2000년 한국정보처리학회 추계 학술발표논문집 제7권 제2호

슈퍼컴퓨터상에서 광선추적 알고리즘의 병렬화에 대한 성능분석

이효종*, 강줄기*
*전북대학교 전자공학과
e-mail:hlee(jkkang)@sel.chonbuk.ac.kr

A Performance Analysis of the Parallel Ray Tracing Algorithm on a Supercomputer

Hyo Jong Lee*, Jul-Ki Kang*
*Dept of Electronic Engineering, Chonbuk National University

요 약

컴퓨터를 활용하여 사진영상을 얻는 기술은 여러 분야에서 많은 응용이 이루어지고 있는데, 그 중에서도 광선추적기법은 가장 많이 쓰이면서도 현실감 있는 사진영상을 얻는 음영처리 기법중의 하나이다. 하지만 영상이 복잡해짐에 따라 컴퓨터로 처리하는 시간도 그만큼 많이 소요되는데 슈퍼컴퓨터상에서 병렬처리기법을 적용함으로써 처리시간을 상당히 줄일 수가 있다. 본 논문에서는 IBM RS/6000 SP 슈퍼컴퓨터를 활용하여 순차적 광선추적 알고리즘을 메시지 교환방식을 통한 병렬처리 기법으로 성능분석을 하고자 하였다. 실험을 위해서 슈퍼컴퓨터의 노드수를 최대 16개까지 증가시켜가면서 복잡한 영상에 대해 병렬 광선추적 알고리즘의 성능분석을 하였고, 메시지 교환방식 중에서 블락킹 통신과 비블락킹 통신에 대해서 그 성능을 각각 비교하였다.

1. 서론

일반적으로 실생활에서 접하는 물체는 빛에 의해 반사되는 빛의 명암에 의해서 인식된다. 즉, 여러 가지 형태의 빛이 자연에 존재하고 이러한 빛들이 물체에 반사, 굴절, 투과되어 우리 눈에 들어오는 것을 눈에서 감지하여 물체를 인식하게 된다. 따라서, 모든 빛의 경로를 추적하여 우리 눈에 들어오는 모든 빛을 수집하면 사진영상과 같이 정밀하게 표현된 영상을 얻을 수 있다. 이와 같이 다양한 형태의 물체가 조명을 받고 있을 때 그 복합영상을 표현하는 방법은 오래 전부터 컴퓨터 그래픽스분야에서 연구되어 왔다.

물체에 직접 또는 간접적으로 영향을 미치는 빛들이 반사나 굴절되는 경로를 모델화하여 빛의 영향을 체계적으로 계산하는 광선추적 알고리즘은 일반적으로 오랜 컴퓨터 처리시간을 필요로 한다. 이러한 문제를 보다 쉽게 해결하고자하는 여러 가지의 개량된 광선추적표현 알고리즘이 학계에 발표되었다[1,2].

또한 광원에서 직접 투사되는 광선의 이동경로들은 독립적으로 계산될 수 있기 때문에 빛의 모델은 쉽게 병렬화 될 수 있다는 점에 착안하여 병렬프로세서 시스템에 의하여 컴퓨터 처리시간을 단축시키고자 하는 연구결과들도 발표되었다[3].

본 논문에서는 일반적으로 오랜 컴퓨터 처리시간을 요구하는 순차적 광선추적 기법을 병렬화한 병렬 광선추적 알고리즘을 슈퍼컴퓨터와 병렬프로그래밍 개발도구인 MPI(Message Passing Interface)를 적용하여 실험하였다. 병렬 알고리즘에서는 이미지의 크기와 프로세서의 수와는 독립적으로 실행될 수 있는 병렬성을 적용하여 복잡한 영상에 대해서 노드수를 변화시켰고, MPI 함수중 블락킹 방식과 비블락킹 방식에 대해서 각각 실험을 하였다.

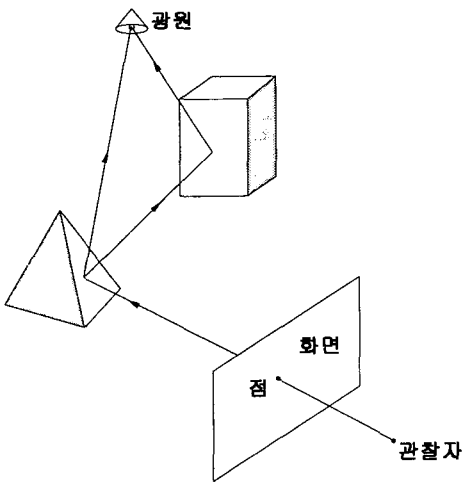
본 논문의 구성은 제 2절에서는 순차적 광선추적 기법중 정방향 광선추적법(Forward Ray Tracing)과 역방향 광선추적법(Backward Ray Tracing)에 대해서 비교하였다. 제 3절에서는 MPI 병렬프로그래밍

에 대해서 설명을, 제 4절, 5절에서는 실험결과 및 결론에 대해서 설명하였다.

2. 광선추적 알고리즘

(1) 순차적 알고리즘

광선추적 알고리즘은 시각 시스템으로 인식할 수 있는 모든 빛을 추적하는 방법으로 보통 정방향 광선추적법과 역방향 광선추적법으로 이루어진다. 정방향 광선추적법에서는 광원의 입장에서 광선을 추적하여 나가기 때문에 자연에서 일어나는 현상을 정확하게 모델링하고 있으나 반사되는 빛들의 대부분이 산란되는 경우 컴퓨터 계산상의 엄청난 낭비를 가져오게 된다. 역방향 광선추적법은 이러한 문제를 극복하여 속도향상을 위한 광선추적 알고리즘에 적용되고 있다. 관찰자의 눈과 시야에 들어오는 화면, 그리고 실제 물체들이 조명을 받고 있는 상황에서 빛을 역방향 추적하는 방법은 다음과 같이 설명될 수 있다. 추적하고자 하는 광선을 관찰자의 눈에서 출발하여 화면의 한 점을 통과하여 지나가게 된다. 이 빛이 실제 물체들과 부딪히는지를 계산하고, 부딪히는 경우 그 물체의 특성에 따라 반사, 굴절, 투과현상을 시뮬레이션하고 각각의 경우에 합당한 빛의 강도를 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

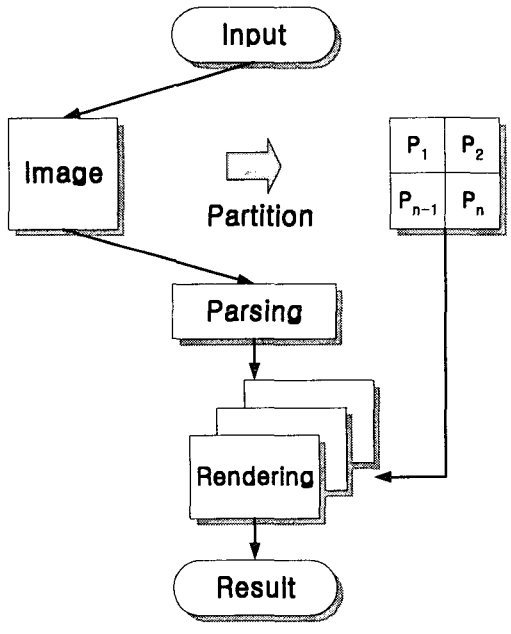


(그림 1) 광선추적에서의 광선경로

(그림 1)은 광선추적 알고리즘에서 빛을 역추적하는 간단한 광선의 경로를 보여주고 있다. 이를 이용한 대표적인 순차적 광선추적 알고리즘은 Pov-Ray[4]와 Rayshade[5]등에서 구현되었다.

(2) 병렬화된 알고리즘

광선추적 기법에서 시간이 가장 많이 소요되는 부분은 빛과 물체가 부딪히는 점을 찾는 부분이다. 이것을 위해서는 추적하는 광선마다 화면에 있는 모든 물체와의 부딪히는 점을 비교해야 한다. 특히 작은 물체들이 모여서 이루어진 큰 물체를 비교할 때는 많은 시간이 걸리는 게 보통인데 보통 Bounding Slab의 방법을 통해서 부딪히는 점의 비교횟수를 줄일 수 있고, 결과적으로 전체적인 속도를 향상시킬 수 있다. Bounding Slab방법은 비슷한 위치에 있는 복잡한 물체를 정사각형의 물체로 재정의 하고 이 정사각형을 먼저 검사한 후 부딪힌다고 판단될 때 내부의 물체에 대한 부딪히는 점을 다시 검사하는 방법이다. 그리고 광선추적 알고리즘을 MPI 개발도구를 적용해서 성능을 향상시킨다는 것은 광선추적 알고리즘 자체와는 상관이 없으므로 이미 최적화된 순차적 광선추적 알고리즘을 사용해서 효율적인 병렬화를 하는것이 주된 문제이다. 따라서 순차적 알고리즘을 병렬화하기 위해서는 작업을 어떻게 분할하는가하는 작업의 분할방법도 고려해야 한다.



(그림 2) 병렬 광선추적 알고리즘

광선추적 방법은 화면의 다른 점들간에는 직접적인 관련성이 없으므로 모든 점들을 독립적으로 계산하여 병렬성을 살릴 수 있는 특징이 있다. 이론적으로 모든 점들을 각각 한 개의 프로세서에 할당을 하여 계산을 할 수 있지만 실제 응용에서는 한정된 프로세서만을 사용하여야 한다. 작업의 분할방법으로 (그림 2)와 같이 전체의 이미지를 바둑판과 같이 일정한 블락으로 나누어서 여러 프로세서에 분배하고 그 결과를 조합하는 방법을 사용하였다[6].

3. MPI 병렬 프로그래밍

병렬처리기술에서 메시지 전달을 기반으로 하는 소프트웨어는 MPI와 PVM(Parallel Virtual Machine)이 있다. 현재까지 PVM을 적용해서 병렬 광선추적 알고리즘을 구현한 사례는 많이 보고 되고있다. 그런데 두 병렬처리 개발도구를 비교해 볼 때 MPI는 프로세스내 메시지 송수신 함수들이 직접 메시지를 송수신하는데 비하여 PVM의 경우는 사용자 프로세스가 자신의 컴퓨터상에 떠있는 데몬 프로세스를 경유하여 메시지를 송수신 하여야 하므로 MPI가 PVM보다 성능이 더 우수한것으로 알려져 있다[7]. 그래서 본 논문에서는 MPI 도구를 순차적 광선추적 기법의 병렬화에 적용하였다.

병렬 광선추적 알고리즘에서는 할당된 노드가 최소 2개 이상이 필요한데, 마스터 노드는 입력정보를 해석하고, 입력이미지를 각 프로세서에 나눠주기 위해서 분할을 하고, 각각의 노드들이 담당할 작업을 다른 슬레이브 노드에 전달하는 역할을 하며 마지막으로 나머지 슬레이브 노드들이 렌더링한 계산결과를 종합하는 역할을 한다. 마스터 노드가 해석된 입력정보를 다른 슬레이브 노드들에게 나눠줄 때는 Blocking 통신방식에서는 표준모드 MPI 함수인 MPI_Send()/MPI_Recv() 함수를 사용하고, Non-Blocking 통신방식에서는 MPI_Isend()/MPI_Irecv() 함수를 사용하여 메시지를 주고 받게 된다. 정보를 나눠준 후에 각각의 슬레이브 노드들은 각 프레임들을 렌더링하게 된다. 각각의 슬레이브 노드에 할당할 일을 최대한 균등하게 위해서는 전체 이미지를 바둑판과 같이 일정한 블락으로 분할하는 방식을 택했다. 각각의 이미지 블락은 다른 슬레이브 노드에 의해서 계산이 이루어지는데, 예를 들면 10개의 노드가 존재한다고 가정하면 각 노드들은 자신에게 할당된 이미지 블락을 렌더링하게 된다. 최종적으로 슬레이브 노드에서 계산된 렌더링 결과는 마스터 노드에서 결과를 종합하기 위해서 다시 마스터 노드에 전송되어진다.

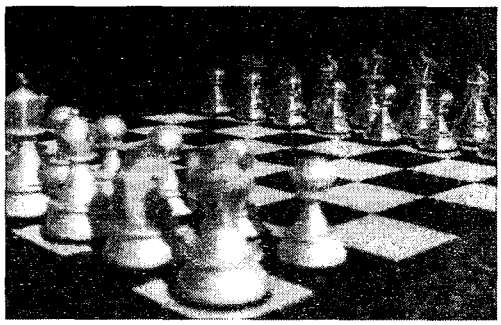
4. 실험 결과

본 논문에서 병렬 광선추적 알고리즘의 성능분석을 위해서는 <표 1>과 같은 환경을 사용하였다.

<표 1> 슈퍼컴퓨터의 실험환경

기종	IBM RS/6000 SP 9076-550
CPU	RISC 방식 200 Mhz 64bit Power3
Node	16 개
네트워크 대역폭	150 MB

실험결과를 얻기 위해서 최소 1 노드에서 최대 16 노드까지에서 실험을 통해서 얻어진 결과를 비교 분석하였는데, 실험결과의 뚜렷한 비교분석을 위해서 비교적 시간이 많이 걸리는 이미지를 생성하도록 하였다. 본 논문에서 입력 데이타로 사용된 이미지는 POV-RAY 벤취마크 웹사이트에서 벤취마크용으로 사용된 체스판으로 총 471개의 폴리간으로 구성되어 있다. 실험에서 최종적으로 얻어지는 이미지는 640×480의 해상도로 (그림 3) 과 같다.



(그림 3) 실험을 통해서 얻어진 이미지 (640×480)

<표 2>는 실험결과를 보여주고 있는데, 노드가 1개인 경우는 순차적 광선추적 알고리즘과 병렬 광선추적 알고리즘의 성능 비교를 위해 병렬화되지 않은 순차적 광선추적 알고리즘을 실험한 결과 값을 나타내었다. 각각의 실험은 노드를 4, 8, 16개를 사용하였고,

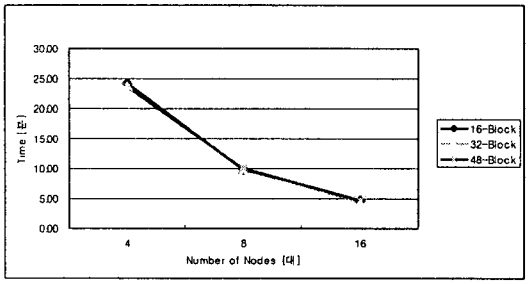
<표 2> 병렬 광선추적 알고리즘의 결과 (단위: 분)

Node (개)	Blocking			Non-Blocking		
	16	32	48	16	32	48
1	121.13			121.13		
4	24.20	23.65	23.48	23.82	23.65	23.47
8	9.90	9.97	9.97	10.02	10.13	9.97
16	4.72	4.35	4.67	4.85	4.50	4.73

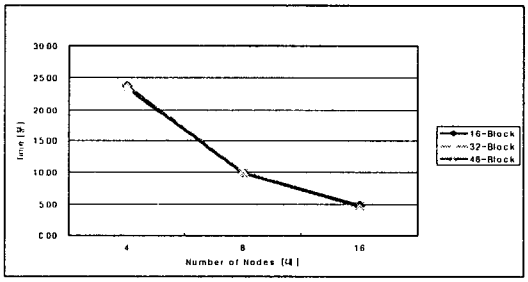
이미지를 분할하는 방법에서 블락크기(가로*세로)를 16, 32, 48로 변경하면서 실험을 하였다. 블락크기는

메시지 크기와 관련이 있다. 그리고 MPI의 통신방식인 Blocking, Non-Blocking방식에 대해서 성능분석을 하였다. <표 2>에서 보는바와 같이 병렬처리를 하였을 때 최대 28배의 성능향상이 있었다. 블락크기와 비교해서 대부분 블락크기가 32, 48일때 가장 좋게 나왔다. 하지만 블락을 너무 크게 잡아주면 메모리 문제와 통신간에 블락킹을 유발할 수 있기 때문에 너무 큰 값은 피했다.

(그림 4), (그림 5)는 노드수를 증가시면서 성능분석을 한 것으로서 꾸준한 성능향상을 엿볼 수 있다.

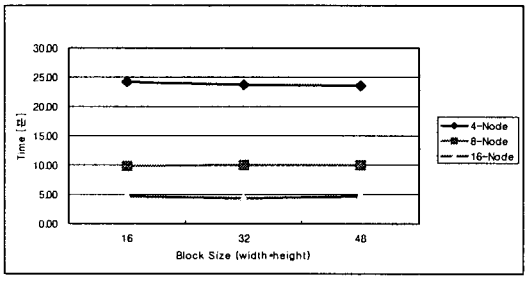


(그림 4) Blocking Communication

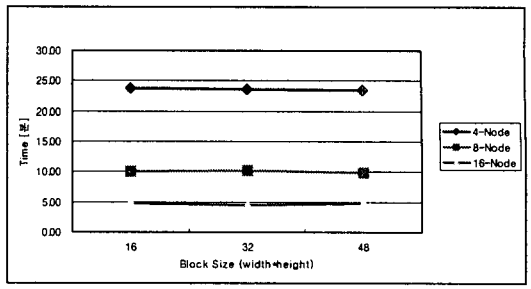


(그림 5) Non-Blocking Communication

(그림 6), (그림 7)은 블락크기와 관련하여 성능분석을 한 것이다. 일반적으로 Blocking 통신방법에서 성능이 약간 좋게 나왔고, 이미지의 블락크기가 32, 48일 때 성능향상이 비교적 좋게 나왔다.



(그림 6) Blocking Communication



(그림 7) Non-Blocking Communication

5. 결론

본 논문에서는 순차적 광선추적 알고리즘에 MPI 개발도구를 활용해서 메시지교환 기법을 이용한 병렬처리기술을 적용하여 병렬 광선추적 알고리즘을 구현하였고, 이를 슈퍼컴퓨터에서 성능을 분석하였다. 순차적 광선추적 알고리즘에 비해서 병렬화된 알고리즘에서는 최대 28배의 성능향상을 얻어낼 수 있었다. 노드수가 16개까지 증가함에 따라서 꾸준한 성능향상을 볼 수 있었고, 메시지 크기에 따라서 오버헤드가 발생하는걸 볼 수 있었다. 결론적으로 MPI는 지금도 표준화가 계속 진행중인 병렬프로그래밍 도구로서 본 논문에서와 같이 앞으로 많은 병렬 소프트웨어 개발에 적용되어 훌륭한 결과를 얻을 수 있을 것이다.

참고문헌

[1] Glassner, A. S. "Space Subdivision for Fast Ray Tracing", IEEE Computer Graphics and Applications, vol.4, no 10. pp 15-22, Oct, 1984

[2] Arvo, J. and Kirk, D. "Fast Ray Tracing by Ray Classification", ACM Computer Graphics, vol. 21, no. 4, pp. 55-64, Jul. 1987

[3] Cook, R. L., Porter, T., and Carpenter, L., "Distributed Ray Tracing", ACM Computer Graphics, vol. 18, no. 3, pp. 137-145, Jul. 1984

[4] POV-Ray Team, "Persistence of Vision Ray Tracer (Pov-Ray) Version 2.0 User's Documentation", Private Publication, 1993

[5] Craig E. Kolb, "Rayshade User's Guide and Reference Manual", Draft 0.1, Jan 15, 1991

[6] 임훈철 ,"트랜스퓨터 프로그래밍 도구와 응용프로그램의 개발", 졸업논문집, pp. 71-92. February 1996

[7] 조승호, "병렬컴퓨터상에서 MPI와 PVM의 성능비교에 관한 연구", 한국정보과학회 가을학술발표논문집, Vol. 23, No. 2, pp. 1631-1634, October 1996