제 18 회 한국정보처리학회 추계학술발표대회 논문집 제 9 권 제 2 호 (2002.11)

효과적인 Adaptive Load Balancing 을 위한 성능 지표 분석

임유진*, 이원규*, 한영태**, 이동훈**, 최은미*
*한동대학교 전산전자공학부
**건국대학교 컴퓨터공학부
e-mail : emchoi@handong.edu

Performance Counter Analysis for Effective Adaptive Load Balancing

Yoojin Lim*, Wonq Lee*, Youngtae Han**, Donghoon Lee**, Eunmi Choi*
*School of Computer Science & Electronic Engineering, Handong Global University
**Dept. of Computer Engineering, Konkuk University

요 약

웹 서비스를 제공하는 분산 서버 시스템에서 각 서버의 부하 상태를 파악하여 처리해야 할 부하를 조절하여 주면 서버의 부하가 균등하게 되어 더 나은 성능을 얻을 수 있게 된다. 서버의 부하 상태는 시스템의 자원에 영향을 미치는 여러 가지 요소에 의하여 분석을 할 수 있다. 본 논문에서는 다양한 스트레스 테스트를 통하여 서버의 자원의 고갈을 나타내는 주요 성능 지표들을 변화 상태를 분석하였다. 고려된 성능지표로는 Available Memory 양, Page Read 수, Processor Utilization, Processor Queue Length, 네트웍으로 전달된 Transmitted Bytes, 연결된 Connection 개수이다. 실제로 이 중 하나의 요소를 적용시켜서 ALBM (Adaptive Load Balancing Mechanism)을 실행을 하였을 때 일반 LVS Round Robin 보다 성능이 좋은 결과를 낳았다.

1. 서론

인터넷의 사용이 점차 넓은 사용자 계층과 다양한 어플리케이션으로 확대되면서 대용량의 부하를 처리해 내는 확장성이 뛰어난 고성능 분산 서버 시스템 아키텍처가 주목을 받고있다[1,2]. 클러스터형의 분산 서버 시스템[4]은 확장성이 뛰어나고 장애에 대한 안정성도 높으면서 저가라는 점에서 Mission-Critical 한 인터넷 비즈니스 서비스용 서버로 적당하다. 하지만 공유 메모리가 없는 독립적인 이기종의 분산 서버들로 구성되어 있어서 분산 어플리케이션의 개발이 힘들고 분산된 자원의 효과적인 운영이 시스템의 성능에 큰 영향을 미친다.

고성능 분산 시스템의 성능에 큰 영향을 미치는 자원 관리 이슈의 하나가 서버 부하 균등화(Load Balancing)이다. 분산 시스템에서 부하 균등화에 대한 연구는 다각도에서 이루어져왔으며 그 종류도 다양하다[3,4]. 분산 서버들간의 부하 균등에 대한 시작을 어떤 서버 시점에서 하느냐에 따라 Sender-initiated, Receiver-initiated, 두개를 절충한 Hybrid 방법, 그리고 부하균등을 한번만 할지 아니면 부하 분산의 상황을 고려하며 재분산을 시킬 것인지에 따라 정적 부하균등 또는 동적 부하균등으로 나뉘다. 또한 부하균등에 관련된 정책들, 즉 Transfer Policy, Selection Policy, Location Policy 등이 서버의 현 상태를 반영하는가에 따라 Adaptive 부하 균등 방법이 있다. 하지만 이러한 연구들은 고전적 분산 시스템 모델을 전제로 하고 있거나, 고성능 병렬처리에 적합한 방법들이어서 Web Farm 이나 Web Cluster 등 웹 서버용 분산 시스템에는 적당하지 못하다. 분산 웹 서버용 부하균등에 대하여는 주로 L4 Switch 가 지원하는 서버 부하균등이 대부분이다. 이러한 Switch 단에서 부하 균등을 하는 제품군으로는 Alteon 의 ACEdirector, CISCO 사의 Local Director, Foundry 사의 ServerIron, F5 사의 Big-IP 등이 있다[5]. 하지만 이 L4 용 서버 부하 균등 방법은 단순한 웹 문서의 처리에는 적당하지만 현재와 같이 웹을 이용한 복잡하고 다양한 처리나 멀티미디어를 요구하는 경우에는 적당치 못하다.

본 연구는 기존의 L4 Switch 에서 사용하는 방법과는 달리 서버의 부하상태를 이용하여 트래픽을 조절

하는 방법인 Adaptive Load Balancing Mechanism (ALBM)을 다루고 있다. 다양한 종류의 부하를 요구하는 클라이언트의 요청을 사전 지식 없이 여러 이기종의 서버에 균등하게 배분하는 최적알고리즘은 NP Complete 문제이다. 동적으로 변하는 클라이언트의 Request 의 불예측 상황과 알고리즘의 복잡성을 고려해볼 때, 현실적으론 어느 정도의 서버간의 비균등을 허용하다가 과부하가 된 서버에게 트래픽을 감소시켜서 부하균등으로 유도하는 휴리스틱한 방법이 가능하다. ALBM 방법에서는 각 서버에 Agent 들이 돌면서 그 서버의 부하 상태를 판단하여 들어오는 트래픽을 분산시키는 Load Distributor 에게 알려준다. Load Distributor 는 이 정보를 이용하여 자신의 스케줄링 알고리즘에 따라 트래픽을 분산된다. 따라서, 이러한 ALBM 방법을 구현하기 위하여는 각 서버에서 부하를 측정하여 과부하 상태인지를 결정하는 것이 중요하다.

본 논문에서는 다양한 스트레스 테스트를 통하여 서버의 부하 상태를 나타내는 성능 지표(Performance Counter) 들의 특성을 분석한다. 고려된 성능지표로는 Available Memory 양, Page Read 수, Processor Utilization, Processor Queue Length, 네트웍으로 전달된 Transmitted Bytes, 연결된 Connection 개수이다. 이 성능 지표들의 분석 결과를 토대로 RR 보다 나은 성능을 보이는 간단한 ALBM 방법을 제시한다. 좀 더 자세한 실험과 더 나은 ALBM 방법의 개발이 필요하겠지만, 본 논문의 초기 실험 결과는 앞으로 분산 서버의 성능을 향상시키는 ALBM 방법을 개발하는 것이 가능함을 시사하고 있다.

2 절에서는 본 연구에서 고려한 성능 지표들의 특성에 대하여 설명하고 그 다음 절에서는 이런 성능 지표에 대한 다양한 실험 결과를 해석한다. 4 절에서는 이 실험결과를 토대로 만든 ALBM 의 성능을 LVS 의 RR 와 비교한다. 그리고 마지막 절에서 결론을 내린다.

2. Load Balancing에 영향을 미치는 성능 지표들

서버의 성능을 반영하는 많은 요소들 중에 본 논문에서는 Load Balancing 에 영향을 주는 성능 지표들 중 몇 개를 선택하여 연구 분석하였다. 이들 성능 지표들은 시스템을 구성하는 하드웨어(e.g. Memory, Processor, 그리고 Network)의 성능을 나타내는 것들과 여러 클라이언트의 요청을 동시에 처리함으로 인해 생겨나는 성능치(e.g. 연결된 Connections 수)의 두 종류로 나누어 볼 수 있다. 고려된 이들 성능 지표들을 나열해 보면 다음과 같다.

- Memory: Available Memory (KBytes), Number of Page Read
- Processor: Processor Utilization (%), Processor Queue Length
- Network: Total Transmitted Bytes (Bytes)
- Number of Connections: Number of Connections Established

이 성능 지표들에 대한 좀더 자세한 내용이 다음의 표 1 에 나타나 있다[6].

Counter	Indicates
Computername\ Memory \ Available Kbytes	현재 사용 가능한 Physical Memory 의 KBytes 양을 나타낸다. 일정시간 동안의 Average 값이 아닌 측정된 값을 표시한다.
Computername\ Memory \ Page Reads/sec	Page Fault 를 해결하기 위해 디스크가 읽히는 횟수 즉, Hard Page Faults 를 나타낸다.
System\ Processor Queue Length	Waiting Thread 의 수를 나타낸다. Processor 의 Activity 를 측정하는데 유용한 정보이다. Multiprocessor 인 경우는 모든 Processor 들이 공유하고 있는 Single Queue 에서 수행되기를 기다리고 있는 Thread 의 수를 알려준다.
Processor\ % Processor Time (Total instance)	사용한 모든 Processor 의 처리 양을 나타낸다. Processor 가 Idle Thread 를 실행하는 시간의 Percentage 를 100%에서 뺀 값이다.
Network Interface\ Bytes Total/sec: Adapter#	Data Link Layer 에서 Network Adapter 를 Monitoring 함으로 NIC 의 통해 보내고 받은 모든 Bytes 들의 합을 보여낸다. 이 값은 Retransmit 되는 Bytes 와 모든 Frame Header 의 Bytes 를 포함한다.
TCP\ Connections Established	TCP 에 의해서 제공되는 동시 접속의 수의 가장 최근의 값을 나타낸다. ESTABLISHED 와 CLOSE_WAIT 의 상태를 가진 Connection 들을 측정하여 보여준다.

표 1. 고려한 성능 지표에 대한 설명

3. 실험 결과

본 연구의 실험에서는 46 개의 클라이언트 PC 가 생성해 내는 세 가지 종류의 부하에 대한 서버의 성능에 대한 성능지표 값들을 측정하여 결과를 분석해 보았다. 사용된 클라이언트와 서버의 사양은 다음과 같다. 여기서, 서버도 클라이언트와 같은 네트웍 세그먼트에 존재한다.

실제서버	Dual CPU (pⅢ-900Mhz *2) Memory (512MB) OS (windows 2000 advanced server)
클라이언트	CPU (pⅣ-1.4Ghz) Memory (256MB) OS (windows XP)
Network bandwidth	100MB

표 2. 실험 환경에 대한 설명

스트레스를 발생시키기 위한 툴로는 Web Stress Test[7] 툴과 Web Bench[8] 툴이 사용되었다.

1) 실험 1: Fine-Grain Request 테스트

클라이언트들이 한 대의 서버에 내용이 하나도 없는 이름뿐인 짧은 문서를 다량으로 요청하는 테스트이다. Web Bench 를 사용하여 클라이언트로 사용되는 컴퓨터는 1 대부터 46 대까지 늘려가면서 최다의 Request 를 만들도록 하여 한 서버가 처리할 수 있는 용량을 측정하였다.

그림 1 에서 볼 수 있듯이 Processor Queue Length 가 증가하면서 2 정도 도달하면 Processor 의 처리 능력이 다소 떨어지면서 Request 의 양도 한계치에 도달하는 것을 알 수 있다. 이에 따라서 Connection 수도 증가율이

낮아지고 클라이언트 30 개 위치에서는 오히려 줄어들기 시작했다. 다량의 Request 를 처리함으로 인하여 % Processor Time 이 최대의 사용치로 근접해 가는 것을 볼 수 있다. 한편 워크로드 자체가 서버에 특별한 자원을 요구하는 것이 아니고 단지 많은 Connection 을 맺는 것이므로 Available Memory 에는 큰 영향을 주지 않은 것으로 나타났다.

이 실험 결과로 알 수 있는 서버 성능 변화 원리는 크기가 작은 다량의 작은 작업을 요청할 때는 Processor 에 부하를 주게 되며 Processor Queue 에 밀리는 작업들이 생겨나면서 클라이언트는 서버의 서비스가 늦어지게 되는 것을 인식하게 된다.

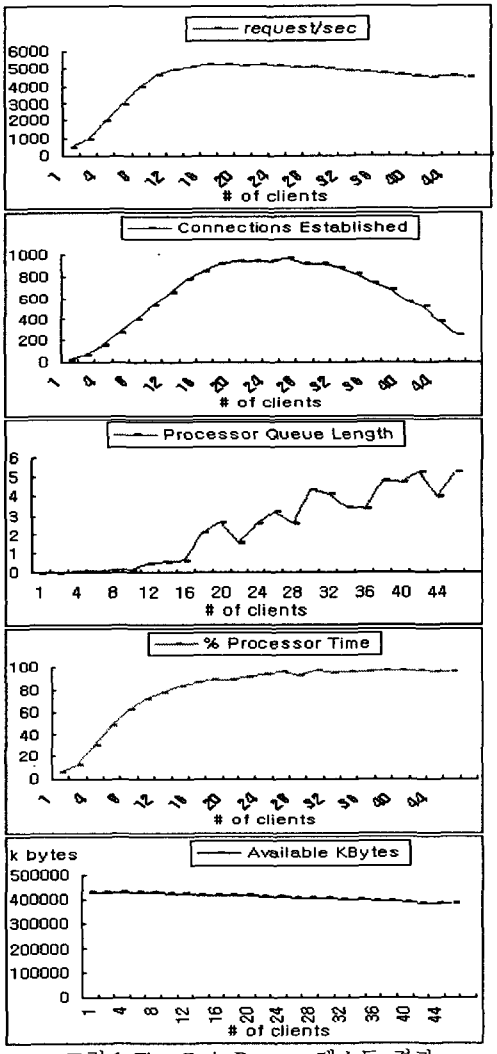

그림 1. Fine-Grain Request 테스트 결과

2) 실험 2: Coarse-Grain Request 테스트

실제 상황과 같이 다양한 크기의 그림 화일을 Random 하게 요청하는 일반적 웹 상황의 테스트를 하였다. Web Bench Tool 을 사용하여 클라이언트로 쓰는 컴퓨터의 개수를 46 대까지 늘려가면서 그에 따른 서버의 Performance Counter 들을 살펴보았다. 다양한 그림 파일 들이 서버의 파일 디렉터리 구조에 Web Tree 로 구성되도록 하였다.

그림 2 에서 알 수 있듯이 크기가 유동적인 일반적인 파일들을 요청하는데 8 개의 클라이언트에서 네트워크 사용의 한계치인 100Mbps NIC Bottleneck 이 생겼다. 그로 인해 프로세서는 더 일할 수 있음에도 불구하고 네트워크 한계치에 걸리면서 % Processor Time 이 더 이상 증가하지 않은 것을 볼 수 있다. Request/Sec 역시 네트워크 한계 때문에 더 이상 증가하지 않는다.

이 실험에서는 클라이언트들이 다양한 종류의 그림을 다운하는 경우 다운하며 생기는 네트워크의 부하가 서버의 성능을 저하시킴을 볼 수 있다.

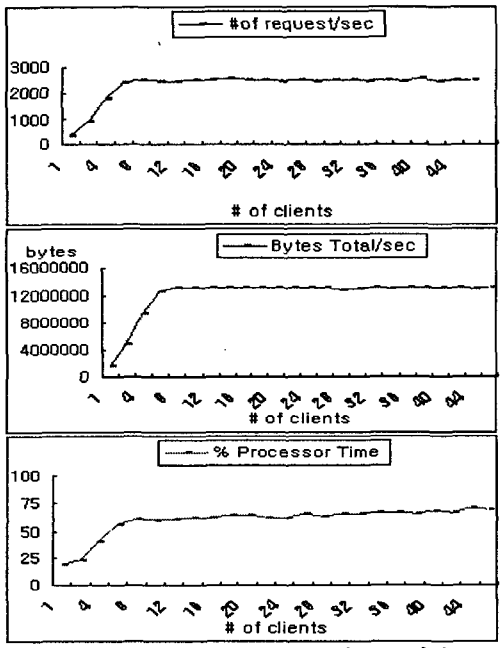

그림 2. Coarse-Grain Request 테스트 결과

3) 실험 3: Memory-Intensive Job Request 테스트

여기서는 클라이언트 개수를 40 개로 고정하고 WAS Tool 을 사용하여 이 클라이언트들이 서버에 다양한 양의 메모리 할당을 요청하도록 하였다. 그림 3 의 X 축 좌표에는 메모리 할당량을 나타내고 Y 축은 그에 따른 서버의 성능 지표를 나타낸다.

그림 3 에서와 같이 서버에 요청하는 메모리가 많아질수록 Available Memory 는 줄어들고 Hard Page Fault 는 점차 늘어 난다. 즉, Page Reads/sec 의 값이 올라간다. 현재 테스트 환경에서는 Available Memory 가 83MB 와 135MB 사이에서 급격히 떨어지고, 또한 그때 Page Reads/sec 이 급격히 증가하므로 이 부분이 바로 Memory 의 한계치 임을 알 수 있다. Request/sec 역시 Available Memory 가 떨어지는 Request Memory Size 27~28MB 인 곳에서 급격하게 떨어졌다. Available Memory 가 부족하게 되면 Hard Page Fault 를 발생하는

상관관계도 이 결과를 통하여 확인할 수 있다.

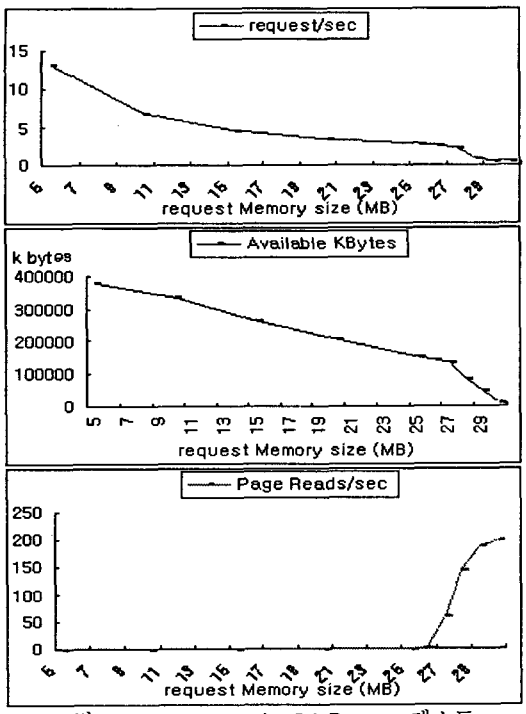

그림 3. Memory-Intensive Job Request 테스트

4. 성능 지표 결과 분석을 이용한 ALBM 실험

앞 절에서 얻은 결과를 Adaptive Load Balancing Mechanism(ALBM)에 적용해 보았다. ALMB 클러스터 시스템은 Linux LVS[9]를 이용하여 실제 서버를 4 대로 구성된 1 개의 클러스터를 만들었다. 부하를 생성하는 클라이언트들은 같은 네트워크 세그먼트 내에 두었으며 그 개수는 36 대까지 늘려가면서 실험을 하였다. 클라이언트가 생성하는 Workload 는 1byte Request 50%, 5Mbytes 를 점유하는 Request 40%, 15Mbytes Request 는 10%의 비율에 따라서 Random 하게 생성하였다. ALBM 이 제대로 부하균등을 수행하기 위하여는 각 서버에 Agent 를 두어 그 서버의 성능을 측정하게 하였으며 그 측정결과를 LVS 를 변형하여 만든 Load Distributor 에게 전달하게 하였다. Load Distributor 는 그 서버 성능 정보를 이용하여 자신의 ALBM 방법에 따라 스케쥴링을 한다.

1) Performance Count 에 대한 Threshold 설정

메모리 부하가 큰 트래픽을 고려하였으므로 측정하는 성능지표로는 Available Memory 를 사용하였고 그 Threshold 를 100MB 로 설정하였다. 이는 앞절의 실험 (다)에서 Available Memory 가 135 와 85MB 사이에서 급격히 떨어졌기 때문에 100MB 를 Threshold 로 잡았다. 서버의 Available Memory 가 Threshold 이하로 떨어지면 서버의 상태가 좋아질 때까지 Round Robin 스케쥴링에서 제외함으로써 과부하가 된 서버가 더 이상의 Request 를 받지 않게 ALBM 방식을 설정하였다..

2) LVS RR 과 ALBM 의 성능 비교

그림 4 는 LVS Round Robin 방식과 ALBM 과의 성능 결과를 비교하여 나타내고 있다. 일반적인 LVS RR 방식으로 부하를 분배했을 때는 다양한 부하를 발생시키는 클라이언트 특성으로 인하여 서버간의 불균형이 생기게 된다. 그러므로, LVS RR 을 쓰는 클러스터 시스템에서는 시스템 전체적인 Request 처리양이 줄어들면서 서버의 상태도 불안정해진다. 한편, 성능 지표 분석을 통하여 적절한 Threshold 값을 찾아내어 적용한 ALBM 에서는 그 결과 그래프가 안정된 선을 유지하고 있다. 이는 Memory 과부화 상태에 있는 서버에게 더 이상의 부하를 주지 않게 함으로서, 클라이언트 쪽에서 요청하는 Request 를 다른 서버로 할당하여 준다. 이는 ALBM 클러스터 시스템이 Workload 를 모든 서버들에서 안정적으로 처리할 수 있음을 알 수 있다.

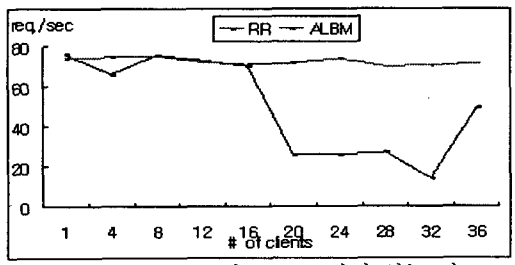

그림 4. LVS RR 과 ALBM 과의 성능 비교

5. 결론

웹 서비스를 제공하는 분산서버시스템의 성능에 영향을 미치는 성능 지표들의 실험을 통하여 병목현상을 일으키는 요소에 대한 연구를 하였다. 이 연구를 통하여 추출된 Threshold 를 적용한 ALBM 를 클러스터 시스템에 이용하였을 때 LVS RR 보다 안정적이고 그에 따라 더 많은 클라이언트의 request 양을 처리할 수 있었다. 또한 서버의 특성에 따라 클라이언트에게 가용성과 최적의 performance 를 내도록 유도할 수 있었다.

참고 문헌

[1] Andrew S. Tanenbaum and Maarten van Steen, " Distributed Systems-Principles and Paradigms," Prentice Hall, 2002

[2] George Coulouris, Jean Dollimore and Tim Kindberg, "Distributed Systems-Concepts and Design," 3rd Edition, Addison Wesley, 2001

[3] Randy Chow and T. Johnson, "Distributed Operating System & Algorithms", Addison Wesley, 1997

[4] Rajkumar Buyya, "High Performance Cluster Computing volumn 1 & 2," Prentice Hall, 1999

[5] Jeffray S. Chase, "Server switching: yesterday and tomorrow," Internet Applications, 2001, Page(s): 114 -123

[6] "Monitoring and Tuning Your Server", Microsoft Technews http://www.microsoft.com/technet/treeview/default.asp?url=/technet/prodtechnol/iis/reskit/iis50rg/iischp5.asp

[7] "Web Application Stress Tool," http://webtool.rte.microsoft.com/

[8] "Web Bench Tool," http://www.etestinglabs.com

[9] "LVS documents" http://www.linuxvirtualserver.org/Documents.html