GML 문서를 위한 변환 및 전송 시스템 개발†

황승연°, 강홍구, 김동오, 한기준
건국대학교 컴퓨터공학과
{syhwang°, hkkang, dokim, kjhan}@db.konkuk.ac.kr

Development of a Conversion and Transformation System for GML Documents

Seung-Youn Hwang°, Hong-Koo Kang, Dong-Oh Kim, Ki-Joon Han
Dept. of Computer Engineering, Konkuk University

요 약

OGC에서는 다양한 분야의 지리공간 정보를 손쉽게 상호 운용하기 위해 이질적인 환경의 지리공간 정보를 XML로 인코딩하는 GML 명세를 제시하였다. 그러나 GML 문서는 텍스트 구조로 되어있어 데이타 용량이 커지며 전송 시 속도가 현저히 저하되고, 문서 스캔 비용이 많이 든다는 문제가 있다. 그러므로, 대용량의 GML 문서를 정보 손실 없이 변환하여 전송 속도와 문서 스캔 속도를 향상시키기 위한 방법이 필요하다. 이에 OGC에서는 최근 바이너리 XML 형식인 BXML을 제안하였다.

본 논문에서는 BXML 형식을 사용하여 GML 문서를 토큰화하고, 토큰을 정의된 바이트 코드로 변환하여 문서의 크기를 줄이고 전송 속도와 문서 스캔 속도를 향상시키는 효율적인 GML 문서 변환 및 전송 시스템을 설계하고 구현하였다. GML 문서 변환 및 전송 시스템은 GML 문서와 BXML 문서를 상호 변환하는 기능과 BXML 문서에서 공간 데이타를 추출하여 디스플레이하는 기능을 제공한다. 성능 평가 결과 GML 문서 변환 및 전송 시스템 사용 시 GML 문서의 크기는 최대 80% 감소하였고 전송 속도는 최대 3.5배 향상되었다.

1. 서 론

최근 지리공간 정보에 대한 관심이 증가함에 따라 지리공간 정보의 활용 분야가 점차 다양해지고, 이를 효율적으로 관리하기 위해 다양한 지리 정보 시스템(GIS : Geographic Information System)이 구축되었다. 이렇게 다양한 지리 정보 시스템에 독자적으로 구축된 대량의 지리공간 정보들을 다른 응용 분야의 기초 데이타로 활용하거나 다른 지리 정보 시스템과 지리공간 정보를 공유할 필요성이 발생하게 되었다[1,6,12,13].

이에 OGC(Open GIS Consortium)에서는 다양한 분야의 지리정보를 효율적으로 유통 및 활용하기 위해 이질적인 환경의 지리 공간 정보를 XML[7]로 인코딩하는 GML(Geography Markup Language) 명세[3,4]를 제시하였다. 그러나, GML 문서는 텍스트를 이용해 인코딩 되어있으며 테그가 상당히 반복적이어서 데이타 용량이 커지고 전송 속도가 느리며, 문서의 스캔 비용이 많이 든다는 문제를 가지고 있다[2,5,14].

그러므로, 텍스트로 구성된 대용량의 GML 문서를 정보 손실 없이 변환하여 전송 속도와 문서 스캔 속도를 향상시키기 위한 방법이 필요하다. 이에 OGC에서는 최근 GML 문서를 바이너리 XML 형식으로 인코딩하는 BXML(Binary-XML) 인코딩 명세[2]을 제안하였다.

본 논문에서는 GML 문서의 크기를 줄여 전송 속도와 문서 스캔 속도를 향상시키기 위해 OGC의 BXML 인코딩 명세를 이용해 GML 문서와 BXML 문서를 상호 변환하고, BXML 문서에서 공간 데이타를 추출하여 디스플레이하는 기능을 제공하는 효율적인 GML 문서 변환 및 전송 시스템을 설계 및 구현하였다.

본 논문의 구성은 다음과 같다. 2장의 관련 연구에서는 GML 3.0 명세와 BXML 인코딩 명세를 분석한다. 3장과 4장에서는 효율적인 GML 문서 변환 및 전송 시스템의 설계와 구현에 대해서 각각 설명한다. 5장에서는 다양한 GML 문서를 이용한 GML 문서 변환 및 전송 시스템의 성능 평가 결과를 언급한다. 마지막으로, 6장에서는 결론 및 향후 연구에 대해 기술한다.

2. 관련연구

본 장에서는 OGC의 GML 3.0 명세와 BXML 인코딩 명세에 대하여 상세히 기술한다.

2.1 GML 3.0

OGC에서는 다양한 분야의 지리공간 정보를 손쉽게 상호 운용하기 위해 이질적인 환경의 지리공간 정보를

† 본 논문은 한국산업기술재단의 대학보유기술이전사업의 연구비 지원에 의해 수행되었음.

XML[7]로 인코딩하여 다양한 분야에 사용 가능하도록 하는 GML 명세를 제시하였다. GML 명세는 1999년에 GML 1.0 발표이후, 2001년에 XML 스키마[8,9,10]에 기반을 둔 GML 2.0[3]을 발표하였고, 2003년 GML 3.0[4]을 발표하였다.

GML 명세는 GML의 사용 목적과 장점, GML에서 사용하는 객체(Object) 모델의 정의, GML로 엔코딩하는 방법, GML에서 제공하는 기본 스키마, 응용 스키마(Application Schema)를 제작하기 위해 지켜야 할 사항 등을 포함하고 있다. 그림 1은 GML 3.0에서 제공하는 다양한 GML 객체의 구조를 보여준다.

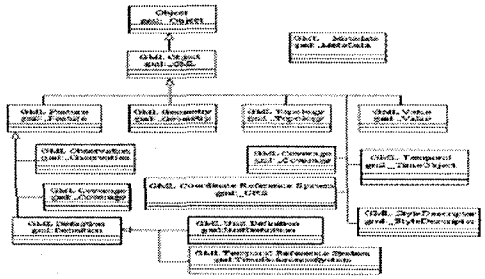

그림 1. GML 3.0 클래스 구조

2.1.1 GML 3.0 기본 스키마

GML 3.0에서는 지리공간 정보를 표현하기 위해 총 21개의 기본 스키마를 정의하고 있다. 표 1은 이러한 GML 3.0 기본 스키마를 보여준다.

표 1. GML 3.0 기본 스키마

gmlBase 스키마	GML에서 사용할 추상 데이터 타입들을 정의하고 있는 스키마
feature 스키마	Feature나 feature collection을 표현하기 위한 스키마
geometry 스키마	공간 데이터를 표현하기 위한 스키마
units 스키마	uom(unit of measure)을 인코딩하는 방법을 기술한 스키마
basicType 스키마	GML에서 사용하기 위한 simpleContent 타입들을 지원하기 위한 스키마
measures 스키마	Units 스키마와 baseType 스키마의 확장으로 특정 데이터 타입의 측정을 위한 단위를 나타내는 스키마
referenceSystems 스키마	참조 시스템의 추상 데이타 타입을 지원하는 스키마
datums 스키마	CRS(Coordinate Reference Systems)의 Datums 패키지를 인코딩하기 위한 스키마
coordinateSystems 스키마	CRS의 좌표 시스템 패키지를 인코딩하기 위한 스키마
coordinateReferenceSystems 스키마	CRS의 좌표 참조 시스템 패키지를 인코딩하기 위한 스키마
operations 스키마	한 좌표에서 다른 좌표 시스템으로의 좌표 변환을 위한 수식을 정의하고 있는 스키마
accuracy 스키마	데이타의 정밀도에 관해 정의하는 스키마
temporal 스키마	움직이는 객체의 상태나 움직인 Feature를 표현하기 위한 스키마
observation 스키마	특정 Feature의 시간에 따른 속성 변화를 관찰하기 위한 스키마
direction 스키마	방향을 나타내기 위한 데이터 타입이나 엘리먼트를 정의하는 스키마
dynamicFeature 스키마	움직이는 객체의 추적을 지원하기 위해 필요한 스키마
coverage 스키마	특정 영역에 관련된 값들의 집합을 표현하는 스키마
topology 스키마	위상 관계를 표현을 위한 스키마
grid 스키마	Coverage 스키마에서 사용할 지리 객체를 정의하는 스키마
value 스키마	관찰 값을 정의하는 스키마
defaultstlye 스키마	Feature를 위한 스타일을 나타내는 스키마

2.1.2 GML 3.0 응용 스키마

사용자가 표현하려는 다양한 지리공간 정보의 데이타 구조를 GML 명세에서 제공하는 기본 스키마로는 표현할 수 없으므로, 이러한 기본 스키마를 기반으로 실제 GML 3.0 문서에서 참조할 GML 3.0 응용 스키마를 제작하여 사용하여야 한다.

GML 3.0 명세에서 제시하는 GML 3.0 응용 스키마 생성 규칙으로는 Features나 Feature Collections를 정의하는 방법, Coverages를 정의하는 방법, Observations를 정의하는 방법, Dictionaries나 Definitions를 정의하는 방법, Coordinate Reference System을 정의하는 방법, Values를 정의하는 방법 등이 있다.

2.2 BXML

BXML은 제한된 환경에서 XML 문서에 대해 의미 손실 없이 크기를 줄여, 보다 효과적인 전송을 위해 제시 되었다[11]. 특히, OGC에서 2003년에 BXML 인코딩 명세 0.0.8[2]을 발표하였다.

BXML 화일은 헤더와 바디로 구성된다. 헤더는 화일이 BXML 화일임을 나타내는 식별자, 화일을 읽기 위해 필요한 버전, 압축 유무, 인코딩에 사용된 문자 셋 정보가 저장되어있다. 바디는 토큰값의 연속으로 구성되며, 토큰은 토큰의 타입에 따라 이미 정의되어 있는 1바이트의 16진수 코드 값과 엘리먼트의 이름 참조 값 또는 데이타 값으로 표현된다. 바디의 마지막 토큰은 정의된 TrailerToken이어야만 한다. 표 2는 각각의 토큰 타입들에 따른 코드값을 보여준다.

표 2. 토큰 타입 코드 값

토큰 타입	값	설명
EmptyElementCode	0x00	<element/>
EmptyAttrElementCode	0x01	<element..../>
ContentElementCode	0x02	<element>....
ContentAttrElementCode	0x03	<element.....>....
ElementEndCode	0x04	</element>
AttributeStrartCode	0x05	attr="
AttributeListEndCode	0x06	end of attributes
CharContentCode	0x10	character content
CharContentRefCode	0x11	string-table char content
CDataSectionCode	0x12	<![CDATA[content]]>
WhitespaceCode	0x13	whitespace character content
BlobSectionCode	0x14	raw-binary data
EntityRefCode	0x15	&entity_ref
CharEntityRefCode	0x16	&# char_ref
CommentCode	0x17	<!- comment→
XmlDeclarationCode	0x20	<?xml .....?>
BangCode	0x21	<!name..... >
BangBracketCode	0x22	<![name[.....]]>
ProcessingInstrCode	0x23	<?name.....?>
StringTableCode	0x30	String table
IndexTableCode	0x31	Index table
TrailerCode	0x32	trailer

엘리먼트는 EmptyElementToekn, EmptyAttrElement Token, ContentElementToken, ContentAttrElement Token 중 하나로 시작하며, 각각은 토큰 타입 코드와 엘리먼트 이름에 해당하는 문자열 테이블 참조 값을 갖는다. 어트리뷰트를 갖는 EmptyAttrElementToken과 ContentAttrElementToken 뒤에는 하나 이상의 AttributeStartToken이 오며 AttributeListEndToken으로 끝난다. 데이타를 포함한 ContentElementToken과 ContentAttrElementToken은 ElementEndToken으로

끝난다. ElementEndToken은 엘리먼트의 이름값을 가지지 않는다.

3. 시스템 설계

본 장에서는 효율적인 GML 문서 변환 및 전송 시스템의 전체 모듈 구성을 살펴보고, 각 모듈에 대해 상세히 설명한다.

3.1 전체 시스템 모듈 구성

그림 2는 효율적인 GML 문서 변환 및 전송 시스템의 전체 모듈 구성을 보여준다.

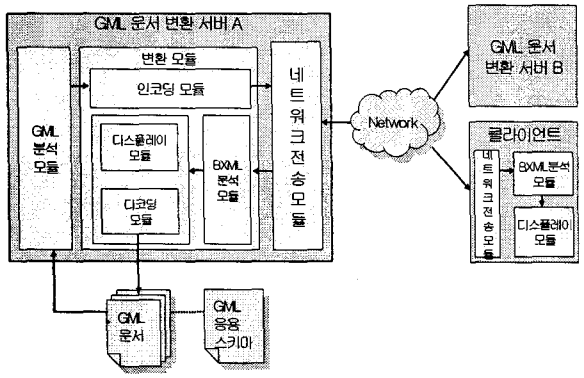

그림 2. 전체 시스템 모듈 구성

본 시스템은 크게 서버 컴포넌트와 클라이언트 컴포넌트로 구분된다. 서버 시스템은 GML 문서를 분석하는 GML 문서 분석 모듈, GML 문서를 바이너리로 인코딩하는 인코딩 모듈, BXML 문서를 분석하는 BXML 분석 모듈, BXML 문서를 GML 문서로 디코딩하는 디코딩 모듈, BXML 문서로부터 공간 데이타를 읽어 디스플레이 하는 디스플레이 모듈, 네트워크를 통해 데이타를 전송하는 네트워크 전송 모듈로 구성된다. 클라이언트 컴포넌트는 네트워크 전송 모듈, BXML 문서 분석 모듈, 디스플레이 모듈로 구성된다.

3.1.1 GML 분석 모듈

GML 분석 모듈은 GML 문서를 분석하기 위한 모듈이다. GML 분석 모듈에서는 GML 3.0 문서가 참조하는 GML 3.0 응용 스키마를 따르는지를 검사한다. 그리고 GML 문서를 엘리먼트, 데이타, 어트리뷰트, 주석 등의 토큰으로 구분한다. 엘리먼트는 어트리뷰트를 포함한 타입인지, 데이타를 포함한 타입인지에 따라 코드 값이 달라지므로 구분하여 토큰 타입 정보와 필요한 데이타를 인코딩 모듈에 전달해 주고, 엘리먼트 이름을 문자열 테이블로 작성한다.

3.1.2 인코딩 모듈

인코딩 모듈은 GML 문서를 BXML로 인코딩하기 위한 모듈이다. 인코딩 모듈은 문서 식별자, 화일 버전, 압축 유무, 인코딩에 사용된 문자 셋 정보를 가진 BXML 문서 헤더를 인코딩하고, 토큰 타입에 따라 엘리먼트, 어트리뷰트, 데이타들을 바이너리로 인코딩한다. 엘리먼트는 타입에 따라 미리 정의된 타입 코드 값과 엘리먼트 이름을 문자열 테이블의 참조 값으로 인코딩한다. 어트리뷰트는 미리 정의된 어트리뷰트 시작 코드 값과 어트리뷰트 값으로 인코딩되며, 어트리뷰트 값은 문자열 그대로 인코딩된다. 어트리뷰트 값 중에 사용 빈도가 높은 몇몇 문자열의 경우 엘리먼트 이름처럼 문자열 테이블에 추가되어 참조 값으로 인코딩될 수도 있다. 데이타, 주석 등도 각각 해당하는 타입 코드 값과 문자열 값으로 인코딩된다.

3.1.3 BXML 분석 모듈

BXML 분석 모듈은 BXML 문서를 분석하기 위한 모듈이다. BXML 분석 모듈은 BXML 문서를 순차적으로 검색하여 미리 정의된 코드 값이 인식되면 이 코드 값이 엘리먼트, 데이타, 어트리뷰트 또는 주석인지를 구분한다. 엘리먼트 코드 값이 인식되면 엘리먼트 타입 코드 값과 엘리먼트 이름 참조 값을 디코딩 모듈에 전달해 주고, 데이타나 어트리뷰트 코드 값이 인식되면 각각의 문자열 값을 디코딩 모듈에 전달해 준다. 디스플레이를 할 때는 공간 데이타를 포함하는 엘리먼트 코드를 찾아 공간 데이타를 추출하여 디스플레이 모듈에 전달해 준다.

3.1.4 디코딩 모듈

디코딩 모듈은 BXML 문서를 GML 문서로 변환하기 위한 모듈이다. 디코딩 모듈은 BXML 분석 모듈에서 받은 엘리먼트 이름 참조 값으로 문자열 테이블을 참조하여 엘리먼트를 생성하고, 엘리먼트의 타입에 따라 데이타와 어트리뷰트를 삽입하여 GML 문서로 변환한다.

3.1.5 디스플레이 모듈

디스플레이 모듈은 BXML 문서에서 추출된 공간 데이타를 디스플레이해 주는 모듈이다. 디스플레이 모듈에서는 공간 데이타를 화면에 디스플레이 할 수 있도록 실제 좌표를 화면 좌표로 변환하는 기능, 디스플레이 된 데이타를 축소/확대 하는 줌(ZOOM) 기능, 영역(동, 서, 남, 북)을 이동하는 팬(PAN) 기능을 제공한다.

3.1.6 네트워크 전송 모듈

네트워크 전송 모듈은 BXML로 인코딩된 데이타를 네트워크를 통해 전송해 주는 모듈이다. 네트워크 전송 모듈은 클라이언트에게 GML 문서 목록를 보내주는 기능과 데이타를 압축 전송하는 기능을 제공한다.

4. 시스템 구현

본 장에서는 GML 문서 변환 및 전송 시스템을 이용하여 GML 문서를 BXML 문서로 변환하여 전송하고, BXML 문서를 이용하여 지리정보 데이타를 디스플레이 하거나 GML 문서로 디코딩하는 과정을 설명한다. 그림 3은 문서 변환 및 전송 시나리오를 보여준다.

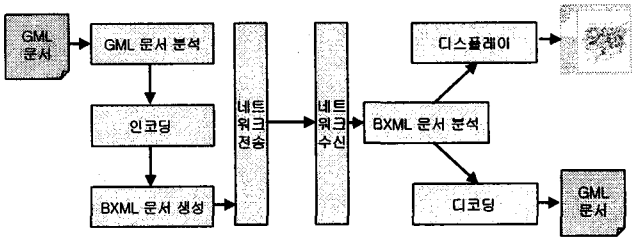

그림 3. 문서 변환 및 전송 시나리오

4.1 GML 문서 분석

본 시스템에서 GML 문서를 BXML 문서로 인코딩하기 위해 GML 문서를 분석하여야 한다. GML 문서 분석은 DOM 파서를 이용하여 GML 문서를 엘리먼트, 데이타, 어트리뷰트, 주석 등으로 토큰화한다. 엘리먼트는 다시 속성을 갖고 있는지, 데이타를 갖고 있는지 여부에 따라 엘리먼트의 형태를 구분하고, 타입 정보, 어트리뷰트 값, 데이타 값을 인코딩 모듈에 전달해 준다. 그리고 엘리먼트 이름을 문자열 테이블로 작성한다.

4.2 BXML 문서 생성

GML 문서 분석 모듈에서 GML 문서를 토큰화하여 넘겨준 토큰 타입 정보를 이용해서 엘리먼트를 찾는다. 그리고, 엘리먼트의 타입에 따라 미리 정의된 토큰 코드 값들과 문자열 테이블의 엘리먼트 이름 참조 값을 기반으로 해당하는 바이너리 값으로 인코딩한다. 또한 데이타, 어트리뷰트, 주석 등은 각각 해당하는 토큰 코드 값과 문자열로 인코딩한다. 그림 4는 GML 문서의 모습이고, 그림 5는 그림 4의 GML 문서를 BXML 문서로 변환한 모습이다.

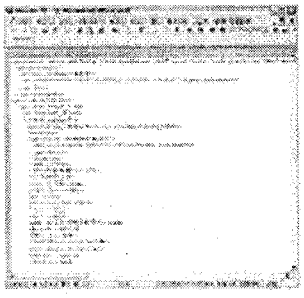

그림 4. GML 문서

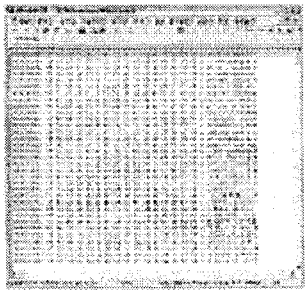

그림 5. BXML 문서

4.3 BXML 문서 디스플레이

디스플레이 모듈은 BXML 문서로부터 공간 데이타를 추출하여 화면에 지도 형식으로 디스플레이해 준다. 또한 디스플레이 결과를 확대나 축소할 수 있는 줌 기능과 동, 서, 남, 북으로 영역을 이동하는 팬 기능을 제공한다.

공간 좌표 데이타를 화면에 출력하기 위해서는 디스플레이 화면의 영역을 설정하고, 실제 좌표를 화면상에 디스플레이 가능한 수치로 변환해야 한다. 그림 6은 BXML 문서로부터 공간 데이타를 읽어 디스플레이한 모습이다.

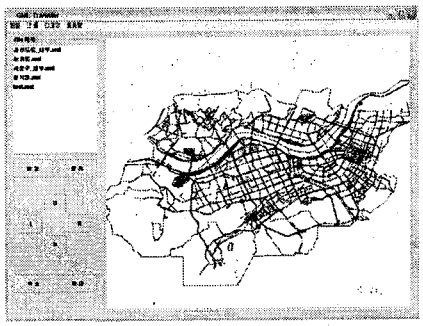

그림 6. BXML 문서를 디스플레이한 모습

그림 6의 디스플레이 결과에 이전의 영역 좌표 값보다 축소된 영역 좌표 값을 적용하면 디스플레이 시 확대되고, 일정한 화면 크기에 이전의 영역 좌표 값보다 확대된 영역 좌표 값을 적용하면 디스플레이 시 축소된다. 그림 7은 그림 6의 BXML 문서를 축소/확대한 모습을 보여주고 있다.

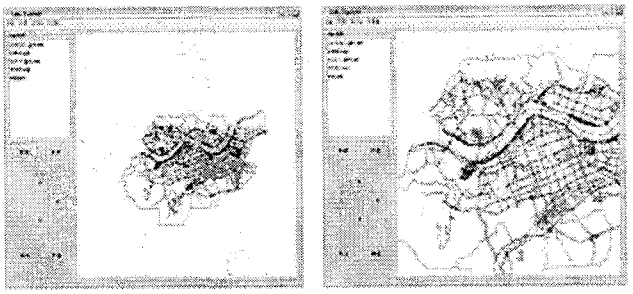

그림 7. 그림 6을 축소/확대한 모습

디스플레이 결과를 동, 서, 남, 북으로 영역을 이동하는 기능을 제공하기 위해 먼저 중심 좌표 값을 구해야 한다. 중심 좌표 값을 이용하여 디스플레이할 새로운 최소 좌표 값과 최대 좌표 값을 얻을 수 있다. 이 최대, 최소 좌표 값에서 이동할 x 좌표 값만큼 더하면 동쪽으로 이동, 이동할 x 좌표 값만큼 빼면 서쪽으로 이동, 이동할 y 좌표 값만큼 더하면 남쪽으로 이동, 이동할 y 좌표 값만큼 빼면 북쪽으로 이동한 결과를 디스플레이할 수 있다. 그림 8은 그림 6의 BXML 문서를 이동한 모습을 보여주

고 있다.

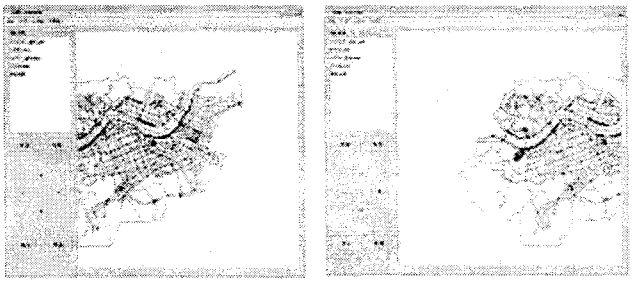

그림 8. 그림 6을 이동한 모습

4.4 BXML 문서 변환

BXML 문서 변환은 BXML 문서를 원래의 GML 문서로 정보 손실 없이 변환하도록 구현되었다. 먼저 BXML 문서를 분석하여 토큰 타입 코드를 구분한다. 엘리먼트를 나타내는 코드 값이 인식되면 문자열 테이블의 참조 값을 나타내는 다음 1바이트 값을 읽어 타입 코드 값과 함께 디코딩 모듈에 전달하여 준다. 이때 어트리뷰트 및 데이타가 포함된 경우 어트리뷰트 값과 데이타 값을 추출하여 디코딩 모듈에 전달하여 준다.

디코딩 모듈에서는 토큰 타입 코드와 문자열 테이블을 참조하여 엘리먼트를 생성하고, 속성과 데이타를 삽입하여 GML 문서로 변환을 한다.

5. 성능 평가

GML 변환 및 전송 시스템은 Windows XP 환경에서 Java로 구현되었고, 사용된 컴퓨터 시스템은 Intel Pentium4 2.6GHz 이고 메모리는 512MB 이다.

그림 9는 GML 문서와 변환된 BXML 문서의 크기를 보여주는 그림이며, 사용된 GML 문서는 point 데이타로 구성된 GML 문서, polygon 데이타로 구성된 GML 문서, point와 polygon 데이타로 구성된 GML 문서이다.

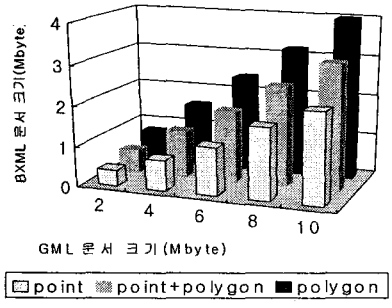

그림 9. GML 문서 크기 감소 결과

그림 9에서 보듯이 같은 크기의 GML 문서 중에서 point 데이타로 구성된 GML 문서의 크기 감소 효과가 가장 큰 것을 알 수 있다. 이는 태그나 속성을 나타내는 문자열은 1바이트로 변환되는 반면 좌표 데이타와 같은 일반 데이타 문자열은 변환 없이 그대로 인코딩되기 때문에 문서 내에서 일반 문자열 데이타의 비율이 높아지면 상대적으로 문서 크기 감소 효과가 감소하게 되는 것이다.

그림 10은 클라이언트의 요청에서부터 클라이언트의 화면에 데이타를 디스플레이 해 주는데까지 걸리는 시간을 보여주는 그림이며, 각각 GML 문서를 전송하여 디스플레이 했을 경우, GML 문서를 GZIP으로 압축 후 전송하여 디스플레이 했을 경우, BXML 문서로 변환하여 전송하여 디스플레이 했을 경우, BXML 문서를 GZIP으로 압축 후 전송하여 디스플레이 했을 경우를 측정한 결과이다.

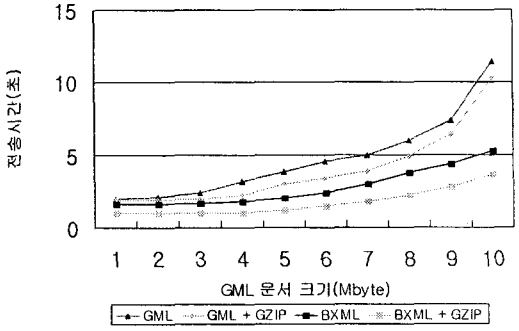

그림 10. 문서 요청 결과

그림 10에서 보듯이 GML 보다 BXML을 이용하는 것이 빠르며 화일 크기가 커질수록 그 차이가 점점 커지는 것을 볼 수 있다. 이는 BXML은 공간 데이타를 추출하기 위해 1바이트 코드만 비교하지만 GML은 문자열을 비교해야 하기 때문이다. 따라서 BXML을 이용하는 것이 더 빠르며 화일의 크기가 커질수록 차이가 더 커진다. 그리고, GML 문서를 GZIP로 압축하여 전송할 경우 전송 속도는 향상되지만, 문서 스캔 속도가 떨어져 디스플레이 하는 시간이 길어진다.

6. 결론 및 향후 연구

본 논문에서는 정보 손실 없이 GML 문서와 BXML 문서를 상호 변환하고, BXML 문서에서 데이타를 추출하여 디스플레이를 할 수 있는 효율적인 GML 문서 변환 및 전송 시스템을 설계 및 구현하였다.

본 논문에서 제시한 GML 문서 변환 및 전송 시스템은 GML 문서를 바이너리 형태인 BXML로 인코딩 함으로써 GML 문서의 크기를 줄여 전송 속도를 향상시킬 수 있으며, 문서의 스캔 속도를 향상시켜 빠르게 지리 정보 데이타를 디스플레이 할 수 있다.

성능평가 결과 GML 문서를 BXML 문서로 변환했을 때 문서의 크기는 최대 80% 감소하는 효과를 보였고, GML 문서를 전송하고 공간 데이타를 디스플레이하는 시간은 최대 3.5배 빨라지는 것을 볼 수 있었다. 그러나 GML 문서의 크기 감소율은 문서에 있는 좌표 데이타의 양에 따라 다소 차이를 보이고 있다. 따라서 앞으로 좌표 데이타의 양이 많은 GML 문서의 크기 감소효과를 높이기 위한 인코딩 기법에 관한 연구가 필요하겠다.

참고 문헌

[1] ISO/TC 211, *ISO 19136: Geographic Information - Geography Markup Language*, N 1276, 2002.

[2] OpenGIS Consortium, Inc., *Binary-XML Encoding Specification 0.0.8*, 2003.

[3] OpenGIS Consortium, Inc., *Geography Markup Language(GML) Implementation Specification 2.0*, 2001.

[4] OpenGIS Consortium, Inc., *Geography Markup Language(GML) Implementation Specification 3.0*, 2003.

[5] Piras, A., Demontis, R., Vita, E. D., Sanna, S., "Compact GML : merging mobile computing and mobile cartography," GML And Geo-Spatial Web Service Conference, 2004.

[6] Tom, H., "GIS Standardization : The American Experience," 개방형 GIS 연구회 논문지, 1권 1호, 1999, pp.99-180.

[7] W3Consortium, *Extensible Markup Language (XML) 1.0*, 1998.

[8] W3Consortium, *XML Schema Part 0: Primer*, 2001.

[9] W3Consortium, *XML Schema Part 1: Structures*, 2001.

[10] W3Consortium, *XML Schema Part 2: Data types*, 2001.

[11] Wireless Application Protocol Forum, Ltd., *Binary XML Content Format Specification 1.3*, 2001.

[12] 손훈수, 김동오, 한기준, "GML 기반 지리공반 정보 엔크딩 서비스 시스템의 설계 및 구현," 개방형지리정보시스템학회 2003년 추계 학술대회, pp.91-97.

[13] 오병우, 한기준, "지리공간 정보 시스템을 위한 표준화," 한국정보과학회 정보과학회지, 13권 10호, 1995, pp.46-55.

[14] 최은정, 한동원, 임경식, "무선 인터넷 서비스를 위한 WAP 게이트웨이용 WML 컴파일러의 설계 및 구현," 한국정보과학회 정보과학회지, 7권 2호, 2001, pp.165-180.