УДК 004.7

О.І. Провотар, А.А. Заміховський, О.В. Галкін, М.М. Верес, Л.О. Катеринич

РЕАЛІЗАЦІЯ КОНЦЕПЦІЇ АДАПТИВНОГО МОВЛЕННЯ ТА СИСТЕМИ АВТОМАТИЧНОЇ ПІДГОТОВКИ КОНТЕНТУ

Запропоновано реалізацію концепції адаптивного мовлення за допомогою сервера потокового медіа для вирішення проблеми «Останньої милі» – забезпечення достатньої швидкості передачі даних користувачу при використанні сервісів он-лайн медіа трансляції.

Вступ

Загальновідомим є той факт, що більшість користувачів мережі Інтернет знають і користуються сервісами он-лайн медіа трансляції. До найбільш популярних можна віднести такі проекти: як YouTube та російський аналог RuTube. Але не один із них не враховує проблему «Останньої милі», тобто швидкість передачі каналу користувача фізично не дозволяє програвачу відтворювати та завантажувати медіа контент високої якості, який представлений на сторінках відео сервісів. Варіанти вирішення даної проблеми викладені в концепції адаптивного мовлення. На відміну від традиційного мовлення, в процесі відтворення контенту враховуються умови в яких знаходиться кінцевий користувач, тобто його поточна пропускна швидкість мережі. Якщо швидкість з'єднання недостатня, то якість зображення зменшується і навпаки, якщо є можливість перегляду медіа кращої якості, то відбувається перехід на вищу якість даних.

Існує багато варіантів реалізації технології адаптивного мовлення, однією з цікавих є трансляція на сервері потокового медіа двох потоків певного контенту з різними показниками якості та використання кадрів з потоку вищої якості та нижчої у залежності від умов на стороні клієнта [1]. Не менш цікавим є варіант трансляції контенту використовуючи кодування в реальному часі де зміна показників якості контролюється ключовим кадром, за допомогою якого можливо здійснювати кодування з будь-якої частини медіа контенту [2]. Але така можливість потребує надто багато ресурсів сервера.

Існує також варіант використання специфічного контейнера для зберігання відеоданих у декількох потоках з можливостями компресії. Даний контейнер має назву MVC (Multiview Video Coding), який входить до набору алгоритмів H.264/AVC, що запатентовані та не доступні для широкого використання [3].

Найвідомішою реалізацією даної ідеї є технологія IIS Smooth Streaming від компанії Microsoft [4]. В основі даної технології використовується сервер Microsoft IIS, а як протокол передачі між клієнтом та сервером використовується протокол HTTP. Для можливості організовувати адаптивне мовлення на сервері заздалегідь підготовляються відеоролики з різними показниками якості, такими як бітрейт та роздільна здатність. За допомогою даної технології вирішена початкова проблема швидкого відтворення. На початку клієнту передається фрагмент найнижчої якості, який швидко доставляється і починається його відтворення, потім, якщо дозволяє пропускна швидкість каналу, якість збільшується відповідно до апаратних можливостей користувача. Взаємодія між сервером і клієнтом проходить у такій послідовності:

– програвач проводить запит до сервера за протоколом HTTP, і запитує відеофрагмент певної довжини (в більшості випадків це фрагмент довжиною в 2 секунди);

– на початку відтворення запитується фрагмент найнижчої якості, який швидко доставляється до програвача і починається його відтворення;

– на основі даних отриманих при передачі першого фрагмента визначаються можливості користувача та наступний фрагмент запитується з урахуванням даних, які отримані при доставці попереднього фрагмента.

Цей процес повторюється постійно поки не буде припинено відтворення медіа-контенту на стороні користувача. Таким чином медіа-контент може відтворюватися в різні моменти часу з різними показниками якості, які залежать від поточних можливостей користувача.

Для даної технології використовується специфічна підготовка медіа-контенту. В результаті кодування отримується набір підготовлених медіа-потоків з різною роздільною здатністю, які упаковані в файл з розширенням «.ismv». Також підготовлюється файл маніфесту (.ism). У якому зберігається інформація про медіа-контент, а також список доступних відео-потоків. На початку відтворення проводиться запит до файлу маніфесту, який і надає інформацію про доступні ресурси. Щоб отримати інформацію про наявні ресурси необхідно провести запит до .ism файлу з параметром «Manifest» тобто вигляду: «*http://../media.ism/Manifest*», у відповідь отримаємо XML-файл з описом доступних відео потоків та їх показників якості, саме на основі цих даних програвач приймає рішення щодо зміни якості зображення в залежності від умов відтворення. Також у файлі маніфесту задається шаблон URL для запиту фрагмента відео-файлу, в більшості випадків використовується даний шаблон *«QualityLevels({bitrate})/Fragments(video==*{start time})». У секції *{bitrate}* задається показник якості відео-фрагмента, а у секції *{start time}* позиція відтворення фрагмента.

Таким чином клієнт і сервер знаходяться в постійному зв'язку між собою, що дає можливість реагувати на зміни умов клієнта, а також проводити збір певної статистичної інформації.

На сьогодні відомі декілька варіантів реалізації технології адаптивного мовлення, це вищезгадані Microsoft Smooth Streaming for Silverlight, Apple HTTP Adaptive Streaming for iPhone/iPad та Adobe Dynamic Streaming for Flash. Всі вони мають певну особливість, для їх функціонування необхідно мати спеціалізований сервер для проведення трансляції за протоколом HTTP. Запропонований далі підхід, дозволяє реалізувати технологію адаптивного мовлення без використання спеціалізованих серверів відповідних компаній. У його основі лежить принцип використання технології передачі відео за протоколом RTMP та зміна характеристик відео потоку в залежності від умов користувача, а в подальшому перенесення проміжної обробки на клієнтську сторону за допомогою технології псевдо потоків (*англ. pseudo streaming*). Окрім того, запропонована система також включає програмне забезпечення для автоматичної обробки відео контенту на основі статистичних даних, отриманих від користувачів даного ресурсу.

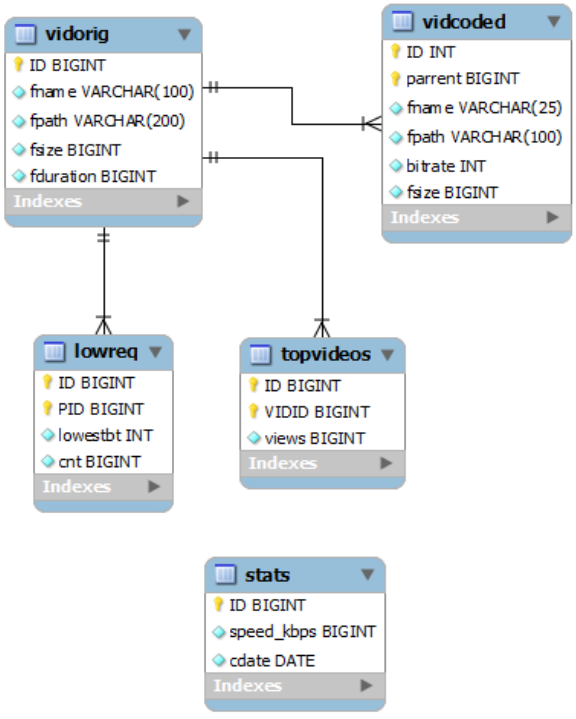
Опис системи

Основна ідея системи полягає в тому щоб надати користувачеві якісно підготовлений контент для перегляду в режимі он-лайн, зважаючи на пропускну можливість його мережі. Одним із можливих варіантів реалізації даної ідеї є підготовка відеоматеріалів з різними показниками якості, щоб можливо було передавати контент по мережі без затримок. Зважаючи на проблему «останньої милі» також необхідно контролювати процес відтворення для оперативного реагування на зміну пропускної швидкості каналу мережі.

Система складається з двох частин. Перша частина відповідає за доставку контенту кінцевому користувачеві, контроль за зміною пропускної швидкості мережі та відповідне реагування на ці зміни. Друга частина являє собою програмний комплекс, який створений для полегшення процесу додавання нового контенту на сервер та його попередню обробку, також він відповідає за обробку статистичних даних та автоматичну підготовку відеоматеріалів у залежності від потреб користувачів.

Серверна частина написана у вигляді Java Enterprise проекту, з'єднання з

базою даних забезпечується за допомогою технології EJB. Під час входу на початкову сторінку проекту відбувається вимірювання пропускної швидкості мережі клієнта. На основі отриманих даних генеруються посилання на контент, який у даний момент знаходиться на сервері. Вибір оптимального варіанта відео файлу проводиться на стороні сервера, servlet на основі отриманих даних проводить підбір розміру файлу, який в даний момент наявний у системі. Якщо не знайдено жодного відео файлу, який задовольнив би потреби користувача, то обирається найближчий, із наявних даних, до отриманого запиту зі сторони користувача. Дані про пропускну спроможність та дату відвідання сайту заносяться до бази даних для подальшої обробки та підготовки відповідних матеріалів. Під час перегляду наданого контенту може виникнути ситуація коли швидкість мережі може змінитись і користувачеві буде некомфортно переглядати обраний матеріал, саме на цю ситуацію орієнтований набір коду, який виконується на стороні користувача і відслідковує такого роду зміни. В даній реалізації присутня система перевірки в основі якої лежить відслідковування часу прогнозованого відтворення в порівнянні з фактичним часом, впродовж якого відбувалося відтворення. Якщо фактичні й прогнозовані результати не збігаються, користувачеві автоматично проводиться зміна контенту на один із тих, що на даний момент наявні в системі. Якщо такий матеріал відсутній то зміна не проводиться, а в базу даних додаються дані про необхідність підготовки відео з відповідними показниками якості. Зміна відеороликів «на льоту» є однією із основних проблем відтворення контенту, тому що необхідно приховати сам факт зміни від користувача. Дану проблему можна вирішити лише за рахунок медіа сервера, додавши до нього можливість попереднього завантаження фрагментів даних до початку їх відтворення. У разі використання комерційного варіанта WOWZA Media Server, то тут лише присутня можливість створити додаток до сервера, використовуючи відкрите API для розробників. У випадку Open Source проектів, таких як Red5 Media Server, залишається лише можливість редагування вихідних кодів та ручного збирання проекту. Технологія Microsoft IIS Smooth Streaming [4] вирішує дану проблему за рахунок протоколу передачі даних, кожен фрагмент відеоряду відправляється сервером та одночасно відтворюється програвачем на стороні клієнта, у разі отримання інформації про зміну пропускної спроможності просто змінюється наступний пакет на інший у залежності від необхідних показників якості. Отже, повертаючись до аспектів реалізації проекту, слід відмітити не останню роль програмного забезпечення та його використання. Для збереження інформації про наявний контент необхідно використовувати високоефективну базу даних, виходячи з міркувань загальнодоступності інструментарію обрано СУБД MySQL, яка на даний момент є найдоступнішою та відносно стабільною серед інших Open Source проектів. Далі показана на рисунку структура бази даних (БД), яка використовується в системі.



Рисунок

Як видно із структури БД, основною є таблиця, яка містить інформацію про наявні в системі ролики в оригінальному (незжатому) вигляді під назвою

"vidorig". Саме звідси веб інтерфейс отримує інформацію про наявний контент. Таблиця*"vidcoded"* містить інформацію про кодовані відео ролики, одне з полів яких зв'язане з ключовим полем таблиці оригінальних даних, що дозволяє контролюючим частинам на стороні сервера отримувати достовірну інформацію про наявні розміри та бітрейти варіантів основного відео ролика.

Не менш важливою є таблиця *"lowreq",* в якій розміщуються запити на зміну якості відео у процесі його перегляду. Дана таблиця містить у собі ключове поле за яким відбувається зв'язок з даними про оригінальні відео ролики, також присутнє поле, що відповідає за кількість таких запитів та поле зі значенням найнижчого бітрейту загалом за певним роликом. Під час роботи системи автоматичної підготовки матеріалів, дані відсортовуються за кількістю запитів і контент з більшим рейтингом обробляється в першу чергу. Для ведення певних статистичних спостережень, а в подальшому і прогнозування певних дій спирається на таблиці *"topvideos"* та *"stats"*. Перша з них використовується для аналізу популярності певних відео роликів та в подальшому збільшення їх пріоритету при попередній обробці, в іншій накопичується інформація про пропускну спроможність каналу користувачів ресурсу. На основі цих даних планується проводити аналіз оптимального розміру відео файлу для його комфортного перегляду. В подальшому планується відмовитися від збереження інформації про кодовані відео ролики у базі даних, більш ефективним та зручним буде створення файлу метаданих, що й буде містити всю необхідну інформацію про наявність інформації у системі, також це дозволить зменшити навантаження на сервер бази даних.

Як згадувалося раніше, друга частина проекту відповідає за автоматичну підготовку необхідних матеріалів. Для виконання цих та інших функцій необхідно мати програмний продукт, який має можливість перекодування відео файлів з одного формату в інший з використанням необхідних кодеків (*англ. CODEC – **Co**der **Dec**oder*). Всім цим вимогам задовольняє безкоштовна Open Source утиліта під назвою FFMpeg (http://ffmpeg.org) , даний програмний продукт надає користувачеві інтерфейс командного рядка для виконання всіх задач кодування та підтримує ті формати відео, які використовуються для трансляції по мережі.

Під час розробки даної частини проекту реалізовано програмний інтерфейс до командного рядка FFMpeg, його основна задача полягає у підготовці командного рядка в залежності від заданих параметрів якості відповідного контенту. В процесі розробки виникла необхідність зчитування метаданих (довжина ролика, бітрейт, розмір файлу, роздільна здатність), для вирішення цієї проблеми використано також FFMpeg, але в режимі зчитування інформації про файл. Щоб отримати ці дані використовується перенаправлення потоків стандартного вводу/виводу (stdin/stdout) та подальший розбір отриманих даних. Інформація про новий підготовлений контент заноситься до бази даних і стає доступною для використання. У даній частині проекту реалізований прямий доступ до бази даних, що спрощує передачу інформації між сервером бази даних та даної програми. Програма виконується постійно і з заданим інтервалом перевіряє базу даних на наявність нових запитів, якщо такі присутні, то вона починає підготовку і кодування відповідних відео файлів. Паралельно з потоком в якому проводиться аналіз та обробка, працює потік який обробляє команди користувача. Завдяки цим командам користувачеві надана можливість керування обробкою інформації. Для задоволення потреб користувачів система дозволяє автоматично готувати необхідний матеріал на основі аналізу отриманої статистики. В процесі активності користувача збирається інформація про переглянуті ним відеоролики, пропускну швидкість його Інтернет з'єднання та кількість запитів на зміну якості матеріалу. Отримані дані сортуються за спаданням від найпопулярнішого контенту до менш популярного, таким чином готується в першу чергу матеріал який найбільш затребуваний

користувачами. Процес підготовки починається відразу після перевірки наявності необхідної інформації в системі, інтервал перевірки задається адміністратором системи в залежності від навантаження.

Доцільно також зупинитися більш детально на функціоналі програми підготовки контенту. Як згадувалося раніше користувачеві надана можливість змінювати параметри виконання під час роботи. Основні операції керування здійснюються за допомогою командного рядка та мають певний синтаксис. Детальний опис команд наведено далі:

– /bwp – виведення інформації про місцезнаходження файлів основного контенту, тобто папки в яких знаходиться відеоматеріал як кодований, так і оригінальний;

– /dbh – виведення Інтернет адреси сервера бази даних у вигляді “host:port/”;

– /dbu – відображення назви активного профілю користувача бази даних;

– /newdb – за допомогою даної команди користувачеві надається можливість змінювати параметри з’єднання з сервером баз даних. Команда відпрацьовує у діалоговому режимі та пропонує заповнити поля необхідні для успішного з’єднання, у разі якщо введені дані не відповідають дійсності неможливо встановити зв'язок, то повертаються параметри попереднього успішного з’єднання. Відповідна процедура повторюється також під час першого запуску програми і не дозволяє запуск основного циклу обробки без успішного під’єднання до бази даних. Саме на етапі роботи з базою даних виникла основна проблема безкоштовного програмного забезпечення, неякісні допоміжні бібліотеки. Справа в тому, що драйвер з’єднання з БД, який розповсюджується у комплекті з СУБД MySQL, абсолютно неможливо використовувати при розробці. Окрім того, що некоректно встановлюється з’єднання, яке може бути розірвано без видимих на те причин, відтік оперативної пам’яті є більш ніж помітним, після завершення одного процесу комунікації з базою даних, кількість оперативної пам’яті збільшується на 1 Mb, що у будь-якому випадку є неприпустимим. Після недовгих пошуків з’ясувалося, що дана проблема саме у драйвері, тому знайдено більш функціональну та стабільну заміну в проекті MySQL++, ознайомитися можна за посиланням http://tangentsoft.net/mysql++/;

– /currint – виведення інтервалу затримки при повторній перевірці та підготовці відеоматеріалів;

– /chint <interval> – команда для зміни інтервалу перевірки. Значення інтервалу вводиться у форматі “seconds-minutes-hours-days”, що допускає введення лише необхідного значення, але в строгому порядку зліва направо;

– /exit – завершення роботи основної програми;

– /addv – команда використовується для додання нових відео файлів у систему. Основне її призначення максимально зменшити кількість дій зі сторони користувача. Отже у діалоговому режимі користувачеві пропонуються ввести інформацію про місцезнаходження відео файлу та ім’я яке буде відображатися у веб інтерфейсі системи. В автоматичному режимі проводиться внесення відповідної інформації у базу даних та розпочинається позачерговий процес підготовки кодованого відео у найвищій якості, яка доступна в налаштуваннях програми. Для отримання всієї інформації використовується декілька допоміжних засобів для попередньої обробки, використовуються такі засоби: отримання інформації з метаданих вихідного відео файлу та внесення її у кодовані файли.

Отримання метаданих з файлу проводиться шляхом запуску утиліти FFmpeg у режимі зчитування, щоб провести всі необхідні маніпуляції відбувається захоплення вихідного потоку утиліти та його подальший розбір з перетворенням отриманої інформації у необхідну форму. Перекодування проходить за допомогою утиліти FFmpeg, на вхід якої подається підготовлений командний рядок, який генерується в автоматичному режимі та включає всю необхідну інформацію як про вхідні дані, так і про вихідні.

Приклад реалізації головного циклу перевірки та підготовки контенту можна побачити далі.

```
DWORD WINAPI Statistical_Analyzer(LPVOID
lp)
{
   vector <int> mygrid;
   mygrid.push_back(100);
   mygrid.push_back(300);
   mygrid.push_back(500);

   BitrateGrid * btg = new BitrateGrid(mygrid);
   vector<Encoder*> encvec;
   vector<vidcoded*> codvec;
   vector<stats*> mystat;
   vector<topVideos*> tpv;
   vector<lowReq*> lrw ;
   while(true){
      system("cls");
      Connector * cnt = new
Connector(DBHost,DBUser,DBPass);
      mystat = cnt->GetStatistic();
      tpv = cnt->GetTop();
      lrw = cnt->GetRequests();
      if(lrw.size()==0){goto endpoint;}
      sort(tpv.begin(),tpv.end(),sortForTop);

      //prepare encoding tasks
      for(UINT i=0;i<tpv.size();i++){
         int k= FindInByID(lrw,tpv[i]->VIDID);
         if(k!=-1){
            int z = btg->lowerThan(lrw[k]->
lowestBitrate);
            if(z!=0){
               ExifTool * einfo = new ExifTool();
               encvec.push_back(new Encoder());
               vidorig vd = cnt->
getOrigVideoByID(tpv[i]->VIDID);
               MetaData * mtd = new MetaData();
               mtd=einfo->
GetInfoForFile(basePath+vd.fpath);
               encvec[encvec.size()-1]->
SetSourceVideoFile(basePath+vd.fpath);
               encvec[encvec.size()-1]-
>SetBitrate(z);
               encvec[encvec.size()-1]->
SetDimensions(mtd->width,mtd->height);
               string prep =
BitrateGrid::split(vd.fpath,"/")[1];
               prep =
BitrateGrid::split(prep,".")[0];

prep+="_"+BitrateGrid::toString(z)+"."+"flv";
               encvec[encvec.size()-1]->
SetDestinationVideoFile(basePath+"codedvid/"+p
rep);
               vidcoded * vdcd = new vidcoded();
               vdcd->bitrate=z;
               vdcd->fpath="codedvid/"+prep;
               vdcd->PID=tpv[i]->VIDID;
               codvec.push_back(vdcd);
               delete mtd;
               delete einfo;
            }
         }
      }
      for(UINT i=0;i<encvec.size();i++)
      {
      encvec[i]->encode();
      }
      for(UINT i=0;i<codvec.size();i++)
      {
         ExifTool * einfo = new ExifTool();
         MetaData * mtdt = new MetaData();
         mtdt = einfo->
GetInfoForFile(basePath+codvec[i]->fpath);
         codvec[i]->fsize=mtdt->file_size;
         cnt->InsertCodedVideo(*codvec[i]);
         delete einfo;
      }
      encvec.clear();
      codvec.clear();
      cnt->DeleteStatisticalTables();
endpoint: delete cnt;
      cout<<endl<<"new iter"<<endl;
      CListener((LPVOID)false);
      Sleep(timeInterval);
   }
   delete btg;
   return 0;
}
```

Як бачимо вище зв'язок з базою даних встановлюється за допомогою класу Connector в якому реалізовано всі необхідні методи для роботи з БД, а також присутні методи які відповідають за отримання необхідної інформації і її представлення у вигляді придатному для подальшого використання. Також одним із основних класів, що відповідає за обробку відео контенту, – ExifTool, який є по суті обгорткою (wrapper) навколо утиліти FFMpeg і за допомогою якої отримується вся необхідна інформація для подальшого використання. Вся отримана з файлу інформація зберігається в структурі MetaData для подальшого використання в процесі підготовки. А клас, який відповідає безпосередньо за конвертацію носить назву Encoder,

котрий також являє собою обгортку навколо FFMpeg. Інші вищепредставлені класи й методи є допоміжними у процесі підготовки й конвертації та використовуються для проміжного зберігання інформації.

Висновок

У даній роботі продемонстровано один із варіантів реалізації концепції адаптивного мовлення за допомогою сервера потокового медіа. Ключовою відмінністю даного підходу є те, що для організації адаптивного мовлення не потрібно використовувати спеціалізовані Web-сервери, використання та обслуговування яких потребує значних капіталовкладень.

Запропоновано варіант системи автоматичного адміністрування та підготовки медіа контенту на основі аналізу статистичних даних зібраних іншою частиною проекту, яка спрощує адміністрування та дозволяє раціонально використовувати простір носіїв зберігання інформації. Проте в подальшому планується взагалі відмовитися від використання серверів потокового медіа на користь системи псевдо потоків *(англ. Pseudo streaming)*. Дана технологія дозволяє, лише доданням до Web-сервера спеціалізованого обробника запитів, реалізувати систему відтворення відео з довільної позиції. Завдяки цьому з'являється можливість реалізації концепції адаптивного мовлення без використання допоміжного програмного забезпечення для потокового мовлення.

1. *Xiaoyan Sun, Feng Wu, Shipeng Li, Wen Gao, Ya-Qin Zhang*. "Seamless switching of scalable video bitstreams for efficient streaming" IEEE Transaction on multimedia, Vol. 6, no 2, 291-303, 2004.
2. *Xiaoyan Sun, Feng Wu, Shipeng Li, Guobin Shen, Wen Gao*. "Drift-Free Switching of Compressed Video Bitstreams at Predictive Frames", IEEE Transaction on multimedia, Vol. 16, no 5, 565-576, 2006.
3. *Xun Guo, Yan Lu, Feng Wu, Wen Gao, Shipeng Li.* "Free Viewpoint Switching in Multi-view Video Streaming Using Wyner-Ziv Video Coding", SPIE Visual Communications and Image Processing, VCIP 2006, San Jose, CA, USA, Jan. 2006.
4. *Microsoft* IIS Smooth Streaming technology,
5. http://www.iis.net/download/SmoothStreaming

Одержано 30.03.2012

Про авторів:

Провотар Олександр Іванович,
доктор фізико-математичних наук,
професор, завідувач кафедри інформаційних систем факультету кібернетики,

Заміховський Андрій Андрійович,
студент кафедри інформаційних систем факультету кібернетики, 5 курс.

Галкін Олександр Володимирович,
кандидат фізико-математичних наук,
доцент кафедри інформаційних систем факультету кібернетики,

Верес Максим Миколайович,
кандидат фізико-математичних наук,
доцент кафедри інформаційних систем факультету кібернетики,

Катеринич Лариса Олександрівна,
кандидат фізико-математичних наук,
асистент кафедри інформаційних систем факультету кібернетики.

Місце роботи авторів:

Київський національний університет
імені Тараса Шевченка,
03680, м. Київ,
Проспект Академіка Глушкова, 4Д.
Тел.: (044) 259 0511,
e-mail: islab@unicyb.kiev.ua