데이터센터 네트워크의 지연시간 분석

오상훈, 심재균, 이석한, 안정호
서울대학교 융합과학부
e-mail: {shine.oh, acvbf7sim7, infy1026, gajh}@snu.ac.kr

An Analysis of Network Latency on Datacenters

Sanghoon Oh, Jaekyun Shim, Sukhan Lee, Jung Ho Ahn
Department of Transdisciplinary Studies, Seoul National University

요 약

유무선 네트워크 기술의 급격한 발전 및 개인용 휴대 장치의 보급 증가와 그에 맞물린 소셜네트워크 서비스 등의 활성화로 인해 데이터의 전송량이 폭발적으로 늘어나고 있으며, 이러한 추세는 향후 지속될 것으로 전망된다. 이에 데이터센터의 수요가 증가하고 있으며, 확장 및 유지보수가 쉬우면서도 높은 성능을 갖는 시스템에 대한 요구가 높아지고 있다. 본 논문에서는 데이터센터 네트워크의 구성요소를 분석하고 각각의 지연시간을 알아보았으며, 그 중 확장 및 유지보수가 쉬우나 상대적으로 지연시간이 높은 이더넷을 데이터센터에 적용하였을 때 지연시간 개선을 위한 방법으로 Intel DPDK를 적용할 경우, 미적용시와 비교하여 약 86%의 지연시간 감소를 확인하였고, 추가 향상 방안을 조망하였다.

1. 서론

최근 몇 년 사이 유무선 네트워크 기술의 급격한 발전 및 개인용 휴대 장치의 보급 증가와 그에 맞물린 소셜 네트워크 서비스의 활성화로 인해 데이터의 전송량이 폭발적으로 늘어나게 되었다. 이 추세는 앞으로도 계속되어 클라우드 서비스 및 사물인터넷(IoT)등이 등장하는 최신 IT 환경 하에 2014년 생성한 데이터 량의 10배를 2020년에 생성하게 될 것으로 전망된다[1]. 이러한 추세에 힘입어 대용량의 데이터를 처리하기 위한 서버 시스템, 즉 데이터센터를 확장 및 유지보수가 쉬우면서도 높은 성능을 가지는 시스템으로 구축 하고 있다.

데이터센터는 수많은 서버와 이들을 연결하는 네트워크 시스템으로 구성되어 있으며, 서버 시스템에서 오랫동안 성능 향상의 발목을 잡던 스토리지 분야는 SSD의 등장 및 RAMCloud 프로젝트와 같은 기술의 도입으로 많은 발전을 이루고 있다[2][3]. 이로 인해 그동안 HDD의 느린 접근속도에 의해 발전의 의미가 미미하였던 서버 간 네트워크에서의 지연시간이 데이터센터의 성능에서 중요한 비중을 차지하게 되면서 저지연 네트워크 구성을 위한 다양한 연구들이 활발히 진행되고 있다[4][5][6].

본 논문에서는 데이터센터 네트워크의 구성요소를 분석하고 각각의 지연시간을 알아보았고, 그중 확장 및 유지보수가 쉬우면서 상대적으로 지연시간이 높은 이더넷을 데이터센터에 적용하고자 하였을 때 이더넷의 지연시간을 개선하기 위한 방법으로 Intel DPDK를 적용하여 단점을 최소화 하였다[7].

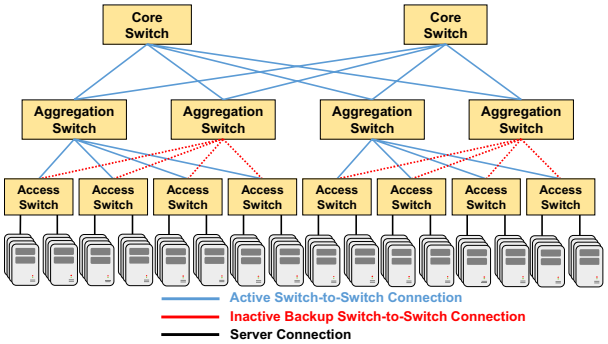
2. 관련연구

2-1. QPI (QuickPath Interconnect)

QPI는 프로세서 간 통신을 위해 기존에 사용하던 FSB(Front-Side Bus)를 대체하기 위해 2008년에 Intel에서 만든 인터페이스이다. 네할렘 아키텍처를 발표하면서 메모리 컨트롤러와 노스브릿지를 CPU에 통합하면서 확장성과 최대 속도에 단점을 가지는 버스 방식의 FSB를 포기하고 점대점 연결 방식의 QPI를 새롭게 적용하였다. 전송속도는 4.8GT/s 5.86GT/s, 6.4GT/s, 8.0GT/s 및 9.6GT/s이며, 지속적으로 Intel에 의해 버전 업데이트가 되고 있다. 현재 프로세서 간에만 주로 사용되지만 Intel의 QuickAssist Acceleration Technology 등을 통해 외부 칩을 연결하여 사용하는 과제가 임베디드 시장을 위주로 진행되고 있다[8]. 하지만, 데이터센터 내 다수의 노드를 연결할 수 있는 확장성을 지니고 있지는 않다.

2-2. PCI Express (PCIe)

PCI Express는 PCI-SIG에 의해 만들어진 고성능 입출력 인터페이스로, 주로 단일 컴퓨터 내에서 프로세서와 주변장치 간의 연결을 위한 용도로 사용되며 오늘날 대부분의 컴퓨터 시스템에 탑재되어 있다. 특히 기존의 병렬전송 방식 시스템 버스들의 한계를 극복하기 위하여 고속 직렬전송 방식의 점대점 연결 구조를 채택하였다. 2002년 Lane 당 2.5Gbps 전송 성능의 1.0 표준이 공개된 이래로 2006년 5Gbps의 2.0, 2010년 8Gbps의 3.0 표준이 공개되었으며, 2015년 16Gbps의 4.0 표준이 공개될 예정이다.



(그림 1) ANSI/TIA-942 Tier3 네트워크 구성

빠른 속도와 낮은 전력소모, 저지연성의 특징으로 인해 이를 시스템 간의 인터커넥트로 사용하고자 하는 다양한 시도가 이루어지고 있다[9][10].

2-3. 인피니밴드(Infiniband)

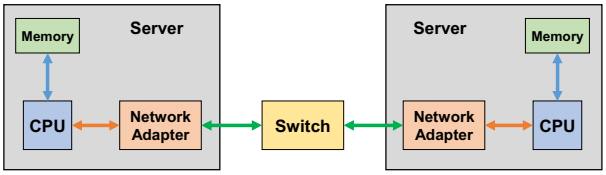
90년대 인터넷의 발전으로 인해 서버의 입출력과 서버 간 통신 성능이 중요해지며 차세대 인터커넥트의 필요성이 대두되었다. 1999년 IBTA(InfiniBand Trade Association)에서 발표한 인피니밴드는 스위치 기반의 점대점 연결망 구조의 고속 직렬연결 방식이며, Virtual Interface 아키텍처를 바탕으로 한 채널기반 메시지 패싱 인터페이스를 통해 메시지 전송 처리속도를 향상시키고 RDMA 전송을 제공하여 대용량 데이터의 전송을 지원하고 있다. Lane 당 2.5Gbps의 SDR을 시작으로 5Gbps의 DDR, 10Gbps의 QDR, 14Gbps의 FDR 그리고 25Gbps의 EDR 규격이 있다. 인피니밴드는 1us 수준의 낮은 지연시간, 4x 기준 100Gps의 고성능, 그리고 높은 확장성에 의해 많은 데이터센터와 고성능 컴퓨터의 인터커넥트로 채용하고 있다[11][12]. 하지만 인피니밴드는 현재 Intel과 Mellanox 단 두 회사에서만 IBA(InfiniBand Adapter)를 공급하고 있기 때문에 특정 벤더에 기술 종속적이라는 단점을 가지고 있다.

2-4. 이더넷(Ethernet)

이더넷은 비용 효율적이고 확장이 용이하며, 개방된 규격으로 인한 많은 기업의 참여 등으로 큰 시장을 확보하게 되면서 가장 널리 쓰이는 근거리 통신망 프로토콜이 되었다. Xerox, Intel, DEC社가 공동 개발하여 1980년에 제품화 시킨 방식으로 1983년 IEEE에 의해 IEEE 802.3으로 표준화 되었다. IEEE 802.3은 OSI 모델의 물리계층에서 신호와 배선을, 데이터 링크 계층에서 MAC(Media Access Layer)과 LLC(Link Layer Control) 부계층을 규정하고 있다.

이후 10Mbps, 100Mbps, 1000Mbps의 규격이 고속 전송의 필요성에 의해 도입되었다. 그리고 2002년 10Gbps의 전송속도를 가지는 10GbE가 발표되었고, 이후 2010년 10GbE를 묶어 사용하는 40GbE와 100GbE가 표준으로 정의되었다.

이러한 이더넷을 데이터 센터 네트워크로 사용하려는 연구들이 활발히 진행되어, 데이터 센터 네트워크를 위한



(그림 2) 서버간 네트워크 구성

다양한 표준들을 통해 이더넷이 갖는 QoS(Quality of Service) 관리의 어려움이나, 패킷 손실, 높은 지연시간 등의 문제를 해결하고 있다[13][14].

3. 데이터센터의 지연시간 분석

일반적으로 데이터센터는 높은 가용성을 위해 다중화되어있으며 이런 데이터센터 가용성에 대한 신뢰도를 데이터센터의 설계, 배선, 설치, 공조 등에 대한 표준화 지침인 ANSI/TIA-942에서 등급별로 제시하고 있다. 그 중 데이터센터 Tier3의 네트워크 구성(그림1)을 기준으로 하여 네트워크의 지연시간을 단계별로 살펴보도록 하겠다[15].

3.1 서버 내부의 지연시간

기본적으로 데이터센터는 여러 컴퓨팅 노드들로 구성되며 노드의 내부를 간략히 나타내면 (그림2)와 같다.
각 CPU는 메모리와 DDR3/4 와 같은 CPU-메모리 버스로 연결된다. 그리고 CPU와 CPU 사이는 QPI나 HyperTransport와 같은 인터커넥트로 연결되어 있다. CPU나 메모리의 종류에 따라 가변적이나 Intel의 실험결과에 따르면 CPU에서 메모리에 접근하는 시간은 70~75ns이고, QPI에서 발생하는 지연시간은 30~35ns로 볼 수 있다[16]. 각 컴퓨팅 노드는 NIC(Network Interface Card)를 거쳐 스위치로 연결된다. NIC는 CPU와 시스템 버스로 연결되어 있으며 현재 주로 PCI Express가 사용된다.

리눅스에서 PCI 버스에 연결된 장치 정보를 출력하는 lspci 명령을 수정하여 PCI Express로 연결된 장치들까지의 명령전달시간을 측정한 결과 장치에 따라 170~325ns가 소요되었다. 따라서 메모리부터 NIC에 도달할 때 까지 발생하는 지연시간은 270~450ns 수준이라고 할 수 있다.

3.2 네트워크의 지연시간

네트워크에서의 지연시간은 변조, 패킷화, SERDES (Serialize/Deserialize)등에 소요되는 인터페이스 지연시간, 라우터나 스위치에서 발생하는 스위칭 지연시간, 신호를 전송할 때 전송선로에서 소요되는 전달 지연시간 등으로 구분할 수 있다. 전달지연시간은 구리도선과 광섬유가 큰 차이 없이 미터 당 5ns 정도의 전달지연시간을 가진다.

PCI Express 패브릭(Fabric)은 PCI Express를 시스템간의 인터커넥트로 사용하기 위한 기술이며 NIC 대신, PCI Express 리피터나, 리타이머가 사용된다. 리피터는 1ns 이하, 리타이머는 70ns 이하의 지연시간을 갖는다[17]. 또한 PCI Express 패브릭 스위치는 130~150ns의 지연시간을 보인다[18].

구분	대역폭	종단간지연시간	확장성 (노드수)
QPI	384Gbps	30~35ns	~4
PCIe(3.0/x16)	256Gbps	170~325ns	~256
PCIe Fabric	256Gbps	720~1,500ns	~92,544
Infiniband(FDR/x4)	56Gbps	0.7us~	~65,536
Ethernet(10GbE)	10Gbps	2.03us~	~2^{48}

<표 1> 데이터센터 내 네트워크 기술별 지연시간

패킷 크기(Bytes)	지연시간(nsec)
64	3,568
128	3,640
256	3,748
512	4,041
1024	4,523

<표 2> DPDK 적용 시 지연시간

PCI Express 패브릭의 종단 간 지연시간을 계산해보면 서버 내부의 지연시간을 270ns라고 하고, 스위치의 지연시간을 130ns라고 하고, 서버와 스위치 사이가 각각 5m 케이블로 연결되어있다고 하면 종단 간 지연시간은 270 + 25 + 130 + 25 + 270 = 720ns가 된다.

인피니밴드의 경우 종단 간 지연시간은 1us 정도이며, 그 중 스위치가 차지하는 시간은 100ns정도이다[19].

이더넷의 종단 간 지연시간은 위의 두 기술에 비해 종단 간 지연시간 기준 10~50us 정도로 높은 편이며, 이 때문에 지연시간을 개선하기 위해 네트워크를 통해 다른 컴퓨터의 메모리에 직접 접근하는 RDMA(Remote Direct Memory Access)나 CPU의 TCP/IP 패킷처리에서 발생하는 부하를 NIC의 하드웨어에서 처리하는 TOE(TCP/IP Offload Engine) 등의 기술을 사용하여 저지연시간을 달성하고자 하고 있다[20].

수십 마이크로초를 소요하던 이더넷 스위치 또한 지연시간이 중요한 성능지표가 됨에 따라 Cut-Through 방식을 지원하는 저지연시간 특화된 제품들이 출시되고 있으며 최저 190ns인 제품도 시장에서 판매되고 있다[21].

4. Intel DPDK를 적용한 지연시간 개선 실험

Intel DPDK(Data Plane Development Kit)은 Intel 아키텍처 기반의 패킷 처리 속도 최적화를 위한 라이브러리와 네트워크 인터페이스 컨트롤러 드라이버의 집합으로, 고속 패킷 처리를 위한 어플리케이션 개발을 빠르게 할 수 있는 프레임워크를 제공한다. 이러한 고속 패킷 처리 성능을 통해 다양한 네트워크 요소, 보안기기, 모바일 및 브로드밴드 장비 등을 위한 고성능 솔루션 개발이 이루어지고 있으며, 대표적인 예로 Open vSwitch가 있다[22].

DPDK 프레임워크는 EAL(Environment Abstraction Layer) 생성을 통해 특정 하드웨어, 소프트웨어에 맞는 라이브러리 세트를 구성한다. EAL은 환경 구성요소들에 대한 표준화된 인터페이스를 제공하며, EAL이 생성되면 개발자는 라이브러리를 어플리케이션에 연결하여 사용할 수 있다.

하드웨어	
CPU	Intel Xeon E5-2670 v2 2.5GHz
칩셋	Intel C600
소켓수	2
CPU당 코어갯수	10
Last Level 캐시	25MB
QPI	8.0GT/s
주메모리	128GB(DDR3-1600 4ch)
NIC	Intel Ethernet Converged Network Adapter X710-DA2

소프트웨어	
운영체제	CentOS 6.5 64bit
커널 버전	2.6.32-431.el6
NIC 드라이버 버전	1.2.37
DPDK 버전	1.7.1
User priority	Realtime priority

<표 3> 지연시간 측정 실험환경

실험을 위한 하드웨어 환경은 한 대의 서버에서 이더넷 어댑터의 듀얼포트를 서로 연결하여 0번 포트에서 TX를 수행하고, 1번 포트로 RX를 수행하도록 구성하였다. 소프트웨어 환경은 DPDK의 Burst TX 기능을 사용하여 시스템 메모리의 mbuf 영역에서 패킷을 순차적으로 전송하고, Burst RX 기능을 반복 사용하여 패킷을 수신한 후 mbuf 영역에 쓰기를 수행한 다음 하드웨어 카운터(x86 Time Stamp Counter)를 사용하여 클록 단위로 지연시간을 측정하는 함수를 작성하였다. 또한 실험 어플리케이션이 0번 포트를 이용하여 패킷을 전송한 후 1번 포트로 패킷이 수신 완료될 때까지 수신을 반복 수행하도록 하고, 수신이 완료된 이후 다시 다음 패킷을 0번 포트를 이용하여 전송하도록 수정하여 어플리케이션을 작성하였다.

구축된 환경에서 DPDK 미적용 지연시간 측정결과는 64 Byte 패킷 기준으로 약 25us가 소요되었다. DPDK 적용 시 측정된 결과는 <표 2>와 같다.

64 Bytes 기준 DPDK 미적용 시에 비해 본 실험에서는 3,568ns로 약 86% 감소된 결과를 보였다. 이러한 지연시간의 차이가 발생하는 이유는 DPDK 미적용 시 어플리케이션의 사용자모드의 함수가 복잡한 커널스택을 모두 통과하는 반면, DPDK를 적용한 경우는 EAL을 통해 리눅스 커널의 TCP 처리부를 우회하고 NIC에 접근하여 패킷을 처리하므로 지연시간을 줄인다. 또한 대부분의 패킷 입출력 라이브러리들이 CPU에 연결된 원격 및 지역 메모리를 구분하지 않고 할당하기 때문에 메모리 접근시간이 늘어날 수 있는 반면 DPDK의 경우 지역 메모리만을 할당할 수 있다.

<표 4>는 본 실험에서 분석한 DPDK 적용시의 단계별 지연시간 분석 결과이다. NIC 내부동작은 DPDK 적용 여부에 따라 큰 차이가 없을 것으로 추정되므로 지연시간은 소프트웨어 부분에서 크게 줄어든 것으로 분석할 수 있다. 하지만 이는 TOE를 적용한 Mellanox사의 테스트 결과(2.08us) 및 NetFPGA로 커스터마이징한 NIC의 테스트 결과(1us)에 비해서 높은 지연시간을 보였다[23][24].

구분	항목	내용	시간	
S/W	rte_eth_tx_burst	Burst TX	115ns	150ns~
	i40e_rxtx	Device Driver		
H/W	DMA 읽기동작	메모리접근	170ns	3us~
	NIC Tx MAC	NIC 내부동작		
	NIC Tx PHY	Serialize/CRC	100ns	
	3m Cable	전달지연시간	15ns	
	NIC Rx PHY	Deserialize/CRC	100ns	
	NIC Rx MAC	NIC 내부동작		
	DMA 쓰기동작	메모리접근	170ns	
S/W	i40e_rxtx	Device Driver		310ns~
	rte_eth_rx_burst	Burst RX	216ns	

<표 4> DPDK 적용시 단계별 지연시간 분석(64B)

5. 결론

본 논문에서는 데이터센터 내부에 적합한 네트워크를 연구하기 위해 현존하는 네트워크 인터페이스인 QPI, PCI Express, PCI Express 패브릭, 인피니밴드, 10기가비트 이더넷의 지연시간을 정리 또는 측정하였다. 그 중 저비용 고효율 인터페이스인 이더넷의 단점인 지연시간을 줄이기 위해 Intel의 DPDK를 적용한 경우 지연시간이 약 86% 감소함을 확인할 수 있었다. 이는 기존의 소켓 프로그래밍을 사용한 서버 간 네트워크 구현 방식에 비해서는 대폭 향상되었지만, 하드웨어 오프로딩과 특화된 디바이스 드라이버를 사용하여 저지연에 특화된 설계를 구현한 Mellanox의 이더넷 인터페이스나 NetFPGA에 비해서는 여전히 상대적으로 낮은 성능을 가지는 한계를 보였다. 본 실험보다 더 높은 성능을 얻기 위해서는 이더넷의 특정 스펙만 사용하거나, 더 넓은 영역에서 하드웨어 가속을 수행하고, 드라이버 계층에서 커널 메모리 접근을 최소화하는 방법 등에 대한 추가 연구가 필요할 것으로 보인다.

6. 사사

본 연구는 산업통상자원부와 KSRC(Korea Semiconductor Research Consortium)가 공동으로 지원하는 미래반도체소자 원천기술개발사업(과제번호 10044735)의 연구결과로 수행되었음. 또한 2014년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No. NRF-2014R1A2A1A11052936).

참고문헌

[1] “EMC 디지털 유니버스보고서: 디지털 유니버스의 기회,” 한국EMC, 2014

[2] Samsung SSD 845DC Pro brochure, 삼성전자, 2014

[3] John Ousterhout et al., “The Case for RAMCloud,” Communications of the ACM, Vol. 54, Jul. 2011

[4] Mohammad Alizadeh, “Less is More: Trading a little Bandwidth for Ultra-Low Latency in the Data Center,” NSDI, 2012

[5] Jung Ho Ahn et al., “HyperX: Topology, Routing, and Packaging of Efficient Large-Scale Networks,” SC, 2009

[6] Ting Wang et al., “NovaCube: A Low Latency Torus-Based Network,” GLOBECOM, 2014

[7] “Intel Data Plane Development Kit Overview Packet Processing on Intel Architecture,” Intel, 2012

[8] Intel QuickAssist Acceleration Technology for Embedded Systems, Intel, 2015.03.13 from http://www.intel.com/content/www/us/en/io/quickassist-technology/quickassist-technology-developer.html

[9] Derek Percival, “Implementing System using PCI Express as a Fabric,” PCI-SIG, 2012

[10] Timothy P. Morgan, “PCI Express Switching Takes On Ethernet, InfiniBand,” 2015.03.13. from http://www.enterprisetech.com/2014/03/13/pci-express-switching-takes-ethernet-infiniband/

[11] “InfiniBand Architecture Specification,” IBTA, 2015

[12] 박경, “InfiniBand의 개요,” 주간정책동향 967호, p.13~22, 정보통신산업진흥원, 2000

[13] Behrouz A. Forouzan, “Data Communications and Networking 5th Edition,” McGraw-Hill, 2012

[14] “IEEE Standard for Ethernet,” IEEE, 2012

[15] “Telecommunications Infrastructure Standard for Data Centers (TIA-942),” TIA, 2005

[16] “Memory Latencies on Intel® Xeon® Processor E5-4600 and E7-4800 product families,” 2015.03.13. from http://intel.ly/15KlFCT

[17] Dave Brown, “PCIe 3.0: 8.0GT/s Digital Retimer,” PCI-SIG, 2012

[18] “Product Snapshot,” PLX Technology, 2014

[19] “Financial & Trading solutions,” Mellanox, 2015.03.13. from http://www.mellanox.com/page/financial_trading

[20] W. Feng et al., “Performance Characterization of a 10-Gigabit Ethernet TOE,” HOTI, 2005

[21] “Nexus 3548 switch performance validation,” Cisco, 2012

[22] http://www.openvswitch.org

[23] “HP Mellanox Low Latency Benchmark Report 2012,” Mellanox technologies, 2012

[24] John W. Lockwood et al., “A Low Latency Library in FPGA Hardware for High Frequency Trading,” HOTI, 2012