플랫폼 독립적인 대용량 데이터 동기화 서버의 설계

경명진, 이민순°, 이병수
인천대학교 컴퓨터공학과
{mjcid, zerone°, bsl}@incheon.ac.kr

Design of Platform Independent Large-Scale Data Synchronization Server

Myoung-Jin Kyoung, Min-Soon Lee°, Byung-Soo Lee
Dept. of Computer Engineering, University of Incheon.

요 약

IMF 이후 기업들은 구조조정을 통한 체질개선에 총력을 기울임과 동시에 그 동안의 외형적 성장에서 탈피하여 투자 자본의 효율 향상을 위해 기업가치를 극대화하려고 노력하고 있다. 따라서, 각 사업체의 중점 육성 부분과 매각 부분을 선별하게 되었으며, 이해 사 간의 인수 합병이 빈번히 일어나고 있다. 그러나, 이렇게 생성된 기업체의 경우, 기존 전산 시스템의 하드웨어, 소프트웨어, 데이터베이스 등의 불일치로 통합에 많은 시간과 비용이 발생하고 있다. 따라서, 본 논문에서는 기존 전산 시스템의 플랫폼 불일치를 극복하며, 기존 시스템으로부터 개발되는 새로운 시스템으로의 전이를 자연스럽게 해줄 플랫폼 독립적인 대용량 데이터 동기화 서버(SyncDB2DB)를 설계한다.

1. 서 론

1997년 말, 우리에게 찾아온 IMF는 국내 많은 기업들에 변화를 요구했다. 불필요한 기업 자산에 대한 매각과 주력 사업 분야에 대한 대대적인 투자 등 기존 대기업들이 문을 닫는 현상까지 나타나게 되었다. 이러한 변화들 중 IMF로 인한 최대 변화는 무분별한 사업 확장이 아닌 서로의 시너지 효과나 시장 지배력의 확대를 위한 기업 합병이 많아 졌다는 것이다. 이렇게 합병된 기업은 합병된 기업의 전산 시스템으로 대체하기 전까지, 기존 기업들의 전산 시스템을 계속 이용해야 함과 더불어, 새롭게 개발되어지는 시스템을 추가로 이용할 수 있어야 하며, 나아가 개발이 완료된 이후에는 새로운 시스템으로의 자연스러운 전이가 가능해야 한다. 따라서 이질적인 하드웨어, 운영체제, 데이터베이스를 가진 업체간에 서로의 작업에 대한 데이터 동기화가 필요하다. 본 논문에서는 플랫폼 독립적인 데이터 동기화 서버 솔루션인 SyncD2D를 이용한다. SyncD2D는 모바일 클라이언트들을 대상으로 하는 데이터 동기화 서버 시스템이다[4]. 따라서, 다음과 같은 문제점을 가지고 있다.

- 서버 시스템간 동기화 문제를 처리할 수 없다.

이러한 서버간의 데이터 동기화 문제를 해결하기 위해 SyncD2D의 구조를 변경, 확장한다.

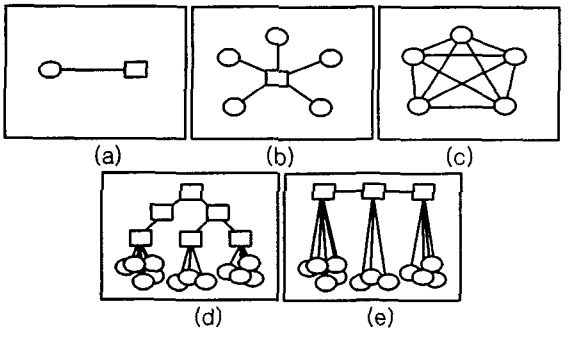
본 연구는 과학기술부, 한국과학재단 지정 인천대학교 동북아 전자물류센터의 지원에 의한 것임.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구로, 첫째, 데이터 동기화 형태들을 살펴보고, 기존 여러 시스템과 통합 시스템의 동시 운용이 가능한 형태를 찾는다. 둘째, SyncD2D에 대하여 살펴본다. 셋째, 서버 분산 운용이 가능하게 해줄 JavaSpaces™에 대해서 알아본다. 3장에서는 SyncD2D를 수정한 SyncDB2DB의 아키텍쳐를 살펴본다. 4장에서는 예를 통해서 SyncDB2DB의 운용 단계를 알아본다. 5장에서는 결론과 향후 과제를 기술한다.

2. 관련연구

2.1 데이터 동기화 형태

[그림 1]은 다양한 데이터 동기화 형태들을 나타내고 있다. 사각형은 서버를, 원은 클라이언트를 나타낸다.



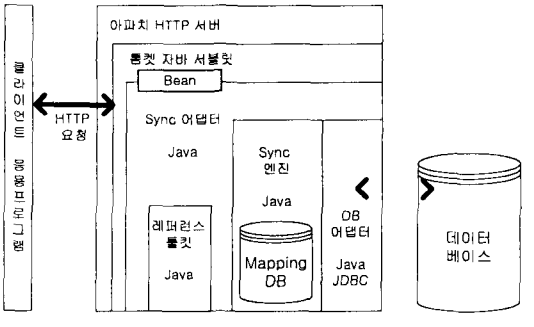
[그림 1] 데이터 동기화 형태

[그림 1]의 (a)는 가장 간단한 동기화 형태로, 하나의 클라이언트와 하나의 서버가 연결된다. 클라이언트에서 수정된 데이터는 서버로 복사되어 보관된다. 즉, 서버는 데이터의 백업 용도로만 이용된다. [그림 1]의 (b)는 일부 상용 시스템에서의 형태이다. 여러 종류의 클라이언트를 가진 사용자가 하나의 서버에 접속하여, 데이터 동기화를 이루는 형태이다. [그림 1]의 (c)는 중앙에 서버가 없고, 클라이언트 각각이 서버의 역할도 수행하도록 되어 있다. 즉, 임의의 클라이언트는 자신의 갱신 정보를 모든 다른 클라이언트에 알려주어야 하며, 또한 다른 클라이언트의 갱신 정보를 얻어 자신의 데이터를 갱신해야 한다. 이러한 구조는 구현하기 어렵게 만들며, 클라이언트 자체가 서버의 역할도 겸하게 됨으로, 운용시 많은 자원을 필요로 하게 된다. [그림 1]의 (d)는 계층형(Hierarchy) 구조로 클라이언트들이 물리적으로 하나의 서버에 연결되어지는 것이 아니라, 작업요청 순서에 따라 임의의 최하단 서버에 연결되어 요청 작업을 처리하게 된다. 이러한 구조는 임의 서버가 정지하더라도 작동중인 다른 서버에서 클라이언트의 요청을 처리할 수 있다는 장점이 있다. [그림 1]의 (e)는 클러스터(Cluster) 구조로 서버레벨과 클라이언트 레벨 두개로 되어있다. 클라이언트는 인증된 하나의 서버와 데이터를 동기([그림 1]의 (b) 형태) 시키며, 다른 서버들에 자신의 데이터 사본을 유지한다. 따라서, 서버들 중 하나가 정지되더라도 다른 서버들의 작동에는 아무런 영향을 주지 않는다[2].

본 논문에서는 각 기업이 기존 전산 시스템을 독자적으로 운용하며, 서로의 데이터베이스 정보를 공유 할 수 있게 하기 위하여 [그림 1]의 5가지 형태 중에 (e) 형태를 선택하도록 한다.

2.2 SyncD2D

SyncD2D는 SyncML과 Java 프로그래밍 언어로 개발된 플랫폼 독립적인 데이터 동기화 서버이다. 임의의 여러 클라이언트는 하나의 서버와 데이터 동기화를 수행한다 ([그림 1]의 (b) 의 형태).



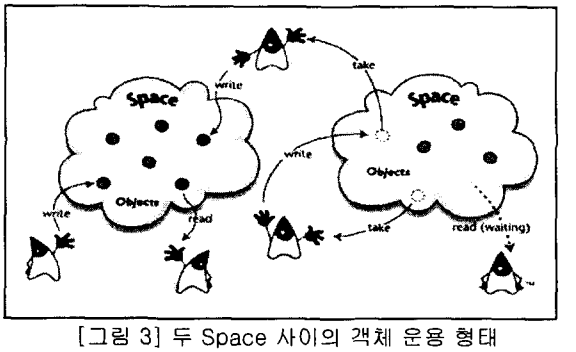
[그림 2] SyncD2D의 아키텍쳐

SyncD2D는 " Sync 어댑터" , " 레퍼런스 툴킷" , " Sync 엔진" , " DB 어댑터" 로 구성된다. Sync 어댑터는 클라이언트의 요청을 수신하며, 처리결과를 클라이언트에 되돌려 보낸다. 레퍼런스 툴킷은 클라이언트에서 수신한 SyncML 메시지를 인코딩(Encoding)하고 디코딩(Decoding)하는 기능과 통신 기능들을 가지고 있는 툴킷이다. Sync 엔진은 디코딩된 데이터를 이용하여 서버의 데이터와 실질적인 데이터 동기화를 수행한다. DB 어댑터는 데이터베이스와의 인터페이스를 수행한다.

SyncD2D 는 하나의 서버 시스템에서만 운용 가능하기 때문에 본 논문의 대용량 데이터 동기화 서버로는 부적합하다.

2.3 JavaSpaces™

JavaSpaces™는 선 마이크로 시스템즈에서 개발되었으며, 분산 애플리케이션을 개발할 수 있도록 해주는 높은 수준의 코디네이션 도구이다. 하나나 그 이상의 Space에 객체(Object)를 넣어 이용할 수 있으며, Space가 객체들의 상태와 프로세스간의 공유 문제를 대신해 준다. 이는 많은 원격 프로세스들간에 공유 메모리(Shared Memory)를 통하여 객체를 사용한다는 의미이다. 프로세스에 의해 Space에 적제된 객체는 제거(take)하기 전까지는 지속(Persistence)된다. 물론, 적제시 일정한 시간이 지나면 소멸되게 설정할 수도 있다[1].



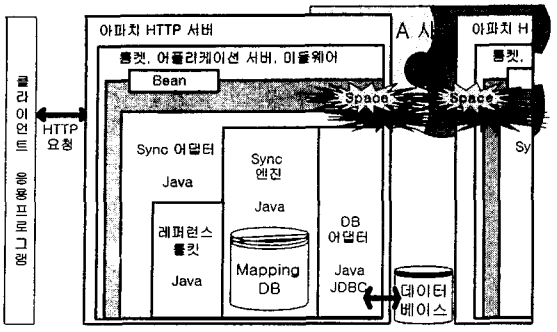
[그림 3] 두 Space 사이의 객체 운용 형태

JavaSpaces™는 JNDI(Java Naming & Directory Interface)를 이용하여 객체를 관리하기 때문에 자원의 물리적인 위치를 몰라도 이용할 수 있다. 또한, Space 내에서 처리된 명령들은 그들의 실행 결과를 살펴 볼 수 있으며, 실패시 실행이전의 상태로 복구할 수 있도록 제공해주는 장점이 있다. 반면, JavaSpaces™의 구현 형태가 중앙 집중적인 클라이언트, 서버 방식으로 되어있다([그림 1]의 (d)). 따라서 (e)의 형태로 구현하기 위해 복제 JavaSpaces™를 이용한다[3].

3. SyncDB2DB의 설계

SyncDB2DB는 SyncD2D 데이터 동기화 모듈에 톰캣 자바 서블릿 서버뿐만 아니라, 어플리케이션 서버에서도 작동될 수 있도록 인터페이스 부분이 변경되었다. 또한 개별적으로 운용되는 SyncD2D에 JavaSpaces™ 레이어를 추가하여, 또 다른 SyncD2D와 클러스터 되도록 하였다.

[그림 4]는 분산된 A사와 B사의 서버 시스템이 통합되

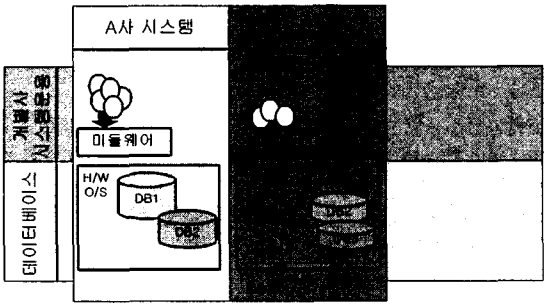


[그림 4] SyncDB2DB의 아키택쳐

어 운용되는 형태이다. 즉, A사와 B사의 서버는 각 사의 클라이언트들의 요청을 처리하며, JavaSpaces™에서 제공하는 read, write, take 등의 메소드를 통하여 두 서버 시스템간에 객체를 공유한다. 이렇게 공유된 객체는 A사와 B사의 SyncDB2DB 동기화 엔진에 의해 시스템 동기화를 이루게 된다.

4. SyncDB2DB의 운용

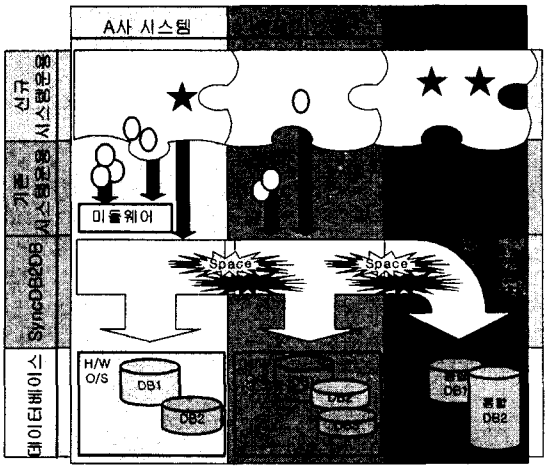
설계된 SyncDB2DB는 통합회사에서 운용할 새로운 시스템을 개발하는 중에도 각 회사의 기존 전산 시스템을 이용 할 수 있어야 한다. 따라서, 본 장에서는 기존 시스템과의 연계 운용이 가능한 단계에서, 새롭게 개발되는 시스템을 적용하는 단계로, 나아가 최종 개발 시스템으로 전이되는 과정을 살펴본다.



[그림 5] 기존 전산 시스템

[그림 5]는 기존 두 회사의 전산 시스템을 나타내고 있으며, " Database" 레이어와 " 개별사 시스템운용" 레이어 두개로 구성되어 있다. 세로 좌측은 A 사의 기존 전산 시스템이며, 클라이언트는 미들웨어를 통하여 데이터베이스에 접근한다. 우측 B사의 기존 전산 시스템은 미들웨어 없이 클라이언트 프로그램에서 직접 데이터베이스에 접근하고 있다. 따라서, [그림 5]와 같이 A사와 B사의 전산 시스템은 구성 뿐만 아니라, 하드웨어, 운영체제, 미들웨어 , 데이터베이스 등의 시스템 이질화가 발생한다.

이러한 이질화를 해결하기 위하여 [그림 6]의 확장된



[그림 6] 확장된 전산 시스템

전산 시스템은 기존 두 레이어 사이에 " SyncDB2DB" 레이어가 추가되었으며, 제일 상단에 " 신규 시스템운용" 레이어가 추가되었다. 또한 통합사에서 운용할 " 통합 전산 시스템" 이 새롭게 추가되었다. 신규 시스템운용 레이어는 통합 시스템이 개발되어감에 따라 기존 시스템의 클라이언트들을 흡수하며, 개발이 완료되면 새로운 클라이언트(★)로 대체되게 된다. 또한 각 사의 데이터베이스는 통합 시스템의 데이터베이스로 통합되게 된다.

5. 결 론

본 논문에서는 SyncD2D를 JavaSpaces™를 이용하여 분산 시스템에서 대용량의 데이터 처리가 가능한 SyncDB2DB를 새롭게 설계하였다. 이렇게 설계된 시스템은 기존 클라이언트를 수정하지 않고 운용가능하며, 새로운 통합 시스템의 개발 이후 새로운 시스템으로 자연스럽게 전이할 수 있다. 향후 연구과제로 설계한 시스템에 대한 구현과 유효성 검증이 필요하다.

참고문헌

[1]" JavaSpaces™ Specification" , Sun 2002.4

[2] Uwe Hansmann.Riku, Riku Mettàlà, Apratim Purakayastha, Peter Thompson, " SyncML Synchronizing and Managing Your Mobile Data" pp. 3 ~ 9, September 2002.

[3] 문남두, 이근웅, 구형서, 박양수, 이명준. "그룹통신 시스템을 이용한 복제 JavaSpace의 설계", 정보과학회, VOL.29 NO.01 pp. 0502 ~ 0504, 2002. 04.

[4] 이민순, 이병수. " SyncML을 이용한 플랫폼 독립적인 동기화 서버의 설계" , 정보처리학회, VOL.09 NO.02 pp. 1337 ~ 1340, 2002.10.