공학석사 학위논문

조선응용을 위한 컴퓨터 클러스터 기반 스테레오 가시화 시스템 구축

Construction of stereoscopic visualization system based on computer cluster for shipbuilding application

지도교수 남 종 호

2009년 2월

한국해양대학교 대학원

해양시스템공학과

김 현 철

本 論文을 김 현 철의 工學碩士 學位論文으로 認准함.

위원장 : 공학박사 박 주 용 (인)

위 원 : 공학박사 손 경 호 (인)

위 원 : 공학박사 남 종 호 (인)

2009년 2월

한 국 해 양 대 학 교 대 학 원

목 차

List of Tables

List of Figures

Construction of stereoscopic visualization system based on computer cluster for shipbuilding application

Hyun-Chul, Kim

Division of Naval Architecture and Ocean systems Engineering
Graduate School of Kore Maritime University

Abstract

Applications of virtual-reality based visualization systems whose realtime response is essential in performing graphic simulation can be found in enormous fields. The design process of ships or ocean structures where the geometric shapes are complicated and tons of small and large parts are gathered and assembled would be efficiently improved with prompt and accurate graphic simulations in advance. Stereoscopic visualization systems used in the graphic simulation have become popular for their ability to represent realistic images. The construction of such a system, however, is not readily affordable due to the high cost of required hardware and software.

In this dissertation, a practical technique that replaces the expensive hardware based performance by economical software programming is introduced. A stereoscopic clustering visualizations system consisting of five personal computers and two beam projectors is implemented. Passive stereoscopic principle, message passing parallel programming technique, frame synchronization as well as data handling process are integrated together to achieve a high performance computing that generates dependable stereoscopic projection images. The implemented system is verified with complex ship models.

1. 서 론

1-1. 연구배경

최근 삼차원(3D) 가상현실 가시화 시스템은 고성능 하드웨어, 효율적인 프로그래밍 소프트웨어, 첨단 입출력 장치와 같은 관련 기술의 발전에 힘입어 나날이 그 성능이 진보하고 있고, 응용분야 또한 무한히 확장되고 있다. 특히 실시간 처리를 필요로 하는 삼차원 가시화 시스템은 공학적으로 그래픽 시뮬레이션 영상을 구현하고, 그 영상을 보다 사실적으로 표현하기 위하여 필수적으로 요구되는 사항이다. 조선구조물과 같이 기하학적 형상이 복잡하고, 크고 작은 부품들이 어울려 복합적인 객체를 이루는 분야에서는 정밀한 그래픽시뮬레이션을 통해 설계 효율을 향상시킬 수 있다(이창민 등, 2004). 특히 인간의 좌우양안의 원리를 이용한 스테레오스코피(Stereoscopy) 시스템은 시각적인 사실감을 높여 보다 현실에 가까운 영상을 표현할 수 있어 공학설계 분야뿐만 아니라, 오락, 의료, 건축, 군사, 항공, 우주 등 다양한 분야의 가시화 시스템(Fig. 1)에 활용하려는 시도가 끊임없이 계속되고 있다. 그리고 이미 다양한 3D CAD 프로그램에 플러그인 형식으로 스테레오스코피를 지원하는 기술들이 개발되어 정밀한 설계를 요하는 분야에서 널리 사용되고 있다.

하지만 가상현실 기반의 가시화 시스템을 구축하기 위해서는 고성능 그래픽 하드웨어 플랫폼이 필요한데, 삼차원 스테레오를 구현하기 위한 시스템의 비용이 너무 높아 일반적인 스테레오스코피 기초연구, 교육 그리고 관련 어플리케이션 개발 목적으로 활용하기에는 현실적인 어려움이 있다. 또한 하드웨어 생산업체나 모델에 따라 해당 기술이 특화되어 있기 때문에 기술의 투명성이 낮고, 특화된 하드웨어 성능에 매우 의존적이어서 사용자가 시스템을 자유롭게 제어하기 어렵다.

반면 오늘날 중앙처리장치(CPU) 및 그래픽처리장치(GPU) 성능이 비약적으로 발전하고 있고, 이로 인해 고가장비의 성능에 준하는 COTS (Commercial

Off The Shelf) 제품을 일반 시장에서 손쉽게 구할 수 있다. 즉 여러 대의 PC 자원들을 결합해 저렴한 비용으로 다양한 목적의 클러스터 시스템을 구축할 수 있고, 나아가 기존의 단일 고성능 플랫폼의 성능에 준하는 가시화 시스템을 구축할 수 있다(윤재문 등, 2005). 특히 최근 연산을 위한 처리장치(CPU & GPU) 및 병렬처리 라이브러리가 듀얼 이상의 다중코어를 지원해 클러스터링을 통한 시스템 성능향상 집적도 또한 극대화 되고 있기 때문에, 이와 같은 클러스터링 기법은 가시화를 위한 고가의 고성능 그래픽 하드웨어 플랫폼의 대안으로 정착되어 가고 있다.

Fig. 1 Three Dimensional Cube Stereo Visualization Cluster System

1-2. 논문 구성

본 논문에서는 하드웨어의 특화된 기능을 소프트웨어적으로 대체하는 경제적이고, 사용자 친화적인 스테레오 가시화 클러스터 시스템(Stereoscopic visualization cluster system, 이하 SVCS)과 스테레오 시스템을 구동하기 위한 소프트웨어적인 가시화 기법에 대해 논하고자 한다. 즉 Fig. 2와 같이 기존의 PC 클러스터링 기법과 스테레오스코피 원리, 그리고 스테레오스코피를 클러스터 시스템에서 구현할 때 필요한 대체기법에 대해 연구하고 이를 통해 SVCS를 구축하는 방법에 대해 상세히 기술한다.

Fig. 2 Concept of Stereoscopic Visualization Cluster System

스테레오스코피 원리를 바탕으로 양안의 시야영역을 정의한 스테레오 알고리즘을 유도한다. 스테레오스코피의 기반이 되는 SVCS는 그래픽전용 카드를 장착하지 않은 세 대의 개인용 컴퓨터와 네트워크 장치 그리고 두 대의 프로젝터로 구성된다. 내부적으로 메시지전달을 이용한 병렬 프로그래밍 기술과 프레임 동기 기술을 결합시켜 수동형 스테레오스코픽 영상을 구현하고, 모델 데이터의 분산 및 취합 과정을 통해, 클러스터 시스템의 주요 목적인 고성능컴퓨팅(High Performance Computing, 이하 HPC)을 실현한다. 시스템을 검증하기 위해 응용 소프트웨어를 제작하고, 이를 통해 SVCS를 다양한 각도에서 검증 테스트하여, 해당 시스템의 유용성을 가늠해 본다. 검증된 내용을 바탕으로 시스템의 문제점을 파악하고 문제에 대한 해결방안을 모색함으로써 향후 SVCS의 발전방향을 제시한다.

2. 삼차원 스테레오 가시화 시스템 구성요소

2-1. 스테레오스코피 원리

삼차원 스테레오 입체영상의 가장 기본적인 원리는 인간의 시각시스템이 왼쪽 눈과 오른쪽 눈의 위치 차이에 의해 서로 다른 영상이 들어오고, 뇌는 그것을 입체로 받아들여 거리감을 갖게 되는 과정에서 입체감을 형성하게 되는 것이다. 즉 인간의 양안이 가로방향으로 약 65㎜ 떨어져서 존재하는 양안시차가 입체감의 가장 중요한 핵심이다. 이 원리를 기반으로 스테레오스코픽 알고리즘을 설계할 수 있다.

알고리즘은 크게 세 가지 특성을 고려하고 있다. 첫 번째로 Fig. 3과 같이 인간의 양안을 기준으로 영상이 맺히는 면과의 각과 비율을 조합해 독립적인 방정식을 유도하여 양안의 시야범위를 각각 표현할 수 있다. 아래는 스테레오 좌우 양안의 시야범위를 유도하는 과정을 설명하고 있다.

스테레오 알고리즘을 유도하기 위해 모노 시점의 좌우 눈의 위치가 정의되어야 한다. Fig. 2에서의 디스플레이의 기하학적 관계를 이용하여 아래의 식을 유도할 수 있다(Ericksson, G, 2003).

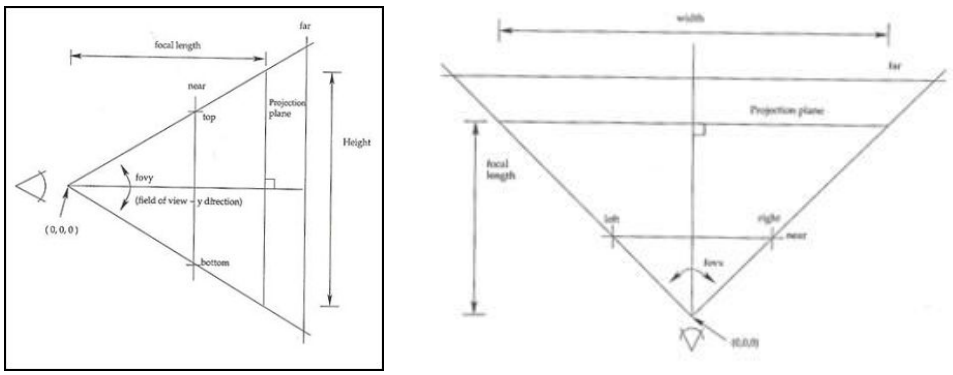


Fig. 3 Mono Field of View (Side & Top View)

$$\text{aspect ratio} = \frac{\text{width}}{\text{height}} \tag{1}$$

$$\tan\left(\frac{\text{fovy}}{2}\right) = \frac{\text{top}}{\text{near}} = \frac{\text{bottom}}{\text{near}} = \frac{\text{height}/2}{\text{focal length}}, \qquad \tan\left(\frac{\text{fovx}}{2}\right) = \frac{\text{width}/2}{\text{focal length}} \tag{2}$$

$$\tan\left(\frac{\text{fovx}}{2}\right) = \frac{\text{aspect ratio} \times \text{height}/2}{\text{focal length}} = \text{aspect ratio} \times \tan\left(\frac{\text{fovy}}{2}\right) \tag{3}$$

(3)식에서 만약 영상비(aspect ratio)가 1이 되면 (width와 height가 같다면) fovx, fovy 역시 같아진다는 사실을 알 수 있다.

$$\tan\left(\frac{\text{fovx}}{2}\right) = \frac{\text{left}}{\text{near}} = \frac{\text{right}}{\text{near}}, \text{left} = \text{right} = \text{near} \times \tan\left(\frac{\text{fovx}}{2}\right)$$

$$\therefore \text{left, right} = \text{near} \times \text{aspect raion} \times \tan\left(\frac{\text{fovy}}{2}\right) = \text{top} \times \text{aspect ratio} \tag{4}$$

이렇게 기하학적 관계를 이용해 모노 디스플레이에서 눈의 영역 값을 정의할 수 있다. 모노 디스플레이기 때문에 좌, 우 값이 같음을 주목할 수 있다.

계산된 모노 시야범위를 기반으로 스테레오 디스플레이의 좌, 우 양안의 시야범위를 각각 유도할 수 있다. 이 유도과정은 Fig. 2와 Fig. 3의 기하학적 관계를 활용한다.

$$\tan\theta = \frac{\text{left}}{\text{near}} = \frac{\text{width}/_2 + \text{ios}/_2}{\text{focal length}}, \tan\left(\frac{\text{fovx}}{2}\right) = \frac{\text{width}/2}{\text{focal length}} = \text{aspect ratio} \times \tan\left(\frac{\text{fovy}}{2}\right) \tag{5}$$

여기서 두 번째 식은 모노 기하학적 관계에서 유도된 (3)식을 변형한 것이다.

$$\frac{\text{width}/_2 \pm \text{ios}/_2}{\text{focal length}} = \text{aspect ratio} \times \tan\left(\frac{\text{fovy}}{2}\right) \pm \frac{\text{ios}/_2}{\text{focal length}}$$

$$= \tan\theta \text{ or } \tan\varphi = \frac{\text{left}}{\text{near}} \text{ or } \frac{\text{right}}{\text{near}} \tag{6}$$

(5)의 두 번째 식 양 변에 $\frac{\pm ios/2}{\text{focal length}}$를 더한 후 기존의 식을 변형하게 되면 (7)과 같이 한쪽 눈의 좌, 우 시야영역을 표현할 수 있다.

Right Eye

$$\text{left} = \text{near} \times \text{aspect ratio} \times \tan\left(\frac{\text{fovy}}{2}\right) + \frac{\text{near} \times ios/2}{\text{focal length}}$$

$$\text{right} = \text{near} \times \text{aspect ratio} \times \tan\left(\frac{\text{fovy}}{2}\right) - \frac{\text{near} \times ios/2}{\text{focal length}} \quad (7)$$

현재, 오른쪽 눈에 대한 시야영역을 정의하였으므로 왼쪽 눈에 대해서도 (5) ~ (7)의 과정을 통해 시야영역을 정의한다. Fig. 4에서 mono eye position이 0 이므로 (8)에서 보듯이 전체 식의 부호가 반대로 결정된다.

Left Eye

$$\text{left} = -\left(\text{near} \times \text{aspect ratio} \times \tan\left(\frac{\text{fovy}}{2}\right) + \frac{\text{near} \times ios/2}{\text{focal length}}\right)$$

$$\text{right} = -\left(\text{near} \times \text{aspect ratio} \times \tan\left(\frac{\text{fovy}}{2}\right) - \frac{\text{near} \times ios/2}{\text{focal length}}\right) \quad (8)$$

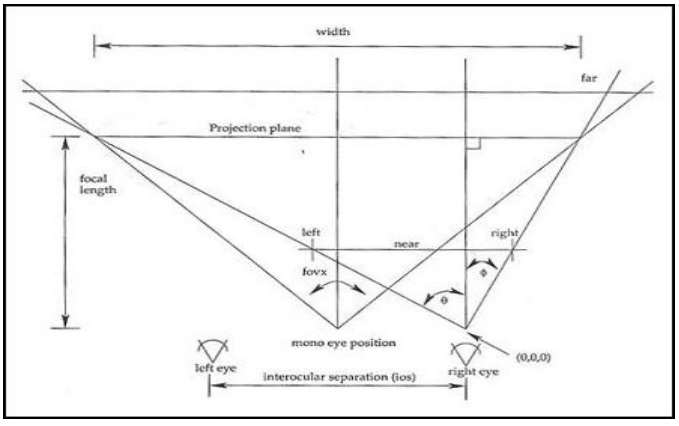


Fig. 4 Stereo Field of View (Top View)

다음으로 삼차원 모델 공간 내에서의 카메라의 이동을 고려해야 한다. 카메라는 삼차원 모델 공간에서 좌, 우를 직접 구분할 수 없지만, 주어진 데이터의 좌표를 바탕으로 좌, 우의 방향을 계산함으로써 삼차원 영역 내에서 카메라의 이동 위치 값을 제어할 수 있다. 즉 실제 눈 위치벡터의 스칼라 값을 계산한 후 삼차원 모델과의 거리의 축소비를 서로 곱해 누적시켜가면서 카메라 값을 설정해주어야 정확한 스테레오스코픽 카메라의 위치를 지정할 수 있다.

마지막으로 Parallax(시차)를 고려해야 한다. 시차란, Fig. 5에 보이듯이, 투영 스크린과 가상 오브젝트 사이의 거리로서 보통 zero 시차가 사용된다. 하지만 보다 향상된 삼차원 효과를 부여하기 위해 Positive, Negative 시차를 사용하기도 한다. 주의할 점은 스크린에 생성되는 두 점 사이의 거리가 두 눈 사이의 거리보다 커질 경우 불편함을 느끼므로 알고리즘 설계 시 이를 잘 고려해야 한다.

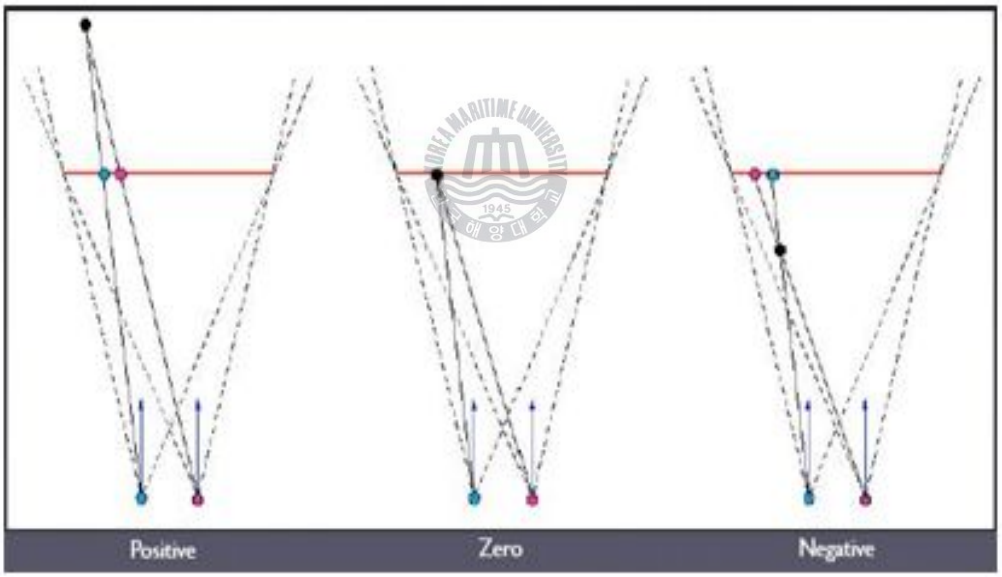


Fig. 5 Three Method of Parallax

2-2. Stereo 기술 지원

스테레오스코픽 영상을 표현하기 위해서는 삼차원 스테레오를 지원하는 그래픽 라이브러리와 하드웨어, 그리고 그래픽 디스플레이가 필요하다. 지원되는 그래픽 라이브러리로는 대표적으로 SGI의 OpenGL 계열이 있다. 그리고 OpenGL의 Stereo API를 사용하기 위해서 그래픽 하드웨어는 쿼드 버퍼링기술을 지원해야 한다. 쿼드 버퍼링은 각 더블 버퍼가 좌, 우 영상에 할당되어 연속된 프레임을 형성함으로서 스테레오 영상을 구현하는 기술이다. 이를 지원하는 그래픽 하드웨어로는 대표적으로 nVidia社의 Quadro Series, ATI社의 Fire Series 등이 있다(Fig. 6).

Fig. 6 Stereoscopic-Supported Graphic Hardwares

그래픽 디스플레이는 그래픽 하드웨어로부터 좌우로 분리된 영상을 인간이 시각적으로 삼차원 스테레오스코피를 인지하게 만드는 장비로서, 현재도 다양한 방식의 첨단 입체 디스플레이가 개발되고 있다. 하지만 일반적으로 크게 안경착용 여부에 따라 두 가지 방식으로 구분할 수 있다(여진욱, 2005).

첫 번째로 안경방식이 있다. 안경방식에는 passive 방식인 편광(Polarization) 방식과 active 방식인 시분할 방식이 주로 사용되며, 그 외에 HMD(Head Mounted Display), 색차방식 등이 있다. 편광 방식은 Fig. 7에서 보이는 바와

같이 직교한 편광소자의 조합에 의한 차광효과를 이용해 좌, 우안의 화상을 분리하는 방식으로 편광방식 디스플레이 장치에서 동시에 두 영상이 출력될 때, 좌우 시야각을 확보하기 위해 전체화면을 두 개의 수직방향으로 분할하여 동시에 출력한다. 동시에 출력된 영상을 편광 안경을 통해 서로 반대 방향의 편광을 걸러내게 되면 인간의 뇌는 이를 합성하여 삼차원 스테레오스코픽 영상을 인식하게 된다.

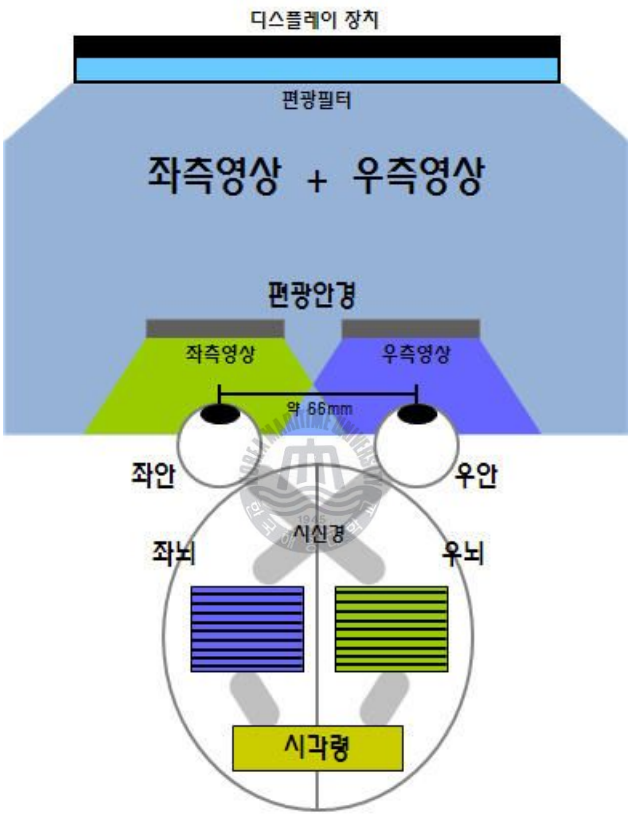


Fig. 7 Principles of Polarized Stereoscopy Display

편광방식은 편광필터가 빛을 일부 흡수하는 구조상 전체적인 빛의 투과량이 낮아지기 때문에 디스플레이 장치 표면에 부착하는 편광필터와 편광안경까지 더해져 상당한 광량 감소효과를 가져오는 단점이 있다. 그래서 이에 대한 대책

으로 밝기가 밝은 제품을 베이스로 사용해야 할 필요가 있다. 하지만 편광방식은 다른 방식들에 비해 두 채널의 영상을 분할하는 방향에 따라 좌우나 상하 한 방향으로의 시야각을 굉장히 넓게 가질 수 있어 사용자의 활동성을 보장하고, 다수의 사용자가 동시에 스테레오 효과를 느낄 수 있는 장점이 있다 또한 간단한 시스템에 편광필터를 설정만 하면 스테레오스코피가 가능하므로 상대적으로 설치가 용이하다. 이러한 장점 때문에 일반적인 교육이나 연구를 목적으로 하는 사용자에게 특히 유용하며, 본 연구에서도 여기 설명한 편광방식을 사용해 SVCS를 구축한다. Fig. 8은 본 연구실의 수동형 스테레오스코픽 디스플레이 시스템을 시연하는 전경이다.

Fig. 8 Passive Stereoscopy Display System

시분할 방식은 Fig. 9에서 보이는 것처럼 하나의 화면에서 셔터안경을 이용해 좌우 채널의 영상을 빠른 속도로 번갈아 차단, 개폐하여 한쪽 눈에 한쪽 방향의 이미지를 투영시키는 방식이다. 좌우 전환속도가 초당 60회 이상으로 빠르게 작동하기 때문에 실제 눈에서는 깜빡임이 느낄 수 없지만 CRT 모니터에서 최적화될 수 있는 방식인 만큼 최근에는 널리 사용되지 않는다. 또한 셔터안경과 시분할 디스플레이 장치의 동기가 중요하기 때문에 설정이 까다롭고 상대적으로 스테레오 플랫폼의 가격이 비싼 편이다.

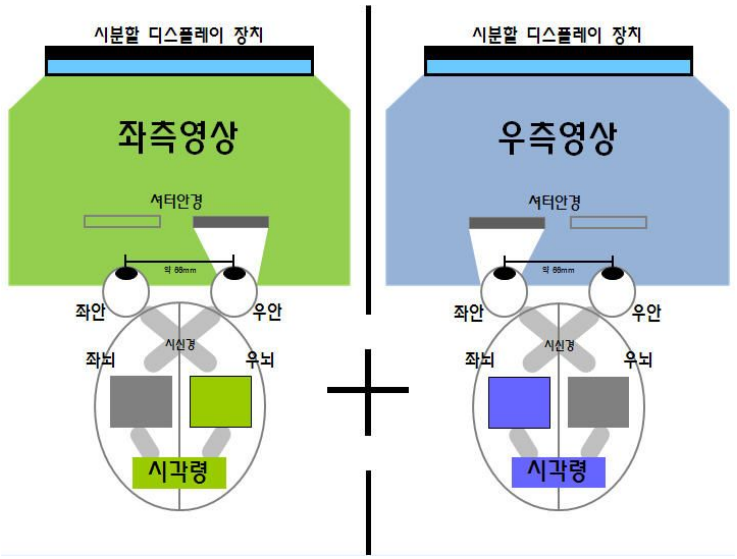


Fig. 9 Principles of Temporal Separation Stereoscopy Display

색차 방식은 Fig. 10에서처럼 편광 방식의 원리와 비슷하게 양 쪽이 붉은색, 파란색으로 구성된 색안경을 통해 양 채널의 화상을 색상 차원에서 분리하여 스테레오스코피를 구현하는 방식이다. 하지만 이 방식은 입체감의 질을 떠나 색안경을 사용하는 만큼 색의 왜곡 문제가 있어 점점 쇠퇴하고 있는 추세이다.

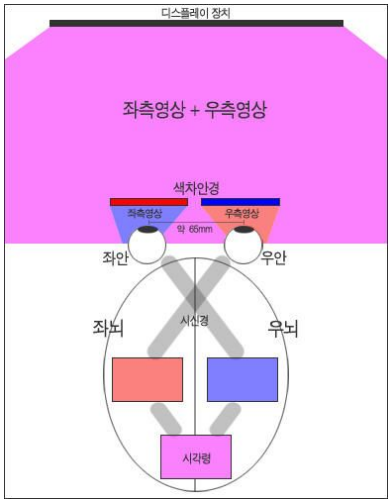


Fig. 10 Principles of Anaglyph Stereoscopy Display

다음으로 무안경 방식이 있다. 무안경 방식은 LCD와 같은 평판 디스플레이가 본격적으로 생산되기 시작하면서 본격적으로 개발이 이루어지고 있고 현재까지 개발된 방식으로는 Parallax Barrier방식과 Lenticular방식이 있다. Fig. 11에서처럼 이 두 방식은 안경을 사용하지 않아도 되므로 안경방식에 비해 큰 강점을 가지고 있지만 시야각의 제약이 많아 일반적인 대중을 목적으로 사용하기에는 어려움이 따르며, 디스플레이 장치의 기술력에 의존적이므로 역시 그 구축비용이 만만치 않다는 단점이 있다. 그렇지만 크게 시야각에 구애받지 않는 공학설계나, 게임과 같은 1인 사용자를 대상으로 하는 분야에서는 그 유용성을 인정받고 있다.

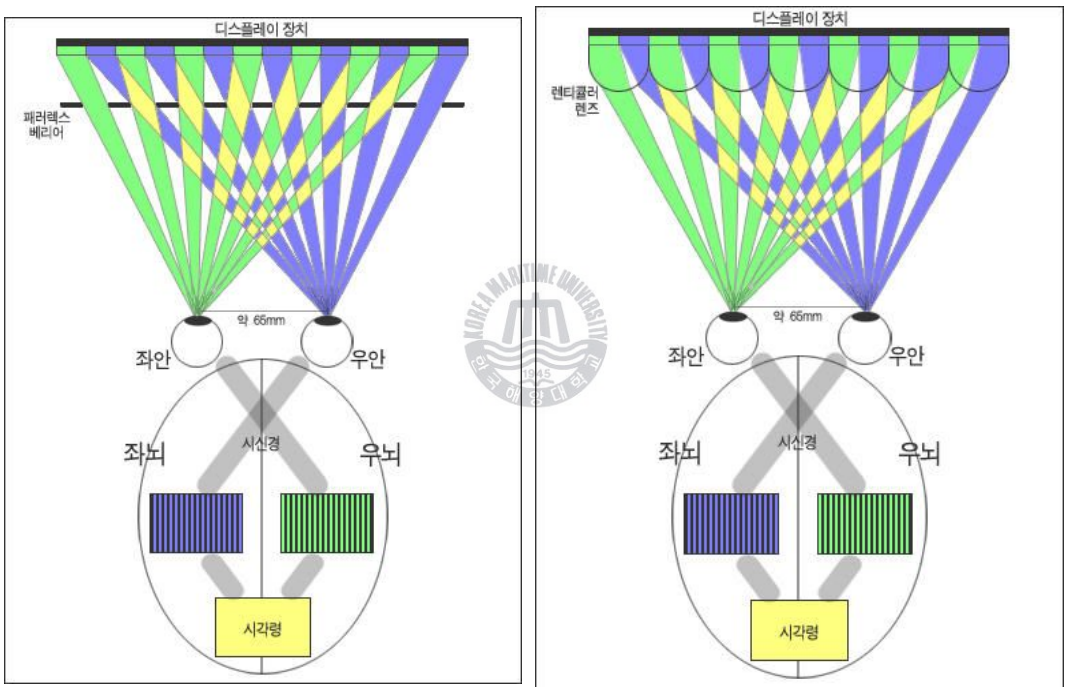


Fig. 11 Principles of No-Glasses Stereoscopy Display

2-3. 시스템 구현의 제약조건 및 개발방향

앞서 기술한 그래픽 하드웨어들의 특징은 모두가 고가장비라는 점이다. 각 그래픽 라이브러리에서 요구하는 삼차원 스테레오 API를 사용하기 위해서는 하드웨어의 기술적인 지원이 뒷받침되어야 하고, 고가의 고성능 하드웨어 장비들이 필요하게 된다. 그리고 이러한 고성능 하드웨어 장비들은 생산업체나 모델명에 따라 해당 기술이 특화되어 있기 때문에 기술의 투명성이 낮고, 특화된 하드웨어 성능에 매우 의존적이어서 사용자가 시스템을 자유롭게 제어하기 어렵다. 따라서 일반적인 가시화 기반의 연구를 목적으로 하는 개인 사용자의 경우 현실적으로 스테레오 시스템을 구축 하려면 위와 같은 많은 제약조건들을 피하거나 대체하는 기술을 고려해야 한다.

이러한 제약조건을 해결하기 위해 본 논문에서는 PC 클러스터링 기술을 삼차원 스테레오 가시화 시스템에 접목시켜 고가의 하드웨어에 특화된 그래픽 시스템을 대체할 수 있는 SVCS를 구축한다. 나아가 메시지 전달 기반의 병렬 프로그래밍 기법을 이용해 가시화 클러스터의 성능을 향상시키기 위한 프레임 동기화, 데이터 분산, 취합 방법에 대해 연구하고 최종적으로 시스템 검증을 위한 응용 소프트웨어를 제작하여 SVCS의 성능을 다양한 각도에서 비교, 분석한다.

3. 스테레오 가시화를 위한 요소 기술

3-1. Quad 버퍼 스테레오

통상적으로 일반 그래픽 하드웨어와 OpenGL API는 더블 버퍼링을 지원한다. 더블 버퍼링이란 화면에 보여줄 버퍼와 내부 계산에 사용할 버퍼를 메모리 영역에서 따로 유지하여, 내부 버퍼에 계산 작업을 미리 렌더링한 후 화면 버퍼로 고속 전송하며, 그리는 중간 과정을 숨겨진 내부 버퍼에서 처리하는 방식이다. 이 과정을 통해 두 버퍼를 교대하면서 복잡한 연산과 모델의 렌더링 작업을 끊임없이 반복할 수 있다(최현호, 2005). 나아가 삼차원 스테레오 영상을 구현하기 위해서는 네 개의 프레임 버퍼를 사용해 기존의 더블 버퍼링 절차를 두개의 독립된 작업으로 분리하여 같은 시간 내에 빠르게 좌, 우 영역을 렌더링할 수 있어야 한다. 이와 같은 원리를 쿼드 버퍼링이라고 하는데, 일반 그래픽 하드웨어에서는 지원하지 않는다.

3-2. 클러스터 시스템

클러스터란 네트워크로 연결된 컴퓨터의 그룹이 하나의 공통 작업을 처리할 수 있도록 구성된 시스템을 뜻하며 몇 대 또는 많은 수의 컴퓨터가 네트워크로 서로 연결되어 있고 모든 노드가 동시에 접근할 수 있는 공유 저장소가 있는 기본 구조를 가지게 된다. 일반적으로 클러스터는 특수한 하드웨어 대신 값싼 일반 컴퓨터에 사용하는 하드웨어들로 구성되기 때문에 비슷한 성능의 전용 하드웨어 플랫폼에 비해 매우 비용이 절감되는 특징이 있다. 또한 클러스터링 기술이 보편화되어 필요한 하드웨어와 프로그램을 쉽게 구할 수 있고, 사용자 스스로 개인적인 용도에 맞는 클러스터를 비교적 쉽게 구축할 수 있다.

삼차원 실시간 컴퓨터그래픽 렌더링과 같은 그래픽 처리장치의 연산 작업이 매우 빠르게 끊임없이 수행되어야 하는 고사양의 컴퓨터가 필요한 작업에서 클

러스터 시스템을 사용하게 되면 상당한 비용절감 효과를 얻을 수 있기 때문에 이미 Fig. 12와 같이 기초연구를 비롯한 많은 분야에서 다양한 목적으로 그래픽 시스템에 클러스터 시스템을 적용시켜 사용하고 있다. 그들이 사용하는 프로그램은 대개 그 분야에 맞게 상업적으로 만들어져 판매되는 고가의 프로그램들이지만 클러스터 사용자가 개인으로 확대되어감에 따라 공개적으로 구할 수 있는 그래픽 관련 프로그램도 점차 많아지고 있는 실정이다.

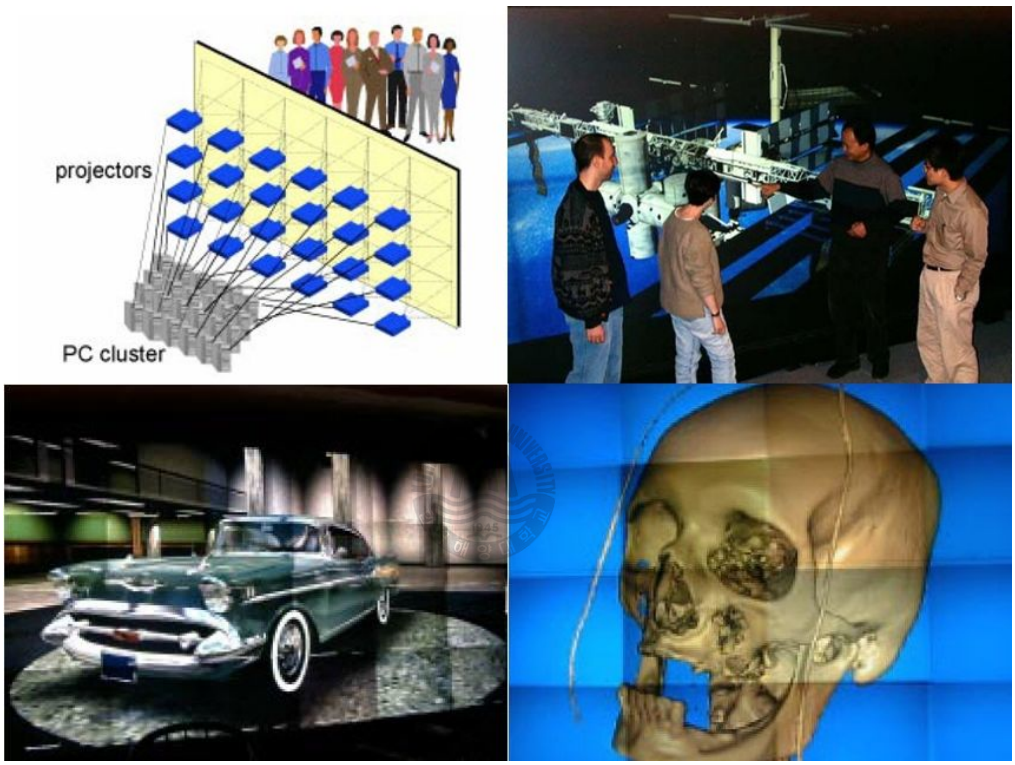


Fig. 12 Graphic Cluster System with Various Purpose (Han Chen 등, 2007)

다만 그래픽 클러스터는 네트워크가 중요한 구성 요소 중 하나이므로 네트워크의 성능에 의해 효율이 많이 좌우되고 클러스터의 규모가 커질수록 효율적으로 성능을 높이기 어려운 단점이 있다. 또한 클러스터의 관리는 그래픽 시스템과 내부 네트워크의 직접적인 관리가 필요하기 때문에 단일 시스템에 비해 사용자의 전문적이고도 특화된 관리 지식이 요구된다.

3-3. 병렬 프로그래밍

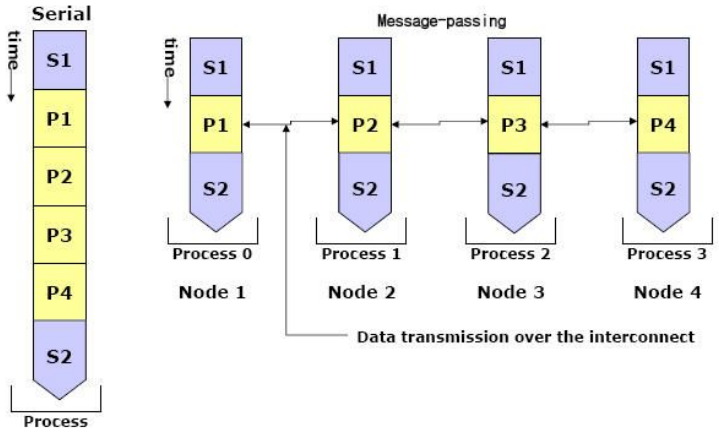


Fig. 13 Message Passing in Multi-node Cluster System

Fig. 13과 같이 클러스터 내에서 지역적으로 메모리를 따로 가지는 프로세스들이 서로 데이터를 공유하기 위해서 메시지를 송수신하여 통신해야 한다. 메시지 전달 기법이라 불리는 이 방법은 병렬화를 위한 작업할당, 데이터 분배, 통신의 운용 등 모든 부분을 프로그래머가 담당하므로 해당 시스템에 아주 유연하게 적용되는 특징이 있다. 또한 단일 프로세서 시스템을 포함한 다양한 하드웨어 플랫폼에서 구현 가능하기 때문에 이식성 또한 뛰어나다. 이러한 메시지 전달 기법을 지원하는 병렬 라이브러리로는 크게 MPI, PVM, Shmem 등이 있지만, 본 논문에서는 표준 규약인 MPI를 사용하여 시스템을 병렬화한다.

MPI는 병렬 실행 프로그램에서 노드에 분산된 프로세스 사이에 정보와 명령 등 메시지를 주고받는 방법을 정의한 규약 중 하나로써 1992년에 표준안 제정이 시작되어 1994년에 최초의 표준인 MPI-1, 이듬해 6월에 현재 MPI-1의 표준인 1.1이 제정되었다. 그 후 계속된 표준안의 확장 노력으로 1997년 새로운 확장 개념이 많이 도입된 MPI-2가 발표되었다. 이와 같은 표준화가 MPI가 병렬 프로그래밍 규약 중 가장 많이 사용되는 이유 중의 하나이다(MPI, 2008).

MPI 규약에 따라 실제로 병렬 프로그램의 메시지 전달을 전담하도록 지원하는 것이 바로 MPICH이다. 클러스터 시스템에서 응용 병렬 어플리케이션을 활용하기 위해서는 표준안인 MPI 패키지를 설치해야 하므로, 본 SVCS에서는 비상용버전인 MPICH를 설치한다. MPICH의 모든 설정은 설치시의 configure 단계에서 결정되므로 이 단계에서 클러스터 시스템의 특성을 반영해야 한다.

클러스터 내에서 각 노드 간의 프로세스 제어와 데이터의 송수신을 위해 병렬 프로그래밍 작업이 필요하다. 클러스터 기반의 초병렬처리(MPP: Massive Parallel Processing) 머신에서 실행되는 대표적인 병렬 프로그램인 MPI-1표준의 MPICH를 사용하여 SVCS 병렬 응용 어플리케이션을 구현할 수 있다. 본 연구에서 개발하고자 하는 스테레오 가시화 시스템은 SIMD(Single Instruction, Multiple Data) 구조를 기반으로 하여 모든 노드에서 같은 프로그램이 실행되지만, 각 노드가 서로 다른 데이터를 처리할 수 있도록 설계되는 것이 타당하다. SIMD의 실행 모델은 Fig. 14에 나타나 있다(신창균, 2006). 즉 사용자가 모든 노드의 작업 프로세스를 하나의 프로그램 내에서 제어할 수 있도록 하여 시스템의 편의성을 도모한다.

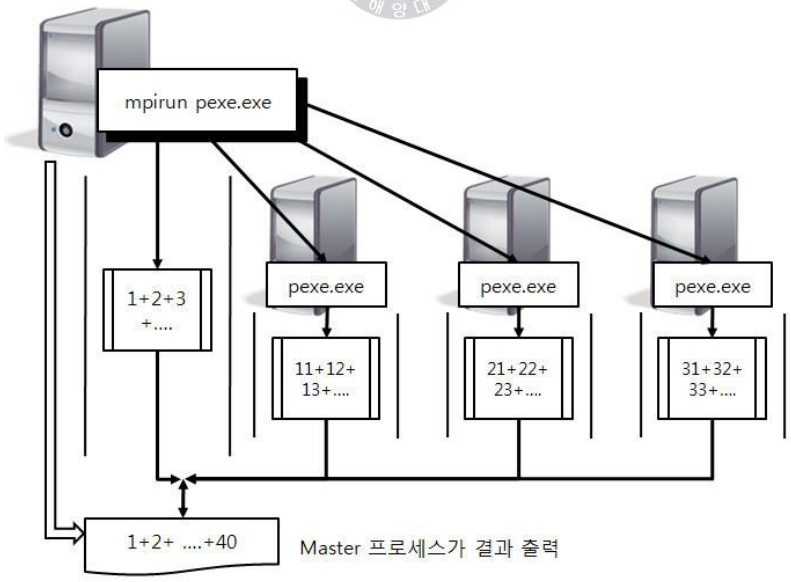


Fig. 14 SIMD Basic Model

MPICH 프로그램은 Fig. 15와 같은 기본구조로 구성되어 있다.

```
processor 선언
int main(int argc, char *argv[])
{
    변수선언;
    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &nproc);
    MPI_Comm_rank(MPI_COMM_WORLD, &myrank);
    실행본문
    for(;;)
    {
        if(노드 프로세스 분기)
        {
            병렬화된 실행 본문;
        }
        ....
    }
    실행본문;
    MPI_Finalize();
}
```

Fig. 15 Basic Structure of MPICH Program

MPI가 다른 일반 프로그램과 특히 다른 점은 프로그램이 노드마다 서로 다른 일부분, 혹은 중복되는 부분의 작업을 처리하도록 사용자가 직접 명시적으로 제어해야 한다는 사실이다. 즉 사용자는 작업이 몇 개 노드에서 실행될 것인지 사전에 인지해야 하고, 프로그램 스스로가 자신이 어떤 노드에서 실행되고 있는지 실행 중에 판단할 수 있어야 한다.

mpirun으로 프로그램을 시작하면 MPI는 사용할 노드의 이름과 개수를 저장한 통신자(Communicator)라는 변수를 시작할 때 정의하고 프로그램 중간에 MPI의 모든 함수는 통신자로부터 노드 수와 자신의 순번(myrank)과 같은 병렬 실행에 필요한 정보를 얻을 수 있다. 통신자는 MPI_Init()에서 정의되고 MPI_Finalize()에서 해제되므로 병렬로 실행되는 부분은 반드시 두 함수 사이에서 실행되어야 한다.

3-4. 동기화

안정된 스테레오 영상을 구현하려면 좌, 우 노드는 한 번에 같은 단계의 프레임을 렌더링 해야 한다. 클러스터 시스템의 계산노드가 모두 같은 CPU 클럭을 가진다 하더라도, 이 사실이 동시에 같은 프레임을 렌더링 한다는 사실을 보장해주진 않는다. 왜냐하면 CPU 클럭을 포함한 시스템의 각종 외부요인들이 실질적으로 서로 정확히 일치할 수 없기 때문이다. 그러므로 스테레오 영상을 구현하기 위해서는 프레임 동기화가 꼭 필요하다(윤재문 등, 2005).

MPI 동기화 API중 MPI_Barrier()라는 동기화 지원함수가 있지만 가시화 시스템의 특성상 렌더링 작업은 반복문 내에서 수행되기 때문에 MPI_Barrier()를 호출할 경우 실행속도가 매우 느려지는 단점이 있다. 가상현실 가시화 시스템에서는 최소한 초당 특정 수 이상의 프레임 갱신속도를 만족해야 하므로 이 함수는 신중하게 사용되거나 만족스럽지 않을 경우 다른 방법을 강구하여야 한다.

3-5. 스테레오스코피 효과

스테레오스코피의 몰입감을 강화하기 위해 2장 1절의 스테레오스코피 원리 요소를 이용할 수 있다. 이 때 스테레오스코픽 효과는 좌우안의 영역범위를 결정하는 기하학적 요소인 fovy, aspect ratio, near, focal length, ios 등과 카메라의 위치, 시차의 영향을 받는다. 즉 이 값들을 적절히 변경함으로써 사용자는 외부여건에 최적화된 스테레오스코피 설정 수치 값을 찾을 수 있다. 특히 최적화된 설정 값을 데이터베이스화하여 응용 소프트웨어에 반영하게 되면 사용자는 쉽게 스테레오스코피 환경을 제어할 수 있다.

그리고 응용 소프트웨어를 통한 몰입감 제어 시 실제 입체감 형성과는 아무 상관없는 어플리케이션의 프레임, 메뉴, 툴바, 마우스 커서 등과 같은 정보 패널들까지 스테레오스코픽 효과를 가진 채 표시된다면 사용자는 조작 미스와 같은 결과를 초래할 것이다. 특히 마우스 커서가 좌우로 분리될 경우 커서에도 거리감이 부여되어 설계와 같은 미세한 조작 시 어려움을 겪을 수 있다. 그러므로 어플리케이션의 프레임부와, 오브젝트 구현부를 스테레오 효과로부터 분리하는 작업이 필요하다. 이는 스테레오스코피 데이터베이스를 통해 자동으로 정보패널과 오브젝트를 구분하여 표시하는 방법으로 해결할 수 있을 것이다.

4. 스테레오 가시화 클러스터 시스템 구축

4-1. 시스템 개요

스테레오스코피 가시화 시스템에서 6프레임 이하의 렌더링 수행결과는 인간의 뇌에 연속적인 동작에 대한 착시를 일으킬 수 없기 때문에 삼차원 입체영상 효과를 얻을 수 없다. 즉 안정된 프레임 속도를 확보하기 위해서는 클러스터 시스템의 최적화와 병렬 프로그래밍을 통한 효율적인 송수신 알고리즘이 필요하다. 또한 각 노드의 그리는 버퍼가 동일한 프레임을 동시에 정확히 디스플레이해야 하므로 좌우 노드의 프레임 동기화를 위한 제어가 매우 중요하다. 이와 같은 이유로 안정적인 스테레오 가시화 클러스터 시스템(SVCS)을 구축하기 위해서는 시스템에 특화된 세부설정과 다양한 스테레오스코피 관련기법들이 적용되어야 한다.

본 장에서는 3장에서 기술한 요소기술을 바탕으로 SVCS를 구축한다. 우선 클러스터링을 위해 5개의 독립된 컴퓨터를 네트워크를 통해 하나의 시스템으로 구성한다. 스테레오스코피 영상 투시를 위해 필요한 하드웨어를 연결하여 시스템 제어와 좌우영역을 담당하는 역할을 각 클러스터 노드에 할당시킨 후, 클러스터 최적화를 위한 세부적인 하드웨어 설정을 완료한다. 다음으로 기존의 그래픽 하드웨어에 특화된 쿼드 버퍼링 기술을 소프트웨어적으로 대체시키는 방안에 대해 논한다. 즉 동일한 데이터의 그래픽 작업을 좌우 노드로 할당한 후, 각 그래픽 계산 작업이 슬레이브 노드의 더블 버퍼링을 통해 렌더링 되는 SVCS의 기본 동작 방식을 정의한다. 마지막으로 안정적인 스테레오스코피 영상을 구현하기 위해 적용한 다양한 기법에 대해 설명하고, 고성능 클러스터로의 확장기법에 대해서 논한다.

4-2. 클러스터 노드 설정

모든 노드가 로컬 하드 디스크를 갖고 있는 완전한 컴퓨터들을 노드로 묶어 구축하는 COW(Cluster of Workstation) 형식으로 모든 노드를 구성하고, Fig. 16에서처럼 사용자 디렉터리를 공유시켜 작업 노드가 모델 데이터 파일이나 응용시스템에 접근할 때 보다 빠른 성능을 발휘할 수 있다. 또한 이 방식은 모든 노드에 같은 경로로 파일을 유지하지 않아도 되어 사용자의 작업효율을 향상시킬 수 있어, 본 SVCS와 같은 연구, 설계용 시스템의 경우에 적합한 방식이다(신창균, 2006).

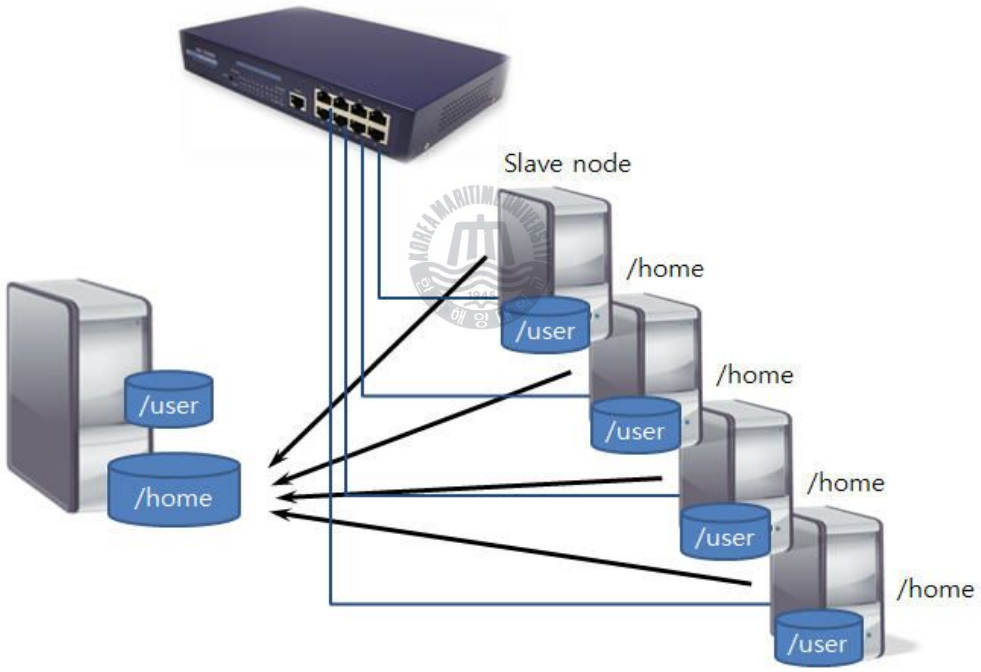


Fig. 16 Directory Structure of COW Cluster

SVCS에서 스테레오 영상을 구현하기 위해서는 좌, 우 영상을 따로 분리해야 하기 때문에 제어를 위한 노드를 포함하여 모두 홀수개의 노드가 필요하다. 즉 전체 클러스터를 총괄적으로 관리하고 모니터링 하는 마스터(Master) 노드를 기준으로 하여 좌, 우의 삼차원 모델 데이터 정보를 계산 및 디스플레이 하는 각 슬레이브(Slave) 노드를 네트워크 스위치를 통해 연결하였다. 가시화 클러스터 시스템을 주목적으로 하므로 각 노드에 GUI기반의 운영체제(리눅스 Fedora7) 및 기본운용 소프트웨어를 설치하였고, 상호 접근성, 안정성 향상을 위해 같은 수준의 커널 컴파일 및 서비스 설정을 완료하고 Fig. 17과 같은 3-노드 SVCS을 설계하였다.

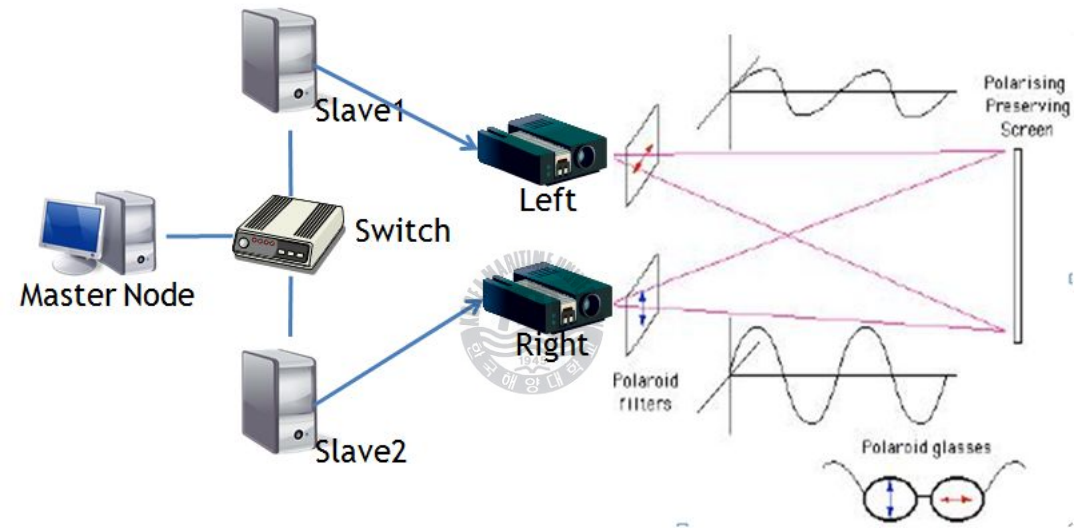


Fig. 17 3-Node Stereoscopic Visualization Cluster System

클러스터를 구성하는 하드웨어의 세부 역할 및 사양은 Table. 1에서 설명하고 있다.

Table. 1 Role & Specification of SVCS`s Hardware

Hardware	Role	Specification
Master Node	- 전체 시스템 모니터링 및 Slave제어 - 렌더링 허가 메시지 방송 (동기화) - 외부 입력 이벤트 처리	- CPU : Intel(R) Pentium(R) 4 3.00GHz - HDD : 80GBytes - RAM : 1GBytes - LAN : Gigabit
Slave Node	- 실질적인 렌더링 작업 분산처리 - 작업단위 완료 메시지 송신 (동기화) - 해당 디스플레이로 출력	- CPU : AMD Athlon(tm) 64 Processor 3200+ ×2 - VGA : nVidia Geforce 6600GE ×2 - HDD : 80GBytes - RAM : 1GBytes - LAN : Gigabit
Switch	- 각 노드를 네트워크로 연결	- 3Com Gigabit Switch 8-Ports
Projector	- 스테레오 영상 투사	- Optima EP 759 3500ANSI ×2
Display	- 스테레오 영상 디스플레이	- 120 inch / rear projection plate

4-3. 운영체제 및 커널 설치

클러스터의 운영체제는 헤드 노드가 다른 노드에게 저장소를 공유할 NFS와 rsh 등 네트워크 데몬을 구동하고 계정 하나로 모든 노드를 제어할 수 있어야 하므로 서버급 운영체제가 되어야 한다. 리눅스는 오픈소스를 지향하는 열린 운영체제로, 거의 모든 배포판이 非상용이며 기존의 유닉스 기반 클러스터 프로그램을 쉽게 이식할 수 있다. 또한 운영체제 자체를 커널 컴파일을 통해 사용자의 요구에 맞게 설정할 수 있는 장점이 있어 현재 대부분의 연구, 교육용 클러스터는 리눅스를 기반으로 하고 있다.

그래서 본 SVCS의 운영체제 또한 리눅스를 선택하였으며 그래픽모드인 X-Window를 지원해야 하므로 Fedora Core7 배포판을 모든 노드에 설치하였다. 클러스터 시스템의 최적화를 위해 최신버전의 커널(v-2.6.27.3) 을 모든 노드에 설치하고 최종 컴파일 완료하였다. 그리고 컴파일 설정 로그파일을 일정 주기로 백업하여 시스템의 안정성을 확보할 수 있도록 설정하였다.

4-4. 네트워크 및 NFS 설정 최적화

각 노드의 로컬 네트워크 주소를 수동으로 설정 해주고, 마스터 노드의 경우 외부 네트워크에 장치를 연결하여 마스터 노드의 제어 하에 전체 클러스터 시스템이 하나의 시스템으로 동작하도록 설정하였다. 각 노드의 DNS(Domain Name Server)와 Alias를 부여하여 노드 식별을 용이하게 하고, 이 모든 설정 값들을 마스터 노드의 /etc/hosts 파일에 저장하여 하위 노드와의 네트워크 연결을 완료하였다.

NFS(Network File System)는 다른 노드의 저장소를 로컬 컴퓨터에 마운트 시켜 하위 노드가 마스터 노드의 저장소를 로컬 디렉터리처럼 사용하는 프로토콜이다. NFS 활성화를 위하여 모든 계산 노드는 nfs클라이언트 데몬을 실행시켜 마운트 된 마스터 노드의 주요 디렉터리에 접근할 수 있다. NFS 프로토콜

의 속도 최적화를 위해서는 /etc/export 파일내의 요소들을 변경해 주어야 한다. 마스터 노드의 경우 /etc/export 파일내의 sync(동기모드)와 async(비동기모드)가 속도에 크게 영향을 끼치는 요소이다. sync의 경우 컨트롤러는 먼저 받은 데이터를 디스크에 기록하기 전까지 다음요청을 받지 않고 대기하게 되므로 여기서 시간 지연이 발생해 입출력 속도가 느려지는 원인을 제공한다. 하지만 async 모드로 설정하게 되면 NFS는 마스터에 데이터를 보내고 그 이후는 컨트롤러가 알아서 처리할 것으로 간주하고 지속적으로 데이터를 보내게 되어 입출력속도를 향상시킬 수 있다. 그리고 슬레이브 노드에서는 rsize, wsize, noac 요소가 속도에 크게 영향을 끼치는 요소들이다. rsize, wsize를 통해 마운트된 디렉터리에 읽고 쓰는 데이터 단위를 설정할 수 있고, noac를 설정하여 슬레이브 쪽에서 입출력을 캐시에 두었다가 전송하지 않고 곧바로 입출력을 하여 속도를 향상시킬 수 있다. 하지만 noac는 서버 측의 async와 상충하므로 둘 중 하나만 선택해야 한다.

본 SVCS에서는 NFS설정 최적화를 통해 최초 시스템 실행 시 각 계산 노드가 NFS 프로토콜을 이용하여 마스터 노드로부터 마운트 된 모델 데이터 파일 및 병렬 프로그램에 직접 접근하게 된다. 이는 결과적으로 전체 시스템 및 병렬 프로그램의 실행 속도 향상효과를 가져다 줄 수 있다.

4-5. 병렬 라이브러리 설정 및 적용

MPI는 병렬 실행 프로그램에서 노드에 분산된 프로세스 사이에 정보와 명령 등 메시지를 주고받는 방법을 정의한 규약 중 하나로써 MPI 자체는 프로그램이 아닌 표준 규약이며 이 규약에 따라 실제로 병렬 프로그램의 메시지 전달을 전담하도록 지원하는 것이 바로 MPICH이다. 클러스터 시스템에서 응용 병렬 모델을 활용하기 위해서는 MPI 패키지를 설치해야 하므로, 본 SVCS에서는 비상용버전인 MPICH를 설치하였다. 특히 MPICH의 모든 설정은 설치시의 conf 단계에서 결정되므로 이 단계에서 클러스터 시스템의 특성을 반영해주었다.

본 SVCS에는 --with-device 설정을 최적화하여 노드 프로세스 사이의 메시지 전달 방법을 정의하였다. 즉 하나의 단일 클러스터에서 단일 메모리에 상주하는 정보를 프로세스들이 나누어 계산할 때, 실제로는 노드 단위로 분리된 메모리에서 프로세스마다 필요한 데이터를 로드하게 되므로 프로세스가 다른 노드의 메모리에서 네트워크를 통해서 데이터를 입출력할 방법이 필요하게 된다. 따라서 --with-device의 ch_p4 요소를 정의해 노드 사이 통신을 담당하는 파이프 역할을 하는 sleep 프로세스를 노드마다 호출하여 이 프로세스가 일종의 네트워크 서비스 데몬으로서 메시지 전달과 실제로 실행되는 프로세스의 통제를 담당하게 설정할 수 있다. Fig. 18과 같이 마스터 노드에는 종속된 프로세스의 수만큼 sleep 프로세스가 만들어지고 다른 노드와 1:1로 연결되는 형식을 갖게 됨으로서 원활한 데이터 입출력 파이프를 구축할 수 있게 된다.

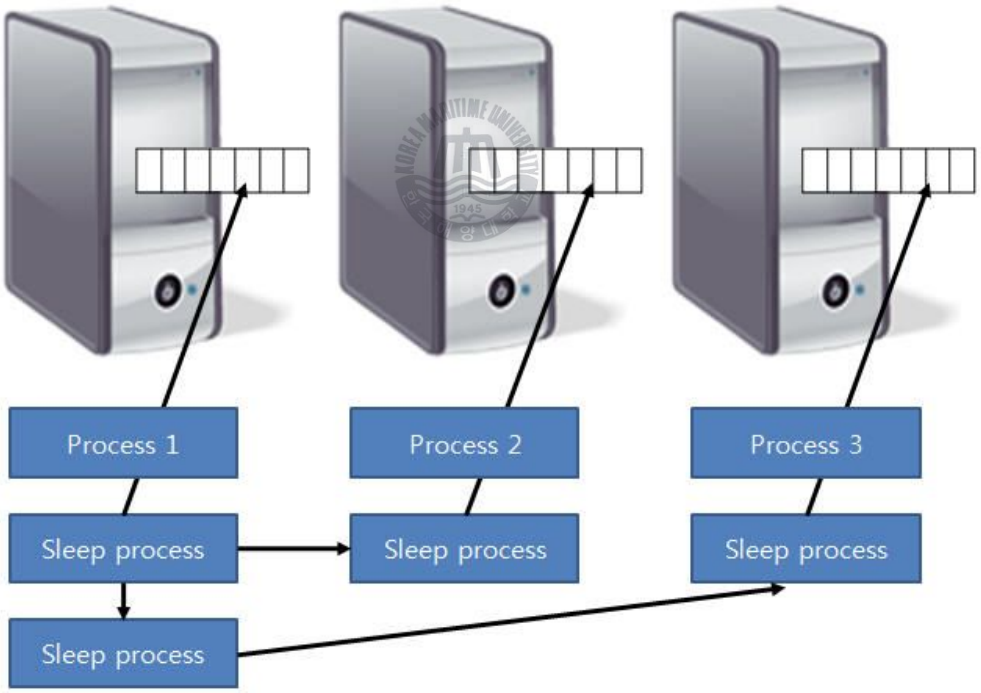


Fig. 18 Message Passing between the processes

본 시스템에서 MPI 병렬 라이브러리의 쓰임은 크게 세 가지로 나뉜다.

우선 프로세스 랭크 함수를 사용해 동일 프로그램 내에서 프로세스의 식별을 위해 각 노드마다 프로세스 번호를 순차적으로 부여 받을 수 있다. 사용자는 부여 받은 번호를 통해 각 노드를 명시적으로 제어할 수 있고, 노드 사이의 통신에서도 종착점의 식별을 위해 사용된다.

그리고 MPI의 논블록킹 통신 기반의 표준송수신 함수를 사용해 프레임 동기화를 제어하고, 데이터를 분산, 취합하는 프로세스의 진행과정을 구현할 수 있다. 여기서 논블록킹 통신이란 노드간의 통신이 시작되면 그 작업의 완료여부와 상관없이 결과 값이 리턴 되고, 이후 대기 또는 검사 함수를 사용해 인위적으로 완료여부를 검사하여 사용자가 프로세서를 효율적으로 제어할 수 있는 통신방법이다. 즉 시스템 내의 교착 가능성을 제거하고 통신 부하를 감소시킬 수 있어 본 SVCS의 내부 통신을 효율적으로 제어할 수 있다.

마지막으로 점대점, 집합통신 함수를 사용해 시스템 내에서 통신되는 데이터를 효과적으로 송수신, 분산, 취합할 수 있다. 점대점 통신은 Fig. 19에서 보듯 노드간의 일대일 통신을 뜻하며 메시지 전송에 이용된 메모리 위치에 안전하게 접근할 수 있기 때문에 안정적으로 통신이 가능하다. 하지만 점대점 통신에서 양방향 송수신을 실시할 때는 프로세스가 교착하는 경우가 많고, 디버깅 시에도 명시적으로 표시되지 않기 때문에 프로그래머는 프로세스의 진행에 세심한 주의를 기울여야 한다. 그리고 집합통신은 노드간의 일대다 통신을 뜻하며 그 종류에는 Fig. 20에 나타낸 것처럼 방송, 취합, 환산, 확산, 장벽 등 많은 병렬 통신 API가 존재한다. 특히 점대점 통신에 비해 편리하고 성능 면에서 최적화되어 빠르기 때문에 집합통신 루틴을 잘 이용하면 효율적인 통신 알고리즘을 구현할 수 있다. 하지만 집합통신은 논블록킹 루틴이 없어 본 시스템에서의 프레임 동기화시 점대점 통신과 병행하여 사용해야 하는 단점이 있다.

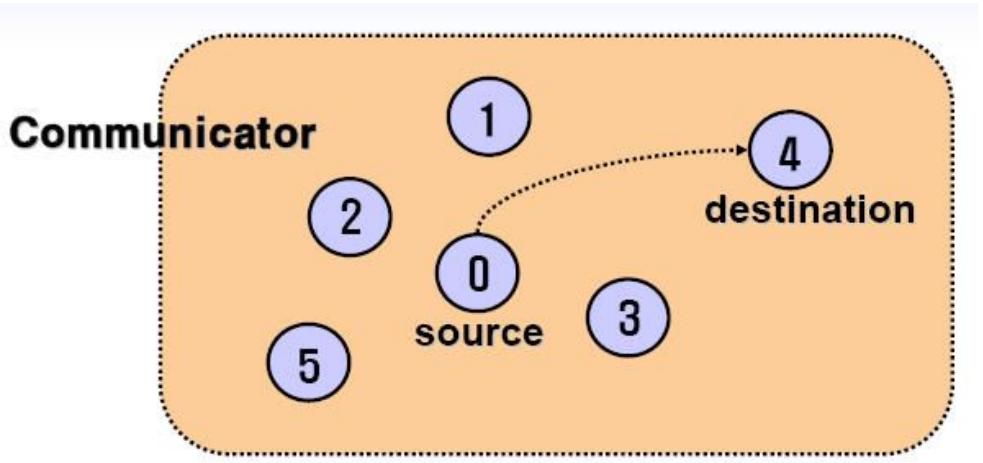


Fig. 19 Point to Point Communication on MPI (MPI, 2008)

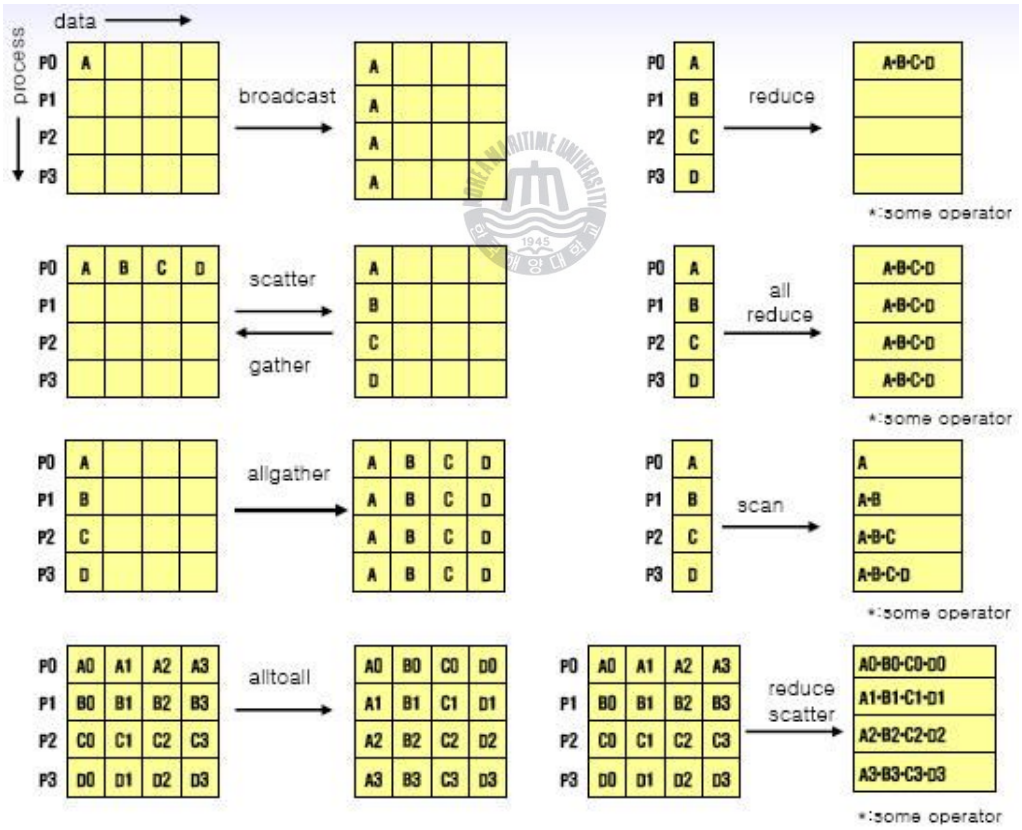


Fig. 20 Concurrence Communication on MPI (MPI, 2008)

4-6. 프레임 동기화 구현

스테레오 영상의 프레임 동기화를 구현하기 위해서는 쿼드 버퍼링 기술이 필요하다. 본 시스템에서는 Fig. 21과 같이 네 개의 프레임 버퍼를 디스플레이를 담당하는 각 노드가 분담해 각각의 렌더링 작업을 독립적으로 처리하였다.

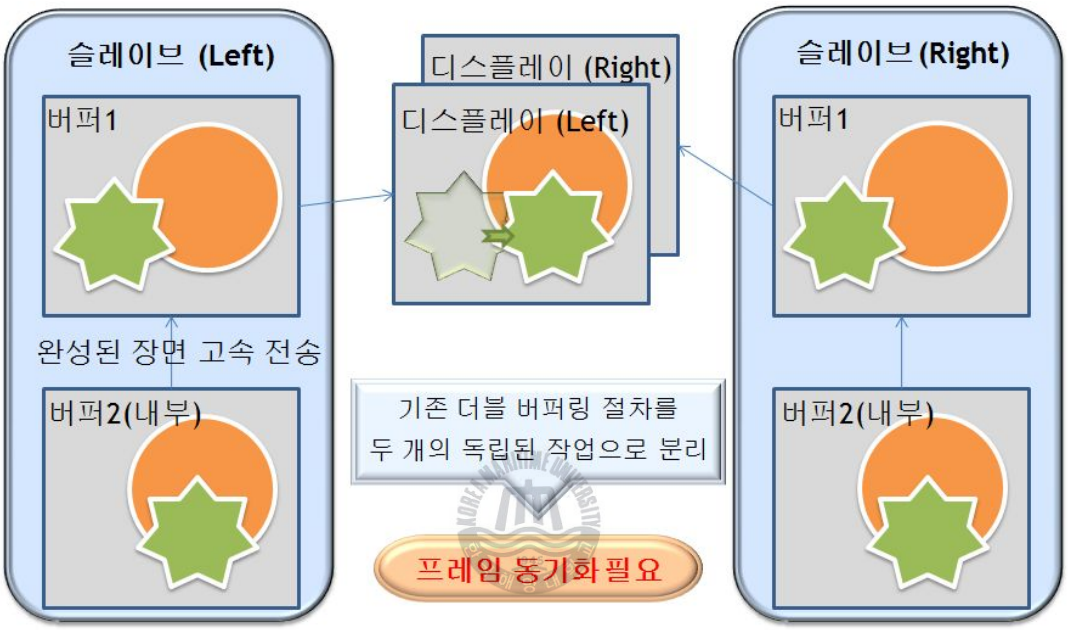


Fig. 21 Quad Buffering and Synchronization of SVCS

클러스터 시스템에서 스테레오 영상을 구현하기 위해서는 프레임 동기화가 꼭 필요하다고 전술한 바 있다. 어떤 가시화 시스템보다 정밀하고 신속한 동기화가 필수적이나 프레임수와 역비례관계를 가지는 현실적인 문제가 있어 서로 간의 타협이 필요하게 된다. 즉 스테레오스코픽 영상을 구현하기 위해 SVCS 각 노드의 작업을 한 프레임 단위로 처리하려면 내부적으로 같은 단계의 프레임을 제어하는 루틴, 즉 프레임 동기화 작업이 반드시 필요하다.

병렬 프로그래밍 함수인 MPI_Barrier()를 사용해 프레임 동기화를 시도했을 경우 복잡한 데이터를 처리할 때 초당 5프레임 이하로 프레임율이 하락하는 경

우가 쉽게 관찰되었다. 또한 계산노드를 확장했을 경우 네트워크에 부하가 걸려 메시지 통신이 간헐적으로 지연되는 현상을 보였다. 스테레오스코픽 시스템의 경우 초당 5프레임 이하로 영상이 주사되면 사람의 뇌에 연속적인 동작에 대한 착시를 일으킬 수 없고, 양안으로 들어오는 화상정보가 어긋나버리기 때문에 영상의 부드러움을 떠나 스테레오스코피 효과 자체를 느낄 수 없게 된다 (Olson, E, 2002).

이와 같은 MPI 동기화 API의 한계를 극복하고자 상대적으로 지연속도가 빠른 MPI 점대점 통신과, 집합통신을 활용해 단일 신호 메시지를 송수신하고 프레임 렌더링작업 직전마다 임의로 블록킹을 해주는 방법으로 프레임 동기화 알고리즘을 구현하였다. 나아가 앞서 설명한 NFS 서버의 async모드를 이용해 클러스터 시스템 내에서 네트워크 속도를 최적화하여 동기화에 의한 프레임 지연을 최소화하는 효과를 얻을 수 있었다. 아래는 SVCS 응용 병렬 시스템 내부의 일련의 프레임 동기화 작업을 서술한 명세이다.

1. *프로그램이 실행되면 각 노드는 미리 저장되어 있는 데이터를 마스터 노드로부터 마운트된 자신의 디렉터리에서 직접 읽은 뒤 마스터 노드에게 확인 신호를 전송한 후 대기한다.*

2. 각 슬레이브 노드들로부터 확인 신호를 수신 대기하던 마스터 노드는 자신의 작업이 끝남과 동시에 슬레이브 노드들로부터 모든 신호가 도착했는지 검사한다.

3. 모든 신호가 도착했다면 마스터 노드는 프레임 버퍼 스와핑을 위한 스왑 신호를 브로드캐스팅하고, 자기 자신도 모니터링을 위해 버퍼 스와핑을 수행한다. 대기하고 있던 슬레이브 노드들도 스왑신호를 수신하는 즉시 버퍼 스와핑을 수행한다.

이로써 한 프레임에 대한 동기화가 이루어지고, 렌더링 루프를 통해 반복되면서 모든 노드의 모든 프레임에 대한 동기화가 이루어지게 된다. 그리고 만약 모니터링 노드인 마스터 노드에서 마우스나 키보드 등에 의한 외부 이벤트가 발생하면 프로세스는 루프 처음으로 돌아가 각 슬레이브 노드로 외부 이벤트 데이터를 전송한다.

메시지 전달에 의한 프레임 동기화 기법의 절차를 도시화하면 Fig. 22와 같다.

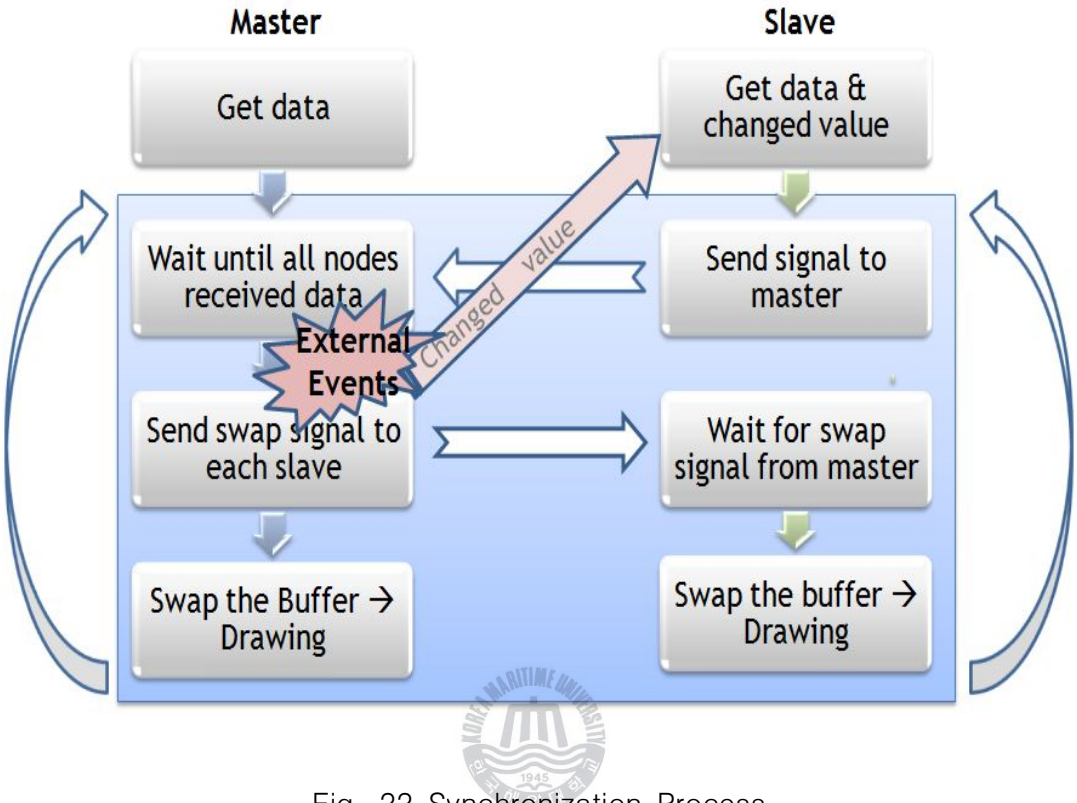


Fig. 22 Synchronization Process

4-7. 데이터 분산처리

단지 스테레오스코피를 구현하는 것이 목적이라면 하나의 마스터 노드와 두 개의 슬레이브 노드로 구성되는 3-노드 클러스터만으로도 충분히 구현 가능하다. 하지만 클러스터링은 전체 작업을 시스템 내의 계산노드로 분산시켜 HPC를 실현하는데 그 의미가 있다. 3-노드 SVCS에서는 마스터 노드가 전체 시스템을 제어하고, 두 개의 슬레이브 노드는 자신에게 부여된 그래픽 작업만을 계산하고 디스플레이하는 역할을 수행할 뿐, 데이터 분산처리와 같은 병렬 컴퓨팅을 수행하지 않는다. 즉 이 시스템은 엄밀히 말해 데이터 차원의 병렬처리가 아니며 연산속도 상승과 같은 클러스터링의 효과를 얻을 수 없다.

즉 시스템의 계산 성능을 향상시키려면 클러스터 시스템의 확장이 필요하며, 필요한 하드웨어를 추가하고 클러스터 최적화 설정, 그리고 별도의 병렬처리 프로그래밍을 통하여 HPC 시스템을 구현할 수 있다. 병렬 프로그램 내부의 메시지 전달식 데이터 분산, 취합 기법의 절차는 Fig. 23과 같이 간단히 도시화 할 수 있을 것이다.

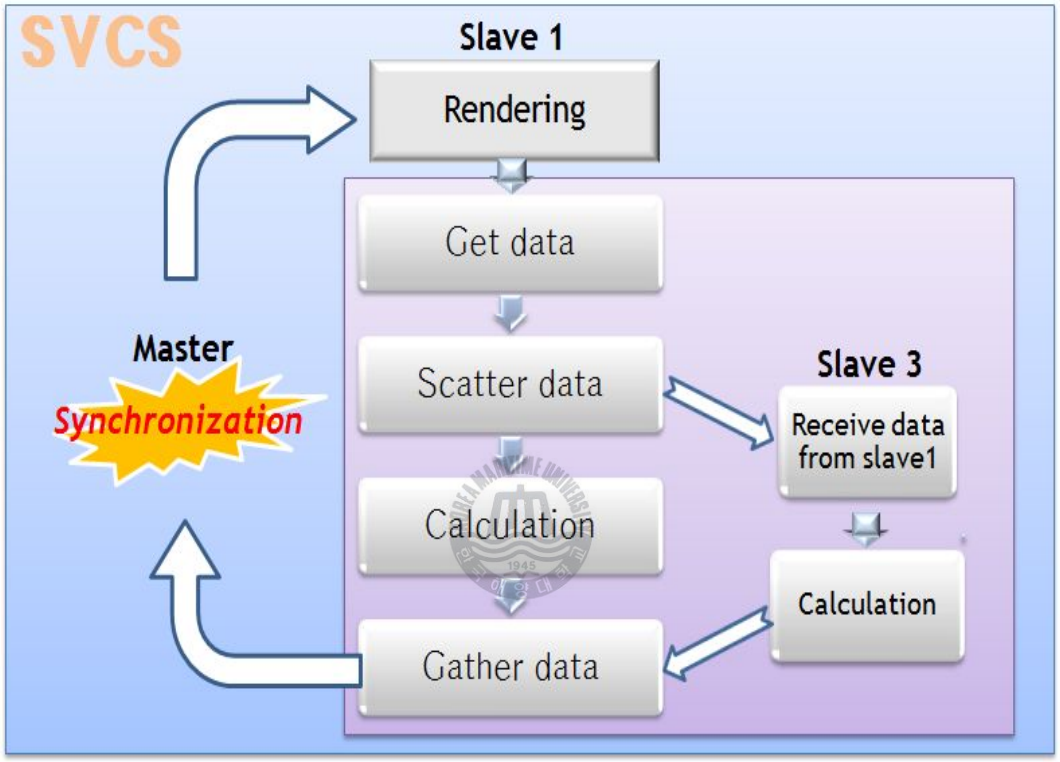


Fig. 23 High Performance Computing Process

본 논문에서는 연구 여건상 5-노드 SVCS를 실질적으로 구현하지 못했기 때문에 이 절을 통해 데이터 분산처리에 대한 개념과 구현방식에 대해 간단하게나마 설명하였다. 데이터 분산처리에 대한 세부적인 개념과 처리방법은 Appendix C에서 참고할 수 있다.

4-8. SVCS 구동

Fig. 24는 구동된 SVCS의 전경이다.

Fig. 24 Stereoscopic Visualization Cluster System

5. 통합 가시화 GUI 환경 구축 및 시스템 성능 검증

5-1. 개요

SVCS의 성능을 검증하기 위해 이를 활용할 수 있는 응용 소프트웨어가 필요하다. 4장에서 기술한 스테레오스코피 가시화 요소기술의 대체기법을 활용하여 기본적인 X-Window 기반의 콘솔 어플리케이션을 구현한다. 어플리케이션의 병렬 프로그래밍과 렌더링 처리에 대한 동작 테스트를 마친 후 본격적으로 통합 가시화 GUI환경을 구축한다. GUI환경을 구축하기 위해 크로스 플랫폼을 지원하는 Code::Blocks 통합개발환경과 wxWidget GUI Toolkit을 시스템에 설치한다. 기존 X-Window 기반의 콘솔 어플리케이션을 GUI 환경으로 포팅하고, 일반적인 CAD 시스템의 기능을 이식하여 통합 가시화 GUI환경을 구축한다. 최종적으로 본 소프트웨어를 활용해 SVCS을 다양한 각도에서 검증 테스트 하여 그 유용성을 가늠해보았다.

5-2. GUI 시스템 개발 환경

5-2-1. 통합개발환경(IDE)

통합개발환경(Integrated Development Environment, 이하 IDE)은 코딩, 디버깅, 컴파일, 배포 등 프로그램 개발에 관련된 모든 작업을 하나의 프로그램 안에서 처리하는 환경을 제공하는 소프트웨어이다. 다양한 언어를 지원하는 IDE는 프로그래밍 언어의 수만큼이나 많다. 본 SVCS의 GUI 통합 환경 구축을 위해서 오픈소스, 크로스 플랫폼, C++언어를 지원하는 Code::Blocks를 설치하였다. Code::Blocks는 Windows, Mac, Linux와 같은 다양한 플랫폼에서의 개발을 지원하는 IDE로서 다른 IDE에 비해 가볍고, 사용자의 요구가 수시로 반영되어 릴리즈 되는 특징이 있다(CodeBlocks, 2008). 특히 Code::Blocks는 wxSmith라는 플러그인이 자체 내장되어 오픈 GUI toolkit인 wxWidget에 최적

화되어 있고, 나아가 다양한 컴파일러를 통한 작업이 손쉽게 가능하기 때문에 최근 많은 크로스 플랫폼 기반의 소프트웨어 개발자들이 선호하고 있는 IDE중 하나이다. Fig. 25는 본 SVCS의 통합 GUI 환경을 구축하기 위한 Code::Blocks의 개발환경 전체 프레임이다.

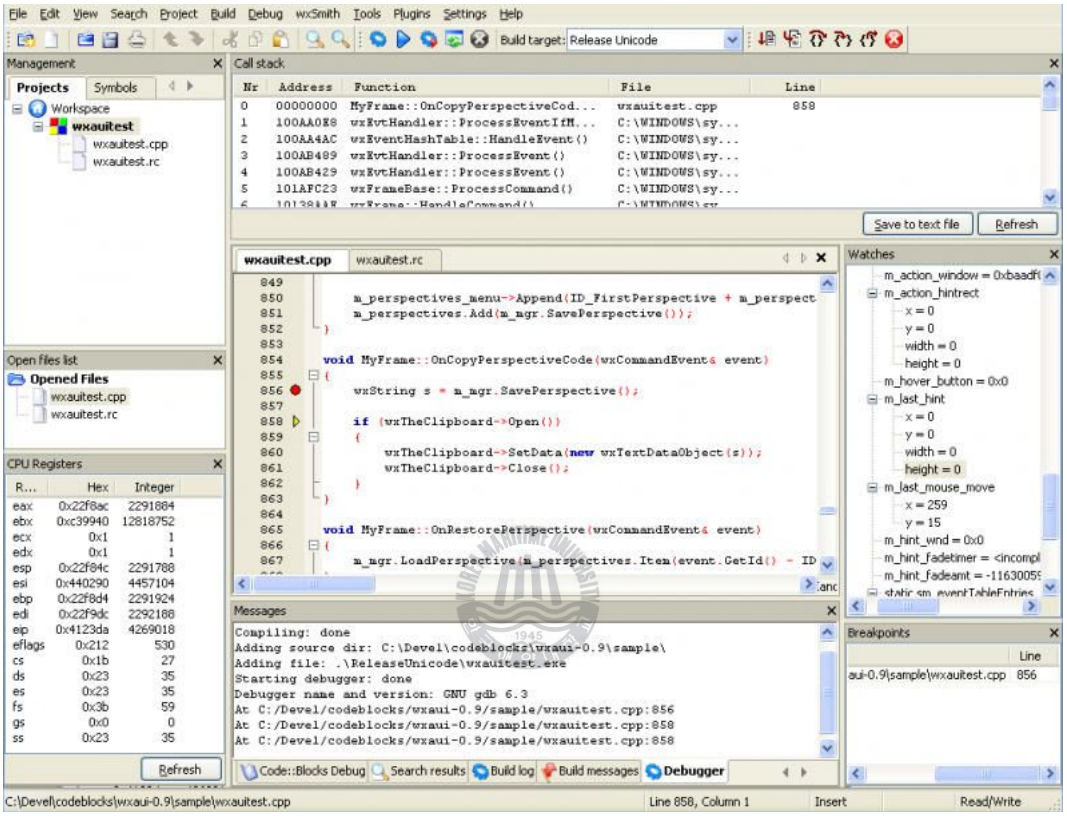


Fig. 25 Integrated Development Environment - Code::Blocks

5-2-2. GUI Toolkit

가시화 시스템의 경우 그래픽 사용자 인터페이스(Graphic User Interface, 이하 GUI)는 필수적으로 요구되는 사항이다. 여기서 GUI는 Fig. 26과 같이 SVCS가 사용자의 명령을 전달받아 그 명령을 해석하고, 실행되면서 현재 시스템의 상태를 사용자에게 전달하는 역할을 한다. 즉 GUI의 주요 목적은 사용자

가 시스템으로 보다 쉽고 편리하게 접근 가능 하도록 만드는 것이다.

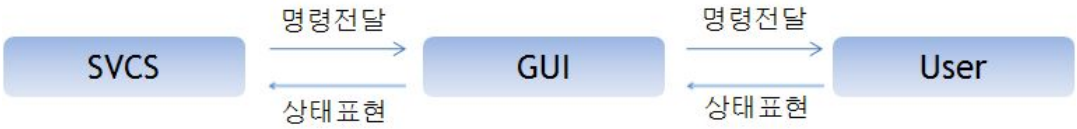


Fig. 26 A Central role of Graphic User Interface

SVCS의 GUI 통합 환경을 구축하기 위해 Code::Blocks와 함께 wxWidget을 사용하였다. wxWidget은 Code::blocks의 3rd-party 크로스플랫폼 라이브러리로서, 서로의 호환성이 좋을 뿐만 아니라 GUI 환경을 구축하는데 필요한 방대한 API를 IDE에 제공한다. 즉 거의 모든 응용프로그램을 개발할 수 있는 풍부한 프레임워크이기 때문에 특정 운영체제에 독립적인 프로그램을 개발하는데 최적의 조건을 갖추고 있다. 특히 Table. 2에서 알 수 있듯 wxWidget은 Windows MFC 어플리케이션의 클래스 구조를 기반으로 하고 있기 때문에 기존의 MFC에서 구축되었던 어플리케이션을 wxWidget 기반으로 포팅 하는 최적화된 루틴을 자체적으로 제공하고 있다.(wxWidget, 2008)

Table. 2 Class Comparison between MFC and wxWidget

Class	MFC class	wxWidget class
Document	CDocument	wxDocument
View	CView	wxView
Edit view	CEditView	n/a
Template class	CMultiDocTemplate	wxDocTemplate
MDI parent frame	CMDIFrameWnd	wxDocMDIParentFrame
MDI child frame	CMDIChildWnd	wxDocMDIChildFrame
Document manager	n/a	wxDocManager

그리고 통합 가시화 환경의 그래픽 작업을 위해 OpenGL(Open Graphic Library)이 필요하다. OpenGL은 그래픽스 하드웨어에 대한 소프트웨어 인터페이스로서, 대화형 삼차원 어플리케이션 제작에 필요한 오브젝트나 연산을 구성하는데 사용할 수 있는 다양한 API로 구성되어 있다. OpenGL은 복잡한 객체를 모델링할 때 오직 점, 선, 폴리곤 등과 같은 기하학적 원형요소(Geometric Primitives)만 사용할 수 있는 저수준의 그래픽 라이브러리이다. 즉 프로그래머에게 높은 자율성을 부여함으로써 가시화 시스템에서의 그래픽 활용 범위를 크게 높일 수 있다.

또한 wxWidget 환경에서 OpenGL을 통해 삼차원 그래픽을 표현할 수 있다(Julian S, 2008). 비록 wxWidget 자체에도 다양한 그래픽 API가 존재하지만, Fig. 27과 같이 복잡한 삼차원 모델을 점, 선 등으로 표현하기 위해서는 OpenGL과 같은 저수준 그래픽 라이브러리의 연결이 필요하다. wxWidget과 OpenGL을 연결하기 위해서는 wxGLCanvas와 wxGLContext 클래스가 필요하다. 여기서 wxGLCanvas는 OpenGL 그래픽을 디스플레이 하는데 필요한 함수들을 제공하는 클래스이며, wxGLContext는 OpenGL과 해당 시스템의 연결 상태를 표현하는 상태함수의 클래스이다. 상기 두 클래스를 활용하여 통합 가시화 환경의 그래픽 작업을 수행할 수 있다.

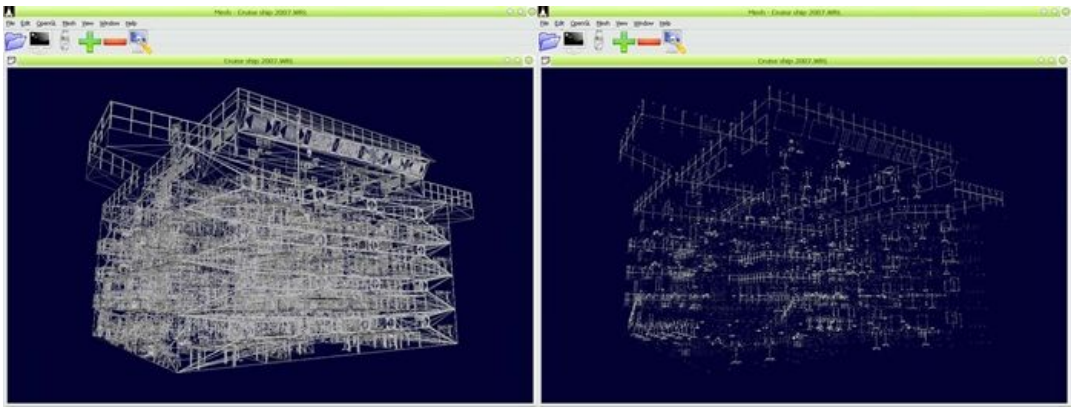


Fig. 27 3D Model presented by Lines and Points

5-3. 가시화시스템 전용 GUI 통합 환경 구축

앞서 전술한 스테레오스코피 가시화 요소기술의 대체기법을 바탕으로 가시화 시스템 전용 GUI 통합 환경을 구축하였다. 본 프로그램은 삼차원 오브젝트 모델의 import와 export가 가능하고, vrml, 3ds, stp 등의 데이터포맷을 지원한다. Fig. 28은 GUI 통합 환경의 메인프레임부와 모델 구현부를 나타낸다.

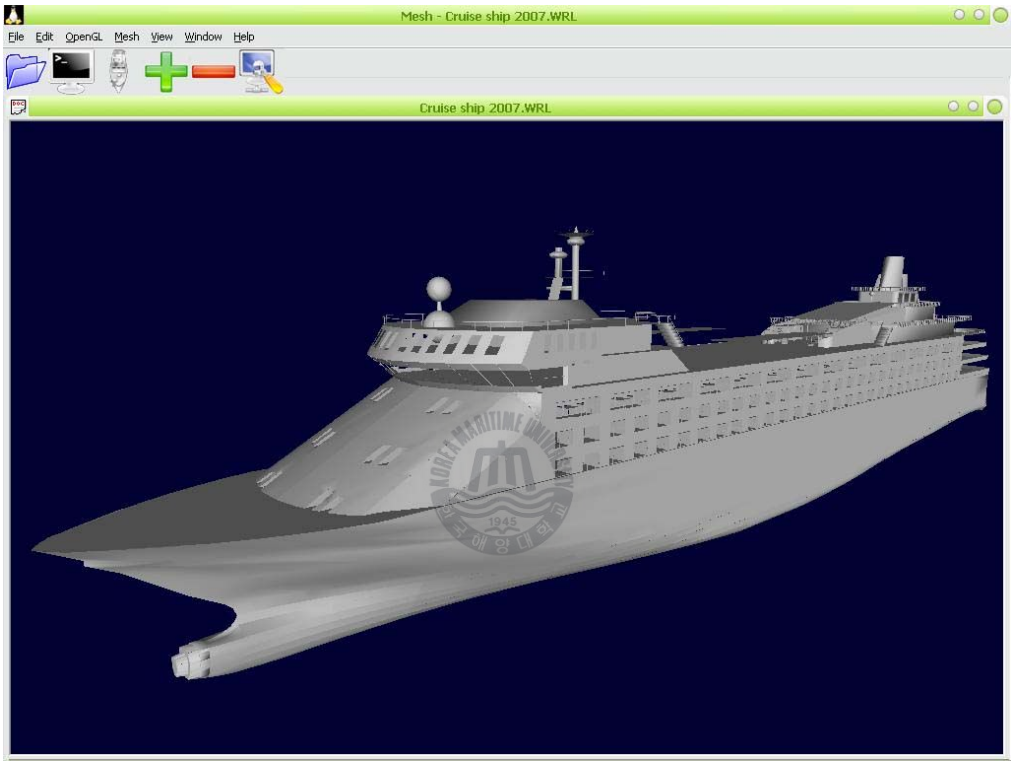


Fig. 28 Main Frame of GUI Integrated Environment on SVCS

메인프레임부에서는 통합 가시화 GUI 환경의 모든 기능이 메뉴와 툴바로 구성되어 삼차원 모델의 회전, 이동, 확대, 축소와 OpenGL 세부기능, 클러스터링, 스테레오 제어기능이 GUI를 통해 쉽게 구현 가능하다. 모델 구현부에서는 스테레오스코피 가시화를 위한 삼차원 모델 오브젝트 객체가 OpenGL을 기반으로 렌더링 된다. 그리고 마우스, 데이터 글러브 등의 외부 입력장치의 신호값을 프로그램으로 전달해 주는 역할을 수행한다.

Fig. 29는 GUI 통합 환경의 제어 패널부를 나타낸다.

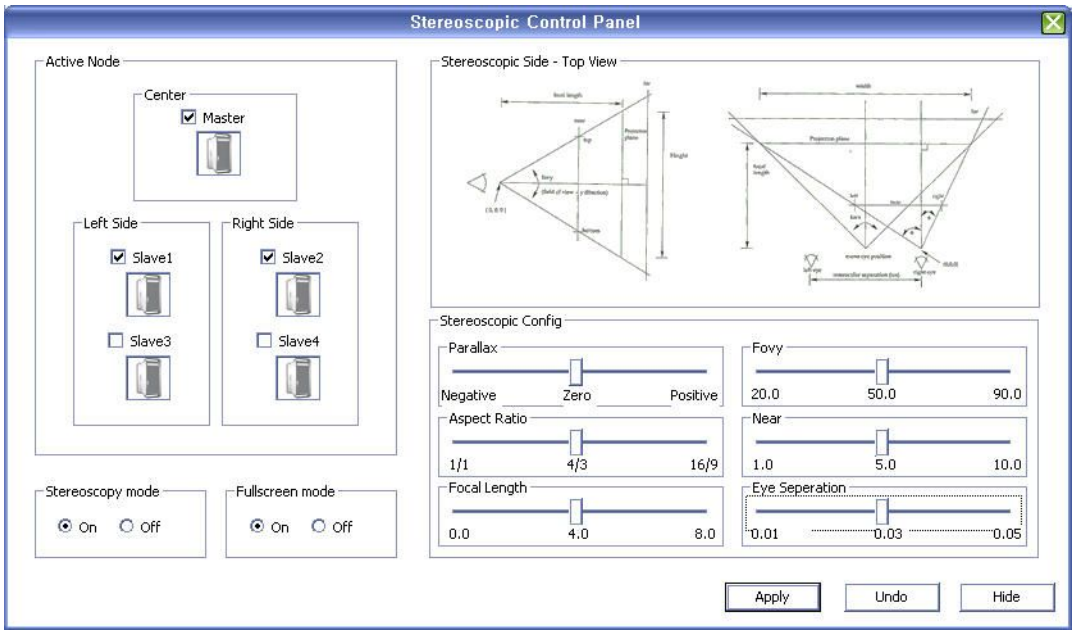


Fig. 29 Control Panel of GUI Integrated Environment on SVCS

제어 패널부에서는 SVCS에서의 클러스터링과 스테레오스코피 구현에 필요한 각 설정 값들을 제어할 수 있다. 제어 패널 부 내에서 직접 활성화할 노드를 선택하게 되면 곧바로 각 노드가 프로그램 실행에 포함되어 노드 확장시 프로그램을 재시작할 필요가 없다. 그리고 스테레오스코피 효과를 담당하는 각 요소를 마우스 드래그를 통해 제어할 수 있도록 설계하여 사용자가 느끼는 입체감을 개인의 취향에 따라 쉽게 변경할 수 있도록 하였다. 그 외에 스테레오스코피 효과를 제거할 수 있는 Stereoscopy 모드, 삼차원 모델 구현부를 전체화면으로 전환하는 Fullscreen 모드 등을 추가하였다. 제어 패널부는 모달리스 다이얼로그 기반으로 제작되어 변경 값이 실시간으로 모델 구현부에 반영되는 특징이 있다.

5-4. SVCS 검증 및 성능 분석

5-4-1. 개요

시스템 검증을 위한 수행 항목은 다음과 같다.

- 고성능 그래픽 하드웨어 시스템과의 성능비교
- 동기화 성능 테스트
- 스테레오스코피 및 몰입감 비교

5-4-2. 고성능 그래픽 하드웨어 시스템

특정 기술에 특화된 고성능 그래픽 하드웨어는 고성능 GPU를 탑재하여 그래픽 작업 시에 탁월한 성능을 발휘할 수 있다. 스테레오스코피를 비롯해 각종 시뮬레이션 가시화를 위한 넓은 시야의 삼차원 그래픽 영상 재현, 다채널(multiple display channel) 이미지 재생을 위한 다중복합 디스플레이 지원 등 고성능 그래픽 하드웨어가 지원하는 다양한 기술력 덕분에 다양한 목적의 가시화가 가능하다. 최근에는 nVidia 社의 주도하에 클러스터링 기법을 응용한 GPU 클러스터링, SLI, CUDA 언어환경 등이 개발되어 그 성능을 확장하고 있다. 특히 SLI는 두 개 이상의 그래픽 하드웨어를 직렬로 연결하는 기법으로서 스테레오스코피와 같은 특화된 기술을 지원할 뿐만 아니라, 단일 플랫폼의 멀티뷰, 멀티 플랫폼의 모자이크 모드 등 가시화를 위한 성능을 극대화시킬 수 있다.

본 SVCS 구축에 앞서 단일 쿼드로 FX4500 그래픽카드를 장착한 스테레오스코피 전용 플랫폼인 쿼드로 머신(Single High Performance Graphic System, 이하 SHPGS)을 본 연구실에서 구축한바 있다. 이 SHPGS를 이용해 스테레오스코픽 영상을 재현하여 가상현실 시스템 환경에서 많은 관련연구가 수행되어져 왔다. 본 논문에서는 SVCS와의 성능비교를 위해 SHPGS를 구동한다.

5-4-3. SVCS 성능 측정

SVCS의 성능을 초당 프레임 레이트(Frames Per Second, 이하 FPS) 측정을 통한 프레임 연산 속도관점에서 비교하여 가시화 시스템으로서의 적합성을 검증한다. 시스템의 성능을 비교분석하는 차원에서 앞서 기술된 고성능 그래픽 하드웨어 시스템과의 비교를 우선적으로 수행하였다. 물론 해당 시스템 하드웨어의 성능이 동일하지 않아 절대적인 비교를 수행할 수 없었지만, 어느 정도의 성능지표를 산출한다는 점을 염두에 두고 참고적으로 활용하였다.

각 시스템의 제원은 아래 Table. 3과 같다.

Table. 3 Comparison between SVCS and SHPGS

		SVCS	SHPGS
CPU		Intel(R) Pentium 4 3.00GHz × 1 AMD Athlon(tm)64 3200+ × 2	Pentium D CPU 3.00GHz
VGA		nVidia Geforce 6600GE 128M × 2	nVidia Quadro FX4500 512M
	제조사	nVidia	
	GPU 동작클럭	400MHz	
	메모리 스펙	GDDR3 SDRAM 128MB	GDDR3 SDRAM 512MB
	메모리 버스	128bit	256bit
	메모리 동작클럭	450(900)MHz	525(1050)MHz
	출시가	12만 원대	250만 원대
Support		Practical Application	OpenGL Stereo API Stereoscopic driver
Model		Cruise ship model (200,000 vertices. The number of vertices can be flexibly increased or decreased for the test.)	

테스트에 앞서 우선 그래픽 하드웨어 각각의 OpenGL 연산 능력을 알아보기 위해 SPECviewperf 8.1을 활용할 수 있다. SPECviewpert는 OpenGL에서 지오메트리 연산능력을 측정하는 벤치마크 프로그램으로 Fig. 30과 같이 3DsMax를 비롯한 Catia, ensight, Light Wave, Maya, PRO-E, SolidWorks, UGS등 총 8개 AP 툴을 기반으로 완성되어 있다. 8가지 프로그램 모두 OpenGL을 사용하며 광원 처리, 쉐이딩 등을 통한 렌더링 성능을 측정하여 서로 객관적인 수치로 비교할 수 있기 때문에 성능 비교에 널리 사용되는 프로그램이다.

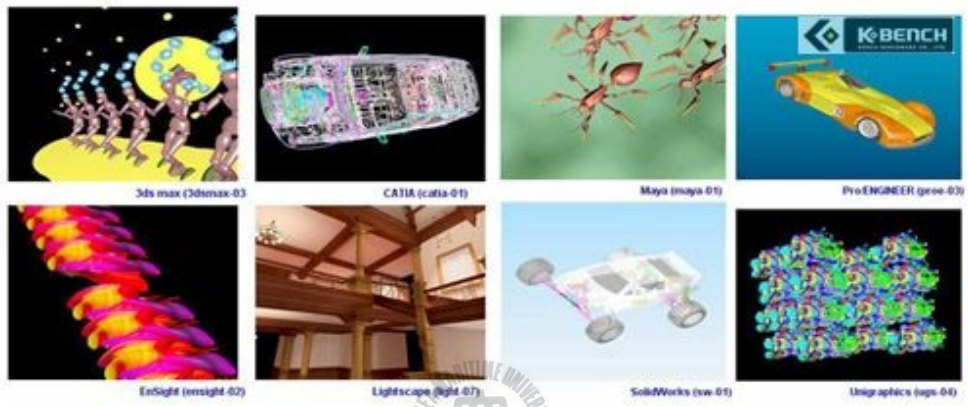


Fig. 30 8-Programs, being used in SPECviewperf 8.1

각 그래픽 하드웨어의 OpenGL 연산능력 테스트 결과는 Fig. 31과 같다.

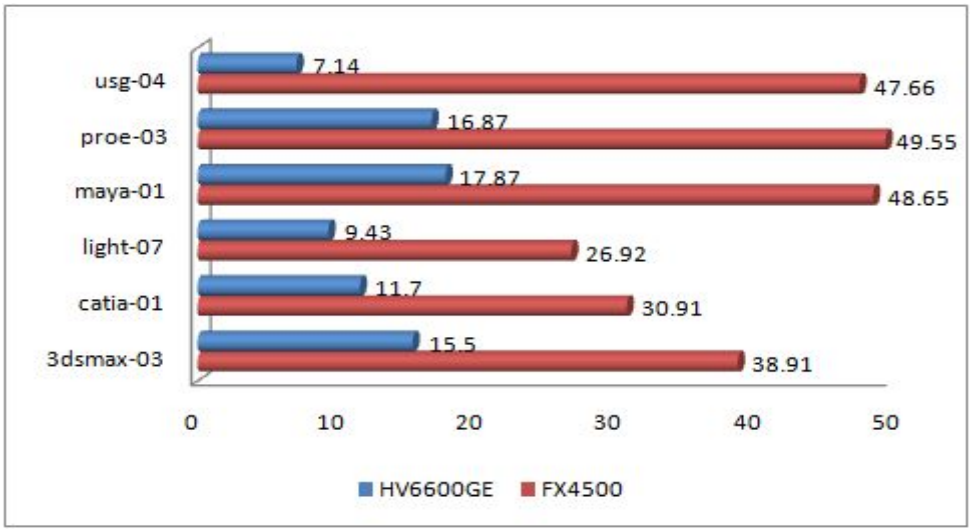


Fig. 31 nVidia 6600GE VS Quadro FX4500 by SPECviewperf 8.1

SPECviewperf 테스트 결과 3dsmax나 catia같은 각종 전문가용 3D 프로그램에서 작게는 두 배에서 많게는 다섯 배 이상으로 Quadro FX4500이 월등한 수치를 보여주는 것을 확인할 수 있다. 또 하나의 고려할 사실은, 본 연구에서 개발된 시스템의 범용 소프트웨어는 일반적으로 고성능 그래픽 하드웨어의 전용 드라이버에 비해 성능이 떨어질 수밖에 없다는 것이다. 드라이버는 운영체제와 하드웨어 사이에서 구동되는 미들웨어(Middleware)이기 때문에 아무래도 운영체제와 미들웨어를 다 거쳐서 하드웨어에 신호를 전달해야 하는 본 시스템보다는 작동이 훨씬 효율적이다. 체감 적으로도 일반 범용 소프트웨어는 전용 드라이버에 비해 상당한 FPS 저하를 느낄 수 있다(여진욱, 2005).

이 같은 사실을 통하여 SVCS가 성능을 최대로 발휘하는 기준은 SHPGS 성능의 절반 이하 수준이라는 사실을 예측할 수 있었다. 이를 사전에 염두에 두고, SVCS 성능을 고성능 그래픽카드를 내장하고 있는 연구실의 SHPGS와 비교 테스트하여 시스템의 유용성을 확인하였다.

테스트는 200,000 Vertices 로 모델링된 크루즈 선을 데이터 모델로 선정하고 이 모델이 각 시스템에서 1024X768 해상도 기준으로 렌더링 될 1분 동안의 최고, 최저, 평균 FPS를 측정하는 방식으로 진행되었다. 여기서 FPS는 가시화 시스템의 성능을 평가하는 척도로 사용된다. 가상현실 시스템의 조건을 만족하는 FPS는 명확히 정의되진 않았지만 일반적으로 15프레임 이상의 FPS 성능을 낼 수 있어야 하고, 또한 개인차가 있지만 스테레오스코피 효과는 5 FPS 이하에서는 느낄 수 없다고 보고된 바 있다(여진욱, 2005).

본 테스트에서는 Unix C언어 기반의 Time API를 활용해 직접 FPS 함수를 구현하여 각 시스템의 응용 소프트웨어에 적용하였다. 그리고 SVCS에서는 본 연구에서 구현한 응용 병렬 소프트웨어를, SHPGS에서는 OpenGL 스테레오 지원 API를 통해 MFC로 구현한 응용 소프트웨어와 스테레오스코픽 전용 드라이버를 각각 시스템 지원 모듈로 설정해 주었다.

테스트 결과 주목할 만한 사실은 본 테스트 환경에서 SVCS와 고성능 그래픽 플랫폼의 FPS를 비교해 보았을 때 사전 두 배 이상의 상당한 성능차이를 고려했던 예상보다 그 차이가 크지 않았다는 점이다(Fig. 32).

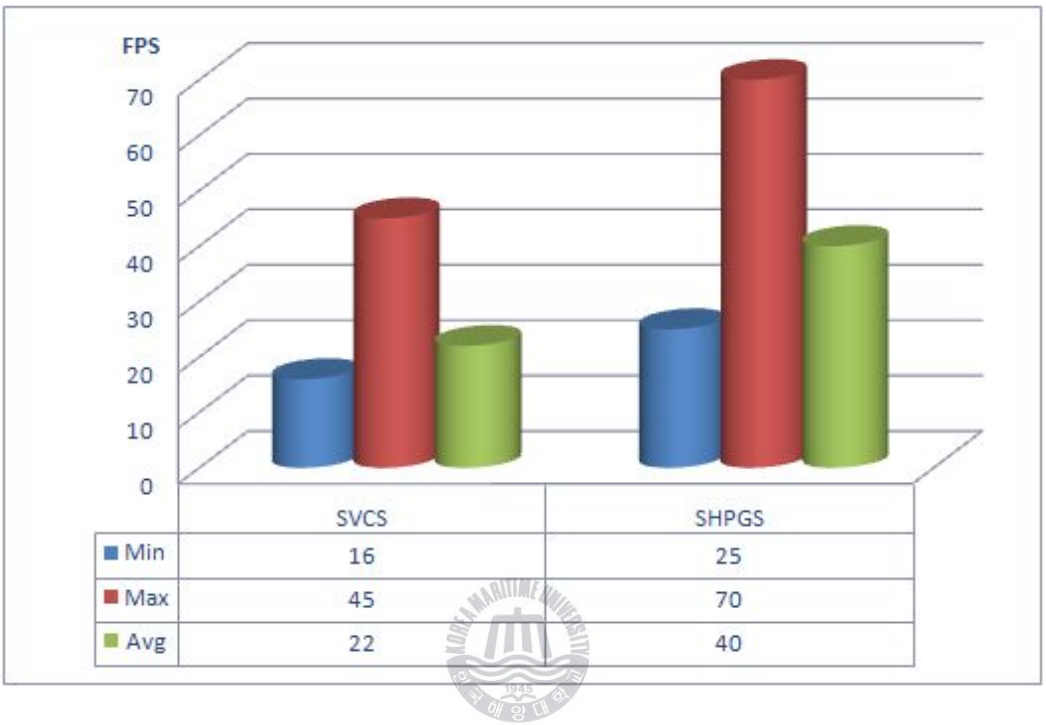


Fig. 32 Frame Rate Chronometry - SVCS vs SHPGS

물론 해당 시스템 하드웨어의 성능이 동일하지 않아 절대적인 비교를 할 수 없고, SHPGS에 비해 SVCS가 성능이 떨어지는 것은 명백한 사실이다. 하지만 SVCS의 표면적인 성능 측정 결과로만 본다면 본 테스트 환경에서 평균 22 FPS 결과를 보인 SVCS는 스테레오스코피 전용 시스템으로서의 역할을 충분히 수행 가능한 가시화 시스템임이 확인되었다.

SVCS의 성능을 보다 구체적으로 파악하기 위해 보다 복잡한 삼차원 모델을 검증대상으로 선정하였다. 즉 모델 데이터양에 따른 SVCS의 성능을 검증하기 위해 기존 검증 모델의 약 두 배인 500,000 vertices (40Mbytes) 크루즈 선박 모델을 SVCS와 SHPGS에서 렌더링하고 이때의 FPS를 측정하였다(Fig. 33).

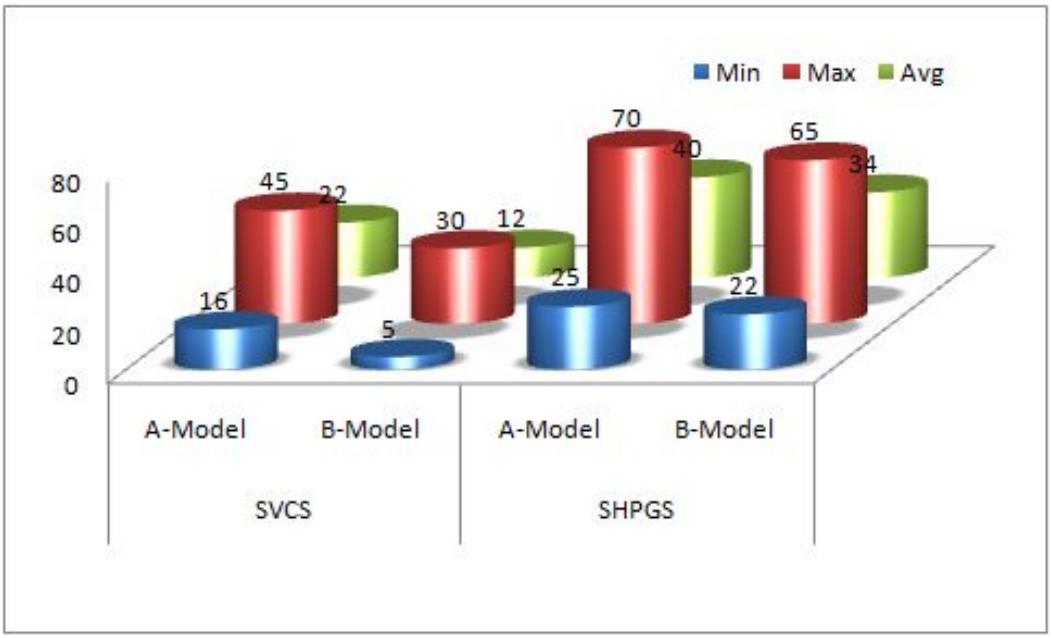


Fig. 33 Frame Rate Chronometry - A Model vs B Model

그 결과 SVCS의 경우 복잡한 B-모델의 최소 FPS가 최저 5프레임까지 떨어져 스테레오스코픽 효과를 느낄 수 있는 최저 FPS를 만족시키지 못하고 있다. 그에 반해 SHPGS의 경우에는 모델 데이터양이 약 2배 증가했음에도 불구하고 FPS 성능 하락이 SVCS의 하락폭에 비해 크지 않았다. 그 이유는 여러 가지가 있겠지만 가장 중요한 요소는 SHPGS의 경우 전용 미들웨어를 통해 직접 그래픽 하드웨어와 통신하기 때문이다. 하지만 일반적으로 SVCS와 같은 가시화 클러스터 시스템의 경우 네트워크를 통해 복잡한 모델 데이터의 계산결과를 범용 소프트웨어를 통해 송수신한다.

결국 SVCS의 성능은 크게 네트워크 성능 개선과 통신 API 효율성에 달려있다. 즉 SVCS에서 복잡한 B모델의 FPS 성능을 향상시키기 위해서는 앞서의 두 가지 환경을 SHPGS의 그래픽 하드웨어 내부 통신 속도에 준할 만큼의 성능으로 향상 시켜야 한다. 그러므로 SVCS 네트워크 성능 개선을 위해 우선 기존의 100Mbps 네트워크 스위치를 1Gbps로 업그레이드 시킨 후, 같은 조건 하에서 다시 두 종류 크루즈 모델의 FPS 테스트를 실시하였다(Fig. 34)

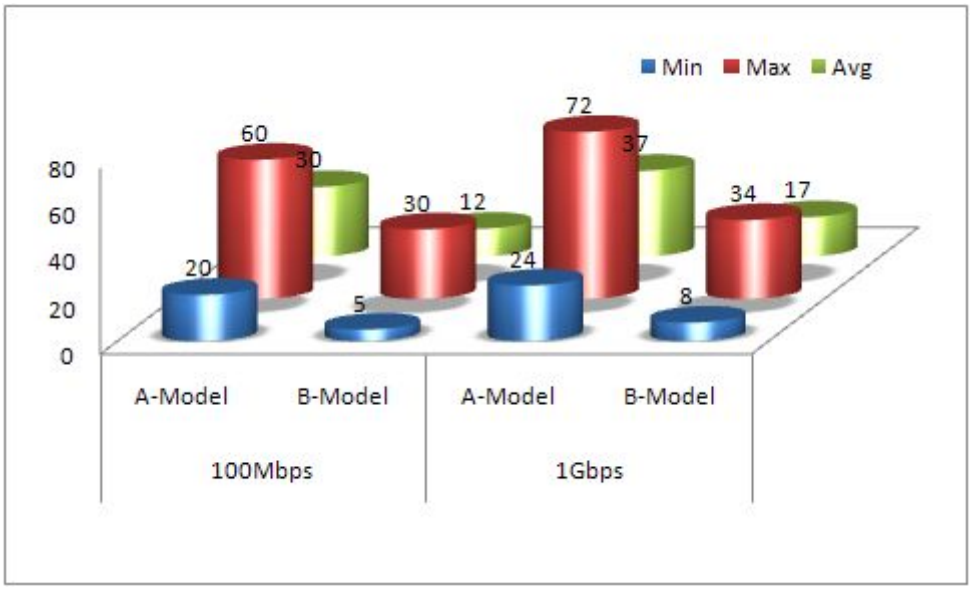


Fig. 34 Frame Rate Chronometry - 100Mbps vs 1Gbps

테스트 결과 전반적으로 SVCS의 FPS 성능이 향상되었다. 하지만 1Gbps 네트워크 환경임에도 불구하고 FPS가 큰 폭으로 상승하지 않았고, 안정적인 가시화 시스템을 보장하는 평균 15 FPS를 약간 넘어서고 있다.

따라서 네트워크 환경 원인뿐만 아니라, 시스템 내부의 메시지 이동 시 통신 API의 호출이 렌더링 루프 내에서 너무 빈번히 일어나 예상보다 저조한 성능향상의 원인을 제공하지 않을까 하는 결론을 도출할 수 있었다. 현재 한 번 렌더링 루프를 돌 때 동기화를 위한 메시지 송수신을 1회씩 실시하고 있으므로 이 빈도를 느슨하게 제어함으로써 시스템 전체 성능을 향상시킬 수 있다.

하지만 시스템 내부의 통신 API 호출 빈도의 제어는 프레임 동기화 성능, 나아가 스테레오스코피 효과에 직접적인 영향을 끼치므로 다음 절의 동기화 성능 테스트와 연계하여 호출 빈도의 최적화를 추구할 필요가 있다.

5-4-4. 동기화 성능 테스트

SVCS에서 스테레오스코피를 구현하기 위해서는 각 슬레이브 노드에서 디스플레이 되는 프레임의 동기화 성능이 매우 중요하다. 즉 어떠한 외부 간섭이나 네트워크 지연과 같은 극한 상황에서도 스테레오스코피 시스템으로의 역할을 제대로 수행하기 위해서는 두 영상의 프레임 동기화가 이루어져야만 한다.

동기화 성능을 테스트하기 위해 삼차원 영역 내에서 약 200,000 Vertices 크루즈 모델의 회전배수 값을 단계적으로 높여가며 FPS를 측정하였다. 여기서의 FPS는 결과적으로 두 노드의 프레임이 서로 동기화된 초당 프레임 수를 뜻하게 된다. 삼차원 영역의 원점을 중심으로 선박을 점점 많이 회전시켜 회전량을 단계적으로 상승시키면 한 프레임마다의 계산 작업량이 증가하게 되고 결국 FPS를 저하시키는 원인을 제공한다. 그러므로 가시적인 프레임 동기화 확인을 통해 다양한 상태에서의 시스템의 동기화 성능을 가늠할 수 있다.

Fig. 35는 회전배수에 따른 크루즈 모델의 FPS 변화를 나타내고 있다.

Fig. 35 Frame Rate Variation by Rotation Time of 3D Object

모든 단계에서의 프레임 동기화 성능을 시각적으로 판단해 보았을 때 두 영상이 정확히 같은 단계에서 투사되어 스테레오스코피 영상을 인지하는데 전혀 지장이 없었다. 즉 동기화 성능 테스트 결과 일반적인 모델 데이터를 처리할 경우 가시적으로 만족할만한 동기화 성능을 보였다. 단지 회전량이 증가함에 따라 시스템의 FPS 또한 단계적으로 감소하였고 특히 32배 이상의 경우 5프레임 이하로 FPS가 떨어져 모델의 회전효과를 부드럽게 느낄 수 없었다. 이 결과를 통해 SVCS가 가시화 작업을 수행함에 있어서 적정량 이상의 계산 작업량에 대해 가시화 시스템으로서의 한계를 가지고 있다는 사실을 알 수 있다.

앞 절에서 완료한 시스템 성능 테스트와 본 절의 프레임 동기화는 SVCS와 같은 가시화 클러스터 시스템에서는 서로 상충하는 기법이므로 이 두 가지 기법의 최적화가 필요하다. 즉 전체적인 시스템의 성능 향상을 도모하게 되면 동기화와 데이터 분산, 취합을 위한 메시지 전달 량과 대역폭이 증가해 프레임을 렌더링 하는데 더 오랜 시간이 필요하게 된다. 또한 동기화 및 데이터 분산, 취합을 위한 메시지 전달 API의 호출 빈도를 인위적으로 낮추어 제어하게 되면 모델에 따라 프레임 동기화 성능, 또는 프레임 영상 효과에 문제가 발생한다. 그러므로 SVCS의 내부적인 계산 작업의 성능향상을 위해서 통신 API의 호출 빈도 제어 혹은 효율적인 병렬 처리를 통한 동기화 성능의 최적화가 향후 연구에 반드시 필요한 부분이다.

5-4-5. 스테레오스코피 및 몰입감 효과

통합 GUI 시스템을 수행한 후, 다양한 종류의 모델을 가시화하여 스테레오스코피 효과를 검증하였다. 만약 연속되는 프레임들이 부드러운 영상으로 인식될 수 있다면, 즉 시스템의 성능이 보장된다면 SVCS에서는 SHPGS의 프로그램과 같은 스테레오스코피 알고리즘을 사용하기 때문에 어떤 플랫폼에서든지 입체효과를 보장될 수 있다. 즉 고성능 그래픽 하드웨어 시스템 환경에서의 입체효과와 차이가 없다. 나아가 SVCS 응용 GUI 소프트웨어를 통해 손쉽게 스테레오 요소를 변경할 수 있기 때문에 개인의 특성을 반영하기가 더 쉬워졌고 결과적으로 보다 나은 입체효과를 체감할 수 있었다.

아래 Fig. 36은 SVCS에서 좌, 우 프레임을 동시에 생성해 테스트 동안 각 디스플레이 영역에 크루즈 모델 객체가 스테레오로 투사되는 장면이다.

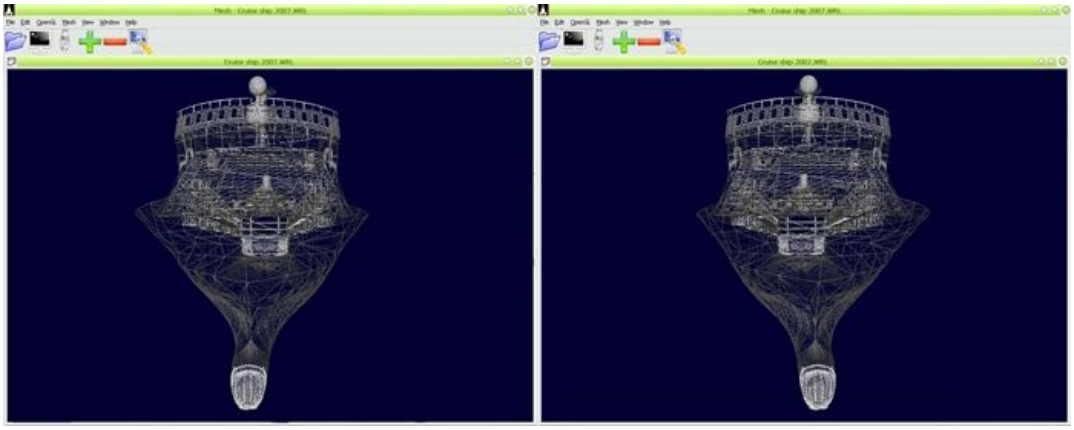


Fig. 36 A Stereoscopic Object on SVCS (Left & Right)

몰입감의 경우, 측정지표는 인간의 개별적인 감각에 달려있기 때문에 그 표준을 정할 수가 없다. 본 논문에서는 일반인 50여명에게 시스템을 경험하는 기회를 주고 개개인의 의견을 수렴하는 것으로 몰입감의 정도를 판단하기로 하였다. 다양한 의견들이 있었지만 대표적인 의견을 간추려 Table. 4에 정리하였다.

Table. 4 An Opinion Poll of Stereoscopic effect on SVCS

1st	입체감이 뛰어나며 CAD 설계에 크게 효율적일 것 같다. (24명)
2nd	입체효과는 느껴지나 전체적으로 어둡고 어지럽다. (18명)
3rd	입체영상과 일반영상과 큰 차이를 못 느끼겠다. (5명)

대부분의 사용자가 입체영상을 효과적으로 느끼고 있다. 어둡고 어지러운 현상은 스테레오스코피의 고질적인 문제점으로, 주변 환경개선과 사용자 맞춤형 GUI 제어를 통해 어느 정도 극복할 수 있을 것이다.

5-4-6. SVCS 성능 테스트 종합

본 시스템은 3-노드 클러스터 기반의 스테레오스코픽 시스템이다. 약 500,000 Vertices 삼차원 모델을 평균 17 FPS로 렌더링하고 스테레오스코피 효과 및 프레임 동기화가 성공적으로 수행됨을 다양한 검증절차를 통해 확인하였다. 또한 SVCS는 비교대상인 고가의 SHPGS와의 객관적인 성능 차이에 비해 비교적 선전하는 모습을 보였다. 즉 SVCS는 일반적인 CAD작업을 수행하는데 무리 없는 경제적인 스테레오스코피 전용 가시화 시스템이다.

하지만 SVCS가 처리할 수 있는 일정 데이터양을 초과하게 되면 그 때의 FPS는 비교대상인 SHPGS에 비해 전반적으로 그 성능 감소폭이 크다. 또한 프레임 동기화 제어를 위한 통신 API의 호출 빈도가 너무 잦아서 성능향상 폭에 한계를 가져온다. 즉 보다 복잡한 작업이 필요할 경우 반드시 현재보다 고성능의 클러스터 가시화 시스템을 구축할 필요가 있다.

본 5장에서 수행되었던 테스트들로부터 제기된 문제점들의 원인은 여러 가지가 있지만 가장 큰 원인은 통신부하, 즉 네트워크 속도의 한계에 있다. 따라서 보다 효율적인 병렬 처리 루틴을 개발하고 시스템의 노드 확장을 통해 고성능 클러스터를 구현하여 이들 간의 시스템 최적화 연구가 요구된다.

6. 결 언

가상현실 기반 가시화는 첨단 기술의 발전에 힘입어 그 수요가 점점 늘어나는 추세이며, 그 활용분야 또한 다양하다. 그리고 다양한 계층의 수요를 충족시킬 수 있는 제반 기술에 대한 요구 및 개발도 꾸준히 증가하고 있다. 하지만 일반연구를 목적으로 할 경우 고가의 그래픽 하드웨어 플랫폼을 구축하기에는 현실적인 어려움이 있고, 해당 기술이 생산업체에 특화되어 있기 때문에 사용자가 자유롭게 시스템을 제어하는데 한계가 있다.

본 논문에서는 클러스터링 기법을 활용해 보다 경제적이고, 사용자 친화적인 SVCS을 구축하는 과정과 방법에 대해 소개하였다. 특히 스테레오 가시화를 위한 요소기술에 대해 고찰하고, 클러스터 시스템에 해당 기술을 특화하여 보다 안정적인 SVCS를 구축하는 방법에 대해 논하였다. 나아가 시스템을 검증할 수 있는 GUI기반의 통합 환경을 구축해 성능을 테스트하여 시스템의 유용성과 문제점을 파악하고 해결 방법을 모색함으로써 향후 연구 방향을 제시하였다.

본 연구를 통해 구축된 SVCS는 스테레오스코피의 몰입감 제어, 클러스터 세부노드의 설정 그리고 디스플레이 확장 등의 사용자 위주의 제어가 가능하다. 따라서 임의의 상황에 맞게 적절히 설정 요소들을 변경함으로써 다양한 가시화 연구에 대해 유연한 시스템을 구현할 수 있다. 향후 응용 어플리케이션 검증을 통해 도출된 시스템의 문제점을 수정하고, 연결되는 하위 노드의 개수를 늘려 시스템을 점진적으로 확장할 계획에 있다. 더불어 각종 첨단 입출력 장치와 디스플레이, 센서들을 시스템에 동기화시켜 가상현실 기반의 조선설계 응용 프로그램을 개발하고, 나아가 SVCS을 활용하여 다양한 제반 연구에 필요한 삼차원 스테레오 가시화 연구를 지원하고자 한다.

본 논문은 삼차원 스테레오스코피 기초 연구, 교육 혹은 관련 응용 소프트웨어를 제작하는 사용자에게 유용하게 활용될 수 있을 것이라 사료된다. 아울러 SVCS가 다른 일반 시뮬레이터와의 결합을 통해, 보다 현실감 있는 다양한 가상현실 시스템을 구현하는데 미력하게나마 도움이 되길 희망한다.

참 고 문 헌

- 신창균 (2006). "클러스터로 도전하는 슈퍼컴퓨터 구축과 활용", 1판, 혜지원.

- 여진욱 (2005). “3D Display Full Review", 산업사원부(enFun) 주최 리뷰 공모전 우수상. http://asnote.net/review/3d/

- 윤재문, 이창민, 윤근항 (2005). “삼차원 컴퓨터 그래픽 가시화 클러스터를 위한 프레임 동기화 기법에 관한 연구”, 선박해양기술, 제40호, pp.141-148.

- 이만용 (1996). "X-window Programming”, KLDP Forum, http://kldp.org/

- 이창민, 윤재문, 이장현, 김원돈 (2004). “선박 설계 지원 DMU 구현을 위한 3D Real-Time Graphics 기법 연구,” 대한조선학회 추계학술대회 논문집, 10월, pp.20-22.

- 최현호 (역) (2005). "OpenGL SuperBible", 제3판, 정보문화사.

- Code::Blocks (2008). "Code::Blocks User Manual", Code::Blocks Official Web Site, http://www.codeblocks.org/

- Ericksson G (2003). "Stereo Rendering using OpenGL and GLUT", Technical Report.

- Han Chen (2007). "Data Distribution Strategies", Master Dissertation, Princeton University, pp 3-9.

- Julian S, Kevin H (2008). "Cross-Platform GUI Programming with wxWidgets", 5th Printing, PRENTICE HALL.

- K-BENCH (2006). "Quadro FX4500 Review", K-BENCH.

- MPI (2008). "MPI를 이용한 병렬 프로그래밍", KISTI 슈퍼컴퓨팅 센터.

- Olson E (2002). "Cluster Juggler - PC cluster virtual reality", Master Dissertation, Iowa State University, pp.41-43.

- wxWidget (2008). "Community of wxWidget", wxWidget Official Web Site, http://wxwidgets.org/

Appendice

Appendix A : Terms

- 가상현실 : 컴퓨터 기술을 통해 인간의 오감을 자극하여 현실과 유사한 또 다른 현실을 창조하는 활동으로써 일정한 틀을 갖고 있는 것이 아니며 모든 상상 가능한 환경 자체가 가상현실의 범위 안에 들어갈 수 있다. 본 논문에서의 가상현실은 연구 및 개발에 수반되는 실질적인 가상현실 시스템 플랫폼을 의미하며 주로 Virtual Reality (VR)라 일컫는다.

- 교착상태 : 실행중인 프로그램에서 여러 개의 프로세스가 서로 상대방의 자원을 요구하면서 무한정 기다리고 있는 상태로써 무엇보다도 교착상태의 발생가능성을 배제하는 것이 가장 중요하다. 본 논문에서는 점대점 통신의 양방향 통신시의 교착 가능성을 제거하기 위해 논블록킹 통신을 사용하였다. 논블록킹 통신이란 통신이 시작되면 완료와 상관없이 리턴하고 이후 완료 여부를 검사하는 방식으로 대기를 통해 교착 가능성을 제거할 수 있다.

- 리눅스 : 1989년 핀란드 헬싱키대학에 재학 중이던 리눅스 토르발스(Linus Torvalds)가 유닉스를 기반으로 개발한 공개용 오퍼레이팅시스템(OS)으로, 1991년 11월 버전 0.02가 일반에 공개되면서 확대 보급되기 시작하였다. 유닉스(Unix)가 중대형 컴퓨터에서 주로 사용되는 것과는 달리, 리눅스는 워크스테이션이나 개인용 컴퓨터에서 주로 활용한다. 리눅스는 소스 코드를 완전 무료로 공개하여 전 세계적으로 약 5백만 명이 넘는 프로그램 개발자 그룹을 형성하게 되었으며, 이들에 의해 단일 운영체제의 독점이 아닌 다수를 위한 공개라는 원칙하에 지속적인 업그레이드가 이루어지고 있다. 파일 구성이나 시스템기능의 일부는 유닉스를 기반으로 하면서, 핵심 커널 부분은 유닉스와 다르게 작성되어 있다. 인터넷 프로토콜인 TCP/IP를 강력하게 지원하는 등 네트워킹에 특히 강점을 지니고 있으며, 유닉스와 거의 유사한 환경을 제공하면서 무료라는 장점 때문에 프로그램 개발자 및 학교 등을 중심으로 급속히 사용이 확대되고 있으며 각종 주변기기에 따라 혹은 사용하는 시스템의 특성에 맞게 소스를 변경할 수 있으므로 다양한 변종(배포판)이 출현하고 있다. 현재까지 Mandrakelinux, SUSE Linux, Fedora Project,

Red Hat, Ubuntu Linux 그리고 한컴리눅스 등 다양한 배포판이 존재한다.

- 마운트 : 저장 장치에 접근할 수 있는 경로를 디렉터리 구조에 편입시키는 작업으로써 마운트 명령어를 사용하게 되면 저장 장치의 접근 경로를 사용자가 원하는 위치에 자유롭게 생성할 수 있다. 마운트를 이용하면 분산 파일 시스템으로 확장하기에 용이한 장점이 있다.

- 미들웨어 : 분산 컴퓨팅 환경에서 서로 다른 기종의 하드웨어나 프로토콜, 통신환경 등을 연결하여, 응용프로그램과 그 프로그램이 운영되는 환경 간에 원만한 통신이 이루어질 수 있게 하는 소프트웨어를 말한다. 본 논문에서의 미들웨어는 좁은 의미의 소프트웨어 드라이버를 지칭하였다.

- 병렬처리 : 다중처리 시스템에서 하나 또는 그 이상의 운영체제가 여러 개의 프로세스를 관리하며 동시에 수행하는 시스템으로 프로세스를 하나 혹은 여러 프로세스에 작업 분산시켜 동시에 수행함으로써 빠른 시간 내에 효율적인 계산을 처리하는 방법이다. 본 논문에서는 병렬처리 방법 중 하나인 메시지 전달식 병렬처리기법을 사용하여 클러스터 시스템의 병렬화를 구현하였다.

- 포팅 : 서로 다른 플랫폼에서 설계된(다른 CPU, OS, 라이브러리) 실행 가능한 프로그램을 특정 플랫폼에서 실행하기 위한 소프트웨어 적응 과정으로 서로 다른 환경에서 사용가능하게 소프트웨어를 특정 플랫폼의 요구에 적절히 변화시키는 것이 핵심이다. 일반적으로 기존의 다른 플랫폼에서 제작된 소프트웨어의 포팅과정을 거치는 것이 새롭게 소프트웨어를 개발하는 것보다 경제적인 관점에서 효율적이다.

- 크로스플랫폼 : 컴퓨터 프로그램, 운영체제, 컴퓨터 언어, 프로그래밍 언어, 소프트웨어 등이 여러 종류의 컴퓨터 플랫폼에서 동작할 수 있다는 것을 뜻한다. 크로스플랫폼 응용 프로그램은 하나 이상의 플랫폼에서 실행할 수 있기 때문에 이러한 종류의 소프트웨어를 멀티 플랫폼 소프트웨어라고도 한다. 표준 플랫폼이 확립되지 않는 이상 이 클로스플랫폼 기반의 소프트웨어는 호

환성을 위해서 꼭 필요하며 현재도 다양한 분야의 소프트웨어들이 크로스플랫폼 기반으로 제작되어 소프트웨어 적용의 범위를 넓히고 있다.

- COTS : Commercial off-the-shelf (COTS)는 이미 만들어져, 일반인들에게 판매, 임차 또는 허가가 가능한 소프트웨어 제품이나 하드웨어제품을 일컫는 말이다. COTS는 본 논문에서는 하드웨어 제품을 의미하며 이러한 상용 기성품을 통하여 시스템의 전반적인 개발비용과 개발시간을 단축시킬 수 있다.

- DNS(Domain Name Server) : TCP/IP 네트워크에서 사용되는 네임 서비스의 구조이다. TCP/IP 네트워크에서는 도메인이라고 하는 논리적 그룹을 계층적으로 설정할 수 있고, 그 논리적 그룹 명칭인 도메인명을 컴퓨터의 명칭(호스트명)의 일부에 포함시켜 이용하는 방법을 찾고 있다. 도메인 혹은 호스트 이름을 숫자로 된 IP 주소로 해석해 주는 TCP/IP 네트워크 서비스로서, 계층적 이름 구조를 갖는 분산형 데이터베이스로 구성되고 클라이언트·서버 모델을 사용한다.

- MPP : 초병렬처리란 클러스터에서의 필요한 데이터만 네트워크를 통해 교환하면서 각 노드는 자신이 가지고 있는 데이터를 참조해서 실행하는 별개의 프로세스들의 협동과정이다. MPP에서 효율적으로 수행되는 프로그램들은 방대한 데이터를 처리할 때 프로그램을 여러 독립적인 부분으로 나누어 실행시킬 수 있는 것들이다. 예를 들어 데이터마이닝 같은 경우 정적인 데이터베이스에 대하여 여러 가지 독립적인 검색을 수행한다. 또 체스 게임 같은 인공지능에서는 여러 가지 다른 수를 분석하여 볼 필요가 있다. 많은 경우 MPP 시스템은 프로세서의 클러스터로 구성된다. 각 클러스터는 SMP처럼 작동하며, 메시지 패싱은 클러스터 사이에서만 일어난다. 인자를 어드레싱 하는 것이 메시지를 통해서나 메모리 어드레스를 통해 가능하기 때문에 어떤 MPP 시스템은 NUMA 시스템이라 부른다.

- OpenGL Stereo API : OpenGL 스테레오 API는 다양한 GUI Toolkit 환경에서 스테레오스코피 효과를 구현하기 위한 플래그를 지원한다. MFC의 경우

SetWindowPixelFormat 클래스의 PIXELFORMATDESCRIPTOR 구조체 내부 PDF_STEREO와 PFD_STEREO_DONTCARE이 스테레오를 지원하는 플래그이다. wxWidget의 경우 OpenGL 그래픽을 디스플레이하기 위한 GLCanvas 클래스의WX_GL_STEREO가 스테레오를 지원하는 OpenGL 스테레오 API이다. 이 API를 이용해 쿼드 버퍼를 지원하는 하드웨어에 한해서 스테레오스코피 효과를 구현할 수 있지만 본 논문에서는 이 특화기술을 클러스터링 기법을 이용해 소프트웨어적으로 구현하였다.

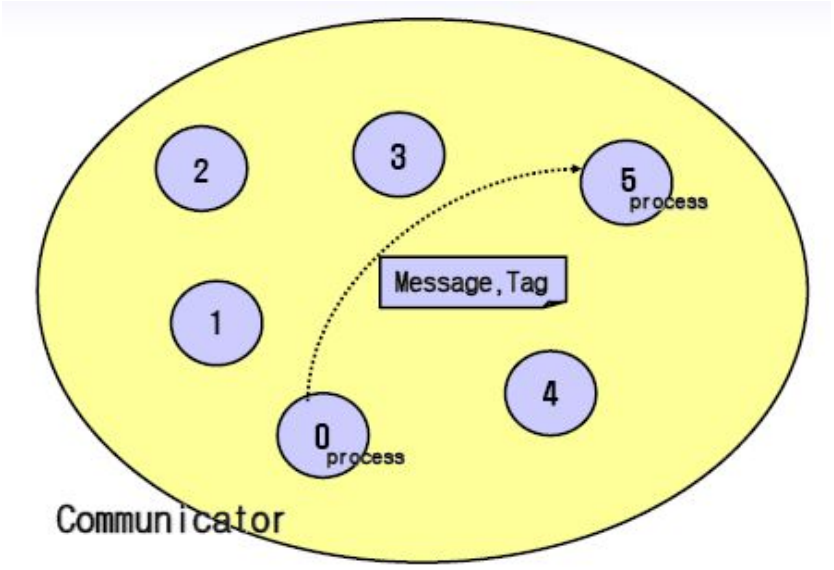
- SIMD : Single Instruction, Multiple Data의 약자로 모든 노드에서 같은 프로그램이 실행되지만 각 노드마다 다른 데이터를 처리하는 방법으로 Flynn의 분류에 의거하여 성립된다. MPI-1이 이 방법을 기본모델로 하고 있으며 모든 노드가 똑같은 프로그램을 실행하므로 프로그래밍 자체는 비교적 간단해지지만 모든 프로세스가 어느 순간에도 같은 동작을 하므로 프로세스 수가 많아지게 되면 효율이 떨어지는 단점이 있다. 또한 노드마다 다른 작업을 처리하는 복잡한 프로그램을 만들 때 SIMD 모델은 프로그램 자신이 모든 루틴을 내장하고 있어야 하므로 프로그램 크기도 커지고 복잡해질 수 있다. 이에 반해 MIMD (Multiple Instruction, Multiple Data)는 노드마다 다른 데이터를 프로세스마다 다른 프로그램이 서로 도우면서 실행하는 방법으로 여러 사람이 동시에 다른 종속 프로그램을 만들어 그것으로 큰 병렬 프로그램을 쉽게 만들 수 있고 마스터의 규칙에 맞는다면 새로운 기능의 종속 프로그램을 쉽게 확장할 수 있다. 즉 MIMD의 경우 큰 규모의 프로그래밍에 보다 적합하고 확장성이 뛰어난 방법이다.

- X-Window : X 윈도우[엑스 윈도우] 시스템은 리눅스를 비롯해 대부분의 유닉스에 채용되어 있는 혁신적이면서 네트워크 투명성을 보장하는 그래픽 환경 기반의 시스템 소프트웨어이다. 1984년 미국의 MIT에서 공개해 현재 모습으로 발전했고, 최근 리눅스가 각광을 받으면서 보다 보편적인 환경으로 다가서는 상황이다. X 윈도우는 윈도우 시스템에 있어 사실상의 표준이라 할 수 있으며, 오픈그룹에 의해 관리되고 있다. 비록 현재 많은 사용자들은 윈도우95나 윈도우NT가 윈도우 시스템의 전부인 것처럼 인식하고 있지만, 그

것은 어디까지나 특정 시스템에 설치되고 운영될 수 있도록 마이크로소프트가 만든 것으로, 많은 X 윈도우 제품 중 하나에 지나지 않는다고 해야 할 것이다. 1987년 가을에 현재의 모습과 유사한 X 윈도우인 X11이 만들어졌다. X윈도우의 참조 모델은 Palo Alto 연구소의 Parc와 Star로, 이는 단순히 윈도우, 아이콘, 메뉴 및 포인터 등의 인터페이스를 화면에 나타낸 것뿐만 아니라 컴퓨터 역사의 한 획을 긋는 혁명적인 시도였다. X11이 공개된 후 수많은 워크스테이션 업체에서 X 윈도우를 표준 윈도우 시스템으로 채택하기 시작했고, 기술개발과 배포를 위해 여러 업체가 모여 X 컨소시엄을 결성, 관리하게 되었다. 그 후 X 컨소시엄이 오픈그룹 산하단체로 바뀌었지만 근본적인 변화는 없으며 비교적 자유로운 라이선스로 인해 무료 또는 저렴한 가격으로 누구나 사용할 수 있다. 이렇듯 X 컨소시엄의 규격에 부합한다면 누구든지 X 윈도우를 개발할 수 있기 때문에 다양한 상용 또는 공개 X 서버가 발표된 상태이다. 리눅스에 있어 표준으로 사용되는 것은 XFree86이다. XFree86은 인텔의 x86 프로세서 기반의 환경에서 동작하는 X 서버로, 별도로 제품을 구입할 필요 없이 무료로 사용할 수 있다. X 윈도우의 핵심은 기본 윈도우시스템이고, X 윈도우를 구성하는 각 계층은 기본 윈도우시스템 위에 놓이게 된다. 기본 윈도우시스템은 특정 정책 대신에 메커니즘을 제공하는 역할을 가지고 있는데, 예를 들면 그림을 그린다거나 폰트를 출력하는 등의 메커니즘이 그것이다.

Appendix B : MPICH - Message Passing Interface API

1. MPI의 기본 개념



- MPI는 프로세스 기준으로 작업을 할당하며 일대일, 일대다, 다대일, 다대다 통신을 지원한다.

- 메시지 : MPI에서 메시지는 송수신하는 데이터를 담는 봉투로 정의할 수 있다(데이터 + 봉투). 여기서 데이터는 버퍼, 데이터 개수, 데이터 타입의 정보가 필요하며 봉투는 송수신자 랭크, 태그, 커뮤니케이터 정보가 필요하다. 즉 어떤 프로세스가 어디에 있는 어떠한 데이터를 얼마나 보내는가?, 그리고 어떤 프로세스가 받아 어디에 저장하며 얼마나 받을 준비를 하고 있어야 하는가에 대한 정보를 담고 있는 MPI 통신 규약이다.

- 태그 : 메시지 매칭과 구분에 이용하는 것으로 태그번호에 의해 순서대로 메시지 도착을 처리하게 하는 기능이다. 송수신 데이터를 나타내는 고유한 정수로 표시할 수 있으며 그 범위는 0 ~ 1073741823(2^{30}-1)이다.

- 커뮤니케이터 : 서로 간에 통신이 허용되는 프로세스들의 집합으로 모든 MPI 통신 루틴에는 커뮤니케이터 인수가 포함된다. 물론 다중 커뮤니케이터를 생성하는 것도 가능하며 같은 커뮤니케이터를 공유하는 프로세스들끼리만 통신이 가능하다. MPI_COMM_WORLD는 프로그램 실행시 정해진, 사용 가능한 모든 프로세스를 포함하는 기본 커뮤니케이터이다.

- 랭크 : 동일한 커뮤니케이터 내의 프로세스들을 식별하기 위한 식별자로 메시지의 송수신자를 구별하기 위해 사용된다. 프로세스가 n개 있으면 0부터 n-1까지 번호를 할당하며 MPI_Comm_rank 루틴에 의해 자동적으로 부여된다.

2. 점대점 통신과 집합통신

- 점대점 통신 : 두 개 프로세스 사이의 통신으로 하나의 송신 프로세스에 하나의 수신 프로세스가 대응되는 방식이다. 통신은 커뮤니케이터 내에서만 이루어지며 송수신 프로세스의 확인을 위해 커뮤니케이터와 랭크를 사용한다. 점대점 통신을 사용할 경우 통신의 완료는 메시지 전송에 이용된 메모리 위치에 안전하게 접근할 수 있음을 의미한다.

- 집합통신 : 동시에 여 개의 프로세스가 통신에 참여하여 일대다, 다대일, 다대다 대응이 가능하다. 즉 여러 번의 점대점 통신사용을 하나의 집합통신으로 대체할 수 있어 보다 오류 가능성을 줄이고 최적화 되어 일반적으로 빠른 장점이 있다. 집합통신 루틴은 커뮤니케이터 내의 모든 프로세스에서 호출되며 동기화가 보장되지 않는다는 특징이 있다. 즉 점대점 통신과 달리 논블록킹 루틴과 태그가 없기 때문에 프로세스 교착에 주의해야 한다. 집합통신의 종류로는 방송(Broadcast), 취합(Gather), 환산(Reduce), 확산(Scatter), 장벽(Barrier) 등이 있다.

3. 블록킹통신과 논블록킹 통신

통신 모드	MPI 호출 루틴	
	블록킹	논블록킹
동기 송신	MPI_SSEND	MPI_ISSEND
준비 송신	MPI_RSEND	MPI_IRSEND
버퍼 송신	MPI_BSEND	MPI_IBSEND
표준 송신	MPI_SEND	MPI_ISEND
수 신	MPI_RECV	MPI_IRECV

: 점대점 통신의 경우 통신모드에 따라 프로세스 진행을 제어하기 위해 블록킹 통신과 논블록킹 통신 루틴이 구분되어 있다. 일반적으로 최적화되어 있는 표준 송신 모드를 사용한다.

- 블록킹 : 통신이 완료된 후에만 루틴으로부터 리턴 되므로 데이터 통신의 신뢰성이 높은 방식이다. 하지만 프로세스 교착 가능성이 커지므로 성공적인 통신을 위해서는 송수신자 랭크를 명확히 하고, 동일한 커뮤니케이터 내에서 메시지 태그가 일치하는 데이터를 서로 송수신해야 할 필요가 있다. 또한 표준송신 모드가 아닐 경우 수신 버퍼의 크기도 임의로 제어해주어야 한다.

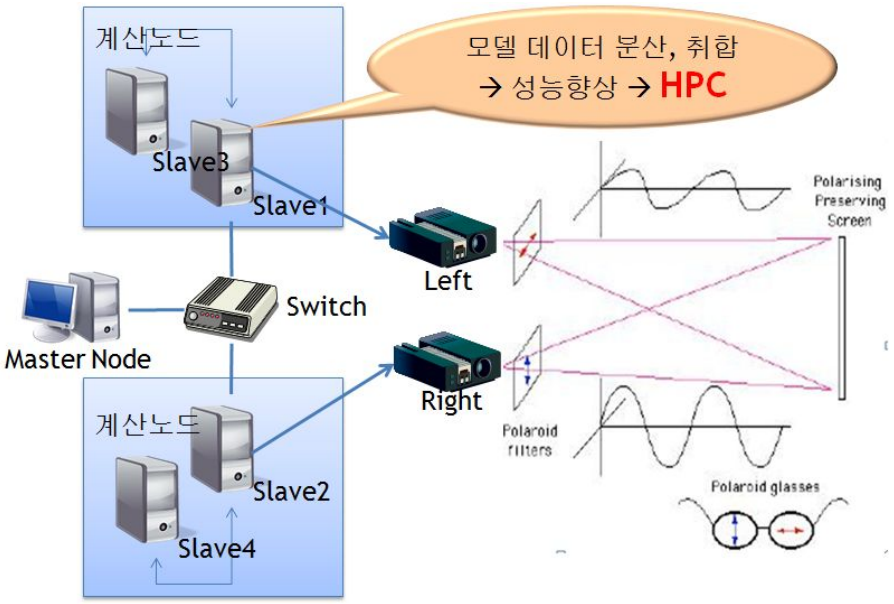
- 논블록킹 : 통신을 세 가지 상태로 분류하는 방식이다. 즉 논블록킹 통신의 초기화 상태로부터 송수신 포스팅을 실시하고, 전송 데이터를 사용하지 않는 동안 다른 작업 수행하여 통신과 계산 작업을 동시 수행하며, 통신 완료시 대기 또는 검사하는 상태로 분류할 수 있다. 이 논블록킹 통신 루틴에 의해 프로세스의 교착 가능성과 통신 부하를 감소시킬 수 있는 특징이 있다. 특히 통신 완료 후 대기 또는 검사를 통해 결과적으로 블록킹 통신의 효과를 가져올 수 있기 때문에 통신의 신뢰성을 보장할 수 있다.

Appendix C : High Performance Cluster
(데이터 분산 / 취합 과정을 통한 클러스터 시스템의 성능 향상)

연구 여건상 HPC(High Performance Cluster)에 대해서는 실질적인 구현과 검증이 이루어지지 못했다. SVCS의 성능향상은 차후 연구와 연계되는 부분이지만 HPC를 구성하기 위한 방법과 개념에 대해 본 Appendix C에서 간단히 설명하여 이해를 돕고자 한다.

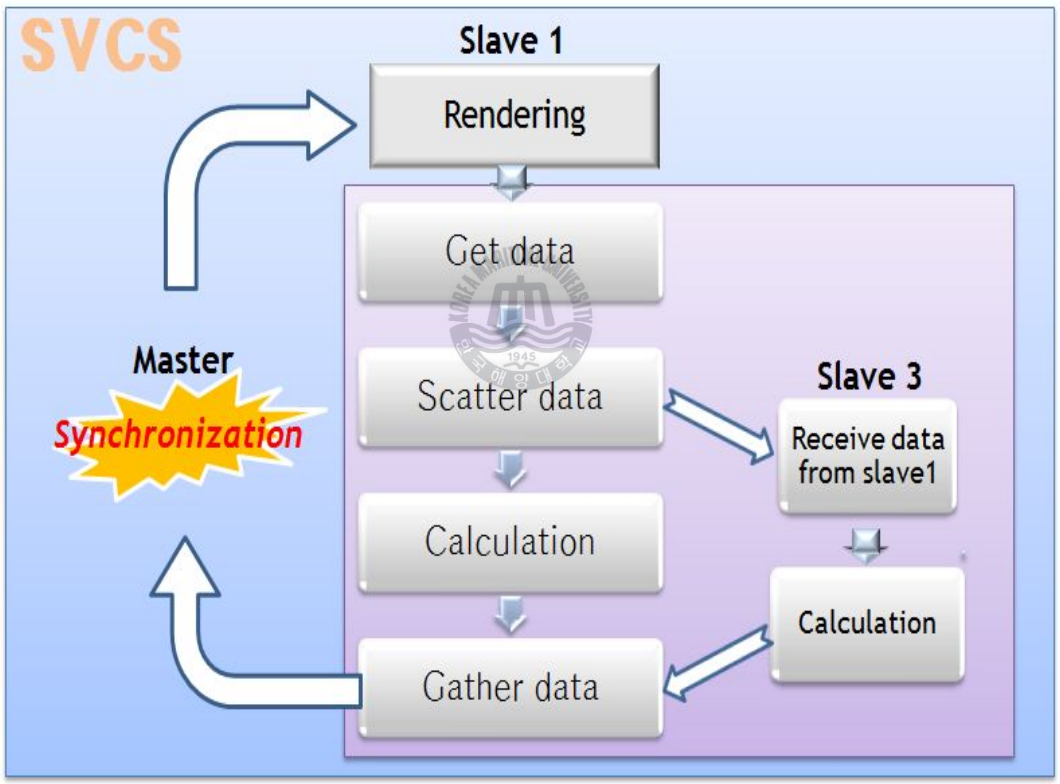
본문의 결론부에서 거론했던 소프트웨어 차원의 병렬처리 기법인 통신 API의 호출 빈도를 최적화하는 방안도 성능향상의 한 방법이 될 수 있겠지만 실질적인 SVCS의 성능을 향상시키기 위해서는 계산노드의 추가가 이루어져야 한다.

클러스터는 네트워크를 통해 통신을 하는 특성상, 임의의 노드 확장이 용이한 장점이 있다. 다시 말해 클러스터 시스템에 추가할 노드의 네트워크 장치를 서로 연결하고 서버 측에서 간단한 인증에 관한 설정만 해주면 쉽게 시스템을 확장할 수 있다. 또한 시스템 작업 실행 시 계산할 노드의 숫자만 명시적으로 결정해주면 프로그램을 수정하지 않고도 확장한 노드를 쉽게 활용할 수 있다.



즉 위의 그림처럼 기존의 계산노드 Slave1에 종속되는 Slave3, Slave5...를 추가할 수 있다. 반드시 왼쪽과 오른쪽을 담당하는 노드의 숫자가 같을 필요는 없지만, 만약 서로의 개수가 다르다면 프레임 동기화에 좀 더 많은 네트워크 부하가 걸릴 것이기 때문에 가능하다면 같은 수의 종속노드를 설치해야 한다.

실질적인 병렬 처리를 위해 MPI를 통해 병렬처리 루틴을 구현해야 한다. 즉 데이터의 분산 및 취합을 통해 복잡한 데이터를 각 노드에서 나누어 처리하여 전체 시스템의 계산효율을 증가할 수 있다. 병렬 루틴의 알고리즘 구현과정은 다음과 같다.



복잡한 데이터를 가진 상위 슬레이브 노드는 MPI 점대점 통신 API를 통해 자기 자신 및 하위 계산노드로 균등하게 데이터를 분산시켜 노드의 작업량을 분산시킨다. 이 때 집합통신의 MPI_Scatter() 함수를 통해 데이터가 담긴 배열의 행렬 값을 자신을 포함한 두 개의 노드로 분산시키도록 병렬 알고리즘을 구

성하였다. 그 반대로 행렬 값을 취합하여 연산이 완료된 데이터를 다시 상위 슬레이브 노드로 전달하는 작업이 필요하다. 이 역시 집합통신의 MPI_Gather() 함수를 통해 병렬 알고리즘을 구성하였다. 이 두 과정을 렌더링 루프 내에서 반복시켜 한 노드에 집중되어 있는 데이터 연산 작업을 분산시킴으로써 결과적으로 개별 노드의 부하를 줄여 렌더링에 필요한 프레임 단위의 연산 시간을 절약할 수 있다. 또한 클러스터 시스템의 확장성을 이용해 노드의 수를 점점 늘려준다면 전체 시스템의 성능 향상 효과를 기대할 수 있을 것이다.

실제로 HPC를 구현하기 위해서는 위에서 설명한 하드웨어 정렬과 병렬 루틴 외에도 많은 고려사항이 필요하다. 몇 가지 예를 들자면, 보다 저렴한 가시화 클러스터를 구축하기 위해서는 본 논문에서 기준으로 했던 완전한 컴퓨터들을 노드로 묶어 구축하는 COW형식이 아니라 Diskless 방식을 고수해야 할 것이다. 또한 병렬처리 효율을 위해서 SIMD (Single Instruction Multiple Data, Parallel Processing) 구조가 아닌 MIMD (Multiple Instruction Multiple Data, Parallel Processing) 구조를 기본 병렬 처리 모델로 고려할 수도 있다. 그 외에도 본 연구에서 선택했던 많은 제반사항들을 다시 한 번 점검해서 SVCS의 HPC 구현에 최적화 시키는 작업들이 필요할 것이다.