2001년 한국정보처리학회 추계 학술발표논문집 제8권 제2호

병렬 광선 추적 알고리듬의 성능

이효종, 임범현*
전북대학교 전자공학과

{hlee, bhlim}@sel.chonbuk.ac.kr

Performance of Parallel Ray Tracing Algorithm

Dept. of Electronics, Chonbuk National University

요 약

광선추적기법은 사진과 같은 고해상도의 영상을 만들어내는 렌더링 기법중의 하나이다. 이 기법은 이미지를 합성하는데 많은 양의 계산 시간을 필요로 한다. 병렬처리 기법이 광선추적에 계산양의 처리 기간을 감소하기 위하여 사용될 수 있다. 본 논문에서는 병렬 광선추적 기법을 MPI(Message Passing Interface)를 사용하여 IBM Supercomputer 상에서 노드의 개수의 증가에 따른 속도 향상과 노드간에 전달되는 메시지의 크기에 따른 성능 향상을 실험하였다. 본 논문에서 실험한 병렬 광선 추적 기법으로 IBM SP 시스템 상에서 다양한 영상을 생성하였다. 영상은 분할가능하고 노드에 분배할 수 있기 때문에 병렬화 범주에 들수 있으며 부하균형을 맞출 수 있다. 실험에서 프로세서수의 증가에 따른 이상적인 속도향상률(Speed-up rate)을 15개의 프로세서를 사용하여 얻을 수 있었다. 광선을 추적하여 영상을 합성해 낼 때 표현하고자 하는 영상이 단순한 객체로 이루어져 있다면 각 노드에 분산해줘야 할 작업의 크기는 복잡한 객체들로 구성된 영상보다 클 때 더 높은 성능을 나타내었다. 분산작업의 크기가 작아 상대적으로 통신횟수가 증가할 때 렌더링시 효율저하를 나타내었다.

1. 서론

빛은 물체의 표면상에서 반사, 굴절, 흡수, 투과될 수 있다. 일반적으로 인간은 물체들에게서 반사되는 빛을 통해서 물체를 인지할 수 있다. 복잡한 물체들로 구성되어진 사진과 같은 고해상도의 영상을 얻어내는 기술들은 이미 오래 전에 컴퓨터그래픽스에서 제시되었다. 물체 표면과 빛의 진행 방향사이의 각도의 차이에 기반을 둔 쉐이딩 알고리즘은 [8] 밝기를 보간 한다. 반면에 광선추적 기법[11, 12]은 컴퓨터에서 현실적으로 보이는 영상을 만들어내는 기술로서 사용되어왔으며 가장 강력한 기법 중 하나로 인식되어왔다.

광선추적기법에서 광선은 물체표면의 반사율과 투명도에 의존해서 반사되고 굴절된다. 이러한 반사와 굴절은 모든 교차점에서 반복적으로 이루어지며 이를 통해서 빛의 강도가 바뀌게 된다. 광선추적기법은 수 백만 개의 광선의 이동경로 추적을 위해서 엄청난 양의 부동소수점 계산을 필요로 하며 이는 많은 처리시간을 요구하다. 처리시간은 영상이 복잡한 물체들로 구성되었을 때 더 많이 요구되어 진다.

계산시간을 감소시키기 위한 광선추적 알고리즘이 학계에 발표되었다[7,1]. 광선이 이동하는 방향은 독립적으로 측정될 수 있기 때문에 몇몇 과학자들은 광선추적을 병렬 환경에서 추적하는 연구를 발표하였다[6, 9, 10, 5]. Gaudet[6]는 다중프로세서 환경에서 광선추적알고리듬을 적용하였고 Freisleben et al. 과 Kwon et. al은 각기 TMS320[9]과 웍스테이션 클러스터에 적합한 광선추적 알고리듬을 발표하였다. 본 논문에서는 IBM Supercomputer상의 각 노드의 상태를 확인해서 각 노드의 상태에 따른 작업량을 분할해주는 광선추적 알고리듬을 적용했다. 병렬 광선추적 알고리듬을 적용하여 프로세서의 수와 분

산노드에 적용할 작업량의 크기에 따른 성능 향상을 측정하였다. 본 논문은 2절에서 순차적 광선추적 알고리듬을 기술하고, 3절에서는 병렬광선추적 알고리듬을 설명하였다. 4절과 5절에서는 구현한 알고리듬의 실험결과와 결론을 기술하였다.

2. 순차적 광선추적 알고리듬

광선추적기법은 순방향이든 역방향이든 눈으로 볼 수 있는 모든 광선을 추적한다. 순방향 광선추적기법은 빛이 지나가는 모든 경로를 추적하기 때문에 자연계에서 발생하는 현상을 정확하게 영상화 할 수 있다는 장점이 있다. 하지만 반사되는 빛들이 산란되는 경우 계산시간의 낭비를 가져온다.

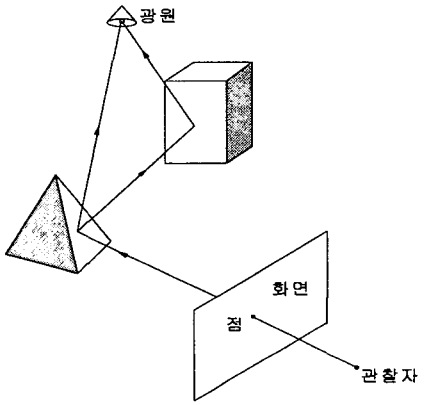

그림 1. 역방향 광선추적

역방향 광선추적기법은 이런 문제를 극복하여 하기 위하여 그림1과 같이 관찰자의 시야에 들어오는 화면, 그리고 실제 물체들에게 영향을 미치는 빛을 역방향으로 추적하는 방법이다. 이 빛이 실제 물체들과 부딪히는 면을 계산하고, 부딪히는 경우 그 물체의 특성에 따라 반사, 굴절, 투과현상에 합당한 빛의 세기를 계산한다. 이러한 반사, 굴절 및 투과된 빛들은 또 다시 다른 물체와 부딪히는지를 계산하고 이와 같은 과정을 계속 반복한다. 결국 빛의 세기가 임계값 이하로 약해지거나 빛이 어떤 물체와도 부딪히지 않을 때 그 화면에 대한 점의 빛 역추적 과정은 종료된다. 이 방법을 화면상에 존재하는 모든 점에 적용해서 완전한 영상을 구성하게 된다

3. 병렬 광선추적 알고리듬

광원에서 직접 투사되는 광선의 이동경로들은 독립적으로 계산될 수 있기 때문에 빛 모델은 쉽게 병렬화 할 수 있다. 분산 컴퓨터 환경 안에 광선추적 알고리듬을 적용하면 각 노드의 효율을 증가시킬 수 있고 빠른 이미지 생성 시간을 기대 할 수 있다.

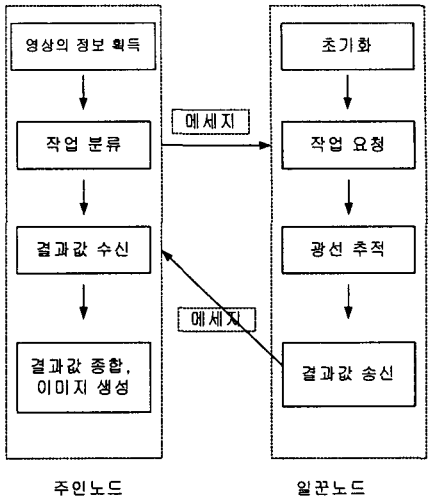

그림 2. 병렬 알고리듬

병렬 광선 추적 알고리듬은 그림 2 에서 볼 수 있듯이 슈퍼컴퓨터의 각 노드를 유일한 주인노드와 다수의 일꾼노드로 구성한다. 주인노드는 계급(rank)번호를 0으로 일꾼 노드는 0인 아닌 중복되지 않는 수로 정하여 준다. 각 주인과 일꾼 노드간의 메시지를 주고받는 통신은 MPI를 사용한다. 주인 노드는 전체적인 병렬 순서를 조절하고 일꾼 노드에 일을 분배하며 일꾼에서 만들어 낸 결과를 수집하는 일을 모든 작업이 끝날 때까지 수행한다. 일꾼 노드는 주인 노드로부터 작업을 받아 수행하고 그 결과 값을 주인 노드에 돌려주는 역할을 한다.

프로그램이 시작할 때 모든 노드들은 초기화되고 공유메모리를 통하여 프로그램이 각 노드들에 적재된 후에 주인 노드는 생성할 영상에 대한 정보를 토대로 영상을 각 노드에 분배하기 위하여 영상을 나누는 작업을 한다. 이때 모든 일꾼 노드들은 광원의 위치와 종류, 각 오브젝트에 대한 정 보를 읽어 들인다. 일꾼 노드에서 작업을 요청하면 주인 노드에서는 작업을 할당해 주며 이때 작업내용은 전체적인 영상을 구성하는 작은 영상의 위치를 지정해주는 것이다. 각 일꾼 노드들은 부분영상을 만들어 내는 작업을 주인노드에 부분영상이 남아 있지 않을 때까지 계속 수행한다. 일꾼노드들은 다른 일꾼노드들을 고려할 필요가 없고 주인 노드에 노드에서 주어진 작업만을 수행한다. 일률적으로 일꾼노드들의 상태에 관계없이 작업을 분배해주는 병렬 알고리듬과 비교했을 때 일꾼 상태를 고려한 병렬 알고리듬이 더 높

은 성능을 나타낸다고 알려져 왔다. 본 연구에서는 위의 알고리듬을 적용하여 노드 수를 최대 16개까지 증가하여 성능 향상을 분석해 보았고 동일한 일꾼노드 수 안에서 작업(부분영상)의 크기에 따른 영상 합성 시간의 차이를 얻을 수 있었다

4. 실험 결과

병렬 광선추적 알고리듬은 IBM SP Supercomputer 환경에서 실험되었으며 슈퍼컴퓨터는 32개의 노드로 구성되어지며 각 노드는 2개의 프로세서를 가지고 있다. 각 프로세서는 RISC Power-3 200Mhz의 성능을 나타내며 512Mb 메모리를 장착하고 있다. 실험에서는 16개의 프로세서를 사용하여 영상을 렌더링하였다. 병렬성을 적용하기 위하여 MPI Forum[4]에서 제공하는 MPI 2.0(Message Passing Interface)을 사용하여 메시지 통신을 하였다. 결과물은 800 x 600의 크기에 24bit 해상도를 가진 영상을 만들었다[그림9]. 실험엔 POV Ray[3]에서 제공한 광선추적 프로그램을 사용하였다. 실험에서 찻잔, 체스판, 변형체스판 영상이 입력으로서 사용되었다. 영상의 복잡도는 찻잔, 체스판, 변형체스판 순서에 따라 증가하며 이에 사용된 삼각형의 수는 대략 1.24×10^{24}, 3.44×10^{6}, 5.41×10^{6}이다. 각 이미지를 생성하는데 걸린 시간은 초단위로 측정되었다.

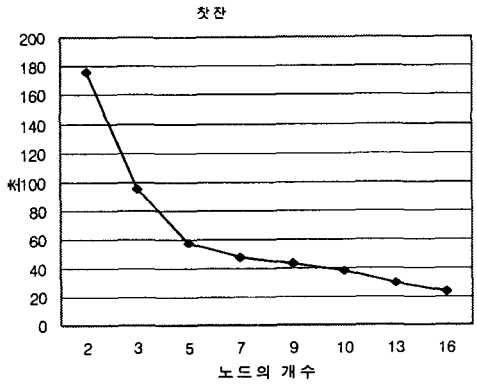

그림 3. 찻잔의 병렬광선추적 시간

그림 3-5는 병렬광선추적 알고리듬을 사용하여 각각 영상을 만들어내는 데 소비한 시간의 그래프이다. 노드의 수가 증가함에 따라 이미지 생성 시간은 선형적으로 감소하는 것이 관찰되었다.

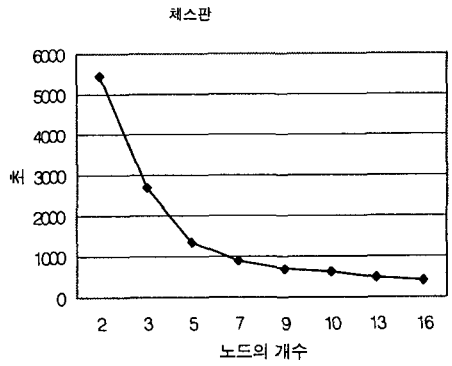

그림 4. 체스판의 병렬광선추적 시간

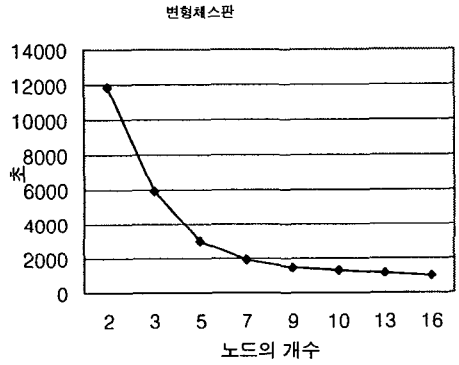

그림 5. 변형체스판의 병렬광선추적 시간

그림 6은 병렬광선추적 알고리듬을 적용한 실험결과의 속도 향상률을 도식화 한 것이다. 광선추적 알고리듬을 제시한 병렬화 알고리듬에 적용했을 때의 렌더링 속도는 프로세서의 수의 증가에 선형적으로 증가하였다. 제시한 병렬 광선추적 알고리듬의 속도 향상률은 이상적인 속도 향상률의 70-80%정도를 나타냈다

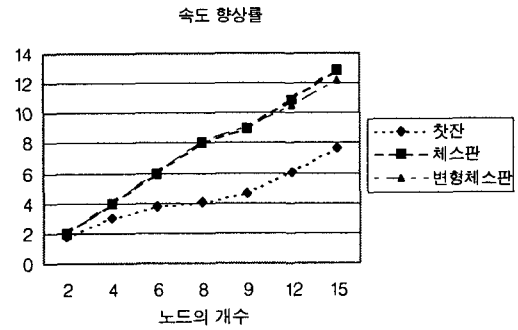

그림 6. 속도 향상률

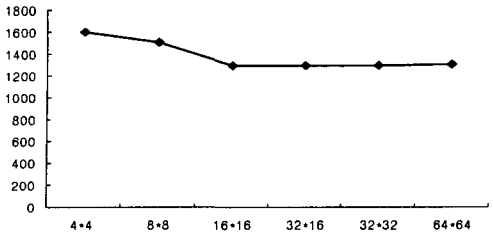

그림 7. 작업의 크기에 따른 광선추적 시간

그림 7은 변형체스판 영상을 인조 할 때, 일꾼 노드에서 계산하는 부분 영상의 크기의 변화에 따른 전체 영상의 생성 시간을 측정한 것이다. 크기가 작은 부분영상은 상대적으로 긴 렌더링 시간을 나타내었다. 어쨌든, 부분영상의 크기가 16×16일 때부터 렌더링 시간이 매우 큰 폭으로 감소한다는 것을 알 수 있었다.

(a) 찻잔

(b) 체스판

(c)변형체스판

그림 8 . 결과 영상

5. 결론

광선추적 알고리듬은 많은 계산시간을 소비하지만 기본적으로 병렬성을 포함하고 있다. 주인-일꾼 모델을 적용하여 전체 작업을 모든 노드에 노드의 상태를 고려하여 분배하여 노드의 효율성을 상승시키고 광선을 역추적하여 영상을 생성하는데 필요한 시간을 단축시킬 수 있었다. 또한 일꾼노드에서 계산하는 부분영상의 크기를 조절하는 것이 성능에 영향을 미친다는 것을 확인 할 수 있었다.

참고문헌

[1] J Arvo and D Kirk. Fast ray tracing by ray classifi-cation. *ACM Computer Graphics*, 21(4):55 64, July 1987.

[2] Ronald S Cok. *Parallel Programs for the Transputer*. Prenticd-Hall, 1986.

[3] A Enzmann, L Kretzschmar, and C Young. *Ray Tracing Worlds with POV-Ray*. The Waite Group, 1994.

[4] MPI Forum. *MPI-2:Extension to the Message Passing Interface*. University of Tennessee, 1997.

[5] B Freisleben, D Hartmann, and T Kielmann. Parallel raytracing: a case study on partitioning and scheduling on workstation clusters. *Conference on System Sciences*, 1:596 605, 1997.

[6] S Gaudet, R Hobson, P Chilka, and T Cavert. Multiprocessor experiments for high-speed ray tracing. *Transactions on Graphics*, 3(7):151 179, July 1988.

[7] A S Glassner. Space subdivision for fast ray traci ng. *IEEE Computer Graphics and Applications*, 4(10):15 22, October 1984:

[8] Donald Hearn and M. Pauline Baker. *Computer Graphics*. Prentice-Hall, 1996.

[9] Chang-Geun Kwon, Hyo-Kyung sung, and Heung-Moon Choi. An implementation of a parallel ray tracing algorithm on hybrid parall architecutre. *IEEE Conference on Acoustics, Speech and Signal Processing*, 3:1745 1748, August 1998.

[10] W Lefer. An efficient parallel ray tracing scheme for distributed memory parallel computers. *Parallel Rendering Symposium*, pages 77 80, August 1993

[11] Walter F Taylor. *The Geometry of Computer Graphics*.Wadsworth, 1992.

[12] N Wilt. *Object-Oriented Ray Tracing in C++*. The Wait Group, 1994.