Journal of the Korean Society of Surveying, Geodesy, Photogrammetry and Cartography
Vol. 35, No. 3, 187-194, 2017
https://doi.org/10.7848/ksgpc.2017.35.3.187

ISSN 1598-4850(Print)
ISSN 2288-260X(Online)
Original article

대용량 위성영상의 무감독 분류를 위한 K-means 군집화 알고리즘의 병렬처리

Parallel Processing of K-means Clustering Algorithm for Unsupervised Classification of Large Satellite Imagery

한수희[1)]
Han, Soohee

Abstract

The present study introduces a method to parallelize k-means clustering algorithm for fast unsupervised classification of large satellite imagery. Known as a representative algorithm for unsupervised classification, k-means clustering is usually applied to a preprocessing step before supervised classification, but can show the evident advantages of parallel processing due to its high computational intensity and less human intervention. Parallel processing codes are developed by using multi-threading based on OpenMP. In experiments, a PC of 8 multi-core integrated CPU is involved. A 7 band and 30m resolution image from LANDSAT 8 OLI and a 8 band and 10m resolution image from Sentinel-2A are tested. Parallel processing has shown 6 time faster speed than sequential processing when using 10 classes. To check the consistency of parallel and sequential processing, centers, numbers of classified pixels of classes, classified images are mutually compared, resulting in the same results. The present study is meaningful because it has proved that performance of large satellite processing can be significantly improved by using parallel processing. And it is also revealed that it easy to implement parallel processing by using multi-threading based on OpenMP but it should be carefully designed to control the occurrence of false sharing.

Keywords : K-means Clustering, Parallel Processing, Unsupervised Classification, Satellite Imagery

초 록

본 연구는 대용량 위성영상의 신속한 무감독 분류를 위해 k-means 군집화 알고리즘을 병렬처리하는 방법을 소개한다. K-means 군집화 알고리즘은 대표적인 무감독분류 알고리즘으로서 주로 감독분류의 전처리 단계로 활용되지만 연산 집약적이고 사용자의 개입이 적어 병렬처리의 효과를 분명하게 나타낼 수 있다. 병렬처리 코드는 OpenMP 기반의 멀티쓰레딩을 이용하여 구현하였다. 실험은 1대의 PC에서 시행하였으며 이 PC의 CPU에는 8개의 멀티코어가 집적되어 있다. 실험 영상으로는 7개 밴드로 구성한 30m 해상도의 LANDSAT 8 OLI 영상과 8개 밴드로 구성한 10m 해상도의 Sentinel-2A 영상을 사용하였다. 각각 10개 군집을 사용하여 순차처리 및 병렬처리를 수행한 결과 병렬처리가 순차처리에 비해 6배 내외의 속도를 나타내었다. 순차처리와 병렬처리 결과의 일치성 평가를 위해 각 군집의 중심값과 분류된 화소의 수를 비교하고 분류 결과 영상간 차분을 수행하였고 결과로 모든 정보가 일치하였다. 본 연구는 병렬처리를 통해 대용량 위성영상의 처리 속도를 상당히 향상시킬 수 있음을 입증하고 있다는 점에서 의미가 있다고 판단된다. 아울러 OpenMP 기반의 멀티쓰레드를 이용하면 비교적 쉽게 병렬처리를 구현할 수 있지만 false sharing의 발생을 억제하도록 코드를 설계하는데 주의를 기울여야 함도 확인할 수 있었다.

핵심어 : K-means 군집화, 병렬처리, 무감독 분류, 위성영상

Received 2017. 05. 29, Revised 2017. 06. 13, Accepted 2017. 06. 20
1) Member, Dept. of Geoinformatics Engineering, Kyungil University (E-mail: scivile@kiu.ac.kr)

1. 서 론

오늘날 위성센서와 통신 기술의 발전으로 인해 위성으로부터 취득한 자료의 용량이 급격히 증가하고 있다. 즉, 위성센서의 기하, 분광, 방사 해상도가 동반 개선되면서 위성영상의 장면(scene) 당 크기가 기하급수적으로 증가하고 있으며, 위성과 지상국간의 통신 속도가 향상됨에 따라 지상국에서 수신, 축적하는 위성영상의 양도 날로 증가하고 있다. 저장매체의 집적도 증가와 가격 하락에 힘입어 자료 저장소(data warehouse)에서 유지할 수 있는 위성영상의 크기도 꾸준히 증가하고 있으며 컴퓨팅 기술의 발달에 힘입어 방대한 위성영상을 처리할 수 있는 여건이 꾸준히 개선되고 있다. 그러나 위성영상 처리에 최적화된 소프트웨어 기술의 개발 없이 단순한 컴퓨터 하드웨어의 개선만으로는 대용량 위성영상을 처리하기 어려우며 새롭고 복잡한 위성영상 처리 알고리즘의 개발은 위성영상 처리 시간을 증가시키는 원인이 되고 있어 처리 시간 단축을 위한 연구의 필요성이 제기되고 있다. 그럼에도 불구하고 대용량 위성영상을 효율적으로 처리하기 위한 투자와 연구가 국내에는 활발하지 않은 것으로 판단된다.

최근 공학, 수학, 과학 등 다양한 분야에서 대용량 자료 처리 및 고연산의 난제를 해결하기 위해 병렬처리를 사용하고 있다. 병렬처리란 크고 복잡한 문제를 작게 나누어 여러 컴퓨터에서 동시에 처리하는 컴퓨팅 기술을 말한다(Wikipedia, 2017c). 공간정보 분야의 자료 역시 방대한 크기를 가지고 있으며 처리 방법 또는 알고리즘의 복잡도가 크게 상승하고 있어 지리정보시스템, 원격탐사, 레이저측량 분야를 필두로 병렬처리의 활용 가능성을 꾸준히 제기하였다(Healey *et al.,* 1997; Clematis *et al.,* 2003; Yang and Hung, 2000; Plaza and Chang, 2007; Han *et al.,* 2009). 특히 원격탐사 분야에서는, 초분광영상의 PPI(Pixel Purity Index) 산출을 병렬화하여 다양한 플랫폼의 병렬처리 하드웨어에서 성능 비교를 수행하였고, 초분광영상의 endmember 분석을 목표로 linear mixing model에 기반한 다양한 알고리즘의 조합인 ORASIS(Optical Real-time Adaptive Spectral Identification System)의 병렬화를 수행하였으며, 야생 화재를 (준)실시간으로 모니터링하고 추적하기 위해 초분광영상에 morphological 분류 방식과 SOM(Self-Organizing Map) 분류 방식을 병렬화하여 적용하는 등 다양한 연구를 수행하였다(Plaza and Chang, 2007). 대체로 국외의 연구들은 초분광영상의 병렬처리라는 공통점을 가지고 있으며 분류 및 인식을 위한 다양한 알고리즘의 병렬화가 시도되었다. 아울러 하드웨어의 특성에 따른 병렬처리 알고리즘의 변형 및 개발이라는 노력도 높이 평가할 수 있다.

국내에는 Koo(2012)가 위성영상에 대한 MTF(Modulation Transfer Function) compensation, 공삼차 보간, 이산웨이블릿 역변환 등을 CUDA를 이용하여 병렬처리하였으며 Lee *et al.*(2016)은 NDVI 영상 생성을 위해 역시 CUDA를 이용하였다. 그 외 GPU기반 위성영상의 모자익 병렬처리 프로토타입 개발, 멀티쓰레딩(multi-threading) 기반 sobel operator 적용, 초병렬처리 알고리즘을 일반적인 병렬처리 환경에서 사용하기 위한 포팅(porting) 등의 사업적 접근이 존재하지만 국외에 비해 미약한 편이다. 이상 과거의 국내외 연구를 종합하면 많은 시간을 필요로 하는 고연산의 알고리즘을 신속하게 처리하는데 집중되어 있지만 최근 실질적으로 문제가 되고 있는 대용량 위성영상의 처리에는 충분히 대응하지 못한 것으로 판단된다.

최근에는 클라우드 컴퓨팅에 대한 관심이 고조되면서 대용량 위성영상을 고성능 서버에서 처리하여 그 결과를 클라이언트에게 공급하는 방식의 활용 가능성도 높아지고 있다. Wang *et al.*(2013)은 최대우도분류와 Mahalanobis distance clustering 알고리즘을 클라우드 컴퓨팅 환경에서 구현하였으며 Sugumaran *et al.*(2015)는 고성능 컴퓨팅 구현의 비용 및 기술적 용이성으로 인해 대용량 원격탐사 자료의 처리에 적합함을 강조하였다. 그러나 클라우드 컴퓨팅은 고성능 컴퓨팅 자원을 인터넷을 통해 다수 클라이언트와 공유하는 것을 근본적인 목표로 하며, 개인 또는 기관이 소유하고 있는 대용량 위성영상을 클라우드 서버에 업로드하여 처리하는 것은 오히려 비효율을 초래할 수 있다. 한편 분산처리는 자료를 분할하여 동시에 처리하는 방식으로서 병렬처리와 유사한 개념으로 사용되지만, 각 노드가 자료를 독립적으로 처리하는 loosely coupled 방식으로서 노드간 적극적인 자료 공유가 전제인 tightly coupled 방식의 병렬처리와 차이가 있다.

본 연구는 대용량 위성영상의 신속한 처리를 위해 병렬처리 활용의 중요성을 역설함에 목표가 있으며, 그 예로서 k-means 군집화 알고리즘을 병렬처리하는 방법을 제시한다. 대표적인 무감독분류 방식인 k-means 군집화 알고리즘은 비교적 코드 구현이 쉽지만 높은 분류 정확도를 기대하기 어려워 주로 감독분류의 전처리 단계로 활용한다. 그럼에도 k-means 군집화 알고리즘을 선택한 이유는 k-means 군집화 알고리즘이 연산 집약적이고 사용자의 개입이 적어 병렬처리의 효과를 분명하고 객관적으로 나타낼 수 있기 때문이다. 아울러 분류 과정중 클래스 정보가 수시로 업데이트되며 모든 화소가 영향을 미치는 전역적 최적화(global optimization) 문제로서 단순히 자료를 나누어 처리하는 분산처리로는 구현하기 어렵다. 본 연구는 일반적인 컴퓨팅 환경에서 병렬처리를

구현하는 것을 목표로 PC 또는 워크스테이션 1대를 사용하며 CPU에 집적된 다중코어(multi-core)를 이용한다. 실험에 사용하는 위성영상은 일반적인 PC 또는 워크스테이션의 메인 메모리 크기에 근접하는 다중분광영상으로 한다.

2. 본 론

2.1 K-means 군집화 알고리즘

MacQueen(1967)이 소개한 k-means 군집화 알고리즘은 수치 또는 공간적으로 분산된 자료에서 k개의 가까운 샘플을 이용하여 값을 산출하는 의미와 전체 자료를 k개의 군집으로 분류하는 의미로 해석될 수 있다. 위성영상을 무감독분류하기 위한 k-means 군집화 알고리즘은 후자에 해당하며 다음과 같은 과정으로 구성된다.

① k개 군집의 밴드별 초기 중심 설정
② 모든 화소를 k개의 군집 중심 중 가장 가까운 군집으로 분류
③ 모든 화소의 분류가 끝나면 각 군집의 중심 재설정
④ ②~③의 과정을 반복하다가 군집이 변경되는 화소의 비율이 임계치 이하일 때 종료

과정 ①에서는 화소의 최대, 최소 차이를 k 개로 분할하여 각 구간의 중심을 군집의 초기 중심으로 삼는다. 일반적으로 과정 ②에서는 유클리디언 거리(Euclidean distance)를 계산하여 가장 가까운 군집을 결정한다. 과정 ③에서는 각 군집에 분류된 화소들의 밴드별 평균값을 계산하여 새로운 군집 중심으로 삼는다. n개의 밴드로 구성된 다중분광 위성영상에서 화소 (i, j)가 분류될 군집 $\hat{k}$는 Eq. (1)과 같이 결정한다.

$$\hat{k} = \text{argmin}(d(k))$$
$$d(k) = \sqrt{\sum_{b=1}^{n}(DN_{i,j,b} - C_{k,b})^2} \quad (1)$$

where $DN_{i,j,b}$: Digital value of band b of pixel (i, j), $C_{k,b}$: Center of band b of cluster k.

위의 과정에 대한 pseudo 코드는 4중 for loop으로 구성되어 있다(Fig. 1). 첫 번째 for loop는 모든 분류 과정의 반복, 두 번째 for loop는 모든 화소의 검색(retrieval), 세 번째 for loop는 모든 군집의 검색, 네 번째 for loop는 모든 밴드의 검색을 의미한다. 네 개의 loop 중 두 번째 loop가 가장 긴 구간으로 이후 병렬화의 핵심 대상이 된다.

2.2 K-means 군집화 알고리즘의 병렬화

K-means 군집화 알고리즘의 병렬화는 Fig. 1의 두 번째 for loop를 CPU의 멀티코어가 그 수대로 분할하여 각자 처리함으로서 구현한다. 즉 n개의 코어가 각각 1/n 크기의 영상을 동시에 분류하는 구조를 가진다. 이는 OpenMP(OpenMP ARB, 2016) 기반의 멀티쓰레딩 프로그래밍을 이용하며 실현할 수 있으며, 두 번째 for loop 앞에 #pragma omp parallel for 구분

```
for (iter = 0 to nMaxIter) //iteration
        for (i = 0 to nRow * nColumn) //for all pixels
                for (j = 0 to nClass) //for all classes
                        for (k = 0 to nBand) //for all bands
                                c = EuclideanDist(p_Image[j][i], p_Centers[j][k])
                p_Class[i] = argmin_j(c) //classifying pixel i to class j to minimize c
                if (isChanged(p_Class[i])) nChanged++ //updating # of reclassified pixels
                p_Members[p_Class[i]]++ //updating # of pixels in class j
                update p_Centers[p_Class[i]] //updating center of class j
        if (nChanged < nThreshold) break //terminating condition
```

Fig. 1. Pseudo code of k-means clustering

```
for (iter = 0 to nMaxIter) //iteration
#pragma omp parallel for private(i, j, k, c) reduction(+:nChanged) //implementation of multi-thread
```

```
for (i = 0 to nRow * nColumn) //for all pixels
        for (j = 0 to nClass) //for all classes
                for (k = 0 to nBand) //for all bands
                        c = EuclideanDist(p_Image[j][i], p_Centers[j][k])
        p_Class[i] = argmin_j(c) //classifying pixel i to class j to minimize c
        if (isChanged(p_Class[i])) nChanged++ //updating # of reclassified pixels
        p_Members[p_Class[i]]++ //updating # of pixels in class j
        update p_Centers[p_Class[i]] //updating center of class j
if (nChanged < nThreshold) break //terminating condition
```

Fig. 2. Implementation of multi-thread based on OpenMP

을 넣어 구현한다(Fig. 2).

그러나 OpenMP는 false sharing이라는 문제점을 안고 있으며 false sharing을 해결하지 않으면 오히려 병렬처리를 사용하지 않는 경우(이후로는 순차처리)보다 많은 시간을 소모하게 된다. False sharing이란 서로 다른 코어가 동일한 캐시라인에 있는 변수를 수정할 때 발생하는 현상으로 캐시 일관성을 유지하도록 메모리가 강제 업데이트되기 때문에 처리 속도가 크게 저하되는 현상을 말한다(Wikipedia, 2017b). For loop 안에 존재하는 동일한 변수의 내용을 서로 다른 코어가 동시에 변경시키면 false sharing 현상이 발생하며, 변수의 내용이 변경되는 for loop를 단순하게 병렬화할 경우 빈번하게 발생한다. False sharing의 발생을 억제하기 위해서는 Fig. 2의 private(i, j, k, c)와 같이 parallel for 구분 이후에 사용되는 변수 중 그 값이 변하는 변수에 대하여 private() 구문에 넣어 주어야 한다. private() 구문에 넣은 변수들은 코어마다 독립적으로 선언되어 서로 간섭을 일으키지 않는 효과를 낸다고 볼 수 있다. 특히 nChanged 변수는 모든 코어에서 군집이 바뀐 화소의 수를 합산하기 위한 변수로서 false sharing 현상을 발생시키지 않으면서 합산을 구하기 위해 reduction(+:nChanged)라는 추가적인 구문을 사용한다. 한편 p_Class는 각 화소가 분류된 군집을 저장하기 위한 포인터 배열로서 i 값의 범위가 코어마다 다르므로 p_Class[i]는 코어별로 배타적인 주소를 가리킨다. 따라서 false sharing 문제를 발생시키지 않는다.

그럼에도 불구하고 false sharing은 여전히 발생할 수 있다. p_Members와 p_Centers는 포인터 배열로서 각 군집에 속한 화소의 수와 밴드별 중심값을 담는다. 하지만 코어별로 i가 다르더라도 화소 i가 분류된 군집(p_Class[i])은 같을 수 있으므로 서로 다른 코어에서 동시에 같은 변수에 접근하여 값을 변경하려 할 수 있으며 이 때 false sharing이 발생한다. 이 문제를 해결하기 위하여 Fig. 3과 같이 p_Members와 p_Centers를 전역변수로 선언하고 p_Members_local과 p_Centers_local을 코어마다 로컬(local)변수로 선언한다. 각 코어에서는 전역변수의 군집 중심값을 이용하여 분류를 수행하고 로컬변수에 군집별 화소의 수 및 화소값을 누적한다. 분류가 끝나면 모든 코어의 로컬변수를 전역변수에 합산하고 평균하여 군집의 중심값을 재계산한다(Fig. 4). 이러한 과정을 통해 false sharing을 최종적으로 억제한다.

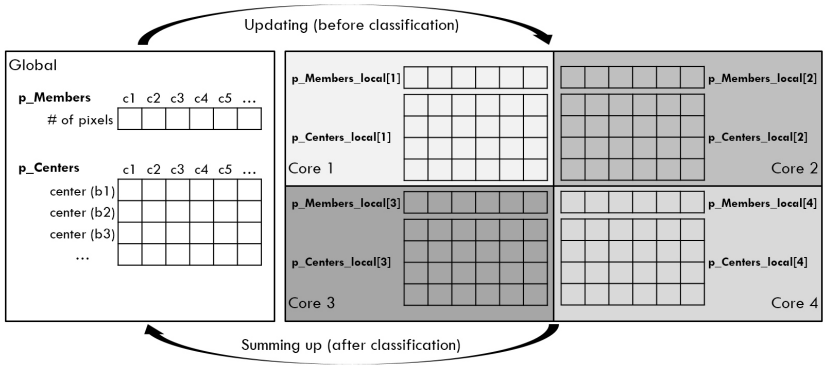

Fig. 3. Global and local declaration of p_Members and p_Centers

```
for (i=0 to nClass) //for all classes
        for (j=0 to nBand) //for all bands
                p_Centers_local[i][j] = 0; //local class center
                p_Members_local[i] = 0; //local # of class members
                for (n = 0 to nThread) //for all threads(cores)
                        p_Centers[i][j] += p_Centers_local[n][i][j];
                        p_Members[i] += p_Members_local[n][i];
                        if (p_Members[i] != 0) //recalculation of class centers
                                p_Centers[i][j] = p_Centers[i][j]/p_Members[i];
                        else
                                p_Centers[i][j] = -1;
```

Fig. 4. Recalculation of class centers

Table 1. Specifications of satellite images

Sensor	LANDSAT 8 OLI	Sentinel 2A
Row	7841	30978
Column	7691	40980
Applied no. of bands	7 (1, 2, 3, 4, 5, 6, 7)	8 (2, 3, 4, 5, 6, 7, 8, 8A)
Geometric resolution	30m	10m (5, 6, 7, 8A resampled)
Radiometric resolution	16bit	16bit
Image size	868.38MB	18.92GB

Table 2. Specifications of processing system

CPU	AMD FX - 8150 eight cores @ 3.60GHz
RAM	32GB (DDR3)
HDD	512GB (7200rpm)
OS	Windows server 2008
Compiler	Visual studio 2010

Table 3. Parameters of k-means clustering

No. of classes(=k)	10
Max no. of iterations	100
Terminating condition	1.5% of pixels moved to different classes

3. 적용 및 평가

순차코드와 병렬처리 코드를 두 가지 다중분광 위성영상에 적용하여 처리 시간을 비교하고 분류 결과의 일치성을 확인하였다. 위성센서는 LANDSAT 8 OLI와 Sentinel 2A이며 처리에 사용한 PC의 CPU에는 8개의 물리적 멀티코어가 장착되어 있다. 위성영상, 처리 시스템, k-means 군집화 알고리즘의 파라미터는 Tables 1, 2 and 3과 같다.

처리 시간은 영상 입력 시간, 분류 시간, 결과 저장 시간으로 구분하여 측정하였으며 그 결과는 Tables 4 and 5와 같다. LANDSAT 8 OLI 영상의 경우 병렬처리가 순차처리에 비해 5.4배 빠르며 영상 입출력 시간을 제외한 순수 분류과정은 5.9

배 빠르다. 분류 반복 횟수는 8번이며 분류가 종료될 때의 화소 이동 비율은 1.34%이다. Sentinel-2A 영상의 경우 병렬처리가 순차처리에 비해 5.9배 빠르며 영상 입출력 시간을 제외한 순수 분류과정은 6.4배 빠르다. 분류 반복 횟수는 15번이며 분류가 종료될 때의 화소 이동 비율은 1.49%이다.

영상의 입출력 시간은 하드디스크 또는 기타 시스템의 상태에 따라 다소 다르게 측정된 것으로 보인다. CPU에 의한 순수 분류 시간만으로 평가하면 병렬처리를 적용할 때 LANDSAT 8 OLI 영상에 대해서는 5.9배, Sentinel-2A 영상에 대해서는 6.4배로서 6배 내외의 처리속도를 나타내었다. 코어가 8개이므로 이상적으로는 8배의 속도를 기대할 수 있지만 모든 처리과정을 병렬화할 수 있는 것이 아니며, 병렬처리의 성능 체감 효과, 즉 Amdahl's law(Wikimedia, 2017a)에 의해 달성하기 어려운 것으로 알려져 있다.

순차처리와 병렬처리 결과의 일치성 평가를 위해 각 군집의 중심값과 분류된 화소의 수를 비교하였으며 결과로 모든 정보가 일치함을 확인하였다(Table 6). 다음으로 분류 결과 영상간 차분을 수행하였으며 결과로 차분 영상에서도 0값 이외의 이상점을 발견할 수 없었다(Figs. 5 and 6). 따라서 순차처리의 결과와 병렬처리의 결과가 완벽히 일치함을 알 수 있다.

4. 결 론

본 연구에서는 대용량 위성영상의 무감독 분류를 위해 OpenMP를 기반으로 k-means 군집화 알고리즘의 병렬처리 코드를 개발하고 8 멀티코어 CPU가 장착된 PC에서 다중분광 위성영상에 대한 병렬처리를 수행하였다. 그 결과 순차처리에 비해 6배 내외의 속도 향상 효과를 거둘 수 있었다. 최근 위성영상의 분류 정확도를 향상시키기 위한 다양한 알고리즘이 개발되고 있는 상황에서 k-means 군집화 알고리즘은 분류 정확도 면에서 우수하다고 평가하기 어렵다. 또한 본 연구에서 개발한 코드가 다른 알고리즘의 병렬처리에 바로 사용될 수 없다는 한계를 지니고 있다. 그럼에도 불구하고 본 연구는 병렬처리를 통해 대용량 위성영상의 처리속도를 상당히 향상시킬 수 있음을 입증하고 있다는 점에서 의미가 있다고 판단된다. 아울러 OpenMP와 멀티코어 CPU를 이용하면 비교적 쉽게 병렬처리를 구현할 수 있지만 false sharing의 발생을 억제하도록 코드를 설계하는데 주의를 기울여야 함도 확

Table 4. Processing time of LANDSAT 8 OLI

	Sequential processing	Parallel processing
Image input	1.23 sec	1.37 sec
Classification	272.07 sec	45.93 sec
Result output	4.07 sec	4.09 sec
Total	277.37 sec	51.39 sec

Table 5. Processing time of Sentinel-2A

	Sequential processing	Parallel processing
Image input	228.81 sec	108.08 sec
Classification	11879.01 sec	1846.00 sec
Result output	102.69 sec	99.45 sec
Total	12210.51 sec	2053.53 sec

Table 6. Sample of class information of LANDSAT 8 OLI

class	band	center	members	class	band	center	members
0	0	11572.87	18536730	1	0	10884.18	17020681
	1	10617.72	18536730		1	9955.15	17020681
	2	9088.32	18536730		2	9447.00	17020681
	3	8316.74	18536730		3	8242.52	17020681
	4	7906.71	18536730		4	18773.11	17020681

class	band	center	members	class	band	center	members
0	5	7157.54	18536730	1	5	11848.03	17020681
	6	6789.57	18536730		6	8517.65	17020681
2	0	12010.57	5191841	3	0	12879.99	812003
	1	11347.66	5191841		1	12518.95	812003
	2	11012.07	5191841		2	12527.42	812003
	3	10640.80	5191841		3	12925.43	812003
	4	17726.02	5191841		4	18672.48	812003
	5	14637.83	5191841		5	18367.14	812003
	6	11672.85	5191841		6	15553.45	812003
…	…	…	…	…	…	…	…

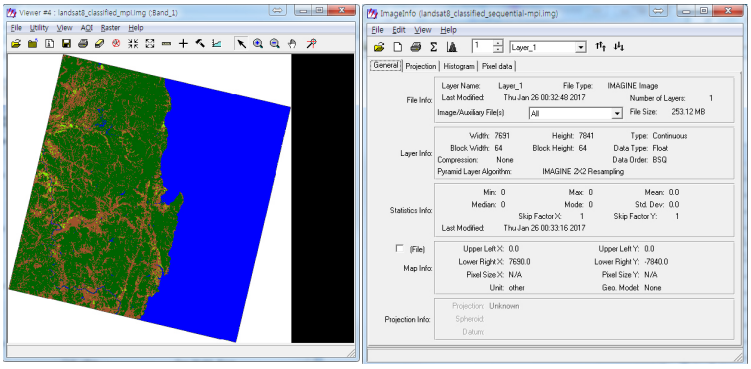

Fig. 5. Results of LANDSAT 8 OLI: (left) classified results, (right) statistics of differential image(min=max=0)

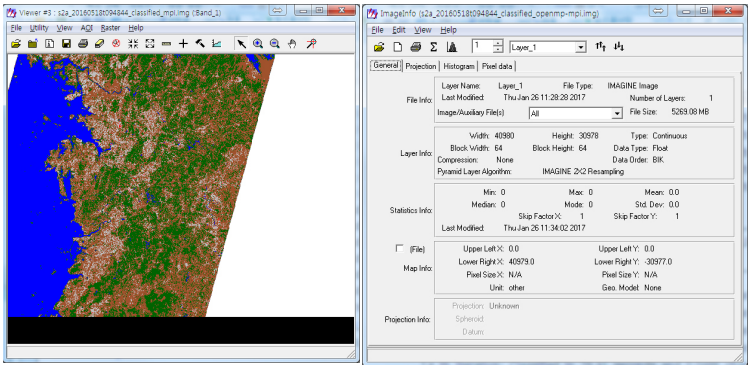

Fig. 6. Results of Sentinel-2A: (left) classified results, (right) statistics of differential image(min=max=0)

인할 수 있었다.

향후 연구에서는 여러 대의 PC로 구성된 PC-cluster를 기반으로 out-of-memory, 즉 PC 1대로 처리하기 어려운 초대용량 위성영상을 효율적으로 처리할 수 있는 병렬처리 코드를 개발하여 적용하고자 한다.

감사의 글

이 연구는 2016년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행되었음 (NRF-2016R1C1B1013973)

References

Clematis, A., Mineter, M., and Marciano, R. (2003), High performance computing with geographical data, *Parallel Computing,* Vol. 29, Issue 10, pp. 1275-1279.

Han, S.H., Heo, J., Sohn, H.G., and Yu, K. (2009), Parallel processing method for airborne laser scanning data using a PC cluster and a virtual grid, *Sensors,* Vol. 9, Issue 4, pp. 2555-2573.

Healey, R., Dowers, S., Gittings, B., and Mineter, M.J. (1997), *Parallel Processing Algorithms for GIS,* CRC Press, UK.

Koo, I.H. (2012), *High-speed Processing of Satellite Image Using GPU,* Master's thesis, Chungnam National University, Daejeon, Korea, pp. 28-42. (in Korean with English abstract)

Lee, K., Jo, M., and Lee, W. (2016), Parallel processing of satellite images using CUDA library: focused on NDVI calculation, *Journal of the Korean Association of Geographic Information Studies,* Vol. 19, No. 3, pp. 29-42. (in Korean with English abstract)

MacQueen, J. (1967), Some methods for classification and analysis of multivariate observations, *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics,* University of California Press, Berkeley, California, USA, 21 June–18 July, pp. 281-297.

OpenMP ARB (2016), The OpenMP API specification for parallel programming, *OpenMP ARB,* http://www.openmp.org (last date accessed: 25 May 2017).

Plaza, A.J. and Chang, C. (2007), *High Performance Computing in Remote Sensing,* CRC Press, UK.

Sugumaran, R., Hegeman, J.W., Sardeshmukh, V.B., and Armstrong, M.P. (2015), Processing remote-sensing data in cloud computing environments, In: Thenkabail, P.S. (ed.), *Remotely Sensed Data Characterization, Classification, and Accuracies,* CRC Press, UK, pp. 549-558.

Wang, P., Wang, J., Chen, Y., and Ni, G. (2013), Rapid processing of remote sensing images based on cloud computing, *Future Generation Computer Systems,* Vol. 29, Issue 8, pp. 1963-1968.

Wikipedia (2017a), Amdahl's law, *Wikimedia Foundation, Inc.,* https://en.wikipedia.org/wiki/Amdahl%27s_law (last data accessed: 25 May 2017).

Wikipedia (2017b), False sharing, *Wikimedia Foundation, Inc.,* https://en.wikipedia.org/wiki/False_sharing (last data accessed: 25 May 2017).

Wikipedia (2017c), Parallel computing, *Wikimedia Foundation, Inc.,* https://en.wikipedia.org/wiki/Parallel_computing (last data accessed: 25 May 2017).

Yang, C. and Hung, C. (2000), Parallel computing in remote sensing data processing, *Proceedings of the 21st Asian Conference on Remote Sensing,* ACRS, 4-8 December, Taipei, Taiwan, unpaginated CD-ROM.