리눅스 가상 서버의 구성방식과 스케줄링 알고리즘의 성능평가

박동국° 이용우
서울시립대학교 전자전기컴퓨터 공학부
dongugi@kolidong.net°, ywlee@uoscc.uos.ac.kr

Performance Evaluation of Organization Methods and Scheduling Algorithms in a Linux Virtual Server

Dong-Kook Park° Yong-Woo Lee
Dept. Electrical & Computer Engineering, University of Seoul

요 약

본 논문에서는 멀티미디어 데이터의 증가로 인한 네트워크 전송량 폭주 현상 및 서비스 비용의 증가를 감소시키기 위한 방법으로, 부하분산과 관련된 가상서버의 개념에 대해서 알아보고, 대표적인 가상 서버 중 하나인 리눅스 가상 서버 (LVS : Linux Virtual Server)의 성능 측정을 통해 멀티미디어 컨텐츠 전송에 사용되는 CDD 시스템에 가장 적합한 가상서버의 구성 및 알고리즘을 도출하였다. 이를 위해서 리눅스 가상서버 시스템을 구축하고, 3가지 구성방식과 8가지 스케줄링 알고리즘에 대한 성능 평가를 수행하였으며, 각 경우에 대한 비교분석을 하였다. 이를 토대로, 추후 CDD 시스템과의 접목을 통하여 네트워크 부하량을 감소시키고, 서버로부터 발생할 수 있는 병목현상을 적은 비용으로 해결할 수 있는 방향을 제시하고자 하였다.

주요어(key words) : CDD, 리눅스 가상 서버 (LVS), 로드밸런싱, Network address translation, IP Tunneling, Direct Routing

1. 서론

인터넷의 사용이 대중화되어가면서 멀티미디어 데이터 전송이 크게 증가하고 있다. 이로 인해 네트워크 전송량이 폭주하게 되고, 서비스를 하는 사이트의 경우, 많은 사용자들의 서비스요청을 신속하고 안정적으로 처리할 수 있도록 지속적으로 시스템을 유지 보수를 하는 데 많은 비용을 들이고 있다.

네트워크 전송량의 폭주는 현재 캐쉬[4]기술을 통해 많이 감소하고는 있지만 데이터 전송량은 갈수록 늘어나는 추세이므로 지속적인 연구가 필요하다. 최근에 많은 사람들이 클러스터(MPI[5], PVM[6] 등.)나 부하분산[7]과 관련되어 연구를 하고 있다.

본 논문에서는 부하분산과 관련된 가상서버의 개념에 대해서 알아보고, 대표적인 가상 서버 중 하나인 리눅스 가상서버(LVS : Linux Virtual Server)[1]의 성능 측정을 통해 멀티미디어 컨텐츠 전송에 사용되는 CDD (Contents Delivery and Distribution)[8] 시스템에 가장 적합한 가상서버의 구성 및 알고리즘을 도출하고자 한다.

2. 성능평가 실측환경

2.1 측정 네트워크 구성

실측을 위하여 [그림 1] 같은 측정용 네트워크를 구성하였고, 각 컴퓨터의 네트워크 설정은 [표 1]과 같다.

로드밸런서에는 하나의 네트워크 인터페이스 카드에 두 개의 IP를 할당하는 식으로 구성하였다. 따라서, 추후에 여러 개의 네트워크 상에 클라이언트와 서버를 분리하는 방식과, 로드밸런서에 두 개의 인터페이스 카드를 설치하여서 내부 IP와 가상 IP(VIP)를 별도의 카드에 할당하는 방식에서의 성능 측정도 실시하여, 성능의 변화를 관찰할 필요가 있다.

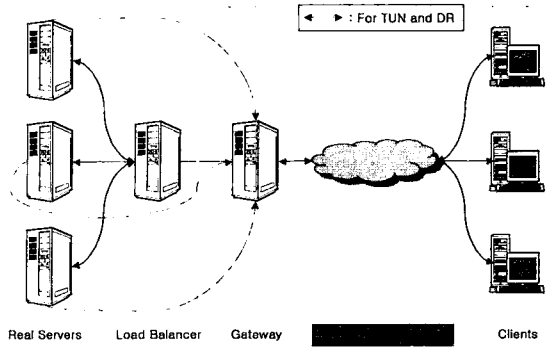


그림 1. 측정용 네트워크 구성도

표 1. 실측에 사용된 각 컴퓨터의 네트워크 설정

	컴퓨터	IP 주소
LVS Server	Load Balancer	192.168.0.1 VIP : 192.168.0.135
	Real Server1	192.168.0.2
	Real Server2	192.168.0.3
	Real Server3	192.168.0.4
Gateway	Gateway	192.168.0.100
Client	Client1	192.168.0.130
	Client2	192.168.0.131
	Client3	192.168.0.132

3. 측정대상 분석 및 측정방식 선정

3. 1. 측정대상 분석

실제적인 LVS의 성능을 측정하기 위해서 [그림 1]과 같은 실측 환경을 구축했고, LVS의 3가지 방식에서 모두 동작이 가능하도록 설정했다.

CDD시스템에서는 파일 내용의 버퍼링 및 다운로드 특성이 성능과 크게 관련되기 때문에, LVS의 성능을 평가하기 위해서 스트리밍 방식 파일의 다운로드받는 속도를 측정하였다. 측정에는 wget[2]이라는 프로그램을 사용하였고, 이 프로그램을 사용하여, 원하는 URL에 위치한 파일을 다운로드받고, 평균 수신속도를 측정하였다. 본 측정에서는 wget을 3대의 클라이언트에서 수행시켰고, 서비스요청의 수를 Round-Robin 방식으로 증가시켜 가면서 측정하였다.

3. 2. 측정 방식

우선, 테스트에 사용된 클라이언트와 LVS가 동일 네트워크 상에 위치하고, 접속 형태도 100Mbps Ethernet이므로, 클라이언트가 서버에 접속하는데 까지 걸리는 시간이나, 로드밸런서가 내부의 실제 서버에 접속하는 데 걸리는 시간은 매우 작게 된다. 따라서 이를 별도로 구분하지 않고, 서비스 받는 시간에 포함시켰다.

LVS를 구성하는 방식에는 Network Address Translation (NAT), IP-Tunneling(TUN), Direct Routing(DR)의 세 가지가 있고, 제공되는 스케줄링 방식에는 8가지가 있다. 본 논문에서는 이 3가지 구성 방식과 8가지의 스케줄링 알고리즘의 조합으로 총 24가지의 구성에 대해서 다음의 과정을 반복하면서 측정을 하였다.

① 우선 20MB크기의 파일에 대해서, 세 대의 클라이언트가 서비스요청의 수를 1개부터 10개까지 증가시키면서 각 반복마다 평균 수신율(R_{AT})을 측정한다.
② 파일 크기를 40MB, 60MB로 변경하면서 위의 ①의 과정을 반복한다.
③ 각 측정 결과를 토대로 다음 값들을 계산한다.
(1) 왕복응답시간(Turn-around time) (T_{TA})

$$T_{TA} = Filesize/R_{AT}$$

(2) 평균 시스템 작업 처리량 (T_{HS})

$$TH_s = \sum_{n=1}^{N}(Filesize/T_{TAn}) = \sum_{n=1}^{N}(R_{ATn})$$

(3) 서비스 요청 당 평균 대기시간 (T_{AW})

$$T_{AW} = AVG\,(T_{TAn} - T_{TA1})$$

T_{TA1}이 단일 서비스요청의 왕복응답시간을 의미하고, 네트워크의 속도가 매우 빠르다는 가정 하에, T_{TA1}을 지연이 없는 순수한 처리시간(processing time)으로 가정하였다. 따라서, $T_{TAn}-T_{TA1}$은 n번째 서비스요청의 왕복응답시간과 순수 처리시간과의 차이 값이 되고, 이것이 n번째 서비스요청의 대기시간이 된다. 실측을 통해 구한 각 구성 방식별 순수 처리시간 (T_{TA1})은 [표 2]와 같다. (각 10회 측정 후 최소 값으로 선정)

표 2. 각 구성별 순수 처리시간

	20MB	40MB	60MB
NAT	3.23 s	6.78 s	10.34 s
TUN	3.28 s	7.69 s	10.91 s
DR	2.90 s	6.56 s	10.91 s

4. 성능평가

4. 1. 평균 응답시간 분석

사용자의 입장에선 각 구성방식의 순수 처리시간보다는 전체적인 응답시간이 더 큰 의미를 가진다. 평균 응답시간의 측면에서 보았을 때, TUN이나 DR 방식에 비해서 NAT방식이 더 나은 성능을 보임을 알 수 있다.

NAT 방식의 경우 클라이언트에게 돌아가는 패킷이 모두 로드밸런서를 통해 넘어가게 되고, TUN과 DR의 경우 하나의 게이트웨이를 통해서 넘어가게 된다. 이 과정에서의 속도 차이가 실제 측정시의 응답시간에 영향을 크게 미친다고 볼 수 있다.

그래프의 증가형태는 이론적으로는 exponential 형태를 보여야 하지만, 서비스요청의 수가 10개로 매우 적은 형태이기 때문에 [그림 2]와 같이 exponential 형태가 약간 나타나기는 하지만, 거의 linear한 증가형태라고 볼 수 있다.

클라이언트 당 서비스요청의 수가 많아지게 될 경우, 클라이언트 내부에서 자체적인 contention이 발생하게 되기 때문에, 본 논문의 실측 과정에서 최대 서비스요청의 수를 10개로 제한하였다. 하지만, 클라이언트의 수가 많아지고, 여러 곳에 분산이 되어있는 경우 서버 자체의 처리능력에 의한 성능차이 뿐만 아니라, 중간에 거치게 되는 장비나, 네트워크 상황 등 성능의 변화에 영향을 줄 수 있는 요소들이 많이 작용하게 되므로, 객관적인 서버의 성능을 측정할 수가 없게된다.

본 측정은 그러한 가변요소를 최소화하기 위해서 서버와 클라이언트를 같은 네트워크 상에 위치시키고, 동일한 사양의 클라이언트를 사용해서, 클라이언트간의 처리능력 차이가 없이 동일한 상황에서 이루어졌다.

4. 2. 시스템 작업 처리량 분석

[그림 3]에서 볼 수 있듯이, 서비스요청의 수가 4-5개 정도까지 증가하는 동안은 점차적으로 서버로부터 전송되는 데이터의 크기가 커지면서 전체적인 시스템의 작업 처리량도 증가하게 된다.

5개 이상의 서비스요청이 존재하는 경우에는, 시스템 작업 처리량이 비슷한 수준으로 유지되는 데, 이는 각 실제 서버들이 클라이언트의 서비스요청을 처리하는 과정에서 발생하는 디스크 I/O 과정에서 contention[3]이 발생하기 때문이라고 볼 수 있다. 서비스요청의 수가 많아질수록 이러한 현상은 심해지게 되고, 이로 인해 각 서비스요청들의 평균 응답시간이 exponential하게 증가하게 된다.

평균 응답시간의 경우와 마찬가지로 시스템 작업 처리량에서도 NAT방식이 대체적으로 다른 방식에 비해 더 나은 성능을 보였다.

스케줄링 알고리즘별 시스템 작업 처리량을 살펴보면, 서비스요청별 응답시간과 마찬가지로, NAT방식과 TUN방식에서는 LBLC 스케줄링 알고리즘, DR방식에서는 LBLCR 스케줄링 알고리즘의 경우 좋은 성능을 보였다. 이 부분도 추후에 많은 수의 서비스요청에 대한 측정을 실시하면 보다 일관성 있는 결과를 얻을 수 있으리라 본다.

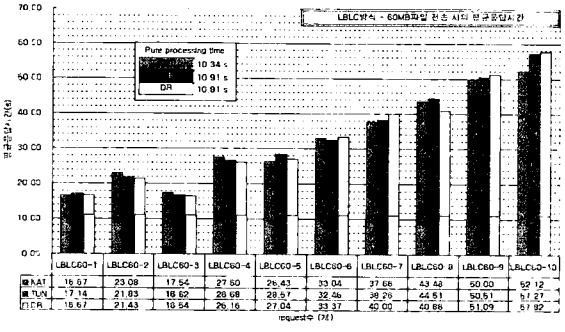


그림 2. LBLC 알고리즘에서 60MB파일 전송시의 서비스 요청별 평균 응답시간

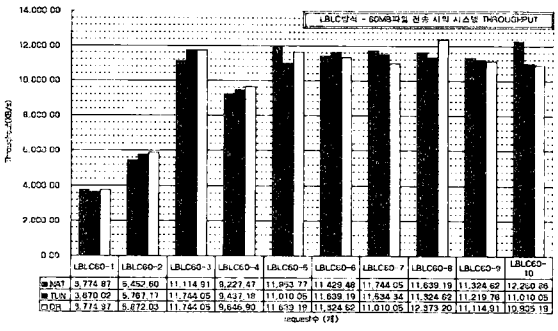


그림 3. LBLC 알고리즘에서 60MB파일 전송시의 시스템 작업 처리량

4. 3. 종합 비교 분석

서비스요청별 평균 대기 시간과, 전체적인 시스템의 작업 처리량을 기준으로 성능을 비교해 보았을 때, 구성 방식 중에는 NAT 방식이 가장 우수한 성능을 보였고, 스케줄링 알고리즘에 있어서는 NAT와 TUN에서는 LBLC 알고리즘이, DR에서는 LBLCR 알고리즘이 가장 좋은 성능을 보였다.

스케줄링 알고리즘 중에서, LBLC와 LBLCR의 경우에는 특정 클라이언트에 대해서 서비스를 해 주는 서버가 거의 변하지 않게 된다. 대부분의 OS에서 특정 파일에 대한 I/O를 수행할 때, 요청된 내용과 관련된 부분을 미리 읽어들이게 된다. 따라서 하나의 클라이언트가 하나의 서버로부터 계속적인 서비스를 받을수록 성능이 더 좋아지게 된다. 멀티미디어 전송의 경우, 파일의 내용을 순차적으로 읽어오는 경우가 많기 때문에, 이러한 과정에서 많은 성능향상을 볼 수 있게 된다. 다른 스케줄링 알고리즘들의 경우 하나의 클라이언트가 서비스요청을 발생할 때마다 서비스를 해주는 서버가 바뀌게 되므로 멀티미디어 전송의 경우에는 LBLC와 LBLCR이 다른 알고리즘에 비해 좋은 성능을 나타내게 된다.

5. 결론 및 추후 연구방향

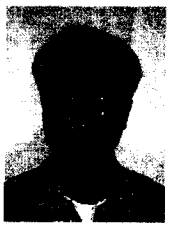
본 논문에서는 인터넷 방송과 같은 멀티미디어 데이터 전송에서 최적의 환경을 제공하는 CDD 시스템의 구현을 위한 과정의 일환으로, LVS를 통한 고 가용성 서버를 구축하고, 각 구성방식별로 성능을 측정하여 보았다.

각 구성 방식별로 보았을 때에는 NAT방식이 TUN이나, DR 방식의 경우에 비해 더 우수한 속도를 보였고, 스케줄링 알고리즘별로 비교를 했을 때에는 여러 가지 제약점으로 인해서 일관적인 결과를 얻을 수는 없었지만, NAT 방식과 TUN방식의 경우 LBLC 스케줄링 알고리즘이 가장 우수한 성능을 보였고, DR방식의 경우 LBLCR 알고리즘에서 좋은 성능을 보였다.

실제적인 멀티미디어 데이터 전송의 경우 많은 클라이언트들을 대상으로 서비스가 이루어지게 되므로, 추후에 보다 광범위한 네트워크를 대상으로 하는 성능 측정을 실시할 필요가 있다. 이 경우, 가변 요인들에 대한 파악과, 해결방안에 대한 연구도 병행되어야 할 것이다.

6. 참고문헌

[1] Wensong Zhang, "Linux Virtual Server Project", http://www.linuxvirtualserver. org, May 1998.
[2] Hrvoje, "GNU Wget:The noninteractive downloading utility",http://www.gnu.org /manual/wget/html_mono/wget.html
[3] Silberschatz, Galvin, "Operating System Concepts, fifth edition", Addison Wesley, pp414-427, February 1998.
[4] Duane Wessels, "SQUID Frequently Asked Questions", http://www.squid- cache.org/Doc/FAQ/FAQ.html, 2001
[5] William Gropp, "Tutorial on MPI: The Message-Passing Interface", http://www-unix.mcs.anl.gov/mpi/tutorial/gropp/talk.html.
[6] Al Geist, "Advanced Tutorial on PVM 3.4 New Features and Capabilities", http://www.csm.ornl.gov/pvm/EuroPVM97/, 1997.
[7] Colajanni, M.; Yu, P.S.; Dias, D.M., "Analysis of task assignment policies in scalable distributed web-server systems", June 1998
[8] Feelamint Networks, "CDN서비스란", http://www.feelamint.com/f_cdn/1-1-1_cdn.html,2002.



박 동 국
2000:서울시립대학교 전자전기컴퓨터공학부 학사
2002:서울시립대학교 전자전기컴퓨터공학부 석사
관심분야:Metacomputing, System Software, Clustering, Load balancing.

이 용 우
1981:서울대학교 공과대학 전기공학과학사
1981:Schlumberger Technical Services Inc. 에서 International Engineer로 호주, 동남아 등에서 근무
1982--1998:과기부/정통부기관이었던 KIST /KAIST에서 선임연구원으로 근무.
영국 Edinburgh대학 Computer Science Dept. 석사(월반),박사.
1998:교육부기관이었던 KMEC에서 책임 연구원으로 근무.
1999-현재:서울시립대학교 전자전기컴퓨터 공학부에서 근무.
관심분야: Metacomputing, CSCW, 시스템 소프트웨어, 초고속 통신, 가상학교, 성능평가공학 등등.