위성영상 GIS를 위한 클러스터링 시스템 설계

조정우[0] 김학두 김진석
서울시립대학교 컴퓨터·통계학과
{jwjo95, hdkim94, jskim}@venus.uos.ac.kr

Design of Clustering System for Satellite-image Geographic Information System

Jeongwoo Jo[0] Hak Du Kim and Jin Suk Kim
Dept. of Computer Science & Statistics, University of Seoul

요 약

최근 위성영상을 이용한 GIS 시스템이 많이 생겨나면서 위성영상을 이용한 연구가 많이 진행중이다. 하지만 위성영상의 경우 파일 자체의 사이즈가 크기 때문에 본 영상을 처리하기가 쉽지 않으며 시간 또한 많이 소모되게 된다. 또한 효율적인 자료처리를 하기 위해서는 고성능의 하드웨어가 필요하다는 문제점이 있다. 따라서 본 논문에서는 병렬처리를 이용하는 클러스터링 시스템을 사용하여 대용량의 위성영상을 보다 빠르고 효율적으로 처리할 수 있는 시스템을 설계하였다. 본 논문에서 제안한 시스템을 사용하면 앞의 문제점을 해결할 수 있으며 빠른 영상 분석이 가능하게 된다. 병렬 컴퓨터의 노드를 증가시키면서 제안한 시스템의 속도가 빨라지는 것을 실험을 통해 보였다.

1. 서 론

최근 위성영상을 이용한 GIS 시스템이 많이 생겨나면서 위성영상을 이용한 연구가 많이 진행중이다. [7, 8, 9, 10, 11] 하지만 위성영상을 사용하려면 여러 문제점을 해결하여야 한다. 그 문제점은 위성영상의 경우 파일 자체의 사이즈가 크기 때문에 본 영상을 처리하기가 쉽지 않다는 것이다. 또한 처리시간이 많이 소모되게 되며 효율적인 자료처리를 하기 위해서는 고성능의 하드웨어가 필요하다는 문제점이 발생한다. 따라서 본 논문에서는 병렬처리를 이용하는 클러스터링 시스템을 사용하여 대용량의 위성영상을 보다 빠르고 효율적으로 처리할 수 있는 시스템을 설계하였다. 즉 여러 프로세서에서 하나의 위성영상을 처리하는 방법이다. 본 논문에서 제안한 시스템을 사용하면 앞의 문제점을 해결할 수 있으며 빠른 영상 분석이 가능하게 된다. 본 논문의 구성은 다음과 같다. 2장에서는 본 논문에서 제안한 시스템과 기존 시스템의 문제점을 살펴보고, 3장에서는 위성영상을 병렬처리 하는 방법을 살펴본다. 그리고 4장에서 본 논문에서 구현한 시스템의 수행결과를 살펴보고 마지막으로 5장에서 결론 및 향후 연구방향을 고찰한다.

2. 클러스터링 시스템 설계

본 논문에서 제안한 시스템의 구조는 <그림 1>과 같다. 즉 대용량 위성영상을 처리하고자 할 때 위성영상을 기존의 방법처럼 하나의 시스템에서 처리하는 것이 아니라 여러개의 시스템을 두고 각 시스템들끼리 위성영상의 일정 부분을 처리하게 된다. 따라서 하나의 시스템을 사용하는 것 보다 속도면에서 이점을 가지게 된다. 본 논문에서는 위성영상을 위한 GIS 소프트웨어로 GRASS를 사용하였다. [11] GRASS는 LINUX 상에서 동작하는 GIS 툴로서 소스코드가 공개되어 있어서 실험을 하는데 적당하다. 대부분의 GIS 시스템에서는 위성에서 수신한 영상 그 자체를 사용하기보다는 여러 가지 처리를 통하여 변형시켜 사용하게 된다. 그 하나의 예를 살펴보면 다음과 같다. 위성에서 촬영한 영상은 지구가 구로 이루어져 있기 때문에 실제 지도상의 영상과 편차가 생기게 된다. 이러한 편차를 교정해 주는 작업을 Grass는 Rectification 작업을 통하여 지원하고 있다.

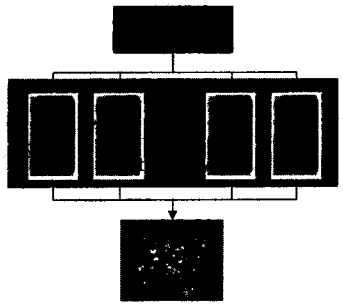

그림 1. 시스템의 구조

하지만 위성영상의 크기가 몇 백 Megabyte에서 몇 Gigabyte에 이를 정도로 거대하기 때문에 Rectification 작업은 수십 분 내지 수백 분의 시간이 소요된다. 이러한 작업을 고성능의 시스템을 통하여 보다 빠른 시간 내에 처리할 수도 있지만 그에 따르는 많은 비용을 감수해야 한다. 하지만 네트워크로 연결되어 있는 기존의 시스템들을 이용하여 병렬로 처리한다면 보다 저렴한 비용으로 같은 성능을 기대할 수 있으며, 확장성 또한 효율적이기 때문에 유지비용 또한 저렴해 질 수 있는 강점이 있다. 따라서 본 논문에서는 클러스터링 시스템을 이용하여 Grass의 Rectification작업을 처리한다.

3. 병렬처리 방법

클러스터링 시스템은 여러 개의 노드로 구성된다. 각 노드는 완전히 독립적인 컴퓨터 시스템이다. 클러스터를 이루는 각 노드는 하나의 작업을 분할하여 처리하게 되고 네트워크를 통하여 서로 데이터를 교환하게 되며, 보통 하나의 마스터 노드가 존재하여 다른 노드들에게 데이터를 전달하거나 수집하는 역할을 담당한다.

3.1 노드들간의 작업 분할 방법

다음 <그림 2>는 본 논문에서 사용한 병렬 방법을 나타낸다

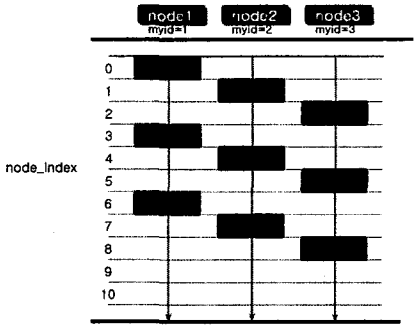

그림 2. 병렬처리 수행 방법

위 그림에서 node_index는 while문이 반복될 때마다 1씩 증가하는 변수이며 myid는 각 노드에 부여된 유일한 값이다. 각 노드는 같은 프로그램을 수행하게 되고, while문으로 진입하게 되면 node_index 변수를 참고하여 몇 번째 반복이 자신이 수행할 부분인지를 결정하게 된다. 예를 들면, node1은 myid가 1이며 while문을 반복하다가 node_index를 노드의 개수로 나눈 나머지가 1일 때만 그 부분을 수행하게 되고 다른 값일 때는 무시하게 된다.

3.2 시간의 흐름에 따른 병렬처리 과정

각 노드들은 소켓을 통하여 데이터를 전달한다. 다음 <그림 3>은 시간의 흐름에 따른 병렬 수행 과정을 나타낸다. 병렬처리과정을 시작하게 되면 마스터 노드는 다른 노드로부터 데이터를 전달받기 위해 서버 소켓을 생성하게 되고 다른 노드들은 마스터 노드에게 데이터를 전달하기 위해 클라이언트 소켓을 생성하게 된다. 연결이 이루어지게 되면 마스터를 포함한 노드들은 각자 자신이 맡은 역할을 수행하게 된다.

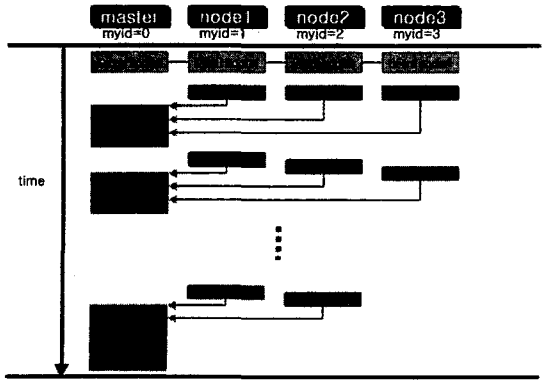

그림 3. 시간흐름에 따른 병렬처리 흐름

마스터는 다른 노드들로부터 데이터를 받기 위해 대기하게 되고, node1부터 node3까지 순차적으로 데이터를 받게 된다. rectify() 함수는 이미지를 처리함에 있어 while문을 반복할 때마다 임시파일에 덧붙여 가며 하나의 파일을 생성해 나가게 되고 while문이 끝나면 다른 수행부분을 처리한 후에 전체적으로 새로운 파일에 임시파일을 옮기게 된다. 따라서 마스터가 받는 데이터의 순서는 반드시 지켜져야 한다. 모든 반복과정이 끝나게 되면 마스터를 제외한 노드들은 새로운 파일을 생성하지 않고 마스터와의 통신을 끊고 프로그램을 끝내게 되며 마스터는 실제 새로운 영상 파일을 생성하게 된다.

<표 1>은 rectify.c 파일의 수정 부분이다.

표 1. rectify 소스 수정

```
int rectify (char *name, char *mapset, char *result, int order)
{
    int node_index;
    .....
    if ( myid!=0 )
    {
        _server_sock();
        _connection();
    }
    else
        create_client_sock();

    node_index = 0;

    while (ncols > 0) {
        while (nrows > 0) {
            if ( myid == 0 ) {
                compute_georef_matrix (&cellhd, &win,order);
                recv_matrix(node_index % (nodes-1), win.rows,
win.cols);
                write_matrix(row,col);
            }
            else {
                if ( (node_index % (nodes-1)+1) == myid )
                {
                    compute_georef_matrix
(&cellhd, &win,order);
                    perform_georef(infd,rast);
                    send_matrix();
                }
            }
            node_index++;
        }
    }
    close_socks();
    if ( myid==0 )
        write_map(result);
    return 1;
}
```

4. 결 과

<그림 4>는 실험에 사용된 이미지이며 <그림 5>는 Rectification 처리를 거친 이미지이다 <그림 4>는 각 물체의 선이나 윤곽이 고르지 않는 반면 <그림 5>에서는 그러한 문제점들이 수정된 것을 볼 수 있다.

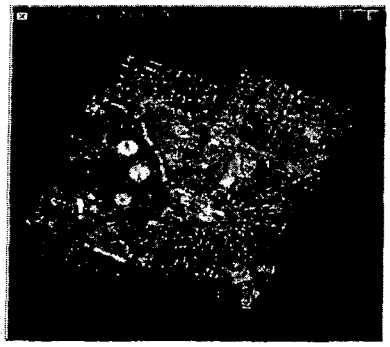

그림 4. 실험에 사용된 이미지

그림 5. Rectification 수행 결과 이미지

실험에 사용되는 장비는 Alpha 시스템 4 노드로 구성된 클러스터링 시스템이다. 한 노드는 600MHz의 Alpha CPU와 786Mbyte의 메모리로 구성되어 있고 노드들은 Myrinet으로 연결되어 있다. 실험에 사용된 이미지는 162.2Mbyte의 크기를 가진 위성영상이며 TIFF확장자를 가진 영상을 Grass의 파일 포맷에 맞게 수정된 이미지를 사용하였다. 다음 <표 2>와 <그림 6>은 클러스터를 구성하는 노드의 수에 따른 Rectification 처리 수행 시간을 나타낸다. 노드의 수가 증가함에 따라 수행시간이 단축됨을 알 수 있다. 그래프에서 삼각형으로 구성된 그래프가 이상적인 기대되는 결과이며 사각형으로 구성된 그래프가 실험의 결과이다. 이상적인 값과 실험 값의 차이는 통신시간, 노드의 전송 대기시간, 마스터 노드의 파일 기록시간이 실험의 값에 포함되기 때문이다.

표 2. 노드 수에 따른 처리 시간

	1 node	2 node	3 node
time(sec)	7702.2	4709	3785.5

Rectification 과정에서 사용되는 메모리의 크기는 global.h 파일에 선언된 NCOLS와 NROWS에 의해서 결정된다. 시스템의 메모리가 허용하는 선에서 이 상수들의 값을 크게 하면 while문의 반복횟수를 줄일 수 있다. 그렇게 되면 노드의 순수한 연산 비용은 증가하게 되지만 통신비용과 같은 외적인 비용을 줄일 수 있으므로 전체적인 병렬처리의 성능을 향상시킬 수 있다.

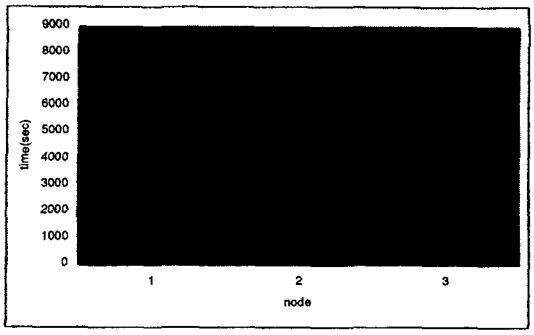

그림 6. 노드 수에 따른 처리 시간

5. 결론 및 향후 연구 방향

본 논문에서는 위성영상을 고속으로 처리하기 위해 클러스터링 시스템을 이용하였다. 기존의 GIS에서처럼 하나의 프로세서를 가지고 위성영상을 처리할 때 발생할 수 있는 여러 문제점을 해결할 수 있었다. 본 연구에서 추가적으로 필요한 사항은 위성영상을 여러 개의 노드들로 나누어서 처리할 경우 네트워크 통신 속도 때문에 속도향상이 점점 낮아지는 문제를 해결하는 것이다. 이를 해결하기 위해서는 네트워크의 통신 속도를 향상시켜야한다. 그렇게 되면 네트워크의 통신으로 인한 오버헤드가 제거되어 최종 속도향상을 높일 수 있을 것이다. 또한 본 연구를 위성영상의 rectification 연산 뿐만 아니라 다른 GIS연산에 적용해 보아야 할 것이다.

참고문헌

[1] W.S. Lin, W.H. Lau, K. Hwang, Fellow, X. Lin, and Y.S. Cheung, "Adaptive Parallel Rendering on Multiprocessors and Workstation Clusters," IEEE Trans. on Paralle and Distributed systems, vol. 12, pp. 241-258, Mar, 2001.

[2] W. Li, D. Zhang, Z. Liu, and X. Qiao, "A Parallel Algorithm for Image Information Restoration," Proc. of High Performance Computing in the Asia-Pacific Region, vol. 2, pp. 790-793, 2000.

[3] M. Ishii, "Cluster Technologies for High Performance Computing," Proc. of Parallel Architectures, Algorithms, and Networks, pp. 168-170, 1999.

[4] J. L. Martin, G. Aranguren, J. Ezquerra, and P. Ibanez, "Parallel Raster Image Processor for PCB Manufacturing," Conf. on 20th International IECON 94, vol. 2, pp. 1184-1189, 1994.

[5] S. Barsamian, "Uses of Parallel Processors in a Photo Based Image Generator," IEEE Proc. of Aerospace and Electronics Conference NAECON 90, vol. 2, pp.688-693, 1990.

[6] S. Shekhar, S. Ravada, V. Kuma, D. Chubb, and G. Turner, "Declustering and Load_balancing Methods for Parallelizing Geographic Information Systems," IEEE Trans on. Knowledge and Data Engineering, vol. 10, pp. 632-655, 1998.

[7] ESRI, "http://www.esri.com".

[8] Intergraph, "http://www.intergraph.com".

[9] Mapinfo, "http://www.mapinfo.com".

[10] KSIC(한국공간정보통신), "http://www.ksic.net".

[11] GRASS, "http://grass.itc.it".