반도체디스플레이기술학회지 제16권 제1호(2017년 3월)
Journal of the Semiconductor & Display Technology, Vol. 16, No. 1. March 2017.

병렬처리 기반 정지영상 인식자 생성

고미은[*]·박제호[**†]·박용범[*]·서원택[***]
[*]단국대 컴퓨터과학과, [**†]단국대 소프트웨어학과, [***]R&D Center of DRTECH Corp.

Parallel Processing based Image Identifier Generation

Mieun Ko[*], Je-Ho Park[†], Young B. Park[*] and Wontaek Seo[***]**
[*]Dankook Univ., Dept. of Computer Eng., [**†]Dankook Univ., Dept. of Software Sci.,
[***]R&D Center of DRTECH Corp.

ABSTRACT

Recent enhancement in the still image acquisition devices has been widely perpetrated into the daily life of the common people. Due to this trend, the voluminous still images, that are produced and shared in the personal or the massive storage, need to controlled with effective and efficient management. The human-devised or system-generated still image identifiers used for the identification of the images are at risk in the situation of unexpected changing or eliminating of the identifiers. In this paper, we propose a parallel processing based method for still image identifier generation by utilizing the still image internal features.

Key Words : Image, Identification, Parallel Processing

1. 서 론

최근 모바일 기기, 노트북, 고화질 카메라 등으로부터 수집되는 정지영상의 양이 급증하고 있다. 처리해야 할 정지영상이 기하급수적으로 증가함에 따라 생성된 정지영상을 클라우드 컴퓨팅 인프라로 효율적으로 검색하고 저장할 필요성 또한 높아지고 있다[1]. 특정 시스템에서 정지영상을 식별하기 위한 방식으로 자동으로 식별자를 생성하는 방법이나 사용자가 임의의 식별자를 부여하는 방법을 사용한다. 그러나 임의의 식별자의 부여는 다수의 시스템에서 생성된 정지영상을 하나의 시스템으로 통일하였을 때 동일한 식별자를 가지는 경우가 발생할 수 있다는 문제점이 존재한다. 이러한 정지영상 속성과 무관하게 부여되는 임의의 식별자 문제를 해결하기 위해 직선성분 계수기반 다중 인덱싱 구성 방법을 사용할 수 있다[2-3]. 본 논문은 직선성분 계수기반 다중 인덱싱 구성 방법을 Hadoop Map-Reduce 프로그래밍 모델에 적용하였다.

[†]E-mail: dk_jhpark@dankook.ac.kr

Hadoop은 대규모의 데이터를 분산 처리 및 저장하는 Java기반의 오픈 소스 프레임워크로, Map-Reduce 프로그래밍 모델을 이용하여 대용량 정지영상을 빠르게 처리할 수 있다[4-5]. Hadoop Image Processing Interface(HIPI)는 대용량의 영상을 처리할 수 있는 Hadoop Map-Reduce 라이브러리로, 영상 처리에 최적화된 라이브러리인 OpenCV는 Hadoop과 연동하여 데이터를 분석 및 활용할 수 있다[6-7].

본 논문에서는 직선성분 계수기반 다중 인덱싱 구성 방법에 Hadoop Map-Reduce 프로그래밍 모델을 적용하였고, 제안한 시스템의 성능과 단일 시스템을 비교 분석한 결과 구현한 모델의 수행시간이 단축되었음을 확인할 수 있었다. 2장에서는 제안 시스템의 구성 및 방법에 대해 서술하였고, 3장에서는 제안 시스템의 성능에 대해 분석 및 평가한다. 그리고 결론 및 향후 연구를 제시하였다.

2. 본 론

2.1 정지영상 인식자

정지영상을 식별하기 위해 프로토콜에 따라 식별자를

자동적으로 생성하는 방법이나 사용자가 임의 방식으로 부여하는 방법을 사용한다. 그러나 이러한 방법은 다수의 기기 또는 시스템에서 생성된 정지영상을 하나의 시스템에 통합할 경우 정지영상 속성과 무관하게 인덱스가 부여된다는 단점이 존재한다[6].정지영상 속성과 무관하게 부여되는 식별자 생성방법의 문제를 해결하기 위한 방법으로 직선성분 계수기반 다중 인덱싱 구성 방법이 있다. 직선성분 계수기반 다중 인덱싱 구성 방법은 정지영상의 선형요소 검출을 위해 여러 알고리즘을 적용하여 직선요소를 추출한 후, 설정된 8개의 가상직선과 검출된 직선요소와 교차점을 계수하는 일련의 과정을 거쳐 인덱스의 구성 값을 생성하는 방법이다[7].

2.2 병렬처리를 이용한 인식자 생성

Hadoop은 대용량 데이터를 분산 처리하고 운영 기반을 위한 프레임워크이다[4]. 대용량 데이터를 처리하기 위한 대표적인 프레임워크인 Hadoop은 Map-Reduce 프로그래밍 모델을 제공한다. Hadoop Map-Reduce는 하둡 분산 파일 시스템(HDFS)[8]를 기반으로 하고, Java기반의 API를 제공하여 클러스터 환경에서 병렬처리를 가능하게 한다[9]. 일반적으로 Hadoop Map-Reduce Job은 사용자 정의 함수인 Map과 Reduce로 구성된다. Map-Reduce의 입력은 키-값 쌍 (K_{input}, V_{input})의 집합이고, Map 함수는 이러한 각 쌍에 대해 호출되고 0개 이상의 중간 키-값 쌍 ($K_{intermediate}$, $V_{intermediate}$)을 생성한다. 이후 Hadoop Map-Reduce 프레임워크는 이러한 중간 키-값 쌍을 중간 키 $K_{intermediate}$로 그룹화하고 각 그룹에 대해 Reduce 함수를 호출한다. 마지막으로 Reduce 함수는 0개 이상의 합계된 결과인 (K_{output}, V_{output}) 생성한다[10].

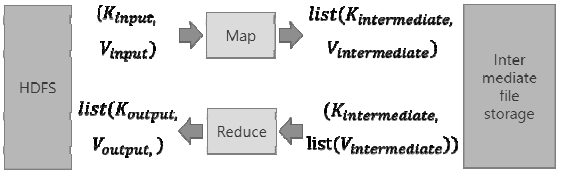

Fig. 1. Hadoop Map-Reduce Architecture [10-11] .

HIPI는 대용량 영상 처리를 하도록 설계된 Hadoop Map-Reduce 라이브러리로, 컴퓨터 비전 오픈소스 라이브러리인 OpenCV와의 통합을 제공한다. HIPI 프로그램에 대한 기본 입력은 HipiImageBundle(HIB)이고, HIB은 HDFS에서 단일 파일로 표시되는 이미지 모음이다. HIPI 프로그램의 첫 번째 처리 단계는 Cull 단계로 공간 해상도나 이미지 메타데이터와 같은 사용자 정의 조건을 기반으로 HIB에서 이미지를 필터링 한다. 이후 Culling된 이미지를 각 Map tasks에 할당하고 Hadoop Map-Reduce 프로그래밍을 진행한다[6-7].

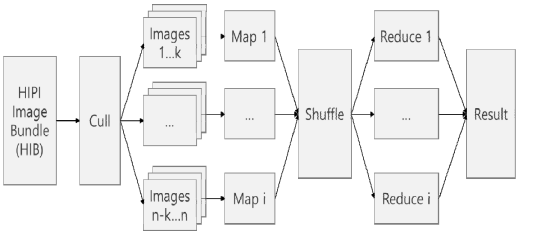

Fig. 2. Hadoop Image Processing Interface(HIPI) Architecture [8-9].

2.3 병렬처리 인식자 생성 절차

본 논문에서는 대용량 정지영상 데이터의 내부 속성을 고려한 식별자 생성을 Hadoop기반으로 처리하기 위해 HIPI에서 제공하는 HIB 이미지 모음을 이용하여 Hadoop Map-Reduce를 구현하였고, 직선성분 계수기반 다중 인덱싱 구성 방법을 구현을 위해 컴퓨터 비전 오픈소스 라이브러리인 OpenCV를 사용하였다. 기술한 시스템에서 정지영상을 생성하는 절차는 다음과 같다.

1. **입력 데이터 준비 단계:** 대용량 정지영상 식별자 생성을 위해 0.2MB 이하 크기의 이미지 데이터 10,000장을 테스트 데이터로 선정하였다. 선정된 테스트 데이터의 전체 데이터 사이즈는 1215012 KB로 하둡 분산 파일 시스템에서 관리하기에 충분한 크기이다.
2. **데이터 입력 단계:** 선정된 테스트 데이터를 하둡 분산 파일 시스템에 입력하기 위해 하둡 분산 파일 시스템 디렉토리를 생성하였다. 생성한 하둡 분산 파일 시스템 디렉토리에 테스트 데이터를 입력하기 위해 HIPI에서 제공하는 HibImport 도구를 이용하여 HIB 이미지 모음을 생성하였다.
3. **Map-Reduce 단계:** 입력된 HIB 이미지의 식별자 생성을 위해 Map-Reduce 프로그램을 구현하였다. 첫째, 직선성분 계수기반 다중 인덱싱 구성 방법을 적용하기 위해 필터링, 직선요소 검출, 직선요소 확장, 규정화 단계를 수행하였다. 필터링 단계는 정지영상을 직선요소 추출에 적합한 형태로 필터링하기 위해 영상을 그레이 스케일로 변환한다. 직선요소 검출 단계는 Canny 에지 검출 알고리즘을 적용하여 직선요소가 강화된 영상을 도출하고, 허프 변환 알고리즘을 사용하여 직선요소를 검출한다. 직선요소 확장 단계는 DDA 알고리즘을 적용하여 8개의 가상직선을 설정한다. 규

정화 단계는 설정한 가상직선과 검출된 직선요소와의 교차점을 계수하여 교차 횟수는 최종적으로 식별자의 부분값으로 사용된다.

둘째, 직선성분 계수기반 다중 인덱싱 구성 방법을 Hadoop Map-Reduce에 적용한다. 이때 8개의 가상직선 연산을 4개의 Map-Reduce쌍에 각각 2개씩 분산하여 총 4개의 Job에서 연산을 처리 한다. 그림 2는 입력 HIB 이미지 모음을 4개의 Map-Reduce쌍에 전달하고, 전달받은 각 Map-Reduce는 직선성분 계수기반 다중 인덱싱 구성 방법을 수행하고, 수행 시 가상직선 연산을 2개씩 수행하여 결과를 도출하는 처리구조를 보여준다.

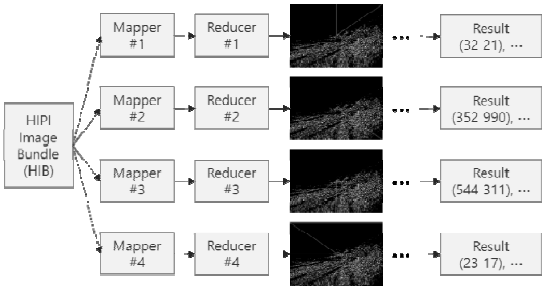

Fig. 3. Parallel Processing based Massive Still Image Identification Generation Architecture.

3. 실험 및 분석

본 논문에서 Hadoop 기반의 대용량 정지영상 식별자 생성 구현을 통해 대용량 정지영상 식별자를 효율적으로 처리하는 방법을 설명하였다. 제안한 Hadoop 기반의 대용량 정지영상 식별자 생성 구현의 데이터처리 성능을 실험하기 위해 1개의 마스터 노드와 4개의 슬레이브 노드로 구성하였다. 마스터 노드에서는 NameNode, Secondarynamenode, ResourceManager, JobHistoryServer가 실행되게 하였고, 나머지 4개의 슬레이브 노드는 DataNode와 NodeManager가 실행되도록 하였다.

그리고 제안한 방법의 데이터처리 성능을 비교하기 위해 단일 시스템에서 구동되는 정지영상 식별자 생성 프로그램을 구현하였고, 실험은 마스터 노드에서 진행하였다. 해당 프로그램은 Java를 기반으로 하고 OpenCV 라이브러리를 이용하였다.

Table 1은 각 시스템 별 실험에 이용한 도구와 도구 별 버전을 보여주고, Table 2는 Hadoop 클러스터 노드들의 물리적 사양을 나타낸다.

Table 1. Experimental Environment

시스템 구분	프로그램 명	버전
Hadoop 기반 시스템	Hadoop	2.6.0
	HIPI	2.0
	OpenCV	2.4.11
	Java	JDK-1.7
	OS	Ubuntu 14.04
단일 시스템	OpenCV	2.4.11
	Java	JDK-1.7
	OS	Ubuntu 14.04

Table 2. Hadoop Cluster Node Specification

노드 명	사양	사양 내용
Hadoop-Master	CPU	Intel(R) Core i5-3470 CPU@3.20GHz, 4 Cores
	Memory	8062724 KB
Hadoop-Slave1	CPU	Intel(R) Core i5-3470 CPU@3.20GHz, 4 Cores
	Memory	3920764 KB
Hadoop-Slave2	CPU	Intel(R) Core 2 Duo CPU E7500@2.93GHz, 2 Cores
	Memory	3920764 KB
Hadoop-Slave3	CPU	Intel(R) Core2 Quad CPU Q8400@2.66GHz, 4 Cores
	Memory	3920764 KB
Hadoop-Slave4	CPU	Intel(R) Core i3-3220 CPU@3.30GHz, 2 Cores
	Memory	3920764 KB

데이터처리 성능을 비교분석하기 위해 Hadoop기반 시스템은 4개의 Job 소요시간의 평균을 총 소요시간으로 도출하였고, 단일 시스템은 프로그램의 시작시점과 종료시점간의 차로 총 소요시간을 도출하였다. 도출된 결과는 Table 3과 같다.

Table 3. Time Ananlysis

시스템 구분	시간 구분	소요 시간
Hadoop 기반 시스템	Job1	7min 36sec 885ms
	Job2	11min 36sec 4ms
	Job3	7min 54sec 104ms
	Job4	13min 53sec 884ms
	평균 시간	10min 15sec 419ms
단일 시스템	총 소요시간	2h 10min 51sec 651ms

Hadoop기반 시스템의 총 소요시간은 10분 15초 419밀리세컨드이고 단일 시스템의 총 소요시간은 2시간 10분 51초 651밀리세컨드로 Hadoop기반 시스템으로 실험 데이터

를 분산 처리하는 것이 훨씬 효율적임을 알 수 있다.

Hadoop기반 시스템의 효율성을 구체적으로 검증하기 위해 Hadoop기반 시스템 내부구조를 수행 단계별로 나누어 분석하였다. Hadoop 기반 시스템을 수행 단계별로 나누어 검증하는 이유는 시스템 수행 과정에서 발생하는 네트워크 I/O로 인한 부하를 각 단계별로 분석하기 위함이다.

수행 단계는 Job이 시작하는 JobSubmitted, Map이 시작되는 MapAttemptStarted, Map이 종료되는 MapAttemptFinished, Reduce가 시작되는 ReduceAttemptStarted, Reduce가 종료되는 ReduceAttemptFinished, Job이 종료되는 Job Finished 단계로 구성된다. 단계간 소요시간은 T1~T5로 표현된다. T1과 T4는 커뮤니케이션 시간을 나타내고, T2는 Map 소요시간, T3는 Reduce 소요시간을 나타낸다. 각 단계 구체적인 구조는 Fig. 3에서 볼 수 있다.

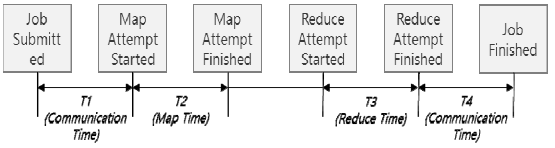

Fig. 4. Time Analysis Model.

Table 4는 Hadoop 기반의 대용량 정지영상 식별자 생성 구현의 수행 단계별 소요시간을 나타낸다. 각 수행 단계는 분산처리되기 때문에 Job이 시작되고 종료된 시점을 의미하는 것으로 T1~T4 소요시간의 총합이 총 소요시간은 아니다.

Table 4. Time Analysis for Processing Units

Job	T1	T2	T3	T4	Time
Job 1	0min 12sec 656ms	7min 13sec 23ms	3min 6sec 259ms	0min 0sec 271ms	7min 36sec 885ms
Job 2	0min 11sec 418ms	11min 19sec 705ms	5min 25sec 663ms	0min 0sec 226ms	11min 36sec 4ms
Job 3	0min 6sec 343ms	7min 33sec 279ms	1min 48sec 9ms	0min 0sec 215ms	7min 54sec 104ms
Job 4	7min 42sec 550ms	6min 7sec 977ms	5min 27sec 26ms	0min 0sec 230ms	13min 53sec 884ms

Fig. 4는 Hadoop 기반의 대용량 정지영상 식별자 생성 구현의 수행 단계별 소요시간을 분석한 결과이다. Fig. 4를 보면 실질적인 영상처리는 Map에서 이루어지므로 T2는 평균적으로 8분 3초 496밀리세컨드가 소요되고, Reduce인 T4는 235밀리세컨드가 소요된다. 커뮤니케이션 시간인 T1은 2분 3초 442밀리세컨드, T3는 3분 56초 912밀리세컨드를 소모하고 있는데, 이는 T2가 8분 3초 496밀리세컨드가 소요되는 것과 비교하면 상당이 높은 점유율을 가짐을 알 수 있다. 여기에서 T3를 주목해서 보면, T3는 Map의 결과를 Reduce태스크로 전달하는 단계로 본 실험에서 높은 Network I/O시간이 소모됨을 알 수 있다. 현재 제안한 Hadoop 기반의 대용량 정지영상 식별자 생성 방법은 단일 시스템에 비해 효율적이게 대용량 정지영상을 처리하고 있지만, 커뮤니케이션에 많은 시간을 소모하고 있어 개선이 필요하다.

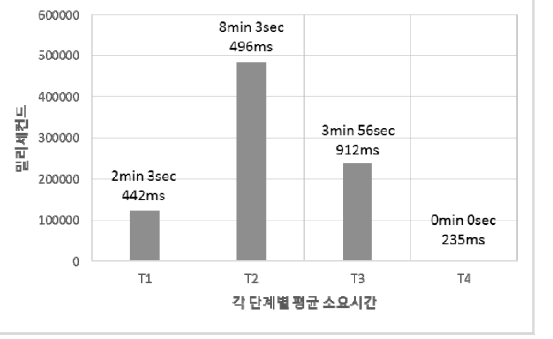

Fig. 5. Average Processing Time in Hadoop based Parallel Processing System.

4. 결 론

본 논문에서는 대용량 정지영상 데이터베이스 환경에서 각 정지영상을 구분할 수 있는 인식자 생성을 병렬처리 환경에 적용을 하여, 실제로 인식자 생성의 병렬처리 환경에서 개선될 수 있는 특성에 대하여 논의를 했다. 정지영상의 크기가 큰 경우, 병렬처리를 통해 얻을 수 있는 성능의 개선은 분명히 확인할 수 있었으나, 정지영상의 크기가 커짐에 따라 네트워트에서 전송에 걸리는 시간이 전체 성능 개선을 악화시키는 잠재성을 고려하는 알고리즘의 필요함을 배울 수 있었다. 따라서, 향후 연구 방향은 성능개선은 유지할 수 있는 새로운 알고리즘 개발에 있다.

감사의 글

이 논문은 2016년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 지역신산업선도인력양성사업 성과임(No. NRF-2016H1D5A1909989)

참고문헌

1. Cheolyoung Jung, “According to rapid Increase in World Digital Data Amount, 40ZB in 2020…300 times of that of 2005”, Digital Times, (2012).
2. Jeho Park, “Still Image Identifier based over Low-frequency Area”, Journal of the Semiconductor & Display Technology, 11(3), pp. 393-398, (2010).
3. Jeho Park, Sangmin Lim, “Composition and Analysis of Linear Component Counting based Multiple Indexing”, Journal of the Semiconductor & Display Technology, 9(3), pp. 17-21, (2010).
4. The Apache Hadoop Software Foundation, “Hadoop 2”, http://hadoop.apache.org, (2017).
5. The Apache Hadoop Software Foundation, “Hadoop 2.6 Mapreduce”, http://hadoop.apache.org/docs/r2.6.4/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html (2017).
6. University of Virginia, HIPI : Hadoop Image Processing Interface, http://hipi.cs.virginia.edu, (2017).
7. Chris Sweeney, Liu Liu, Sean Arietta, Jason Lawrence, “HIPI: A Hadoop Image Processing Interface for Image-based MapReduce Tasks”, B.S. Thesis. University of Virginia, (2011).
8. The Apache Hadoop Software Foundation, Hadoop 2.6 hdfs, https://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/HdfsUserGuide.html, (2017).
9. Ronald C Taylor, “An Overview of the Hadoop/MapReduce/HBase Framework and its current Applications in Bioinformatics”, Proceedings of the 11th Annual Bioinformatics Open Source Conference (BOSC), 11, pp. 1-6, (2010).
10. Jens Dittrich, Jorge-Arnulfo Quiané-Ruiz “Efficient big data processing in Hadoop MapReduce”, Journal Proceedings of the VLDB Endowment, 5, pp. 2014-2015, (2012).

접수일: 2017년 2월 7일, 심사일: 2017년 3월 20일,
게재확정일: 2017년 3월 24일