

mmu_notifier: can we use it better?

Claudio Imbrenda

IBM

6 October 2026

Disclaimers

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries. For a complete list of IBM Trademarks, see www.ibm.com/legal/copytrade.shtml:

IBM, the IBM logo, IBM Z, IBM z Systems, IBM z15, WebSphere, DB2 and Tivoli are trademarks of IBM Corporation in the United States and/or other countries. For a list of additional IBM trademarks, please see <https://www.ibm.com/legal/copytrade.shtml>.

The following are trademarks or registered trademarks of other companies: Java and all Java based trademarks and logos are trademarks of Sun Microsystems, Inc., in the United States and other countries or both Microsoft, Windows, Windows NT and the Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries, or both. Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries. UNIX is a registered trademark of The Open Group in the United States and other countries or both. Linux is a trademark of Linus Torvalds in the United States, other countries, or both. Cell Broadband Engine is a trademark of Sony Computer Entertainment Inc. InfiniBand is a trademark of the InfiniBand Trade Association. Other company, product, or service names may be trademarks or service marks of others.

Linux penguin image courtesy of Larry Ewing (lewing@isc.tamu.edu) and The GIMP

Any performance data contained in this document was determined in a controlled environment. Actual results may vary significantly and are dependent on many factors including system hardware configuration and software design and configuration. Some measurements quoted in this document may have been made on development-level systems. There is no guarantee these measurements will be the same on generally-available systems. Users of this document should verify the applicable data for their specific environment. IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

Information is provided “AS IS” without warranty of any kind. All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area. All statements regarding IBM’s future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

- 1 Introduction
- 2 KVM/s390 and unique architectural features
- 3 How the event field can help

When the primary MMU of a process changes (i.e. qemu's), the `mmu_notifier` is notified, to allow the secondary MMUs (i.e. KVM) to update the mappings accordingly.

Almost all changes trigger `invalidate_range_start` / `invalidate_range_end`.

There are also other callbacks: `test_young`, `clear_young`, `clear_flush_young`, `release...` those are not the subject of this talk.

When invalidating, the `mmu_notifier` will pass an event field together with the range to `invalidate`, to indicate the reason of the unmap operation.

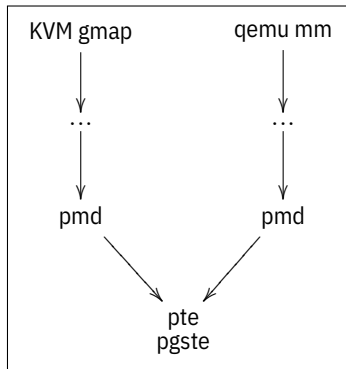
Why it matters to us

- Why knowing the reason (event field) can help
- Why would it be beneficial to have more fine-grained reasons
- Why could it be helpful to have a return value from the notifier to the caller

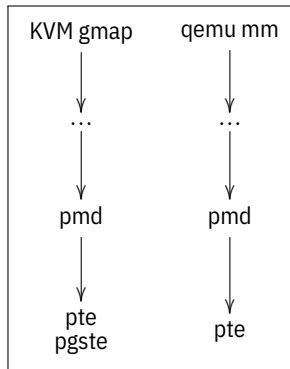
To understand that, I need to explain some unique architectural features of s390 and some implementation details of KVM/s390.

I will try to limit it to the strictly necessary details, feel free to ask me later/offline if you want to know more, I will be happy to infodump you with all the details.

Gmap rewrite



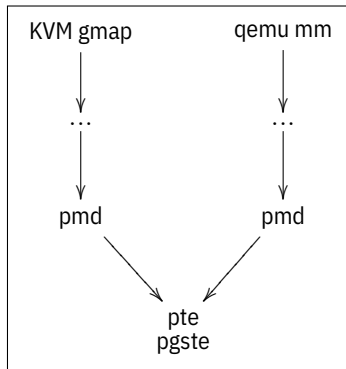
Old implementation



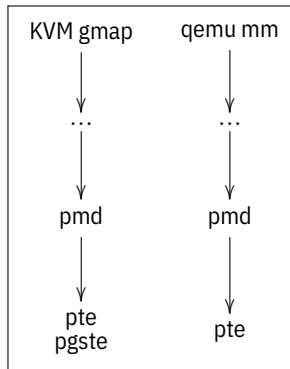
New implementation

- Memory management for KVM/s390 rewritten from scratch.
- Uses `mmu_notifier` now, instead of custom pte invalidation notifier.
- KVM and userspace (qemu) no longer share ptes!
- PGSTEs only handled by KVM

Gmap rewrite



Old implementation

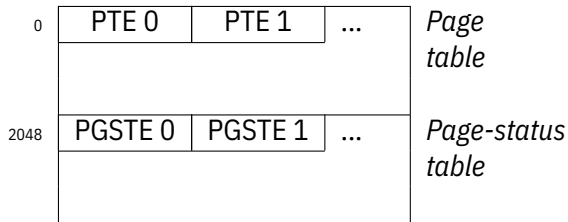


New implementation

- Memory management for KVM/s390 rewritten from scratch.
- Uses mmu_notifier now, instead of custom pte invalidation notifier.
- KVM and userspace (qemu) no longer share ptes!
- PGSTEs only handled by KVM

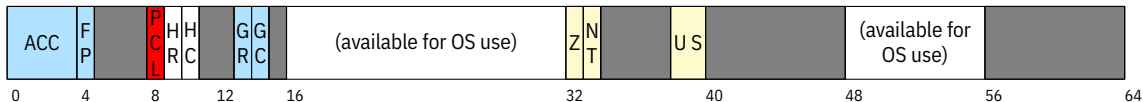
For more details: talk at KVM Forum about the gmap rewrite.

PGSTEs



When used to run a guest, page tables have an extension called Page-status Table, located just after the page table itself.

The Page-status table entry (PGSTE) for each PTE is located at the corresponding index, i.e. at an offset of 2kiB from the PTE itself.



Each PGSTE has a lock bit (PCL), which **must** be taken whenever hardware PGSTE bits are touched, or when the Invalid or Protected bits in the PTE are changed.

- The oldest memory protection mechanism on s390 (straight from S/360)
- 7 bits of information for each *physical* page, stored in hardware
- SSKE/ISKE/RRB and variants to manipulate storage keys
 - Quite slow instructions
- Guest storage key manipulation affects PGSTEs
 - When PTE is invalid, value stored in PGSTE
 - When PTE is valid, 2 PGSTE bits still used (GR, GC)
 - Need to transfer state to/from PGSTE when invalidating/validating PTEs
- Incorrect implementation is a bug and can crash the guest!

- A guest can indicate that a page is not needed anymore (ESSA)
- Usage indication is stored *by hardware* in the PGSTEs.
- When an unused page is attempted to be swapped out, simply throw it away instead
- This is an optimization, throwing away the page attributes (i.e. unused) is not a bug, just a missed opportunity to reduce memory pressure.

The event field

MMU_NOTIFY_UNMAP

either munmap() that unmap the range or a mremap() that move the range

MMU_NOTIFY_CLEAR

clear page table entry (many reasons for this, like madvise() or replacing a page by another one, swapping ...).

MMU_NOTIFY_PROTECTION_VMA

update is due to protection change for the range ie using the vma access permission (vm_page_prot) to update the whole range is enough no need to inspect changes to the CPU page table (mprotect() syscall)

Event types that are not important for KVM:

MMU_NOTIFY_SOFT_DIRTY	MMU_NOTIFY_PROTECTION_PAGE
MMU_NOTIFY_MIGRATE	MMU_NOTIFY_EXCLUSIVE

KVM ignores the event field from the mmu_notifier.

KVM calls the arch callback (`kvm_unmap_gfn_range()`) while holding a spinlock (plain or r/w in write mode)!

If `kvm_unmap_gfn_range()` takes too long, we get RCU stall notifications, since we can't sleep or reschedule.

Even if we don't trigger a stall notification, why waste time?

Knowing event can help

- If we know the reason behind an invalidate (e.g. `munmap()`), we might simply want to throw away all page attributes.
- Conversely, for `MMU_NOTIFY_PROTECTION_VMA` we might want to keep everything as is, and only update the soft-bits.

More fine-grained events

`MMU_NOTIFY_CLEAR` Does too much. Split between `swap`, `advise()`, and other things.

- More fine-grained reasons should exist.
- When discarding memory (e.g. `advise(MADV_DONTNEED)`), it should behave like `MMU_NOTIFY_UNMAP`.
- When swapping, a different event code should be used, to allow the callback to optionally convey that the page is unused and can be thrown away.

Return a value from the invalidation callback

- Need to actually set the “unused” bit in userspace ptes
- Walking the userspace page tables from the invalidation callback is silly
- Would be nice to be able to return a flag to the caller

Discussion time!

Appendix: References and further info

<https://media.ccc.de/v/gpn19-88-a-living-fossil-introduction-to-the-s390x-architecture>

<https://media.ccc.de/v/gpn21-184-history-of-the-mainframe-from-s-360-to-linux>