Containing RDMA and High Performance Computing

Liran Liss

ContainerCon 2015

Mellanox TECHNOLOGIES

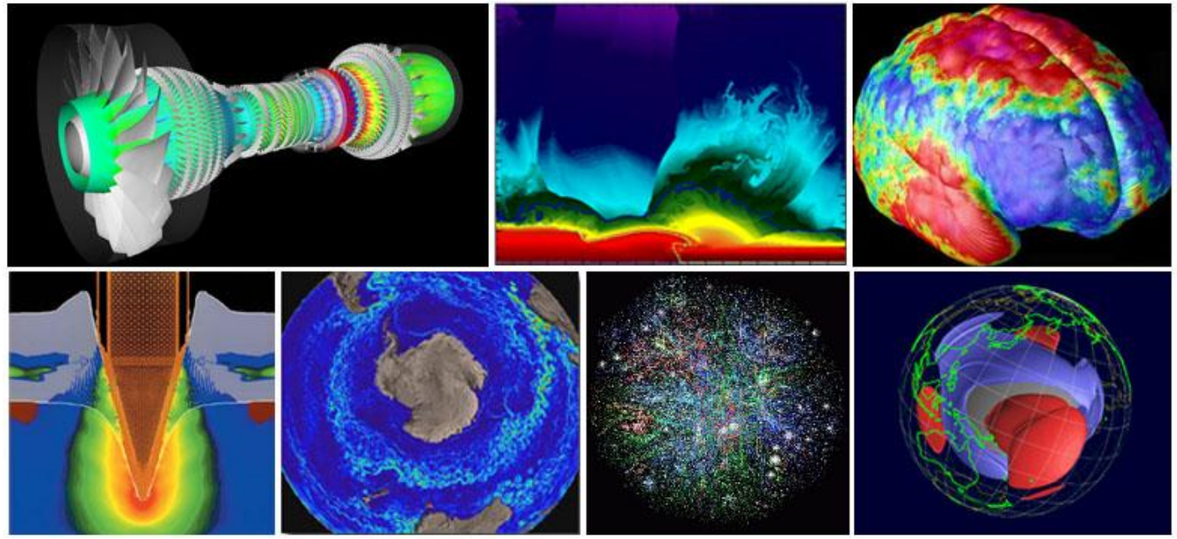
Connect. Accelerate. Outperform.™

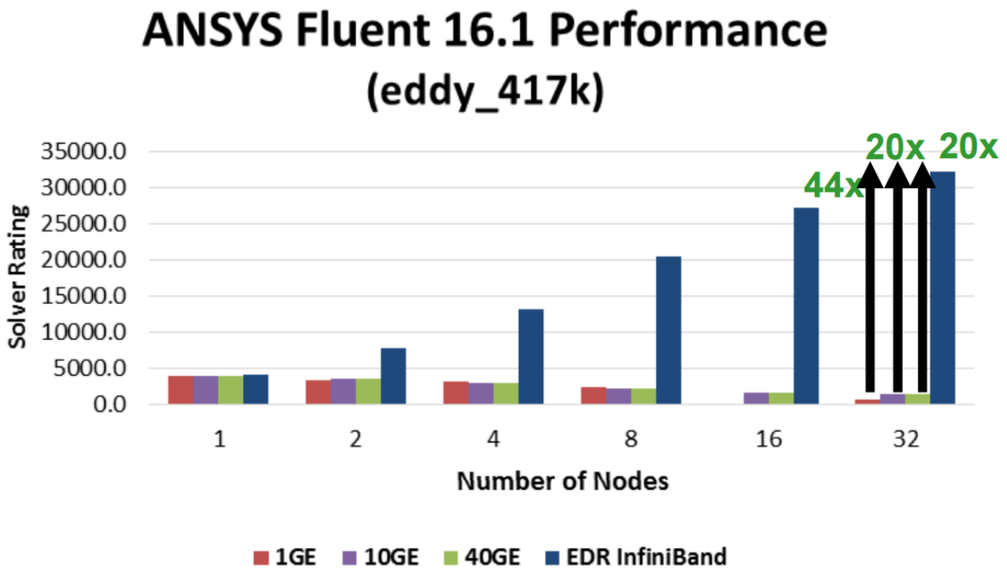
Agenda

- High Performance Computing (HPC) networking
- RDMA 101
- Containing RDMA
  - Challenges
  - Solution approach
- RDMA network namespace support
- RDMA controller
- Putting it all together
  - RDMA: **Infiniband** + **RoCE** (RDMA over Converged Ethernet)
  - Raw Ethernet: **DPDK** + **user-level TCP**
- Conclusions

HPC Networking

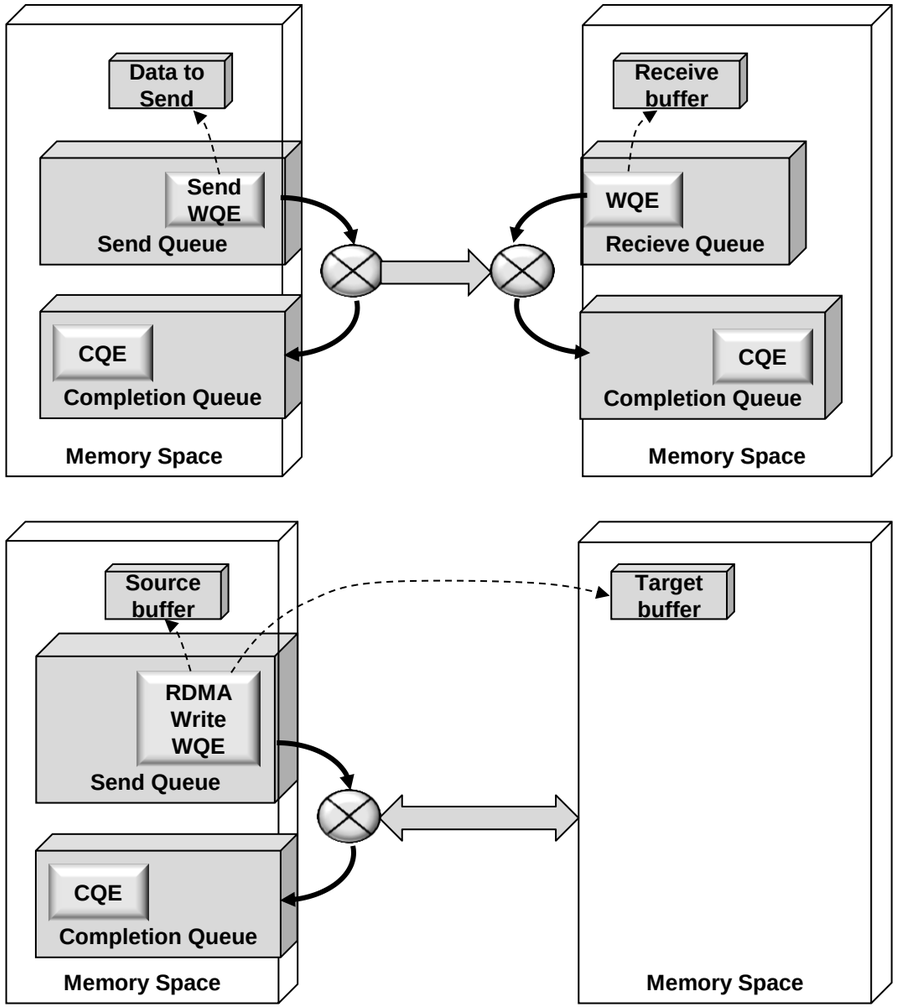
- Modern Super-Computers are typically clusters
  - Commodity servers
  - Commodity OSes
- Efficient communication is key to scaling
  - It's a lot harder to do the same at less time than do more at the same time
  - Communication / compute ratio increases with system size
- Traditional network stack challenges
  - Per message / packet / byte overheads
  - User-kernel crossings
  - Memory copies
- RDMA eliminates these overheads
  - 600ns application-to-application latencies
  - 100Gbps throughput





RDMA 101

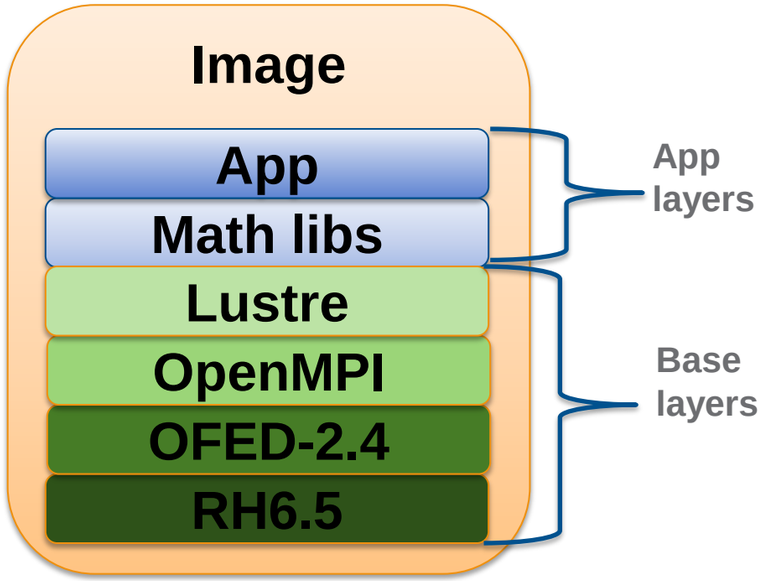
- Move traditional OS tasks to HW
  - Process isolation
  - Reliable delivery and protocol processing
  - Transport context
- User-level networking
  - System calls used for
    - Creating resources
    - Setting up connections
    - Registering memory
  - Data path is done entirely from user-space
    - Posting work requests
    - Polling for completions
- Asynchronous IO
  - Memory management delegated to applicaton
  - Zero-copy IO for all operations
- Semantics
  - Channel (sends and receives)
  - RDMA (Write / Read / Atomics)



Containing RDMA

- Is HPC and virtualization a contradiction?
  - Not if performance isn't sacrificed
    - MMU/IOMMU overheads
    - Interrupt delivery
    - Memory footprint
  - HPC applications may benefit from
    - Easy packaging of application dependencies
      - Independent infrastructure and application layers
    - Ease of deployment
      - Multiple user environments
  - HPC clouds are already happening

- Containers + RDMA: the best of both worlds
  - Efficient isolation and agility of containers
  - Performance of RDMA

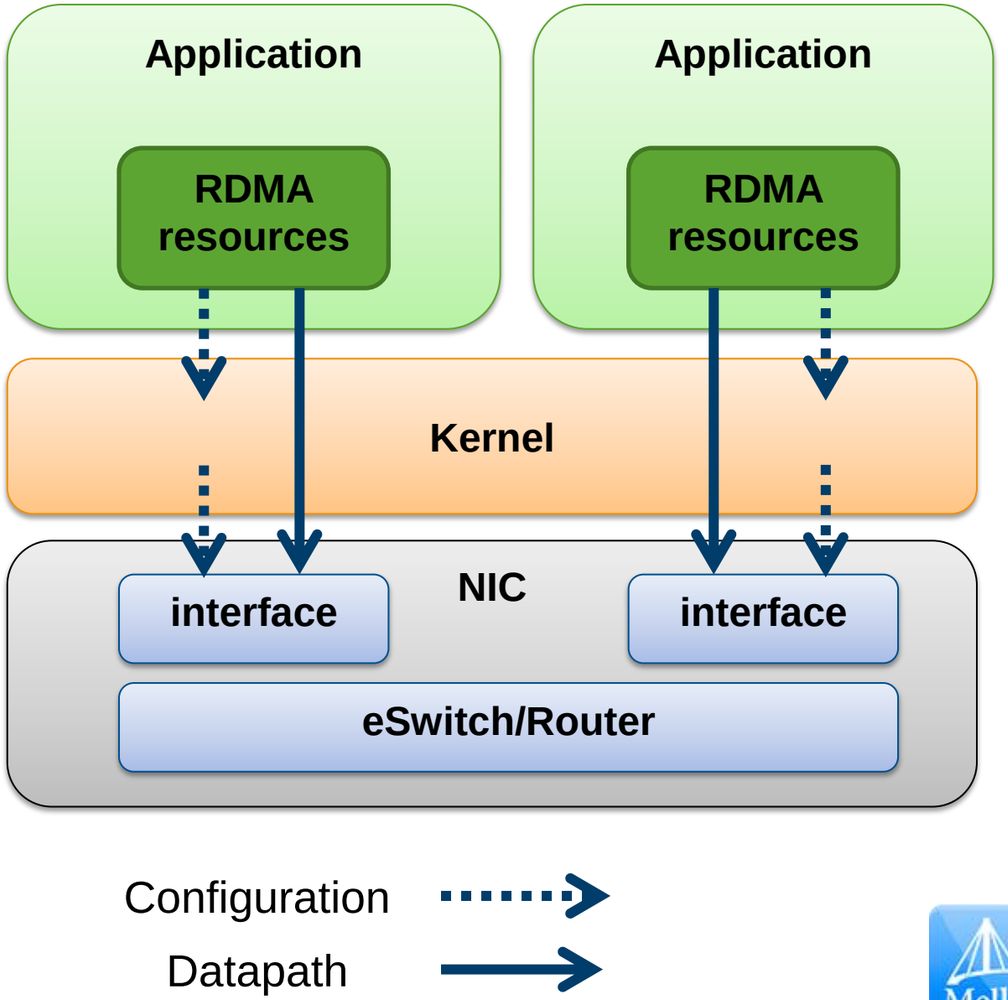


Challenge: Direct User Access to HW

- IO is initiated directly by the application
- Kernel not involved in the data path
  - Cannot classify or tag packets
  - Cannot modify packets

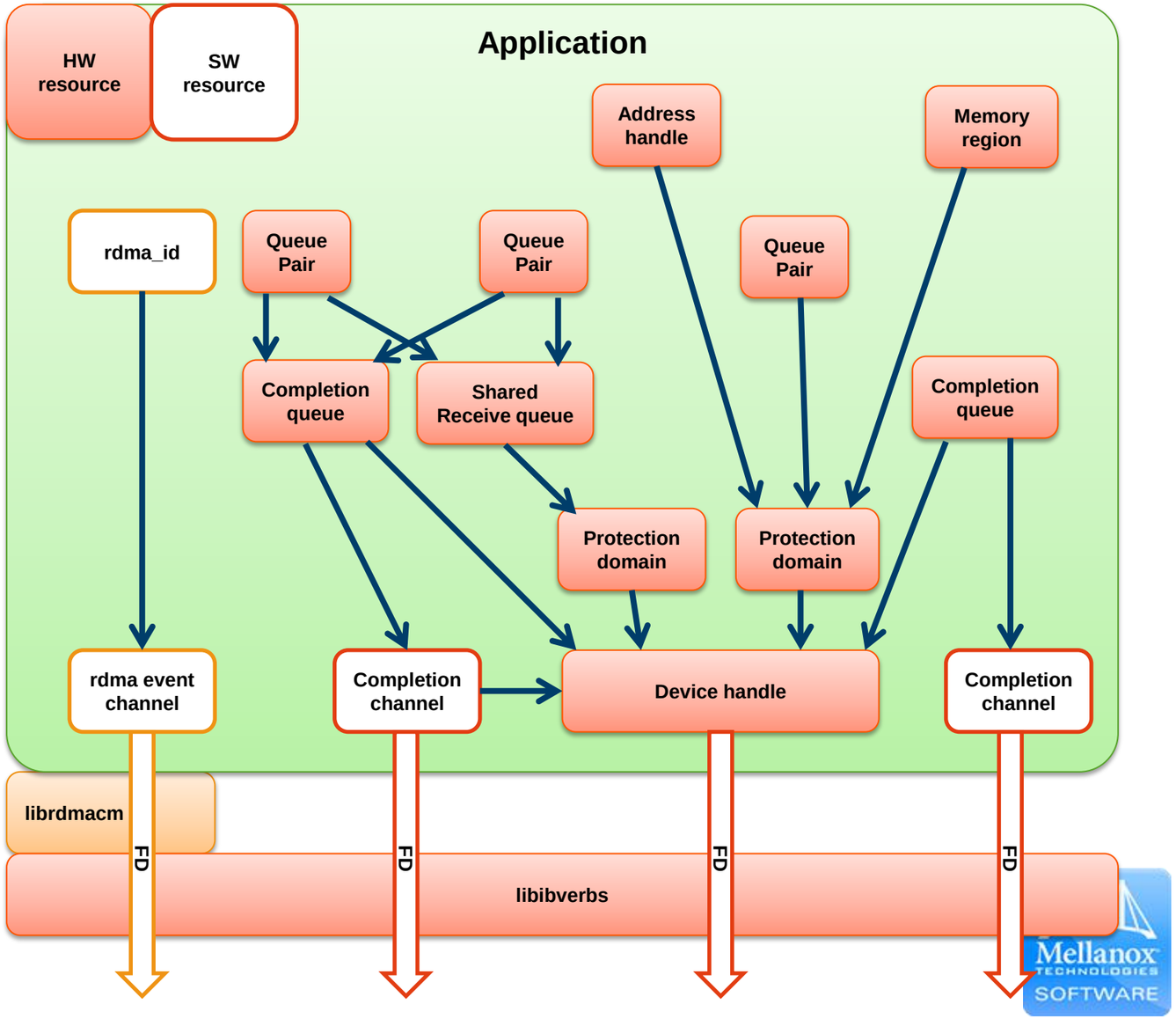
- Consequences
  - Cannot apply net_cls
  - Cannot apply net_prio
  - Cannot reflect arbitrary Linux routing or bridging

- Solution approach
  - Support interfaces that represent HW properties
    - Standard (untagged) Ethernet interface
    - VLAN interfaces
    - macvlan interfaces
    - IPoIB interface
  - Apply traffic constraints during resource creation
    - Addressing
    - QoS (user-priority / Service Level)



Challenge: Resource Rich

- ***Verbs*** API exposes Multiple objects
  - QPs, CQs, SRQs, MRs, PDs, AHs…
  - Backed by (finite) HW resources
  - Accessed by a single FD
- Consequence
  - Existing controllers/limits not granular enough
    - Memory
    - FD
    - Device files
- Solution approach
  - Introduce a new granular controller group



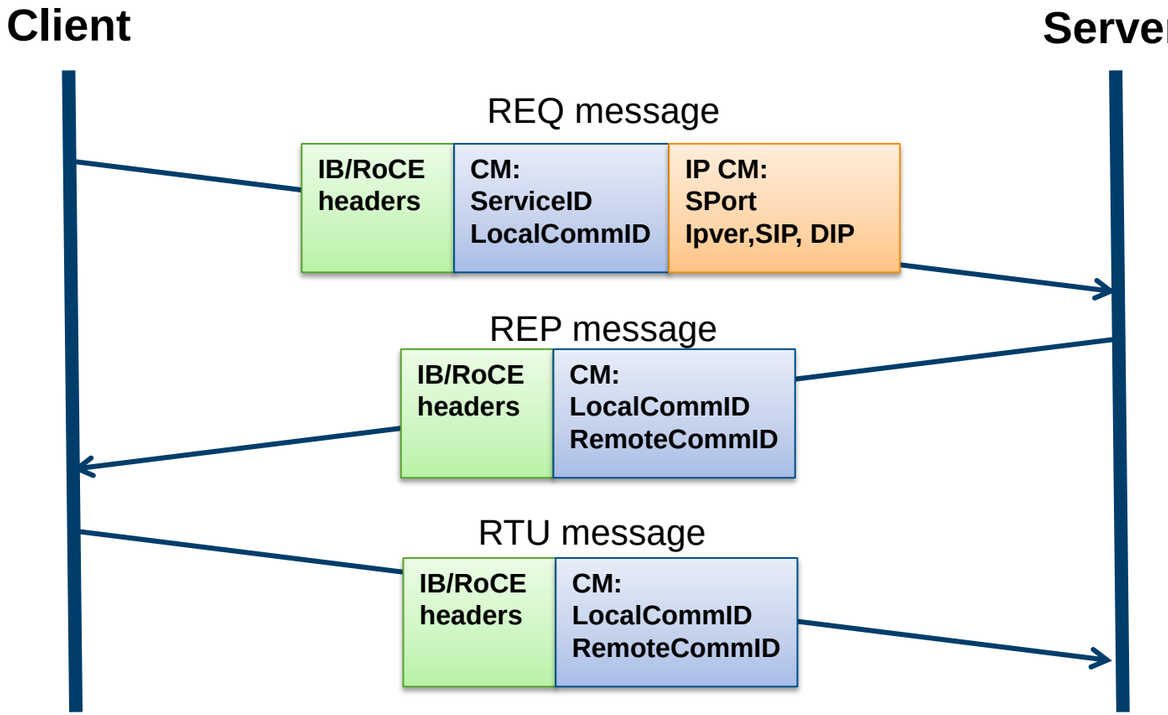
Challenge: RDMA Addressing

- Services are identified by ServiceIDs
  - 64-bit namespace
  - No well-known QP numbers
- RDMA addresses are different than TCP/IP
  - Infiniband uses LIDs and GIDs
  - RoCE (v2) uses UDP encapsulation
- IP CM
  - Maps TCP/UDP port spaces into ServiceIDs

0	1	2	3	4	5	6	7
0x00				0x01	IP prot	port	

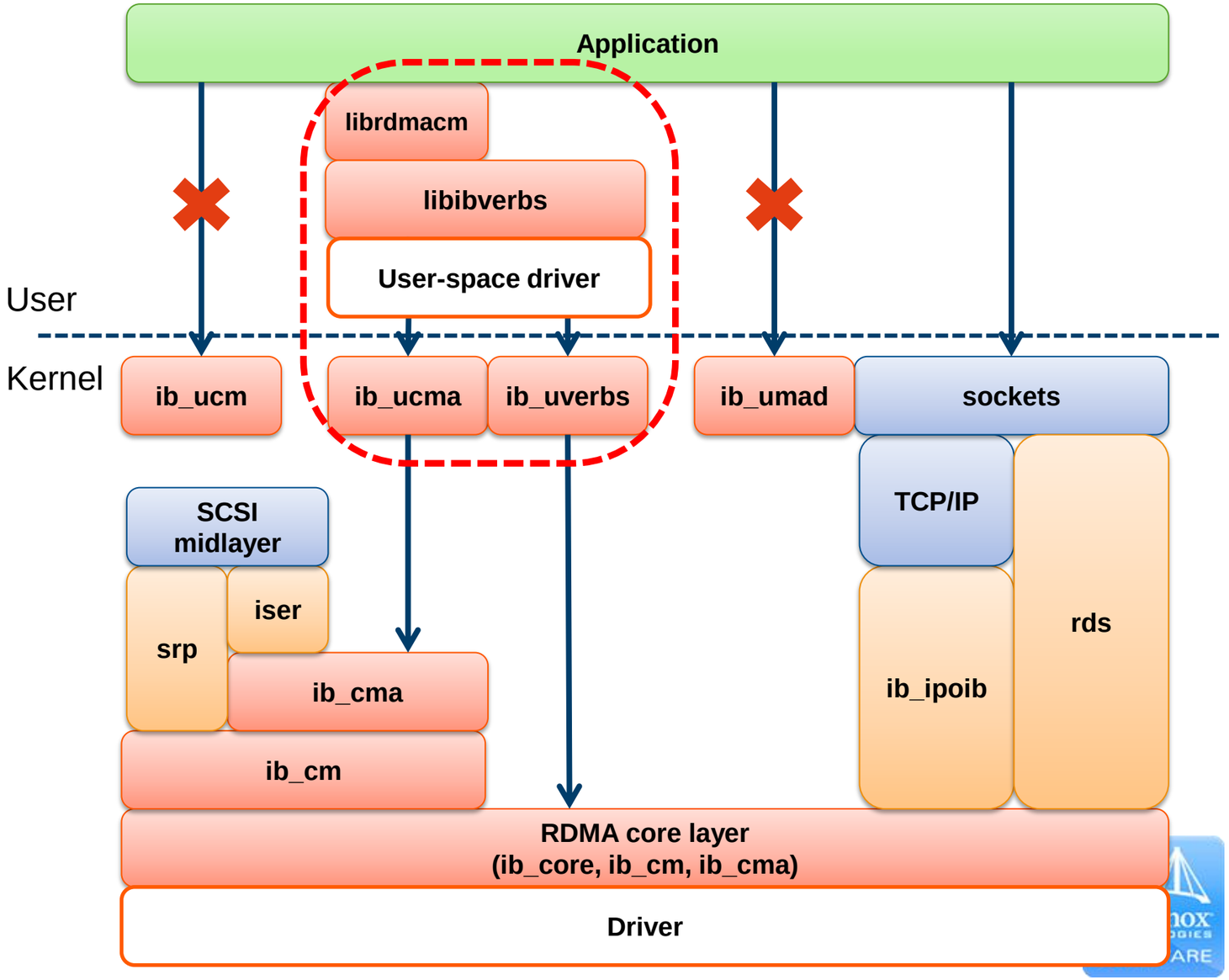
  - Carries IP addresses in extended message data
  - Implemented by librdmacm / CMA
- Consequence
  - Standard network namespaces do not apply directly to native RDMA addressing

- Solution approach
  - Support network namespaces for RDMACM connections



RDMA Containment Principles

- Focus on application APIs
  - Verbs / RDMACM
  - Exclude management and low-level APIs
    - E.g., umad, ucm
    - Deny access using device controller
  - Exclude kernel ULPs (e.g., iSER, SRP)
    - Not directly exposed to applications
    - Controlled by other means (blk_io)
    - Subject for future work
- Simplicity and efficiency
  - Containers may share the same RDMA device
  - Leverage existing isolation infrastructure
    - Native RDMA process isolation
    - Network namespaces and cgroups



Namespace Observations

- Isolating Verbs resources is not worthwhile
  - Only QPNs and RKeys are visible on the wire
  - Both don't have well-known names
    - Applications don't choose them
  - Share device ***RDMA namespace*** among multiple processes
    - Scales to 10K's of containers
- rdmacm maps nicely to network namespaces
  - IP addresses stem from network interfaces
  - Protocols and port numbers map to ServiceID port-spaces
- Network namespace required for RoCE L3→L2 address resolution
  - Connected QPs
  - Address handles

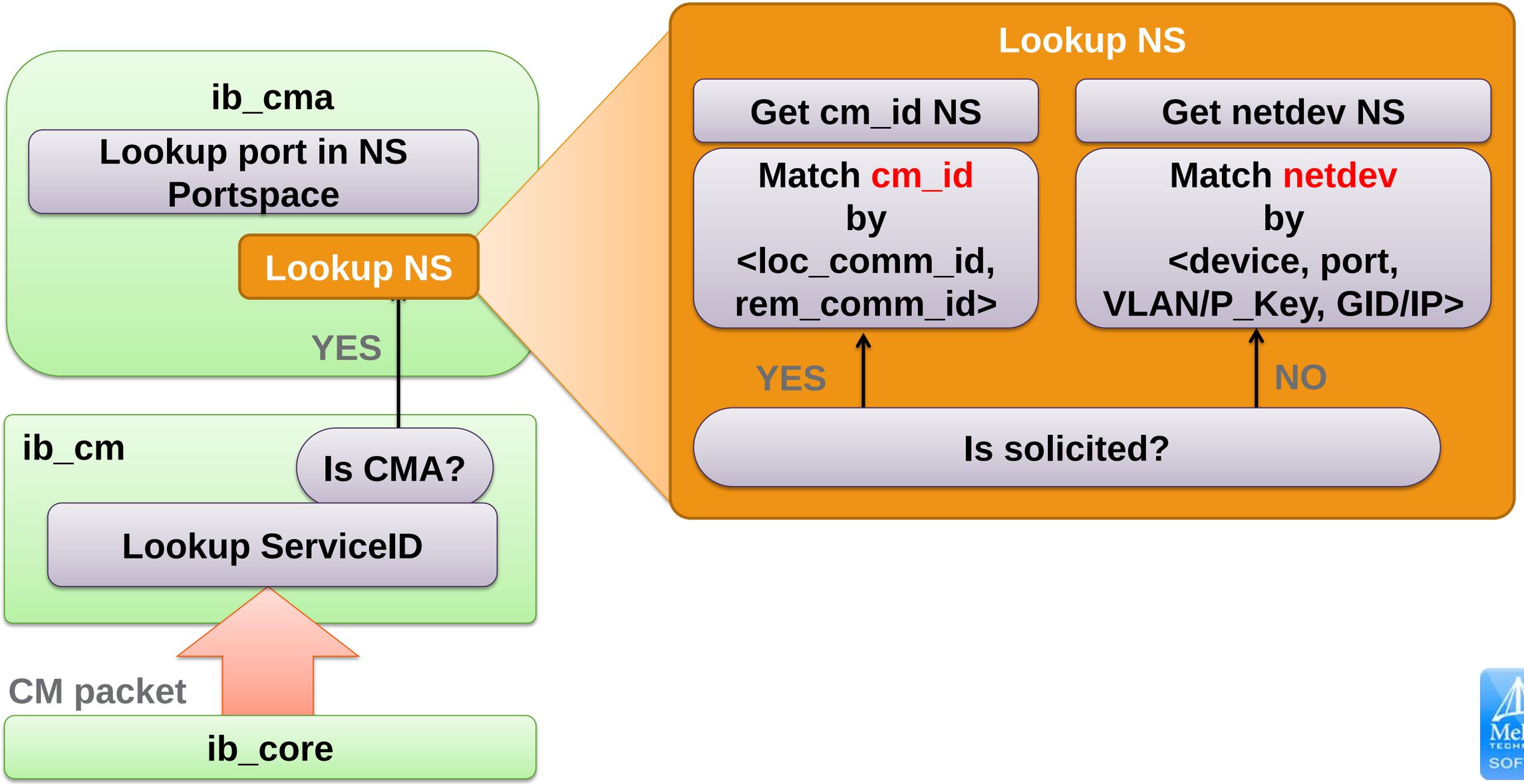
Conclusions

- **Support standard network namespaces via Isolated RDMACM port-spaces**
- **QP and AH API calls should be processed within a namespace context**
- **Associate RDMA IDs with namespaces**

Resource Namespace Association

- QP and AH namespaces
  - Determined by the selected GID index during API calls
    - Selects interface, namespace, and source IP
- RDMA IDs namespaces
  - Determined by the process namespace upon creation
  - Matched asynchronously with incoming requests
  - Default to Host namespace for kernel threads
- Namespace determined by HW interfaces
  - Physical port interfaces of PFs/VFs
  - Multiple IPoIB child devices on same / different P_Key
  - VLAN child devices
  - macvlan child devices

ServiceID Resolution



RDMA cgroup

- Governs application resource utilization
  - Per RDMA device
- Control resource usage
  - Opened HCA contexts
  - HCA resources
    - CQs, PDs, QPs, SRQs, MRs, AHs
- Control spoofing and QoS
  - Service Levels (SLs) and User Priorities (UPs)
  - Partition keys
    - List of allowed P_Key values
  - Interfaces (RoCE)
    - List of allowed GIDs (each represents an interface)
- Enforcement
  - During system calls
    - E.g., while creating QPs
  - During policy changes
    - Depends on resource type
  - During network changes
    - E.g., partition changes

Putting it All Together

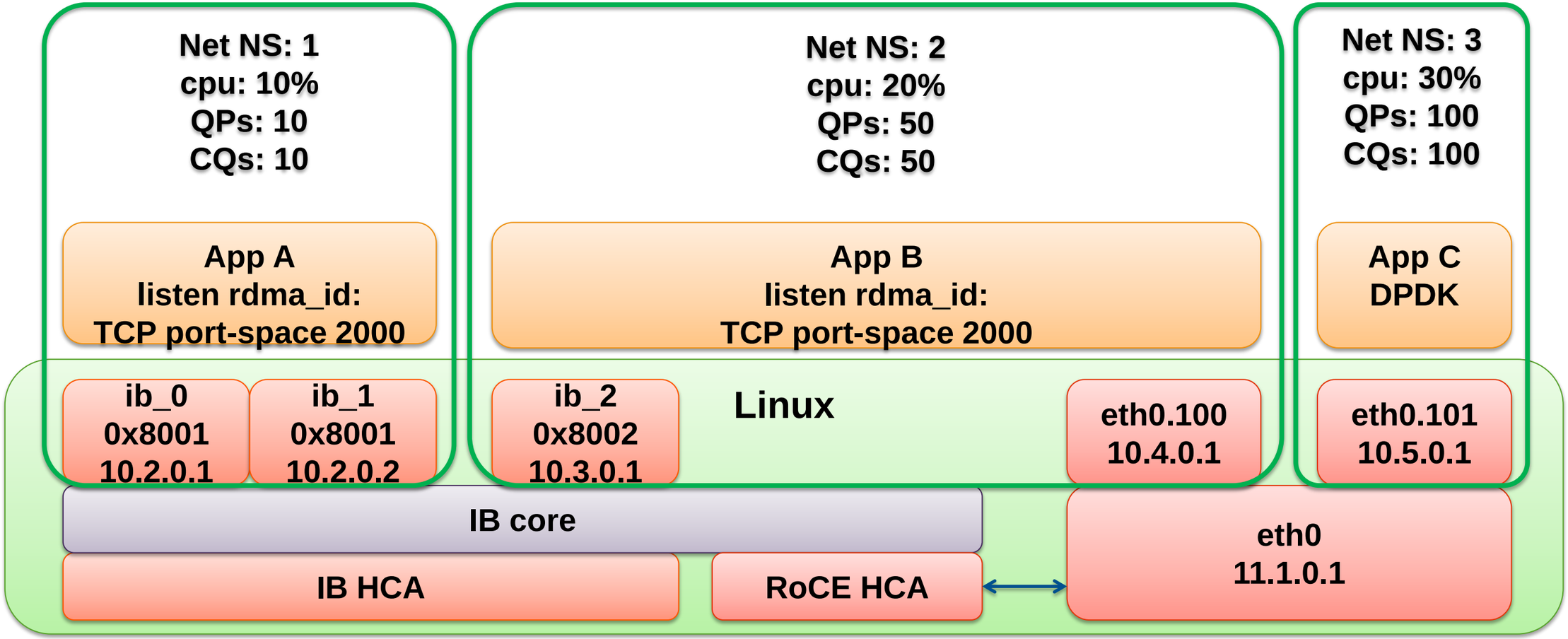
- Available today
  - Infiniband and RoCE in “host” namespace
  - Raw Ethernet queues (DPDK, user-space TCP)
    - Requires CAP_NET_RAW
- ServiceID namespace support for IB completed
  - Supports all IPoIB interfaces
  - First patch-set accepted for Linux 4.3
    - Multiplexes multiple RDMAIDs over a single ServiceID
- Coming up
  - Complete upstream IB namespace integration
  - RoCE namespaces
  - RDMA cgroup controllers
  - Runtime integration

```
# ./docker run
       --device=/dev/infiniband/uverbs0
       --device=/dev/infiniband/rdma_cm
       --ulimit memlock=-1
       -t -i centos /bin/bash
```

```
# ip link add link ib0 name ib0.8001 type
       ipoib pkey 8001

# pipework ib0 $CONTAINERID 10.1.0.1/16
```

```
# yum install -y libibverbs-utils libibverbs-
devel libibverbs-devel-static libmlx4 libmlx5
ibutils libibcm libibcommon libibmad
libibumad

# yum install -y rdma  librdmacm-utils
librdmacm-devel librdmacm libibumad-devel
perftest

# rdma_server
```

Putting it All Together (cont.)



Conclusions

- The intrinsic efficiency of containers make them an attractive virtualization and deployment solution for high-performance applications
  - E.g., HPC clouds, Supercomputers
- Infiniband, RoCE, DPDK, and user-space TCP/IP supported today in “host” namespace
  - SRIOV not required (!)
  - Scale to any number of containers
- RDMA namespace support allows running multiple rdmacm applications in isolation
  - physical interface assignment, bridging, and “pod” network models
  - Zero-overhead: forwarding is done by the HW embedded switch
- RDMA controllers shall prevent contained applications from monopolizing RDMA resources

Thank You

Mellanox TECHNOLOGIES

Connect. Accelerate. Outperform.™