Kernel Samepage Merging

Overview

KSM is a memory-saving de-duplication feature, enabled by CONFIG_KSM=y, added to the Linux kernel in 2.6.32. See `mm/ksm.c` for its implementation, and http://lwn.net/Articles/306704/ and https://lwn.net/Articles/330589/

KSM was originally developed for use with KVM (where it was known as Kernel Shared Memory), to fit more virtual machines into physical memory, by sharing the data common between them. But it can be useful to any application which generates many instances of the same data.

The KSM daemon ksmd periodically scans those areas of user memory which have been registered with it, looking for pages of identical content which can be replaced by a single write-protected page (which is automatically copied if a process later wants to update its content). The amount of pages that KSM daemon scans in a single pass and the time between the passes are configured using :ref:`sysfs intraface <ksm_sysfs>`

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\mm\[linux-master][Documentation][admin-guide][mm]ksm.rst, line 19); *backlink***
>
> Unknown interpreted text role "ref".

KSM only merges anonymous (private) pages, never pagecache (file) pages. KSM's merged pages were originally locked into kernel memory, but can now be swapped out just like other user pages (but sharing is broken when they are swapped back in: ksmd must rediscover their identity and merge again).

Controlling KSM with madvise

KSM only operates on those areas of address space which an application has advised to be likely candidates for merging, by using the madvise(2) system call:

```
int madvise(addr, length, MADV_MERGEABLE)
```

The app may call

```
int madvise(addr, length, MADV_UNMERGEABLE)
```

to cancel that advice and restore unshared pages: whereupon KSM unmerges whatever it merged in that range. Note: this unmerging call may suddenly require more memory than is available - possibly failing with EAGAIN, but more probably arousing the Out-Of-Memory killer.

If KSM is not configured into the running kernel, madvise MADV_MERGEABLE and MADV_UNMERGEABLE simply fail with EINVAL. If the running kernel was built with CONFIG_KSM=y, those calls will normally succeed: even if the KSM daemon is not currently running, MADV_MERGEABLE still registers the range for whenever the KSM daemon is started; even if the range cannot contain any pages which KSM could actually merge; even if MADV_UNMERGEABLE is applied to a range which was never MADV_MERGEABLE.

If a region of memory must be split into at least one new MADV_MERGEABLE or MADV_UNMERGEABLE region, the madvise may return ENOMEM if the process will exceed `vm.max_map_count` (see Documentation/admin-guide/sysctl/vm.rst).

Like other madvise calls, they are intended for use on mapped areas of the user address space: they will report ENOMEM if the specified range includes unmapped gaps (though working on the intervening mapped areas), and might fail with EAGAIN if not enough memory for internal structures.

Applications should be considerate in their use of MADV_MERGEABLE, restricting its use to areas likely to benefit. KSM's scans may use a lot of processing power: some installations will disable KSM for that reason.

KSM daemon sysfs interface

The KSM daemon is controlled by sysfs files in `/sys/kernel/mm/ksm/`, readable by all but writable only by root:

pages_to_scan

> how many pages to scan before ksmd goes to sleep e.g. `echo 100 > /sys/kernel/mm/ksm/pages_to_scan`.
>
> Default: 100 (chosen for demonstration purposes)

sleep_millisecs

> how many milliseconds ksmd should sleep before next scan e.g. `echo 20 > /sys/kernel/mm/ksm/sleep_millisecs`
>
> Default: 20 (chosen for demonstration purposes)

merge_across_nodes

specifies if pages from different NUMA nodes can be merged. When set to 0, ksm merges only pages which physically reside in the memory area of same NUMA node. That brings lower latency to access of shared pages. Systems with more nodes, at significant NUMA distances, are likely to benefit from the lower latency of setting 0. Smaller systems, which need to minimize memory usage, are likely to benefit from the greater sharing of setting 1 (default). You may wish to compare how your system performs under each setting, before deciding on which to use. `merge_across_nodes` setting can be changed only when there are no ksm shared pages in the system: set run 2 to unmerge pages first, then to 1 after changing `merge_across_nodes`, to remerge according to the new setting.

Default: 1 (merging across nodes as in earlier releases)

run

- set to 0 to stop ksmd from running but keep merged pages,
- set to 1 to run ksmd e.g. `echo 1 > /sys/kernel/mm/ksm/run`,
- set to 2 to stop ksmd and unmerge all pages currently merged, but leave mergeable areas registered for next run.

Default: 0 (must be changed to 1 to activate KSM, except if CONFIG_SYSFS is disabled)

use_zero_pages

specifies whether empty pages (i.e. allocated pages that only contain zeroes) should be treated specially. When set to 1, empty pages are merged with the kernel zero page(s) instead of with each other as it would happen normally. This can improve the performance on architectures with coloured zero pages, depending on the workload. Care should be taken when enabling this setting, as it can potentially degrade the performance of KSM for some workloads, for example if the checksums of pages candidate for merging match the checksum of an empty page. This setting can be changed at any time, it is only effective for pages merged after the change.

Default: 0 (normal KSM behaviour as in earlier releases)

max_page_sharing

Maximum sharing allowed for each KSM page. This enforces a deduplication limit to avoid high latency for virtual memory operations that involve traversal of the virtual mappings that share the KSM page. The minimum value is 2 as a newly created KSM page will have at least two sharers. The higher this value the faster KSM will merge the memory and the higher the deduplication factor will be, but the slower the worst case virtual mappings traversal could be for any given KSM page. Slowing down this traversal means there will be higher latency for certain virtual memory operations happening during swapping, compaction, NUMA balancing and page migration, in turn decreasing responsiveness for the caller of those virtual memory operations. The scheduler latency of other tasks not involved with the VM operations doing the virtual mappings traversal is not affected by this parameter as these traversals are always schedule friendly themselves.

stable_node_chains_prune_millisecs

specifies how frequently KSM checks the metadata of the pages that hit the deduplication limit for stale information. Smaller milllisecs values will free up the KSM metadata with lower latency, but they will make ksmd use more CPU during the scan. It's a noop if not a single KSM page hit the `max_page_sharing` yet.

The effectiveness of KSM and MADV_MERGEABLE is shown in `/sys/kernel/mm/ksm/`:

pages_shared
: how many shared pages are being used

pages_sharing
: how many more sites are sharing them i.e. how much saved

pages_unshared
: how many pages unique but repeatedly checked for merging

pages_volatile
: how many pages changing too fast to be placed in a tree

full_scans
: how many times all mergeable areas have been scanned

stable_node_chains
: the number of KSM pages that hit the `max_page_sharing` limit

stable_node_dups
: number of duplicated KSM pages

A high ratio of `pages_sharing` to `pages_shared` indicates good sharing, but a high ratio of `pages_unshared` to `pages_sharing` indicates wasted effort. `pages_volatile` embraces several different kinds of activity, but a high proportion there would also indicate poor use of madvise MADV_MERGEABLE.

The maximum possible `pages_sharing/pages_shared` ratio is limited by the `max_page_sharing` tunable. To increase the ratio `max_page_sharing` must be increased accordingly.

-- Izik Eidus, Hugh Dickins, 17 Nov 2009