http://dx.doi.org/10.14400/JDC.2014.12.5.1

고확장성, 고가용성을 위한 ALTIBASE^{TM} LOG ANALYZER 기법에 관한 연구

양형식*, 김선배**
호서대학교 벤처전문대학원 융합공학과*, 호서대학교 벤처전문대학원 정보경영학과**

A Study on ALTIBASE^{TM} LOG ANALYZER method for highly scalable, high-availability

Hyeong-Sik Yang*, Sun-Bae Kim**
Dept. of Fusion Engineering, Graduate School of Venture, Hoseo University*
Dept. of Information Management, Graduate School of Venture, Hoseo University**

요 약 최근 인터넷 뱅킹, 전자결제, 전자상거래, 홈쇼핑, 증권 거래, 민원 업무 등 미션 크리티컬한 비즈니스가 증가함에 따라 무정지 서비스에 대한 요구가 높아지면서, 기존의 단일 데이터베이스에서 클러스터링, 이중화 기법 등에 관한 고가용성 기법에 관한 연구가 증가하고 있다. ALTIBASETM Log Analyzer(이하 ALA)는 이중화 기법과는 별개로 Active Log를 기반으로 API를 제공하여, 이기종 또는 동일 기종간의 통신 및 확장성을 제공한다. 본 논문에서는 ALA를 사용하여 고가용성과 고확장성, 실시간 동기화 기능을 모두 만족시킬 수 있는 데이터베이스 시스템의 설계를 제시하고 ALA에 대한 성능평가를 하였다.

주제어 : 분산 데이터베이스, 클러스터링, 이중화, 고가용성, 고확장성

Abstract Recently, the need for non-stop service is increasing by the business mission-critical Internet banking, e-payment, e-commerce, home shopping, securities trading, and petition business increases, clustered in a single database of existing, redundant research on high-availability technologies related to technique, etc. is increasing. It provides an API based on the Active Log in addition to the technique of redundancy, ALTIBASETM Log Analyzer (below, ALA), provides scalability and communication of the same model or between heterogeneous. In this paper, we evaluated the performance of ALA by presenting the design of the database system that you can use the ALA, to satisfy all the synchronization features high scalability and high availability, real-time.

Key Words : Distributed database, Clustering, Replication, High availability, High scalability

1. 서론

최근 인터넷 뱅킹, 전자결제, 전자상거래, 홈쇼핑, 증권 거래, 민원 업무 등 미션 크리티컬한 비즈니스가 일반 생활에 깊숙이 사용되고 있고, 이러한 서비스의 증가함에 따라 무중간 서비스에 대한 요구도 점점 높아지고 있다

Received 21 January 2014, Revised 8 April 2014
Accepted 20 May 2014
Corresponding Author: Sun-Bae Kim(Graduate School of Venture, Hoseo University)
Email: hsyang@hoseo.edu

ISSN: 1738-1916

[1]. 이에 따라, 기존의 인터넷 환경에서 단일 대용량 데이터베이스 서버를 구성하여 고가용성(High availability)의 서비스를 제공하는 것이 한계에 도달하였고, 이를 극복하기 위한 클러스터링 기법, 이중화 기법 등 고가용성에 대한 연구 및 개발이 활발히 이루어지고 있다[2],[3].

ALTIBASE™ 하이브리드 DBMS 시스템(이하 ALTIBASE)는 하나의 엔진에서 디스크와 메인메모리를 주기억 장치로 사용할 수 있는 하이브리드 DBMS이다[4]. ALTIBASE는 데이터베이스의 고가용성에 대한 요구를 만족시키기 위해, 즉, 어떠한 장애 상황에서도 무정지 데이터베이스 서비스를 위해 실시간 네트워크 데이터 동기화를 통한 이중화 기법을 제공한다.

ALTIBASE에서 제공하는 이중화 기법은 테이블 그룹 단위의 이중화 기법이다[5]. 이중화를 구성하는 각 노드는 동일한 DISK 혹은 동일한 MEMORY 데이터 공간을 별도로 구성하게 된다. 이중화 대상에 속한 테이블은 모든 DML에 대해 자동으로 동작하게 된다. 이와 같은 특징 때문에, 특정 테이블의 특정 컬럼 값을 기준으로 일부 데이터만을 동기화하는 선택적 이중화에 대한 기능은 지원하지 않는다.

따라서, 최근 ALTIBASE는 ALA(Altibase Log Analyzer)를 제공하고 있다[6]. ALA는 발생되는 DML의 Active Log(Redo Log)를 기반으로 DML 관련 트랜잭션의 이력을 제공하는 DBMS 내의 모듈, 이 모듈과 통신으로 연결된 외부 모듈, 그리고 사용자가 Active Log를 기반으로 변경된 XLog를 사용하기 위해 제공되는 API의 집합이다. 사용자는 변경 DML이 발생될 경우, ALA에서 제공하는 API를 이용하여 변경 DML에 대한 XLog를 타 DBMS에 실시간으로 가공, 전송하여 동기화가 가능하다.

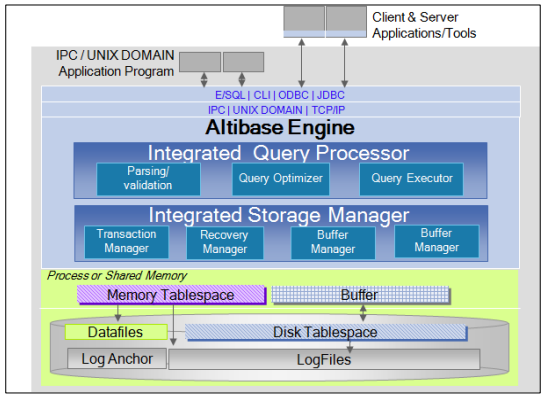
본 논문에서는 ALTIBASE의 구조를 설명한 후, ALTIBASE에서 제공하는 이중화와 비교하여 ALA를 비교 기술한다. 또한, ALA를 사용하여 고가용성과 고확장성, 실시간 동기화 기능을 모두 만족시킬 수 있는 데이터베이스 시스템의 설계를 제시한다.

본 논문의 구성은 다음과 같다. 1장 서론에 이어, 2장에서 ALTIBASE의 구조 및 주요 기능에 대하여 설명한다. 3장에서는 ALA의 구조와 동작방식 및 ALA에서 제공하는 API에 대해 알아보고, 고가용성, 고확장성(High scalability)을 제공하기 위한 ALA 설계 방안을 제안한다. 4장에서는 ALA를 적용하였을 때, ALA의 초당 처리량, CPU 부하율, 트랜잭션 처리시간, 전송시간에 대한 성능평가 결과를 설명한다. 끝으로 5장에서는 결론과 향후 연구과제에 대해 요약한다.

2. 관련 연구

2.1 ALTIBASE™ 하이브리드 DBMS

ALTIBASE™ 하이브리드 DBMS 시스템은 관계형 데이터 모델을 지원하는 RDBMS로써 고속의 실시간 데이터 처리를 위한 MMDBMS와 대용량 데이터의 관리를 위한 DRDBMS가 결합한 형태이다[7]. 다음 [Fig. 1]은 Structure of ALTIBASE™ hybrid DBMS이다.



[Fig. 1] Structure of ALTIBASETM hybrid DBMS

위 [Fig. 1] Structure of ALTIBASE™ hybrid DBMS와 같이 응용 프로그램 작성을 위한 클라이언트 라이브러리, 클라이언트와 서버 간 통신을 하기 위한 통신 모듈, 그리고 클라이언트로부터 요청받은 SQL을 처리하기 위한 서버 엔진 부분으로 구성되어 있다. 클라이언트 라이브러리 부분은 프리컴파일러 또는 ODBC 등의 범용적인 라이브러리를 제공한다. 통신 모듈은 JDBC, CLI 등의 다양한 인터페이스를 제공하며 서버와 통신을 위한 IPC, Unix Domain Socket, TCP/IP 방식을 제공한다. 서버 엔진 부분은 통합 쿼리 프로세서(Integrated Query Processor)와 통합 스토리지 매니저(Integrated Storage Manager) 부분으로 구성된다.

통합 쿼리 프로세서는 요청받은 SQL을 파싱하는 역

할을 하는 질의 파서(Query Parser), 파싱된 SQL을 효과적으로 수행할 수 있게 실행 계획을 수립하는 질의 최적화기(Query Optimizer), 질의 최적화기가 수립한 실행계획을 실제로 수행하는 질의 실행기(Query Executer)로 구성된다.

통합 스토리지 매니저는 트랜잭션이 가지는 고유한 성질인 ACID을 보장하기 위한 트랜잭션 매니저(Transaction Manager), 백업 및 복구를 위한 리커버리 매니저(Recovery Manager), 디스크 테이블 사용 시 메모리에 로드 및 언로드 하는 버퍼 매니저(Buffer Manager), DBMS에서 제공하는 인덱스 생성 및 관리를 하는 인덱스 매니저(Index Manager)로 구성된다.

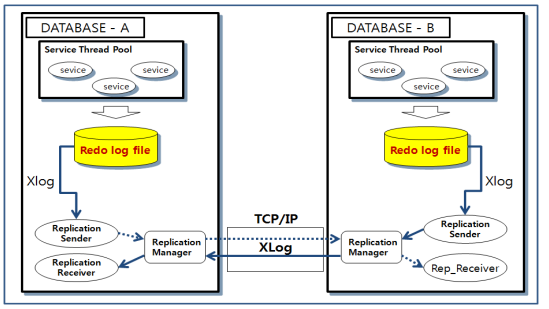
메모리 테이블스페이스의 데이터일 경우 트랜잭션의 ACID 속성을 보장하기 위해 다중 버전 기법(Multi-Version Concurrency Control)을 이용한 동시성 제어를 수행한다. 다중 버전 기법은 여러 개의 데이터 버전을 유지하여 동일 레코드에 대한 읽기와 쓰기 연산에 대한 충돌을 방지하는 것이다[8].

데이터의 영속성(Durability) 보장을 위해 디스크 파일에 데이터를 저장한다. 이와 같이 테이블스페이스의 데이터가 저장되는 파일을 체크포인트 이미지라고 하며, 두 벌의 체크포인트 이미지를 두고 하나씩 번갈아 가면서 테이블스페이스의 데이터를 저장하는 퍼지&핑퐁 체크포인트를 수행한다. 디스크 테이블스페이스의 데이터일 경우 디스크로부터 읽은 데이터를 메모리에 캐싱하는 버퍼를 통한 디스크 I/O 회수를 최소화하여 저장한다. 이러한 일련의 작업은 데이터베이스 안정성과 영속성을 위하여 변경된 데이터베이스 내용에 대해 로깅(logging)을 수행한다.

2.2 ALTIBASE™ 이중화 기법

이중화는 장애 발생 시 서비스 정지 시간을 최소화하여 고가용성을 구성하는 방법 중 하나이다[9]. ALTIBASE에서 제공하는 이중화는 서비스 중인 데이터베이스의 Active Log를 원격 데이터베이스로 전송하는 것을 기본 방식으로 취하고 있다. 다음 [Fig. 2]는 Operation the Replication thread를 나타낸다.

위 [Fig. 2] Operation the Replication thread 처럼, 로컬 서버(local server)는 데이터베이스의 INSERT, UPDATE, DELETE의 변경 트랜잭션 로그를 XLog로 변환하여 원격 서버(remote server)로 실시간 전송하고, 원격 서버는 전송받은 XLog를 자신의 데이터베이스에 적용하여 양쪽 데이터베이스는 최신의 같은 데이터를 유지하게 된다.



[Fig. 2] Operation the Replication thread

이중화 모듈은 이중화 서버간의 통신을 담당하는 이중화 관리 쓰레드(Replication_Manager)와 XLog를 원격 서버에 보내는 전송 쓰레드(Replication_Sender), 원격 서버의 XLog를 전달 받아 자신의 데이터베이스에 반영하는 수신쓰레드(Replication_Receiver)로 구성된다. 전송 쓰레드는 변경 트랜잭션에 의해 발생한 XLog를 원격 서버로 보내는 쓰레드다. 로컬 서버의 이중화 대상 테이블에 변경 트랜잭션을 수행함으로써 생성되는 Active Log를 데이터의 변경정보를 가지고 있는 XLog 형태로 변환하여 원격 서버로 보낸다. 수신 쓰레드는 XLog를 받아서 이중화 대상 테이블에 반영한다.

이중화는 비동기(Lazy)방식과 동기(Eager) 방식을 제공한다[10],[11]. 비동기 방식은 성능 우선의 데이터 동기화 방식이며, 동기 방식은 데이터 정합성 우선의 데이터 동기화 방식이다. 동기 방식이 비동기 방식에 비해 성능 저하를 감수하고 높은 데이터 정합성을 보장하지만, 두 방식 모두 본질적으로 2개 이상의 저장소의 데이터 일치성을 완전히 보장할 수는 없다. 이중화 기법은 네트워크를 통한 데이터 동기 방식으로 동작하기 때문에 본질적으로 2개 이상의 저장소의 데이터 정합성을 완전히 보장할 수는 없다.

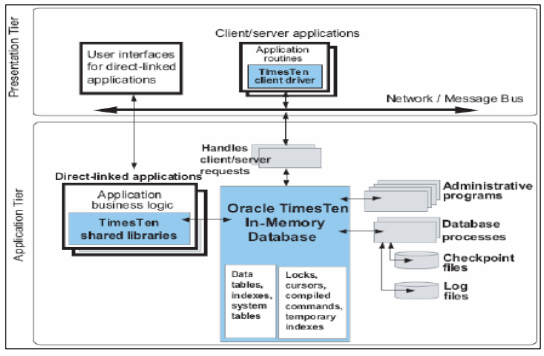
2.3 오라클™ TimeSten 이중화 기법

오라클 TimeSten은 ALTIBASE DBMS와 같은 IN-MEMORY 기반의 MMDBMS이다[12]. TimeSten DBMS와 통신하기 위한 공유 라이브러리를 제공한다.

〈Table 2〉 Requirements of Interface about ALA

Terms	Concepts
XLog	· An XLog is a logical form of log that results from the conversion of a physical log. It is used to store a history of transactions involving DML (INSERT, UPDATE and DELETE) statements such that it can be accessed by users.
XLog Sender	· The XLog Sender is the module that analyzes active logs to create XLogs, which it then passes to the XLog Collector.
XLog Collector	· The XLog Collector is the module that receives meta data and XLogs from the XLog Sender.
Log Analysis API	· The Log Analysis API is used to obtain XLogs and meta data that are used to interpret the XLogs.
Handshake	· Handshaking is the task of checking the protocol version, meta data, etc. before the XLogs are sent from the XLog Sender to the XLog Collector.
XLog Queue	· The XLog Queue is the place where available XLogs are stored before they are accessed by users.
XLog Pool	· The XLog Pool is memory that has been allocated for the storage of XLogs.
Transaction Table	· The transaction table is the place where information about transactions, including the status of transactions, is stored.
SN	· The SN (Sequence Number) is the serial number of an individual log record.
Restart SN	· The Restart SN is the SN of the active log from which reading will resume when the XLog Sender is restarted.
Replication	· Replication is a module that is used to synchronize data between two Altibase databases. For more information, please refer to the ALTIBASE HDB Replication Manual.

TimesTen 로그 버퍼는 공유메모리 영역에 할당되며 데이터베이스내의 모든 사용자 데이터와, 시스템 메타 테이블, 임시 테이블스페이스 등을 저장한다. 이렇게 메모리에 상주된 데이터 및 수행되는 트랜잭션의 정합성을 보장하기 위해 실행되는 모든 트랜잭션은 로그파일에 기록되고, 디스크에 주기적으로 체크포인트를 발생하여 동기화한다. 완료된 트랙잭션에 대한 로그 파일은 체크포인트 발생 시 자동 삭제되며, 만약 DB가 비정장 종료될 경우 저장된 로그파일을 기반으로 자동복구 가능하다. 아래 그림[Fig. 3]은 TimeSten MMDBMS의 구조를 나타낸다.



[Fig. 3] Structure of TimeSten™ MMDBMS

오라클 TimeSten은 고가용성을 위한 방안으로 실시간 트랜잭션 기반 Replication을 제공한다. TimeSten Replication은 Master서버와 Subscriber서버로 구분되며, replication 생성 시 명시함으로써 역할이 지정된다. Replication은 크게 3가지로 제공되며 아래 <table 1>에서와 같이 No Return, Return, Two-Safe 모드로 구분된다.

〈Table 1〉 Classification by TimeSten replication operation

Terms	Concepts
No Retuen	The asynchronous transfer to the Subscriber the Commit transaction
Retuen	Before responding to the application transactions Commit, and ensure that it is passed to the subscriber
Two-Safe	Before responding to the application transactions Commit, to ensure that is transmitted to the subscriber, it is applied to the log file

위 <Table 1>에서 제공하는 Replication에 대한 동작 구분을 ALTIBASE에서 제공하는 이중화 기능과 비교하면, No-Return 모드는 ALTIBASE의 Lazy 모드, Return 모드는 Acked 모드, Two-safe 모드는 Eager 모드와 같다. 하지만, TimeSten의 Replication은 Master-Subscriber 구분을 Replication 생성 시 지정하기 때문에, Active-Standby 구조만 사용이 가능하며, Full-Mesh 형태의 Active-Active 구성은 불가능하다. 또한, 테이블 그룹 단위의 데이터스토어를 확장을 하려면 Replication 객체를 재생성하여여 하는 단점이 있다.

3. ALTIBASE™ Log Analyzer 기법

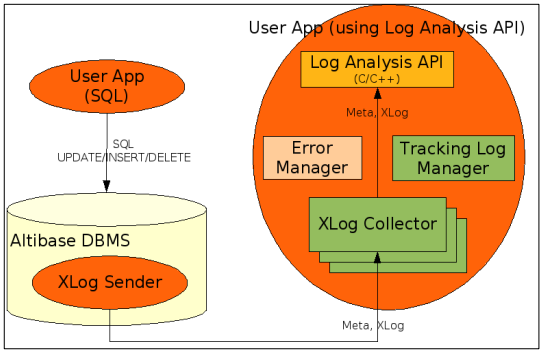
3.1 ALTIBASE™ Log Analyzer 구조

ALA는 Active Log를 기반으로 DML 관련 트랜잭션의 이력을 제공하기 위한 DBMS 내의 프로세스 모듈, 외부 통신 모듈, XLog 모듈의 API 집합이다. ALA는 아래 <Table 2> Requirements of Interface about ALA에서 제공하는 XLog Sender, XLog Collector, XLog Receiver 모듈과 Log Analysis API, Handshake, 이중화 기능을 사용하여 사용자가 응용 프로그램을 작성 시, DBMS간 연동, DBMS 내부의 변경 사항을 DBMS 외부에서 감지 및 처리 등의 용도로 사용할 수 있다.

ALA는 이중화 모듈의 Sender와 Receiver 모듈을 기반으로 동작한다. 따라서, XLog Sender는 이중화와 거의 동일한 SQL를 사용하여 관리되며, 이중화의 속성이 적용된다. XLog Sender는 XLog를 생성하고, 생성된 XLog와 Meta 정보를 XLog Collector에게 Handshake 및 XLog 송신을 능동적으로 수행한다.

XLog Collector는 사용자의 애플리케이션 내에 존재하고, XLog Sender로부터 수신된 XLog와 Meta 정보를 Log Analysis API를 통해 사용자에게 XLog와 메타 정보를 제공한다. 사용자는 제공된 XLog와 메타 정보를 이용하여 데이터를 가공, 처리, 감시가 가능하게 변경하여 XLog Receiver에 전송이 가능하다.

Log Analysis API를 호출에 실패하면, 오류 원인에 따라 적절한 조치를 취해야 한다. 가장 최근의 오류 정보를 보관하는 곳이 Error Manager이다. 또한 문제 추적을 위해 Log Manager를 제공한다. Log Manager는 간단한 추적 정보와 오류 정보를 Log Manager생성시 지정한 로그 파일에 기록한다. 아래 [Fig. 4]는 The structure of ALA를 나타낸다.



[Fig. 4] The structure of ALA

위 그림 [Fig. 4] The structure of ALA와 같이 사용자는 Log Analysis API를 사용하여 XLog Collector에서 메타 정보와 XLog를 얻는다. 메타 정보는 ALA를 동작한 이후 Handshake시 XLog Sender에서 수신하며, 사용자가 임의로 ALA를 종료하거나, Handshake를 종료하기 전까지 유효하다.

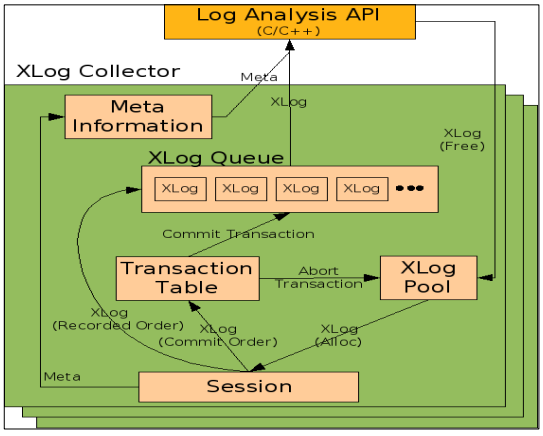
XLog와 Meta 정보 다음과 같은 순서로 이동된다. 먼저, XLog에 할당될 메모리를 재사용하고 과다한 메모리 사용을 방지하기 위해 XLog Pool에서 XLog의 메모리를 얻는다. XLog Collector는 XLog Sender로부터 수신된 XLog와 Meta 정보를 기반으로 XLog Queue에 저장한다. 이때, 아래 정의된 <Table 3> The operating division of ALA Features에 따라 다르게 처리된다.

〈Table 3〉 The operating division of ALA Features

Operation Division	Features
Recored Order	· AUTOCOMMIT MODE · Transaction XLog run in order to Record
Commit Order	· NON - AUTOCOMMIT MODE · Keep Transaction Table · Order to run the Commit Transaction XLog · COMMIT XLog is received, then that transaction can be obtained XLog · Savepoint related XLog is not provided because it does not require · XLog of the Transaction rollback can not be obtained

<Table 3> The operating division of ALA Features에 의해 사용자는 XLog Queue로부터 Log Analysis API를 사용해서 XLog를 분석/변환하여 XLog Receiver에 전송한다. 사용이 끝난 XLog에 할당된 메모리는 XLog Pool에 반환하여 XLog Collector에서 재사용하게 된다. 이를 그림으로 나타내면 아래 [Fig. 5] The internal structure XLog Collector와 같다.

XLog Analyzer의 제약 사항으로 이중화 모듈을 사용하므로, SYS 사용자만 XLog Sender를 구동할 수 있다. 또한, 이중화 제약사항과 마찬가지로 분석 대상을 지정하는 단위는 Table이고 분석 대상 Table에는 반드시 Primary Key가 존재해야 하며, Primary Key를 구성하는



[Fig. 5] The internal structure XLog Collector

컬럼의 데이터는 UPDATE가 불가능하다. 하지만, Primary Key를 구성하는 컬럼의 데이터에 대해 INSERT와 DELETE는 가능하다. 또한, ALA로 연결된 Table에 대해서는 DDL을 수행할 수 없으며, XLog Sender와 Replication Sender를 합쳐 최대 32개까지 만들 수 있다. 하지만, 이중화와 달리 API기반으로 동작되기 때문에, Table에 따라 FK를 가진 Table도 로그 분석 대상이 가능하고 Lazy 모드만 지원이 가능하며, 이중화 SYNC 기능을 지원하지 않는다.

3.2 Log Analysis API

Log Analysis API는 C/C++ 언어 기반으로 사용자의 애플리케이션에서 호출하는 API이며, XLog Sender에서 XLog를 받고 이를 분석/변환하는 기능을 제공한다. 아래 <Table 4>는 Files required to create the application을 나타낸다.

<Table 4> Files required to create the application과 같이 Log Analysis API를 사용하기 위해 응용프로그램의 Header 파일로서 alaTypes.h, alaAPI.h 파일을 include 하고, Library 파일로 링크시 libala_sl.x 또는 libala.x을 링크해야한다. header 파일인 alaTypes.h에는 XLog를 분석하기 위한 13 가지의 트랜잭션 관련 XLog와 2 가지의 Control 관련 XLog가 정의되어 있다. 트랜잭션 관련 XLog에는 트랜잭션의 Commit/ Rollback 구분, Insert, Update, Delete와 같은 DML 구분, Non-Autocommit Mode 사용 시 Savepoint 관련 구분,

〈Table 4〉 Files required to create the application

File Type	File name	Description
Header	alaTypes.h	This file contains the definitions of data types and macros that are necessary when writing a client program using the Log Analysis API.
	alaAPI.h	This file must be included when authorizing a client program using the Log Analysis API. It includes the alaTypes.h file.
Library	libala_sl.x	This is the Log Analysis API shared library.
	libala.x	This is the Log Analysis API static library.

LOB와 같은 큰 데이터에 대한 구분이 상수로 정의되어 있다. Control 관련 XLog에는 XLog Sender가 네트워크 연결 유지 혹은 정상 종료를 의미하는 값이 정의되어 있다. 따라서, 사용자는 정의된 값을 기준으로 XLog를 분석하여 트랜잭션 분류에 따른 변환 및 처리가 가능하다.

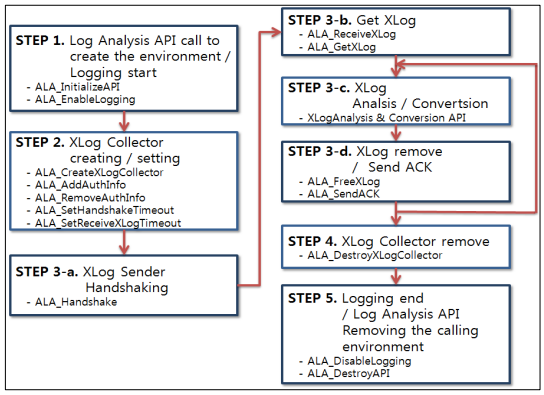
header 파일인 alaAPI.h 파일에는 환경 관리를 위한 API와 XLog Collector 관련 API, XLog Analysis & Conversion API가 정의되어 있다. 아래 <Table 5>는 The main function of the Log Analysis API를 나타낸다.

〈Table 5〉 The main function of the Log Analysis API

Funtion Type	Log Analysis API	Function Name
Environmental Management API	Creating and Destroying the Log Analysis API Environment	ALA_InitializeAPI
		ALA_DestroyAPI
	Logging	ALA_EnableLogging
		ALA_DisableLogging
XLog Collector API	Creating and Preparing the XLog Collector	ALA_CreateXLogCollector
		ALA_AddAuthInfo
		ALA_RemoveAuthInfo
		ALA_SetHandshakeTimeout
		ALA_SetReceiveXLogTimeout
	Receiving Meta Data and XLogs	ALA_Handshake
		ALA_ReceiveXLog
		ALA_GetXLog
		ALA_SendACK
		ALA_FreeXLog
	Terminating an XLog Collector	ALA_DestroyXLogCollector

XLog Analysis & Conversion API	XLog	ALA_GetXLogHeader
		ALA_GetXLogPrimaryKey
		ALA_GetXLogColumn
		ALA_GetXLogSavepoint
		ALA_GetXLogLOB
	Meta Information	ALA_GetProtocolVersion
		ALA_GetReplicationInfo
		ALA_GetTableInfo
		ALA_GetTableInfoByName
		ALA_GetColumnInfo
		ALA_GetIndexInfo
	ALTIBASE Internal Data Type	ALA_GetInternalNumericInfo
		ALA_GetAltibaseText
		ALA_GetAltibaseSQL

위 <Table 5> The main function of the Log Analysis API에 정의된 Log Analysis API 주요 함수를 기반으로 전체적인 ALA의 Log Analysis API 사용 순서를 설명하면 그림 [Fig. 6] Order of use of the Log Analysis API of ALA과 같다.



[Fig. 6] Order of use of the Log Analysis API of ALA

위 그림 [Fig. 6] Order of use of the Log Analysis API of ALA에서와 같이 ALA는 최초 STEP 1.에서 ALA_InitializeAPI(), ALA_EnableLogging() API를 호출하여 Log Analysis API 호출 환경 생성 및 Logging 시작한다. STEP 2.에서는 ALA_CreateXLogCollector()를 호출하여 XLog Collector를 생성하고, XLog Collector에 대한 설정을 한다. 설정이 완료되면, STEP 3.에서 ALA_Handshake()를 호출하여 XLog Sender와 생성된 XLog Collector와 Handshake 한다. 이후, STEP 3-b.에서 XLog Sender는 사용자 트랜잭션 Active Log를 변환한 XLog를 XLog Collector에게 전송한다. 전송된 XLog는 ALA_ReceiveXLog(), ALA_GetXLog()를 이용해 XLog Collector에 의해 XLog Queue에 수집된다. 수집된 XLog는 STEP 3-c.에서 XLogAnalysis & Conversion API에 의해 분석되고 변환하여 XLog Receiver에 전송한다. 전송된 XLog는 STEP 3-d.에서 ALA_FreeXLog()에 의해 XLog queue에서 제거되고 제거된 공간은 재사용된다. 이후, XLog Collector는 XLog Sender에 ALA_SendACK()를 보낸다. STEP 3-b.에서 STEP 3-d.까지는 XLog Sender가 종료되기 전까지 XLog Queue에 있는 Xlog에 대해 반복 수행된다. XLog Sender를 종료하면 STEP 4.에서 ALA_DestroyXLogCollector()가 호출되어 XLog Collector을 제거하고, STEP 5.에서 ALA_DestroyAPI(), ALA_DisableLogging()를 호출하여 Log Analysis API환경을 제거하고 Logging을 종료한다.

3.3 고가용성을 위한 ALA 통신 모델

ALTIBASE에서 제공하는 이중화 기능과 ALA는 DBMS에서 내부에서 제공되는 동일한 이중화 모듈을 사용한다. 아래는 <Table 6> Feature comparison of ALA with Replication이다.

〈Table 6〉 Feature comparison of ALA with Replication

Division	Feature
Replication	· Can be set up to up to 32 linked Replication Recevier and Replication Sender · In One Way system, in order to connect to multiple servers, it is necessary to Replication Receiver and Replication Sender of each to produce only the number of servers of each · Replication transaction will not be sent · The user can not modify the replication transaction XLog
ALA	· Can be set up to up to 32 linked Replication Recevier and Replication Sender · There is no limit to the number for the connection to multiple servers from XLog Collector in the One Way system. · XLog Collector and can be located on any server · Replication transaction can be transferred · User analysis / convertible XLog the replication transaction

위 <Table 6> Feature comparison of ALA with Replication과 같이 이중화 기능은 이중화 모듈의 수신

쓰레드를 사용하여 통신하므로, 서버 확장에 대한 용이성과 사용자에 의한 XLog의 분석 및 변환을 제공하지 않는다. 따라서, ALA는 장점은 다음과 같이 3가지로 요약될 수 있다.

첫째로, 이중화와 동시에 ALA를 독립적으로 추가 구성할 수 있다. 다시 말하자면, 동일한 Table에 대해 이중화와 ALA를 동시에 구성할 수 있으며, ALA의 XLog Sender는 XLog Collector와 쌍으로 구성되기 때문에 최대 32개 Xlog Receiver의 제약이 없어진다.

둘째로, API 기반으로 동작하기 때문에 분석 대상 Table에 따라 Xlog를 분석/처리하여 사용자가 임의로 필요한 데이터만을 추출하는 선택적 이중화가 가능하다.

셋째로, 사용자 응용프로그램과 동시 배포될 수 있으며, 이중화 Sender를 여러 개 생성할 필요가 없어 응용프로그램의 위치에 따라 DBMS의 CPU 사용률을 줄일 수 있다.

ALA를 사용하여 통신을 구성하는 노드의 역할에 따라 다음 <Table 7> ALA node classification과 같이 세 가지로 구분될 수 있다.

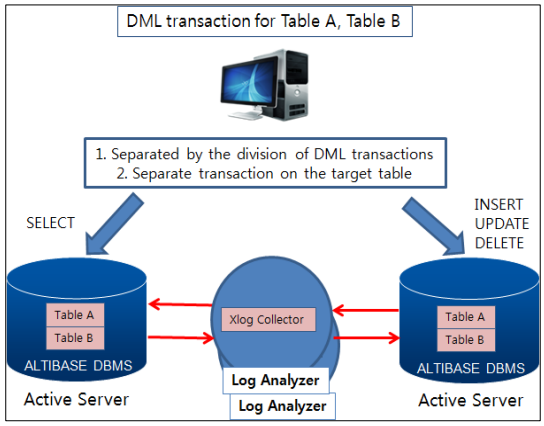
〈Table 7〉 ALA node classification

Classification	Feature
Active server	· The actual service server · XLog applying remotely connected servers.
Standby server	· Server that is not the actual service (Failure / server switching for an HA configuration) · XLog applying remotely connected servers. (Auto-recoverable failure)
Master server	· Server that is not the actual service (statistics management) · XLog applying remotely connected servers.

위 <Table 7> ALA node classification과 같이 실제 서비스를 하는지 유무에 따른 구분과, XLog를 원격으로 연결된 서버에 적용하는지 유무에 따른 각 노드의 역할에 따라 정의가 가능하다. 정의된 <테이블 7>를 기반으로, 아래 그림과 같이 ALA 통신 모델을 아래 그림과 같이 분류할 수 있다.

3.3.1 Active-Active 통신 모델

Active - Active 통신 모델은 구성된 노드 모두가 실제 서비스를 하면 변경된 데이터데 대한 XLog를 원격으로 연결된 서버에 전송하는 구조이다. 이러한 통신 모델은 주로 서비스의 부하 분산을 위해 주로 사용된다. 그림 [Fig. 7]은 Communication model of Active - Active를 나타낸다.



[Fig. 7] Communication model of Active – Active

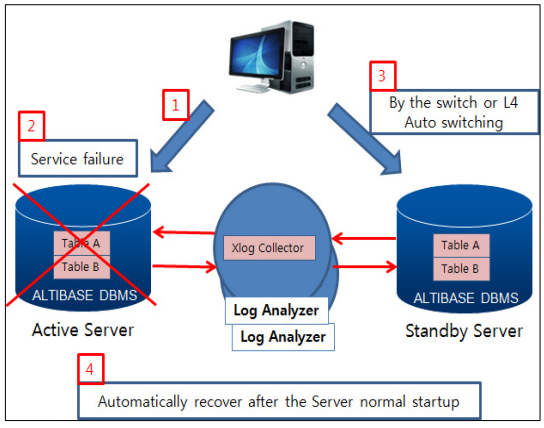
위 그림[Fig. 7] Communication model of Active - Active 같이 각 서버는 요청된 트랜잭션을 XLog로 변환하여 XLog Collector에게 전송하고, XLog Collector는 XLog를 분석 변환하여 원격 서버에 실시간으로 전송한다. 전송단위는 ALA를 구성하기에 따라 위 그림과 같이 Table A, Table B를 하나의 그룹으로 처리하거나, 또는 Table A와 Table B를 각각 하나의 그룹으로 분할하여 처리가 가능하다.

하지만, Active - Active 통신 모델은 각 DBMS의 저장장치를 공유하지 않기 때문에 동일 시간대에 동일 Table에 변경 트랜잭션이 일어날 경우, 네트워크 지연에 의한 데이터 충돌이 발생할 수 있다. 또한, 특정 Active 서버에 장애가 발생하면, 트랜잭션은 장애가 발생되지 않은 Active 서버로 모든 트랜잭션이 순간적으로 집중되고, 이로 인해 서버 과부하가 발생될 수 있다. 따라서, 위 그림과 같이 DML 트랜잭션을 분리하여 처리하거나, 혹은, 대상 Table 구분에 따른 트랜잭션으로 분리 및 적용하여, 각 트랜잭션의 결과를 조합하는 방식으로 데이터의 정합성을 유지하고 실시간으로 동기화한다.

3.3.2 Active-Standby 통신 모델

Active - Standby (혹은, Active - 다중 Standby)통신

모델은 구성된 노드 중 실제 서비스를 하면서 변경된 데이터데 대한 XLog를 원격으로 연결된 서버에 전송하는 Active 서버와 실제 서비스를 하지 않는 서버가 구성된 구조이다. 이와 같은 통신 모델은 주로 서비스에 장애 발생 시, 고가용성을 위해 주로 사용된다. 다음 그림 [Fig. 8]은 Communication model of Active - Standby이다.



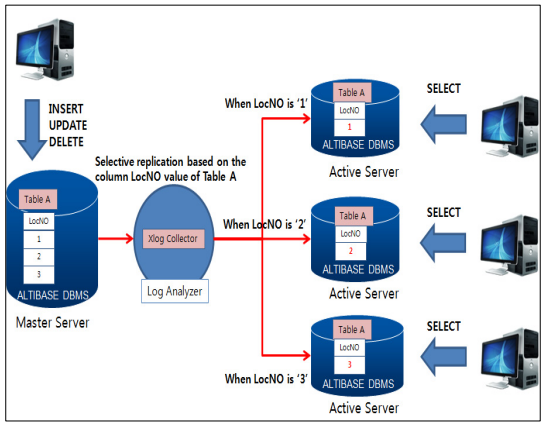
[Fig. 8] Communication model of Active – Standby

위 그림[Fig. 8] Communication model of Active - Standby 같이 사용자 클라이언트는 최초 (1)처럼 정상적으로 서비스를 하다가 (2)처럼 서버에 장애가 발생하면 해당 서버에 대한 연결을 끊고, (3)처럼 Standby 서버로 자동으로 절체되어 서비스를 계속한다. 이때, Xlog Collector는 Standby 서버에 변경된 DML이력을 순차적으로 XLog Queue에 저장한다. 장애가 발생되었던 이전 Active 서버가 복구되면, (4)처럼 XLog Collector는 XLog Queue에 쌓인 DML 이력을 순차적으로 전송하여 최근 이력까지 자동 복구되어 최초 Active-Standby 모델로 서비스가 가능하다.

Active - Standby 통신 모델은 Active 서버로 들어오는 트랜잭션의 부하를 줄일 수 없기 때문에 Active 서버의 부하가 높을수록 서비스에 대한 안정성은 떨어진다. 하지만, Active 서버에 장애가 발생하더라도 즉시 절체가 가능하여 장애 발생 없이 지속적인 서비스가 가능하다.

3.3.3 Master-Active 통신 모델

ALA는 XLog Collector에서 분석되는 XLog에 따라 사용자가 변환이 가능하다. 따라서, 아래 그림과 같은 Master - Active(혹은, Mastar-다중 Active)통신 모델의 구성이 가능하다. 이러한 통신 모델은 주로 메인 서버의 특정 테이블 혹은 특정 테이블의 컬럼 값을 기준으로 선택적 이중화를 하여 서비스를 분산하기 위해 주로 사용된다. 다음 그림 [Fig. 9]는 Communication model of Master - Active이다.



[Fig. 9] Communication model of Master-Active

위 그림 [Fig. 9] Communication model of Master - Active 같이 Master 서버는 Table A의 컬럼 LocNO의 값을 가지고 있다. Table A에 대한 트랜잭션이 발생하면 XLog를 XLog Collector에서 수집하고 Table A의 LocNO 컬럼의 값을 분석한다. XLog Collector는 사용자에 의해 정의된 논리에 따라 LocNO의 값을 기반으로 각 Active 서버로 데이터를 전송한다. 이러한 방식으로 전송된 데이터는 각 Active 서버에서 다른 클라이언트에 의해 서비스를 하게 된다.

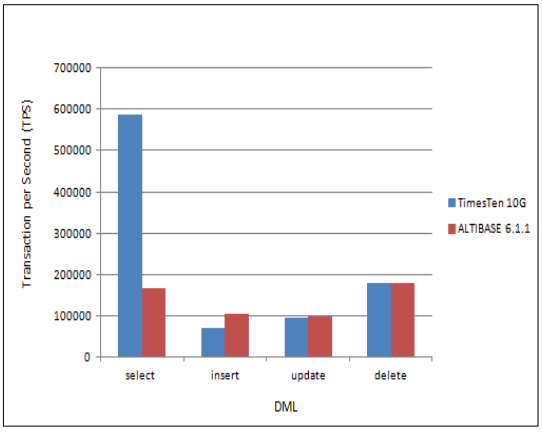
Master-Active 통신 모델은 Master 서버로 들어오는 DML 트랜잭션을 관리하고, 실제 서비스는 각 Active 서버에서 하여 Active 서버에 장애가 발생하더라도 Master 서버를 기준으로 복구가 가능하다. 또한, 동일한 방법으로 XLog Collector 혹은 replication을 추가하여 Active - Standby 형태로 Active 서버의 데이터를 백업 혹은 고가용성을 위한 절체 용도로 사용 가능하다.

4. ALA 성능평가

본 절에서는 ALA의 트랜잭션 수행에 대한 TPS(Transaction per Second)와 CPU부하율에 대한 성능평가를 보인다. 본 절에서 수행한 성능평가는 2.33GHz CPU 8개와 15G 바이트 메모리를 보유한 "Linux as48-x64" 운영체제 장비와 1.59 GHz CPU 4개와 32G 바이트 메모리를 보유한 "HP-UX B.11.31" 운영체제 장비에서 ALTIBASE™ 하이브리드 DBMS version 6.1.1.2.1과 오라클 TimeSten 10G로 비교 수행하였다.

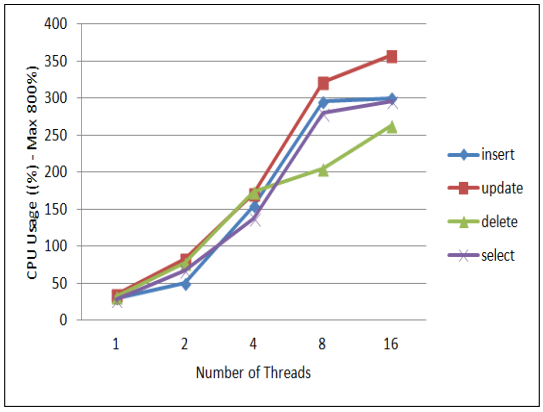
성능평가 대상 테이블은 'nteger', 'varchar','date' 등의 여러 가지 속성들로 구성되는, 총 10개의 속성을 갖는 TB_TEST1 단일 테이블이며, 모두 질의 처리기(Query Processor 모듈)를 거쳐 최적 수행 계획에 따라 수행되며, 네트워크 지연에 따른 간섭은 실험에 평가되지 않는다. 실험에 사용된 트랜잭션은 모두 4종류, INSERT, UPDATE, DELETE 그리고 SELECT 트랜잭션이다. 동시 실행하는 쓰레드 개수는 실험에 따라 단일 사용자에서 16개의 쓰레드로 변화를 주었으며, 레코드의 개수는 총 4,000,000(일백만)개의 레코드로 성능측정 하였다.

다음 그림 [Fig. 10]은 ALTIBASE와 TimeSten의 Measurement of TPS performance이다. SELECT 구문을 제외한 나머지 INSERT, UPDATE, DELETE 구문의 TPS는 ALTIBASE가 높게 측정되었으며, 분석 결과, SELECT 수행 시 HASH인덱스를 통한 메모리 주소로 직접 접근하여 ALTIBASE에 비해 빠른 성능을 보인다.



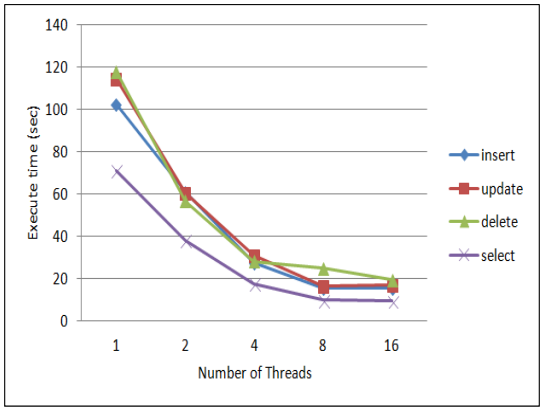
[Fig. 10] Measurement of TPS performance

다음 그림 [Fig. 11]과 [Fig. 12]는 성능측정 당시 ALTIBASE DBMS 프로세스만의 CPU performance와 Execute Time performance를 보여주고 있다.



[Fig. 10] Measurement of CPU performance

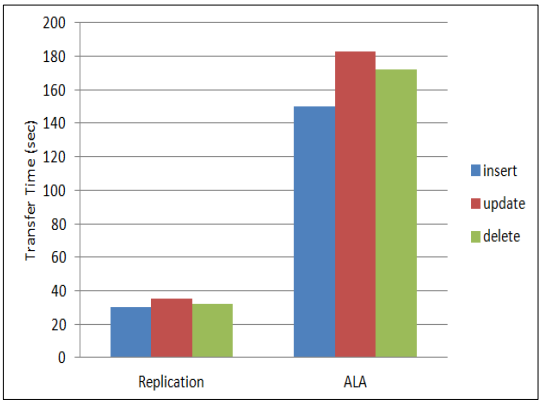
트랜잭션 구분에 따른 성능은 UPDATE 트랜잭션이 다른 트랜잭션보다 CPU를 많이 사용하고 있다. 또한, 아래 그림 [Fig. 12] Measurement of Execute-Time performance 같이 모든 트랜잭션들은 쓰레드 개수에 비례하여 CPU 사용량 및 수행시간이 점점 증가 또는 감소하는 모습을 보이고 있다.



[Fig. 12] Measurement of Execute-Time performance

하지만, 쓰레드 개수가 증가하면 증가할수록 쓰레드간의 CPU 경합이 발생하여 CPU 사용률은 감소한다.

아래 그림 [Fig. 13] Performance measurements on log transfer rate of ALA and Replication이다.



[Fig. 13] Performance measurements on log transfer rate of ALA and Replication

위 그림 [Fig. 13] Performance measurements on log transfer rate of ALA and Replication 같이 ALA를 적용한 로그 전송 속도는 평균 150초 정도로 이중화의 평균전송 속도 보다 차이가 많이 발생한다. 이는 이중화는 내부 모듈 통신을 통해 DBMS의 QP (Query Processor) 모듈을 통하지 않고, SM (Storage manager)에 직접 적용하는 방식이다. 하지만, ALA의 경우 DBMS에서 작성된 REDO LOG를 기반으로 XLog 로 변환해서 반대편 서버의 QP 모듈을 거쳐 처리되므로 전송된 데이터의 적용 시간은 ALA가 상대적으로 느린 것을 확인할 수 있다.

하지만, 위에서 언급하였듯이 ALA의 위치에 따라 DBMS의 CPU 사용률을 최소화할 수 있으며, 특히, 사용자가 API를 호출하여 Active Log를 직접 수정/ 가공하여 이기종간의 DB와 연동이 가능한 장점이 있다.

5. 결론

최근 미션 크리티컬한 비즈니스가 일반 생활에 깊숙이 사용되고 있다. 이러한 서비스의 증가함에 따라 기존의 인터넷 환경에서 고가용성(High availability)의 서비스를 제공하는 것이 한계에 도달하였고, 이를 극복하기 위한 클러스터링 기법, 이중화 기법 등 고가용성에 대한 연구 및 개발이 활발히 이루어지고 있다.

ALTIBASE™ DBMS는 하이브리드 DBMS으로서 고가용성 기능에 대한 요구를 만족시키기 위해실시간 네트워크 데이터 동기화를 통한 이중화 기법을 제공한다. 하지만, 이중화 기법은 테이블 그룹 단위의 이중화 기법이기 때문에, 특정 테이블의 특정 컬럼 값을 기준으로 일부 데이터만을 동기화하는 선택적 이중화 및 이기종간 동기화 기능은 지원하지 않는다.

따라서, 본 논문에서는 최근 개발된 ALA의 구조 및 동작 방식을 연구하고, ALA를 활용한 시스템 구성 방안을 제시하였다. ALA는 사용자가 DBMS내의 Active Log를 기반으로 변경된 XLog를 사용하기 위해 제공되는 API의 집합이다. 사용자는 변경 DML이 발생될 경우, ALA에서 제공하는 API를 사용하여 변경 DML에 대한 XLog를 원격 DBMS에 전송하여 동기화가 가능하다.

이러한 API 기반의 동작방식으로 인해 이중화를 이용한 시스템과는 별도로 ALA를 사용하여 타 DBMS에 추가적인 동기화 및 전용 전송서버를 두어 기존의 A-A, A-S 방식이 아닌 M-A 방식이 가능하다.

향후 연구과제로는 최근 급증하고 있는 빅 데이터 시장에 초점을 맞추어, ALA를 사용하여 정형데이터는 RDBMS에 저장하고 비정형 데이터는 NoSQL DBMS에 동시 저장하여 정형 데이터와 비정형 데이터를 동시에 분산 처리하는 연구를 수행할 필요가 있다.

REFERENCES

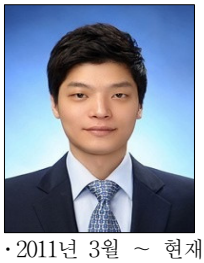
[1] R. Buyya, “High Performance Cluster Computing” , Prentice Hall, Vol 1&2, 1999.

[2] Byoung-Yup Lee, “Fail Over Analysis and Management for the Database Implement of the High Availability Solution”, Journal of Korea Contents Association, Vol. 10, NO. 7, pp. 49-54, 2010.

[3] Yong-Chang Kim, Jea-Woo Chang, Hong-Yeon Kim “implementation, and design of high availability cluster management functions for the DBMS cluster-based”, journal of korea information science society, Vol. 12, No. 1, pp. 21-30, 2006.

[4] Hwa-Jung Lim, Kyu-Woong Lee, Kwang-Chal Jung, “Design of ALTIBASETM Storage Manager for High Performance and High Availability”, journal of korea information science society, Vol. 31, NO. 1, pp. 196-198, 2004.
[5] Kyu-Woong Lee, “Managerment of Database Replication in Main Memory DBMS ALTIBASETM for High Availability”, Journal of the Korea Society for Internet Information, Vol. 6, NO. 1, pp. 73-84, 2005.
[6] ALTIBASETM, “Log Alnayzer Manual”, 2013
[7] ALTIBASETM, “Administrator’s Manual”, 2013
[8] P. A. Bernstein, V. Hadzilacos, and N.Goodman. "Concurrency Control and Recovery in Database Systems", Addison-Wesley Publishing Company, 1987.
[9] ALTIBASETM, “Replication Manual”, 2013
[10] Bettina Kemme and Gustavo Alonso, "A New Approach to Developing and Implementing Eager Database Replication Protocols", ACM Transactions on Database Systems, Vol. 25, NO.3, 2000.
[11] Jim Gray, Pat Helland, Patrick O'Neil, and Dennis Shasha, "The Dangers of Replication and a Solution", Proc. of the AMC SIGMOD International Conference on Mangement of Data, 1996.
[12] ORACLE™ Timesten “Replication Guide”, 2013.

양 형 식(Hyeong-Sik Yang)



- 2008년 2월 : 호서대학교 컴퓨터공학과 졸업(공학사)
- 2011년 2월 : 건국대학교 컴퓨터공학과 졸업(공학석사)
- 2012년 2월 ~ 현재 : 호서대학교 벤처전문대학원 융합공학과 박사과정 재학 중
- 2011년 3월 ~ 현재 : ALTIBASE(주) 기술서비스본부 재직중
- 관심분야 : 데이터베이스 시스템, DB모델링, SQL튜닝, DB 최적화, 분산데이터베이스, GIS, 운영체제
- E-Mail : sepaid12@naver.com

김 선 배(Kim Sun Bae)

- 1973년 3월 : 연세대학교 경영학과 (경영학사)
- 1991년 5월 : 美國 뉴욕대 경영대학원 (MBA)
- 2006년 3월 : 건국대 컴퓨터정보통신공학(공학박사)
- 1993년 2월 ~ 2004년 12월 : 현대정보기술 대표이사사장
- 2005년 1월 ~ 2007년 1월 : 한국정보통신수출진흥센터 원장
- 2007년 2월 ~ 2009년 2월 : 정보통신국제협력진흥원 원장
- 2009년 3월 ~ 현재 : 호서대학교 교수
- 관심분야 : 정보통신, 인터넷비즈니스, 소셜미디어
- E-Mail : sunbkim@gmail.com