GI-Fachgruppentreffen 2013 - Proposed Talk:

XLH: More effective memory deduplication scanners through cross-layer hints

Konrad Miller Fabian Franz
Marc Rittinghaus Marius Hillenbrand Frank Bellosa

Karlsruhe Institute of Technology (KIT)

In cloud computing, virtual machines (VMs) permit the flexible allocation and migration of services as well as the consolidation of systems onto fewer physical machines, while preserving strong service isolation. However, in that scenario the available main memory size limits the number of VMs that can be colocated on a single machine.

Prior work has shown great opportunity for memory deduplication. There may be plenty of redundant data between VMs (inter-vm sharing), e.g., if similar operating systems (OSes) or applications are used in different VM instances. Moreover, previous studies have shown that the memory footprint of VMs often contains a significant amount of pages with equal content within a single instance (self-sharing) [3]. In both cases, memory can be freed by collapsing redundant pages to a single page and sharing it in a copy-on-write fashion. However, such pages cannot be identified using traditional sharing mechanisms due to the semantic gap [5] caused by the isolation of VMs.

Two techniques have been used in the the past to make the detection of such sharing opportunities and thus deduplication of redundant pages possible.

Paravirtualization closes the semantic gap through establishing an appropriate interface between host and guest [4, 6] to communicate semantic information. This implies modifying both host and guest.

Such an interface has previously been used to help deduplicating *named* memory pages—memory pages backed by files: Satori [6] successfully merges named pages in guests employing sharing-aware virtual block devices in Xen [2]. Paravirtualization-based approaches have only been used selectively and rudimentarily to make sharing of *anonymous* memory (e.g., heap/stack memory) possible, through hooking calls such as `bcopy` [4].

Applying these modifications to all guests and keeping them compatible with latest developments at kernel and hypervisor level is at least a great burden. It might not even be possible at all to modify commercial or legacy guests due to license restrictions or the lack of source code. Moreover, the lack of semantic information that the host has about guest activities is actually one of the key features of virtualization: The host does not know nor needs to know the OS, file system, etc. inside the VM.

Memory scanners mitigate the semantic gap by scanning for duplicate content in guest pages [7, 1]. They index the contents of memory pages at a certain rate, regardless of their usage semantics.

Scanners have their downside when it comes to efficiency. Especially the merge latency, the time between establishing certain content in a page and merging it with a duplicate, is higher in systems based on content scanning compared to paravirtualization-based systems that merge pages synchronously when they are established.

Memory scanners trade computational overhead and memory bandwidth with deduplication success and latency. Current scanners need a considerable amount of time to detect new sharing opportunities (e.g., 5 min) and therefore do not exploit the full sharing potential. In our analyses, we have confirmed these results; however, we have found memory scanners to work well only for deduplicating fairly static memory pages.

XLH is our contribution that combines the key benefits of both previous approaches. We have observed that:

- All types of memory contents (named and anonymous) contribute to memory redundancy.
- Many shareable pages in the host's main memory originate from accesses to background storage: when multiple VMs create or use the same programs, shared libraries, configuration files, and data from their respective virtual disk images (VDIs).

The main contribution is to observe guest I/O in the host and to use it as a trigger for memory scanners in order to speed up the identification of new sharing opportunities. For this purpose, XLH generates page hints in the host's virtual file system (VFS) layer, whenever guests access their background store. XLH then indexes these hinted pages soon after their content has been established and thus moves them earlier into the merging stage. In consequence, XLH can find short-lived sharing opportunities and shares redundant data longer than regular, linear memory scanners without raising the overall scan rate.

Figure 1 depicts the significance of the merge latency on how many pages are shared at any given point in time: The later memory is indexed by the scanner, the later a shared page can be established. Indexing sharing opportunities earlier adds to the sharing potential and to longer sharing time-frames.

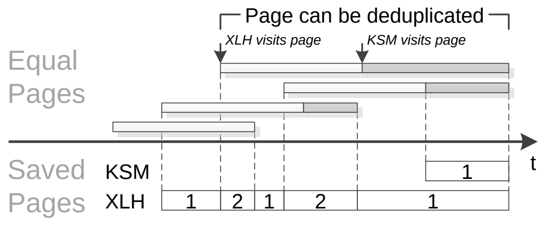

Figure 1: Linear or random memory scanners index pages after an expected value of half a scan cycle. XLH visits I/O pages immediately after they are established. If a duplicate is not found until a large proportion of the mean sharing time is over, the deduplication effectiveness is lowered significantly.

We only modify the *host* in our approach—XLH would not benefit from and thus does not make use of paravirtualization. In fact, due to the generality of our approach, XLH also works for deduplicating native processes when no virtualization is involved. Note that XLH does not solely target disk accesses but issues hints for all I/O that goes through the VFS interface, including network file systems such as NFS. Overall, I/O-advised scanning makes more effective detection of sharing opportunities possible without the need to modify guests.

We have implemented our approach in Linux' Kernel Samepage Merging (KSM) and evaluated its properties. We have shown that XLH is able to quickly deduplicate the memory of newly booted VMs, which is especially beneficial when sandboxing short-running jobs or migrating many VMs at once. Measurements of kernel build and web server scenarios show that XLH deduplicates equal pages that stem from the VDI earlier by 2-4 minutes and is capable of merging between up to 10x as many sharing opportunities than the baseline system. For the kernel build benchmark, XLH performs constantly better than KSM even if the scan rate is set 5x lower. We have also evaluated XLH in an unfavorable scenario and found that it did not worsen the sharing performance compared to KSM. XLH reaches its effectiveness with little to no additional CPU overhead or loss in I/O throughput compared with KSM.

The proposed talk will cover our findings in the area of main memory duplication, main memory deduplication techniques in general, as well as our contribution XLH with its properties. It will also give an outlook on future research directions in the area of main memory deduplication.

References

[1] ARCANGELI, A., EIDUS, I., AND WRIGHT, C. Increasing memory density by using KSM. In *Linux Symposium 2009*.

[2] BARHAM, P., DRAGOVIC, B., FRASER, K., AND ET AL. Xen and the art of virtualization. SOSP 2003.

[3] BARKER, S., AND ET AL. An empirical study of memory sharing in virtual machines. In *USENIX ATC 2012*.

[4] BUGNION, E., DEVINE, S., GOVIL, K., AND ROSENBLUM, M. Disco: running commodity operating systems on scalable multiprocessors. *Transactions on Computer Systems 1997*.

[5] CHEN, P. M., AND NOBLE, B. D. When virtual is better than real. In *HotOS 2001*.

[6] MIŁÓS, G., MURRAY, D. G., HAND, S., AND FETTERMAN, M. A. Satori: Enlightened page sharing. In *USENIX ATC 2009*.

[7] WALDSPURGER, C. A. Memory resource management in VMware ESX server. *SIGOPS Operating System Review 2002*.