2010 18th IEEE Symposium on High Performance Interconnects

Multi-Root Share of Single-Root I/O Virtualization (SR-IOV) Compliant PCI Express Device

Jun Suzuki Yoichi Hidaka† Junichi Higuchi

Teruyuki Baba Nobuharu Kami Takashi Yoshikawa

System Platforms Research Laboratories, NEC Corporation
Kawasaki, Japan
†IP Network Division, NEC Corporation
Abiko, Japan
{j-suzuki@ax, y-hidaka@bp, j-higuchi@ax, t-baba@ax, n-kami@ak, yoshikawa@cd}.jp.nec.com

***Abstract*—We have achieved sharing a single-root I/O virtualization (SR-IOV) compliant PCI Express (PCIe) I/O device among multiple computers. A device share not only inside a single computer among virtual machines, but also among multiple computers attracts a great interest because it provides efficient utilization of computer resources. Because PCIe is originally a single-root system, realizing multi-root I/O virtualization is much more difficult than SR-IOV. We allocate virtual instances called VF of an SR-IOV-compliant I/O device to an individual computer by virtualizing IOV configuration and translating memory address of VF to that of the allocated computer. With the FPGA implementation, we have achieved sharing a commercially-available network interface card among three computers without modification in OS/driver and device itself. In addition, the performance reaches 99% of the device in the best case whereas the implementation is in the early stage, indicating this method provides not only MR sharing but high performance at the same time.**

Keywords-virtualization; resource share; Ethernet; I/O device; PCI; SR-IOV

I. INTRODUCTION

Virtualization of computing resources is one of the key technologies for efficiently using computing platforms by sharing them among different system entities. Input/output (I/O) devices are one of the main factors that constitute computing platforms.

Recent advances in virtual machines (VMs) enable us to share computer I/O devices with less performance degradation and common interfaces [1-5]. Furthermore, the advent of an input/output memory management unit (IOMMU) [6] and PCI Express (PCIe) single-root I/O virtualization (SR-IOV) [7] is expected to further reduce still remaining performance degradation for I/O virtualization, because they directly assign hardware resource of I/O devices to a VM [8-10].

The above-mentioned technologies were developed for sharing I/O devices among VMs inside a single computer. In addition, sharing I/O devices among different computers provides further efficient I/O device use. Previously proposed methods, which have reported sharing I/O devices among multiple computers, include one with a device controller [11], and another with PCIe multi-root I/O virtualization (MR-IOV) [12].

In the device controller method, a computer accommodates I/O devices and provides I/O services to other computers connected to it by a network. The method provides common I/O interfaces and secures a computer platform from device-driver bugs by encapsulating it into a device controller. It enables device sharing without change to the device and its driver. However, both the communication process between a device controller and a client computer, and software arbitration process for I/O requests at the device controller lowers the shared I/O performance. The study in [11] proposes interconnecting a client computer and a device controller using Infiniband and implementing special software stacks to provide a high-performance I/O system.

MR-IOV was recently standardized for I/O sharing among multiple computers. It extends the conventional specification of PCIe. The sharing efficiency of I/O devices by MR-IOV is expected to be high, since it processes I/O data traffic by hardware. However, its I/O system becomes complex because each I/O device must adapt to the specification of MR-IOV where the tree topology of an I/O fabric of each computer must be individually managed over the PCIe network interconnecting computers and I/O devices.

In this paper, we realize simultaneous sharing of an SR-IOV-compliant PCIe I/O device among multiple computers. We follow the approach that does not alter a device and its driver used inside a single computer, to perform device sharing among multiple computers. For interconnecting computers and I/O devices, we use our Ethernet-based I/O interconnection technology, ExpEther (Express Ether), which we previously reported [13]. ExpEther allows multiple computers and I/O devices to be connected using a standard Ethernet. It constitutes a PCIe tree of individual computer over an Ethernet. Its grouping mechanism enables non-IOV I/O devices to be allocated to a certain computer and used without modification to the device and its driver. The

978-0-7695-4208-9/10 $26.00 © 2010 IEEE
DOI 10.1109/HOTI.2010.21

allocation of an I/O device to a computer can be altered by changing the computer-I/O grouping. The use of ExpEther and an SR-IOV-compliant device realizes simultaneous I/O device sharing among multiple computers in a simple way without modification to an I/O device. The method is so simple that it is implemented in a field programmable gate array (FPGA) and realizes high-speed device sharing.

The rest of the paper is organized as follows. We discuss our design goals for I/O device sharing in Section 2. In Section 3, we describe our proposed method and its architecture. In Section 4, we show the implementation of our prototype, and in Section 5, we present the evaluation results. We conclude our study in Section 6.

II. Design Goals

When designing the method for sharing an I/O device, we set the following design goals:

- I/O device should be shared among multiple computers without modification to the device and its driver which follow the SR-IOV specification.
- Sharing mechanism should be simple and be implemented using hardware to provide high-performance device sharing.
- Shared resources of an I/O device should be flexibly allocated to a desired computer using a hot-plug and remove mechanism.

III. Architecture

In the PCIe technology, an I/O device is accommodated in a single computer and used by a single entity. A device is connected to a tree topology of PCIe bus. Its root is within an I/O controller chipset of a CPU and leaves are I/O devices.

Our previous proposal of ExpEther connects multiple computers and multiple I/O devices using a standard Ethernet. The multi-computer topology is enabled by transporting transaction layer packets (TLPs), which are PCIe packets, using an Ethernet frames by encapsulation. A non-IOV I/O device is allocated to a certain computer with the grouping method. The allocation of an I/O device to a computer can be flexibly altered by changing the computer-I/O grouping.

The advent of SR-IOV equips an I/O device with multiple virtual instances which are respectively allocated to VMs. However, the device is supposed to use inside a single computer. In our proposal, we allocate these virtual instances to different computers by use of ExpEther and perform device sharing among multiple computers without modification to the driver and device itself.

In subsection 3.1, we describe how ExpEther interconnects multiple computers and I/O devices, and enables non-IOV I/O devices to be allocated to a certain computer using a grouping method. It constitutes a virtual PCIe switch over an Ethernet. In subsection 3.2, we explain an SR-IOV-compliant I/O device. In subsection 3.3, we propose simultaneous I/O sharing by ExpEther.

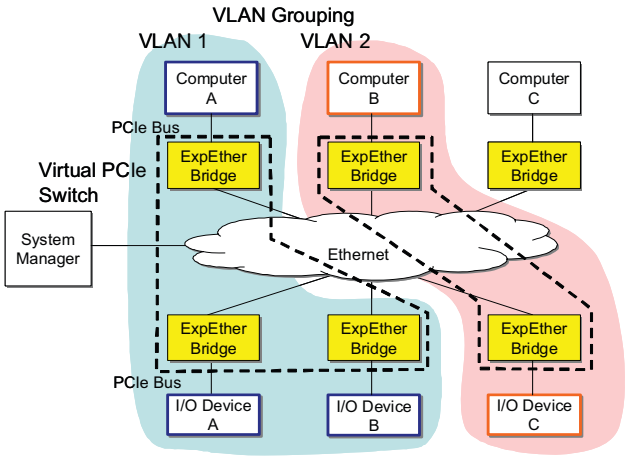


Figure 1. System with ExpEther. Virtual PCIe switch is configured within grouping VLAN. For example, computer A has 1:2 switch and computer B has one of 1:1.

A. *Virtual PCIe Switch with I/O Device Grouping*

ExpEther interconnects multiple computers and I/O devices using a standard Ethernet. Figure 1 shows a system with ExpEther. ExpEther bridges encapsulate TLPs into Ethernet frames and transport them between a computer-side and an I/O-side bridge over an Ethernet.

When multiple computers and I/O devices are connected using a network method, the PCIe address space of each computer needs to be separated to perform I/O processing correctly. In ExpEther, we use VLAN grouping for address separation. The individual computer to which a given I/O device is to be connected is determined by the VLAN. ExpEther bridges for I/O devices connected to the same computer are assigned a VLAN number corresponding to that computer.

An individual computer can use a grouped I/O device without modification to the device and its driver, when an Ethernet-connected I/O device is managed in the same way as other devices within its PCIe tree, and when it does not have to care the interconnecting Ethernet. We enable this usage by configuring a virtual PCIe switch over an Ethernet. The combination of ExpEther bridges in a computer side and I/O sides and an Ethernet works as a single virtual PCIe switch. It belongs to the PCIe tree of the computer. The virtualization is performed by emulating the response of a PCIe switch at ExpEther bridges to the system software. A PCIe switch is a PCIe component specified in the PCIe specification [14]. It divides a PCIe bus for connecting multiple I/O devices to a computer. The virtual PCIe switch enables each computer to extend its PCIe bus over an Ethernet. An Ethernet-connected I/O device can be used as a PCIe I/O device since the configured virtual PCIe switch over an Ethernet is compliant to the PCIe specification.

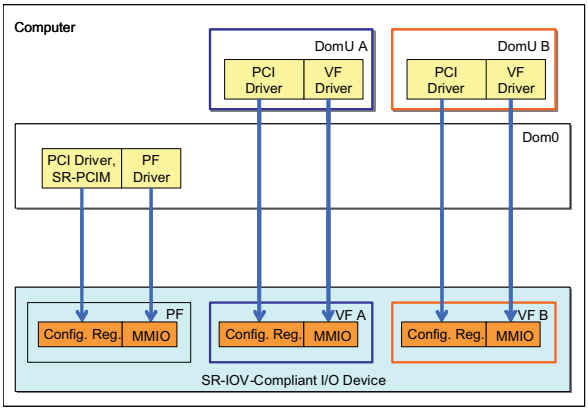


Figure 2. SR-IOV-compliant I/O device used in Xen environment.

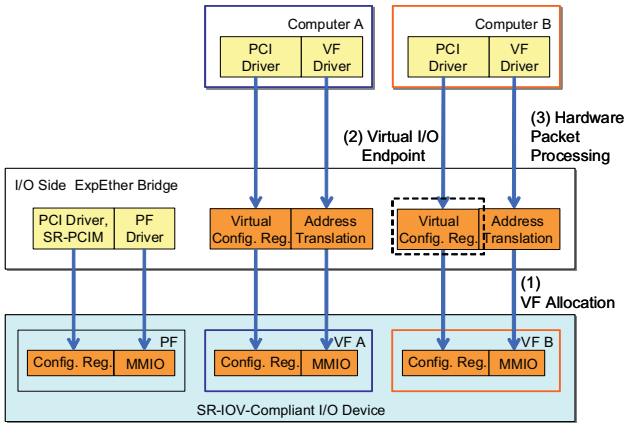


Figure 3. SR-IOV-compliant I/O device shared among multiple computers.

We also enable an I/O device to be hot-plugged and removed among grouping VLANs without a halt of computer operation. To change a connection for a given I/O device, a system manager alters the VLAN number assignment. To trigger PCIe-compliant hot-plug and removal of an I/O device, ExpEther bridges periodically broadcast keep-alive frames. When a frame broadcast by the ExpEther bridge of a newly attached I/O device is received by a computer-side ExpEther bridge, it interrupts a system and triggers a hot-plug process. When no frame has been received for a certain period of time, it starts a hot-remove process.

When TLPs are transported over an Ethernet using encapsulation, they should not be lost during transmission and its transmission latency should be minimized not to degrade I/O performance. We make ExpEther bridges perform lossless transmission and congestion control within an Ethernet protocol layer in an end-to-end (bridge-to-bridge) manner [15].

By above mechanisms, a virtual PCIe switch within a grouping VLAN separates the address space of each computer, and extends a PCIe bus of each computer over an Ethernet. An Ethernet connected I/O device can be used without modification. An I/O device also can be hot-plugged and removed among different computers.

B. SR-IOV-Compliant I/O Device

In our proposal, we share an SR-IOV-compliant I/O device using ExpEther. In this subsection, we explain the architecture of an SR-IOV-compliant I/O device. An SR-IOV device has multiple virtual instances which are individually allocated to VMs. By assigning hardware instance to each VM, virtualization overhead of an I/O device can be suppressed.

Figure 2 shows an SR-IOV-compliant I/O device used inside a single computer with a Xen VM environment. An SR-IOV-compliant I/O device is shared among two VMs (DomU A and B). The I/O device has virtual instances called virtual functions (VFs) and they are respectively allocated to VMs. In a non-IOV environment, a privilege domain (Dom0) arbitrates I/O requests of DomUs using software. The software arbitration process is slow and it degrades the I/O performance. The PCIe SR-IOV was specified to suppress the software overhead using hardware. It directly accepts I/O requests from VMs and arbitrates accesses to the I/O resources inside the I/O device.

As shown in Figure 2, an SR-IOV-compliant I/O device has one physical function (PF) and the same number of VFs as VMs. The PF is the interface for basic configuration of an I/O device. Its configuration register is set by a PCI driver and single-root PCI manager (SR-PCIM) in a Dom0. The memory mapped region for the PF, which designated as memory-mapped I/O (MMIO) in Figure 2, is accessed by a PF driver. VFs are individually assigned to VMs and accept their I/O requests. Its configuration register is set by a PCI driver in a DomU, while its region for MMIO is accessed by a VF driver. Each VF has its own resources for direct memory access (DMA) and interrupt. The combination of an SR-IOV-compliant I/O device and IOMMU at a host bridge provides DMA access between the I/O device and the memory region allocated to the VM without the intermediation of Dom0. By this configuration, an SR-IOV-compliant I/O device performs I/O processing of each VM through a high-speed communication path individually assigned to each VM. The I/O requests received from these interfaces are arbitrated using hardware inside an I/O device. This mechanism suppresses the software overhead regarding I/O resource virtualization and provides fast I/O sharing among multiple VMs.

C. Multi-Root Share of SR-IOV-Compliant I/O Device

Our proposal shares an SR-IOV-compliant I/O device among multiple computers. However, an SR-IOV-compliant device is designed for use not being shared among multiple computers but inside a single computer. In our method, we allocate virtual instances of an SR-IOV-compliant I/O device to each computer by a virtual PCIe switch of ExpEther.

Figure 3 shows an SR-IOV-compliant I/O device shared among multiple computers with the proposed method. We designed three technologies to meet our design goals which are mentioned in Section 2: (1) allocating I/O resources in the VF unit to an individual computer by address translation. This enables sharing of an I/O device, designed for use

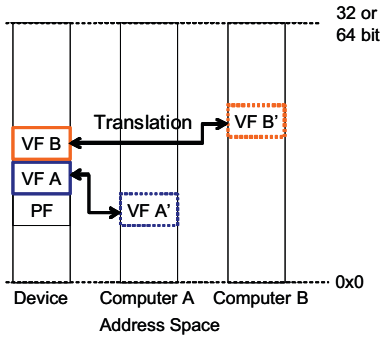


Figure 4. Allocating VF to individual computer by address translation. VF A is allocated to computer A as VF A' and VF B is to computer B as VF B'.

inside a single computer, to be shared among multiple computers and partitioning of I/O resources, (2) the virtual I/O endpoint (VE) mechanism enables flexible assignment of I/O resources and using them with an original device driver without modification, and (3) hardware packet processing enables efficient I/O resource sharing. In the following subsection, we describe each of these three technologies in detail.

1) Allocating VF to Individual Computer by Address Translation

We share an SR-IOV-compliant I/O device among multiple computers by allocating I/O resources in the VF unit to individual computers. Because each VF in an I/O device has an independent interface for receiving the control for I/O processing and resources for DMA and interrupt, the communication paths between computers and the I/O devices are partitioned. Also, because many SR-IOV-compliant I/O devices support the partition of its internal I/O resources, such as bandwidth in a network-interface card (NIC) among VFs, the internal resources of the I/O device can also be partitioned along VFs. We can partition I/O resources among computers sharing the device with this mechanism.

To share an I/O device in VF unit, we had to address the difference in memory mapping. Because an SR-IOV-compliant device is designed for use inside a single computer, it is memory-mapped to a memory space of a single computer. On the other hand, in our device-sharing environment, the I/O resources which correspond to each VF should be memory-mapped to the memory space to the each allocated computer. For this purpose, we first create a device address space to map the whole address region of the shared device. Then, we remap the I/O resources which correspond to each VF from the device address space to the address space of the individual computer. When the TLPs traverse the ExpEther bridge in an I/O side, their source and destination addresses are translated between the address in the address space of each computer sharing the device and the one in the device address space.

The mechanism of address translation in more detail is shown in Figure 4. VF A and B in an SR-IOV-compliant I/O device are respectively allocated to computer A and B. All of the I/O interfaces, i.e., the PF and VFs are memory-mapped to the device address space. The mapped regions of VF A and B are remapped as those of VF A' and B' which are respectively located in the address space of each computer. When a TLP is sent from computer A to VF A, the destination address is translated using the deviation of the base address of VFs A and A'. By way of contrast, when a TLP is sent from VF A to the computer A, the destination address is not translated but the source ID of the TLP is translated to that of the VE. The VE is an I/O endpoint that is detected by the computer A when it configures the assigned VF A. We describe VE in the next section. The reason we do not translates the destination address when TLPs are sent from VF A to computer A is based on the DMA mechanism. In DMA, a device driver in computer A sets DMA operation to VF A using the address space of computer A. As a result, the transmitted TLPs from VF A already have the address of computer A as their destination address.

With these mechanisms, we can memory-map a VF to its allocated computer. It enables an I/O device which is designed for use inside a single computer to be accessed from multiple computers. The allocation in the VF unit partitions I/O resources among computers sharing the device.

2) Virtual I/O Endpoint

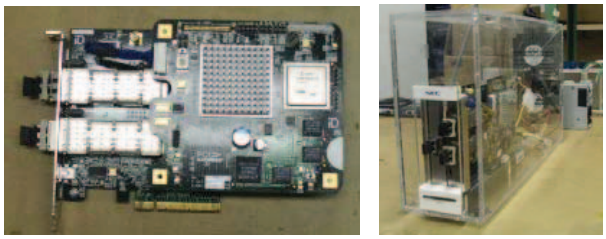
In I/O device sharing, we should be able to flexibly assign and use partitioned I/O resources of the device without interrupting other computers sharing the same device. We should also be able to use partitioned I/O resources with the device driver without software modification. However, an SR-IOV-compliant I/O device is designed for use inside a single computer, and it cannot accept separate controls from multiple computers.

To flexibly assign and use the partitioned I/O resources, and use them with its original device driver, we introduced a VE into our method. As shown in Figure 3, VEs are implemented in an ExpEther bridge in an I/O side and realized by a virtual configuration register. They individually correspond to the VFs in the I/O device. VE makes each computer recognize the assigned VF as a non-IOV I/O device. When a computer accesses its allocated I/O resources, an ExpEther bridge forwards TLPs for configuration to a VE and other TLPs to an assigned VF by address translation. Therefore, in the boot process of a computer, the computer recognizes assigned I/O resources as a non-IOV I/O device.

By the combination of the VE mechanism and hot-plug and remove functions of ExpEther, we can flexibly assign partitioned I/O resources to a computer as a non-IOV I/O device over an Ethernet The hot-plug and remove do not interrupt computers sharing the same device. The use of the VE also allows computers to perform exclusive access to the configuration register of an I/O device. Without VE, the access of each computer to the configuration register of a shared I/O device would interrupt the operation of other computers sharing the same device. Moreover, the assigned I/O resources can be used with its original device driver for a VF.

3) Hardware Packet Processing

The process for routing accesses from computers to a shared SR-IOV-compliant I/O device should be fast to reduce the sharing overhead. We implement device sharing



(a) (b)

Figure 5. Prototype for I/O device sharing. (a) ExpEther bridge in I/O device side with functions for I/O sharing. (b) I/O box.

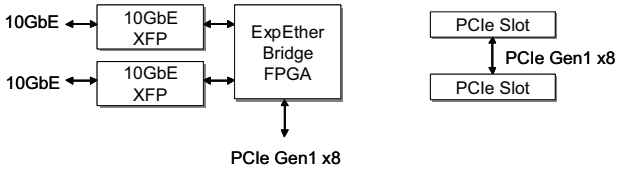


Figure 6. Block diagram of prototype. (a) ExpEther bridge in I/O device side with functions for I/O sharing. (b) I/O box.

mechanism using hardware, and perform high-speed and low-overhead device sharing. The hardware implementation is possible because our method is simple using multiple virtual instances of an SR-IOV-compliant I/O device to perform device share among multiple computers. The hardware TLP forwarding and address translation functions of an ExpEther bridge in an I/O-side allocates each virtual instance to each computer.

IV. Implementation

We have implemented the function for simultaneous I/O sharing in an ExpEther bridge in an I/O device side. Figure 5 (a) shows a diagram of our ExpEther bridge prototype and Figure 6 (a) is its block diagram. We have implemented the bridge using an FPGA, Xilinx Virtex5. The bridge prototype is a PCIe card and is inserted into an I/O box. The ExpEther bridge card has a PCIe Gen1 x8 interface. It also has two 10GbE Ethernet interfaces for high throughput reliable data transmission [16]. Figure 5 (b) shows the I/O box prototype and Figure 6 (b) is its block diagram. It accommodates an ExpEther bridge and an SR-IOV-compliant PCIe I/O device. The bridge and the I/O device are connected with a PCIe bus implemented at the board of the I/O box. With these configurations, an ExpEther bridge connects an SR-IOV-compliant I/O device to a standard Ethernet.

V. Experimental Evaluation

We performed the sharing of a 10-GbE NIC among multiple computers using our FPGA prototype. Note that the NIC was commercially available SR-IOV NIC from Exar without modification to the device and its driver. We first showed that a computer recognizes an allocated VF of an SR-IOV-compliant NIC as a non-IOV I/O device, and is used with its unmodified device driver for the VF. Next, we performed simultaneous I/O sharing among multiple computers and measured its performance.

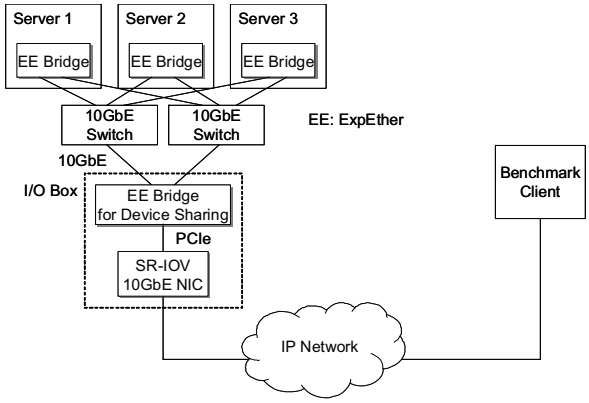


Figure 7. Experimental setup.

TABLE I. Specifications of Components Used in Experiment.

Component		Specification
OS		CentOS5.4(2.6.18-164.el5)
CPU / Memory	Server 1	Intel Core 2 Quad (2.83 GHz) / 8 GB
	Server 2	Intel Core 2 Quad (2.83 GHz) / 8 GB
	Server 3	Intel Core 2 Duo (3.16 GHz) / 8 GB
	Client	Intel Core 2 (2.66 GHz) / 2 GB

The experimental setup is shown in Figure 7. Three servers shared an SR-IOV-compliant NIC. ExpEther bridges in server sides were implemented as PCIe adapter cards and inserted into the I/O slots of the servers. ExpEther bridges in server sides and an I/O side were interconnected with a two standard 10GbE switches. These two switches formed two disjoint 10GbE networks to provide two paths. The prototype of our ExpEther bridges can load-balance traffic between the two networks and achieve 20 Gb/s [16]. We connected shared NIC to an IP network. A client for measuring a network performance was connected through the IP network. Other specifications of the components used in the experiment are listed in Table 1.

In the proposed method, a VE mechanism enables each computer to use an allocated VF as a non-IOV I/O device. Figure 8 shows the part of the output of "lspci" command in server 1, which shows its PCI tree. The Exar (previously Neterion) NIC was indicated as "Neterion Inc. X3100 Series", which was the same result under a non-IOV environment. The allocated VF was recognized as a non-IOV NIC, and configured and used with its original vendor-provided driver for the VF without any modification.

Next, we performed a network benchmark test using iperf [17] in an I/O sharing environment. Figure 9 shows the measured network bandwidth when the NIC was shared with up to three servers. In the "send" case, the client performed the iperf evaluation to the tested servers. On the other hand, in the "receive" case, the servers simultaneously performed the benchmark to the client. The aggregated bandwidths when the NIC was shared among one, two, and three servers were 8.9, 9.9, and 9.9 Gb/s in the send case, and 7.7, 6.1, and

```
[root@**** ~]# lspci -vt
-[0000:00]-+-00.0  Intel Corporation 82X38/X48 Express DRAM
                   Controller
           +-01.0-[0000:01]---00.0  Neterion Inc. X3100 Series 10
                                    Gigabit Ethernet PCIe
           +-06.0-[0000:02]--+-00.0  Intel Corporation 82571EB
                             |                Gigabit Ethernet Controller
           |                 \-00.1  Intel Corporation 82571EB
                                      Gigabit Ethernet Controller
           +-19.0  Intel Corporation 82566DM-2 Gigabit
                                     Network Connection
```

Figure 8. PCIe tree of server 1.

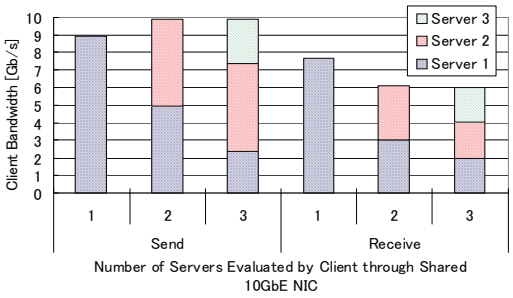


Figure 9. Measured network bandwidth in I/O sharing environment.

6.0 Gb/s in the receive case. The parameters of the maximum transmission unit (MTU) and the txqueuelen of each interface of each machine were set to 9000 and 12000, respectively, to maximize the achieved bandwidth in all the evaluations. The time for each measurement was 60s.

In the send case of three servers, the bandwidth resource was unequally allocated to each server. This unbalance is solved with the QoS control function of the NIC. At the time of evaluation, we had not yet implemented the function to control the QoS of the NIC to our ExpEther bridge.

In the receive case, there was overhead in the achieved bandwidth. To evaluate the cause of this performance overhead, we monitored traffic of TLPs between one of the three servers and the shared NIC. Figure 10 shows the monitored traffic. In the receive case, the NIC transmitted up to "N" read requests to the server for DMA memory read. Then, it waited to send the next request until it received the completion of its previous requests. The limitation of the number of read requests by which the NIC send to the server at one time became the bottleneck when we placed the server and the shared NIC apart using an Ethernet. The limitation of read requests is the implementation matter of the NIC. By increasing the number of read requests, we can solve the performance overhead seen in the receive case of Figure 9.

The shared NIC can also be used for the communication between the servers. A virtual Ethernet switch is implemented inside a NIC. Figure 11 shows the evaluated bandwidth when server 1 performed the benchmark with server 2 and 3 through the shared NIC. The cause of the bandwidth overhead seen in Figure 11 was the same for that of the receive case in Figure 9. With this inter-server communication function, we can reduce the total number of the NIC in the system for both inter-server and external communication.

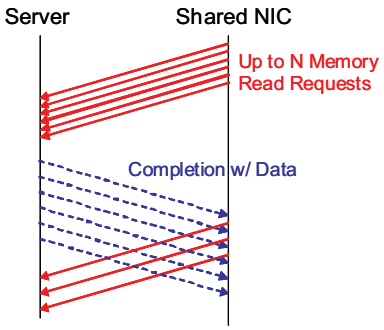


Figure 10. TLP traffic between server and shared NIC.

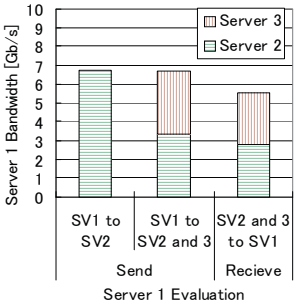


Figure 11. Measured network bandwidth of inter-server communication. SV is server.

These experimental results showed we can share a standard PCIe SR-IOV-compliant NIC among multiple computers without modification to the device and its driver. We achieved the total bandwidth of 9.9 Gb/s in a device sharing environment in the best case. The sharing efficiency was as high as 99%, even using our FPGA prototype. This shows our method realizes high-performance I/O device sharing by implementing its simple method using hardware. The proposed method is applicable to various kinds of SR-IOV-compliant I/O device.

VI. CONCLUSION

We have achieved the sharing of a PCIe SR-IOV-compliant I/O device among multiple computers. Our method connects computers and an I/O device using our Ethernet-based I/O interconnection technology, ExpEther. It enables an I/O device, which is designed for use inside a single computer, to be shared among multiple computers. By allocating an individual VF to each computer, an I/O device is shared without modification to the device and its driver. The proposed simple sharing mechanism enables its hardware implementation, and we can perform efficient and high-speed I/O sharing among multiple computers.

We have implemented three technologies in the ExpEther bridge in an I/O side for simultaneous I/O sharing: (1) the address translation mechanism enables allocating partitioned I/O resources in the VF unit to each computer; (2) the VE mechanism enables using allocated I/O resources as a non-

IOV I/O device with its device driver provided by the device's vendor; and (3) the hardware forwarding of TLPs enables efficient sharing of an I/O device. The experimental results showed that the sharing efficiency of a 10-GbE NIC was 99% in the best case, which is high enough to efficiently use I/O resources among multiple computers.

The proposed method enables the sharing of not only interface cards, which we have shown, but also various kinds of SR-IOV-compliant I/O device.

ACKNOWLEDGEMENT

We thank people who supported and collaborated on this work. Masahiko Takahashi gave us many technical comments and helpful suggestions. This project was led by Yukiko Yano and Toshiyuki Kanoh. We discussed together with the People in Exar, Leonid Grossman and Arpit Patel on the implementation matters on both their SR-IOV NIC "X3100" and ExpEther. Hiroki Iwasaki helped us a lot in the implementation of the FPGA prototype. Kota Marusero Fujita helped us evaluate the prototype.

A part of this work is supported by National Institute of Information and Communication Technology (NICT).

REFERENCES

[1] K. Fraser, S. Hand, R. Neugebauer, I. Pratt, A. Warfield, and M. Williamson, "Safe Hardware Access with the Xen Virtual Machine Monitor", 1st Workshop on Operating System and Architectural Support for the On-Demand IT Infrastructure (OASIS 2004), Oct. 2004.

[2] R. Russell "virtio: Towards a De-Facto Standard For Virtual I/O Devices", ACM SIGOPS Operating System Review, vol. 42, issue 5, pp95-103, Jun. 2008.

[3] J. R. Santos, Y. Turner, G. Janakiraman, and I. Pratt, "Bridging the Gap between Software and Hardware Techniques for I/O Virtualization", 2008 USENIX Annual Technical Conference, 2008.

[4] K. K. Ram, J. R. Santos, Y Turner, A. L. Cox and S. Rixner, "Achieving 10 Gb/s using Safe and Transparent Network Interface Virtualization", The 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environment (VEE 2009), March 2009.

[5] J. Liu and B. Abali, "Virtualization Polling Engine (VPE): Using Dedicated CPU Cores to Accelerate I/O Virtualization", 23rd International Conference on Supercomputing (ICS'09), June 2009.

[6] "Intel Virtualization Technology for Directed I/O Architecture Specification", Sep. 2008.

[7] "Single Root I/O Virtualization and Sharing Specification Revision 1.1", PCI-SIG, Jan. 2010.

[8] J. Liu, W. Huang, B. Abali, D. K. Panda, "High Performance VMM-Bypass I/O in Virtual Machines", 2006 USENIX Annual Technical Conference, 2006.

[9] H. Raj and K. Schwan, "High Performance and Scalable I/O Virtualization via Self-Virtualized Devices", IEEE International Symposium on High Performance Distributed Computing (HPDC'07), June 2007.

[10] P. Willmann, J. Shafer, D. Carr, A. Menon, S. Rixner, A. L. Cox, W. Zwaenepoel, "Concurrent Direct Network Access for Virtual Machine Monitors", IEEE 13th International Symposium on High-Performance Computer Architecture (HPCA-13), Feb. 2007.

[11] J. Satran, L. Shalev, M. B. Yehuda, Z. Machulsky, "Scalable I/O – a Well-Architected Way to Do Scalable, Secure and Virtualized I/O", USENIX First Workshop on I/O Virtualization (WIOV'08), Dec. 2008.

[12] "Multi-Root I/O Virtualization and Sharing Specification Revision 1.0", PCI-SIG, May 2008.

[13] J. Suzuki, Y. Hidaka, J. Higuchi, T. Yoshikawa, and A. Iwata, "ExpressEther – Ethernet-Based Virtualization Technology for Reconfigurable Hardware Platform", the 14th IEEE Symposium on High-Performance Interconnects (HOTI'06), pages 45-51, Aug. 2006.

[14] "PCI Express Base Specification Revision 2.1", PCI-SIG, March 2009.

[15] H. Shimonishi, J. Higuchi, T. Yoshikawa, and A. Iwata, "A Congestion Control Algorithm for Data Center Area Communications", 2008 International CQR Workshop, April 2008.

[16] N. Enomoto, H. Shimonishi, J. Higuchi, T. Yoshikawa, and A. Iwata, "High-speed, Short-latency Multipath Ethernet Transport for Interconnections", the 16th IEEE Symposium on High Performance Interconnects(HOTI'08), pages 75-84, Aug. 2008.

[17] iperf. http://sourceforge.net/projects/iperf/