분산 모바일 멀티에이전트 플랫폼을 이용한 사용자 기반 디지털 라이브러리 구축

(A Personal Digital Library on a Distributed Mobile Multiagents Platform)

조 영 임 †

(Young Im Cho)

요 약 분산 환경에서 디지털 라이브러리 구축시 기존의 단일 에이전트를 이용한 클라이언트/서버 방식으로 시스템을 구축할 경우, 일차원적인 자료검색으로 인해 검색결과의 관련성이 없고, 검색 결과에 대한 사용자의 성향이 반영되지 않으며, 클라이언트가 서버에 접속할 때마다 인증을 받아야 하므로 다수의 서버 접근시 문서 처리 효율이 낮고 사용하기 불편하다는 문제점을 갖는다.

따라서 본 논문에서는 이의 해결을 위해 기존의 멀티 에이전트 플랫폼인 DECAF와 표준안으로 제시되는 모바일 ORB인 Voyager를 응용해 새로운 모바일 환경에 적합한 멀티 에이전트 플랫폼을 개발 제안하였고, 이를 이용한 사용자 기반의 디지털 라이브러리 시스템(PDS)을 구축하였다. 이러한 접근방법은 국내외적으로 처음 시도되는 연구이다.

새로운 플랫폼은 관련정보의 검색문제를 위해 신경회로망을 이용한 문서분류를 통해 관련 문서의 검색을 세분화시킴으로써 검색결과의 관련성을 높였고, 사용자 성향을 반영하기 위해 모듈화된 클라이언트를 구성하여 신경회로망을 이용함으로써 사용자의 성향과 탐색 결과를 최적화 시켰으며, 네트워크 문제를 위해 멀티에이전트 플랫폼과 모바일 클래스를 이용한 모바일 기능을 개발하였다. 또한 모바일 시스템과 멀티에이전트 시스템을 적절히 결합하고 멀티 에이전트 사이의 협상 알고리즘과 스케쥴링 방법을 개발함으로써 제안한 플랫폼이 효율적으로 동작하도록 구성하였다.

시뮬레이션한 결과, 분산환경에서 모바일 서버의 개수와 에이전트의 개수가 늘어날수록 PDS는 기존의 디지털 라이브러리보다는 탐색시간이 훨씬 줄어들었고 결과에 대한 사용자 만족도도 기존 C/S 방식에 비해 약 4배 정도 향상됨을 알 수 있었다.

키워드 : 분산시스템, 모바일 멀티에이전트 플랫폼, 사용자기반 디지털 라이브러리

Abstract When digital libraries are developed by the traditional client/server system using a single agent on the distributed environment, several problems occur. First, as the search method is one dimensional, the search results have little relationship to each other. Second, the results do not reflect the user's preference. Third, whenever a client connects to the server, users have to receive the certification. Therefore, the retrieval of documents is less efficient causing dissatisfaction with the system.

I propose a new platform of mobile multiagents for a personal digital library to overcome these problems. To develop this new platform I combine the existing DECAF multiagents platform with the Voyager mobile ORB and propose a new negotiation algorithm and scheduling algorithm. Although there has been some research for a personal digital library, I believe there have been few studies on their integration and systemization.

For searches of related information, the proposed platform could increase the relationship of search results by subdividing the related documents, which are classified by a supervised neural network. For the user's preference, as some modular clients are applied to a neural network, the search results are optimized. By combining a mobile and multiagents platform a new mobile, multiagents platform is developed in order to decrease a network burden. Furthermore, a new negotiation algorithm and a scheduling algorithm are activated for the effectiveness of PDS.

† 종신회원 : 평택대학교 컴퓨터과학과

yicho@ptu.ac.kr

논문접수 : 2003년 10월 22일

심사완료 : 2004년 9월 13일

The results of the simulation demonstrate that as the number of servers and agents are increased, the search time for PDS decreases while the degree of the user's satisfaction is four times greater than with the C/S model.

Key words : Distributed system, Mobile multiagents platform, Personal digital library

1. 서 론

에이전트 시스템은 분산 환경에서의 다수 에이전트들이 상호 협동하여 문제를 해결함으로써 작업의 효율을 높이게 되며 각 에이전트는 공통의 업무를 분배해서 처리하거나 각기 다른 일을 맡아서 처리한 후 그 결과를 분석하여 문제를 해결하는 특징을 갖고 있다[1]. 또한 기존의 원격 프로시저 호출(RPC) 방식과 비교 했을 때 네트워크의 상태에 대해 보다 안정적이고 간섭을 덜 받는 에이전트는 네트워크 내의 이동에 보다 자율성을 가지며 네트워크 트래픽을 현저히 감소시키는 장점을 갖는다[2,3].

분산 환경에서 에이전트를 이용하여 개발할 수 있는 시스템은 매우 다양하며, 실제로 많은 분야에 응용되고 있다. 그 중 최근에 주목받고 꾸준히 연구되고 있는 분야로써 에이전트를 이용한 디지털 라이브러리 시스템이 있다. 디지털 라이브러리는 전자 도서관 또는 가상 도서관 등으로 불리우는 것으로, 컴퓨터 및 관련 산업의 발달로 네크워크 기반 시설이 충분하게 구축됨에 따라 기존의 문서 기반 도서관을 디지털화된 자료로 데이타베이스를 구축해 기존 도서관의 기능인 정보 제공 및 보존을 대신하고자 하는 개념이다[4].

그러나 분산 환경에서 디지털 라이브러리 구축시 단일 에이전트에 의한 클라이언트/서버 방식의 시스템은 검색시 효율면에서 몇가지 문제점이 발생하게 된다. 즉, 검색결과의 낮은 관련성, 사용자 성향 미반영, 네트워크에 종속적인 문제점을 갖는다. 최근에는 사용자 기반 디지털 라이브러리 구축을 위한 연구들이 시도되고 있으나[5,6], 기존 디지털 라이브러리의 한계점을 여전히 갖고 있다.

따라서 본 논문에서는 이의 해결을 위해 분산 모바일 멀티 에이전트 시스템에 기반을 둔 새로운 플랫폼을 제안하고자 한다. 이를 위해 분산 에이전트 플랫폼인 DECAF(Distributed Environment Centered Agent Framework)[2]와 최근 분산 모바일 표준안으로 제안되고 있는 모바일 ORB(Object Request Broker)인 Voyager[7]를 응용해 새로운 모바일 환경에 적합한 멀티 에이전트 플랫폼을 구축하고, 이를 이용한 사용자 기반의 디지털 도서관 서비스 시스템(PDS: Personal Digital Library System)을 구축하고자 한다.

지금까지 디지털 라이브러리에서 개인화된 정보검색이나 웹 서치 및 분석 스파이더들은 개발되어 있고, 여러 서버들로부터 사용자가 원하는 정보를 검색 및 필터링하여 사용자의 컴퓨터 또는 서버로 이동하여 데이타베이스를 자동으로 구축해주는 개인화된 디지털 라이브러리 구축 및 개발에 관한 연구가 부분적으로 이루어지고 있으나, 새로운 플랫폼 개발과 사용자 중심의 디지털 라이브러리의 특성들을 통합한 개념의 체계적인 개발 사례는 국내외적으로 아직까지 연구결과가 발표되지 않고 있다.

본 논문의 구성은 다음과 같다. 먼저 2장에서는 에이전트를 이용한 디지털 라이브러리의 국내외 연구현황과 문제점을 분석하고, 3장에서 기 개발된 에이전트 관련 기술들을 분석하고자 한다. 4장에서는 PDS의 제안동기와 주요기술을 설명하고 5장에서 시뮬레이션 및 결과를 분석하며, 마지막으로 6장에서 결론 및 향후 연구과제를 제시하고자 한다.

2. 디지털 라이브러리 현황 및 문제점

디지털 라이브러리의 장점은 네트워크가 연결된 사용자의 컴퓨터만 있으면 시간과 장소에 구애받지 않고 접근이 가능하며 검색 엔진을 이용한 간단한 조작으로 원하는 정보를 찾을 수 있다. 최근 분산화 경향에 따라 데이타의 확장성, 융통성, 데이타베이스 분산화, 다양한 서비스 등을 장점으로 하는 에이전트를 이용한 디지털 라이브러리에 대한 관심이 높아지고 있는데 대표적인 연구들을 살펴보면 다음과 같다.

미국의 미시간 대학이 에이전트를 이용한 디지털 라이브러리를 연구하는 대표적인 곳이다[8]. 이곳에서는 분산환경에서의 에이전트 동작 문제 등에관해 주로 연구하고 있으나, 모바일 멀티 에이전트의 새로운 플랫폼이 아닌 기존 에이전트 플랫폼에 에이전트간 통신 방법을 개선함으로써 디지털 도서관을 구축하였다. 또한 사용자 중심의 자료검색이나 사용자의 관심도를 구체화하는 프로파일 생성 작업은 이루어지지 않고 있다. 미국의 스탠포드 대학의 전자도서관 프로젝트는 주로 컴퓨터 관련 문헌을 디지털화하고 있으며, 일반적인 정보는 네트워킹을 통해서 접근할 수 있도록 하고 있다[9]. 그러나 단순 검색시 주어지는 질의어로 인한 사용자에 맞는 데이타베이스 추천 기능일 뿐 사용자의 종합적인 관심정보를 학습하여 이를 제공하는 기능은 없다. 이밖에 미

국의 UC 버클리 대학이나 카네기 멜론 대학의 디지털 라이브러리는 주로 데이타베이스 구축이나 멀티미디어 기술 등을 연구하고 있으나 분산환경에 따른 모바일 멀티에이전트의 연구와는 차이가 있다[10].

국내에서는 상용화된 디지털 도서관으로는 연구개발정보센터[11]와 국립 중앙도서관[12] 등이 있어서 다양한 검색방법을 제공하고 있으나 에이전트에 의한 사용자 관심도 반영이나 멀티에이전트 등에 관해 연구되어 있지 않다.

기존의 단일 에이전트에 기반한 디지털 라이브러리에서의 정보검색상의 문제점으로는 첫째, 일차원 검색으로 해당 단어에 대한 존재 유무만을 판별하므로 결과값이 단순하고, 둘째 사용자에 대한 사전정보가 없는 상태에서 원하지 않는 결과까지 포함하고 있으며 셋째, 클라이언트가 서버에 접속할 때 매번 인증을 받아야하며 네트워크의 영향을 현저히 받는다는 것이다. 개인 디지털 라이브러리에서는 검색 결과를 인덱싱 작업을 통한 개인용 컴퓨터에 저장하고 모니터링하는 과정을 통한 지속적인 서비스 또한 필요하다.

이러한 사용자 중심의 디지털 라이브러리에서는 매번 인증절차를 거쳐 많은 디지털 라이브러리에 접속해야 하며 관련있는 정보들의 일차원적 검색을 통해 사용자 스스로 관련성 여부를 판단해야 하며, 자신의 컴퓨터에 라이브러리를 구축하기 위해 디렉토리 설정 등 일일이 사용자 스스로 해야 하므로 사용자 입장에서 보면 다소 번거롭다. 이것은 회원들에게 제공되는 전자 논문 원문 서비스와는 차별화된 것이므로 사용자의 관심도를 학습하여 사용자 컴퓨터에 자동으로 데이타베이스를 구축해 줄 수 있는 보다 지능적인 개인화된 사용자 중심의 디지털 라이브러리 구축 시스템에 관한 연구가 필요하다.

본 논문에서는 에이전트를 이용한 효율적인 사용자 기반 디지털 라이브러리 구축을 하기 위한 주요 기술인 에이전트와 모바일 에이전트에 대한 기존 연구를 분석하고자 한다.

3. 에이전트 분석

3.1 분산환경과 에이전트 플랫폼

에이전트 플랫폼의 국제 표준화 규격을 정의하는 에이전트 기술표준화 기구에서 제안한 FIPA(Foudation for Intelligent Physical Agents)가 에이전트 표준 플랫폼으로 추진되고 있는데, AP(agent platform)가 이 플랫폼의 기본 단위가 된다. FIPA에서 에이전트는 하나 이상의 플랫폼에 등록되어 자신이 속한 플랫폼에 소속한 에이전트에 대한 서비스를 제공하게 된다[14]. FIPA 에이전트 플랫폼에서는 에이전트의 등록, 동적 구동, 그리고 종료 등의 전반적인 에이전트의 관리를 수행한다. 그러나 FIPA 에이전트 플랫폼은 중앙집중적 구조로 인해 시스템의 자동성이 부족하며 멀티에이전트 환경에 적합하지 않은 단점을 갖는다.

반면, 분산환경에 적합하도록 미국의 University of Delaware에서 개발된 DECAF 멀티 에이전트 플랫폼은 에이전트 표준 규격을 준수하면서도 지능적 에이전트를 신속하게 설계할 수 있는 중앙집중적이 아닌 분산환경에 매우 적합한 멀티 에이전트 플랫폼으로 평가되고 있는 플랫폼이다[2].

DECAF는 그림 1에서와 같이 에이전트 통신, Planning, Scheduling, Monitoring, Coordination, 진단, 학습 등을 평가하고 생성하기 위한 모듈화된 플랫폼을 제공하는 등 일종의 에이전트 운영체제 역할을 수행한다. 또한 DECAF는 스스로 소켓 프로그램을 생성하고 메시

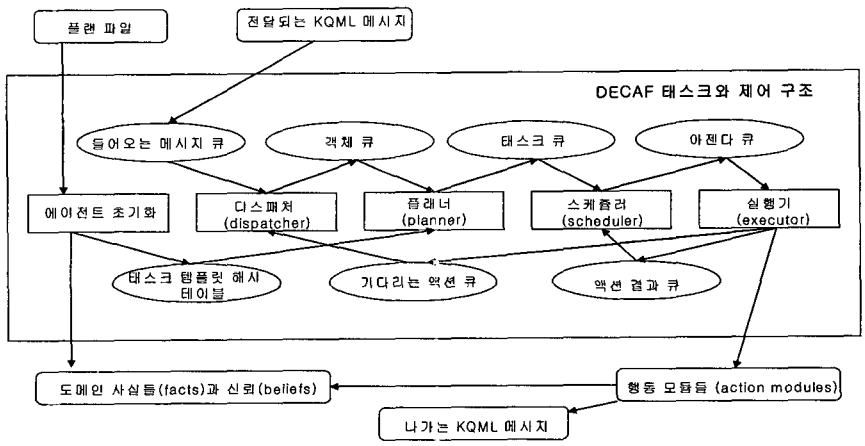


그림 1 DECAF 멀티에이전트 플랫폼

지를 포맷하여 에이전트 통신을 수행하는 빌딩블럭을 제공하는 등 기존의 자바 기반 에이전트 프레임워크와는 차별된 기능을 가진다. 따라서 사용자나 프로그래머는 API 접근 방법에 대한 지식이 없이도 에이전트를 생성할 수 있게 되며 프로그래머는 직접 통신코드를 작성할 필요가 없는 장점이 있다. DECAF를 이용하여 멀티 에이전트 시스템을 구축함에 있어 장점으로는 에이전트 생성, 제어에 필요한 기본적인 모듈을 갖추고 있다는 점이다.

그러나 DECAF 플랫폼은 비모바일 환경에서 개발되었으므로 에이전트 간의 협상, 스케쥴링 등에서 매우 제한적인 기능을 갖고 있어서, 모바일 환경으로의 새로운 플랫폼 개발이 필요하다.

3.2 모바일 에이전트와 모바일 멀티에이전트 시스템

분산 환경에서의 고전적이고 일반적인 처리 방법으로 많이 사용되는 것이 원격 프로시저 호출(RPC)이다. RPC는 1970년대 제안된 방식으로 모든 메시지가 네트워크를 타고 이동하며, 각 메시지는 프로시저를 요청하거나 확인할 수 있다[15].

그러나 RPC에는 클라이언트와 서버가 네트워크를 통해 이동하는 데이타를 사용한다는 부담이 있으므로 이러한 분산 컴퓨팅에 대한 대안이자 새로운 패러다임을 형성하고 있는 것이 바로 모바일 에이전트다[15]. 모바일 에이전트는 원격 프로그래밍에 기초한 것으로서 통신시 상대가 가지고 있는 프로시저의 호출 뿐 아니라 수행해야 할 프로시저까지 제공한다. 모바일 에이전트는 자신이 이동할 때만 네트워크를 사용하므로 네트워크 대역폭을 많이 차지하지 않으며, 자신이 이동하고 난 후 호스트와의 네트워크 연결이 끊어졌을 경우도 계속적인 실행이 가능하다. 모바일 에이전트는 IBM Japan에서 개발된 Aglets가 대표적인데[16], Aglets은 applet에 이동성을 부여하려는 목표를 가진 자바기반의 모델이며 다양한 애플리케이션에 적용되도록 하였다.

모바일 멀티 에이전트는 멀티 에이전트와 모바일 에이전트의 결합 시스템으로서, 멀티 에이전트는 분산 환경에서의 효과적인 시스템 자원의 사용과 작업 분할과 할당, 그에 따른 에이전트들의 협동 등이 중요시 되는 반면에 모바일 에이전트는 에이전트 간 메시지 전달, API 등 프로시저를 가진 객체가 네트워크를 이동하여 네트워크의 안정성, 신뢰성에 구애 받지 않고 작업을 수행할 수 있도록 하는 것 등이 중요시된다.

이와 같이 모바일 시스템과 멀티 에이전트 시스템은 어느 정도 공유하는 영역을 지니고 개발되고 있으나, 실제 모바일 멀티 에이전트에 대한 연구나 개발은 다양하게 이루어지지 않고 있는데, 그 이유는 보안, 에이전트 간 협상이나 스케쥴링 등에 대한 연구가 부족하기 때문이다. 실험적으로 Norwegian University의 Javaspace를 이용한 시스템인 CAGIS DIAS가 연구중에 있으나 에이전트간 협상 등에 대한 연구는 매우 미흡하다[17].

4. 개인 디지털 도서관(PDS) 구축

4.1 제안동기 및 시스템 개요

본 논문에서는 새로운 분산 모바일 멀티 에이전트를 이용한 개인 인증 절차를 거쳐서 기 구축된 국내외 디지털 라이브러리에 접속하여, 학습된 사용자 프로파일을 바탕으로 서버 간을 이동하면서 정보 검색을 하여 관심있는 문서를 추출하여 사용자의 컴퓨터 또는 서버에 자동으로 인덱싱 과정을 통해 데이타베이스 구축을 함으로써, 사용자가 수시로 볼 수 있게 하는 사용자 기반 개인 디지털 라이브러리 시스템(PDS)을 구축하고자 한다. 기존의 클라이언트/서버 모델이 기본적으로 일대일 탐색을 지원한다면, 멀티 에이전트를 이용한 탐색은 다수의 서버에 대한 동시 탐색을 지원한다. 이때 단순 탐색만을 지원하는 것이 아니라 탐색 결과를 협상을 통해 필터링하는 기능도 제공한다. 검색한 결과의 필터링은 신경회로망을 이용한 협상 알고리즘을 이용하여 사용자 성향을 반영할 수 있다.

본 논문에서 제안한 모바일 멀티 에이전트 플랫폼의 특징은 분산 환경에서 동작의 최적화를 위해 모바일 시스템과 분산 처리 시스템을 결합한 형태의 프레임워크를 사용한 것이다. 이를 위해 단일/비모바일 에이전트 플랫폼을 분산 처리 환경에서 멀티 에이전트를 지원하는 DECAF 프레임워크를 사용하여 멀티 에이전트 플랫폼으로 확장하였으며, 모바일 환경을 지원하는 모듈을 추가하였다. 모바일 기능을 위해 분산 모바일 표준안으로 제안되고 있는 Voyager 모바일 에이전트 플랫폼을 응용하였다. Voyager는 ObjectSpace사에서 개발한 분산 시스템을 효율적으로 개발 하는데 사용할 수 있는 자바 기반의 분산 컴퓨팅 플랫폼이다[7]. Voyager는 RMI와 같이 자바 문법을 이용하여 원격 객체를 만들어 이동하게 할 수 있으며, 프로그램 사이에서 메시지를 이동시킬 수 있는 특징을 가지고 있다. 기본 구조의 분산 애플리이션과 모바일 에이전트를 모두 만들 수 있는 플랫폼이며 ORB로서의 성능도 뛰어나다. 본 논문에서는 에이전트가 처리해야 할 각각의 태스크를 멀티 에이전트에서 사용되는 GPGP[18] 협상 알고리즘을 개선한 새로운 협상 알고리즘을 이용하고 있다. 협상이론을 바탕으로 사용자가 요구한 전체 태스크를 분할하여 자료구조화 시킬 수 있는 TAEMS(Task Analysis, Environment Modeling and Simulation)[19] 구조를 이용해 멀티에이전트 스케쥴링을 통해 분할 처리한다.

GPGP는 미국의 University of Massachusetts에서

연구된 광범위하게 사용되는 멀티에이전트 협상알고리즘이다. GPGP는 에이전트 상호간의 중복작업을 줄이기 위해 발생한 과다한 통신문제와 그로 인해 발생한 시스템의 오버헤드를 감소시키는 것과 특정 멀티 에이전트 시스템이 도메인 영역에 종속적이지 않게 한다. 즉, 서로 다른 기능을 가지는 이질적 에이전트로 구성된 멀티 에이전트 시스템의 구성을 가능하게 하였다. 그러나 GPGP에서의 멀티 에이전트간 협상방식이 에이전트간 제어권의 이동 정도이므로 복잡한 환경에 적합하지 않다는 단점을 갖는다.

다음 그림 2는 TAEMS 태스크 구조를 표현한 것이다. 제일 위쪽의 루트 태스크는 작은 서브 태스크로 분류할 수 있고 서브 태스크는 더 이상 분해할 수 없는 메소드로 분해된다. 가장 밑에 있는 leaf node는 메소드의 역할을 하는데 각 에이전트가 실질적으로 수행하는 행동을 나타내는 것이다. DECAF는 에이전트가 수행해야 할 전체 태스크를 TAEMS라는 내부적 자료 구조를 형성함으로써 전체 시스템의 목표를 합리적으로 수행해 나간다.

모바일 멀티에이전트에서는 보다 복잡하고 다양한 에이전트들이 생성되므로 에이전트간의 제어권 협상은 물론 결과에 대한 협상도 수반되어야 한다. 따라서 본 논문에서는 기존 연구된 GPGP와 TAEMS를 분석하여 새로운 협상 알고리즘을 개발하여 적용하였다.

또한 기존의 라이브러리 모델이 사용자에게 단순히 원격지로부터의 검색 결과만을 제공하는 반면, 제안한 모델에서는 사용자 프로파일 학습에 의한 체계적인 정보의 필터링 및 저장을 통한 사용자의 로컬 디지털 라이브러리를 구축하도록 하였다.

본 논문에서 구축한 에이전트를 이용한 디지털 라이브러리는 크게 네 가지 구성요소를 가지고 있으며, 각 요소는 정보를 요구하는 사용자, 디지털화된 데이타베이스, 문서를 디지털화해 저장하는 저자, 그리고 데이타베이스로부터 정보를 검색/가공하여 사용자에게 전달해주는 에이전트가 있다.

본 논문에서 제안한 시스템은 모바일 멀티 에이전트 플랫폼을 기반으로 하여 클라이언트가 네트워크 상태에 대해 안정적으로 동작할 수 있도록 하였으며, 이는 여러

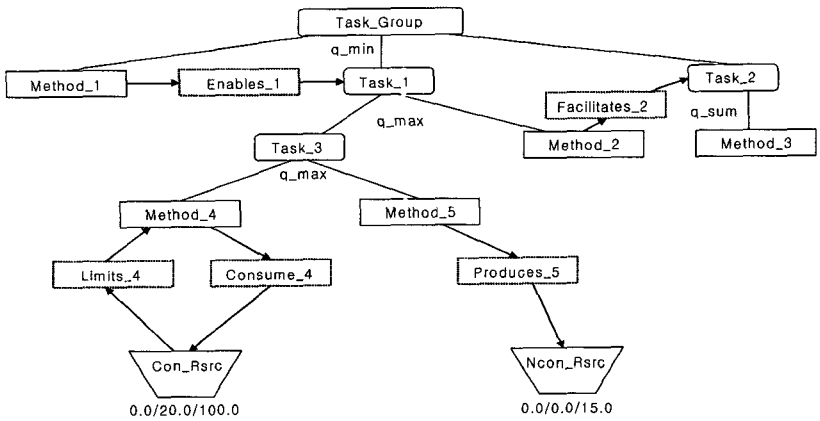
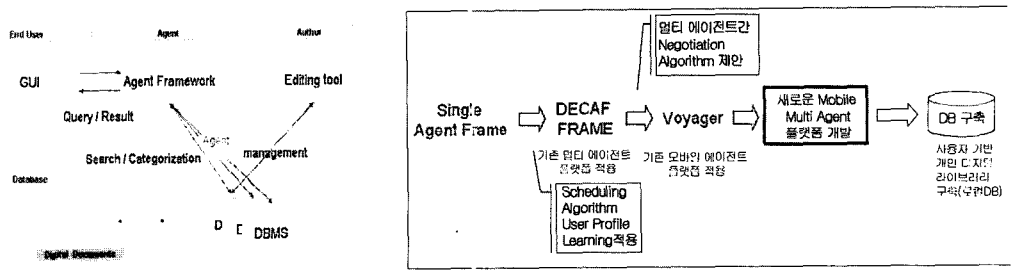


그림 2 TAEMS 태스크 구조



(a) 에이전트 기반 디지털 라이브러리의 개념 (b) 본 논문에서 제안하는 연구의 흐름

그림 3 모바일 멀티 에이전트 기반의 디지털 라이브러리 개요

서버를 다수의 에이전트를 이용하여 동시에 검색하는 것이 가능하게 된다. 또한 사용자 프로파일을 데이타베이스화 함으로써 사용자의 성향에 맞는 결과를 보여준다. 또한 멀티 에이전트의 특성상 네트워크 점유율이 낮은 이점도 가지게 된다.

4.2 시스템 구조

PDS의 전체적인 구조는 그림 4와 같이 기존의 클라이언트/서버 시스템을 개선하여 신경회로망을 이용한 사용자의 성향 유지 분석과 모듈화된 파트를 가진 클라이언트 측과 디지털 문서를 저장하고 있는 서버군으로 시스템을 구성한다.

4.2.1 클라이언트

클라이언트는 크게 세부분으로 구성되어 있다. 기존의 DECAF 구조에 첫째 사용자에게 라이브러리에 대한 직접적이고 직관적인 제어를 가능하게 해주는 사용자 인터페이스와 사용자의 경험이 반영되어 사용자 성향을 데이타베이스화하여 정보를 유지하는 사용자 프로파일 제어부, 그리고 에이전트를 생성하여 원격지 서버에 대한 검색을 수행하고 그 결과를 구축된 사용자 성향을 적용해 결과를 돌려주는 PLA(Personal Library Agent)로 구성된다.

① PLA(Personal Library Agent)

PLA는 두개의 모듈인 모니터링 에이전트 모듈과 협상에이전트 모듈과 검색결과를 저장하는 두개의 DB로 구성된다. 모니터링 에이전트는 Voyager와 DECAF의 효과적인 제어를 위한 부분으로 에이전트의 이동과 실행을 제어하고 모니터링 하기위한 것이다.

사용자의 검색요청에 대한 키워드가 PDS에 들어오면 먼저 PLA에 전달되게 되고 모니터링 에이전트는 현재 라이브러리와 온라인으로 연결된 가용한 서버가 있는지 서버와 연결 상태를 확인한다. 그리고 서버마다 에이전트를 생성하여 보내고 에이전트의 동작 상태를 확인한다. 이때 에이전트가 성공적으로 동작하지 않으면(즉, 주어진 시간내에 결과가 도착하지 않으면) 모니터링 에이전트는 다시 서버에 에이전트를 보내게 된다. 각 에이전트들의 검색된 결과는 PLA내의 임시 저장소내에 일차적으로 저장된다. 이는 다시 협상 에이전트를 거쳐 검색결과를 필터링하게 되며 최종결과를 결과 저장소에 저장한다. 이렇게 구축된 결과 저장소의 데이타베이스는 사용자 인터페이스를 통해 최종 결과로써 보여지게 된다.

협상 에이전트에서는 본 논문에서 제안한 그림 5와 같은 멀티 에이전트간 협상 알고리즘을 이용하여 협상을 한 후 결과를 임시기억장소를 통해 사용자 인터페이스에 나타내준다.

본 논문에서 제안하는 협상 에이전트 알고리즘은 기존에 제안된 GPGP와는 달리, 태스크를 TAEMS 구조에 의해 분류하고 가장 합리적인 각 에이전트의 실행순서를 결정하기 위해 에이전트내 메소드간 관계를 시스템내에서 자동으로 정의하여 에이전트는 스케쥴링 한 점이 큰 차이점이다.

```
/* improved negotiation algorithm of multiagents */
Switch(one of 5 relations) { /* relation_name(sender,
receiver) */
 case 1: Add_R(Ai_Mi, Aj_Mj) /* (i≠j), Ai=agent,
         Mi=method */
          while(Aj_Mj is finished)
                no-operation (Ai-Mi); break;
 case 2: Compensate_R(Ai_Mi, Aj_Mj)
         Ai_Mi and Aj_Mj are operated in real
         situation by call(inference) function;
         Inference() is from user profiles or user
         information repository;
         break;
```

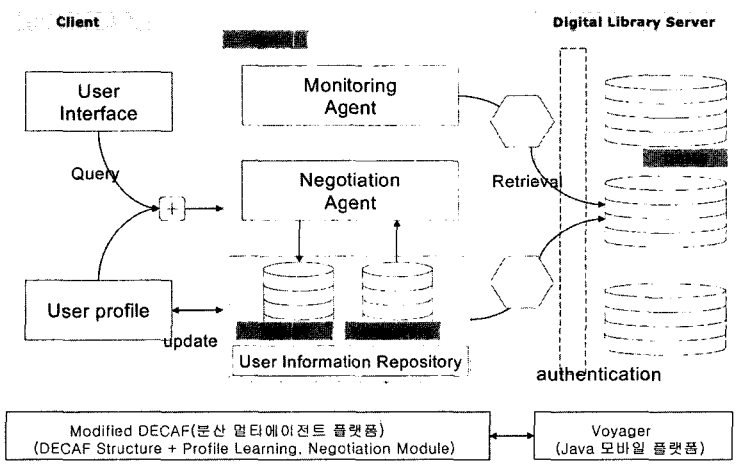


그림 4 제안한 PDS의 전체 구조

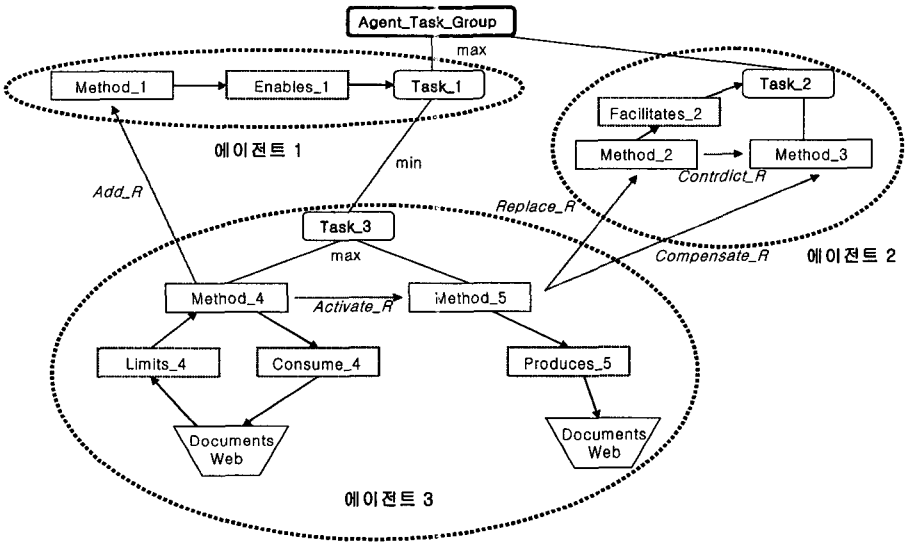


그림 5 협상 에이전트 모듈 상호작용

```
    case 3: Replace_R(Ai_Mi, Aj_Mj)
            Aj_Mj=Ai_Mi;
            break;
    case 4: Contradict_R(Ai_Mi, Aj_Mj)
            Aj_Mj=Aj_Mj;
            break;
    case 5: Activate_R(Ai_Mi, Aj_Mj)
            if wait(Aj_Mj) then awake(Aj_Mj) and restart;
            break;
    default:
}
```

Agent_Task Group은 에이전트 1, 에이전트 2, 에이전트 3에 따른 각 Task_1, Task_2, Task_3을 생성한다. Task들은 메소드들로 자동 분할되며 각각 할당된 일을 수행하게 된다. 각 메소드들은 5가지 메소드간의 관계를 갖게 되는데, Add_R은 한 메소드의 결과를 다른 메소드의 결과에 추가하라는 관계이며, Activate_R은 활성화시킴으로써 계속 그 메소드가 일을 할 수 있도록 하는 것이며, Compensate_R은 메소드간 결과에 대한 조정이 필요한 관계를 말한다. Replace_R은 받는 메소드 결과를 주는 메소드의 결과로 대치하라는 관계이며, Contradict_R은 받는 메소드 결과를 무시하라는 관계를 말한다. 또한 메소드와 태스크, 메소드와 리소스 간에는 Enable, Facilitate, Produce, Consume, limit 등의 관계를 갖는데, Enable은 서로 영향을 미치는 관계, Facilitate는 조정하는 관계, Consume은 소비하는 관계, Limit는 제한하는 관계, Produce는 생산하는 관계를 나타낸다. 협상알고리즘에서는 기본적으로 동일 레벨의 태스크이며 서로 다른 일을 수행하는 에이전트인 경우는 max의 연산을 취하며, 생성된 하위레벨의 태스크인경우는 min의 연산을 취함으로써 각 문서에 대한 최소값을 취하게 된다.

PLA가 PDS내의 다른 모듈과 원격지 서버에 대해 상호작용하는 과정에 대한 순서는 다음 그림 6과 같다.

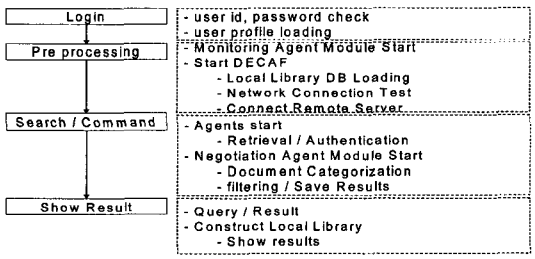


그림 6 PDS의 세부 모듈간 상호작용

먼저 사용자가 Login 하게 되면 PLA는 모니터링 에이전트 모듈을 작동시키게 된다. 전처리 과정으로 모니터링 에이전트는 원격지 서버에 대한 호출(call)을 보내어 원활한 접속이 되는지 확인한다. 사용자의 요청이 들어오게 되면 모니터링 에이전트는 에이전트를 생성하여 원격지 서버에 보내게 된다.

② 사용자 프로파일 구축

사용자 프로파일은 사용자가 최초 입력한 정보를 기반으로 1차 프로파일을 작성하게 된다. 작성된 프로파일은 DB 형태로 디지털 라이브러리 모듈에 저장되게 된다. 디지털 라이브러리를 이용한 문서 탐색시 PLA에서 생성된 에이전트는 네트워크내의 서버로 이동하여 문서에 대한 결과를 가져온다. 이때 사용자의 프로파일 성향

에 따라 문서 분류에 가중치를 두어 결과를 보여준다.

사용자 성향에 대한 업데이트는 그림 7과 같이 사용자의 검색 한 결과에 의해서 사용자가 사용한 DB에 해당하는 키워드와 검색 결과에서 찾아진 문서를 다운로드시 각각 피드백을 통한 가중치를 할당, DB에 업데이트된 결과를 유지하게 된다.

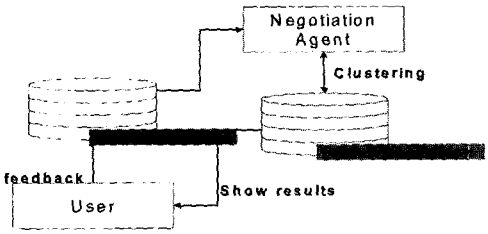


그림 7 사용자 프로파일 업데이트 모델

4.2.2 디지털 라이브러리 서버

① 문서 탐색 및 분류

PLA로부터 생성된 에이전트는 원격지 라이브러리 서버로 이동하여 검색 조건에 맞는 문서를 탐색하는 작업을 수행한다. 제안한 시스템은 내용 기반 문서 여과를 사용하고 있으며 이는 유한개의 키워드를 사용하여 문서의 특성을 나타내는 요소들로 N개의 특징을 정의한 후 문서를 표현하는 것이다. 문서의 여과에 사용되는 기법에는 사례 기반 추론, 결정 트리, 신경회로망등이 있는데, 신경회로망을 이용하는 사용하는 경우 N개의 특징 벡터가 입력층을 구성하며 결과에서 얻어온 문서를 학습 세트로 해서 각 입력에 연결된 벡터의 가중치를 조절하여 각 문서가 속한 카테고리를 결정하게 되므로 효율적이다.

본 논문에서는 Kohonen의 SOM(Self-Organizing Map) 네트워크[20]를 이용하여 문서의 분류를 효과적으로 하고자 한다. SOM은 무교사 학습 방법의 일종으로 N차원의 입력 데이타들을 군집화하여 정해진 차원으로 매핑시킨다. 본 논문에서는 단층의 SOM 네트워크를 사용하였으며 구조는 N차원의 입력 노드와 2차원으로 이루어진 k개의 분류 영역을 표현하기 위한 출력노드로 이루어져 있다. 모든 입력 노드들은 출력 노드와 연결되어있고 연결 가중치 W_{ij}을 가진다.

그림 9는 입력 노드 i와 출력 노드 j를 연결하는 가중치 벡터 W_{ij}의 행렬을 나타낸다. 행렬에서 i번째 행은 입력 노드 i로부터 출력 노드 j로 들어오는 연결 가중치를 나타낸다. 여기서 j번째 열로 나타내어지는 벡터는 j번째 가중치 벡터라고 하며 입력 벡터와의 유클리디언 거리 계산에 쓰인다.

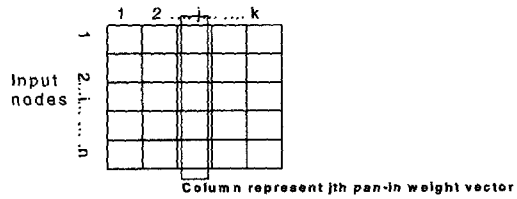


그림 9 가중치 벡터 Wij에 대한 행렬표현

초기 상태에서는 임의의 연결 가중치를 각 연결 벡터에 할당한다. 임의의 연결 가중치를 할당한 후 입력 벡터와의 유사성을 측정하는데 이때 유클리디언 거리를 측정하여 거리를 측정한다. 즉 유클리디언 거리가 가장 작은 j번째 가중치 벡터를 찾아 그 입력 벡터에 대해 j번째 출력 노드가 승자가 되는 것이다. 이렇게 선택한 승자는 입력 가중치를 갱신하는데 갱신하는 식은 다음과 같다.

$$W^j(t+1) = w^j(t) + \alpha(t)[x(t) - W^j(t)]$$

$$\alpha(t) = learning \quad rate$$

각 입력에 대해 승자 노드가 정해지고 그에 따라 가중치 벡터가 수정됨에 따라 네트워크는 각 분류 영역에 대해 대표값을 가지게 된다. 학습을 거친 SOM 네트워크는 새로운 입력을 받을 경우 맵 상에 분류된 가장 유사한 노드가 승자가 되어 해당 노드가 속한 클래스로 분류되게 된다. 다수의 서버로 이동한 에이전트는 검색에 대한 결과를 임시 저장소에 저장하게 된다. 협상에이전트는 SOM을 이용하여 이 결과들에 대한 분류를 수행하여 사용자에게 중복되거나 불필요한 정보를 제거하고 결과를 보여준다. 사용자는 이에 대해 결과에 대한 피드백을 제공함으로써 프로파일을 업데이트 시킬 수 있다.

② Voyager를 이용한 에이전트 이동

Voayger는 코드를 전혀 고치지 않고도 어떤 자바 클래스이든 원격에서 활성화시킬 수 있고, IDL과 IIOP의

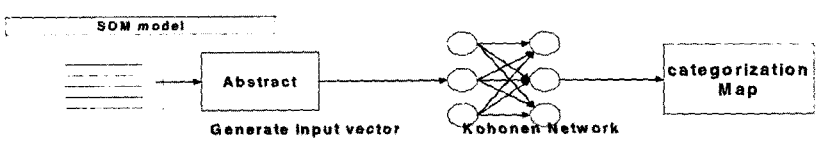


그림 8 SOM을 이용한 문서 분류 지도 생성

CORBA 기능을 모두 완벽하게 지원한다. 따라서 코드를 전혀 고치지 않고도 자바 객체를 CORBA에서 동작하게 바꿀 수 있다. 특정 자바 클래스를 원격 개체로 활성화시키고 싶다고 해서 별도의 수정없이 런타임시에 활성화가 가능하다. 런타임시에 이미 실행되고 있는 주 객체 (Primary)에 보조 객체(facet)을 부착시켜 활성화시킬 수 있다.

Voyager 스펙에 따라 모바일 애플리케이션 개발을 위해 다음의 네 단계를 사용하였다.

(1) 원격 인터페이스를 정의하고 구현한다.
(2) 원격 인터페이스를 사용하는 클라이언트를 만든다.
(3) Voyager 서버를 실행한다.
(4) 클라이언트를 실행한다.

5. 시뮬레이션

5.1 사용자 인터페이스 및 시스템 구축

사용자 인터페이스는 크게 네 개의 Pane으로 구성되어 있으며 각 Pane은 내부 기능과 직접적으로 연관되어 동작이 편리하도록 배치하였다. 각 기능별로 구분해 보면 크게 사용자 정보 관리, 데이타베이스 검색 결과 확인으로 나뉘며 다시 검색 결과는 로컬 데이타베이스 검색 결과와 원격지 서버 탐색 결과인 원격 데이타베이스 검색 결과로 구분할 수 있다.

세부 기능을 살펴보면, 먼저 사용자 인증 기능의 로그인 윈도우와 원하는 문서의 질의 기능을 가진 질의 윈도우, 그리고 에이전트의 동작상태를 체크해 주는 모니터링 윈도우가 포함된 User Pane, 원격지의 디지털 라이브러리 서버의 정보를 나타내어주고 검색 결과를 보여주는 Remote Pane, 사용자 컴퓨터 내에 로컬 라이브러리가 구축된 현황을 보여주는 Local Pane으로 구분된다.

본 논문에서 구현한 PDS의 인터페이스와 내부 모듈의 작업 수행 과정을 에이전트의 수행 동작을 중심으로 나타내면 그림 10과 같다.

① 사용자가 인터페이스를 통해 로그인한다.
② PDS는 모니터링 에이전트(MA)를 실행하고, MA는 현재 연결된 원격지 라이브러리(remote digital library)를 확인한다. 이때 사용자 프로파일(user profile)을 데이타베이스로부터 읽어들인다.
③ 사용자의 요청에 의해 질의가 PLA로 전송된다.
④ MA는 검색 에이전트를 생성시키고 이를 ANS (Agent Name Server)에 등록한다.
⑤ 생성된 에이전트 1은 원격지 라이브러리로 이동한다.
⑥ 에이전트 1은 검색 조건을 PLA로부터 파라메터로 넘겨받아 검색을 수행한다.
⑦ 결과값은 문서가 위치한 원격지 라이브러리 이름과 인덱스 값, 문서의 요약(abstract)을 가지고 에이전트 1에게 넘겨져 PLA내의 협상 에이전트(NA)에게 보낸다. 이는 MA에게 보고되고 MA는 ANS에 이를 통보한다.
⑧ NA는 얻은 결과값을 SOM을 이용해 분류(clustering)한다.
⑨ 사용자 프로파일에 의해 얻어진 결과값은 인터페이스의 Remote pane에 나타난다.
⑩ 사용자는 필요한 결과값을 얻어 자신의 로칼 라이브러리로 이동시킨다.
⑪ 이때 사용자 프로파일의 업데이트가 동시에 이루어진다.

5.2 SOM을 이용한 문서 분류

사용자 프로파일을 구축하기위해 본 논문에서는 단일층의 SOM 네트워크를 이용하여 탐색한 문서의 결과값을 분류하였다. 입력 벡터는 벡터공간 모델을 사용하였으며 특정단어(feature word)는 검색 결과에서 얻어진 문서의 요약(abstract) 부분에서 추출하였다. 샘플로 사

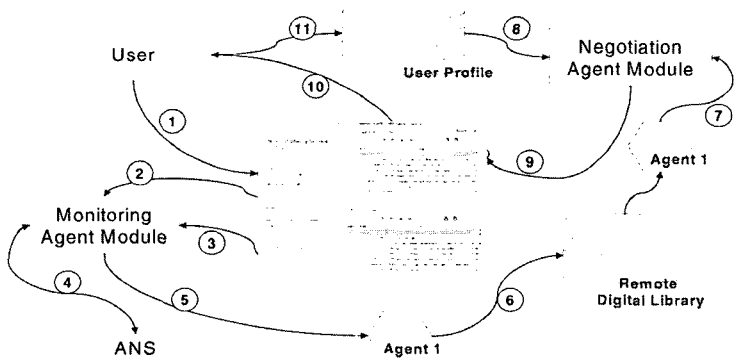


그림 10 PDS 내의 에이전트의 수행동작

용된 데이타는 크게 세 분류의 문서로 나누어지며 각 분류에 대한 예약어를 정해 사용하였다. 각 카테고리의 샘플 문서는 문서 검색 시스템인 Citeseer(http://citeseer.nj.nec.com)에서 얻은 결과를 기본으로 작성하였으며 입력 벡터는 각 문서의 요약에서 얻어진 단어의 빈도수대로 순위를 매긴 후 해당 카테고리에 적합한 단어를 임의로 추출하여 나타내었다(표 1).

입력 벡터를 나타내기 위하여 각 개별 단어를 하나의 벡터로 할당하였으며 30차원의 입력 벡터를 만들었다. 학습과정은 세 개의 원격 디지털 라이브러리를 만들어 놓고 하나당 100개의 문서를 카테고리 구분없이 랜덤하게 데이타베이스로 구축하였다.

이때 SOM에 적용한 파라미터는 출력차원을 위해 5×5 의 2차원 배열을 사용하였고, 분류시 이웃반경을 2로 하였고 학습횟수와 학습율을 각각 500번, 0.05로 하였다. 시뮬레이션에 사용한 클라이언트/서버 컴퓨터 사양은 CPU 1.8 GHz(RAM 1024MB)의 윈도우 NT기반이며, MySql에서 실험하였다.

SOM 네트워크에서 실험한 결과를 위의 세가지 카테고리별로 분류해 보면 다음 그림 11과 같다. 그림에서 숫자는 분류된 문서들의 개수를 나타내는데 각 카테고리별로 숫자들을 모두 더하면 문서의 총 갯수인 100이 된다.

위의 시뮬레이션 결과에서 큰 편차는 보이지 않으나 각 카테고리에 대한 문서의 검색비율이 일정한 경향을 띄는 것을 알 수 있다. 동그라미 부분이 각 카테고리별로 예약어에 따라 분류된 문서들의 개수인데, 실제 결과값의 반영을 위해 사용자의 성향을 세 카테고리에 대한 가중치의 합을 1로 하여 각 카테고리별로 가중치를 두어 구별하였다. 만약 사용자의 성향을 A(Agent) 0.5, N(Neural) 0.3, D(Digital Library) 0.2로 한다면, 사용자는 각 카테고리별 결과에 가중치를 곱한 결과를 학습의 최종 결과값으로 돌려받게 되어 사용자의 컴퓨터에

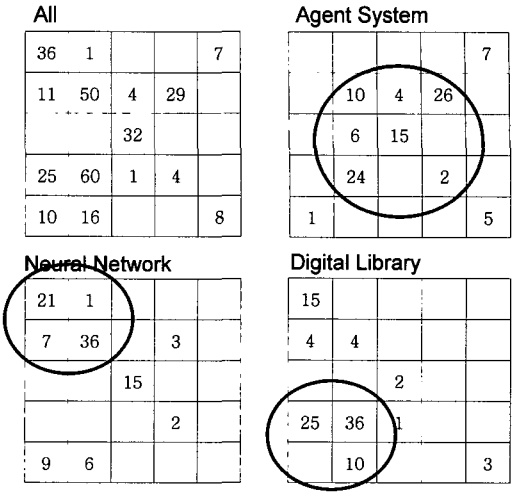


그림 11 카테고리별 SOM의 시뮬레이션 결과

해당 카테고리에 저장될 수 있도록 한다. 그러나 이 가중치들은 해당문서를 열어보았거나 지웠다던지 하는 등의 사용자 액티비티에 의해 값이 계속적으로 변하게 된다. 예를 들어, 사용자의 성향에 따라 특정 문서들의 성향이 위의 예와 같다면, 문서의 최종 카테고리는 가장 큰 가중치를 갖는 쪽으로 분류되어 컴퓨터에 사용자 컴퓨터에 인덱싱되어 저장된다

5.3 시뮬레이션 결과 및 분석

분산 환경에서의 PDS의 효율성을 측정하기 위하여 기존의 단순 클라이언트/서버 기반의 디지털 라이브러리와 모바일 멀티 에이전트 기반의 디지털 라이브러리의 비교를 하였다. 서버의 개수를 달리하여 라이브러리 시스템에서 문서를 검색하여 사용자에게 보여주기까지의 데이타 검색에 걸리는 시간을 측정하였으며, 그때 CPU의 부하와 메모리 점유율을 측정하였다. 측정 조건은 SOM을 이용하여 원격지 디지털 라이브러리 서버의 개

표 1 문서의 각 카테고리별 예약어

카테고리 / 번호	Agent System	Neural Network	Digital Library
1	framework	learning	virtual
2	mobile	neural	library
3	java	architecture	indexing
4	multi	network	structure
5	personalized	simulation	distributed
6	distributed	layer	agent
7	environment	agent	retrieval
8	neural	artificial	autonomous
9	architecture	associative	neural
10	autonomous	algorithm	multi

수를 1개, 3개, ... 10개로 하였을 때이며 그 결과값은 그림 12와 같다.

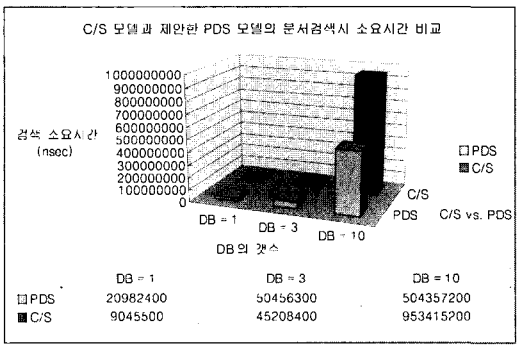


그림 12 C/S 모델과 PDS 모델의 검색 소요시간 비교

데이타베이스가 하나일 경우, 별도의 모듈을 로딩하지 않는 C/S 모델의 검색 결과가 훨씬 시간이 적게 걸림을 알 수 있다. 하지만 위의 그래프에서 알 수 있듯이, 서버의 개수가 증가함에 따라 PDS는 C/S 모델보다 안정성 있는 성능을 보인다. 위의 결과를 살펴보면 데이타베이스의 수가 증가할수록 검색 시간은 일정 비율로 줄어든다는 사실을 알 수 있으며 서버 개수가 3개일 때 거의 같은 시간을 보이고 그 후로 서버 개수가 증가함에 따라 차이가 남을 알 수 있다. 따라서 본 실험 결과 에이전트와 서버개수와의 관계에 따른 검색시간은 에이전트와 서버개수에 반비례함을 알 수 있었다. 이러한 성질은 분산환경에서는 서버의 개수가 증가할 수밖에 없으므로 서버가 증가할수록 검색시간이 단축되는 것이 매우 효율적임을 알 수 있다.

$$\text{검색 시간} \propto \frac{1}{server \text{ 개수}}$$

$$\text{검색 시간} \propto \frac{1}{agent \text{ 개수}}$$

또한 결과의 정확성을 측정하기 위해 SOM에서 수행했던 총 100개의 문서를 동일한 환경에서 실험해 보았는데, 결과 C/S와 PDS 시스템의 결과의 비교는 다음 표 2와 같았다. C/S는 검색된 결과를 살펴본 후 수동으로 분류, 저장한 경우이고 PDS는 자동으로 시스템에서 분류, 저장된 경우를 나타낸다. 실험결과 C/S와 PDS의 검색결과의 정확도는 각각 26%, 98%이므로 PDS에서 약 4배 정도 향상되었다. 검색 후 인덱싱 및 저장시간은 C/S인 경우만 평균 약 3분 정도 소요되었고, 특정기간 동안 검색된 결과 문서에 대한 활용율은 모두 90%이상의 높은 값을 나타내었다. 이로부터 PDS는 전문 문서에 대한 사용자 중심의 검색이 가능하였고 문서의 재활용율도 매우 높으므로 기존 C/S 방식보다 훨씬 효율적임을 알 수 있었다.

한편, 자료 처리하는 동안 CPU와 메모리의 최대 점유율을 비교 측정하기 위해 앞과 동일한 조건에서 서버의 개수를 달리하여 측정한 결과, 서버의 개수가 증가하여도 C/S 기반의 라이브러리는 특별한 변화가 없었으나 PDS는 최대 CPU 점유율이 C/S보다는 다소 높아졌다. 그러나 이것은 PDS 특성상 서버의 개수를 많이 사용해야 하나 제한된 시뮬레이션 환경에서 실험하였기 때문인 것으로 분석되었으므로 실제 환경하에서 실험하게 되면 오히려 효율적일 것으로 분석된다.

또한 멀티에이전트는 자바의 멀티쓰레드로 구현하였으므로 멀티에이전트 생성에 따른 각 에이전트의 생성에서 종료시까지의 평균 반환시간(turnaround time)을 비교하면 다음 그림 13과 같다.

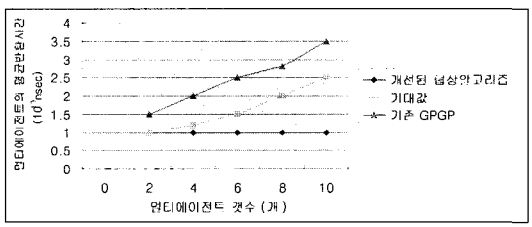


그림 13 기존 GPGP 협상알고리즘과 개선된 협상 알고리즘 비교

에이전트를 늘려가면서 실험한 결과, 개선된 협상알고리즘을 사용할 경우 멀티 에이전트의 평균 반환시간은 거의 변화가 없었고, 기존 GPGP는 에이전트 수에 대략적으로 비례함을 알 수 있었다. 이것은 개선된 협상알고리즘에서는 각 에이전트와 수행하는 메소드의 관계를 미리 정의하여 스케쥴링에 의해 에이전트 생성을 융통적으로 하고 있기 때문이나, 기존 GPGP에서는 미리 에이전트를 생성한 후 실행하면서 에이전트 제어권을 이동하기 때문에 준비큐(ready queue)나 대기큐(waiting

표 2 C/S와 PDS 모델의 시뮬레이션 결과 비교

비교 \ 시스템 모델	C/S 모델	PDS 모델
검색결과의 사용자 정확도(%)	26%	98%
검색 후 인덱싱 및 평균 저장시간(분)	3분	검색과 동시에 자동저장
특정 기간동안 검색된 문서의 활용율(%)	90%	99%

queue)에서 기다리는 시간이 늘어나기 때문이다. 이 표에서 기대값은 DB의 개수를 1에서 10으로 증가시켰을 때 평균적으로 발생하는 값을 의미한다.

6. 결 론

본 논문에서는 원격 프로시저 호출을 이용하는 기존의 클라이언트/서버 기반의 디지털 라이브러리의 단점을 보완한 모바일 멀티 에이전트 플랫폼을 기반으로 한 사용자 중심의 개인 디지털 라이브러리 시스템(PDS)을 구축하였다.

PDS에는 사용자 도서관 서비스를 구현하기 용이한 플랫폼으로서 모듈화된 에이전트 프레임 워크를 설계하였으며, 네트워크 특성에 영향을 거의 받지 않는 검색 시스템일 뿐 아니라 사용자 컴퓨터 내에 로컬 라이브러리를 구축하여 자료 저장의 효율성을 높여준다. 이러한 구조는 기존의 에이전트 시스템이 갖는 모바일이나 멀티 어느 한쪽의 특징을 상호 보완한 것으로 더 나은 성능향상을 시킬 수 있다. 또한 사용자 중심의 라이브러리 구축으로 PDS의 검색 결과는 SOM 네트워크에 의해 분류되어 얻어지며 사용자의 성향이 반영되어 최종 결과를 제공하게 된다. 실험결과 본 논문에서 제안한 PDS는 기존 방식에 비해 약 4배 정도의 사용자의 결과에 대한 정확도를 얻을 수 있었다.

앞으로 디지털 라이브러리 접근의 인증, 보안, 분산 데이타베이스의 효율적인 검색 및 인덱싱 기술, XML 문서로의 확장 저장 등에 대한 기술이 좀더 연구 보완되어야 할 것이다.

참 고 문 헌

[1] Stuart Russel, Peter Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall International Editions, 1995.

[2] John R. Graham, Keith S. Decker, Towards Distributed, Environment Centered Agent Framework. Appearing in Intelligent Agents IV, Agent Theories, Architectures, and Languages Springer-Verlag, 2000, Nicholas Jennings, Yves Lesperance, Editors.

[3] IATLite Homepage : http://java.stanford.edu/

[4] J.Alfred Sánchez, John J.Leggett, John L.Schnase, "AGS: Introducing Agents as Services Provided by Digital Libraries," 2nd ACM International Conference on Digital Libraries, Philadelpia, Penn., July, pp.75-82, 1997.

[5] Jonas Holmstrom, "A Framework for Personalized Library Services," (Internet), October, 2002.

[6] J.Alfred. Sánchez, John J.Leggett, "Agent services for users of digital libraries," Journal of Network and Computer Applications, Vol.20, No.1, pp.45-58, January, 1997.

[7] ObjectSpace Voyager Core Technology 2.0 User Guide. ObjectSpace, 1998.

[8] Peter C.Weinstein, William P.Birmingham, Edmund H.Durfee, "Agent-based Digital Libraries: Decentralization and Coordination," IEEE Communication Magazine, Vol. 37, No. 1, pp.110-115, 1999.

[9] M. Roscheisen, M. Baldonado, K.Chang, L.Gravano, S.Ketchpel, "The Stanford InfoBus and It's Service Layer:Augmenting the Internet with Higher-Level Information Management Protocols," Medoc Dagstuhl Workshop: Electronic Publishing and Digital Libraries in Computer Science, 2003.

[10] J.L.Schnase, D.L.Kama, K.L.Tomlinson, J.A.Sánchez, ElL.Cunnius, "The Flora of North America digital library," A case study in biodiversity database publisging, Journal of Networks and Commputer Applications, Vol.20, No.1, pp.87-103, 1997.

[11] 연구개발정보센타 http://www.dlibrary.go.kr

[12] 국립중앙도서관 http://www.nl.go.kr

[13] Finin T., Fritzson R., McKay D., and McEntire R., "KQML as an agent commu-nication language," Proceedings of CIKM '94, pp.126-130, 1994.

[14] FIPA Agent Management Specifications, http://www.fipa.org/repository/managementspecs.html

[15] Colin G. Harrison, David M. Chess, Aaron Kershenbaum, "Mobile Agents: Are They a Good Idea?," IBM research division, T.J.Watson Research Center, Technical Report, March, 1995.

[16] IBM Japan Aglets http://www.trl.ibm.com/aglets/index_e.htm

[17] AlfInge Wang, Dept. of Computer and Information Science, NTNU Using JavaSpaces to Implement to Mobile Multi-Agent System, 2002.

[18] Keith S. Decker, Victor R. Lessor, "Generalizing the partial global algorithm," Intelligent Cooperative information systems, Vol.1, No.2, pp.319-346. 1992.

[19] John Graham, Real-Time Scheduling for Distributed Agents AAAI-Spring Symposium on Real-Time Autonomous Systems, Palo Alto, CA, March, 2000.

[20] 조영임, 인공지능시스템, pp.98-81, 홍릉과학출판사, 2003.10.

조 영 임

정보과학회논문지 : 소프트웨어 및 응용
제 31 권 제 2 호 참조