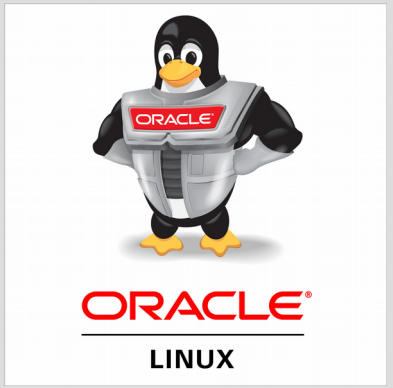
Xenwatch Multithreading

Dongli Zhang
Principal Member of Technical Staff
Oracle Linux
http://donglizhang.org



ORACLE

domU creation failure: problem

 # xl create hvm.cfg

Parsing config from hvm.cfg
libxl: error: libxl_device.c:1080:**device_backend_callback: Domain 2:unable to add device with path /local/domain/0/backend/vbd/2/51712**
libxl: error: libxl_create.c:1290:domcreate_launch_dm: Domain 2:unable to add disk devices
libxl: error: libxl_device.c:1080:device_backend_callback: Domain 2:unable to remove device with path /local/domain/0/backend/vbd/2/51712
libxl: error: libxl_domain.c:1097:devices_destroy_cb: Domain 2:libxl__devices_destroy failed
libxl: error: libxl_domain.c:1000:libxl__destroy_domid: Domain 2:Non-existant domain
libxl: error: libxl_domain.c:959:domain_destroy_callback: Domain 2:Unable to destroy guest
libxl: error: libxl_domain.c:886:domain_destroy_cb: Domain 2:Destruction of domain failed

Reported by: *https://lists.xenproject.org/archives/html/xen-devel/2016-06/msg00195.html*

Reproduced by: *http://donglizhang.org/xenwatch-stall-vif.patch*

ORACLE

domU creation failure: observation

- incomplete prior domU destroy
- stalled xenwatch thread in ‘D’ state
- xenwatch hangs at kthread_stop()

```
dom0# xl list
Name      ID Mem  VCPUs State  Time(s)
Domain-0  0   799        4  r-----     50.2
(null)       2     0        2  --p--d    24.8
```

```
dom0# ps 38
  PID  TTY  STAT   TIME  COMMAND
   38  ?    D      0:00  [xenwatch]
```

```
dom0# cat /proc/38/stack
[<0>] kthread_stop
[<0>] xenvif_disconnect_data
[<0>] set_backend_state
[<0>] frontend_changed
[<0>] xenwatch_thread
[<0>] kthread
[<0>] ret_from_fork
[<0>] 0xffffffffffffffff
```

ORACLE

domU creation failure: cause

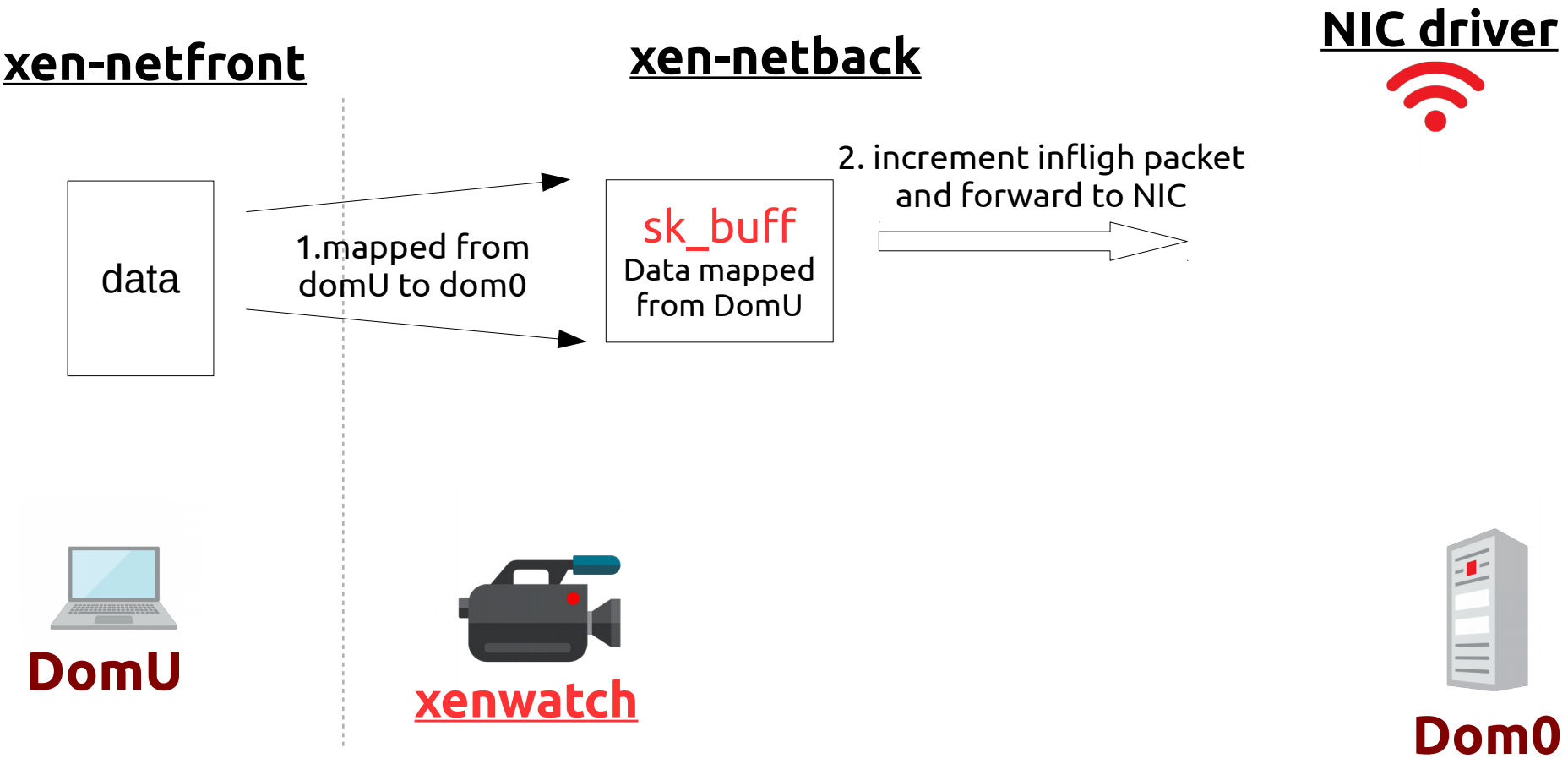
- vif1.0-q0-dealloc thread cannot stop
- remaining inflight packets on netback vif
- vif1.0 statistics: sent > success + fail
- sk_buff on hold by other kernel components!

```
static bool
xenvif_dealloc_kthread_should_stop(struct xenvif_queue *queue)
{
    /* Dealloc thread must remain running until all inflight
     * packets complete. */
    return kthread_should_stop() &&
                !atomic_read(&queue->inflight_packets);
}
```

```
# ethtool -S vif1.0
NIC statistics:
     rx_gso_checksum_fixup: 0
     tx_zerocopy_sent: 72518
     tx_zerocopy_success: 0
     tx_zerocopy_fail: 72517
     tx_frag_overflow: 0
```

ORACLE

xen-netback zerocopy



xen-netback zerocopy

xen-netfront

xen-netback

NIC driver

data

1.mapped from domU to dom0

2. increment infligh packet and forward to NIC

sk_buff
Data mapped from DomU

3. NIC driver does not release the grant mapping correctly!

DomU

xenwatch

pending

4. xenwatch stall due to remaining inflight packet (unmapped grant) when removing xen-netback vif interface

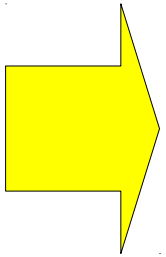
Dom0

domU creation failure: workaround?

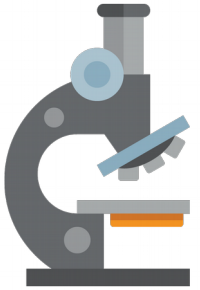
Workaround mentioned at xen-devel:
https://lists.xenproject.org/archives/html/xen-devel/2016-06/msg00195.html

dom0# ifconfig ethX down

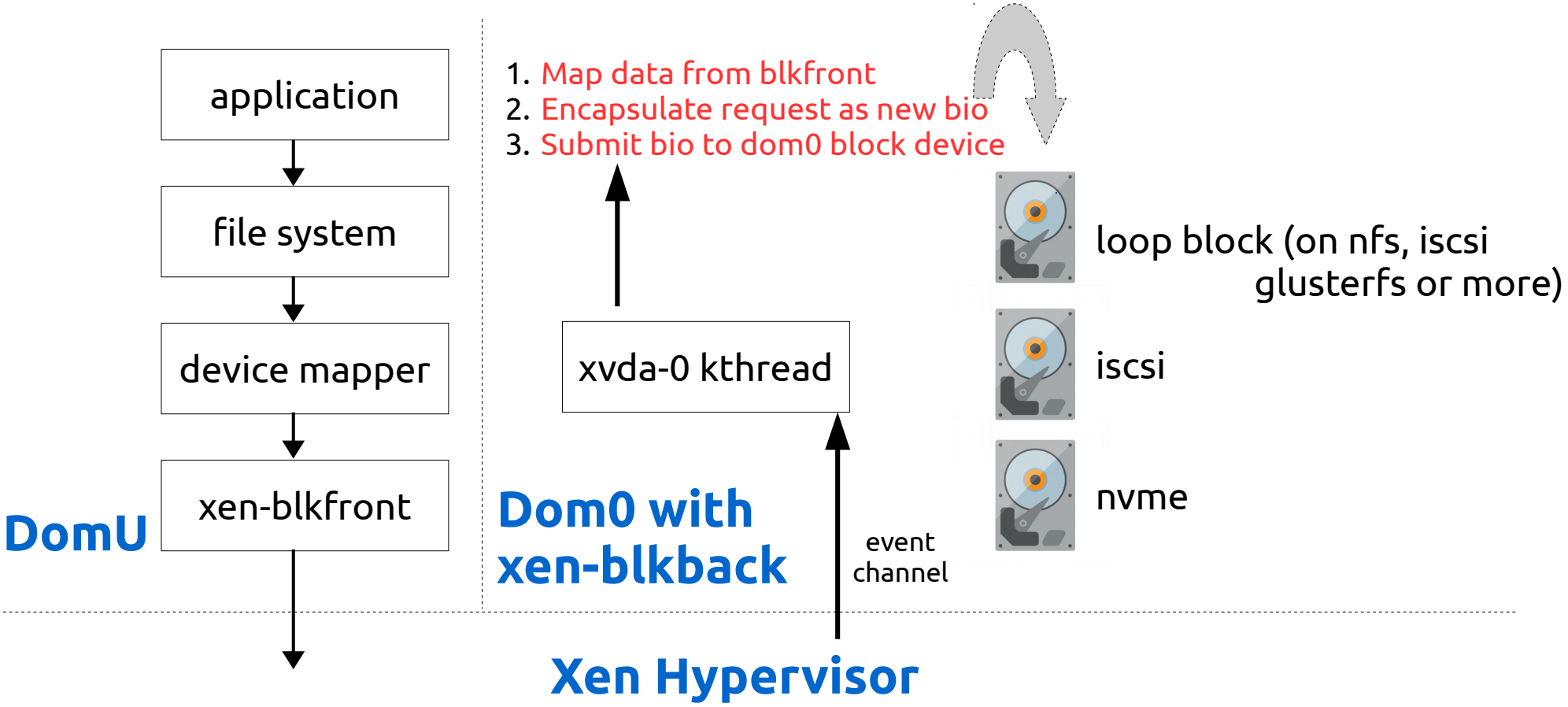
dom0# ifconfig ethX up



Reset DMA buffer and unmap inflight memory page from domU netfront



xenwatch stall extra case prerequisite



xenwatch stall extra case 1

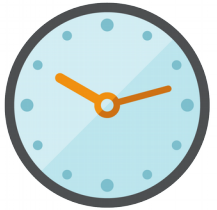
xenwatch waiting for ***3.xvda-0*** hang and waiting for idle block mq tag

[<0>] kthread_stop
[<0>] xen_blkif_disconnect
[<0>] xen_blkbk_remove
[<0>] xenbus_dev_remove
[<0>] __device_release_driver
[<0>] device_release_driver
[<0>] bus_remove_device
[<0>] device_del
[<0>] device_unregister
[<0>] frontend_changed
[<0>] xenbus_otherend_changed
[<0>] frontend_changed
[<0>] xenwatch_thread
[<0>] kthread
[<0>] ret_from_fork

[<0>] bt_get
[<0>] blk_mq_get_tag
[<0>] __blk_mq_alloc_request
[<0>] blk_mq_map_request
[<0>] blk_sq_make_request
[<0>] generic_make_request
[<0>] submit_bio
[<0>] dispatch_rw_block_io
[<0>] __do_block_io_op
[<0>] xen_blkif_schedule
[<0>] kthread
[<0>] ret_from_fork

Lack of free mq tag due to:

- loop device
- nfs
- iscsi
- ocfs2
- more block/fs/storage issue...



ORACLE

xenwatch stall extra case 2

xenwatch

[<0>] gnttab_unmap_refs_sync
[<0>] free_persistent_gnts
[<0>] xen_blkbk_free_caches
[<0>] xen_blkif_disconnect
[<0>] xen_blkbk_remove
[<0>] xenbus_dev_remove
[<0>] device_release_driver
[<0>] bus_remove_device
[<0>] device_unregister
[<0>] frontend_changed
[<0>] xenbus_otherend_changed
[<0>] frontend_changed
[<0>] xenwatch_thread
[<0>] kthread
[<0>] ret_from_fork

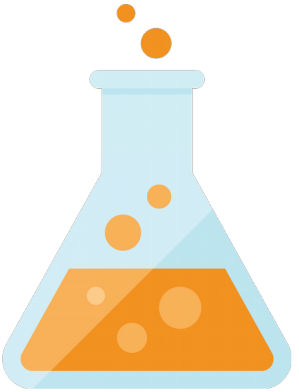
When disconnecting xen-blkback device, wait until all inflight persistent grant pages are reclaimed

```
static void
__gnttab_unmap_refs_async(...)
{
   … …
   for (pc = 0; pc < item->count; pc++) {
      if (page_count(item->pages[pc]) > 1) {
         // delay grant unmap operation
         … …
      }
   }
   … …
}
```

page_count is invalid as the page is erroneously on-hold due to iscsi or storage driver

ORACLE

xenwatch stall symptom



- ‘(null)’ domU in ‘xl list’
- xenwatch stall at xenstore update callback
- DomU creation/destroy failure
- Device hotplug failure
- Incomplete live migration on source dom0
- Reboot dom0 as only option (if workaround is not available)

Xenwatch is victim!

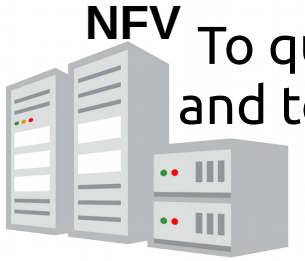
ORACLE

More Impacts

DomU = application

unikernel

More domU running concurrently



To quickly setup and tear down NF

The problem is much more severe...

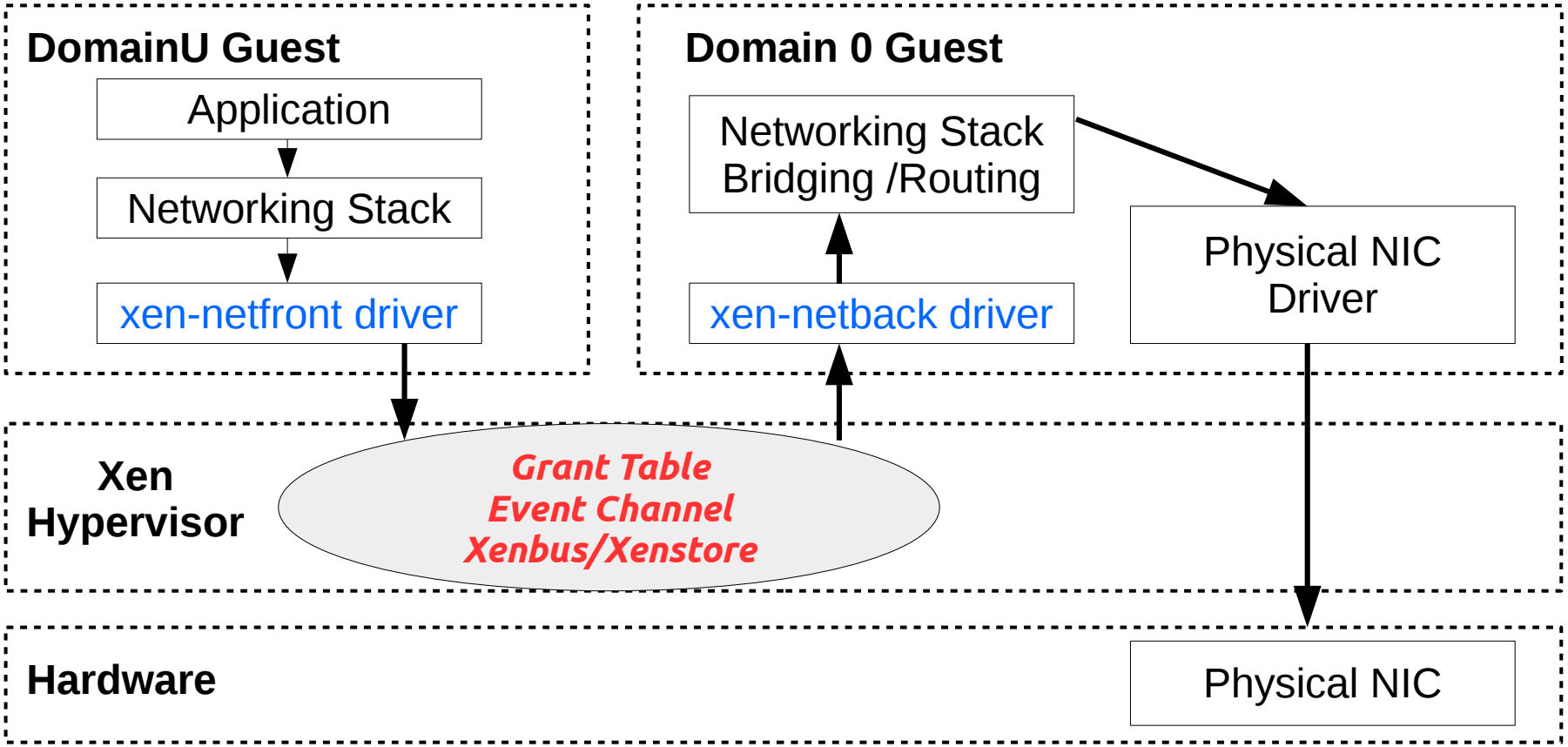
KVM

Let's give up xen!

Xen developers are fired!

ORACLE

xen paravirtual driver framework



Paravirtual vs. PCI

	PCI Driver	Xen Paravirtual Driver
device discovery	pci bus	xenstore
device abstraction	pci_dev / pci_driver	xenbus_device / xenbus_driver
device configuration	pci bar/capability	xenstore
shared memory	N/A or IOMMU	grant table
notification	interrupt	event channel

device init and config

Motherboard
(**hardware** with
many slots)

plug into slots

Physical NIC	Physical Disk
Physical CPU	Physical DIMM

pci bus
- struct pci_dev
- struct pci_driver

dom0# xenstore-ls

```
local = ""
 domain = ""
  0 = ""
   name = "Domain-0"
   device-model = ""
    0 = ""
     state = "running"
   memory = ""
    target = "524288"
    static-max = "524288"
    freemem-slack = "1254331"
   libxl = ""
    disable_udev = "1"
vm = ""
libxl = ""
```

Xenstore
(Dom0 **software**
daemon and database
for all guests)

insert/update entries

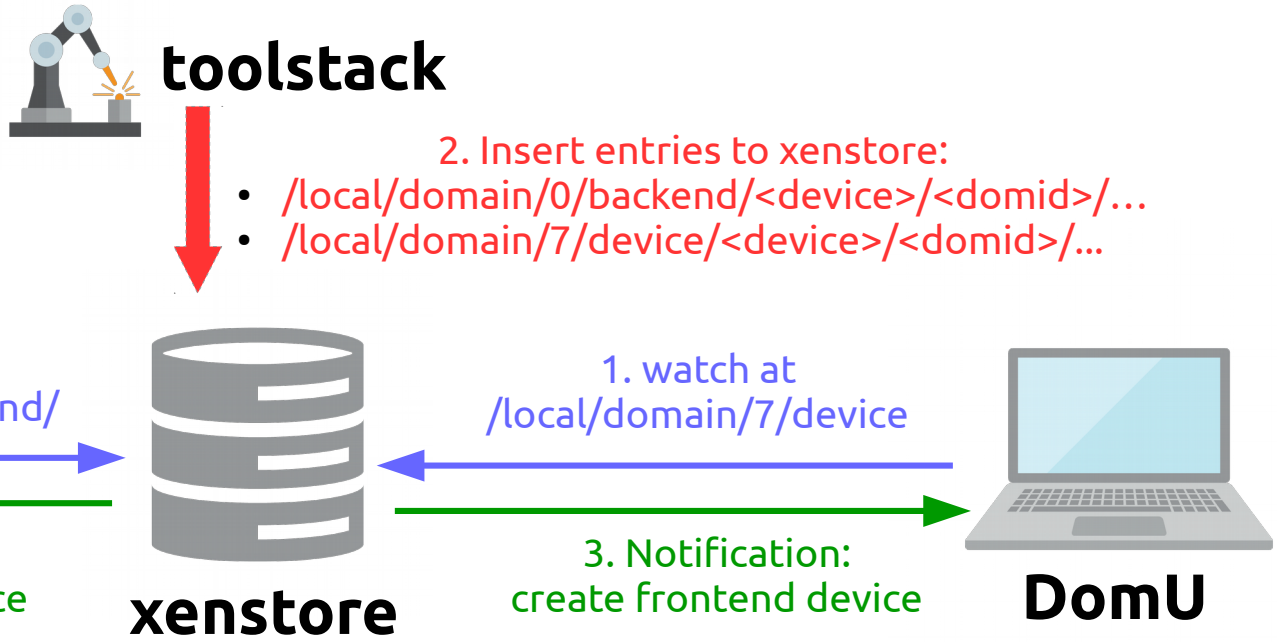
Virtual NIC	Virtual Disk
Virtual CPU	Virtual Memory

xenbus bus
- struct xenbus_device
- struct xenbus_driver

ORACLE

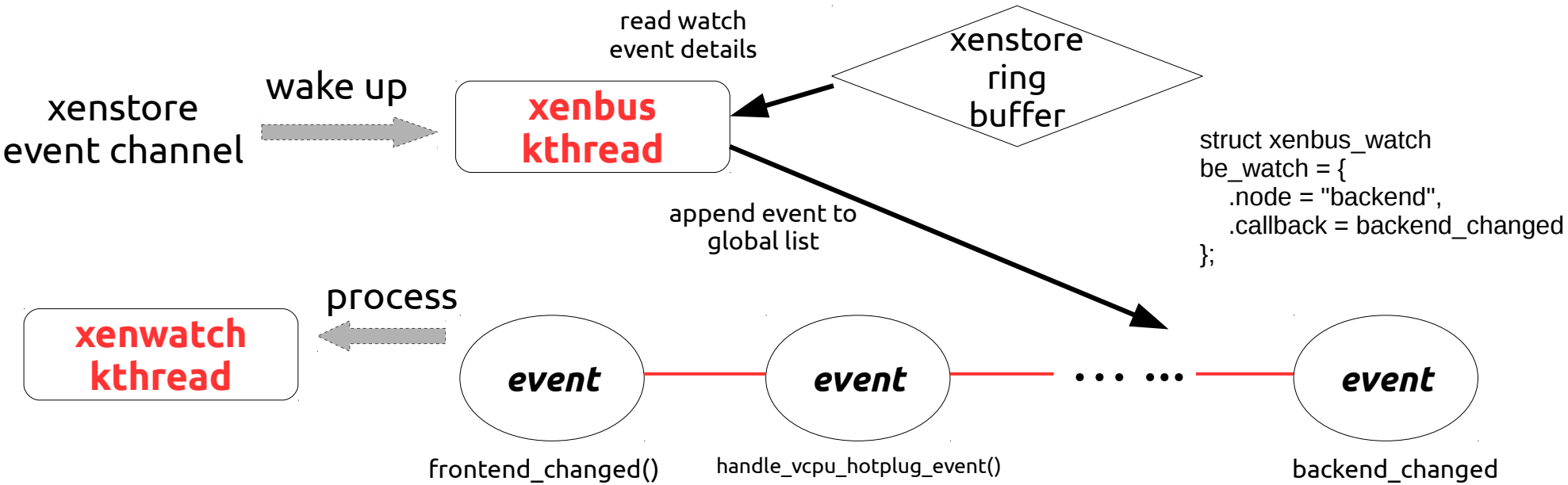
xenstore and xenwatch

- watch at **xenstore node** with **callback**
- callback triggered when xenstore node is updated
- both dom0/domU kernel and toolstack can watch/update xenstore



xenwatch with single thread

- **xenbus_thread** appends new *watch event* to the list
- **xenwatch_thread** processes *watch event* from the list



Xenwatch Multithreading Solution

To create a per-domU xenwatch kernel thread on dom0 for each domid

solution: challenges

- When to create/destroy per-domU xenwatch thread?
- How to calculate the domid given xenstore path?
- Split global locks into per-thread locks

xenwatch event path	watched node
/local/domain/1/device/vif/0/state	/local/domain/1/device/vif/0/state
backend/vif/1/0/hotplug-status	backend/vif/1/0/hotplug-status
backend/vif/1/0/state	backend
backend	backend

solution: domU create/destroy 1/2

xl create vm.cfg

dom0# xenstore-watch /

/
/local/domain/7
/local/domain
/vm/612c6d38-fd87-4bb3-a3f5-53c546e83674
/vm
/libxl/7
… …
@introduceDomain
/libxl/7/dm-version
/libxl/7/device/vbd/51712
/libxl/7/device/vbd
/libxl/7/device
/libxl/7/device/vbd/51712/frontend
/libxl/7/device/vbd/51712/backend
/local/domain/7/device/vbd/51712
… …

xl destroy 7

dom0# xenstore-watch /

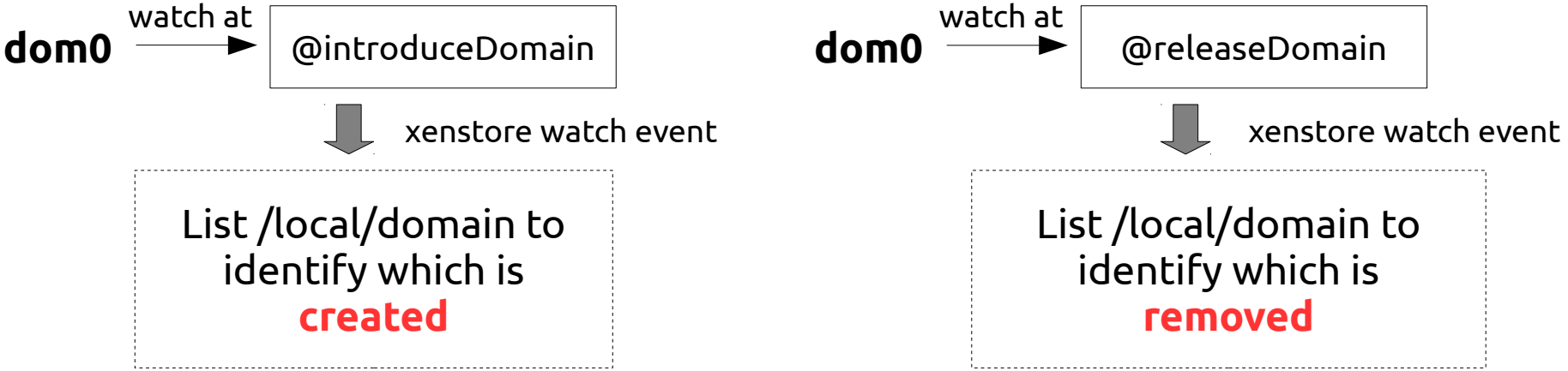
/
/local/domain/0/device-model/7
/local/domain/7/device/vbd/51712
… …
/local/domain/0/backend/vif/7/0/frontend-id
/local/domain/0/backend/vif/7/0/online
/local/domain/0/backend/vif/7/0/state
/local/domain/0/backend/vif/7/0/script
/local/domain/0/backend/vif/7/0/mac
… …
/local/domain/0/backend/vkbd
/vm/612c6d38-fd87-4bb3-a3f5-53c546e83674
/local/domain/7
/libxl/7
@releaseDomain

ORACLE

solution: domU create/destroy 2/2

- creation: watch at “@introduceDomain”
- destroy: watch at “@releaseDomain”
- list “/local/domain” via XS_DIRECTORY

Suggested by Juergen Gross



solution: domid calculation

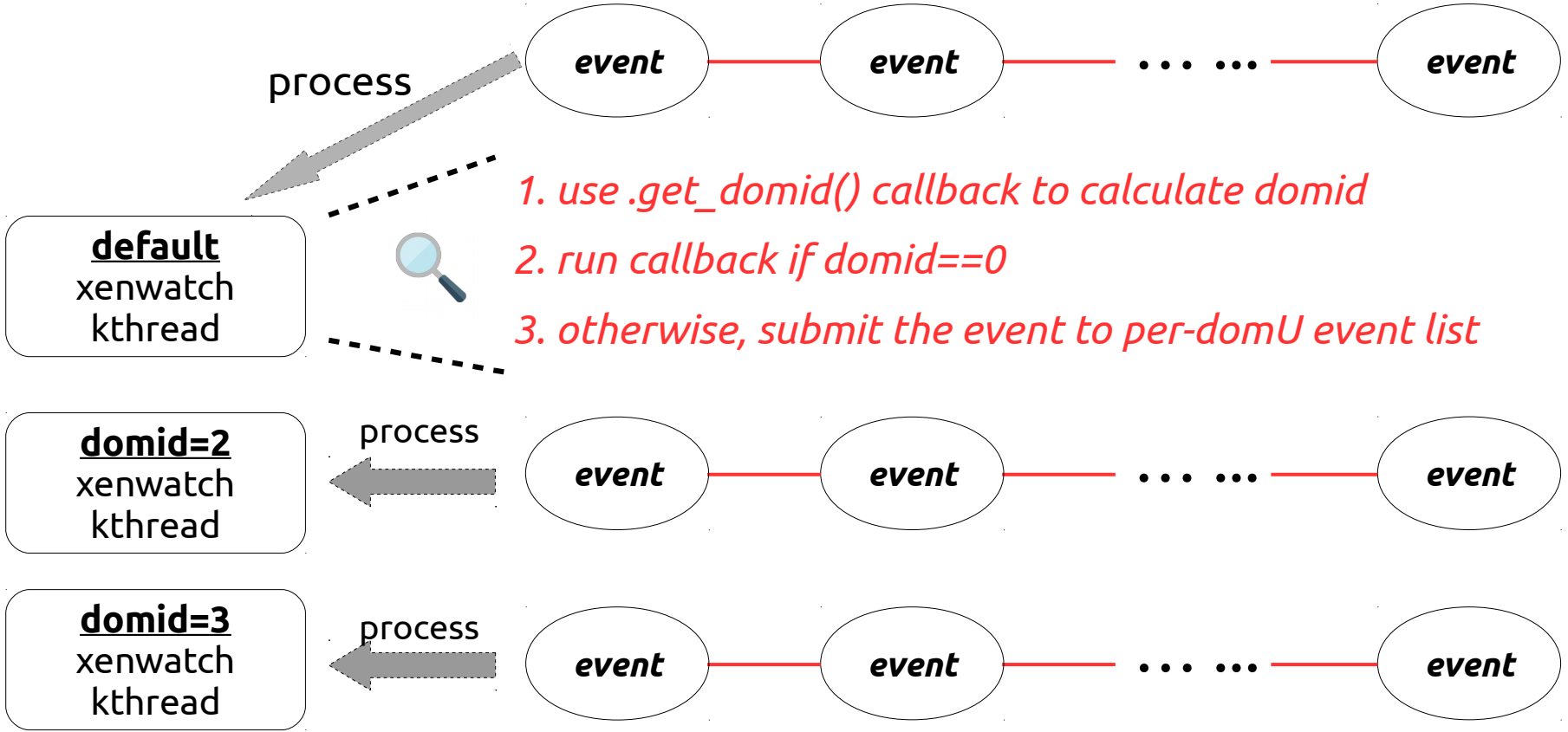
- Xenwatch subscriber should know the pattern of node path
- New callback for ‘struct xenbus_watch’: **get_domid()**
- Xenwatch subscriber should implement the callback

```
struct xenbus_watch
{
   struct list_head list;

   const char *node;

   void (*callback)(struct xenbus_watch *,
                    const char *path, const char *token);

   domid_t (*get_domid)(struct xenbus_watch *watch,
                        const char *path, const char *token);
};
```

be_watch callback

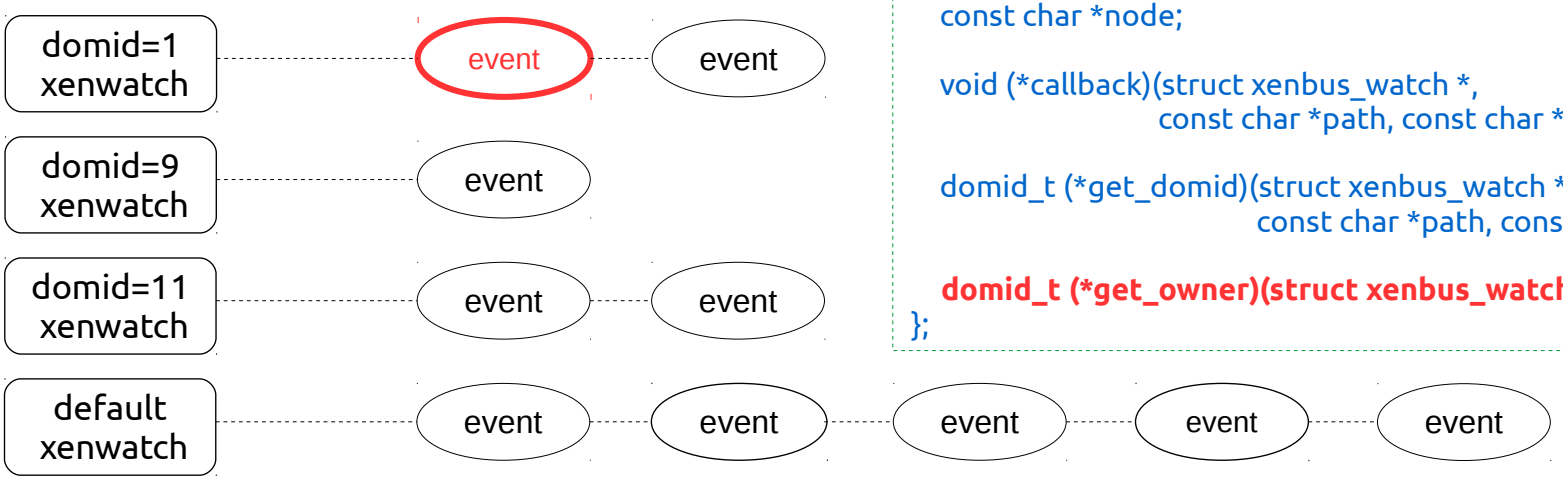
```
/* path: backend/<pvdev>/<domid>/... */
static domid_t be_get_domid(struct xenbus_watch *watch,
                            const char *path,
                            const char *token)
{
   const char *p = path;

   if (char_count(path, '/') < 2)
      return 0;

      p = strchr(p, '/') + 1;
      p = strchr(p, '/') + 1;

      return path_to_domid(p);
}
```

Xenwatch Multithreading Framework



xenbus_watch unregistration optimization

- By default, traverse **ALL lists** to remove pending xenwatch events
- .get_owner() is implemented if xenwatch is for a specific domU
- Only traverse **a single list** for per-domU xenwatch




```
struct xenbus_watch
{
   struct list_head list;

   const char *node;

   void (*callback)(struct xenbus_watch *,
                    const char *path, const char *token);

   domid_t (*get_domid)(struct xenbus_watch *watch,
                        const char *path, const char *token);

   domid_t (*get_owner)(struct xenbus_watch *watch);
};
```

Switch to xenwatch multithreading

Step 1: implement .get_domid()
Step 2: implement .get_owner() for per-domU xenbus_watch

```
// e.g., /local/domain/1/device/vbd/51712/state

static int watch_otherend(struct xenbus_device *dev)
{
   struct xen_bus_type *bus =
        container_of(dev->dev.bus, struct xen_bus_type, bus);

+   dev->otherend_watch.get_domid = otherend_get_domid;
+   dev->otherend_watch.get_owner = otherend_get_owner;
+
    return xenbus_watch_pathfmt(dev, &dev->otherend_watch,
                                bus->otherend_changed,
                                  "%s/%s", dev->otherend, "state");
```

```
+static domid_t otherend_get_domid(struct xenbus_watch *watch,
+                                  const char *path,
+                                  const char *token)
+{
+   struct xenbus_device *xendev =
+   container_of(watch, struct xenbus_device, otherend_watch);
+
+   return xendev->otherend_id;
+}
+
+
+static domid_t otherend_get_owner(struct xenbus_watch *watch)
+{
+   struct xenbus_device *xendev =
+         container_of(watch, struct xenbus_device, otherend_watch);
+
+   return xendev->otherend_id;
+}
```

Test Setup

- Patch for implementation:
  - http://donglizhang.org/xenwatch-multithreading.patch
- Patch to reproduce:
  - http://donglizhang.org/xenwatch-stall-vif.patch
- Intercept sk_buff (with fragments) sent out from vifX.Y
- Control when intercepted sk_buff is reclaimed

ORACLE

Test Result

1)sk_buff from vifX.Y is intercepted by xenwatch-stall-vif.patch
2)[xen-mtwatch-2] is stalled during VM shutdown
3)[xen-mtwatch-2] goes back to normal once sk_buff is released

```
dom0# xl list
Name      ID Mem  VCPUs State  Time(s)
Domain-0  0   799        4  r-----     50.2
(null)     2    0        2  --p--d     29.9
```

```
dom0# ps -x | egrep "mtwatch|xen-xenwatch"
  PID   TTY  STAT   TIME  COMMAND
   39    ?     S      0:00   [xenwatch]
 2196    ?     D      0:00   [xen-mtwatch-2]
```

```
dom0# cat /proc/2196/stack
[<0>] kthread_stop
[<0>] xenvif_disconnect_data
[<0>] set_backend_state
[<0>] frontend_changed
[<0>] xenwatch_thread
[<0>] kthread
[<0>] ret_from_fork
[<0>] 0xffffffffffffffff
```

Current Status

- Total LOC: ~600
- Feature can be enabled only on dom0
- Xenwatch Multithreading is enabled only when:
  - **xen_mtwatch** kernel param
  - xen_initial_domain()
- Feedback for [Patch RFC] from xen-devel

```
Documentation/admin-guide/kernel-parameters.txt |   3 +
drivers/xen/xenbus/xenbus_probe.c               |  39 +-
drivers/xen/xenbus/xenbus_probe_backend.c       |  56 +++
drivers/xen/xenbus/xenbus_xs.c                  | 451 ++++++++++++++++++++++++-
include/xen/xenbus.h                            |  59 ++++
5 files changed, 604 insertions(+), 4 deletions(-)
```

Future work

- Extend **XS_DIRECTORY** to **XS_DIRECTORY_PART**
  - To list 1000+ domU from xenstore
  - Port d4016288ab from Xen to Linux
- Watch at parent node only (excluding descendants)
  - Only parent node's update is notified
  - Watch at "/local/domain" for thread create/destroy

```
Author: Juergen Gross <jgross@suse.com>
Date:   Mon Dec 5 08:48:47 2016 +0100

    xenstore: support XS_DIRECTORY_PART in libxenstore
```

ORACLE

Take-Home Message

- There is **limitation** in single-threaded xenwatch
- It is imperative to address such limitation
- **Xenwatch Multithreading** can solve the problem
- Only OS kernel is modified with ~600 LOC
- Easy to apply to existing xenbus_watch

Question?

ORACLE