (Lim et al. 2009, Lim et al. 2012). All of this is driving IT environments to evolve from traditional IT to the emerging composable infrastructure over the last decade, in order to achieve the goals of maximizing resource utilization and minimizing configuration complexity.

Today, whether you operate on public clouds (e.g. AWS, Azure or GCP) or private clouds (e.g. On-Promise Infrastructure), limited resource utilization combined with performance, flexibility and cost is always the primary concern in managing a large-scale data center. Whereas data collected from related research shows that typical data center memory utilization is as high as 50% (Meng et al. 2010, Reiss et al. 2012, Samih et al. 2011, Zhang et al. 2011), it is persuasive that DRAM pool unbundling or "Composable Memory Subsystem" will improve utilization, alleviate related concerns, and benefit contemporary data centers (Abali et al. 2015).

Meanwhile, memory-intensive workloads such as Big Data analytics and In-Memory Database (Graefe et al. 2014) aim to occupy a large amount of memory simultaneously and avoid the performance penalties introduced by moving data across memory hierarchies (Gu et al. 2014). others 2017). This kind of trend will further restrict these apps from running on machines with huge directly attached memory; where traditional virtual memory cannot help as the costs of swapping memory pages from/to external spinning disks are simply unacceptable (Mel Gorman 2010). However, in general data center deployment cases, only limited nodes can support physical memory at scale; which comes with severe limitations: only a fraction of "large nodes" can support memory-intensive workloads, although other nodes may offer sufficient memory resources in a summation point of view, and these "large nodes" will try to be reserved only for memory-intensive workloads and avoid resource provisioning and deployment latency; which leads to further underutilization of resources.

Unlike the cases of red and memory structure, the memory component (for example, CPU DIMM slots) follows being a "physically limited" recourse for contemporary infrastructure; including the most popular public cloud provider, AWS, does not provide the price of low memory capacity demanded. The main reasons are: the physical restrictions of the CPU

memory channel and the number of pines limit the maximum number of DIMM slots (Lim et al. 2009),the latency and signal integrity requirements for DDR/DIMM are more restricted in comparison with other components (Abali et al. 2015, Li et al. 2017), and (3) the CPU and the DIMMs are strictly approved; the interconnection technology that can meet the rack-level cache coherence requirements or even a prohibitively expensive and proprietary PoD escalator (Friedrich et al. 2014).

Maturation of network technology: RDMA over Converged Ethernet (RoCE) and related studies (Guo et al. 2016, Aguilera et al. 2017) lead us to believe that they have huge potential to become an alternative solution to maximize sharing and memory usage within the rack. -level (within a distance of 3 meters) or even the PoD scale (within a distance of 50 meters). Furthermore, recent works continuously exploit the possibilities of using local memory as a remote memory cache using the virtual memory management subsystem (Han et al. 2013, Rao and Porter 2016). Therefore, we propose a method to take advantage of a RAM-based file system (tmpfs), SCSI/iSER storage protocol, RoCE handshake, and OS VMM to enable composable memory.

2. BACKGROUND

In the age of big data, to efficiently store and process rapidly growing data, software stacks have been reshaped accordingly (for example, in-memory databases or distributed computing frameworks); the IT infrastructure of the data center has also evolved to meet the service level agreement.

Traditional IT relies on manual configurations to separately assemble each subsystem such as processing, storage, and networking; all configurations work in static/fixed mode for all workload types. Converged System transfers some of the configuration complexities to the software and exposes them in a predefined and unified way. For example, Storage Attached Network (SAN) can support the deployment of storage over a high-speed network such as FoE to expand application scenarios (storage and network are "converged"). However, knowledge of network configuration is still required to properly configure two separate subsystems as use cases/requirements change. The hyper-converged system (Koziris 2015) goes one step further by providing a software-defined layer on top of the H/W subsystem. For example, applications may require 1TB storage space with some QoS restrictions; The system can automatically select HDDs/SSDs from different nodes and virtually add them through software layers with appropriate configurations that eliminate the complexities encountered by the converged system. However, workload virtualization is the main prerequisite and full mode is not supported. Composable Infrastructure (Li et al. 2017) takes a more aggressive move on how to use resources through multiple to interconnect innovations like SAS (Serial Attached SCSI), PCIe switches, image streaming, and its software composition layers. All subsystems (computing, storage, and network infrastructure) are treated as components within a common "pool of fluid resources" and "physically (dis)connected" when necessary. The most significant difference is that composable infrastructure can provide more granular resource partitioning with less system overhead: all resources (any combination of a single compute unit, storage device, and network infrastructure) run on real wires, just like traditional IT. of software-defined tiers and are fully compatible with entire workloads.

3. RELATED WORK

Fiber Switch/Interconnect - As optical-based interconnect technology and high-speed/low-latency networking technologies are maturing, they can facilitate cross-CPU to cross-node CPU LOAD/STORE memory access and maximize memory utilization. memory (Kachris et al. 2013, Zervas et al. 2018, Kachris and Tomkos 2012). However, when it comes to rack level (within 3m distance) or even PoD scale (within 50m distance), the potential NUMA issues will re-occur and the complexity will surpass the state of the art. The NUMA rolling art implementation (Lepers 2014, Chiang et al. 2018) due to scale and latency requirements for DIMMs is more constrained than other components. While the optics-based technology may be commercialized, its latency is still a punch (Calient, Li et al. 2017): the speed of light is close to 300K km/s or 300M m/s; trips 3m apart (within a rack) will take about 1/100ms or 10ns. With a latency of around 30 ns from the latest generation optical switch, the sum total (10 + 10 + 30 = 50 ns) can reach double or more than CPU load/store latency to/from memory DDR4. The additional latency through copper to optical transceivers adds hundreds of nanoseconds.

Software-Based VMM Extension:InfiniSwap - Because a block device and a system daemon are present on almost every machine and can work together without any central coordination. InfinitSwap takes advantage of both and offers a new software-based solution to extend VMM and provide efficient paging system on the RDMA network. However, the new "InfiniSwap block device" requires a large number of patches tied to a specific version of the Linux kernel (Gu et al. 2017) and inevitably tightly coupled to a specific version of Linux device drivers and associated NICs. Furthermore, it will take a long time for any system software to mature, validate, and then be widely adopted for implementation in the fields.

In short, 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 degradation (Gu et al. 2017) or a severe memory beat; where the system takes longer to retrieve and store the swapped pages than to perform the actual computation (Goichon et al. 2013, Moltó et al. 2013).

Therefore, we need an affordable infrastructure design, which can be based on COTS components without expensive proprietary hardware, to build efficient platforms with scalable memory capacity to support these widely adopted workloads.

4. PROCEDURES

We propose a method to take advantage of RAM-based file system (tmpfs), SCSI/iSER storage protocol, RoCE handshake, and OS VMM to enable composable memory; We want to maximize the use of the memory subsystem at the rack level or PoD scale without the problems mentioned: The memory subsystem physical pin count issue. (Lim et al. 2009) The latency and signal integrity requirements of directly attached/fabric DIMMs. (Abali et al. 2015, Li et al. 2017) Pipeline latency affects CPU LOAD/STORE instructions; VMM works with page granularity (for example, 4 KB) instead of cache line (for example, 64 bytes) with a focus on consistency. (Abali et al. 2015, Han et al. 2013)

RoCE/tmpfs-based Composable Memory

To build a composable memory subsystem, there are two key components; one is for physical interconnection and the other is for enabling memory access between nodes. Technology combinations will drive a specific range of cost and performance and benefit workloads in different ways. In this paper, we use two techniques to create these two key components: RoCE enables interconnection between clients and remote memory pools; clients rely on SCSI/iSER on RDMA NICs to mount one or more remote block devices as VMM swap partitions. RAM-based tmpfs reserves a dedicated area of memory on a remote node, which is exported as a block device and acts as a memory pool.

Therefore, we integrated all the components listed in Figure 1 to build the RoCE/tmpfs-based composable memory and built below three testbeds on top of that subsystem.

Latency Measurement: LMbench

LMbench (Staelin 2005) provides a built-in utility to simulate the cost of page swapping: lat_pagefault(). Leverage multiple POSIX calls to enable seamless 4K page swapping, calculate its duration, and simulate the cost of OS VMM. With its capacity, we can separately insert three identical files into local HDD based block device, local RAMDisk and RoCE/tmpfs based block device created as Figure 1; then use the results to simulate the cost of VMM page fault on these swap devices. However, we must recognize that the implementation of the MS_INVALIDATE flag is not well defined in the POSIX standard and Linux will not flush memory via the msync(); which can lead to cached hits and underestimated latency. Therefore, using hard kernel patches to flush the swap cache, excluding the drop_caches overhead, and accessing pages in serialized streams are necessary steps to get accurate results and evaluate essential features on Linux: Average latency of page faults: the basic building block of performance measurement;

CPU Overload: Monitor CPU usage of remote nodes (acts as a memory pool) and client nodes when performing a benchmark pagefault(); Impact of parallelism: Understand the impact of parallelism on performance, average page fault latency, and CPU utilization.

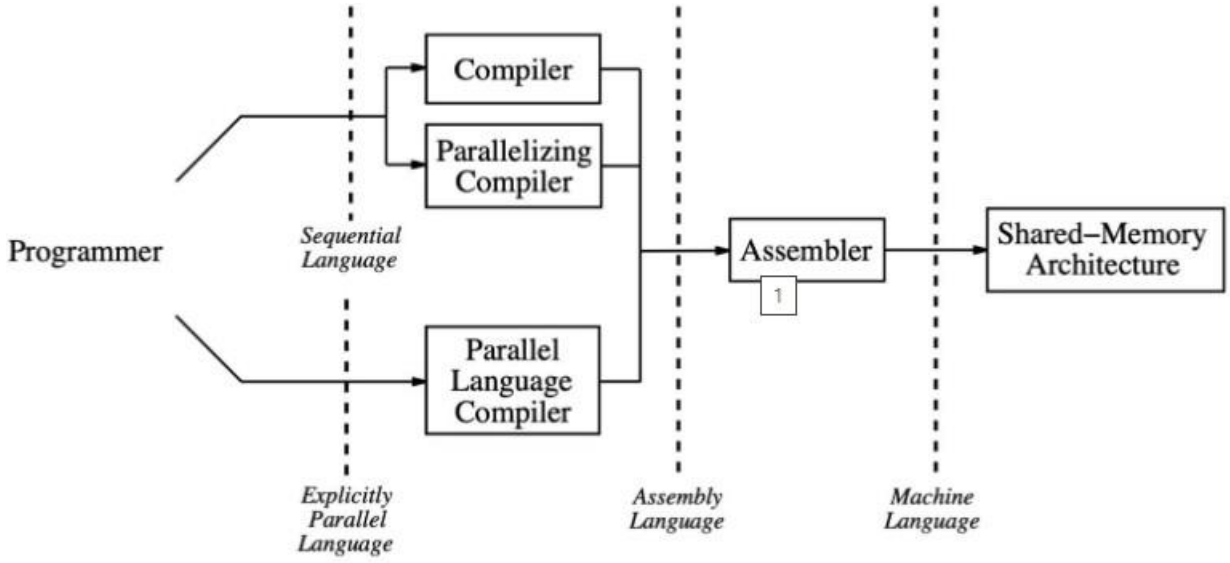

Fig -1: Various Interfaces between programmer and system

Micro-benchmark: Intensive-write

To further evaluate the potential gain, let's implement a simple 'paging_w' micro-benchmark that will allocate memory and then do a continuous (8 byte) WRITE until the total allocated memory is touched. There are three system configurations designed to evaluate this microbenchmark: full memory with no swap device (that is, the baseline), limited memory with RoCE/tmpfs-based swap device, and limited memory with RoCE/tmpfs based local swap device. We implemented a test suite of this micro-benchmark on these three system configurations and evaluated two key characteristics: Duration of use of different paging devices: Run "paging_w" on client node with 307GB, 358GB, 512GB, 768GB and 1TB data size to track your total runtime. Scaling Constraint: Evaluate whether the ratio (difference in lifetime) of full memory to swap device will converge to a certain level as the overcommit ratio increases.

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

RoCE/tmp f s v.s. FullMemory

	1 +C1	v.s.	1	
	Dpaging +C2	v.s.	1	n
+)	Dpaging +Cn	... v.s.	1	

$$1 + (n \quad 1) \qquad (Dpaging) + \sum_{k=1}^{n} Ck \qquad \text{v.s.} \qquad \frac{n}{\quad}$$

(1)

Each row represents the duration of running a fixed mount of the workload size; which is normalized to 1 where the workload is allowed to reside in physical memory. Dpaging represents the multiple of the paging latency between

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

Table -1: VoltDB-TPC-C-performance-comparison

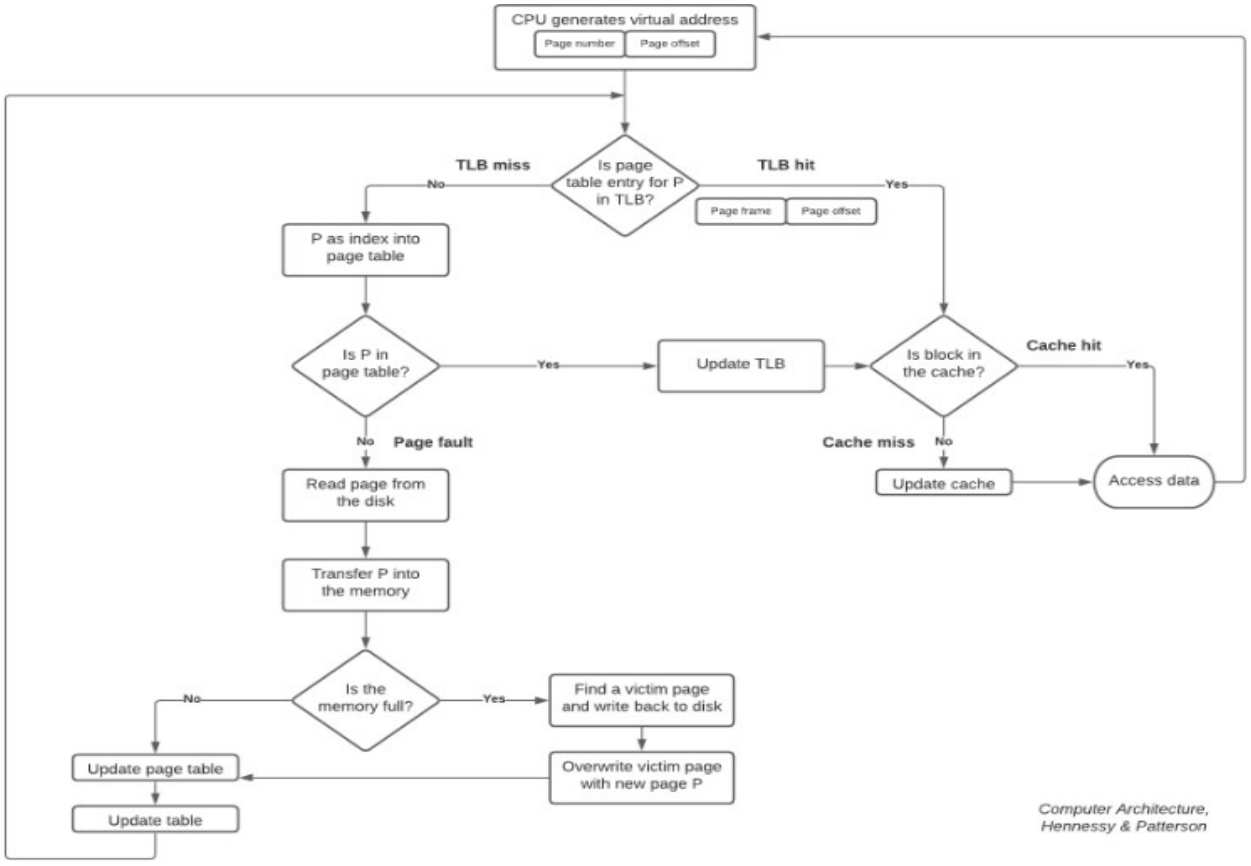

Table -2: Cache Miss, TLB Miss, and Page Fault

We use this value to model the runtime difference between using RoCE/tmpfs based swap devices and full memory configurations. Ck represents the cost of the VMM OS itself, which is assumed to be a relatively small number and can be ignored before reaching the overactive memory state (Denning 1968). Considering the "RoCE/tmpfs" configuration with 64 GB of physical memory: the client node runs a 256 GB workload (multiple overcommit = 4) and only the first 64 GB can reside in the physical memory with a normalized lifetime = 1; three more 64GB sections will be on top of the RoCE/tmpfs based swap device with lifetime = (S1+S2). In short, "RoCE/tmpfs based" requires total execution time = 1 + 3 (S1+S2), compared to 4 for "Full Memory"; which represents the rate of performance degradation where n = 4: n 1 +(n− 1)Dpaging +∑ n 4 1Ck → 1 +3(Dpaging) +∑ 4 Ck Furthermore, ∑ n Ck can be ignored and the degradation rate will approximate to 1 before reaching Dpaging the memory thrashing state where OS VMM overhead becomes significant (Denning 1968).

Industrial Benchmark: VoltDB/TPC-C

In addition to latency analysis and micro-benchmarking, we evaluated a commercial-grade open source in-memory database - VoltDB using the TPC-C "transactions per second" benchmark. For a full memory configuration, it is deployed on a separate node with 12 * 64GB = 768GB of physical memory; which can support "in-memory processing" to get full TPC-C benchmark data (around 400GB on our benchmark). For two hotswap configurations, deploy on a client server with 6 * 16GB = 96GB of physical memory and leverage RoCE/tmpfs or HDDbased hotspot to expand limited physical memory to support "on demand processing". memory" for the entire process. TPCC benchmark data (about 400 GB on our benchmark). Also, run the TPC-C benchmark every 30 minutes incremental and log information about total database size and average throughput (transaction /sec).Since TPC-C can only have one result set (database size and throughput) for any single run at a granularity of 30 minutes, we use a logarithmic approach to calculate the number of throughputs (BOLD in Table 4). that is, between the two closest valid sets of outputs.

5. ANALYSIS

Latency Measurement: LMbench

Average page fault latency: The output in Table 2 shows the relationship between the allocated file size and the average page fault latency of the local RAMDisk and RoCE/tmpfs-based block device (RDMA). On the local RAMDisk, latency remains around 0.94 microseconds regardless of file size; on RoCE/tmpfs-based block devices (RDMA), larger file sizes converge latency to a stable 3.83 microseconds; therefore, we choose 2GB as the target file size for the following experiments, as there is no significant difference after that. Compared to the 0.94us full memory (RAM) configuration, the RoCE/tmpfs-based block device (RDMA) simulated page fault latency is approximately 3.8us, leading to a multiple of 4 latency of page fault as (equation (1)) stated.

Overhead of CPU: The main reason to choose RoCE as the interconnect is that RDMA can offload network protocol processing from CPU to dedicated blocks inside NICs, achieve low latency requirement and facilitate our composable memory design. In Figure 2, the client node (blue) shows an approximately linear increase in CPU usage along with the additional processes running lat_pagefault(). During that time, the remote node (red; the memory pool) is still consuming about 3% CPU cycles when the total throughput of the 4 KB page swap (compatible with tmpfs on the remote node) grows up to 8x ; demonstrate scalability of RoCE/tmpfs-based composable memory.

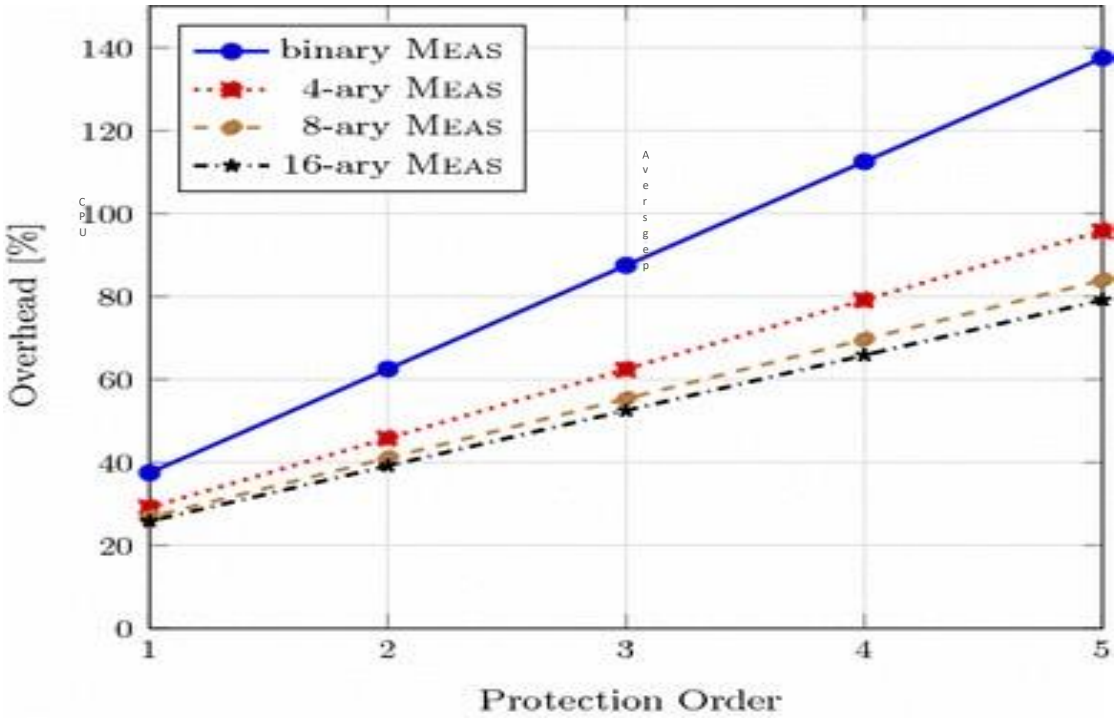

Figure -2: Memory overhead of Meas depending on arity and protection order

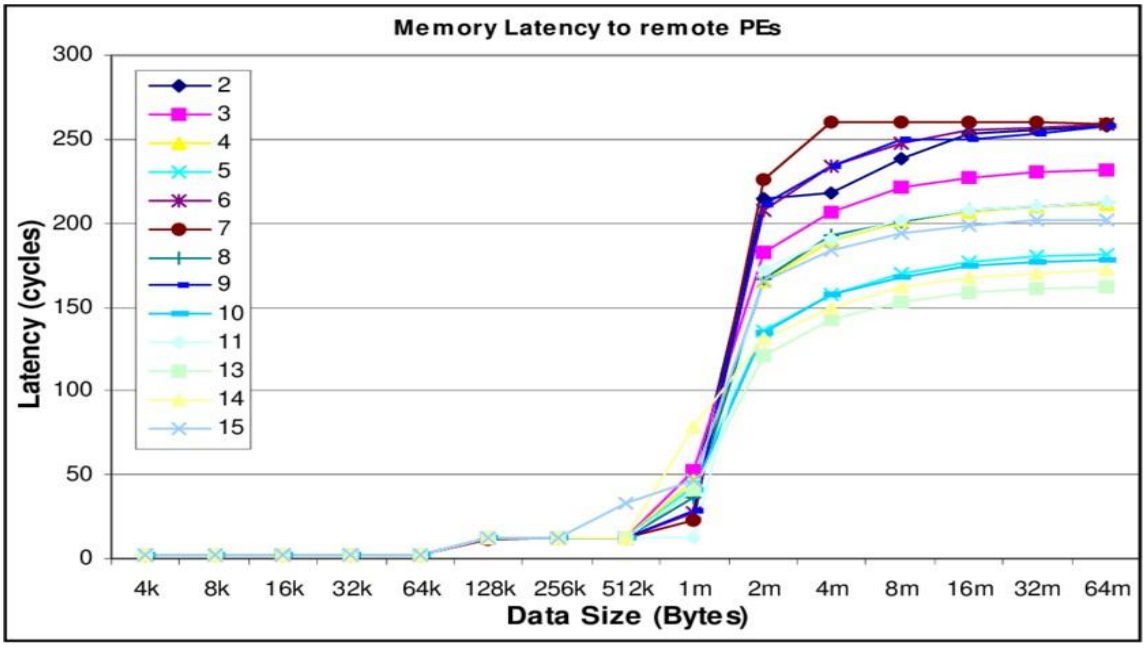

Figure -3: Memory overhead of Meas depending on arity and protection order

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 by 16.6%, 23.5%, 32.9% for devices based on blocks in RoCE/tmpfs (RDMA) and 22.9%, 26.9%, 52.8% for local RAMDisk (RAM). We can find that the latency ratio of RDMA to RAM decreases 4 to 3 times; This demonstrates an important finding: before hitting the limits of I/O limits, we can leverage parallelism on RoCE/tmpfs-based composable memory to deliver higher performance if its latency can meet the application requirements. In this case, we can increase the data exchanged to 800% with a latency degradation of 32.9% and achieve a 537% gain in total throughput. Micro-benchmark: Intensive-Write Duration of using different swap devices: Figure 4 shows the results of the Intensive-Write measurement in three configurations and using full memory as the baseline (100%). The vertical axis represents related performance versus full memory: DurationFullMemory DurationRoCE/tmp f s

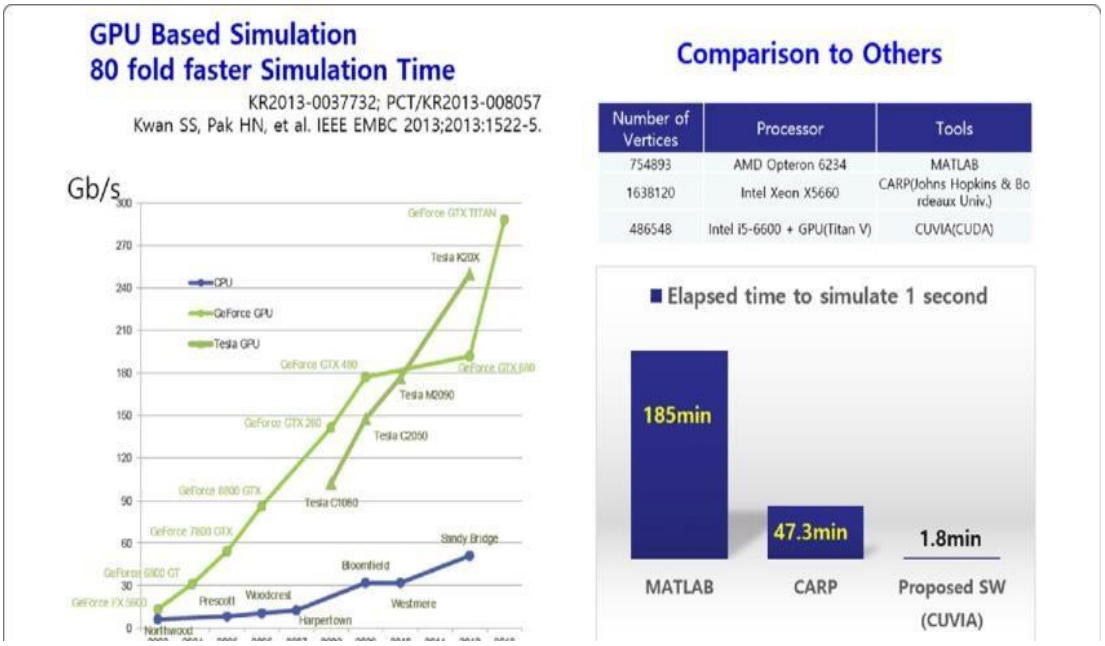

Figure -4: Memory bandwidth growth rate of CPU and GPU

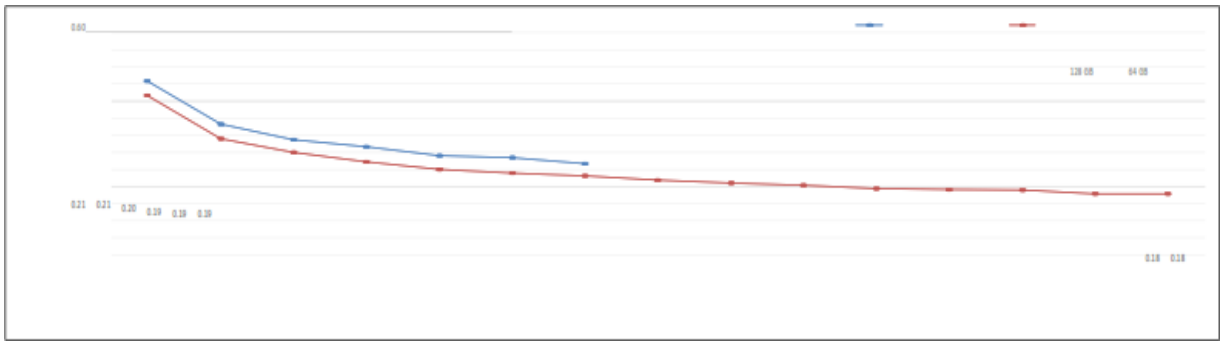

Figure 5: Limitation of Scale: Intensive-Write

Limitation of scale: Figure 5 shows the results on a larger scale. For 128GB configuration (blue), performance degradation is 45%, 35%, 31% with multiple overcommits 2, 3, 4; which work similar to the 200%, 300%, and 400% overcommit ratio in Figure 4. When we further expand the multiple of overcommit from 5 to 8 (800%; or run 1TB of data in addition to 128GB of memory physics), his performance drops by 29% to 25%. For the 64GB (red) configuration, its performance continues to decline with a more even rate of multiple overcommits from 9 to 16, dropping to the lowest number we can measure: 18%. We can see that there is a fixed 3-4% difference between 64GB (red) and 128GB (blue); which is because the OS can take up a fraction of the memory (available physical memory is smaller) and cause the overcommit multiple of 64GB (red) to be slightly larger than 128GB (blue). According to Table 2, the multiple of the page fault latency between the RoCE/tmpfs-based block device and the full memory is 4; which leads to the result of equation (1) as: n 1 + (n− 1)(4) + ∑ 4 Ck If we temporarily ignore the overhead of OS VMM ∑4 Ck and set n = 2, 4, 8 and 16, we can derive the theory degradations such 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 for OS effects). We can see, except for the multiple 16 (the difference is about 20%), all the experimental values are very close to the theoretical results (the difference is 0-10%), where the OS VMM overhead is ignored. Based on the results of the experiment, we can develop a hypothesis: before the multiple reaches 8, the degradation rate follows the prediction of Equation (1) where paging latency dominates and OS VMM overhead ∑n Ck can be ignored; Sense, Beyond this multiple of 8, OS VMM may need to be revised to further optimize RoCE/tmpfs-based composable memory.

CONCLUSION

This document reaffirms the known remote memory paging problem in the context of using COTS HW and SW components without modifying applications, libraries, operating systems, or hardware. With performance/scalability models based on LMbench results and assessments unchanged on VoltDB/TPC-C, we understand that: • For memory-intensive workloads, the proposed system deserves further study. Compared with local HDD, its swap latency of 4KB is much lower (Table 2) and thus can offer significant improvement for both micro reference system (Figure 4) and industrial reference system (Figure 6) that we have proved. • Relieving the problem of thrashing pursuant to art. 4.3 and Figure 6, the proposed system provides a new direction for distributing memory-intensive workloads on memory-poor nodes; which is impractical and never considered as an option. For example, VoltDB's official documentation suggests turning off virtual memory to maximize performance. • The performance and usability of the proposed system can benefit newly emerging research such as advanced memory management, application-centric memory overallocation, memory-driven processing, and TensorFlow by alleviating the limitations of memory overallocation; The key is to discover the degradation sensitivity and cost-performance matrix of applying Composable Memory to a specific workload. RoCE/tmpfs-based composable storage can further extend the concept of composable infrastructure. We believe that the combination of VMM OS and a high speed/low latency network has great potential for related research and can develop in many directions. First, we can analyze the performance bottleneck within the Linux kernel and discover opportunities for optimization, such as VMM overhead when memory overactivity is reached. Second, we can develop the cost-performance matrix for widely adopted memory-intensive workloads, which may impact contemporary data center design and public cloud (e.g., AWS) pricing model.

REFERENCES

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6966966/

https://www.techtarget.com/searchdatacenter/feature/Everything-you-need-to-know-about-composable-infrastructure

https://dl.acm.org/doi/10.1145/1065944.1065952

Memory Controllers for Real-Time Embedded System by Benny Akesson

https://docs.oracle.com/cd/E19455-01/806-4750/6jdqdflsg/index.html

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6555441/