상태 정보 학습을 이용한 효율적인 이동 에이전트 이주기법

최신일° 엄영현 국윤규 정계동 최영근
광운대학교 컴퓨터 과학과
{karlsen°, class76, ykkook, gdjung, ygchoi}@kw.ac.kr

Efficient Migration Approach of Mobile Agent
using learning state information

Shinil Choi° Young-Hyun Eum, oun-Gyou Kook, Kye-Dong Jung, Young-Keun Choi
Department of Computer Science, KwangWoon University

요 약

네트워크 환경의 발전으로 분산 처리의 필요성이 증가하였고 그에 따라 네트워크 관리, 이동 컴퓨팅 및 정보 수집 등 다양한 분야에서 이동 에이전트의 연구가 활발히 진행되고 있다. 에이전트의 이주에 있어 노드들의 동적인 상태는 이주기법의 성능에 영향을 미친다. 본 논문에서는 노드들의 동적인 상태를 고려한 에이전트의 이주기법을 제안한다. 에이전트의 이주 기법은 바이노미얼 트리 위상을 기반으로 하며, 노드들의 컴퓨팅 능력에 따라 우선순위를 부여하고 우선순위가 높은 노드들부터 바이노미얼 트리 위상의 루트 레벨로부터 하위 레벨로 배치하고 에이전트를 이주시킴으로 효율적이고 안정적인 에이전트의 이주를 하도록 한다.

1. 서 론

네트워크 환경의 발전으로 분산 처리의 필요성이 증가함에 따라 네트워크 관리, 이동 컴퓨팅 및 정보 수집 등 다양한 분야에서 이동 에이전트의 연구가 활발히 진행되고 있다[1][2]. 이동 에이전트는 분산 응용에 있어서 네트워크의 부하 및 대기시간을 줄일 수 있는 기술 중 하나이며, 네트워크 환경에 대한 적응성이 좋다. 그리고 네트워크의 오류 시 다른 시스템에 비해 신뢰도가 높기 때문에[3] 향후 기대되는 분산 응용기반 기술이라 할 수 있다. 그러나 이동에이전트의 성능은 네트워크 및 노드들의 동적인 상태에 의해 영향을 받는다[4]. 기존의 대표적인 이동 에이전트 시스템인 Aglets, Voyager, Odyssey, Tacoma 그리고 Concordia는 분산된 노드들의 동적인 상태나 네트워크 상태를 고려하지 않았다.

이에 본 논문에서는 노드 관리 서버(Node Management Server)를 사용하여 참여하는 노드들의 동적인 상태를 고려하는 이동 에이전트의 효율적인 이주기법을 제안한다.

먼저, 참여를 원하는 노드들은 노드 관리 서버에 등록하기 위해 서버에 대기한다. 이때 서버는 노드들의 컴퓨팅 능력을 측정하고 측정한 데이터를 저장한다. 이 정보를 탐색하여 각 노드에 우선순위를 부여하고 우선순위를 이용하여 바이노미얼 트리를 구성한다. 구성된 바이노미얼 트리를 이용하여 Preprocessing을 수행한다. Preprocessing을 수행함으로써 각 노드들의 컴퓨팅 요소별 최적의 가중치를 구할 수 있기 때문이다. 이렇게 구해진 가중치는 이 후에 참여하게 되는 노드들에게 적용해 최적의 우선순위를 배정할 수 있게 된다.

본 논문은 2장에서 관련연구에 대하여 살펴보고, 3장에서 본 논문에서 제안하고 있는 노드 관리 시스템에 대해 소개하고 4장에서 성능평가에 대해 기술하고 마지막으로 결론 및 향후연구로 구성된다.

2. 관련 연구

2.1 바이노미얼 트리

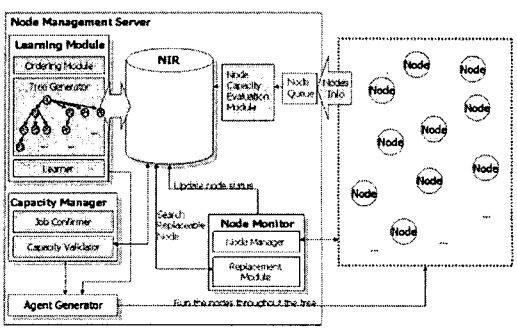
분산 환경에서 이동에이전트의 이동경로는 네트워크 위상을 따른다. 바이노미얼트리 위상은 브로드캐스팅 전송에서 O(log2n)의 시간이 걸린다. 바이노미얼 트리는 계층형 위상으로서 재귀적으로 구성되는 트리이다. 이 위상은 단일 포트 모드에서 최적의 브로드 캐스팅 시간을 갖는 유일한 위상이다[5][6]. 바이노미얼 트리를 이용한 브로드캐스팅 전송은 병렬적으로 에이전트를 전송하므로 수행시간을 효율적으로 절약할 수 있다.

바이노미얼 트리는 재귀적으로 정의되는 순서트리이다. B1은 B0를 재귀하고, B2는 B1, B0를 재귀한다. 이러한 바이노미얼 트리 Bk는 Bk-1, Bk-2 … B2, B1, B0가 차례로 재귀되어 구성한 트리이다. 이 트리 Bk는 2k노드들을 가지며, 트리의 높이는 k이고, 루트는 k차수를 가진다. 또한 바이노미얼 트리에서 어떤 노드의 최대차수는 $\log_2 2k$로서 k가 된다.

3. 학습을 통한 에이전트 이주기법

이 장에서는 노드의 동적인 상태를 고려하여 우선순위를 배정하고 바이노미얼 트리 위상의 상위에 우선순위가 높은 노드를 구성해 에이전트를 이주시킴으로써 전체 성능의 효율성을 향상시키는 이주기법을 제안하였다.

3.1. 시스템 구성



[그림 1] 전체 구성도

이주기법의 전체 구성은 그림 1과 같이 이동 에이전트가 실행될 노드들과 Node Management Server가 있다. Node Management Server에는 노드가 서버에 등록하기 위해 기다리는 Node Queue, 노드들의 성능을 측정하는 Node Capacity Evaluation Module, 노드들의 정보를 저장하는 NIR(Node Information Registry), 노드의 동적 상태를 관리하는 Learning Module, 구성된 노드의 컴퓨팅 파워를 측정하는 Capacity Manager, 에이전트를 생성하는 Agent Generator 그리고 구성된 노드들을 관리하는 Node Monitor로 구성된다.

3.2 이주기법

시스템에 참여하고자 하는 노드들은 시스템에 등록을 하기위해 Node Queue에서 기다린다. 자신의 차례가 오면 NIR에 정보를 입력한다. NIR에 등록하기 위해 필요한 노드의 입력 데이터 구조는 그림 2와 같다.

Node name	CPU	Disk Space	Network Bandwidth	Memory	Job Size	Desire time of complete execution	Priority	Used	Fault

[그림 2] Registry 데이터 구조

노드의 정보 중에 CPU, Disk Space, Memory 그리고 Network Bandwidth는 시스템의 Node Capacity Evaluation Module에서 측정하게 된다. Learning Module에서는 이 정보를 이용해 Preprocessing을 수행한다. Preprocessing은 노드들에 우선순위를 주기위한 절차이다. Learning Module의 Ordering Module은 Registry로부터 노드들의 정보를 얻어와 참여하는 노드들에 우선순위를 주고 Tree Generator에서 그 우선순위를 바탕으로 바이노미얼 트리를 구성한 후 Learner에서 노드들의 컴퓨팅 요소인 CPU, Disk Space, Memory 그리고 Network Bandwidth에 가중치를 부여한다. 가중치 및 학습에 대한 내용은 3.3에서 언급하도록 한다.

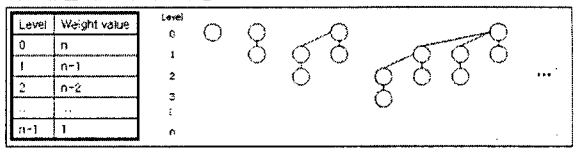
Capacity Manager에서는 노드가 시스템에 참여할 때 입력한 Job의 정보를 이용하여 처리하고자 하는 Job이 무엇인지 판단하고 구성된 노드들이 주어진 Job을 수행할 수 있는지를 평가하는 역할을 한다. Agent Generator의 Job Confirmer에서는 Job의 크기와 Job을 처리하기 위해 필요한 컴퓨팅 파워를 계산하고, Capacity Validator에서는 현재 구성된 시스템의 컴퓨팅 파워를 사용하여 처리하고자 하는 Job을 수행할 수 있는가에 대해 판단을 한다. Agent Generator는 Tree Generator로부터 우선순위와 바이노미얼 트리 구성도를 얻고, Capacity Manager로부터 전체 구성 노드들을 이용하여 잡을 처리하는 데 충분한 컴퓨팅 파워가 있는지의 여부에 대한 정보를 얻어 에이전트를 생성하고 바이노미얼 트리 위상으로 에이전트를 이주한다. 이동 에이전트는 이주를 하면서 주어진 일을 하는 동시에 노드들의 새로운 정보를 가지고 돌아와 노드 매니저의 Registry에 저장한다.

3.3 Preprocessing 및 학습방법

3.3.1 Preprocessing

컴퓨팅 요소들에 따라 적절한 가중치를 주기위해 Learning Module은 누적되는 노드들의 상태정보를 이용해 Preprocessing을 하게 된다. Learning Module은 그림 4와 같이 Ordering Module, Tree Generator 그리고 Learner로 구성된다. Ordering Module은 Registry로부터 노드의 컴퓨팅 요소 값을 얻어 임의의 가중치를 적용하여 노드들에 우선순위를 부여한다. 이를 Tree Generator로 보내고 Tree Generator는 전달된 정보를 이용하여 바이노미얼 트리 위상을 구성한다. 바이노미얼 트리 위상의 최상위 레벨에 있는 노드의 역할이 구성 노드들 중에 가장 많은 기여를 하고 하위 레벨로 갈수록 기여도는 하락하기 때문에 다음 그림 3과 같이 바이노미얼 트리 위상의 레벨에 따라 가중치를 적용한다.

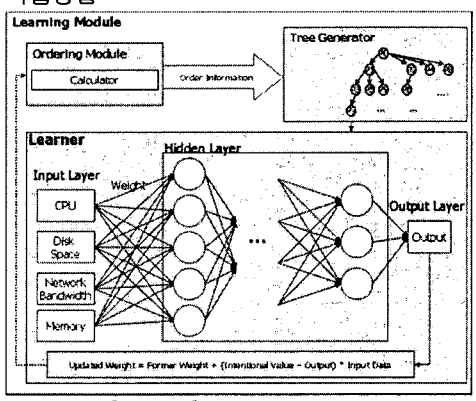
Level	Weight value
0	n
1	n-1
2	n-2
...	...
n-1	1



[그림 3] 트리의 레벨에 따른 가중치 값

모든 노드의 컴퓨팅 요소별 값의 총합을 구하고 이 값을 이용하여 학습을 한다. 이렇게 학습된 가중치 값은 후에 새로운 노드들의 정보, 즉 학습되지 않은 입력에 대해서도 올바른 결과를 출력하게 된다. 그러나 구성노드의 일부분 이상(10%)이 바뀌었을 때는 Preprocessing을 다시 수행하여 최적의 가중치를 갱신하도록 한다. 학습에 대한 내용은 다음 장에서 다루겠다.

3.3.2 학습방법



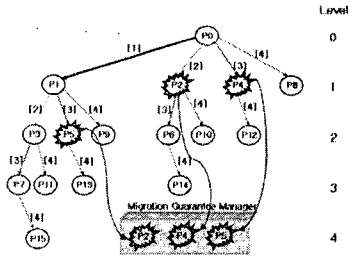
[그림 4] Learning Module

학습방법으로는 신경망 구조의 역전파 알고리즘을 사용했다. Input Layer에는 입력 데이터로서 이동 에이전트가 실행될 각 노드들의 요소별 총합을 입력한다. 최초에는 Learner 자체에서 임의의 가중치를 주어 입력 값과의 곱의 합을 구한다. 이 값은 Hidden Layer를 거쳐 Output Layer에서 Output값으로 나오게 된다. 원하는 목표 값과 Output값의 차이로 얻어진 새로운 가중치와 새로운 노드들의 정보를 가지고 반복하여 Output값을 구한다. 이러한 과정을 Output값과 원하는 목표 값의 차이가 0에 가까워 질 때까지 반복한다. 이 차이가 0에 가까울수록 가중치는 최적으로 수렴한다.

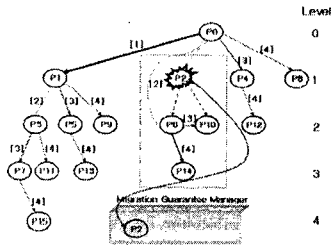
3.4 에이전트의 이주보장

Node Monitor의 Node Manager는 Monitoring Agent를 이용하여 참여하고 있는 노드들의 상태를 실시간 감시한다. 결함의 발견 시간에 따라 두 가지 경우의 수가

있다. 모니터링 에이전트가 먼저 결함을 발견했을 경우(그림 5)와 일을 수행하는 에이전트가 일을 처리하면서 발견했을 경우(그림 6)이다. [7]의 이주 보장 정책을 참조하였다.



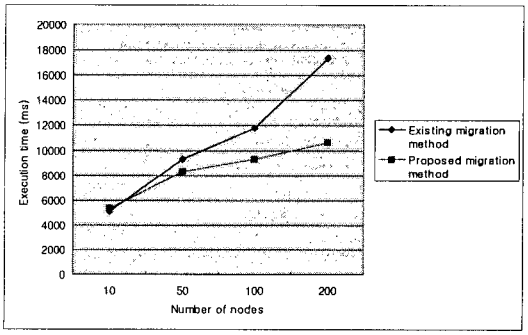
[그림 5] 이주 전 결함 호스트 분석



[그림 6] 이주 중 결함 호스트 재구성

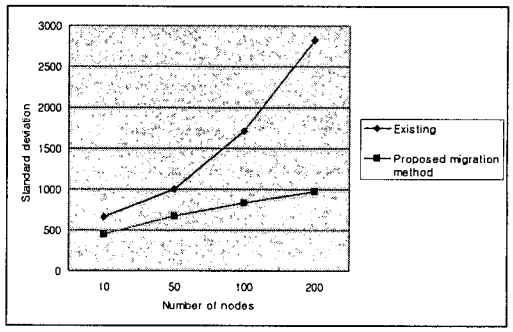
4. 성능평가

노드의 수에 따른 수행시간을 그림 7에서 보인다. 그림 7에서 Existing migration method는 동적 재배치 방법을 사용하지 않은 이주 기법을 나타내며 Proposed migration method는 제안 하는 기법으로서 동적 재배치 방법을 사용한다. 성능평가를 위해 두 기법에 모두 노드의 수를 10, 50, 100, 200개로 정하고 각 각에 대해 이동 에이전트를 10번씩 이주시켜 보았다. 그림 7을 보면 두 기법 모두 노드의 수가 증가할수록 수행시간도 증가한다. 그러나 기존의 이주기법에 비해 제안된 이주기법의 수행시간의 증가 비율이 낮은 것을 알 수 있다. 결국 제안된 이주 기법을 사용하면 이동 에이전트의 이주의 효율성을 높일 수 있게 된다.



[그림 7] 노드의 수에 따른 수행시간

다음으로 그림 8은 노드 수의 증가에 따른 표준편차의 변화이다. 두 기법 모두 노드의 수가 증가 할수록 표준편차의 크기는 커지지만, Existing migration method는 노드의 수가 증가 할수록 표준편차가 더 커진다. 반면에 제안된 기법을 사용하면 일정한 비율로 표준편차의 크기가 증가하는 것을 볼 수 있다. 이 그래프를 보면 제안된 논문이 기존의 논문 보다 안정성의 측면에서도 우수한 성능을 보였다.



[그림 8] 노드의 수에 따른 표준편차

5. 결론 및 향후연구

이동 에이전트는 분산 처리의 필요성이 증가함에 따라 향후 기대되는 분산 응용기반 기술이다. 그러나 에이전트 이주의 성능은 네트워크 및 노드들의 동적인 상태에 영향을 받는다.

본 논문에서는 이러한 문제점을 고려하여 미리 학습된 최적의 가중치를 바탕으로 동적으로 바이노미얼 트리 위상을 재배치하여 에이전트를 이주하는 방법을 제시함으로 에이전트의 이주의 효율성과 안정성을 높였다. 향후 연구 과제로는 Preprocessing과 학습방법의 신뢰성 있는 향상이다.

참고문헌

[1] K. Takashio, G. Seoda, and H. Tokuda, "A Mobile Agent Framework for the Follow-Me Applications in Ubiquitous Computing Environment", Proc. Int'l Workshop Smart Appliances and Wearable Computing(IWSAWC '01), pp.202-207, 2001

[2] Misikangas. P, Raatikainen. K, "Agent migration between incompatible agent platforms", Distributed Computing System, 2000.

[3] D. B. Lange and M. Oshima, "Programming And Deploying Java Mobile Agents with Aglets," Addison Wesley, 1998.

[4] K. Koukoumpetsos and N. Antonopoulos, "Mobility Patterns: An Alternative Approach to Mobility Management", Pro. the 6thworld Multi-Conference on Systemics, pp.14-18, 2002

[5] Demetres Kouvatsos, Is-Haka Mkwawa and Irfan Awan, "One-to-All Broadcasting on the net", Intel Press, pp181-191, 2002.

[6] Jai Eun Jang, "AnOptimal Fault-Tolerant Broadcasting Algorithm for a Cube-Connected Cycles Multiprocessor", IEEE, 1990.

[7] 배명훈, 국윤규, 김운용, 정계동, 최영근 "특성 프로파일을 이용한 이동 에이전트의 효율적인 전송 및 이주 보장 시스템", JCCI 2004