ISBN :

KRONET 기반 Science DMZ 적용을 위한 분석

일자: 2016년 10월 18일
부서: 첨단연구망센터 첨단연구망응용지원실
제출자: 문정훈,문정훈
jhmoon@kisti.re.kr, wthong@kisti.re.kr

한국과학기술정보연구원
305-806 대전광역시 유성구 어은동 52번지
TEL (042)869-0676 / FAX (042)869-0679
www.kisti.re.kr

Contents

KREOENT 기반 Science DMZ 적용을 위한 분석

1. Science DMZ ····· 8
   1.1. 데이터 집약형 과학 데이터 전송 최적화를 위한 네트워크 모델 설계 ····· 8
   1.2. 동기(Motivation) ····· 9
   1.3. 구조(Architecture) ····· 11
   1.4. 첨단 서비스(Advanced Service) ····· 15
   1.5. 가상 회선 전략(Virtual Circuit Strategies) ····· 16
   1.6. 소프트웨어 정의 네트워킹(Software Defined Network) ····· 18
   1.7. 클라우드 리소스(Cloud Resource) ····· 22
   1.8. 데이터 전송 노드(Data Transfer Node) ····· 23
   1.9. 성능 모니터링(Performance Monitoring) ····· 27
   1.10. 보안 (Security) ····· 28
2. 호스트 튜닝(Host Tuning) ····· 32
   2.1. 배경(background Information) ····· 33
   2.2. 리눅스(Linux) ····· 35
   2.3. NIC 튜닝(NIC Tuning) ····· 38
   2.4. 패킷 페이싱(Packet Pacing) ····· 40
   2.5. 40G/100G 튜닝(40G/100G Tuning) ····· 41
3. Network Tuning ····· 42
   3.1. TCP 성능 향상을 위한 튜닝 ····· 42
   3.2. UDP 튜닝(UDP Tuning) ····· 46
4. Data Transfer Tool ····· 49
   4.1. 배경 ····· 49
   4.2. Globus ····· 51
   4.3. GridFTP ····· 52
   4.4. scp와 sftp ····· 54
5. Network Test Tools ····· 55
   5.1. 네트워크 측정 도구 ····· 55
   5.2. 네트워크 테스트 도구 ····· 56
6. KREONET 기반 SaaS 응용에 대한 자동화된 Cloud 실행 환경 구축 ····· 56
7. KISTI Cloud 설치 및 검증 ····· 57
   7.1. 설치 환경 및 구성 요소 ····· 57
   7.2. 설치 검증 ····· 58
8. 자동화된 OverCloud 환경의 KREONET 적용을 위한 DTN 구축 ····· 61

표 목차

표 1 DTN-FIONA Hardware Specification ····· 26
표 2 PerfSONAR Hardaware Specification ····· 27
표 3 네트워크 성능 측정 도구 ····· 56
표 4 네트워크 테스트 도구 ····· 56
표 5 KREONET DTN Specification ····· 62

그림 목차

그림 1 Science DMZ 의 단순구조 ····· 12
그림 2 슈퍼컴퓨터 센터 네트워크 ····· 13
그림 3 빅데이터 사이트 네트워크 ····· 14
그림 4 SDN 일반적인 사용의 경우 ····· 18
그림 5 SDN분리의 경우 ····· 19
그림 6 Science DMZ 기본 구조 ····· 19
그림 7 SDN 적용을 위한 Science DMZ 구조 ····· 20
그림 8 SDN 적용을 통한 Science DMZ 망분리 ····· 21
그림 9 TCP 버퍼 사이즈 조정에 따른 성능 영향 ····· 35
그림 10 Mathis Equation ····· 44
그림 11 성능과 대기시간의 관계 ····· 44
그림 12 성능과 대기시간과의 관계 - 정밀 버전 ····· 45
그림 13 nuttcp 결과 ····· 48
그림 14 CPU 코어 확인1 ····· 48
그림 15 CPU 코어 확인2 ····· 48
그림 16 KISTI Cloud 테스트베드 구성도 ····· 58
그림 17 Nova 컴퓨트 서비스 리스트 ····· 59
그림 18 Neutron 에이전트 리스트 ····· 59
그림 19 관리자 GUI에서의 하이퍼바이저 요약 ····· 60
그림 20 일반 사용자 GUI에서의 Tenant 네트워크 토폴로지 ····· 61
그림 21 KREONET DTN Architecture ····· 63
그림 22 KREONET환경에서의 Science DMZ ····· 63

KRONET 기반 Science DMZ 적용을 위한 분석

1. Science DMZ

1.1. 데이터 집약형 과학 데이터 전송 최적화를 위한 네트워크 모델 설계

o Science DM[1]는 실험실이나 캠퍼스 크기의 근거리 네트워크를 위해 만들어진 네트워크의 일부로 범용의 비즈니스 시스템이나 기업 시스템을 위해서라기보다는 고성능의 과학적 응용분야의 실험장비, 설정, 보안 정책 전송 성능 등을 최적화시킨다.

o ESnet 엔지니어에 의해 제안되고 발전되어온 Science DMZ모델은 대용량 벌크 데이터 전송, 실험 원격 제어 및 데이터 시각화를 포함한 고성능 과학 어플리케이션의 요구에 맞게 설계된 환경을 구성함으로써, 연구 기관에서 발생할 수 있는 일반적인 네트워크 성능 문제를 해결한다.

o Science DMZ는 확장가능하고(scalable), 점진적 전개(incrementally deployable)가 가능하며 100 Gigabit 이더넷 서비스, 가상회선(virtual circuit), 소프트웨어 정의 네트워킹 기능(software-defined networking capability)과 같은 첨단 신기술과 쉽게 접목 가능하다.

o Science DMZ는 다음의 네 가지 핵심 개념을 하나로 통합한다.

- 과학 네트워크와 범용 네트워크가 구분되고 고성능 어플리케이션을 위해 명확히 설계된 네트워크 아키텍처
- 데이터 전송을 위한 전용 시스템의 사용
- 네트워크 성능 보장과 모니터링 및 문제 해결을 위한 성능 측정 및 네트워크 테스트 시스템
- 고성능 과학 환경에 맞춘 보안 정책과 강화 메커니즘

로컬 사이트에 Science DMZ 통합

o 운영 모범 사례로 Science DMZ의 구성 요소는 많은 연구 기관에서 과학 어플리케이션의 지원을 위한 확장 가능한 모델을 형성한다. Science DMZ모델은 NERSC와 같은 슈퍼컴퓨터센터에서 강입자 충돌기 협업(the Large Hadron Collider Collaboration)에 관련된 시설까지 다양한 과학 환경에서 성공적으로 적용되었다. 또한, Science DMZ모델은 많은 연구소와 대학에서 채택되었고, 특히 Internet2와 협력을 통해 Science DMZ는 혁신플랫폼아키텍처(innovation Platform architecture) 에도 중요한 역할을 한다. 혁신플랫폼아키텍처는 개선된 네트워크 기능의 장점을 활용하도록 돕는 것을 목표로 만들어진 아키텍처로써

대학의 구성원들이 캠퍼스에서 빠른 연구개발이 가능하도록 돕는다.

o Science DMZ 모델은 데이터 집약적인 과학 분야에서 다음의 몇 가지 주요 문제를 해결한다.

- TCP 성능 저하를 유발하는 패킷 손실을 줄이거나 제거
- 고성능 응용 프로그램이 불필요한 제약에 의해 방해되지 않도록 적절한 보안 아키텍처와 제어를 구현
- 100 Gigabit 이더넷 서비스, 가상회선, 소프트웨어 정의 네트워킹 기능과 같은 넓은 분야의 과학 서비스를 접근할 수 있도록 지역 과학 자원(local science resources)에 대한 접점(on-ramp)을 제공
- perfSONAR의 배치를 통해 네트워크 테스트, 네트워크 측정 및 성능 분석을 통합

1.2. 동기(Motivation)

o Science DMZ의 등장배경은 실험실이나 캠퍼스 네트워크는 일반적으로 여러 조직적 임무를 지원한다. 첫째, 이메일, 조달 시스템 및 웹 브라우징과 같은 조직의 정상적인 업무와 관련된 네트워크 트래픽을 위한 인프라를 제공한다. 네트워크는 또한 조직의 재무 및 인사 데이터를 보호하는 보안 기능을 구축해야 한다. 이러한 네트워크는 동시에 과학 연구 과정의 인프라로 사용되며 연구자들이 다양한 외부 소스로부터 조사된 데이터를 분석, 공유 및 저장하기 위해 이 인프라를 사용한다.

o 그러나 대부분의 경우, 비즈니스 운영에 최적화된 네트워크는 데이터 집약적 과학의 데이터 전송 요구를 지원할 수 있도록 설계되지도 않았고, 그런 수용능력도 없다. 과학자들은 소위"범용" 네트워크를 통해 대량 데이터 처리가 필요한 응용 프로그램을 수행하려 한다면 그 결과는 흔히 성능을 저하시키는 결론을 낳는다–대부분의 경우 과학적 미션에 대단히 안 좋은 영향 끼친다. 범용 네트워크의 많은 부분이 과학 애플리케이션의 성능을 개선하는 데 필요한 방식으로 변경하기 어렵거나 불가능하기 때문에, 네트워크는 범용 네트워크의 동작에 영향을 주지 않고 과학 애플리케이션을 지원하는 것을 허용하도록 구성되어야 한다.

o Science DMZ 모델은 범용적 사용을 위한 지원을 포함하지 않고 과학 애플리케이션을 위해 특별히 변형된 네트워크의 일부를 구성하여 이를 달성한다. 범용 네트워크와 고성능 과학 네트워크(Science DMZ)를 분리함으로써, 각각은 서로 다른 간섭 없이 최적화할 수 있다.

o Science DMZ의 핵심 목표는 고성능 과학 애플리케이션을 지원하는 것이지만, 이

는 Science DMZ 단독으로 해결할 수 없다. 과학적 협업은 다른 네트워크 기반 시도처럼 본질적으로 종단간(end-to-end)이다. Science DMZ는 쉽게 가상 회선(Virtual Circuit) 및 소프트웨어 정의 네트워킹(Software-defined networking), 100 기가비트 이더넷(100 gigabit Ethernet)과 같은 광역 과학 지원 서비스를 통합할 수 있다. 범용 네트워크는 이와 같은 기술 및 서비스를 효과적으로 사용하기 힘들겠지만, Science DMZ는 범용네트워크 인프라와 간섭 없이 지역적 과학자원과 과학업무 수행에 필요한 네트워크 서비스가 연결되도록 해준다.

Science DMZ의 개발 역사

o Science DMZ 구조는 네트워킹의 설계, 운영 및 보안과 같은 여러 측면과 연관이 있다. Science DMZ 라는 용어는 네트워크 보안 아키텍처의 공통 요소인"DMZ 네트워크"에서 비롯되었다. 기존의DMZ는 특수목적 네트워크의 일부이다. 이것은 네트워크 경계나 그 주변에서 외부 웹, 이메일 수신, 및 신뢰할 수 있는DNS 서버와 같은 외부와 연결되는 사이트 서비스를 처리하기 위해 설계되었다. 보안 정책, 네트워크 장치 구성과 같은 것들이 DMZ에 맞게 조정되고, 내부의 근거리 통신망(LAN) 인프라의 보안 정책이나 구성과는 융합되지 않는다.

o Science DMZ는 대량의 데이터 이동과 데이터 집약적 실험 패러다임을 포함한 고성능 과학 애플리케이션을 지원하는 작업에 이 개념을 적용한다. ScienceDMZ는 한 사이트나 캠퍼스의 전용 네트워크인데, 최대한 네트워크 경계에 가까이 위치하고, 고성능 과학에 최적 지원을 제공하도록 설계 및 구성된다. Science DMZ가 네트워크 문제를 해결하고 그 네트워크를 특징짓는 능력을 포함하고 있어 그 결과로 성능문제를 빠르게 해결할 수 있다. 이것은 일반적으로 Science DMZ에perfSONAR 호스트를 배치함으로써 가능하다. Science DMZ는 넓은 지역에서, 그리고 협업하는 실험실이나 대학 내에 있는 다른 Science DMZ들에서 perfSONAR 호스트들을 테스트 할 수 있다.

Science DMZ는TCP 성능 문제를 해결

o TCP는 네트워크 프로토콜 분야에서"취약한 장비"로 특징지어졌다. 비록 신뢰할 만한 데이터 이동을 요구하는 대부분의 과학 응용분야들이 데이터 이동을 위해 TCP 기반 도구를 사용하고 있지만, 패킷 손실로 대변되는 TCP는 성능저하 문제를 발생시킬 수 있다. TCP는 패킷의 손실을 네트워크 혼잡 탓으로 해석하기 때문에 손실이 발생할 때마다 TCP는 데이터 송신 속도를 급격하게 감소시킨다. 전송률은 다시 천천히 상승하지만, 만약 더 많은 손실을 마주하게 된다면 전송 속도를 더욱 줄이게 된다. 이러한 현상은 통신 호스트 사이의 거리가 멀수록 급격한 변화를 보게 된다. 실제로 1% 미만의 조그만 손실일지라도 100배에 가까운

TCP성능 저하를 야기하기에 충분하다.

o 일반적으로 네트워크가 손실에 강하도록 하기 위해서 TCP를 수정하는 것 보다 TCP를 수용하기 위한 네트워크를 설계하는 것이 훨씬 쉽다. 이것은 고성능 TCP 기반 과학 어플리케이션을 지원하는 네트워크 인프라가 일반적인 경우에 반드시 TCP에 손실 없는IP 서비스를 제공해야 한다는 것을 의미한다. Science DMZ 모델은 실험실이나 캠퍼스, 혹은 과학시설에 고성능 애플리케이션이 성공적일 수 있도록 필요한 서비스를 제공할 수 있는 특수 용도의 인프라를 구축하도록 해준다.

1.3. 구조(Architecture)

Science DMZ 구조

o 고성능 과학 응용 프로그램을 효율적으로 배포하고 지원하기 위해 필요한 능력은 높은 대역폭, 고급 기능(advanced features), 그리고 성능을 보장하는 좋은 장비 등이다. 운영에 필요한 조건으로 단순성, 책임성, 정확성, 그리고, 실험과 계측 서비스의 수월한 통합에 대한 필요성을 요구한다. 보안 요구 사항은 정확성을 보장하고, 오용을 방지하고, 사이트 또는 과학적 활동에 대한 손상이나 난처함을 피하기 위해 필요하다.

o Science DMZ 아키텍처는 단순하고, 확장 가능한 네트워크 부분을 만듦으로써 이러한 필요성을 충족한다. 이러한 네트워크는 범용컴퓨팅과 수반되는 추가적 복잡성을 명시적으로 배제하는 동시에 고성능 과학 어플리케이션을 명시적으로 수용한다.

o 이상적으로, Science DMZ는 WAN구간 라우터, 즉 백본에 직접 연결되는데, 이는 고성능의 데이터 전송과 다른 과학적 어플리케이션들을 지원하기 위해 구성되어야만 하는 장치의 수를 최소화하기 위함이다. 높은 성능을 달성한다는 것이 기본으로 설정된 네트워크 장비나 시스템을 가지고는 매우 어렵다. 그리고, 사이트 주변의 Science DMZ의 위치는 시스템 및 네트워크 튜닝 프로세스를 단순화한다. 또한, 성능 문제가 있는 경우에도 더 큰 범용LAN 인프라보다도 차라리 몇몇 기기의 문제를 해결하기가 훨씬 용이하다.

간단한 Science DMZ 다이어그램

o 간단한 Science DMZ는 몇 가지 필수 구성 요소가 있다. 여기에는 고성능 광역 네트워크 및 고급 서비스 인프라, 고성능 네트워크 장비, 데이터 전송 노드(Data Transfer Nodes, DTN)와 같은 과학 전용 자원에 대한 전용 액세스를 포함한다.

데이터 경로를 따라 이러한 구성 요소를 보여주는 간단한 Science DMZ의 개념적 다이어그램은 다음과 같다:

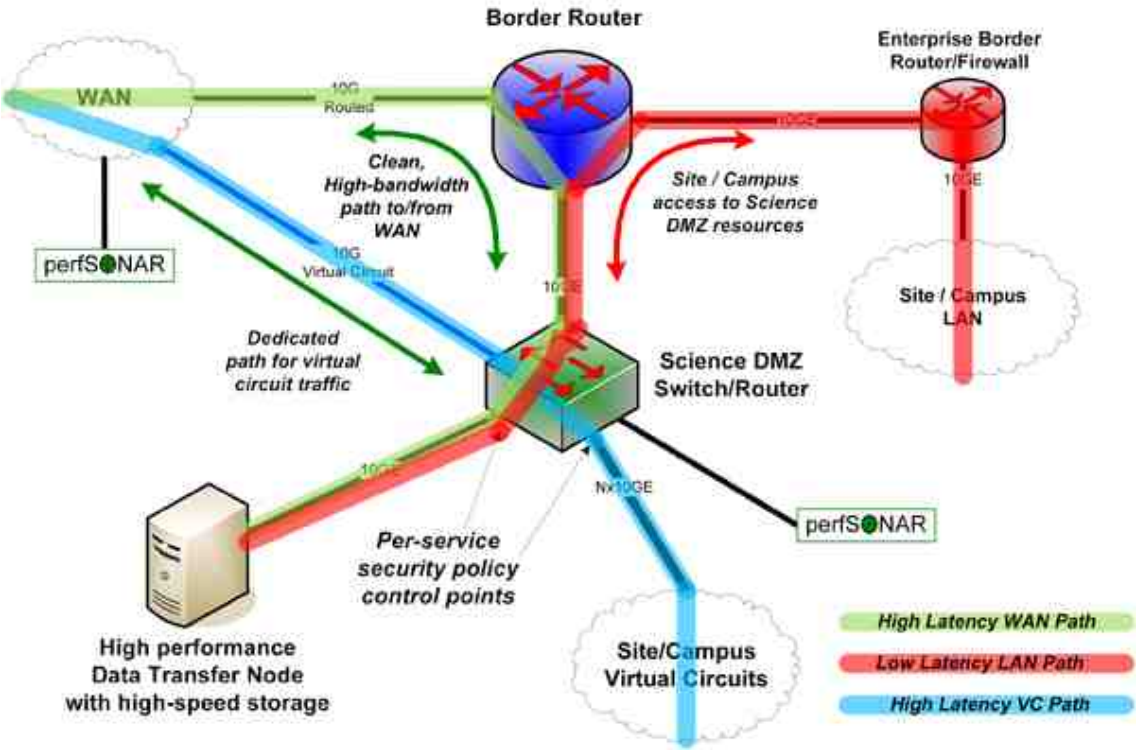

그림 1 Science DMZ 의 단순구조

o Science DMZ의 필수 구성요소와 단순 구조가 위의 그림에 나타나 있다. 데이터 전송 노드(DTN)는 경계 라우터에 직접 연결된 고성능 Science DMZ 스위치 또는 라우터에 직접 연결된다. DTN의 일은 원격 사이트 및 시설로, 혹은 원격사이트나 시설에서 과학데이터를 효율적이고 효과적으로 이동하는 것이다. Science DMZ에 있는 모든 것이 이 목표의 대상이 된다. DTN에 대한 보안 정책 강화는 별도의 방화벽을 통해서가 아니라, Science DMZ 스위치나 라우터에 대한 액세스 제어 목록을 사용하여 수행된다.

o 시퀀서, 현미경, 망원경, 과학 처리과정을 통합하기 위해 설계된 다른 목적의 하드웨어와 같은 과학적 장비는 점점 네트워킹 과정의 일부가 된다. 이들 중 많은 장치는 로컬 스토리지 장치로 수집 된 데이터의 이동을 용이하게 하는 직접화된 네트워크 연결을 갖는다. 이러한 장치들은 공공 네트워크에 노출되는 것에 대해 준비되지 않았으며, 다른 비즈니스 장치들(예를 들어 프린터, 소프트폰, HVAC 시스템)이 그런 것처럼 반드시 보호되어야 한다: ACL처럼 보안의 범위를 넘어선다. 만일 방화벽이 정책에 의해 필요한 경우라면 장치와 외장 스토리지 사이의 로컬 액세스는 극복해야 할 짧은 지연시간을 가지게 될 것이다. 그래서 영향을 받지 말아야 할 TCP와 같은 프로토콜에 영향을 준다. (위 그림1의 빨간색 경로)

슈퍼컴퓨터 센터 네트워크

o 아래 그림은 단순화된 슈퍼컴퓨터 센터 네트워크를 보여준다. 아래 그림이 위에서 본 그림의 훨씬 간단한 Science DMZ와 유사해 보이지 않지만, 동일한 원리로 설계된다.

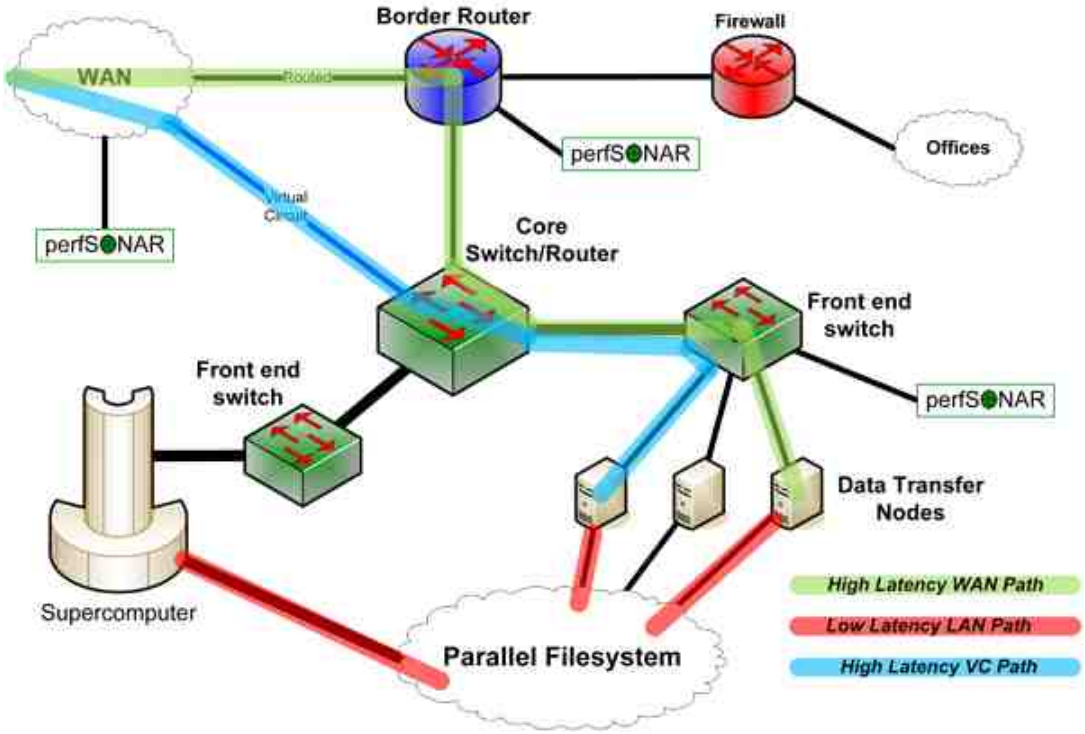

그림 2 슈퍼컴퓨터 센터 네트워크

o 슈퍼컴퓨터 센터 네트워크의 경우, 네트워크 인프라의 대부분은 Science DMZ라 할 수 있다. 즉,패킷 손실 없이 고속 데이터 플로우를 처리하도록 구축되고, 쉬운 문제해결 및 결함 분리를 허용하도록 설계된다. 처음부터 테스트와 측정이 인프라에 집적되었고, 그래서 로컬 인프라에 문제가 있는 것과 별개로 문제가 생긴 위치를 찾아내고 문제가 생기면 바로 처리가 가능하다. 주목할 사항은 광역데이터 전송에 의한 병렬 파일 시스템에 대한 액세스가 광역 데이터 전송 태스크만을 위한 데이터 전송 노드를 통해서 이루어진다. 데이터 집합이 DTN으로 전송되고 병렬 파일 시스템에 기록 되는 경우, 이중 복사 없이도 데이터는 슈퍼컴퓨터 리소스가 바로 사용 가능하게 된다. 그러나, DTN의 모든 장점(전용 호스트, 적절한 도구 및 올바른 구성)은 유지된다.

빅데이터 사이트

o 매우 큰 데이터 볼륨(예: LHC와 같은 큰 실험 등)를 처리하는 사이트의 경우, 개

별 데이터 전송 노드는 충분하지 않다. 이러한 사이트들은 데이터 전송 클러스터를 구성하고, 한 그룹의 기계가 수 페타바이트(multi-petabyte)의 데이터 저장소로부터 데이터를 서비스한다. 그러나, 여전히 Science DMZ의 원리가 적용된다 즉, 전용 시스템이 데이터 전송에 사용되며, 광역네트워크에 대한 경로에 문제가 없고 단순하며 문제해결이 쉽다. 결함격리(fault isolation)를 위해 테스트 및 측정이 여러 위치에 통합되어 있다. 이 네트워크는 광역 환경의 데이터 경로가 전체 네트워크 전단(front-end)을 포함한다는 점에서 슈퍼컴퓨터 센터의 예와 유사하다.

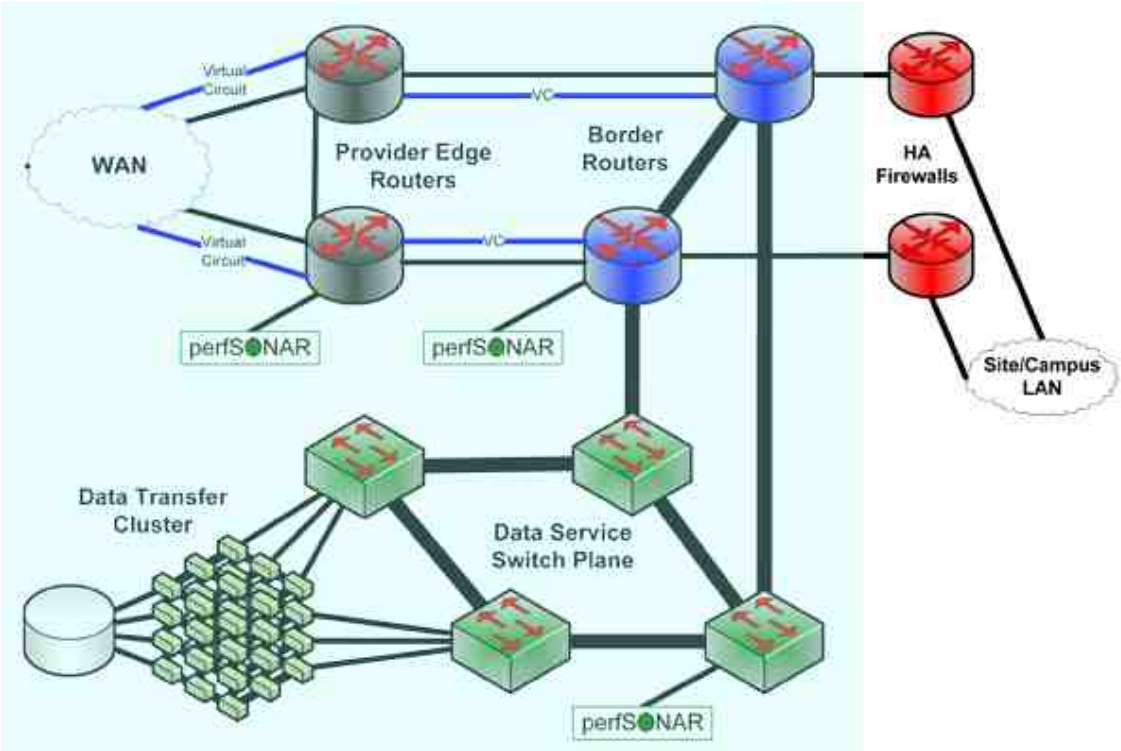

그림 3 빅데이터 사이트 네트워크

이러한 네트워크는 기본적인 연구 네트워크 환경의 백본에 여분의 액세스 포인트를 갖는다. 각각은 IP 라우팅(routed IP) 과 가상 회선 서비스(virtual circuit service)를 할 수 있다

o 네트워크의 비즈니스 부분은 과학데이터를 지원하기 위해 배치된 고용량 추가 인프라의 장점을 이용한다. 그리고, 서비스 시간을 보장하기 위해 여분의 방화벽도 배치한다. 그러나 과학데이터 플로우는 이런 장비들을 거치지 않는다. 데이터 서비스를 위한 적절한 보안 제어가 라우팅과 스위칭 면에 구현된다. 이것은 성능 문제를 일으키는 방화벽을 유지하기 위해 수행되고, 초당 기가비트 또는100 기가비트의 매우 높은 데이터 전송율이 방화벽 하드웨어의 능력범위를 넘어서기 때문에라도 수행되어야 한다.

o 이러한 예는 Science DMZ 원리의 유연성과 다용성을 보여준다. 일반적으로 고성능의 과학 어플리케이션은 범용 네트워크에서는 필요한 성능을 달성 할 수 없지만 Science DMZ에서 달성할 수 있다.

1.4. 첨단 서비스(Advanced Service)

첨단 및 유망 서비스를 위한 Science DMZ 지원

o Science DMZ는 가상 회로 및 소프트웨어 정의 네트워킹과 같은 넓은 영역에서 사용할 수 있는 첨단 네트워킹 서비스를 위해 연구 기관에 접근하는 이상적인 진입점(entry point)을 제공한다. 만일 Science DMZ가 역량이 있는 적정한 장비로 구성되고, 범용의 비즈니스 연결 수요를 지원해야 한다는 제약이 없는 경우 Science DMZ에 있는 지역 리소스가 첨단 광대역 네트워크 서비스를 이용하도록 만드는 것은 매우 간단하다.

가상 회로(Virtual Circuits)

o ESnet에서 개발 한 오스카스 플랫폼(OSCARS platform)과 같은 가상 회로 서비스는 Science DMZ 스위치에 직접 연결할 수 있고, 필요에 따라 개별 스위치를 연결함으로써 Science DMZ 스위치에 연결할 수도 있다. 캠퍼스나 실험실의 도메인 간 컨트롤러(IDC, inter-domain controller)는 로컬 스위치를 제공할 수 있고, DTN이나 다른 Science DMZ 자원이 원거리 기관의 과학서비스를 액세스할 수 있도록 다중 도메인 광역 가상 회선 연결(multi-domain wide area virtual circuit connectivity)을 시작할 수 있다. 이 구성의 예는 NSF가 투자한 Internet2 DYNES 프로젝트였다.이 프로젝트는 미국 전역60개 이상의 대학 캠퍼스에 이 아키텍처의 배포를 지원하고 있다.

100 기가비트 이더넷

o 100 기가비트 이더넷(GE) 기술은 미국에서 과학 네트워크로 구축되고 있으며, 국제적으로 데이터 집약적인 과학을 지원하기 위해 구축되고 있다. 100GE는 차세대 장비와 시설을 지원할 수 있다고 자부하고 전례없는 규모의 분산 데이터 세트의 과학적인 분석을 수행할 능력이 있다고 자부하지만, 100GE 기술은 연구 기관에서 범용 네트워크에 대한 중요한 문제에 직면한다. 예를 들어, 일반적으로 비즈니스 환경에 적용된 방화벽은100GE 과학 서비스를 효과적으로 지원할 수 없다. Science DMZ 모델은 연구 기관의 과학 임무에100GE 서비스를 통합하기 위한 확장 가능한 플랫폼을 제공한다. 100GE 서비스는 과학 네트워크가 제공하는

고급 서비스와 Science DMZ에서 이용 가능한 과학 자원들 사이의"빠른 경로"를 제공하기 위해 Science DMZ에 직접 연결할 수 있다.

클라우드 리소스

o 상용 혹은 R&E 서비스 공급자들의 의해 제공되는 클라우드의 사용은 제한된 시간에 자원을 처리해야 할 필요가 있는 연구자들에게는 매우 매력적이다. 클라우드 솔루션과 독자의 DMZ를 통합한다는 것은 클라우드로의 네트워크 경로를 안다는 전제와 데이터 이동성에 대한 기대성능을 만족시켜야 가능하다.

1.5. 가상 회선 전략(Virtual Circuit Strategies)

o OSCARS나 Internet2 Advanced Layer 2 Services (AL2S)와 같은 가상회로(Virtual Circuit, VC)는 R&E 네트워킹 구축의 일반적인 부분이 되어 가고 있다. 이런 기능들을 Science DMZ와 데이터 전송 노드(DTN)로 통합하는 다양한 방법이 있고, 그러한 환경의 필요성과 유지보수자의 능력에 의존적으로 이루어진다.

o 흔히 VC가 가장 유용한 경우는 캠퍼스 또는 연구 기관을 함께 연결하는 경우이다. 이것이 종단에서 사용자에 의해 직접적으로 제어되지 않을지도 모르는 인프라의 몇몇 구성요소를 확장해야 한다는 것을 의미한다. 다음 문서는 Science DMZ를 통해서 직접 연결의 목표를 달성할 수 있는 몇 가지 옵션들을 알아본다. 그런 방법 중에 하나는Layer 2 스위칭 및 Layer 3 라우팅이 목표를 달성하기 위해 공존할 가치가 있을 수 있다는 것이다.

태그된 VLAN을 사용하는 인터페이스

o VLAN은 한 스위치의 논리 세그먼트를 형성하는 포트들의 그룹이다. VLAN의 포트는 노드에 의해 생성된 트래픽이VLAN 내에 남아있는 독립 트래픽 도메인을 형성한다. 이것은 스위치의 관리 인프라를 통해 사용자의 네트워크를 분할 수 있도록 해준다. 그 결과 노드들은LAN상의 논리 세그먼트로 그룹이 될 수 있다. 혹은 WAN상일 지라도 OSCARS가 사용된다면 논리 세그먼트로 그룹이 가능하다.
만일 양쪽의DTN들이 그들 사이(예를 들어 RFC 1918 주소공간을 사용할 경우)의 연결에 상응하는 태그 된VLAN을 설치하고 사용할 능력이 있다면, 그리고, 같은 VLAN 태그들을 수용할 기계들 사이에 중계 네트워크를 성사시켰다면, Layer 2 접속을 설정하는 것이 가능하다. 어플리케이션 프로그램은 여전히VLAN의 다른 구성원과 연결하기 위해 Layer 3 로컬 주소를 사용한다. 전역 자원에 대한 접근은 듀얼호밍(dual homing)기술을 사용해서 완성할 수 있다

o 이 설정은 작은 수로 구성된 컴퓨터 그룹이 포함된 작업 플로우에 아주 특화된다. 그 컴퓨터들은 일반 인터넷 라우팅 인프라로부터 분리 할 수 있는 컴퓨터들이다. DYNES 협업은 이런 설정을 사용하는데 여기서 동적 회선은 참가하고 있는 장비들 사이에서 점대점 기반으로 설정될 수 있다.

BGP를 따르는 라우팅 장치 상의 P2P/VLAN

o 정적이거나 OSCARS나AL2S와 같은 시스템을 통해서 구성된 가상회선은 같은 라우팅 장치들 사이에서 설정된다. 그래서 접속된 어떤 DTN들도 정상적으로 작동될 수 있고, 라우터는 보통의 Layer 3 경로를 통해서라기보다는 가상선로 상의 DTN들 사이로 트래픽을 전달할 수 있다. 이는 라우터가 가상회선의 종단에서 주소를 가지고 있어야 가능하다. 이것은 더 좋은 지속성을 의미한다. 즉, 일단 이것이 설정되기만 하면, 그리고, 가상회선이 설정되면 트래픽은 그 회선을 탐색하고, 그렇지 않으면 트래픽은 최선의 경로를 탐색한다. 이것은 종종 가상회선상의 두 개의 Science DMZ 라우터사이에 BGP(Border Gateway Protocol)를 실행하고, BGP세션에서 DTN을 위한 경로를 교환함으로써 달성된다.

o LHC 실험은 여러 해 동안 이 모델을 사용해 왔다. 점대점 가상회선은 장치들 사이에 만들어지고, 주소 공간은 회로에 구성되고, BGP 시그널링에 사용된다.

정적 정책 라우팅(Static Policy Routing)

o 가상 회로를 통해BGP 세션을 설정하는 것이 바람직하지 않다면, 오랫동안 지속 가능한 가상회선을 설정하는 것이 가능하고 정적 정책 라우팅을 설정할 수 있다. 이것은BGP에 대한 필요성을 피할 수 있지만, 만일 가상 회로가 다운되고 종단에 연결된 DTN이 디폴트인 경우에는 통신할 수 없다는 것을 의미한다. 이 정책은 고장자동대체 능력이 있도록 구성한다. 그러나 이것이 종종 그 정책 라우팅이 가상회선의 상태를 감지할 필요가 있음을 의미한다. (BGP는 자동으로 수행하는 어떤 것)

통합 이더넷 상의 RDMA (RoCE)

o InfiniBand 아키텍처와 같은 RDMA 프로토콜은 전통적으로 데이터 센터 환경에서 패브릭 상호연결 스위칭을 통해 낮은 대기 시간 및 높은 처리량을 가능하게 만드는데 중요한 역할을 해왔다. RDMA는 한 시스템의 사용자 정의 메모리로부터 다른 시스템으로 데이터를 직접 전송하는 원리에 따라 작동한다. 이 전송 동작은 네트워크를 통해 수행되고, 사용자와 커널 메모리 공간 사이의 데이터 복사를 불필요하게 만들면서 운영 체제(OS)를 우회 할 수 있다. 직접 메모리 조작은 네트워크 어댑터가 어플리케이션에 의해 할당 된 버퍼를 등록 할 수 있도록 지

원한다. 통합이더넷(RoCE) 표준 상에 떠오르는 기술 RDMA는 사용자가 폭 넓게 확장된 이더넷 네트워크를 통해 이러한 효율적인 통신 패턴을 이용할 수 있도록 한다.

o 어떤 경로 특성은 광역 네트워크 상에서 RoCE프로토콜을 효율적으로 사용하기 위해 필요하다. 해당 경로는 트래픽 손실이 거의 없어야 하고 결정적이어야 하고 OSCARS 회선에 의해 지원되는 것처럼 대역폭 보장이 있어야 한다. 작은양의 손실이나 재순서화가 RoCE 성능에 해로운 영향을 미칠 수 있다. 주목할 만한 것은 RoCE 을 할 수 있으려면Mellanox 사의 은RoCE 를 지원하는 네트워크 인터페이스 카드(NIC)가 필요하다.

o 만일 네트워크에DTN이 RoCE(통합 이더넷 상의 RDMA)을 사용하고자 하는 경우, Layer 2 연결이 이 프로토콜을 사용하도록 요구한다. OSCARS 회선이DTN 어댑터를 직접 연결하는 것을 제안한다. 중간에서 스위칭 인프라는 그 회선에 의해 요구되는 QoS 파라메터를 따른다. 이 목적을 위해 정적으로 구성된 VLAN을 사용하는 것이 가능하겠지만 그 프로토콜의 성능 기대치는 달성하기 어렵게 될 것이다.[2]

1.6. 소프트웨어 정의 네트워킹(Software Defined Network)

o 소프트웨어 정의 네트워킹 기능은 Science DMZ에서 하드웨어에 의해 지원 될 수 있다. 소프트웨어 정의 네트워킹과 OpenFlow가 과학적 플로우에 대한 경로 설정을 위해 유연한 프로비저닝(provisioning)을 할 수 있도록 해준다. 사이트 경계 근처에서 단일 위치에 Science DMZ 구성 요소를 가진다는 것이 의미하는 것은 Internet2와 GENI 프로젝트, 또는 기타 유사한 자원과 같은 새로운 기술을 구성하고 설치하기 위한 단일 위치가 있다는 것이다.

o 아래 그림에서와 같이 SDN 대한 일반적인 사용의 경우는 일반 인터넷으로부터 높은 프로파일 플로우(high profile flow)을 식별하고, 그들을 범용 인프라로부터 제거할 수 있도록 지원하는 것이다.

o SDN 구성 요소가 있으면 식별된 큰 플로우(코끼리 플로우)은 범용 인프라 구조로부터 제거 될 수 있고, 동적으로 제어 링크 상에 배치될 수 있다. 이런 분리는 모든 경우에 성능 향상을 유발한다.

o ESnet, 인디애나대학과 델라웨어대학의 공동 작업으로 OSCARS와 같은 가상 회선 서비스와 상호 운용하는OpenFlow 기반 Science DMZ 아키텍처를 실증해왔다. 이런서비스들이Science DMZ와 상호 운용되는 방법이 아래에 설명된다.

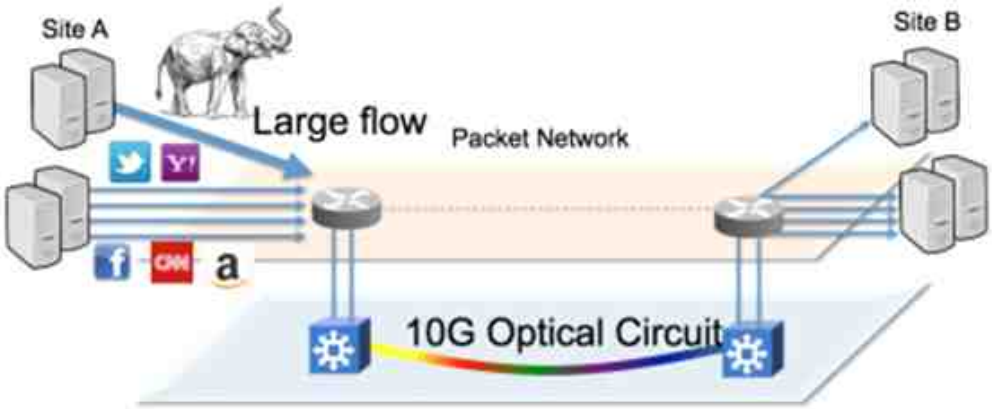

그림 4 SDN 일반적인 사용의 경우

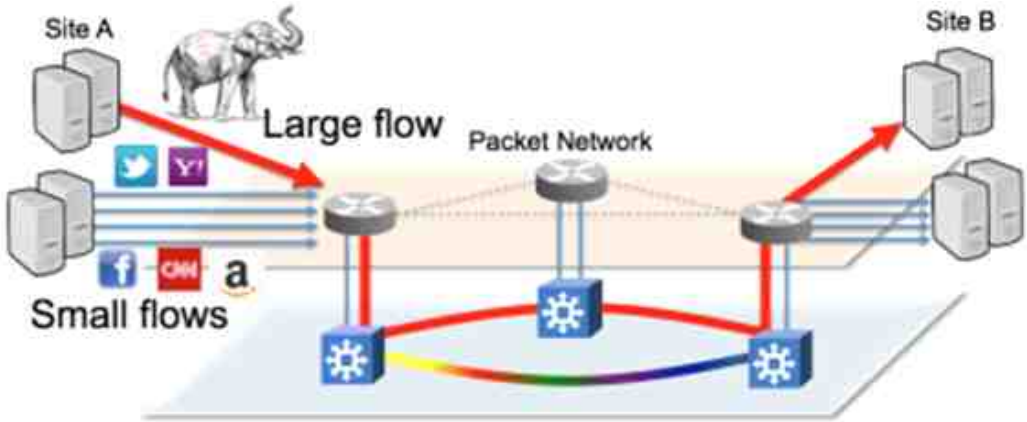

그림 5 SDN분리의 경우

서비스 통합: 테스트에서 배포까지

o Science DMZ 모델은 운영 체제는 일단 새로운 서비스가 견고하게 동작한다고 증명되기만 하면 새로운 서비스를 테스트, 검증 및 제품으로 출시되도록 해준다. 특히 플랫폼으로OpenFlow를 사용하는 네트워킹 정의 소프트웨어의 테스트와 배포는 이 모델이 어떻게 사용될 수 있는지에 대한 적절한 예이다. 잠재적으로 제품화할 서비스의 실험 전에 연구 Science DMZ를 생성 할 것을 권장한다.

o 초기에OpenFlow가 가능한 연결이 Science DMZ 영역으로 유입될 수 있고 독립형 스위치에 접속 될 수 있다. 별도의 시험 호스트는 프로토타입을 목적으로 독립형 스위치에 접속 될 수 있다.

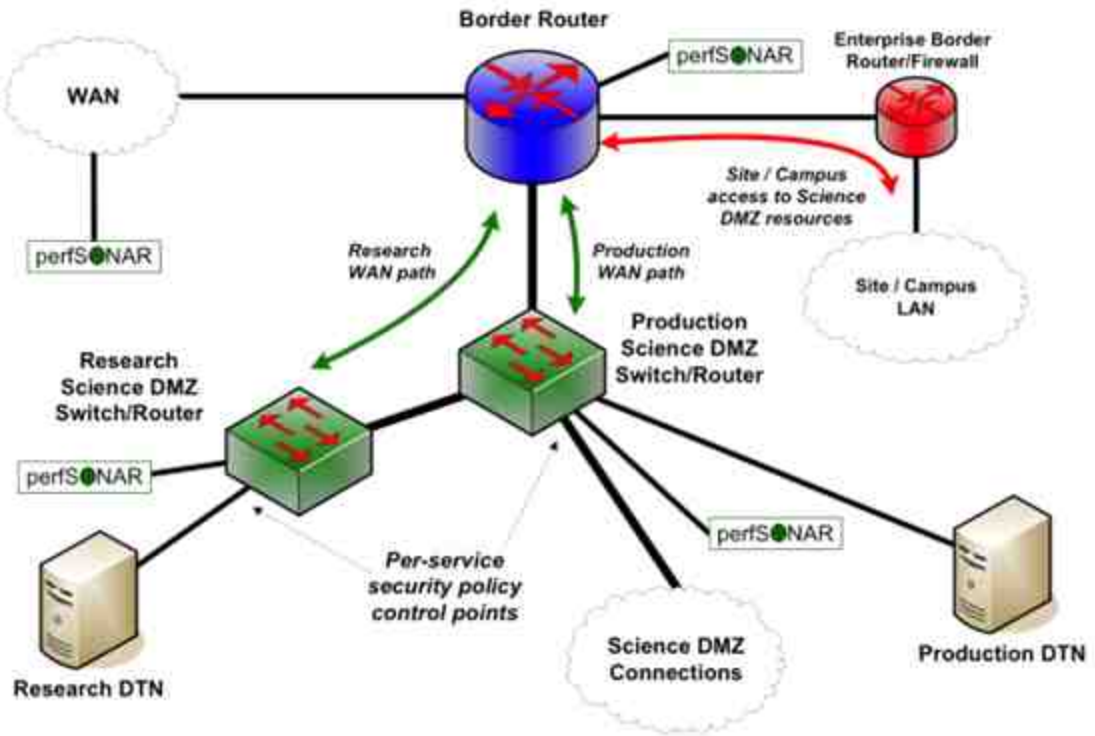

그림 6 Science DMZ 기본 구조

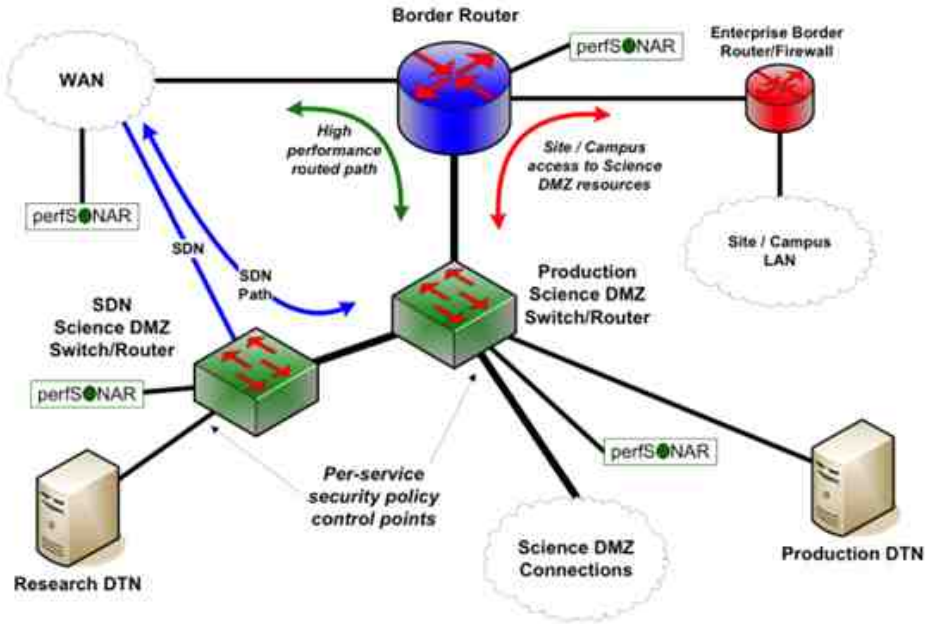

그림 7 SDN 적용을 위한 Science DMZ 구조

o Science DMZ 모델의 여러 기능들은 이미 현 시스템에 적용이 되어 동작하고 있다: OpenFlow 스위치는 프로토타입 서비스를 테스트하기 위해 필요한 최소의 호스트 집합만 접근할 수 있도록 허용할 필요가 있다. 그래서 제품 인프라의 보안상 위험은 없다. 이런 방법으로 프로토타입을 프로비저닝(권한설정, provisioning)

함으로써, 서비스는 생산 배포할 준비가 되기 전에 네트워크 연결 상태를 추적할 수 있는 방화벽(stateful firewall) 이나 보안 메커니즘이 최첨단 서비스를 지원해야 하는 선행 조건 없이 테스트 될 수 있다.

o 서비스가 즉시 생산 가능한 것으로 결정되고 새로운 서비스에 대한 보안 모델의 점검이 끝난 후에, 테스트 호스트는OpenFlow 스위치로부터 제거 될 수 있고, OpenFlow 스위치가 Science DMZ에 접속될 수 있다. 이렇게 함으로써, Science DMZ는 효과적으로 기존 Science DMZ환경에 최소한의 변경만 수행하면서 OpenFlow 가능 서비스를 포함하도록 확장 할 수 있다. 일단 OpenFlow 기술이 또 다른 제품화Science DMZ기능들을 지원하는 장비에서 사용할 수 있게 되면 Science DMZ 코어 하드웨어는 완전히 새로운OpenFlow 기반 서비스를 통합하는 방법으로 업그레이드가 가능하다.

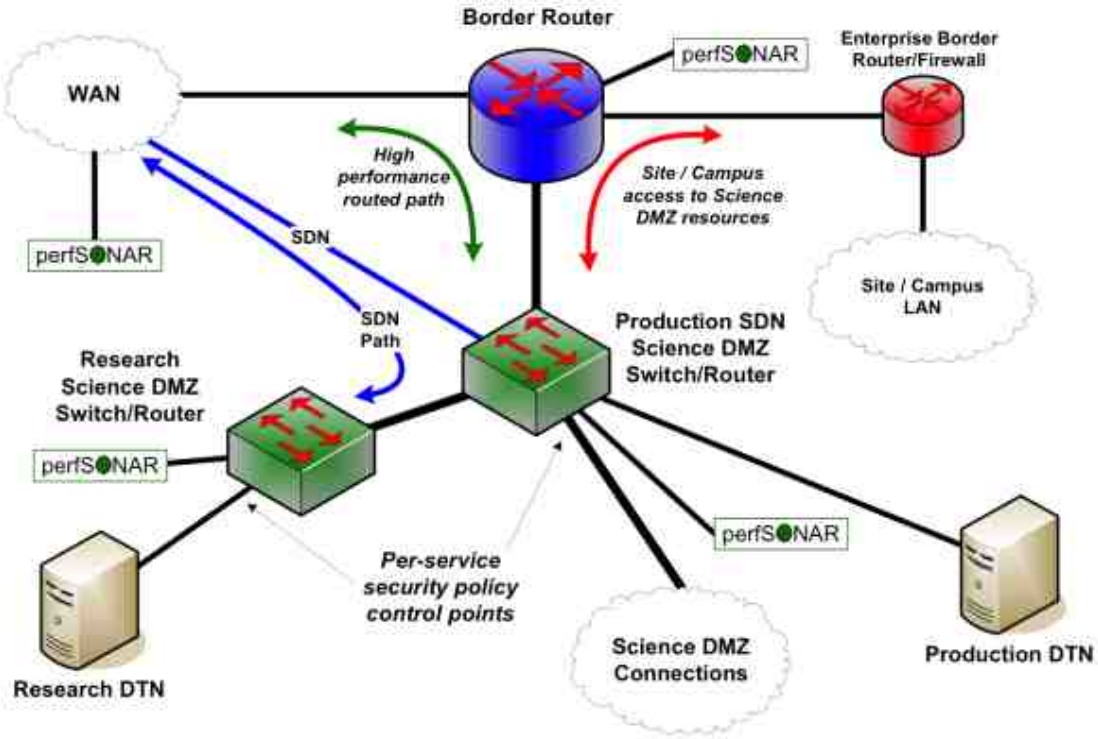

그림 8 SDN 적용을 통한 Science DMZ 망분리

오픈 익스체인지 소프트웨어 스위트(OESS)와 플로우 스페이스 방화벽(FSFW)

o OESS는 OpenFlow가 활성화된 스위치상에서 동적(사용자 제어) layer 2 가상 회선(VLAN) 네트워크를 제어하고 설정하기 위해 사용하는 일련의 소프트웨어 집합이다. OESS는 초단위 이하의 회선 프로비저닝, 자동 회선 장애 조치, 인터페이스 별 권한 및VLAN 당 자동 통계를 제공한다.

o FlowSpace 방화벽(FSFW)는OpenFlow 활성화 스위치의 네트워크 가상화를 제공한다. 가상화는VLAN 태그 별로 발생하거나 각 인터페이스 기반 별로 발생한다. FlowSpace 방화벽은 규칙을 해석하거나, FlowVisor처럼 플로우모드를 수정하려는 시도하지 않고, 규칙을 통과시키거나 거부한다. 그리고, 컨트롤러에 오류 메제시를 보낸다. (GlobalNOC Monitoring)[3]

1.7. 클라우드 리소스 (Cloud Resource)

o 상용 혹은 R&E 서비스 공급자들의 의해 제공되는 클라우드의 사용은 제한된 시간에 자원을 처리 해야 할 필요가 있는 연구자들에게는 매우 매력적이다. 클라우드 솔루션과 독자의 DMZ를 통합한다는 것은 클라우드로의 네트워크 경로를 안다는 전제와 데이터 이동성에 대한 기대성능을 만족시켜야 가능하다.

클라우드의 종류

o 클라우드는 아마존과 마이크로소프트 같은 상업적 제안에서 Jetstream,Clouldlab, 그리고 Chameleon Cloud와 같은R&E 자금지원으로 만들어진 것까지 다양하다. 네트워킹 관점에서, 다음 선택은 사용자가 클라우드와 인터페이스하는 방법이다: 즉, 데이터 이동성 프로파일을 파악하는 것이다. 일반적인 경우는 다른 컴퓨팅 리소스와 클라우드를 같이 취급하는 것이다; 데이터가 클라우드로 전송되고 결과가 얻어지는 구조이다. 클라우드 서비스 제공자와 어떤 관계에 있는냐에 따라 데이터 유입에 대한 요금이 발생할 수 있다.

네트워크 경로

o 직접, 또는 지역 또는 백본 네트워크를 통해 클라우드 제공 업체와 피어링(peering)을 수립하는 것이 트래픽 플로우를 관리할 수 있는 효율적인 방법이다. 이렇게 하면 잠재적 트래픽 혼잡 네트워크를 통과하지 않도록 함으로써 정확한 데이터 전송의 가능성을 증가시킬 것이다.

데이터 이동

o Globus와 같은 툴들은 많은 클라우드 포털의 데이터 전송 엔진으로 사용된다. 가능한 경우 프로그램 가능한 API를 원격 소스에서 로컬로 자동으로 데이터 이동을 용이하게 하기 위해 사용할 것을 권고한다.

성능 측정

o 클라우드 리소스에 대한 성능 검증은 네트워크와 호스트 가상화의 성질 때문에

어려울 수 있지만 일반적인 경우, 네트워크 문제의 원인을 제거하기 위해 피어링 관계가 클라우드 서비스 제공자와 형성되었을 경우와 클라우드 내부에서 구입된 장비가 더 많은 가용 시스템 자원을 부여 받았을 때 고수준의 성능을 볼 수 있다.

1.8. 데이터 전송 노드(Data Transfer Node)

o 광역 데이터 전송을 위해 사용되는 컴퓨터 시스템들은 그 시스템들이 범용으로 만들어졌거나 광역 데이터 전송 기능을 위해 특별히 구성될 때보다 훨씬 성능이 좋다. 데이터 전송 노드(DTN)라고 부르는 이런 시스템들은 고품질 컴포넌트와 광역 데이터 전송을 위해 특별히 구성된PC 기반 리눅스 서버이다. DTN은 또한 로컬 저장장치에 대한 접근 권한이 있다. 로컬 저장장치의 구성은 Lustre나GPFS, 또는 이들의 조합과 같은 고속 병렬 파일 시스템의 다이렉트 마운트나 SAN과 같은 로컬 저장 인프라에 대한 연결인 로컬 고속 디스크 서브시스템이다. DTN은 원격 시스템에 고속 데이터 전송을 위해 설계된 소프트웨어 툴을 실행하다. - 전형적인 소프트웨어 패키지를 의미하며 GridFTP와 서비스 지향적인 Globus Online, XRootd 와 같은 특정 분야의 도구, 고성능 패치가 적용된 SSH / SCP와 같은 기본 툴셋 버전을 포함한다.

o DTN은 일반적으로 고속 네트워크 인터페이스를 가지고 있지만(40Gbps의DTN 실험이 이미 진행되고 있지만, 현재는100Gbps), 핵심은DTN이 주요 사이트 및 광역 네트워크 인프라 구조의 기능과 일치한다는 것이다. 따라서, 예를 들어, 한 사이트에서 WAN을 잇는 네트워크 연결이1 기가비트 이더넷 인 경우에DTN 상의10 기가비트 이더넷 인터페이스는 역효과가 날 것이다.

o 보안 위험을 완화하기 위해 어떤 범용의 컴퓨팅 작업도 DTN상에 허용되지 않는다. 웹브라우저, 미디어 플레이어, 문서편집기나 스프레드시트 편집기와 같은 비즈니스 생산성 도구, 이메일 클라이언트 등, 오늘날의 환경에서 범용 컴퓨팅의 안전을 보장하기 위해 필요한 보안 제어가 필요한 것들은 모두 허용되지 않는다. 여기서는 하드웨어 선택과 DTN 노드의 튜닝에 도움이 될 만한 내용으로 설명된다. 또한 소프트웨어와 성능 테스트에 대한 정보도 함께 설명된다.

o DTN 디자인에 대한 자세한 내용은 2012 년1 월, Baton Rouge, LA.에서 개최된 ESnet/Internet2 Joint Techs conference에서 유타 대학의 엔지니어들이 발표한 튜토리얼 “Science DMZ 달성”에서 DTN 섹션에서 확인할 수 있다

FIONA- Flash I/O Network Appliance

o 캘리포니아 샌디에고 대학의 연구원들이 10G 및40Gbit/s 캠퍼스 규모의 데이터 인프라를 정의한 공로로 NSF에서 수여하는CC-NIE 상의 수상자가 되었다. 이러한 노력의 일환으로, Phil Papadopoulos와 Larry Smarr 는 이 인프라의 사용자를 위한 저가형의 고성능 Science DMZ 데이터 전송 노드(DTN)를 개발했다. 이 플래시I/O 네트워크 기기(FIONAs)는UCSD가 원 부품으로부터 구축한 PC들 이다. 그러나 "데이터 슈퍼 커패시터" 역할을 하고 데이터 중심의 어플리케이션에 최적화되어 있다. 이러한 노드는 원래 40Gbps 네트워킹 인프라를 디버깅하기 위해 만들어 졌지만 데이터 전송 활동을 위해 폭넓게 사용되고 있다.

o 2014년 12 월 스탠포드 대학은 하루 동안의 워크샵을 개최하여 다수의 CENIC캠퍼스에서 온 네트워크 엔지니어들이 제안된 인프라의 기존 구성요소들에 위의 Science DMZ들 사이에서 고성능 데이터 전송 원리 증명 데모에 대해 2015년 초반 10주동안 집중적으로 작업하자는 결정을 이끌었다. 이것은 CENIC의 존 헤스(John Hess)가 주도하였고 캠퍼스들 사이에 광범위한 협력이 요구되었다. 그 결과 Pacific Research Platform (PRPv0) 이 2015 년3월 9일 캘리포니아 어바인 대학(UC Urvine)에서 에 개최된 CENIC 2015에서 공표되었다. 결과에는 한 캠퍼스의 Science DMZ 내에서 다른Science DMZ로 DTN-to-DTN 데이터 전송을 포함한다. 반복 및 튜닝 후에 실행한 테스트(예를 들어, UCB, UCI, UCD 와 UCSC 캠퍼스에서 UCSD 캠퍼스로 전송)에서GridFTP를 사용하여 10중 9.6Gb/s의 성능을 데모하였고, FDT를 사용한 테스트(UCLA와Caltech 캠퍼스에서 UCSD캠퍼스로 전송)에서40 중에 36Gb/s의 속도로 두 번의 전송을 데모했다. 이DTNs의 대부분은 데이터 중심의 네트워킹에 최적화된FIONA PC들이었다. FIONA PC들은 Calit2/QI 소속의 존 그래햄(John Graham) 과 조 키페(Joe Keefe)에 의해 UCSD에서 만들어져서 이번 실험을 위해 캠퍼스들에 설치 되었다.

o FIONAs는 10-40-100Gb/s의 네트워크에서 대용량 데이터(수십에서 수백 테라 바이트)를 저장, 포워딩, 혹은 직접 사용을 위해 최적화된 단일 또는 클러스터된 데이터 전송 노드(DTN)이다. FIONAs는 SDSC의 고든(Gordon) 슈퍼컴퓨터 노드의 PC 상품 버전으로 개발되었다. FIONAs는 확장성이 있는 플래시 디스크(2-16TB)와rotating disk(5-120 TB) 그리고 최대 메인 메모리를 가진6-20개의 코어들을 가지고 두 개의40G/10G NIC 호스트와 GPU는 옵션으로 사용하는 조건하에 면밀하게 제작되고 테스트했다. 그것들은perfSONAR 40G 메모리간 네트워크 테스트 박스를 위해2000 달러, 빠른 네트워크에서 DTN역할을 하기 위해 충분한 플래시 디스크를 위해 7000 달러, 중요한 데이터 전송 노드와 보증된 로컬 서버로 사용될 100TB 이상의 디스크를 위해 15000달러 이상의 비용이 든다.[4]

o 우리는 그것들을 실험실 환경을 위해 데스크 사이드 유닛(조용하고 큰 팬 상자,

데이터 수집 및 서버 역할을 의미)으로 구축하였다. 또한 네트워크 스위치 룸으로 랙 마운트2U 또는3U 시스템으로 구축하였다. 모든 것은 10G 및1G에 쉽게 적용시킬 수 있는 40G NIC과 호환된다. AC 전원 없이 48V DC 버전은 통신 사업자형 스위치 룸에서 사용하기 위해 테스트되었다. FIONAs는perfSONAR, GridFTP 및 실시간 성능 기능, 전송의 측정 시각화를 제공하고 이런 측정지표의 보관을 위한 수단을 제공하는 JAVA FDT 를 실행한다.

o DTN Hardware Specification (October 2015)

표 1 DTN-FIONA Hardware Specification

항목	사양
Chassis:	Supermicro 3U Chassis 836BE1C-R1K03B (black) CSE-836BE1C-R1K03B
HDD Module	Supermicro MCP-220-83605-0N Rear window 2x 2.5" HDD module for 836B series chassis
Motherboard:	SUPERMICRO MBD-X10SRL-F Server Motherboard (N.B. the X10SRA-F is also an acceptable choice)
Processor	Intel Xeon E5-2643 v3 Hexa-core (6 Core) 3.40 GHz Processor - Socket R3 (N.B. the Xeon E5-1650v3 works as well)
RAM:	8 x (128GB Total) MEM-DR416L-SL01-ER21 - SUPERMICRO - 16GB (1X16GB) 2133MHZ PC4-17000 CL15
SSD Storage	***SSD Boot Drives***: 2 x Intel 535 Series SSDSC2BW240H601 2.5" 240GB SATA III MLC Internal Solid State Drive (SSD)
LSI HBA	LSI 9300-16i(LSI00447) PCIe 3.0 SAS 12Gb/s SAS Host Bus Adapter
CPU Cooler	Supermicro SNK-P0048AP4 CPU Cooling
SAS Cables	4 x Supermicro CBL-SAST-0532 Cable
TPM	SuperMicro AOM-TPM-9655V
40GbE NIC:	2 x Mellanox ConnectX-3 Pro EN Single-Port 40/56 Gigabit Ethernet Adapter Card
40GbE to 10GbE SFP+ Adapter	2 x Quad to Serial Small Form Factor Pluggable (QSA) Adapter

o perfSONAR Hardware Specification (October 2015)

표 2 perfSONAR Hardaware Specification

항목	사양
Chassis:	Supermicro CSE-813MTQ-R400CB Black 1U Rackmount Server Case 400W Redundant
Motherboard:	Supermicro MBD-X10SRL-F Server Motherboard (N.B. the X10SRA-F is also an acceptable choice)
Processor	Intel Xeon E5-1620 v3 Haswell-EP 3.5GHz 4 x 256KB L2 Cache 10MB L3 Cache (N.B. the Intel Xeon E5-2643 works as well)
RAM:	4 x (32MB Total) Samsung 8GB 288-Pin DDR4 SDRAM ECC Registered DDR4 2133 (PC4-17000) M393A1G40DB0-CPB
40GbE NIC	Mellanox ConnectX-3 Pro EN Single-Port 40/56 Gigabit Ethernet Adapter Card
40GbE to 10GbE SFP+ Adapter	Quad to Serial Small Form Factor Pluggable (QSA) Adapter
Storage:	2 x (480GB Total) Intel 535 Series SSDSC2BW240H601 2.5" 240GB SATA III MLC Internal Solid State Drive (SSD) - OEM
Storage Adapter	2 x Supermicro - storage bay adapter MCP-220-00043-0N
PCIe Riser:	Supermicro RSC RR1U-E16 - riser card
CPU Cooler:	Supermicro SNK-P0057PS CPU Cooling
TPM:	SuperMicro AOM-TPM-9655V

1.9. 성능 모니터링(Performance Monitoring)

Science DMZ 성능 모니터링

o Science DMZ 아키텍처는 perfSONAR를 기반으로 한 테스트 및 측정 호스트를 포함한다. 이 호스트는 Science DMZ상의 고장 진단에 유용하고 협력 사이트들에 perfSONAR가 설치되었다면 그 사이트를 이용한 종단간 테스트에 유용하다. perfSONAR 호스트는 OWAMP를 사용하여 대기 시간 변경 및 패킷 손실에 대한 연속 검사를 실행할 수 있을 뿐만 아니라 BWCTL을 사용하여 원격지의 정기적인 처리량 테스트를 수행할 수 있다. 라우팅 및 스위칭 인프라 문제를 해결하기 위해 네트워크 엔지니어를 필요로 하는 문제가 발생하면 문제를 처리하기 위한 패키지들이 이미 개발 완료되어 사용되고 있으며, 이런 도구들은 문제 해결을 시작하기 전에 설치 될 필요가 없다.

1.10. 보안 (Security)

o Science DMZ 의 배경 핵심 아이디어는 보안 정책에 초점을 맞추고 있다. 보안 방법에는 두 가지가 있다:

- 이 연구를 기반으로 완화 전략의 생성과 위험의 식별
- 다양한 제어의 구현

Science DMZ 는 위 두 방법을 모두 사용한다. 위험을 완화하기 위해 네트워크를 분할하고 확인 된 항목에 대해 적절한 제어를 설치할 것을 강조한다.

요약(Summary)

o Science DMZ 는 성능면에서 부정적인 영향 때문에 Science DMZ를 보호하기 위해 방화벽을 사용하지 말 것을 제안한다. 대신 라우터와 스위치의ACL 및 기타 보안 모범 사례를 사용할 수 있다. 이것은 논쟁이 있을 수 있는 문제이지만 중요한 이슈이다 정보 시스템의 보안은 매우 필수적이고 중요하지만 대다수의 과학적 어플리케이션은 매우 높은 네트워크 성능을 필요로 한다 - 링크 속도에서 고성능을 의미하는 것이 아니라, 그 어플리케이션에 전달된 처리량에서의 고성능도 함께 포함한다.

o 방화벽은 일반적으로 고성능 과학 환경에 잘 맞지 않도록 설계되고 만들어진다. 아래에 설명처럼 방화벽은 과학 어플리케니션을 위한 특별한 분석 기능이 없다. 과학 어플리케이션을 위해 방화벽이 할 수 있는 것은 단지 IP 주소와 포트 번호에 의한 네트워크 트래픽 필터 기능이다. 이는 라우터 액세스 제어리스트(ACL)가 보안 보호를 제공하기 위해 사용하는 방법인IP 주소 및 포트 번호에 의한 트래픽 필터링과 매우 유사하다.

o Science DMZ 모델의 가장 큰 장점은 네트워크 및 보안 설계자가 과학을 위해 매우 중요한 시스템의 방어에 사용되는 도구와 기술을 최적화할 수 있도록 해준다는 것이다. Science DMZ 모델에서ACL은 고성능 과학 응용 프로그램을 보호하기 위해 사용되며, 기관 또는 부서 방화벽은 현재 방화벽이 하는 역할과 마찬가지로 비즈니스와 말단 사용자의 시스템을 보호하기 위해 사용된다. ACL은 일반적으로 라우터의 포워딩 하드웨어로 구현되기 때문에 그것들은 일반적으로 고성능 애플리케이션의 성능을 손상시키지 않는다.

o 데이터 집약적인 과학이 과학의 많은 분야에서 표준이 됨에 따라, 고성능 데이터 이동이 빠르게 과학 인프라의 핵심 요구사항이 되고 있다. Science DMZ를 배치함으로써, 연구 기관은 네트워크 보안 및 기관의 과학 임무 사이에서 어느 한쪽

을 포기하는 선택을 해야 할 필요 없이 높은 성능을 달성 할 수 있으며, 기관의 시스템을 방어할 수 있다.

o Science DMZ에 있는 데이터 집약적 과학 환경에 대한 보안은 Science DMZ의 데이터 전송 시스템에 맞추어 질 수 있다. 이러한 호스트들은 사용자 어플리케이션의 일반적인 나열보다는 차라리 일반적으로 잘 정의되고 제한된 특수 목적의 애플리케이션을 실행한다. Science DMZ 자원은 외부 시스템과 상호작용할 것을 가정하고, 내부 시스템과 분리되고, 내부 시스템에 대한 아주 세심하게 관리되는 접근만 있기 때문에 Science DMZ에 대한 보안 정책은 일반 사이트LAN의 내부에서 보호하기 위해서라기 보다는 이러한 기능에 맞춰져 있다.

방화벽 기능(Firewall function)

o 방화벽 룰세트의 주 기능은 각각의 패킷이 일반적으로 방화벽 룰세트에 대해 일치하는 지 확인하는 과정에서 패킷 헤더 정보를 이용하여 네트워크 트래픽을 허용하거나 거부하는 것이다. 패킷이 보안 정책에 부합하는지 여부를 결정하는 데 사용되는 기본 기준은 소스IP 주소, 소스 포트(패킷이TCP 또는UDP 패킷 인 경우), 목적지IP 주소, 목적지 포트이다. 일반적인 관점에서 이 섹션은 대부분의 방화벽에 의해 수행되는 고수준 작업을 설명한다.

o 방화벽은 실시간으로 방화벽을 탐색하는 개별적으로 허용된 연결의 프로토콜 상태(소스/목적지 주소와 포트로 구성된 4 튜플에 의해 식별)를 추적하는 lookup테이블을 유지한다. 패킷이 방화벽에 도착하면, 방화벽은 해당 접속 상태 테이블에서 수신 패킷의 주소/포트4 튜플을 살펴본다. 패킷이 상태 테이블 엔트리와 일치하는 경우, 상태 테이블 엔트리가 업데이트 되고 패킷은 허용된다. 만일 상태 테이블에 엔트리가 없는 경우, 패킷은 방화벽 룰세트과 맞춰본다. 만일 패킷이 방화벽 룰세트에 의해 허용되는 경우, 새로운 상태 테이블 엔트리가 생성되고, 패킷은 허용된다. 만일 패킷이 룰세트에 의해 허용되지 않는 경우, 패킷은 삭제된다. 상태 테이블은 방화벽 작동의 핵심이다. - 상태 테이블이 차면 새 항목을 만들 수 없다(따라서 방화벽을 통해 어떤 새로운 연결도 설립 할 수 없다). 주소/포트4 튜플 조회가 빠른 동작 때문에 상태 테이블은 성능에 영향을 미치는 중요한 요인이 된다. 그러나, 상태 테이블의 자원은 유한하여 상태 테이블은 반드시 관리되어야 한다.

o 연결이 정상적으로 종료되면(예를 들면 만일 연결이 TCP 접속이고 방화벽이 양방향에서 FIN/ACK 시퀀스를 보거나, 혹은 만약 방화벽이 TCP 접속의 재설정을 보게 된다면) 방화벽은 상태 테이블로부터 관련 접속 상태를 제거한다. UDP와 같은 일부 프로토콜은 명시적 연결 상태가 없다. 그래서 연결 상태를 정리할 타

이밍을 결정하는 것이 어렵다. 또한, TCP 연결은 자주 정상적으로 종료하지 않는다(예를 들어, 만일 회의에 늦어서 그냥 랩탑을 닫으면 열린 연결은 연결상태를 정리하지 못하고 트래픽 전송을 멈춘다). 상태 테이블이 채워지는 것을 방지하기 위해서 상태 테이블 항목은 타이머에 의해 관리된다. 만일 상태 테이블 엔트리는 타임 아웃 간격(사용하는 프로토콜마다 잠정적으로 다르지만 전형적으로5 내지15 분) 이후 갱신되지 않은 경우, 방화벽은 연결이 끊어진것으로 간주하고, 상태 테이블에서 연결을 제거한다. 그러나, 일단 상태 테이블 항목이 사라지고 나면 해당 연결에서 전송된 패킷은 방화벽에 의해 거부된다 – 방화벽은 설정된 연결의 중간에서 패킷을 위한 상태 테이블 항목을 다시 설정하지 않는다.

o 현대 방화벽은 매우 큰 상태 테이블을 관리 할 수 있다 – 수백만의 연결이 일반적으로 지원된다. 이 트래픽 처리 모델은 상대적으로 많은 수의 낮은 데이터 볼륨으로 짧은 연결 지속시간을 갖는 현대의 비즈니스 기업의 트래픽 프로파일과 잘 매칭된다. 방화벽은 많은 병렬 패킷 처리 엔진들로 만들어 질 수 있다. 이런 엔진들은 초당10 기가비트의 속도로 수백의 접속을 처리할 수 있는 방화벽을 생성하기 위해 결합될 수 있다.

o 주소/포트 매칭과 연결 상태 관리와 더불어 더 진보된 많은 방화벽은 어플리케이션 레이어의 동작을 추적하기 위해 심도 있는 패킷 조사를 할 수 있다. 그런 방화벽은 이메일 트래픽을 감지하고, 즉시 바이러스 검사를 위해 이메일을 스캔할 수 있고, 악의적인 동작을 찾아내기 위해 웹 트래픽을 분석할 수 있다. 이런 어플리케이션 레이어 함수는 그 어플리케이션을 실행하는 많은 수의 기업 고객들 때문에 존재한다– 이런 기능에 대해서 넓은 시장이 있고, 방화벽에 공통 프로토콜을 위한 더 고급 분석 기능을 만들기 위해 R&D에 대해 투자 할 가치가 있다. 반면, 방화벽 공급 업체는 일반적으로 과학 애플리케이션을 위한 응용 계층 분석을 포함하지 않는다- 방화벽 기기에 분석 툴을 구축해 넣을 만큼 가치가 있도록 만들기에 시장이 너무 작다.

데이터 집약적인 과학과 방화벽의 상호작용(Interaction of firewalls with data-intensive science)

o 데이터 중심 과학의 공통 태스크는 다른 위치에서 많은 양의 데이터(테라 바이트 또는 그 이상)의 이동이다. 그 이유는 저장 장치 또는 몇몇 종류의 자원 분석 데이터를 얻기 위해 일반적이다. 큰 데이터 세트의 전송은 통상적으로 사용 가능한 경로 대역폭의 상당 부분을 사용한 소수의 TCP 연결을 포함한다. 또한, 이러한 전송은 긴 시간(때때로 몇 시간)이 소요될 수 있다. 그리고, 데이터 전송 어플리케이션은 전송의 시작과 끝에서 통신할 필요가 있지만, 대량 데이터의 이동 중단

에서는 항상 그렇지는 않다. 이 트래픽 프로파일은 여러 가지 면에서 일반적인 방화벽 설계와 일치하지 않는다.

- 방화벽은 다수의 연결을 관리하도록 설계된다 - 데이터 전송 애플리케이션은 일반적으로 단지 소수의 연결만 사용한다.
- 방화벽은 자주 전체 장치 대역폭보다 현저히 작은 최대 성능을 가진 많은 수의 처리 엔진으로 구성된다. (예를 들면, 10Gbps의 방화벽에1.2Gbps의 능력을 갖춘 8 패킷 처리기의 집합)

o 네트워크 장치의 내부 데이터 경로가 앞서 기술한10Gbps 방화벽에 대한 경우와 같이 장치의 인터페이스 속도보다 더 느릴 때, 고성능 애플리케이션은 명목적 네트워크 대역폭보다 현저히 낮은 데이터 전송율에서 패킷 손실을 유발할 수 있다. TCP의 버스티 특성 때문에 이런 방식으로 만들어진 방화벽 내부에서 종종 손실이 생기기 쉽다. 10GE 인터페이스가 있는 데이터 전송 호스트의 예를 고려하면 그 호스트는 상단의 방화벽이1.2Gbps로 처리할 수 있는 10Gbps의 패킷 버스트를 전송한다. 방화벽은 패킷이 낮은 속도로 처리되는 동안10Gbps의 버스트를 버퍼링해야 한다. 그리고, 방화벽이 패킷을 처리 할 수 있을 때까지 방화벽의 버퍼가 버스트를 보유할 수 없다면 어떤 패킷들은 삭제될 것이다. 방화벽이 신뢰할 수 있는 손실 카운터를 유지하지 않는다면 (그리고, 방화벽을 소유하고 있는 보안 그룹이 그 카운터를 공개하도록 설득될 수 없다면) 과학자는 방화벽이 원인이 되는 패킷 손실 때문에 네트워크의 성능이 저조하다는 것만 보게 된다.

o 특히 활동을 하지 않는 짧은 시간 이후에 유휴 연결의 제거와 같은 방화벽의 상태 테이블의 관리는 장기 실행 과학 어플리케이션과 네트워크 연결 상태를 추적할 수 있는 방화벽(stateful firewalls) 사이의 중요한 점이다. 만일 방화벽이 상태 테이블이 비는 것을 방지하도록 구성된다면, 이것은 장기 실행 애플리케이션과 종종 테라 스케일 데이터 세트에 대해 어플리케이션은 다른 일을 하고 있는 동안 유휴상태인 네트워크 연결을 끊는다. 만일 방화벽이 호스트TCP/IP 스택(2시간)에서 표준 지속 타이머에 상응하는 제한시간이 있는 연결상태를 유지하도록 구성되었다면 방화벽은 상태 테이블의 고갈 때문에 발생하는 고장에 취약하다. 이것은 이론적인 문제가 아니다– 우리는 이 문제 때문에 데이터 손상이나 전송실패와 같은 문제를 겪어왔다(한 경우는 데이터 전송이 완료되기 전에 제어 연결들이 방화벽의 상태 테이블에서 시간초과로 제거되는 대용량 데이터 전송을 들 수 있다. 결국 전송실패가 된다).

o 그들 자신의 지역 Science DMZ에 접근하려 하는 사이트 방화벽 뒤의 호스트는 종종 합리적인 성능을 달성 할 수 있다는 것에 주목하자. 그 이유는 지역 Science DMZ와 로컬 사용자 사이에 아주 낮은 대기시간이 실용적 관점에서 거

의 문제가 되지 않는 사이트 경계 방화벽에 의해 야기되는 일련의 문제가 되기 때문이다. TCP는 낮은 대기시간에서 손실을 빠르게 복구한다. 그리고, 단거리 TCP다이나믹스는 장거리 전송에 사용되는 TCP 다이나믹스와 충분히 다르다. 광대역 데이터 전송이 방화벽을 검색한다면 존재할 수 있는 패킷 손실은 아마도 로컬 사용자가 Science DMZ 자원을 접근할 때조차 일어나지 않는다. 요점은 서비스 장거리TCP 연결이 비혼선, 손실로부터 자유로운 서비스를 제공한다는 것이다.[5][6]

2. 호스트 튜닝(Host Tuning)

o 광대역 네트워크 전송을 이해 최대 I/O성능을 위한 1Gbps의 속도 이상으로 연결된 리눅스, 맥OSX와FreeBSD 호스트를 튜닝하는 방법에 대해 알아보자. 주목할 사항은 여기에 설명된 몇 가지 튜닝 설정 방법은 호스트의성능을감소시킬수있다.

- 배경 정보: 적절한 호스트 튜닝은 최대100 배 성능이 향상 될 수 있다. 배경정보 섹션에서 확인한다.
- 리눅스 튜닝: 리눅스 운영체제 튜닝 페이지는 리눅스2.6 튜닝, TCP 튜닝, NIC 튜닝 그리고 리눅스 2.6버전 이상의 버전에 대한 빠른 참조 가이드를 포함하고 있다.[7][8][9]

o 40G / 100G 조정: 40G/100G 이더넷NIC을 가진 호스트에 대해 처리량을 최대화하기 위해 몇 가지 추가적인 것이 있다. 구성하기 위해 가장 중요한 것들은 다음과 같다: CPU관리자는 성능을 위해 TCP 버퍼 크기를 최대치(2GB)로 설정한다. 개개 환경에 대해 좋은 페이싱율을 가진 공정큐잉(FQ)을 사용하는 IRQ와 사용자 프로세스에 대한 올바른 코어를 사용하고 있는지 확인해야 한다. 현대 리눅스 OS(RHEL/CentOS 7.2 이상, 또는4.x의 커널)에 대해서는 다른 어떤 튜닝도 필요하지 않다. [10]

o 패킷 페이싱: 빠른 호스트에서 느린 호스트로 전송 시, 패킷 손실 및TCP 백오프를 야기하고 수신기를 오버런하기 쉽다. 10G 호스트가 서브10G 가상 회선으로 데이터를 전송하거나 또는10G 호스트로 전송하는40G 호스트 또는 빠른CPU가 있는40G/100G 호스트가 느린CPU 가 있는 40G/100G 호스트로 데이터를 송신할 때 유사한 문제가 발생한다. 이런 문제는 GridFTP와 같은 병렬 스트림을 사용하는 도구를 사용하는 경우에 더 많이 보고되었다.[11]

o 가상 기계: 네트워크 처리량을 위해 가상 머신을 튜닝하는 경험이 많지 않지만, 다른 전문가가 알려준 몇가지 중요한 점은 다음과 같다. 이 페이지에 대한 업데이트나 정정할 사항이 있다면 보내주기 바란다. 네이티브 리눅스 호스트에서 실

행되는 리눅스VM에서 최고의 네트워크 성능을 얻으려면, 호스트OS에txqueuelen를 증가시키고 게스트OS의 다른 모든 튜닝 매개 변수를 설정한다. 자세한 내용은 다음을 참조한다: KVM튜닝, XEN 튜닝, VMWare튜닝(VMXNET3 드라이버를 사용)

2.1. 배경(background Information)

TCP 버퍼 크기

o TCP는 한 번에 보낼 수 있는 패킷 수를 결정하기 위하여 이른바"혼잡 윈도우" 또는CWND를 이용한다. 혼잡 윈도우 크기가 클수록 처리량은 커진다. TCP의 "슬로우 스타트" 및"혼잡 회피" 알고리즘은 혼잡 윈도우 크기를 결정한다. 최대 정체 윈도우는 커널이 각 소켓에 할당하는 버퍼 공간의 양과 연관된다. 각 소켓에 대해 버퍼 크기에 대한 기본 설정값이 있다. 이 설정값은 소켓 오픈 바로 전에 시스템 라이브러리 호출을 통해 변경될 수 있는 값이다. 또한, 커널이 요구하는 최대 버퍼 크기가 있다. 버퍼의 크기는 송신측과 수신측 모두에 대해 수정될 수 있다.

o 최대 처리량을 얻기 위해 사용하고 있는 링크의 최적 TCP 송수신 소켓 버퍼 크기를 사용하는 것이 아주 중요하다. 만일 버퍼가 너무 작으면 TCP 혼잡 윈도우는 결코 완전히 개방되지 못할 것이다. 또한 송신측 버퍼가 너무 크다면 TCP 플로우 제어가 끊어져 송신기는 수신기를 오버런 할 수 있다. 이것이 TCP 윈도우가 종료되는 현상을 야기할 것이다. 이런 현상은 전송 호스트가 수신 호스트보다 빠른 경우에 발생하기 쉽다. 수신 측의 지나치게 큰 윈도우는 여분의 메모리만 있다면 일반적으로 문제가 되지 않는다; 여기서 주목할 점은 모든 TCP 소켓이 짧은 연결에도 상당한 량의 메모리를 요구할 수 있기 때문에 시스템 자원을 고갈시킬 가능성이 있다는 것이다.[12]

o 최적의 버퍼 크기는 대역폭*지연의 두 배로 설정하는 것이다.

버퍼 크기= 2 * 대역폭* 지연

o ping 프로그램은 지연을 알아보기 위해 사용될 수 있다. 종단 간 용량(경로에서 최저 홉의 대역폭)를 결정하는 것이 다소 까다롭고 그 경로에서 다양한 네트워크의 용량을 알아내기 위해 주변에서 알려주도록 요청해야 할 수 도 있다. Pathrate와 같은 도구는 네트워크의 용량의 추정치를 제공한다. ping 프로그램이 왕복시간(RTT, Round Trip Time)을 알려주기 때문에 이전 수식보다 아래 수식이 사용될 수 있다.

버퍼 크기= 2 * 대역폭* RTT

o 예를 들어 ping 시간이 50 밀리 초이고 종단간 네트워크가 모두 1G 또는10G 이더넷으로 구성되어 있다면 TCP 버퍼는 다음과 같다.

0.05 *(1 Gbit/8 bits)= 6.26 Mbytes.

o 역사적으로 전체 대역폭을 얻기 위해 사용자가 사용되고 있는 네트워크 패스에 대한 버퍼 사이즈를 명시하도록 요구한다. 그리고, 어플리케이션 프로그래머는 BSD setsockopt() 호출의 SO_SNDBUF 및SO_RCVBUF 옵션을 사용하여 송신자 및 수신자의 버퍼 크기를 설정해야한다. 다행히도 리눅스, FreeBSD, 윈도우 및 Mac OSX 모두는 지금TCP 오토 튜닝을 지원한다. 그래서 기본 버퍼 크기의 설정에 대해 더 이상 걱정할 필요가 없다.[13]

TCP 자동튜닝

o 리눅스2.6, 맥OSX 10.5, Windows Vista 및FreeBSD 7.0을 시작으로 각 경로마다 수작업으로 TCP 송신과 수신의 버퍼를 설정한 필요성을 제거하도록 송신자와 수신자의 자동튜닝이 제공된다. 그러나 많은 고속 네트워크 경로에 대해 최대 버퍼 크기는 여전히 너무 작아서 각 운영체제마다 다음 페이지를 참고하여 상향 설정되어야 한다.[14]

TCP 자동튜닝 최대 크기

o 기본으로 설정된 리눅스 최대 TCP 버퍼 크기의 변경은 자동튜닝 알고리즘이 긴 경로에서 가용 대역폭을 최대한 활용하도록 송수신 윈도우의 크기를 조정할 수 있도록 한다는 것이다. 각 운영체제는 이러한 설정에 각기 다른 반응을 한다. 권고안의 운영체제별 자원 운용 부분을 참고한다.

다음 그래프는 왕복시간이 7.5 밀리 초의 간격으로 분리된 두 리눅스 서버에서 이 값을 변경한 영향을 묘사한다.

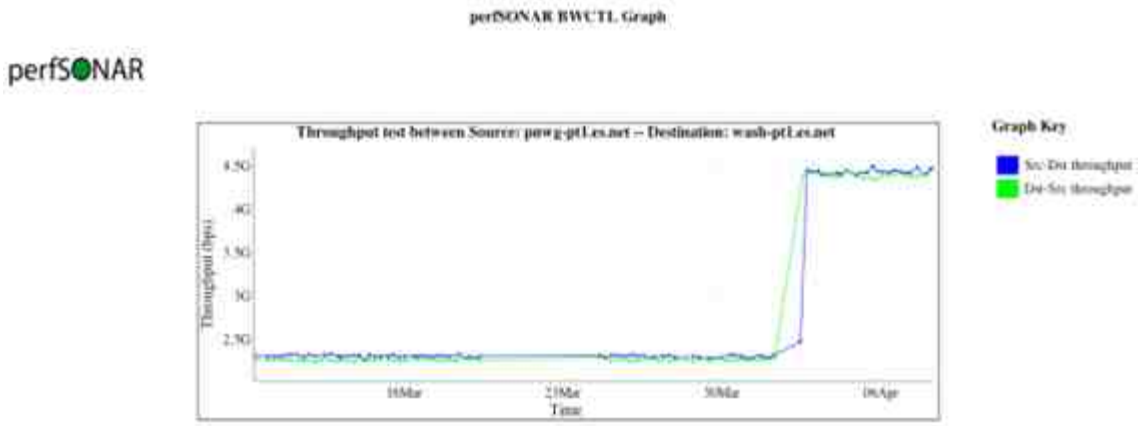

그림 9 TCP 버퍼 사이즈 조정에 따른 성능 영향

o ESnet은 이 값에 대해 32M에서 128M사이의 적정한 기본값을 사용할 것을 추천한다. 10Gbps, 단일 스트림, 100ms의 경로 통과시간에 대한 예상 버퍼 크기는 네트워크 손실을 감안하고 120MB 일 것이다. 더 빠른 속도나 더 긴 거리에 대한

기대치를 가지는 호스트는 더 큰 크기가 필요할 것이다. 메모리 소모를 방지하기 위해 병렬 스트림을 사용하려는 시도는 적게 사용해야 한다.

TCP 혼잡 회피 알고리즘

o TCP 혼잡 회피 알고리즘은 몇 년 동안 모든TCP 구현의 기본이었다. 그러나 네트워크가 더욱더 빨라질수록 는 고 대역폭 지연 생산 네트워크에서 잘 동작하지 않는다는 것이 명확해 졌다. 이 문제를 다루기 위해 다음과 같은 많은 새로운 혼잡 회피 알고리즘이 개발되었다:[15]

- reno: 대부분의 다른 모든 운영체제에서 사용하는 전통적TCP (default)
- cubic: CUBIC-TCP
- bic: BIC-TCP
- htcp: Hamilton TCP
- vegas: TCP Vegas
- westwood: 소실 네트워크에 최적화

o 최근 배포되는 리눅스는 기본으로 cubic을 사용한다. 윈도우는 이제 compound tcp를 사용한다. 만일 이전 리눅스 버전을 사용하고 있다면 기본으로 설정된 reno에서 cubic이나 htcp로 변경한다.

2.2. 리눅스(Linux)

o 이 페이지는1Gbps의 이상의 속도로 연결된 데이터 전송 호스트에 대한 리눅스 2.6+ 튜닝에 대한 빠른 참조 가이드가 포함되어 있다.

o 주목할 것은 이 페이지 상의 설정은 단일 플로우로 완전한10G를 달성하기 위해 시도하는 것이 아니다. 이러한 설정은 병렬 스트림을 지원하는 도구를 사용하고 있다는 가정을 하고, 병렬로 발생하는 다중 데이터 전송을 가정하고, 또 플로우들 사이에 공정 공유를 원하다는 가정을 한다. 그런 것과 마찬가지로 최대값은 단일 스트림을 지원하기 위해 요구되는 값 보다 2배 내지 4배 작다. 예를 들어, 100ms의 네트워크를 통과하는10Gbps의 플로우는 120MB 버퍼링 크기를 요구한다. (BDP 계산기 참조). GridFTP와 같은 대부분의 데이터 이동 응용 프로그램은 효율적으로 이 작업을 수행하고 혼잡 패킷 손실을 방지하기 위해2-8 스트림을 사용하려 할 것이다. 소캣 마다 32M-64M를 소비할 능력이 있는 10G호스트를 설정하는 것은 병렬 스트림이 잘 동작한다는 보장이 있어야 하고, 시스템 자원의 대부분을 소비하지 않는다는 보장이 있어야 한다.

o 사용하고 있는 시스템의 설정을 확인하려면 'sysctl name'(예: 'sysctlnet.ipv4.tcp_rmem')을 사용한다. 설정을 변경하려면 'sysctl –w'를 사용한다. 설정을 영구적으로 사용하려면 그 설정을 'sysctl.conf' 파일에 추가한다.

o 대부분의 현대의OS처럼 리눅스는 TCP 버퍼의 자동튜닝을 잘하고 있다. 그러나, 리눅스 TCP 버퍼 크기의 기본값은 여전히 너무 작다. 다음은 다른 종류의 호스트를 위한 sysctl.conf 명령어의 예이다.

o 최대100ms RTT의 네트워크 경로에 대해 최적화되고, 단일 및 병렬 스트림 도구에 친화성을 위해 최적화된 10G NIC를 가진 호스트의 경우 다음을 에 추가한다.

```
# 최대64MB의 버퍼를 사용하여 테스트를 할 수 있다.
  net.core.rmem_max = 67108864
  net.core.wmem_max=67108864

# 리눅스 오토 튜닝TCP 버퍼 제한을 32MB로 증가
  net.ipv4.tcp_rmem = 4096 87380 33554432
  net.ipv4.tcp_wmem=40966553633554432

# 권장 기본 혼잡 제어는htcp
  net.ipv4.tcp_congestion_control=htcp

# 점보 프레임 가용 호스트에 대한 권장 사용
```

```
net.ipv4.tcp_mtu_probing=1
```

CentOS 5/6 호스트에서는 이것을 /etc/rc.local(여기서 N은10G NIC의 번호)에 다음을 추가한다.:

```
/sbin/ifconfig ethN txqueuelen 10000
```

o 최대200ms RTT의 네트워크 경로에 대해 최적화되고, 단일 및 병렬 스트림 도구에 친화성에 최적화된 10G NIC를 가진 호스트의 경우, 또는 최대 RTT 50ms의 경로의40G NIC의 경우:

```
# 최대128MB의 버퍼를 이용하여 테스트를 할 수 있다
net.core.rmem_max = 134217728
net.core.wmem_max=134217728

# 리눅스 오토 튜닝TCP 버퍼 제한을 64MB로 증가
net.ipv4.tcp_rmem = 4096 87380 67108864
net.ipv4.tcp_wmem=40966553667108864

# 권장 기본 혼잡 제어는htcp
net.ipv4.tcp_congestion_control=htcp

# 점보 프레임 가용 호스트에 대한 권장 사용
net.ipv4.tcp_mtu_probing=1
```

참고: 기본으로 설정된 값이 좋기 때문에 는 그대로 둔다. 많은 성능 전문가들은를 그러나 우리는 어떤 차이도 발견하지 못했다. 어떤 전문가는 CPU 부하를 줄이기 위해 및 WAN 성능을 위한 권고안에는 절대 동의하지 않는다.
이전 운영체제의 경우(예: CentOS5/6), net.core.netdev_max_backlog를 증가시키면 성능도 향상된다. 리눅스는 장착형 혼잡 제어 알고리즘을을 지원한다. 커널(커널2.6.20+)에서 사용할 수있는 혼잡 제어 알고리즘의 목록을 얻으려면 다음을 실행한다

```
sysctl net.ipv4.tcp_available_congestion_control
```

대부분의 배포판이 적재가능한 커널 모듈로 포함하고 있기 때문에 만일 cubic이나 htcp가 리스트에 없으면 다음을 시도한다.

```
/sbin/modprobe tcp_htcp
/sbin/modprobetcp_cubic
```

참고: 버전2.6.33까지의 상당수의 리눅스 커널에 대해서는 bic나 cubic 모두에 버그가 있는 것처럼 보인다. 안전한 사용을 위해 더 오래된 커널의 경우 를 사용하길 권고한다.

```
sysctl -w net.ipv4.tcp_congestion_control=htcp
```

점보 프레임을 사용하는 경우, MTU 블랙홀 문제를 피하는데 도움을 주는 tcp_mtu_probing = 1을 설정을 사용하길 권고한다.

NIC 튜닝

o 부팅 시 실행되도록/etc/rc.local에 다음을 추가 할 수 있다:
주목할 점은 1G 호스트 또는 더 느린 호스트로 전송하는 10G 호스트에 대해 부정적인 효과를 줄 수도 있다. 우리는 그것이 하드웨어 부하제거 기능을 중지시킬 수 있을 때 호스트/NICS의 리눅스와VLAN을 사용하여 약30 %의 성능 향상을 관측했다.

2.3. NIC 튜닝(NIC Tuning)

인텔 1GE 및 10GE NICs

o e1000 칩을 사용하는 경우(인텔1GE, 보통 메인 보드에 통합되었다. 주목할 점은 이 사실이 새로운 변형 제품에는 적용되지 않는다) 드라이버는 256 Rx 기술자 및256 Tx기술자가 기본값이다. 이것은 칩셋의 초기 버젼만이 이것을 지원하기 때문이다. 모든 최근 버전은4096이지만 드라이버가 이를 자동으로 감지하지 않는다. 기술자의 수를 늘리면 일부 호스트에 극적으로 성능을 향상시킬 수 있다.
이것은 인텔 10GE NIC에서도 같다 – 하드웨어는 4096까지 지원하지만 드라이브는 Rx, Tx 모두에 대해 256 기술자를 기본으로 하다.

o 시스템의 NIC의 기술자 수를 확인하기 위해 ethtool을 사용할 수 있으며, 드라이버가 그것들을 사용하도록 구성되어 있는지 여부를 확인할 수 있다. 기본 설정을 사용하는 인텔10GE NIC의 일부 샘플 리눅스 출력은 다음과 같다:

```
[user@perfsonar ~]# ethtool -g eth2
 Ringparametersforeth2:
 Pre-setmaximums:
 RX:        4096
 RXMini:     0
 RXJumbo:   0
 TX:       4096
```

```
Currenthardwaresettings:
RX:        256
RXMini:   0
RXJumbo:   0
TX:        256
```

리눅스 환경에서 사용되는 드라이버를 확인하고 싶다면 다음을 수행한다:

```
ethtool -i eth0
```

o 만일 e1000(1GE)나 ixbge(10GE)를 사용하고 있고 하드웨어가 현재 사용하고 있는 기술자보다 더 많은 기술자를 지원한다면 추가적인 기술자를 더 사용하도록 구성을 변경할 수 있다. 리눅스에서 다음을 /etc/modprobe.conf 에 추가한다.(e1000 가정).

```
alias eth0 e1000
options e1000 RxDescriptors=4096,4096
TxDescriptors=4096,4096
```

같은 결과를 얻도록 다음을 /etc/rc.local 에 추가한다.

```
ethtool -G ethN rx 4096 tx 4096
```

이것이 작동한다는 것을 검증하기 위해 다음을 루트에서 실행한다.

```
ethtool -g eth0
```

FreeBSD에서 다음을 /boot/loader.conf에 추가한다.

```
hw.em.rxd = 4096
hw.em.txd = 4096
```

다음 재부팅한다. 이 설정이 제대로 작동하는지 확인하기 위해 다음을 루트에서 실행한다.

```
sysctl dev.em.0.debug=1
```

그리고, /var/log/messages를 살펴보라.

Linux 용 Myricom 10 Gig NIC 튜닝

o Myricom NIC은 많은 튜닝 기술을 제공한다. 특히 인터럽트 합체(coalenscing)을 설정하면 처리량에 상당한 도움이 된다:

```
/usr/sbin/ethtool -C ethN rx-usecs 75
```

o 특히, Mryicom의 "Throttle" 옵션을 사용하여 방화벽 환경에서 매우 극적인 개선을 보았다. 우리는 또한Myricom에서Redhat/CentOS배포판의 기본 드라이버를 사용하는 것 대신 최신 드라이버를 다운로드하고 소스를 컴파일해서 사용하여 25%까지 성능개선이 된다.

CentOS7 문제

o CentOS 7.1은 Myricom의MSI-X를 기본값으로 설정하여 사용하지 않아서 성능 저하를 가져오는 것으로 보고되었다. 이 문제를 해결하려면 다음처럼 수행한다:
다음 사항을 포함한 파일 만들기:

```
options myri10ge
myri10ge_fw_name=myri10ge_rss_eth_z8e.dat
myri10ge_gro=0 myri10ge_max_slices=-1
```

Cheliso 10Gig NIC 리눅스와 FreeBSD

o Chelsio NIC의 TCP 분할 부하제거 (TSO, TCP Segmentation Offload) 및 TCP 부하제거 엔진(TOE, TCP Offload Engine) 모두는 WAN상에 근본적으로 성능을 손상시킨다(LAN상의 처리량에 영향을 주지 않고CPU의 부하를 줄이는 데 도움을 준다) 것으로 보고 되었다. 이 경우 매우 빠른CPU에서 특히 그렇다.
TSO를 끄려면:

```
ethtool -K interface tso off
```

Chelsio는 리눅스에서 다양한 설정 조정이 가능한 'perftune.sh'라는 스크립트를 제공한다.

2.4. 패킷 페이싱(Packet pacing)

o 빠른 호스트에서 느린 호스트로 전송할 때, TCP backing off와 패킷 손실을 야기하면서 수신자를 overrun하는 것은 쉽다. 10G 호스트가 10G 이하의 가상 회선으로 데이터를 전송하거나 또는40G 호스트가10G 호스트로 전송, 또는 빠른CPU 송신자를 가진40G/100G 호스트가 더 늦은 CPU를 가진 40G/100G 호스트로 데이

터를 전송할 때 유사한 문제들이 발생한다. GridFTP와 같은 병렬 스트림을 사용하는 도구를 사용하는 경우 이러한 문제는 더욱 뚜렷하다. 일부 긴 경로(50-80ms RTT)에서 우리는 패킷 페이싱을 활성화 한 후2-4X의TCP 성능 향상을 보았다.[16]

FQ(공정 큐잉) 스케줄러를 사용하는 패킷 페이싱

o 리눅스 커널3.11 (페도라20, 데비안8, 우분투13.10에서 사용 가능) 이상부터 빠른 호스트에서도 패킷 패이싱을 잘하는 코드를 포함한 새로운 "공정큐잉" 스케줄러가 있다.[17]

o RHEL 기반의OS들에 대해 FQ는 V7.2에서 3.10.0-327 커널에 백 포트되었다.
기본적으로 꺼져있는 공정큐잉을 활성화 하려면 아래와 같이 수행한다:

```
tc qdisc add dev $ETH root fq
```

또는/etc/sysctl.conf에 다음을 추가한다:

```
net.core.default_qdisc = fq
```

대역폭을 형성하고 패이싱하기 위해 다음을 수행한다:

```
tc qdisc add dev $ETH root fq maxrate Ngbit
```

o 모든 호스트에서FQ를 켜기를 추천한다. GridFTP를 실행하고4 병렬 스트림을 기본을 설정하여 사용하는10G 데이터 전송 노드(DTN)에 대하여 FQ 설정을 다음과 같이 할 것을 추천한다:

```
tc qdisc add dev $ETH root fq maxrate 2.5gbit
```

o 또한SO_MAX_PACING_RATE 옵션으로'setsockopt' 시스템 호출을 사용하여 응용 프로그램에FQ 기반 페이싱을 추가 할 수 있다. iperf3 도구는 대역폭 옵션을 위해 이것을 사용한다.

o FQ를 사용하는 경우에는 대부분의NIC에서 기본적으로 꺼져있는TSO를 비활성화로 설정해야 한다.

2.5. 40G/100G 튜닝(40G Tuning)

o 40G/100G 이더넷NIC의 호스트들에 대해서 처리량을 최대화하기 위해 조정하기를 원하는 추가적인 몇 가지가 있다.
조정할 가장 중요한 것들은:

- 성능을 위한 CPU 관리자
- 최대 2GB로 설정된 TCP 버퍼 크기
- IRQ 및 사용자 프로세스에 대해 올바른 코어를 사용하고 있다는 확인
- 여러분의 환경에 좋은 페이싱율을 가진 공정큐잉(FQ)의 활성화

o 만일 단일 플로우의 처리량에 관심이 있고, 더 높은 CPU 클럭속도가 중요하다면 CPU의 클럭속도는 여전히 40G/100G 플로우에 대해서 상당한 문제이다. 일반적으로 플로우마다 40Gbps를 달성하기 위해 적어도3GHz의CPU의 클럭속도가 필요하다.[18]

3. Network tuning

3.1. TCP 성능 향상을 위한 튜닝

o TCP의 성능에 영향을 미치는 여러 가지 이유가 많지만, 크게 세 가지 이유는 패킷 손실, 지연(또는RTT – Round Trip Time), 및/ 윈도우 버퍼 사이즈이다. 세 가지가 상호 연관 있다.

o 지연/ RTT는 수신자에서 발신자로 패킷이 이동하는 데 걸리는 시간과 다시 돌아오는데 걸기는 시간이다. 이것은 하나의 호스트가 데이터를 보내고, 보내진 데이터에 관한 정보가 다른 호스트로부터 돌아오는데 걸리는 최소 시간이다.

o 버퍼/윈도우사이즈는 커널이 접속을 위해 버퍼에 유지하는 데이터의 양을 결정하고 또한 TCP 연결(TCP를 통해 전송 된 윈도우 정보는 가용 버퍼의 크기를 반영)를 통해 광고하는 윈도우를 결정한다. 윈도우가 클수록 두 개의 호스트 사이에 더 많은 데이터를 전송할 수 있다. 주목할 만한 사항은 윈도우가 대역폭 지연곱(the Bandwidth Delay Product)으로 알려진 지연에 의해 배가된 가용 대역폭보다 작다면 그 송신자 윈도우 전체 크기의 데이터를 보내게 될 것이고, 그리고, 수신자가 그 데이터를 확인할 때까지 앉아서 기다리게 될 것이다. 이는 성능 저하를 초래하게 되고, TCP 버퍼 자동튜닝이 필요한 주요원인이 된다. – 자동튜닝이 없다면, 시스템전역 인자들을 각 목적지에 맞게 최적화해야 하거나 응용프로그램이 그 버퍼의 설정을 정확하게 하기 위해 하부 네트워크에 대해 알고 있어야 한다. 버퍼의 크기를 너무 크지 않게, 그러나 충분히 큰 최대 버퍼사이즈를 지원하는 자동튜닝이 존재한다면 커널은 각 연결을 위한 지연에 필요한 TCP 버퍼/윈도우사이즈를 정확히 계산할 것이다.

o 세 번째 요인은 패킷 손실이다 – 복잡하게 얽힌 부분이다.
TCP는 패킷 손실이 발생하면, 그 전송 속도를 줄인다. 이것을 실행하기 위한 메커니즘은 송신자의 윈도우를 줄이자는 개념인데 그 결과 송신하는 데이터를 줄이려고 시도하는 것이다. 그런 다음 TCP는 손실이 일시적일 거란 희망을 가지고 송신률을 다시 증가시킨다. htcp와 큐빅(cubic)과 같은 알고리즘이 예전 방식인 TCP Reno와 같은 안보다 훨씬 공격적으로 속도를 다시 증가시키는 안을 사용하여 수년전보다 현재 일이 훨씬 괜찮아 진 경우도 있다.

o TCP는 손실이 발생하면 복구해야 한다 하지만 시작은 작은 윈도우로 하고 시간이 지남에 따라 윈도우 크기를 조금씩 키운다. 지연이 길어질수록 이 동작을 수행하는 제어루프도 길어진다. 그래서 모든 것이 동등해지면 손실로부터 복구하기 위해 TCP 연결을 위해 필요한 시간이 증가하고, 따라서 RTT(Round Trip Time)도 증가한다.

o 그러나, 라우터, 스위치, 방화벽에 작은 버퍼를 추가할 때 손실과 지연 사이의 상호 작용을 복잡하게 하는 추가적인 문제가 있다. 대기 시간이 큰 경우 충분한 처리량을 위해 큰 윈도우가 요구된다. (대기 시간이 약10 밀리 초 이상이 되면 문제가 되기 시작하고, 20 밀리초 이상은 정말 까다로운 경우가 된다.) 윈도우가 큰 경우, TCP는 한번에 많은 양의 데이터를 전송할 수 있다. 네트워크 카드는 일반적으로TCP에 대해 모르거나 상관하지 않는다. 네트워크 카드는 전송할 패킷이 있을 때 그것을 전송할 패킷이 없을 때까지 링크 상으로 패킷을 넘긴다. 이것은 네트워크 카드가 예를 들어10Gbps처럼 어떤 링크 속도에 맞추어져 있어도 발생한다.

o 그래서 만약TCP 윈도우 크기가 4MB라면 TCP는 4MB의 데이터를 네트워크 카드로 보낼 것이고, 네트워크 카드는10Gbps의 링크 상으로 데이터를 보내게 된다. 이것은 송신기와 수신기 사이의 경로에 있는 모든 장치가 고속 패킷 버스트를 보는 것을 의미한다. 만일 전송 경로상에 버퍼링 또는 큐잉 문제가 있다면TCP가 전송속도를 낮추어 그 패킷의 일부는 손실될 것이다.

o 짧은 대기 시간이 있는 연결의 경우 TCP 윈도우가 훨씬 작기 때문에, 이러한 문제는 종종 대기시간이 짧은 경우 발견되지 않는다. 패킷 손실이 임의의 오류(예: 오염된 광섬유, 한계 광학, 스펙보다 긴 길이의 케이블)에 의해 발생된 것이어서 오류가 여기저기서 발생하고 대기시간도 짧다면, TCP는 아주 빠르게 오류를 복구하여 일반적으로 사람들이 성능 저하가 있었는지 확인할 수 없을 것이다.

o 만약에 전송 경로 상에 작은 스위치 버퍼가 있다면 TCP가 손실을 유발할 만한

정도의 크기를 가지는 버스트를 보내지 않을 것이기 때문에 일반적으로 대기시간이 짧은 연결의 경우 손실이 발생하지 않는다.

o 그러나 만일 아주 먼 호스트로 데이터를 보내기 위해 같은 인프라를 사용한다면 성능도 급격하게 저하될 것이다. 무작위 에러의 경우 TCP는 항상 복구 모드일 것이고, 그래서 항상 작은 윈도우를 유지할 것이다. 버퍼의 크기가 작은 경우에 TCP는 전송률을 높이고, 다시 패킷 손실을 발생시키고, 윈도우 크기를 줄이고, 다시 그렇게 전송율을 높이는 과정을 반복한다. 그래서 TCP는 항상 작은 윈도우 크기를 유지하고, 성능의 저하를 발생시킬 것이다.

패킷손실(Packet Loss)

o 비록 매우 작은 양의 패킷 손실이라 할지라도 성능에 큰 영향을 미칠 수 있다. ESnet에서 최근에 발생한 고장난 라우터 라인 카드의 예를 살펴보자. 패킷 필터 카운터를 사용하여, 우리는 한 방향으로0.0046 %의 패킷, 혹은22,000 패킷 중에 중 평균1 패킷이 손실되었음을 확인하였다.

o 모든 호스트는 10G NIC 에 접속되고, 어떤 경로도 혼잡하지 않았다. MTU는 9000bytes로 설정하였다. 참고로 기본 1500 MTU일 때 패킷 손실의 영향이 더 크다. 우리는 기존의 TCP Reno, Hamilton TCP (htcp), 및 Mathis 방정식에 의해 예측 처리량을 비교한다. Mathis 방정식은 TCP Reno에 대해서 최대 처리량이 다음 식으로 결정된다.

$$\frac{\text{maximum segment size}}{\text{round trip time}} \times \frac{1}{\sqrt{\text{packet loss rate}}}$$

그림 10 Mathis Equation

다음 그림에서 주석은 대기시간이 길어짐에 따라 손실이 성능에 미치는 영향을 보여준다.

o 아래 그림은 하단부분의 확대 버전이다. 아래 그림에서는 htcp가 Reno보다 약 두 배 정도 빠르다(cubic의 결과와 매우 유사하다.).

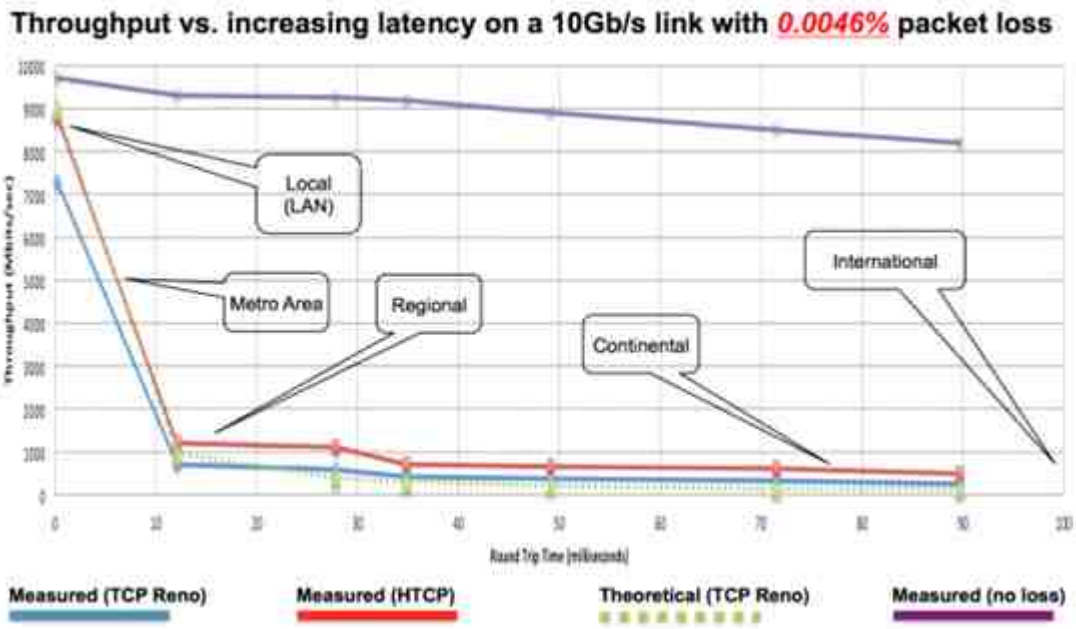

그림 11 성능과 대기시간의 관계

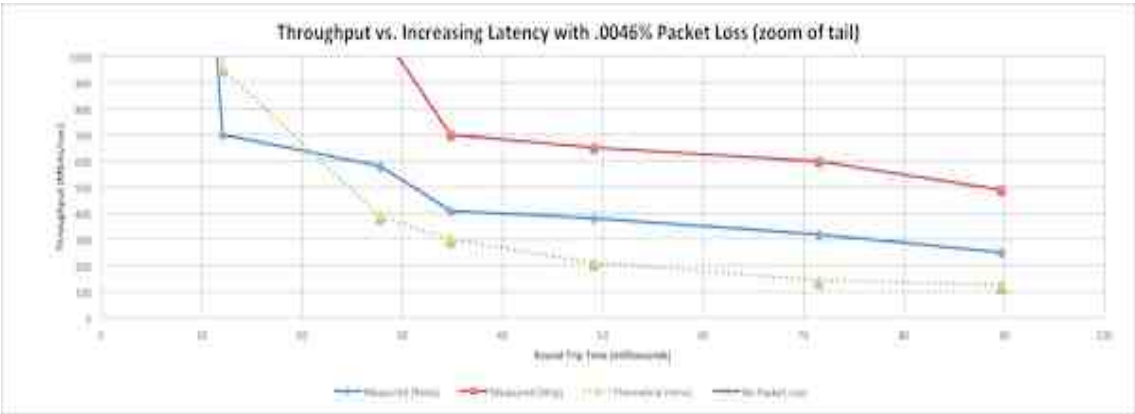

그림 12 성능과 대기시간과의 관계 – 정밀 버전

0.0046 % 손실의 성능에 미치는 영향은 대기시간이 긴 경로에서는 매우 크다. 90ms의 경로에패킷손실이 없는 단방향 전송에서 8.2Gbps 대490Mbps를 획득했다. 이것은 거의 17배 더 느린 것이다.

TCP 순서변경(TCP ordering)

o 전송 제어 프로토콜(TCP)은 종단간(end-to-end) 전송 된 패킷의 동작에 매우 민감하다. 도착 시간의 변화(jitter, 지연변이)는 순서를 변경하거나 완벽한 데이터 손실, 데이터의 일부를 바꾸는 등의 비정상적인 이벤트와 결합되면 복구하기 어려운 문제를 유발한다.

o 이러한 문제는 관찰되는 처리량이 낮아지는 형식으로 말단 사용자에게 전달된다. 사건은 여전히 종단간에 발생하지만 그런 교란이 전혀 발생하지 않았을 때 보다 더 느려질 것이다. 통상적으로 경로상에 존재하는 패킷의 재순서화의 잠정적인 모든 가능성을 제거하려 시도해야 한다. 통상적으로 비춰봤을 때 낮은 수준의 순서 재변경의 경우에도, 누락 된 패킷이 밀리초 내에 전달이 되고 그 결과로 성능에 상당한 악화가 발생할 수 있음을 보여 주었다.

o 이 페이지에 첨부 된 문서는 netem 도구가 지리적으로 분리된 두 개의 호스트 사이에서 사용되고, 다양한 수준의 통제 불능상태와 그리고 지연이 iperf3 테스트에 적용된다. 요약하자면:

- 두 가지 요인이 발생할 때(예를 들면, OOP(비정렬 패킷, Out of Order Packets)의 발생 가능성, 그것이 도착할 수 있는 것(지연변이, jitter)) OOP는 TCP 스트림에 해를 끼치게 된다.
- 송수신 호스트의 TCP 조정되었고, 호스트가 충분한 메모리와 빠른 처리기를 가졌다는 것만 보장되면 이것은 낮은 수준에서 복구 가능하다.
- 패킷의 비정렬 가능성이 증가하거나 지연이 허용 한도를 넘어 증가하는 경우에는 복구 불가하다.
- IOP(정렬 패킷, In Order Packet)과 OOP(비정렬 패킷) 사이의 시간이 증가할 때 이것은SACK 동작을 시작시킨다. 이것은 하나의 패킷을 다시 보내도록 시도하는 예방 조치이다. 이것은 윈도우에서 백업을 유발하고, 데이터의 전체 부하의 재전송을 야기 할 수 있다.
- 만일 원인이 될 지연이 일어날 수 밖에 없다면 일반적으로OOP 기회를 최소화해야 한다. 일반적으로 다음의 실제 상황은 이런 것이 발생할 원인을 제공할 수 있다.
- 단일TCP 스트림을 처리하는 과정을 병렬화할 수 있는 장치의 탐색(방화벽, 패킷쉐이퍼(packet shaper))
- 해싱알고리즘이 한 플로우에서 모든 패킷에 대해 동일한 경로를 사용하지 않을 때, 동등한 것을 보여지는 링크들 사이의 트래픽을 분할하는 장치의 탐색(예: 호스트 상의 결속링크, 네트워크 장치 상의 LAG)

3.2. UDP 튜닝(UDP Tuning)

o UDP는 약간의 튜닝도 없이는 최대 10Gbps혹은 그 이상의 속도를 내지 못한다. 중요한 요인은 다음과 같다:

- 점보 프레임의 사용: 성능이9K MTU를 사용할 때보다 4 ~ 5 배 더 향상될 것이다.

- 패킷 크기: MTU 크기- 패킷 헤더의 크기 일 때 최고의 성능을 낸다. 예를 들면, 9000Byte MTU의 경우, IPV4는 8972를 사용하고, IPv6는 8952를 사용
- 소켓 버퍼 크기: 버퍼 크기가TCP의 경우 RTT와 관련 있지만, UDP의 경우는 무관하다. 그러나 기본값으로 설정된 크기는 아직 충분히 크지 않을 것이다. 소켓 버퍼의 크기를 4M로 설정하는 것이 대부분의 경우에 많은 도움이 될 것으로 보인다
- 코어 선택: 10G에서UDP는 일반적으로CPU의 제약을 받는다. 그래서 올바른 코어를 선택하는 것이 중요하다. 이것은 Sandy/Ivy Bridge 메인 보드에 경우 특히 그러하다.

```
iperf , iperf3 및 nuttnuttcp의 명령어 예 :
nuttcp -l8972 -T30 -u -w4m -Ru -i1 -xc4/4 remotehost
iperf3 -l8972 -T30 -u -w4m -b0 -A 4,4  -c remotehost
numactl -C 4 iperf -l8972 -T30 -u -w4m -b10G –c remotehost
```

o 독자의 호스트를 위한 최적의 것을 찾기 위해 다른 종류의 코어를 실험해봐야 할 수도 있다. NIC 인터럽트 처리를 위해 어떤 코어가 사용되고 있는지 알아내기 위해 'mpstat -P ALL 1' 명령어를 사용할 수 있다. 그리고, 같은 코어가 아니라 같은 소켓에서 코어를 시도해볼 수 있다.

o 일반적으로 nuttcp가 UDP 에서 가장 빠른 것 같다. 만일 "-xc"옵션을 위해서라면 nuttcp V7.1 이상이 필요할 것이다. 이 튜닝으로는 최대10Gbps를 얻기 위해 빠른 코어가 필요하다. 예를 들어, 2.9GHz 인텔 Xeon CPU는 최대10Gbps 회선 속도를 얻을 수 있지만, 2.5GHz의 인텔 Xeon CPU로는 단지5.9Gbps의 속도만 볼 것이다. 2.9GHz의CPU는40G NIC를 사용하여UDP의22 Gbps의 속도를 낼 수 있다.

o 프로세서 아키텍처는 클록 속도를 희생하여 총 용량(예를 들면 더 많은 코어)을 용이하게 하도록 설계되고 있다. 몇 가지 사용 사례에 대해서 수확체감을 야기할 지라도클럭속도를 높게 설정하면 문제가 되었었다. 많은 새로운 장비는 더 많은 코어를 제공하고, 이것은 일반적으로 가상머신(VM)과 대다수의 소규모 네트워크 플로우에 대해서 아주 훌륭히 작동한다. 단일 스트림 성능 테스트는 적은 코어, 높은 클럭속도의 혜택을 볼 수 있는 하나의 사용 사례이다. 만일 각 코어당 2배의 UDP 플로우를 가지고 얼마 만큼 얻을 수 있는지 확인하고 싶다면 다음과 같이 실행해 볼 수 있다:

```
nuttcp -i1 -xc 2/2 -Is1 -u -Ru -l8972 -w4m -p 5500 REMOTEHOST & \
nuttcp -i1 -xc 3/3 -Is2 -u -Ru -l8972 -w4m -p 5501 REMOTEHOST & \
```

CPU 제한 결정

o 위의 명령을 실행하고 있지만 여전히 사용 뛰어난 성능 표시되지 않는 경우 'mpstat -P ALL 1' 을 사용해서 얼마나 많은CPU를 사용할 것인지를 결정한다. 예를 들어, 아래 그림은 제시된 명령 행 옵션을 사용하여nuttcp 시험이며, 그 결과는5.9 Gbps이다:

```
[root@jz@llnl-pt1 ~]# nuttcp -18972 -T10 -u -w4m -Ru -i1 -xc6/6 -p 8888 -P 8889 nersc-pt1.es.net
  698.6701 MB /   1.00 sec = 5860.7579 Mbps     0 / 81655 ~drop/pkt  0.00 ~%loss
  698.2508 MB /   1.00 sec = 5857.3054 Mbps     0 / 81606 ~drop/pkt  0.00 ~%loss
  698.6273 MB /   1.00 sec = 5860.5749 Mbps     0 / 81650 ~drop/pkt  0.00 ~%loss
  699.1834 MB /   1.00 sec = 5865.1700 Mbps     0 / 81715 ~drop/pkt  0.00 ~%loss
  698.7214 MB /   1.00 sec = 5861.2882 Mbps     0 / 81661 ~drop/pkt  0.00 ~%loss
  698.8497 MB /   1.00 sec = 5862.3942 Mbps     0 / 81676 ~drop/pkt  0.00 ~%loss
  697.6519 MB /   1.00 sec = 5851.8364 Mbps     0 / 81536 ~drop/pkt  0.00 ~%loss
  698.6358 MB /   1.00 sec = 5861.0686 Mbps     0 / 81651 ~drop/pkt  0.00 ~%loss
  698.1053 MB /   1.00 sec = 5856.1321 Mbps     0 / 81589 ~drop/pkt  0.00 ~%loss
  699.1064 MB /   1.00 sec = 5864.5357 Mbps     0 / 81706 ~drop/pkt  0.00 ~%loss

 6985.8878 MB /  10.00 sec = 5860.1019 Mbps 99 %TX 47 %RX 0 / 816455 drop/pkt 0.00 %loss
```

그림 13 nuttcp 결과

o Nuttcp는 송신 호스트에서99 %의CPU를 보여주고, 수신 호스트에서mpstat는 6개의 코어가 포화되지 않았다는 것을 확인해준다:

```
03:02:41 PM  CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest   %idle
03:02:42 PM  all    0.27    0.00   18.22    0.00    0.00    1.64    0.00    0.00   79.86
03:02:42 PM    0    0.00    0.00   17.58    0.00    0.00    2.20    0.00    0.00   80.22
03:02:42 PM    1    1.08    0.00   21.51    0.00    0.00    1.08    0.00    0.00   76.34
03:02:42 PM    2    0.00    0.00   14.29    0.00    0.00    1.10    0.00    0.00   84.62
03:02:42 PM    3    0.00    0.00    6.59    0.00    0.00    0.00    0.00    0.00   93.41
03:02:42 PM    4    0.00    0.00    3.30    0.00    0.00    1.10    0.00    0.00   95.60
03:02:42 PM    5    0.00    0.00   13.33    0.00    0.00    1.11    0.00    0.00   85.56
03:02:42 PM    6    1.04    0.00   61.46    0.00    0.00    6.25    0.00    0.00   31.25
03:02:42 PM    7    0.00    0.00    6.74    0.00    0.00    0.00    0.00    0.00   93.26
```

그림 14 CPU 코어 확인1

전송 호스트에서mpstat은 6개의 코어가 포화되었다는 것을 확인해준다:

```
03:02:35 PM  CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest   %idle
03:02:36 PM  all    0.38    0.00   12.55    0.00    0.00    0.25    0.00    0.00   86.82
03:02:36 PM    0    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00  100.00
03:02:36 PM    1    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00  100.00
03:02:36 PM    2    0.00    0.00    1.02    0.00    0.00    0.00    0.00    0.00   98.98
03:02:36 PM    3    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00  100.00
03:02:36 PM    4    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00  100.00
03:02:36 PM    5    1.01    0.00    0.00    0.00    0.00    1.01    0.00    0.00   97.98
03:02:36 PM    6    1.98    0.00   97.03    0.00    0.00    0.99    0.00    0.00    0.00
03:02:36 PM    7    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00  100.00
```

그림 15 CPU 코어 확인2

이런 호스트들에 대해서 다른 코아들에서 다중nuttcp 클라이언트를 실행하는 것이 총 처리량을 증가시킬 것이다.

MTU 문제

o 점보 이더넷 프레임은 예전 하드웨어 상에서 10G 경로의2-4 배의 성능을 높일 수 있다. 심지어 현대 하드웨어에서도UDP에 대해 9.9Gbps 대5Gbps, TCP에 대해 9.9Gbps 대9.3Gbps의 차를볼수있다. 40G의 경로에 대해서는 성능 향상도 더 크다. TCP의 단일 스트림 테스트는25Gbps 대 10Gbps로 관찰되었다. UDP 테스트는15Gbps 대8Gbps의 향상을 보여 주었다.

o 점보 프레임의 사용은 두 가지 장점이 있다: 하나의 장점은 주어진 데이터 처리에 대해, 패킷 처리를 위한 낮은CPU를 필요로 하는 점보 프레임 대 표준 프레임의 패킷률(packet rate)이 적다는 점이다. 다른 장점은 패킷 손실이 발생한 후에 데이터 처리량의 회복이 최대 세그먼트 크기에 비례한다는 것이다. - 그래서 점보 프레임을 사용하면 손실 이벤트에서6 배까지 빠른 회복 속도를 얻을 수 있다. MTU 크기를 확인하기 위해 사용될 수 있다. 예를 들어, 리눅스에서 다음과 같이 확인할 수 있다:

```
ping -s 8972 -M do -c 4 10.200.200.12
```

o Scamper 나 tracepath는 경로MTU 크기를 확인하는 좋은 도구이다.
점보 프레임의 몇 가지 단점도 있다. 싱글브로드캐스트 도메인에 있는 모든 호스트는 반드시 같은 크기의 MTU로 구성되어야 한다. 이것이 어렵고 에러를 발생시키는 경향이 있을 수 있다. 이더넷은MTU 불일치를 검출 할 방법이 없다- 이것은 제대로 작동하려면ICMP 시그널링을 필요로 하는 layer 3 함수이다. (불행하게도 일부 사이트는ICMP를 차단하고, 경로 MTU 검색을막는다 실패했을 때 발생하는 것과 같다.).

o 따라서 좋은 방법은 고속 데이터 전송을 위한 새로운 호스트 점보 프레임 활성화 서브넷을 생성하는 것이다. 점보 프레임을 사용하여 리눅스 호스트에서 MTU 블랙홀 문제를 피할 수 있도록tcp_mtu_probing = 1을 설정할 것을 추천한다. 때때로 2로 설정하면 성능 문제를 일으킨다.[20]

4. Data Transfer Tool

4.1. 배경

o 보편적으로 광역 대량 데이터 전송은 여러 가지 이유로 성능 저하에 시달려왔다. 성능저하의 요인으로는 부적절한 송수신 호스트 구성, 소프트웨어 설계 문제, 방

화벽, 및 기타 요인이 있다. 그러나 대부분의 경우, 대량의 데이터 세트는 현재 구성된 네트워크를 이용하여 긴 거리를 전송시킬 수 있다.

o 파일 전송 속도를 높일 수 있는 방법 중에 하나는 한 파일을 병렬 전송이 가능한 작은 조각으로 나누는 것이다. 다양한 수단을 통해서 병렬 전송을 할 수 있다. 예를 들어 만일 복사해야 할 파일이 많다면, 한 번에 여러 파일을 복사하여 병렬 전송을 할 수 있다. 그러나 일반적으로 작은 파일보다 큰 파일을 복사해서 병렬 전송하는 것이 효율적이므로 작은 크기의 파일 여러 개를 tar나 zip으롤 압축해서 크게 만든 다음 전송하는 것이 좋다.

o 파일 전송 도구를 선택할 때, 우선 결정해야 할 사항은 아래 옵션들 중에 요구되는 보안 모델을 선택하는 것이다. 익명(anonymous): (예: FTP, HTTP) 아무나 데이터에 액세스 할 수 있다.간단한 패스워드(simple password): (예: FTP, HTTP) 암호를 쉽게 캡처 할 수 있기 때문에 대부분의 사이트가 더 이상이 방법을 허용하지 않는다. 비밀번호 암호화(password encrypted): (예: bbcp, bbftp, Globus Online/ GridFTP, FDT) 제어 채널은 암호화되지만 데이터는 암호화되지 않는다. 모두 암호화(everything encrypted): (예: scp, sftp, ssh상의 rsync, GridFTP, HTTPS기반의 웹 서버) 제어채널과 데이터 채널 모두 암호화된다

o 일반적으로 대부분의 공개 과학 프로젝트들(open science projects)은 세 번째 옵션을 선호하는 것 같다. 만일WAN상에서 네 번째 옵션을 사용해야 한다면, WAN상에서 좋은 성능을 발휘하는 X509 키를 가진 GridFTP와 HPN-패치SSH 툴의사용으로국한될것이다. (예: HPN-패치 scp/sftp, HPN-패치 ssh상의 rsync). 다른 이슈는 서버를 설치할 수 있는 능력이나 요구가 있느냐의 여부이다. HTTP나FTP 기반 도구는 웹 또는FTP 서버를 설치할 수 있는 시스템 관리자를 필요로 한다. bbcp 과 GridFTP와 같은 도구는 단지 관리자가 sshd 서버만 관리자에 의해 설치되도록 제한하고 다른 모든 것들은 일반 사용자에 의해 설치될 수 있다. 소스와 목표 위치와 같은 엔드포인트만 시스템 내에서 알려져 있다면, Globus Online만이 어떤 소프트웨어의 설치도 사용자가 할 필요가 없는 유일한 도구입니다

o 고속WAN을 통해 최대 처리량을 얻기 위해 필요한 한 가지는 병렬 데이터 스트림에 대한 지원을 포함하는 파일 전송 도구를 사용하는 것이다. 현재 아주 많은 수의 파일 전송 프로그램이 있지만 불행하게도 그것들 중 어느 것도 이러한 기능을 모두 제공하는 것은 없다. 이런 특징과 지원의 최상의 조합을 갖춘 Globus Online 이나ssh모드의 GridFTP를 추천한다.

4.2. Globus

개요

o Globus는 GridFTP 서버간에 또는GridFTP 서버와 사용자의 컴퓨터(윈도우, 맥 또는 리눅스)간에 데이터를 이동 할 수 있는 빠르고, 신뢰할 수 있는 파일 전송 서비스이다.

o Globus는 성능 모니터링, 실패한 전송의 재개, 가능할때마다 자동적으로 오류의 복구, 상태보고와 같은 파일 전송을 관리하는 활동을 자동화하고 있다. 예를 들면, 50MB보다 작은 파일들에 대해서는 2개의 스트림, 50MB에서 250MB사이의 파일들에 대해서는 4개의 스트림, 250MB이상의 파일들에 대해서는 8개의 병렬 스트림을 사용하는 것처럼 시작된 병렬 스트림들의 수가 전송되고 있는 파일의 크기에 의존한다. 작업을 위해 필요한 어떤 종류의 사용자 정의 인프라도 없다- Globus 서비스로서의 소프트웨어(software-as-a-service)로 사용자가 어떤 기능도 구축하지 않고 바로 사용할 수 있다.

o 다음 설명은 Globus 접근을 위해 설정된 호스트들 사이에서 파일을 전송하기를 원하는 사용자를 위한 빠른 실행 가이드이다.

o Globus 사용자 이름과 패스워드를 이용하여 Globus 에 가입(Sign in) 상단 바의 드롭 다운 메뉴를 이용하거나 'FileTransfer'에서 'Start Transfer'를 선택

o 'Start Transfer'페이지에서'Endpoints' 드롭 다운 상자에 있는 버튼을 클릭하여 사용 가능한 엔드포인트의 목록을 볼 수 있다. 사이트 이름(NERSC, ALCF, ESnet 등)을 입력하면 해당 사이트에 대한 Globus 엔드포인트의 목록이 나타난다.

o 예를들어 NERSC 엔드포인트를 선택했을 때 로그인 창이 나타난다. 단순히 NERSC 사용자 이름과 암호를 사용하여 Globus의NERSC 엔드포인트에 액세스 할 수 있다. 'Username' 필드에 NERSC 사용자이름을 입력하고 'Passphrase'필드에 SSH 암호가 아닌 NERSC 암호를 입력하고, 'Authenticate'를 클릭한다. 다른 필드는 무시해도 된다.

o NERSC에서 홈 디렉토리의 내용의 목록을 볼 수 있다. 디렉토리의 내용을 보고싶다면 더블 클릭으로 가능하다. 파일이나 디렉토리를 선택하고 '화살표 버튼'을 클릭하면 전송이 시작된다.

o ALCF BlueGene/P등의 포함한 다른 엔드포인트들에 대해서도 절차는 동일하다. ESNet DTNs (Globus에서'esnet'을 입력하면 네 개의 서버가 나타날 것이다.)는

익명GridFTP 서버이다. 그래서 액세스하기 위해 사용자 이름과 암호를 입력 할 필요가 없다.

o 전송프로토콜로 TCP대신에 UDT를 사용하기 위해서는 'Label this Transfer' 상자에'useudtplease'를 입력한다.

4.3. GridFTP

o 고속WAN을 통해 대용량 파일의 전송을 위해GridFTP 사용을 추천한다. 엔드포인트 중 하나 또는 둘 모두 일반GridFTP 서버 설정인 경우는 Globus Online의 사용을 검토해야 한다. 여기서는 SSH의 지원만 있는 즉, X509 지원이 없는 GridFTP를 설치하기 위한 '빠른 시작 가이드'이다. 클라이언트와 서버 호스트 모두에서 다음 단계를 수행한다.

o 먼저 Globus repo를 설치:

```
 rpm -hUv
http://www.globus.org/ftppub/gt6/installers/repo/globus-toolkit-repo
-latest.noarch.rpm
```

그런 다음 'yum install' 실행 :

```
yum install yum-plugin-priorities  # globus installer needs this
yum install globus-data-management-client globus-data-management-server
```

다음 명령을 사용하여sshftp 를 사용가능하도록 설정 :

```
/etc/init.d/globus-gridftp-sshftp reconfigure
```

o 이제 GridFTP 서버는 자동으로 sshd를 통해 시작된다. 소스와 목적지 호스트 모두에 ssh를 할 수 있어야 한다. 다음은 몇 가지 샘플 명령들인데 이 명령들을 수행할 때 root가 아니어야 한다.

```
# directory listing
 globus-url-copy -list sshftp://gridhost.foo.gov/tmp/
# copy file /etc/group
 globus-url-copy sshftp://gridhost.foo.gov/etc/group file:/tmp/group
# parallel transfer of file /tmp/mybigdatafile
 globus-url-copy -p 4 sshftp://gridhost.foo.gov/tmp/mybigdatafile \
     file:/tmp/myfile
# test network throughput
 globus-url-copy -vb -p 4 sshftp://gridhost.foo.gov/dev/zero
```

```
file:///dev/null
```

o GridFTP를 설치했다면 사용을 테스트 할 수 있습니다 GridFTP test Data Transfer Nodes(https://fasterdata.es.net/performance-testing/DTNs/)를 사용하여 시험해 볼 수 있다.

서버 로깅 옵션

o 성능 분석 및 문제 해결을 훨씬 쉽게 만들어줄 GridFTP의 추가 로깅 기능을 활성화하는 것을 추천한다. 더 자세한 로그는 가장 빠르고 느린 엔드포인트를 식별하고, 원격 사용자가 요청하는 TCP 버퍼의 크기와 병렬 스트림의 개수와 같은 설정을 확인하기 위해 사용할 수 있다.

o 이 추가 로깅을 사용하려면 /etc/gridftp.conf에 다음을 추가 :

```
log_level ERROR,WARN,INFO
log_single /var/log/gridFTP/gridftp-auth.log
log_transfer /var/log/gridFTP/gridftp.log
log_module stdio_ng
```

로그 디렉토리는 모든 GridFTP 사용자에 의해 쓰기 가능해야 한다.

방화벽 이슈

o 대부분의 사이트가 GridFTP의 동작을 방지하는 방화벽을 실행한다. 동적으로 할당 된 포트를 사용하는 FTP와 같은 프로토콜은 종종 방화벽에 의해 차단된다. 흔히 방화벽은 들어오는 연결은 블록하고 나가는 연결을 차단하지 않도록 구성된다. 이와 같은 경우에는 방화벽이 있는 사이트 내부에서 전송을 시작하여 방화벽 문제를 해결 할 수 있다.

o 두 사이트가 모두 방화벽이 있고 들어오는 연결을 차단한다면 상황이 더 까다롭다. 아마 데이터 전송 연결을 위한 일부 포트 개방에 대해 방화벽 관리자에게 문의해야 할 것이다. 또한 방화벽 외부에 데이터 서버를 배치하는 것도 고려해 볼 수 있다. 이 경우의 예가 Science DMZ아키텍처이다. 이것은 방화벽에 의해 야기될 수 있는 잠정적인 성능의 이슈들을 피할 추가 혜택이 있다.

GridFTP 서버에 대한 포트 범위를 지정하는 방법

o 아래의 파일을 편집하여 GridFTP 서버가 사용하는 포트를 지정할 수 있다:

```
/etc/grid-security/sshftp
/etc/gridftp.conf
```

o 다음과 같이 원하는 포트에 GLOBUS_TCP_PORT_RANGE을 수정한다. 예를 들면:

```
GLOBUS_TCP_PORT_RANGE = 50000,50050
```

o 클라이언트가 사용하는 포트를 지정하려면 다음 파일을 수정할 수 있다:

```
/usr/share/globus/gridftp-ssh
```

o 다음 라인을 찾는다:

```
/usr/bin/ssh $port_str $remote_host $remote_program
```

o 아래와 같은 것을 대신 사용한다:

```
/usr/bin/ssh $port_str $remote_host GLOBUS_TCP_PORT_RANGE=x,y
$remote_program
```

4.4. scp 와sftp

o 유닉스 환경에서 scp 와sftp그리고rsync는 일반적으로 호스트 사이에 데이터를 복사하기 위해 사용된다. 이 툴들이 로컬 환경에서 잘 작동할 지라도 제대로 성능을 발휘하지 못한다. 버전4.7 이전의 openssh에서는 기껏64 KB였었는데 scp 와 sftp의 openssh버전은 1MB 버퍼를 차지하고, 이것은 WAN에서의 성능에 심각한 제약이다. rsync가 openssh 배포판의 일부가 아닐지라도 rsync는 일반적으로 전송을 위해 ssh를 사용하기 때문에ssh의 하부구조를 구현할 때 피할 수 없는 제약의 대상이 된다. 5ms정도 보다 더 긴RTT를 가진 네트워크 경로를 통해 대규모 데이터 세트를 전송해야 하는 경우 이런 도구를 사용하지 말것을 권장한다

o 이 문제를 해결하기 위한 패치는 피츠버그 슈퍼컴퓨터 센터 (Pittsburgh Supercomputer Center, http://www.psc.edu/index.php/hpn-ssh)에서 구할 수

있다. 이 패치는WAN상에서 단일 스트림 성능의 최적화가 가능하도록 한다. 그러나, WAN을 통해 대량 데이터 전송을 완벽히 최적화하기 위해서 GridFTP 등과 같은 병렬 스트림 도구 중 하나를 사용하는 것을 추천한다.[21]

sftp: 보안 파일 전송 프로그램(Secure File Transfer Program)

o 전술 한 바와 같이 PSC에서HPN 패치를 설치하지 않았다면 WAN 전송을 위해서 이 프로그램을 사용하는 것을 생각조차 하지 마라. 그러나 이 패치가 설치되었다 하더라도 SFTP는 다른 모든 것들보다 상위에 플로우 제어 메커니즘을 배치하고 있는 성가신 특징이 있다. sftp는 기본값으로 16 32KB크기의16개 메시지로 수를 제한한다. 각 데이터 그램은 별개의 메시지이기 때문에512KB 데이터 제한에 부딪친다. 명령 줄에서 메시지의 수('-R')와 메시지('-B')의 크기를 모두 증가시킬 수 있다.

128MB window에 대한 *sftp* 예:

```
sftp -R 512 -B 262144 user@host:/path/to/file outfile
```

5. Network Test Tools

5.1. 네트워크 측정 도구

o 네트워크 테스트 도구의 최신 버전을 설치하는 가장 쉬운 방법은 perfsonar.net 소프트웨어의 레포지토리를 이용하는 것이다.

```
RHEL 기반 시스템:
 rpm -hUv

 http://software.internet2.edu/rpms/el6/x86_64/main/RPMS/Internet2-repo-0.6-1.noarch.rpm

yum -y install perfsonar-tools

Debian "wheezy" 와 Ubuntu "Precise" 기반 시스템:
cd /etc/apt/sources.list.d/
wget
http://downloads.perfsonar.net/debian/perfsonar-wheezy-release.list
wget -qO -
 http://downloads.perfsonar.net/debian/perfsonar-wheezy-release.gpg.key | apt-key add
apt-get update
```

```
apt-get install perfsonar-tools

OSX 시스템:
brew install iperf iperf3 nuttcp bwctl owamp
```

참고 : http://fasterdata.es.net/host-tuning/ 에서 설명된 대로 호스트의 TCP 설정을 조정한다

표 3 네트워크 성능 측정 도구

측정 도구	특징
bwctl	스케줄 검사를 한 번에 하나씩 ((wrapper for iperf and nuttcp))
iperf/iperf3	end-to-end TCP/UDP 성능 측정
nuttcp	end-to-end TCP/UDP 성능 측정
owamp	지연, 손실, 지터(jitter) 측정
traceroute/scamper	네트워크 경로와 MTU를 측정
pathrate/pathload	가능한 대역폭 및 기능 측정
UDPmon	네트워크 성능 측정 도구
SYNACK	icmp를 차단하는 사이트들에 대한 교체 핑(ping)을 제공

5.2. 네트워크 테스트 도구

표 4 네트워크 테스트 도구

테스트도구	목적
NDT / NPAD : Web100 기반 네트워크 테스트 도구	네트워크 문제 영역의 광범위한 테스트를 제공
tcpdump	특정 소스/목적지에 대한 모든 TCP 헤더 정보를 덤프
tcptrace	xplot을 사용하여 분석을 위한 tcpdump 출력을 형식화

6. KREONET 기반 SaaS 응용에 대한 자동화된 Cloud 실행 환경 구축

o 현재 KREONET은 소프트웨어 정의 네트워크 구성을 통해 연구망의 새로운 패러다임을 추진하고 있으며, 향후 이러한 망 환경하에서 100G, SDN, Science DMZ, Cloud as a Service 등의 다양한 미래 기술과 응용기술들의 접목을 추진 중에 있

다.

o 또한 글로벌 트렌트이면서 국내 연구진들의 활발한 연구활동을 통해 데이터 집약적 과학분야의 발전과 이를 위한 IT 인프라의 획기적인 발전과 지원이 절실한 시점이다.

o 미국의 ESnet을 중심으로 미국전역의 유수의 대학과 연구소 등의 협의로 진행되고 있는 Science DMZ를 통해 차세대 미국 연구망 환경의 고속화를 위한 PRP Project(Pacific Research Platform)가 진행 중에 있다. PRP 프로젝트는 현재 Science DMZ로 구축되어 있는 데이터 집약형 과학 분야의 빅데이터 전송 기술을 토대로 빅데이터 기반의 응용연구 분야에 대해 획기적 연구환경의 제공을 목표로 하고 있다.

o 이러한 프로젝트를 통하여 '빅데이터 하이웨이'를 슬로건으로 하여 미국 서부 해안지역을 시작으로 거대 백본을 통한 미국 전역에 대한 새로운 연구환경이 구성되고 있다. 100G 네트워크를 기반으로 빅데이터의 획기적인 전송과 Super Facility들의 획기적인 연동을 통해 차세대 연구 환경을 구축한다.
주요 연구 대상으로는1)입자물리, 2)천문, 3)바비오, 4)지구과학, 5)가시화 등의 주요 5개분야에 대한 중점지원을 통해 빅데이터 기반의 응용연구를 지원한다.

o KREONET에서도 2015년 10월에 미국 PRP프로젝트의 국제 파트너로서 참여 중에 있으며 서울-대전간 SDN 기반의 망 분리를 통해 과학데이터 전용 환경을 구축하고, 이를 토대로 Science DMZ 및 DTN 환경의 구축을 통하여 국내 데이터 집약형 과학분야의 응용연구 활성화와 PRP프로젝트와의 연계를 통해 국제간 빅데이터 기반의 응용연구의 활성화를 추진하고 있다.

7. KISTI Cloud 설치 및 검증

7.1. 설치 환경 및 구성 요소

o 본 문서에서는 아래 그림16과 같이 KISTI Cloud 및 DTN 노드들로 구성된 테스트베드에 대해 기술한다. 이용된 OpenStack 버전은 2016년 상반기에 발표된 Mitaka 버전이 활용되었고, 컨트롤러, 네트워크, 컴퓨트의 세 개의 노드들로 구성된다. 각 노드에는 OpenStack Mitaka 버전 설치 시 권장되는 Ubuntu 14.04.2LTS가 설치된다.

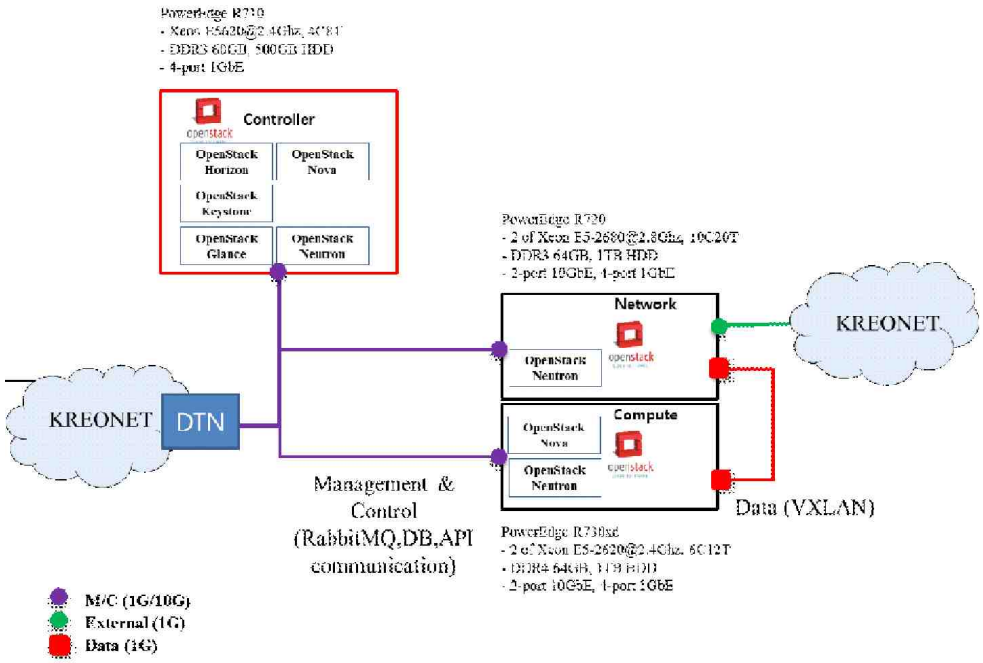

그림 16 KISTI Cloud 테스트베드 구성도

o 주요 노드들을 대상으로 Keystone, Glance, Nova, Neutron, Horizon등의 서비스들이 설치된다. Nova 컴퓨트 노드들을 통해 생성된 VM들의 내부 통신을 위한 Tenant 네트워크 구성을 위해 VxLAN 기반의 내부 네트워크가 구성되고, GRE, VxLAN 등의 오버레이 Network type을 지원하는 Open vSwitch 에이전트가 네트워크 및 컴퓨트 노드들에 설치된다.

o Tenant 네트워크 외에 각 노드들의 관리 및 제어를 위한 네트워크, VM들을 외부에서 접근하기 위한 External 네트워크가 분리 구성된다. 또한, 관리 및 제어를 위한 네트워크는 DTN과의 연동을 위해 추가적인 분리 구성이 고려되어 확장될 수 있다.

7.2. 설치 검증

o 아래 그림16은 테스트베드상의 Nova 컴퓨트 서비스 리스트를 보여준다. 주요 서비스인 nova-consoleauth, nova-scheduler, nova-conductor, nova-compute 서비스들이 각각 컨트롤러 노드와 컴퓨트 노드에 구동 중인 것을 확인할 수 있다.

```
user@RL2-controller:~$ . admin-openrc
user@RL2-controller:~$ openstack compute service list
+----+------------------+----------------+----------+---------+-------+----------------------------+
| Id | Binary           | Host           | Zone     | Status  | State | Updated At                 |
+----+------------------+----------------+----------+---------+-------+----------------------------+
|  7 | nova-consoleauth | RL2-controller | internal | enabled | up    | 2016-08-16T07:00:50.000000 |
|  8 | nova-scheduler   | RL2-controller | internal | enabled | up    | 2016-08-16T07:00:52.000000 |
|  9 | nova-conductor   | RL2-controller | internal | enabled | up    | 2016-08-16T07:00:54.000000 |
| 16 | nova-compute     | RL2-compute1   | nova     | enabled | up    | 2016-08-16T07:00:53.000000 |
+----+------------------+----------------+----------+---------+-------+----------------------------+
```

그림 17 Nova 컴퓨트 서비스 리스트

o 아래 그림17은 테스트베드 상의 Neutron 에이전트 리스트를 보여준다. 주요 에이전트들인 neutron-openvswitch-agent, neutron-l3-agent, neutron-dhcp-agent, neutron-metadata-agent들이 각각 네트워크 노드와 컴퓨트 노드에 구동 중인 것을 확인할 수 있다.

```
user@RL2-controller:~$ neutron agent-list
+--------------------------------------+--------------------+----------------+-------------------+-------+----------------+---------------------------+
| id                                   | agent_type         | host           | availability_zone | alive | admin_state_up | binary                    |
+--------------------------------------+--------------------+----------------+-------------------+-------+----------------+---------------------------+
| 0a47eb7c-6b2d-4cf7-ad2a-4314b3eea1fa | Open vSwitch agent | RL2-network    |                   | :-)   | True           | neutron-openvswitch-agent |
| 4aa36ccf-92e7-406f-a271-11dc3d8cbb28 | Open vSwitch agent | RL2-compute1   |                   | :-)   | True           | neutron-openvswitch-agent |
| 564f917e-f6ea-4ee4-9dd2-020754f3648a | L3 agent           | RL2-network    | nova              | :-)   | True           | neutron-l3-agent          |
| 762dd734-7619-454b-81ce-dcef3e77fdcc | DHCP agent         | RL2-network    | nova              | :-)   | True           | neutron-dhcp-agent        |
| e399c206-f098-432f-8ad7-791b70446d4c | Metadata agent     | RL2-network    |                   | :-)   | True           | neutron-metadata-agent    |
+--------------------------------------+--------------------+----------------+-------------------+-------+----------------+---------------------------+
```

그림 18 Neutron 에이전트 리스트

o 그림18에서는 OpenStack의 Dashboard 서비스인 Horizon 서비스를 통해 관리자 모드로 로그인을 했을 때, 하이퍼바이저를 통해 제공 중인 자원들의 요약 정보를 보여준다. 테스트베드 구성도의 컴퓨트 노드의 하드웨어 스펙에 준하는 VCPU, 메모리, 로컬 디스크의 총량 및 사용 중인 자원들의 정보를 얻을 수 있다.

그림 19 관리자 GUI에서의 하이퍼바이저 요약

o 그림19에서는 Horizon 서비스를 통해 일반 사용자 모드로 로그인을 했을 때, 사용자가 생성한 VM 및 VM들이 속한 Tenant 네트워크인 “demo-net”, “demo-router” 및 외부 네트워크와의 통신을 위한 “ext-net”에 대한 네트워크 토폴로지 정보를 보여준다.

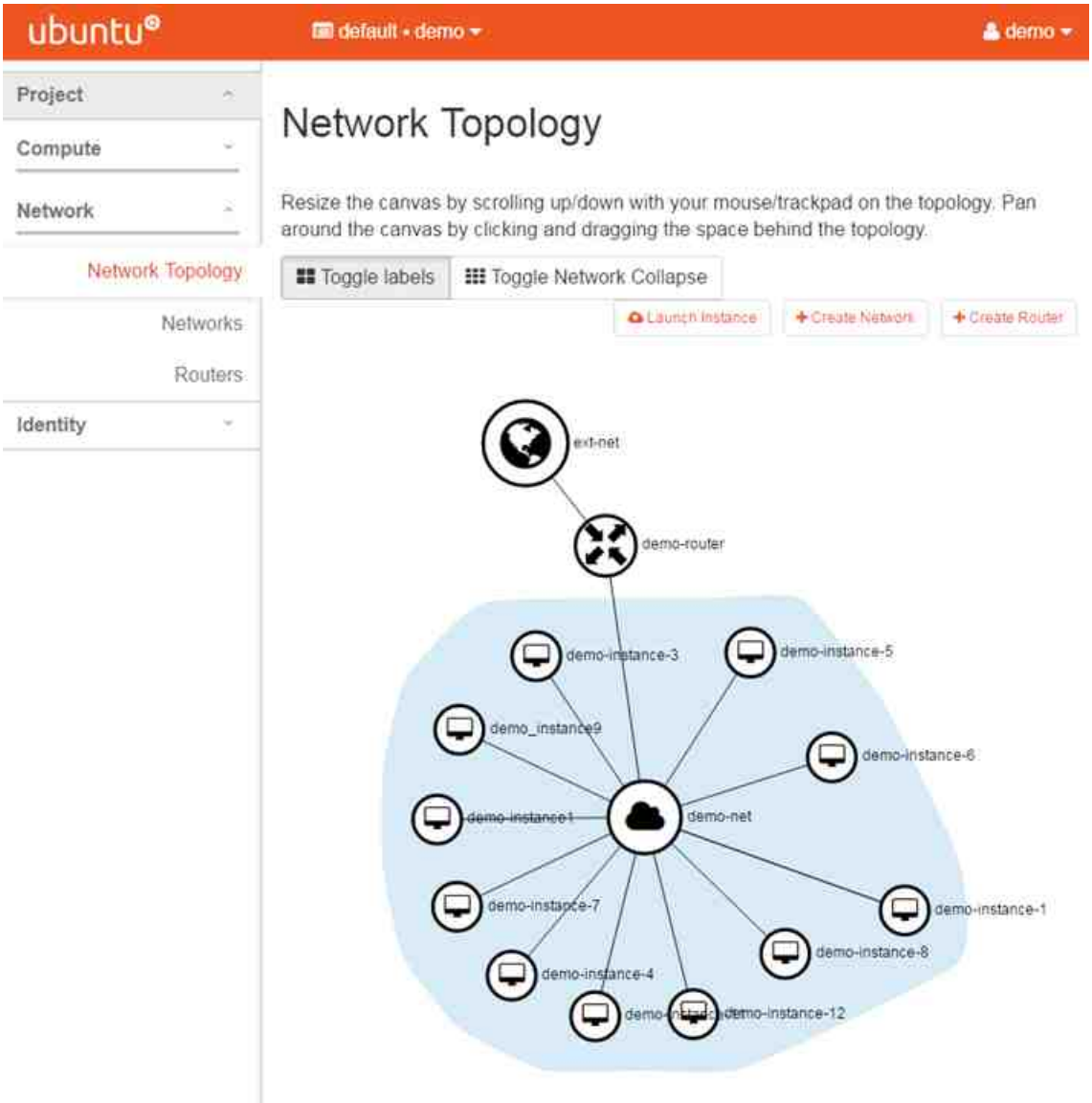

그림 20 일반 사용자 GUI에서의 Tenant 네트워크 토폴로지

8. 자동화된 OverCloud 환경의 KREONET 적용을 위한 DTN 구축

o SmartX 공용개발환경을 활용한 통합환경의 구축과 기본적인 SaaS 응용에 대한 자동화된 OverCloud 실행환경의 구현을 위한 초고속 데이터 전송을 위한 ScienceDMZ의 구축을 통한 DTN(Data Transfer Node)의 구축을 추진한다.

o 앞절에서 언급되었듯이 Science DMZ의 개념과 구축은 과학분야의 빅데이터들의 효과적인 전송과 향후 100G환경, SDN 환경, 네트워크 가상화 등의 첨단 응용들의 유연한 접목을 위해 발전되어 왔으며, 본 과제에서도 동일한 네트워크 환경의

구축과 성능 검증을 통해 자동화된 OverCloud실행 환경의 제공을 추진한다.

o 현재 KREONET 상에서는 서울-대전 간 SDN 기반의 논리적 전용망 구성이 가능하며, 이러한 토대 위에 과학데이터와 일반데이터의 분리를 통해 과학분야 빅데이터들의 전송 성능에 대한 획기적이 성능 향상을 제공한다.

o KREONET Science DMZ 구축을 위한 DTN의 스펙은 다음과 같다.

표 5 KREONET DTN Specification

항목	사양
CPU	Intel® XeonTM E5-2643v3 (3.4GHz, 20MB ,6-Core)
Memory	128GB DDR4 ECC REG PC4-17000R(8 x DDR4 slots, max. 1TB)
Mother Board	SUPERMICRO MBD-X10SRL-F Server Motherboard, Cache : 1GB
Network Interface Card	Mellanox ConnectX-3 Pro EN Single-Port 40/56 Gigabit Ethernet - driver: mlx4_en - version: 3.4-1.0.0 (25 Sep 2016) - firmware-version: 2.36.5150
Hard Disk	SSD Samsung 850 Evo 1TB - Sequential Read : 540MB/s - Sequential Write : 520MB/s
OS	CentOS7.2 - Kernel: 3.10.0-327.36.3.el7.x86_64

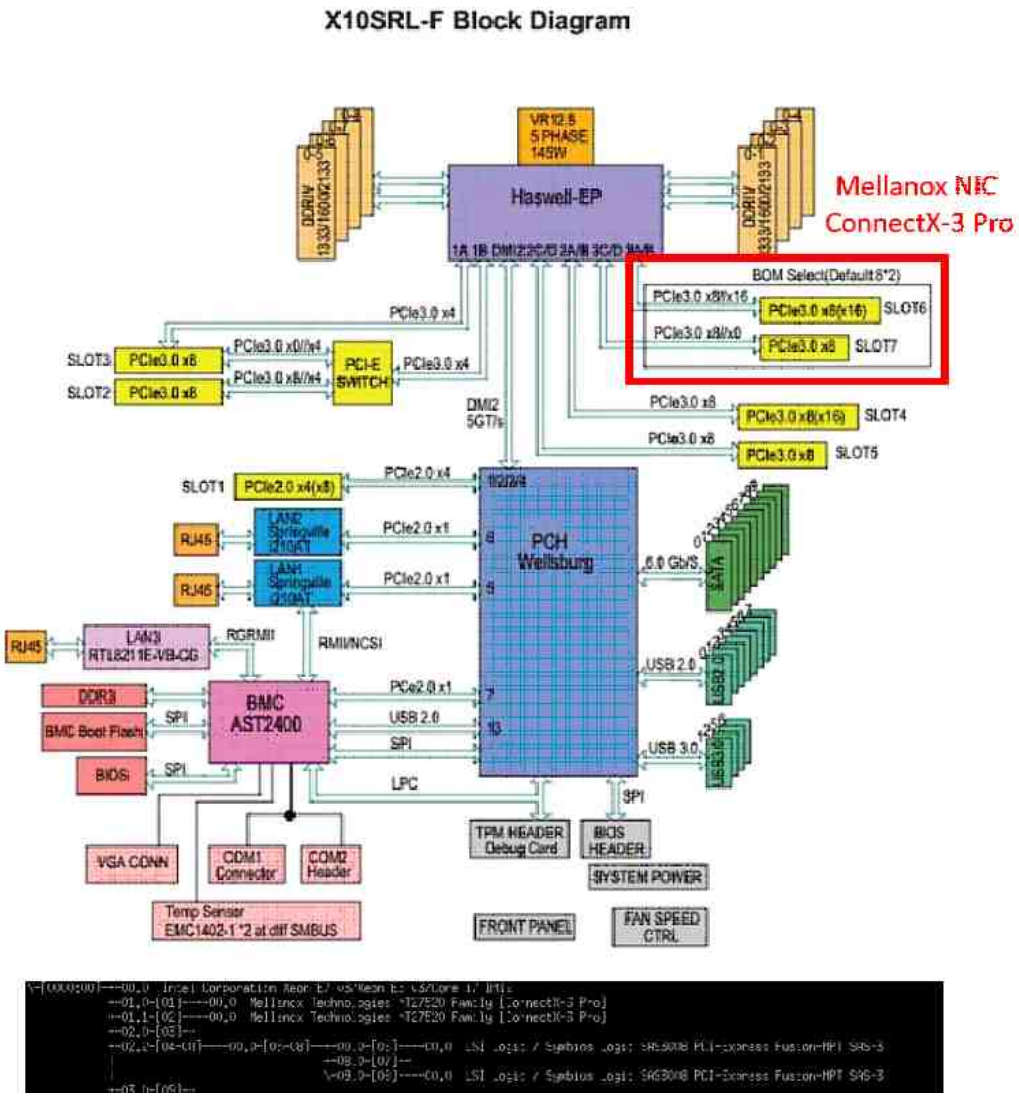

그림 21 KREONET DTN Architecture

o openFlow기반의 SDN 환경을 활용하여VDN(Virtual Dynamic Network)기능을 통해 기존 망과의 분리가 되며, OSCARS 등과 같은 프로비저닝을 위한 NSI프로토콜의 적용도 가능하다. 사용자가 요구에 따라 망의 설정을 통해 Science DMZ 환경의 서비스를 제공할 수 있다

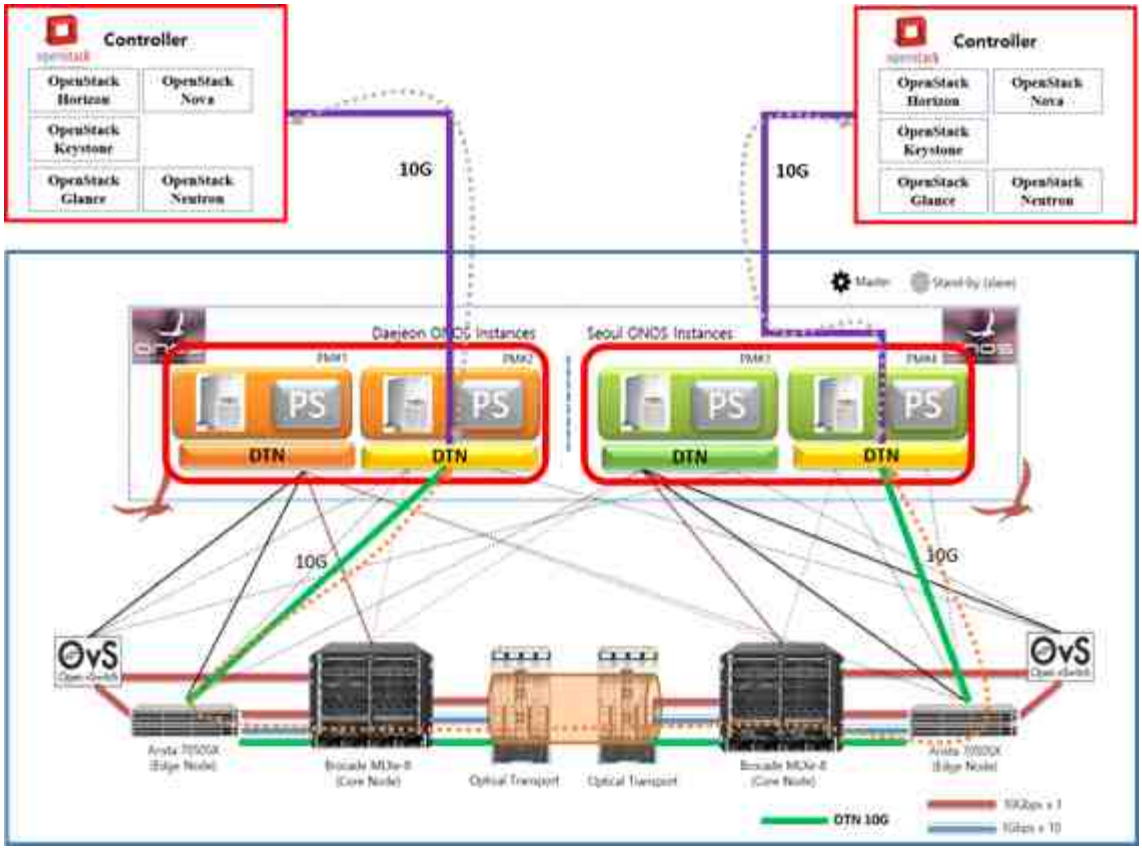

그림 22 KREONET환경에서의 Science DMZ

o KREONET Science DMZ기반의DTN 구축을 통해 향후 100G 환경에서 핵심적인 기능인 고속전송, 빅데이터 공유, 데이터 케싱 등, 차세대 첨단 네트워크 기술의 접목 등을 통해 데이터 집약형 과학분야의 획기적인 연구환경의 제공이 가능하다. 이러한 빅데이터들의 활용도 향상을 통해 국내 빅데이터 기반의 응용분야 활성화 뿐만 아니라 Science DMZ를 토대로 구성되는 국제간 대규모 연구환경에 대한 서비스가 이루어진다.

References

[1] Science DMZ https://fasterdata.es.net/science-dmz/

[2] RDMA, RoCE http://www.es.net/news-and-publications/esnet-news/2012/at-100-gbps-esnet-puts-network-research-on-fast-track/

[3] GlobalNOC http://globalnoc.iu.edu/sdn/oess.html

[4] PerfSONAR https://fasterdata.es.net/performance-testing/perfsonar-testing-to-cloud-resources/

[5] Firewall Problem https://fasterdata.es.net/assets/fasterdata/Firewall-tcptrace.pdf

[6] Science DMZ Security Eli Dart's ScienceDMZSecuritytalkatTIP2013 https://meetings.internet2.edu/2013-01-jt/program/?go=session&id=10002778&event=1261

[7] Linux Tuning https://fasterdata.es.net/host-tuning/linux/expert/

[8] Host Tuning/testhttps://fasterdata.es.net/host-tuning/linux/test-measurement-host-tuning/

[9] TCP performance enhance https://fasterdata.es.net/host-tuning/linux/recent-tcp enchancements/

[10] Interrupt Binding https://fasterdata.es.net/host-tuning/100g-tuning/interrupt-binding/

[11] HTB Packet Pacinghttps://fasterdata.es.net/host-tuning/packet-pacing/htc-based-pacing/

[12] LAMP http://www.onlamp.com/pub/a/onlamp/2005/11/17/tcp_tuning.html

[13] PSC TCP tuning http://www.psc.edu/index.php/networking/641-tcp-tune

[14] TCP Auto Tuning http://www.psc.edu/index.php/networking/641-tcp-tune

[15] TCP Congestion Avoidance Algorithm http://en.wikipedia.org/wiki/TCP_congestion_avoidance_algorithm

[16] 10G Network https://fasterdata.es.net/host-tuning/packet-pacing/htc-based-pacing/

[17] Fare Queuing(FQ) https://fasterdata.es.net/assets/fasterdata/FQ-pacing-results.pdf

[18] 100G Tuning https://fasterdata.es.net/assets/Papers-and-Publications/100G-Tuning -TechEx2016.tierney.pdf

[19] MTU http://staff.psc.edu/mathis/MTU/

[20] Jumbo Frame http://kb.pert.geant.net/PERTKB/PathMTU

[21] Pittsburgh Supercomputer Center, http://www.psc.edu/index.php/hpn-ssh