k-결함 허용 검사점 기법을 이용한 이동 에이전트 복구 기법에 관한 연구

강수석, 변일수, 박태순
세종대학교 컴퓨터공학과
e-mail: tspark@sejong.ac.kr

A Mobile Agent Recovery Scheme based on k-Fault-Tolerant Checkpointing.

Sooseok Kang, Ilsoo Byun, Taesoon Park
Dept. of Computer Engineering, Sejong University

요 약

신뢰할 만한 이동 에이전트 시스템을 구축하기 위해서는 이동 에이전트의 결함 내성 기능이 중요하다. 지금까지 여러 결함 내성 기법이 제안되었는데, 그 중의 하나가 검사점 기법이다. 에이전트의 중간 상태를 저장하는 검사점 기법은 에이전트 복제 기법에 비해 훨씬 적은 비용을 보장하는 반면, 검사점이 저장된 사이트의 결함 발생 시 에이전트 실행이 일시 또는 영구히 중지되는 문제가 발생한다. 따라서 본 논문에서는 k-결함 허용 검사점 기법을 제안한다. 이 기법에서는 에이전트 이동 경로에 저장된 검사점을 관리하는 관찰자들을 두어, 에이전트 실행 사이트의 결함 발생 시 관찰자간의 여론 수렴 과정을 통해 결함에 영향을 받지 않은 최근 검사점으로부터 에이전트의 실행을 재개시킨다.

1. 서론

이동 에이전트란 사용자가 부여한 임무를 수행하기 위해 여러 사이트를 이동해 다니며 수행되는 프로그램이다. 이동 에이전트는 자율적으로, 그리고 비동기적으로 수행된다. 에이전트가 일단 실행되면 목적지를 스스로 설정할 수 있으며 사용자와의 상호 반응 없이 수행된다. 그러기 위해 에이전트는 프로그램 코드와 각 사이트들을 돌아다니며 수집한 자료들을 가지고 다닌다. 또한 에이전트 수행의 연속성을 위해 에이전트의 중간 상태도 같이 이동한다.

이동 에이전트가 좀더 다양한 분야에 적용되기 위해서는 신뢰할만한 에이전트의 수행이 요구된다. 에이전트는 이동 중이나 수행 중에 시스템 오류로 손실될 수 있다. 반대로, 잘못된 오류 복구는 에이전트를 중복 실행시킬 수도 있다. 신뢰할 만한 에이전트 시스템은 에이전트가 아무 오류 없이 정확히 한번(exactly-once) 수행되는 것을 보장해야한다.

신뢰할 만한 에이전트 시스템을 위해 제안된 결함 내성 기법들은 크게 복제 기법[1,2,3]과 검사점 기법[4,5,6]으로 분류된다. 복제 기법에서는 각 단계마다 에이전트의 복제 본을 여러 사이트에 전송하며 중복 수행을 막기 위해 분산 트랜잭션[7], 리더 선출 절차[3], 혹은 동의 절차[1,2]를 거치게 된다.

복제 기법의 단점은 에이전트의 사이즈가 크거나 네트워크 비용이 비싼 환경에서는 비효율적이라는 점이다. 이에 반해, 검사점 기법에서는 에이전트를 하나의 사이트에만 보내며 미래의 오류를 대비해 에이전트의 중간 상태를 방문한 사이트에 저장한다. 오류가 발생할 경우 저장된 상태로부터 에이전트를 복구시킨다. 이 기법은 오류가 발생한 사이트가 복구될 때까지 에이전트 실행이 중단된다는 단점을 가지고 있다.

복제 기법은 2k+1개의 에이전트 복제 본을 전송하였을 경우, k개의 오류를 견딜 수 있는 반면, 검사점 기법에서는 하나의 오류가 심각한 지연을 발생시킬 수 있다. 하지만 오류가 없을 경우, 전체 에이전트의 실행 시간을 고려해 볼 때 검사점 기법이 복제

기법에 비해 훨씬 나은 성능을 보인다. 복제 기법은 에이전트의 복제와 동의 과정에 따르는 부하가 존재하기 때문이다. 본 논문에서 제안되는 k-결함 허용 검사점 기법은 검사점 기법의 저 비용을 보장하면서 기존 검사점 기법에서 발생하는 에이전트 실행 봉쇄를 없앰으로서 신뢰도를 향상시킬 수 있는 기법이다.

k-결함 허용 검사점 기법에서는 에이전트의 이동 경로 상에 저장된 검사점들을 관리하기 위해 관찰자 에이전트를 둔다. 관찰자 에이전트는 해당 에이전트가 실행 중인 사이트에 결함 발생 시 저장된 검사점을 이용하여 저장된 상태로부터 에이전트의 실행을 재개시키는 역할을 한다. 이때, 여러 관찰자에 의해 동시에 여러 상태의 에이전트가 실행 재개되는 것을 막기 위해, 관찰자간 여론 수렴 과정을 통해, 결국 한 에이전트만이 실행 재개되도록 하였다.

또한, k개의 결함 가능성을 고려하여, 실행 중인 에이전트 외에 2k개의 관찰자 에이전트를 설정하였으며, 이중 k개는 결함 발생 시 실행 재개를 위해 사용되며 나머지 k개의 관찰자들은 단순히 동의 과정에만 참여하면 된다. 따라서 제안된 기법은 기존 검사점 기법과 같은 적은 비용으로 k개의 결함에 대처할 수 있는 높은 신뢰도를 갖게 된다.

2. 이동 에이전트 시스템 모델

이동 에이전트 시스템은 네트워크로 연결된 여러 개의 사이트로 구성되어 있다. 각 사이트는 에이전트의 실행환경을 제공하는 하나 이상의 플레이스(place)를 제공한다. 플레이스는 에이전트의 이동과 등록, 실행을 책임지며, 각 사이트에서 제공하는 데이터에 대한 접근을 제어하는 역할도 한다. 다시 말해, 에이전트는 플레이스에서 실행되며, 플레이스들 사이를 이동해 다닌다. 한 플레이스에서의 에이전트 실행을 하나의 스테이지(stage)라고 부른다. *SGij*는 이동 에이전트 *MAi*의 *j*번째 스테이지를 나타내며, 에이전트의 실행은 일련의 스테이지와 스테이지간의 이동으로 볼 수 있다.

에이전트 시스템에서 발생할 수 있는 오류에는 다음과 같은 세 가지가 있다

- **에이전트 오류**: 에이전트에 오류가 발생했으며 실행을 중단한다.
- **플레이스 오류**: 플레이스에 오류가 발생했으며 실행을 중단한다. 플레이스에서 실행중인 모든 에이전트가 중단된다.
- **시스템 오류**: 시스템에 오류가 발생한 경우로, 시스템 상에서 제공되는 모든 플레이스의 서비스가 중단된다.

검사점 기법을 사용하는 결함 내성 이동 에이전트 시스템에서는 위와 같은 오류가 발생하더라도, 초기 상태에서 다시 시작할 필요가 없다. 에이전트 이동 경로 상에 있는 관찰자 중 하나가 결함을 발견하고, 저장된 검사점으로부터 수행을 재개한다.

3. k-결함 허용 검사점 기법

3.1 검사점 기법

검사점은 에이전트의 실행 상태를 주기적으로 안전한 저장소에 저장함으로써, 시스템내의 결함 발생 시 에이전트를 저장된 상태로부터 재개시키는 역할을 한다. 이동 에이전트 시스템에서는 에이전트의 복구를 지역화하기 위해, 각 스테이지의 시작 지점에서 에이전트의 상태를 저장한다. 이 경우 해당 사이트의 결함 발생 후, 복구 과정에서 에이전트는 저장된 상태로부터 복구되어, 해당 사이트에 에이전트가 도착한 직후의 상태로부터 실행을 재개한다. 그러나 해당 사이트가 결함으로부터 신속히 복구되지 못하는 경우, 에이전트의 복구가 늦어지고, 최악의 경우 에이전트가 손실될 우려가 있다.

검사점을 설정하는 다른 방법은 각 스테이지의 끝 지점에서 검사점을 설정하는 방법이다. 이 경우, 검사점은 해당 사이트의 결함이 아니라, 다음 스테이지가 실행되는 사이트의 결함 발생 시 이용된다. 즉, 다음 스테이지에서의 에이전트의 결함이 발견되면 저장된 상태로부터 에이전트를 재개한다. 이 경우, 에이전트는 결함이 발생한 사이트의 복구를 기다려 에이전트를 재전송 할 수도 있지만, 결함 사이트의 복구가 늦어지는 경우, 대체 사이트로 에이전트를 전송하여, 에이전트 실행이 지연되거나 손실되는 것을 막을 수 있다. 따라서 본 논문은 스테이지 끝에서 검사점을 설정하는 기법을 따른다.

3.2 k-결함 허용 검사점

각 스테이지의 끝 지점에서 검사점을 설정하는 경우, 다음 사이트의 결함 발생을 관찰하고, 저장된 검사점으로부터 에이전트를 재개시키는 작업을 수행할

관찰자(observer) 에이전트가 필요하다. 그러나 에이전트와 관찰자 에이전트 양쪽 모두 오류가 발생 한 경우에는 에이전트의 손실이 불가피해진다. 따라서 k개의 결함 극복을 보장하기 위해서는 적어도 k개의 관찰자가 필요하게 된다. 본 논문에서는 관찰자 에이전트의 생성에 따른 비용을 최소화하기 위해, 에이전트의 이동 경로에 관찰자를 두기로 한다.

대부분의 이동 에이전트 시스템에서는 에이전트의 이동 시, 먼저 에이전트를 복사하여 원격 메소드 호출(RMI)등의 방법으로 복제 에이전트를 이동 시키고, 이동 성공 시 남아 있는 에이전트를 제거한다. 이때 남아있는 에이전트를 제거하지 말고, 관찰자 에이전트로 활용하게 되면, 관찰자 에이전트 생성에 따른 비용을 줄이 수 있다. 관찰자 에이전트는 에이전트가 이동 직전 생성한 검사점을 관리하며, 다음 스테이지에서의 에이전트의 결함이 발견되면 검사점으로부터 에이전트를 재개시키는 역할을 한다.

그림 1은 에이전트의 이동 경로 상에 N개의 검사점과 N개의 관찰자 노드를 설정한 경우를 보인다. 그림상의 MAi는 i가 이름인 이동 에이전트를, $Oi1$과 $Oi2$는 MAi의 두 관찰자 에이전트를 나타내며, $SGij$는 MAi의 j번째 스테이지를 나타낸다. 그림상의 검은 상자는 MAi의 검사점을 나타내며, 그림은 N값이 2인 경우이다.

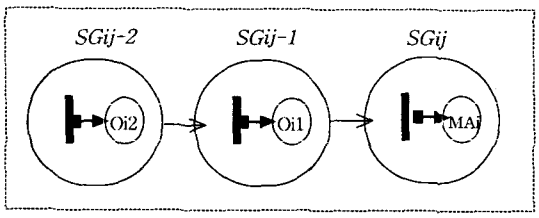

그림 1. 이동 에이전트와 관찰자 에이전트

에이전트의 이동 경로 상에 N개의 관찰자 에이전트를 유지하기 위해, 각 관찰자는 카운터 변수인 O_count의 값을 관리한다. 또한 에이전트는 각 사이트에서의 실행이 끝나고 검사점을 설정 한 후 다음 사이트로 이동하면, 현재의 관찰자 노드들에게 M_exec_end 메시지를 보낸다. 이 메시지를 수신한 관찰자 노드는 자신의 O_count 값을 1 만큼 증가시키고, 이 값이 N보다 커진 관찰자 에이전트는 스스로 실행을 중단한다.

본 논문에서는 에이전트 실행 사이트의 결함 발견을 위해 소요되는 비용을 줄이기 위해 타임아웃을 사용한다. 즉, 각 관찰자 에이전트는 에이전트가 j번째 스테이지로의 이동 후 일정한 시간이 지나도록 M_exec_end 메시지가 오지 않으면, 에이전트의 현재 실행 사이트에 결함이 발생하였음을 가정하고, 각 관찰자가 관리하는 검사점으로부터의 복구를 수행한다.

이때, 관찰자들의 복구 시점이 적절히 조정되지 못하면, 다음과 같이 다중 실행(duplicate execution)의 문제가 발생 할 수 있다. 첫 번째 경우는 부적절한 타임아웃 값에 의해 단순히 진행이 느린 에이전트를 결함 발생 한 에이전트로 오인하는 경우이다. 이 경우 실행 중인 에이전트와 관찰자 에이전트가 중복하여 실행되게 된다. 두 번째 경우는 여러 관찰자 에이전트들이 동시에 에이전트의 결함을 발견하고 복구를 시도하는 경우이다. 이 경우 여러 관찰자 에이전트에 의해 서로 다른 상태의 검사점으로부터 실행 재개가 이루어진다.

이러한 다중 실행의 문제를 해결하기 위해 널리 사용되는 방안으로는 여론 수렴(consensus) 기법이 있다. 여론 수렴 기법은 에이전트 복제를 이용한 결함 내성 시스템의 설계 시 복제 에이전트간의 실행 조정을 위해 사용되는 기법으로 이동 에이전트와 관찰자 에이전트로 이루어진 본 기법에서도 동일한 방법으로 사용될 수 있다. 에이전트 복제기법을 위한 여론 수렴은 여러 방법이 제안되었지만, 본 논문은 에이전트 간 우선순위를 활용한 여론 수렴 방법[3]을 따른다.

즉, 각 관찰자 에이전트의 O_count값이 각 관찰자의 우선순위를 나타내도록 하며, 실행 중인 에이전트의 오류 발생 시 가장 우선순위가 높은 관찰자가 오류 발견 사실을 다른 관찰자들에게 알리고, 응답 메시지를 받은 후 오류가 발생한 에이전트의 실행을 떠맡게 된다. 우선순위가 j번째인 관찰자는 $j-1$의 우선순위를 갖는 관찰자 에이전트의 결함 발생 시에만 복구를 시도한다.

이때 여러 관찰자들의 다중 결함 상황에서도 에이전트의 단일 실행(exactly-once execution)을 보장하기 위해, 에이전트는 각 스테이지 실행 후 투표(voting)에 의한 여론 수렴 과정을 거친다. 즉, 에이전트는 한 스테이지에서의 실행이 끝나면 검사점 설정 후 모든 관찰자들에게 M_vote 메시지를 보내고, 이 메시지를 받은 관찰자들은 M_vote_reply 메시지를 보낸다. 과반수이상의 M_vote_reply 메시지를 받

은 에이전트만이 성공적으로 다음 스테이지에 진입하게 되고, 중복 실행된 다른 에이전트들의 실행 결과는 해당 사이트에서 적절히 무효화(undo)가 된다. 그림 2는 4개의 관찰자를 갖는 시스템에서 에이전트와 관찰자간의 여론 수렴 과정의 예를 보여준다.

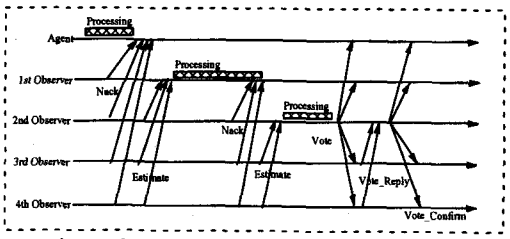

그림 2. 에이전트와 관찰자간 여론 수렴 과정

에이전트 및 관찰자에 대한 k개의 결함 발생 상황에서도, 위와 같은 투표에 의한 여론 수렴 과정이 성공적으로 수행되기 위해서는 시스템 내에 적어도 2k개의 관찰자가 필요하다. 이 가운데, 우선순위가 높은 k개의 관찰자는 검사점을 관리하며 에이전트와 자신 보다 높은 우선순위 관찰자의 결함 발생 시 실행 재개를 위한 것이며, 나머지 k개의 관찰자는 여론 수렴 과정에 참여하기 위한 것이다. 이와 같이 2k+1개의 에이전트 및 관찰자를 이용하면, k개의 결함 발생 시에도, 적어도 한 관찰자는 과반수의 투표를 얻어 실행을 계속할 수 있게 된다.

4. 결론 및 향후과제

본 논문에서는 검사점을 기반으로 하는 결함 내성 기능을 갖춘 이동 에이전트 시스템의 설계 방법을 제안하였다. 기존의 검사점 기법은 에이전트 복제 기법과 비교하여 저 비용의 장점을 가지지만, 단일 결함에 의한 에이전트 손실의 가능성이 있어, 신뢰도가 떨어진다고 볼 수 있다.

이에 반해, 본 논문에서 제안된 기법은 실행 중인 에이전트 외에 여러 관찰자 에이전트를 두어, 결함 발생 시 이전에 설정된 검사점으로부터 실행 재개가 가능하도록 하였다. 또한, 에이전트 복제 기법에서 사용하는 여론 수렴과 비슷한 과정을 거쳐 k개의 결함을 극복하도록 하였다. 현재, 본 논문에서 제안된 기법은 이동 에이전트 시스템인 Aglet[8] 시스템 상에 구현 중이며, 곧 정확한 성능 평가가 이루어지리라 본다.

Acknowledgements

본 연구는 한국과학재단 목적기초연구 지원으로 수행되었음 (과제번호: R04-2002-000-20102-0).

참고문헌

[1] S. Pleisch and A. Schiper, "Modeling Fault-Tolerant Mobile Agent Execution as a Sequence of Agreement Problems," *Proc. of the 19th Symp. on Reliable Distributed System, pp. 11-20,* 2000.

[2] S. Pleisch and A. Schiper, "FATOMAS-A Fault Tolerant Mobile Agent System Based on the Agent Dependent Approach," *Proc. of the Int'l Conf. on Dependable Systems and Networks,* pp. 215-224, 2001.

[3] M. Strasser and K. Rothermel, "Reliability Concepts for Mobile agents," *International Journal of Cooperative Information Systems,* Vol. 7, No. 4, pp. 255-283, 1998.

[4] M. Dalmeijer, E. Rietjens, D. Hammer, A. Aerts and M. Soede, "A Reliable Mobile Agents Architecture," *Proc. of the Int'l Symp on Object-Oriented Real-time Distributed Computing,* 1998.

[5] E. Gendelman, L.F. Bic, and M.B. Dillencourt, "An Application-Transparent, Platform-Independent Approach to Rollback-recovery for Mobile Agent Systems," *Proc. of the 20th Int'l Conf. on Distributed Computing Systems,* 2000.

[6] M. Strasser and K. Rothermel, "System Mechanism for Partial Rollback of Mobile Agent Execution," *Proc. of the 20th Int'l Conf. on Distributed Computing Systems,* 2000.

[7] H. Vogler, T. Kunkelmann and M.L. Moschgath, "An Approach for Mobile Agent Security and Fault tolerance using Distributed Transaction," *Proc. of the Int'l Conf. on Parallel and Distributed Systems,* pp. 268-274, 1997.

[8] G. Karjoth, D.B. Lange and M. Oshima, "A Security Model for Aglets," *IEEE Internet Computing,* 1997.