웹서버 및 디스패쳐의 상태 정보를 이용한 경험적 동적 부하 분산 기법

구현우°, 홍영식
동국대학교 컴퓨터공학과
{ hwgoo°, hongys }@dongguk.edu

Heuristic and Dynamic Web Load Balancing Using the information of Web server and Dispatcher

Hyun-Woo Koo, Young-Sik Hong
Department of Computer Engineering, Dongguk University

요 약

인터넷 사용에 따른 수많은 정보의 요청을 처리하기 위해 웹 서버 클러스터 기법에 대한 많은 연구가 진행되어 왔다. 웹 서버 클러스터 기법은 사용자들의 많은 요청을 하나의 서버에서 처리하지 않고 다수의 서버가 처리한다. 따라서 클러스터를 구성하는 다수의 서버가 클라이언트의 요청을 적절하게 분배하여, 웹 서버가 응답해야 할 부하를 분담해야 한다. 특히 금융 웹 서비스와 같은 특정 시간에 부하가 집중되는 환경에서 최소한의 클라이언트의 요청에 대한 서비스를 제공해야 한다. 이번 연구에서 DNS 기반 부하 분산 기법 및 디스패쳐 기반 부하 분산 기법을 이용한 경험적 동적 부한 분산 기법 적용을 통해 각 기법의 약점을 줄이고 부하가 급격히 증가한 시점에 전체 시스템의 성능을 떨어뜨리지 않는 기법에 대하여 논한다. 혼합 기법을 사용하기 위해 웹서버 및 디스패처의 부하 정보를 수집하고 실험을 통해 얻은 한계치를 결정한다. 그리고 이 한계치를 이용하여 부하 분산 기법을 변경함으로써 최소한의 서비스를 제공할 수 있다. 실험을 통해 제시된 부하 분산 기법이 높은 부하 상태를 보이는 시점에서 최소한의 서비스를 사용자에게 제공하여 최악의 경우를 막을 수 있음을 보인다.

1. 서 론

인터넷의 보급과 사용 증가로 인터넷 사용자들의 웹 정보 요청이 폭증하고 있다. 이로 인하여 웹 서비스를 제공하는 웹 사이트는 계속적으로 더 많은 수의 요청을 처리해야 하고, 더욱 빠른 처리를 요구 받고 있다. 또 웹 서비스 제공자는 인터넷의 발전으로 다양한 서비스를 제공하고자 한다. 따라서 한 요청에 대한 응답에 많은 부수적인 작업이 뒤따르게 되어 한 응답에 필요한 서버의 능력을 더 많이 필요로 한다.

웹 발전의 초기에는 증가하는 요청을 수용하기 위하여, 서버에 높은 성능의 프로세서를 장착하거나 슈퍼컴퓨터와 같이 여러 개의 CPU를 이용하거나 네트워크 대역폭을 증가시키는 것과 같은 시스템의 하드웨어 성능을 향상시키는 방법을 이용하였다. 그러나 단일 서버의 성능 향상은 비용이 높고, 증가하는 요청을 수용하는 데에는 한계가 있다. 이에 비용 대비 성능이 우수하면서 서버의 확장성도 우수한 웹 서버 클러스터 기법에 대하여 많은 연구가 진행되고 있다.[1, 6, 8]

웹 서버 클러스터 시스템은 사용자들의 많은 요청을 하나의 서버에서 처리하지 않고 다수의 서버가 처리한다. 따라서 클러스터를 구성하는 다수의 서버가 클라이언트의 요청을 적절하게 분배하여, 웹 서버가 응답해야 할 부하를 분담해야 한다. 특히 금융 웹 서비스와 같은 특정 시간에 부하가 집중되는 환경에서 최소한의 클라이언트의 요청에 대한 서비스를 제공해야 한다.

이번 연구에서 DNS 기반 부하 분산 기법 및 디스패쳐 기반 부하 분산 기법을 이용하여 동적 부하 분산 기법 적용을 통해 각 기법의 단점을 줄이고 부하가 급격히 증가한 시점에 전체 시스템의 성능을 떨어뜨리지 않는 기법에 대하여 논한다.

본 논문 2장의 관련 연구에서는 웹 서버 클러스터 부하 분산 기법에 대하여 살펴보고, 3장에서는 이 논문에서 제안하고자하는 경험적 부하분산 기법을 살펴본다. 4장에서는 실험과 결과 분석을 통해 제안한 기법에 대해 평

가하고 5장에서 결론 및 향후 연구 과제를 제시한다.

2. 관련 연구

현재까지 연구된 부하분산 방법에는 부하를 분배하는 주체에 따라 크게 클라이언트, DNS 기반 방식, 디스패쳐(Dispatcher) 기반 방식 그리고 서버기반 방식으로 나눌 수 있다. [2, 6]

클라이언트 기반의 부하분산 기법은 클라이언트에서 요청이 발생하면 클라이언트가 알고 있는 서버들 중에서 하나가 선택되어 그 서버로 요청을 보내고 그 서버로부터 응답을 받는 형식이다. 이 방식은 웹 서버 클러스터의 부하 분산에 대한 부담을 덜어주는 이점이 있지만 서버들의 부하상태를 파악하기 용이하지 않기 때문에 부하상태 파악을 위한 모듈을 클라이언트에 별도로 추가해야 하는 부담이 있다. 그리고 정확한 부하 확인을 위해 클라이언트와 서버간의 동기화가 필요하다. 또한 서버 추가, 삭제 등의 변경이 있을 경우 클라이언트 측에서도 변경이 있어야 하기 때문에 응용범위가 한정적이다.

DNS 기반 부하분산은 클라이언트가 접근하고자 하는 웹 서비스의 주소를 입력하여 페이지를 요청할 때 DNS에서 도메인에 매핑된 IP주소를 조건에 따라 분배횟수를 달리하여 사용자에게 전달하여 요청을 여러 서버에 분배한다. 일반적인 기법들은 라운드 로빈 알고리즘을 이용한다. 하지만 클라이언트와 DNS의 통신 사용량을 줄이기 위해 도메인과 IP를 캐싱하기 때문에 DNS는 서버로 향하는 모든 클라이언트의 요청을 제어하는데 한계가 있다.

디스패쳐 기반의 부하분산은 클라이언트에서 발생하는 모든 요청은 디스패쳐에서 서버로 분배된다. 디스패쳐는 서버들의 부하상태를 파악하고 있다가 요청이 들어올 때마다 부하가 적은 서버로 요청을 분배한다. 모든 요청이 디스패쳐를 통과하므로 각 서버가 담당하는 부하를 디스패쳐가 제어하고 디스패쳐는 사용자의 요청을 각 서버로 분배한다. 따라서 부하분산 효율이 다른 부하분산 기법보다 높다. 하지만 모든 요청이 디스패쳐를 통과하므로 디스패쳐에 병목현상이 생길 수 있으며, 디스패쳐가 제 기능을 못하게 되었을 때 서버들의 상태와 상관없이 웹 클러스터 환경 전체가 서비스를 하지 못하는 단점이 있다. 병목현상을 해결하고 디스패쳐 고장 시에도 서비스가 지속되도록 복수 디스패쳐를 두는 등의 방안이 연구되었다.[3, 4]

서버 기반의 부하 분산 기법은 DNS를 통해 서버가 결정되고 결정된 서버로 클라이언트의 요청이 발생한 경우 서버에서 전달 받은 요청을 다른 서버로 재 할당하는 방법이다. 대표 적인 서버 기반의 부하 분산 기법은 HTTP 리다이렉션 이다.[5] 클라이언트의 요청을 받은 서버는 자신의 상태를 확인하고 만약 처리를 하지 못하는 상태이면 다른 서버의 주소를 클라이언트에게 전달하고 클라이언트는 다른 서버의 주소로 재요청한다. 이 기법은 앞서 언급한 DNS 기반 부하 분산 기법의 문제점을 해결할 수 있지만 서버와 클라이언트의 재접속을 요구함으로써 응답 시간이 느려지는 단점을 가지고 있다.

본 연구의 목적은 하나의 디스패쳐를 이용하면서 디스패쳐 기반의 장점을 최대한 살리고 만약 클라이언트의 요청이 폭주하는 경우에도 클라이언트에서 발생할 수 있는 최악의 서비스를 막기 위해 DNS 와 서버 기반 부한 분산 기법을 혼합하여 전체 시스템의 최악의 서비스를 방지하는데 있다.

3. 경험적 동적 부하 분산 기법

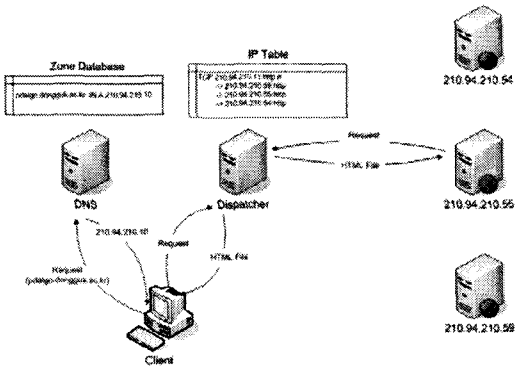

그림 1 디스패쳐 기반 부하 분산 기법

그림 1은 기본적인 부하 분산 기법인 디스패쳐의 라운드 로빈 기법을 보여주고 있으며 경험적 동적 부한 분산 기법에서는 클라이언트의 요청이 빈번하지 않은 경우는 관련 연구에서 살펴본 부하 분산 기법 중 가장 응답 시간이 빠른 디스패쳐 기반 부하 분산 기법을 적용한다.

금융 웹 서비스에서 특정 시간에 서비스 요청이 집중되어 디스패쳐의 성능이 저하되고 최악의 경우 서비스를 전혀 제공하지 못하는 경우가 발생하면 금전적 또는 신뢰의 문제가 발생할 수 있다. 이러한 문제를 해결하기 위해 클라이언트의 요청이 폭주되는 시점에 응답시간은 늦어지지만 최악의 서비스를 막을 수 있는 기법으로 전환함으로써 서비스를 전혀 제공치 못하는 경우를 미리 방지하는 것이 경험적 동적 부하 부산 기법의 목적이라 하겠다.

클라언트의 요청이 폭주하면 일부 클라이언트의 모든 요청을 디스패쳐가 처리하는 것이 아니라 일부의 요청은 DNS에서 클라이언트에게 요청된 웹 서버의 주소를 직접 전달함으로써 디스패쳐의 부담을 줄어 준다. 또한 DNS 기반의 부하 분산 기법은 클라이언트 요청에 대한 완벽한 제어를 하지 못하기 때문에 발생 할 수 있는 유휴한(Idle) 서버가 발생할 수 있기 때문에 HTTP 리다이렉션 기법을 적용하여 클라이언트의 요청을 최대한 서비스 하려고 노력한다.

그림 2는 클라이언트 요청의 폭주에 의해 DNS, 디스패

처, HTTP 리다이렉션 기법의 혼합한 동적 부하 분산 기법을 보여주고 있다.

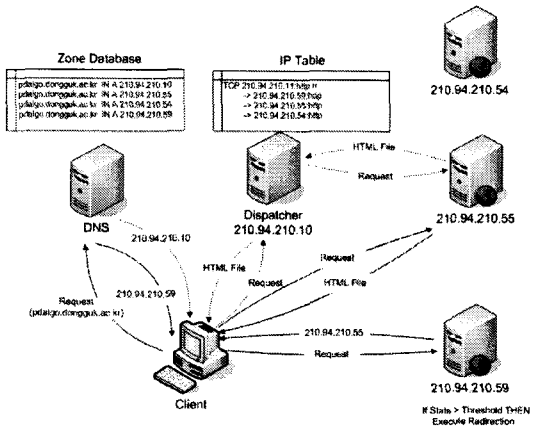

그림 2 경험적 동적 부하 분산 기법

이와 같이 클라이언트의 요청 양에 따라 부하 분산 기법을 동적으로 변경하여 최악의 서비스 상태를 최소화하는데 목적이 있다.

부하 분산 기법을 동적으로 변화시키기 위해 DNS와 디스패처의 모니터링 도구를 이용한다. LVS(Linux Virtual Server)를 이용한 디스패처에서 서버와 클라이언트 간의 연결 정도를 파악하여 아래의 알고리즘을 적용하여 DNS 기반과 혼합된 동적 부하 분산 기법으로 변경을 진행한다.

```
In Dispatcher Approach
  If Avr(ActiveConns) >= Avr(InActConns) for P then
      BindUpdate() // change to Hybrid Approach
      Store MaxActiveConns
  Else Keep Only Dispatcher Approach

In Hybrid Approach
  If Avr(ActiveConns) < Avr(InActConns) for P and
ActiveConns <= MaxActiveConns/2 Then
     BindUpdate() // change to Only Dispatcher Approach
  Else Keep Hybrid Approach
```

각 서버 당 일시적인 접속 수와 일정 기간 파악된 접속 수를 파악하여 전체 접속 수의 평균을 비교하여 폭주가 일정 기간 진행되고 있는지 아니면 폭주가 시작되는지를 파악하여 DNS 기법을 혼합할지 여부를 결정하게 된다. 그리고 혼합 기법이 적용된 경우 디스패처로 집중되는 요청의 수가 현저히 적어지는 시점에 다시 디스패처만 사용하는 부하 분산 기법을 적용하여 클라이언트 응답 시간을 최소화 시킨다.

HTTP 리다이렉션 기법의 적용 여부는 아래의 알고리즘에 따른다.

```
SA = {x |x is all Web Server IPs }
SH = {x | the load of x is exceed the threshold }
SL = {x | x ∈SA ∧ x !∈ SH }
Start Redirection
   If Load(SA_i) >= TH Then
      Set SA_i in SH
   End if
   broadcast Min(Load(SL_j)).IP to SH

Suspend Redirection
   If  Load(SH_i) < TL Then
     Send Redirection suspend command to SH_i
     Remove SH_i in SH
   End if
```

서버의 부하에 대한 정보를 DNS에서 수집하고, 수집된 정보를 위의 알고리즘에 맞춰 부하가 집중된 서버로 요청이 전달되는 경우 해당 서버에 현재 서버 중 유휴한 서버의 주소를 전달하여 리다이렉션 기법을 적용하게 한다. 위의 알고리즘에서 사용된 기호를 살펴보면 SA는 전체 실제 웹서버의 IP 주소 집합의 원소이고 SH는 현재 많은 요청을 처리하고 있는 웹 서버들의 IP 주소 집합의 원소이다. 즉 미리 결정된 한계치의 요청 양을 넘어서 일을 처리하고 있는 웹서버들을 뜻한다. SL는 한계치를 넘지 않은 요청 양을 처리하고 있는 웹 서버들의 집합의 원소이다. 모니터가 웹서버들의 부하를 감시하다 만약 웹 서버들 중에서 하나 이상의 웹서버의 부하가 한계치를 넘으면 리다이렉션이 시작된다. 이때 한계치 부하를 넘는 웹서버들의 주소를 하나의 집합으로 생성시키고 전체 집합에서 SH들을 제외한 집합을 만든다. 한계치를 넘지 않은 웹서버들 중에서 부하가 가장 적은 웹서버를 선정하고 이 웹서버의 주소를 SL들에게 전달하여 SH에게로 전달되는 요청을 부하가 가장 적은 웹서버로 재 전달함으로써 SH들의 부담을 덜어준다.

리다이렉션이 실행되고 있는 웹서버의 부하가 한계치 이하가 되면 리다이렉션을 중지시키고 SH에서 해당 원소를 삭제한다.

[그림 3]은 모니터링을 통해 기법을 전환하는 전체 시스템 구조를 보여주고 있다.

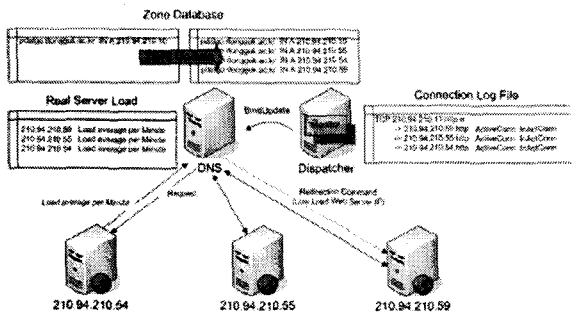

그림 3 경험적 동적 부하 분산 기법의 구조도

4. 실험 및 분석

제안한 경험적 동적 부하 분산 기법에 대한 평가를 위해 디스패쳐만 사용한 경우와 DNS만 사용한 경우 그리고 디스패쳐와 DNS을 혼합한 경우와 시스템의 상태에 따라 디스패쳐와 혼합한 형태를 변형하는 경우에 대한 실험을 통하여 제안한 경험적 동적 부하 분산 기법이 클라이언트의 요청이 폭주되는 시점에서 안정된 서비스를 제공함을 보인다.

실험을 위한 클라이언트의 HTTP 요청은 (주)뱅크 타운에서 개발한 LoadCube를 사용하여 생성한다. LoadCube는 웹 시스템의 부하에 대한 반응을 검사하기 위해 개발된 어플리케이션이며 명령어 입력기, 부하 생성기, 분석기 및 모니터링 도구로 구성되어 있으며 입력 받은 요청에 대한 응답시간을 측정하고 이를 분석하여 웹 페이지로 사용자에게 보여준다. [7, 8]

각 테스트를 위한 환경에서 1초 동안 약 1000~3000개의 같은 크기의 페이지를 요청하여 클라이언트의 응답시간과 웹 서버들의 부하를 측정한다. 약 2시간동안 700,000에서 1,000,000개의 웹 페이지를 서버에 요청하고 부하들을 측정한다.

그림 4는 클라이언트의 초당 요청 페이지 수에 따른 응답시간의 결과를 보여준다. 초당 2000개 이하인 경우에는 디스패쳐만 사용한 경우의 응답시간이 월등히 빠른 것을 볼 수 있지만 3000개 이상인 경우에는 디스패쳐의 부하가 증가되어 혼합된 기법의 응답시간이 디스패쳐에 비해 빠른 것을 볼 수 있고 두 개의 그래프를 비교하면 혼합 기법의 응답시간들이 비교적 변화 없이 안정되어 있음을 알 수 있다.

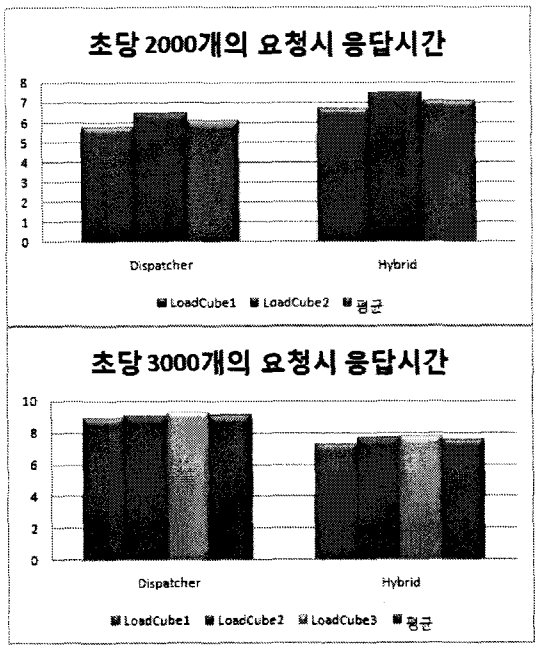

그림 4 Client 요청 수에 따른 응답시간

그림 5는 10분 간격의 전체 웹서버들의 평균 부하와 최대 부하와 최소 부하의 차이를 비교하는 결과이다. 이를 살펴보면 3000개의 요청이 있는 경우 혼합 기법을 사용하면 전체 부하의 평균과 최대 부하와 최소 부하의 차이가 디스패쳐만 사용하는 경우 보다 적음을 볼 수 있다.

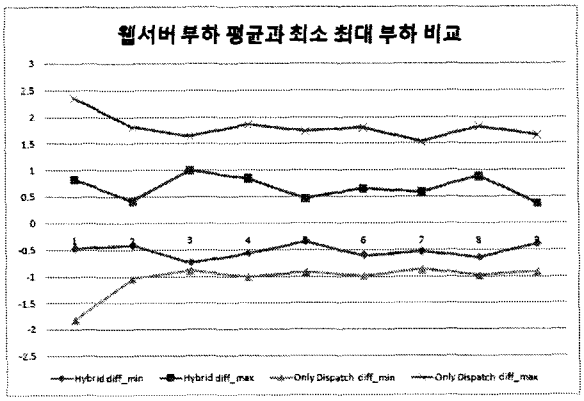

그림 5 웹서버 부하 비교

위의 실험결과를 통해 부하가 적은 상태에서는 디스패쳐를 이용한 부하 분산 기법을 적용하여 부하가 증대되는 시점부터는 혼합 기법을 사용함으로써 디스패쳐에 부과되는 부하를 조절하고 그리고 클라이언트의 최소한의 응

답시간을 가짐으로써 최악의 경우 서비스 자체를 제공받지 못하는 경우를 막을 수 있음을 알 수 있다.

5. 결론 및 향후 연구

다양한 부하의 형태를 보이는 웹 클러스터 환경에서 부하가 집중되는 시점에 발생되는 서비스를 제공하지 못하는 문제를 제거하기 위해 경험적이고 혼합된 부하 분산 기법을 제시하였다. 실험을 통해 제시된 부하 분산 기법이 높은 부하 상태를 보이는 시점에서 최소한의 서비스를 사용자에게 제공하여 최악의 경우를 막을 수 있음을 보였다.
향후 연구로는 경험적 정보를 취하기 위해 발생하는 부하를 최소화할 수 있는 기법으로 확장이 필요하고 확장된 웹 클러스터 환경에서의 실험을 추가함으로써 경험적 혼합 기법의 신뢰성을 높이고자 한다.

[참고 문헌]

[1] V. Cardellini, E. Casalicchio, M. Colajanni, "The State of the Art in Locally Distributed Web-server Systems", ACM Computing Surverys, Vol.32, No.2, 2002
[2] V. Cardellini, M. Colajanni, P. S. Yu, "Dynamic Load Balancing On Web-Server Systems", Internet Computing, IEEE Volume 3, Issue 3, May-June 1999 Page(s):28 - 39
[3] W. Zhang, "Linux Virtual Servers for Scalable Network Services", Proceedings of the OTTAWA Linux Symposium, Ottawa,. Canada, 19-22 July 2000. 21
[4] LVS Documentation, http://www.linuxvirtualserver.org/Documents.html
[5] V. Cardellini, M. Colajanni, P.S. Yu, "Request redirection algorithms for distributed Web systems", IEEE Transactions on Parallel and Distributed Systems, Vol.14, No.4, pp. 355-368, April 2003
[6] H. Bryhni, "A comparison of Load Balancing Techniques for Scalable Web Servers", Network, IEEE Volume 14, Issue 4, Page(s):58 - 64, July-Aug. 2000
[7] Y. S. Hong, J. H. No, S. Y. Kim, "DNS-Based Load Balancing in Distributed Web-Server Systems", SEUS 2006/WCCIA 2006, Gyeongju, Korea, 2006.4
[8] Y. S. Hong, J. H. No, "Request Distribution in Distributed Web-Server Systems", WTAS2006, Calgary, Canada, 2006.7