최적의 이주를 위한 이동에이전트 설계

김완성, 복경수, 유재수
충북대학교 정보통신공학과
e-mail: wskim, ksbok@netdb.chungbuk.ac.kr
yjs@cbucc.chungbuk.ac.kr

Design of Mobile Agents for The Optimal Migration

Wan Sung Kim, Kyoung Soo Bok, Jae Soo Yoo
Dept. of Computer and Communication Engineering, Chungbuk National University

요 약

네트워크의 발달과 다양한 서비스의 요구에 따라 새로운 소프트웨어의 패러다임에 대한 요구와 함께 이동 에이전트에 대한 많은 연구가 진행 중이다. 이동 에이전트의 수행에 있어 이주비용은 이동 에이전트의 성능에 많은 영향을 미친다. 본 논문에서 이동에이전트의 이주비용을 최적화하기 위한 기법을 제안한다. 제안하는 이주기법의 특징은 다음과 같다. 첫째, 네트워크 상태 및 플랫폼 상태변화에 적절하게 대응할 수 있는 동적 경로를 생성하여 에이전트 수행 효율을 높인다. 둘째, 수행할 코드를 프리패칭하여 이동 데이터량을 줄이고, 필요한 에이전트를 미리 인스턴스시켜 수행 시간을 단축한다. 셋째, 체크포인트기법을 사용하여 에이전트 수행 중에 에러가 발생할지라도 에이전트는 재 수행을 하지 않고 에러 이전의 상태로 복구하는 방법을 사용하여 수행 효율을 높인다.

1. 서론

최근 컴퓨터, 통신기술의 발전은 인터넷이라는 거대한 통신망을 만들어 놓았다. 이러한 거대한 네트워크 통신망을 유지, 보수, 관리에 어려움이 따르고, 네트워크 환경의 불확실성 하에서 예측하지 못한 여러 가지 문제를 발생시킬 수 있다. 이로 인하여 특정 작업을 수행 할 수 없게 되거나 필요 이상의 비용 발생과 자원 소모를 유발 할 수 있다. 따라서 이에 적합한 새로운 소프트웨어 기법에 대한 요구와 함께 이동 에이전트 기반 시스템 기술들이 개발되었다. 에이전트 기반 기술들은 사용자를 대신하여 특정 작업을 자동으로 수행해주는 지능을 가진 소프트웨어로써 자율성(Autonomy), 지능(Intelligence) 그리고 이동성(Mobility)등의 특징이 있다[1]. 이동성을 강조한 이동 에이전트는 기존의 클라이언트/서버가 모든 작업이 중앙 시스템에 집중되어 있었기 때문에 서버 부하 및 네트워크 부하로 인하여 원활한 서비스를 제공하지 못했던 문제점을 극복했다. 이러한 이동 에이전트의 성능은 이동 데이터 크기 및 네트워크의 상태 그리고 이동할 플랫폼의 상태에 따라 이동 에이전트의 성능에 영향을 미친다[2]. 그러나 기존의 Tacoma, Agent-Tcl, Aglets, Voyager, Odyssey, Concordia, JAMES 그리고 AJENTA 등의 이동 에이전트의 플랫폼들은 동적인 환경변화 및 이주비용에 대한 특징을 고려하지 못하고 있다. 또한 일관성 있는 이주기법 즉 결함에 대한 처리를 제공하지 못하는 문제점 때문에 성능에 문제를 야기시킨다. 이러한 문제점을 해결하고자 본 논문에서는 정적, 동적 경로를 혼합한 방법, 프리패칭 및 체크포인트 기술을 사용하여 최적의 이주(Migration) 비용과 일관성을 제공하는 이주기법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 기존에 개발된 에이전트 시스템의 이주기법 및 특징을 기술한다. 3장에서는 제안하는 최적의 이주비용을 위한 에이전트 이주기법에 대해서 기술한다. 4장에서는 제안한 방법과 기존의 방법들을 시뮬레이션을 통해

"본 논문은 2003년도 산업자원부 차세대신기술개발사업의 지원에 의하여 수행되었음"

서 비교 분석한다. 마지막으로 5장에서 결론에 대해 기술한다.

2. 관련연구

이동 에이전트의 이주기법에 대한 많은 연구가 진행중이다. JAMES는 클라이언트가 서비스를 요청하게 되면 중앙 호스트는 서비스를 수행하기 위해 이동 에이전트를 생성하고, 정적 경로(Itinerary)를 생성한다. 경로 정보에 따라 필요한 코드를 각 에이전트 플랫폼에게 메시지를 보낸다. 메시지를 받은 에이전트 플랫폼에서는 자신이 필요한 코드를 자신의 캐쉬 메모리, 이전 에이전트 플랫폼 캐쉬 메모리, 자신의 보조기억장치에서 코드를 찾고, 코드가 없으면 JAMES 코드 서버에 필요한 코드를 요청하여 미리 코드를 받게 된다[5]. 이러한 프리패칭(Prefetching) 기법을 이용한 JAMES 플랫폼은 이주할 때 이동 데이터량을 줄여 네트워크 트래픽(Traffic)을 개선하였다[3][4]. AJANTA는 클래스 로딩(Class Loading)기법을 이용하여 수행에 필요한 코드만을 이동하는 방법을 사용하여 데이터의 양을 최소화함으로써 네트워크 부하를 줄였고 성능을 향상시켰다. 그리고 에이전트의 수행에 적합한 이주 패턴(Pattern)을 선택하고, 선택된 이주 패턴에 따라 에이전트를 수행함으로써 성능을 향상시켰다[5].

그러나 JAMES의 이주기법은 동적인 환경변화에 적절하게 대응하지 못하는 문제점과 결함에 대한 처리를 제공하지 못했고, AJANTA의 이주기법은 이동 데이터 크기를 고려하지 못하는 문제점 때문에 성능의 저하 요인 되었다.

3. 최적의 이주 비용을 위한 에이전트 설계

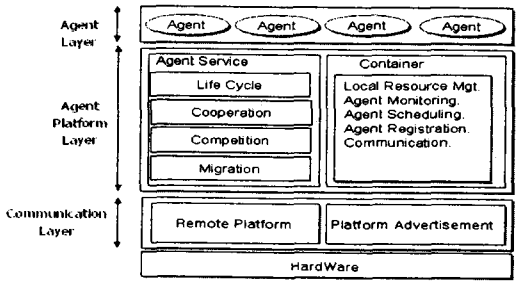

그림 1 플랫폼 구조

이 장에서는 이동 에이전트 최적의 이주기법을 위한 방법을 제안한다. 제안하는 방법의 기본적인 목적은 플랫폼의 환경변화와 네트워크의 상태변화 그리고 시스템 결함에 적절하게 대응하여 성능을 낼 수 있도록 하는 것이다. 이를 위한 플랫폼 구조는 그림 1과 같이 구성한다.

3.1 이주 기법

에이전트 수행과정은 그림 2 에서 보는 것과 같이 4단계의 과정을 거쳐 수행한다. 1단계에서는 클라이언트의 서비스 요청에 의해 에이전트가 생성되고, 2단계에서는 에이전트를 수행하며, 실질적인 이주단계는 3단계와 4단계에서 이루어지며, 에이전트 생성시에만 1단계를 수행하고, 생성 후 에이전트 수행이 완료될 때까지 2단계에서 4단계까지 반복 수행하면서 에이전트의 임무를 수행한다.

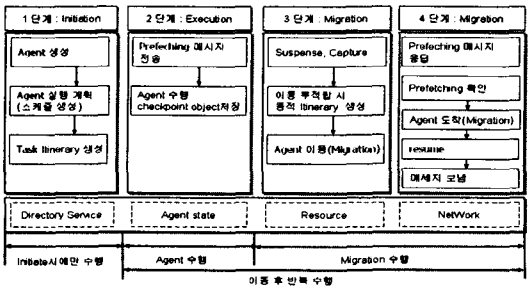

그림 3 에이전트 수행 과정

3.2 동적 경로 생성

기존의 에이전트 시스템들은 처음 에이전트 생성할 때 디렉토리서비스(Directory Service)를 통해 에이전트가 수행해야 할 플랫폼 리스트를 생성하고 정적인 이동경로를 만든다. 정적 경로는 이동 에이전트 생성 시 한번만 생성하여 사용한다는 장점이 있는 반면, 에이전트 수행 중, 네트워크의 환경변화(네트워크 부하, 지연시간) 및 다음 목적지의 플랫폼 상태변화(플랫폼 자원 부족)에 따라 적절하게 대응하지 못하는 단점이 있다. 동적 경로는 에이전트가 생성되어 다음 목적지 플랫폼으로 이동 전 네트워크 및 플랫폼의 상태를 검사하여 에이전트 수행가능 여부를 확인하고 에이전트 실행환경에 적합한 경로를 생성한다. 동적 경로는 이주 시 매번 네트워크 및 플랫폼의 상태를 검사하여 새로운 경로를 생성하기 때문에 경로생성에 따른 소요시간이 증가하게 되는 단점이 있다[6]. 이러한 정적 및 동적 경로의 특징을 고려한 경로 생성방법을 제안한다. 에이전트를 생성할 때 컨테이너의 에이전트스케줄링(Agent Scheduling), 에이전트모니터링(Agent Monitoring),

로컬자원관리(Local Resource Management), 에이전트등록(Agent Registration) 서비스를 통하여 최적의 이주경로를 생성하고, 에이전트가 이동하면서 각각의 플랫폼에서 에이전트 실행한다. 실행한 다음 이주할 플랫폼의 상태 및 네트워크상태(네트워크 부하, 지연시간)를 컨테이너의 서비스들을 통해 검사한 후, 에이전트가 수행에 적합할 경우 이전에 생성된 경로에 따라 에이전트를 수행하고, 수행 불가능할 경우 컨테이너의 서비스들을 통해 새롭게 최적의 경로를 생성하고 새로운 경로정보에 따라 에이전트를 수행한다.

3.3 결함허용(Fault-Tolerant) 에이전트

이동 에이전트는 여러 플랫폼으로 이주하면서 실행하기 때문에 결함은 에이전트 수행 중 시스템 결함과 이주 시 네트워크에 따른 결함으로 구분할 수 있다. 이와 같은 결함의 발생으로 인해 컴퓨팅 손실시간 및 에이전트 수행시간 증가 그리고 에이전트 소멸을 초래한다. 제안하는 에이전트는 그림 3 에서 같이 결함에 대해 체크포인트 에러복구기법을 사용하여 결함에 따른 에이전트 복구시간을 줄여 전체적인 에이전트 수행 효율과 일관성을 높이고자 한다.

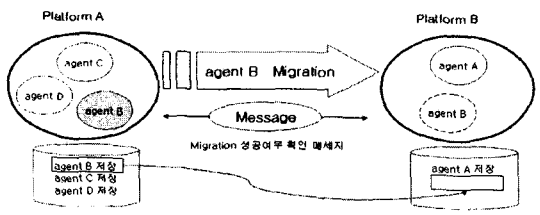

그림 4 이주 시 체크포인트 기법

결함허용 에이전트는 다음과 같은 과정으로 수행한다. 첫째, 에이전트 수행 중 시스템 결함에 대한 처리는 에이전트의 수행결과를 주기적으로 체크포인트 하여 에이전트를 재 수행하지 않고 결함 이전의 상태로 복구하는 방법을 사용하였다. 둘째, 이주 시 네트워크에 따른 결함은 이주 시 에이전트의 데이터 및 상태정보를 주기적으로 체크포인트하고 있기 때문에 안정된 디스크에 저장되어 있다. 즉 이전 플랫폼에서는 체크포인트 되어 있는 정보를 삭제하지 않고 기다렸다가, 다음 목적지 플랫폼으로 완벽한 이주가 되어 인스턴스(Instance)를 생성 한 후 실행할 준비가 되었다는 메시지를 받으면, 삭제하기 때문에 이주 시에 따른 결함을 해결할 수 있다. 이주가 완료되었다는 메시지를 받으면 이전 플랫폼에 체크포인트 되어 있는 정보를 삭제한다. 삭제하는 이유는 디스크의 활용을 높이는데 있다. 이러한 체크포인트 기법을 이용하여 에러에 대한 처리 및 복구시간을 단축하여 성능을 향상시킬 수 있다.

3.4 네트워크 부하 및 지연시간 해결방법

이동 에이전트는 플랫폼에서 플랫폼으로 이동하면서 에이전트의 임무를 수행한다. 이주 시 코드 및 데이터 그리고 상태정보를 이동하여 수행한다[6]. 에이전트의 경로가 정적 또는 동적으로 생성되면 그림 3과 같이 경로 정보에 따라 이동할 에이전트 플랫폼(메세지를 보내는 플랫폼의 다음 플랫폼을 제외한 플랫폼)에 필요한 코드 메시지를 보낸다. 메시지를 받은 플랫폼은 송신측 플랫폼에게 메시지를 받았다고 전달해 주고 필요한 코드를 원격디스커버리(Remote Discovery)을 통해 정보를 얻어 코드를 받아온 후 인스턴스를 생성한다. 다음 이동할 플랫폼에 메시지를 전달하지 않는 이유는 지금 수행 후 바로 이동해서 수행하기 때문에 성능 향상에는 효과가 없으므로 보내지 않으며, 핸드셰이크(Handshake)를 통해 메시지 전달 결과를 알 수 있다. 이동 에이전트가 이주 후 새로운 플랫폼에서 제일 먼저 프리패칭 메시지를 통해 받아온 코드가 있는가를 검사하게 된다. 메시지를 제대로 전달되었지만, 미리 코드를 받아오지 못했을 경우 필요한 코드를 다시 받아오게 된다. 이러한 프리패칭 기법을 이용하여 필요한 코드를 미리 보내어 네트워크 부하 및 네트워크 지연시간을 줄여 수행 성능을 높일 수 있다.

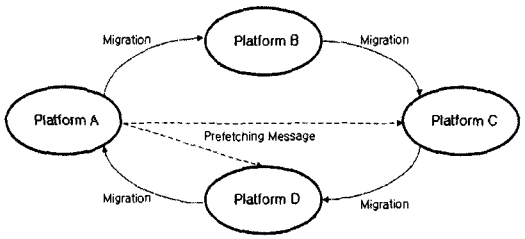

그림 5 프리패칭 메시지(Prefetching Message)

3.5 설계된 이주 기법의 특징

설계된 최적의 이주 비용을 위한 이동 에이전트는 이동 데이터 크기 및 네트워크의 상태 그리고 이동할 플랫폼의 상태를 고려하여 이주기법을 설계한다. 이러한 이주기법을 사용함으로써 다음과 같은 이점이 있다. 첫째, 각 플랫폼에서 수행 가능한 서비스

검사, 통신비용, 플랫폼 상태 그리고 자원할당 등의 비용검사를 통해 최소의 비용으로 에이전트 수행 가능한 플랫폼의 리스트를 생성하여 에이전트 생성 시 정적 경로를 구성하고, 에이전트 수행 중 네트워크 부하 및 플랫폼 자원 부족 등의 문제가 발생 시 동적 경로를 생성함으로써 실행환경변화에 적절하게 대응할 수 있으며 성능향상을 가져온다. 둘째, 코드 및 데이터 그리고 실행상태를 이주 후 새로운 플랫폼의 안정된 저장장치에 체크포인트 기법을 이용하여 에이전트 정보(코드, 데이터, 실행상태)들을 저장 후 이전 플랫폼에 이주에 대한 메시지를 보내어 플랫폼 결함이 발생하거나 이주 시 네트워크 에러에 대해 에이전트를 재 수행하지 않고 결함전의 상태로 복구하여 수행효율을 높였다. 셋째, 프리패칭 기법을 이용하여 필요한 코드를 미리 각 플랫폼에 이동하여 코드의 이동 비용(네트워크 트래픽, 네트워크 지연시간)을 최소화한다.

4. 시뮬레이션

3장에서 제안하는 방법에 대해서 기술했다. 제안한 방법과 기존의 에이전트 플랫폼에서 사용한 이주 서비스 2가지를 시뮬레이션을 통해서 비교 평가한다. 시뮬레이션 환경은 윈도우 2000서버 운영체제에 Intel Pentium(R) III 1.80GHz CPU, 256MB의 메인 메모리를 갖는 시스템이 사용되었으며, 구현은 AweSim 3.0 student version을 이용하였다. 이 실험에서 사용한 성능관련 파라미터 및 실행 파라미터는 표 1과 같다. 이주는 2번으로 정하여 실험을 수행했고 도착율(에이전트 수행시간, 시스템 부하)은 식 1의 지수분포를 따른다. 우리는 이 수식의 λ를 3 ~ 20 으로 변화시켜 가면서 실험을 수행하였다.

$$f(x) = \lambda e^{-\lambda x} \quad (식\ 1)$$

표 1 성능평가 파라미터

파라미터		값
실행 파라미터	플랫폼 수	2, 4
	네트워크 상태	지수분포(λ: 0.1 ~ 0.25)
	코드 사이즈	4KB
성능 파라미터	에이전트 도착율	지수분포(λ: 3 ~ 20)
	이주 횟수	2

그림 5 에서는 각각의 에이전트 플랫폼에서 사용하는 이주기법을 사용한 시뮬레이션 결과와 제안하는 이주기법의 시뮬레이션 결과를 보여준다. 제안하는 이주방법은 AJANTA보다 평균 88%, JAMES보다는 평균 116% 성능이 향상되었다.

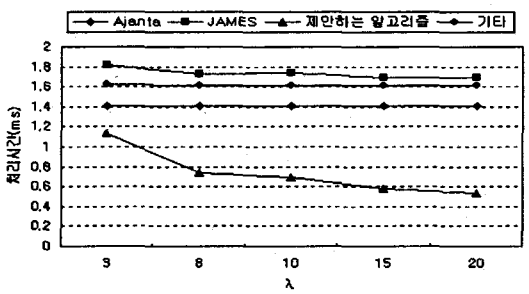

그림 6 에이전트 도착율 vs 처리시간

5. 결론

기존의 이동 에이전트 시스템의 이주는 네트워크 트래픽(Traffic) 및 일관성 있는 에이전트 이주를 보장하지 못했다. 본 논문에서는 이러한 문제점을 해결하기 위해 정적, 동적 경로를 혼합한 방법, 프리패칭 및 체크포인트 기술을 사용하여 이주성능을 향상시키기 위한 기법을 제안하였다. 제안된 이주 기법은 이동 데이터 크기 및 플랫폼 상태 그리고 네트워크 상태 변화에 따라 성능이 향상되었다.

참고문헌

[1] Kurt Rothermel and Radu Popescu-Zeletin, "Mobile Agents", Proceedings of the First International Workshop, 1997

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

[3] Luis Moura Silva and Paulo Simoes, "JAMES : A platform of Mobile Agents for the Management of Telecommunication Networks", Pro. International Workshop on Intelligent Agents for Telecommunication Applications, pp.76-95, 1999

[4] Luis Moura Silva, "Optimizing the Migration of Mobile Agents", Pro. Mobile Agents for Telecommunication Applications, 1999

[5] Anand Tripathi and Tanvir Ahmed, "AJANTA", http://www.cs.umn.edu/ajanta/

[6] Christian Erfurth, Peter Braun and Wilhelm Rossak, "Some Thoughts on Migration Intelligence for Mobile Agents", Technical Report No. 09/01, Computer Science Department, 2001