움직이는 물체의 시공간 분석을 위한 동영상 빅 데이터 처리 방안

정승원*, 김용성*, 정상원**, 김윤기*, 황인준*
*고려대학교 전기전자공학과
**고려대학교 영상정보처리협동과정
e-mail:{jsw161, kys1001, emperorous, vardin, ehwang04}@korea.ac.kr

Video Big Data Processing Scheme for Spatio-Temporal Analysis of Moving Objects

Seungwon Jung*, Yongsung Kim*,
Sangwon Jung**, Yoonki Kim*, Eenjun Hwang*
*School of Electrical Engineering, Korea University
**Program in Visual Information Processing, Korea University

요 약

최근 블랙박스 및 CCTV 같은 영상 촬영 장치가 보편화되면서, 방대한 양의 영상 데이터가 실시간으로 생성되고 있다. 만약 이 대용량 데이터 안의 차량 정보를 추출할 수 있다면 범죄 차량 추적, 교통 혼잡도 측정 등의 활용이 가능할 것이다. 이를 구현하기 위해서는 수많은 자동차에서 실시간으로 생성되는 영상 데이터를 처리할 수 있는 시스템이 필수적이나, 이러한 시스템을 찾기 힘든 것이 현실이다. 이를 위해 이 논문에서는 아파치 카프카, Hbase를 이용한 영상 빅데이터 처리 시스템을 제안한다. 아파치 카프카는 시스템 내에서 영상 손실이 없는 전송과 영상 처리 노드의 스케줄링을 수행하며, Hbase는 처리된 데이터를 테이블로 저장하고 사용자가 보낸 쿼리를 처리한다. 더불어, Hbase에 인덱스를 구성하여 빠른 쿼리 처리가 가능하도록 만든다. 실험 결과, 제안된 시스템은 인덱스가 없을 때보다 뛰어난 쿼리 처리 성능을 보이는 것을 확인할 수 있었다.

1. 서론

최근 영상을 녹화하고 저장할 수 있는 매체들이 빠르게 발달하면서 다양한 영상 촬영 장치들이 개발되고 활용되고 있다. 특히 차량의 블랙박스는 보급률이 매우 높으며 택시와 버스는 블랙박스 장착이 의무화되면서 차량 대부분에 장착되어 있다고 보아도 무방하다. 또한, CCTV는 교통관제, 감시 등 다양한 목적으로 설치되며 최근에는 설치되지 않은 곳을 찾기 힘들 정도가 되었다. 이런 장치들은 상당히 많은 양의 동영상 데이터를 생성해내며, 전국 단위로 볼 경우, 상당한 양의 데이터가 매일 생성된다는 것을 확인할 수 있다.

블랙박스 및 CCTV 영상은 사고 정보 분석, 범죄 감시 등과 같은 다양한 곳에서 활용할 수 있다. 특히 실시간으로 생성되는 대규모 영상 데이터를 분석해서 차량 정보(번호판, 위치, 시간) 등을 추출한다면 활용도가 매우 다양해질 수 있다. 우선 저장된 데이터를 통해 차량이 원하는 장소의 실시간 차량 밀도를 계산하고 차량에 전달해 줄 수 있다. 이를 이용하면, 차량이 많은 지역을 피하여 차량이 목적지에 빠르게 도달할 수 있도록 만들 수 있다. 더하여, 누적된 통계를 기반으로 도시 행정에 사용할 수 있으며, 도난 차량이나 범죄 차량 같은 특정 차량의 위치 및 경로를 추적하여 치안 유지에 도움을 줄 수 있다.

그러나 이를 위해 모든 자동차가 실시간으로 영상 데이터를 생성하여 시스템과 통신한다고 가정하면, 국내에 등록된 자동차 수[1]를 미루어 보아 국내에서만 최소 2천만 개 이상의 영상 데이터가 실시간으로 발생하고 이를 처리해야만 한다. 시스템에서 처리해야 할 데이터의 크기가 매우 방대하므로 시스템은 대용량 데이터를 수집하고 처리하여 저장할 수 있는 능력을 필수적으로 갖추어야 한다.

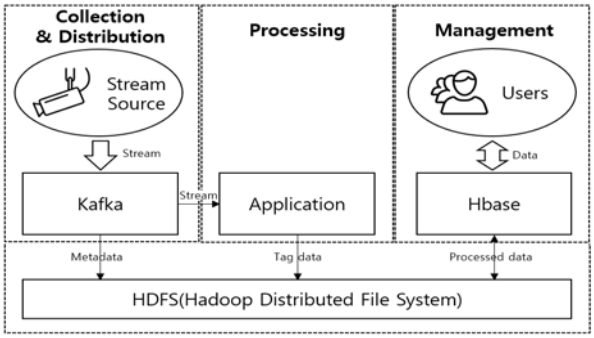
그러나 하둡에서 맵리듀스로 영상 처리를 하기 위한 인터페이스[2], CNN을 이용한 실시간 영상 처리 시스템[3] 등 대용량 영상 처리만을 위한 연구는 쉽게 찾을 수 있지만, 영상 데이터를 처리하여 추출된 정보를 저장하고 관리할 수 있는 통합 시스템은 찾기 힘든 것이 현실이다.

따라서 본 논문에서는 차량에서 발생하는 동영상 빅데이터를 효과적으로 처리, 저장, 검색할 수 있는 시스템을 제시한다. 우선 아파치 카프카(Apache Kafka)[4]를 활용하여 대규모 실시간 영상(블랙박스, CCTV) 스트림 데이터를 저장하고 전송한다. 그리고 애플리케이션 클러스터에서 영상 속 차량의 번호판 및 위치 정보 등을 추출하여 저장한다. 마지막으로 모든 데이터는 Hbase[5]에 저장되며, 인덱스(index)를 구성하여 저장된 데이터를 효과적으로 검색할 수 있도록 한다.

2. 시스템 구조

본 논문에서 제안하는 전체 시스템의 구조는 그림 1과 같다. 크게 수집(Collection) 및 분배(Distribution), 처리

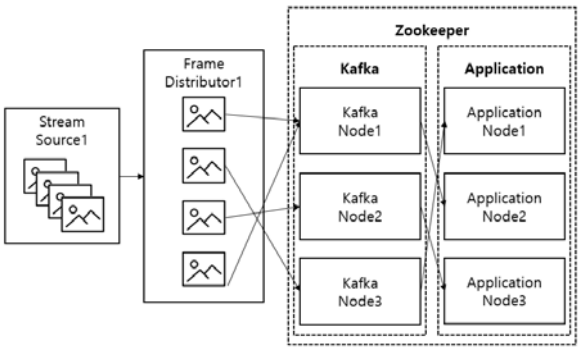
(Processing) 그리고 관리(Management)로 총 세 부분으로 나누어지며, 모두 하둡 분산 파일 시스템(Hadoop Distributed File System)을 기반으로 하여 작동한다.



(그림 1) 시스템 구조도

1) 대용량 영상 데이터의 분배 및 처리

블랙박스 및 CCTV에서 입력되는 영상 데이터는 스트림 형태로 입력되기 때문에 영상이 생성되는 양이 처리되는 양보다 많아질 경우 데이터 손실이 생길 수 있다. 이를 해결하기 위해 [6]을 이용하여 영상 데이터를 수집하고, 애플리케이션 클러스터에서 처리한다. 그림 2는 스트림 소스가 하나일 때를 가정하여 영상이 처리될 때까지의 과정을 그린 그림이다. 각 부분이 맡은 역할은 다음과 같다.



(그림 2) 실시간 처리 시스템 구조도

(1) 프레임 분배기 : 스트림 형태로 생성된 영상 데이터는 프레임 분배기로 전송된다. 프레임 분배기는 각 노드의 자원에 따라 할당된 스트림 소스에서 스트림 데이터를 받아 프레임 단위로 분할하고 다음 단계인 카프카 클러스터로 전송한다.

(2) 카프카 클러스터 : 카프카 클러스터는 주키퍼(Zookeeper)[7]에 의해 관리되는 카프카 노드들로 이루어져 있으며, 분배기에서 보내진 데이터를 받아 토픽(Topic)이라는 단위로 나누고 실제 영상 처리를 수행하는 노드에 분배하는 스케줄링을 담당한다. 또한, 수신한 데이터를 임시로 저장해놓기 때문에 데이터 손실을 막기 위한 버퍼 역할도 한다.

(3) 애플리케이션 클러스터 : 차량 정보를 뽑기 위한 실제 영상 처리를 수행하는 노드들의 클러스터로, 카프카 클러스터에서 보내는 데이터를 받아 처리한다. 카프카 클러스터와 마찬가지로 애플리케이션 클러스터도 주키퍼에 의해 관리되어, 두 클러스터 사이의 원활한 통신이 가능하다.

이 과정을 거치면 영상 내에 찍힌 차량 번호가 추출되고 영상이 찍힌 시각, GPS 정보 등 영상의 메타 데이터와 함께 하둡 분산 파일 시스템에 저장된다. Hbase 클러스터는 처리된 데이터를 모아 Hbase 테이블에 저장하고 관리하게 된다.

2) 처리된 데이터의 관리

앞에서 언급했듯이, 추출된 차량번호(tag)는 영상이 찍힌 시각(*time*), 위도(*latitude*), 경도(*longitude*) 데이터와 함께 하둡 분산 파일 시스템에 저장된다. 저장된 데이터가 일정량 이상 쌓일 때마다 Hbase 클러스터는 맵리듀스를 이용하여 데이터를 Hbase 테이블에 저장하는 작업을 수행한다. Hbase 테이블의 스키마는 그림 3과 같다.

	column family 1	
row key	latitude	longitude
11가1111_1488960532177	37.58268	127.02621
11가1111_1488960532183	37.58274	127.02672
11가1111_1488960532199	37.58307	127.02813
...	...	...

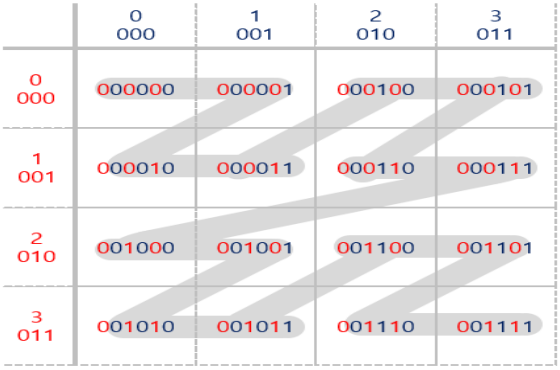
(그림 3) Hbase 테이블의 스키마

Row key는 차량번호와 영상이 찍힌 시간의 조합으로 이루어져 있어, 데이터가 존재하는 차량에 대해 새로운 데이터가 들어왔을 때 각 row에 업데이트를 하지 않고 새로운 row가 추가된다. row key 조합에 들어간 속성인 tag, *time*의 column은 따로 생성하지 않고 row key를 파싱(parsing)하여 값을 얻어오도록 설계하였다. 이러한 설계는 각 row에 업데이트를 하지 않기 때문에 할 수 있는데, 칼럼 지향 데이터베이스인 Hbase에서 row key가 각 cell마다 중복으로 저장된다는 점을 고려해보면 모두 저장하는 것보다 약 절반가량의 저장 공간을 아낄 수 있다는 장점이 있다.

Hbase 테이블은 기본적으로 row key에 대해 정렬되어 있기 때문에 특정 차량의 데이터는 모여 저장된다. 이 때문에 특정 차량의 GPS 데이터를 요구하는 쿼리(query)에 대해서는 쉽게 처리할 수 있다. 그러나 Hbase는 row key가 아닌 다른 속성들에 대해서는 인덱스를 구축해놓지 않기 때문에, 특정 차량의 데이터가 아닌, 특정 위치와 특정 시간을 조건으로 하는 쿼리를 처리할 때는 모든 데이터를 스캔해야 한다. 이러한 단점을 보완하기 위해 [8]처럼 인덱스 테이블을 구축하여 쿼리를 처리하는 방식을 사용한다. 인덱스 테이블의 row key는 식 (1)과 같이 여러 개의 속성이 조합되어 결정된다.

$$Rowkey = Z(latitude, longitude) + time + tag + latitude + longitude \quad (1)$$

식 (1)에서 $Z(latitude, longitude)$는 Z-ordering[9] 값을 뜻한다. Z-ordering 곡선은 공간 채움 곡선(space-filling curve) 중 하나로, 2차원인 GPS 데이터를 1차원으로 표현하기 위해 사용된다. 인덱스 테이블의 row key 구성으로 인해 인덱스 테이블은 일차적으로 Z-ordering 값에 의해 정렬된다. 이러한 점 때문에 조건에 포함되는 장소의 Z-order 값을 구한다면, 쿼리에 대해 해당하는 데이터의 위치를 쉽게 찾을 수 있다. 다만, Z-ordering 값으로 변환된 쿼리 조건 범위는 실제 조건 범위에 해당하지 않는 경우도 포함할 수 있어, row key에 포함된 *latitude*와 *longitude* 값을 통해 쿼리 조건에 부합하는 결과인지 확인하는 절차를 거쳐야 한다.



(그림 4) Z-ordering 곡선 예시

3. 실험 및 결과

구축된 시스템을 통해 차량 정보를 추출해보고, 저장된 데이터를 인덱스 테이블을 이용하여 가져오는 실험을 수행하였다.

1) 데이터 추출

표 1과 그림 5는 실제 블랙박스 영상을 넣었을 때 추출되는 데이터와 메타 데이터의 예시이며, 이 정보는 Hbase 테이블로 저장되어 관리된다.

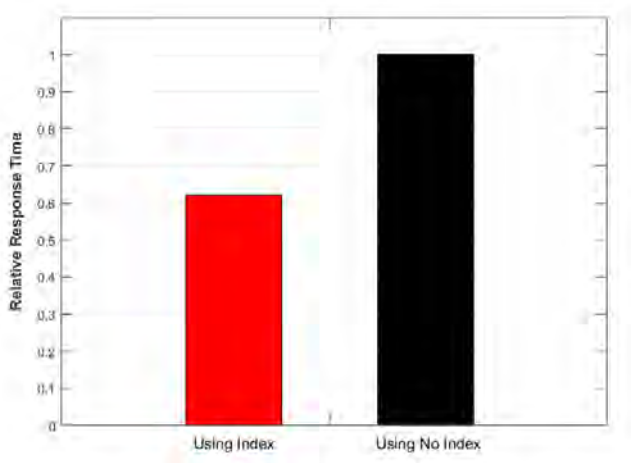
<표 1> 추출 데이터의 예시

tag	recordTime	latitude	longitude
40머51xx	1486001630884	37.63676	127.06777
80우26xx	1486001632901	37.63659	127.06780
80우26xx	1486001668445	37.63650	127.06791
80우26xx	1486001670519	37.63589	127.06816
80우26xx	1486001672387	37.63567	127.06785
80우26xx	1486001674253	37.63545	127.06741

(그림 5) 블랙박스 영상과 추출 데이터의 예시

2) 인덱스 테이블을 이용한 쿼리 처리

임의의 공간 쿼리를 여러 개 생성하고 인덱스 테이블의 유무에 따른 쿼리 처리 시간을 비교하였다.



(그림 6) 인덱스 유무에 따른 쿼리 처리 시간

<표 2> 인덱스 구성 시 가장 빨랐던 경우와 가장 느렸던 경우의 쿼리 처리 시간

	Best Case	Worst Case
Relative Response Time	0.00552	0.987

그림 6은 인덱스 테이블이 없을 때 쿼리 처리 시간을 1이라 가정했을 때, 인덱스 테이블 구성 시의 쿼리 처리 시간을 나타낸 후 평균을 낸 결과다. 인덱스 구성 시 평균 쿼리 처리 시간은 0.621로, 인덱스가 없을 때의 쿼리 처리 속도보다 평균적으로 1.6배 이상 빠르게 처리할 수 있었다.

다만, 이는 평균일 뿐, 쿼리 조건 범위와 부합하는 데이터의 유무에 따라 인덱스로 얻을 수 있는 이득이 달랐다.

표 2는 인덱스 테이블을 이용했을 때 인덱스 테이블이 없을 때와 비교하여 쿼리 처리 시간이 가장 빨랐던 경우와 가장 느렸던 경우의 시간을 나타낸 것이다. 가장 큰 이득을 얻는 경우는 조건에 부합하는 데이터가 없을 경우로, 인덱스 테이블을 이용했을 때 100배 이상 빠른 속도를 보였다. 반대로, 이득이 거의 없는 경우는 실제 쿼리의 조건 범위보다 Z-ordering에 의해 변환된 조건 범위가 매우 크게 변환된 경우 혹은 데이터 대부분이 포함될 만큼 넓은 범위의 조건 범위일 경우였다.

4. 결론

본 논문에서는 동영상 빅데이터에서 데이터 추출을 하기 위해 카프카를 활용하여 영상 스트림 데이터를 저장 및 분배하고, 이를 애플리케이션 노드에서 처리한 후 Hbase에서 데이터를 관리하는 시스템을 제시하였다. 이를 통해 실시간 영상 데이터를 처리하고 데이터를 저장하는데 있어, 영상의 손실 없이 영상을 전송 가능하다는 점, 인덱스 테이블을 이용하여 더욱 빠른 공간 쿼리를 수행할 수 있다는 점 등의 성과를 거둘 수 있었다.

그러나 클러스터의 노드 수와 데이터 스트림 소스의 수가 늘었을 때 시스템이 실시간 처리에 어떤 성능을 보일지는 좀 더 많은 실험을 수행해야 한다는 점, 실제 시스템이 활용성을 갖기 위해서는 다양한 쿼리를 처리할 수 있는 인덱스 방식과 효율적으로 결과를 보여줄 수 있는 인터페이스가 필요하다는 점은 더욱 더 나은 시스템을 위해 해결해야 할 요소들일 것이다. 따라서 추후 이를 위한 연구를 진행할 계획이다.

5. Acknowledgment

이 논문은 2016년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원(NO. R0190-15-2012, 빅데이터 처리 고도화 핵심기술개발사업 총괄 및 고성능 컴퓨팅 기술을 활용한 성능 가속화기술 개발)과 2016년도 정부(교육부)의 재원으로 한국연구재단의 지원(NRF-2016R1D1A1A09919590)을 받아 수행된 연구임.

참고문헌

[1] 국토교통부, “자동차등록현황,” 2016

[2] C. Sweeney, L. Liu, S. Arietta, and J. Lawrence, "HIPI: a Hadoop image processing interface for image-based mapreduce tasks," Chris. University of Virginia, 2011.

[3] J. Muller, J. Muller, and R. Tetzlaff, "A new high-speed real-time video processing platform," in Cellular Nanoscale Networks and Their Applications (CNNA), 2014 14th International Workshop on, 2014, pp. 1-2: IEEE.

[4] Apache Kafka, 2017 [online] Available: https://kafka.apache.org/

[5] Apache Hbase, 2017 [online] Available: https://hbase.apache.org/

[6] Y. Kim and C. Jeong, "LARGE SCALE IMAGE PROCESSING IN REAL-TIME ENVIRONMENTS WITH KAFKA,", 2017

[7] Apache Zookeeper, 2017 [online] Available: https://zookeeper.apache.org

[8] Y. Zou, J. Liu, S. Wang, L. Zha, and Z. Xu, "CCIndex: A complemental clustering index on distributed ordered tables for multi-dimensional range queries," in IFIP International Conference on Network and Parallel Computing, 2010, pp. 247-261: Springer.

[9] G. M. Morton, A computer oriented geodetic data base and a new technique in file sequencing. International Business Machines Company New York, 1966.