High-Throughput Direct Data Transfer between PCIe SSDs

Jun Suzuki, Masato Yasuda, Masahiko Takahashi, Yoichi Hidaka †, Junichi Higuchi, Yoshikazu Watanabe, and Takashi Yoshikawa

System Platforms Research Laboratories, NEC Corporation, † Applied Appliance Division, NEC Corporation

{j-suzuki@ax, m-yasuda@ct, m-takahashi@ex, y-hidaka@bq, j-higuchi@ax, y-watanabe@fa, yoshikawa@cd}.jp.nec.com

Abstract

Data transfer between storage devices for data reallocation in a storage system and backing up data from a client to a network-attached storage are examples of data processing that transfers data between I/O devices without modifying data. In current systems, transferred data between I/O devices must be sent once to the main memory of the server hosting the I/O devices, but the bandwidth between the host and the I/O devices becomes a bottleneck for high-throughput data transfer. This paper proposes a method to transfer data directly between I/O devices. Evaluation using two PCI Express (PCIe) solid-state drives (SSDs) showed the proposed method increased the bandwidth of the data transfer to the maximum throughput of the SSD.

1. Introduction

Server platforms need to have various kinds of I/O devices to provide their respective services. Such I/O devices include network interface cards (NICs), PCI Express (PCIe) solid-state drives (SSDs), and storage host bus adaptors (HBAs). For this purpose, technologies have been introduced that connect servers and I/O devices using a network method and assign a server its necessary I/O devices by configuring the network connection [1, 2].

Many of the different kinds of data processing that use these I/O devices just transfer data between I/O devices in a block level without modifying them [3]. For example, when a network-attached storage system accepts writing to its disk from a client, the data received from its NIC are written to its disk. In another example, the Hadoop framework uses a method called pipelining to distribute data between HDFS nodes. Its node relays data by receiving them from its NIC and sending them to its NIC again.

However, even when transferred data are not modified, all of them must be sent once to the main memory of the server hosting the I/O devices. This leads to inefficient transfer, because all the data traverse the network link that connects a server to I/O devices, so its bandwidth becomes a bottleneck. In addition, the latency of data transfer accumulates when the data are copied within a server memory between kernel space and user space.

This paper proposes a method called *Direct Connect* that directly transfers data between PCIe-compliant I/O devices. These I/O devices are connected to a host server using PCIe over Ethernet technology implemented in a PCIe-to-Ethernet bridge LSI. Data are transferred by relaying direct memory accesses (DMAs) of a source device and a destination device by using an intermediate buffer in a PCIe-to-Ethernet bridge. Because the transferred data do not traverse the host server, low-latency and high-throughput transfer is performed.

The evaluation using two SSDs confirmed high-throughput data transfer without the bottleneck of the bandwidth between the server and the SSDs.

The proposed method is also designed so as to minimize alternation to current systems. I/O devices used are standard PCIe devices and are controlled by a host server using their device drivers.

2. Direct Connect Architecture and Implementation

Different I/O devices cannot directly communicate with each other or exchange data. Instead, they are controlled with driver software in a host server and can only transfer data by way of a server memory. Therefore, the bandwidth between a host server and I/O devices becomes a bottleneck of the data transfer. In our method, we directly transfer data using a memory in a PCIe-to-Ethernet bridge as an intermediate buffer so that the DMAs of a source device and a destination device can be relayed.

The architecture of the proposed Direct Connect is illustrated in Figure 1. I/O devices accommodated in an I/O resource box are connected to a host server through an Ethernet. A PCI-to-Ethernet bridge called an ExpEther bridge encapsulates a PCIe packet to an Ethernet frame and transports it between a server and an I/O device [1]. Because the PCIe bus of a server is virtually extended over the Ethernet, a server can use a connected I/O device as its local device. An ExpEther bridge also has a Direct Connect (DC) buffer internally to relay DMAs of I/O devices. This buffer is memory-mapped to the physical address of a server, and an I/O device can access it using its DMA.

The software part of Direct Connect in a host server consists of three components: DC commands, a DC operator, and a DC manager. DC commands are user-

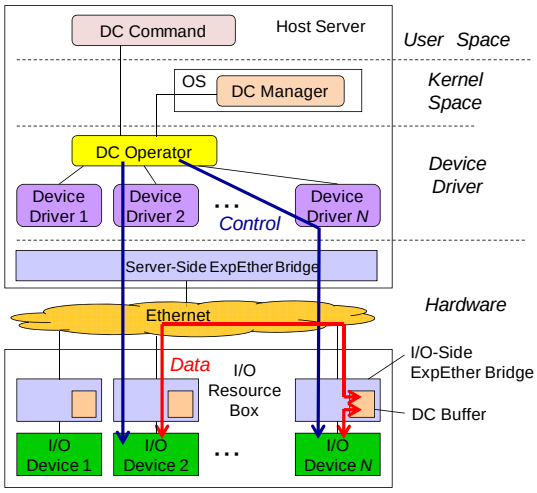

Figure 1: Proposed Direct Connect architecture.

level functions used to request data transfer between I/O devices. The actual transfer is performed by a DC operator that calls device drivers of I/O devices. When data are transferred between two devices, a DC operator calls the data read function of a source device driver and makes the device write its data to one of the DC buffers. Then, it calls the data write function of a destination device driver and makes the device read the data from the same DC buffer. In this way, two DMAs made by the source and destination devices are relayed and the data are directly transferred. A DC operator is also implemented as a device driver to minimize kernel modifications. When a DC operator calls a DC manager, the manager associates the names of I/O devices designated by DC commands to their data structures in kernel space. The sought data structure is used by a DC operator to call the functions of device drivers.

We implemented a prototype that directly transfers data between two SSDs. We used one PCIe SSD for a source device and another for a destination device, and a 10-Gb-Ethernet to connect an Intel Xeon server and the SSDs. The OS of the server was CentOS 5.5. A DC manager was implemented to the I/O stack of the kernel. Part of the device driver of the SSD was also modified to direct the DMA of the SSDs to a DC buffer. Because of the hardware limitation of the ExpEther bridge, we implemented a DC buffer on a separate field-programmable gate array (FPGA) card and inserted it into another slot of an I/O resource box.

3. Evaluation

The evaluation confirmed that our method enables high-throughput data transfer between two SSDs without the bottleneck caused by traversing a server.

First, we used a PCIe packet analyzer and confirmed that data were transferred not through the server but the DC buffer. Next, we measured the throughput of data

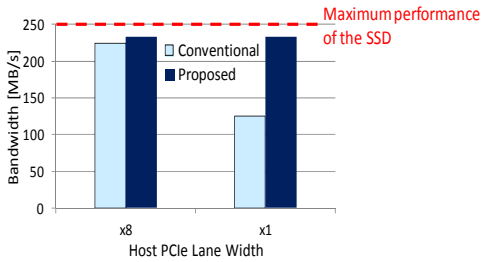

Figure 2: Measured bandwidth of data transfer.

transfer when 1-GB data were transferred between two SSDs. Figure 2 shows the measured bandwidth. For comparison with a conventional method that needs the transferred data to be sent once to the memory of a server, we also measured the throughput when transferred data traversed the server memory. The throughput was measured in two cases: the PCIe lane width of the server-side ExpEther bridge was set to x8 or x1. When the lane width was x8, the bandwidth between the server and the SSDs did not become a bottleneck for the data transfer. On the other hand, when the lane width was x1, the bandwidth became a bottleneck. The evaluation results show that the bandwidth of the proposed method was the same in both x8 and x1 cases and was increased by 86% compared with the conventional method in the x1 case. The obtained bandwidth was limited by the specification of the SSD. These results show that the proposed method enables high-throughput data transfer between I/O devices even if the bandwidth between a host server and I/O devices is as low as PCIe lane width of x1.

4. Future Work and Conclusion

We have applied our method to high-throughput direct data transfer between two SSDs. We will also apply it to the transfer between a NIC and an SSD. In addition, we will develop a method to handle storage metadata when its addressing data are directly transferred between two SSDs.

Acknowledgements

A part of this work was performed of the METI R&D Program supported by NEDO.

References

[1] J. Suzuki *et al.*, "ExpressEther – Ethernet-Based Virtualization Technology for Reconfigurable Hardware Platform," *IEEE Symp. on High-Performance Interconnects (HOTI'06)*, pp. 45-51, 2006.

[2] HP Virtual Connect Technology: http://h20195.www2.hp.com/v2/GetPDF.aspx/4AA0-5821ENW.pdf

[3] B. Rhoden *et al.*, "Improving Per-Node Efficiency in the Datacenter with New OS Abstractions," *Symp. on Cloud Computing (SOCC'11)*, 2011.