멀티 에이전트 시스템 기반 쓰레기 처리 기법

이대원*O 정광식** 이화민* 이원규* 유헌창*
* 고려대학교 컴퓨터교육과 ** 런던대학교 컴퓨터학과
{daelee, zelkova, lee, yuhc}@comedu.korea.ac.kr, k.chung@ucl.ac.uk

A Garbage Collection based on Multi-Agent Systems

Dae-Won Lee*O Kwang-Sik Chung** Hwa-Min Lee* Won-Gyu Lee* Heon-Chang Yu*
* Dept. of Computer Science Education, Korea University ** Dept. of Computer Science, University College London

요 약

본 논문은 기존의 멀티 에이전트 기반의 결함 포용 기법에서 결함 포용 정보를 쓰레기 처리하는 쓰레기 처리 에이전트를 제안한다. 쓰레기 처리 에이전트는 정보 에이전트의 영역지식에서 불필요한 결함 포용 정보의 제거 시점을 찾아내고, 이를 정보 에이전트에 알린다. 쓰레기 처리기법에 멀티 에이전트 개념의 도입은 부가적인 메시지 전송을 하지 않고 쓰레기 처리 에이전트를 이용하여 운영체제에서 독립적인 쓰레기 처리를 가능하게 한다. 결함 포용 정보의 쓰레기 처리는 프로세스의 검사점 간격을 이용하여 쓰레기 처리 시점을 결정하기에 결함 발생 후 재수행시 손실메시지 발생으로 인한 불필요한 복귀를 막을 수 있고 에이전트를 사용함으로 쓰레기 처리 기법의 이식성과 확장성의 증대를 기대할 수 있다.

1. 서 론

분산 컴퓨팅 시스템에서의 결함 포용 기법은 프로세스의 결함 발생 후 회복시 안정 저장 장치에 있는 결함 포용 정보를 이용하여 프로세스를 재수행하는 기법이다[1]. 프로세스가 작업을 수행할수록 이러한 결함 포용 정보의 양이 늘어나면 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 유지 및 삭제가 필요하다[1,2].

쓰레기 처리의 필요성은 기존의 연구중에서 멀티 에이전트 개념을 도입하여 운영체제로부터 독립적인 복귀 회복 기법 시스템을 제안한 연구[3]에서도 제안한 바 있다. 이에 본 논문에서는 멀티 에이전트를 이용한 결함 포용 기법에 쓰레기 처리에이전트를 도입하여 부가적인 메시지를 전송하지 않고 독립적인 쓰레기 처리가 가능한 기법을 제안한다. 쓰레기 처리에이전트는 프로세스의 생성과 함께 생성되며 프로세스간 통신 메시지에 검사점 간격을 첨부하여 전송함으로써 프로세스에서 발생한 검사점과 로깅정보 등을 저장하는 정보에이전트에게 쓰레기 처리조건에 맞는 시점을 알려준다. 쓰레기 처리에이전트로부터 쓰레기 처리 시점을 받은 정보에이전트는 불필요한 결함 포용 정보를 삭제함으로써 운영체제로부터 쓰레기 처리를 독립시켰다. 또한 검사점 간격을 이용한 쓰레기 처리방법은 손실메시지의 발생으로 인한 불필요한 복귀를 방지할 수 있다.

본 논문에서는 에이전트를 이용한 쓰레기 처리기법을 제안하기 위해 프로세스의 쓰레기 처리를 위한 쓰레기 처리에이전트를 정의하고 설계하며 결함 포용 정보의 관리를 위한 정보에이전트, 에이전트간의 통신을 담당하는 조정에이전트를 기존의 연구에서 발전시켰다. 멀티 에이전트 기반 쓰레기 처리기법의 제안으로 결함포용 에이전트 시스템의 구현 가능성을 증가시키며 에이전트를 사용함으로 이식성과 확장성의 증대를 기대할 수 있다. 본 논문에서는 비동기적 검사점기법과 송신자 기반 비관적 메시지 로깅기법을 기반으로 한다.

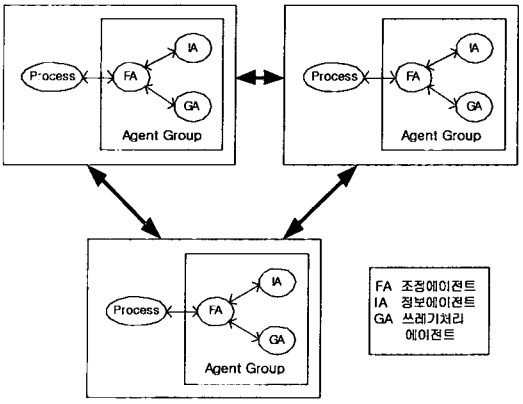
2. 멀티 에이전트 기반 쓰레기 처리 시스템의 구조

2.1 기존의 결함 포용 정보의 쓰레기 처리

쓰레기 처리란 프로세스의 복귀 회복 시 필요한 결함 포용 정보 중에서 불필요한 정보를 찾아내고 이러한 불필요한 결함 포용 정보를 삭제하는 것이다. 기존의 분산 컴퓨팅 시스템에서의 쓰레기 처리 방법은 결함 포용 기법에서의 검사점 기법과 로깅 기법을 기반으로 하고 있다. 검사점 기법의 경우 검사점간의 일관된 회복선(consistent recovery line)을 찾고 일관된 회복선을 이루는 검사점 이전의 모든 검사점들을 쓰레기 처리한다. 로깅기법에서는 프로세스에서 발생한 비결정적 사건들의 순서 로그를 가지고 회복한다. 순서 로그를 가지고 쓰레기 처리 대상을 선정하기에 메시지 내용 로그의 쓰레기 처리를 위한 부가적인 메시지 교환이 필요하였다[1][2]. 기존의 연구에서 이러한 부가적인 메시지의 송수신을 하지 않고 쓰레기 처리를 하는 기법이 제안되었다[4].

2.2 시스템 모델

본 논문에서 제안하는 시스템 모델은 응용프로그램을 수행하는 프로세스와 쓰레기 처리를 담당하는 쓰레기 처리에이전트, 결함 포용 정보의 관리를 담당하는 정보에이전트, 에이전트간의 통신을 관리하는 조정에이전트 그리고 통신 채널로 구성되어 있는 분산 컴퓨팅 기반 멀티 에이전트 시스템 환경이다. 분산 컴퓨팅 시스템은 메시지 전달 시스템을 기반으로 하기에 통신환경은 네트워크의 분할이 발생하지 않는다고 가정하고 프로세스간 통신의 신뢰성기반 통신을 보장하여 메시지 선입선출방식(FIFO)으로 메시지를 전송한



<그림 1> 쓰레기 처리 멀티 에이전트 시스템

본 연구는 한국과학재단 목적기초연구(R01-2001-00354) 지원으로 수행되었음.

다고 가정한다. 프로세스에서 결함 발생시 결함 발생 모델은 고장-멈춤(fail-stop) 모델에 따르며 결함 발생시 휘발성 메모리의 저장 정보만 잃어버리고 실행을 중지한다고 가정한다. 저장장치는 휘발성 메모리와 결함 발생시에도 정보가 보장되는 안정 저장 장치로 구분한다[2,4].

<그림 1>는 본 논문에서 제안하는 쓰레기 처리를 위한 멀티 에이전트 시스템의 구성도이다.

2.3 결함 포용 정보의 쓰레기 처리

기존의 결함포용 정보의 쓰레기처리는 결함 발생 후 복귀시 발생하는 손실메시지로 인한 불필요한 복귀가 발생되었다. 본 논문에서는 불필요한 복귀를 피하기 위해, 메시지 m에 마지막 검사점 번호(last_ckpt_num)와 메시지 송신 번호(m.ssn)를 첨부하여 전송함으로 쓰레기 처리의 시점을 결정하는 방법을 사용한다.

> [정의 1] 시스템 메시지에는 검사점 간격을 확인할 수 있는 정보를 포함한다. 다음과 같이 정의한다.
>
> $m \stackrel{def}{=} \{content_m, m.ssn, p_{id}, last\_ckpt\_num\}$

결함 회복시 필요한 결함 포용 정보는 검사점이나 로그에 저장되고 [정의 1]의 마지막 검사점 번호를 비교하여 검사점 간격을 확인 할 수 있으며 손실 메시지를 위한 결함 포용 정보는 송신자 기반 로그를 사용하여 관리한다.

> [조건 1] 결함 포용 정보의 쓰레기 처리 조건은 신뢰적인 통신 기법이 보장되고 프로세스 p_i에서 p_j로 메시지 m이 송신되었을 때
>
> i) $\exists\ C_{p_i,x}$, $C_{p_i,x} \rightarrow receive_j(m)$이고,
>
> ii) $\exists\ C_{p_j,y}$, $receive_j(m) \rightarrow C_{p_j,y}$ 이다.

[조건 1]의 i)과 ii)를 만족할 때 결함 포용 정보의 쓰레기 처리는 가능하다. $C_{p_j,y}$은 m수신 이후에 취한 검사점을 의미한다. 송신 프로세스의 검사점이나 로그에 저장되어 있는 결함 포용 정보의 쓰레기 처리는 수신 프로세스로부터 메시지 수신이후에 $C_{p_j,y}$을 취했음을 아는 시점에서 이루어질 수 있다.

3. 에이전트의 통신 언어

3.1 쓰레기 처리 기법을 위한 온톨로지

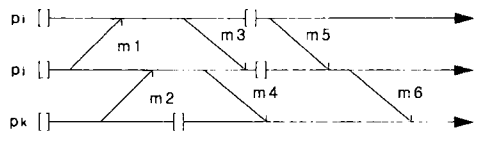
분산 컴퓨팅 시스템에서 쓰레기 처리를 위해 사용하는 어휘들을 Garbage_Collection으로 정의한다. <표 1>은 정의한 Garbage_Collection의 온톨로지이다.

<표 1> Garbage_Collection 온톨로지

어휘	의미
checkpoint	프로세스가 검사점을 취한다.
send	다른 프로세스로 메시지를 전송하다.
receive	다른 프로세스로부터 메시지를 전송받는다.
before	앞의 이벤트는 뒤의 이벤트보다 먼저 발생한다.
log	송신 메시지의 로그를 취한다
log_rec	수신한 메시지에서 마지막 검사점 정보를 추출하여 저장한다.
garbage	불필요한 결함 포용 정보를 삭제하다.
last_ckpt_num	마지막 검사점 번호($C_{i,k}$)를 의미한다.
ckpt_interval	$C_{i,k-1}$과 $C_{i,k}$사이의 간격을 의미한다.
m.ssn	메시지송신 번호를 의미한다.

3.2 KIF를 이용한 에이전트의 영역 지식 구축

쓰레기 처리를 위해 에이전트의 영역지식을 위해 정의한 KIF의 BNF문법을 사용하여 다음의 <그림 2>에서의 p_i의 영역지식의 표현은 <그림 3>와 같다.



<그림 2> 프로세스간의 통신 예

```
DK
(checkpoint p_i c_i,0)(log pi 1)(before (checkpoint p_i c_i,0)(log pi 1))(log_rec p_k c_k,0)(before (log pi (send p_i m1))(log_rec p_k c_k,0))(m.ssn p_k 2)(before (log_rec p_k c_k,0)(m.ssn p_k 2))(log_rec p_i c_i,0)(before (m.ssn p_k m2)(log_rec p_i c_i,0))(m.ssn p_i 3)(before (log_rec p_i c_i,0)(m.ssn p_i 3))(log p_k 4)(before (m.ssn p_i m3)(log p_k 4))(checkpoint p_i c_i,1)(before (log p_k 4)(checkpoint p_i c_i,1))(log_rec p_i c_i,1)(before (checkpoint p_i c_i,1)(log_rec p_i c_i,1))(m.ssn p_i 5)(before (log_rec p_i c_i,1)(m.ssn p_i 5))
(ckpt_interval c_i,0 c_i,1) ---> can start garbage collection
(before (m.ssn p_i 5)(ckpt_interval c_i,0 c_i,1))(send p_k 6)(before (ckpt_interval c_i,0 c_i,1)(send p_k 6))
```

<그림 3> KIF로 구축된 p_i의 영역지식

3.3 KQML을 이용한 에이전트 간의 통신언어

에이전트간의 통신언어인 KQML과 KIF를 사용하여 <그림 4>와 <그림 5>에서 에이전트 간의 간단한 통신과 쓰레기처리의 예를 살펴보겠다.

```
(ask-one   :  sender GA_j
              receiver FA_i
              reply-with id1
              language KIF
              ontology Garbage_Collection
              content (last_ckpt_num p_i ?c))
(tell      :  content (last_ckpt_num p_i c_i,2))
```

<그림 4> 에이전트간의 간단한 통신 예

```
(tell      :  sender GA_j
              receiver FA_i
              reply-with id2
              language KIF
              ontology Garbage_Collection
              content (garbage p_i ((checkpoint p_i c_i,0)(log pi 1)(log_rec p_i c_i,0)(m.ssn p_i 3)))
```

<그림 5> 쓰레기 처리의 예

4. 에이전트를 이용한 쓰레기 처리 알고리즘

4.1 자료구조

<그림 6>는 본 논문에서 제안하는 쓰레기 처리 알고리즘에서 사용하는 프로세스 p_i에 대한 자료구조는 다음과 같다.

- FA_i : 조정에이전트
- IA_i : 정보에이전트
- GA_i : 쓰레기 처리에이전트
- a_i : 조정에이전트, 정보에이전트, 쓰레기 처리에이전트를 통칭
- R : 조정에이전트에 등록된 에이전트 집합
- $ProF_i$: 조정에이전트에 등록된 에이전트 프로파일 리스트
- $ProF_i$: 임의의 에이전트 a_i의 프로파일
- DK : 정보에이전트의 영역지식
- addDK : 영역지식에 저장
- register message : FA에게 GA와 IA가 보내는 등록 메시지
- ack message : 등록 허가 메시지
- system message_call : 프로세스에 새로운 메시지 도착을 알림
- eventlist$_i$ ($ckpt_k$, log, $c_{j,x}$) : 이벤트 리스트는 프로세스 p_i에서 취한 검사점과 로그 그리고 $p_i(i \neq j)$로부터 전송받은 검사점번호로 구성된다.

<그림 6> 자료구조

4.2 쓰레기 처리 알고리즘

본 논문에서 제안하는 쓰레기 처리 알고리즘은 프로세스의 생성과 함께 에이전트가 생성되고 등록을 하는 초기화 부분과 프로세스가 시스템 메시지를 받았을 때 [조건 1]에 따라 쓰레기 처리 조건에 만족하는 결함 포용 정보를 쓰레기 처리하는 부분으로 구성된다.

알고리즘에서 $c^e_{i,k}$와 $c^m_{i,k}$는 이벤트리스트(e)와 메시지(m)을 구성하는 프로세스 p_i의 k번째 검사점 번호이다. 쓰레기 처리에이전트는 이벤트리스트에서 [조건 1]에 부합하는 검사점, 로그, 그리고 검사점 번호등을 쓰레기 처리 대상으로 선정하고 쓰레기 처리를 요청한다.

쓰레기 처리 알고리즘은 <그림 7>과 같다.

```
process pi send the mj to pk
GAi is created with process pi;
Do
  GAi sends register message to FAi;
  GAi receives ack message from FAi;
  if GAi receives system message_call from FAi then
    GAi requests eventlist to IAi;
    GAi waits eventlist from IAi;
    if GAi receives eventlist from IAi then
     {
      for(eventlisti(ckptx, log, c^e_{i,x}) relating to pi in pi) {
          if ( c^m_{i,y} > c^e_{i,x}) and (m.ssn^m > m.ssn^e) then
           declare the eventlist as a garbage
       }
     }
    GAi requests garbage collection to IAi;
  fi;
 fi;
oD;
```

<그림 7> 쓰레기 처리 에이전트의 동작 알고리즘

4.3 정보 에이전트와 조정 에이전트

다음 <그림 8>와 <그림 9>는 결함 포용 정보의 관리를 담당하는 정보 에이전트와 에이전트 간의 통신을 담당하는 조정 에이전트의 동작이다.

```
IAi is created with process pi;
Do
  IAi sends register message to FAi;
  IAi receives ack message from FAi;
  if IAi is requested addDK new_ckpt from FAi then
    IAi translates addDK new_ckpt to KIF;
    IAi adds KIF to DK;
  fi;
  if IAi is requested last_ckpt_num from FAi then
    IAi finds last_ckpt_num in DK;
    IAi translates last_ckpt_num to KQML;
    IAi sends last_ckpt_num to FAi;
    IAi is requested addDK message from FAi;
    IAi translates addDK message to KIF;
    IAi adds KIF to DK;
  fi;
  if IAi is requested addDK last_ckpt_num and m.ssn from
    FAi then
    IAi translates last_ckpt_num and m.ssn to KIF;
    IAi adds KIF to DK;
  fi;
  if IAi is requested eventlist from GAi then
    IAi finds eventlist in DK;
    IAi translates eventlist to KQML;
    IAi sends eventlist to GAi;
  fi;
  if IAi is requested garbage collection from GAi then
    IAi translates the eventlist to KIF;
    IAi finds the eventlist in DK;
    IAi deletes the eventlist;
  fi;
oD;
```

<그림 8> 정보 에이전트의 동작 알고리즘

다음 <그림 8>와 <그림 9>는 결함 포용 정보의 관리를 담당하는 정보 에이전트와 에이전트 간의 통신을 담당하는 조정 에이전트의 동작이다. 정보 에이전트의 영역지식에서 프로세스가 송신한 메시지의 로그와 프로세스가 취한 검사점 그리고 수신 메시지에서 추출한 마지막 검사점 번호와 메시지 송신번호를 저장한다. 프로세스와 에이전트, 에이전트와 에이전트 간의 통신은 조정에이전트를 통하여 이루어진다.

```
FAi
  FAi is created with process pi;
    R ← ∅;   ProF ← ∅;
    if FAi receives register message from ai then
          if (ai ∉ R) then
             R ← R ∪ {ai};  ProF ← ProF ∪ {ProFi};
          fi;
          FAi sends ack message to ai;
    fi;
  Do
    if FAi receives process pi take checkpoint then
      FAi translates process pi take checkpoint to KQML;
      FAi requests addDK new_ckpt to IAi;
    fi;
    if FAi sends system message to process pj then
      FAi requests last_ckpt_num to IAi;
      FAi receives last_ckpt_num to IAi;
      FAi translates last_ckpt_num to system language;
      FAi adds last_ckpt_num to system message;
      FAi sends system message to process pj;
      FAi translates system message to KQML;
      FAi requests addDK message to IAi;
    fi;
    if FAi receives system mesage from process pj then
      FAi translates system message to KQML;
      FAi finds last_ckpt_num and m.ssn;
      FAi sends system message_call to GAi;
    fi;
  oD;
```

<그림 9> 조정 에이전트의 동작 알고리즘

5. 결론 및 향후 연구 과제

본 논문에서는 멀티 에이전트 기반 결함 포용 기법[3]에서 쓰레기 처리 에이전트를 도입하여 운영체제에 독립적인 멀티 에이전트 기반 쓰레기처리 기법을 제안하였다. 쓰레기 처리 에이전트는 프로세스의 생성과 함께 생성되고, 송수신 메시지를 통하여 결함 포용 정보를 관리하는 정보 에이전트에게 쓰레기 처리조건이 맞는 시점에 불필요한 결함 포용 정보의 삭제를 요청한다. 또한 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. 본 논문에서 에이전트 기반 쓰레기 처리 기법을 제안함으로써 기존 [3]의 구현 가능성 증가와 쓰레기 처리 기법의 이식성과 확장성을 증가시킬 수 있다.

향후 과제로는 에이전트 기반의 통신환경인 코바 기반으로 구현하는 것이고 멀티 에이전트 환경에서 낙관적 메시지 로깅 기법에서의 쓰레기 처리 기법을 연구할 예정이다.

6 참고문헌

[1] Mootaz Elnozahy, Lorenzo Alvisi, Yi-Min Wang, David B. Johnson, "A Survey of Rollback-Recovery Protocols in Message-Passing Systems" *Technical Report CMU-CS-96-181*, 1996.

[2] M. V. Sreenivas, Subhash Bhalla, "Garbage Collection in Message Passing Distributed Systems", *IEEE*, 1995.

[3] 이화민, 정광식, 윤태명, 이원규, 유헌창, "분산 컴퓨팅 시스템에서 에이전트를 이용한 회복 기법", *정보과학회*, 2001.

[4] 정광식, 유헌창, 백맹순, 손진곤, 황종선, "인과적 메시지 로깅 기법에서 부가적 메시지 교환없는 메시지 로그 쓰레기 처리 기법", 정보과학회 논문지: 시스템 및 이론, 제 28권 제7·8호, pp331-340, 2001.

[5] Jiannong Cao, G.H. Chan, Weija. Jia, Tharam S. Dillon, "Checkpointing and Rollback of Wide-Area Distributed Applications Using Mobile Agents", *IEEE*, 2001.