논문 17-42-11-10 The Journal of Korean Institute of Communications and Information Sciences '17-11 Vol.42 No.11
https://doi.org/10.7840/kics.2017.42.11.2122

RDMA 기반 고성능 네트워크 기술 동향

최 성 윤◆, 문 양 세*, 최 미 정°

Towards RDMA-Based High Performance Network Technologies

Choi Seong-Yun◆, Yang-Sae Moon*, Choi Mi-Jung°

요 약

최근 빅데이터가 이슈화되며 빠르게 증가하는 대용량의 데이터를 처리하기 위한 클라우드 컴퓨팅이 중요해지고 있다. 클라우드 컴퓨팅은 여러 서버들이 병렬처리를 수행하게 되는데, 클라우드의 연산 능력이 빨라지면서 네트워크의 전송횟수가 증가하게 되었다. 이는 결국 기존의 이더넷 네트워크 환경이 데이터를 처리함에 있어서 병목현상으로 작용하는 결과를 낳았다. 이에 따라 기존의 이더넷보다 성능이 좋은 RDMA (Remote Direct Memory Access) 기반의 네트워크 기술이 등장하게 되었다. RDMA 기법은 네트워크 내의 다른 컴퓨터의 기억 장치에 직접 접속하여 데이터를 교환, 사용하는 통신 방식으로 운영체제(OS)나 다른 자원들을 거치지 않기 때문에 지연 시간이 짧고 대량의 정보 전송이 가능하다. 또한 이러한 장점으로 대형 병렬 컴퓨터에 많이 활용된다. RDMA 기반의 네트워크는 VIA(Virtual Interface Architecture) 네트워크에서 확대된 기술로 인피니밴드, RoCE, iWARP가 존재한다. 최근에는 클러스터 환경에서 고속의 데이터 처리를 위해 RDMA 기반의 네트워크를 적용하는 연구들이 이루어지고 있다. 대다수의 연구들이 RDMA 네트워크 기술 중 인피니밴드를 사용하여 진행하고 있다. 본 논문은 RDMA 기반의 고성능 네트워크 기술들을 알아보고, 인피니밴드를 사용한 최근 연구 동향에 대해 요약하고자한다. 또한, 분산 환경에서 고속의 대용량 데이터스트림 처리를 위해 인피니밴드를 적용한 우리의 연구 사례를 소개하고자 한다.

Key Words : system area network, communication, rdma, data center, interconnect

ABSTRACT

Recently, as big data has become an issue, cloud computing has become important for processing large amount of data that grows rapidly. In cloud computing, several servers perform parallel processing. As the computing power of the cloud increases, the number of transmissions of the network traffic increases. This resulted in the existing Ethernet network environment becoming a bottleneck in processing the data. As a result, RDMA(Remote Direct Memory Access) based network technology, which has better performance than the existing Ethernet, has emerged. The RDMA method is a communication method of directly accessing to the memory of another computer in the network and exchanging data, and it bypasses operating system (OS) or other resources, so that the latency is low and a large amount of information can be transmitted. The RDMA-based network, including Infiniband, RoCE, and iWARP, is an expanded technology in the VIA(Virtual Interface Architecture) network. Recently, studies of applying RDMA-based network for high speed data processing in cluster environment

※ 이 논문은 2017년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No.R7117-17-0214, 데이터 스트림 정제를 위한 지능형 샘플링 및 필터링 기술 개발)
◆ First Author : Kangwon National University Department of Computer Science, seongyun@kangwon.ac.kr 학생회원
° Corresponding Author : Kangwon National University Department of Computer Science, mjchoi@kangwon.ac.kr, 종신회원
* Kangwon National University Department of Computer Science, ysmoon@kangwon.ac.kr, 정회원
논문번호 : KICS2017-08-212, Received August 3, 2017; Revised October 31, 2017; Accepted October 31, 2017

are being conducted. The majority of research is conducted using Infiniband among the RDMA network technologies. This paper examines RDMA-based high-performance network technologies and summarizes recent research trends using Infiniband. We also introduce our case study of high-speed large-capacity data stream processing based Infiniband in a cluster environment.

Ⅰ. 서 론

데이터는 곧 가치로 직결되는 4차 산업혁명 시대에 데이터를 다루는 기술은 중요해졌다. 사물인터넷(IoT), 모바일, 클라우드, 빅데이터 등의 기술들을 통해 데이터는 쉽게 만들어지고 기록이 가능해지고 가공되어져 우리에게 의미 있는 결과 혹은 피드백을 준다. 하지만 이러한 데이터들의 수집 양은 처리 가능한 한계를 크게 웃돌며 증가하고 있다. IDC(International Data Corporation)는 2020년까지 매 2년마다 데이터의 양이 2배씩 증가할 것으로 예측하고 있다[1].

이렇게 급속도로 증가하는 데이터는 고성능의 서버 한 대로 처리하기에는 저장용량과 네트워크 대역폭에 한계가 있다. 따라서, 비교적 저성능의 다수 서버들이 위치하는 데이터 센터 등에서 하둡과 같은 데이터 처리 플랫폼 등을 이용하여 효율적으로 병렬처리하기 위한 클라우드 기술이 많이 사용된다[2]. 다수의 서버들을 이용하여 병렬 처리를 하게 되면 엄청난 양의 빅데이터를 효율적으로 빠른시간에 처리할 수 있다. 이러한 성능을 필요로 하는 빅데이터 처리에서 네트워크로 인한 병목현상이 발생하지 않기 위해 고성능 네트워크의 고려는 필수적이다[3].

이러한 고성능 네트워크에서 공통적으로 채택하는 기술로 RDMA(Remote Direct Memory Access) 기술이 존재한다. RDMA 기술은 고성능 네트워크를 통해 메모리 간에 데이터를 전송한다[4]. CPU를 사용하지 않고 원격 노드의 메모리로 직접 데이터를 전송하여 CPU의 사용률을 낮추고, CPU의 간섭 없이 데이터를 전송하기에 고속의 데이터 전송이 가능한 기술이다. 이러한 RDMA 기반의 고성능 네트워크로 인피니밴드, RoCE(RDMA over Converged Ethernet), iWARP 등이 있다. 최근 동향에서는 인피니밴드가 성능이 좋아서 주로 사용된다. 실제로 TOP500 슈퍼컴퓨터에서 인피니밴드 기술의 점유율을 확인할 수 있다. TOP500 슈퍼컴퓨터는 매년 2회에 걸쳐 전세계에서 가장 성능이 좋은 컴퓨터 시스템의 성능 순위를 500위까지 메긴 것이다. 2017년 6월 조사 결과에 따르면 500대의 슈퍼컴퓨터 중 35.4%가 인피니밴드를 사용하고 이더넷 기술은 41.6%를 차지한다. 하지만 성능부문 상위권 100개의 슈퍼컴퓨터에서 이더넷 기술은 1%의 비율을 차지하고 인피니밴드는 42%의 비율을 차지하여 성능부문 상위권의 슈퍼컴퓨터에서 인피니밴드가 이더넷보다 많이 채택되고 있는 것을 알 수 있다[5].

본 논문에서는 RDMA 기반의 대용량 고성능 데이터 처리 네트워크 기술들에 대해서 알아보고, RDMA 네트워크를 사용하기 위한 프레임워크/라이브러리를 소개한다. 최근에 이루어지는 고성능 컴퓨팅을 위한 클러스터 환경에서 인피니밴드를 적용하는 기법들과 성능 분석에 대한 최근 연구 동향에 대해서 정리하고자 한다. 또한 대용량 데이터스트림 처리를 위해 우리가 진행 중인 실시간 분산형 데이터 처리 플랫폼 스톰에 인피니밴드 기술을 적용하는 연구를 소개하고자한다. 2장에서는 RDMA 프로토콜과 VIA 기술에 대해서 설명하고, 3장에서는 RDMA 기반의 네트워크 기술인 인피니밴드, RoCE, iWARP에 대해서 알아보고, 4장에서는 RDMA 프레임워크, 라이브러리에 대해서 소개한다. 5장에서는 대용량 데이터 처리를 위해 인피니밴드를 적용하는 최신 연구들과 우리가 진행하는 연구에 대하여 소개하겠다. 마지막으로 6장에서 결론으로 본 논문을 맺고자 한다.

Ⅱ. VIA 구조 모델

본 장에서는 인피니밴드, RoCE, iWARP 기술의 기반이 되고 RDMA 기능을 지원하는 네트워크 추상 모델 VIA(Virtual Interface Architecture)[6]에 대하여 설명하고 RDMA 오퍼레이션에 대하여 설명한다. 기존의 네트워크에서는 네트워크 프로토콜 스택들이 호스트의 커널 단에서 소프트웨어적으로 구현되어져 운영이 되었다. 이는 결과적으로 CPU에게 부하로 작용되었고, 대용량 데이터를 고속으로 처리할 필요가 있는 시점에서 네트워크 병목현상을 야기하는 문제점이 되었다. 대역폭, 지연시간, CPU 부하 등의 문제점에 대한 성능향상을 위해서 VIA가 제안되었다. VIA는 원격 노드의 메모리에 직접 접근하는 RDMA 기술과 버퍼 간의 복사를 최소화하는 Zero-Copy 기능을 지원하여 성능을 높이는 네트워크 추상모델이다. 1997년

12월 18일에 Microsoft, Intel, Compaq에서 발표한 VIA 1.0 모델에서 RDMA 기술이 처음 소개되었다. RDMA(Remote Direct Memory Access) 기술은 고성능 네트워크를 통해서 각 노드에서 메모리 대 메모리에 직접적으로 데이터를 전송하는 기술이다. 이러한 VIA 추상 모델을 구현한 기술이 인피니밴드, RoCE, iWARP이다.

그림 1은 VIA 구조 모델을 보여준다[6]. VIA 구조는 VI(Virtual Interface), CQ(Completion Queues), VI Provider, VI Consumer의 네 가지 기본 요소로 이루어진다. VI Provider는 물리적 네트워크 하드웨어(VI Network Adapter)와 소프트웨어적으로 구현된 Kernel Agent로 구성되어 있다. VI Consumer는 Socket이나 MPI를 통해서 통신이 가능한 OS 통신 인터페이스와 에플리케이션, VI User Agent로 구성되어 있다. VI Consumer가 데이터 송수신 오퍼레이션을 전달하면 VI Provider가 오퍼레이션을 처리하고 CQ를 통해 요청에 대한 완료 알림을 VI Comsumer에게 전달한다. VI Consumer에서 연결에 대한 제어 오퍼레이션은 User Agent에서 Kernel Agent로 전달하지만 데이터 송수신 오퍼레이션은 CPU 부하를 줄이기 위하여 운영체제 커널을 거치지 않고 VI를 통해 직접 Consumer의 User Agent에서 Provider의 Network Adapter에게 바로 전달하여 처리한다.

VIA의 데이터 전송 모델은 기존의 Send/Receive 메시징 모델과 RDMA 모델이 존재한다. Send/Receive 메시징 모델은 기존의 소켓 통신 모델과 같이 데이터 패킷을 보내거나 받는 오퍼레이션이다. RDMA 모델은 로컬 노드의 메모리 영역에서 원격 노드의 메모리 영역으로 데이터를 전송하거나 전송받는다. 데이터 전송 오퍼레이션은 RDMA Write와 RDMA Read의 두 가지 타입이 존재한다. RDMA Write는 VI Consumer가 로컬 메모리 영역에서 원격의 메모리 영역에 데이터를 전송하는 것이다. 반대로 RDMA Read는 VI Consumer가 원격의 메모리 영역 데이터를 로컬 메모리 영역에 전송받는 오퍼레이션이다.

Ⅲ. 인피니밴드, RoCE, iWARP

RDMA 기반의 고성능 네트워크 기술에는 인피니밴드, RoCE, iWARP 등이 있다. 인피니밴드, RoCE, iWARP는 모두 VIA를 기초로 확대된 기술로 높은 대역폭, 낮은 지연시간, RDMA 기법 등의 장점이 있다. 인피니밴드와 RoCE는 IBTA(InfiniBand Trade Association)에서 표준화를 주관[7]하고 있고 iWARP는 IETF에서 표준화를 주관한다. 이 장에서 각각의 기술에 대해 자세히 알아본다.

3.1 인피니밴드

인피니밴드는 높은 대역폭과 낮은 지연시간 등의 특징으로 고성능 컴퓨팅 등에서 사용하는 통신 네트워크 표준이다. 인피니밴드는 소규모 서버 시스템에서부터 대규모 인터넷 서버 및 슈퍼컴퓨터에 이르기까지 다양한 영역에서 사용가능하다. 초기에는 Compaq, Dell, HP, IBM, Intel, Sun, Microsoft에서 IBTA를 결성하여 표준화를 이끌었으며 2003년에 업계표준으로 인피니밴드 명세서 1.1버전을 공개하였다[7]. 2008년에 인피니밴드 1.2버전이 공개되었고, 2012년에 1.3버전이 공개되었으며, 가장 최근 버전은 2016년에 공개된 1.3.1 버전이다.

그림 2는 인피니밴드의 평면적인 네트워크 구성도

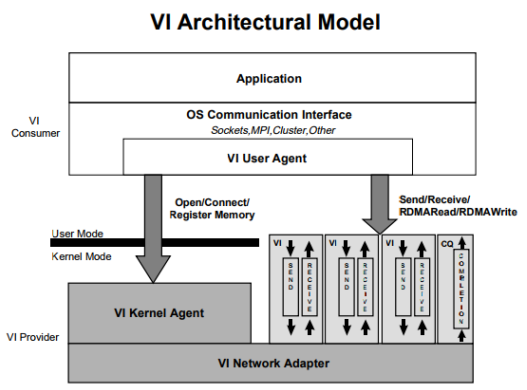

그림 1. Virtual Interface Architecture Model
Fig. 1. Virtual Interface Architecture Model

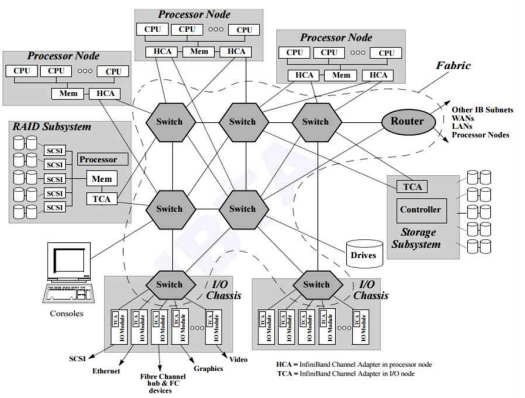

그림 2. 인피니밴드 구성도
Fig. 2. Infiniband Architecture

를 보여준다[8]. 인피니밴드 기반의 SAN(System Area Network)은 계산 노드(Processor Node), I/O 장치(I/O Chassis, RAID Subsystem, Storage Subsystem)들이 인피니밴드 스위치와 라우터에 패브릭(Fabric) 방식으로 연결되어진다. 기존의 계층적 구조의 이더넷은 서버간의 통신에서 상위 스위치 장비를 거쳐 수직적으로 패킷이 전달되었다. 이러한 계층적 구조의 문제점은 패킷이 동에서 서로 이동하려면 최상위층 스위치를 거쳐 불필요한 트래픽을 유발하는 것이다. 패브릭 스위칭 구조는 그림 2와 같이 노드들이 서로 긴밀히 연결되어 패킷이 수평적으로 전달된다. 또한, 모든 네트워크 환경을 통합적으로 관리하여 네트워크 인프라를 단순화하고 비용절감을 실현한다. 개념적으로 네트워크 인터페이스 카드는 계산 노드의 HCA(Host Channel Adapter)와 I/O장치 내의 TCA(Target Channel Adapter)로 나뉜다. 계산 노드의 HCA가 I/O 장치의 TCA에게 오퍼레이션을 요청하면, I/O 장치의 TCA에서 RDMA Write/Read 등의 오퍼레이션이 수행된다. 예를 들면, 계산 노드가 I/O 장치의 메모리 영역에 데이터를 쓰려고 하면 계산 노드의 HCA가 Storage Subsystem의 TCA에게 RDMA Write 오퍼레이션을 보내게 되고, 쓰고자하는 계산 노드의 데이터가 Storage Subsystem 내에 저장되게 된다. 이 때, HCA와 TCA 사이에서 오퍼레이션을 수행하기 위해 발생한 데이터 패킷들은 패브릭 네트워크를 통해 계산 노드의 메모리와 I/O 장치의 메모리 영역 간에 전달된다.

지금까지 인피니밴드의 평면 패브릭 네트워크 구조에 대해서 살펴보았다면 이제부터는 인피니밴드의 프로토콜 스택에 대해서 설명하고자 한다. 그림 3은 인피니밴드의 프로토콜 계층을 보여준다[8]. 인피니밴드 계층은 물리 계층, 링크 계층, 네트워크 계층, 전송 계층, 상위 계층의 5계층으로 이루어져있다. 물리 계층은 그림 2의 각 컴포넌트 간에 비트 단위로 데이터를 전송하는 역할을 담당하며, 시그널링 속도에 따라 대역폭이 결정된다. 네트워크 인터페이스 카드는 노드와 PCI-Express 커넥터를 통해 연결되어 데이터의 손실 없이 높은 대역폭을 유지하게 한다. 전송 매체인 케이블은 구리선과 광선이 존재하며 케이블과 네트워크 인터페이스 카드 사이에는 QSFP+(Quad Small Form-factor Pluggable) 또는 CXP 규격의 소형 트랜시버를 사용한다.

링크 계층은 링크 간의 흐름 제어와 데이터 전송을 담당한다. 이더넷은 링크 간의 흐름 제어를 제공하지 않아 스위치 버퍼에서 오버플로우로 인해 패킷 손

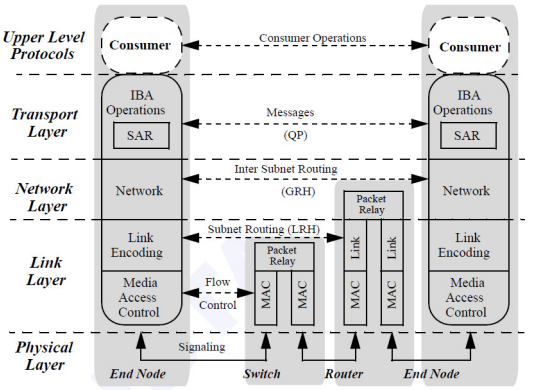

그림 3. 인피니밴드 프로토콜 스택
Fig. 3. Infiniband protocol stack

실이 발생할 수 있으나, 인피니밴드는 링크 간의 흐름 제어 기능을 제공하여 패킷 손실을 예방할 수 있다. 링크 계층의 패킷 종류에는 데이터 패킷과 링크 관리 패킷이 존재한다. 링크 관리 패킷은 두 개의 물리 포트 사이의 전송률, 링크 대역폭, 흐름 제어 신뢰도 등의 파라미터 정보를 포함하여 링크를 관리하고 유지하는 역할을 수행한다. 서브넷 내에서의 라우팅을 위해 MAC기반의 LID(Local Identifier) 주소를 사용하며 패킷의 헤더인 LRH(Local Route Header)에 출발지와 목적지 LID가 담겨진다[9].

네트워크 계층은 전송 계층에서 생성된 패킷의 헤더 정보를 사용하여 경로를 설정하고 서브넷 간의 라우팅, 즉 글로벌넷 라우팅을 가능하게 해준다. 네트워크 계층에서 만들어진 패킷은 패킷의 헤더인 GRH(Global Route Header)에 IPv6 기반의 출발지와 목적지 GID(Global Identifier) 주소 정보 등이 담겨져서 전송된다. 링크 계층과 네트워크 계층에서 서브넷 매니저는 LID/GID 패킷을 설정하거나 경로 설정, 포워딩 테이블 구성의 역할을 수행한다. 또한, 서브넷 매니저는 SDN(Software Defined Network)의 개념을 도입하여 제어 평면을 소프트웨어적으로 분리하여 데이터센터 적용에 맞게 최신 패러다임을 따르고 있다.

전송 계층은 상위 계층에 제공할 서비스 타입을 결정하고 무결성 검사, 연결 관리, 혼잡 제어 등의 기능을 제공한다. 서비스 타입은 RC(Reliable Connection), RD(Reliable Datagram), UD(Unreliable Datagram), UC(Unreliable Connection), XRC(eXtended Reliable Connection) 등의 5가지 타입이다. RC, RD, XRC는 신뢰 서비스로 패킷이 에러나 손실이 없게 체크하고 재전송을 수행하여 신뢰성을 보장한다. UD, UC는 신뢰성을 보장할 필요가 없어 패킷의 손실이 발생할 가

능성이 있지만 지연시간이 작은 장점이 있다. RC, UC, XRC는 로컬 QP(Queue Pair) 하나에 원격지 QP 하나가 연결이 되어 통신한다. 이와 반대로, RD, UD는 연결을 맺지 않고 하나의 로컬 QP로 다수의 원격지 QP와 통신을 한다. Send work queue, Receive work queue, Completion Queue로 이루어진 QP 버퍼는 상위 계층의 에플리케이션에게 제공하는 가상의 인터페이스로 이더넷의 소켓과 비슷한 개념이다. Send work queue는 로컬 노드의 상위 에플리케이션에서 발생한 데이터 송수신 오퍼레이션을 저장하고, Receive work queue는 다른 노드의 상위 에플리케이션에서 로컬 노드에게 지시한 데이터 송수신 오퍼레이션을 저장한다. 네트워크 인터페이스 카드는 Send work queue와 Receive work queue로부터 오퍼레이션을 가져와 처리하고, Completion queue에 완료된 오퍼레이션을 반환한다. 상위 에플리케이션은 Completion queue로부터 지시한 오퍼레이션이 완료되었음을 알 수 있다.

다음으로 인피니밴드가 제공하는 성능에 대해서 살펴보고자 한다. 그림 4는 IBTA에서 공개한 인피니밴드 성능 개발 로드맵[10]이다. 로드맵에 따르면 IBTA는 2001년에 SDR(Single Data Rate), 2005년에 DDR(Double Date Rate), 2007년에 QDR(Quad Data Rate), 2011년에 FDR(Fourteen Data Rate), 2014년 EDR(Enhanced Date Rate)을 발표하였다. 전송 대역폭은 시그널링 속도와, 링크 수로 결정된다. 링크 수는 케이블 자체의 와이어 수를 의미한다. 1x는 와이어 수 4개를 의미하고, 4x는 16개의 와이어를 의미하며, 12x는 와이어 수가 48개이다. 가장 최근 인피니밴드 시그널링 대역폭은 HDR(High Data Rate)로 4x기준으로 대역폭이 200Gbps이다. 즉, HDR은 1x 당 대역폭이 50Gbps이며 와이어가 12x라면 600Gbps까지 대역폭이 가능하다. 인피니밴드 제품 제조사로는 멜라녹스와 Qlogic을 인수한 인텔이 있다. 멜라녹스가 인피니밴드 제품에 관해서는 선두 주자로 시장을 거의 독점하고 있으며 2017년에는 200Gbps 대역폭의

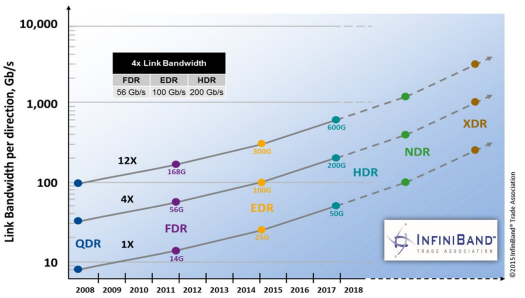

그림 4. IBTA 성능 로드맵
Fig. 4. IBTA Performance Roadmap

HDR 성능을 가진 제품을 출시하고 있다. 이외에도 2016년에는 오라클이 자체적으로 인피니밴드 칩과 스위치를 제조하여 자사의 Exadata 데이터베이스와 Exalogic 미들웨어 클러스터 등에 적용하였다[11].

3.2 RoCE

RoCE(RDMA over Converged Ethernet)[12]는 이더넷 기반에서 동작 가능한 인피니밴드 프로토콜 기반 기술로 RDMA 기능을 지원한다. IBTA가 배포한 RoCE는 여러 곳에서 이미 구축된 이더넷 네트워크를 기반으로 RDMA 기술을 적용한 고속의 데이터 처리가 가능하다는 장점이 있다. IBTA는 RoCEv1 명세서를 2010년에 배포하였고 2014년에 RoCEv2 명세서를 배포했다. RoCEv1은 네트워크 계층까지 명세하고 있지만, RoCEv2는 전송 계층까지 명세하고 있다. 즉, v2는 IP를 사용하기 때문에 서브넷 간의 통신을 위한 이더넷 기반의 라우팅이 가능하다.

그림 5는 RoCEv2의 패킷 포맷을 보여준다[12]. Ether/Type값은 네트워크 계층의 프로토콜을 의미하며, RoCEv1에서는 0x8915를 갖지만, v2에서는 일반 이더넷과 똑같이 IPv4이면 0x0800, IPv6이면 0x86DD 값을 갖는다. IP Header와 UDP Header는 일반 이더넷과 같은 형식을 갖으며, 이는 RoCE가 IP, UDP 기반에서 동작하기 때문이다. IP Header에는 IPv4 또는 IPv6의 정보를 담을 수 있다. UDP Header

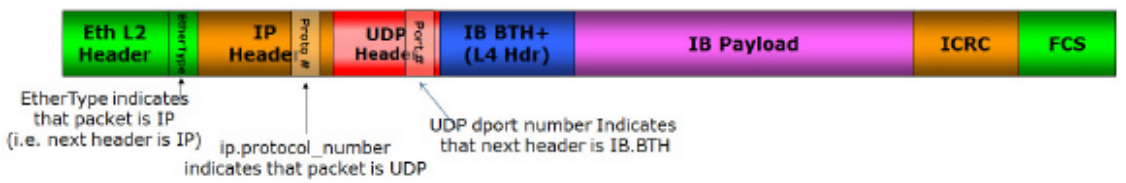

그림 5. RoCEv2 패킷 포맷
Fig. 5. RoCEv2 Packet Format

에는 출발지 포트 주소와 목적지 포트 주소가 담긴다. ICRC(Invariant Cyclic Redundancy Check)는 IP 계층에서 종단간의 패킷 무결성을 검증하기 위한 필드이고 FCS(Frame Check Sequence)는 링크 계층에서 종단간의 패킷 무결성을 검증한다.

그림 6은 인피니밴드 프로토콜 스택, RoCE 프로토콜 스택, RoCEv2 프로토콜 스택을 보여준다[12]. RoCE, RoCEv2는 인피니밴드에서 동작하는 API(Verbs)들을 이더넷 네트워크에서 사용할 수 있게 해준다. 이에 따라 인피니밴드 기반에서 사용하는 RDMA 에플리케이션이 큰 수정 없이 이더넷 네트워크 기반의 RoCE, RoCEv2에서 동작이 가능하다. 즉, RoCE 프로토콜을 통해 RDMA 에플리케이션들이 기존의 이더넷 네트워크 환경에서 수행이 된다는 것이 RoCE 프로토콜의 가장 큰 장점이다. RoCEv1은 네트워크 계층은 인피니밴드 프로토콜을 사용하고, 링크 계층은 기존의 이더넷 표준을 따르기 때문에, 기존의 이더넷 환경에서의 서브 네트워크 내부 통신은 가능하지만 서브넷 간의 통신은 불가능하다. 그러나, RoCEv2는 그림 6과 같이 UDP, IP 프로토콜을 사용하기 때문에 기존의 구축된 이더넷 네트워크에서 서브넷 간의 통신이 가능하다. 이러한 RoCE, RoCEv2는 인터페이스 카드를 하드웨어적으로 구현하여 CPU 사용률을 낮추었다. 더 나아가서 멜라녹스는 소프트웨어적으로 구현하는 Soft-RoCE를 개발하여 표준 이더넷 네트워크 인터페이스 카드 기반에서도 RDMA 에플리케이션 동작이 가능하게 하였다.

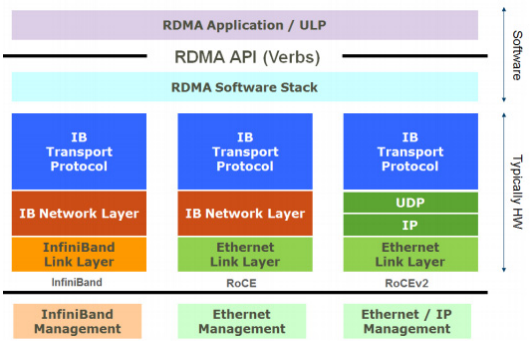

그림 6. 프로토콜 스택 비교(인피니밴드 vs RoCE)
Fig. 6. Protocol Stack Comparison(Infiniband vs RoCE)

3.3 iWARP

iWARP(Internet Wide Area RDMA Protocol)는 TCP 환경에서 RDMA 기능을 지원하는 고성능 네트워크 기술로 이더넷 네트워크 기반에서 효율적인 데이터 전송을 수행한다. iWARP는 TCP 환경에서 동작하나, RoCE는 UDP 기반으로 동작한다는 점에서 iWARP와 다르다. 또한 RoCE는 전송 계층을 인피니밴드와 공유하여 같지만, iWARP는 인피니밴드 프로토콜과는 별개로 자체적으로 프로토콜을 명세하였다. IETF는 2007년에 iWARP를 명세하는 여러 RFC 문서들을 발표하였다. RFC 5040은 DDP(Direct Data Placement) 위에서 동작하는 RDMAP(Remote Direct Memory Access Protocol)에 대한 사양서이다. DDP는 RFC 5041에서 명세한 프로토콜로 상위 계층의 버퍼에 중간 버퍼 없이 데이터를 바로 전송하게 해주는 프로토콜로 CPU 사용률과 메모리 점유율을 낮춘다. RFC 5043은 차세대 전송 프로토콜 SCTP(Stream Control Transmission Protocol) 위에서 DDP의 동작을 가능하게 해주는 적응 계층을 명세한다. RFC 5044는 기존의 전송 프로토콜 TCP 기반으로 DDP를 동작하게 해주는 MPA(Marker PDU Aligned) 프로토콜을 명세한다. RFC 5042는 DDP와 RDMAP에 보안 관련 이슈에 대한 내용을 포함하고 있다. RFC 6580, RFC 6581은 2012년, RFC7306은 2014년에 iWARP 기술의 기능 확장 문서가 추가로 발표되었다. RFC 6580은 iWARP에서 사용할 에러 코드, 오퍼레이션 코드, 함수 코드에 대한 값들을 정의한다. RFC 6581은 RFC 5043과 RFC 5044에서 RDMA 단의 연결 설정 등의 기능을 추가하고 기타 사양을 업데이트한 MPA 프로토콜을 명세한다. RFC 7306은 RFC 5040에서 설명한 RDMAP 프로토콜을 확장하여, 동기화 오퍼레이션, 데이터 중재 오퍼레이션 등의 기능을 추가하였다.

iWARP는 VIA를 확장한 개념으로 RDMA 기능을 지원하며, zero-copy 전송을 가능하게 해준다. 앞서 언급했듯이 zero-copy는 버퍼간의 복사를 최소화하여 CPU의 사용률을 낮추는 방법이다. 기존의 이더넷은 TCP/IP 계층이 커널에서 소프트웨어적으로 구현되어서 패킷을 나누거나 합치는 등의 작업이 CPU에게 부하로 작용한다. 이러한 CPU 부하 문제점을 해결하기 위해 TCP/IP 계층을 하드웨어적으로 구현하는 기술이 TCP Offload Engine(TOE)이다. iWARP는 TOE 기술에 원격 노드의 메모리에 직접 접근하는 RDMA 기술이 결합된 것으로 2017년 기준 100GbE(Gigabit Ethernet) 대역폭 성능을 지원한다.

그림 7은 iWARP 프로토콜 스택을 나타낸다. iWARP는 IP/Ethernet과 ULP(Upper Layer Protocol) 사이에 존재하며 RDMAP, DDP, MPA, TCP, SCTP 등이 하드웨어적으로 구현된다. RDMAP는 RDMA Write, RDMA Read 등의 RDMA 오퍼레이션을 통해

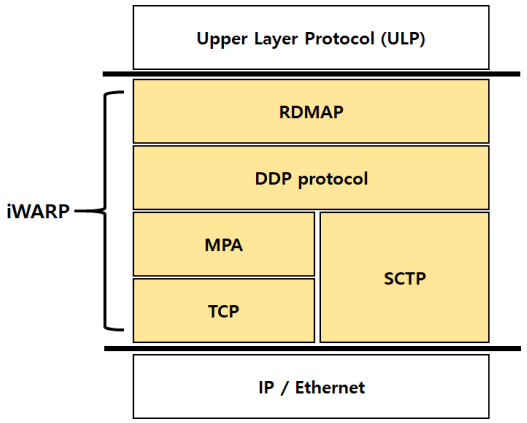

그림 7. iWARP 프로토콜 스택
Fig. 7. iWARP protocol stack

상위 계층의 버퍼에 직접 접근하거나, 데이터를 DDP(Direct Data Placement) 메시지로 변환하는 프로토콜이다. DDP는 DDP 메시지를 세그먼트로 나누거나 합치는 프로토콜로 전송에 대한 규약이 없다. 따라서, 기존의 전송 프로토콜인 TCP 또는 차세대 전송 프로토콜인 SCTP(Stream Control Transmission Protocol)를 사용한다. SCTP는 한 연결에서 다중 스트리밍 서비스를 제공하는 프로토콜로 DDP와 SCTP 사이에 적응 계층이 존재하고 신뢰 DDP 스트림 통신을 DDP 계층에게 제공하여 통신이 가능하게 한다. MPA는 TCP와 DDP의 호환성을 제공하여 데이터 전송 시 버퍼링을 줄이고 성능을 개선하기 위한 프로토콜이다.

iWARP는 TCP/IP 프로토콜을 기존의 커널 단에서 구현하지 않고 네트워크 인터페이스 카드에서 하드웨어적으로 구현함으로써 CPU 부하를 줄일 수 있었다. iWARP 기술은 운영체제를 거치지 않고 에플리케이션 버퍼에서 네트워크 인터페이스 카드 단의 버퍼로 데이터를 바로 전송하는 Zero Copy 기능이 가능하고 I/O 명령을 운영체제를 거치지 않고 I/O 장치에 직접 내린다[13]. iWARP 기반의 네트워크 인터페이스 카드는 Chelsio라는 회사와 Intel 회사에서 주로 만들고 2017년에 100GbE의 속도를 지원하는 NIC 네트워크 인터페이스 카드를 출시해놓았다.

Ⅳ. RDMA 라이브러리

인피니밴드, RoCE, iWARP 등의 RDMA 기술을 지원하는 고성능 네트워크 기술은 OFED (Open Fabrics Enterprise Distribution) 미들웨어에서 제공하는 API를 통해 프로그램 레벨에서 RDMA 기술을 사용한다. 그러나 OFED에서 제공하는 API는 노드간의 연결을 위한 사전 설정 함수, RDMA 오퍼레이션을 위한 메모리 영역 설정 함수들의 사용이 복잡하다. 이렇게 복잡한 OFED API를 개발자가 개발하기 쉽게 추상화한 API로는 Accelio, DiSNI 등이 있다. 본 장에서는 OFED, Accelio, DiSNI에 대해서 살펴보고 국내 연구에서 제안한 경량 RDMA API를 알아본다.

OFED는 RDMA 에플리케이션을 사용하기 위한 오픈소스 미들웨어이다. OFED에는 드라이버, RDMA 오퍼레이션, 유저 단의 API, MPI 등을 포함하고 있다. 즉, OFED는 에플리케이션 단에서 RDMA API를 제공해주며 효율적인 네트워크 사용, 연결성, 병렬 처리를 가능하게 해준다. RDMA 기반의 에플리케이션을 만들 때 User API 단에 제공하는 API 또는 Verbs라는 함수들을 사용한다. InfiniBand, RoCE, iWARP는 모두 OFED의 ULP(Upper Layer Protocol)를 통해 제어 및 통신을 수행한다. 특히 많이 사용하는 IPoIB(IP over InfiniBand)는 기존의 이더넷 기반의 에플리케이션을 큰 소스코드 수정 없이 인피니밴드 네트워크 환경에서도 쉽게 사용가능하게 해주는 프로토콜로 소켓 통신을 한다. 또한, SCSI 인터페이스를 RDMA 기반으로 지원하기 위한 프로토콜로 SRP, iSER 등이 존재한다. OFED API는 레드햇 리눅스, 수세 리눅스, 오라클 리눅스뿐만 아니라 윈도우 서버까지 다양한 운영체제에서 실행 가능하며, OFED는 legacy 10 Gigabit 이더넷 환경에서도 사용이 가능하다.

Accelio[14]는 인피니밴드를 제작하는 멜라녹스 회사가 오픈소스로 배포한 인피니밴드 기반 RPC 라이브러리이다. Accelio는 GPL 2.0 라이센스로 제공하는 오픈소스 프로젝트로 비동기적으로 메시지를 전송하는 RPC API를 제공한다. 그림 8은 Accelio의 구조이다[14]. Accelio API는 OFED 등의 RDMA 네트워크 환경에서도 동작이 가능할 뿐만 아니라 그림 8과 같

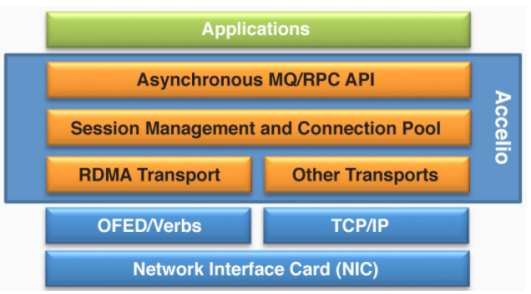

그림 8. Accelio 구조
Fig. 8. Accelio Structure

이 기존의 TCP/IP 네트워크기반에서도 동작이 가능하다. Accelio API는 기존의 OFED API에서 연결 설정, 세션 관리, 메모리 영역 설정 등의 복잡한 부분을 쉽게 제공하는 연결 세션 관리 및 연결 풀 등의 기능들을 지원하여 개발자 입장에서는 복잡한 OFED API보다 사용하기 쉽고 종단 간의 신뢰성을 보장한다. 비동기 MQ/RPC API에서는 Send/Receive 모델과 비동기식의 Request/Reply 모델을 제공하고 전송할 때에 Zero Copy가 가능하게 해준다. 또한, RDMA, TCP, Shared-Memory 등의 여러 전송 기술을 지원한다. 기본적으로 C 라이브러리로 제공되며 JNI(Java Native Interface)를 통해 Java에서도 사용가능하다.

DiSNI(Direct Storage and Networking Interface)[15]는 스토리지를 RDMA 인터페이스로 직접 접근하는 Java 기반의 프레임워크/라이브러리이다. 이 프로젝트는 에플리케이션 단에서 스토리지나 네트워크에 직접 접근하게 해주는 API들을 제공한다. 그림 9는 DiSNI 프레임워크의 구조를 보여준다[15]. 그림 9의 libibverbs, librdmacm은 OFED에서 제공하는 RDMA 라이브러리이다. 이 라이브러리를 통해 DiSNI는 추상화된 API를 Java 에플리케이션에게 제공하게 된다. Control path는 연결에 대한 설정이나 오퍼레이션에 관한 제어 정보들이 오가는데 사용되고, 실제적인 데이터 통신을 빠르게 수행하기 위해서는 past path가 이용된다. DiSNI는 인피니밴드, iWARP, RoCE 기반에서 모두 동작 가능하다.

국내에서는 전자부품연구원 IoT융합연구센터에서 인피니밴드를 위한 RDMA API를 설계 및 구현하였다[16]. 기존의 소켓 프로그램과는 다르게 MR (Memory Region), QP(Queue pair) 등의 세부적인 설정을 요구하는 함수들이 필요하다. MR은 데이터가 직접적으로 저장될 메모리 영역을 지정해주는 역할이고, QP는 RDMA 전송에서 노드와 노드 사이에서 데이터 전송하는 엔진이다. MR, QP 등의 세부적인 설정이 필요하여 복잡한 RDMA API들로는 단기간에 RDMA 에플리케이션을 개발하기가 힘들다. 그래서 [16] 연구에서는 성능상의 장점을 유지하면서 단기간에 에플리케이션을 개발하기 쉬운 기존 소켓 API와 유사한 형태의 경량 RDMA API 설계 및 구현하였다. 그림 10은 [16] 연구에서 설계 및 구현한 경량 RDMA API 프로그램 순서도이다. lw_create_id(), lw_bind(), lw_listen() 함수를 통해 통신을 준비하고, lw_accept() 함수와 lw_connect()함수를 통해 connection을 만든다. lw_send(), lw_recv() 함수를 통해 RDMA 통신을 수행하고 통신이 끝나면 lw_close() 함수를 호출한다. Rsocket은 인텔이 개발한 RDMA 기반의 소켓 API로 제안한 경량 RDMA API와 성능 비교를 통해 검증을 수행하였다. 데이터 전송 사이즈가 1~512 Byte 일 때 데이터 전송 지연시간은 제안한 경량 RDMA API보다 Rsocket이 약 21% 성능이 더 좋은 것을 확인할 수 있지만, 데이터 전송 사이즈가 1KByte ~ 512KByte에서는 경량 RDMA API가 약 12% 정도로 성능이 더 좋다는 결과를 보였다.

Accelio API는 멜라녹스에서 제조한 인피니밴드 장비에 최적화되었으며, C언어 기반으로 비동기 통신을 지원한다. 이와는 달리 DiSNI는 IBM에서 제조한 인피니밴드 장비에 최적화되었으며, JAVA언어 기반으로 동작한다. 또한 스토리지 시스템을 효율적으로 지원한다는 특징이 있다. OFED Verbs API는 메모리

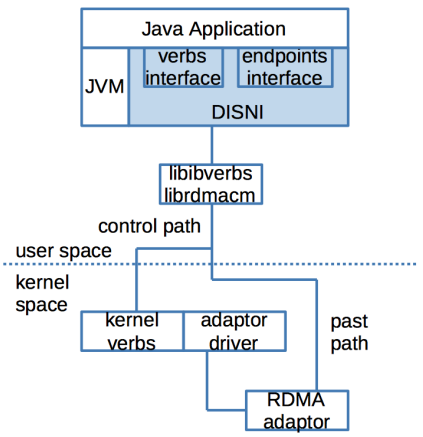

그림 9. DiSNI 프레임워크 구조
Fig. 9. DiSNI Framework Structure

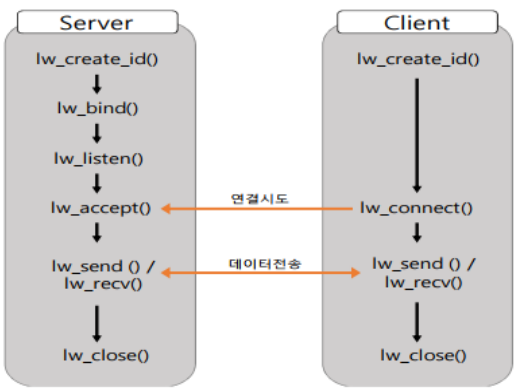

그림 10. 경량 RDMA API 프로그램 순서도
Fig. 10. Flowchart of lightweight RDMA API program flowchart

영역 설정, 버퍼 설정 등의 더 낮은 레벨의 API를 지원한다. 이로 인해 API를 사용함에 있어서 세세한 컨트롤이 가능하지만 개발자 입장에서 사용이 복잡하다는 단점이 있다.

V. 인피니밴드 최신 연구 사례

인피니밴드 기술은 전술하였듯이 2003년에 최초로 1.1 버전으로 표준이 명세 되었고 현재 IBTA에서 주관하고 있는 고성능 네트워크 기술이다. 현재, 인피니밴드 기술은 고성능의 계산을 위한 슈퍼컴퓨터, 고용량 데이터 저장과 처리를 위한 데이터센터, 인피니밴드를 이용한 고효율의 병렬처리 기계 학습, 클라우드 등에서 대표적으로 쓰인다. 빅데이터 기술이 이슈화되고 있음에 따라 급격히 증가하는 데이터를 고성능으로 저장하고 처리할 것인지가 중요해지고 있다. 본 장에서는 빅데이터 처리 및 데이터센터에 적용 가능한 인피니밴드 연구 사례들을 선별하여 정리하였다. 본 장에서 언급하는 연구 사례들은 크게 스토리지 관련 유형, API 관련 유형, 가상화 관련 유형들로 나누어 볼 수 있다. 인피니밴드를 사용하여 고성능 데이터 처리를 수행한 국내외 연구 사례를 요약정리하고, 우리가 수행한 인피니밴드 적용 사례에 대해서 간략히 소개하고자 한다.

5.1 국내 연구 사례

인피니밴드 기반의 고성능 클러스터를 위한 효율적인 데이터 선반입(pre-fetching) 기법에 대해 제안하는 연구 사례가 있다[17]. 기존에는 데이터를 디스크 기반으로 저장하였지만 최근에는 다수의 컴퓨팅 노드에서 Redis, Memcached와 같은 인-메모리 기반의 키-값 저장소를 활용하여 데이터를 분산 저장하고 있다. 이를 통해 데이터를 고속으로 접근하고 데이터를 처리하여 클러스터 시스템의 처리 성능을 높인다. 이 제안에서는 고성능 클러스터 구현에 인피니밴드를 적용한 것이 더하여, 인피니밴드 네트워크 기술의 송수신 데이터 크기별 전송지연 특성을 활용한 데이터 선반입 기법을 제안하고 있다. 제안 기법은 키-값 저장소에서 데이터 접근시 클러스터를 구성하는 컴퓨팅 노드의 접근 패턴을 고려하여 추후 참조 가능성이 큰 다수의 키-값 데이터 쌍을 동시에 선반입 함으로써 데이터 송수신에 필요한 소요시간을 줄인다. 이렇게 데이터 송수신에 필요한 소요시간을 줄임으로 전체 클러스터 컴퓨팅 시스템의 데이터 처리 성능을 향상시킬 수 있었다. 앞서 제안한 데이터 선반입 기법을 적용한 시뮬레이션에서 평균 데이터 전송 소요시간을 최대 28% 이상 줄 일 수 있음을 확인하였다.

이외에도 인피니밴드 스토리지 네트워크를 적용하는 오픈스택 클라우드 스토리지 시스템의 설계를 제안하고 스토리지 가상화 성능 평가를 진행하는 연구 사례가 있다[18]. 오픈스택은 오픈소스 클라우드 컴퓨팅 플랫폼으로써 컴퓨팅 서비스를 제공하는 Nova, 오브젝트 스토리지 서비스를 제공하는 Swift, 이미지 서비스를 제공하는 Glance, 인증 서비스를 제공하는 Keystone, 네트워크 서비스를 제공하는 Neutron, 블록 스토리지 서비스를 제공하는 Cinder 등의 서비스들로 구성되어 있으며, Management network를 통해 서비스 컴포넌트들을 연동하여 클라우드 서비스를 제공한다. Cinder는 하이퍼바이저를 통해 서버를 가상화하고 가상화된 자원에 대한 컴퓨팅 작업을 진행하고 Cinder는 LVM(Logical Volume Manager)을 통해 여러 개의 물리 디스크들을 하나의 볼륨 그룹으로 묶어 블록 스토리지로 사용하고, 이 블록 스토리지들은 가상화되어 사용자에게 제공된다. 기존에는 블록 스토리지 가상화를 구현하기 위해서 HDD와 기가비트 이더넷을 사용하였다. 그러나 고성능 스토리지 SSD를 사용하여 스토리지 시스템을 구축하는 경우 네트워크의 낮은 대역폭으로 병목현상이 발생할 수 있다. 이에 따라 대역폭이 높고 RDMA 기법으로 낮은 지연시간으로 데이터를 처리하는 인피니밴드를 적용하여 병목현상을 제거하였다. 그림 11은 인피니밴드 스토리지 네트워크를 이용하여 구성한 RAID-60 스토리지의 구조를 보여준다[18]. 인피니밴드 스토리지 프로토콜로 iSER(iSCSI Extensions for RDMA)을 사용하였다. iSER은 SCSI 타겟과 SCSI 이니시에이터 사이에서 RDMA Write, Read 명령을 전달하여 스토리지 입출력을 수행한다. Cinder 스토리지 노드에 장착된 8개의 스토리지를 RAID-6 구조로 묶어 구성한 뒤, 이를 인피니밴드를 통해 가상화하였다. 가상화된 3개의

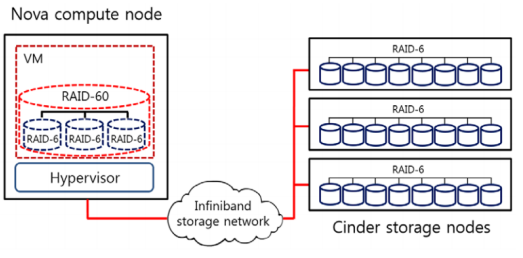

그림 11. 인피니밴드 스토리지 네트워크를 이용하여 구성한 RAID-60 스토리지
Fig. 11. RAID-60 Storage Structure using InfiniBand Storage Networks

RAID-6 Flash array를 하이퍼바이저를 통해 가상머신에 할당하고 할당된 3개의 Flash array에 RAID-0 구조 기반 클라우드 스토리지를 구현하였다. 제안한 RAID-60 Flash array 스토리지는 최대 3.2GB/s에 달하는 읽기 성능을 나타내었다. 이는 인피니밴드가 고성능 스토리지를 가상화하기 위한 오픈스택 클라우드 스토리지 시스템에 적합함을 보여주었다.

5.2 해외 연구 사례

통신 프레임워크 UCX(Unified Communication X)를 인피니밴드 네트워크에서 성능을 측정하고 오버헤드를 분석하여 최적화하는 연구 사례가 있다[19]. UCX는 슈퍼컴퓨터, 데이터센터에서 Lower-level, Upper-level로 통신 API를 제공하는 프레임워크이다. Lower-level 타입의 UCT(Unified Communication Transport) API는 다양한 네트워크들을 위해 통신 기능들을 추상화하였고 Upper-level 타입의 UCP(Unified Communication Protocol)는 UCT API 기반으로 tag-matching, RMA(Remote Memory Access), atomic 오퍼레이션, 통신 함수들을 포함하는 API들을 제공한다. 이러한 두 레벨의 API를 통해 제안하는 프레임워크가 여러 패브릭 네트워크에서 이식성을 가질 수 있었다. 그 외에 UCS(Unified Communication Service) API는 메모리 관리, 데이터 구조 등 UCX, UCT에서 기본적으로 필요한 것들을 제공한다. 또한 [19] 연구에서는 UCP, UCT, OFED에서 제공하는 Verbs에 대하여 지연시간을 측정 및 비교하였다. 결과적으로 UCT 기반의 UCP 에서 메시지 길이가 짧을 때 오버헤드가 많이 발생하였지만 Verbs 기반의 UCT는 오버헤드가 많이 발생하지 않는 것을 확인하였다. 이러한 오버헤드의 원인을 찾기 위해 UCP의 실행시간을 분석하였다. 그 결과 오퍼레이션을 만들고 지시하는 put 함수와 오퍼레이션의 완료를 기다리는 flush함수에서 오버헤드가 발생하였고, put보다는 flush에서 오버헤드가 더 많이 발생함을 알 수 있었다. 이에 따라 오버헤드를 분석하여 RDMA functions, UCP 엔진에 대하여 최적화를 진행하였다. 최적화 결과는 RDMA Write, RDMA Read 오퍼레이션에 대한 대역폭과 초당 메시지 수가 짧은 메시지에서 UD(Unreliable Datagram)였을때 14% 향상되었고 긴 메시지에서 6~7% 향상되었음을 보여주었다.

SR-IOV(Single Root I/O Virtualization) 기술을 활용하여 인피니밴드 기반의 클러스터 환경에서 MPI(Message Passing Interface) 에플리케이션 기반의 VM(Virtual Machine) 마이그레이션 프레임워크를

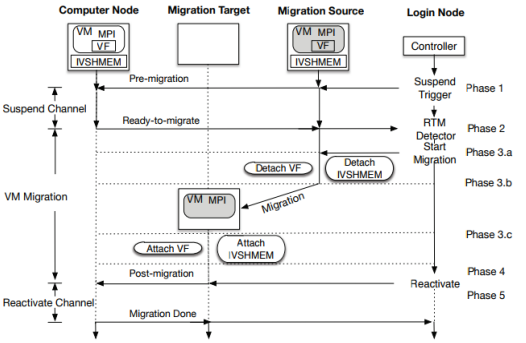

그림 12. 마이그레이션 시퀀스 다이어그램
Fig. 12. Sequence Diagram of Process Migration

제안하는 연구 사례가 있다[20]. VM 마이그레이션은 한 물리적 하드웨어 환경에서 다른 물리적 하드웨어 환경으로 VM을 이동하는 기술이다. SR-IOV 기술은 PCIe(PCI-Express) 어댑터를 가상화하고 VF(Virtual Function)는 가상화된 인피니밴드 장치를 의미한다. IVShmem(Inter-VM Shared Memory)는 VM들 사이에서 Zero-copy로 공유 메모리를 접근하는 기능을 제공한다. SR-IOV 기술은 마이그레이션 기능을 지원하지 않아 특별한 솔루션이 필요하며, [20] 연구에서 솔루션을 제안하고 있다. 그림 12는 마이그레이션 과정을 시퀀스 다이어그램으로 보여주고 있다[20]. 제안한 프레임워크에서 컨트롤러는 마이그레이션 과정 중에 명령을 내리고, 관리한다. 1 단계에서 마이그레이션 요청을 받고 각 노드에게 마이그레이션 전처리를 각 노드에게 명령한다. 2단계에서 마이그레이션 전처리 상황을 확인하고 완료시 다음 단계를 수행한다. 3단계 (a)에서 VF와 IVShmem은 마이그레이션이 불가능하므로 VM에서 제거한다. 3단계 (b)에서 하이퍼바이저에게 마이그레이션 요청을 한다. 3단계 (c)에서 VF와 IVShmem을 다시 연결한다. 4단계에서 각 노드에게 마이그레이션이 완료되었고 네트워크 서비스 시작을 명령한다. 5단계에서 MPI가 정상동작하고 마이그레이션이 완료되었음을 컨트롤러에게 전달한다. 마이그레이션 시간은 VM의 메모리 크기가 512MB일때, TCP, IPoIB, 인피니밴드에 대하여 마이그레이션 시간을 비교하였고, RDMA 기반의 인피니밴드에서 마이그레이션 시간을 20% 줄였음을 확인하였다.

5.3 인피니밴드를 적용한 실시간 데이터 처리 플랫폼 Apache Storm

본 절에서는 인피니밴드의 RDMA를 Apache Storm에 적용한 우리의 연구 사례에 대해 설명한다.

Storm[21]이란 대용량의 데이터 스트림을 실시간으로 처리하기 위한 분산 처리 시스템으로, 분산된 서버에 다수의 워커 프로세스가 동작하는 구조를 갖는다. 현재 Storm에서 다수의 워커 간 통신은 TCP/IP 기반의 Netty로 구현되어 있다. 따라서, 워커 간의 데이터 통신은 제어가 운영체제로 넘어가는 문맥 전환(context switch), 운영체제로의 버퍼 복사, 운영체제 내에서의 버퍼 복사 등의 이유로 CPU에 높은 부하를 야기시킬 수 있다. 이러한 문제점은 대역폭 대비 전송하는 메시지의 크기가 작을수록 더욱 심각해진다. 예를 들어, 1Gbit Ethernet 장비를 통해 데이터를 전송할 경우, 전송하는 메시지의 크기가 커질수록 데이터를 전송하는 시간이 길어지고 문맥 전환과 버퍼 복사의 수가 줄어들어, 결과적으로 CPU의 오버헤드가 감소할 수 있다. 하지만, 고성능 네트워크 장비인 InfiniBand에서 제공하는 IPoIB를 사용할 경우, Ethernet과 동일한 실험 환경에서도 여전히 많은 수의 문맥 전환과 버퍼 복사가 발생하여, CPU 과부하가 발생할 수 있다. 따라서, 이러한 문제점을 해결하기 위해 인피니밴드에서 제공하는 RDMA 기법을 Storm에 적용하는 방안을 설계 및 구현하였다.

그림 13은 제안하는 RDMA 기반 Storm의 구조로써, 기존의 Netty 기반 Storm 구조에서 일부 클래스를 추가 및 수정하였다. 그림 13과 같이 JXIO 라이브러리를 이용하여 Netty와 동일한 기능을 수행하도록 구현하고 새로운 메시지 전송 계층을 추가하였다. 또한, 기존 Storm의 TransportFactory 클래스를 기존 메시지 전송 계층인 Netty 뿐만 아니라, 새로 추가한 JXIO를 선택할 수 있도록 수정하였다.

표 1은, 기존 Storm의 이더넷 및 IPoIB 기반 통신과 제안하는 RDMA 기반 Storm을 비교한 결과이다. 먼저, 표 1(a)은 각 통신 프로토콜에 따른 메시지 전송 수를 비교하며, 제안하는 RDMA 기반 Storm이 이더넷 대비 최대 25배, IPoIB 대비 최대 2배 증가하였다. 다음으로, 표 1(b)는 스파우트가 생성한 튜플이 처리가 완료될 때까지의 처리 지연시간을 비교하며, 제안하는 RDMA 기반 Storm이 이더넷 대비 최대 52배, IPoIB 대비 최대 5배 감소하였다.

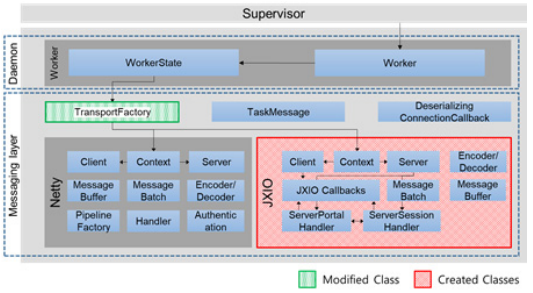

그림 13. RDMA 기반 Storm의 워커 간 통신을 위한 클래스 구조
Fig. 13. Class Diagram for Communication based RDMA among Storm Workers

추가적인 실험으로 CPU의 부하 정도를 나타내는 load average를 측정한 결과, Ethernet은 메시지 크기가 커질수록 CPU의 부하가 줄어들었지만, IPoIB는 20 이상의 수치를 유지하였다. 이는 테스트에 사용된 시스템 사양 상 load average 수치가 12를 넘으면 CPU가 과부하 상태라 할 수 있고, IPoIB 통신은 그 정도가 심각하다는 것을 알 수 있다. 하지만, Storm에 RDMA를 적용함으로써 IPoIB 기반 통신에서 20 이상을 유지하던 load average를 7 이하로 감소시킬 수 있었다. 결론적으로, InfiniBand의 RDMA를 통해 IPoIB에서 발생하던 CPU 부하 문제를 해결하였다. 또한, 이더넷과 IPoIB 대비 높은 성능 향상 효과를 얻을 수 있다.

Ⅵ. 결 론

빅데이터의 수요가 늘어나고부터, 빅데이터를 고속으로 처리하기 위해 하둡, 스파크, 스톰 등의 분산 환경 데이터 처리 플랫폼이 많이 사용되고 있다. 이렇듯 분산 환경에서 데이터 처리 연산 능력이 크게 향상되면서 분산 네트워크를 연결하는 기존 네트워크가 병목현상으로 작용할 우려가 생겼다. 이러한 문제점을 해결하기 위해 기존의 네트워크를 대체 가능한 고성능 네트워크 기술로 RDMA 기반의 InfiniBand,

표 1. 메시지 크기에 따른 통신 프로토콜 별 성능 비교
Table 1. Performance Comparison by Protocol

(a) 메시지 전송 수(개)

	Ethernet	IPoIB	RDMA
1 KB	434,301,130	1,603,289,790	2,432,686,750
10 KB	43,546,140	443,629,520	1,125,181,240
100 KB	4,279,660	40,686,220	105,224,360

(b) 처리 지연시간(ms)

	Ethernet	IPoIB	RDMA
1 KB	172.051	40.922	18.006
10 KB	1811.203	149.662	34.343
100 KB	16831.108	1667.145	344.768

RoCE, iWARP 등이 등장하였다. 이들은 모두 VIA에서 확대된 개념으로 높은 대역폭, 낮은 지연시간, 낮은 CPU 사용률을 특징으로 갖고 있다. RDMA 기반의 네트워크 에플리케이션을 개발하기 위해서는 OFED가 제공하는 라이브러리를 사용한다. 하지만 OFED 라이브러리는 에플리케이션 개발에 있어서 사용이 까다롭다. OFED 라이브러리를 개발자가 쉽게 사용할 수 있게 추상화하여 멜라녹스가 개발한 Accelio RPC API와 IBM이 개발한 자바 기반의 DiSNI 라이브러리를 살펴보았다. 또한 인피니밴드를 적용하여 고성능 데이터 처리를 수행한 국내외 최신 연구 사례에 대해서 살펴보고, 우리가 수행한 분산 환경에서의 스트리밍처리를 위한 인피니밴드 적용 사례에 대해서 소개하였다. 현재 고속 데이터 처리를 위해서 RDMA 기반의 인피니밴드 등의 기술이 가장 각광받고 있음을 알 수 있었다. 향후 우리는 SR-IOV, IVShmem와 같은 가상화 기술에 대하여 연구하고 고성능 서버들로 이루어진 클러스터 환경에서 가상화된 노드를 이용하여 데이터 처리 플랫폼의 성능을 향상시키는 연구를 진행하고자 한다.

References

[1] J. Gantz and D. Reinsel, *The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east* (2012), Retrieved Jul. 20, 2017, from http://www.emc.com/collateral/analyst-reports/idc-the-digital-universein-2020.pdf.

[2] I. A. T. Hashem, et al., “The rise of “big data” on cloud computing: Review and open research issues,” *Inf. Syst.*, vol. 47, pp. 98-115, Jan. 2015.

[3] S. Kaisler, F. Armour, J. A. Espinosa, and W. Money, “Big data: Issues and challenges moving forward,” in *IEEE HICSS*, pp. 995-1004, Hawaii, USA, Jan. 2013.

[4] R. Recio, et al., *An RDMA protocol specification*(2005), Retrieved Jul. 20, 2017. from https://tools.ietf.org/html/

[5] Top500, *List Statistics: November 2016* (2016), Retrieved Jun. 18, 2017, from http://www.top500.org/statistics/list/

[6] D. Dunning, G. Regnier, G. McAlpine, D. Cameron, B. Shubert, F. Berry, and C. Dodd, “The virtual interface architecture,” *IEEE Micro*, vol. 18, no. 2, pp. 66-76, Apr. 1998.

[7] K. Park and S. Moh, “InfiniBand : Next generation system area network,” *Commun. KIISE*, vol. 19, no. 3, pp. 43-51, Mar. 2001.

[8] InfiniBand Trade Association, *InfiniBand Architecture Specification*, vol. 1, Release 1.3, Mar. 2015.

[9] P. MacArthur, et al., “An integrated tutorial on InfiniBand, Verbs and MPI,” *IEEE Commun. Surv. & Tuts.*, vol. PP, no. 99, Aug. 2017.

[10] IBTA, *InfiniBand Roadmap*, Retrieved Sept. 15, 2017, from https://www.infinibandta.org/

[11] Timothy Prickett Morgan, *Oracle Engineers Its Own InfiniBand Interconnects*(2016), Retrieved Jul. 20, 2017, URL: https://www.nextplatform.com/2016/02/22/oracle-engineers-its-own-infiniband-interconnects/

[12] InfiniBand Trade Association, *InfiniBand™ Architecture Specification Release* 1.2.1 Annex A17: RoCEv2, Sept. 2, 2014.

[13] Intel, *Understanding iWARP: Delivering Low Latency to Ethernet* (2010), Retrieved Jun. 30, 2017, from http://www.intel.com

[14] Accelio, *Accelio – The OpenSource I/O, Message, and RPC Acceleration Library* (2013), Retrieved Jun. 29, 2017, from http://www.accelio.org.

[15] IBM, *Direct Storage and Networking Interface* (2017), Retrieved Jun. 30, 2017, from http://developer.ibm.com.

[16] T. Kim, B. Kim, and H. Jung, “Design and implementation of easily applicable and lightweight RDMA API for infiniband clusters,” in *Proc. Symp. KICS*, pp. 1483-1484, Jesu Island, Korea, Jun. 2015.

[17] B. Kim, J. Jung, H. Min, J. Heo, and H. Jung, “Efficient data pre-fetching scheme for infiniband based high performance clusters,” *KIISE Trans. Comput. Practices*, vol. 23, no. 5, pp. 293-298, May 2017.

[18] H.-S. Heo, K.-S. Lee, M. Pirahandeh, and D.-H. Kim, “Design of OpenStack cloud storage systems - Applying infiniband storage

network and storage virtualization performance evaluation," *KIISE Trans. Comput. Practices*, vol. 21, no. 7, pp. 470-475, Jul. 2015.

[19] N. Papadopoulou, L. Oden, and P. Balaji, "A performance study of UCX over InfiniBand," in *Proc. 17th IEEE/ACM Int. Symp. Cluster, Cloud and Grid Computing*, IEEE Press, pp. 345-354. Madrid, Spain, 2017.

[20] J. Zhang, X. Lu, and D. K. Panda, "High-performance virtual machine migration framework for MPI applications on SR-IOV enabled InfiniBand clusters," *IEEE IPDPS*, pp. 143-152, Orlando, Florida, USA, May 2017.

[21] Apache Storm, *Apache Storm*(2017), Retrieved Sept. 21, 2017, from http://storm.apache.org/.

최 성 윤 (Choi Seong-Yun)

2016년 2월 : 강원대학교 컴퓨터과학과 졸업

2016년 3월~현재 : 강원대학교 컴퓨터과학과 석사과정

<관심분야> 네트워크 관리, IoT, NFV, 클라우드

문 양 세 (Yang-Sae Moon)

1991년 2월 : 한국과학기술원 공학사

1993년 2월 : 한국과학기술원 공학석사

2001년 8월 : 한국과학기술원 공학박사

1993년~1997년 : 현대전자산업㈜ 주임연구원

2001년~2002년 : ㈜ 현대시스콤 선임연구원

2002년~2005년 : ㈜ 인프라밸리 기술위원(이사)

2005년~2008년 : 한국과학기술원 첨단정보기술연구센터 연구원

2008년~2009년 : 미국 퍼듀대학교 방문연구원

2012년~2013년 : 강원대학교 기획부처장

2014년~2016년 : 강원대학교 IT대학 부학장

2005년 3월~현재 : 강원대학교 컴퓨터학부 컴퓨터과학전공 교수

<관심분야> 데이터베이스, 데이터마이닝, 빅데이터 처리

최 미 정 (Choi Mi-Jung)

1998년 2월 : 이화여자대학교 공학사

2000년 2월 : 포항공과대학교 공학석사

2004년 2월 : 포항공과대학교 공학박사

2004년~2005년 : 프랑스 INRIA 연구소 박사후 연구원

2005년~2006년 : 캐나다 워터루대학 박사후 연구원

2006년~2008년 : 포항공대 컴퓨터공학과 연구 조교수

2008년~현재 : 강원대학교 컴퓨터학부 컴퓨터과학전공 부교수

<관심분야> 네트워크 관리, 정보보안, SDN/NFV 관리 및 통합