The Journal of GIS Association of Korea, Vol. 11, No. 3, pp. 213-226, November 2003

PC 클러스터를 이용한 래스터 GIS 연산의 병렬화

신윤호* · 박수홍**

Parallelization of Raster GIS Operations Using PC Clusters

Yun-Ho Shin* · Soo-Hong Park**

요 약

GIS를 이용한 대용량의 지리정보 처리가 요구되고 있으나 단일 프로세서만으로 복잡한 GIS 연산을 처리하는 데는 능력의 한계성이 대두되고 있다. 특히, GIS 데이터의 증가속도에 프로세서 발전 속도가 미치지 못하고, 증가되는 광범위한 데이터를 처리하는 작업 또한 많은 시간이 걸리는 문제점이 나타나고 있다. 이에 대한 대안으로 계산의 양이 많고 또한, 대용량의 입·출력이 빈번히 일어나는 GIS 연산 작업을 여러 프로세서에 분산시켜 동시에 수행하도록 하는 GIS 작업의 병렬화에 대한 연구가 최근 활발히 진행되고 있다.

본 연구에서는 고가의 병렬 컴퓨터로만 수행되던 병렬 처리를 일반적인 GIS 사용자들이 사용하는 PC 기반으로 MPI(Message Passing Interface)를 사용하여 기존의 단일 프로세서로만 진행되던 래스터 GIS 연산에 대해서 병렬화 과정을 적용하여 연산의 처리 능력을 향상시키고자 한다. 이를 위해, GIS 연산들에 대한 체계적인 분석과 분류를 제시한 Tomlin(1990)의 래스터 GIS 연산을 기준으로 각 연산에 대해 적합한 데이터 분할 기법을 통한 병렬화 과정을 연구하였다.

주요어 : 래스터 GIS, 병렬처리, MPI(Message Passing Interface)

ABSTRACT : With the increasing demand of processing massive geographic data, conventional GISs based on the single processor architecture appear to be problematic. Especially, performing complex GIS operations on the massive geographic data is very time consuming and even impossible. This is due to the processor speed development does not

* 인하대학교 지리정보공학과 대학원
** 인하대학교 지리정보공학과 조교수

keep up with the data volume to be processed. In the field of GIS, this PC clustering is one of the emerging technology for handling massive geographic data effectively.

In this study, a MPI(Message Passing Interface)-based parallel processing approach was conducted to implement the existing raster GIS operations that typically requires massive geographic data sets in order to improve the processing capabilities and performance. Specially for this research, four types of raster GIS operations that Tomlin(1990) has introduced for systematic analysis of raster GIS operation. A data decomposition method was designed and implemented for selected raster GIS operations.

Keywords : Raster GIS, Parallel Processing, MPI(Message Passing Interface)

1. 서 론

1.1 연구배경과 목적

현재 래스터 GIS 연산의 알고리즘은 CPU의 처리속도와 메모리 크기의 비약적인 증가에도 불구하고 증가하는 데이터 양에 비해서 처리성능의 한계점을 드러내고 있다. 이는 특히 래스터 GIS 연산을 기반으로 하는 광역적 혹은 범지구적 범위의 공간 모델링 혹은 시뮬레이션 등의 작업시에 고가의 시스템 사양을 요구하게 된다. 이러한 요구사항을 기존의 컴퓨팅 환경에서 효율적으로 처리하기 위해서 GIS 분야에서도 병렬처리기법의 도입과 다양한 관련 알고리즘 개발에 관한 연구들이 수행되고 있다. 하지만, 선행 연구들은 주로 고가의 massively parallel한 컴퓨터 환경을 벗어나지 못하였으며 특히, GIS 연산들의 특성들을 반영하는 병렬화된 GIS 소프트웨어 개발을 위한 기본적인 알고리즘 개발보다는 특정 활용분야에 대한 임시적인 병렬화 접근방법을 취해왔다. 그리고 개발환경 자체가 모두 고가의 장비를 필요로 하고 있어 병렬화된 GIS 연산들이 다양하고 실제적인 목적에 활용되기에는 많은 어려움이 있었다.

최근 PC를 이용한 클러스터링 방법이 소개되면서 저렴한 구축비용으로 수퍼컴퓨터에 버금가는 연산처리능력을 활용할 수 있게 되었고 또한 표준으로 자리 잡은 메시지 패싱 방법에 기반한 MPICH, LAM 등 여러 개발도구가 제공되면서 이를 통한 GIS 연산의 병렬화가 논의되기 시작하고 있다.

본 연구의 목적은 Tomlin(1990)이 제시한 기본적인 래스터 연산(map algebra)들의 특성에 적합한 병렬화된 알고리즘을 개발하고 그러한 병렬처리 기법들을 일반 PC 환경에서 가장 효율적인 GIS 연산의 병렬화 알고리즘을 적용하여 그 성능 평가에 목적이 있다.

1.2 국내외 연구사례

초기 선행 연구들은 주로 병렬 연구가 시작되었던 고가의 massively parallel한 컴

퓨터 환경을 이용하기 때문에 병렬화 알고리즘의 개발에 한정되어 왔으며, 대부분 영상처리 관련 분야에서 연구되었다(Ioannis Pitas, 1993). 또한, 선행 연구들은 GIS 연산들의 특성들을 체계적으로 분석하여 병렬화된 GIS 소프트웨어 개발을 위한 기본적인 알고리즘 개발보다는 특정 활용분야 예를 들어 가시권 분석(David B. Kidner, Philip J. Rallings, 1997), 내삽(M.P. Armstrong and R. Marciano, 1994), 토지이용 변화 모델링(Hazen, B.C and M.W Berry, 1997), 도시 교통량 분석(Xiong, D. and D.F Marble) 등에 한정되어 왔다.

기존의 병렬 컴퓨터 아키텍쳐의 경우 병렬 프로그래밍이 필요 없는 공유 메모리 방식(하나의 메모리에 여러 프로세서가 연결되어 있는 경우)으로 연산 처리속도는 빠르지만 확장성에 어려움이 있고 고가의 비용을 요구함으로서 GIS 일반 사용자에게는 활용이 어려웠다. 하지만 최근에는 PC를 이용한 클러스터 기법들이 개발되어 표준 메시지 기반에 병렬 프로그래밍 기법들이 연구되고 있다.

이러한 메시지 기반 병렬처리 기법 중에도 GIS 분야에서 이루어지고 있는 연산들은 벡터 데이터 연산들에 대부분 집중되고 있으며 래스터 데이터 연산에서는 아직 고가의 병렬 컴퓨터를 이용한 처리시간을 향상시키는 방법이 사용되고 있다(Thomas B, Hans-Peter K, Bernhard S, 1996). 그 외에 PC에서 사용하는 경우는 아직 데이터 크기와 입력, 출력 처리시간의 상대적인 증가 때문에 많은 진전이 이루어지지 않았다.

따라서, 본 연구의 래스터 연산에서는 우선 각 프로세서 수에 따라 연산에 적합한 분할정책을 결정한 후 각 프로세서가 분할된 데이터와 상호 통신하는 시간을 줄이는 데이터 분할방식을 이용하였다. 즉, 각 프로세서에서 처리해야 하는 데이터 양과 전송처리 시간을 줄이는 방법으로 프로세싱 시간을 줄이는 방법이 연구되었다.

1.3 연구내용과 방법

래스터 GIS 연산에 적합한 병렬화 작업을 수행하기 위해서는 우선 래스터 GIS가 가지고 있는 기능들을 유형별로 분류하고 각 유형의 특성이 정의되어야 한다. 이를 위해서 Tomlin이 제시한 래스터 연산의 분류체계(local, focal(incremental), and zonal operations)에 따라 GIS 연산을 유형별로 정의 및 분류하고 각 유형의 기본적인 연산 특성(알고리즘 특성)을 분석하고자 한다. 실제로 이 연구에서 수행할 병렬화 대상 연산들은 각 유형별로 기본적이고 활용도 및 중요도가 높은 대표적인 연산들을 GIS 관련 문헌과 공간 모델링 혹은 시뮬레이션 사례를 참고한다. 선정된 대표적인 GIS 연산들의 실제 실행속도를 다양한 크기의 실제 데이터 혹은 가상 데이터를 사용하여 측정하고 이 중에서 병렬화가 가능한 연산들을 최종적으로 선별하여 이후의 병렬화 대상 연산으로 삼는다.

성능분석은 순차적 작업 수행과 병렬작업 알고리즘의 수행시간 비교를 통해 이루어지는데, 구체적으로는 각 연산에서 요구되는 다양한 크기의 데이터를 사용하여 데이터의 크기와 클러스터에 사용되는

노드수의 변화에 따른 수행시간을 측정하여 전반적인 병렬화 알고리즘의 특징을 파악하고자 한다. 또한 PC 환경에서 사용이 가능한 상용의 GIS 소프트웨어(예를 들어 Idrisi)에서 동일한 데이터를 사용하여 연산의 수행시간을 측정하여 상대적인 성능비교 결과를 제시하고자 한다.

시스템 구성은 일반 GIS 사용 환경인 Windows 2000 환경에서 병렬처리 라이브러리인 MPICH를 이용하였다. 각 프로세서 노드 수는 일반적으로 GIS 사용자그룹에서 사용하는 것을 고려할 때 한 부서, 혹은 팀 내에서 존재하는 컴퓨터 수가 8개 이상 정도의 PC가 존재하는 것을 가정하여 1개의 프로세서에서 2개, 4개, 8개까지 프로세서 수를 증가시키며 각 수행시간을 체크하여 성능향상을 확인한다. 각 래스터 연산에서 사용되는 데이터 크기는 일반적으로 사용되는 1024×1024 크기가 많이 사용되므로 래스터 데이터의 크기를 1024×1024(1K)에서부터 최대 6144×6144(6K)까지 테스트한다. 수행될 데이터의 최대크기는 현재 가장 일반적이고 큰 래스터 데이터 자료인 인공위성 자료들이 하나의 영상에서의 데이터 크기가 6000~7000개 이르기 때문에 이에 대한 처리가 가능하도록 고려하였다.

2. 병렬처리 모델과 MPI

2.1 병렬처리 모델

컴퓨팅 처리능력을 증가시키기 위해서는 다양한 방법이 있다. 다수의 프로세서가 하나의 메모리를 공유하는 SMP(Symmetric MultiProcessing) 방식, 다수의 프로세서가 각각 독립된 메모리를 가지고 있는 MPP(Massively Parallel Processing) 방식, 크게 두 가지로 나누어 생각할 수 있다. 그러나 기존 몇몇 벤더에 의해 제공되던 병렬컴퓨터 혹은 슈퍼컴퓨터들은 매우 고가이기 때문에 일반 GIS 사용자들이 쉽게 접할 수가 없다. 최근 PC급에서의 프로세서들의 사양이 높아짐에 따라, 또한 100Mbps이상의 고속네트워크도 널리 보급되어 일반 GIS 사용자 환경에 이용되고 있다. 따라서 단일 컴퓨터들을 네트워크로 연결함으로써 새로운 개념의 병렬컴퓨터를 만드는 것이 가능하게 되었다. 이러한 개념의 병렬컴퓨터를 클러스터라고 하며 PC를 이용한 병렬 컴퓨터를 PC 클러스터라고 한다.

일반적으로 메시지 전달 프로그래밍 모델에서는 각각의 프로세서가 독자적인 메모리 공간을 가지게 된다. 즉 앞서 설명한 분산 메모리방식의 병렬 처리 방식으로 하나의 문제 해결 혹은 작업을 수행하기 위해서 프로세서들은 정보(메시지)들을 교환하게 된다. 이러한 메시지에는 다음과 같은 정보들을 포함하고 있다.

- 메시지를 전송하는 프로세서에 대한 정보
- 보내는 데이터의 타입, 길이(크기)
- 메시지를 수신하는 프로세서 정보
- 메시지를 수신하는 위치, 수신하는 메시지의 크기 등.

2.2 MPI

MPI는 MPI 포럼(병렬처리 연구기관, 학

회, 컴퓨터 하드웨어, 소프트웨어를 만드는 관련 기관들이 참여, 1990)에 의해서 처음으로 만들어진 메시지기반의 병렬처리 프로그램을 위한 표준 규약으로써, 메시지 전달 형태의 병렬 프로그램의 호환성, 그리고 효율성, 기능성 등을 충족시키기 위한 목적으로 표준 기법을 제공하고 있다. MPI는 프로그램 및 라이브러리 호환성, 동일한 프로그램 인터페이스 제공, 이기종간의 연결 기능, 효율적인 통신 기법 제공의 특징을 가지고 있다. 이러한 특징은 최종 사용자와 라이브러리 개발자에게도 필요한 기능들이다. 본 논문에서 사용되는 MPI 라이브러리는 미국의 Argonne 국립 연구소가 공개한 MPICH 소프트웨어 패키지를 사용하였다.

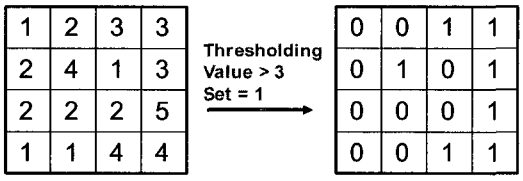
3. 래스터 GIS 연산과 데이터 분할

3.1 래스터 연산

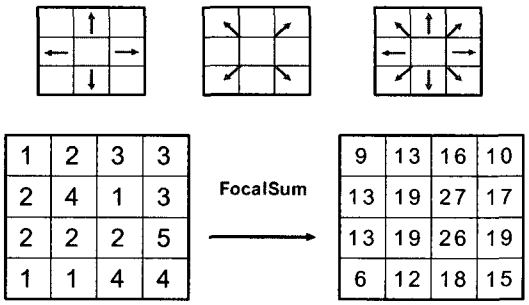
래스터 GIS의 연산을 분류하고 정의하는데 널리 사용되는 접근방법으로는 지도 모델링을 들 수 있다(Tomlim, 1990). 이 분류체계는 래스터 GIS가 아니지만 래스터 GIS의 연산들을 정의하는데 자주 언급되며 사용되고, 대부분의 래스터 GIS 연산들은 지도 모델링 방법론에서 소개된 연산들에 따르고 있다.

첫 번째로 국지연산(local operator)는 단일 혹은 복수 레이어에 있는 개개의 셀에 부여되는 연산으로 각 셀의 값을 처리하여 새로운 레이어의 동일한 위치에 그 값을 출력한다. 국지연산은 사용되는 레이어의 수에 따라 두 유형(단일 레이어 국지연산과 복수 레이어 국지연산)으로 나눌수 있다. 단일 레이어에 적용되는 국지연산에는 재분류(local reclassification), 수학함수, 그리고 상수 함수(scalar function)등이 있다.

그 다음 근린연산(focal operator)과 증가연산(incremental operation)은 각 셀의 값을 주변(neighborhood)에 있는 셀들에 의해 결정하는 연산들로, 증가연산들은 여기서는 근린 연산의 특별한 경우로 간주한다(Soohong Park, 1996). 근린 연산은 주변 셀의 유형에 따라 크게 두 그룹으로 나뉘는데, 그 첫 번째 그룹은 공간적으로 인접한 주변 셀에 바탕을 둔 근린 연산이다.



[그림 1] 국지연산 : 예) Thresholding

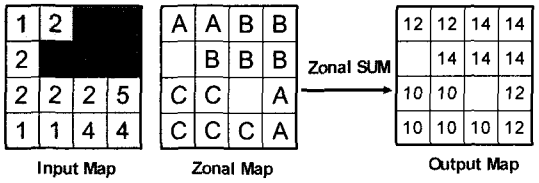


[그림 2] 근린연산 예) : Focal Sum

증가 연산자는 공간적으로 인접한 주변 셀들을 사용하고 기하학적 특성에 의해 결정하기 때문에 이 그룹에 속한다. 두

번째 그룹의 근린 연산은 보다 확장된 주위 셀들을 사용하는 연산들로 주변 셀들이 중심 셀로부터 자유롭게 구성될 수 있다.

구역(zonal)연산은 구역의 형태와 크기로 거리가 아닌 위상학적 관계(연속적인 셀로 구성될 필요가 없는)에 의해 정의되며 다음과 같은 예의 연산형태로 분류될 수 있다.

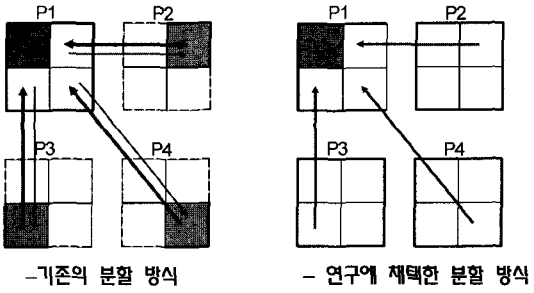


[그림 3] 구역연산, 예) : Zonal Sum

3.2 래스터 데이터 분할

래스터 데이터의 일반적인 특징은 데이터의 포함된 내용보다 데이터의 크기가 프로세싱 시간에 영향을 미친다. 이러한 래스터 데이터를 처리하는 분할방식에도 두 가지 방식이 있는데 하나는 각 프로세서가 처리하는 데이터 크기가 동일한 균등 분할방식과, 각 프로세서 마다 서로 다른 데이터 크기가 처리되는 비균등 분할방식이다. 기존의 래스터 데이터 분할에 대한 연구의 예(M.J.Mineter, G.G.Wilkinson, S.Dowers and R.G. Healy, 1998)는 단순히 래스터 데이터가 가지고 있는 값을 분할하고자 하기 때문에 작업 스케쥴러에 의한 비균등 분할 방식을 이용한 데이터 분할 정책을 수행하였다.

비균할 방식의 방법은 사실 각 프로세서 속도에 대한 정확한 사전정보가 요구되며 실제 병렬 연산 수행시 확인에 어려움 있다. 또한 각 프로세서 수와 데이터 크기 사이에 관련성이 각 연산마다 다르기 때문에 실행시 하나의 최적 매칭방법을 구하기는 어렵다. 또한, 가장 커다란 단점인 래스터 데이터 자체가 지니고 있는 공간상의 위치에 대한 상호참조정보에 대한 고려가 불가능한 비균등 분할방식은 적용해야할 연산활용의 제약으로 래스터 GIS연산을 위한 데이터 분할에는 적합하지 않다. 또한 일반적인 GIS 연산을 수행하는 부서단위, 혹은 팀 단위에서는 비슷한 사양의 시스템을 유지하고 있으며 이런 현 상황을 고려해 볼 때 균등분할 방식을 이용하는 것이 유리하다.



[그림 4] 데이터 분할방식 개념도

이 연구에서는 실제 GIS 연산의 입력, 출력 과정에서 시간이 많이 걸리는 단점을 개선하기 위해서 연산에 따른 데이터 분할 정책이 설정된 후에 각 프로세서가 처리해야 될 데이터 분할 위치에 직접 접근하는 방식을 취함으로써 기존의 병렬처리 방법들보다 입력처리시간이 감소되는 방법을 적용하였다. 많은 선행 방식들이 작업 스케쥴러에 의해 메시지 전송과 반

송의 두 번의 정보교환이 이루어지는 과정을 한번의 메시지 전송처리로 작업을 빠르게 진행하여 전체적인 연산처리 효율성을 높일 수 있다.

3.3 래스터 연산에 따른 분할

데이터 분할은 앞서 설명하였듯이 맨 처음 몇 개의 영역으로 분할 할 것인지는 몇 개의 프로세스를 가지고 프로그램을 수행할 것인지에 관련되어 있다. 즉 사용가능한 수의 프로세스에 따라서 데이터 크기의 분할 정책이 수행되고 각 연산에 조건에 따라 최종 결정된다. 본 연구에서 사용될 래스터 연산은 일반적으로 사용빈도가 높고 각 연산에 특성을 대표하는 각 연산들로써 상수연산과 중첩연산, 그리고 평균 필터연산과 경사도 연산, 마지막으로 버퍼연산을 선정하였다.

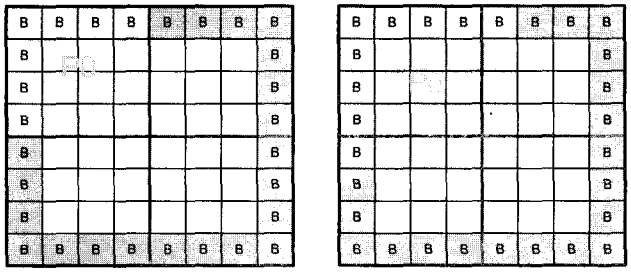
1) 상수연산과 중첩연산

상수연산과 중첩연산은 국지 연산으로 각 연산과정에 대한 특성은 대상 셀에 대해서만 수행이 이루어진다. 이러한 연산의 특성 때문에 연산이 반복되어 수행되어 진다고해도 해당 셀은 그 자체 값에 대해서만 영향을 받게 된다. 따라서 병렬 처리시 해당 프로세서가 가지고 있는 데이터 블록 크기는 분할은 [그림 5]의 좌측과 같다.

2) 평균 필터연산과 경사도 연산, 버퍼 연산

평균 필터연산과 경사도 연산은 근린연산의 특성상 주변의 셀(일종의 버퍼 크기)의 정보가 필요하고 이 정보들은 처음 데이터를 분할 할 때 함께 포함되어 각 프로세서에 할당하게 된다. 이 경우 위의 그림의 오른쪽에서 설명되었듯이 원래 구하고자 하는 데이터 영역보다 필요한 추가 영역의 데이터가 필요하다. 버퍼 연산의 경우는 위의 근린 연산과 동일한 경우로 취급되었다.

연산의 수행 시에는 각 프로세서에 할당되는 데이터 영역에 대한 경계정보, 연산이 수행되는 시작 위치와 마지막 위치, 그리고 전체 데이터에 대한 상대적인 위치가 메시지로 전달된다. 이러한 정보들은 연산 수행 후 최종 결과로 모아질 경우에 참조된다.



버퍼 데이터 공간이 없는 경우 : Local 연산자

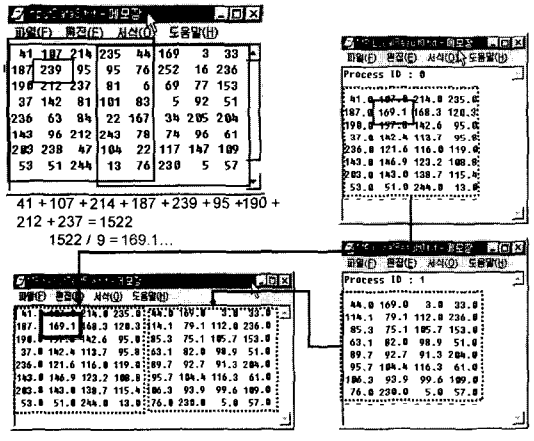
버퍼 데이터 공간이 있는 경우 : Focal 연산자

[그림 5] 연산에 따른 데이터 분할 크기

4. 정확도 및 수행 평가

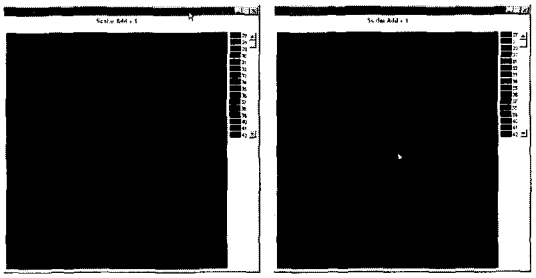
4.1 정확도 분석

우선 병렬 처리 프로그램에서 각 프로세서에서 수행된 결과가 하나의 결과로써 수집되는 결과 값과 동일한 결과 값을 가지고 있는 가를 검증하기 위해서 우선 임의의 작은 데이터 8×8 크기의 데이터를 가지고 각 프로세서에서 수행한 후 결과 값을 출력하였다. 다음 그림에서 근린 연산의 경우에는 데이터 분할 정책시 데이터 분할이 4×8로 정확히 나누어지는 것이 아니라 좌측 상단의 그림과 같이 5×8의 크기로 근린 연산에 필요한 버퍼 크기를 가지고 분할을 하게 된다. 위 예제에서 2행 2열에 존재하는 239의 값은 데이터 분할정책에 의해 첫 번째 프로세서에 의해 주변의 값(3×3) 필터에 의해서 169.1의 값으로 계산된 후(좌) 최종결과에 2행2열에 저장된다.

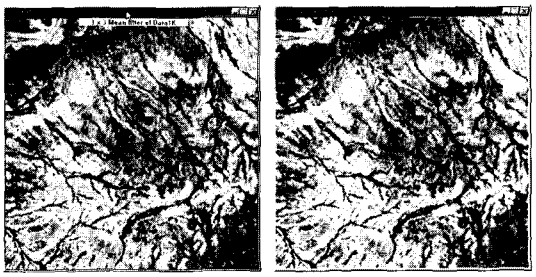


[그림 6] 병렬 처리 결과 정확성 검증
(예, 평균필터 연산)

또한 테스트의 데이터 형태는 binary 형태로 입력, 저장되며 GIS 프로그램인 Idrisi에서 처리된 데이터와 아래와 같이 비교하면 동일한 결과임을 확인할 수 있다.



[그림 7] Idrisi & 병렬처리(상수 연산)



[그림 8] Idrisi & 병렬처리(평균필터 연산)

4.2 Idrisi 프로그램 수행 성과 분석

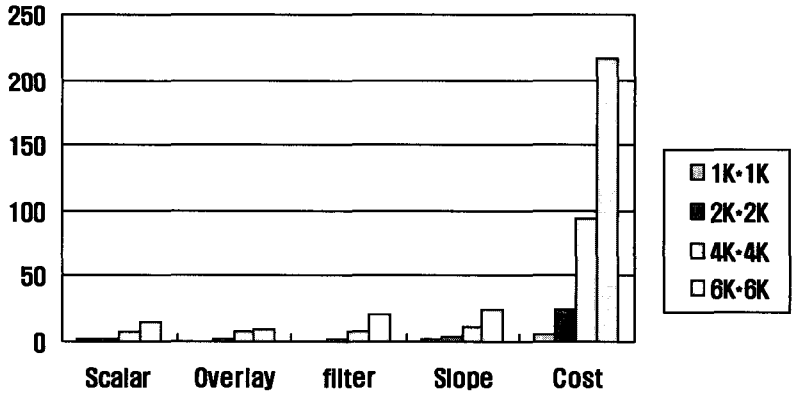
Idrisi 프로그램에서는 Visual Basic Script를 이용하여 선정된 연산들이 어떤 알고리즘을 통해 수행되고 또 Idrisi에서 어느 정도의 연산 시간이 소요되는지 연산 시간을 각 데이터 사이즈별로 증가시키면서 테스트하였다.

Idrisi에서 수행된 결과에서의 특징은 국지 연산자인 상수 연산자와 중첩 연산시간이 데이터 크기가 증가함에 따라 연산의 속도도 선형적으로 증가한다는 것이다. 이런 결과는 데이터 증가에 따른 시간 증가

<표 1> Idrisi 프로그램에서의 연산 수행시간

time(sec)

	1024×1024	2048×2048	3072×3072	4096×4096	5120×5120	6144×6144
Scalar Operation	1.109375	2.583008	4.675781	7.880859	12.05664	15.12109
Overlay Operation	0.5507813	1.592773	3.425781	6.760742	10.33398	14.33203
Filter Operation	0.671875	1.993164	4.816406	8.020508	12.33789	20.19922
Slope Operation	0.953125	2.844727	6.679688	10.60645	17.78516	23.73438
Buffer Operation	35.1875	132.7109	281.9648	525.0654	791.2285	1167.85
Cost Distance Operation	6.308594	24.66602	56.00195	94.15625	148.082	216.7129



[그림 9] Idrisi에서의 래스터 연산 결과 그래프

가 거의 데이터 사이즈의 배수와 일치하는 것으로 확인 할 수 있다. 상대적으로 국지 연산자들보다 근린 연산자의 연산이 시간이 많이 걸리는 것은 하나의 셀이 가지고 있는 연산의 수가 많이 존재하기 때문이다.

평균 필터 연산과 경사도 연산시에는 평균필터 연산이 하나의 셀에 대해 매번 8개의 셀에 대한 정보를 가지고 수행하기 때문에 좌우 6개 셀에 대해 수행되는 경사도 연산보다 빠를 수 있다고 예상할 수도 있지만 Idrisi 프로그램에서는 경사도 연산시에 연산 비용이 높은 삼각함수 연산을 수행하기 때문에 부동소수점 연산보다 상대적으로 많은 연산비용을 가지게 되어 경사도 연산의 수행시간이 더 증가되는 것을 확인할 수 있다.

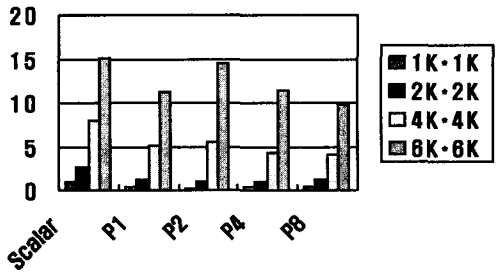
4.3 MPI 병렬 프로그램 수행 성과 분석

병렬프로그램을 이용한 국지연산의 경우에는 연산의 처리 과정이 단순한 사칙연산이기 때문에 프로세서의 수를 증가시키며 데이터를 분할하더라도 얻는 성능의 향상은 그렇게 크기 않은 것으로 나타났다.

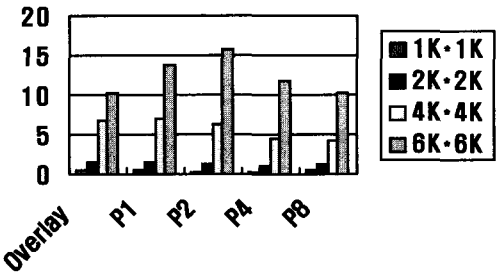
<표 2> MPI 병렬 프로그램에서의 국지 연산 수행시간

time(sec)

Operation	Pro.\ Size	1K	2K	3K	4K	5K	6K
Scalar	No 1	0.319	1.230	2.830	5.110	7.937	11.210
	No 2	0.267	1.100	2.600	5.470	8.287	14.551
	No 4	0.361	1.046	2.371	4.333	7.392	11.357
	No 8	0.369	1.139	2.293	4.083	6.868	9.878
Overlay	No 1	0.409	1.534	3.444	7.060	9.484	13.731
	No 2	0.314	1.315	3,084	6.304	9.106	15.755
	No 4	0.349	1.120	2.508	4.563	7.832	11.860
	No 8	0.421	1.145	2.382	4.196	6.907	10.290



[그림 10] 병렬처리에서의 상수 연산시간



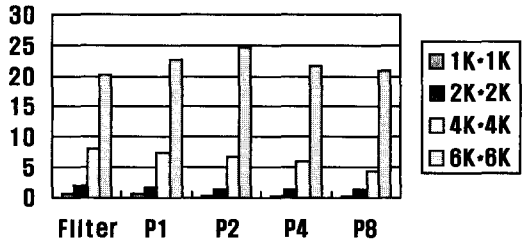
[그림 11] 병렬처리에서의 중첩 연산시간

국지 연산보다 연산비용이 많은 근린 연산의 경우에서는 어느 정도 프로세서의 수가 증가되면서 그 효율이 나타나기도 한다. 근린 연산 중에서 시간이 많이 걸렸던 경사도 연산의 경우에서 보듯이 프로세서를 증가시키면서 그 효율성이 증가되어 50%정도의 성능향상을 보이기도 한다.

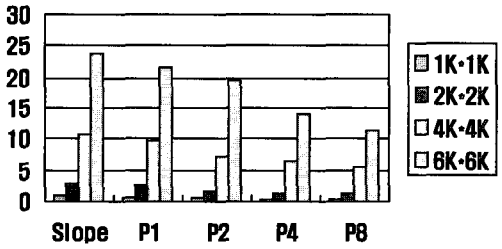
<표 3> MPI 병렬 프로그램에서의 근린 연산 수행시간

time(sec)

Operation	Pro.\ Size	1K	2K	3K	4K	5K	6K
Filter	No 1	0.509	1.819	4.122	7.265	11.168	22.689
	No 2	0.390	1.366	3.178	6.699	9.917	24.798
	No 4	0.397	1.251	2.702	5.903	8.475	21.776
	No 8	0.390	1.192	2.513	4.476	7.091	20.974
Slope	No 1	0.701	2.552	6.302	9.788	18.426	21.666
	No 2	0.498	1.754	3.971	7.330	13.794	19.502
	No 4	0.460	1.349	3.110	6.513	10.186	13.975
	No 8	0.459	1.261	2.671	5.506	7.591	11.351



[그림 12] 병렬처리에서의 평균필터 연산시간



[그림 13] 병렬처리에서의 경사도 연산시간

각 연산의 결과에서 특이한 경우는 데이터 크기가 작은 경우(여기서는 1K 정도의 데이터)이다. 하나의 프로세서에서 처리하는 속도가 오히려 데이터를 분할하여 다시 하나의 데이터로 모으는 시간보다 적게 수행될 수 있기 때문에 처리효과가 더 떨어질 수 있다. 하나의 CPU가 수행하는 속도는 네트워크에서 전송하는 속도보다 훨씬 빠르기 때문이다. 대부분의 1K에서의 처리 비용은 2개의 프로세서의 증가시키면 오히려 수행시간이 길어지는 것을 결과로 확인 할 수 있다. 그리고 데이터가 너무 큰 경우에도 처리 결과를 되돌려주는 시간이 네트워크를 통해서 전해지기 때문에 시간이 증대된다. 이런 예는 평균필터연산의 경우에 보듯이 5K까지는 처리 효율이 증대되지만 6K에서부터는 프로세서의 수를 증가시켜도 처리 효율이 더 이상 높아지지 않고 멈춰 있는 것으로 나타난 경우의 예이다.

버퍼 연산의 경우에는 수행된 테스트 연산중에서 해당 셀에 2번의 Pass Step을 거쳐서 생성하는 연산이다. 이 반복된 연산으로 병렬 처리시간 효율성이 프로세서

를 증가시키면서 단순한 연산이었지만 50%정도까지의 좋은 효율성을 보이고 있다. 버퍼연산의 경우에는 기존의 GIS 프로그램에서 제공되는 알고리즘을 이용하여 동일한 결과의 버퍼를 생성하는데 상대적인 시간 차이가 크게 나타난다.

연구를 통해 밝혀진 일반적인 결과중 하나는 해당 셀이 하나의 단순한 사칙연산으로 이루어져 있을 경우에 성능향상이 적고 또 데이터의 크기가 작을 경우(1K)에는 그 성능향상 역시 미미한 것으로 나타났다.

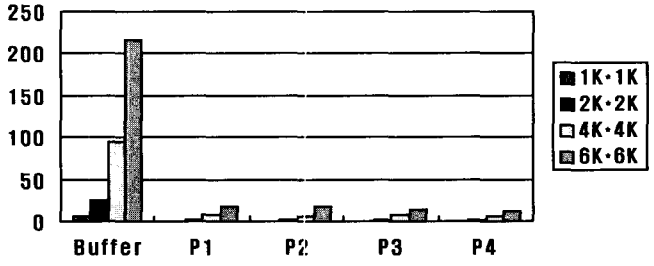
5. 결 론

현재의 이 논문은 일반적인 GIS 사용자들이 이용하는 PC 기반에서 병렬처리로 적용 가능한 GIS 연산들을 분석하고 이러한 연산들을 구현하여 실제 대용량의 1K×1K에서 6K×6K까지의 데이터 크기에 대해서 프로세서 수를 증가시켜 테스트한 후 그 결과를 확인하였다. 일반적으로 많이 사용되는 래스터 연산으로 선정된 연산들 중에 국지 연산에 경우에는 프로세서의 수를 증가시켜도 성능 향상이 미흡했으며 데이터 크기가 커져도 성능 향상이 크기 않았다. 하지만 주변 셀에 의해서 연산을 수행하는 근린 연산자였던 평균필터 연산자와 경사도 연산자의 경우에는 데이터 사이즈가 작은 1K를 제외하고 그 이상의 데이터에 대해서는 보다 높은 처리 성능의 향상을 보였으며 8개의 프로세서를 사용했을 경우 처리 비용을 최대 절반이하로 줄이는 결과를 보여주었

<표 4> MPI 병렬 프로그램에서의 근린 연산 버퍼 수행시간

time(sec)

Operation	Pro.\ Size	1K	2K	3K	4K	5K	6K
Buffer	No P 1	0.511	1.950	4.373	7.786	12.2136	17.141
	No P 2	0.337	1.453	3.283	6.507	10.631	17.518
	No P 4	0.420	1.256	2.723	7.046	8.643	13.270
	No P 8	0.426	1.248	2.496	5.078	7.442	11.069



[그림 14] 병렬처리에서의 버퍼 연산 수행 시간

다. 또한 다른 일반 GIS 프로그램에서 단일 프로세서에 수행되었던 연산과 병렬처리기법을 이용하여 나온 결과 값이 동일함을 확인하였고 그 성능을 비교하였을 경우 버퍼 연산의 경우 그 이상의 처리 성능을 보여주었다. 이 연구에서는 기존의 래스터 데이터가 가지고 있는 공간참조정보를 잃지 않는 데이터 분할방식을 이용하여 기본적인 래스터 연산을 병렬처리 연산으로 재구현 하는데 그 성능향상이 있음을 보여 주었다.

지금까지 이 연구에서는 많은 GIS 래스터 연산 중에 대표적인 연산에 대해서 병렬처리 성능을 평가하였다. 그러나 향후 기존의 순차적으로 단일프로세서에서 수행되었던 다양한 래스터 연산들이 병렬처리로 수행하기 위해서는 여기서 주요하게 적용하였던 데이터 분할만이 아닌 기존의 각 래스터 알고리즘을 기반으로 병렬처리에 적용할 수 있도록 다양한 연구가 필요하다.

이러한 병렬처리로 구현 가능한 래스터 연산들을 이용하면 기존의 도시성장 모델, 적지분석 등 여러 단계 연산을 종합적이고 광범위하게 적용해야 하는 래스터 GIS분야에서는 많은 성능향상 효과를 얻을 수 있을 것이다. 향후 이러한 분야에 병렬연산으로 얻을 수 있는 알고리즘 연구와 성능테스트에 많은 연구가 진행되어져야 할 것이다.

참고문헌

Dana Tomlin, 1990, "Geographic information systems and Cartographic Modeling", Prentice Hall, New York.

David B. Kidner, Philip J. Rallings, 1997, "Parallel Processing for Terrain Analysis in GIS-Visibility as a Case", pp.183-207, GeoInformatica 1:2.

Hazen, B.C and M.W. Berry, 1997, "The simulation to land-cover change using a distributed computing environment", Simulation Practices and Theory, vol. 5, pp. 489-514.

Healey. R, Dowers. B, Gtiiings and M. Mineter, 1998, "Parallel processing algorithm for GIS", Tayor and Francis.

Ioannis Pitas, 1993, "Parallel algorithms for digital image processing", computer vision and neural networks, John wiley & sons.

M.J.Mineter, G.G.Wilkinson, S.Dowers and R.G. Healy, 1998, "Parallel processing algorithm for GIS : Implementation Case Study : Generalisation of Raster Data", Tayor and Francis.

Monmonier, Mark, 1982, "Computer-Assisted Cartography : Principles and Prospects" Page 76 - 80. Englewood cliffs, N.J. Prentice-Hall Inc.

M.P. Armstrong and R. Marciano, 1994, "Inverse-Distance-Weighted Spatial Interpolation Using Parallel Supercomputers," Photo - grammetric Engineering and Remote Sensing, Vol. 60, No. 9, September, pp.1097-1103.

MPI. A Message-Passing Interface Standard, 1995, MPI Forum. / MPI-2, 1997.

Neil MacDonald, Elspeth Minty, Tim Harding, Simon Brown, "Writing Message-Passing Parallel Programs with MPI", Edinburgh Parallel Computing Centre, The University of Edinburgh.

Soohong Park, 1996, "Integration of cellular automata and geographic information system for modeling spatial dymamics", Degree of Doctor of Philosophy in University of South Carolina

Xiong, D. and D.F, Marble, "Strategies for real-time spatial analysis using massively parallel SIMD computers : an application to urban traffic flow analysis", International Journal of Geographical Information systems, vol.10, No.6, pp 756-789.