# 이동 에이전트 환경에서 최적화된 작업할당 방법에 대한 연구

김용호° 김영균 오길호
{kimyh, ygkim, gilho}@cespc1.kumoh.ac.kr

# A Study of Optimized Task Allocation Method using Mobile Agents

Yong-Ho Kim° Young-Gyun Kim Gil-Ho Oh
School of Computer Engineering, Kumoh National University of Technology

요 약

인터넷 환경에서 유휴 컴퓨팅 자원들을 활용하여 병렬 작업들을 분산 처리하는 많은 연구들이 선행되어왔다. 기존의 방법들은 하나의 작업을 여러 개의 작은 단위로 나누어 각각을 다수의 노드에서 분산 처리하는 방식으로 분해할 수 없는 다수의 작업에 대한 고려가 되어 있지 않다. 이러한 방법은 각 노드의 성능과 자원에 대한 고려가 없기 때문에 서로 다른 성능과 자원을 가지는 노드들로 구성된 시스템에서는 시스템을 효율적으로 사용할 수 없는 문제점이 있다. 이러한 문제점을 해결하기 위해 본 논문에서는 이동 에이전트를 이용하여 각 노드의 성능과 자원에 대한 정보를 수집하고 이를 이용하여 주어진 작업에 대해 최적의 노드를 선택하여 할당하는 방법을 제안하고 구현하였다. 분해할 수 없는 작업의 경우 최적의 노드를 선택하여 작업을 할당하는 것은 작업 처리량을 극대화하여 시스템의 효율성을 높일 수 있을 것이다.

## 1.서론

서로 다른 성능과 자원들을 가지는 분산시스템에서 분해할 수 없는 여러 개의 작업을 다수의 작업 노드를 이용하여 분산 처리할 때 각 노드의 성능과 자원 현황을 고려하여 주어진 작업에 대한 최적의 노드를 선택하여 작업을 할당한다면 시스템의 처리 능력을 극대화할 수 있을 것이다.

본 논문에서는 이질적인 성능을 가지는 노드들로 구성된 일반적인 지역 네트워크 환경 하에서 이동 에이전트를 이용하여 각 노드들을 주기적으로 검사하여 노드의 성능 정보를 취득한 후 이를 토대로 주어진 작업들에 대한 최적의 노드를 선택해서 작업을 할당하는 방법을 연구하고 구현하였다.

## 2.관련연구

### 2.1 이동 에이전트(Mobile Agent)

이동 에이전트는 네트워크 내에서 스스로 이동하면서 사용자 또는 다른 개체 대신 행동할 수 있는 컴퓨터 프로그램을 말한다. 이동 에이전트는 분산 컴퓨팅 환경에 동적으로 이식이 가능하며(portability), 사용자가 요구하는 작업을 특정한 호스트로 이동하여 수행할 수 있으며(code mobility), 다른 에이전트의 직접적인 지시나 간섭 없이도 스스로 판단하여 행동하는 자율성(autonomy)을 가진다[1,3,4].

### 2.2 부하 균등(Load Balancing)

부하 조절은 분산 컴퓨팅 환경에서 다수의 작업에 대한 처리를 프로세서들에게 균형 있게 분배하는데 목표를 둔 작업 스케줄링으로 시스템의 작업 처리량을 극대화할 수 있다[6].

부하 균등 알고리즘은 정적 방법과 동적 방법이 있는데 정적 방법은 현재 네트워크의 부하를 무시하고 임의의 프로세서에게 작업을 할당하며 동적 방법은 네트웍 부하를 일부 고려하여 부하가 적은 프로세서에게 새로운 작업을 할당한다[5].

### 2.3 연구 모델의 특성

기존의 연구들은 분해할 수 있는 작업에 대한 부하 균등에 초점을 두었다. 본 논문에서는 이동 에이전트를 이용한 동적 부하 균등 기법을 이용하여 분해할 수 없는 다수의 작업을 작업 처리 노드의 성능을 고려하여 처리함으로써 분산 시스템의 효율을 높이는 데에 초점을 두고 연구하였다.

## 3.이동 에이전트를 이용한 작업 할당 방법의 설계 및 구현

서버 노드는 이동 에이전트를 이용하여 작업 노드들의 처리능력을 측정하고 측정된 결과를 토대로 노드간의 상대적인 순위를 정해 분해할 수 없는 작업을 할당하게 된다.

### 3.1 노드의 성능과 유휴 자원 검사 및 노드간 순위 결정

각 노드에 대한 성능 정보는 표1과 같은 단위로 구성되며 식1과 같이 계산되어 상대적인 순위가 결정된다. $C_i$는 i번째 노드의 프로세서 처리 성능을 의미하고 $M_i$는 노드의 물리적인 가용 메모리 크기이며 $D_i$는 해당 노드와 서버 노드간의 네트워크 지연 시간이다. $\alpha$, $\beta$, $\gamma$는 가중치로서 그 합이 1인 값으로 각 항목의 전체 처리 능력에 대한 중요도에 따라 측정 항목에 곱해지는 상수 값으로 합이 1이다.

각 노드의 정보는 주기적으로 노드간을 이동하는 에이전트에

의해 수집되며 이를 서버 노드에서 취합하여 순위를 결정한다.

표 1. 노드별 성능 측정 항목

| | |
|---|---|
| 프로세서 | 부하를 고려한 프로세서의 수행 성능 |
| 메 모 리 | 가용 메모리 크기(MB) |
| 네트워크 | 서버 및 작업 노드간 지연시간(ms) |

$$P_{node} = \alpha C_i + \beta M_i + \gamma D_i \qquad (식1)$$

3.2 처리할 작업의 순위 결정 및 할당 방법

본 논문에서는 서로 다른 작업 크기를 가지는 k개의 (k=4,8,16) 작업에 대한 데이터 크기를 미리 알고 있다는 가정 하에 처리할 작업의 우선순위를 결정하며 이 순위를 이용하여 이질적인 성능을 가지는 8개의 호스트에서 처리하도록 하였다. 작업 할당 방법은 다음과 같은 두 가지 방법을 사용한다.

3.2.1 절대적인 최적의 노드를 선택하여 작업을 할당

처리할 작업의 크기를 고려하지 않고 작업처리 순서에 따라 가용한 노드 중 최고의 성능을 가진 노드부터 선택하여 작업을 할당한다.

표2에서 네 개의 작업을 순서대로 처리할 경우 작업 크기는 고려하지 않고 처리 순서에 따라 가용한 노드 중 최고의 성능을 나타내는 노드로 작업을 할당하게 된다.

표 2. 절대적인 최적의 노드 결정

| 작업테이블 | | | 호스트테이블 | | |
|---|---|---|---|---|---|
| 작업 번호 | 작업 크기 | 크기 순위 | 호스트 번호 | 호스트 성능 | 성능 순위 |
| Task1 | $m_1 \times n_1$ | 3 | Host1 | 13 | 2 |
| Task2 | $m_2 \times n_2$ | 1 | Host2 | 8 | 3 |
| Task3 | $m_3 \times n_3$ | 4 | Host3 | 15 | 1 |
| Task4 | $m_4 \times n_4$ | 2 | Host4 | 5 | 4 |

3.2.2 상대적인 최적의 노드를 선택하여 작업을 할당

처리할 작업의 크기를 고려하여 작업간의 크기 순위를 결정하고 가용한 노드의 성능 순위와 비교하여 상대적으로 적합한 노드를 선택해서 작업을 할당하게 된다.

표3처럼 호스트의 성능 순위와 처리할 작업의 크기 순위를 비교해서 상대적으로 최적인 노드를 선택하여 작업을 할당하는 방법이다. 상대적으로 최적인 노드가 존재하지 않을 경우 다음의 최적인 노드로 처리할 작업을 할당한다.

표 3. 상대적인 최적의 노드 결정

| 작업테이블 | | | 호스트테이블 | | |
|---|---|---|---|---|---|
| 작업 번호 | 작업 크기 | 크기 순위 | 호스트 번호 | 호스트 성능 | 성능 순위 |
| Task1 | $m_1 \times n_1$ | 3 | Host1 | 13 | 2 |
| Task2 | $m_2 \times n_2$ | 1 | Host2 | 8 | 3 |
| Task3 | $m_3 \times n_3$ | 4 | Host3 | 15 | 1 |
| Task4 | $m_4 \times n_4$ | 2 | Host4 | 5 | 4 |

3.3 처리할 작업의 종류

m× n의 크기를 가지는 부동소수점 수를 원소로 가지는 임의의 행렬을 생성하여 행렬의 제곱을 구하는 것이 각 노드에서 처리할 작업이다.

3.4 제안한 시스템의 구조

JSDK(Java Standard Development Kit) 1.1.8과 IBM의 ASDK(Aglets Software Development Kit) 1.1 beta3를 이용하여 구현하였다[8,9]. 시스템의 구성은 관리 에이전트, 노드 정보 획득 에이전트, 작업 에이전트로 이루어지며 각각의 역할은 다음과 같다.

•관리 에이전트(Manager Agent)

서버 노드에 정적으로 존재하면서 8개로 구성되는 작업 노드들에 대한 IP 주소 테이블, 노드 성능 정보 테이블, 작업 크기 정보 테이블을 가진다. 각 노드를 거치면서 노드에 대한 정보를 획득할 노드 정보 획득 에이전트를 생성하고 이주시킨다. 노드 정보 획득 에이전트가 전송하는 정보를 토대로 노드들간의 성능에 따른 상대적인 순위를 결정하고 이를 이용하여 처리할 작업을 작업 에이전트를 통해 특정 노드에 할당한다.

•노드 정보 획득 에이전트(Node Information Gathering Agent)

정보 획득 에이전트는 각 노드들을 방문하면서 해당 노드에 대한 성능을 검사하고 그 결과를 서버 노드에 있는 관리 에이전트에게 전달한 뒤 다음 노드로 이동하게 된다.

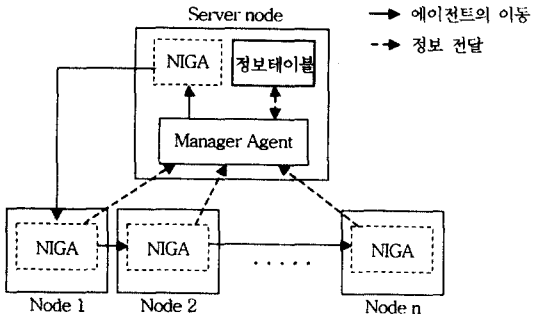


그림 1. 이동 에이전트의 동작과정

그림1을 보면 관리 에이전트로부터 생성된 NIGA가 각 노드를 방문하면서 노드 성능 정보를 관리 에이전트에게 전달하는 것을 볼 수 있다.

•작업 에이전트(Worker Agent)

3.3에서 설명한 작업을 서버 에이전트가 지정한 노드에서 수행하고 서버 노드로 귀환한다.

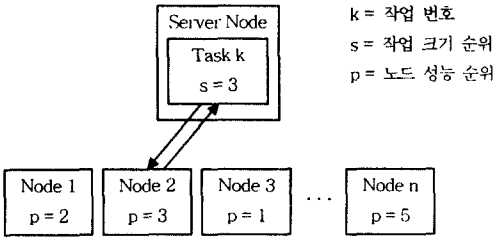


그림 2. 상대적 순위에 따른 작업 할당

그림2와 같이 서버 노드는 처리할 작업의 순위를 각 노드들의 상대적인 처리 능력에 따른 순위와 비교하여 최적의 노드를 선택하고 주어진 작업을 할당하게 된다. 작업을 할당 받은 노드

에 대한 관리 에이전트의 노드 성능 정보가 NIGA에 의해 갱신되기 전에는 작업 노드는 추가의 작업을 할당 받지 않는다. 하나의 노드에 대한 동일한 성능 정보로 두 개의 작업을 할당하는 것을 방지하기 위함이다. 작업 처리가 종료되어 서버 노드로 귀환한 작업 에이전트는 작업의 종료를 관리 에이전트에게 통보하며 관리 에이전트는 해당 노드에 대한 성능 정보를 갱신하게 된다.

## 4.성능 평가

구현에 대한 성능 평가는 표4,5과 같은 환경에서 수행되었다. 노드의 수는 8개로 고정하고 서로 다른 크기를 가지는 작업의 수를 각각 4개, 8개, 16개로 하여 절대적인 최적의 노드 선택법, 상대적인 최적의 노드 선택법을 적용해서 전체적인 처리 시간을 측정하여 비교하였다.

표 4. 각 노드의 성능 결정 변수

| 노드번호 | 프로세서 | 메모리크기 | 지연시간 |
|---|---|---|---|
| Host1 | P3-700 | 256MB | 0.0ms |
| Host2 | P3-700 | 128MB | 0.1ms |
| Host3 | P2-350 | 192MB | 0.0ms |
| Host4 | P2-350 | 64MB | 0.1ms |
| Host5 | P2-350 | 32MB | 0.2ms |
| Host6 | P2-350 | 32MB | 0.2ms |
| Host7 | Cel-466 | 192MB | 0.1ms |
| Host8 | Cel-433 | 192MB | 0.0ms |

표 5. 작업 크기

| 작업번호 | 크기 | 비고 |
|---|---|---|
| Task i | (j× 100) × (j× 100) | 1 ≤ i ≤ k<br>k = 4, 8, 16<br>1 ≤ j ≤ 8<br>i, j, k는 정수 |

작업 에이전트가 작업 노드에서 생성하여 계산하는 행렬의 크기는 JVM(Java Virtual Machine)의 메모리 한계로 인해 800×800 이상의 크기를 생성할 수 없었다.

지역 네트워크라는 환경의 특성상 서버 노드와 작업 노드간의 지연시간은 평균 0.1ms정도로 측정되어 작업 노드의 성능을 결정짓는데 큰 영향을 주지는 않았으며 NIGA가 8개의 작업 노드들을 방문하여 노드 정보를 획득하는데 걸리는 시간은 평균 955ms로 측정되었다.

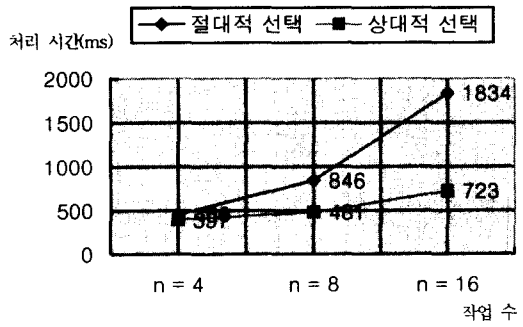


그림 3. 작업 수에 따른 각 방법의 작업 처리 시간

그림3은 작업 수가 각각 4, 8, 16개일 때 각각의 노드 선택 방법을 사용하여 노드를 할당할 경우에 측정된 시스템의 전체적인 처리 시간을 보여준다. 작업 수가 노드 수보다 적을 경우 두 가지 할당 방법이 비슷한 성능을 보여 주었으나 작업 수가 노드 수보다 많을 때 두 방법의 성능 차이가 크다는 것을 볼 수 있다.

## 5.결론

본 논문에서는 이동 에이전트를 이용하여 각 호스트의 성능에 대한 정보를 수집하고 최적의 노드를 선택하여 작업을 할당함으로써 분해할 수 없는 다수의 작업을 처리할 때 작업 크기와 노드 성능을 고려하여 상대적인 최적의 노드를 선택하는 작업 할당 방법이 더 좋은 성능을 나타내는 것을 알 수 있다.

이번 연구에서는 호스트의 수를 고정하고 작업의 수를 가변적으로 한 뒤 노드 정보 획득 에이전트가 주기적으로 성능 정보를 획득하도록 구현하였으나 차후에는 호스트 수가 동적으로 변하는 환경에서 작업에 따라 정보 획득 에이전트의 주기를 조절하면서 작업을 처리하는 연구를 수행할 예정이다.

## 참고문헌

[1] OMG, " Mobile Agent Facility Specification", http://www.omg.org/, pp12-13, 2000.

[2] F. Kurfess, D. Shah, " Monitoring Distributed Processes with Intelligent Agents", Proceedings of the 1999 IEEE Conference and Workshop on Engineering of Computer-Based Systems, 1999.

[3] V. A. Pham, A. Karmouch, "Mobile software agents: an overview", IEEE Communications Magazine, Vol. 36, Issue 7, pp. 26 - 37, July 1998.

[4] D. Chess, C. Harrison, A. Kershenbaum. "Mobile agents: Are they a good idea?, In Mobile Object Systems: Towards the Programmable Internet", Vol. 1222 of Lecture Notes in Computer Science, Springer-Verlag, 1997.

[5] 김영균, 김영학, 오길호, " 인터넷 환경에서 이동 에이전트 이주 전략을 사용한 큰 입자도를 갖는 작업들의 결함 허용 동적 부하 균등화 기법", 정보과학회 봄 학술발표논문집, pp562-564, 4월, 2001년.

[6] 김지균, 김태윤, " 이동 에이전트 기반의 동적 작업 부하 균형 프레임워크", 정보과학회 논문집, 제28권, 제2호, pp196-206, 6월, 2001년.

[7] Aglet workbench, http://www.trl.ibm.com/aglets, IBM Japan, 2000.

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