하둡 및 스파크를 이용한 초고품질 영상 실시간 처리 시스템 개발

허진강, *김용환
전자부품연구원
jingang4394@keti.re.kr, *yonghwan@keti.re.kr

Development of Real-time High-Fidelity Video Processing System using Hadoop and Spark

Jingang Huh, *Yonghwan Kim
Korea Electronics Technology Institute (KETI)

요 약

최근 4K/8K 급 초고품질 콘텐츠의 서비스에 관심이 집중되는 만큼 스트리밍 서비스에 대한 연구도 활발히 이루어지고 있다. 하지만 단일 PC 성능의 한계로 인해 SW 기반 영상 처리에 어려움을 겪고 있다. 본 논문에서는 분산 처리를 통해 실시간 영상 처리가 가능하도록 시스템을 제안한다. 제안한 시스템은 영상 패킷 분석 및 분할, 분산 트랜스코딩, 패킷 통합 단계로 이루어지며 Hadoop 과 Spark 를 이용하여 실시간 분산 처리를 지원한다. 실험 결과는 초고품질 입력 영상(3840x2160@60Hz, YCbCr 4:2:2, 10-bit)에 대해 평균 74.47fps 의 트랜스코딩 속도를 보인다.

1. 서론

고성능 디바이스의 대중화로 인해 최근 4K/8K UHD(Ultra High Definition) 콘텐츠 관련 서비스에 대한 관심도가 높아지고 있다. 그에 따라 UHD 영상 스트리밍에 대한 연구도 활발하게 이루어지는 가운데 단일 PC 성능의 한계로 인해 SW(Software)기반 초대용량 데이터 처리에 어려움을 겪고 있다. 현재 단일 PC 에서는 3840x2160@60Hz, YCbCr 4:2:0 8-bit 영상(약 248Mbytes)에 대해서는 SW 기반 영상 처리가 가능하다. 하지만 3840x2160@60Hz, YCbCr 4:2:2 10-bit 영상(약 415Mbytes)에 대해서는 데이터 크기가 약 2 배정도 차이 나기 때문에 SW 기반 영상 처리가 어려운 실정이다. 따라서 초고품질 콘텐츠에 대해 MPEG-4 AVC|H.264 대비 약 2 배이상의 압축률을 보이는 HEVC(High Efficiency Video Coding)기반 초고효율 코덱 기술이 필수적이다. 하지만 HEVC 압축 기법 또한 약 3 배 이상의 고복잡도를 보이기 때문에 여전히 단일 PC 로는 영상 처리에 한계를 나타낸다.

이러한 문제점을 해결하고자 최근 국내외에서는 병렬/분산 처리를 이용한 비디오 코딩 기법에 대해 연구되었다[1]. 그러나 UHD 콘텐츠를 지원하는 시스템은 여전히 미흡하며 초고품질 영상의 특성으로 인해 기존 방식으로는 SW 기반 실시간 영상 처리 시스템을 제공하기 힘들다. 따라서 본 논문에서는 단일 PC 로 인한 초고품질 영상 처리의 한계점을 극복하고 SW 처리가 가능한 하둡 및 스파크를 기반 실시간 영상 처리 시스템을 제안한다.

본 논문의 구성은 다음과 같다. 2 장에서는 본 논문에서 제안하는 전체 시스템에 대해 설명하고, 3 장에서는 제안한 방법을 이용한 구현 및 사례를 설명한다. 그리고 마지막으로 4 장에서는 결론을 맺는다.

2. 분산 트랜스코딩 시스템

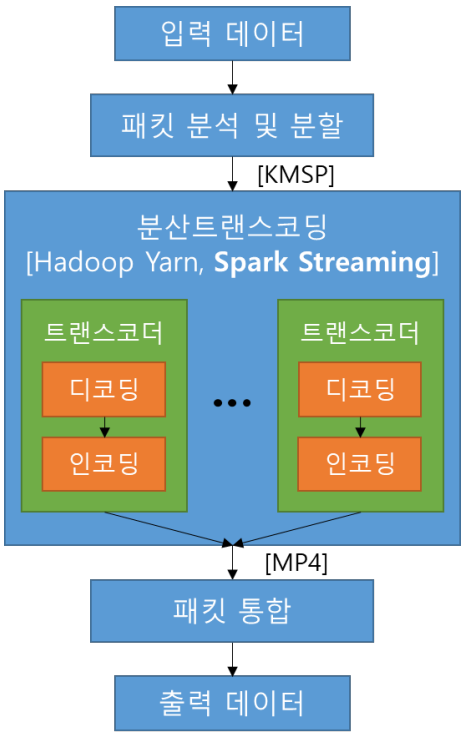

그림 1. 분산 트랜스코딩 전체 시스템 구조

그림 1 은 본 논문에서 제안하는 분산 트랜스코딩 시스템의 전체 구조를 나타낸다. 크게 3 파트로 나뉘며 패킷 분석 및 분할, 분산 트랜스코딩, 패킷 통합 단계로 이루어진다.

먼저 '패킷 분석 및 분할' 단계는 입력 영상이 업로드 됐을 때 데이터를 분석하고 분할하는 단계이다. 입력 영상으로는 AVI, MP4, MXF, RAW 등의 파일 포맷이 사용될 수 있으며 FFMPEG[2]을 통해 분석된다. 이때 입력 영상이 RAW 파일일 경우엔 다른 파일 포맷에 비해 비교적 크기가 크기때문에 MPEG-4 AVC|H.264 를 통해 1 차 압축을 하고 분석이 진행된다. 분석된 입력 영상은 분산 처리를 위해 KMSP(Keti Multi-Stream Protocol format, 그림 2) 형태의 패킷 단위로 분할된다. 분할 기법은 시분할과 공간분할 두 가지 기법 중 선택할 수 있으며 기본값으로 시분할 기법이 적용된다. 시분할 기법의 경우 프레임 단위로 시간에 따라 분할되는 것을 의미하며 기본값으로 300 프레임(약 5 초)이 설정되어있다. 또한 공간분할 기법의 경우 한 프레임 내에서 공간적으로 분할되는 것을 의미한다. 고복잡도를 가진 영상에 적용할 경우 효과적으로 사용된다.

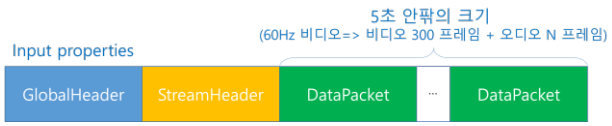

그림 2. KMSP 내부 구조

두 번째는 '분산 트랜스코딩' 단계이다. 분할된 KMSP 패킷은 Spark Streaming[3] 방식으로 분산 트랜스코더에 전송된다. Spark Streaming 방식은 영상을 모두 분할 한 후 일괄적으로 처리하는 배치 프로세싱과 달리 패킷이 만들어질 때마다 처리하는 인메모리 프로세싱이다. 따라서 배치 프로세싱 방식인 Hadoop[4]과 달리 고효율의 성능을 보인다. 또한 Map/Reduce 연산도 메모리에서 이루어지기 때문에 높은 처리속도를 보인다.

트랜스코딩된 각 패킷은 MP4 형태로 '패킷 통합' 단계로 전송된다. '패킷 통합' 단계에서는 전송된 각 MP4 패킷을 하나의 파일로 통합하는 단계다. 모든 분산 처리가 완료되면 최종적으로 스트리밍이 가능한 형태로 출력된다.

3. 구현 및 적용사례

제안된 방법을 기반으로 Hadoop 및 Spark 를 이용한 실시간 영상 처리 시스템을 개발하였다. 클러스터 구성은 Master 1 대, Slave 7 대로 구성하였으며 Hadoop 패키지는 Windows OS 를 지원해주는 Hortonworkd[5]의 HDP(Hortonworks Data Platform) 2.4 를 이용하여 Windows 7/10 클라우드 환경을 구축했다. Hadoop 버전은 2.7.1, Spark 버전은 1.6.3 이다. 분산 트랜스코더의 디코더 기능은 FFMPEG 라이브러리를 이용하였으며 인코더는 KETI(Korea Electronics Technology Institute)에서 개발한 HEVC 부호화기를 이용하였다.

표 1. 실험 환경

Master node	CPU	RAM	OS
R7910 x 1EA	Xeon E5-2867W v4 @3.0GHz	64GB	Windows 7
Slave node	**CPU**	**RAM**	**OS**
R7910 x 3EA	Xeon E5-2867W v4 @3.0GHz	128GB	Windows 10
R7910 x 2EA	Xeon E5-2867W v4 @3.0GHz	64GB	Windows 10
R7910 x 2EA	Xeon E5-2867W v3 @3.1GHz	64GB	Windows 10

표 2. 실험 결과

Sequence (Hz, #Frame,Format)	입력 비트율 [Mbps]	출력 비트율 [Mbps]	출력 화질 [dB]	트랜스코딩 속도 [fps]
Arte_Faust (60Hz, 21386, ProRes)	1706	75.30	42.6	67.00
Arte_SymphonySong (60Hz, 18073, ProRes)	1653	68.27	44.9	67.99
BigBuckBunny (60Hz, 38072, ProRes)	1887	12.04	50.2	71.32
GagHighlight (60Hz, 10929, XAVC)	600	24.00	47.5	91.58
평균	1462	44.90	46.3	74.47

4. 결론

4K/8K 급 UHD 콘텐츠에 대한 관심이 높아지는 만큼 원활한 스트리밍 서비스도 함께 요구된다. 본 논문에서는 Hadoop 및 Spark 를 이용하여 초고품질 영상 실시간 처리 시스템을 개발하였다. SW 기반 초고품질 영상 실시간 처리의 한계를 극복하였으며 초고품질 UHD 콘텐츠에 대한 실시간 스트리밍 서비스를 가능하게 했다. 추후 연구로는 분산 트랜스코더 시스템의 전체 프로세싱 방식을 인메모리 방식으로 처리하여 4K 이상의 영상, 즉, 8K/16K 영상에 대해서도 실시간 영상 처리가 가능하도록 개발하고자 한다.

감사의 글

본 연구는 문화체육관광부 및 한국콘텐츠진흥원의 2017 년도 문화기술 연구개발 지원사업으로 수행되었음 (R2017030018)

참고문헌

[1] 문희철, 김용환, 김동혁. "클라우드 기반 UHD 영상 트랜스코딩 시스템." 한국방송미디어공학회 학술발표대회 논문집, (2014.11): 203-205.

[2] FFmpeg, http://ffmpeg.org

[3] Spark, http://spark.apache.org

[4] Hadoop, http://hadoop.apache.org

[5] Hortonworks, http://hortonworks.com