# 리눅스 클러스터 웹 서버에서 스케줄링 기법 성능비교 및 분석

이규한° 이종우
한림대학교 컴퓨터공학과
khlee@center.cie.hallym.ac.kr jwlee44@hallym.ac.kr

# Scheduling technique and Performance Evaluation in Linux Cluster Web Server

Kyuhan Lee° Jongwoo Lee
Dept of Computer Engineering, Hallym University

**요 약**

컴퓨터의 급속한 보급과 인터넷의 사용이 급증하면서 사용자에게 좀 더 나은 서비스를 제공하기 위해 고성능 웹서버가 필요하게 되었다. 따라서 최근에는 리눅스 가상서버를 이용한 클러스터 웹 서버의 구축 사례가 늘고 있다. 이는 기존의 단일 서버 시스템에 비해 가격이 저렴하면서도 지속적인 서비스를 할 수 있는 가용성과 서버의 수를 쉽게 늘릴 수 있는 확장성을 갖고 있다. 오픈소스인 리눅스와 여러 대의 저렴한 PC들을 이용하여 서버로 들어오는 부하를 분배서버가 여러 실제 서버에게 분산시킴으로써 웹 서버의 성능을 향상시키고, 서버의 가용성을 높이는 것이 클러스터 웹 서버의 목적이라 할 것이다. 본 논문에서는 서버에 들어오는 요청을 실제 서버들에게 분산시켜주는 스케줄링 알고리즘들을 알아보고 각각의 성능을 비교, 분석하였다. 그 결과 정적 스케줄링 알고리즘보다는 각 실제 서버의 부하를 고려한 알고리즘이 더 좋은 성능을 보인다는 것과 캐쉬 친화적 알고리즘이 캐쉬를 고려하지 않은 알고리즘 보다 더 좋은 성능을 보인다는 것을 알 수 있었다. 가장 중요한 결론으로는 알고리즘의 성능보다는 노드 개수 증가율이 클러스터 웹 서버 성능에 더 큰 영향을 미친다는 것을 알 수 있었다.

## 1. 서 론

최근 인터넷의 급속한 확산으로 웹서버의 부하는 늘어나고 있는 실정이다. 이러한 부하들을 여러 서버에 나눔으로써 서버의 성능향상을 목적으로 하는 것이 클러스터 웹서버이다. 인터넷의 영역이 확대되고 사용자수 또한 늘어남으로써 이러한 클러스터 웹서버의 필요성은 더해지고 있다[1, 2].

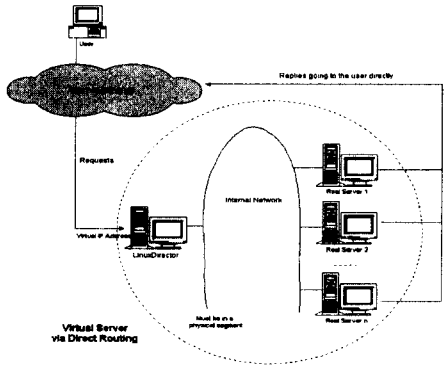
기존 단일 서버 시스템의 단점은 고가의 가격과 가용성을 보장할 수 없다는 것이다. 이러한 단점을 보완한 것이 클러스터 웹서버인데 이는 여러 대의 값 싼 PC들을 연결하여 서버로 들어오는 요청을 로드 밸런서가 각각의 서버들에게 스케줄링 알고리즘을 통해 분산시키는 구조를 갖는다. 일반적인 클러스터 웹서버의 대표적인 특징은 고가용성과 확장성을 들 수 있다[1]. 첫째로 고가용성이란 사용자 요청이 급증하여 더 이상 요청을 받을 수 없는 상태가 되거나 서버가 다운되었을 때, 다른 서버가 로드 밸런서의 역할을 대신하여 사용자로부터의 요청을 실제(real) 서버에게 보내줌으로써 서버의 서비스가 중단되지 않는 특성을 말한다. 두 번째로 확장성이란 클러스터 내에 새로운 노드의 삽입이 쉬워 서버 추가가 용이하다는 점이다. 로드 밸런서의 실제 서버 리스트에 한 줄만을 추가시킴으로써 클러스터의 확장이 가능하다. 이러한 장점과 가격 면에서의 경쟁력을 갖추고 있기 때문에 최근 클러스터 웹 서버 구축사례는 늘고 있다. 본 논문에서는 리눅스 클러스터 웹 서버를 구축하고 요청 분배 알고리즘을 살펴본 후, 각 알고리즘에 따른 웹 서버의 성능을 비교, 분석한 결과를 제시하고자 한다.

## 2 관련 연구

리눅스 클러스터를 구성할 때 가장 널리 사용되는 패키지는 리눅스 가상 서버(Linux Virtual Server, 이하 LVS)[1]이다. LVS는 소프트웨어적으로 H/W Layer-4 스위치를 구현한 것이라 할 수 있다. 실제 서버는 LAN이나 WAN에 연결되어 있고, 실제 서버의 전면에 로드 밸런서를 두어 이로 하여금 요청을 분배하도록 한다. LVS에서 클러스터 환경을 구축하는 방식에는 NAT(Network Address Translation)와 IP 터널링, 직접 라우팅 이렇게 3가지가 있다[1]. NAT 방식에서는 클라이언트로부터 들어오는 요청 패킷을 로드 밸런서가 실제 서버에게 전송해주고 그 결과를 돌려 받아 클라이언트에게 전송해준다. 때문에 로드 밸런서의 병목 현상이 가장 큰 약점으로 작용한다. IP 터널링과 직접 라우팅 방식에서는 이런 약점을 보완하였는데, 로드 밸런서는 단순히 들어오는 요청 패킷을 실제 서버에게 전송만 해준다. 실제 서버는 이를 처리하여 결과를 클라이언트에게 직접 전송해준다. 때문에 로드 밸런서에서의 병목 현상은 나타나지 않는다. 따라서 대부분의 클러스터 웹 서버에서는 실제 서버가 클라이언트에게 직접 서비스하는 IP 터널링 이나 직

접 라우팅 방식을 사용하고 있다.

### 2.1 LVS를 사용한 클러스터 구축




[그림 1] 직접 라우팅을 통한 LVS 구성도

실험에 사용할 클러스터 웹 서버를 구축하기 위해 본 논문에서는 우선 LVS를 사용하여 클러스터 시스템을 구축하였다. 또한 요청 패킷을 지정된 스케줄링 알고리즘에 따라 선택된 실제 서버에게 직접 라우팅하는 직접 라우팅 방식을 사용하였다. [그림 1]에서 LinuxDirector라고 표시된 로드 밸런서는 클라이언트로부터 들어오는 요청을 스케줄링 알고리즘을 통해 각 실제 서버에게 분산시켜주며 그 요청은 실제 서버에 의해 처리된 후, 클라이언트에게 직접 전달된다[3].

### 2.2 스케줄링 알고리즘

본 논문에서 평가하고자 하는 요청 분배 스케줄링 알고리즘은 다음과 같다.

● RR(Round-Robin) : 실제 서버들에게 한 번씩 요청을 할당, 실제 서버들의 활성 접속 수나 응답 시간을 고려하지 않는다. 실제 서버가 ABC라면 ABCABC 의 순서로 요청이 분배된다.

● WRR(Weighted RR) : 실제 서버들에게 서로 다른 처리용량을 지정하여 그에 따라 RR방식으로 할당한다. A(3)B(2)C(1) 일 때(괄호안이 가중치), ABCABA 의 순서로 분배된다.

● LC(Least Connection) : 동적 스케줄링 알고리즘의 하나로 활성 접속수가 가장 적은 실제 서버에게 요청을 할당하는 방법이다. 하지만 서로 다른 성능의 실제 서버들로 클러스터를 구성했을 때에는 성능은 고려하지 않고 활성 접속 수만을 고려하여 스케줄링 하므로 실제로 효율적이지 못하다. 처리량이 A(4), B(16)의 성능을 갖고 있을 때, 현재 활성 접속 수가 A는 4, B는 5라면 들어오는 요청은 포화 상태인 A로 스케줄링 될 것이다.

● WLC(Weighted LC) : 실제 서버들에게 가중치를 지정하여 가중치에 따라 활성 접속수가 적은 서버에 요청을 할당한다. 기본가중치는 1이며 LC보다 부가적인 배분작업이 필요하다. 실제 서버의 활성 접속 수를 가중치로 나눈 값이 최소인 실제 서버에게 스케줄링 된다.

● LBLC(Locality-Based LC) : 요청은 어떤 한 실제 서버가 포화될 때까지 계속 한 서버로만 전달된다. 실제 서버가 포화되면 그렇지 않은 실제 서버들을 WLC 방식으로 스케줄링 하여 또 다시 선택된 실제 서버로만 요청이 전달된다. 한 실제 서버내의 캐쉬를 최대한 활용할 수 있다는 장점이 있다.

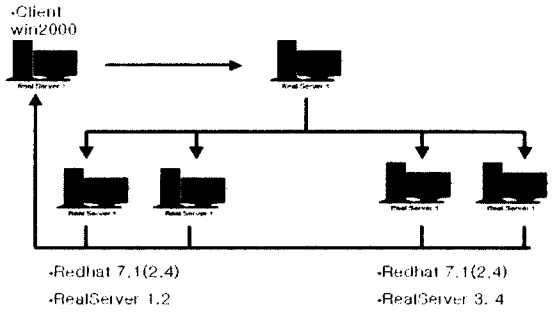
● LBLCR(Locality-Based LC with Replication) : 실제 서버들을 몇 개의 그룹으로 묶은 후, 현재 실제 서버 그룹 내의 서버가 모두 포화상태가 되면 LC방식으로 다른 서버 그룹 내의 실제 서버에게 요청을 할당하고, 그 실제 서버를 현재 모두 포화된 실제 서버 그룹으로 포함시킨다. 포화 그룹의 상태가 일정기간 이상 안정되면 재그룹하여 원상복구 한다. 실제 서버의 수가 매우 많은 클러스터에서 유용하다.

● DH(Destination Hashing) : 패킷들의 목적지 IP주소들로 할당된 해쉬 테이블의 통계 분석을 통하여 실제 서버로 스케줄링 된다.

● SH(Source Hasing) : 패킷들의 출발지 IP주소들로 할당된 해쉬 테이블의 통계 분석을 통하여 실제 서버로 스케줄링 된다.

### 3. 실험 환경

본 논문에서 사용한 실험 환경은 [그림 2]와 같이 로드 밸런서 한 대와 4대의 실제 서버, 한 대의 클라이언트로 구성된 클러스터 웹 서버이다.



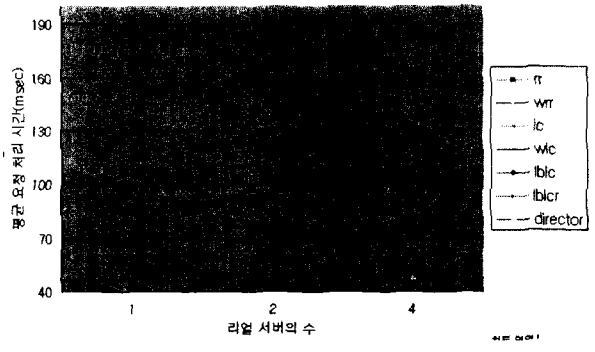

[그림 2] 실험을 위한 클러스터 웹 서버

실험에 사용된 로드 밸런서에는 P3 933MHz dual, 512M RAM 과 8.6GB 하드디스크, 100Mbps Ethernet Card 2개가 장착되어 있다. 실제 노드에는 P3 650MHz, 128M RAM 과 10GB 하드디스크, 100Mbps Ethernet Card 1개씩이 장착되어 있다. 로드 밸런서와 실제 노드들은 스위칭 허브를 통해 연결되어 있다. 클라이언트에만 10Mbps LAN Card가 장착되어 있다. 각 서버 노드에 설치된 리눅스커널은 2.4.2 이며 웹 서비스를 위해 아파치 1.3.19를 사용하였다. 로드 밸런서 기계에는 LVS를 위한 ipvs-0.2.7 패키지를 설치하였다. 클라이언트 운영체제는 Win 2000 Professional을 사용하였다.

실험에서는 앞서 제시한 8개의 요청 분배 스케줄링 알고리즘 중 목적지/소스 해쉬을 제외한 6가지 방식만을 성능 평가 대상으로 하였는데, 이는 클라이언트를 한 대만 사용했기 때문이다. 웹 서버의 성능을 평가하기 위한 작업부하(workload) 발생기로서는 Web server stress tool V4.4를 클라이언트에서 사용하였다. Web server stress tool은 초당 700-1000 개의 HTTP 요청을 생성한다.[4] 이는 평균 요청 처리시간과 평균 사용자 대기시간을 그래프와 텍스트 값으로 출력해주는데, 본 연구에서는 평균 요청 처리 시간의 값을 성능평가의 기준으로 삼았다.

## 4. 성능 평가 및 분석

[그림3]에서 director라고 표시된 것이 단일 웹 서버(standalone)일 때의 처리시간을 나타내는데 실제 서버의 수와는 관계없이 각 알고리즘과 비교를 위해 표시하였다. 실제 서버의 수가 1개일 때는 director의 처리 시간으로 일괄 처리했는데 이 또한 비교를 위한 것일 뿐 실제 처리 시간은 아니다.




*[그림 3] 클러스터 노드 수에 따른 평균 요청 처리 시간*

실제 서버의 수를 2대로 설정했을 때에는 LC 계열 알고리즘들간의 성능 차가 거의 나타나지 않는다. 다만 RR에 비해 LC 계열의 알고리즘이 좀 더 좋은 성능을 보이고 있다. 실제 서버수가 2대일 때에는 전체 알고리즘들이 director 보다 4.7% ~ 14.6%의 성능 향상만을 보였고, 실제 서버수가 4대 일 때는 director 보다 46.4% ~ 77.1%의 증가율을 보였다. 실제 서버수가 2대일 때와 4대일 때를 비교하면 4대일 때 lblc가 63%로 가장 큰 성능 향상을 보였으며 wlc가 37.6%로 상대적으로 가장 적은 성능 향상을 보였다. wlc의 성능이 lc보다 약간 떨어지는 이유는 실제 서버들의 가중치(weight)에 따른 부가적인 배분작업이 있기 때문이다[5]. 따라서 실제 서버들의 성능이 동일한 경우에는 가중치를 고려한 알고리즘이 효과적이지 않음을 알 수 있다. 실제 서버가 4대일 때의 평균 요청 처리 시간을 순서대로 살펴보면, lblc, (wrr, lblcr, lc, rr), wlc, director의 순으로 나타났다. 지금까지의 실험 결과를 분석해 본 결과 다음과 같은 사실을 확인할 수 있었다.

첫째, 클러스터 내의 노드 수가 적을 때에는 캐쉬 친화적 요청 분배 알고리즘인 LC 계열의 알고리즘이 RR 계열의 알고리즘보다 좋은 성능을 보인다는 것이다. 이는 서버 수가 제한된 환경에서는 스케줄링 알고리즘이 클러스터 웹 서버의 성능에 큰 영향을 미친다는 것을 의미한다. 둘째, 클러스터에 소속된 실제 서버들의 성능이 동일(homogeneous)한 경우에는 가중치를 고려한 알고리즘이 성능에 오히려 악영향을 미친다는 점이다. 이는 가중치를 고려한 요청 분배 알고리즘이 실제 서버들의 성능 차로 인한 오버헤드를 줄이기 위한 것이라는 측면을 고려할 때 당연한 결과라 하겠다. 셋째, 클러스터 웹 서버에서 작업 분배 알고리즘의 성능보다는 실제 서버의 수가 전체 성능에 가장 큰 영향을 미치는 요소라는 사실이다. rr 알고리즘의 경우에서 이 같은 사실을 확인할 수 있는데, 2대의 실제 서버 환경에서 성능이 가장 좋지 않았던 rr 알고리즘이 서버의 개수를 늘리자 성능이 급격히 좋아졌음을 확인할 수 있었다. 따라서, 본 논문에서 사용한 실험 환경을 기준으로 판단하건데, 동일한 성능을 갖는 소규모 클러스터 웹 서버 환경에서는 캐쉬 친화적 스케줄링 기법을 사용함과 동시에 가중치를 고려하지 않는 알고리즘을 사용하는 것이 최적의 성능을 보인다고 할 것이다.

## 5. 결론 및 향후 연구 과제

본 논문에서는 LVS-리눅스가상서버를 이용한 클러스터 웹 서버를 구현하고 6개의 스케줄링 알고리즘들의 성능을 비교해 보았다. 단일 서버 시스템보다 2대의 실제 서버, 4대의 실제 서버를 구축했을 때, 더 나은 성능 향상을 얻을 수 있었으며, 실제 서버의 활성 접속 수를 기반으로 요청 분배를 스케줄링 하면서 캐쉬 친화적인 lblc가 가장 좋은 성능을 나타냄을 알 수 있었다. 또한 실제 서버 개수가 알고리즘보다 전체 성능에 더 큰 영향을 미친다는 것을 알 수 있었다.

향후에는 실제 서버의 수를 계속적으로 증가시키면서 그 성능 향상을 알아보고, 각 스케줄링 알고리즘을 자세히 분석하여 상황에 따라 더 좋은 성능을 보일 수 있는 스케줄링 알고리즘을 연구, 개발할 것이다. 특히 HTTP 프로토콜을 인식하는 스케줄링 알고리즘을 연구할 계획이다.

## 6. 참고 문헌

[1] Wensong Zhang and et al. Linux Virtual Server Project. http://www.linuxvirtualserver.org
[2] EnCluster-WhitePaper, http://www.clunix.com/support
[3] Joseph Mack, "LVS-mini-HOWTO", http://www.linuxvirtualserver.org
[4] Webserver stress tool V4.4 http://www.web-server-tools.com/tools/
[5] Patrick O'Rourke, Mike Keefe, "Performance Evaluation of Linux Virtual Server", http://www.linuxvirtualsrver.org