클러스터 기반 웹 서버에서의 분산 QoS

박 성 우 , 정 규 식 , *김 동 승
숭실대학교 정보통신전자공학부, *고려대학교 전기공학과
knocks@q.soongsil.ac.kr , kchung@q.soongsil.ac.kr , *dkim@classic.korea.ac.kr

Seong-Woo Park , Kyusik Chung , *Dongseung Kim
School of Electronic Engineering, Soongsil University
*Dept. of Electrical Engineering, Korea University
knocks@q.soongsil.ac.kr , kchung@q.soongsil.ac.kr , *dkim@classic.korea.ac.kr

A distributed QoS system for cluster based web server systems

Abstract

This paper introduces a new distributed QoS (Quality of Service) control system for clusters of web servers. The proposed system can employ not only network bandwidth but also other metrics such as processor load, memory usage, and storage access load that affect the overall system performance. Moreover, it controls over clustered workstations in order to utilize idle resources among workstations. This architecture maximizes overall usage of cluster of web servers while it provides predictable and differentiated performance for each contents volume. We implemented a prototype of introduced system, and the test results showed the proposed method can control QoS in a cluster server system.

I. 서론

인터넷 트래픽의 증가와 함께 서비스의 품질에 대한 관심이 증폭되어 이에 따른 연구가 진행중에 있다. QoS(Quality of Service)는 광범위 하게는 서비스의 안전성을 나타내는 말로 사용되나 ISP (Internet service provider) 에서는 네트웍에 대한 서비스 품질 보장을 의미한다. 반면 웹 서비스에 대해서 이러한 서비스 품질 적용을 하기에는 많은 변수가 존재하기 때문에 다양한 각도에서 연구가 진행되고 있다. 웹 서버에서의 QoS는 네트웍 QoS에 대한 차별화 서비스에 대한 연구[1-3], 컨텐츠 기반 차별화 서비스에 대한 연구[4], 자원분배를 이용한 차별화 서비스에 대한 연구[5-7]가 진행중이다.

본 논문에서는 QoS를 하기 위한 모듈을 분산시키고 주문형 컴퓨팅 시스템을 도입함으로써 클러스터 기반의 웹 서버에서 차별화 서비스 및 품질을 보장하기 위한 조건을 제시하고 품질의 단위를 이용하여 조건을 만족시키는 시스템을 제안하였으며 기존 시스템의 문제점을 제기하고 클러스터링 기반 서버에 QoS를 수행할 수 있는 관리자를 모듈화 하여 이식시키는 시스템을 설계하고 실험하였다. Linux 를 바탕으로 하여 클러스터링 서버를 구성하고 여기에 Kernel 모듈를 구현하였다. 실험은 네트웍 대역(Bandwidth)의 관점에서 수행하였으며 능동적으로 성능을 조절하는 것에 대하여 우수한 동작을 보여주었다.

II. 분산 QoS 시스템

2.1 웹 서버에서의 QoS

QoS는 일반적으로 서비스를 안정적으로 하기위한 서비스 공급자와 가입자 사이의 계약에 의해 이루어진다. ISP의 경우는 네트웍의 실제 서비스 되는 가동율과 대역폭이 관심의 대상이 되고 현재도 이를 이용하

여 서비스 되고 있다. 반면 웹 서비스의 경우 대역폭 이외에 CPU사용량, 저장공간, 메모리, IO등 다양한 시스템 자원과 사용자 컨텐츠에 따라 웹 서비스의 중요한 기준인 응답시간이 바뀔 수 있다. 그러므로 접속자의 요구에 의한 서비스의 보장은 ISP의 그것과는 다르다. 하나의 웹 서버에 대하여 서비스를 보장하려고 하는 시도는 Barnes의 연구[8]에서 찾아 볼 수 있으며 이러한 방식을 클러스터링 시스템에 도입하려는 연구는 Li의 연구[7]에서 찾아볼 수 있다. 본 논문에서는 위 연구에서 발전하여 하나의 부하분산기에서 담당하던 QoS를 각 서버로 분산시키고, QoS Metric에 맞는 주문형 컴퓨팅 모델을 도입하였다.

2.2 주문형 컴퓨팅 모델(computing on demand)

주문형 컴퓨팅 모델은 하나의 서비스에 대하여 부하가 증가하였을 경우 다른 서비스의 여유 노드를 추가시켜 부하를 줄일 수 있도록 하는 개념이다. 복수의 서버를 가지고 있는 클러스터링 환경에서 한 서비스의 부하가 증가할 경우 다른 서비스의 부하가 적으면 시스템의 설정을 변경시켜 새로운 클러스터링을 구성함으로 서비스에 대하여 동적인 부하 관리가 가능하다.
예를 들면 메일 서비스 클러스터의 부하가 증가하나 웹 서비스 클러스터의 부하가 적을 경우 가장 부하가 적은 노드를 메일 서버 클러스터 쪽으로 중첩시켜서 시스템 효율을 증가시키고 부하를 줄일 수 있다.

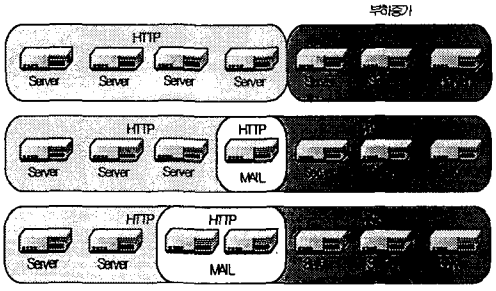

그림 3 주문형 컴퓨팅 모델

2.3 분산 QoS 시스템의 장점

기존시스템에서의 문제점은 모든 처리를 부하분산기가 단독으로 처리하기 때문에 부하증가 및 확장성에 제한이 된다. 이런 단점을 해결하기 위하여 실제적으로 처리하는 부분은 각 노드에서 처리할 수 있도록 분산시키고 부하분산기는 본래 기능인 부하 분산만을 담당하는 것이 과부하를 막고 각 노드에서의 정밀한 부하분산 및 QoS를 적용할 수 있는 형태이다.

분산 구조 시스템	기존 시스템
QoS를 위한 장치를 각각의 노드안에 추가하기 때문에 LB의 부하를 덜어줌	모든 트래픽과 QoS를 모두 처리하기 때문에 과부하 상태에 빠질 가능성 높음
각각의 노드가 자체적으로 부하를 계산하고 관리하기 때문에 세세한 부분까지 감시하고 대비할 수 있음	네트웍 대역만으로 QoS 지원 서버 부하상태 알 수 없음
여유 시스템 자원을 이용하여 추가적인 서비스 지원 가능	대역 제한 기능만 함

표 1 분산구조 시스템과 기존 시스템의 비교

III. 클러스터 기반 분산형 QoS 시스템의 설계 및 구현

3.1 클러스터 환경에서의 QoS

클러스터 환경에서의 QoS를 수행하기 위하여 각 노드가 할 수 있는 일의 양을 특정 metric에 의한 용량으로 환산하여 서비스를 최대 최소를 이용하여 조정 한다.
클러스터가 할 수 있는 서비스의 최대 용량(C_T)
j 노드에서 서비스가 단위시간당 사용하는 용량 (L_j)
j 노드에서 i 서비스의 사용량 ($S_{i,j}$)
j 노드에서 i 서비스의 요청량 ($R_{i,j}$)
C_j는 클러스터 내의 특정 노드 j의 용량으로 정의한다면 클러스터내의 모든 노드를 이용하여 서비스 할 수 있는 용량 C_T 는 $C_T \approx \sum_j C_j$ 라고 할 수 있다. 식(1)은 L_j와 C_T의 관계를 나타내며, 특정 노드 j내에 있는 서비스 양과 각 클러스터의 서비스 양은 식(2)와 같다.

$$0 \leqq L_j \leqq C_j \quad \text{식(1)}$$

$$\sum_i S_i = L_j, \sum_j L_j \leqq C_T \quad \text{식(2)}$$

클러스터 내의 R_i, S_i를 표현하면 식(3)과 같다.

$$\sum_j R_{i,j} \leqq Q_{\min,i} \quad \text{식(3-a)}$$

$$Q_{\min,i} \leqq \sum_j S_{i,j} \leqq Q_{\max,i} \quad \text{식(3-b)}$$

$$Q_{\max,i} \leqq \sum_j S_{i,j} \quad \text{식(3-c)}$$

서비스 사용량에 대한 보장량은 식(4)과 같이 표현할 수 있다.

$$0 \leqq Q_{\min,i} \leqq Q_{\max,i} \leqq C_T \quad \text{식(4)}$$

클러스터내에서 각 노드에서 서비스 i에 대한 사용량

은 식(5)을 이용하여 구할 수 있다.

$$S_{min,\,i} = \sum_j S_{min,\,i,\,j} \quad \text{식(5-a)}$$

$$S_{max,\,i} = \sum_j S_{max,\,i,\,j} \quad \text{식(5-b)}$$

$$0 \leqq \sum_i Q_{min,\,i} \leqq \sum_i Q_{max,\,i} \leqq C \quad \text{식(5-c)}$$

$$\sum_j \sum_i S_{min,\,i,\,j} + \sum_j \sum_i S_{max,\,i,\,j} \leqq C \quad \text{식(6-a)}$$

$$\sum_j \sum_i S_{min,\,i} + \sum_j \sum_i S_{max,\,i} > C \quad \text{식(6-b)}$$

식(6-a)는 클러스터의 상태가 과부하가 아닌 경우이다. 이 경우에는 Q_{min}이 충분하게 처리될 수 있고 또한 여유 리소스가 있기 때문에 추가적인 Q_{max}에 대한 서비스를 수행한다. 식(3-a)를 만족하는 서비스는 반드시 수행하고 식(3-b)를 만족하는 경우에는 시스템 자원이 여유가 있는 경우에 부분적으로 수행하고 수행이 불가능한 경우에는 가장 사용량이 작은 서버로 커넥션을 이전시킨다. 식(3-c)의 조건에서는 대해서는 서비스가 거절된다. 식(6-b)는 과부하 상황의 식이다. 이 경우에는 식(3-a)를 만족하는 것만을 수행하고 나머지는 서비스가 거절된다.

3.2 QoS Metric

1초에 10KByte크기를 갖는 한 개의 웹 컨텐츠를 처리하였을 때, 1 QU/sec라고 표기할 수 있다. 10KByte는 웹 컨텐츠의 평균적인 크기이다. 이 단위의 장점은 단위 시간당 요구의 숫자와 대역폭을 하나의 단위로 묶어서 처리할 수 있다는 것이다. 예를 들면 하나의 커넥션에 하나의 1KByte 요청이 있는 경우나 하나의 10KByte가 있는 경우는 모두 1QU를 소모하지만 하나의 커넥션에서 1KByte 10개를 처리하는 것과 10KByte 크기의 요구를 처리하는 것과는 같다. 이것은 새로운 커넥션을 수용하는데 들어가는 시스템의 부하가 더 크다는 점을 고려할 수 있다. 또한 새로운 커넥션이나 동적인 컨텐츠에 핸디캡을 적용할 수 있다. 100Mbps를 갖는 하나의 노드에서 최대로 할당할 수 있는 QU의 크기는 1250QU이다. 이것은 단순히 최대 대역폭을 10Kbyte로 나눈 것이며 간단한 실험을 통하여 약 1180QU정도의 성능을 보여 줌을 확인 할 수 있다.

3.3 구현 방법

구현은 크게 세부분으로 하였다.

첫 번째는 각 사용자 요구를 차별화 하는 것이다. 3.1에서 언급했던 정책을 이용하여 각 사용자의 요구는 커넥션 단위로 차별화된 Queue에 누적되었다가 높은 우선순위부터 하나씩 웹 서버로 이동된다. VH(very high), H(high), L(low)의 세단계 큐를 가지고 있으며 식(3-a)를 만족할 경우 VH, 식(3-b)를 만족할 경우 H, 식(3-c)를 만족할 경우 L로 Queuing된다. HTTP 서버의 큐가 비었을 경우 VH의 큐에 있는 모든 커넥션에 대하여 먼저 HTTP 서버 리스닝 큐에 넣기 위해 시도하고, 실패했을 경우 High큐는 시도하지 않는다. High큐와 Low큐에 대해서는 시도가 실패할 경우 더 이상 시도하지 않는다. 그림 2는 우선권 Q의 동작 다이어그램이다.

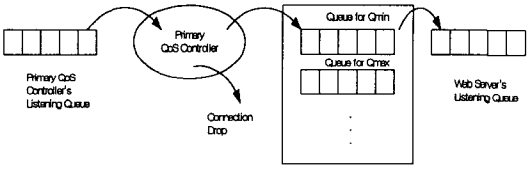

그림 4 우선권 Q의 동작

두 번째는 대역폭 조정이다. 각 서비스 별로 모든 노드에서 입출되는 패킷을 이용하여 대역폭을 조정한다. 이것에 의해 임의의 서비스가 다른 서비스에 의해 방해를 받아서 할당된 충분한 서비스를 하지 못하는 상황을 개선하기 위함이다. 이에따라 입력되는 패킷에 대하여 drop 컨트롤 출력되는 패킷에 대하여 rate 컨트롤을 수행한다.

세 번째는 서버의 부하가 증가하여 임계치에 다다르면 클러스터를 재구성하여 서비스의 품질을 유지한다. 가령 웹 서비스와 메일 서비스에서 웹 서비스 사용자가 증가할 경우 부하가 적은 메일 서비스의 서버를 웹 서비스로 대체 시킨다.

IV. 실험 방법 및 결과

4.1 실험환경

인텔 펜티엄 III CPU 700Mhz가 장착된 6대의 PC를 고속 이더넷 스위치로 연결하여 클러스터를 구성하였다. 사용한 운영체제는 RedHat Linux 7.1 이며 커널은 2.4.2-2이다. 실험에서 CPU의 부하를 최대한 줄이기 위하여 Kernel Thread web service인 TUX를 이용하여 실험하였다. 부하 생성기는 httperf를 사용하였는데 HP에서 서버 실험용으로 제작되었으며 다양한 실험을 위해 많은 파리미터(parameter)를 조절할 수 있도록 고안되어 있다.

4.2 실험결과 및 향후 과제

(1) 응답성 실험

표 2는 6대의 클라이언트 머신에서 1대의 서버로 1MByte 사용자 요구를 계속시도하고 우선권이 높은 서비스에 대하여 하나의 사용자 요구를 했을 경우에 응답에 걸리는 시간을 LAN환경에서 실험하여 측정한 결과이다.

	Non-QoS	QoS
40회 평균 접속시간	757.98msec	0.52msec

표 2. 응답성 실험

이 실험결과는 특정서비스에 부하가 걸릴 경우 다른 서비스를 보호하는 동작을 보여준다.

(2) 차별화 큐의 동작

그림 3은 차별화된 큐의 동작을 보여주는 그래프이다. 서비스 B가 서비스 A보다 우선권이 높을 경우 서비스 B의 수행중에 서비스 A의 입력이 들어오는 경우 먼저 처리하도록 되어있다.

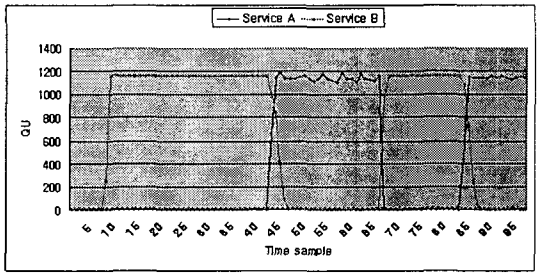

그림 5 우선권 Q의 동작 결과

(3) 서버의 재구성

표 3은 서비스가 처리하는 QU의 양에 따라서 작업 보류중인 서버를 동적으로 추가하여 서비스를 확장시켰을 때 LAN환경에서 응답시간을 측정한 것이다.

	1 Server	2 Servers
40회 평균 접속시간	0.61msec	0.18msec

표 3. 서버의 재구성 응답실험

위의 결과는 분산된 차별화 된 큐와 대역폭 조정이 시스템의 응답성을 조절할 수 있음을 보여준다. 향후 연구방향은 분산 QoS가 높은 확장성을 가질 수 있는 연구를 수행하여 성능을 높이는 것에 있다.

참고문헌

[1] R. Pandey, J.F. Barnes, and R. Olsson. "Supporting Quality of Service in HTTP service" In Proceeding of the 17th SIGACT-SIGOPS Symposium on Principles of Distributed Computing

[2] Valeria Cardellini, Emiliano Casalicchio, Michele Colajanni, "A Performance Study of Distributed Architectures for the Quality of Web Services", 2001

[3] J. Almeida, M. Dabu, A. Manikutty, and P. Cao. ''Providing Differentiated Quality of service in Web Hosting Services.'' In Proceedings of the Workshop on Internet Server Performance, Madison, WI, June 1998.

[4] Nina Hhatti, Rich Friedrich, "Web Server Support for Tired Services", HP Lab, HPL-19s99-160

[5] S. Floyd and V. Jacobson "Link-sharing and resource management models for packet networks", IEEE/ACM

[6] M. Aron. Differentiated and Predictable Quality of service in Web Server Systems. Ph.D. Thesis, Computer Science Department, Rice University, October 2000.

[7] Chang Li, "Quality of Service Guarantee for Cluster-Based Internet Service", Master Thesis, Computer Science Department, State University of Stony Brook, May 2001

[8] J. Fritz Barnes, "Supporting Quality of Service in HTTP Servers", Symposium on principles of distributed computing, 1998