고장 감내 CORBA를 위한 로깅 및 회복 기법

김 홍 식*○ 구 경 이* 김 중 한** 김 유 성*
인하대학교 전자계산공학과*, 광주과기원 정보통신공학과**
{g1991279, g9721046}@inhavision.inha.ac.kr* yskim@inha.ac.kr*
s911669@banana.kjist.ac.kr**

A Logging and Recovery Mechanism for Fault Tolerant CORBA

Hong-Sik Kim○* Kyong-I Ku* Joong-Han Kim** Yoo-Sung Kim*
Dept. of Computer Science & Engineering, Inha Univ.* DIC, KJIST**

요 약

분산 객체 시스템을 통합하기 위한 표준인 CORBA(Common Object Request Broker Architecture)는 분산 환경에서 더욱 더 자주 발생하는 고장에 대한 회복 수단을 제공하지 않기 때문에, 높은 신뢰성을 요구하는 상업용 어플리케이션에 도입되지 못하고 있다. 이에 따라, CORBA 구조 내에 자체적인 고장 감내 구조를 추가해 신뢰성 높은 서비스를 제공할 수 있는 고장 감내 CORBA에 대한 연구가 진행되고 있다. 그러나, 2000년 4월에 채택된 고장 감내 CORBA에 대한 명세서에서는 객체의 인터페이스를 정의하는 기술 언어인 CORBA IDL(Interface Definition Language)로 객체 단위의 중복과 이를 위한 시스템의 구조 및 각 모듈의 설계를 요약하고 있으나, 그 세부적인 메커니즘이 제시되지 않고 있다. 따라서, 본 논문에서는 CORBA에 고장 감내성을 부여하기 위해 세부적인 고장 감내 CORBA를 위한 로깅 및 회복 기법을 제안한다.

1. 서론

CORBA는 네트워크 상에 광범위하게 분산되어 있는 이질적인 분산 객체 시스템을 통합하기 위한 표준으로 만들어졌다[1]. 그러나, CORBA는 특히 분산 환경에서 더욱 더 자주 발생하는 고장에 대한 회복 수단을 제공하지 않아, 높은 신뢰성을 요구하는 상업용 어플리케이션에 도입되지 못하고 있다. 따라서, CORBA 구조 내에 자체적인 고장 감내 구조를 추가해 신뢰성 높은 서비스를 제공할 수 있는 고장 감내 CORBA에 대한 연구가 늘어가고 있다. 현재 OMG(Object Management Group)에서는 CORBA에 고장 감내성을 부여한 고장 감내 CORBA에 관한 지금까지의 연구들을 바탕으로 고장 감내 CORBA에 관한 표준화작업을 진행하고 있다[1,2,3]. OMG는 2000년 4월에 고장 감내 CORBA에 대한 명세서([4])를 채택하였으며, 기존의 CORBA구조에 고장 감내성을 부여하기 위해 객체 단위의 중복과 이를 위한 시스템의 구조 및 각 모듈의 역할을 제시하고 있다. 그러나, 명세서는 객체의 인터페이스를 정의하는 기술 언어인 CORBA IDL로 그 설계를 요약하고 있어, 그 세부적인 메커니즘이 제시되지 않고 있다[4]. 따라서, CORBA에 고장 감내성을 부여하기 위해 세부적인 고장 감내 CORBA를 위한 로깅 및 회복 기법의 연구가 필요하다.

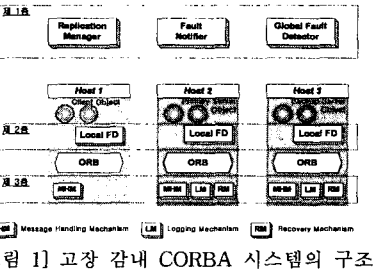
본 논문에서는 고장 감내 CORBA 시스템에서 정상 수행 시에 수행되는 로깅 기법과 객체가 고장 발생한 후, 정상 수행 시 저장한 로그 정보를 이용해서 회복하는 회복 기법을 제시한다. 이러한 회복 기법은 기존의 CORBA 시스템이 고장 감내성을 제공하지 못하는 단점을 해결하여, 분산 환경에서 사용자에게 고장 투명성을 제공하면서, 신뢰성 있는 서비스를 지속할 수 있도록 한다.

본 논문의 구성은 다음과 같다. 2장은 본 논문에서 채택한 고장 감내 CORBA 시스템의 구조와 이 시스템에 사용되는 로깅에 관련된 큐들에 대해 설명한다. 3장은 고장 감내 CORBA를 위한 로깅 기법과 이를 바탕으로 회복하는 회복 기법을 설명한다. 마지막으로 4장에서는 결론 및 향후 연구 방향으로 본 논문을 맺는다.

2. 관련 연구

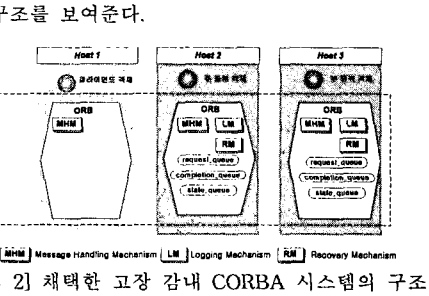
2.1 고장 감내 CORBA 시스템의 구조

고장 감내 CORBA 시스템의 전체적인 구조는 [그림 1]과 같으며 이는 2000년 4월에 OMG에서 채택한 고장 감내 CORBA 명세서에서 제안된 구조를 따르고 있다.



[그림 1] 고장 감내 CORBA 시스템의 구조

위와 같은 구조에서 제 3층의 MHM(Message Handling Mechanism), 로깅 메커니즘(Logging Mechanism), 회복 메커니즘(Recovery Mechanism)은 로깅과 회복을 담당하는 부분이다. 본 논문에서는 로깅 및 회복 메커니즘을 위해 제 3층의 모듈들을 ORB(Object Request Broker)에 통합한 구조를 따른다. [그림 2]는 본 논문에서 채택한 고장 감내 CORBA 시스템의 제 3층의 구조를 보여준다.



[그림 2] 채택한 고장 감내 CORBA 시스템의 구조

2.2 로깅에 필요한 큐들

[그림 2]에서 보이는 request_queue, completion_queue, state_queue는 로깅 관련 메시지 전달에 필요한 큐들로서 ORB가 초기화될 때 생성된다. <표 1>은 큐의 역할 및 송신 측과 수신 측을 나타낸다.

<표 1> 로깅에 필요한 큐들

	역할	Sender	Receiver
request_queue	request에 대한 로그 정보 전달	주 중복 객체의 ORB	주 중복 객체와 부 중복 객체들의 LM
completion_queue	request에 대한 로그 정보 저장 완료 전달	주 중복 객체의 LM	주 중복 객체의 ORB
state_queue	주기적 로그 정보 전달	주 중복 객체의 LM	부 중복 객체들의 LM

3. 고장 감내 CORBA를 위한 로깅 기법과 회복 기법

본 논문에서의 고장 감내 CORBA는 고장 감내성을 제공하기 위해 하나의 주 중복 객체만 클라이언트의 request에 대해 응답하고 여러 개의 부 중복 객체들은 단지 로깅 메커니즘이 로그 정보를 관리하는 Warm Passive 중복 정책([5])을 따른다. 따라서, 만약 주 중복 객체가 고장이 발생하면 ORB가 부 중복 객체들 중에서 새로운 주 중복 객체를 선정하여 새로운 주 중복 객체의 로깅 메커니즘에 저장된 로그 정보를 이용하여 회복한다.

3.1 고장 감내 CORBA를 위한 로깅 기법

고장 감내 CORBA를 위한 로깅은 크게 클라이언트가 주 중복 객체에게 request를 요청할 때, request에 대한 정보를 얻어 저장하는 request에 대한 로깅과 주 중복 객체에 대해서 주기적으로 객체의 상태를 얻어 저장하는 주기적 로깅으로 나눌 수 있다.

3.1.1 request에 대한 로깅

request에 대한 로깅은 클라이언트가 주 중복 객체에게 request를 요청했을 때, 주 중복 객체와 부 중복 객체들의 로깅 메커니즘이 request_queue를 통해 request에 대한 정보를 얻어 저장하며 이 로그 정보는 회복 시 사용된다. request에 대한 로그 정보의 형식은 [그림 3]과 같다.

request_id	source_id	operation	parameter_list	object_group_id	CSN

- request_id : request에 대한 식별자
- source_id : 클라이언트에 대한 정보(Port 정보)
- operation : request에 대한 operation 이름
- parameter_list : IDL에 정의된 operation에 대한 파라미터들
- object_group_id : 해당 객체 그룹 식별자
- CSN : 객체의 체크 포인팅을 나타내 주는 순차 번호

[그림 3] request에 대한 로그 정보 형식

request에 대한 로그 정보는 [그림 3]과 같이 6가지로 구분된다. request_id는 클라이언트가 주 중복 객체에게 요청하는 request에 대한 식별자이며, source_id는 클라이언트에 대한 포트 정보를 나타낸다. operation은 request에 대한 operation의 이름을 나타내며 parameter_list는 IDL에서 정의된 operation에 대한 파라미터들을 나타내 주는 정보로 바이트 배열로 구성된다. object_group_id는 request에 대한 목적 객체가 속한 그룹 식별자를 나타낸다. 마지막으로, CSN(Checkpoint Sequential Number)은 주 중복 객체가 체크 포인팅을 할 때마다 증가하는 순차 번호로 체크 포인트 시점이전의 request에 대한 로그 정보를 삭제하거나 회복 시 가장 최근의 체크 포인트 시점이후의 request에 대한 로그 정보를 얻어올 때 사용된다. CSN은 클라이언트가 주 중복 객체에게 request를 요구할 때 ORB가 로깅 메커니즘의 getCSN() 함수를 이용해 로깅 메커니즘이 가지고 있는 CSN을 얻어서 request에 대한 로그 정보에 설정한다.

3.1.2 주기적 로깅

주기적 로깅을 하기 위해서는 객체가 Checkpointable 인터페이스를 상속받아야 하고 이 인터페이스에 정의된 get_state() 메소드를 이용해 객체의 상태를 주기적으로 얻는다. 2000년 4월에 채택된 고장 감내 CORBA 명세서에서는 Checkpointable 인터페이스를 다음과 같은 IDL로 정의하고 있다[4].

```
interface Checkpointable {
  State get_state() raises(NoStateAvailable);
  void set_state(in State s) raises(InvalidState);
};
```

[그림 4] Checkpointable 인터페이스의 IDL

Checkpointable 인터페이스는 get_state()와 set_state() 메소드를 정의하고 있다. get_state() 메소드는 객체의 상태를 얻어오며, set_state() 메소드는 get_state() 메소드에서 얻은 상태를 객체에 설정하는 역할을 한다. 따라서, get_state() 메소드는 로깅 시 사용되며, set_state() 메소드는 회복 시 사용된다. 이 메소드들의 구현은 객체가 Checkpointable 인터페이스를 상속받기 때문에 그 객체 구현자의 책임이 된다. get_state() 메소드로 주기적으로 객체의 상태를 얻으면, 얻은 객체 상태는 객체 그룹 식별자와 같이 부 중복 객체들의 로깅 메커니즘에게 state 큐를 통해 전달되어 저장된다. 주기적 로깅에서의 로그 정보 형식은 [그림 5]와 같다.

object_group_id	State값

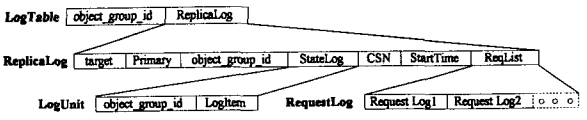
- object_group_id : 해당 객체 그룹 식별자
- State값 : 객체의 상태를 나타내는 값

[그림 5] 주기적 로깅에서의 로그 정보 형식

주기적 로깅에서의 로그 정보는 [그림 5]와 같이 2가지로 구분된다. object_group_id는 주 중복 객체가 속해 있는 객체 그룹의 식별자를 나타내고, State값은 주 중복 객체의 상태를 나타내 주는 정보로 바이트 배열로 구성된다.

3.1.3 로깅 메커니즘의 자료 구조

로깅 메커니즘은 다음과 같은 자료 구조로 나타내어진다.



[그림 6] 로깅 메커니즘에서의 자료 구조

로깅 메커니즘은 [그림 6]과 같이 ReplicaLog 객체를 가지고 있으며 이는 해쉬 테이블인 LogTable 객체로 관리된다. ReplicaLog 객체는 해당 객체, 주 중복 객체인지 아닌지를 나타내는 Primary 플래그, 객체 그룹 식별자, 주기적 로그 정보(StateLog객체), 객체의 체크 포인팅 순서를 나타내는 CSN, 현재의 시스템 시간을 나타내는 StartTime, 그리고 마지막으로 request에 대한 로그 정보를 저장하는 ReqList 테이블로 구성되어 있다. 위와 같은 자료 구조는 주 중복 객체와 부 중복 객체들의 로깅 메커니즘에서 관리하며, 새로운 주 중복 객체를 회복할 때 사용된다.

3.1.4 로깅 메커니즘

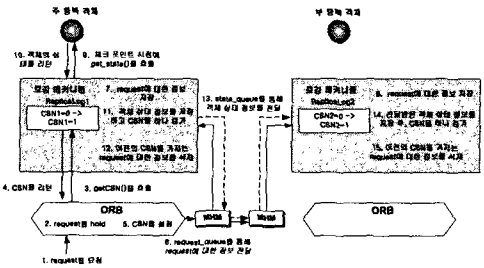
고장 감내 CORBA 시스템에서 로깅 메커니즘은 각 호스트마다 존재하며, request에 대한 로깅과 주기적 로깅을 수행한다. 로깅 메커니즘은 ORB에 의해서 startLog() 함수가 호출되면서 시작된다. 이 함수는 객체마다 request에 대한 로그 정보와 주기적 로그 정보를 저장하는 ReplicaLog 객체를 생성하고 이를 LogTable 해쉬 테이블에 삽입한다. 그 후, LMStart() 함수가 호출된다. 실질적인 로깅 작업을 하는 LMStart() 함수는 [그림 7]과 같은 알고리즘으로 동작한다.

```
public void LMStart() {
 StartTime = 현재의 시스템의 시간;
 while(TRUE) {
  if (request_queue <> empty) then {
    request에 대한 로그 정보 RL을 얻어 ReplicLog 객체 R에 저장;
    if (ReplicaLog 객체의 primary 플래그 P == TRUE) then
        completion 큐에 R을 삽입; }
  if (현재의 시스템 시간-StartTime >= 체크 포인트 시간) then {
    StartTime = 현재의 시스템의 시간;
    while(LogTable의 R == empty) {
      if (P == TRUE) then {
        StateLog = get_state();
        R의 CSN++;
        state 큐에 StateLog 객체를 삽입;
        이전의 CSN을 가지는 모든 RL를 삭제; }
      else {            // 부 중복 객체이면
        if (state_queue <> empty) then {
          state 큐에서 처음 객체 FO를 읽음;
          if (ReplicaLog의 group_id == FO의 group_id) then {
            R = StateLog 객체;
            R의 CSN++;
            이전의 CSN을 가지는 모든 RL를 삭제;  }}}}}
```

[그림 7] LMStart() 함수의 알고리즘

3.1.5 로깅 메커니즘에서 CSN의 역할

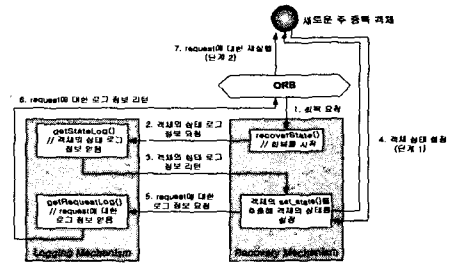
CSN은 request에 대한 로그 정보와 ReplicaLog 객체에 존재해 주 중복 객체와 부 중복 객체들의 로그 정보의 동기화를 위해 사용된다. 따라서, 주 중복 객체의 로깅 메커니즘에서 체크 포인트를 한 후에, 이전의 request에 대한 로그 정보를 삭제할 때 올바른 request에 대한 로그 정보를 삭제할 수 있고, 회복 시, 가장 최근의 체크 포인트 시점이후의 request에 대한 로그 정보를 얻을 때 올바른 request에 대한 로그 정보를 얻을 수 있다. [그림 8]은 로깅 메커니즘에서 CSN의 역할을 보여준다.



[그림 8] 로깅 메커니즘에서의 CSN의 역할

3.2 고장 감내 CORBA를 위한 회복 기법

고장 감내 CORBA를 위한 회복 기법은 크게 객체 상태 설정 단계와 request에 대한 재실행 단계로 나뉜다. 고장 감내 CORBA 시스템에서의 회복 메커니즘의 동작 원리는 [그림 9]와 같다.



[그림 9] 회복 메커니즘의 동작 원리

회복 메커니즘은 ORB가 선정한 새로운 주 중복 객체에 대해 실행되며 해당 회복 메커니즘의 recoverState() 함수를 호출함으로써 시작된다. 실제적인 회복 작업을 하는 recoverState() 함수는 [그림 10]과 같은 알고리즘으로 동작한다.

```
public FT.RequestLog[] recoverState(객체 그룹 식별자 GID,
                                    회복하려는 객체 obj) {
  상태 로그 정보 S=로깅 메커니즘 LM의 getStateLog(GID);
  obj.set_state(S);          // 새로운 주 중복 객체의 상태 설정
  return LM의 getRequestLog(GID); //request에 대한 정보 리턴 }
```

[그림 10] recoverState() 함수의 알고리즘

새로운 주 중복 객체는 객체 상태 설정 단계에서 가장 최근의 체크 포인트 시점의 객체 상태로 설정되고 request에 대한 재실행 단계에서 가장 최근의 체크 포인트 시점이후의 request를 재실행됨으로써 사용자에게 지속적인 서비스 제공하게 된다.

4. 결론 및 향후 연구 방향

본 논문에서는 고장 감내 CORBA를 위한 로깅 및 회복 기법을 제안했다. 제안된 로깅 및 회복 기법은 CORBA에 고장 감내성을 부여하기 위한 핵심 부분이다. 로깅 기법은 크게 정상 수행 시, request에 대한 정보를 저장하는 request에 대한 로깅과 주 중복 객체에 대해 주기적으로 객체의 상태를 얻어 저장하는 주기적 로깅으로 나뉜다. 회복 기법은 주 중복 객체가 고장이 발생하면, 새로운 주 중복 객체에 대해 정상 수행 시에 저장된 가장 최근의 체크 포인트 시점의 객체 상태 로그 정보와 가장 최근의 체크 포인트 시점이후의 request에 대한 로그 정보를 이용해 객체 상태 설정 단계와 request에 대한 재실행 단계로 회복한다. 그러므로, 고장 감내 CORBA는 사용자에게 고장 투명성을 제공하면서 신뢰성 있는 지속적인 서비스를 보장할 수 있다.

향후 연구로는 계속적인 실험으로 제안된 고장 감내 CORBA를 위한 로깅 및 회복 기법의 동작 원리를 검증하고 현재 ETRI에서 개발중인 iORB(Internet Object Request Broker)와의 통합을 통해 고장 감내 CORBA를 구현한다.

참고문헌

[1] Object Management 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 FTF - Draft Adopted Submission For Fault Tolerant CORBA." *OMG TC Document ptc/00-03-04*, 2000.

[4] OMG, "Fault Tolerant CORBA Specification 1.0 - Fault Tolerance Final Adopted Specification," *OMG TC Document ptc/00-04-04*, 2000.

[5] Inprise Corporation Inc., L.C., "Fault Tolerant CORBA using Entity Redundancy," *OMG TC Document orbos/98-10-09*, 1998.