2001년 한국정보처리학회 추계 학술발표논문집 제8권 제2호

부하를 고려한 동적 가중치 기반 라운드로빈 스케쥴링 알고리즘

김성, 김경훈, 류재상, 남지승
전남대학교 대학원 컴퓨터공학과

e-mail:sung@kjssa.co.kr

Dynamic Weight Round Robin Scheduling Algorithm with Load

Sung Kim, Kyong-Hoon Kim, Jae-Sang Ryu, Ji-Seung Nam
Dept of Computer Engineering Chonnam National University

요약

멀티미디어 스트리밍 서비스를 제공하는 서버의 동적 부하분산을 위한 동적 가중치 기반 라운드 로빈 스케쥴링 알고리즘을 제안한다. 기존의 가중치 기반 라운드로빈 알고리즘은 서버의 처리 용량만을 이용하여 가중치를 부여하므로 요청이 폭주할 경우 동적 부하 불균형을 갖게 된다. 동적 부하 불균형을 해결하기 위해 제안한 동적 가중치 기반 라운드로빈 알고리즘은 서버의 처리 용량뿐만 아니라 서버의 동적 부하를 이용하여 가중치를 부여하므로 동적 부하 불균형에 잘 적응하여 부하를 균형있게 조절할 수 있다. 제안한 알고리즘은 각 서버의 처리용량을 기준으로 가중치를 계산하고 동적으로 변하는 서버의 부하값에 가중치를 적용한다. 그 결과 동적 부하 불균형 문제를 해결했으며, 더 세밀한 부하 조절 기능을 수행할 수 있었다.

1. 서론

정보화 시대로 접어들면서 인터넷의 이용자가 계속적으로 증가하고 있고 사용자의 요구 또한 다양해지고 있다. 최근 멀티미디어 등 대용량 컨텐츠에 대한 수요의 증가로 인해 데이터의 트래픽은 기하급수적으로 늘어남으로 서버시스템 뿐만 아니라 네트워크에 많은 부하를 주게 된다.

서버의 성능과 부하를 고려하여 로드밸런싱 함으로써 서버의 병목을 줄이고 많은 클라이언트에게 더 나은 서비스를 제공할 수 있다. 이러한 로드밸런싱 방법들은 목적이 약간씩 다르지만 서버의 시스템 자원을 지준으로 부하 분산을 하고 서버의 병목을 해결하는 것이 일반적인 목적이다. 클라이언트의 요청을 스케쥴링하는 스케쥴링 알고리즘은 라운드로빈, 가중치 기반 라운드 로빈, 최소 접속, 가중치 기반 최소 접속 스케쥴링 알고리즘이 있다.

라운드 로빈 스케쥴링 알고리즘은 시스템의 성능을 고려하지 않아 다른 처리 용량을 갖는 서버에 적용하기 힘들다. 그리고 최소 접속, 가중치 기반 최소접속 스케쥴링은 스트리밍 서비스를 하는 서버의 부하는 클라이언트 수에 크게 영향을 받지 않으므로 적용하기 힘들다. 그래서 서버의 처리 용량을 이용한 가중치 기반 라운드 로빈 방식을 택하였는데 가중치 기반 라운드 로빈은 요청이 폭주할 경우 서버의 동적 부하 불균형 상태가 생길수 있는 단점이 있다. 그래서 동적으로 변하는 부하 불균형 상태를 체크하고 이를 서버 처리용량을 기반으로 계산된 가중치에 적용한 동적 가중치 기반 라운드 로빈 방식을 제안하였다.

제안한 라운드 로빈 방식은 부하 불균형이 심각해질 수 있는 문제점을 갖는 가중치 기반 라운드 로빈 방식의 문제점을 개선하여 부하의 불균형이 발생하더라도 점차 부하를 균형있게 맞추어 부하의 불균형 문제를 해결하였으며 더욱 세밀한 부하 조절기능을 수행할 수 있다.

2. 기존의 스케쥴링 알고리즘

스케쥴링 알고리즘은 외부에서 들어온 요청을 로드밸런서가 어떻게 서버들에게 나누어 주는가를 정의하는 것이다. 기존의 스케쥴링 알고리즘은 다음과 같다.

1) 라운드 로빈 스케쥴링[1]

사용자 요구를 차례대로 각 서버에 균등하게 분배하는 방식으로 클라이언트의 새로운 연결 요청을 로드밸런서는 순차적으로 균등하게 연결 설정을 수행한다. 서버 커넥션 수나 응답시간에 상관없이 그룹내의 모든 서버를 동일하게 처리하여 일반적인 구성에 있어서 다른 알고리즘에 비해서 가장 빠르다는 장점이 있다. 그러나 서버사이에 동적인 부하 불균형이 심각해질수 있다.

2) 가중치 기반 라운드 로빈 스케쥴링[1]

서버에 서로 다른 처리 용량을 지정할 수 있다. 기본 가중치는 1이며 서버의 처리 용량에 따라 계산된 가중치를 부여하고 그 가중치에 따라 서버의 선택 순서를 변형하는 방식이다. 이 스케쥴링을 사용하면 서버에서 네트워크 접속을 카운트할 필요가 없고 동적 스케쥴링 알고리즘보다 스케쥴링 부담이 적으나 요청이 폭주할 경우 실제 서버사이에 동적인 부하 불균형 상태가 생길 수 있다.

3) 최소 접속 스케쥴링[1]

연결된 클라이언트의 수가 가장 적은 서버로 사용자 요구를 연결하는 방식으로 모든 서버가 균등한 트래픽을 유지하기 위해서 처리 속도가 빠른 서버가 더 많은 접속을 받게 된다. 각 서버에서 동적으로 실제 접속한 숫자를 세어야 하므로 동적인 스케쥴링 알고리즘 중 하나이다. 서버들의 성능이 비슷하게 구성되었을 경우에 가장 효과적인 트래픽 분산이 가능하다.

4) 가중치 기반 최소 접속 스케쥴링[1]

가중치 기반 최소 접속 스케쥴링은 최소 접속 스케쥴링에 서버의 성능 가중치를 추가한 것으로, 요구가 동일한 경우 가중치가 높은 서버에서 더 많은 요구를 받게 설계되어 있다. 예를 들어 가중치가 3인 서버가 가중치가 1인 서버보다 3배가 더 많은 요구를 받게 된다. 각 실제 서버에 가중치를 할당함으로써 관리자는 더 빠른 서버가 더 많은 트래픽을 받도록 운영할 수 있다.

3. 동적 가중치 기반 라운드로빈 스케쥴링 알고리즘

서버의 동적 부하 불균형 문제를 해결하기 위해 제안한 동적 가중치 기반 라운드로빈 스케쥴링 알고리즘은 서버의 성능뿐만 아니라 실시간으로 변하는 부하를 이용하므로 부하 불균형 문제를 해결할 수 있다. 또한 기존의 방식의 가중치는 서버가 클라이언트의 요청을 받는 수를 나타내지만 제안한 방식의 가중치는 서버가 클라이언트의 요청을 받기 위해 할당된 시간이므로 기존 방식보다 더 세밀한 부하 조정기능을 수행 할 수 있다. 동적 가중치 기반 라운드로빈 스케쥴링 알고리즘을 로드밸런싱 시스템에 적용하여 구현하였다. 적용한 로드밸런싱 시스템은 로드밸런서와 서버로 구성되어 있다. 로드밸런서의 기능은 서버와 연결 및 데이터 송수신기능, 맵핑테이블 작성기능, 가중치 계산 및 가중치를 이용한 라운드로빈 기능, 리다이렉트기능을 갖으며 서버는 로드밸런서에 연결 및 데이터 송수신 기능, 서버의 부하 체크기능, Alarm기능을 갖고 있다.

시스템의 구조는 그림 1과 같다.

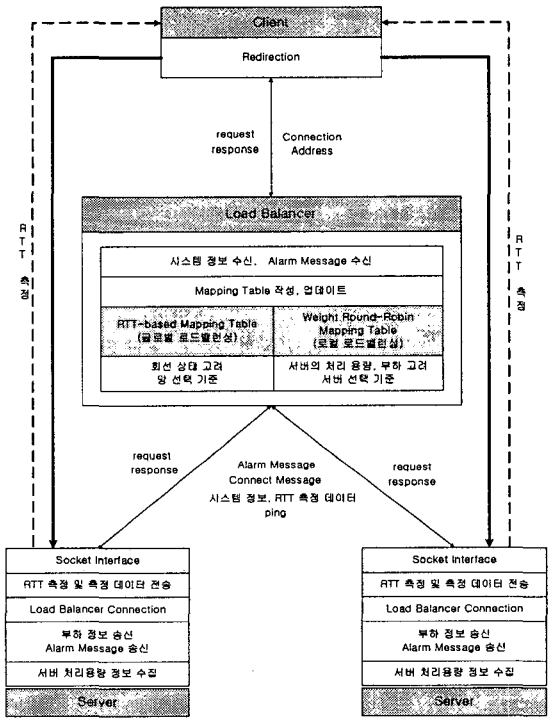

그림 1 로드밸런싱 시스템 구조

로드밸런서에 각 서버와 연결을 설정하고 서버로부터 서버의 처리 용량과 부하 정보를 수신한다. 서버의 처리 용량과 부하를 이용하여 맵핑테이블을

작성하고 주기적으로 계속 수신되는 부하정보를 이용하여 맵핑테이블의 가중치 값을 갱신한다. 로드밸런서는 각 서버에 할당된 가중치를 시간으로 설정하고 그 시간만큼 라운드로빈되어 클라이언트의 요청을 분배한다. 클라이언트의 요청이 들어오면 네트워크에 대한 RTT를 이용하여 네트워크를 선정하고 선정된 네트워크의 여러서버 중 현재 연결을 대기하고 있는 서버을 선택하여 클라이언트 요청을 리다이렉트한다.

제안한 라운드 로빈 스케줄링 알고리즘을 위한 변수는 표 1과 같다.

변수명	변수값
RR_T	서버에 할당된 라운드로빈 시간 각각의 서버에 할당된 기본적인 시간으로 초기 W_C값과 같음
S_{CNT}	서버 수
L_C	서버의 부하
L_T	서버 부하의 합
I_C	서버의 처리 용량
I_T	서버 처리 용량의 합
S_L	서버의 상대적 부하
PT_C	능력 가중치
PT_T	능력 가중치의 합
W_C	라운드로빈 가중치 값 연결설정을 위해 서버에 할당된 시간

표 1 동적 가중치 기반 라운드 로빈 스케줄링 알고리즘을 위한 변수

로드밸런싱을 위해 가중치를 구하는 알고리즘은 아래와 같다.

$$L_T = \sum_{i=1}^{S_{CNT}} L_i$$

$$I_T = \sum_{i=1}^{S_{CNT}} I_i$$

$$S_L = \frac{(\frac{L_T - L_C}{L_T})}{S_{CNT}-1}$$

$$PT_C = \frac{I_C}{I_T} * (RR_T * S_{CNT})$$

$$PT_T = \sum_{i=1}^{S_{CNT}} (PT_C * S_L)$$

$$W_C = (\frac{PT_C * S_L}{PT_T}) * (RR_T * S_{CNT})$$

동적 가중치 기반 라운드로빈 스케줄링 알고리즘은 서버의 성능과 부하를 고려하여 $RR_T * S_{CNT}$으로 계산된 전체 라운드로빈 값을 균등하게 분배한다. 부하가 많은 서버는 가중치를 낮추고 부하가 적은 서버는 가중치를 높이기 위해서 S_L을 이용한다. 그래서 서버의 상대적 부하를 이용하여 서버의 가중치에 영향을 주었다. $(L_T - L_C)/L_T$에서 얻은 결과는 $S_{CNT}-1$ 퍼센트의 값이 얻어지므로 이 값을 $S_{CNT}-1$로 나누었다. PT_C는 서버에 할당된 가중치 값으로써 서버의 용량에 비례하여 라운드로빈 시간을 나누어 적용하였다. PT_T는 부하와 처리 용량을 고려한 라운드로빈 할당 시간인 PT_C 와 동적으로 변하는 S_L을 곱한 값의 합이다. 각 서버의 부하와 처리 용량을 고려한 라운드로빈 할당 시간을 그 전체의 합 PT_T로 나누어 서버에 할당될 비율을 구한 후 이를 라운드로빈 1 사이클 시간인 $RR_T * S_{CNT}$를 곱하여 W_C값을 얻을 수 있다.

3. 실험 및 결과분석

실험 환경은 서로 다른 처리 용량을 갖는 서버 2대를 네트워크에 배치한 것으로 가정하여 시뮬레이션하여 결과를 얻었다. 부하는 HTTP 서비스 세션에 대한 부하와 스트리밍 서비스에 대한 부하로 나누었고 HTTP 서비스 세션에 대한 부하는 10KB, 스트리밍 서비스 세션에 대한 부하는 200KB로 설정하였다. 표 2는 설정된 서버의 처리 용량과 그에 따른 부하를 나타낸다.

서버	서버A	서버B
처리 용량	30MB	60MB
HTTP 세션 부하(10KB)	0.033%	0.016%
스트리밍 세션 부하(200KB)	0.666%	0.333%

표 2 서버에 처리 용량에 따른 부하

시뮬레이션은 기존의 가중치 기반 라운드 로빈 스케줄링 알고리즘과 제안한 동적 가중치 기반 라운드로빈 스케줄링 알고리즘에 대한 실험을 하였다. 그리고 서버에 대한 부하는 클라이언트의 연결수의 30%를 스트리밍 서비스 연결수로 설정하여 부하를 주었고 서버의 부하가 10%되는 시점에서 A서버에 B서버의 2배의 과부하를 주어 실험한 결과를 얻었다.

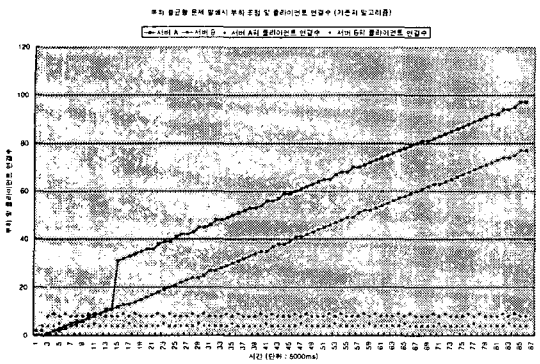

그림 2 부하 불균형 문제 발생시 부하 조정 및 클라이언트 연결수 (기존의 알고리즘)

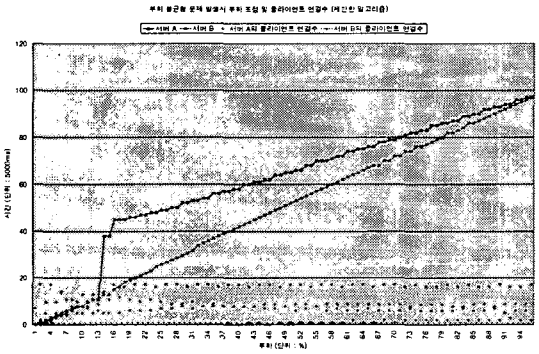

그림 3 부하 불균형 문제 발생시 부하 조정 및 클라이언트 연결수 (제안한 알고리즘)

기존의 알고리즘은 서버의 처리용량을 기준으로 가중치를 부여한다. 부여된 초기 가중치는 1이고 이후 서버의 처리용량을 고려하여 서버A에는 1을 서버B에는 2의 가중치를 부여한다. 그래서 1:2의 비율로 클라이언트가 연결되는 것을 볼 수 있다. A와 B서버 모두에 똑같이 클라이언트 연결의 30%를 스트리밍 서비스 연결설정 함으로써 A와 B의 부하차가 없는 것을 볼 수 있다. 그러나 A서버에 B서버의 2배의 과부하를 주었을 때 부하 불균형이 발생하지만 부하를 고려하지 않으므로 클라이언트의 연결수를 일정하게 유지하므로 부하를 균형있게 조절하지 못한다. 제안한 알고리즘은 초기 가중치는 각 서버에 할당된 시간이 되고 이후 서버의 처리용량을 고려하여 서버A에는 전체라운드 로빈 시간의 33%를 할당하고 B서버에는 전체라운드 로빈 시간의 67%를 할당한다. 기존 방식과 똑같이 부하를 주었을 때 과부하 발생하기 전은 기존방식과 같은 결과를 얻을 수 있었다 그러나 부하의 불균형이 발생하면 가중치 값을 변화시켜 클라이언트의 연결수를 조절함으로써 부하를 균형있게 조절하는 것을 볼 수 있다.

4. 결론 및 향후 연구

본 논문에서 제안한 동적 가중치 기반 라운드로빈 스케줄링 알고리즘을 적용함으로써 서버의 처리 용량과 서버의 부하를 이용하여 로드밸런싱 할 수 있었다. 그 결과 기존의 가중치 기반 라운드로빈 방식은 실시간적으로 변하는 부하에 적응하지 못하여 시스템에 부하 폭주시 시스템의 부하 불균형 문제를 해결하지 못했으나 제안한 가중치 기반 라운드로빈 방식은 실시간적으로 변하는 부하에 잘 적응하여 시스템에 부하 폭주시 부하의 불균형 문제를 해결하였다. 앞으로 서버의 부하를 체크하고 전송하는 작업을 일정 주기로 하였는데 이를 보완하여 서버의 부하차가 심할 때만 전송함으로써 로드밸런서와 서버사이의 데이터 송수신량을 줄이고 알람기능 및 페일오버 기능에 대한 연구를 할 계획이다.

5. 참고 문헌

[1] 리눅스 가상 서버 기술
http://www.clunix.com/support/sw/about_lvs/index.html

[2] T. Brisco, "DNS Support for Load Balancing", RFC1794, 1995. 4.

[3] Valeria Cardellini and Michele Colajanni and Philip S. Yu, "Redirection Algorithms for Load Sharing in Distributed Web-server Systems", IEEE 1999.

[4] Network Load Balancing Technical Overview White Paper

[5] Valeria Cardellini, Michele Colajanni, Philip S. YU, "Dynamic Load Balancing On Web-Server System", IEEE Internet Computing, June 1999.

[6] Rajkumar Byyya, "high performance cluster computing", Prentice Hall PTR, 1999.