韓國 컴퓨터情報學會 論文誌
第13卷 第1號, 2008. 1.

2008-13-1-2-4

작업영역의 동적 할당을 통한 고화질 애니메이션의 병렬 렌더링

이 윤 석*

Parallel Rendering of High Quality Animation based on a Dynamic Workload Allocation Scheme

Yunseok Rhee *

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 POV-Ray 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. 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.

Abstract

Even though many studies on parallel rendering based on PC clusters have been done, most of those did not cope with non-uniform scenes, where locations of 3D models are biased. In this work, we have built a PC cluster system with POV-Ray, a free rendering software on the public domain, and developed an adaptive load balancing scheme to optimize the parallel efficiency. Especially, we noticed that a frame of 3D animation are closely coherent with adjacent frames, and thus we could estimate distribution of computation amount, based on the computation time of previous frame. The experimental results with 2 real animation data show that the proposed scheme reduces by 40% of execution time compared to the simple static partitioning scheme.

▸ Keyword : 고화질 애니메이션(High Quality Animation), 병렬 렌더링(Parallel Rendering), 계산량 추정(Computation Workload Estimation), 프레임 연관성(Frame Coherence), 작업량 균일성(Workload Uniformity)

• 제1저자 : 이윤석
• 접수일 : 2007. 12. 3, 심사일 : 2008. 1. 7, 심사완료일 : 2008. 1.25.
* 한국외국어대학교 전자정보공학부 교수
※ 이 논문은 2001년도 한국학술진흥재단 지원에 의하여 연구되었음(KRF-2000-003-E00266)

Ⅰ. 서 론

3차원 렌더링은 고화질 의료 영상, 게임, 애니메이션, 과학 데이터 가시화 (visualization) 등의 분야에서 널리 활용되는 핵심 기술이다. 신형 자동차의 모델링 작업 등에서 대형 스크린에 실물 크기의 고화질 모델 영상을 비추고 협동 작업을 하거나, 의료진이 가시화 데이터를 실시간으로 관찰하며 치료 과정을 논의하는 등은 쉽게 생각할 수 있는 활용 사례이다.

특히 수백만 화소 이상의 고화질 영상을 실시간으로 생성해야 하는 어플리케이션에서는 막대한 계산량으로 인해 고가의 렌더링 장비들이 사용되고 있다. 수백만 개의 다각형으로 이뤄진 3차원 모델을 초당 약 30 프레임의 속도로 렌더링하려면 대개는 SGI Infinite Reality 엔진 [1]과 같이 고가의 전문 렌더링 시스템을 구입해야 했다.

그러나 PC의 성능이 기가 플롭스(flops) 대에 육박하고, 이들을 Fast Ethernet이나 Myrinet과 같은 고속 네트웍으로 연결한 PC 클러스터가 새로운 병렬 컴퓨팅 시스템으로 등장하면서, 클러스터를 활용한 3차원 렌더링에 대한 연구도 본격화되고 있다.[2, 3, 4] 더욱이 클러스터 시스템은 비교적 싼 가격의 상용 부품을 사용하여 시스템 구성이 유연하고 확장성이 뛰어난 특징을 갖는다. 따라서 가격 성능비는 기존의 고성능 렌더링과 비교할 수 없이 우수하며, 심지어 PC에 장착된 그래픽 가속기 중에는 SGI 그래픽 엔진에 맞먹는 성능을 보이는 것도 개발되었다.

효과적인 병렬 렌더링 시스템을 개발하기 위해 우선 고려할 점은 각 프로세싱 노드의 처리 성능을 극대화하고 병렬화 오버헤드를 최소화함으로써, 확장성이 우수한 알고리즘을 고안하는 것이다. PC 클러스터 기반의 렌더링 시스템에서도 모든 프로세서들에 가능한 고르게 작업 부하를 할당하여 프로세서 활용율을 극대화하고, 병렬화에 따른 프로세서 간의 통신 부담을 최소화해야 한다. 오래 전부터 sort-first, sort-middle, sort-last [5] 등의 작업 분할 기법을 통해 병렬 렌더링 작업 부하를 배분하는 연구들이 있었지만, 클러스터 시스템이 기존의 강결합(tightly-coupled) 다중 프로세서와 다른 특성을 지녀, 이 연구의 결과들을 동일하게 적용하거나, 기대한 효과를 얻기 어려운 실정이다.

고성능 강결합 다중 프로세서와 비교할 때, PC 클러스터는 공유 메모리에 대한 접근 속도가 느리고, 프로세서 간의 통신 지연시간이나 네트웍 대역폭이 좋지 않다. 더구나 상용 그래픽 가속기를 활용하거나 Maya [6] 등과 같은 상용 렌더링 소프트웨어를 사용하는 경우, 렌더링의 과정에서 생성되는 중간 결과(예를 들면 기하학적 변환 후의 3차원 객체)에 접근하는 것이 어려워, 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 가능한 많은 수의 프로세서를 통해 병렬화를 높이는 것이 바람직하고, 이는 정지(still) 영상뿐 아니라 애니메이션과 같은 연속 영상의 생성에도 마찬가지 효과를 보일 것으로 기대된다. 따라서 프레임 내에서의 작업 영역을 분할하는 시도를 통해 각 프로세서들의 처리 효율을 극대화하면 클러스터 시스템의 성능 제약을 극복하고 실시간 렌더링이 가능할 것으로 기대한다.

앞서 언급한 대로, 클러스터 기반의 병렬 렌더링 시스템에서는 렌더링 프로세스의 중간 과정에 개입하기 어렵고, 따라서 전체 화면 영역을 같은 크기로 나눈 후에 각 프로세서에게 한 영역씩 배정하는 것이 손쉬운 방법이다. 그러나, 일반적으로 3차원 객체 데이터의 분포가 다르고, 각각의 계산량도 크게 달라, 화면 영역의 균등 분할이 계산량의 균등 분배를 보장하지 않는다. 다시말해, 임의의 영상을 구성하는 3차원 모델들의 계산량 분석이나 사전 정보없이 단일 영상의 렌더링 작업량을 고르게 분할하기는 기술적으로 어렵다.

이와 달리, 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. 위의 사실을 바탕으로, 본 연구에서는 PC 클러스터를 기반으로한 병렬 렌더링 시스템을 구축함에 있어, 3차원 그래픽스를 활용한 애니메이션 렌더링 작업을 효과적으로 수행하도록 프레임의 연관성과 계산량 정보를 활용하여 부하를 추정하고, 작업 부하를 균형있게 배분하는 기법을 설계하고자 한다. 특히, 본 연구에서는 고가의 상용 렌더링 시스템에 의존하지 않고, 공개된 렌더링 소프트웨어인 POV-Ray[9]를 활용하여 병렬 렌더링 클러스터 시스템을 구성하였다.

본 논문의 2장에서는 이제까지의 병렬 렌더링 기법에서 사용된 영역 분할 기법들과 프레임 연관성을 활용한 관련 연구들을 간략히 소개하고, 3장에서는 본 연구에서 제안하는 시스

템 구성 방법과 작업 분배 알고리즘을 설명할 것이다. 4장에서는 실제 애니메이션 렌더링 결과를 통해 성능을 보이고, 함께 다양한 분포를 갖는 모의 데이터를 통해 성능을 예측하고, 마지막으로 5장에서 결론을 맺는다.

II. 관련 연구

기존의 연구들은 대부분 단일 영상에 대해, 계산량에 따라 적응적으로 영역을 분할하는 방법들을 제안하였다. Whelan의 Median-cut 알고리즘[10]은 영상을 구성하는 각 객체들의 중심(centroid) 분포에 따라 화면을 분할한다. 분할선(cuts)은 전체 분할 영역의 수가 프로세서의 수와 같아질 때까지 재귀적으로 화면 영역을 계속 분할해 간다. 그러나, 이 방법은 기본적으로 3차원 객체 정보의 접근이 가능해야 하고, 그 수가 많은 경우에는 화면 분할 과정이 오래 소요될 뿐 아니라, 각 객체의 중심만을 살피고 크기를 고려하지 못해 올바른 분할이 이뤄지지 않는 문제가 있다.

한편 Whitman이 제안한 top-down 분할 기법 [11]은, 화면 영역을 매우 세밀한 메쉬로 나누고, 각 3차원 객체의 외곽(bounding box)에 포함되는 메쉬 셀들에게 점수를 부여해 간다. 인접한 메쉬 셀들을 하나의 영역으로 결합하고, 가장 많은 객체들이 포함된 영역부터 나누어 가는 방법이다. 그러나, 이 방법 역시 생성된 분할 영역들이 포함하는 객체의 수에 있어 여전히 큰 편차를 보일 수 있다. 이 연구에서는 프로세서의 수의 열배 만큼의 분할 영역을 만들고 이들을 동적으로 할당, 가능하면 부하량의 편차를 줄이려 했다. 그런데 세밀한 영역 분할은 프로세서의 잦은 영역 배정 요구로 인해 상당한 오버헤드를 동반하고, 한 객체가 지나치게 분할되면서 서로 다른 프로세서에 의해 처리될 경우에, 앞서 계산한 결과를 효과적으로 활용할 수 없게 된다. 즉, 데이터의 공간적 지역성(spatial locality)을 잃게 되는 것이다.

또한 Samanta 등의 연구에서는 다수의 프로젝터를 사용한 고화질 표현을 위해 PC 클러스터를 활용하였으며 [2, 3, 4], sort-first 기법[12]을 적용하여 전체 화면 영역을 타일(tile)이라는 작은 영역들로 분할하여, 이들을 각 프로세서에게 차례로 할당한다. 이를 위해, 3차원 영상 데이터를 분석하여 각 객체가 속한 타일 영역을 파악하고, 해당 객체들의 계산량을 추정하여 각 타일의 누적 계산량을 구하고, 이를 적절히 분배해 부하 균형을 이루는 방법이다. 그러나, 이 방법은 영상 자료를 분석할 수 있는 경우에 한해 적용이 가능하며 상용 모델러를 사용한 대단한 복잡한 영상 데이터를 분석하는 것은 거의 실현이 어렵다.

한편 본 연구를 뒷받침하는 연속 영상의 특징으로, 프레임 연관성(frame coherence)을 들 수 있는데, 이는 인접 프레임 간의 화소 데이터들의 연속성, 즉 변화량이 매우 적은 경우에 높은 연관성이 있음을 뜻한다. 즉 [그림 1]에 보이는 것처럼, 각 프레임에 나타나는 객체들의 연관성과 시간적 연속성이며, 동일 객체가 거의 같은 지점에 위치하므로 나타나는 현상이다.

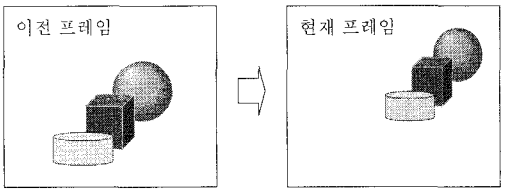

그림 1. 프레임 간 연관성을 보이는 예
Fig 1. Inter-frame coherence example

프레임 연관성은 Sutherland 등이 렌더링에 적용할 수 있는 여러 형태의 연관성들을 소개하면서 처음 언급하였으나 [13], 연관성을 활용하여 병렬화를 적용하려는 시도는 없었다. 다각형 기반의 렌더링 분야에서, Hubschman과 Zucher는 이동 카메라와 고정 물체를 갖는 영상들에서 프레임 연관성을 활용하여 영상 생성 시간을 줄이는 초기 알고리즘을 제안했다 [14]. 이 방법은 카메라의 위치에 따른 두 프레임 간의 차이를 구하고, 차이를 보이지 않는 대부분의 영역은 이전 프레임으로부터 화소를 복사하는 과정을 통해 새 프레임을 구성한다. 또한 프레임 연관성은 스테레오 영상을 생성할 때도 활용될 수 있는데, Adelson 등이 연구한 바로는, 좌안 영상(left-eye view)과 우안 영상의 차이는 약 5%에 불과했다 [15]. 이와 같이 매우 높은 영상의 연관성을 선험적으로 활용하면 50% 이상의 계산 시간을 줄일 수 있다고 주장한다. 이 외에도 광선 추적 알고리즘에서 이전 프레임에서 사용된 모든 광선 정보를 저장하고, 다음 프레임에서는 카메라의 이동에 의해 바뀐 광선 정보만을 갱신하는 Reprojection 알고리즘 [16]이나 점증적 광선 추적법 [17] 등도 프레임의 연관성을 적극 활용하는 연구에 해당한다.

이와 같이 프레임 간의 연관성은 연속 영상 분야에서 적극적으로 활용되는 특징임을 알 수 있으나, 이 성질을 프레임의 계산량을 추정하고 예측하는데 활용하여 부하 배분에 적용한 연구 사례는 없었다.

III. 병렬 렌더링클러스터의 설계 구현

3.1 시스템 구성

본 연구에서는 먼저 [그림 2]에 보이는 것처럼, 리눅스를 탑재한 PC 클러스터에 POV-Ray 3.6 렌더러를 설치, 렌더링 팜(farm)을 구축하고, 3차원 그래픽스를 이용한 애니메이션 프레임을 생성하는 서비스를 구현하였다. 클러스터는 전체 8개의 노드이며, 각 노드는 팬티엄 III 933MHz, 512MB 메모리를 갖추었다. 노드들은 100MB/s Fast Ethernet으로 연결되며, 하나의 Front-end 서버가 작업 요청을 접수하고, 계산량에 따라 작업 영역을 Back-end 서버들에게 배정하는 기능을 수행한다.

POV-Ray 렌더링 엔진이 서버 간의 통신을 지원하지 않기 때문에, 본 시스템에서는 각 노드의 외부 인터페이스를 담당하는 중개자(agent) 프로세스를 구현, 데몬의 형태로 실행시킨다. 중개자 간의 통신은 TCP 소켓으로 구현하였고, 중개자가 렌더링 명령을 받으면 쉘 명령을 통해 POV-Ray 렌더러를 실행한다.

클라이언트로부터 전달받은 3차원 영상 데이터는 Front-end 서버가 저장한 후, 모든 노드에 동보 전송하여 각 노드가 독립적으로 관리하게 한다. 이는 실행 중에 영상 데이터에 읽기 위해 빈번하게 네트웍에 접근하는 것을 방지한다. 작업 시작 시점에 사용자로부터 영역 분할에 대한 사전 정보가 없으면, Front-end 서버는 참여 노드의 수만큼 생성할 화면 영역을 해상도에 따라 가로, 세로 방향으로 균등 분할하고, 작업 프레임 번호와 해당 작업 영역 정보를 각 서버 노드에 전달한다.

각 서버 노드에서 전달받은 영역에 대한 계산이 완료되면 생성된 부분 영상과 계산 과정에 얻은 자원 사용 정보(CPU time, Memory 사용량, 현재 부하상태 등)를 함께 Front-end 서버에게 전달한다. 모든 서버들이 해당 프레임의 할당된 작업을 완료하면, Front-end 서버는 다음 프레임의 작업 할당을 준비한다. 이 때 각 노드로부터 수집한 자원 사용 정보를 바탕으로, 작업 영역을 재조정하며, 다시 다음 프레임 번호와 조정된 작업 영역 정보를 전달하여 병렬 작업을 반복적으로 수행시킨다.

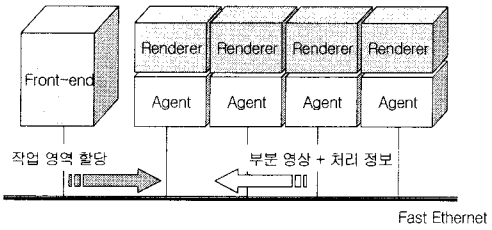

그림 2. 클러스터 기반 렌더링시스템의 구성
Fig 2. Cluster-based Rendering System

3.2 프레임 연관성을 활용한 작업 영역 분할

sort-first 기법[16]은 병렬 렌더링 시스템에서 채택하는 가장 일반적인 병렬화 방법으로, 작업 초기에 각 프로세서에게 화면 분할을 할당하고, 그래픽 객체들을 영역에 따라 정렬시켜 각 담당 프로세서에게 전 과정의 처리를 맡기는 방식이다. 그러나, 이 방식은 영상 내의 객체들이 고르게 분포되지 않은 경우에 특정 프로세서에 부하가 집중되어 병렬화 효과가 나쁘며, 이런 부하 불균형의 문제는 시스템의 규모가 클수록 더욱 심각해진다.

본 연구에서는 초기에 서버 노드의 수에 맞게 화면 영역을 분할하는 sort-first 기법을 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 w1, w2, w3, w4의 계산량 비율을 보였고, w1+w2 〉 w3 + w4, w1+w3 〉 w2+w4, w1 〉 w2, w3 〈 w4 ($0 \leq w_i \leq 1$, $\sum w_i = 1$)라고 가정하자. 즉, A 영역에 객체 분포가 집중된 경우이다.

처음에는 당연히 A, B, C, D가 동일한 크기를 같지만, 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. 만

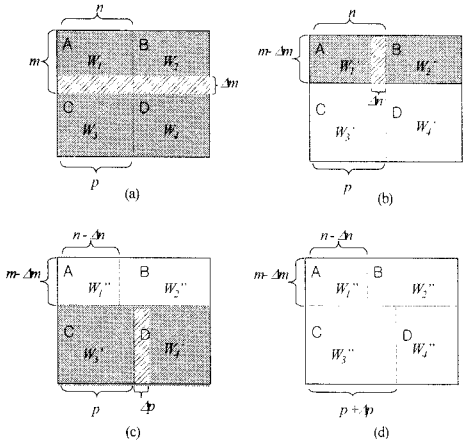

그림 3. 이전 프레임의 계산량에 따른 분할 영역 재조정 과정
Fig 3. Allocation Window Adjustment based on the Previous Workload

일 하나의 분할 영역 내에서는 부하가 고르게 분포한다고 가정한다면, 분할선은 계산량이 큰 A, B를 Δm 만큼 줄이도록 이동해야 하고, 이동 후에 (w1+w2)-(w1+w2) * Δm / m = (w3+w4)+(w1+w2) * Δm / m 이어야 한다. 정리하면, Δm = (w1+w2-w3-w4) * m / 2 이 되며, 새로 조정된 수평 분할점까지의 거리 m′ = m - Δm 이 된다. 이는 하위 영역 C, D의 계산량 w3+w4가 큰 경우에도 대칭적으로 적용할 수 있으므로 설명을 생략한다.

[그림 3(b)]는 수평 분할점이 조정한 후의 상태를 보이는데, m′으로 조정된 후의 분할 영역 A′, B′, C′, D′ 은, 각각 아래와 같은 w1′, w2′, w3′, w4′ 의 계산량을 갖는다고 추정할 수 있다.

$$w1' = w1 - \Delta m * n$$
$$w2' = w2 - \Delta m * (h - n)$$
$$w3' = w3 + \Delta m * p$$
$$w4' = w4 + \Delta m * (h - p)$$

다음으로는 [그림 3(b)]의 조정된 상위, 하위의 각 분할 영역들에 대해 수직 분할선을 조정하는 과정을 수행한다. 상위 두 영역의 추정 계산량이 w1′ > w2′일 때, 분할선은 w1′을 Δn 만큼 줄이도록 조정되어야 하며, w1′-Δn = w2′ + Δn이어야 하므로, Δn = (w1′-w2′)* n / 2 로 조정되어야 한다. 대소 관계가 반대인 경우에도 대칭적으로 적용되므로 설명은 생략한다. 하위 영역 역시 w3′, w4′의 대소 관계에 따라 똑같은 과정으로 분할선 조정이 필요하다.

이상은 노드의 수가 4인 경우에 관한 예를 보였지만, 일반적으로 2N 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.

IV. 실험 및 결과분석

4.1 애니메이션 렌더링의 결과

실제 애니메이션 렌더링에서의 효과를 알기 위해, [표 1]에 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 움직이는 애니메이션으로 프레임 내에서는 계산량이 편중되어 있으나 전체적으로는 작업량이 비교적 널리 분포된 형태를 보인다.

정적인 균등 분할과 제안된 기법을 적용하여 비교 실험을 실시하였으며, [표 2]에 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 1. 실험에 사용된 애니메이션 특징
Table 1. Feature of Animations used in Experiments

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

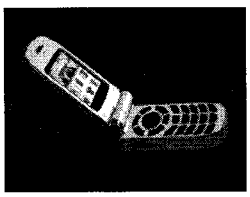

그림 4. Cell-Phone의 한 프레임
Fig 4. A Frame of Cell-Phone

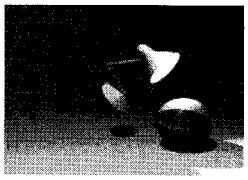

그림 5. Lux-Ball의 한 프레임
Fig 5. A Frame of Lux-Ball

〔표 2〕의 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 네트워크 지연 시간이 큰 영향을 미치고 있어, 향후 기가비트 네트워크 등을 채택하면 보다 나아질 것으로 기대된다.

4.2 모의실험에 의한 성능 평가

다양한 형태의 실제 애니메이션 데이터를 사용하여 성능을 측정하는 것이 보다 현실성있는 결과를 얻을 수 있으나, 대부분의 3차원 모델과 애니메이션 데이터가 소유권이 보호되는 저작물로 실험 자료를 수집하는데 어려움이 있다. 그리고 설령 다양한 데이터들을 구해도 해당 데이터들이 계산량 분포를 객관적 지표로 나타내기도 어려운 것이 사실이다. 따라서, 본 절에서는 모의 3차원 영상 데이터를 모델링하여 그 계산량 분포를 인위적으로 구성하고, 이에 대한 각 기법의 처리 성능을 추정하는 모의실험 결과를 제시할 것이다.

다양한 계산량 분포를 갖는 모의 데이터를 구성하기 위해, 계산량의 균일한 분포 정도를 정의할 필요가 있으며, 본 연구에서는 이를 균일성(uniformity, 이하에서 u)이라 부르고 다음과 같이 정의한다.

$$u = 1.0 - 2^* \sum_{i=1}^{n} |w_i - \overline{w}| / (n-1)^* \overline{w}$$

이 식에서 n : 분할영역의 수

w_i : 각 분할영역(i)의 실제 계산량

$\overline{w}$: 분할영역 당 평균 계산량.

즉, 전체 계산량의 n개의 영역에 고르게 분포되어 있으면 1.0, 절반의 영역에 집중되어 있으면 0.5, 한 곳에 집중되어 있으면 0.0으로 계산된다. 주어진 균일도를 만족하는 모의 데이터는 각 분할된 영역에 적절히 계산량을 할당한 배열 자료로 쉽게 구현하였다. 실험에서는 프레임 연관성이 높은 경우와 애니메이션을 구성하는 프레임의 수가 매우 큰, 즉 시간이 긴 영상을 생성하는 경우를 가정했고, 이에 따라 영상에 대한 사전 정보가 없는 시작 상태의 영향은 고려하지 않았다.

실험 방법은 한 프레임 생성에 평균적으로 64초가 걸리는 모의 3차원 영상 데이터에 대해, 정적 분할(static), 세분화 정적 분할(static4), 제안 기법(proposed)의 성능을 모의 실험 프로그램을 통해 측정하였다. static 기법은 초기에 같은 크기로 분할한 작업 영역을 고수하는 방법이고, static4 기법은 분할 영역을 균등하게 나누되 참여하는 프로세서 수의 4배만큼 세밀하게 영역을 분할하여 부하 균형을 이루려는 방법이다.

표 2. 렌더링 실험 결과
Table 2. Rendering Experiment Results

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

측정 항목은 각 프레임 생성에 소요되는 CPU 프로세싱 시간(process), 부하 불균형으로 불필요하게 소요되는 시간(imbalance), 그리고 static4의 경우에 작업 할당 서버로부터 아직 처리되지 않은 작업 영역을 얻기 위해 요구되는 상호 통신 시간(comm)으로 구성된다.

본 모의 실험은 계산량 분포에 따른 각 기법의 성능 추이를 살펴보는 의도이므로, 〔그림 6〕에 보이는 바와 같이 균일성이 0.75, 0.25인 두 경우를 대표적으로 살펴보았다. 쉽게 알수 있듯이, process 시간은 계산량과 참여한 프로세서의 수에 의해 결정되고, 작업 분할 기법과 무관하다. 그러나, static과 static4 기법에서는 각각 고정된 작업 영역 할당으로 인한 계산량의 차이가 나타나고, 이는 불균형 시간(imbalance)으로 이어진다. 정적 분할의 경우에도, 영역의

크기 정도(granularity)가 큰 static 기법에서 상대적으로 컸으며, 이는 프로세서의 수가 늘어나 작업 영역이 보다 작아지면서 감소하는 경향이 있었다. 그러나, static 4 기법의 경우, 불균형 시간이 상대적으로 적어진 대신, 세분화된 만큼 자주 작업 영역을 전달받기 위한 통신 오버헤드를 감수해야 했다. 그러나, 최근 기가비트 스위치들이 등장하면서, 통신 오버헤드는 무시해도 될 만큼 네트웍 성능이 향상되었다. 그럼에도, 특정 객체가 포함된 영역이 지나치게 세분화되면, 분할 영역들이 서로 다른 프로세서에 의해 처리될 수 있고, 이 경우에 프로세서 자원들이 상당한 동일 계산에 낭비되거나, 캐쉬이나 광선 자료 구조 등과 같이 어느 프로세서에 의해 기껏 확보된 지역성(locality)이 제대로 활용되지 못하는 문제가 오히려 심각할 수 있다. 또한 정적 분할 기법들은 균일성이 저하되면 그에 상응하는 성능 저하를 초래한다. 예를 들어, static 기법은 균일성이 0.75인 경우에 비해 0.25인 경우, 약 32%의 처리 시간이 증가하였고, static4 기법의 경우에는 39%가 증가하였다.

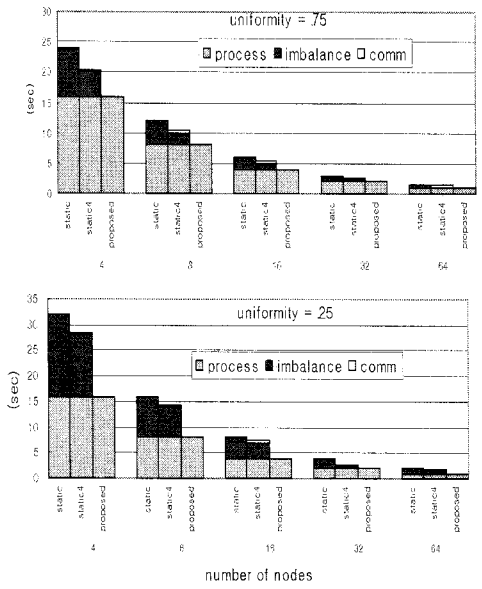

그림 6. 균일성 (uniformity) 에 따른 작업 분할 기법의 성능 추이 (모의실험 결과)
Fig 6. Performance of the Proposed scheme with Uniformity

이에 비해 제안한 기법은 부하 분포에도 매우 적응적이며, 당연히 불균형으로 인한 낭비 시간도 거의 없다. 다만, 매 프레임마다 작업 영역 재조정을 위한 알고리즘이 수행되어야 하므로, 이 알고리즘의 복잡도에 따른 오버헤드가 성능 향상을 감소시킬 수 있지만, 성능 이득을 넘을 정도는 아니다. 그러나, 본 실험에서 가정한 프레임의 연관성이 나타나지 않는 영상 데이터에 대해서는 목적하는 효과는 얻지 못하고 오히려 부하 조정 오버헤드만이 가중되어 다른 기법들에 비해 성능이 떨어질 수 있으며, 이에 대해서는 향후 연구에서 프레임의 연관성 정도를 판단하여 제안 기법의 적용 여부를 판단하는 과정을 고안하여야 할 것이다.

VI. 결론 및 향후 과제

본 연구에서는 POV-Ray[9]를 채용한 PC 클러스터 기반의 병렬 렌더링 시스템을 구축하고, 병렬화 성능을 높이기 위한 효과적인 부하 균형 기법을 개발하였다. 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.

참고문헌

[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] T. Crockett, "Parallel Rendering", Technical Report TR95-31, Inst. of Computer Applications in Science and Engineering, NASA Langley Research Center, 1995.

[6] Alias Wavefront Technical Support, "Maya Community Online Tutorial", http://www.alias.co- m/eng/support/maya/faqs-tutorials/index.jhtml, 2003.

[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 3.6 Documentation", http://www.povray.org/documentation/, 2006.

[10] Deniel Whelan, "Amimac: A Multiprocessor Architecture for Real-Time Computer Animation", Ph.D. Dissertation, California Institute of Technology, 1985.

[11] Scott Whitman, "Multiprocessor Methods for Computer Graphics Rendering", AK Peters, Wellesley, Massachusetts, 1992.

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

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

[14] H. Hubschman and S. Zucker, "Frame-to-frame coherence and the hidden surface computation: Constraints for a convex world", In Proc. of SIGGRAPH '81, 15(3), pp. 45-54, 1981.

[15] S. Adelson and L. Hodges, "Stereoscopic ray-tracing", The Visual Computer, 10(3), pp. 127-144, 1993.

[16] S. Adelson and L. Hodges, "Generating Exact Ray-traced Animation Frames by Reprojection", IEEE Computer Graphics and Applications, 15(3), pp. 43-52, 1995.

[17] K. Murakami and K. Hirato, "Incremental Ray Tracing", In Proc. of the Eurographics Workshop on Photosimulation, Realism, and Physics in Computer Graphics, pp. 15-29, 1990.

저 자 소 개

이 윤 석 (Yunseok Rhee)

1988년 2월: 서울대학교 계산통계학과 학사

1999년 2월: 한국과학기술원 전산학과 박사

1999년-현재 : 한국외국어대학교 교수

관심분야: 분산시스템, 임베디드컴퓨팅