멀티에이전트 환경에서 결함 포용 정보의 쓰레기 처리 기법

(Garbage Collection Protocol of Fault Tolerance Information in Multi-agent Environments)

이 대 원 † 정 광 식 †† 이 화 민 ††† 신 상 철 †

(Dae won Lee) (Kwang Sik Chung) (Hwa Min Lee) (Sang Chul Shin)

이 영 준 †††† 유 헌 창 ††††† 이 원 규 †††††

(Young Jun Lee) (Heon Chang Yu) (Won Gyu Lee)

요 약 분산 시스템에서는 단일 시스템보다 높은 결함 발생 확률을 가지기에 기존의 많은 연구에서는 분산 시스템에서 결함 발생에 대한 많은 결함 포용 기법들이 연구되어 왔다. 하지만 저장된 결함 포용 정보의 증가에 따른 저장 공간의 부족으로 인해 전체 시스템 성능의 저하를 가져오게 하였다. 시스템 성능의 저하를 막기 위하여 불필요한 결함 포용 정보의 삭제가 필요하게 되었고 이 논문에서는 결함 포용 정보의 쓰레기 처리를 위한 방법을 제안한다. 이에 본 논문에서는 결함 포용 정보의 쓰레기 처리를 담당하는 쓰레기 처리 에이전트, 결함 포용 정보를 유지 관리하는 정보 에이전트, 그리고 전체 에이전트간의 통신 기능을 담당하는 조정 에이전트를 정의 및 설계하고, 쓰레기 처리 에이전트를 이용한 쓰레기 처리 알고리즘을 제안한다. 복귀회복 기법은 독립 검사점(independent checkpoint)기법과 송신자 기반 비관적 메시지 로깅(sender based pessimistic message logging)기법을 사용한다. 제안된 쓰레기 처리 기법에서의 쓰레기 처리, 정보, 조정 에이전트는 프로세스와 동시에 생성되며 정보 에이전트에 프로세스에서 발생하는 검사점과 비결정적인 사건들에 대한 로깅 정보들을 영역 지식으로 구축한다. 그리고 쓰레기 처리 에이전트는 쓰레기 처리 시점을 선정하고 정보 에이전트와 조정에이전트의 협력을 통하여 영역 지식에 구축된 불필요한 결함 포용 정보의 쓰레기 처리를 한다. 제안한 에이전트를 이용한 쓰레기 처리기법의 타당성 증명을 위하여 결함을 발생시켜 복귀 회복 후 쓰레기 처리를 하는 시스템과 하지 않는 시스템의 영역지식을 비교하여 같은 결과를 같는지의 여부를 검사한다.

키워드 : 멀티에이전트, 쓰레기처리, 결함포용 정보, 복귀회복

Abstract Existing distributed systems have higher probability of failures occurrence than stand-alone system, so many fault tolerant techniques have been developed. Because of insufficient storage resulting from the increased fault tolerance information stored, the performance of system has been degraded. To avoid performance degradation, it needs delete useless fault tolerance information. In this paper, we propose a garbage collection algorithm for fault tolerance information. And we define and design the garbage collection agent for garbage collection of fault tolerance information, the information agent for management of fault tolerant data, and the facilitator agent for communication between agents. Also, we propose the garbage collection algorithm using the garbage collection agent. For rollback recovery, we use independent checkpointing protocol and sender based pessimistic message logging protocol. In our proposed garbage collection algorithm, the garbage collection,

· 본 연구는 한국과학재단 목적기초연구(R01-2001-000-00354-0(2003))지원으로 수행되었음

† 학생회원 : 고려대학교 컴퓨터교육과
daelee@comedu.korea.ac.kr
sangchul@comedu.korea.ac.kr
†† 비 회 원 : 삼성 SDS 컨설턴트
k.chung@samsung.com
††† 비 회 원 : 고려대학교 컴퓨터교육과
zelkova@comedu.korea.ac.kr
†††† 종신회원 : 한국교원대학교 컴퓨터교육과
yjlee@knue.ac.kr
††††† 종신회원 : 고려대학교 컴퓨터교육과 교수
yuhc@comedu.korea.ac.kr
lee@comedu.korea.ac.kr

논문접수 : 2002년 9월 9일
심사완료 : 2003년 12월 17일

information, and facilitator agent is created with process, and the information agent constructs domain knowledge with its checkpoints and non-determistic events. And the garbage collection agent decides garbage collection time, and it deletes useless fault tolerance information in cooperation with the information and facilitator agent. For propriety of proposed garbage collection technique using agents, we compare domain knowledge of system that performs garbage collection after rollback recovery and domain knowledge of system that doesn't perform garbage collection.

Key words : multi-agent, garbage collection, fault tolerance information, rollback recovery

1. 서 론

분산 컴퓨팅 시스템의 발달로 기존의 단일 컴퓨팅 시스템에서의 수행과는 달리 하나의 작업을 네트웍으로 연결된 여러 프로세스에서 수행할 수 있다. 하지만 분산 시스템에서는 단일 시스템보다 높은 결함 발생 확률 때문에 많은 결함 포용 기법이 개발되고 연구되어왔다. 결함 포용 기법은 크게 검사점과 메시지 로깅 기법으로 나누어지며, 결함포용을 위해 검사점 정보와 메시지 로그를 안정된 저장소에 저장을 하여 결함 발생 후 회복 시 이러한 결함 포용 정보를 이용하여 복귀한다[1,2]. 하지만 안정된 저장소에서 결함 포용 정보의 관리는 매우 중요하다. 프로세스의 실행에 따라 결함 포용 정보의 양은 많아지고 이를 저장하는 안정 저장 장치의 용량은 무제한이 아니라면 저장 공간의 포화를 초래하여 전체 시스템의 성능을 저하시키며, 새로운 결함으로 처리될 수 있다. 따라서 안정 저장 장치에서의 결함 포용 정보의 쓰레기 처리(garbage collection)가 필요하다[2,3]. 쓰레기 처리란 프로세스의 복귀 회복 시 필요한 결함 포용 정보 중에서 불필요한 정보를 찾아내고 이러한 불필요한 결함 포용 정보의 삭제시기를 결정하는 것이다. 검사점 기법은 동기적 검사점(uncoordinated checkpointing) 기법과 비동기적 검사점(coordinated checkpointing) 기법으로 분류되며, 결함 포용을 위하여 프로세스의 상태 정보(state)를 검사점에 저장하며 쓰레기 처리의 대상은 검사점이고 검사점간의 일관된 회복선(consistent global checkpoint) 이전의 모든 검사점들을 쓰레기 처리한다[4,5]. 그리고 메시지 로깅 기법은 비관적(pessimistic), 낙관적(optimistic), 그리고 인과적(causal)메시지 로깅 기법으로 분류되고, 쓰레기 처리대상은 메시지 로그이며 순서 로그를 가지고 쓰레기 처리 대상을 선정하기에 메시지 내용 로그의 쓰레기 처리를 위한 부가적인 메시지 교환이 요구된다[6,7]. 기존의 연구 중에는 이러한 부가적인 메시지의 송수신을 하지 않고 쓰레기 처리를 하는 기법도 제안되었다[8].

기존의 연구에서 멀티 에이전트 개념을 도입하여 운영체제로부터 독립적인 복귀 회복 기법 시스템을 제안한 연구[9]에서도 쓰레기 처리의 필요성이 제시되었다. 본 논문에서는 기존의 연구에서 필요한 영역지식의 쓰레기 처리를 위하여 부가적인 메시지를 전송하지 않고 에이전트를 이용한 쓰레기 처리 기법을 제안한다. 쓰레기 처리 에이전트는 분산 시스템에서 운영체제의 시스템 영역에서 수행된 쓰레기 처리를 에이전트가 담당하게 하여 쓰레기 처리 기능을 어플리케이션 계층과 독립적인 별도의 계층으로 계층화하였다. 멀티 에이전트 개념의 도입으로 계층화와 모듈화가 이루어지면 분산 컴퓨팅 시스템을 비롯한 여러 컴퓨팅 시스템에 결함 포용 기능과 쓰레기 처리기능의 탑재가 쉬워지는 이식성의 증대와 확장성의 증대를 가져올 수 있다. 그리고 멀티 에이전트 기반 쓰레기 처리기법의 제안으로 기존의 연구중 결함 포용 에이전트 시스템의 구현 가능성을 증가시켰다. 본 논문에서는 비동기적 검사점(uncoordinated checkpointing) 기법과 송신자 기반 비관적 메시지 로깅(sender based pessimistic message logging) 기법을 기반으로 한다.

이 논문의 구성은 다음과 같다. 2장에서는 본 논문의 시스템 모델과 쓰레기 처리 기법에 대해 설명한다. 3장에서는 에이전트를 이용한 쓰레기 처리 알고리즘을 정의하고 제안한다. 4장에서는 CORBA 환경에서 Java를 이용한 에이전트를 구현하고 본 논문의 정당성을 증명하였다. 마지막으로 5장에서는 연구의 결론과 향후 연구 과제에 대해 기술한다.

2. 시스템 모델과 쓰레기 처리

2.1 시스템 모델

90년대부터 관심과 필요성이 대두된 에이전트는 특정 목적을 위해 사용자를 대신하여 작업을 수행하는 자율적인 프로세스이며 독자적으로 존재하지 않고 어떤 환경의 일부이거나 그 안에서 동작하게 된다. 에이전트는 작업 엔진, 영역 지식, 그리고 통신 모듈로 구성된다. 작업 엔진에서는 에이전트가 가지는 기능을 정의하고, 영역 지식에서는 작업 수행을 위한 규칙과 정보로 구성되어 에이전트의 역할을 정의하는 부분이며, 통신 모듈은 사용자 및 다른 에이전트와 상호작용을 할 수 있게 해준다[10-13].

본 논문에서 제안하는 시스템 모델은 응용프로그램을 수행하는 프로세스, 쓰레기처리를 담당하는 쓰레기 처리 에이전트, 결함 포용 정보의 관리를 담당하는 정보 에이전트, 에이전트간의 통신을 관리하는 조정 에이전트, 그리고 통신 채널등으로 구성되어 있는 분산 컴퓨팅 기반 멀티 에이전트 시스템 환경이다. 메시지 전달 시스템을 기반하며 다음 몇 가지를 가정한다. 분산 컴퓨팅 시스템의 통신환경은 통신 네트워크의 절단이 발생하지 않는다고 가정하고 프로세스간의 통신에 신뢰성 기반 통신을 보장하여 메시지 선입선출방식(FIFO)으로 메시지를 전송한다고 가정한다. 프로세스에서 결함 발생 시 결함 발생 모델은 고장-멈춤(fail-stop)모델에 따르며 결함 발생 시 휘발성 메모리의 저장정보만 잃어버리고 실행을 중지한다고 가정한다. 저장장치는 휘발성 메모리와 결함 발생 시에도 정보가 보장되는 안정 저장 장치로 구분한다[9,14].

기존의 연구인 멀티 에이전트 기반 결함포용 시스템[9]에서의 쓰레기 처리를 쓰레기 처리 에이전트를 도입함으로서 기존의 운영체제 기반에서의 쓰레기 처리작업을 독립시킨 멀티 에이전트 시스템 기반 쓰레기 처리 시스템을 제시한다. 그림 1은 본 논문에서 제안하는 쓰레기 처리 멀티 에이전트 시스템의 구성도이다.

기존의 연구에서는 쓰레기 처리 프로시져는 사용자 애플리케이션과 신뢰성 있는 통신 프로토콜 사이의 복귀회복기법에 위치하였지만 기존의 쓰레기 처리기법에 멀티 에이전트를 도입한 경우에는 회복기법과 프로세스간 통신은 그림 2와 같다.

본 논문에서는 에이전트들이 상호 협력을 통하여 정보를 주고받기 위한 이들 간의 통신 수단으로 현재 사용되는 여러 에이전트 통신 언어 중 표준으로 알려진 KSE(Knowledge Sharing Effort working group)가 제시한 KQML[15]을 사용하였다.

KSE에서는 KQML을 제외한 언어들 내에 전재하는 온톨로지(ontology)를 정의하였고, 이 언어를 표현하는 규칙으로 KIF(Knowledge Interchange format)를 제시하였다[11].

이 장에서는 본 논문에서 쓰레기 처리 에이전트, 정보 에이전트, 그리고 조정 에이전트간의 통신을 위해 쓰레기 처리 기법과 복귀 회복 기법에 관한 어휘들의 의미를 정의하는 온톨로지, 정보 에이전트의 영역지식 표현 언어인 KIF, 그리고 에이전트간의 통신 규약인 에이전트 외부 언어로 KQML을 정의하여 사용한다.

2.2 결함 포용 정보의 쓰레기 처리

기존의 결함 포용 기법의 결함 포용 정보의 쓰레기처

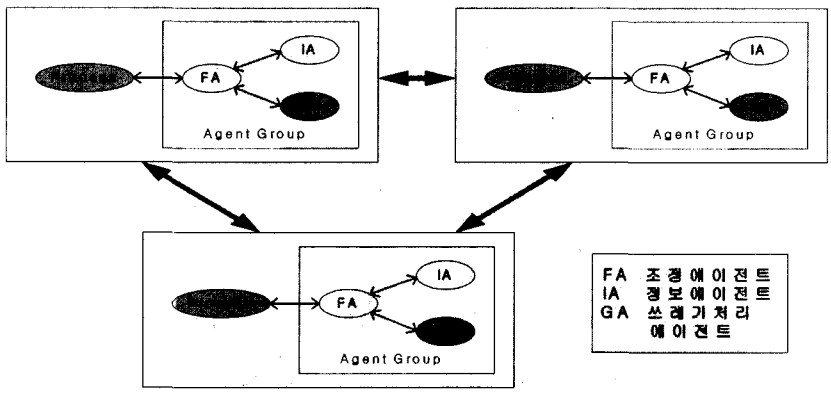


그림 1 쓰레기 처리 에이전트를 도입한 에이전트 시스템

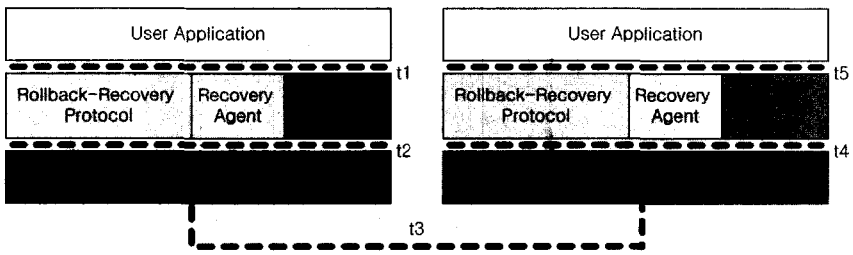


그림 2 멀티 에이전트를 도입한 경우 수정된 3계층

리는 결함 발생 후 복귀시 발생하는 손실메시지로 인한 불필요한 복귀가 발생되었다. 기존의 연구에서 검사점 간격을 사용하여 손실메시지를 발생시키지 않는 쓰레기 처리기법이 제안되었다[8,14]. 본 논문에서는 불필요한 복귀를 피하기 위해, 메시지에 마지막 검사점 번호를 첨부하여 전송함으로 쓰레기 처리 시점을 결정하는 방법을 사용한다.

[정의 1] 시스템 메시지에는 검사점 간격을 확인할 수 있는 정보를 포함한다.

$MSG\ p_k \stackrel{def}{=} \{content_m, m.ssn, p_{id}, last\_chpt\_num\}$ □

메시지 p_k는 다음 $content_m$, m.ssn, p_{id}, last_chpt_num로 구성되어 있다. $content_m$은 메시지의 내용이며 m.ssn은 송신 프로세스측에서 전송한 메시지 송신 번호이다. p_{id}는 송신 프로세스의 프로세스 아이디를 의미하며 last_chpt_num는 송신 프로세스에서 취한 가장 마지막 검사점 번호이다.

결함 회복시 필요한 결함 포용 정보는 검사점이나 로그에 저장되고 [정의 1]의 마지막 검사점 번호를 비교하여 검사점 간격을 확인할 수 있으며 손실 메시지를 위한 결함 포용 정보는 송신자 기반 로그를 사용하여 관리한다. 쓰레기 처리의 대상은 결함 포용 정보가 저장되어 있는 검사점과 로그이며 [조건 1]을 만족할 때 결함 포용 정보를 쓰레기 처리한다.

[조건 1] 결함 포용 정보의 쓰레기 처리 조건은 신뢰적인 통신 기법이 보장되고 프로세스 p_i에서 p_j로 메시지 m이 송신되었을 때 결함 포용 정보의 쓰레기 처리 조건은

i) $\exists\ chpt_j^{p-1},\ chpt_j^{p-1} \rightarrow receive_j(m)$이고,

ii) $\exists\ chpt_j^{p},\ receive_j(m) \rightarrow chpt_j^{p}$이다. □

[조건 1]의 조건 i)과 ii)를 만족할 때 결함 포용 정보의 쓰레기 처리는 가능하다. $chpt_j^{p-1}$은 메시지 m을 수신한 이전에 취한 검사점을 의미하고 $chpt_j^{p}$은 메시지 m을 수신한 이후에 취한 검사점을 의미한다. 고로 송신 프로세스의 검사점이나 로그에 저장되어 있는 결함 포용 정보의 쓰레기 처리는 수신 프로세스가 메시지 수신이후 $chpt_j^{p}$을 취했음을 아는 시점에서 이루어질 수 있다.

3. 에이전트를 이용한 쓰레기 처리 알고리즘

3.1 자료구조

그림 3은 본 논문에서 제안하는 쓰레기 처리 알고리즘에서 사용하는 자료구조이다.

3.2 쓰레기 처리 알고리즘

본 논문에서 제안하는 쓰레기 처리 알고리즘은 두 부분으로 구성되어 있다. 프로세스의 생성과 함께 에이전

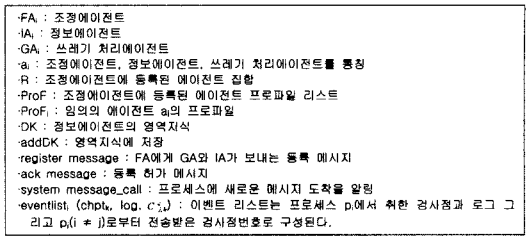


그림 3 자료구조

트가 생성되고 등록하는 초기화 부분과 프로세스가 시스템 메시지를 받았을 때 [조건 1]에 따라 쓰레기 처리 조건에 만족하는 결함 포용 정보를 쓰레기 처리하는 부분으로 구성된다. 쓰레기 처리 에이전트는 프로세스 pi의 생성과 함께 생성되고 생성과 동시에 조정 에이전트에 등록을 요청한다. 등록을 마친 후 쓰레기 처리 에이전트는 조정 에이전트로부터 새로운 메시지 도착 정보를 받고 호출된다. 조정 에이전트로부터 메시지와 함께 전송된 마지막 검사점 번호와 메시지 송신 번호를 받은 쓰레기 처리 에이전트는 정보 에이전트가 관리하는 영역지식에 저장되어 있는 로그정보를 요청하고 대기한다. 정보 에이전트로부터 요청한 로그정보를 받으면 MSG p_j에 실려있는 검사점 번호(C_p,k)와 메시지 송신 번호(m.ssn)를 정보 에이전트로부터 받은 로그정보의 프로세스 id가 같은 마지막 검사점 번호(C_p,k)와 비교한다. 이때 [조건 1]에 의하여 쓰레기 처리 에이전트는 MSG p_j에 실려 있는 검사점 번호(C_p,k)보다 작은 메시지 로그의 쓰레기 처리를 결정할 수 있다. 만약 쓰레기 처리 에이전트가 쓰레기 처리 대상을 결정하였으면 쓰레기 처리 에이전트는 정보 에이전트에 결정한 쓰레기 처리 대상 메시지 송신번호를 가진 로그의 삭제를 요청한다.

그림 4는 쓰레기 처리 에이전트의 동작 알고리즘이다.

```
Garbage Collection Agent
process p_i sends the message m_j to p_i
GA_i is created with process p_i;
Do
  GA_i sends register message to FA_i;
  GA_i receives ack message from FA_i;
  if GA_i receives system message_call from FA_i then
    GA_i requests eventlist to IA_i;
    GA_i waits eventlist from IA_i;
    if GA_i receives eventlist from IA_i then
     {
      for(eventlist_i(chpt_x, log, c^i_{j,x}) relating to p_i in p_i) {
          if { c^i_{j,x} > c^i_{j,x}) and (m.ssn^m > m.ssn^e) then
            declare the eventlist as a garbage
        }
     }
    GA_i requests garbage collection to IA_i;
  fi;
fi;
oD;
```

그림 4 쓰레기 처리 에이전트의 동작 알고리즘

3.3 정보 에이전트

정보 에이전트는 프로세스에서 발생한 결함 포용정보의 관리를 담당하는 에이전트이다. 정보에이전트는 프로세스에서 새로운 이벤트의 발생을 조정에이전트로부터 통보 받으면 영역지식에 프로세스가 송신한 메시지의 로그, 프로세스가 취한 검사점, 그리고 수신 메시지에서 추출한 마지막 검사점 번호 및 메시지 송신번호를 저장한다.

정보 에이전트에서의 쓰레기 처리동작은 쓰레기 처리 에이전트에서 쓰레기 처리 시점에 관련된 이벤트 리스트의 요청에 대해 응답해 줌으로 쓰레기 처리 대상 선정을 할 수 있게 해주며 선정된 쓰레기 처리 대상의 쓰레기 처리 요청시 불필요한 결함 포용 정보의 쓰레기 처리를 해주는 동작을 한다.

그림 5는 정보 에이전트의 동작 알고리즘이다.

```
Information Agent
IA_i is created with process p_i;
Do
  IA_i sends register message to FA_i;
  IA_i receives ack message from FA_i;
  if IA_i is requested addDK new_chpt from FA_i then
    IA_i translates addDK new_chpt to KIF;
    IA_i adds KIF to DK;
  fi;
  If IA_i is requested last_chpt_num from FA_i then
    IA_i finds last_chpt_num in DK;
    IA_i translates last_chpt_num to KQML;
    IA_i sends last_chpt_num to FA_i;
    IA_i is requested addDK message from FA_i;
    IA_i translates addDK message to KIF;
    IA_i adds KIF to DK;
  fi;
  if IA_i is requested addDK last_chpt_num and m.ssn from FA_i then
    IA_i translates last_chpt_num and m.ssn to KIF;
    IA_i adds KIF to DK;
  fi;
  if IA_i is requested eventlist from GA_i then
    IA_i finds eventlist in DK;
    IA_i translates eventlist to KQML;
    IA_i sends eventlist to GA_i;
  fi;
  if IA_i is requested garbage collection from GA_i then
    IA_i translates the eventlist to KIF;
    IA_i finds the eventlist in DK;
    IA_i deletes the eventlist;
  fi;
oD;
```

그림 5 정보 에이전트의 동작 알고리즘

3.4 조정 에이전트

조정 에이전트는 에이전트간의 통신을 담당하며 프로세스에서 발생한 이벤트들을 모니터링 하는 역할을 한다. 프로세스에서 이벤트의 발생을 감지하면 조정 에이전트는 새로운 이벤트의 발생을 각각의 쓰레기 처리, 정보 에이전트에 통보하며 정보 에이전트에 발생한 이벤트의 저장을 요청한다.

조정 에이전트의 쓰레기 처리 동작은 실제적으로 쓰레기 처리에 관련된 부분은 없지만 쓰레기 처리 에이전트와 정보 에이전트간의 통신을 도와준다.

그림 6은 조정 에이전트의 동작 알고리즘이다.

```
Facilitator Agent
FA_i Created with process p_i;
   R ← ∅;   ProF ← ∅;
   If FA_i receive register message from a_i then
      if (a_i ∉ R) then
         R ← R ∪ {a_i};  ProF ← ProF ∪ {ProF_i};
      fi;
      FA_i send ack message to a_i;
   fi;
Do
  if FA_i receive process p_i take checkpoint then
    FA_i translate process p_i take checkpoint to KQML;
    FA_i send addDK_request new_chpt to IA_i;
    FA_i send new_chpt_call to GA_i;
  fi;
  If FA_i send system message to process p_k;
    FA_i send last_chpt_num_require message to IA_i then :
    FA_i translate last_chpt_num to system language ;
    FA_i add last_chpt_num to system message;
    FA_i send system message to process p_k then
    FA_i translate system message to KQML;
    FA_i send addDK_request message to IA_i;
  fi;
  If FA_i receive system mesage from process p_k then
    FA_i translate system message to KQML;
    FA_i find last_chpt_num and m.ssn then
    FA_i send system message_call and m.ssn to GA_i;
    FA_i send last_chpt_num and m.ssn to GA_i;
    FA_i send addDK_last_chpt_num and m.ssn to IA_i;
  fi;
oD;
```

그림 6 조정 에이전트의 동작 알고리즘

4. 구현

4.1 구현 환경

본 논문에서는 에이전트간의 ACL 교환을 가능하게 해주기 위하여 OMG에서 개발된 CORBA를 사용하였고 Java로 에이전트의 컴포넌트를 구현하였다.

코바를 이용한 에이전트 간의 통신 환경은 그림 7과 같다.

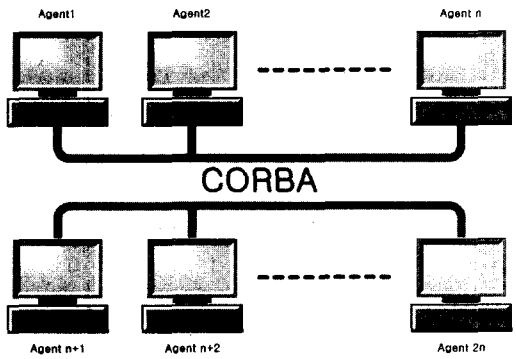


그림 7 코바를 이용한 에이전트간의 통신 환경

구현환경은 Microsoft windows 2000 운영체제 하에서 Java 컴파일러를 이용하였고 P-III 933MHz에서 실험하였고 각 프로세스의 구성은 표 1과 같다.

구현하는 멀티 에이전트 시스템은 3개의 노드로 구성된 분산 환경이며 각 노드에서는 하나의 프로세스가 수행되며 각각의 프로세스는 본 논문에서 제안하는 FA,

표 1 각 노드의 시스템 환경

구분	P0	P1	P2
IP	163.152.91.86	163.152.91.90	163.152.91.105
OS	Windows 2000	Windows 2000	Windows 2000
CPU	Intel P-III 933	Intel P-III 933	Intel P-III 700
RAM	192MB	192MB	256MB
HDD	20GB	20GB	30GB

GA, IA를 가지고 있다.

구현을 위한 가정은 다음과 같다.

- 프로세스간의 통신은 메시지 교환만으로 이루어진다.
- 에이전트간의 통신은 FA를 통한 메시지 교환만으로 이루어진다.
- 모든 프로세스는 동시에 시작한다.
- 에이전트의 결함은 발생하지 않는다.

실험에서 IA의 영역지식에 저장되는 결함 포용 정보 중 본 논문에서 사용하는 알고리즘에 따라 쓰레기 처리 시점에 대상 선정 후 그 대상을 쓰레기 처리하고 영역 지식의 결함 포용 정보의 양을 쓰레기 처리하지 않은 상태의 영역지식과 비교함으로 쓰레기 처리가 이루어짐을 알 수 있다. 실험 결과의 타당성 증명은 쓰레기 처리를 하는 시스템과 쓰레기 처리를 하지 않는 시스템에 같은 작업을 수행하게 하고, 수행도중 강제적인 결함을 발생(failure injection)시켜 복귀 회복 후 두 시스템의 영역 지식을 비교하여 같은 결과를 같는지 여부를 검사한다.

실험 결과의 분석에서 복귀 회복 기법은 기존의 연구[9]에서 제안된 분산 컴퓨팅 시스템에서 에이전트를 이용한 회복기법을 기반으로 하였다.

4.2 실험

본 논문에서 제안하는 에이전트를 이용한 쓰레기 처리기법을 구현하고 이에 대한 결과를 분석후 쓰레기 처리 기법의 올바른 수행을 증명하고자 한다.

그림 8은 쓰레기 처리 에이전트가 생성되어 자신의 정보를 조정 에이전트에 전송하여 등록하는 과정을 보

그림 8 쓰레기 처리 에이전트의 생성 및 등록과정

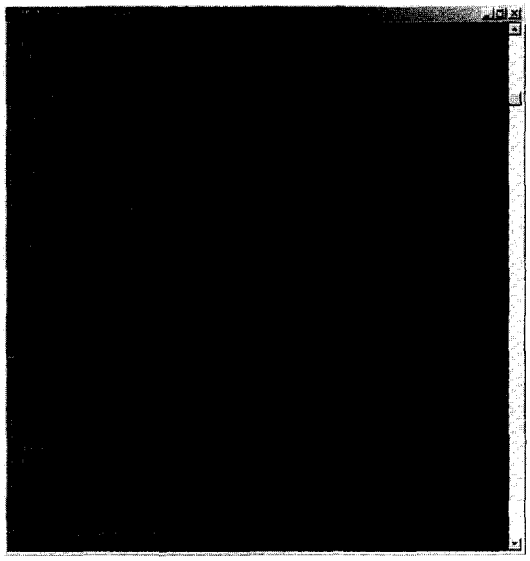


그림 9 쓰레기 처리 작업 수행 시 정보 에이전트의 동작

여준다. 정보 에이전트와 회복 에이전트도 동일한 과정을 수행한다.

그림 9는 정보 에이전트가 조정 에이전트를 통하여 쓰레기 처리 에이전트로 프로세스 p_i와 관련된 이벤트 리스트 전송과 쓰레기 처리요청 시 쓰레기 처리의 결과이다.

그림 10은 쓰레기 처리를 하지 않은 정상 수행 시 프로세스 p_j의 영역지식에 저장되어 있는 결함포용 정보를 보여주고 쓰레기 처리를 한 후 프로세스 p_j의 영역지식이다. 두 영역 지식을 비교하면 불필요한 결함 포용 정

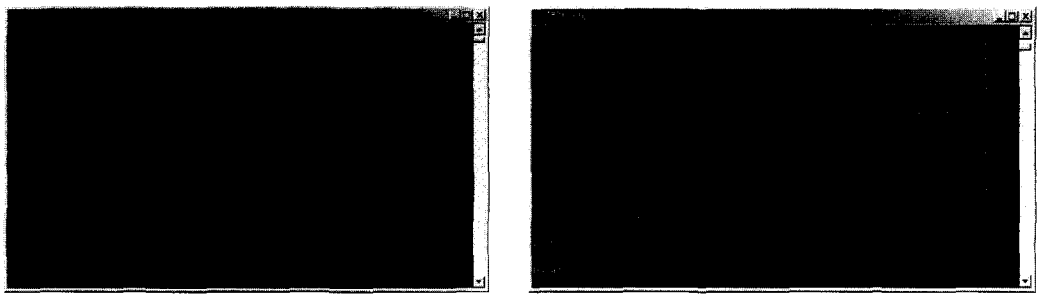


그림 10 쓰레기 처리하지 않은/쓰레기 처리한 영역지식간의 비교

보의 삭제가 이루어졌음을 알 수 있다.

불필요한 결함 포용 정보가 시스템에 어떠한 영향을 미치는 지를 알아보기 위한 부가적인 실험으로 쓰레기 처리를 수행하는 시스템과 쓰레기처리를 수행하지 않는 시스템에 각 시스템의 안정된 저장소의 용량을 600으로 가정하고 약 10000개의 메시지를 무작위로 선출하여 각 프로세스에서 발생하는 결함포용 정보의 누적으로 발생하는 안정된 저장소의 포화로 인한 시스템 결함이 발생하는가에 대하여 비교 실험을 하였다. 본 논문에서 결함 포용을 위한 정보인 checkpoint, log, log_rec는 각각 50, 5, 2의 크기로 가정하였다.

그림 11은 쓰레기 처리를 하지 않는 시스템에서 p_j의 진행에 따른 결함 발생을 보여주고 있습니다. p_j에서 발생한 event인 checkpoint, log, log-rec 등의 결함 포용 정보가 안정된 저장소에 누적에 의하여 안정된 저장소의 포화용량인 600를 넘기면 시스템 결함이 발생하여 시스템에서 불필요한 결함 포용 정보를 선택하여 삭제하여 다시 시스템을 시작하게 함을 보여주었습니다. 그림 11의 쓰레기처리를 하지 않는 시스템에서는 약 10000개의 이벤트 발생시 25번의 시스템 결함이 발생하였다.

그림 12에서는 쓰레기 처리 에이전트를 첨가한 시스템에서 p_j의 진행에 따른 결함 발생을 보여주고 있습니다. p_j에서 발생한 event인 checkpoint, log, log-rec등의 결함 포용 정보는 본 논문에서 명시한 쓰레기 처리 조건에 따라 불필요한 정보로 판별되면 즉시 삭제를 함으로 안정된 저장소의 포화로 인한 시스템 결함의 발생이 그림 11과 비교하였을 때 25 대 2의 비율을 보여주고 있다.

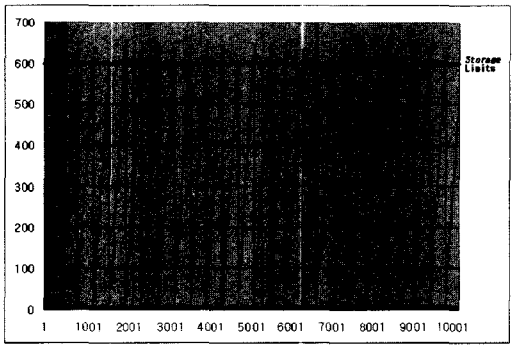


그림 11 쓰레기 처리를 수행 하지 않은 시스템의 결함 발생

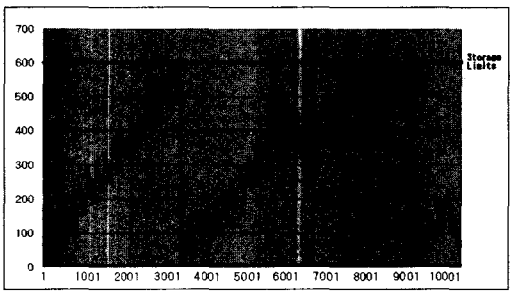


그림 12 쓰레기 처리를 수행한 시스템의 결함 발생

이 실험 결과를 통하여 본 논문에서 제안하는 에이전트를 이용한 쓰레기 처리 기법을 이용하여 결함 포용 정보의 누적으로 인해 발생하는 시스템 결함의 감소를 보여주었다.

4.3 비교 분석

에이전트를 이용한 쓰레기 처리가 올바른가를 증명하기 위하여 시스템에 강제적인 결함(failure injection)을 발생시켜 쓰레기 처리를 하지 않은 시스템과 비교해 보았다.

그림 13은 기존의 연구[9]에서 보여준 복귀 회복시스템을 본 논문에서 제안하는 시스템에 부가하여 프로세스 P_k에서 발생한 결함에 대해 복귀 회복을 하는 과정을 보여준다.

그림 14는 쓰레기 처리를 수행하지 않은 시스템에서 p_j의 영역지식이다.

그림 15는 쓰레기 처리를 한 시스템에서 프로세스 p_j의 영역지식이다. 결함으로부터 복귀 회복 후에 올바를 수행을 하는 지를 확인하기 위하여 eventindex 12번 die와 같이 failure injection을 하였다. 그림 14와 비교하면 쓰레기 처리 작업을 수행했음을 알 수 있고 결함 발생 후 복귀 회복 후에도 같은 순서의 결함 포용 정보를 저장하고 있음을 보여주며 이전의 쓰레기 처리로 인한 복귀 시 문제도 발생하지 않았다.

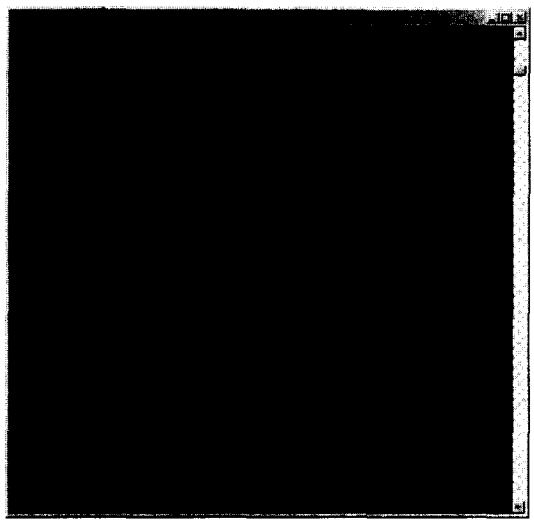


그림 13 회복 에이전트의 복귀회복 과정

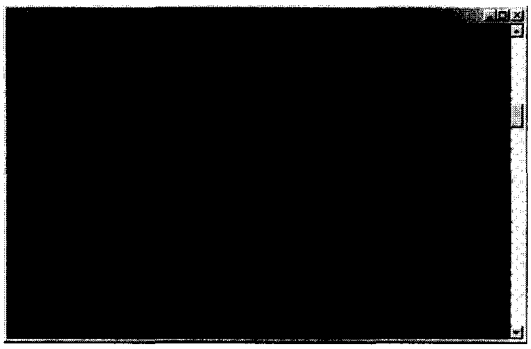


그림 14 쓰레기 처리를 수행하지 않은 시스템에서 p_j의 영역지식

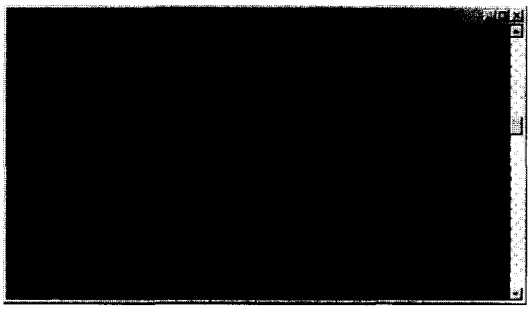


그림 15 쓰레기 처리를 수행한 시스템에서 p_j의 영역지식

본 논문에서 제안하는 에이전트를 이용한 쓰레기 처리 구현 결과 프로세스에서 발생한 모든 이벤트들을 조정 에이전트가 모니터링하여 정보 에이전트에 전송하여 저장하게 함으로 각 프로세스에서 발생한 모든 이벤트 정보가 영역지식에 정확하게 저장되었다. 그리고 쓰레기 처리 에이전트는 조정 에이전트와 정보 에이전트와의 협력을 통하여 영역지식에 저장되어 있는 결함 포용 정보를 쓰레기 처리 시점에 정확히 처리하였음을 알 수 있고 프로세스에 결함이 발생하여 복귀 회복 작업을 수행하였을 때도 쓰레기 처리를 하지 않은 멀티 에이전트 시스템과 같은 영역지식을 보장함으로 기존의 연구[9]를 보완하였다.

5. 결 론

분산 컴퓨팅 시스템에서 결함 발생에 대한 많은 복귀 회복기법들이 연구되었고 이러한 연구에서 새로운 개념인 독립적인 에이전트를 이용하여 운영체제에 독립적인 에이전트를 이용한 회복기법이 연구되었다. 기존의 결함 회복 기법에서의 쓰레기 처리는 운영체제의 관리하에 모든 쓰레기 처리 작업을 부가적인 메시지 전송으로 수행하여 전체 분산 컴퓨팅 시스템의 성능을 저하시키는 원인이 되었다. 본 논문에서는 운영체제에 독립적인 멀티 에이전트 기반 쓰레기 처리 기법을 제안하였다. 쓰레기 처리 에이전트는 프로세스의 생성과 함께 생성되어 송수신 메시지를 통하여 결함 포용 정보를 관리하는 정보 에이전트에서 쓰레기 처리조건이 맞는 시점에 불필요한 결함 포용 정보를 삭제하여 운영체제에서 독립적인 쓰레기 처리 기법을 제안하였다. 결함 발생시 손실 메시지 발생을 고려하여 불필요한 회피를 막으며 송수신 메시지를 통하여 쓰레기 처리 시점을 정하기에 기존의 쓰레기 처리 기법에서의 쓰레기 처리를 위한 부가적인 메시지를 전송하지 않음으로 시스템의 성능 향상을 가져온다. 그리고 코바 환경에서 자바와 에이전트 통신 언어로 개발된 에이전트들이 상호작용하며 쓰레기 처리 과정을 수행하는 실험을 하였다. 본 논문에서 제안한 에이전트를 이용한 쓰레기 처리는 운영체제에 독립적이며 프로세스의 성능을 향상시킨다. 또한 에이전트를 이용한 쓰레기 처리는 분산 컴퓨팅 시스템에서 쓰레기 처리 기능을 어플리케이션 계층과 독립적인 별도의 계층으로 계층화하였고 이는 분산 컴퓨팅 시스템의 개방성 증대에 기여하며 쓰레기 처리 메카니즘의 이식성의 증대와 확장성의 증대를 기대할 수 있고 기존의 제안된 멀티 에이전트를 이용한 결함 포용 기법의 완성도를 높였다.

향후 연구 과제로는 복귀회복을 위해 본 논문에서는 독립 검사점 기법과 송신자 기반 비관적 메시지 로깅 기법을 사용하였다. 그러나 이와 다른 기법인 조정 검사점 기법, 낙관적 메시지 로깅 기법, 인과적 메시지 로깅 기법을 적용하여 수행되는 시스템에서 멀티 에이전트를 이용한 쓰레기 처리를 연구할 예정이다.

참 고 문 헌

[1] L. Alvisi, "Understanding the message logging paradigm for masking process crashes," *Ph.D. Thesis, Department of Computer Science, Cornell University*, Jan. 1996.

[2] E. N. Elnozahy, D. B. Johnson and Y. M. Wang, "A Survey of Rollback-Recovery Protocols in Message Passing Systems," *CMU Technical Report CMU-CS-99-148*, June 1999.

[3] M. V. Srcenivas, Subhash Bhalla, "Garbage Collection in Message Passing Distributed Systems," *IEEE Computer Society Press*, pp.213-218, March 1995.

[4] Sylvain R. Y. Louboutin, Vinny Cahill, "On Thorough Garbage Collection in distributed Systems," *Third IEEE Symposium on Computer & Communications, pp. 576-581*, 1998.

[5] M. V. Sreenivas, Subhash Bhalla, "Garbage Collection in Message Passing Distributed Systems," *Proceeding of International Symposium on Parallel Algorithms/Architecture Synthesis, IEEE Computer Society Press, pp.213-218*, March 1995.

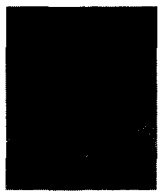
[6] E. L. Elnozahy, W. Zwanepoel. "Manetho: Transparent rollback-recovery with low overhead, limited rollback and fast output commit," *IEEE Transactions on Computers*, 41(5):526-531, May 1992.
[7] D. B. Johnson, W. Zwaenepoel, "Sender-based Message Logging," *In Digest of papers:17 Annual International Symposium on Fault-Tolerant Computing, pp.14-19, IEEE Computer Society*, June 1987.
[8] 정광식, 유헌창, 백맹순, 손진곤, 황종선, "인과적 메시지 로깅 기법에서 부가적 메시지 교환없는 메시지 로그 쓰레기 처리 기법", *정보과학회 논문지: 시스템 및 이론, 제28권 제7,8호*, pp.331-340, 2001.
[9] 이화민, 정광식, 신상철, 이대원, 이원규, 유헌창, "분산 컴퓨팅 시스템에서 에이전트를 이용한 회복 기법", *정보과학회 가을학술발표논문집, 제28권 2호*, pp.556-558, 2001.
[10] Franklin S. and Graesser A., "Is it an agent, or just a program? : A taxonomy for autonomous agents," *Proc. of Third International Workshop on Agent Theories, Architectures and Languages*, 1996.
[11] Genesereth M. abd Fikes R., "Knowledge interchange format version 3.0 reference manual," *Technical Report Logic-92-1, Computer Science Department, Stanford University*, 1992.
[12] Goodwin R., "Formalizing properties of agents," Technical Report CMU-CS-93-159, *School of Computer Science, Carnegie Mellon University*, 1993.
[13] Wooldridge M. and Jennings N., Intelligent Agents, *Lecture Notes in Artificial Intelligence #890, Springer-Verlag*, 1995.
[14] Jiannong Cao, G.H. Chan, Weija. Jia, Tharam S. Dillon, "Checkpointing and Rollback of Wide-Area Distributed Applications Using Mobile Agents", *Parallel and Distributed Processing Symposium*, 2001.
[15] Finin T., Fritzson R., Mckay D. and McEntire R., "KQML as an agent communication language," *Proc. of CIKM '94*, pp.126-130, 1994.

이 대 원

2001년 순천향대학교 전자공학과 졸업(공학사). 2003년 고려대학교 대학원 컴퓨터교육전공 졸업(교육학석사). 2003년~현재 고려대학교 대학원 컴퓨터교육전공 박사과정. 관심분야는 결함포용시스템, 이동컴퓨팅시스템, 그리드컴퓨팅시스템, 분산시스템, 멀티에이전트시스템 등

정 광 식

1993년 고려대학교 컴퓨터학과 학사
1995년 고려대학교 컴퓨터학과 석사
2000년 고려대학교 컴퓨터학과 박사
2002년 삼성 SDS 책임 컨설턴트. 관심분야는 분산시스템, 멀티에이전트시스템, 이동컴퓨팅시스템, 그리드컴퓨팅 등



이 화 민

2000년 고려대학교 사범대학 컴퓨터교육과(이학사). 2002년 고려대학교 대학원 컴퓨터교육전공(교육학석사). 2002년~현재 고려대학교 대학원 컴퓨터교육전공 박사과정. 관심분야는 결함포용시스템, 그리드컴퓨팅시스템, 분산시스템, 멀티에이전트시스템 등

신 상 철

2001년 고려대학교 컴퓨터교육과(이학사) 2001년~현재 고려대학교 대학원 컴퓨터교육전공 석사과정. 관심분야는 그리드컴퓨팅시스템, 분산시스템, 멀티에이전트시스템 등

이 영 준

1988년 고려대학교 전산과학과(이학사) 1994년 미국 미네소타대학교 전산학(Ph.D.). 2003년~현재 한국교원대학교 컴퓨터교육과 교수. 관심분야는 정보통신, 지능형 분산 시스템, 컴퓨터교육 등

유 헌 창

1989년 고려대학교 이과대학 전산과학과 졸업(이학사). 1991년 고려대학교 대학원 석사과정 전산과학과 졸업(이학석사) 1994년 고려대학교 대학원 박사과정 전산과학과 졸업(이학박사). 1995년~1998년 서경대학교 이공대학 컴퓨터공학과 조교수. 2004년~2005년 미국 Georgia Tech. 컴퓨터학과 방문교수. 1998년~현재 고려대학교 사범대학 컴퓨터교육과 부교수. 2002년~현재 한국컴퓨터교육학회 부회장. 관심분야는 결함포용시스템, 분산시스템, 그리드컴퓨팅시스템 등

이 원 규

1985년 고려대학교 문과대학 영어영문학과 졸업(문학사). 1989년 츠쿠바대학 대학원 이공학연구과 졸업(공학석사). 1993년 츠쿠바대학 대학원 공학연구과(전자정보공학 전공). 졸업(공학박사). 1996년~현재 고려대학교 사범대학 컴퓨터교육과 부교수. 2002년~현재 한국컴퓨터교육학회 회장. 관심분야는 컴퓨터과학교육, 데이타베이스, 정보검색, 분산시스템 등