



Design and implementation of initial OpenSHMEM on PCIe NTB based cloud computing

Cheol Shim¹ · Kwang-ho Cha² · Min Choi¹

Received: 4 December 2016 / Revised: 1 September 2017 / Accepted: 3 January 2018 / Published online: 1 February 2018
© Springer Science+Business Media, LLC, part of Springer Nature 2018

Abstract

Cloud computing services are provided by key roles of data centers in which homogeneous and heterogeneous computation nodes are connected by high speed interconnection network. The rapid development of cloud-based services and applications has made data center networks more difficult. The PCI Express is a widely used system bus technology that connects processors and peripheral I/O devices. So, the PCI Express is regarded as a de-facto standard in system area interconnection network. It is currently validating the possibility of using PCI Express as a system interconnection network in areas such as high-performance computers and cluster/cloud computing. With the development of PCI Express non-transparent bridge (NTB) technology, the PCI Express has become available as a system interconnection network. NTB allows two PCI Express subsystems to be interconnected and, if necessary, isolated from each other. Partitioned global address space (PGAS) is one of the shared address space programming models. Due to the recent spread of multicore processors, PGAS has been attracting attention as a parallel computing framework. We make use of the PCI Express NTB to realize the PGAS shared address space model. In this paper, we designed and implemented the interconnection network using PCI Express x8 using a RDK, the PEX8749 based PCI Express evaluation board. We performed some Openshmem applications from Github to verify the accuracy of our initial OpenSHMEM API implementation.

Keywords PCI Express · Non-transparent bridge · Interconnection network · RDMA · One-sided communication

1 Introduction

Data centers have become increasingly part of in the cloud computing. The rapid development of cloud-based services and applications has made data center networks more difficult. Traditional electronic interconnect networks can hardly meet all requirements for bandwidth, device cost, and power dissipation. To solve this problem, a PCI Express interconnect network characterized by high-speed interconnect networks, low cost, and high bandwidth has been proposed as shown in Fig. 1. Recently, it has become a topic in the field of research. Nowadays, hundreds of applications have been implemented using high-speed networking and local networking. In these applications, we can

achieve more benefits of high-performance, low power and reduced costs when it is used in field of interconnection network.

As the computing technology and network technology development, the amount of data to be processed by processors was increased, resulting in high-performance computing. Recent high-performance computing system are based on a system interconnect technology such as Infiniband and Ethernet, and PCI Express.

PCI Express bus is originally a technology for the connection among processors, memory, and peripheral I/O devices. PCI Express has been developed from 2.5Gbps Gen1 specification to Gen3 specification technology. With the increase of speeds and performance needs, there were many requests to make use of bus bandwidth efficiently on system interconnection network [1,2].

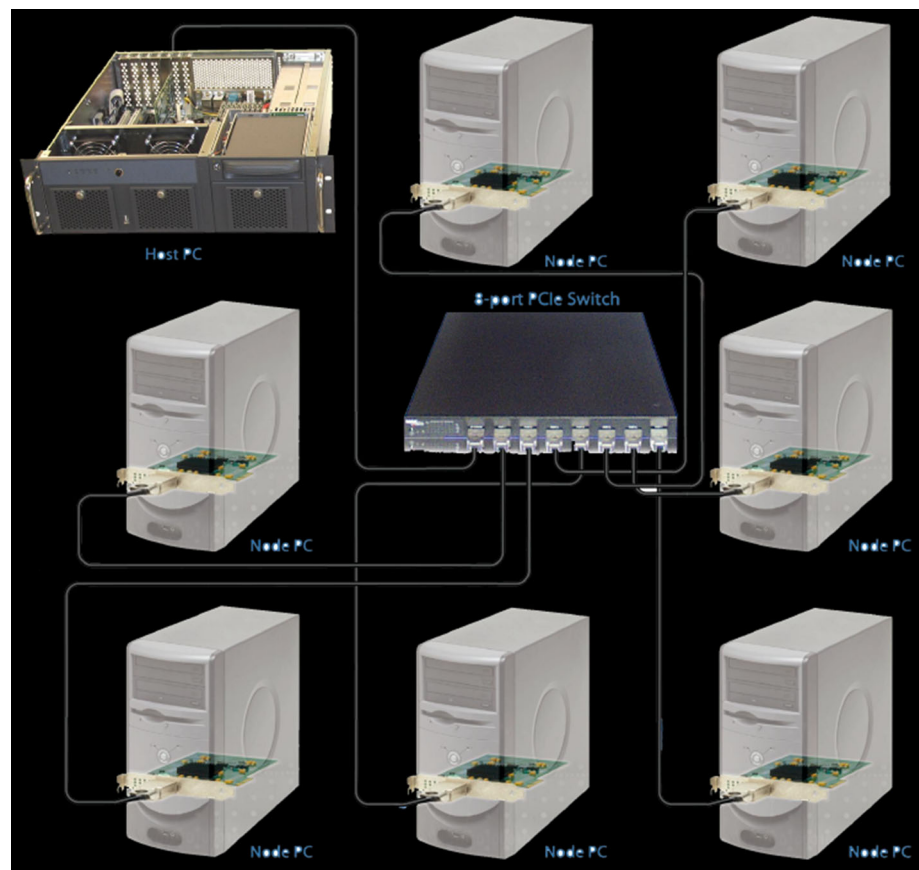
In field of high performance computing and cluster computing, interconnection network technology has been widely studied for a long time. Majority of interconnection network technologies are Infiniband and Ethernet. Heymian [3–5] use the PCI Express non-transparent bridge (NTB) to implement

✉ Min Choi
mchoi@cbnu.ac.kr

¹ Department of Information and Communication Engineering, Chungbuk National University, Cheongju 28644, Korea

² Center for Supercomputer Development, Korea Institute of Science and Technology Information, Daejeon, Korea

Fig. 1 Technology trends of cloud data center interconnection [1]



a multi-host system. The PCI Express NTB solves clock synchronization issues and provides isolation between hosts. RCT Magazine [6] implemented the DMA transfer with a contiguous buffer using the PCI Express bus [7–9]. This approach enables a DMA transfer for non-contiguous memory. This paper design and implemented PC cluster system based on PCI Express interconnection network technology. In this paper, we use the PCI Express non-transparent bridge to implement a DMA transfer of non-contiguous memory between two systems.

This paper is organized as follows: Sect. 1 provides introduction, In Sect. 2, we show our design and implementation of OpenSHMEM on PCI Express. In Sect. 3, we analyze and compare the existing and proposed scheme. Section 4 concludes this paper.

2 Initial design of OpenSHMEM on PCIe NTB

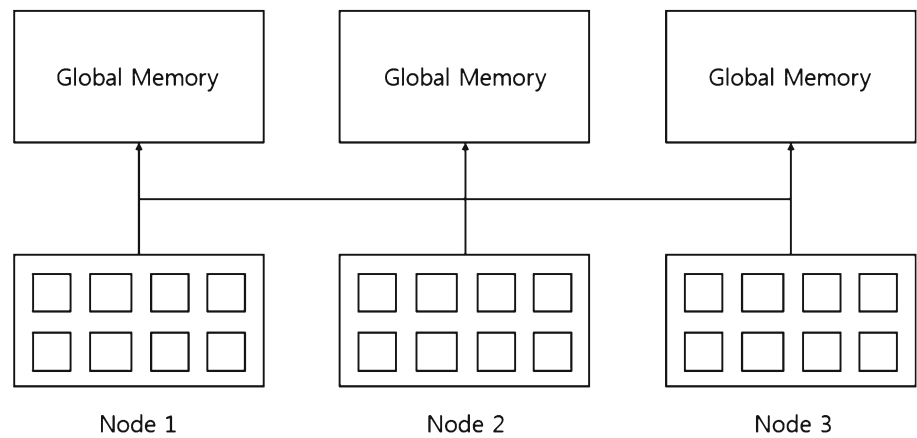
PCI Express is a further improved version than PCI bus technology developed by INTEL. PCI Ex-press is a technology by which can be connected with various peripherals or host, support high I/O scalability and high-performance, high-speed communication. PCI express non-transparent bridge is widely used in the multi-host and host-failover. Fig-

ure 2 shows the system architecture in which two hosts are connected through PCI Express non-transparent bridge [4,10].

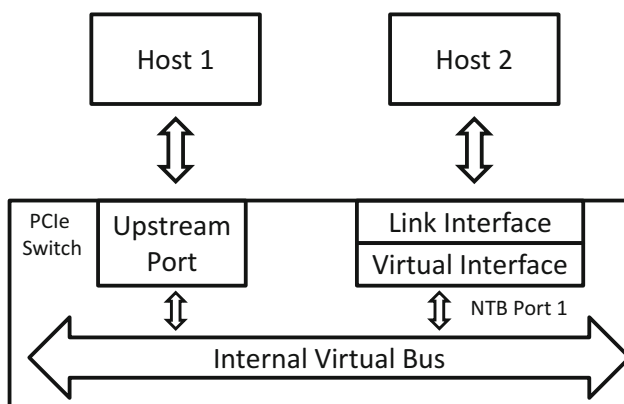
2.1 System architecture

PCI Express is transmitting and receiving data (full-duplex) through an independent link tied to an integrated line in one lanes of transmit and receive. PCI Express may be used by a plurality of lanes (x2, x4, x8, x16) to control the links from a single lane width (X1). PCI Express has the 8Gbps performance of a single-lane Gen3 data rate. Table 1 represents the data transfer rates depending on the generation and link width [6,11,12].

PCI Express non-transparent bridge is used for the mutual isolation in the PCI Express based inter-connection network [13]. NTB has integrated two conceptual interfaces(endpoints) for two roles in one port. We cannot see the fact that the two systems are connected in the BIOS emulation process due to the logically separated interface of PCI Express NTB. Figure 3 depicts an example in which two systems are connected through PCI Express based interconnection network [14].

Fig. 2 Multi-host system hierarchy**Table 1** Transfer speed of PCI Express

Generation	Lane (width)				
	X1	X2	X4	X8	X16
Gen1 (Gbps)	2.5	5	10	20	30
Gen2 (Gbps)	5	10	20	40	60
Gen3 (Gbps)	8	16	32	64	96

**Fig. 3** Interconnect system with non-transparent bridge

2.2 Address translation

In the left side of Fig. 4, Host A and B can communicate via logical endpoint interface of virtual side and link side through address translation. It is necessary to know the address of the external memory on the address translation in the host. There are some scratchpad registers that can be accessed from each of the interface. Two hosts may share data with each other via the scratchpad register. It can also cause an interrupt to the other party through the Doorbell register. The interconnection network determines where the packet is directed through the bus and device information field in the packet header.

The bus number and device number are different, so the interconnection network returns a proper ID depending on the

bus number and device number during address translation. In order to do this, there is an address-translation lookup table (LUT) to which we can register an ID, lookup the ID by the index. In the right side of the Fig. 4, each host is located behind its own NTB BAR(s). Each N host address range within the global space is also divided into Nx segments. The figure illustrates a host A slice of the address range, going to each Host. The host A A-LUT points to one segment within each address range, at a fixed host A offset [15,16].

Figure 5 shows the address translation process of a PCI Express packet [15]. There are also necessary to translate memory address on remote DMA transfer, because the destination address on target machine has totally different address management from that on source machine. We make use of direct address translation in that the offset address are maintained but only the base address is replaced by that of the target machine as shown in Fig. 5.

After the address translation in Figs. 4 and 5, we can communicate each other through Remote Direct Memory Access (RDMA) as shown in Fig. 6. The standard interface of RDMA is described in red colored in Fig. 6, for example, `shmem_int_put()`, `shmem_double_put()`, `shmem_int_get()`, and `shmem_double_get()`. After those APIs are called, the first task is attempt to allocate a physically contiguous, page-locked buffer which is safe for use with DMA operation. Another step is to map a previously allocated physical memory into user virtual space. Then, the memory allocation and mapping are done, so the block `DMATransfer()` function will be called and executed.

2.3 Symmetric heap

The symmetric heap of OpenSHMEM consists of a symmetric data object which is accessible from a remote location and a private data object which is accessible only to the local PE [17,18]. A symmetric data object is allocated to a symmetric heap area. The symmetric heap area are physically allocated to a different address space for each PE node. A symmetric

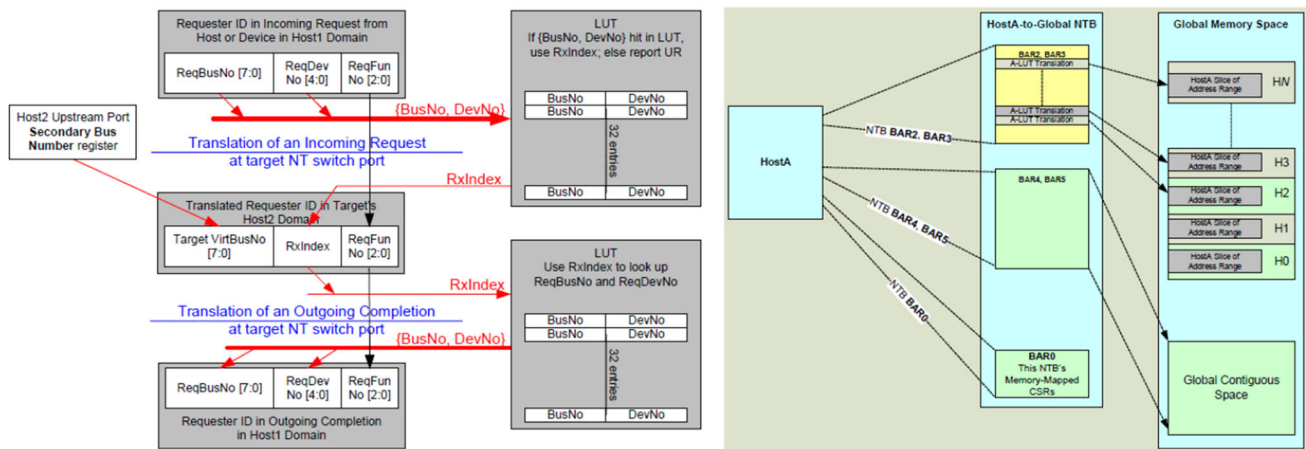


Fig. 4 Requester ID translation sequence on PCIe bus

Fig. 5 Memory address translation sequence

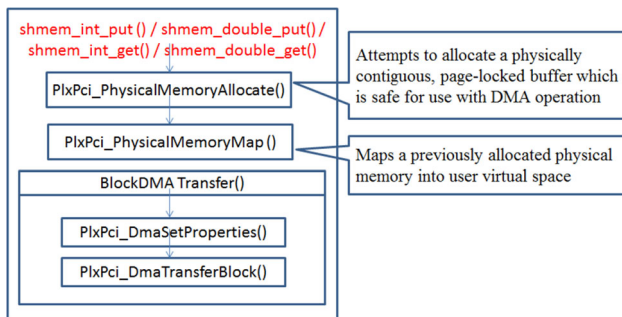
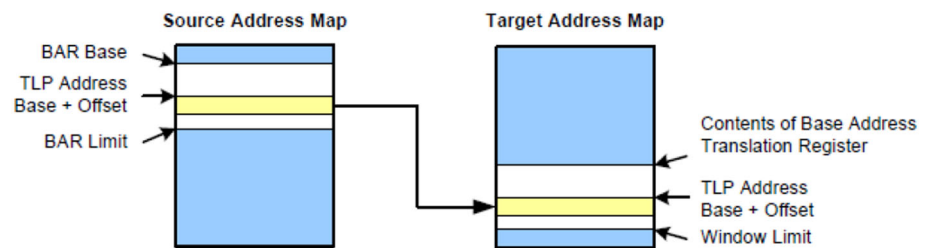


Fig. 6 Our RDMA implementation

data object is created with the same name/size/type through the memory allocation function. The `shmem_malloc` function assigns a common symmetric data object to each node by calling all PEs together. The offset in the symmetric heap is the same, even though the physical address of the symmetric data object assigned to each PE is different.

Fig. 7 Symmetric heap architecture

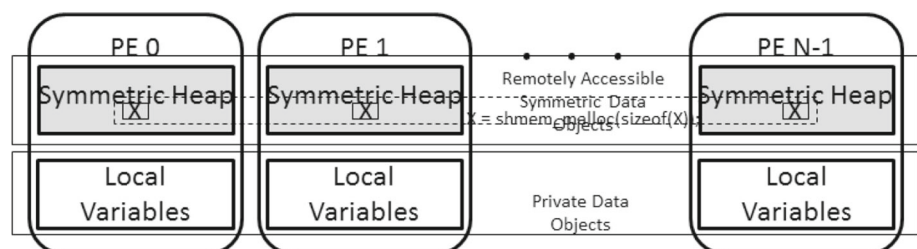


Figure 7 describes the design of memory allocation on OpenSHMEM and Fig. 8 shows the flow charts of the barrier implementation. Since the symmetric variables must be assigned to the symmetric heap, it is designed to check if the symmetric heap is already created first by calling a `malloc` function on each PE. If a symmetric heap does not exist, we create a new symmetric heap. After the allocation of the symmetric heap, each PE shares the address of the allocated symmetric heap to access the symmetric variable in the data transfer function. If a symmetric heap already exists, we check the remaining symmetric heap space to see if we can allocate additional symmetric variables.

If there is not enough memory left in the symmetric heap, a new symmetric heap is allocated. The symmetric variable assigned to each PE has the same offset in the symmetric heap of each PE. Therefore, symmetric variables are allocated in order from the start address of the symmetric heap to the size of the requested symmetric variable.

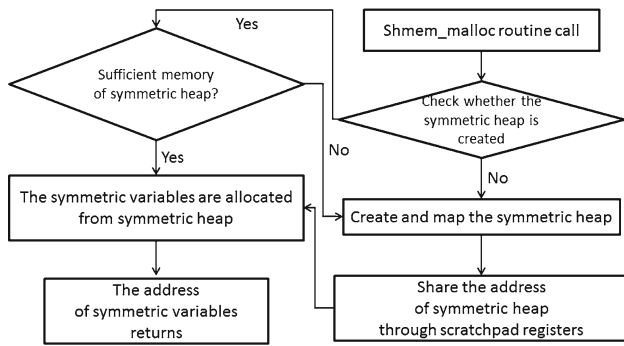
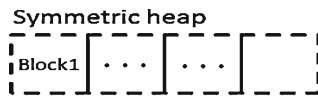
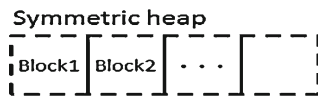


Fig. 8 Flow chart of barrier implementation

```
int Block1 = shmem_malloc(sizeof(Block1));
```



```
int Block2 = shmem_malloc(sizeof(Block2));
```



```
int BlockN = shmem_malloc(sizeof(BlockN));
```

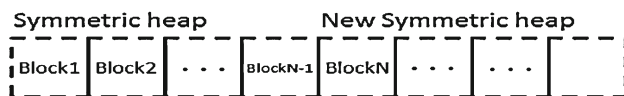


Fig. 9 Symmetric memory implementation for supporting large sized symmetric variable using small chunks

The Fig. 9 shows how symmetric variables are assigned to each PE when calling the malloc function. First, we call the malloc function to allocate the symmetric variable block1 (assuming that the malloc function was called first). Then, we allocate from the start address of the symmetric heap as much as the requested size from the memory of the symmetric heap. However, the malloc function is called to allocate the nth symmetric variable, but the memory of the original symmetric heap is exhausted and there is no remaining memory. We create a new unit size symmetric heap, and we allocate additional symmetric variables by taking up memory from the starting address of the newly created symmetric heap. This process is the same for all PEs because all PEs call shmem_malloc at the same time.

2.4 Synchronization primitive: barrier

A barrier is a type of synchronization method. As shown in Fig. 10, a barrier for a group of threads or processes in the source code means any thread/process must stop at this point and cannot proceed until all other threads/processes reach this barrier. Figure 10 is a conceptual representation of barrier. In

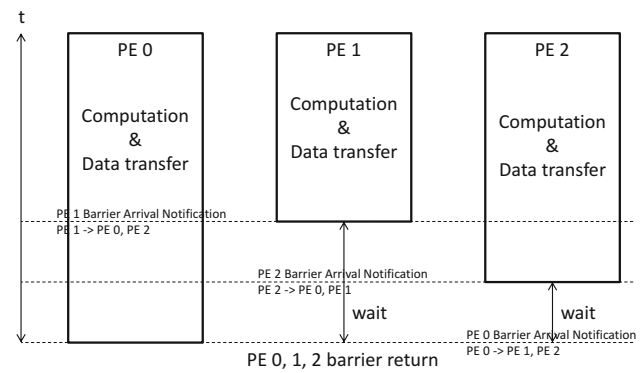


Fig. 10 Conceptual description of barrier

Table 2 Classical barrier implementation

Barrier implementation (ACM TOCS91')

1. Centralized barrier
2. Software combining tree barrier
3. Dissemination barrier (butterfly barrier)
4. Tournament barrier

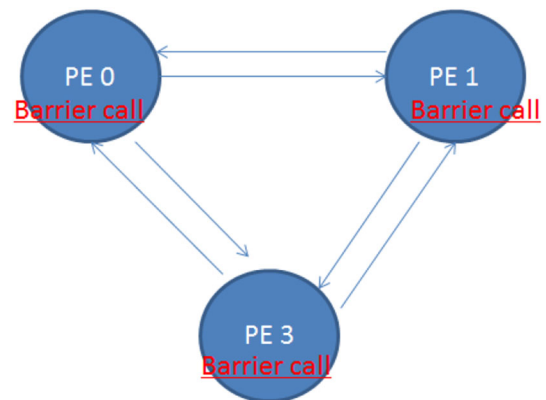


Fig. 11 Our implementation: the centralized barrier

this study, when the DMA transfer is terminated, it notifies other PEs that the barrier function has been performed at the PE and waits until other PEs perform the barrier call.

As shown above, each PE may have different execution time for computation and data transfer. PE 1 confirms that the data transfer is completed after the computation is completed, notifies PE 0 and PE 2 that the barrier has been called, and waits for the barrier arrival of PE 0 and PE 1. When all PE calls the barrier, the synchronization is completed and the process can proceed to the next phase.

In this study, one of the 4 methods, shown in Table 2, presented in the ACM TOCS paper [19] was used to implement the barrier. The Centralized Barrier method is the most typical barrier implementation method as shown in Fig. 11. When each PE (processing element) reaches the barrier, the central-

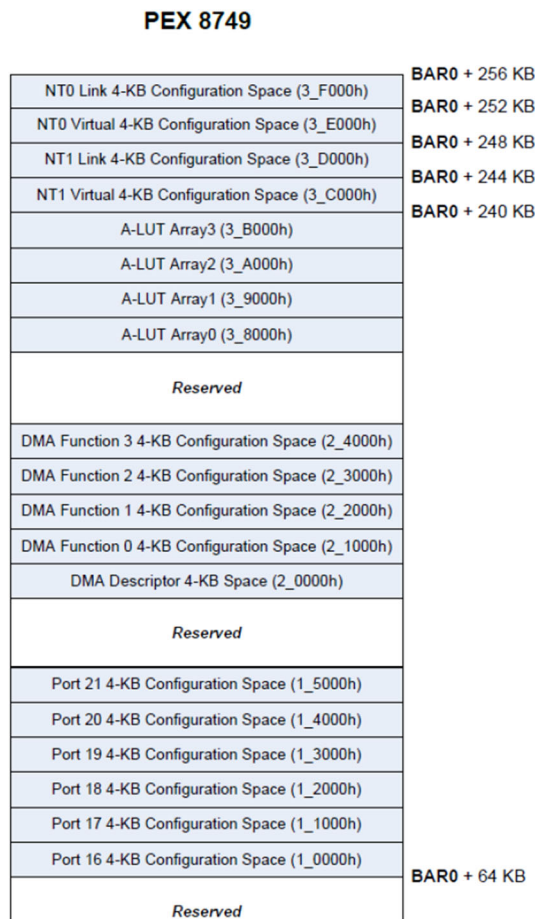


Fig. 12 Interrupt generation through the doorbell register

ized barrier sends an interrupt to all the PE nodes except for itself. Each node can proceed through the barrier if it receives an interrupt of the total number of nodes (N)-1.

To implement a barrier in a PCI Express-based interconnection network, the doorbell interrupt of the configuration space register in Fig. 12 should be used. Doorbell inter-

rupts enable interrupt transfer between each PE node. Before counting the number of doorbell interrupts, we first need to check that the DMA transfer has been completed by `PlxPci_BarrierDMANotificationWait`. Then, an interrupt signal is generated by writing to the doorbell IRQ register through the function `PlxPci_PlxMappedRegisterWrite`. The function `PlxPci_BarrierDBNotificationWait` is implemented to check the number of door bell interrupts received from other PEs.

We also implemented interrupt messaging mechanism using doorbell interrupt. To realize this functionality, we have to access the NT0 link side configuration space and NT0 virtual side configuration space as shown in Fig. 12. For sending an interrupt on link side, we make use of address 0x3FC5C. For sending an interrupt on virtual side, we make use of address 0x3EC4C.

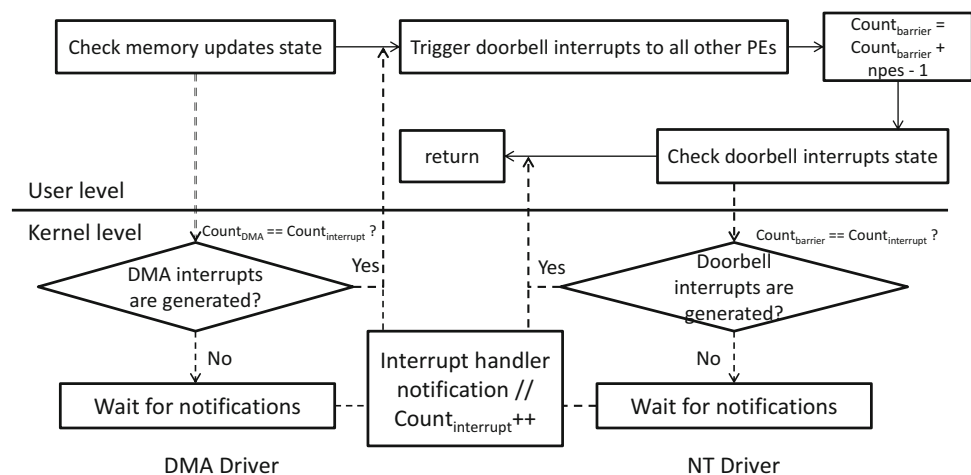
Figure 13 depicts an overall flow of barrier implementation. The `PlxPci_BarrierDMANotificationWait` function in Fig. 14 checks whether the DMA transfer is completed. Next, to count the number of doorbell interrupts, an interrupt is generated by controlling the doorbell IRQ register with the function `PlxPci_PlxMappedRegisterWrite`. `Plx_BarrierDBNotificationWait()` function calls `PlxIoMessage()`, the IOCTL function. The IOCTL function executes system call of operating system, especially `IOCTL_BARRIER_DB_NOTIFICATION_WAIT`.

The `PlxMappedRegisterRead()` function uses the device driver to read the value of a register at a specific address. And this function checks the `VIRTUAL_IRQ_SET` and `LINK_IRQ_SET` bits.

3 Experimental analysis

This section provides the design and implementation of evaluation system. Table 3 shows the experimental environment of the system. We make use of Core-i5 3.2 GHz, with 2 GB 1333 MHz DDR memory. The OS is Centos Linux 7

Fig. 13 Overall flow of barrier implementation



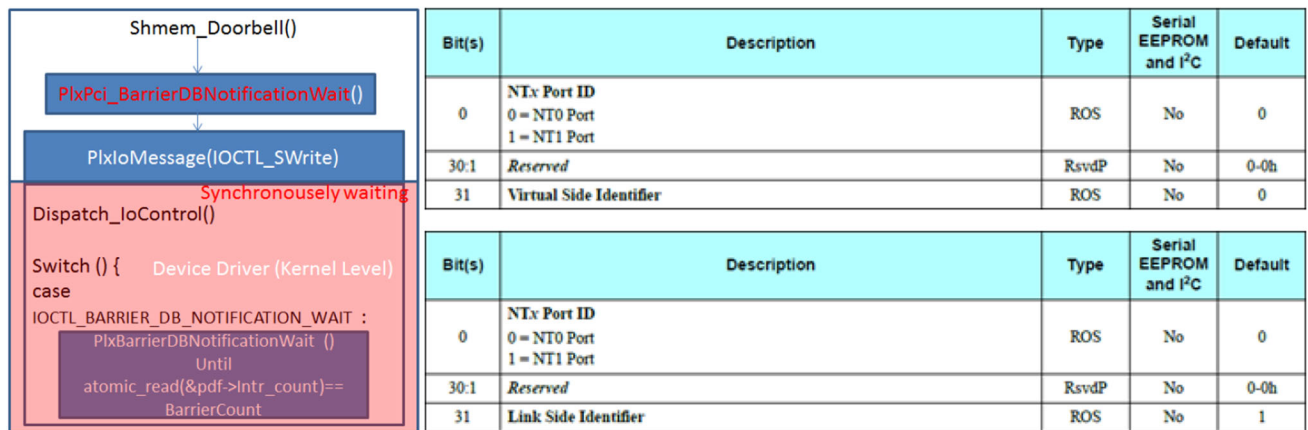


Fig. 14 PxlPci_BarrierDBNotificationWait() register

Table 3 Testbed specification

Host PC hardware environment	
M/B	GIGABYTE GA-H61M-DS2V
Processor	Intel® Core™ i5-3470 CPU @ 3.20GHz X 4
Memory	Samsung DDR3 1333MHz 2GB
O/S	Centos Linux 7 64bit Kernel : 3.10.0-327.el7
Interconnect hardware environment	
PCI Express NTB evaluation board	PLX PEX8749 RDK 48 Lane, 18 Port, PCI Express Gen 3 Switch
NIC Driver	PLX PEX8732 Cable Adapter X 4 PLX SDK Device Driver

Table 4 Execution command

```

gcc -c rotget.c
gcc -o rotget rotget.o ./Library/openshmem.a -lm -ldl

```

pile application source code such as `rotget.c` using `gcc` as shown in Table 4. We run the same compiled binaries on each PE through the NFC shared file system.

This section shows the performance of PCI Express link. These days we usually utilize the PCI Express generation 3, so we can get link performance up to 8 Gbit/sec, theoretically. But, the actual performance depends on variable situation such as link status, protocol overhead and so on.

To connect a host PC to PEX8749 evaluation board, we need a PEX8732 chipset based network interface card for each side. Since the evaluation host supports up to PCI Express generation 2 specification, we make use of generation 2 specification both for host and PCI Express NTB evaluation board. Evaluation software runs on Centos 7.0 linux operating system (kernel 3.10.0). For device control, PEX8749 device driver was installed as shown in Fig. 16.

First, we show the performance difference between non-DMA transfer using ordinary `memcpy` and PCI Express

64bit, kernel 3.10.0. The interconnection network is based on PEX8749 Chipset of PLX Technology.

Figure 15 depicts the experimental environment. Each PE make use of the shared file system by network file system (NFS) in order to see the same files and directories. We com-



Fig. 15 Experimental environment

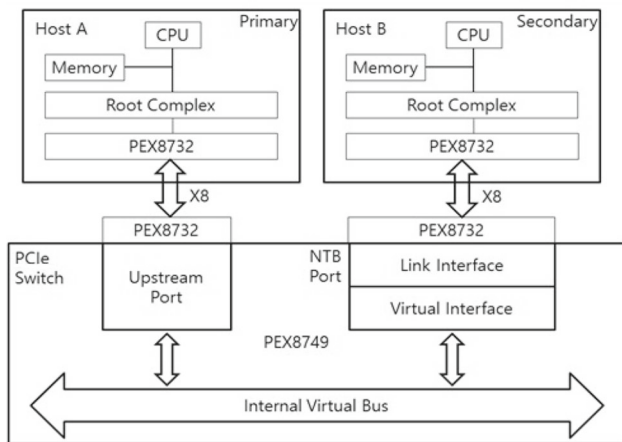


Fig. 16 Implemented interconnection network system

RDMA transfer and we also evaluate the performance improvement between Block RDMA transfer and scatter-gather RDMA.

In each experiment, each host transmits the data through buffer located in BAR (base address register). The actual physical address, the virtual address in kernel, and buffer size are send and received to each other via scratchpad register in host. In this case, the host buffer were assigned to the same size. We transmit the buffered data to the target host after address translation by using the BAR register.

We limit the buffer size to a maximum of 1 MB. In this experiment, we transfer the data to Host A from Host B by using one DMA channel. We measured the transmission rate with respect to the size and number of buffers. When an interrupt occurs at the end of each transfer, we measured the average data transfer rate as a way to send data back and forth.

The left side of Fig. 17 depicts the evaluation between non-DMA transfer and block RDMA transfer and between Block DMA and SGDMA. The x axis shows the size of transfer data and the y axis represents the transfer rate as MB/s,

commonly in both graphs. When the size of the buffer is small, memcpy and DMA transfer shows low performance at the same time. The larger the size of the buffer gets better performance in DMA transfer. This is because the data transfer at a time to increase. When the data size is below 2 kB, memcpy showed higher transmission rate than DMA transfer. Even if we increase the buffer size in memcpy, it tends to be satisfied at the transfer rate of up to 150 MB/s. In more than 4 kB buffer size, we got a relatively high performance of up to about 500 MB/s in DMA transfer.

The SGDMA transfer descriptor has information which is required for DMA such as source address, destination address, and byte count. This descriptor enables us to transmit multiple DMA buffers or discontinuous address spaces. The right side of Fig. 17 shows the performance of SGDMA transfer. The right side of Fig. 17 shows the evaluation between block RDMA transfer and scatter-gather RDMA transfer. The x axis shows the size of transfer data and the y axis represents the transfer rate as MB/s, commonly in both graphs. We used the 8, 32, 128, 256 descriptors for SGDMA transfer. We specify the data size for each descriptor to a value obtained by dividing evenly the same size to the buffer size. The smaller the size of the buffer, the more the number of descriptor, we got lower the transmission rate of the SG DMA. As a result of execution with 8 descriptors, we got up to 520 MB/s speed. This is slightly higher than the performance Block DMA. Evaluation result shows that the non-DMA transfer performance is 150 MB/s and the Block RDMA transfer is improved to 500 MB/s. Finally, scatter-gather DMA performance is shown up to 520 MB/s. In this paper, it delivers high-performance computing and widely used Interconnect Technology of the Non-Transparent PCI Express Switch and connect two PC interconnect system using cluster computing Bridging that the DMA transfer performance were analyzed between two PC.

Non-DMA (memcpy) and DMA Block exhibited a maximum transmission speed of each to increase the larger the

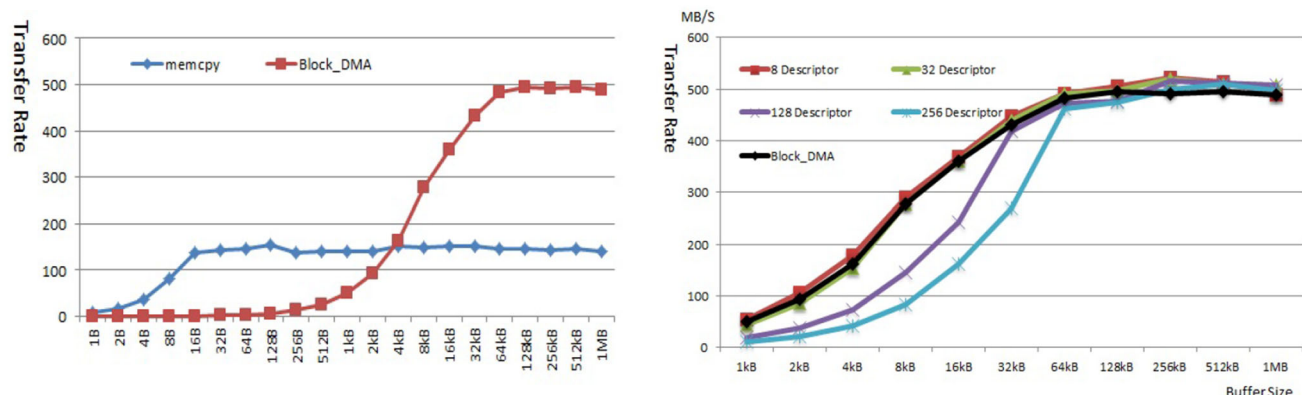


Fig. 17 Evaluation result of non-DMA and block DMA transfer, block DMA and SGDMA (MB/sec)

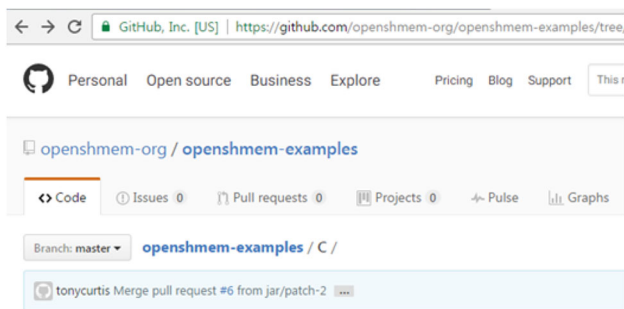


Fig. 18 Result of block DMA and SGDMA transfer (MB/sec)

size of the buffer transfer rate of about 150, 500MB/s. For SGDMA larger the size of the buffer, the number of descriptor was confirmed to represent a slightly better performance than the DMA Block. The transfer rate is low in transmission rate of up to about 520MB/s. Due to the limitation of allocation for physically contiguous buffer the performance resulted in a lower performance than theoretical bound. By reducing the overhead due to buffer copy and non-contiguous physical buffer, we can achieve higher bandwidth up to theoretically bound.

Figure 18 shows the benchmark application which we used in this experiment. 7 applications of github.com/openshmem-

examples are working on our platform : rotget.c, rotput.c, randput.c, shmem_all.c, dip.c, swap.c, and matrix multiplication.c. The reason why the rest of applications on the github.com are not working is because of limited APIs support at the initial stage of our implementation.

Figures 19, 20, and 21 provide the results of execution the applications mention above. The rotput.c rotates PE id to right neighbor (dest), with wrap-around. The rotget.c gets from right neighbor, with wrap-around. randput.c gets from 0 (master) to some other random PE. The shmem_all.c shows how to use shmem_put to simulate MPI_Alltoall. Each processor send/rec a different random number to/from other processors. The dip.c is a double value put application from PE 0 to PE 1 to shmem_malloc'ed variable. The swap.c swap values between odd numbered PEs and their right (modulo) neighbor. The shmem_matrix.c calculates the product of 2 matrices A and B based on block distribution. This is adopted from the mpi implementation of matrix multiplication based on 1D block-column distribution.

From Figs. 22, 23, 24, 25, 26 and 27 are the result of execution. The x-axis is block size and y-axis is the proportional execution time. All applications, with exception of shmem_all, commonly provide that larger block size will result in longer execution time. The shmem_all application

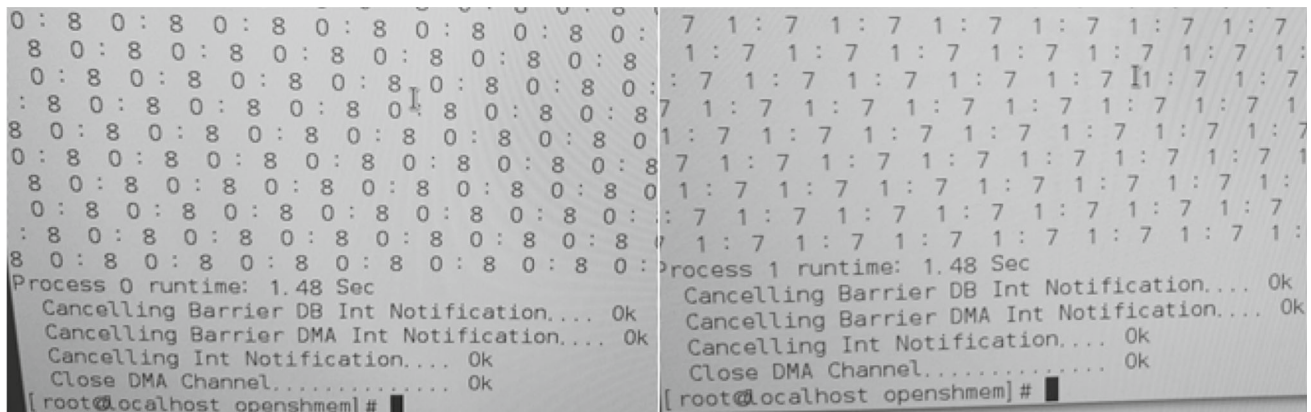


Fig. 19 Result of execution rotget.c

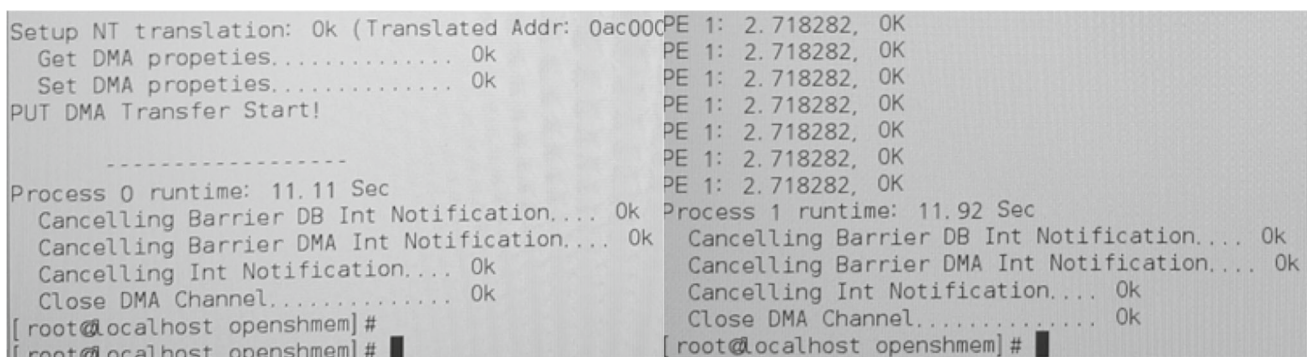


Fig. 20 Result of execution dip.c

```

Result: 1 @ localhost.localdomain: 866
Result: 1 @ localhost.localdomain: 390
Result: 1 @ localhost.localdomain: 551
Result: 1 @ localhost.localdomain: 178
Result: 1 @ localhost.localdomain: 571
Result: 1 @ localhost.localdomain: 391
Result: 1 @ localhost.localdomain: 750
Result: 1 @ localhost.localdomain: 654
Result: 1 @ localhost.localdomain: 470
Result: 1 @ localhost.localdomain: 579
Result: 1 @ localhost.localdomain: 340
Result: 1 @ localhost.localdomain: 206
Result: 1 @ localhost.localdomain: 203
Result: 1 @ localhost.localdomain: 514
Result: 1 @ localhost.localdomain: 396
Result: 1 @ localhost.localdomain: 836
Result: 1 @ localhost.localdomain: 925
Result: 1 @ localhost.localdomain: 788
Result: 1 @ localhost.localdomain: 32
Process 1 runtime: 2.97 Sec
Cancelling Barrier DB Int Notification... Ok
Cancelling Barrier DMA Int Notification... Ok
Cancelling Int Notification... Ok
Close DMA Channel... Ok
[root@localhost openshmem]#

```

```

1070 3290 3710 4130 4550 4970 5390 5810 6230 6650 7070 7490 7910 8330 8750 9170 9590 10010 10430 10850
11270 11690 12110 12530 12950 13370 13790 14210 14630 15050 15470 15890 16310 16730 17150 17570 17990 18410 18830
19250 19670 20090 20510 20930 21350 21770 22190 22610 23030 23450 23870 24290 24710 25130 25550 25970 26390 26810
27230 27650 28070 28490 28910 29330 29750 30170 30590 31010 31430 31850 32270 32690 33110 33530 33950 34370 34790
35210 35630 36050 36470 36890 37310 37730 38150 38570 38990 39410 39830 40250 40670 41090 41510 41930 42350 42770
43190 43610 44030 44450 44870 45290 45710 46130 46550 46970 47390 47810 48230 48650 49070 49490 49910 50330 50750
51170 51590 52010 52430 52850 53270 53690 54110 54530 54950 55370 55790 56210 56630 57050 57470 57890 58310 58730
59150 59570 60000 60420 60840 61260 61680 62100 62520 62940 63360 63780 64200 64620 65040 65460 65880 66300 66720
67140 67560 67980 68400 68820 69240 69660 70080 70500 70920 71340 71760 72180 72600 73020 73440 73860 74280 74700
75120 75540 75960 76380 76800 77220 77640 78060 78480 78900 79320 79740 80160 80580 81000 81420 81840 82260 82680
83100 83520 83940 84360 84780 85200 85620 86040 86460 86880 87300 87720 88140 88560 88980 89400 89820 90240 90660
91080 91500 91920 92340 92760 93180 93600 94020 94440 94860 95280 95700 96120 96540 96960 97380 97800 98220 98640
99060 99480 99900 100320 100740 101160 101580 102000 102420 102840 103260 103680 104100 104520 104940 105360 105780 106200 106620 107040
107460 107880 108300 108720 109140 109560 109980 110400 110820 111240 111660 112080 112500 112920 113340 113760 114180 114600 115020
115440 115860 116280 116700 117120 117540 117960 118380 118800 119220 119640 120060 120480 120900 121320 121740 122160 122580 123000
123420 123840 124260 124680 125100 125520 125940 126360 126780 127200 127620 128040 128460 128880 129300 129720 130140 130560 130980 131400
131820 132240 132660 133080 133500 133920 134340 134760 135180 135600 136020 136440 136860 137280 137700 138120 138540 138960 139380 139800
140220 140640 141060 141480 141900 142320 142740 143160 143580 144000 144420 144840 145260 145680 146100 146520 146940 147360 147780 148200
148620 149040 149460 149880 150300 150720 151140 151560 151980 152400 152820 153240 153660 154080 154500 154920 155340 155760 156180 156600
157020 157440 157860 158280 158700 159120 159540 159960 160380 160800 161220 161640 162060 162480 162900 163320 163740 164160 164580 165000
165420 165840 166260 166680 167100 167520 167940 168360 168780 169200 169620 170040 170460 170880 171300 171720 172140 172560 172980 173400
173820 174240 174660 175080 175500 175920 176340 176760 177180 177600 178020 178440 178860 179280 179700 180120 180540 180960 181380 181800
182220 182640 183060 183480 183900 184320 184740 185160 185580 186000 186420 186840 187260 187680 188100 188520 188940 189360 189780 190200
190620 191040 191460 191880 192300 192720 193140 193560 193980 194400 194820 195240 195660 196080 196500 196920 197340 197760 198180 198600
199020 199440 199860 200280 200700 201120 201540 201960 202380 202800 203220 203640 204060 204480 204900 205320 205740 206160 206580 207000
207420 207840 208260 208680 209100 209520 209940 210360 210780 211200 211620 212040 212460 212880 213300 213720 214140 214560 214980 215400
215820 216240 216660 217080 217500 217920 218340 218760 219180 219600 220020 220440 220860 221280 221700 222120 222540 222960 223380 223800
224220 224640 225060 225480 225900 226320 226740 227160 227580 228000 228420 228840 229260 229680 230100 230520 230940 231360 231780 232200
232620 233040 233460 233880 234300 234720 235140 235560 235980 236400 236820 237240 237660 238080 238500 238920 239340 239760 240180 240600
241020 241440 241860 242280 242700 243120 243540 243960 244380 244800 245220 245640 246060 246480 246900 247320 247740 248160 248580 249000
249420 249840 250260 250680 251100 251520 251940 252360 252780 253200 253620 254040 254460 254880 255300 255720 256140 256560 256980 257400
257820 258240 258660 259080 259500 259920 260340 260760 261180 261600 262020 262440 262860 263280 263700 264120 264540 264960 265380 265800
266220 266640 267060 267480 267900 268320 268740 269160 269580 270000 270420 270840 271260 271680 272100 272520 272940 273360 273780 274200
274620 275040 275460 275880 276300 276720 277140 277560 277980 278400 278820 279240 279660 280080 280500 280920 281340 281760 282180 282600
283020 283440 283860 284280 284700 285120 285540 285960 286380 286800 287220 287640 288060 288480 288900 289320 289740 290160 290580 291000
291420 291840 292260 292680 293100 293520 293940 294360 294780 295200 295620 296040 296460 296880 297300 297720 298140 298560 298980 299400
299820 300240 300660 301080 301500 301920 302340 302760 303180 303600 304020 304440 304860 305280 305700 306120 306540 306960 307380 307800
308220 308640 309060 309480 309900 310320 310740 311160 311580 312000 312420 312840 313260 313680 314100 314520 314940 315360 315780 316200
316620 317040 317460 317880 318300 318720 319140 319560 319980 320400 320820 321240 321660 322080 322500 322920 323340 323760 324180 324600
325020 325440 325860 326280 326700 327120 327540 327960 328380 328800 329220 329640 330060 330480 330900 331320 331740 332160 332580 333000
333420 333840 334260 334680 335100 335520 335940 336360 336780 337200 337620 338040 338460 338880 339300 339720 340140 340560 340980 341400
341820 342240 342660 343080 343500 343920 344340 344760 345180 345600 346020 346440 346860 347280 347700 348120 348540 348960 349380 349800
350220 350640 351060 351480 351900 352320 352740 353160 353580 354000 354420 354840 355260 355680 356100 356520 356940 357360 357780 358200
358620 359040 359460 359880 360300 360720 361140 361560 361980 362400 362820 363240 363660 364080 364500 364920 365340 365760 366180 366600
367020 367440 367860 368280 368700 369120 369540 369960 370380 370800 371220 371640 372060 372480 372900 373320 373740 374160 374580 375000
375420 375840 376260 376680 377100 377520 377940 378360 378780 379200 379620 380040 380460 380880 381300 381720 382140 382560 382980 383400
383820 384240 384660 385080 385500 385920 386340 386760 387180 387600 388020 388440 388860 389280 389700 390120 390540 390960 391380 391800
392220 392640 393060 393480 393900 394320 394740 395160 395580 396000 396420 396840 397260 397680 398100 398520 398940 399360 399780 400200
400620 401040 401460 401880 402300 402720 403140 403560 403980 404400 404820 405240 405660 406080 406500 406920 407340 407760 408180 408600
409020 409440 409860 410280 410700 411120 411540 411960 412380 412800 413220 413640 414060 414480 414900 415320 415740 416160 416580 417000
417420 417840 418260 418680 419100 419520 419940 420360 420780 421200 421620 422040 422460 422880 423300 423720 424140 424560 424980 425400
425820 426240 426660 427080 427500 427920 428340 428760 429180 429600 430020 430440 430860 431280 431700 432120 432540 432960 433380 433800
434220 434640 435060 435480 435900 436320 436740 437160 437580 438000 438420 438840 439260 439680 440100 440520 440940 441360 441780 442200
442620 443040 443460 443880 444300 444720 445140 445560 445980 446400 446820 447240 447660 448080 448500 448920 449340 449760 450180 450600
451020 451440 451860 452280 452700 453120 453540 453960 454380 454800 455220 455640 456060 456480 456900 457320 457740 458160 458580 459000
459420 459840 460260 460680 461100 461520 461940 462360 462780 463200 463620 464040 464460 464880 465300 465720 466140 466560 466980 467400
467820 468240 468660 469080 469500 469920 470340 470760 471180 471600 472020 472440 472860 473280 473700 474120 474540 474960 475380 475800
476220 476640 477060 477480 477900 478320 478740 479160 479580 480000 480420 480840 481260 481680 482100 482520 482940 483360 483780 484200
484620 485040 485460 485880 486300 486720 487140 487560 487980 488400 488820 489240 489660 490080 490500 490920 491340 491760 492180 492600
493020 493440 493860 494280 494700 495120 495540 495960 496380 496800 497220 497640 498060 498480 498900 499320 499740 500160 500580 501000
501420 501840 502260 502680 503100 503520 503940 504360 504780 505200 505620 506040 506460 506880 507300 507720 508140 508560 508980 509400
509820 510240 510660 511080 511500 511920 512340 512760 513180 513600 514020 514440 514860 515280 515700 516120 516540 516960 517380 517800
518220 518640 519060 519480 519900 520320 520740 521160 521580 522000 522420 522840 523260 523680 524100 524520 524940 525360 525780 526200
526620 527040 527460 527880 528300 528720 529140 529560 529980 530400 530820 531240 531660 532080 532500 532920 533340 533760 534180 534600
535020 535440 535860 536280 536700 537120 537540 537960 538380 538800 539220 539640 540060 540480 540900 541320 541740 542160 542580 543000
543420 543840 544260 544680 545100 545520 545940 546360 546780 547200 547620 548040 548460 548880 549300 549720 550140 550560 550980 551400
551820 552240 552660 553080 553500 553920 554340 554760 555180 555600 556020 556440 556860 557280 557700 558120 558540 558960 559380 559800
560220 560640 561060 561480 561900 562320 562740 563160 563580 564000 564420 564840 565260 565680 566100 566520 566940 567360 567780 568200
568620 569040 569460 569880 570300 570720 571140 571560 571980 572400 572820 573240 573660 574080 574500 574920 575340 575760 576180 576600
577020 577440 577860 578280 578700 579120 579540 579960 580380 580800 581220 581640 582060 582480 582900 583320 583740 584160 584580 585000
585420 585840 586260 586680 587100 587520 587940 588360 588780 589200 589620 590040 590460 590880 591300 591720 592140 592560 592980 593400
593820 594240 594660 595080 595500 595920 596340 596760 597180 597600 598020 598440 598860 599280 599700 600120 600540 600960 601380 601800
602220 602640 603060 603480 603900 604320 604740 605160 605580 606000 606420 606840 607260 607680 608100 608520 608940 609360 609780 610200
610620 611040 611460 611880 612300 612720 613140 613560 613980 614400 614820 615240 615660 616080 616500 616920 617340 617760 618180 618600
619020 619440 619860 620280 620700 621120 621540 621960 622380 622800 623220 623640 624060 624480 624900 625320 625740 626160 626580 627000
627420 627840 628260 628680 629100 629520 629940 630360 630780 631200 631620 632040 632460 632880 633300 633720 634140 634560 634980 635400
635820 636240 636660 637080 637500 637920 638340 638760 639180 639600 640020 640440 640860 641280 641700 642120 642540 642960 643380 643800
644220 644640 645060 645480 645900 646320 646740 647160 647580 648000 648420 648840 649260 649680 650100 650520 650940 651360 651780 652200
652620 653040 653460 653880 654300 654720 655140 655560 655980 656400 656820 657240 657660 658080 658500 658920 659340 659760 660180 660600
661020 661440 661860 662280 662700 663120 663540 663960 664380 664800 665220 665640 666060 666480 666900 667320 667740 668160 668580 669000
669420 669840 670260 670680 671100 671520 671940 672360 672780 673200 673620 674040 674460 674880 675300 675720 676140 676560 676980 677400
677820 678240 678660 679080 679500 679920 680340 680760 681180 681600 682020 682440 682860 683280 683700 684120 684540 684960 685380 685800
686220 686640 687060 687480 687900 688320 688740 689160 689580 689999 690419 690839 691259 691679 692099 692519 692939 693359 693779 694199 694619
695039 695459 695879 696299 696719 697139 697559 697979 698399 698819 699239 699659 700079 700499 700919 701339 701759 702179 702599 703019
703439 703859 704279 704699 705119 705539 705959 706379 706799 707219 707639 708059 708479 708899 709319 709739 710159 710579 710999 711419
711839 712259 712679 713099 713519 713939 714359 714779 715199 715619 716039 716459 716879 717299 717719 718139 718559 718979 719399 719819
720239 720659 721079 721499 721919 722339 722759 723179 723599 724019 724439 724859 725279 725699 726119 726539 726959 727379 727799 728219
728639 729059 729479 729899 730319 730739 731159 731579 731999 732
```

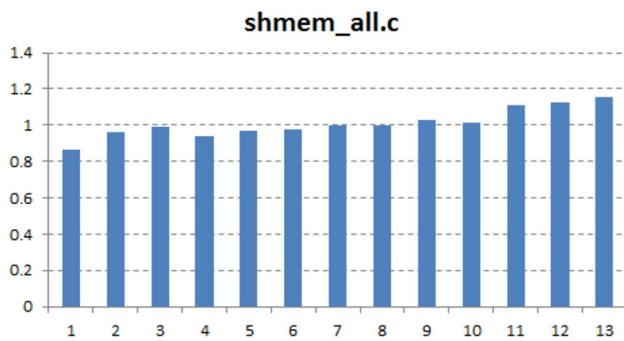


Fig. 26 Execution result of shmem_all.c

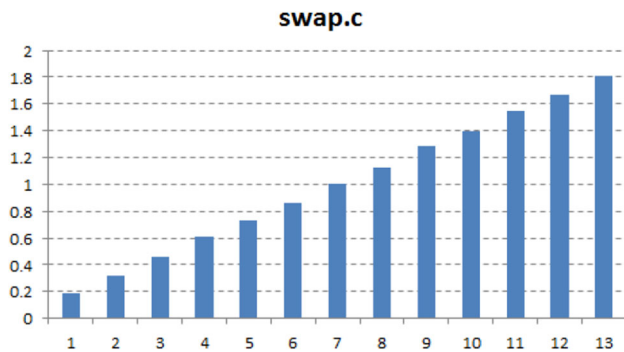


Fig. 27 Execution result of swap.c

The API we have implemented is shown in the following Fig. 28. Since the result of retrieving the API from the project source code, the same API was searched once in the c file and once in the .h file. It indicates that the initial version of GPI / OpenSHMEM works well in the PCIe interconnection network as originally intended.

Fig. 28 Supported APIs of our initial OpenSHMEM

Symbol	File	Line
shmem_barrier_all	Samples\openshmem\rdma.c	3230
shmem_double_get	Samples\openshmem\rdma.c	1948
shmem_double_p	Samples\openshmem\rdma.c	1637
shmem_double_put	Samples\openshmem\rdma.c	1528
shmem_finalize	Samples\openshmem\rdma.c	272
shmem_free	Samples\openshmem\rdma.c	2342
shmem_init	Samples\openshmem\rdma.c	451
shmem_int_put	Samples\openshmem\rdma.c	1838
shmem_long_get	Samples\openshmem\rdma.c	2059
shmem_long_put	Samples\openshmem\rdma.c	1728
shmem_long_swap	Samples\openshmem\rdma.c	2170
shmem_malloc	Samples\openshmem\rdma.c	1144
shmem_my_pe	Samples\openshmem\rdma.c	1050
shmem_n_pes	Samples\openshmem\rdma.c	1036
shmem_barrier_all	Samples\openshmem\rdma.h	33
shmem_double_get	Samples\openshmem\rdma.h	22
shmem_double_p	Samples\openshmem\rdma.h	16
shmem_double_put	Samples\openshmem\rdma.h	14
shmem_finalize	Samples\openshmem\rdma.h	6
shmem_free	Samples\openshmem\rdma.h	30
shmem_init	Samples\openshmem\rdma.h	8
shmem_int_put	Samples\openshmem\rdma.h	20
shmem_long_get	Samples\openshmem\rdma.h	24
shmem_long_put	Samples\openshmem\rdma.h	18
shmem_long_swap	Samples\openshmem\rdma.h	26
shmem_malloc	Samples\openshmem\rdma.h	28
shmem_my_pe	Samples\openshmem\rdma.h	12
shmem_n_pes	Samples\openshmem\rdma.h	10

4 Conclusions

Although PCI Express-based interconnection network environment is not popular now, we have established a PCI Express-based interconnection network experimental environment. OpenSHMEM technology has already become widespread in HPC interconnection networks such as InfiniBand and RoCE. However, research on PCI Express-based HPC interconnection networks is only the beginning of the world. We conducted this study in case the PCI Express NTB-based HCA adapter is widely available. Through this research, we propose the possibility of using it as the OpenSHMEM framework of the next generation Interconnection Network. This research is applicable to HCA for HPC interconnection network. This study is also meaningful in terms of securing next generation system software technology. We have learned the driving and control technology of the PLX PEX8749. We designed and implemented the initial OpenSHMEM framework and performed verification of the correctness and compatibility of the framework. Finally, a few APIs of our initial OpenSHMEM implementation are demonstrated by running 7 OpenSHMEM examples of github.com.

Acknowledgements This research has been performed as a subproject of Project No. K-17-L01-C01 (Development of key enabling technologies for massively parallel and high-density computing) supported by the KOREA INSTITUTE of SCIENCE and TECHNOLOGY INFORMATION (KISTI). This research is also jointly Supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (NRF-2016R1C1B1012189).

References

- Helal, A.A., Kim, Y.W., Ren, Y., Choi, W.H.: Design and implementation of an alternate system inter-connect based on PCI Express. *J. Inst. Electron. Inform. Eng.* **52**(8), 74–85 (2015)
- Liu, J., Mamidala, A., Vishnu, A., Panda, D.K.: Evaluating infiniband performance with PCI Express. *IEEE Micro* **24**(1), 20–29 (2005)
- Heymian, W.: PCI Express multi-root switch reconfiguration during system operation. M. Eng. Thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, Cambridge (2011)
- Krishnan, V.: Towards an integrated IO and clustering solution using PCI express. 2007 IEEE International Conference on Cluster Computing [Online]. pp. 259–266. <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4629239> (2007)
- Mohrmann, L., Tongen, J., Friedman, M., Wetzol, M.: Creating multicomputer test systems using PCI and PCI Express. IEEE AUTOTESTCON [Online]. pp. 7–10. <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5314043> (2009)
- RCT Magazine. Enabling Multi-Host System Designs with PCI Express Technology [Internet]. <http://www.rctmagazine.com/articles/view/100015.8>
- Jo, H., Jeong, J., Lee, M., Choi, D.: Exploiting GPUs in Virtual Machine for BioCloud. *BioMed Research International*, vol. 2013, Article ID 939460 (2013)
- Vienne, J., Chen, J., Wasi-ur-Rahman, M., Islam, N., Subramoni, H., Panda, D.: Performance Analysis and Evaluation of Infiniband FDR and 40GigE RoCE on HPC and Cloud Computing Systems, IEEE 20th Annual Symposium on High-Performance Interconnects (2012)
- Choi, M., Park, J.H.: Feasibility and performance analysis of RDMA transfer through PCI Express. *J. Inform. Process. Syst.* **13**(1), 95–103 (2017)
- Top500.org.: Interconnect Family Statistics [Internet]. <http://top500.org/statistics/list> (2015)
- Chen, L., Zhou, W., Wu, Q.: A design of high-speed image acquisition card based on PCI EXPRESS. 2010 International Conference on Computer Application and System Modeling (ICCSM 2010) [Online]. vol. 3, pp. V3-551–V3-554. <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5620696> (2010)
- Kavianipour, H., Bohm, C.: High performance FPGA-based scatter/gather DMA interface for PCIe. Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), 2012 IEEE [Online]. pp. 1517–1520. <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6551364> (Oct. 27 2012–Nov. 3 2012)
- Buschettu, A., Sanna, D., Concas, G., Eros, F.: A platform based on kanban to build taxonomies and folksonomies for DMS and CSS. *J. Conver. (JoC)* **6**(1), 1–8 (2015)
- Rota, L., Caselle, M., Chilingaryan, S., Kopmann, A., Weber, M.: A new DMA PCIe architecture for Gigabyte data transmission. Real Time Conference (RT), 2014 19th IEEE-NPSS [Online]. pp. 1–2. <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7097561> (2014)
- ExpressLane PEX8749 PCI Express Gen 3 Multi-Root Switch with DMA Data Book, PLX Technology (2013)
- Richter, A., Herber, C., Wild, T., Herkersdorf, A.: Resolving Performance Interference in SR-IOV Setups with PCIe Quality-of-Service Extensions. Euromicro Conference on Digital System Design (2016)
- Sinha, A., Lobiyal, D.K.: Performance evaluation of data aggregation for cluster wireless sensor network. *Human-centric Comput. Inform. Sci. (HCIS)* **3**(13), 2–17 (2013)
- Naoui, M., Mahmoudi, S., Belalem, G.: Feasibility study of a distributed and parallel environment for implementing the standard version of AAM model. *J. Inform. Process. Syst. (JIPS)* **12**(1), 149–168 (2016)
- Mellor-crummey, J.M.: Algorithm for scalable synchronization on shared-memory multiprocessors. *ACM Trans. Comput. Syst.* **9**(1), 21–65 (1991)



Internet of Things in Embedded systems and software application in Web Application Server.



is currently a senior research staff. His research interests include parallel and cluster computing, parallel file system, and high-performance computer systems.



include high performance computing, cloud computing, interconnection network, and embedded computing.

Cheol Shim was born in Cheongju-si, Republic of Korea, in 1990. He received the B.S. degree in Computer Engineering from the Chungbuk National University, Cheongju-si, Republic of Korea, in 2016, and He is currently in the Master's course of Information and Communication Engineering at the Chungbuk National University, Cheongju-si, Republic of Korea. His research interests include Cloud Computing and Embedded Software, Real-Time Operating System,

Kwangho Cha received the B.S. degree in Computer Science from Soongsil University, Korea in 1999 and the M.E. degree in Information and Computer Engineering from Information and Communications University (ICU), Korea in 2002 and the Ph.D. degree in Computer Science from Korea Advanced Institute of Science and Technology (KAIST) in 2012. Since 2002, he has been with Division of Supercomputing of Korea Institute of Science and Technology Information, where he

Min Choi received the B.S. degree in Computer Science from Kwangwoon University, Korea, in 2001, and the M.S. and Ph.D. degrees in Computer Science from Korea Advanced Institute of Science and Technology (KAIST) in 2003 and 2009, respectively. From 2008 to 2010, he worked for Samsung Electronics as a Senior Engineer. Since 2011 he has been a faculty member of Department of Information and Communication of Chungbuk National University. His current research interests