리눅스 클러스터를 이용한 비행환경 다중모니터 병렬 그래픽 처리

Parallel Multi-Monitor Display of Flight Simulation Graphics Using Linux Cluster

김 병 수* 김 기 영*
Kim, Byoungsoo Kim, Ki-Young

ABSTRACT

In this paper, our research efforts and activities for displaying flight simulation graphics on multi-monitors connected to a Linux cluster is introduced. In this pilot system graphics rendering as well as view calculation including LOD implementation for each monitor is performed on each sub-node computer connected to the monitor rather than using an expensive main server. The Linux cluster is constructed by combining general-purpose desktop computers, and MPI library is used for communication between sub-nodes. It could be concluded from our experience that it is possible to construct a massive multi-monitor display system by adding to the cluster as many sub-node computers and monitors as possible with economic efficiency.

주요기술용어(주제어) : Parallel Processing(병렬 처리), Linux Cluster(리눅스 클러스터), Multi-Monitor Display(다중-모니터 디스플레이)

1. 머리말

컴퓨터의 성능이 급격히 발전함에 따라 과거 고성능의 서버급 컴퓨터에서나 가능하던 데이터 수치계산 및 그래픽 랜더링 프로그램이 일반 개인용 컴퓨터(이하 PC)를 이용하여 수행할 수 있게 되었다. 또한 저가의 PC를 서로 연결하여 슈퍼컴퓨터의 계산속도에 버금가는 클러스터를 저렴하게 구축하여 사용하고 있는 것이 현실이다. 따라서 성능이 좋은 고가의 장비를 이용하는 것 보다는 저가의 PC를 이용하여 최대한의 효과를 보는 것이 경제적으로나 대중화 측면에서 중요하다고 생각된다.

UDP 또는 TCP 통신을 이용하여 다중모니터에서의 비행환경을 구축하는 것이 일반적인 방법이다. 한편, UDP 또는 TCP 통신을 이용한 비행환경 그래픽 처리를 위해서는 메인이 되는 서버에서 그래픽을 제외한 모든 수치 계산을 수행하고 수행한 수치계산 결과를 각 노드에 전달해준다. 따라서 메인 서버가 다른 서브 노드들보다 성능이 뛰어나야 된다는 것을 의미하고 수치 계산량이 증가할수록 비행 반응 속도에 영향을 미치게 된다. 이를 위해 고가의 메인 서버를 필요로 하게 되는 것이다. 이에 반해 본 논문에서는

† 2006년 6월 8일 접수~2006년 7월 20일 게재승인
* 충남대학교(Chungnam National University)
주저자 이메일 : kbskbs@cnu.ac.kr

MPI를 이용하여 각 노드간의 통신이 이루어지고 가시화 랜더링 뿐만 아니라 비행 움직임에 필요한 각종 수치 계산, 지형의 LOD(Level of Detail)등을 각 노드별로 수행할 수 있도록 구현하였다.

본 실험실에서 PC기반의 리눅스 클러스터를 구축하고 클러스터 각 노드의 디스플레이 모니터를 이용하여 구현해온 모델 랜더링 경험을 바탕으로 이 논문에서는 MPI 기반의 리눅스 클러스터 환경[1]에서 다중모니터를 이용하여 비행환경 그래픽 처리[2]를 위한 시스템 환경설정방법을 소개하고 개발된 프로그램의 성능에 대해 설명한다.

2. 하드웨어 시스템 구축

본 연구는 병렬 가시화 랜더링을 구현한다는 면에서는 기존의 WireGL과 비슷한 방식[3]이라고 볼 수 있지만, 추가적으로 프로그램에서는 각 노드에 단순히 랜더링 작업만을 담당시키는 것이 아니라, 수치 계산 및 여타의 병렬화 계산이 각 서브노드에서 수행될 수 있는 구조로 시스템 설정을 하였다.

본 연구를 위해 3대의 동일한 PC를 구입하여 베어울프 방식의 리눅스 병렬 클러스터 환경을 구축하였다.

[표 1] 하드웨어 구성

구 분	내 용
CPU	Pentium4 2.6GHz
RAM	1GByte
Network card	100Mbps
Hub	100Mbps
VGA	GeForce2 MX 400

가. 다중 모니터 이용 방법

리눅스 클러스터에서 다중 모니터를 이용한 병렬 그래픽 처리, 즉 실행파일을 이용하여 각 노드의 모니터에서 가시화 프로그램이 실행될 수 있도록 하는 방법은 다음과 같은 설정을 필요로 한다.

xhost를 이용하여 메인 노드에서 각 서브 노드의 display를 접근할 수 있는 권한을 설정한다. 프로그램 코드에서 각 서브노드에 가시화될 비행환경의 view를 설정하고 실행파일을 생성한 후 display option을 이용하여 가시화 프로그램을 실행한다. 예를 들면, 실행파일의 이름이 simulate이고, 가시화에 사용할 클러스터 노드수가 3개일 때는 다음과 같다. “$mpirun -np 3./simulate -display mainnode.cnu.ac.kr:0.0” 이다. mainnode.cnu.ac.kr은 디스플레이 하고자하는 컴퓨터의 hostname이고 0.0은 host 컴퓨터의 기본 모니터를 의미한다. 즉, mainnode.cnu.ac.kr라는 hostname을 갖는 컴퓨터의 기본 모니터에 프로그램을 실행하라는 것이다. 특별히 다른 설정을 하지 않으면, 위의 방법에 의해 두 개의 서브 노드에서 구현된 그래픽 화면이 메인노드의 모니터인 mainnode.cnu.ac.kr:0.0에 서로 별개의 그래픽 윈도우 창으로 나타나게 된다. 본 연구에서와 같이 각 서브노드에서 구현된 그래픽 화면이 각각의 모니터에 자신의 그래픽 화면이 구현되도록 하기 위해서 프로그램 내부에서 노드별 디스플레이 대상 모니터가 자신의 디스플레이로 설정되도록 하였다.

나. 하드웨어 시스템 구축 시 유의점

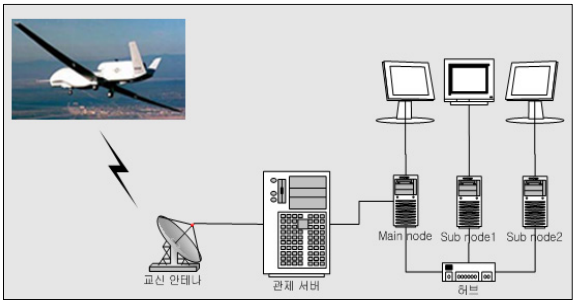
시스템 구축에서 유의해야 될 사항은 각 노드의 VGA가 동일해야 되고, 리눅스 설치 시에 제공되는 VGA driver를 사용하지 않고, 제품 개발사에서 제공하는 전용 드라이버를 사용하여야만 랜더링 속도면에서 최적의 성능을 얻을 수가 있다.

3. 소프트웨어 구현

지금까지 구현된 프로그램은 비행 자세 및 위치 변경을 메인노드에 연결된 키보드와 마우스를 이용하여 수행하는 단계이지만, 추후 비행 시뮬레이션에 필요한 동력학적 변수를 실제 비행중인 비행체에서 직접 받아들여 실시간으로 가시화 할 수 있도록 구현할 계획이고, 이와 관련한 시스템 구조를 그림 1에서 보여주고 있다. 특히 무인기에서 보내오는 데이터를 이용하여 쉽게 가시화할 수 있는 모듈을 개발하고, 또한

[표 2] 소프트웨어 구성

구 분	내 용
OS	RedHat Linux 7.2 (Kernal : 2.4.13)
컴파일러	gcc(Ver 2.96)
3차원 지형지물	디지털 고도데이터(DTED)
그래픽 라이브러리	Opengl 1.2(glut)
메시지 전송방식	MPI



[그림 1] 비행 데이터를 이용한 비행환경 시스템 구조

[그림 2] 비행환경 구현 화면

기존 비행체 상태 자료를 읽어 들여 실시간 비행경로를 다시 살펴볼 수 있도록 할 계획이다. 지형 처리는 실제 지형 고도 데이터(DTED)를 이용하여 지형을 구현하였고 여기에 적절한 환경이 시뮬레이션 될 수 있도록 건물 등 간단한 지형지물을 추가하여 구현하였다.

본 연구에서 개발된 비행환경 병렬 그래픽 처리 시스템으로 구현된 다중 모니터 비행환경 구현의 예를 그림 2에서 보여주고 있다. 그림 2는 위/아래 그림으로 나뉘며, 3화면 모두가 연결되어 있다는 것을 보여주기 위해 view 각도를 약간씩 회전시켜 나타내고 있다.

위의 그림에서 중앙 화면의 건물 그림이 회전에 의해 아래 그림에서는 왼쪽 화면에서 나타나고 있는 것을 볼 수 있다. 모니터의 화면 틀에 의해서 가려지는 모니터와 모니터 사이의 공간은 조종사의 시야를 가리는 비행기 창문틀로 가정하고 이 공간에 의해서 가려지는 외부공간이 화면에서 자연스럽게 배제되도록 프로그래밍하였다. 그림 2에서 구현된 것과 같은 방법으로 더 많은 노드를 추가하여 view의 영역을 넓힌다면 비행환경 뿐만 아니라, 돔 형태의 가상 환경 또한 구축할 수 있다.

4. 기존 전문업체 개발 제품과의 비교

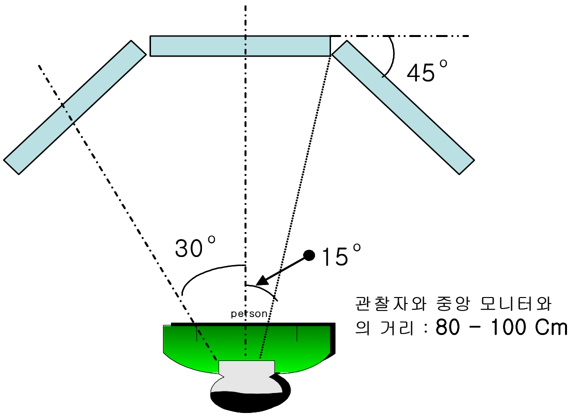
그림 3은 (주) 도담시스템스에서 개발한 F-5 비행 시뮬레이션 장치이다. 모의 조정석에서 비행환경을 가시화 하여 보여주고 있다.

본 연구에서 개발한 시스템과의 비교를 위하여 주요 기능을 살펴보면 다음의 표 3과 같다. 그림 3에 보이는 전문 시스템의 경우 대형화면을 이용하는데 비하여, 본 시스템의 경우 실험실에서 일상적으로 운

[그림 3] F-5 비행 시뮬레이터(도담시스템스)

[표 3] F-5 시뮬레이터 기능

구 성 품	기 능
Image Generator	30Hz 이상 화면 갱신률 제공 남한 전역 영상 데이터 베이스 제공 HUD Symbology Overlay 특수효과 : 시정(가시거리), 안개, 구름, 목표물, 폭발
Display System	3개의 채널을 이용하여 135 x 30 시야 각 제공
특징	실제와 동일한 형상의 조종석 및 스위치 류 제공 LCD를 이용한 그래픽 계기 VFD를 이용한 자료입력 장치



[그림 4] 모니터와 관찰자와의 위치 계산

용되는 모니터(18.1″ LCD 중앙모니터, 19″ CRT 좌우모니터)를 이용하여 구축된 기본적인 시스템으로서 모니터의 크기와 배치된 각도 등을 고려하여 그림 4와 관찰자와 모니터와의 거리 및 각도 등을 고려하여 시야각을 계산하여 프로그램이 구현되었고, 차후 모니터의 크기나 수가 변경이 될 경우 이에 맞추어 프로그램 조정이 될 것이다.

그림 5의 경우는 그림 2의 상태로부터 항공기가 옆으로 롤을 취한 상태를 보여주고 있다. 모니터와 모니터 사이의 공간을 계산하여 환경을 가시화를 했기 때문에 기울어짐이 어색하지 않게 나타난다. 이는 프로그램 내에서 중앙 모니터의 그래픽을 결정하는 Viewing Matrix를 좌/우 모니터 상에 구현될 그래픽 화면에 맞추어 필요한 각도만큼 회전변환하고, 모니터 사이의 가려지는 공간을 고려하여 Viewing Volume을 설정해주었기 때문이다.

디지털 고도데이터(DTED)에 대한 실사 인공 위성사진을 구할 수 없어 텍스쳐 맵핑이 구현되지 않았기 때문에 현실감은 그림 5의 본 시스템이 그림 3의 전문 시스템에 비해 덜하지만, 그리픽 구현 속도면에서는 본 다중 모니터와 리눅스 클러스터를 이용한 시스템을 통하여 전문 시스템에 못지않은 성능을 얻을 수 있었다.

추후 조이스틱, 프로젝터와 같은 몇 가지의 추가적인 하드웨어 장비가 더해진다면, 고가장비에서 구현할 수 있었던 비행 환경을 현실감 있게 구축할 수 있을 것이고, 나아가 무인항공기와의 통신을 통한 실시

[그림 5] 항공기 롤링 상태의 프로그램 실행 화면

[표 4] 구현된 기능

구 성 품	기 능
Image Generator	20~25Hz 화면 갱신률 제공 특수효과 : 시정(가시거리), 안개, 구름, 목표물
Display System	3개의 모니터를 이용하여 90도 시야 각 제공
특 징	LOD를 이용한 그래픽 지형생성

간 비행 환경 구현이나 기록된 비행 데이터와의 연동을 통하여 항공기 설계 및 운용 과정에 요긴하게 활용 될 수 있을 것이다.

5. 맺음말

본 연구에서는 대용량의 지형지물 데이터를 효과적으로 처리하고 실시간으로 가시화할 수 있는 리눅스 클러스터를 이용한 다중 모니터 그래픽처리 구현을 위한 환경설정을 소개하고 이를 비행환경에 적용하였다. 지금까지의 연구로서는 리눅스 클러스터를 이용한 비행환경 그래픽처리가 싱글 클라이언트에서 그래픽처리와 비교했을 때 가시화 구현속도가 확연히 개선되었다고 할 수는 없다. 이는 개발된 프로그램에서 병렬 처리해야 하는 대상 데이터 량이 충분히 많지 않기 때문으로 조사 되었다. 하지만, 고가의 장비를 이용하지 않고서도 그에 준하는 비행환경과 동일한 가상환경을 구현하는데 일반 PC급을 이용해서도 충분히 가능하다는 것을 확인 할 수 있었다. 나아가, 현재까지 구축된 시스템에 노드 컴퓨터와 모니터를 원하는 개수만큼 추가함으로써 더욱 큰 규모의 다중-모니터 병렬 디스플레이 시스템의 구축이 큰 비용 부담 없이 구축이 가능하다는 결론을 얻었다.

참 고 문 헌

[1] Marc Snir, Steve Otto, Steven Huss-Lederman, David Walker, Jack Dongarra, "MPI : The Complete Reference".
[2] Edward Angel, "Interactive Computer Graphics".
[3] 김창식, 조진연, "대형 유한요소 구조 해석용 MPI 기반 병렬 가시화 알고리즘의 개발과 그 구현", 한국항공우주공학회 춘계 학술 발표회 논문집, pp.164~168, 2003. 4.
[4] Kurt Wall, Mark Watson, Mark Whitis, "Linux Programming".
[5] Foley, Van Dam, Feiner, Hughes, Phillips, "Introduction To Computer Graphics".
[6] Will Schroeder, Ken Martin, Bill Lorensen, "The Visualization Toolkit".