Semantic Information of Deduplicated Pages in Virtual Machines

Summary— **The efficacy of the memory subsystem often determines the overall system performance, especially for applications that access a huge amount of data. A well-designed memory subsystem should, in one hand try to reduce the memory footprint by avoiding storing unnecessary data and at the same time, reduce the cost of access to data. The OS and the hypervisor plays a critical role in achieving the above goals. Content Aware Page Deduplication techniques, such as KSM, tries to reduce the memory footprint by merging pages with duplicate contents.**
We have build a tool to get semantic information of merged pages in order to get an idea of what pages are getting merged, what percentage of them are intra-vm merging and how much are inter-vm merging.

I. Introduction

A. Types of pages

Our tool gets semantic information of merged pages in order to get an idea of what pages are getting merged. The types of pages are being characterized as under:

1) Kernel pages: There are the pages which are a part of kernel data structures or allocated by kernel threads.

2) Page cache: The page cache is the main disk cache used by the Linux kernel. In most cases, the kernel refers to the page cache when reading from or writing to disk. New pages are added to the page cache to satisfy User Mode processes's read requests. If the page is not already in the cache, a new entry is added to the cache and filled with the data read from the disk. If there is enough free memory, the page is kept in the cache for an indefinite period of time and can then be reused by other processes without accessing the disk.

3) Anonymous pages: These are memory mapped pages that are not part of any files.

4) Buddy pages: There are free memory blocks managed by the buddy system allocator.

B. Types of Sharing

We separate the sharing arising into two different categories: intra-VM and inter-VM.

1) Intra-VM sharing: Memory deduplication within a single virtual machine, i.e. when different pages belonging to the same virtual machine are found to have same content and merged, is termed as intra-VM sharing. Such pages contribute to Intra-VM merging percentage.

2) Inter-VM sharing: Memory deduplication arising across virtual machines are called inter-VM sharing. When pages belonging to different virtual machines are found to have same content and merged, its called inter-VM merging. Such pages contribute to Inter-VM merging percentage.

II. Experimental Setup

The experiments were performed on an Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz CPU, having 32 GB of memory. KVM was used as hypervisor. Each Virtual Machine was given 10 GB of memory.

A. Tool

We have built a tool to get the semantic information of pages merged by KSM. The semantics are characterized by kernel pages, page cache pages, anon pages and pages with buddy allocator. It uses /proc file system interfaces to get statistics of the pages being currently allocated by the virtual machine. For the KSM pages, we get the guest frame number with the help of a kernel module. These guest frame numbers are sent to the guest machine to generate the semantic details of the pages referred to by those physical addresses using a kernel module. This tool makes heavy usage of files while generating semantic information. This affects the readings taken as these pages gets included in the list of merged pages. We have excluded the page cache that is generated because of this tool.

B. Workloads

We have setup two virtual machines starting nearly at the same instant.

1) Bare Virtual Machines: No workloads run on these machines.

2) Redis: Redis instance is started as server on both VMs and is loaded with exactly the same content.

3) Apache: We have setup apache server to serve random files in both the virtual machines. The same set of files is present in both the servers. Hey[1] workload generator instance generates the load on both the servers. It generates a total of 2000 queries with each query being generated after an interval of 2 seconds.

4) Graph500: Graph500 benchmark performs breadth-first-search over undirected graphs. We run the benchmark around the same time on both the Virtual Machines.

5) MySQL: Each Virtual Machine starts MySQL instance as server and loads inmemory tables with exactly the same data on both. The data is being populated by files.

III. Analysis of Memory Deduplication

A. Deduplication Semantics

The semantic information of pages deduplicated varies across workloads. The types of pages getting merged depend entirely on the workloads running in the Virtual Machines. In our experiment, we run same workload on two different VMs to get the semantic information of the pages being

merged by KSM. We observe that the type of pages getting merged depends entirely on the applications running on the VMs. In the case of bare VMs (Fig 1), most of the deduplication comes from kernel pages and pagecache. In this case, the page cache mostly consists of library files, system default daemons (such as systemd, journald) and binary files. In the case of Apache (Fig 3), most of the deduplication comes from page cache. This was expected because of the setup of our workload, which serves a random set of files on requests. In Redis and MySQL workloads, we populate the databases from files in the disk. Hence, initially page cache and anonymous pages being merged increases. After the database gets filled, there is not much increase in the anonymous pages being merged. From this point of time, the number of pagecache pages that are being merged also starts decreasing. Most of the deduplication comes from anonymous pages in Redis (Fig 2), MySQL (Fig 6) and Graph500 (Fig 4) workloads. We conducted the experiment of graph500 workload without excluding the page cache pages being generated as a result of our tool. Fig 5 shows us the graph in this case. We can see from the graph that at the same time when the workload ends, the anonymous pages merged are broken and buddy pages being merged increases sharply and then keeps in decreasing. In turn, the page cache pages getting merged keeps on increasing.

We also conducted experiment expecting merged pages to break. We ran redis instance on both the virtual machines, filled them up with exactly the same content. We then updated each redis key on one of the Virtual Machine with random value. This will start breaking the already merged pages. Fig 7 shows us what exactly happens in this case. The anonymous pages are first merged, and then starts breaking.

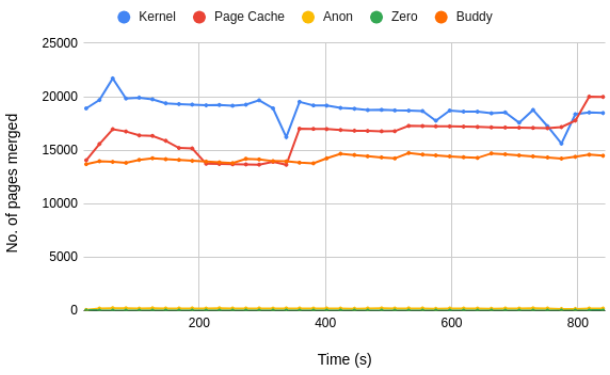

Fig. 1. Bare VMs

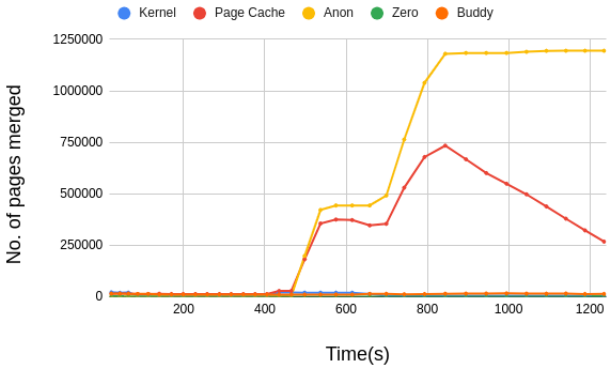

Fig. 2. Redis

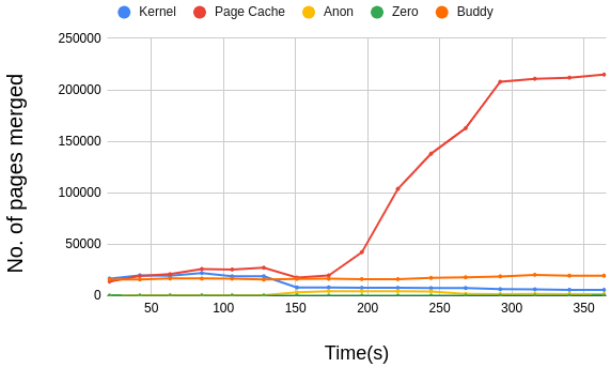

Fig. 3. Apache

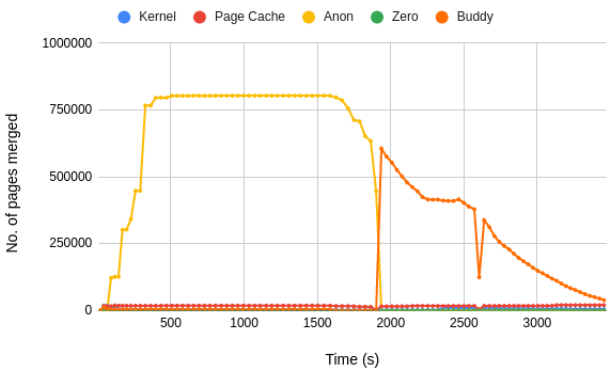

Fig. 4. Graph500

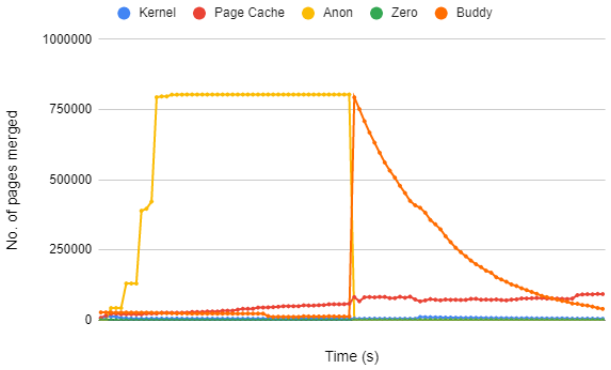

Fig. 5. Graph500: Without removal of page cahe pages being generated by tool

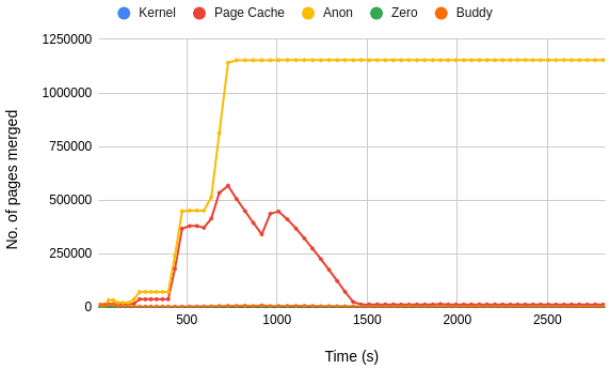

Fig. 6. MySQL

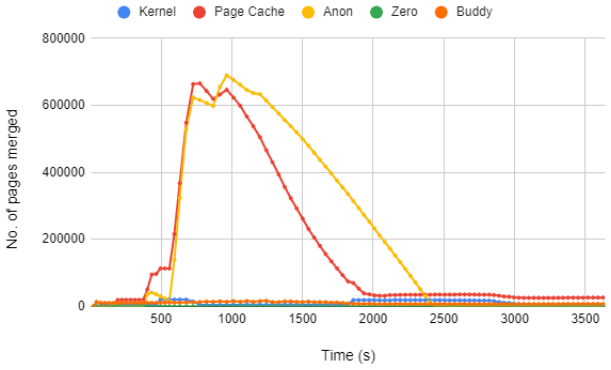

Fig. 7. Redis: Breaking of merged pages

B. *Inter vs Intra merging of pages*

When a VM boots, most of the pages that gets merged are intra-VM. But as the applications start in the Virtual Machine, the inter-VM merging percentage increases. While running two bare Virtual Machines, the intra-VM merging percentage is always higher than the inter-VM merging percentage. Whereas, if we consider Redis (Fig 9), Apache (Fig 10), MySQL (Fig 12) or Graph500 (Fig 11) workloads, initially the intra-VM merging percentage is higher. As the applications start, the intra-VM merging percentage decreases and inter-VM merging percentage increases. This was expected to happen as we are running homogeneous workloads on the Virtual Machines. Our workloads are such that it fills up the memory with identical content. So, once the workloads starts on the virtual machines, the inter-VM merging is bound to happen, which in turn increases the percentage of inter-VM merging.

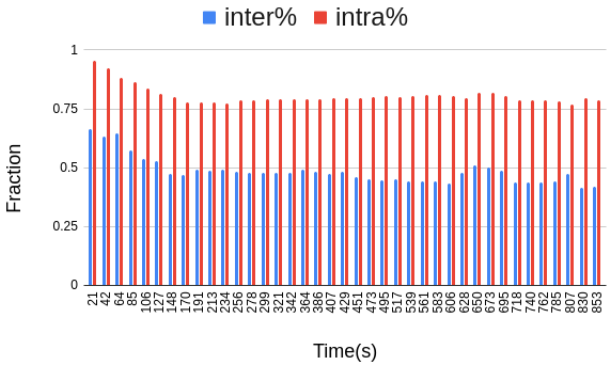

Fig. 8. Bare VMs: Inter-VM vs Inta-VM merging

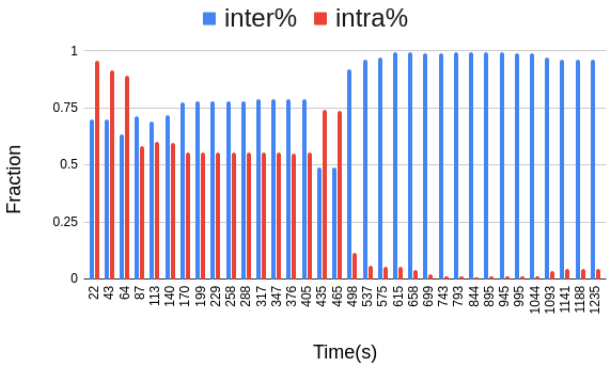

Fig. 9. Redis: Inter-VM vs Inta-VM merging

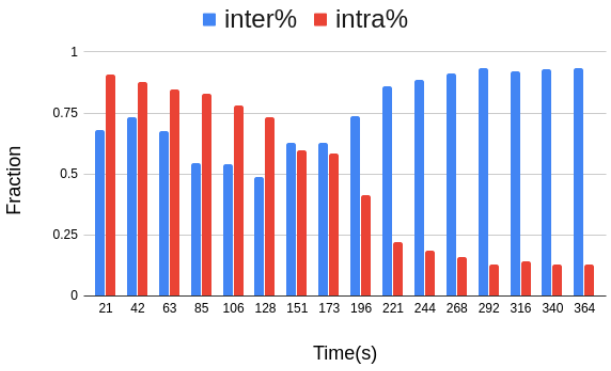

Fig. 10. Apache: Inter-VM vs Inta-VM merging

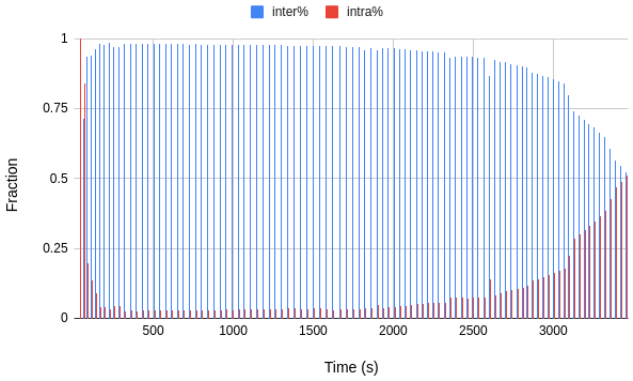

Fig. 11. Graph500: Inter-VM vs Inta-VM merging

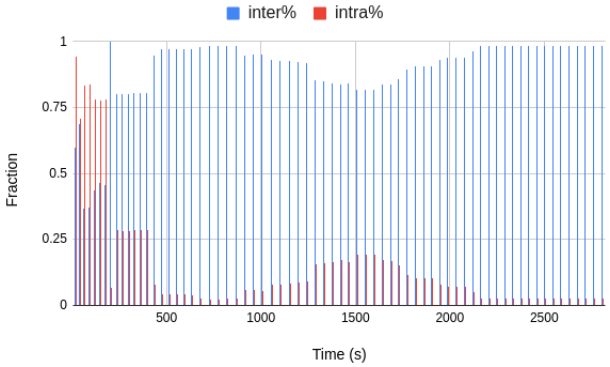

Fig. 12. MySQL: Inter-VM vs Inta-VM merging

IV. Conclusion

We can clearly see that the semantic information of pages deduplicated varies across workloads. The types of pages getting merged depend entirely on the workloads running in the Virtual Machines. When a VM boots, most of the pages that gets merged are intra-VM. But as the applications start in the Virtual Machine, the inter-VM merging percentage increases. When homogeneous workloads are running on different virtual machines with same memory footprint, then a large percentage of merging is across VMs.

References

[1] Hey HTTP load generator https://github.com/rakyll/hey.git
[2] The Page Cache https://www.oreilly.com/library/view/understanding-the-linux/0596005652/ch15s01.html
[3] Pagemap, from the userspace perspective https://www.kernel.org/doc/Documentation/vm/pagemap.txt