# Incorporating Memory Sharing-awareness in Multi-VM Live Migration

Roja Eswaran, Mingjie Yan, and Kartik Gopalan
Computer Science, Binghamton University
Contact: {reswara1,myan28,kartik}@binghamton.edu

***Abstract*—One of the key challenges of edge computing is managing the limited resources available at the edge, especially memory and network bandwidth. Virtual machines (VMs) can ensure both isolation and efficient resource utilization within the edge computing infrastructure.**

**Live migration is a crucial technique in edge computing infrastructure to transfer running VMs from one physical node to another. This can occur either within the same host (Intra-host) or between different hosts (Inter-host). Current live migration techniques face challenges, such as lack of awareness of duplicated pages for inter-host migration and inefficient handling of co-located memory for intra-host migration.**

**In this paper, we describe our work on three efficient ways to incorporate sharing-awareness in live migration of multiple VMs while avoiding memory and network resource contention. For inter-host migration, our techniques rely on existing *Copy-On-Write* (COW) optimization performed by the host/hypervisor. This enables the transfer of a copy of the page only once and preserves existing COW sharing by remapping them at the destination. For intra-host migration, our technique implements a mechanism to identify shared pages and transfer their ownership (via a *userfaultfd*-based mechanism) instead of copying them. Besides reducing network traffic and memory footprint by eliminating unwanted copying, our techniques also result in a shorter total migration time, thereby freeing additional resources involved in migration as quickly as possible.**

***Index Terms*—Cloud Computing, Live Migration, Operating System, Deduplication, Copy-on-Write Page Sharing.**

## I. Introduction

Live migration facilitates the seamless transfer of active VMs from one physical node to another either within the same or different hosts. Inter-host live migration involves relocating a VM from one node to another within the same data center. During this process, the memory, CPU, and I/O states of the VMs are transferred. Since the disk images can be stored in a shared network file system, there is no need to transfer the disk. This technique is widely used for a variety of purposes, such as load balancing [1], [2], [3], infrastructure maintenance, meeting service level agreements [4], security-related updates, energy savings [5], hardware failure and seamless maintenance of physical edge nodes. Intra-host live migration involves relocating a VM within the same host from under the control of one emulation runtime instance (such as a QEMU process) to another instance, for reasons such as live runtime updates, bug fixes, and VM introspection [6], [7], [8].

The problem with the current inter-host live migration is that it is not aware of the COW memory sharing among pages of one or more VMs that might be migrated concurrently. As a result, a shared page may be transferred multiple times to the destination increasing the VMs' memory footprint, network traffic, and total migration time. Though intra-host live migration is performed within the same host, it may end up copying VMs' memory instead of only transferring the ownership of the pages. In multi-VM intra-host migration, COW-shared pages will be copied multiple times, as with inter-host migration. Our contributions are to make both single- and multi-VM live migrations aware of COW-shared memory in order to avoid unnecessary page transfers.

1) We identify and demonstrate the limitations of current inter-host and intra-host live migration techniques and motivate a better solution with awareness of COW-shared pages.
2) We present a Generic Template-aware Live Migration (Generic TLM) of Virtual Machines, which addresses the limitations of pre-copy live migration being unaware of the COW sharing from templated VMs. We also present a more general approach, called Sharing-aware Live Migration (SLM) that identifies COW sharing among VMs irrespective of the underlying memory optimization techniques, such as Kernel Same-page Merging (KSM), VM Templating, and others.
3) We present an Intra-host TLM that efficiently identifies COW-shared memory among multiple co-located VMs and transfer their page ownership instead of copying them.
4) We implemented a prototype of Generic TLM, SLM, and Intra-host TLM in the KVM/QEMU [9] virtualization platform and evaluated it using several benchmarks. Besides reducing the memory footprint, our techniques also significantly reduce the total migration time by up to 95%, 60%, 85% respectively.

This doctoral symposium paper summarizes results from our prior publications in SEC EdgeComm 2023 [10] and CCGrid-2024 [11] besides introducing our ongoing work on intra-host TLM. In the rest of this paper, we briefly describe our main contributions in Generic TLM, SLM, and Intra-host TLM, and a summary of the results.

## II. Inter-host Live Migration

### A. Generic Template-aware Live Migration (Generic TLM)

In this section, we first describe the background of the existing COW technique, VM Templating. We then discuss

979-8-3503-9566-2/24/$31.00 ©2024 IEEE
DOI 10.1109/CCGrid59990.2024.00084

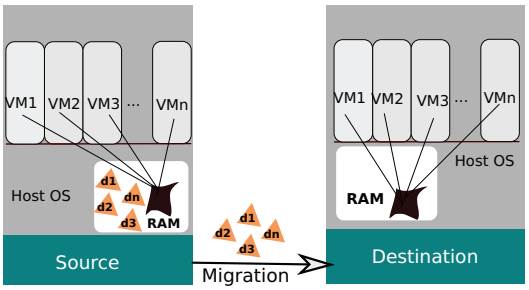



Fig. 1: Template-aware Live migration works by migrating only the *delta* pages during migration. The shared VM template is available to the destination either over a networked storage or transferred ahead of time before migration begins.

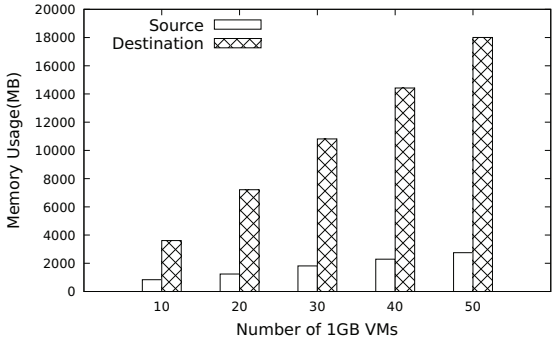



Fig. 2: Pre-copy Live Migration is unaware of the COW-shared base template and breaks the sharing at the destination.

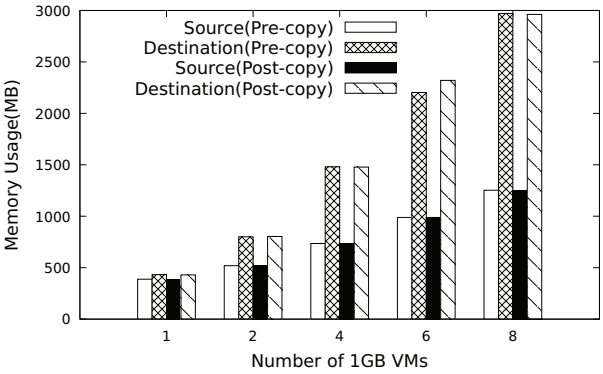



Fig. 3: Memory footprint of VMs expands at destination after both pre-copy and post-copy live migration, because pages shared among VMs at the source are replicated for each VM at the destination.

the limitation of Pre-copy live migration, which is unaware of VM Templates. Finally, we conclude with our prototype Generic TLM, which retains VM Template sharing during live migration.

**VM Templating:** Instantiating a VM from scratch typically takes a long time because it involves initialization of software, guest OS, and virtual hardware, including time to load the corresponding contents to memory from the disk. VM templating allows multiple new VM instances to be quickly instantiated from a single pre-checkpointed VM image (or template). Figure 1 (Source) shows that multiple templated VMs can be booted from a shared VM template which is COW-mapped into each VM instance. Any additional memory dirtied by the templated VMs (called *delta*) is stored separately for each VM instance.

**Limitation of Pre-copy Live Migration:** In Figure 2, we show the problem that arises during the migration of templated VMs using pre-copy. The X-axis represents the number of VMs instantiated from a shared template that are being migrated, and the Y-axis represents their collective memory footprint as measured by the `free` command in Linux. Prior to migration, we measured the memory usage of the templated VMs at the source. Subsequently, we measured memory usage again at the destination after migration. Since the traditional pre-copy live migration is unaware of page sharing among templated VMs, it redundantly transfers and replicates identical pages, breaking the underlying COW sharing.

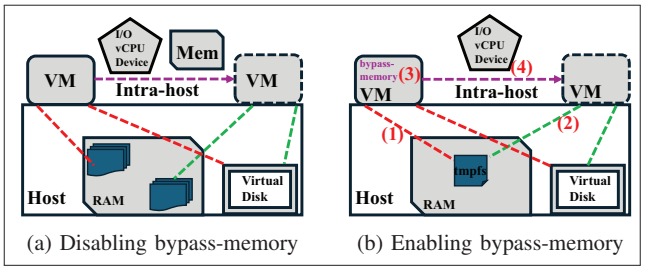
**Generic TLM:** We proposed Generic Template-aware Live Migration [10], as shown in Figure 1, which addresses this shortcoming of pre-copy migration by ensuring that multiple templated VM instances maintain their COW page sharing with the base template even at the destination node and transfer only the delta pages that differ among various VM instances. Generic TLM requires that we first track delta (or dirty) pages for VM instances before migration. Then, during migration, only the delta pages are transferred for each instance. Generic TLM, besides preventing memory footprint expansion, also reduces the total migration time by up to 95.37% and network traffic by up to 92.15%.

### *B. Sharing-aware Live Migration (SLM)*

In this section, we first describe the background of the existing COW technique, Kernel Samepage Merging. We then discuss the limitations of Pre-copy, Post-copy, and Generic TLM. Finally, we conclude with our prototype SLM, which retains all COW page sharing for all types of VMs.

**Kernel Samepage Merging:** Deduplication techniques, such as Kernel Samepage Merging [12] (KSM), perform memory deduplication among co-located VMs to fit more VMs into physical memory. Many identical pages exist when the same guest OS and applications run in different co-located VMs, consuming extra memory. KSM regularly scans the memory of all VMs, identifies identical pages, and replaces them with a single COW-shared page.

**Limitation of Pre-copy, Post-copy and Generic TLM:** Figure 3 shows that both pre-copy and post-copy, which are unaware of existing COW-shared pages among VMs, result in a larger memory footprint at the destination than at the source after live migration completes. While the Generic TLM approach works well in efficiently migrating multiple




(a) Disabling bypass-memory (b) Enabling bypass-memory

Fig. 4: (a) Without the `bypass-memory` flag, the memory of the VM is copied again. (b) The `bypass-memory` flag skips copying the memory and only transfers I/O, vCPU, and Device states.

templated VMs, we realized that the problem of sharing-awareness in live migration extends beyond just templated VMs. Specifically, Generic TLM does not account for pages shared among VMs due to other memory sharing mechanisms besides templating, such as memory deduplication performed by KSM in Linux, or simple COW mappings due to process fork and file I/O operations.

**SLM:** We proposed Sharing-Aware Live Migration (SLM) [11] to address the above problem. The key insight behind SLM is that irrespective of the underlying page-sharing mechanism, multiple COW-mapped guest pages will map to the same page in the physical memory. SLM examines the Physical Frame Number (PFN) and Virtual Page Number (VPN) of each guest page being transferred and classifies the pages into three types. Unique Pages are those that have not been transferred yet. Duplicate Pages are those that have already been transferred. Dirty Pages are those that require re-transmission due to being dirty in the previous pre-copy round. For Unique and Dirty pages, SLM transfers the entire page, including the page type and its PFN at the source, as a unique identifier. However, for Duplicate Pages, SLM does not send the page content, instead it only sends the page type and the PFN. At the destination, SLM COW maps the Duplicate Pages to the corresponding common page thus retaining the COW-shared mappings the same as the source. Besides preserving all pre-existing page sharings at the destination machine, SLM reduces the total migration time by up to 59% and network traffic by up to 62%.

## III. Intra-host Live Migration

Intra-host live migration involves relocating a VM within the same host from under the control of one emulation runtime instance (such as a QEMU process) to another instance, for reasons such as live runtime updates, bug fixes, and VM introspection [6], [7], [8].

**Limitations of existing techniques**: Using the traditional pre-copy live migration is inefficient for migration within the same host as the existing memory pages are duplicated instead of transferring the ownership as shown in Figure 4 (a). Although using the backend-file with the help of `bypass-memory` flag can transfer the ownership of pages

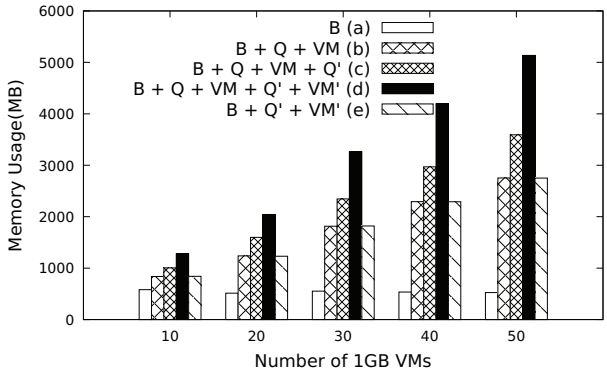



Fig. 5: Generic TLM is inefficient for migrating multiple templated VMs within the same host because it copies already existing delta pages instead of transferring their ownership thereby causing a memory bump (c)(d).

instead of copying them for regular non-templated VMs as shown in Figure 4 (b), it cannot be applied to templated VMs as it breaks the COW-shared base template after migration. Using Generic TLM still requires copying the delta pages generated by multiple templated VM instances. In Figure 5, we show the problem that arises during the migration of templated VMs using Generic TLM within the same host. The X-axis represents the number of VMs instantiated from a COW-shared template that is being migrated, and the Y-axis represents their collective memory footprint as measured by the `free` command in Linux. The legend of the graph provides the following explanations: (a) represents the system memory before executing any QEMU (Q) or VM. (b) signifies the system memory usage after booting a VM using Q. (c) illustrates the system memory usage after initiating a new Q' in listening mode. (d) showcases the system memory usage after live migration, where Q, Q', VM, and VM' are all alive. Finally, (e) displays the system memory usage after terminating Q and VM, leaving only Q' and VM' alive. When using Generic TLM for live migrating templated VMs within the same node, there is an observable rise in the memory bump (c) and (d) as shown in Figure 5, due to the duplication of delta pages.

**Intra-host TLM**: As shown in Figure 6, Intra-host TLM tracks delta pages diverging from the base template and transfers their page ownership to the destination templated VMs, preventing unnecessary copying of additional dirtied pages. *Userfaultfd* - a Linux mechanism to handle page faults in user space - is used by QEMU to continuously monitor for any write events and redirect them to a dedicated memory backend-file. During migration, the source only transfers the backend-file offset for each page without copying the page content during the downtime. The destination then remaps the virtual address of the pages to the corresponding location at the backend-file using the received offsets.

**Evaluation:** We evaluated the performance of Intra-host TLM using idle templated VMs. Our experimental setup

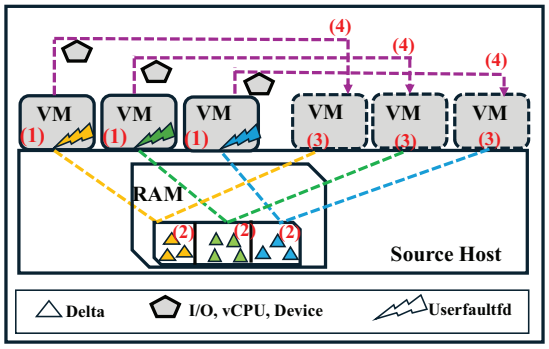



Fig. 6: High-level overview of Intra-host TLM which uses userfault-based mechanism to transfer the ownership of delta pages thus avoiding unnecessary copying overhead.

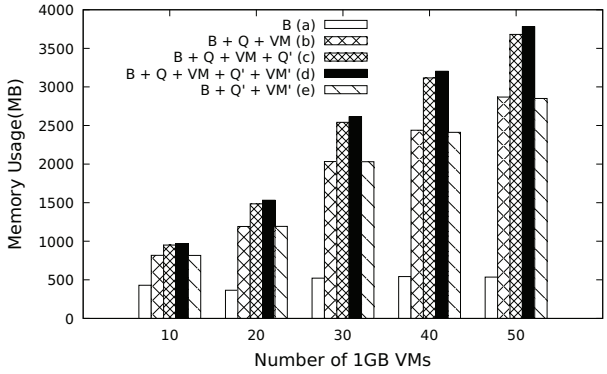



Fig. 7: Intra-host TLM efficiently transfers the ownership of delta pages within the same host for multiple templated VMs thereby eliminating the copy overhead and removing the memory bump (c)(d).

consists of three machines with two Intel Xeon E5-2620 v2 processors and 128GB DRAM. We implemented Generic and Intra-host TLM versions of pre-copy in the KVM/QEMU [9] virtualization platform on Linux. We modified QEMU's traditional pre-copy algorithms, with no changes to the guest operating system. Each experiment was repeated at least five times on idle templated VMs to compute average values.

As shown in Figure 5 and Figure 7, we have multiple metrics involved in this evaluation for Generic and Intra-host TLM: (a) represents the system memory before executing any QEMU (Q) or VM. (b) signifies the system memory usage after booting a VM using Q. (c) illustrates the system memory usage after initiating a new Q' in listening mode. (d) showcases the system memory usage after live migration, where Q, Q', VM, and VM' are all alive. Finally, (e) displays the system memory usage after terminating Q and VM, leaving only Q' and VM' alive.

Though Generic TLM failed to eliminate the delta bump (c)(d) due to the duplication of delta pages as shown in Figure 5, Intra-host TLM eliminated the delta bump (c)(d) by transferring the ownership and avoiding unnecessary copying as shown in Figure 7. The slight increase in (d) is attributed to the size of destination QEMU Q' not because of the copying overhead.

## 4. Conclusion

In this doctoral symposium report, we addressed the problem of traditional pre-copy live VM migration techniques being unaware of memory sharing within the same or different hosts, resulting in higher memory and network resource contention. For effective inter-host migration, we presented Generic TLM and SLM, which rely on existing Copy-On-Write (COW) optimizations such as VM templating, KSM, fork, and others performed by the host/hypervisor, and preserve the COW-shared mapping at the destination after migration. Additionally, we introduced Intra-host TLM, which transfers the ownership of the COW-shared base template and additional delta pages diverging from them without introducing unnecessary copy overhead for effective intra-host migration. Our evaluation of Generic TLM, Intra-host TLM, and SLM shows significant reductions in memory footprint, as well as quicker release of migration resources by minimizing migration time.

## References

[1] W. Attaoui, E. Sabir, H. Elbiaze, and M. Guizani, "Vnf and cnf placement in 5g: Recent advances and future trends," *IEEE Transactions on Network and Service Management*, 2023.

[2] A. Ruprecht, D. Jones, D. Shiraev, G. Harmon, M. Spivak, M. Krebs, M. Baker-Harvey, and T. Sanderson, "VM live migration at scale," *ACM SIGPLAN Notices*, 2018.

[3] D. Fernando, J. Terner, K. Gopalan, and P. Yang, "Live migration ate my VM: Recovering a virtual machine after failure of post-copy live migration," in *Proc. of IEEE Conference on Computer Communications Workshops Annual Computer Security Applications Conference*, 2019.

[4] S. Mubeen, S. A. Asadollah, A. V. Papadopoulos, M. Ashjaei, H. Pei-Breivold, and M. Behnam, "Management of service level agreements for cloud services in IoT: A systematic mapping study," *IEEE access*, 2017.

[5] A. Verma, P. Ahuja, and A. Neogi, "pMapper: Power and migration cost aware application placement in virtualized systems," in *Proc. of Middleware*, 2008.

[6] A. Ho, M. Fetterman, C. Clark, A. Warfield, and S. Hand, "Practical taint-based protection using demand emulation," in *Proc. of the ACM European Conference on Computer Systems (EuroSys)*, 2006.

[7] P. Dovgalyuk, N. Fursova, I. Vasiliev, and V. Makarov, "Qemu-based framework for non-intrusive virtual machine instrumentation and introspection," in *Proc.of the Joint Meeting on Foundations of Software Engineering*, 2017.

[8] J. Wei, L. K. Yan, and M. A. Hakim, "Mose: Live migration based on-the-fly software emulation," in *Proc. of the 31st Annual Computer Security Applications Conference*, 2015.

[9] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori, "kvm: the linux virtual machine monitor," in *Proc. of the Linux Symposium*, 2007.

[10] R. Eswaran, M. Yan, and K. Gopalan, "Template-aware live migration of virtual machines," in *In Proc. of ACM/IEEE Symposium on Edge Computing (SEC) Workshop on Edge Computing and Communications (EdgeComm)*, 2023.

[11] ——, "Tackling memory footprint expansion during live migration of virtual machines," in *Proc. of International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, 2024.

[12] I. Eidus and H. Dickins, "Kernel Samepage Merging," https://www.kernel.org/doc/Documentation/vm/ksm.txt, 2009.