제36회 한국정보처리학회 추계학술발표대회 논문집 제18권 제2호 (2011. 11)

거대 데이터의 실시간 가시화를 위한 분산 가시화 서버의 설계 및 구현

이중연, 김민아, 허영주
한국과학기술정보연구원 슈퍼컴퓨팅본부
jylee@kisti.re.kr

Design and Implementation of Distributed Visualization Server for Real-time Visualization of Massive Dataset

Joong-Youn Lee, MinAh Kim, Youngju Hur
Supercomputing Center, KISTI

요 약

일반 PC의 메인 메모리에 올릴 수 없는 거대 용량의 데이터의 경우 가시화를 통한 해석을 수행하는 데 어려움이 많다. 본 논문에서는 이러한 거대 용량의 데이터를 실시간으로 처리하기 위한 분산 환경에서의 가시화 서버의 설계를 제안한다. 본 논문에서 제안하는 가시화 서버는 가시화 관리자, 네트워크 관리자, 데이터 관리자로 구분되며 이들 관리자를 통해 복수의 사용자에 대한 가시화 서비스 제공, 거대 데이터의 실시간 동적 데이터 분할 및 할당 및 실시간 가시화를 지원한다.

1. 서론

컴퓨터 시뮬레이션은 물리, 화학, 생물, 기계공학 등 자연과학·공학뿐만 아니라 인문·사회과학 분야에서도 많이 적용되고 있는 연구방법으로 계산(computation)과 가시화(visualization)라는 두 가지 기술에 의해 발전되고 있다. 고성능 컴퓨터(HPC: High Performance Computer)를 포함한 계산 기술의 빠른 발전은 기존에 적용하기 어려웠던 문제를 빠른 시간 내에 해결하는 것을 가능하게 하였으며 새로운 현상의 발견도 가능하게 함으로써 컴퓨터를 이용한 시뮬레이션이 널리 사용되고 있다. 시뮬레이션 결과는 내용을 그대로 보면서 해석하는 것이 매우 어렵기 때문에, 쉽게 알아볼 수 있도록 데이터에 그래픽처리를 하여 보여주는 기술을 과학적 가시화 기술이라고 한다. 그러나 기하급수적으로 발전하는 계산 기술에 비하여 가시화 기술의 발전은 이를 따라가지 못하고 있으며 시뮬레이션 연구자가 워크스테이션이나 데스크탑을 이용한 기존의 방식으로는 대용량의 복잡한 결과 데이터를 이해하기가 어려워지고 있다. 한편, 시뮬레이션의 결과 데이터를 가시화하는 방법은 매우 다양하고 때로는 복잡한 가시화 알고리즘을 필요로 하기도 한다. 이런 복잡한 알고리즘을 지원하는 툴은 매우 다양한데, 이중 특히 VTK(Visualization Tool Kit)[1,2]는 과학 데이터 가시화에 많이 사용되는 공개 소프트웨어 라이브러리로, 매우 다양한 자료구조와 가시화 관련 알고리즘 기능을 지원하며, 실제적으로 CFD, 의료, 화학, 구조역학 등 다양한 분야에서 유용하게 사용되고 있다. 그러나 VTK는 일반적으로 병렬 또는 분산 처리에 매우 제약적이며 병렬/분산 처리를 위한 데이터 분할 기법 역시 정적 분할 기법을 사용하는 등 매우 기초적인 병렬/분산 처리만을 지원한다. 본 논문에서는 이러한 VTK의 기능을 보완하여 일반 PC의 메인 메모리에 올릴 수 없는 거대 용량의 데이터를 빠르게 처리할 수 있도록 하는 분산 가시화 서버를 제안하고 이의 실제 구현 사례를 소개하고자 한다.

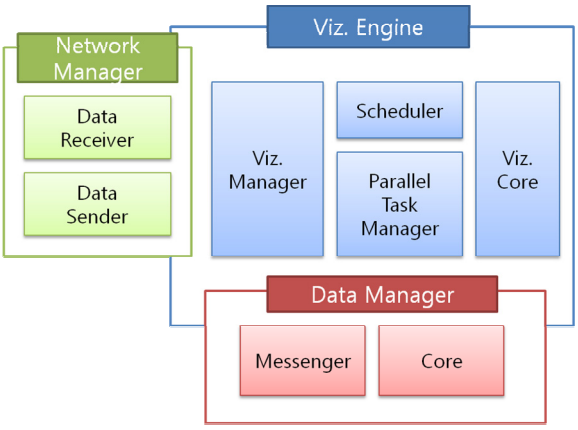
2. 분산 가시화 시스템의 설계

본 논문에서 제안하고자 하는 분산 가시화 서버는 다음과 같은 특징의 구현을 목표로 한다.

1. 일반 PC에서 실시간 처리가 불가능한 거대 용량의 데이터의 실시간 가시화
2. 두 대 이상의 복수의 클라이언트 연결이 가능
3. 클라이언트 요청에 따라 최적의 작업 프로세스를 유연하게 할당, 해제
4. 가시화 작업에 최적화된 실시간 동적 데이터 분할 및 할당

여기서 2번은 하나의 서버를 여러 클라이언트가 공유함으로써 자원을 효율적으로 사용할 수 있게 한다. 이는 3번의 경우도 마찬가지인데, 동시에 여러 클라이언트의 요청을 처리해야 하기 때문에 서버의 모든 자원을 한 클라

이언트를 위해 사용하지 못한다. 따라서 각 클라이언트의 요청을 분석하여 각 요청을 처리하는데 필요한 최적의 프로세스 개수를 판단한 뒤 작업을 할당하도록 한다. 마지막으로 4번의 경우 가시화 작업의 성격과 양에 따라 각 가시화 프로세스가 필요로 하는 데이터를 데이터 관리자가 동적으로 공급할 수 있도록 하기 위해 동적인 데이터 분할 및 할당 기능을 추가한다. 이를 위해 본 논문에서 제안하고자 하는 가시화 서버의 구조는 그림 1과 같이 가시화 엔진(visualization engine), 데이터 관리자(data manager), 네트워크 관리자(network manager)로 구성된다.



(그림 1) 가시화 서버 구조

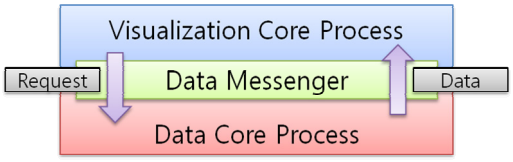
2.1. 가시화 엔진

가시화 엔진은 가시화 서버(visualization server)에서 실제 가시화 작업을 담당하는 부분으로 크게 가시화 관리자(visualization manager), 스케줄러(scheduler), 병렬 작업 관리자(parallel task manager), 가시화 코어(visualization core)로 나뉜다.

가시화 관리자는 전반적인 가시화 엔진의 작업을 관리한다. 네트워크 관리자로부터 클라이언트 요청을 받아 스케줄러와 병렬 작업 관리자로 전송하고 생성된 가시화 코어로 명령을 전달 한 뒤 결과를 네트워크 관리자로 전송하는 모든 흐름을 관리한다. 스케줄러는 가시화 요청의 종류와 데이터 크기에 따라 필요한 프로세스의 개수를 파악하고 필요한 프로세스 크기의 병렬 작업을 병렬 가시화 관리자에 요청한다. 병렬 가시화 관리자는 요청받은 개수의 프로세스를 수행할 수 있는 병렬 작업을 생성하고 이를 가시화 관리자에 돌려준다. 각 병렬 가시화 작업은 복수의 가시화 코어 프로세스로 이루어진다. 각 가시화 코어는 분산된 노드/프로세스에서 병렬 작업으로 실행되어 일반 PC에서 처리가 불가능한 거대 데이터의 처리를 가능하게 한다.

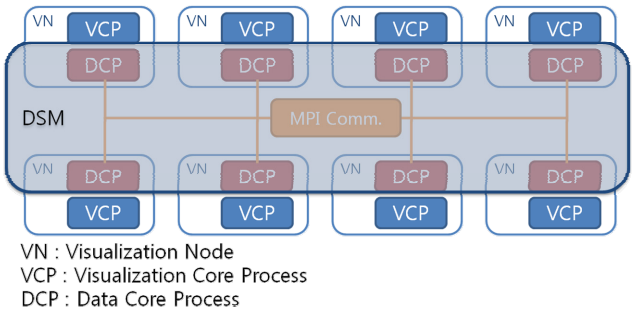
2.2. 데이터 관리자

데이터 관리자는 가시화 엔진에 데이터를 공급하는 작업을 담당하는 부분으로 크게 데이터 관리자 코어(data manager core)와 데이터 메신저(data messenger)로 나뉜다. 데이터 관리자 코어는 분산 환경에서 병렬로 실행되며 각 가시화 코어 프로세스에 데이터를 공급한다. 데이터 메신저는 서로 다른 프로세스로 실행되는 데이터 관리자 코어 프로세스와 가시화 코어 프로세스 간의 통신을 담당한다. 가시화 코어 프로세스의 요청을 받고 이 요청에 따라 데이터를 공급하는 통로가 된다.



(그림 2) 데이터 관리자 구조

각 데이터 관리자 코어 프로세스는 분산 공유 메모리(Distributed Shared Memory) 환경에서 구현되어 거대 데이터를 각 프로세스(노드)가 분할하여 저장하지만 각 프로세스는 마치 공유 메모리에 데이터가 위치한 것처럼 전역에서 필요한 데이터에 접근이 가능하다. 이를 통해 실시간으로 동적 데이터 분할이 가능하다.



(그림 3) MPI 기반의 분산 공유 메모리 환경에서의 데이터 관리자 및 가시화 코어 프로세스 실행 구조

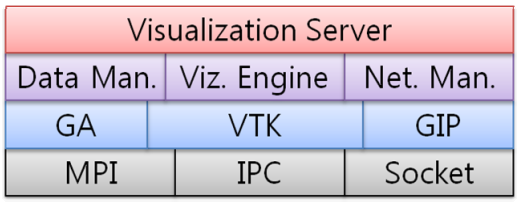
2.3. 네트워크 관리자

네트워크 관리자는 클라이언트와 서버 간의 네트워크 작업을 담당하는 부분으로 데이터 수신부(receiver)와 송신부(sender)로 나뉘며 데이터 수신부의 경우 동시에 여러 클라이언트의 처리가 가능하도록 쓰레드로 동작한다. 데이터 수신부는 미리 설정해둔 특정 포트로부터 명령을 받아 가시화 관리자에 전달한다. 이때, 데이터 통신은 동기식 통신 모드(Syncronous mode)와 비동기식 통신 모드(Asyncronous mode)로 나뉜다. 이는 클라이언트의 특성에 따라 나뉘는 것인데, 저사양의 컴퓨터에서 작동되는 씬 클라이언트(thin client)에서는 동시에 여러 작업의 처리가 어려운 만큼 동기식 모드로 동작하고 가상현실 환경이나 타일 디스플레이 환경에서의 고사양 컴퓨터에서 작동하는 클라이언트에서는 비동기식 모드로 동작한다. 동기식 모드로 작동할 경우 하나의 데이터 수신부가 동시에 여러 클라이언트의 접속을 처리하지 못하므로 여러 개의 데이터

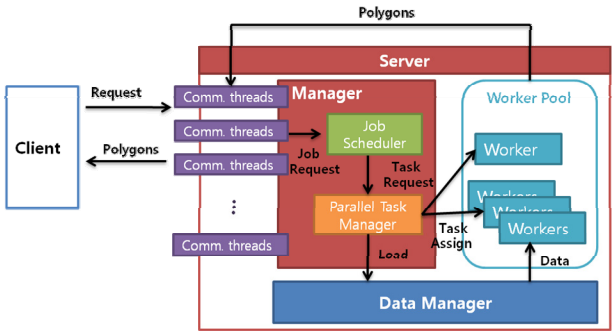
수신 쓰레드를 생성해서 처리하도록 한다.

3. 구현 방법

2장에서 소개한 가시화 서버의 구현을 위해 본 논문에서는 VTK, MPI(Message Passing Interface), GA(Global Array)[3], IPC(Inter Process Communication) 등을 이용했다. 2장에서 제안한 설계의 실제 구현 구조는 그림 5와 같다. 서버 내의 작업자(worker)는 가시화 코어를 가지고 있으며 실제 작업을 수행하는 역할을 수행한다.



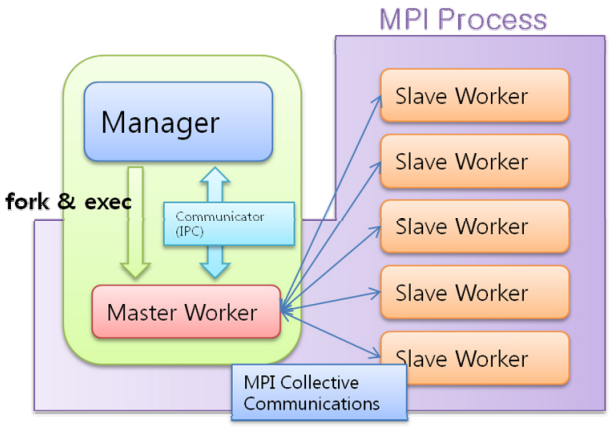
(그림 4) 가시화 서버 구현 구조



(그림 5) 가시화 서버의 실제 구현

각 구성 프로세스별 작업 순서는 다음과 같다. 먼저, 네트워크 관리자의 통신 쓰레드는 계속적으로 대기하며 클라이언트로부터 새로운 요청이 들어오는지 확인한다. 각 쓰레드는 클라이언트의 특성에 따라 동기식 또는 비동기식으로 통신을 하며 요청을 받고 응답을 한다. 이를 위해 네트워크 관리자는 동기식과 비동기식 통신을 위한 포트 번호를 별도로 할당한다.

네트워크 관리자가 받은 요청은 가시화 엔진의 관리자로 전달된다. 가시화 관리자는 스케줄러에 요청 내용을 전달하고 스케줄러는 필요한 프로세스의 개수를 판단하여 병렬 작업 관리자에 병렬 작업을 요청한다. 병렬 작업 관리자는 exec 함수를 이용하여 MPI 작업을 생성, 실행하고 해당 MPI 작업의 rank 0 프로세스의 프로세스 아이디를 가시화 관리자에 전달한다. 이때, rank 0 프로세스는 마스터 작업자(master worker)가 되고 1 이상의 rank를 가지는 프로세스는 노예 작업자(slave worker)가 된다. 가시화 관리자는 해당 프로세스 아이디를 이용 IPC 큐(queue)를 통해 가시화 작업 명령을 전송한다. 가시화 작업자가 작업이 끝난 계산을 돌려줄 때는 먼저 작업이 끝났다는 명령을 IPC 큐를 통해 전달하고 실제 데이터는 IPC 공유 메모리(shared memory)를 통해 전달한다. 가시화 관리자의 전체적인 구조는 그림 6과 같다.



(그림 6) 가시화 관리자 및 작업자 실행 및 통신 구조

데이터 관리자는 GA 기반의 MPI 작업으로 가시화 코어(작업자)와 동일 노드에서 분산 실행된다. 그림 2 및 3의 구조와 같이 각 노드의 가시화 코어 프로세스와 데이터 관리자 코어 프로세스는 데이터 관리자 메신저를 통해 명령을 주고 데이터를 받는다. 이때, 메신저 역시 IPC 큐와 공유 메모리를 이용해서 구현된다. 각 데이터 관리자의 프로세스는 MPI를 통해 통신하고 GA를 이용한 분산 공유 메모리 구조를 이용해서 전체 데이터를 효율적으로 공유하도록 한다.

4. 실험 및 결과

지금까지 설명한 가시화 서버를 이용, 다양한 크기의 로터 동역학 데이터에 대한 가시화 실험을 수행했다. 실험에 사용한 컴퓨팅 노드의 가용 메모리가 2.8TB였기 때문에 전체 데이터를 전체 컴퓨팅 노드에 분산해서 로드하는 것이 가능했다.

<표 1> 실험 데이터 크기

데이터	한 time step 크기	time step 개수	전체 크기
Fuselage	0.205GB	30	6.15GB
Coarse grid	0.336GB	90	30.27GB
Tera scale	12.64GB	98	1.21TB

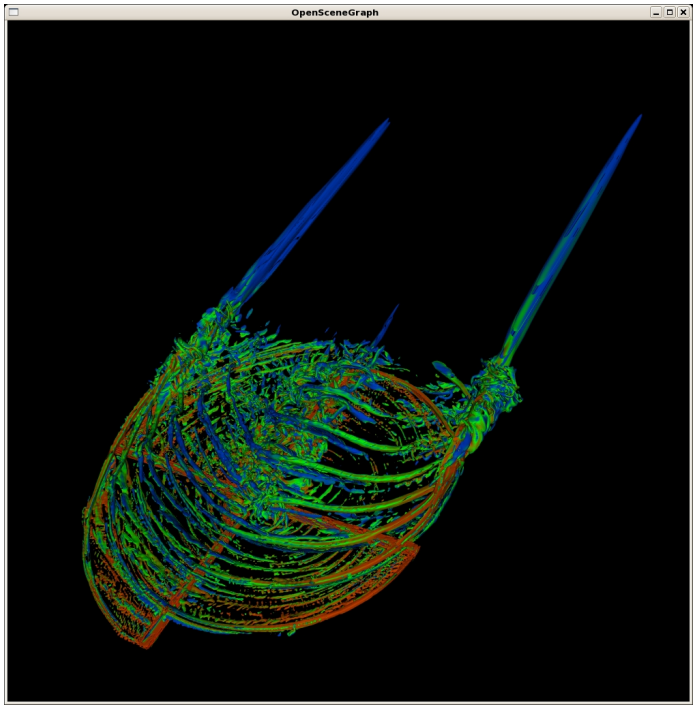
<표 2> 실험 환경

총 노드 수	90
노드 당 CPU 코어	4 (Xeon X5450 3.0GHz)
총 CPU 코어 수	360
노드 당 메모리 용량	32GB
총 메모리 용량	2880GB

<표 1>의 각 데이터에 대해 한 노드, 16노드, 전체 노드를 이용해서 Iso-surface와 cutting plane 그리고 contour line 실험을 했다. 실험 결과는 표 3에서 볼 수 있다. 가장 작은 크기인 Fuselage 데이터의 경우에는 병렬화 효과를 크게 보기 어려웠다. 90개의 전체 노드를 사용한 경우 전체 데이터 크기에 비해 과도하게 많은 노드를 사용함으로 인해 전체 적인 성능은 16 노드를 사용한 경우 보다 오히려 떨어졌다. 반면, tera scale 데이터의 경우에는 90노드를 사용한 경우 한 노드만을 사용한 경우보다 20배 빨랐으며 16 노드를 사용한 경우에 비해서도 약 2배 빠른 것을 알 수 있었다.

<표 3> 실험 결과

		Iso-surface	Cutting plane	Contour line
Fuselage				
노드 수	1	1.5265	1.2268	1.3185
	16	0.5173	0.0853	0.1010
	90	0.54593	0.0431	0.0530
Coarse grid				
노드 수	1	0.5954	0.7692	0.8733
	16	0.0448	0.0492	0.0560
	90	0.0434	0.0490	0.0670
Tera scale				
노드 수	1	20.4623	20.2432	21.698
	16	2.3001	2.3812	2.705
	90	0.9076	1.5536	1.829



(그림 7) 테라 스케일 데이터의 가시화 결과

5. 결론

본 논문에서는 분산 환경에서의 가시화 서버의 설계를 제안하고 실제 구현 사례를 소개 했다. 본 논문에서 설계하고 구현한 가시화 서버는 일반 PC에서 실시간 처리가 불가능한 거대 용량의 데이터의 실시간 가시화를 가능하게 하고 두 대 이상의 복수의 클라이언트로의 서비스가 가능하며 클라이언트 요청에 따라 작업 프로세스를 유연하게 할당, 해제하여 자원의 활용을 최적화할 수 있다. 또한 분산 가시화 작업 시에 각 작업에 최적화하여 실시간으로 데이터 분할함으로써 전체 가시화 작업의 부하를 균일하게 할 수 있었다. 또, 세 가지 종류의 로터 시뮬레이션 데이터에 적용해서 가시화 성능의 향상을 확인할 수 있었다.

참고문헌

[1] W.Schroeder, K.Martin, B.Lorensen, "The Visualization Toolkit, an Object-Oriented Approach to 3D Graphics, 4th Edition", Kitware, 2006.

[2] VTK, http://www.vtk.org

[3] J. Nieplocha et.al., "Advances, Applications and Performance of the Global Arrays Shared Memory Programming Toolkit", International Journal of High Performance Computing Applications, Vol. 20, No. 2, 203-231p, 2006.