고장 감내형 CORBA를 위한 객체 그룹간 고장인지 및 회복 메커니즘의 설계

°박 종 필, 김 유 성
인하대학교 전자계산공학과

Design of Fault Detection and Recovery Mechanism for Fault Tolerant CORBA

°Jong-Phill Park, Yoo-Sung Kim
Dept. of Computer Science & Engineering, Inha University

요 약

고장 감내형 CORBA(Fault Tolerant Common Object Request Broker Architecture)는 분산객체 시스템 통합기술의 표준인 CORBA에 고장에 대한 회복수단을 제공하기 위해 제안되었다. CORBA에 고장 감내성을 추가하기 위해서는 객체단위의 중복그룹의 관리, 호출 구조 및 이에 따른 고장인지 및 회복기법이 필요하다. 기존에 제안된 분산 시스템 환경에서의 고장인지 및 회복기법들은 프로세스 단위의 동작, 실행시간에 생성된 객체의 동적 환경구성 기능의 부재 등의 문제로 고장 감내형 CORBA에 적용시키기에는 많은 문제점을 가지고 있다 따라서, 본 논문에서는 사용자에게 고장투명성과 연속적인 서비스 제공을 보장하는 고장 감내형 CORBA에 필요한 핵심기술인 객체 그룹간 고장인지 방법 및 고장으로 부터의 회복 메커니즘을 제안한다.

1. 서 론

90년 초까지 자신들만의 정보시스템을 개발, 이용해 왔던 여러 기업 및 기관들은 네트웍 및 인터넷 기술의 발전으로 인해, 다른 기관의 정보시스템과 네트웍을 통한 상호연동이 필요하게 되었다. CORBA는 이렇게 광범위하게 분산되어 있는 다양한 정보 시스템을 통합하기 위한 표준으로 OMG(Object Management Group)에 의해 1991년에 제정되었다[1]. 그러나 CORBA는 점대점(Point-to-point) 통신을 기반으로 하고 있어 고장 감내성이 없다[2]. 따라서, 높은 신뢰성을 요구하는 은행업무나 항공 예약 시스템등에서 사용하는데에 많은 제약으로 작용하고 있다 현재, CORBA에 고장 감내성을 추가한 고장 감내형 CORBA에 관한 연구들이 많이 진행되고 있고, 이를 바탕으로 OMG에서는 고장 감내형 CORBA에 관한 표준화작업을 진행중이다[3].

지금까지 고장 감내형 시스템에 관한 많은 연구들이 진행되었다. 그러나 이러한 연구의 고장인지 및 회복 기술을 고장 감내형 CORBA에 직접 직용시키기에는 많은 문제점이 있다[4]. 우선, CORBA환경에서는 하나의 프로세스내에 많은 객체들이 포함되므로, 객체 하나의 고장이 프로세스의 고장을 발생시키지 않는다. 또한 분산객체들은 상속 및 내부 멤버 구조를 가지고 있고, 실행시간에 이러한 환경정보가 동적으로 구성된다. 따라서 고장인지 및 회복은 실행시간에 동적환경정보를 이용해 객체레벨로 이루어져야 한다. 기존 프로세스기반의 고장인지 메커니즘은 객체고장을 인지하지 못하며, 동적환경정보를 이용한 회복작업도 지원하지 않는다. 따라서, 본 논문에서는 고장 감내형 CORBA에 적용가능한 고장인지방법 및 회복 기법을 제안하고자 한다.

본 논문의 구성은 다음과 같다. 2장은 관련 연구로서 중복 시스템과 현재까지 제안된 고장 감내형 CORBA시스템에 대해 알아보고, 3장에서는 본 논문에서 제안하는 객체그룹간 중복객체의 고장인지방법 및 회복기법을 설명한다. 마지막으로 4장에서 결론 및 향후연구 계획으로 본 논문을 맺는다.

2. 관련 연구

분산환경에서의 중복 기법은 시스템의 가용도와 성능을 향상시키기 위해 사용된다. 이는 재무 시스템이나 예약 시스템처럼 약간의 손실이 치명적일 수 있는 시스템이나 통신의 가용성이 떨어지는 노트북과 같은 이동환경에서 사용된다. 중복 시스템에서 중요한 설계 요소로 중복정책을 들 수 있다. 중복 정책이란 중복데이터를 어떤 사이트에 이는 정도 중복시키며, 중복 데이터간의 일치성을 보장하기 위한 상호일관성유지 방법등을 결정하는 정책을 의미한다. 이러한 중복 정책은 일반적으로 클라이언트의 요청 처리방법에 따라 Active와 Passive 중복정책으로 분류한다[5] Active중복정책은 모든 복사본이 클라이언트의 요청을 수행하는 방법이고, Passive중복정책은 미리 선정된 주복사본만이 클라이언트의 요청을 수행하는 방법이다

지금까지 제안된 고장 감내형 CORBA시스템은 중복객체의 그룹관리 및 메시지 전달방법, 모듈의 위치 등에 따라 통합 방식, 서비스 방식, 인터셉트 방식, 이벤트 서비스 방식 등으로 분류할 수 있다. 이들은 고장 감내성을 제공하는 방법으로 객체의 중복기법(Replication Scheme)을 사용하며, 각 중복객체들은 그룹단위로 동작, 관리된다 따라서, 각 시스템들은 중복객체의 그룹을 관리하는 그룹 관리모듈과, 그룹내의 객체간 일관성유지를 위한 메시지 전달의 순서성을 지원하는 딜티캐스트(Multicast) 메시징 모듈을 포함하고 있어야 한다

통합방식은 현재의 ORB(Object Request Broker)내에 객체그룹 관리 모듈과 멀티캐스팅 모듈을 추가해 새로운 ORB를 만드는 방식을 사용하고 있다[6]. 클라이언트의 호출을 받은 ORB는 이를 개개의 중복객체들의 호출로 변환, 이를 각 서버객체에게 전달한다. 이에 대한 다중 응답을 처리해 클라이언트에게 올바른 응답을 전달하는 책임도 ORB가 담당한다. 서비스 방식은 CORBA 서비스에 그룹관리를 할 수 있는 새로운 서비스를 추가해, 중복호출 및 응답을 처리하는 방식이다[6]. 이는

*본 논문은 한국전자통신연구원(계약번호 : 99121) 위탁 과제로부터 부분적인 재정 지원을 받았음.

현재 개발된 ORB를 변경 없이 사용할 수 있는 장점이 있는 반면, 그룹 툴킷을 사용하지 않아 통합방식보다 고성능 멀티캐스트를 지원하지 못한다[2][6]. 인터셉트 방식은 클라이언트의 중복객체 그룹에 대한 호출 메시지를 가로채 이를 중복된 서버객체에 대한 호출로 변경 후, 중복 서버객체에게 이를 전달하는 방식이다. 이는 메시지의 인터셉트 모듈이 운영체제에 종속적인 단점이 있다. 마지막으로 이벤트 방식은 CORBA 서비스 중 이벤트서비스를 이용한다 즉 클라이언트의 호출을 이벤트로 등록해서 중복객체의 호출을 관리하는 방식이다.

3. 객체 그룹간 고장인지 및 회복 메커니즘

현재 고장 감내형 CORBA에 대한 표준안이 정립되어 있지 않기 때문에 지금까지 제안된 모든 방식을 모두 고려한 고장인지 기법 및 회복 메커니즘을 제안한다. 본 논문에서는 강상동작시 빠른 속도를 보이나, 회복시간이 긴 단점을 가지는 Passive방식의 중복시스템을 가정한다 그러나 Active방식의 중복시스템에서도 이 메커니즘을 적용할 수 있다 제안된 메커니즘은 전반적인 중복객체 관리와 호출/응답을 담당하는 Replcation Coordinator(RC), 고장인지 및 통보, 통계등을 담당하는 Fault Detector (FD), 로그정보관리 및 검사점 작업을 하는 Operation Logger(OL), 고장시 회복기능을 하는 Recover(RV)등로 구성된다(<그림 1>참조) 각 모듈의 구현과 위치는 중복관리 및 멀티캐스팅 모듈의 구현방법에 따라 달라진다. 통합방안을 사용한 경우에 각 모듈은 ORB내에 포함되어 ORB내의 내부 메커니즘으로 작동한다. 서비스 방안의 경우, 고장 감내형 서비스내에 IDL로 각 기능이 구현된다. 인터셉트 방안에서는 ORB로의 메시지를 가로채는 그룹 툴킷의 일부로 구현된다.

3.1 메커니즘 모듈 소개

RC는 중복객체 그룹의 생성, 멤버쉽 변경, 삭제 등의 전반적인 객체 중복의 관리기능과 호출/응답의 조정을 담당하는 모듈이다. 한 개의 중복그룹을 관리하는 RC는 전체 사이트에 한 개만이 존재한다. 중복객체 그룹의 정보는 RCT라 명명한 RC내에 데이블 형태로 저장된다. RCT는 <표 1>과 같이 객체 그룹의 이름, 객체의 식별자, 중복객체가 존재하는 사이트정보, 주복사본의 고장시 새로운 주복사본 선정의 기준이 되는 각 객체에 대한 핑(Ping) 시간, 객체의 상태 등으로 구성된다. 고장 감내형 객체는 고장 감내성을 제공하는 고장 감내성 클래스를 상속받아야 한다. 객체의 생성시 고장 감내성 클래스는 RC에 객체 생성사실을 보고한다. RC는 새로 생성된 객체에 대한 객체그룹이 생성되지 않은 경우에는 새로운 객체 그룹을 테이블에 추가하고, 새로 생성된 객체의 객체상태필드를 'Primary'로 설정한다. 이에 대한 구현은 명명서비스(Naming Service)의 연동으로써 이루어진다. 객체의 생성시 구현 객체는 자신의 객체 참조자를 명명서비스에 등록하게 된다. 이때 구현서비의 객체참조 대신에 RC의 객체참조를 등록시킴으로써 클라이언트의 호출을 RC가 받아서 적절한 처리를 할 수 있게 한다. 이미 같은 그룹의 객체가 있는 객체를 생성할 경우, 그 객체이름에 대응되는 이름이 이미 명명서비스에 등록되어 있으므로, 생성사실을 RC에게만 알리면 된다 RC는 새로 생성된 객체를 그룹의 멤버로써 추가시키고, 이 객체의 상태필드를 'Backup'으로 설정한다 객체 삭제 시에도 RC에 삭제 정보가 보고된다 삭제된 객체가 그 그룹의 주 복사본일 경우는 RC 테이블의 핑 시간을 참조해 새로운 주 복사본을 선택해야 한다

객체그룹이름	객체식별자	사이트정보	핑 시간	객체상태
Account	Account1	165.246.31.164	1.1ms	Primary
	...	..	..	....

<표 1> RCT 테이블의 구성

FD는 내부에 저장하고 있는 타임아웃 데이블을 참조해 타임아웃기법으로 고장을 판단, 보고하는 모듈이다 타임아웃 테이블은 각 객체의 타임아웃 시간이 저장되어 있는 테이블이다 타임아웃 시간은 객체의 생성시와 주복사본 변경시에 설정된다. 클라이언트의 호출정보가 들어오면 FD는 어떤 객체에 대한 호출인지를 검사해 타임아웃 데이블에 설정되어 있는 시간만큼의 타이머를 기동한다 여기서 타임아웃시간을 너무 짧게 설정하면 거짓고장인지(False Fault Detection)을 발생시킬 수 있다 거짓고장인지란 FD가 정상수행중인 객체나 호스트에 고장 발생을 검출하는 것을 말한다[7] 반대로 타임아웃시간을 너무 길게 설정하면 고장인지의 시간이 길어져 전체적인 시스템성능이 저하되므로 적절한 타임아웃시간의 설정이 필요하다 추가적으로 FD는 고장에 대한 통계자료를 산출하는데도 사용될 수 있다. FD에는 *Start_Timer(), Stop_Timer(), Retrieve_Time(), Report_Fault()* 등의 메소드를 포함하고 있다.

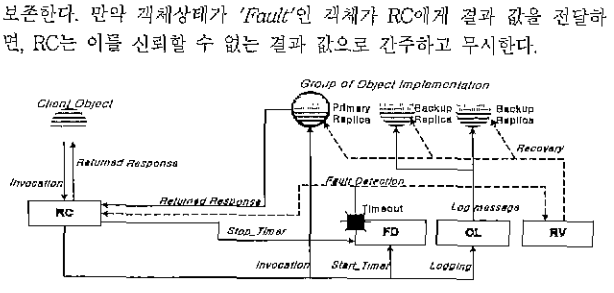
OL은 주복사본의 고장시 새로 선정된 주복사본과 이전 주복사본과의 상호일관성유지를 위한 로깅작업과 회복시간의 단축을 위한 검사점작업을 담당한다 몇 번의 갱신연산마다 검사점 작업을 수행할지는 사용자가 설정할 수 있다. 빈번한 검사점 작업은 시스템의 성능을 저하시킬 수 있고, 실행간격이 너무 긴 검사점 작업은 고장회복시간을 길게 하므로, 시스템 설계시 적절한 검사점 실행간격의 설정이 필요하다. OL에는 *Logging(), Retrieve_Log()* 등의 메소드를 포함한다.

RV는 고장객체를 재시작시키고, 객체간 상태전이와 로그정보를 이용해 이를 회복한다. 이를 위해 RV는 *Restart_Object(), Make_Primary(), Make_Backup(), Get_Internal_State(), Set_Internal_State()* 등의 메소드를 포함하고 있어야 한다. 고장객체가 주 복사본일 경우, 새로운 주 복사본의 선정과 상호일관성(Mutual Consistency)유지 등의 작업도 RV에서 담당한다

3.2 객체그룹간 고장인지

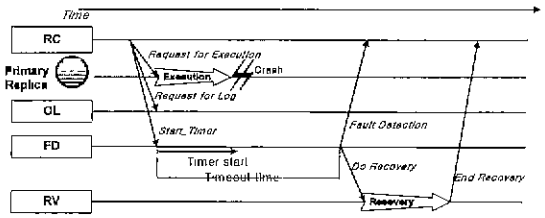
고장 감내형 CORBA에서는 프로세스단위가 아닌 객체단위의 고장인지 메커니즘이 필요하다[4]. 이를 위해 객체호출시 마다 객체의 고장을 검출하는 객체단위의 타임아웃기법을 사용한다. 또한 호출이 빈번하지 않은 객체의 고장인지를 위해 폴링(Polling)기법을 사용하고, 관리모듈의 고장인지를 위해 하트비트(Heartbeat)기법을 사용한다.

<그림 1>은 제안된 메커니즘이 어떻게 고장을 인지, 보고, 회복하는지를 설명한 모듈간의 상호동작을 나타낸 그림이다. 그림에서 실선은 정상수행시의 모듈간 메시지전달을, 점선은 고장시의 모듈간 메시지전달을 나타낸다. 클라이언트의 호출을 전달받은 RC는 RCT를 참조해 현재 객체 중복그룹중 주 복사본으로 설정된 복사본을 찾아 요청을 전달한다. 동시에 RC는 FD에게 'Start_Timer' 메시지를 보내고, OL에게 로깅을 위해 호출정보를 전달한다. FD는 'Start_Timer' 메시지가 도착하는 즉시 내부의 타임아웃 테이블을 참조해 호출된 객체에 설정되어 있는 시간만큼의 타이머를 시작한다. 만약 FD가 타임아웃을 발생시키기 전에 복사본이 호출의 처리를 마치고, 그 결과 값이 RC에 도착하면 RC는 결과 값을 클라이언트에게 전달해 요청을 종료한다. 또한 RC는 FD에게 'Stop_Timer' 메시지를 보내 타이머를 종료시킨다. 만약 주복사본의 수행 결과값보다 FD의 타임아웃메시지가 RC에게 먼저 도착하면, RC는 RCT의 객체상태필드를 'Fault'로 바꾸고 RV가 객체를 회복한 후 회복완료 메시지를 보낼 때까지 이 객체로의 호출을 큐(Queue)에 보존한다. 만약 객체상태가 'Fault'인 객체가 RC에게 결과 값을 전달하면, RC는 이를 신뢰할 수 없는 결과 값으로 간주하고 무시한다.



<그림 1> 메커니즘모듈간의 상호동작

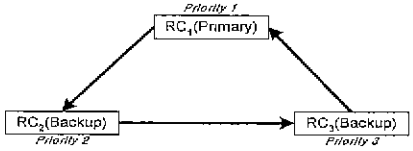
<그림 2>는 고장발생시의 고장인지 및 회복 모듈간 메시지 전달과정을 시간축을 기준으로 작성한 것이다. 가로축은 시간축이고, 화살표로 그린 것이 메시지의 전달과 그 방향, 사각형으로 표시된 것이 메시지의 내용이다.



<그림 2> 고장발생시 메시지 전달과정

호출이 빈번하게 일어나지 않는 객체의 고장인지를 위해 폴링기법을 사용한다. 이는 정기적으로 중복그룹객체들에게 핑 메시지를 전송하여 객체의 생존여부를 판단하는 방법이다

RC등의 메커니즘모듈에 고장이 발생하면 시스템 전체의 동작이 정지되므로, 객체의 고장이외에 메커니즘모듈의 고장도 고려되어야 한다[8] 이를 위해서는 메커니즘모듈 또한 중복되어야 하며, 이에 대한 고장인지 및 회복기능도 가져야 한다 본 논문에서는 메커니즘모듈의 고장인지를 위해 중복된 메커니즘모듈의 원형구조와 이들간의 하트비트 메시지를 교환하는 방법을 사용한다. 하트비트란 객체가 다른 객체나 조정자에게 자기 자신의 존재를 정기적으로 알리는 방법이다. <그림 3>은 메커니즘모듈인 RC의 고장인지를 위한 중복된 메커니즘모듈에서의 하트비트 메시지의 교환을 나타낸 그림이다.



<그림 3> 메커니즘모듈의 고장인지를 위한 구조

중복된 메커니즘모듈에는 우선순위를 부여하고 정기적으로 하트비트 메시지를 다음 우선순위를 갖는 메커니즘모듈에 전송하게 된다. 만약 예정된 시간에 하트비트 메시지가 도착하지 않으면, 이전 우선순위를 가지는 메커니즘모듈에 고장이 발생하였다고 판단한다. 예를 들어 <그림 3>에서 RC_2에 예정된 하트비트 메시지가 도착하지 않으면 RC_2는 현재의 주메커니즘모듈 RC_1와 고장을 인지하고 자신이 주 메커니즘모듈이 되고 다른 RC들에게 이를 알린다.

3.3 검사점 작업 및 회복

본 논문에서는 주 복사본의 고장시 회복시간의 단축을 위해 검사점 기법을 사용한다. 이는 두 가지 방법으로 구현될 수 있다 우선, 주복사본의 내부상태(Internal State)를 전이시킴으로서, 부 복사본을 주복사본의 상태와 일치시키는 방법이 있다 이는 객체가 가지고 있는 데이터가 많은 경우, 주복사본의 내부상태를 읽기 위한 병목현상이 발생하는 단점이 있다 두 번째로 모든 부복사본내에 저장되어 있는 로그정보내의 연산을 각 부 복사본이 실행하는 방법이 있다 이 방법은 같은 연산을 중복그룹내의 모든 객체가 실행해야 하므로, 시스템 전체의 자원을 낭비를 가져올 수 있다. 검사점 작업이 완료되면, 각 부 복사본내의 로그정보는 더 이상 유효하지 않으므로 삭제할 수 있다

본 논문에서는 객체 제시작 및 로깅을 이용한 회복 방법을 사용한다. FD는 고장 발생사실과 고장이 발생한 객체를 RC와 RV에게 보고한다. RC는 RV가 고장회복완료메시지를 보낼 때까지 대기한다 회복시 RV는 고장객체를 재시작 시키다. 제시작된 객체는 RV의 *Get_Internal_State()* 메소드와 *Set_Internal_State()* 메소드를 통해 정상동작중인 객체의 내부 상태를 전이받게 된다.

만약 고장객체가 주복사본인 경우는 주복사본과 부복사본간의 상호일관성(Mutual Consistency)유지를 위한 추가적인 작업이 필요하다. 우선 RV는 RC내에 저장되어 있는 RCT에서 그룹에 대한 정보와 복사본의 정보를 얻어와 새로 주 복사본이 될 복사본을 선정한다. 새로 선정된 주 복사본은 마지막 검사점 작업까지는 이미 반영된 상태이다. 따라서, 자신이 저장하고 있는 로그를 순차적으로 수행함으로써, 고장이 발생된 예전 주 복사본과 동일한 상태를 가지게 된다. RV가 수행하는 회복의 알고리즘은 <그림 4>와 같다

```
RE..Recovery(Object_ID) {
  Restart(Object_ID),
  Set_Internal_State(Object_id, Get_Internal_State(Other_Normal_Object)),
  if(RC .RCT(Object_ID) = Primary_state) {
    New_Primary_Object = Select_New_Primary(RC_RCT(All_Object)),
    While ( Not_Exist(OL Retrieve_Log())) {
      DO_Execution(OL' Retrieve_Log()),
    }
    RC' Set_RTC_Backup(Object_ID),
    RC Set_RCT_Primary(New_Primary_Object),
  }
  Message_to_RC("Recovery is Done").
}
```

<그림 4> 회복 알고리즘

각 복사본마다 유지되는 로그정보는 OL이 부여한 로깅 메시지의 고유 식별자, 호출적용 그룹 식별자, 호출된 메소드의 종류와 식별자, 메소드에 따른 파라미터 등이 기록된다. 멀티캐스트 엔진이 메시지에 대한 완전 순서화(Total Ordering)와 호출의 원자성(Atomicity of Invocation)을 제공하므로, 회복 수행시에는 단지 기록된 로그를 순차적으로 실행한다.

RV의 회복완료메시지를 받은 RC는 새로운 주복사본에게 현재 큐에 쌓여있는 호출을 차례대로 전달해 사용자에게 고장투명성과 계속적인 서비스를 보장한다.

4. 결론 및 향후연구

본 논문에서는 고장 감내형 CORBA에 적용 가능한 객체 고장인지 메커니즘을 제안했다. 기존 프로세스 기반의 고장 감내형 시스템의 고장인지 및 회복기법은 프로세스기반 구조와 정적 환경정보이용 등으로 인해 실행시간에 동적으로 환경이 구성되는 객체기반의 CORBA환경에 적용되지 못한다. 제안된 메커니즘은 객체단위의 동적, 실행시간에 객체생성 및 삭제 등의 동적환경 지원 등 고장 감내형 CORBA을 위한 고장인지 및 회복을 제공한다. 또한 사용자에게 고장투명성을 보장하고 연속적인 서비스를 가능케 한다.

향후연구로 ARIES등의 보다 빠른 회복 수단을 제공하는 알고리즘을 객체회복에 적용해 회복 속도를 개선하고, 곧 제정될 고장 감내형 CORBA에 대한 표준안에 준하는 고장 감내형 시스템의 구현을 통해 보다 성능이 우수한 고장인지 및 회복 메커니즘을 연구가 필요하다.

참고문헌

[1] Object Managament Group, "The Common Object Request Broker' Architecture and Specification," http://www.omg.org, 1999
[2] 김기영, 최 훈, "장애 감내형 CORBA," 한국정보과학회 정보과학회지, 제 17권 제 7호, 1999
[3] OMG, "Fault Tolerance RFP," http://www.omg.org/techprocess/meetings/schedule/Fault_Tolerance_RFP.html, 1998.
[4] P.Chung, Y. Huang, S Yajnik, "DOORS Providing Fault Tolerance for CORBA Applications," The Proceedings of IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing, 1998
[5] J Gray, P. Holland, P O'Neil and D Shasha, "The Dangers of Replication and a Solution," *The Proceedings of 1996 ACM SIGMOD*, 1996.
[6] P. Felber, B. Garbinato, R. Guerraoui, "Towards Reliable CORBA Integration vs. Service Approach," The Proceedings of 11th European Conference on Object-Oriented Programming, May 1996
[7] M. Takemoto, "Fault-tolerant Object on Network-wide Distributed Object-oriented Systems for Future Telecommunications Applications" The Proceedings of Pacific Rim International Symposium on Fault-Tolerant Systems, December 1997