Dotloom: Toward a Decentralized Data Platform for Massive Three-dimensional Point Clouds

Taehoon Kim, Kyoung-Sook Kim*, Jun Lee and Akiyoshi Matono
Artificial Intelligence Research Center (ARIC)
National Institute of Advanced Industrial Science and Technology (AIST)
Tokyo, Japan
Email: {kim.taehoon, ks.kim, jun.lee, a.matono}@aist.go.jp
*corresponding author

***Abstract*—Sharing data securely and reliably is challenging, mainly when dealing with big spatial data. Data portals, web services, and platforms often struggle when uploading and downloading such data, requiring significant investments in IT infrastructure and expensive high-bandwidth network connectivity to achieve adequate performance for many applications. Dotloom aims to change this and make the sharing of data easier, straightforward, secure, and efficient.**

Dotloom is a distributed data platform for synchronizing, replicating, indexing, and processing terabytes of point-cloud data with peer-to-peer technologies. The distributed nature allows instant exchange between data producers and data consumers. Processing pipelines have the power to stream data from multiple peers, and the generated output can be shared again instantly. Remote indexing can be implemented using partial data, reducing transfer costs. Building on the existing "DAT Project" infrastructure, Dotloom adds the functionality needed to manage, query, and visualize point-cloud data. These novel features of Dotloom have the potential to not only transform how we deal with point-clouds but to be influential across the wider big-data research and development community.

***Index Terms*—Point Cloud, Decentralized Network, Point Cloud Package, DGGS, Dat Project, Dotloom Project**

I. Introduction

Big data has brought many challenges to collect, store, manage, and analyze for service innovation and value creation across sectors in the last decade [1]. A centralized cloud computing environment based on high-performance computer hardware and distributed, parallel computing software like Apache Hadoop[1] and Apache Spark[2] helps to handle a massive amount of data. As addressed in [2], high-volume or high-velocity collection of geospatial data required to change the architecture of typical geographic information systems (GIS) and the role of data platform becomes much important to gather, analyze, distribute, and visualize only the data fit for each specific use.

A point cloud is a collection of points defined by a given coordinates system to represent a three-dimensional (3D) shape or feature with X, Y, and Z coordinates and additional attributes in some cases. Point clouds are most often created by methods used in photogrammetry or remote sensing. The combination of Light Detection and ranging (LiDAR) sensors and positioning systems like a Global Positioning System (GPS) are the most common instruments used to collect point cloud data about the geographic features with intensity and angle. The data type of point cloud has both characteristics of vector and raster data, and it is recently emerging as a fundamental data to make high definition (HD) maps [3]. However, the tremendous volume of the data makes it challenging to exchange and share them among people and machines. An example of data size is about 11 TB of data in a Netherlands (about $40,000km^2$) area, although it varies depending on the detail of the data and type of LiDAR sensors [4]. It is almost 12,000 times the data capacity of OpenStreetMap (OSM), which is a two-dimensional vector map service, in the same area. That is the reason why small pieces of files usually distribute the point data (e.g., $200m \times 200m$) among users. However, it causes other problems in data integration and searches. Moreover, their relative or local coordinate systems depending on locations or sensors obstruct to develop a data ecosystem.

In this paper, we design a new data platform, *Dotloom*, for the distribution and processing of a worldwide scale 3D point cloud data. It allows for instant and direct data sharing and seamless processing with security and efficiency over a peer-to-peer (P2P) network over the Web. In particular, we leverage *Dat Protocol* [5], a decentralized file-sharing protocol designed for syncing folders of data, to make data transfer easy and secure for data producers and consumers. We also use the *Discrete Global Grid System (DGGS)* [6] to integrate and search data collected from multiple locations as a common reference system. DGGS expresses the surface of the earth as a hierarchical tessellation with an equal area. Based on two existing technologies, we define our basic operations to store, sync, share, query, and visualize point cloud packages (PCP) decomposed into primary elements of a set of points.

The remainder of this paper is organized as follows: In Section II, we describe the needs and requirements of the Dotloom. Following this, in Section III, we describe the underlying technology that can be used to implement the platform. Section IV defines a set of functions for achieving platform goals. Finally, in Section V, we summarize the contribution points and discuss the necessary work in the future.

[1]Open-source software for reliable, scalable, distributed computing: http://hadoop.apache.org
[2]Lightning-fast unified analytics engine: https://spark.apache.org

978-1-7281-5606-4/19/$31.00 ©2019 IEEE
DOI 10.1109/ISM46123.2019.00059

II. Requirements

In order to design an architecture of Dotloom, we take a three-tier framework as referring to Fog computing to support multilevel management of large volumes of 3D point cloud data, as shown in Figure 1. Fog computing helps to increase throughput with better security. It reduces both latencies in the edge and the storage needed for big geospatial data in the cloud [7].

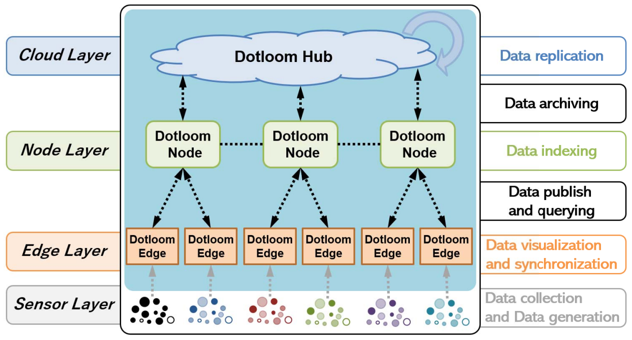

Fig. 1. Requirement of Dotloom

Edge layer: It collects data and client-to-client data interchange. *Dotloom Edge* is an application that requires the support of the following functions:
- Data interchange: There are two ways to collect and secure data across layers; data from the Sensor layer and other Dotloom Edge.
- Data synchronization: When the data update at the original edge, the data at the subscription Edges can be updated in near-real-time.
- Local data processing: It is necessary to filter the data for transmission.

Node layer: *Dotloom Node* organizes Edges into an overlay network and manages the metadata of the datasets in the edge and cloud layer. Also, it creates a distributed index structure for data search with a common coordinate reference system. Finally, it is necessary to act as a broker between Edge and Cloud layers for data archiving.

Cloud layer: *Dotloom Hub* provides long-term storage and data reliability. In the hub, the archived data can be used for gaining deeper insights by using machine learning and deep learning techniques, such as object classification, semantic segmentation, and vectorization.

The platform also needs to visualize and integrate data generated at multiple peers and locations. In particular, we need to consider the following factors for data integration and unified data access:
- Data may use different coordinate reference systems (CRS).
- Duplicate data may be present.
- Data may have different resolutions or level of detail.

To address this issue, we must either convert the CRS of different data into a single unified CRS or use a spatial reference system (SRS) that spans the entire globe. This allows us to access and manage data on a common reference system.

III. Key Technology

This chapter explains the fundamental techniques for satisfying the requirements of the Dotloom platform.

A. Dat Protocol

The Dat protocol [5] is a peer-to-peer protocol, and each peer can be publicly accessed using a unique address on the distributed network. It aims to improve the inefficient performance of existing HTTP based on a centralized structure by adopting a decentralized structure, the design of which is inspired by Git [8], BitTorrent [9], and Dropbox [10]. The Dat protocol has the following characteristics:

Distributed network: The Dat protocol operates in a distributed network environment.

Content integrity: Data and publisher integrity are verified through the use of signed hashes of the content.

Decentralized mirroring: Users sharing the same Dat discover each other and exchange data in a swarm automatically.

Synchronization: The Dat protocol supports full synchronization in real-time using Merkel tree.

Network privacy: The Dat protocol provides certain privacy guarantees, including end-to-end encryption.

Incremental versioning: Datasets can be efficiently synced, even in real-time, to other peers.

Random access: Huge file hierarchies can be efficiently traversed remotely.

B. Discrete Grid Global System (DGGS)

The Discrete Grid Global System (DGGS) [6] is a spatial reference system that represents the Earth as hierarchical sequences of equal area tessellations on the surface of the Earth. DGGS is working on standardization in the Open Geo Consortium (OGC) and the International Organization for Standardization (ISO). DGGS was designed with the following objectives:

Data integration: The goal of the DGGS is to be able to quickly combine spatial data from multiple sources without the difficulty of using the coordinate reference system (CRS).

For everyone to use: DGGS defines the geospatial world much more simply by replacing location information represented by coordinates with cells.

Minimize distortion: Most conventional CRSs (e.g., Geohash) use planar geometry to represent space and projection. However, if we un-project it back onto the surface of the Earth, we can get some distortion in our data (especially in the polar regions). DGGS solves this problem by dividing the surface of the Earth with equal area and shape.

Data analysis of very large data: DGGS has a set of functional algorithms that enable rapid data analysis for vast numbers of cells and are inherently suited for parallel processing.

Efficient data visualization: DGGS has a hierarchical structure, which is suitable for the level of detail (LoD) representation for visualization.

IV. Dotloom

A. Architecture and Design

Dotloom is a decentralized data platform for working with huge point cloud datasets. From data collection from remote sensors to consumer-ready data products, Dotloom strives to make every step of working with point cloud data as secure, simple, and efficient as possible. Dotloom is built with speed, security, and efficiency in mind. The architecture of Dotloom is shown in Figure 2.

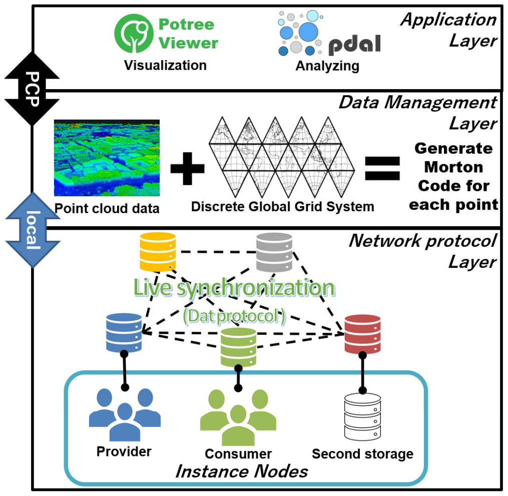

Fig. 2. Dotloom Platform Architecture

1) Data Communication with Dat:

Data communication is performed using the Dat-protocol. The data generated by the providers are shared using a dat-link, and consumers can also fetch data using a dat-link. Also, when there is an update to the shared data, real-time synchronization is performed. These functions are packaged via the Point Cloud Package (PCP) described in the IV-B section.

2) Instance Nodes:

Platform-participating peers perform one or more of three roles: Provider, Consumer, Second storage. The Provider is responsible for creating and sharing point cloud data. The Consumer takes the data generated by the Provider and makes use of it. Secondary storage is used to store data if the Provider no longer wants to supply data on its own.

3) Data Management:

The data is managed by using the Morton code generated using DGGS for the point cloud data as the key value.

4) Applications with PCP:

The primary functions for data utilization are provided through PCP: local data search, global data search, and data visualization. Local data search is a function that searches for point cloud data on the local disk. Global data search is a function to retrieve data stored in Second storage. Data visualization is the ability to visualize data on a local disk or data contained in a shared dat-link.

B. Functionalities

This section discusses the functionality provided by Dotloom. The platform's goal is to store, manage, and service large volumes of 3D point cloud data. To do this, we designed the necessary functions based on the *Point Cloud Package* (PCP); **Create, Publish, Get, Export, Search, View**. PCP is a package created to provide services on this platform. The flow of each designed function is shown in Figure 3. Also Table I shows the open APIs for PCP functions.

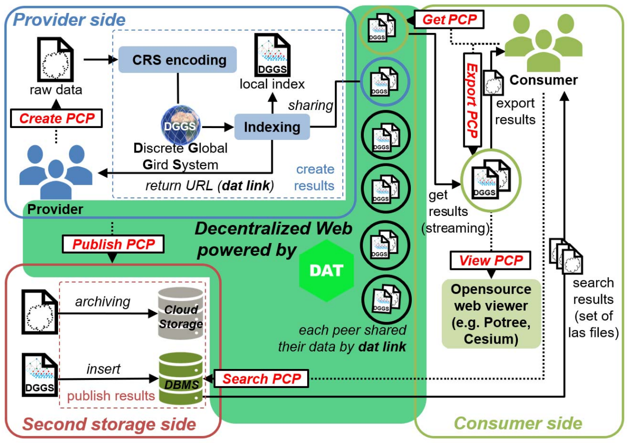

Fig. 3. PCP processing flow in Dotloom

1) Create PCP:

Create PCP generates a PCP to share point cloud data, mainly operated on the provider node. It accompanies two tasks: PCP generation and daemon execution; PCP generation is a step to share a point could data using *Dat protocol*. Daemon execution launches a process that integrates an SRS of each point cloud data, i.e., when the daemon detects an additional data input, it generalizes each coordinate of entered point based on both CRS of point cloud data and DGGS. Note that this function does not make public a PCP; thus, only a node which holds *Dat link* to access is accessible to the created PCP. *Create PCP* is defined in Table I. Field *taeget_dirpath* is the path of stored point cloud data to share, and field *crsinfo* means CRS information of the point cloud data. Finally, *CreatePCP* returns a URL code of *Dat link* as a result.

2) Publish PCP:

Publish PCP is a function for sharing an original point cloud data and DGGS code generated from *Create PCP* to other users. This function is defined in Table I. Field *server_url* is an address of a server that will be

TABLE I
OPEN APIS FOR PCP FUNCTIONS

Name	Description	Mandatory fields
create_pcp	generates a PCP to share point cloud data	taeget_dirpath, crsinfo
publish_pcp	share raw point cloud data with DGGS code	server_url, user_id, user_pw, is_publish
get_pcp	get shared data	target_datlink, dirpath
export_pcp	fetch data which belongs to the desired area from local data	mbb_info, dirpath
search_pcp	retrieve data which belongs to the desired area from second storage	server_url, mbb_info, dirpath
view_pcp	visualizes point cloud data	target_datlink or dirpath

connected. Field *user_id* and *user_pw* are user authentication information, and field *is_publish* determines whether data is shared or not. If it is not shared, the data will be kept in a data storage to maintain data consistency.

3) Get PCP: **Get PCP** is to obtain the shared data under the decentralized network. The obtained data is automatically synchronized when the original storage is updated. This function is designed by the data download and the real-time synchronization functions from *Dat protocol*. *Get PCP* is defined in Table I. Field *target_datlink* is a *dat link* to obtain a PCP, and field *dirpath* is a path of a local storage to save the obtained data.

4) Export PCP: **Export PCP** fetches data which belongs to the desired area from *Get PCP* function. The desired area is represented by a minimum bounding box (MBB). *Export PCP* is defined in Table I. Field *mbb_info* indicates MBB of the desired area, represented by two coordinates of three dimensions; e.g., $(x_1, y_1, z_1, x_2, y_2, z_2)$. It can be retrieved using *Morton* code of DGGS. Field *dirpath* is a path to store a result of a query.

5) Search PCP: **Search PCP** retrieves data which belongs to the desired area from *Publish PCP*, similar to *Export PCP*. This function can cover the entire collection of shared datasets, whereas *Export PCP* only provides local data. *Search PCP* is designed in Table I. Field *server_url* is an address of a server that will be connected. Field *mbb_info* and *dirpath* perform the same roles as in *Export PCP*. It is necessary here that field *mbb_info* in the *Search PCP* be limited to a maximum volume size to prevent network overflow; e.g., less than 0.01 percent of the Earth's total surface area.

6) View PCP: **View PCP** visualizes point cloud data based on Potree and Cesium libraries. These libraries provide various APIs to visualize point cloud data, distributed as open-source software. *View PCP* can be defined as two functions depending on the data source. Field *dirpath* retrieves the data from the local storage for visualization. However, the data will be visualized through a streaming method if the data is specified by field *target_datlink* as *Dat link*. In this case, LOD of the data should be carefully considered in order to limit network load and ensure a smooth visualization. We also take into account the LOD based on the DGGS local index hierarchy.

V. CONCLUSION AND FUTURE WORK

In this paper, we presented the design of a platform for processing, managing, and servicing large three-dimensional point cloud data. First, we summarized the requirements for implementing this platform. Following this, we introduced Dat protocol and Discrete Global Grid System (DGGS), which are key technologies to meet the requirements. Finally, we described in detail the design of the platform based on core technologies and presented the Point Cloud Package (PCP) as a result. We plan to provide this platform as open-source software in the future[3].

ACKNOWLEDGMENT

This work was partially supported by the New Energy and Industrial Technology Development Organization (NEDO).

REFERENCES

[1] R. V. Zicari, "Big data: Challenges and opportunities," *Big data computing*, vol. 1, pp. 103–128, 2014.

[2] J.-G. Lee and M. Kang, "Geospatial big data: Challenges and opportunities," *Big Data Research*, vol. 2, no. 2, pp. 74 – 81, 2015, visions on Big Data.

[3] X. He, J. Zhao, L. Sun, Y. Huang, X. Zhang, J. Li, and C. Ye, "Line-based road structure mapping using multi-beam lidar," *CoRR*, vol. abs/1804.07028, 2018. [Online]. Available: http://arxiv.org/abs/1804.07028

[4] P. van Oosterom, O. Martinez-Rubi, M. Ivanova, M. Horhammer, D. Geringer, S. Ravada, T. Tijssen, M. Kodde, and R. Gonçalves, "Massive point cloud data management: Design, implementation and execution of a point cloud benchmark," *Computers & Graphics*, vol. 49, pp. 92–125, 2015.

[5] M. Ogden, "Dat-distributed dataset synchronization and versioning," *Code for Science & Society*, 2017.

[6] M. B. Purss, R. Gibb, F. Samavati, P. Peterson, and J. Ben, "The ogc® discrete global grid system core standard: A framework for rapid geospatial integration," in *Geoscience and Remote Sensing Symposium (IGARSS), 2016 IEEE International*. New York, USA: IEEE, 2016, pp. 3610–3613.

[7] R. K. Barik, H. Dubey, A. B. Samaddar, R. D. Gupta, and P. K. Ray, "Foggis: Fog computing for geospatial big data analytics," in *2016 IEEE Uttar Pradesh Section International Conf. on Electrical, Computer and Electronics Engineering (UPCON)*, 2016, pp. 613–618.

[8] S. Chacon and B. Straub, *Pro git*. New York, USA: Apress, 2014.

[9] D. Qiu and R. Srikant, "Modeling and performance analysis of bittorrent-like peer-to-peer networks," in *ACM SIGCOMM computer communication review*. New York, USA: ACM, 2004, pp. 367–378.

[10] I. Drago, M. Mellia, M. M Munafo, A. Sperotto, R. Sadre, and A. Pras, "Inside dropbox: understanding personal cloud storage services," in *Proc. of the 12th Internet Measurement Conference*. New York, USA: ACM, 2012, pp. 481–494.

[3]https://github.com/dotloom