КОНЦЕПЦІЯ АДАПТИВНОГО МОВЛЕННЯ ТА СИСТЕМИ АВТОМАТИЧНОЇ ПІДГОТОВКИ КОНТЕНТУ

УДК 004.7:004.272.26

ГАЛКІН Олександр Володимирович
к.ф.-м.н., доцент кафедри інформаційних систем Київського національного університету імені Тараса Шевченка.
Наукові інтереси: методи об'єктно-орієнтованого програмування, системи штучного інтелекту, Java.
E-mail: galkin@unicyb.kiev.ua

ВСТУП

Загальновідомим є той факт, що більшість користувачів мережі Інтернет знають про сервіси онлайн-трансляції медіа і користуються ними. До найбільш популярних можна віднести такі проекти, як YouTube та його російський аналог RuTube. Але жоден з них не враховує проблеми «останньої милі», тобто швидкість передачі каналу користувача фізично не дозволяє програвачеві відтворювати та завантажувати медіа-контент високої якості, який доступний 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, зокрема таким є трансляція на сервері потокового медіа двох потоків певного контенту з різними показниками якості та використання кадрів з потоку вищої якості та нижчої в залежності від умов на стороні клієнта [1]. Не менш цікавим є варіант трансляції контенту з використанням кодування в реальному часі, коли зміна показників якості контролюється ключовим кадром, за допомогою якого можливо проводити кодування з будь-якої частини медіа-контенту [2]. Але така можливість потребує надто багато ресурсів серверу. Існує також варіант використання специфічного контейнеру для зберігання відеоданих в декількох потоках з можливостями компресії. Даний контейнер має назву MVC (Multiview Video Coding), який входить до набору алгоритмів H.264/AVC, що запатентовані і не доступні для широкого використання [3].

Найвідомішою реалізацією останньої ідеї є технологія IIS Smooth Streaming від компанії Microsoft [4]. В основі даної технології використовується сервер Microsoft IIS, а в якості протоколу передачі між клієнтом та сервером використовується протокол HTTP. Для того, щоб була можливість організувати адаптивне мовлення, на сервері заздалегідь підготовлюються відеоролики з різними показниками якості, такими як бітрейт та роздільна здатність. Ця технологія допомагає вирішити проблему швидкого початку відтворення: клієнту з самого початку 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 в наступній послідовності.

1.Програвач проводить запит до сервера по протоколу HTTP, і запитує відео-фрагмент певної довжини (в більшості випадків це фрагмент довжиною в 2 секунди).

2.На початку відтворення запитується фрагмент найнижчої якості, який швидко доставляється до програвача і починається його відтворення.

3.На основі даних, отриманих при передачі першого фрагменту, визначаються можливості користувача та наступний фрагмент запитується з врахуванням даних, що були отримані при доставці попереднього фрагменту.

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 файлі маніфесту зберігається інформація про медіа-контент, а також список доступних відео-потоків. На початку відтворення проводиться запит до файлу маніфесту, котрий 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 фрагменту відео-файлу, у більшості випадків використовується шаблон *«QualityLevels({bitrate})/Fragments(video={start time})»*. У секції *{bitrate}* задається показник якості відео-фрагменту, а у секції *{start time}* позиція відтворення фрагменту.

Отже, 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 згадана вище 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 по протоколу HTTP.

Постановка задачі – реалізувати технологію адаптивного мовлення без використання спеціалізованих серверів відповідних компаній, використовуючи технології передачі відео по протоколу RTMP. Більш того, в системи має бути реалізована можливість зміни характеристик відео потоку в залежності від умов користувача, а в подальшому перенесення проміжної обробки на клієнтську сторону за допомогою технології псевдопотоків**.**

Мета роботи полягає у розробці практично діючої системи адаптивного мовлення, яка може динамічно змінювати якість відео, яке транслюється через Інтернет, в залежності від пропускної здатності інтернет-каналу між сервером та клієнтом. Також система повинна містити програмне забезпечення для автоматичної обробки відео контенту на основі статистичних даних, отриманих від користувачів даного ресурсу.

ОСНОВНИЙ МАТЕРІАЛ

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 такої 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 задля того, щоб оперативно реагувати 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, створений щоб полегшити процес додавання нового контенту на сервер та його попередню обробку, також він відповідає за обробку статистичних даних та автоматичну підготовку відеоматеріалів в 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 в даний момент знаходиться на сервері. Вибір оптимального варіанту відео файлу проводиться на стороні сервера, servlet на основі отриманих даних проводить підбір розміру файлу, який в даний момент наявний в 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 код, 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. Один із можливих варіантів - використання комерційного варіанту WOWZA Media Server, але тут лише присутня можливість створити додаток до серверу, 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 в 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 було обрано СУБД MySQL, яка на даний момент є найдоступнішою та відносно стабільною серед інших Open Source проектів. Нижче продемонстрована структура бази даних, яка використовується в системі (рис. 1).

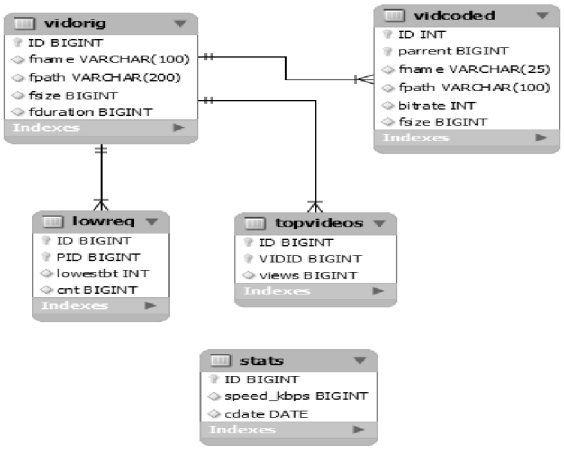

Рисунок 1 – Структура бази даних, яка використовується в системі

Як видно із структури БД, основною є таблиця яка містить інформацію про наявні в системі ролики в оригінальному (нестиснутому) 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 боці 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, по якому 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 по певному ролику. Під час роботи системи автоматичної підготовки матеріалів дані відсортовуються по кількості запитів, і контент з більшим рейтингом обробляється в першу чергу. 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 роз-

міру 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 в системі, також це дозволить зменшити навантаження на сервер бази даних.

Друга частина проекту відповідає за автоматичну підготовку необхідних матеріалів. Для виконання цих та інших функцій необхідно мати програмний продукт, що має можливість перекодування відео файлів з одного формату в інший з використанням необхідних кодеків. Всім цим вимогам задовольняє безкоштовна Open Source утиліта під назвою FFMpeg. Цей програмний продукт надає користувачеві інтерфейс командного рядка для виконання усіх задач кодування та підтримує ті формати відео, які використовуються для трансляції по мережі.

Під час розробки згаданої частини проекту було 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 ролику, бітрейт, розмір файлу, роздільна здатність), для вирішення цієї проблеми було 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 вони є, 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. Отриманні дані сортуються по спаданню 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 детальніше 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. Наведемо детальний опис команд.

/bwp – виведення інформації про місцезнаходження файлів основного контенту, тобто папки, в якій 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 в діалоговому режимі і пропонує заповнити необхідні для успішного з'єднання поля, у разі якщо введені дані не відповідають дійсності і 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. Виявилося, що драйвер з'єднання з БД, який розповсюджується у комплекті з СУБД MySQL, використовувати при розробці неможливо. Крім того, некоректно встановлюється з'єднання, яке може бути розірване без видимих 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 неприпус-

тимим. Наведена проблема властива саме драйверу БД, тому було знайдено більш функціональну та стабільну заміну в проекті 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 з боку користувача. Отже, у діалоговому режимі користувачеві пропонується 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 доступній у налаштуваннях програми якості. Для отримання всієї інформації використовується декілька допоміжних засобів для попередньої обробки; використовуються такі засоби, як 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 усі необхідні маніпуляції відбувається захоплення вихідного потоку утилітою 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 у автоматичному режимі і включає всю необхідну інформацію як про вхідні дані, так і про вихідні.

Наведемо приклад реалізації головного циклу перевірки та підготовки контенту.

```
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/"+prep);
          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 видно з прикладу, зв'язок з базою даних встановлюється за допомогою класу Connector, в якому реалізовано всі необхідні методи для роботи з БД, а також присутні методи, що відповідають за отримання необхідної інформації та її представлення в придатному для подальшого використання вигляді. Також одним із основних класів, що відповідає за обробку відео контенту, є 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. Клас Encoder відповідає безпосередньо за конвертацію і також є обгорткою навколо FFMpeg. Решта з наведених вище класів та методів є допоміжними у процесі підготовки й конвертації та використовуються для проміжного збереження інформації.

ВИСНОВКИ

Розглянуто 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 веб-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, що спрощує адміністрування та дозволяє раціонально використовувати простір носіїв зберігання інформації. Проте в подальшому планується взагалі відмовитися від використання серверів потокового медіа на користь системи псевдопотоків*(англ. Pseudo streaming)*. Дана технологія дозволяє за допомогою лише додавання до веб-серверу 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.

ЛІТЕРАТУРА:

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, N.2, 2004, pp 291-303
2. Xiaoyan Sun, Feng Wu, Shipeng Li, Guobin Shen, Wen Gao "Drift-Free Switching of Compressed Video Bitstreams at Predictive Frames"// IEEE Trans. on Circuits Systems for Video Technology. Vol. 16, no. 5, May 2006, pp. 565-576.
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 // http://jmvm.vse.cz/wp-content/uploads/ 2011/05/t_IIS_ Smooth_ Streaming_ Technical_ Overview.pdf