2001년 한국정보처리학회 춘계 학술발표논문집 제8권 제1호

분산 멀티미디어 스트림 상의 멀티 미디어 메일을 위한 분산 메일 서버 설계 및 구현

박용희*, 전성미*, 고희선*, 임영환**
*숭실대학교 대학원 컴퓨터학부
**숭실대학교 대학원 미디어학부
e-mail : mirinae@media.soongsil.ac.kr

A Design and Implementation of Distributed Mail Server for Multimedia Mail on Distributed Multimedia Streams

Yong-Hee Park*, Sung-Mi Chon*, Hee-Seon Ko*, Young-Hwan Lim**
*School of Computing, Soongsil University
**School of Media, Soongsil University

요 약

기존의 멀티미디어 메일은 대용량 메일이라 개인 사용자에게 그 부담이 크지 않을 수가 없었다. 본 연구에서는 멀티미디어 스트림엔진인 Essence 를 이용하여 동영상 멀티미디어 메일을 전송하는데 필요한 Essence 내에 데이터베이스를 지원하기 위한 DBMedium 개발과 데이터베이스 라이브러리를 개발하기 위한 일련의 과정을 제안하고 있다.

1. 서론

최근 컴퓨터와 통신 기술의 발달과 더불어 영상 및 비디오, 오디오 등을 중심으로 한 멀티미디어 정보 서비스에 대한 요구가 증가하고 있다. 뿐만 아니라 컴퓨터와 관련된 하드웨어와 소프트웨어 기술의 발전으로 음성, 문자, 영상 등의 다양한 정보를 일반 사용자에게 제공하는 것이 가능하게 되었다. 그 중에서 일반인에게 가장 친숙한 멀티미디어 응용으로는 VOD, 화상회의, 원격진료 및 원격 교육 등이 있는데, 여기서는 멀티미디어 메일이라는 새로운 멀티미디어 서비스를 제공하고 있다. 인터넷상의 동영상 멀티미디어 메일은 동영상을 포함한 멀티미디어 연출 프로그램을 프리젠테이션 편집기로 작성하여 인터넷의 이메일로 첨부해서 보내면 받은 연출 프로그램을 실시간으로 수행해 볼 수 있는 메일이다. 즉, 송신자는 동영상이 포함된 멀티미디어 메일의 내용을 VIP(Visual Interface Player)라는 멀티미디어 연출 편집기로 작성하여 인터넷에서 기존에 사용되고 있는 이메일(e-mail)에 첨부해서 보내면 수신자는 일반 이메일을 통하여 받은 것을 두 번 클릭하여 "Open"하거나 "Play"하면 멀티미디어 연출 프로그램이 수행하면서 동영상 메일을 실시간으로 연출해 볼 수 있도록 하는 것이다. 동영상이 포함된 멀티미디어 메일과 관련된 기술은 많이 개발된 상태이나 다음과 같은 문제점 때문에 저속망의 인터넷에서 실용화가 되어 있지 못하다. 첫 번째로 멀티미디어 이메일은 그 데이터 자체가 대용량이라 개인 PC 사용자에게 있어서 대용량의 하드디스크 공간을 요구하게 된다. 이를 위해 대용량 멀티미디어 메일서버가 필요하게 되었다. 본 연구에서는 서버급으로 NT 급으로 멀티미디어 데이터베이스로 한국과학기술원에서 개발한 '오딧세우스'를 이용하여 멀티미디어 데이터를 저장할 것이다. 두 번째로 대용량의 멀티미디어 파일을 어떻게 보내는가에 있다. 이는 VIP 라는 멀티미디어 편집기를 이용하여 프리젠테이션 파일(.prs 라는 확장자로)로 만들어 프리젠테이션 정보만을 가지는 파일로서 실제 데이터의 크기는 1KB 정도만을 차지하게 된다. 실제 대용량의 데이터는 보내는 송신자의 PC 에 존재하기 때문에 전송하는데 불편함이 없다. 세 번째로 송신자가 보낸 멀티미디어 메일을 열어 프리젠테이션 파일(.prs)을 실행시키면 항상 소스가 되는 멀티미디어 데이터가 존재하는 컴퓨터는 항상 On-line 상태로 존재해야 한다. 개인 컴퓨터 사용자일 경우에는 이런 점이 불편함이 아닐 수가 없다. 이에 앞에 언급하였듯이 On-Line 상태로 작동할 수 있는 메일 서버가 마찬가지로 필요하게 되었다. 이에 기본적으로 멀티미디어 스트림 엔진인 Essence 를 이용하여 멀티미디어 데이터를 전송하고 앞에서 언급한 멀티미디어 DBMS 인 '오딧세우스'를 이용하여 Essence 에서 오딧세우스를 이용하여 멀티미디어 데이터를 저장하거나 질의문

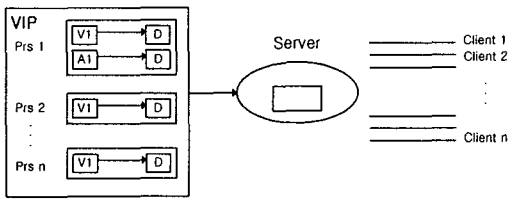
본 논문의 연구과제는 첨단기술정보센터의 지원을 받은 것입니다.

을 실행하여 결과 값을 받을 수 있는 Essence DBMS Medium 구현과 Essence 에서 Database 에 관하여 모듈화하여 라이브러리로 구현하여 멀티미디어 데이터를 실시간으로 데이터베이스에 저장하도록 하겠다.

2. 시스템 요구사항

2.1. 전체적 기능에 대한 요구사항

본 시스템은 멀티미디어 스트림엔진(Essence)을 기반으로 하여 멀티미디어편집기를 이용하여 멀티미디어 메일을 작성하여 데이터베이스인 '오딧세우스'에 저장하도록 한다.



<그림 1>전체 시스템

첫째, 비디오 파일이나 Prs 파일이 여러 개 있다고 가정했을 때 본 시스템은 다음과 같은 가정을 설정해 보자.

Vi : 중복, Prs i: 중복 $1 \le i \le n$.

첫째, Prs i 를 연출할 때, $1 \le i \le n$. 예를 들어 홍보물인 경우, 극단적인 경우 특정 Prs 파일 k 를 모든 클라이언트가 연출할 때, 또는 Prs 파일이 전부 다를 경우, 이를 만족해야 한다.

둘째, 서비스의 타입과 재생되는 시간에 따른 QoS 를 만족해야 한다. 만약 Live 이면 1 개의 실제 데이터를 멀티캐스팅(multicasting)기술이 요구되며 재생되는 시간차를 극복해야 한다. 이미 연출되어진 이전의 것은 보이지 않는다. On-Demand 인 경우 재생시간은 처음부터 보여지게 된다. 그리고 자료 검색을 할 수 있는 인터페이스가 요구되어 진다.

위와 같은 조건을 만족하기 위해 아래와 같은 요구사항을 만족해야 한다.

첫째, 연속된 멀티미디어 데이터의 저장이 가능해야 한다. 이미 저장된 멀티미디어 데이터의 저장뿐만 아니라 실시간으로 생성되는 멀티미디어 데이터의 저장이 가능해야 한다.

둘째, 특정 프리젠테이션 파일을 복수의 클라이언트가 연결하여 재생 시 QoS 를 만족해야 한다.

셋째, 복수의 프리젠테이션 파일을 복수의 클라이언트가 연결하여 재생 시 QoS 를 만족해야 한다.

넷째, 클라이언트 수 증가에 따른 시스템의 성능을 만족해야 한다.

다섯째, 인터넷기반이 아닌 어플리케이션 기반에서 로컬 PC 의 파일을 원격지 서버에 저장할 수 있는 Upload 기능이 있어야 한다.

여섯째, On-Demand 서비스 경우를 위해 자료 검색 기능이 지원해야 한다.

일곱째, 원본 멀티미디어 데이터 파일을 logical 하게 나눌(clipping) 수 있는 기능이 있어야 한다. 이는 다음 2 차버전에서 구현할 예정이고, 지금 설계준비 중에 있다.

3. 설계 및 구현

앞으로의 개발방향은 멀티미디어 편집기인 VIP 를 이용하여 멀티미디어 스트림을 구성하여 연출 데이터를 '오디세우스'를 이용하여 대용량 멀티미디어 데이터를 저장하고 한다. 이를 위해 개발방향은 첫째, Essence 에 Database 에 대한 정보를 가지는 미디움을 구현해야 한다. Essence 는 실시간으로 멀티미디어 데이터를 전송해 주는 스트림 엔진으로 데이터베이스에 대한 정보를 지원해주는 모듈이 아직 없다. 이 모듈을 스트림으로 구성될 수 있는 부분을 미디움으로 제작해야 한다. 둘째, Essence 에서 스트림으로 구성이 안 되는 질의문에 대하여 라이브러리로 모듈화 하여 Essence 가 데이터베이스를 동작하게끔 지원할 수 있는 라이브러리를 구현해야 한다. 다음은 미디움에서 지원하는 Essence API 를 정의해 보았다.

3.1. DBMedium API

int DBMedium::Open();
데이터베이스에 대한 초기화 단계와 재생하기 전의 준비과정 단계이다.

int DBMedium::Play();
실제 스트림의 데이터를 가져와 데이터 베이스에 저장하거나 데이터베이스에 있는 데이터를 읽어오는 일을 수행한다.

int DBMedium::Stop();
수행 중이든 일을 멈춰 데이터베이스의 트랜젝션(Transaction)을 멈추는 단계이다.

int DBMedium::Close();
스트림의 Close 를 수행하게 되면, DBMedium 의 Close()는 데이터베이스의 트랜젝션을 정상적으로 종료시키고 데이터베이스를 디스마운트 시킨다. 데이터베이스를 정상적으로 종료 시켜줘야 수행했던 일이 정상적으로 데이터베이스에 반영된다.

3.2. DBManager API

void DBManager::SetUserid(char* pstrUserid);
사용자의 계정을 지정할 때 호출한다.

void DBManager::SetUserName(char* pstrUserName);
사용자의 이름을 지정할 때 호출한다.

void DBManager::SetPassword(char* pstrPassword);
사용자의 비밀번호를 지정할 때 호출한다.

void DBManager::SetDBName(char* pstrDatabase);
데이터베이스의 이름을 지정할 때 호출한다.

void DBManager::SetFileName(char* pFilename);
파일 이름을 지정할 때 호출한다.

void DBManager::SetValidTerm(char* pValidTerm);
유효기간을 설정할 때 지정한다.

void DBManager::SetFileDescription(char* pFiledescription);
파일에 대한 설명을 지정할 때 호출한다.

void DBManager::SetProtection(char* pstrProtection);
파일에 대한 사용자 권한을 설정할 때 호출한다.

void DBManager::SetFindMode(char* pfindmode, char* pkeyword);
Find 시 Find 의 모드 설정과 키워드를 지정할 때 호출한다. 모드 값은 만든이 중심으로 찾을 때는 "USERID"값을 파일 이름으로 찾을 때는 "FILENAME"값을 지정하면 된다. 키워드 값은 찾고자 하는 키워드를 입력하면 된다.
void DBManager::Register();
사용자 등록 시 호출된다.
void DBManager::Logon();
사용자 접속 시 호출된다.
void DBManager::DeleteFileName();
해당 파일을 삭제할 때 호출된다.
FilelistStruct* DBManager::Find();
데이터베이스에서 해당 키워드에 대해서 찾기 할 때 호출한다.
FilelistStruct* DBManager::ShowList();
데이터베이스에 저장된 데이터의 리스트를 보여줄 때, 호출된다.
int DBManager::InitializeDataBase();
데이터베이스를 초기화 할 때 호출된다.
int DBManager::TerminateDataBase();
데이터베이스를 종료할 때 호출된다.
int DBManager::GetNumResultRows();
Find 나 ShowList 시 질의 결과를 구성하는 열의 개수를 리턴한다.

3.3. 알고리즘

데이터베이스에 대한 기능 중에 Find 나 ShowList 인 경우는 질의 결과 DataStruct 라는 구조체의 형태로 값을 리턴 해야 한다. 스트림 엔진인 Essence 에서는 모든 데이터의 자료 타입을 Xdr 이라는 프로토콜에 미리 정의가 되어야 한다. 기본적인 데이터타입에 대해서는 정의가 되어있지만, 가변적으로 변할 수 있는 구조체에 대해서는 따로 정의를 해야 한다. 하지만 이는 구조체를 생성할 때 마다 그 객체를 클라이언트/서버 쪽 각각에 생성해야 한다는 불편함이 있어 기존에 있는 데이터타입을 이용하여 구조체의 값을 리턴 받을 수 있는 알고리즘을 정립해 보았다.

1. DBManager 로부터 Find/ShowList 의 결과 값을 받는다.
2. 스트림 엔진의 스트림에서 각 멤버들의 값을 받아 각 멤버들끼리 구별자(#)를 두어 하나의 char* 타입으로 데이터를 넘겨준다.
3. char* 타입으로 받은 클라이언트 쪽에서는 이를 다시 식별하여 찾은 개수만큼 구조체를 생성하여 각 멤버에 다시 값을 넣어준다.
4. 이 구조체를 리턴 받은 클라이언트 프로그램에서는 찾은 결과 값만큼 출력한다.

3.3.1. 스트림 엔진의 서비스 쪽의 리턴 받는 모듈

```
PocSvcStatus dbmanager_find(PocSvcInfo *PocContext)
{
    Client *pClient;
    Obj *pObj;
    :
    int nResult = (pDataSturct+0)->nResult);
    for (int nCount = 0 ; nCount < nResult ; nCount++)

    sprintf(string, "%d#%s#%s#%d#%s#%s#%s#",
    ((pDataStruct + nCount)->filenum), ((pDataStruct + nCount)->userid), ((pDataStruct + nCount)->filename), ((pDataStruct + nCount)->filesize), ((pDataStruct + nCount)->regdate), ((pDataStruct + nCount)->validterm), ((pDataStruct + nCount)->f_desc) );

    strcpy(buf, string);
    strcat(retStrr, buf);
    :
     delete []realValue;
     return(PocSvcOk);
}
```

3.3.2. 스트림 엔진의 클라이언트 쪽의 리턴 받는 모듈

```
DataStruct* DBManager::Find();
{
  :
  char *token;
  char sep[]="#";
  token = strtok(retString, sep);

  while(token != NULL)
  {
     for ( int i =0; i < nResultNum ; i++)
     {
          (pDataStruct+i)->filenum = atoi(token);
          token = strtok(NULL, sep);
          strcpy( (pDataStruct+i)->userid, token );
          token = strtok(NULL, sep);
          strcpy( (pDataStruct+i)->filename, token );
          token = strtok(NULL, sep);
          (pDataStruct+i)->filesize = atoi(token);
          token = strtok(NULL, sep);
          strcpy( (pDataStruct+i)->regdate, token );
          token = strtok(NULL, sep);
          strcpy((pDataStruct+i)->validterm, token );
          token = strtok(NULL, sep);
          strcpy( (pDataStruct+i)->f_desc, token );
          token = strtok(NULL, sep);
     }
  :
  return pDataStruct;
}
```

3.4. 테이블 정의

3.4.1. 사용자 등록 테이블 (USER_TAB)

USER_TAB

USERNUM
USERNAME
USERID
USERPASSWD
USERGRADE

3.4.2. 파일관리 테이블(FILELIST_TAB)

FILELIST_TAB

USERNUM
FILENUM
FILENAME
FILESIZE
REGDATE
VALID_TERM
FILEDESC

3.4.3. 파일정보 테이블(FILEINFO_TAB)

FILEINFO_TAB

FILENUM
PLAYCNT
READDATE
PROTECTION

3.4.4. 파일 데이터 테이블(FILEDATA_TAB)

FILEDATA_TAB

FILENUM
FILEDATA

3.5. API Program

여러 API 프로그램 중 대표적인 두 예제만을 기술하겠다.

3.5.1. Camera to DB

실시간으로 카메라로부터 캡쳐받은 데이터를 데이터베이스에 저장하는 예제이다.

```
#include "clEssence.h"

void main(void)
{
        // Local Host
        Essence     *pEssence_Local;
        Source      *pSource_Local;
        Destination *pDestination_Local;
        Stream      *pStream_Local;
        :
        DBManager           *pDBManager;

        pEssence_Local = new Essence("203.253.21.68");
        pEssence_Local->SyncMessage();
        pEssence_Remote = new Essence("203.253.21.69");
        pEssence_Remote->SyncMessage();
        pDBManager = new DBManager(pEssence_Remote);
        pEssence_Remote->SyncMessage();
        :
        //Local Host
        pDestination_Local = new Destination( pEssence_Local,
"TCPCableWrite", (unsigned long) 0, NULL );
        pEssence_Local->SyncMessage();
        pSource_Local = new Source( pEssence_Local,
"SWH263Encoder", (unsigned long ) 0, NULL );
        pEssence_Local->SyncMessage();
        pStream_Local = new Stream( pEssence_Local,
pSource_Local, pDestination_Local );
        pEssence_Local->SyncMessage();
        //Remote Host
        pDestination_Remote=new Destination
( pEssence_Remote, "DBWriteMedium", pDBManager, "3" );
        pEssence_Remote->SyncMessage();
        pSource_Remote = new Source( pEssence_Remote,
"TCPCableRead", (ulong)pDestination_Local, "203.253.21.68" );
        pEssence_Remote->SyncMessage();
        pStream_Remote = new Stream( pEssence_Remote,
pSource_Remote, pDestination_Remote );
        pEssence_Remote->SyncMessage();
        //Open
        pStream_Remote->Open();
        pEssence_Remote->SyncMessage();
        pStream_Local->Open();
        pEssence_Local->SyncMessage();
        //Play
        pStream_Remote->Play();
        pEssence_Remote->SyncMessage();
        .
        .
        // Stop
        pStream_Remote->Stop();
        pEssence_Remote->SyncMessage();
        pStream_Local->Stop();
        pEssence_Local->SyncMessage();
        // Close
        pStream_Remote->Close();
        .
        .
}
```

3.5.2. Find

Find 질의문을 실행하여 수행한 결과 값을 리턴 받는 예제이다.

```
#include "clEssence.h"
#include <stdio.h>
#define ServerIP        "203.253.21.69"

void main(void)
{
        :
        pEssence = new Essence(ServerIP);
        pEssence->SyncMessage();
        pDBManager = new DBManager(pEssence);
        pEssence->SyncMessage();
        :
        pDataStruct = pDBManager->Find();
        pEssence->SyncMessage();

        int n = pDBManager->GetNumResultRows();
        pEssence->SyncMessage();
        :
        pDBManager->TerminateDataBase();
        pEssence->SyncMessage();
        :
}
```

4. 결론

이 연구는 인터넷 기반이 아니라 어플리케이션 모듈에서 본 연구실의 프로덕트인 스트림 엔진과 한국과학기술원에서 개발한 데이터베이스인 '오디세우스'의 통합으로 분산 멀티 미디어 DBMS 의 국산 핵심 기술을 확보하는데 그 의의가 있다. 본 문에서 제시된 시스템을 이용하여 향후 중/소규모 On-Demand 형 서버의 확보로 국애 산업계에 객체지향 관계형 DBMS 의 보급 및 확산이 촉진될 것이며, 핵심기술의 확보로 인터넷 상의 멀티미디어 메일이나 전자결재, 전자 상거래 등과 같은 SI 소프트웨어의 국제경쟁력을 강화할 것으로 예측한다. 아직까지 개발된 단계는 초기단계로 그 연구결과는 작지만, 이를 기반으로 하여 대용량 멀티미디어 메일 서버로서 서비스 타입뿐만 아니라 연출 시간에 따른 QoS 만족뿐만 아니라 시스템의 성능 향상을 위해 앞으로 그 연구는 더 계속되어야 할 것이다.

참고문헌

[1] 임영환 등, 멀티미디어 컴퓨터 공동연구개발 연구보고서, 1994. 7.

[2] Baker R, A. Downing. K. Finn, E. Rennison, D.H. Kim, and Y.H. Lim, "Multimedia Processing Model for a Distributed Multimedia I/O System," 3rd International Workshop on Network and Operating System Support for Audio/Video, 1993.

[3] 함재욱 외 4 명, "멀티미디어 프리젠테이션 동기화를 위한 선행 스케쥴링 기법," 정보과학회 논문지, 제 21 권, 제 12 호, 1994, pp.2187-2197.

[4] Soon M. Chung , "MULTIMEDIA INFORMATION STORAGE AND MANAGEMENT" KLUWER ACADEMIC PUBLISHERS.