바이노미얼 트리를 이용한 이동 에이전트의 빠른 전송

조 수 현† · 김 영 학††

요 약

네트워크 환경이 좋아지고 인터넷 사용이 급증함에 따라 이동 에이전트(Mobile Agent) 기술이 정보검색, 네트워크관리, 전자상거래, 병렬/분산처리 분야에 널리 활용되고 있다. 최근에 다수의 연구자들이 이동 에이전트를 기반으로 한 병렬/분산처리 개념을 연구하고 있다. SPMD(Single Program Multiple Data)는 하나의 프로그램이 병렬환경에 참여하는 모든 컴퓨터에 전송되어 다른 자료를 사용하여 작업을 수행하는 병렬처리 방법이다. 따라서 하나의 프로그램을 모든 컴퓨터에 빠르게 전송하는 것은 전체 수행시간을 줄이기 위한 주요한 요소 중의 하나이다. 본 논문에서는 이동 에이전트 시스템으로 구성된 병렬환경에서 SPMD의 병렬처리를 효율적으로 수행하기 위해, 바이노미얼 트리를 이용하여 하나의 이동 에이전트 코드를 모든 컴퓨터에 빠르게 전송하는 새로운 방법을 제안한다. 제안된 방법은 IBM's Aglets에서 실험적 평가를 통하여 다른 방법과 비교되었으며 다른 방법에 비해서 상당히 좋은 성능을 보였다. 또한 본 문에서는 바이노미얼 트리에서 에이전트 전송 중에 발생될 수 있는 결함허용에 관한 문제를 다룬다.

A Fast Transmission of Mobile Agents Using Binomial Trees

Soo-Hyun Cho† · Young-Hak Kim††

ABSTRACT

As network environments have been improved and the use of internet has been increased, mobile agent technologies are widely used in the fields of information retrieval, network management, electronic commerce, and parallel/distributed processing. Recently, a lot of researchers have studied the concepts of parallel/distributed processing based on mobile agents. SPMD is the parallel processing method which transmits a program to all the computers participated in parallel environment, and performs a work with different data. Therefore, to transmit fast a program to all the computers is one of important factors to reduce total execution time. In this paper, we consider the parallel environment consisting of mobile agents system, and propose a new method which transmits fast a mobile agent code to all the computers using binomial trees in order to efficiently perform the SPMD parallel processing. The proposed method is compared with another ones through experimental evaluation on the IBM's Aglets, and gets greatly better performance. Also this paper deals with fault tolerances which can be occurred in transmitting a mobile agent using binomial trees.

키워드 : 바이노미얼 트리(Binomial Tree), 이동 에이전트(Mobile Agent), 마스터/슬레이브(Master/Slave), 완전이진 트리(Complete Binary Tree), 결함 허용(Fault-tolerance)

1. 서 론

최근에 인터넷 기술이 발달함에 따라 다양한 사용자들의 요구를 만족시키기 위해 이동 에이전트 기술이 여러 분야에서 연구되고 있다. 이동 에이전트는 수행되는 범위가 한 컴퓨터에 제한되지 않으며, 스스로 네트워크에 연결된 다른 컴퓨터로 이동할 수 있다[1-3]. 즉, 에이전트 코드 스스로가 한 컴퓨터에서 다른 컴퓨터로 이동할 수 있는 능력을 갖는다. 또한 비연결성(Non-connectivity) 특성으로 인해 에이전트 이동과 결과 전송 시에만 네트워크 연결이 이루어지므로, 네트워크 부하와 트래픽을 감소시킨다.

최근에 이러한 장점들을 이용하여 이동 에이전트 기술이 정보검색, 전자상거래, 병렬/분산처리, 네트워크 관리 및 보안 등의 응용분야에 널리 이용되고 있다. 이동 에이전트를 이용한 병렬/분산 시스템에서 전체 수행시간에 영향을 미치는 요소는 다음과 같다.

- 이동 에이전트의 크기
- 이동 에이전트의 데이터(전송 데이터 양)
- 이동 에이전트의 전송 시 네트워크 상태
- 효율적인 이동 에이전트의 전송 방법

현재 자바를 기반으로 하는 다수의 이동 에이전트 시스템들이 개발되었으며, 대표적으로 IBM's Aglets Software De-

※ 본 연구는 2001년도 금오공과대학교 학술연구비 지원에 의해서 연구되었음.
† 준 회 원 : 금오공과대학교 대학원 컴퓨터공학과
†† 종신회원 : 금오공과대학교 컴퓨터공학부 교수
논문접수 : 2002년 1월 25일, 심사완료 : 2002년 9월 4일

velopment Kit(이하 Aglets)[1, 2], Odyssey[4], Concordia[5], Voyager[6] 등이 널리 알려져 있다. 최근에 많은 연구자들이 이동 에이전트를 기반으로 한 병렬/분산처리 방법을 연구하고 있다[3, 7, 8]. 이동 에이전트를 기반으로 하는 가상 병렬처리 환경에서 하나의 작업을 다수의 컴퓨터에서 병렬수행 하기 위해 SPMD(Single Program Multiple Data) 방법이 가장 널리 사용되고 있다. SPMD 방법은 마스터 역할을 하는 컴퓨터가 자신이 소유한 하나의 프로그램을 병렬환경에 참여하는 모든 컴퓨터에 전송한다. 그러면 병렬환경에 참여하는 각 컴퓨터는 서로 다른 자료를 사용하여 같은 프로그램을 병렬로 수행하여 주어진 작업을 효율적으로 처리할 수 있다. 따라서 SPMD 방식을 사용하는 병렬처리 환경에서 하나의 프로그램을 모든 컴퓨터에 빠르게 전송하는 것은 전체 수행시간을 줄이기 위한 주요한 요소 중의 하나이다. 그러나 현재 이동 에이전트를 기반으로 하는 병렬처리 환경에서 하나의 이동 에이전트 코드를 참여하는 모든 컴퓨터에 빠르게 전송하는 방법은 활발한 연구가 진행되지 않고 있다.

본 논문에서는 이동 에이전트 시스템으로 구성된 병렬환경에서 SPMD 형식의 병렬처리를 효율적으로 수행하기 위해, 바이노미얼 트리(Binomial Tree)를 이용하여 하나의 이동 에이전트 코드를 모든 컴퓨터에 빠르게 전송하는 새로운 방법을 제안한다. 제안된 방법은 IBM's Aglets에서 실험적 평가를 통하여 비교된다. 현재 에이전트를 기반으로 하는 병렬 컴퓨팅 환경에 하나의 에이전트 코드를 모든 컴퓨터에 전송하기 위해 대표적으로 마스터/슬레이브 방식이 사용되며, 좀더 효율적인 전송을 위해 완전이진 트리 개념을 사용한 방법들이 알려져 있다[9].

본 연구의 실험적 평가에 의하면 바이노미얼 트리를 사용한 방법이 기존에 알려진 방법에 비해서 상당히 좋은 결과를 보임을 알 수 있다. 또한 본 연구에서는 성능이 우수한 바이노미미얼 트리를 이용한 에이전트 코드 전송중에 발생할 수 있는 오류 허용 문제를 다루고, 각 오류에 관한 해결책을 제안한다. 본 연구에서 실험적 고찰을 위해 네트워크 상의 컴퓨터들을 단순하게 바이노미이얼 트리로 구성하였으나, 실제 응용영역에서는 간단한 알고리즘을 사용하여 동적으로 구성될 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구를 개괄적으로 설명하며, 3장에서는 바이노미얼 트리를 이용한 에이전트 전송 모델과 이론적인 환경에서 몇 개의 다른 방법과 에이전트 전송시간을 비교한다. 4장에서는 실제환경에서 제안된 방법의 실험결과를 평가 및 분석하고, 5장에서는 바이노미얼 트리 환경에서 결함허용 문제에 대한 해결 방법들을 살펴본다. 마지막으로 6장에서 결론 및 향후 연구과제를 설명한다.

2. 관련 연구

본 논문에서는 기존의 네트워크에 연결된 컴퓨터들로 구성된 병렬 컴퓨팅 환경에서 SPMD의 병렬처리를 효율적으로 수행하기 위해, 하나의 프로그램을 병렬환경에 참여하는 모든 컴퓨터에 빠르게 전송하는 방법을 연구한다. 이러한 연구의 배경은 병렬처리 분야에서 널리 활용되는 MPI[10] 라이브러리의 집단화 통신(Collective Communication) 방법을 근거로 하고 있다. MPI에서 집단화 통신은 병렬 컴퓨팅 환경에 참여하는 컴퓨터들간에 메시지를 효율적으로 전송하기 위해 사용된다[11-14]. 그러나 본 논문에서는 이동 에이전트 환경의 병렬 컴퓨팅을 고려하며, 이러한 환경에서 메시지가 아니라 에이전트 코드를 어떻게 하면 빠르게 모든 컴퓨터에 전송할 수 있는가에 초점을 두어 연구를 진행한다.

MPI 라이브러리에서 집단화 통신은 서로 다른 컴퓨터들간에 통신을 수행하기 위해 주로 마스터/슬레이브 방식을 이용한다. 그러나 마스터/슬레이브 방법은 구조가 단순하다는 장점이 있지만, 계산할 작업량이 많고 이동 에이전트 시스템에 참여하는 컴퓨터들의 수가 일정 크기 이상 증가할 경우 마스터 컴퓨터에 부하가 집중되는 단점을 갖는다.

[15]에서는 몇 개의 분산 시스템 모델에 대한 수행방식과 특징을 비교하였으며, 응용 프로그램이 수행하는 동안 네트워크 상에 전송되는 총 데이터 양과 네트워크 트래픽을 수식으로 평가하였다. 에이전트를 이용한 결함 허용 방법에 관하여 [16]에서는 이동 에이전트가 이주할 때 발생하는 컴퓨터 혹은 호스트의 결함에 대처하기 위한 이주 정책이 제시되었다. 즉, 결함이 발생한 컴퓨터 혹은 호스트에 대해 이동 에이전트의 이주를 보장하는 경로 재조정과 후위 복구 기법을 통한 신뢰성 있는 이주정책을 제안하였다. [17]에서는 이동 에이전트에 대한 고아(Orphan) 컴퓨터 찾기와 성공적인 종료를 위한 그림자(Shadow) 프로토콜이 제안되었다.

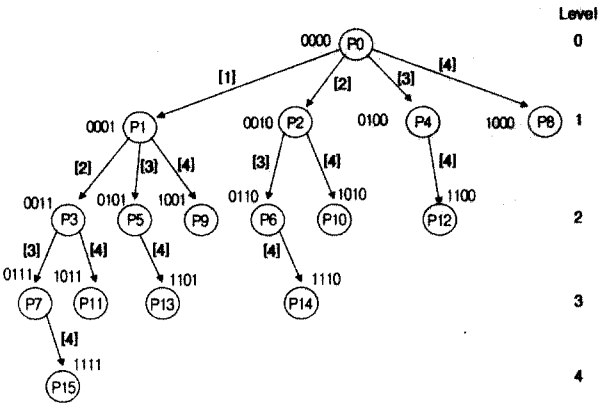
[18]에서는 데이터 마이닝을 위한 이동 에이전트의 효율적인 이주 전략 알고리즘이 제시되었다. 알고리즘 평가를 위해 이동 에이전트, RPC, locker 패러다임을 적용하였으며, 네트워크 상에서 전송되는 총 데이터 양을 기준으로 비교하기보다, 각각의 컴퓨터에서 발생하는 네트워크 소요시간을 수식화 하였다. 이 수식을 기초로 데이터 마이닝 및 실제 분산 응용 시스템 개발시에 적합한 수행 패러다임 결정을 위한 지침서를 제시하였다.

3. 새로운 에이전트 전송모델

바이노미얼 트리는 [19-22]에서 소개되었으며 MPI 라이브러리의 집단화 통신 혹은 병렬컴퓨터에서 멀티캐스팅 분야 등에 응용되고 있다. 본 논문에서는 기존의 네트워크 환경에

서 연결된 컴퓨터로 구성된 이동 에이전트 기반의 병렬 컴퓨팅 환경을 고려하며, 이러한 환경에서 SPMD의 병렬처리를 효율적으로 수행하기 위해 한 에이전트 코드를 모든 컴퓨터에 빠르게 전송하는 방법을 연구한다. 본 장에서는 이러한 문제를 해결하기 위해 바이노미얼 트리 개념을 에이전트 전송을 위한 새로운 모델로 적용한다. 또한 제안된 방법과 비교될 평가 모델들을 개략적으로 설명한다.

3.1 바이노미얼 트리



(그림 1) 4비트의 형식을 갖는 바이노미얼 트리의 예

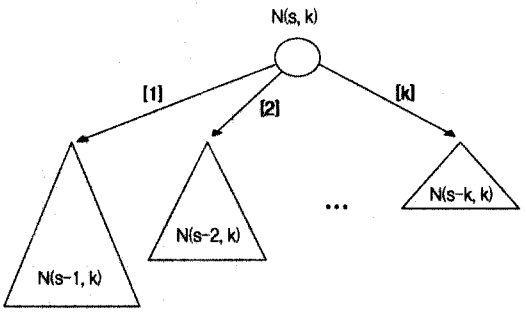
이제 본 논문에서 에이전트 전송을 위한 기본 아이디어로 사용된 바이노미얼 트리의 개념을 소개한다. (그림 1)은 4 비트(Bit) 16개의 노드로 구성된 스패닝 바이노미얼 트리(이하 바이노미얼 트리)를 보여준다. (그림 1)에서와 같이 루트 노드는 레벨 0에 위치되어 있으며, 자식 노드들은 레벨 2, 레벨 3, 레벨 4에 있다. 바이노미얼 트리에서 각 노드의 번호는 이진형식으로 표현되며, 자식 노드는 부모 노드 비트를 물려받은 후 최상위 비트 방향으로 서로 다른 한 비트의 값을 갖는다. 예로서, 노드 0010(2)의 값을 갖는 자식 노드는 0110(6)과 1010(10)이 된다.

(그림 1)의 경우 참여하는 노드의 수는 루트 노드를 포함하여 16개이고, 각 노드에 대한 차수(Degree)는 0에서 4까지로 구성된다. 즉 어떤 노드는 차수가 4일수 있고, 어떤 노드는 차수가 3, 2, 1(또는 0)일수 있다. 일반적으로 n비트의 바이노미얼 트리는 2^n개의 노드를 가지며, 한 노드는 차수가 n이고 다른 한 노드는 차수가 n-1이 되고 그 밖의 노드들의 차수는 $n-i(1 < i \leq n)$가 된다. (그림 1)의 트리 관점에서 노드는 병렬 컴퓨팅에 참여하는 하나의 컴퓨터를 의미한다.

본 논문에서는 이동 에이전트 환경에 참여하는 컴퓨터들을 바이노미얼 트리 형태로 구성하고, 루트 노드에 해당하는 컴퓨터가 에이전트 코드 전송을 시작하여 각 레벨에 따라 에이전트를 병렬 적으로 전송하는 것을 기본 아이디어로 한다. 예로서, (그림 1)에서 루트 노드 P0가 전송될 최초의 에이전트 코드를 가지고 있으며 이 에이전트를 노드 P1에 전송한다. 다음에 노드 P0와 P1은 각각 P2와 P3에게 에이전트를 전송하고, 다음 단계에서는 노드 P3은 노드 P7, 노드 P1은 노드 P5, 노드 P2와 P0는 각각 노드 P6과 P4에게 에이전트를 전송한다. 만일 병렬 컴퓨팅에 참여하는 전체 컴퓨터의 수가 N이라 가정하면 이러한 단계는 logN번이 수행되며, 이러한 과정이 완료되면 모든 컴퓨터는 같은 에이전트 코드를 갖게 된다. (그림 1)의 각 노드에서 [숫자] 형식의 표현은 에이전트 전송이 이루어지는 단계의 순서를 의미한다.

(그림 2)와 같이 루트 노드를 기준으로 서브 트리의 수를 k라 하고 k비트 바이노미얼 트리의 깊이를 s라고 가정하면, 아래의 식에 따라 전체 노드의 수를 계산할 수 있다[20]. 여기서 N(s, k)는 k비트 바이노미얼 트리에 의해 s 단계에서 루트 노드를 포함한 전체 노드 수를 의미한다.

$$N(s,k) = \begin{cases} 2^s & \text{조건 : } s \leq k \\ 1 + N(s-1,k) + N(s-2,k) + \cdots + N(s-k,k) & \text{조건 : } s > k \end{cases}$$



(그림 2) s > k 일 때, s 단계에서 k비트 바이노미얼 트리에 참가한 노드 수

위의 식에서 조건이 $s \leq k$ 일 때, 전체 노드 수는 $N(s, k) = 2^s$가 되고, $s > k$이면 (그림 2)와 같은 트리가 구성된다. 루트 노드는 k 개의 서브 트리를 가지고 각 서브 트리 자신도 k 비트 바이노미얼 트리를 구성한다. 그러므로 k비트 바이노미얼 트리에 참여하는 전체 노드 수를 N(s, k)라 하면 첫 번째 서브 트리에 참여하는 노드 수는 N(s-1, k)가 되고, 두 번째 서브 트리에 대해서는 N(s-2, k)가 된다. 따라서 N(s, k)는 각 서브 트리에 참여하는 노드 수의 합에다 루트 노드를 합한 것이 된다.

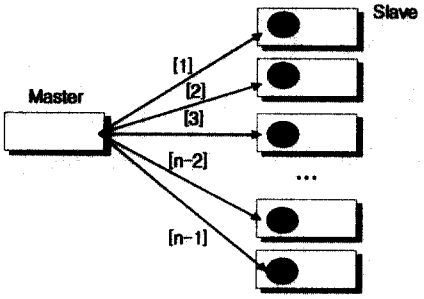
3.2 비교 평가될 전송모델

대표적인 에이전트 전송 모델로는 선형 배열 혹은 마스터/슬레이브 형식을 고려할 수 있다. 선형 배열에 의한 에이전트 전송은 전체 컴퓨터 수에 비례한 시간을 필요로 하기 때문에 이 방법은 비교 대상으로 고려하지 않고, 주로 마스터/슬레이브 방식에 의한 에이전트 전송을 제안된 방법과 비교한다. 또

한 바이노미얼 트리와 유사한 모형을 갖는 완전이진 트리 방식을 비교 대상으로 하여 두 결과를 비교한다.

3.2.1 마스터/슬레이브 방식

병렬 컴퓨팅 환경에 참여하는 컴퓨터들은 하나의 마스터와 다수의 슬레이브들로 구성된다. 즉, 마스터/슬레이브 방식은 마스터가 자신이 갖는 에이전트를 순서적으로 여러 슬레이브에게 전송하는 방법이다[1, 2]. 예로서 (그림 3)에서 마스터 컴퓨터가 각 슬레이브로 에이전트를 순서적으로 보내면, 각 슬레이브는 전송 받은 에이전트를 독립적으로 수행할 수 있으며 자신이 수행한 결과를 다시 마스터 컴퓨터에게 전송하게 된다.



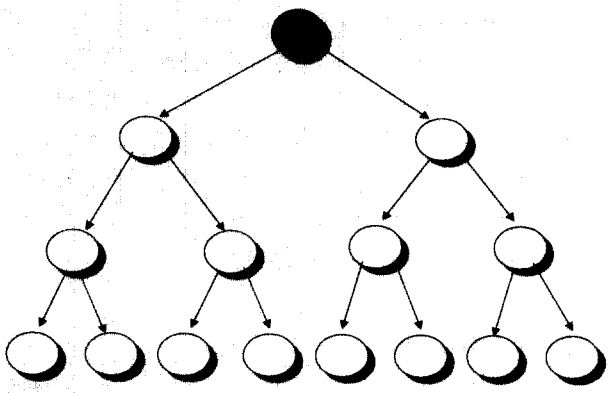
(그림 3) 마스터/슬레이브

이동 에이전트 환경에 참여하는 컴퓨터의 수가 N이라면 마스터 컴퓨터는 전체 컴퓨터로 에이전트를 보내기 위해 N-1번의 에이전트 전송을 수행하게 된다. 따라서 마스터 컴퓨터가 각 슬레이브에게 순서적으로 에이전트를 전송하기 때문에, 참여하는 컴퓨터의 수가 많을 경우 에이전트 전송을 위한 많은 시간이 소모된다.

3.2.2 완전이진 트리 방식

다음은 바이노미얼 트리에 의한 에이전트 전송 비교 관점으로 완전이진 트리를 이용한 에이전트 전송방법을 고려한다. 에이전트 시스템에 참여하는 컴퓨터의 수를 N, 깊이를 k라고 정의하면 완전이진 트리는 최대 2^k-1 개의 컴퓨터를 갖는다((그림 4) 참조). 본 논문에서 완전이진 트리는 일반성을 상실하지 않고 정확히 최대 2^k-1 개의 컴퓨터를 갖는 경우를 고려한다. 완전이진 트리는 모든 컴퓨터의 차수가 2이하를 갖기 때문에 바이노미얼 트리에 비해서 구현이 간단하다는 장점을 갖는다.

완전이진 트리에 의한 에이전트 전송 방법은 각 컴퓨터의 자식이 최대 2개라는 점을 제외하고는 바이노미얼 트리의 에이전트 전송 방식과 동일하다. 또한 각 단계에 참여하는 컴퓨터들이 병렬 적으로 에이전트를 전송할 수 있으며, 참여하는 컴퓨터의 수가 N이라 가정하면 logN 단계 후에 모든 컴퓨터가 에이전트를 갖게 된다.



(그림 4) 완전이진 트리

3.3 동일한 환경에서의 이론적 평가

먼저 이론적인 평가를 위해 에이전트 시스템에 참여하는 모든 컴퓨터가 같은 성능을 갖는 컴퓨터로 구성되어 있으며, 네트워크 상에 발생하는 대기시간(Latency)은 일정하다고 가정한다. 또한 네트워크 상에서 발생되는 트래픽은 고려하지 않는다. 그러나 실제 환경에서 에이전트 시스템에 참여하는 컴퓨터는 서로 다른 성능을 가질 수 있고, 여러 가지의 네트워크 환경 요인에 따라 에이전트 전송시간이 달라질 수 있다. 따라서 실제 환경의 모든 요인을 고려하여 이론적인 평가를 수행하는 것은 쉽지가 않다.

실제 환경을 고려한 실험적인 평가는 다음 장에서 설명하고 본 절에서는 위에서 기술한 가정 하에 본 논문에서 제안한 바이노미얼 트리를 이용한 전송모델을 마스터/슬레이브 방법과 완전이진 트리 방법을 이론적으로 비교한다. 평가를 위해 에이전트 전송에 사용되는 몇 개의 용어를 다음과 같이 정의한다.

- 에이전트 전송 대기시간(t_{send}) : 에이전트를 보내는 컴퓨터에서 에이전트 전송을 위해 준비되는 소프트웨어적인 대기시간을 의미한다.
- 에이전트 수신 대기시간(t_{recv}) : 수신 컴퓨터에서 에이전트를 전송 받기 위한 소프트웨어적인 오버헤드를 의미한다.
- 네트워크 대기시간(t_{net}) : 네트워크를 통해 에이전트가 전송되기 위해 대기되는 시간을 의미한다. 일반적으로 네트워크의 대역폭(Bandwidth), 스위칭 기법(Switching Mechanism), 블록킹 시간(Blocking Time) 등의 요소들이 에이전트 전송시간에 중요한 영향을 주게된다.
- 연속적인 에이전트 전송 간격시간(t_{hold}) : 에이전트 전송에 있어 다음 에이전트 전송을 위해 소요되는 대기시간을 의미한다.

다음은 세 가지 전송모델의 비교를 위해 에이전트 전송에 필요한 기본적인 상수 값을 정의한다. 이러한 상수는 단지 제한

된 환경에서 이론적인 평가를 위해 정의된 값임을 기억한다.

$t_{send} = 2(ms)$, $t_{net} = 1(ms)$, $t_{recv} = 2(ms)$, $t_{hold} = 2(ms)$

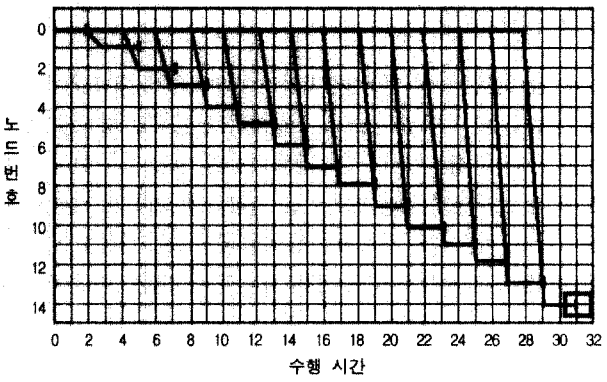
에이전트의 크기에 따라 실제 전송시간은 달라질 수 있지만 여기에서는 같은 크기를 갖는 경우를 적용하기 때문에, 실제 에이전트의 전송시간은 고려하지 않고 전체적인 대기시간만을 고려한다. 분석을 위해 15개의 컴퓨터가 에이전트 시스템에 참여한다고 가정한다.

〈표 1〉 세 가지 전송모델의 에이전트 전송 시 대기시간 비교 (시간 : ms)

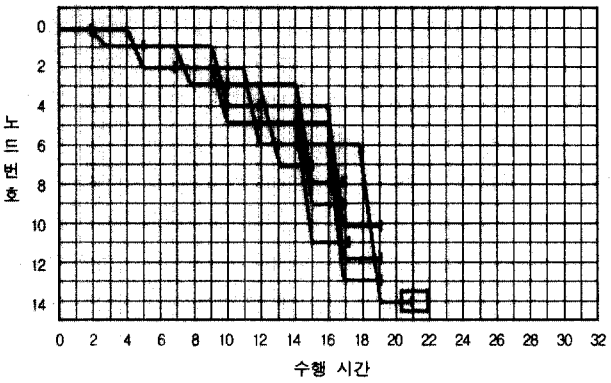
	마스터/슬레이브	완전이진 트리	바이노미얼 트리
P0	2	2	2
P1	5	5	5
P2	7	7	7
P3	9	10	9
P4	11	12	11
P5	13	12	10
P6	15	14	12
P7	17	15	14
P8	19	17	12
P9	21	17	14
P10	23	19	14
P11	25	17	15
P12	27	19	17
P13	29	19	17
P14	31	21	17

<표 1>은 이러한 가정 하에서 세 가지 전송모델의 에이전트 전송시 각 컴퓨터에 전송되는 에이전트의 대기시간을 보여준다. (그림 5), (그림 6), (그림 7)은 <표 1>의 자료를 이용하여 각 컴퓨터에 에이전트가 전달되는 대기시간을 시간-노드(컴퓨터)수 다이어그램으로 나타내었다.

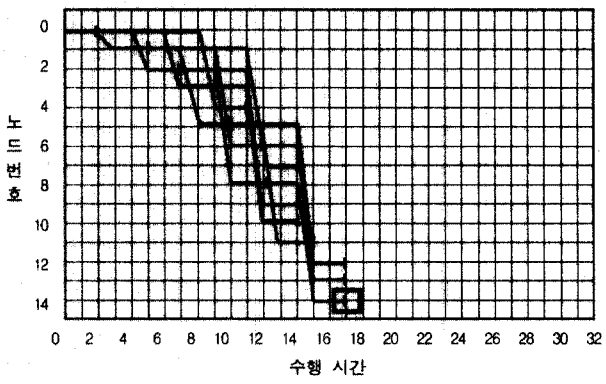
(그림 5)는 위에서 정의된 상수 값을 마스터/슬레이브 전송 모델에 적용한 결과를 보여준다. 일정한 전송 대기시간(2ms)씩 증가하여 결국 31ms 일때 모든 슬레이브 컴퓨터들이 에이전트를 수신하게 된다. (그림 6)은 완전이진 트리 전송모델에 대한 이론적인 에이전트 전송 대기시간의 결과이다. 매 단계마다 2개의 자식 컴퓨터에 병렬 적으로 에이전트를 전송하여 21ms가 되었을 때 모든 컴퓨터들에게 에이전트가 전송되어진다. (그림 7)은 4비트 바이노미얼 트리 전송모델에 대한 전송 대기시간을 나타내고 있다. <표 1>을 통해 알 수 있듯이 몇 번의 단계까지는 완전이진 트리 방법과의 별다른 성능 차이가 없지만 단계 수가 증가할수록 점차적으로 차이를 보이며 결국 17ms가 되면 모든 컴퓨터들이 에이전트를 수신하게 된다.



(그림 5) 마스터/슬레이브



(그림 6) 완전이진 트리



(그림 7) 바이노미얼 트리

위의 세 가지 전송모델의 이론적인 결과에서 알 수 있듯이 마스터/슬레이브 방법이 가장 저조한 성능을 보였는데, 이는 하나의 컴퓨터에서 나머지 컴퓨터에게 순차적으로 에이전트 전송을 담당하기 때문이다. 완전이진 트리와 바이노미얼 트리 방법에서는 참여하는 컴퓨터 수가 증가할수록 바이노미얼 트리 방법의 성능이 우수함을 알 수 있다. 이는 완전이진 트리 방법의 전송경로가 매번 일정하게 두 갈래로 전송되는 것에 비해 바이노미얼 트리는 같은 단계 수를 가지지만 매 단계에서의 전송 경로수가 한 개 이상이기에 컴퓨터 수 증가에 따라 보다 많은 성능 차이를 예상 할 수 있다.

4. 성능 평가

본 장에서는 앞서 언급한 세 가지 전송모델에 대해 컴퓨터들을 성능순 및 무작위 순으로 구성하여 실험적인 평가를 한다. 실제 인터넷 환경에서는 각 컴퓨터들의 성능이 다양하기 때문에 본 논문의 실험에서도 이를 고려하여, CPU의 성능과 메모리의 용량이 다른 컴퓨터들을 실험대상으로 선정하였다. 또한 실험 결과치 값은 10회 반복 결과의 평균값을 사용하였다.

4.1 평가 방법

본 논문에서는 이동 에이전트를 참여한 컴퓨터로 보다 빨리 전송하기 위해 마스터/슬레이브, 완전이진 트리, 바이노미얼 트리 전송 모델들을 고려한다. 또한 각각의 방법에 컴퓨터들의 성능 순 및 무작위 순에 따라 실험적인 전송시간을 평가한다. 마스터/슬레이브 전송 모델에서 컴퓨터들을 무작위로 구성하였을 때의 성능평가는 이미 벤치마킹한 컴퓨터들의 성능 순에 따라 컴퓨터들을 구성한 결과와 평가하였으며, 마스터/루트 컴퓨터는 참여하는 컴퓨터 중 성능이 중간에 해당하는 컴퓨터를 선정하였다. 그것은 마스터/루트 컴퓨터의 성능 여부에 따라 전체 성능평가에 영향을 미칠 수 있는 점을 고려하였기 때문이다. 나머지 슬레이브 컴퓨터들에 대해서는 단지 성능이 우수한 컴퓨터 순으로 구성하여 평가한다.

완전이진 트리 전송 모델에서의 무작위 순 평가방법은 마스터/슬레이브 전송모델의 성능평가 방법과 같으며, 성능순 평가방법에서는 성능이 우수한 컴퓨터 순으로 완전이진 트리를 구성하였다.

바이노미얼 트리 전송모델에서도 무작위순 성능평가 방법은 이전 방법들과 같이 구성하여 평가하였고, 성능 순에서는 우선 벤치마킹한 결과를 기준으로 성능이 우수한 컴퓨터 순으로 구성하고 바이노미얼 트리 구조에 맞게 컴퓨터들을 구성하여 평가하였다. 즉 이전 성능 순 평가방법들은 단지 성능이 우수한 컴퓨터들로 구성된 반면, 바이노미얼 트리 전송모델에서는 자식 컴퓨터들에게 에이전트 전송을 담당하는 컴퓨터들의 성능을 보다 우수한 컴퓨터들로 구성하였다.

마지막으로 세 가지 전송모델에 대해 성능 순으로 구성하였을 때의 결과를 비교 평가하였다. 이는 각 전송모델에 컴퓨터들을 성능 순으로 구성하였을 때 가장 우수한 모델을 알아보고자 하는 것이다.

4.2 실험 환경

성능평가를 위한 실험 환경은 <표 2>와 같다. 실제 인터넷 및 네트워크 환경에 연결된 컴퓨터가 Windows 기반이 대다수를 이루고 있기 때문에, 보다 현실성을 고려하여 Windows의 기반의 컴퓨터를 사용하였다. 그리고 에이전트 시스템은 자바를 기반으로 하는 IBM의 Aglets1.1.b3을 사용하였다.

또한 실제 환경에 참여한 컴퓨터들의 성능이 다양한 형태로 구성되어 있기 때문에, 실험 환경에 참여한 CPU/Memory 구성을 <표 2>와 같이 구성하였다. 이렇게 함으로서 본 논문에서 제안한 세 가지 전송모델에 참여한 컴퓨터들의 성능에 따른 구성이 에이전트 전송에 어떤 영향을 미칠 수 있는지를 고려할 수 있다.

〈표 2〉 실험 환경

운영체제	Windows 95/98/ME
소프트웨어	JDK1.1.8, IBM Aglets1.1.b3
CPU	펜티엄Ⅲ 733, 펜티엄Ⅲ 464, 펜티엄Ⅲ 451, 펜티엄Ⅱ MMX 360, 펜티엄Ⅱ MMX 262, 펜티엄 200
Memory	128M, 96M, 64M, 32M

결함 허용을 고려한 성능평가에서도 같은 환경을 이용하였으며, 실험에 참여하는 컴퓨터들은 10Mbps 이더넷으로 연결된 12대의 PC로 구성하였다.

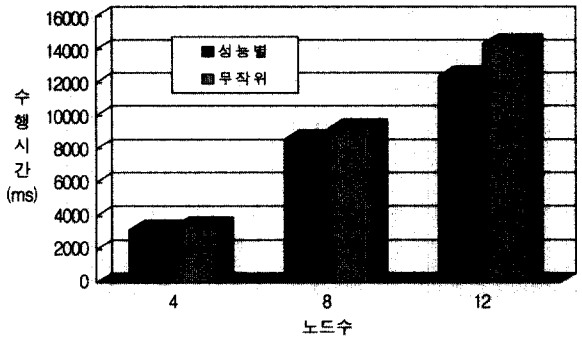
4.3 평가결과 및 분석

(그림 8), (그림 9), (그림 10), (그림 11)은 이전 평가방법들을 기반으로 한 실험적인 결과를 보여준다. (그림 8)에서 마스터/슬레이브 전송모델은 컴퓨터들을 무작위로 구성한 것보다 성능순으로 구성하였을 때 우수함을 알 수 있다. 물론 참여하는 컴퓨터 수가 증가할수록 무작위/성능순간의 성능 차이가 있음을 알 수 있지만 다른 두 가지 전송방법에 비해 성능향상이 저조했다. 즉 다른 전송 모델과 비교해서 참여하는 컴퓨터 수가 증가할수록 무작위/성능순 간의 성능차이가 작다는 것을 보여주고 있다.

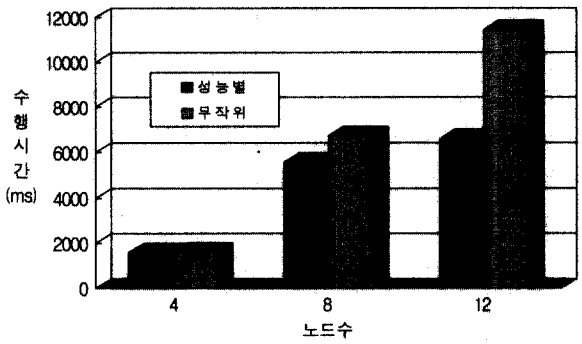
(그림 9)는 완전이진 트리 전송모델의 무작위/성능 순에 대한 실험결과이다. 실험 결과에서 컴퓨터들의 수가 증가할수록 무작위/성능순간의 결과 값의 차이가 커짐을 알 수 있다. 이것은 무작위로 컴퓨터들을 구성했을 경우에는 에이전트 전송을 담당하는 컴퓨터가 전송을 위해 소요되는 시간이 그 만큼 길어지기 때문이다.

바이노미얼 트리 전송모델도 (그림 10)을 통해 알 수 있듯이 참여하는 컴퓨터수가 증가할수록 무작위 구성에 대한 평가 결과와 성능 순에 대한 결과값의 차이가 커짐을 알 수 있고, 이전의 2가지 전송모델보다 더 큰 차이의 결과 값을 나타낸다. 이것은 이전 완전이진 트리 전송방법과 같은 이유이다.

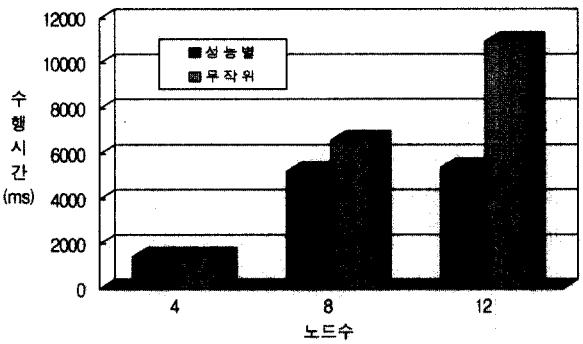
(그림 11)은 컴퓨터들의 성능 순 구성에 대한 세 가지 전송모델의 성능평가 결과를 보여준다. 전체적으로 마스터/슬레이브 모델에 비해서 완전이진 트리와 바이노미얼 트리 전송모델이 좋은 결과를 보였다. 이러한 결과는 에이전트의 분산을 여러 컴퓨터에 위임한 결과로 볼 수 있다. 또한 바이노미얼 트리는 완전이진 트리에 비해서 성능이 다소 우세함을 확인할 수



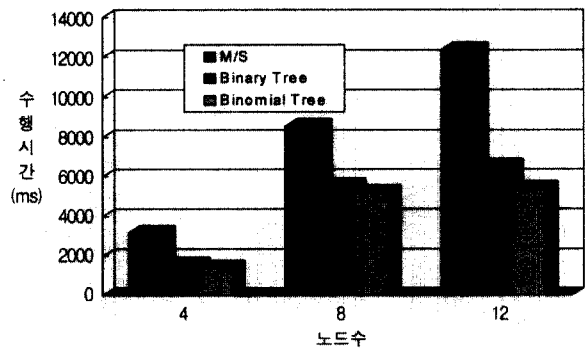
(그림 8) 마스터/슬레이브 전송모델 결과



(그림 9) 완전이진 트리 전송모델 결과



(그림 10) 바이노미얼 트리 전송모델 결과



(그림 11) 성능순 구성에 따른 세가지 전송모델 결과

있다. 이것은 매 단계마다 2개의 자식컴퓨터 보다 다수개의 자식 컴퓨터에게 에이전트를 전송함으로서 각 부모 컴퓨터의 유휴시간(Idle Time)을 줄인 결과로 볼 수 있다. 그리고 이전에 세 가지 전송 모델에 대해 이론적인 에이전트 전송 대기시간에서 보여준 결과와 같은 결과가 나왔음을 알 수 있다.

실험 결과에서 에이전트를 보다 빠르게 참여하는 컴퓨터에게 전송하기 위해서는 전송모델에서 컴퓨터의 구성을 무작위로 배치하기 보다 성능 순으로 구성하여야 함을 알 수 있었다. 또한 완전이진 트리와 바이노미얼 트리 전송모델간의 성능차이에서 알 수 있듯이, 단순히 컴퓨터들을 성능순으로 구성하기 보다 에이전트 전송을 담당하는 컴퓨터를 적절히 성능이 우수한 컴퓨터들로 구성해야 함을 알 수 있다.

5. 결함 허용을 고려한 전송기법

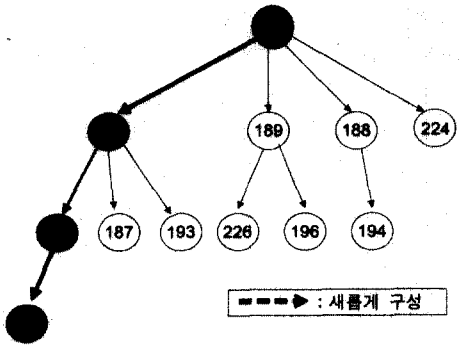
병렬 컴퓨팅 환경에서 때때로 하드웨어 혹은 소프트웨어 결함으로 하나 이상의 컴퓨터에서 결함이 발생할 수 있으며, 이런 경우의 발생은 전체 작업 수행에 큰 지장을 초래할 수 있다. 본 장에서는 4장의 실험적 결과에서 알아본 것과 같이 성능이 가장 좋은 바이노미얼 트리 환경으로 구성된 모델에서 오류 발생 가능성 및 결함 허용 방법들을 다룬다.

5.1 결함 원인 및 해결책

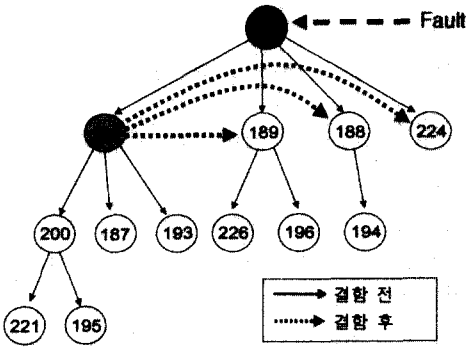
바이노미얼 트리 구조를 이용한 이동 에이전트 전송 시스템에서 컴퓨터들의 결함이 발생 했을 때 결함 컴퓨터로 인한 시스템 전체의 정지 및 부정확한 결과 초래를 막기 위해, 본 논문에서는 이미 구성된 구조를 새롭게 구성하기 보다 결함이 발생한 컴퓨터의 위치에서 이웃/자식 컴퓨터들에게 결함 컴퓨터의 역할을 위임하는 결함허용 정책을 제안한다. 이는 새롭게 바이노미얼 트리를 구성하는데 걸리는 시간을 줄임으로서 결국 전체 수행시간을 단축시킬 수 있다는 장점을 갖는다. 본 논문에서는 바이노미얼 트리 구조에서 발생할 수 있는 결함 원인들을 세 가지(루트 컴퓨터, 부모 컴퓨터, 리프 컴퓨터) 경우로 분류하였으며, 다음에서 각각의 원인 및 해결방법을 살펴본다.

5.1.1 루트 컴퓨터의 결함

자식 중에서 가장 성능이 우수한 컴퓨터가 루트 컴퓨터의 역할을 대신한다. 즉, (그림 12)와 같이 루트 컴퓨터의 결함이 발생할 경우 바이노미얼 트리를 새롭게 재구성하는 것이 아니라, (그림 13)에서 보여준 것과 같이 자식 컴퓨터 중에서 제일 성능이 좋은 컴퓨터(190번)가 이웃하는 컴퓨터들에게 루트가 해야할 일을 대신하게 된다. 다시 말해 190번 컴퓨터는 루트 역할뿐만 아니라 200, 187, 193 컴퓨터의 부모 역할을 수행하게 된다. 이러한 절차는 결함 발생시 새롭게 바이노미얼 트리를 재구성하는데 소요되는 시간을 줄일 수 있으며, 별도의 백업용 루트 컴퓨터를 두는 단점을 보완할 수 있다.

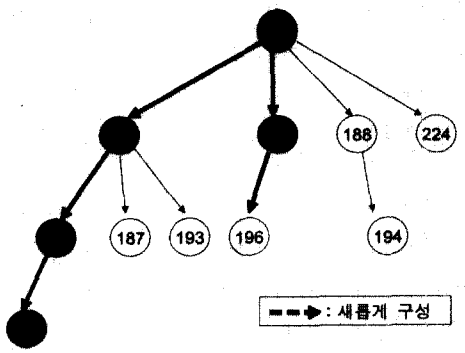


(그림 12) 루트 컴퓨터 결함 시 새롭게 컴퓨터들을 구성한 경우

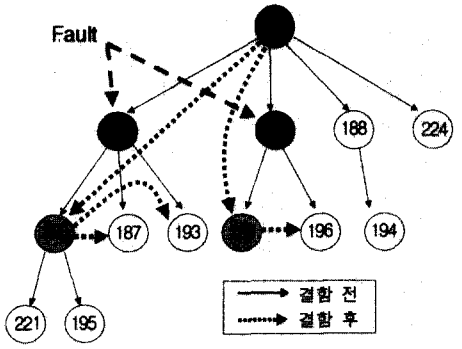


(그림 13) 루트 컴퓨터 결함시 새롭게 컴퓨터들을 구성하지 않은 경우

5.1.2 부모 컴퓨터 결함



(그림 14) 부모 컴퓨터 결함 시 새롭게 컴퓨터들을 구성한 경우



(그림 15) 부모 컴퓨터 결함시 새롭게 컴퓨터들을 구성하지 않은 경우

(그림 15)는 부모 컴퓨터 190, 189번 컴퓨터가 결함이 발생했을 경우를 보여주고 있다. 자식 중 가장 성능이 우수한 컴퓨터가 부모 컴퓨터를 대신하여 이동 에이전트를 전송한다. (그림 15)에서 나머지 컴퓨터들의 구성은 변하지 않는다. 즉, (그림 14)와 같이 200, 226, 221번 컴퓨터들이 자신들의 부모 컴퓨터 위치로 새롭게 구성되는 것이 아니라 200, 226번 컴퓨터만이 결함이 발생한 컴퓨터인 190, 189번 컴퓨터의 역할을 대신한다. 이는 바이노미얼 트리 구조에서 부모 컴퓨터를 기준으로 왼쪽 자식 컴퓨터가 오른쪽 자식 컴퓨터보다 성능이 우수한 컴퓨터들로 구성되어지기 때문에 200, 226번 컴퓨터가 부모의 역할을 대신하게 된다.

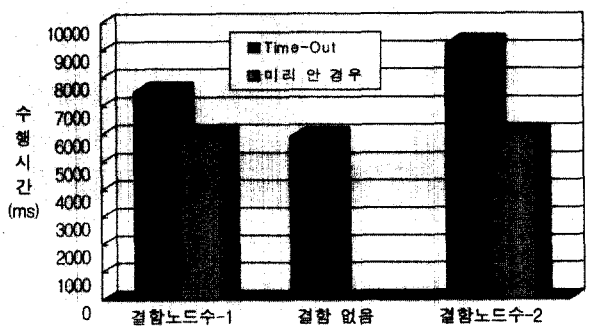
결국 루트 컴퓨터인 186번 컴퓨터는 결함이 발생한 190, 189번 컴퓨터들에게 에이전트를 전송하지 않고 200, 226번 컴퓨터들에게 전송함으로서 결함 문제를 해결하게 된다. 또한 새롭게 부모 컴퓨터가 된 200, 226번 컴퓨터들은 190, 189번 역할뿐만 아니라 계속해서 221, 195번의 부모 컴퓨터 역할도 하게 된다.

5.1.3 리프 컴퓨터 결함

결함이 발생한 컴퓨터의 부모 컴퓨터는 나머지 자식 중 성능이 우수한 컴퓨터에게 다시 이동 에이전트를 전송한다. (그림 13)에서 195번 컴퓨터가 결함일 때 200번 부모 컴퓨터는 221번 컴퓨터에게 전송한다. 만약에 자식 컴퓨터 모두가 결함이 발생했다면 부모 컴퓨터 자신이 작업을 수행하게 된다.

5.2 평가결과 및 분석

(그림 16)은 5.1.2 절에서 설명한 부모 컴퓨터의 결함시에 대한 실험 결과를 보여준다. 첫 번째 실험은 결함 컴퓨터의 수가 1개 일 때 결함 컴퓨터에 일정시간 동안 아무런 응답이 없으면, 자식 중 성능이 우수한 컴퓨터로 이동 에이전트를 전송하여 수행한 결과와 이미 190번 컴퓨터가 결함임을 알고 바로 자식 중 성능이 우수한 컴퓨터로 전송하여 수행한 결과를 보여준다. 두 번째 실험은 참여하는 컴퓨터들이 결함이 없이 정상적으로 수행된 결과이다. 마지막 실험은 결함 컴퓨터 수가 2개 일 때 수행된 결과를 나타낸다.



(그림 16) 결함 허용 수행결과 비교

실험결과에서 결함 컴퓨터에 대한 정보를 미리 알고 있는 경우는 그렇지 않은 경우 보다 빠른 수행시간을 나타내고 있다. 물론 결함 컴퓨터수가 2개인 경우도 같은 결과를 보여주고 있다. 또한 미리 결함 컴퓨터에 대해 알고 있는 경우 결함이 발생하지 않고 정상적으로 수행한 경우와 거의 같은 시간을 나타내고 있다. 이것은 미리 결함이 발생한 컴퓨터들에 대한 정보를 알고 있을 경우, 결함이 발생한 컴퓨터로부터의 응답을 기다린 후 결함 컴퓨터의 자식 컴퓨터에게 다시 전송하기 위해 소요되는 시간을 줄여 바로 결함 컴퓨터의 자식 컴퓨터에게 전송되기 때문이다.

6. 결론 및 연구과제

최근에 네트워크 환경이 좋아지고 인터넷 사용이 급증함에 따라 이동 에이전트 기술이 정보검색, 네트워크관리, 전자상거래, 병렬/분산처리 등 여러 분야에 활용되고 있다. 본 논문에서는 기존의 네트워크 환경에서 이동 에이전트를 기반으로 하는 병렬처리 분야에서, SPMD형식의 병렬처리를 효율적으로 수행하기 위해 하나의 에이전트 코드를 병렬처리에 참여하는 모든 컴퓨터에 빠르게 전송하는 방법을 연구하였다.

본 연구에서는 이러한 문제를 해결하기 위해 바이노미얼 트리를 이용한 새로운 에이전트 코드 전송방법을 제안하였으며, 이러한 결과는 기존에 알려진 마스터/슬레이브 및 완전이진 트리 방법에 비해서 성능이 우수함을 입증하였다. 또한 바이노미얼 트리 모델 상에서 각 컴퓨터의 결함 발생 가능성과 해결책을 제시하였다.

참 고 문 헌

[1] D. B. Lange and M. Oshima, "Programming and deploying Java Mobile Agents with Aglets," Addison Wesley Press, 1998.

[2] IBM, "The Aglets Workbench," URL : http://www.trl.ibm.co.jp/aglets/.

[3] C. G. Harrison, D. M. Chess, and A. Kershenbaum, "Mobile Agents : Are They a Good Idea?," IBM Watson Research Center, Mar., 1995.

[4] General Magic Odyssey, URL : http://www.genmagic.com/agents/.

[5] Concordia, URL : http://www.meitca.com/HLS/Projects/Concordia/.

[6] Voyager, URL : http://www.objectspace.com/voyager/.

[7] M. Straber, J. Baumann, and M. Schwehm, "An Agent-Based Framework for the Transparent Distribution of Computations," PDPTA, Vol.1, pp.376-382, 1999.

[8] M. Starber and M. Schwehm, "A Performance Model for Mobile Agent Systems," PDPTA, Vol.II, pp.1132-1140, 1997.

[9] B. Wilkinson and C. M. Allen, "Parallel Programming : Techniques and Applications Using Networked Workstations and Parallel Computers," Prentice Hall, 1998.

[10] Message Passing Interface Forum, "MPI-2 : Extensions to the Message-Passing Interface," URL : http://www.mpi-forum.org/docs/mpi-20-html/mpi2-report.html.

[11] A. Afsahi, "Design and Evaluation of Communication Latency Hiding-Reduction Techniques for Message-Passing Environments," Ph. D. Dissertation, University of Victoria, British Columbia, Apr., 2000.

[12] V. Moorthy, D. K. Panda, and P. Sadayappan, "Fast Collective Communication Algorithms for Reflective Memory Network Clusters," CANPC '00, pp.100-114, Jan., 2000.

[13] T. L. Williams, "A General-Purpose Model for Heterogeneous Computation," Ph.D. Dissertation, University of Central Florida, Orlando, Dec., 2000.

[14] R. Rabenseifner and A. E. Koniges, "Effective Communication and File-I/O Bandwidth Benchmarks," Proceedings of the 8th European PVM/MPI Users' Group Meeting, EuroPVM/MPI 2001, pp.24-35, Sep., 2001.

[15] A. Carzaniga, G. P. Picco, and G. Vigna, "Designing Distributed Applications with Mobile Code Paradigms," Proceedings of the 19th International Conference on Software Engineering, Boston, 1997.

[16] 전병국, 최형근, "이동 에이전트를 위한 효율적인 이주정책의 설계 및 구현", 정보처리학회논문지, 제6권 제7호, Jul., 1999.

[17] J. Baumann, "A Protocol for Orphan Detection and Termination in Mobile Agent Systems," TR-1997-09, Stuttgart University, Jul., 1997.

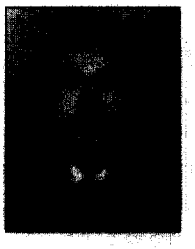
[18] 권혁찬, 유우종, 김홍환, 유관종, "데이터 마이닝을 위한 이동 에이전트의 효율적인 이주 전략", 정보처리학회논문지, 제7권 제5호, May, 2000.

[19] S. L. Johnsson and C. T. Ho, "Optimum Broadcasting and Personalized Communication in Hypercubes," IEEE Trans. Computers, Vol.38, No.9, pp.1249-1268, Sep., 1989.

[20] R. Kesavan and D. K. Panda, "Optimal Multicast with Packetization and Network Interface Support," Technical Report OSU-CISRC-2/97-TR10, The Ohio State University, 1996.

[21] N. F. Tzeng and A. Kongmunvattana, "Distributed Shared Memory Systems with Improved Barrier Synchronization and Data Transfer," In Proceedings of the 1997 ACM International Conference on Supercomputing, Vienna, Austria, Jul., 1997.

[22] M. Banikazemi and D. K. Panda, "Efficient Scatter Communication in Wormhole k-ary n-cubes with Multidestination Message Passing," Technical Report OSU-CISRC-9/96-TR46, The Ohio State University, 1996.



조 수 현

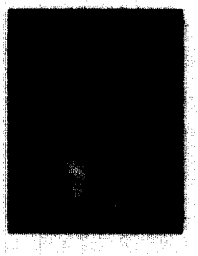
e-mail : shcho@cespc1.kumoh.ac.kr

2000년 금오공과대학교 컴퓨터공학과(학사)

2002년 금오공과대학교 컴퓨터공학과(석사)

2002년~현재 금오공과대학교 컴퓨터공학과 박사과정

관심분야 : 병렬/분산처리, 이동 에이전트, 병렬 프로그래밍, Cluster Computing



김 영 학

e-mail : yhkim@cespc1.kumoh.ac.kr

1984년 금오공과대학교 전자공학과(학사)

1989년 서강대학교 전자계산학과(석사)

1997년 서강대학교 전자계산학과(박사)

1989년~1997년 해군사관학교 전산과학과 교수

1998년~1999년 여수대학교 멀티미디어학부 교수

1999년~현재 금오공과대학교 컴퓨터공학부 조교수

관심분야 : 병렬 알고리즘, 분산 및 병렬처리 등