Journal of Knowledge Information Technology and Systems

ISSN 1975-7700

http://www.kkits.or.kr

The Performance Test Method Execution of Distributed File System for Segmented Media File

Myeong-Ki Jeong[1], Jae-Woong Kim[*2]

[1]*Software Research & Development Institute, Keysco E&M*
[2]*Department of Computer Engineering, Kongju National University*

ABSTRACT

Recently, cloud file system is distributed in the cluster storage media files to the specified size. By the way, media streaming method has been widely used for MPEG-DASH protocol or HTTP Live Streaming protocol because there are many mobile devices of a low performance system. The media file encoded with MPEG-DASH protocol or HTTP Live Streaming protocol system is divided into a segmented media files. Cloud file system stores by distributing a segmented media file to the cluster. Thereafter, the clients will request the playback of the segmented media files. Then the server reads its role as a gateway files stored on a cluster. And the gateway server provides the segmented media files to the clients. This way is the same as how to provide a single media file in server storage. The bottleneck of network traffic occurs on the gateway server. In this paper, we implemented a segmented media files distributed file system proposed in previous research. And check the stability of the segmented media files distributed file system. The method to establish a test device configuration and testing process. Then measure the network traffic of each of the clusters. The result confirm the stability of the segmented media files distributed file system.

KEYWORDS: MPEG-DASH, HTTP live streaming, Cloud, Distributed file system, Metadata modeling

ARTICLE INFO: Received 20 June 2016, Revised 12 August 2016, Accepted 12 August 2016.

[*]Corresponding author is with the Department of Computer Engineering, Kongju National University, 275, Budae-dong, Seobuk-gu, Cheonan-si, Chungcheongnam-do, (1223-24, Cheonandae-ro), 31080, KOREA.
E-mail address: jwkim@kongju.ac.kr

1. 서 론

현존하는 클라우드 파일 시스템들은 범용의 데이터들을 분산 저장하기 위해 연구된 분산파일 시스템을 사용하고 있다. 또한 클라우드 서버에 저장된 데이터를 접근하기 위해서 일종의 게이트웨이를 사용한다. 그런데 단일 파일로 구성된 동영상 미디어의 경우 화면전환(scene-cut)등의 미디어 특성을 고려하지 않고 단순히 파일 크기 기준으로 분산 저장하고, 추후 클라이언트들이 해당 미디어 파일을 재생하는 경우, 클라우드 시스템의 게이트웨이를 통해 재생하므로 트래픽이 한 곳에 집중될 수 도 있다는 점이 단점으로 부각된다. 또한 분할된 미디어 파일 즉, 청크(chunk)를 사용하는 HTTP Live Streaming 방식 또는 MPEG-DASH 방식을 이용하는 경우에는 클라우드 파일 시스템이 분할된 미디어 파일보다 훨씬 큰 크기의 할당량에 배치되므로 클라우드 내부의 저장용량 낭비를 초래하게 된다. 이에 이전 논문에서 미디어 분할 파일을 위한 분산시스템 메타데이터 모델링[1]을 제시한바 본 논문에서는 해당 모델링 시스템을 검증하기 위해 클러스터와 테스트용 클라이언트를 구성하고, 테스트 프로세스를 수립하여 기존 제안된 분할된 미디어 파일을 위한 분산시스템의 안정성을 검사할 수 있는 클러스터의 네트워크 전송률을 테스트하고자 한다.

2. 이전 연구

2.1 미디어 분할 파일 분산시스템 메타데이터 모델링

<그림 1>은 분산파일시스템의 대표적인 구글 파일 시스템(Google File System : 이하 GFS)에서 주로 사용되는 메타데이터이다. “File namespace”는 GFS 청크서버 상에 저장된 파일의 이름을 영역별로 기록한 것이며, “Access Control Information”은 해당 파일에 접근하는 방법을 기록하며, 전체 파일이 어떤 청크 파일로 구성되어 있는지를 “Mapping Information”으로 관리한다. “Chunks Locations”는 해당 청크가 어느 청크서버에 위치하는지를 기록한다[2].

- File namespace
- Access control information
- Mapping information between the file and the chunk
- Chunk locations

그림 1. 구글 파일시스템의 마스터 메타데이터
Figure 1. Google File System master metadata

<그림 2>는 미디어 분할 파일 분산시스템 메타데이터이다. “Media File Sequence Number”는 분할된 미디어 파일의 순서번호를, “Relay Server Sequence Number”는 GFS 청크서버에 해당하는 클러스터의 고유한 ID를 일련번호로 나타낸 것이며, “Segment File Name”은 GFS의 “File namespace”와 같은 기능을 하여 파일이름을 연속적인 순서번호로 인덱싱되어 있다. “Duplicate Segment Flag”는 분할 미디어 파일이 몇 개의 서버로 분배되었는지를 나타낸다[1].

- Media File Sequence Number
- Relay Server Sequence Number
- Segment File Name (or Segment Index Number)
- Duplicate Segment Flag

그림 2. 미디어 분할 파일 분산 시스템 메타데이터
Figure 2. Media Segment File Distribution System metadata

<그림 3>은 분할 파일 메타데이터의 물리모델을 나타낸 것이다. SB_MEDIA_SEGMENT_LIST는 미디어 파일을 분할하여 각 분할된 파일들을 클러스터에 분배한 것을 모델링한 것이다[3]. 이 테이블에서

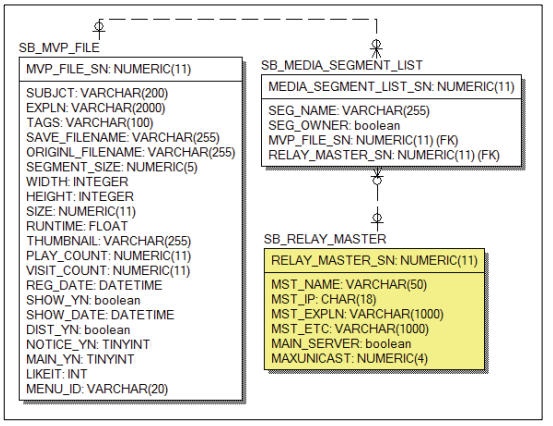


그림 3. 분할 파일 메타데이터 물리모델
Figure 3. Segment file metadata's physical model

SEG_NAME은 분할된 미디어의 파일 이름이고 그 원본 파일은 SB_MVP_FILE의 MVP_FILE_SN을 외부키로 가진다. 또한 각 분할된 파일들이 저장되는 클러스터는 SB_RELAY_MASTER의 RELAY_MASTER_SN과 1:N으로 관계한다. 원본 미디어 영상파일 정보는 SB_MVP_FILE 테이블로, 클러스터의 정보는 SB_RELAY_MASTER 테이블로 모델링되었다.

2.2 클러스터 미디어 배포 프로세스

영상미디어를 배포하기 위해 사전에 MPEG-4 형태로 인코딩 한 후에 파일을 서버에 업로드 한다. 그리고 인코딩된 파일을 MPEG-DASH[4][5] 또는 HTTP Live Streaming[6](이하 HLS) 방식으로 전송하기 위해 분할 파일(Segment file)을 생성한다[7].

MPEG-DASH 또는 HLS는 분할된 파일 이름들의 목록을 가지고 있는 인덱스 파일 형태로 관리된다. 그 중 HLS은 MPEG-2 Transport Stream으로 인코딩된 확장자명이 .TS 인 파일들의 목록으로 재생목록 메타데이터(확장자명 .m3u8)을 생성한다[7].

이 후 메타데이터 파일과 분할된 미디어 파일들은 웹서버에서 서비스된다. 해당 미디어들을 재생하고자 할 때에는 재생 목록 메타데이터를 웹서버에 요청하여 다운로드한다. 다운로드한 후 재생 목록 메타데이터 내에 들어 있는 분할된 미디어 파일 목록을 이용하여 각각의 분할 파일도 웹서버에서 다운로드한다. 분할 파일을 연속으로 다운로드 받고 계속 재생하여 하나의 완성된 미디어를 재생하게 된다[8].

영상 미디어 파일의 분할은 검색(seeking) 시에 이동하려고 10초 단위로 영상을 scene-cut을 하는 것이 일반적이나 해당 영상을 초단위로 검색 시에는 10초 단위의 범위가 커서 3초 단위로 분할하였다. 3초 이하로 분할하는 경우, 너무 많은 분할 파일이 존재하여 효율이 나빠지는 단점이 있어서 3초를 기준으로 하였다[3].

```
result_RPC_ID ← get_RPC_ID_list();
count_RPC ← count_RPC( result_RPC_ID );
count_ts ← count_segment_file();
nScale ← get_backup_scale();
nCnt ← 0;
for ( i = 0; i < count_ts; i++ ) {
  for ( j = 0; j < nScale; j++) {
    segment_name ← filename + i + ".ts";
    save_segment( segment_name, result_RPC_ID );
    if ( nCnt > count_RPC -1 ) nCnt ← 0;
  }
}
```

그림 4. 분할 파일 분배 알고리즘
Figure 4. Segment file distribution Algorithm

분할 된 부분 파일은 <그림 4>의 분배 알고리즘에 의해 배치될 여러 중계 서버를 할당 받는다. <그림 4>에서 result_RPC_ID는 해당 시점의 활성화되어 있는 클러스터들의 ID를 목록으로 저장한다. 유지보수 목적 또는 기타 목적으로 정지되어 있는 클러스터는 제외한 목록이다. 클러스터 목록의 생성은 미디어 저장을 최우선으로 하여야 하므로 해당 클러스터의 저장소 용량이 높은 것부터, 해당

서버의 네트워크 사용량이 낮은 것부터, 해당 서버의 CPU 사용률이 낮은 것부터, 메모리 사용률도 낮은 것부터 정렬하여 목록을 작성한다. count_ts는 분할된 미디어 파일의 총 개수를, nScale은 한 개의 분할 미디어 파일을 몇 개의 클러스터에 배분할 것인지를 나타내는 사용자 입력 변수이다[3][9].

2.3 클러스터 내 분배 파일 목록 생성 프로세스

분할 미디어 파일들이 클러스터에 분배, 배포 된 후 클라이언트가 미디어의 재생을 웹서버에 요청을 하게 되면 웹서버는 해당 미디어의 재생을 위한 기존 재생 목록 메타데이터를 이용해 새로운 재생 목록을 만들어 클라이언트에 전달한다[3].

```
Segment_File_Count <- ReadCountofSegments();
Client_Relay_Server_SN <- getClientRelayServerSN();

Media <- get_orginal_index_file( Media_Name );
for (s = 0; s < Segment_File_Count; s++)  {
  Segment_Relay_Server_List <- getSegmentList(s)
  Min_Relay_Server_SN
    <- FindRS( Client_Relay_Server_SN,
               Segment_Relay_Server_List );
  Min_Relay_Server_IP
    <- getRelayServerIP( Min_Relay_Server_SN )
  Media
    <- StringReplace( Media, s,
                      Min_Relay_Server_IP )
}
```

그림 5. 분배 파일 목록 생성 알고리즘
Figure 5. Distribution play list generation Algorithm

<그림 5>는 HLS의 m3u8 인덱스 파일을 해당 시점에서 최적인 상태의 클러스터를 선택하여 재설정하여 클라이언트에게 제공하는 재생 목록을 생성하는 알고리즘이다[10]. 이 알고리즘은 클라이언트가 요청한 해당 미디어의 분할 파일 개수 Segment_File_Count를 읽어와 해당 분할 파일 각각에 대해 <그림 4>의 nScale과 같이 기존에 지정된 개수의 분배된 클러스터의 목록을 취득한다. 해당 클러스터 목록 중에서 클라이언트가 소속된 지역의 서버를 기준으로 분배 저장된 클러스터를 최소 공통조상과 SNMP로 취득된 각 클러스터의 네트워크 트래픽, CPU 사용량, Memory 사용량 값을 이용한 클러스터 탐색을 통해 최적의 클러스터를 탐색한다. <그림 5>에서 최적의 클러스터를 탐색하는 함수는 FindRS이고 이 때 전달되는 변수인 Client_Relay_Server_SN은 미디어를 요청한 클라이언트가 소속된 지역의 클러스터의 고유번호(Identify Number)이고 Segment_Relay_Server_List는 분할 파일이 분배된 클러스터들의 ID 리스트이다. 이렇게 탐색된 클러스터의 ID를 이용해 getRealyServerIP 함수에서 실제 물리적인 IP 주소를 추출하고 해당 IP 주소를 m3u8 미디어 목록 파일의 해당 분할 미디어 파일 앞에 http 주소 형태로 추가하여 준다. 이렇게 모든 분할 파일에 대해 클러스터를 선택하고 이를 목록화 하여 클라이언트에 전달하여 재생하도록 한다.

3. 성능 평가

3.1 테스트용 클러스터 시스템의 구성 및 사양

클러스터 시스템 성능 평가를 위한 시스템의 구성은 <그림 6>과 같이 웹서버와 클러스터 그리고 각 클러스터의 성능 평가를 수행하는 클라이언트 PC로 구성되어진다.

기존 클라우드 파일 시스템 또는 분산 파일 시스템의 경우에는 GFS에서는 GFS master, 하둡(HADOOP)에서는 name node라 하는 게이트웨이 역할을 수행하는 서버를 이용한다. 각 master 또는

name node에 자원을 요청할 시에는 해당 자원이 할당되어 있는 청크서버 또는 data node를 클라이언트에 매칭해 주고 자원을 직접 접근하여 가져가도록 한다.

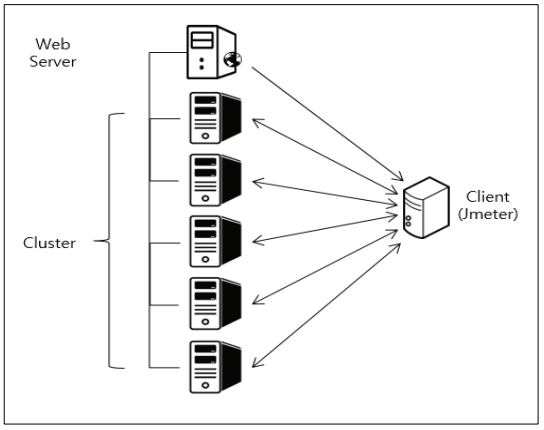


그림 6. 클러스터 시스템 테스트 구성도
Figure 6. Cluster System Test Diagram

하지만 웹 페이지를 통해 자원을 접근 하는 경우, 특히 미디어를 브라우저에서의 직접적인 재생의 경우 현재의 HTML5의 특성상 클라우드 파일시스템에 자동으로 분배된 미디어를 연속적으로 개별 클러스터에 접근하여 재생할 수 없다. 그러므로 MPEG-DASH 또는 HLS를 이용하여 재생목록 파일은 <그림 6>의 웹서버에 저장하고 분할된 미디어 파일들은 각 클러스터에 분산 저장한다. 분산 저장된 분할 미디어 파일들은 각 클러스터에서 HTTP 서비스를 제공하도록 설정한다.

본 논문에서 테스트를 위해 사용된 웹서버의 사양은 <표 1>과 같이 우분투 서버 14.04 LTS 버전의 OS로 사용하며, 웹서버의 데몬은 NginX 1.9.12 버전을 사용하였고 WAS는 PHP FPM 5.5.9 버전을 사용하였다. 또한 각 클러스터는 <표 2>와 같이 데스크탑 버전인 우분투 14.04 LTS 버전의 OS를 사용하고, 웹서버 데몬은 NginX 1.9.12 버전을 사용하였다. WAS는 클러스터에는 필요하지 않아 사용하지 않았다.

표 1. 웹서버 사양
Table 1. Web Server Specification

구분	내용
CPU	Intel Core i3-3220 @ 3.30GHz
Memory	4GB
HDD	500GB IDE
NIC	100/1000MBps Ethernet adapter
OS	Ubuntu Server 14.04 LTS

표 2. 클러스터 사양
Table 2. Cluster Specification

구분	내용
CPU	Intel Core2 Quad Q9330 @ 2.50GHz
Memory	2GB
HDD	500GB IDE
NIC	100/1000MBps Ethernet adapter
OS	Ubuntu Desktop 14.04 LTS

3.2 미디어 분할 파일 분산시스템 메타데이터를 적용한 클러스터 시스템의 전송률 측정 방법

본 논문에서의 전송률 측정을 위해 해당하는 미디어는 영상시간 총 11분 56초짜리인 일반 홍보영상을 이용하였다. 해당 영상의 원본은 윈도우미디어 비디오 영상을 영상크기 가로 960 픽셀, 세로 540 픽셀로 하여 비디오는 비트레이트 1Mbps인 H.264 비디오 코덱으로 인코딩하였고, 오디오는 비트레이트 64kbps인 AAC 오디오 코덱을 이용해 인코딩하여 MP4 컨테이너[11]를 이용한 MPEG-4[12] 미디어 파일을 생성하였다.

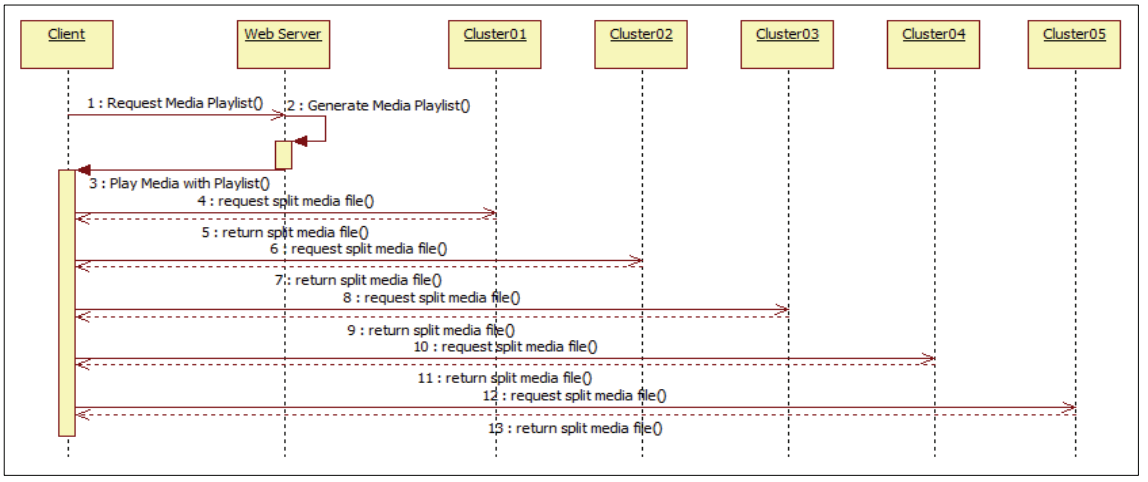


그림 7. 클러스터 시스템 테스트 순차 다이어그램
Figure 7. Cluster System Test Sequence Diagram

이를 각 클러스터에 분배하기 위해 스트림 세그멘터[13]를 이용하여 3초 단위로 하여 해당 미디어 파일을 분할하였다. 이렇게 분할된 파일은 총 177개이며 이를 <그림 4>의 분할 파일 분배 알고리즘을 이용하고 클러스터 중복 저장 개수인 nScale을 3으로 하여 각각의 클러스터에 분할 파일을 분배하였다. 이는 분할 미디어 파일 1개를 3개의 클러스터에 중복으로 분배하는 것을 의미한다.

<그림 7>은 클러스터 시스템 테스트 순차 다이어그램으로 클라이언트가 각 클러스터에 있는 분할된 미디어에 접근하여 해당 파일들을 다운로드하는데 사용되는 네트워크 전송률을 측정하기 위한 테스트를 진행하는 프로세스를 나타낸 다이어그램이다. 우선 클라이언트에서 <그림 7>의 1과 같이 재생할 미디어의 파일명 또는 시리얼 번호를 가지고 웹서버에 해당 미디어의 재생 목록 인덱스 파일을 요청한다. 웹서버에서는 <그림 7>의 2와 같이 요청된 미디어 파일의 기준 재생 목록 인덱스 파일(.m3u8 파일 또는 .mpd 파일)을 가지고 <그림 5> 분배 파일 목록 생성 알고리즘 이용해 개정된 미디어 재생 목록 인덱스 파일을 생성한 후 <그림 7>의 3과 같이 클라이언트에 전달하여 준다. 클라이언트는 전달 받은 미디어 재생 목록 인덱스 파일을 파싱하여 각각의 분할 파일의 URL 경로를 취득하고 각각의 분할 파일 URL 경로를 접속하여 해당 미디어 파일을 다운로드한다. 이렇게 다운로드하는 프로세스를 <그림 7>의 4에서부터 13까지 나타내었으며 이는 순차적으로 이루어지는 것이 아닌 웹서버에서 개정된 미디어 재생 목록 인덱스 파일내의 분할 파일 목록의 순서대로 수행되어진다.

위의 클라이언트에서 행해지는 모든 프로세스는 Apache JMeter[14]를 이용하여 테스트를 진행하였다. 테스트 방법은 Apache JMeter에서 각각 1, 10, 20, 30, 40, 50, 100개의 동시접속을 실행하여 웹서버에서 개정된 미디어 재생 목록 인덱스 파일을 받아와 각각의 클러스터에 접속하여 파일을 다운로드 하였다. 이 때 각각의 클러스터에 nmon (Nigel's performance Monitor)[15]이라는 네트워크 성능 모니터링 툴을 이용하여 초당 네트워크 전송률을 수집하였다.

표 3. 동시접속자 수 당 각 클러스터의 네트워크 전송률
Table 3. The Bit rate of each Concurrent Users per Cluster

(단위 : kbit/s)

Connection / Cluster	1	10	20	30	40	50	100
Cluster 1	2082.98	2528.22	2528.63	2537.87	2542.45	2493.81	2622.28
Cluster 2	2411.07	2813.18	2813.10	2844.39	2844.78	2832.91	2844.06
Cluster 3	2045.24	2372.47	2372.43	2372.45	2372.51	2383.03	2275.63
Cluster 4	2058.21	2363.28	2363.25	2363.72	2370.24	2369.21	2429.43
Cluster 5	1659.82	1921.37	1921.40	1928.58	1932.21	1934.27	1888.96

3.3 시스템 전송률 결과 및 결과 분석

위 3.2절에서의 방법으로 각각의 클러스터에서의 미디어 네트워크 전송률을 테스트한 결과는 <표 3> 동시접속자 수 당 각 클러스터의 네트워크 전송률과 같다.

<표 3>에서 Connection은 동시 접속자 수를 의미한다. 각 Cluster는 각각의 동시접속자 수 상황에서 전체 미디어 다운로드 전송량을 전체 접속 시간으로 나눈 전송률이다. 동시접속자 수가 100명인 경우 각각의 접속자는 동시가 아닌 시간차를 두고 분할된 미디어 파일을 받아 가기 때문에 1명이 접속한 시간 보다는 많아지게 된다. 하지만 미디어의 전체 상영 시간인 11분 56초에 92.7MByte인 영상의 경우 산술적으로는 1초를 재생하려면 0.13MByte (1.04Mbit)가 필요하다. 그러기 때문에 위의 표 3에서와 같이 초당 최저 전송률이 클러스터 5의 1 접속수인 경우 1.6Mbit (1659.82kbps)이고 나머지는 그 이상의 전송률을 가지므로 영상이 제대로 재생되면서 차례로 다음 영상의 다운로드가 가능하다.

클러스터 5의 경우에는 다른 클러스터에 비해 전송률이 낮은 이유는 미디어가 분할되고 분할된 미디어들이 초기화된 클러스터 환경에 분배 될 때에는 분할된 미디어의 개수에 의존하여 마지막 클러스터 군에 배포가 되지 않을 확률이 많기 때문에 저장된 분할 미디어 개수가 타 클러스터 보다 적으므로 전송률이 작게 나오는 것이다.

위에서 계산한바와 같이 표 3의 전체 클러스터 중에서 초당 최저 전송률이 1.6Mbit 이고 영상 재생 최저 조건인 초당 1.04Mbit를 상위하므로 안정적으로 전송되어지고 있다는 것을 증명한다. 그리고 동시 접속인원이 늘어나도 각 클러스터의 네트워크 전송률은 일정하게 증가한다.

<그림 8>은 위의 <표 3>을 접속자수 대비 전송률 그래프를 도시한 것이다. <그림 8>에서 보는 바와 같이 약간의 차이는 있으나 접속자 수가 증가함에도 클러스터별 네트워크 전송률은 일정하게 유지 또는 소폭 증가되고 있음을 보여준다.

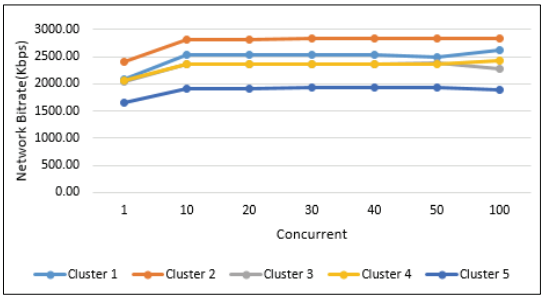


그림 8. 클러스터별 네트워크 전송률
Figure 8. Cluster Network Transfer Rate Chart

4. 결 론

본 논문에서는 기존에 연구되었던 미디어 분할 파일을 위한 분산 파일 모델링 시스템의 성능 테스트를 위해 클러스터 미디어 배포 프로세스 알고리즘을 이용하여 미디어를 분산하고 클러스터 내 분배 파일 목록 파일 생성을 하여 각 클러스터에서 분할 미디어를 다운로드 하여 재생하였다. 성능 테스트를 위해 웹서버와 클러스터로 구성된 서버군과 접속 테스트를 위한 클라이언트로 테스트한 결과 고화질의 미디어를 전송할 때에 웹서버에만 네트워크 트래픽이 집중되지 않고 각각의 클러스터에 분산되는 것을 알 수 있었다. 또한 위의 실험에서와 같이 각 클러스터의 CPU사양과 기타 다른 하드웨어 사양이 낮더라도 안정적인 네트워크 전송률을 나타내는 것을 알 수 있었다. 이에 미디어 분할 파일을 위한 분산 파일 모델링 시스템이 안정적임을 입증하였다.

References

[1] M. K. Jung, and S. H. Cho, and J. W. Kim, *Metadata modeling of cloud file system in accordance with the attributes of the VOD media segment files*, Proceedings of 16th Autumn Conference of The Korean Knowledge Information Society, pp. 10-13, 2014.

[2] S. Ghemawat, H. Gobioff, and S. T. Leung, *The google file system.* In ACM SIGOPS Operating Systems Review. Vol. 37, No. 5, pp. 29-43, ACM, Oct. 2003.

[3] M. K. Jung, & J. W. Kim, *The VOD service is based on cloud file system which is used to fixed-node server of private dedicated network.* Journal of Knowledge Information Technology and Systems, Vol 9, No. 5, pp. 607-615, 2014.

[4] T. Stockhammer, *Dynamic adaptive streaming over HTTP--: standards and design principles.* Proceedings of the second annual ACM conference on Multimedia systems. pp. 133-144, 2011.

[5] Dynamic Adaptive Streaming over HTTP, https://en.wikipedia.org/wiki/Dynamic_Adaptive_Streaming_over_HTTP, Jun. 2016.

[6] R. Pantos, W. May, *HTTP live streaming draft-pantos-http-live-streaming-05.* Published by the Internet Engineering Task Force, 24, 2011.

[7] HTTP live streaming overview, https://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/StreamingMediaGuide/Introduction/Introduction.html, 2016.

[8] HTTP live streaming draft-pantos-http-live-streaming-19 https://tools.ietf.org/html/draft-pantos-http-live-streaming-19. Apr. 2016

[9] M. K. Jung, and J. W. Kim, *A design and implementation of high quality live stream backup transport system using shortest path in private network.* Journal of Knowledge Information Technology and Systems, Vol 9, No. 4, pp. 409-416, 2014.

[10] M. K. Jeong, and J. W. Kim, *Generation method of VOD media segment play list using an optimum relay server search algorithm,* Proceedings of 17th Spring Conference of The Korean Knowledge Information Society, pp. 96-99.

[11] MPEG-4, Advanced video coding (Part 10)(H.264), http://www.digitalpreservation.gov/formats/fdd/fdd000081.shtml, Oct. 2014.

[12] ISO/IEC 14496-2:2004 Information technology -- Coding of audio-visual objects -- Part 2:

Visual, http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=39259, Jun. 2004.
[13] HTTP streaming architecture, https://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/StreamingMediaGuide/HTTPStreamingArchitecture/HTTPStreamingArchitecture.html, 2016.
[14] Apache JMeter, http://jmeter.apache.org/, 2016.
[15] nmon for Linux, http://nmon.sourceforge.net/pmwiki.php, May 2016.

미디어 분할 파일을 위한 분산 파일 시스템의 성능 테스트 방법 수행

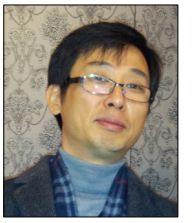
정명기[1], 김재웅[2]
[1]*(주)키스코이앤엠 소프트웨어개발연구소*
[2]*공주대학교 컴퓨터소프트웨어공학 전공*

요 약

현존하는 클라우드 파일 시스템은 지정된 크기로 파일을 분할하여 단일 미디어 파일을 각 클러스터에 분산 저장하고 있다. 그런데 모바일 환경에서의 스트리밍 방식으로는 MPEG-DASH 또는 HTTP Live Streaming 방식을 많이 사용하고 있다. 이러한 MPEG-DASH 또는 HTTP Live Streaming 방식으로 인코딩된 미디어는 분할된 미디어 파일들로 구성되어 있다. 이렇게 분할되어진 미디어 파일을 현존하는 클라우드 파일 시스템을 이용하여 클러스터에 분배할 경우 이 후 클라이언트들이 해당 미디어의 재생을 요청하는 경우 클러스터들의 게이트웨이 역할을 하는 서버의 네트워크 트래픽이 단일 미디어 파일을 재생할 때와 똑같이 집중되게 된다. 이에 본 논문에서는 이전 연구에서 제안되어진 분할 미디어 파일 분산 파일 모델링 시스템을 구현하고 해당 분산 파일 모델링 시스템의 안정성을 확인하기 위해 테스트 기기 구성 및 테스트 프로세스 수립하여 클러스터의 네트워크 트래픽을 측정하고, 분할 미디어 파일 분산 파일 모델링 시스템의 안정성을 입증하였다.

감사의 글

이 논문은 2014년 공주대학교 학술연구지원사업의 연구지원에 의하여 연구되었음.



Myeong-Ki Jeong received the bachelor's degree in the Department of Electric Engineering Education from the Chungnam National University in 1994. He received the M.S. degree in the Department of Computer Multimedia Engineering from Kongju National University in 2004. He received the Ph.D. degree in the Department of Computer Engineering from Kongju National University in 2015. He has been a Chief Technology Officer in the Keysco E&M since 2015. His current research Interests include software engineering, web engineering, Social Media Broadcasting Platform. He is a regular member of the KKITS.
E-mail address: bright.jung@gmail.com

Jae-Woong Kim received the bachelor's degree and the MS degree in the Department of Computer Engineering from the Jungang University in 1983 and 1988, respectively. He received the Ph.D. degree in the Department of Computer Engineering from Daejun University in 2000. He has been a professor in the Department of Computer Software Engineering at Kongju National University since 1992. His current research interests include software engineering. He is a life member of the KKITS.
E-mail address: yjkim@kongju.ac.kr