韓國精密工學會 2005 年度 秋季學術大會論文集

에이전트 모델링에서 효율적인 쓰레드 사용에 관한 연구

임선종*, 송준엽, 이승우, 김동훈(한국기계연구원)

Study on the Effective Use of Thread in Agent Modeling

S. J. Lim, J. Y. Song, S. W. Lee, D. H. Kim(KIMM)

ABSTRACT

An agent is an autonomous process that recognizes external environment, exchanges knowledge with external machines and performs an autonomous decision-making function in order to achieve common goals. The techniques for tackling complexity in software need to be introduced. That is decomposition, abstraction and organization. Agent-oriented modeling has the merits of decomposition. In decomposition, each autonomous unit may have a control thread. Thread is single sequential flow in program. The use of thread in agent modeling has an important meaning in the performance of CPU and the relation of autonomous units

Key Words : 에이전트(Agent), 모델링, 인공 지능(Artificial Intelligent), 쓰레드(Thread), 프로세스(Process), 지능 기계(Artificial Intelligence)

1. 서론

에이전트는 전체 목적을 수행하기 위해 직접적인 환경 인식, 외부 에이전트와 지식 교환 및 의사결정을 수행하는 자율적인 소프트웨어이다. 복잡한 시스템은 서브 시스템 사이의 계층적인 구조와 이들의 상호 관계로 특징지어진다. 이러한 복잡한 시스템을 다루기 위해서는 분해(decomposition), 추상(abstraction) 및 조직화(organization)에 대한 기술이 필요한데 이러한 면에서 에이전트 모델링은 장점을 가지게 된다.[1-5]

복잡한 시스템에 대해서는 자율적인 요소들로 분해하고 이들의 관계를 기술하는 것이 보다 쉬운 취급 방법일 것이다. 이때 각 자율적인 요소들은 자신의 행동을 위해 쓰레드를 가질 수 있다.

쓰레드는 프로세스내에서 제어의 단일 순차 흐름을 의미한다. 쓰레드는 하나의 프로세스 내에서 주소 공간, 코드, 전역 데이터를 공유하면서 독립적인 실행 경로를 갖는다. 프로세스 내에는 많은 쓰레드가 있을 수 있으며 각각은 고유의 레지스터 셋, 스택, 메시지 큐와 같은 입력 메커니즘을 가진다. 다중 쓰레드는 서로 다른 작업들을 수행하는 복수개의 쓰레드를 의미하며 이때 메모리를 공유하게 된다. 멀티 쓰레드에서 메모리 사용에는 많은 각 쓰레드의 중복 사용을 피하기 위해 많은 주의가 요구된다.

본 논문은 에이전트 모델링에서 쓰레드의 적합한 사용 용도에 대해 기술하고 있다. 이를 위해 변수 변환과 네트워크 기반의 통신 기능에 대해 폴링과 쓰레드를 비교하였다.

2. 에이전트 모델링과 쓰레드 특징

복잡한 시스템을 다루기 위한 소프트웨어 기술은 시스템의 분해, 추상 그리고 조직화 등이 있다. 복잡한 시스템은 대개 계층적인 구조를 이루고 있으며 서브 시스템의 상호 관계로 구별하는 것이 가능하다. 서브 시스템은 자율적이며 독립된 것으로 생각될 수 있다. 이러한 표현 방법은 복잡한 시스템에 대해서 자연스러운 것이며 의사 결정 혹은 동작 등이 자율적 요소에 기반을 두고 있다.

위와 같이 복잡한 시스템에 대한 기술 방법은 에이전트 모델링의 기술 방법과 일치한다. 복잡한 시스템에 대해 에이전트 모델링을 적용하는 기준들 중의 하나는 다음과 같다: 자율적인 행동 결정의 필요성이다. 이것은 선정된 객체가 목적 달성을 위

해 자신이 언제 그리고 어떻게 행동할 것인가를 결정하는 기능이다. 자율적인 각 객체간의 관계로 표현되는 에이전트 모델링 표현은 자율적인 서브 시스템으로 표현하는 분해 기술과 일치한다.

Table. 1 Feature of thread

의미	프로그램의 실행 단위
주요 기능	같은 메모리 공간에서 자원을 공유하면서 실행 명령어를 실행.
도입 배경	새로운 프로세스를 생성하는 것보다 생성과 종료에서 시간이 짧게 걸린다. 쓰레드 통신은 메모리를 통해 이루어지므로 시간이 짧다. (프로세스는 커널을 사용)
물리적인 표시	하나의 프로그램 또는 프로그램 집함을 가지고 있어야 함. 프로그램에 관련된 전역 변소, 지역 변수, 상수 데이터를 포함.
쓰레드 위치	한 프로세스 내의 모든 쓰레드는 같은 주소 내에 존재
구성 요소	프로그램 코드, 프로그램 카운터, 처리기 레지스터
쓰레드의 통신	모든 쓰레드는 그 프로세스의 지원과 상태를 공유. 같은 주소 공간에 존재하며 동일한 데이터에 접근. 한 프로세스 내의 쓰레드는 메모리의 데이터를 공유. 파일을 개방하는 경우도 공유.

즉 에이전트 모델링은 복잡한 시스템을 기술하는데 적합한 특성을 가진다. 에이전트는 시스템 기술에 있어서 문제 영역에 기반을 두고 있으며 동작과 제어를 위해 쓰레드를 이용한다. Agent 는 기능을 수행하기 위한 module 과 module 의 방법론인 model 로 구성된 구조를 가진다. Module 은 대화 interface, 인식(perception), 실행(execution), 사회적 지식(social knowledge), 자기 표현(self representation), 목적 표현(project representation), 지식 관리(knowledge management), 학습(learning), 추론(reasoning), 계획(planning)과 일정 관리(scheduling), 제어(control), 충돌 관리(conflict management) 등으로 이루어져 있다.[6]

프로세스는 컴퓨터의 메모리에 실행되는 프로그램이며 쓰레드는 프로세스를 이루는 기본 단위이다. 프로세스는 다수의 쓰레드로 구성되어 있다. 쓰레드는 한 프로세스 내에서 주소 공간, 코드, 전역 데이터를 공유하면서 독립적인 실행 경로를 갖는다. 쓰레드는 다른 쓰레드와 메모리를 공유하기 때문에 멀티 쓰레드에서 메모리 사용에는 주의가 요구된다. 쓰레드는 다음 점에서 사용에 이점을 제공한다. 첫째, 응용 프로그램의 일부가 비동기적이면서 병렬적으로 수행되게 한다. 둘째, 같은 기능의 멀티 클라이언트를 실행할 수 있다. 표 1 은 쓰레드의 특징을 보이며 표 2 는 프로세스의 특징을 보인다. 그림 1 은 프로세스와 스레드의 구성을 보이고 있다

Table. 2 Feature of process

의미	메모리 할당의 기본 단위
주요 기능	주기억 장치에 저장된 실행 명령어를 실행하는 것
도입 배경	시스템에서 어떠한 일이 어떠한 순서로 언제 발생할 것인지를 예측할 수 없으므로 프로세서의 개념을 도입하여 작업의 진행을 통제.
구성 요소	프로그램 코드, 프로그램 카운터, 처리기 레지스터, 스택
물리적인 표시	하나의 프로그램 또는 프로그램 집합을 가지고 있어야 함. 프로그램에 관련된 전역 변수, 지역 변수, 상수 데이터를 포함. 프로그램과 데이터를 위한 메모리가 필요. 프로시저 호출과 프로시저 매개변수 전달 과정 등을 기억할 스택이 요구.
프로세스 제어 블록	프로세스에서 사용되는 속성들의 집합
프로세스 이미지	프로그램과 데이터, 스택, 속성을 모두 포함
프로세스 위치	메모리 내의 연속된 블록에 위치되며 운영 체제가 관리
프로세스 관리	운영 체제는 프로세스 이미지의 일부를 주기억 장치 내에 유지함으로써 프로세스 관리.

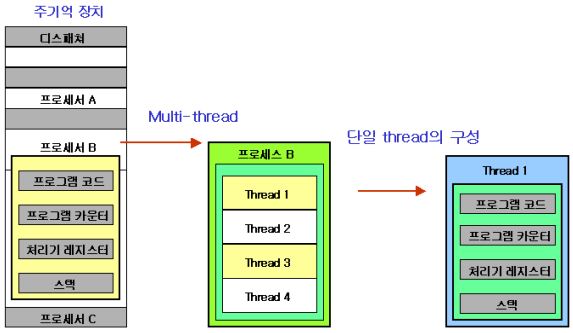

Fig. 2 Process and thread

3. 에이전트 모델링에서 쓰레드 사용

에이전트 모델링에서 쓰레드 사용의 적합성을 판단하기 위해 먼저 쓰레드와 폴링에 대해 실험을 시행하였다. 실험은 두 경우에 대해 시행하였다. 첫째, 변수 변화에 추종을 비교하였고 둘째, 네트워크 통신에서 전송된 패킷의 수신율을 비교하였다.

변수 변화의 추종은 현재 변수와 이전 변수를 설정하고 현재 변수가 계속증가 할 때 현재 변수와 이전 변수의 차가 항상 1 이 되도록 추종하는 것으로 실험하였다. 그림 2 은 쓰레드와 폴링에 대해 변수 변화 추종을 보이고 있다.

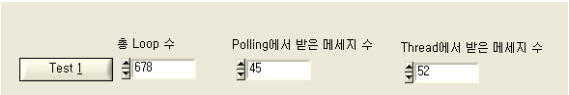

Fig. 2 Comparison thread with polling

변수 변화 추종에 대한 실험에서 폴링과 쓰레드에서 차이를 보였으며 쓰레드 방식이 변수 변화의 추종을 잘 수행함을 볼 수 있었다. 폴링과 쓰레드의 차이가 항상 일정한 것은 아니다.

또한 네트워크 상에서 전송된 패킷의 수신율을 폴링과 쓰레드에 대해 비교하였다. 그림 3 은 패킷 전송을 위해 구성된 서버이며 그림 4 는 전송된 패킷에 대한 폴링과 쓰레드의 수신율을 보이고 있다.

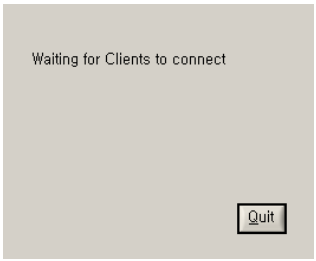

Fig. 3 Server for packet test

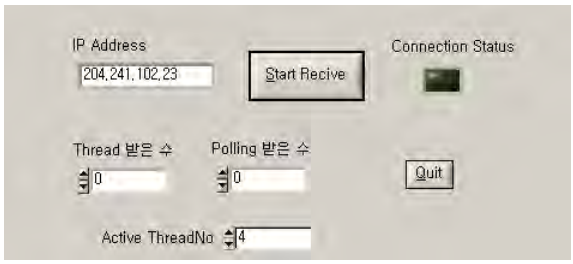

Fig. 4 Receive rate test for packet

서버는 접속된 클라이언트에 일정한 포맷의 데이터를 계속적으로 보내고 있다. 클라이언트는 폴링 방식과 쓰레드 방식으로 데이터를 수신하는데 실험 결과 쓰레드는 서버에서 전송된 데이터를 모두 수신하였으나 폴링 방식을 수신을 하지 못하였다.

위의 실험 결과 에이전트를 선정하는데 다음과 같은 경우가 적합함을 알 수 있다.

첫째, 매우 급격한 환경의 변화가 일어나며 이에 대한 실시간 감시가 요구되는 곳. 둘째, 연산에 필요한 변수가 빠르게 변하고 이에 대해 결과가 요구될 때. 셋째, 의사 결정 기능과 수행 결과가 행동으로 실시간으로 나타나야 할 경우. 넷째, 네트워크 통신과 같이 많은 클라이언트가 접속되어 데이터 전송이 요구되는 경우 등이다.

이밖에 에이전트를 선정하기 위해 다음의 기준이 이용된다.

1. 환경 변화에 대한 적응력의 필요성.
에이전트로 표현되는 객체가 환경의 변화를 계속적으로 인식할 수 있는 기능이다.
2. 협동성
목적 달성을 위해 다른 에이전트와 협력을 필요로 하는가.

4. 결론

본 연구는 에이전트의 선정에 관련된 쓰레드의 효율적인 사용 방법에 관해 수행되었으며 다음의 결론을 얻을 수 있다.

1. 급격한 환경의 변화에 대한 추종에는 에이전트 모델이 적합하며 이것은 쓰레드를 이용하는 것에 기반을 두고 있다.
2. 쓰레드를 메모리를 이용해 통신을 수행하므로 다중 쓰레드의 경우 메모리 공유에서 발생되는 문제에 대한 대책이 필요하다.
3. 서버에서 접속된 많은 클라이언트에게 데이터를 전송하는 것과 같이 빠른 응답성이 요구되는 곳에 에이전트 모델이 적합

하다.

참고문헌

1. Wooldridge, M., “Agent-based software engineering,” IEE Proceedings on Software Engineering, Vol. 144, No. 1, pp. 26-37, 1997.
2. Wooldridge, M., “Agent-based software engineering,” Software Engineering, Vol. 144, No. 1, pp. 26-37, 1997.
3. Jennings, N. R., “On agent-based software engineering,” Artificial Intelligence, Vol. 117, No. 2, pp. 277-296, 2000.
4. Jennings, N. R., “Agent Software,” Proc UNICOM Seminar on Agent Software, 1995.
5. Jennings, N. R and Bussmann, S., “Agent-based control systems: Why are they suited to engineering complex system?,” Control Systems Magazine IEEE, Vol. 23, No. 3, pp. 61-73, 2003.
6. Weiming, S., Douglas, H. N. and Jean-Paul, A. B., “Multi-Agent Systems for Concurrent Intelligent Design and Manufacturing,” Taylor & Francis, pp. 101-120, 2001.