웹 서버 클러스터에서 차별화된 서비스 제공을 위한 서버 노드의 분할 기법

장인재* 최창열* 박기진** 김성수*
아주대학교 정보통신전문대학원* 안양대학교 소프트웨어학과**
{jangij*,clchoi,sskim}@ajou.ac.kr* kiejin@aycc.anyang.ac.kr**

A Partition Mechanism of Server Nodes for SLA in Web Server Cluster

Injae Jang* Changyeol Choi* Kiejin Park** Sungsoo Kim*
Graduate School of Information and Communication, Ajou University*
Dept. of Software, Anyang University**

요 약

최근 웹 서비스가 다양한 컨텐츠와 전자상거래 등 비즈니스와 관련된 서비스로 변화함에 따라 대용량 서비스 뿐만 아니라 고품질 서비스(QoS)를 제공하기 위한 연구가 진행되고 있다. 웹 서버 클러스터에서도 서버 성능 향상과 함께 QoS를 제공하기 위한 차별화된 서비스가 필요하다. 본 논문에서는 사용자 계층별로 차별화된 서비스를 제공하기 위해서 서버 노드를 동적으로 분할하는 기법을 제안한다.

1. 서론

웹 서버 클러스터 구조는 크게 웹 스위치, 전위 서버(Front-end Server), 후위 서버(Back-end Server)로 구성된다. 웹 스위치는 사용자의 요구를 받아들여서 실제로 서비스를 처리하는 전위 서버로 보낸다. 전위 서버는 웹 서버로써 사용자의 정적 요구를 직접 처리하여 응답하고 동적 요구(예: CGI, DB query)에 대해서는 후위 서버에게 서비스 처리를 요청한 후 처리된 결과를 응답하는 구조이며, 구성도는 그림 1과 같다.

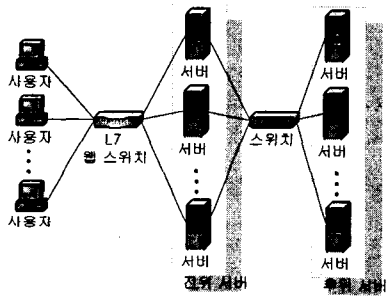

그림 1 웹 서버 클러스터 구성도

위와 같은 구조의 장점은 동적 요구에 대한 것은 전위 서버에서 접속만을 유지하고 실제 서비스는 후위 서버를 이용하게 되므로 정적 요구에 대한 빠른 처리가 가능하다는 것이다. 웹 스위치는 내용기반의 L7 방식을 이용하여 사용자 요구를 상위 계층, 하위 계층, 정적 요구, 동적 요구로 분류하고 웹 서버의 응답을 웹 스위치를 거치지 않고 바로 사용자에게 전달하는 One-way 방식을 따른다[1].

한편, 웹 서버에서 사용자 요구의 증가에 따른 시스템 과부하 상황에서도 QoS를 고려한 차별화된 서비스(SLA, Service Level Agreement)를 제공하기 위한 방법들이 요청되고 있으며 현재까지 서비스 분류, 요구 승인, 서비스 관리 등에 관하여 연구가 진행되고 있다[2]. 그림 1과 같은 웹 서버 클러스터 구조에서는 QoS를 제공하기 위한 자원 관리에 대한 연구가 필요하며, 본 논문에서는 사용자 계층별로 차별화된 서비스를 제공하기 위한 서버 노드를 동적으로 분할 하는 기법을 제안한다.

2. 관련연구

웹 서버 클러스터 구조에서 QoS를 고려한 차별화된 서비스를 제공하는 서버 분할 기법은 크게 정적 분할 기법과 동적 분할 기법으로 나눌 수 있다. 정적 분할 기법은 계층별 사용자의 요구 수준에 따라 고정된 서버 노드를 할당하는 방법으로써 사용자 요구의 수가 자주 변하는 웹에는 적합하지 않다. 동적 분할 기법은 사용자 요구의 수나 필요에 따라 동적으로 서버 노드를 할당하며 휴지 서버 노드를 줄여줌으로써 높은 자원 활용을 제공하는 방법이며 DDSD[3], ARDP[4], DynamicPart[5] 기법등이 있다. 그런데, 이러한 기법들은 일정한 시간 간격으로 서버 노드를 분할하기 때문에 갑작스런 과부하 상태를 처리하기에 적합하지 못하고 특히, DynamicPart 기법은 정적 요구를 전혀 고려하지 않고 동적 요구만을 고려한 기법으로 정적 요구량이 많아질 경우 시스템의 성능 저하가 발생하게 된다.

이 논문은 2003년도 두뇌한국21사업에 의하여 지원되었음.

3. 적응적인 서버 분할 기법 (AdaptivePart)

적응적인 서버 분할 기법은 웹 스위치에서 현재 처리해야 할 정적 요구량과 동적 요구량에 따라 전위 서버와 후위 서버를 각각 동적으로 분할하여 상위 계층의 사용자에 대한 SLA를 만족하는 수준의 서비스를 제공하는 동시에 하위 계층의 사용자에게도 시스템의 과부하가 발생하지 않는 범위내에서 최대한 서비스의 질을 보장할 수 있는 방법이다. 서비스의 계층은 크게 상위 계층(High Class)과 하위 계층(Low Class)으로만 나누고 서비스의 종류는 정적 요구(Static Request)와 동적 요구(Dynamic Request)로 분류한다.

또한, 이와 같이 서비스를 분류했을 때 서버 노드들을 총 4개의 집합으로 분할하여 전위 서버를 상위 계층과 하위 계층으로 각각 {FE_HS} 집합과 {FE_LS} 집합으로 나누고 후위 서버도 동일하게 {BE_HS} 집합과 {BE_LS} 집합으로 나눈다. 이때, 상위 계층과 하위 계층의 정적 요구는 분할된 전위 서버 중 {FE_HS} 집합과 {FE_LS} 집합의 서버에서 각각 처리하고 동적 요구는 분할된 후위 서버중 {BE_HS} 집합과 {BE_LS} 집합의 서버가 각각 처리한다. 4개의 집합으로 서버 노드를 나눈 이유는 서비스 계층과 서비스 종류에 따라 처리 시간이 다르기 때문이다. 즉, 정적 요구와 동적 요구에 따라 서버의 서비스 시간이 서로 다르므로 전위 서버와 후위 서버의 처리량이 다르게 된다. 따라서, SLA를 만족하는 응답시간을 제공할 수 있는 서버 한대의 최대 접속수도 달라진다.

이러한 서버 노드의 분할은 웹 스위치에 접속하는 사용자 요구량에 따라 동적으로 분할하고 서버 노드의 분할 방법은 그림 1과 2의 알고리즘을 이용한다. 서버의 부하량으로 사용자의 접속수를 사용하였는데, 이는 웹 스위치에서 서버의 부하량을 주기적으로 모니터링할 필요없이 각 서버에 연결된 접속수를 쉽게 얻을 수 있어 오버헤드가 발생하지 않기 때문이다.

정적 요구에 대한 전위 서버의 최대 접속수는 $MaxConn_{static}$으로 정하고 동적 요구에 대한 전위 서버와 후위 서버의 최대 접속수는 각각 $MaxConn_{dynamic(FE)}$, $MaxConn_{dynamic(BE)}$ 로 정한다. 또한, 전위 서버의 상위 집합(High Set)에 현재 연결된 접속수에 대해서는 TotalConn으로 나타내고 TotalConn 중 동적 요구에 대한 접속수는 $TotalConn_{dynamic}$ 으로 나타내기로 한다. 그리고 서버 집합중 서버 노드의 개수는 n{서버 집합}으로 나타낸다.

```
T1high = MaxConnstatic × n{FE_HS}
T2high = MaxConndynamic(FE) × n{FE_HS}
T1low = MaxConnstatic × (n{FE_HS} - 1)
T2low = MaxConndynamic(FE) × (n{FE_HS}-1)
if TotalConn > T1high or
    TotalConndynamic > T2high then
    {FE_HS} ← least loaded node {FE_LS};
if TotalConn < T1low and
    TotalConndynamic < T2low then
    {FE_LS} ← most loaded node {FE_HS};
```

그림 2 전위 서버의 분할 알고리즘

그림 2는 전위 서버에 대한 분할 알고리즘을 나타낸다. 첫번째 조건에서는 상위 집합의 서버들이 과부하가 발생한 경우 하위 집합중 부하가 가장 적은 서버 한대를 상위 집합으로 보내고 두번째 조건에서는 반대로 부하량이 일정수준($T1_{low}$, $T2_{low}$)이하로 내려갈 때 부하가 가장 많은 서버 한대를 하위 집합으로 보낸다.

```
Thigh = MaxConndynamic(BE) × n{BE_HS}
Tlow = MaxConndynamic(BE) × (n{BE_HS} - 1)
if TotalConndynamic > Thigh then
    {BE_HS} ← least loaded node {BE_LS};
if TotalConndynamic < Tlow then
    {BE_LS} ← most loaded node {BE_HS};
```

그림 3 후위 서버의 분할 알고리즘

그림 3은 후위 서버에 대한 분할 알고리즘인데 전위 서버에서와 마찬가지로 상위 집합의 부하량에 따라 서버 노드의 추가 및 제거가 발생하므로 상위 계층의 서비스에 대한 SLA를 제공할 뿐만 아니라 하위 계층의 서비스에 대해서도 서버 노드의 여유만큼의 충분한 서비스를 제공할 수 있다. 이러한 방법으로 사용자의 요구 변화에 따라 서버 노드를 동적으로 분할하게 된다.

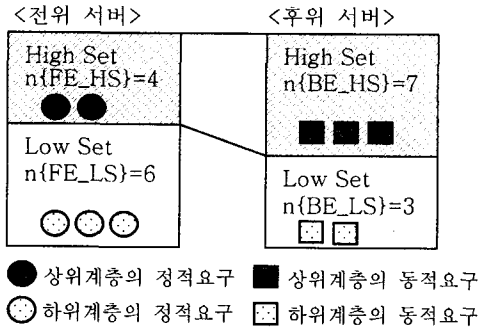

그림 4 서버 노드의 동적 분할

그림 4는 서버 노드의 동적 분할의 예이다. 전위 서버의 개수와 후위 서버의 개수는 동일하게 10개이고, 한 시점에서 전위 서버는 High Set = 4, Low Set = 6개로 나뉘어 졌으며, 후위 서버는 High Set = 7, Low Set = 3개로 나뉜 상태이다. 현재 상위 계층의 SLA를 만족하는 수준의 서비스를 제공한다고 할 때, 정적 요구를 처리하는데 4대의 서버가 필요하고 동적 요구를 처리하는데 7대의 서버가 필요함을 나타낸다. 하위 계층에 대해서는 전위 서버는 6대와 후위 서버는 3대로 정적 요구와 동적 요구를 각각 서비스하며, 이것 이상의 사용자의 요구는 거절함으로써 하위 계층의 SLA를 만족하는 수준의 서비스를 동일하게 제공한다. 또한, 웹 스위치에 접속된 사용자 요구량에 따라 서버 노드의 분할은 동적으로 변하게 됨으로써 갑작스런 사용자 요구증가와 다양한 컨텐츠(정적 요구와 동적 요구)의 변화에 만족할 만한 사용자 응답시간을 보장해 줄 수

있다. 그리고, 그림 4와 같이 서버 노드를 분할했을 때, 서버 집합의 노드들은 해당되는 사용자의 요구만을 처리하게 되므로 서버 노드의 자원들(CPU, 메모리, I/O 등)의 효율과 사용자의 동적 요구량과 정적 요구량을 동시에 고려하여 서버를 할당함으로써 한가지만을 고려하여 할당하는 DynamicPart와 ARDP 기법보다 서버 노드의 활용을 최대한 높일 수 있다.

4. 실험 및 결과

3장에서 제안한 동적 분할 기법(AdaptivePart)의 성능을 분석하기 위해서 시뮬레이션을 수행했다. 웹 서버 클러스터의 전위 서버와 후위 서버의 개수가 각각 10개이고 디스크 전송속도는 20MBps, 메모리 전송속도는 100MBps이다. 또한, 정적 요구와 동적 요구에 대한 작업 부하의 모델은 DynamicPart 기법에서 가정한 것을 이용하였고 표 1과 같다[5].

표 1 웹 클러스터의 작업 부하 모델

정적		Inverse Gaussian	μ=3.86 λ==9.46
		Pareto	α=1.33 κ=2
동적		Hyper exponential	μ_i=700,100,10 λ_i=0.01,0.14,0.85

전체 사용자의 요구 중 상위 계층과 하위 계층의 요구수의 비율은 1:1로써 동일한 요구수로 가정하였다. 그리고, AdaptivePart 기법에서 필요한 서버 한대의 최대 접속수(MaxConn)를 구하기 위해서 웹 스위치와 전위 서버, 후위 서버 한대씩을 가지고 실험하여 각각 $MaxConn_{Static}$ = 30, $MaxConn_{dynamic(FE)}$ = 15, $MaxConn_{dynamic(BE)}$ = 5 를 구했다.

그림 5는 사용자의 정적 요구수와 동적 요구수가 각각 80%, 20%의 비율로 하였을 때 사용자 요구수의 증가에 따른 응답시간 변화이다. 결과에서 보듯이 제안한 기법은 시스템의 과부하 상황에서도 상위 계층의 응답시간이 4초 이하로써 SLA를 만족하는 수준의 서비스를 제공하고 있으며, DynamicPart기법 보다 더 짧은 응답시간을 나타내고 있다.

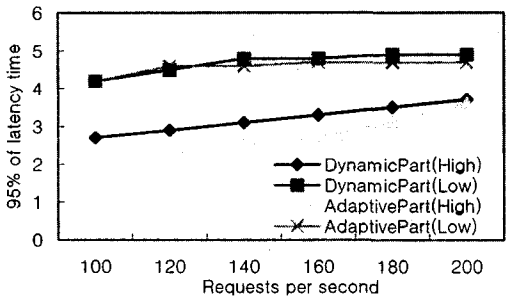

그림 5 사용자 요구수의 증가에 따른 응답시간 변화

그림 6은 사용자의 요구가 초당 200개인 경우 정적 요구수와 동적 요구수의 비율을 변화시켰을때 응답시간을 나타낸다. AdaptivePart 기법은 사용자의 정적 요구와 동적 요구에 상관없이 일정수준의 응답시간을 유지하지만 DynamicPart 기법은 응답시간이 일정치 못하다. 이는 DynamicPart 기법은 동적 요구만을 고려하지만 AdaptivePart 기법은 동적 요구뿐만 아니라 정적 요구를 고려해서 서버를 분할하기 때문이다.

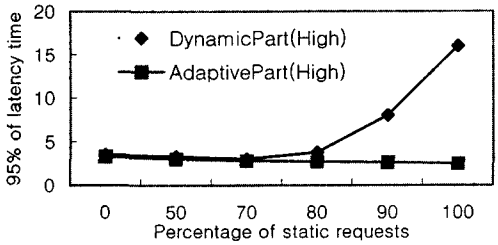

그림 6 정적/동적요구 비율에 대한 사용자 응답시간

5. 결론

웹 서버 클러스터는 고가용성과 고성능을 제공해 주는 비용효율적인 대안으로서 웹 서비스를 위한 확장성이 우수한 시스템이지만 지금까지 서버의 성능 향상만을 고려하였다. 따라서, 본 연구에서는 QoS를 위한 차별화된 서비스를 제공할 수 있는 서버의 동적 분할 기법(AdaptivePart)를 제안하였고 사용자의 정적 요구와 동적 요구를 동시에 고려한 기법으로서 상위 계층의 서비스에 대한 SLA를 보장해 줄 수 있다. 실험을 통하여 기존의 기법보다 더 좋은 서비스를 제공할 수 있음을 알수 있었다. 앞으로, 웹 스위치에서 QoS를 고려한 차별화된 서비스를 제공할 수 있는 다양한 방법들에 대한 연구가 요청된다.

참고문헌

[1] T.Schroeder, S.Goddard, and B.Ramamurthy, "Scalable Web Server Clustering Technologies," IEEE Network, pp. 38-45, May 2000.

[2] N.Bhatti and R.Friedrich, "Web Server Support for Tiered Services," IEEE Network, pp. 64-71, Sep. 1999.

[3] H.Zhu, H.Tang, and T.Yang, "Demand-Driven Service Differentiation in Cluster-based Network Servers," Proceedings of IEEE Infocom, pp. 679-688, Apr. 2001.

[4] J.Zhang, T.Hamalainen, and J.Joursensalo, "A New Mechanism for Supporting Differentiated Services in Cluster-based Network Servers," Proceedings of 10th IEEE International Symposium MASCOTS, Vol. 10, pp. 427-432, 2002.

[5] V.Cardellini, E.Casalicchio, M.Colajanni, and M. Mambelli, "Web Switch Support For Differentiated Services," ACM Performance Evaluation Review, Vol. 29, pp. 14-19, 2001.