ISBN 978-89-6211-605-2

원격렌더링을 위한 응용 독립적 가시화 프로토콜 연구

(Visualization Network Protocol for Remote rendering)

김 민 아 (petimina@kisti.re.kr)

한 국 과 학 기 술 정 보 연 구 원
Korea Institute of Science & Technology Information

목차

그림 차례

표 차례

1. 개 요

지구과학, 기상, 천체물리, 항공우주공학 등 과학과 엔지니어링 분야에서 전산 시뮬레이션은 복잡한 과학 현상을 모사하여 분석한다. 시뮬레이션 데이터의 분석 방법에는 그래프, 수치 데이터를 이용한 정량적 분석과 데이터의 가시화를 이용한 직관적이며 정성적인 분석의 방법이 존재한다. 우리는 가시화를 이용한 정성적 분석 방법으로 전체 데이터의 양상과 흐름을 알 수 있다. 따라서, 전산 시뮬레이션이 올바른 방향으로 진행되고 있는 지에 대해 즉각적 판단이 가능하다.

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. 시뮬레이션 데이터의 양이 증가함에 따라 과학적 가시화 관련 기술도 병렬 데이터 처리, 고해상도 렌더링, 복잡한 데이터를 알아보기 쉽게 보여주기 위한 표현 방법 등을 지원하도록 발전해 왔다. 또한 이들을 수행하기 위한 컴퓨팅 및 디스플레이 환경도 고성능, 고해상도를 필요로 한다.

그러나, 이러한 대용량 데이터의 가시화를 필요로 하는 실험실의 환경은 가시화를 위한 렌더링을 직접 수행할 수 있는 경우가 극히 드물다. 이 때문에, 가시화 도구들은 실험실에서 가시화를 요청하고, 렌더링은 고성능의 슈퍼컴퓨터가 있는 슈퍼컴퓨팅 센터에서 수행하여 그 결과를 다시 실험실로 전송할 수 있는 원격 렌더링을 지원한다. 그러나 원격 렌더링 요청은 가시화 도구에 의해 숨겨져 있으며, 가시화 도구에 특화되어 있다.

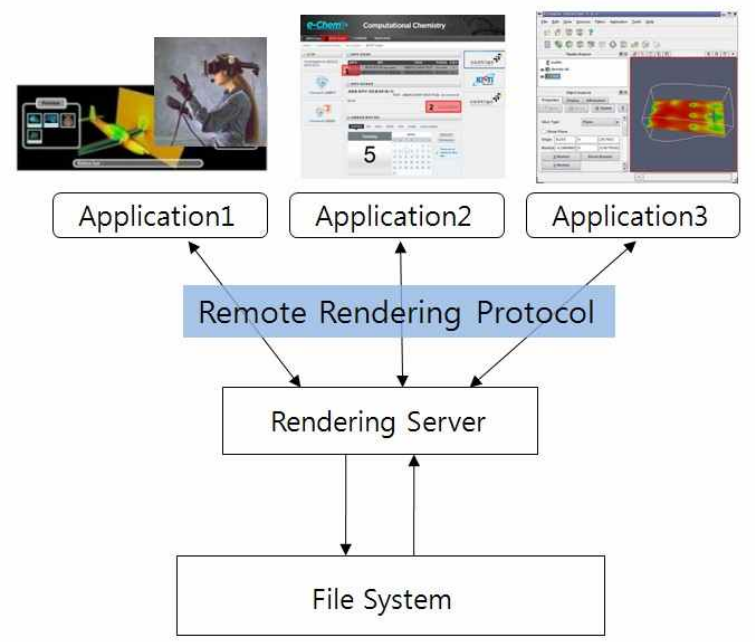
본 연구에서는 어떤 가시화 도구나 응용이 요청을 하더라도 표준화된 프로토콜로 가시화를 요청할 수 있도록 기본적인 원격 렌더링 요청에 대한 프로토콜을 정의하여 응용 독립적인 프로토콜을 설계하고 구현한다.

이러한 프로토콜의 구현을 위해 본 문서에서는 먼저 표준 프로토콜의 요구사항을 분석하고, 요구사항을 만족시키기 위한 프로토콜로써의 GIP을 상세 설계한다. 마지막으로 이러한 GIP 프로토콜을 구현한 GLOVE에서의 GIP의 구현 및 실행 메커니즘을 설명한다.

2. 원격 가시화 렌더링 프로토콜의 요구 사항

가. 응용 독립성

가시화를 필요로 하는 계산 시뮬레이션 과학 응용들은 다양하다. 천체과학, 유체역학, 화학, 생물 등 그 분야의 다양성만큼 이들 응용들의 데이터 형태와 분석에 필요한 가시화 방법 또한 다양하다. 원격 가시화 렌더링 프로토콜이 하나의 표준화된 렌더링 프로토콜이 되기 위해서는 이들 응용이 요구하는 가시화 방법을 모두 지원해야 한다. 다행히 응용이 다르더라도 이들이 원하는 가시화의 결과는 컴퓨터 그래픽의 측면에서는 하나의 일관성을 가진다. 이 때문에 일반적인 가시화 도구들이 존재할 수 있다[1,2]. 이러한 가시화 도구들도 대용량 데이터의 경우 병렬 렌더링을 수행하기 위해 다른 노드로 데이터와 가시화 명령을 전달하는 방법이 있다. 그러나, 이러한 방법은 일반적으로 계산 과학에서 병렬 렌더링을 수행할 때 사용하는 인터페이스로 MPI와 같이 가시화에 특화된 방법은 아니다. 따라서, 명령의 구성이나 구현 방법은 가시화 도구에 tightly-coupled 되어 있어 어떤 표준화된 형태를 가진 프로토콜이라 볼 수는 없다. 원격 가시화 렌더링 프로토콜이라 함은 어떤 응용이나 가시화 도구가 렌더링 노드에 요청을 하더라도 하나의 프로토콜로 처리할 수 있는 응용 독립성이 있어야 한다.



[그림 1] Application independent remote rendering protocol for visualization

나. 가시화 요청에 대한 명세

원격 렌더링 프로토콜은 가시화를 위한 렌더링을 요청하는 프로토콜이므로 하나의 가시화 결과를 위해 데이터와 가시화에 대한 명확한 명세를 표현할 수 있어야 한다. 이를 위해 기본적으로 렌더링 프로토콜이 지원해야 하는 기능은 다음과 같다.

1) Scalar value distribution

가시화를 표현하는 어떤 데이터 셋도 좌표와 그 좌표에 해당하는 scalar 값을 가진다. 응용 입장에서는 다른 의미로 해석될 수 있지만, 가시화 처리 서버의 입장에서 모든 데이터는 좌표와 scalar 값으로 표현될 수 있다. scalar 데이터의 값은 일정한 분포를 가지며, 분석을 위해 이들 분포를 가시화하여 보여주는 요구는 가장 일반적인 요구이다. 따라서, 프로토콜은 scalar value의 distribution을 요청할 수 있어야 한다.

2) Contour line

Contour line은 동일한 범위의 값을 가지는 데이터를 하나의 선으로 연결하여 보여준다. 이러한 표현 방식은 scalar distribution의 일종으로 면과 색이 아닌 선과 색으로 데이터의 양상을 보여 준다. 원격 렌더링 프로토콜은 이러한 형태의 가시화도 지원하여야 한다.

3) Iso-surface

동일한 범위의 값을 가지는 데이터를 면으로 연결하여 표면 데이터로 object화 시키는 가시화 표현 방법이다. 의료 데이터의 경우, 뼈나 살갗 등 구분 된 값으로 구성된 하나의 오브젝트로 보여줄 수 있다. 유체 데이터의 경우 vorticity 와 같은 현상을 하나의 오브젝트로 볼 수 있다. 가시화의 중요한 표현 방법으로 프로토콜은 이를 지원해야 한다.

4) Streamline

유동 데이터의 흐름을 보여주는 가시화 표현 방법이다. 유체 역학에서 물이나 공기의 흐름, 인체 데이터의 혈류 등 다양한 데이터를 가시화 할 수 있으므로 가시화 프로토콜은 이를 지원해야 한다.

5) Pathline

Pathline은 일정 기간이상 유체의 이동 경로를 보여 준다. 이는 유체의 움직임을 보여주는 매우 중요한 가시화 요소로 가시화 프로토콜은 이를 지원해야 한다.

6) Probe by plane

데이터의 분석을 위해 응용 가시화 도구의 사용자는 전체 데이터가 아니라 특정 부분의 데이터를 자세히 보고자 하는 경우가 많다. 사용자의 관심 영역을 plane으로 지정하여 데이터를 표현할 수 있는 방법으로 가시화 프로토콜은 이를 지원해야 한다.

다. 시변환 데이터의 애니메이션

유체 역학데이터와 같은 시변환 데이터의 가시화는 시간에 따른 애니메이션 지원이 필수적이다. 따라서, 원격렌더링 가시화 프로토콜 또한 이를 지원해야 한다. 또한 애니메이션의 효율을 향상시키기 위해 캐쉬에 대한 처리도 수행할 수 있어야 한다.

라. 가시화 결과 전달

원격 렌더링 가시화 프로토콜은 가시화에 대한 요청뿐만 아니라 그 결과 또한 이를 요청한 가시화 클라이언트에 전달할 수 있어야 한다. 렌더링의 결과는 가시화 도구나 응용이 요청한 형태로 제공되어야 하며, 데이터가 클 경우 이를 효율적으로 전달할 수 있는 방법도 포함해야 한다. 가시화의 결과는 geometry 데이터 일수도 이미지 데이터 일 수도 있다. 원격 렌더링 가시화 프로토콜은 어떤 형태의 데이터이든 클라이언트에 전달할 수 있어야 한다.

마. WAN 환경의 네트워크 프로토콜 지원

가시화를 요청하는 응용이나 가시화 도구는 일반적으로 이를 수행하는 렌더링 팜과 원거리에 존재하여 WAN으로 연결되어 있는 경우가 많다. 따라서, 가시화 프로토콜의 기반이 되는 네트워크 프로토콜은 WAN 환경을 지원할 수 있어야 한다. 그러나, 이와 동시에 내부의 특수한 환경도 지원할 수 있어야 한다. Infiniband와 같이 대용량의 데이터를 전달할 수 있는 네트워크는 물론 기본적인 TCP/IP 외에도 RBUDP와 같은 기본적인 프로토콜을 지원해야 한다.

바. 다중 클라이언트 지원

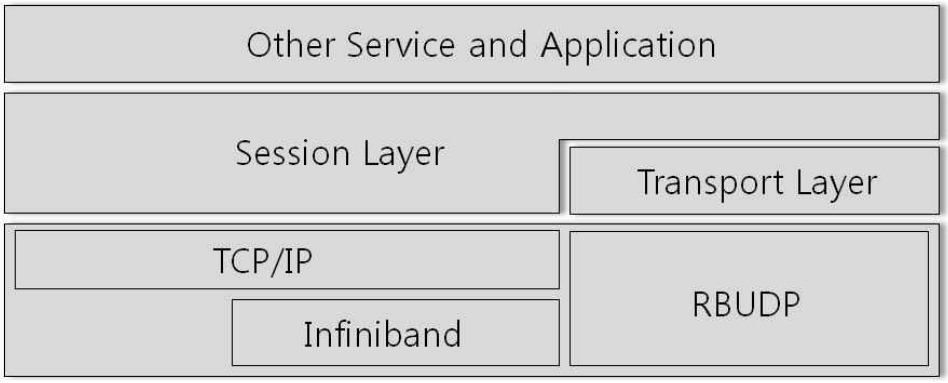
원격 렌더링 프로토콜은 렌더링 서버와 가시화 도구간의 프로토콜이다. 렌더링

서버는 기본적으로 하나의 가시화 도구에 대한 서비스를 위해 설계하지는 않는다. 다양한 응용을 수용하고, 여러 클라이언트에 대해 서비스 할 수 있는 렌더링 서버를 고려할 때, 프로토콜은 여러 클라이언트를 관리할 수 있어야 한다.

3. GIP (GLOVE Interface Protocol)의 설계

가. GIP Architecture

GIP은 실험실에서 가시화 렌더링을 요청하는 가시화 도구와 고성능의 렌더링 팜을 구축하여 렌더링을 수행할 수 있는 렌더링 서버 사이의 통신을 위한 프로토콜이다.



[그림 2] GIP Architecture

또한, 렌더링 결과를 이미지나 혹은 geometry 데이터로 클라이언트에 전송하여야 한다. 따라서, 결과 데이터는 동영상과 같은 스트림 데이터처럼 어느 정도의 손실을 감수할 수 있는 데이터가 아니라 손실 없는 결과의 전송을 보장하여야 한다.

그림 2는 GIP의 구조를 보여준다. 결과의 손실 없는 전송을 위해 GIP은 기본적으로 TCP/IP 위에서 동작한다. LAN 환경에서는 Infiniband 도 지원한다. 또한, 대용량 데이터의 전달을 위해 RBUDP도 지원한다. 그러나, RBUDP는 데이터 손실이 발생하므로 GIP transport layer에서는 재전송과 트랜젝션의 관리를 담당한다.

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 한 클라이언

트에 그 요청에 대한 응답을 정확히 전달하여야 한다. GIP은 요청에 대해 응답을 줄 때까지 다른 요청을 처리하지 않는 동기식 프로토콜이 아니라 요청을 전달하면 응답이 오지 않더라도 바로 다음 요청을 처리하는 비동기식 프로토콜이다. Session layer는 GIP 서버가 다수의 클라이언트를 지원할 수 있도록 GIP 서버와 클라이언트 사이의 연결을 담당한다. 그 연결 내에서 기본적인 primitive를 지원한다. Session Layer 자세한 내용은 아래 기본 Primitive에서 다룬다.

나. GIP Packet

GIP 의 패킷은 그림 2의 형태를 가진다. GIP 은 크게 두 가지 모드의 primitive를 가진다. 클라이언트가 요청하여 응답을 받는 two way transaction을 다루는 cmd.req, cmd.res와 캐쉬와 같이 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 가짐으로써, 요청과 응답의 쌍을 구분한다. 따라서, msgid는 전 시스템에서 유일해야 한다. Type은 다음에 올 Data 영역에 기술된 데이터의 형태를 구분한다. GIP은 데이터에 따라 패킷의 길이가 가변적이다. Len은 가변적인 Data Field의 길이를 명시한다.

GIP Packet

msgid	Type	Session Id	Len	Data
2	1	32	32	Len

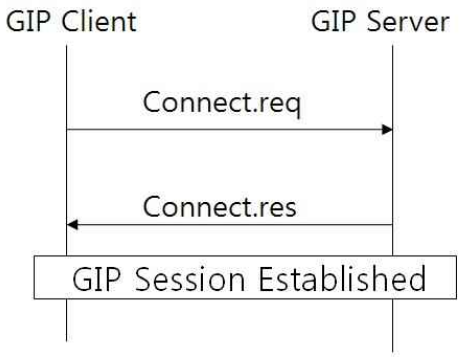
[그림 3] GIP Packet

Session Id는 Connect 서버가 클라이언트에 주는 값으로 클라이언트와 서버의 세션을 구분하는 전 시스템에 유일한 값이다. Session Id 역시 매 transaction 마다 함께 전송하여야 한다. 단, Connection.req의 경우, 아직 서버로부터 값을 받지 못하였으므로 의미 없는 값이 전달된다. 아래 표는 Type에 존재할 수 있는 값들이다.

[표 1] connection.req types

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	캐쉬 등 서버에서 요청 없이 결과를 요청

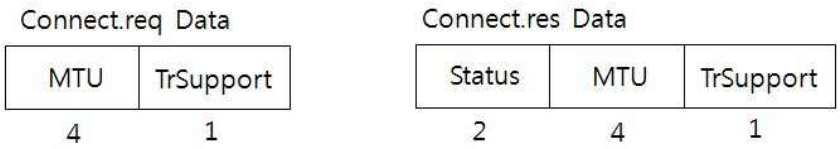
다. GIP Session 요청을 위한 Connect.req, Connect.res



[그림 4] GIP Session 요청

GIP은 기본적으로 비동기 프로토콜로 요청에 대한 응답이 올 때까지 기다리지 않고 다음 트랜젝션을 처리한다. 이 때문에, GIP 서버는 요청에 대한 응답의 목적지에 대한 정보를 세션의 개념으로 가지고 있어야 한다. GIP 서버는 GIP의 클라이언트가 연결 요청을 할 때 GIP의 클라이언트에 대한 정보를 세션 정보를 저장하고, 연결을 끊을 때 이를 삭제해야 한다. 그림 3은 GIP Session 이 만들어지는 과정을 보여 준다.

GIP은 세션의 연결을 위해 Connect.req와 Connecr.res를 제공한다. Connect.req와 Connect.res primitive는 다음과 같이 정의한다.

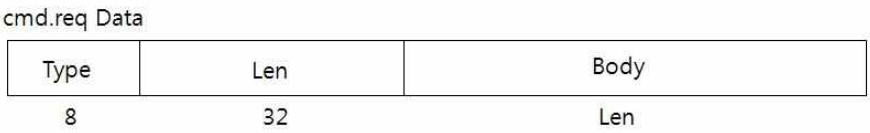


[그림 5] Connect.req와 connect.res

Connect.req Data와 Connect.res는 GIP Packet의 Data Field를 보여준다. Connect.req 요청 시 GIP 클라이언트는 클라이언트의 네트워크에 대한 명세를 서버에 전달한다. MTU는 mtu 크기를 의미하고, TrSupport는 RBUDP로 데이터를 전송할지 여부를 나타낸다. 0x00이면 GIP의 transport layer를 지원하지 않음, 즉 TCP/IP를 의미하고, 0x01 이면 GIP Transport layer를 지원하는 것을 의미한다. Connect.res는 Connect.req에 Status를 더한다. Status는 클라이언트와 서버 사이에 GIP 세션이 만들어 졌는지 여부에 대한 상태를 명시한다. 만일 클라이언트와 서버의 네트워크 명세가 다를 경우, Connect.res는 실패에 대한 오류 코드를 Status에 명시한다.

라. 가시화 요청과 응답

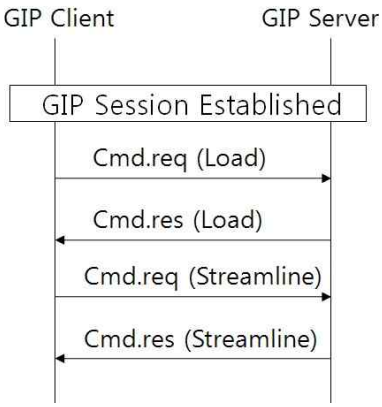
Connect.req와 Connect.res 로 하나의 GIP 세션이 만들어진 후에 GIP 클라이언트는 GIP서버에 cmd.req를 보낼 수 있다. 클라이언트가 cmd.req로 요청할 수 있는 내용은 그림5와 같다.



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

cmd.req의 data field는 다음과 같이 구성된다. Type은 GIP 클라이언트가 서버에 요청할 수 있는 cmd.req의 종류를 기술한다. GIP 클라이언트는 모든 요청의 전에 Load를 수행하여야 한다. 기본적으로 GIP은 가시화를 요청할 서버로 응용의 원 데이터를 전송하는 프로토콜은 아니다. 따라서, Load는 가시화해야 할 데이터 셋의 저장 위치를 명시해 준다. cmd.req에 의해 load 가 끝나면, 클라이언트는 가시화를 cmd.req로 요청할 수 있다. load 는 한번 요청하면, 새로운 데이터 셋에 대한 요구를 하기 전까지는 다시 보내지 않아도 된다.



[그림 7] 일반적인 가시화 sequence

cmd.req에 대한 요청은 cmd.res 로 응답한다. 그림 7은 cmd.res의 body packet의 형태를 보여준다. Type field는 값들로 data field의 종류를 의미하며, 4가지 형태의 값을 가질 수 있다. Poly Data는 cmd.req의 Load를 제외한 가시화 요청에 대해 geometry data로 결과를 보낼 때 사용한다. Image는 가시화 요청에 대한 응답을 이미지로 보낼 때 사용한다. Status는 cmd.req가 결과를 만들지 못하고 실패했을 때, 실패의 원인인 오류코드를 보낼 때 사용한다. Transfer function key는 Load 에 대한 응답으로 전체 데이터의 transfer function을 만들 수 있는 팔레트를 보낼 때 사용한다.

cmd.res Body

Type	Len	Data
8	32	Len

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

[그림 8] 가시화 응답의 형태

다음 절에서는 이들 각각의 cmd.req 와 cmd.res의 형태에 따른 data field의 구성에 대해 설명한다.

마. 가시화 요청을 위한 cmd.req

1) Load

Load 는 cmd.req 위에서 언급한 바와 같이 가시화 하고자 하는 데이터를 읽어 들일 때 보내는 요청의 한 형태이다. Load의 data field는 읽어 들일 데이터가 존재하는 위치를 나타내는 문자열이다. 렌더링 서버와 데이터가 동일한 파일 시스템을 가지고 있지 않을 경우, 그의 네트워크 주소를 포함한 문자열을 포함하여 보냄으로써, 데이터 위치를 기술한다.

cmd.req Load Primitive Body

filename prefix of simulation results to load

Type	Value
Load	0x01

[그림 9] cmd.req Load 의 데이터 필드

2) Scalar Distribution

Scalar distribution은 스칼라 값에 대한 분포를 가시화하기 위한 요청이다. Scalar distribution은 이에 대한 요청을 위해, 원하는 scalar 값에 대한 type과 시변환

데이터의 경우 현재 time step을 보낸다. 그림 9에 type 값들은 로터 블레이드 시뮬레이션 데이터의 예이다. 다른 응용이 존재할 경우 type에 다른 의미를 부여하면 된다. 이 값들에 의미를 부여하는 것은 GIP 프로토콜을 사용하는 응용의 몫이다.

cmd.req Scalar distribution Body

Type	Time Step
8 bit	16 bit

Type	Value
Blade Pressure	0x01
Field Pressure	0x02
Vorticity	0x03
Velocity	0x04

Time Step : Current time step

[그림 10] Scalar distribution body

3) Graph

GIP은 정성적 분석을 위한 가시화 뿐 아니라 정량적 분석을 위한 가시화 요청도 수행할 수 있다. Graph 는 이를 지원한다. 그림 10은 로터 블레이드 시뮬레이션 데이터의 그래프 요청 형태에 대한 값들이다. GIP의 가시화 응용들은 이를 다른 의미로 사용할 수 있다. 또한 시변환 데이터의 경우 그림 11의 형태로 요청가능하다.

cmd.req Graph Primitive Body

Type	Body
8	Len-8

Graph	Representation	Value
Pitch angle variation	At a span position	0x01
Sectional normal force	At a span position	0x02
Sectional chord force	At a span position	0x03
Sectional span force	At a span position	0x04
Sectional x-moment	At a span position	0x04
Sectional y-moment	At a span position	0x05
Sectional z-moment	At a span position	0x06
Pressure distribution	As azimuth angle	0x07
Pitch angle variation	As azimuth angle	0x10
Sectional normal force	As azimuth angle	0x11
Sectional chord force	As azimuth angle	0x12
Sectional span force	As azimuth angle	0x13
Sectional x-moment	As azimuth angle	0x14
Sectional y-moment	As azimuth angle	0x15
Sectional z-moment	As azimuth angle	0x16
AverageV data	Whole time step	0x17

[그림 11] Graph Type

cmd.req Graph primitive Body, when the graph at a span position is drew

Time Step	Span position
16	16

Type	Value
Span position	0x01

[그림 12] 시변환 데이터의 그래프 가시화 요청

4) Iso-Surface

Iso-surface 형태의 가시화에 대한 요청은 많은 필드를 포함한다. Iso-surface는 특정범위의 값을 가진 데이터들을 하나의 면으로 보여 주는 것이다. Type 필드는 iso-surface로 가시화할 데이터가 어떤 값인지를 나타낸다. Time step은 시변환 데이터일 경우 어느 time step인지를 명시한다. Value type은 뒤에 기술

될 값의 범위에서 값이 %인지 실제 물리적 값인지를 나타낸다. Filler type은 가시화된 iso-surface의 표면에 어떤 transfer function을 적용할 지를 명시한다. IS Num은 특정 범위 값들을 등간격의 여러 개의 iso-surface로 가시화할 때 iso-surface의 개수를 나타낸다. 마지막으로 Range와 start, end 값은 그림 12에 명시된 Range의 값과 같이 범위를 나타내기 위한 부등호와 그 값을 의미한다.

cmd.req Iso-surface primitive Body

Type	Time Step	Value Type	Filler Type	IS Num	Range	Start	End
8	16	8	8	16	8	64	64

Type	Value
Vorticity	0x01
Q-Criteria	0x02
Velocity	0x03

Value Type	Value
%	0x01
Physical value	0x02

Range	Value
>=Start	0x01
<=Start	0x02
>=Start and <= End	0x03
== Start	0x04

[그림 13] Iso-surface 데이터 필드

5) Streamline

Streamline은 데이터의 흐름의 모양을 보여준다. 이에 대한 가시화를 요청하기 위해 GIP은 그림 13 같은 데이터 필드를 가진다. Vector type은 어떤 벡터 값에 대한 흐름을 보고자 하는지를 명시한다. Time step은 가시화 하고자 하는 time step을, presentation type 은 흐름을 line 으로 보여줄지 particle 로 보여 줄지에 대한 표현방식을 나타낸다. SP generation method는 streamline의 seed point를 점, 선, 면에서 자동으로 만들지 seed point list를 줄 지를 명시한다. 이 SP generation method에 따라 SP generation body는 달라진다. SP generation method 가 Auto 일 경우 SP generation body는 없다.

cmd.req Streamline primitive Body

Vector Type	Time Step	Presentation Type	SP Generation Method	SP Generation Body
8	16	8	8	

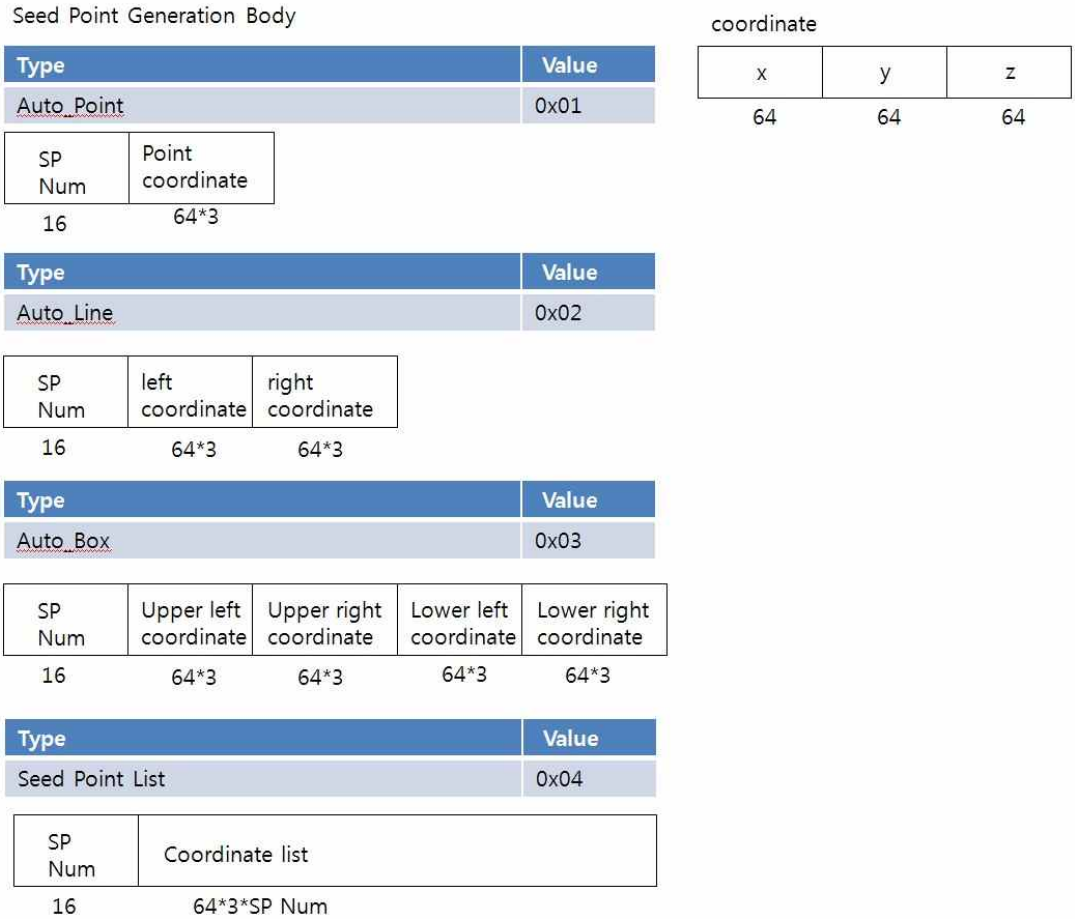
Scalar Type	Value
Vorticity	0x01
Velocity	0x02
	0x03

Presentation Type	Value
line	0x01
particle	0x02

Seed Point Generation Method	Value
Auto	0x01
Auto Point	0x02
Auto Line	0x03
Auto Plane	0x04
Seed Point List	0x05

[그림 14] Streamline 데이터 필드

Auto Point, Auto Line, Auto Plane일 경우 그림 14와 같이 SP generation body를 갖는다. Point, line, plane 각각에 대해 3차원 좌표계 상의 좌표로 seed point의 위치를 기술할 수 있다. 또한 SP generation method 가 Seed Point List 일 경우 그림 14와 같이 seed point generation body는 각각의 seed point 에 대한 좌표의 리스트를 명시할 수 있다.



[그림 15] Seed Point Generation Body

6) Pathline

Pathline은 주어진 기간 이상 하나의 유체가 따라온 길이다. 이러한 가시화는 주로 유체 역학 응용에서 많이 사용되는 가시화 기법이다. GIP은 pathline을 요청하기 위해 그림 15의 데이터 필드를 가진다. type은 어떤 데이터 값에 대한 pathline을 보여 줄지를 명시한다. Pathline을 위해 주어진 기간을 명시하기 위해 사용한다. 기간은 시작시각과 끝시각이 있어야 하지만, pathline은 start time부터 데이터가 존재하는 시각까지의 기간 이상을 보여주기 때문에 요청은 시작 시각인 start time만 포함한다. Presentation type과 SP Generation Method와 SP Generation Body는 streamline 에 대한 요청과 동일하다.

cmd.req Pathline primitive Body

Type	Start Time	Presentation Type	SP Generation Method	SP Generation Body
8	16	8	8	

Type	Value
Vorticity	0x01
Velocity	0x02
	0x03

Presentation Type	Value
line	0x01
particle	0x02

Seed Input Type	Value
Auto	0x01
Auto Point	0x02
Auto Line	0x03
Auto Plane	0x04
Seed Point List	0x05

SP Generation Body is same to Streamline primitive body

[그림 16] Pathline 데이터 필드

7) Animation

시변환 데이터의 경우 애니메이션은 데이터를 가시화하기 위한 좋은 방법이다. GIP은 애니메이션에 대한 요청을 지원한다. 애니메이션은 type과 body를 가진다. type은 애니메이션을 제어하는 opration의 type을 명시한다. GIP은 start, stop, pause, resume, next의 operation을 제공한다. Animation start body의 animation id는 클라이언트가 애니메이션을 관리하기 위해 부여하는 값으로 하나의 클라이언트 내에서 유일하다. 이 값은 클라이언트의 요청 시, 클라이언트에서 애니메이션 start 마다 새로운 값으로 부여해서 전송한다. 이외의 operation에 대해서는 start의 값과 동일한 값을 사용한다. Start time step, end time step은 애니메이션을 수행할 시작과 끝 시각을 의미한다. Num of time steps to cache 는 애니메이션 가시화의 효율을 위해 cache 할 데이터의 수이다. GIP은 애니메이션을 위해 cache 기능을 지원하므로 클라이언트가 요청한 cache의 수만큼 서버는

클라이언트가 따로 요청하지 않아도 가시화 결과를 전송해야 한다.

cmd.req Animation primitive

Type	Body
8	Len-8

Type	Value
start	0x01
stop	0x02
pause	0x03
resume	0x04
next	0x05

cmd.req Animation primitive type start body

Animation id	Start Time step	End Time step	Num of time steps to cache	Animation Data Type	Context
16	16	16	16	8	

cmd.req Animation primitive type stop body

Type	Animation id
8	16

cmd.req Animation primitive type next/pause/ resume body

Type	Animation id	Time step
8	16	32

[그림 17] Animation 데이터 필드

cmd.req Animation primitive Animation data type

Animation data type	Value
Scalar Distribution	0x01
Graph	0x02
Iso-surface	0x03
Streamline	0x04
Pathline	0x05
Probe by Plane	0x06

cmd.Req Animation primitive Context

Type	Body
8	Len-8

Body is same to cmd.req Body for each type

[그림 18] 애니메이션 데이터 형태와 Context

Animation data type은 실제로 애니메이션을 수행할 데이터의 표현방식이다. cmd.req로 요청할 수 있는 모든 형태를 포함하며(load를 제외한), 뒤에 표현될 cont

ext 는 이 형태에 따라 결정된다. Context는 cmd.req의 각 요청에 대한 데이터 필드의 형태와 같다.

Animation stop은 animation id 로 구분되는 animation을 멈춘다. 따라서, animation의 데이터 필드는 animation id 만 포함한다.

Animation next와 pause, resume 역시 animation id를 가진 애니메이션에 대한 operation 으로 next는 다음 scene에 대한 요청이며, pause는 잠시 멈춤, resume은 다시 시작을 위해 사용한다. 이들 역시 animation id로 구분되는 애니메이션에 대한 operation이다. 그러나, GIP의 클라이언트와 서버는 cache를 수행하고 있어 미리 특정 time step에 대한 결과를 전송하고 있으며, 이들 간의 물리적 거리로 인하여 현재 수행하고 있는 time step은 다를 수 있다. 따라서, 이들 간의 동기화를 위해 원하는 time step을 함께 명시한다.

8) Probe by Plane

cmd.req Probe by plane Body

Value Type	Presentation Type	Time Step	Plane coordinate	Presentation Body
8	8 bit	16 bit		

Presentation Type	Value
Contour line	0x01
Scalar distribution	0x02

cmd.req "probe by plane" plane coordinate

Upper left coordinate	Lower right coordinate
64*3	64*3

Coordinate is same to Seed Point Generation Body's coordinate of cmd.req Streamline primitive

If a presentation type is scalar distribution, the presentation body does not exist

[그림 19] Probe by plane의 데이터 필드

Probe By Plane은 사용자 인터페이스에서 사용자가 3차원 공간상에 하나의 평면을 만들면, 그 평면에 존재하는 데이터를 원하는 표현방식으로 보여준다. GIP은 Probe By Plane을 그림 18의 형태로 요청한다. Value type은 표현할 값을 명시한다. Presentation type은 평면에 표현될 방법을 의미하며, contour line과 scalar distribution 의 두 가지 형태를 지원한다. Plane coordinate은 3차원 공간상의 평면의 좌표로 Upper left 와 lower right 의 두 점으로 평면의 좌표를 결

정한다. Presentation type 에 따라 presentation의 body는 달라진다. Presentation type이 scalar distribution 일 경우 가시화에 대한 모든 항목이 결정되었기 때문에 presentation body는 없다. Presentation type이 contour line일 경우, 그림 19와 같이 두 가지의 presentation body가 존재한다. Gen method가 AutoLogScale이나 Auto 인 경우에는 전체 라인의 개수만 명시하고, Value list 일 경우 실제 value의 list를 명시한다. Value list는 double precision의 리스트이다.

probe by plane primitive contour line body, when Gen Method is AutoLogScale or Auto

Gen Method	Line Num
8	16

Gen Method	Value
AutoLogScale	0x01
Auto	0x02
Value List	0x03

probe by plane primitive contour line body, when Gen Method is value list

Gen Method	Line Num	Value List
8	16	Line Num * 64

[그림 20] Probe by plane의 contour line presentation body

9) Transfer function

cmd.req Transfer function Body

Scalar Value Type
8 bit

Scalar Value Type	Value
Pressure	0x01
Velocity	0x02
Vorticity	0x03
Qcriteria	0x04

[그림 21] Transfer function 요청 데이터 필드

GIP 클라이언트는 load 요청에 대한 응답과 별도로 transfer function을 만들 수 있는 팔레트를 GIP 서버에 요청할 수 있다. 요청의 데이터 필드는 원하는 값의 형태이다. 그림 20은 로터 데이터에 대한 요청의 예이다. 이 값은 응용에 따라

달라 질 수 있다. 이를 결정하는 것은 GIP을 사용하는 응용 프로그램이다.

바. 요청에 대한 결과 전달 Cmd.res

cmd.res Body

Type	Len	Data
8	32	Len

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

[그림 22] 가시화 응답의 형태

1) Poly data 와 Image

Cmd.req이 요청에 대한 응답은 가시화의 결과이다. 이를 표현하기 위한 방법으로 크게 두 가지가 있다. 하나는 가시화의 결과를 geometry data 로 전달하여 실질적으로 화면에 렌더링 하는 것은 클라이언트에 맡기는 방법이다. 두 번째는 서버에서 렌더링까지 수행하여 이미지를 전송하는 방법이다. 첫 번째 방법은 사용자와의 상호 작용에 의한 view 의 변화에 따라 이루어지는 모든 변화를 클라이언트에서 수행하므로, 한번 데이터를 전송한 후 서버가 해 줄일은 없다. 두 번째 방법은 사용자의 시점이 이동할 때 마다 그 시점에서의 scene을 서버가 이미지로 전송하여야 하므로 사용자의 인터렉션을 모두 서버에 전달하여야 한다. 저성능의 클라이언트가 대용량의 데이터를 다룰 경우 두 번째 방법이 사용되기도 하지만, 대부분의 가시화 도구들은 실시간 인터렉션을 위해 첫 번째 방법을 사용한다. GIP은 cmd.res의 데이터 형태로 이 두 가지를 모두 지원하며, 그림 21의 Len에 명시된 길이만큼의 바이너리 형태로 클라이언트에 전송된다.

2) Status

Cmd.res의 두 번째 형태로 cmd.req의 요청에 대한 응답을 만드는 데 실패할 경우 cmd.res의 형태로 클라이언트에 전송된다. cmd.res의 type이 status일 경우 Len 은 4이며 오류코드는 아래의 표와 같다. 아래의 오류코드는 프로토콜에 대한 오류만 포함한 것으로 GIP 프로토콜을 사용하는 응용에 따라 오류코드를 추가할 수 있다.

[표 2] cmd.res status code

Error Name	Error Code Value
CMD_RES_STATUS_SUCCESS	0x00010001
CMD_RES_STATUS_FAIL	0x00010002
CMD_RES_STATUS_UNKNOWN_COMMAND	0x00010003
CMD_RES_STATUS_WRONG_DATA	0x00020001
CMD_RES_STATUS_WRONG_SCALAR_ID	0x00020002
CMD_RES_STATUS_UNKNOWN_PROBE_TYPE	0x00020003
CMD_RES_STATUS_NO_POLYS	0x00030001
CMD_RES_STATUS_WRONG_ANI_ID	0x00040001
CMD_RES_STATUS_DUPLICATE_ANI_ID	0x00040002
CMD_RES_STATUS_WRONG_TIMESTEP	0x00040003

3) Transfer function

cmd.res의 transfer function 은 cmd.req load에 대한 응답으로 클라인트에 전송된다. Load 명령으로 전체 데이터를 읽은 후, 전체 데이터에 대한 칼라 팔레트를 자동으로 생성하여 클라이언트에 전송함으로써, geometry 데이터를 클라이언트에서 렌더링할 때 transfer function 생성의 입력이 될 수 있도록 한다. 그림 19는 transfer function의 데이터 필드를 보여준다. scalar value type은 어떤 데이터 값으로 팔레트를 생성하였는지를 말한다. num 과 palette는 그 값에 대한 팔레트를 나타낸다.

cmd.res Transfer function Body

Scalar Value Type	Num	palette
8 bit	16 bit	Num *(3* 8+32)

Scalar Value Type	Value
Pressure	0x01
Velocity	0x02
Vorticity	0x03
Q-criteria	0x04

color

Key	R	G	B
32	8	8	8

Num means the number of elements in the palette and the type is integer.
cmd.req Transfer function palette is a list of color data

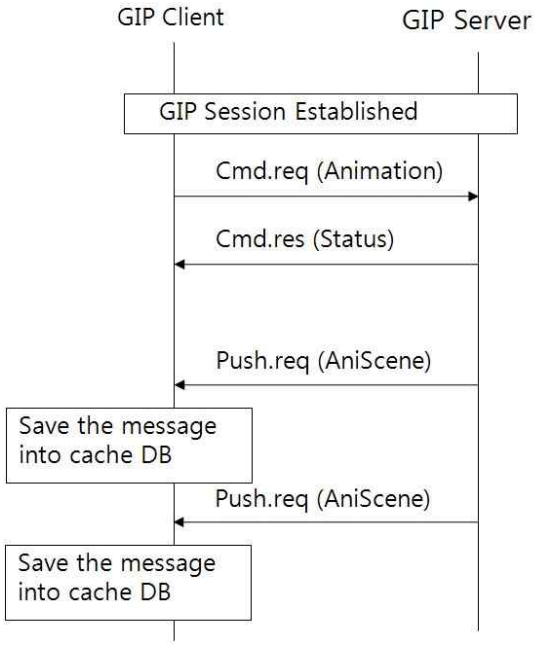
If the number of elements in palette is zero, GLORE sends transfer function by own palette

[그림 23] Transfer function 데이터 필드

사. 애니메이션 캐쉬 데이터의 전달 Push.req

1) GIP Cache Sequence Diagram

GIP 은 애니메이션 요청에 cache를 지원한다. Animation 요청의 데이터 필드에 cache num은 서버가 미리 만들어 전송해야할 캐쉬의 수를 의미한다. 클라이언트가 캐쉬 데이터 베이스를 유지할 경우 이 수는 캐쉬 데이터 베이스가 수용할 수 있는 캐쉬 데이터의 수가 될 수 있다. GIP는 cache를 위해 push.req를 지원한다. push.req는 캐쉬를 위해 클라이언트가 요청하지 않더라도 서버가 데이터를 클라이언트에 전송할 수 있는 방법을 제공한다. 클라이언트는 push.req를 받으면 그 데이터 형태가 Animation scene일 경우 해당 데이터를 캐쉬에 저장하여 사용할 수 있다. 그림 22는 클라이언트가 서버에 애니메이션을 요청하고 서버가 push.req를 이용해 생성된 데이터를 전송하는 과정을 보여준다.



[그림 24] GIP cache sequence

2) Push.req

push.req도 cmd.req 와 마찬가지로 기본적인 GIP packet의 데이터 필드이다. 따라서, push.req를 요청하기 전에 반드시, GIP 세션이 존재하여야 한다. push.req는 애니메이션의 캐쉬를 위해 사용되기도 하지만, 서버의 변화를 알리는 다른 용도로 사용할 수도 있다. 그림 23은 애니메이션을 위해 사용되는 push.req의 animation scene을 위한 데이터 필드를 보여준다. Animation id는 animation start 시 클라이언트에서 부여해준 animation id를 그대로 사용한다. Time step은 전송하는 데이터의 time step을 의미한다. data는 cmd.res의 poly data와 image 형태를 지원한다.

push.req Animation Scene Body

Type	Len	Body
8	64	Len

Type	Value
Animation Scene	0x01

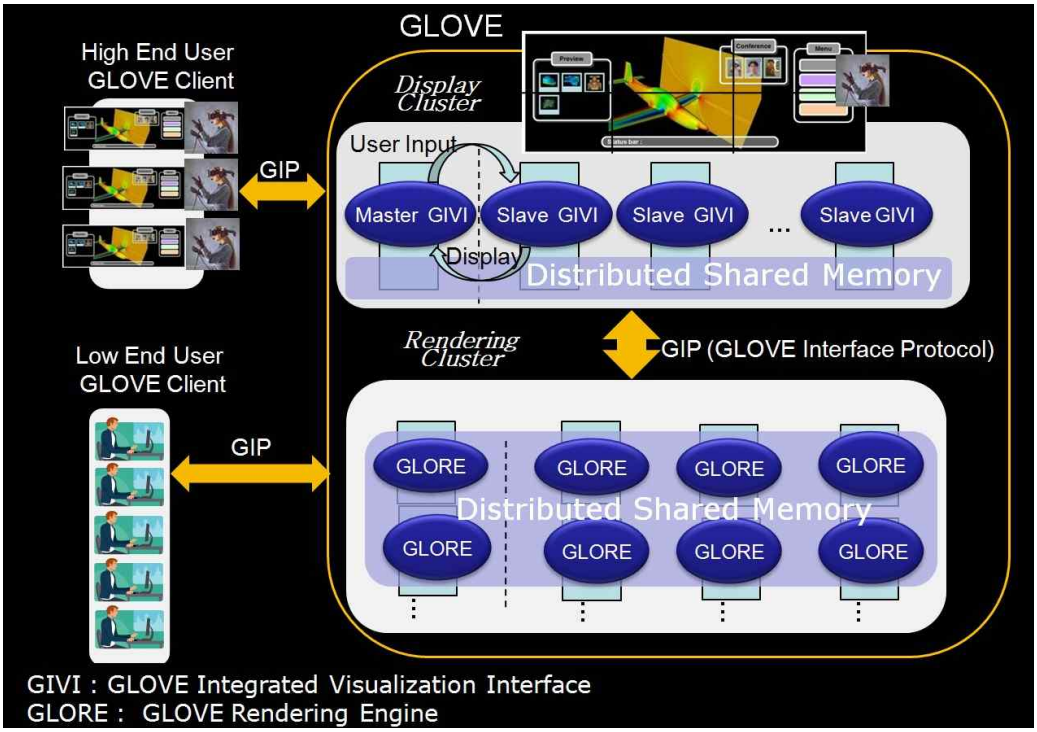
push.req Animation Scene Body

Animation id (integer)	Time step	Data
32	32	

[그림 25] push.req animation scene 데이터 필드

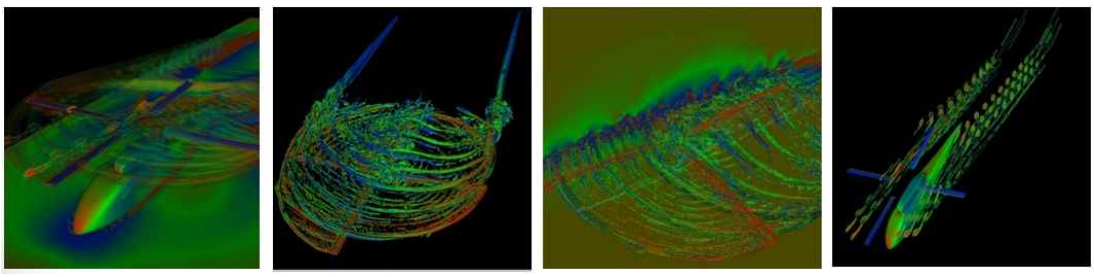
4. GLOVE 에서의 GIP

GIP은 GLOVE의 통신 프로토콜로 개발되었다. GLOVE (GLObal Virtual Environment for collaborative research)시스템은 KISTI에서 개발한, 대용량 데이터를 가상현실에서 구현하는 가시화 도구이다[3]. 그림 26은 GLOVE 에서의 GIP의 활용 예를 보여 준다. 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. GIP은 이러한 명세를 표현하고, 또한 그 결과를 다시 GLORE 에 전달한다. GIP은 명령의 전달 경로와 데이터의 전달 경로를 분리하여 프로토콜 자체로 데이터를 전달하지 않는다. GIP은 cmd.req에서 load 기능을 지원하여, 응용이 모든 데이터를 메모리에 올려 놓을 수도 있고, 자신만의 방법으로 데이터를 다룰 수 있도록 데이터의 처리는 응용에 맞는 방법으로 수행하도록 유연성을 제공한다. GLOVE는 GIP의 이러한 점을 활용하여 GDM을 사용하여 대용량 데이터를 분산공유메모리에서 처리하도록 하였다.



[그림 26] GLOVE 에서의 GIP

그림 27은 GLOVE에서 가시화 프로토콜인 GIP을 이용하여 가시화한 결과이다. 로터 데이터 시뮬레이션의 결과로 iso-surface와 probe by plane, scalar distribution의 다양한 결과를 GIP으로 구현하였다.



[그림 27] GIP을 활용한 GLOVE 로터 데이터 시뮬레이션 가시화 결과

5. 결론

이상에서 살펴 본 바와 같이 GIP은 응용 독립적 원격 가시화 렌더링 프로토콜이 가져야 하는 조건을 잘 만족 시킨다. 세션 관리를 통해 다중 클라이언트를 수용하고, 다양한 WAN 환경의 네트워크를 지원할 뿐 아니라, 기본적인 가시화 기능들을 잘 수행하도록 설계되었다. 또한, 시변환 데이터의 애니메이션 캐쉬 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, 렌더링 서버 개념의 서비스로 슈퍼컴퓨터의 또 다른 서비스 영역으로 활성화 시킬 수 있다.

6. 참고문헌

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

[2] COVISE, http://www.hlrs.de/covise

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

[4] S.Byron, "Virtual Reality in Scientific Visualization", Communications of t he ACM, 39(5):62-71, 1996.

[2] A.van Dam, A.S.Forsberg, D.H.Laidlaw, J.J.Laviola, R.M.Simpson, "Immer sive VR for Scientific Visualization: A Progress Report", IEEE Computer Gra phics and Applications, 2000.

[3] ParaView, http://paraview.org/New/index.html

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

[5] COVISE, http://www.hlrs.de/organization/av/vis /covise

[6] A.Bierbaum, C.Just, P.Hartling, K.Meinert, A.Baker, C.Cruz-Neira, "VRJu ggler: A Virtual Platform for Virtual Reality Application Development", Proce edings of IEEE VIrtual Reality, 2001.

[7] VRJuggler, http://www.vrjuggler.org

[8] CaveLib, http://www.mechdyne.com/integrated Solutions/software/products /CAVELib/CAVELib.html

[9] M.Schirki, A.Germdt, T.van Reimersdahl, T.Kuhlen, P.Adorneit, O.Lang, S.Pischinger, C.Bischof, "Vista FlowLib - A Framework for Interactive Visua lization and Exploration of Unsteady Flows in Virtual Environments", Eurogra

phics Workshop on Virtual Environments, 2003.

[10] Andrew Forsberg, Prabhat, Graff Haley, Andrew Bragdon, Joseph Levy, Caleb I.Fassett, David Shean, James W.Head III, Sarah Milkovich, Mark Duch aineau, "Adviser: Immersive Field Work for Planetary Geoscientists", IEEE C omputer Graphics and Applications, July 2006.