Journal of Korean Institute of Intelligent Systems, Vol. 24, No. 3, June 2014, pp. 239-244
http://dx.doi.org/10.5391/JKIIS.2014.24.3.239

레코드 단위의 동기화를 지원하는 개별 클라우드 구현 기법

Implementations of Record_Level Synchronized Safe Personal Cloud

홍동권

Dong-Kweon Hong

계명대학교 컴퓨터공학과

Computer Engineering Department, Keimyung University

요 약

클라우드 컴퓨팅 (Cloud Computing) 환경의 중요성, 편리성이 점점 커지면서 개인 정보 (프라이버시, privacy)에 대한 염려도 점점 더 커지고 있다. 클라우드 환경에서 개인 정보의 보호에 대한 연구와 노력은 계속 지속되고 있지만 개인 정보의 침해에 대한 염려는 항상 존재하고 있다. 또한 대부분의 클라우드 환경이 제공하는 파일 단위의 단순화된 동기화 기법은 파일의 부분 변경을 어렵게 한다. 특히 파일에 데이터베이스를 생성하는 SQLite와 같은 내장형 DBMS를 사용하는 데이터 중심 앱에서 단순한 파일 단위의 동기화는 많은 정보를 잃어버릴 수 있게 한다. 본 논문에서는 모바일 기기에서 데스크탑 컴퓨터를 이용하여 레코드 단위의 세밀한 동기화를 지원하는 개별 클라우드를 구축하는 기법을 제안하고, 데모 시스템을 구축하여 그 기능을 확인한다. 데모 시스템은 데스크탑 컴퓨터에 RESTFul 웹 서비스 기능을 구현한 후 모바일 기기의 스키마와 동일한 스키마를 클라우드 저장소인 데스크탑 컴퓨터에 구현하고, 낙관적 방식의 레코드 단위의 동기화를 지원하게 한다.

키워드 : 클라우드 컴퓨팅, 웹 서비스, 레코드 단위 동기화, 낙관적 방식 동기화

Abstract

As the usefulness of mobile device is kept growing the privacy of the cloud computing is receiving more attentions. Even though many researches and solutions for privacy matters are suggested we are still worrying about the security problems. In addition most of cloud computing systems uses file-level synchronization which make it difficult to modify a part of a file. If we use data-centric app that stores data on embedded DBMS such as SQLite, a simple synchronization may incur some loss of information. In this paper we propose a solution to build a personal cloud that supports record-level synchronization. And we show a prototype system which uses RESTful web services and the same schema on mobie devices and the cloud storage. Synchronization is achieved by using a kind of optimistic concurrency control.

Key Words : Cloud Computing, Web Services, Record-Level Synchronization, Optimistic Concurrency Control

1. 서 론

아이폰, 안드로이드폰 등의 스마트폰은 데스크톱 컴퓨터 또는 노트북 등의 컴퓨터와는 그 기능과 용도에 많은 차이가 있다. 스마트폰은 일반 PC와 비교했을 때 음성 녹음, 이미지 및 동영상을 촬영하는데 매우 편리하게 하드웨어가 만들어져 있지만 키보드가 따로 존재하지 않아 텍스트 입력이 어려우며 또 작은 디스플레이로 인해 다양한 정보를 한 화면에 구성하는데 불편함이 존재한다. 뿐만 아니라 저장 용량에서도 최근의 스마트폰은 수십 기가 (Giga)의 저장 용량을 가지고 있는 것이 일반적이지만 수백기가에서 테라바이트의 저장 용량을 가지고 있는 PC 또는 서버 컴퓨터에 비하면 저장 용량은 많이 부족하다. 하지만 스마트폰은 저장 용량이 일반 컴퓨터에 비해 상대적으로 부족함에도 불구하고 구조적인 특성으로 인해 입력이 불편한 문자 데이터보다는 용량을 많이 차지하는 음성, 이미지, 동영상 등의 멀티미디어를 더 많이 저장하게 된다. 이런 스마트폰 저장 용량의 불편함을 해소하기 위해 또는 데이터의 백업을 위해 스마트폰 사용자들은 유료 또는 무료 클라우드 기능을 사용하고 있다.

클라우드 기능은 편리하지만 기업, 병원, 공공기관 등의 예민한 정보를 다루는 많은 기관들은 오픈된 공공 클라우드 (Public Clouds) 서비스의 적용이 어려우며, 비록 각 기관 내부 구성원만 사용 가능한 내부 클라우드 (Private Clouds)를 제공한다고 해도 정보의 프라이버시에 대한 염려는 여전히 남아있다 [1].

본 논문에서는 1) 민감한 데이터가 많이 존재하여 클라우드

접수일자: 2014년 2월 19일
심사(수정)일자: 2014년 3월 19일
게재확정일자 : 2014년 3월 20일

† Corresponding author

서비스의 사용이 부담되거나, 2) 경제적, 문화적 제약으로 인하여 좀 더 폐쇄적인 방식으로 스마트폰의 데이터가 관리되어야 하거나, 3) 기존 서버 데이터베이스 중심의 데이터 관리 환경을 계속 유지하면서 스마트폰의 데이터를 기존의 컴퓨팅 환경과 안전하게 공유가 필요한 경우를 위하여 개인용 PC를 이용한 "개별 클라우드 (personal cloud)" 구현 기법을 제안하고, 데모 시스템을 구현하여 그 기능을 확인한다. 본 연구에서 데모 시스템은 지금 현재 가장 널리 사용되고 있는 환경을 고려하여 애플 아이폰과 오라클 데이터베이스 서버를 사용하는 윈도우 PC 응용 프로그램에서 REST (Representational State Transfer) 웹 서비스를 사용하여 개발한다.

2. 관련 연구

NIST (National Institute of Standards and Technology)의 정의에 따르면 클라우드 컴퓨팅은 컴퓨팅 자원 (네트워크, 서버, 저장소, 응용 프로그램, 서비스 등)의 풀에 언제 어디서든지 편리하게 복잡한 절차 없이 쉽게 접근할 수 있게 하는 모델이며, 5개의 중요한 특성과 3개의 서비스 모델 (service Model), 4개의 배치 모델 (deployment model)로 표현된다[1]. 4개의 배치 모델은 public 클라우드, private 클라우드, community 클라우드, hybrid 클라우드로 표현되며, 클라우드의 서비스 모델은 클라우드 환경에서 제공되는 응용 프로그램을 제공하는 Software as a Service (SaaS), 프로그래밍 언어, 라이브러리, 도구 등을 제공하는 Platform as a Service (PaaS), 저장장치와 네트워크 등의 컴퓨팅 자원을 제공하는 Infrastructure as a Service (IaaS)로 분류된다[1, 2, 3, 4].

내부 구성원들만 사용하는 private cloud와는 달리 개별적으로 사용하는 개별 클라우드의 [5] 개념은 최근에 널리 사용되기 시작했으며 가트너 리서치에 의하면 2014년에 개별 클라우드가 개인용 컴퓨터를 대체할 것으로 예측하고 있다. Seagate사의 Personal Cloud [6]는 Personal Cloud Storage의 개념으로 데스크탑, 태블릿, 모바일 기기에서 인터넷으로 쉽게 접속 가능한 저장 공간의 의미를 가진다.

2.1 이기종간의 데이터 전송 방법

스마트폰과 개별 클라우드 사이의 파일 전송 방법은 전용 케이블을 사용하는 방법과 와이파이를 사용하는 방법들이 제공되고 있다. 본 연구에서는 스마트폰과 개별 클라우드 사이의 파일 전송은 1) 와이파이가 제공되는 곳에서는 웹 서비스를 이용하고, 2) 와이파이가 제공되지 않을 경우 전용 케이블로 데이터를 전달하는 방법을 사용한다. 샌드박스 (Sandbox) 메커니즘을 사용하는 아이폰에서 전용 케이블을 이용한 파일 전송 방법은 개별 클라우드 쪽에 iTunes가 설치되어 있어야 한다. 그리고 아이폰 앱의 plist 파일에서 "Application supports iTunes file sharing" 항목을 YES로 한다. (즉 UIFileSharingEnabled 선언) 해당 앱에서 PC와 공유하고 싶은 파일을 아이폰의 Documents 폴더에 저장 (read, write가 모두 가능하게 하기 위해서)하면 된다. 예를 들어 Documents 폴더에 BackupFolder를 만들고 공유할 파일들을 BackupFolder에 저장한 후 폴더를 PC로 옮기는 방법을 사용하면 한번에 폴더에 있는 모든 파일을 옮길 수 있다. 다음의 [그림 1]과 같이 앱을 선택한 후 "다음으로 저장"을 사용하면 아이폰의 폴더를 PC로 옮길 수 있다.

그림 1. iTunes로 확인한 백업 폴더
Fig. 1. Backup folder on iTunes

2.2 이기종간의 데이터 호환성

본 논문에서는 파일 내부의 내용을 레코드로 사용하므로 그 내용을 분석할 수 있는 형태로 데이터가 전달되어야 한다. 대부분의 DBMS는 텍스트 데이터를 XML (Extensible Markup Language), JSON (JavaScript Object Notation), CSV로 익스포트, 임포트하는 기능을 제공한다. 따라서 본 논문에서 데이터 컨텐츠는 XML 또는 JSON으로 생성하여 데이터를 서로 전달하게 한다. 하지만 데이터베이스에 연결되어 있는 음성, 이미지와 같은 멀티미디어는 이기종간에 쉽게 플레이할 수 있는 형식을 사용한다. 본 논문에서는 이미지는 PNG 형식을, 동영상 및 음성은 MPEG4의 형식을 사용하여 스마트폰과 개별 클라우드로 사용하는 PC에서도 쉽게 플레이할 수 있게 한다.

3. 레코드 단위의 개별 클라우드를 위한 동시성 제어

본 논문에서 사용하는 개별 클라우드는 개별적인 저장 공간을 사용하며, 레코드 단위의 동시성을 지원한다. 레코드 단위의 데이터 일치성을 위해서는 모바일 기기와 개별 클라우드 사이에 데이터의 동시성 제어가 필요하다. DBMS에서 사용하는 동시성 제어 방법은 잠금 (locking) 방식을 사용하지만 모바일 기기의 데이터와 클라우드 사이에 잠금 정보를 유지하는 것은 매우 비효율적이다[7]. 따라서 본 논문에서는 낙관적 동시성 방법[7]을 적용하여 레코드 단위의 동시성을 제공한다.

3.1 SQLite primary key 선정 방법

레코드 단위의 동시성을 위해서는 각 레코드를 식별할 수 있는 키가 필요하다. 따라서 모바일 기기의 앱 스키마에서는 각 레코드를 고유하게 식별하기 위하여 키를 부여한다. 본 논문에서는 2가지의 방법을 고려한다.

1) 시퀀스 번호만을 사용하는 방법
2) UDID (Unique Device Identifier)와 시퀀스 번호를 결합하여 사용하는 방법

시퀀스 번호 (sequence number)를 키로 사용하는 것은 모바일 기기의 로컬 데이터베이스에서는 데이터의 고유성을 보장할 수 있기 때문에 올바른 선택일 수 있지만 클라우드의 데이터베이스와 같이 사용될 때는 동일한 레코드인지를

검사하는데 적절히 사용되지는 못한다. 즉 스마트폰의 로컬 DB에서 사용하는 시퀀스 번호와 클라우드에 있는 데이터베이스의 시퀀스 번호가 같아도 서로 동일한 데이터인지에 대한 확실성이 없으므로 데이터의 다른 컬럼을 다시 비교하는 과정이 필요하게 된다. 하지만 2)번 방법은 각 장비의 고유한 번호와 시퀀스 번호를 결합하여 사용하므로 항상 고유한 번호를 가질 수 있다.

3.2 이기종 다중 모바일 기기 확장 지원

비록 개별 클라우드를 사용하지만 각 개인이 항상 1대의 모바일 기기만을 사용하는 것은 아니다. 본 논문에서는 [그림 2]와 같이 각 개인이 이기종 다중 모바일 기기를 사용하는 경우에도 UDID와 시퀀스를 결합하여 레코드의 키를 사용하는 방법을 사용하여 그 문제점을 해결한다.

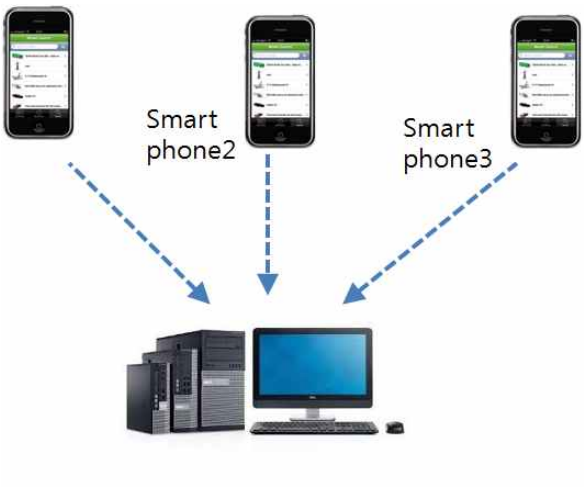

그림 2. 이기종 다중 모바일 기기의 지원
Fig. 2. Support of multiple mobile devices

UDID를 사용하지 않을 경우 다중 스마트폰 (다른 종류 즉 아이폰, 구글폰, 윈도폰에 대한 제약이 없다)의 경우 각 스마트폰에서 생성한 키 값의 중복 현상이 발생할 수 있다. 이 문제를 해결하기 위하여 각 스마트폰의 고유한 값을 추가하는 방법을 사용한다. 스마트폰의 고유한 값으로는 길이가 40인 문자열로 구성된 스마트폰의 고유한 시리얼 넘버 UDID (Unique Device Identifier)를 사용해왔지만 2012년부터 개인정보의 논란에 따라 UDID는 더 이상 사용하지 못하게 되었다. 대신 애플에서는 UDID값이 아닌 UUID(Universally Unique Identifiers, GUID Globally Unique Identifier))를 이용하라는 권고를 내렸다. UUID는 128bit 값으로 생성된 유일값이며 시간과 공간값으로 생성된 유일값이므로 중복될 염려가 없다.

iOS5에서 유일값 UUID 생성 방법은 다음과 같다.

```
CFUUIDRef uuid = CFUUIDCreate(NULL);
NSString *UUID = CFUUIDCreateString(NULL, uuid);
```

또 다른 방법으로 iOS6에는 다음의 방법을 사용한다.

```
NSString *UUID = [[NSUUID UUID] UUIDString];
```

그림 3. iOS6의 UUID
Fig. 3. UUID on iOS6

UUID 값은 128비트로 구성되어 있지만 스트링으로 변환하면 32개의 문자와 4개의 하이픈 (36개)으로 표시된다. 데이터베이스에서 primary key가 길면 키 값의 색인에 사용되는 B-tree의 fan out이 작아져서 B-tree의 높이가 높아지게 되며, 이는 검색 성능을 떨어뜨리는 원인이 된다. UUID 값은 128비트이지만 문자열로 변환하면 36개의 문자열로 구성되어 있으므로 다음의 [표 1]과 같이 서버에 UUID 테이블을 유지하는 것이 매우 효과적이다. 변환 테이블은 각 UUID에 대해서 길이가 짧은 정수 ID를 반환하도록 한다.

표 1. UUID 변환 테이블
Table 1. UUID Conversion Table

UUID	sid
10C57A5A-9A7E-441E-8BE1-27E8D926230B	1
2C938711-C4E9-4058-8D20-8EA1475A5C5B	2
.....	..
.....	..

3.3 모바일 기기에서의 낙관적 동시성 제어 (optimistic concurrency control)

모바일 기기에서 새로 입력된 레코드는 나중에 클라우드에 저장을 요청할 때 클라우드에 같은 키값을 가진 레코드가 존재하지 않으므로 클라우드에 새로운 레코드로 추가된다. 서로 다른 모바일 기기에서 생성한 레코드의 키 값은 고유하므로 새로운 레코드의 입력 시 충돌은 발생하지 않는다. 하지만 클라우드에서 모바일 기기로 읽어온 후 변경된 레코드의 저장은 동시성 제어에 의해 관리된다. 본 논문에서 모든 변경 (update) 트랜잭션은 단일 연산으로 이루어지며 이때 낙관적 동시성 제어는 각 레코드에 대하여 다음의 4단계로 진행한다[7].

<변경 트랜잭션 동시성 제어>

1단계 Begin:
클라우드에서 모바일 기기로 레코드를 읽어올 때 클라우드 레코드의 타임스탬프를 복사하여 읽어온다.
2단계 Modify:
클라우드의 레코드를 읽어와서 모바일 기기에서 레코드를 변경한다.
3단계 Validate:
모바일 기기에서 변경된 레코드를 클라우드로 다시 저장할 때 충돌이 있는지를 검사한다. (비록 같은 사용자이지만 서로 다른 모바일 기기를 사용하는 경우에 충돌이 발생할 수 있다). 이때 비교 대상이 되는 레코드의 식별은 각 레코드의 키 값을 사용한다. 같은 키 값을 가진 레코드가 존재할 경우 레코드의 타임스탬프를 비교하여 클라우드에 있는 타임스탬프가 더 크면 충돌이 발생한 것으로 판단한다.
4단계 Commit/Rollback:
만약 충돌이 없으면 모든 변경을 유효하게 하며, 이때 클라우드의 레코드의 새로운 타임스탬프를 부여한다. 만약 충돌이 발생하면 그 변경을 취소하며, 클라우드 레코드의 타임스탬프는 변화되지 않는다.

클라우드에 생성된 레코드는 초기 타임스탬프를 가진다. 이후 클라우드의 레코드는 변경이 발생될 때만 새로운 타임스탬프를 부여 받는다. 따라서 다음의 [그림 4]에서 2개의 모바일 기기에서 읽어온 레코드의 타임스탬프는 클라우드에 있는 동일 키 값을 가진 레코드의 타임스탬프와 같은 값을 가진다. 이때 2개 중에서 먼저 변경을 update하는 레코드의 타임스탬프는 같은 값을 가지므로 (ts1은 데이터의 타임스탬프) validation phase를 거쳐 Commit을 하게 되어 그 타임스탬프가 새로운 값인 ts3으로 변경된다.

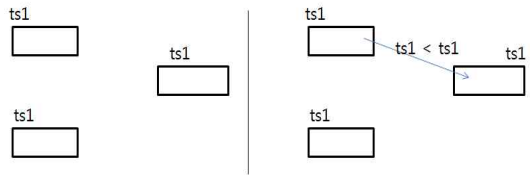

그림 4. 변경되지 않은 데이터의 update
Fig. 4. Update on unmodified data

[그림 5]에서는 클라우드의 데이터가 이미 변경이 되었으므로 모바일 기기의 데이터는 ts1 < ts3가 되어 validation phase에서 Rollback을 호출하게 된다.

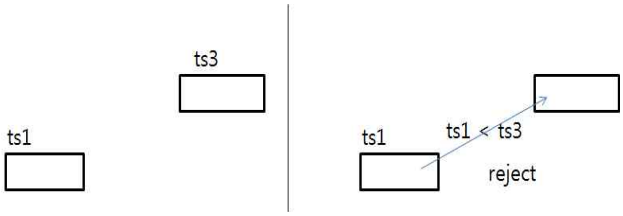

그림 5. 변경된 데이터에 update 시도 (validation 실패)
Fig. 5. update on modified data (validation fails)

4. 시스템 구현

데모 시스템은 모바일 기기로 애플의 아이폰과 아이패드를 사용하며, SOA 형식의 구조를 위하여 웹 서비스를 제공하기 위한 시스템은 Windows 7 데스크탑 컴퓨터에서 IIS (Internet Information Services)와 WCF (Windows Communication Foundations)를 사용하였다[8, 9, 10, 11]. 개별 클라우드 공간은 같은 데스크탑 공간에서 제공해도 되지만 본 논문에서는 Linux 서버에 설치되어 있는 오라클 서버의 공간을 사용하였다.
본 논문에서 모바일 기기의 응용 프로그램은 SQLite를 사용하는 데이터베이스 응용 앱이다.
데이터의 종류는 데이터베이스에 직접 저장되는 텍스트 데이터와 데이터베이스에 직접 저장되지 않고 파일로 저장되는 다양한 종류의 멀티미디어 데이터를 사용하였다. 파일로 저장되는 멀티미디어 데이터는 데이터베이스에 그 파일의 경로를 저장하는 방식을 사용하였다. 본 논문에서는 아이폰에서만 시험하였지만 안드로이드폰으로의 확장도 쉽게 가능하다.

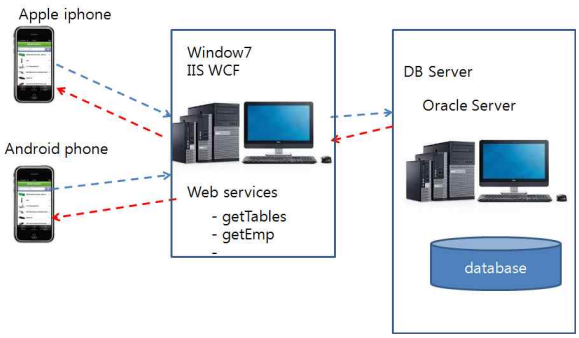

그림 6. 개별 클라우드 시스템 구성도
Fig. 6. Structures of personal cloud

4.1 웹 서비스 공급자

웹 서비스는 Windows 7에서 MS의 WCF 기능을 이용하여 구현하였다[8]. WCF는 SOAP과 REST 방식 모두를 지원하는데 본 논문은 REST 방식을 사용하여 웹 서비스를 구현하였다. REST 방식에서 클라우드와 레코드를 주고 받는 형식의 예제 코드는 다음과 같다. 모바일 기기들이 레코드를 클라우드에서 읽어오기 위해서는 다음과 같이 WebInvoke GET 메소드를 사용하는 웹 서비스를 사용한다.
반면에 REST 방식에서 모바일 기기에서 데이터를 클라우드로 보내기 위해서는 POST 메소드를 사용하는 웹 서비스를 사용한다. 정의된 웹 서비스 인터페이스는 사용자에게 알려져야 하며 사용자는 정의된 웹 서비스 인터페이스만을 활용하여 웹 서비스를 소비하게 된다.

```
// 웹 서비스 인터페이스 정의
[ServiceContract]
    public interface IGetEmployees
    {
      [OperationContract]
      //attribute for returning JSON format
      [WebInvoke(Method = "GET",
      ResponseFormat = WebMessageFormat.Json,
      BodyStyle = WebMessageBodyStyle.Wrapped,
      UriTemplate = "json/employees")]
      //method
      List<Employee> GetAllEmployeesMethod();
    }
```

```
// 모바일에서 POST로 전송 받은 JSON 데이터를
//오라클 DB에 저장 인터페이스 구현
public string PushEmployeeMethod(Stream data)
{
      List<Employee> yourlist = new
                          List<Employee>();
      StreamReader reader = new StreamReader(data);
      string jsonstring = reader.ReadToEnd();
      // DataContractJsonSerializer 클래스를 이용
      using (MemoryStream stream = new
      MemoryStream(Encoding.Unicode.GetBytes(json
                                              string)))
      {
          ....
          // JSON 데이터를 데이터베이스에 저장
          using (OracleConnection conn =
            new OracleConnection
            ("DATA SOURCE=ORA9;
                ... PASSWORD =xxxx"))
            {

}
```

따라서 사용자에게 숨겨져 있지만 웹 서비스 공급자는 인터페이스를 적절히 구현하여야 하며 구현된 인터페이스의 예제는 앞과 같다.

4.2 웹 서비스 소비자

본 논문에서는 Windows 7에 앞에서 구현한 웹 서비스를 네트웍 연결이 가능한 환경에서 다양한 모바일 기기를 사용하여 클라우드와 레코드를 보내고 받을 때 사용한다. 다음의 코드는 애플의 아이폰에서 http://210.125.31.156에 구현되어 있는 PushEmployeeMethod를 사용하여 클라우드로 레코드를 보내는 예제이다. 웹 서비스 코드는 스마트폰에서 전달받은 데이터를 낙관적 동시성 제어 방법을 거친 후 PC의 응용 프로그램에서 서버 데이터베이스에 저장하고 관리한다. 이때 앱의 관점에서 보면 앱의 데이터는 클라우드 데이터의 부분집합이 되며, 앱의 기능은 일부분의 데이터에 대한 사용자이며 동시에 데이터의 수집자의 역할을 하게 된다. 반면 클라우드에 의해 관리되는 데이터는 앱에서 수집한 정보의 전체를 보관하고 있으며, 클라우드 역할을 하는 PC의 프로그램은 전체 데이터에 대한 관리와 분석 기능도 할 수 있다.

```
  // 웹 서비스 소비자
...
NSString *request =
[NSString stringWithFormat:
"http://210.125.31.156/getEmps/Service1.svc
 /PushEmployeeMethod"];
NSURL *URL = [NSURL URLWithString:
[request
stringByAddingPercentEscapesUsingEncoding:
                              NSUTF8StringEncoding]];
NSMutableURLRequest *URLrequest =
[NSMutableURLRequest requestWithURL:URL];
    [URLrequest setHTTPMethod:@"POST"];

// WCF로 POSTING 하는 데이터를 여기에,
// NSData 형식으로
[URLrequest setHTTPBody:self.postingData];
....
```

5. 결론 및 향후 연구 방향

모바일 기기의 대중화에 따라 모바일 기기에 저장되어 있는 데이터의 안전한 저장과 동기화에 대한 요구가 점점 커지고 있다. 아마존, 구글과 같은 글로벌 클라우드 제공 기관들이 편리한 기능을 제공하고 있지만 각 개인의 민감 정보와 데이터를 제 3자에게 맡기는 것을 부담스러워 하는 것도 사실이다. 따라서 최근 개별 클라우드에 대한 관심이 점점 커지며 앞으로 개인용 저장장치를 사용하듯이 각 개인이 개별 클라우드를 사용할 것으로 예측되고 있다.

본 논문은 스마트폰으로 대표되는 모바일 기기를 위한 레코드 단위의 동기화를 지원하는 개별 클라우드를 각 개인의 데스크탑 컴퓨터를 활용하여 구현하는 기법을 제안하였다. 각 개인이 관리하는 개별 클라우드는 파일 단위의 동기화만을 지원하는 public, private cloud에 비하여 1) 개인의 민감 데이터에 대한 안전성 보장과, 2) 레코드 단위의 세밀한 동기화를 지원하는 큰 장점이 있다. 또한 클라우드에 보관된 데이터에 대한 개인화된 새로운 기능의 추가가 쉬워진다. 본 논문에서 제안한 개별 클라우드 기능은 개별 클라우드 저장소 (personal cloud storage)의 개념을 더 확장했다. 따라서 각 개인이 보유하고 있는 데스크탑 컴퓨터에 (본 논문에서는 개인용 컴퓨터 (http://210.125.31.156) 구축) RESTFul 웹 서비스 기능을 구현하고, 모바일 기기의 스키마와 동일한 스키마를 클라우드 저장소인 데스크탑 컴퓨터에 구현한 후 모바일 기기와 클라우드 사이에 낙관적 방식의 레코드 단위의 동기화를 지원하게 하였다.

낙관적 방식과 잠금 방식의 성능 비교는 데이터베이스 분야에서 많이 연구되어 왔다. 데이터의 변경이 자주 일어나지 않는 상황에서 낙관적 방식의 성능은 잠금 방식의 성능보다 더 우수하므로 본 연구에서도 개별 클라우드의 특성을 고려하여 낙관적 방법을 사용하였다.

본 논문의 개별 클라우드 구현 기법은 개인의 데스크탑 컴퓨터를 활용하여 구현되므로 public 또는 private cloud에서 발생할 수 있는 프라이버시 염려를 완화시킬 수 있다.

본 논문의 결과는 앞으로 사물 인터넷 (Internet of Things)의 활성화와 함께 개인의 데이터, 또는 가족의 데이터, 한 가정의 기기 데이터 등 개별적으로 안전하게 유지되어야 하는 데이터의 관리로 확장하여 사용될 수 있다.

References

[1] Peter Mell, Timothy Grance, "NIST Definition of Cloud Computing". *National Institute of Standards and Technology. NIST Special Publication* 800-145, Sep 2011.

[2] Michael Armbrust at el, "A View of Cloud Computing", *Communications of ACM*, Vol 53, No.4. April 2010.

[3] A rmbrust, M., et al., "Above the clouds: A Berkeley view of cloud computing". *Tech. Rep. UCB/EECS-2009-28, EECS Department, U.C. Berkeley,* Feb 2009.

[4] Sanjay P.Ahuja, Alan C. Rolli, "Survey of the State-of-the-Art of Cloud Computing, *Internation Journal of Cloud Applications and Computing*, Volume 1, Issue 4, October 2011.

[5] P. Sasikala, "Cloud Computing and E-Gorvanance: Advances, Opportunities and Challenges", *Internation Journal of Cloud Applications and Computing,* Volume 2, Issue 4, October 2012.

[6] "Seagate Personal cloud", http://www.seagate.com.

[7] Luiz Alexandre, Silva Maciel, Celso Hirata, "A timestamp-based two phase commit protocol for web services using rest architectural style", *Journal of Web Engineering,* Volume 9, Issue 3, pp 266-282 September 2010.

[8] Cott Klein., *Professional WCF Programming. NET Development with the Windows Communication Foundation,* Wiley publishing, pp. 47-58.

[9] Thomas Erl, *Service-Oriented Architecture (SOA)* ISBN 0-13-185858-0, Prentice Hall, 2005.

[10] "Amazon Web Services. TC3 Health Case Study", http://aws.amazon.com/solutions/case-studies/tc3-health/.

[11] Haitao Du, Bo Zhanf, Dingfang Chen, "design and actualization of SOA-based data mining system" in *9th Conference on Computer-Aided industrial Design and Conceptual Design*, 2008,

저 자 소 개

홍동권(Dong-Kweon Hong)

1985년:경북대학교 전자과 공학사.
1992년: U. of Florida 컴퓨터공학 석사
1995년: U. of Florida 컴퓨터공학 박사
1997년～현재 계명대학교 컴퓨터공학과 교수

관심분야 : 데이터베이스, 데이터 마이닝, 머신 러닝
Phone : +82-53-580-5281
Fax : +82-53-580-5165
E-mail : dkhong@kmu.ac.kr