퍼지 기반 웹서버 성능 분할 기법

박범주*, 박기진**, 김성수***
*삼성전자 첨단기술연수소
*아주대학교 산업정보시스템공학부
***아주대학교 정보통신전문대학원
*e-mail : bumjoo@samsung.com

A Fuzzy Based Web Server Performance Isolation Method

Bumjoo Park*, Kiejin Park**, Sungsoo Kim***
*Advanced Technology Training Institute, Samsung Electronics
**Division of Industrial & Information Systems Engineering, Ajou University
***Graduate School of Information and Communication, Ajou University

요 약

본 논문에서는 차별화 서비스(Differentiated Service)를 구현하기 위하여 퍼지 이론을 적용한 웹 서버 컴퓨팅 노드들의 동적인 성능 분리(Performance Isolation) 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 요청률의 급격한 변화에 대응하여, 계층별 요청을 처리하는 담당 노드의 수를 동적으로 조절할 수 있게 하였다. 성능분석을 통해 제안된 퍼지 기반 성능 분리 방식의 서비스 응답시간이 퍼지기법을 사용하지 않은 일반적인 성능 분리에 비해 개선되는 것을 확인하였다.

1. 서론

차별화 서비스(Differentiated Service)를 제공하는 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) [1] 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. 성능 분리 기법중에서 정적 분할(Static Partition)은 계층별 사용자의 요청률에 따라 고정된 수의 서버 노드를 최초에 한 번만 할당하는 방식으로 계층별 사용자 요청률이 자주 변하는 웹 환경에는 적당하지 않다[2]. 동적 분할(Dynamic Partition) 기법은 사용자의 계층별 요청 수준 혹은 필요에 따라서 동적으로 서버 노드를 할당하며, 사용자 요청률의 변화에 따라 CPU 와 디스크 I/O 용량을 할당함으로써 차별화된 서비스를 제공하기 때문에 멀티미디어 혹은 동적 콘텐트(Dynamic Content)처럼 CPU 와 디스크 I/O 처리가 많은 상황에 적합하다[3,4]. 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 컨텐

이 논문은 2005 년도 1 학기 정착연구비 지원에 의하여 연구되었음.

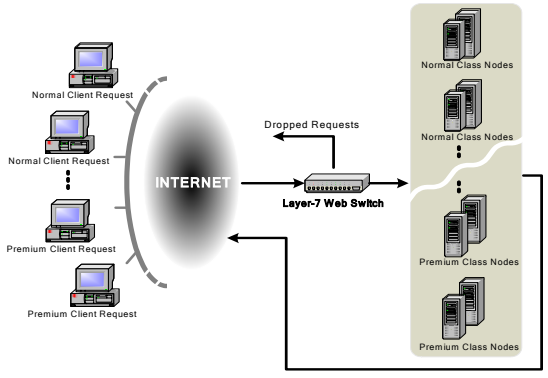
츠(Static Content)만을 처리하는 경우일 때는, 노드에 부하를 많이 주지는 못한다. 이처럼 웹 서버 부하량 판단 혹은 사용자 요청률의 변화시에 발생하는 애매모호한 상황(Fuzziness)을 표현하기 위해, 퍼지제어 알고리즘에 기초한 부하분배 메커니즘을 고려할 필요성이 있다[5].

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.

2. 시스템 모델

본 논문에서 고려하고 있는 클러스터 기반 웹 서버 컴퓨팅 노드들은 두 계층 이상의 사용자를 가정한다. 두 계층의 사용자(NC: Normal Class User, PC: Premium Class User)를 가정할 경우 서비스 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 사용자의 요청 내용에 대하여 상세한 정보를 고려한 처리가 가능하다.

즉 정적 서비스 또는 무료 사용자를 일반 사용자로, 동적 서비스나 유료 사용자를 고급 사용자로 구별하며, 이들로부터의 요청은 Layer-7 스위치를 통해 해당 서비스를 담당하여 처리하는 노드에 도착된 후, 그 노드에서 직접 결과를 전달받는 One-way 웹 서버 구조를 채택하였다.



(그림 1) Layer-7 스위치 기반 One-way 웹 서버 구조

3. 퍼지 기반 웹 서버 성능 분할 기법

일반적인 동적 분할 기법은 상위 계층의 사용자의 SLA(Service Level Agreement)를 만족하는 서비스를 제공하기 위해, 계층별 부하량에 따라 서버 노드를 동적으로 분할하는 기법으로, 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 하고있다. 동적 분할 개념은 부하 변화에 맞게 대처할 수 있어 여러 사용 계층자들에게 효율적이면서도 질 높은 서비스를 제공할 수 있다.

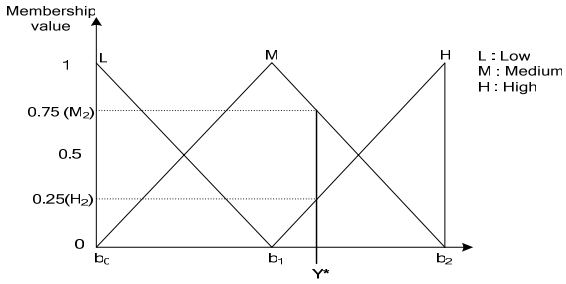
퍼지 기반 웹 서버 성능 분리 알고리즘을 구현하기 위해 설계하고자 하는 퍼지 제어기의 퍼지 변수(Fuzzy Variable)를 현재 재구성(임대 및 대여) 가능한 웹 서버의 부하량 및 사용자 계층별 요청률(Arrival Rate)로 설정하였다. 웹 서비스에서는 사용자가 최초 접속하여 원하는 일을 모두 마치고 나갈 때까지 여러 단계의 연속된 작업(Session)을 거치기 때문에 단순히 접속자 수만을 기준으로 웹 서버의 부하량을 판단하기는 어려우며, 각 세션을 기준으로 요청 주기, 사용자 요청 검토 시간(Think Time), 요청한 내용의 종류(Static/Dynamic), 및 대기 작업의 수 등을 통합적으로 고려하여야 한다. 하지만 이와 같은 사용자와 관련된 이벤트들의 정량적인 값에 대한 확률 분포를 구하기가 용이하지 않다. 또한 사용자 계층별 요청률이 특정 시간대에 급격하게 증가하거나 감소할 수가 있기 때문에, 어느 특정값을 기준으로 요청률이 “높다” “낮다”를 판단하기 보다는 “매우 증가” “점진적 증가” “급격 감소”등의 상태 정보를 반영할 수 있어야 한다.

각 계층별 사용자 요청률은 낮은 경우(L: Low), 중간인 경우(M: Medium), 높은 경우(H: High) 등 3 단계의 퍼지 변수로 구분되며, 이들의 애매모호함의 정도를 나타내는 멤버쉽 함수는 다음의 식(1) ~ 식(3)과 같이 정의된다(i=1 일 때 일반사용자, i =2 는 고급사용자를 의미)

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

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

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



(그림 2) 사용자 요청률 멤버쉽 함수

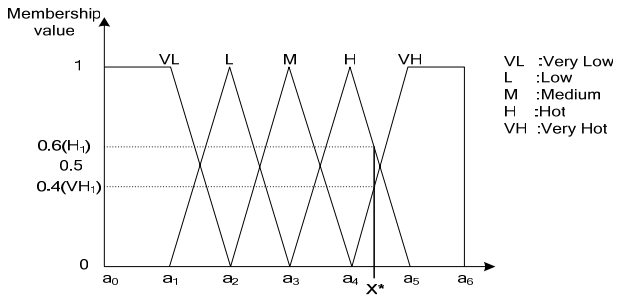
두 사용자 계층을 서비스 하는 서버 노드의 부하는

매우 낮은 경우(VL: Very Low), 낮은 경우(L: Low), 중간인 경우(Medium), 높은 경우(L: High), 매우 높은 경우(VL: Very High) 등 5 단계의 퍼지 변수로 구분된다. 이들의 애매모호함의 정도를 나타내는 멤버쉽 함수는 다음의 식(4) ~ 식(5)과 같이 정의된다(i=1 일 때 일반 사용자, i =2 는 고급사용자를 의미)

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

…

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



(그림 3) 서버 노드 부하 멤버쉽 함수

퍼지 제어기로 입력된 요청률과 부하량(Crisp)은 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 함수값(Max-Min 알고리즘)을 결과로 출력한다. 이와 같은 방식에 의거하여, 모든 퍼지 입력 변수의 애매모호함이 결정되면, 표 1 에 나타난 규칙에 의해 추론 엔진은 최종 퍼지 추론 결과를 출력한다.

표 1 퍼지 추론 규칙

요청률 / 부하량	Low	Medium	High
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

만약 서버의 부하가 매우 낮고, 요청률이 높은 경우에 현 서버 노드를 약간만 감소하라는 의미의 규칙과 서버의 부하가 매우 크고, 요청률이 높을 경우 서버 노드를 크게 증가하라는 규칙은 각각 아래와 같은 If-Then 구조로 표시될 수 있다.

Rule 1: If *Sever Load is Very Low* ***And*** *Arrival Rate is High* **Then** *Negative Small.*

Rule 2: If *Sever Load Very High* ***And*** *Arrival Rate High* **Then** *Positive Large.*

이와 같이 퍼지입력 변수의 단계별 조합에 따라 모두 15 개의 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 유도해 낼 수 있다. 이 과정을 나타내면 다음과 같다.

1) Input: x is X^* and y is Y^*

2) 규칙
Rule 1: If x is X_1 And y is Y_1 Then z is C_1
Rule 2: If x is X_2 And y is Y_2 Then z is C_2
Rule n: If x is X_n And y is Y_n Then z is C_n

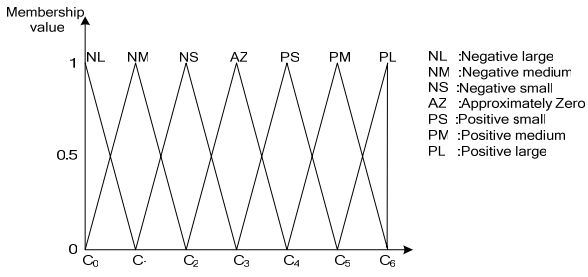
3) Conclusion: z is C^*

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 7 단계의 퍼지 함수로 구분된다. 이들의 애매모호함의 정도를 나타내는 멤버쉽 함수는 다음의 식(6) ~ 식(7)과 같이 정의된다(i=1 일 때 일반사용자, i =2 는 고급사용자를 의미)

$$NL_i = \begin{cases} 0 & when \quad z \ge c_{i1} \\ (z - c_{i0})/(c_{i1} - c_{i0}) & when \quad c_{i0} < z < c_{i1} \\ 1 & when \quad z = c_{i0} \end{cases} \tag{6}$$

…

$$PV_i = \begin{cases} 0 & when \quad z \le c_{i5} \\ (z - c_{i5})/(c_{i6} - c_{i5}) & when \quad c_{i5} < z < c_{i6} \\ 1 & when \quad c = b_{i6} \end{cases} \tag{7}$$



(그림 4) 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(식 8 참

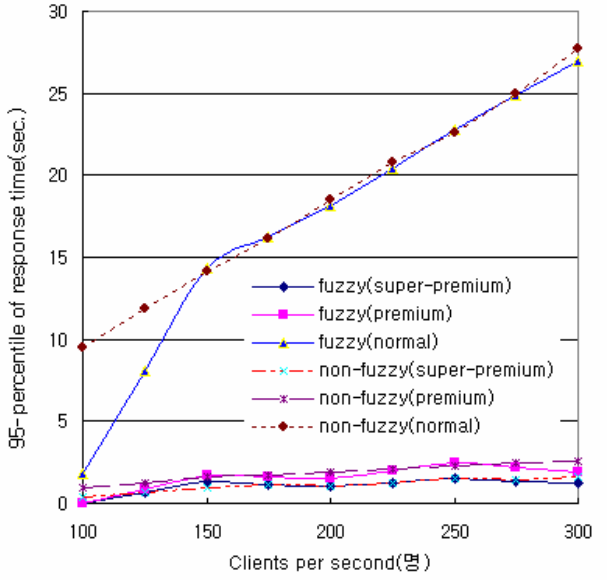
조). 여기서 Wi 는 i 번째 추론 규칙의 퍼지값이고, Bi 는 i 번째 추론 규칙의 중앙 값이다. 이렇게 함으로써 도착률의 변화에 신속하게 대처할 수 있게 된다.

$$P_i^{COA} = \left\lfloor \frac{\sum_{i=1}^{n} W_i * B_i}{\sum_{i=1}^{n} W_i} \right\rfloor \qquad (8)$$

퍼지 제어기의 규칙에 따른 추론 결과를 고려하여, 성능 분리기에 전달할 최종 결과를 결정하게 되는데, 우선권은 항상 고급 사용자에 있다. 고급 사용자 계층이 노드 임대를 요청할 경우 일반사용자 계층을 서비스하는 서버 노드 집합으로부터 노드를 임대 받는다. 고급 사용자의 추론 결과가 Negative 일 경우에만 일반사용자 노드는 고급 사용자 노드로부터 서버를 대여받을 수 있다.

4. 성능 분석

실제적인 웹 서버 클러스터 시스템의 성능 분할을 원활히 시뮬레이션하기 위해, 사용자의 요청률에 따른 응답시간 및 승인 거절된 요청 수에 관하여 분석하였으며, 95-percentile of response time 은 들어온 요청 중 95% 이상은 서비스 제공자와 고객간의 계약된 시간 안에 응답시간을 만족한다는 것을 의미한다. 웹 서버는 각 클라이언트 계층으로부터 정적 요청과 동적 요청을 받아들인다. 기본적으로 클라이언트 요청은 동적 요청 20%, 정적 요청 80% 비율로 구성되어 있으며, 동적 요청의 서비스 시간은 700 msec., 정적 요청의 서비스 시간은 100 msec.이다. 한편, 클라이언트 계층을 Super-premium · premium · normal 3 개의 수준으로 나누어 성능을 분석했다.



(그림 5) 퍼지 및 非퍼지 기법간의 응답시간 비교

그림 5 에서는 차별화 서비스를 위한 3 개 계층을 대상으로, 동적 분할에 따른 퍼지 및 非퍼지 기법간의 응답 시간을 사용자의 요청률의 변동에 따라 표시하였다. Super-premium · premium 계층의 경우 非퍼지기법의 경우 요청률 증가에 따라 응답시간이 완만하게 상승하는 반면, 퍼지기법의 경우 일정 수준을 유지하는 결과를 보여주고 있다. 이는 퍼지기법이 사용자 요청률 및 서버 부하량 증가에 따라 서버 노드수를 보다 탄력적으로 조정한 결과 때문이라 판단된다. 그러나, normal 계층의 경우 일정 수준 이하(150 명)의 사용자 요청률에서는 퍼지기법이 非퍼지기법에 비하여 응답시간이 작게 되지만, 요청률이 커질수록 유의할만한 차이를 보여주기 못함을 알 수 있다. 이러한 현상은 상위계층 중심의 성능 분리가 이루어지게 때문이라 할 수 있다.

5. 결론

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 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 상위 계층을 중심으로 非퍼지기법에 비해 개선되는 것을 확인할 수 있었다. 향후, 보다 정교한 퍼지 멤버쉽 함수 구성을 통해 응답시간 이외에 다양한 성능 파라메타에 대해서도 非퍼지기법과 비교분석할 계획이며, 차별화 서비스를 보다 완벽하게 지원하기 위하여 퍼지기법을 승인제어에 적용하는 기법을 연구할 예정이다.

참고문헌

[1] M. Aron, P. Druschel, and W. Zwaenepoel. “Cluster reserves: A mechanism forresource management in cluster-based network servers.” In Proc. of ACM Sigmetrics 2000, pp. 90-101, 2000.

[2] Valeria Cardellini, Emiliano Casalicchio, Michele Colajanni, Marco Mambelli: “Web switch support for differentiated services.” SIGMETRICS Performance Evaluation Review 29(2): pp. 14-19, 2001.

[3] H. Zhu, H. Tang, and T. Yang. “Demand-driven service differentiation in cluster based network servers.” In Proc. of IEEE Infocom 2001, pp. 679-688, 2001.

[4] M. Andreolini, E. Casalicchio, M. Colajanni and M. Mambelli, “A cluster-based web system providing differentiated and guaranteed services,” Cluster Computing, 7(1): pp. 7-19, 2004.

[5] A. Shaout, P. McAuliffe, “Job scheduling using fuzzy load balancing in distributed system,” Electronics Letter, 34(20): pp. 1983-1985, 1998.