효율적 부하분산 클러스터 시스템 구축 및 알고리즘에 대한 연구

김석찬 이 영
계명대학교 산업시스템공학과 kalsman@kmu.ac.kr yrhee@kmu.ac.kr

Abstract

웹 클러스터 시스템은 동일 서버 군에서 하나 이상의 서비스를 제공하여 비용대비 효과 면에서 높은 가용성 및 보다 빠른 서비스를 제공할 수 있는 시스템이다. 본 연구는 폭주하는 부하를 웹 클러스터 시스템 내의 각 서버에 균등하게 분산 시키는 방법에 대하여 연구하며, 부하분산의 방법론 및 부하 분산 알고리즘을 제시한다. 기존의 WLC 알고리즘의 변형인 PWLC 알고리즘은 주기적인 웹 클러스터의 부하 측정을 통해 가중치를 적용하고 주어진 기간 동안 부하를 가중치에 따라 점진적으로 분산 시키는 방법이다. 이 알고리즘을 동일 기종 및 이기종 웹 클러스터 시스템에 적용하였고, 또한 기존에 제안된 DWRR 알고리즘과의 시뮬레이션을 통한 비교에서 제안된 PWLC 알고리즘이 보다 효과적임을 알 수 있었다.

키워드 : PWLC, 가중치, 클러스터 시스템,

1. 서론

인터넷에 대한 수요의 증가는 인터넷 트래픽의 급격한 증가를 가져와 웹 서버 과부하의 원인이 되고 있다. 현재 서버 과부하 문제를 해결하기 위한 방법으로 서버-클러스터링 (server-clustering) 기술이 비용대비 효과적인 대안으로 인식되고 있다. 클러스터 시스템은 동종 또는 이종 노드들을 네트워크로 연결하여 시스템을 확장하는 방법으로 클러스터 내의 개별 시스템을 서버 또는 노드(node) 라고 한다. 클러스터 시스템은 비용 대비 효과 면에서 고가용성 및 고성능을 보장하며 사용자에게 투명한(user transparent) 서비스를 제공하는 시스템을 구성할 수 있다[1, 2].

부하분산 클러스터 시스템은 클라이언트로부터의 요청을 클러스터 내의 다른 노드로 할당 역할을 하는 부하 분배기와, 요청에 대해 서비스를 제공하는 노드로 구성된다. 부하분산 클러스터 시스템에서 부하 분배기는 서비스 요청이 있을 경우, 특정한 알고리즘을 바탕으로 적절한 노드를 선발하여 그 요청을 처리하게 한다. 즉, 하나의 작업을 여러 대의 노드가 나누어 처리하는 것이 아니고, 분산 알고리즘에 의해 선택된 노드가 배정받은 작업 전체를 처리한다. 그러므로 병렬 시스템과는 달리 쇄도하는 모든 요청이 여러 대의 노드에게 골고루 할당되어 부하가 분산되도록 한다[1].

본 연구에서는 효율적인 부하 분산 기능이 있는 클러스터 시스템의 구성과 부하분산 알고리즘을 제안하고자 한다. 특히 본 연구에서는 Zhang[1]에 의해 연구된 Linux 커널(kernel) 기반의 클러스터 시스템을 대상으로 하였다. 본 연구의 대상이 되는 클러스터 시스템 역시 네트워크로 연결된 하나의 부하 분배기와 다수의 노드로 구성된다. 경우에 따라 장애극복(fail-over)을 위해 다수의 부하 분배기 들이 존재할 수 도 있으며, 부하 분배기가 웹 서비스를 제공하는 노드의 역할을 수행 할 수 도 있다.

기존의 클러스터 시스템은 각 노드의 접속 수에 근거하여 작업을 할당하고 있다. 그러나 접속수만을 근거로 하여 작업을 할당하는 경우, 실제 서비스를 제공하는 노드에 대한 부하 분산이 효과적으로 실현될 수 있을지는 의문이다. 왜냐 하면 접속 수에 근거한 부하 분산은 각 요청의 부하 정도와는 관계없이 부하를 분산시키므로 부하가 한쪽으로 편중되는 경우, 부하가 편중된 노드에서 자원의 소모가 심각하게 일어나 효율성이 저하되는 문제가 발생할 수 있기 때문이다. 그러므로 본 연구에서는 클러스터 시스템의 부하에 기초하여, 각 노드의 가중치를 주기적으로 산정함으로써 클러스터 시스템 전체의 효율을 제고하는데 목적을 둔다. 본 연구에서는 각 노드의 자원

소모량에 기초한 가중치 변경 알고리즘을 제시하고, 모의실험 등을 통하여 제안된 알고리즘이 클러스터 시스템 내에서 부하 분산에 기여하는 효과를 분석하고자 한다.

본 연구의 구성은 2장에서 관련 연구에 대해 소개하며, 3장에서 클러스터 시스템에 대하여 소개한다. 4장에서는 본 연구에서 제시하는 부하 분산 알고리즘에 대하여 소개하고, 5장에서는 제시된 부하 분산 방법에 의한 수치 모의실험 결과 비교 및 분석이 언급된다. 6장에서는 모의실험을 통한 각 클러스터 시스템의 성능평가에 대하여 언급하고 마지막으로 7장에서 결론을 맺는다.

2. 관련연구

기존의 부하분산 연구는 부하 분산의 주체에 따라 클라이언트 측면과 서버 측면 그리고 DNS 측면으로 나눌 수 있다. 클라이언트 측면의 부하분산의 경우, 가장 단순하고, 쉽게 적용할 수 있는 미러링(Mirroring) 방식이 있다. 사용자가 동일한 서비스를 제공하는 다수의 서버 중에서 임의로 하나를 선택하여 부하를 분산시키는 방식으로 클라이언트의 캐시로 인해 처음 선택된 노드로 계속해서 연결될 수 있다는 점에서 그 효율성이 낮은 방식이다[2].

DNS 측면에서의 부하분산 방식인 Round Robin DNS는 DNS에서 특정 도메인과 다수의 IP 주소를 연결하여 해당 도메인에 대한 연결 요청이 발생할 경우 다수 IP주소로 차례로 혹은 전체 클러스터 노드 혹은 클라이언트의 상태에 따라 연결을 할당하는 방식이다. 이 방식의 문제점은 DNS 캐시(cache)가 남아 있거나 Proxy Server를 이용할 경우 부하가 특정 노드로 편중될 수 있다는 단점이 있다[2, 3].

서버 측면의 경우, 서버의 상태를 주기적 혹은 실시간으로 검사하여 그 결과에 따라 클라이언트의 요청을 최소 부하를 가지는 서버에게 할당하는 방식이 있다. 이 방식은 클러스터의 첨단에 부하분배기를 두고 각 노드의 상태를 주기적 혹은 실시간으로 검사하여 부하를 할당한다[1, 2, 3, 4]. 이와 달리 부하분산을 위해 요청을 분석하여 부하를 분산시키는 방법도 연구되고 있다. 이 방식의 핵심은 클러스터 첨단에서 클라이언트의 요청을 분류하여 요청된 컨텐츠의 파일 크기 또는 미리 마련된 기준에 따라 각 서버에 요청을 할당한다. 이 기법은 서버의 부하를 측정하는 방식의 단점으로 지적되는 추가적인 부하(overhead)를 줄일 수 있다는 장점은 있지만, 작업을 할당하는 부하분배기 상에서 병목현상이 발생될 수 있다는 단점이 지적되고 있고, 컨텐츠 별로 분류한 부하의 크기와 실제 부하의 크기 차이가 심한 경우, 장기적 관점에서 서버의 부하정도를 오판 할 수 있다는 단점이 있다[5].

3. Clsuter 시스템

클러스터 시스템은 부하분배기와 실제 서비스를 제공하는 노드들로 구성된다. 클러스터 시스템은 논리적인 topology 형태에 따라 다소 차이가 있으나 공통적으로 클러스터 시스템을 대표하는 가상 주소를 이용하여 클라이언트에게 투명한 서비스를 제공한다. 특정 노드로 할당된 요청을 해당 노드로 보내는 방법으로는 요청을 다른 패킷으로 감싸거나 혹은 도착지 주소를 변경하는 방법 등이 이용되고 있다[1].

클러스터 시스템은 일반적으로 클라이언트들의 요청이 쇄도하는 사이트에 적용되고 있다. 이러한 경우, 클러스터 시스템으로 들어오는 다양한 요청이 노드들에 미치는 영향은 서로 다를 수 있다. 클러스터 시스템은 각 노드가 처리하는 접속 수에 근거하여 작업 할당을 결정하기 때문에 노드의 자원소모에 있어 불균형을 초래할 소지가 있다. 클라이언트로부터의 요청이 발생시키는 부하의 크기가 거의 동일하다면 상관없지만, 실제 각 요청이 노드에 미치는 부하는 일정하지 않다. 이것은 클라이언트들이 서버의 자원을 소모하는 정도가 일정하지 않다는 것과 같은 의미로 서버 자원의 편중된 소모는 더욱 심화될 가능성이 내포되어 있다. 이와 같은 점을 고려할 때, 기존의 클러스터 시스템은 실제 서비스를 하는 노드 자체에 대한 부하 분산에 적극적으로 대처하는 시스템이라고 볼 수 없다.

4. 부하분산 알고리즘 및 부하의 할당

기존의 접속 수에 근거한 알고리즘으로는 Round-Robin Scheduling(RR) 알고리즘과 weighted Round-Robin Scheduling (WRR) 알고리즘, Least Connection Scheduling(LC) 알고리즘, Weighted Least Connection Scheduling(WLC) 알고리즘이 있다[1].

본 연구에서 제안하는 PWLC 알고리즘은 시스템의 성능 측정에 의한 추가적인 부하(overhead)를 줄이기 위해, 일정 주기마다 시스템 상태정보에 측정하여 부하의 정도를 파악한다. 그리고 측정된 각 노드의 부하에 따라 가중치가 부여되고 이 가중치를 근거로 부하가 할당되는 알고리즘이다. 즉 PWLC 알고리즘의 작업 할당 기준은 할당 시점에서 해당 노드의 가중치와 접속 수에 근거한다. n개의 노드에 대한 각 가중치(w_i)와 각 노드별 접속 수 $C_i (i = 1, 2, ..., n)$를 나누어 이 값 중 최소 값을 가지는 쪽에 접속을 할당한다. 이를 표현하면 (1)과 같다. 여기에서 S는 클러스터 시스템의 전체 접속 수를 의미한다. 새로운 서비스 요청이 들어올 경우 n 개의 서버 중 (1)을 만족하는 임의의 서버 j 에게 접속을 할당하게 되는 것이다.

$$\frac{C_j}{w_j} = \min\left\{\frac{\left(\frac{C_i}{S}\right)}{w_i}\right\} \qquad (1)$$

$$= \min\left\{\frac{C_i}{w_i}\right\}$$

기존의 접속 수에 근거한 알고리즘 중 WLC 알고리즘이 전술한 알고리즘 보다 더 신뢰할 수 있는 알고리즘이라고 할 수 있으나 이 또한 클러스터 상에 있는 각 노드의 상태정보가 수반되지 않은 부하분산 방식이다. 기존 알고리즘은 고정된 가중치를 이용함으로써 부하가 어느 한 서버로 편중되는 경우가 발생하더라도 이를 보정하는 능력이 없기 때문에 부하의 편중이 가중 될 수 있다. 이에 대한 대안으로 PWLC 알고리즘은 주기적으로 노드의 부하 정도를 측정하여 가중치를 갱신하고 잘못된 가중치에 의해 노드간의 부하의 불균형 현상이 발생하더라도 보정하는 능력을 가지도록 하였다. 또한 노드간의 부하 불균형을 낮은 수준에서 유지하기 위해 (2)에서처럼 가중치가 적용되는 가중치 산정 시점들 사이(가중치 산정 주기)에서 발생하는 평균적인 요청 수를 각 노드의 가중치 비율로 할당하는 방식을 적용하였다. 이러한 방식은 가중치 산정주기 동안 부하의 불균형 상태를 균형상태로 유도하는 기능을 수행하게 된다. C_{sp}를 가중치 산정주기 사이에서 발생하는 평균 도착 수, w_{base}를 각 노드에 할당된 기본 가중치, n을 클러스터의 노드 수, $L_{\max}$, $L_{\min}$을 각 노드들이 가지는 최대 부하와 최소 부하, $w_{\max}, w_{\min}$을 노드들에 할당될 최대 가중치와 최소 가중치라고 하면

$$C_i = \frac{w_i}{\sum_{j=1}^{n} w_j} \times C_{sp}$$

이다. 그러므로

$$C_{sp} \sim \sum_{i=1}^{n} w_i$$

$$\alpha = L_{\max} - L_{\min} \qquad (2)$$

$$w_{base} = \frac{C_{sp}}{n}$$

라는 조건을 만족 시킬 경우

$$w_{\max} = w_{base} + \frac{\alpha}{2},\ w_{\min} = w_{base} - \frac{\alpha}{2}$$

의 결과를 얻을 수 있고, 이때 $w_{\max}, w_{\min}$을 각각 $L_{\min}, L_{\max}$를 가지는 노드에 할당하고 나머지 노드는 부하의 비율에 따라 할당한다. 이 결과, 클러스터에 도착하는 요청들이 각 노드의 가중치의 차만큼씩 더 배분되게 된다. 즉, 최대 가중치 노드는 최소 가중치 노드보다 최대 가중치와 최소 가중치의 차이만큼 더 많은 요청을 할당 받는 것이다. 만약 예상한 평균

도착수보다 더 작거나 더 많다 하더라도 이 방식은 전체 요청을 가중치의 비율에 따라 각 노드에 할당하게 된다. 또한 이 알고리즘은 첫 번째 부하분산 시도에서 부하가 완전한 균형 상태가 되지 않더라도 차후의 시도에서 점진적으로 균형 상태로 유도하려는 성질이 있다는 사실이다[6].

PWLC 알고리즘의 부하할당 방식에서 현재의 연결 수는 가중치 산정 주기 동안의 누적된 연결 수로서 전체 부하의 할당은 서버의 상태와 기존 접속 수가 복합적으로 적용된 결과이다. 만약 각 노드에 $w_{t,1} : w_{t,2} : w_{t,3} : w_{t,4}$ = 6 : 3 : 2 : 3이라는 가중치가 부여될 때, 클러스터로 들어오는 일련의 부하에 대하여 할당되는 순서는 다음과 같다.

$$n_1 \rightarrow n_2 \rightarrow n_4 \rightarrow n_3 \rightarrow n_1 \rightarrow n_2 \rightarrow n_4$$
$$\rightarrow n_1 \rightarrow n_1 \rightarrow n_3 \rightarrow n_1 \rightarrow n_2 \rightarrow n_4 \rightarrow n_1 \ldots$$

이 결과로 정확하게 일정 주기 동안 가중치의 차만큼 각 노드에 할당됨을 알 수 있다.

5. 알고리즘의 효율성 비교

본 연구에서 클러스터 시스템에 적용한 PWLC 알고리즘을 효율성을 알아보기 위해 기존 Li[7] 등에 의해 제안된 DWRR 알고리즘과 비교하였다.

DWRR 알고리즘은 전통적인 RR 알고리즘에서 유래 된 것으로, 클러스터 시스템의 각 노드의 상태 정보를 주기적으로 수집하여 가중치를 설정하고 부하를 조절한다는 측면에서 본 연구에서 제안하는 알고리즘과 동일하다. 그러나 점진적으로 부하 정도를 균등 상태로 유도하려는 PWLC 알고리즘과는 달리, DWRR 알고리즘은 가중치 산정 주기 초반에 부하를 균등 상태로 유도하고 남은 기간동안 RR 방식으로 부하를 동일하게 할당하는 알고리즘이다.

모의실험은 클러스터 시스템 전체가 과부하(heavy load) 상태에 있는 것으로 가정하며, 두 알고리즘에 대하여 동일한 환경을 가정했고, 요청의 도착 간격 및 요청에 대한 서비스 시간 또한 동일하게 가정하였다.

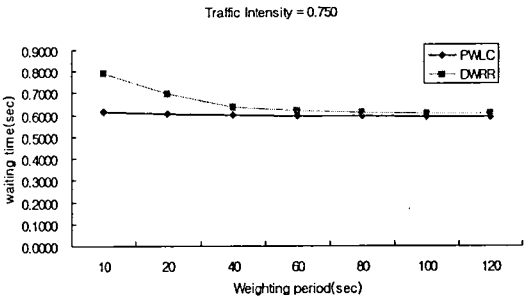

그림 2 Traffic Intensity = 0.750에 대한 순수 대기시간 비교

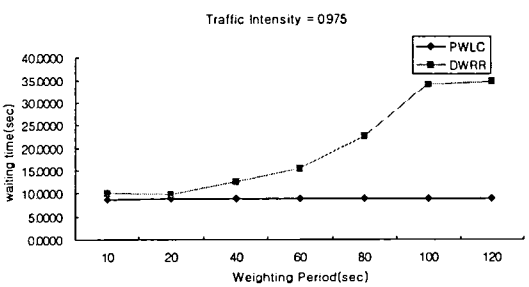

그림 3 Traffic Intensity = 0.975에 대한 순수 대기시간 비교

그림 1과 그림 2는 Traffic Intensity 즉, 클러스터 시스템에 부과되는 부하의 정도와 가중치 산정주기에 따른 순수 대기시간의 변화에 대한 모의실험 결과이다. 이 결과는 과부하가 적용되는 경우와 낮은 부하가 적용되는 경우에 따라 전혀 다른 결과를 나타내고 있다. 과부하가 적용되는 경우 가중치 산정주기를 짧게 적용하는 것이 좀 더 좋은 효과를 보이는 반면 상대적으로 낮은 부하에서는 가중치 산정주기를 길게 하는 것이 효과적임을 실험적으로 알 수 있었다. 그림 1의 경우, Traffic Intensity가 낮기 때문에 전체 클러스터 시스템의 부하를 균등 상태로 유도하기 전에 주기가 종료되어 알고리즘의 적절한 역할을 못하기 때문이다. 그러나 가중치 산정주기가 길어질수록 주어진 주기 내에 부하를 균등상태로 유도할 수 있게 된다. 반대로 Traffic Intensity 높은 그림 2의 경우는 짧은 주기에서도 충분히 균형 상태로

유도되지만, 주기가 길어질수록 오차가 누적되어 알고리즘의 효율성이 떨어지는 것이다.

6. 알고리즘이 적용된 cluster 실험

본 알고리즘에 대하여 동일기종으로 구성된 클러스터 시스템과 이기종으로 구성된 클러스터 시스템에 대하여 실제 실험을 수행하였다. 부하 측정에 필요한 각 노드의 성능 지표를 선정하기 위해 제안된 알고리즘이 적용되지 않은 클러스터 시스템의 과부하 상태에서 CPU의 가용도(utilization)과 free 메모리의 변화 상태를 측정하였다. 그림 3과 그림 4는 실제 과부하 상태에서의 CPU의 가용도 및 free 메모리의 변화를 측정한 결과이다.
그림 3처럼 과부하 상태에서 전체 노드의 CPU의 이용 율이 지속적으로 평균 96~97 %로 높게 나타나 성능지표로서 제외하였으나, 그림 4의 경우, free 메모리의 변화를 측정한 결과 특정 노드로 편중되는 현상이 보인다.

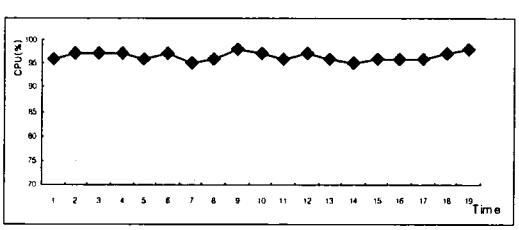

그림 3 CPU 가용도 변화

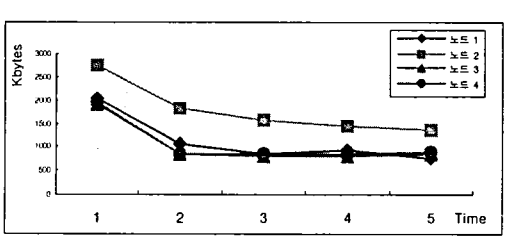

그림 4 기존 알고리즘에 의한 free 메모리의 변화

메모리의 사용이 편중되는 것은 한쪽으로 더 많은 부하가 편중되었다고 볼 수 있으므로 free 메모리의 변화를 클러스터 시스템 상에 있는 각 노드의 성능지표로 선택하였다.

본 연구를 위하여 동일기종 및 이기종 클러스터 시스템을 구성하였고, 제안된 PWLC 알고리즘을 적용한 클러스터 시스템의 성능을 측정하였다. 실험은 동일한 가중치 산정주기에 대하여 동시사용자 수를 증가시켜 그에 따른 각 알고리즘이 적용된 클러스터 시스템들의 성능의 변화를 측정하였다. 또한 클러스터 시스템의 성능척도는 반응시간(Response time)을 사용하였고, 이것은 클라이언트가 요청을 보내서 요청에 대한 결과가 클라이언트에게 최초로 도착한 시간을 의미한다.

다음의 그림 5는 제안된 PWLC 알고리즘이 적용된 클러스터 시스템의 free 메모리의 변화를 측정한 결과이다. 그림 4와 그림 5를 비교할 때 제안된 알고리즘이 적용된 경우, 특정 노드의 free 메모리 편중 소모 현상이 점차 완화됨을 알 수 있다. 이것은 PWLC 알고리즘에 의해 free 메모리가 작은 노드의 부하가 free 메모리가 많은 노드로 이전되었기 때문이다. 이 결과는 반응시간의 측정결과에서 직접적인 영향을 미친 것으로 판단된다. free 메모리의 편중 소모 현상을 완화시킨 결과 표 1처럼 기존의 알고리즘 보다 반응시간을 단축시키는 것을 목격할 수 있었다.

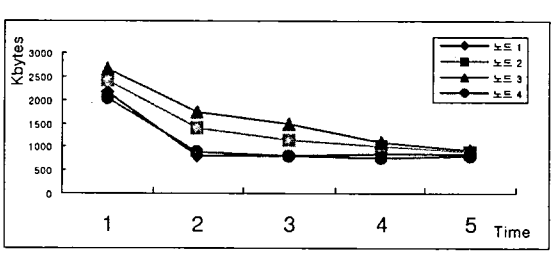

그림 5 PWLC 알고리즘의 free 메모리의 변화

이기종의 노드들로 구성된 이기종 클러스터 시스템의 경우, 동일 부하가 각 노드의 성능에 미치는 영향의 차이가 뚜렷함을 알 수 있다.

표 1 노드의 수가 4대인 경우의 반응시간(동일기종)

실험군 \ 동시사용자	90	110	130	150
기존 알고리즘	0.034	0.047	0.093	0.109
PWLC 알고리즘	0.028	0.036	0.075	0.094

표 2 노드의 수가 4대인 경우의
반응시간(이기종)

실험군 \ 동시사용자	200	300	400	500
기존 알고리즘	0.251	0.510	0.735	0.984
PWLC 알고리즘	0.247	0.480	0.594	0.738

표 2는 이기종 클러스터 시스템에 대한 반응시간의 차이를 측정한 결과이다. 결과에서 알 수 있듯이 동일기종 클러스터 시스템과 마찬가지로, 이기종 클러스터 시스템에서도 PWLC 알고리즘이 적용된 경우가 기존 클러스터 시스템 보다 더욱 효과적임을 목격할 수 있다. 특히 동시사용자 수가 증가하는 경우, 동일기종 클러스터 시스템의 결과보다 그 효율성의 차이는 더욱 분명하게 나타남을 볼 수 있다. 이러한 이유는 이기종 클러스터 시스템의 경우, 최하 성능의 노드는 언제나 낮은 가중치를 가질 확률이 높아진다. 특히 free 메모리의 관점에서 메모리가 작은 최하 성능의 노드는 다른 노드에 비해 free 메모리를 빠르게 소모시키게 되고, free 메모리가 고갈되는 한계시점에서는 메모리 부족현상으로 스왑(Swap)이 빈번히 발생하게 되어 노드의 성능이 더욱 악화된다. 그러나 PWLC 알고리즘을 적용한 경우, 주기적으로 각 노드의 상태를 파악하여 free 메모리가 가장 작은 노드는 부하를 경감시키기 때문에 전체 클러스터에 작용하는 부하를 성능에 따라 균형적으로 분산시킬 수 있었다.

그러므로 PWLC 알고리즘은 성능의 차가 뚜렷한 이기종 클러스터 시스템에 더욱 적합한 알고리즘이라고 할 수 있다.

7. 결론 및 추후 연구방향

본 연구는 클러스터 시스템의 효율적인 부하분산 방법론과 제고된 알고리즘을 제안하고 있다. 이를 위해 가중치 산정주기 동안 클러스터 시스템에 부과되는 부하를 점진적으로 각 노드에 분산시키는 PWLC 알고리즘을 제안하였고, 기존의 알고리즘과 효용성을 비교하여 그 더욱 효율적임을 보였다. 또한 제한적인 성능지표를 사용하는 동일 기종 클러스터 시스템과 이기종 클러스터 시스템에 대한 반응시간의 측정 실험에서도 PWLC 알고리즘이 효율적임을 보였다. 특히 PWLC 알고리즘은 이기종의 노드로 구성된 클러스터 시스템의 부하분산에 더욱 효과적임을 알 수 있었다.

PWLC 알고리즘이 부하를 분산시키는 능력이 있지만 제한적인 성능지표를 사용하였기 때문에 다양한 클러스터 시스템 환경 하에서 주어진 가중치 산정주기 동안 정확하게 균형 상태로 유도하지는 못할 수도 있다. 그러므로 다양한 자원들의 사용 상태를 성능 지표로 표현할 수 있는 보다 정확한 모형이 수립될 경우, 더욱 정교한 부하분산이 가능할 것이다. 또한 클러스터 시스템의 성능 측정에 의해 발생하는 부하를 최소로 하며, 과부하 상태에서 가장 효율적인 가중치 산정주기를 알 수 있는 모형에 대한 연구가 추후 연구되어야 할 것이다.

참고문헌

[1] Wensong zhang, *The paper Linux Virtual Server for Scalable Network Services*, Ottawa Linux Symposium 2000.

[2] Valeraia C, Michele c, Philip S. Yu, *Dynamic load balancing on web-server systems*, IEEE Internet Computing 3 (3), 1999.

[3] Patrick Killelea, *Web Performance Tuning*, O'Reilly second edition October, 2001.

[4] Cisco Systems, *Cisco LocalDirector 400 Series Content Switches Relevant Technologies*", http://www.cisco.com/en/US/products/hw/contnetw/ps1894, 2002.

[5] IBM corp, *IBM Technical Report CS 260*, htttp:/www.ibm.com/, 2004.

[6] Y. Rhee, S.C Kim, *System Infrastructure of Efficient Web Cluster System to decrease the Response Time using the Load Distribution Algorithm*, Korean Information Science Society, Vol. 10-6, pp. 507-513, Dec 2004.

[7] Der-Chiang Li, Chihsen Wu, Fengming M. Chang, *Determination of the parameters in the dynamic weighted Round-Robin method for network load balancing*, computer & operations research, Vol. 32, pp. 2129-2145, 2005