타일형 디스플레이 장치를 이용한 볼륨 데이터 가시화

허영주*
*한국과학기술정보연구원
e-mail : popea@kisti.re.kr

A Volume Data Visualization Method Using Tiled-Display

Hur,YoungJu
Korea Institute of Science and Technology Information

요 약

볼륨 렌더링은 스칼라 데이터로 구성된 3 차원 볼륨 데이터를 가시화하는 기법을 가리키며, 유체 역학, 지진, 기상, 해안, 천문, 의료 등 다양한 분야에서 데이터를 분석하는데 널리 사용된다. 최근에는 대용량 볼륨 데이터가 생성되면서 고해상도 디스플레이에 대한 요구가 높아졌으며, 이에 따라 타일형 디스플레이 장치에서 볼륨 데이터를 가시화하려는 시도가 많이 이뤄지고 있다.

본 논문에서는 타일형 디스플레이 장치에서 볼륨 데이터를 가시화하는 기법을 구현했다. 볼륨 데이터 렌더링은 타일형 디스플레이 장치와 연결된 PC-클러스터에서 그래픽스 하드웨어를 사용하는 볼륨 렌더링 기법으로 수행했으며, 이렇게 렌더링된 결과 이미지를 컴포지팅함으로써 해당 디스플레이 장치에 적절한 이미지를 생성했다.

1. 서론

볼륨 렌더링은 스칼라 데이터로 구성된 3 차원 스칼라 볼륨 데이터를 가시화하는 기법을 가리키며, 샘플링, 물질 분류(classification), 쉐이딩(shading), 컴포지팅(compositing)의 단계를 거쳐서 처리된다. 이런 볼륨 렌더링 기법은 유체역학, 지진, 기상, 해안, 천문, 의료 등 다양한 분야에서 널리 사용되고 있으며, 수요가 많은 만큼 다양한 분야에서 많은 연구가 이뤄지고 있다.

볼륨 렌더링 기법 중 가장 널리 쓰이는 방법은 ray-casting 인데, 이 방식은 영상 평면의 각 픽셀에서 볼륨 데이터로 광선을 쏘아서 볼륨 데이터의 각 복셀과 만나는 점의 값을 누적시켜서 이미지를 생성하는 방식이다[1]. 그러나, 이 방식은 속도가 느리다는 단점이 있다. 이런 단점을 해소하기 위해 개발된 방식이 GPU 를 이용한 볼륨 렌더링 기법이다. GPU 를 이용한 볼륨 렌더링 기법은 ray-casting 과정이 텍스쳐 맵핑과 비슷하다는 생각에 착안해서 볼륨 렌더링을 GPU 에서 수행하게 한 방식이며, 샘플링과 컴포지팅을 비교적 빠른 시간에 처리할 수 있게 해준다. 그러나, 이 방식은 그래픽스 하드웨어의 텍스쳐 메모리 용량에 의해 많은 제약을 받고 있으므로 최근 늘어나고 있는 대용량 데이터 가시화에는 부적절하다.

GPU 볼륨 렌더링의 이런 제약 사항을 극복할 수 있는 방법은 여러 대의 PC 를 사용해서 병렬 렌더링을 수행하는 것이다. 최근에는 일반 PC 를 사용한 PC 클러스터도 많이 보급되고 있으며, 고해상도 디스플레이 장치를 이용한 데이터 분석에 대한 요구도 늘어나고 있는 실정이다. 따라서, 대용량 데이터를 PC 클러스터를 사용, 고해상도 디스플레이 장치에서 실시간으로 가시화하는 기법에 대한 필요성이 대두되고 있다.

본 논문에서는 6 개의 렌더링 노드를 갖춘 PC 클러스터에서 그래픽스 하드웨어 볼륨 렌더링 기법으로 렌더링한 이미지를 고해상도 디스플레이 장치인 타일형 디스플레이 장치에서 가시화하는 시스템의 구현에 대해 논하겠다.

2. 관련 연구

최근에는 PC 클러스터의 보급과 함께 대용량 데이터의 렌더링에 대한 필요성이 증가되고 있으며, 이런 이유로 병렬 렌더링 기법에 대한 연구가 많이 이뤄지고 있다. 병렬 렌더링 기법은 각 노드로 데이터의 분배가 이뤄지는 시점에 따라 sort-first, sort-middle, sort-last 기법으로 분류[9]할 수 있다. Sort-first 기법은 그래픽스 파이프라인의 초기 단계인 geometry 단계에서 데이터를 분배하는 방식인데, 화면을 여러 영역으로 나누고 각 렌더러가 화면의 영역을 하나씩 맡아서 렌더링하는 식으로 렌더링을 수행한다. Sort-middle 방식은 데이터를 geometry 단계와 rasterization 단계 사이에서 재분배하며, 그래픽스 하드웨어의 특성상 SGI 의 특정 하드웨어 외에는 적용하기 힘들다는 단점이 있다. Sort-last 방식은 rasterization 단계 이후에 데이터를 분배하는 방식이며, 데이터를 무작위로 나눠서 화면 영역상의 위치와는 관계 없이 픽셀값을 계산한 다음, 컴포지팅 작업을 통해 최종 영상을 생성한다. 본 논문에서는 sort-last 방식을 사용했는데, 이 방식에 대한 연구는 주로 시스템간의 통신량을 최소화하고 컴포지션 작업의 로드를 분산하는 작업에 초점이 맞춰져 있다.

Ma[10]는 기존의 binary composition 방식을 개선해서 binary-swap 방식을 고안했는데, 이 방식은 기존의 binary 방식에 비해 컴포지션에 걸리는 시간을 단축시킬 수 있는 알고리즘이다. Stompel[18]은 컴포지션 작업을 최소화하면서 프로세서 간에 균등하게 배분하기 위해 SLIC 이라는 방식을 고안했다. 이 방식은 컴포지션을 스캔라인 단위로 수행하며, 인터리빙(interleaving)을 사용해서 스캔라인을 프로세서에 균등하게 분배하게 된다. 인터리빙은 프로세서간 로드 밸런싱에는 적합하지만, 자칫 연산이 복잡해질 수 있기 때문에 사용에 매우 주의가 요구된다. 인터리빙은 스캔라인뿐만 아니라 불륨 데이터에도 적용할 수 있으며, Garcia[4]는 인터리빙 방식을 이미지 공간과 오브젝트 공간에 모두 적용한 알고리즘을 고안했다.

최근에는 컴포지션에 걸리는 시간을 줄이기 위해 하드웨어로 컴포지션을 수행하는 방식도 많이 연구되고 있다. Muraki[15][16]는 하드웨어 컴포지터를 구현함으로써 컴포지션에 걸리는 시간을 소프트웨어 컴포지터에 비해 획기적으로 감소시킬 수 있었으며, 앞으로는 하드웨어 컴포지터의 사용이 점차 확대될 전망으로 보인다.

최근에는 병렬 렌더링에 가격이 저렴하면서 확장성이 좋은 PC 클러스터를 많이 사용하는 추세이며, PC 클러스터의 사용 범위는 하루가 다르게 확대되고 있다. Humphreys[6][7]는 PC 클러스터로 병렬 렌더링을 수행, 그 결과를 타일형 디스플레이 장치에 디스플레이 할 수 있는 시스템을 개발했다. Chromium 으로 불리는 이 시스템은 sort-first 방식을 사용해서 해당 디스플레이 장치에 병렬 렌더링 결과로 생성된 이미지를 보여준다.

그림 1. VISC 와 타일형 디스플레이

3. 하드웨어 및 소프트웨어 사양

본 시스템의 구현에 사용된 클러스터 시스템은 한국 과학기술정보연구원(KISTI)에서 보유하고 있는 가시화 전용 PC 클러스터 시스템(VISC)으로, 그 사양은 <표 1>과 같다.

VISC 는 전체 7 대의 DELL PC 로 구성되는데, 그 중 6 대는 렌더링 노드로 사용되고 1 대는 마스터 노드로 사용된다. 또, 타일형 디스플레이에 적합한 디스플레이 디바이스도 갖추고 있다.

시스템 구현에는 주로 C 를 사용했는데, 하드웨어 렌더링을 위한 그래픽스 하드웨어 프로그래밍에는 Cg 를 사용했고, 프로세스간 통신에 필요한 병렬 코드는 MPI 로 작성했다.

		마스터 노드	렌더링 노드
CPU	Model	Intel Xeon	Intel Xeon
	Clock	3.06 GHz	3.06 GHz
	CPU	Dual (2 개)	Dual (2 개)
Video card		GeForceFX 5900	QuadroFX 3000G
Main memory		4GB	4GB
HDD	Local	80GB(OS)	80GB(OS)
		160GB(Home)	NFS 로 공유
	RAID	800GB	NFS 로 공유
OS		Debian Linux	Debian Linux
Network		Gigabit ethernet	Gigabit ethernet

표 1. KISTI VISC 사양

4. 타일형 디스플레이 장치를 이용한 볼륨 데이터 가시화

본 논문에서는 6 개의 렌더링 노드를 갖춘 PC-클러스터를 이용, 그래픽스 하드웨어 볼륨 렌더링 기법을 사용해서 렌더링한 결과를 타일형 디스플레이 장치에 보여주는 시스템을 구현했다.

본 논문에서는 512 x 512 x 1252 크기의 Visible Human 볼륨 데이터를 사용해서 실험을 수행했으며, 이 데이터를 256x512x256 크기로 분할해서 6 개의 노드에서 나눠서 렌더링했다. 또, 타일형 디스플레이 장치의 해상도는 가로 1280x3, 세로 1024x2 로 설정했다.

본 논문에서 구현한 시스템의 전체 구성은 <그림 2>와 같다. 가장 먼저 수행하는 작업은 볼륨 데이터를 분할하는 사전 작업으로, 이 사전 작업으로 분할된 부분 볼륨 데이터는 각 렌더링 노드에 위치하게 된다. 클러스터의 렌더링 노드는 부분 볼륨 데이터를 읽어서 GPU 를 이용한 텍스쳐 렌더링을 수행하게 되는데, 이 때, 렌더링 작업은은 디스플레이 장치의 해당 위치에 대해 한번씩 수행되며, 전체 디스플레이 장치의 개수만큼 반복해서 수행된다. 이렇게 렌더링된 결과는 해당 이미지를 디스플레이할 노드로 전송돼서 컴포지션 과정을 거치게 되며, 컴포지션이 수행된 최종 결과 이미지는 타일형 디스플레이의 해당 위치에 디스플레이된다. 각 노드는 컴포지션이 일어나기 전까지는 다른 노드와 정보 교환 없이 독립적으로 렌더링 작업을 수행하게 된다. 이제, 각각의 과정에 대해 좀더 자세히 알아보자.

4.1 사전 작업(Preprocessing)

그래픽스 하드웨어를 이용한 볼륨 렌더링 기법에서 대용량 데이터를 가시화할 때 가장 문제시되는 것은 텍스쳐 메모리의 용량이다. 현재 그래픽스 하드웨어의 텍스쳐 메모리 용량은 그다지 크지 않으므로, 이 용량을 넘어서는 크기의 대용량 데이터를 가시화하려면 데이터를 적절하게 분할하는 과정이 필수적이며, 이 작업은 렌더링을 하기 전에 사전 작업으로 수행해야 한다. 본 실험에서는 512x512x1252 크기의 Visible Human 데이터를 사용했는데, 실험을 위해 해당 데이터를 256x512x256 크기의 부분 볼륨 데이터로 분할했다.

4.2 렌더링 및 컴포지팅

렌더링 노드에서는 부분 볼륨 데이터를 그래픽스 하드웨어로 렌더링한다. 본 논문에서는 3D 텍스쳐를 사용한 볼륨 렌더링 기법[1]을 사용했다. 텍스쳐 맵핑을 이용해서 볼륨 샘플링을 수행하기 위해서는 볼륨 데이터를 텍스쳐로 변환하고, 이렇게 변환된 텍스쳐를 프록시 도형(proxy geometry)에 부착해야 한다. 텍스쳐를 프록시 도형에 부착하는 데는 텍스쳐 좌표가 사용된다.

타일형 디스플레이에서 이미지를 보여주려면 각 노드에서 디스플레이하는 위치에 해당되는 이미지를 매번 렌더링해서 보내줘야 한다. 각 노드에서는 자신이 렌더링하는 부분 볼륨 데이터에서 각각의 타일에 해당하는 부분을 렌더링하는데, 이 때 렌더링 위치는 텍스쳐 좌표로 조절하며, 렌더링 작업은 타일의 개수만큼 반복 수행된다.

해당 위치에 대한 렌더링 작업이 끝나서 해당 타일과 연결된 노드로 이미지를 전송하고 나면, 각 노드에서는 전송된 이미지를 컴포지션해서 디스플레이하는 작업을 수행한다. 컴포지팅은 back-to-front 순서로 이뤄지며, OpenGL 의 glDrawPixels 함수를 사용해서 블렌딩을 수행한다. 블렌딩은 결합 법칙이 성립하므로, 각 이미지를 다시 블렌딩하더라도 최종적으로 생성된 결과 이미지는 전체 데이터를 렌더링해서 생성한 이미지와 동일하다.

사용자 입력은 마스터 노드에서 키보드 입력을 받을 수 있게 구성돼 있으며, 사용자 입력은 MPI 를 통해 렌더링 노드로 전달한다. 이 시스템은 사용자 입력

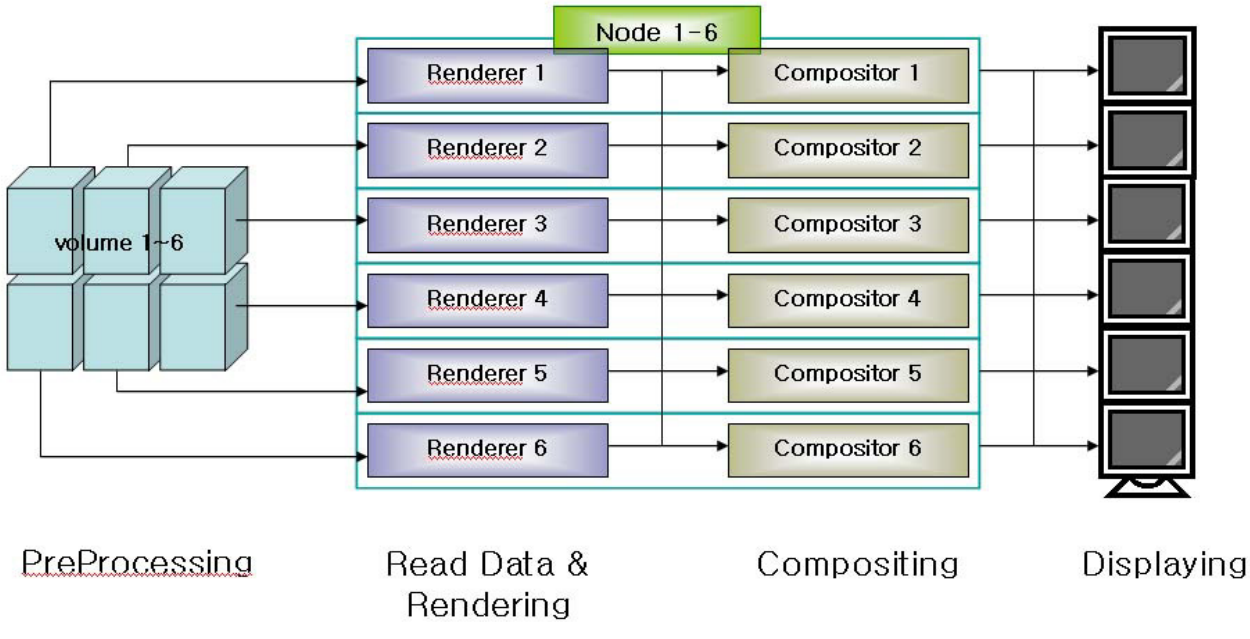

그림 2. 시스템 구조

을 통해 볼륨 데이터를 회전시키면서 렌더링된 이미지를 관찰할 수 있게 구성돼 있다.

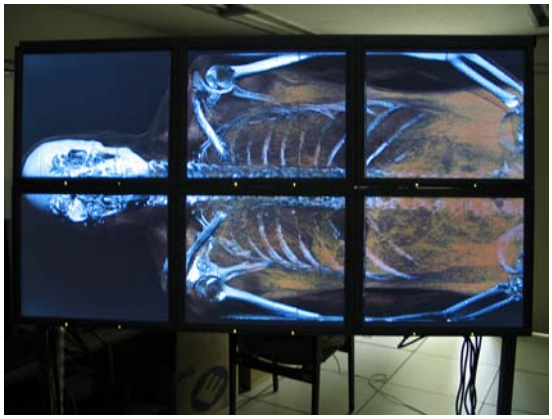

그림 3. 타일형 디스플레이 장치에서 본 VH

5. 결과 및 결론

실험에 사용된 데이터는 Visible Human 데이터이며, 디스플레이 장치의 해상도는 1280 x 1024 로, 타일형 디스플레이는 이 디스플레이 장치를 가로 3 개, 세로 2 개씩 배치해서 구성했다. 한 프레임을 렌더링하는데 각 노드에서 렌더링하는 횟수는 디스플레이 노드의 개수와 동일하다. 따라서, 성능을 최적화하기 위해서는 렌더링 횟수를 최대한 줄여야 하므로, 향후 이 부분에 대한 성능 향상을 계획하고 있다.

각 노드에서 렌더링된 이미지는 glReadPixels 함수를 사용해서 읽어 들여서 해당 부분의 디스플레이를 담당하는 노드로 전송된다. 이미지 렌더링이 끝난 각 노드는 전송된 이미지를 컴포지팅해서 최종 이미지를 디스플레이한다. 타일형 디스플레이 장치에 나타난 최종 이미지는 <그림 3>과 같다.

병렬 렌더링은 단일 프로세서로는 렌더링할 수 없는 대용량 데이터를 렌더링할 수 있게 해주며, 타일형 디스플레이 장치는 고해상도의 이미지 디스플레이를 가능하게 해준다. 향후에는 KISTI 에 새로 도입된 디스플레이 장치에 적합하게 알고리즘을 최적화함으로써 실시간 렌더링을 구현할 계획이다.

참고문헌

[1] 이 중연, "그래픽스 하드웨어를 이용한 볼륨 렌더링", KISTI 기술문서, 2004

[2] James Ahrens, Charles Law, Will Schroeder, Ken Martin, Machael Papka, "A Parallel Approach for Efficiently Visualizing Extremely Large, Time Varying Datasets", LANL Technical report, 2000

[3] James Ahrens, Kristi Brislawn, Ken Martin, Berk Geveci, C.Charles Law, Machael Papka, "Large-Scale Data Visualization Using Parallel Data Streaming", Proceedings of IEEE Computer Graphics & Application Special Issue on Large Data Visualization, 2001

[4] Antonio Garcia, Han-Wei Shen, "An Interleaved Parallel Volume Renderer with PC-Clusters", Proceedings of the 4th Eurographics Workshop on Parallel Graphics and Visualization, 2002

[5] Greg Humphreys, Ian Buck, Mattew Eldridge, Pat Hanrahan, "Distributed Rendering for Scalable Displays", Proceedings of Super Computing, 2000

[6] Greg Humphreys, Mattew Eldridge, Ian Buck, Gordon Stoll, Mathew Everett, Pat Hanrahan, "WireGL: A Scalable Graphics System for Clusters", Proceedings of SIGGRAPH 2001

[7] Greg Humphreys, Mike Houston, Ren Ng, Randall Frank, Sean Ahern, Peter D. Kirchnet, "Chromium: A Stream-Processing Framework for Interactive Rendering on Clusters", Proceedings of SIGGRAPH 2002

[8] Joe Kniss, Patrick McCormick, Allen McPherson, James Ahrens, Jamie Painter, Aln Keahey, Charles Hansen, "Interactive Texture-based Volume Rendering for Large Datasets", Proceedings of IEEE Computer Graphics and Application, 2001

[9] C.Charles Law, Amy Henderson, James Ahrens, "An Application Architecture for Large Data Visualization: A Case Study", Proceedings of IEEE symposium on Parallel and Large-Data Visualization and Graphics, 2001

[10] Kwan-Liu Ma, James S. Painter, Charles D. Hansen, "Parallel volume rendering using Binary-Swap composition", Proceedings of IEEE Computer Graphics and Applications, 1994

[11] Marcelo Magallon, Matthias Hopf, Thomas Ertl, "Parallel Volume Rendering Using PC Graphics Hardware", Proceedings of 9th Pacific Conference on Computer Graphics and Applications, 2001

[12] Steven Molnar, Michael Cox, David El Worth, Henry Fuchs, "A Sorting Classification of Parallel Rendering", Proceedings of IEEE Computer Graphics and Applications, 1994

[13] Kenneth Moreland, David Thompson, "From Cluster to Wall with VTK", Proceedings of IEEE Symposium on Parallel and Large-Data Visualization and Graphics, 2003

[14] Kenneth Moreland, Brian Wylie, Constantine Pavlakos, "Sort-Last Parallel Rendering for Viewing Extremely Large Datasets on Tiled Displays", Proceedings of IEEE Symposium on Parallel and Large-Data Visualization and Graphics, 2001

[15] Shigeru Muraki, Masato Ogata, Kwan-Liu Ma, Kenji Koshijuka, "Next-Generation Visual Supercomputing using PC Clusters with Volume Graphics Hardware Devices", Proceedings of ACM/IEEE SC 2001 Conference(SC'01), 2001

[16] Shigeru Muraki, Eric B.Lum, Kwan-Liu Ma, Masato Ogawa, Xuezhen Liu, "A PC Cluster System for Simultaneous Interactive Volume Modeling and Visualization", Proceedings of IEEE Symposium on Parallel and Large-Data Visualization and Graphics, 2003

[17] Rudrajit Samanta, Thomas Funkhouser, Kai Li, Jaswinder Pal Singh, "Hybrid Sort-First and Sort-Last Parallel Rendering with a Cluster of PCs", Proceedings of SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware, 2000

[18] Aleksander Stomplel, Kwan-Liu Ma, Eric B.Lum, "SLIC: Scheduled Linear Image Compositing for Parallel Volume Rendering", Proceedings of IEEE Symposium on Parallel and Large-Data Visualization and Graphics, 2003

[19] Brian Wylie, Constantine Pavlakos, Vasily Lewis, Ken Moreland, "Scalable Rendering on PC Clusters", Proceedings of IEEE Computer Graphics and Application, 2001