Non-Transparent Bridge 기반 링 네트워크 통신 방식 구현

김상겸*, 이양우*, 임승호*
*한국외국어대학교 컴퓨터전자시스템공학부
e-mail:slim@hufs.ac.kr

Implementation of Non-Transparent Bridge Interface-based Ring Topology

Sang-Gyum Kim*, Yang-Woo Lee**, Seung-Ho Lim*
*Division of Computer and Electronic Systems Engineering
Hankuk University of Foreign Studies

요 약

다수의 계산노드를 초고성능 상호연결망으로 연결하여 클러스터 시스템으로 구성하는 HPC에서 인터콘넥션 기술로 Infiniband, Ethernet등의 기술이 많이 사용된다. PCIe 기반의 노드간 직접 연결 기술로는 NTB(Non-Transparent Bridge) 기반의 인터콘넥션 기술이 있으나, NTB의 기본은 두 노드간에 분리된 메모리를 공유하는 방식이다. 본 논문에서는 다중 NTB 포트에 직접 연결된 다수의 호스트들간에 무스위치 네트워크를 구성하여 NTB 통신을 이용한 데이터 공유 방법의 설계와 구현에 대해서 다룬다. 각 호스트에 연결된 두 개의 NTB포트를 이용해서 링 네트워크를 구성하고, 링 네트워크 상에서 NTB 인터컨넥션을 이용한 데이터 공유 방식의 구현을 하였다. 이와 같이 PCIe 기반 무스위치 네트워크를 통해서 Cost-Effective한 HPC 상호연결망을 구성할 수 있다.

1. 서론

HPC(High Performance Computing) 시스템은 다수의 계산노드(Computation Node)를 초고성능 상호연결망(Interconnection Network)로 연결하여 클러스터 시스템으로 구성하고 있으며, 이러한 상호연결망 기술은 HPC 시스템의 성능과 안정성을 결정하는 중요한 기술 요소이다.

현재 대부분의 HPC 시스템의 Interconnect 기술은 Infiniband 및 Ethernet Interconnect 기술에 의존하고 있음으로 인해서 두 기술에 대한 기술 의존성이 많이 있다. 이에, 주요 HPC 리딩 연구기관들은 자체 Interconnect 기술을 확보하기 위해서 많은 연구를 진행하고 있으며 이를 기반으로 Top500의 주요 HPC를 선보이고 있다. 예로써 중국의 텐허는 TH Express-2를 일본의 Kei는 Tofu를 사용하고 있으며, 미국 또한 타이탄의 Cray Gemini와 Intel의 Omni-Path와 같은 독자적인 상호연결망 시스템을 사용하고 있다.

고성능 수퍼컴퓨터(HPC)를 자체적으로 개발하기 위해서는 기술의 존성이 높은 인터커넥트 기술인 Infiniband/Ethernet 인터컨넥트가 아닌 대체 인터컨넥트 기술로써, Low Power와 High Performance 기능을 달성할 수 있는, 컴퓨터 내부의 CPU와 메모리/주변장치등을 연결하는데 사용되는 PCIe 기반 Interconnect 기술을 고려할 수 있다.

PCIe 스위칭 기술의 경우, PCIe 스위칭 칩 제조사의 패브릭 네트워크 기능에 대한 기술지원이 원활하게 이루어지지 못함에 따라 PCIe 스위칭 기술을 인터커넥션 네트워크로 활용하기 위한 대체 방안으로 PCIe의 NTB(Non-Transparent Bridge)기술을 활용하는 방안을 연구 개발할 필요가 생기게 되었다.

PCIe NTB(Non-Transparent Bridge)는 PCI-Express bridge chip에서 지원하는 모드 중 하나로서, 2대 컴퓨터의 서로 분리된 메모리 시스템을 같은 PCI-Express fabric 으로 연결시키는 기술이다. 현재까지 진행된 PCIe 기반 NTB 인터컨넥션 기술은, 2대의 호스트간에 서로 분리된 메모리를 접근하도록 구성하는 연결방식에 대해서 연구 개발이 이루어져 왔으며, 3대 이상의 호스트간에 서로 부리된 메모리를 구성하도록 하는 다중 호스트 기반으 NTB 인터컨넥션 기술은, 스위칭 기반의 NTB 연결방식에 대한 연구 및 직접적인 호스트간 연결에 의한 클러스터 구성에 대한 연구가 진행중이긴 하나 그 결과물이 미비한 상황이다.

본 연구개발에서는 PCIe의 NTB(Non-Transparent Bridge) 기술을 활용해서 다중 호스트 기반의 PCIe 무스위치 인터컨넥션 통신방안(Switchless Interconnection Network)에 대해서 연구해보도록 한ㄷ. 구체적으로는 다중 포트 기반의 호스트간 직접적인 NTB 연결을 통하여 링 토폴로지를 구성하고, NTB 기반의 다중 호스트 클러스터 내에서 데이터 공유 방법에 대한 설계 및 구현을 하도록 한다.

2. NTB Ring Topology 구현

PCIe NTB(Non-Transparent Bridge)는 PCI-Express bridge chip에서 지원하는 모드 중 하나로서, 2대 이상의 컴퓨터의 서로 분리된 메모리 시스템을 같은 PCI-Express fabric 으로 연결시키는 기술이다. PCIe NTB(Non Transparent Bridge)는 TB(Transparent Bridge)와 마찬가지로 독립적인 PCI bus(PCI 또는 PCI Express bus)에 대해서 데이터 전송 경로(path)를 제공한다는 점에서 유사하다. 그러나, TB와의 가장 큰 차이점은 NTB가 사용될 경우에 bridge의 하향부분(downstream side)에 위치한 장치들은 상향부분(upstream side)에서는 보이지 않는다는 점이다. 이는 bridge의 하향부분(downstream side)에 위치한 인텔리전트(intelligent)한 시스템이 자신의 downstream side에 위치하는 서브시스템 내 각종 장치들을 독립적으로 관리할 수 있다.

두 대의 컴퓨터를 NTB를 이용해서 연결하는 방법은 Primary-Secondary 연결 방식과 B2B(Back to Back) 연결 방식을 생각할 수 있다. 첫 번째로 오른쪽 그림과 같이, PCIe NTB는 또한 primary host의 PCI bus로 구성된 서브시스템 계층구조에 secondary host를 연결하는 데 사용될 수 있다. 두 번째로, 왼쪽 그림과 같이 Back to Back(B2B) NTB를 두 호스트간에 연결하게 되면, 하나의 호스트의 primary address가 NTB에 의해서 secondary address로 변환되어 해당 호스트의 address space에 보이지 않게 되며, 마찬가지로, secondary address 역시 다른 쪽 host의 NTB에 의해서 address map에 포함되지 않게 된다. 이 경우, 어느 한쪽에서 doorbell 인터럽트를 발생시키거나 scratchpad register를 통해서 interprocessor commmunication을 하기 위해서는 secondary side의 BAR 0/1를 접근하여 해당 doorbell register와 scratchpad register에 값을 쓸 수 있다.

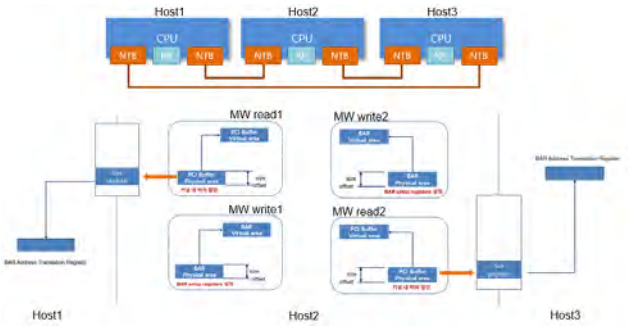

그림 1 다중포트 기반 NTB Ring Topology 구성 및 Memory Window 구성

기본적으로 NTB는 두 대의 호스트를 서로 다른 메모리 영역을 통해서 데이터 공유 또는 전송하는 방식을 제공하는데, 본 논문에서는 하나의 호스트에 두 개 이상의 NTB 포트를 위한 호스트 아답터를 연결하고, 각 호스트의 포트를 양쪽의 다른 호스트에 연결하는 방식으로 세 대 이상의 호스트를 링 형식으로 연결하여 다중 호스트를 구성한다. 그림 1은 다중 포트 기반의 NTB 링 클러스터를 구성한 그림을 도식화한 것이다.

그림에서와 같이, NTB 링 토폴로지를 구성하면, 각 호스트당 각각의 NTB를 이용해 연결된 다른 쪽의 호스트에 대해서 분리된 메모리 어드레스 영역을 통해서 데이터 공유를 할 수 있다. 그러나, 자신에게 직접 연결되지 않은 호스트와 데이터를 공유하기 위해서는 해당 호스트를 지정해주는 Id 같은 것들이 필요하게 된다. 즉, Id와 메모리 주소를 통해서 데이터를 공유할 수 있다.

그림 2는 초기 링 구성시 각각의 호스트에서 설정해주는 Id와 메모리 윈도우를 나타낸 것이다. 그림과 같이 초기 설정 시 각각의 호스트는 Id를 부여받게 되며, 각각의 NTB 포트당 연결된 호스트위 NTB와 데이터를 주고받을 메모리 윈도우를 할당한다. 할당된 메모리 윈도우는 NTB의 Translation Register를 통해서 address translation되어 상대방 호스트의 메모리에 접근할 수 있는 통로가 된다.

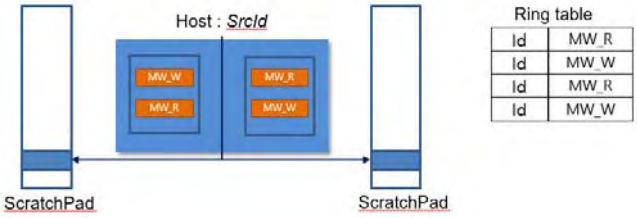

그림 2 NTB 구성에서의 Ring 구성 셋업 단계

다중 포트 NTB로 구성된 Ring Topology 네트워크에서 데이터의 데이터 전송은 한 방향 통신을 통해서 데이터를 주고 받으며, 데이터 전송은 연결된 호스트간의 NTB로 공유된 공유영역을 통해서 전송한다. 전송하는 방식은 공유 데이터 영역의 메모리 복사 또는 DMA 둘다 가능하다. 반면, 데이터 전송이 아닌 다른 정보들의 전송은 둘간에 연결된 NTB의 ScratchPad register를 통해서 전송한다. 이러한 정보들은 SrcId, DestId, Memory 주소, 전송 데이터 크기등이 해당된다.

다음과 같은 데이터 프로토콜을 통해서 수행된다. 특정 호스트에서 데이터를 전송하려고 할 때, 해당 호스트는 송신자가 되며, SrcId를 자기 자신으로 설정하며, 수신하려는 호스트의 Id를 지정하여 DestId가 된다. 만약, DestId가 데이터 전송이 가능한 자신과 직접 연결된 호스트이면, 해당 데이터는 둘 간의 직접적인 공유 데이터 메모리에 데이터 쓰기를 수행한다. 그리고, SrcId, DestId, 메모리 주소, 전송 크기등은 ScratchPad Register를 통해서 전달한다. DMA를 통해서 데이터 전송이 완료되면, Doorbell Register를 통해서 해당 호스트에게 DMA 데이터 전송을 알리는 인터럽트를 발생시켜, 상대 호스트에게 전송된 데이터가 있음을 알린다. 인터럽트를 받은 호스트는

ScratchPad Register를 통해서 자신이 수신자인지 알게 되면, 자신이 수신자인 경우, 해당 메모리 주소로부터 직접 자신에게 전송된 데이터를 볼 수 있게 된다. 만약 자신과 직접 연결되지 않은 호스트에게 데이터를 전송하려면, DestId에 전송하려는 호스트의 Id를 기재하여, NTB로 연결된 호스트이 공유 메모리로 데이터를 전송한다. 마찬가지로 SrcId, DestId, 메모리 주소, 전송데이터 크기 등은 ScratchPad Register로 보내며, Doorbell Register를 이용해서 인터럽트를 발생시킨다. 인터럽트를 받은 이웃 호스트는 DestId를 확인해서, 자신이 송신자가 아님을 확인하면, 전송받은 데이터를 다른 이웃에게 데이터를 포워딩해 준다. 데이터를 포워딩하는 방식은 마찬가지로 다른 이웃과 공유된 메모리 영역을 통해서 DMA로 전송이 가능하다. 이와 같이 수신자 호스트가 데이터를 전송받을 때까지 계속 이어진다.

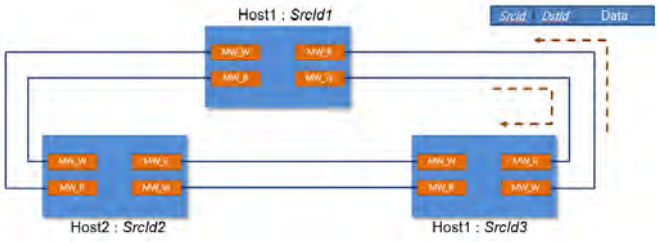

그림 3 NTB 구성에서의 Ring Data 전송

3. 구현 테스트

다중 NTB 포트 기반으로 링 네트워크를 구성하는 구현을 위해서 본 논문에서는 PLX사의 PEX8749/PEX8733을 이용해서 구현 및 개발한 PCIe NTB Adapter를 이용해서 구성하였다. 3대의 호스트에 각각 두 개의 NTB Adapter를 설치하고, 각각의 NTB Port를 서로 물려서 링 네트워크를 구성하도록 한 후, NTB Device Driver, NTB DMA device driver를 로딩하여 서로 NTB 동작이 가능하도록 구성하였다. 그 후, 위에서 설명한 링 네트워크 통신 환경 구성을 위한 셋업 단계를 거쳐서 각각의 호스트가 가지는 두 개의 NTB 포트 각각이 서로 공유 가능한 메모리를 가지도록 구현하였다. 링 네트워크 셋업 설정이 완료된 후에는, 서로 공유하고 있는 메모리 주소를 통해서 데이터 송수신이 가능하게 된다.

다수의 노드간의 데이터 공유 및 프로세싱의 어플리케이션으로 OpenSHMem과 같은 메모리 공유 방식의 프로그래밍 인터페이스를 주로 사용할 수가 있다. 그림 4는 OpenSHmem API를 지원하기 위한 PCIe NTB 지원 함수의 구현에 대한 예를 기술하였다. 그림에서와 같이 다중 포트 기반의 NTB로 구성된 호스트들간에 NTB 인터커넥트를 통한 HPC 클러스터를 구성하여, 노드들 간의 데이터 공유 및 병렬 프로세싱을 통하여 가속화할 수 있는 어플리케이션 개발이 가능하다

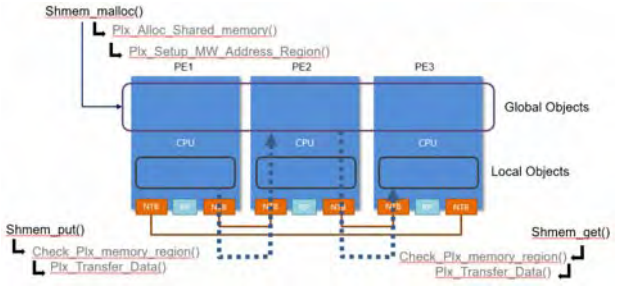

그림 4 NTB-based openSHMEM을 이용한 메모리 공유방식 구현 예

4. 결론

대부분의 HPC 시스템은 다수의 계산노드(Computation Node)를 Infiniband, Ethernet과 같은 고가의 고성능 상호연결망(Interconnection Network)로 연결하여 클러스터 시스템을 구성한다. 두 개의 노드간에 메모리 공유를 통해서 쉽게 데이터 송수신을 할 수 있는 인터커넥트 기술로서 PCIe NTB 기술이 있다. 그런데, 기본적인 PCIe NTB는 두 노드간의 분리된 메모리 영역을 통해서 데이터 공유를 할 수 있기 때문에, 다수의 노드를 연결하여 구성하기가 쉽지 않다. 본 논문에서는 다중 NTB 포트로 구성된 다수의 노드를 상호 연결하여 링 네트워크를 구성한 후, 네트워크 내의 노드들간의 상호 데이터를 공유할 수 있는 시스템을 구현하였다. 이러한 구현을 통해서 PCIe NTB 기반의 Cost-Effective하면서 고속의 데이터 공유가 가능한 클러스터 구성이 가능하다.

Acknowledgments

The Research was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (NRF-2016R1C1B2009914). 이 논문은 한국과학기술정보연구원(KISTI)의 2018년 위탁연구지원사업의 지원에 의해서 이루어진 것임.

참고문헌

[1] Ravi Budruk, PCI Express Basics, PCI-SIG
[2] Wikipedia, https://en.wikipedia.org/wiki/PCI_Express
[3] http://www.tldp.org/LDP/tlk/dd/pci.html
[4] Mark J. Sullivan, Intel Xeon Processor C5500/C3500 Series Non-transparent Bridge, Intel white paper
[5] Jack Regula, Using Non-transparent Bridging in PCI Express Systems, PLX Technology, Inc.
[6] Non-transparent Bridging with IDT 89HPES32NT24G2 PCI Express NTB Switch, Application Note AN-724, IDT

[7] Kwok Kong and Ale Chang, PCI express System Interconnect Software Architecture for x86-based Systems, Application Note AN-571, IDT
[8] Kwok Kong, Enabling Multi-peer Support with a Standard-Based PCI Express Multi-ported Switch, white paper, IDT
[9] Weihang Jiang, Jiuxing Liu, hyun-Wook Jin, D.K. Panda, W. Gropp and R. Thakur, "High performance MPI-2 one-sided communication over InfiniBand", Cluster Computing and the Grid, 2004. CCGrid 2004. IEEE International Symposium on, pp. 531-538, April. 2004.
[10] Yong-Hwan Lee, Do-Suk Kim and Sang Yoon Oh, "QoS and Flow Control Support on PCI Express Interface Architecture", Korea Institute of Information Technology Magazine, pp. 45-52, Dec. 2009.
[11] NTB white papers, AVAGO TECHNOLOGIES, http://www.avagotech.com/support/download-search. [Accessed: August. 29, 2016]
[12] OpenSHMEM Specification document, http://openshmem.org/site/Specification.