장애 감내형 디지털도서관 서비스 구조 설계

김기영, 설동명, 최훈
충남대학교 컴퓨터공학과

Architecture Design of a Fault Tolerant Digital Library Service

Ki-Young Kim, Dong-Myoung Sul, Hoon Choi
Dept. of Computer Engineering, Chungnam National University

신뢰성 향상은 분산 시스템의 기본 목표이며, 이를 위하여 서비스를 제공해 주는 서버 객체의 중복이 불가피 하다 본 논문에서는 중복과 함께 고려되어야 할 기본적인 사항들을 알아보고, 현재의 시스템들을 CORBA 환경과 CORBA에 기반하지 않은 환경으로 나누어, 각각의 시스템 환경하에서 신뢰성 있는 분산 어플리케이션의 구현을 제공하기 위해 진행되었던 연구 사례로 Orbix + Isis, Electra 와 Chameleon 을 소개한다. 또한 디지털 도서관에서 핸들 서비스를 제공하고 있는 핸들 서버의 장애 감내를 위한 장애 감내형 다중화 서버 모델을 제시하고, 이를 제공하기 위한 ILU의 인터페이스를 제시 함으로써, 장애 감내형 디지털 도서관 서비스 구조 설계에 관한 연구를 소개한다.

1. 서론

신뢰성의 향상은 분산시스템의 기본 목표이자 반드시 제공해야 할 과제이다. 신뢰성 있는 분산시스템을 구현하기 위하여 사용자에게 중요한 서비스를 제공하는 객체 즉, 서비스 객체를 중복시켜야 하는데 이때 아래와 같은 몇 가지 사항들이 함께 고려되어야 만 된다.

첫째, 여러 호스트 상에 분산되어 있는 서비스 객체들은 그룹 멤버쉽을 가지면서 하나의 그룹으로 묶여 관리되고, 사용자에게는 단일 서비스 객체로 보여져야만 한다.

둘째, 서비스 객체나 서비스 객체가 위치한 호스트에 장애가 발생하면 이를 발견하여 다른 서비스 객체 멤버들과 그룹을 관리하는 객체 또는 시스템 관리자에게 장애 발생 사실을 알리고, 장애가 발생한 호스트나 서비스 객체를 복구해야만 한다.

셋째, 같은 그룹에 속한 서비스 객체들은 어떠한 상황 – 새로운 서비스 객체가 그룹 멤버로서 참여를 원하는 경우, 기존의 서비스 객체가 그룹을 나가는 경우, 서비스 객체에 고장이 발생하여 예비 서비스 객체가 그 역할을 대신하는 경우 등 – 에서도 서로 동일한 내부 상태와, 일관된 뷰 – 현재 그룹에 참여하고 있는 서비스 객체들의 동적인 리스트 – 를 가지고 있어야만 한다

분산된 서비스 객체들의 중복과 더불어 위의 사항들이 함께 고려되어 질 때 진정한 신뢰성 향상은 이루어질 수 있다. 위의 사항들을 고려하여, 많은 연구들이 활발히 진행되고 있다. 현재까지 이루어진 많은 연구들은 객체 지향 기법의 개방형 분산 시스템 디자인에 관한 표준을 제공하고 있는 CORBA(Common Object Request Broker Architecture)[1] 기반 환경에서의 연구와 CORBA에 기반하지 않은 환경에서의 연구로 크게 두 가지로 나눌 수 있는데, 전자의 경우 대표적인 연구로는 Orbix + Isis, Electra[2]가 있으며 후자의 경우 대표적인 연구로는 Chameleon[3]을 들을 수가 있다.

본 논문에서는 이들을 간략히 소개하고, 현재 연구 중인 디지털 도서관에서의 장애 감내형 다중화 서버 구조를 소개하고자 한다. 논문의 구성은 다음과 같다. 2 절에서는 Orbix + Isis 와 Electra 를 살펴보고, 3 절에서는 CORBA 에 기반하지 않은 환경 연구인 Chameleon, 4 절에서는 장애 감내형 디지털 도서관을 위한 핸들 서버 다중화 방식 연구결과를 설명하고 마지막으로 5 절에서는 결론을 맺는다.

2. Orbix + Isis / Electra

현재 CORBA는 점 대 점 커뮤니케이션 패러다임을 기본으로 하고 있기 때문에 신뢰성 있는 분산 어플리케이션의 구현에 적합하지 않다. 따라서, 신뢰성 있는 분산 어플리케이션을 CORBA 기반 환경에서 구현하기 위해서는 기존 CORBA의 확장이 필요하다. 기존의 ORB에 앞서 말한 세가지 조건을 모두 고려하여 부가적인 기능을 추가 시킨 새로운 형태의 ORB에 관한 연구가 Orbix + Isis 와 Electra 이다.

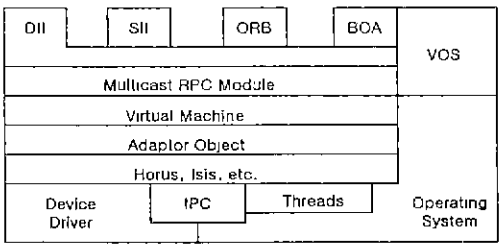
• Orbix + Isis

Orbix 는 표준화된 객체 지향 프로그래밍 환경, 인터페이스 정의에 관한 초록등의 CORBA 환경에서의 이점을 제공하며, Isis 는 가상 모델, 요청의 순서성, 상태 전달, 뷰 관리, 프로세

스 그룹등과 같은 이점을 제공하므로 Orbix + Isis 시스템 디자인은 많은 강점을 가지고 있다. Orbix + Isis 에서 서비스 객체들은 두개의 기본 클래스 – Active Replica, Event Stream – 로 부터 상속 받아 신뢰성을 제공할 수 있다. Active Replica 클래스에는 모니터링 기능과 상태 전달 서비스가 정의되어 있다. 또한, Active Replica 클래스는 기본적으로 세가지 형태 – Multicast, Client's Choice, Coordinator/Cohort – 의 커뮤니케이션 스타일[2]을 제공한다.

Event Stream 클래스는 객체 그룹들이 공표/신청(publication/subscription) 패러다임을 사용하여 비동기화 요청을 제공한다. 사용자는 Event Stream 에게 비동기화 event 를 보내면, Event Stream 은 자신의 Event log 에 모든 사용자로부터 받은 event 들을 기록해 놓고 event 를 받고자 신청한 객체 즉, Event Group 에 참여한 Event Receiver 들에게만 event 들을 전달한다.

• Electra



[그림 1] Electra 구조

Electra 는 그림 1 과 같이 계층화된 구조를 가지고 있다. DII, SII, ORB, BOA 가 핵심(core) ORB 가 되어 비동기화 RPC 를 제공하며, Multicast RPC Module 구현은 유연성과 이동성을 위해 Virtual Machine 이라는 툴킷에 독립적이고 기본적인 모듈을 구성해야만 한다. 어플리케이션이 실제 하부 구조에서 운용되고 있는 운영체제와 작용하기 위해서는 VOS(Virtual Operating System)계층을 통해야만 하고 VOS 는 파일이나 메모리 관리 오퍼레이션들을 제공한다. 툴킷에 종속적인 Adaptor Object 는 Virtual Machine 인터페이스를 하부 구조에서 실제로 동작중인 툴킷에서 제공하는 API 로 사상 시켜준다 새로운 툴킷에 Electra 를 동작시키기 위해서는 툴킷에 종속적인 Adaptor Object 를 개발해야만 한다

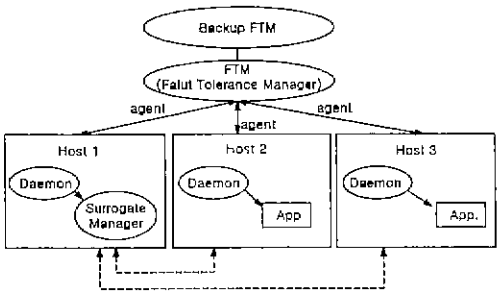
Electra 는 장애 감내와 그룹 커뮤니케이션 기능을 추가 시키기 위해 두개의 새로운 인터페이스를 정의 하였다

다음의 BOA 클래스는 그룹에 속한 서버 객체들간의 멤버쉽을 유지하기 위한 기능들을 제공하고, Environment 클래스를 통하여 동기화, 비동기화, 중간적 형태(deferred synchronous)요청과 더불어 두 가지 형태의 멀티캐스트 스타일을 제공한다.

```
Class BOA{
        ... .. ...  . .... . . ...  .. ....
        Static void create_group(Object_ptr group, const  ProtocolP
        olicy&policy=default_protocol_policy, Environment_ptr=0).
        void join(Object_ptr group, Environment_ptr=0);
        void leave(Object_ptr group, Environment=0);
        static void destroy_group(Object_ptr group, Environment=0);
        virtual get_state(AnySeq&state, Boolean&done, Environment_ptr env);
        virtual set_state(const AnySeq&state, Boolean done, Environment_ptr
        env);
        virtual view_change(const View& newView);
};
```

3. Chameleon

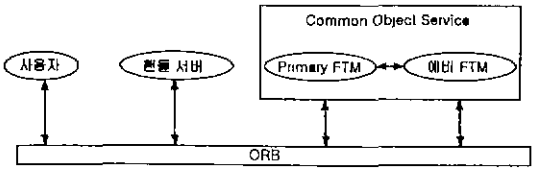
현재 CORBA 가 개방형 분산시스템 디자인에 관한 표준을 제공하고 있지만, 현재 많은 시스템들이 아직 CORBA 에 기반하지 않은 환경에서 동작하고 있다. 이러한 시스템들 상에서 신뢰성 있는 분산 어플리케이션 구현을 제공하기 위해서는 서비스 객체들을 관리하고 감시하는 새로운 형태의 연구 필요성에 의해 진행된 것이 Chameleon 이다.



[그림 2] Chameleon 의 전체 구조

위의 그림에서 보는 바와 같이 Chameleon 은 여러 에이전트들로 구성되어 있다. Chameleon 환경에 참여한 각 호스트들은 초기화 상태를 마친 후, FTM 으로부터 네트워크를 통하여 에이전트를 받아 신뢰성 있는 분산 어플리케이션 환경을 갖추게 된다. 각 호스트 상에 존재하는 어플리케이션은 에이전트에 의해 모니터링 되고, 호스트 데몬은 에이전트나 Surrogate Manager 를 모니터링 한다. 어플리케이션에 장애가 발생한 경우 에이전트는 이 사실을 Surrogate Manager 에게 알리고, Surrogate Manager 에게 장애가 발생하면 호스트 데몬은 이 사실을 FTM 에게 알려 계속 해서 서비스를 제공할 수 있도록 한다.

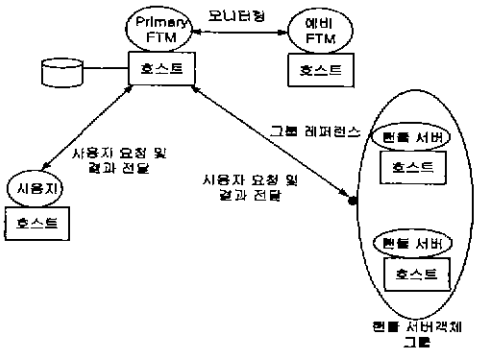
4. 장애 감내형 다중화 서버 구조



[그림 3] 객체 서비스 형태의 장애 감내형 다중화 서버

위의 그림은 장애 감내형 다중화 서버가 순수한 CORBA 환경 위에서 객체 서비스 형태로 서비스를 제공하는 것을 간략히 보여주고 있다.

Electra 와 Orbix + Isis 그리고 Chameleon 은 신뢰성 있는 분산 어플리케이션의 구현을 제공할 수 있지만, Chameleon 의 경우 신뢰성을 제공하기 위해 너무 많은 시스템 자원과 코드가 필요하여 오버헤드가 발생하고, 복잡하다는 단점이 있다. 또한 Orbix + Isis 와 Electra 는 기존의 ORB 에 수정을 가하여 변형된 ORB 내에서만 신뢰성을 제공할 수 있고, 순수한 CORBA 환경 위에서는 호환성에 문제가 있다.



[그림 4] 장애 감내형 다중화 서버 구조

그림 4 은 디지털 도서관의 문서 객체들의 Naming 서비스를 제공하는 핸들 서버를 위한 장애 감내형 다중화 서버 구조를 나타낸 것이다. 핸들 서버를 포함한 디지털 도서관의 각 객체들은 ILU C++ runtime library[4] 위에 구현 되었다.

핸들 서비스를 제공하는 다중화된 핸들 서버 객체들은 Naming Service 를 통하여 FTM 과 연결을 설정한 후, 한 그룹의 멤버로 관리 된다

FTM 은 그룹을 관리하기 위하여 아래와 같은 인터페이스를 제공하며, 그룹 멤버들의 객체 레퍼런스를 테이블 형태로 관리하고 그룹 생성, 삭제, 참여, 탈퇴 등에 관한 기능을 제공하면서 주기적으로 그룹의 모든 멤버들을 모니터링 한다. 또한 사용자 서비스 요청에 대한 로그를 가지고 있어, 주기적으로 예비 서버 객체에게 전달하여 Primary 서버와 같은 내부 상태를 유지하게 한다. 예비 FTM 은 주기적으로 Primary FTM 을 모니터링 하면서 사용자의 요청에 관한 로그를 백업 받아 항상 최근의 로그를 유지한다. 위의 모든 사항들은 사용자에게 투명하게 제공되며, 사용자는 핸들 서버 그룹이 아닌 단일 핸들 서버 객체로부터 서비스를 받는 것처럼 느낀다.

```
INTERFACE FTM
        TYPE FTM OBJECT COLLECTIBLE
        METHODS
        Create-Group(ServerInfo : Istring) : Result,
        Join_Group(ServerInfo : Istring) : Result,
        Leave-Group(ServerInfo : Istring) : Result,
        Destroy-Group(ServerInfo : Istring) : Result,
        My-Ranking(ServerInfo : Istring) . Ranking,
END;
```

5. 결론

Orbix + Isis 와 Electra 그리고 Chameleon 은 신뢰성 있는 분산 어플리케이션의 구현을 제공하는데 있어서 효과적이지만, 몇 가지 단점을 가지고 있는 시스템들이다. 따라서, 장애 감내형 다중화 서버 연구에서는 각 시스템들의 장/단점을 파악하여 모델을 성립하였으며, ILU(Inter-Language Unification)에서 제공하는 인터페이스를 정의하고, 구현 중에 있다.

현재 OMG 에서는 본 연구에서와 같이, 장애 감내형 객체 서비스를 지원하는 COS 서비스의 정의를 계획하고 있다. 앞으로 우리의 장애 감내형 서비스 기술이 OMG 의 표준화에 반영되도록 추진하고자 한다.

참고문헌

[1] OMG, "Service Component Specification," March 1995

[2] Silvano Maffeis, "Building Reliable DistributedSystem with CORBA," April 1997

[3] R.K.Iyer, Z.Kalbarczyk, S.Bagchi, "Chameleon . A Software Infrastructure and Testbed for Reliable High-Speed Networked Computing," July 1997

[4] 충남대학교 소프트웨어 연구센터, "디지털 도서관을 위한 분산지원환경 및 통신프로토콜에 관한 연구," March 1998