확장형 디스플레이를 위한 분산 렌더링 시스템의 네트워크 대역폭 감소 기법

(A New Network Bandwidth Reduction Method of Distributed Rendering System for Scalable Display)

박 우 찬 † 이 원 종 †† 김 형 래 ††† 김 정 우 †††
(Woo-Chan Park) (Won-Jong Lee) (Hyung-Rae Kim) (Jung-Woo Kim)
한 탁 돈 †††† 양 성 봉 †††††
(Tack-Don Han) (Sung-Bong Yang)

요 약 확장형 디스플레이(Scalable Display)는 큰 화면과 높은 화질의 영상을 생성하여 사용자들에게 보다 현실적인 느낌을 주고 이에 몰입할 수 있는 환경을 제공하는 시스템이다. 최근 들어서 이러한 확장형 디스플레이는 자체 그래픽 가속기와 메모리, CPU, 저장장치를 갖는 개별 PC들을 네트워크로 연결한 클러스터 환경에서 구축되고 있다. 하지만 클러스터 환경에서 분산 렌더링을 수행하면 제한된 대역폭 때문에 네트워크 병목점을 갖게 된다. 본 논문은 이러한 네트워크 트래픽을 줄이는 새로운 알고리즘을 제안하고, 이를 기존의 분산 렌더링 시스템에 적용하여 구현한 내용을 소개한다. 제안하는 기하 추적(geometry tracking) 알고리즘은 전송되는 데이터들을 색인화하여 중복된 기하정보 전송을 방지하여 네트워크 부하를 줄이는 방법으로, 실험을 통해서 최대 42%까지 네트워크 트래픽을 감소시킬 수 있었다.

키워드 : 3차원 그래픽스, 분산 렌더링, 병렬 렌더링, PC 클러스터

Abstract Scalable displays generate large and high resolution images and provide an immersive environment. Recently, scalable displays are built on the networked clusters of PCs, each of which has a fast graphics accelerator, memory, CPU, and storage. However, the distributed rendering on clusters is a network bound work because of limited network bandwidth. In this paper, we present a new algorithm for reducing the network bandwidth and implement it with a conventional distributed rendering system. This paper describes the algorithm called geometry tracking that avoids the redundant geometry transmission by indexing geometry data. The experimental results show that our algorithm reduces the network bandwidth up to 42%.

Key words : 3D Graphics, Distributed Rendering, Parallel Rendering, PC Cluster

· 이 논문은 2001년도 한국학술진흥재단의 지원에 의하여 연구 되었음 (KRF-2001- 003-E00264).

† 정 회 원 : 연세대학교 연구교수
chan@kurene.yonsei.ac.kr
†† 비 회 원 : 연세대학교 대학원 컴퓨터학과
airtight@kurene.yonsei.ac.kr
††† 학생회원 : 연세대학교 대학원 컴퓨터학과
kimhr@cs.yonsei.ac.kr
mosh@kurene.yonsei.ac.kr
†††† 종신회원 : 연세대학교 공과대학 컴퓨터과학과 교수
hantack@kurene.yonsei.ac.kr
††††† 비 회 원 : 연세대학교 공과대학 컴퓨터과학과 교수
yang@mythos.yonsei.ac.kr

논문접수 : 2001년 1월 10일
심사완료 : 2002년 7월 20일

1. 서 론

최근까지 3차원 그래픽스 분야에서는 현실적인 영상을 만들기 위해 기하학 모델 데이터들의 정확도를 증가시켜왔고, 그에 따라 데이터의 크기가 양적으로 증가되어 왔다[1]. 따라서 이러한 높은 화질의 영상을 보다 빠른 프레임비율로 생성하는 그래픽 가속기를 설계하기 위해 병렬처리 기술이 적용되어 왔고, 이에 관련한 다양한 방법들이 연구되었다. 특히, 최근 들어 이러한 병렬 렌더링의 중요 응용분야로, 출력 디스플레이를 다수 개 연결하여 큰 화면의 높은 화질 영상을 확장가능하게 생성하고 보여줄 수 있는 확장형 디스플레이(scalable

display)에 대한 연구가 각광을 받고 있는 추세이다.

초정밀 과학 영상화, 비행 시뮬레이션, CAD, 의료영상 가시화 등 고화질 영상의 세밀함이 중요시되는 어플리케이션들은 해상도의 한계 때문에 일반 데스크탑 수준의 디스플레이에서 영상을 출력하기가 어렵다. 따라서 이러한 제한된 해상도 문제를 해결하기위해 출력 영상의 크기를 늘리게 되었고, 이를 위해 단순히 화소의 크기를 확장하는 것이 아니라 해상도 자체의 확장으로 높은 해상도를 얻는 다양한 시도가 이루어졌다.

이러한 확장형 디스플레이를 위한 병렬 렌더링 시스템을 구현하는 방법은 크게 두 가지가 있는데, 첫째는, 고성능의 그래픽 연산능력을 가지고 있는 병렬 워크스테이션을 이용하는 밀결합 형태의 방법이다. Power Wall[2], Interactive Mural[3], InfinityWall[4] 등이 대표적인 시스템이다. 하지만 이 방법은 하드웨어 제작에 따른 매우 많은 비용과 시간이 요구되며, 디스플레이의 확장성이 떨어지는 단점을 갖게 된다.

두 번째는, 그래픽 가속기, 메모리, CPU, 저장장치가 장착된 각 PC들을 고속 네트워크로 연결한 클러스터를 이용하는 약결합 형태의 방법으로, 병렬 워크스테이션에 비해 몇 가지 장점이 있다. 컴퓨터간의 통신은 네트워크를 통해 이루어지고 각 노드사이에 프로토콜을 이용하여 제어하기 때문에 유연하게 새로운 PC를 추가, 제거할 수 있고, 각각의 연결된 PC마다 자체의 CPU, 메모리, 그래픽 가속기를 제어하는 버스를 가지고 있기 때문에 PC의 수를 늘릴수록 클러스터의 전체 하드웨어 계산 성능, 메모리 크기, 버스 대역폭을 선형적으로 증가시킬 수 있다. 또한 각 PC의 일반 하드웨어를 이용하기 때문에 부분 하드웨어 교체가 빈번히 이뤄질 수 있고, 따라서 상대적으로 싼 가격으로 제작할 수 있으며 짧은 기간 내에 지속적인 성능향상이 이루어 질 수 있다는 것이다.

하지만 병렬 워크스테이션과 달리 PC 클러스터는 고속 접근이 가능한 가상 공유메모리 공간을 갖고 있지 않고, 네트워크를 통해 프로세서 사이의 통신이 이루어지기 때문에 대역폭이 상대적으로 제약을 받게 된다. 따라서 PC 클러스터를 이용할 때 가장 중요한 것은 연산능력, 저장장치, 통신 특성을 잘 이용하여 확장성을 높일 수 있는 분산 렌더링 알고리즘을 개발하는 것이다[5][6].

AT&T InfoLab[7], Princeton[8], Stanford[9]에서는 클러스터를 이용하여 분산 렌더링 시스템을 구축하였고, 이를 위해 다양한 알고리즘을 구현하였다. Princeton에서는 렌더링과 영상 합성의 부하조정을 위해 전정렬(sort-first)와 후정렬(sort-last) 분할 방법을 결합시킨 새로운 방법을 제안하였다[10]. Stanford에서는 WireGL[9][11]이라는 원격 렌더링 소프트웨어 시스템을 구현하였는데, 이는 클러스터의 각 노드마다 표준 3차원 그래픽 API인 OpenGL[12]과 유사한 그래픽 드라이버를 제공하여 기존의 OpenGL 어플리케이션들을 소스 코드의 수정 없이 렌더링 될 수 있도록 하였다.

하지만, OpenGL과 같은 즉시 모드(immediate mode)의 그래픽 API는 기하 데이터(geometry data)들을 각각의 함수 호출로 기술하기 때문에, 복잡한 영상의 기하학 정보를 표현하는 어플리케이션의 경우 프레임당 수백만 번 이상의 함수 호출이 일어나게 된다. 이렇게 발생된 함수 호출들과 생성된 기하 데이터들은 모두 네트워크를 통해 전송되어야만 하고, 이때 제한된 네트워크 대역폭으로 데이터들을 전송하기 위한 부하가 데이터들이 렌더링 서버들에서 연산을 수행하는 부하보다 더 크기 때문에 병목점으로 작용하게 된다. 따라서 전송되는 데이터들의 양을 최소화하기 위한 효과적인 전송 알고리즘 설계가 요구된다.

본 논문에서는 분산 렌더링 시스템의 네트워크 트래픽을 줄이는 새로운 방법을 제안하였다. 네트워크로 전송되는 데이터들 중 가장 큰 비율을 차지하는 기하 데이터들의 패턴을 분석하여, 동일한 기하 데이터들이 중복적으로 발생하는 특성을 알아내었고, 또한 이러한 데이터들을 색인화하여 중복 생성되는 데이터들을 다시 전송하지 않도록 하는 알고리즘인 기하 추적(geometry tracking) 기법을 제안하였다. 제안하는 알고리즘은 WireGL의 일부 코드를 수정하여 구현되었고, *SPECViewperf*[13]와 *Quake III* 벤치마크를 이용한 실험을 통해 최대 42%의 네트워크 대역폭을 줄일 수 있음을 알 수 있었다.

본 논문의 구성은 다음과 같다. 2장에서 WireGL의 데이터 전송 방법을 살펴보고, 벤치마크들의 기하 데이터의 패턴을 분석하여 중복 발생비율을 측정한다. 3장에서는 제안하는 기하 추적 기법을 설명하고, 4장에서 실험을 통해 수행된 성능평가 결과를 기술한다. 마지막으로 5장에서 결론을 맺는다.

2. 기하학적 중복(Geometric Redundancy)

3D 그래픽 어플리케이션에서 모델의 대부분은 기하 데이터들로 기술된다. 본 절에서는 WireGL을 통해 분산 렌더링 환경의 데이터의 전송방법을 살펴보고, 그 중복 발생 비율을 측정하기 위한 기하 데이터의 패턴을 분석한다.

2.1 WireGL의 데이터 전송방법

WireGL은 그래픽 드라이버 형태로 구현되어, 클라이

언트측에서 구동된 어플리케이션의 함수 호출을 가로채고, 어플리케이션의 명령어와 데이터를 분류하여 각각의 영역을 전담하는 분산 렌더링 서버들로 전송하게 된다. 이때 클라이언트와 분산 서버들 각각에 버퍼를 두게 되는데, 각 버퍼들은 명령어와 데이터를 별도로 분리하여 저장한다. 클라이언트측의 버퍼인 기하 버퍼는 꽉 차게 되거나 상태(state)가 변경되는 명령어가 발생할 때 플러쉬되고, 이때 버퍼에 저장된 명령어와 데이터들이 한 패킷에 묶여 전송된다. 전송된 패킷은 서버측의 수신 버퍼에 저장되어 렌더링이 수행된다. 즉, WireGL을 통한 클라이언트-서버들 사이의 네트워크 전송단위는 클라이언트의 기하 버퍼와 동일한 크기를 갖는 하나의 패킷이 된다.

2.2 기하 데이터의 중복성

네트워크 트래픽을 줄이기 위한 방법을 찾기 위하여, 클라이언트에서 서버로 전송된 모든 패킷들에 포함된 데이터들의 패턴을 분석하였다. [그림 1]은 OpenGL 성능 평가 벤치마크인 SPECViewperf의 4개의 모델을 렌더링 하였을 때와, QuakeIII 게임 데모를 렌더링하였을 때, 서버로 전송된 총 데이터들을 프로파일링 한 결과이다. [그림 1]에서 볼 수 있듯이 OpenGL 명령어들 중 기하 데이터를 기술하는 기하 명령어(glVertex*, glNormal*, glTexCoord*)에 의해 생성되는 데이터의 비율이 전체의 45%~95%정도를 차지하였고, 이는 다른 명령어에 의해 생성된 데이터들보다 월등히 큰 것을 알 수 있다.

OpenGL에서 정점 데이터에 대한 정보들은 glBegin과 glEnd 명령어 사이에서 기하 명령어들로 기술된다. 정점과 경계정보를 공유할 수 있도록 메쉬(mesh) 형태로 기술된 모델이라 할지라도 WireGL에서는 기하 명령어들은 한 쌍의 glBegin()/glEnd() 기하 명령어군이 아닌 네트워크 패킷 하나를 기본 단위로 전송하기 때문에 정점 정보의 중복이 발생할 수 있게 된다. 한 패킷 안에 여러 쌍의 glBegin/glEnd() 기하 명령어군들이 존재할 수 있기 때문이다.

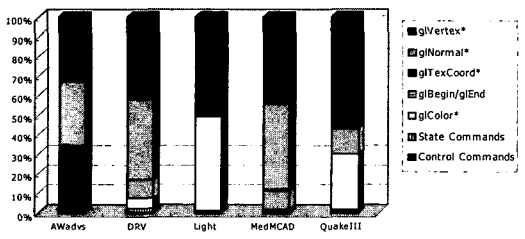


그림 1 OpenGL 어플리케이션을 렌더링하였을 때 기하 명령어와 기타 명령어에 의해 생성된 데이터들의 크기 비교 결과

[표 1]은 클라이언트의 기하 버퍼에서 플러쉬될 때마다 각 전송 패킷 내에서 기하 명령이 중복적으로 발생되는 비율의 평균값이다. [표 1]에서 볼 수 있듯이 WireGL의 전송 패킷내의 기하 데이터들은 평균 35%에서 61%까지의 중복율을 가지고 있다는 것을 알 수 있다. 따라서 이러한 중복되는 데이터가 발생하였을 때 이것을 다시 전송하지 않고 이미 전송된 데이터를 다시 사용한다면 전체적인 네트워크 대역폭을 줄일 수 있음을 알 수가 있다.

표 1 WireGL 네트워크 패킷들 내부에서 발생되는 OpenGL 기하 명령어들의 중복 발생 비율의 평균값

벤치마크	프레임수	총 정점의 수	총 중복된 정점의 수	중복율 (%)
AWadvs-04	600	44,013,480	26,000,664	59.07
DRV-07	370	87,499,104	46,790,684	53.48
Light-04	100	62,312,104	38,305,636	61.47
MedMCAD-01	723	1,155,849	403,092	34.87
Quake III	1,346	13,126,642	6,027,431	45.91

3. 제안하는 기하 추적 기법

본 절에서는 분산 렌더링 환경에서의 네트워크 트래픽을 감소시키기 위해 제안하는 새로운 방법인 기하 추적 알고리즘을 소개한다. 또한 WireGL를 수정하여 이를 구현한 내용과 그 동작을 기술한다.

3.1 기하 추적 알고리즘

제안하는 기하 추적 알고리즘은 한 패킷 내의 모든 기하 데이터들에 대한 색인화(indexing)와 추적(tracking) 작업을 수행한다. 즉, 기하 명령어의 같은 매개변수 값이 중복적으로 발생할 때 정점의 기하 데이터를 다시 전송하지 않고, 단지 색인화된 기하 데이터의 색인값을 추적한 다음 이를 전송하여 전체적인 데이터 전송량을 줄이는 방법이다.

중복되는 기하 명령에 대한 색인값을 전송하고 렌더링 서버에서 중복 기하 명령을 검출하기 위하여 다음 세 가지 종류의 새로운 명령어들을 정의하였다. glIndex_Vertex*, glIndex_Normal*, glIndex_TexCoord*. 이 명령어들은 클라이언트와 서버사이에서만 사용되며, 구동되는 어플리케이션에는 감추어지게 된다. [그림 2]는 클라이언트와 서버측에서의 기하 추적 알고리즘의 흐름도를 보여준다.

3.2 WireGL을 이용한 기하 추적 알고리즘의 구현

제안하는 알고리즘은 WireGL 코드를 수정하여 구현되었다. 클라이언트에서는, 구동된 어플리케이션이 OpenGL

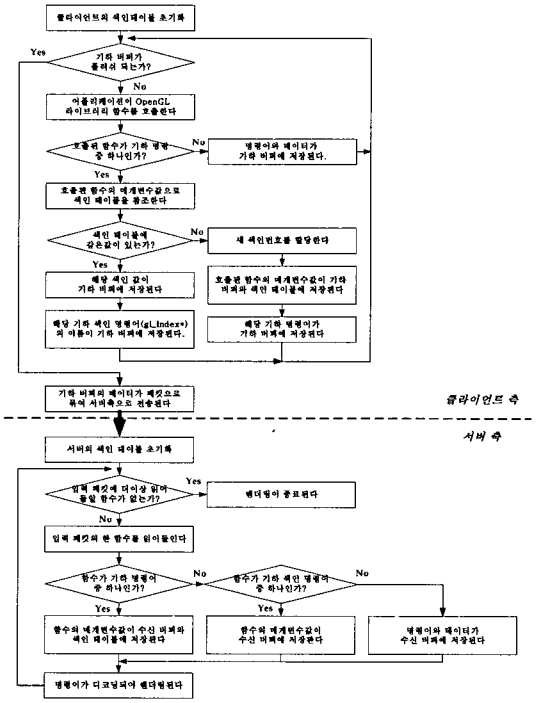


그림 2 기하 추적 알고리즘의 서버, 클라이언트에서의 처리 흐름도

라이브러리 함수를 호출하게 되고, WireGL은 어플리케이션의 함수 호출을 가로채게 된다. 이때 호출된 함수가 기하 명령어(glVertex*, glNormal*, glTexCoord*) 중 하나라면, 해당 함수의 매개변수 값을 읽어 색인 테이블을 참조한다. 이 값이 색인 테이블에 있다면 이는 이전에 한 번 이상 발생되었던 중복되는 함수 호출을 의미하므로 함수 매개변수 값 대신, 색인 테이블의 해당 색인 값을 기하 버퍼에 저장하고, 새롭게 정의한 색인 명령어군(glIndex_*)의 이름으로 기하 버퍼에 인코딩되어 저장된다. 만일 호출된 함수의 매개변수 값이 색인 테이블에 존재하지 않는다면, 아직 중복된 함수 호출이 일어나지 않은 경우이므로 기하 버퍼에 함수 매개변수 값을 저장하고, 이 후 발생할 중복 호출 명령어의 추적 작업을 위해, 새로운 색인 값을 할당받은 후 색인 테이블에 복사하게 된다. 이런 방법으로 기하 버퍼가 플러쉬될 때까지 반복하여 버퍼에 기하 데이터와 색인 값들을 저장한 후, 분산 렌더링 서버들로 전송하게 된다.

서버에서는, 클라이언트에서 전송되어 온 패킷의 명령어와 데이터를 순서대로 읽게 된다. 읽어 들인 함수가 기하 명령어들 중 하나라면, 해당 함수의 매개변수들은

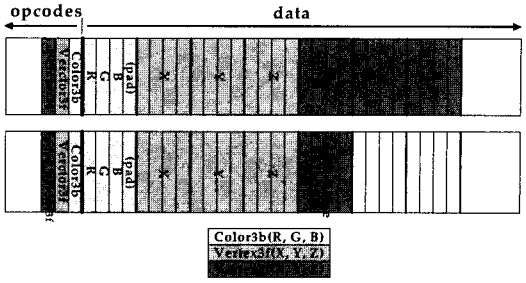


그림 3 WireGL의 기하 버퍼와 제안하는 기하 추적 버퍼의 크기 비교

중복되지 않은 명령어라는 것을 의미하므로 직접 서버의 색인 테이블과 서버의 버퍼인 수신 버퍼에 저장된다. 만일 패킷에서 읽혀진 함수가 색인 기하 명령어들(glIndex_*) 중 하나라면 이는 중복되었던 명령어라는 것을 의미하므로, 해당 색인 값을 이용하여 서버의 색인 테이블을 참조한 후 해당 함수의 매개변수 값을 읽어 서버의 수신 버퍼에 저장하게 된다. 최종적으로 버퍼내의 명령어들은 디코딩되어 렌더링을 수행하게 된다.

[그림 3]은 WireGL의 기하 버퍼와 제안된 기하 추적 방법을 사용했을 때의 기하 버퍼들의 구조를 비교하여 보여준다. [그림 3]의 버퍼들 내부의 사각형들은 각 1바이트를 의미한다. 클라이언트의 어플리케이션에서 기하 명령이 호출되면 각각의 데이터들을 기하 버퍼에 담아 전송하게 되는데, 예들 들면, glVertex3f 함수가 호출되면 명령어로 1바이트 그리고 부동 소수점인 세 정점에 대한 데이터로 12바이트가 사용된다. 이 때 만일 중복되는 정점에 대한 기하 명령이 발생한다면, 해당 정점을 색인 테이블에 보관한 후 추적하여 색인 값만을 전송하게 되고, 명령어로 1바이트, 정수형 색인 값으로 4바이트만을 사용하여 상대적인 전송 데이터의 크기를 줄일 수 있게 된다. 이때 정수형 색인 값으로 2바이트만을 사용할 수 있지만, 빠른 메모리 접근을 위해 2바이트대신 4바이트를 사용하였다.

[그림 4]는 기하 추적 방법을 사용하여 클라이언트와 분산 서버들 간의 데이터 전송하는 도중의 한 장면이다. 이는 클라이언트측의 어플리케이션 소스 코드 25번까지 수행한 후의 모습이다. 기하 버퍼는 꽉 찼을 때뿐만 아니라, 상태가 변경되었을 때에도 플러쉬 되는데, 클라이언트측에서, 15번에서 색상값이 2번과 비교하여 변경되었기 때문에 기하 버퍼가 플러쉬되고 그 때까지 저장되었던, 1번부터 13번까지의 명령어들에 의해 생성된 데이터들이 서버로 전송되었다. 서버측에서는, 수신 버퍼의 기하 데이터를 순서대로 읽어 서버의 색인 테이블을 재구

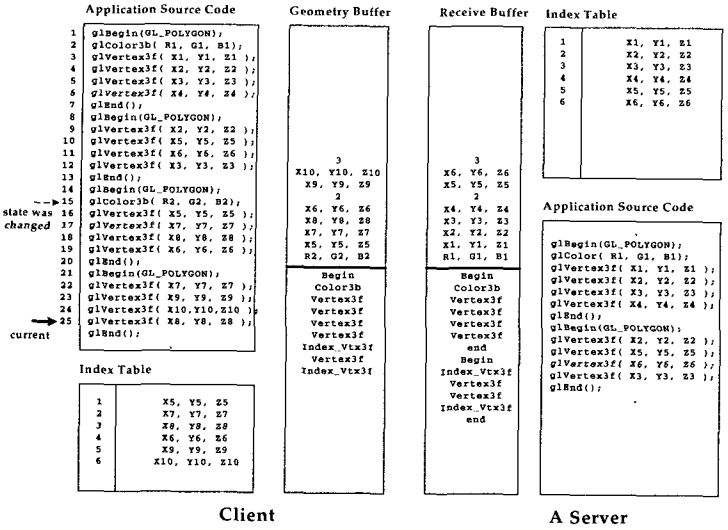


그림 4 제안하는 기하 추적 방법을 사용한 데이터 전송 시의 한 장면

(a) DRV-07 (b) Atlantis (c) Quake III

그림 5 실험에 사용된 벤치마크들

성하고, 색인 테이블과 수신 버퍼의 명령어들을 함께 읽은 후 디코딩하여 어플리케이션 소스 코드를 생성시킨다.

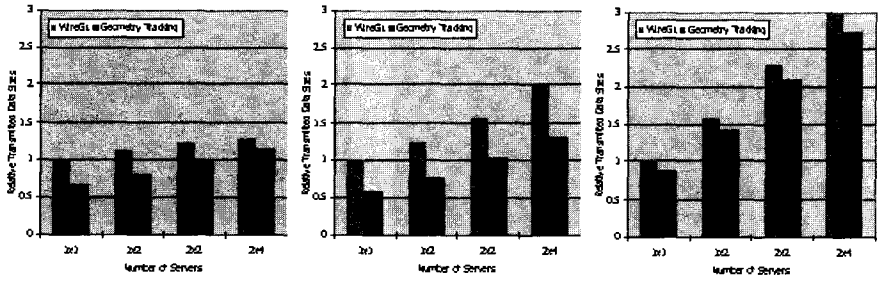
한 패킷이 전송된 후 클라이언트에서는 색인 테이블이 다시 초기화되고, 다음 데이터들을 마찬가지의 방법으로 처리하게 된다. 그림에서 16번부터 19번까지의 명령어와 데이터는 순서대로 색인 테이블과 기하 버퍼에 보관되었다. 22번의 glVertex3f 명령어가 호출되었을 때 17번의 데이터와 중복되기 때문에 17번 데이터의 색인값인 2가 22번을 위한 정점 데이터로 기하 버퍼에 저장되었다. 마찬가지로 25번 코드의 정점 데이터를 위해서도 18번 데이터의 색인값인 3이 기하 버퍼에 저장되었다.

4. 실험 결과

우리는 클라이언트 머신 하나와 8개의 렌더링 서버들로 구성된 8-노드의 클러스터 시스템을 구축하였다. 각 렌더링 서버에는 펜티엄4 1.6GHz 프로세서와 NVIDIA GeForce3 Ti 200 그래픽 가속기를 장착하였고, 클라이언트 머신은 듀얼 AMD Athlon XP 1800+ 1.5Ghz 프로세서와 NVIDIA GeForce3 Ti 200 그래픽 가속기가 장착되었다. 각 렌더링 서버는 1024x768 해상도의 비디오 출력신호를 생성하게 된다. 이 시스템에서 실험을 위해 3개의 벤치마크를 사용하였다. [그림 5]는 실험에 사용된 벤치마크의 영상을 보여준다.

1. *DRV-07*는 SPECViewperf의 벤치마크중 하나이다. 367,178개의 정점과 42,821의 다각형으로 구성되고, 각 객체들이 메모리에 상주된 뒤 렌더링된다. 다른 SPECViewperf의 벤치마크들에 비해, 데이터 크기만 50 메가바이트가 넘는 복잡한 영상을 생성하는 벤치마크이기 때문에 실험을 위해 사용되었다.

2. *Atlantis*는 표준 GLUT 라이브러리 배포판에 포함되어 있는 것으로 OpenGL 데모 프로그램 중 하나이다. 상어, 고래, 돌고래의 경영 장면을 시뮬레이션 한 프로



(a) DRV-07 (b) Atlantis (c) Quake III

그림 6 WireGL과 기하 추적 기법으로 총 전송된 상대적인 데이터 크기 비교

그램으로 각 객체들의 본체들이 실시간으로 계산되어 렌더링 된다. 프로그램 특성상 무한 렌더링 되는 프로그램이므로 실험을 위해 100프레임으로 렌더링을 제한하였다.

3. *Quake III*는 많은 관련 연구들에서 벤치마크로 사용하고 있는 비디오 게임의 하나이다. 가시성 제거(visibility culling)방법이 적용된 건축학적 이동구조(architectural work-though) 기법이 사용된 대표적인 프로그램이다. 200프레임을 갖는 게임 데모중 하나가 실험에 사용되었다.

4.1 한 패킷 내의 데이터 저장 효율

제안하는 기하 추적 방법을 사용하면, 중복된 정점을 기하 버퍼로 저장할 때 WireGL과 비해 적은 메모리만을 필요로 한다. 따라서 고정 크기의 패킷을 사용하여 전송하기 때문에 WireGL에 비해 한 패킷 내에 담을 수 있는 정점 데이터 수를 증가시킬 수 있고, 결과적으로 기하 추적 방법을 사용하면, 전체 전송되는 패킷의 수를 줄일 수 있게 된다. [표 2]는 WireGL과 기하 추적 방법을 사용했을 때의 전체 전송된 패킷의 수의 비교된 결과이다. 이는 하나의 클라이언트와 서버에서 실험되었고, [표 2]에서 볼 수 있듯이 전송 패킷은 최대 45%까지 감소 될 수 있었다.

4.2 네트워크 트래픽의 감소

각 벤치마크들은 4가지의 렌더링 서버 설정(1x1, 1x2, 2x2, 2x4)의 환경들에서 렌더링되어 실험되었다. [그림 6]는 서버의 수를 증가시켜 가며 실험한 총 전송된 데이터들의 비교된 결과이다. WireGL의 경우와 제안하는 기하 추적 기법을 비교하기 위해, WireGL의 1x1서버로 전송된 데이터의 크기 대 기타의 전송된 데이터의 크기의 비율로 계산하여 상대적 전송 비율(relative transmission rate)을 측정하였다.

표 2 WireGL과 기하 추적 기법이 사용된 경우의 총 전송된 패킷의 수의 비교

	DRV-07	Atlantis	Quake III
WireGL	774,806	2,723	10,228
Geometry tracking	494,572	1,489	9,096
Reduction rate (%)	36.17	45.32	11.08

[그림 6]에서 보듯이, DRV-07의 경우는 11%에서 27%까지, Atlantis의 경우는 34%에서 42%까지, 그리고 Quake III의 경우는 8%에서 12%까지의 범위로 네트워크 트래픽이 감소됨을 알 수 있다. Quake III는 많은 양의 텍스쳐 데이터를 각 서버에 전송해야 했기 때문에 상대적으로 다른 벤치마크에 비해 감소율이 크지 않았다.

5. 결 론

본 논문은 분산 환경에서의 원격 렌더링 시스템을 구축할 시 중복된 데이터를 색인화하여 네트워크 대역폭을 감소시키는 기하 추적 기법을 제안하였다. 실험을 통해 기하 추적 기법은 최대 42%까지 네트워크 대역폭을 감소시킬 수 있음을 알 수 있었다. 하지만 Quake III와 같은 텍스쳐 집약적인 어플리케이션의 경우 큰 텍스쳐의 트래픽이 발생하기 때문에 다른 벤치마크만큼 감소율이 크지 않았다.

기하 추적 기법은 텍스쳐 트래픽을 감소시킬 수 있도록 알고리즘이 확장될 것이고, 또한 좀 더 많은 중복율을 기대할 수 있는 볼륨 렌더링과 같은 어플리케이션을 지원하도록 API가 구현될 예정이다. 이는 볼륨 렌더링 시스템에 적용되어 전체 성능의 향상을 가져다 줄 것으로 기대되고, 폭넓게 확장형 디스플레이의 응용분야에 접목될 것으로 예상된다.

참 고 문 헌

[1] D. Kirk, "Unsolved Problems and Opportunities for High-quality High-performance 3-D Graphics on a PC Platform," *Keynotes Address In 1998 ACM*

SIGGRAPH/Eurographics Workshop on Graphics Hardware, pp. 11-13, Sep. 1998.
[2] University of Minnesota. PowerWall. *http://www.lcse.umn.edu/research/powerwall/powerwall.html.*
[3] G. Humphreys and P. Hanrahan, "A Distributed Graphics System for Large Tiled Displays," *Proceedings of the Conference on IEEE Visualization '99*, pp. 215-223, Oct. 1999.
[4] M. Czernuszenko, D. Pape, D. Sandin, T. DeFanti, G. Dawe, and M. Brown, "The ImmersaDesk and InfinityWall projection-based virtual reality displays," *Computer Graphics*, pp. 46-49, May 1997.
[5] D. Bartz, B. Schneider and C. Silva, "Rendering and Visualization in Parallel Environments," *Proceeding of the SIGRAPH 2000 Course Notes*, 2000.
[6] B. Wylie, C. Pavlacos, V. Lewis and K. Moreland, "Scalable Rendering on PC Clusters," *IEEE Computer Graphics and Applications*, Vol. 21, No. 4; pp. 62-70, July/Aug. 2001.
[7] B. Wei, C. Silva, E. Koutsofios, S. Krishnan, and S. North, "Visualization Research with Large Displays," *IEEE Computer Graphics and Applications*, Vol. 20, No. 4, pp. 50-54, July/Aug. 2001.
[8] K. Li, H. Chen, Y. Chen, D. W. Clark, P. Cook, S. Damianakis, G. Essl, A. Finkelstein, T. Funkhouser, T. Housel, A. Klein, Z. Liu, E. Praun, R. Samanta, B. Shedd, J. P. Singh, G. Tzanetakis, and J. Zheng "Building and Using a Scalable Display Wall System," *IEEE Computer Graphics and Applications*, Vol. 20, No. 4, pp. 29-37, July/Aug. 2001.
[9] G. Humphreys, I. Buck, M. Eldridge and P. Hanrahan, "Distributed Rendering for Scalable Displays," *Proceedings of Supercomputing 2000*, pp. 30-38, 2000.
[10] R. Samanta, T. Funkhouser, K. Li, and J. Pal Singh, "Hybrid Sort-First and Sort-Last Parallel Rendering with a Cluster of PCs," *SIGGRAPH/Eurographics Workshop on Graphics Hardware*, pp. 87-108, August, 2000.
[11] G. Humphreys, M. Eldridge, I. Buck, G. Stoll, M. Everett and P. Hanrahan, "WireGL: A Scalable Graphics System for Clusters," *Proceedings of SIGGRAPH 2001*, pp. 129-140, 2001.
[12] M. Woo, J. Neider, T. Davis and D. Shreiner, *OpenGL Programming Manual 3rdEd.*, Addison & Wesley, 1997. http://www.spec.org/gpc/opc.static/opcview.htm
[13] SPECViewperfTM, .

박 우 찬

1993년 연세대학교 이과대학 전산과학과 졸업(학사). 1995년 연세대학교 대학원 전산과학과(이학석사). 2000년 연세대학교 공과대학 컴퓨터과학과(공학박사). 2001년 ~ 현재 연세대학교 연구교수. 관심분야는 3차원 그래픽 가속기, ASIC 설계, 병렬 렌더링, 고성능 컴퓨터 구조, Computer arithmetic

이 원 종

1999년 인하대학교 전자계산공학과 졸업(공학사). 2001년 연세대학교 컴퓨터과학과 졸업(공학석사). 2001년 현재 연세대학교 대학원 컴퓨터과학과 박사과정. 관심분야는 3차원 그래픽 가속기, 볼륨 렌더링, 분산 렌더링, 고성능 컴퓨터 구조, 모바일 컴퓨팅

김 형 래

2001년 연세대학교 공과대학 기계·전자공학부 컴퓨터과학과 졸업(학사). 2001년 ~ 현재 연세대학교 대학원 컴퓨터과학과 석사과정. 관심분야는 3차원 그래픽 가속기, 병렬 볼륨 렌더링, 3차원 그래픽 엔진 설계

김 정 우

2001년 성균관대학교 공과대학 전기·전자 및 컴퓨터공학부 졸업(학사). 2001년 ~ 현재 연세대학교 대학원 컴퓨터과학과 석사과정. 관심분야는 3차원 그래픽 가속기, 볼륨 렌더링 아키텍처, 영상 처리

한 탁 돈

1978년 연세대학교 공과대학 전자공학과 졸업(학사). 1983년 Wayne State University 컴퓨터공학(공학석사). 1987년 University of Massachusetts 컴퓨터공학(공학박사). 1987년 - 1989년 Cleveland 주립대학 조교수. 1989년 ~ 현재 연세대학교 공과대학 컴퓨터과학과 교수. 관심분야는 Wearable computer, HCI, ASIC 설계, 고성능 컴퓨터구조

양 성 봉

1984년 University of Oklahoma 컴퓨터과학(학부과정). 1986년 University of Oklahoma 컴퓨터과학(석사). 1992년 University of Oklahoma 컴퓨터과학(박사). 1992년 ~ 1992년 University of Oklahoma, Adjunct Assistant Professor. 1993년 ~ 1994년 전주대학교 전자계산학과 전임강사. 1994년 ~ 현재 연세대학교 공과대학 컴퓨터과학과 부교수. 관심분야는 병렬처리, Computational geometry, Spatial data processing, 3D graphics algorithm. GIS