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

Linux LVS 와 ALBM 클러스터 시스템의 성능 평가 및 비교 분석

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

Performance Evaluation and Comparison of Linux LVS and ALBM Cluster Systems

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

요 약

클러스터는 부하 분산 스케줄링의 방법에 따라서 그 시스템의 성능이 달라진다. 본 논문에서는 클러스터의 한 종류인 LVS 에서 제시되었던 RR, WRR, LC, WLC 의 부하 분산 스케줄링을 이용하여 시스템의 각종 부하를 처리하는 방법과 클러스터의 각 서버의 성능을 측정하여 부하 분산을 결정하는 ALBM(Adaptive Load Balancing Mechanism) 방식을 비교하였다. 이 성능 평가를 하기 위하여 여러 실험 환경 설정을 하여 다양한 실험을 통하여 비교 분석 하였다. 실험을 통하여 ALMB 이 기존의 여러 스케줄링 방식에 비하여 전체적으로 좋은 성능 결과를 낳았으며 클러스터 서버간의 안정적인 부하 분산이 되는 결과를 얻었다.

1. 서론

인터넷의 급속한 확대로 서비스를 제공하는 서버들은 증가하는 사용자 수와 요구에 따르는 부하의 가중을 처리해야 할 뿐만 아니라 서비스를 제공하는 서버의 성능확장을 요구한다. 한대의 서버에서의 네트워크 대역폭이나 자원의 확장에 인한 한계로 인하여, 같은 서비스를 제공하는 여러 서버들을 한 구조의 형태로 묶는 클러스터링(Clustering)이 대처방법으로 부각 되었다.[1]

클러스터링은 독립적인 컴퓨터들을 연결하여 확장성(Scalability), 고가용성(High Availability), 경제성(Cost-Effectiveness), 이상 상황(Failure)으로부터 신뢰성(Fault Tolerance)을 가진 하나의 고성능 시스템으로 만드는 것이다. 이렇게 한데 묶여진 여러 컴퓨터들을 구성하는 클러스터는 서로 통신과 작업을 미들웨어 단에서 함으로서 주어진 다량의 요구 사항들을 나눠서 처리하게 된다. 클러스터의 또 다른 특성은 서버를 투명하게(Transparency) 추가 및 삭제하여 확장성을 높일 수 있다. 서버 노드나 서비스 데몬에 문제가 생기면 사용자가 인식하지 못하게 서버 시스템을 재구성하므로 높은 가용성을 얻을 수 있다.

리눅스 가상 서버(Linux Virtual Server: LVS)는[2] 클러스터링의 한 종류이다. 실제 서버를 클러스터로 구성하여 높은 확장성과 가용성을 만족시키는 리눅스의 서비스이다. 클라이언트 쪽에서 보면 클러스터는 SSI(Single System Image)를 만족하기 위하여 하나의 가상 서버를 제공하고 있으며, 실제 서버의 앞단에 있어서 부하분산(Load Balancing)을 하는 서버이다. 부하분산서버는 들어오는 클라이언트의 요청들을 스케줄링을 통하여 여러 서버에 분산시킨다.

LVS 는 서버 부하 분산을 위하여 RR, WRR, LC, WLC 등 여러 가지 부하 분산 알고리즘을 제공한다. 하지만 이 방법들은 각 요구가 어느 정도 균등한 부하를 가지고 있는지 또는 네트워크 연결 속도로 서버의 성능을 파악하는 등 실제 서버의 성능으로 부하를 판단하기 보다는 외부에서 피상적인 방식으

로 부하를 판단하고 있다. 따라서, 어플리케이션 단계에서 할 작업들이 점점 많아지고 있는 요즘의 Mission Critical 한 엔터프라이즈 서비스에는 적당하지 못하다.

본 논문에서는 LVS 를 변형하여 실제 각 서버의 부하의 상태를 반영하여 부하 분산을 하는 Adaptive Load Balancing Mechanism (ALBM) 및 이를 위한 ALBM 아키텍쳐를 제시한다. 이 ALBM 방법은 각 서버의 다양한 자원의 상태 정보를 각 서버에서 돌고 있는 Node Agent 가 모아 Load Distributor 로 전달한다. Load Distributor 는 이 정보들을 사용하여 다양 스케줄링 알고리즘을 제공한다. 본 논문에서는 또한 ALBM 클러스터 시스템의 성능을 기존의 LVS 의 여러 LB 방식의 성능과 비교하였다.

다음 절에서는 Linux LVS 의 구조와 제공되는 부하분산 스케줄링 알고리즘을 설명한다. 3 절에서는 우리가 만든 ALBM 클러스터의 구조와 그 메커니즘을 제시한다. 그 다음절에서는 ALBM 클러스터와 원래 LVS 클러스터의 성능을 다양한 각도에서 비교분석하고 그 실험결과를 설명한다. 마지막 절에서는 본 논문을 결론 내린다.

2. Linux Virtual Server 의 구조

본 연구에서 개발한 ALBM 클러스터는 Linux LVS 클러스터를 변형하여 만든 것이다. 따라서, 본 절에서는 LVS 의 구조와 관련 알고리즘들을 살펴본다. LVS 의 구조[3]는 서버 전체로 이루어진 클러스터의 가상 IP(VIP: Virtual IP)를 가지고 서버들은 각기 다른 내부 IP 들을 가진다. 클라이언트의 Request 는 VIP 를 가지고 있는 LVS 로 전달되며, LVS 는 내부의 스케줄링 알고리즘에 의해 선택된 서버에, 내부의 라우팅 방법에 따라서 변형된 패킷을 전달한다. 라우팅 시 LVS 에 도착한 패킷의 헤더부분이 변경되어 패킷이 서버에게 전송이 된다. 또한 LVS 는 연결되어 있는 여러 서버들 중에 어떤 서버에게 현재 패킷을 전달하여 서비스를 받게 할지 결정하는 여러 가지 부하 분산 스케줄링 방법을 제공하고 있다.

LVS 에서 제공되는 라우팅 방식에는 NAT, IP Tunneling, Direct Routing 의 세가지 방식이 있다[6]. (1)먼저, NAT 방식은 앞 단에 위치한 가상서버(Virtual Server)에게 외부에서 접근할 수 있는 Public IP 주소를 할당하고, 이 가상 서버에게 들어오는 클라이언트의 Request Packet 내의 IP 주소를 실제 서버의 IP 주소로 바꾸어서 부하 분산을 하는 방법이다. (2) 다음으로, IP Tunneling 방식은 가상서버로 들어온 IP Datagram 을 할당하는 실제서버의 IP 와 함께 포장(Encapsulation)시키는 방식('Wrap & Redirect')을 쓴다. Redirect 된 Datagram 은 실제서버로 가서 처리가 된다. (3) 마지막으로, Direct Routing 방식은 가상서버로 들어온 Data Frame 의 MAC Address 를 바꿔서 실제서버 쪽으로 보내주는 것이다. Load Balancer 역할을 하는 가상 서버와 실제 서버들은 물리적으로 같은 Segment 에 있어야 한다. 모든 실제 서버들은 Virtual IP 로 설정된 Loopback Interface 를 가지고 있고, Load Balancer 는 패킷을 받기 위해 Virtual IP 가 설정되어 있어야 한다. 이 방식도 Response 가 실제 서버에서 클라이언트로 직접 가기 때문에 실제서버도 Public IP 를 가지고 있어야 한다. 본 연구에서는 이상의 세가지 라우팅 방식들 중에 가장 성능이 좋은 Direct Routing 방식을 사용하여 비교하였다.

부하 분산 스케줄링 방법으로 LVS 에서는 7 가지 방식을 제공하고 있다. 본 절에서는 4 절의 실험에서 사용된 4 가지 방식에 대하여 간단히 설명한다. (1) 먼저, RR(Round Robin) 방식은 Connection 의 개수나 Response Time 과 관계 없이 순차적으로 네트워크 Connection 을 각 서버들에게 전달 시킨다. Round Robin DNS 는 단일 domain 을 다른 IP Address 로 해석하지만 스케줄링은 host 별로 하고 DNS Caching 때문에 실제 서버들에 동적인 부하 불균형을 가져올 수 있다. (2) 둘째로, WRR(Weighted Round Robin)은 RR 과 비슷하게 순차적으로 스케줄링하나 실제 서버간에 서로 다른 처리 용량을 지정할 수 있다. WRR 을 사용하면 실제 서버에서 Connection 을 순간마다 셀 필요가 없어서 동적 스케줄링 알고리즘보다 과부하가 적으므로 더 많은 실제서버를 붙일 수 있다. 그러나 요청에 대한 부하가 매우 많을 경우 실제 서버들 사이에 동적인 부하 불균형 상태가 생길 수 있다. (3) 다음으로, LC(Least Connection) 방식은 가장 접속이 적은 서버로 요청을 연결하는 것이다. 순간마다 각 서버의 연결 개수를 세는 것이 필요한 동적인 스케줄링 방식이다. 클러스터의 가상 서버가 비슷한 성능의 서버로 구성되고 request 마다 동일한 작업 양을 요구하게 된다면, 아주 큰 요구가 한 서버로만 집중되지 않기 때문에, 접속수가 많은 경우 효과적으로 분산을 할 수 있다. 가장 빠른 서버에서 더 많은 Connection 을 처리할 수 있으므로 다양한 처리 용량을 지닌 서버로 구성했을 경우에도 적절하게 작동 한다는 것을 알 수 있다. 그렇지만 실제로는 TCP 의 TIME_WAIT state 때문에 아주 좋은 성능을 낼 수 없다. 또한, LC 를 이용할 때 다양한 처리용량을 지닌 서버로 구성되었을 경우 부하분산이 효율적으로 되지 못할 수 있다. (4) 마지막으로, WLC(Weighted Least Connection)방법은 LC 방법으로 스케줄링과 각각의 실제서버에 성능 가중치를 부여하는 방식을 같이 할 수 있어서 가중치가 높은 서버에서 더 많은 양의 Connection 을 받을 수 있다.

3. ALBM 클러스터 시스템의 구조

ALBM Cluster System 은 LVS 구조와 같이 앞 단에 Load Distributor 가 있으며 가상 IP 주소를 가진 Load Distributor 는 Direct Routing 방식과 같이 MAC 주소를 바꾸어 실제서버로 클라이언트의 Request 를 넘겨준다. 실제서버에는 클러스터 어플리케이션 서비스를 담당하는 서버가 탑재되어있으며, 서버 부하를 측정하여 분석하는 에이전트가 실행되고 있다. 외부 클라이언트의 요청은 Load Distributor 을 통하여 내부 네트워크를 통하여 과부하가 된 서버를 피

하여 할당이 되고 결과는 외부 네트워크를 통하여 직접 클라이언트에 전달이 된다.

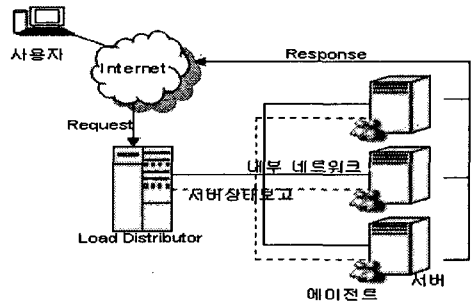

그림 1. ALBM 의 구조

Load Distributor 는 소프트웨어 Layer 4 Switch 의 일종으로 클라이언트로부터 요청이 오면 특정한 스케줄링 Algorithm 이나 Policy 를 통해서 클러스터링된 서버들로 Traffic 을 분배 시킨다. Load Distributor 는 주기적으로 서버의 정보를 에이전트를 통하여 모으게 된다. 이 정보는 Load Balancing 이나 Fault-Tolerance 에 사용된다. 서버에 위치한 에이전트는 주기적으로 서버의 부하 상태를 점검하고 부하 여부를 판단한다. 서버가 과부하가 되어 작업 수행에 지장이 있을 것으로 판단되면 에이전트는 Load Distributor 에게 과부하 서버에게 Request 를 보내지 않도록 상태보고를 전달한다.

4. 실험 결과

본 연구에서는 LVS 클러스터와 ALBM 클러스터의 성능을 비교하기 위하여 두 종류의 실험을 하였다. 하나의 실험은 동종의 서버들로 구성된 클러스터를 사용하였고 다른 하나는 이종의 서버로 구성된 클러스터를 사용하였다. LVS 의 RR 및 LC 스케줄링 방법들과 성능 지표 Available Memory 로 서버들의 부하 분산을 평가하는 ALBM 방식과 결과 비교하였다. Unbalance 상황을 만들기 위하여 Coarse-Grain Request 들의 혼합으로 Workload 를 생성하였다. Stress Testing 툴로는 Web Bench[5]와 WAS[4] 툴을 사용하였다. 성능으로는 Request/Sec 를 측정하였다.

(가) **동종 (Homogeneous) 서버 클러스터 실험**

이 실험에는 Dual CPU 를 갖는 동종의 4 개의 서버로 클러스터를 구성하고 Web Bench Tool 을 사용하여 클라이언트의 수를 32 대까지 순차적으로 늘려감으로써 Request 양을 증가시켰다. Workload 는 1byte Request 를 50%, 5Mbytes Request 를 40%, 15Mbytes Request 를 10% 비율로 Random 하게 생성되게 하였다. 표 1 은 실험환경을 요약한 것이다.

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
Network bandwidth	100MB
Test tool	Web bench, PerfMon

표 1. 동종 서버 클러스터 환경

그림 2 의 그래프를 보면 초반 어느 정도까지의 Request 는 모든 경우에 적절히 처리해줄 수 있음을 알 수 있다. 하지만 클라이언트의 수가 16~20 대가 넘어 서면서 그 차이를 보이기 시작하였다. LVS 의 RR 과 LC 는 초당 발생된 요청의 수가 급속히 줄어든 반면, ALBM 은 어느 정도의 Workload 발생을 유지 하였다. 이것은 Web Bench 툴이 각 서버의 처리 능력에 비례하여 Workload 의 양을 조정하기 때문이다. 즉, 16 대 이상의 클라이언트가 과다한 Workload 를 발생시키자 RR 과 LC 스케줄링의 경우 서버의 성능 저하가 급격히 일어났음을 알 수 있다. 이에 비하여 ALBM 은 과부하된 서버에게는 과부하 상태가 해결될 때까지 추가적인 부하를 할당하지 않으므로 각 서버들에 부하가 균등히 분배되어 다른 두 스케줄링 방식에 비하여 서버의 성능저하가 급격히 일어나지 않고 어느 정도의 성능을 유지할 수 있음을 알 수 있다.

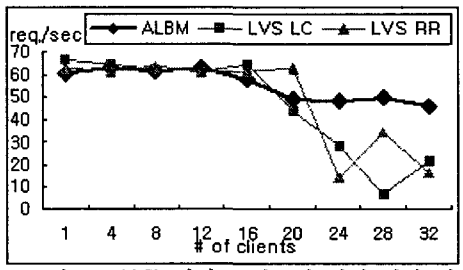

그림 2. 동종 서버 클러스터 실험 결과 비교

(나) **이종 (Heterogeneous) 서버 클러스터 실험**

앞의 실험과는 다르게 클러스터를 다른 종류의 서버들로 구성하였다. 이종 서버들이므로 스케줄링 방식에서 서버별로 Weight 를 할당하여 서버의 성능에 따른 부하의 담당 비율을 정할 수 있도록 하였다. Workload 는 1byte Request 를 50%, 5Mbytes Request 는 30%, 12.5Mbytes Request 는 20%비율로 Random 하게 생성되게 하였다. 이것은 앞의 실험보다 Request 의 사이즈를 조금 줄인 것인데 이는 이종 클러스터 구성 시 서버 2 대를 성능이 떨어지는 컴퓨터로 교체를 했기 때문에 전체적인 Workload 를 다소 줄이기 위해서 이다. 표 2 는 이종 클러스터를 구성하는 두 종류의 서버 사양을 보여주고 있다. Workload 는 앞의 실험과 동일하게 Web Bench 툴을 이용하여 32 대의 클라이언트를 순차적으로

붙여가며 Workload 의 양을 증가시켰다.

Real server	Dual CPU (pⅢ-900MHz *2) Memory (512MB) OS (windows 2000 advanced server) Weight : 2
	CPU (pⅢ-733MHz) Memory (256MB) OS (windows 2000 advanced server) Weight : 1

표 2. 이종 서버 클러스터 환경

그림 3 은 각 스케줄링 방식의 Request/Sec 의 성능을 비교한 그래프들을 보여준다. RR 은 불안정한 성능을 보인데 비해 나머지 스케줄링 알고리즘은 비교적 안정된 성능을 보여 주고 있다. 여기서 WLC 는 이종 서버의 성능 차이를 고려하여 각 서버에 Weight 를 부여하고 LC 를 적용한 경우이다. 여기서 Weight 는 성능이 좋은 서버에게 성능이 나쁜 서버보다 2 배 큰 Weight 를 부여하였다. 이 경우 WLC 가 LC 보다 좀 더 좋은 성능을 보임을 알 수 있다. ALBM 은 이중 가장 좋은 성능을 보여 주었다.

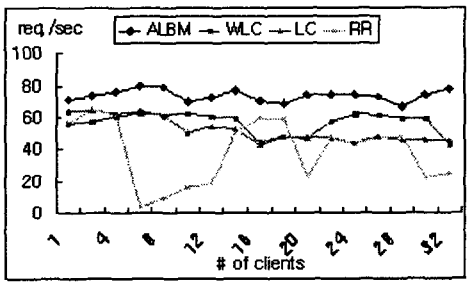

그림 3. 이종 서버 클러스터의 실험 결과 비교

ALBM 이 부하 균등화에 의해 좋은 성능을 나타냄을 확인해 보기 위하여 서버들의 각각의 Available Memory(남은 메모리) 양을 시간대에 따라서 측정하여 보았다. 그림 4 는 RR 경우의 결과이고 그림 5 은 ALBM 경우의 결과다. 그림 4 그래프를 보면 클러스터에 속한 서버들 중 한대가 과로하여 일을 할 수 없음에도 스케줄링 특성상 계속적으로 그 서버에 부하를 나눠주게 된다. 그렇게 과부화된 상태에서 계속적으로 메모리 할당을 요구하는 부하를 받으면 Available Memory 는 시스템을 유지하는데 필요한 양조차 부족하게 된다. 이는 Request 를 처리하는데 매우 부적격한 상태이며 전체적인 성능을 심하게 떨어뜨리고 또 다른 서버가 연쇄적으로 과부하에 걸리게 되는 원인이 된다.

ALBM 의 경우(그림 5) 서버들 사이에 성능 차이가 있더라도 각 서버의 능력을 넘지 않는 범위 내에서 부하를 할당하므로 모든 서버들이 과부하 되지 않고 적절한 메모리를 사용하였다는 것을 볼 수 있다. 과부하로 인한 서버의 성능저하를 줄임으로 가장 좋은 성능을 나타냄을 알 수 있다.

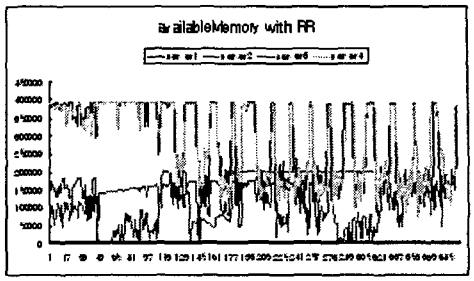

그림 4. RR 경우 서버들의 Available Memory 양

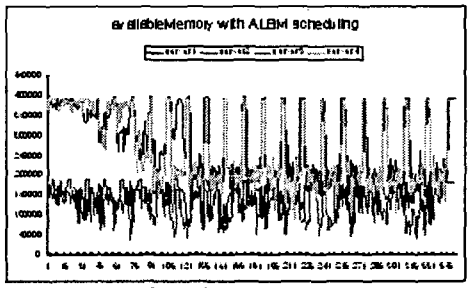

그림 5. ALBM 경우 서버들의 Available Memory 양

5. 결론

본 논문에서는 각 서버들의 성능을 측정하지 않는 LVS RR 방식 또는 간접적인 방법으로 측정하는 LVS LC 방식으로 부하 분산을 하는 기존의 방법과 ALBM 과의 성능 비교를 하였다. ALBM 은 Node Agent 들을 통하여 서버들로부터 직접적으로 성능을 측정하여 부하 균등화 방법을 써서 각 서버의 상태에 따라서 적절히 Request 를 분산할 수 있었다. 본 연구의 실험 결과에 의하여, 동종 서버들로 이루어진 클러스터와 이종 서버들로 이루어진 클러스터 환경 모두에서 다른 스케줄링방식 보다 ALBM 방식이 더 좋은 성능 결과를 얻었으며 클러스터 내의 서버들의 전체적인 안정된 성능을 나타내었다.

참고문헌

[1] Rajkumar Buyya, "High Performance Cluster Computing vol. 1", Prentice Hall, 1999

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

[3] Wensong Zhang, Shiyao Jin, Quanyuan Wu, "Scaling Internet Services by LinuxDirector", The Fourth International Conference on High-Performance Computing in the Asia-Pacific Region-Volume 1, May 14 - 17, 2000

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

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

[6] "TCP/IP transport", http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wcepb40/htm/_pbplatman_tcpip.asp