X/Open DTP 기반 증권사 트레이딩 시스템에서의 XA/Non-XA 인터페이스 방법

(XA and Non-XA Interface Methodology of an X/Open DTP-based Trading System in Finance Industry)

김 용 태 † 변 창 우 †† 박 석 †††
(Yong Tae Kim) (Chang-Woo Byun) (Seog Park)

요 약 금융산업에서 증권사의 주식 트레이딩 시스템은 단 1분의 장애에도 손실이 매우 큰 응용 시스템이다. 특히, 트레이딩 시스템 구축 환경이 메인 프레임 환경에서 클라이언트/서버 환경으로 전환됨으로써 추가적으로 안정성이 시스템의 최우선 목표로 인식되고 있다. 이와 같은 인식 하에 일반적으로 IT환경에서 제시하고 있는 가이드라인에 의한 시스템 구축이 당연시 되고 있지만, 트레이딩 시스템은 업무의 특성상 정규화된 표준 가이드라인을 따를 경우 신뢰도를 보장할 수 없음으로 비정규화된 방법으로 해당 시스템을 구축하는 것이 현재 실정이다.

본 논문은 TP 모니터 계열의 미들웨어를 통한 3-Tier 클라이언트/서버 환경에서 X/Open DTP 모델에서 정규화된 표준 가이드라인인 XA 인터페이스 기반 시스템과 비정규화된 방법인 Non-XA 인터페이스 기반 시스템을 증권사 트레이딩 시스템 환경 하에 구축하여 비정규화 방법의 신뢰성을 증명한다. 그 신뢰성 증명의 방법으로는 시스템 실패를 가정한 극복(take-over) 테스트 시 주문 데이타의 에러와 이의 복구 방안을 XA와 Non-XA 인터페이스 두 경우를 대상으로 수행한다.

키워드 : 미들웨어, TP-모니터, XA 인터페이스, Non-XA 인터페이스, 주식 트레이딩

Abstract In the field of finance, Trading System of Securities is a very vulnerable application when it faces any small problems just for one minute. Since Trading System changes its environment from mainframes to client/server, its safety becomes the most important factor. Even though most IT systems are configured by general guidelines currently, Trading System is an exception that it is configured by specific and rather ad hoc guidelines in order to ensure its safe management.

In this thesis, I will prove the validity of specific and ad hoc configuration in the environment of Trading Systems where I use both XA interface system and Non-XA interface to configure its system based on 3-Tier Client/Server computing environment through middleware, TP-Monitor, in the X/Open DTP Model. In order to validate the Trading System, I will compare and analyze the error of data of an order and ability to restore using both XA and Non-XA interfaces while testing take-over scenario on the assumption of the system's failure.

Key words : Middleware, TP-Monitor, XA Interface, Non-XA Interface, Trading System of Security

1. 서 론

최근 인터넷의 폭발적인 확산에 따라 모든 시스템이 웹을 기반으로 신규 구축되거나, 또는 전환이 진행 중이다. 그러나, 현실은 인터넷 기술들의 한계로 실시간 트랜잭션 처리시 응용 시스템이 데이타베이스와 직접 연동되어 해당 트랜잭션을 처리할 경우 여전히 많은 제약이 따르고 있다. 물론, 인터넷 기술의 비약적인 발전으로, 불과 2~3년 전만 해도 단순히 HTML문서의 하이퍼링크를 통한 조회 수준에서 현재는 실시간으로 웹 응용 시스템이 데이타베이스와 연결되어 해당 트랜잭션의 처리가 가능함에 따라 인터넷에서도 데이타베이스 연동 서비스가 상당 부분 가능하게 되었다[1]. 특히, 시스템

· 본 연구는 서강대학교 산업기술연구소의 지원으로 수행되었음
† 비 회 원 : SKC&C 금융사업본부 과장
bmkyt@skcc.com
†† 학생회원 : 서강대학교 컴퓨터학과
chang@dblab.sogang.ac.kr
††† 종신회원 : 서강대학교 컴퓨터학과 교수
spark@dblab.sogang.ac.kr
논문접수 : 2002년 6월 12일
심사완료 : 2003년 7월 21일

하부 구조 역시 효율성과 안정성의 최적화에 목표를 두고 2-Tier형태의 클라이언트/서버 구조에서 규모의 대형화됨에 따라 분산처리시스템 상의 네트웍 트래픽 증가와 이에 따른 트랜잭션 처리 속도의 저하라는 문제의 해결책으로 TP-모니터 계열의 미들웨어를 포함한 3-Tier형태의 클라이언트/미들웨어/서버 구조가 등장하게 되었다[2,3,4]. 이와 같은 미들웨어 환경에서는 제시하는 가이드라인을 따르는 것이 당연시 되고 있지만, 실제 구축 시에는 시스템의 성능이나 안정성만이 아닌 또 다른 요소와의 상쇄현상(trade off) 때문에 변칙적인 구축 방법을 따르는 것이 현 실정이다.

본 논문은 정규화된 구축 방법과 비정규화된 구축 방법을 비교/분석하여 비정규화된 구축 방법의 신뢰성을 보이고, 객관적인 비정규화 방법을 제시하는데 목표를 두고 있다. 즉, 증권사 트레이딩 시스템의 구축 사례를 예로 들어, 미들웨어 환경 하에 X/Open에서 DTP 모델의 표준 구축 가이드라인으로 제시하고 있는 정규화된 구축 방법인 XA 인터페이스와 비정규화된 구축 방법인 Non-XA 인터페이스, 각각에 대해 구축을 해보고 파일럿 시스템의 테스트를 XA와 Non-XA 인터페이스 두 경우에 대하여 실시하여 그 결과치를 비교/분석하고 비정규화된 구축 방법의 객관적인 신뢰성 및 타당성을 제시한다.

본 논문의 구성은 다음과 같다. 2장에서는 기본 인프라 환경인 미들웨어에 대한 간단한 개념 및 전역 트랜잭션의 일반적인 2 단계 완료 개념을 설명하고, XA/Non-XA 인터페이스에 대한 차이점을 기술한다. 3장에서는 본 논문의 적용 환경인 증권사 트레이딩 시스템에 대해 언급하고 이 환경에서의 정규화된 방법인 XA 기반 트레이딩 시스템의 문제점을 제시하고, 비정규화된 방법인 Non-XA 기반 트레이딩 시스템에 대해 설명한다. 4장에서는 극복(take-over) 테스트를 통해 Non-XA 기반 트레이딩 시스템의 신뢰성 및 타당성을 설명하며 5장에서 결론을 맺는다.

2. 미들웨어의 기본 개념

2.1 미들웨어란

미들웨어란 서로 다른 통신 프로토콜, 시스템 아키텍처, 운영체제, 데이타베이스와 다양한 응용 시스템을 지원하기 위하여 하드웨어에 독립적으로 네트웍을 연결하여 주는 소프트웨어를 의미한다. 그리고, 미들웨어의 목표는 "Any-to-Any Operability"로서 파일 교환 및 공유, 원격 프로시져 호출 등의 다양한 방법을 활용하여 응용 프로그램 사이에 호환성을 유지하여 줌으로써 응용 프로그램이 다양한 시스템 환경에서 동작할 수 있도록 도와주는 것이며 그 기능은 개발자의 입장에서 보면 사용하기 쉽고, 관리하기 편한 중간층을 제공함으로써 사용자가 원하는 응용 프로그램을 빠르고 안전하게 개발할 수 있도록 도와주는 것이다. 이는 통신을 위해 쉽고 간단한 함수들을 제공해 개발 생산성을 높여준다는 의미이며, 안전하다는 것은 기업의 응용 시스템이 문제없이 돌아갈 수 있도록 지원한다는 것이다. 또 다른 미들웨어의 기능으로는 관리 및 감시 기능을 들 수 있다. 이는, 모든 기업의 응용 시스템은 운영 중 각종 문제와 이에 따른 동작 상태를 상시 감시 및 관리함으로써 에러를 사전에 예방할 수 있는 기능을 제공한다는 것이다[5,6].

한편, 미들웨어의 종류에는 데이타베이스 미들웨어, 원격 프로시져 호출(Remote Procedure Call, 이하 RPC), 메시지 중심 미들웨어(Message-Oriented Middleware, 이하 MOM), 트랜잭션 처리(Transaction Processing, 이하 TP) 모니터, 객체 요구 브로커(Object Request Brokers, 이하 ORB) 미들웨어 등의 여러 종류가 있으나 본 논문에서는 TP-모니터 계열의 미들웨어인 BEA사의 텍시도에 국한하여 본 논문을 진행한다[7,8,9,10].

TP 모니터 계열인 텍시도 6.5는 클라이언트에서 전송된 데이타를 접수하여 전역 트랜잭션을 관리하는 트랜잭션 관리 서버 부분과, 요청된 서비스 내에서 데이타베이스에 대해서 실제 비즈니스 로직의 수행을 요청하는 자원 관리기, 이렇게 두 부분으로 구성되어 있다. 이 미들웨어 모듈에서 트랜잭션의 처리 방법에 따라서 XA와 Non-XA 인터페이스의 두 가지 방법이 사용되고 있다.

2.2 2-단계 완료(2-Phase Commit)의 기본 개념

먼저, 텍시도를 바탕으로 전역 트랜잭션에 관하여 간단히 정의한 후 2-단계 완료에 관하여 기술한다.

2.2.1 전역 트랜잭션(Global Transaction)

X/OPEN DTP 모델에서 트랜잭션 관리기(Transaction Manager, 이하 TM)는 트랜잭션 식별자(Transaction Identifier, 이하 XID)를 사용하여 그림 1에서 볼 수 있는 것처럼 하나의 전역 트랜잭션 내에서는 모두 동일한 전역 트랜잭션 식별자(Global Transaction Identifier, 이하GTRID)를 사용한다. 물론, TM에서 사용되는 XID는 자원 관리기(Resource Manager, 이하 RM)가 제공하는 XA 인터페이스에서 사용하는 XID를 지칭하는 것이다.

또한, 하나의 그룹 내에서 발생된 모든 작업은 동일한 트랜잭션 부분(Transaction Branch) 안에 포함된다. 이들은 동일한 자원 관리기를 사용함으로, 모두 동일한 XID를 사용하고, 모든 트랜잭션 부분은 루트 노드 바로 아래 노드에 위치하여 2-단계 트리를 형성한다. 이를 특히 단일 도메인이라 하고 이와 같은 전체적인 개념을

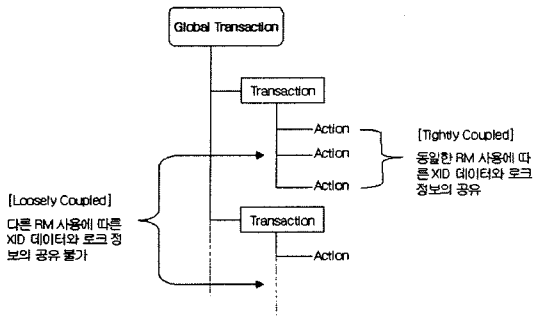

그림 1 X/OPEN DTP 모델에서의 Global Transaction 구성도

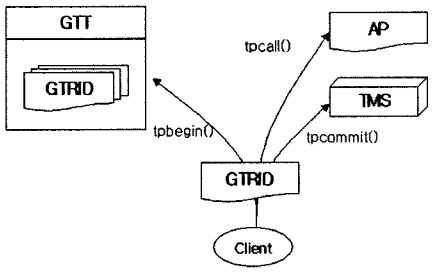

그림 2 Global Transaction 생성도

전역 트랜잭션이라 한다[11].

2.2.2 2-단계 완료(2-Phase Commit)

턱시도는 자원과 트랜잭션을 관리하는 부분이 분리되어 있다. 즉 응용 프로그램이 자원 관리기에서 제공하는 내장 SQL(Embedded SQL)과 같은 인터페이스를 사용하여 필요로 하는 작업을 수행한다면, 이와 관련된 트랜잭션을 처리하기 위한 전역 트랜잭션은 트랜잭션 관리 서버가 처리하도록 구성되어 있다. 이 자원관리기와 트랜잭션 관리 서버 사이의 전역 트랜잭션에 대한 정보 교류는 GTRID를 매개로 하여 이루어지며, 이 GTRID는 클라이언트가 전역 트랜잭션의 시작을 요청할 때 부여 되어 전역 트랜잭션 테이블(이하, GTT라 함)에 등록된다. 그 처리 과정을 단계별로 정리 해보면 다음과 같다.

첫째, 전역 트랜잭션 생성 단계

그림 2에서 전역 트랜잭션 생성에 대한 설명을 하고 있다. 클라이언트가 tpbegin()을 통하여 전역 트랜잭션의 생성을 요청하면 턱시도는 이에 고유한 GTRID를 할당하고, 이 전역 트랜잭션의 GTRID를 GTT에 등록한다. 이후 클라이언트가 tpcall()을 사용하여 필요로 하는 서비스를 호출하면, 턱시도는 GTRID를 사용하여 클라이언트가 요청한 서비스의 정보를 GTT에 기록한다.

둘째, XA 세션 처리 단계

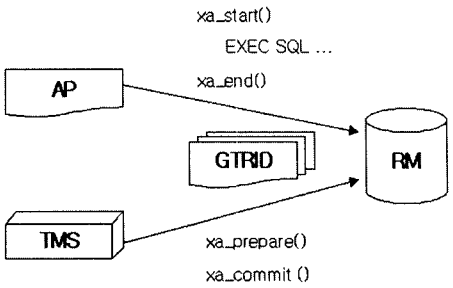

그림 3 XA Session 생성도

그림 3은 XA 세션 생성에 대한 설명을 하고 있다. tpcall()시 서비스로 요청이 들어가기 직전에 턱시도는 xa_start()을 통하여, 데이타베이스에 XA 세션의 시작을 알린다. 서비스는 필요로 하는 작업을 수행하고 (EXEC SQL …), 클라이언트에 tpreturn()를 통하여 처리결과를 반환하면, 턱시도는 xa_end()를 호출함으로써 데이타베이스에게 XA 세션의 종료를 알린다. 이때 서비스가 자원 관리기에 요청한 작업결과는 자원 관리기의 로그에 남게 되고, 이후 xa_prepare(), xa_commit()를 통하여, 트랜잭션 처리 요청이 자원 관리기에 전달되면 로그를 바탕으로 트랜잭션을 처리한다.

이제 클라이언트가 기동한 전역 트랜잭션과 이를 처리하는 서버의 해당 서비스 상의 프로세스가 분리되어 있으므로, 서버는 또 다른 서비스 요청을 즉시 실행할 수 있게 된다. 이와 같이 응용 프로그램을 전역 트랜잭션으로부터 분리시킴으로써 높은 가용성을 달성할 수 있다. 이는 다음 전역 트랜잭션이 이전 응용 프로그램이 종료될 때까지 기다릴 필요가 없음을 의미하는 것이며, 필요한 서버의 갯수를 최소로 유지할 수 있음을 의미한다.

셋째, 트랜잭션 완료 단계

서비스 호출이 정상적으로 반환되면 tpcommit()을 통하여 전역 트랜잭션의 종료를 요청한다. 턱시도는 GTRID를 이용하여 해당 전역 트랜잭션의 관련된 모든 그룹을 GTT에서 알아내고, 이 중 하나를 조정자로 지정하여 전역 트랜잭션의 처리를 요청한다. 조정자는 자기 자신과 모든 참여자에게 xa_prepare()를 호출하도록 지시함으로써 첫번째 단계를 시작한다. 트랜잭션 관리 서버들은 xa_prepare()를 호출하고 그 결과를 반환한다. 모든 트랜잭션 관리 서버로부터 정상적으로 호출이 반환되면 조정자는 트랜잭션 로그에 첫번째 단계 완료에 대한 로그를 남김으로써 과정을 종료한다.

이후 자기 자신과 모든 참여자에게 xa_commit()을 호출하도록 지시함으로써 두번째 단계를 시작한다. 트랜잭션 관리 서버들은 xa_commit()를 호출하고 그 결과

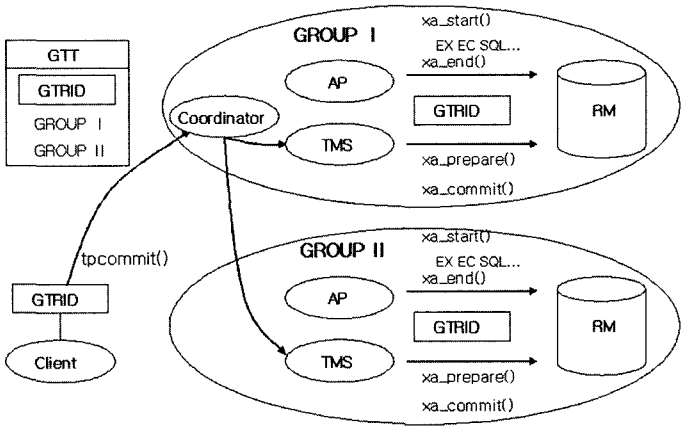

그림 4 전역 트랜잭션의 2-단계 완료 처리도

를 조정자에게 반환한다. 모든 트랜잭션 관리 서버로부터 정상적으로 호출이 반환되면 조정자는 트랜잭션 로그의 로그를 지우고 클라이언트에게 전역 트랜잭션 처리 완료를 반환함으로써 2-단계 완료를 종료한다. 만약, 전역 트랜잭션의 2-단계 완료 처리 중에 비정상적인 상태가 발생하면 트랜잭션 로그의 로그와 자원 관리기의 로그를 이용하여 복구가 진행된다. 이들에 대한 요약된 상황이 그림 4에서 설명되어 지고 있다.

물론, 이와 같은 전역 트랜잭션의 2-단계 완료 처리 과정은 XA 인터페이스에서는 자동으로 처리되며, Non-XA 인터페이스에서는 위와 같은 일련의 과정을 서버 간의 서비스 내에 프로그래머가 직접 처리를 해야 하는 것으로서, 주로 첫번째 단계에서 완료를 처리하고, 그 결과를 바로 클라이언트에 반환한다[12,13].

2.3 XA/Non-XA 인터페이스의 차이점

이번 절에서는 TP 모니터 중 Unix 기반의 Open 시스템에서 텍시도를 기반으로 하는 미들웨어와 오라클 데이타베이스 관리시스템을 기반으로 전역 트랜잭션을 처리할 때 정규화된 방법인 XA와 비정규화된 방법인 Non-XA 방식에 대한 개념을 설명하고 차이점을 기술한다. XA 인터페이스를 사용하는 서버의 서비스 응용은 텍시도에서 제공하는 ATMI(Application Transaction Manager Interface) 함수를 사용하여 데이타베이스 관리시스템에서 트랜잭션을 처리할 수 있다. 즉, 응용이 요청한 트랜잭션은 ATMI 함수를 거쳐 텍시도에 전달되고, 이는 다시 XA 인터페이스를 사용하여 데이타베이스 관리시스템에 전달됨으로써 원하는 트랜잭션이 처리된다. 한편, Non-XA 인터페이스를 사용하는 서버의 서비스 응용은 데이타베이스 관리시스템에서 트랜잭션을 처리할 때 ATMI 함수를 사용하지 않고, 직접 내장 SQL를 사용하여 처리하게 된다.

이와 같은 두 가지 방법인 갖는 장/단점은 다음과 같다.

첫째, XA 인터페이스의 장/단점

장점 : 텍시도의 전역 트랜잭션을 활용할 수 있음으로, 한 클라이언트가 여러 개서버의 서비스를 호출할 수 있고, 이를 한 트랜잭션 단위로 묶을 수 있다. 즉, 서비스들을 세분화 시킬 수 있고 이를 통하여 성능향상을 가져올 수 있다.

단점 : 텍시도 프로세스와 데이타베이스 관리시스템 간에 통신이 한번 더 발생하므로 오버헤드를 초래할 수 있다.

둘째, Non-XA 인터페이스의 장/단점

장점 : 규모가 작은 트랜잭션의 경우 서버의 서비스에서 직접 SQL 완료를 수행함으로써 트랜잭션이 좀더 빨리 처리될 수 있다.

단점 : 텍시도의 전역 트랜잭션을 활용할 수 없음으로, 트랜잭션 단위별로 서비스의 소스 프로그램을 작성하여야 하고, 이는 서버의 프로세스가 커질 수 있고, 이에 따른 성능저하를 가져올 수 있다.

이와 같은 단점을 갖고 있는 Non-XA 인터페이스 방식을 본 논문에서 선택한 이유는 증권사 트레이딩 시스템에서 일어나는 전역 트랜잭션은 트랜잭션의 규모가 작기 때문에 트랜잭션 단위별로 처리할 수 있는 소스 프로그램을 쉽게 작성할 수 있어 그리 문제가 되지 않으며, 트레이딩 시스템에서 소단위 트랜잭션의 빈번한 발생을 바라보는 관점은 서버의 프로세스를 점차적으로 증가시킬 수 있어 문제가 된다는 측면보다는 빨리 처리할 수 있는 방법은 무엇인가에 대한 관점으로 바라보기 때문이다.

3. 증권사 트레이딩 시스템

3.1 트레이딩 시스템 소개

그림 5에서 보듯이, 최초 고객의 주문 데이타는 PC의 (1)클라이언트 화면을 통하여 입력되며, 이는 네트워크를 경유하여, (2)턱시도로 그 데이타가 전달된다. 또한, (3)~(7)까지의 과정은 서버에 있는 서비스 프로그램에서 하나의 트랜잭션으로 처리되는 한 개의 응용 프로그램이다. 또한, (7)에서는 대외 전송을 하기 위하여 전문 트랜잭션 처리과정에서 턱시도의 자원 관리기 통제 밖에 놓이는 부분이다. 즉, 현재의 증권거래소와 같은 대외기관과의 전문 송/수신 시스템 하에서는 위의 문제는 결코 회피할 수 없는 문제이며, 이와 같은 대외기관과의 접속환경의 문제에서 본 논문의 연구 대상인 XA와 Non-XA 인터페이스 방식의 차이에 따른 주문 트랜

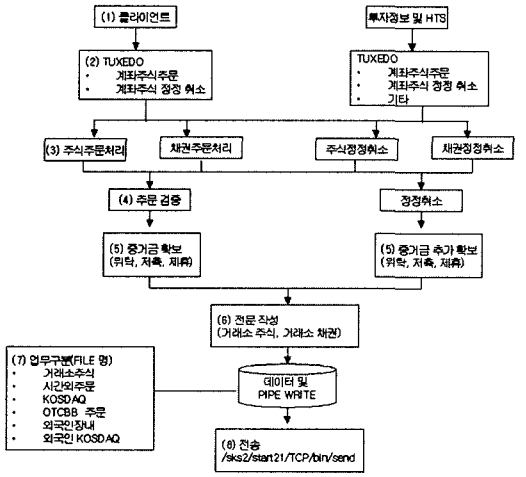

그림 5 트레이딩 시스템 업무 흐름도

잭션 처리 시 미기장 주문 데이타가 발생하는 문제가 생기게 되었다[14].

이를 시스템적인 측면에서 보면 그림 6과 같다. 이는 개발자의 클라이언트 PC와 턱시도 시스템이 기동되는 응용 프로그램 서버 그리고, 해당 데이타를 관리하는 데이타베이스 서버와 증권거래소로 주문 전문 파일의 송/수신을 담당하는 FEP(Front-End-Processor)로 구성되어 있다. 또한, XA 또는 Non-XA 인터페이스 구분은 그림 6의 응용 프로그램 서버 상의 턱시도 시스템 워크스테이션 부분의 XA 서버와 Non-XA 서버 모듈의 어느 모듈을 통과하여, 해당 업무에 대한 트랜잭션이 데이타베이스 서버에 전달되느냐에 따라서 해당 트랜잭션이 XA 또는 Non-XA 인터페이스로 처리되었는지가 결정된다.

3.2 XA 기반 트레이딩 시스템의 문제점

현재 유닉스를 기반으로 운영중인 상당수의 증권사 시스템은 턱시도를 사용하고 있으며, 대부분의 트랜잭션은 XA 인터페이스로 처리하도록 구축되어 있다. 그러나, 증권사의 핵심 응용 시스템이라 할 수 있는 트레이딩 시스템은 Non-XA 인터페이스로 해당 트랜잭션을 처리하고 있다. 물론, 개발 초기의 트레이딩 시스템은 그림 7의 (A)경로와 같은 XA 서버를 거쳐 해당 주문 트랜잭션을 데이타베이스 서버에 전달하는 XA 인터페이스 방식으로 트랜잭션을 처리하였다. 그러나, 실제 극복 테스트 과정에서 주문 데이타를 데이타베이스 서버의 데이타베이스에 기록한 후 데이타베이스 서버에서 완료를 처리하는 시점과 FEP에 발송되는 송신 전문 파일이 생성되는 시점 사이의 매우 미세한 시차가 원인으

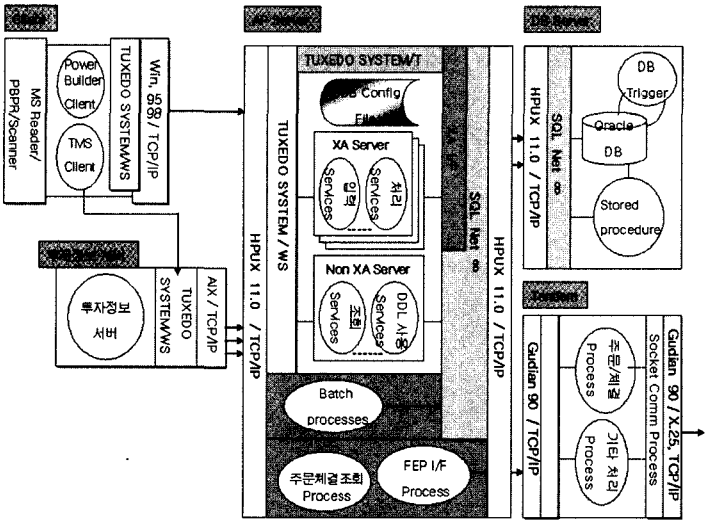

그림 6 시스템 고려 사항

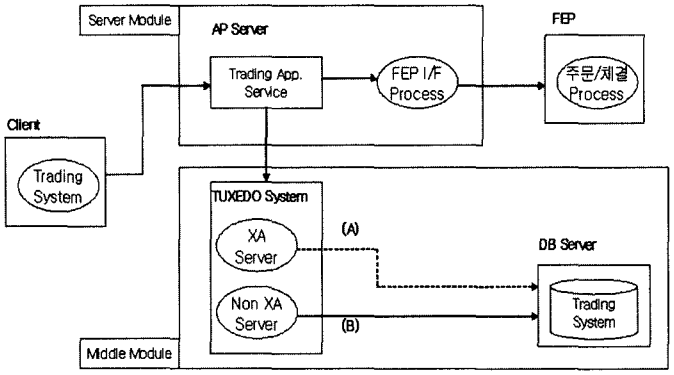

그림 7 극복 테스트 시스템 고려사항

로 의심되는 에러가 발생하였다.

30명의 테스트 인원이 Rush라는 테스트 프로그램을 이용하여 순간적으로 시스템에 최대 부하를 줌으로써 발생되는 시스템 실패에 따른 시스템의 극복 테스트로서, 실제 시스템 운영 상황에서는 99.9% 발생할 수 없는 상황이나, 시스템의 최대 부하와 돌발상황에 따른 시스템 실패란 가상의 상황을 설정하여, 시스템의 극복 테스트를 진행하는 과정에서, 시스템이 정상적으로 복구되어 수행되었음에도 불구하고, 데이타베이스에 기록된 주문 데이타 건수와, 대외기관에 발송하기 위해서 FEP서버에 만들어진 발송 전문 파일의 총 갯수에 차이가 발생하였고, 이는 극복 테스트가 최대 부하치에 근접할수록 그 차이가 늘어났다. 이와 같은 에러 상황은 XA 인터페이스에서 데이타베이스 무결성을 보장하기 위하여 2-단계 완료와 같은 부수적인 확인 과정에 소요되는 오버헤드가 늘어남에 따라 데이타베이스의 미기장 건이 발생하는 문제로 결론이 지어졌다. 또한, XA 인터페이스에 기반한 트레이딩 시스템 상에서는 시스템 고장에 따른 시스템 극복 시 발생될 수 있는 트랜잭션 복구를 위해서, 현재 SAM 파일 형태로 저장되어 있는 송신 전문 파일의 데이타만으로 주문 데이타를 복구하여 데이타베이스에 기록하여야 하나, 실제 송신 전문 파일의 데이타는 매우 간략한 주문 데이타만을 가지고 있음으로 100% 데이타베이스 상의 주문 데이타를 복구할 수 없었다. 이는 증권사의 트레이딩 시스템은 기본적으로 실패가 발생할 수 없으며, 만약 시스템 실패라는 상황이 발생하여 트랜잭션 처리에 에러가 발생하여도, 반드시 100% 복구가 보장되어야 한다는 대전제에 위배됨으로 XA 인터페이스에 기반한 구축방안은 수용할 수 없었다.

3.3 Non-XA 기반 트레이딩 시스템 분석

트레이딩 시스템의 구축시 프로그램 내에서 직접 완료를 처리하는 Non-XA 인터페이스 방식을 채택하여, XA 인터페이스의 2-단계 완료와 같은 부수적인 처리 과정을 수행하지 않고, 해당 주문 프로그램내에서 직접 완료를 처리한 후 FEP로 발송되는 전문 파일을 생성할 경우, 3.2 XA 기반 트레이딩 시스템의 문제를 해결할 수 있었다. 즉, 트레이딩 시스템이 최대 부하치에 근접함에 따라 시스템 고장시 극복 수행과 이에 따른 주문 데이타의 데이타베이스 내 미기장건 발생 문제를 해결하였음을 의미한다.

그러나, 그림 7의 (B)경로와 같은 Non-XA 인터페이스를 이용한, 직접 완료(Direct Commit)를 이용하여 주문 트랜잭션을 처리하여도, 여전히 시스템의 실패에 따른 극복 시에는 XA-인터페이스와 반대로 데이타베이스 내에는 주문 데이타가 기록되었으나, FEP에 전송되는 전문 파일이 생성되지 않을 수 있다는 문제점이 제기되었고, 또한 실제 테스트 시 이와 같은 에러 상황이 발생하였다. 그러나, 그 건수가 테스트상의 오류인지 또는 실제 Non-XA 인터페이스의 로직 상의 문제인지를 구별할 수 없을 만큼 그 횟수는 무시할 수 있을 정도의 수준이였으며, 또한, 해당되는 문제가 발생하더라도, 실제 데이타베이스에 있는 주문 데이타를 이용하여 FEP 발송 전문 파일을 100% 복구할 수 있었다. 즉, Non-XA 인터페이스에 기반한 트레이딩 시스템을 구축함으로써, 증권사의 트레이딩 시스템의 실패 시 100% 복구라는 대전제를 만족시킬 수 있었다.

4. 극복 시나리오를 통한 XA/Non-XA 인터페이스 분석

4.1 극복 시나리오 환경

현재 클라이언트/서버의 개방 환경에 구축되어 있는 증권사의 트레이딩 시스템은 대부분 3-Tier를 기반으로 구축되어 있다. 이는, 서론에서도 논의한 바와 같이 PC에 설치되어서 고객의 주문을 입력받는 클라이언트 모듈과 이를 해당하는 비즈니스 로직에 따라 주문데이타를 처리하여 데이타베이스에 기록하는 서버 모듈, 그리

고, 본 논문의 연구 대상인 클라이언트와 서버 사이에서 트랜잭션 처리를 관장하는 미들웨어 모듈로 구성되어 있으며, 또한 극복 테스트 파일럿 시스템의 구성요소도 당연히 이와 같은 3부분으로 구성되어 있다. 따라서, 각각의 구성요소에 대해서 그 역할을 간단히 기술함으로써 파일럿 시스템의 환경을 설명한다.

4.1.1 클라이언트 모듈

그림 7에서 PC에 설치되는 클라이언트 프로그램으로서, 사전에 미리 SQL문장에 의해 생성되어 있는 테스트 데이타를 입력 데이타로 받아들여서 마치 고객이 객장에서 수백 번의 주문을 연속적으로 입력하는 것처럼 동작하게 작성된 파워빌더로 작성된 응용 프로그램이다. 또한, 자동으로 서버의 주문 관련 서비스를 호출함으로써 주문 데이타를 네트워크를 통하여 서버 모듈에 발송하는 역할까지만을 수행한다.

클라이언트 모듈은 파워빌더 6.5의 Patch 1.0 버전으로 작성되었으며, 내부 로직은 단순한 루프를 이용하여 주문 데이타의 전송만을 담당하고 있으며, 개별 주문 트랜잭션에 대한 완료 등의 일련의 트랜잭션 처리와 관련된 기능은 존재하지 않는다. 그러나, 소스상으로는 통신 설정과 턱시도 세션 초기화와 같은 미들웨어의 기능을 턱시도 라이브러리 함수를 사용하여 수행하나, 여전히 미들웨어 모듈의 제어를 받는다.

4.1.2 서버 모듈

그림 7에서 응용 프로그램 서버로 명명되어 있는 클라이언트/서버의 서버 부분으로서, 클라이언트 모듈의 PC에서 네트워크를 경유하여 전달된 주문 데이타의 처리 요청에 따라 해당 비즈니스 로직을 수행한 후 트랜잭션 처리 결과값을 생성한다. 이 부분에서는 단순히 비즈니스 로직에 따라서 데이타의 처리 및 결과값에 대한 생성일 뿐, 여타 트랜잭션과의 연계 및 기타 작업의 종료와 같은 기능은 배제되어 있다. 그러나, 서비스 프로그램 소스를 보면 트랜잭션 처리와 관련하여 턱시도 라이브러리가 사용되고 있으나, 이는 결과값을 리턴하고 세션의 종료를 알리는 라이브러리로서, 클라이언트 모듈에서 전송된 데이타에 대한 트랜잭션 처리에 대한 제어는 여전히 미들웨어인 턱시도에 있다.

서버 모듈은 운영체제는 HP-UX11 유닉스이며, 데이타베이스MS는 오라클 8.1.6.2이다. 또한 서버 모듈의 서비스 프로그램은 ANSI C 11.0.1.20으로 작성되어 있으며, 데이타베이스 핸들링을 위하여 오라클 Pro*C/++ 8.1.6.2을 사용하여 SQL구문을 컴파일하였다.

4.1.3 미들웨어 모듈

그림 7에서 AP 서버 부분에 탑재되어 있는 모듈로서 실제 서비스 프로그램 소스 상에서는 본 극복 테스트 시스템에서와 같이 서버의 서비스를 작성하는 C 소스 내에 턱시도 라이브러리를 사용하였는지 여부에 따라 XA인터페이스 기반인지 Non-XA 인테페이스 기반인지 구분이 가능하다. 그러나, 미들웨어 모듈과 서버 모듈이 모두 동일 서비스 C 소스 내에 존재함으로 소스 내에서는 구분이 명확하지 않을 수 있고, 미들웨어 모듈의 데이타베이스 세션 초기화와 버퍼의 설정 및 데이타 로딩과 같은 역할을 수행하는 소스는 클라이언트 모듈에 위치함으로 그 범위를 정확히 구분 하기가 쉽지 않다. 즉, 트랜잭션 관리 서버와 자원 관리기 그리고, 사용자 인터페이스와 같은 관리 모듈의 차원에서는 서버 모듈과 미들웨어 모듈, 그리고 클라이언트 모듈이 명확히 구분이 되나, 실제로 비즈니스 로직을 처리하는 서비스 응용 소스 레벨에서는 데이타 처리 로직과, 통신 설정 및 세션 처리에 턱시도 라이브러리가 혼재되어 사용됨으로 해당 모듈을 명확히 구분하기가 어렵다. 그러나, 이 부분이 극복 테스트를 수행하는데 있어서 중요한 XA와 Non-XA 인터페이스를 구분하는 요소가 된다.

4.2 극복 시나리오

4.2.1 시나리오 환경

극복 테스트는 XA와 Non-XA 인터페이스 두 경우에 대하여 두 가지 방법으로 수행되었으며, 테스트시 H/W 성능은 표 1과 같다.

또한, 시스템 환경은 표 2와 같다.

테스트 참가 인원 30명이며 데이타 건수는 1인당 2,500건으로 1회 데이타 건수는 75,000건이다. 24시간으로 삼등분하여 일일당 3회 테스트를 진행하여 3일 동안 총 9회의 테스트를 가졌다. 표 3에 설명되고 있다.

극복이 발생하는 경우를 시스템 실패에 의한 재부팅인 때와 네트웍이 단절되었을 때 두 가지 시나리오를 가정한다. 시스템 실패에 의한 시스템 재부팅에 의한 극복은 일반적으로 연간 1회 미만이 발생할 정도로 그 경우의 수가 희박하고, 네트웍 선 단절에 의한 극복 역시 월 1회 미만으로 발생할 정도로 그 발생 빈도가 매우 낮다. 본 논문에서 이렇게 발생 빈도가 낮으면서도 극복

표 1 Hardware System 성능

항 목	성 능	항 목	성 능	항 목	성 능
CPU 성능	552Mhz	주메모리	12G Byte	서버갯수	20개
CPU 갯수	8개	초당Tr건수	100 Tr/sec		

* 초당 Tr건수: 초당 처리 트랜잭션의 건수. * 서버 개수: 턱시도의 ubb.dev에 설정된 서버 갯수

표 2 시스템 환경

항 목	Version	항 목	Version	항 목	Version
OS	HP-UX11	오라클	Ver8.1.6.2	PB	Ver6.5 패치1.0
턱시도	Ver6.5	ANSI C	Ver11.0.1.20		

표 3 테스트 데이타 내역 [9회테스트/테스트 방법]

참여 인원	30 명	데이타 건수	2,500건/1인
총 발생 데이타 건수	2,500건 * 30명 = 75,000 건		
총 테스트 횟수	3회(일) * 3일 * 2 Way= 18회		

표 4 System Rebooting 시 데이타베이스와 SAM 파일의 갯수 차이

	1회	2회	3회	4회	5회	6회	7회	8회	9회	평균
XA	2	1	1	2	1	1	1	1	1	1.2
Non-XA	1	1	0	1	0	0	1	0	0	0.4

테스트에의 두 가지 경우로 선택한 이유는 XA와 Non-XA 인터페이스 접근 방식의 장애 복구 능력과 안정성을 확인하고자 함이다. 따라서, 9회의 테스트를 실시하였으며, 본 테스트의 궁극적인 목적은 차이 건수가 다르다는 측면보다는 실제 어떤 상황의 차이가 발생하고 있는가 그리고, 이런 발생에 대한 장애 극복은 어떻게 이루어질 수 있는가에 초점을 둔다.

4.2.2 시나리오 1 : 시스템 실패에 의한 재부팅

실제 상황의 시스템 운영 시 이와 같이 시스템을 재부팅하는 경우는 시스템이 실패 상태에 빠지는 경우이거나 더 이상의 서비스를 수행하지 못하는 펜딩(Pending) 상태에 빠지는 경우 또는 부득이하게 시스템을 재부팅하는 경우가 있다. 본 논문은 시스템 실패에 대한 재부팅을 가정한다. 물론, 어떠한 경우에도 현재 주식 트레이딩에 관한 트랜잭션 정보는 증권사 또는 거래원에 이중으로 로그가 남음으로 100% 에러 트랜잭션에 대한 복구가 가능하며, 이와 같은 상황은 년간 1회 미만 정도로 발생할 경우가 매우 희박하다.

따라서, 본 테스트 중에는 시스템에 부하를 주는 방법으로는 위와 같은 시스템 펜딩(Pending)상황을 만들 수 없음으로 그림 8의 Dual System중 응용 프로그램 머신 1에서 주문 테스트를 진행하는 중에 임의의 시점에 강제적으로 응용 프로그램 머신 1에 대해 재부팅을 함으로써, 시스템을 순간적으로 응용 프로그램 머신 2로 극복시켰으며, 극복 시 데이타베이스에 저장되어 있는 주문 데이타 건수와, 또한 거래소에 전송되기 위해서 생성되는 전문 파일의 갯수를 비교하여, 그 건수의 차이를 비교/분석함으로써 XA와 Non-XA 인터페이스 방법의 차이에 따른 에러율에 대한 비교/분석을 각각의 인터페이스 방법에 따라 9회 실시 하였다.

그 결과 표 4와 같다.

4.2.3 시나리오 2 : 네트워 선의 절단

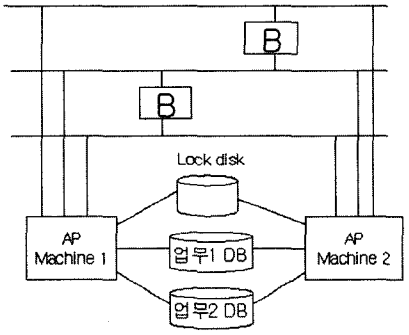

그림 8 Multi-Active Architecture

실제 상황의 시스템 운영 시 이와 같은 네트웍 선이 절단되어 장애가 발생하는 경우는 결코 없으나, 이와 유사한 상황으로는, 이중으로 설치되어 있는 스위칭 허브(Switching Hub) 중 한쪽이 다운되는 경우나, 서버의 프로토콜 에러가 발생하는 경우 그리고, 서버의 서비스 포트가 다운되는 경우에는 위와 유사한 상황의 에러 상태에 빠졌다고 볼 수 있다.

그러나, 이와 같은 네트워크상의 실패가 발생하는 경우에는 자동으로 시스템 또는 스위칭 허브에서 극복이 수행되며, 이 또한 실제 상황에서는 월 1회 미만으로 발생할 정도로 그 발생 빈도가 매우 낮다. 물론 위와 같은 에러가 발생하였을 경우, 기존의 시스템에서는 주식 트레이딩과 관련한 트랜잭션에 대해서 에러 처리가 아닌, 정상적인 데이타로서의 처리를 100% 보장하여 준다.

또한, 테스트 중에는 단순히 하드웨어 또는 네트워크에 부하를 주는 방법으로는 이와 같은 네트웍 다운 상황을 만들 수 없음으로 그림 8의 Dual System 중 응용 프로그램 머신 1에서 주문 테스트를 진행하는 중에 임의의 시점에 강제적으로 응용 프로그램 머신 1에 연결되어 있는 네트웍 선을 단선시킴으로써, 시스템을 순간

표 5 Network Line 단절시 데이타베이스와 SAM 파일의 갯수 차이

	1회	2회	3회	4회	5회	6회	7회	8회	9회	평균
XA	0	0	1	0	0	0	0	0	0	0.1
Non-XA	0	0	0	1	0	0	0	0	0	0.1

표 6 XA와 Non-XA 상의 에러 유형

	에러 유형
XA	총 12건은 데이타베이스에 저장된 주문건수가 대외기관에 발송되기 위해서 만들어진 SAM 파일보다 그 건수가 적음
Non-XA	총 5건은 대외기관에 발송되기 위해서 만들어진 SAM 파일의 갯수가 데이타베이스에 저장된 주문건수보다 그 갯수가 적음

적으로 응용 프로그램 머신 2로 극복시켰으며, 극복 시 데이타베이스에 저장되어 있는 주문 데이타 건수와 거래소에 전송되기 위해서 생성되는 전문(SAM 파일)의 갯수를 비교하여, 그 건수의 차이를 비교/분석함으로써 XA와 Non-XA 인터페이스 방법의 차이에 따른 에러율에 대한 비교/분석을 각각의 인터페이스 방법에 따라 9회 실시하였으며, 그 결과 표 5와 같다.

4.3 극복 시나리오의 에러 분석

첫번째 시스템 실패 시 재부팅과 두번째 네트웍 선의 단절인 경우에 수행된 극복 시, 데이타베이스 상의 주문 데이타 건수와 전문 SAM 파일의 갯수에 차이가 발생하였으며, 이는 XA와 Non-XA 인터페이스의 방식 차이에 따라 그 차이 건수가 다르게 발생하였으며, 또한, 에러 형태도 다르게 나타났다.

표 6과 같은 에러 유형은 해당 주문 데이타를 정상적으로 복구를 하는데 있어서도, XA 인터페이스의 경우에는 주문 데이타의 데이타베이스 내 기록이 누락된 경우로 장 중에 주문 데이타를 복구할 수 없는 상태로서, 이와 같은 경우는 장 종료 후 증권거래소에 실제 전송되어, 처리된 주문 트랜잭션과 비교함으로써 복구가 가능하였으며, 이는 고객의 주문이 실제로 미처리 될 수 있음을 의미한다. 이 사실은 증권사로서는 절대로 받아들일 수 없는 상황이다. Non-XA 인터페이스 경우, 데이타베이스에는 주문 데이타가 기록되었으나 미처 대외기관에 발송되기 위한 SAM 파일이 만들어지지 않은 상황이 발생할 수 있지만, 극복이 수행된 후 데이타베이스의 주문내역과 SAM 파일을 비교하여 극복 시에 대외 기관에 미 전송된 전문을 내부 복구 프로그램에 의하여 생성하고 재전송할 수 있다. 이런 상황은 장 중에도 100% 주문 데이타가 복구될 수 있음을 의미한다. 그러나, 전문의 복구와 재전송 사이의 시차적인 오류는 발생할 수 있다.

4.4 Non-XA 기반 트레이딩 시스템의 타당성

Non-XA인터페이스에 기반한 트레이딩 시스템 구축 방안에 대한 타당성을 제시하기 위하여, 시스템 실패를 가정하여 극복 테스트를 XA와 Non-XA 두 방법에 대하여 수행하였고, 그 결과를 분석한 결과, Non-XA 인터페이스에 기반을 두고 구축된 트레이딩 시스템의 경우 시스템 재부팅 시에 9회의 테스트 중 총4건의 데이타베이스 내 미기장 주문 데이타가 발생하였으나, 이는 평균 0.4건이 발생한 것으로써, 위와 같은 강제적인 시스템 재부팅을 강행하여야 하는 경우는 실제 시스템 운영 시 년간 1회 미만이 발생할 정도의 상황이기 때문에 평균 0.4건은 실제 시스템 운영 시에는 발생하지 않는다고 봐도 무방한 수준으로 결론내릴 수 있다.

또한, 네트웍 선 절단에 따른 극복 테스트 시에는 XA와 Non-XA 두 방법 모두가 총 9회에 걸쳐서 총 1건의 데이타베이스 내 미기장 주문 데이타가 발생하였으나, 이는 테스트 상의 오류 내지는 오차로 볼 수 있는 수준으로서 에러로서의 의미가 없다고 결론내릴 수 있다. 물론, 트레이딩 시스템의 업무 성격상 XA 인터페이스 방법으로 구축된 경우 극복 시 발생된 미기장 주문 데이타는 장 중에 100% 복구할 수 없는 반면, Non-XA 인터페이스 방법으로 구축된 경우 미기장 주문 데이타는 장 중에도 100% 복구될 수 있다는 점이 CIO의 Non-XA 인터페이스에 기반을 둔 트레이딩 시스템의 구축이라는 어려운 결정을 내릴 수 있도록 해준 면도 부정할 수는 없지만, 위와 같은 극복 테스트를 통하여 얻게 된 테스트 시나리오 자료는 그 자체로서 XA 인터페이스 방법으로 트레이딩 시스템을 구축할 경우 테스트 시나리오의 확률 만큼 에러가 발생 될 수 있음을 수치적으로 보여주는 것이다. 또한, Non-XA 인터페이스 방법으로 구축 시 막연히 문제가 해결될 수 있다는 우연적인 문제 해결 방안이 아닌 연간 0.4%의 발생 확률에 따른 실제 상황에서는 무시할 수 있는 수준의 에러율이라는 객관적인 근거 자료를 제시할 수 있게 됨으로써, 향후의 유사한 시스템 구축 시에는 추가적인 검증 작업 없이, 곧바로 활용할 수 있는 Non-XA 인터페이스에 기반을 둔 트레이딩 시스템의 구축사례를 제시할 수 있게 되었다.

5. 결 론

지금까지 본 논문의 연구를 통하여 클라이언트/서버 환경 중 유닉스를 기반으로 하는 개방 환경에서 X/Open DTP 모델의 Non-XA 인터페이스 방법이 업무 시스템을 구축하는데 있어서 기본적인 구축 방법으로 제시되고 있는 XA 인터페이스 방법의 부분적인 취약점을 보완해 줄 수 있는지 여부에 관하여 살펴보았다. 물론, 대부분의 업무 시스템은 XA 인터페이스 상에서 구축될 수 있으며, 또한, 본 논문의 연구 소재가 되었던 BEA사의 터시도와 같은 TP 모니터 계열의 미들웨어를 사용하는 가장 궁극적인 목적인 "시스템 구축의 편리성"과 데이타 무결성과 같은 "안정성의 확보", 각종 "감시 및 관리 방안의 제시"와 같은 기능을 모두 활용할 수 있다.

그러나, 실제 업무 시스템의 구축 시에는 양적인 구축보다는 질적인 구축이 더욱 중요시 될 수 있으며, 이와 같은 경향은 증권사의 주식 트레이딩 시스템과 같이 업무의 성격에 크게 좌우된다. 즉, 단 1%의 시스템 내 응용 프로그램 비중을 차지하고 있는 프로그램이지만, 이 1%의 프로그램이 필수적으로 요구하는 신뢰도가 전체 시스템의 신뢰도를 대변하는 경우가 비일비재하다. 신뢰도 확보를 위하여는 정규화된 구축 방안이 아닌 비정규화된 구축 방안이라 할 지라도 구축 방법이 명확하고 객관적인 근거 자료를 제시할 수 있다면, 충분히 신뢰성 높은 구축 방법으로 사용될 수 있음을 본 논문의 연구를 통해 설명하였다.

두 가지 시나리오를 갖고 극복 테스트를 수행하여 첫째, 시스템 실패에 의한 시스템 재부팅 시 XA 인터페이스 구축 방법이 Non-XA 인터페이스 구축 방법에 비해 데이타베이스에 누락되는 주문 데이타 건수가 훨씬 많이 발생하며, 또한 업무적인 특성상 손실된 주문 데이타에 대한 복구 또한 매우 어렵다는 것을 알 수 있었다. 둘째, 네트웍 선 단선을 통한 시나리오는 XA와 Non-XA 인터페이스 방법 모두 다 주문 데이타 손실율이 매우 낮은 수준으로 무시할 수 있는 수준이었다. 이와 같은 결과는 XA 인터페이스 구축 방법이 가장 커다란 취약점인 "다수 자원 핸들링(Multi Resource Handling)"은 상대적으로 긴 무결성 검증에 필요한 CPU 시간의 부하에서 발생되는 것이고, 2-단계 완료 역시 실제적인 원인 중 하나라고 결론을 내릴 수 있다.

끝으로, 객관적인 근거 자료의 제시를 통하여, 일반 기업체의 업무 시스템을 구축하는데 좀 더 신뢰도 높은 구축 방안을 제시할 수 있었다. 본 논문에서 제시한 방법은 모든 기업 간의 각종 업무 시스템이 네트워크를 통항 하나로 통합되어 가는 상황에서는 더욱 더 중요성이 부각될 것이라 본다. 또한, 여러 업체의 다양한 미들웨어 성능 평가 작업 시 해당 기능의 지원 내지는 보완 정도에 따른 상쇄현상의 객관적 근거 자료로서 제시될 수 있을 것이다.

참 고 문 헌

[1] 강순덕, 미들웨어와 인터넷, 인솔 미디어.

[2] http://www.cs.chungnam.ac.kr

[3] http://www.byte.com/art/9504/sec11/art4.html

[4] http://kon.kyungpook.ac.kr/members/gwh/ai1.html

[5] Hector A. Duran, Gordon S. Blair, "A Resource Management Framework for Adaptive Middleware," Proceedings of the Third IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, 2000. 3.

[6] Alexey Vaysburd, "Fault Tolerance in Three-Tier Applications: Focusing on the Database Tier," Proceedings of the 1999 18th IEEE Symposium on Reliable Distributed Systems, 1999. 10, Page 21~45.

[7] Park JH, Kang SJ, Moon KD, "Middleware architecture for supporting both dynamic reconfiguration and real-time services," IEEE Transactions on Consumer Electronics, V.46 N.3, 2000. 8.

[8] Xiaolin Gui, Depei Qian, G. He, X. S. Dong, "An Object-oriented Middleware for our Metasystem on Internet," Proceedings of the 36th International Conference on Technology of Object-Oriented Languages and Systems, 2000. 10.

[9] http://www.posdata.co.kr/pos_kor/info.html

[10] http://csd.hei.co.kr/hyun/html/product/nx800/middleware.html

[11] Andrade, Juan M. (Edt)/Dwyer, Terence/Felts, Stephen/Andrade, Juan, The Tuxedo System, Addison-Wesley Pub Co (Sd).

[12] 한창민 비이에이 수석 연구원, 'Tuxedo 2PC (2-Phase Commit)', BEA사 2000년 TUXEDO 세미나.

[13] 박준 비이에이 수석 연구원, '터시도 개괄', BEA사 2000 TUXEDO 세미나.

[14] Joo Kyung-Soo, "A Design of Middleware Components for the Connection between XML and R 데이타베이스," Proceedings of the 2001 IEEE International Symposium on Industrial Electronics - Volume 3, V.3, 2001.6, Page 7~16.

김 용 태

1993년 인하대학교 전자계산 공학과(공학사). 2002년 서강대학교 정보통신대학원 정보처리학과(공학석사). 1993년 3월~1995년 10월 ㈜C.N.I. R&D 개발팀원 1995년 11월~1997년 11월 LG-CNS 기술연구부문 연구원. 1998년 9월~1999년 9월 Newave㈜ 개발실 팀장. 1999년 9월~현재 SKC&C, 금융사업본부, 금융사업1팀 과장. 관심분야는 Open환경에서의 분산처리와 이에 따른 업무 트랜잭션 처리 및 부하 관리, 데이타 모델링, 개발 방법론

변 창 우

1999년 서강대학교 컴퓨터학과 학사 2001년 서강대학교 컴퓨터학과 공학석사 2001년~현재 서강대학교 컴퓨터학과 박사과정. 관심분야는 트랜잭션 관리, 워크플로우, 웹과 데이타베이스, 접근제어

박 석

1978년 서울대학교 계산통계학과(이학사). 1980년 한국과학기술원 전산학과(공학석사). 1983년 한국과학기술원 전산학과(공학박사). 1983년 9월~현재 서강대학교 컴퓨터학과 교수. 1989년~1991년 University of Virginia 방문교수. 1996년~1998년 한국정보과학회 데이타베이스 연구회 운영위원장. 1997년 2월~현재 한국통신정보보호학회 이사. 2001년 12월 한국정보과학회 이사 2000년 4월~현재 DASFAA Steering Committee 멤버. 관심분야는 데이타베이스 보안 멀티미디어 데이타베이스, 트랜잭션 관리, 데이타웨어하우스, 웹과 데이타베이스, 워크플로우