Journal of The Korea Society of Computer and Information
Vol. 19, No. 9, September 2014
www.ksci.re.kr
http://dx.doi.org/10.9708/jksci.2014.19.9.011

클라우드 컴퓨팅에서 N-스크린 서비스를 위한 동영상 트랜스 코딩 기법

임 헌 용*, 이 원 주**, 전 창 호*

Video Transcoding Scheme for N-Screen Service Based on Cloud Computing

Heon-Yong Lim*, Won-Joo Lee**, Chang-Ho Jeon*

요 약

본 논문에서는 클라우드 컴퓨팅 기반의 N-스크린 서비스를 위한 실시간 동영상 트랜스 코딩 기법을 제안한다. 이 기법은 동영상을 분할하여 하나의 인트로 블록과 재생 블록을 생성한다. 그리고 최초 서비스 요청이 오면 인트로 블록을 전송한 후 재생 블록들을 실시간으로 트랜스 코딩하여 전송한다. 이때 각 블록의 재생시간 내에 트랜스 코딩을 완료하기 위해 각 노드의 성능에 따라 재생 블록을 분할하여 할당한다. 또한, 기존 동영상 재생 기법은 실시간 재생 서비스를 위해 모든 포맷과 화질로 동영상을 변환하였다. 하지만 본 논문에서 제안한 기법은 클라이언트의 디바이스와 플랫폼에 적합한 화질의 포맷으로 동영상을 변환함으로써 기존 동영상 재생 기법에 비해 스토리지 사용량을 줄인다. 본 논문에서는 시뮬레이션을 통하여 제안한 동영상 재생 기법이 기존의 기법에 비해 N-스크린 서비스를 위한 실시간 동영상 재생에 효과적임을 보인다. 또한, 제안한 동영상 트랜스 코딩 기법이 기존 방법에 비해 스토리지 사용량이 적음을 보인다.

▸ Keywords : 클라우드컴퓨팅 N-스크린 서비스, 트랜스코딩, 재생 블록

Abstract

In this paper, we propose a real-time video transcoding scheme for N-Screen service based on cloud computing. This scheme creates an intro-block and several playback blocks by splitting the original video. And there is the first service request, after transmitting the intro-block, transmits the playback blocks that converting the blocks on real-time. In order to completing trans-coding within playback time of each block, we split and allocate the block to node according to performance of each node. Also, in order to provide real-time video playback service, the previous

•제1저자 : 임헌용 •교신저자 : 이원주
•투고일 : 2014. 6. 26, 심사일 : 2014. 7. 24, 게재확정일 : 2014. 8. 18
* 한양대학교 ERICA 캠퍼스 컴퓨터공학과(Dept. of Computer Science & Engineering, Hanyang University ERICA Campus)
** 인하공업전문대학 컴퓨터정보과(Dept. of Computer Science, Inha Technical College)

scheme convert original video into all format and resolution. However we show that the proposed scheme can reduce storage usage by converting original video into format with proper resolution suitable to device and platform of client. Through simulation, we show that it is more effective to real-time video playback for N-screen service than the previous method. We also show that the proposed scheme uses less storage usage than previous method.

▸ Keywords : Cloud Computing, N-Screen Service, Transcoding playback block

I. 서 론

최근 스마트 디바이스의 보급과 IT 기술 발전에 따라 기업들은 다양한 IT 기술을 활용하여 N-스크린 서비스를 준비하거나 시작하고 있다[1]. Apple은 아이폰, 아이패드 등 자사의 모든 디바이스에서 iCloud의 콘텐츠를 제공하고 있다[2]. 그리고 학계에서는 디바이스의 크기에 맞도록 멀티미디어를 압축하는 기술[3], 콘텐츠 전송을 효과적으로 전송할 수 있는 기술[4], 디바이스 간 협업 기술[5] 등을 연구하고 있다.

N-스크린 서비스는 하나의 콘텐츠를 다양한 디바이스에서 연속적으로 이용할 수 있는 서비스를 의미한다[6]. N-스크린 콘텐츠 서비스 중 동영상 콘텐츠 서비스는 사용자들이 가장 많이 이용하는 콘텐츠 서비스이다. 현재 국내에서는 SKT, KT, LG U+ 같은 통신 사업자들과 방송 사업자(SBS, KBS, MBC)들이 자사의 방송 콘텐츠를 활용하여 다양한 디바이스에서 시청 할 수 있도록 N-스크린 서비스 플랫폼을 개발하여 동영상 서비스를 제공하고 있다.

이러한 N-스크린 동영상 서비스는 여러 디바이스에서 동일한 콘텐츠를 연속적으로 볼 수 있어야 하며 각 디바이스에 적합한 포맷과 화질을 제공할 수 있어야 한다. 따라서 N-스크린 동영상 서비스를 제공하는 기업들은 동영상을 여러 디바이스에 적합하도록 포맷과 화질별로 변환하여 N-스크린 서비스를 제공하고 있다. 그러나 여러 포맷과 화질로 트랜스 코딩한 동영상에 대해 요청이 없다면 스토리지 자원 낭비가 발생하며, 여러 디바이스에 적합한 트랜스 코딩을 수행할 때 많은 수의 포맷과 화질로 트랜스 코딩을 수행하기 때문에 긴 서비스 준비시간이 필요하다. 트랜스 코딩은 하나의 압축된 비디오 포맷을 먼저 미처리 상태의 비디오 프레임들로 디코딩한 후 새로운 포맷으로 다시 인코딩하여 다른 압축 포맷으로 변경하는 것이다[7]. 일반적으로 트랜스 코딩은 고성능 서버급 컴퓨터 또는 소수의 클러스터를 구성하여 수행한다. 하지만 클라우드 컴퓨팅 환경에서 트랜스 코딩을 수행하면 인프라 소요 비용이 적으며 다수의 노드를 클라우드에 추가함으로써 트랜스 코딩 시간을 단축할 수 있다.

트랜스 코딩 방법은 동영상 분할 방법에 따라 균등 분할(uniform split)과 차등 분할(non-uniform split) 방법으로 분류한다. 균등 분할 트랜스 코딩 방법은 전체 트랜스 코딩 완료시간이 가장 성능이 낮은 가상 머신의 트랜스 코딩 시간에 따라 결정된다. 이것은 이기종 클라우드 환경의 전체 성능을 저하시키는 요인이 된다.

따라서 본 논문에서는 각 가상머신의 성능 차이로 발생하는 대기시간을 줄일 수 있는 차등 분할 방법을 적용하여 클라우드 컴퓨팅 기반의 N-스크린 서비스 성능을 향상시킬 수 있는 실시간 동영상 트랜스 코딩 기법을 제안한다. 이 기법은 각 가상머신의 성능에 따라 동영상을 차등 분할하여 재생 블록을 생성한다. 이때 각 가상머신의 성능을 정확하게 측정하고, 그 성능에 따른 재생 블럭의 크기를 결정함으로써 가상머신의 성능 차로 발생하는 대기시간을 줄일 수 있다. 또한 On-Demand 방식으로 트랜스 코딩하기 때문에 다양한 동영상 포맷과 화질 요청에 대한 실시간 재생 서비스를 제공함으로써 스토리지 사용량을 줄일 수 있다.

본 논문의 구성은 다음과 같다. 2장에서 관련 연구를 소개하고 3장에서는 본 논문에서 제안하는 클라우드 컴퓨팅 기반의 실시간 동영상 재생 기법에 대하여 설명한다. 4장에서는 성능평가 및 분석결과를 제시하고 5장에서 결론을 맺는다.

II. 관련 연구

클라우드 기반의 멀티미디어 서비스를 제공하는 MEC(

Media-Edge Cloud) 아키텍처는 그림 1과 같이 CPU 클러스터, GPU 클러스터, 클라우드 스토리지로 구성된다[8].

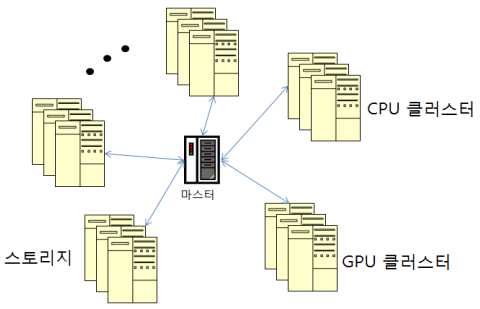

그림 1. MEC(Media-Edge Cloud) 구조
Fig. 1. Architecture of MEC(Media-Edge Cloud)

클라우드 스토리지는 오디오, 비디오, 이미지 등의 미디어 파일을 저장한다. CPU 클러스터와 GPU 클러스터는 이미지 추출, 동영상 트랜스 코딩 등의 사용자가 원하는 멀티미디어 서비스를 제공한다. 동영상 트랜스 코딩은 클러스터에 있는 여러 노드에 작업을 할당하고 병렬 처리하여 스토리지에 저장하거나 사용자에게 전송한다. 여러 클러스터를 활용하여 트랜스 코딩하기 때문에 빠른 트랜스 코딩이 가능하다. 그러나 이기종 노드로 구성된 클러스터에서 각 노드들이 동일한 작업을 수행하면 노드의 성능이 다르기 때문에 작업완료시간의 차이가 발생한다. 따라서 이기종 노드로 구성된 클러스터를 고려한 병렬처리 기법이 필요하다.

Encoding.com은 클라우드 컴퓨팅 기반에서 이미지, 오디오, 비디오 등의 미디어 파일을 사용자가 원하는 포맷으로 변환하는 서비스를 제공한다[9]. 사용자가 동영상을 클라우드에 업로드하면 사용자가 원하는 포맷으로 트랜스 코딩하여 원하는 디바이스로 전송한다. Encoding.com은 동영상의 크기가 500MB보다 작다면 가장 성능이 좋은 노드 하나를 할당하여 트랜스 코딩을 수행하기 때문에 코덱이나 화질에 따라 트랜스 코딩 속도의 차이가 발생한다.

분할(split)-병합(merge) 아키텍처는 클라우드 환경에서 동영상을 다양한 포맷으로 트랜스 코딩하기 위해 MapReduce 프레임워크를 사용한다[10]. 먼저 동영상을 64MB 단위의 chunk나 정해진 크기로 분할하여 여러 노드에게 전송한다. 분할된 파일을 전송 받은 노드는 트랜스 코딩을 수행한 후 병합 노드에게 전송한다. 병합 노드는 파일들을 병합하여 클라우드 스토리지에 저장한다. 분할-병합 아키텍처는 트랜스 코딩 소요시간을 단축시킨다. 하지만 이기종 노드로 구성된 클라우드 환경에서는 노드의 성능차이로 인해 각 노드의 트랜스 코딩 완료시간이 다르게 나타날 수 있다. 따라서 병합과정을 수행할 때, 트랜스 코딩을 수행하는 모든 노드들이 동일하지 않은 작업완료시간을 가지면 병합하는 노드에서 유휴시간(idle time)이 발생하여 전체 트랜스 코딩 소요시간이 증가하는 문제점이 있다.

기존의 트랜스 코딩 기법을 N-스크린 서비스에 적용하면 OSMU(One Source Multi Use)특징을 만족한다. 그러나 N-스크린 동영상 서비스를 위해 여러 디바이스에 적합한 포맷과 화질에 따라 트랜스 코딩을 수행하면 긴 서비스 준비 시간이 필요하다. 그리고 트랜스 코딩을 완료한 동영상 포맷과 화질에 대해 N-스크린 서비스 요청이 없을 경우, 스토리지 자원의 낭비가 발생한다. 이러한 문제를 해결하기 위해 본 논문에서는 N-스크린 서비스의 OSMU 특징을 반영하고 여러 디바이스에서 하나의 콘텐츠를 연속적으로 서비스 할 수 있는 실시간 동영상 트랜스 코딩 기법을 제안한다.

III. 클라우드 컴퓨팅 기반의 실시간 트랜스 코딩 기법

1. 실시간 동영상 트랜스 코딩 기법

본 논문에서 제안하는 동영상 트랜스 코딩 기법은 서비스 요청이 오면 동영상의 시작점부터 부분적으로 트랜스 코딩하여 사용자에게 전송한다. 각 가상머신은 성능에 따라 동영상을 분할하여 인트로 블록과 재생 블록을 생성하고, 트랜스 코딩한다.

1) 인트로 블록 생성

인트로 블록은 동영상의 시작점부터 일정구간 트랜스 코딩한 블록이다. 다양한 포맷과 화질로 트랜스 코딩하여 생성한 인트로 블록은 그림 2와 같다.

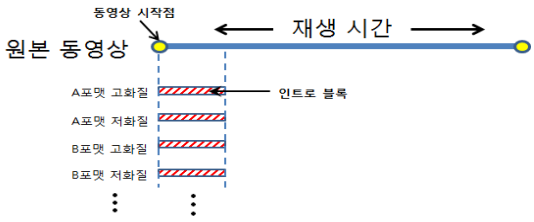

그림 2. 인트로 블록 생성
Fig. 2. Intro block creation

인트로 블록 생성 목적은 두 가지이다. 첫째, 사용자로부터 최초 재생 요청이 오면 요청 디바이스에 적합한 인트로 블록을 전송하여 즉시 동영상을 재생할 수 있도록 한다. 둘째, 인트로 블록을 제외한 나머지 부분을 트랜스 코딩할 때, 트랜스 코딩을 수행할 노드의 트랜스 코딩 성능을 측정한다.

2) 재생 블록 생성

재생 블록은 원본 동영상에서 인트로 블록을 제외한 나머지 부분을 차등 분할하여 생성한 블록이다. 재생 블록은 실시간 재생 서비스를 제공하기 위해 제한시간 내에 트랜스 코딩해야 한다. 재생 블록을 생성하는 과정은 그림 3과 같다.

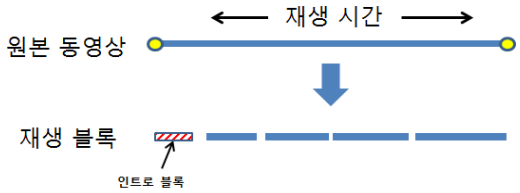

그림 3. 재생 블록
Fig. 3. Video Blocks

그림 3에서 첫 번째 재생 블록은 인트로 블록을 제외한 나머지 부분의 동영상 시작 부분부터 특정 길이만큼을 분할하여 생성한다. 그 다음 재생 블록들은 이전 블록의 길이보다 1.5배 길게 분할하여 생성한다.

3) 분산 트랜스 코딩

분산 트랜스 코딩을 수행하기 전에 다음과 같은 순서로 동작해야 한다. 각 단계는 이전 블록의 재생시간 안에 다음 블록을 트랜스 코딩하기 위해 요구되는 초당 트랜스 코딩 양을 구하고, 참여할 노드를 구성한 후, 블록을 차등 분할한다.

본 논문에서 제안하는 기법은 이기종 노드로 구성된 클라우드 컴퓨팅 환경을 전제로 하고 있기 때문에 각 노드들의 성능이 동일하지 않다. 따라서 블록을 균등 분할하여 노드들에게 할당한다면 블록의 트랜스 코딩 소요시간은 성능이 가장 낮은 노드의 작업완료시간과 동일하기 때문에 상대적으로 성능이 높은 노드에서는 유휴시간(idle time)이 발생한다. 이로 인해 트랜스 코딩 완료시간이 제한시간을 초과하여 실시간 재생 서비스가 불가능할 수 있다. 이러한 문제점을 방지하기 위해 각 노드들의 성능을 고려하여 동영상을 차등 분할하여 트랜스 코딩한다. 분산 트랜스 코딩 과정은 그림 4와 같다.

과정 1: 초당 트랜스 코딩 양을 구한다.
과정 2: 트랜스 코딩에 참여할 노드를 구성한다.
과정 3: 각 노드의 성능에 따라 블록을 차등 분할하여 할당 한다.
과정 4: 마지막 블록까지 분산 트랜스 코딩을 반복한다.

그림 4. 분산 트랜스 코딩 과정
Fig. 4. Distributed transcoding Process

그림 4의 과정 1에서 초당 트랜스 코딩량(Mbyte/sec)은 동영상 정보를 나타내는 평균 비트(average bit rate)를 통해 구한다. 이전 블록의 재생시간 안에 트랜스 코딩을 완료하기 위해 요구되는 초당 트랜스 코딩량은 다음 식(1)로 구할 수 있다.

$$\text{트랜스코딩량}(Mbyte/\text{sec}) = \frac{P_{n+1} \times AvgBitrate}{P_n - R} \quad \cdots\cdots (1)$$

식(1)에서 P_n은 사용자가 시청 중인 블록의 재생시간이며 R은 전송준비 시간이다. P_{n+1}은 다음 블록의 재생시간을 의미하며 AvgBitrate는 트랜스 코딩 할 블록의 평균 비트 레이트를 의미한다.

과정 2에서 트랜스 코딩에 참여할 노드를 구성하는 알고리즘은 그림5와 같다.

먼저 요구되는 초당 트랜스 코딩량(Mbyte/sec)과 인트로 블록을 생성할 때 측정한 노드들의 초당 트랜스 코딩량의 합(Mbyte/sec)을 비교한다. 만약 노드들의 초당 트랜스 코딩량이 요구되는 초당 트랜스 코딩량 보다 작다면, 같거나 커질 때까지 노드를 추가한다. 반면에 노드들의 초당 트랜스 코딩량이 요구되는 초당 트랜스 코딩량 보다 크다면, 요구되는 초당 트랜스 코딩량을 만족하는 선에서 노드를 작업에서 제외시킨다.

과정 3에서는 노드의 성능에 따라 블록을 차등 분할하여 각 노드에게 할당한다. 식(2)와 같이 각 노드의 성능에 따라 블록을 차등 분할하여 할당한다.

$$P_{n-1} - R = \frac{S_1}{C_1} = \frac{S_2}{C_2} = \ldots = \frac{S_{n-1}}{C_{n-1}} = \frac{S_n}{C_n} \quad \cdots\cdots\cdots\cdots (2)$$

여기서 Pn - R은 사용자가 현재 시청 중인 블록의 재생시간을 의미한다. 따라서 각 노드가 수행할 총 트랜스 코딩 시간과 같다. Si는 차등 분할된 블록 크기를 나타내며 Ci는 각 노드의 성능(Mbyte/sec)을 나타낸다. 따라서 식(2)를 바탕으로 노드 i 에 차등 할당할 파일의 크기는 다음 식(3)으로 정한다.

$$S_i = (P_{n-1} - R) \times C_i \quad \cdots\cdots (3)$$

차등 분할할 크기가 정해지면 블록을 분할하여 각 노드에게 할당한다. 모든 노드에서 할당된 작업이 완료되면 그 결과를 병합하여 블록의 트랜스 코딩을 완료한다.

```
Nodeset = {a,b...,x,z} : Performace of Nodes
Allocateset  = { 0 }
RequirePeformance = q;
Sum = 0;

If( ∑_{i=0}^{m} Node_i ≥ RequirePerformance)
{
      sort( Nodeset )
      for(int i =0; Nodeset .length; i++)
      {
          int c = Nodeset
          if(RequirePerformance > c+ sum)
          {
               sum = sum + c
               Allocate.Add(Nodei )
               Nodeset.Remove(Nodei )
          }
          else if(RequirePerformance <= c + sum)
          {
               sum = sum + c
               Allocate.Add(Nodei )
               Nodeset.Remove(Nodei )
               break;
          }

      }
}
Else
{
   while(RequirePerformance > sum)
   {
       Allocate.Add(Nodenew)
       sum = ∑_{i=0}^{m} Node_i + Nodenew
   }
}
```

그림 5. 트랜스 코딩 노드 수 결정 알고리즘
Fig. 5. Algorithm that determines the number of Transcoding nodes

과정 1: 동영상을 분할하여 인트로 블록과 재생 블록을 생성한다. 그리고 생성한 인트로 블록을 트랜스 코딩할 노드들에게 전송하여 포맷과 화질별로 트랜스 코딩한다.
과정 2: 각 노드의 성능 정보를 얻기 위해 각 노드들의 트랜스 코딩 소요시간을 측정한다.
과정 3: 사용자로부터 재생 요청이 오면 요청한 포맷과 화질에 적합한 인트로 블록을 사용자에게 전송한 후 다음 재생 블록을 트랜스 코딩한다. 이때 다음 재생 블록의 트랜스 코딩은 사용자가 인트로 블록의 재생을 끝마치기 이전에 완료되어야한다. 그렇지 않으면 인트로 블록의 재생이 끝난 후, 다음 재생 블록의 트랜스 코딩이 끝날 때 까지 대기시간이 발생하여 실시간 동영상 재생 서비스를 제공할 수 없다.
과정 4: 마지막 블록까지 분산 트랜스 코딩을 반복한다.

그림 6. 실시간 동영상 재생기법
Fig. 6. Real-time video playback scheme

4) 실시간 동영상 재생

동영상 재생 서비스를 제공하기 위해서는 먼저 인트로 블록과 재생 블록을 생성한 후 사용자로부터 서비스 요청이 오면 인트로 블록을 전송하고 재생 블록을 트랜스 코딩하여 서비스를 제공한다. 이러한 동영상 재생 기법은 그림 6과 같다.

과정 3에서 사용자가 연속적으로 동영상 재생 서비스를 받기 위해서는 블록 n이 재생되는 동안 블록 n+1은 트랜스 코딩이 완료되어야 한다. 각 블록의 트랜스 코딩 소요시간은 그림 7에서 설명한다.

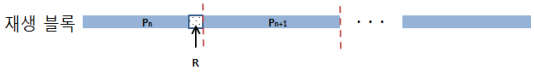

그림 7. 블록의 재생시간과 전송시간
Fig. 7. Play time and transmission time of block

그림 7에서 P_n은 블록 n의 재생시간을 의미하고 R은 블록을 다운로드 받는 전송 시간을 의미한다. 여기서 전송 시간이란 사용자가 연속적으로 동영상을 재생 할 수 있도록 블록을 다운로드 받는 시간을 의미한다. 따라서 사용자가 재생 중인 블록 n의 재생 시간 동안 블록 n+1을 트랜스 코딩하기 위해서는 Pn-R 시간동안 트랜스 코딩을 완료해야 한다.

사용자는 동영상 재생 중 재생시점을 변경할 수 있다. 또한, 동영상 재생 중 다른 디바이스로 전환하여 동영상 서비스를 요청할 수 있다. 이러한 두 가지 경우를 처리하는 방법은 다음과 같다.

첫째, 사용자로부터 재생시점 변경요청이 들어온 경우, 재생시점이 포함되어 있는 재생 블록이 이미 트랜스 코딩되어 있는지 확인한다. 만약 요청하는 시점이 포함된 재생 블록이 트랜스 코딩되어 있다면 사용자에게 요청하는 시점이 포함된 재생 블록을 전송한다. 그러나 트랜스 코딩이 되어 있지 않다면 요청된 재생시점부터 다시 블록을 분할하고 분할된 첫 블록의 트랜스 코딩을 우선적으로 수행하여 사용자에게 전송한다. 그 이후 과정은 재생 서비스를 제공하는 방법과 동일하다.

둘째, 사용자가 동영상을 재생하던 중 디바이스를 변경할 경우, 스토리지에서 변환한 디바이스에 적합하도록 트랜스 코딩된 재생 블록이 존재하는지 확인한다. 만약 재생 블록이 존재하면 사용자에게 전송한다. 그러나 재생 블록이 존재하지 않는다면 사용자로부터 요청한 디바이스에 적합한 포맷과 화질로 트랜스 코딩한다. 이때 이전 포맷의 재생시점과 동일한 재생시점부터 동영상을 분할하여 재생 블록을 생성하며 분할된 첫 블록부터 트랜스 코딩한다. 그 이후 과정은 재생 서비스를 제공하는 방법과 동일하다.

IV. 성능평가

본 논문에서 제안한 동영상 트랜스 코딩 기법은 연속적인 동영상 재생 서비스를 제공하고, 스토리지 사용량을 줄일 수 있다. 따라서 성능평가에서는 클라우드 컴퓨팅의 N-스크린 서비스에서 연속적인 동영상 재생 서비스가 가능한지와 기존 방법과 비교하여 스토리지 사용량을 줄일 수 있는지를 검증한다. 이를 위해 동영상의 크기와 포맷을 다르게 하여 각 블록의 재생시간과 다음 블록을 트랜스 코딩하는데 소요되는 시간을 비교한다. 그리고 서비스 요청 빈도를 달리하여 스토리지 사용량을 측정한다.

1. 성능평가 환경

본 논문에서는 이기종 노드 8대로 구성된 클러스터를 이용하여 성능평가를 수행한다. 클러스터는 마스터 노드, 클라이언트 노드, 스토리지 노드, 그리고 트랜스 코딩 노드로 구성한다. 각 노드의 성능은 표 1과 같다.

표 1. 클러스터 노드 사양
Table 1. Cluster node specification

노드 역할	성 능	비고
클라이언트	Intel Pentium G640, 2.8GHz, Memory: 2GB	
마스터	Intel Core i5 3.3GHz, 4GB	
트랜스 코딩	Intel Core2 Duo 1.66GHz, Memory: 1GB	노드 A
	Intel Core2 Duo 2.93GHz, Memory: 2GB	노드 B
	Intel Celeron 1.8 GHz, 2GB	노드 C
	Intel Pentium 2.8GHz, Memory: 2GB	노드 D
	Intel Core i5 3.3GHz, Memory: 4GB	노드 E
스토리지	Intel Core Quad Q8200 2.33GHz, Memory: 3GB	

표 1에서 마스터 노드는 동영상을 블록으로 분할하고 블록을 노드에 할당하다는 기능을 한다. 클라이언트 노드는 동영상 서비스를 요청하는 기능을 한다. 스토리지 노드는 원본 동영상과, 트랜스 코딩된 동영상을 저장한다. 그리고 트랜스 코딩 노드는 마스터로부터 블록을 할당받아 트랜스 코딩을 수행한다. 트랜스 코딩에는 FFmpeg라이브러리를 사용한다[7]. 클러스터의 네트워크 대역폭은 1 Gbits으로 설정한다.

성능평가는 표 2의 동영상을 각 노드의 성능에 따라 차등 할당하여 H.264과 WebM 포맷으로 트랜스 코딩 소요시간을 측정하였다. 그리고 각 포맷에 대해 480p, 720p 해상도로 트랜스 코딩하는 소요시간도 측정한다. H.264와 webM은 현재 널리 사용되고 있는 포맷으로 H.264는 Apple에서, WebM 포맷은 Google에서 채택하여 사용하고 있다.

표 2. 동영상 정보
Table 2. Video Information

포맷	크기	비트 레이트	재생시간
avi	500 MB	1375 kb/s	48분 50초
avi	1.0 GB	10600 kb/s	12분 55초
mp4	1.31 GB	1546 kb/s	108분 53초
mp4	2.17 GB	1637 kb/s	185분 02초

각 동영상을 균등하게 분할하여 트랜스 코딩한 결과는 표 3과 같다.

표 3. 균등 분할 트랜스 코딩 수행결과
Table 3. Result of Transcoding by Equal Split

동영상 크기 (균등 분할 크기)	포맷	화질	트랜스 코딩 소요시간
500 MB (100 MB)	H.264	고	42분
		저	18분
	WebM	고	62분
		저	30분
1.0 GB (204.8 MB)	H.264	고	87분
		저	37분
	WebM	고	126분
		저	62분
1.17 GB (238.8 MB)	H.264	고	102분
		저	43분
	WebM	고	147분
		저	72분
2.21 GB (451.6 MB)	H.264	고	193분
		저	81분
	WebM	고	279분
		저	136분

트랜스 코딩을 완료하는데 소요되는 시간은 노드 A에 할당된 작업의 완료시간과 동일한 것을 확인할 수 있다. 이것은 성능이 다른 노드들에게 작업을 균등한 크기로 분할하여 할당하면 그 작업완료시간은 성능이 가장 낮은 노드 A의 작업완료시간과 동일하기 때문이다.

2. 성능평가 결과분석

동영상을 연속적으로 실시간 재생하기 위해서는 이전 블록의 재생시간 안에 다음 블록에 대해 트랜스 코딩을 완료해야

한다. 따라서 본 논문에서 제안한 동영상 실시간 재생 기법은 이전 블록의 재생 시간 안에 다음 블록을 트랜스 코딩을 완료할 수 있는지 검증한다.

500 MB 동영상을 실시간 재생하기 위해 8개 이기종 노드에 분산 트랜스 코딩하면서 소요시간을 측정한 결과는 그림 8과 같다.

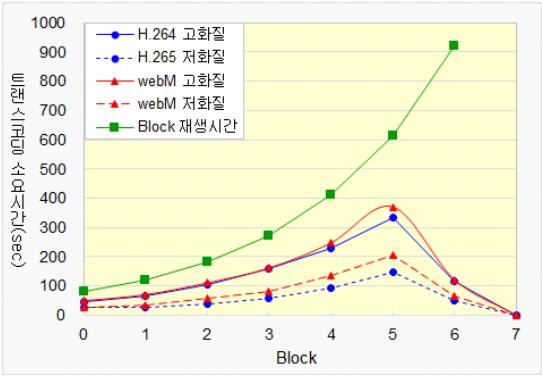

그림 8. 실시간 트랜스 코딩 결과(500MB)
Fig. 8. Result of Real-time Transcoding(500MB)

그림 8에서 블록 0는 인트로 블록을 의미하며 블록 1~7은 재생블록이다. 인트로 블록은 생성할 때 블록의 크기가 30 MB 이하로 제한한다. 각 블록의 재생시간 안에 다음 블록을 트랜스 코딩을 완료해야만 실시간 동영상 재생 서비스가 가능하다. 그림 8을 살펴보면 모든 포맷과 화질의 동영상이 블록 재생시간 내에 트랜스 코딩을 완료함을 알 수 있다. 따라서 본 논문에서 제안한 기법을 사용하면 연속적인 실시간 재생 서비스가 가능함을 알 수 있다.

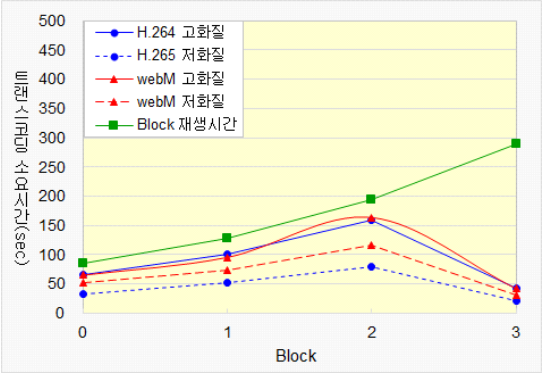

그림 9. 실시간 트랜스 코딩 결과(1GB)
Fig. 9. Result of Real-time Transcoding(1GB)

1GB 동영상을 실시간 재생하기 위해 8개 이기종 노드에 분산 트랜스 코딩하면서 소요시간을 측정한 결과는 그림 9와 같다. 1GB 동영상은 재생시간이 짧고 높은 비트레이트를 가지는 고화질의 뮤직 비디오이다. 그림 9을 살펴보면 모든 포맷과 화질의 동영상이 블록 재생시간 내에 트랜스 코딩을 완료함을 알 수 있다. 그리고 블록 0에서 H.264 포맷 고화질과 WebM 포맷 고화질은 트랜스코딩을 완료한 시간이 블록의 재생시간에 근접한 것을 볼 수 있다.

1.31GB 동영상을 실시간 재생하기 위해 8개 이기종 노드에 분산 트랜스 코딩하면서 소요시간을 측정한 결과는 그림 10과 같다.

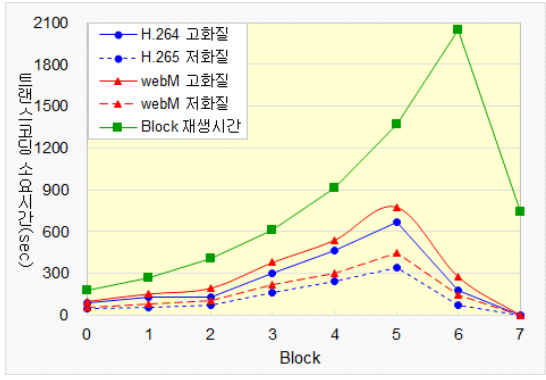

그림 10. 실시간 트랜스 코딩 결과(1.31GB)
Fig. 10. Result of Real-time Transcoding(1.31GB)

그림 10을 살펴보면 모든 포맷과 화질의 동영상이 블록 재생시간 내에 트랜스 코딩을 완료하기 때문에 연속적인 실시간 재생 서비스가 가능함을 알 수 있다.

2.17GB 동영상을 실시간 재생하기 위해 8개 이기종 노드에 분산 트랜스 코딩하면서 소요시간을 측정한 결과는 그림 11과 같다. 2.17GB 동영상은 재생시간이 긴 야구 시합 동영상이다. 야구시합 동영상은 움직임이 많기 때문에 블록마다 높은 비트 레이트를 지니고 있다.

그림 11을 살펴보면 블록 2와 블록 5에서 H.264 고화질과 WebM 고화질의 트랜스 코딩 소요시간이 블록의 재생시간에 근접함을 볼 수 있다. 그 이유는 초당 트랜스 코딩 량(Mbyte/sec)과 동일하게 수행할 수 있는 노드를 구성할 때 오차가 발생했기 때문이다. 이러한 오차로 인해 블록의 재생시간보다 트랜스 코딩 시간이 길어지는 문제점이 발생할 수 있다. 이러한 문제점을 방지하기 위해서는 초당 트랜스 코딩 량을 구할 때 전송 준비시간을 길게 설정하면 오차에 대한 실시간 서비스 재생을 보장할 수 있다. 이러한 성능평가 결과에

서 모든 포맷과 화질의 동영상이 블록 재생시간 내에 트랜스 코딩을 완료함에 따라 본 논문의 제안한 기법을 사용하면 실시간 재생 서비스가 가능함을 알 수 있다.

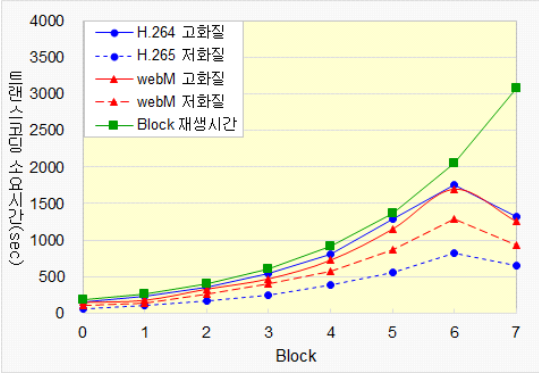

그림 11. 실시간 트랜스 코딩 결과(2.17GB)
Fig. 11. Result of Real-time Transcoding(2.17GB)

그림 12는 7일 동안 동영상 재생요청 횟수가 1000번, 5000번, 10000번, 50000번일 때 스토리지 사용량을 측정한 결과이다. 성능평가에 사용된 동영상의 수는 500개이며 동영상 크기의 총합은 625GB이다. 재생 요청할 동영상과 트랜스 코딩 할 포맷, 화질은 균등분포를 따르는 랜덤함수로 결정한다.

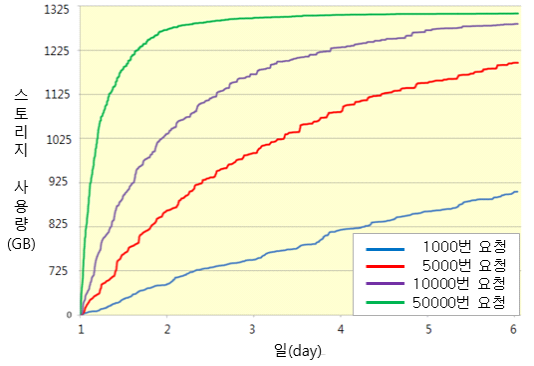

그림 12. 스토리지 사용량
Fig. 12. Storage Usage

그림 12를 살펴보면 재생 요청 횟수가 10,000일때에 비해 50,000번일 경우 스토리지 사용량은 빠르게 증가함을 볼 수 있다. 이는 재생 요청 빈도가 많아질수록 트랜스 코딩되는 동영상의 수도 증가하기 때문이다. 기존의 방법은 동영상 재생 서비스를 수행하기 전에 동영상에 대한 모든 포맷과 화질별로 트랜스 코딩한 동영상을 저장하기 때문에 고정적으로 1312GB 저장 공간이 필요하다. 하지만 본 논문에서 제안한 기법은 재생 요청된 포맷과 화질에 대해서만 트랜스 코딩하기 때문에 스토리지 사용량은 요청 횟수에 비례하여 증가한다.

클라우드 컴퓨팅에서는 사용한 자원만큼 비용을 지불한다. 따라서 본 논문에서 제안한 기법은 서비스 요청에 따라 스토리지를 사용하기 때문에 기존 방법에 비해 스토리지 사용 비용을 줄일 수 있다.

V. 결 론

본 논문에서는 클라우드 컴퓨팅 기반의 N-스크린 서비스를 위한 실시간 트랜스 코딩 기법을 제안하였다. 이 기법은 동영상을 차등 분할하여 인트로 블록과 재생 블록을 생성한다. 그리고 최초 서비스 요청이 오면 인트로 블록을 전송한 후 재생 블록을 트랜스 코딩하여 실시간으로 전송한다. 동영상을 연속적으로 실시간 재생하기 위해서는 이전 블록의 재생시간 안에 다음 블록을 트랜스 코딩해야한다.

본 논문에서는 500MB, 1GB, 1.31GB, 2.17GB 동영상을 분할하여 8개 이기종 노드에 분산 트랜스 코딩하면서 이전 블록의 재생 시간 내에 다음 블록을 트랜스 코딩 할 수 있는지 검증하였다. 그 결과 본 논문에서 제안한 동영상 트랜스 코딩 기법을 사용하면 연속적인 동영상 실시간 재생 서비스가 가능함을 알 수 있었다. 또한, 제안한 기법은 On-Demand 방식으로 트랜스 코딩하기 때문에 기존 방법에 비해 스토리지 사용량을 줄일 수 있음을 알 수 있었다.

향후 연구과제는 이기종 노드 수가 많은 상업용 클라우드 컴퓨팅 환경에서 제안한 기법의 성능을 평가해 보는 것이다.

참고문헌

[1] Won Joo Lee, Jung Pyo Lee, and Y. I. Yoon, "A Design and Implementation of N-Screen Emulator Based on Cloud," Journal of The Korea Society of Computer and Information, Vol. 18, No. 3, pp. 11-18, Mar. 2013.

[2] U. K. Park, J. G. Kim, "NScalable Video Coding Standard for N-Screen Service," Communications of the Korean Institute of Information Scientists and Engineers, Vol. 29, No. 7, pp. 38-44, July 2011.

[3] J. H. Yang, S. H. Song, and J. K. Choi, "Standardization for N-Screen Environment over NGN," Communications of the Korean Institute of Information Scientists and Engineers, Vol. 29, No. 7, pp. 45-54, July 2011.
[4] Y. T. Seok, D. W. Lee, and H. W. Lee, "Session Cloud Service Framework -Web Service Based Session Cloud Service Framework for Simultaneous Use of Multiple Devices-," HCI Korea 2012, pp.188-101, Jan. 2012.
[5] H. S. Kim, H. J. Lee, and G. S. Cho, "Overview of Status and R&D Issues on N-Screen Service," Communications of the Korean Institute of Information Scientists and Engineers, Vol. 29, No. 7, pp. 9-15, July 2011.
[6] Y. I. Yoon, B. Kim, "N-Screen Service Standardization Based on Platform Type," KSCI Review, Vol. 20, No. 1, pp. 1-9, June 2012.
[7] E. Ohwovoriole and Y. Andreopoulos, "Rate-distortion performance of contemporary video codecs: Comparison of Google/WebM VP8, AVC/H.264, and HEVC TMuC," LENS Symp., London, Sept. 2010.
[8] W. Zhu, C. Luo, J. Wang and S. Li, "Multimedia Cloud Computing," IEEE Signal Processing Magazine, Vol. 28, Issue:3, pp.59-69, May 2011.
[9] *Encoding.com, 2012.7, http://www.encoding.com/*
[10] R. Pereira, M. Azambuja, K. Breitman, M. Endler, "An Architecture High Performance Video Processing in the Cloud," In Proc. of 2010 IEEE 3rd International Conference on Cloud Computing, pp. 482-489, 2010.

저 자 소 개

임 헌 용

2010: 인하공업전문대학 컴퓨터정보과 전문학사
2013~현재: 한양대학교 컴퓨터공학과 공학석사
관심분야: 성능분석, 클라우드컴퓨팅, 모바일 컴퓨팅, N-스크린 서비스,
Email: heonyong.lim@gmail.com

이 원 주

1989: 한양대학교 전자계산학과 공학사.
1991: 한양대학교 컴퓨터공학과 공학석사.
2004: 한양대학교 컴퓨터공학과 공학박사.
현 재: 인하공업전문대학 컴퓨터정보과 교수.
관심분야: 병렬처리시스템, 모바일컴퓨팅, 성능분석, 클라우드컴퓨팅, N-스크린 서비스
Email: wonjoo2@inhatc.ac.kr

전 창 호

1977: 한양대학교 전자공학과 학사.
1982: Cornell University 컴퓨터공학과 석사.
1986: Cornell University 컴퓨터공학과 박사.
1977-1979: 전자통신연구소 연구원.
현 재 : 한양대학교 전자컴퓨터공학부 교수.
관심분야: 병렬처리시스템, 성능분석, Grid 컴퓨팅, 클라우드 컴퓨팅
Email: chj5193@hanyang.ac.kr