RDMA RESET SUPPORT

Liran Liss
Mellanox Technologies
[ April 4th, 2016 ]
Several events may require re-initializing/suspending the operation of a device

- PCI errors
- Device errors
- Unresponsive device
- Device isolation
- Hot unplug (e.g., VM migration)
- Driver restart

In such cases, device resets are required to

- Idle the device
- Bring the device into a known state

Example

- tx_timeout() handler when an Ethernet interface doesn’t complete transmissions
RESET DESIRABLES

- Avoid dependencies on device consumers
- Complete in a timely manner
  - Often, a timeout is what brought us here in the first place...
- Minimize affects to consumers
  - Recover automatically
  - “Invisible” to system operation
- Disable device as a last resort
  - E.g., if normal operation cannot be resumed
  - Ensure that device is idle

Stop! I need to reset

Stop! I need to reset

Just let me finish sending this packet first...
RDMA CHALLENGES

- **Device is stateful**
  - Resources, connection state, in-flight WRs
  - During a reset, this state may be lost

- **Applications and ULPs manipulate HW resources directly**
  - Maximum efficiency, minimum abstraction
  - Resets are observable

- **User-space holds device references**
  - Direct via uverbs, Indirect via ucma
  - Cannot be trusted to release them in a timely manner
RDMA CHALLENGES (CONT.)

- Multi-layer dependency
  - For example:
    1. iSER → CMA → CM → MAD → QP → ib_dev
    2. iSER → QP → ib_dev

  - Which layer do you tear down first?
    1. iSER depends on MADs, so iSER should go down first
    2. How can the MAD layer complete operations if the device is not working?
Kernel Reset Support

- **Reset = abortive shutdown + reinit**
  - Leverage normal dependency order of remove/add ()
  - Adding another asynchronous state is complex

- **Abortive shutdown**
  - Place device in “error mode”
  - Raise IB_EVENTDEVICEFATAL event
  - Unregister device
    - Triggers remove() sequence

- **Device “error mode”**
  - Complete in error all in-flight + new WRs
    - Alternatively, return immediate error for new Post_Send/Recv()’s
  - Successfully “complete” all Verbs that close resources
    - Otherwise, ULPs will hang or risk memory corruption!!!
  - Return immediate errors for all remaining Verbs
ULP ASSUMPTIONS

- **Upon receiving DEVICE_FATAL event**
  - Assume that underlying device is in “error” mode
  - Service API calls in a *timely* manner
    - Do not condition on successful control or data path device operation
    - Optionally return immediate errors (optimization)
    - Optionally avoid internal reset sequences (optimization)
    - E.g., attempt reopening a QP following a completion in error

- **Upon receiving remove()**
  - Close all directly held device resources
  - Free logical instance…

Service ULPx+1 without relying on *successful* HW operation

No assumptions on ULPx-1

ibdev
HW resources must be dereferenced prior to closing a device
- Kernel ULPs may be trusted to do so

Some (non-)options for user-space applications
- Let the application hold the kernel hostage
- Force the application to release resources! Well, not really…
- Kill the application!

Solution: zombify open device instances
- Zombie: a SW implementation of a device in “error mode”
  - Doesn’t hold any reference to HW
- Zombies persists until the last reference is dropped
- Application may attempt to reopen the same device
**SPAWNED A ZOMBIE**

### uverbs
- Disassociate HW from existing uverbs context
  - Free all resources in IDR trees
  - Call provider `disassociate_ucontext()` entry point
    - Redirect memory mappings, free resources, etc.
  - Return EIO for all system calls

### ucma
- Destroy underlying RDMA IDs
- Mark `ucma_context` as closed
  - Avoid duplicate closing when App releases RDMA ID
  - Return EIO for all other system calls

### No change required in umad/ucm
PROVIDER RESET SUPPORT

- **Kernel driver**
  - Implement “error mode”
  - Implement `disassociate_ucontext()`
    - For example
      - Remove MMIO mappings to device
      - Free related resources
      - Notify user-space driver

- **User-space driver**
  - Implement “error mode”
UPSTREAM STATUS

- **Linux 4.3**
  - Reset flow framework
    - `ib_uverbs`, `ib_ucma`
  - ConnectX-3 complete kernel driver support

- **Linux 4.4**
  - ConnectX-4 PCI reset support

- **Ongoing work**
  - ConnectX-4 complete kernel driver support
FUTURE WORK

- **ib_uverbs**
  - Graceful abort
    - Allow grace period for apps to close their references

- **librdmacm**
  - Respond to RDMA_CM_EVENTDEVICE_REMOVAL
    - Refresh device list

- **Maintain ULP context and SW state during reset**
  - Introduce new IB client ops:
    - `stop()` – release all references to HW resources
    - `start()` – re-create HW resources
    - Fallback to remove()/add() if not implemented

- **Persistent names**
  - Kernel and udev support for renaming RDMA devices based on Node GUIDs
Thank You