제18회 한국정보처리학회 추계학술발표대회 논문집 제9권 제2호 (2002.11)

LVS 클러스터 시스템의 요구 패턴에 따른 부하 분산 알고리즘 분석

이선홍*, 김성기, 나용희, 민병준
인천대학교 컴퓨터공학과
e-mail:{lishanhong, proteras, ramen, bjmin}@incheon.ac.kr

Analysis of the Load Balancing Algorithms according to the Request Patterns on the LVS Cluster Systems

Shan Hong Li*,Sung Ki Kim, Yong Hee Na, Byoung Joon Min
Dept. of Computer Science, Incheon University

요 약

갈수록 증가하는 인터넷 사용자의 서비스 요구량에 대처하기 위해, 부하 분산 기능을 갖는 클러스터 시스템의 이용이 늘어가고 있다. 본 연구에서는 클라이언트에게 보다 향상된 응답 성능을 제공하기 위해 사용되는 RR(Round Robin), WRR(Weighted Round Robin), LC(Least Connection), WLC(Weighted Least Connection) 부하 분산 알고리즘에 대해서, 클라이언트로부터 인입되는 7 가지 요구 수신 패턴에 따른 부하 분산 응답 특성을 분석하고 그 결과를 논한다. 이를 위해, 실제 시스템의 측정 결과를 토대로 단위 시간 당 인입되는 클라이언트의 요구량 변화를 7 가지 패턴으로 분류하였고, 리눅스 가상 서버(LVS: Linux Virtual Server) 클러스터 시스템을 대상으로 7 가지 요구 패턴에 대한 부하 분산 응답 특성을 얻었다. 본 연구를 통해서 클라이언트 요구량 변화 패턴에 따른 최적의 부하 분산 알고리즘을 제시할 수 있었다. 본 연구 결과는 향후 효율적인 동적 부하 분산 연구에 좋은 참고가 될 것이다.

1. 서론

인터넷이 대중화되면서 사용자의 인터넷 사용시간과 서비스 요구량이 증가함에 따라 서비스를 제공하는 서버의 부하 문제를 해결하려는 노력이 다방면에서 요구되고 있다. 이러한 노력의 일환으로 동일한 컨텐츠를 담은 다중 서버를 하나의 클러스터로 묶어, 부하를 분산시켜 서비스하는 클러스터 시스템의 사용이 널리 보급되고 있다. 이러한 클러스터 시스템의 일반적인 구조는 클라이언트로부터 요구를 수신하는 부하 분산(load balancing) 노드와 실제 서비스를 수행하는 real 서버 군으로 구성된다. 모든 노드는 클라이언트에게 대표하는 가상 IP 주소를 공유하고 각자의 실제 IP 주소를 가지고 서로를 인식한다. 이때 클라이언트로부터 인입되는 요구를 부하 분산 노드에서 작업 서버(real server)에게 분산하는 대표적인 방법으로는 RR(Round Robin), WRR(Weighted Round Robin), LC(Least Connection), WLC(Weighted Least Connection) 알고리즘이 있다[1,5].

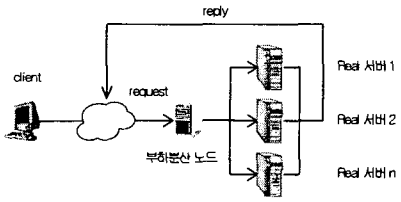
이러한 부하 분산 알고리즘의 목적은 클라이언트 요구를 작업 서버에게 고르게 분산함으로써 클라이언트에 대한 서버의 응답 성능을 높이고 QoS 를 좋게 하는 것이다. 그러나 이들 4 가지 부하 분산 알고리즘은 단순히 작업 서버의 물리적 성능 사양과 연결 수에 기초하여 클라이언트 요구를 분산하고 있다[2]. 이점을 보완하여, 작업 서버의 프로세서, 메모리 등의 컴퓨팅 자원에 부과된 부하를 반영함으로써, 동적으로 부하 분산하는 방법도 있지만, 모두 시스템에 인입되는 클라이언트 요구량 변화에 따른 부하 분산 응답 특성을 간과하고 있다. 실제 웹 서버의 경우, 수신하는 요구량(request/sec)이 많아질수록 웹 서버의 처리율(connection/sec)이 저하되며, 짧은 시간에 요구량이 폭주하는(bursty condition) 경우에는 그러한 성능 저하가 크게 나타난다[4]. 따라서, 특정 요구 패턴에서는 현재의 서버가 받는 부하나, 연결 수 등을 부하 분산에 반영하는 자체가 오버헤드로 작용할 수 있다.

본 연구에서는 일정 처리 시간을 요하는 서비스 수행 코드를 작업 서버에 준비해두고, 단위 시간당 인입되는 클라이언트 요구량 변화에 따라 4 가지 알고리즘에 대한 응답 특성을 조사하여 분석한다. 이를 통해, 실시간으로 인입되는 클라이언트 요구량 변화에 어떤 부하 분산 알고

리즘이 최적인지를 제시한다. 2 장에서는 실험 환경과 4 가지 알고리즘에 대하여 간략히 소개하고, 3 장에서 는 7 가지 요구량 패턴에 대한 모델을 제시하고, 특성 분석 방법에 대해서 논한다. 4 장에서는 실험을 통해 얻은 결과 분석을 논한 다음, 5 장에서 결론을 맺는다.

2. 관련 연구

2.1 리눅스 LVS 클러스터 시스템



(그림 1) 리눅스 LVS 클러스터 시스템 환경

본 연구에서는 (그림 1)과 같이 리눅스 기반의 LVS 클러스터 시스템 환경에서 일정 시간을 요하는 웹 기반 서비스 수행 코드를 각 작업 서버에 중복시키고 Apache 웹 서버를 실행하였다. (그림 1)에서 부하 분산 노드와 각 작업 서버들은 자신의 실제 IP 주소를 가지면서 LVS 클러스터 시스템 환경에서 설정된 가상 IP 주소를 공유한다. 전체적인 서비스는 가상 IP 주소를 통하여 이루어지는데, 클라이언트는 이 가상 IP 주소로 요구 메시지를 보낸다. 클라이언트로부터 수신된 요구 패킷은 부하 분산 노드에서 IP Tunneling 기술을 이용하여 작업 서버에게 재지향(redirect) 된다. 이때, 부하 분산 노드는 정해진 부하 분산 알고리즘에 따라 재지향 될 작업 서버를 선택하고 자신의 해쉬 테이블에 선택된 작업 서버에 대한 연결을 기록한다. 재지향된 요구 패킷은 작업 서버에서 처리된 후, 부하 분산 노드를 거치지 않고, 곧 바로 요청한 클라이언트로 전송된다.

본 연구에서는 LVS 클러스터 시스템의 부하 분산 알고리즘으로 RR(Round Robin), WRR(Weighted Round Robin), LC(Least Connection), WLC(Weighted Least Connection) 알고리즘을 적용하였다[1,5].

2.2 부하 분산 알고리즘

가. RR(Round Robin) 알고리즘

부하 분산 방법이 단순히 들어온 요구를 순서대로 번갈아 가면서 작업 서버에게 분산하는 방법이다. 작업 서버의 물리적 성능을 고려하지 않는다.

나. WRR(Weighted Round Robin) 알고리즘

물리적 성능이 우수한 작업 서버가 좀더 많은 요구를 처리하도록 작업 서버마다 가중치를 두고 분산하는 방법이다.

다. LC(Least Connection) 알고리즘

- 서버의 현재 연결 수를 기준으로 연결이 더 적은 서버에게 다음 요구를 할당하는 방법이다. 작업 서버의 물리적 성능 차이를 고려하지 않는다.

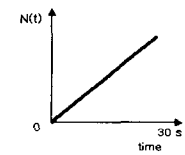
라. WLC(Weighted Least Connection) 알고리즘

현재 상태에서 연결 수가 최소인 작업 서버에게 다음 요구를 할당하되 작업 서버의 물리적 성능을 고려하여 분산하는 방법이다. 현재의 연결 수와 작업 서버의 물리적 성능에 따라 설정된 가중치를 계산해 다음 요구를 분산하는 오버헤드가 있다.

3. 요구량 변화 모델 및 실험 방법

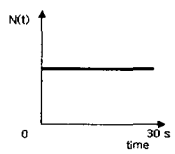
3.1 요구량 변화 모델

시스템에 시시각각 들어오는 클라이언트 요구량의 변화에 따라 4 가지 부하 분산 알고리즘의 응답 특성을 알아보기 위해서는 먼저 실제 시스템을 대상으로 시간에 따라 어떤 유형의 클라이언트 요구 패턴이 들어오는지에 대한 조사가 필요하다. 본 연구에서는 네트워크에서 수신되는 요구 패턴 분석에 대한 [2,3]의 결과를 토대로 다음 (그림 2)와 같은 7 가지 요구 패턴을 모형화 하였다. 각 요구 패턴은 임의의 시간 동안 시스템이 수신하는 단위 시간당 요구량(N(t))을 의미한다.



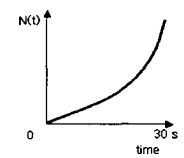
요구 패턴 1

$N(t) = at \ (0 < t < 30)$



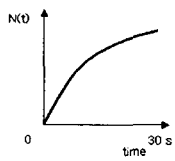
요구 패턴 2

$N(t) = a \ (a > 0)$



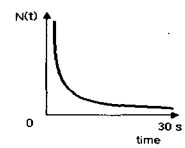
요구 패턴 3

$N(t) = at^2 \ (a > 0)$



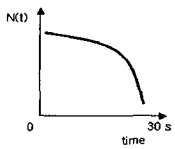
요구 패턴 4

$N(t) = \log_a t$



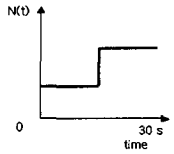
요구 패턴 5

$N(t) = a / t \ (a > 0)$



요구 패턴 6

$N(t) = \sqrt{a^2 - t^2} \ (a > 0)$



N(t) = a (0 < t < 15)
N(t) = b (15 < t < 30)
(a, b > 0

요구 패턴 7

(그림 2) 단위 시간당 인입되는 요구량 패턴

3.2 요구 생성 도구

본 연구에서는 30초 동안 매 초 마다 (그림 2)에 표현된 각 패턴의 요구량을 발생시키는 요구 생성 툴을 구현하였다. Java 언어를 이용하여 구현한 요구 생성 툴은 클러스터 시스템으로 요구 메시지를 전송하는 가상 클라이언트로 사용하였다.

3.3 리눅스 LVS 클러스터 시스템 환경

다음은 실험에 사용된 컴퓨팅 자원이다.

구분	Processor	Memory	Network
작업 서버 1	펜티엄 2 400 MHz	192 MB	100 Mbps
작업 서버 2	펜티엄 3 600 MHz	192 MB	100 Mbps
부하 분산 노드	펜티엄 4 1.7 GHz	256 MB	100 Mbps

3.4 응답 특성 측정 방법

앞 절에서 소개한 요구 생성 툴을 이용하여 요구 메시지를 전송하기 전에, 작업 서버에 일정한 서비스 수행시간이 소요되는 코드를 중복 시켜 두고, 실제 서버의 작업 처리시간을 매 요구 때마다 데이터베이스에 저장하게 하였다. 데이터베이스는 각 작업 서버에 위치하며 데이터베이스 안에는 log라는 table이 존재한다. log table은 각 패턴별로 수신된 모든 요구 에 대한 처리 시간을 저장하며, SQL query를 이용하여 평균 응답 시간을 구할 수 있다. 다음 (표 1)은 log table의 구조이다.

log table
id
curr_time
exec_time

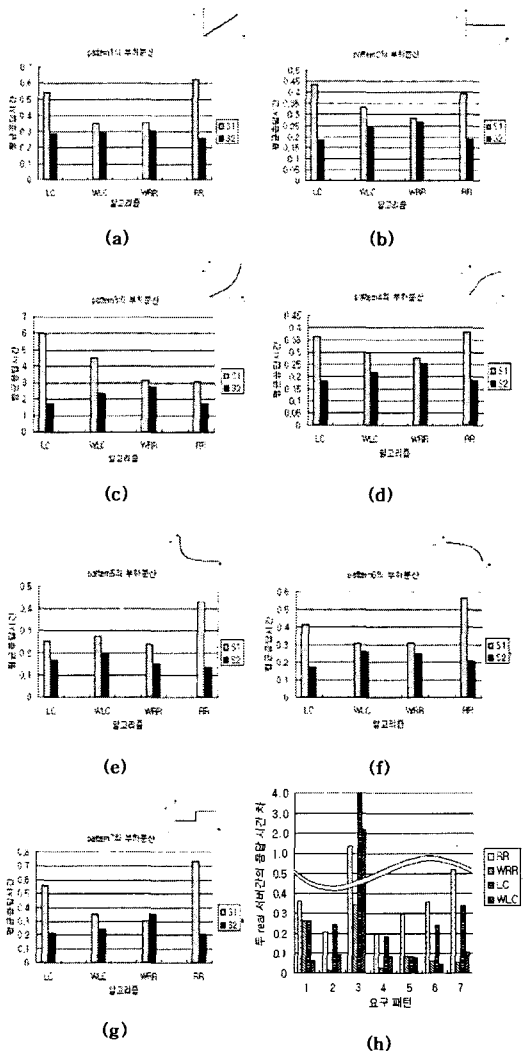
(표 1) log table 구조

- id : 30초 동안에 접속된 request의 순서
- curr_time: 접속한 시간
- exec_time: 한 request 에 대한 처리 시간

본 연구에서는 클라이언트 요구에 대한 응답시간을 서버가 서비스 수행코드를 실행한 시간과 결과를 전송한 시간으로 볼 때, 클라이언트로 결과가 전송되는 시간은 동일한 실험 조건이므로 무시하였다. 따라서, 클라이언트에 대한 응답 시간을 서버의 서비스 수행 완료시간으로 계산하였다.

4. 분석 결과

앞장에서 소개한 요구 생성 툴을 이용하여 LVS 클러스터 시스템의 4 가지 부하 분산 알고리즘의 응답 특성을 얻었다. 다음 (그림 3)은 7 가지 각 패턴별로 4 가지 알고리즘의 응답 특성 결과이다.



(그림 3) 7 가지 패턴별 부하분산 응답 특성

(그림 3)의 각 요구 패턴 별 결과는 LVS 클러스터 시스템에서 30 초 동안 4 가지 부하 분산 알고리즘의 응답 시간을 측정한 후, 각 측정값에 대해 30 초 동안의 평균을 구한 결과이다. 각 패턴 별로 응답 특성을 살펴보면, 그림 (a)의 패턴과 같은 요구량 변화에서는 서버의 물리적 사양과 현재의 연결 수를 근거로 가중치를 적용한 WRR과 WLC 알고리즘의 응답 성능이 두 작업 서버간의 차이

가 없이 좋은 응답 특성을 보였다. 이것은 가중치에 의해 두 서버의 물리적 성능과 연결 수를 고르게 분산 시켰기 때문인 것으로 판단된다. 또한, 동일한 조건에서 서버의 물리적 사양이 떨어지는 작업 서버 1 과 물리적 사양이 상대적으로 좋은 작업 서버 2 의 응답 성능의 차이를 비교 할 때, RR 과 LC 이 WRR 과 WLC 에 비해서 이들 두 서버의 응답 성능 차이가 큰 것은, 단위 시간당 요구량의 증가가 서버의 프로세서 성능에 민감하며, 이는 동시에 연결 수에도 영향을 주는 것으로 판단 할 수 있다. [4]에서는 웹 서버가 일정한 요구 수신율(request/sec)에 도달하면 요구 메시지 처리보다 프로토콜 처리에 사용하는 프로세서 이용률이 높아지는 특성 때문에, 웹 서버의 성능 한계점(saturation point)에 도달하고, 그 결과 웹 서버의 요구 메시지 처리 능력(connection /sec)이 서서히 저하되는 특성을 지적하고 있다. 따라서, 프로세서 물리적 성능을 고려하지 않고 부하를 분산한 RR과 LC는 두 작업 서버간의 응답 성능에 차이가 상대적으로 큰 것으로 나타났다. 이러한 양상을 띄우는 결과는 그림 (b), (d), (f), (g)에서도 비슷하게 나타나며, 단위 시간당 인입되는 요구량의 변화에서 그 기울기가 클수록 가중치를 적용한 부하 분산 알고리즘과 그렇지 않은 알고리즘 간의 응답 성능의 차이는 더 뚜렷해짐을 알 수 있다.

그림 (c)와 같은 패턴의 폭주 조건(bursty condition)에서는 앞에서 논의한 분석과는 다르게, WRR 그 다음 RR 알고리즘이 최적의 부하 분산 알고리즘을 보여준다. 이것은 폭주 조건에서 현재의 연결 수와 연결 수에 가중치를 재계산하여 부하 분산에 반영하는 그 자체가 오버헤드로 작용하는 것으로 판단할 수 있다. 오히려 서버의 물리적 성능에 가중치를 두고 부하 분산시킨 WRR이 가장 응답 시간의 편차가 적으면서 빠른 응답 특성을 보이고 있다. 특히 그림(c)의 실험에 있어서 일정수의 수신 요구량(request/sec)에 도달하면 더 이상의 연결이 허용되지 않았음을 확인 할 수 있었다. 이는 LVS 클러스터 시스템의 부하 분산 기능의 일면이기보다는 웹 서버의 특성인 것으로 본다. 실제로도 본 연구에서, LVS 클러스터 시스템이 각 패턴에 대한 부하 분산 알고리즘을 수행하는데 걸리는 부하를 측정한 결과, 추가적인 별다른 부하가 발생하지 않았다.

그림(e)의 요구 패턴과 같이 초기 요구량이 많으면서 현격히 그 요구량이 감소하는 경우에는, 단순히 들어오는 순서대로 요구 메시지를 분산시키는 RR을 제외하고는 나머지 부하 분산 알고리즘의 응답 특성이 두 작업 서버간의 응답 성능의 차이 없이 고르게 나타났다.

그림(h)는 두 작업 서버간의 평균 응답 시간의 차이를 각 알고리즘별로 도시하고 있다. 그림(h)는 그림(c)와 같은 폭주 조건을 제외하고는 전 알고리즘에서 WLC 알고리즘이 두 작업 서버간의 응답 시간 차이가 가장 적은 부하 분산 알고리즘인 것을 보여 주고 있다.

이상의 분석 결과는 효율적인 동적 부하 분산 알고리즘 설계에 있어서 2 가지 요구사항을 제시하는데, 첫 째는 기존의 부하분산 방법에 추가적으로 실시간으로 변화하는 수신 요구량의 변화에 대해서도 탄력적으로 적응하는 부하 분산 방안 연구가 필요하다는 것이다. 또 하나는 클라이언트에 대한 QoS 제공 측면에서, 서버의 부하를 고르게 분산한다는 것이 클라이언트 응답 특성을 전적으로 개선하는 것이라고 볼 수 없다는 것이다. 이는 부하 분산 방법도 중요하지만 클라이언트 요구에 응답하는 응용의 특성이 부하 분산 방안에 반영되어함을 의미한다.

5. 결론

본 논문에서는 LVS 클러스터 시스템 환경에서 RR, WRR, LC, WLC 알고리즘의 응답 특성을 7 가지 요구 패턴에 따라 분석해 보았다. 이를 위해, 실제 시스템에서 네트워크를 통해 들어오는 요구량 변화를 7 가지로 모델링하였고, 이 모델들을 반영할 수 있는 가상 클라이언트로서 요구 생성 도구를 java 언어로 구현하였다.

구현된 요구 생성 도구를 이용해, 단위 시간당 인입되는 요구량 변화에 따라, 4 가지 부하 분산 알고리즘의 부하분산 응답 특성을 얻었다. 실험을 통해 얻은 결과를 분석하고, 어떤 요구량 변화 패턴에서 어느 알고리즘이 최적인지를 제시하였다.

본 연구 결과는 효율적인 동적 부하 분산 알고리즘 설계에 좋은 참고가 될 것이다.

참고문헌

[1] Wensong Zhang,Shiyao Jin, Quanyuan Wu. *"Creating Linux Virtual servers"*, http://linux-vs.org /linuxexpo.html, February 1999.

[2] Kevin Thompson, Gregory J.Miller, *"Wide-Area Internet Traffic Patterns and Characteristics"*, IEEE Network, November/December 1997.

[3] Paul Barford, Mark Crovella, *"Generating Representative Web Workloads for Network and Server Performance Evaluation"*, In SIGMETRICS 98/Performance 98 Joint International Conference (Madison, WI, USA, June 1998).

[4] Gaurav banga and Peter Druschel. Measuring the capacity of a web server. In Proceedings of the USENIX Annual Technical Conference. Monterey. CA, December 1997.

[5] Joseph Mack, "LVS-HOWTO," http://www.linuxvirtualserver.org/Joseph.Mack/HOWTO/LVS-HOWTO.html. 1999-2002, released under GPL jmack(at)wm7d(dot)net