Journal of Korea Multimedia Society Vol. 14, No. 6, June 2011(pp. 835-848)

대용량 지형 데이터를 위한 웹 기반 분산 가시화 시스템

황규현[†], 윤성민[††], 박상훈[†††]

요 약

본 논문에서는 방대한 지형 데이터의 효과적 가시화를 위한 클라이언스-서버 기반의 분산/병렬 시스템을 제안한다. 이 시스템은 웹 기반으로 수행되는 클라이언트 GUI 프로그램과 복수의 PC 클러스터에서 구동되는 분산/병렬 서버 프로그램으로 구성된다. PC 뿐만 아니라 모바일 기기에서도 클라이언트 프로그램이 수행될 수 있도록 자바 기반의 OpenGL 그래픽스 라이브러리인 JOGL을 사용하여 GUI를 설계하였으며, 사용하는 기기의 현재 사용 가능한 메모리 크기와 화면의 최대 해상도 정보를 서버에게 전달하여 서버의 작업을 최소화하였다. 서버로 사용된 PC 클러스터는 분산된 지형 데이터를 액세스 하고 이를 클라이언트로부터 받은 정보에 따라 적절히 리샘플링 한 후 이를 다시 전송하는 작업을 담당한다. 서버의 각 노드들뿐만 아니라 클라이언트까지 캐시 자료구조를 유지함으로써 분산된 방대한 지형 데이터의 반복 접근 시 발생되는 지연 시간을 최소화하도록 설계하였다.

Web-Based Distributed Visualization System for Large Scale Geographic Data

Gyuhyun Hwang[†], Seongmin Yun[††], Sanghun Park[†††]

ABSTRACT

In this paper, we propose a client-server based distributed/parallel system to effectively visualize huge geographic data. The system consists of a web-based client GUI program and a distributed/parallel server program which runs on multiple PC clusters. To make the client program run on mobile devices as well as PCs, the graphical user interface has been designed by using JOGL, the Java-based OpenGL graphics library, and sending the information about current available memory space and maximum display resolution the server can minimize the amount of tasks. PC clusters used to play the role of the server access requested geographic data from distributed disks, and properly re-sample them, then send the results back to the client. To minimize the latency happened in repeatedly access the distributed stored geography data, cache data structures have been maintained in both every nodes of the server and the client.

Key words: Geographic Data(지형 데이터), Web-Based Visualization(웹기반 가시화), Java OpenGL (JOGL)(자바 오픈지엘(JOGL)), Distributed/Parallel Processing(분산/병렬 처리), PC Cluster(PC 클러스터), Cache Data Structures(캐시 자료구조)

※ 교신저자(Corresponding Author) : 박상훈, 주소 : 서울시 중구 필동 3가 26 동국대학교 멀티미디어학과(100-715), 전화 : 02)2260-3765, FAX : 02)2260-3766, E-mail : mshpark@dongguk.edu
접수일 : 2010년 9월 28일, 수정일 : 2011년 3월 14일
완료일 : 2011년 4월 25일

[†] 준회원, 동국대학교 멀티미디어학과
(E-mail : spony@dongguk.edu)
[††] 준회원, 동국대학교 멀티미디어학과
(E-mail : tajoal902@naver.com)
[†††] 정회원, 동국대학교 멀티미디어학과
※ 이 연구는 2010학년도 동국대학교 연구년 지원에 의하여 이루어졌음.

1. 서 론

GPGPU(General Purpose Graphics Processing Unit) 컴퓨팅 기술과 이를 응용한 실시간 렌더링 기법들이 소개되면서 과거에는 불가능하던 범용 PC를 이용한 다양한 형태를 갖는 데이터들의 실시간 가시화가 가능하게 되었다[1,2]. 하지만 대부분의 알고리즘들이 가시화의 대상이 되는 데이터 크기를 그래픽 메모리 보다 작은 경우로 가정하고 있기 때문에, 수십에서 수 백 기가 바이트(giga byte)에 이르는 방대한 데이터의 실시간 가시화에는 한계를 갖는다[3-5].

기존에 소개된 방대한 데이터 가시화 기법에는 데이터의 자료구조를 변형하여 알고리즘에 응용하는 방법과 분산/병렬 처리에 기반을 둔 PC 클러스터를 이용하는 방법 등이 있다. 첫 번째 방법은 전체 데이터를 트리와 같은 특정 형태의 자료구조로 저장하거나 압축 형태로 인코딩한 후, 가시화에 필요한 일부 또는 전체 데이터를 로드하여 가시화하는 방법으로[6-9], 변형된 자료구조 또는 압축된 상태로도 전체가 메모리에 로드 될 수 없는 경우에는 스와핑(swapping)으로 인해 여전히 실시간 가시화가 어렵다는 한계를 갖는다[2].

두 번째 방법은 GPU가 탑재된 다수의 범용 PC들로 구성된 클러스터를 병렬 렌더링(parallel rendering)에 이용하는 것이다[5,10,11]. 이 기법은 전체 작업을 데이터 공간 또는 스크린 공간으로 분할하고, 나눠진 부분 작업을 클러스터를 구성하는 각 노드(node)들에 할당하여 GPU를 이용하여 병렬 가시화하는 알고리즘으로 구현된다. 노드의 수를 추가하는 만큼 메모리 공간과 계산 성능의 확장을 기대할 수 있기 때문에 이러한 병렬 렌더링 기법들을 이용함으로써 많은 연산이 요구되는 방대한 크기의 데이터를 효율적으로 가시화할 수 있다[12,13]. 그러나 실제 구현에서는 각 노드들의 작업 부하 균형(load balancing) 문제를 해결해야 하고, 순차처리(sequential processing) 유무에 따라 병렬 가시화 알고리즘의 성능이 제한되며, 대용량 데이터를 반복적으로 액세스 하는 렌더링 알고리즘 특성에 따라 네트워크 응답 지연과 그에 따른 동기화(synchronization) 비용이 발생한다[14-17]. 따라서 효과적인 분산/병렬 가시화 프로그래밍 모델뿐만 아니라 빠른 데이터 액세스를 위한 기법의 개발이 중요한 요소로 고려되어야 하며, 각 모듈간의 통신에서 발생되는 지연시간을 최소화하기 위한 알고리즘도 설계되어야 한다[18-21].

본 논문에서는 웹 기반 응용을 위한 대용량 지형 데이터의 분산/병렬 가시화 시스템을 제안한다. 클라이언트-서버 기반으로 동작되는 이 시스템은 분산된 데이터를 다수의 PC 클러스터를 이용하여 분산/병렬 가시화하게 된다. 이때 각 클러스터를 구성하는 노드들은 분산된 데이터 접근 시 발생되는 지연 시간을 최소화하기 위해 가시화에 사용된 데이터를 캐시(cache) 자료구조에 저장하여 가시화에 이용될 데이터를 반복적으로 액세스 하지 않도록 설계하였다. 클라이언트, 서버를 포함한 전체 시스템을 구성하는 각 요소마다 계층적으로 캐시 구조를 유지하도록 하여 빠른 데이터 액세스가 가능함을 실험을 통해 보인다.

제안한 서버-클라이언트 기반 분산/병렬 가시화 시스템은 원격 PC 클러스터에서 가시화 작업을 수행하고, 가시화 된 결과를 사용자의 단말기로 전송받아 이를 디스플레이하기 때문에 고성능 GPU가 탑재 되어 있지 않은 모바일 기기 혹은 넷북(net-book)과 같은 저성능 PC 상에서도 대규모 데이터를 이용하여 높은 품질의 가시화 결과를 대화식으로(interactively) 확인할 수 있다. 이 때 사용된 단말기에 탑재된 메모리의 크기나 화면의 최대 디스플레이 크기 등의 정보를 서버에게 전달하고, 서버는 그에 맞는 결과를 보내도록 하여 필요한 만큼의 최소 연산과 데이터 액세스만이 수행되도록 구현하였다.

2. 시스템 설계

2.1 전체적인 구조

그림 1은 클라이언트-서버 기반 분산/병렬 가시화 시스템의 전체적인 구조를 나타낸 것으로 방대한 크기의 지형 데이터를 효율적으로 가시화하기 위해 다수의 범용 PC로 구성되는 여러 개의 클러스터를 사용자가 선택하여 렌더링을 요청할 수 있는 구조를 갖는다. PC 클러스터는 한 대의 마스터 노드와 n개의 슬레이브(slave) 노드, 그리고 파일 서버로 구성된다. 클라이언트의 요청에 따라 마스터 노드는 요청 작업을 병렬로 처리할 수 있도록 전체 작업 W를 $W \subset \bigcup_{i=1}^{n} W_i$ 를 만족하는 부분 작업 W_i 들로 데이터 공간에서 분할한다. 각 슬레이브 노드들은 할당받은

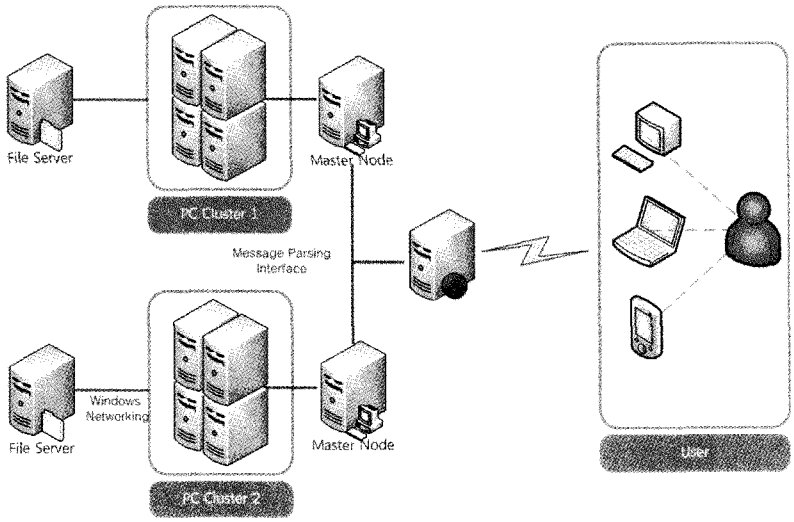


그림 1. 클라이언트-서버 기반의 분산/병렬 가시화 시스템의 구조

작업을 수행하고 그 결과를 마스터 노드로 전송하며, 마스터 노드는 각 프로세스들로부터 받은 결과들을 취합하여 클라이언트에게 전달한다. 슬레이브 노드들은 마스터 노드의 명령에 따라 분산된 데이터를 액세스 하고 리샘플링 하는 작업을 병렬로 수행하는데, 이를 구현하기 위해 MPI(message passing interface)를 이용하였다. 클라이언트는 요청에 따라 처리된 결과를 전달 받아 이를 JOGL(Java OpenGL)로 구현된 GUI를 통해 대화식으로 가시화한다.

중복된 연산을 피하고 전체 시스템의 성능을 향상시키기 위해, 마스터 및 슬레이브 노드 뿐만 아니라 클라이언트에도 별도의 캐시 구조를 유지하도록 설계하였다. 또한 클러스터에서의 작업량을 최소화하기 위해, 클라이언트 시스템의 메모리 크기와 디스플레이 화면의 최대 해상도를 고려하여 최종적으로 전송할 데이터의 LOD(Level Of Details)를 마스터 노드가 결정하도록 하고, 그에 맞도록 병렬 수행이 이뤄지도록 구현하였다.

2.2 분산/병렬 처리 모듈

본 연구에서 제안하는 시스템은 클라이언트가 가시화를 위해 특정 지역의 데이터를 요청하면, 대규모 지형 데이터에서 요청 지역 데이터를 병렬로 추출하여 클라이언트에 전송하는 구조로 되어 있다. 작업의 요청에 따른 작업 분할은 마스터 노드에 의해 직렬로 수행되며, 분할 작업들은 슬레이브 노드에 의해 병렬로 처리된다.

클라이언트가 TCP/IP를 이용하여 마스터 노드에게 작업을 요청하면, 마스터 노드는 요청된 작업이 수행된 적이 있는지 여부를 확인하기 위해 마스터 캐시의 테이블을 검색한다. 이 테이블은 이전에 수행된 작업을 통해 획득한 결과를 저장하고 있기 때문에 이 테이블에 등록되어 있지 않은 작업인 경우에만 마스터 노드는 슬레이브 노드에게 작업을 지시한다. 이 때 분해된 작업은 데이터 액세스의 지역성(locality)에 따라 슬레이브 노드로 할당된다. 슬레이브 노드에서 처리된 작업의 결과가 취합될 때, 마스터 노드는 그 결과를 캐시 자료구조에 저장한다. 저장된 작업의 결과는 클라이언트 시스템의 성능에 따라 LOD가 적용되어 전달되며, 클라이언트는 이를 이용하여 대화식으로 렌더링을 수행한다.

마스터 노드는 작업 요청의 유무에 따라 슬레이브 노드의 캐시 메모리를 관리하게 된다. 대기상태의 마스터 노드는 표 1과 같은 인자를 클라이언트로부터 전달받으며, 이를 병렬 처리를 위한 작업 분할에 이용한다.

클라이언트가 작업을 요청하면 마스터 노드는 해당 작업을 분할하기 위해 분산되어 있는 데이터의 정보를 우선 확인한다. 예를 들어, 그림 2와 같이 분산되어 있는 다수의 데이터가 가시화 작업을 위해 사용되는 경우, 분산된 데이터의 정보를 취합하고 수치표고의 간격 및 정사영상의 배율을 이용하여 전체 작업을 블록 형태로 분할한다. 그림 2에서 분산된 4

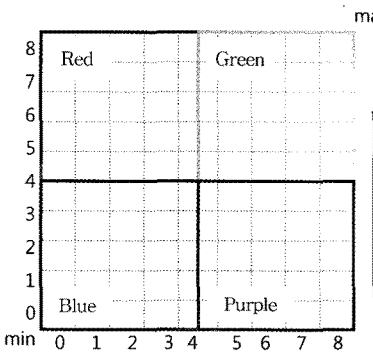
표 1. 가시화를 위해 마스터 노드에 전달되는 인자

항 목	설 명
요청 지역	요청 지역의 부호화된 지역 인덱스
$\overrightarrow{min}$	요청 지역 중 가시화 대상인 영역의 최소 좌표
$\overrightarrow{max}$	요청 지역 중 가시화 대상인 영역의 최대 좌표
수치표고의 간격	요청 지역 중 가시화 대상 영역의 샘플링 간격
정사영상의 배율	요청 지역 중 가시화 대상 영역의 정사영상 배율

개의 전체 데이터는 클라이언트의 요청에 따라 세부 블록들로 분할된다.

클라이언트로부터 그림 3과 같이 파란색과 빨간색으로 정의되는 영역이 요청된 경우, 마스터 노드는 가시화 작업을 위해 클라이언트로부터 전달 받은 인자 $\overrightarrow{min}$와 $\overrightarrow{max}$를 이용하여 액세스 영역을 선택한 다음, 요청 작업의 분할은 블록을 기반으로 수행한다.

그림 3과 같이 분해된 작업 블록들은 마스터 노드에 의해 태스크 큐(task queue)에 저장된 다음, 태스크 큐의 순서대로 슬레이브 노드들에게 분배된다. 마스터 노드는 분배된 작업의 처리 정도를 슬레이브 노드에게 질의하여, 휴식 상태의 슬레이브 노드가 존재하는지 여부를 판단한다. 태스크 큐에 작업이 있을 경우, 마스터 노드는 휴식 상태의 슬레이브 노드에게 태스크를 다시 할당하게 된다. 태스크 큐의 작업이 모두 완료되면 마스터 노드는 요청 영역에 해당하는 데이터만 취합하여 클라이언트에게 요청 결과를 전송하게 된다. 그림 4는 클라이언트로부터 작업이 요청된



Color	Encoded Index
Red	33611005
Green	33611006
Blue	33611015
Purple	33611016

그림 2. 블록 기반 영역 분할

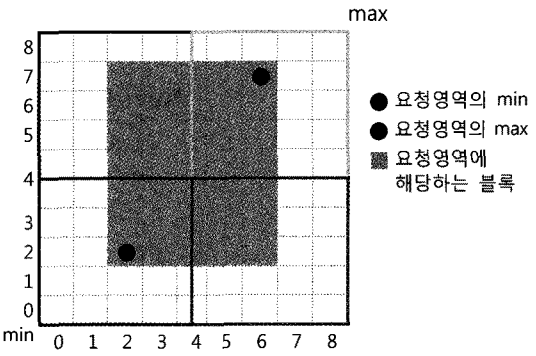


그림 3. 분산/병렬 처리를 위한 부분 작업 생성

경우, 마스터 노드의 병렬 처리를 위한 의사 코드이다.

코드 1은 마스터 노드의 작업 할당에 따른 슬레이브 노드의 작업 처리 의사 코드이다. 작업 요청이 없는 경우 슬레이브 노드는 대기 상태인 휴식 상태로 전환되며, 태스크 할당이 이루어진 경우 미리 설정된 프로시저를 수행하게 된다. 코드 2의 두 번째 줄과 같이 슬레이브 노드는 할당 받은 영역을 원본 데이터에 맞게 분할한다. 만약 할당 받은 영역이 복수개의 원본 데이터 경계 부분에 걸쳐 있다면, 할당된 영역

```
1 : procedure Parallel_Processing (Region Index, min→, max→, Interval, Magnification)
2 : Block-based task decomposition
3 : Creates the task queue
4 : n ← number of process
5 : m ← number of tasks
6 : for i ← 1, n do
7 : Allocates a task at the front of the queue to i-th slave and then pop front
8 : end for
9 :
10 : while m ≠ 0 do
11 : Receive data of result of a task from any slave
12 : m ← m − 1
13 : if the queue is not empty then
14 : Allocates a task at the front of the queue to the slave and then pop front
15 : end if
16 : end while
17 :
18 : Merge and adjust received block data to be equivalent to requested region
19 :
20 : end procedure
```

코드 1. 마스터 노드의 병렬 처리 의사 코드

은 경계가 걸쳐진 m개의 세부 블록으로 분할된다. 슬레이브 노드는 분할된 영역에 따른 모든 원본 조각 데이터를 읽은 후 필요한 데이터만 추출하게 된다. 할당 받은 영역의 데이터 추출이 이루어지면 슬레이브 노드는 설정된 수치표고의 간격과 정사영상의 배율에 따라 데이터를 조정한 후 작업 결과를 마스터 노드에게 전송하게 되며, 작업 결과 전송이 끝나게 되면 슬레이브 노드는 다시 휴식상태로 전환되고 마스터 노드로부터 작업이 할당될 때까지 대기하게 된다.

```
1: procedure Slave_Procedure (Task Index, min, max)
2:     Splits requested region with original patch data
3:     m ← number of splitted region
4:     for i ← 1, m do
5:         Extracts data from original patch data
6:     end for
7:     Scaling the data with given condition
8:     Send the data to the master
9: end procedure
```

코드 2. 슬레이브 노드의 작업 처리 의사 코드

2.3 캐시 자료구조

그림 4는 캐시 분할의 예를 나타낸 것으로 마스터 노드는 클라이언트 요청에 따라 가시화 결과를 효율적으로 전송하기 위해 메모리에 캐시 자료구조를 위한 영역을 확보하고 이를 관리한다. 클라이언트로부터 최초 작업 요청이 이루어질 때, 수치표고 간격과 정사영상 배율에 따라 캐시의 초기화가 이루어진다. 최초 작업 요청 이후 수치표고 간격과 정사영상 배율 등의 캐시 정보가 변경되지 않는 경우 캐시에 저장된 데이터가 활용되며, 클라이언트의 요청에 따라 캐시 정보가 변경된 경우 캐시의 초기화가 수행된다.

캐시 삽입과 삭제의 기본 단위는 해상도 크기가 32×32인 블록 형태로 되어 있으며, 클라이언트의 요청에 따라 캐시 블록의 크기가 결정 된다. 클라이언트가 요청한 수치표고 간격이 5미터, 정사영상 배율이 0.5인 경우 하나의 캐시 블록은 32×32개의 고도 데이터를 가진다. 따라서 이것은 (32×5)×(32×5)미터에 해당되는 지역의 고도 데이터와 0.5배 확대한 정사영상의 정보를 의미한다.

요청된 데이터 블록이 캐시에 존재하는지 여부를 빠르게 검색하게 위해, 마스터와 슬레이브 노드의 캐시 시스템은 내부적으로 해시(hash) 자료구조를 이용하여 데이터를 관리한다. 해시 구조의 검색 비용은 $O(1)$이고, 캐시 시스템 내부적으로 다차원 해시 구조를 사용하였다. 표 2는 마스터 캐시 시스템의 해시 데이터 구조를 표현한 것으로 캐시 블록의 1차원 인덱스에 따른 데이터 포인터를 전달 받도록 설계 하였다.

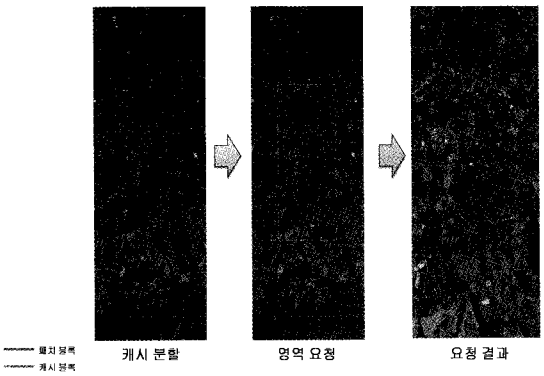


그림 4. 영역 요청에 따른 캐시 분할의 예

표 2. 마스터 캐시 시스템의 해시 자료구조

키	해당 블록의 1차원 인덱스
값	블록 데이터에 대한 포인터

마스터 노드 병렬 처리(코드 1)에서 작업의 분해와 할당 시 데이터의 존재 유무를 캐시에서 확인한 후, 캐시에 저장되어 있지 않은 작업만을 슬레이브 노드에게 할당하게 된다. 캐시 데이터 존재 여부를 확인하기 위해 캐시 작업 큐에 저장된 블록 인덱스로 캐시 시스템의 해시 테이블을 검색하여 캐시 블록 데이터의 존재를 확인하도록 하였다.

슬레이브 노드 역시 데이터 추출 성능 및 연산 속도의 향상을 위하여 각 노드마다 캐시를 유지하도록 설계하였다. 슬레이브 캐시 시스템은 캐시에 저장된 각 원본 조각 데이터에 대한 이용률을 별도로 저장하고, 캐시 오버플로가 발생하는 경우 캐시 이용률이 가장 낮은 캐시 데이터를 삭제하도록 하였다. 캐시 데이터를 확인하기 위해 슬레이브 노드는 표 3과 같은 해시 자료구조를 이용한다.

캐시 자료구조가 유지되는 장치는 클라이언트인 웹 기반 응용 프로그램이 동작되는 환경에 따라 결정된다. 주기억장치의 용량이 1 기가 바이트 이상인 범용 PC상에서 클라이언트 프로그램이 동작되는 경우에는 주기억장치를 이용하여 캐시 데이터가 저장하

표 3. 슬레이브 캐시 시스템의 해시 자료구조

키	조각 데이터의 부호화된 인덱스
값	조각 데이터에 대한 포인터

고 관리하게 되지만, 주기억장치의 용량이 충분하지 않은 스마트폰, 넷북과 같은 저성능 환경에서는 보조 기억장치를 이용한다.

2.4 웹 기반 응용을 위한 클라이언트 모듈

클라이언트는 그림 5와 같이 각 기능에 따라 분류된 다수의 세부 모듈로 구성되어 있으며, 각 세부 모듈이 지원하는 기능은 표 4와 같다. 각 모듈은 구조화되어 있지 않은 입력 데이터를 트리 형태로 구조화하고 프로그래머블 쉐이더(programmable shaders)를 이용한 실시간 가시화 기능을 지원한다. 웹을 기반으로 클라이언트 프로그램이 다운로드 되고 구동될 수 있도록 하기 위해 Java 환경에서 OpenGL 함수들을 구현한 JOGL을 이용하여 렌더링과 가시화에 관련된 클라이언트 모듈들을 구현하였다. 실제로 지형 데이터의 렌더링, 시간가변(time-varying) 데이터의 입력과 처리 등의 기능은 JOGL을 기반으로 프로그램 되었으며, 각 모듈은 2개 이상의 보조 모듈과 데이터 처리를 위한 공통 모듈로 구성되어 있다.

사용자는 클라이언트 GUI가 제공하는 다양한 메뉴 버튼을 이용해 원하는 가시화 작업을 수행하게 된다. GUI를 통해 이용할 PC 클러스터에 접속한 후, 가시화를 원하는 지형 데이터를 선택하고 특정 지역을 마우스로 선택하면, 마스터 노드의 명령에 따라 슬레이브 노드들은 분산 디스크에 저장된 대규모 지형 데이터에 접근하여 해당 데이터를 액세스하고, 클라이언트가 수행되는 플랫폼의 성능에 따라 LOD가

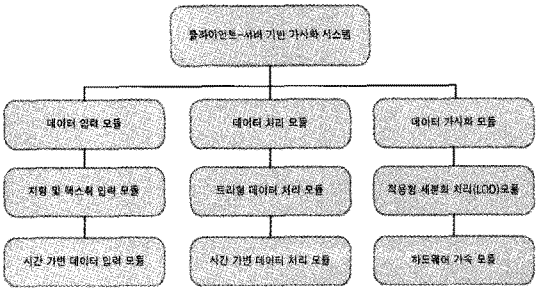


그림 5. 클라이언트 모듈의 구조도

적용된 결과를 생성하여, 이를 클라이언트에게 전달한다. 따라서 클라이언트는 현재 사용 가능한 메모리 크기와 디스플레이 가능한 최대 해상도에 따라 적절하게 처리된 최적의 데이터만을 받게 된다.

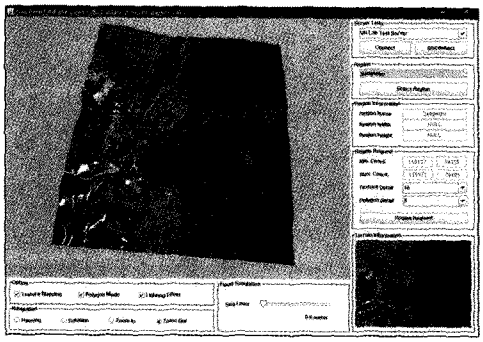
그림 6은 지형 데이터의 가시화 결과의 예로서, 약 100만개의 다각형과 15,604×18,564 해상도의 정사영상으로 구성된 지형 모델이 사용되었다. 저성능의 기기에서도 대화식으로 디스플레이가 가능하도록 클라이언트에서도 적응형(adaptive) LOD와 캐시 자료구조를 지원하도록 구현하였다. 또한 클라이언트의 그래픽스 모듈 부분은 JOGL을 기반으로 구현되어 있기 때문에 플랫폼에 독립적으로 수행될 수 있는 장점을 갖는다. 또한, 클라이언트 GUI와 마스터 노드는 다중스레드(multi-thread)를 이용할 수 있도록 구현되었기 때문에 요청받은 작업을 수행중인 상황에도 또 다른 새로운 작업을 받아들여 처리할 수 있다.

3. 가시화 결과

본 연구를 통해 개발된 분산/병렬 가시화 시스템은 방대한 크기를 가지는 지형 데이터뿐만 아니라

표 4. 클라이언트 모듈들의 기능

모 듈 명	기 능
데이터 입력 모듈	가시화에 필요한 강수량 또는 기후 정보 입력을 위한 모듈
지형 및 텍스쳐 입력 모듈	가시화 대상 지형 및 텍스쳐 정보의 입력을 위한 모듈 (*.dem, *.obj, *.txt, BMP, PNG, JPG 등의 형식을 지원)
시간가변 데이터 입력 모듈	시간가변 데이터의 입력을 위한 모듈(*.obj, *.dbf등의 형식을 지원)
데이터 처리 모듈	입력 데이터를 가시화가 가능한 형태로 처리하기 위한 모듈
트리형 데이터처리 모듈	적응형 세분화(adaptive refinement)를 위해 입력 데이터를 트리 구조로 변환하는 모듈
시간가변 데이터 처리 모듈	시간가변 데이터를 실시간으로 가시화 하고 처리하는 모듈
데이터 가시화 모듈	입력 데이터 렌더링 및 사용자 인터페이스를 위한 모듈
적응형 세분화 모듈	방대한 데이터의 실시간 가시화를 위한 적응형 LOD 모듈
하드웨어 가속 모듈	GPU를 이용한 실시간 렌더링 모듈



(a) 클라이언트-서버 기반 가시화 시스템을 이용한 가시화 결과

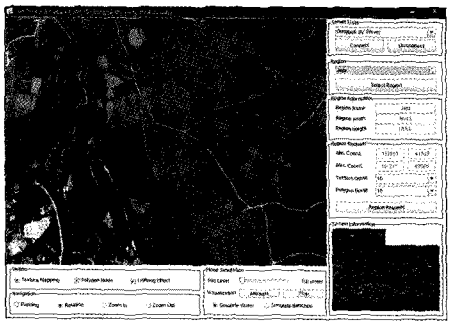
(b) 가시화된 결과 중 일부를 확대한 결과 영상

그림 6. 클라이언트-서버 기반 가시화 시스템을 이용한 가시화 결과

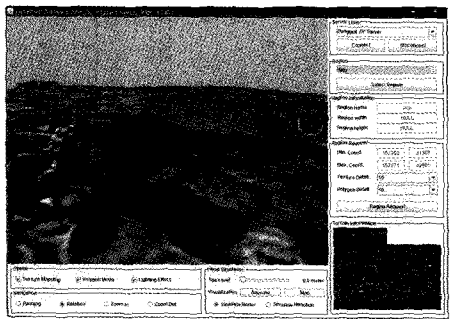
특정 지역에 대한 시간가변 침수 시뮬레이션 결과를 동시에 렌더링 할 수 있도록 구현되었다. 표 5는 가시화 대상으로 설정된 지형 데이터의 구성을 나타내는 것이다. 수치표고 자료를 가시화하기 위해서는 수치표고 자료 이외 정사영상, 지상좌표화 정보 모두를 이용하여야 한다.

실험에 사용된 데이터는 총 9개의 패치로 구성되며, 각 패치 마다 대응되는 수치표고 자료, 정사영상, 그리고 지상좌표화 정보를 포함하고 있다. 패치 하나의 해상도는 7,792×9,258이며, 약 200 메가바이트(mega byte)의 크기를 갖는다. 수치표고 자료는 측량 간격 5미터인 데이터를 사용한 경우, 하나의 패치를 가시화하기 위해 약 25만개 정도의 다각형이 필요하고, 보다 세밀히 관찰하기 위해 측량 간격 1미터인 데이터를 이용하면 약 550만개의 다각형이 필요하게 된다.

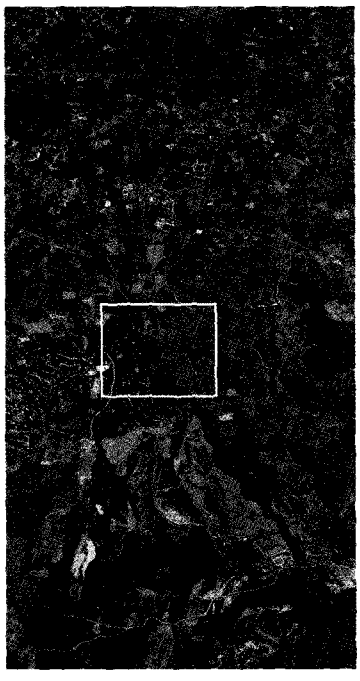
그림 7은 분산/병렬 가시화 시스템을 이용하여 9개의 패치로 구성된 지형 데이터를 대화형으로 가시화한 결과 영상이다. 그림 7의 (a)는 요청된 지역을



(a) 지역 탐색 및 요청 결과 영상



(b) 회전 및 줌-인을 적용한 결과 영상



(c) 미니맵 중 요청 지역만 확대한 영상

그림 7. 지형 데이터의 대화식 가시화 결과 영상

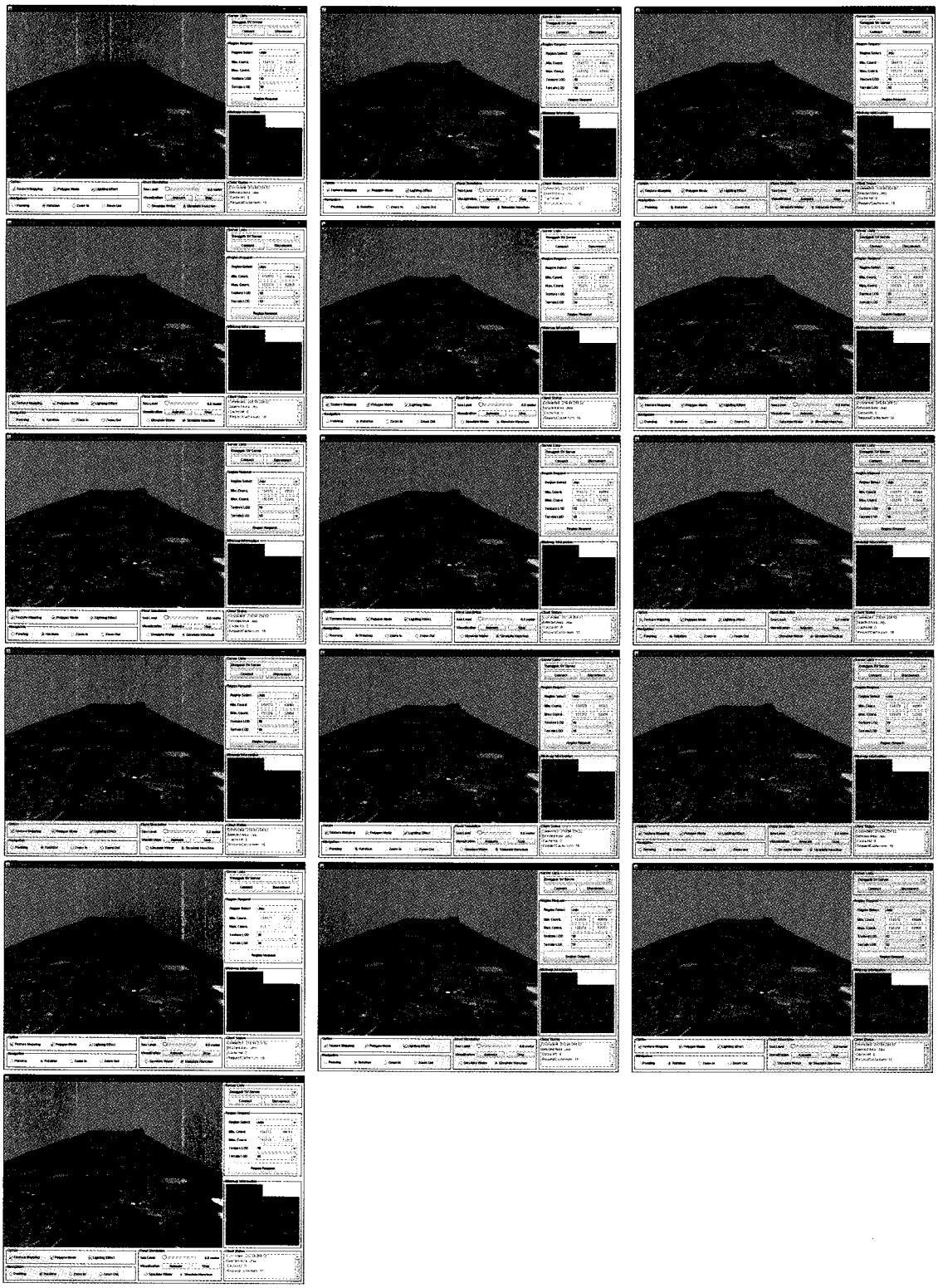


그림 8. 시간 변화에 따른 하천 유량 및 범람 지역의 가시화(좌에서 우, 위에서 아래)

표 5.지형 데이터의 구성 정보

구 분	설 명
수치표고 자료	2차원 지상 좌표와 그에 따른 표고와 측량 간격, 경계 정보 등이 저장. 횡단 메카토르(transverse mercator) 좌표계.
정사영상	항공사진 또는 위성사진으로 수치표고 자료에 대응
지상좌표화 정보	영상 좌표계를 지상좌표계로 사상시키기 위한 이동, 회전 그리고 크기 변환 등의 정보가 저장

위에서 내려다 본 결과 이고, (b)는 (a)의 결과를 회전하고 줌-인한 것이다. (c)는 (a)의 미니맵에 표시되는 붉은색 요청 지역을 확대한 영상으로 푸른색으로 표시된 영역은 요청 지역 중 현재 화면에 출력 되는 부분을 나타내는 것이다. 가시화 결과는 저성능 플랫폼 상에서 대화형 가시화가 가능하도록 약 60만개의 다각형을 이용하여 가시화 하였다. 시간가변 시뮬레이션 데이터를 이용하여 그림 8과 같이 시간의 흐름에 따라 침수되는 지역을 대화식으로 확인 할 수 있다. 시간가변 시뮬레이션 데이터와 지형 데이터를 중첩시켜 가시화하여 태풍 발생 시 하천의 흐름과 범람되는 지역을 효과적으로 표현할 수 있다. 이 실험에서는 클라이언트로 범용 PC를 이용하였다.

4. 실험결과 및 분석

본 연구에서 제안하는 시스템의 성능을 실험하기 위해 40대의 범용 PC를 클러스터로 구성하였다. 클러스터 구성에 사용된 PC는 Intel Core2 Duo 2.66 Ghz의 CPU와 2 GB의 메모리 그리고 ATI 3850 그래픽 카드가 탑재되었으며, 클러스터를 관리하는 마스터 서버 및 클라이언트 역시 동일한 사양의 PC를 이용하였다. 실험에 사용된 데이터는 총 9개의 패치로 구성되고, 각 패치 마다 대응되는 수치표고 자료와 정사 영상 그리고 지상좌표화 정보를 포함하고 있다. 각 수치표고 자료는 간격이 1미터인 2500×3000개의 높이 값을 가지고 있으며, 각 정사 영상의 해상도는 약 7792×9285의 크기를 가진다.

4.1 확장성 실험

그림 9는 클러스터를 구성하는 슬레이브 노드의 수와 슬레이브 노드의 캐시 크기 변화에 따른 처리시

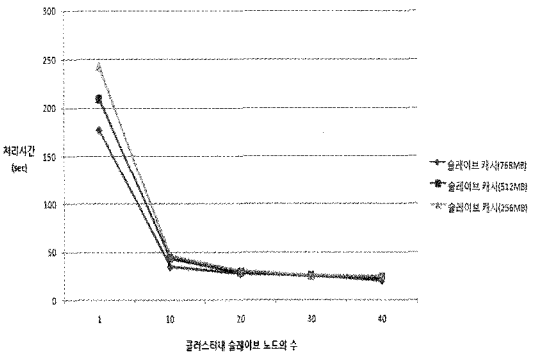


그림 9. 슬레이브 노드 추가에 따른 병렬 처리 시간(캐시 격자 해상도 128×128

간 변화를 나타낸 것으로, 슬레이브 노드의 수를 1대에서부터 40대 까지 증가 시키면서 측정하였다. 실험을 위해 사용된 지형은 패치 9개에 해당하는 4,611 ×13,875 (m^2) 영역이며, 수치표고 자료는 5미터 간격, 정사 영상은 원본 크기의 20%로 축소하도록 요청하였다. 지형의 LOD 적용 및 추출을 위한 슬레이브 캐시의 크기 변화에 따라 성능에 차이가 발생되지만 전반적으로 노드 수의 증가에 따라 처리 속도가 향상되는 것을 확인할 수 있다.

그림 10은 슬레이브 노드의 추가에 따른 성능 향상을 그래프로 나타낸 것으로 클러스터를 구성하는 노드의 수를 10대로 한 경우, 1대의 노드만을 이용하여 처리하는 것에 비해 약 4배 이상의 성능이 향상되는 것을 확인 할 수 있었다. 또한 클러스터를 구성하는 노드의 수를 증가한 경우 역시 성능이 점진적(progressive)으로 향상되는 것을 확인할 수 있었다. 하지만 일정 이상의 노드를 추가하는 경우 성능 향상 증가폭이 감소되는 것을 알 수 있다.

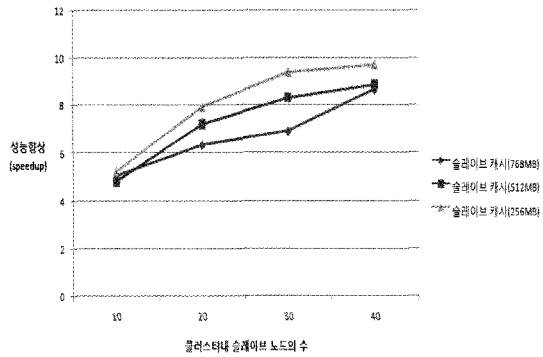


그림 10. 슬레이브 노드 추가에 따른 병렬 처리 성능(캐시 격자 해상도 128×128

4.2 병렬 처리 실험

그림 11은 캐시 블록의 크기 변화에 따른 병렬 처리의 성능을 나타내는 그래프로써, 실험 환경 및 사용 데이터는 4.1절의 확장성 실험과 동일하게 설정하였다. 캐시 중복에 따른 처리는 고려하지 않고, 특정 노드에 작업 부하가 걸리지 않게 전제 지역을 블록단위로 분할한 다음 요청된 지역에 해당하는 블록들을 병렬로 처리하게 된다. 캐시 블록의 크기에 따라 생성되는 태스크의 수는 표 6과 같으며, 클러스터를 구성하는 슬레이브 노드의 수가 증가함에 따라 전반적으로 성능이 향상되는 것을 확인할 수 있었다. 하지만 캐시 블록의 크기가 16×16인 경우 태스크의 수가 약 1만개 정도로 너무 많은 태스크의 요청이 발생된 경우 성능이 감소되는 것을 확인 수 있었다. 일반적으로 큰 영역을 요청하였을 경우 캐시 블록의 크기가 커질수록 처리 속도가 향상되지만, 작은 영역을 요청하는 경우 큰 캐시 블록의 사용이 메모리 낭비뿐만 아니라 잠재적인 병렬성 감소의 원인이 될 수 있다. 따라서 시스템 성능 향상을 위해서는 요청 되는 영역의 크기에 맞게 캐시 블록의 크기를 가변적으로 변경하여 병렬성을 극대화 하는 것이 중요하다고 할 수 있다.

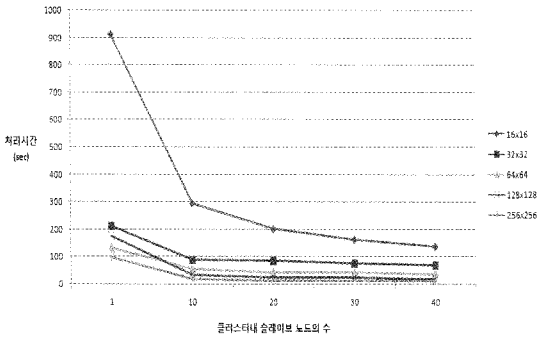


그림 11. 캐시 블록 크기 변화에 따른 병렬 처리 시간

표 6. 블록 크기 변화에 따른 생성 태스크 수

블록 크기	16×16	32×32	64×64	128×128	256×256
태스크 수	10092	2523	660	176	44

4.3 캐시 성능 실험

그림 12는 마스터 노드의 캐시 활용 비율에 따른 성능 향상 결과를 그래프로 나타낸 것이다. 캐시 성능 실험을 위해 그림 13과 같이 녹색으로 표시된 영역의 요청이 발생된 이후, 파란색으로 표시되는 영역이 추가로 요청되는 경우, 녹색과 파란색 영역의 중복 비율 변화에 따른 처리 시간을 측정하였다. 실험을 위해 전체 9개의 패치 중 2개의 패치에 해당하는 4,611×2,773 (m^2)의 크기의 지역을 요청하였으며, 수치표고 자료의 간격은 5미터, 정사 영상은 원본 크기의 50%로 축소하도록 요청하였다. 실험에서 캐시 블록의 크기는 64×64로 고정하였으며, 클러스터를 구성하는 슬레이브 노드는 40대로 설정하였다.

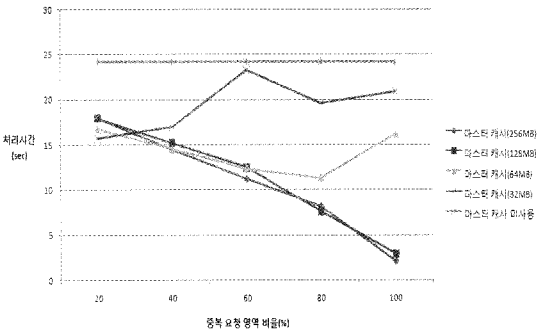


그림 12. 마스터 캐시 사용 및 중복 영역 비율에 따른 병렬 처리 시간의 변화(캐시 격자해상도 64×64)

마스터 캐시를 사용하지 않는 경우, 중복 요청 영역의 비율과 상관없이 병렬 처리를 위해 평균 24.198초의 데이터 처리 시간이 요구되는 것을 확인할 수 있었으며, 마스터 캐시를 이용하는 경우, 그림 11에서

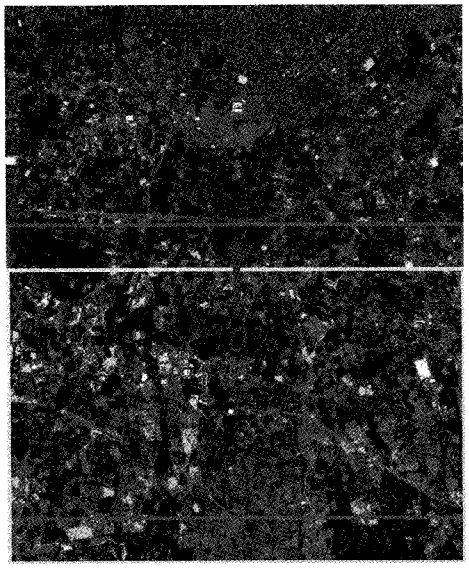


그림 13. 가시화 영역 이동에 따른 데이터 중복 비율

와 같이 처리 시간이 감소되는 것을 확인할 수 있다.

마스터 캐시의 크기를 128 MB 이상으로 설정한 경우 중복 요청 영역의 비율이 증가됨에 따라 처리시간이 점진적으로 감소되는 것을 확인할 수 있었지만, 마스터 캐시의 크기를 64 MB 이하로 설정한 경우 처리시간이 감소하다 오히려 증가되는 부분이 발생되는 것을 확인할 수 있었다. 이러한 문제는 크게 두 가지 원인으로 인해 발생될 수 있는데, 먼저 지역성을 가지는 데이터를 저장할 마스터 캐시의 크기가 충분하지 않은 경우 발생될 수 있다. 그림 12에서 마스터 캐시의 크기를 64 MB로 한 경우가 이에 해당되는데, 중복 영역이 80%인 경우 보다 오히려 100%인 경우가 더 많은 처리 시간이 요구되는 것을 알 수 있다. 이는 캐시에 저장되어 재활용 되는 데이터의 크기가 캐시보다 큰 경우, 캐시 처리 큐에 저장되는 순서에 따라 데이터를 삭제하기 때문에 캐시 미스가 발생되기 때문이다. 처리시간이 증가되는 두 번째 원인으로 요청되는 영역이 캐시 블록의 경계에 해당하는 경우 발생되는데, 이는 캐시 시스템 상 데이터를 블록 단위로 처리하기 때문에 경계에 해당하는 캐시 블록 모두를 처리하기 때문이다. 그림 12에서 마스터 캐시의 크기가 32 MB인 경우, 중복 요청 영역이 증가함에도 불구하고 처리 시간이 점진적으로 감소되지 않는 이유는 앞에서 언급한 두 가지 원인 모두에 해당 되는 경우이기 때문이다.

캐시 시스템은 지역성을 가지는 요청이 발생되는 경우 이전에 요청된 작업들을 캐시에 저장한 후 재활용 되며, 그림 14와 같이 마스터 캐시의 크기와 중복 요청 영역의 비율에 따라 캐시를 사용하지 않은 경우에 비해 약 1.3배에서 11배 성능 향상이 이루어지는 것을 확인할 수 있었다.

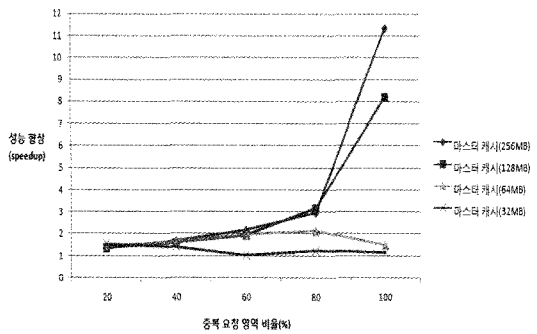


그림 14. 중복 영역 요청 비율에 따른 캐시 시스템의 성능 향상 변화(캐시 격자해상도 64×64)

이론적으로 중복된 가시화 영역이 많을수록 캐시 시스템의 성능이 향상되는 것이 당연한 하지만, 그림 14에서와 같이 성능 향상이 선형적으로 이루어지지 않는 것은 앞에서 언급한 바와 같이 캐시 미스와 경계에 해당하는 캐시 블록 모두를 처리하기 때문에 발생되는 문제이다. 하지만 병렬 처리시 캐시를 사용하지 않은 경우 보다 캐시를 사용한 경우, 중복 요청 영역의 비율에 따라 성능 향상이 이루어지는 것을 확인할 수 있었으며, 사용자의 데이터 액세스 패턴이 공간적 지역성을 나타내는 지형 데이터의 가시화 경우 이러한 캐시 자료구조의 유지를 통항 성능 향상을 기대할 수 있다.

4.4 노드별 부하 균형 실험

분산/병렬 시스템을 이용하는 경우 네트워크의 지연시간 및 클러스터를 구성하는 슬레이브 노드의 성능에 따라 특정 노드에 부하가 걸릴 수 있는데, 본 연구에서는 전체 지역을 블록 단위로 분할한 다음 요청된 지역에 해당하는 블록을 병렬로 처리하기 때문에 특정 노드의 부하를 최소화 할 수 있다.

본 연구에서 제안하는 시스템의 부하 균형(load balancing)이 적절히 이루어지는지를 확인하기 위해 4.1절의 실험과 동일한 환경과 데이터를 이용하여 노드별 작업 부하를 측정하였다. 아래 그림 15는 캐시 격자 변화에 따라 클러스터내 각 노드들의 처리 시간을 나타낸 것으로 캐시 중복에 따른 처리는 고려하지 않았다. 패치 9개에 해당하는 4,611×13,875 (m^2) 영역을 캐시 격자 크기마다 동일하게 요청하였으며, 수치 표고 자료는 5미터 간격, 정사 영상은 원본 크기의 20%로 축소하도록 요청하였다.

그림 15에서와 같이 전체 지역을 분할하는 격자의 크기가 큰 경우 작업 처리 시간이 고르지 못하게 나타나는데 이는 네트워크 전송 지연과 클러스터내 각 노드의 성능 차이로 인해 발생되는 것으로, 블록의 크기를 64×64로 한 경우 부하 균형이 적절히 이루어지는 것을 확인할 수 있다. 그림 16은 블록의 크기가 64×64인 경우 클러스터를 구성하는 각 슬레이브 노드의 작업량을 나타내는 것으로 각 슬레이브 노드의 성능과 네트워크 지연 시간 등의 문제로 인해 특정 노드에 작업이 편중되는 것을 확인할 수 있다. 하지만 그림 17과 같이 작업을 처리하는 각 슬레이브 노드들이 작업량과 관계없이 고른 처리시간을 가지는

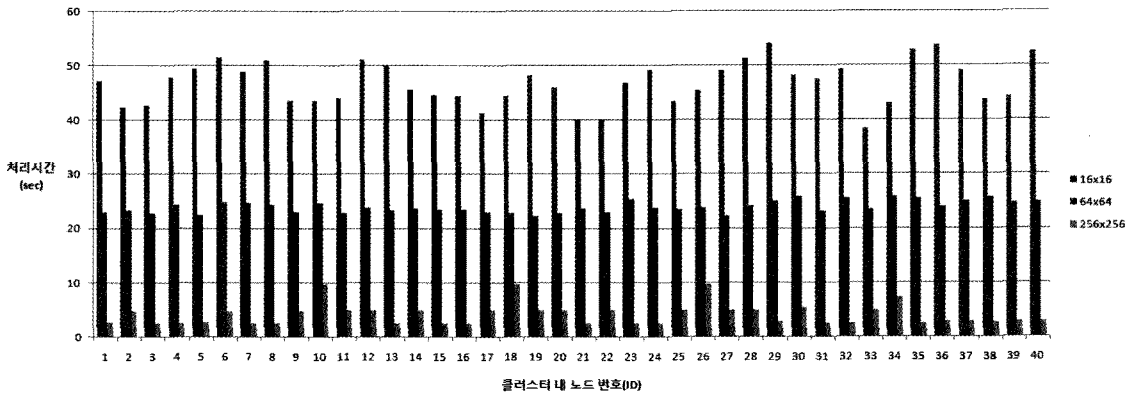


그림 15. 클러스터 내 각 슬레이브 노드의 작업 처리 시간

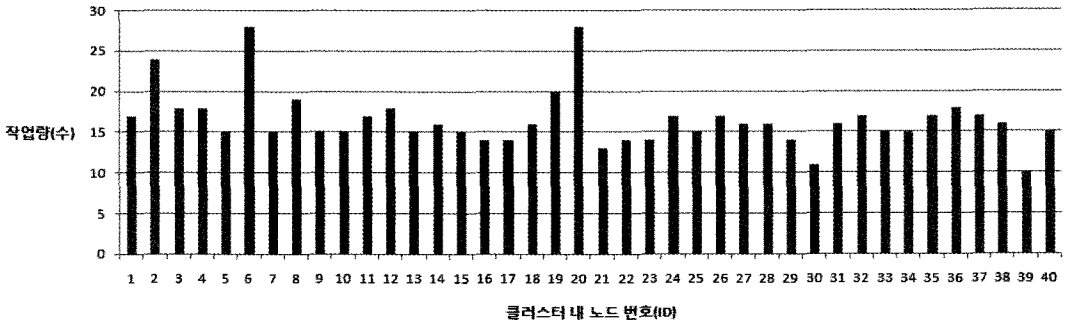


그림 16. 클러스터 내 노드별 작업량(블록크기 64×64)

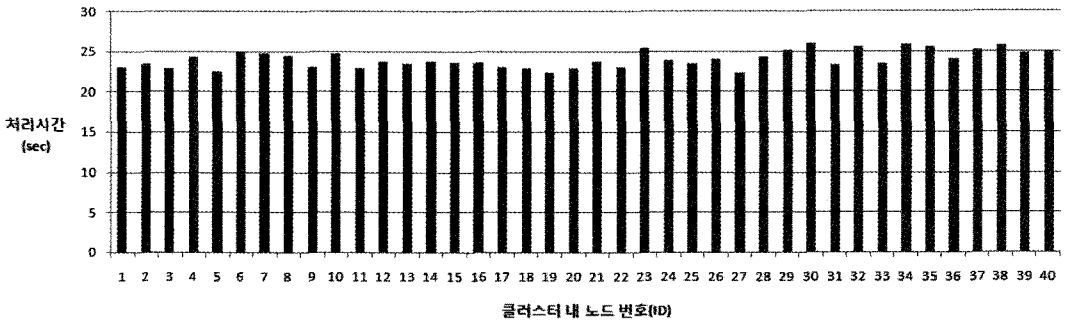


그림 17. 클러스터 내 노드별 작업 처리시간(블록크기 64×64)

것을 확인할 수 있었다.

5. 결론 및 향후 연구

인공위성이나 항공기를 이용한 촬영 기법이 발전하면서 다양한 형태의 지형 데이터들이 생성되고 있고, 최근 몇 년간 이와 같은 데이터들을 이용한 다양한 응용 프로그램들이 실생활에서 활용되고 있다. 하지만 고해상도의 지형 데이터의 경우 상당히 방대한 크기를 갖기 때문에 사용자의 요청에 따라 특정 지역의 지형 정보를 빠르게 액세스하여 전달하고 이를 가시화하는 효과적인 시스템의 개발이 필요하다. 또한 일반 사용자들이 다양한 기종의 클라이언트 시스템을 이용해 데이터를 액세스하고 이를 원하는 형태로 가시화 할 수 있는 환경을 제공하기 위한 GUI 프로그램의 개발도 요구된다.

본 논문에서는 방대한 지형 데이터 가시화를 위한 클라이언스-서버 기반의 분산/병렬 시스템을 제안

하였다. 이 시스템은 웹 기반으로 수행되는 클라이언트 GUI 프로그램과 복수의 PC 클러스터에서 구동되는 병렬 서버 프로그램으로 구성된다. PC 뿐만 아니라 자바를 지원하는 모바일 기기에서도 클라이언트 프로그램이 수행될 수 있도록 자바 기반의 그래픽스 라이브러리인 JOGL을 사용하여 GUI를 구현하였으며, 사용하는 기기의 현재 사용 가능한 메모리 크기와 디스플레이 가능한 화면의 최대 해상도 정보를 서버에게 전달하여 서버에서 수행해야 하는 작업의 양과 불필요한 통신비용을 줄이고 최소의 비용으로 가시화 작업을 수행하도록 구현하였다. 서버로 사용된 PC 클러스터는 분산된 지형 데이터를 액세스 하고 이를 클라이언트로부터 받은 정보에 따라 적절히 리샘플링 한 후, 이를 클라이언트에게 전달하는 작업을 담당한다. 서버의 각 노드뿐만 아니라 클라이언트까지도 캐시 자료구조를 유지하도록 설계함으로써 분산된 데이터 접근 시 발생되는 지연 시간과 중복된 영역을 가시화 할 때 발생되는 비용을 최소화하도록 하였다.

현재 실험 환경의 제약으로 분산/병렬 처리에 대한 성능 분석이 제한된 수의 PC 상에서 수행되었으나 추후 더 많은 노드로 구성된 클러스터를 이용하여 테스트를 수행하고 성능을 개선할 예정이다. 또한 현재의 시스템의 기능을 더욱 확장하여 GPU가 탑재되어 있지 않은 모바일 기기에서도 효과적으로 태풍, 홍수와 같은 대규모 시간가변 볼륨 데이터의 대화형 가시화가 가능하도록 성능을 향상시킬 계획이다.

참 고 문 헌

[1] GPGPU, http://gpgpu.org, 2010.

[2] D. Weiskopf, *GPU-Based Interactive Visualization Techniques*, Springer, 2006.

[3] J. Kruger and R. Westermann, "Acceleration Techniques for GPU-based Volume Rendering," In Proceedings of IEEE Visualization, 2003.

[4] B. Wylie, C. Pavlakos, V. Lewis, and K. Morelan, "Scalable Rendering on PC Clusters," Proceeding on IEEE Computer Graphics and Applications, Vol.21, pp. 62-70, 2001.

[5] 이원종, 박우찬, 한탁돈, "계층 자료구조의 결합과 3차원 클러스터링을 이용하여 적응적으로 부하 균형된 GPU-클러스터 기반 병렬 볼륨 렌더링," 정보과학회논문지: 시스템 및 이론, 33권, 1·2호, pp. 1-14, 2006.

[6] S. Callahan, M. Ikits, J. Comba, and C. Silva, "Hardware-Assisted Visibility Sorting for Unstructured Volume Rendering," *IEEE Transactions on Visualization and Computer Graphics*, pp. 285-295, 2005.

[7] J. Comba, J. Klosowski, N. Max, J. Mitchell, C. Silva, and P. Williams, "Fast Polyhedral Cell Sorting for Interactive Rendering of Unstructured Grids," In Computer Graphics Forum (Proc. EUROGRAPHICS '99), 1999.

[8] R. Cook, N. Max, C. T. Silva, and P. Williams, "Efficient, Exact Cisibility Ordering of Unstructured Meshes," *IEEE Transactions on Visualization and Computer Graphics*, Vol.10, No.6, pp. 695-707, 2004.

[9] H. Vo, S. Callahan, N. Smith, C. Silva, and W. Martin. "iRun: Interactive Rendering of Large Unstructured Grids," Eurographics Symposium on Parallel Graphics and Visualization, 2007.

[10] Z. Fan, F. Qiu, A. Kaufman, and S. Yoakum-Stover, "GPU Cluster for High Performance Computing," Proceedings of ACM/IEEE conference on Supercomputing, 2004.

[11] 박우찬, 이원종, 김형래, "확장형 디스플레이를 위한 분산 렌더링 시스템의 네트워크 대역폭 감소 기법," 정보과학회논문지: 시스템 및 이론, 29권, 10호, pp. 582-588, 2002.

[12] High Performance Communication, hhttp://www. csag.cs.uiuc.edu/projects/communication.html, 2010.

[13] T. Mattson, B. Sanders, and B. Massingill, *Patterns for Parallel Programming*, Addison-Wesley, 2004.

[14] K. Nishihashi, T. Higaki, K. Okabe, B. Raytchev, T. Tamaki, and K. Kaneda, "Volume Rendering Using Grid Computing for Large-Scale Volume Data," *International Journal of CAD/*

CAM, Vol.9, No.1, pp. 111-120, 2009.

[15] 김영수, 김명호, 최홍문, "이더넷 다중 클러스터에서 GHT의 병렬 분산 구현," 한국전자공학회 논문지, 46권, 3호, pp. 96-106, 2009.

[16] 이미영, "클라우드 기반 대규모 데이터 처리 및 관리 기술," 한국전자통신연구원, 전자통신동향분석, 24권, 4호, pp. 41-54, 2009.

[17] 신승선, 백성하, 어상훈, 이동욱, 김경배, 정원일, 배해열, "그리드 데이터베이스에서 질의 처리를 위한 캐쉬 관리 기반의 부하분산 기법," 멀티미디어학회논문지, 11권, 7호, pp. 914-927, 2008.

[18] S. Dragićević, "The Potential of Web-based GIS," *Journal of Geographical Systems*, Vol. 6, Issue 2, pp. 79-81, 2004.

[19] R. Kingston, S. Carver, A. Evans, and I. Turton, "Web-based Public Participation Geographical Information Systems: An Aid to Local Environmental Decision-making," *Computers, Environment and Urban Systems*, Vol.24, Issue 2, 31, pp. 109-125, 2000.

[20] C. Yang, D. Wong, R. Yang, M. Kafatos, and Q. Li, "Performance-Improving Techniques in Web-Based GIS," *International Journal of Geographical Information Science*, Vol.19, No. 3, pp. 319-342, 2005.

[21] 정무경, 박성모, 엄낙용, "병렬 프로세서 기술 및 동향," 한국전자통신연구원, 전자통신동향분석, 24권, 6호, pp. 86-93, 2009.

황 규 현

2006년 2월 대구대학교 멀티미디어공학과 졸업(공학사)
2008년 2월 동국대학교 멀티미디어학과 졸업(공학석사)
2008년 3월~현재 동국대학교 멀티미디어학과 박사과정

관심분야 : 컴퓨터 그래픽스, 모바일 3D 그래픽스, 가상현실 등

윤 성 민

2009년 2월 동국대학교 멀티미디어공학과 졸업(공학사)
2011년 2월 동국대학교 멀티미디어학과 졸업(공학석사)
관심분야 : 컴퓨터 그래픽스, 과학적 가시화, 고성능 컴퓨팅 등

박 상 훈

1993년 8월 서강대학교 수학과 졸업(이학사)
1995년 8월 서강대학교 컴퓨터학과 졸업(공학석사)
2000년 2월 서강대학교 컴퓨터학과 졸업(공학박사)
2000년 3월~2000년 6월 서강대학교 컴퓨터학과 박사후연구원
2000년 7월~2002년 8월 University of Texas at Austin 박사후연구원
2002년 9월~2005년 2월 대구가톨릭대학교 컴퓨터정보통신공학부 조교수
2005년 3월~현재 동국대학교 영상대학원 멀티미디어학과 부교수
관심분야 : 컴퓨터 그래픽스, 과학적 가시화, 가상현실, 모바일 컴퓨팅, 컴퓨터 게임 등