Kernel Samepage Merging

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/

The userspace interface of KSM is described in :ref:`Documentation/admin-guide/mm/ksm.rst <admin_guide_ksm>`

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\vm\[linux-master][Documentation][vm]ksm.rst, line 11); *backlink***
>
> Unknown interpreted text role "ref".

Design

Overview

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\vm\[linux-master][Documentation][vm]ksm.rst, line 19)**
>
> Unknown directive type "kernel-doc".
>
> ```
> .. kernel-doc:: mm/ksm.c
>    :DOC: Overview
> ```

Reverse mapping

KSM maintains reverse mapping information for KSM pages in the stable tree.

If a KSM page is shared between less than `max_page_sharing` VMAs, the node of the stable tree that represents such KSM page points to a list of struct rmap_item and the `page->mapping` of the KSM page points to the stable tree node.

When the sharing passes this threshold, KSM adds a second dimension to the stable tree. The tree node becomes a "chain" that links one or more "dups". Each "dup" keeps reverse mapping information for a KSM page with `page->mapping` pointing to that "dup".

Every "chain" and all "dups" linked into a "chain" enforce the invariant that they represent the same write protected memory content, even if each "dup" will be pointed by a different KSM page copy of that content.

This way the stable tree lookup computational complexity is unaffected if compared to an unlimited list of reverse mappings. It is still enforced that there cannot be KSM page content duplicates in the stable tree itself.

The deduplication limit enforced by `max_page_sharing` is required to avoid the virtual memory rmap lists to grow too large. The rmap walk has O(N) complexity where N is the number of rmap_items (i.e. virtual mappings) that are sharing the page, which is in turn capped by `max_page_sharing`. So this effectively spreads the linear O(N) computational complexity from rmap walk context over different KSM pages. The ksmd walk over the stable_node "chains" is also O(N), but N is the number of stable_node "dups", not the number of rmap_items, so it has not a significant impact on ksmd performance. In practice the best stable_node "dup" candidate will be kept and found at the head of the "dups" list.

High values of `max_page_sharing` result in faster memory merging (because there will be fewer stable_node dups queued into the stable_node chain->hlist to check for pruning) and higher deduplication factor at the expense of slower worst case for rmap walks for any KSM page which can happen during swapping, compaction, NUMA balancing and page migration.

The `stable_node_dups/stable_node_chains` ratio is also affected by the `max_page_sharing` tunable, and an high ratio may indicate fragmentation in the stable_node dups, which could be solved by introducing fragmentation algorithms in ksmd which would refile rmap_items from one stable_node dup to another stable_node dup, in order to free up stable_node "dups" with few rmap_items in them, but that may increase the ksmd CPU usage and possibly slowdown the readonly computations on the KSM pages of the applications.

The whole list of stable_node "dups" linked in the stable_node "chains" is scanned periodically in order to prune stale stable_nodes. The frequency of such scans is defined by `stable_node_chains_prune_millisecs` sysfs tunable.

Reference

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\vm\[linux-master][Documentation][vm]ksm.rst, line 82)**
>
> Unknown directive type "kernel-doc".
>
> ```
> .. kernel-doc:: mm/ksm.c
>    :functions: mm_slot ksm_scan stable_node rmap_item
> ```

-- Izik Eidus, Hugh Dickins, 17 Nov 2009