Virtio-SCIF: Enabling Xeon Phi capabilities on Virtual Machines

PhD candidate: Stefanos Gerangelos
Advisor: Professor Nectarios Koziris
Computing Systems Laboratory,
National Technical University of Athens
{sgerag,nkoziris}@cslab.ece.ntua.gr

Introduction

In recent years heterogeneous systems equipped with accelerators or coprocessors are gaining popularity in the High-Performance Computing (HPC) community. At the same time, cloud computing infrastructures provide flexibility, dedicated execution, and isolation to a vast number of applications from diverse domains. Bridging the gap between virtualization, which is the key concept behind cloud-based deployments, and strict requirements of HPC applications is always a challenging problem.

In this work we propose Virtio-SCIF: a virtualization scheme for Intel Xeon Phi coprocessor systems. This approach targets cloud environments hosting HPC applications. Virtio-SCIF provides the offload capabilities of a single Xeon Phi coprocessor to many virtual machines residing in the same host system.

Background and Motivation

Intel Xeon Phi [10] coprocessor consists of an integrated, large set of cores (Intel Many Integrated Core (MIC) Architecture [3]) on a PCIe card. It provides x86 application compatibility, while running a version of Linux as a micro operating system. Xeon Phi's software stack includes a virtual network device driver, so as to provide the view of an independent node in a heterogeneous cluster environment. It also supports several popular programming models, including MPI and OpenMP.

Xeon Phi offers the flexibility for application developers to use it in different modes: native, offload and symmetric. In native mode a developer can build binaries solely for Xeon Phi platform and execute them in a coprocessor-only way. Alternatively, the coprocessor can be used as an accelerator by offloading certain workloads, while the main flow of execution is taking place on the host CPU. Finally, the symmetric model of execution allows both the host CPUs and the coprocessors to be involved in the execution of MPI processes and the corresponding MPI communication.

Intel has developed SCIF (Symmetric Communication Interface), a generic communication channel between Xeon Phi coprocessors and host CPUs in a heterogeneous computing environment. It provides communication capabilities within a single platform through a uniform sockets-like API for communicating across PCIe system bus. SCIF exposes the DMA capabilities of Xeon Phi coprocessor and can also be used to map host or Xeon Phi memory into the address space of a process running on either of them. Many MPI implementations (Open MPI [8], MPICH [6], MVAPICH [7], Intel MPI [4]) currently support SCIF network communication by providing the respective SCIF transport layer.

At the same time, extensive research over the last decade has shown that acceptable I/O virtualization performance can be achieved by utilizing special techniques. Each kind of device or interconnect being virtualized has its own characteristics and imposes different virtualization semantics. To our knowledge, the only currently available solution, in order to provide access to a single Xeon Phi device in a virtualized environment, is to use direct device assignment through PCIe passthrough [2]. Following this approach, however, results in exclusive access to Xeon Phi by a single virtual machine. Thus, PCIe passthrough offers near-native performance at the cost of the inability to share a coprocessor to many virtual machines.

In order to overcome this effect, we introduce Virtio-SCIF, a paravirtualized approach for Xeon Phi's SCIF transport layer. By using Virtio-SCIF, a user of a virtual machine can execute applications in native/offload mode and also launch MPI processes both on host CPUs and Xeon Phi coprocessors in symmetric mode, without the network stack overhead. Currently, MPI communication between processes running on Xeon Phi and processes running on virtualized CPUs is accomplished by using MPI's TCP/IP transport layer. This has the inevitable effect of unnecessarily traversal of the TCP/IP net-

work stack both on guest virtual machine as well as the host one, which has an impact on performance. Virtio-SCIF eliminates this overhead by using SCIF semantics and redirecting the respective calls to the host, which forwards them to the physical Xeon Phi device.

Virtio-SCIF Architecture

Virtio-SCIF can be utilized in KVM [5] virtualized environments on physical nodes equipped with one or more Intel Xeon Phi coprocessors. It consists of a virtio [9] guest device driver and the respective device implementation in QEMU [1] host userspace. Virtio-SCIF provides transparency at the application and library level. As a result existing Xeon Phi-aware applications can be used without any source code or binary modifications.

Virtio-SCIF proposed software architecture is depicted in Figure 1. The proposed changes in software are represented with colored boxes, while the grey boxes represent unmodified software components.

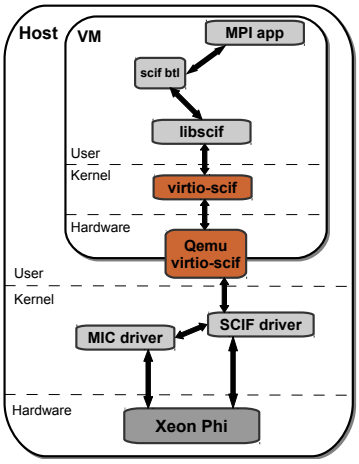

Figure 1: Virtio-SCIF Proposed Architecture

Figure 1 presents an example of an MPI application running in a virtual machine. A symmetric model of execution is supposed by using SCIF as MPI *Byte Transfer Layer* (BTL), although Virtio-SCIF can be used in native and offload context as well. In this example an MPI process is launched inside the virtual machine and another MPI process is launched on Xeon Phi coprocessor (the latter is not shown in the figure due to lack of space). For the initial spawn of processes, TCP/IP can be used, since this procedure is not in the critical path and it is not expected to affect performance.

During their lifetime, MPI processes need to communicate by exchanging messages. SCIF BTL translates MPI communication calls to the relevant SCIF calls with the use of *libscif* library. The library performs system calls to a character device, namely `/dev/mic/scif`. Virtio-scif driver (frontend) intercepts these calls and forwards them to the respective QEMU virtual device (backend). It accomplish this by writing requests to a *virtqueue*, which is the communication data structure between frontend and backend, and notifying QEMU. The notification handler of QEMU reads these requests and delivers them to the QEMU Virtio-SCIF component, which performs the actual system calls on behalf of the requesting process. Finally, the results of these calls are sent back to the process following the reverse path and notifying guest side by injecting virtual interrupts.

Summary

Virtio-SCIF allows virtual machines to share the capabilities of Intel Xeon Phi coprocessors in heterogeneous cloud environments. It consists of a backend/frontend virtual device and offers application transparency. Virtio-SCIF supports native, offload and symmetric model of execution on Xeon Phi systems. It is expected to achieve optimized performance compared to MPI applications communicating over TCP/IP, due to the elimination of processing overhead of network stack.

References

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

[2] 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`.

[3] Intel Many Integrated Core (MIC) Architecture. `http://www.intel.com/content/www/us/en/architecture-and-technology/many-integrated-core/intel-many-integrated-core-architecture.html`.

[4] Intel MPI. `https://software.intel.com/en-us/intel-mpi-library`.

[5] 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.

[6] MPICH. `http://www.mpich.org`.

[7] MVAPICH. `http://mvapich.cse.ohio-state.edu`.

[8] Open MPI. `http://www.open-mpi.org`.

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

[10] 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`.