연속 영상의 효과적 병렬 렌더링을 위한 작업분할 기법

최영운° 이윤석

한국외국어대학교 전자정보공학부
{unier°, rheeys}@hufs.ac.kr

A Task Decomposition Scheme for Parallel Rendering of Continuous Images

Youngwoon Choi° and Yunseok Rhee
School of Electronics & Info. Eng., Hankuk Univ. of Foreign Studies

요 약

고화질 입체 영상의 효과적인 재생을 위해 PC 클러스터를 활용한 여러 형태의 병렬화 기법이 제안되었지만, 영상을 구성하는 객체의 분포가 균일하지 않은 경우 충분한 성능을 발휘하지 못하였다. 본 연구에서는 Maya 렌더러를 채택한 PC 클러스터 기반의 병렬 렌더링 시스템을 구축하고, 병렬화 성능을 높이기 위한 효과적인 부하 균형 기법을 개발하였다. 특히 애니메이션을 구성하는 연속 프레임 작업에서 프레임 간의 연관성(coherence)이 높다는 사실에 근거하여, 임의 프레임의 각 분할 영역에 소요된 계산량을 바탕으로 다음 프레임의 부하 분포를 예측하고 이에 맞게 각 프로세서의 작업 영역을 재조정하는 기법을 제안하였다.

1. 서론

3차원 렌더링은 고화질 의료 영상, 게임, 애니메이션, 과학 데이터 가시화 (visualization) 등의 분야에서 널리 활용되는 핵심 기술이다. 특히 수백만 화소 이상의 고화질 영상을 실시간으로 생성해야 하는 어플리케이션에서는 막대한 계산량으로 인해 SGI Infinite Reality 엔진 [1]과 같이 고가의 렌더링 장비들이 사용되고 있다. 그러나 고성능 PC 클러스터가 새로운 병렬 컴퓨팅 시스템으로 등장하면서, 이를 활용한 3차원 렌더링에 대한 연구도 본격화되고 있다 [2, 3, 4].

고성능 강결합 다중 프로세서와 비교할 때, PC 클러스터는 공유 메모리에 대한 접근 속도가 느리고, 프로세서 간의 통신 지연시간이나 네트웍 대역폭이 좋지 않아 병렬화 방법을 고안하는데 제약이 따른다. 결국 이전의 클러스터 기반 병렬 렌더링 시스템들은 대부분 한 프레임 내 (intra-frame)의 영역을 분할하기 보다는, 여러 프레임을 각각 나누어 맡아 작업하는 프레임 사이 (inter-frame)의 병렬화를 구현한 정도였다 [7, 8]. 그러나, 고화질 프레임의 실시간 재생을 위해서는 한 프레임에 대한 고성능 병렬처리가 필수적이며, 병렬화 효과를 제고하기 위해서는 프로세서들의 작업량 분배가 가능한 균일하게 이뤄져야 한다.

한편, 동작이나 배경의 연속성을 갖는 3차원 애니메이션의 경우에는 각 프레임 간의 상호 연관성(frame coherence)으로 인해 계산량 분포를 예측할 수 있다. 즉, 임의의 프레임은 인접한 선후 프레임들과 높은 유사성(similarity)을 갖는다. 따라서 앞선 프레임의 작업 결과로 얻어진 각 분할 영역의 계산량을 바탕으로, 후속 프레임의 계산량을 추정하면 프로세서 간 계산량의 편차를 상당히 줄일 수 있게 된다. 이 사실을 바탕으로, 본 연구에서는 PC 클러스터를 기반으로한 병렬 렌더링 시스템을 구축함에 있어, 3차원 렌더링 작업을 효과적으로 수행하도록 프레임의 연관성과 계산량 정보를 활용하여 부하를 추정하고, 작업 부하를 균형있게 배분하는 기법을 설계하고자 한다.

프레임 연관성(frame coherence)은 인접 프레임 간의 화소 데이터들의 연속성, 즉 변화량이 매우 적은 경우에 높은 연관성이 있음을 뜻한다. 즉 그림 1에 보이는 것처럼, 각 프레임에 나타나는 객체들의 연관성과 시간적 연속성이며, 동일 객체가 거의 같은 지점에 위치하므로 나타나는 현상이다. 프레임 연관성은 Sutherland 등이 렌더링에 적용할 수 있는 여러 형태의 연관성들을 소개하면서 처음 언급하였으나 [5], 연관성을 활용하여 병렬화를 적용하려는 시도는 없었다.

본 논문의 2절에서는 본 연구에서 제안하는 시스템 구성 방법과 작업 분배 알고리즘을 설명할 것이다. 3절에서는 실제 애니메이션 렌더링 결과를 통해 성능을 보이고, 마지막으로 결론을 맺는다.

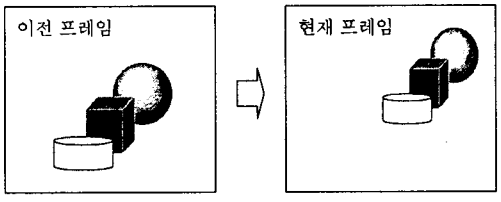

그림 1 프레임 간 연관성을 보이는 예

2. 병렬 렌더링 클러스터의 설계 및 구현

2.1 시스템 구성

본 연구에서는 먼저 그림 2에 보이는 것처럼, 리눅스

를 탑재한 PC 클러스터에 AliaseWavefront 사의 Maya 4.5(Mental ray 렌더러 포함)를 설치하여 렌더링 팜(farm)을 구축하고, 3차원 그래픽스를 이용한 애니메이션 프레임을 생성하는 서비스를 구현하였다. 클러스터는 전체 8개의 노드로 구성했고, 각 노드는 펜티엄 III 800MHz, 512MB 메모리, nVIDIA GeForce4 Ti4200-8X 칩셋을 채용한 그래픽 카드로 구성했다. 노드들은 100MB/s Fast Ethernet으로 연결되며, 하나의 Front-end 서버가 작업 요청을 접수하고, 계산량에 따라 작업 영역을 Back-end 서버들에게 배정하는 기능을 수행한다.

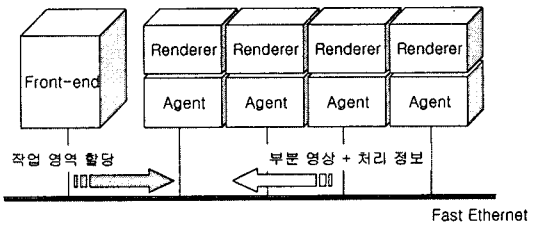

그림 2 병렬 클러스터 기반 렌더링 시스템의 구성

2.2 프레임 연관성을 활용한 작업 영역 분할

본 연구에서는 초기에 서버 노드의 수에 맞게 화면 영역을 분할하는 sort-first 기법[6]을 기본적으로 채택하였다. 이 때 분할된 영역을 타일(tile)이라 부르고, 타일의 수를 노드 수에 맞춘 것은 더 이상의 세분화가 부하 균형의 이점 보다는 불필요한 프로세서 간 통신과 계산 정보의 지역성을 떨어뜨리기 때문이다.

먼저, 4개의 노드로 구성된 클러스터 시스템에서 연속 프레임 영상을 생성하는 과정을 예로 들어 보자. 초기에는 계산량 분포에 따른 정보가 없다고 가정하면, 그림 3(a)에 보이는 바와 같이 최초 프레임에 대해, 참여하는 프로세서의 수만큼의 동일 규격의 A, B, C, D 4개의 타일들로 화면 영역을 분할한다. 그리고, 각 영역을 해당 프로세서를 통해 렌더링한 결과, 각각 w_1, w_2, w_3, w_4의 계산량 비율을 보였고, $w_1+w_2 > w_3 + w_4$, $w_1+w_3 > w_2+w_4$, $w_1 > w_2$, $w_3 < w_4$ ($0 \leq w_i \leq 1$, $\sum w_i = 1$)라고 가정하자. 즉, A 영역에 객체 분포가 집중된 경우이다.

처음에는 당연히 A, B, C, D가 동일한 크기를 갖지만, 분할 영역의 재조정 작업이 계속되다 보면, 상위 A, B의 분할 지점과 하위 C, D의 분할 지점이 달라질 수 있다. 따라서 일반화를 위해, 상위의 분할지점은 n, 하위의 분할지점은 p로 각각 표기한다.

먼저 수평 분할을 통해 상하의 계산량을 균등하게 나누려면 기존의 수평 분할선을 Δm 만큼 위로 이동시켜야 한다. 만일 하나의 분할 영역 내에서는 부하가 고르게 분포한다고 가정한다면, 분할선은 계산량이 큰 A, B를 Δm 만큼 줄이도록 이동해야 하고, 이동 후에 $(w_1+w_2)-(w_1+w_2) * \Delta m / m = (w_3+w_4)+(w_1+w_2) * \Delta m / m$ 이어야 한다. 정리하면, $\Delta m = (w_1+w_2-w_3-w_4) * m / 2$ 이 되며, 새로 조정된 수평 분할점까지의 거리 $m' = m - \Delta m$ 이 된다. 이는 하위 영역 C, D의 계산량 w_3+w_4가 큰 경우에도 대칭적으로 적용할 수 있으므로 설명을 생략한다.

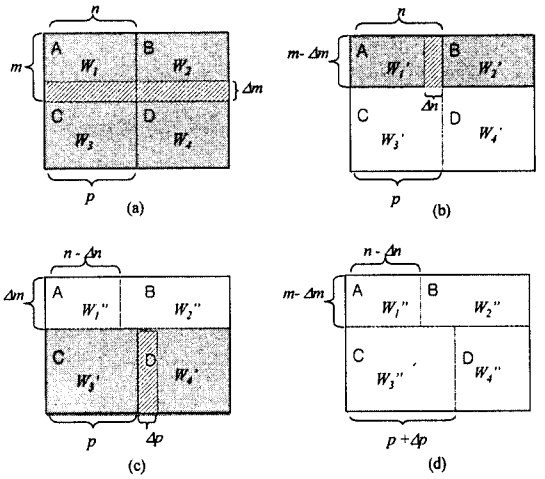

그림 3 이전 프레임의 계산량에 따른 분할 영역 재조정 과정

그림 3(b)는 수평 분할점이 조정한 후의 상태를 보이는데, m'으로 조정된 후의 분할 영역 A', B', C', D' 은, 각각 아래와 같은 w_1', w_2', w_3', w_4' 의 계산량을 갖는다고 추정할 수 있다.

$$w_1' = w_1 - \Delta m * n$$
$$w_2' = w_2 - \Delta m * (h - n)$$
$$w_3' = w_3 + \Delta m * p$$
$$w_4' = w_4 + \Delta m * (h - p)$$

다음으로는 그림 3(b)의 조정된 상위, 하위의 각 분할 영역들에 대해 수직 분할선을 조정하는 과정을 수행한다. 상위 두 영역의 추정 계산량이 $w_1' > w_2'$일 때, 분할선은 w_1'을 Δn 만큼 줄이도록 조정되어야 하며, $w_1'-\Delta n = w_2' + \Delta n$이어야 하므로, $\Delta n = (w_1'-w_2')* n / 2$ 로 조정되어야 한다. 대소 관계가 반대인 경우에도 대칭적으로 적용되므로 설명은 생략한다. 하위 영역 역시 w_3', w_4'의 대소 관계에 따라 똑같은 과정으로 분할선 조정이 필요하다.

이상은 노드의 수가 4인 경우에 관한 예를 보였지만, 일반적으로 2^N 개의 노드에 맞춰 영역을 분할하는 경우, 수평, 수직 방향으로 전체 N 번의 분할을 차례로 수행하게 되며, 위의 과정을 순환적으로 반복함으로 영역 조정을 완료할 수 있다.

3. 실험 결과

실제 애니메이션 렌더링에서의 효과를 알기 위해, 표 1에 보이는 2개의 애니메이션 데이터를 사용하였다. Cell-Phone은 핸드폰 모델의 플립이 열고 닫히는 동작과 함께 회전을 보여주는 애니메이션으로 전체 화면에서 계산량이 일부 영역에 편중된 모습을 보였다. Lux-Ball은 공과 스탠드의 두 물체가 화면 전체에서 활발히 움직이

표 2 애니메이션 실험 결과

분할	데이터	frame time (sec)	imbalance (sec)	efficiency
정적	Cell-Phone	38.2	11.6	0.24
	Lux-Ball	94.1	18.2	0.31
제안	Cell-Phone	22.7	2.3	0.40
	Lux-Ball	58.5	7.1	0.49

표 3 실험에 사용된 애니메이션 특징

애니매이션	해상도	프레임 수	구성 객체수
Cell-Phone	640x400	45	524
Lux-Ball	640x400	60	327

는 애니메이션으로 프레임 내에서는 계산량이 편중되어 있으나 전체적으로는 작업량이 비교적 널리 분포된 형태를 보인다.

정적인 균등 분할과 제안된 기법을 적용하여 비교 실험을 실시하였으며, 표 2에 보이는 것처럼 평균 프레임 시간 (frame time), 부하 불균형 (imbalance), 효율성 (efficiency)를 측정하였다. 부하 불균형은 각 프레임이 최종적으로 완료되기까지 각 서버가 마지막으로 작업을 완료한 서버를 기다려야 했던 평균 시간을 전체 프레임에 대해 합한 것이다. 이상적인 프레임 시간은 병렬화에 따른 오버헤드를 제외하고 이상적인 속도 향상을 통해 얻을 수 있는 시간으로, 단일 서버에서 걸리는 시간을 전체 서버의 수로 나눈 값이다. 효율성은 실제 소요되었던 평균 프레임 시간에 대한 이상적인 프레임 시간의 비율이다.

표 2의 실험 결과를 살펴보면, 전반적으로 제안한 기법이 정적 분할 기법에 비해 우수함을 보인다. 제안된 기법은 정적 분할 기법보다 Cell-Phone과 Lux-Ball 각각에 대해 약 41%, 38%의 성능 향상을 보였다. 특히 부하 편중으로 인한 지연 시간이 크게 줄고, 부하의 균형적인 분배에 따른 속도 향상도 좋은 것으로 보인다. 한편, Lux-Ball의 경우에는 전체 시간대에 걸쳐 작업량이 비교적 넓게 분포한 탓에 정적 분할도 그리 나쁘지 않은 결과를 나타냈다. 그러나, 여전히 프레임 내에서의 편중이 심해 평균 프레임 시간이 길고, 이로 인해 imbalance도 커지게 된다.

시스템의 효율성 역시 제안된 기법이 약 50%에 가까운 우수한 결과를 보이는데, 효율성을 저하시키는 요인으로 네트웍 지연 시간이 큰 영향을 미치고 있어, 향후 기가비트 네트웍 등을 채택하면 보다 나아질 것으로 기대된다.

4. 결론

본 연구에서는 Maya 렌더러를 채용한 PC 클러스터 기반의 병렬 렌더링 시스템을 구축하고, 병렬화 성능을 높이기 위한 효과적인 부하 균형 기법을 개발하였다. 특히 공개 소프트웨어인 POV-Ray[17]를 활용한 기존의 방법과 달리, 본 시스템에서는 영상 정보를 분석하거나 렌더링의 중간 과정에 개입하지 않는다. 대신 애니메이션을 구성하는 연속 프레임 작업에서 프레임 간의 연관성(coherence)이 높다는 사실에 근거하여, 임의 프레임의 각 분할 영역에 소요된 계산량을 바탕으로 다음 프레임의 부하 분포를 예측하고 이에 맞게 각 프로세서의 작업 영역을 재조정하는 기법을 제안하였다. 따라서, 이 방법은 렌더링 시스템의 특성에 독립적으로 동작한다.

정적 분할 방법은 구현이 쉬운 반면 객체 분포가 균일하지 못한 영상의 경우에 작업 불균형이 크게 나타나는 문제가 있다. 이를 개선 방법으로 작업 영역을 더욱 세밀하게(fine-grained) 분할할 수 있는데, 이 방법은 부하 불균형은 일부 완화시킬 수 있지만, 각 프로세서는 자신의 작업 영역을 배정받기 위해 잦은 서버 간 통신을 요구하고, 더욱 큰 문제는 계산 과정에서 각 프로세서가 확보한 충분한 지역성을 활용하기 어렵다는 것이다.

제안 기법의 성능을 평가하기 위해, 충분하지는 않지만 2개의 실제 애니메이션 데이터에 대한 적용 결과, 정적 분할에 비해 약 40% 가량의 성능 향상을 보였다. 또한 다양한 부하 분포에 대한 각 기법의 성능을 추정하기 위해 수행한 모의실험에서, 정적 분할 기법에 대해 부하 균형, 확장성 측면에서 우월한 것으로 예측되었다. 그러나, 만일 프레임 간의 연관성이 낮은 경우, 즉 프레임 간의 데이터가 크게 달라지는 경우에는 부하 재조정 효과를 기대할 수 없다는 점이 문제이다. 따라서, 프레임 간의 연관성이 낮은 데이터에 대한 적용 방법을 향후 연구를 통해 개선할 필요가 있다.

참고 문헌

[1] J. S. Montrym, D. R. Baum, D. L. Dignam, and C. J. Migdal, "InfiniteReality: A Real-time Graphics System", In Proc. of Computer Graphics (SIGGRAPH97), pp. 293-303, 1997.

[2] R. Samanta, J. Zheng, T. Funkhouser, K. Li, and J. Pal Singh, "Load Balancing for Multi-Projector Rendering Systems", SIGGRAPH Eurographics Hardware Workshop in Computer Graphics, pp. 107-116, 1999.

[3] R. Samanta, J. Zheng, T. Funkhouser, K. Li, and J. Pal Singh, "Hybrid Sort-First and Sort-Last Parallel Rendering with a Cluster of PCs", SIGGRAPH Eurographics Hardware Workshop in Computer Graphics, 2000.

[4] Bengt-Olaf Schneider, "Parallel Rendering on PC Workstations", Int'l Conf. on Parallel and Distributed Processing Techniques and Applications, 1998.

[5] I. Sutherland and G. Hodgman, "Reentrant Polygon Clipping", Comm. of the ACM, 17(1), 1974.

[6] Carl Mueller, "The Sort-First Rendering Architecture for High Performance Graphics", Computer Graphics, ACM SIGGRAPH Special Issue, 1995.

[7] Carl Mueller, "Hierarchical Graphics Database in Sort-first", In Proc. of the IEEE Symp. on Parallel Rendering, pp. 49-57, 1997.

[8] Pixar, "PhotoRealistic Rendering Toolkit", 1998.

[9] POV-Team, "POV-Ray 3.5 Documentation", http://www.povray.org/documentation/, 2003.