Journal of the Korea Academia-Industrial cooperation Society
Vol. 18, No. 9 pp. 1-8, 2017

https://doi.org/10.5762/KAIS.2017.18.9.1
ISSN 1975-4701 / eISSN 2288-4688

대용량 BIM 형상 데이터 스트리밍을 위한 캐쉬 구조

강태욱
한국건설기술연구원

BIM Geometry Cache Structure for Data Streaming with Large Volume

Tae-Wook Kang
Korea Institute of Civil Engineering and Building Technology

요 약 본 연구의 목적은 물리적 메모리 할당이 어려운 대용량 BIM(Building Information Modeling) 형상 데이터를 처리하기 위한 캐쉬(cache) 구조를 제안한다. 조달청 등 공공기관에서 BIM 발주가 많아짐에 따라 대용량 BIM 형상 데이터를 가시화하고, 계산해야 하는 경우가 많아지고 있다. 규모가 크고 복합적인 시설물의 경우, 렌더링 및 계산해야하는 형상 수가 많아 사용자가 BIM 모델을 검토하고, 단면을 확인하는 데 어려움을 겪는 경우가 있다. 예를 들어, 설계, 검토 협업 시, 대용량 BIM 데이터를 네트워크를 통해 전달받아야 할 경우, 다운로드에 많은 시간이 걸릴 수 있고, 물리적 여유 메모리 한계를 넘어가면, 에러로 가시화나 형상정보 추출이 불가능할 수도 있다. 물리적 메모리가 부족하거나 대역폭이 적은 네트워크 상에서 대용량 BIM 데이터를 활용하기 위해서는, BIM 형상 렌더링 및 계산 시점에 필요한 데이터만 메모리로 캐쉬(cache) 처리하는 것이 유리하다. 이 연구는 물리적 메모리 할당이 어려운 대용량 BIM 형상 데이터를 효과적으로 렌더링하고 계산하기 위한 BIM 형상 캐쉬 구조를 제안한다.

Abstract The purpose of this study is to propose a cache structure for processing large-volume building information modeling (BIM) geometry data,whereit is difficult to allocate physical memory. As the number of BIM orders has increased in the public sector, it is becoming more common to visualize and calculate large-volume BIM geometry data. Design and review collaboration can require a lot of time to download large-volume BIM data through the network. If the BIM data exceeds the physical free-memory limit, visualization and geometry computation cannot be possible. In order to utilize large amounts of BIM data on insufficient physical memory or a low-bandwidth network, it is advantageous to cache only the data necessary for BIM geometry rendering and calculation time. Thisstudy proposes acache structure for efficiently rendering and calculating large-volume BIM geometry data where it is difficult to allocate enough physical memory.

Keywords : BIM, Cache, Large Volume, Streaming, Structure

1. 서론

BIM 모델 형상 데이터는 수많은 객체에 대한 렌더링, 가시화, 단면추출, 계산 등 많은 컴퓨팅 자원을 필요로 한다. 단면추출의 경우, 수많은 객체들을 메모리 상에 로딩한 상태에서 실수형 기하 교차 계산을 해야 하므로, 형상 데이터 크기가 물리적 여유 메모리 한계를 넘어가면, 메모리 할당 에러가 발생된다. 특히, 대역폭이 작은 네트워크 상에서 대용량 BIM 데이터를 가시화할 경우, 데이터를 다운로드 받는 데 많은 시간이 걸리고, 사용자가 그 시간 동안 대기할 수 밖에 없어 불편하다. 이 연구는 대역폭이 작은 네트워크 상에서 물리적인 메모리 한계를

본 연구는 국토교통부 '인터페이스가 용이한 BEMS용 신재생에너지 설비 및 BIM 연계모듈 개발'사업의 연구비지원(17AUDP-B099686-03)에 의해 수행되었습니다.
*Corresponding Author : Tae-Wook Kang(Korea Institute of Civil Engineering and Building Technology)
Tel: +82-10-3008-5143 email: laputa99999@gmail.com
Received July 20, 2017 Revised (1st August 14, 2017, 2nd August 30, 2017)
Accepted September 15, 2017 Published September 30, 2017

넘어갈 수 있는 대용량 BIM 형상 데이터를 가시화하거나 계산할 때 효과적인 데이터 캐쉬 방법을 제안한다.

제시하는 캐쉬 방법은 BIM 데이터 중 형상 데이터에 초점을 맞춘다. BIM 모델을 구성하는 수많은 객체들은 형상과 속성정보로 구성된다. 속성정보는 텍스트 및 숫자로 구성되어 있어, 형상에 비해 데이터 용량이 그리 크지 않다. 형상은 수많은 솔리드 및 메쉬 데이터로 구성되고, 메쉬 데이터에 포함된 수많은 좌표들은 실수형 데이터로 하나의 좌표 값이 최소 24(크기가 8바이트인 배정도 실수 x, y, z)바이트를 차지한다. 형상은 솔리드 정보 자체 뿐 아니라, 시각화를 위한 수많은 메쉬데이터를 포함하고 있다 메쉬데이터는 다시 수많은 삼각형, 정점들과 모서리 정보로 구성된다. 이런 이유로 보통, 대용량의 BIM 모델을 계산할 때 메모리 문제가 발생한다면, 대부분 형상정보를 처리하면서 물리적 메모리가 부족한 경우이다. 만약, BIM 모델을 구성하는 객체 수가 천 만개이고, 객체들이 육면체의 형상을 표현하고 있다면, 물리적 메모리는 최소 1.96GB 여유를 가져야 메모리 상에서 계산이 가능하다. 여기서 계산에 필요한 메모리 크기는 매우 이상적인 가정에서 산출된 것으로, 육면체의 정점 수 6개의 데이터만 관리한다는 가정으로 계산된 것이다. 실제로, 정점과 정점을 연결하는 모서리 정보, 계산과정에서 요구되는 메모리는 더 필요하다.

본 연구는 계산에 요구된 메모리 크기가 물리적 여유 메모리를 넘어가더라도, BIM 모델 객체들에 대한 기하학적 계산을 할 수 있는 데이터 캐쉬 방법을 제안한다. 이 연구는 대역폭이 적은 네트워크 상에서도 데이터를 청크 단위로 스트리밍해 계산할 수 있는 방법을 포함한다. 다만, 이 연구는 대용량 BIM 형상 데이터 스트리밍을 위한 기반연구로 다양한 응용 사례 및 건물 유형 별 분석은 포함되지 않는다. 연구 대상은 대용량 BIM 모델링이 많은 공공시설물을 대상으로 한다. 대용량 BIM 모델 형상 뷰를 위한 목적이므로, 설계 변경과 같이 모델러에 필요한 솔리드 위상 정보 변경 기능은 고려하지 않는다.

2. 연구 방법 및 동향 조사

2.1 연구 방법

본 논문의 연구 방법은 다음과 같다.

우선, 제안 기술에 대한 국내외 기술 동향을 조사한다. 대용량 BIM 형상 데이터 캐쉬 구조를 도출하기 위해, 형상 데이터 계산 과정을 분석하고, 이를 바탕으로 형상 데이터 캐쉬 구조를 설계한다. 설계된 캐쉬 구조에 대한 평가를 위해 구현 사례를 분석한다. 본 연구의 흐름은 Fig. 1과 같다.

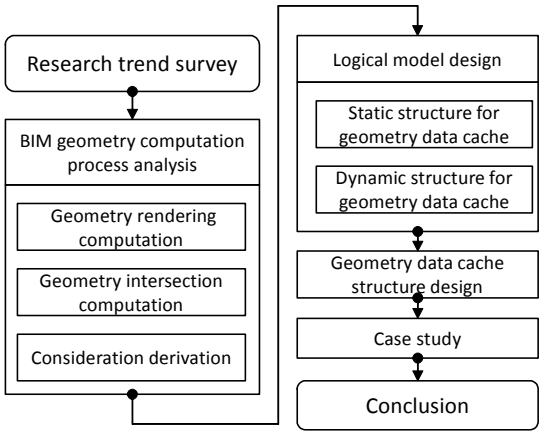

Fig. 1. Research Flow

2.2 연구 동향 조사

대용량 BIM 형상 데이터의 효과적인 스트리밍을 위한 캐쉬 구조 연구 동향은 다음과 같다.

건물 시설물 관리 관점에서 GIS기반 대용량 BIM 형상 객체 표현을 위한 경량 BIM 형상 포맷 구조 개발에 관한 연구가 있었다[1]. 이 연구는 GIS기반에서 수많은 BIM 객체를 관리하고 필요한 정보를 표현하는 데 초점이 맞춰져 있다. 이를 위해 경량화 된 BIM형상 가시화 방법과 포맷 구조를 제안하였다. IFC(Industry Foundation Classes) 파일로부터 건물요소의 형상모델 추출에 관한 연구가 있었다[2].

이 연구는 IFC 기반 건물요소의 기하형상을 추출하기 위한 연구내용을 제시하고 구현된 프로그램에 의한 적용한 사례이다. 의미중심 여과를 통한 BIM 기반 협업설계 정보교환의 경량화 및 자동화에 관한 연구가 있었다[3]. 이 연구는 BIM에 포함된 의미론적 지식공유가 가능한 지식규칙을 정의해 필요 정보 경량화방법을 제안한다. 공간데이터 LOD 기반 3차원 대용량 객체 경량화 알고리즘 연구가 있었다[4].

해외에서는 클라우드 BIM을 통한 정보 교환 방법에 대한 연구가 있었다[5]. 이 연구는 클라우드 플랫폼 기반에서 BIM 정보를 협업하기 위해 필요한 웹서비스 기반

상호운용성 방법을 제안하고 있다. 가상공간 상에서 BIM기반 시설물 관리를 할 경우 문제점을 밝힌 연구가 있었다[6]. 이 연구에서는 게임엔진을 이용해 대용량 BIM 형상 데이터를 가시화할 때 여러 가지 문제점을 언급하고 있다. IFC기반 3차원 형상 모델링 및 변환 방법에 대한 연구가 있었다[7]. 이 연구는 BIM에서 건설 프로세스 각 단계에서 필요한 형상 표면 추출 방법을 제안한다.

관련된 연구들은 대부분 경량화를 위한 메쉬 포맷, 형상 모델 추출, 속성 정보 필터링, 대용량 형상 가시화 시 문제점 도출에 관련된 내용으로 본 연구와는 차이가 있다. 본 연구는 물리적 메모리 한계가 넘어가는 대용량 BIM 형상 데이터를 계산하기 위한 캐쉬 구조를 제안한다.

3. 형상 데이터 계산 과정 분석

3.1 개요

이 장에서는 대용량 BIM 형상 데이터 캐쉬 구조를 도출하기 위해 필요한 형상 데이터 계산 과정을 분석한다. BIM의 형상 데이터는 많은 연산 시간이 렌더링이나 단면 추출 등 기하학적 계산에 많은 시간이 소요된다.

3.2 형상 렌더링 및 교차 계산 과정

렌더링되는 형상들은 3차원 공간에서 가시화 가능한 메쉬(mesh) 구조이다. 메쉬는 3차원 형상을 삼각형으로 표현한 구조이므로, 수학적으로 표현된 형상 정보를 메쉬로 표현하면, 필요한 메모리 크기는 급격히 늘어난다.

카메라에서 먼 거리에 있는 메쉬를 정확히 표현할 필요는 없으므로, 거리에 따라 메쉬 상세 수준인 LoD (Level of Detail)을 가변적으로 표현하는 것이 효과적이다. 모든 메쉬 LoD 데이터를 메모리로 로딩해 사용하는 것은 불필요하다. 카메라 뷰 정보를 고려해, 렌더링에 필요한 LoD 데이터를 캐쉬에 로딩하고, 카메라와 렌더링 할 형상 거리에 따라 적절한 LoD 데이터를 동적으로 로딩해 가시화하는 방법이 필요하다.

형상 데이터 기하학적 계산의 대표적인 유스케이스(use case)는 3차원 형상 간 간섭체크, 2차원 단면추출, 3차원 형상 다면 추출 등이다. 대부분 교차 계산을 위한 형상의 수학 모델이 필요하다. 수학 모델은 선형, 곡선, 평면, 곡면 등을 표현하기 위한 파라메터(parameter)로 구성된다. 대용량 형상 계산에 필요한 정보는 캐쉬에 로딩하고, 계산할 순간에 필요한 형상 정보를 메모리에 로딩하는 방법이 필요하다.

다음은 엔지니어가 모델 단면 검토 시 많이 활용하는 모델 객체 단면추출(sectioning) 기능 수행 시나리오이다.

1. BIM 모델 메모리에 로딩
2. 3차원 공간상에서 절단할 영역을 단면추출할 면이나 육면체로 정의함
3. 정의된 단면추출 영역에 대해, 교차되는 BIM 모델의 객체 형상들을 획득함
4. 획득된 교차 가능한 객체 형상들에 대한 면 교차 계산을 수행
5. 교차 계산을 통해 얻은 교차점, 교차선들을 이용해, 위상적으로 연결함
6. 연결된 연결선 및 폐합면을 사용자에게 리턴함

시공성 검토를 위해 수행되는 간섭체크의 경우에도 앞에서 기술한 단면 추출과 유사한 방식으로 선-선, 선-면, 면-면, 솔리드-솔리드 간 교차 계산을 수행한다. 이 교차 계산 과정은 많은 컴퓨팅 자원과 시간이 필요하다.

3.3 형상 데이터 캐쉬 구조 설계 고려사항

물리적 메모리 한계가 있어 형상 데이터를 메모리에 로딩하기 어렵거나, 대역폭이 작은 네트워크 상에서 대용량 BIM 데이터를 활용할 때, 형상 데이터 캐쉬 구조 설계 고려사항은 다음과 같다.

1. 계산이 필요한 영역의 형상만 다운로드할 수 있도록 공간 인덱싱(Spatial Indexing) 개념 지원
2. 데이터 스트리밍이 가능하도록, 다운로드된 단위마다 렌더링 및 계산이 가능한 청크(chunk) 개념 지원
3. 공간 인덱스 기반 탐색 효율성을 위해, 인덱스된 영역에 포함된 형상 정보 중복 허용 개념 지원

4. 논리 구조 설계

4.1 형상 데이터 캐쉬 정적 구조

대용량 형상 데이터 캐쉬 구조는 메모리 한계를 벗어난 BIM 데이터를 메모리로 로딩해 계산하거나, 대역폭

이 작은 네트워크 상에서 대용량 형상 데이터를 스트리밍하기 위한 방법을 제공한다.

캐쉬는 3장에서 분석한 형상 데이터 계산 과정에 따라 대용량 형상 렌더링을 지원하는 Geometry Rendering Cache, 대용량 형상 계산을 지원하는 Geometry Computation Cache로 구분될 수 있다.

캐쉬는 렌더링 및 계산에 사용할 객체를 신속하게 검색하기 위해, 공간 인덱싱을 지원하는 Spatial Index File을 사용한다. Spatial Index File은 Octree 기반으로 동작하며, Spatial Node를 구성한다. Spatial Node는 공간 상에 객체가 포함되는 영역을 관리하며, 하위 8개의 공간을 분할해, 관리하여, 공간 격자 트리를 구성한다. 각 공간격자의 Spatial Node는 객체정보를 가리키는 Spatial Object Handle을 관리한다.

Cache Object는 BIM객체 데이터를 접근할 수 있는 handle을 유지한다. Cache Handle은 캐쉬가 관리하는 캐쉬 객체에 대한 핸들이며, 이 핸들을 통해, 저장된 BIM 형상 데이터 파일을 접근한다. 형상 계산이 필요하다면, 이 핸들을 통해 물리적 메모리로 데이터를 로딩하고, 계산이 끝나면, 언로딩(unloading)할 수 있다. Fig. 2는 데이터 캐쉬 논리 구조이다.

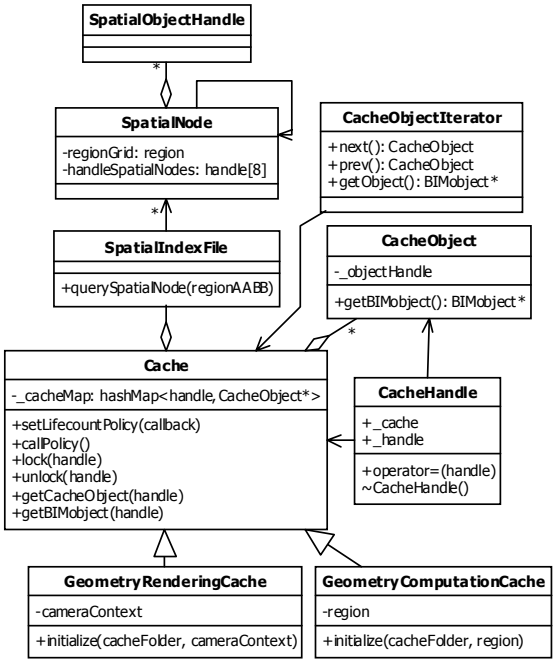

Fig. 2. Data cache logical structure for BIM large volume geometry

Table 1은 각 클래스에 대한 역할 정의이다.

Table 1. Classes definition

Class	Role
Cache	BIM geometry cache information management class. BIM is divided into a geometry rendering cache and a calculation cache, depending on the viewpoint of processing geometry.
Geometry Rendering Cache	Cache for geometry rendering. The cache manages the LoDs for the shape. When mesh is visualized, it loads LoD of shape necessary for rendering into memory and transfers it to visualization function.
Geometry Computation Cache	Manage the cache information for geometry calculation such as geometry section cutting. Loads the information of the shape needed for shape calculation into memory and transfers it to the calculation function.
CacheObject	The cache object manages the handle, which is reference information for loading information necessary for the cache from data file.
CacheHandle	The cache handle refers to the object shape file to be loaded into memory. With the handle, the shape file can be easily loaded into memory.
SpatialIndex File	A spatial indexing file used to spatially search objects to be managed by the cache. The spatial index information of each object is managed by Octree structure.
SpatialNode	As a node of spatial indexing, it manages SpatialObjectHandle which is the handle of the spatial region and the object belonging to that region.
CacheObject Iterator	It determines when to load the BIM file for rendering or calculation of geometry into the memory from the data file and provides the reference pointer that can be used to visualize the object reference point in the loaded memory or to access it from the calculation function.
SpatialObject Handle	Spatial object handles are managed by the SpatialNode of the spatial index and manage the handle information of the objects needed for rendering or calculating the shape.

4.2 형상 데이터 캐쉬 동적 구조

3장에서 설명한 형상 렌더링 및 계산 과정에 따라, 형상 데이터가 필요할 때 데이터를 메모리에 로딩하고, 불필요한 데이터는 언로딩하는 방식으로 캐쉬가 동작되어야 한다. 이 장은 형상 데이터 캐쉬의 동작 과정을 정리한다.

형상 데이터를 사용하기 전에, 캐쉬를 초기화(initialize)하고, 캐쉬에서 아직 메모리에 로딩되지 않은 객체 데이터를 얻기 위한 열거자(iterator)를 획득한다. 열거자는 객체를 얻는 연산자를 실행할 때, Fig. 2에서 제시한 논리 구조의 객체 핸들을 이용해, 해당 데이터가

저장된 파일을 로딩한다. 캐쉬객체(CacheObject)는 메모리에 객체가 어느 기간 동안 존재하고 있어야 하는 지를 결정하는 life count를 관리하고 있다. life count가 0일 때, 캐쉬는 해당 객체의 메모리를 해제하며, life count가 0 이상일 때 객체 정보를 파일에서 메모리로 로딩한다.

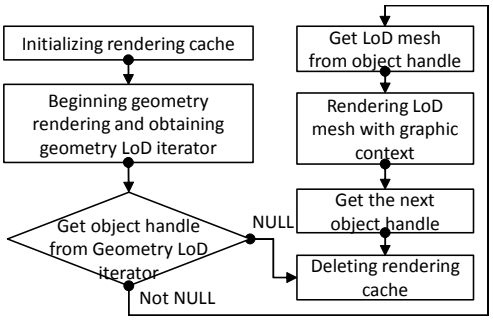

Fig. 3. Rendering cache process

이를 의사코드로 표현하면 다음과 같다.

```
BIM_cache_rendering_iterator listCacheBucketR = beginGe
ometryRendering(cameraContext);
for(objectR in listCacheBucketR)
  renderObject(objectR);
endGeomeryRendering(listCacheBucketR);
listCacheBucketA = beginGeometryCalculation(regionAABB1);
listCacheBucketB = beginGeometryCalculation(regionAABB2);
  for(objectA in listCacheBucketA)
  {
   for(objectB in listCacheBucketB)
     caculateSomething(objectA, objectB);
  }
endGeometryCalculation(listCacheBucketB);
endGeometryCalculation(listCacheBucketA);
```

캐쉬에 보관된 BIM 형상 데이터의 핸들은 캐쉬 메모리 로딩 정책에 따라 언제 메모리로 해당 형상 파일을 로딩할지 결정한다. 이러한 캐쉬의 데이터 파일 로딩 정책은 형상 렌더링 및 계산에 대한 유스케이스 시나리오에 따라 다르다. 이런 이유로, 이 정책을 사용자가 결정할 수 있도록 캐쉬 메모리 로딩 정책 방법을 사용자화할 수 있어야 한다.

예를 들어, 캐쉬 메모리 로딩 정책은 렌더링이나 캐쉬 목적에 따라 다른 정책을 사용할 수 있다. 정책은 규칙으로 정의될 수 있으며, 규칙에 따라, 메모리에 존재하는 life count를 조정하도록 처리될 수 있다.

다음은 렌더링 정책 의사코드의 예이다.

```
rendering_policy(cache, handle)
{
 if(cache[handle].accessLastTime()
   < currentTime() + cacheDurationTime())
    cache[handle].increaseLifecount();
 cache[handle].setAccessLastTime();
}
```

다음은 렌더링 및 형상 계산에 필요한 실제 메모리에 올려 진 객체의 참조를 접근할 수 있는 iterator 의 operator++() 연산자 의사코드이다.

```
object* iterator::operator++() // 연산자
{
 _index++;
 handle = handles[_index];
 if(_cache[handle] ····· == NULL)
  lifeCount = _cache.load(handle); // 메모리 로딩
 return _cache[handle].getBIMObject(); // 참조리턴
}
```

5. 형상 데이터 캐쉬 파일 구조 설계

5.1 개요

이 장에서는 BIM의 대용량 형상 데이터 캐쉬 정보 저장을 위한 파일 구조를 기술한다. 대용량 형상 데이터를 캐쉬하기 위해서는 서버에 있는 BIM 형상 데이터들을 클라이언트에 다운로드 받은 후, 렌더링 및 계산해야 할 객체만 공간인덱스파일을 이용해, 계산에 필요한 형상 데이터만 효율적으로 탐색하여 메모리로 로딩할 수 있어야 한다.

5.1 헤더 구조

캐쉬 파일은 Fig. 4와 같이 캐쉬 파일 헤더(header), 공간 인덱싱을 위한 인덱스 파일(region index section), 객체 정보 파일 부분(objects section)으로 구분된다. 객체 정보 파일은 공간 영역 별로 구분된 객체들 섹션(objects section)으로 구분되며, 각 객체 청크(object chunk)는 객체 청크 헤더, 형상 및 LOD들로 구분된다.

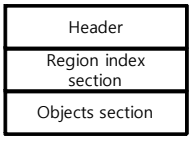

Fig. 4. Cache header structure

제안된 구조로 인해, region index section의 index chunk를 이용해, 필요한 객체 형상을 공간 검색하여 쉽게 얻을 수 있고, index chunk의 objects section position 정보를 통해, 직접적으로 해당 객체의 형상 정보를 파일로부터 바로 읽을 수 있다.

5.2 공간영역 인덱스섹션 구조

Fig. 5는 region index section 구조를 보여준다. 인덱스 파일만 있으면, 필요한 정보를 찾을 수 있으므로, 서버에서 클라이언트로 필요한 정보만 얻기가 용이하며, 이런 이유로 서버에서 클라이언트로 필요한 데이터만 스트리밍 서비스가 가능하다. 이는 클라이언트의 물리적 메모리 용량이 부족할 때 사용하기 좋은 방식이다.

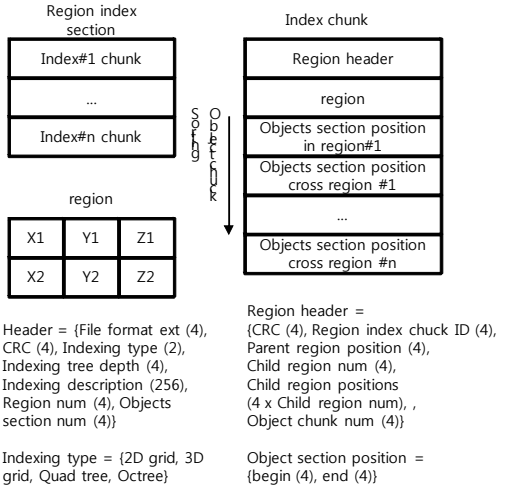

Fig. 5. region index and index chunk structure (The number in parentheses is the byte number)

5.3 객체섹션 구조

인덱스 파일로부터 직접적으로 객체 정보가 있는 파일을 얻을 수 있다. Fig. 6은 객체 정보를 보관하고 있는 objects section 구조이다.

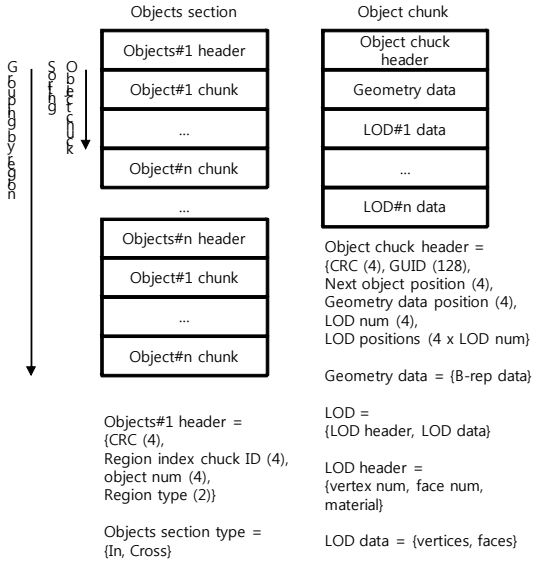

Fig. 6. objects section and object chunk structure

5.4 인덱스 청크 구조

index region은 객체 탐색 성능을 높이기 위해, 영역 간 데이터 중복을 허용한다.

예를 들어, index region은 Fig. 7과 같이 영역 간 걸쳐져 있는 형상 정보들을 다른 objects section 에 중복되어 저장된다. index chunk는 중복되는 데이터를 최소화하도록, index chunk에 region 안에 있는 'in region' 형상들에 대한 데이터 파일 위치와 region 경계와 겹치는 형상들에 대한 데이터 파일 위치를 저장하고 있는 'cross region'에 대한 objects section을 관리한다.

Fig. 7에서 1, 2, 5, 6, 7, 8은 영역 경계에 걸쳐지는 객체 형상이다. 이는 별도로 objects section position cross region에 저장된다.

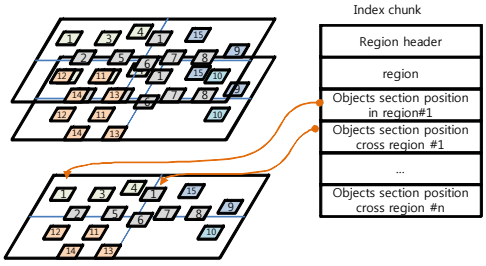

Fig. 7. Index chunk duplicate processing method for position of overlapping shape data between index regions

6. 구현 사례 분석

제안된 대용량 BIM 형상 데이터 캐쉬 구조의 효과를 평가하기 위해, 다음과 같은 대용량 BIM 파일을 서버에 저장하고, 클라이언트에서 렌더링하거나, 교차계산량이 많은 단면추출을 구현해 보았다. BIM 형상은 IFC파일로 저장되어 있으며, 캐쉬 구조는 서버에 IFC파일을 등록할 때, 미리 계산해 두었다. 캐쉬 형상 정보는 IFC의 IfcElement에 포함된 IfcProductDefinitionShape와 IfcLocalPlacement와 관계가 있다. 캐쉬를 위해 관련 정보를 파싱하여 참조하였다.

클라이언트에서 구현된 BIM 뷰어는 렌더링 및 단면추출 기능이 구현되어 있고, 기능을 수행할 때 제안한 캐쉬 구조를 사용한다.

테스트한 대용량 BIM 데이터는 다음과 같다.

Table 2. BIM data rendering and computation test samples

No	Project name	Size (mb)	Object numbers
#1	K.P.E. project	917	14,436,546
#2	Sewage management project	807	14,968,017
#3	KICT	445	7,236,429

다음 그림은 각 테스트 파일을 3차원 렌더링 및 단면추출 계산을 수행한 모습이다.

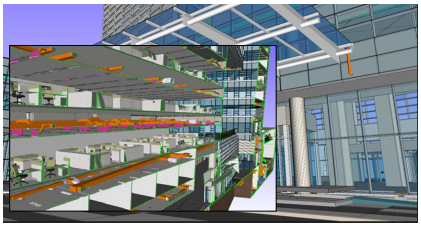

Fig. 8. Sample #1 rendering & sectioning results

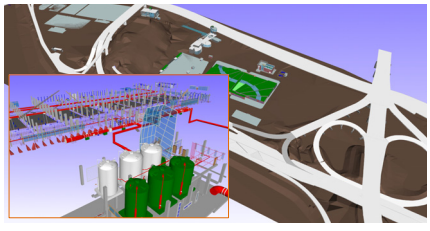

Fig. 9. Sample #2 rendering & sectioning results

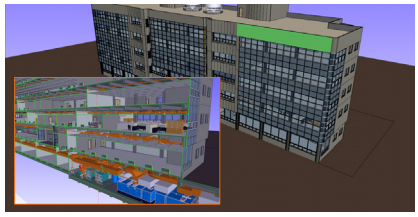

Fig. 10. Sample #3 rendering & sectioning results

Table 3과 같이, 본 연구에서 제안된 방법을 사용하면, 사용전과 후의 렌더링 및 계산 성능이 차이가 있으며, 개선됨을 알 수 있다.

Table 4는 캐쉬 정보 생성 비용 측면에서 분석한 표이며, 수치는 전체 생성 시간 대비 각 단계 생성 시간의 비율이다. 기존 대용량 BIM 형상 정보를 제안하는 캐쉬 정보로 변환하기 위해서는 BIM 정보 파싱 단계, LOD 생성단계, 인덱스 생성단계(Indexing)이 필요하다. Table 4는 이와 관련된 지표 성능을 확인한 것이며, 결과와 같이 LOD를 계산하는 것에 상대적으로 많은 시간이 소요된다. 각 단계의 계산 성능은 모델 복잡도와도 관련성이 있는 것으로 보인다. 모델이 복잡할수록, LOD 및 인덱싱 비용이 높아진다.

Table 3. Test Performance results (second. FPS=Frame Per Second)

No	Inactive Proposed Method		Active Proposed Method	
	Rendering (FPS)	Sectioning (s)	Rendering (FPS)	Sectioning (s)
#1	2	25.2	20	5.8
#2	3	23.8	23	5.3
#3	8	16.7	32	1.6

본 연구에서는 BIM 모델을 서버에 등록할 때, 해당 단계를 한번만 계산하므로, 본 연구에서 기술한 유스케이스 실행 시에는 큰 문제가 없을 수 있으나, 설계 변경과 같이 형상 위상 정보가 실시간으로 변하는 유스케이스 적용 시에는 문제가 될 수 있다. 이런 이유로 BIM 모델러와 뷰어 간의 모델 뷰 성능 차이는 있을 수 있다.

Table 4. Cache information calculation performance ratio (%)

No	Parsing	LOD generation	Indexing	Etc
#1	12.3	43.6	28.5	15.6
#2	11.8	40.3	31.8	16.1
#3	17.3	35.2	25.6	21.9

7. 결론

본 연구에서는 물리적인 메모리 한계를 넘어갈 수 있는 대용량 BIM 형상 데이터를 가시화하거나 계산할 때 효과적인 데이터 캐쉬 방법을 제안한다. 이를 위해, BIM 형상을 활용하는 주요 유스케이스 인 렌더링, 교차계산 과정을 분석하고, 캐쉬 구조 설계 고려사항을 도출하였다. 이를 통해, 캐쉬의 동적, 정적 구조를 도출하고, 캐쉬 파일 포맷을 설계하였다. 제안된 대용량 형상 데이터 캐쉬 구조는 관련 유스케이스를 효과적으로 지원할 수 있다.

다만, 본 연구는 다양한 유형의 샘플을 테스트하지 못한 한계가 있다. 향후, 본 연구와 관련된 클라우드 기반 BIM 서비스 플랫폼 연구에 적용해 건물 유형별 다양한 샘플을 기반으로 적용 효과를 분석할 계획이다.

References

[1] T. W. Kang, C. H. Hong, “A Study on the Lightweight BIM Shape Format(LBSF) Structure Development to Represent the Large Volume BIM Geometry Objects based on GIS as the Viewpoint of the Building Facility Management”, Korea Spatial Information Society, vol. 21, no. 3, pp. 79-87, 2013.
DOI: https://doi.org/10.12672/ksis.2013.21.3.079

[2] G. I. Du, C. J. H, E. D. Kim, J. M. Lee, “Extracting Building Element Geometry from BIM/IFC Physical Files, Computational Structural Engineering Institutes of Korea”, vol. 22, no. 2, pp. 163-172, 2009.

[3] J. H. Jung, S. A. Kim, “Study on the Lightweighting & Automation of Data Exchange by Semantic-Filtering Method in the BIM-based Collaborative Design Process In the initial step of BIM based architectural design process, workloads are increased & the decision making process becomes more complex than those of the convent”, Architectural Institute of Korea, vol. 30, pp. 71-78, 2014.
DOI: https://doi.org/10.5659/JAIK_PD.2014.30.10.71

[4] J. Y. Na, C. H. Hong, A Study on the Weight Lightening Algorithm of 3-Dimensional Large Object based on Spatial Data LOD, Korea Spatial Information Society, vol. 21, no. 6, pp. 1-9, 2013.
DOI: http://dx.doi.org/10.12672/ksis.2013.21.6.001

[5] Redmond, A., Hore, A., Alshawi, M., & West, R, Exploring how information exchanges can be enhanced through Cloud BIM. Automation in construction, 24, pp. 175-183, 2012.
DOI: https://doi.org/10.1016/j.autcon.2012.02.003

[6] Nopachinda, S., Ergan, S. Challenges in Converting Building Information Models into Virtual Worlds for FM Operations and User Studies in the Built Environment.

[7] Jianping, Z., Zhang Yang, Z. X. Methodology of 3D geometric modeling and model conversion of IFC-based BIM. Journal of Information Technology in Civil Engineering and Architecture, vol. 1, no. 1, pp. 40-50, 2009.

강 태 욱(Tae-Wook Kang) **[정회원]**

- 2005년 2월 : 숭실대학교 소프트웨어공학 (공학석사)
- 2009년 3월 : 중앙대학교 건설환경공학 (공학박사)
- 2010년 6월 ~ 2011년 5월 : 중앙대 겸임교수
- 2011년 6월 ~ 2012년 6월 : 한길아이티 BIM본부장
- 2012년 7월 ~ 현재 : 한국건설기술연구원 수석연구원

<관심분야>
CAD, BIM, GIS, SW공학, 비전, 역설계, 로보틱스