ISBN 000-00-000-0000-0

비정렬격자 데이터를 위한 GLOVE 데이터 변환기의 설계와 구현

한국과학기술정보연구원

저자소개

이세훈 (Sehoon Lee)

Korea Institute of Science and Technology Information
Division of Supercomputing
Senior Researcher
Email : sehooi@kisti.re.kr

차 례

1. 개요

1.1. 대용량 과학데이터 가시화

지구과학, 기상, 천체물리, 항공우주공학 등의 분야에서 복잡한 과학 현상을 분석하기 위해 사용되는 시뮬레이션 데이터의 분석 방법은 크게 그래프, 수치 데이터를 이용한 정량적 분석과 데이터 가시화를 통한 정성적 분석의 두 가지로 나누어진다. 이 중 데이터 가시화를 이용한 정성적 분석을 통하면 전체 데이터의 양상과 흐름을 알 수 있고, 시뮬레이션이 올바른 방향으로 진행되고 있는지에 대한 즉각적인 판단도 가능하다.

최근 컴퓨팅 자원의 급격한 성능 향상으로 시뮬레이션을 통해 더 정밀하고 규모가 큰 실험이 가능해 짐에 따라 시뮬레이션 결과 데이터의 양도 테라(tera), 페타(peta) 바이트 단위로 기하급수적으로 증가하는 추세이다. 이에 따라 과학 데이터 가시화 관련 기술도 병렬 데이터 처리, 고해상도 렌더링, 복잡한 데이터를 알아보기 쉽게 보여주기 위한 표현 방법 등을 지원하도록 발전해왔다.

1.2. GLOVE(GLobal Virtual reality visualization Environment for scientific simulation)

GLOVE[1]는 이러한 기술적 발전과 요구를 수용하기 위해 개발된 가시화 도구들 중 하나로, 고성능 컴퓨팅 환경에서 대용량 시뮬레이션 데이터를 효과적으로 분석하고 가시화하여 공유한다. 고해상도의 Tiled Display와 VR(Virtual Reality)환경은 응용 연구자에게 데이터 분석에 있어 직관적으로 다른 시각의 관점을 제공할 뿐만 아니라 다계층의 데이터 구조는 응용이 달라지더라도 최소한의 수정으로 응용에 맞는 하나의 새로운 인터페이스를 제공한다.

그림 1은 GLOVE 시스템의 구조를 보여준다. GLOVE는 사용자 인터페이스인 GIVI(GLOVE Integrated Visualization Interface)와 데이터 가공 및 렌더링을 담당하는 GLORE(GLOVE Rendering Engine)의 두 부분으로 이루어진다. GIVI는 가상현실 입출력 장치를 총괄하며, 사용자로부터 입력을 받아서 GLORE에 전달하고, GLORE의 실행 결과를 전달 받아서 화면에 출력한다. GLORE는 대용량 데이터의 가공 및 렌더링을 위한 GIVI의 서브 시스템이다.

GLOVE에서 GLORE와 GIVI는 서로 독립적으로 운영되는 클러스터에서 실행된다. 이러한 구조는 GLOVE의 인터페이스(GIVI)와 핵심 렌더링 기능(GLORE)을 물리적으로 분리할 수 있기 때문에 충분한 그래픽 처리능력을 갖추지 못한 시스템을 가지고 있는 사용자가 자신이 보유하고 있는 시스템에서 GIVI를 실행하고, 원격 장소의 고성능 렌더링 엔진인 GLORE를 이용해 대용량 데이터를 렌더링 할 수 있다.

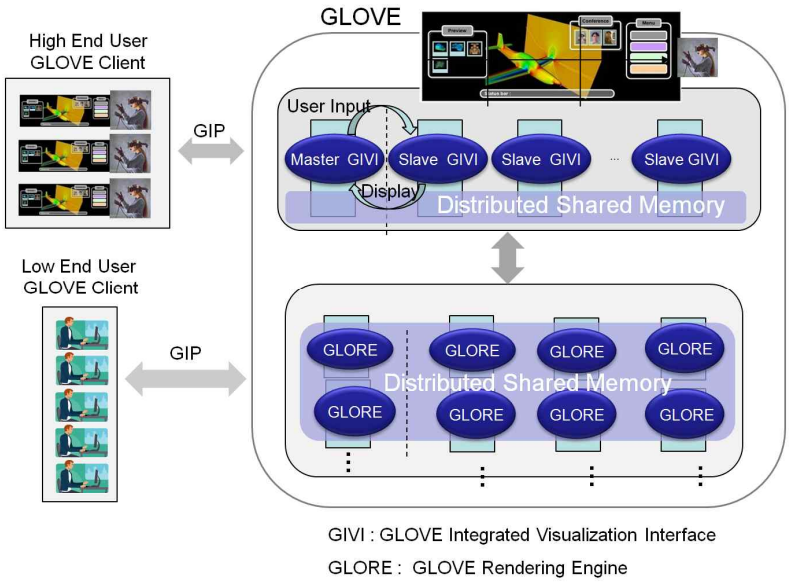

그림 1. GLOVE 시스템 구조

1.3. 데이터 변환의 필요성

계산과학분야의 사용자들은 다양한 포맷의 데이터를 사용한다. 비정렬격자 데이터를 저장하기 위해 주로 사용되는 포맷은 CGNS[2], EnSight[3], OpenFOAM[4] 등이 있다. 다양한 분야에서 활용하고 있는 비정렬격자 데이터 포맷을 지원하는 것은 GLOVE의 활용도를 높이고, 더 많은 사용자를 확보하는데 있어 필수적이다. 따라서 GLOVE 데이터 변환기는 다양한 비정렬격자 데이터 포맷을 지원할 수 있어야 한다.

2. GLOVE 데이터 변환기의 설계

2.1. GLOVE 데이터 변환 요구사항

2.1.1. 데이터 셋의 메타 정보 추출

GLOVE 데이터 포맷(이하 GLV 포맷)은 효율적인 병렬 입출력을 구현하기 위해 설계된 KISTI의 가시화 도구 GLOVE에서 사용하는 데이터 포맷이다. GLV 포맷은 데이터 셋에 대한 메타 정보를 meta.xml 파일에 관리한다. 메타 정보는 격자의 정렬/비정렬 여부, 타임스텝 정보, 격자가 시간에 따라 변하는 격자인지에 대한 정보 및 데이터 셋의 공통적인 특징을 포함한다. 표 1은 비정렬격자 GLV 포맷의 meta.xml 파일의 내용 구성을 보여준다. GLOVE 데이터 변환기는 CGNS, EnSight, OpenFOAM과 같은 비정렬격자 데이터 파일에서 필요한 메타 정보를 추출해 meta.xml 파일을 구성할 수 있어야 한다.

```
〈gdmMetaData〉
   〈gridType〉 USG 〈/gridType〉
   〈timestep〉 전체 타임스텝의 수 〈/timestep〉

   〈values〉
      〈value〉
         〈id〉 변수 ID 〈/id〉
         〈name〉 이름 〈/name〉
         〈vectors〉 변수를 구성하는 벡터 이름 (콤마로 구분) 〈/vectors〉
         〈type〉 데이터 타입 ( int | float | double ) 〈/type〉
      〈/value〉
      〈value〉
      ...
      〈/value〉
   〈/values〉

   〈elements〉
      〈element〉
         〈id〉 Element ID 〈/id〉
         〈name〉 이름 〈/name〉
         〈mesh〉
            〈dynamic〉 시간에 따른 격자의 변화 여부 ( true | false ) 〈/dynamic〉
            〈dimension〉 격자 차원 ( 1 | 2 | 3 ) 〈/dimension〉
            〈type〉 격자 데이터 타입 ( int | float | double ) 〈/type〉
         〈/mesh〉
         〈blockCount〉 Element가 포함하는 전체 블록 수 〈/blockCount〉
         〈values〉 Element가 포함하는 전체 변수 목록 (콤마로 구분) 〈/values〉
      〈/element〉
      〈element〉
      ...
      〈/element〉
   〈/elements〉
〈/gdmMetaData〉
```

표 1. 데이터 셋에 대한 meta.xml의 구성

2.1.2. GLOVE 데이터 디렉토리 구성

GLOVE는 데이터 로드(load)와 병렬 처리를 위해 유동해석 솔버(solver)가 만든 데이터를 변환하여 다음과 같은 디렉토리 구조로 저장한다. GLOVE 데이터 디렉토리는 데이터 셋의 메타 정보를 기술한 meta.xml 파일과 3개의 서브 디렉토리(mesh, data, cellInfo)를 가진다.

Mesh 디렉토리는 격자 데이터를 저장한다. 격자는 실험자가 구성한 물리적 단위인 Element로 구분되고, 각 Element는 시변환(time-varying) 데이터를 고려하여 다시 타임스텝 별로 구분된다. Element-타임스텝 별로 분할된 격자 데이터 블록을 저장한다. 격자는 실험의 종류에 따라 시간에 따라 변하기도, 그렇지 않기도 한다. 그리고 Element 별로 시간에 따라 격자가 변하는지 여부를 설정할 수 있다. 격자가 일정한 경우(static mesh)는 타임스텝 0만 유지하고, 그렇지 않은 경우(dynamic mesh)는 각 타임스텝마다 별도로 격자를 저장한다. 최하위 디렉토리에 있는 blkN.glv 파일은 실제 격자 데이터를 저장한 바이너리(binary) 파일이다.

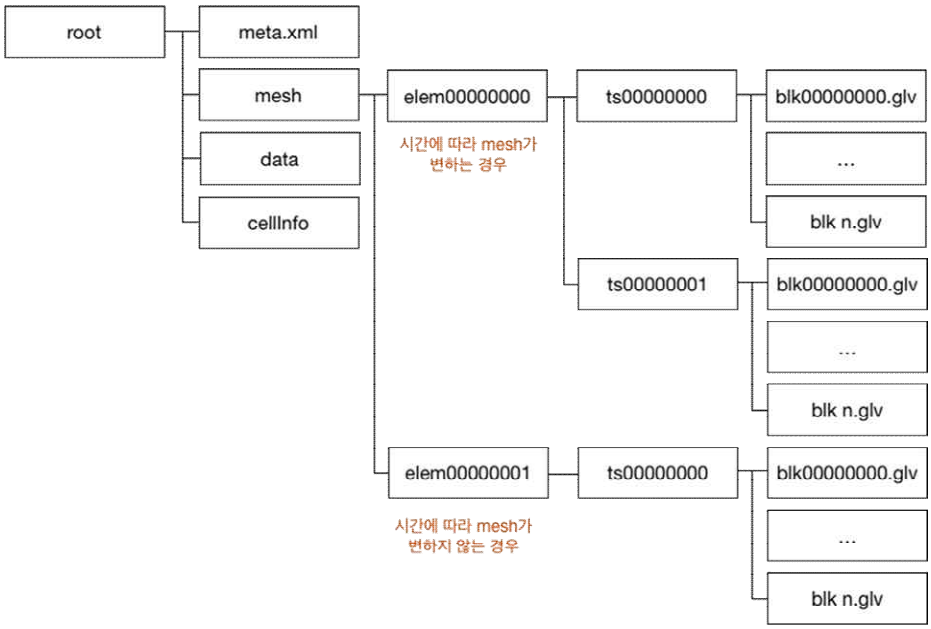

그림 2. Mesh 디렉토리의 구성

CellInfo 디렉토리는 비정렬격자 데이터에만 존재하며, 셀 ID, 타입, 위치 정보를 포함하고 있고, 그 구조는 Mesh 디렉토리와 동일하다.

Data 디렉토리는 각 격자점이 가지고 있는 물리값을 표현한 데이터를 저장한다. 병렬 입출력 효율을 높이기 위해 변수 별로 구분해 저장한다. Mesh나 CellInfo 디렉토리와는 다르게 변수-Element-타임스텝 순서로 구성되는데, 이는 일부 변수만 선택해 로드하거나, 2차 변수를 생성하는데 편리한 구조이기 때문이다.

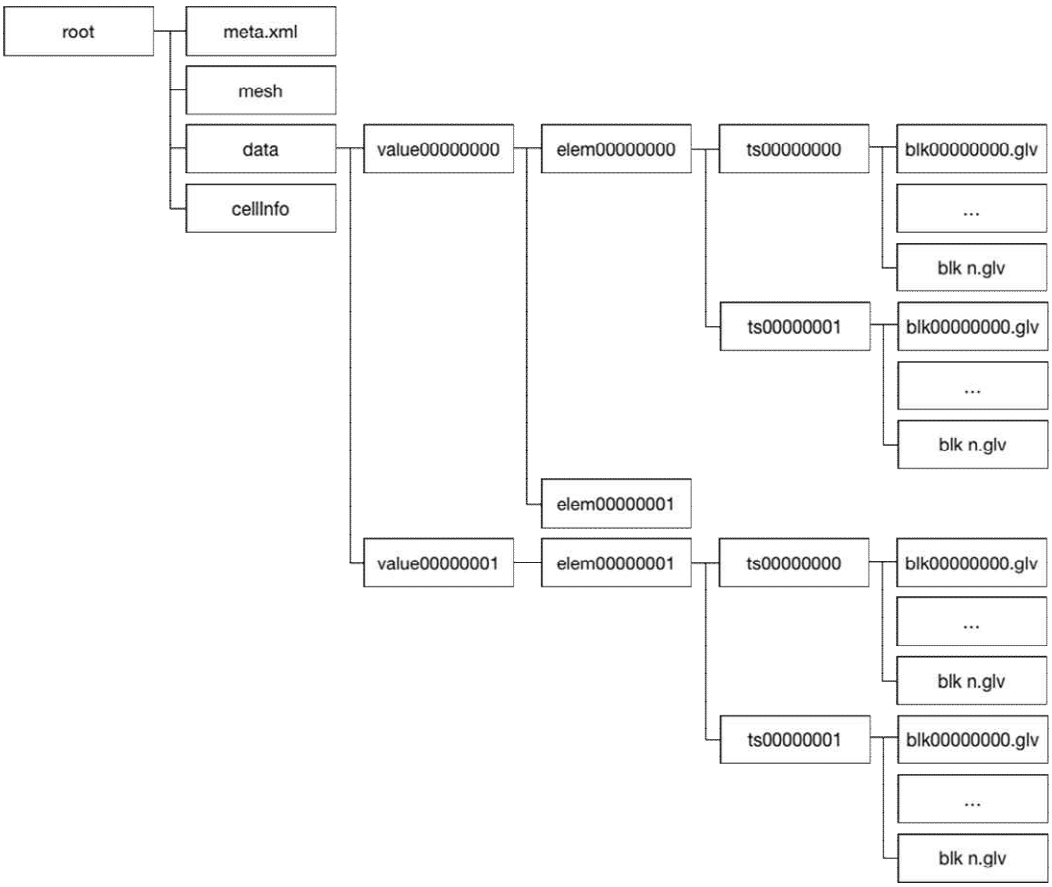

그림 3. Data 디렉토리의 구성

GLOVE 데이터 변환기는 CGNS, EnSight, OpenFOAM과 같은 비정렬격자 데이터 파일에서 필요한 메타 정보를 추출해 meta.xml을 구성하고, 이를 바탕으로 데이터 디렉토리를 구성할 수 있어야 한다.

2.1.3. 병렬 데이터 분할(decompose)

CFD 실험에서 생성되는 대용량 데이터는 일반적으로 하나의 타임스텝마다 수십 기가(giga) 바이트에 달하고, 100 타임스텝 이상의 실험을 진행하여 테라 바이트 단위의 데이터가 생성되는 경우도 빈번하다. 타임스텝 별로 데이터를 따로 저장하더라도 파일 하나의 크기는 수십 기가 바이트이므로 후처리 가시화 도구들이 병렬 처리를 하더라도 입출력에 많은 시간이 소요된다. 이를 해결하기 위해서는 파일 공유 시스템을 활용

한 병렬 데이터 입출력 방식을 사용할 수 있다. 이 경우 병렬 데이터 입출력의 효율은 클러스터의 모든 노드가 얼마나 공평하게 데이터를 읽을 수 있는가에 따라 결정된다. 따라서 대용량의 데이터를 노드 수에 맞게 공평하게 분할(decompose)할 수 있는 방법이 필요하고, 변환에 소요되는 시간을 줄이기 위해 이를 병렬로 처리할 수 있는 방법도 필요하다.

2.1.4. 데이터 변환의 범위 결정

GLOVE 데이터 변환기는 타임스텝이 많은 시변환 데이터를 한번에 자동으로 변환하는 것을 목표로 한다. 시간이 오래 걸리는 CFD 해석 실험과 마찬가지로 변환 시간이 긴 경우, 클러스터 시스템의 장애나 기타의 이유로 변환을 도중에 중단할 수 있다. 문제를 해결한 후, 처음부터 다시 변환을 시작하는 것은 비효율적이므로 중단된 시점부터 변환을 재개할 수 있는 기능이 필요하다. 이를 위해 변환할 타임스텝의 범위를 지정할 수 있도록 한다.

또, 해석 실험의 결과 데이터는 많은 변수를 저장하고 있는 경우가 대부분이다. 하지만 실제 가시화 단계에서는 그 중 일부만을 사용하는 경우가 많기 때문에 모든 변수를 다 변환하는 건 비효율적이다. 사용자가 원하는 변수만 골라서 변환할 수 있는 기능을 제공해야 한다.

2.1.5. 다양한 비정렬격자 데이터 포맷 지원

CFD 실험 결과는 여러 종류의 데이터 포맷으로 저장된다. 비정렬격자 데이터를 저장하기 위해 주로 사용되는 포맷은 CGNS, EnSight, OpenFOAM 등이 있다. 따라서 GLOVE 데이터 변환기는 다양한 비정렬격자 데이터 포맷을 지원할 수 있어야 한다.

2.1.6. 벡터 재결합

실험에서 생성된 데이터에서 벡터 변수의 구성요소(component)를 각각의 스칼라 변수로 저장해놓은 경우가 있다. 후처리 가시화를 위해 이를 벡터로 다시 합치고 싶은 경우, 변환기에서 이를 처리할 수 있어야 한다.

2.1.7. Element 병합

비정렬격자 데이터의 격자는 수십 개의 Element로 구성되어 있는 경우가 많다. 이 때문에 후처리 가시화 단계에서 조작에 어려움을 경우가 발생한다. 따라서 사용자가 원하는대로 기존 Element를 병합해서 새로운 Element로 만들 수 있어야 한다.

2.2. 설계

2.2.1. 데이터 변환 순서도

데이터 변환은 2.1의 GLOVE 요구사항을 바탕으로 다음과 같은 절차로 이루어진다. 그림 4는 데이터 변환 과정을 보여준다. 먼저 변환기는 데이터 변환에 대한 전반전인 정보를 gcm 파일을 읽어서 확보한다. gcm 파일은 원본 데이터 포맷, 격자 종류, 데이터의 위치 등을 담고 있으며, 이에 대한 자세한 내용은 3.1에서 다시 설명한다.

다음은 첫 번째 타임스텝의 데이터 파일을 읽어 필요한 메타 정보를 추출한다. 이 작업은 마스터 프로세스 단독으로 수행하며, 변수 정보 및 Element의 격자 정보를 추출하고, 격자 데이터 크기를 바탕으로 나눠질 블록 수도 미리 계산한다.

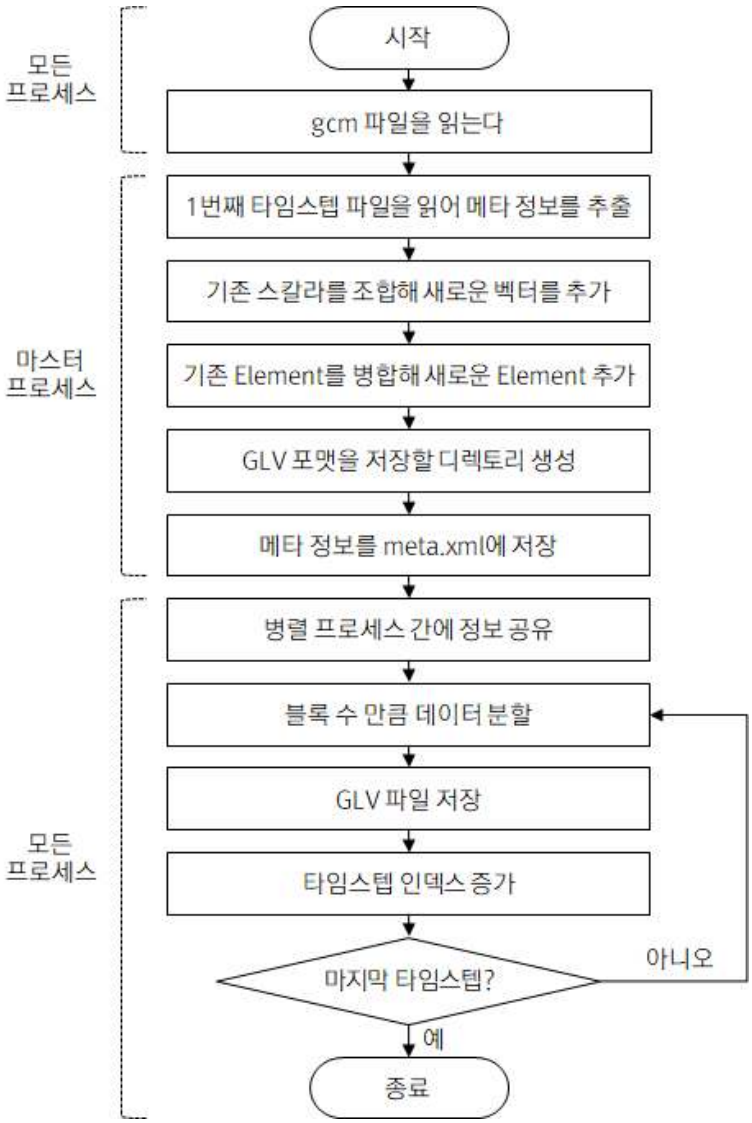

그림 4. GLOVE 데이터 변환 흐름도

그리고 사용자가 원하는 경우(gcm 파일에서 설정), 기존 스칼라 변수들을 조합해 새로운 벡터 변수를 추가하거나, 기존 Element를 병합해 새로운 Element를 추가한다. 모든 메타 정보가 구성되면 이를 바탕으로 GLV 포맷을 저장할 GLOVE 디렉토리를 생성하고, meta.xml을 저장한다.

마스터 프로세스가 메타 정보 구성을 마치면 모든 프로세스 간에 정보를 공유한다. 그리고 나서 모든 타임스텝 파일을 차례로 읽어, 미리 계산한 블록 수로 데이터를 분할한 후 이를 GLOVE 디렉토리에 GLV 포맷으로 저장한다.

2.2.2. 데이터 흐름도

데이터 변환이 시작되면 프로세스들 간의 데이터의 흐름은 그림 5과 같다. 먼저 Executor는 decompose.cnf에서 노드 리스트를 읽고, 병렬 MPI 프로세스들을 실행한다. 모든 Decomposer 프로세스는 gcm 파일을 공통으로 읽는다. 그리고 마스터 프로세스인 Decomposer 0는 첫 번째 타임스텝의 데이터 파일을 읽고 구성한 메타 정보를 다른 프로세스들과 공유한다. Decomposer들은 분할한 데이터 블록을 GLOVE 디렉토리에 GLV 포맷으로 저장한다.

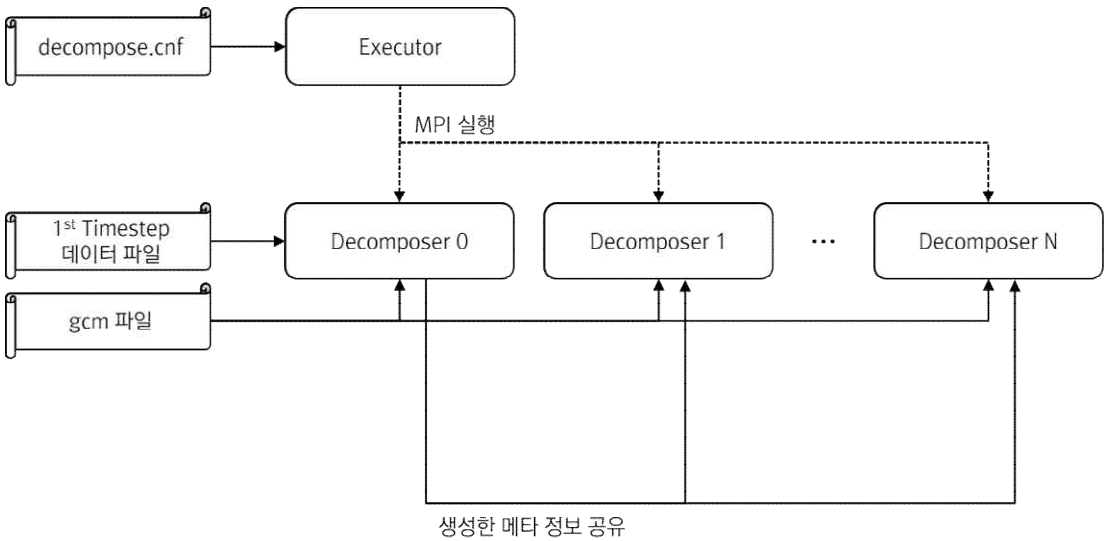

그림 5. GLOVE 데이터 변환을 위한 데이터 흐름도

3. GLOVE 데이터 변환기의 구현

3.1. 데이터 변환을 위한 gcm 파일

사용자는 데이터 변환을 위해 데이터 셋에 대한 전반전인 정보를 gcm(GLOVE converter metadata) 파일을 통해 제공해야 한다. gcm 파일은 변환 전 데이터가 가지고 있지 않은 메타 정보를 사용자가 직접 기술하기 위한 목적으로 사용되며, 그 내용은 크게 데이터, 변수, 벡터 변수 생성, Element 병합, 타임스텝, 비정렬격자의 6가지 섹션으로 구성된다. 다음은 데이터 변환을 위한 gcm 파일의 예제이다.

```
# 데이터
data.format=ensight
data.grid.type=usg
data.source=/tmp/ensight
data.target=/tmp/glv

# 변수
value.pressure.type = double
value.pressure.name=Pressure
value.velocity.name=Velocity
value.density.name=Density
value.temperature.name=Temperature
value.x_velocity.ignore=true
value.y_velocity.ignore=true
value.z_velocity.ignore=true

# 벡터 변수 생성
vector.Vorticity=x_vorticity,y_vorticity,z_vorticity

# Element 병합
elementmerge.Merged=wall,farfield,interior

# 타임스텝
timestep.start=1
timestep.end=5
timestep.inc=1

# 비정렬격자
usg.filename.pattern=%Ts.encas
usg.mesh.dynamic=true
```

표 2. 데이터 변환을 위한 gcm 파일 예제

3.1.1. 데이터 정보

변환 전 데이터와 변환 후 데이터에 대한 정보를 기술한다.

항목	내용
data.format	• vtk | cgns | ensight | ensight.ascii | openfoam
	• 변환 전 데이터의 포맷 • ensight = EnSight 바이너리 • ensight.ascii = EnSight ASCII
data.grid.type	• sg | usg
	• 변환 전 데이터의 격자의 종류 • sg : 정렬격자, usg : 비정렬격자
data.source	• 변환 전 데이터의 디렉토리
data.target	• 변환 후 데이터의 디렉토리 • 동일한 이름의 디렉토리가 있는 경우, 뒤에 붙는 숫자를 증가하며 자동으로 디렉토리가 생성된다. 예를 들어, data라는 디렉토리가 존재한다면 data.00, data.01, … 과 같이 자동으로 생성된다.

3.1.2. 변수 정보

변수에 대한 정보를 기술한다.

항목	내용
value.{NAME}.type	• int | float | double
	• 변수의 타입을 변경하고 싶은 경우에 설정 • {NAME}은 변수 이름을 의미
value.{NAME}.name	• 변수의 이름을 변경하고 싶은 경우에 설정
value.{NAME}.ignore	• true | false
	• 변환에서 해당 변수를 제외할 때 설정 • true로 설정하면 해당 변수는 변환 결과에 포함되지 않음

3.1.3. 벡터 변수 생성

벡터 변수 생성에 대한 정보를 기술한다.

항목	내용
vector.{NAME}	• 스칼라 변수 이름의 리스트(콤마로 구분)
	• 기존의 스칼라 변수를 합성해 벡터 변수를 새로 생성 • {NAME}은 벡터 변수 이름을 의미 • 스칼라 변수로만 합성 가능 • 새 벡터 변수가 추가되면 기존 스칼라 변수들은 변환 결과에 포함되지 않음 • value.{NAME}.name으로 수정하기 전 이름을 사용해야 함

3.1.4. Element 병합

새로운 Element를 병합하기 위한 정보를 기술한다.

항목	내용
elementmerge.{NAME}	• Element 이름의 리스트(콤마로 구분)
	• 기존 Element들을 병합해 새로운 Element를 생성 • {NAME}은 Element 이름을 의미 • 기존 Element들은 변환 결과에 포함되지 않음

3.1.5. 타임스텝 정보

타임스텝에 대한 정보를 기술한다. 타임스텝을 기술하는 방식은 start-end-inc 방식과 리스트 방식 중에 하나를 선택한다.

항목	내용
timestep.start	• 변환에 포함되는 첫 번째 타임스텝
timestep.end	• 변환에 포함되는 마지막 타임스텝
timestep.inc	• 변환에 포함되는 타임스텝의 증가분(increment) • start와 end가 동일하면 inc에 관계없이 하나의 타임스텝만 선택
timestep.list	• 타임스텝의 리스트(콤마로 구분)
	• 변환에 포함되는 타임스텝을 리스트로 직접 설정한다. • 예) timestep.list=17,25,38

3.1.6. 비정렬격자 정보

비정렬격자에 대한 정보를 기술한다.

항목	내용
usg.filename.pattern	• 변환 전 데이터 파일명의 패턴 • %T : 타임스텝으로 치환 - 예) F16.1.case → F16.%T.case • %0[길이]T : 부족한 자릿수를 왼쪽부터 0으로 채움 - 예) F16.01.case → F16.%02T.case • 타임스텝이 필요 없는 데이터의 경우 %T를 사용하지 않을 수 있다. %T가 없으면 3.1.5. 타임스텝 정보는 무시된다.
usg.mesh.dynamic	• true | false
	• 시간에 따른 격자 데이터의 변화 여부 • true : 시간에 따라 격자 데이터가 변하는 dynamic • false : 시간에 따라 격자 데이터가 동일한 static • 격자가 일정한 경우(static mesh)는 변환 전 데이터가 격자 데이터를 타임스텝 별로 존재해도 첫 번째 타임스텝의 격자 데이터만 변환

3.2. gdm::converter::Properties 클래스

3.1에서 설명한 GLOVE 데이터 변환을 위한 gcm 파일을 파싱(parsing)한다. std::map을 상속한 클래스로, gcm 파일의 "key=value" 형식을 파싱하여 이를 보관한다.

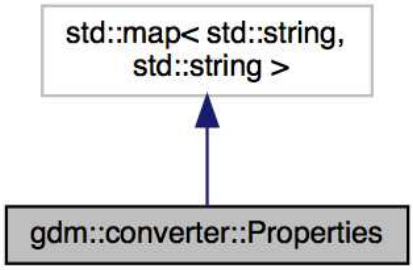

그림 6. gdm::converter::Properties 클래스의 협업 다이어그램

void load(std::istream &in)
• 주어진 입력 스트림으로부터 gcm 파일을 읽어 파싱 후 (key, value)로 유지
int getInt(const std::string &property, int defaultValue)
• 주어진 property가 가지는 int 타입의 값을 반환, 없으면 defaultValue를 반환
bool getBool(const std::string &property, bool defaultValue)
• 주어진 property가 가지는 bool 타입의 값을 반환, 없으면 defaultValue를 반환
std::string getBool(const std::string &property, const char *defaultValue)
• 주어진 property가 가지는 string 타입의 값을 반환, 없으면 defaultValue를 반환
static std::string trim(const std::string &s)
• 주어진 문자열 앞뒤로 공백을 제거

3.3. gdm::converter::Config 클래스

3.1에서 설명한 GLOVE 데이터 변환을 위한 gcm 파일에서 읽은 정보를 유지 및 관리한다. gdm::converter::Properties 클래스를 mProperties라는 멤버 변수로 가지며 이를 통해 파싱한 gcm 파일 정보를 추출한다. 이로부터 GLOVE 데이터 변환에 필요한 정보를 추가적으로 구성한 후 이를 외부에 인터페이스를 통해 제공한다.

그림 7. gdm::converter::Config 클래스의 협업 다이어그램

int Configure(string path)
• 주어진 path에 있는 gcm 파일을 읽어 데이터 변환에 필요한 정보를 구성
fs::path GetConfigFilePath()
• gcm 파일의 경로를 반환
string GetDataFormat()
• 변환 전 데이터의 포맷을 반환
int GetNextMergedElementId()
• 병합 Element를 생성하는 경우 Element ID를 발급
fs::path GetSourceDir()
• 변환 전 데이터의 디렉토리를 반환
string GetSourceFilePath(string pattern, int block, int timestep, int &errCode)
• 변환 전 데이터 파일명의 패턴에 타임스텝을 적용한 실제 데이터 파일의 전체 경로를 반환(단, OpenFOAM은 파일명 대신 타임스텝 값을 반환)
fs::path GetTargetDir () const
• 변환 후 GLV 데이터의 디렉토리를 반환
string GetValueNewName(string name)
• 주어진 변수의 변경 후 이름을 반환
void SetFirstMergedElementId(int eid)
• 병합 Element의 첫 번째 ID를 설정
void SetTargetDir(fs::path path)
• 변환 후 GLV 데이터의 디렉토리를 설정

3.4. gdm::converter::Common 클래스

데이터 포맷, 격자의 정렬/비정렬 여부에 관계없이 GLOVE 데이터 변환을 위한 클래스에서 공통으로 활용하는 기능을 관리한다. gdm::converter::Config 클래스를 mConfig라는 멤버 변수로 가지며 이는 생성자 내부에서 Config.Configure()를 호출하여 변환 정보를 초기에 구성한다.

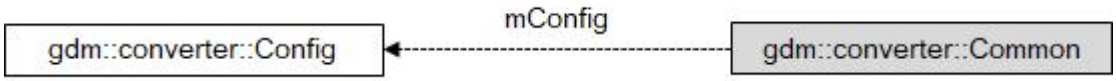

그림 8. gdm::converter::Common 클래스의 협업 다이어그램

int BuildMetaData()
• GLV 데이터를 위한 MetaData를 구성한다. 1) 첫 번째 타임스텝 데이터로부터 변수/격자 정보를 추출하고, 2) 기존 스칼라 변수들을 조합해 새로운 벡터 변수를 생성하고, 3) 기존 Element를 병합해 새로운 Element를 추가한다.
ConfigPtr GetConfig()
• GLOVE 데이터 변환 정보를 구성하고 있는 Config 객체의 포인터를 반환
static int WriteArray(FILE* fp, vtkDataArray* array)
• vtkDataArray를 파일로 write
static int SaveMesh(const Config &config, int element, int ts, int block, vtkDataSet* ds)
• 격자 정보 및 데이터를 파일로 write
static int SaveCellInfo(const Config &config, int element, int ts, int block, vtkDataSet* ds)
• 셀 정보 및 데이터를 파일로 write
static int SaveData(const Config &config, int element, int ts, int block, vtkDataSet* ds)
• 변수 데이터를 파일로 write
static void ExtractValueInfo(ConfigPtr config, MetaDataPtr meta, int element, vtkPointData* pd)
• VTK 데이터로부터 변수 정보를 추출하여 MetaData에 반영
virtual vtkSmartPointer<vtkDataObject> GetReaderOutput(string srcfile) = 0
• 비정렬격자 데이터를 읽어 VTK 데이터로 반환(인터페이스로 실제 구현은 각 데이터 포맷 별로 되어 있음)

virtual int ExtractValueMeshInfo() = 0
• 변환 전 데이터를 읽어 격자 및 변수 정보를 추출하여 MetaData를 구성(인터페이스로 실제 구현은 정렬격자/비정렬격자 별도로 되어 있음. 비정렬격자는 gdm::converter::UnstructuredGridLib 클래스에 구현)
void AddMergedElements()
• 기존 Element를 병합한 새로운 Element를 추가
void AddVectors()
• 기존 스칼라 변수들을 조합해 새로운 벡터 변수를 생성
int CreateTargetDir()
• 변환 후 GLV 데이터의 디렉토리를 생성한다. 동일한 이름의 디렉토리가 있는 경우, 뒤에 붙는 숫자를 증가하며 자동으로 디렉토리가 생성된다. 예를 들어, data라는 디렉토리가 존재한다면 data.00, data.01, … 과 같이 자동으로 생성된다.

3.5. gdm::converter::UnstructuredGridLib 클래스

변환기와는 별도로 GLOVE 데이터 관리자(이하 GDM)에서 Import 기능을 위해 런타임(runtime)에 필요한 기능만 따로 관리한다. 라이브러리 형태로 GLOVE 컴파일을 위해 제공된다.

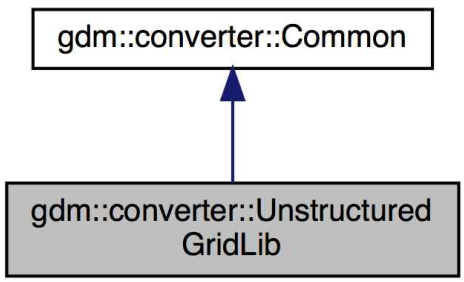

그림 9.
gdm::converter::UnstructuredGridLib
클래스의 협업 다이어그램

vtkSmartPointer<vtkPointSet> CellDataToPointData(vtkSmartPointer<vtkPointSet> ps)
• Point 데이터 없이 Cell 데이터만 있는 경우 이를 PointData로 변환
virtual int ExtractValueMeshInfo()
• 변환 전 비정렬격자 데이터를 읽어 격자 및 변수 정보를 추출하여 MetaData를 구성한다. 이 과정에서 Element를 분할할 블록 수도 계산한다.
int GetElementBlockCount(long long int elementSize)
• 주어진 Element 크기로 분할할 블록 수를 계산하여 반환 • $Element\,블록수 = \left\lceil \frac{Element\,크기}{최대\,블록\,크기} \right\rceil$ • 단, Element 블록 수는 MPI 프로세스 수를 넘을 수 없다. 넘으면 자동으로 MPI 프로세스 수로 설정된다.

3.6. gdm::converter::UnstructuredGrid 클래스

비정렬격자 데이터를 GLV 포맷으로 변환하는 전체 프로세스를 구현한다. Element 크기가 정해진 값을 넘으면 여러 블록으로 분할하게 되는데 이를 위해 병렬 MPI 프로세스로 실행된다.

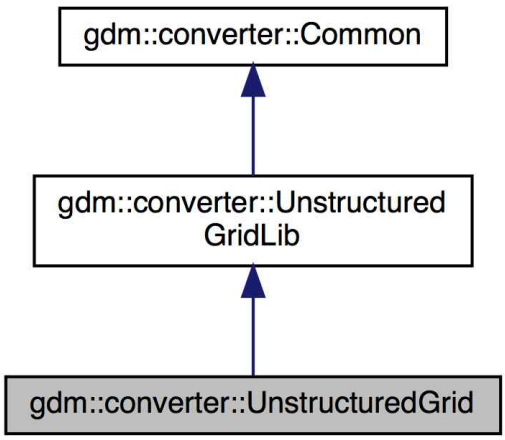

그림 10. gdm::converter::UnstructuredGrid 클래스의 협업 다이어그램

void Execute()
• GLOVE 데이터로 변환하는 전체 프로세스를 구현한다. GLV 데이터를 위한 MetaData를 구성, 변환 후 GLV 데이터의 디렉토리를 생성하고, MetaData를 meta.xml 파일에 저장한 후 병렬 프로세스 간에 정보를 공유한다. 타임스텝을 차례대로 증가시키며 변환을 진행하고, 모든 변환이 끝나면 수정된 변수명을 meta.xml에 반영 후 종료한다.
int Decompose(map<string, int> &ignoredValues, int ts, int eid, vtkSmartPointer<vtkPointSet> ps)
• 미리 계산한 Element의 블록 수만큼 vtkPointSet 데이터를 분할한다. 내부적으로 vtkDistributedDataFilter를 사용해 데이터를 분할한다.
int CreateMetaXML()
• GLV 데이터를 위한 MetaData를 구성하고, 변환 후 GLV 데이터의 디렉토리를 생성한 후 MetaData를 meta.xml 파일에 저장한다.

3.7. gdm::converter::Executor 클래스

GLOVE 데이터 변환 프로세스를 MPI로 병렬 실행한다. 병렬 실행하고자 하는 노드 리스트는 decompose.conf에 기술된다.

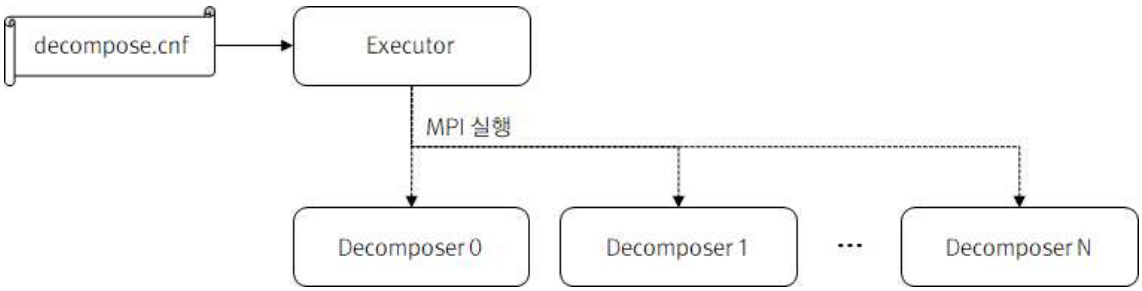

그림 11. gdm::converter::Executor의 MPI 병렬 프로세스 실행

static void Execute(const char* cmd, char* configFile, const vector<string> &hosts)
• 주어진 이름을 가진 실행 파일을 주어진 노드에서 MPI 병렬 프로세스로 실행

3.8. 클래스 관계도

GLOVE 데이터 변환을 위해 전체 클래스의 관계도는 다음과 같다. 각 데이터 포맷 고유의 기능은 개별 클래스에서 관리하고, 공통의 부분은 gdm::converter::UnstructuredGrid와 gdm::converter::UnstructuredGridLib에서 공유한다.

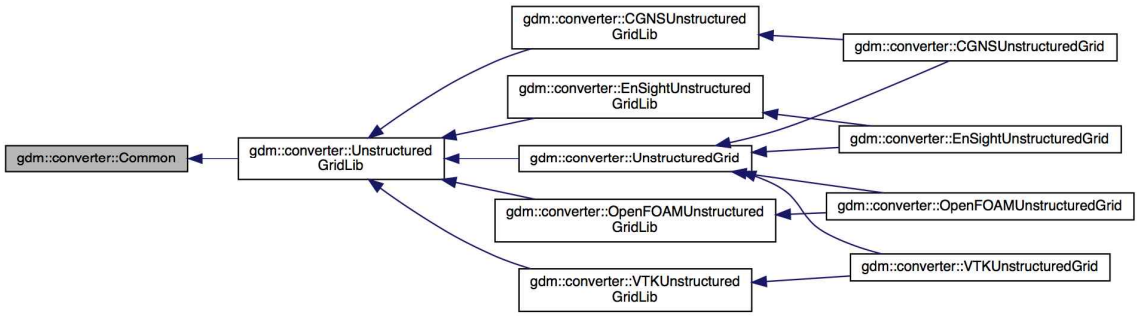

그림 12. gdm::converter::Common 클래스의 상속 다이어그램

4. CGNS 변환

4.1. CGNS 데이터 포맷

CGNS(CFD General Notation System)는 CFD 분석 데이터의 입/출력 표준화를 목표로 배포되고 있는 라이브러리이다. 기본적으로 정적/동적 라이브러리를 모두 제공하며, 내부적인 저장 방식은 HDF5/ADF(Advanced Data Format)를 사용한다. 최신 버전에서는 병렬 입출력을 지원하는 HDF5를 기본으로 지원한다. CGNS 데이터는 다음 그림과 같이 계층적 구조로 이루어진다.

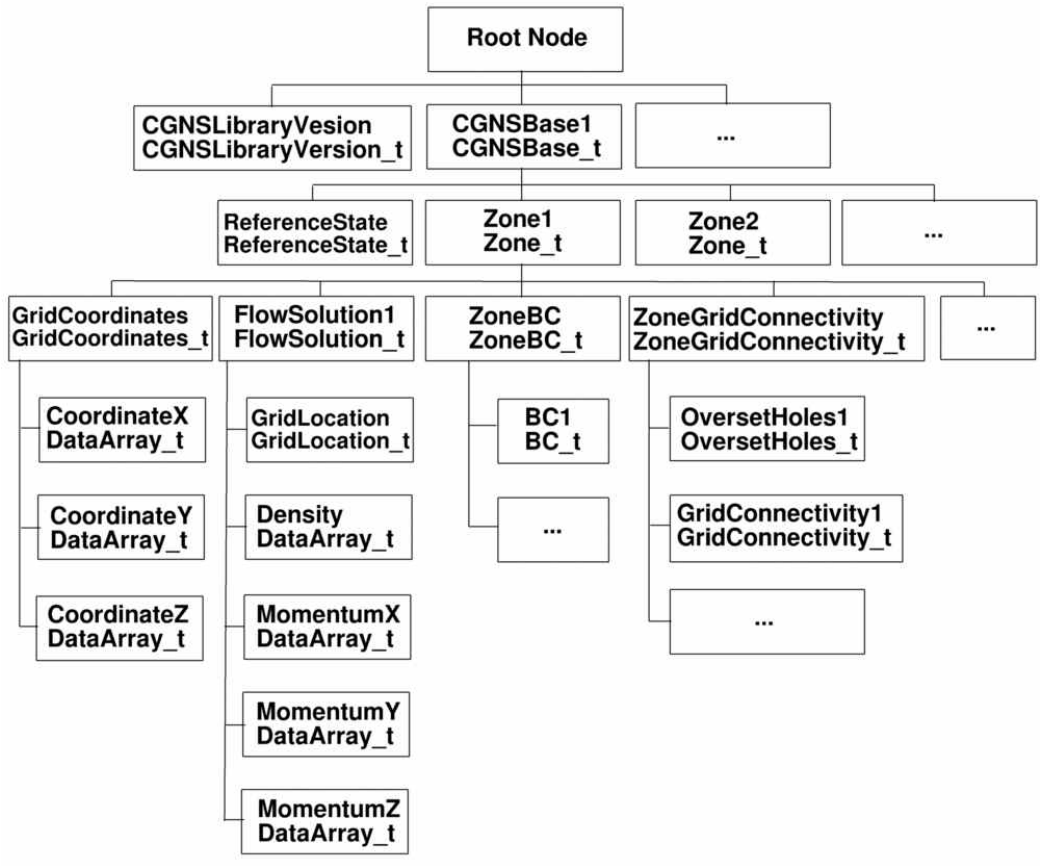

그림 13. CGNS 데이터의 구조

4.2. gdm::converter::CGNSUnstructuredGridLib

변환기와는 별도로 GLOVE 데이터 관리자(이하 GDM)에서 CGNS 데이터 Import 기능을 위해 런타임(runtime)에 필요한 기능만 따로 관리한다. 라이브러리 형태로 GLOVE 컴파일을 위해 제공된다.

virtual vtkSmartPointer<vtkDataObject> GetReaderOutput(string srcfile)
• CGNS 비정렬격자 데이터를 읽어 VTK 데이터로 반환

4.3. gdm::converter::CGNSUnstructuredGrid

병렬 MPI 프로세스로 CGNS 비정렬격자 데이터를 GLV 포맷으로 변환한다.

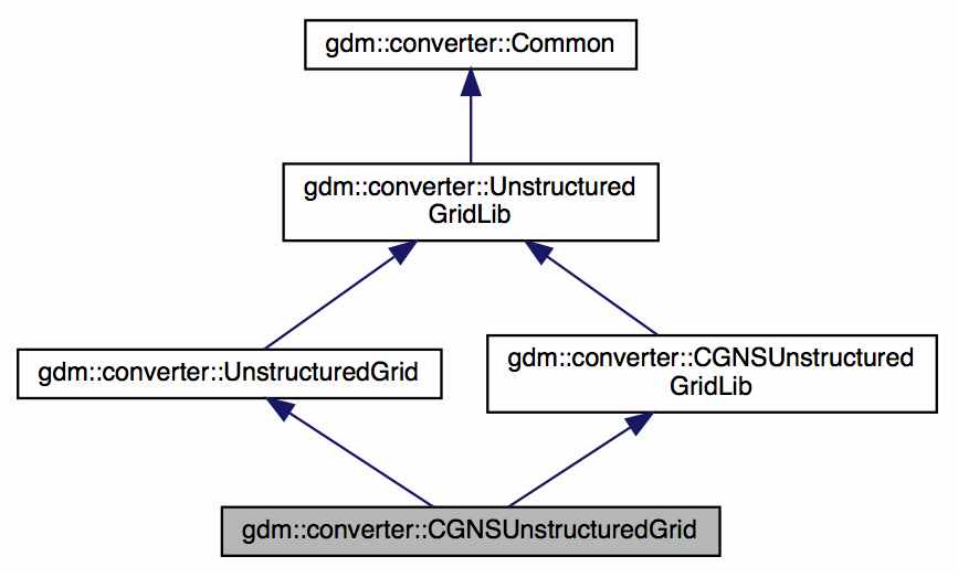

그림 14. gdm::converter::CGNSUnstructuredGrid 클래스의 협업 다이어그램

virtual vtkSmartPointer<vtkDataObject> GetReaderOutput(string srcfile)
• CGNS 비정렬격자 데이터를 읽어 VTK 데이터로 반환 • gdm::converter::CGNSUnstructuredGridLib::GetReaderOutput() 호출

5. EnSight 변환

5.1. EnSight 데이터 포맷

EnSight는 미국 CEI(Computational Engineering International) 사에서 개발한 CFD 가시화 도구이다. EnSight는 데이터에 대한 메타 정보를 case 파일에서 다루고, 이는 변수의 이름 및 파일명 뿐만 아니라 타임스텝에 대한 정보도 유지한다. 그러나 현재는 case 파일의 타임스텝 정보는 없다 가정하고, 각 타임스텝마다 case 파일이 따로 있는 경우만 지원한다.

다음은 EnSight gold case file의 예를 보여준다. case 파일은 크게 네 부분으로 구성된다. 첫 번째 FORMAT은 데이터 포맷의 버전을 알려준다. 두 번째 GEOMETRY는 격자점의 geometry에 대한 정보를 기술한다. VARIABLES는 값을 표현하는 변수들에 대한 정보이다. scalar per node, scalar per element, vector per node, vector per element의 네 가지 형태가 존재한다. "per node"는 nodal value이며, "per element" 는 cell-centered value를 의미한다. 각 변수의 특성을 scalar와 vector 변수의 위치에 따라 분류하여 기술할 수 있다. 다음에 나오는 숫자는 time set을, 그 다음은 변수의 이름, 마지막은 변수를 저장한 파일의 이름이다. 네 번째 TIME은 타임스텝에 대한 정보를 기술한다. time set은 하나 이상 가능하다.

```
FORMAT
type: ensight gold
GEOMETRY
model: 2 sedan_4m-1.geo
VARIABLE
scalar per node: 1 pressure                    sedan_4m-1****.scl1
scalar per node: 1 velocity_magnitude          sedan_4m-1****.scl2
scalar per node: 1 x_velocity                  sedan_4m-1****.scl3
scalar per node: 1 y_velocity                  sedan_4m-1****.scl4
scalar per node: 1 z_velocity                  sedan_4m-1****.scl5
vector per node: 1 velocity                    sedan_4m-1****.vel
TIME
time set: 1 Model
number of steps: 20
filename start number: 10
filename increment: 10
time values: 1.17035e-02 1.18035e-02 1.19035e-02 1.20035e-02 1.21035e-02 1.
22035e-02 1.23035e-02 1.24035e-02 1.25035e-02 1.26035e-02 1.27035e-02 1.28
034e-02 1.29034e-02 1.30034e-02 1.31034e-02 1.32034e-02 1.33034e-02 1.3403 4e-02
1.35034e-02 1.36034e-02
time set: 2 Model
number of steps: 1
time values: 1.17035e-02
```

5.2. gdm::converter::EnSightUnstructuredGridLib

변환기와는 별도로 GLOVE 데이터 관리자(이하 GDM)에서 EnSight 데이터 Import 기능을 위해 런타임(runtime)에 필요한 기능만 따로 관리한다. 라이브러리 형태로 GLOVE 컴파일을 위해 제공된다.

virtual vtkSmartPointer<vtkDataObject> GetReaderOutput(string srcfile)
• EnSight 비정렬격자 데이터를 읽어 VTK 데이터로 반환

5.3. gdm::converter::EnSightUnstructuredGrid

병렬 MPI 프로세스로 EnSight 비정렬격자 데이터를 GLV 포맷으로 변환한다.

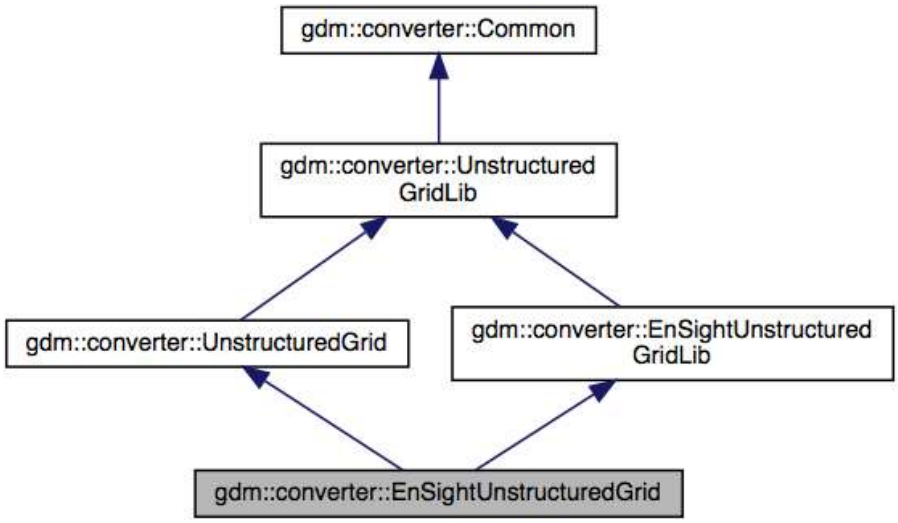

그림 15. gdm::converter::EnSightUnstructuredGrid 클래스의 협업 다이어그램

virtual vtkSmartPointer<vtkDataObject> GetReaderOutput(string srcfile)
• EnSight 비정렬격자 데이터를 읽어 VTK 데이터로 반환 • gdm::converter::EnSightUnstructuredGridLib::GetReaderOutput() 호출

6. OpenFOAM 변환

6.1. OpenFOAM 데이터 포맷

OpenFOAM 데이터의 디렉토리 구조는 다음과 같다. constant 폴더는 격자 정보를 polyMesh 폴더 안에 담고 있고, 그 밖에 물리적 속성에 관한 파일들을 포함한다. system 폴더는 실험 과정에 대한 파라미터를 설정하는 용도이다. time 폴더는 특정 필드에 대한 개별 파일을 담고 있고, 초기값, 경계조건 등을 포함한다.

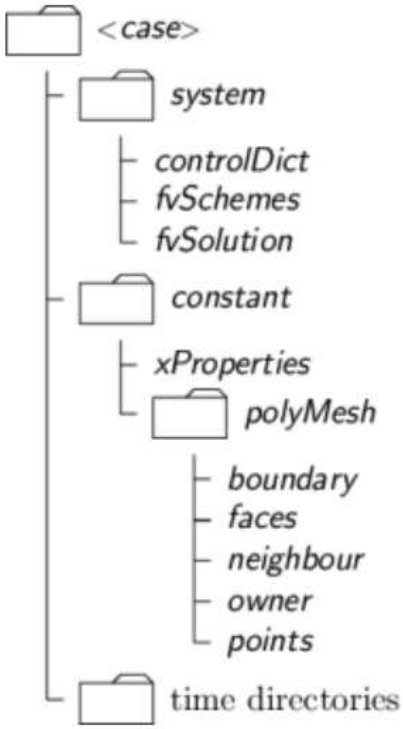

그림 16. OpenFOAM 데이터 디렉토리 구조

6.2. gdm::converter::OpenFOAMUnstructuredGridLib

변환기와는 별도로 GLOVE 데이터 관리자(이하 GDM)에서 OpenFOAM 데이터 Import 기능을 위해 런타임(runtime)에 필요한 기능만 따로 관리한다. 라이브러리 형태로 GLOVE 컴파일을 위해 제공된다.

virtual vtkSmartPointer<vtkDataObject> GetReaderOutput(string srcfile)
• OpenFOAM 비정렬격자 데이터를 읽어 VTK 데이터로 반환

6.3. gdm::converter::OpenFOAMUnstructuredGrid

병렬 MPI 프로세스로 OpenFOAM 비정렬격자 데이터를 GLV 포맷으로 변환한다.

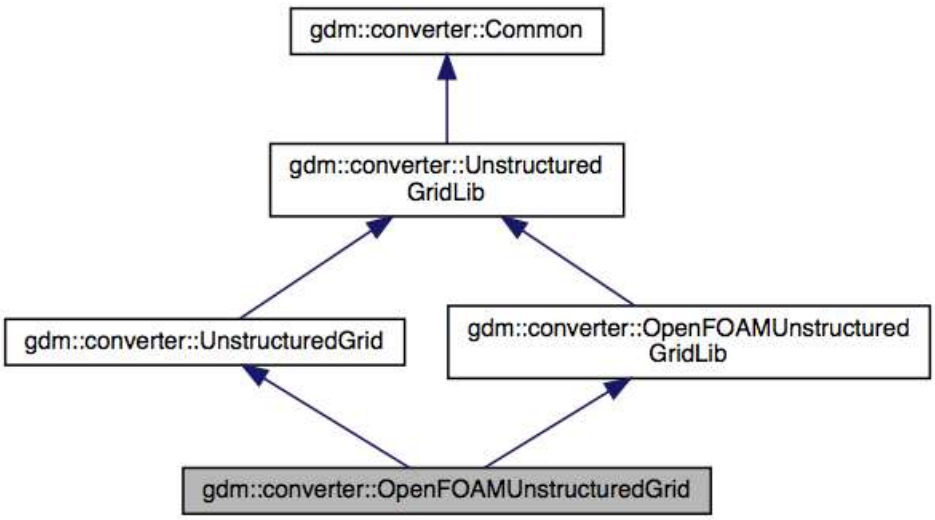

그림 17. gdm::converter::OpenFOAMUnstructuredGrid 클래스의 협업 다이어그램

virtual vtkSmartPointer<vtkDataObject> GetReaderOutput(string srcfile)
• OpenFOAM 비정렬격자 데이터를 읽어 VTK 데이터로 반환 • gdm::converter::OpenFOAMUnstructuredGridLib::GetReaderOutput() 호출

7. 결론

지금까지 고성능 컴퓨터 환경에서 대용량 시뮬레이션 데이터를 실시간으로 병렬 가시화하기 위한 도구 GLOVE를 위해 다양한 포맷의 비정렬격자 데이터를 변환하기 위한 GLOVE 데이터 변환기의 설계 및 구현에 대해 살펴보았다.

계산과학분야의 사용자들이 주로 사용하는 CGNS, EnSight, OpenFOAM의 포맷에 대한 데이터 변환을 지원하는 것은 GLOVE의 활용도를 높이고, 더 많은 사용자를 확보하는데 있어 필수적이다. GLOVE 데이터 변환을 위해 1) 데이터 셋의 메타 정보를 추출하고, 2) GLOVE 데이터 디렉토리를 구성하고, 3) 데이터 크기가 큰 경우 병렬처리를 위해 적당한 크기로 분할하고, 4) 원하는 경우 데이터 변환의 범위를 설정하고, 5) 다양한 비정렬격자 포맷을 지원하고, 6) 실험 데이터의 스칼라 변수를 합쳐서 벡터로 결합하고, 7) 후처리 가시화 단계에서 조작의 용이성을 위해 기존 Element를 병합해 새로운 Element를 생성할 수 있도록 했다.

향후에는 Fluent를 비롯한 더 많은 비정렬격자 데이터를 지원할 뿐만 아니라 데이터 변환에 소요되는 시간을 줄이고, 더 균등하게 데이터를 분할하고, 궁극적으로는 변환 과정 없이도 다른 데이터 포맷을 바로 로드할 수 있는 기능을 만들어갈 예정이다.

8. 참고문헌

[1] Min-Ah Kim et al, "GLOVE(GLObal Virtual reality visualization Environment for scientific simulation): VR환경에서의 대용량 데이터 가시화 시스템" 2010 한국컴퓨터종합학술대회 논문집 제37권 제2호(B), 267-271

[2] CGNS, http://cgns.github.io/

[3] EnSight, https://www.ensight.com/

[4] OpenFOAM, https://www.openfoam.com/