Towards a Lightweight RDMA Para-Virtualization for HPC

Shiqing Fan¹, Fang Chen¹, Holm Rauchfuss¹, Nadav Har’El², Uwe Schilling³, Nico Struckmann³

¹IT Software Infrastructure Laboratory, Huawei
²ScyllaDB
³HLRS, University of Stuttgart
Background

- Virtualized HPC and HPC Cloud become more popular and requires lightweight solutions (unikernel-like OS)
  - Cloud: Flexibility, Scalability, Reliability, …
  - HPC: Power and extreme speed of computation and data analysis
- Challenging workloads regarding latency throughput, may benefit from RDMA
  - RDMA virtualization is required

Why not SR-IOV?
RDMA virtualization: State of the art

- **Control Path:** RDMA verbs translated to host

- **Data Path:** RDMA memory mapped to backend driver and RDMA HCA

### vRDMA
- Solution from VMWare
- Support RDMA API
- Supports ESXi platform
- Not open source
- Public results

### Endure
- Solution from Microsoft
- Support RDMA API
- Support Azure cloud and Windows/Linux guest
- Not open source
- No public results

### HyV
- Solution from IBM
- Support RDMA API
- Support Linux guest/host
- Hypercall for control path
- Open source
- Public results
**virtio-rdma: Design Overview**

- High bandwidth, low latency network I/O
- Lightweight, flexible, and portable
- Targeting at unikernel-like OS, **OSv**:
  - Fast, lightweight, less overhead
  - Bridge Cloud and HPC
- Support socket and RDMA verbs API
- Shared memory for the intra-host communication
- Support RoCE* and InfiniBand

*RoCE: RDMA over Converged Ethernet*
virtio-rdma: Control Path

- Re-use the HyV backend driver
  - Ported to a newer Linux kernel
- Guest side implementation on Osv
  - Re-implemented frontend driver, hypercall
  - Streamlined/minimized OFED support
- Data path: RDMA memory is mapped directly from guest to host and HCA
  - Completion Queue, Queue Pairs, and Work Requests

Lightweight, flexible and portable
**virtio-rdma: Context Switch**

- Context switches are reduced
  - OSv runs application in the same privilege level, single address space
  - No boundary of user/kernel space
  - Less calls to external libraries
- Minimum number of memory copy
  - Only a few memory copies are needed for OFED core data structure

![Diagram of User Command Buffer](image1)

![Diagram of Kernel Command Buffer](image2)

Less context switch & 2-4 copies avoided per call
**virtio-rdma: Memory Mapping**

- Guest allocate contiguous memory in virtual address space
- Frontend driver remap the virtual address to guest physical address with chunks
  - HyV needs to take care of the non-contiguous physical address and the offset in the page
  - OSv always tries to allocate contiguous mem.
  - virtio-rdma only cares for the offset
- Backend driver remaps the chunks into host physical address

**Efficiency on address translation:**
Saved N-3 translation
(N >=3: num of pages)
virtio-rdma: Future Work

• Shared memory
  ○ Based on IVSHMEM
  ○ Protocol switch in RDMA verbs API
  ○ Communication buffers, e.g. user memory, shared to the VMs of same host
  ○ Update RDMA memory regions directly, e.g. Completion Queue

• Socket API support
  ○ forward to use RDMA and shared memory communications
  ○ re-implement rsocket or libvma onto OSv
  ○ no modification to the user application is needed

• HPC Integration
  ○ Setup the necessary environment via Torque when VM starts
Demo

root@host1:/home/demo
root@host1:/home/demo#
Conclusion

- Redesigned and Implemented the new virtio-rdma driver on OSv
  - Compatible with HyV backend driver, i.e. same hypercall
  - Fewer OFED dependencies and fewer context switches
  - Simpler memory translation
  - Flexible and portable for other unikernel-like OS
- Streamlined OFED user libraries and core structures for OSv
- Reused the HyV backend driver and ported it to newer Linux kernel
- HPC Cloud enabled
Acknowledgement

- This work is supported by the MIKELANGELO project co-founded by the European Commission (H2020 framework programme)
- Many thanks to the co-authors
- Special thanks to Jonas Pfefferle at IBM Zurich (author of HyV)
Thank you!