Memtrade: Marketplace for Disaggregated Memory Clouds

Hasan Al Maruf
University of Michigan
USA

Yuhong Zhong
Columbia University
USA

Hongyi Wang
Columbia University
USA

Mosharaf Chowdhury
University of Michigan
USA

Asaf Cidon
Columbia University
USA

Carl Waldspurger
Carl Waldspurger Consulting
USA

ABSTRACT

We present *Memtrade*, the first practical marketplace for disaggregated memory clouds. Clouds introduce a set of unique challenges for resource disaggregation across different tenants, including resource harvesting, isolation, and matching. Memtrade allows producer virtual machines (VMs) to lease both their unallocated memory and allocated-but-idle application memory to remote consumer VMs for a limited period of time. Memtrade does not require any modifications to host-level system software or support from the cloud provider. It harvests producer memory using an application-aware control loop to form a distributed transient remote memory pool with minimal performance impact; it employs a broker to match producers with consumers while satisfying performance constraints; and it exposes the matched memory to consumers through different abstractions. As a proof of concept, we propose two such memory access interfaces for Memtrade consumers – a transient KV cache for specified applications and a swap interface that is application-transparent. Our evaluation shows that Memtrade provides significant performance benefits for consumers (improving average read latency up to 2.8×) while preserving confidentiality and integrity, with little impact on producer applications (degrading performance by less than 2.1%).

ACM Reference Format:
Hasan Al Maruf, Yuhong Zhong, Hongyi Wang, Mosharaf Chowdhury, Asaf Cidon, and Carl Waldspurger. 2023. Memtrade: Marketplace for Disaggregated Memory Clouds. 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.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3578338.3593553

1 MOTIVATION

Cloud resources are increasingly being offered in an elastic and disaggregated manner. Memory, however, is still largely provisioned statically, especially in public cloud environments. In public clouds, a user launching a new VM typically selects from a set of static, pre-configured instance types, each with a fixed number of cores and a fixed amount of DRAM. Although some platforms allow users to customize the amount of virtual CPU and DRAM, the amount remains static throughout the lifetime of the instance. Even in serverless frameworks, which offer elasticity and auto-scaling, a function has a static limit on its allocation of CPU and memory.

Long-running applications deployed on clouds are commonly highly over-provisioned relative to their typical memory usage. For example, cluster-wide memory utilization in Google, Alibaba, and Meta datacenters hovers around 40%–60%. Large-scale analytics service providers that run on public clouds, such as Snowflake, fare even worse – on average 70%–80% of their memory remains unutilized. Moreover, in many real-world deployments, workloads rarely use all of their allocated memory all of the time. Often, an application allocates a large amount of memory but accesses it infrequently. For example, in Google's datacenters, up to 61% of allocated memory remains idle [3]. In Meta's private datacenters, within a 10-minutes window, applications use only 30–60% of the allocated memory [7]. Since DRAM is a significant driver of infrastructure cost and power consumption, excessive underutilization leads to high capital and operating expenditures, as well as wasted energy (and carbon emissions). Although recent remote memory systems address this by satisfying an application's excess memory demand from an underutilized server [1, 2, 4, 6], existing frameworks are designed for private datacenters.

Furthermore, with the emergence of coherent interfaces like Compute Express Link (CXL), next-generation datacenter designs are moving towards tiered-memory subsystems [5, 7]. Servers within a rack can be connected through CXL switches and access each other memory. In such a system, CPUs of one server will have access to heterogeneous memory types with varied latency, bandwidth, and performance characteristics. While running multiple applications in such a rack-scale system, system-wide application-level performance will highly depend on an application's share to different memory tiers. Efficiently rightsizing different memory tiers, moving cold pages from faster to slower memory tiers, and matching harvested memories on different tiers to appropriate applications with a view to ensuring performance is challenging in such a disaggregated system.

In this regards, we introduce Memtrade that harvests both unallocated and allocated-but-idle application memory to realize a practical remote memory marketplace in both public clouds.

2 DISAGGREGATION CHALLENGES IN PUBLIC CLOUDS

Realization of Memtrade must address following challenges:

- **Immediately Deployable.** In many cases (e.g., public cloud setting, server configuration restrictions), a tenant cannot modify host-level system software which would require the operator to manage the remote memory service. In addition, prior

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/10.1145/3578338.3593553

work assumes the latest networking hardware and protocols (e.g., RDMA) [1–4, 6]; availability of these features in public clouds is limited, restricting adoption.

- **Efficient Harvesting.** Memory harvesting needs to be lightweight, transparent and easily deployable without impacting performance. Most prior work includes only a VM's *unallocated* memory in the remote memory pool. Leveraging idle application-level memory – allocated to an application but later unused or accessed infrequently – significantly enhances remote memory capacity. Existing *cold page detection*-based [3] proactive page reclamation techniques need significant CPU and memory resources, along with host kernel or hypervisor modifications [7].
- **Performant Consumption.** To ensure producer-side performance, Memtrade must return a producer's harvested memory seamlessly when needed. Memory offered to consumers may also disappear due to a sudden burst in the producer's own memory demand, or if a producer leaves unexpectedly. Memtrade needs to manage this unavailability to provide a high-performance memory interface.
- **Incentivization and Resource Matching.** Unlike prior work, which assumes cooperative applications, in a public cloud setting, we need to create a *market* where producers and consumers have monetary incentives to participate. Producers must be compensated for leasing memory, and the price must be attractive to consumers compared to alternatives (e.g., existing in-memory caching services or spot instances). In addition, producers have varied availability and bursty workload demands, while consumers may have their own preferences regarding remote memory availability, fragmentation, network overhead, and application-level performance, all which must be considered when matching producers to consumers.

3 MEMTRADE: OVERVIEW

Memtrade is an ubiquitous solution that allows users hop on any existing machine and readily deploy it without disrupting the running application(s) and modifying the underline kernel. It consists of three core components: **(i) producers**, which expose their harvested idle memory to the remote-memory market; **(ii) the broker**, which pairs producers with consumers while optimizing cluster-wide objectives, such as maximizing resource utilization; and **(iii) consumers**, which request remote-memory allocations based on their demand and desired performance characteristics.

Producers. A producer employs a collection of processes to harvest idle memory within a VM, making it available to the remote-memory market. A producer voluntarily participates in the market by first registering with the broker. The producer then monitors its resource usage and application-level performance metrics, periodically notifying the broker about its resource availability. When the broker matches a consumer's remote memory request to the producer, it is notified with the consumer's connection credentials and the amount of requested memory. The producer then exposes harvested memory through fixed-sized slabs dedicated to that consumer. A producer may stop participating at any time by deregistering with the broker.

Broker. The broker arbitrates between producers and consumers, matching supply and demand for harvested remote memory while considering consumer preferences and constraints. While Memtrade supports untrusted producers and consumers from diverse tenants, its logically-centralized broker component should be run by a trusted third party – such as a caching-as-a-service provider or the public cloud operator. The broker decides on the per-unit remote memory price for a given lease time, based on monitoring the current price of spot instances offered in the same public cloud. Appropriate pricing provides incentives for both producers and consumers to participate in the market; the broker receives a cut of the monetary transactions it brokers as commission.

Consumers. A consumer voluntarily participates in the remote-memory market by registering its connection credentials with the broker. Once approved by the broker, the consumer can submit a remote memory request by specifying its required remote memory, lease time, and preferences. After matching the request with one or more producers, the consumer then communicates directly with assigned producers through a simple KV cache `GET` / `PUT` / `DELETE` interface to access remote memory. We also implement a transparent remote-paging interface for the consumer. To ensure confidentiality and integrity of consumer data stored in producer memory, consumer interfaces offer an optional cryptographically access to remote memory in a transparent manner.

4 KEY RESULTS

We evaluate Memtrade using both synthetic and real-world cluster traces. The key results of our evaluation are as follow:

- Memtrade can harvest significant amounts of memory, even from right-sized VMs. A notable portion of the total harvested memory is extracted from the application's idle memory (on average, 1.1–51.4% across the entire workload) at a lower application-level performance degradation cost of 0–1.6%. Producer-side CPU and memory overheads due to the harvester are less than 1%.
- broker arbitrates between consumers and producers and benefits the both. In a simulated marketplace, involvement of broker increases the cluster-wide memory utilization from 56.8% to 97.9%, consumer hit ratios improve by a relative 18.2%, and the consumer's cost of renting extra memory reduces by an average of 82.1% compared to using spot instances.
- Memtrade improves consumer average latency by up to 2.8×, while impacting producer latency by less than 2.1%, and improves memory utilization (up to 97.9%).

REFERENCES

[1] I. Calciu, M. T. Imran, I. Puddu, S. Kashyap, H. A. Maruf, O. Mutlu, and A. Kolli. Rethinking software runtimes for disaggregated memory. In *ASPLOS*, 2021.
[2] J. Gu, Y. Lee, Y. Zhang, M. Chowdhury, and K. G. Shin. Efficient memory disaggregation with Infiniswap. In *NSDI*, 2017.
[3] A. Lagar-Cavilla, J. Ahn, S. Souhlal, N. Agarwal, R. Burny, S. Butt, J. Chang, A. Chaugule, N. Deng, J. Shahid, G. Thelen, K. A. Yurtsever, Y. Zhao, and P. Ranganathan. Software-defined far memory in warehouse-scale computers. In *ASPLOS*, 2019.
[4] Y. Lee, H. A. Maruf, M. Chowdhury, A. Cidon, and K. G. Shin. Hydra : Resilient and highly available remote memory. In *FAST*, 2022.
[5] H. Li, D. S. Berger, S. Novakovic, L. Hsu, D. Ernst, P. Zardoshti, M. Shah, S. Rajadnya, S. Lee, I. Agarwal, M. D. Hill, M. Fontoura, and R. Bianchini. Pond: CXL-based memory pooling systems for cloud platforms. 2023.
[6] H. A. Maruf and M. Chowdhury. Effectively Prefetching Remote Memory with Leap. In *USENIX ATC*, 2020.
[7] H. A. Maruf, H. Wang, A. Dhanotia, J. Weiner, N. Agarwal, P. Bhattacharya, C. Petersen, M. Chowdhury, S. Kanaujia, and P. Chauhan. TPP: Transparent page placement for CXL-enabled tiered memory, 2022.