정보검색 시스템의 이동 에이전트 결함허용 처리 기법

Sun-Jung Kim[a] and 강대욱[b]

[a]Department of Computer Science, Chonnam National University
300 Yongbong-dong, Buk-gu, Gwangju 500-757, Korea
Tel:82-62-530-5114, Fax:82-62-530-1139, E-mail:kamlove1@nate.com

[b] 전남대학교 전산학과
광주 북구 용봉동 300, 500-757
Tel:82-62-530-5114, Fax:82-62-530-1139, E-mail: dwkang@chonnam.ac.kr

Abstract

원격 에이전트 서버는 에이전트 개발자가 제어권한을 가지지 못한다. 원격 에이전트 서버 중 일부가 다운된 상황에서는 에이전트를 정상적으로 동작시키기 어렵다. 이런 경우 일반적으로 이용하는 방법은 이동 에이전트가 각 에이전트 서버에 도착하면 복제본을 저장해 두는 것이다. 그러나 서버 고장 상태로부터 온라인 상태로 돌아올 때까지 복제본을 이용할 수 없게 되므로 적합한 해결책이 되지 못한다. 본 논문은 에이전트 서버 고장시 이동 에이전트가 계속 동작할 수 있는 결함허용 처리기법을 제안한다. 제안 기법은 이동 에이전트를 생성하는 홈 에이전트 서버와 이동 에이전트가 방문한 이전 에이전트 서버에서 수행되며 원격 에이전트 서버에 고장이 발생한 경우 제안기법을 통해 결함허용을 처리하며 실험을 통해 제안 기법에서의 요소별 오버헤드를 분석한다.

***Keywords**: fault tolerance, mobile agents*

1. Introduction

분산 시스템은 동기화, 동시성(병행성), 분산처리를 수용해야 함으로 복잡한 시스템이다. 이동 에이전트는 자동화되고 보안 공격에 개방되어 있고(악의를 가진 에이전트나 호스트에 의해 직접), 에이전트 서버 고장, 자원의 검색 실패 등의 새로운 문제점을 갖는다[1]. 원격 에이전트 서버 고장으로 인한 이동 에이전트 소멸은 이동 에이전트 결함허용 연구에 있어서 주요 관심사이다[1][2][3].
이동 에이전트는 다음과 같은 상황이 발생하였을 때 소멸될 우려가 있다.

□ o 이동 에이전트가 실행 중인 에이전트 서버에 고장이 발생할 때

□ o 이동 에이전트가 자식에이전트를 생성시키고 다른 에이전트 서버로 이주한 경우에 자식 에이전트는 실행을 완수하여 부모 에이전트에게 결과를 돌려주려고 하는데 에이전트 서버 고장으로 인해 통신이 불가능하여 결과를 돌려줄 수 없는 경우

□ o 이동 에이전트는 자식 에이전트를 생성하고 원격 에이전트 서버로 이주한다. 이때 자식 에이전트가 위치한 원격 에이전트 서버가 고장나면 자식 에이전트는 소멸된다. 이러한 경우 부모 에이전트는 자식 에이전트가 돌아올 때까지 무한정 계속해서 기다리게 된다.

사용자가 에이전트를 항공사에 전송하여 서울에서 광주행의 가장 싼 티켓을 구매하려 한다고 가정해보자. 이때 이동 에이전트는 각 항공사의 원격 에이전트 서버로 이동하여 현재 구한 정보 중 가장 싼 티켓 정보를 반영하기 위해 자신의 내부 상태를 갱신해야 한다. 그런데 만일 이동 에이전트가 실행하는 동안 에이전트 서버에 고장이 발생하면 가장 싼 티켓에 관련한 모든 정보를 잃게 된다. 일부 어플리케이션은 에이전트 서버의 고장에 영향을 받게 된다. 즉, 이동 에이전트가 수행되면 에이전트 내부 상태와 에이전트 서버들의 상태를 모두 수정한다. 이러한 경우 결과적으로 에이전트 서버에 트랜잭션 처리가 필요하게 된다. 하지만 이동 에이전트 정보 검색 어플리케이션에는 트랜잭션 처리가 필요하지 않다. 왜냐하면 정보검색 어플리케이션에는 이동 에이전트와 원격 에이전트 서버 사이에 상태 종속성이 없기 때문이다. 결과적으로 트랜잭션 기반 해결책은[1,2,5,6] 불필요한 성능 오버헤드를 일으킨다. 일부 기술들은[7] 결함허용을 위해 에이전트 서버 플랫폼을 수정한다. 즉, 에이전트 서버는 실행 전에 각 이동 에이전트를 복제할 수 있다. 그러나 이 기술상에서 동작되는 정보 검색 시스템은 다른 종류의 에이전트 서버에서 정보를 이용하는 경우

문제를 야기한다. 에이전트 서버 플랫폼을 수정하여 결함허용을 처리하는 경우는 수정된 에이전트 서버 플랫폼을 허용하는 엔터프라이즈로 정보 검색이 제한된다.

Vogler[6]는 이동 에이전트가 에이전트 서버에 도착하면 복제본을 만드는 기법을 제안하였다. 그러나 에이전트 서버에 고장이 발생하면 복제본은 이용할 수 없게 된다. 이 논문은 에이전트 서버에 고장이 발생했을 때 이동 에이전트가 계속 동작할 수 있도록 결함허용 기법을 제안하고 실험을 통해 제안기법에서의 요소별 오버헤드를 분석한다.

본 논문은 다음과 같이 구성되어 있다.

2절은 관련연구이고 3절은 제안기법을 기술하고 4절은 시뮬레이션 프로그램을 통해 제안기법을 구현하고 실험을 통해 요소별 오버헤드를 분석한다. 5절은 결론 및 향후 연구를 기술한다.

2. 관련연구

에이전트 서버 고장 발생시 기존 대안기법[2][4][5]은 적절한 서비스를 제공한 에이전트 서버들에 각 이주 단계에서 이동 에이전트를 복제한다. 에이전트 서버 고장으로 이동 에이전트가 소멸되면 투표(voting) 알고리즘을 동작시켜 복제본 중 새로운 리더를 선출한다. 복제본은 다른 에이전트 서버에서 즉시 이용 가능하기 때문에 회복하는 작업의 성능이 향상된다. 그러나 새로운 리더를 선출하고 이주 경로상의 각 단계에 있는 여러 에이전트 서버에 복제본을 전송함으로써 오버헤드는 여전히 발생한다.

3. 제안기법 설계

원격 에이전트 서버 고장을 처리하기 위해 *primary* 에이전트 *PA*와 *secondly* 에이전트 *SA*를 생성한다. 는 홈 에이전트 서버 *HAS*에서 생성되고 이주 경로에 따라 각 호스트에서 task *T* 를 수행한다. *PA*는 첫 번째 에이전트 서버 RAS_i로 이주하기 전에 홈 에이전트 서버에 SA_{home}을 생성하고 이주 경로상의 첫 번째 에이전트 서버 RAS_1에서 작업을 수행한다. *PA*가 이주 경로상에 있는 다음 호스트로 이주하기 전에 즉, RAS_{i+1}로 이주하기 전에 *PA*는 RAS_i에 SA_i를 생성하고 SA_{home}에게 *die*메시지를 보낸다. SA_i는 *PA*에게 *die* 메시지를 받을 때까지 에이전트 서버 RAS_{i+1}에 계속해서 ping을 보낸다. 그림 1은 제안기법의 디자인 알고리즘이다.

*SA*는 *PA*로부터 *die* 메시지를 받으면 소멸된다. *PA*는 에이전트 서버 RAS_{i+1}에서 실행을 완료하고 에이전트 서버 RAS_{i+2}를 모니터하기 위해 새로운 복제본 SA_{i+1}를 생성시킨다.

```
// agent task
 search()
 // mobile agent execution
 if primary && atHome()
   createSecondly()
   primary.location = itinerary.next()
   if isPrimary
     secondly.location = itinerary.next()
     ping(secondly.location.isAlive)
     search()
     createSecondly()
     send(die)
     primary.location =  itinerary.next()
 else
   ping(primary.isAlive)

 // monitor primary agent
 ping(primary.location.isAlive)
 while(alive && !receive(die))
   if !alive
     createSecondly()
     primary.location = itinerary.next()
 // timeout
 while !dispatch_list.isEmpty()
 {
   wait(t)
   if returnHome
     remove(MA, dispatch_list)
   resend(dispatch_list)
 }
```

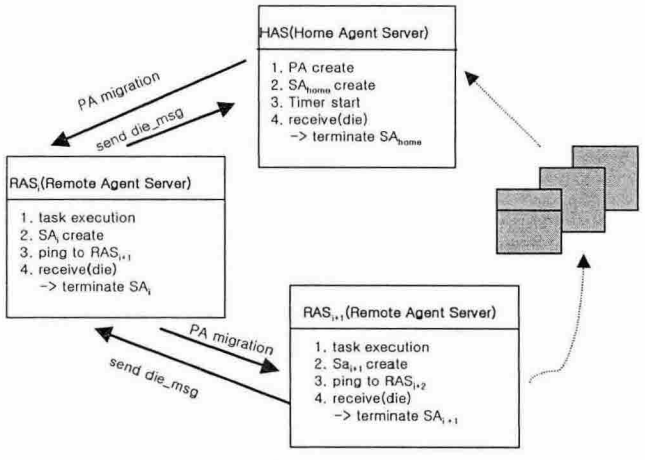
[그림 1] primary/secondly agent 알고리즘

*PA*가 에이전트 서버 RAS_{i+1}의 고장 발생으로 인해 소멸되는 경우를 가정해보자. 예를 들면 에이전트 서버 RAS_{i+1}에 *PA*가 이주하거나 또는 실행하기 전에 고장이 발생할 수도 있다. RAS_i에서의 SA_i가 *PA*의 고장을 탐지하는 경우 새로운 복제본 SA_i를 생성하고 에이전트 서버 RAS_{i+2}를 방문하게 한다. 결과적으로 SA_i는 새로운 *PA*가 된다.

*PA*는 RAS_{i-1}의 SA_i에게 ping을 보내고 동시에 task *t*를 수행한다. 문제가 발생하지 않는 경우에 *PA*는 실행을 완료하고 다음 에이전트 서버 RAS_{i+1} 을 모니터 하기 위해 새로운 복제본 SA_i를 생성한다.

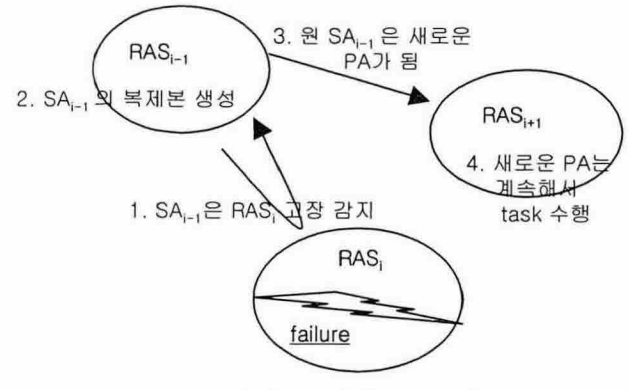
*PA*가 이주하기 전에 에이전트 서버 RAS_{i-1}에 SA_{i-1}를 끝내기 위해 *die* 메시지를 보낸다. 그림 2는 각 에이전트 서버에서 수행되는 작업의 동작 순서를 나타낸다. *PA*가 *SA*의 고장을 탐지하면 *PA*는 다시 *SA*를 생성하여 이전 에이전트 서버 RAS_{i-k}에 *SA*를 보낸다. *PA*가 다음 에이전트 서버로 이주하기 전에 RAS_{i-k}에 있는 *SA*를 끝내기 위해 *die* 메시지를 보낸다. 그림1의 알고리즘을 살펴보면 *PA*가 홈 에이전트 서버에 있는 경우 *SA*를 생성하고 첫 번째 호스트로 이주한다.(*primary.location = itinerary.next()*)

홈에 있지 않으면 다음 호스트로 이주하기 전에 새로운 *SA*를 생성하고 이전 *SA*에게 *die* 메시지를 보내서 종료시킨다.



[그림 2] 제안기법의 동작 순서

*PA*는 *SA*가 작업중인 현재 에이전트 서버를 모니터하기 위해 ping을 보낸다. *PA*가 동작 중이고 메시지가 처리되지 않으면(*alive && !receive(die)*) 계속 ping을 보낸다. *PA*가 고장이 감지되면(*!alive*) *SA*는 새로운 *SA*를 생성시키고 자신이 새로운 *PA*가 된다. 그렇지 않으면 *SA*는 *die* 메시지를 받고 소멸된다. 그림 3은 원격 에이전트 서버나 *PA*의 고장을 *SA*가 감지하는 경우 처리되는 작업을 나타낸다.



[그림 3] 에이전트 서버 또는 에 고장이 발생한 경우

복제본을 *PA*의 이전 에이전트 서버에서 사용할 수 있으므로 현재 *PA*가 작업을 수행하는 에이전트 서버에 고장이 발생하더라도 이전에 방문했던 다른 에이전트 서버들에 재방문할 필요가 없다. 그러나 다음 에이전트 서버로 이주하기 전에 *PA*가 매번 *SA*를 생성해야 하므로 이동 에이전트에게 더 큰 실행 오버헤드를 줄 수도 있다. 또한 primary와 secondly 에이전트가 동시에 고장을 발생시킬 때 대처할 수 있도록 타임아웃을 지정한다. 타임아웃 지정은 이동 에이전트나 원격 에이전트 서버에 별다른 오버헤드를 추가하지 않고도 결함허용이 가능한 장점이 있다. 하지만 프로세서 속도와 통신 지연의 정확한 값이 정해져 있지 않기 때문에 비동기 시스템에서 타임아웃 값을 결정하는 것이 어렵다.

4. 실험

본 논문의 제안기법은 IBM Aglets V2.0.2 상에서 자바 개발 도구인 J2SDK 1.4.2와 MySql 3.2.3을 사용하여 구현하였고 Windows XP에서 동작한다. 실험은 상품 구매를 위해 이동 에이전트가 이주경로에 따라 상품 정보를 검색하는 중 한 에이전트 서버에서 고장이 발생했을 때 이주를 마치고 최적의 구매 정보를 가지고 홈 에이전트 서버로 돌아와 알리는데 소요되는 시간을 측정한다. 이때 *SA* 에이전트 생성 시 소요되는 시간과 *PA* 에이전트의 고장을 탐지하는데 소요되는 시간, 그리고 고장 발생시 *SA*를 생성하여 손실된 작업을 회복하는데 소요되는 시간을 측정하여 평균값을 통해 오버헤드를 분석하였다.

측정 요소	소요시간(s)
SA 에이전트 생성 시간	0.8
PA 에이전트의 고장 탐지 시간	7.5
고장 발생시 *SA* 생성 및 작업 회복 시간	13.4

[표 1] 오버헤드 분석

*SA*는 *PA*가 이주할 다음 에이전트 서버로 ping을 보낸다. 이때 *SA*가 *PA*의 에이전트 서버 고장을 감지하는 데 걸리는 시간은 평균 7.5초이다. *SA*를 생성하고 동작시키는 데 걸리는 시간은 평균 0.8초이다. 그리고 *SA*가 *PA*의 고장으로 자신의 복제본을 생성하고 자신은 *PA*로 전환되기 위해 수행되는 작업에 걸리는 시간은 평균 13.4초이다.

5. 결론

본 논문에서는 한 쌍의 에이전트를 동작시켜 이동 에이전트가 작업을 수행하는 원격 에이전트 서버에 고장이 발생한 경우 손실된 작업 또는 에이전트를

회복하고 계속해서 정상적인 동작이 가능하도록 결함 허용을 처리하였다.

본 논문에서의 구현은 간단한 환경 조건을 사용하여 기술되었기 때문에 더욱 엄격한 조건의 실패 모델에서도 정상적인 동작을 수행할 수 있도록 더욱 강화 된 모듈의 연구가 필요하다.

References

[1] L.M.Silva, V.Batista and J.G.Silva, "Fault-Tolerant Execution of Mobile Agents." in Proc. International Conference on Dependable Systems and Networks, New York, June 2000, pp.144-153.

[2]M.Strasser, K.Rothermel and C.Maihofer, "Providing Reliable Agents for Electronic Commerce," in Trends in Distributed Systems for Electronic Commerce (TREC'98), LNCS 1402, Springer-Verlag, 1998, pp.241-253.

[3] F.Schneider, "Towards Fault-Tolerant and Secure Agentry," in Proc. 11th International Workshop on Distributed Algorithms, Saabrucken, September 1997, pp.1-14.

[4] S.Pleisch and A.Schiper, "Modeling Fault-Tolerant Mobile Agents as a Sequence of Agreement Problems," in Proc. 19th Symposium on Reliable Distributed Systems(SRDS), Nuremberg, October 2000, pp.11-20.

[5] F.M.Silva and R.Popescu-Zeletin, "Mobile Agent-Based Transactions in Open Environments," IEICE Transactions on Communications, E83-B(5), pp.973-987, 2000

[6] H.Vogler, T.Hunklemann and M.Moschgath, "An Approach for Mobile Agent Security and Fault Tolerance Using Distributed Transactions," in Proc. International Conference on Parallel and Distributed Systems(ICPADS'97), Seoul, December 1997, pp.268-274

[7] Mohindra, A. Purakayastha and P. Tahiti. "Exploiting Nondeterminism for Reliability of Mobile Agent Systems," in Proc. International Conference on Dependable Systems and Networks, New York, June 2000, pp.144-153