SLA를 지원하는 동적 성능 분리 기법

박범주[0] 박기진[00] 강명구[00] 김성수[000]
[0]삼성전자 첨단기술연수소
[00]아주대학교 산업정보시스템공학부
[000]아주대학교 정보통신전문대학원
bumjoo[0]@samsung.com {kiejin[00], mkkang81[00] , sskim[000]}@ajou.ac.kr

Dynamic Performance Isolation for SLA

Bumjoo Park[0] Kiejin Park[00] Myeongkoo Kang[00] Sungsoo Kim[000]
Advanced Technology Training Institute, Samsung Electronics[0]
Division of Industrial & Information Systems Engineering, Ajou University[00]
Graduate School of Information and Communication, Ajou University[000]

요 약

본 논문에서는 사용자 계층별 요청률에 따라 웹 서버 컴퓨팅 노드들의 성능 분리를 동적으로 수행하는, 퍼지 이론을 적용한 웹 서버 성능 분할 기법에 관하여 논하였다. 제안된 기법은 컴퓨팅 노드의 현재 부하량, 사용자 계층별 요청률을 퍼지 입력 변수(Fuzzy Variables)로 하여, 애매모호한 노드의 정량적 부하를 정성적으로 표현할 수 있도록 하였으며, 이를 통해 계층별 요청률이 급격한 변화에 대응하여, 계층별 요청을 처리하는 담당 노드의 수를 동적으로 조절할 수 있게 하였다. 제안된 기법에 대한 성능분석을 통해 퍼지정리를 활용한 기법이, 이를 사용하지 않은 기법에 비해 우수한 응답시간 성능을 보여주고 있음을 검증하였다.

1. 서 론

SLA(Service Level Agreement)를 지원하는 클러스터 기반 웹 서버에서 계층별 사용자 요청에 따라 그에 알맞은 특정 컴퓨팅 노드들을 할당하는 컴퓨팅 노드 분할 기법은 웹 상에서 차별화 서비스를 제공하기 위한 개념 중 하나인 성능 분리(Performance Isolation)에 해당하며, 성능 분리된 노드마다 서로 다른 계층의 요청 처리를 담당하게 하고, 상위 계층의 사용자 요청일수록 더 많은 컴퓨팅 노드를 할당해 줌으로써, 차별화 서비스를 보장할 수 있게 된다. 이때 부하분배기(Load Balancer)는 계층별 사용자의 요청을 처리할 컴퓨팅 노드를 결정하며, 웹 서버의 상태(요청률, 각 노드의 현재 부하)를 정확히 파악하고 있어야, 사용자 계층별 요청을 최적으로 분배할 수 있다. 따라서 부하분배기의 핵심적인 성능 요소는 웹 서버를 구성하는 각 노드의 부하 균형도 및 응답시간이라 볼 수 있다.

하지만 컴퓨팅 노드의 부하량 판단 작업에 내재하는 불확실성으로 인해 최적 분배를 달성하기는 현실적으로 어려운 상황이다. 과연 "현재 특정 노드의 CPU 사용률이 높다고, 혹은 대기하고 있는 작업의 수가 많다고, 또는 메모리 사용량이 많다고 특정 노드가 바쁘다고 정확히 말할 수 있는가?" 예를 들어 비록 노드에서 처리 대기중인 사용자 요청의 수가 많아도, 대기 중인 개개의 요청 작업이 간단한 정적 컨텐츠(Static Content)만을 처리하는 경우일 때는, 노드에 부하를 많이 주지는 못한다. 이처럼 웹 서버 부하량 판단 혹은 사용자 요청률의 변화 시에 발생하는 애매모호한 상황(Fuzziness)을 표현하기 위해, 퍼지제어 알고리즘에 기초한 부하분배 메커니즘을 고려할 필요성이 있다[1].

본 연구는 한국학술진흥재단 선도연구자지원사업(KRF-2005-041-D00630) 지원으로 수행되었음

2. 관련연구

Layer-4 기반 부하 분배 방식은 TCP/IP 레벨에서 동작하며, HTTP요구가 보내지기 전에 TCP/IP 연결 설정이 이루어지기 때문에 컴퓨팅 노드에 대한 선택이 요구된 내용에 의해 결정될 수 없으나, 이에 비하여 Layer-7 기반 부하 분배 방식은 컴퓨팅 노드를 결정하기 전에 HTTP요구를 분석할 수 있기 때문에 클라이언트 요구 내용에 대하여 상세한 정보를 고려한 분배가 가능하다[2]. 기존에는 클러스터 웹 서버의 부하 조절(Load

Balancing)을 위해서 Layer-4기반 스위치가 사용되었으나, 최근 내용 기반 분배가 가능한 Layer-7스위치로 급속히 대체되고 있다.

LARD(Locality-Aware Request Distribution)[3]는 웹 서버 클러스터에서 내용 기반 부하 분배에 대한 초기의 연구로써, 정적(Static)인 웹 문서는 컴퓨팅 노드와 일대일 대응을 유지하며 주어진 문서에 대하여 첫 번째 요청이 도달하면 가장 적은 부하가 걸려 있는 컴퓨팅 노드에 우선적으로 할당한다. 한편, 상위 계층의 사용자의 SLA를 만족하는 서비스를 제공하기 위해, 계층별 부하량에 따라 서버 노드를 동적으로 분할하는 기법은 특정 계층을 서비스하는 노드에 과부하가 걸렸을 경우, 이보다는 더 낮은 계층을 서비스하고 있는 서버를 추가적으로 이용하거나, 과부하 상황이 해소되면 다시 서버를 반납하는 것을 기본 개념으로 하고 있다. 동적 분할 기법은 사용자의 계층별 요청 수준 혹은 필요에 따라서, 동적으로 노드를 할당하며, DDSD(Demand-driven Service Differentiation), Dynamic Partitioning 기법이 있다[4].

DDSD기법은 사용자의 요청량에 따라 CPU와 디스크 I/O용량을 할당함으로써 차별화된 서비스를 제공하고, 동적 요청이 많은 상황에 적합하다. Dynamic Partitioning 기법은 SLA(Service Level Agreement)에 기반한 서비스를 상위 계층 사용자에게 제공하기 위해, 웹 서버의 부하량에 따라 컴퓨팅 노드를 동적으로 분할하는 기법이다. 이러한 동적 분할 개념은 부하 변화에 따라 여러 사용자 계층에게 효율적이고 질 높은 서비스를 제공할 수 있지만, 정적 요청 비율이 증가할 경우 정적 분리 기법이 더 좋은 성능을 제공하는 문제점을 가지고 있다. .

3. 시스템 모델

그림1은 본 논문에서 대상으로 하는 Layer-7 스위치 기반 One-way 웹 서버 구조이다. 두 계층의 사용자(NC: Normal Class User, PC: Premium Class User)를 가정하였으며, 컴퓨팅 노드는 일반 사용자의 요청을 처리하는 일반 노드(이하 NC), 고급 사용자를 서비스 하는 상위 노드(이하 PC)로 나눈다. 내용기반 부하분배가 가능한 L7 스위치를 사용할 경우, 컴퓨팅 노드를 결정하기 전에 HTTP 요청을 분석할 수 있기 때문에 사용자 요청 내용에 대하여 상세한 정보를 고려한 처리가 가능하다. 웹 서버 구조에 설계에 적용된 가정은 다음과 같다.

- 계층별 사용자 요청은 일반 사용자 요청과 고급 사용자 요청으로 구분되며, 각각의 요청은 해당되는 컴퓨팅 노드에서만 처리된다.
- 컴퓨팅 노드의 부하 및 계층별 사용자 요청률 변화에 따라 해당 계층을 서비스하는 컴퓨팅 노드의 임대(Borrowing) 및 대여(Lending) 동작이 발생한다. 예를 들어 고급 사용자의 요청이 증가하여, PC의 성능이 떨어질 경우, PC는 일반 사용자를 대상으로 서비스하는 노드(NC)를 임대하여 사용한다.

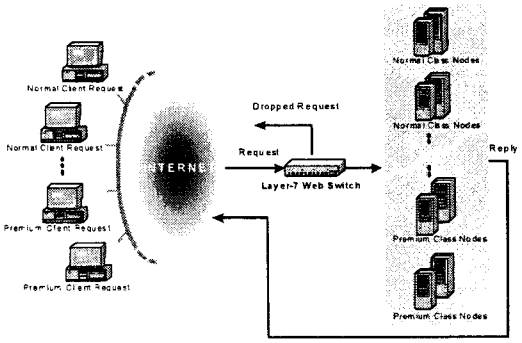

그림 1. Layer-7 스위치 기반 One-way 웹서버 구조

4. 퍼지기반 웹서버 성능 분리 기법

퍼지 기반 웹 서버 성능 분리 기법에 핵심적인 역할을 수행하는 퍼지 제어기에 입력되는 퍼지 변수를 재구성(임대 및 대여) 가능한 웹 서버 컴퓨팅 노드의 부하량 및 사용자 계층별 요청률(Arrival Rate)로 설정하였다. 웹 서비스에서는 사용자가 최초 접속하여 원하는 일을 모두 마치고 나갈 때까지 여러 단계의 연속된 작업(Session)을 거치기 때문에 단순히 접속자 수만을 기준으로 웹 서버의 부하량을 판단하기 어려우며, 각 세션을 기준으로 요청 주기, 사용자 요청 검토 시간(Think Time), 요청한 내용의 종류(Static/Dynamic), 및 대기 작업의 수 등을 통합적으로 고려해야 한다. 하지만 이와 같은 사용자 요청과 관련된 이벤트들의 정량적 분석이 용이하지 않으며, 사용자 계층별 요청률이 특정 시간대에 급격하게 증가하거나 감소할 수 있기 때문에, 어느 특정값을 기준으로 요청률이 "높다" "낮다"를 판단하기보다 "매우 증가" "점진적 증가" "급격 감소" 등의 애매모호한 요청률 상태 정보를 반영할 수 있도록 하였다.

4.1 멤버쉽 함수

각 계층별 사용자 요청률은 낮은 경우(L: Low), 중간인 경우(M: Medium), 높은 경우(H: High) 등 3단계의 퍼지 변수로 구분되며, 이들의 애매모호함의 정도를 나

타내는 멤버쉽 함수는 식(1) ~ 식(3)과 같이 정의된다.

$$L_i = \begin{cases} 0 & when \quad y \geq b_{i1} \\ (y - b_{i0})/(b_{i1} - b_{i0}) & when \quad b_{i0} < y < b_{i1} \\ 1 & when \quad y = b_{i0} \end{cases} \quad (1)$$

$$M_i = \begin{cases} 0 & when \quad y = b_{i0} \text{ or } y = b_{i2} \\ (y - b_{i0})/(b_{i1} - b_{i0}) & when \quad b_{i0} < y < b_{i1} \\ (y - b_{i1})/(b_{i2} - b_{i1}) & when \quad b_{i1} < y < b_{i2} \\ 1 & when \quad y = b_{i1} \end{cases} \quad (2)$$

$$H_i = \begin{cases} 0 & when \quad y \leq b_{i1} \\ (y - b_{i1})/(b_{i2} - b_{i1}) & when \quad b_{i1} < y < b_{i2} \\ 1 & when \quad y = b_{i2} \end{cases} \quad (3)$$

두 사용자 계층을 서비스 하는 서버 노드의 부하는 매우 낮은 경우(VL: Very Low), 낮은 경우(L: Low), 중간인 경우(Medium), 높은 경우(L: High), 매우 높은 경우(VL: Very High) 등 5단계의 퍼지 변수로 구분된다. 이들의 애매모호함의 정도를 나타내는 멤버쉽 함수는 다음의 식(4)~식(5)과 같이 정의된다.

$$VL_i = \begin{cases} 1 & when \quad x \leq a_{i1} \\ (a_{i1} - x)/(a_{i2} - a_{i1}) & when \quad a_{i1} < x < a_{i2} \\ 0 & when \quad x \geq a_{i2} \end{cases} \quad (4)$$

...

$$VH_i = \begin{cases} 1 & when \quad x \leq a_{i5} \\ (x - a_{i4})/(a_{i5} - a_{i4}) & when \quad a_{i4} < x < a_{i5} \\ 0 & when \quad x \geq a_{i5} \end{cases} \quad (5)$$

4.2 퍼지 추론 엔진

표 1 퍼지 추론 규칙

Very Low	Negative Large	Negative Moderate	Negative Small
Low	Negative Moderate	Negative Small	Approximately Zero
Medium	Negative Small	Approximately Zero	Positive Small
High	Approximately Zero	Positive Small	Positive Moderate
Very High	Positive Small	Positive Moderate	Positive Large

퍼지 제어기로 입력된 요청률과 부하량은 멤버쉽 함수에 의해 해당 집합에 속하는 정도인 퍼지 입력값으로 변환되어야 하며, 이때 Crisp 입력값이 여러 멤버쉽 함수 구간의 값을 동시에 가질 경우, 최대 퍼지 값을 갖는 멤버쉽 함수값은Max-Min 알고리즘을 적용하여 출력한다[5]. 이와 같은 방식에 의거하여, 모든 퍼지 입력 변수의 애매모호함이 결정되면, 표 1에 나타난 규칙에 의해 추론엔진(Inference Engine)은 최종 추론 결과를 출력한다.

4.3 퍼지 출력

퍼지 추론을 거치면 단일 퍼지 출력값(서버 노드 수의 증가/감소 정도)이 나오며, 이를 처리하는 두 사용자 계층에 대한 퍼지 출력 변수의 멤버쉽 함수는 매우 작게(NL), 작게(NM), 약간 작게(NS), 변동 없음(AZ), 약간 크게(PS), 크게(PM) 및 매우 크게(PL)로 7단계의 퍼지 함수로 구분된다. 이들의 애매모호함의 정도를 나타내는 멤버쉽 함수를 그래프로 나타내면 그림 2와 같다.

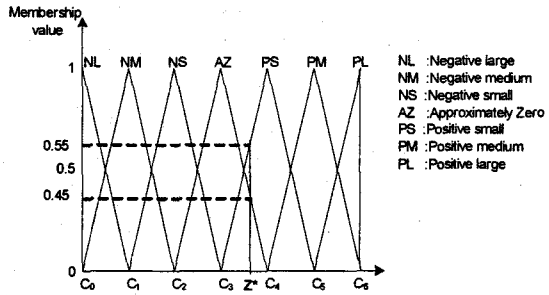

그림 2. 퍼지 출력값의 멤버쉽 함수

퍼지 출력값은 비퍼지화 과정을 거쳐 최종 제어값(Crisp)으로 변환하는데 데, 이 경우 COA(Center of Area) 기법을 적용하여 출력값을 계산하였다[5].

5. 성능 분석

퍼지기반 웹 서버 클러스터 시스템 성능 분리 기법의 성능 평가를 위해, 응답시간에 대한 非퍼지기법과의 비교분석을 진행하였다. 웹 서버는 각 사용자 계층으로부터 정적 요청과 동적 요청을 받아들이며, 시뮬레이션에 사용된 시스템의 운영 파라미터는 표 2, 3 과 같다.

표 2. 정적 동적 요청 부하 모델

Dynamic Requests	700 msec.	0~100
Static Requests	100 msec.	0~100

표 3. 시스템 파라미터

서버 수	10 ~ 30 대
계층별 요청 도착 비율 $[class_P, class_N]$	[0.2, 1.0]

그림 3은 Premium • Normal 계층별 정적 • 동적 요청에 따른 응답시간을 퍼지 • 비퍼지 기법에 따라 표시하였다. 전체적으로, 동적 요청과 정적 요청 모두 퍼지기법이 비퍼지 기법에 비해 Premium 계층과 Normal 계층 응답시간 성능이 좋게 나왔다. 이는 퍼지기법의 경우, Premium 계층이 과부하 상태일 경우 Normal 계층을 서비스하는 서버를 상위 계층에 효과적으로 제공해주기 때문이다. 한편, 정적 요청은 변하지 않는 문서 형식의 파일이고, 동적 요청은 자주 변하면서 처리하는데 시간이 많이 걸리는 VOD와 같은 동영상 파일이라 퍼기지법과 비퍼지기법 모두 대체적으로 정적 요청이 동적 요청에 비해 응답시간 성능이 좋게 나온 것을 알 수 있다.

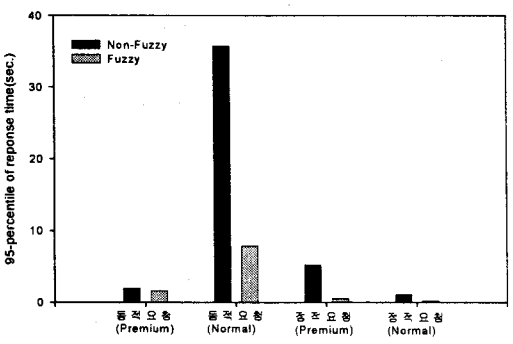

그림 3. 퍼지기법 사용여부에 따른 정적•동적 요청의 응답시간 비교

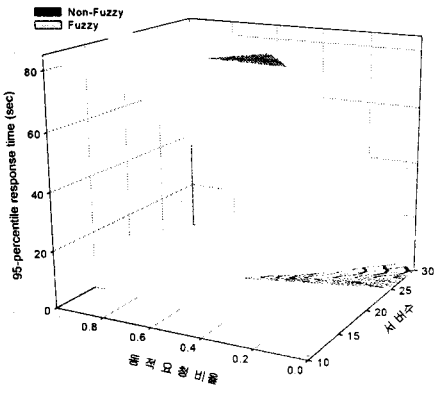

그림 4. 서버수/동적요청 비율 변화에 따른 응답시간 비교

그림 4에서는 Normal 계층에 대해 서버수 및 동적요청 비율 변화에 따른 응답시간을 비교분석 하였다. 기본적으로, 퍼지기법과 비퍼지 기법 모두에서 서버수가 증가할수록 응답시간이 감소하는 경향을 보이고 있으며, 동적요청 비율이 증가하면 응답시간이 증가하는 경향을 보이고 있다. 하지만, 퍼지기법의 경우 증가폭이 완만한 경향을 보이게 되므로 유의할만한 시스템 성능저하가 일어나지 않지만, 비퍼지기법의 경우 동적요청 비율 변화에 응답시간이 민감하게 반응하고 있음을 알 수 있다. 이러한 경우, 서버수 변화보다 동적 요청 비율이 급격히 증가할 때 시스템 가동성능 저하가 예상된다.

6. 결 론

본 논문에서는 퍼지 이론을 적용한 웹 서버 성능 분할 기법에 관하여 논하였다. 제안된 기법은 컴퓨팅 노드의 현재 부하량, 사용자 계층별 요청률을 퍼지 입력 변수로 하여, 애매모호한 노드의 정량적 부하를 정성적으로 표현할 수 있게 함으로써 이를 통해 계층별 요청률이 급격한 변화에 대응하여, 계층별 요청을 처리하는 담당 노드의 수를 동적으로 조절할 수 있게 하였다. 성능분석을 통해 제안된 퍼지 기반 성능 분리 방식의 웹 서버 응답시간이 차별화 서비스를 위해 非퍼지기법에 비해 개선되는 것을 확인할 수 있었다.

향후, 퍼지기법을 적용한 승인제어 알고리즘을 제안하고 해당 알고리즘에 대한 퍼지 기법과 비퍼지기법의 성능 비교분석을 통해 차별화 서비스를 보다 체계적으로 지원하기 위한 연구를 수행할 예정이다.

[참고문헌]

1. A. Shaout, P. McAuliffe, "Job scheduling using fuzzy load balancing in distributed system," Electronics Letter, 34(20): pp. 1983-1985, 1998.
2. M. Aron, et al., "Scalable Content-aware Request Distribution in Cluster-based Network Servers", Proceedings of the 2000 Annual Usenix Technical Conference, 2000.
3. L. Cherkasova and M. Karlsson, "Scalable Web Server Cluster Design with Workload-aware Request Distribution Strategy WARD", 3rd International Workshop on Advanced Issues of E-Commerce andWeb-Based Information Systems pp. 212 -221, 2001.
4. M. Andreolini, et al., "A cluster-based web system providing differentiated and guaranteed services," Cluster Computing, 7(1): pp. 7-19, 2004.
5. J. Mendel, "Fuzzy logic systems for engineering: A tutorial," Proceedings of IEEE, vol. 83, pp. 345-377, Mar. 1995.