제36회 한국정보처리학회 추계학술발표대회 논문집 제18권 제2호 (2011. 11)

대용량 데이터의 분산/병렬 가시화를 위한 응용 독립적 가시화 프로토콜

김민아*
*한국과학기술정보연구원 슈퍼컴퓨팅본부 차세대연구환경개발실

Application Independent Network Protocol for Distributed and Parallel Visualization

Min-Ah Kim*
*Supercomputing Centerm Dept. of Cyber Environment Development,
Korea Institute of Science and Technology Information

요 약

대용량 데이터의 분산/병렬 가시화를 위해서는 가시화 클라이언트와 서버 사이의 프로토콜이 필요하다. 기존 가시화 도구들은 개발 도구에 특화된 프로토콜을 사용하고 있으며, 이 때문에 클라이언트와 서버는 매우 tightly-coupled 되어 있다. 본 논문에서는 응용에 독립적인 분산/병렬 가시화를 위한 가시화 프로토콜을 설계하고 구현한다. 또한, 시변환 데이터의 효율적 가시화를 위해 animation을 구현할 수 있는 프리미티브를 설계하고 status machine으로 병렬 전송된 데이터들 간의 동기화를 구현한다. 이러한 응용 독립적 가시화 프로토콜을 도입함으로써 가시화는 병렬 분산 가시화를 수행하는 그리드의 서비스나 슈퍼컴퓨팅의 서비스로 확장될 수 있을 것이다.

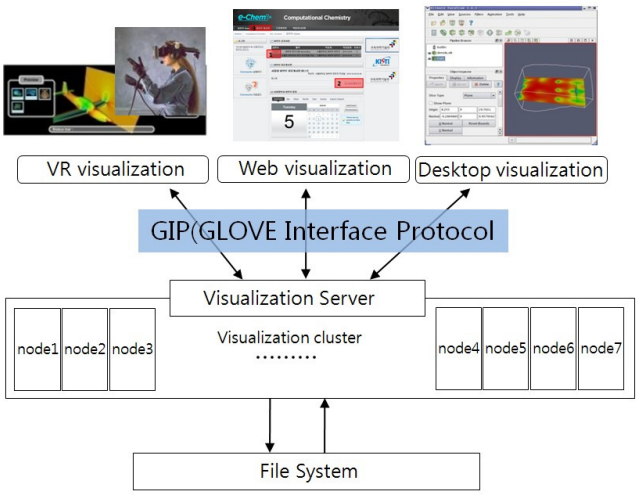
1. 서론

컴퓨팅 자원의 급격한 성능 향상으로 전산 시뮬레이션은 좀 더 정밀하고 규모가 큰 실험이 가능하게 되었다. 이에 따라 시뮬레이션 결과 데이터의 양도 테라, 페타바이트 단위로 기하급수적으로 증가하고 있다.

이 때문에, 대용량 데이터 가시화가 가능한 도구들은 사용자 인터페이스를 포함한 클라이언트에서 가시화를 요청하고, 가시화를 위한 계산은 고성능의 클러스터에서 수행하여 그 결과를 다시 클라이언트로 전송하는 분산/병렬 가시화를 지원한다.

분산/병렬 가시화를 필요로 하는 계산 시뮬레이션 과학 응용들은 다양하다. 천체과학, 유체 역학, 화학, 생물 등 그 분야의 다양성만큼 이들 응용들의 데이터 형태와 분석에 필요한 가시화 방법 또한 다양하다. 그러나, 응용이 다르더라도 이들이 원하는 가시화의 결과는 컴퓨터 그래픽의 측면에서는 하나의 일관성을 가진다. 이 때문에 범용 가시화 도구들이 존재할 수 있다[1],[2].

기존의 가시화 도구들도 대용량 데이터를 가시화 할 경우 경우 분산/병렬 가시화를 수행하기 위해 다른 노드로 데이터와 가시화 명령을 전달하는 프로토콜들이 존재한다. 그러나, 이러한 프로토콜들의 명령 구성이나 구현 방법은 가시화 도구에 tightly-coupled 되어 있어 응용에 의존적이다. 응용 독립적 가시화 프로토콜이라 함은 그림 1과 같이 어떤 응용이나 가시화 도구가 렌더링 노드에 요청을 하더라도 하나의 프로토콜로 처리할 수 있는 표준화된 프로토콜을 의미한다.



[그림 1] 응용 독립적 가시화 프로토콜 GIP

본 연구에서는 이러한 응용 독립적인 프로토콜을 설계하고 구현한다.

2. 기존 연구

시뮬레이션 데이터를 가시화하는 도구들 중 일부는 클러스터에서의 분산/병렬 가시화를 지원한다. 그러나, 병렬 가시화의 사용 빈도는 거의 없을 정도로 낮다. 이는 일반적으로 가시화를 수행하는 사용자인 응용 과학자들이 병렬/분산 가시화를 위해 수행하는 과정이 매우 복잡하기

때문이다. 대표적인 가시화 도구인 ParaView, VisIt, EnSignt 등은 병렬 분산 가시화를 수행하기 위해 먼저 가시화를 원하는 데이터의 크기를 고려해 필요한 노드를 수를 정하고 mpi를 이용해 서버를 구동한 다음 클라이언트를 실행시키는 다소 복잡한 과정을 수행해야 한다. 또한, VisIt과 같은 도구는 매우 광범위한 가시화 기능을 지원하는 범용의 가시화 도구이지만, 병렬/분산 가시화 과정에서는 좀 더 복잡한 최적화의 과정을 추가적으로 수행해야 한다. Paraview나 VisIt은 병렬 가시화를 수행할 때, 사용자 인터페이스에서 기능을 선택한 후, mpi로 구동한 서버들이 가시화를 위한 계산을 수행하고 이를 geometric model 이나 image로 다시 client로 전송한다[1],[2]. 이때 사용되는 프로토콜은 VTK pipeline에 의존적이다[5].

이 때문에, 그리드나 다른 시뮬레이션을 수행하는 응용의 입장에서도 Paraview 나 VisIt과 같은 가시화 도구의 가시화 기능을 활용하기 위해 플러그인 형식으로 따로 구동하는 방법을 사용한다[4].

본 연구에서는 한번 기동한 병렬 분산 가시화 서버에 여러 종류의 client 가 접속하여 어떤 응용이든 (그리드 middleware나 In-situ visualization 등 다른 가시화 사용자 인터페이스라 할지라도) 가시화 본연의 기능과 데이터만 기술함으로써, 응용에 독립적으로 병렬 가시화를 수행하고 그 결과를 전달할 수 있는 프로토콜인 GIP을 설계하고 구현한다.

3. GIP(GLOVE Interface Protocol)의 설계

GIP은 세 가지 관점으로 응용 독립적 프로토콜을 구현한다. 첫째, 다양한 응용을 수용할 수 있어야 하며, 둘째, 응용에 따라 달라지는 데이터를 프로토콜 패킷에 기술할 수 있어야 하며, 셋째, 다양한 가시화 기능을 일반화 하여 프로토콜에 표한할 수 있어야 한다.

먼저, 다양한 응용의 수용을 위해, GIP은 세션과 세션의 정보를 관리한다. GIP은 클라이언트의 IP와 Port 서버의 IP와 포트의 네 가지 정보의 쌍으로 구성된 session id로 구분되는 세션을 정보를 관리하며, 또한 이들 응용에 관련된 정보를 세션을 맺는 과정에 negotiation 정보로 교환함으로써 응용에 맞는 정보를 보낼 수 있다.

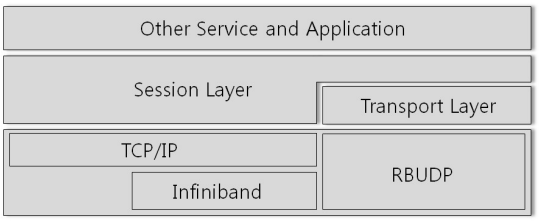
또한, GIP은 응용에 따라 달라지는 가시화 데이터의 종류를 프로토콜에 기술하기 위해 데이터와 명령 전달의 경로를 분리한다. 이를 위해, GIP은 가시화 기능을 수행하기 전에 load 명령을 서버에 전송한다. 가시화 서버는 이 명령을 받아 이들 응용 데이터를 meta 정보와 순수한 데이터 정보로 분리한다. 일반적으로 전산 시뮬레이션 결과 데이터는 structured or unstructured grid의 형태를 가진다. 따라서, 순수 데이터 정보는 배열로 변환하여 어떠한 형태의 요청에도 대응할 수 있다. 또한 결과 데이터도 순수 geometry 정보나 image 정보와 meta 정보로 구분하여 응용 클라이언트가 응용에 맞게 변환하여 사용할 수 있도록 지원한다. load 의 결과는 meta dat로 클라이언트에 보내져 서버와 데이터 정보를 공유한다.

마지막으로, 응용 독립적 가시화 기능의 기술은 범용 가시화 도구가 존재할 정도로 어느 정도 일반화가 가능하다. 단 이들 가시화 기능에 필요한 데이터를 프로토콜에 어떻게 기술 할 것인가 하는 문제만 남는다.

이를 위해 GIP API는 데이터 기술의 책임을 어느 정도 응용에 남겨 둔다. 응용은 meta 데이터를 이용해 의미론적 데이터를 그들의 인덱스로 GIP에 표현한다.

① GIP Architecture

GIP은 대용량 데이터 가시화를 위한 프로토콜로 사용자 인터페이스가 존재하는 클라이언트와 실제 가시화를 수행하는 분산/병렬 가시화 도구사이에 가시화 명령을 전달하고 그 결과를 받을 수 있어야 한다.



[그림 2] GIP Architecture

가시화 결과는 이미지나 혹은 geometry 데이터로 클라이언트에 전송할 수 있다. 가시화 요청에 대한 결과 데이터는 동영상과 같은 스트림 데이터처럼 어느 정도의 손실을 감수할 수 있는 데이터가 아니라 손실 없는 결과의 전송을 보장하여야 한다. [그림 2]는 GIP의 구조를 보여준다. 결과의 손실 없는 전송을 위해 GIP은 기본적으로 TCP/IP 위에서 동작한다. LAN 환경에서는 Infiniband 도 지원한다.

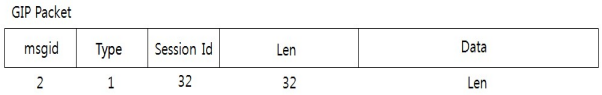
GIP은 가시화를 요청하는 클라이언트와 요청을 받아 그 결과를 응답으로 돌려주는 서버로 동작한다. GIP 서버는 다수의 클라이언트를 수용할 수 있다. 따라서, GIP 서버는 다수의 클라이언트를 관리하여야 하며, 요청을 보낸 클라이언트에 그 요청에 대한 응답을 정확히 전달하여야 한다. GIP은 요청을 보낸 후 응답이 오지 않더라도 바로 다음 요청을 처리하는 비동기식 프로토콜과 응답에 대한 요청이 올 때까지 클라이언트가 대기하는 동기식 프로토콜을 모두 지원한다.

일반적으로 비동기식 프로토콜은 클라이언트가 고해상도 디스플레이와 고성능의 네트워크 및 서버에서 실행될 때 사용하고, 동기식 프로토콜은 저해상도의 일반적인 클라이언트에서 사용한다. 병렬/분산 가시화 GIP 서버는 이들을 port 번호로 구분하여 서비스한다.

② GIP Packet

GIP 의 패킷은 [그림 3]의 형태를 가진다. GIP 은 크게 두 가지 모드의 primitive를 가진다. 클라이언트가 요청하여 응답을 받는 two way transaction을 다루는 cmd.req, cmd.res와 애니메이션 pre-fetch와 같이 서버가 생성된 데이터를 미리

전송할 때 사용하는 one way transaction인 push.req이다. GIP과 같은 비동기식 프로토콜에서 msgid 는 two way transaction 의 경우 요청과 응답이 동일한 값을 가지며, msgid는 전 시스템에서 유일해야 한다. Type은 다음에 올 data 영역에 기술된 데이터의 형태를 구분한다. GIP은 데이터에 따라 패킷의 길이가 가변적이다. Len은 가변적인 데이터 필드의 길이를 명시한다.



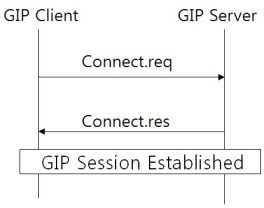
[그림 3] GIP Packet

Session Id는 서버가 클라이언트에 주는 값으로 클라이언트와 서버의 세션을 구분하는 전 시스템에 유일한 값이다. Session Id 역시 매 transaction 마다 함께 전송하여야 한다. type에 존재할 수 있는 GIP primitive들은 <표 2>와 같다. Data 필드에 올 패킷의 형태는 표 2의 type에 기술된 프리미티브의 종류에 따라 결정된다.

<표 2> GIP Primitives

Type	Value	Use case
connect.req	0x01	session 의 연결을 요청
connect.res	0x02	session의 연결 요청에 대한 응답
cmd.req	0x03	가시화 요구
cmd.res	0x04	가시화 결과
push.req	0x05	서버에서 클라이언트로의 요청

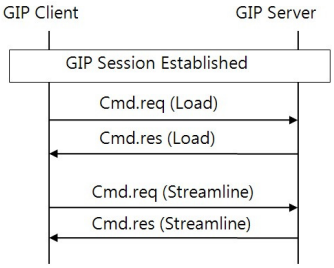
GIP은 다양한 응용에 대한 정보를 관리하고, 클라이언트에서 보내는 병렬 요청을 수용하기 위해, GIP 서버는 요청에 대한 응답의 목적지에 대한 정보를 세션의 개념으로 가지고 있어야 한다. 이를 위해 GIP은 클라이언트의 정보를 받을 수 있는 connect.req와 connecr.res를 제공하여 세션을 관리한다.



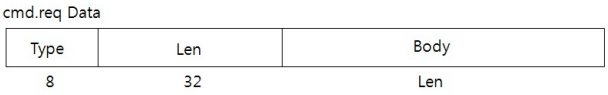
[그림 4] GIP Session 요청

③ 가시화 요청과 응답

하나의 GIP 세션이 만들어지면 GIP 클라이언트는 GIP서버에 cmd.req를 보낼 수 있다. cmd.req에 대한 요청은 cmd.res로 응답한다. cmd.req의 data field는 [그림 6]과 같이 구성된다. Type은 GIP 클라이언트가 서버에 요청할 수 있는 cmd.req의 종류를 기술한다. GIP 클라이언트는 가시화에 대한 요청을 보내기 전 반드시 Load를 수행하여야 한다. 기본적으로 GIP은 가시화를 요청할 서버로 응용의 원 데이터를 전송하는 프로토콜은 아니다. 따라서, Load는 가시화해야 할 데이터 셋의 저장 위치를 URI로 명시해 준다. cmd.req에 의해 load 가 끝나면, 클라이언트는 가시화를 cmd.req로 요청할 수 있다.



[그림 5] Cmd.req 와 Cmd.res



Type	Value
Load	0x01
Scalar Distribution	0x02
Graph	0x03
Iso-surface	0x04
Streamline	0x05
Animation	0x06
Pathline	0x07
Probe by Plane	0x08
Transfer function	0x09

[그림 6] cmd.req data type

[그림 7]은 cmd.res의 body packet의 형태를 보여준다.

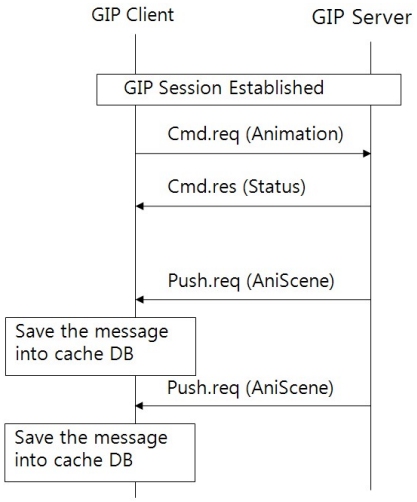
Type	Value
Poly Data	0x01
Image	0x02
Status	0x03
Transfer function key	0x04

[그림 7] 가시화 응답의 형태

④ 애니메이션 캐쉬 데이터의 전달 Push.req

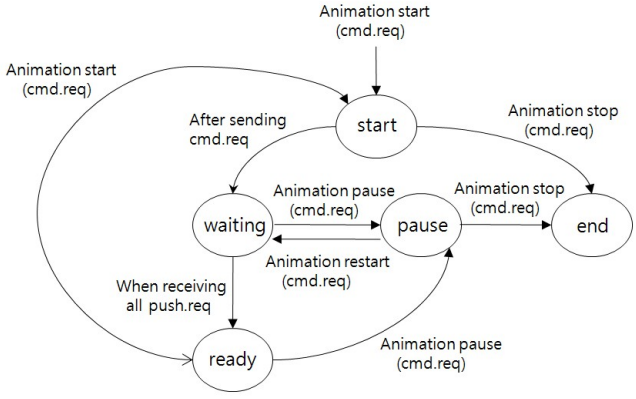
GIP 은 시변환 데이터의 효율적인 애니메이션을 위해 push.req를 제공한다. push.req는 pre-fetch를 위해 클라이언트가 요청하지 않더라도 서버가 데이터를 클라이언트에 전송할 수 있는 프리미티브이다. 애니메이션을 위해 먼저, 클라이언트는 cmd.req의 type이 animation인 요청을 서버에 보낸다. 이 요청의 데이터 필드는 cache num을 포함한다. 이는 서버가 pre-fetch할 데이터의 수를 의미한다. 클라이언트는 push.req를 받으면 그 데이터 형태가 animation scene일 경우 해당 데이터

를 pre-fetch 데이터베이스에 저장한다. [그림 8]은 클라이언트가 서버에 애니메이션을 요청하고 서버가 push.req를 이용해 생성된 데이터를 전송하는 과정을 보여준다. 서버는 그려질 가시화 오브젝트가 생성되는 대로 하나의 push.req를 생성하여 클라이언트로 전송한다.



[그림 8] GIP prefetch sequence

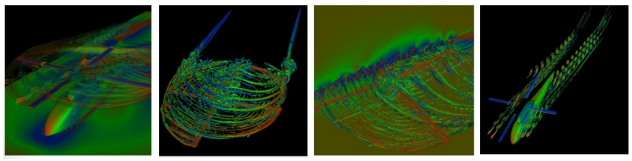
push.req를 받은 GIP 클라이언트는 애니메이션에 대한 state machine을 관리하여 오브젝트에 대한 동기화를 수행한다. 현재 time step에서 하나의 animation scene을 구성하는 모든 오브젝트들이 push.req로 도착하면, state는 ready 상태가 되고, 이 상태에서만 클라이언트의 viewer는 이들 오브젝트를 그릴 수 있다.



[그림 9] Animation state machine

4. GLOVE 에서의 GIP

GIP은 GLOVE의 통신 프로토콜로 개발되었다. GLOVE (GLObal Virtual Environment for collaborative research) 시스템은 KISTI에서 개발한, 대용량 데이터를 가상현실에서 구현하는 가시화 도구이다[6]. GIP은 GLOVE에서 가상현실 사용자 인터페이스인 GIVI와 렌더링 엔진인 GLORE 간의 통신을 담당한다. GLOVE는 응용 의존적 사용자 인터페이스인 GIVI와 응용 독립적 렌더링 엔진인 GLORE로 구성된다. GIVI는 가시화를 위한 렌더링을 요청하기 위해 GIP을 사용하여 GLORE에 가시화하기 위한 데이터와 그 표현방법에 대한 명세를 전달한다. [그림 10]은 GLOVE에서 가시화 프로토콜인 GIP을 이용하여 가시화한 결과이다. 로터 데이터 시뮬레이션의 결과로 iso-surface와 probe by plane, scalar distribution의 다양한 결과를 GIP으로 구현하였다.



[그림 10] GIP을 활용한 GLOVE 로터 데이터 시뮬레이션 가시화 결과

5. 결 론

이상에서 살펴 본 바와 같이 GIP은 세션 관리를 통해 다중 클라이언트를 수용하고, 다양한 WAN 환경의 네트워크를 지원할 뿐 아니라, 기본적인 가시화 기능들을 수행하도록 설계되었다. 또한, 시변환 데이터의 애니메이션 시에 pre-fetch 기능을 지원함으로써, 응용의 효율을 높일 수 있다. 무엇보다 GIP은 가시화 데이터와 명령의 이동 경로를 분리하여, 데이터 loading 시에 다양한 기술을 적용할 수 있는 유연성을 제공한다. 이러한 유연성은 다양한 응용을 수용할 수 있을 뿐 아니라, 가시화 서버 개념의 서비스로 슈퍼컴퓨터의 또 다른 서비스 영역으로 활성화 시킬 수 있다.

참고 문헌

[1] Andy Cedilnik, Berk Geveci, "Remote Large Data Visualization in ParaView Framework", Eurographics Symposium on Parallel Graphhics and Visualization, 2006

[2] VisIt, http://www.visitusers.org/index.php

[3] John Biddiscombe, Jerome Soumagne, Guillaume Oger, David Guibert, Jean-Guillaume Piccinali, "Parallel Computational Steering and Analysis for HPC Applications using a ParaView Interface and HDF5 DM Virtual File Driver", Eurographics Symposium on Parallel Graphics and Visualization, 2011

[4] Arnas Kaceniauskas, Ruslan Pacevic, "Efficient Visualization By using ParaView software on BalticGrid".

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

[6] Min Ah Kim, "GLOVE(GLObal Virtual reality Environment for scientific simulation): VR환경에서의 대용량 데이터 가시화 시스템", 정보과학회, 2010.