A CLOUD COMPUTING APPROACH TO PARTIAL EXCHANGE OF BIM MODELS

Jack C.P. Cheng, Assistant Professor, cejcheng@ust.hk
Moumita Das, Ph.D. Candidate, mdas@ust.hk
Civil and Environmental Engineering, Hong Kong University of Science and Technology, Hong Kong, China

ABSTRACT

Currently, construction project participants collaborate with each other using building information modeling (BIM) technology in a file based manner. BIM models are often highly information rich and huge in size with a complex data structure, making file based BIM model exchange difficult. Several BIM schematic standards are available to formally represent and facilitate efficient sharing of BIM model information. Industry Foundation Classes (IFC) is a BIM standard widely used in the architecture, engineering and construction (AEC) industry to facilitate information sharing. As the AEC industry is project-based, fragmented, and dependent on fast changing information by nature, a server based approach for storage and exchange of BIM models is desirable to integrate fragmented project participants over the web for building design and management. In this paper, a distributed cloud computing server based approach to storing and sharing BIM models is proposed. Cloud computing is a recent technology for storing data and performing complex computations on a distributed manner. Cloud computing facilitates parallel data processing by splitting the job to multiple virtual machines. Cloud computing also provides on demand hardware and software services to avoid expensive initial costs. Unlike in the traditional approach where every end user has his/her own version of BIM model, the cloud based framework presented in this paper provides a centralized server where integrated BIM models are stored and customized BIM models can be obtained. The end users could download or upload partial models by executing updates on the integrated building model in the server. This enhances the information sharing efficiency as only the key information is shared instead of sharing the entire model. In addition, data retrieval and updating become faster due to the concept of parallel processing. An example scenario is presented in this paper to demonstrate the retrieval and updating of IFC building models in our proposed cloud based framework.

Keywords: Building Information Modeling (BIM), Cloud Computing, Collaboration, Industry Foundation Classes (IFC), partial information exchange

1. INTRODUCTION

Building information modeling (BIM) has immense potential to facilitate the information flows in the architecture, engineering and construction (AEC) industry. However, BIM technology is currently used to facilitate the communication and information sharing among construction project participants in a file based manner. The file based way of sharing BIM information incorporates versioning problems, due to which the project partners may not be looking at the most updated file for a BIM model. Working with the wrong version of files can lead to rework and potential delay in the project. Also, BIM files are usually huge making them time consuming to be shared through the Internet. A server based BIM framework which centralizes the information stored in the BIM models like BIMServer (Building Information Modelserver 2013) and Autodesk 360 (Autodesk 2013) are capable of solving these problems of file based sharing. These sever based BIM systems provide functionalities like querying, merging, downloading and annotating on BIM models. However, none of these systems allow partial revisions to be made by each individual user on his own model by uploading partial IFC files. Our distributed cloud server based BIM framework allows storage, querying, updating, partial retrieval, and partial revisions of BIM building models. Unlike the previous server based models, our framework is specifically designed to be deployed on a cloud platform which will make the initial investment on infrastructure become minimal. The data model used to design and deploy our framework on the cloud provides higher throughput for reading and writing data compared to that used to for other server based BIM models. Our framework also provides a simple query language for querying and modifying the data.

In this paper, we present the deployed server based BIM system that we have developed, namely BIM-PDE (BIM Partial Data Exchange) server. BIM-PDE has been deployed using Industry Foundation Classes (IFC) (buildingSMART 2013) and Apache Cassandra (Apache Software Foundation 2013). IFC is a neutral and open standard data model developed by buildingSMART (formerly called International Alliance for Interoperability, IAI) to describe building and construction industry data. The main objective of IFC is to facilitate ease of interoperability between different proprietary BIM software platforms. However, IFC building models are simply shared as files instead of being used as an integrated model of building information. Therefore, sharing of building information seamlessly is still a problem in the construction Industry. The data model of Apache Cassandra is used to implement BIM-PDE server. The system architecture of Cassandra is distributed and is compatible with popular cloud infrastructure providers like Amazon Elastic Compute Cloud (EC2) (Amazon.com 2013). An instance of Cassandra may run on one or more virtual machines called nodes located in the cloud datacenter. Hence, due to the parallel processing of information, the response to the queries to Apache Cassandra is faster.

The remaining of the paper is structured as follows – Section 2 provides some research background on server based BIM, cloud computing, and partial BIM information exchange. It is followed by Section 3 which describes the conceptual framework of the BIM-PDE server. Section 4 discusses the implementation of the BIM-PDE server with IFC and Apache Cassandra. An example scenario is presented in Section 5 describing the functionality of the BIM-PDE server. The paper is summarized with future work in Section 6.

2. RESEARCH BACKGROUND

2.1 Server based BIM and cloud computing

Kang and Lee (2010) proposed an IFC server which uses web based object-relational database to enable multiple users to import and export IFC models. This server also handles version control and user authority management. Nour et al.(2006) developed a web based IFC versioning management system for managing the different versions of a BIM building model in a collaborative environment. This system stores the IFC building elements as versioned objects. The most updated objects are integrated to compose an IFC building file on demand by the user. Nour and Karl (2008) presented an IFC server based model for version control and manipulation of the building entities. However, the existing approaches to server based BIM do not consider the necessity of partial BIM information exchange. Also the collaboration among the partners in the construction industry is project-based and hence short lived. Therefore, the server based BIM should easily deployable for a project on a platform which requires minimal investment on initial infrastructure. The existing approaches to server based BIM, however do not fulfill this criteria.

Cloud computing can be briefly defined as the technology to access services on the Internet. The services can be softwares (Software as a Service), infrastructure (Infrastructure as a Service), and platform (Platform as a Service) (Amazon Web Services 2013). The key benefits of using cloud computing in relation to BIM are accessibility from any location, scalability, and most importantly, cost-effectiveness. The end users pay for only those services that they use. The importance of cloud computing in the construction industry has been identified by some researchers in the recent years. Kumar et al. (2010) proposed a cloud based model for supply chain management. Fathi et al. (2012) proposed a cloud computing based framework for sharing project information. This framework uses the context information of the end users like role, preferences, and site to improve collaboration between the project partners. Redmond et al. (2012) conducted a semi-structured interview of 11 BIM experts and concluded that web based BIM exchanges on a cloud platform can lead to enhanced interoperability between different construction applications. Apart from these, there are commercially available cloud computing based platforms like BIMServer and Autodesk 360 which store BIM information and provide functionalities like querying, merging, annotation on real time BIM model. However, none of the current approaches towards integrating BIM and cloud computing facilitates partial exchange of BIM information or manipulation of the BIM information like adding and deleting of building components in real time.

2.2 Partial BIM information exchange

The construction industry is fragmented and multidisciplinary in nature. The partners in a construction project come from diverse backgrounds like architecture, structural engineering, and HVAC engineering. These project partners work in parallel on the same building model during the design phase. They however work on their individual parts and later integrate the building model when the design process is completed by all the project participants. Also within the same group of project partners like designers, there are sub-members who work in parallel on the design of different sections of the same building. Building design is an iterative process which goes through a phase of design and redesign. Therefore, it is important that all the project partners and their sub-members have access to the most updated integrated building model so that they can identify any relative change that should be made on their part of the model due to the changes made in other part or domain of the building. This would prevent the delay due to iterative redesign. This problem can be solved by facilitating retrieval and updating of partial building model from and to the main building model .Nour (2007) proposed an approach for splitting and merging IFC models. Redmond and Smith (2011) proposed simple XML formats, called Simplified Markup Language (SML) to represent BIM building model for cloud based partial information exchange. The SML mainly represents the geometry of the BIM model. Hence, the current approaches either do not keep all the information from BIM models or do not support the distributed working environment of the construction industry.

3. THE PROPOSED DISTRIBUTED CLOUD SERVER BASED FRAMEWORK FOR PARTIAL DATA EXCHANGE OF BIM

Figure 1 shows our proposed framework for updating IFC building models through partial exchange of the building model. Our framework comprises of a cloud based server called BIM-PDE which stores the building model information in a distributed cloud based system. Our framework takes input in the IFC data format. The output of our framework is also in the same format. Therefore, a translation mechanism is required to convert the data in IFC data model to our cloud based data model. In our framework, we have four translation layers for reading and writing IFC data. The four translation layers are ‘input data transformation layer’, ‘data writing layer’, ‘data reading layer’, and ‘output data transformation layer’, as shown in Figure 1. Upon translation, the information from the IFC files are stored in the prototyped BIM-PDE server which resides in the layer ‘data storage layer’ in our framework. The function of these layers are explained in the following sections.

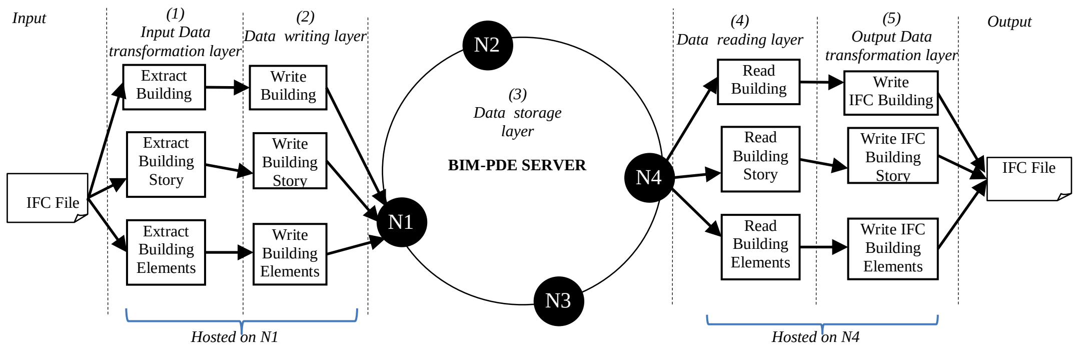


Figure 1: Prototyped framework for partial data exchange for IFC building models

3.1 Input data transformation layer

This layer extracts the key information from the inputted IFC building file and makes the data ready to be loaded in the BIM-PDE server. This layer consists of an IFC parser developed with JSDAI (AGPL v3 2009), which reads the building model in IFC data model. JSDAI is an API for reading, writing, and runtime manipulation of data defined by an EXPRESS based data model like IFC. The IFC parser has three different functions to either read the complete IFC file or partially – (i) extract building, (ii) extract building story, and (iii) extract building element (as shown in Figure 1). If the function ‘extract building’ is invoked, the IFC parser parses the whole file and extracts all the information stored in it like building geometry, site location, and project related information associated with the building model. The second function ‘extract building story’ extracts a building story when the unique GUID of the story is passed as an input parameter. This function parses the IFC file partially and extracts information related to the respective story like story placement and story height along with that of its building elements like wall, door, and window. The input parameters to the third function, ‘extract building element’ are the unique GUIDs of the building elements like wall, slab, door, and roof. This function parses the IFC file partially for information like placement, story number, geometric representations of one or more building elements whose GUIDs have been inputted.

3.2 Data writing layer

This layer writes to the prototyped BIM-PDE server through a client. In our implementation, a Java client for Apache Cassandra called Hector (GitHub 2013) has been used. The translated data from the ‘input data transformation layer’ can be written to the BIM-PDE server in three ways depending upon the function invoked in the same layer. The three functions available in the ‘data writing layer’ corresponding to those of the previous layer (as shown in Figure 1) are – (i) write building, (ii) write building story, and (iii) write building elements. If the first function ‘write building’ is invoked, then the complete building model is written to the BIM-PDE server. This function is used to upload a new building to the server. The second function ‘write building story’ writes only the information needed to represent a building story. This function is invoked to update all the elements of a particular building story of the integrated building model stored in the BIM-PDE server. Similarly, the third function is used to update some particular building elements (depends on the output of ‘extract building elements’ function of the previous layer) of the building model stored in the BIM-PDE server.

3.3 Data storage layer: BIM-PDE server

This is the key part of the framework that actually stores the building model. The system architecture and data model of the implementation of this server will be described in Sections 4.2 and 4.3, respectively. As shown in Figure 1, it has a distributed architecture comprising one or more nodes (individual machines or partitions). In our prototype framework, the data translation layers are hosted on one of the nodes in the cluster as indicated in Figure 1.

3.4 Data reading layer

This layer reads the building information stored in the BIM-PDE server through a Java client. This information can be retrieved in three ways – (i) read building, (ii) read building story, and (iii) reading building elements. As the names suggest, this functions are for fetching the information from the BIM-PDE server for a whole building, for one building story, and for one or more building elements, respectively.

3.5 Output data transformation layer

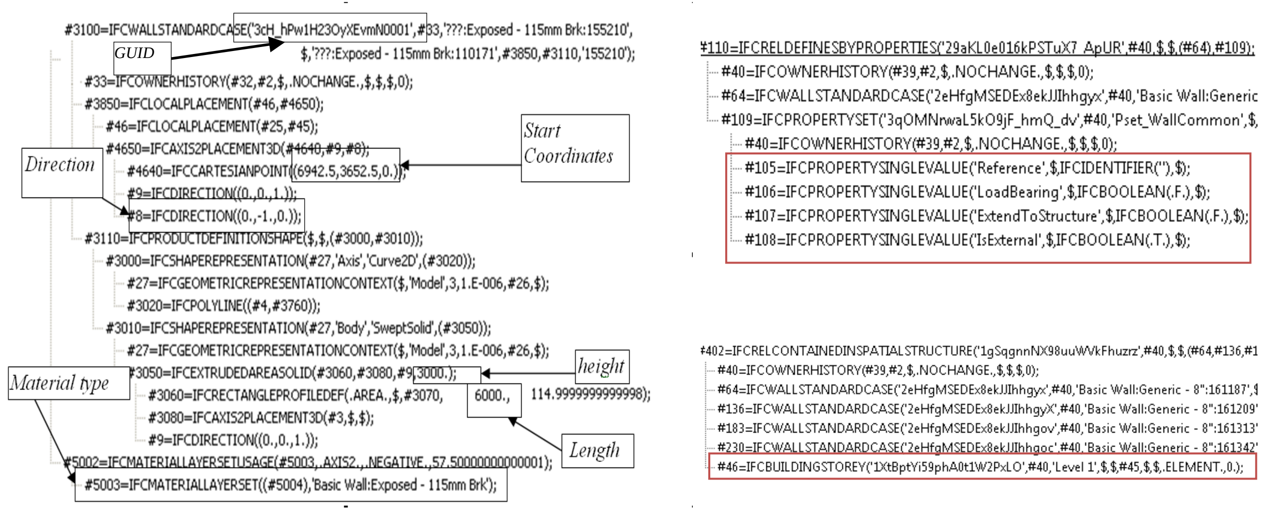
This layer has three functions – (i) write IFC building, (ii) write IFC building story, (iii) write IFC building elements. When the user invokes a function in the ‘data reading layer’, the corresponding function is invoked in the ‘output data transformation layer’ (Figure 1) for writing IFC files containing the whole building model information, information of one building story, or information of individual building elements as requested by the end user.

4. THE PROPOSED BIM-PDE SERVER

The implementation of the BIM-PDE server with IFC and Apache Cassandra is discussed in this section. IFC is an extensive and information rich data model for BIM building models. Most of the current BIM software platforms used in the construction industry allow import and export of IFC files. It is therefore a suitable medium for sharing building information in the construction industry. The distributed architecture and the compatibility of Apache Cassandra with cloud computing platforms makes it suitable for representing an integrated source of building information among the project partners in a collaborative framework. The data model of IFC and the architecture of Apache Cassandra will be discussed in Sections 4.1 and 4.2.

4.1 Representation of a building components in IFC format

In this section we are investigating IFC to understand its data model. In IFC, the building components like walls, doors and windows are defined in terms of IFC entities which are organized in a tree-like structure (Figure 2). IFC has specific keywords for each building element. For example, the keyword IfcWallStandardCase is for walls, IfcWindow is for windows, and IfcDoor is for doors. Each of these entities have a tree-like sub-entities to represent its geometry in terms of shape representation models (like BREP) and location with respect to a world coordinate system established uniquely for each new project. For example, for a standard wall (Figure 2(Left)), the entity IfcWallStandardCase defines the geometry and the placement through the sub-trees, IfcProductDefinitionShape(#3110) and IfcLocalPlacement (#3850), respectively. The material associated with a building element is defined by the entity, IfcRelAssociatesMaterial. Figure 1 shows that the wall in the example is a basic brick wall. The properties of a building element, like its relation with other building elements and structural properties are defined by the entity IfcRelDefinesByProperties. Figure 2 (Right) shows the properties associated with a standard wall in IFC. All the building elements like walls, doors and windows that belong to a particular floor/level in an IFC building is identified by the element IfcRelContainedSpatialStructure. Figure 2 (Right) shows IfcRelContainedSpatialStructure (#402) which refers to all the walls belonging to level 2 of the IFC building. Through this study of the IFC data model, we have identified the key information required to represent any IFC building element. For example, for a standard wall, (Figure 2 (Left)), the minimum information required to represent it in IFC format is– (1) GUID of the wall, (2) placement coordinates of wall (3) direction of wall, (4) length, (5) width, (6) height, (7) material type, and (8) floor number.



```
#3100=IFCWALLSTANDARDCASE('3cH_hPw1H23OyXEvmN0001',#33,'???:Exposed - 115mm Brk:155210',
                                        $,'???:Exposed - 115mm Brk:110171',#3850,#3110,'155210');
  #33=IFCOWNERHISTORY(#32,#2,$,.NOCHANGE.,$,$,$,0);
  #3850=IFCLOCALPLACEMENT(#46,#4650);
    #46=IFCLOCALPLACEMENT(#25,#45);
    #4650=IFCAXIS2PLACEMENT3D(#4640,#9,#8);
      #4640=IFCCARTESIANPOINT((6942.5,3652.5,0.));
      #9=IFCDIRECTION((0.,0.,1.));
      #8=IFCDIRECTION((0.,-1.,0.));
  #3110=IFCPRODUCTDEFINITIONSHAPE($,$,(#3000,#3010));
    #3000=IFCSHAPEREPRESENTATION(#27,'Axis','Curve2D',(#3020));
      #27=IFCGEOMETRICREPRESENTATIONCONTEXT($,'Model',3,1.E-006,#26,$);
      #3020=IFCPOLYLINE((#4,#3760));
    #3010=IFCSHAPEREPRESENTATION(#27,'Body','SweptSolid',(#3050));
      #27=IFCGEOMETRICREPRESENTATIONCONTEXT($,'Model',3,1.E-006,#26,$);
      #3050=IFCEXTRUDEDAREASOLID(#3060,#3080,#9,3000.);
        #3060=IFCRECTANGLEPROFILEDEF(.AREA.,$,#3070,     6000.,    114.9999999999998);
        #3080=IFCAXIS2PLACEMENT3D(#3,$,$);
        #9=IFCDIRECTION((0.,0.,1.));
#5002=IFCMATERIALLAYERSETUSAGE(#5003,.AXIS2.,.NEGATIVE.,57.50000000000001);
  #5003=IFCMATERIALLAYERSET((#5004),'Basic Wall:Exposed - 115mm Brk');
```

```
#110=IFCRELDEFINESBYPROPERTIES('29aKL0e016kPSTuX7_ApUR',#40,$,$,(#64),#109);
  #40=IFCOWNERHISTORY(#39,#2,$,.NOCHANGE.,$,$,$,0);
  #64=IFCWALLSTANDARDCASE('2eHfgMSEDEx8ekJJIhhgyx',#40,'Basic Wall:Generic
  #109=IFCPROPERTYSET('3qOMNrwaL5kO9jF_hmQ_dv',#40,'Pset_WallCommon',$,
    #40=IFCOWNERHISTORY(#39,#2,$,.NOCHANGE.,$,$,$,0);
    #105=IFCPROPERTYSINGLEVALUE('Reference',$,IFCIDENTIFIER(''),$);
    #106=IFCPROPERTYSINGLEVALUE('LoadBearing',$,IFCBOOLEAN(.F.),$);
    #107=IFCPROPERTYSINGLEVALUE('ExtendToStructure',$,IFCBOOLEAN(.F.),$);
    #108=IFCPROPERTYSINGLEVALUE('IsExternal',$,IFCBOOLEAN(.T.),$);
```

```
#402=IFCRELCONTAINEDINSPATIALSTRUCTURE('1gSqgnnNX98uuWVkFhuzrz',#40,$,$,(#64,#136,#1
  #40=IFCOWNERHISTORY(#39,#2,$,.NOCHANGE.,$,$,$,0);
  #64=IFCWALLSTANDARDCASE('2eHfgMSEDEx8ekJJIhhgyx',#40,'Basic Wall:Generic - 8":161187',$
  #136=IFCWALLSTANDARDCASE('2eHfgMSEDEx8ekJJIhhgyX',#40,'Basic Wall:Generic - 8":161209'
  #183=IFCWALLSTANDARDCASE('2eHfgMSEDEx8ekJJIhhgov',#40,'Basic Wall:Generic - 8":161313'
  #230=IFCWALLSTANDARDCASE('2eHfgMSEDEx8ekJJIhhgoc',#40,'Basic Wall:Generic - 8":161342'
  #46=IFCBUILDINGSTOREY('1XtBptYi59phA0t1W2PxLO',#40,'Level 1',$,$,#45,$,$,.ELEMENT.,0.);
```

Figure 2: (Left) Representation of a placement, geometry and material type of a standard wall in IFC, (Right) Representation of a the properties wall in IFC and a building story in IFC

4.2 Cassandra system architecture and data model

Apache Cassandra is an open source distributed database management system. The architecture of a Cassandra instance consists of a set of independent nodes (computers/ disk space) configured together in a cluster (Figure 3 (left)). The data stored in a Cassandra database is distributed among the nodes which is called data partitioning. There are two important factors involved with data partitioning – (1) to determine on which node on the Cassandra instance cluster should the data be stored and (2) replication replacement strategy, which determines the number of copies of data to be stored (Datastax 2013). For example, if the replication replacement strategy of a Cassandra cluster is 3, then one piece of data will be replicated to three different nodes on the cluster.

The data model of Cassandra is schema-optional and column-oriented (Datastax 2013). This means that unlike relational databases, there is no need to model all the columns of a column family (analogous to table in relational databases) upfront. Each row may have a different set of columns which may be added during the runtime. Each row is identified by a row key which acts as a primary key. In this example, as shown in Figure 3 (Right), the username is the row key. A column value may be fetched in two ways – (i) fetching by row key and (ii) fetching by column name by creating a secondary index on the column. The two features of Cassandra which makes it suitable for our prototyped framework (as shown in Figure 3 (Right)) – (i) dynamic column family, which means that metadata of the column family is not required to be predefined and columns may be added dynamically during runtime, and (ii) super columns, which adds levels of nestling to the regular column family structure. The flexibility to add columns dynamically makes our model scalable to incorporate changes when new IFC entities are added in an existing domain or a new domain is added to the IFC data model. Also, IFC data model stores the building data in a hierarchical structure. The concept of super columns can represent hierarchical data in nested columns in a database. The implementation of an IFC data model in Cassandra through dynamic columns and super columns are explained in detail in the section 4.3.

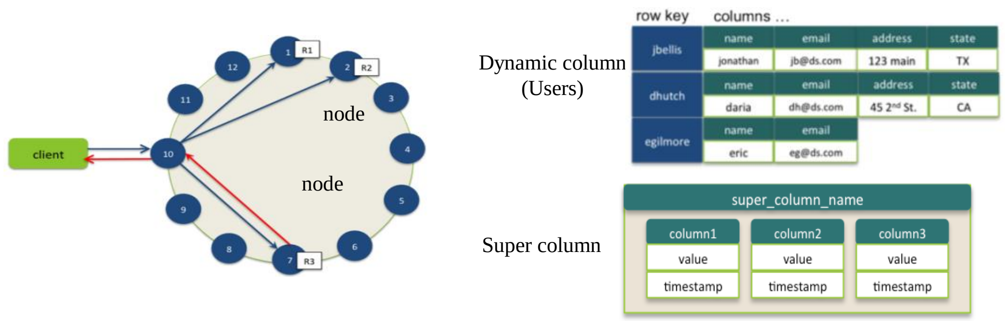
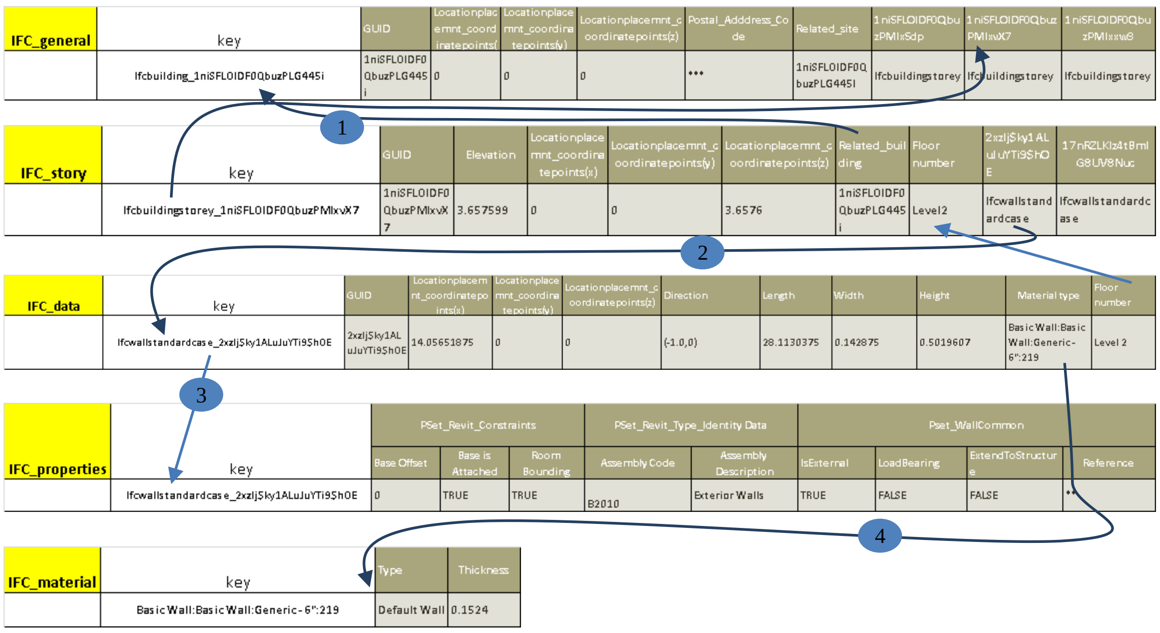


Figure 3: (Left) Architecture of a Cassandra instance, (Right) Cassandra data model- Example of representation of a column family that stores user information in Cassandra

4.3 Description of BIM-PDE Server: representation of IFC building information in Cassandra data model

Figure 4 shows the representation of IFC BIM building information in Cassandra data model. We have designed five column families (w.r.t the types of IFC entities required to represent a building element, as described in section 4.1) to store all the BIM building information extracted from an IFC file. The five column families are- (1) IFC_data, (2) IFC_story, (3) IFC_material, (4) IFC _properties, and (5) IFC_general. Out of these five column families, IFC_data, IFC_story, IFC_material, and IFC_general have been designed as regular dynamic column families of Cassandra data model. However due to complex hierarchical representation of the properties of building element in the IFC, a dynamic super column family, namely IFC_propeties has been designed. Figure 4 shows the important columns of each column family required to represent an IFC wall in the Cassandra data model. The column family, IFC_data contains all the geometry and placement related data of the building elements from the IFC file (IFC entity IfcWallStandardCase for wall). Figure 4 shows that IFC_data contains all the eight entities required to represent an IFC wall as mentioned in section 4.1. The column family, IFC_story contains data needed to represent a building story (IFC entity IfcRelContainedSpatialStructure) like height of the story, location of the story, and the relation with the building elements in the story. Figure 4 shows that IFC_story contains relations to two walls (with GUIDs ‘2xzIj$ky1ALuJuYTi9$hOE’ and ‘17nRZLKlz4tBmlG8UV8Nuc’) which are located on the second story. The column family, IFC_material stores material information (IFC entity IfcRelAssociatesMaterial) of the building elements. In Figure 4, IFC_data has a column for the material type for a wall. This value of this column is the row key to the related columns in IFC_material (as shown with the arrows). The column family, IFC_ properties stores the properties of a building element. It is clear from the figure below that a building element wall with GUID of ‘2xzIj$ky1ALuJuYTi9$hOE’ is an external wall which bounds a room. The column family, IFC_general stores the remaining information associated with a building like project, site, and owner.



IFC_general	key	GUID	Locationplacemnt_coordinatepoints(	Locationplacemnt_coordinatepoints(y)	Locationplacemnt_coordinatepoints(z)	Postal_Adddress_Code	Related_site	1niSFLOIDF0QbuzPMIxSdp	1niSFLOIDF0QbuzPMIxvX7	1niSFLOIDF0QbuzPMIxxw8
	Ifcbuilding_1niSFLOIDF0QbuzPLG445i	1niSFLOIDF0QbuzPLG445i	0	0	0	•••	1niSFLOIDF0QbuzPLG445I	Ifcbuildingstorey	Ifcbuildingstorey	Ifcbuildingstorey

IFC_story	key	GUID	Elevation	Locationplacemnt_coordinatepoints(x)	Locationplacemnt_coordinatepoints(y)	Locationplacemnt_coordinatepoints(z)	Related_building	Floor number	2xzlj$ky1ALuJuYTi9$hOE	17nRZLKlz4tBmlG8UV8Nuc
	Ifcbuildingstorey_1niSFLOIDF0QbuzPMIxvX7	1niSFLOIDF0QbuzPMIxvX7	3.657599	0	0	3.6576	1niSFLOIDF0QbuzPLG445i	Level2	Ifcwallstandardcase	Ifcwallstandardcase

IFC_data	key	GUID	Locationplacemnt_coordinatepoints(x)	Locationplacemnt_coordinatepoints(y)	Locationplacemnt_coordinatepoints(z)	Direction	Length	Width	Height	Material type	Floor number
	Ifcwallstandardcase_2xzlj$ky1ALuJuYTi9$hOE	2xzlj$ky1ALuJuYTi9$hOE	14.05651875	0	0	(-1.0,0)	28.1130375	0.142875	0.5019607	BasicWall:BasicWall:Generic-6":219	Level 2

IFC_properties	key	PSet_Revit_Constraints			PSet_Revit_Type_Identity Data		Pset_WallCommon			
		Base Offset	Base is Attached	Room Bounding	Assembly Code	Assembly Description	IsExternal	LoadBearing	ExtendToStructure	Reference
	Ifcwallstandardcase_2xzlj$ky1ALuJuYTi9$hOE	0	TRUE	TRUE	B2010	Exterior Walls	TRUE	FALSE	FALSE	••

IFC_material	key	Type	Thickness
	BasicWall:BasicWall:Generic-6":219	Default Wall	0.1524

Figure 4: BIM-PDE Server: Cassandra data model for IFC

5. EXAMPLE SCENARIO

In this section, we will present a demonstrative scenario of partial BIM building data exchange through our prototyped framework (Figure 5). This example will discuss the downloading/uploading of partial BIM model from the main model stored in the BIM-PDE server. The BIM-PDE server in our framework is invoked via a user interface as shown in Figure 6. The welcome page checks credentials of the users and authenticates the authorized user to the ‘BIM Partial Data Exchange’ web page. This web page performs three functions – (1) uploading a new building, (2) downloading a partial model, and (3) uploading a partial model. These functionalities that facilitates partial building information exchange between projects partners have been explained in the following sections through the scenarios –(1) upload a new building, (2) get partial model, and (3) update partial model.

5.1 Upload a new building

In this example, a designer creates a new building model in Autodesk Revit Architecture and exports an IFC file, IFC1 (Figure 5). This file contains the whole building model in IFC. Upon authentication, the designer selects this IFC file and clicks on ‘Upload IFC file’ button on the ‘BIM Partial Data Exchange’ web page (Figure 6). The contents of IFC1 is parsed by the function ‘extract Building’ of the ‘input data transformation layer’ and inserted in the BIM-PDE server through the function ‘write Building’ in the ‘data writing layer’(Figure 1). At the backend, all the column families (Figure 4) are populated in the BIM-PDE server – (i) building and its stories (IFC_general and IFC_story), (ii)the building elements like wall, windows, doors, and roof (IFC_data), and (iii) material and properties of the building elements (IFC_material and IFC_propeties).

5.2 Get partial model

An architect working on the same project intends to change the design by removing one wall from the second story and adding another. The architect authenticates himself through the frontend and selects the radio button, 'get by story name' and selects 'Level 2' from the dropdown box in section 3, 'Get partial model' on the 'BIM Partial Data Exchange' web page (Figure 6) and clicks on the 'Get' button. This action downloads IFC file, IFC2 (Figure 5) containing a partial building model containing the elements of only the second story of the building. This action can be explained as follows. When the information is received through the frontend, the 'read building story' function of the 'data reading layer' executes queries and selects the data representing the second story of the building model from the BIM-PDE server. The Hector API in the 'read building story' function executes 'fetch' queries on the BIM-PDE server. The queries may fetch one or multiple rows of a column family at a time. The keywords in the web page are mapped to the row keys of the column families stored in BIM-PDE server. In this example 'level 2' (Figure 6) is mapped to the row key 'Ifcbuildingstorey_1niSFLOlDF0QbuzPMlxvX7' (Figure 4) of the column family IFC_story. The fetch query consists of the following steps (marked by numbered on the arrow in Figure 4) - (1) get columns from IFC_data for key 'Ifcwallstandardcase_2xzIj$ky1ALuJuYTi9$hOE', (2) get columns from IFC_properties for key 'Ifcwallstandardcase_2xzIj$ky1ALuJuYTi9$hOE', and (3) get columns from IFC_material for key 'Basic Wall:Basic Wall:Generic - 6':219'. After the relevant columns are retrieved, the 'write building story' function of 'output data transformation layer' , writes the file IFC2 (Figure 5) which is downloaded to the architect's computer. In this process, the architect downloads a partial building file, which is much smaller in size in compared to that of the whole building.

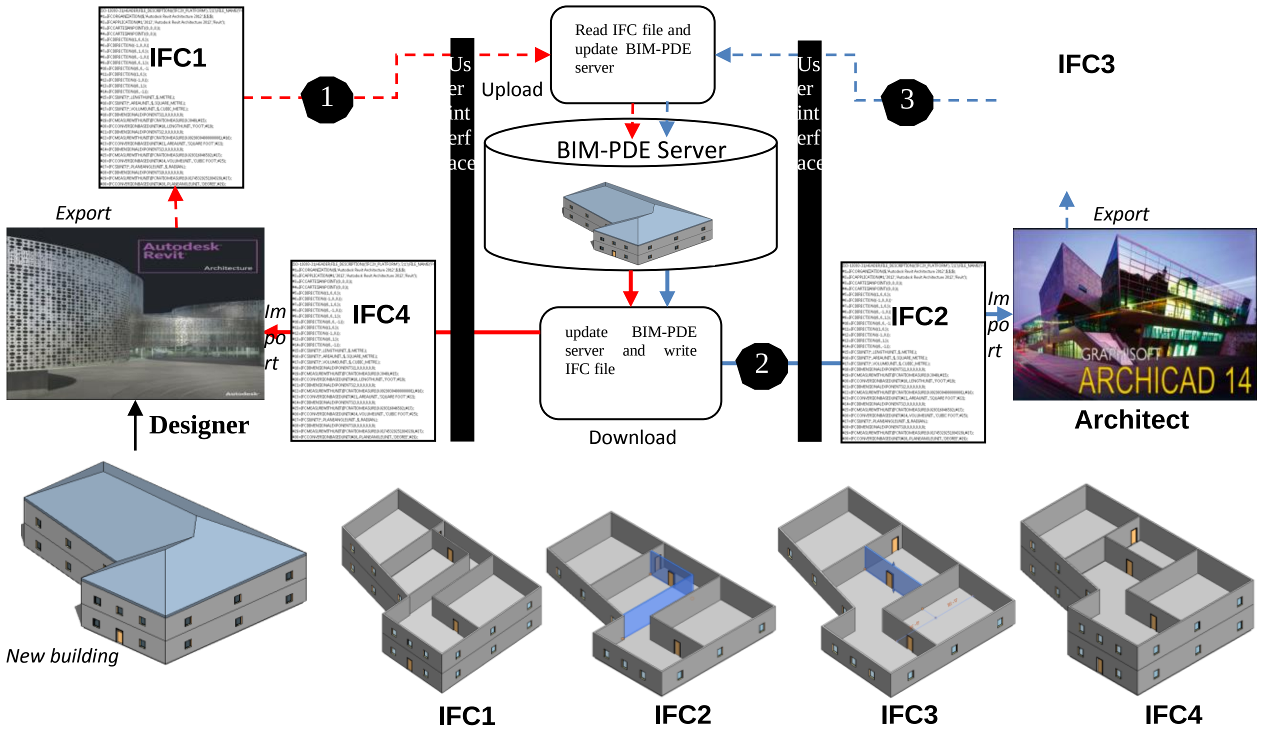
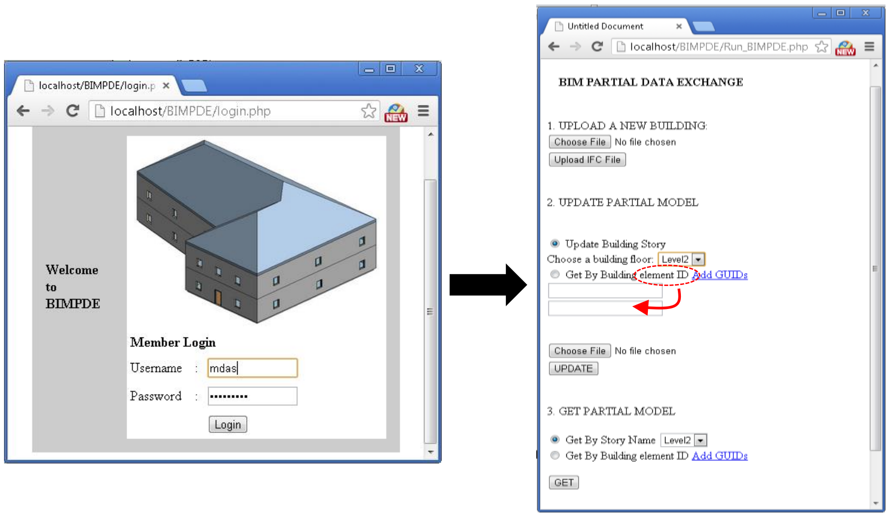


Figure 5: Example scenario describing partial BIM building model transfer (the roof is hidden in IFC1 and IFC4)


localhost/BIMPDE/login.php

Welcome to BIMPDE

Member Login

Username : mdas

Password : ••••••••••

Login

localhost/BIMPDE/Run_BIMPDE.php

BIM PARTIAL DATA EXCHANGE

1. UPLOAD A NEW BUILDING

Choose File No file chosen

Upload IFC File

2. UPDATE PARTIAL MODEL

(●) Update Building Story

Choose a building floor: Level2

( ) Get By Building element ID Add GUIDs

Choose File No file chosen

UPDATE

3. GET PARTIAL MODEL

(●) Get By Story Name Level2

( ) Get By Building element ID Add GUIDs

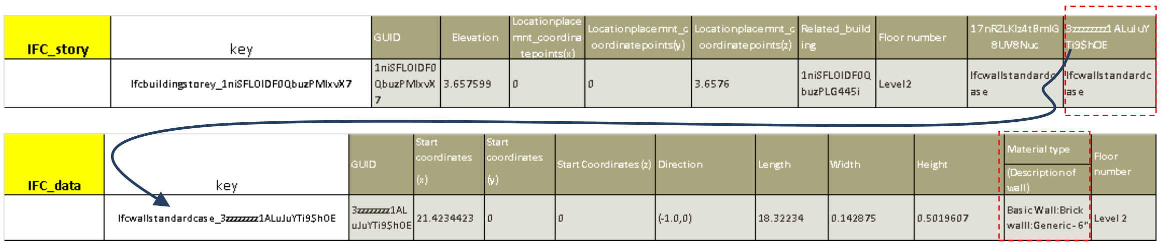
GET

Figure 6 : Frontend to access the BIM-PDE server for partial BIM building model retrieval and updating

5.3 Update partial model

The architect imports file IFC2 to a BIM software, Graphisoft ArchiCAD and for visualization makes changes to the partial building model. The architect deletes the highlighted wall in IFC2 (Figure 5), adds a new wall, and exports an IFC file (IFC3) for the modified partial building model. The changes made by the architect in the second story can be clearly observed by comparing IFC2 and IFC3 in Figure 5. The architect intends to update these changes to integrated building model stored in the BIM-PDE without affecting the other parts of the same model. To do so, the architect uploads the file IFC3 through the frontend (Figure 6) by clicking on the browse button in section 2 of the web page. The architect then selects the radio button, ‘Update building story’ and chooses ‘Level 2’ and clicks on the button ‘Update’. This action invokes the function ‘extract building story’ of the ‘input data transformation layer’ (Figure 1). This function parses the file, IFC3 and extracts key information of the second story. The function ‘write building story’ of the ‘data writing layer’ (Figure 1) is then invoked. This function updates the key information read from the partial building model file, IFC3 to the BIM-PDE server. As shown in figure 7, the column ‘2xzIj$ky1ALuJuYTi9$hOE’ (of Figure 4), which represents the deleted wall has been replaced by ‘3zzzzzzzz1ALuJuYTi9$hOE’, which represents the new wall. The other columns in the same row are unaffected. The rows in the column families ‘IFC_data’, ‘IFC_properties’, and ‘IFC_material’ relative to ‘2xzIj$ky1ALuJuYTi9$hOE’ (shown in Figure 4 with arrows 2, 3, 4) are deleted and the same relative to that of ‘3zzzzzzzz1ALuJuYTi9$hOE’ have been created (Figure 7, IFC_material and IFC_properties are not shown). This completes the updating of the BIM-PDE sever for the changes made to the partial building model. In this process, the architect is able to integrating the changes made by him in the partial model very easily in the main model by exchanging a partial building file.

The designer then downloads the whole building model, IFC4 from the BIM-PDE server. The relevant changes are reflected in the main model (Figure 5).



IFC_story	key	GUID	Elevation	Locationplacemnt_coordinatepoints(x)	Locationplacemnt_coordinatepoints(y)	Locationplacemnt_coordinatepoints(z)	Related_building	Floor number	17nRZLKlz4tBmlG8UV8Nuc	3zzzzzzzz1ALuJuYTi9$hOE
	Ifcbuildingstorey_1niSFLOIDF0QbuzPMlxvX7	1niSFLOIDF0QbuzPMlxvX7	3.657599	0	0	3.6576	1niSFLOIDF0QbuzPLG445i	Level2	Ifcwallstandardcase	Ifcwallstandardcase

IFC_data	key	GUID	Start coordinates (x)	Start coordinates (y)	Start Coordinates (z)	Direction	Length	Width	Height	Material type (Description of wall)	Floor number
	Ifcwallstandardcase_3zzzzzzzz1ALuJuYTi9$hOE	3zzzzzzzz1ALuJuYTi9$hOE	21.4234423	0	0	(-1.0,0)	18.32234	0.142875	0.5019607	Basic Wall:Brick walll:Generic-6"	Level 2

Figure 7 : Representation of the column family ‘IFC_story’ and ‘IFC_data’ after partial update

6. SUMMARY AND FUTURE WORK

This paper presents a cloud based data model, BIM-PDE server for storing the information in a BIM building model and the framework to access the data model. we address two main problems in the construction industry through our prototyped BIM-PDE server. The two main problems are – (1) interoperability issues between BIM softwares and (2) lack collaboration among the project partners. BIM standards like IFC data model have been developed to promote interoperability between BIM softwares. However, information in BIM building models are shared among the construction project partners in a file based manner. This attempt to share information in a file based manner makes the communication between the project partners inefficient and delayed. BIM-PDE acts as central repository of BIM building information. The project partners can update a part of the main model very easily by uploading the partial model they are working on. BIM-PDE server due to its distributed architecture reads the data in parallel threads and can generate IFC files very quickly from the real time data stored in it. IFC files can be imported and visualized on most of the BIM softwares.

In future the BIM-PDE server will be implemented on a cloud platform, Amazon EC2. The framework may be extended to include the other BIM standards like Green Building XML (GbXML) and CityGML to improve the interoperability. Other domains and functionalities like energy simulation and scheduling may be added to the framework.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the support by the Hong Kong Research Grants Council, Grant No. DAG09/10.EG02. Any opinions and findings are those of the authors, and do not necessarily reflect the views of the Hong Kong Research Grants Council.

REFERENCES

AGPL v3. “JSDAI”. http://www.jsdai.net/home (5/18/2013, 2009).Amazon.com. “Amazon Elastic Compute Cloud”. http://aws.amazon.com/ec2/ (5/18/2013, 2013).

Amazon Web Services.“What is Cloud Computing?”. http://aws.amazon.com/what-is-cloud-computing/ (5/18/2013, 2013).

Apache Software Foundation. “Apache Cassandra”. http://cassandra.apache.org/ (5/18/2013, 2013).

Autodesk. “Autodesk 360”. https://360.autodesk.com/landing (5/18/2013, 2013). buildingSMART.

“Industry Foundation Classes”. http://www.buildingsmart.org/standards/ifc (5/18/2013, 2013).

Datastax. “Apache Cassandra 1.0 Documentation”. http://www.datastax.com/doc-source/pdf/cassandra10.pdf (5/18/2013, 2013).

Fathi, F. S., Abedi, M., Rambat, S., Rawai, S., & Zakiyudin, M.Z. (2012). “Context-Aware Cloud Computing for Construction Collaboration”. *Journal of Cloud Computing, 2012.*

GitHub. “Hector”. http://hector-client.github.io/hector/build/html/index.html (5/18/2013, 2013).

Kang, H., & Lee, G. (2009). “Development of an Object-Relational IFC Server”. International Conference on Construction Engineering and Management/Project Management (ICCEM/CCPM), Jeju, Korea.

Kumar, B., Cheng, J.C.P., & McGibbney, L. (2010). “Cloud computing and its implications for construction IT. Proceedings of the International Conference on Computing in Civil and Building Engineering, Nottingham, UK, 315-324.

Nour, M. M. (2007). “Manipulating IFC sub-models in Collaborative Teamwork Environments”. *Proceedings of the 24th CIB W-78 Conference*. Maribor, Slovenia.

Nour, M. M., Firmenich, B., Richter, T., Koch, C., (2006), “A Versioned IFC database for multidisciplinary synchronous cooperation”. *Proceedings of the CIB W078 23nd Joint International Conference on Computing and Decision Making in Civil and Building Engineering,* , Rotterdam , Netherlands, 636-645.

Nour, M. M., & Karl, B. (2008). “An Open Platform for Processing IFC Model Versions”. *Journal of Tsinghua Science and Technology*,13, 126-131.

Redmond, A., Hore, A., Alshawi, M., West, R. (2012). “Exploring how information exchanges can be enhanced through Cloud BIM”. *Journal of Automation in Construction,* 24, 175-183.

Redmond, A. & Smith, B. (2011). “Exchanging Partial BIM Information through a Cloud-Based Service: testing the efficacy of a major innovation”. Pending Publication, IBEA Conference, South Bank University, London.