퍼지기법에 기초한 로드분배 방식에 의한 웹서버 성능향상

(A Fuzzy Technique-based Web Server Performance Improvement Using a Load Balancing Mechanism)

박 범 주 † 박 기 진 †† 강 명 구 ††† 김 성 수 ††††
(Bumjoo Park) (Kiejin Park) (Myeongkoo Kang) (Sungsoo Kim)

요 약 본 논문에서는 차별화 서비스를 지원하는 웹 서버의 응답시간 성능을 향상시키기 위해 기존의 동적 성능 분리 기법에 퍼지 기법을 접목한다. 특히, 클러스터 기반 웹서버 시스템의 부하량에 대한 판단 기준 혹은 사용자 요청율 및 동적요청 비율 변화시에 발생하는 애매모호한 상황을 효과적으로 반영하기 위해, 퍼지제어 기법에 기초한 로드분배 메커니즘을 제안하였다. 이를 통해, 기존의 퍼지 기법을 활용하지 않은 성능분리 기법과 퍼지기법을 활용한 경우에 대해 응답시간(95-percentile of Response Time) 성능 비교 평가를 통해 퍼지기반의 성능분리 기법이 차별화 서비스 시스템의 성능을 더욱 강건하고 효율적으로 향상시킬 수 있다는 점을 검증하였다.

키워드 : 차별화 서비스, 성능분리, 퍼지 제어 기법

Abstract This paper combines fuzzy concepts with an existing dynamic performance isolation technique in order to improve the response time performance of a Web server supporting differentiated services. A load balancing mechanism based on the fuzzy control technique is developed in such a way that ambiguous situations caused by workload estimation of cluster-based Web servers, client request rates, and dynamic request rates can be represented in an effective way. In addition, we verify that the fuzzy-based performance isolation technique improves the performance and robustness of differentiated service systems efficiently through comparing 95-percentile of response time between the fuzzy-based performance isolation technique and the existing one, which do not use the fuzzy concept.

Key words : Differentiated Services, Performance Isolation, Fuzzy Control Technique

1. 서 론

클러스터 기반 웹 서버에서 고품질 차별화 서비스를 제공하기 위해서는 계층별 사용자 요청 처리 능력을 높이는 것이 중요하며, 이를 위해서는 웹 서버를 구성하는 개별 컴퓨팅 노드의 부하(Load)를 정확히 파악해야 한다. 차별화 서비스는 웹 서비스 공급자가 SLA(Service Level Agreement)와 같은 방식을 활용하여 고객에게 다양한 서비스 수준에 따라 웹 서비스를 제공하는 것을 의미한다. 일반적으로 컴퓨팅 노드의 부하는 CPU 사용률, 메모리 사용량, 혹은 대기 작업의 수 등으로 판단하고 있으며, 노드의 부하가 매우 높은 상황(Heavily-loaded) 혹은 부하가 작게 걸린 상황(Lightly-loaded)에 대한 판단 기준 등이 응용 분야에 따라 매우 복잡할 수 있다. 웹 서버를 구성하는 컴퓨팅 노드 사이의 부하 조절을 위해 최근 많이 사용되고 있는 Layer7 스위치는

· 본 연구는 2007년 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행되었음(KRF-2006-331-D00438)

† 학생회원 : 아주대학교 정보통신전문대학원
bumjoo@samsung.com
†† 종신회원 : 아주대학교 산업정보시스템공학부 교수
kiejin@ajou.ac.kr
††† 정 회 원 : TmaxSoft R&D 센터 SoA실
myeongkoo_kang@tmax.co.kr
†††† 종신회원 : 아주대학교 정보통신전문대학원 교수
sskim@ajou.ac.kr

논문접수 : 2006년 9월 11일
심사완료 : 2008년 1월 10일

정보과학회논문지: 시스템 및 이론 제35권 제3호(2008.4)

컴퓨팅 노드를 결정하기 전에, HTTP(Hypertext Transfer Protocol) 요청을 분석할 수 있기 때문에 사용자 요청을 고려한 분배(*Context Based Load Balancing*)가 가능하다[1].

한편, 계층별로 구분된 사용자 요청에 따라 그에 알맞은 특정 컴퓨팅 노드들을 할당하는 성능 분리(Performance Isolation) 기법을 통해, 특정 컴퓨팅 노드마다 서로 다른 계층의 요청 처리를 담당하게 하고, 상위 계층의 사용자 요청일수록 더 많은 컴퓨팅 노드를 할당해 줌으로써, 차별화 서비스를 보장할 수 있다. 이때 계층별 사용자의 요청을 처리할 컴퓨팅 노드를 결정하는 부하 분배기(Load Balancer)는, 웹 서버의 상태(요청율, 각 노드의 현재 부하)를 정확히 파악하고 있어야만, 계층별 사용자 요청을 최적으로 분배할 수 있다. 따라서 로드분배기의 핵심적인 성능 요소는 웹 서버를 구성하는 각 노드의 부하 균형도 및 응답시간이라 볼 수 있다.

하지만, 웹 서버 컴퓨팅 노드의 부하량 판단 작업에 내재하는 불확실성으로 인해 최적 분배를 달성하기는 현실적으로 어려운 상황이다. 과연 "현재 특정 컴퓨팅 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 요청율의 변화시에 발생하는 애매모호한 상황(Fuzziness)을 표현하기 위해, 퍼지제어 알고리즘에 기초한 로드분배 기법을 고려할 필요성이 있다[2,3].

2장에서는 본 연구 주제와 관련된 기존의 연구 내용에 대해 정리하였고, 3장에서는 본 논문에서 대상으로 하는 웹 서버 시스템 모델에 대해 기술하였다. 4장에서는 퍼지 기반 웹 서버 성능 분리 기법의 세부 내용을 다루었으며, 5장에서는 퍼지기반 웹 서버 클러스터 시스템 성능 분리 기법의 성능을 다양한 시뮬레이션 결과를 통해 평가하였고, 마지막으로 6장에서는 연구결과를 요약하고 향후 연구방향에 대해 기술하였다.

2. 관련 연구

네트워크 단계의 QoS(Quality of Services)만으로는 인터넷 비즈니스 양단간 QoS를 지원할 수 없는 문제점을 해결하기 위하여, 웹 서버 단계에서 클라이언트 요청의 우선 순위에 따라 차별화된 서비스를 제공할 수 있는 기법에 대한 연구가 수행된 바 있다[4]. 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 Layer7 기반 부하 분배 방식은 컴퓨팅 노드를 결정하기 전에 HTTP 요구를 분석할 수 있기 때문에 클라이언트 요구 내용에 대하여 상세한 정보를 고려한 분배가 가능하다[5,6]. 기존에는 웹 서버의 부하 조절(Load Balancing)을 위해서 Layer-4기반 스위치가 사용되었으나, 최근 내용 기반 분배가 가능한 Layer7스위치로 급속히 대체되고 있다.

LARD(Locality-Aware Request Distribution)[7]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 기법 등이 있다[8,9].

DDSD기법은 사용자의 요청량에 따라 CPU와 디스크 I/O용량을 할당함으로써 차별화된 서비스를 제공하고, 동적 요청이 많은 상황에 적합하다. Dynamic Partitioning기법은 SLA에 기반한 서비스를 상위 계층 사용자에게 제공하기 위해, 웹 서버의 부하량에 따라 컴퓨팅 노드를 동적으로 분할하는 기법이다. 즉, 상위 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. 이러한 동적 분할 개념은 부하 변화 따라 여러 사용자 계층에게 효율적이면서도 질 높은 서비스를 제공할 수 있지만, 정적 요청 비율이 증가할 경우 정적 분리 기법이 더 좋은 시스템 성능을 제공하는 문제점을 가지고 있다.

본 논문에서는 사용자 계층별 요청율에 따라 웹 서버 컴퓨팅 노드들의 성능 분리를 동적으로 수행하는, 퍼지 이론을 적용한 웹 서버 성능 분할 기법에 관하여 논한다. 제안된 기법에서는 컴퓨팅 노드의 현재 부하량, 사용자 계층별 요청율을 퍼지 입력 변수(Fuzzy Variables)로 하여, 애매모호한 노드의 정량적 부하를 정성적으로 표현할 수 있으며, 이를 통해 계층별 요청율이 급격한 변화에 대응하여, 계층별 요청을 처리하는 담당 노드의

수를 동적으로 조절할 수 있다. 제안된 기법에 대한 성능분석을 통해 퍼지정리를 활용한 기법이, 이를 사용하지 않은 기법에 비해 우수한 성능을 보여주고 있음을 검증하였다.

3. 시스템 모델

그림 1은 본 논문에서 대상으로 하는 Layer7 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 서비스하는 상위 노드(이하 PC)로 나눈다. 내용기반 로드분배가 가능한 Layer7 스위치를 사용할 경우, 컴퓨팅 노드를 결정하기 전에 HTTP 요청을 분석할 수 있기 때문에 사용자의 요청 내용에 대하여 상세한 정보를 고려한 처리가 가능하다.

정적 서비스 또는 무료 사용자를 일반 사용자로, 동적 서비스나 유료 사용자를 고급 사용자로 구별하며, 이들로부터의 요청은 Layer7 스위치를 통해 해당 서비스를 담당하여 처리하는 노드에 도착된 후, 그 노드에서 직접 결과를 전달받는 One-way 웹 서버 구조를 채택하였다. One-way 구조에서는 들어오는 패킷은 웹 스위치를 거쳐 웹 서버로 전달되는 반면에 나가는 패킷은 웹 스위치를 거치지 않고 직접적으로 사용자에게 전달되는 구조를 가지며, Two-way 구조는 클러스터 안의 각각의 서버는 유일한 하나의 IP 주소로 설정되어 들어오는 패킷과 나가는 패킷은 웹 스위치에 의해 재 작성된다. 따라서 One-way 구조는 Two-way 구조에 비추어 스위치의 오버헤드를 대폭 줄일 수 있다.

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 요청율 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. 이와 같은 동적인 임대와 대여 동작을 통해, 계층별 서비스 노드간의 부하 균형 및 계층별 사용자 요청에 대한 차별화 서비스를 제공할 수 있다.

4. 퍼지기반 웹서버 성능 분리 기법

퍼지 기반 웹 서버 성능 분리 기법에 핵심적인 역할을 수행하는 퍼지 제어기에 입력되는 퍼지 변수(Fuzzy Variable)를 재구성(임대 및 대여) 가능한 웹 서버 컴퓨팅 노드의 부하량 및 사용자 계층별 요청율(Arrival Rate)로 설정한다. 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 판단하기는 어려우며, 각 세션을 기준으로 요청 주기, 사용자 요청 검토 시간(Think Time), 요청한 내용의 종류(Static/

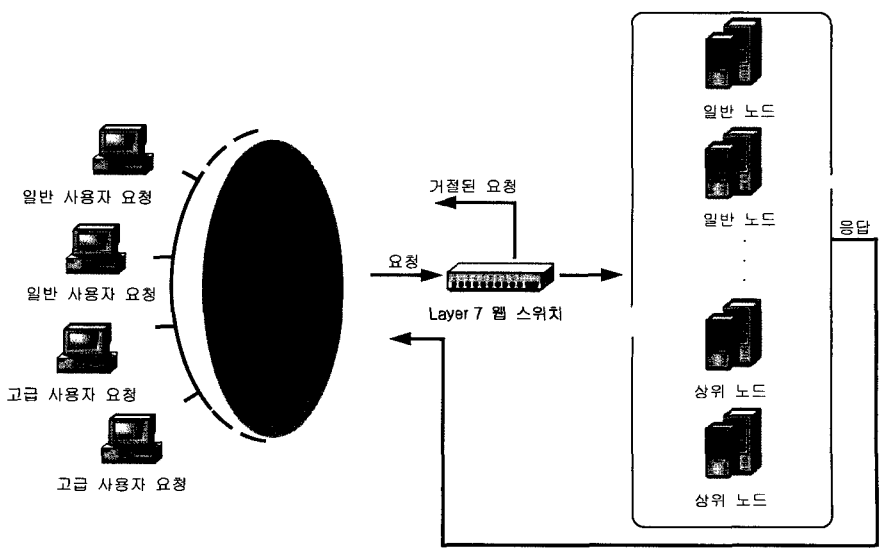

그림 1 Layer7 스위치 기반 One-way 웹 서버 구조

Dynamic), 및 대기 작업의 수 등을 통합적으로 고려하여야 한다. 하지만 이와 같은 사용자 요청과 관련된 이벤트들의 정량적인 분석이 용이하지 않으며, 사용자 계층별 요청율이 특정 시간대에 급격하게 증가하거나 감소할 수가 있기 때문에, 어느 특정값을 기준으로 요청율이 "높다" "낮다"를 판단하기 보다는 "매우 증가" "점진적 증가" "급격 감소" 등의 애매 모호한 요청율 상태 정보를 반영할 수 있도록 한다.

위에 언급한 상황을 표현할 퍼지 변수를 도입한 퍼지 성능 분할 제어기의 구조는 그림 2와 같다. 계층별 요청율과 서버 노드의 부하량은 퍼지화기에 의해 퍼지 변수 입력값으로 변환된 후, 퍼지 추론 엔진에서는 퍼지 규칙 및 멤버쉽 함수를 이용하여, 최종 퍼지 출력값을 산출한다. 퍼지 출력값은 여전히 퍼지한 성질을 가지고 있기 때문에 이를 제거하기 위해 비퍼지화기에서 처리된 후, 최종적인 제어 동작(즉 서버 노드의 임대 및 대여를 위한 Crisp 한 값)을 성능 분리기(e.g. 로드분배기)에 지시한다. 정리하면, 퍼지 성능 분할 제어기의 입력과 출력은 Crisp 한 이진 값을 가지며, 퍼지 제어기 내부에서만 애매모호한 성질을 가진 퍼지 변수로 처리된다. 이와 같은 제어 동작은 요청율 및 노드 부하량 변동에 따라 계속해서 반복하여, 동적인 성능 분할 기능을 수행한다.

4.1 멤버쉽 함수

각 계층별 사용자 요청율은 낮은 경우(L: Low), 중간인 경우(M: Medium), 높은 경우(H: High) 등 3단계의 퍼지 변수로 구분되며, 이들의 애매모호함의 정도를 나타내는 멤버쉽 함수는 다음의 식 (1)~식 (3)과 같이 정의되고 그래프로 나타내면 그림 3과 같다(i=1일 때 일반사용자, i =2는 고급사용자를 의미). 예를 들어 고급사용자의 사용자 요청율이 Y*일 때, 이에 해당하는 퍼지 변수의 값은 L2 = 0, M2 = 0.75, H2 = 0.25 가 된다(그림 3 참조). 즉, 고급사용자의 요청율이 75% 정도는 중간 수준이고 25% 정도는 높다는 의미이다. 멤버쉽 함수는 삼각형 형태를 활용하였으며, 퍼지 모델의 파라미터 값은 Heuristic을 이용하여 퍼지 함수에 의한 퍼지값이 가장 최적인 포인트를 찾아 사용하였다.

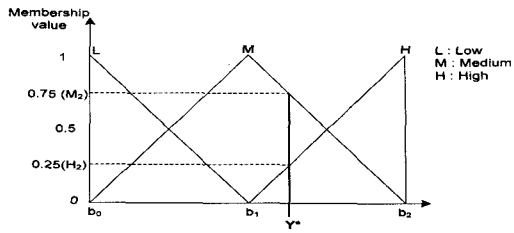

그림 3 사용자 요청율 멤버쉽 함수

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

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

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

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 (6)과 같이 정의되고 그래프로 나타내면 그림 4와 같다(i=1일 때 일반사용자, i=2는 고급

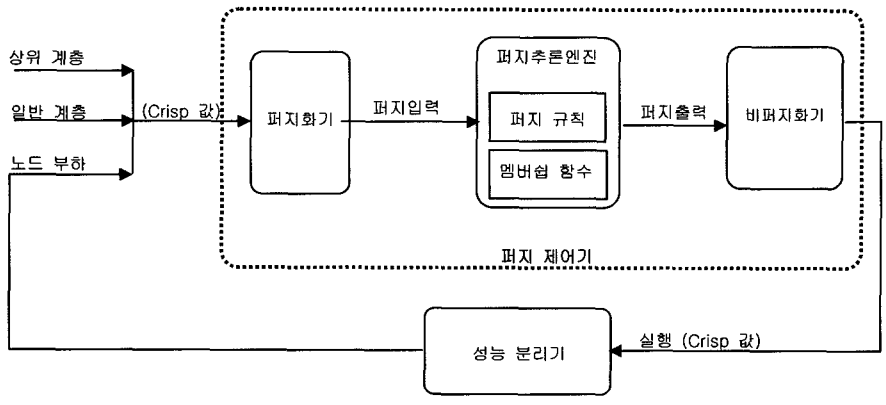

그림 2 퍼지 성능 분할 제어기의 블럭도

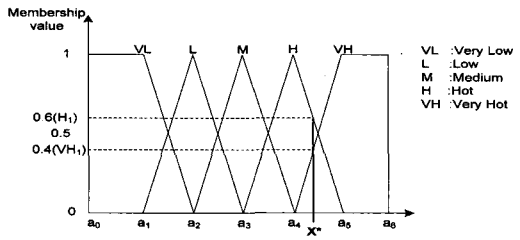

그림 4 서버 노드 부하 멤버쉽 함수

사용자를 의미). 예를 들어, 서버 노드의 부하가 X*일 때, 이에 해당하는 퍼지 변수의 값은 VL1 = 0, L1 = 0, M1 = 0, H1 = 0.6, VH1 = 0.4가 된다(그림 4 참조). 즉, 서버 노드 부하가 60% 정도는 높은 수준이고 40% 정도는 매우 높다는 의미이다.

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

$$L_i = \begin{cases} 0 & when\ x \ge a_{i1} \text{ or } x \ge a_{i3} \\ (x - a_{i1})/(a_{i2} - a_{i1}) & when\ a_{i1} < x < a_{i2} \\ (a_{i3} - x)/(a_{i3} - a_{i2}) & when\ a_{i2} < x < a_{i3} \\ 1 & when\ x = a_{i2} \end{cases} \quad (5)$$

...

$$VH_i = \begin{cases} 1 & when\ x \le a_{i5} \\ (x - a_{i4})/(a_{i5} - a_{i4}) & when\ a_{i4} < x < a_{i5} \\ 0 & when\ x \ge a_{i5} \end{cases} \quad (6)$$

4.2 퍼지 추론 엔진

퍼지 제어기로 입력된 요청율과 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 함수값은 Max-Min 알고리즘을 적용하여 출력한다[10]. 이와 같은 방식에 의거하여, 모든 퍼지 입력 변수의 애매모호함이 결정되면, 표 1에 나타난 규칙에 의해 추론 엔진(Inference Engine)은 최종 추론 결과를 출력한다.

이와 같이 퍼지입력 변수의 단계별 조합에 따라 모두 15개의 추론 규칙이 생성되며, 이들의 집합을 규칙 베이스라고 한다. 추론엔진은 규칙베이스와 앞 절에서 설명한 멤버쉽함수를 결합하여 최종 단일 퍼지 결론(MISO: Multi Input Single Output)을 유도하며, 이 과정을 나타내면 다음과 같다.

1) Input: x is X* and y is Y*
2) 규칙
- Rule 1: If x is X1 And y is Y1 Then z is C1
- Rule 2: If x is X2 And y is Y2 Then z is C2
  ...
- Rule n: If x is Xn And y is Yn Then z is Cn

3) Conclusion: z is C*

표 1 퍼지 추론 규칙

요청율 / 부하량	낮은 경우	중간인 경우	높은 경우
매우 낮은 경우	매우 작게	작게	약간 작게
낮은 경우	작게	약간 작게	변동 없음
중간인 경우	약간 작게	변동 없음	약간 크게
높은 경우	변동 없음	약간 크게	크게
매우 높은 경우	약간 크게	크게	매우 크게

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 나누어진 7단계의 퍼지 함수로 구분된다. 이들의 애매모호함의 정도를 나타내는 멤버쉽 함수는 다음의 식 (7)~식 (9)와 같이 정의되고, 그래프로 나타내면 그림 5와 같다(i=1일 때 일반사용자, i=2는 고급사용자를 의미). 예를 들어, 퍼지 출력 변수가 Z*일 때, 이에 해당하는 퍼지 변수의 값은 NL = 0, NM = 0, NS = 0, AZ = 0.45, PS = 0.55, PM = 0, PL = 0가 된다(그림 5 참조). 즉, 서버 노드 수의 증가/감소 정도를 나타내는 퍼지 출력값이 55% 정도는 서버노드 수의 변동이 없는 수준이고 45% 정도는 노드수를 약간 크게 하는 수준이라는 의미이다.

$$NL_i = \begin{cases} 0 & when\ z \ge c_{i1} \\ (z - c_{i0})/(c_{i1} - c_{i0}) & when\ c_{i0} < z < c_{i1} \\ 1 & when\ z = c_{i0} \end{cases} \quad (7)$$

$$NM_i = \begin{cases} 0 & when\ z = c_{i0} \text{ or } z = c_{i2} \\ (z - c_{i0})/(c_{i1} - c_{i0}) & when\ c_{i0} < z < c_{i1} \\ (z - c_{i1})/(c_{i2} - c_{i1}) & when\ c_{i1} < z < c_{i2} \\ 1 & when\ z = c_{i1} \end{cases} \quad (8)$$

...

$$PL_i = \begin{cases} 0 & when\ z \le c_{i5} \\ (z - c_{i5})/(c_{i6} - c_{i5}) & when\ c_{i5} < z < c_{i6} \\ 1 & when\ c = b_{i6} \end{cases} \quad (9)$$

퍼지 출력값은 비퍼지화 과정을 거쳐 최종 제어값(Crisp)으로 변환하는데, 이 경우 COA(Center of Area) 기법[11]을 적용하여 출력값을 계산하였다(식 (10)). 여기서, Wi는 i 번째 추론 규칙의 퍼지값이고, Bi는 i번째 추론 규칙의 중앙 값이다. 이렇게 함으로써 요청율의 변화에 신속하게 대처할 수 있게 된다.

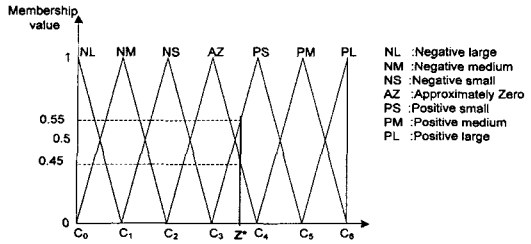

그림 5 퍼지 출력값의 멤버쉽 함수

$$P_i^{COA} = \left\lfloor \frac{\sum_{i=1}^{n} W_i * B_i}{\sum_{i=1}^{n} W_i} \right\rfloor \quad (10)$$

퍼지 제어기의 규칙에 따른 추론 결과를 고려하여, 성능 분리기에 전달할 최종 결과를 결정하게 되는데, 우선권은 항상 고급 사용자에 있다. 고급 사용자가 노드 임대를 요청할 경우 일반사용자 계층을 서비스하는 서버 노드 집합으로부터 노드를 임대 받게 된다.

5. 성능 평가

퍼지기반 웹 서버 클러스터 시스템 성능 분리 기법의 성능 평가를 위해, 응답시간에 대한 비퍼지 기법과의 비교분석을 진행하였다. SLA를 고려하기 위해 95% 응답시간을 사용하였다. 95-percentile 응답시간은 들어온 요청 중 95% 이상은 서비스 제공자와 사용자간의 계약된 시간 안에 응답시간을 만족한다는 것을 의미한다. 웹 서버는 각 사용자 계층으로부터 정적 요청과 동적 요청을 받아들이며, 시뮬레이션에 사용된 웹 서버 클러스터 시스템의 운영 파라미터는 표 2, 3과 같다[12].

사용자 요청율을 의미하는 요청 도착 시간 간격을 지수(Exponential) 분포를 사용하여 표현하였고, 이에 대응하는 사용자의 서비스 요청 사건(Event)은 서비스 시간이 상이한 동적 요청과 정적 요청을 랜덤(Random)한 비율로 발생시킨다. 동적 요청의 평균 서비스 시간 간격은 700 msec., 정적 요청의 평균 서비스 시간 간격은 100 msec.로 하였으며, 발생된 요청은 서버 내 각 자원의 점유를 위해 해당 대기큐에 입력되고, 이미 해당 자원이 서비스를 수행하고 있으면 큐에 대기하게 된다. 요청 패킷의 경로는 요청의 계층별 우선순위에 분리되어 그에 상응하는 서버 집합(Set)이 결정된다. 각 자원의 사용 시간과 대기 시간으로부터 시스템의 응답 시간이 계산된다. 한편, 0~100% 구간의 다양한 동적/정적 구성 비율의 조합으로 실험을 수행하였고, 웹 서버 컴퓨팅 노드 수는 10대, 클라이언트 도착에 따른 Premium 계층의 웹 서버 컴퓨팅 노드 수는 30대로 적용하였으며,

표 2 정적·동적 요청 부하 모델

요청타입 / 파라미터	서비스 시간	구성 비율(%)
Dynamic Requests	700 msec.	0~100
Static Requests	100 msec.	0~100

표 3 시스템 파라미터 (단위:second)

서비스 요청 도착시간 간격	Exp(평균 서비스 시간)
서버 수	10~30 대
계층별 요청 도착 비율 $[class_P, class_N]$	[0.2, 1.0]

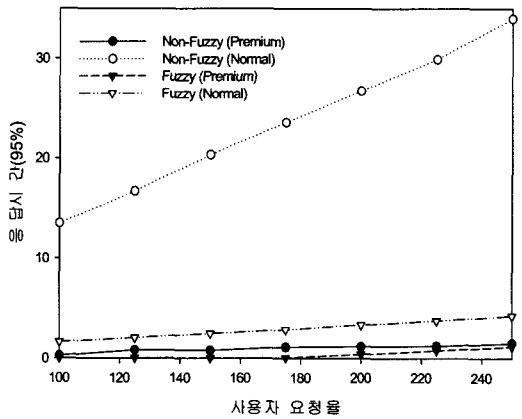

그림 6 사용자 요청율 변화에 따른 응답시간 비교

사용자 계층은 Premium·Normal 2개의 계층으로 나누어 성능을 평가했다. 사용자 요청율은 [100, 300]의 구간으로 적용 하였고, 사용자 요청율에 따른 서비스 응답시간, 거절된 요청 수, 프리미엄 서비스를 위한 서버 수를 측정하였다. 또한 사용자 요청율에 따른 동적/정적 요청의 경우, 평균 서비스 시간을 측정하여 퍼지 및 비퍼지의 성능을 비교하였다.

그림 6에서는 차별화 서비스를 위한 2개 계층을 대상으로, 동적 분할에 따른 퍼지 및 非퍼지 기법간의 응답시간을 사용자의 요청율의 변동에 따라 표시하였다. 퍼지 기반 성능 분리 기법과 비퍼지기법 모두 Premium 계층이 Normal 계층에 대해 응답시간 성능이 우수하게 나타나고 있다. 특히, 퍼기 기반 기법이 비퍼지기법에 비해 Normal 계층에서 월등한 응답시간 성능을 보이고 있는데, 이는 퍼지기법이 사용자 요청율 및 서버 부하량 증가에 따라 서버 노드수를 보다 탄력적으로 조정한 결과 때문이라 판단된다. Premium 계층에서는 퍼지기반 기법이 비퍼지기법에 비해 사용자 요청율이 적은 구간에서는 약간 우수한 성능을 보이고 있으나, 요청율이 커질수록 응답시간 성능이 거의 동일한 수준을 보이고 있다. 결국, 퍼지기법이 Normal 계층 중심으로 비퍼지기

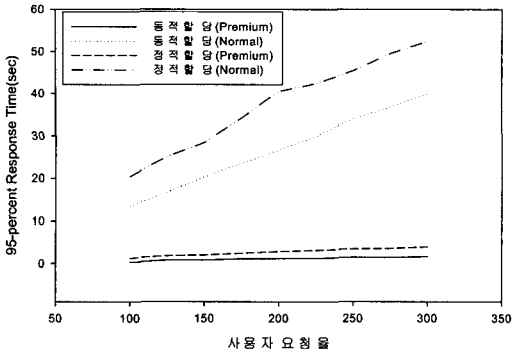

그림 7 동적 할당과 정적 할당에 따른 서버 활용율

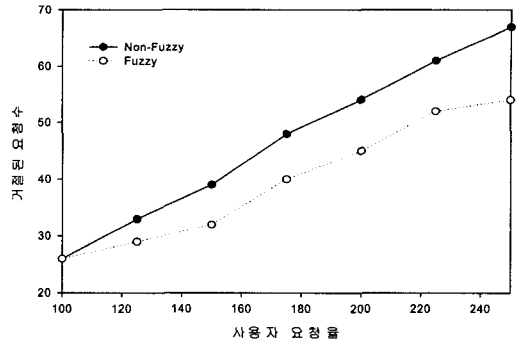

그림 9 요청율에 따른 거절된 요청의 수 비교

법에 비해 우수한 성능을 보이고 있음을 알 수 있으며, Premium 계층에서의 성능 차별화는 퍼지 규칙 및 멤버쉽 분석을 통해 계산될 수 있다.

그림 7은 비퍼지 기법에 대하여 서버의 작업량 분배에 동적 할당과 정적 할당을 적용하였을 때를 95-percentile 응답시간 관점에서 비교 분석한 결과이다. Premium의 경우에는 우선적으로 서비스를 보장해 주므로 큰 차이를 보이지 않지만, Normal 서비스의 경우에는 사용자의 수에 따라 응답시간의 차이가 크게 나타나고 있다. 그림 6과 그림 7의 결과를 종합적으로 비교 분석해 보면, 퍼지기법이 비퍼지 기법에 비하여 Premium 서비스 보다는 Normal 서비스의 경우에 차별화 효과가 더욱 크게 나타난 것을 알 수 있다.

한편, 그림 8에서는 서버 수와 사용자 요청율에 따른 서버들의 활용율 평균에 대한 결과를 보여주고 있다. 서버 수가 증가하면 각 서버의 평균 활용율은 감소하며, 사용자 요청율이 증가하면 각 서버의 활용율은 증가하게 된다. 사용자 요청율이 300에 가까울수록 서버의 평균 활용율은 100%에 근접함으로써, 거의 모든 서버가 가동되고 있음을 알 수 있다.

그림 9에서는 Normal 계층을 대상으로 퍼지·비퍼지 기법에 따른 거절된 요청 수를 클라이언트의 요청율의 변동에 따라 표시하였으며, 이때 Premium 계층은 클라이언트의 요청을 모두 수용하는 것으로 가정하였다. 승인제어를 통해서 서비스가 거절된 요청수 측면에서 퍼지기반 기법이 비퍼지 기법에 비해 상대적으로 적게 나옴을 알 수 있다. 특히, 사용자 요청율이 커질수록 그 차이가 증가함을 알 수 있다.

그림 10은 정적·동적 요청에 따른 응답시간을 퍼지·비퍼지 기법에 따라 표시하였다. 동적 요청과 정적 요청 모두에서 퍼지기법이 비퍼지 기법에 비해 Premium 계층과 Normal 계층 응답시간 성능이 좋게 나왔다. 이는 퍼지기법의 경우, Premium 계층이 과부하 상태일 때 Normal 계층을 서비스하는 서버를 상위 계층에 효과적으로 제공해주기 때문이다. 한편, 정적 요청은 변하지 않는 문서와 같은 형식의 파일이고, 동적 요청은 자주 변하면서 처리하는데 시간이 많이 걸리는 VOD와 같은 동영상 파일이므로 퍼지지법과 비퍼지 기법 모두 대체적으로 정적 요청이 동적 요청에 비해 응답시간 성능이 좋게 나온 것을 알 수 있다.

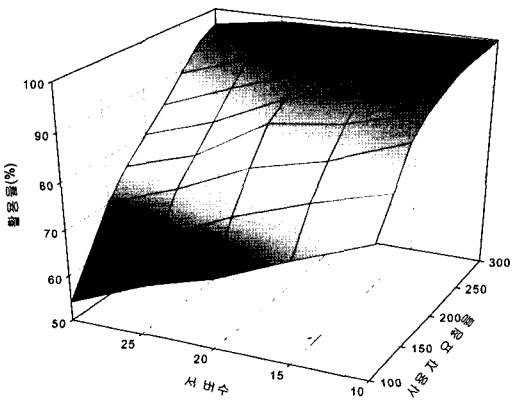

그림 8 서버 수와 사용자 요청율에 따른 서버 활용율

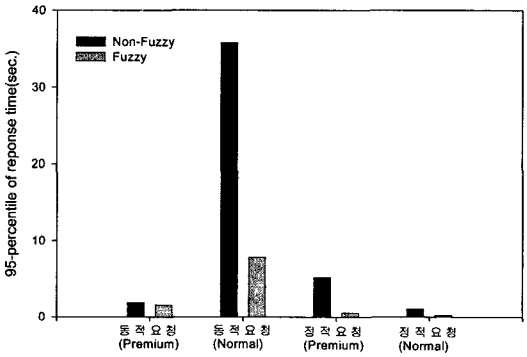

그림 10 퍼지기법 사용여부에 따른 정적·동적 요청의 응답시간 비교

그림 11에서는 Premium 계층을 대상으로 한 퍼지·비퍼지 기법에 따른 서버 수 변화(서버 수 30대일 경우)를 사용자 요청율에 따라 표시하였다. 퍼지·비퍼지 기법 모두 기본적으로 동적분할 기법을 적용하고 있으므로 사용자 요청율이 커질수록 Premium 계층을 서비스하는 서버 수에 변동이 생기게 됨을 알 수 있다. 비퍼지 기법의 경우, Premium 계층의 처리 용량이 부족한 상황에서 Normal 계층을 서비스하는 서버를 강제적으로 빌려오므로 서버 수가 지속적으로 증가한다. 퍼지 기법의 경우, 사용자 요청율과 서버 노드 부하가 동시에 높은 수준일때만 서버 노드를 조정하므로 상대적으로 변동 폭이 작음을 알 수 있다.

그림 12에서는 Normal 계층에 대해 서버 수 및 동적

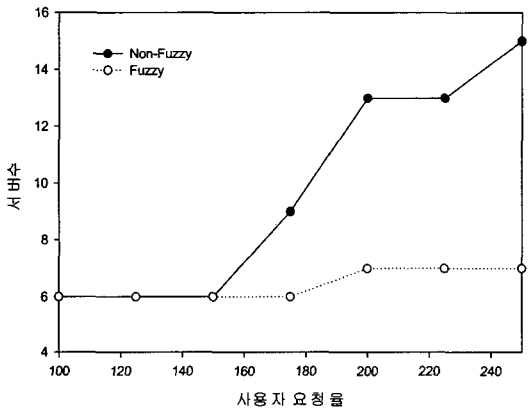

그림 11 요청율에 대한 Premium 계층을 서비스하는 서버 수 변화

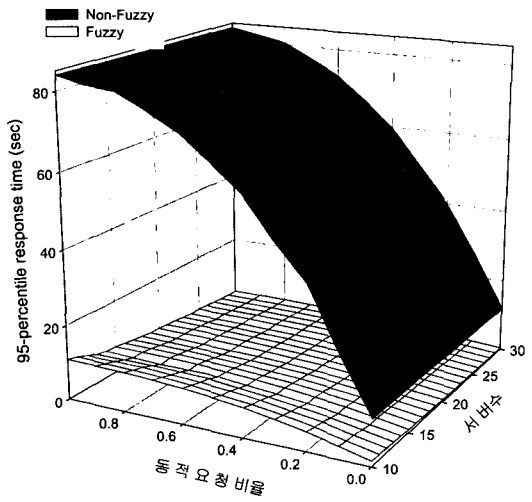

그림 12 서버 수/동적요청 비율 변화에 따른 응답시간 비교

요청 비율 변화에 따른 응답시간을 비교분석 하였다. 기본적으로, 퍼지기법과 비퍼지 기법 모두에서 서버 수가 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, 서버 수 변화보다 동적 요청 비율이 급격히 증가할 때 시스템 가동성능 저하가 예상된다.

상기의 성능평가 실험을 통해, 퍼지기반 성능 분리 기법이 퍼지기법을 사용하지 않은 경우에 비하여 사용자 요청율 및 시스템 환경 변화에 따른 변수에 훨씬 강건하게 대응하고 있음을 확인할 수 있다.

6. 결 론

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

특히, 퍼지 기법을 적용했을 때 Premium 계층보다는 Normal 계층의 요청에 대한 응답 시간 성능이 상대적으로 더 효과적임을 알 수 있다. 즉, Premium 계층 서비스의 경우 요청에 대한 처리를 우선적으로 보장을 해주며 낮은 요청율으로 인해 부하 분산의 효과가 낮아지게 되지만, Normal 계층 서비스 요청의 경우 요청율은 Premium 계층보다 상대적으로 많이 높기 때문에 부하 분산 효과도 커지게 된다. 한편, 정적 요청보다는 동적 요청이 서비스 횟수도 높고 전송할 데이타 양도 많기 때문에 부하 분산의 기회가 많아 퍼지 기법을 적용한 응답 시간 성능이 더 우수함을 알 수 있었다.

향후, 사용자 요청율 및 서버노드 부하량 특성을 보다 정밀하게 반영한 퍼지 멤버쉽 함수 구성을 통해 성능 개선 효과를 확대할 예정이다. 삼각퍼지숫자 형태 이외의 다양한 함수 형태를 참조하여 멤버쉽 함수를 재구성하고자 한다. 또한, 95-percentile of response time이외의 다양한 성능 파라메타에 대해서도 비퍼지기법과 비교분석 연구를 수행할 예정이다. 한편, 웹서버는 과부하 상황에서 우선순위가 높은 사용자의 요청이 거절되는 경우를 발생시킬 수 있다. 이러한 경우 사용자의 우선순

위에 따라 상위 계층의 사용자 요청을 받아들이고 하위 계층의 사용자 요청을 거절하는 승인제어(Admission Control)가 향상된 차별화 서비스를 위해 중요한 요소라 할 수 있다. 따라서, 퍼지기법을 적용한 승인제어 알고리즘을 제안하고 해당 알고리즘에 대한 퍼지 기법과 비퍼지기법의 성능 비교분석을 통해 차별화 서비스를 보다 체계적으로 지원하기 위한 연구를 수행할 예정이다.

참 고 문 헌

[1] A. Cohen, S. Rangarajan, and H. Slye, "On the Performance of TCP Splicing for URL-aware Redirection," In: Proceedings of the USENIX Symposium on Internet Technologies and Systems, pp. 117-125, 1999.

[2] A. Shaout, P. McAuliffe, "Job scheduling using fuzzy load balancing in distributed system," Electronics Letter, Vol.34, No.20, pp. 1983-1985, 1998.

[3] Y. Wei, et al., "Class-Based Latency Assurances for Web Servers," *High Performance Computing* and Communications, pp. 388-394, 2005.

[4] N. Vasiliou, H. Lutfiyya, "Providing a differentiated quality of service in a World Wide Web server," SIGMETRICS Performance Evaluation Review, Vol.28, No.2, pp. 22-28, 2000.

[5] M. Aron, et al., "Scalable Content-aware Request Distribution in Cluster-based Network Servers," Proceedings of the 2000 Annual Usenix Technical Conference, 2000.

[6] E. *Casalicchio and M. Colajanni*, "A *Client-aware* Dispatching Algorithm for Web Clusters Providing Multiple Services," Proceedings of the 10th International World Wide Web Conference, pp. 535-544, 2001.

[7] 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 and Web-Based Information Systems, pp. 212-221, 2001.

[8] M. Andreolini, et al., "A cluster-based web system providing differentiated and guaranteed services," Cluster Computing, Vol.7, No.1, pp. 7-19, 2004.

[9] H. Zhu, H. Tang, and T. Yang, "Demand-driven service differentiation in cluster-based network servers," Proceedings of IEEE Infocom. 2001, pp. 679-688, 2001.

[10] J. Mendel, "Fuzzy logic systems for engineering: A tutorial," Proceedings of IEEE, Vol.83, pp. 345-377, Mar. 1995.

[11] T. Chu, K. Huang, and T. Chang, "COA defuzzification method for evaluating Cpk under fuzzy environments," Journal of Discrete Mathematical Sciences ε Cryptography, Vol.7, No.3, pp. 271-280, 2004.

[12] V. Cardellini, "Web switch support for differentiated services," SIGMETRICS Performance Evaluation Review, Vol.29, No.2, pp. 14-19, 2001.

박 범 주

1989년 서울대학교 조선공학과(공학사) 1992년 포항공과대학교 산업공학과(공학석사). 1992년~1995년 삼성종합기술원 그룹CAE센타 주임연구원. 1995년~1998년 삼성전자 소그룹 전략기획총괄 첨단기술센타 과장. 1998년~현재 삼성전자 첨단기술연수소 부장. 2002년~현재 아주대학교 정보통신전문대학원 박사과정. 관심분야는 결함허용, 성능분석, 클러스터컴퓨팅, 소프트웨어재활, Dependable Systems

박 기 진

1989년 한양대학교 산업공학과(공학사) 1991년 POSTECH 산업공학과(공학석사). 2001년 아주대학교 컴퓨터공학과(공학박사). 1991년~1997년 삼성종합기술원, 삼성종합기술원/삼성전자 선임연구원. 2001년~2002년 한국전자통신연구원 네트워크장비시험센터 선임연구원. 2002년~2004년 안양대학교 컴퓨터학과 전임강사. 2004년~현재 아주대학교 산업정보시스템공학부 조교수/부교수. 관심분야는 In-vehicle Network, Dependable Embedded Computing, Ubiquitous Computing & Networks

강 명 구

2005년 한국과학기술원(KAIST) 산업공학과(공학사). 2007년 아주대학교 산업공학과(공학석사). 2007년~현재 (주)TmaxSoft 전임연구원. 관심분야는 Dependable System & Networks, Business Intelligence 등

김 성 수

1982년 서강대학교 전자공학과(공학사) 1984년 서강대학교 전자공학과(공학석사) 1995년 Texas A&M University, 전산학과(공학박사). 1983년~1996년 삼성전자/삼성종합기술원 연구원/주임연구원/선임연구원/수석연구원. 2002년~2003년 Texas A&M University 교환교수. 1996년~현재 아주대학교 조교수/부교수/정교수. 2006년~현재 아주대학교 정보통신전문대학원 원장 겸 정보통신대학 정보및컴퓨터공학부 학부장. 관심분야는 Dependable Systems & Networks, System Performance Evaluation, Ubiquitous Computing & Networks 등