ISSN 1598-9798

DATABASE
데이타베이스연구
RESEARCH

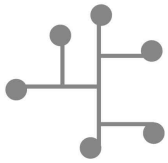
제38권 제3호 2022년 12월호

한국정보과학회
데이터베이스 소사이어티
KIISE Database Society of Korea

DATABASE RESEARCH

VOLUME 38 NUMBER 3, December 2022

Database Society
The Korean Institute of Information Scientists and Engineers



효율적인 빅데이터 처리를 위한 고속 분산 클러스터 기반 RDMA 라이브러리

High-speed Distributed Cluster-based RDMA Library for Efficient Big Data Processing

정래원(Laewon Jeong)[1] 길명선(Myeong-Seon Gil)[2]
문양세(Yang-Sae Moon)[3] 최형진(Hyung-Jin Choi)[4]

요 약

본 논문에서는 분산 클러스터에 최적화된 RDMA(Remote Direct Memory Access) 라이브러리를 제안한다. RDMA는 CPU를 거치지 않고 노드 간 메모리에 직접 데이터를 송수신하는 고성능 네트워크 프로토콜이다. 그러나, 기존 RDMA 개발 방식은 복잡도가 매우 높고, 일대일 통신 기반 모델로 인해 다수 노드 환경에 적용이 어려운 문제가 있다. 이를 해결하기 위해, 본 논문에서는 기존 API를 일반화하여 다수 노드에도 쉽게 적용할 수 있는 새로운 RDMA 라이브러리를 설계하고 구현한다. 제안 라이브러리는 1) 변수 설정, 2) 통신 정보 생성, 3) RDMA 수행, 4) RDMA 종료의 네 단계로 통신을 수행한다. 제안한 RDMA 라이브러리를 사용한 통신 모듈 구현 결과, 다수 노드 간의 데이터 송수신이 정상적으로 수행됨을 실제 실험을 통해 확인하였다. 결과적으로, 제안 라이브러리는 진입 장벽이 높았던 RDMA 응용 개발 난이도를 낮추고, 기존 분산 처리 환경에도 쉽게 활용할 수 있는 효율적인 연구 결과라 사료된다.

주제어: 인피니밴드, Remote Direct Memory Access, 고속 분산 처리, 분산 클러스터, 빅데이터 처리

1 강원대학교 컴퓨터공학과, 석사과정.
2 강원대학교 정보통신연구소, 선임연구원.
3 강원대학교 컴퓨터공학과, 교수.
4 강원대학교 컴퓨터공학과, 교수, 교신저자.
+ 이 논문은 2022년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원(No. 2021-0-00859, 초거대 그래프의 지능적 고속 처리를 위한 그래프 DBMS 기술 개발)과 한국연구재단의 지원을 받아 수행된 연구임(No. NRF-2022R1A2C1003067).
+ 논문접수: 2022년 11월 18일, 최종 심사완료: 2022년 12월 14일, 게재승인: 2022년 12월 21일.

Abstract

In this paper, we deal with an RDMA (Remote Direct Memory Access) development technique optimized for distributed clusters. RDMA is a high-performance network protocol that directly transmits and receives data between nodes in memory bypassing a CPU. However, the existing RDMA development method has a very high complexity, and it is very difficult to apply to multi-node environments due to its one-to-one communication-based model. To solve this problem, in this paper we design and implement a new RDMA library that can be easily applied to multiple nodes by generalizing the existing API. The proposed library performs RDMA communications in four steps: 1) setting variables, 2) generating communication information, 3) executing RDMA, and 4) ending RDMA. Through the real communication module implementation using the proposed RDMA library, we confirm that data transmission and reception between multiple nodes operate correctly. We believe that our library can be an efficient way to reduce the difficulty of RDMA application development, which has high entry barriers, and be easily utilized in the existing distributed processing environments.

Keywords: InfiniBand, Remote Direct Memory Access, High-speed distributed processing, Distributed cluster, Big data processing

1. 서 론

최근 다양한 분야에 활용되고 있는 빅데이터 분석이나 딥러닝 등의 복잡한 알고리즘 수행에는 분산 클러스터가 필수이다. 특히, HPC(High Performance Computing)급 사양에서는 각 노드를 연결하는 네트워크 통신이 전체 성능에 큰 영향을 끼친다. RDMA(Remote Direct Memory Access)는 이와 같은 고성능 분산 처리 환경에 사용되는 대표적인 데이터 통신 프로토콜로, 이더넷과 인피니밴드[1] 환경에서 사용 가능하다. RDMA는 CPU를 거치지 않고 노드 간 메모리에 직접 데이터를 송수신하며, 이러한 특성으로 인해 대용량 데이터 송수신 시 CPU 사용량과 데이터 전송 지연 시간을 크게 감소시킬 수 있다. 일반적으로 이러한 RDMA는 Verbs[2]라는 프로그래밍 API로 개발한다. 그러나, 기존 API는 개발 복잡도가 매우 높고, 실제 응용 개발 시 매우 많은 단계 정의가 필요하다. 또한, 일대일 통신 기반으로 매뉴얼이 구성되어 있어 분산 환경에 적용하기 위해서는 여러 시행 착오를 거칠 수밖에 없다. 따라서, 본 논문에서는 이러한 RDMA의 기존 프로그래밍 과정을 일반화하여 개발 복잡도를 낮추고, 다수 노드 환경에도 쉽게 적용할 수 있는 새로운 RDMA 라이브러리를 설계 및 구현한다[5].

제안 라이브러리는 1) 변수 설정, 2) 통신 정보 생성, 3) RDMA 수행, 4) RDMA 종료의 네 단계로 RDMA 통신을 수행한다. 변수 설정과 통신 정보 생성 단계에서는 일대일 통신, 다대일 통신, 일대다 통신에 필요한 네트워크 정보와 파라미터들을 설정한다. 이렇게 각 단계에서 설정된 정보들을 바탕으로 RDMA 기반 데이터 송수신을 실행하고, 송수신이 끝나면 RDMA 통신 종료를 위해 설정 정보와 자원을 초기화한다. 기존 Verbs API의 경우 다대다 환경에 적용하기 위해서는 단순히 일대일 모델을 복제하여 통신 연결 수를 늘리는 것이 아니라 통신에 필요한 연결 정보 설정, 통신에 사용되는 버퍼 할당 및 연결 등의 추가 작업이 필요하다. 분산 클러스터의 노드 수가 수 대인 경우에는 이 과정이 크게 복잡하지 않을 수 있다. 그러나, 클러스터의 물리 노드 수가 많아질수록, 응용 프로그램 내의 통신 연결 수가 많아질수록 설정 과정의 복잡도가 크게 증가한다. 제안하는 RDMA 라이브러리는 이러한 다수 노드의 분산 환경에 특화된 프로그래밍 모델이다. 분산 노드에서 다대다 통신을 구현해 실험한 결과, 제안 방법을 이용한 RDMA 통신이 정상적으로 처리됨을 확인하였으며, 자체 분산 클러스터 환경에서 최대 64MB 크기의 메시지도 안정적으로 전송됨을 보였다.

제안하는 RDMA 라이브러리는 분산 데이터 처리 환경에서 기존의 복잡한 RDMA 구현 과정을 보다 쉽게 재구성하고, 이를 일반화하여 다대다 통신 환경에 적용하였다. 이는 진입 장벽이 높았던 RDMA 응용 구현 난이도를 낮추고, 기존 분산 처리 환경에도 쉽게 적용·활용하여 성능 향상 효과를 얻을 수 있는 효율적인 연구 결과라 할 수 있다.

본 논문의 구성은 다음과 같다. 제2절에서는 제안 방법의 관련 연구를 설명한다. 제3절에서는 제안하는 RDMA 라이브러리의 전체 동작 구조와 세부 설계를 보인다. 제4절에서는 실험을 통해 제안 방법의 동작 과정을 확인한다. 마지막으로, 제5절에서 본 논문의 결론을 맺는다.

2. 관련 연구

RDMA는 CPU를 거치지 않고 원격 호스트 메모리에 직접 데이터를 전송하는 기술이다. 이러한 메모리 직접 접근 방식으로 지연시간, CPU 사용률,

[5] 제안하는 RDMA 라이브러리는 github(https://github.com/laewonJeong/rdma_library)에 공개되어 있다.

메모리 대역폭 병목 현상을 줄일 수 있으며, CPU 간섭없이 데이터를 송수신하기 때문에 고속 전송이 가능하다[3]. 이로 인해, 다양한 분야에서 RDMA를 사용하여 성능 향상을 시도하고 있다[4-7]. RDMA를 사용하기 위해서는 인피니밴드, RoCE(RDMA over Converged Ethernet), iWARP(Internet Wide Area RDMA Protocol) 프로토콜을 지원하는 하드웨어가 필요하다. 본 논문에서는 이 중 인피니밴드를 사용한 RDMA 통신을 활용한다.

인피니밴드[1]는 높은 대역폭과 낮은 지연시간을 특징으로 고성능 컴퓨팅 등에서 사용하는 통신 네트워크 표준으로, 최근에는 NDR(Next Data Rate) 장비 기준 최대 1.2Tbps 대역폭까지 지원한다. 이러한 인피니밴드 응용 구현을 위해서는 OFED (Open Fabric Enterprise Distribution)에서 제공하는 Verbs API를 사용해야 한다[2]. OFED는 RDMA 사용하기 위한 오픈소스 미들웨어로, Verbs 함수를 통해 인피니밴드 기반 RDMA 오퍼레이션을 수행하는 역할을 한다. 이때, RDMA 오퍼레이션은 MLNX_OFED[8] 드라이버를 통해 제공된다. 현재 RDMA 통신에는 SEND, SEND w/Imm, RECEIVE, WRITE, WRITE w/Imm (with immediate), Atomic 오퍼레이션을 사용할 수 있으며, 본 논문에서는 이 중 Atomic을 제외한 모든 오퍼레이션을 지원한다.

RDMA의 주요 오퍼레이션에 대한 특징을 간단히 설명하면 다음과 같다. 첫째, SEND/RECEIVE는 TCP/IP 통신과 유사하게 데이터를 보내고 이를 원격 노드에게 알린 뒤, 원격에서 받은 데이터를 확인하는 방식이다. 둘째, WRITE는 원격 노드 메모리에 데이터를 쓰고 이를 알리지 않는 방식이다. 셋째, WRITE w/Imm은 WRITE와 달리 데이터를 쓰고 원격 노드에 이를 알리는 방식이다. 오퍼레이션들 중 통신 이후 원격 노드에게 알리지 않는 방식은 one-sided 오퍼레이션이라 하고, 전송 결과를 알리는 방식은 two-sided 오퍼레이션이라 한다[9].

RDMA 통신을 위해서는 송수신 양쪽 모두 사전에 사용할 메모리 영역(memory region)을 지정해야 한다. 실제 통신은 인피니밴드의 HCA(Host Channel Adaptor)를 통해 이루어지며, 각 노드는 소켓과 유사하게 송수신을 위한 QP(Queue Pair)를 가진다. 송수신 작업은 WR(Work Request)를 통해 관리되며, 처리가 완료된 WR는 CQE(Completion Queue Entry)를 통해 확인할 수 있다. 이러한 속성들을 바탕으로 실제 RDMA를 수행하는 과정은 다음과 같다.

(1) 인피니밴드 컨텍스트 생성: 드라이버를 통해 HCA를 연결하고, 통신에 필요한 사용자 정의 공간을 생성한다.
(2) 프로텍션 도메인 생성: RDMA 통신 프로세스를 위한 메모리 영역을 할당한다. 이때, 할당 가능한 영역에는 MR, MW(memory window), QP, SRQ(shared receive queue), AH(address handles) 등이 있다.
(3) CQ(Completion Queue) 생성: 통신을 위한 큐의 나머지 쌍을 생성한다.
(4) QP 생성: 메시지 송수신을 실제로 동작시키는 QP를 생성한다. 이때 QP의 타입은 RC(Reliable Connection), UC(Unreliable Connection), UD(Unreliable Datagram)가 있다.
(5) QP 식별자 정보를 교환한 연결 설정: QP의 식별자를 통해 통신할 대상을 확인한다.
(6) QP 상태 변경: QP를 초기 상태인 RESET 상태에서, 통신이 가능한 상태(RTR(Ready to Receive), RTS(Ready to Send))로 변경한다.
(7) MR 등록: 단계 (6)까지의 초기화 과정이 끝나면, 메모리 영역 등록을 통해 실제 통신에 사용할 메모리를 할당한다.

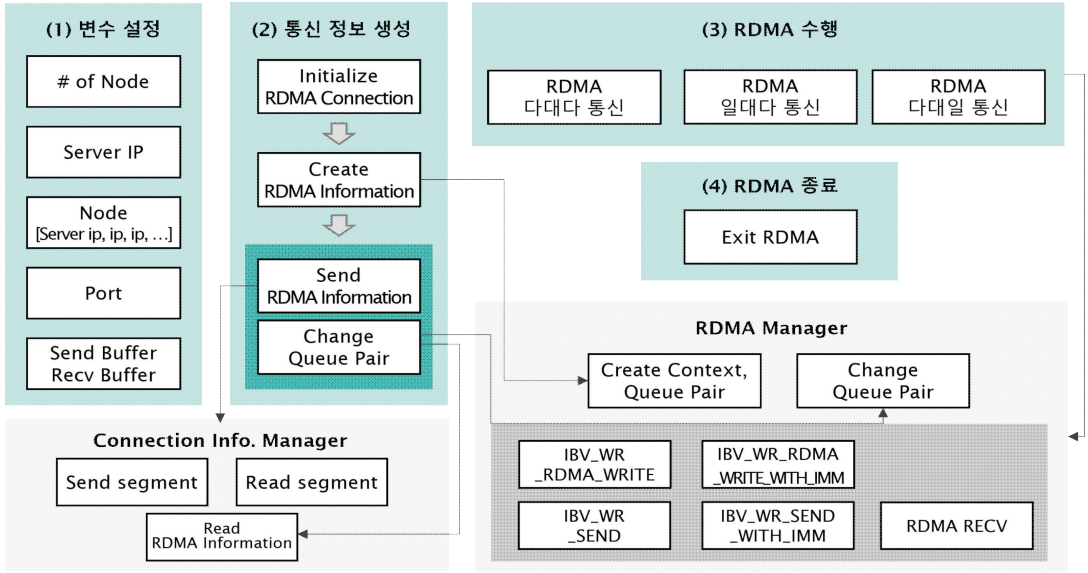
(8) MR 정보 교환 및 데이터 송수신: RDMA 통신 오퍼레이션을 통해 등록된 MR 및 QP를 기반으로 데이터를 송수신한다.

이와 같이, Verbs API를 이용한 구현은 통신을 위한 초기화 과정이 매우 복잡하기 때문에 해당 방식으로 RDMA 응용을 개발하는 것은 매우 비효율적이다. 이러한 기존 API의 단점을 해결하기 위해 RDMA 프로그래밍의 경량화 연구가 진행되었다. Infinity [10]는 인피니밴드를 위한 경량 C++ RDMA 라이브러리다. 기존 Verbs API 기반 RDMA 구현을 지원하며, 성능 저하 없이도 손쉬운 RDMA 응용 개발을 목적으로 한다. 국내에서 진행한 연구로는 김태일 외 2인이 개발한 경량 RDMA API가 있다[11]. 해당 연구에서는 세부적인 설정을 소켓 프로그램과 유사한 형태로 설계하여 RDMA 구현의 어려움을 해결하고자 하였다. 기존 연구들은 본 연구와 유사하게 RDMA 통신 구현의 난이도를 낮추는데 중점을 두었으나, 모두 일대일 통신만을 지원하여 다수 노드 환경에 적용하기에는 여전히 큰 불편을 초래한다. 따라서 본 논문에서는 일대일 통신뿐만 아니라 다대다 통신 환경에서도 쉽게 활용할 수 있는 새로운 RDMA 라이브러리를 제안한다.

3. 분산 노드 기반 RDMA 라이브러리

3.1 전체 아키텍처

본 절에서는 다대다 통신 지원 RDMA 라이브러리와 이를 기반으로 한 통신 모듈을 설명한다. 그림 1은 제안 라이브러리를 활용한 RDMA 통신 모듈의 동작 구조를 나타낸다. 제2절에서 설명한 바와 같이, 기존 RDMA 통신 메커니즘을 사용하면 다수의 노드로 구성되는 분산 클러스터에서 개발 복잡도가 매우 높아진다. 특히, 노드 수가 증가할수록, 노드 내부에 RDMA 통신을 요구하는 응용이 많아질수록 연결쌍을 코드 레벨에서 일일이 설정해야 하는 기존 개발 방식은 비효율적일 수 밖에 없다. 따라서, 본 연구에서는 이러한 초기화 과정을 간략화하여 1) 변수 설정, 2)



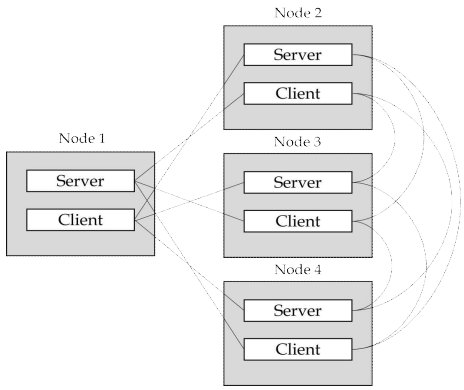
[그림 1] RDMA 다대다 통신 라이브러리 구조도.

통신 정보 생성, 3) RDMA 수행, 4) RDMA 종료의 네 단계로 구성한다.

각 단계의 동작을 간단히 설명하면 다음과 같다. 첫째, 변수 설정 단계에서는 통신에 필요한 노드 수, 서버 IP, 포트 번호, 송수신 버퍼 주소 등의 변수들을 설정한다. 둘째, 통신 정보 생성 단계에서 RDMA 통신에 필요한 정보와 Queue Pair, Completion Queue 등 데이터 송수신 관련 버퍼들의 생성, 등록 과정을 수행한다. 이를 위해, 변수 설정 단계에서 지정한 변수들을 등록하고 각 노드들을 소켓 통신으로 연결시킨다. 연결이 완료되면, RDMA 통신에 사용되는 큐들을 생성하고, 이들을 메모리 영역(memory region)에 등록한다. 각 노드들은 소켓 통신으로 데이터 송수신에 사용되는 큐들의 정보를 교환하며, 이를 통해 큐의 상태를 변경한다. 셋째, RDMA 수행 단계에서는 앞 단계에 설정한 정보와 Verbs API를 기반으로 일대다, 다대일, 다대다 통신을 수행한다. 이때, 제안하는 RDMA 라이브러리는 SEND, SEND w/Imm, RECEIVE, WRITE, WRITE w/Imm 오퍼레이션들을 지원한다. 모든 데이터 통신이 완료되면, 네 번째인 RDMA 종료 단계로 넘어간다. RDMA 종료 단계에서는 통신을 위해 생성했던 각종 큐들을 모두 삭제하고, 큐의 상태 및 등록을 관리하는 영역들 역시 초기화한다. 해당 과정이 끝나면 모든 통신 프로세스를 종료한다. 제안 라이브러리를 이용한 RDMA 통신 과정은 제3.3절에서 주요 함수와 함께 보다 자세히 설명한다.

3.2 라이브러리 주요 함수

본 절에서는 제안 라이브러리의 각 단계에서 사용되는 주요 함수에 대해 설명한다. 첫째, 변수 선언 단계의 `initialize_rdma_connection()`은 사용자가 선언한 변수를 등록하고 소켓 통신으로 노드들을 연결시키는 함수이다. 해당 함수는 현재 노드의 IP, 사용할 노드의 IP들, 사용할 노드의 개수, 포트 번호, send_buffer, recv_ buffer 파라미터를 입력받아 스레드를 이용해 그림 2와 같이 각 노드의 서버와 다른 노드의 클라이언트를 각각 연결시킨다.



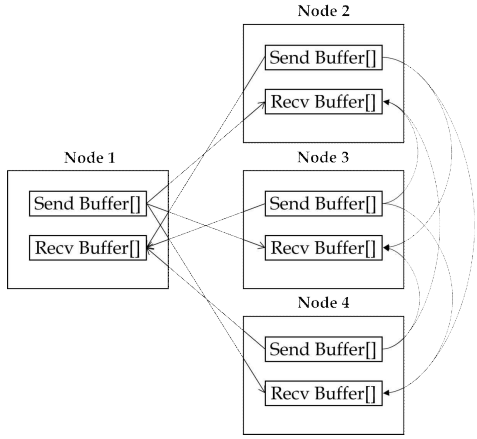
[그림 2] 노드 연결 예제.

둘째, 통신 정보 생성 단계의 `create_rdma_info()`는 RDMA 통신 정보를 생성하는 함수이다. 해당 함수에서는 Context, QP, CQ (Completion Queue) 등 RDMA 통신에 필요한 정보들을 생성하고, 사용할 버퍼들의 메모리 영역을 등록한다. 이 함수를 사용하면 기존 RDMA 구현 시 단계별로 정의했던 복잡한 정보들을 한 번에 생성·등록할 수 있다. 이때, 생성된 각 노드 정보를 이용해 큐의 상태를 변경하기 때문에 노드들끼리 서로 해당 정보를 공유할 수 있도록 각 노드에서 생성된 정보는 각자의 벡터에 저장하는 구조를 갖는다.

앞서의 `create_rdma_info()` 함수와 마찬가지로 통신 정보 생성 단계인 `send_info_change_qp()` 함수는 벡터에 저장된 RDMA 통신 정보를 각 노드들끼리 교환하고 QP의 상태를 변경하는 함수이다. 정보 교환은 소켓 통신을 이용하여 설계하였다. 여기에서, 정보 교환과 QP 상태 변경은 send_buffer, recv_buffer 정보들을 기준으로 각각 발생한다. 이는 send_buffer 정보와 recv _buffer 정보 교환 후

각 버퍼의 QP도 버퍼에 맞게 상태를 변경해야 하기 때문이다. QP의 상태는 초기화 상태인 Init, 수신받을 수 있는 상태인 RTR(Ready to Receive), 송신할 수 있는 상태인 RTS(Ready to Send)가 있다. 다음으로, `send_info_change_qp()` 함수에서는 각 노드들이 보낸 정보를 수신한 후, 그 정보들을 이용해 send_buffer는 데이터를 송신할 수 있는 상태인 RTS로, recv_buffer는 데이터를 수신할 수 있는 상태인 RTR로 변경하게 설계하였다.

셋째, `rdma_comm()` 함수는 RDMA 수행 단계에서 그림 3(a)와 같이 실제 RDMA를 이용한 다대다 데이터 송수신을 처리하며, 이를 위해 오퍼레이션별 동작 과정을 상세히 정의한다. 먼저 SEND, SEND w/Imm, WRITE w/Imm은 로컬이 원격에 데이터를 보내는 오퍼레이션인데, 이때 전송할 데이터와 수신 확인을 위한 32비트의 정수 값을 함께 보내도록 설계한다. 원격에서는 CQ에 쌓여 있는 정수 값을 폴링(polling)하여 수신이 완료되었음을 알 수 있다. 반면에, WRITE는 다른 오퍼레이션들과는 다르게 로컬이 원격에 데이터를 보낸 뒤 수신이 성공해도 판단할 수 있는 방법이 없다. 따라서, WRITE 오퍼레이션과 함께 소켓 통신을 이용해 "send" 시그널을 보내고, 해당 시그널을 받으면 수신 성공으로 판단하도록 하였다. 그리고 `rdma_comm()` 함수는 사용할 오퍼레이션과 서로에게 송신할 데이터를 인수로 넘겨주면 된다. 인수로 전달할 오퍼레이션은 string형으로 "send", "send_with_imm", "write", "write_with_imm"으로 지정할 수 있다. 송신할 데이터 역시 string형으로 지정해야 한다. 인수로 오퍼레이션이 결정되면, 그에 따라 데이터 송신과 수신을 스레드를 이용해 동시에 수행하고, 수신이 완료되면 recv_buffer에 수신된 데이터가 저장된다. 해당 함수는 그림 3(b)와 같이 사용할 수 있다.



(a) 다대다 통신 아키텍처.

```
string msg = "Hello World";
string opcode = "send";
myrdma.rdma_comm(opcode, msg);
```

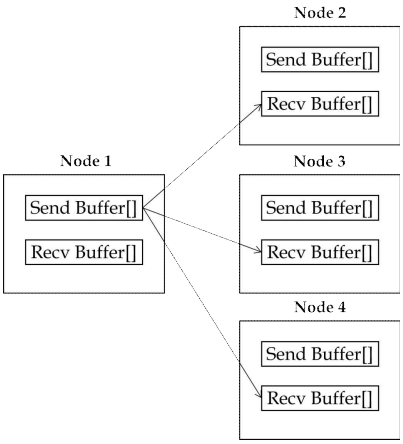
(b) 다대다 통신 구현 예제.

[그림 3] RDMA 다대다 통신.

넷째, `rdma_one_to_many_send_msg()`와 `rdma_one_to_many_recv_msg()`는 RDMA 수행 단계에서 일대다 통신을 위해 사용하는 함수이다. 일대다 통신은 그림 4(a)와 같이 변수 설정 시 정의한 서버 IP를 가진 노드가 메시지를 송신하고, 서버 IP가 아닌 노드들이 메시지를 수신하는 구조이다. 즉, 서버 IP에 해당하는 노드는 `rdma_one_to_many_send _msg()` 함수를 통해 사용할 오퍼레이션과 송신할 메시지를 인수로 넘겨주고, 서버가 아닌 노드들은 `rdma_one_to_many_recv_msg()` 함수를 통해 사용할 오퍼레이션만 전달한다. 이 함수들은 그림 4(b)와 같이 구현할 수 있다.

다섯째, `rdma_many_to_one_send_msg()`와 `rdma_many_to_one_recv_msg()`는 RDMA 수행 단계에서 다대일 통신을 위해 사용하는 함수이다. 다대일 통신은 그림 5(a)와 같이 서버가 아닌 노드들이 메시지를 송신하고, 서버 노드가 각 노드에서 오는 메시지들을 수신하는 구조이다. 즉, 서버에 해당하는 노드가 `rdma_many_to_one_recv_msg()` 함수로 사

용할 오퍼레이션을 전달하고, 서버가 아닌 노드들은 `rdma_many_to_one_send_msg()` 함수로 오퍼레이션과 송신할 메시지를 인수로 넘겨주면 송수신이 동작한다. 다대일 통신은 그림 5(b)의 코드와 같이 구현할 수 있다.



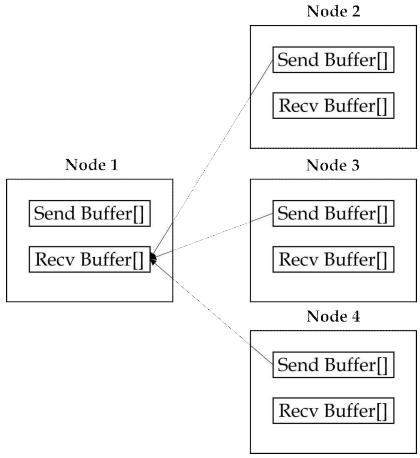
(a) 일대다 통신 아키텍처.

```
if(is_server(ip)){
    msg = "Hello World";
    myrdma.rdma_one_to_many_send_msg(opcode, msg);
}
else
    myrdma.rdma_one_to_many_recv_msg(opcode);
```

(b) 일대다 통신 구현 예제.

[그림 4] RDMA 일대다 통신.

마지막으로, `exit_rdma()` 함수는 RDMA 종료 단계에서 통신을 끝내기 위해 생성한 정보를 정리하는 기능을 제공한다. 즉, 통신에 사용한 큐를 모두 삭제하고, 큐 관리를 위해 생성한 메모리 영역들 역시 할당 해제하는 과정을 진행하게 된다. 따라서, 해당 함수는 RDMA 통신이 모두 끝난 뒤 프로그램 종료 전에 호출된다.



(a) 다대일 통신 아키텍처.

```
if(is_server(ip))
    myrdma.rdma_many_to_one_recv_msg(opcode);
else{
    msg = "Hello World";
    myrdma.rdma_many_to_one_send_msg(opcode, msg);
}
```

(b) 다대일 통신 구현 예제.

[그림 5] RDMA 다대일 통신.

3.3 RDMA 통신 과정

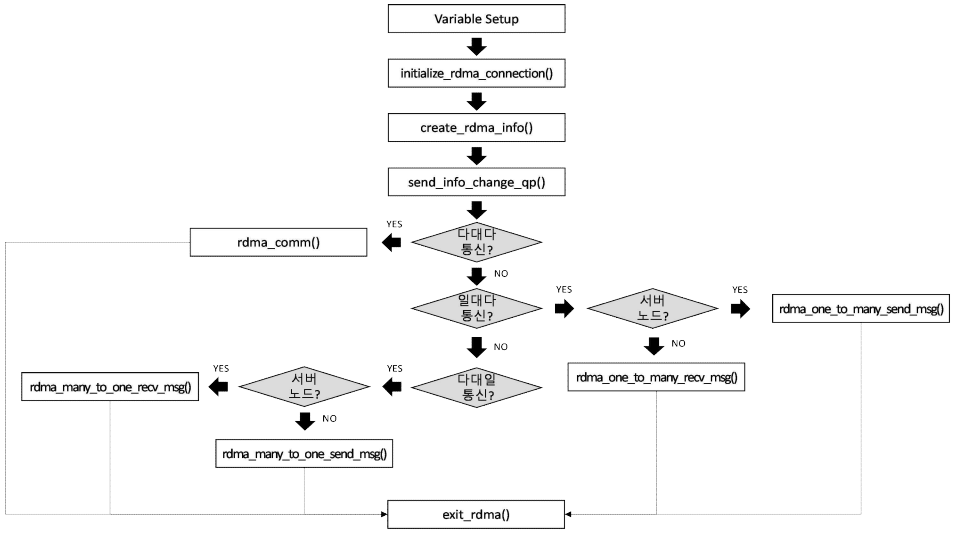
본 절에서는 제안 라이브러리 기반 RDMA 통신 과정을 설명한다. 그림 6은 RDMA 통신 순서를 나타내며, 전체 동작 과정은 다음과 같다.

① RDMA 통신에 필요한 변수들을 설정한다.

② `initialize_rdma_connection()` 함수로 설정한 변수들을 등록하고, 각 노드들을 소켓 통신으로 연결한다.

③ `create_rdma_info()` 함수로 RDMA 통신에 필요한 QP, CQ 등의 정보들을 생성하고, 버퍼들의 Memory Region을 등록한다.

④ `send_info_change_qp()` 함수에서 노드들끼리 생성한 정보를 소켓 통신으로 서로 교환하고, 각 버퍼의 QP 상태를 갱신한다.



[그림 6] 전체 RDMA 통신 과정.

⑤ 다대다 통신, 일대다 통신, 다대일 통신 중 사용할 통신 방식에 맞는 함수를 이용해 RDMA 기반 데이터 송수신을 진행한다.

⑥ 모든 작업이 완료되면 `exit_rdma()` 함수를 호출해 통신에 사용한 큐를 모두 삭제하고, 큐 관리를 위해 생성한 메모리 영역을 할당 해제한 뒤 전체 통신 프로세스를 종료한다.

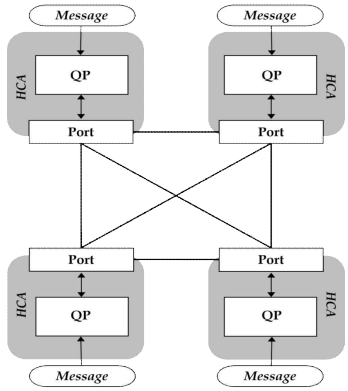
4. 구현 및 동작 평가

본 절에서는 제안 RDMA 라이브러리 기반으로 통신 모듈을 구현하고, 이의 정상적인 동작을 확인한다. 특히, 제안 라이브러리에서 지원하는 다대다, 일대다, 다대일 통신 프로세스를 모두 구현하고, 그 수행 결과를 보인다. 실험에는 표 1에서 볼 수 있듯이, Intel Xeon 2.4Ghz 6 Core, 64GB RAM을 장착한 서버 4대와 Mellanox 네트워크 스위치를 사용한다. 통신에는 1MB, 16MB, 64MB 크기의 메시지를 송수신하며, 다대다 통신, 일대다 통신, 다대일 통신 모두 SEND 오퍼레이션을 사용한다.

[표 1] 실험 환경.

장비	사양
노드	- Intel Xeon 2.4GHz 6 Core - 64GB RAM
네트워크	- InfiniBand: Mellanox SwitchX®-3 MSX6012F-1BFS Managed FDR 56Gbps

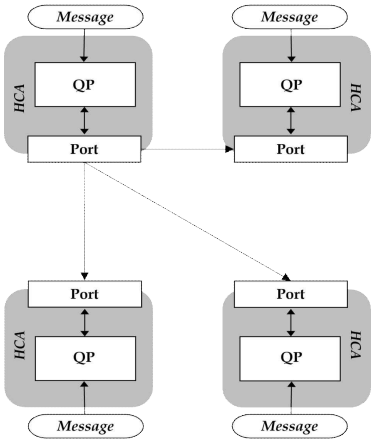
실험에서 다대다 통신의 경우 모든 노드는 송신과 수신을 함께 수행하기 때문에 각 노드는 서버와 클라이언트 역할을 동시에 담당한다. 또한, 모든 노드가 서로 완전히 연결된 매시 토폴로지 형태로 동작한다. 그림 7은 실험 중 다대다 통신 환경의 네트워크 토폴로지를 나타낸다.



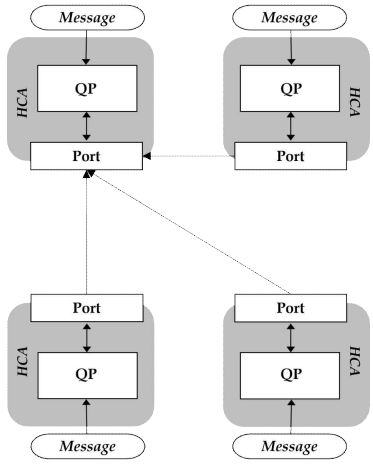
[그림 7] 다대다 통신 토폴로지.

일대다 통신의 경우, 그림 8과 같이 서버 IP를 가진 노드는 데이터를 송신하고 그 외 노드는 수신만 진행한다. 또한, 다대일 통신의 경우 서버 노드는 데이터를 수신하고 그 외 노드는 서버 노드에게 데이터를 송신하게 된다. 그림 9는 이러한 다대일 통신의 네트워크 토폴로지이다.

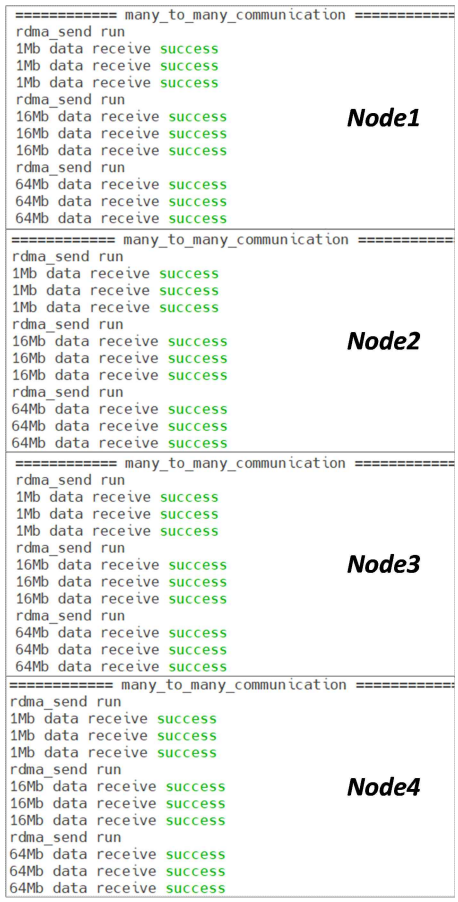
그림 10과 그림 11은 제안 라이브러리 기반으로 구성한 각 통신 토폴로지를 순서대로 수행한 결과이다. 그림을 보면, 총 네 개의 노드에서 다대다 통신, 일대다 통신, 다대일 통신 모두 정상적으로 메시지를 송수신함을 확인할 수 있다. 다대다 통신에서는 SEND 오퍼레이션을 통해 모든 노드에서 1MB, 16MB, 64MB의 다양한 데이터를 서로 송수신한다. 이때, 모든 노드에서 메시지 사이즈가 커져도 문제없이 데이터 전송이 가능하다. 일대다 통신의 경우, 노드 1에서 데이터를 송신하면 나머지 노드에서 데이터를 정상 수신함을 알 수 있다. 다대일 통신 역시 노드 2~4에서 송신한 데이터를 노드 1이 모두 정상적으로 수신함을 확인하였다.



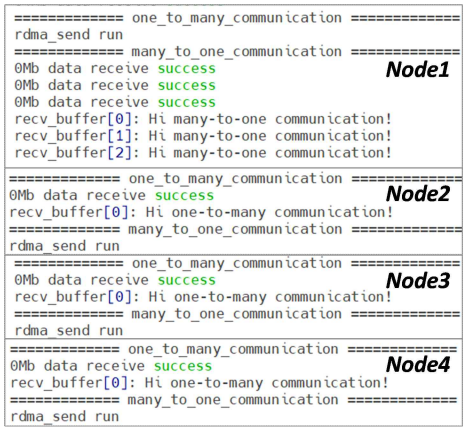
[그림 8] 일대다 통신 토폴로지.



[그림 9] 다대일 통신 토폴로지.



[그림 10] RDMA 다대다 통신 수행 결과.



[그림 11] RDMA 일대다, 다대일 통신 수행 결과.

5. 결 론

본 논문에서는 고성능 분산 클러스터 환경의 성능 향상에 활용할 수 있는 RDMA 라이브러리를 제안하였다. RDMA는 고속 통신에 최적화된 네트워크 통신 기술이지만, 개발 복잡도가 매우 높고 일대일 통신 기반 프로그래밍 모델로 인해 다수 노드로 구성된 환경에는 적용이 어려운 문제가 있다. 이를 해결하기 위해, 본 논문에서는 기존 프로그래밍 모델을 일반화하여 다수 노드 환경에 쉽게 적용할 수 있는 새로운 RDMA 라이브러리를 설계하고, 이를 인피니밴드 환경에서 구현 및 평가하였다. 다대다, 일대다, 다대일 환경으로 각각 구성한 데이터 송수신 실험을 통해 제안 라이브러리를 이용한 통신 모듈이 다수 노드 간의 다양한 데이터 송수신을 정상적으로 수행함을 확인하였다. 이러한 결과로 볼 때, 제안 라이브러리는 RDMA 응용 개발 복잡도를 낮추고, 다대다 통신 환경에도 쉽게 적용할 수 있는 효과적인 기술이라 사료된다.

참고 문헌

[1] Paul Grun “Introduction to InfiniBand for End Users,” *InfiniBand Trade Association*, 2010.

[2] Mellanox Technologies, RDMA Aware Networks Programming User Manual, Rev 1.7, Mellanox User Manual, 2015.

[3] S. Yang, S. Son, M.-J. Choi, and Y.-S. Moon, “Performance Improvement of Apache Storm using InfiniBand RDMA,” *The Journal of Supercomputing*, Vol. 75, No. 10, pp. 6804-6830, 2019.

[4] E. Zamanian, X. Yu, M. Stonebraker, and T. Kraska, “Rethinking Database High Availability with RDMA Networks,” In *Proc. of the VLDB Endowment*, Vol. 12, Issue 11, pp. 1637-1650, July 2019.

[5] C. Link, J. Sarran, G. Grigoryan, M. Kwon, M. M. Rafique, and W. R. Carithers, “Container Orchestration by Kubernetes for RDMA Networking,” In *Proc. of the IEEE 27th Int'l Conf. on Network Protocols*, Chicago, IL, pp. 1-2, Oct. 2019.

[6] D. Kim, T. Yu, H. H. Liu, Y. Zhu, J. Padhye, S. Raindel, C. Guo, V. Sekar, and S. Seshan, “FreeFlow: Software-based Virtual RDMA Networking for Containerized Clouds,” In *Proc of the 16th USENIX Symp. on Networked Systems Design and Implementation*, Boston, MA, pp. 113-126, 2019.

[7] N. S. Islam, M. Rahman, J. Jose, R. Rajachandrasekar, H. Wang, H.Subramoni, C. Murthy, and D. K. Panda, “High Performance RDMA-based Design of HDFS over Infiniband,” In *Proc. of the Int'l Conf. on High Performance Coumputing, Networking, Storage and Analysis*, Salt Lake City, UT, pp. 1-12, Nov. 2012.

[8] MLNX_OFED, https://developer.nvidia.com/networking/infiniband-software/.

[9] Z. Wang, H. Chen, X. Dong, W. Cai, and X. Zhang, "LogSC: Model-based One-sided Communication Performance Estimation," *Future Generation Computer Systems*, Vol. 132, pp. 25-39, July 2022.

[10] Infinity, https://github.com/claudebarthels/infinity.

[11] T. Kim, B. Kim, and H. Jung, "Design and Implementation of Easily Applicable and Lightweight RDMA API for InfiniBand Cluster," In *Proc. of the Symp. KICS*, pp. 1483-1484, Jeju, Korea, Jun. 2015.

정 래 원

2021년 강원대학교 컴퓨터과학과 학사
2021년~현재 강원대학교 컴퓨터과학과 석사과정

관심분야: 고성능 분산처리 시스템, 빅데이터 분석, 분산처리, 딥러닝, 데이터마이닝, 그래프 데이터

길 명 선

2007년 강원대학교 컴퓨터과학과 학사
2009년 강원대학교 컴퓨터과학과 석사
2021년 강원대학교 컴퓨터과학과 전산학전공 박사
2009년~2012년 강원대학교 중앙정보전산원
2021년~현재 강원대학교 정보통신연구소 선임연구원

관심분야: 데이터마이닝, 시계열 분석, 빅데이터 분석, 분산처리, 딥러닝, 그래프 데이터

문 양 세

1991년 한국과학기술원 전산학과 학사
1993년 한국과학기술원 전산학과 석사
2001년 한국과학기술원 전자전산학과 전산학전공 박사
1993년~1997년 현대전자산업(주) 주임연구원
2001년~2002년 (주)현대시스콤 선임연구원
2002년~2005년 (주)인프라벨리 기술위원(이사)
2005년~2008년 KAIST 첨단정보기술연구센터 연구원
2008년~2009년 미국 퍼듀대학교 방문연구원
2005년~현재 강원대학교 컴퓨터과학과 교수

관심분야: 데이터마이닝, 스트림데이터, 저장 시스템, 데이터베이스 응용, 분산처리, 빅데이터 분석, 딥러닝

최 형 진

1982년 영남대학교 물리학과 학사
1987년 일본동경공업대 정보공학 석사
1990년 일본동경공업대 정보공학 박사
1990년~1991년 ETRI 선임연구원
1991년~현재 강원대학교 컴퓨터과학과 교수

관심분야: 인공지능, 화상처리, 멀티미디어 처리