이동 에이전트 시스템 기반의 병렬 계산을 위한 효율적인 분산 방법

김경하° 김영균 김영학 오길호
금오공과대학교 컴퓨터공학과
{khkim, ygkim, yhkim, gilho}@cespc1.kumoh.ac.kr

An Efficient Distribution Method for Parallel Computation Based on a Mobile Agent System

Kyoung-Ha Kim° Young-Gyun Kim Young-Hak Kim Gil-Ho Oh
Dept. of Computer Engineering, Kumoh National University of Technology

요 약

인터넷 상에 분산되어 있는 다수의 일반 컴퓨터들을 이용하여 높은 컴퓨팅 파워를 요구하는 응용문제를 병렬분산 처리함으로써 값 비싼 고성능의 슈퍼컴퓨터를 사용하는 것 보다 경제적인 효과를 얻을 수 있다. 본 연구에서는 다중 에이전트 시스템을 이용해서 가상 병렬 시스템을 구성하고, 기존 방법들 보다 더 효과적인 방법으로 워커 에이전트와 작업 패키지를 분산하고 결과를 얻는 새로운 방법을 제안한다. IBM의 Aglet 시스템을 이용하여 이동 컴퓨팅 환경을 모델링 하였고, 제안된 분산 기법에 관한 성능 모델을 수학적으로 유도하여 그 결과를 기존 결과와 비교함으로써 본 논문에서 제안된 방법이 더 효율적임을 보인다.

1. 서론

최근에 높은 컴퓨팅 파워를 요구하는 응용 분야에 값 비싼 고성능의 슈퍼컴퓨터를 대신하여, 네트워크 상에 분산되어 있는 컴퓨터를 이용하여 문제를 해결하기 위한 다양한 연구가 진행되고 있다[1,2,3]. 다중 에이전트 시스템을 이용한 병렬 컴퓨팅 환경을 구성하려는 연구도 이들 분야중의 하나이다. 이러한 시스템은 네트워크 상에 분산되어 존재하는 이기종 시스템들을 가상 병렬 컴퓨팅 시스템으로 구성하고, 하나의 큰 작업을 분할하여 다수의 컴퓨터 상에서 효율적으로 분산처리 할 수 있다[3,4,5].

본 논문에서는 가상 병렬 컴퓨팅환경에서 효율적인 방법으로 워커 에이전트와 작업 패키지를 분산하는 방법과 이에 대한 성능모델을 제안한다. 여기서 워커 에이전트는 가상 병렬 컴퓨팅환경에 참여하는 컴퓨터로 이동 되어 할당 받은 문제를 해결하는 프로그램이고, 작업 패키지는 각 워커 에이전트가 수행 할 자료들의 집합을 의미한다.

기존연구에서는 단순히 하나의 중앙 코디네이터에 의해 워커 에이전트가 다른 여러 호스트로 이동되고, 주어진 작업 패키지는 단위 작업 패키지로 분할되어 각 호스트에 전송된다[3]. 그래서 에이전트 수가 일정한 값을 초과하면 오히려 중앙 코디네이터에 많은 부하가 걸려 성능이 저하된다. 본 논문에서는 이러한 문제를 개선하기 위한 새로운 분산 기법을 제안하고, 그 결과를 기존 방법과 비교하여 성능이 더 우수함을 보인다. 또한 본 논문에서는 가상 병렬 컴퓨팅 환경을 구성하기 위해서 IBM의 Aglet 시스템을 기본 모델로 사용하였다. Aglet은 Java로 구현되어 있기 때문에 플랫폼에 독립적이며 사용자가 그 기능을 확장할 수 있는 효율적인 시스템이다.

2. 관련연구 및 문제점

다중 에이전트 시스템은 에이전트를 구성하는 작은 단위의 에이전트들이 복잡하게 구성되어, 상호작용과 협동을 통해 대단위 병렬성을 가진 문제를 해결하는데 이용될 수 있다. 그러한 시스템의 예로는 IBM의 Aglet[6], 독일 Stuttgart 대학의 Mole[7], 그리고 FIPA(Foundation for Intelligent Physical Agents)의 FIPA AP(Agent Platform)[8] 등이 있다.

최근에 Stuttgart 대학에서 Mole 시스템 기반의 가상 병렬 컴퓨팅환경에서 응용문제의 분산기법과 이에 대한

성능평가 모델이 제안되었다[3]. 여기서 제안된 분산 아이디어는 다음과 같다. 하나의 코디네이터가 다수의 호스트에 워커 에이전트를 배치시켜 주어진 응용문제를 여러 개의 단위 작업 패키지로 분할하여 전송하고, 워커 에이전트에서 계산된 결과를 다시 수집하는 방법을 사용하였다. 이러한 방법의 문제점을 살펴보면, 첫째 작업의 양에 따라 성능 향상을 위해 워커 에이전트의 수를 증가 하지만 워커 에이전트의 수가 일정 범위 이상이 되면 더 이상의 성능 향상이 없다. 둘째, 코디네이터에서의 분산 수행시간이 각 워커 에이전트의 평균 수행시간보다 더 길게 되면 코디네이터에서 병목현상이 발생된다.

본 논문에서는 위에서 열거한 문제점을 개선하기 위해 코디네이터의 역할을 각 워커 에이전트에게 위임하는 방법을 사용한다.

3. 새로운 분산 컴퓨팅 모델

먼저 동적으로 이진트리와 유사하게 가상 병렬 환경 시스템을 구성한다. 이러한 시스템에 참여하는 각 워커 에이전트는 전체 시스템의 IP 테이블을 유지하며, 자신이 부모 호스트로부터 전송 받은 워커 에이전트와 작업 패키지를 다시 분할하여 자식 호스트에 전송하는 위임 능력을 갖는다. 이러한 분산 방법을 사용함으로써 코디네이터의 병목현상에 따른 네트워크 부하를 감소시킬 수 있고, 또한 성능 측면에서 기존 방법보다 더 효율적임을 보인다.

3.1 기본 구조

본 논문의 성능평가를 위해 [3]과 동일한 가정을 사용한다. 먼저 네트워크는 완전 연결 되어있으며 신뢰성을 제공한다. 모든 워커 에이전트들은 동일한 속도로 수행되며 작업 패키지의 평균 실행 시간, 작업 패키지 결과의 평균 통합 시간, 그리고 워커 에이전트 시작 시간은 이미 알려져 있다. 각 워커 에이전트 사이의 통신 지연과 처리량은 모두 동일하다. 따라서 작업 패키지의 전송 시간, 결과 패키지 전송시간, 워커 에이전트들의 이주 시간은 일정하다. 각 워커 에이전트들은 동일한 기능을 수행한다.

그림 1은 본 논문에서 제안한 가상 병렬 컴퓨팅 환경의 기본 구조이다. 번호는 워커 에이전트가 분산된 호스트별로 순서대로 생성되는 단계를 의미한다. 첫번째 단계에서는 (1)번에 연결된 워커 에이전트가 생성 되고

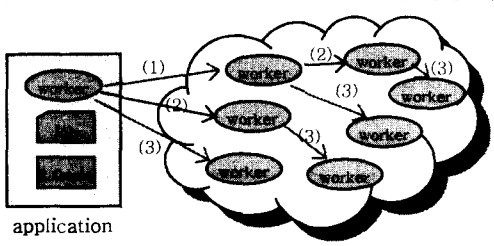

그림1. 에이전트 기반의 병렬 컴퓨팅 기본 구조

그 다음 단계에는 (2)번에 연결된 워커 에이전트가 동시에 생성된다. 따라서 그림에서 처럼 3단계가 경과한 후에는 8개의 워커 에이전트가 생성된다.

3.2 분산 알고리즘

m개의 워커 에이전트를 사용해서 문제를 해결하려고 할 때, 새로운 방법의 분산 알고리즘은 다음과 같다.

```
for i=0 to log2m-1
    for j=0 to 2^i-1 in parallel
        j번째 IP 호스트 → 2^i + j번째 IP 호스트
```

▶ IP 테이블 유지 : 시스템에 참여할 컴퓨터의 성능이 모두 동일하다고 가정하고, 가상 컴퓨팅 환경에 참여하는 각 호스트의 IP 테이블을 구성한다. 최초에 주 코디네이터가 이러한 테이블을 구성하고 각 호스트에 이동 컴퓨팅을 이용하여 전송하게 된다.

▶ 분산 전략 : 그림 2는 IP 테이블의 구조이며, 각 워커 에이전트들의 배치 순서와 IP 테이블의 관계를 도식화 한 것이다. 그림에서 H_m은 호스트 IP를 의미하고, step은 위의 분산 알고리즘에서 i의 증가 값, 오른쪽의 개수는 각 step의 생성 워커 에이전트들의 수를 의미한다. 간단하게 수행 절차를 살펴보면 어플리케이션 수행과 함께 워커 에이전트가 생성되고 step0일 때, 자신을 복제하고 H_2 호스트로 이동한다. step1일 때, H_1과 H_2에 워커 에이전트들이 자신을 복제하고 H_3와 H_4 호스트로 각각 이동한다. step3일 때, H_1, H_2, H_3, H_4의 워커 에이전트들이 자신을 복제하고 H_5, H_6, H_7, H_8로 이동을 한다. 이렇게 step이 증가 할 때 마다 2의 배수로 증가하면서 워커 에이전트를 호스트로 배치한다.

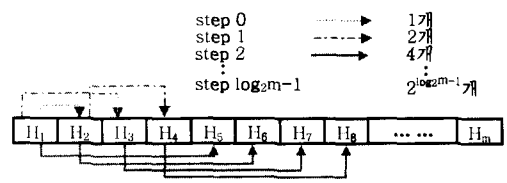

그림2. IP 테이블 구성과 각 단계에 따른 워커의 생성

▶ 동기화 문제 : 각 에이전트의 동기화는 step이 증가할 때 마다 카운트(cnt)를 가짐으로써 가능한데, i번째 단계에서 IP 테이블의 인덱스 cnt^i-1까지의 호스트 주소에는 모두 워커 에이전트가 배치되어 있다고 볼 수 있다.

모든 호스트에 워커 에이전트가 전송되면 작업 패키지를 전송한다. 전송 방법은 워커 에이전트 분산전략과 동일한 방법을 적용한다. H_1 호스트에 n개의 작업 패키지가 있다면, step0에서 H_2로 n/2개를 전송하고, step1에서 H_1, H_2가 동시에 자신의 반, 즉 n/4을 H_3, H_4에 각각 전달하게 된다. 그 다음에서도 동일한 방법으로 step$\log_2 m$-1 까지 각 호스트에 작업 패키지를 전송하면 결국 각각의 워커 에이전트는 n/m만큼의 작업 패키지를 가지고 있게 된다. 모든 워커 에이전트에게 작업 패키

지가 전송되면 동시에 워커 에이전트들은 자신의 작업 패키지를 수행하고, 그 결과는 자신에게 작업 패키지를 주었던 워커 에이전트에게 되돌려주는 방법으로 처음 호스트까지 결과를 통합해서 최종결과를 내게 된다.

4. 성능 평가

3장에서 제안한 성능 모델을 바탕으로 기존 방법과 성능을 비교하기 위해 제안한 알고리즘의 성능 평가식을 유도한다.

4.1 성능 모델

먼저 주어진 문제가 n개의 작업 패키지로 구성되어 있고, m개의 워커 에이전트에 의해서 수행된다고 가정한다. 성능 분석을 위해 다음과 같은 기호들을 정의한다.

t_{start}:워커 시작 시간, t_{mig}:워커 에이전트 이동시간, t_{send}:작업 패키지가 워커 에이전트에 전송되는 시간, t_{work}:전체 패키지의 수행 시간, t_{rec}:결과 받는 시간, t_{int}:결과 통합 시간, t_{exe}:단위 작업 패키지 수행 시간

앞에서 제안한 분산 알고리즘에 관한 수행시간은 다음과 같은 수식으로 표현될 수 있다.

- 각 호스트에 워커 에이전트를 배치하는 시간

$$t_{ini} = (t_{start} + t_{mig})\log_2 m$$

- 모든 작업 패키지가 워커 에이전트에 전송되는 시간

$$t_{pack} = n(1/2+1/4+1/8 + \cdots + 1/(2^{\log_2 m})) * t_{send}$$
$$= n(1-(1/2)^{\log_2 m}) * t_{send}$$

- 결과를 위한 워커 에이전트 수행 시간

$$t_{work} = n/m * t_{exe}$$

- 계산결과 받고 통합하는 시간

$$t_{fin} = (t_{rec} + t_{int})\log_2 m$$

따라서, m개의 워커 에이전트를 이용했을 때 전체 소요 시간(t_{total})은 다음과 같이 계산될 수 있다.

$$t_{total} = t_{ini} + t_{pack} + t_{work} + t_{fin}$$

4.2 성능 비교

앞 절에 유도한 수식을 이용해서 성능을 평가하기 위해 2개에서 16개 까지 두 개씩 워커 에이전트를 추가하여 500개의 작업 패키지의 실행환경을 고려한다.

그림 3은 t_{exe}=20, t_{start}=200, t_{mig}=100, t_{send}=t_{rec}=1, t_{int}=3으로 주었을 때, [3]의 방법과 본 논문에서 제안한 방법의 성능을 비교한 그래프이다. 그래프에서 Coord는 [3]에서 제안한 방법의 성능을 나타내고, Modify는 본 논문에서 제안한 방법의 성능을 나타낸다.

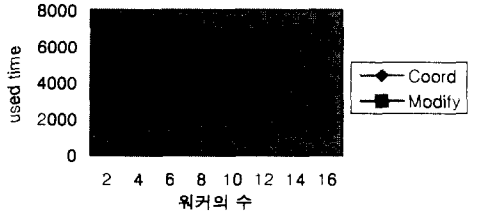

그림3. 워커의 수에 따른 성능 비교

기존방법은 워커 에이전트 수가 4개 이상이 되면 더 이상의 성능 향상이 없다. 이것은 워커 에이전트 수행시간 보다 코디네이터 수행시간이 더 많이 걸리게 되어 코디네이터에 부하가 집중되기 때문이다. 반면, 개선된 방법은 워커 에이전트 수가 증가하면 더 좋은 성능을 낸다. 이것은 코디네이터 부하를 개선하기 위해 각 워커 에이전트에게 코디네이터 기능을 위임하여 부하를 분산함으로써 얻어진 결과이다.

그림 4는 워커 에이전트의 수를 8, 단위 작업 패키지 수행 시간을 20으로 고정하고 작업 패키지 수를 100에서 500까지 100씩 증가시켰을 때의 성능 비교 그래프이다. 그래프에서 보는 바와 같이 작업 패키지 수의 증가에 따른 성능도 [3]에서 제안한 방법보다 더 좋은 성능을 낸다.

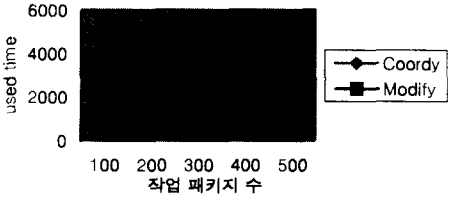

그림4. 작업 패키지 수에 따른 성능 비교

5. 결론

본 논문에서는 이동 에이전트를 기반으로 하는 가상 병렬 컴퓨팅 환경에서 병렬계산을 분할하여 효율적으로 배치하고 수행하는 방법을 제안하였다. 또한 그에 대한 성능을 기존 방법과 비교하여 더 우수 함을 보였다.

향후 과제는 작업 패키지의 전송 단계가 증가함에 따라 통신 트래픽이 증가되는 문제를 해결하고, 이기종 환경에서 IP 테이블 관리하는 기법과 각 호스트의 능력에 따라 작업을 분산하는 방법들의 연구가 고려된다.

참고문헌

[1] 김종권 외 3인, "MPI 표준 기능 및 기술동향 고찰", 병렬처리 시스템 연구회지 7권 1호, 1996.

[2] L.F.G. Sarmenta, "Bayanihan: Web-Based Volunteer Computing Using Java", http://www.cag.lcs.mit.edu/-bayanihan, 1998.

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

[4] D. Chess, C. Harrison, A. Kershen-baum, "Mobile Agents: Are They a Good Idea?", 1995.

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

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

[7] J.Baumann, F.Hohl, K.Rothermel, M.Straber, "Mole-Concepts of a Mobile Agent System", 1997.

[8] FIPA, "Agent Management", FIPA version 1.0, http://fipa.org/spec/fipa98.html, 1998.