Journal of KIIT. Vol. 17, No. 4, pp. 21-26, Apr. 30, 2019. pISSN 1598-8619, eISSN 2093-7571 21
http://dx.doi.org/10.14801/jkiit.2019.17.4.21

GPGPU를 사용한 효율적인 공간 데이터 클러스터링 기법

오 병 우*

An Efficient Technique for Clustering of Spatial Data Using GPGPU

Byoung-Woo Oh*

이 연구는 금오공과대학교 학술연구비에 의하여 지원된 논문임

요 약

최근 위치 및 지도의 활용이 늘면서 공간 데이터의 사용이 급격히 증가하고 있고 공간 데이터로부터 지식을 추출하는 기술이 활발히 연구되고 있다. 빅 데이터나 데이터마이닝 기술에서 클러스터링이 중요한 역할을 차지하고 있듯이 공간 데이터 분야에서도 지식을 추출하기 위해서는 공간 클러스터 발견이 중요하다. 공간 데이터는 대용량 데이터이고 점뿐만 아니라 복잡하고 가변 길이인 선과 면을 처리해야 하므로 특히 성능이 강조된다. 본 논문에서는 공간 데이터 클러스터링을 효율적으로 수행하기 위하여 기존의 공간 데이터 클러스터링 알고리즘을 GPGPU를 활용하도록 개선하는 방법을 제안한다. 다수의 GPGPU 코어를 활용하여 공간 데이터를 병렬로 처리하여 성능을 향상시키는 방법을 사용한다. 제안한 방법의 개선된 성능을 측정하기 위하여 실제 공간 데이터 셋을 구축하고 비교 실험을 수행한다.

Abstract

Recently, the use of spatial data has been increasing rapidly due to the increased use of location and map, and techniques for extracting knowledge from spatial data have been actively studied. Just as clustering plays an important role in big data and data mining technologies, finding spatial clusters is important for extracting knowledge in the field of spatial data. The performance of the spatial data processing is particularly emphasized because the spatial data is a large amount of data and requires processing of not only points but also complex and variable length lines and polygons. In this paper, I propose a method to improve spatial data clustering efficiently by using GPGPU. I use a method to improve the performance by processing the spatial data in parallel by utilizing multiple GPGPU cores. In order to measure the improved performance of the proposed method, an actual spatial data set is constructed and a comparative experiment is performed.

Keywords
spatial data, grid, clustering, parallel processing, GPGPU

* 금오공과대학교 컴퓨터공학과 교수
- ORCID: https://orcid.org/0000-0003-4700-9961

· Received: Mar. 10, 2019, Revised: Apr. 06, 2019, Accepted: Apr. 09, 2019
· Corresponding Author: Byoung-Woo Oh
Dept. of Computer Engineering, Kumoh National Institute of Technology, Korea
Tel.: +82-54-478-7531, Email: bwoh@kumoh.ac.kr

Ⅰ. 서 론

최근 사회 전반에 걸친 데이터의 공급 및 수요가 늘어나면서 컴퓨터에 축적된 데이터로부터 지식을 추출하기 위한 데이터 마이닝 및 빅 데이터 기술에 대한 관심이 높아지고 있다. 다차원의 데이터로부터 서로 관련성이 높은 클러스터를 발견하기 위한 클러스터링은 빅 데이터 처리나 데이터 마이닝을 위한 전처리 단계로 활용될 수 있는 중요한 기술이어서 클러스터링 기법에 대한 연구가 꾸준히 지속되고 있다[1].

특히 모바일 환경의 발전에 따라 위치 및 지도를 사용한 서비스가 급증하면서 공간 데이터의 활용이 늘어나고 이에 따라 공간 데이터의 분석에 대한 수요도 증대되고 있다. 그러나 공간 데이터는 주로 2차원 데이터로 표현하며 기존의 데이터 처리 방법으로는 처리하기 힘든 특성을 가지고 있다. 예를 들어, 1차원 비공간 데이터간의 유사도는 값의 차로 간단히 구할 수 있다면, x축과 y축으로 구성된 2차원 공간에서 Polygon 형태의 면간의 유사도는 간단히 구할 수 없는 특성을 가지고 있다. 공간 데이터를 처리하기 위해서는 기존의 방법과 다른 별도의 처리 방법이 필요하므로, 대용량 2차원 가변 길이 데이터인 공간 데이터 특성을 반영한 공간 클러스터링 성능 향상에 대해서도 다양한 연구가 활발히 진행되고 있다[2][3].

성능 향상이 필요한 다양한 분야에서 기존 CPU의 한계를 인식하고 GPU(Graphics Processing Units)를 범용 계산에 활용하는 GPGPU(General-Purpose Computing on GPU) 기술이 활용되고 있다. 다수의 코어를 활용하여 병렬로 데이터를 처리함으로써 성능을 향상시키기 위한 방법으로서 공간 클러스터링에 활용하는 연구도 활발히 진행되고 있다[4]-[9].

본 논문에서는 기존의 해시-기반 공간 클러스터링 방법에 GPGPU 기반의 병렬 처리 기법을 적용하여 공간 데이터 클러스터링의 효율성을 높이기 위한 방법을 제안한다. 공간 클러스터를 발견하기 위해서는 임계치보다 큰 밀도를 갖는 Grid Cell들을 연결하는 방법을 사용하므로 Grid를 구축하는 과정이 필요하다. 본 논문에서는 해시 함수를 통해 공간 데이터를 해당 Grid Cell에 소속시켜서 전체적인 Grid를 구축하는 과정을 GPGPU를 활용하여 병렬로 처리함으로써 성능 향상을 도모하는 방법을 제안한다. 크기가 다른 실험 데이터 셋을 구성하고 Grid 크기를 변경하면서 기존 알고리즘과 제안한 병렬 알고리즘을 비교하여 성능 향상 정도에 대해 실험한다.

본 논문의 구성은 다음과 같다. 2장은 관련연구로서 H-Scan에 대해 살펴본다. 3장에서는 본 논문에서 제안하는 GPGPU를 사용한 병렬 H-Scan 클러스터링 방법에 대해 설명한다. 4장에서는 기존의 H-Scan과 본 논문에서 제안하는 GPGPU를 사용하는 공간 클러스터링 방법의 성능 평가를 실시하고 결과를 설명한다. 마지막으로 5장은 결론으로서 본 연구의 의의 및 향후 연구 방향에 대하여 기술한다.

Ⅱ. 관련 연구

밀도에 근거한 클러스터링 방법으로 많이 활용되는 방법으로는 DBSCAN이 있다[10]. DBSCAN 및 DBSCAN에서 파생된 GPGPU를 활용한 개선 알고리즘들은 다차원 Point 데이터를 처리하기에 유용하지만 Point뿐만 아니라 Polyline이나 Polygon 타입을 갖는 공간 데이터 처리에는 데이터간의 거리를 계산하기 힘들어 적용이 어렵다. 그 이외의 GPGPU를 활용한 개선 알고리즘들도 일반적으로 Polyline이나 Polygon 타입을 처리하기 힘든 단점이 있다.

공간 데이터 처리에 적합한 해시 기반의 공간 클러스터링 방법으로는 H-Scan 알고리즘이 있다[11]. H-Scan 알고리즘은 공간을 일정한 크기의 Grid Cell로 분할하고 Point, Polyline, Polygon 타입의 공간 데이터가 위치한 Cell에 배치하여 Cell 단위로 처리한다. 각 i 축에 대해 $\min_i$로부터 $\max_i$까지의 공간을 m_i개로 나누어 d-차원을 총 k개의 Cell로 분할한다. k개의 Cell에 대해, 주어진 $threshold$ 임계치를 가지고 이웃하는 Cell간의 직접 연결 및 연결 관계를 분석하여 공간 클러스터를 발견한다. 이 과정에서 $threshold$로 잡음을 제거할 수 있다. 본 논문에서는 H-Scan 알고리즘을 GPGPU 기술을 활용하여 개선하는 병렬 H-Scan 알고리즘을 제안한다.

III. 병렬 H-Scan 클러스터링 방법

본 논문에서는 공간 클러스터링을 위한 기존의 H-Scan 알고리즘을 GPGPU 기술을 활용하여 병렬로 처리함으로써 성능을 향상시킬 수 있는 병렬 H-Scan 방법을 제안한다. 병렬 H-Scan은 H-Scan에서 Grid를 구축하는 부분을 병렬 알고리즘을 사용하여 성능을 향상시킨다.

3.1 자료 구조

제안하는 공간 데이터 클러스터링의 전체 프로세스는 데이터 로딩, Grid 구축, 클러스터 발견, 데이터 해제의 4단계로 구성된다. 데이터 로딩 단계는 원본 공간 데이터를 저장 장치로부터 메모리로 읽어 들이는 과정이다. 본 논문에서는 원본 공간 데이터로 미국의 TIGER/Line 데이터를 사용한다[12]. TIGER/Line 데이터는 미국에서 인구 조사를 위해 구축되었고 누구에게나 무료로 공개되어 있으므로 연구 목적으로 활용하기에 적합하며 실제 공간 데이터 응용 분야에서도 활용이 가능하다. TIGER/Line 데이터는 ESRI사의 Shape 파일 포맷으로 저장되어 있다. Shape 파일 포맷은 Point, Polyline, Polygon 등의 공간 데이터를 저장할 수 있으며, 다양한 GIS (Geographic Information System)에서 지원하는 De Facto 표준이다. 국내 지도 제작 및 유통에도 널리 사용되고 있다.

본 논문에서는 Shape 파일의 Polyline 타입을 사용하며 공간 데이터를 처리하기 위해 메모리에 저장하는 구조는 그림 1과 같다.

그림 1에서 공간 데이터 셋 전체의 MBR (Minimum Bounding Rectangle)을 저장하는 변수는 mX1, mY1, mX2, mY2이다. mCountData는 전체 공간 데이터 개수를 저장하고, mCountPoints는 공간 데이터 셋에 포함된 모든 좌표의 개수를 저장한다. Polyline 타입 공간 데이터는 여러 개의 파트로 구성될 수 있으며 각 파트는 선을 구성하는 좌표를 갖는다. mCountParts는 공간 데이터 셋 전체의 파트 수를 저장한다. 그리고 H-Scan을 처리하기 위해 차원 수 d, 각 축의 Cell 길이인 cell_size[], 각 축의 Cell 개수 m[], 전체 Cell 개수 k, Grid 구축의 결과로 Cell에 소속된 공간 데이터 개수의 최댓값인 max_n_of_cells가 있다.

mData는 공간 데이터를 저장하는 배열로서 타입은 SpatialData이고 배열의 크기는 mCountData이다. 공간 데이터별로 MBR을 rect에 저장하고, 몇 개의 파트로 구성되는지 nParts, 몇 개의 좌표로 구성되는지 nPoints를 갖고, mParts와 mPoints에서 해당 공간 데이터가 어디서부터 시작되는지 인덱스를 각각 indexParts와 indexPoints에 저장한다.

mPoints는 POINT 타입의 배열로 전체 공간 데이터 셋에 포함된 모든 좌표를 저장하고 배열의 크기는 mCountPoints이다. 본 논문에서는 공간 데이터를 효율적으로 처리하기 위하여 연속적인 메모리 공간에 모든 좌표를 배열로 저장한다.

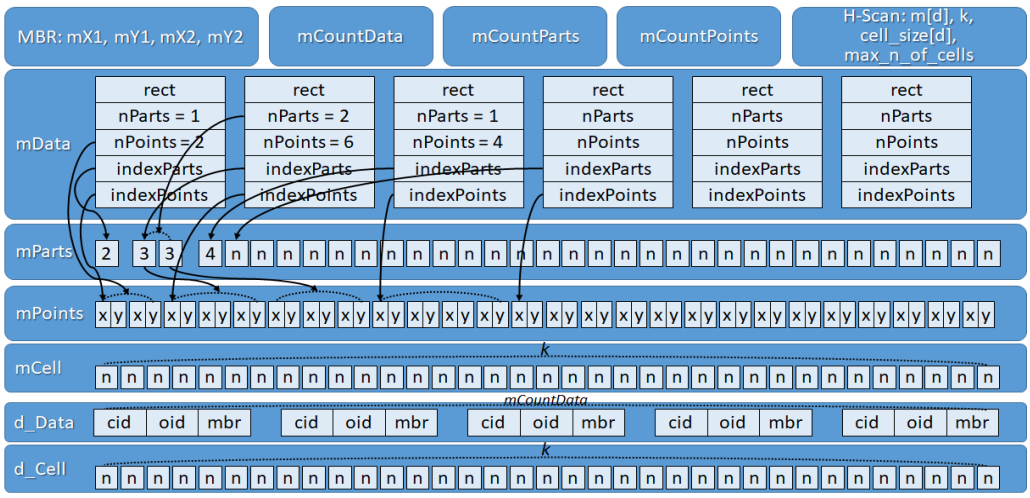


그림 1. 병렬 공간 데이터 클러스터링을 위한 자료 구조
Fig. 1. Data structure for the parallel clustering of spatial data

각각의 공간 데이터는 mPoints 배열에서 몇 번째부터 해당 데이터의 좌표가 시작되는지 앞서 mData에서 설명한 바와 같이 indexPoints 변수에 mPoints 배열의 index를 저장하고 있다.

int 타입의 mParts는 Shape 파일의 Polyline 타입의 공간 데이터가 여러 개의 파트로 구성될 수 있으므로 몇 개로 구성되는지 파트의 수들을 mCountParts 개의 배열로 저장한다. mData의 indexParts가 해당 데이터의 시작 인덱스를 저장한다. 그림 1에서 두 번째 공간 데이터는 2개의 파트(nParts=2)로 구성되어 있고, 첫 번째 파트는 3개의 좌표를 가지고, 두 번째 파트도 3개의 좌표를 가지고 있어 총 6개의 좌표(nPoints=6)로 구성되는 예제를 볼 수 있다.

mCell은 Grid를 구성하는 Cell 데이터로서 몇 개의 공간 데이터가 해당 Cell에 소속하는지를 int 타입으로 저장한다. 배열의 크기는 k로서 m_i를 모두 곱한 수이고, 2차원에서는 x축에 해당하는 m_0와 y축에 해당하는 m_1의 곱으로서 mCell[m_1][m_0] 형태의 2차원 배열로 인식할 수 있다. 실제로는 mCell[m_0*m_1]의 1차원 배열로 처리한다. 배열에 저장된 가장 큰 정수는 max_n_of_cells에 저장한다.

d_Data 및 d_Cell은 GPU에서 사용하는 포인터로서 device를 의미하는 “d_”를 prefix로 갖는다. d_Data는 데이터 로딩 단계에서 클래스 ID인 cid, 객체 ID인 oid, 공간 데이터가 차지하는 영역인 MBR을 mbr에 저장하고 CPU로부터 GPU로 전송한다. 이 과정은 최초 데이터 로딩 단계에서 한 번만 전송한 뒤에 추후의 클러스터링에서는 계속 재사용된다. d_Cell은 GPGPU를 사용한 병렬 클러스터링 과정을 통해 Grid 구축 단계가 완료된 후에 GPU로부터 CPU의 mCell로 전송한다.

3.2 병렬 처리

데이터 로딩 단계 후 GPU에 데이터가 로딩되면 Grid 구축 단계를 수행한다. 그림 2는 Grid 구축을 위한 병렬 H-Scan 알고리즘을 보여준다. 1라인의 InsertSpatialData() 함수는 모든 공간 데이터에 대해 Grid를 구축하는 함수로서 2라인에서 InsertToCell() 함수를 병렬로 호출한다.

```
1  function InsertSpatialData() in CPU {
2     InsertToCell() in a Parallel Way
3     MemcpyDeviceToHost(mCell, d_Cell, sizeof(CELL)*k)
4     for each cell in mCell
5        assign maximum value to max_n_of_cells
6  }
7  function InsertToCell() in GPGPU Device {
8     cells = Hash(MBR of the given spatial data)
9     for each cell in cells
10       atomicAdd(cell, 1)
11 }
```

그림 2. 병렬 H-Scan 알고리즘
Fig. 2. Algorithm of parallel H-Scan

7라인의 InsertToCell() 함수는 다수의 GPU 코어에서 병렬로 각각의 공간 데이터를 해당하는 Cell에 할당한다. 기존 H-Scan에서는 InsertToCell() 함수가 CPU에서 순차적으로 실행되었지만 병렬 H-Scan에서는 GPU에서 병렬로 실행되는 차이점이 있다.

공간 데이터가 어느 Cell에 할당될 지를 8라인의 Hash()함수를 통해 cells로 반환받고, 9라인에서는 cells에 속한 각각의 Cell들에 대해 10라인에서 밀도 값을 1 증가시킨다. 이 때 여러 쓰레드가 GPU에서 병렬로 처리되고 있으므로 GPU의 global 메모리에 할당된 d_Cell의 같은 Cell 주소에 밀도 값을 동시에 증가시키면서 값이 누락되는 것을 방지하기 위하여 atomicAdd()를 사용한다.

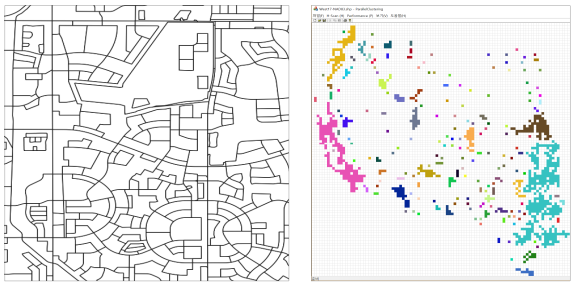
모든 공간 데이터를 병렬로 처리한 뒤에는 3라인과 같이 GPU의 메모리에 할당된 d_Cell을 CPU에서 사용할 수 있도록 mCell로 메모리 복사한다. 4라인에서는 mCell에 속한 Cell들에 대해서 루프를 돌면서 5라인에서 Cell들 중에서 가장 큰 밀도 값을 찾아서 max_n_of_cells에 할당한다. 그 뒤에 클러스터 발견 단계와 데이터 해제 단계를 거쳐 전체 과정을 종료한다.

IV. 성능 평가

4.1 실험 환경

본 논문에서 제안하는 알고리즘의 성능을 평가하기 위한 실험 환경은 Intel(R) Core(TM) i7-6700K CPU 4.00GHz, 64GB RAM, MS Windows 10 Pro 64 비트 운영체제이고, GPU는 3,584개의 코어를 가지고 11GB 메모리를 가진 GeForce GTX 1080Ti이다.

실험 데이터는 TIGER/Line의 Census Block 데이터를 미국의 서부 캘리포니아부터 동부 방향 주(State)별로 점진적으로 병합하여 사용한다.



(a) 공간 데이터 셋 예제 (b) 공간 클러스터 예제

그림 3. 데이터 예제

Fig. 3. Example data, (a) Census block data set, (b) Spatial clusters

그림 3의 (a)는 실험 데이터로 사용한 Census Block 데이터의 예제, (b)는 결과로 생성된 클러스터의 예제를 보여준다.

표 1은 실험에 사용한 5개 공간 데이터 셋을 보여준다. West-1 데이터는 캘리포니아 주이고, West-4는 워싱턴, 오레곤, 네바다 주를 병합한 데이터이다. West-7은 아이다호, 유타, 애리조나를 병합한 데이터이고, West-11은 몬테나, 와이오밍, 콜로라도, 뉴멕시코 주를 병합한 데이터이다. West-17은 노스 다코다로부터 텍사스까지를 병합한 데이터이다. 더 많은 주 데이터를 병합하면 Shape 파일의 크기가 4GB 한계를 초과하여 병합이 불가능하므로 17개 주까지 병합한 파일을 실험 데이터로 사용한다.

표 1. 실험 공간 데이터

Table 1. Experimental spatial data

name	file size	num. of data & points	shape
West-1	667,359KB	710,145 data 40,220,787 points	
West-4	1,302,542KB	1,186,878 data 79,199,571 points	
West-7	1,920,913KB	1,693,792 data 116,996,152 poits	
West-11	2,711,801KB	2,281,955 data 165,548,529 points	
West-17	4,127,307KB	4,119,385 data 249,695,062 points	

4.2 실험 결과

제안한 병렬 H-Scan 알고리즘의 성능 평가를 위해서 본 연구에서는 앞서 살펴본 5개 공간 데이터 셋별로 Grid를 구성하는 Cell의 크기를 변화시키며 각각 100번을 수행한 평균값을 측정한다. 표 2는 실험 결과를 보여준다.

표 2에서 CPU에서는 Grid를 구성하는 Cell 개수 및 공간 데이터 개수가 증가함에 따라 처리 시간이 증가하지만, GPGPU를 사용할 때는 공간 데이터의 개수에는 영향을 현저히 적게 받고 Cell 개수에 의한 영향도 비교적 적게 받는 것을 확인할 수 있다. 특히, 성능 차이는 평균 19.09배로서 최소 3.92배에서 최대 44.99배까지 제안한 병렬 알고리즘이 우수함을 확인할 수 있다. 병렬 H-Scan은 4백만 개 이상의 Polyline 공간 데이터에 대해 5ms 이내로 Grid를 구축할 수 있어서 다른 클러스터링 알고리즘과 시간적으로 비교하면 매우 효율적이다[4]-[9].

그림 4는 Grid를 구성하는 Cell의 개수가 1000×1000일 때 (즉, k=1,000,000) 공간 데이터 셋별 성능을 그래프로 표현한 것으로서 기존의 H-Scan에 비해 본 논문에서 제안한 병렬 H-Scan 방법의 성능이 우수함을 알 수 있다.

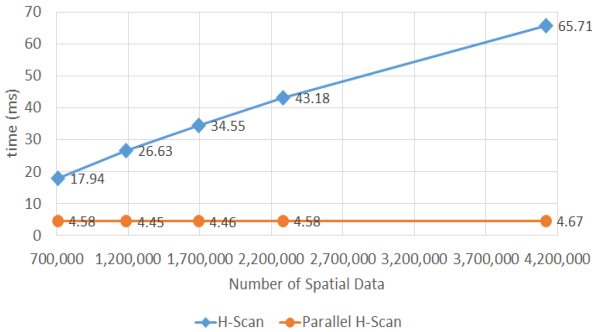


그림 4. 성능 실험 결과

Fig. 4. Results of performance experiments

표 2. 실험 결과 (시간, ms)

Table 2. Experimental results (time, ms)

Grid Cells	West-1		West-4		West-7		West-11		West-17	
	CPU	GPU	CPU	GPU	CPU	GPU	CPU	GPU	CPU	GPU
20x20	6.49	0.46	10.79	0.61	15.13	0.75	20.44	0.97	36.85	1.67
100x100	7.32	0.43	11.98	0.47	16.63	0.53	22.11	0.62	38.83	0.86
500x500	11.90	1.43	19.03	1.41	24.55	1.45	31.12	1.50	50.19	1.60
1000x1000	17.94	4.58	26.63	4.45	34.55	4.46	43.18	4.58	65.71	4.67

V. 결 론

본 논문에서는 기존의 공간 클러스터링 알고리즘을 병렬로 처리하는 방법을 제안하였다. 이를 위해 기존의 해시 기반 클러스터링 방법인 H-Scan에서 Grid를 구축하는 단계를 GPGPU를 사용하여 병렬로 구현하고 실제 사용되는 공간 데이터 셋을 기반으로 실험을 수행하여 기존 방법에 비해 평균 19배 향상되었음을 측정하였다. 본 논문에서 제안한 병렬 H-Scan 알고리즘은 Point, Polyline, Polygon으로 구성된 공간 데이터를 처리할 수 있고, 복잡한 모양의 클러스터도 발견할 수 있는 장점이 있다.

그러나 발견된 클러스터들의 크기와 모양이 클러스터마다 크게 차이날 수 있으므로, 모양이 비슷하거나 크기에 제한이 필요한 클러스터를 찾는 응용을 위해서는 제안된 알고리즘을 확장하여 k-means 등의 다른 알고리즘과 병합하는 연구가 향후 연구로 가능하다.

References

[1] D. Xu and Y. Tian, "A comprehensive survey of clustering algorithms", Annals of Data Science (AODS), Vol. 2, No. 2, pp. 165-193, Jun. 2015.

[2] J. Mennis and D. Guo, "Spatial data mining and geographic knowledge discovery-An introduction", Computers, Environment and Urban Systems, Vol. 33, No. 6, pp. 403-408, Nov. 2009.

[3] G. Sheikholeslami, S. Chatterjee, and A. Zhang, "Wavecluster: A multiresolution clustering approach for very large spatial databases", in Proc. of the 24th VLDB Conf., Vol. 98, pp. 428-439, Aug. 1998.

[4] K. S. Chang, Y. W. Peng, and W. M. Chen, "Density-based clustering algorithm for GPGPU computing", IEEE Int. Conf. on Applied System Innovation, pp. 774-777, May 2017.

[5] G. R. L. Silva, R. R. D. Medeiros, B. R. A. Jaimes, C. C. Takahashi, D. A. G. Vieira, and A. D. P. Braga, "CUDA-Based Parallelization of Power Iteration Clustering for Large Datasets", IEEE Access, Vol. 5, pp. 27263-27271, Oct. 2017.

[6] M. Li, J. Huang, and J. Wang, "Paralleled fast search and find of density peaks clustering algorithm on GPUs with CUDA", 17th IEEE/ACIS Int. Conf. on SNPD, pp. 313-318, May 2016.

[7] E. Zhou, S. Mao, M. Li, and Z. Sun, "PAM spatial clustering algorithm research based on CUDA", 24th Int. Conf. on Geoinformatics pp. 1-7, Aug. 2016.

[8] J. Zhang, S. You, and L. Gruenwald, "Large-scale spatial data processing on GPUs and GPU-accelerated clusters", Sigspatial Special, Vol. 6, No. 3, pp. 27-34, Nov. 2014.

[9] R. Wu, B. Zhang, and M. Hsu, "Clustering billions of data points using GPUs", Proc. of ACM UCHPC-MAW, pp. 1-6, May 2009.

[10] M. Ester, H.P. Kriegel, J. Sander, and X. Xu, "A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise", Proc. of 2nd Int. Conf. on KDD, pp. 226-231, Aug. 1996.

[11] B. W. Oh and K. J. Han, "H-SCAN: A Hash-based Spatial Clustering Algorithm for Knowledge Extraction", Journal of KISS (B): Software and Applications, Vol. 26, No. 7, pp. 857-869, Jul. 1999.

[12] TIGER/Line Shapefiles and TIGER/Line Files, https://www.census.gov/geo/maps-data/data/tiger-line.html. [accessed: Mar. 08, 2019]

저자소개

오 병 우 (Byoung-Woo Oh)

1999년 2월 : 건국대학교 컴퓨터공학과(공학박사)
1999년 ~ 2004년 : ETRI 선임연구원
2004년 ~ 현재 : 금오공과대학교 컴퓨터공학과 교수
관심분야 : 공간 데이터베이스, 모바일 소프트웨어, CUDA 병렬 처리