한국퍼지 및 지능 시스템학회 논문지 1997, Vol. 7, No. 5. (1997) 8~13

다중 서버 환경에서의 퍼지 개념을 이용한 작업 할당 기법

A Job Scheduling Method using Fuzzy Concepts in Multi-Server Environment

정연돈 · 김종수 · 이지연 · 오석균 · 이광형 · 이윤준 · 김명호

Yon Dohn Chung, Jong Soo Kim, Ji Yeon Lee, Seok Kyun Oh, Kwang Hyung Lee, Yoon Joon Lee and Myoung Ho Kim

한국과학기술원 전산학과

요 약

다중 서버 환경이란 어떤 작업이 처리될 수 있는 서버가 다수 존재하는 환경을 말한다. 이러한 환경에서는 사용자의 요구를 처리하는 서버를 결정하는데 있어 시스템의 전체 성능을 극대화 시키고 요구들의 응답 시간을 최소화 하여야 한다. 이 과정에서 기존에는 서버 부하량만을 기준하여 최소 부하를 지닌 서버를 선정하고 있다. 본 논문에서는 서버의 성능 정도와 부하 정도 그리고 서비스 수행 시간의 크기를 퍼지화하고 전문가 지식을 사용하는 새로운 서버 선정 방법을 제시한다. 퍼지 기법을 사용함으로써 기존 방법에 비하여 우수한 성능을 보임을 실험을 통해 보인다.

ABSTRACT

In multi-server environment there are many servers which are able to process job requests. So we have to design a mechanism that selects appropriate servers for processing each job request while maximizing server throughput and minimizing average response time of requests. Conventional methods adopt the load of each server as criteria of server selection, that is, they select a server whose load is not bigger than the others. In this work we propose an approach that uses the degree of server performance, server load and the estimated service time of requested job as guidelines of server selection. We incorporate fuzzification techniques and expert knowledge in this approach. Comparing the performances of our approach to that of conventional one, experiments show that the proposed approach provides better performances.

1. 서 론

다중 서버 환경이란 분산 컴퓨팅 환경이나 다중 처리기 컴퓨팅 환경에서 특정한 작업들을 수행할 수 있는 컴퓨터 호스트나 프로세스가 여러 개 존재하는 환경을 말한다[7]. 이러한 환경에서 하나의 작업이 요구되었을 때 어떤 호스트 혹은 어떤 서버 프로세스에게 수행을 하도록 할 것인지를 결정하는 것을 작업 할당의 문제라고 한다.

일반적으로 여러 개의 서버 중에서 요청된 작업이 할당될 하나의 서버를 선택할 경우, 기존의 기법들은 서버들 중에서 가장 부하가 작은 것을 선정한다. 즉 그림 1과 같이 3개의 서버가 존재하고 그들의 부하량이 각각 1470, 1450, 1400일 경우, 작업 할당기(scheduler)는 이들 중에서 가장 작은 부하를 지닌 3번째 서버에게 작업을 할당하게 된다. 서버가 지니는 부하량은 일반적으로 서버의 대기 큐내에 존재하는 작업의 수 혹은 작업량을 측정하여 사용한다.

본 연구에서는 기존의 작업 할당 방식에 있어 다음의 변화를 시도하고자 한다. 실제로 클라이언트/서버의 작업 환경에서 각 서버의 작업 처리 능력은 시간이나 환경의 변화에 따라 달라지게 되며, 경우에 따라서는 기계 자체의 처리 능력이 서로 달라질 수 있게 된다. 따라서 동적으로 변화하는 각 서버들의 작업 처

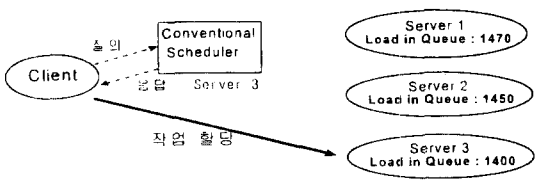

그림 1. 다중 서버에서의 기존의 작업 할당 방법.

리 능력을 작업 할당시에 고려하고자 하는 방법이다. 또한 사용자로 부터 요구된 작업의 예상 수행 시간 정보를 할당기가 사용할 수 있도록 한다. 이는 요구된 작업의 예상 수행 시간에 따라 작업 할당 서버를 결정하는데 도움을 주기 위함이다. 또한 이러한 두 가지 방식을 사용함에 있어 서버의 부하량과 작업 처리 능력 그리고 요구 작업의 수행 시간을 퍼지화하고 전문가의 지식을 사용하여 작업 할당 문제를 해결하고자 한다.

논문의 구성은 다음과 같다. 먼저 2장에서 본 논문이 배경으로 하는 다중 서버 환경과 작업 할당의 문제에 대하여 기술하고, 3장에서는 본 논문에서 제안하고자 하는 퍼지 작업 할당 기법에 대하여 설명한다. 제안된 방법에 대한 성능 평가 실험 및 결과 분석을 4장에서 제시하고 마지막으로 5장에서 결론을 맺는다.

2. 다중 서버 컴퓨팅 환경에서의 작업 할당 문제

클라이언트/서버 시스템은 하나의 일을 클라이언트와 서버라는 두개의 프로세스로 나누어 작업함으로써 작업의 효율을 높이고, 그 결과로 하나의 프로그램에서 실행되어 왔던 전통적인 컴퓨터 프로그램의 단점을 보완해 주는 컴퓨팅 구조이다. 특히 최근에는 클라이언트/서버 컴퓨팅을 위한 분산 소프트웨어로서 미들웨어라고 정의되는 시스템 소프트웨어에 대한 관심도 늘고 있다.

미들웨어의 종류로는 데이타베이스, 트랜잭션 RPC, 그룹웨어, 객체 지향 미들웨어등이 있다[3,4]. 이들 미들웨어 중에서 트랜잭션 미들웨어인 Transaction Processing Monitor(TP Monitor)에 대하여 살펴보자. TP 모니터는 큰 규모의 항공 예약 시스템, 은행 업무, 증권 업무 같은 On Line Transaction Processing(OLTP) 시스템에 적용되어 다수의 사용자가 대용량의 데이타베이스를 공유하면서 관련 업무를 실시간으로 신속하게 처리할 수 있도록 한다[4,5]. 이러한 OLTP 시스템에서는 사용자의 작업 요구들을 원활히 수행시키기 위해 대부분 다수의 서버를 생성하여 사용하는 방식을 채택하고 있다. 본 논문은 이와 같이 사용자들의 작업 요구를 처리하는 서버가 둘 이상 존재하는 다중 서버 환경에서 작업들의 빠른 처리와 서버들 사이의 균형된 부하 분산을 위한 작업 할당기에 관하여 다루고자 한다.

먼저 기존의 방법으로 Unix 환경의 대표적인 TP 모니터의 부하 분산(load balancing) 방법에 대하여 알아보자. TP 모니터에서는 기본적으로 최소의 부하를 갖는 응용 서버에게 클라이언트의 서비스 요구를 할당해 주는 방식을 사용한다. 각 응용 서버의 부하 척도로는 각 서비스 고유의 가중치(weight)와 우선 순위, 메세지 큐잉의 크기 등이 사용된다. 그리고 클라이언트가 서비스를 요청할 때 할당기는 각 응용 서버 중에서 최소 합을 갖는 서버를 할당해 준다[6].

그런데 기존의 TP 모니터에서 사용하는 부하 척도를 살펴보면 서비스의 가중치가 일정한 값으로 항상 고정되어 있다[5]. 따라서 시스템의 일 처리 능력 변화에 대한 반영의 여지가 없다. 그러나 시스템의 부하가 높을 때 어떤 작업의 가중치와 시스템 부하가 낮을 때 그 작업의 가중치 정도는 다를 것이다. 우선 순위 메카니즘 또한 그 정도가 미리 고정된다. 따라서 미리 정의된 우선 순위에 따라 할당기는 그 값을 참조한다. 기존의 할당기는 단지 각 응용 서버가 갖는 척도들의 합에 의존해 부하 분산이 이루어져 전문가의 지식을 반영할 방법을 제공하고 있지 못하다. 따라서 본 연구에서는 퍼지 할당기의 설계를 통하여 기존 TP 모니터 시스템의 부하 분산 알고리즘을 개선하고자 한다.

본 논문에서 기반하는 다중 서버 시스템 환경은 다음과 같다. 첫째 사용자가 요구하는 모든 작업들은 모든 서버에 의하여 수행 가능하다. 즉 어떤 작업들은 특정 서버만이 처리할 수 있는 제약된 환경이 아니다. 둘째, 각 서버의 작업 처리 능력은 시간에 따라 변화하며 이는 측정 가능하다. 실제 분산(distributed) 다중 서버 시스템이나 다중 처리기(multiprocessor) 시스템에서 각 서버의 컴퓨팅 능력은 다양한 요인으로 인하여 시간에 따라 달라질 수 있다. 세째, 사용자가 요구하는 작업들의 예상 수행 시간을 예측할 수 있다.

3. 퍼지 작업 할당

본 연구에서 퍼지 작업 할당을 위하여 퍼지 개념을 사용한 목적은 요구된 작업의 예상 수행 시간과 각 서버의 부하 정도에 따라 효율적으로 할당 서버를 결정하기 위함이다. 이는 일반적으로 전문가의 지식이 언어적 용어를 기반으로 표현되기 때문이다. 가령 "부하가 큰 서버"와 같은 표현을 보면 그 예를 알 수 있다. 기존의 방법은 부하가 얼마인지를 측정하여 그 측정된 결과에 따라 서버 할당이 이루어 졌지만 퍼지 개념을 사용하면 상례의 표현을 퍼지 숫자로 나타낼 수 있게 된다.

3.1 시스템 구성 및 퍼지화

지금부터 본 연구에서 제안하는 퍼지 개념을 사용

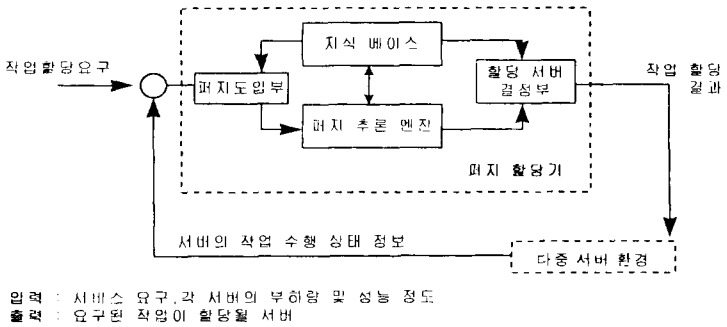

그림 2. 퍼지 작업 할당기의 시스템 구성

하는 다중 서버 환경에서의 작업 할당 방법에 대하여 알아보기로 한다. 그림 2는 퍼지 개념을 사용하여 본 논문에서 설계한 작업 할당기(퍼지 할당기)의 구성을 나타내고 있다. 사용자들로부터 작업 할당 요구가 입력으로 들어오면 현재 수행중인 각 서버들의 수행 상태 정보들과 지식 베이스에 나타나있는 퍼지 규칙들을 사용하여 퍼지 추론(fuzzy inference)을 하게 된다. 추론의 결과를 비퍼지화(defuzzification)하여 작업 할당이 이루어질 서버를 결정하게 된다.

먼저 작업 할당기에서 사용하는 변수들과 그들의 퍼지화 방식에 대하여 알아보자. 작업 할당기에서 사용하는 입력 변수들은 사용자로 부터 요구된 작업과 현재 각 서버들의 부하 정도 및 처리 능력이다. 사용자의 작업 요구는 예상되는 수행시간을 퍼지화하여 사용하게 된다. 즉, 수행시간이 긴 작업과 짧은 작업이라는 두 가지의 기준에 따라 그림 3(a)와 같이 퍼지화하여 표현된다. 그리고 각 서버들의 부하 정도는 전체 서버들의 부하량에 대한 각 서버별 부하량의 비율

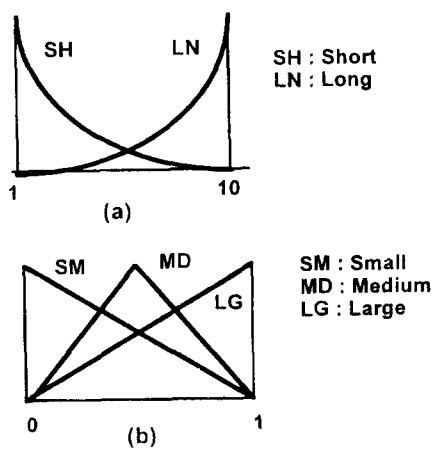

그림 3. (a) 사용자 요구의 예상 수행 시간(EST)과 (b) 서버별 부하 정도(SL)에 대한 퍼지화

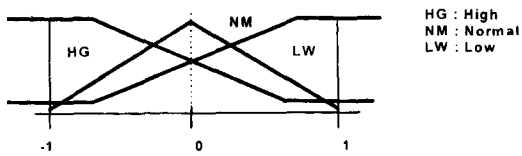

그림 4. 서버의 처리 능력(성능 정도 : SP_i)의 퍼지화

을 측정하여 그 값에 따라 그림 3(b)와 같은 퍼지화 함수를 사용하여 나타낸다. 마지막으로 각 서버의 처리 능력에 대한 퍼지화는 어떤 서버가 처리한 사용자 요구의 처리 시간과 수행 예상 시간과의 차이를 계산하여 사용한다(SPi=작업 A의 예상 수행시간-작업 A의 서버 i에서의 실제 수행시간/작업 A의 예상 수행시간). 서버의 작업 처리 능력(성능 정도)에 대한 퍼지화 함수는 그림 4에 나타나 있다.

3.2 작업 할당을 위한 퍼지 규칙

본 연구에서는 퍼지 할당기를 위한 제어 규칙의 입력 변수로 작업별 예상 수행 시간(estimated service time : EST), 각 서버의 현재 성능(server performance : SP), 그리고 각 서버의 부하 정도-현재 큐에 할당된 일의 양(server load : SL)을 사용한다. 퍼지화를 위해 사용된 언어적 변수들은 표 1과 같이 정하였다.

퍼지 할당기의 출력 변수는 일반적인 퍼지 제어기가 해당 시스템을 위해 어떤 제어 값을 출력으로 하는 것과 달리 작업을 할당할 서버를 선택하여 출력한

표 1. 퍼지화를 위해 사용된 변수

작업 요구의 예상 수행 시간	SH : short, LN : long
각 서버의 현재 큐에 할당된 일의 양 (각 서버의 부하 정도)	SM : small, MD : medium, LG : large
각 서버의 현재 성능 (작업 처리 능력)	HG : high, NM : normal, LW : low

다. 제안된 퍼지 할당기를 위한 퍼지 규칙은 다음과 같다.

1. If EST is SH , SP is LW and SL is SM, then Si
2. If EST is SH , SP is LW and SL is MD, then Si
3. If EST is SH , SP is NM and SL is SM, then Si
4. If EST is LN , SP is NM and SL is SM, then Si
5. If EST is LN , SP is HG and SL is SM, then Si
6. If EST is LN , SP is HG and SL is MD, then Si

위에 기술된 6개의 퍼지 규칙은 시스템을 구성하고 있는 서버 각각에 대해 실행되므로 규칙의 결론부에 i 번째 서버임을 나타내는 Si라는 기호를 사용하였다. 따라서 시스템에 존재하는 각 서버에 대해 위 규칙들이 기술되어야 한다.

제안된 퍼지 규칙에 사용된 전문가의 지식은 현재 성능이 우수한 서버일수록 예상 수행 시간이 긴 서비스를 할당하고, 현재 성능 저하되고 있는 서버 쪽은 예상 수행 시간이 짧은 서비스를 할당하자는 것이다. 이때 각 서버의 큐에 쌓여 있는 작업들의 개수와 요청 서비스들의 예상 수행 시간의 누적치를 고려해야 한다. 만일 각 서버의 부하량을 고려하지 않고 성능만으로 부하 분산을 하려 하면 현재 성능이 우수한 서버에 예상 수행 시간이 긴 서비스만 무한히 큐에 쌓이게 되므로 이 서버가 점차 성능이 떨어지게 될 경우 큐에 쌓인 서비스를 감당할 수 없는 문제가 발생한다. 따라서 기존의 부하 분산 기법과 마찬가지로 각 서버에 대한 부하량을 고려하는 부분을 퍼지 규칙에 함께 넣어 주었다. 위의 규칙에 대하여 설명하면 규칙 6 경우, 현재 서버 i의 작업 처리 성능이 좋은 상태이고 부하량이 작거나 중간일 경우에 예상 수행 시간이 긴 사용자 작업을 할당한다는 내용이다. 나머지 규칙들 역시 같은 맥락을 지니고 있다.

정의한 규칙들의 추론을 위하여 Mamdani의 MIN 연산을 사용하였고, 결과에 대한 비퍼지화(defuzzification) 방법으로는 최대값(maximum) 방법을 사용하였다[1].

4. 성능 평가

본 장에서는 본 논문에서 제안한 퍼지 작업 할당기의 성능 평가를 위한 실험 및 결과를 제시한다. 먼저, 성능 평가를 위한 실험 환경 및 내용에 대하여 기술하고, 실험을 통해 얻어진 결과 및 그에 대한 분석을 제시한다.

4.1 실험 환경

일반적으로 다중 서버 환경의 작업 할당기의 성능은 전체적인 작업 처리량(throughput)과 작업의 평균

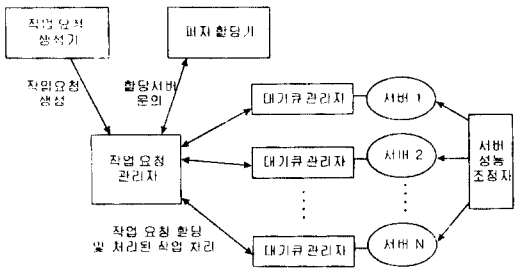

그림 5. 퍼지 할당기의 성능 평가를 위한 실험 환경

응답 시간(average response time)으로 나타내어 진다. 작업 처리량은 단위 시간 동안 전체 서버에서 처리된 작업 요청의 합을 의미하며, 평균 응답 시간은 각 작업 요청 별로 이들이 요청된 시간부터 처리되어 결과가 반환될 때까지의 응답 시간을 평균한 값이다.

본 논문에서는 제안된 퍼지 할당기의 성능을 평가하기 위하여 다음과 같은 실험을 수행하였다. 실험의 내용은 미리 정의된 몇 가지 작업 요청을 임의로 생성하고, 생성된 작업 요청에 대하여 퍼지 할당기가 결정한 서버로 작업 요청을 할당한 후, 이들을 처리하는 과정에 대한 시뮬레이션이다. 다음의 그림 5는 실험 환경의 개략적인 구조를 나타낸다.

· 작업 요청 생성기 : 사용자 작업 요청들을 임의적으로 생성한다. 이때 생성된 각 작업 요청의 총 합은 균일 분포를 갖는다.

· 퍼지 할당기 : 작업 요청 관리자로부터 전송된 작업 요청에 대하여, 퍼지 추론 과정을 수행하여 전송된 작업 요청이 할당될 서버를 결정한다.

· 서버 성능 조정기 : 각 서버의 성능을 변화시키는 역할을 한다. 각 서버의 성능 변화 정도는 실험 패러미터로 지정되며, 서버 성능 조정자는 지정된 패러미터내의 범위에서 각 서버의 성능을 동적으로 변화시킨다.

· 대기큐 관리자 : 각 서버 별로 존재하며, 해당 서버에 할당된 작업을 저장하는 대기큐를 관리한다. 작업 요청 관리자로부터 새로운 작업 요청이 전송되면, 전송된 작업 요청을 대기큐의 끝에 첨가하고, 서버의 현재 작업이 완료되면 대기큐의 처음에서 새로운 작업 요청을 끌어낸다.

· 작업 요청 관리자 : 전체 시뮬레이션을 제어하는 역할을 수행한다. 작업 요청 생성기에서 새로운 작업 요청이 생성되면, 이를 퍼지 조정자에게 전송하여 할당될 서버를 결정한다. 작업 요청이 할당될 서버가 결정되면, 해당 서버의 대기큐 관리자로 작업 요청을 전송하고, 각 서버의 대기큐 관리자에 처리 완료된 작업 요청이 존재할 경우, 이를 클라이언트로 전송하는 역할을 담당한다. 본 실험의 관찰 내용에 해당하는 전체

표 2. 실험 패러미터

실험 패러미터	설정치
서버의 개수	4
전체 시간	10,000 ut
단위 시간당 작업 요청 발생 수	2.5
서버 성능의 변화 정도(%)	0, 20, 30, 50

작업 처리량 및 평균 응답 시간을 계산하는 역할도 수행한다.

다음의 표 2는 실험에서 사용된 패러미터 및 그 설정치를 나타낸다. 표에서 ut는 시뮬레이션 과정에서 적용된 단위 시간을 의미한다. 또한 서버의 성능 변화 정도가 30% 라는 의미는 서버의 평상시 성능을 1로 하였을 때, 서버의 성능이 0.7~1.3 사이에서 동적으로 변화함을 의미한다.

4.2 실험 결과 및 분석

앞서 언급한 바와 같이, 본 실험에서는 제안된 퍼지 할당기를 이용하여 각 작업 요청의 할당 서버를 결정할 경우 전체 작업 처리량 및 평균 응답 시간을 측정하였다. 제안된 퍼지 할당기와의 성능 비교를 위하여, 기존의 할당기를 사용한 경우에 대한 실험도 수행하였으며, 이 경우는 앞서 기술한 내용에서 퍼지 할당기가 기존 할당기로만 대체될 뿐 그 밖의 사항은 동일한 환경에서 결과를 관찰하였다.

그림 6은 서버의 성능 변화 정도에 따른 퍼지 작업 할당 방법과 기존 방법 사용 시의 전체 작업 처리량을 나타낸다.

그림에서 볼 수 있듯이 퍼지 할당기를 사용하는 경우가 기존 방법보다 동일 시간 내에 더 많은 작업을 처리함을 알 수 있다. 즉, 퍼지 할당기를 사용한 경우 기존 방법 보다 약 1.4배 정도의 성능 향상을 얻을 수 있다. 또한 서버 성능의 변화 정도가 심화될수록, 퍼지 할당기와 기존 방법간의 성능 차이는 더욱 커짐을 볼 수 있다. 이는 서버의 성능 변화를 고려하지 않은

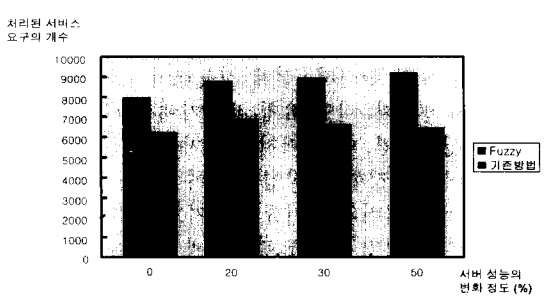

그림 6. 서버 성능의 변화 정도에 따른 작업 처리량

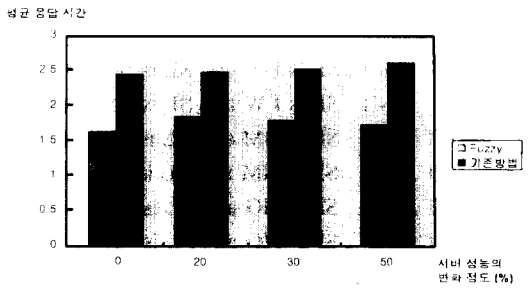

그림 7. 서버 성능의 변화 정도에 따른 평균 응답 시간

기존 방법의 경우, 서버의 성능이 동적으로 변화하는 환경에 부적합한데 반하여, 본 논문에서 제안된 퍼지 할당기의 경우 서버 성능의 변화에도 우수하게 적응함을 알 수 있다.

그림 7은 서버 성능을 변화시켜 가며, 각 작업 요청의 평균 응답 시간을 측정한 결과이다. 앞서 제시한 전체 작업 처리량의 경우와 마찬가지로, 퍼지 할당기를 사용한 경우 기존 방법보다 약 30% 정도의 성능 향상을 얻을 수 있다. 또한 서버 성능의 변화 정도가 심해질 수록 퍼지 방법과 기존 방법간의 성능 차이는 더욱 커짐을 알 수 있다.

5. 결 론

본 논문에서는 다중 서버 환경에서 사용자 작업 요구에 대한 서버 할당기에 대하여 퍼지 개념을 사용하는 방안을 제안하였다. 작업의 예상 수행 시간과 서버별 부하량, 그리고 각 서버의 작업 처리 능력 정도를 퍼지화하고 전문가 지식을 퍼지 규칙으로 사용하였다. 실험을 통하여 본 논문이 제안한 방법이 기존 방법에 비하여 30% 이상 성능이 우수함을 보였다. 특히 서버의 작업 처리량과 작업별 평균 응답 시간 모두에서 좋은 결과를 제공한다는 점에서 제안한 방법의 우수성이 있다고 본다.

향후에는 보다 다양한 변수들을 사용하여 여러가지 환경에 적합한 작업 할당 할당기에 대하여 연구가 필요하다고 생각된다. 분산 다중 서버 환경이나 다중 처리기 환경과 같이 시스템 환경의 차이에 따라 서로 다른 접근 방법이 필요하기 때문이다. 또한 퍼지 기법의 사용에 있어 실시간적 적용 가능성을 판단하기 위한 수행 시간적인 측면의 고찰이 필요하다고 본다.

참고문헌

[1] 이광형, 오길록, "퍼지 이론 및 응용 I, II," 홍릉과학

출판사, 1991.
[2] Mukesh Singhal *et. al.*, "Advanced Concepts in Operating Systems," McGraw-Hill, 1994.
[3] Robert Orfali *et. al.*, "Essential Client/Server Survival Guide," *John Wiley & Sons*, 1994.
[4] Jim Gray *et. al.*, "Transaction Processing : Concepts and Processing," *Morgan Kaufman*, 1993.
[5] Novell, "TUXEDO System Product Overview," 1995
[6] Novell, "TUXEDO ETP Systems Release 5.0," 1995.
[7] J. Mark Stevenson *et. al.*, "Client-Server Interactions in Multi-Server Operating Systems : The MACH-US Approach," CMU TR-94-191, 1994.

정연돈

1994년: 고려대학교 컴퓨터학과 학사
1996년: 한국과학기술원 전산학과 석사
1996년 ~ 현재: 한국과학기술원 전산학과 박사과정 재학중

이지연

1993년: 동국대학교 컴퓨터공학과 학사
1996년: 한국과학기술원 전산학과 석사
1996년 ~ 현재 한국과학기술원 전산학과 박사과정 재학중

이광형

1978년: 서울대학교 산업공학 학사
1980년: 한국과학기술원 산업공학 석사
1985년: 프랑스 응용과학원(INSA) 전산학 석사, 박사
1985년 ~ 현재: 한국과학기술원 전산학과 교수

김명호

1982년: 서울대학교 컴퓨터공학과 학사
1984년: 서울대학교 컴퓨터공학과 석사
1989년: Michigan State Univ. 전산학 박사
1989년 ~ 현재: 한국과학기술원 전산학과 교수

김종수

1995년: 한국과학기술원 전산학과 학사
1997년: 한국과학기술원 전산학과 석사
1997년 ~ 현재: 한국과학기술원 전산학과 박사과정 재학중

오석균

1985년: 충남대학교 계산통계학과 학사
1987년: 충남대학교 계산통계학과 석사
1987년 ~ 현재 현대전자 주식회사 근무
1996년 ~ 현재 한국과학기술원 전산학과 박사과정 재학중

이윤준

1977년: 서울대학교 계산통계학과 학사
1979년: 한국과학기술원 전산학과 석사
1983년: INPG-ENSIMAG 전산학 박사
1984년 ~ 현재: 한국과학기술원 전산학과 교수