분산 메모리 환경에서의 방대한 볼륨데이터의 압축기반 광선추적법

송동섭 박상훈 임인성
서강대학교 컴퓨터학과 컴퓨터그래픽스연구실

Compression-Based Ray-Casting of Huge Volume Data on Distributed Memory Environments

Dong-sub Song Sanghun Park Insung Ihm
Computer Graphics Lab., Dept. of Computer Science, Sogang Univ.

요 약

기존의 병렬 볼륨 렌더링 방법들은 프로세서간의 발생하는 많은 통신량 때문에 통신 속도가 매우 빠른 병렬 컴퓨터를 이용하였고 통신 속도가 느린 분산 환경에서는 구현이 불가능해 보였다. 또한 가시화하려는 볼륨 데이터도 점점 방대해지고 있는 실정이다. 이에 본 논문에서는 통신 속도에 구애받지 않을뿐더러 매우 큰 볼륨 데이터를 다루는 병렬/분산 볼륨 렌더링을 제안한다. 본 방법은 고비용을 필요로하는 원격 메모리 접근 대신에 압축을 기반으로하여 필요한 데이터를 지역 메모리에서 빠르게 복원함으로써 좋은 성능향상(speedup)을 나타낸다. 이것은 각 프로세서가 전체 볼륨 데이터를 모두 적재하고 있다는 것을 의미한다. 따라서 렌더링 과정 중에 발생하는 프로세서간의 통신을 최소화시킬 수 있었고, 이런 방식은 높은 통신 비용으로 효율적 병렬/분산 처리가 힘든 분산 메모리 병렬 컴퓨터나 PC/워크스테이션 클러스터상에서 매우 적합하다.

1. 서론

과학적 가시화중에서 가장 활발한 연구활동이 이루어지고 있는 볼륨 가시화는 CT, MRI등의 방법을 통해서 직접적으로 얻어진 데이터나 시뮬레이션 등을 통해 간접적으로 얻어진 데이터를 이용하여 과학자들에게 의미있는 정보를 제공하는 기술이다. 볼륨 가시화 방법은 그 방식에 따라 여러 가지로 세분되어지는데 본 논문에서 다루는 방법은 여러개의 프로세서를 이용한 병렬/분산 광선추적법으로서, 고화질과 빠른 렌더링시간을 동시에 만족시킬 수 있다. 기본적으로 병렬/분산 처리를 하는 목적은 계산량을 분산시켜 시간적 이득을 얻고자 함이기 때문에 통신으로 소비되는 시간을 최소로 줄여야 한다. 하지만 광선추적법의 특성상 전체 볼륨데이터가 각 프로세서의 지역메모리로 분산되어 적재되기 때문에 많은 양의 데이터 혹은 영상정보가 프로세서간에 이동되어야 하기때문에 시간을 줄이기 위해서는 프로세서간 통신시간이 매우 빠른 병렬컴퓨터를 사용하는 방법밖에 없었다. 하지만 병렬 컴퓨터를 사용하는 것은 비용이 높고 비실용적이라는 단점이 있기 때문에 일상적으로 이용이 가능한 PC와 워크스테이션들을 클러스터링(clustering)하는 방법이 대안적이다. 물론 PC/워크스테이션 클러스터링은 프로세서간의 통신시간이 느린 것이 단점으로 지적된다. 따라서 본 논문에서는 [3]에서 제안되고 [4, 10]에서 그 성능을 향상시킨 압축방법을 이용하여 전체 볼륨을 각 프로세서의 지역메모리에 모두 적재시킴으로써 프로세서간의 통신이 필요하지 않도록 하는 방법으로 클러스터링의 단점을 자연스럽게 보완할 수 있었다. 각 프로세서는 렌더링에 필요한 데이터를 다른 프로세서의 지역메모리에서 가져오지 않고 자신의 지역메모리에서 빠르게 압축을 복원하므로 빠른 렌더링 속도뿐만 아니라 좋은 성능향상(speedup)이 가능하게 되었다. 통신량이 없으므로 PC/워크스테이션 클러스터에서 뿐만아니라 병렬 컴퓨터상에서도 적용이 가능하다는 특징을 부가적으로 얻을 수 있었다.

2. 관련 연구

본 논문과 가장 관련이 있는 몇가지 병렬 알고리즘을 살펴보기로 하자. [8]에서는 렌더링에 참여하는 프로세서들을 클러스터 단위로 묶어서 각 클러스터에 전체 데이터를 할당하였다. 다시말해 클러스터 수만큼 전체 데이터가 중복되어 있다. 클러스터 내의 각 프로세서들은 균등하게 분배된 데이터의 일부만을 적재하여 물체 분할방식(data partitioning)으로 영상을 생성한다. 전체적으로는 영상을 스트립(strip) 형태로 나누어 하나의 스트립은 하나의 클러스터에게 할당하여 전담하도록 하였다. 클러스터 수가 많아질수록 중복되는 데이터의 양이 많아지므로 렌더링 속도가 빨라진다. 이 방법은 두가지 병렬 알고리즘 - 물체분할, 영상분할 -을 모두 이용한 경우이다. [9]는 영상 분할 방식으로 각 프로세서에 전체 영상의 일부분을 정적으로 할당하다. 이때 할당된 부분 영상은 더 작은 타일로 세분되어 큐(queue)에 넣어둔다. 프로세서는 큐에서 타일을 하나씩 가져와서 렌더링을 수행하는 Task queue image partitioning 방식을 이용한다. 한편 자신에게 할당된 부분 영상을 모두 생성한 프로세서는 이웃 프로세서의 큐에서 아직 처리되지 않은 타일을 가져와서 렌더링을 수행하기 때문에 좋은 부하 균형을 이룬다. [6]에서는 영상 분할 방식의 최대 단점인 데이터의 재분배(redistribution) 문제를 해결하는 방안을 제시하였다. 각 프로세서의 지역 메모리를 데이터 영역과 캐쉬 영역으로 나누고 지역 메모리에 없는 데이터 블록을 원격 접근해 캐쉬에 적재한 후 그 블록을 통과하는 모든 광선을 한꺼번에 처리하였다. 따라서 한번 원격 접근해온 블록은 재접근의 필요가 없어져 통신량을 대폭 줄일 수 있다.

3. 클러스터 환경과 데이터 압축

3.1 클러스터 환경에서의 통신 문제

본 논문이 제안하는 방법에 앞서 분산 환경의 통신 속도를 측정하는 실험을 수행하였다. 우선 512x512x512의 해상도를 갖는 볼륨데이터를 프로세서

개수만큼 임의로 분산시킨다. 데이터를 분산시키는 방법은 부하균형(load balancing)이 가장 우수한 블록(block)을 이용하였다. 블록의 크기는 해상도 16x16x16로 하였고 전체 볼륨은 총 32x32x32개의 블록으로 이루어지게 된다. 프로세서는 하나의 마스터 PE와 다수의 슬레이브 PE로 나뉘며 마스터 PE는 각 블록이 소재한 프로세서의 정보를 담고있는 블록 테이블을 갖고 있다. 마스터 PE는 블록의 인덱스를 차례로 검사하여 해당되는 블록이 슬레이브 PE의 메모리에 적재되어 있는 경우에만 네트워크를 통해 블록을 원격접근하였다. 슬레이브 PE는 기본적으로 대기 상태이며 마스터 PE의 블록 요청에 해당되는 블록을 넘겨준다.

실험은 이더넷상에서 2개의 PE에 의해 이루어졌고 마스터 PE는 MIPS R10000 195MHz CPU를 갖는 워크스테이션, 슬레이브 PE는 Pentium II 450MHz CPU를 갖는 PC에서 NLM의 Visible Human dataset에 대해서 이루어졌다. 또한 시간비교를 위해 마스터 PE에서만 단독으로 같은 크기의 데이터를 렌더링하였고 그 결과는 표 1과 같다.

	2 PE 이용	1 PE 이용
전체 블럭	847.72	364
피부 블럭	282.55	275

표 1) PE수에 따른 렌더링시간 (단위:초)

즉, 위 실험에서와 같이 많은 양의 데이터가 느린 연결 네트워크상에서 이동되면 작업을 분산시키는 것이 이득이 없다는 결론에 이르렀다.

3.2 방대한 볼륨 데이터의 압축

압축방법의 선택에 있어서 압축율 뿐만아니라 복원영상의 화질, 빠른 임의접근, 다해상도 표현, 데이터 응집성의 활용도, 블록단위의 압축등이 고려되었다. 벡터 양자화 (vector quantization)이 몇몇 요건을 만족시키나 이보다 압축률과 복원시간, 복원화질이 더 좋다고 증명된[11] 3차원 웨이블릿(wavelet) 기반의 압축방법[3]에 그 성능을 더 향상시킨 제로비트 인코딩[4][10]을 이용하기로 하였다.

압축률과 복원영상의 화질은 반비례 관계이기 때문에 화질을 최상으로 유지하는 최대한의 압축률을 찾아내기 위해 압축시 포함되는 계수값의 비율을 3%부터 10%까지 변화시키면서 실험한 결과 표 2를 얻어낼 수 있었다[4]. 계수값을 7%이상 사용하여 복원한 영상은 압축을 하지 않고 생성한 영상과 육안으로 구분이 가지 않는다.

		Desired Ratio of the Wavelet Coef's Used			
		3%	5%	7%	10%
Compression	Size (MB)	24.59	35.06	45.43	60.50
Performance	Ratio	29.27	20.54	15.58	11.90
Errors in	SNR(dB)	22.64	25.94	28.57	31.99
Voxel Values	PSNR(dB)	44.49	47.79	50.41	53.84

표 2) 웨이블릿에 기반한 압축 효율 및 오차

또한 복원속도를 측정하기 위해서 압축을 하지 않은 데이터의 접근속도와 비교 실험을 수행하였다. 실험은 임의 복셀과 임의 블록에 대해 각각 100만번 접근하는 속도를 측정하여 그 시간을 비교하는 방식이다. 블록은 64번의 셀로 접근되었고 그 이유는 앞서 수행한 통신시간의 측정에서와 같이 압축스킴의 특성 때문이다. 최소, 최대값을 이용한 범위검사와 중복 복원을 피하기 위해 캐쉬를 사용하였다. 실험결과는 표 3에서 볼 수 있듯이 임의의 복셀에 대한 접근은 복원하는 경우가 0.55초에서 1.07초 가량 늦지만 블록에 대한 접근 속도는 오히려 복원하는 경우가 더 빠른 것으로 나타났다[4]. 이것은 압축단위가 블록단위이고 최소, 최대값의 범위내에 있는 블록만이 복원의 대상이기 때문에 많은 블록의 복원을 피할 수 있었으며 한번 복원한 블록은 재복원의 필요가 없었기 때문이다.

이와같이 지역 메모리에 적재가 불가능했던 방대한 볼륨 데이터를 압축하여 적재하고 렌더링을 하면서 필요한 복셀들은 필요할 때마다 빠른 속도로 복원하므로 개념적으로는 큰 데이터가 모두 적재되어있는 것으로 생각될 수 있다. 이로서 다른 프로세서로의 원격접근이 필요하지 않게되어 병렬성을 극대화 시킬 수 있다.

		Uncompressed	Desired Ratio of the Wavelet Coef's Used			
			3%	5%	7%	10%
Pure Random		2.78	3.33	3.49	3.62	3.85
Cell-Wise	All	18.88	3.88	4.88	5.99	7.52
	Skin	6.50	2.28	2.91	3.53	4.36

표 3) 복셀 복원 속도

4. 클러스터상에서의 분산볼륨 렌더링

제안하는 분산 볼륨 렌더링 방법은 영상 분할법에 기초하며 한 프로세서가 처리하는 작업의 단위는 정사각형의 타일 형태이다. 모든 데이터가 지역 메모리에 적재되는 본 방법의 특성상 데이터의 분할은 없다. 알고리즘은 다음과 같다.

우선 각 프로세서는 제로비트 인코딩으로 압축된 데이터들을 메모리에 적재시킨다. 프로세서마다 성능이 조금씩 다르므로 적재를 빨리 끝낸 프로세서는 아직 적재를 다 끝내지 못한 다른 프로세서가 데이터를 모두 적재할 때까지 기다려서 동기화를 한다. 모든 프로세서에서 데이터 적재가 끝나면 실제 렌더링 작업을 시작한다. 이때 작업에 참여하는 프로세서는 하나의 마스터 PE와 다수의 슬레이브 PE들로 나뉘어 지는데 마스터 PE는 우선 자신을 포함하여 작업에 참여하는 프로세서의 수를 파악한 후 그 프로세서들에게 32x32 해상농의 렌더링할 타일을 할당해 준다(broadcasting). 이때 최종 영상의 해상도에서 해당 타일이 차지하는 위치를 (i,j) 인덱스 형태로 할당해 준다. 타일을 할당받은 슬레이브 PE들은 해당영역을 렌더링 한다. 모든 필요한 데이터는 지역 메모리에 있으므로 필요할 때마다 셀단위로 복원하여 사용한다. 이때 한번 복원된 셀은 캐쉬에 넣으므로 추후에 똑같은 셀을 또 복원해야 하는 비용이 절감될 수 있게 하였다. 또한 블록의 최대값, 최소값을 루트로 하고 셀을 리프로 하는 min-max octree를 이용하므로 렌더링에서 대부분의 시간을 차지하는 광선과 물체의 최초 교차점 계산을 매우 빠른 시간에 수행할 수 있도록 하였다.

```
tile index = (0,0)
termination counter = 0
for(i=0;i<# of PEs;i++) {
      send tile index to PE i
      increment tile index
}
while (true) {
      receive tile index from any slave PE
      if (tile index==(-1,-1) && termination counter==# of PEs)
            break
      else if (tile index == ( 1, 1))
            termination counter++
      else {
            receive rendered tile image
            send tile index to the PE
            increment tile index
      }
}
```

그림 1) 마스터 PE의 알고리즘

```
while(true) {
      receive tile index from master PE
      if (tile index > total # of tiles to render)
            break
      render the tile
            send tile index
            send rendered tile image
}
send tile index as (-1,-1)
```

그림 2) 슬레이브 PE의 알고리즘

할당받은 타일을 모두 렌더링한 슬레이브 PE는 마스터 PE에게 렌더링된 타일(부분 영상)을 넘겨주고 렌더링할 다른 타일을 요구한다. 마스터 PE는 렌

더링된 타일을 넘겨받고 아직 렌더링되지 않은 다음 차례의 타일을 바로 그 프로세서에 넘겨주는 방식으로 계속 반복된다. 이런 동적 할당 방식을 택하므로 프로세서는 쉴새없이 계속 작업하고 따라서 부하균형과 효율이 높아지게 된다. 이 과정을 의사코드로 나타내면 그림 1, 그림 2와 같다.

5. 실험 및 결과

본 논문에서 제안한 알고리즘은 이기종의 PC와 워크스테이션에서 구현되었다. 앞서 실험한 결과에서 증명된 느린 이더넷상에서 구현되었으며 프로세서들을 연결시켜줄 네트워크 라이브러리는 PVM(Parallel Virual Machine)[2]을 사용하였다. MPI(Message Passing Interface)도 일반적으로 많이 사용되나 현재 많이 보급되어 있는 PC에는 포팅되어있지 않아 PC에서도 사용할 수 있는 PVM을 선택하였다. 실험에 사용된 데이터는 많은 볼륨 렌더링 방법에서 표준적으로 사용하고 있는 NLM의 Visible Human dataset을 사용하였으며 밀도값의 범위는 $0 \sim 2^{12}-1$ 이다. 해상도는 512x512x512와 512x512x1024의 두가지에 대하여 실험하였으며 각각 256MB, 512MB로 일반적인 메모리의 크기를 넘어서는 방대한 크기이다. 영상의 크기는 각각 512x512, 512x1024이다.

프로세서 하나만을 이용하여 렌더링 시간을 측정하였는데 이때 렌더링되는 부분이 전체 영상에서 차지하는 비율에 따라 렌더링 시간이 달라지므로 정면에서만 렌더링하지않고 20° 씩 시계방향으로 돌아가며 18번의 렌더링을 수행하였다. 그 결과는 그림 3과 같다.

다음으로 렌더링의 참여하는 프로세서의 수를 증가시켰다. 증가시킬때마다 18장의 영상을 만드는 시간의 평균을 구했고 그 결과는 그림 4와 같다. 8개의 프로세서를 이용해서 각 볼륨을 3.98, 7.46초에 렌더링한다. 그림 5와 그림 6은 각각 성능향상(speedup)과 부하균형을 보여준다. 그림에서 보듯이 성능향상은 8개의 프로세서에서 각 해상도에 대해 각각 6.99, 7.10으로 나타났으며 이것은 각각 87%, 88%의 높은 효율을 나타낸다. 이것은 확장성이 좋고 통신에서 낭비되는 시간이 적음을 뜻하며 느린 이더넷으로도 병렬 컴퓨터의 빠른 통신망의 효과를 본 것이다. 병렬 컴퓨터 상에서의 알고리즘인 [1, 5, 6, 7]과 비교해 보더라도 우수한 결과이다.

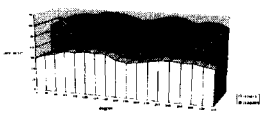

그림 3) 각도에 따른 렌더링 시간

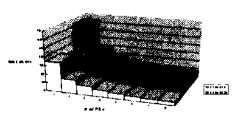

그림 4) 평균 렌더링 시간

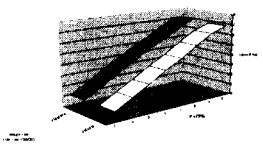

그림 5) 성능 향상

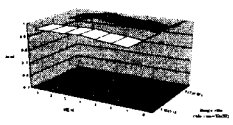

그림 6) 부하 균형

6. 결론

볼륨 렌더링은 작은 볼륨데이터를 하나의 프로세서에서 가시화하는 것에서부터 출발해서 현재는 방대한 볼륨데이터를 여러대의 프로세서를 사용한다고해서 항상 선형 성능향상이 가능한 것은 아니다. 그것은 볼륨 데이터가 각 프로세서의 볼륨에 분산되어 있기 때문에 어떤 방식으로든 통신이 불가피하기 때문이다. 따라서 이제까지 제안된 많은 볼륨 렌더링들이 프로세서간의 통신 속도가 빠른 병렬 컴퓨터를 이용하였다. 하지만 고가의 병렬컴퓨터는 비용면에서 현실적이지 못한 해결방법이다. 이에 본 논문에서는 일반적으로 이용이 가능한 PC/워크스테이션 클러스터를 이용하는 분산 볼륨 렌더링을 제안하였다.

PC와 워크스테이션의 클러스터를 이용하는 것은 실용적인 방법이 될 수는 있으나 이들을 연결시켜주는 느린 네트워크의 속도는 많은 통신량이 필요한 일반적인 병렬 볼륨 렌더링의 구현이 불가능하다. 따라서 본 논문에서는 웨이블렛을 이용한 압축에 기반하여 방대한 볼륨을 모든 프로세서의 지역메모리에 적재하는 방식으로 다른 프로세서와의 통신이 필요하지 않아 80%이상의 고효율과 그에따른 좋은 성능향상 및 빠른 렌더링 시간이 가능하다는 것을 실험적으로 증명하였고 SGDVNT를 구현하여 그 실용성을 보였다. 앞으로는 압축을 하여도 지역 메모리에 적재될 수 없는 더 방대한 볼륨을 빠르게 가시화하는 방법이 연구되어야 할 것이다.

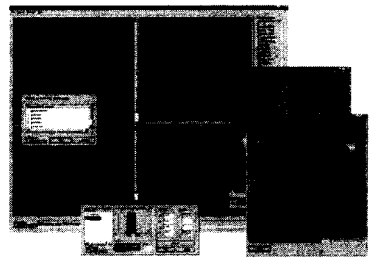

그림 7) SGDVNT에서 렌더링을 하는 모습

참고문헌

[1] M. Amin, A. Grama and V. Singh, "Fast volume rendering using and efficient scalable parallel formulation of the shear-warp algorithm", Proceedings of the 1995 Parallel Rendering Symposium, pp. 7-14, Atlanta, October 1995.
[2] A. Geist et. al., "PVM 3 User's Guide and Reference Manual", Oak Ridge National Laboratory, 1994.
[3] I. Ihm and S. Park "Wavelet-based 3D compression scheme for very large volume data", Proceedings of Graphics Interface '98, pp. 107-116, Vancouver, Canada, June 1998.
[4] I. Ihm and S. Park "Wavelet-based 3D compression scheme for interactive visualization of very large volume data", Computer Graphics Forum, 1999. (In print.)
[5] P. Lacroute, "Real-time volume rendering on shared memory multiprocessors using the shear-warp factorization", Proceedings of the 1995 Parallel Rendering Symposium, pp. 15-22, Atlanta, October 1995.
[6] A. Law and R. Yagel, "Multi-frame thrashless ray casting with advancing ray-front", Proceedings of Graphics Interface '96, pp. 70-77, Toronto, Canada, May 1996.
[7] P. Li, S. Whitman, R. Mendoza, and J. Tsiao, "ParVox - a parallel splatting volume rendering system for distributed visualization", Proceedings of the 1997 Symposium on Parallel Rendering, pp. 7-14, Phoenix, October 1997.
[8] C. Montani, R. Perego, and R. Scopigno, "Parallel rendering of volumetric dataset on distributed memory architectures", Concurrency: Practice and Experience, Vol. 5 Num. 2, pp. 153-167, 1993.
[9] J. Nieh and M. Levoy, "Volume rendering on scalable shared-memory MIMD architectures", Proceedings of 1992 Workshop on Volume Visualization, pp. 17-24, 1992.
[10] S. Park, G. Koo, and I. Ihm, "Wavelet-based 3D compression schemes for the visible human dataset and their applications", CD-ROM Proceedings of the 2nd Visible Human Project Conference, Bethesda, October 1998.
[11] S. Yi, S. park, and I. Ihm, "Performance Enhancement of Light Field Rendering Using a New Compression Technique", KISS Conference (Spring), pp. 653-655, Apr. 1999.