논문 17-42-10-21 The Journal of Korean Institute of Communications and Information Sciences '17-10 Vol.42 No.10
https://doi.org/10.7840/kics.2017.42.10.1999

종단간 전송효율성 극대화를 위한 오픈소스기반 병렬데이터 전송도구 성능측정 및 평가

박 종 선◆, 노 민 기*, 김 승 해°

Performance Measurement and Evaluation of Open Source Based Parallel Transfer Tools for End to End Transfer Efficiency Maximization

Jong-seon Park◆, Min-ki Noh*, Seung-hae Kim°

요 약

거대과학데이터의 경우 단일 파일의 크기가 매우 크고 이를 연구자간의 협업에 활용하기 위해서는 전송효율성이 매우 중요한 이슈이다. TCP를 기반으로 한 전송응용은 링크대역폭이 크고 데이터 전송거리가 긴 네트워크 환경에서 전송효율성이 크게 떨어진다. 이러한 경우 전송효율성 개선은 전송프로토콜과 전송도구 관점에서 해결 가능하다. 본 논문에서는 거대과학데이터 전송 효율성을 극대화하기 위한 오픈소스기반의 병렬 데이터 전송도구의 성능을 측정 후 평가한다. KREONET 및 GLORIAD 네트워크 환경에서 병렬전송을 통한 최적의 성능은 채널수가 대략 6개에서 10개 사이에서 측정됨을 실험을 통해 확인할 수 있었다.

Key Words : Science data, Transfer efficiency, UDP based transfer, Transfer tools, Parallel transfer

ABSTRACT

In case of science data, the size of single file is very large and transfer efficiency is a critical issue in order to utilize it for collaboration of researchers. The performance of applications based on TCP is significantly reduced in network environment where link bandwidth is large and data transfer distance is long. In this case, improvement of transfer performance can be solved in terms of transfer protocol and transfer tool. In this paper, we evaluate the performance of an open source based parallel data transfer tool to maximize the efficiency of large scientific data transfer. Experimental results between KREONET to GLORIAD network environment show that the optimal performance of parallel transfer is measured between 6 channels to 10 channels.

Ⅰ. 서 론

네트워크의 전송기술 및 전송시스템의 기술발전으로 인한 사용자의 전송속도에 대한 요구가 증대함에 따라 전송성능 극대화를 위한 전송효율성은 매우 중요하다. 특히 고에너지물리, 천체우주, 기상기후와 같은 거대과학분야 커뮤니티에서는 관측 장비나 시뮬레이션을 통해 생산된 데이터의 크기가 매우 방대하며

※ 본 연구는 2017년도 한국과학기술정보연구원(KISTI) 주요사업 과제로 수행되었음
◆ First Author : Korea Institute of Science and Technology Information, jspark@kisti.re.kr, 정회원
° Corresponding Author : Korea Institute of Science and Technology Information, shkim@kisti.re.kr, 정회원
* Korea Institute of Science and Technology Information, mknoh@kisti.re.kr
논문번호 : KICS2017-06-174, Received June 16, 2017; Revised September 11, 2017; Accepted October 23, 2017

연구자간의 협업을 위한 데이터 공유를 위해 신속한 데이터 전송이 매우 중요한 이슈이다.

최근 과학데이터 전송모델(Science DMZ)의 경우에도 고대역폭의 데이터 전송 전용 네트워크를 기반으로 전송 시스템을 활용한 전송환경을 제안한다[1]. 하지만 네트워크기술 발전에 따른 고대역폭의 네트워크 인프라 확충에도 불구하고 사용자가 얻는 현실적인 전송성능은 기대 수준에 미치지 못하고 있다. 이에 대한 주요 원인은 사용자 전송응용의 경우 대부분 TCP(Transmission Control Protocol)를 사용하는데서 찾을 수 있다[2,3]. TCP의 경우 패킷손실로 인한 성능이 급격하게 줄어들기 때문에 전송효율성이 매우 낮다. 따라서 TCP 성능보장을 위해서는 병목구간으로 인한 패킷손실을 방지를 최소화하는 것이 매우 중요하다. TCP의 패킷손실에 따른 성능감소는 링크 대역폭이 크고 데이터 전송거리가 증가함에 따라 즉, BDP(Bnadwidth Delay Product)가 클수록 더욱 크게 영향을 받는다. 따라서 TCP 전송성능개선을 위한 연구들은 혼잡제어 알고리즘의 접근을 통해 혼잡을 예측함으로써 패킷손실을 최소화하는 방안들이 주를 이룬다.

대용량 과학데이터 전송을 위한 네트워크에서는 데이터 전송효율성을 위해 보다 공격적인 전송기법이 요구된다. TCP의 경우 전송효율성을 위한 다양한 혼잡제어 알고리즘이 제안되었다. 하지만 개선된 혼잡제어 기반의 TCP일지라도 과학데이터 전송을 위한 전용네트워크의 경우처럼 BDP(Bandwidth Delay Product)가 매우 큰 네트워크 환경에서는 전송효율성이 떨어진다. 따라서 전송률 극대화를 위한 전송프로토콜이나 효율적인 전송기법이 필요하다.

최근 연구에서는 전송성능을 고려해 UDP 기반의 전송이 TCP 대안으로 연구되고 있다[4]. UDP 기반 전송의 경우 사용자 계층에서 TCP와 같은 별도의 혼잡제어기법을 추가적으로 구현함으로써 TCP와 같은 데이터 전송의 안정성은 물론 전송효율성까지 보장한다. TCP의 경우 매번 보낸 패킷에 대한 응답을 확인하는 구조로 인해 패킷을 보내는 과정에서 패킷손실로 인한 타임아웃이 발생하면 패킷의 양을 크게 줄이기 때문에 전송성능이 떨어지게 된다. 하지만 UDP를 기반으로 개선된 혼잡제어 알고리즘을 적용하게 되면 TCP가 갖는 성능 문제를 해결하면서 전송효율성을 크게 높일 수 있다. UDT(UDP based Data Transfer)의 경우 대표적인 UDP 기반 전송프로토콜이며 BDP가 큰 네트워크 환경에서 높은 전송성능을 보장한다[5].

BDP가 큰 네트워크 환경에서 데이터 전송효율성 극대화를 위한 전송기법은 다수의 전송채널을 활용하는 병렬전송이다. 병렬전송은 다수의 전송채널을 이용해 동시에 패킷의 전송이 가능하기 때문에 전송효율성을 극대화 할 수 있다[6,7]. 최근 데이터 전송극대화를 위해 오픈소스기반의 병렬전송도구의 활용이 빈번하다. 병렬전송기법은 사용자 전송응용에 따라 구현이 가능하다. 즉, TCP를 기반으로 하는 전송응용이나 UDP를 기반으로 하는 전송응용의 경우 모두 병렬전송기법을 적용해 전송성능을 극대화 할 수 있다. 최근 오픈소스기반의 병렬전송도구의 경우 Globus-online, MDTMFTP, FDT 같은 전송도구의 활용이 증가하고 있다.

본 논문에서는 대표적인 오픈소스기반 병렬전송도구의 성능을 측정하고 평가한다. UDT의 경우 대표적인 UDT 기반 전송프로토콜이며 이를 활용한 병렬전송도구 설계 및 구조에 대해 기술한다. 또한 Gbolus-online, FDT의 경우에는 대표적인 TCP 기반 병렬전송도구이며 이에 대한 구조를 기술한다. MDTMFTP의 경우 TCP 기반 병렬전송도구이면서 사용자 측의 전송시스템 자원을 활용하여 시스템의 병목구간을 제거함으로써 전송효율성 극대화가 가능하다. 본 논문에서는 이러한 오픈소스기반의 병렬전송도구의 구조에 대해 기술하고 성능을 평가한다.

본 논문의 구성은 다음과 같다. 2장에서는 전송프로토콜 관점에서 TCP 및 UDP 전송성능 개선을 위한 연구에 대해 기술한다. 3장에서는 전송도구 관점에서 TCP 및 UDP를 이용한 대표적인 병렬전송도구의 구조 및 특성에 대해 기술한다. 4장에서는 채널수에 따른 각각의 병렬전송도구의 전송성능을 측정 및 비교하고 5장에서 결론을 맺는다.

Ⅱ. 종단간 전송효율성 개선을 위한 전송프로토콜

대부분 사용자 전송응용의 경우 TCP에 기반하고 있다. 최근에는 BDP가 큰 네트워크 환경에서 전송성능 극대화를 위해 TCP를 사용하는 대신 UDP 활용이 증가하고 있다. TCP의 경우 패킷손실에 따른 전송성능의 영향이 크기 때문에 네트워크 혼잡상태를 고려해 혼잡제어를 얼마나 적응적으로 수행하느냐에 따라 전송효율성이 결정된다. 이번 장에서는 TCP, UDP 전송프로토콜 기반의 종단간 전송효율성 극대화 방안에 대해 기술하기로 한다.

2.1 TCP 기반의 전송효율성 개선 연구

TCP는 1980년대 중반부터 사용자 대부분의 전송

응용을 위한 프로토콜로써 사용되어져 오고 있다. TCP는 데이터 전송 신뢰성과 안정성을 보장하는 것이 특징이다. TCP가 제안된 당시의 네트워크 환경이 대역폭이 작고 사용자간의 전송거리가 작은 LAN (Local Area Network) 환경임을 고려하면 TCP의 전송효율성이 매우 중요한 이슈는 아니었다. 하지만 네트워크 환경이 변화함에 따라 TCP의 전송효율성을 고려한 연구들이 활발하게 진행되었다[8]. 즉 TCP의 성능 개선을 위한 연구는 혼잡으로 인한 패킷손실을 고려한 LAN 구간에서의 성능개선을 목표로 한 것과 BDP가 큰 네트워크 환경에서의 성능개선을 목표로 한 연구로 구분할 수 있다.

전송프로토콜의 경우 블랙박스와도 같은 네트워크 상태를 예측하기 위한 요소로는 패킷손실과 RTT (Round Trip Time)이다. TCP가 제안된 초기에는 패킷손실을 네트워크 혼잡으로 고려했기 때문에 정확하게 네트워크 상태를 예측하는 것이 어려웠다. 또한 패킷손실이 이미 발생한 후에 혼잡회피를 통한 패킷의 양을 조절하기 때문에 전송효율성이 크게 떨어진다. 이후 네트워크 혼잡을 보다 정확하게 예측함으로써 전송효율성을 높이기 위해 RTT나 단방향 지연 (OWD: One Way Delay)를 활용한다. RTT는 패킷을 보내고 이에 대한 응답이기 때문에 네트워크 상태에 대한 예측이 가능하다. OWD는 단방향 지연이기 때문에 더욱 정교한 예측이 가능하다. 이후에는 패킷손실과 네트워크 지연을 혼합한 기법을 사용함으로써 보다 정확하고 신속한 패킷의 조절이 가능한 혼잡제어 알고리즘이 제안되었다.

그림 1은 혼잡제어 수행 방식에 따른 TCP 성능개선에 대한 연구를 구분하여 나타내며 손실기반, 지연기반 그리고 이를 혼용한 연구로 구분된다. 손실기반은 패킷손실을 곧 네트워크 혼잡으로 간주하고 패킷의 양을 줄이는 데 비해 지연기반은 RTT나 OWD 지연으로 네트워크 상태를 판단하기 때문에 손실기반에 비해 정교하다. 이를 혼용한 기법의 경우 네트워크 상태와 패킷의 양을 결정하기 때문에 전송효율성을 크게 높인다.

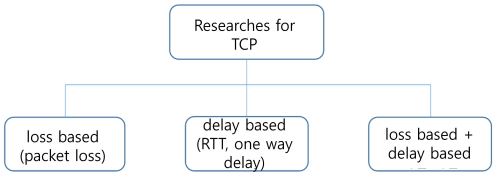

그림 1. TCP 혼잡제어 수행 방식에 따른 성능개선 연구
Fig. 1. Classification of TCP performance improvement techniques according to congestion control method

BDP가 큰 네트워크 환경에서 TCP 성능이 제한되는 이유는 TCP의 혼잡제어 알고리즘인 혼잡회피에서 기인한다. TCP는 패킷손실 이후에 혼잡회피 구간에서 패킷의 선형증가를 하게 되는데 BDP가 큰 네트워크 환경에서는 신속한 가용대역폭 확보가 힘들다. 따라서 TCP 전송성능 개선을 위한 주요 기법은 패킷손실의 최소화를 통한 혼잡회피 구간 진입의 수를 얼마나 잘 줄이는 데 있다. RTT나 OWD를 이용해 네트워크 상태를 예측하면 이러한 패킷손실률을 크게 줄일 수 있다. RTT의 급작스런 변화는 병목구간에서의 혼잡으로 판단해 패킷의 양을 줄임으로써 패킷손실을 방지할 수 있다. TCP 연구의 가장 개선된 방법으로는 네트워크 상태를 예측함과 동시에 가용대역폭을 예측하는 방법이다. 즉 RTT나 OWD를 이용하여 현재 네트워크의 가용대역폭을 계산하고 패킷손실이 발생하더라도 계산된 가용대역폭의 값으로 전송 패킷의 양을 정확하게 설정할 수 있다. 이는 패킷 손실이후의 혼잡회피 단계에서 인위적으로 패킷의 양을 줄이는 것보다 획기적으로 전송효율성을 높일 수 있다. HSTCP(High Speed TCP), FTCP(Fast TCP), TCP-Illinois와 같은 전송프로토콜들이 가용대역폭 예측을 통해 성능을 개선한 대표적인 전송프로토콜이다.

2.2 UDP 기반의 전송효율성 개선 연구

TCP의 경우 근본적인 전송메커니즘으로 인해 BDP가 큰 네트워크 환경에서 전송효율성이 제한되는 문제를 갖는다. UDP 기반 전송프로토콜은 TCP 대안으로 BDP가 네트워크 환경을 목표로 제안된 방안이다. UDP 기반 전송프로토콜은 UDP와 TCP의 특성을 혼합한 기법을 사용한다. UDP의 경우 보낸 패킷에 대한 응답확인이 없는 대신 연속적으로 데이터의 전송이 가능하다. 하지만 TCP와 같은 데이터 전송의 안정성과 신뢰성을 제공하지 못한다. 이러한 UDP의 데이터 전송 신뢰성 확보를 위해 사용자 계층에서 TCP와 같은 기능을 추가적으로 구현한다. UDP 기반 전송프로토콜은 TCP의 혼잡회피 방법을 궁극적으로 제거하고 해결함으로써 전송효율성을 크게 높인다. TCP가 보낸 패킷에 대한 응답을 확인하는 구조를 제거한다. 보낸 패킷에 대한 응답대기지연은 BDP가 큰 네트워크 환경에서는 전송성능의 제한을 갖게 하는 주요한 원인이다. UDP 기반 전송프로토콜은 일정 시간동안 연속적으로 데이터를 전송하고 일정 주기마다 수신측으로부터 응답을 확인함으로써 매우 공격적인 구조를 갖는다.

표 1. TCP 기반 전송프로토콜 및 UDP 기반 전송프로토콜 특성 비교
Table 1. Comparison of high speed transfer protocol based on UDP and TCP

transfer protocol	data transfer channel	packet control	congestion control
TCP based transfer protocol	- data and control message transfer with sing channel	- calculation of packet according to available bandwidth	- waiting for a response from the receiver after sending the packet - congestion avoidance when retransmission due to packet loss and timeout occur
UDP based transfer protocol	- separation of data and control message transfer channels · data transfer : UDP · control message : UDP or TCP	- control of packet sending period according to available bandwidth	- continuous packet transfer for a certain period - receives a response from receiver and retransmits lost packets in case of packet loss

그림 2는 UDP 기반 전송프로토콜의 혼잡제어를 도식하여 나타낸다. 그림 2의 경우와 같이 UDP 기반 전송프로토콜은 일정시간동안 연속적으로 데이터를 전송하고 이에 대한 응답을 받는 구조임을 알 수 있다. 이는 TCP가 보낸 패킷에 대한 응답을 수신하고 다음 패킷을 보내는 방법에 비해 매우 공격적인 기법으로 효율성을 높일 수 있는 방안이다. 또한 데이터와 제어정보의 전송을 위한 채널을 구분한다. 이는 데이터 전송효율성은 높이고 데이터 전송 신뢰성과 안정성을 제공하기 위한 방법이다. 가용대역폭에 따른 패킷의 양을 결정할 때도 차이를 보인다. TCP의 경우 가용대역폭에 따른 패킷의 양을 매번 계산하는 반면 UDP 기반 전송프로토콜의 경우 증가시킬 패킷의 양을 계산하고 그에 따른 패킷의 간격을 조절한다. 이 또한 가용대역폭을 신속하게 점유하기 위한 방안이다.

TCP는 패킷손실이 발생하면 수신측으로부터 송신측으로 중복 응답을 보냄으로써 패킷의 전송량을 줄인다. 계속적인 중복 응답을 받게 되면 타임아웃이 발생하고 패킷의 양을 매우 작은 양으로 설정하여 혼잡

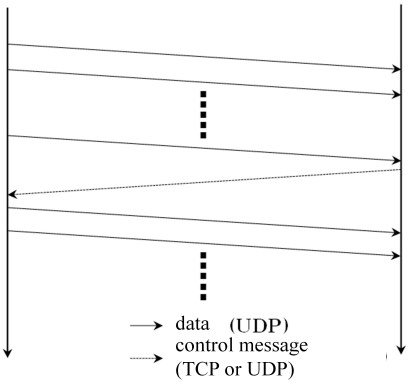

그림 2. UDP 기반 전송프로토콜의 데이터 전송 구조
Fig. 2. Architecture of UDP based transfer protocol

회피 구간을 수행하게 된다. 이러한 방법이 TCP의 전송성능을 크게 제한하게 한다. 반면 UDP 기반 전송프로토콜에서는 일정 주기마다 응답을 보내며 응답에는 손실된 패킷에 대한 기록을 포함하여 전송하게 된다. 이러한 방법을 통해 UDP 기반 전송프로토콜은 BDP가 큰 네트워크 환경에서 가용대역폭을 신속하게 확보하면서 전송효성을 크게 높일 수 있다. 표 1은 TCP 기반 고속전송프로토콜과 UDP 기반 전송프로토콜 주요 특성을 비교하여 나타낸다. TCP 대안으로 제시된 UDP 기반 주요 특징은 데이터 전송 채널의 분리, 전송 패킷 조절 방식 그리고 혼잡제어를 수행하는 방법으로 요약할 수 있다.

Ⅲ. 종단간 전송효율성 극대화를 위한 병렬전송도구

데이터 전송을 위한 전용의 네트워크가 보편화됨에 따라 데이터 전송효율성은 매우 중요한 이슈가 되고 있다. 거대과학데이터의 경우처럼 데이터 규모가 방대하고 연구자간 글로벌 협업이 요구되는 네트워크 환경에서는 전송률 극대화를 위한 방안이 필수적이다. 데이터 전송을 위한 전용 네트워크 환경이 연동되는 경우에는 혼잡으로 인한 패킷 손실의 확률이 줄어들기 때문에 공격적인 방법을 통해 전송효율성을 높일 수 있다. 여러 개의 병렬 데이터 전송채널을 사용해 데이터를 전송할 경우 전송효율성을 크게 개선할 수 있다[9].

그림 3은 단일채널 및 병렬전송채널을 통한 데이터 전송의 비교를 나타낸다. TCP의 경우 보낸 패킷의 대한 응답대기로 인해 다음 패킷전송에 대한 지연이 발생하는데 비해 병렬전송의 경우 다수의 채널을 통해 연속적으로 패킷전송이 가능하기 때문에 데이터 전송거리가 길더라도 대역폭을 충분히 활용하여 파이프라인을 신속히 채울 수 있다. 반면 네트워크 대역폭이

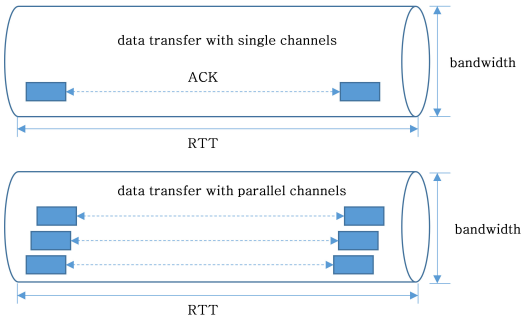

그림 3. 단일채널을 통한 데이터 전송과 병렬채널을 통한 데이터 전송 비교
Fig. 3. Comparison of data transfer through single channel and parallel channels

큰 경우에는 종단간 시스템의 성능으로 인한 병목이 발생될 가능성이 크며 이로 인해 성능이 감소될 수 있다. 이러한 경우에는 네트워크 대역폭을 충분히 활용 가능한 종단간 시스템의 성능도 고려해야 한다. MDTMFTP 병렬전송도구는 시스템 자원을 이용해 시스템의 병목을 해결하여 성능을 개선한 경우이다. 이번 장에서는 오픈소스기반의 병렬전송도구에 대해 기술한다.

3.1 UDT 병렬전송

UDT는 BDP가 큰 네트워크 환경에서 대용량 데이터 전송의 효율성을 목표로 TCP 대안으로써 제안된 대표적인 UDP 기반 전송프로토콜이다. TCP 혼잡제어와는 차별화된 전략을 추가적으로 구현함으로써 신속한 가용대역폭 확보를 고려하여 설계되었다. 특히 BDP가 큰 환경에서 패킷손실을 고려해 전송효율성이 급격하게 감소하는 원인을 해결하기 위한 새로운 혼잡제어를 적용함으로써 매우 우수한 전송효율성을 보인다.

그림 4는 TCP와 UDT 전송프로토콜 스택 구조를 간략히 도식하여 나타낸 그림이다. TCP 스택의 경우 TCP socket API를 통한 사용자계층 구조인 반면 UDT의 경우 별도의 UDT socket API(Application Programing Interface)를 제공한다. 이는 데이터 전송 효율성을 위한 UDP 사용과 사용자 계층에서 별도의 TCP와 같은 데이터 전송 신뢰성 기능을 추가적으로 제공하기 위한 방법이다.

[10]에 의해 제안된 UDT는 데이터 전송과 제어정보의 전송을 별도의 UDP 채널을 통해 전송한다. 또한 UDT는 사용자계층에서 UDT socket API를 제공함으로써 사용자가 필요한 기능을 추가적으로 구현 가능한 기능을 제공한다. [11]에서는 이러한 특성을 이용

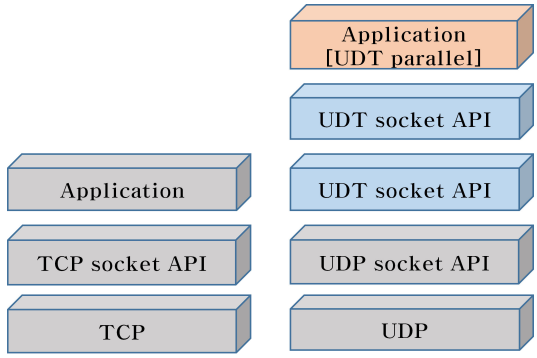

그림 4. TCP 및 UDT 전송 아키텍처 비교
Fig. 4. Architectures of TCP and UDT

하여 UDT 기반의 병렬전송을 구현하였다.

그림 5는 UDT 병렬전송구조를 간략히 도식하여 나타낸다. [11]에서 제시한 병렬전송은 사용자 설정을 통해 채널 수 결정이 가능하게 설계되어 있다. 그리고 사용자가 설정한 채널을 통해 데이터 전송 시 파일에 대한 유효성 검사를 위한 별도의 기능을 제공한다. 그림 5에서 보는 것과 같이 데이터 전송은 UDT 즉 UDP를 통해 수행하는 반면 데이터 전송 관련한 모든 정보들은 TCP를 통해 전송한다. 데이터 전송을 위한 초기의 연결 설정도 TCP를 통해 이루어진다. 이와 같이 전송채널과 정보채널을 분리함으로써 데이터 전송 신뢰성 및 안정성을 확보할 수 있다.

그림 5의 주요 기능을 간략히 요약하여 정리하면 다음과 같다.

- parallel transmitter: 사용자가 설정한 다수 병렬전송 채널 생성 기능을 제공한다. 송신측의 경우 send buffer를 통해 수신측으로 데이터를 전송하고

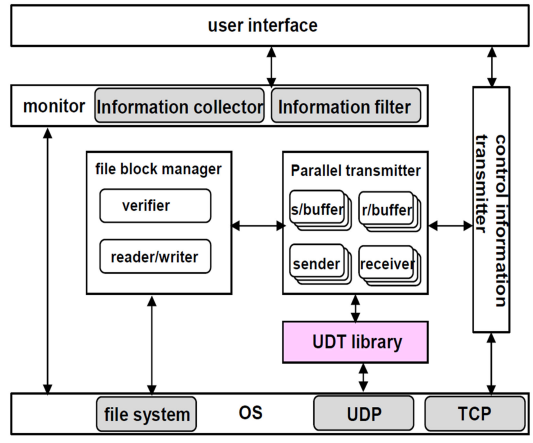

그림 5. UDT 기반 병렬전송 소프트웨어 아키텍처
Fig. 5. UDT based parallel software architecture

수신측 경우에는 반대로 receive buffer를 통해 데이터를 수신한다.

- fileblock manager: 데이터 전송블록을 관리한다. 이 경우에도 사용자가 임의의 전송블록 크기를 결정 가능하도록 되어 있다. 전송블록의 크기를 크게 설정할 경우 동시에 더 많은 데이터를 보낼 수 있다. 또한 송수신 측에서 주고받는 데이터에 대한 유효성을 검사하여 송신측에서는 reader를 통해 파일시스템으로부터 데이터를 읽어 send buffer로 전달한다. 수신측의 경우에는 receiver buffer를 통해 데이터를 수신하고 유효성을 검사하여 file writter를 통해 파일시스템에 순차적으로 기록한다.
- control information transmitter: 데이터 전송 시 연결 수립 설정 및 사용자가 설정 가능한 모든 정보들이 TCP를 통해 전달된다.

그림 5의 아키텍처를 기반으로 한 UDT 병렬전송의 데이터 전송은 최초 사용자 인터페이스를 통해 병렬전송채널 수를 결정하는 것으로 시작한다. 예를 들어 사용자는 데이터 전송채널의 수를 4개에서 8개 사이의 적절한 값으로 설정 가능하다. 채널수를 결정하고 전송을 시작하면 UDP를 통해 데이터를 연속적으로 전송하게 된다. 데이터는 사용자가 설정한 수만큼의 send buffer를 통해 수신측으로 전송되어 진다. 수신측에서도 사용자가 설정한 수만큼의 read buffer를 통해 데이터를 수신하게 된다. 그리고 데이터 시퀀스 비교 및 데이터 유효성 검사를 위해 file block manager를 통해 데이터를 하드디스크에 기록하게 된다.

3.2 Globus-online 병렬전송

Globus-online은 과학데이터 커뮤니티에서 발생한 대용량 데이터 전송 및 연구자들 간의 글로벌 협업을 목표로 설계된 소프트웨어 기반 서비스(software-as-a-service)를 위한 병렬전송도구이다[12]. Globus-online은 GridFTP 전송플로토콜을 지원하는 서버를 이용해 데이터를 전송하며 방화벽으로 인한 성능제한 및 최소화 인증을 통해 데이터 전송이 가능한 구조로 되어 있다.

그림 6은 Globus-online의 데이터 전송구조를 도식하여 나타낸다. 현재, Globus-online의 큰 특징 중의 하나는 직관적인 사용자 인터페스를 제공하는 것이다. Globus-online 클라이언트가 설치되어 있는 시스템을 이용하여 GCS(globus Connect Server) 간의 데이터 전송이 가능하다. 일반적으로 GCS는 데이터 전송을 위한 전용 시스템으로 구성되며 대용량 데이터 저장 및 공유를 위해 고사양의 cpu, 메모리, 네트워크 인터

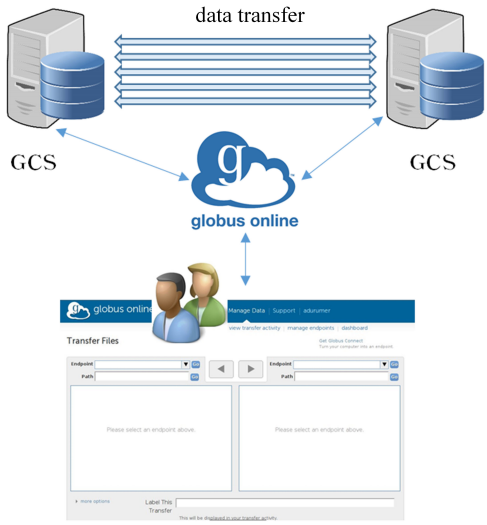

그림 6. Globus-online 데이터 전송 구조
Fig. 6. Globus-online data transfer architecture

페이스 카드를 사용하는 것이 보편적이다.

기능상으로는 데이터 전송 신뢰성, 신속성, 안정성을 제공하며 사용자 편의성을 제공한다. 윈도우, 맥, 리눅스 등 다양한 운영체제를 지원함으로써 운영체제 간의 호환성을 제공한다. Globus-online의 경우 별도의 CLI(Command Line Interface)를 제공한다. CLI의 경우 클라이언트 쉘스크립트를 통한 명령어 기반의 Globus-online 제어가 가능하다.

표 2는 Globus-online 데이터 전송을 위한 CLI 주요기능을 정리하여 나타낸다. CLI 모드를 사용할 경우 데이터 전송에 대한 옵션 선택이 가능하고 전송상

표 2. Globus-online 데이터 전송을 위한 CLI 명령어
Table 2. CLI commands to globus-online data transfer

function	command	description
data transfer	ls	- information of file and directory on GCS
	transfer	- data transfer between GCS
	scp	- scp data transfer between GCS
monitoring	status	- status of data transfer time and complete time and etc.
	details	- error information of data transfer
	events	- events of transfer, stop, errors
control	cancel	- cancel of data transfer
	wait	- waiting of data transfer
	modify	- modify of data transfer

태에 대한 모니터링 및 전송에 관한 다양한 기능을 제어할 수 있다.

3.3 MDTMFTP 병렬전송

MDTMFTP(Multicore-aware Data Transfer Middleware)는 멀티코어를 기반으로 한 고속병렬전송도구이다. DOE ASCR(Advanced Scientific Computing Research) 지원을 받아 Fermilab과 Brookhaven 국립연구소에서 개발해 무료 공개서비스를 제공하고 있다.

그림 7은 MDTMFTP 아키텍처를 간략한 그림으로 도식하여 나타낸다[13]. MDTM 미들웨어는 시스템이 제공하는 멀티코어를 최대한 활용하기 위한 기능을 제공한다. 사용자 계층에서는 parallelism, concurrency, pipelining의 병렬기능을 통해 데이터를 전송한다. MDTM 미들웨어를 기반으로 사용자 계층에서 병렬전송기법의 FTP를 구현한 MDFMFTP 주요 특징은 다음과 같이 요약할 수 있다.

- 효율적인 데이터 전송을 위한 파이프라인 I/O(Input/Output) 기능을 제공한다. 전용의 I/O 쓰레드를 통해 네트워크와 디스크 I/O 오퍼레이션 기능을 확장한다.
- 멀티코어 시스템의 최대한 효율적인 사용을 위해 MDTMFTP 미들웨어 서비스를 제공한다.
- 최적의 전송효율성을 고려해 제로카피(zero copy), 비동기 I/O(asynchronous I/O), 파이프라이닝(pipelining), 버퍼 풀(buffer pool) 메커니즘 기능을 제공한다.

연구커뮤니티의 경우 거대 관측 및 시뮬레이션 데이터 전송을 위해 네트워크 대역폭이 고속화됨에 따라 대역폭의 크기가 10Gb/s에서 100Gb/s 수준으로 고속화되고 있다. 이러한 경우 병목구간은 종단간 시스템에서 발생하는 경우가 일반적이다. 이러한 경우에는 시스템 I/O를 최대한 활용함으로써 디스크의 기록속도를 높이면 전송성능을 크게 개선할 수 있다. 또한 데이터 전송을 위한 전용 서버의 경우 멀티코어 수준의 CPU 규격이 보편화 되고 있다. MDTMFTP는 이러한 멀티코어 활용을 기반으로 병렬전송 FTP를 구현한 병렬전송도구이다.

그림 8은 파이프라인 기반 MDTMFTP 데이터 전송 구조를 나타낸다. 데이터 전송은 멀티코어를 활용하여 파이프라인 방식을 통해 전송된다. 각각의 I/O 쓰레드를 통해 네트워크 인터페이스 카드와 디스크 I/O를 제어함으로써 병렬 데이터 전송이 가능하다.

그림과 같이 MDTMFTP는 데이터 송신측의 read buffer, send buffer와 수신측의 read buffer, write buffer를 통해 디스크로부터 데이터를 읽고 기록한다. 데이터 송수신을 위한 쓰레드의 기능은 다음과 같이 요약할 수 있다.

- 디스크 read 쓰레드를 통해 디스크나 스토리지로부터 데이터를 읽는다.
- 디스크나 스토리지 write 쓰레드를 통해 데이터를 기록한다.
- 네트워크 sender 쓰레드를 통해 네트워크 인터페이스 카드를 통해 데이터를 전송한다.
- 네트워크 receiver 쓰레드를 통해 네트워크 인터페이스 카드를 통해 데이터를 읽는다.

이러한 각각의 쓰레드를 통해 다수의 송수신 버퍼에 데이터가 분산되어 제어 및 분산 처리된다. 동시에 네트워크 sender 쓰레드를 통해 네트워크 인터페이스를 통해 병렬 데이터 전송을 하게 된다. MDTMFTP의 경우 다수의 TCP 스트림을 통해 데이터를 전송한

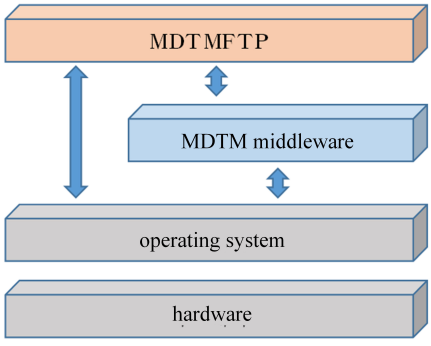

그림 7. MDTMFTP 전송 아키텍처
Fig. 7. MDTMFTP transfer architecture

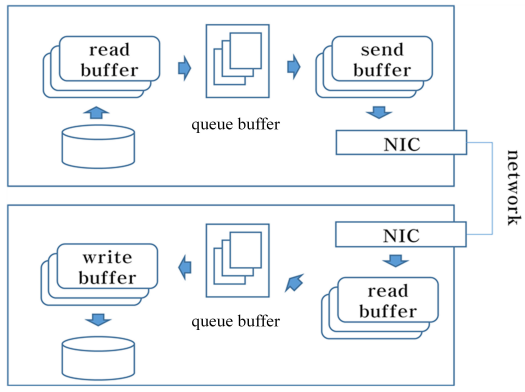

그림 8. 파이프라인 I/O 기반의 MDTMFTP 데이터 전송 구조
Fig. 8. Pipeline I/O based MDTMFTP data transfer architecture

다. 수신측에서는 이와 반대의 과정을 통해 데이터를 수신하고 디스크에 기록하게 된다. 각각의 쓰레드 제어를 위해 사용자 계층에서 구현된 MDTMFTP는 MDTM 미들웨어를 호출한다.

일반적으로 고속데이터 전송프로토콜의 경우 시스템에서 메모리 사용이 확연하게 크며 큰 버퍼 공간을 사용한다. 이러한 메모리 자원의 할당 문제를 피하기 위해 MDTMFTP는 미리 할당된 다수의 버퍼를 활용하며 이를 위해 데이터 버퍼 풀을 통해 메모리 버퍼를 제어한다. 이는 메모리 페이징이나 스왑 공간 문제를 해결할 수 있다. 이처럼 MDTMFTP는 병렬전송채널을 통해 공격적인 전송과 동시에 데이터 전송 시스템의 자원을 활용함으로써 시스템의 병목을 제거함으로써 전송성능을 극대화한 도구이다.

지금껏 TCP, UDP 전송프로토콜의 혼잡제어를 통한 전송 효율성 개선은 물론 다양한 병렬전송기법을 통한 성능개선이 진행되어 왔음에도 불구하고 MDTMFTP 는 다음 두 가지 원인에 의한 성능 이슈를 해결하고자 한다.

- 기존 데이터 전송응용도구의 경우 운영체제 지원에 의한 멀티코어를 충분히 활용할 수 없다. 최근에는 데이터 전송시스템의 규격이 높아짐에 따라 MDTMFTP는 불균일 기억장치인 NUMA(Non-Uniform Memory Access) 사용을 지원한다.
- 데이터 전송시스템에서 parallelism, concurrency, pipelining 병렬전송기법을 모두 지원하지 않는다. 대부분 대용량 전송도구 및 병렬전송도구가 parallelism이나 concurrency만을 지원하는 반면 MDTMFTP의 경우 세 가지 병렬전송기법을 지원한다.

3.4 FDT 병렬전송

FDT(Fast Data Transfer)는 TCP 기반 병렬전송도구이며 BDP가 큰 네트워크에서의 전송효율성을 고려해 설계되었다. 자바 기반으로 대부분 운영체제에서의 호환성을 보장하며 사용자 편의성을 고려해 설계되었다[14]. 비동기 방식의 멀티 쓰레드 기법을 사용한 FDT의 특징은 다음과 같이 요약할 수 있다.

- 다수의 TCP 소켓을 통해 버퍼 풀을 관리하며 데이터 셋의 연속적인 전송이 가능하다.
- 시스템 내에서 read, write 각각의 독립적인 버퍼를 사용한다.
- 사용자에 의한 다수의 병렬전송채널 설정이 가능하다.
- 디스크 I/O 및 네트워크 대역폭을 고려해 적당한 수준으로 버퍼 크기의 조절이 가능하다.
- 비동기 방식으로 전송된 파일을 조립한다.

그림 9는 FDT 데이터 전송 구조를 그림으로 도식하여 나타낸다. 다수의 TCP 스트림을 통한 대용량 데이터 셋을 전송 가능하도록 sernder 측에서는 분산형태의 파일 읽기가 가능하고 receiver 측에서는 전송데이터에 대한 조립이 가능한 구조이다. FDT의 경우에도 다수 전송스트림을 통한 데이터 전송을 위해 큐 버퍼를 통해 데이터를 순차적으로 보내고 수신측에서는 데이터를 받는 구조를 지닌다. 또한 서로 다른 디스크로부터 서로 다른 파일들을 동시에 읽어 들여 동시에 파일전송이 가능한 것이 특징이라 할 수 있다.

FDT 병렬전송도구의 데이터 전송과정은 서버, 클라이언트 구조로 동작한다. 원격의 데이터 전송서버에서 서버모드를 실행한 후 클라이언트에서 데이터 전송 옵션을 설정하고 데이터를 전송한다. 클라이언트 모드 실행 시 parallelism 옵션을 통해 데이터 전송 채널의 수를 결정할 수 있다. 그림 9와 같이 클라이언트 모드에서 사용자가 설정한 수만큼 큐 버퍼가 생성되며 로컬디스크에 데이터를 기록하게 된다. 또한 클라이언트 모드의 -r 옵션을 통해 서버의 디렉토리를 한꺼번에 전송할 수 있다. FDT의 concurrency는 그림 9에서 보이는 것처럼 서버의 각각 디스크로부터 독립적인 쓰레드를 통해 데이터를 전송하는 것이다. 이처럼 concurrency 기능을 사용하는 경우 데이터 전송효율성을 크게 개선할 수 있다. FDT의 경우 기본 적으로 4개의 parallelism과 2개의 concurrency로 설정되어 있으며 클라이언트 옵션을 통해 사용자에 의해 조절 가능하다.

지금까지 기술한 병렬전송도구의 경우 모두 다수의 채널을 이용한 매우 공격적인 전송기법을 사용함으로써 네트워크 가용대역폭의 신속한 확보가 가능함을 알 수 있다. 이처럼 공격적인 전송 전략을 위해서는

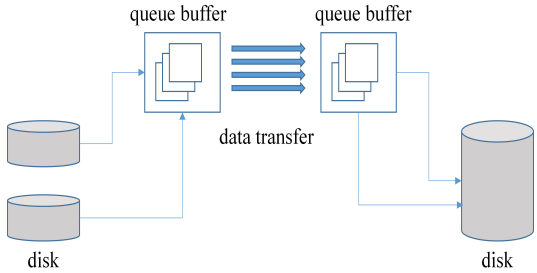

그림 9. FDT 데이터 전송 구조
Fig. 9. FDT data transfer architecture

표 3. 각각의 병렬전송기법 주요 특성 비교
Table 3. Comparison of main characteristics of parallel transfer tools

parallel transfer tool	주요 병렬전송기법	주요 기능 및 특징
UDT	parallelism	- use efficient congestion control algorithm optimized for high-speed network differentiated from TCP - data transfer through UDP and control information transfer through TCP - various transfer and monitoring options through CLI function
Globus-online	parallelism, concurrency, pipelining	- TCP based data transfer and control information transfer - peer to peer transfer as well as data transfer via grid distributed network environment - user-friendly interface - various transfer and monitoring options through CLI function
MDTMFTP	parallelism, concurrency, pipelining	- TCP, UDP based data transfer - MDTM middle-ware - parallel transfer utilizing multicore-based system resources - docker based installation
FDT	parallelism, concurrency	- TCP based data transfer - Compatibility between operating systems through Java-based implementation

프로세스, 쓰레드, I/O에 대한 관리 기법이 요구된다. 즉, parallelism, concurrency, pipelining 기법의 선택적인 구현을 통해 병렬전송기법의 구현이 가능하다. parallelism, concurrency, pipelining 기법은 다음과 같이 요약할 수 있다.

- parallelism : 동일 전송파일이 다수의 병렬전송채널을 통해 동시에 연속적으로 전송된다.
- concurrency : 서로 다른 여러 개의 전송파일들이 각각 채널을 통해 동시에 전송된다.
- pipelining : 동일 전송채널(데이터전송 및 제어정보전송)을 통해 서로 다른 전송파일들이 연속적으로 전송된다.

UDT 병렬전송, Globus-online, MDTMFTP, FDT 모두 이러한 병렬전송기법을 통해 각각의 특성에 맞는 전송효율성을 높이고 있다. UDT 병렬전송의 경우 parallelism을 활용한 병렬전송기법을 사용하고 FDT의 경우에는 parallelism과 concurrency를 사용한다. 그리고 Globus-online과 MDTMFTP의 경우 parallelism, concurrency, pipelining 기법 모두를 사용한 병렬전송도구이다. 특히 MDTMFTP의 경우에는 고대역의 네트워크를 고려해 시스템에서의 병목 해결을 위해 멀티 코어를 기반으로 한 병렬전송기법을 활용하는 것이 차별화된 전략이다. 통상 과학데이터 전송을 위한 네트워크 환경에서는 전송을 위한 전용의 네트워크를 구축해 사용하며 이러한 경우에는 사용자 측에 있는 전송시스템에서 병목구간이 될 수 있다. 병목구간에 따른 혼잡은 패킷손실로 인한 전송효율성 감소의 원인이 되기 때문에 MDTMFTP의 병렬전송기법은 매우 의미가 있다고 할 수 있다. 표 3은 각각의 병렬전송도구에 대해 주요 병렬전송기법과 기능을 요약하여 나타낸다.

Ⅳ. 성능 평가

이번 장에서는 병렬전송도구에 대한 성능을 측정하고 평가한다. 대역폭이 큰 WAN(Wide Area Network) 구간에서의 성능평가를 위해 KREONET (Korea Research Environment Open NETwork)과 GLORIAD(GLObal RIng network for Advanced Application Development)에 연결된 각각의 성능측정서버를 이용하여 측정한다. 성능측정서버는 대전과 미국에 각각 위치한다. 성능측정을 위한 시스템은 1Gb/s 링크를 통해 각각 연결하고 RTT는 대략 150ms의 값을 갖는 네트워크 환경이다. 각각의 병렬전송도구 파일전송테스트는 5GByte 파일을 전송 후 throughput

표 4. 성능테스트를 위한 시스템 규격
Table 4. System specification for performance testing

specification	KREONET server	GLORIAD server
model	DELL R710	DELL R720
cpu	Intel Xeon X5680 3.33GHz/24	Intel(R) Xeon(R) E5-2680 2.70GHz/24
memory	32GB	32GB
NIC	10Gb/s	1Gb/s

을 측정한다. 병렬전송도구 파일전송을 수행하기에 앞서 iperf 성능측정도구를 사용해 튜닝 여부에 따른 네트워크 속도를 측정하고 이후 각각의 병렬전송도구에 따른 파일전송 테스트를 측정한다. 표 4는 성능측정을 위한 각각의 서버 규격을 간략히 요약하여 나타낸다.

4.1 iperf를 사용한 네트워크 성능측정

파일전송 성능측정에 앞서 iperf 전송도구를 활용한 성능측정을 수행하였다. iperf는 디스크 기록 속도가 아닌 메모리의 성능테스트이며 통상 네트워크 전송속도를 의미한다. RTT가 150ms 이면서 1Gbps 네트워크 환경에서 파이프라인을 충분히 활용하기 위해서는 종단간의 전송 시스템의 별도의 튜닝이 요구된다. 리눅스의 경우 컨피그파일 수정을 통한 커널 접근이 가능하다.

표 5는 시스템 튜닝을 통한 커널 파라미터와 설정값을 나타낸다. 시스템 튜닝을 통한 성능개선은 메모리 버퍼 크기, 오토튜닝버퍼 크기, 백로그 크기, 혼잡제어 알고리즘, 점보프레임 설정을 통해 가능하다. 버퍼의 경우 최대 송수신 관련한 버퍼의 크기를 의미하여 일반적으로 BDP 설정에 따른다. 통상적으로 BDP는 링크대역폭과 RTT의 곱으로 구할 수 있다.

본 논문에서는 미국 에너지연구망 ESNET에서 제시한 [15]의 설정 값에 따라 충분히 큰 값으로 설정하였다. 네트워크 대역폭이 10Gb/s 이하이면서 RTT가 200ms 이하인 구간에서는 [15]의 값에 따라 최적의 전송 성능을 얻을 수 있다. mtu_probing의 값이 1이면 점보프레임을 설정한 값이다.

그림 10은 iperf를 이용해 100초 동안 측정한 네트워크 성능을 나타낸다. 시스템 튜닝 전 성능은 200Mb/s 이하의 throughput이 측정된다. 하지만 표 4

표 5. 커널튜닝을 위한 파라메터 및 설정 값
Table 5. Parameters for kernel tuning and value

kernel parameters	value
net.core.rmem_max net.core.rmem_max	134,217,728 Byte 134,217,728 Byte
net.ipv4.tcp_rmem net.ipv4.tcp_wmem	4,096 Byte, 87,380 Byte, 67,108,864 Byte 4,096 Byte, 65,536 Byte, 67,108,864 Byte,
net.core.netdev_max_backlog	30,000 Byte
net.ipv4.tcp_congestion_control	HTCP
net.ipv4.tcp_mtu_probing	1(enable)

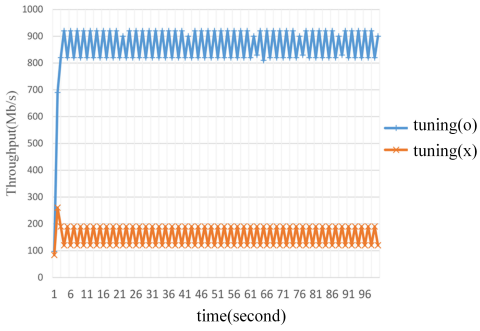

그림 10. 시스템 튜닝에 따른 성능측정 결과
Fig. 10. Throughput test result according to system tuning

의 값으로 튜닝 후 성능은 대략 800Mb/s에서 900Mb/s의 성능을 보인다. 성능측정을 통해 확인가능하듯이 시스템 튜닝 여부에 따라 매우 큰 성능 차이를 확인할 수 있다. 일반적으로 WAN 환경의 링크대역폭이 큰 환경에서 대역폭을 충분히 활용하기 위해서는 시스템 튜닝이 병행되어야 한다. 이처럼 [15]에서 제시한 튜닝 값에 따라 성능이 크게 높아지는 것은 TCP 혼잡제어 알고리즘 중 시스템 버퍼의 크기를 높인 효과로 판단할 수 있다. 즉 기본적으로 버퍼의 크기가 작게 설정되어있던 것을 크게 함으로써 보다 공격적인 패킷전송을 가능하게 하는 방법이다.

4.2 병렬전송도구 파일전송 성능측정

이번 장에서는 대표적인 병렬전송도구인 UDT, Globus-online, MDTFTP, FDT 대한 파일전송 테스트 결과에 대해 평가한다. 파일전송테스트는 서버, 클라이언트 구조로 5GByte 파일을 전송한 throughput을 측정한다. 각각의 병렬전송도구를 이용해 전송 채널의 수를 최대 10개까지 증가시키면서 throughput을 측정하고 성능의 변화를 측정한다.

그림 11은 UDT 병렬전송에 대한 throughput 결과를 채널 수 증가에 따라 측정한 결과이다. UDT 병렬전송도구의 경우 채널수를 10개까지 증가시키면서 성능을 측정하였다. 채널수에 따른 비례적인 증가를 보이면서 채널 수 9에서 가장 높은 성능 784.54Mb/s가 측정되었다. 하지만 채널수를 10개로 설정하였을 경우에는 더 이상 성능이 증가하지 않고 throughput이 감소하는 결과를 보인다.

그림 12는 Globus-online 성능측정 결과를 나타낸다. Globus-online의 경우에는 단일 채널을 통한 데이터 전송에서 대략 321Mb/s의 성능을 보인 후 이후 채널수를 증가시켜도 throughput의 큰 증가를 보이지 않

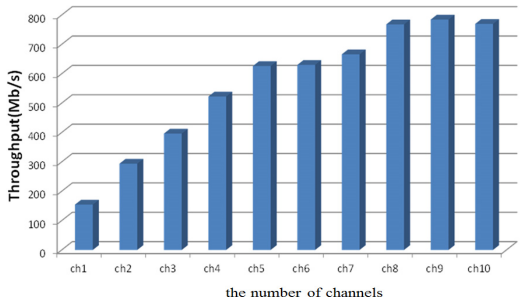

그림 11. 채널수에 따른 UDT 병렬전송
Fig. 11. UDT parallel transfer according to the number of channels

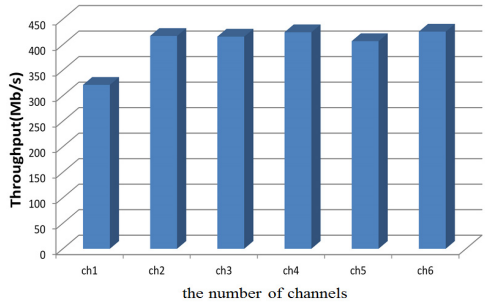

그림 12. 채널수에 따른 Globus-online 병렬전송
Fig. 12. Globus-online parallel transfer according to the number of channels

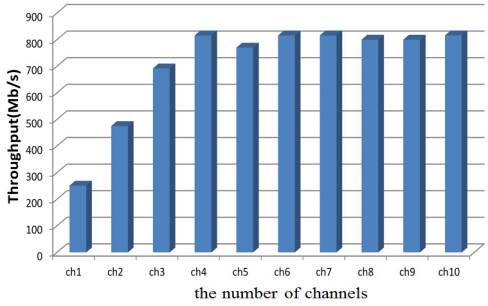

그림 13. 채널수에 따른 MDTMFTP 병렬전송
Fig. 13. MDTMFTP parallel transfer according to the number of channels

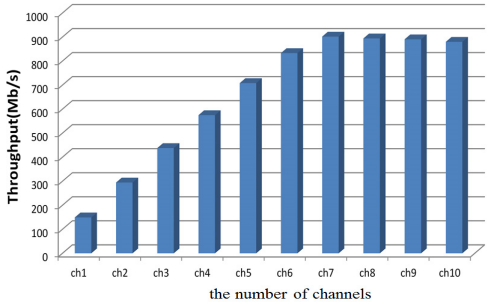

그림 14. 채널수에 따른 FDT 병렬전송
Fig. 14. FDT parallel transfer according to the number of channels

는다. 따라서 이 경우에는 채널수를 6개까지만 측정하였다. 채널수를 6개까지 증가하면서 측정한 결과 채널수 6개에서 가장 높은 425.52Mb/s의 throughput이 측정됨을 확인하였다.

그림 13은 MDTMFTP 병렬전송에 대한 결과를 나타낸다. MDTMFTP의 경우에도 채널수 증가에 따른 비례적인 throughput 증가를 보였다. 채널수 4개에서 이미 최적의 throughput을 보이며 이후에서는 성능이 유지되는 결과를 보인다. 채널 수 4개에서의 성능은 대략 800Mb/s이다. MDTMFTP의 경우 다른 병렬전송도구와 비교해 채널수에 따라 다소 급격하게 throughput이 증가하는 특징을 보인다. 이는 MDTMFTP의 경우 시스템 자원을 활용하여 시스템 병목을 해결한 결과로 판단할 수 있다. MDTMFTP의 경우 멀티코어 기반의 시스템 성능이 충분히 갖추어진 상태라면 더 높은 throughput을 기대할 수 있다.

그림 14의 경우 FDT의 성능결과이다. FDT의 경우에도 채널수에 따라 비례적인 throughput의 증가를 보이며 채널수 7에서 901Mb/s의 throughput을 보인다. 이후 구간에서는 더 증가하지는 않고 조금 감소하는 결과를 확인할 수 있다. 본 논문에서는 FDT의 경우에서 비교 대상인 다른 병렬전송도구들에 비해 채널수에 따라 가장 높은 throughput이 측정됨을 확인하였다.

표 6은 각각의 병렬전송도구에 따라 가장 높은 throughput이 측정되는 채널수를 정리하여 나타낸다.

최적의 성능을 보이는 채널수의 범위는 6개에서 10개 사이임을 확인할 수 있다. 일반적으로 병렬전송채널을 통해 데이터를 전송할 경우 통상적으로 채널수 8개를 전후로 최적의 성능이 측정됨을 경험적인 연구

표 6. 병렬전송도구별 최적 throughput을 보이는 채널 수
Table 6. The number of channel of optimal throughput according to parallel transfer tools

parallel transfer tool	number of channels	throughput (Mb/s)
UDT	9	784.54
Globus-online	6	425.52
MDTMFTP	10	815.85
FDT	7	901.7

를 통해 알려져 있다[16,17]. 본 논문의 병렬전송도구 성능측정결과에서도 범위 이내에서 최적의 성능이 측정됨을 실험결과를 통해 확인할 수 있다.

V. 결 론

네트워크 대역폭이 크고 데이터 전송거리가 긴 WAN 구간에서의 대용량 데이터 전송효율성을 매우 중요한 이슈이다. 특히 관측데이터나 고성능컴퓨터를 활용한 시뮬레이팅 데이터를 생산하는 경우와 같은 연구커뮤니티에서는 전송 성능을 극대화할 수 있는 방안이 필수적이다.

전송프로토콜 및 전송도구 관점에서 TCP 전송성능 한계를 극복하기 위한 주요 방안은 UDP를 사용하거나 다수의 전송채널을 이용하는 병렬전송기법이 대표적이다. UDP를 이용하는 경우에는 사용자 계층에서 추가적인 기능 구현을 통해 TCP와 같은 데이터 전송 안정성 제공이 가능하다. 병렬전송기법의 경우 TCP, UDP 기반 전송프로토콜 기반으로 구현이 가능하다. TCP 기반의 병렬전송의 경우 동시에 다수의 스트림을 보냄으로써 TCP의 응답대기지연을 줄임으로써 전송성능을 획기적으로 개선할 수 있다. 또한, 병렬전송도구의 경우 데이터 전송 채널과 데이터 전송 제어 채널을 분리함으로써 데이터 전송의 안정성과 효율성을 제공한다.

본 논문에서는 대표적인 TCP, UDP 기반 병렬전송도구에 대해 기술하고 전송성능을 측정 및 평가하였다. UDT 병렬전송의 경우 오픈소스기반의 API를 이용해 사용자 계층에서 별도의 기능을 구현하고 전송효율성 극대화를 위한 병렬전송기법 설계가 가능함을 확인하였다. 또한 Globus-online과 FDT의 경우 TCP를 기반으로 한 대표적인 병렬전송도구이며 parallelism, concurrency, pipelining과 같은 병렬기능을 구현함으로써 WAN 구간에서 전송효율성 극대화가 가능함을 확인하였다. MDTMFTP의 경우는 TCP를 기반으로 한 병렬전송도구이면서 데이터 전송 시스템 병목을 고려해 설계된 기법을 사용한다. 이는 CPU나 메모리와 같은 시스템 자원을 활용함으로써 시스템 병목을 제거함으로써 네트워크 가용대역폭을 보다 효율적으로 사용 가능하게 한다.

KREONET과 GLORIAD 네트워크 환경을 이용한 WAN 구간에서의 성능측정결과 모든 병렬전송도구에서 채널 수 증가에 비례적인 성능증가를 보였다. 병렬전송채널을 활용한 결과 데이터 전송거리가 길어지더라도 높은 전송효율성을 보인다. 이처럼 병렬전송채널을 활용할 경우 TCP를 사용하더라도 수신측으로부터 응답대기지연시간을 제거하고 패킷손실에 따른 성능감소를 줄여 가용대역폭을 충분히 활용함을 확인하였다. 병렬전송채널을 활용한 데이터 전송의 최적의 성능은 대략 6개에서 10개 사이에서 가장 높은 성능이 측정됨을 실험을 통해 확인할 수 있었다.

References

[1] E. Dart, L. Rotman, and B. Tierney, "The science DMZ: A network design pattern for data-intensive science," in *Proc. SC'13*, Nov. 2013.

[2] T. Kelly, "Scalable TCP: improving performance in high speed wide area networks," *Comput. Commun. Rev.*, vol. 32, no. 2, Apr. 2003.

[3] D. Kliazovich, F. Granelli, and D. Miorandi, "Logarithmic window increase for TCP westwood+ for improvement in high speed, long distance networks," *Comput. Netw.*, vol. 52, no. 12, pp. 2395-2410, Aug. 2008.

[4] M. Masirap, M. H. Amaran, Y. M. Yussoff, and H. Hashim, "Evaluation of reliable UDP-based transport protocols for Internet of Things (IoT)," in *Proc. ISCAIE*, pp. 200-205, May 2016.

[5] Y. Gu and R. L. Grossman, "UDT: UDP-based data transfer for high-speed wide area networks," *Comput. Netw.*, vol. 51, no. 7, pp. 1777-1799, May 2007.

[6] H. Park, S. Lee, and Y. Shin, "High-speed transmission and control plan on high-definition video file using parallel TCP," in *Proc. ICACT'12*, pp. 1205-1208, 2012.

[7] M. A. Alrshah and M. Othman, "Performance evaluation of parallel TCP, and its impact on bandwidth utilization and fairness in high-BDP networks based on test-bed," in *Proc. 2013 MICC*, pp. 23-28, Nov. 2013.

[8] J. A. Arokkiam, W. Xiuchao, K. N. Brown, and C. J. Ireland, "Experimental evaluation of TCP performance over 10Gb/s passive optical networks(XG-PON)," in *Proc. GLOBECOM*, pp. 2223-2228, Dec. 2014.

[9] M. A. Alrshah and M. Othman, “Performance evaluation of parallel TCP, and its impact on bandwidth utilization and fairness in high-BDP networks based on test-bed,” in *Proc. 2013 MICC*, pp. 23-28, Nov. 2013.

[10] Y. Gu and R. L. Grossman, “UDT: UDP-based data transfer for high speed wide area networks,” *Comput. Netw.*, vol. 51, no. 7, pp. 1777-1799, May 2007.

[11] J. Park, S. Kim, G. Hwang, and G. Cho, “A design and implementation of bulk data transmission tool based on UDT,” *J. IEEK TC*, vol. 49, no. 2, pp. 23-31, 2012.

[12] D. Sulakhe, R. Kettimuthu, and U. Dave, “High-performance data management for genome sequencing centers using globus online: a case study,” in *Proc. E-Science*, pp. 1-6, Oct. 2012.

[13] L. Zhang, W. Wu, P. Demar, and E. Pouyoul, “mdtmFTP and its evaluation on ESNET SDN testbed,” *The J. Future Generation Comput. Syst.*, Elsevier, 2017.

[14] http://monalisa.cern.ch/FDT/

[15] http://fasterdata.es.net

[16] F. Inoue, T. Ito, H. Ohsaki, and M. Imase “Implementation and evaluation of GridFTP automatic parallelism tuning mechanism for long-fat networks,” in *Proc. APSITT*, pp. 189-194, Apr. 2008.

[17] M. A. Alrshah, “Performance evaluation of parallel TCP, and its impact on bandwidth utilization and fairness in high-BDP networks based on test-bed,” in *Proc. MICC*, pp. 23-28, Nov. 2013.

박 종 선 (Jong-seon Park)

2009년 2월 : 조선대학교 전자공학과 졸업
2012년 2월 : 전북대학교 컴퓨터공학과 석사
2015년 8월 : 전북대학교 컴퓨터공학과 박사
2015년 8월~현재 : 한국과학기술정보연구원 선임연구원
<관심분야> 대용량데이터전송, 네트워크성능향상, 센서네트워크, 무선네트워크, 무선네트워크보안

노 민 기 (Min-ki Noh)

1998년 2월 : 공주대학교 기계공학과 졸업
2000년 2월 : 공주대학교 영상매체학과 석사
2009년 2월 : 성균관대학교 컴퓨터교육학과 박사
2000년~현재 : 한국과학기술정보연구원 책임연구원
<관심분야> 네트워크성능향상, 통신공학, 차세대네트워킹

김 승 해 (Seung-hae Kim)

1997년 2월 : 한남대학교 정보통신공학과 졸업
2003년 2월 : 전북대학교 정보과학과 석사
2008년 2월 : 전북대학교 정보보호공학과 박사
1996년~현재 : 한국과학기술정보연구원 책임연구원
<관심분야> 이동컴퓨팅, 컴퓨터통신, 분산처리시스템, 무선네트워크, 무선네트워크보안