KIPS Tr. Comp. and Comm. Sys.
Vol.7, No.2 pp.33~40 pISSN: 2287-5891

https://doi.org/10.3745/KTCCS.2018.7.2.33

Performance Enhancement Method Through Science DMZ Data Transfer Node Tuning Parameters

Park Jong Seon† · Park Jin Hyung†† · Kim Seung Hae††† · Noh Min Ki†††

ABSTRACT

In an environment with a large network bandwidth, maximizing bandwidth utilization is an important issue to increase transmission efficiency. End-to-end transfer efficiency is significantly influenced by factors such as network, data transfer nodes, and intranet network security policies. Science DMZ is an innovative network architecture that maximizes transfer performance through optimal solution of these complex components. Among these, the data transfer node is a key factor that greatly affects the transfer performance depending on storage, network interface, operating system, and transfer application tool. However, tuning parameters constituting a data transfer node must be performed to provide high transfer efficiency. In this paper, we propose a method to enhance performance through tuning parameters of 100Gbps data transfer node. With experiment result, we confirmed that the transmission efficiency can be improved greatly in 100Gbps network environment through the tuning of Jumbo frame and CPU governor. The network performance test through Iperf showed improvement of 300% compared to the default state and NVMe SSD showed 140% performance improvement compared to hard disk.

Keywords : End to End Transfer Efficiency, Science DMZ, 100Gbps Data Transfer Node, Tuning Parameters, NVMe SSD

Science DMZ 데이터 전송 노드 튜닝 요소를 통한 성능 향상 방안

박 종 선† · 박 진 형†† · 김 승 해††† · 노 민 기†††

요 약

네트워크 대역폭이 큰 환경에서는 대역폭 활용률을 극대화함으로써 전송효율성을 높이는 것이 매우 중요한 이슈이다. 종단간 전송효율성은 네트워크, 데이터 전송 노드 그리고 기관 내 네트워크 보안정책 등 구성요소에 따라 크게 영향을 받는다. Science DMZ는 이러한 복합적인 구성요소들의 최적의 해결 방안을 통해 전송성능을 극대화하기 위한 혁신적인 네트워크 구조이다. 이 중에서 데이터 전송 노드는 스토리지, 네트워크 인터페이스, 운영체제, 전송응용 도구에 따라 전송성능에 크게 영향을 주는 핵심 요소이다. 하지만 고속네트워크 환경에서는 데이터 전송 노드를 구성하는 요소들의 적절한 튜닝이 수행되어야 높은 전송효율성을 제공할 수 있다. 본 논문에서는 100Gbps 데이터 전송 노드의 튜닝 요소를 통한 전송성능 향상 방안에 대해 제안한다. 성능측정결과 점보프레임, CPU governor 튜닝을 통해 100Gbps 네트워크 환경에서 전송효율성을 크게 개선할 수 있음을 확인하였다. Iperf를 통한 네트워크 성능테스트 결과 default에 비해 300%의 성능향상을 보였으며 NVMe SSD의 경우 하드디스크와 비교해 140%의 성능개선을 확인하였다.

키워드 : 종단간 전송효율성, Science DMZ, 100Gbps 데이터 전송 노드, 튜닝 요소, NVMe SSD

1. 서 론

네트워크 기술 및 고속네트워크 인프라 확충은 종단 간 사용자 전송성능에 대한 기대를 높이고 있다. 이러한 전송성능 즉, 전송효율성은 거대과학데이터를 생산하는 연구커뮤니티의 경우 민감하다. 고에너지물리, 천체/우주, 기상/기후 등 실험 장비 및 관측 장비를 통해 생산한 데이터의 크기는 매우 방대하다. 이러한 데이터를 이용해 연구자 간 글로벌 협업연구를 수행하기 위해서는 신속한 데이터 전송이 우선시된다. 하지만 구축한 네트워크 대역폭에 비해 사용자가 얻는 실질적인 전송속도는 기대에 미치지 못한다. 이에 대한 원인은 범용적인 네트워크 사용, 데이터 전송 노드 그

※ 이 논문은 2018년도 한국과학기술정보연구원(KISTI) 주요사업 과제로 연구되었음.
† 정 회 원 : 한국과학기술정보연구원 첨단연구망서비스실 선임연구원
†† 정 회 원 : 한국과학기술정보연구원 정보시스템운영실 선임연구원
††† 정 회 원 : 한국과학기술정보연구원 첨단연구망서비스실 책임연구원
Manuscript Received : January 22, 2018
Accepted : January 30, 2018
* Corresponding Author : Noh Min Ki(mknoh@kisti.re.kr)

리고 기관 내 네트워크 방화벽 정책으로부터 기인한다. 따라서 사용자 전송성능을 높이기 위해서는 네트워크 구성 요소에 대한 복합적인 접근이 필요하다.

Science DMZ[1]는 이러한 복합적인 접근 요소를 통해 데이터 전송효율성을 극대화하기 위해 제안된 혁신적인 네트워크 구조이다. Science DMZ는 전형적인 DMZ 네트워크 구조처럼 특정 서비스를 위한 격리된 구역으로 볼 수 있다. 과학데이터처럼 거대한 과학데이터 전송을 위해 WAN (Wide Area Network)이 연동되는 경계에 데이터 전송노드를 둠으로써 기관 내 방화벽을 우회함으로써 성능감소를 제거할 수 있다. 대용량 데이터 전송 성능을 고려해 설계된 Science DMZ 구성요소는 크게 전용 네트워크, 데이터 전송 노드, 방화벽 정책 그리고 성능 모니터링 시스템 등의 복합적인 구성요소를 포함한다.

데이터 전송 노드는 전용망을 기반으로 데이터 전송을 위한 시스템으로써 전송효율성을 극대화할 수 있는 Science DMZ의 핵심 요소이다[2]. 따라서 10Gbps 이상 100Gbps의 고속 네트워크 환경에서 데이터 전송 노드 활용을 위해서는 적정 수준 이상의 규격이 요구된다. 하지만 네트워크 대역폭을 충분히 활용하기 위해서는 별도의 튜닝 요소에 대한 접근이 필요하다. 데이터 전송 노드 튜닝 요소는 매우 다양하며 대역폭이 클수록 네트워크 인터페이스 카드, CPU governor, 인터럽트 어피니티, 소켓버퍼크기, 점보프레임에 대한 튜닝은 필수적이다.

본 논문에서는 100Gbps 데이터 전송 노드 튜닝 요소를 통한 성능 향상 방안에 대해 제안한다. 100Gbsp 데이터 전송 노드를 통한 충분한 대역폭 활용을 위해 핵심 튜닝 요소인 인터페이스 카드, CPU governor, 인터럽트 어피니티, 소켓버퍼크기, 점보프레임에 대한 설정 방안에 대해 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 대용량 데이터 전송 연구들에 대해 살펴본다. 3장에서는 100Gbps 데이터 전송 노드 튜닝 요소 설정 방안에 대해 제안한다. 4장에서는 제안기법에 대한 성능을 평가하고 5장에서 본 논문의 결론을 맺는다.

2. 대용량 데이터 전송에 관한 연구

2.1 TCP 기반 대용량 데이터 전송

1980년대 제안된 TCP(Transmission Control Protocol)는 사실상 표준(de facto) 프로토콜로써 지금껏 대부분 전송응용에 사용되어 왔다. 하지만 네트워크 대역폭의 급격한 확장과 글로벌 협업연구 환경이 보편화됨에 따라 전송성능 이슈가 끊임없이 발생하고 있다. 즉, 대역폭이 크고 전송거리가 긴 네트워크 환경에서 TCP는 매우 비효율적인 성능을 보인다. 이는 TCP 전송메커니즘에 원인이 있다. TCP는 패킷손실을 네트워크 혼잡으로 보며 패킷손실 이후 성능을 급격하게 줄인다. 그리고 매번 RTT(Round Trip Time) 마다 수신측으로부터 보낸 패킷에 대한 응답을 확인하고 조심스럽게 다음 전송 패킷의 양을 늘린다. 이러한 소극적인 전송 메커니즘은 데이터 전송거리가 길수록 대역폭 활용률을 낮게 만든다.

최근 TCP 기반 대용량 데이터 전송에 관한 연구들의 경우 네트워크 상태를 미리 예측한 후 전송 패킷의 양을 결정한다. 이는 패킷손실에 매우 민감한 TCP의 성능 개선을 위한 방안이다. 패킷손실을 제거하는 것이 곧 TCP의 전송효율성을 극대화하는 방안이 된다. TCP-Vegas[6], FastTCP [7], HTCP, High Speed TCP[8]와 같은 혼잡제어 알고리즘이 네트워크 상태를 미리 예측하는 기법을 적용한 대표적인 예이다. 그럼에도 불구하고 패킷손실로 인한 TCP의 민감한 대처로 인해 고 대역폭의 WAN 환경에서는 기대 수준의 성능을 얻기 어렵다.

2.2 UDP 기반 대용량 데이터 전송

UDP(User Datagram Protocol) 기반 전송 프로토콜은 고 대역폭 WAN 환경에서의 성능개선을 위한 TCP의 대안으로 제안되었다[9]. UDP는 전송메커니즘 특성 상 별도의 혼잡제어나 오류검사를 수행하지 않기 때문에 연속적인 데이터 전송이 가능하다. 즉, 매우 공격적인 전송이 가능하기 때문에 WAN 환경에서도 높은 전송효율성을 보인다. UDP 기반 전송프로토콜은 이런 UDP 특성을 더해 응용계층에서 TCP처럼 별도의 데이터 전송 신뢰성 기능을 추가적으로 제공한다. 대표적인 UDP 기반 전송프로토콜로써 RBUDP(Reliable Blast UDP[10], Tsunami[11], SABUL(Simple Available Bandwidth Utilization Library, UDT(UDP based Transfer) [12]를 들 수 있다. TCP 대안으로 제안된 UDP 기반 전송 프로토콜의 특징을 다음과 같이 요약할 수 있다.

- 일정 주기 동안 데이터 전송: TCP가 매번 RTT를 확인하고 다음 패킷을 전송하는 반면 UDP 기반 전송프로토콜은 일정 시간동안 연속적으로 패킷을 전송한다. UDT의 경우 10ms 마다 RTT를 확인한다.
- 패킷 손실 처리: TCP는 매번 RTT 마다 패킷 시퀀스를 확인하고 재전송을 수행하는 반면 UDP 기반 전송프로토콜은 일정 시간 마다(10ms) 일괄적으로 손실된 패킷에 대해 재전송을 수행한다.
- 혼잡제어: TCP는 손실 이후 패킷의 양을 급격하게 줄이고 RTT 마다 소극적으로 전송 패킷의 양을 늘리는 반면 UDP 기반 전송프로토콜은 가용한 대역폭을 최대한 채우기 위해 전송 패킷의 양을 급격하게 증가시킨다.

이러한 공격적인 전송메커니즘을 통해 UDP 기반 전송프로토콜은 고 대역폭의 WAN 환경에서 높은 전송성능을 보인다.

2.3 병렬 데이터 전송

병렬전송기법은 다수 채널을 이용해 보다 공격적인 데이터 전송을 가능케 한다[13]. 단일채널을 통한 TCP의 경우 보낸 패킷에 대한 응답 수신을 위해 대기지연이 발생하게 된다. 병렬전송은 이러한 대기지연을 제거함으로써 연속적인 데이터 전송이 가능하다. 이는 UDP 기반 전송프로토콜이 일정 시간마다 데이터를 연속적으로 전송하는 방법과 매우 유사하다고 볼 수 있다. GridFTP[14]는 매우 보편적인 병렬전송기법이며 최근에는 Globus-online이나 MDTMFTP (Multicore-Aware Data Transfer Middleware), FDT(Fast Data Transfer), Parallel UDT[15]가 대표적이다.

2.4 Science DMZ

종단 간 사용자의 전송성능을 높이기 위해서는 네트워크를 구성하는 모든 요소에 대한 복합적인 접근이 필요하다. Science DMZ는 고 대역폭 WAN 환경에서 전송성능을 극대화하기 위한 네트워크 구조이며 최적화된 네트워크 구성 요소를 기반으로 대역폭 활용률 극대화에 목적이 있다. Science DMZ 구축을 위한 구성 요소는 다음과 같이 요약할 수 있다.

- 네트워크 구조: Science DMZ 네트워크 구조는 전용 네트워크, 최소한의 보안정책을 기반으로 한다. 전용 네트워크는 기존에 구축된 네트워크의 재활용이 가능해야하며 가상회선(virtual circuit) 등을 이용해 구축이 가능하다. 네트워크 보안정책은 기관 내 보안시스템을 우회시키고 최소한의 ACL(Access Control List)을 적용한다.
- 데이터 전송 노드: 데이터 전송 노드는 전송을 위한 서버이며 전용 네트워크의 최대한 활용을 위해 높은 하드웨어 규격이 요구된다.
- 성능모니터링: 네트워크 구간에 대한 모니터링이 수행되어야하며 하드웨어 및 소프트웨어로 인한 전송성능 감소에 즉각적인 조치가 가능해야 한다.

이 중에서 데이터 전송 노드는 스토리지, 전송도구 선택에 따라 전송성능을 크게 개선할 수 있는 핵심 구성요소이다. 전송도구의 경우 전송성능극대화를 위해 병렬전송도구 활용이 필수적이다. 또한 기가급 이상 WAN 구간에서 성능을 고려하면 시스템 튜닝이 필수적으로 수반되어야 한다. 데이터 전송 노드 튜닝은 시스템 관련된 모든 요소가 대상이 된다. 이 중에서 네트워크 인터페이스 카드, CPU governor, 인터럽트 어피니티 및 바인딩, 소켓버퍼크기, 점보프레임설정은 기가급 WAN 환경에서 성능 극대화를 위해 꼭 필요한 튜닝 요소이다.

3. 데이터 전송 노드 튜닝 요소를 통한 성능 개선

Science DMZ 네트워크 구조 기술 중 데이터 전송 노드의 충분한 활용을 위해서 시스템 튜닝은 필수적이다. 특히 100Gbps 대역폭을 가진 WAN 환경에서는 대역폭 활용률을 극대화하기 위해 튜닝 요소를 통한 적절한 설정이 요구된다. 이번 장에서는 100Gbps 데이터 전송 노드 튜닝 요소를 통한 성능 개선 방안에 대해 기술한다.

데이터 전송 노드의 튜닝을 수행하기에 앞서 적절한 하드웨어 규격을 선택이 요구된다. 하드웨어 규격은 스토리지, 네트워크 인터페이스 카드, CPU, 메인보드 등 적절한 구성이 필요하다.

스토리지의 경우 데이터를 읽고 쓰는 성능에 따라 전송성능에 크게 영향을 미친다. 스토리지는 크게 인피니밴드나 이더넷을 통해 데이터 전송 노드와 연동하는 방법과 로컬스토리지 구성을 통해 활용하는 방법이 있다. 로컬스토리지의 경우 레이드(raid) 구성을 통해 보다 유연성 있는 구성이 가능하기 때문에 현실적인 방안이 될 수 있다. 레이드 구성은 하드디스크, PCIe 타입 SSD, NVMe 타입 SSD를 통해 가능하다[3, 4]. 100Gbps 대역폭의 충분한 활용을 위해서는 NVMe 타입 SSD를 활용이 요구된다[5]. Fig. 1은 NVMe SSD와 PCIe SSD 스택을 비교하여 나타낸다. NVMe 타입 SSD 경우 커널 스택에서 지연시간이 줄어들어 상대적인 전송성능 개선을 기대할 수 있다.

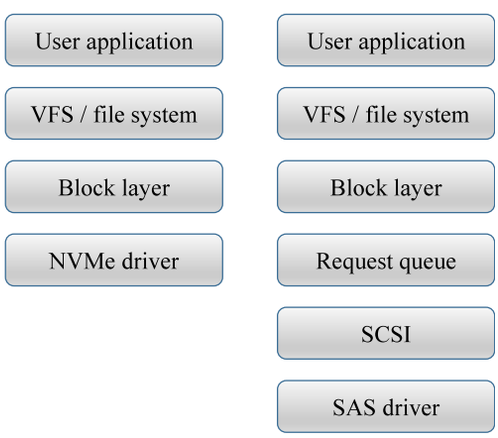

Fig. 1. Comparison of NVMe SSD Stack and PCIe SSD Stack

스토리지 레이드 구성은 레벨에 따라 데이터 전송 효율성과 안정성을 제공한다. 레이드 구성을 위해서는 별도의 컨트롤러가 필요하며 안정성을 제공하는 하드웨어 컨트롤러를 통한 레이드 구성이 요구된다. 레이드 벨은 일반적으로 0번부터 6번을 통해 구성한다. 레벨 0의 경우 스트라이핑 기술을 통해 전송효율성을 제공하며 레벨 5, 6의 경우 데이터 전송 안정성을 제공한다. 본 논문에서 제안하는 데이터 전송노드는 별도의 컨트롤러가 없는 NVMe SSD를 기반으로 한다.

Table 1. Hardware Specification of Proposed 100Gbps Data Transfer Node

Hardware	Description
CPU	- Intel Xeon E5-2667 v4 3.2GHz, 25M Cache,9.60GT/sQPI
File system	- OS: 200GB Solid State Drive - Data transfer: IDell 1.6TB, NVMe, Mixed Use Express Flash
Network interface card	- Data transfer: Mellanox ConnectX-4 Dual Port QSFP28 Network Adapter(100G)
Memory	128GB(16GB RDIMM, 2400MT/s(8ea))

메인보드의 경우 CPU, 메모리, 네트워크 인터페이스 카드 연동을 위한 버스를 제공한다. 메인보드 선택 시 PCIe 타입, 메모리 확장 규모, 아키텍처, 칩셋은 주요 고려사항이다. PCIe의 경우 통상적으로 PCIe 2.0과 PCIe 3.0이 사용되며 읽기 속도는 16레인 기준으로 PCIe 2.0의 경우 6GB/s이며 PCIe 3.0의 경우 16GB/s의 속도를 제공한다. 칩셋의 경우 인텔이 QPI 버스를 통해 AMD에 비해 상대적으로 높은 속도를 제공한다. 본 논문에서 제안하는 100Gbps 데이터 전송 노드 하드웨어 규격은 Table 1과 같다.

고 대역폭의 WAN 환경에서 전송 성능 개선을 위하 시스템 튜닝 요소는 네트워크 인터페이스 카드, CPU governor, 인터럽트 어피니티, 소켓버퍼크기, 혼잡제어알고리즘, 점보프레임이 필수적인 요소이다.

네트워크 인터페이스 카드 인터럽트를 얼마나 효율적으로 처리하는지가 전송성능에 크게 영향을 미친다. 고 대역폭에서는 처리할 인터럽트의 양(IRQ: Interrupt ReQuest)이 크고 사용자 공간으로의 데이터 복사도 매우 크다. 따라서 네트워크 인터페이스 카드 링버퍼 크기를 크게 설정해 주어야 한다. 링버퍼 크기의 경우 기본 설정 값이 작게 설정되어 있기 때문에 네트워크 인터페이스 카드에서 지원하는 최대 크기로 설정해 주어야 한다. 리눅스의 경우 아래 명령어를 통해 최대 지원 가능한 링버퍼 크기와 현재 설정된 링버퍼 크기를 확인할 수 있다.

#ethtool -g “ethN” (조회)

#ethtool -G ethN rx “value” tx “value” (설정)

CPU governor는 CPU 활용률 최대한 가능하게 하는 시스템 튜닝 요소이다. 기본적으로 CPU 설정을 보면 power save 모드로 설정되어 있으며 이 외에도 on demand, conservative, performance 설정모드가 있다. power save는 배터리 소모를 최소화하기 위한 모드로 낮을 클럭속도로 동작하는 방법이다. on demand의 경우 제일 낮은 클럭속도로 동작하다가 프로세스 실행 시에 클럭속도를 높이는 모드이다. conservative는 단계별로 클럭속도를 증가 및 감소시켜주는 모드이다. 성능을 극대화하기 위해서는 아래 명령어를 통해 설정을 performance mode로 변경해 주어야 한다. Rhel 및 Debian 등 운영체제에 따라 설정 및 변경이 가능하다.

#cpupower frequency-set -g performance (Rhel)

#cpufreq-set -r -g performance (Debian)

최근 데이터 전송 노드의 경우 멀티코어를 지원한다. 멀티코어 경우 전송성능을 위해 NUMA(Non Uniform Memory Access) 이슈가 고려되어야 한다. NUMA는 각각의 프로세서들이 할당된 메모리를 배정받는 방식의 아키텍처를 제공한다. Fig. 2는 CPU가 두 개 이상인 상태에서 각각의 CPU가 여러 개의 코어로 구성된 상태를 나타낸다. 기존 아키텍처의 경우 하나의 코어가 메모리에 접근할 경우 나머지 코어들은 대기상태에 있게 된다. NUMA 아키텍처의 경우 Fig. 2의 점선처럼 각각의 코어들이 할당된 메모리에 접근 가능하기 때문에 대기지연상태가 제거된다. 하지만 전송효율성을 고려하면 NUMA 아키텍처의 튜닝이 필요하다. Fig. 2의 빨간색처럼 네트워크 인터페이스 카드 인터럽트 처리가 버스를 거치지 않고 해당 CPU(코어들)에서 수행되어야 추가적인 지연시간을 제거할 수 있다. 이를 위해서는 먼저 interrupt balance 기능을 중지하고 특정 프로세스가 할당된 코어로 배정되도록 설정해 주어야 한다. 본 논문에서는 해당 네트워크 인터페이스 카드의 인터럽트를 처리하는 CPU(코어들)를 확인 후 인터럽트 balance 기능을 종료한다. 그리고 인터럽트 바인딩을 통해 해당 CPU의 코어를 사용하도록 한다.

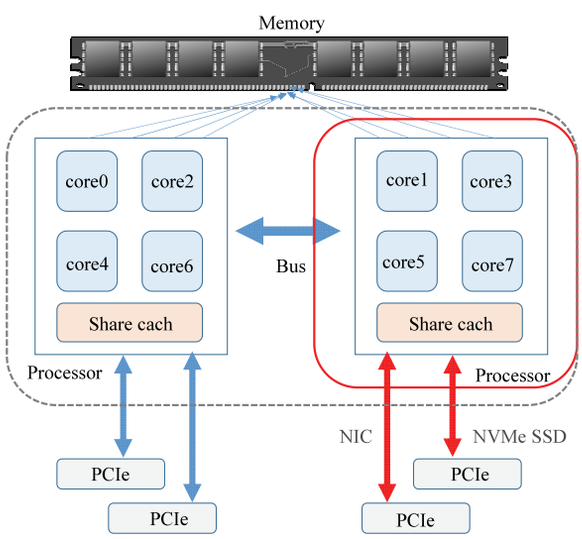

Fig. 2. NUMA Architecture and Interrupt Binding

#chkconfig irqbalance off (irq balance 종료)

#cat /sys/class/net/ethN/device/numa_node(인터럽트처리 cpu 조회)

#cat /proc/irq/32/smp_affinity (cpu 어피니티 조회)

#echo “proc_number” > /proc/irq/irq_number/smp_affinity (인터럽트 처리를 위한 코어 설정)

```
cpu MHz         : 2334.375      cpu MHz         : 1680.125
cpu MHz         : 3502.250      cpu MHz         : 3500.625
cpu MHz         : 1787.875      cpu MHz         : 1549.000
cpu MHz         : 3502.375      cpu MHz         : 3524.500
cpu MHz         : 1879.875      cpu MHz         : 1618.750
cpu MHz         : 3504.250      cpu MHz         : 3501.750
cpu MHz         : 2093.125      cpu MHz         : 1888.125
cpu MHz         : 3500.750      cpu MHz         : 3543.125
```

Fig. 3. CPU Core Utilization After Interrupt Binding

Fig. 3은 아래 명령어를 통해 확인한 CPU 활용률을 나타낸다. 16개 코어에 대해 1, 3, 5, 7, 9, 11, 13, 15 코어의 클럭이 오버클럭되는 것을 확인할 수 있다.

#cat /sys/class/net/ethN/device/local_cpulist

소켓버퍼크기 튜닝은 고 대역폭 WAN 환경에서 필수적인 튜닝 요소이다. 소켓버퍼크기는 전송패킷의 양을 적절하게 조절해주는 것을 의미한다. 리눅스의 경우 소켓버퍼의 크기는 기본적으로 설정되어 있지 않다. 따라서 고속네트워크 환경에서 전송효율성을 높이기 위해서는 소켓버퍼의 크기를 적절한 수준까지 높여 주어야 한다. Fig. 4는 운영체제에 따른 소켓버퍼크기 조절을 나타낸다. 리눅스의 경우 /etc/sysctl.conf 파일을 통해 커널 파라미터의 수정이 가능하다. 여기서 적절한 값은 통상적으로 네트워크 대역폭과 데이터 전송거리의 곱(Bandwidth Delay Product)으로 계산한다.

```
Linux 3.10.0: add to /etc/sysctl.conf
        net.core.rmem_max = 1,250,000,000
        net.core.wmem_max = 1,250,000,000
        # auto tuning min, default, max and max number of bytes to use
        net.ipv4.tcp_rmem = 4096 87830 1,250,000,000
        net.ipv4.tcp_wmem = 4096 65536 1,250,000,000

FreeBSD: add to /etc/sysctl.conf
        net.inet.tcp.sendbuf_max = 1,250,000,000
        net.inet.tcp.recvbuf_max = 1,250,000,000

Mac OSX: add to /etc/sysctl.con
        kern.ipc.maxsockbuf = 1,250,000,000
        net.inet.tcp.sendspace = 625,000,000
        net.inet.tcp.recvspace = 625,000,000
```

Fig. 4. Socket Buffer Size Based on Operating System

예를 들어 대역폭이 100Gbps이고 데이터 전송거리가 100ms인 WAN 환경을 고려하면 최적의 값은 아래와 같다.

◦ BDP(bit/s)
100,000,000,000bps × 0.1 second = 10,000,000,000
◦ BDP(Byte/s)
100,000,000,000 / 8bits=1,250,000,000

점보프레임 튜닝은 데이터 전송 노드 간 MTU(Maximum Transmission Unit) 설정을 통해 전송속도를 높이기 위한 보편적인 방안이다. 기본적인 MTU 크기는 1500Byte로 설정되어 있다. 점보프레임의 크기는 최대 9000Byte까지 높게 설정할 수 있다. 점보프레임 설정 시 고려되어야 할 이슈는 데이터 전송 노드를 포함한 모든 네트워크 장비들이 9000Byte로 설정되어야 한다는 것이다. 만약 어떠한 노드가 1500Byte로 설정되어 있다면 데이터 전송 노드에서 설정한 점보프레임 설정이 의미가 없어진다. 리눅스의 경우 점보프레임 설정은 아래 명령어를 통해 설정 가능하다.

#vi /etc/network/interfaces (Debian)
#vi /etc/sysconfig/network-scripts/ifcfg-“ethN” (Rhel)

4. 성능 평가

데이터 전송 노드 튜닝을 통한 데이터 전송성능을 확인하기 위해 Table 1과 같은 100G 네트워크 인터페이스 카드를 장착한 서버 두 대를 Fig. 5와 같이 직접 연결한다. 100Gbps 네트워크 환경 구성은 KREONET을 통해 서울, 대전 구간에 VLAN 구성을 통해 구성한다. 운영체제는 Centos 7.3 버전을 설치하며 네트워크 인터페이스 카드는 Mellanox ConnectX-4 EN/VPI 100G NICs(EN mode)를 사용한다. 그리고 네트워크 인터페이스 카드 드라이버는 Mellanox OFED Driver mlnx-en-4.1-1.0.2.0-rhel7.3-x86_64을 설치한다. 데이터 전송 노드 간 전송성능은 iperf를 통해 네트워크 성능을 측정한 후 NVMe SSD 카드의 전송성능을 측정하여 하드디스크의 성능과 비교한다.

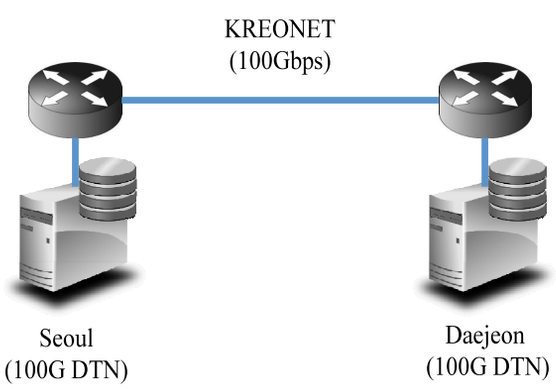

Fig. 5. Test Environment for 100G dtn

네트워크 성능은 먼저 데이터 전송 노드의 튜닝이 없는 default 상태에서 성능을 측정하고 이후 점보프레임, CPU governor, 소켓버퍼크기를 순차적으로 변경하면서 네트워크 성능을 측정한다. 그리고 병렬 스트림 채널의 수를 1개부터 8개까지 증가시키면서 채널수에 따른 성능을 측정한다.

Fig. 6은 점보프레임, CPU governor 튜닝 요소에 따른 전송성능을 채널수를 증가시키면서 측정한 결과를 비교하여

나타낸다. 성능은 iperf를 이용해 측정한 네트워크 성능테스트 결과이다. default 상태에서는 채널수 1개에서 대략 10Gbps 성능이 측정된다. 하지만 채널수를 8개까지 증가시켜도 대역폭을 충분히 활용하지 못하는 결과를 보인다. MTU 크기를 9000Byte로 설정했을 때 네트워크 전송성능이 크게 개선되는 것을 확인할 수 있다. 채널수 1개에서 36Gbps이며 default와 비교해 대략 3배 이상 높은 결과를 보인다. 채널수를 8개까지 증가시켰을 때 98Gbps로 대역폭을 거의 포화시키는 결과를 보인다. CPU governor 설정을 했을 때는 점보프레임을 설정했을 때와 유사한 결과를 보인다. 하지만 채널 수 4개정도에서 이미 대역폭을 거의 포화시키는 결과의 차이를 보인다.

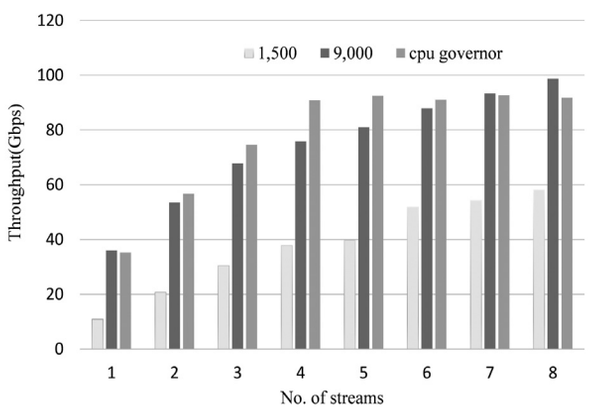

Fig. 6. Throughput Result Comparison According to the Number of Streams and Tuning Parameters

Fig. 7은 점보프레임 및 CPU governor 설정에 따른 파일전송성능을 비교한 그래프이다. 파일전송은 병렬전송도구인 FDT를 이용해 측정한 결과이다. 파일전송테스트 측정을 위해 100GB 파일을 전송 후 그 결과를 비교한 그래프이다.

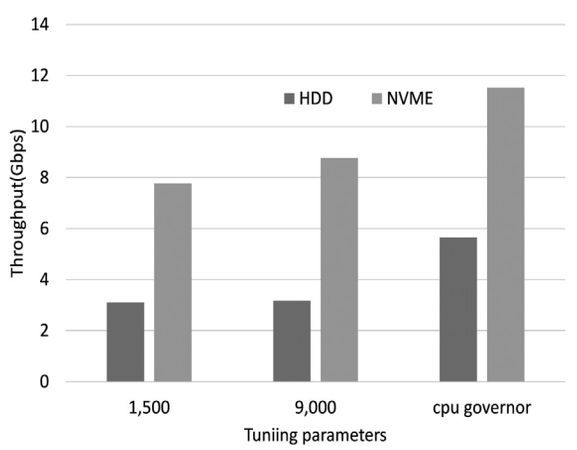

Fig. 7. Throughput Result Comparison of Hard Disk and NVMe SSD According to Tuning Parameters

Default 상태에서 하드디스크는 3Gbps를 보이며 NVMe SSD의 경우 대략 7Gbps의 전송성능을 보인다. 점보프레임 설정의 경우 거의 비슷한 결과를 보이며 튜닝에 따른 영향이 거의 없다. CPU governor를 설정했을 때 하드디스크는 5.6Gbps를 보이며 NVMe SSD는 11.6으로 성능이 개선되었음을 확인할 수 있다. 파일전송테스트의 경우 CPU governor에 따른 영향이 큰 것을 확인할 수 있다. Fig. 8은 하드디스크 및 NVMe SSD의 읽고 쓰기 속도를 비교하여 나타낸다. 읽기 속도의 경우 거의 유사한 결과를 보이지만 쓰기 속도에서는 NVMe SSD의 지연속도가 확연하게 줄어드는 것을 확인할 수 있다.

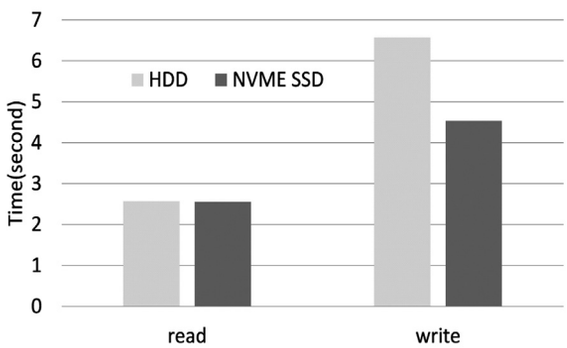

Fig. 8. Read/Write Delay Test Comparison of HDD and NVMe SSD

5. 결 론

고 대역폭의 WAN 환경에서는 네트워크 대역폭을 충분히 활용하지 못하는 이슈가 있다. 이처럼 고속네트워크 환경에서는 데이터 전송 노드의 역할이 중요하며 튜닝을 통해 전송효율성을 개선할 수 있다.

본 논문에서는 고속네트워크 환경에서 데이터 전송 노드 튜닝을 통한 성능개선 방안에 대해 제안하였다. 이러한 튜닝요소는 네트워크 인터페이스 카드, CPU governor, 인터럽트 어피니티, 소켓버퍼크기, 점보프레임 등 매우 다양하다. 그리고 이러한 튜닝요소를 네트워크 상태에 따라 적정한 값으로 설정하는 것은 매우 중요하다. 이러한 튜닝요소는 일반적으로 네트워크 대역폭, 데이터 전송거리, 데이터 전송노드의 규격에 따라 적정한 값으로 설정되어야 한다.

데이터 전송 노드 튜닝에 따른 성능을 측정하기 위해 KREONET을 이용해 100Gbps 서버 두 대를 연결하고 스트림 수를 증가시키면서 전송성능을 측정하였다. Iperf를 통한 네트워크 성능테스트 결과 튜닝을 하지 않은 상태와 비교해 300%의 성능향상을 보였으며 NVMe SSD의 경우 하드디스크와 비교해 140%의 성능개선을 확인하였다.

본 논문에서는 NVMe SSD를 이용해 전송성능을 측정하였다. 향후 연구에서는 100Gbps WAN 환경에서 NVMe 컨트롤러를 사용해 레이드를 구성하고 그에 따른 성능측정이 추가적으로 필요하다.

References

[1] D. Eli, R. Lauren, T. Brian, H. Mary, and Z. Jason, "The Science DMZ: A network design pattern for data-intensive science," in *Proceeding of the International Conference on High Performance Computing, Networking, Storage and Analysis*, 2013.

[2] V. Nagendra, V. Yegneswaran, and P. Porras, "Securing ultra-high-bandwidth science DMZ networks with coordinated situational awareness," in *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*, 2017.

[3] S. Kim and J. Yang, "Work-in-progress: improving NVMe SSD I/O determinism with PCIe virtual channel," in *Proceeding of International Conference on Compilers, Architectures and Synthesis For Embedded Systems*, 2017.

[4] J. Qian, H. Jiang, S. A. Witawas, S. Seth, S. Skelton, and J. Moore, "Energy-Efficient I/O Thread Schedulers for NVMe SSDs on NUMA," in *Proceeding of International Symposium on Cluster, Cloud and Grid Computing*, 2017.

[5] Q. Xu et al., "Performance analysis of NVMe SSDs and their implication on real world databases," in *Proceedings of the 8th ACM International Systems and Storage Conference*, 2015.

[6] K. Mesmin and J. Mbyamm, J. Zhang, "Improved implementation of TCP-vegas method in interchanges of satellite links," in *Proceeding of International Conference on Computer Science and Network Technology*, 2016.

[7] J. Cheng, D. X. Wei, and S. H. Low, "FAST TCP: motivation, architecture, algorithms, performance," in *Proceeding of INFOCOM on IEEE Computer and Communications Societies*, 2004.

[8] S. Floyd, "HighSpeed TCP for large congestion windows," IETF, RFC3649, 2003.

[9] Q. Liu, N. Rao, and C. Q. Wu, "Measurement-based performance profiles and dynamics of UDT over dedicated connections," in *Proceeding of nternational Conference on Network Protocols*, 2016.

[10] E. He, J. Leigh, O. Yu, and T. A. DeFanti, "Reliable Blast UDP : predictable high performance bulk data transfer," in *Proceeding of IEEE International Conference on Cluster Computing*, 2002.

[11] M. Meiss, "Tsunami: a high-speed rate-controlled protocol for file transfer," www.evl.uic.edu/eric/atpTSUNAMI.pdf, 2009.

[12] Y. Gu, "UDT: a high performance data transport protocol," Ph.D. Thesis, Laboratory for Advanced Computing, Univ. of Illinois at Chicago, 2005.

[13] E. Jung, R. Kettimuthu, and V. Vishwanath, "Toward optimizing disk-to-disk transfer on 100G networks," in *Proceeding of 2013 IEEE International Conference on Advanced Networks and Telecommuncations Systems*, 2013.

[14] F. Inoue, T. Ito, H. Ohsaki, and M. Imase, "Implementation and evaluation of GridFTP automatic parallelism tuning mechanism for long-fat, networks," in *Proceeding of Asia-Pacific Symposium on Information and Telecommunication Technologies*, 2008.

[15] J. Park, D. An, and G. Cho, "An adaptive channel number tuning mechanism on parallel transfer with UDT," in *Proceeding of 2013 International Conference on Information Networking*, 2013.

박 종 선

http://orcid.org/0000-0002-1831-8477
e-mail : jspark@kisti.re.kr
2009년 조선대학교 전자공학과(학사)
2012년 전북대학교 컴퓨터공학과(석사)
2015년 전북대학교 컴퓨터공학과(박사)
2015년～현 재 한국과학기술정보연구원 첨단연구망서비스실 선임연구원
관심분야 : 대용량데이터전송, 네트워크성능향상, 센서네트워크, 무선네트워크, 무선네트워크보안

박 진 형

http://orcid.org/0000-0002-9710-9728
e-mail : ntoskr@kisti.re.kr
2009년 경북대학교 전자전기컴퓨터학부 (학사)
2012년 한국대학교 정보통신학과
1996년～현 재 한국과학기술정보연구원 정보시스템운영실 선임연구원
관심분야 : 머신러닝 알고리즘, 분산처리시스템, 정보보안관리체계, 무선네트워킹보안, 네트워크 페더레이션

김 승 해

http://orcid.org/0000-0002-8403-7577
e-mail : shkim@kisti.re.kr
1997년 한남대학교 정보통신공학과(학사)
2003년 전북대학교 정보공학과(석사)
2008년 전북대학교 정보공학과(박사)
1996년～현 재 한국과학기술정보연구원 첨단연구망서비스실 책임연구원
관심분야 : 이동컴퓨팅, 컴퓨터통신, 분산처리시스템, 무선네트워크, 무선네트워크보안

노 민 기

http://orcid.org/0000-0003-0144-7253
e-mail : mknoh@kisti.re.kr
1998년 공주대학교 기계공학과(학사)
2000년 공주대학교 영상매체학과(석사)
2009년 성균관대학교 컴퓨터교육학과 (석사)
2010년～현 재 한국과학기술정보연구원 첨단연구망서비스실 책임연구원
관심분야 : 네트워크성능향상, 통신공학, 차세대네트워킹