2021 OFA Virtual Workshop

SUPPORTING LIVE MIGRATION OF VMS COMMUNICATING WITH BARE-METAL RDMA ENDPOINTS

Jorgen Hansen, Bryan Tan, Rajesh Jalisatgi, Adit Ranadive, Vishnu Dasa, Georgina Chua

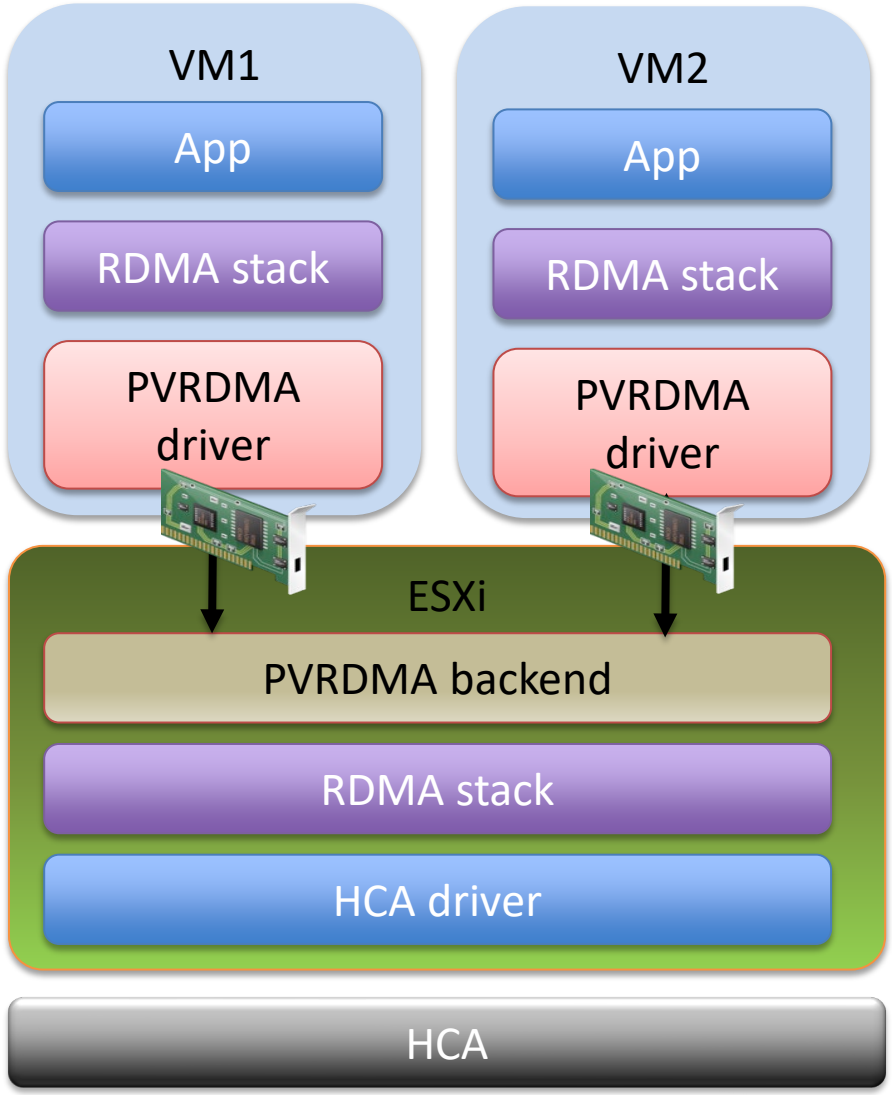
VMware, Inc.

AGENDA

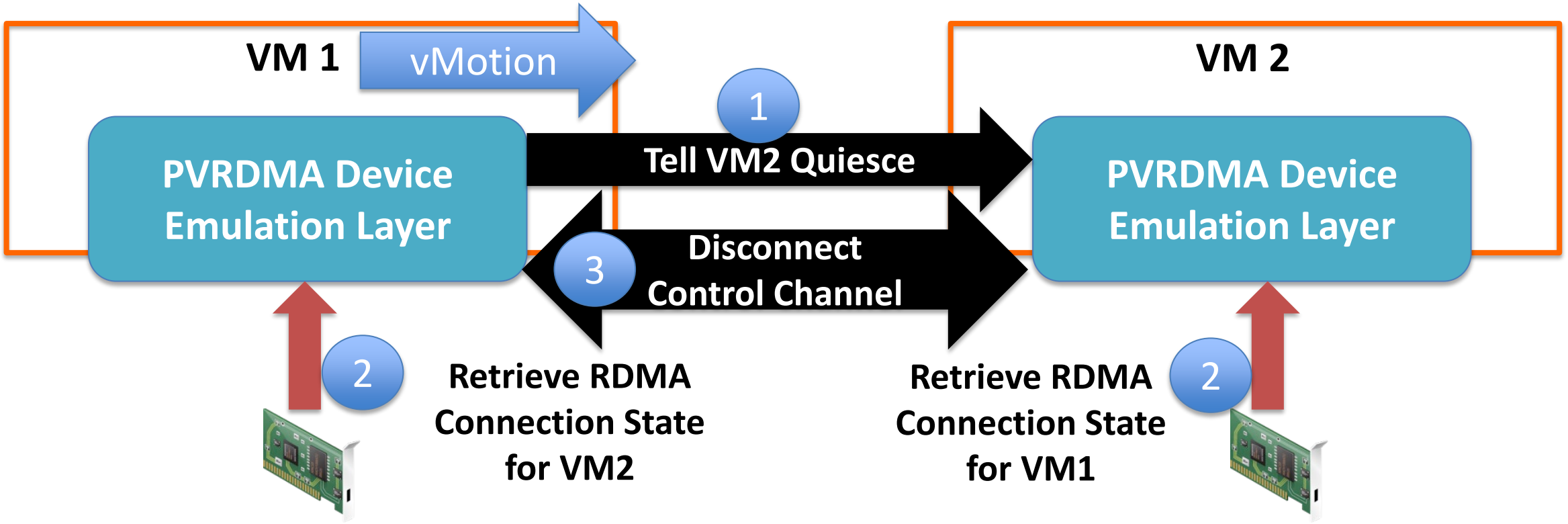
- Overview of Paravirtual RDMA Device
- Current live migration support for virtualized endpoints
- Supporting bare-metal endpoints with live migration
- Bare-metal Endpoint Example
- Conclusion

PVRDMA ARCHITECTURE

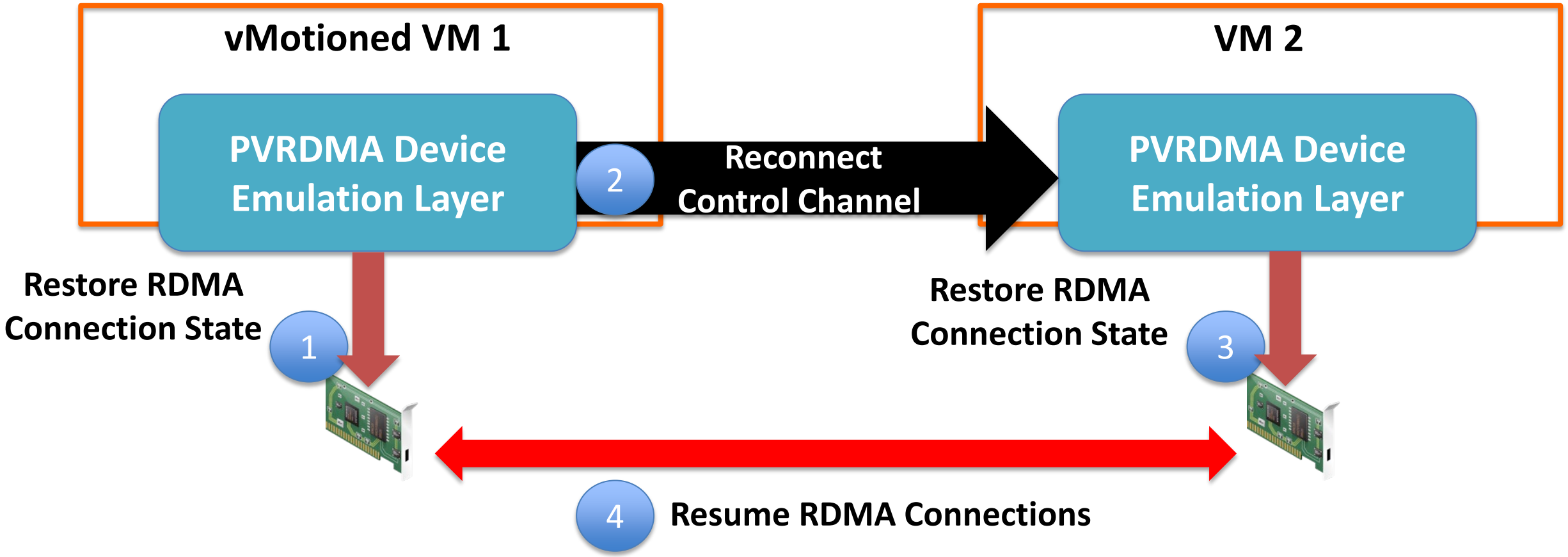
- **VM**
  - Expose a virtual PCIe device
  - Guest Support – Driver/Library
    - Implements RDMA APIs
- **PVRDMA backend**
  - Mediates guest access to HCA
  - Exposes an RDMA resource space for each guest
  - Invokes ESXi RDMA APIs in response to guest RDMA operations
  - Physical RDMA resources created for each guest
- **Physical HCA services all VMs on the host.**



SUSPENDING VM FOR VMOTION

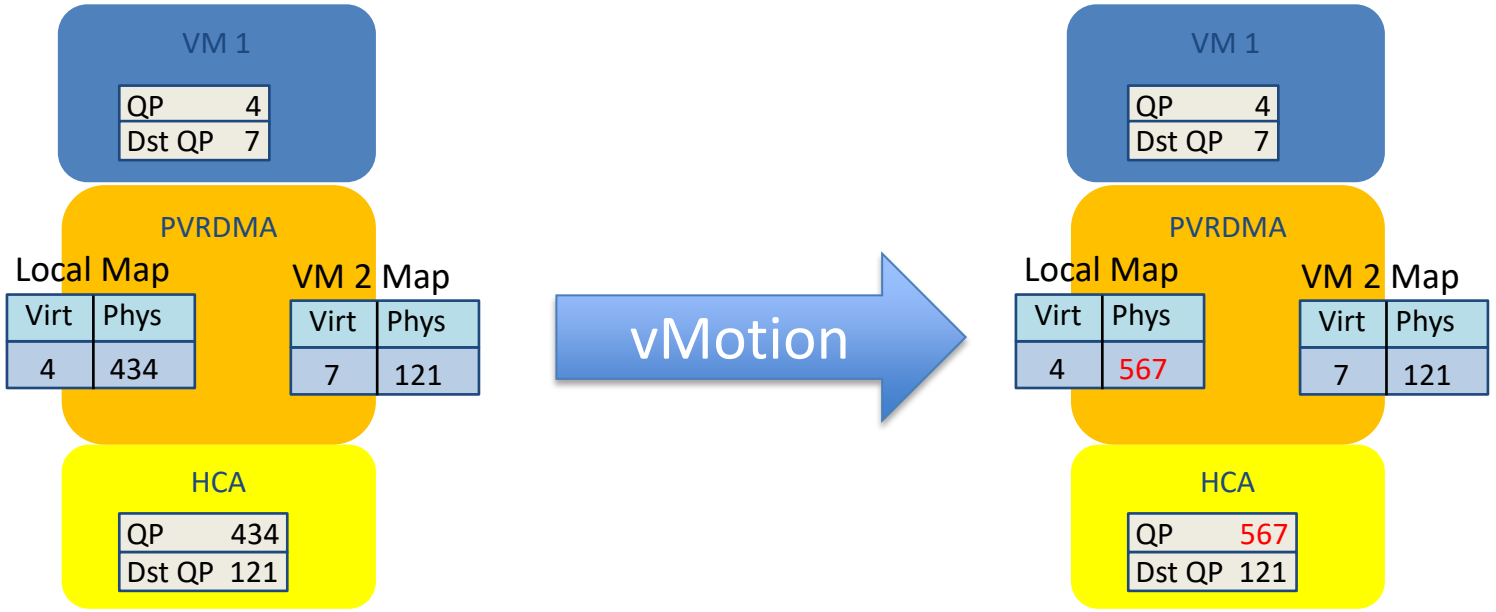


RESUMING VM AFTER VMOTION



RESOURCE MAPPINGS

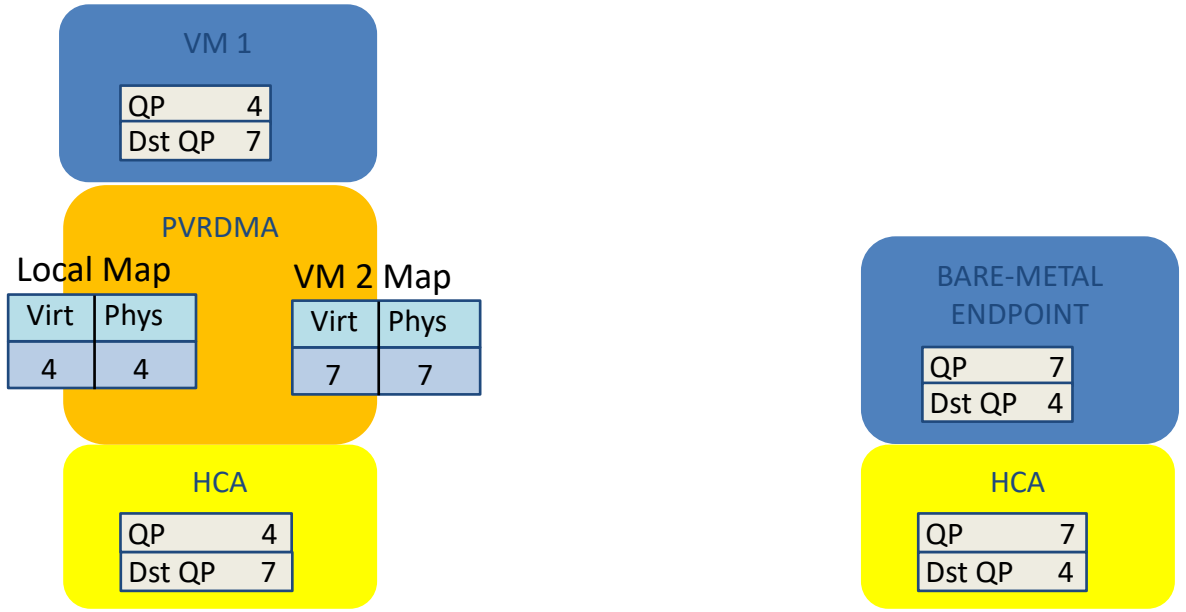
- On vMotion, hardware resources are recreated using normal verbs, so resource IDs change.
- Virtual resource ID are unchanged.
- Only virtualized endpoints perform this mapping.



BARE-METAL ENDPOINTS

COMMUNICATING WITH BARE-METAL ENDPOINTS

- **Bare-metal endpoints exchange physical resource IDs with VM applications/ULPs**
  - Physical resource IDs are now embedded in VMs.
  - Resource mappings must be identity mappings.
- **Use resource IDs allocated by HCA for virtual resources**
- **ESXi 7.0 U1 introduced PVRDMA support for bare-metal endpoints without vMotion.**



CHALLENGES FOR BARE-METAL WITH LIVE MIGRATION

- **The coordination protocol used to ensure consistent queue pair state isn't available for bare-metal endpoints:**
  - PVRDMA can stop outgoing operations.
  - RDMA operations initiated by the bare-metal endpoint are outside our control.
- **VMware vMotion uses a pre-copy phase, where memory is transferred while the VM is still active:**
  - Uses page traces to detect dirty memory that needs to be retransmitted.
  - A notification protocol between the VM and the peers is used to detect memory updates from remote writes.
- **Since multiple VMs may share the HCA – cannot guarantee that a new resource can be created with the same resource ID when restoring device state.**

HARDWARE SUPPORT FOR BARE-METAL LIVE MIGRATION

- **RDMA Namespaces:**
  - Allow the creation of queue pairs and memory regions with caller defined public identifiers, i.e., QPNs and rkeys. The identifier space must be isolated from other VMs on the same host, e.g., two VMs should be able to allocate the same identifier for different GIDs.
- **Hardware suspend/resume support:**
  - Pausing and resuming processing of a queue pair on the HCA.
  - Retrieving and updating the protocol state of a queue pair on the HCA.
- **Dirty page tracking during pre-copy:**
  - Getting notifications, when the HCA accesses guest memory.

RDMA NAMESPACES

Allowing Any Resource ID to be recreated after vMotion

RDMA HCAs support multiple namespaces:

- Each namespace identified through MAC and VLAN

Each namespace allows allocation of full range of resource IDs

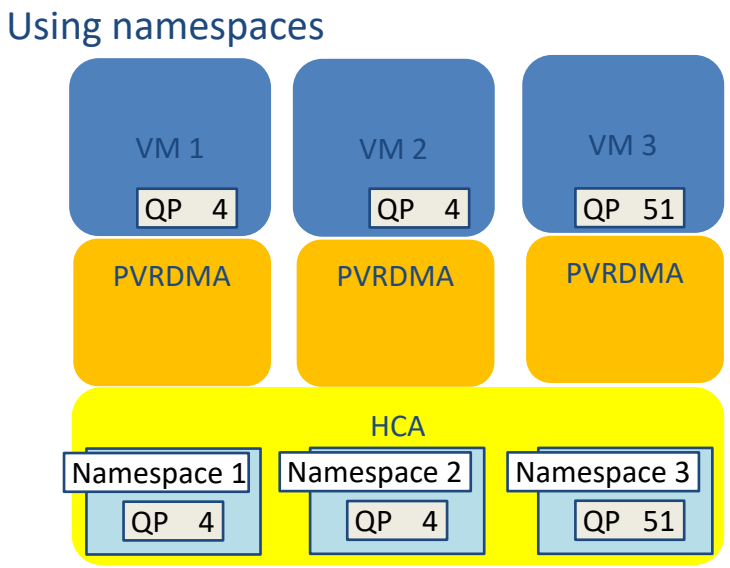
Each VM is allocated a namespace:

- Virtual and physical resource IDs are the same
- No need for translations

On vMotion:

- A namespace is created on destination host
- Existing QPs and MRs are recreated with specific state including resource IDs
- Eliminates the metadata sent between VRDMA peers!!

Improves vMotion Time and PVRDMA Scalability

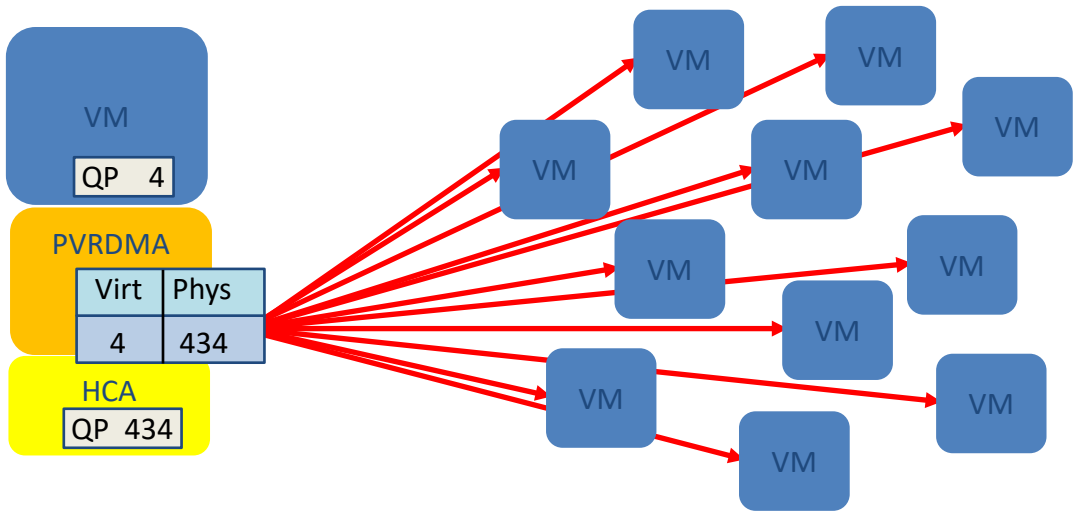


Introduced in ESXi 7.0:

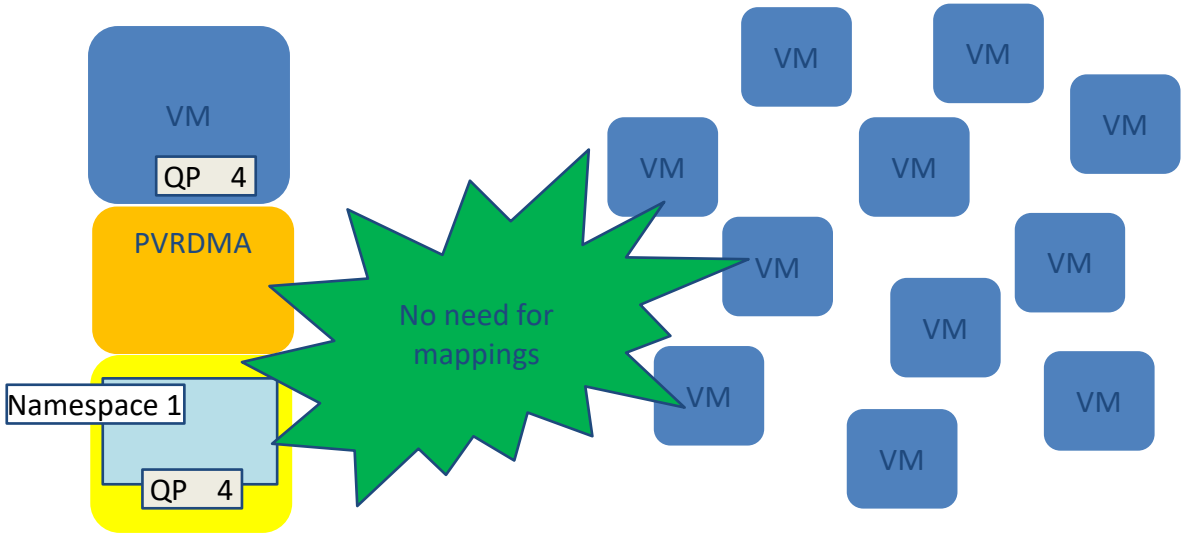
- **Developed in collaboration with hardware partners NVIDIA Mellanox and Marvell**

IMPROVED SCALABILITY THROUGH RDMA NAMESPACES

No namespaces – must distribute mappings



With Namespaces no need for mappings



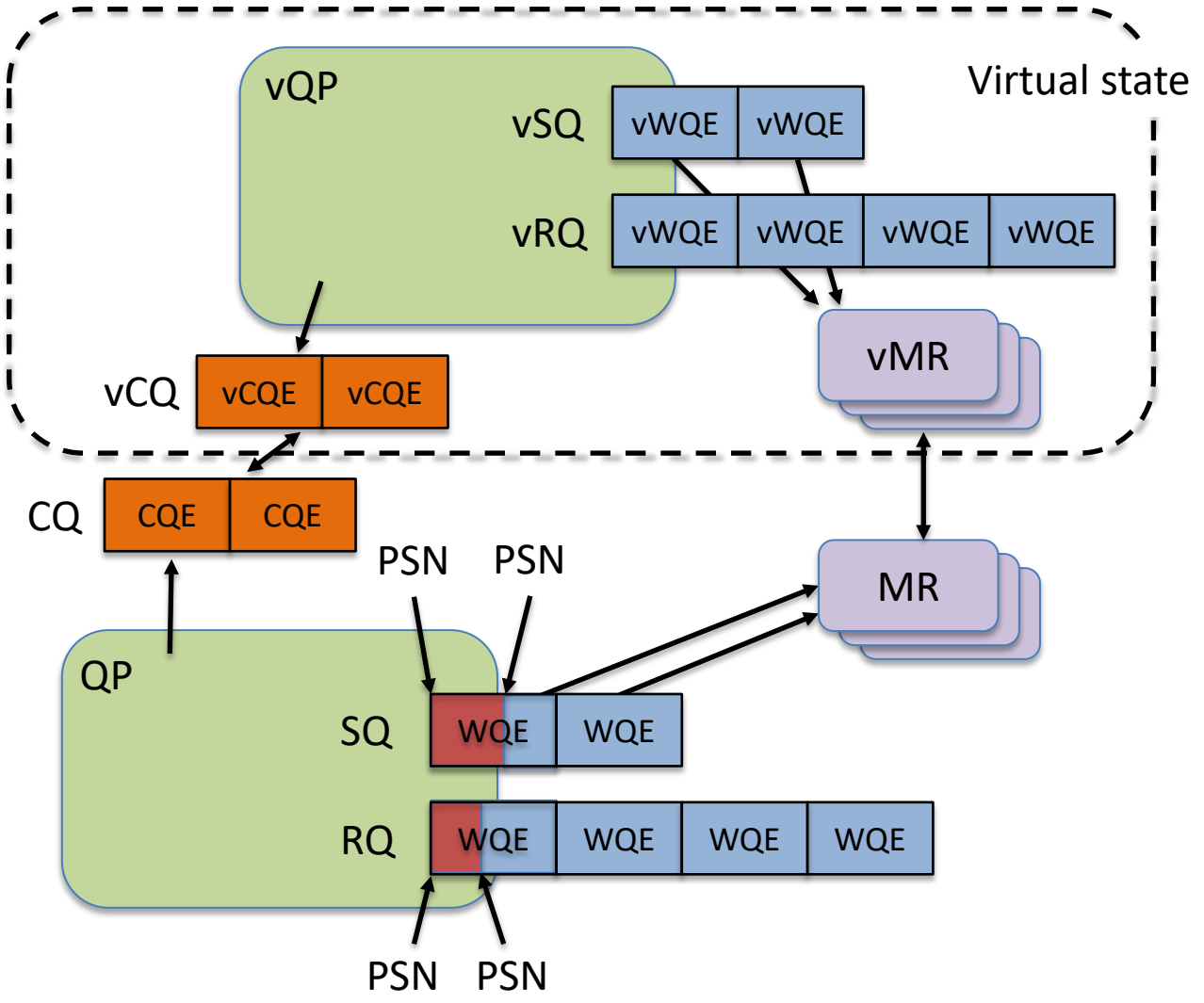
Reduction in book-keeping traffic when using namespaces improves performance – in particular for workloads with high resource churn.

HARDWARE SUPPORTED SUSPEND RESUME

- **Support four new operations:**
  - Suspend queue pair - the queue pair processing will be suspended in its current state.
  - Query suspended queue pair – retrieve the queue pair “runtime” state from a suspended queue pair.
  - Create suspended queue pair – recreates a queue pair with creation attributes and runtime state but in suspended mode.
  - Resume queue pair – resumes processing of a queue pair that was either suspended or created as suspended.
- **While a queue pair is suspended, it may continue to respond to requests from the peer with RNR NAKs, such that the peers will not terminate the connection.**
- **The queue pair runtime state specification is vendor neutral:**
  - How much of currently executing WQEs has been processed.
  - State of active incoming RDMA write and outgoing RDMA read operations.
  - Latest processed atomic operations.
- **Developed in collaboration with hardware partners NVIDIA Mellanox and Marvell.**
- **Will be available in a future release.**

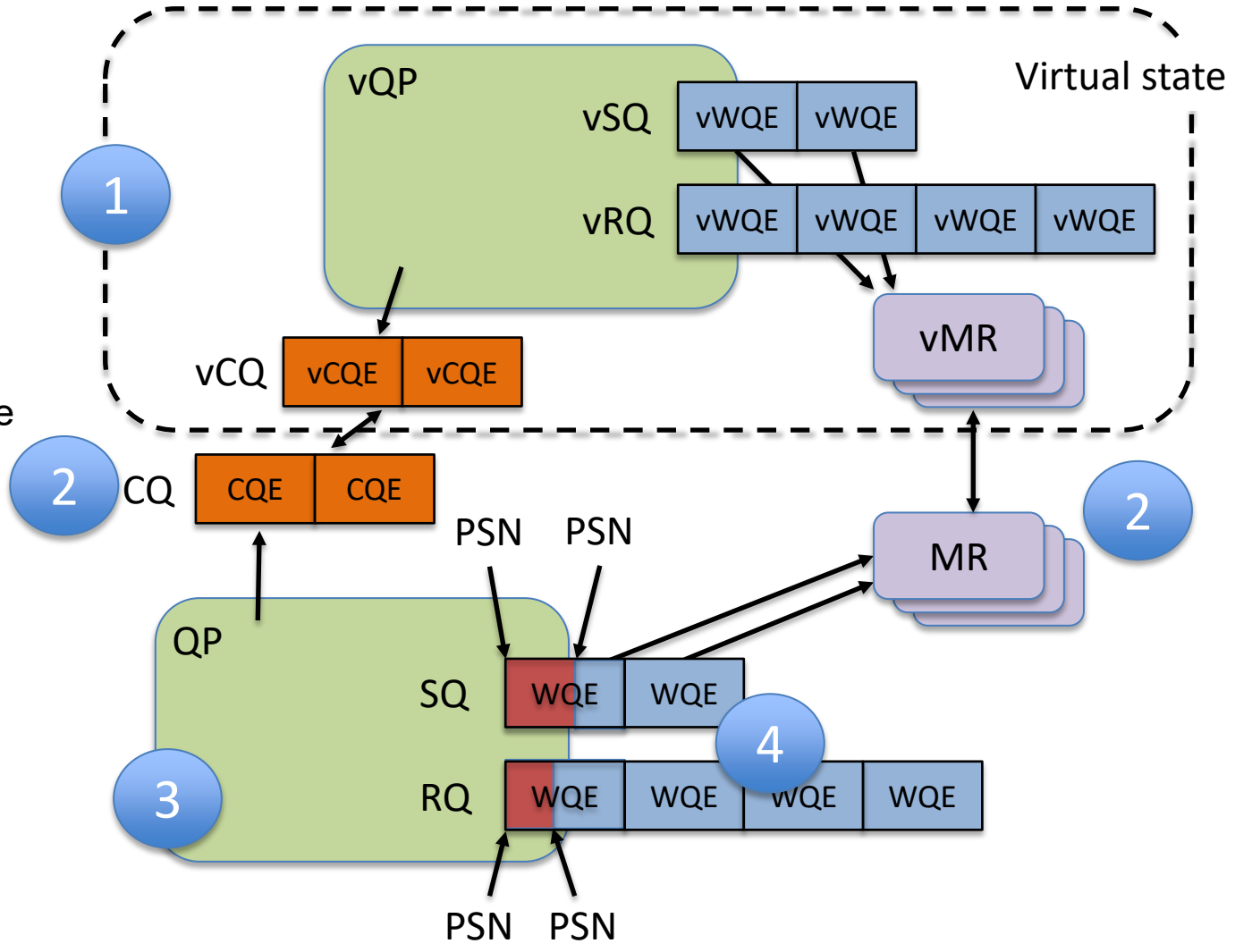
CAPTURING HARDWARE QUEUE PAIR STATE

1. **Suspend queue pair**
   - Once suspended, no further access to host memory.
2. **Poll completion queues**
   - Collect any completions and update virtual send and receive queues.
   - All queue pairs sharing a completion queue will be suspended before the CQ is polled.
3. **Retrieve queue pair state:**
   - Captures state of partially completed WQEs – no need to wait for large transfers to complete:
     - PSNs are used to track ACK'ed data.
   - The actual WQEs in the send and receive queues are not part of this state. This is tracked as part of the virtual state by PVRDMA.



RESTORING HARDWARE QUEUE PAIR STATE

1. **Restore virtual queue pair state**
2. **Recreate physical RDMA resources used by queue pair:**
   - Protection domains
   - Completion queues
   - Memory regions with same rkey/lkey as on the source host
3. **Create suspended Queue Pair:**
   - No processing yet
   - Empty send and receive queues
4. **Repost work requests to send and receive queues:**
   - In same order as on source host
5. **Resume processing of Queue Pair**

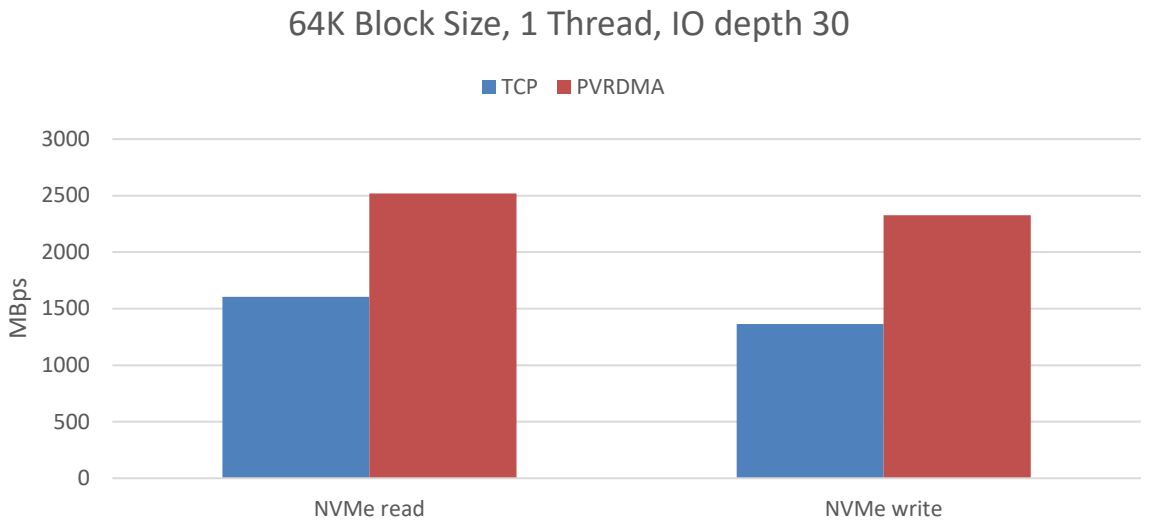


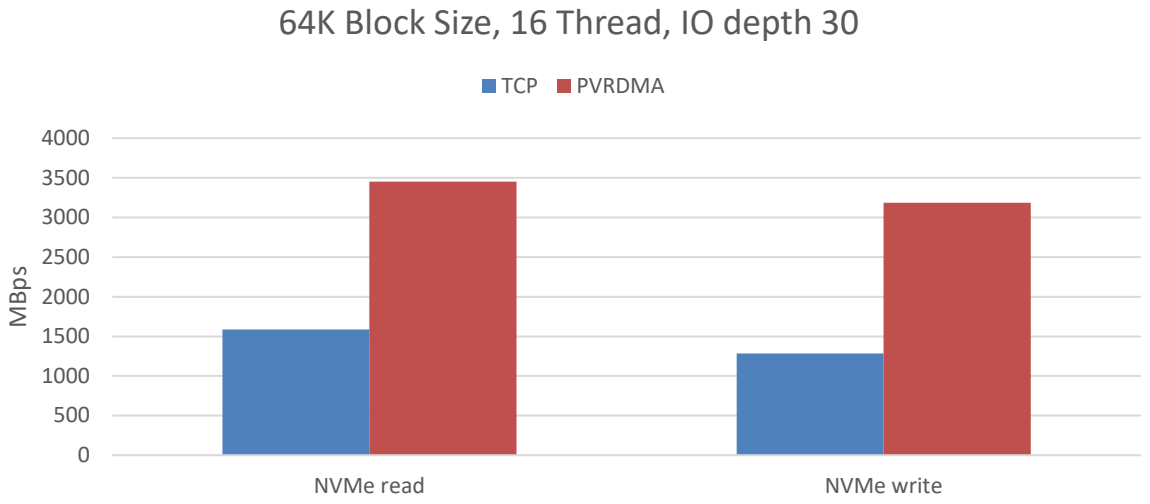
HANDLING PRE-COPYING OF VIRTUAL MACHINE STATE

- **For vMotion, the virtual machine memory state will be pre-copied to the destination to reduce downtime:**
  - Changes to virtual machine memory is tracked, and any dirty pages are resent.
  - The working set of the virtual machine will be copied after the virtual machine is suspended.
- **For RDMA, the HCA may access virtual machine memory directly through RDMA writes:**
  - Dirty pages may not be detected.
- **HCA notifies PVRDMA of dirty pages**
  - HCA collects write traces:
    - Can be coarser grained tracing than page size
  - PVRDMA can request notification when new traces are available or use polling.
  - Only active during pre-copying.
- **Will be available in a future release**
  - Developed in collaboration with hardware partner Marvell.
- **Future alternative to HCA support is to use IOMMU based dirty page tracking.**

VM USING BARE-METAL NVME-OF TARGET

- **Provide virtual machines access to bare-metal storage over RDMA:**
  - Cluster file systems for HPC type workloads
  - Long term: fast storage access for broader range of VMs
- **Bare-metal live migration support allows any VM to use bare-metal storage.**
- **Recent work on PVRDMA has improved the performance of kernel ULPs using DMA MRs:**
  - Measurements with FIO show 50% - 100% improvements for larger block sizes (64K+).
  - Configuration:
    - Initiator is VM with PVRDMA
    - Target is VM with DirectPath I/O HCA using SPDK
    - HCAs are NVIDIA Mellanox CX-4 40G cards





CONCLUSION

- **Support for vendor neutral live migration of any VM regardless of peer endpoints.**

- **Possible with hardware support in the following areas:**
  - Hardware supported suspend/resume
  - Isolated resource ID namespaces
  - Dirty page tracking for RDMA writes

- **Strong collaboration with hardware partners:**
  - NVIDIA Mellanox
  - Marvell

2021 OFA Virtual Workshop

THANK YOU

Jorgen Hansen, Staff Engineer

Vmware, Inc.

BACKUP SLIDES

PERFORMANCE OF NVMEOF OVER PVRDMA AND TCP USING INTEL SPDK

Configuration:

1. Target is Native Endpoint, Initiator is a VM. Only one VM per host. VM has 4 vCPUs configured for test
2. Native endpoint with Mellanox CX-4 card in passthrough mode. Target has Rhel 8.0
3. Initiator is configured in VM. Same VM used for TCP and pvrdma. VM has Rhel 8.3, nvme-tcp module is still in tech preview (> RHEL 8.1)
4. ESXi build from main (mid jan), Tools: FIO, Intel SPDK, vmxnet3 used for TCP. TCP/PVRDMA not optimized for performance.
5. Two hosts connected through switch. Cards used are CX-4 cards of 40Gbps. Last column performance gain with PVRDMA compared to TCP
6. In limited benchmarks with higher block size (64k), PVRDMA seems to have significant performance gain compared to TCP in serial workload(~ 50% – 100%)

Operation	Blocksize	Threads	Iodepth	NVMeOTCP(MBps)	NVMeOPVRDMA(MBps)	% gain with PVRDMA
NVMe READ	64k	1	16	481	761	58.21205821
NVMe READ	64k	1	30	1604	2520	57.10723192
NVMe READ	64k	1	256	1432	2721	90.01396648
NVMe READ	64k	16	30	1588	3454	117.5062972
NVMe Write	64k	1	16	326	662	103.0674847
NVMe Write	64k	1	30	1365	2326	70.4029304
NVMe Write	64k	1	256	1292	2173	68.18885449
NVMe Write	64k	16	30	1284	3184	147.9750779
NVMe Rand Read	64k	1	16	470	753	60.21276596
NVMe Rand Read	64k	1	30	1634	2483	51.95838433
NVMe Rand Read	64k	1	256	1517	1763	16.21621622
NVMe Rand Read	64k	16	30	1582	4120	160.4298357
Nvme Rand Write	64k	1	16	276	673	143.8405797
Nvme Rand Write	64k	1	30	1332	2387	79.2042042
Nvme Rand Write	64k	1	256	1273	1746	37.15632364
Nvme Rand Write	64k	16	30	1340	3240	141.7910448