Virtio-SCIF: Enabling Xeon Phi capabilities on Virtual Machines

Stefanos Gerangelos and Nectarios Koziris {sgerag,nkoziris}@cslab.ece.ntua.gr

National Technical University of Athens
CSLab

Motivation

- **Coprocessor Virtualization**
- Use Case: *Intel Xeon Phi*
- Currently, it is not possible for many VMs to share a single Xeon Phi
- Full coprocessor virtualization involves complex implementation details
- Current "alternatives":
  - Direct device assignment to a single VM (via PCIe passthrough)
  - TCP/IP through Xeon Phi virtual network device

Xeon Phi Modes of Execution

i) **Native Mode**: applications are executed in a coprocessor-only way

ii) **Offload Mode**: the coprocessor can be used by the host as an accelerator by offloading certain workloads

iii) **Symmetric Mode**: allows both the host CPUs and the coprocessors to be involved in the execution of MPI processes in a symmetric way

Symmetric Mode

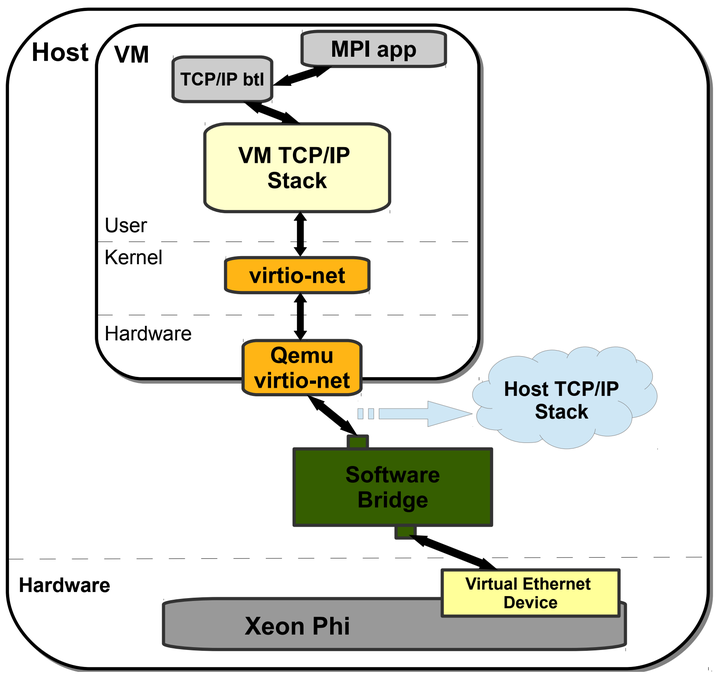
SCIF: Symmetric Communication Interface

- Transport layer between Host and Xeon Phi card(s)
- Directly exposes the DMA capabilities of Xeon Phi
- Uniform API across PCIe
- Socket-like API
- Provides both one-way and two-way communication semantics
- Many MPI implementations currently support SCIF transport

Virtio-SCIF

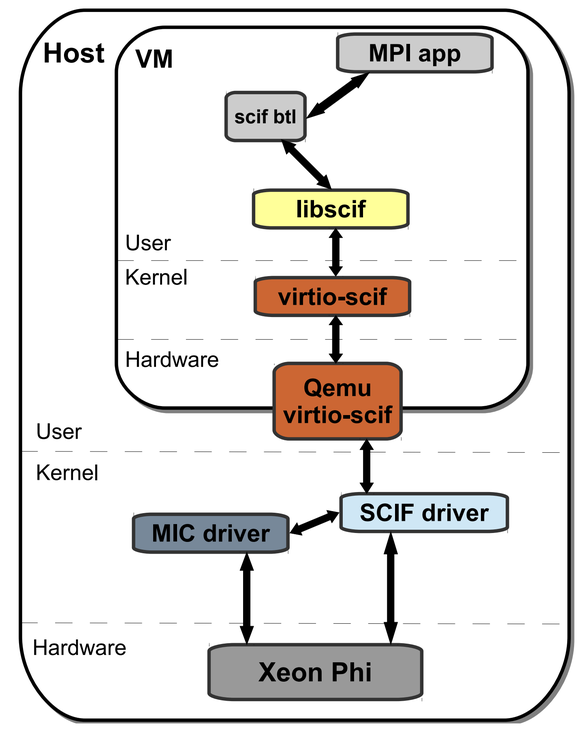
- *Main Idea*: virtualizing a generic transport between host and coprocessor devices
- Targeting cloud environments hosting HPC applications
- *Proof of concept*:
  - Virtualizing SCIF transport layer providing VMs with Xeon Phi capabilities
  - Virtual device implementation using Virtio specification
  - VMs can execute MPI over SCIF transparently with lower latency
- *Vision*:
  - Unify different coprocessors/accelerators transports
  - Provide coprocessor sharing to VMs by cloud providers
  - VMs can benefit and accelerate data-parallel applications

Current I/O Path



- Intel provides an implementation of a virtual ethernet device in its software stack
- VMs can execute MPI over TCP/IP
- Involves unnecessary TCP/IP traversal both in the VM as well as in the host

Proposed I/O Path



- Communication with SCIF semantics
- No application (source or binary) modification needed
- Avoids costly TCP/IP processing
- Low-latency, high-throughput communication
- Better scalability across multiple VMs sharing the same coprocessor card

References

[1] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. A. Pratt, and A. Warfield. Xen and the art of virtualization. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 164- 177, New York, NY, USA, 2003. ACM.

[2] F. Bellard. Qemu, a fast and portable dynamic translator. In *ATEC '05 Proceedings of the annual conference on USENIX Annual Technical Conference*, 2005.

[3] G. Giunta, R. Montella, G. Agrillo, and G. Coviello. A GPGPU Transparent Virtualization Component for High Performance Computing Clouds. In *Proceedings of the 16th International Euro-Par Conference on Parallel Processing: Part I*, 2010.

[4] Getting Kernel-Based Virtual Machine (KVM) to Work with Intel Xeon Phi Coprocessors. https://software.intel.com/en-us/articles/getting-kernel-based-virtual-machine-kvm-to-work-with-intel-xeon-phi-coprocessors.

[5] M. Gottschlag, M. Hillenbrand, J. Kehne, J. Stoess, and F. Bellosa. LoGV: Low-Overhead GPGPU Virtualization. In *Proceedings of the 4th International Workshop on Frontiers of Heterogeneous Computing*, Zhangjiajie, China, 2013

[6] Intel Xeon Phi Coprocessor System Software Developers Guide. https://software.intel.com/sites/default/files/managed/09/07/xeon-phi-coprocessor-system-software-developers-guide.pdf.

[7] S. Kato, M. McThrow, C. Maltzahn, and S. Brandt. Gdev: first-class GPU resource management in the operating system. In *Proceedings of the 2012 USENIX conference on Annual Technical Conference (USENIX ATC'12)*, USENIX Association, Berkeley, CA, USA.

[8] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liquori. kvm: the linux virtual machine monitor. In *Linux Symposium*, pages 225-230, Ottawa, Ontario, Canada, 2007.

[9] C. J. Rossbach, J. Currey, M. Silberstein, B. Ray, and E. Witchel. PTask: operating system abstractions to manage GPUs as compute devices. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles (SOSP '11)*, ACM, New York, NY, USA.

[10] R. Russel. virtio: Towards a de-facto standard for virtual i/o devices. In *ACM SIGOPS Operating Systems*, 2008.

[11] Y. Suzuki, S. Kato, H. Yamada, and K. Kono. GPUvm: why not virtualizing GPUs at the hypervisor?. In *Proceedings of the 2014 USENIX conference on USENIX Annual Technical Conference (USENIX ATC'14)*, USENIX Association, Berkeley, CA, USA.