RDMA Containers Update

Parav Pandit

14th OFA Workshop, April, 2018

Agenda

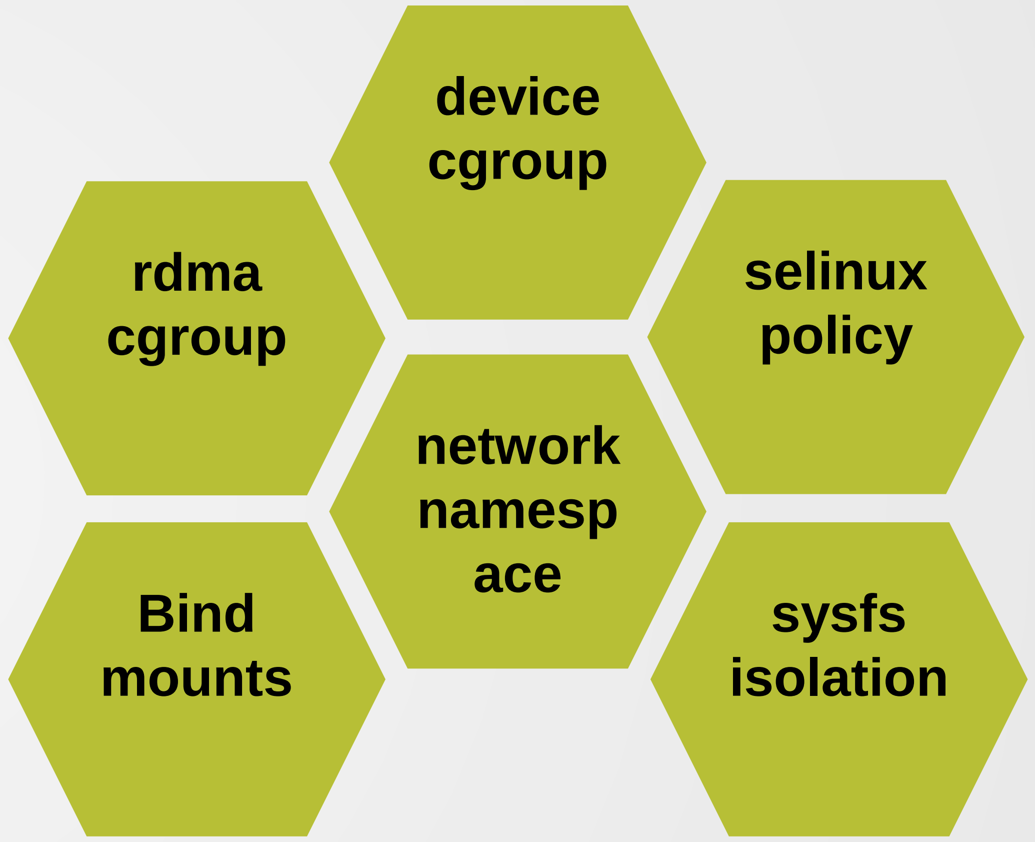
- Quick overview
- Updates

- How to deploy containers

- Challenges and solutions
- Future plans
- Questions

Kernel components for container enablement

- selinux policy enforcement
- rdma cgroup configuration
- device cgroup configuration
- network namespace support
- sysfs isolation



Selinux policy

- Kernel
- Initial version starting from Linux kernel 4.13
- Stable version is 4.15 (IB core)

rdma cgroup update

- Kernel
- Part of kernel from Linux kernel 4.11

- User space
  - Runc spec update
  - https://github.com/opencontainers/runtime-spec

  - containerd/cgroups library
  - https://github.com/containerd/cgroups

Upcoming functionalities

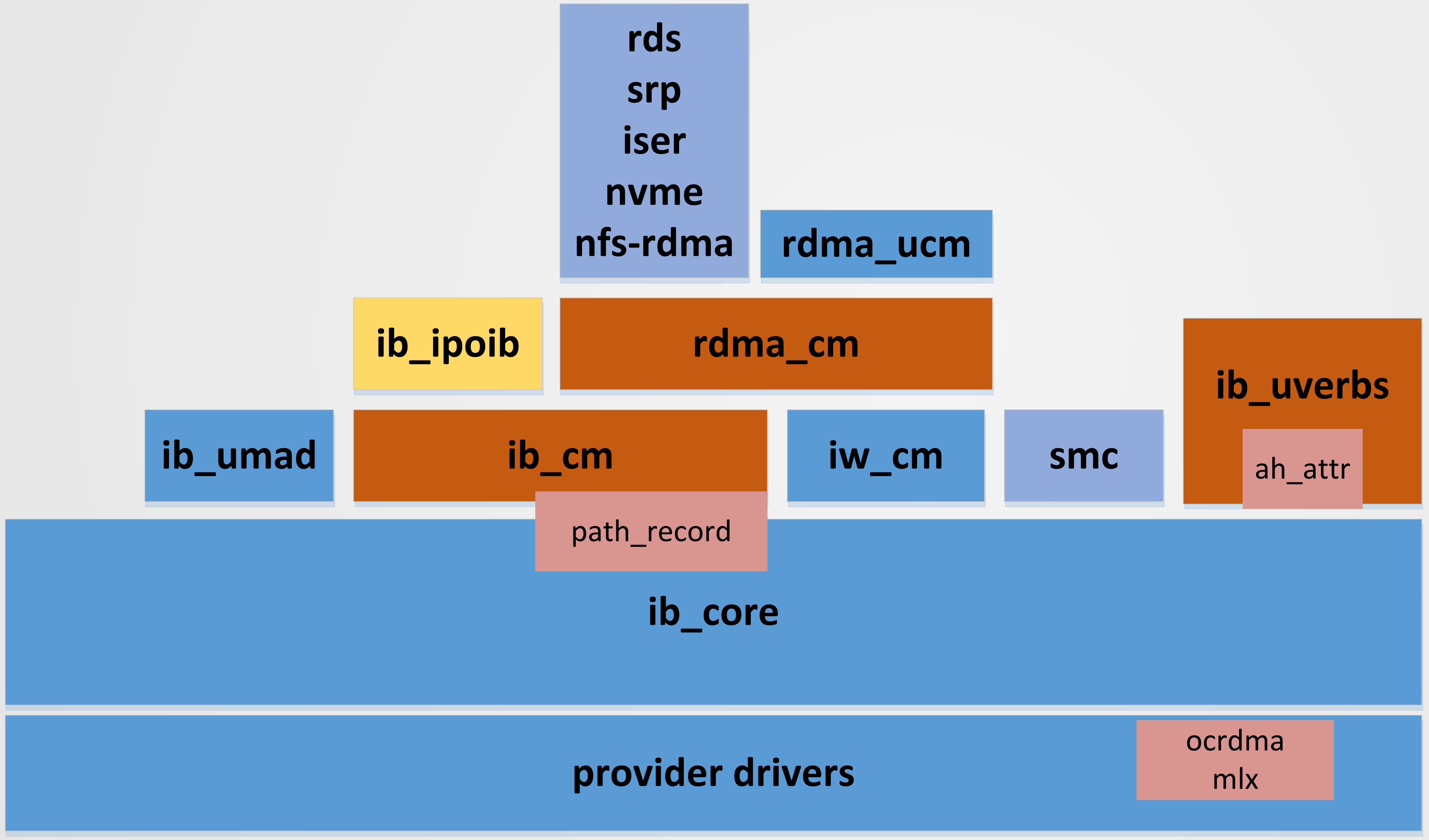
- net namespace support
  - Limiting scope of net namespace involvement
  - GID entry reference counting
  - Net namespace validation
  - rdmacm RoCE extension for net namespace (CM messages)

Net namespace involvement

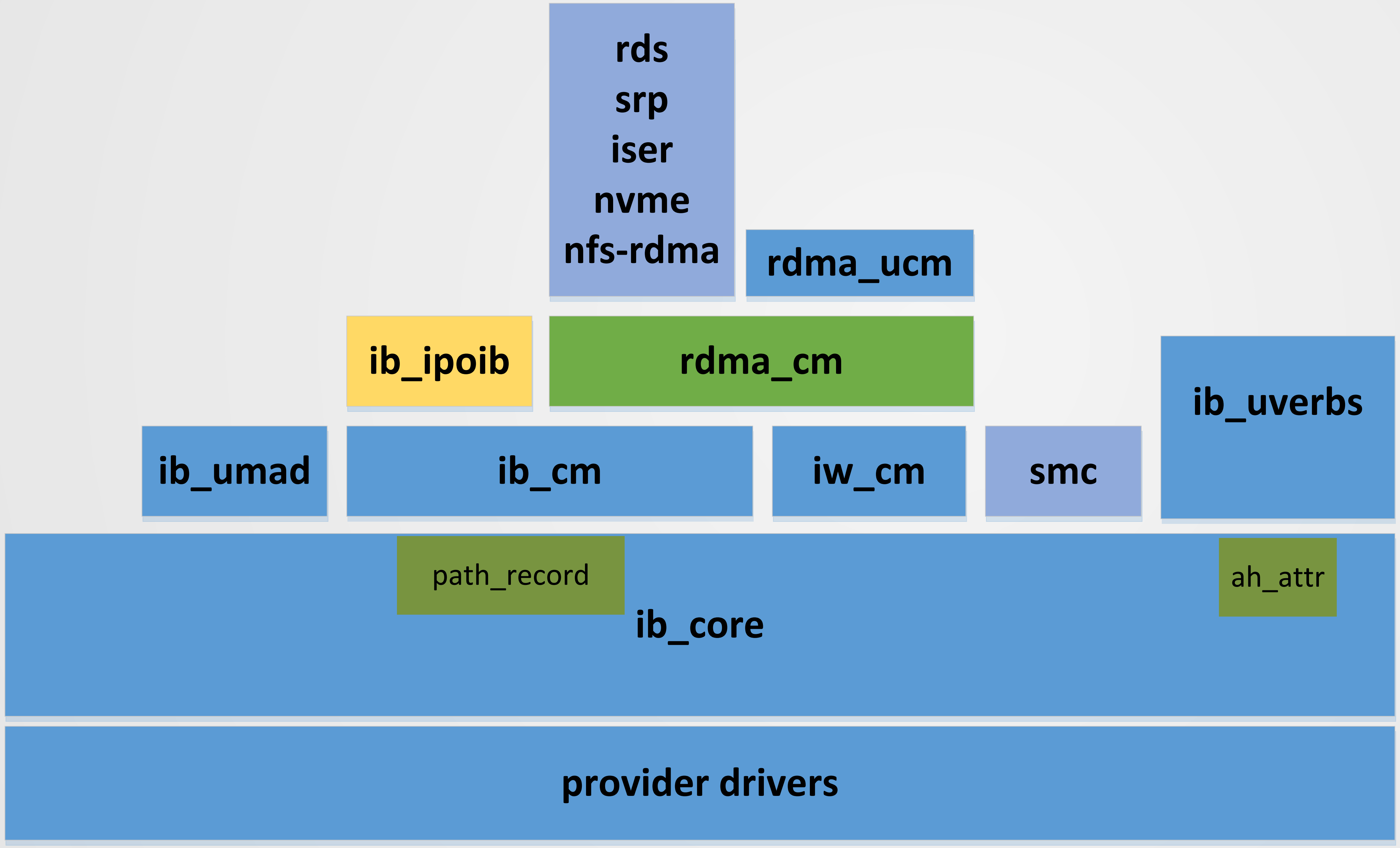
- Limiting scope of net namespace involvement
  - Code consolidation among provider drivers and core
  - Elimination of net+ifndex in RoCE path record
  - Only two modules to get involved – rdma_cm, ib_core

- Why?
  - HCA agnostic code resides in common core module
  - Easy maintainability and reference counting, locks at consolidated routines
  - Avoids possibility of not honoring net namespace

RDMA stack view (net namespace involvement - before)



RDMA stack view (net namespace involvement - after)



GID entry reference counting

- Why?
  - RoCE GID entries are based on network devices, ip addresses
  - netdevices and GID entries migrate among net namespaces
  - Multiple kernel modules (CM, verbs, core, providers) uses the GID entries while GIDs are changing
  - Needs a consistent view among all modules

- GID reference counting
  - Unified APIs for IB transport
  - Referenced GID attributes
  - Garbage collection of GID deletion
  - GID filter/lookup based on optional mac address

GID entry reference counting

- Sample APIs
  - const struct rdma_gid_attr* rdma_get_gid_attr(device, port, index);
  - rdma_put_gid_attr(const struct rdma_gid_attr *);
  - rdma_hold_gid_attr(const struct rdma_gid_attr *);

  - rdma_find_gid(device, port, search_attribute);

- Flow:
  - Single query during work completion (ib_cm)
  - Reuse attribute during processing
  - Single reference during rdma_bind/resolve_addr (rdma_cm)
  - ah_attrr holds reference
    - Used by rdma_cm, ib_cm, ib_core and providers
  - Released during ib_cm_destroy_id(), rdma_cm_destroy_id().

Other changes

- RDMA CM extension
  - IB CM requests are net namespace unaware
  - Extended using sgid_attr for RoCE

- IB core:
  - rcu locked network objects access (net, netdev, route, neigh lookup)
  - addr_resolve() relies on sgid_attr
  - Performs operation under rcu lock to synchronize with change_net_namespace().
  - Isolate sysfs entries, waiting for new GID API

Near future plans

- Per containers rdma statistics/counters
(again triggered through ib_core, provider agnostic)

- Making netlink socket per net_ns for rdmatool

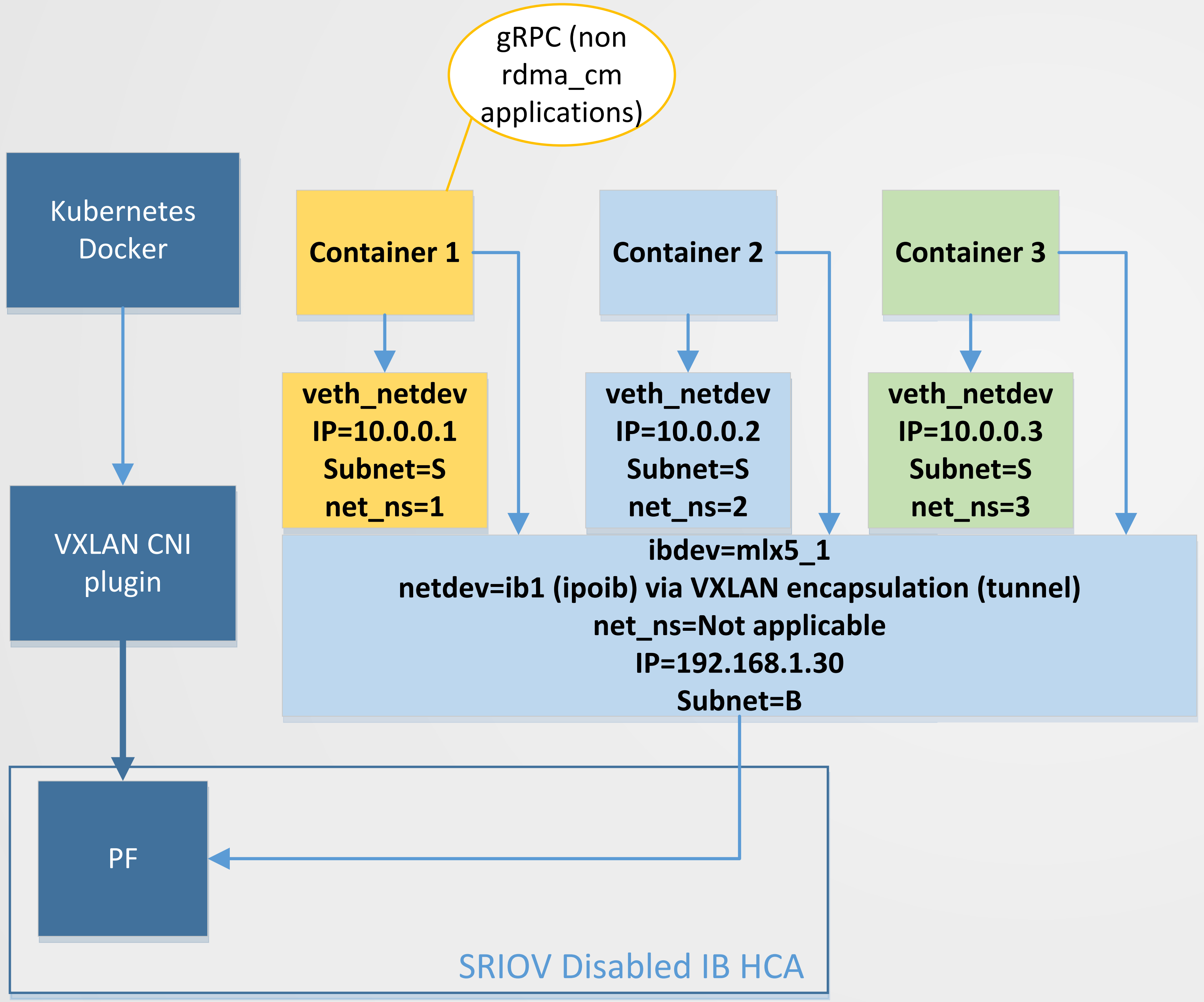
- rdmatool extension for RoCEv1 enable/disable to scale RoCE containers by 2x

- selinux query_pkey needs to honor selinux policy enforcement

How to deploy Containers

- IB Transport (IB, RoCE)
  - Virtual networking in shared device mode
    - Docker swarm
    - Kubernetes
  - Isolated RoCE device per container
    - Docker native
    - Kubernetes using device plugin and cni plugin
    - Custom orchestration tool

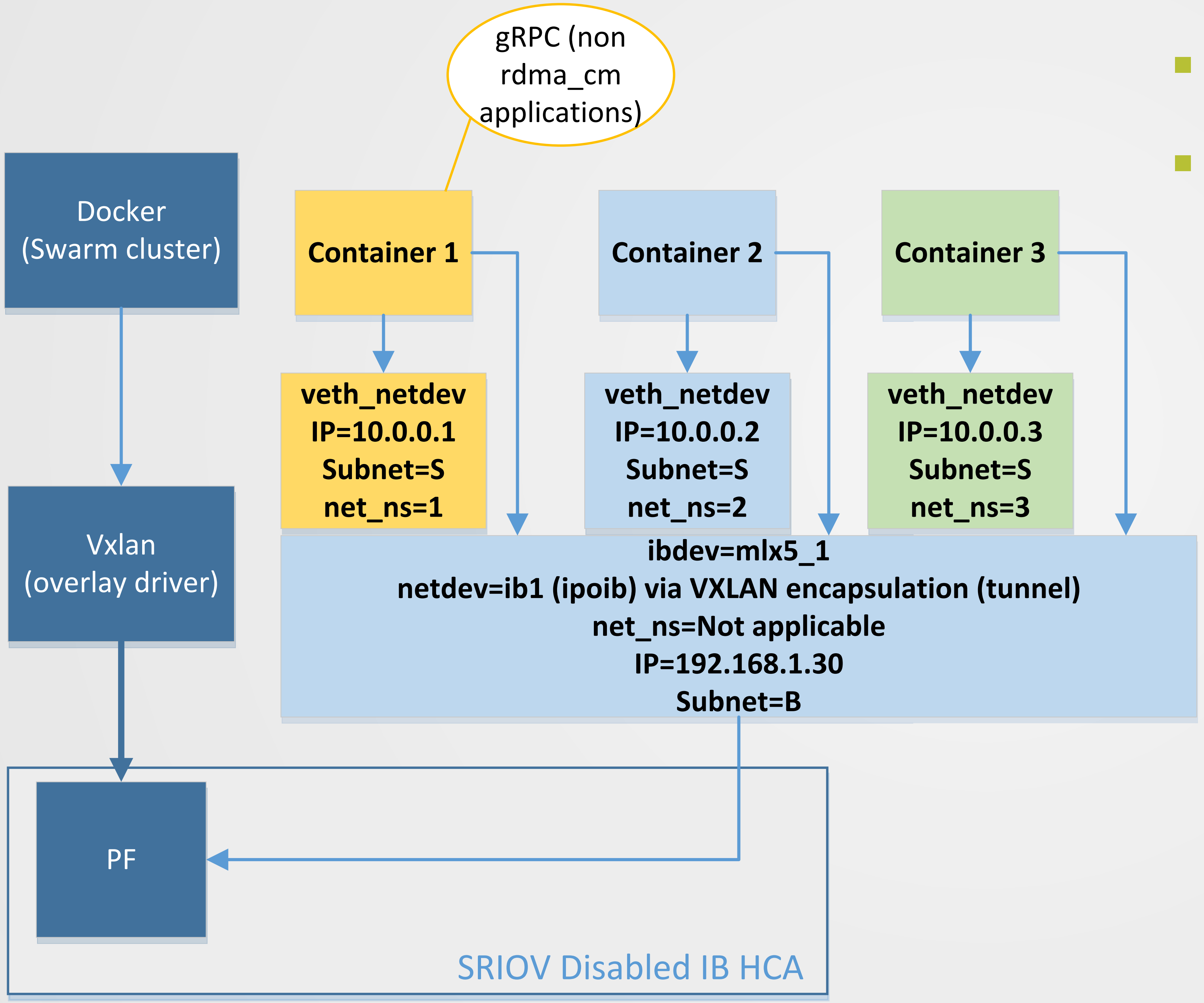
Deploying Container with IB device (K8s)


gRPC (non rdma_cm applications)
Kubernetes Docker
Container 1
Container 2
Container 3
veth_netdev IP=10.0.0.1 Subnet=S net_ns=1
veth_netdev IP=10.0.0.2 Subnet=S net_ns=2
veth_netdev IP=10.0.0.3 Subnet=S net_ns=3
VXLAN CNI plugin
ibdev=mlx5_1
netdev=ib1 (ipoib) via VXLAN encapsulation (tunnel)
net_ns=Not applicable
IP=192.168.1.30
Subnet=B
PF
SRIOV Disabled IB HCA

- Single IB device shared among containers
- Suitable for applications not using rdma_cm
- Easy orchestration:
- --device = /dev/infiniband/uverbsX

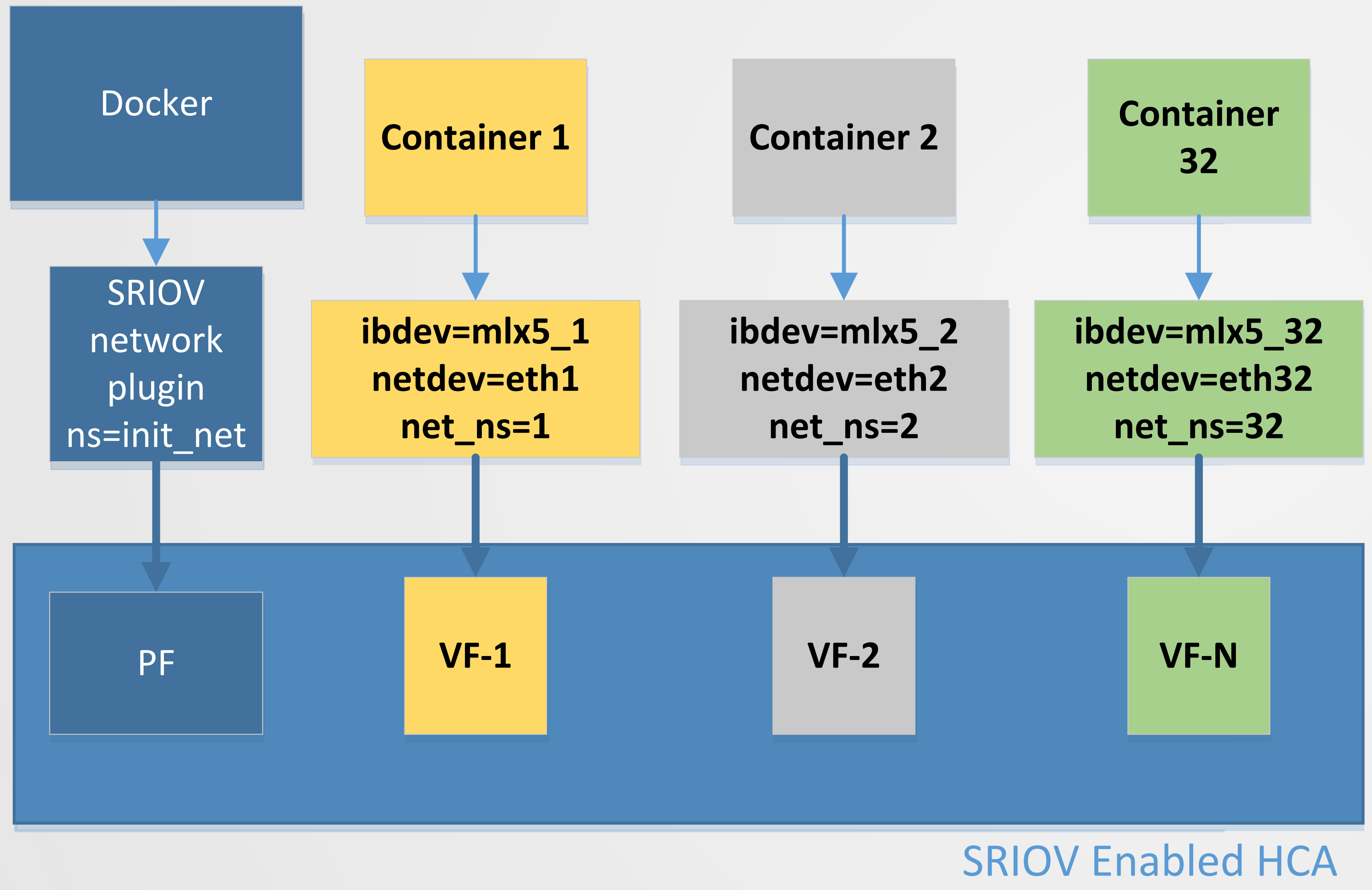
- Optionally:
- needs support of recent rdma_cgroup runc spec
- Needs support of selinux label for pkey

Deploying Container with IB device (Docker swarm)


gRPC (non rdma_cm applications)
Docker (Swarm cluster)
Container 1
Container 2
Container 3
veth_netdev IP=10.0.0.1 Subnet=S net_ns=1
veth_netdev IP=10.0.0.2 Subnet=S net_ns=2
veth_netdev IP=10.0.0.3 Subnet=S net_ns=3
Vxlan (overlay driver)
ibdev=mlx5_1
netdev=ib1 (ipoib) via VXLAN encapsulation (tunnel)
net_ns=Not applicable
IP=192.168.1.30
Subnet=B
PF
SRIOV Disabled IB HCA

- Similar to K8s, but using Docker swarm clustering
- Scalable to large number of active and non active application instances

Deploying Container with IB or RoCE devices using SRIOV (WIP)

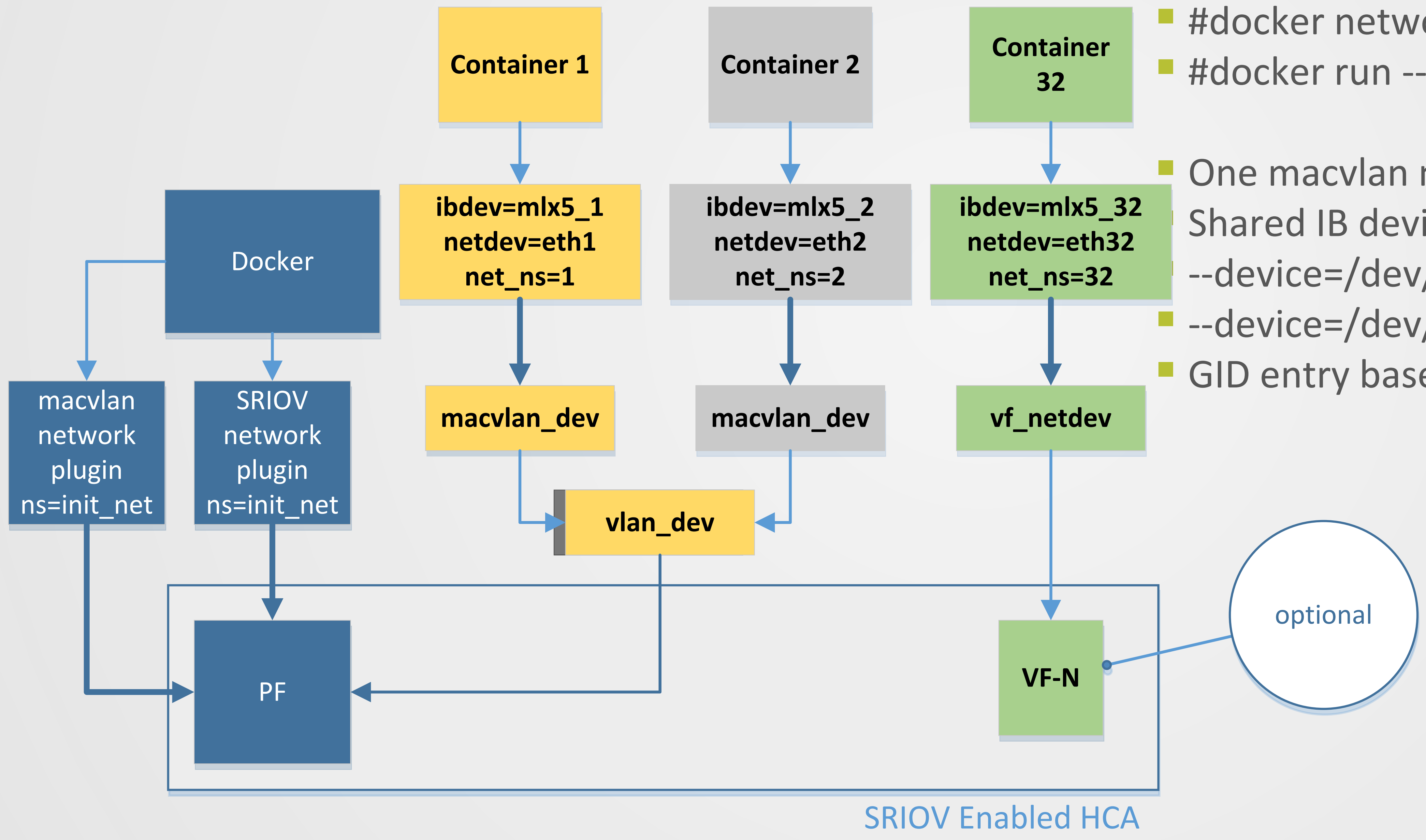


Docker
Container 1
Container 2
Container 32
SRIOV network plugin ns=init_net
ibdev=mlx5_1 netdev=eth1 net_ns=1
ibdev=mlx5_2 netdev=eth2 net_ns=2
ibdev=mlx5_32 netdev=eth32 net_ns=32
PF
VF-1
VF-2
VF-N
SRIOV Enabled HCA

- SRIOV plugin
- One VF per container
- Suitable for RDMA CM and non RDMA CM applications
- Per container device, limited to number of devices per node

- #docker network create -d passthrough [..]
- #docker run –net=sriov_net

Deploying Container with IB or RoCE devices using macvlan driver (WIP)



Container 1
Container 2
Container 32
ibdev=mlx5_1 netdev=eth1 net_ns=1
ibdev=mlx5_2 netdev=eth2 net_ns=2
ibdev=mlx5_32 netdev=eth32 net_ns=32
Docker
macvlan network plugin ns=init_net
SRIOV network plugin ns=init_net
macvlan_dev
macvlan_dev
vf_netdev
vlan_dev
PF
VF-N
optional
SRIOV Enabled HCA

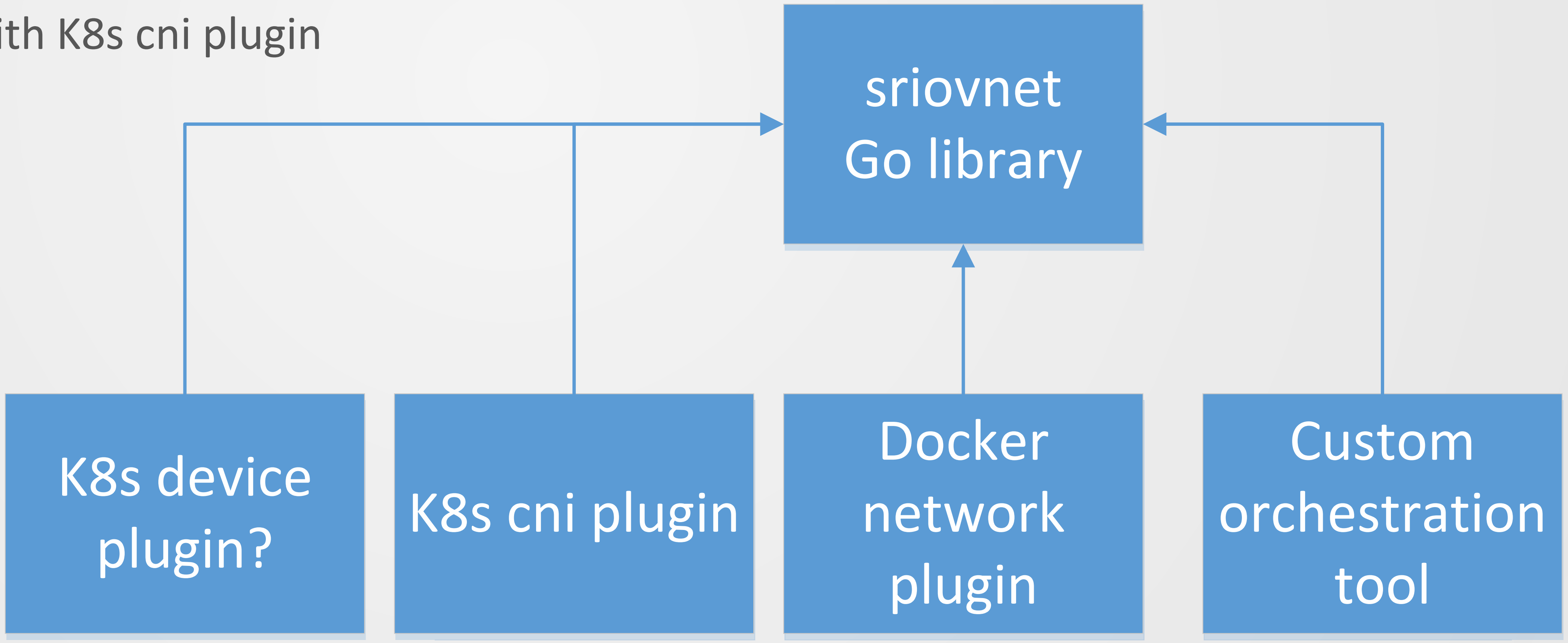
- Docker native macvlan driver
- #docker network create -d macvlan [..]
- #docker run --net=macvlan_1

- One macvlan netdevice per container
- Shared IB device
- --device=/dev/infininband/uverbsX
- --device=/dev/infiniband/rdma_cm
- GID entry based isolation

Custom orchestration

- Possibly – Singularity?
  - MPI application as container

- Network specific orchestration tool
- https://github.com/Mellanox/sriovnet
- https://hub.docker.com/r/mellanox/passthrough-plugin/
- https://github.com/Mellanox/sriov-cni/
- WIP: integrate sriovnet with K8s cni plugin



Challenges and solutions

- Orchestration challenge
  - Isolation of character (network) device!
  - Isolation of sysfs files, attributes
  - CNI extension?
  - Kubernetes
    - device plugin and network plugin interaction?
      - device cgroup configuration
      - rdma cgroup configuration
      - sysfs bind mounts

Challenges and solutions (continue…)

- Net namespace resident IB devices
  - IB device resides in single net namespace
  - Optional mode

Thank You