클러스터 시스템에서의 작업분배 방법 비교
Comparison of Load Balancing Algorithms in Cluster System

안창원, 임은지, 정성인, 차규일
ETRI, 리눅스연구팀, {ahn, ejlim, sijung, gicha}@etri.re.kr

Abstract

인터넷의 사용자가 급증하면서 고가용성(High Availability)과 확장성(Scalability)를 지닌 고성능 인터넷 서버들이 요구되고 있다. 클러스터 시스템은 이러한 요구사항을 만족시킬 수 있는 시스템으로 실제 작업을 처리하는 서버(Server)와 시스템 외부에서 유입되는 사용자 요구를 시스템에 속한 서버 노드에 분배하는 디렉터(Director)로 이루어져 있다.

디렉터는 사용자가 요구한 작업을 서버노드에 분배하는 로드밸런서(Load-Balancer)의 역할을 하게 되는데, 작업을 분배하는 방법으로는 Round-Robin, Least-Connection 등의 방법이 주로 사용되고 있다.

본 논문에서는 각 작업분배 방법 중 효과적인 방법을 비교 선택하기 위해 대기이론을 적용하여 시뮬레이션을 통해 검증하였다.

1. 서론

인터넷의 사용자가 폭발적으로 증가하여 인터넷 서버의 부하가 더욱 가중되고 있다. 인터넷 서버는 사용자 요구가 급증하는 환경 하에서 보다 많은 사용자에게 서비스를 안정적이고 지속적으로 제공할 수 있는 고가용성(High Availability)과 확장성(Scalablility)을 동시에 갖추어야 한다. 클러스터 서버는 이와 같은 사용자 요구사항을 만족시킬 뿐만 아니라 가격 대 성능비 면에서 우수하여 최근 인터넷 서비스에 활용되고 있다.

클러스터 시스템은 독립적으로 사용 가능한 컴퓨터 시스템을 네트웍으로 연결하여 단일 자원으로 사용하는 분산·병렬 처리시스템의 한 종류이며 사용 목적에 따라 시스템 구조가 달리 적용될 수 있다. 본 논문에서는 다양한 클러스터 시스템 중 웹 서비스에 적용되는 구조를 대상으로 분석하였다.[1]

웹서비스용 클러스터 시스템은 인터넷을 통하여 시스템에 들어오는 사용자 요구를 클러스터 시스템 내에 분산시키는 디렉터(Director)와 실제로 사용자 요구를 처리하는 서버(Server)들로 구성된다. 디렉터는 외부에서 입력되는 사용자 요구를 클러스터 시스템을 구성하고 있는 서버들에게 적절히 분산시켜 시스템의 처리능력을 최대한 활용하고 사용자에게 안정적인 서비스를 제공하는 역할을 한다.

클러스터 시스템에서 사용하는 작업분배 방법에는 여러 가지가 있으나 그 중 Round-Robin 방법과 Least-Connection 방법이 대표적이다. 이 중 Round-Robin 방법은 로드밸런서(디렉터)가 사용자 요구를 입력되는 순서대로 서버들에게 차례로 할당하는 것으로, 일견 아무런 정보를 사용하지 않는 것처럼 여겨진다. 그러나 그 이면에는 가장 오래 전에 작업을 할당한 서버가 평균적으로 가장 많은 유휴처리능력을 가지고 있을 것이라는 정보를 활용하는 것이다.[2]

Least-Connection 방법은 로드밸런서(디렉터)가 실시간으로 서버들의 작업처리 상태를 파악하여, 새로운 사용자 요구를 가장 적은 수의 connetion을 유지하는 서버에게 할당하는 방법이다. 이 방법은 각 노드에 배분된 작업의 양을 측정하는 것은 아니지만, 현재 서비스 중인 connection의 숫자로 부하를 추정하는 것으로 볼 수 있다.

이 두 가지 작업분배 방법을 비교하기 위하여 로드밸런서가 그 어떤 정보도 활용하지 않고, 사용자의 요구를 무작위로 서버들에게 분배하는 방법을 그 비교대상으로 삼았다. 즉, 로드밸런서는 새로운 사용자 요구를 클러스터 시스템을 구성하고 있는 서버들 중 무작위로 선택하여 할당하는 것이다.

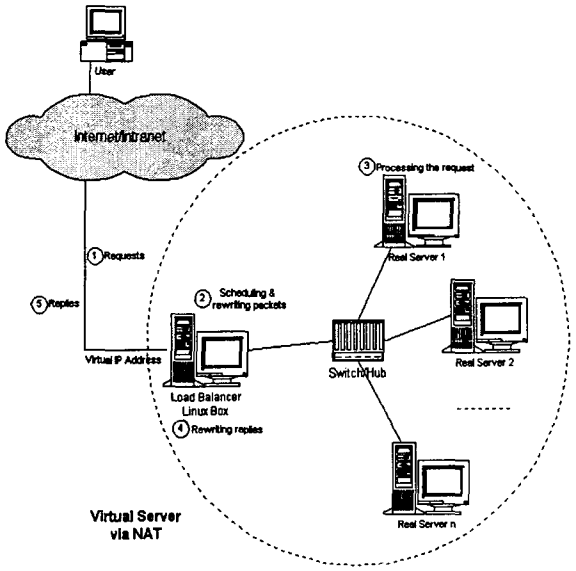
2. 클러스터 시스템 구조

기존의 웹 서비스용 클러스터 시스템으로는 리눅스 기반의 Linux Virtual Server(LVS)가 많이 활용되고 있다. 이와 같이 클러스터 시스템을 가상 서버(Virtual Server)라고 부르는 까닭은 전체 시스템이 사용자에게 마치 하나의 서버가 서비스하고 있는 것처럼 보여지기 때문이다. 이 때 클러스터 시스템은 Virtual IP를 사용하여 서비스를 사용자에게 제공하게 된다.[3]

LVS는 인터넷을 통하여 시스템에 들어오는 사용자 요구를 클러스터 시스템 내에 분산시키는 디렉터와 서버로 구성되어 있으며, 운용되는 방법에 따라 그 구조를 [그림 1]과 [그림 2]와 같이 구별할 수 있다.

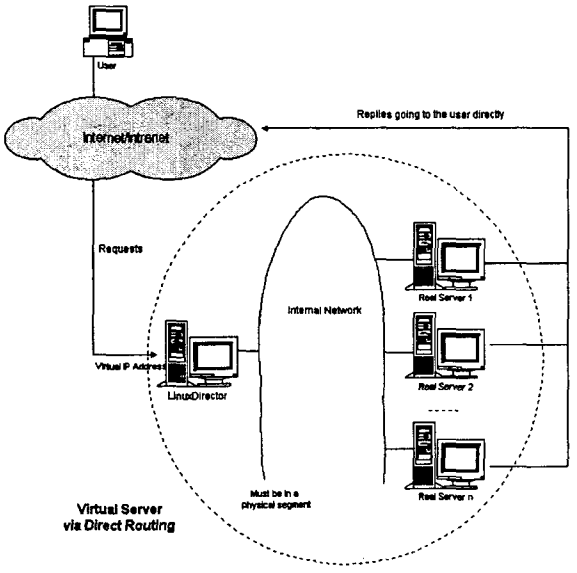
NAT(Network Address Translation) 구조는

[그림 1]에서와 같이 디렉터가 시스템 외부에서 들어오는 사용자 요구를 서버에 분배하여 처리한 후 그 결과를 디렉터가 직접 사용자에게 인터넷을 통하여 전송한다. 이와 같은 구조는 다수의 서버에 비하여 디렉터의 부하가 많아져 확장성이 떨어지므로 클러스터를 구성하는 서버의 수는 10~20여 개로 제한된다.



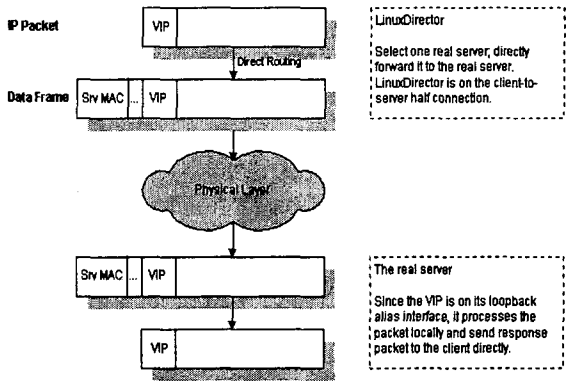
[그림 1] Linux Virtual Server 구조-NAT

이에 비하여 DR(Direct Routing)/TUN(IP Tunneling)방법에서는 디렉터의 역할은 외부에서 들어오는 사용자 요구를 서버에 연결해 주기만 할 뿐 연결 이후의 과정에는 전혀 관여하지 않는다. 따라서 NAT 방법보다는 디렉터의 부하가 줄어들어 클러스터 시스템 내의 서버의 수를 보다 많이 확장시킬 수 있다.



[그림 2] Linux Virtual Server 구조-DR/TUN

클러스터 시스템의 운용을 구체적으로 살펴보면 외부의 사용자 요구를 담고 있는 패킷(Packet)이 서비스용 가상주소(Virtual IP)로 시스템에 유입되면, 디렉터는 클러스터 시스템을 구성하고 있는 서버 리스트를 살펴보고 적당한 서버에게 connection을 보내게 된다. [그림 3]은 디렉터가 사용자 요구를 담고 있는 패킷을 서버에게 보내기 위하여 처리하는 과정을 나타낸다. 여기에서 볼 수 있듯이 디렉터가 처리하는 작업은 외부로부터 유입된 일정한 크기의 패킷에 서버의 MAC 주소를 덧붙이는 동일한 작업이 반복하는 것이다.



[그림 3] direct routing workflow

이에 비하여 서버는 사용자의 요구에 따라 수행할 작업량이 다르며, 특히 웹서비스인 경우에는 정적인 자료(HTML, GIF, 등)를 다루는 서비스와 동적인 자료(CGI 등)를 다루는 서비스가 확연히 다른 양태를 보인다.

그리고 웹서버는 구동 방식에 따라 크게 MP(Multiple Process)와 ED(Event Driven) 방식으로 나뉜다. MP방식의 웹 서버는 새로운 connection이 들어올 때마다 프로세스를 생성한다. 이 때 connection들은 각각의 프로세스에서 수행되며 프로세스의 스케쥴링은 OS의 스케쥴링에 따르므로 웹서버는 관여하지 않는 반면 ED 방식의 웹 서버는 모든 사용자의 request를 하나의 프로세스가 처리한다.

본 논문에서는 확장성이 보다 뛰어난 LVS-DR/TUN 구조만을 대상으로 하여 클러스터시스템에서 디렉터가 행하는 작업분산 방법을 비교하였다.

3. 모형화 및 모의실험

클러스터 시스템은 사용자가 요구한 작업을 시스템의 특성상 다음 두 가지 단계로 구분하여 처리한다.

(1) 디렉터가 외부 사용자의 접속을 서비스노드에 포워딩하는 전처리 단계 ([그림 4]의 System A)
(2) 사용자가 요구한 작업을 서비스노드가 처리하는 실처리 단계 ([그림 4]의 System B)

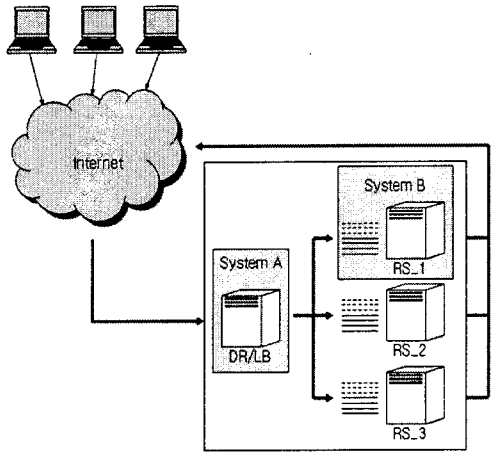
디렉터가 담당하는 전처리 단계는 도착과정을

포아송과정(Poisson Process)으로 가정하면, M/D/1 시스템으로 모형화할 수 있으며 그 성능척도들은 다음과 같다.[4]

$$P(z)=\frac{(1-\rho)(1-z)}{1-ze^{\rho(1-z)}},\qquad \rho=\frac{\lambda}{\mu}$$

$$L=\rho+\frac{\rho^2}{2(1-\rho)},\ L_q=\frac{\rho^2}{2(1-\rho)}$$

$$W=\frac{L}{\lambda},\ W_q=\frac{L_q}{\lambda}\quad \text{(by Little's Law)}$$



[그림 4] 시스템 모형화

서버가 담당하는 실처리 단계는 사용자 요구의 서비스 시간이 지수분포(Exponential Dist.)을 따른다고 가정하면 웹서버 구동 방식에 따라 일반적인 G/M/1 모형과 G/M/1 with process sharing 모형을 따른다고 할 수 있다.

위와 같이 클러스터 시스템을 분해하여 서브시스템별로 고찰하는 경우, 몇 가지 가정을 적용하면 기존의 대기이론의 모형에 대입할 수 있으나 전체를 연동하여 분석하기는 매우 어렵다. 그리고 서버 측면(System B)에서도 입력 프로세스가 디렉터의 작업분산 방법에 따라 달라지므로 이를 분석하는 것 역시 쉽지 않다.

디렉터의 작업분산 방법을 비교하기 위해서는 시스템 전체를 연동하여 분석해야 하므로 시뮬레이션을 통한 실험을 병행하여 그 결과를 분석하였다.

모의실험에서 가정한 내용은 다음과 같다.

(1) 요구도착간격: $A\sim \mathrm{Exp}(\lambda), E[A]=\frac{1}{\lambda}=2.0$

(2) 전처리시간: $S_0=\frac{1}{\mu_0}=1.0$

(3) 실처리시간: $S_1,S_2,S_3\sim \mathrm{Exp}(\mu)$,
$E[S_i]=\frac{1}{\mu}=5.0$

(4) 서비스규칙:FCFS(First Come First Served) Process Sharing

기존의 테스트베드 실험결과에서 볼 수 있듯이 전처리 단계의 소요시간은 사용자 요구의 특성(정적자료요청, 동적자료요청)에 따라 평균 실처리 단계 소요시간의 1/4 ~ 1/6 정도에 불과하다. 따라서 본 모의실험에서는 $\frac{1}{\mu_0}:\frac{1}{\mu}=1:5$ 를 가정하였다.[5]

디렉터의 로드밸런싱 알고리듬을 비교하는 성능척도로는 다음을 사용하였다.

(1) 서버별 utilization factor
(2) 서버별 queue length
(3) response time

모의실험 결과는 아래의 표와 같다. [표 1], [표 2], [표 3]은 웹서버 구동 방식 중 MP에 해당하는 모의실험의 결과를 보여준다.

Scheduling Method	Util. factor	Response time
RS_1	0.820	28.56
RS_2	0.867	29.45
RS_3	0.839	33.44

[표 1] Random Scheduling 성능 (PRSH)

Scheduling Method	Util. factor	Response time
RS_1	0.791	18.36
RS_2	0.811	22.46
RS_3	0.796	29.30

[표 2] Round-robin Scheduling 성능 (PRSH)

Scheduling Method	Util. factor	Response time
RS_1	0.880	12.09
RS_2	0.813	11.22
RS_3	0.711	10.48

[표 3] Least-conn. Scheduling 성능 (PRSH)

[표 4], [표 5], [표 6]은 웹서버 구동 방식 중 ED에 해당하는 모의실험의 결과를 보여준다.

Scheduling Method	Util. factor	Queue length	Response time
RS_1	0.820	4.597	28.79
RS_2	0.867	5.216	30.00
RS_3	0.840	5.670	34.20

[표 4] Random Scheduling 성능 (FCFS)

Scheduling Method	Util. factor	Queue length	Response time
RS_1	0.786	3.039	18.70
RS_2	0.803	2.713	16.65
RS_3	0.813	2.991	18.35

[표 5] Round-robin Scheduling 성능 (FCFS)

Scheduling Method	Util. factor	Queue length	Response time
RS_1	0.882	2.181	12.02
RS_2	0.808	1.884	11.44
RS_3	0.712	1.574	11.02

[표 6] Least-conn. Scheduling 성능 (FCFS)

5. 결론 및 향후 연구과제

앞의 모의실험의 결과에서 볼 수 있듯이 작업분배방식 중 Least-Connection 방법이 두 가지 웹서버 구동 방식에 있어서 모두 우월함을 확인 할 수 있다. 웹서비스를 제공받는 사용자의 관점에서 보면 응답시간이 가장 중요한 성능평가 척도가 된다.

응답시간의 관점에서, Least-Connection 방법이 가장 우수하며, Round-Robin 방법 역시 아무런 정보를 사용하지 않는 경우보다 나은 성능을 보여준다.

또한 디렉터의 작업분배 능력을 평가하기 위해서는 각 서버의 Utilization Factor 값의 크기를 비교하면 된다. 각 서버의 Utilization Factor 값의 편차가 적을수록 디렉터가 작업분배를 고르게 했다고 평가할 수 있다.

이와 같은 관점에서 모의실험 결과를 살펴보면 Least-Connection 방법이 Round-Robin 방법보다 RS_1에 치우치는 경향을 볼 수 있는 데 이는 가장 적은 수의 connection을 유지하고 있는 서버를 찾음에 있어 동률이 발생할 경우 앞선 순서에 있는 서버에 새로운 connection을 할당하는 실험 방식에 따른 것이다. 이를 보정한다면, Utilization Factor의 편차를 줄일 수 있을 것이다.

이상의 실험결과에서 볼 수 있듯이 Least-Connection 방법이 Round-Robin 방법을 모든 평가 척도에서 압도하지만 실제 시스템에 적용할 경우에는 추가적인 환경변수를 고려해야 한다.

Round-Robin 방법은 디렉터와 서버가 새로운 connection을 어느 서버에 할당하는가를 결정하기 위해 소모하는 컴퓨팅 타임이 거의 없지만, Least-Connection 방법은 이벤트가 일어날 때마다 서버가 디렉터에게 현재 자신이 유지하고 있는 connection의 수를 보고해야 한다. 이과 같은 정보를 공유하기 위해서는 디렉터와 서버사이에 오버헤드가 발생하게 되고 이는 전체 클러스터 시스템의 성능을 떨어뜨리게 된다. Least-Connection 방법을 사용함으로써 얻게 되는 성능향상과, 오버헤드로 인한 시스템 부하를 함께 고려하여 작업분배 방법을 결정하여야 한다.

본 논문은 기존의 작업분배 방법의 성능을 평가하기 위한 가장 기본적인 모의실험을 수행한 것으로 추가적인 시스템 환경을 반영하여 보다 정치한 분석과 실험을 병행해야 할 것이다.

6. 참고문헌

[1] Gregory F. Pfister, In search of Clusters, Prentice Hall PTR, 1998.

[2] W. Zhang, "Linux Virtual Server for Scalable Network Services", Ottawa Linux Symposium 2000.

[3] W. Zhang and et al. Linux Virtual Server Project, http://www.linuxvirtualserver.org/.

[4] 이호우, "대기행렬이론", 도서출판 기술, 1996.

[5] eTesting Labs Reports: Internet Appliance, http://www.etestinglabs.com/main/reports/internetapp.pdf