스트리밍 환경에서 초기 대기 시간 감소를 위한 MPEG-4 파일 포맷 분석

배수영°, 마평수
한국전자통신연구원 정보가전연구부

MPEG-4 File Format Analysis for Reducing Initial Latency Time in Streaming System

SuYoung Bae°, PyeongSoo Mah
Electronics and Telecommunications Research Institute

요 약

MPEG-4 파일 포맷은 사용자에게 다양한 기능을 제공하고 다수의 데이터를 유연성있게 제어하기 위해 많은 양의 메타데이터를 정의하고 있고, 이에 접근하는 서버와 로컬 시스템은 이들 메타데이터를 분석하는데 오랜 시간을 소비해야 한다. 본 논문은 이런 분석 지연을 없애기 위해 MPEG-4 파일에서 메타데이타를 실시간으로 분석하여 초기 지연 시간을 줄이는 방법을 제시하며, 그 결과를 보인다.

1. 서론

MPEG-4 워킹 그룹은 애플사의 QuickTime 파일 포맷에 기반을 두어 미디어의 표현, 상호교환, 관리, 편집 등이 용이하도록 확장성 있는 파일 포맷을 ISO/IEC 14496-1에 정의하였다. 이 파일은 추후 ISO Base Media File Format으로 확장되어 14496 시리즈(MPEG-4 표준)가 아닌 다른 포맷의 데이터도 함께 저장할 수 있는 일반적인 포맷으로 확장되었으며, 무선 인터넷 표준 협회와 각 무선 사업자들은 후자를 바탕으로 MPEG-4 비디오, 오디오 데이터 뿐만 아니라, EVRC, AMR 등의 스피치 데이터, H.263 비디오 데이터를 저장하기 위한 포맷 규정을 진행시키고 있다.

하지만, MPEG-4 파일 포맷은 사용자에게 다양한 기능을 제공하고 다수의 데이터를 유연하게 제어하기 위해 많은 양의 메타데이터를 정의하고 있어서, MPEG-4 파일 재생을 위해서 메타데이터를 분석하는 서버와 로컬 시스템은 이들 데이터를 분석하는데 오랜 시간을 소비해야 한다.

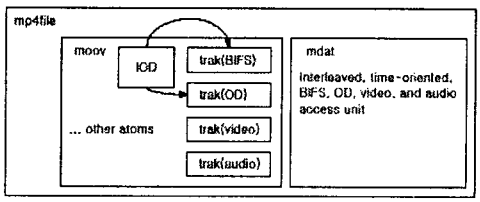
따라서 MPEG-4 파일에서 메타데이타를 실시간으로 분석하여 초기 지연 시간을 줄이는 방법에 대한 연구가 필요하며, 본 논문은 이에 대한 방법을 제시한다.

본 논문의 2장에서는 MPEG-4 파일 포맷에 대한 소개를 다루고, 3장에서는 메타데이타를 실시간으로 분석하는 방법을 소개한다. 4장에서 구현 결과를 보이고, 5장에서 결론을 맺는다.

2. MPEG-4 파일 포맷 – MP4

MPEG-4 파일은 미디어 데이터를 여러 파일로 분리시키고, 미디어 데이타를 인덱싱하기 위한 메타데이터를 가진 파일 형태와 하나의 파일에 모든

미디어 데이터와 메타데이터를 통합한 형태로 구분된다. MPEG-4는 후자의 경우를 권고하고 있으며, 이의 구조는 다음 그림 1과 같다.



<그림 1> MPEG-4 파일의 일반적인 구조

그림 1에서 보는 바와 같이 파일 구조는 일반적인 오디오, 비디오, 이미지 등의 미디어 데이터(mdat)와 미디어 데이터에 접근하기 위한 메타데이터(moov)로 구성된다. 미디어 데이터(mdat)는 재생 시간과 접근 단위(access unit)를 바탕으로 각 샘플이 하나의 단위로 인터리빙(interleaving)된 구조를 가진다. 인터리빙 단위는 포맷에서 청크(chunk)라 명명되고, 이들 청크와 청크를 구성하는 샘플들에 대한 모든 정보 및 이들의 재생에 필요한 모든 환경 정보를 메타데이타(moov)가 소유하고 있다.

메타데이타(moov)는 계층 구조를 이루고 있으며, 각 미디어에 대한 재생과 제어 정보를 트랙(track)이란 개념으로 분류하여 저장한다. 미디어 이외에 장면 구성과 연계된 BIFS(Binary Format for Scenes)와 관련된 정보 또한 하나의 트랙으로 저장된다. BIFS가 포함될 경우, BIFS와 관련되어 재생될 미디어를 연결시켜주는 OD(Object Descriptor)들 또한 하나의 트랙을 차지한다. IOD(Initial Object Decsriptor)는 이들 모든 미디어와 BIFS, OD 트랙들에 대한 프로파일 정보를 가지고 있으며, 메타데이타의 최상위는 무비 아톰(moov)으로 명명된다.

각 미디어 트랙은 트랙 생성/갱신 시간, 트랙 크기 등의 환경 정보 이외에 미디어를 구성하는 샘들들의 MPEG-4 파일 내의 위치와 그들의 재생 시간, 크기, 그리고 청크(chunk) 정보를 테이블로 가진다. 이들 샘플 정보는 모두 샘플 테이블(sample table)에 저장되며, 샘플 테이블은 크게 디코딩 타임 테이블(decoding time to sample), 컴포지션 타임 테이블(composition time to sample), 동기화 샘플 테이블(sync sample map), 샘플 사이즈 테이블(sample size), 청크당 샘플 개수 테이블(sample to chunk), 청크 오프셋 테이블(chunk offset)로 구성된다.

디코딩 타임 테이블은 각 샘플들의 디코딩 시간을 저장하고, 이전 샘플에 비해 얼마의 시간이 지난뒤에 디코딩 되어야 하는지를 저장한다. 컴소지션 타임 테이블은 각 샘플별 재생 시간을 저장한 것으로, 디코딩 타임과 마찬가지로 이전 샘플이 재생된 뒤에 얼마의 시간이 지난 후 디스플레이 되어야 하는지 델타(delta)값을 가지고 있다.

동기화 샘플 테이블은 미디어 재생시 특정 위치로의 점프를 고려해서 해당 장소로 이동하기 위한 것으로서, 비디오에서만 존재하고 I 프레임의 샘플 번호를 가진다. 샘플 사이즈 테이블은 각 샘플이 차지하는 바이트 수를 저장하고 있다. 청크당 샘플의 개수 테이블은 인터리빙된 미디어 데이터(mdat)에서 하나의 청크가 갖는 샘플들의 수를 기록한 것이며, 청크 오프셋 테이블은 파일 내에서 각 청크의 시작 위치를 나타낸다.

이들 미디어와 BIFS 트랙 이외에도 메타데이터(moov)는 힌트(hint) 트랙을 가질 수 있다 힌트 트랙은 MPEG-4 파일이 스트리밍 방식으로 사용자에게 전송될 경우, 전송 계층에서 사용되는 프로토콜에 따라서 오디오, 비디오 데이터를 적절하게 패키징 할 수 있게 하는 정보를 제공한다.

이처럼 메타데이타는 미디어 재생에 필요한 정보를 샘플별로 저장하고 있으므로, 미디어 데이터의 샘플 수에 비례해서 증가하게 된다. 고품질 대용량화 되어가는 미디어의 추이를 볼 때, 이들 데이터가 차지하는 부피도 점점 커지게되어, 이들을 파싱하는데에도 많은 시간이 소비되며, 이를 초기에 모두 분석해서, 재생하기 위해서는 이를 요구한 사용자의 많은 인내심을 필요로 하게 된다.

3. MPEG-4 파일 해석

본 논문에서는 미디어 재생시 메타데이터 전체를 한번에 분석하는 대신, 재생 데이터가 요구될 때마다 필요한 데이터를 읽어오기 위한 메타데이타를 실시간으로 분석하는 방법을 제시한다.

파일 해석 모듈은 메타데이터 중 미디어 데이터를 읽는데 필요한 부분의 인덱스를 소유하고

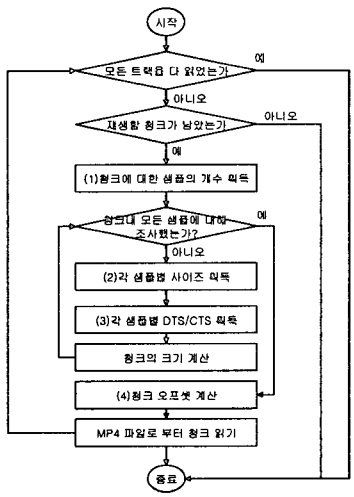
있고, 필요할 경우 이들 인덱스를 이용해 메타정보를 읽고 분석해서 원하는 미디어 데이터를 재생 시스템에 공급한다.

이 방법은 전체 메타데이타 파싱 시간의 접근 단위별 또는 청크 단위의 분할을 가져온다. 총 메타데이터의 분석하는 시간으로 합산해 볼 때, 파일 시스템에 대한 많은 회수의 접근으로 인해 전체 메타데이타를 한번에 분석하는 시간보다 더 커지겠지만, 이들 시간을 청크 단위 또는 접근 단위로 접근할 수 있도록 분할 함으로서 이들 시간이 무시될 수 있는 만큼의 단위가 되고, 또한 초기 재생 지연이 발생하지 않는 장점이 있다.

파일 해석 모듈에서 미디어 데이터를 읽어들이기 위해 필요한 메타데이터는 다음과 같다.

- 샘플 사이즈 (Sample Size, stsz)
- 청크당 샘플수 (Sample To Chunk , stsc)
- 청크 오프셋 (Chunk Offset, stco)
- 샘플별 재생 시간 (Time To Sample, stts)

사용자가 MPEG-4 파일 재생을 요청할 때 파일 해석 모듈은 초기 메타데이타 분석 모듈을 통해 이들 데이터에 대한 인덱스를 트랙별로 저장해서 관리한다. 메타데이터 내의 모든 아이템은 각기 자신의 크기를 헤더 내에 포함하고 있으므로, 필요한 메타데이터가 아닌 경우, 이들을 스킵하는 방식을 통해, 올바른 인덱스를 빠른 시간내에 소유할 수 있다.



<그림 2> 파일 해석 모듈 동작

미디어 데이터를 코덱에 전달해서 끊김없이 재생하기 위해 일정 시간 내에 데이터가 코덱에 전달되어야 하며, 시간 순서대로 데이터를 인터리빙 시키는 MPEG-4 파일의 특성을 고려해서 파일 생성 모듈은 다음의 단계를 밟아 트랙별 청크를 읽어들인다.

파일 해석 모듈은 모든 트랙에 대해 해당 시간에 재생해야할 청크가 있는지 확인 한 후, 각 트랙별 청크 데이터를 읽을 준비를 한다. (1) 청크당 샘플수 (Sample To Chunk) 테이블을 읽어들여 청크를 구성하는 샘플의 개수를 획득하고, 샘플의 개수만큼 (2) 샘플 사이즈 (Sample Size) 테이블을 읽어와서 각 샘플별 사이즈를 획득한다. 샘플 사이즈는 읽어들일 청크의 크기를 결정하는데 사용되며, 파일 해석 모듈은 (4) 청크 오프셋 테이블(Chunk Offset)을 참조하여 파일내 청크의 위치로 이동한 후, (3)의 결과로 생성된 청크의 크기만큼 데이터를 읽어들인다. 또한 해석 모듈은 청크 내에 존재하는 각 샘플들이 동기에 맞게 재생될 수 있도록 (3) 샘플별 재생 시간(Time To Sample) 테이블을 읽어들여 디코딩 또는 프리젠테이션 모듈로 타임 스탬프(Decoding/Composition Time Stamp, DTS/CTS) 값을 전달한다.

메타데이타의 트랙별 샘플 테이블은 전체 크기에 큰 영향을 미치지는 못하지만 각기 자신의 정보를 축소시키고자 중첩된 정보는 한번에 표현하고자 노력한다. 따라서, 그림 2에서 미디어 데이터를 읽어들이기 위해 필요한 4개의 메타데이터에 접근할때 중첩 정보 표현을 고려해야 한다. 다음 그림 3과 4는 (1) 청크당 샘플 개수 획득 하는 방법과 (3) 샘플별 재생 시간 구하는 방법을 보인다.

그림 3과 4의 [I], [II], [III]에서 파일 해석 모듈은 MPEG-4 파일로부터 현재 인덱싱하고 있는 각 테이블의 엔트리를 하나씩 읽어와 현재 샘플에 대한 정보를 담고 있는지 검사한다. 이와 같은 작은 단위로의 파일 시스템 접근은 재생 시작시 모든 데이터를 한꺼번에 파싱할 때 보다 더 많은 파일 시스템 접근을 발생시키지만, 전체 분석 시간의 분할을 가져와서 분석 시간을 충분히 무시할 수 있는 값으로 분산시키는 효과가 있다.

```
typedef struct IndexTableStruct{   // MP4 파일 내에서 각각의 각 테이블별 위치
        long stsc;                 // sample to chunk
        long stsz;                 // sample size
        long stts;                 // sample time stamp, duration
        long stco;                 // chunk offset
};

typedef SampleToChunk{
    int FirstChunk;          // 동일한 샘플 개수를 갖는 연속적인 청크 중 시작 청크 번호
    int SamplesPerChunk;     // 청크가 갖는 샘플의 개수
};

int chunkNo;                 // 현재 조사할 chunk number
IndexTableStruct IndexTable[NumOfTrack];
SampleToChunk cur_stsc, next_stsc;

//------- read 'sample to chunk'

do{
    if( !IsLastEntry(IndexTable[TrackID].stsc) ){ //마지막 엔트리 일경우, 다음 stsc 엔트리 비교 없음
        fseek( fp_MP4File, IndexTable[TrackID].stsc, SEEK_SET );
        fread( &cur_stsc, sizeof(SampleToChunk), 1, fp_MP4File );  //현재 stsc 엔트리 읽어옴 - [i]
        fread( &next_stsc, sizeof(SampleToChunk), 1, fp_MP4File ); //다음 stsc 엔트리 읽어옴 - [ii]

        if( ( cur_stsc.FirstChunk <= chunkNo ) && ( chunkNo < next_stsc.FirstChunk ))
            return cur_stsc.SamplesPerChunk;
                // 조사할 청크번호가 현재 stsc 엔트리의 청크 번호보다 크고,
                // 다음 stsc 엔트리 청크번호보다 작다면 현재 읽어들인 엔트리와 동일한 정보 소유
        else IndexTable[TrackID].STSC += sizeof(SampleToChunk);
    }
    else{     //마지막 엔트리 일경우, 다음 stsc 엔트리 비교 없이 마지막 stsc 엔트리와 동일 값
        lseek( fp_MP4File, IndexTable[TrackID].STSC, SEEK_SET );
        fread( &cur_stsc, sizeof(SampleToChunkTable), 1, fp_MP4File );
        return cur_stsc.SamplesPerChunk;
    }
}while(1);
```

<그림 3> 청크당 샘플수(stsc) 테이블 접근

```
typedef struct TimeToSample {
    int SampleCount;      // 동일한 duration을 가지는 샘플 개수
    int Duration;         // 이전 샘플과의 재생 인터벌
}

TimeToSample cur_stts;
int stts_count[TrackID];

//------- read 'time to sample', 각 샘플마다 아래 루틴 처리.

fseek( fp_MP4File, IndexTable[TrackID].ctts, SEEK_SET );
fread( &cur_stts, sizeof(TimeToSample), 1, fp_MP4File );   // 현재 stts의 엔트리 읽음- [iii]

if( cur_stts.SampleCount - stts_count[TrackID] > 0 ){
    // 읽어 들인 엔트리 내의 samplecount 만큼 샘플을 처리하지 않았다면,
    // 현재 샘플도 동일한 duration을 가짐
    stts_count[TrackID]++;

    if (cur_stts.SampleCount == stts_count[TrackID] ) {
        // 읽어 들인 엔트리 내의 samplecount 만큼 샘플을 처리했다면
        // 다음 엔트리를 읽을 준비 수행
        IndexTable[TrackID].stts += sizeof (TimeToSample);
        stts_count[TrackID] = 0;
    }
    return cur_stts.Duration;
}
```

<그림 4> 샘플별 재생 시간(stts) 테이블 접근

4. 구현

본 모듈은 리눅스 Redhat 7.0 환경에서 c 언어로 구현되었다. 본 모듈의 테스트 환경은 MPEG-4의 적용 분야를 고려해 서버와 클라이언트를 연동한 스트리밍 환경이며, 본 모듈은 서버에 포함되었다. 다음 표1은 본 모듈을 사용하기 전과 후의 초기 지연 시간을 비교를 보인다.

결과는 재생 시스템에서 서버로 스트리밍을 요청한 후, 스트리밍된 첫데이타를 재생하는데까지 소비된 시간이며, 재생기 초기화 시간 5.5~6.0 초 정도를 포함한 시간이다.

표 1. 초기 지연 시간 비교

	QCIF 10 fps 183sec	QCIF 10 fps 401sec	QCIF 10fps 723sec	QCIF 15fps 723sec
초기분석	8.2 sec	12.1sec	29.1sec	34.5sec
실시간분석	6.1 sec	6.5 sec	7.1sec	8.0 sec

5. 결론

본 논문은 사용자에게 다양한 기능을 제공하기 위해 많은 양의 메타데이터를 정의하고 있는 MPEG-4 파일 분석에 소비되는 시간을 적절으로 분산하여, 재생시 소비되는 초기 지연 시간을 감소시키는 방법을 제시하였다.

본 논문에서 제시한 방법은 메타데이터 중 미디어 데이터를 읽는데 필요한 메타데이터 인덱스만을 정리해 소유하고 있도록 설계하여, 필요할 경우 필요한 시간에 이들 인덱스를 이용해 메타 정보를 읽고 분석해서, 원하는 미디어 데이터를 재생 시스템에 공급하게 한다. 이 방법은 다수의 파일시스템 접근을 가져오나, 전체 분석 시간의 분할을 가져와서 분석 시간을 충분히 무시할 수 있는 값으로 분산시킨다.

[참고문헌]

[1] ISO/IEC FCD 14496-1:2002 Systems, 2002-3.

[2] QuickTime FileFormat Specification, Apple Computer Inc, May 1996.

[3] ISO/IEC ISO Base Media File Format, 2001.7.20

[4] 무선인터넷표준화포럼 응용서비스분과 회의록, http://www.kwisforum.org/.

[5] Wireless Multimedia Forum Recommended Technical Framework Document, http://www.wmmforum.com.