VDE 기반 클라우드 컴퓨팅 서버를 위한 클라이언트 단위 네트워크 성능 고립화

차승우°, 유종훈, 홍성수
서울대학교 전기컴퓨터공학부
{swcha°, jhyoo, sshong}@redwood.snu.ac.kr

Per-Client Network Performance Isolation in VDE-based Cloud Computing Servers

Seungwoo Cha°, Jonghun Yoo, Seongsoo Hong
School of Electrical Engineering and Computer Science, SNU

요 약

클라우드 데이터 센터 내의 각 서버는 서로 다른 클라이언트에 속하는 복수의 가상 머신을 수행한다. 이들은 네트워크 자원을 공유하기 때문에 과도한 트래픽을 발생시키는 일부 클라이언트가 다른 클라이언트들의 서비스 품질을 급격히 저하시키는 문제가 발생할 수 있다. 이를 해결하기 위한 기존의 네트워크 고립화 기술들은 클라이언트에 대한 고려가 없거나 클라우드와 같은 대규모 네트워크에 적용하기 어렵다는 한계가 있다. 이 논문에서는 VDE(Virtual Distributed Ethernet)를 기본 프레임워크로 사용한 클라이언트 단위의 네트워크 성능 고립화 기법을 제안한다. 제안된 기법은 각 클라이언트에 가중치를 부여하고 이에 비례하여 네트워크 대역폭을 할당하기 위해 fair share 스케줄링과 송수신 패킷 dispatch를 수행한다. 제안된 기법은 기존 시스템에 최소한의 변경만을 가하여 구현 가능하므로 대규모 네트워크에 도입이 용이하다. 실험 결과 각 클라이언트는 가중치로부터 도출된 대역폭의 99.4% 이상을 할당 받는 것으로 나타나 제안된 기법의 효용성이 입증되었다.

1. 서 론

클라우드 데이터 센터는 수천에서 수만 대의 서버로 구성되며 이들은 각기 다른 요구사항을 가진 클라이언트들에게 컴퓨팅 자원을 제공한다. 시스템 가상화는 클라우드 컴퓨팅의 핵심 기술로서 OS에게 가상 머신이라 불리는 논리적 하드웨어를 제공하여 복수의 OS가 동일한 하드웨어 자원을 공유할 수 있도록 하는 효과적인 기법이다. 이때 가상화된 OS는 게스트 OS라고 불린다.

클라우드 서버는 서로 다른 클라이언트에 의해 소유되는 복수의 가상 머신을 수행한다. 이들 가상 머신은 물리 NIC(network interface card)를 공유하기 때문에 과도한 트래픽을 발생시키는 일부 클라이언트에 의해 다른 클라이언트의 네트워크 성능이 급격히 저하될 수 있다. 따라서 클라우드 서비스의 품질 보장을 위해서는 클라이언트 간 네트워크 성능의 고립이 필수적이다.

기존의 네트워크 성능 고립화를 위한 연구로 Quantized Congestion Notification (QCN) [1], Class of Service (CoS) [2], Seawall [3], TCP flow 제어 기법 등이 존재한다. 그러나 이들 기법은 다음과 같은 이유로 인해 대규모 클라우드 네트워크에 적용에 한계가 있다. 첫째, QCN은 2계층 도메인에 적용이 제한되며 네트워크의 모든 노드의 하드웨어와 소프트웨어 수정을 요구한다. 둘째, CoS는 각 포트마다 단 8개의 서비스 클래스만을 정의할 수 있어 확장성이 제한된다. 셋째, Seawall은 게스트 OS의 네트워크 스택에 수정을 요구하여 기존 시스템과 호환성이 낮다. 마지막으로 TCP flow 제어는 TCP flow 단위의 고립만을 지원하여 클라이언트간 고립에 효과적이지 못하다.

이 논문에서는 이러한 문제를 해결하기 위해 VDE(Virtual Distributed Ethernet)[4]를 기본 프레임워크로 사용하는 네트워크 고립화 기법인 FS-VDE(Fair Share-VDE)를 제안한다. FS-VDE는 각 클라이언트에 부여된 가중치에 비례하여 송수신 링크의 대역폭을 할당한다. 이를 위해 송신 트래픽에 대해 VTRR (Virtual-Time Round-Robin) [5] 알고리즘에 기반한 트래픽 shaping을 수행하고, 수신 트래픽에 대해 토큰 버킷 [6] 알고리즘에 기반한 policing을 수행한다. 제안된 기법은 기존 네트워크 스택의 변경을 요구하지 않으며, 게스트 OS의 전가상화를 지원하고 런타임 오버헤드가 낮다는 장점을 갖는다. 우리는 제안된 기법을 오픈 소스 VDE 스위치 상에 구현하고 실험을 통해 성능을 검증하였다. 실험 결과 각 클라이언트는 가중치로부터 도출된 대역폭의 99.4% 이상을 할당 받았으며 이때 네트워크 대역폭의 오버헤드는 5%에 불과하였다.

논문의 나머지 부분은 다음과 같이 구성된다. 2장에서 기존 VDE에 대해 소개하고, 3장에서 대상 시스템의 문제를 정의한다. 4장에서 FS-VDE 기법을 상세히 설명하고, 5장에서 실험을 통해 검증한다. 끝으로 6장에서 논문의 결론을 맺는다.

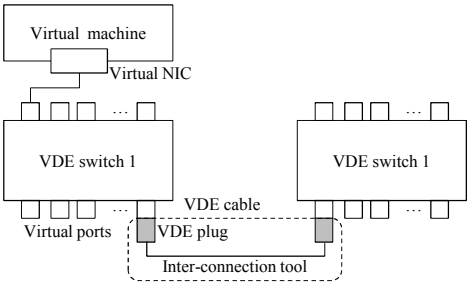

그림 1. VDE 네트워크 예시

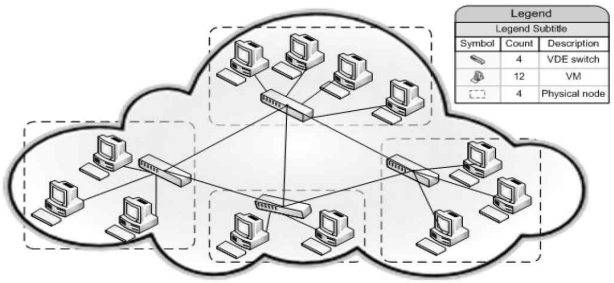

그림 2. 대상 클라우드 컴퓨팅 시스템 구조

2. Virtual Distributed Ethernet (VDE)

VDE는 물리 머신, 가상 머신, 에뮬레이터, 응용 등을 연결하는 가상의 Ethernet 호환 네트워크를 제공하는 소프트웨어이다. VDE는 클라우드 컴퓨팅, VPN (Virtual Private Network), 네트워크 교육 등 다양한 분야에서 널리 활용되고 있으며 최근에는 Eucalyptus[7]를 포함한 오픈 소스 클라우드 컴퓨팅 플랫폼에서 널리 사용되고 있다.

VDE는 크게 스위치와 케이블이라는 두 가지 컴포넌트로 구성된다. 그림 1에 나타난 VDE 네트워크의 예에서 두 VDE 스위치가 VDE 케이블에 의해 연결되어 있고 스위치 1은 가상 머신과 연결되어 있다. VDE 케이블은 plug와 inter-connection tool 두 개의 구성 요소로 이루어져 있다. VDE plug는 입출력 패킷 트래픽을 OS의 표준 스트림 연결로 전환하고, inter-connection tool은 두 plug의 스트림을 양방향으로 전달하는 일을 한다. VDE 스위치와 가상 머신 사이의 연결은 *libvdeplug*라 불리는 VDE 라이브러리가 제공하는 인터페이스를 통해 이뤄진다. VDE 스위치와 물리 NIC와의 연결을 위해 TUN/TAP 가상 네트워크 인터페이스가 사용된다.

VDE 스위치의 포트는 file descriptor를 통해 가상 머신과의 패킷 전송을 관리한다. VDE는 polling 방식으로 관리 중인 모든 file descriptor를 확인하여 새로운 패킷의 도착 유무를 판단한다. 만약 외부로부터 도착한 패킷이 있다면 VDE 스위치가 패킷의 목적지의 주소를 hash 테이블에서 찾는다. 만약 테이블에 해당 주소가 존재하고 이와 연결된 포트가 있다면 그 포트를 통해 패킷을 송신한다. VDE 스위치에는 스위치와 허브의 두 가지 모드가 존재한다. 스위치 모드에서는 목적지 포트로 패킷을 전달하고 허브 모드에서는 모든 포트를 통해 송신한다.

3. 시스템 모델 및 문제 정의

3.1 대상 시스템 구조

클라우드 데이터 센터는 그림 2에 나타난 것과 같이 다수의 서버로 구성된다. 각 서버는 호스트 OS와 가상 머신 모니터를 수행하고 가상 머신 모니터는 복수의 가상 머신을 생성하고 관리한다. 또한 가상 머신 모니터는 그림 2와 같이 VDE 스위치와 케이블을 통해 가상 머신들을 외부 네트워크와 연결한다. 각 물리 노드에는 하나의 VDE 스위치가 생성되고 여기에 가상 머신들과 물리 NIC가 연결된다.

3.2 문제 정의

우리의 목표는 앞 절에서 기술된 클라우드 시스템 위에서 클라이언트 단위의 네트워크 성능 고립화를 위한 메커니즘을 제공하는 것이다. 여기서 클라이언트는 클라우드 서버가 제공하는 서비스를 구매한 개인이거나 기관이다. 이들은 서버 내의 가상 머신을 통해 하드웨어 자원을 공유한다.

임의의 서버를 공유하는 클라이언트의 집합을 $\{C_1, C_2, \ldots, C_M\}$이라 하자. 그리고 클라이언트 C_i의 가중치를 w_i라 하고, C_i가 소유한 가상 머신의 집합을 $\{v_1^i, v_2^i, \ldots, v_n^i\}$이라 하자. 주어진 링크 대역폭을 B라고 할 때 클라이언트 C_i는 다음 과 같이 가중치에 비례하여 대역폭 B_i를 할당 받아야 한다.

$$B_i = B\frac{w_i}{\sum_{j=i}^{m} w_j}$$

이때 클라이언트가 소유한 가상 머신들은 아래와 같이 B_i를 나눠 받는다. $b(v_j^i)$는 가상 머신 v_j^i의 대역폭이다.

$$B_i = b(v_1^i) + b(v_2^i) + \ldots + b(v_n^i)$$

제안된 기법은 위의 조건을 만족시키기 위해 (1) 클라이언트 간 공정한 대역폭을 할당하고 (2) 각 클라이언트가 소유한 가상 머신 간 공정한 대역폭을 할당한다.

4. 해결 방안

이 논문에서 제안된 FS-VDE (Fair share-VDE) 기법은 VDE를 기본 프레임워크로 하여 앞 장에서 정의한 두 문제를 해결한다. FS-VDE는 송신 트래픽에 대한 shaping 메커니즘과 수신 트래픽에 대한 policing 메커니즘으로 구성된다.

4.1 송신 트래픽 shaping 메커니즘

FS-VDE는 송신 트래픽의 클라이언트간 고립화를 위하여 2-level 링크 대역폭 스케줄링을 수행한다. 첫 번째 단계 스케줄링은 클라이언트 사이에서 수행된다. 이를 위해 O(1)의

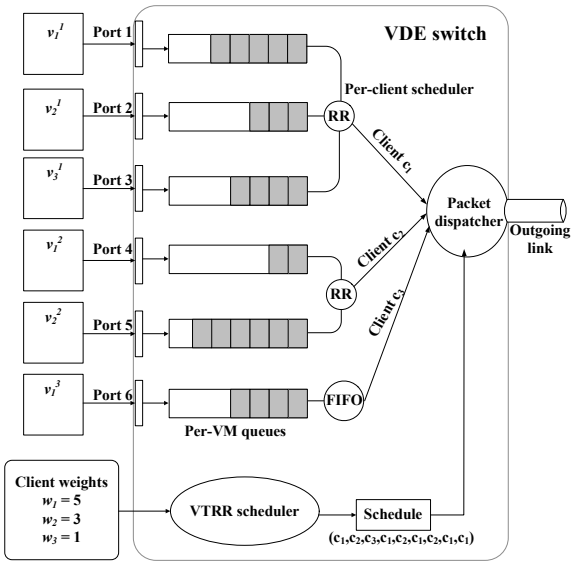

그림 3. 송신 트래픽 shaping 메커니즘

낮은 계산 복잡도를 가지면서 높은 공정성을 보장하는 VTRR 스케줄링 알고리즘을 통해 클라이언트의 스케줄을 생성한다. 그리고 두 번째 단계에서는 동일 클라이언트 내의 가상 머신 사이의 스케줄링이 수행된다.

구체적으로, 우리는 기존 VDE 스위치 안에 (1) VTRR 스케줄러, (2) 송신 패킷 dispatcher, (3) 클라이언트별 스케줄러, (4) 가상 머신별 패킷 큐의 총 네 가지 컴포넌트를 추가하였다. 그림 3은 추가된 컴포넌트가 포함된 VDE 스위치의 구조를 나타낸다.

그림 3에서 가중치 5, 3, 1을 갖는 세 개의 클라이언트 C_1, C_2, C_3가 주어졌을 때 VTRR에 의해 클라이언트의 스케줄 시퀀스 $(C_1, C_2, C_3, C_1, C_2, C_1, C_2, C_1, C_1)$이 생성되는 것을 볼 수 있다. 송신 패킷 dispatcher는 이 시퀀스에 따라 패킷을 송신할 클라이언트를 결정한다. 이어서 클라이언트별 스케줄러를 호출하여 해당 클라이언트가 소유한 가상 머신 중 하나를 선택한다. 마지막으로 선택된 가상 머신의 큐에서 패킷이 꺼내져 물리 NIC를 통해 외부 네트워크로 전송된다.

4.2 수신 트래픽 policing 메커니즘

FS-VDE는 수신 트래픽의 클라이언트간 고립화를 위하여 클라이언트마다 가중치에 의해 정해진 최대 속도 이상으로 유입되는 패킷을 drop한다. 이를 위해 토큰 버킷 알고리즘이 사용된다. 이는 각 클라이언트의 네트워크 대역폭에 상한선을 설정하여 트래픽 과부하를 효과적으로 막을 수 있다.

구체적으로, 우리는 기존의 VDE 스위치 안에 (1) 패킷 dispatcher 와 (2) 가상 머신별 토큰 버킷 두 컴포넌트를 추가하였다. 토큰 버킷은 (c, m, t_r)의 3-tuple로 정의되며 c는 현재 버킷 안에 든 토큰의 수, m은 해당 버킷이 수용할 수 있는 최대 토큰 수, t_r은 가장 최근에 토큰이 보충된 시각을 각각 나타낸다. 그림 4는 추가된 컴포넌트가 포함된 VDE 스위치의 구조를 나타낸다.

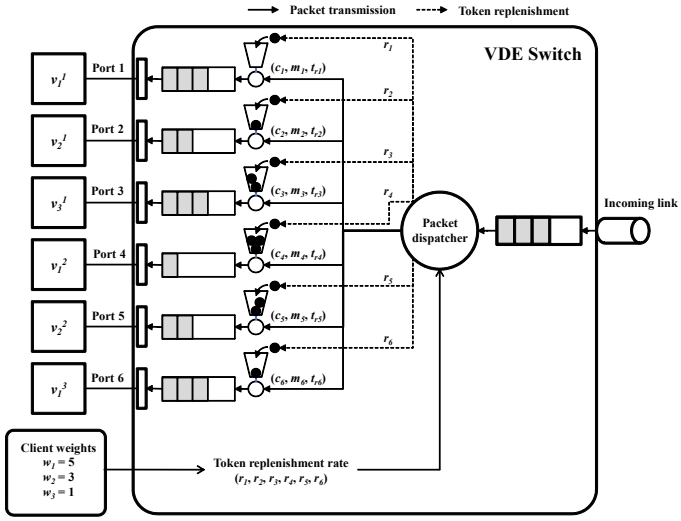

그림 4. 수신 트래픽 policing 메커니즘

수신 패킷의 처리는 forwarding 과정과 대역폭 할당의 두 과정을 거쳐 이루어진다. 먼저, forwarding 과정은 다음과 같다. 수신 버퍼로부터 패킷을 전달받은 패킷 dispatcher는 해당 패킷의 목적지 가상 머신의 토큰 버킷을 확인한다. 만약 토큰이 한 개 이상일 경우 해당 패킷을 가상 머신으로 전달하고 토큰을 하나 감소시킨다. 그리고 만약 토큰이 없다면 해당 가상 머신을 소유한 클라이언트에게 지정된 속도 이상으로 패킷이 전달되었음을 의미하므로 해당 패킷을 drop한다. 이어서 대역폭 할당 과정에서는, 패킷 dispatcher가 클라이언트 가중치로부터 도출된 replenishment rate값을 토대로 버킷에 토큰을 보충한다. 구체적으로, 패킷 dispatcher가 NIC로부터 패킷을 수신한 시간을 t라고 할 때 목적지 가상 머신의 토큰 버킷에 보충되는 토큰의 개수 c'은 다음과 같이 계산된다.

$$c' = \min(\lfloor (t - t_r) \cdot r \rfloor, m - c)$$

토큰이 보충된 후, 해당 버킷의 c와 t_r은 각각 c'와 c'/r만큼 증가된다.

5. 실험 평가

우리는 VDE 버전 2.2.3 상에 제안된 메커니즘을 구현하고 일련의 실험을 통해 네트워크 성능 고립 효과와 오버헤드를 검증하였다. 실험에 사용된 물리 머신와 가상 머신의 구성은

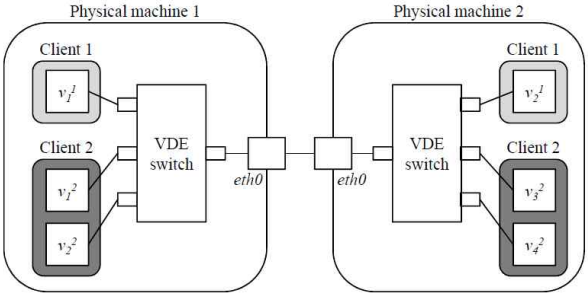

그림 5. 실험 구성

그림 5에 나타난 것과 같다. 물리 머신은 64-bit Intel Core2 Duo 프로세서와 2GB RAM으로 구성되었으며 Linux 2.6.32 커널을 호스트 OS로 사용하였고 KVM을 가상 머신 모니터로 사용하였다. 각 호스트 OS는 세 개의 가상 머신을 운영하며 이들은 VDE 스위치를 통해 연결된다. 물리 머신 사이는 VDE plug와 connection tool을 사용하여 연결하였다.

총 6개의 가상 머신은 클라이언트 1, 2가 그림 5와 같이 소유하고 있으며 이들에게 부여된 가중치는 각각 1과 2이다. 표 2과 같이 송·수신 TCP flow를 생성하고 각 flow와 클라이언트별 throughput을 측정하였다.

실험 결과는 표 1에 나타난 것과 같다. 기존 VDE는 TCP 흐름 제어에 따라 flow 단위로 대역폭이 할당되기 때문에 flow를 많이 생성하는 클라이언트가 더 많은 대역폭을 할당 받는다. 따라서 총 네 개의 flow를 생성한 클라이언트 2가 한 개의 flow를 생성한 클라이언트 1에 비해 약 네 배의 대역폭을 차지했다. 반면 FS-VDE의 경우에는 클라이언트 1과 2의 가중치 비가 1:2, 대역폭 비가 1:2.01로 나타났다. 이는 곧 각 클라이언트가 가중치로부터 도출된 대역폭의 최소 99.4% 이상을 할당 받았음을 의미한다. 따라서 FS-VDE를 통해 클라이언트간 네트워크 성능 고립이 효과적으로 이루어졌음을 알 수 있다. 한편 기존 VDE의 대역폭 총합에 비해 FS-VDE의 대역폭 총합은 95% 수준으로 나타나 제안된 기법의 네트워크 오버헤드가 5%에 불과한 것을 확인할 수 있었다.

표 2. 송·수신 TCP flow 실험 설정

Flows	Client	Source VM	Destination VM
f1	c_1	V_1^1	V_2^1
f2	c_2	V_1^2	V_3^2
f3	c_2	V_1^2	V_4^2
f4	c_2	V_2^2	V_3^2
f5	c_2	V_2^2	V_4^2

표 1. 대역폭 실험 결과 (MB/s)

	Flows					Clients	
	f1	f2	f3	f4	f5	c_1	c_2
기존 VDE	136	110	116	116	104	136	446
FS-VDE	183	107	93	75	93	183	369

6. 결론

이 논문에서 우리는 VDE 기반 클라우드 컴퓨팅 플랫폼 상에서 클라이언트 단위의 네트워크 성능 고립화를 보장하기 위한 Fair Share-VDE를 제안하였다. 구체적으로, 송신 트래픽을 위한 shaping 메커니즘과 수신 트래픽을 위한 policing 메커니즘을 제안하였다. Shaping 메커니즘은 클라이언트와 가상 머신의 2-level 스케줄링을 통해 대역폭의 공정한 할당을 수행하며 이를 위해 O(1) time-complexity를 갖는 VTRR 스케줄링이 사용된다. 그리고 policing 메커니즘은 수신 트래픽을 모니터링하며 각 가상 머신에 할당된 대역폭 이상의 트래픽이 유입되는 경우 패킷을 drop시킨다. 이를 위해 각 가상 머신마다 토큰 버킷을 부여하고 가중치로부터 도출된 replenishment rate에 따라 토큰을 보충해 준다.

실험 결과, FS-VDE는 클라이언트가 기대하는 대역폭의 99.4%이상을 할당하여 클라이언트 단위 네트워크 성능 고립화를 보장하였으며, 기존 VDE에 비교하여 네트워크 오버헤드가 5%에 불과함을 입증했다.

Acknowledgement

본 연구는 삼성전자 DMC 연구소의 지원을 받아 수행하였음. (No. 0421-20110016, 고성능 멀티코어 시스템을 위한 가상화 기술 연구)

참고 문헌

[1] R. Pan, B. Prabhakar, and A. Laxmikantha, “QCN: Quantized Congestion Notification,” http://www.ieee802.org/1/files/public/docs2007/au-prabhakar-qcndescription.pdf, May 2007.

[2] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield, “Class-of-service mapping for QoS: A statistical signature-based approach to IP traffic classification,” Proc. Of Internet Meas. Conf. 2004, Sicily, Italy, pp. 135-148, October 2004.

[3] A. Shieh, S. Kandula, A. Greenberg, and C. Kim, “Seawall: performance isolation for cloud datacenter networks,” 2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud '10), 2010.

[4] R. Davoli, “VDE: Virtual Distributed Ethernet,” Proc. of IEEE/Create-Net Tridentcom 2005, Trento, Italy, pp. 213-220, May 2005.

[5] J. Nieh, C. Vaill, and H. Zhong, “Virtual-Time round-robin: An O(1) proportional share scheduler,” Proc. of USENIX Annual Technical Conference, Berkeley, CA, USA, pp. 245-260, June 2001.

[6] S. Shenker and J. Wroclawski, “General Characterization Parameters for Integrated Services Network Elements,” RFC2215, September 1997.

[7] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, “The Eucalyptus open-source cloud-computing system,” the 9th IEEE Int’l Symp. on Cluster Comp. and the Grid, 2009, (CCGRID '09), Shanghai, China, pp.124-131, May 2009.