Virtual RDMA devices

Parav Pandit
Emulex Corporation
Overview

• Virtual devices
  – SR-IOV devices
  – Mapped physical devices to guest VM (Multi Channel)
  – Para virtualized devices
  – Software based virtual devices

  – Virtual device mapped to physical device
Overview of RoCE/NIC Interfaces (Dedicated QP1)

Connection Manager

GSI QP1

RoCE-NIC Device

Nic_Iface

iSER/NFS-RDMA/Ceph/P9/SMB ULPs

QP

QP

QP

GSI QP1

RoCE-NIC Device

Nic_Iface

OS/Hypervisor Driver IB Stack

HCA/NIC Adapter (PF)
Multiple RoCE/NIC partitions using interfaces

• NIC driver can create one or more adapter specific Nic_Iface each having unique attributes such as MAC address and/or vlan.

• Each such Nic_Iface translates to adapter interface object at OS/Hypervisor level.

• One or more such interfaces can be created per PF.

• Each Nic_Iface can host NIC RSS RQs.

• NIC traffic continues to operate based on the existing Nic_Iface object including RSS support.
RoCE QP to Interface binding

- RoCE QPs are attached to one of the \textit{Nic\_Iface} object during QP state transition.
- RoCE QP can be reused without destroying and recreated to bind to different \textit{Nic\_Iface}.
- Supports accelerated (extensions) QP state transitions to make QP usable for data traffic per \textit{Nic\_Iface}.
- Every interface can host GSI QP1 based on the \textit{Nic\_Iface} adapter interface object.
- QP1 need to scale with each adapter interface object. There is better option that that.
Shared QP1 across multiple NIC interfaces

- TCP/IP Stack
- RDMA Connection Manager
- iSER/NFS-RDMA/Ceph/P9/SMB ULps
- QP
- NIC device
- NIC RSS Qs
- NIC Tx Qs
- RoCE device
- NIC device
- NIC RSS Qs
- NIC Tx Qs
- Nic_Iface
- HCA/NIC PF Adapter
- OS/Hypervisor Driver IB Stack
Shared QP1 across multiple interfaces

- One GSI QP1 per port shared among multiple \texttt{Nic\_Iface}.
- Scales well as number of interfaces grows.
- Filters QP1 packets on \texttt{Nic\_Iface} basis.
- RoCE device driver connection manager performs connection establishment, filtering for the right \texttt{Nic\_Iface} object.
- GSI QP1 accepts packets RoCEv1 and RoCEv2 packets.
- Adapter parses QP1 packets and informs v1/v2 to connection manager stack reducing the need to parse the packets in host software driver.
- Connection manager supports RoCEv1 and RoCEv2 connections on per QP basis.
Linux vrdma overview

Connection Manager

Shared GSI QP1

Primary RoCE-NIC Device

Vlan_Nic_Iface

iSER/NFS-RDMA ULPs

Linux Container

RoCE-NIC Device

QP

RoCE-NIC Device

QP

RoCE-NIC Device

QP

OS/Hypervisor Driver IB Stack

Device cgroup (extended for RDMA devices)

HCA/NIC Adapter (PF)
Virtual RDMA devices

- Multiple virtual RDMA devices are mapped to single physical ibdev RDMA device
- Each virtual RDMA device is mapped to single netdev structure
- Virtual device and physical device bound to same physical netdev during creation time
- Virtual device uses the resource of the physical devices (MR, PD, SRQ, QP)
- Network device movement from network namespaces destroys the virtual RDMA devices
- Network namespace destroys virtual rdma devices
Role of cgroup vs virtual rdma device

• Each virtual device can be provisioned for their own resources (orthogonal to cgroups)
• Device cgroup is extended for resource limitation of user space applications across multiple rdma devices
• Virtual devices provides resource provisioning for kernel consumers and their resources which are not directly bound to user processes, created/destroyed from threads, work_queues, connection management event handlers.
New ibverbs Linux APIs

- Binding creates the virtual rdma device (libibverbs)
  - `ibv_create_virtual_ibdev(struct ib_device *pdev,
    char *netdev_name,
    char *new_ibdev_name);`
  - `ibv_destroy_virtual_ibdev(char *virtual_ibdev_name);`

- Configuration
  - `int ibv_provision_resources(struct ib_device *vdev,
    struct ib_resource_config *resources);`
  - Resources provisioning can be changed dynamically, provided it doesn’t have consume all of them.

- Device flag to indicate virtual or physical flag
  - `Ib_get_device_type(struct ib_device *dev);`

- New APIs for vendor drivers
  - `Ib_register_vdevice(struct ib_device *vdevice);`
  - `Ib_unregister_vdevice(struct ib_device *vdevice);`
Isolation and performance

• Application Isolation
  – via ib_ucontext
• Virtual device object itself isolates itself in kernel space from other devices
• Performance
  – Same as physical device as every resource is mapped to physical device resource
• No need to carve out or over provision resources per VF and/or PF at device level, including MSI-X vectors, GIDs etc.
• Isolation checks for VFs at adapter level can be skipped bring simplicity to deployment and configuration management at adapter level.
Challenges and future extensions

- GSI QP1 extension
  - Single QP1 to multiplex for virtual devices

- GID isolation
  - Proxy GIDs for physical devices, to be omitted during connection establishment.

- Interrupt moderation

- Single virtual device for multiple RDMA devices
- More higher level operations than just pure SEND/WRITE/READs
- Virtual devices can be made usable beyond containers to other hypervisor modes.
Thank You