레디언스 엔진을 위한 CORBA 기반 클러스터링 시스템

최정호° 차현철 김정선
한양대학교 컴퓨터공학과
{choijh°, hccha, jskim}@cse.hanyang.ac.kr

Standard CORBA based Clustering System for Radiance Engines

JeongHo Choi° Hyunchel Cha Jungsun Kim
Dept. of Computer Science and Engineering, Hanyang University

요 약

레디언스 엔진(Radiance Engine)은 미국 Lawrence Berkeley Laboratory(LBL)에서 개발된 빛의 거동을 물리적으로 시뮬레이션한 결과로부터 조도 및 휘도 분포를 계산하고 가시화 하여 빛 환경의 정량적, 정성적 평가가 모두 가능한 전 세계적으로 수많은 사용자 그룹이 형성되어 있는 프로그램이다. 그러나 많은 사용자 층을 보유하고 있음에도 불구하고 텍스트 기반의 인터페이스와 UNIX 환경의 워크스테이션급 컴퓨터에서만 실행되는 한계를 가지고 있기 때문에 그래픽 기반의 인터페이스에 익숙해져 있는 사용자들이 이용하기 어렵고, 비용적으로 고성능 컴퓨터에 대한 많은 부담이 있어 대중화가 되고 있지 못하는 실정이다.

본 논문에서는 아직까지 그 우수성에도 불구하고 불편한 인터페이스와 고성능의 서버를 필요로 하는 이유로 국내에서 활발하게 사용되고 있지 못하는 레디언스 엔진을 다수의 사용자가 쉽고 빠르며 간편하게 이용 할 수 있도록 각각의 사용자자가 제출한 레디언스 엔진 수행 작업을 분산하여 처리할 수 있는 클러스터링 시스템을 설계하고 구현한다.

1. 서 론

인공조명의 경우, 인공광원을 축소하여 재현하는 데에 어려움이 따르므로 주로 컴퓨터 프로그램을 이용하여 빛 환경의 정량적, 정성적인 평가를 내린다. 이에 사용되는 대표적인 프로그램은 미국 Lawrence Berkeley Laboratory(LBL)에서 개발된 레디언스 엔진(Radiance Engine)으로 빛의 거동을 물리적으로 시뮬레이션한 결과로부터 조도 및 휘도 분포를 계산하고 가시화하여 빛 환경의 정량적, 정성적 평가가 모두 가능한 장점을 가지고 있다. 또 전 세계적으로 많은 사용자 그룹이 형성되어 있고, 프로그램의 소스코드 자체가 무료로 배포되고 있어서 조명공학분야의 연구도구로서의 가치가 큰 것으로 평가되고 있다.

그러나, 텍스트 기반의 인터페이스와 UNIX환경의 워크스테이션급 컴퓨터에서만 실행이 가능한 한계를 가지고 있기 때문에 대부분의 사용자들이 사용하기 어렵고 비용적으로 많은 부담이 있어 대중화가 되고 있지 못하는 실정이다.

그래서 본 논문에서는 아직까지 그 우수성에도 불구하고 불편한 인터페이스와 고성능의 서버가 필요하다는 이유로 국내에서 폭넓게 사용되지 못하는 레디언스 엔진을 다수의 사용자가 손쉽고 빠르게 사용할 수 있도록 사용자 각각의 레디언스 엔진 수행 작업을 분산하여 처리할 수 있는 레디언스 엔진을 위한 CORBA 기반 클러스터링 시스템을 설계하고 구현하고자 한다. 2장에서는 관련 연구로써 레디언스 엔진의 개요와 클러스터링 시스템의 정의 및 특징, 미들웨어인 CORBA에 대해서 기술하고, 3장에서는 레디언스 엔진을 위한 CORBA 기반 클러스터링 시스템을 설계 및 구현하며, 마지막으로 4장에서 구현에 대한 결과를 정리하며 결론을 맺는다.

2. 관련 연구

2.1 레디언스 엔진(Radiance Engine)

레디언스 엔진은 미국 국립 Lawrence Berkeley Laboratory(LBL)의 조명 연구팀이 개발한 프로그램으로서 1987년 이래로 버클리의 캘리포니아 대학 건축공학과에서 사용되어 오고 있는 조명 랜더링 및 시뮬레이션 프로그램이다. 레디언스 엔진은 UNIX환경의 워크스테이션급 컴퓨터에서 실행되며, C언어로 컴파일 되어있다.

레디언스 엔진은 무료로 배포되고 있으며, 전 세계적으로 인터넷을 이용한 사용자 그룹이 형성되어 있어 다양하게 사용상의 문제에 대한 토론, 정보와 자료의 교환이 이루어지고 있다. 이러한 상황으로 미루어 보아, 앞으로 레디언스 엔진은 조명관련 연구개발 그룹과 고도의 설계를 수행하는 그룹들에게 표준으로 될 가능성이 크다고 할 수 있다.

레디언스 엔진은 역광선추적기법을 기초로 한다. 이것은 광선이 자연적으로 진행하는 방향의 반대 방향으로 추적하여, 실제 광선이 발생한 광원의 활동을 예측하는 것을 의미한다. 그 과정은 눈으로부터 시작하여 공간의 대상물들의 표면들을 따라 모든 물리적 상호작용을 계산하여 광원까지 광선을 쫓아가게 된다[1]. 이러한 개념은 몬테카를로 방법과 광선추적기법에 바탕을 두고 있다. 실제 공간에서 광원으로부터 반사된 광선은 결국 흡수될 때까지 반사, 투과되는데 우리가 실제로 보게되는 장면은 이러한 광선들이 망막에 영상을 만든 것이다. 이처럼 망막에 영상이 맺히게 되는 것은 공간 내에 존재하는 전체 광선중 일부분만이 재실자의 눈에 들어옴으로써 이루어지는 것이므로, 눈으로부터 주변환경을 거쳐 광원으로까지의 광선을 역으로 추적함으로써, 광원으로부터 나온 광선들의 거동을 확인하여 빛 환경을 가시화 할 수 있다

2.2 클러스터링 시스템

클러스터링(Clustering)이란 여러 대의 서버를 연결하여 가변적인 업무부하를 처리하거나, 특정 서버에 문제가 발생했을 경우에도 운영이 계속되도록 하는 것을 말한다. 즉, 둘 이상의 컴퓨터를 마치 하나의 컴퓨터처럼 운용할 수 있도록 연결하는 것으로, 병렬처리 또는 부하, 배분, 고장 등의 사태에 효율적으로 대처하기 위한 방법이다. 일반적으로 클러스터링(Clustering)의 개념이 서버컴퓨터에 한정되는 경향이 있으나, 클러스터링(Clustering)은 PC나 워크스테이션 등 모든 기종의 컴퓨터를 사용할 수 있다. 클러스터링(Clustering)의 특징으로는 프로세서나 네트워크 장비, 그리고 저장장치 등으로 구성되고, 주로 오픈소스 기반의 운영체계와 소프트웨어를 사용하며, 가격대 성능면에서 기존의 대형 컴퓨터를 능가하고, 업그레이드와 확장성이 우수하다는 것이다. 특징으로는 확장성, 높은 가용성, 작업 부하의 균형적인 밸런싱, 관리기능 등을 들 수 있다.

2.3 CORBA의 구조

레디언스 엔진을 위한 클러스터링 시스템은 미들웨어로서 분산객체 컴퓨팅의 표준인 CORBA(Common Object Request Broker Architecture)를 사용하였다. CORBA는 개발언어나 네트워크, 운영체제 등 이기종 시스템간의 상호 연동을 가능하게 하는 미들웨어로서, 약 800여개 이상의 컴퓨터 관련 단체들이 참여한 OMG(Object Management Group)에 의해 표준화 되었다[2].

CORBA는 컴퓨터 내부의 버스처럼 서로 다른 프로그램들 사이의 Bus 역할을 하는 모듈로서 각기 다른 언어로 구현되고, 또 분산되어 있는 모듈들에 대해 마치 로컬에 있는 것처럼 사용할 수 있도록 하는 기능을 제공해 준다.

최근 CORBA 3.0의 구조는 그림 1과 같다.

클라이언트에서 서버(구현 객체)로의 요청을 중개하는 것이 ORB(Object Request Broker)의 역할이며, ORB는 ORB core, Object Adapter, ORB Interface, Static IDL stub, Dynamic Invocation Interface(DII), Static IDL Skeleton, Interface Repository 및 Implementation Repository로 구성된다.[2]

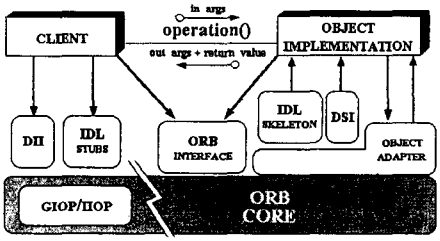

그림 1. CORBA 3.0의 구성

본 클러스터링 시스템은 CORBA 3.0을 만족하는 C++로 구현된 ACE 기반의 CORBA ORB인 TAO를 이용하여 구현하였다[3][4].

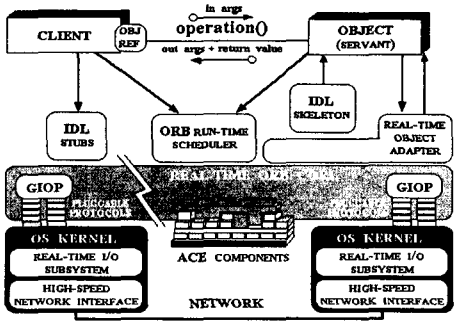

그림 2. TAO의 구성

3. 레디언스 엔진을 위한 클러스터링 시스템의 설계 및 구현

3.1 클러스터링 시스템 동작 시나리오

- 사용자들은 웹서버에 레디언스 엔진을 실행하는데 필요한 데이터 파일을 업로드 하고, 옵션을 선택함으로써 Job을 생성한다.
- 사용자가 생성한 Job은 Execution Job Manager에서 Job 단위로 관리 되며, 우선 순위에 따라 로드 밸런서의 로드 매니저를 통해 유휴 상태의 Cluster Node로 분배 되어 진다. 로드밸런서는 Job 분배시 유휴상태의 Cluster Node가 없을 경우에는 Cluster Node가 유휴상태가 될 때까지 대기한다.
- Cluster Node는 작업을 완료한 후 결과 파일을 Facade Job Manager로 전송한다.
- Facade Job Manager는 사용자에게 이메일을 통해 작업이 완료 되었음을 통보하고, 사용자는 웹서버를 통해 결과를 확인 할 수 있다.

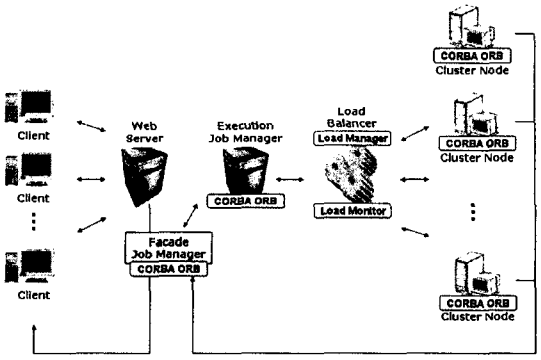

그림 3. 클러스터링 시스템의 구조

3.2 클러스터링 시스템 전체구조

레디언스 엔진을 위한 클러스터링 시스템은 그림 4와 같이 Facade Job Manager, Execution Job Manager, Load Balancer, Cluster Node의 네 부분으로 나누어진다.

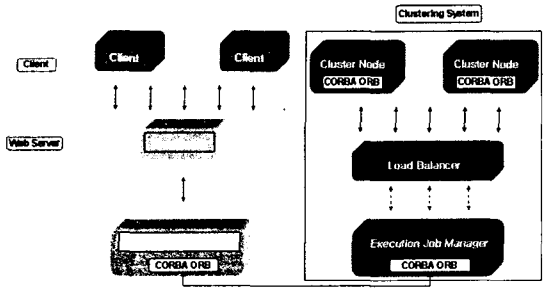

그림 4. 클러스터링 시스템의 전체 구성 요소

3.3 Facade Job Manager

Job Generator와 Completion Job Manager로 구성된다. Job Generator는 웹서버(Web Server)들로부터 제출된 Job에 대한 Job의 ID를 생성 및 우선순위를 지정하며, 레디언스 엔진을 실행하는데 필요한 데이터 통합한 후, 이를 Execution Job Manager 에게 제출한다.

Completion Job Manager는 CORBA의 이벤트 서비스의 이벤트 채널을 통해 전송된 레디언스 엔진의 실행결과를 사용자에게 E-mail로 통보한다.

3.4 Excution Job Manager

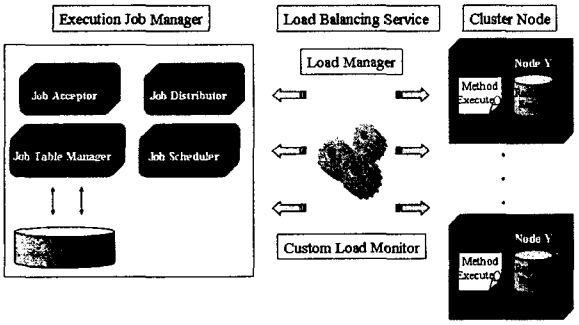

그림5. 클러스터링 시스템의 세부 구성 요소

Execution Job Manager는 레디언스 엔진을 실행할 수 있는 Cluster Node에게 Job을 할당하는 역할을 한다. 세부 구성 요소를 살펴보면 Job Acceptor는 Facade Job Manager로부터 제출된 Job을 Job Table Manager를 통해 Job 테이블에 저장 한다. Job Table Manager는 Job의 저장 및 삭제, 처리상태 등 Job 테이블을 관리한다.

Job Scheduler는 Job 테이블로부터 우선 순위에 따라 분배할 Job을 선택하며, Job Distributor는 Job Scheduler에 의해 선택된 Job을 로드 밸런싱 서비스를 사용하여 클러스터 노드로 분배하는 역할을 한다.

3.5 Load Balancing Service

로드 밸런싱 서비스 정책은 Customize된 Least- Loaded 방식을 사용한다. Customize된 Least-Loaded 방식이란 모든 Cluster Node의 부하가 설정된 값보다 큰 경우 Job Distributor는 임의의 한 클러스터·노드가 작업을 완료할 때까지 Job의 분배를 중단하고 대기하는 방식이다.

즉, 유휴 상태의 Cluster Node가 존재할 경우에만 Job을 분배 하는 방식이다.[5][6][7]

3.6 Cluster Node

레디언스 엔진을 실행할 수 있는 모듈을 가지고 있으며 레디언스 엔진을 실행한다. 레디언스 엔진이 수행되고 나면 그 결과를 CORBA의 이벤트 서비스를 이용하여 Faced Job Manager의 Completion Job Handler로 전송한다.

4. 결론 및 향후 연구 과제

본 논문에서는 조명공학과 관련하여 연구도구로 가치가 충분함에도 텍스트 기반의 인터페이스와 UNIX환경의 워크스테이션급 컴퓨터에서만 실행이 가능한 레디언스 엔진을 다수의 사용자가 사용하기 쉽고, 빠르게 사용할 수 있도록 하기 위해서 사용자 각각의 레디언스 엔진 실행 작업을 분산하여 처리할 수 있는 레디언스 엔진을 위한 CORBA 기반 클러스터링 시스템을 설계하고 구현하였다.

향후 연구 과제로서 기존의 시스템에 클러스터링 시스템의 상태를 효율적으로 관리하기 위한 모니터링 도구와 이상 발생시 자동으로 클러스터링 시스템을 복구할 수 있는 기능을 추가한다면 더 효율적인 클러스터링 시스템이 될 수 있을 것으로 기대된다.

참고문헌

[1]Ward, G. J, "The RADIANCE Lighting Simulation and Rendering System", Computer Graphics, Proceedings of '94 SIGGRAPH conference

[2]www.omg.org

[3]"http://www.cs.wustl.edu/~schmidt/"

[4]Douglas C. Schmit, David L. Levine, and Sumedh, "The Design and Performance of Real-time Object Request Brokers", Computer Communications, 21(4):294-324, April 1998

[5]"Proposed CORBA Load Balancing and Monitoring Specification", Object Manager Group, OMG Document mars/02-10-14 edition, October 2002

[6] Ossama Othman, Carlos O'Ryan, and Douglas C. Schmidt, "Designing an Adaptive CORBA Load Balancing Service Using TAO", IEEE Distributed Systems Online, 2(4), April 2001

[7]Ossama Othman, Douglas C. Schmidt, "The Design and Performance of a Scalable CORBA Load Balancing and Monitoring Service", 2002