클러스터 시스템의 부하분산 알고리즘의 효율성 비교분석

(An Analysis and Comparison on Efficiency of Load Distribution Algorithm in a Clustered System)

김 석 찬 † 이 영 ††
(Seok-chan Kim) (Young Rhee)

요 약 본 연구에서는 클러스터 시스템에 적용되는 새로운 부하할당 알고리즘을 기존의 알고리즘과 비교하여 분석하고자 한다. PWLC 알고리즘은 설정된 가중치 산정주기마다 시스템의 부하를 감지하여, 각 서버에 가중치를 부여하여 다음 주기에 가중치에 의하여 부하를 분산시키는 알고리즘이다. PWLC 알고리즘과 DWRR 알고리즘을 가중치 산정주기를 변화시키면서 분산과 대기시간 등에 비교하였다.

가중치 산정주기가 너무 짧으면 시스템은 부하를 감지하는데 잉여부하가 소요될 수 있으며, 이와 반대로, 가중치 산정주기가 너무 길면 알고리즘 적용에 의한 부하할당이 비효율적으로 될 수 있다. PWLC 알고리즘이 DWRR 알고리즘보다 더 효율적임을 알 수 있다.

키워드 : PWLC, DWRR, 가중치, 클러스터 시스템

Abstract In this thesis, we analyze the efficiency of the algorithm to distribute the load in the clustered system, by comparing with the existed algorithm. PWLC algorithm detects each server's load in the system at weighted period, and following the detection of the loads, a set of weights is given to each server. The system allocates new loads to each server according to its weight. PWLC algorithm is compared with DWRR algorithm in terms of variance, waiting time by varying weighted period.

When the weighted period is too short, the system bears a heavy load for detecting load over time. On the other hand, when the weighted period is too long, the load balancing control of the system becomes ineffective. The analysis shows PWLC algorithm is more efficient than DWRR algorithm for the variance and waiting time.

Key words : PWLC, DWRR, Clustered system

1. 서 론

인터넷 사용의 보편화 및 대형화 컨텐츠로 인한 트래픽 증가는 네트워크뿐 만아니라 서버시스템에 부하를 가중시켜 매우 중요한 문제로 부각되고 있고, 특히 사용자의 QoS(Quality of Service) 측면에서 트래픽에 의한 부하가 미치는 영향은 경제적인 문제로까지 파급되기에 이르렀다. 이러한 부하 문제의 대안으로 트래픽의 집중을 분산시키는 연구가 다각적으로 진행되고 있다. 트래픽의 분산에 대한 초기의 연구는 주로 컴퓨터를 구성하는 컴포넌트의 성능을 향상시켜 전체 컴퓨터 시스템의 성능을 향상시키는 하드웨어적 관점에서 주로 이루어졌다. 이와는 별도로 서버의 측면에서 부하를 경감시키는 연구도 활발히 진행되었다. 이러한 연구로는 웹 캐싱(Web Caching)을 이용한 서버의 부하를 경감시키는 방법, 부하를 분산시키는 방법 등이 연구되었다. 전자의 경우, 사용자에 의해 서버 컨텐츠에 대한 요구가 최초로 발생하면 그 결과를 반환하고 일정 장소에 저장하여 차후에 동일한 결과에 대한 요구가 발생하면 저장된 캐시를 반환한다. 이는 서버의 CPU 혹은 디스크 I/O를 줄여 부하를 경감시키는 방법으로서 컨텐츠가 갱신되는 경우, 컨텐츠의 일관성(consistency) 문제가 발생한다는 단점이 있다. 후자의 경우는 서버 클러스터를 구성하여

· 본 연구는 2005 학년도 계명대학교 비사연구기금으로 이루어졌음
† 정 회 원 : 계명대학교 산업시스템공학과
kalsman@kmu.ac.kr
†† 정 회 원 : 계명대학교 산업시스템공학과 교수
yrhee@kmu.ac.kr
논문접수 : 2005년 9월 15일
심사완료 : 2006년 1월 26일

부하를 분산시키는 방법이 주를 이루고 있다. 특히 웹 서버 혹은 데이타베이스 시스템 등의 응용프로그램에 대한 부하를 여러 대의 시스템으로 분산시킴으로서 시스템에 미치는 부하를 감소시키는 방식이다. 현재 클러스터를 구성하여 부하를 분산시키는 방법에도 다양한 연구가 진행되고 있다. 하드웨어적인 방법과 소프트웨어적인 방법이 모색되고 있다. 하드웨어적인 방법은 별도의 부하분배기를 클러스터의 첨단에 장치하여 부하를 분산시키는 방법이 있으나 많은 비용이 소모된다는 단점이 있다[1]. 이와는 달리 소프트웨어적인 방법은 특정 노드를 부하분배기로 전용하고 여기에 다양한 알고리즘을 적용하여 부하를 분배한다. 본 연구에서는 웹 클러스터 상에서 발생되는 부하를 균형 있게 분배하는 분산 알고리즘에 주안 하고자 한다. 부하를 분산시키는 알고리즘에는 두 가지 정도로 분류 할 수 있는데 부하를 미리 예측하는 방법과 부하의 예측 없이 부하를 분산시키는 방법이 있다. 전자의 경우, 컨텐츠 별 부하정도를 미리 결정하고 이를 바탕으로 사용자의 요청을 부하분배기 차원에서 분석 하여 각 서버로 적절히 분배하는 방식이다. 이 방법은 각 컨텐츠가 미치는 영향이 정확하다는 가정 하에서는 효과적인 부하분산 방법이 될 수 있지만, 컨텐츠 별 부하의 정도 차가 모호하며, 컨텐츠 종류가 방대한 경우 그 적용이 매우 어렵다는 단점이 있다[2]. 후자의 경우 단순히 분배에만 초점을 둔 Random, RR(Round-Robin) 등이 있으며, 예측보다는 시스템의 부분적인 정보를 지속적으로 파악하여 일정한 법칙에 의해 부하를 분배하는 WRR(Weighted Round-Robin), LL(Least Load), LC(Least Connection) 및 FR(Fast Response) 알고리즘이 있다[3,4]. Random 방식은 부하와 관계없이 임의의 서버에 할당하는 방법으로 전체 클러스터를 불안정하게 유도할 수 있는 단점이 있다. RR 알고리즘은 각 서버의 상태를 추적할 필요가 없다는 장점이 있다. WRR 알고리즘은 RR 알고리즘을 개량한 것으로 각 서버의 성능에 기초로 일정한 간격으로 시스템의 부하상태를 파악하여 가중치를 부여하고 가중치와 기존의 RR 방식을 혼용하는 방법이다. LL 알고리즘과 LC 알고리즘은 각 최소 부하와 최소 연결수를 가지는 서버에 요청을 할당하는 방법으로 연결요청 시 마다 부하정보를 파악해야 한다. FR 알고리즘은 수시로 반응 시간(Response Time)을 측정하여 가장 빠른 서버에게 요청을 할당하는 방식이다. 그러나 이러한 알고리즘은 많은 장점이 있음에도 불구하고 실제 적용 시에는 현실적으로 우선 고려해야하는 문제점이 있다. 즉, 시스템 정보를 파악할 경우, 얼마나 빈번한가에 의하여 서버 시스템에 추가된 잉여부하(overhead)가 중 부과될 수 있기 때문이다. 이러한 문제를 해결하기 위하여 시스템 정보를 이용한 부하분산 방법론과 "how often"이라는 빈도수로 인한 추가된 부하와 tradeoff(교환)를 적절히 혼합하는 알고리즘이 제안되었는데, Li 등[3]에 의하여 제안된 DWRR(Dynamic Weighted Round Robin) 알고리즘은 최적의 시스템정보 파악 주기를 산정하고, 이를 이용하여 주기적으로 시스템의 상태정보를 파악하여, 상태에 의한 가중치를 부여하고, 차등적으로 부하를 할당하는 방식으로 RR 알고리즘과의 비교에서 효율성을 입증하였다.

본 연구에서는 리눅스 클러스링 방법론에서 부하 할당방식인 WLC(Weighted Least Connection)[4]를 변형한 PWLC(Periodic Weighted Least Connection) 알고리즘을 DWRR 알고리즘과 비교하여 효율성을 분석하고자 한다. 주기적으로 시스템의 부하를 측정하여 가중치를 산정하는 방법은 동일하나, 이를 근거로 부하를 할당하는 순서에서 의하여 차이가 있음을 알 수 있다. 본 연구에서는 PWLC 알고리즘과 DWRR 알고리즘에 대한 다양한 실험을 수행하여 각 알고리즘의 효과와 영향을 분석하고자 한다.

본 논문의 2장에서 웹 클러스터 시스템과 기존의 부하분산 기법에 대해 설명한다. 3장에서는 기존의 DWRR 알고리즘과 본 논문에서 제시된 PWLC 알고리즘에 대하여 설명하며 아울러 각 알고리즘을 적용한 경우의 부하 할당과정에 대하여서도 설명한다. 4장에서는 다양한 실험을 통하여 얻은 결과를 토대로 분석하여 5장에서 결론을 맺는다.

2. 클러스터 시스템의 부하분산 방법

1990년대에 들어 활발히 논의된 이후로 클러스터 시스템은 확장성 및 고성능, 고가용성 측면에서 효과적인 서버 시스템의 대안으로 평가 받고 있다. 클러스터 시스템은 동종 혹은 이기종의 시스템을 네트워크로 연결한 시스템을 의미하며, 작업을 분배하는 마스터 노드와 실제 작업을 처리하는 노드를 네트워크로 연결한다. 마스터 노드가 작업 요청을 받아들여 적절한 방법으로 다른 노드에게 할당하기 때문에 사용자는 마치 하나의 거대한 단일 시스템으로 인식하게 된다. 전술한바와 같이 클러스터 시스템은 그 특징적인 구성방법으로 인해 고 가용성, 고성능, 높은 확장성을 상대적으로 적은 비용으로 제공한다는 장점을 가지고 있다[4-7]. 클러스터 시스템은 사용목적에 따라 과학 계산용 병렬 클러스터 시스템(parallel cluster system)과 부하분산 클러스터(load balancing cluster) 시스템으로 분류 할 수 있다. 논 연구에서는 부하분산 클러스터 시스템의 핵심이라 할 수 있는 부하분산 알고리즘에 대하여 논의하려고 한다.

그림 1은 클러스터 시스템의 구성을 실제 서비스를

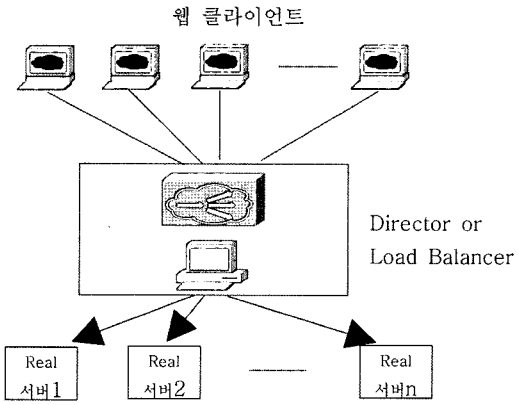


그림 1 클러스터링 웹 시스템

담당하는 Real 서버 군과 부하를 효과적으로 분산하는 역할을 담당하는 Load balancer 등의 실제 구성도를 보여주고 있다. 기존의 부하분산 연구는 단순히 사용자의 선택에 따라 부하를 분산시키는 소극적인 방식의 미러링(mirroring) 방식에서부터 사용자의 요청에 대하여 부하를 예측하여 분산시키는 적극적인 방법까지 다양한 각도에서 수행되어 왔다. 본 절에서는 기존 연구들에 대하여 간략히 소개한다.

2.1 Round Robin DNS

Round Robin DNS는 특정 도메인과 다수의 IP 주소를 연결하여 해당 도메인에 대한 연결 요청이 발생할 경우 다수 IP주소로 차례로 연결을 할당하는 방식으로 부하분산의 기준(최소 부하 원칙, 최소 근거리 원칙 등)이 명확하게 제시된 것은 아니다. 또한 DNS 캐쉬가 남아 있거나 프락시 서버를 이용할 경우 부하가 편중될 수 있다는 단점이 있다[1,7].

2.2 Multicasting 기법의 응용

특정 IP 주소를 멀티캐스트 주소로 지정하여 특정 데이타를 특정 IP 주소로 전송할 수 있도록 하는 멀티캐스팅을 이용하여 부하를 분산하는 기법으로 여러 대의 서버에 같은 멀티캐스트 주소를 지정하되, 각 서버는 특정 URL 클래스의 요청에만 응답하도록 하여 부하를 분산 시킬 수 있다. 하지만 모든 라우터들이 멀티캐스트를 지원해야 하는 단점이 있다. 그렇지 않을 경우, 특정 지역에서 클러스터로 접근하는 것이 불가능해진다[1].

2.3 Content aware approach

부하분산 기법에서 적극적인 방법 중에 하나라고 할 수 있는 기법으로 서버 관점에서 두 가지 정도로 분류되는데, 하나는 요청된 컨텐츠의 파일 크기를 기준으로 할당할 서버를 선택하는 방법과 미리 각 요청에 대한 시스템 자원에 미치는 영향을 분류하고 이를 기준으로 부하를 균등화 시키며 각 서버에 할당 하는 방식을 취한다. 이 기법은 서버의 부하 측정에 따른 잉여부하를 줄일 수 있다는 장점이 있지만, 작업을 배분하는 스위치에서의 처리로 인한 병목이 발생할 수 있다는 단점이 있고, 컨텐츠 별로 부하의 정도를 분류하는 기준이 모호할 수 있다는 단점이 있다[2,8]. 또한 반응시간의 단축을 위해 노드 간 자료를 선 입출 하는 기법을 이용하는 방법[9]과 서비스를 제공하는 노드의 주소와 대상의 이름 및 우선순위에 따라 가중치를 변경하는 방법을 이용하여 부하를 분산시키고자하는 연구 등도 진행되고 있다[10].

3. 부하분산 알고리즘

본 연구에서 LVS 시스템의 핵심이 되는 부하분산 알고리즘에 개선을 위해 PWLC 알고리즘을 제안하며, 이 알고리즘의 우수함을 입증하기 위해 기존의 RR 방식을 응용한 DWRR 알고리즘과 비교하였다.

3.1 DWRR(Dynamic Weighted Round Robin)

DWRR 알고리즘은 전통적인 Round Robin 알고리즘에서 유래 된 것으로, 각 서버의 연결 상태 정보를 추적할 필요가 없다는 장점을 최대한 이용하여, 최소의 노력으로 상태정보를 주기적으로 수집하고자 하는 방법이다[3]. DWRR 알고리즘의 부하분산 방법은 통계학적으로 부하에 대한 분산을 계산함으로써 비롯된다. t 시점의 각 서버의 연결에 대한 분산을 v_t라고 하면, v_t는 다음과 같이 계산된다.

$$v_t = \frac{\sum_{i=1}^{n}\left(load_{t,i} - \overline{load_t}\right)^2}{n} \quad (1)$$

$$where \qquad \overline{load_t} = \frac{\sum_{i=1}^{n} load_{t,i}}{n}$$

분산에 관한 지표가 시사하는 바는 v_t가 적으면 적을수록 균형적인, 즉 반응시간이 최소화되는 부하분산이 이루어지고 있음을 보인다고 할 수 있다.

DWRR 알고리즘은 각 서버의 부하정도를 감지하여, 각 서버에 가중치를 할당하며, 가중치는 통상적으로 부하의 역수로 생각하면 된다. 예를 들어 n개의 서버로 구성된 웹 서버 클러스터 시스템에서 각 서버의 부하정도를 통상적으로 현재 접속된 연결수로 가정한다면, 각 서버의 부하를 $load_{t,1} : load_{t,2} : \ldots : load_{t,n}$이라 할 때, 각 서버에 부과되는 가중치는 (2)와 같이 표시할 수 있다.

$$w_{t,1} : w_{t,2} : \ldots : w_{t,n} = \frac{1}{load_{t,1}} : \frac{1}{load_{t,2}} : \ldots : \frac{1}{load_{t,n}} \quad (2)$$

DWRR 알고리즘에서는 (2)에 근거하여 가중치를 산정하는데, 당연히 초기의 가중치는 정수로 구성될 수는

없다. 따라서 각 서버의 부하에 대한 최소공배수를 구한 다음, 비율을 이용하여 최종적으로 정수를 산출한다. 최종적으로 산정한 가중치는 각 서버에 부과해야 할 부하의 수라고 간주하면 된다. DWRR 알고리즘에서는 초기에는 단순히 RR 알고리즘에 근거하여 순차적으로 각 서버에 할당한 다음, 이어지는 부하는 더 부과해야 할 서버에만 선별적으로 RR 알고리즘을 적용한다.

3.2 PWLC 알고리즘

기존의 접속 수에 근거한 알고리즘으로는 라운드-로빈 스케줄링(Round-Robin Scheduling) 알고리즘과 서버 간의 서로 다른 처리능력을 고려한 가중치 기반 라운드-로빈 스케줄링(weighted Round-Robin Scheduling), 최소 접속 수 스케줄링(Least Connection Scheduling) 알고리즘, 가중치 기반 최소 접속 수 스케줄링(Weighted Least Connection Scheduling) 알고리즘이 있다[4].

PWLC 알고리즘은 WLC 알고리즘을 변형한 것으로, 주기적으로 시스템 상태정보에 의거하여 부하 또는 접속 수 정보를 파악하여, 가중치가 부여되고 이를 근거로 부하가 할당되는 알고리즘이다. 이는 PWLC 알고리즘의 작업 할당 기준은 할당 시점에서 해당 노드의 가중치와 접속 수에 근거한다. 즉 n개의 노드에 대한 각 가중치 (w_i)와 각 노드별 접속 수 $c_i(i=1,2,...,n)$를 나누어 이 값 중 최소값을 가지는 쪽에 접속을 할당한다. 이를 표현하면 아래의 식 (3)과 같다. 여기에서 s는 웹 클러스터 시스템의 전체 접속 수를 의미한다. 새로운 서비스 요청이 들어올 경우 n 개의 서버 중 식 (3)을 만족하는 임의의 서버 j 에게 접속을 할당하게 되는 것이다.

$$\frac{c_i}{w_j} = \min\left\{\frac{\left(\frac{c_i}{s}\right)}{w_i}\right\} = \min\left\{\frac{c_i}{w_i}\right\} \quad (3)$$

WLC 알고리즘이 지금까지 설명한 알고리즘 중 가장 신뢰할 수 있는 알고리즘이라고 할 수 있으나 이 또한 클러스터의 상태정보가 수반되지 않은 단순 획일적인 부하분산 방식에서 벗어나지 못하고 있다. WLC 알고리즘의 가중치(Weight)는 노드의 성능을 예측하여 미리 지정한 정적인 가중치이다. 정적인 가중치를 이용함으로써 부하가 어느 한 서버로 편중되는 경우가 발생하더라도 이를 보정할 하는 능력이 없기 때문에 부하의 편중이 가중 될 수 있다. 이에 대한 대안으로 PWLC 알고리즘은 주기적으로 노드의 부하 정도를 측정하여 가중치를 갱신하여 잘못된 가중치에 의해 노드간의 부하의 불균형 현상이 발생하더라도 보정하는 능력을 가지도록 하였다. 또한 노드간의 부하 불균형 현상의 발생을 낮게 유지하기 위해 다음의 식 (4)에서처럼 가중치를 산정하는 가중치 산정주기 사이에 발생하는 평균적인 요청 수를 각 노드의 가중치의 비율로 할당하는 방식을 적용하였다.

c_{sp}를 가중치 산정주기 동안 발생하는 평균 도착 수, w_{bace}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

$$c_{sp} \sim \sum_{i=1}^{n} w_i \quad (4)$$
$$\alpha = l_{\max} - l_{\min}$$
$$w_{bace} = \frac{c_{sp}}{n}$$

라는 조건을 만족 시킬 경우

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

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. 이러한 가중치 산정 방식은, 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[11-13].

3.3 PWLC 알고리즘과 DWRR 알고리즘의 부하 할당 과정

PWLC 알고리즘과 DWRR 알고리즘의 부하할당 방식은 그 순서에 있어서 분명한 차이를 보이고 있는데, 본 연구에서 탐색하고자 하는 연구 목적이기도 하다. PWLC 알고리즘과 DWRR 알고리즘의 부하 할당 순서에 의한 차이점은 분명하다. PWLC 알고리즘은 현재의 연결수가 동적으로 변화하는 과정에서 얻어진 변수이나, DWRR은 일정 기간 동안 접속을 부과한 후, 이후부터는 RR 알고리즘 방법을 적용한다는 개념이다. 즉 초기에 부하의 차이를 극복하려 한다. 예를 들어, 각 서버의 부하(연결 수)를 $load_{t,1} : load_{t,2} : load_{t,3} : load_{t,4}$ = 1 : 2 : 3 : 2라고 가정할 때, DWRR 알고리즘은 $w_{t,1} : w_{t,2} : w_{t,3} : w_{t,4}$ = 6 : 3 : 2 : 3이라는 가중치가 부여되어, 다음 부하가 할당되는 순서는

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

이다. 여기에서 n_i는 클러스터 내의 노드를 의미한다.

반면에 PWLC 알고리즘은 $w_{t,1} : w_{t,2} : w_{t,3} : w_{t,4} = 6 : 3 : 2 : 3$이라는 가중치가 부여될 때, 다음 부하에 대하여 할당되는 순서는

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

로서 DWRR 알고리즘과 확연한 차이를 보이고 있다[11-13].

본 연구에서는 PWLC 알고리즘과 DWRR 알고리즘의 효율성을 검증하기 위하여 주기적으로 각 알고리즘을 적용한 서버들의 상태 정보를 취하고, 이를 근거로 부하를 할당하는 실험을 하고자 한다.

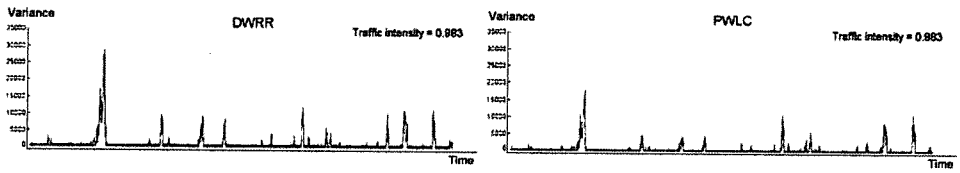
4. 알고리즘 비교 및 분석

본 연구에서는 Li 등[3]에 의해 제안된 DWRR 알고리즘과 본 연구에서 제안된 모델인 PWLC 알고리즘을 비교 분석하기 위하여 다양한 실험을 하였다. 실험 환경은 C++를 이용하여 모의실험을 수행하였다. 실험에 사용된 매개변수는 평균 서비스 시간과 각 실험 별로 다양한 도착 간격과 가중치 설정 주기를 적용하였다. 실험 데이타는 각 조건에 대하여 1회 실험에 100000 개의 요청을 생성하였고, 이를 10회 반복하여 대기시간에 대한 평균과 분산을 비교하여 두 알고리즘의 비교 분석하였다.

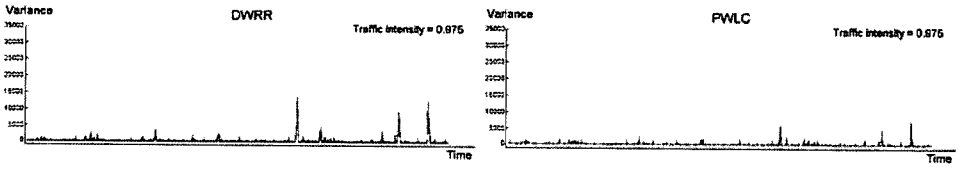
두 알고리즘의 공통점은 일정한 시간마다 시스템 정보를 획득한다는 것이며, 다른 점은 정보를 획득한 후에 정보를 이용하는 부분에서 상당한 차이점을 두고 있다. 부연하면 PWLC 알고리즘과 DWRR 알고리즘은 클러스터 시스템 내의 노드들의 부하 정도를 파악하여 주기적으로 노드들의 상태에 따른 가중치를 부여하는 과정은 유사하나, 부하를 할당하는 순서에서는 다소 차이가 있다. Nadimpalli 등[14]의 연구에 따르면 클러스터 내의 노드들에 대하여 적절한 가중치 산정 주기를 알 수 있다면 DWRR 알고리즘이 기존의 정적인 부하분산 알고리즘인 RR 알고리즘 보다 더 우수하다고 주장하였다. 가장 이상적인 경우는 매 서비스 요청 시점에서 노드들의 부하 정도를 측정하는 것이 가장 좋은 방법이 될 수 있으나 실제 시스템에서 오히려 추가적인 부하를 부여할 수 있으므로 이에 대한 대안으로 일정 간격을 두어 주기적으로 가중치를 부여하였다. 따라서 실험의 최종 목적은 시스템 정보를 획득하는 간격이 중요한 요소라고 할 수 있다.

따라서 본 연구에서는 클러스터 시스템의 상황을 적용하기 위하여 시스템 전체에 과부하(heavy load) 상태를 가정할 것이며, 두 알고리즘에 대하여 동일한 노드 수, 즉 실제 서비스를 담당하는 서버를 4개로 시스템을 구성하였고, 부하분산을 담당하는 Director를 통하여 할당하게 된다. 서비스 요청에 대한 분포를 동일하게 가정하였고, 요청에 대한 서비스 시간 또한 동일하게 가정하였다.

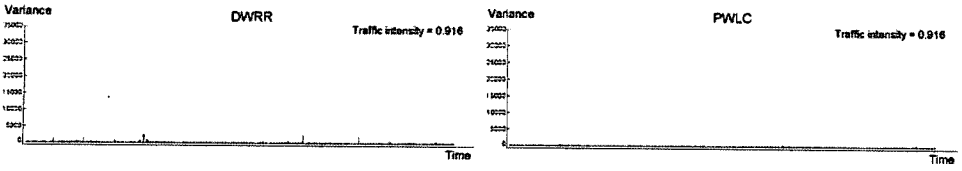
그림 2(a), 2(b), 2(c)는 가중치 설정주기를 30초로 고정하고, 과부하상태를 극과부하 상태(extreme heavy



(a) 극과부하 상태의(ρ = 0.983) DWRR과 PWLC의 분산



(b) 과부하 상태의(ρ = 0.975) DWRR과 PWLC의 분산



(c) 저과부하 상태의(ρ = 0.916) DWRR과 PWLC의 분산

그림 2

load), 과부하 상태, 저과부하 상태로 구분하여 DWRR 알고리즘과 PWLC 알고리즘을 동일하게 적용하여 평균 대기시간에 대한 분산의 비교를 보여주고 있다. 저과부하 상태보다 더 낮은 부하에 대하여서는 분산의 차가 미미하여 생략하였다. 그림 2(a), 2(b), 2(c)에서 보듯이 PWLC 알고리즘이 DWRR 알고리즘보다 상대적으로 효율적임, 즉 반응시간이 작다는 것을 알 수 있다.

이러한 이유는 DWRR 알고리즘과 PWLC 알고리즘의 부하분산 방식의 차이 때문이다. 전자의 경우는 매 주기의 초기에 각 서버의 부하 차를 극복하고 남은 주기 동안은 라운드 로빈 방식으로 요청을 할당하지만, 후자의 경우는 주기 전체 동안 서서히 부하를 감소시키는 방식을 채용하고 있다. 때문에 DWRR 알고리즘은 부하를 극복하고자 하는 주기의 초기 시점에서 특정 서버에 요청을 편중시킬 수 있고, 이것은 대기시간의 급격한 증가를 야기하여 전체 클러스터 시스템의 효율성을 저하시킬 수 있다. 하지만 PWLC 알고리즘은 전체 주기 동안 각 서버에 골고루 할당되므로 DWRR 보다 효율적인 결과를 보였다.

표 1은 DWRR 알고리즘과 PWLC 알고리즘의 평균 대기시간을 나타내며, 5 종류의 부하 상태로 구분하였다. 결과 역시 PWLC 알고리즘이 DWRR 알고리즘보다 효율적임을 보여주고 있다. 표 1에서 간과하지 말아야 할 점이 있다면, 대기시간을 최소화하는 최적의 가중치 설정 주기를 실험적으로 가능함을 보여주고 있다는 것이다.

표 1에서 *표시한 부근이 최적의 가중치 설정주기임을 보여준다. 또한 PWLC 알고리즘과 DWRR 알고리즘 모두 과부하 상태, 즉 traffic intensity가 높은 경우는 가중치 설정주기가 클수록 대기시간도 증가함을 알 수 있다. 반대로 traffic intensity가 낮은 경우는 가중치 설정주기가 클수록 대기시간이 낮아짐을 알 수 있다. 이러한 이유는 traffic intensity 낮고 가중치 설정주기가 짧은 경우는 알고리즘의 반영효과가 미미하지만, traffic intensity 낮고 가중치 설정주기가 길어질수록 알고리즘의 적용 효과가 충분히 반영되기 때문에 대기시간이 감소하는 것이다. 참고적으로 Li 등[3]에 의해 제시된 계수, k=2를 사용하여 DWRR 알고리즘과 PWLC 알고리즘에 적용하였다.

표 2는 다양한 가중치 설정주기에서, [3]에 의하여 detecting period라고 정의, 두 알고리즘의 대기시간의 평균과 분산을 조사하였다. 이 역시 PWLC 알고리즘이 대체적으로 효율적인 것으로 나타났다. 일반적으로 가중치 설정주기가 짧으면, 시스템 정보를 파악하는데 추가된 부하가 빈번히 미칠 수 있다는 단점도 있으나, 한편으로는 시스템 정보를 수시로 파악할 수 있어서 장점으로도 작용할 수 있다. 이와 반대로 가중치 설정주기가

표 1 Traffic Intensity에 따른 가중치 설정주기 별 대기시간

Traffic Intensity (ρ)	가중치 설정주기	대기시간	
		DWRR	PWLC
0.983	10	20.4975	16.8257
	20	**17.5057***	**15.9302***
	40	20.0810	15.9632
	60	24.0230	16.0001
	80	30.4590	16.0144
	100	38.0451	16.0405
	120	45.7711	16.0504
0.975	10	11.8726	10.3469
	20	**11.2888***	**10.1895***
	40	13.7479	10.2238
	60	17.8589	10.2401
	80	24.4945	10.2651
	100	27.3277	10.2862
	120	39.5514	10.2933
0.916	10	**3.2384***	2.7720
	20	3.2801	**2.7644***
	40	3.6076	2.7755
	60	3.8909	2.7851
	80	4.1572	2.7902
	100	4.3904	2.7933
	120	4.4650	2.7950
0.75	10	0.8012	0.6206
	20	0.7080	0.6116
	40	0.6449	0.6088
	60	0.6273	0.6029
	80	0.6186	0.6020
	100	0.6140	0.5997
	120	**0.6108***	**0.5995***
0.5	10	0.1893	0.1231
	20	0.1517	0.1241
	40	0.1321	0.1189
	60	0.1254	0.1169
	80	0.1224	0.1158
	100	0.1202	0.1151
	120	**0.1189***	**0.1147***

길면, 부하분산에 관한 알고리즘의 기여도가 크게 떨어지게 된다. 이를 반영하듯이 DWRR 알고리즘의 특성상 가중치 산정 시 각 서버의 부하가 불균형하면 할수록 가중치의 차는 더욱 커지기 때문에 한쪽으로 부하가 편중되어 전체 클러스터 시스템의 대기시간을 증가시키는 결과를 낳게 된다.

또한 DWRR 알고리즘은 가중치 설정주기의 초기에 각 서버에 할당된 가중치를 이용하여 부하를 분산하고, 부하분산이 끝났다고 판단되는 시점부터는 RR 알고리즘 방식으로 부하를 분산하므로 서버 간의 가중치 차이가 큰 경우, 초기의 부하분산 과정의 마지막 부분에서

표 2 가중치 설정 주기 별 평균과 분산(Traffic intensity = 0.975)

	가중치 설정주기	7	14	28	56	113	225
DWRR	평균	16.6704	11.2445*	11.7595	17.0070	33.6596	72.2860
	분산	639.9102	161.8191	135.3660*	327.8502	1950.5305	12491.4973
PWLC	평균	11.3427	10.2144	10.2004*	10.2325	10.2831	10.3685
	분산	223.1972	128.4897	109.3115	108.4302*	108.9241	110.7532

클러스터 노드 중 가중치가 가장 큰 노드로 연속적으로 할당되어 이때의 영향이 다음 주기에도 영향을 주기 때문에 전체 클러스터 시스템의 성능에 떨어뜨리는 요인이 되는 것이다. 특히 과부하 상태에서 DWRR은 더욱 효율이 떨어지게 되는 것이다.

5. 결 론

본 연구에서는 클러스터 시스템에서 부하할당방식인 WLC 알고리즘을 변형한 PWLC 알고리즘을 DWRR 알고리즘과 비교하여 효율성을 분석하였다. 주기적으로 시스템의 부하를 측정하여 가중치를 산정하는 방법은 동일하나, 이를 근거로 부하를 할당하는 순서에서 의하여 차이가 있음을 알 수 있다. 본 연구에서는 PWLC 알고리즘과 DWRR 알고리즘에 대한 다양한 실험을 통하여 각 알고리즘의 효과와 영향을 분석하였다. 그 결과 PWLC 알고리즘이 전반적으로 더 효율적임을 알 수 있었고, 특히 traffic intensity가 높은 경우 그 성능의 차이가 더욱 뚜렷함을 알 수 있었다. DWRR 알고리즘에서 적용된 가중치 설정주기를 적용한 경우에도 같은 결과를 확인할 수 있었다.

따라서 가중치 설정주기가 짧으면, 시스템 정보를 파악하는데 추가된 부하가 빈번히 미칠 수 있다는 단점도 있으나, 한편으로는 시스템 정보를 수시로 파악할 수 있어서 장점으로도 작용할 수 있다. 이와 반대로 가중치 설정주기가 길면, 부하분산에 관한 알고리즘의 기여도가 크게 떨어진다고 할 수 있다.

참 고 문 헌

[1] Patrick Killelea, *Web Performance Tuning*, O'Reilly 2 edition October, 2001.

[2] IBM corp, *IBM Technical Report CS 260*, www.ibm.com, 2004.

[3] 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.

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

[5] Gregory F. Pfister, *In Search of Clusters Second Edition*, Prentice-Hall PTR, New Jersey, 1998.

[6] microsoft corporation, *Server Cluster Architecture*, http://www.microsoft.com/windows.netserver/techinfo/overview/servercluster.mspx, 2002.

[7] Valeraia C, Michele c, Philip S. Yu, *Dynamic load balancing on web-server systems*, IEEE Internet Computing, pp. 28-39, May, 1999.

[8] Shan Hong Li, Sung Ki Kim, Yong hee Na, Byoung Joon Min, *Analysis of the Load Balancing Algorithms according to the Request Pattern on the LVS Cluster Systems*, Korean Information Proceeding Society, Vol. 9-A No. 2, 10, 2002.

[9] Sun Young Park, Do Hyun Park, Jun Won Lee, Jeong Wan Jo, *Back-end Prefetching Scheme for Improving the Performance of Cluster-based Web Servers*, Korean Information Proceeding Society, Vol. 29-A, No. 5, pp. 265-273, 6. 2002.

[10] Jin Young Kim, Seong Cheon Kim, *Effective Prioritized HRW Mapping in Heterogeneous Web Server Cluster*, Korean Information Proceeding Society, Vol. 30-A, No. 12, pp. 708-713, Dec 2003.

[11] Y. Rhee, S.C Kim, *A study of distributing the load of the LVS clustering system based on the dynamic weight*, Korean Information Proceeding Society, Vol. 8-A, No. 4, pp. 299-310, Dec 2001.

[12] Y. Rhee, *A Study of the load distributing algorithm on the heterogeneously clustered web system*, Korean Information Proceeding Society, Vol. 10-A, No. 3, pp. 225-230, Aug. 2003.

[13] 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.

[14] Nadimpalli S, Manumdar S. *Techniques for achieving high performance Web Servers*. International conference on IEEE Parallel Processing, p. 233-241, 2000.

김 석 찬

1999년 계명대학교 산업공학과 학사. 2001년 계명대학교 산업공학과 석사. 2001년~현재 계명대학교 산업공학과 박사과정. 관심분야는 Network 모델링 및 성능 분석, Web 서버 용량 계획 클러스터 시스템, 네트워크 관리 시스템

이 영
1983년 고려대학교 산업공학과 학사. 1985년 고려대학교 산업공학과 대학원 석사 1990년 오클라호마대학교 산업공학과 석사(전공 : IE). 1994년 노스캐로라이나 주립대학교 박사(전공 : OR & CS). 1990년~1995년 연구원, Center for Comm. Signal Processing at NCSU. 1995년~1998년 수석연구원, 삼성 SDS, 정보처리기술 연구소. 1998년~현재 계명대학교 산업공학과 조교수. 관심분야는 Network 모델링 및 성능분석, Web 서버 용량 계획, 전자 상거래 지불시스템, 전자결재시스템