Journal of the Korean Association of Geographic Information Studies, 25(4) 2022, pp.32-48 ISSN 1226-9719(Print)
https://doi.org/10.11108/kagis.2022.25.4.032 ISSN 2287-6952(Online)

정밀도로지도의 대용량 공간데이터 교환을 위한 직렬화 기법 설계*

이은일[1] · 김덕호[2※]

Serialization Method for large spatial data transmission of High Definition Map*

Eun-Il LEE[1] · Duck-Ho KIM[2※]

요 약

공간데이터의 활용 범위와 서비스가 늘어나면서 정밀한 공간데이터를 필요로 하는 기술들이 많아지고 있다. 특히 정밀도로지도는 3차원 공간데이터를 수집, 가공, 처리하는 것이 필수적이며 이를 통해 자율주행 지원이 가능하다. 본 연구에서는 정밀도로지도를 대상으로 대용량의 공간데이터를 효율적으로 저장 및 전송할 수 있는 공간데이터 직렬화 기법을 설계하고 구현하였다. 효율적인 직렬화를 위해 바이너리 형태의 공간데이터 구조를 정의하였으며 Zigzag-Z-order 곡선을 활용하여 정보의 손실 없는 좌표 값 인코딩 기법을 설계하였다. 설계한 공간데이터 직렬화 기법을 정밀도로지도 대상으로 구현 및 적용하여 Protocol buffer, Geobuf와 인코딩 후 데이터 크기, 인코딩/디코딩 속도를 비교하였다. 그 결과, 경량화 성능과 인코딩 속도는 모든 유형의 공간데이터에서 설계한 직렬화 방식이 우수한 것을 확인하였다. 하지만 디코딩 속도는 선과 면 유형의 공간데이터에서 다른 직렬화 기법의 성능이 우수하였다. 본 연구를 통해 바이너리 형식의 직렬화 기법으로 공간데이터를 효율적으로 인코딩하여 저장 및 전송할 수 있다는 것을 확인하였다.

주요어 : 정밀도로지도, 직렬화, 공간데이터, 데이터 전송

ABSTRACT

This study presented a spatial data serialization technique that can efficiently store and transmit large amounts of spatial data for precision road maps was designed and implemented. For efficient serialization, a binary spatial data structure is defined, and a

2022년 10월 24일 접수 Received on October 24, 2022 / 2022년 11월 04일 수정 Revised on November 04, 2022 / 2022년 11월 10일 심사완료 Accepted on November 10, 2022

* 이 논문은 2022년도 정부(경찰청)의 재원으로 과학치안진흥센터의 지원을 받아 수행된 연구임 (No.092021C28S02000, 협력적 교통제어전략 도입을 위한 교통정보 음영구간 정보 생성 및 운영관리 기술개발)

1 ㈜아이나비시스템즈, 연구원 / Researcher, Inavi systems
2 한국자동차연구원, 선임연구원 / Senior Researcher, Korea Automotive Technology Institute
※ Corresponding Author E-mail: dhkim2@katech.re.kr

coordinate value encoding technique without loss of information is designed using the Zigzag-Z-order curve. The spatial data serialization technique designed for precision road maps was tested, and the data size and encoding/decoding speed after encoding were compared with Protocol buffer and Geobuff. As a result, it was confirmed that the designed serialization method was excellent in data weight reduction performance and encoding speed. However, the decoding speed was inferior to other serialization techniques in linestring and polygon type spatial data. Through this study, it was confirmed that spatial data can be efficiently encoded, stored, and transmitted using binary serialization techniques.

KEYWORDS : HD-Map, Serialization, Spatialdata, Data transfer

서 론

최근 공간데이터의 활용 범위와 서비스가 늘어나며 자율주행, 스마트시티 등 3차원 정밀한 공간데이터를 활용하는 기술들이 발전하고 있다. 자율주행 기술은 실시간으로 3차원 공간데이터를 수집, 가공, 처리하는 것이 필수적인 기술로, 일부 업체에서는 조건부 자율주행 기술을 개발하여 상용화하는 단계에 있다. 하지만 자율주행 기술의 결함 또는 인적 요인 등으로 자율주행 차량의 사고는 지속적으로 발생하고 있다. 이를 보완하기 위한 기술 중 하나로 정밀도로지도를 통한 자율주행 지원 기술이 개발되고 있다.

정밀도로지도는 차로 수준의 도로망 및 도로시설물 정보에 대해 높은 정밀도로 구축된 지도이다. 정밀도로지도에 구축된 정보는 자율주행 차량의 차량 측위 보조, 경로 안내, 차량 궤적 생성 등에 이용된다. 이러한 자율주행 지원 기능이 원활하게 작동하기 위해서는 신속한 정보 교환이 필수적이다. 하지만 정밀도로지도는 3차원 공간 정보를 가지고 있어 구조가 복잡하고 크기가 방대하다. 따라서, 전송 시간에 대한 요구사항을 맞추기 위해서는 효율적인 공간데이터의 전송 방안이 필요하다.

이와 관련하여 다양한 데이터 직렬화 기법이 개발되었으며, SAE J2735에서는 V2X 통신을 위한 WAVE 메시지 규격에 ASN.1 직렬화 기법을 활용하고 있다. OpenStreetMap은 변환 후 크기와 속도에 장점이 있는 바이너리 형식의 직렬화 기법을 활용하여 지도 데이터를 공유하고 있다. 하지만 대부분 기존 바이너리 직렬화 기법은 별도의 동일한 헤더 파일이 필요하며, 이로 인해 다양한 시스템 간의 상호교환에는 어려움이 있다. 이를 개선하기 위한 연구 중, 공간데이터에 초점을 맞춘 직렬화 기법의 연구 사례는 미비하다.

따라서, 본 연구에서는 대용량 공간데이터에 최적화된 데이터 구조 및 바이너리 형식의 직렬화 기법을 설계하고 구현하고자 한다. 또한, 정밀도로지도 데이터를 대상으로 설계한 직렬화 기법을 적용하고 성능을 분석하여 정밀도로지도 전송 시 제안한 공간데이터 직렬화 기법의 효율성을 검증하고자 한다.

연구 방법 및 동향

1. 연구의 범위와 방법

본 연구에서는 대용량 공간데이터의 효율적인 직렬화 기법을 설계하고, 설계한 기법의 성능 분석을 수행하였다. 이를 위해 공간데이터의 표현 형식과 직렬화 기법에 대한 분석을 진행하였다. 분석한 내용을 바탕으로 바이너리 형태로 공간 정보를 표현하기 위한 데이터 구조 및 바이너리 형식의 인코딩/디코딩 방식을 정의하였다. 설계한 기법을 국토지리정보원에서 제공하는 정밀도로지도에 적용하여 기존 직렬화 기법과의 성능을 비교하여 제시한 기법의 효율성을

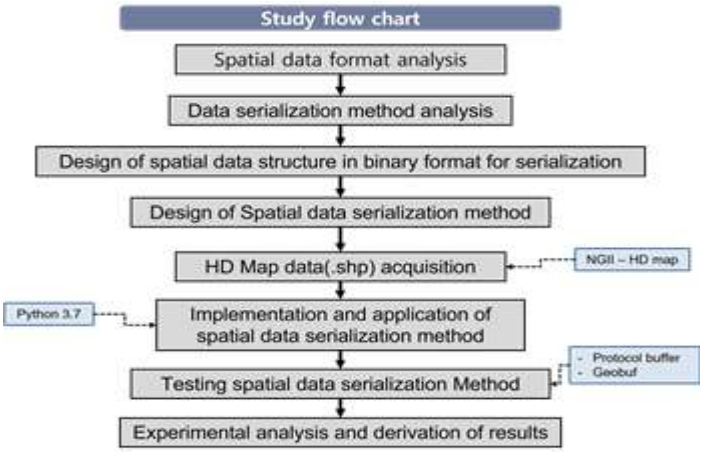

FIGURE 1. Study flow chart

평가하였다. 그림 1은 본 연구의 연구방법을 나타내었다.

2. 선행연구 및 이론적 고찰

1) 정밀도로지도

다수의 기관에서 정밀도로지도를 연구하고 정의하면서 다양한 국제 표준, 산업 표준 등으로 제정된 정밀도로지도 데이터 모델이 존재한다. 국제 표준화 기관인 International Organization for Standardization(ISO)의 Technical Committee (TC) 204에서는 자율주행 지원을 위한 정밀도로지도의 데이터 모델, 지도 데이터 교환 방식 등을 정의하고 있다. 유럽 자동차 회사가 함께 제정한 산업 표준인 Navigation Data Standard Open Lane Model(NDS OLM)에서도 정밀도로지도 모델을 정의하고 있다. 또한 HERE, TOMTOM과 같은 지도 관련 업체에서도 각자의 정밀도로지도 데이터 모델을 정의하여 구축 중에 있다.

양인철, 전우훈(2018)은 정밀도로지도를 기반으로 도로 이벤트 관리 시스템을 개발하여 차로 수준의 이벤트 정보를 생성하여 제공하고자 하였다. 차세대 지능형 교통 체계(C-ITS)에서는 정밀도로지도를 기본 인프라로 활용하여 우회전 안전 주행 지원과 같은 다양한 서비스를 제공한다. 또한 차량 인근의 주행 관련 정보를 공유하기 위해 연구되고 있는 Local Dynamic Map(LDM) 기술은 정밀도로지도를 기본 인프라로 하여 노면표시, 신호등, 차량, 보행자, 사고 등의 정적 및 동적 정보를 융합해 공유한다.

2) 데이터 직렬화

데이터 직렬화는 객체나 메시지와 같이 구조화된 데이터를 메모리에 저장하고 네트워크를 통해 전송될 수 있도록 바이트 또는 비트 단위로 변환하는 프로세스이다(Malin *et al.*, 2011). 데이터 교환 시, 전송 가능한 형태로 데이터를 변환하는 직렬화 과정과 수신 받은 데이터를 원래의 형태로 복원하는 역직렬화 과정을 수반한다. 데이터 직렬화 기법은 변환 형식에 따라 텍스트 기반 직렬화 기법 및 바이너리 기반 직렬화 기법으로 구분할 수 있다.

텍스트 기반의 직렬화 기법은 사람이 읽기 쉬워 오류가 발생했을 시 디버깅에 유리하다는 장점이 있다. 하지만 오버헤드가 많이 발생하여 데이터의 크기가 커지며 데이터 처리 속도가 느리다는 단점이 있다(Krijnen *et al.*, 2017). 대표적인 텍스트 기반 직렬화 형식은 JavaScript Object Notation(JSON)과 eXtensible Markup Language(XML)가 있다. 바이너리 형식의 직렬화 기법은 0과 1로만 표시되어 있어 디코딩 과정 없이 사람이 이해하기 힘들어 오류 발생

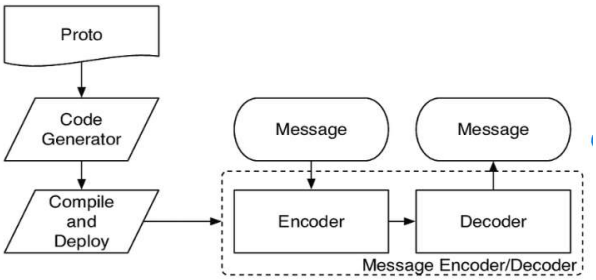

FIGURE 2. Encoding/Decoding process of Protocol buffer (Sayantan Das & Souham Ghosh, 2020)

시 수정이 어렵다는 단점이 있다. 그러나 데이터 변환 속도가 빠르며 문자 기반 직렬화 기법보다 변환 후 데이터 크기가 작다는 장점이 있다(Maeda Kazuaki, 2012). 대표적인 바이너리 형식의 데이터 직렬화 기법은 Protocol buffer, Geobuf, Avro 등이 있다.

Protocol buffer는 구글에서 만든 데이터 직렬화 기법으로 임의의 데이터 구조를 그림2와 같이 '.proto' 파일로 정의하고 데이터를 직렬화한다. 다양한 언어의 API를 지원하고 있어 범용성이 매우 넓지만 '.proto' 파일 없이는 바이너리 형태로 변환된 파일을 해석할 수 없다. 즉, 원활한 데이터 교환을 위해서는 '.proto' 파일이 필수적이라는 특징이 있다.

Geobuf는 Protocol buffer를 기반으로 하여 공간데이터에 최적화된 구조를 정의한 "proto" 파일을 기반으로 직렬화하는 방식이다. shapefile을 바이너리 포맷의 데이터로 변환하거나 바이너리 포맷의 데이터를 GeoJSON 형태로 변환하는 기능을 제공한다.

Avro는 아파치 하둡 프로젝트에서 개발된 데이터 직렬화 프레임워크이다. 데이터의 유형의 명시없이 데이터의 Length만을 통해 데이터를 구분하고 사전 정의된 스카마를 참조하여 데이터 유형을 구분하는 방식이다. 따라서 수신 받은 데이터를 복원하기 위해선 작성자가 사용한 스키마와 정확히 같은 버전의 스키마가 필요하다(Maeda Kazuaki, 2012).

Petersen *et al.*(2017)은 지능형 전력망시스템의 제어 명령에 사용하는 메시지를 활용하여 12개의 직렬화 기법을 비교하였다. 그 결과, 문자 기반 직렬화 방식은 사람이 이해하기 쉽고 간결하나 속도, 용량과 같은 성능은 바이너리 기반 직렬화 방식이 더 뛰어났다.

미국자동차공학회의에서 발간한 SAE J2735 표준에서는 교통정보 센터과 차량 간의 메시지 교환(V2X)을 위해 Dedicated Short Range Communications of Wireless Access in Vehicular Environments(DSRC/WAVE) 기반 메시지 규격 및 데이터 요소를 정의하고 있다. 메시지 규격은 ASN.1 직렬화 방식 기반으로 정의되어 있으며 지도 데이터가 포함된 메시지도 기술되어 있다. 하지만 해당 규격은 DSRC/WAVE 통신 외 프로토콜에 적용하기 어렵다는 단점이 있다.

이처럼 선행연구에서는 정밀도로지도를 대상으로 하여 직렬화 성능을 비교하거나, 다양한 시스템 간 호환이 가능한 새로운 기법을 제시한 사례가 미비하였다. 따라서, 정밀도로지도 데이터를 대상으로 하고, 바이너리 형태의 구조를 가지며 다양한 시스템에서 적용 가능한 인코딩/디코딩 방식을 가진 공간데이터의 직렬화 기법을 새로 설계하고자 한다. 이를 통해 기존 선행연구와 직렬화 기법이 가지는 일부 한계점을 보완할 수 있을 것이다.

공간데이터 직렬화 기법

1. 공간데이터 직렬화 구조 설계

1) 공간데이터 표현 방식 분석

다양한 시스템 간 원활한 데이터 교환을 위해서는 표준화된 데이터 형식을 정의하고 준용하는 것이 유리하다. Open Geospatial Consortium (OGC)과 ISO에서는 공간데이터를 저장하고 접근하는 방식을 표준화하여 정의하고 있다.

ISO 19125-1 Geographic information - Simple feature access - Part 1: Common architecture에서는 Point, Curve와 같은 공간 객체의 형상 정보를 Simple Feature Geometry라는 표준화된 아키텍처로 정의하고 있다. 해당 문서에서 정의한 공간 객체 형상 모델은 형상 유형, 공간 참조 체계 등 형상 관련 정보를 포함한 형태이다. 또한 형상을 바이너리 또는 문자 형식으로 표현하기 위해 Well-Known Binary (WKB), Well-Known Text(WKT)라는 인스턴스를 정의하고 있다.

WKT란 지도, 공간 객체의 공간 참조 시스템 간 변환을 표현하기 위한 텍스트 마크업 언어이다. 형상 유형에 따른 기하 정보 표현 형식을 정의하고 있으며 2차원(x, y), 3차원(x, y ,z), 측정 데이터가 있는 2차원(x, y, m), 측정 데이터가 있는 3차원(x, y, z, m)으로 구분한다.

WKB는 연속적인 바이트 스트림 형태로 공간 객체의 형상을 정의하는 방식이다. 바이너리 기반인 WKB를 통해 공간 좌표 데이터를 효율적으로 저장하고 교환할 수 있다. WKB 형식은 그림 3과 같이 바이트 정렬(Endian 표시), 형상 유형, 공간 좌표 데이터 순으로 스트림되는 구조이다. 바이트 정렬은 1 byte의 정수형을 가지며 형상 유형은 코드화한 형태로 4 byte 크기의 정수형으로 정의된다. 공간 좌표 데이터는 각각의 좌표 값이 8 byte의 실수형으로 정의된다.

Oracle, SQLite 등의 여러 상용 DBMS, API, Protocol에서 해당 표준을 준용하여 공간 정보 표현의 기본 형태로 사용하고 있다. 대부분의 공간데이터베이스를 지원하는 DBMS에서 공통적으로 WKB 형식을 사용하고 있으나, 사용할 수 있는 공간 객체의 유형, 저장 방식, 공간 객체를 다루기 위한 함수 이용방식이 상이한 면이 있다.

Oracle, MySQL에서 사용하는 WKB 형식의 공간 데이터 저장 구조는 바이트 순서(1 byte unsigned integer), 형상 유형(4 byte unsigned integer), 좌표 값(8 byte double)으로 구성된다. 또한 공간 참조 체계(Spatial Reference Identifier, SRID)를 포함하여 WKB 형식으로 공간 객체를 정의할 수 있는 형태이다. 각 DBMS에서 처리할 수 있는 공간 개체 유형은 7종류(Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon, GeometryCollection)이다.

PostgresSQL은 공간 쿼리를 위해 PostGIS라는 확장 프로그램을 사용하고 있으며 WKT, WKB를 확장한 Extended Well-Known Text (EWKT), Extended Well-Known Binary (EWKB) 형식을 정의하고 있다. EWKT 및 EWKB는 기본적인 구조는 WKT 또는 WKB와 동일하나 다른 DBMS에서 처리 가능한 7개의 공간 객체 외 3차원 정보를 추가적으로 표현 및 처리할 수 있도록 확장한 형식이다.

2) 바이너리 기반 공간데이터 직렬화 구조 설계

분석한 공간 데이터베이스들은 공간데이터 처리를 위해 관련 표준을 준용하고 있으며 데이터베이스의 특성에 따라 WKT, WKB의 구조를 일부 변형하고 있다. 이러한 형태로 인해 다양한 데이터베이스 간의 공간데이터 교환 및 처리

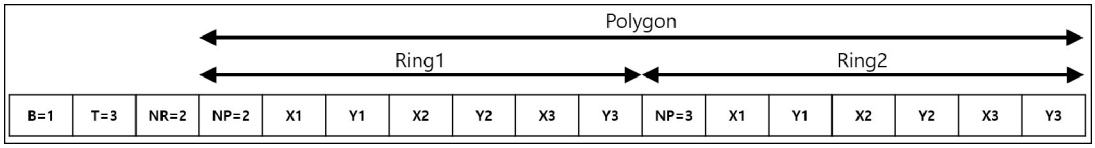

FIGURE 3. WKB example(ISO 19125-1)

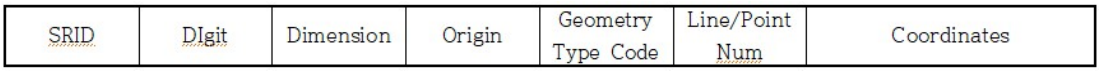

FIGURE 4. Spatial Data Serialization Structure

가 가능하다. 본 연구에서 설계하고자 하는 공간데이터 직렬화 구조는 ISO 19125-1에서 정의한 WKB 형식을 최대한 준용하여 호환이 가능한 구조로 정의하고자 한다.

우선, 공간 데이터베이스 사례를 통해 공통적으로 저장하고 있는 데이터 항목을 분석하여 공간 객체의 형상 정보를 바이너리 형식으로 나타내기 위한 필수적인 항목을 선정하였다. 선정된 데이터 항목은 공간 참조 체계와 형상 유형, 기하를 정의하는데 필요한 선/점의 수, 좌표 데이터이다. 이 외에 효율적인 공간데이터 직렬화를 위한 유효숫자 데이터, 좌표 값들의 차원 데이터, 새로운 원점 데이터 항목을 추가하였다.

선정한 데이터 항목을 바이너리 형태로 스트림하기 위한 공간데이터 직렬화 구조는 그림 4와 같이 설계하였다. 가장 먼저 공간 참조 체계(SRID), 유효 숫자(Digit), 차원(Dimension), 원점(Origin), 형상 유형(Geometry Type Code) 순으로 데이터가 나열된다. 다음으로 형상 유형에 따른 선과 점의 수(LineNum, PointNum), 좌표 값(Coordinates) 데이터 순으로 스트림되는 구조이다.

공간데이터 직렬화 구조에 포함된 데이터 항목별 정의, 데이터 유형 및 크기는 표 1과 같다.

공간 참조 체계(SRID)는 다양한 상용 DMBS에서도 사용하고 있는 방식인 EPSG code로 표기한다. EPSG Code는 European Petroleum Survey Group(EPSG)에서 전 세계의 다양한 좌표계를 표준 코드화한 것이다. WGS 84 타원체의 경위도 좌표계를 나타내는 경우 'SRID = 4326' 이다. EPSG Code는 복잡한 좌표계를 간단한 형태로 나타낼 수 있다는 장점이 있어 이를 활용하였으며 4 byte의 크기로 정의하였다.

유효숫자(Digit)는 좌표 값의 정밀도를 나타내고 실수형의 좌표 값을 정수형으로 변환하여 저장하기 위한 데이터 항목이며 1 byte의 크기를 갖는다. 유효숫자 데이터의 값이 양의 정수인 경우, 소수점 위치를 나타내며 음의 정수인 경우 반올림된 자리수를 의미한다. 예를 들어 (123.45, 987.65) 좌표는 'Digit = 1' 인 경우 좌표를 소수 아래 첫 번째 자리까지가 유효숫자임을 나타내며 (123.5, 987.7)로 변환되고, 'Digit = -2' 인 경우, (120, 990)로 반올림된다.

차원(Dimension)은 공간 객체의 좌표 값이 표현하는 공간적인 수준 또는 축의 개수를 나타내기 위한 데이터 항목이다. 2차원의 좌표 값들을 가지고 있는 경우, 차원은 2의 값을 가지며 3차원인 경우 3으로 저장된다. 차원은 값의 범위가 매우 작을 것으로 판단하여 최소한의 크기인 1 byte로 정의하였다.

원점(Origin)은 가변형 정수로 좌표 값들을 저장할 때 값의 크기를 축소하여 공간데이터의 경량화를 위한 데이터 항목이다. 일반적으로 원

TABLE 1. Data items for spatial data serialization

Data	Description	Type	Size (byte)
SRID	EPSG Code	Integer	4
Digit	Coordinate value precision	Integer	1
Dimension	Coordinate dimension	Integer	1
Origin	The coordinates of the center of the spatial extent	Float	4 per coordinate
Geometry Type Code	Geometry type code of spatial object(WKB)	Integer	1
Line / Point Num	The number of lines or points needed to define the shape	Integer	4 per object
Coordinates	Coordinate values as many as the number of lines or points	Variable Integer	-

점은 공간적 범위의 서쪽 끝, 남쪽 끝, 하단 좌표 또는 중심 좌표로 정의한다. 서쪽 끝, 남쪽 끝, 하단 좌표를 원점으로 하는 경우, 음수없이 좌표를 표현할 수 있다는 장점이 있지만 좌표 값들의 크기가 커진다는 단점이 있다. 중심 좌표를 원점으로 하는 경우, 음수를 사용해야 하지만 좌표들의 절대 값 크기는 작아진다는 장점이 있다.

제안한 공간데이터 직렬화 구조에서는 가변형 정수를 사용하여 인코딩하므로 경량화를 위해서는 좌표 값의 크기를 줄이는 것이 중요하다. 따라서 원점은 공간데이터의 좌표 값이 나타내는 공간적 범위의 중심 좌표 값으로 정의하였으며 각 좌표 값은 4 byte의 크기를 갖는다. 각 축에서의 최대값($coord_{\max}$)과 최소값($coord_{\min}$)을 통해 $(coord_{\min}+coord_{\max})\div 2$로 계산하여 원점을 구하며 유효숫자 내에서 정확한 중심 좌표 값을 나타낼 수 없는 경우, 양의 방향으로 올림한 값을 원점으로 정의한다. 예를 들어 공간데이터의 좌표 값이 1의 자리까지 유효숫자를 가지며 범위가 x축 1~100, y축 −50~50, z축 10~100인 경우 원점은 (51, 0, 55)의 값을 가진다.

형상 유형 코드(Geometry Type Code)는 WKB에서 기술되어있는 유형별 코드 값을 인용하였다. 제안하는 공간데이터 직렬화 구조에서는 Dimension 데이터를 통하여 2차원의 점 유형과 3차원의 점 유형을 구분할 수 있다. 따라서 Geometry Type Code에서는 표 2와 같이 8가지 형상 유형만을 구분하며 1 byte의 크기를 갖도록 정의하였다.

TABLE 2. Geometry Type Code List

Type	Code
Geometry	0
Point	1
LineString	2
Polygon	3
MultiPoint	4
MultiLineString	5
MultiPolygon	6
GeometryCollection	7

형상 유형별 선 또는 점의 수(Line/Point Number)는 정의된 형상 유형에 따라 해당 형상을 정의하기 위한 선의 수(LineNum), 점의 수(PointNum) 데이터가 스트림 되는 데이터 항목이며, 좌표 값(Coordinates)은 실제 위치를 나타내는 값이다. 두 항목은 다수의 공간 데이터베이스와 동일하게 WKB 형식과 동일한 순서로 구조를 갖도록 정의하였다. 예를 들어 'Geometry Type Code = 1' 인 경우 Point를 나타내며 LineNum이나 PointNum없이 바로 좌표 값(Coordinates)이 스트림 된다. 그러나 'Geometry Type Code = 2' 인 경우, Linetring을 나타내며 LineString을 구성하는 점의 수(PointNum), 좌표 값(Coordinates) 순서로 스트림 된다.

2. 공간데이터 인코딩 방식 설계

공간데이터는 객체의 위치를 정의하기 위한 좌표 값 데이터를 가지고 있으며, 좌표 값 데이터로 인해 데이터 용량이 방대해진다. 공간데이터를 효율적으로 직렬화하기 위해서는 좌표 값 데이터를 경량화한 형태로 인코딩하는 과정이 필수적이다. 따라서 본 연구에서는 앞서 정의한 공간데이터 직렬화 구조를 기반으로 좌표 값들이 가변형 정수로 인코딩 되도록 설계하였다. 가변형 정수를 통한 효율적인 좌표 값 저장을 위해 좌표 값들을 작은 값으로 변환하는 인코딩 방안, 실수형을 고정형 정수로 변환하는 인코딩 방안, 마지막으로 2차원의 고정형 정수로 변환된 좌표 값을 1차원의 가변형 정수로 인코딩하는 방안을 정의하였다. 이러한 방식을 통해 정보의 손실 없이 공간데이터의 크기를 최소화하고자 한다.

좌표 값들의 크기를 경량화한 형태로 변환하기 위해 첫 번째, 앞서 정의한 공간데이터 직렬화 구조의 데이터 항목인 원점(Origin) 데이터를 새로운 원점 좌표로 가정하여 좌표를 변환한다. 두 번째, LineString과 같이 하나의 형상을 정의하기 위해 다수의 Point 좌표가 필요한 경우, 직전 Point 좌표 값과의 차이 값으로 변환

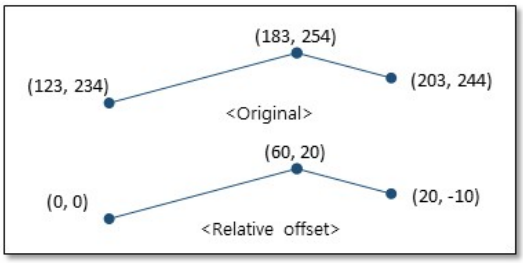

FIGURE 5. Example of coordinate value encoding

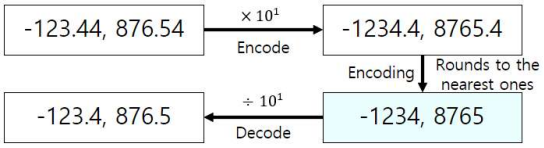

FIGURE 6. Conversion of real number type to integer type through significant digits

하도록 설계하였다.

그림 5는 해당 과정을 통해 3개의 Vertex를 갖는 LineString의 좌표 값을 변환하는 예시를 나타내고 있다. 좌측 Vertex의 좌표 값인 (123, 234)를 원점(Origin) 데이터로 가정하여 첫 번째 Vertex의 좌표가 (0, 0)으로 변환되었다. 이어지는 Vertex들의 좌표는 직전 Vertex 좌표와의 차이 값인 (60, 20), (20, −10)으로 변환된다.

세 번째, 유효숫자(Digit) 데이터를 통해 작은 값으로 변환된 실수형의 좌표 값을 정수형으로 변환한다. 직렬화 시에는 각각의 좌표 값에 10^{digit}만큼 곱한 뒤 소수부는 버리고 반올림한 정수부만을 남겨 이를 정수형으로 저장한다. 역직렬화 시에는 정수형으로 저장되어 있는 좌표 값에 10^{digit}만큼 나눠 유효숫자만큼의 좌표 값을 가지는 데이터로 복원한다.

그림 6과 같이 좌표 값이 (123.44, 876.54)이며 'Digit = 1' 인 경우 모든 좌표 값에 10을 곱하여 반올림한 정수부만 남기면 (1234, 8765)로 변환된다. 이러한 방법을 통해 유효숫자인 소수점 아래 첫 번째 자리까지 정보의 손실 없이 정수형으로 변환할 수 있다. 또한 이를 역직렬화 하는 경우, 다시 10을 나누면 본래 유효숫자까지의 좌표 값인 (123.4, 876.5)로 돌아온다.

네 번째, 'ZigZag−Z−order 곡선' 을 적용하여 2차원 위치를 나타내는 2개의 좌표 값을 단일 값으로 병합한다. ZigZag−Z−order 곡선은 정수로 표현된 2차원의 값을 0 이상의 하나의 정수로 변환하기 위해 설계한 방법이며, 각 좌표 값에 ZigZag encoding 방식을 적용한 후 Z−order curve를 활용하는 방법이다.

ZigZag 인코딩 방식은 값들의 부호를 나타내기 위해 절대값이 낮은 정수부터 0 - 음수 - 양수 순서에 따라 Unsigned integer 자료형으로 인코딩 하는 방식이며, 부호를 나타내는 Sign 비트 없이 값들을 나타낼 수 있어 오버 헤드가 거의 없다. 예를 들어 그림 7과 같이 (0, −1, 1, −2, 2, …) 순으로 이어진 데이터에 ZigZag 인코딩을 적용하면 (0, 1, 2, 3, 4, …)로 변환되며, 가변형 정수를 이용하는 경우 1 byte를 통해 -64부터 63까지의 정수를 나타낼 수 있다. 이러한 장점 때문에 Protocol buffer 및 Avro에서도 ZigZag 인코딩 방식을 사용하고 있다.

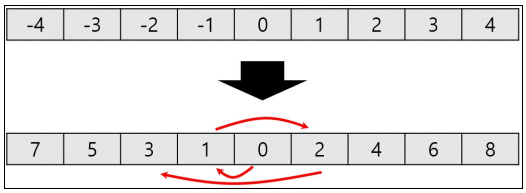

FIGURE 7. ZigZag encoding Example

Z−order 곡선은 2차원 이상의 공간을 일정한 간격으로 분할하여 선형으로 배열하는 공간 채움 곡선(Space Filling Curve) 방식 중 하나이다. 제안한 공간데이터 인코딩 방식은 좌표 값들을 정수형으로 변환하였기 때문에 일정한 간격으로 분할이 되는 구조이므로 이를 활용할

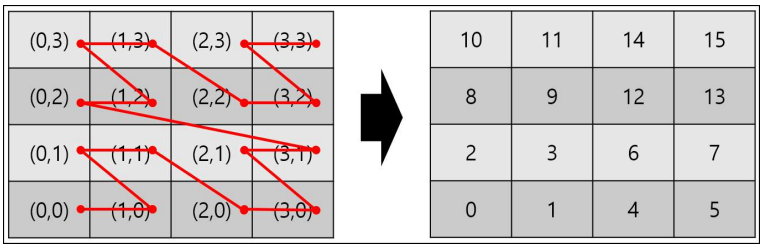

FIGURE 8. Encoding based on Z-order curve

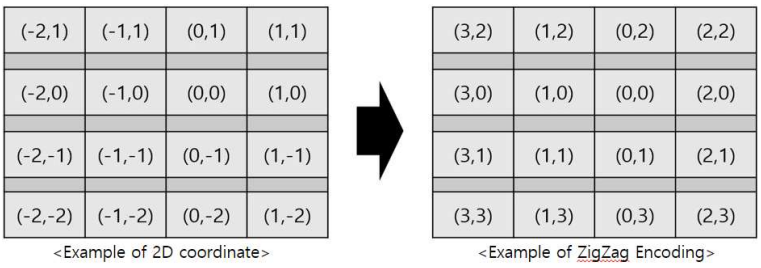

FIGURE 9. ZigZag-Z-order Example 1 - ZigZag Encoding

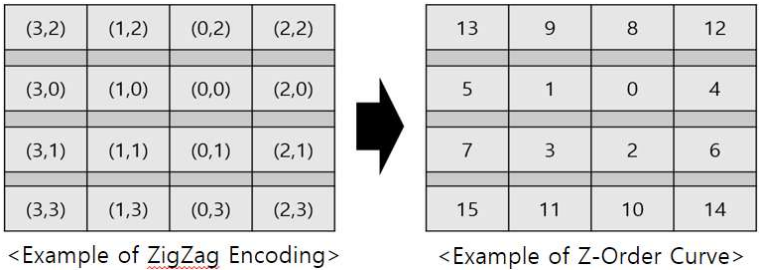

FIGURE 10. ZigZag-Z-order Example 2 - Z-order Curve

수 있다. Z-order 곡선은 바이너리 형태로 변환한 2개의 값을 한 자리씩 교차하는 bit-interleaving 방식을 활용하여 2차원의 값을 하나의 값으로 병합한다. 그림 8와 같이 (2,1)의 좌표에 Z-order 방식을 적용하면 ($10_{(2)}$, $01_{(2)}$)에서 y2,x2,y1,x1 순으로 교차하여 $6(0110_{(2)})$로 변환된다. Z-order 곡선은 간단하게 인코딩·디코딩이 가능하며 원점 인근에 위치한 좌표부터 채워나간다는 특징이 있어 이를 적용하였다.

(-2, -2)부터 (1,1)까지의 범위를 갖는 좌표에 대해 ZigZag-Z-order 곡선을 적용하면 우선 그림 9과 같이 ZigZag encoding이 적용되어 모든 좌표의 값이 0과 양의 정수로 나타난다. 다음으로 Z-order 곡선을 활용하여 그림 10과 같이 변환된 값을 가지게 된다.

마지막으로 정수로 변환된 값들의 크기를 최소화하기 위하여 가변형 정수로 저장되도록 정의하였다. 가변형 정수의 구조는 구분자(Separator)와 값(Value)로 이루어져 바이트 단위로 구성된 형태이다. 각 좌표 값은 바이너리 형태로 변환되어 값의 크기에 따라 최소한의 바이트를 사용하여 나타낼 수 있게 하였다.

구분자(Separator)는 1bit의 크기를 가지며

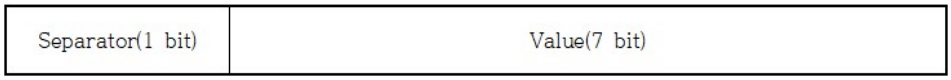

FIGURE 11. Variable Integer Structure

하나의 정수를 표현하기 위해 연결된 Byte가 있는지를 표시하는 자리다. 하나의 정수를 표현한 연결된 Byte가 있는 경우 구분자는 '1' 의 값을 가지며 마지막 Byte인 경우 구분자는 '0' 의 값을 갖는다. 값(Value)은 7bit의 크기를 가지며 실제 좌표 값 데이터를 저장한다.

가변형 정수로 '657' 을 나타내면 바이너리 형태인 '10 1001 0001' 로 변환되고 이는 10bit로 표현이 가능하다. 이러한 경우 하나의 Byte로는 표현이 불가하기 때문에 2개의 Byte로 변환된다. 첫 번째 Byte는 그림 11의 구조에 따라 'Separator = 0' , 'Value = 000 0101' 로 변환되며 두 번째 Byte는 'Separator = 1' , 'Value = 001 0001' 로 변환된다. 즉, 최종적으로 657의 좌표 값은 '0000 0101' , '1001 0001' 로 변환되어 2 Byte의 크기를 가진다. 이를 통해 본래 4 byte의 크기를 가지는 int형보다 더 경량화된 형태를 가지게 된다.

공간데이터 직렬화 성능 분석

설계한 공간데이터 직렬화 기법을 Python 기반으로 구현하여 성능을 분석해보고자 한다. 정밀도로지도 데이터를 대상으로 설계한 직렬화 기법을 적용하여 실험을 진행하였다. 또한 바이너리 형태로 변환된 데이터를 역직렬화 하여 원래의 형태로 복원하는 실험을 진행하였다. 이를 통해 알고리즘이 올바르게 구현됐는지 확인하고 여러 직렬화 기법 간 데이터 크기, 인코딩/디코딩 시간을 비교하였다.

1. 사용 데이터

국토지리정보원에서 제공하는 정밀도로지도 데이터를 사용하여 공간데이터 직렬화 성능 분석하고자 한다. 국토지리정보원에서는 국내의 일부 고속국도 및 일반국도를 대상으로 하여 정밀도로지도를 구축 및 제공하고 있다. 2019년 기준 정밀도로지도 데이터 모델은 차로 수준의 도로망과 인근 도로 시설물을 표현할 수 있도록 총 14개의 객체로 구성되어 있다. 주행경로노드(A1_NODE) 및 주행경로링크(A2_LINK) 등의 객체로 차로 수준 도로망을 구축하고 있으며, 안전표지(B1_SAFETYSIGN), 노면선표시(B2_SURFACELINEMARK) 등의 객체로 도로 시설물을 나타내고 있다.

공간데이터 직렬화 기법의 성능을 평가하기 위해서 공간적 범위가 넓은 지역에 대한 데이터와 좁은 공간적 범위를 가지고 있는 경우로 구분해 분석하는 것이 적합하다고 판단하였다. 또한 점, 선, 면에 따라 설계한 직렬화 기법의 성

TABLE 3. HD Map data model of National Geographic Information Institute of South Korea - Ver. 2019

Item	Geometry Type	Definition
A1_NODE	Point	The connection point of the A2_LINK
A2_LINK	Linestring	Lane centerline
A3_DRIVEWAYSECTION	Polygon	Road structures as part of the road (ex. tunnel, bridge)
B1_SAFETYSIGN	Point	Safety signs defined in the Road Traffic Act and the Enforcement Regulations of the Road Traffic Act are described
B2_SURFACELINEMARK	Linestring	Road marking in the form of lines
B3_SURFACEMARK	Polygon	Non-line road markings
C1_TRAFFICLIGHT	Point	Traffic light as a traffic safety facility

능이 달라질 수 있으므로 다양한 유형으로 정의되어 있는 자료를 이용하였다. 2019년 데이터 모델로 구축되고 넓은 공간적 범위를 가지는 구간인 경부고속도로(약 416.1㎞) 데이터와 좁은 공간적 범위를 가지는 대구 테크노폴리스 인근 도로망 데이터(.shp)를 취득하였다.

공간 객체는 자신의 형상 유형에 따라 형상을 정의하는 구조가 상이하므로 점, 선, 면의 형상 유형에 따른 공간데이터 직렬화 기법의 성능 실험을 진행하였다. 또한 데이터의 크기가 클수록 인코딩 및 디코딩 시간의 오차가 적을 것으로 판단하여 형상 유형별 용량이 가장 큰 주행경로 노드(A1_NODE), 주행경로링크(A2_LINK), 구간(A3_DRIVEWAYSECTION) 데이터를 실험 대상으로 선정하였다.

2. 공간데이터 직렬화 기법 구현

Python 3.7 버전으로 기반으로 설계한 공간데이터 직렬화 기법의 인코딩/디코딩 알고리즘을 구현하여 취득한 정밀도로지도 데이터에 적용하였다. 공간데이터 직렬화 알고리즘 구현에 사용한 컴퓨터 제반 사항은 표 4와 같다.

구현한 공간데이터 직렬화 알고리즘은 shapefile 데이터를 읽고 SRID, Digit, Dimension, Origin, Geometry type code를 바이너리 형태로 스트림 하도록 하였다. 그 후 형상 유형별 기하 정의 구조에 따라 앞서 제시한 경량화 인코딩 방식을 적용한 좌표 값들을 변환하고, 이를 스트림하여 바이너리 파일을 출력하는 형태로 구현하였다. 역직렬화 알고리즘은 변환된 바이너리 파일을 불러 데이터를 읽고 앞의 바이트부터 원래의 데이터 값을 가지도록 복원한 뒤 GeoJSON 포맷으로 변환되도록 구현하였다.

구현한 공간데이터 직렬화 기법을 적용하여 .shp 형식의 정밀도로지도 데이터를 변환한 뒤 바이너리 파일로 출력한 결과는 그림 12와 같다. 첫 번째 byte부터 네 번째 byte까지는 32652의 SRID 값을 나타내며 다섯 번째 byte는 digit, 여섯 번째 byte는 dimension 값을 저장하고 있다. 그 이후로 4 byte씩 Origin 좌표 값과 가변형 정수로 저장된 좌표 값을 확인할 수 있다. 또한 바이너리 포맷의 파일을 geojson 포맷으로 디코딩하였을 때, digit 값에 따른 유

TABLE 4. test environment

HW	Name
CPU	Intel Xeon Gold 5122 Processor 16.5M * 2, 2.60GHz, 4Core
RAM	128GB
GPU	NVIDIA Geforce RTX 2080 Ti * 2
OS	Ubuntu 18.04

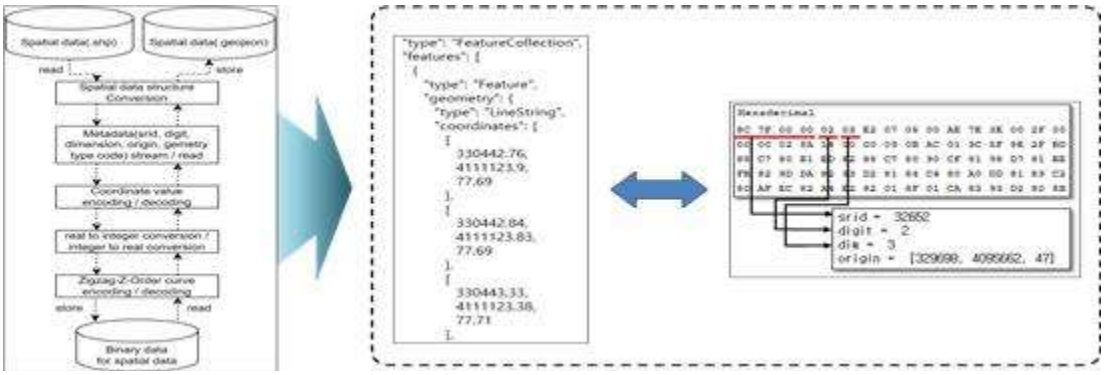

FIGURE 12. Data serialization process

효숫자의 좌표 값으로 공간데이터가 복원되는 것을 확인하였다.

3. 직렬화 기법 간 성능 비교

설계한 공간데이터 직렬화 기법의 성능을 검증하기 위하여 기존의 바이너리 기반 직렬화 방식인 Protobuf, Geobuf와의 성능을 비교하였다. 또한 공간데이터 직렬화 기법을 적용한 파일에 gzip, zlib, lzma와 같은 무손실 압축 알고리즘을 적용한 결과와도 비교를 진행하였다.

실험 과정은 그림 13과 같이 .shp 파일을 각각의 직렬화 기법을 적용한 경우와 바이너리 형태로 변환된 파일을 GeoJSON 포맷으로 역직렬화한 경우로 나눠 진행하였다. 제안한 직렬화 기법은 정밀도로지도에서 요구하는 정밀도를 맞추기 위해 Dimension은 3(3차원), Digit은 2(0.01m 수준)로 설정하여 변환하였다. 직렬화 기법 적용 후 데이터 크기, 인코딩 / 디코딩 속도를 측정하여 성능을 비교하였다. 인코딩 / 디코딩 속도의 경우 100번 실행하여 얻은 수치의 평균값으로 나타냈다.

1) 데이터 경량화

점(Point)의 형상 유형을 가지는 주행경로노드 데이터는 그림 14, 그림 15와 같이 제안한 공간데이터 직렬화 기법을 사용하였을 때 압축률이 가장 높게 나타났다. 공간적 범위에 따른 큰 성능 차이를 보이지 않았으며 제시한 공간데이터 직렬화 기법 사용 후 데이터 크기(New_Geo_bin)는 경부고속도로 데이터의 경우 약 100.4kb, 대구 테크노폴리스 데이터는 약 3.3kb로 경량화 되었다. 이는 .shp파일 대비 압축률이 약 75.8%, 79.9% 로 나타났으며 Geobuf보다도 압축률이 높은 것을 볼 수 있다.

선(LineString)의 형상 유형을 가지는 주행경로링크 객체는 그림 16, 그림 17과 같이 제시한 공간데이터 직렬화 기법을 적용하였을 때 Protocol buffer보다 압축률이 높지만 Geobuf 기법과는 압축률이 거의 동일하였다. 제시한 직렬화 기법을 적용해 변환한 경우 .shp파일 대비 압축률이 약 88.7%에 이르는 것으로 나타났다. 압축률이 Geobuf와 비슷한 이유는 선 유형 공간 데이터의 경우 Geobuf에서도 바로 직전 좌

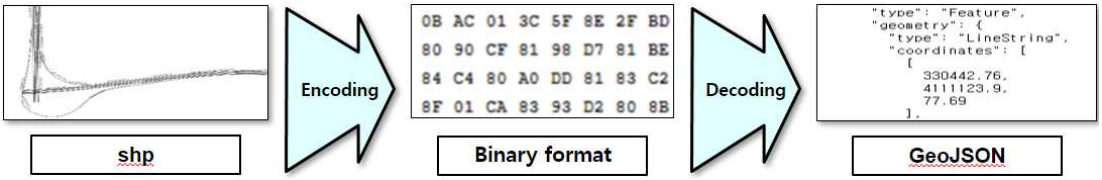

FIGURE 13. HD map data(.shp) encoding process

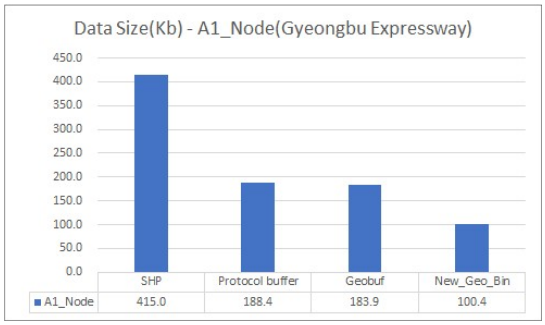

FIGURE 14. A1_NODE(point) data size after encoding -Gyeongbu Expressway

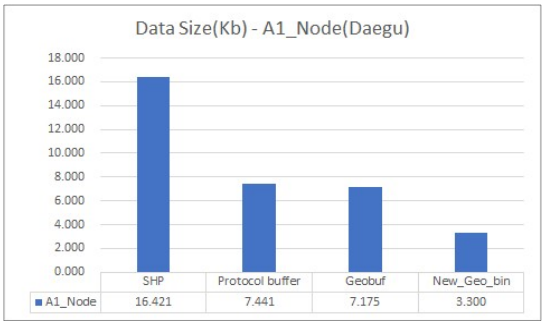

FIGURE 15. A1_NODE(point) data size after encoding - Daegu

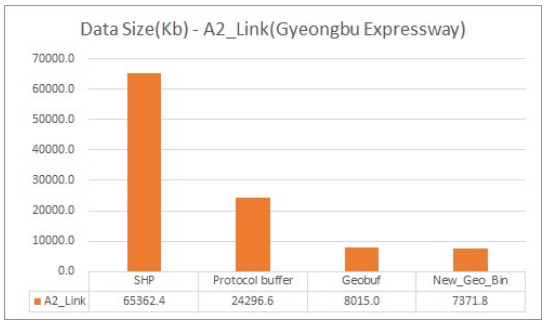

FIGURE 16. A2_LINK(linestring) data size after encoding – Gyeongbu Expressway

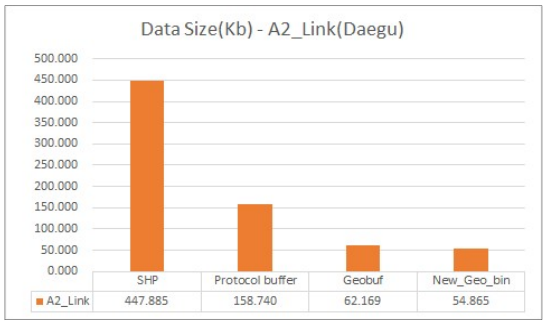

FIGURE 17. A2_LINK(linestring) data size after encoding – Daegu

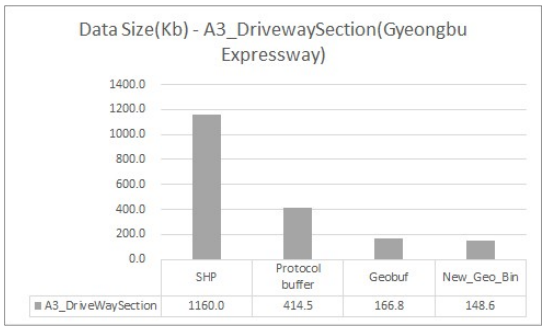

FIGURE 18. A3_Drivewaysection(polygon) data size after encoding - Gyeongbu Expressway

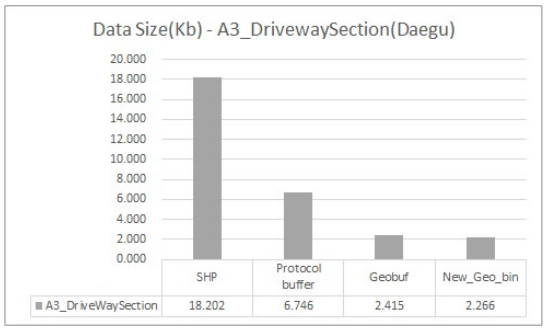

FIGURE 19. A3_Drivewaysection(polygon) data size after encoding - Daegu

표 값과의 Offset값으로 변환하기 때문으로 분석된다.

수치상으로 선 유형의 공간데이터에 바이너리 기반 직렬화 기법을 적용하였을 때 다른 형상 유형의 공간데이터보다 높은 압축률을 보이고 있으나 공간 객체들의 분포 상태에 따라 압축률은 다르게 나타날 것으로 예상된다. 공간적 범위의 경우 점 유형에서의 결과와 동일하게 넓은 지역과 좁은 지역 모두 비슷한 성능을 보인다.

그림 18, 그림 19와 같이 면의 형상 유형을 가지는 구간 객체는 선 유형 객체와 유사하게 제시한 직렬화 기법의 압축률은 약 87.2%이며 Geobuf와 약간 더 좋은 수치를 보인다. 압축률이 비슷한 이유는 선 유형 객체에서의 원인과 동일한 것으로 분석된다. 면 유형 또한 다른 형상 유형과 동일하게 공간적 범위에 따른 큰 성능 차이는 없는 것을 확인하였다.

2) 인코딩/디코딩 속도

주행경로노드 객체에 각 직렬화 기법을 적용하여 인코딩/디코딩 속도는 그림 20, 그림 21과 같다. 제안한 공간데이터 직렬화 기법이 Protocol buffer와 Geobuf 대비 인코딩 속도는 향상되었으며 디코딩 속도 모두 유사한 수치를 나타냈다. 경부고속도로 데이터의 인코딩 속도는 Geobuf 대비 약 19.7% 빨라 전체 인코딩/디코딩 속도에서도 가장 우수한 성능을 보였다. 대구 테크노폴리스 데이터도 여러 직렬화 기법 중 가장 빠른 수치를 보이므로, 점의 형상 유형을 갖는 공간 객체는 제안한 직렬화 기법을 사용하는 것이 가장 효율적인 것으로 판단된다.

선의 유형을 갖는 주행경로링크는 그림 22,

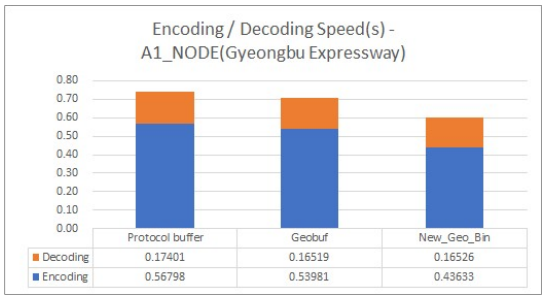

FIGURE 20. A1_NODE(point) encoding / decoding speed – Gyeongbu Expressway

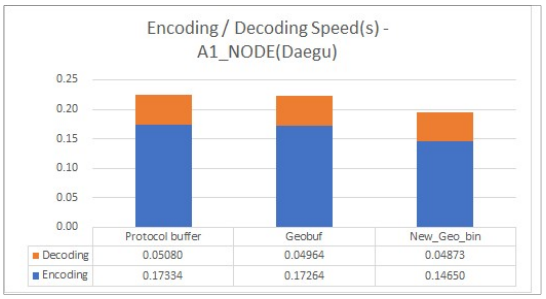

FIGURE 21. A1_NODE(point) encoding / decoding speed – Daegu

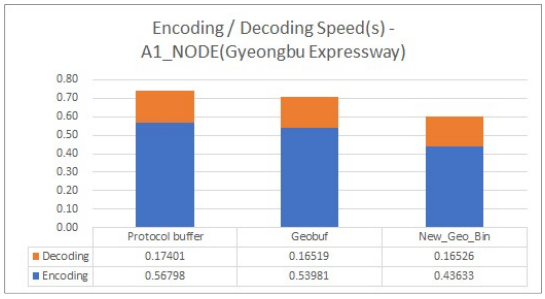

FIGURE 22. A2_LINK(linestring) encoding / decoding speed – Gyeongbu Expressway

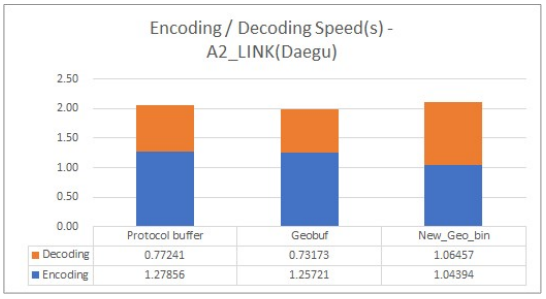

FIGURE 23. A2_LINK(linestring) encoding / decoding speed – Daegu

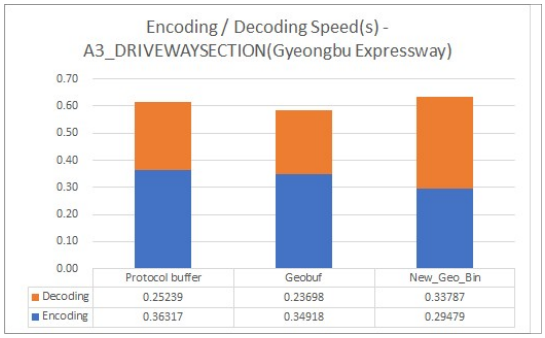

FIGURE 24. A3_Drivewaysection(polygon) encoding / decoding speed – Gyeongbu Expressway

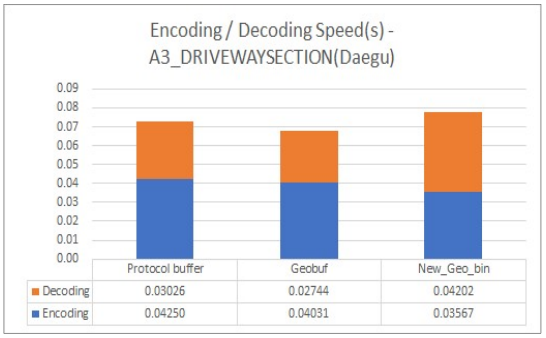

FIGURE 25. A3_Drivewaysection(polygon) encoding / decoding speed – Daegu

그림 23와 같이 인코딩 시간과 디코딩 시간을 모두 합친 경우, Geobuf의 속도가 가장 뛰어난 것으로 나타났다. 인코딩 속도의 경우, 두 정밀도로지도 데이터 모두 제안한 직렬화 기법이 가장 빠른 수치를 보였으나, 디코딩 속도의 경우 제안한 직렬화 기법이 Protocol buffer나 Geobuf 대비 느린 결과를 보였다. 이러한 결과는 가변형 정수로 인코딩되어 있는 좌표 값을 복원하는 과정에서 연산 속도가 느린 알고리즘으로 구현되어 시간이 소요된 것으로 보인다.

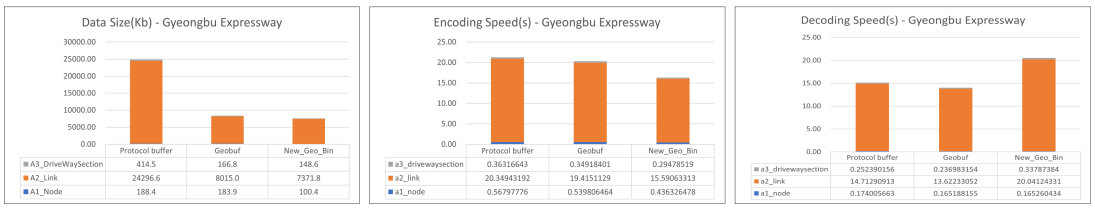

FIGURE 26. Data size and encoding / decoding speed – Gyeongbu Expressway

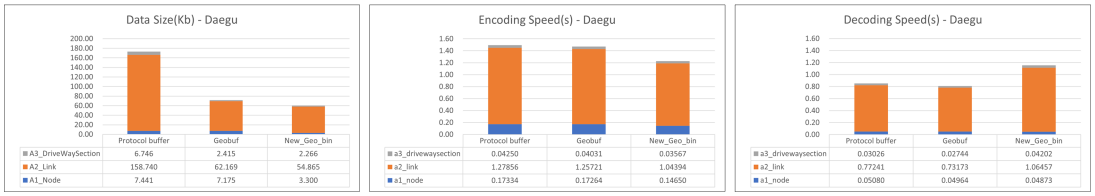

FIGURE 27. Data size and encoding / decoding speed – Daegu

이는 추후 최적화 작업을 통해 일부 개선할 수 있을 것으로 예상된다.

그림 24, 그림 25와 같이 면 유형의 공간 객체인 구간 데이터는 총 인코딩/디코딩 시간은 Geobuf가 가장 빨랐으며 Protocol buffer와 제안한 직렬화 기법은 비슷한 결과를 보인다. 주행경로링크 데이터의 결과와 유사하게 인코딩 속도는 제시한 직렬화 기법이 가장 빨랐으며 디코딩 시간은 가장 오래 걸렸다. 다른 형상 유형의 결과와 동일하게 데이터의 공간적 범위에 따른 성능 차이는 크게 보이지 않았다.

그림 26과 27과 같이 전체 공간데이터에 대한 성능을 비교한 결과, 제안한 직렬화 기법은 데이터의 공간적 범위에 따른 성능 차이는 크게 보이지 않았다. 단, 선 유형의 데이터 비중이 높아 성능별 특성이 선 유형을 따라가는 것을 확인할 수 있었다. 추후 형상 유형별 데이터 비중을 맞춰 추가적인 분석이 필요할 것으로 판단된다.

실험 분석 결과, 설계한 공간데이터 직렬화 기법은 형상 유형별 성능의 차이가 존재하였다. 점 유형의 공간데이터에 적용한 경우, 다른 직렬화 기법 대비 데이터 압축률 및 인코딩/디코딩 속도에서 모두 우수하였다. 선, 면 유형의 경우, 데이터 압축률, 인코딩 속도는 우수하나 디코딩 속도가 오래 걸린다는 단점이 존재하였다. 이는 최적화 작업을 통해 개선 가능할 것으로 보인다. 실험을 통해 제안한 공간데이터 직렬화 기법을 활용하여 대용량의 공간데이터를 경량화 할 수 있음을 확인할 수 있었다. 결과적으로 이를 활용하여 공간데이터의 효율적인 교환이 가능할 것으로 예상된다.

결 론

본 연구를 통해 정밀도로지도에 포함된 방대한 양의 3차원 공간데이터를 신속하게 전송 · 처리가 가능한 형태로 인코딩하고, 다양한 시스템에서 사용 가능한 직렬화 기법을 설계하였다. 이를 위해 첫째, 공간데이터 표현에 필수적인 항목을 선정하였으며, 둘째, 공간데이터에 최적화된 바이너리 기반 직렬화 구조를 정의하였다. 셋째, 공간데이터의 경량화 방안을 고안하여 직렬화 기법을 설계 및 구현하였으며, 넷째, 국토지리정보원에서 제공하는 정밀도로지도 데이터를 활용하여 공간 객체의 공간적 범위 및 형상 유형별 공간데이터 직렬화 기법의 성능을 평가하였다. 실험 결과를 통해 바이너리 기반 직렬화 기법을 사용하여 인코딩 시간과 공간데이터 경량화에 장점이 있다는 점을 확인하였다. 하지만 디코딩 시간의 경우, 점 유형에서는 장점이

있었으나 선과 면 유형 공간 데이터의 디코딩 시간은 더 길어진다는 단점이 있었다. 이는 추후 최적화 작업을 통해 개선해야할 점으로 보인다.

제안한 공간데이터 직렬화 기법은 기존의 직렬화 기법과는 달리 데이터 구조를 정의한 파일 없이 직렬화 및 역직렬화가 가능한 방식이다. 따라서 다양한 시스템 간의 데이터 교환 시 신속하고 원활한 교환이 가능할 것으로 보인다. 특히, 점 유형에서 가장 좋은 성능을 보이는 특성 상 점 유형의 차량 및 도로 시설물 정보를 교환할 때 매우 효율적일 것으로 예상된다.

하지만 본 연구에서는 데이터 전송 테스트가 진행되지 않아 실제 정밀도로지도 데이터 전송 시의 성능 향상 수준을 파악하기에는 한계가 있다. 추후 공간데이터 직렬화 기법을 적용한 교통 정보 메시지 구성 및 데이터 전송 실험을 통해 효용성을 더욱 입증할 수 있을 것으로 기대된다. 또한 속성 데이터 부분을 효율적으로 직렬화할 수 있는 기법들에 대한 연구도 수행되어야 할 것으로 보인다. KAGIS

REFERENCES

Apache Avro 1.10.0 Documentation. https://avro.apache.org/docs/1.10.0/. Apache

Baek C.Y., Park. S.H., and Kang Y.B. 2019. Korean HD Map Data Model Design for Autonomous Vehicles. Journal of Korean Society for Geospatial Information Science. 27(6):13-25 (백창엽, 박수홍, 강우빈. 2019. 국내 주행 환경에 적합한 자율주행용 정밀도로지도 데이터 모델 설계. 대한공간정보학회지, 27(6):13-25).

Bittl, S., Gonzalez, A. A., & Heidrich, W. 2014. Performance comparision of encoding schemes for ETSI ITS C2X communication systems. In Third International Conference on Advances in Vehicular Systems, Technologies and Applications. pp.58-63.

Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., & Yergeau, F. 2000. Extensible markup language (XML) 1.0.

Ecma International. 2017. The JSON Data Interchange Syntax (ECMA-404).

Eriksson, M., & Hallberg, V. 2011. Comparison between JSON and YAML for data serialization. The School of Computer Science and Engineering Royal Institute of Technology, 1-25.

International Oranization for Standardization. 2016. Intelligent transport systems -- Extension of map database specifications for applications of cooperative ITS (ISO 14296:2016).

International Oranization for Standardization. 2018. Intelligent transport systems -- Co-operative ITS -- Local dynamic map (ISO 18750:2018).

International Oranization for Standardization. 2004. Geographic information -- Simple feature access -- Part 1: Common architecture (ISO 19125-1:2004).

International Oranization for Standardization. 2020. Intelligent transport systems -- Geographic Data Files(GDF) -- GDF 5.1 -- Part 1: Application independent map data shared between multiple sources (ISO 20524-1:2020).

International Oranization for Standardization. 2019. Intelligent transport systems -- Dynamic data and map database specification for connected and automated driving system applications -- Part 1: Architecture and logical data model for harmonization of static map data (ISO/AWI TS 22726-1)

Jang, J., Jung, S. J., Jeong, S., Heo, J., Shin, H., Ham, T. J., & Lee, J.W. 2020.

A specialized architecture for object serialization with applications to big data analytics. In 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA) pp.322-334. IEEE.

Kang J.H., Kim H.D., and Kim J.T. 2018. Improving Compatibility Method of New Vworld 3D Data Using the Serialization Technique. Journal of the Korean Association of Geographic Information Studies, 21(1):96-105 (강지훈, 김현덕, 김정택. 2018. 데이터 직렬화 기법을 활용한 차세대 브이월드 3 차원 데이터의 호환성 개선 방안. 한국지리정보학회지, 21(1):96-105).

Krijnen, T., & Beetz, J. 2017. An IFC schema extension and binary serialization format to efficiently integrate point cloud data into building models. Advanced Engineering Informatics, 33:473-490.

Maeda, K. 2012. Performance evaluation of object serialization libraries in XML, JSON and binary formats. In 2012 Second International Conference on Digital Information and Communication Technology and it's Applications (DICTAP) pp.177-182. IEEE.

Navigation Data Standard PSF Physical Storage Format -- Open Lane Model Documentation version 1.0, Navigation Data Standard

National Geographic Information Institue. 2019. High Definition Map building manual (국토지리정보원. 2019. 정밀도로지도 구축 매뉴얼)

National Geographic Information Institue. 2018. (국토지리정보원. 2018. 정밀도로지도 연계 효율화 연구 및 구축갱신 연구보고서)

Proos, D. P., & Carlsson, N. 2020. Performance comparison of messaging protocols and serialization formats for digital twins in IoV. In 2020 IFIP Networking Conference (Networking) pp.10-18. IEEE.

Petersen, B., Bindner, H., You, S., & Poulsen, B. 2017. Smart grid serialization comparison: Comparision of serialization for distributed control in the context of the Internet of Things. In 2017 Computing Conference pp.1339-1346. IEEE.

Protocol Buffers. https://developers.google.com/protocol-buffers. google

SAE International. 2016. Dedicated Short Range Communications (DSRC) Message Set Dictionary J2375

"Schema evolution in Avro, Protocol Buffers and Thrift". 2012. Martin Kleppmann blog. Martin Kleppmann. https://martin.kleppmann.com/2012/12/05/schema-evolution-in-avro-protocol-buffers-thrift.html.

Yang I.C., Jeon W.H., and Lee H.M. 2017. Study on Dynamic Map Data Provision System for Automated Vehicle. The journal of the Korea Institute of intelligent transport systems. 16(6):208-218 (양인철, 전우훈, 이향미. 2017. 자율주행을 위한 동적지도정보 제공에 관한 연구. 한국 ITS 학회 논문지, 16(6):208-218).

Yang I.C., and Jeon W.H. 2018. Digital road event management system using high-precision map. The Journal of Contents Computing. 19(11):2219-2226 (양인철, 전우훈. 2018. 정밀전자지도를 활용한 디지털 도로 이벤트 관리 시스템. 한국디지털콘텐츠학회 논문지, 19(11):2219-2226).