Journal of The Korea Society of Computer and Information
Vol. 19, No. 9, September 2014
www.ksci.re.kr
http://dx.doi.org/10.9708/jksci.2014.19.9.001

대용량 LiDAR 데이터 보간을 위한 MPI 격자처리 과정의 작업량 발란싱 기법

김 선 영*, 이 희 진**, 박 승 규*, 오 상 윤*

Task Balancing Scheme of MPI Gridding for Large-scale LiDAR Data Interpolation

Seon-Young Kim*, Hee-Zin Lee**, Seung-Kyu Park*, Sang-Yoon Oh*

요 약

본 논문은 MPI를 이용하여 LiDAR 데이터를 처리하는 방식에서 각 코어간의 통신을 최소화하고 작업량 발란싱을 위해 격자크기를 다양하게 하여 LiDAR 데이터의 보간 처리 성능을 향상시키는 기법을 제안한다. 항공기 등을 통해 얻어진 LiDAR 데이터는 3차원 공간정보로서 정밀한 관측 성능과 거리 정보를 포함하여 지리정보, 기상관측 등 다양한 분야에 활용되고 있다. 하지만 필요보다 높은 해상도의 데이터를 사용하거나, 비지표정보를 포함하는 경우를 위해 획득된 LiDAR 데이터를 필터링 하여 사용하게 되며, 필터링된 데이터를 사용하기 위해서는 주변을 탐색할 수 있는 자료구조를 이용해서 보간법을 수행하여야만 데이터가 재구성된다. 데이터의 규모에 비례하여 처리시간도 증가하기 때문에 이를 해결하기 위해 MPI를 이용한 고성능 병렬 처리 방식 연구가 활발히 진행되고 있다. 그러나 기존에 병렬 처리를 사용한 기존의 방식은 각 노드에 할당된 데이터의 밀도가 달라 성능 저하가 생길 수 있으며, 경계값 불일치를 해결하기 위해 노드간의 통신이 많아지는 단점을 가진다. 제안한 방법의 효과를 검증하기 위해 기존 연구에서 제안된 방식들과 처리 성능을 비교하였으며, 데이터에 따라 최대 4.2배의 실행시간 단축되는 것을 확인하였다.

▸ Keywords : LiDAR 데이터, 보간법, MPI, 병렬 처리 방식, 작업량 발란싱

Abstract

In this paper, we propose MPI gridding algorithm of LiDAR data that minimizes the communication between the cores. The LiDAR data collected from aircraft is a 3D spatial information which is used in various applications. Since there are many cases where the LiDAR

•제1저자 : 김선영 •교신저자 : 오상윤
•투고일 : 2014. 7. 9, 심사일 : 2014. 7. 23, 게재확정일 : 2014. 8. 11.
* 아주대학교 컴퓨터공학과(Dept. of Computer Science, Ajou University)
** 캘리포니아대학교 (University of California)

data has too high resolution than actually required or non-surface information is included in the data, filtering the raw LiDAR data is required. In order to use the filtered data, the interpolation using the data structure to search adjacent locations is conducted to reconstruct the data. Since the processing time of LiDAR data is directly proportional to the size of it, there have been many studies on the high performance parallel processing system using MPI. However, previously proposed methods in parallel approach possess possible performance degradations such as imbalanced data size among cores or communication overhead for resolving boundary condition inconsistency. We conduct empirical experiments to verify the effectiveness of our proposed algorithm. The results show that the total execution time of the proposed method decreased up to 4.2 times than that of the conventional method on heterogeneous clusters.

▸ Keywords : ALiDAR, interpolation, MPI, parallel approach, task balancing

I. 서 론

LiDAR (Light Detection and Ranging)는 레이저를 이용하여 지표면 공간정보를 측정하는 GIS 기술이다. LiDAR 데이터, 특별히 지형정보 획득에 많이 사용되는 항공 LiDAR는 항공기(고정익기, 헬기 등의 유인기와 드론과 같은 무인기)에 탑재한 LiDAR 센서를 통해 측정된 거리 정보와 GPS 정보 등을 융합하여 3차원 공간정보를 얻는다. Airborne LiDAR 장비에서 얻어진 LiDAR 데이터는 빌딩, 수목과 같은 모든 정보들을 포함하고 있다. 하지만 일부 분야에서는 필터링을 거친 데이터를 사용하거나 지나친 고해상도를 이유로 기존의 LiDAR 데이터가 아닌 새롭게 재구성된 데이터를 필요로 하는데 데이터를 재구성하기 위해서는 주변을 탐색할 수 있는 구조를 통해 보간법을 수행해야 한다.

LiDAR 데이터의 경우 역거리 가중법을 사용한 보간이 많이 사용되어 왔으나, 보다 정확도가 높은 Kriging 보간법에 대한 요구가 증가하고 있다. 이 Kriging 보간법은 연산량이 많아 이에 대한 대책이 필요하며, 또한 현재 LiDAR 센싱 성능이 높아짐에 따라 획득된 LiDAR 데이터의 규모가 커졌기 때문에 기존의 방식으로는 의미있는 시간 내에 데이터를 처리하기 어려워져 병렬방식의 고성능 처리 방식들이 제안되고 있다.

현재까지 제안된 고성능 처리 방식들 중 과학데이터 처리에 많이 사용되는 MPI 프로그래밍 모델을 응용한 연구들이 현실에 가장 근접한 효과적인 방안을 보여주고 있으나, 이 방식들에서는 격자 처리 과정에서 각 코어별로 처리하는 작업량이 다르다는 문제가 발생하여 전체 LiDAR 데이터 처리 성능을 저하시키는 주요 원인이 되고 있다. 작업량이 다른 원인으로는 처리해야 하는 LiDAR 데이터 밀도가 균일하지 않은 경우와 MPI처리에 사용되는 노드들의 성능이 균일하지 않은 경우가 있다. 불균일한 처리량 문제 외에도 각 코어 간에 보간처리를 위해 필요한 경계값을 교환해야 하기 때문에 발생하는 통신 오버헤드 또한 성능저하의 큰 원인 중 하나이다. 본 논문에서는 MPI를 이용한 대용량 LiDAR 데이터 보간처리에서 각 코어간 처리량을 발란싱시키며, 각 코어간의 통신을 최소화 하여 성능을 향상시키는 기법을 제시한다. 이를 위해 코어마다 다른 bucket 크기를 가지는 자료구조(dynamic bucketing)와 이를 처리하는 알고리즘을 제안하였다.

본 제안 기법의 효과를 증명하기 위해 제안 자료구조와 알고리즘에 기반한 MPI 격자 처리 프로그램을 구현하였고 성능 비교를 위해 기존의 방식 또한 소프트웨어로 구현하였다. 실제 LiDAR 데이터를 입력으로 하여 제안된 서로 다른 크기의 bucket 방식의 효과와 서로 다른 크기의 격자 적용 방식의 효과를 비교 분석하여, 제안된 방식이 기존의 방식에 비해 1.6~4.2배의 성능향상이 있었음을 확인하였다.

본 논문의 구성은 다음과 같다. 2장에서는 LiDAR 데이터의 재구성을 위한 자료구조와 보간법에 대해서 소개하고, 3장에서는 본 논문에서 제안하는 dynamic bucketing을 이용하여 MPI에서의 보간 성능을 향상시키는 기법을 설명한다. 그리고 4장에서는 실제 LiDAR 데이터를 입력으로 하여 기존 방식과의 비교 실험 및 그 결과의 분석을 통해 제안한 방법의

성능을 분석하여 제안방법에 대한 평가를 진행하였다. 마지막으로 5장에서 결론을 맺는다.

II. 관련 연구

본 장에서는 데이터를 재구성하기 위해 필요한 자료구조 및 보간법의 종류와 기존의 데이터 재구성에 관한 연구에 대해 알아본다.

1. 탐색을 위한 자료구조

LiDAR 데이터에서 관심 있는 지점의 주위에 있는 데이터들을 찾는 문제는 가까운 이웃(nearest neighbor)을 찾는 문제와 비슷하다. 이 문제는 비슷한 특성을 지니는 데이터들을 인접한 곳에 위치하게 하는 자료구조를 적용함으로써 해결할 수 있는데, 이러한 자료구조로 쿼드 트리(quad tree)[3], k-d tree [4], 버켓팅(bucketing) [1]이 많이 사용되고 있다.

쿼드 트리는 광범위한 공간 데이터를 빠르게 탐색하기 위해 사용되는 자료구조로 자식 노드가 4개인 트리를 의미한다. 하나의 노드는 x축과 y축으로 한 번씩 분할되어 4개의 사분면을 가지고 있을 수 있고 이는 각각 자식노드에 해당한다. 쿼드트리는 루트(root)노드부터 4개의 자식노드로 공간을 분할하는데, 필요하지 않은 부분은 분할하지 않는다. 또 다른 트리 형태인 k-d tree는 이진트리 형태로써 다차원(k-dimension) 공간 데이터를 표현할 수 있다. 데이터는 터미널 노드에만 존재하며 터미널 노드를 제외한 각 노드는 두 개의 자식노드를 가지고 있거나 공간을 나누는 서브파일(subfile)을 가지고 있다. 공간을 분할하기 위해 레벨에 따라 각 차원을 번갈아가며 이용하는데 예를 들어 3차원 데이터를 분할 할 경우 레벨 1에서 x축으로 나누었다면 레벨 2에서는 y축, 레벨 3에서는 z축을 기준으로 분할한다.

위의 자료구조들과는 다르게 배열 형태의 자료구조인 버켓팅은 버켓(bucket)이라 불리는 메모리 공간에 LiDAR 데이터를 위치시키는 것으로서 자료를 배치함에 있어 최소한의 bucket 수를 갖는 것을 목표로 한다 [12]. 버켓팅을 통해 n-차원의 배열에 데이터가 위치하게 되면 배열의 위치를 기준으로 주변의 데이터를 쉽게 찾을 수 있다.

2. 보간법(Interpolation)

데이터 탐색을 위한 자료구조가 생성되면 이 구조를 토대로 보간을 수행한다. LiDAR 데이터의 보간에는 역거리 가중법(Inverse Distance Weight: IDW)과 크리깅(Kriging)[10] 알고리즘을 주로 사용하는데, 역거리 가중법은 기준 점과의 거리에 따라 가중치를 달리 하는 것으로써 가중치는 떨어진 거리 제곱에 반비례하게 된다. 이 보간법은 비교적 연산이 단순하여 실행 시간은 빠르지만 정확도는 다른 보간법에 비해 다소 낮다.

Kriging 알고리즘은 이와는 반대로 정확성은 높지만 복잡한 연산과정 때문에 보간을 수행하는 시간이 오래 걸리는 보간법이다. Kriging 알고리즘에서 보간은 관심 있는 지점의 값을 예측하기 위해 주변 값들의 데이터들의 선형조합을 이용하게 되며, 기준점과 주변 데이터가 떨어져 있는 거리에 따라 가중치를 다르게 부여하고 이를 구하기 위해 반베리오그램(Semi-variogram)을 사용한다.

반베리오그램은 통상 베리오그램으로도 불리며 자료의 유사성을 나타내는 척도로서 가까울수록 값이 작게 나오므로 [2] 거리에 따라 가중치를 다르게 부여할 수 있다. 일반적으로 베리오그램은 거리에 비례하여 값이 증가하지만 어느 순간부터는 거리에 상관없이 값이 일정해진다. 이때 유지되는 베리오그램 값을 문턱값(sill)이라고 하며 값이 일정해지는 거리를 상관거리(range)라고 한다. 또한, 데이터에 따라 떨어진 거리가 0이라도 베리오그램 값을 나타내는 경우가 있는 데 이를 너깃(nugget)이라 한다. 이 값들은 데이터 특성을 나타내는 모델링 식에 사용된다. 모델링 식은 Kriging알고리즘에 사용되는데 알고리즘을 사용하기 위해서는 식에 사용될 점들을 샘플링해야 한다.

데이터를 샘플링하는 방법으로는 데이터 전체를 샘플링하는 [10]방법과 주변의 가까운 데이터만을 샘플링하는 방법이 있다. Kriging은 샘플링된 데이터를 이용하여 보간을 수행하는데 샘플링된 데이터 수에 따라 O(n^3)에 비례하여 실행시간이 증가한다. 데이터 크기가 증가할수록 실행 시간 역시 기하급수적으로 증가하기 때문에 병렬화를 통해 데이터를 분배하여 실행시간의 증가폭을 줄여야 한다.

3. 병렬 환경에서 LiDAR 데이터 보간 처리

최근 LiDAR 센서 기술의 발달로 LiDAR 데이터의 크기가 기하급수적으로 커져 데이터를 처리하는 데 있어 어려움을 겪고 있다. 이에 Soo Hee Han [5]은 대용량 LiDAR 데이터를 처리하기 위해 클러스터에서 MPI를 이용하여 Virtual Grid 자료구조를 적용하는 방법으로 성능을 높이는 방법을 제안하였다. 보간에 필요한 데이터를 노드 자신이 가지고 있지 않았을 때 생기는 문제(경계값 불일치)는 MPI호출을 통해 데이터를 전송받음으로써 해결하였다. 하지만 이 연구에서

는 필터링 된 데이터를 고려하지 않았고 각 노드끼리 전송되는 데이터는 1차 보간을 거친 값이므로, 이 제안방법을 공간좌표 형태 [x,y,z]를 필요로 하는 Kriging 알고리즘에 적용하는 것은 적합하지 않다.

Ma Hongchao[8]는 대용량 LiDAR 데이터 처리를 위해 분산 프레임워크를 사용하였으며 경계값 불일치 문제를 proxy server를 이용하여 데이터를 요청하고 직접 전송받는 방법으로 해결하는 방안을 제안하였다. 데이터는 proxy server에서 나뉘어져 각 데이터 서버로 이동된 뒤, 보간에 필요한 데이터 요청이 있을 경우 proxy server는 데이터 서버들에게 요청을 전달하는 방식이다.

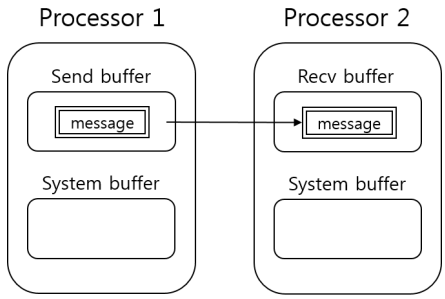
Fei He[6]는 MPI와 OpenMP를 사용하여 작은 크기로 나누어진 대용량 데이터를 마스터가 각 슬레이브에게 분배하는 방법을 제안하였다. Fei가 제안한 방법은 LiDAR 데이터의 필터링 여부에 관계없이 수행할 수 있지만 각 슬레이브에서 IDW보간법을 사용할 때 필요한 데이터의 수가 적을 경우에는 보간을 수행하지 않기 때문에 정확성을 중요시하는 Kriging알고리즘을 사용하기에는 적합하지 않다. 또한 Fang Huang[9]은 오픈소스인 GRASS GIS의 역거리가중법(IDW)을 분석하고 이를 병렬 처리하는 parallel IDW 모듈을 제안하였다. 이 모듈은 포인트 배열의 행을 각 노드에게 배분하여 각 노드에서 데이터를 처리하도록 하게 해준다. 하지만 GRASS GIS에서 Point Array를 이용하여 데이터를 자료구조화를 해주기 때문에 이 부분에서의 성능 향상을 기대할 수 없으며 GRASS GIS에서만 사용이 가능하다.

III. MPI를 이용한 LiDAR 데이터의 보간

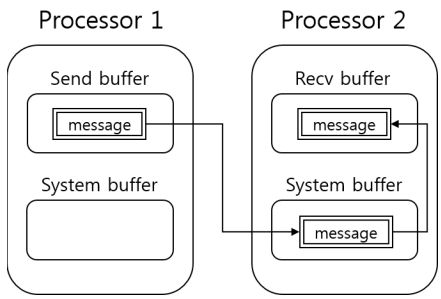
본 장에서는 LiDAR 데이터를 격자구성하기 위해 MPI를 사용하여 처리하는 알고리즘을 제안한다.

1. MPI의 특성

Message Passing Interface(MPI)는 고성능 병렬 프로그램에 사용되는 메시지 패싱의[11]의 전반적인 기능들을 정의해 놓은 표준 라이브러리로 C, C++ 그리고 Fortran 언어에서 사용할 수 있으며 최근 Java 인터페이스가 추가되었고 단일 프로세서 시스템, 멀티프로세서 공유 메모리 시스템, 멀티프로세서 분산 메모리 시스템 등 다양한 하드웨어 플랫폼을 지원한다. MPI에서 데이터는 message passing을 통해 하나의 프로세서에서 다른 프로세서로 이동되며 기본적으로 send함수를 통해 메시지를 전송하고 recv함수를 통해 데이터를 전송받게 된다. 데이터를 전달할 때, 송신 버퍼에서 수신 버퍼로 직접 전달되는 방법(그림 1.(a))과 시스템 버퍼를 거쳐 수신버퍼로 전달되는 방법(그림 1.(b))이 있으며, 그림 1에서는 서로 다른 두 프로세서간의 message passing을 이용한 데이터 이동을 보여준다.



(a) 직접 복사를 사용한 방식



(b) 시스템 버퍼를 사용한 방식

그림 1. MPI에서 데이터 송수신을 위한 메시지 패싱 방식

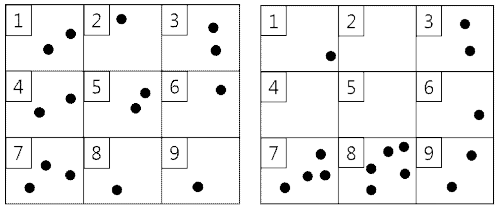
Fig. 1. Message passing method for send/receive data in MPI

MPI는 두 프로세서간의 통신인 점대점(point-to-point) 통신과 일대다 혹은 다대다 프로세서 간에 통신을 할 수 있는 집합(collective) 통신을 지원한다. 통신에서 사용되는 메시지는 송신자 (source), 수신자(destination), 데이터의 크기 및 타입, 꼬리표(tag), 커뮤니케이터(communicator) 등, 메시지 전송 시스템이 메시지를 처리하기 위해 필요한 정보들을 담고 있다. [7].

MPI의 통신은 신뢰성과 편리성을 제공하므로 사용자가 통신을 따로 관리할 필요는 없으나, 프로세서에 할당할 데이터 분배, 불균등한 데이터를 위한 스케줄링 등 병렬화를 위한 작업은 사용자가 직접 처리해야 한다.

2. 제안 기법 알고리즘

분산 메모리 시스템 환경에서 효과적인 LiDAR 데이터의 보간처리를 위해 MPI를 사용하였으며 병렬화에 있어서 고려된 사항은 다음과 같다.



(a) (b)

그림 2. (a) 규칙적인 분포도를 보이는 데이터와 (b) 불규칙적인 분포도를 보이는 데이터

Fig. 2. (a) data of regular distribution and (b) data of irregular distribution

첫째로 필터링 된 LiDAR 데이터의 균일하지 않은 밀도를 갖는 특성과 다양한 성능의 노드로 구성된 (Non-Homogeneous 혹은 Heterogeneous) 클러스터에서 성능 저하의 원인이 되는 노드 간 처리량 불균형 현상을 고려하였다. 균일한 밀도를 보이는 원본 LiDAR 데이터와는 달리 필터링을 거친 LiDAR 데이터는 사용자가 얻고자 하는 데이터의 기준에 부합하지 않는 데이터를 삭제시키기 때문에 데이터의 밀도가 균일하지 않다. 예를 들어, 지형 정보만을 원할 경우에 건물, 자동차와 같은 데이터들은 삭제된다. 일정하지 않은 밀도는 병렬 시스템에서 동일한 크기의 bucket에 데이터를 위치시키는 버켓팅을 수행할 때 문제점이 나타날 수 있는데 각 코어가 동일한 크기의 bucket을 처리할 때 그림 2처럼 불규칙한 분포는 특정 코어에서 더 많은 데이터를 처리하게 하는 문제를 발생시킨다. 그림 2(a)의 코어별 표준편차는 0.7이지만 그림 2(b)의 표준편차는 1.80으로 각 코어에서 처리하는 데이터양의 차이가 그림 2(b)에서 더 높게 나타난다. 동일한 데이터를 가지고 동일 크기의 격자를 처리하는 각 코어에서 데이터 불균형 현상은 전체 시스템의 성능 저하를 야기할 수 있다.

또한, 클러스터를 구성할 때, 구성된 노드들이 모두 동일한 성능을 가지고 있으면 (즉, Homogeneous) 일부 코어에서 작업량 처리에 실패하지 않는 한 처리량은 균등하기 때문에 실행시간은 비슷하다. 하지만 다른 성능을 가진 노드들로 구성되어 있는 클러스터에 작업량을 균등하게 배분한다면 노드간의 실행 시간 차이가 나게 되어 전체적인 성능이 떨어진다. 따라서 다양한 성능을 가진 노드들로 구성되어 있는 클러스터에서는 노드의 성능에 맞게 작업을 분배하여 성능 향상을 도모해야 한다.

두 번째로는 경계값 불일치 현상을 고려하였다. 경계값 불일치 현상은 병렬 환경에서 보간법을 수행할 때 발생하는 현상으로, 주변의 점들을 탐색할 때 인접한 데이터가 위치한 bucket이 다른 코어에 존재하는 경우, message passing과 같은 특별한 접근법 없이는 해당 코어로의 직접적인 접근이 불가능하기 때문에 필요한 데이터를 얻지 못하게 되는 현상이다.

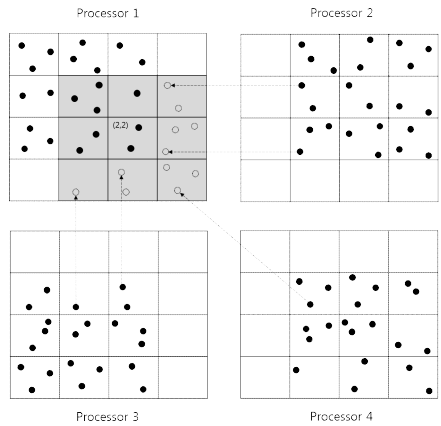


그림 3. 병렬 환경에서 보간을 위해 프로세서 간에 필요한 데이터

Fig. 3. The data necessary for interpolation in parallel environment

보간법을 수행하기 위해서는 주변의 점들을 탐색해야 하는데 순차적인 프로그램과는 달리 그림 3에서 나타난 것처럼 필요한 데이터를 가지고 있는 bucket은 다른 코어에 위치해 있을 수 있기 때문에 message passing 또는 공유 메모리를 통한 메모리 접근 등의 방법을 사용해 필요한 데이터를 전송받아야 한다. 하지만 경계에 위치한 bucket에 있는 데이터가 많을수록 통신 지연시간(latency)는 길어지고 코어의 수에 따라 통신횟수도 증가하기 때문에 네트워크 병목현상(bottleneck)이 발생할 수 있다.

위와 같은 문제들을 해결하기 위해 본 논문에서는 통신을 최소화하여 성능을 향상시킨 새로운 알고리즘을 제안하였으며, 알고리즘의 특성은 다음과 같다: 첫째, 각 코어가 가지는 bucket 크기를 주어진 조건에 맞춰 다양하게 조절한다. 마스터 코어는 bucket 크기를 결정하기 위해 LiDAR 데이터의 최소값, 최대값을 구해 2차원 배열인 bucket의 전체크기를 구한다. 이후 LiDAR 데이터의 필터링 여부에 관계없이 모

든 코어에서 균등한 처리량을 가질 수 있도록 하기 위해서 서버의 성능에 따라 각 서버에게 전송될 행의 크기를 먼저 결정한다. 필터링을 거치지 않은 LiDAR 데이터를 사용하면 단일 서버 내의 코어들의 처리량은 비슷하므로 코어간의 처리량을 고려하지 않아도 되기 때문에 성능을 고려한 크기를 먼저 결정한다. 각 코어에 전송될 행의 크기가 결정되면 이어서 다른 코어에게 할당될 열의 크기도 결정하여 각 코어에게 전송한다. 필터링을 거친 LiDAR 데이터는 성능별 처리량뿐만 아니라 단일 서버의 코어들의 처리량도 다르기 때문에 각 코어에서 수행될 열의 크기도 결정하여 모든 코어의 처리량이 비슷하도록 조절하였다. 이 결과, 동일한 bucket 크기를 사용했을 때보다 실행시간이 짧아짐을 확인할 수 있었다.

또한, 마스터 코어는 각 코어에게 토폴로지 상에서 자신의 위 또는 왼쪽에 위치한 코어가 지닐 행 또는 열의 크기도 함께 전송하는데 이는 각 코어에서 bucket에 위치할 데이터를 찾을 때, 이전의 코어의 크기를 포함하여 찾아야 올바른 데이터가 위치하기 때문이다. 예를 들어, 그림 4는 전체 bucket의 크기를 코어 수만큼 분할한 것으로써 각 격자는 코어를 나타낸다. 마스터 코어가 코어 4에게 행과 열의 크기를 전달하면서 코어 2의 행의 크기와 코어 3의 열의 크기도 같이 전달한다.

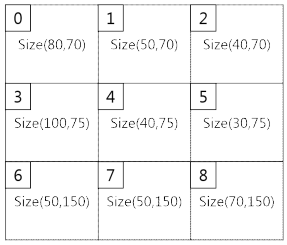


그림 4. 전체 bucket에서 각 코어의 위치 및 각 코어에 할당된 크기
Fig. 4. Locate of each core in total bucket and bucket size in core

전달 받은 값들을 이용하면 코어 4는 x좌표 값이 100과 140(100+40)사이이고 y좌표 값이 70과 145(70+75)사이인 데이터를 찾아 자신의 bucket에 위치시킬 수 있다.

둘째, 각 코어에서 자료구조를 생성할 때, 전송 받은 행과 열의 크기에 탐색범위를 추가하여 생성한다. 버켓팅을 통해 모든 bucket에 데이터를 위치시키지만 보간을 위한 Kriging은 탐색범위가 아닌 bucket에서만 수행된다. 보간을 위해 필요한 점들은 모두 코어 내에 있기 때문에 MPI 통신 없이도 보간을 수행할 수 있다. 통신을 사용해 데이터를 받아오더라도 공간좌표 형태의 데이터를 받아오므로 데이터가 위치할 메모리의 크기는 동일하다. 또한, 버켓팅을 수행하는 데 있어서 O(n)의 비교적 짧은 실행시간이 소요되고 이 실행 시간은 LiDAR 데이터의 크기 n에 비례하기 때문에 bucket의 크기는 실행 시간에 영향을 주지 않는다. 따라서 격자의 크기를 탐색 범위만큼 증가시킨다고 해서 실행 시간이 길어지는 것은 아니므로 성능에 영향을 미치지 않는다. 즉, 단일 코어에 필요한 데이터를 모두 위치시키는 방법을 사용하면 버켓팅에 소요되는 시간은 동일하면서도 통신은 불필요하기 때문에 성능이 향상된다. 특히, 통신 시 소요되는 시간과 데이터양의 증가나 코어의 수가 많아질수록 발생하는 네트워크 병목현상을 고려하지 않아도 된다.

버켓팅을 통해 격자 안에 일치하는 모든 데이터가 위치하게 되면 탐색 범위를 제외한 격자들을 대상으로 탐색 과정을 수행한다. Algorithm 1은 각 코어에서 버켓팅을 수행하는 알고리즘이다. 먼저, 사용자로부터 탐색범위를 입력 받은 후(line 2), 코어가 토폴로지 상의 어느 지점에 위치해 있느냐에 따라 마스터 코어에게 전달받은 행과 열의 크기에 탐색범위를 추가한다(line 3-13). 이 크기를 토대로 2차원 배열을 생성한 후, 모든 격자들에 대해 버켓팅을 수행하여 보간을 위한 자료구조를 생성하고(line 15-19) 이를 기반으로 탐색범위를 제외한 격자들에 대해 보간 알고리즘을 수행한다(line 21-25).

Algorithm 1. bucketing method

```
1 Function Bucketing(row_ gridsize, col_ gridsize)
2  SET grid_radius to user's defined value
3  IF the quotient of rank divided by four is zero or
three THEN
4    ADD grid_radius to row_gridsize
5  ELSE
6    ADD double grid_radius to row_gridsize
7  END IF
8
9  IF the remainder of rank divided by four is zero
or three THEN
10   ADD grid_radius to col_gridsize
11 ELSE
12   ADD double grid_radius to col_gridsize
13 END IF
14
15  MAKE Grid_array which size is row_gridsize *
col_gridsize
16
17 FOR all of LiDAR point are mapped GridPoint
18   Found point of matching GridPoint
19 ENDFOR
20
21 FOR all grid of Grid_array are searched
22   FOR found around GridPoint by grid_radius
23     Calculate grid point using Point set GridPoint
24   ENDFOR
25 ENDFOR
```

IV. 성능비교 및 평가

1. 실험 환경

본 절에서는 제안된 LiDAR 데이터 보간을 위한 알고리즘의 성능 평가를 위해 표 1에 제시된 실험환경에서 실제 LiDAR 데이터를 이용해 실험을 진행하였다.

실험은 LiDAR 데이터의 크기와 탐색 범위를 달리하여 진행하였으며, 격자의 크기는 1m x 1m로 고정하였다. 입력 LiDAR 데이터로는 100만개와 400만개의 점들을 가지고 있는 두 개의 데이터를 실험에 사용하였고 탐색 범위는 3에서 5로 변화를 주었다. 실험에 사용된 LiDAR 데이터의 자세한 사항은 표 2에 나와 있으며 그림 5는 실험에 사용된 필터링을 거친 Data 2의 이미지를 보여준다.

표 1. PC Cluster 실험 환경
Table 1. Experimental environment

	Node 1	Node 2	Node 3
CPU	xeon X3430	xeon E5606	xeon E3-1220
	2.40GHz	2.13GHz	3.1GHz
	quad-core	2x quad_core	quad-core
RAM	8GB	8GB	8GB
OS	Centos 6.4	Centos 6.4	Centos 6.4
Network	1 Gbps Ethernet & Switch Hub		
MPI Library	MPICH 3.1		

표 2 . LiDAR 데이터 스캐닝 자료의 정보
Table 2. Scanned LiDAR data information

	Data 1	Data 2
레이저 센서	Optech Gemini	Optech ALTM 3100 EA
대상 지역	Cedar River Watershed, Washington, USA	Arroyo Seco, California, USA
포인트 수	1,186,845	4,517,569
데이터 크기	35MB	133MB

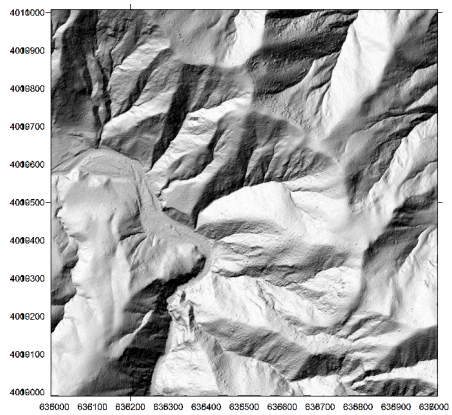


그림 5. 필터링 과정을 거친 Data2의 이미지
Fig. 5. Image of filtered Data2

2. 성능 평가

기존의 LiDAR 데이터 병렬 보간 처리에 관한 많은 연구들은 데이터의 자료구조화를 위해 MPI통신을 사용하였다. 하지만 빈번한 통신의 사용과 데이터 크기에 따른 메시지크기의 증가는 네트워크 병목현상을 일으키는 원인이 될 수 있으므로, 본 논문에서는 통신에 관한 문제점을 해결하기 위해 최소한의 통신을 사용하는 알고리즘을 제안했으며 Soo Hee Han [5]와의 비교를 통해 제안방법에 대해 평가자하고자 한다.

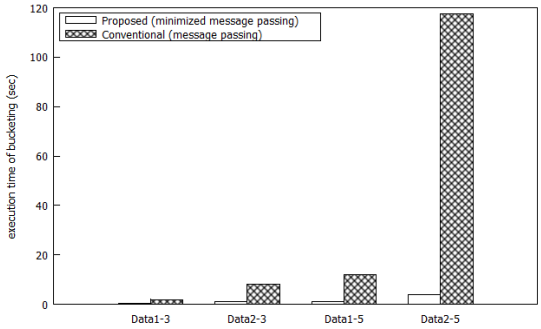


그림 6. 기존의 메시지 패싱을 사용한 자료구조 생성시간과 제안하는 최소한의 메시지 패싱을 사용한 자료구조 생성시간 비교
Fig. 6. Comparison of data structure creation time between conventional and proposed methods

그림 6은 MPI 통신을 일반적으로 사용했을 때(기존)와 최소한으로 사용했을 때(본 논문의 제안)의 자료구조 생성시간을 비교한 것이다. 각 코어의 생성시간 중 가장 오래 걸린 시간을 비교했으며 모든 실험에서 기존 방식인 통신을 사용했을 경우의 생성시간이 더 오래 걸린 것을 볼 수 있는데, 특히

데이터의 크기가 커질수록 또는 전송할 데이터양이 많을수록 생성시간의 차이는 증가한다.

대부분의 코어에서 생성시간은 MPI호출을 사용하지 않았을 때와 큰 차이는 없었지만 버스를 통해 메시지를 전송하는 과정에서 특정 코어에 집중된 데이터의 전달로 인한 병목현상 때문에 전체적인 성능이 저하되었다. 반면에, 통신을 사용하지 않는 경우에는 각 실험에서 코어들은 비슷한 생성시간을 보이는데 이는 버켓팅의 생성시간이 데이터 크기에 비례하기 때문이며 통신을 사용하지 않음으로써 통신에 사용되는 소요시간을 단축시켰기 때문이다. 따라서 다수의 코어들을 사용하여 버켓팅을 수행할 경우에는 최소한의 통신을 이용하는 본 제안방식을 사용할 때 성능을 향상시킬 수 있다.

다음 실험에서는 생성된 자료구조를 병렬처리하기 위해 수행한 기법들의 효과를 분석한다. 실험에 사용된 LiDAR 데이터는 불규칙하게 분포되어 있고 클러스터는 각기 다른 성능을 가지는 노드들로 구성되어 있기 때문에 각 프로세서에 할당될 작업량을 지정해야 한다.

표 3(a)와 표 3(b)는 Data1-3을 이용하여 각 프로세서에서 동일한 격자 크기를 사용했을 때(기존)와 서로 다른 격자 크기를 사용했을 때(제안방식) 실행시간의 표준편차를 나타낸 것으로 여기서 Data1-3은 Data1의 보간을 위해 탐색반경을 3으로 정한 것을 의미한다.

표 3. Data1의 노드별 실행시간 표준편차
Table 3. standard deviation of execution time by node
(a) Data 1-3의 표준편차

	node 1	node 2	node 3
various size (제안방식)	0.388	0.805	0.337
homogeneous size (기존방식)	12.18	8.51	2.03

(b) Data 1-5의 표준편차

	node 1	node 2	node 3
various size (제안방식)	21.72	14.47	21.17
homogeneous size (기존방식)	340.90	197.45	50.73

위의 표는 각 코어가 서로 다른 격자 크기를 가질 때와 동일한 격자 크기를 가질 때의 각 코어 별 실행 시간의 표준편차를 보여준다. 모든 노드에 동일한 격자 크기를 사용했을 때의 표준편차가 큰 것을 알 수 있는데, 이는 실험에 사용한 LiDAR 데이터가 규칙적으로 분포되어 있지 않기 때문이다. 균일하지 않은 밀도를 가진 데이터를 동일한 크기로 분배할 경우, 특정 코어에 데이터가 편중될 수도 있다. 특정 코어에 데이터가 편중되면 실행시간이 길어지고 이는 상대적으로 데이터가 덜 위치한 격자를 처리하는 코어의 기다리는 시간이 많아짐을 의미하기 때문에 성능 향상에 있어 한계점이 생긴다. 그리고 본 실험에서는 다양한 성능으로 구성된 클러스터를 사용했으므로 각 노드의 코어 수 및 cpu 연산 속도를 감안하여 격자의 크기를 결정하였다.

표 4. 탐색범위가 3일 때의 코어별 실행시간 표준편차
Table 4. Standard deviation of execution time by core which search range is 3

	Data1-3	Data2-3
various size (제안방식)	2.36	24.98
homogeneous size (기존)	10.98	289.85

표 4는 각 노드의 성능을 고려하여 실행될 크기를 결정했을 때(제안방식)와 고려하지 않았을 때(기존)의 실행시간 표준편차를 나타낸 것으로 다양한 격자 크기를 사용했을 경우의 표준편차가 동일한 격자 크기를 사용했을 경우의 실행시간 표준편차보다 약 11배정도 작은 것을 알 수 있다. 따라서, 각 노드의 성능과 밀도차에 의한 실행시간의 차이를 줄이기 위해 각 코어마다 서로 다른 격자 크기를 처리하도록 설계하여 처리량이 비슷하게 이루어지도록 하였다. 제안한 기법의 성능을 증명하기 위해 균등한 처리량을 고려하지 않았을 경우와 비교하였다. 그림 7은 최소한의 통신을 사용하는 조건에서 격자크기에 따른 총 실행시간을 비교한 것으로서 다른 격자 크기를 처리할 때 기존의 방식의 실행시간보다 최대 4.2배의 성능향상을 보였다.

서로 다른 크기의 격자처리는 스케줄링과 비슷한 효과를 발휘하여 각 코어의 기다리는 시간을 줄임으로써 성능 향상을 보인다. 이와 같은 실험 결과는 병렬 처리에 있어 데이터 처리량의 균등한 분배가 성능향상에 어떠한 영향을 미치는지를 보여준다.

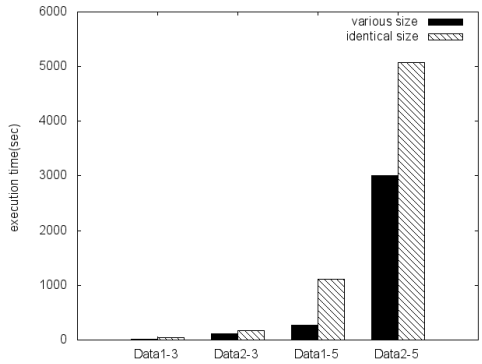


그림 7. 동일한 격자크기를 가질 때(기존방식)와 서로 다른 격자 크기를 가질 때(제안방식) 실행 시간
Fig. 7. Execution time of static grid size and dynamic grid size

V. 결론

본 논문에서는 복잡한 연산처리로 인해 실행시간이 오래 걸리는 Kriging 보간법을 보다 짧은 실행시간으로 수행할 수 있도록 분산 메모리 시스템에서 최소한의 MPI통신을 사용하여 데이터를 보간 처리하는 기법을 제안하였다.

대부분의 LiDAR 데이터는 필터링 과정을 거친 데이터로서 원본 LiDAR 데이터와는 달리 균일하지 않은 밀도를 가지고 있다. 이러한 특성들은 코어간의 처리량 불균형 현상을 초래하여 전체적인 성능 저하의 원인이 될 수 있다. 본 논문에서는 이를 해결하고자 각 코어가 가지는 bucket의 크기를 주어진 조건에 따라 다양하게 조절하였는데, 균일한 밀도를 가진 데이터가 사용되어도 불필요한 작업을 하지 않도록 노드의 성능에 따라 수행될 행의 크기를 먼저 결정한 뒤에 노드 내의 코어에 할당될 열의 크기를 결정하였다. 또한, 병렬 환경에서 데이터를 처리함에 있어서 경계 불일치 현상 및 불필요한 통신 오버헤드의 발생을 방지하기 위해 각 코어에서는 자료구조를 생성할 때 탐색범위를 행과 열에 각각 추가하여 생성하도록 하였다.

제안한 기법을 LiDAR 보간 처리에 적용하고 기존의 방법들과 비교한 결과, 최소한의 메시지 패싱만을 사용한다는 기준으로 인해 알고리즘이 사용하는 네트워크에서는 병목현상이 나타나지 않아 데이터양 또는 이웃한 코어에서 전송받을 bucket의 수가 증가할수록 생성시간의 차이가 증가함을 확인할 수 있었다. 또한, 다양한 크기의 격자처리 적용은 동일한 격자 크기를 적용했을 때보다 최소 1.2배에서 최대 4.2배의 성능 향상을 보였다. 이는 통신 오버헤드 및 코어간의 균등한 처리량은 병렬 처리에 있어 중요한 요소이며 이를 고려하는 것이 효율적인 병렬 처리를 위한 것임을 나타낸다. 따라서 본 논문에서 제안하는 최소한의 통신을 이용한 병렬 처리 기법은 네트워크 대역폭이 작거나 이기종 클러스터에서 보간법을 처리하는 데 유용할 것으로 판단된다.

하지만 이번 연구에서 실험에 사용된 데이터가 병렬 문제점에 대해 어느 측면에서 대표성을 갖는지에 대한 연구가 부족했기 때문에 좀 더 다양한 데이터를 가지고 LiDAR 데이터에서 대표적으로 나타나는 병렬 문제점들을 분석, 연구할 것이다.

VI. 감사의 글

본 연구는 미래창조과학부 및 정보통신산업진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었으며 (NIPA-2014-(H0301-14-1011)), LiDAR 데이터는 NSF-sponsored National Center for Airborne Lase Mapping(NCALM)에 의해 수집되고 제공받았습니다.

참고문헌

[1] Arya Sunil, David M. Mount, and Onuttom Narayan, "Accounting for boundary effects in nearest-neighbor searching," Discrete & Computational Geometry, Vol. 16, No. 2, pp. 155-176, February 1996.

[2] Choe Jonggeun. "Geostatistics," Sigma Press, p.139, 2007.

[3] Finkel Raphael A. and Jon Louis Bentley. "Quad trees a data structure for retrieval on composite keys," Acta informatica, Vol. 4, No. 1 pp.1-9, April 1974.

[4] Friedman Jerome H., Jon Louis Bentley and Raphael Ari Finkel, "An algorithm for finding best matches in logarithmic expected time," ACM Transactions on Mathematical Software, Vol. 3, No. 3, pp.209-226, September 1977.

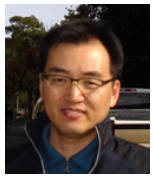
[5] Han S. H., Heo J., Sohn H. G. and Yu K, "Parallel processing method for airborne laser scanning data using a pc cluster and a virtual grid," Sensors, Vol. 9, No. 4, pp.2555-2573, April 2009.

〔6〕 He Fei, Jinyun Fang and Wan Zou, An effective method for interpolation, Geoinformatics, 19th International Conference on. IEEE, pp.1-6, Shanghai, China, June 2011.
〔7〕 MPI Standard Version 3.0, http://www.mpi-forum.org/docs/mpi-3.0/mpi30-report.pdf
〔8〕 Hongchao Ma and Zongyue Wang. "Distributed data organization and parallel data retrieval methods for huge laser scanner point clouds," Computers & Geosciences, Vol. 37, No. 2, pp.193-201, February 2011.
〔9〕 Huang F., Liu D., Tan X., Wang J., Chen Y. and He B., "Explorations of the implementation of a parallel IDW interpolation algorithm in a Linux cluster-based parallel GIS," Computers & Geosciences, Vol. 37, No. 4, pp.426-434, April 2011.
〔10〕 Oliver Margaret, Richard Webster and John Gerrard. "Geostatistics in physical geography. Part I: theory," Transactions of the Institute of British Geographers, Vol. 14, No. 3, pp.259-269, April 1989.
〔11〕 Protopopov Boris V. and Anthony Skjellum., "A multithreaded message passing interface (MPI) architecture: Performance and program issues," Journal of Parallel and Distributed Computing, Vol. 61, No. 4, pp.449-466, April 2001.
〔12〕 Welch Terry A., "*Bounds on information retrieval efficiency in static file structures*," Project MAC, Massachusetts Institute of Technology, pp. 140, 1971.

저 자 소 개

김 선 영
2012: 아주대학교
정보컴퓨터공학과 학사
2012~현재: 아주대학교
컴퓨터공학과 공학석사
관심분야: 분산/병렬처리,
고성능컴퓨팅(HPC),
클라우드 컴퓨팅
Email : k-seon-y@ajou.ac.kr

저 자 소 개



이 희 진
2008: 미 플리다대학교
전기컴퓨터공학 박사
2008~2011: 미 플로리다대학교
박사후연구원
2011~현재: 미 캘리포니아대학교
책임연구원
관심분야: 신호처리,
Lidar Remote Sensing,
영상처리, 패턴인식
Email : heezin.lee@berkeley.edu

박 승 규
1974: 서울대학교 응용수학과 학사
1976: 한국과학원(KAIST)
전산학과 석사
1982: Institut National
Polytechnique de Grenoble
전산학과 박사
1976~1992: KIST, KIET, ETRI
선임/책임연구원
1992~현재: 아주대학교
소프트웨어융합학과 교수
관심분야: 임베디드 컴퓨팅,
차세대 컴퓨터 구조
Email : sparky@ajou.ac.kr

오 상 윤
2006~2007: SK 텔레콤
2008~현재: 아주대학교
컴퓨터 공학과 부교수
관심분야: 고성능컴퓨팅(HPC),
Large Scale Software,
RDF, 클라우드 컴퓨팅
Email : syoh@ajou.ac.kr