제 19 회 한국정보처리학회 춘계학술발표대회 논문집 제 10 권 제 1 호 (2003. 5)

부하 특성에 따른 부하 분산 스케줄링 알고리즘들의 성능 평가 및 비교

임유진, 이원규, 최은미
한동대학교 전산전자공학부
e-mail : {inppo, wonq_lee}@hotmail.com, emchoi@handong.edu

Performance Evaluation and Comparison of Workload Scheduling Algorithms based on Workload Characteristics

Yoojin Lim, Wonq Lee, Eunmi Choi
School of Computer Science & Electronic Engineering, Handong Global University

요 약

클러스터 시스템에 요청되는 부하는 서비스 어플리케이션에 따라 다른 성향을 띄게 된다. 부하를 여러 서버에게 분산시켜서 처리를 할 때 부하의 특성을 파악하여 알맞은 부하 분산 스케줄링 알고리즘을 사용하면 보다 효율적이고 좋은 성능을 얻게된다. 이 논문에서는 다른 성향의 부하 형태들과 스케줄링 알고리즘들을 고려하여 각 부하 상황에서 각 스케줄링 알고리즘이 어떤 성능을 보여주는지 비교 분석하였다. 요청되는 자원의 사용량과 사용 시간에 따라서 알고리즘 특성에 영향을 받아 성능 결과가 정하여 짐을 비교해볼 수 있으며, 다양한 부하 상황에도 Adaptive 알고리즘은 서버의 성능 상태에 따라서 스케줄링을 하므로, 일반적인 스케줄링 방식보다 더 나은 결과를 가져왔다.

1. 서론

급속히 늘어나는 인터넷 사용자수와 비지니스 업종에 따라서 다양해지는 서비스 어플리케이션들로 인하여, 서비스를 제공하는 서버들에게 일어나는 부하(Workload)는 커지고 그 특성도 다양해지고 있다. 부하의 과중은, 같은 서비스를 제공하는 여러 서버들을 하나의 시스템으로 사용하는 클러스터 시스템을 통하여 분산 시킬 수 있게 되었다. 클러스터 시스템은 여러 개의 독립적인 서버들을 묶어 확장성, 고가용성, 경제성, 신뢰성을 높이는 하나의 고성능 시스템으로 만든다. 또한 투명성을 가지고 있어 내부에서 추가로 서버를 붙이거나 제거해도 외부에서는 인식하지 못한 채 여전히 하나의 시스템으로 동작하게 된다. [1]

클러스터 시스템은 클러스터로 들어오는 요청들을 특정 스케줄링 알고리즘에 따라 여러 서버들에게 전달한다. 클러스터 시스템에서 사용되는 부하분산 스케줄링 알고리즘에 따라서 시스템의 성능은 크게 달라진다. 일반적으로 알려진 부하 분산 스케줄링 알고리즘들은, 들어오는 요청을 각 서버들에게 순차적으로 전달하는 Round Robin(RR), 요청을 받들 때에 각 서버의 연결 수를 세어서 순간 접속수가 가장 적은 서버에게 요청을 전달하는 Least Connection(LC), 그리고 서버의 성능에 따라서 Weight 를 사용하는 Weighted RR, Weighted LC 의 방법들이 있다. 이 외에도 과부하 상태가 된 서버가 생기면 그 서버에게 과부하 상태를 벗어날 때까지 부하를 더 이상 주지 않는 Adaptive Algorithm[2]을 적용한 스케줄링 방식들로 Adaptive RR(ARR)와 Adaptive LC(ALC)들이 있다.

클러스터 시스템이 제공하는 서비스의 종류와 사용자 그룹의 성격에 따라서 요청되는 부하들의 성향이 달라지게 된다. 클러스터가 제공하는 서비스 어플리케이션이 어떤 자원에 종속되는가에 따라서 CPU, Network, Memory 등 특정 자원을 주로 사용하거나[6] 여러 자원을 복합적으로 사용하는 것일 수 있고, 자원을 사용하는 양과 사용 시간의 길이가 다르게 나타날 수도 있다.[7] 본 논문에서는 이러한 부하 성향의 차이가 있을 때, 클러스터 시스템에서 사용하는 부하 분산 알고리즘들을 어떠한 것으로 사용하는가에 따라서 달라지는 시스템의 성능을 비교 분석하도록 하겠다.

2. 요청되는 부하 성향에 따른 부하의 종류

실험환경에서 부하에 변화를 줄 수 있는 변수로는 작업 크기와 사용 시간이 있다. 이 두 가지 변수에 변화를 주어 다음과 같이 6 가지의 다른 특성을 가진 부하들로 분류 할 수 있다.

(1) **Fixed Size Fixed Duration (FSFD):** 클라이언트에 의해 요청되는 각 작업의 양과 수행 시간이 일정한 경우이고 정적 웹 페이지의 요청하는 경우를 예로 들 수 있다.

(2) **Fixed Size Various Duration (FSVD):** 클라이언트에 의해 요청되는 각 작업의 양은 일정하지만, 서버 측에서 수행되는 시간이 변하는 경우이다.

(3) **Various Size Fixed Duration (VSFD):** 클라이언트에 의해 요청되는 각 작업의 양이 다르고, 서버 측에서 수행되는 시간은 일정한 경우이다. 수행되는 시간이 충분히 길게 하여 작업의 크기에 영향을 받지 않는 경우이다.

(4) **Various Size Various Duration (VSVD):** 클라이언트에 의해 요청되는 각 작업의 양과 수행 시간이 변하는 경우이고, 동영상 서비스를 예로 들 수 있다.

(5) **Large size Long duration & Small size Short duration (LLSS):** 작업 양이 많고 오래 수행되는 작업과 작업 양이 적고 수행 시간이 짧은 작업을 섞은 부하이다.

(6) **Large size Short duration & Small size Long duration (LSSL):** 작업 양이 많지만 수행시간이 짧은 작업과 작업 양이 적지만 수행시간이 긴 작업을 섞은 부하이다.

위와 같은 성향에 따른 자원 사용량을 가시적으로 측정하기 위하여 메모리를 주된 부하로 정하여 실험을 하였다. 메모리는 컴퓨터의 성능에 직접적으로 영향을 줄 수 있으며 또한 부하를 요청할 때 사용하는 메모리의 크기나 사용시간을 비교적 정확하게 조절할 수 있기 때문에 가상의 부하를 주기에 용이하다.

3. 시스템 및 실험 환경 설정

실험에 사용한 클러스터 시스템의 사양과 클라이언트 컴퓨터들의 사양 및 환경은 표 1 에 나타나있으며, 이 논문에서는 동종의(Homogeneous) 서버들을 대상으로 테스트를 하였으므로 Weight 를 사용한 방식은 배제하였다. Adaptive Algorithm 을 적용시키기 위해 ALBM 클러스터 시스템을 사용하였다.[2] 부하를 만들어내기 위해서 WebBench 스트레스 툴을[4], 각 서버들의 상태를 기록하기 위해 Perfmon[5]을 사용하였다.

Real Server	Dual CPU (pⅢ-900MHz *2) Memory (512MB) OS (windows 2000 advanced server)
Client	CPU (pⅣ-1.4GHz) Memory (256MB) OS (windows XP)
Load Distributor	CPU (pⅣ-1GHz) Memory (512MB) OS (Linux Red hat 7.1) Direct Routing 방식[3]
Network bandwidth	100MB
Test tool	WebBench, PerfMon

표 1. 실험 환경

실험에서 성능 비교치는 시간당 처리하는 Request 수로 하였으며, 의미있는 성능치를 비교하기 위하여 WebBench 의 한계치까지의 requests/sec 값을 고려하였다. 이는 서버의 응답시간이 길어지면 WebBench 가 발생시키는 Request 의 수를 줄이므로, 그 이전까지 제공되는 부하가 서버에게 무리를 줄 수 있는 상태이기 때문이다. 서버에 부하를 주는 요청 파일을 모두 ASP 형식으로 하였다. 비록 IIS 에서 순간적으로 ASP 파일을 처리할 수 있는 수는 제한되고 HTML 파일을 사용했을 때에 비해 requests/sec 값이 낮아지지만, 클러스터 시스템의 성능뿐만 아니라 서버들이 각각 어느 정도의 성능을 내는지 알 수 있다. 즉, 클러스터를 구성하는 서버들간에 부하의 특성으로 인한 불균형이 일어났을 때 각 서버의 분석을 통해 원인을 좀 더 쉽게 접근하는 것이 가능해진다. 이것은 서버들이 Windows 의 Active Server Page 라는 카운터 집합을 기록하게 해줌으로써 가능하다. ASP 카운터 집합은 웹 서비스를 해주는 IIS 에서 들어온 요청 중에 ASP 에 관련되어 있는 값들을 기록하는 것이다. 주로 사용한 것은 ASP 요청들의 request/sec 값이다.[5]

실험에서 정해야 할 요소로, Adaptive Algorithm 을 적용한 ARR, ALC 에서는 Threshold 값과 부하의 정도를 조절하기 위한 WebBench 의 변수들이 있다. 먼저, ARR 과 ALC 의 Threshold 값은 서버의 과부하 여부를 판단하는 구간의 경계역할을 한다. 그리고 WebBench 에서 설정해주어야 하는 변수로는 각 클라이언트 당 실행 시킬 Engine 의 수, 각 Engine 당 Thread 의 수, 각 Thread 당 요청하는 Connection 의 수, 그리고 각 Connection 이 한번 요청을 하고 나서 얼마 후에 다시 요청을 할 것인가를 결정하는 Think Time 이 있다. 실험에서는 Engine 의 수를 2 개로 고정하였고, 나머지 변수들은 각 부하 상황마다 다르게 설정하였다. 다음 절에서 결과와 함께 실제로 실험할 때의 설정치를 명시하도록 하겠다.

4. 실험 결과

실험에서는 메모리를 이용한 부하를 2 절에서 분류한 워크로드 종류별로 만들어 각 상황에서 4 가지 부하 분산 스케줄링 알고리즘의 성능을 비교 분석하였다.

- **Fixed Size Fixed Duration**

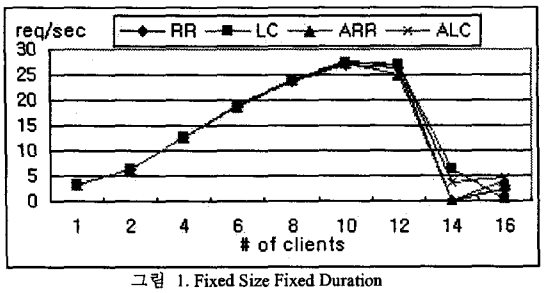

그림 1. Fixed Size Fixed Duration

메모리 크기는 10MByte, 수행 시간은 0 Sec 으로 고정시켰다. 4 Threads, 4 Connection per Thread 로 요청 수를 설정하였고 Think Time 은 10 Sec 으로 하였다. ARR 과 ALC 의 Threshold 는 Available Memory (110MByte, 120M Byte)로 하였다.

고정된 성향의 부하가 만들어졌기 때문에 서버들의 부하 크기와 커넥션 수가 같아 서버간에 불균형이 생기지 않는다. 이 때문에 클라이언트 수를 늘려가면서 부하를 증가시켰을 때 서버들에게 여러 방식으로 부하 분산을 해도 비슷한 성능을 보인다. 같은 이유로 Adaptive Algorithm 을 적용한 ARR, ALC 가 RR, LC 와 성능이 비슷하였다.

● **Fixed Size Variable Duration**

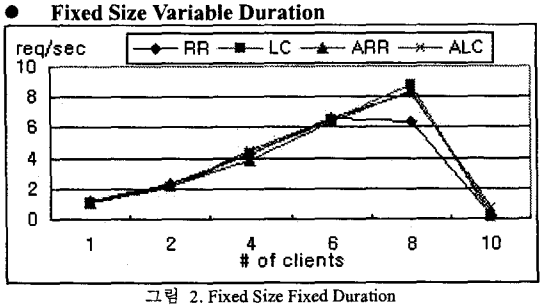

그림 2. Fixed Size Fixed Duration

메모리 크기는 10MByte 로 고정시키고 사용 시간은 0 Sec, 10 Sec, 20 Sec 의 3 종류로 부하의 성향을 다르게 하였다. 4 Threads, 4 Connection per Thread 로 요청 수를 설정하였고 Think Time 은 10 Sec 으로 하였다. ARR 과 ALC 의 Threshold 는 Available Memory (110MByte, 120M Byte)로 하였다.

크기는 같지만 사용 시간이 다른 부하가 만들어짐으로써 서버들간의 순간 접속수 차이가 나게 된다. 요청이 들어온 순간에 접속수가 많으면 부하가 많은 것인데 RR 같은 경우는 서버들에게 순서대로 부하를 전달하기 때문에 접속수가 많아지는 서버가 생기게 된다. 즉, 부하가 많은 그 서버에서 병목현상이 일어나 전체 성능이 떨어지게 된다. 반면에 나머지 세 방식은 서버 자원의 불균형이나 커넥션 개수를 고려하기 때문에 RR 에 비해 좋은 성능을 보였다.

● **Variable Size Variable Duration**

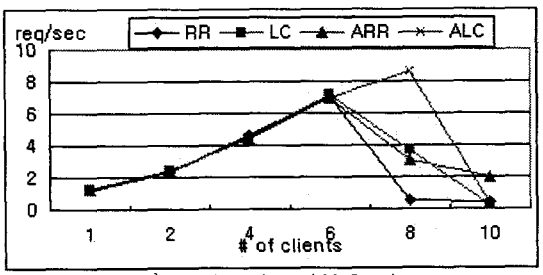

그림 3. Variable Size Variable Duration

메모리 크기는 5MByte 에서 15MByte, 시간은 0 Sec 에서 20 Sec 까지 Uniform Distribution 으로 요청되게 하였다. 4 Threads, 4 Connection per Thread 로 요청 수를 설정하였고 Think Time 은 10 Sec 으로 하였다. ARR 과 ALC 의 Threshold 는 Available Memory (110MByte, 120M Byte)로 하였다.

부하 특성에 따라 크기와 사용 시간 모두 가변적인 부하가 만들어졌다. 그로 인해 자원의 불균형, 그리고 접속수의 불균형이 서버들 간에 심하게 일어났다. VSVD 부하 성향에서는 이 두 가지 가변적인 요소를 모두 고려해서 서버들에게 요청을 나눠주는 ALC 의 성능이 나머지 세 방식보다 우수하게 나왔다.

● **Variable Size Fixed Duration**

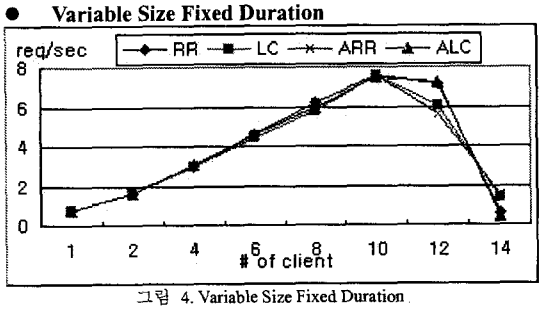

그림 4. Variable Size Fixed Duration

메모리 크기는 5MByte 에서 15MByte 로 Uniform Distribution 으로, 사용 시간은 10 Sec 초로 고정시켰다. 4 Threads, 2 Connection per Thread 로 요청 수를 설정하였고 Think Time 은 10 Sec 으로 하였다. ARR 과 ALC 의 Threshold 는 Available Memory (50MByte, 60M Byte)로 하였다.

실제로 다른 실험의 결과에서 크기가 다른 부하들이 짧은 시간으로 고정되어 있을 때 부하의 양이 커지면 응답 시간이 늘어나면서 사용 시간도 늘어나게 된다. 즉, 사용시간이 고정되는 것이 아니라 가변적으로 변한다. 그래서 시간을 고정시킬 때 수행시간에 영향을 받지 않는 충분히 큰 시간을 사용하였다.

이 경우 모든 스케줄링 알고리즘이 FSFD 의 경우와 마찬가지로 성능이 비슷하다. 사용시간을 고정시켰기 때문에 접속수의 차이가 없고, 작업크기가 가변적이지만 Uniform Distribution 의 특성 때문에 확률적으로 요청수가 많아질수록 서버에 분산되는 부하

가 같아진 것으로 보인다.

그림 5, 6 은 LLSS 와 LSSL 의 실험 결과이다. 두 개의 FSFD 를 섞어놓은 부하상황에서는 어떠한 스케줄링 알고리즘이 효과적인지 비교하기 위하여 실험을 하였다. 두 개의 실험모두 FSFD 부하 상황과 마찬가지로 약간의 차이가 있지만 모든 스케줄링 알고리즘이 비슷한 성능을 보여주고 있다.

● **Large Size Long Duration & Small Size Short Duration**

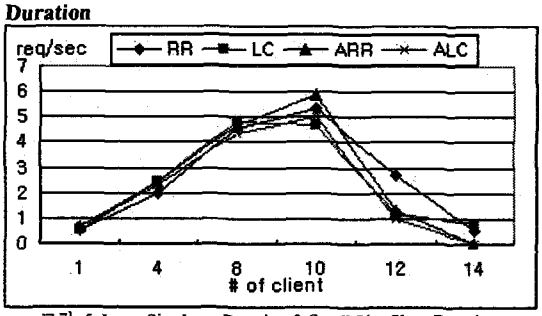

그림 5. Large Size Long Duration & Small Size Short Duration

메모리 크기가 15MByte, 사용 시간은 20 Sec 으로 고정시킨 부하와 5MByte, 0 Sec 으로 고정시킨 부하를 각각 50%씩 요청하도록 하였다. 4 Threads, 2 Connection per Thread 로 요청 수를 설정하였고 Think Time 은 10 Sec 으로 하였다. ARR 과 ALC 의 Threshold 는 Available Memory (110MByte, 120M Byte) 로 하였다.

전체적인 그래프 유형을 보면 비슷한 성능을 보여주고 있다. 이는 FSFD 성향의 부하를 섞은 LLSS 의 성향 때문이라고 보여진다. 하지만 10 클라이언트에서 보면 RR 이 LC 보다 조금 더 나은 성능을 보여준다. ARR 과 ALC 가 각각 RR, LC 보다 조금이지만 나은 성능을 보여준 것으로 보아 어느 정도 서버간의 불균형을 일으키는 성향을 가졌음을 알 수 있다.

● **Large Size Short Duration & Small Size Long Duration**

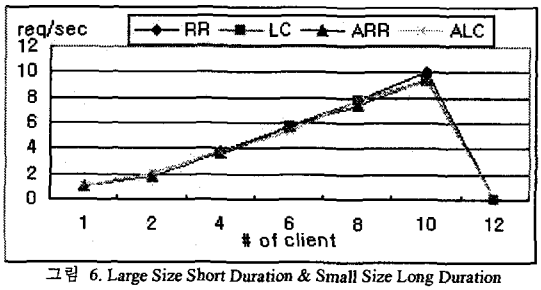

그림 6. Large Size Short Duration & Small Size Long Duration

메모리 크기가 15MByte, 사용 시간은 0 Sec 으로 고정시킨 부하와 5MByte, 20 Sec 으로 고정시킨 부하를 각각 50%씩 요청하도록 하였다. 4 Threads, 4 Connection per Thread 로 요청 수를 설정하였고 Think Time 은 10 Sec 으로 하였다. ARR 과 ALC 의 Threshold 는 Available Memory (110MByte, 120M Byte) 로 하였다.

LSSL 역시 부하 특성상 FSFD 와 마찬가지로 모든 스케줄링 알고리즘이 거의 같은 성능을 보여준다. 작업시간이 큰 부하는 작업량이 적고, 작업시간이 적은 부하는 작업량이 많아서 불균형이 서로 상쇄되는 영향도 있는 것으로 보인다.

서버들간의 불균형이 일어나지 않는 일정한 형태의 FSFD 는 모든 스케줄링 알고리즘이 비슷한 성능을 보였다. 서버들간의 접속수, 자원 불균형이 일어나는 FSVD, VSVD 와 같은 부하 상황에서는 부하의 특성에 따라 클러스터 시스템의 효과적인 성능을 보여주는 특정 스케줄링 알고리즘이 있음을 볼 수 있다. FSVD 는 사용시간의 차이가 순간 접속수의 불균형을 일으키므로 RR 을 제외하고 LC, ARR, ALC 는 좋은 성능을 보였다. VSVD 는 서버간에 자원, 접속수의 불균형을 심하게 하므로 두 불균형을 모두 고려하는 ALC 가 나머지 알고리즘들에 비해 효과적인 결과를 보였다.

5. 결론

이 논문에서는 클러스터 시스템의 중요한 이슈인 부하 분산에서 부하의 특성과 스케줄링 알고리즘의 관계를 고려하였다. 실험에서 6 가지의 부하상황 하에서 4 가지의 알고리즘의 성능을 비교한 결과, 각 부하 상황에 따라 각 알고리즘의 성능이 다르게 나타났다. 다양한 부하 상황에도 Adaptive 알고리즘은 서버의 성능 상태에 따라서 스케줄링을 하므로, 일반 RR 과 LC 보다 더 나은 결과를 가져왔다. 부하의 성향이 복잡해지는 어플리케이션 서비스 제공 시 Adaptive 알고리즘은 전체 클러스터 시스템의 성능에 보다 효과적이다.

참고문헌

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

[2] 임유진, 이원규, 한영태, 이동훈, 최은미, "Linux LVS 와 ALBM 클러스터 시스템의 성능 평가 및 비교 분석", 2002 년도 제 18 회 한국정보처리학회 추계학술발표대회 논문집 제 9 권 제 2 호 p183-186

[3] "LVS Documentations" http://www.linuxvirtualserver.org/VS-DRouting.html

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

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

[6] Xiao, Chen, Zhang, "Dynamic Cluster Resource Allocations for Jobs with Known and Unknown Memory Demands", IEEE Transactions on Parallel and Distributed Systems, Vol 13, No 3, March 2002, p223-240

[7] Arlitt, Williamson, "Internet Web Servers: Workload Characterization and Performance Implications", IEEE/ACM Transactions on Networking, Vol.5, No 5, October 1997, p631-645