Grant Improvements

David Vrabel <david.vrabel@citrix.com>

Draft C

Contents

1 Introduction 2

1.1 Revision History 2

1.2 Purpose 2

1.3 Userspace grant maps 2

1.4 Transmitting foreign pages to guests 3

1.5 Blkback and network storage 3

2 Design 3

2.1 Map onto ballooned pages only 3

2.2 Safe grant unmap 4

2.3 Userspace address to page translation 4

2.4 Identifying foreign pages 5

2.5 Userspace grant performance 5

1 Introduction

1.1 Revision History

Version	Date	Changes
Draft A	22 Sept 2014	Initial draft.
Draft B	13 Oct 2014	Require ballooned pages for grant maps.
		Suggest lazy userspace mappings as a future improvement.
Draft C	7 Jan 2015	Set userspace grant map PTEs as special.

1.2 Purpose

Grant mapping in the Linux kernel has a number of problems:

- Grant mapping from userspace is broken for many real world use cases.
- Netback does not handle sending packets to network storage provided by a VM on the same host.
- Using blkback with network-based storage is unsafe.
- Performance is poor, particularly with userspace grants.

1.3 Userspace grant maps

Certain types of system calls using foreign mappings require translating the virtual address to a page using `get_user_pages()` or `get_user_pages_fast()`. These system calls include direct I/O and asynchronous I/O (AIO).

In the native case this translation is done by walking the userspace page tables and looking up the PFN in the L1 entry. PFN to page is then trivial.

For a PV guest this L1 entry contains an MFN and this first needs to be translated into a PFN. For a normal frame this is a simple lookup in the M2P. For foreign pages, the gntdev driver maintains an additional hash of foreign MFNs to local PFNs called the m2p_override.

The m2p_override table has a fundamental design flaw.

A domain may grant a frame multiple times, using a different grant reference each time. The backend maps each grant reference to a separate page. The 1-to-many MFN-to-page mapping cannot be represented in the 1-to-1 m2p_override table and I/O to or from these mappings cannot get the correct page.

1.4 Transmitting foreign pages to guests

Netback when sending pages to the guest uses a grant copy operation to copy the data into the frames granted by the guest. This grant copy requires either a local GFN *or* a grant reference; it is not possible to grant copy to/from a foreign mapping.

In order to support VM to VM traffic, netback stores the grant reference for the sender VM in the socket buffer structure which may then be used by the receiving netback for the grant copy.

Packets with foreign pages from other sources cannot be successfully copied, since netback does not know the grant reference. Once such configuration is a VM providing an iSCSI or other network-based storage that presents a block device in the backend that is then used by another VM on the same host.

1.5 Blkback and network storage

Blkback unmaps the foreign pages in a I/O request when the request is completed. If networked storage is used it is possible for requests to be completed while the skbs referring to those pages are still queued for transmit (e.g., because a retransmission was queued while the responds to the original packet was in flight).

When the network driver attempts to send the packet with the unmapped page it may:

- Fault while trying to access the unmapped page.
- Transmit from a frame that is no longer granted (potentially transmitting sensitive guest or Xen data).

The fault does not occur with userspace storage backends since gntdev replaces the foreign mapping with one to a local scratch page. It uses GNTOP_unmap_and_replace which atomically replaces the foreign mapping with another (source) mapping. However, this cannot be used with batched operations since it clears the source mapping and it does not prevent against transmitting from a non-granted frame.

2 Design

2.1 Map onto ballooned pages only

Grant maps will only be permitted with ballooned pages.

The original p2m entry for these pages will always be INVALID_MFN and thus the original MFN does not need to saved on map and restored on unmap.

Grant map/unmap will no longer need to use or clobber `page->index`. This allows a workaround in netback to clear `page->pfmemalloc` to be removed (`index` and `pfmemalloc` are part of the same union).

2.2 Safe grant unmap

Grant references will only be unmapped when they are no longer in use. i.e., the page reference count is one.

```
int gnttab_unmap_refs_async(struct gnttab_unmap_grant_ref *unmap_ops,
    struct gnttab_unmap_grant_ref *kunmap_ops,
    struct page **pages, unsigned int count,
    void (*done)(void *data), void *data);
```

The `gnttab_unmap_refs_async()` function will unmap the grant references using the supplied unmap operations and call `done(data)`. The grant unmap will only be done once all pages are no longer in use.

It shall run synchronously on the first attempt (this is expected to be the most common case). If any page is in use, it shall queue the unmap request to be tried at a later time.

Only the blkback and gntdev devices need to use asynchronouse unmaps.

2.3 Userspace address to page translation

The m2p_override table shall be removed.

Each VMA (struct vm_area_struct) shall contain an additional pointer to an optional array of pages. This array shall be sized to cover the full extent of the VMA.

The gntdev driver populates this array with the relevant pages for the foreign mappings as they are mapped. It shall also clear them when unmapping. The gntdev driver must ensure it properly splits the page array when the VMA itself is split.

The fast path of get_user_pages_fast() can be made to fail by setting the PTEs as special. Xen 4.0 and later can do this in the grant map hypercall (see XENFEAT_gnttap_map_avail).

The slow path can then get the page from `vma->pages` (if the PTE is special).

`page->private` will no longer need to be set to the MFN (making to available for other uses, see below).

This is similar to the approach used in the classic kernel.

2.4 Identifying foreign pages

A new page flag is introduced: PG_foreign. This will alias PG_pinned so it does not require an additional bit.

If PG_foreign is set then `page->private` contains the grant reference and domid for this foreign page. This information can only be packed into an unsigned long on 64-bit platforms. 32-bit platforms will have to allocate an additional structure to store the domid and gref.

The aliasing of PG_foreign and PG_pinned is safe because:

- Page table pages will never be foreign.
- Foreign pages shall have `p2m[P] & FOREIGN_FRAME_BIT`.

The use of the private field is safe because:

- The page is allocated by the balloon driver and thus it owns the private field.
- The other fields in the union (ptl, slab_cache, and first_page) will not be used because the page is not used in a page table, slab or compound page.

Netback can thus:

1. Test PG_foreign.
2. Verify that the page is foreign via the p2m.
3. Extract the domid and gref from page->private.

The PG_foreign test is not strictly necessary as the p2m lookup is sufficient, but it should be quicker for non-foreign pages.

2.5 Userspace grant performance

Since the m2p_override table will be removed, the gntdev device may easy batch the grant map and unmap hypercalls that update the kernel mappings.

The use of the scratch pages on unmap will be unnecessary and can be removed.

Other improvements that may be considered are:

- Batch the userspace and kernel map and unmap.

- Lazily map grants into userspace on faults. For applications that do not access the foreign frames by the userspace mappings (such as block backends using direct I/O) this would avoid a set of maps and unmaps. This lazy mode would have to be requested by the userspace program (since faulting many pages would be much more expensive than a single batched map).