Idle Page Tracking

Motivation

The idle page tracking feature allows to track which memory pages are being accessed by a workload and which are idle. This information can be useful for estimating the workload's working set size, which, in turn, can be taken into account when configuring the workload parameters, setting memory cgroup limits, or deciding where to place the workload within a compute cluster.

It is enabled by CONFIG_IDLE_PAGE_TRACKING=y.

User API

The idle page tracking API is located at `/sys/kernel/mm/page_idle`. Currently, it consists of the only read-write file, `/sys/kernel/mm/page_idle/bitmap`.

The file implements a bitmap where each bit corresponds to a memory page. The bitmap is represented by an array of 8-byte integers, and the page at PFN #i is mapped to bit #i%64 of array element #i/64, byte order is native. When a bit is set, the corresponding page is idle.

A page is considered idle if it has not been accessed since it was marked idle (for more details on what "accessed" actually means see the :ref:`Implementation Details <impl_details>` section). To mark a page idle one has to set the bit corresponding to the page by writing to the file. A value written to the file is OR-ed with the current bitmap value.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\mm\[linux-master][Documentation][admin-guide][mm]idle_page_tracking.rst, line 32); *backlink***
>
> Unknown interpreted text role "ref".

Only accesses to user memory pages are tracked. These are pages mapped to a process address space, page cache and buffer pages, swap cache pages. For other page types (e.g. SLAB pages) an attempt to mark a page idle is silently ignored, and hence such pages are never reported idle.

For huge pages the idle flag is set only on the head page, so one has to read `/proc/kpageflags` in order to correctly count idle huge pages.

Reading from or writing to `/sys/kernel/mm/page_idle/bitmap` will return -EINVAL if you are not starting the read/write on an 8-byte boundary, or if the size of the read/write is not a multiple of 8 bytes. Writing to this file beyond max PFN will return -ENXIO.

That said, in order to estimate the amount of pages that are not used by a workload one should:

1. Mark all the workload's pages as idle by setting corresponding bits in `/sys/kernel/mm/page_idle/bitmap`. The pages can be found by reading `/proc/pid/pagemap` if the workload is represented by a process, or by filtering out alien pages using `/proc/kpagecgroup` in case the workload is placed in a memory cgroup.
2. Wait until the workload accesses its working set.
3. Read `/sys/kernel/mm/page_idle/bitmap` and count the number of bits set. If one wants to ignore certain types of pages, e.g. mlocked pages since they are not reclaimable, he or she can filter them out using `/proc/kpageflags`.

The page-types tool in the tools/vm directory can be used to assist in this. If the tool is run initially with the appropriate option, it will mark all the queried pages as idle. Subsequent runs of the tool can then show which pages have their idle flag cleared in the interim.

See :ref:`Documentation/admin-guide/mm/pagemap.rst <pagemap>` for more information about `/proc/pid/pagemap`, `/proc/kpageflags`, and `/proc/kpagecgroup`.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\mm\[linux-master][Documentation][admin-guide][mm]idle_page_tracking.rst, line 73); *backlink***
>
> Unknown interpreted text role "ref".

Implementation Details

The kernel internally keeps track of accesses to user memory pages in order to reclaim unreferenced pages first on memory shortage conditions. A page is considered referenced if it has been recently accessed via a process address space, in which case one or more PTEs it is mapped to will have the Accessed bit set, or marked accessed explicitly by the kernel (see mark_page_accessed()). The latter happens when:

- a userspace process reads or writes a page using a system call (e.g. read(2) or write(2))
- a page that is used for storing filesystem buffers is read or written, because a process needs filesystem metadata stored in it (e.g. lists a directory tree)
- a page is accessed by a device driver using get_user_pages()

When a dirty page is written to swap or disk as a result of memory reclaim or exceeding the dirty memory limit, it is not marked referenced.

The idle memory tracking feature adds a new page flag, the Idle flag. This flag is set manually, by writing to `/sys/kernel/mm/page_idle/bitmap` (see the :ref:`User API <user_api>` section), and cleared automatically whenever a page is referenced as defined above.

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\mm\[linux-master][Documentation][admin-guide][mm]idle_page_tracking.rst`, line 101); *backlink***
>
> Unknown interpreted text role "ref".

When a page is marked idle, the Accessed bit must be cleared in all PTEs it is mapped to, otherwise we will not be able to detect accesses to the page coming from a process address space. To avoid interference with the reclaimer, which, as noted above, uses the Accessed bit to promote actively referenced pages, one more page flag is introduced, the Young flag. When the PTE Accessed bit is cleared as a result of setting or updating a page's Idle flag, the Young flag is set on the page. The reclaimer treats the Young flag as an extra PTE Accessed bit and therefore will consider such a page as referenced.

Since the idle memory tracking feature is based on the memory reclaimer logic, it only works with pages that are on an LRU list, other pages are silently ignored. That means it will ignore a user memory page if it is isolated, but since there are usually not many of them, it should not affect the overall result noticeably. In order not to stall scanning of the idle page bitmap, locked pages may be skipped too.