CORBA 서비스의 성능 향상을 위한 인터페이스 설계

김 상 호† · 지 정 희† · 류 근 호††

요 약

지리정보시스템에 대한 연구는 개방형 구조, 상호운영성과 확장 가능성에 초점을 두고 발전 되어 왔다. 이 연구들은 기존에 구축된 하나의 통합시스템의 형태에서 대용량의 데이터, 고차원 구조의 데이터 그리고, 서로 다른 지리 포멧의 데이터를 처리하기 위하여, 응용 프로그램과 사용자 인터페이스를 레이어 별로 구분하여 각기 관리하는 개방형 구조로 시스템이 구현되어야 함을 의미한다. 현재 구현되는 개방형 지리정보시스템을 위한 미들웨어들 중 CORBA 미들웨어는 특정 운영체제를 기반으로 하지 않아 많은 개발자들이 이용하고 있다. 이때, CORBA의 서비스는 점대점 통신만을 지원하고, 다른 미들웨어보다 데이터 제공 시간이 많이 소요되는 문제점을 가지고 있다. 따라서 이 논문에서는 점대점 통신기법의 수정 방법을 제안하고, 구현한다. 제시된 방법은 클라이언트가 서버의 데이터를 한번의 연결로 데이터를 제공함으로써 그룹 통신과 빠른 데이터 제공 속도를 보장할 수 있게 한다.

Interface Design for Performance Improvment of CORBA Services

Sang Ho Kim† · Jeong Hee Chi† · Keun Ho Ryu††

ABSTRACT

A new research of geographic information system has been focused on an open architecture, interoperability and extensibility in their design. In order to process huge data, multidimensional data and heterogeneous geographic format data, these research move away from monolithic system to open structure system managing application layer and user interface layer. Nowadays, many developers have used CORBA middleware for the OGIS systems. However, CORBA middleware has some problems that support only a point-to-point communication and takes a long time to transfer sever data to client. Thus, in this paper, we propose a method on modifying a point-to-point communication and implement the efficient communication method. The proposed method that transfer data from sever to client in one connection support a group communication and reduce a transfer time delay.

키워드 : CORBA, 개방형 지리정보시스템(OpengGIS)

1. 서 론

지리정보시스템은 기하와 위상정보를 다루는 공간 데이터뿐만 아니라 비공간 데이터에 대한 입력, 저장, 질의, 분석 및 시각화 작업을 수행한다. 정보화 사회의 급속한 발전에 의해 지리정보의 사용이 많아지고, 이러한 지리정보 자원의 생성, 저장, 관리 및 응용을 위한 지리정보시스템의 구축과 응용 개발이 보편화되고 있다. 지리정보시스템은 특정한 목적을 가지고 제작된 지리 정보를 다양하게 분석하기 위해 이용할 수 있으며, 또한 지리 정보를 다른 정보와 결합시켜 분석함으로써 원하는 정보를 추출할 수 있다[1, 2]. 전통적 지리정보시스템에서는, 이러한 모든 기능들을 동일한 시스템 내에서 처리하였다. 그러나 최근의 개발되는 지리정보시스템들은 많은 양의 지리 데이터, 복잡하고 고차원 구조의 데이터, 그리고, 서로 다른 지리 정보 포멧을 처리 하기 위하여, 기능별로 분리된 상호운용을 지원하는 개방형 지리정보시스템 형태로 구현되고 있다[3]. 개방형 지리정보시스템에서 지원하는 상호운용의 개념은 표준화된 지리 데이터의 접근, 교환, 분산 지리정보처리를 수행함으로써 데이터의 공유뿐만 아니라 표준 인터페이스를 통한 지리정보 처리 서비스의 공유까지 가능하게 해주는 것이다. 상호운용을 지원하기 위해서는 서로 다른 데이터 소스들의 지리 데이터 표현에 대한 데이터 접근 모델을 외부 응용 프로그램이 접근할 수 있도록 표준화된 데이터 접근 모델로 변환해야 한다[4, 5].

이러한 서로 다른 데이터 소스들의 접근 모델을 제시하고, 서비스를 제공하기 위하여, OGC(Open GIS Consortium)에서는 상호운용성을 제공하는 개방형 지리정보시스템 서비스 명세서를 제안하고 있다[4, 6, 7]. 이 명세서에서는 개방형 지리정보시스템에서 모든 분산 컴퓨팅 환경에 독립적인 추상 명세와 함께 각 정보기술별 구현 명세를 제시하고 있다.

개방형 지리정보시스템 추상 명세 중 CORBA를 위한 추상 명세는 특정 운영체제에 기반하지 않는 서비스 제공과 객체지향적 모델링을 바탕으로 한 서비스 제공으로 타 추상 명

※ 이 연구는 2003년도 건교부 국가 GIS사업(국토연구원) 및 한국과학재단 RRC(청주대 정보통신 연구센터)의 연구비지원으로 수행되었음.
† 준 회 원 : 충북대학교 대학원 전자계산학과
†† 종신회원 : 충북대학교 전기전자 컴퓨터공학부 교수
논문접수 : 2003년 7월 4일, 심사완료 : 2003년 10월 13일

세보다 구현에 많이 이용되고 있다. 그러나, 개방형 지리정보시스템에서 제시하는 CORBA 서비스는 언어의 특성상 데이터 전송 시 다른 미들웨어보다 전송 시간을 많이 요구하는 문제점[8, 9]을 가지고 있다. 특히, 기존 지리정보시스템에서는 갱신, 삭제 등의 작업보다 검색연산이 주로 발생하고, 전체 데이터를 화면에 보여주고, 원하는 데이터를 검색하는 윈도우 질의가 많이 이루어지기 때문에 CORBA 데이터 제공 속도는 개방형 지리정보시스템 성능에 많은 영향을 준다.

따라서, 이 논문에서는 CORBA의 데이터 제공속도 향상을 위한 새로운 인터페이스를 제안하고, 구현하여 CORBA를 이용한 개방형 지리정보시스템 구축 시 인터페이스 성능을 향상시킬 수 있게 한다. 이를 위해, 기존의 한번에 하나씩의 전송만을 수행하는 CORBA의 Iterator 인터페이스 중 Feature-Iterator에 수정된 get_WKSGeometries 인터페이스를 제안하고, 제안된 인터페이스를 통해서 한번 접속 시 데이터베이스 내의 모든 데이터들을 클라이언트에 넘겨주도록 한다.

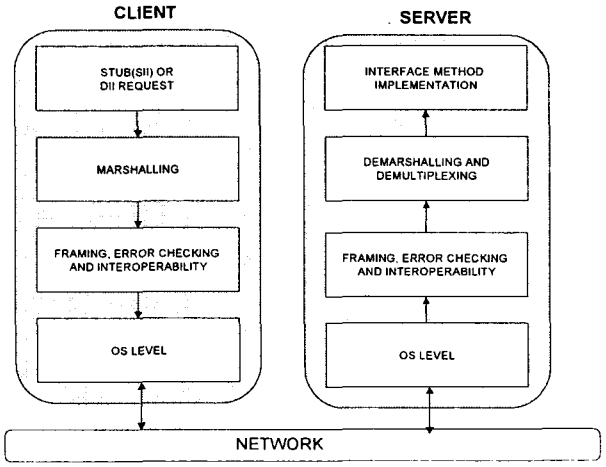
이 논문의 효과적인 전개를 위하여, 2장에서는 관련 연구로 개방형 지리정보시스템 관련 기술의 변화와 CORBA와 다른 미들웨어들 간의 성능에 대한 기존 연구들을 살펴본다. 3장에서는 OGC의 CORBA 구현명세와 표준 인터페이스에 대해 기술한다. 4장에서는 개선된 인터페이스를 제안한다. 5장에서 제안된 인터페이스를 적용한 시스템을 구현하고, 두 인터페이스간의 비교를 위한 실험을 수행하고, 결과 분석 및 기존 방식과 제안된 방식간의 성능 평가를 수행한다. 마지막 6장에서 결론을 맺는다.

2. 관련 연구

개방형 지리정보시스템은 다양한 형태의 데이터 제공, 데이터 처리 분석 등을 클라이언트서버 모델을 기반으로 제공하게 해준다.

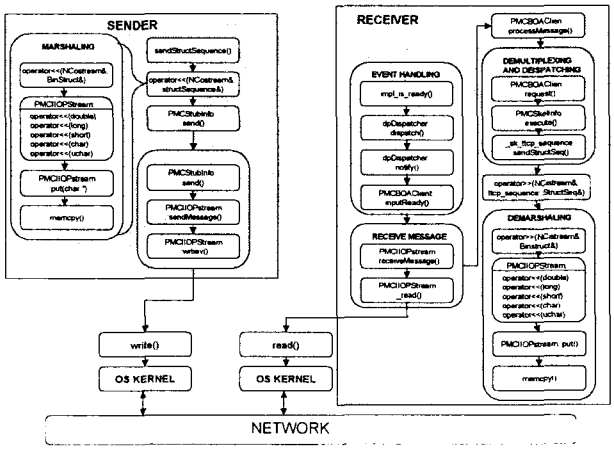
클라이언트서버 모델은 응용 프로그램 개발자로 하여금 요구사항들과 어플리케이션 기능들을 분리하고 다양한 수준에서의 요구사항들을 충족시키는 컴포넌트의 역할과 관계를 파악할 수 있게 한다. 일반적으로 하위 차원의 서비스는 표준 인터페이스를 통해 하위의 서비스 제공자 레이어로 이동하며, 이 레이어는 좀 더 간편하고 환경 적응력이 뛰어나고 매우 특화된 기능을 위한 플랫폼이 된다[4]. 지금까지의 지리정보시스템의 형태는 하나의 단독 시스템에서 모든 처리를 수행하도록 하거나, 아니면 원시적인 형태로 데이터 응용 프로그램과 데이터 제공부분을 구별하였다. 현재 주로 사용되고 있는 분산 컴퓨팅 플랫폼의 단계에서는 응용 프로그램들이 일반적 서비스의 일부를 응용 프로그램 서버로 분리시켜 처리한다. 이때, 데이터베이스 또는 데이터 저장소, 공간 데이터 프로바이더, 응용 프로그램 서버와 응용 프로그램 그리고, 사용자 인터페이스 사이의 연결은 개방형 지리정보시스템 인터페이스를 사용하여 다른 시스템에 레이어를 개방할 수 있게 한다. 이 시스템들은 단독 지리정보시스템에서 중복되어 제공되는 기능들을 분산 컴퓨팅 환경에서 제공할 수 있게 한다.

이러한 분산된 개방형 지리정보시스템 인터페이스를 구성할 때 주로 이용되는 구현 명세는 CORBA, OLE/COM와 SQL 등으로 구분할 수 있고, 이중에서 특정 운영체제에 종속적이지 않는 CORBA 구현명세를 이용한 많은 연구들이 수행되었다[13-15].



(그림 1) CORBA Requests의 일반적인 경로[10]

이러한 CORBA 서비스가 호출되는 일반적인 경로는 (그림 1)과 같다. 이러한 일반적인 경로를 CORBA 구현 시 사용하는 경로는 (그림 2)와 같다. VisiBroker의 Static Invocation Interface(SII)를 통해 구현할 때, 이 구현 모델은 서로의 참조 객체를 주고 받을 때 송신과 확인 과정 등 여러 단계를 거치므로 많은 오버헤드를 갖는다.



(그림 2) SII를 위한 VisiBroker Sender와 Receiver간의 Request 경로

구체적으로 기술하면, 일반적으로 VisiBroker sender는

ttcp_sequence 클래스에서 정의된 sendStructSeq 메소드를 위한 stub을 호출한다. 요청된 사항은 CORBA :: Object와 PMCStubInfo 클래스의 send 메소드를 통해서 전달된다. 마지막으로 Internet Inter-ORB Protocal(IIOP)를 수행하는 PMCIIOPStream 클래스의 메소드를 통해서 요청이 되어 진다. receiver 측에서는 basic object adapter(BOA)에 요청하여, PMCIIOPStreamclass의 메소드를 사용해서 패킷을 읽는다. BOA는 스켈레톤(_sk_ttcp_sequence :: skeleton)을 통해 요청을 demultiplex 작업을 수행한다. 스켈레톤은 ttcp_sequence_i 구현 클래스의 sendStructSeq 메소드에 대한 up call을 만들고, 구현된 객체를 식별한다. 이러한 과정들 중에 marshaling과 데이터 복사하는 단계, 그리고 demarshaling과 demultiplexing 단계에서 오버헤드를 가진다[8, 10-12].

개방형 지리정보 인터페이스 구축 시에도 이러한 CORBA의 기본 문제를 가지고 있기 때문에 CORBA의 구현 명세는 다른 방법들을 이용한 미들웨어 보다 성능이 비효과적임을 알 수 있다[11,12]. 비교의 결과를 살펴보면 원격 데이터 전송 시 CORBA는 C/C++ 버전의 미들웨어보다 75~80% 정도의 전송 속도를 보인다.

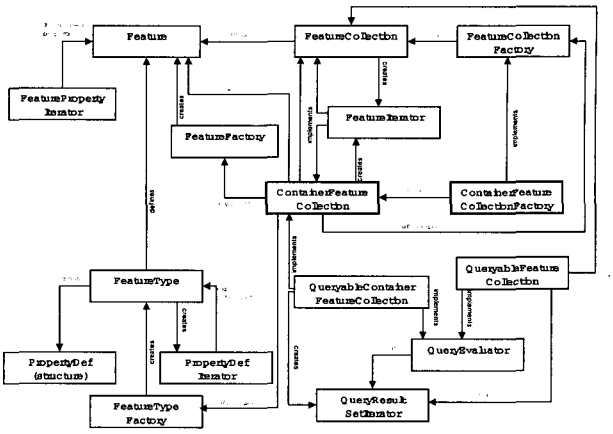
다양한 데이터의 원본으로 많은 양의 데이터를 처리하는 개방형 지리정보 인터페이스의 특성상 오버헤드를 가지는 CORBA를 사용하여 구축한다면 성능 면의 문제를 가질 수 있다. 따라서, 이 논문에서는 CORBA를 이용하여 원격객체를 전송할 때, 데이터 전송속도를 빠르게 할 수 있는 인터페이스를 제시한다.

3. 개방형 지리정보시스템 표준 인터페이스

개방형 지리정보시스템의 CORBA 구현명세인 "Open GIS Simple Feature Specification for CORBA"는 지리정보시스템 소프트웨어 개발자들이 CORBA 분산기술을 이용하여 공간 데이터에 접근하고 처리하기 위한 다양한 인터페이스를 CORBA IDL(Interface Definition Language)로 정의하고 있다. 이 표준 인터페이스는 개발자가 인터페이스를 구현할 때에 최대한 유연성을 허용하도록 구성되었다[16, 17]. 특히, 이 명세서는 여러 가지 객체 랩핑(wrapping)을 통해 데이터베이스에 있거나 파일 저장소에 있는 데이터들을 기존의 응용 프로그램에서 이용하도록 설계 되었다.

(그림 3)에서 피처 모듈은 지형공간 피처들과 피처 집합을 생성, 접근, 질의하기 위한 모듈을 말한다. 여기에서 피처는 공간 데이터 속성과 비공간 데이터 속성들을 가지고 있는 현실세계의 객체를 표현하고, 그 서버구현 객체는 FeatureFactory를 통하여 생성된다.

개방형 지리정보시스템 서버의 구현은 명시된 인터페이스들을 통하여 외부의 클라이언트들에게 이러한 구조를 보여줄 수 있는 방법을 제공한다. (그림 3)은 관련된 다양한 인터페이스 방법의 도식적인 표현을 제공한다[6, 17].



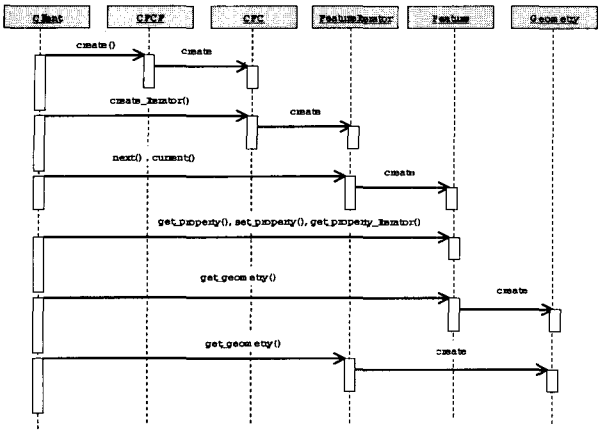
(그림 3) 개방형 지리정보시스템 피처 인터페이스[7]

서버 응용 프로그램은 정적 바인딩을 통하거나, CORBA Naming과 Trader 서비스와 같은 동적 바인딩 메커니즘을 통하여 클라이언트에게 지리정보시스템 데이터베이스와 개념적으로 동등한 CFC(ContainerFeatureCollection) 인터페이스를 사용할 수 있도록 제공한다. 즉, 피처 인터페이스를 통하여 클라이언트에게 사용하게 할 수 있는 피처들 각각의 집합을 포함하거나 소유한다. Feature들은 FeaturePropertyIterator에 의해 반복되는 속성들의 집합을 가지고 있고, FeatureType 객체들은 이러한 집합을 갖는 속성들을 정의하며, 이러한 집합은 피처 인터페이스를 통해서 가능하다. FeatureType 객체는 각각의 속성을 정의하는 구조체 PropertyDef들을 그룹화한다. 이들은 FeatureType 객체에 의해 만들어진 PropertyDefIterator를 통해서 접근된다.

Open GIS simple feature for CORBA 스펙의 구현 명세에는 CFC 인터페이스를 이용하는 방식과 QCFC(QueryableContainerFeatureCollection) 인터페이스를 이용하는 방식이 있다. CFC는 Feature의 타입에 의해 해당하는 Feature들을 접근하는 방식이고, QCFC은 SQL 질의문에 의해 해당하는 Feature들을 접근하는 방식이다. 클라이언트는 미들웨어인 CORBA 서버를 통해서 간접적으로 데이터베이스 서버에 접근하게 된다. 처음 바인딩할때 생성되어있는 CORBA 객체를 식별할 수 있다. CORBA 서버는 TCP/IP를 통해서 데이터베이스 서버에 접근한다. 이 장에서는 두 방식 중 CFC 접근 방법에 대해서만 설명한다. 두 방법 모두 객체를 넘겨줄 때 다른 미들웨어에 비해 데이터 전송 시간을 많이 요구하는 인터페이스[8, 9]를 가지고 있다.

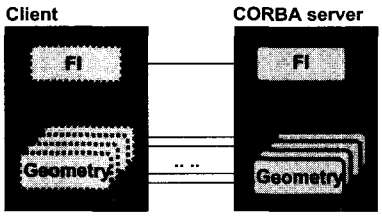
CFC 인터페이스는 (그림 4)와 같이 도식화 될 수 있다. 클라이언트는 먼저 바인딩을 통해 CFC를 생성 시켜주는 Factory 객체인 CFCF(ContainerFeatureCollectionFactory)를 찾아 CFCF의 참조를 얻게 된다. 얻어진 참조를 통해 Feature의 타입과 속성에 대한 정보를 가지고 create를 호출하게 되면, CFCF는 CFC를 생성한 후 참조를 반환한다. 클라이언트는 얻어진 참조를 통해서 Feature를 추가, 삽입, 삭

제, 변경의 작업을 수행할 수 있다.



(그림 4) CFC 인터페이스

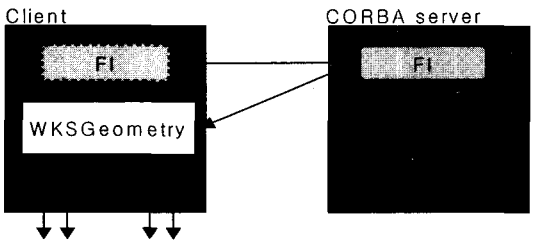
이러한 구조를 가지는 기존의 인터페이스는 (그림 5)와 같다.



(그림 5) 기존 접근 방법

(그림 5)는 기존 FeatureIterator 인터페이스를 통해서 각각의 Feature에 접근하는 방법을 보여주고 있다[7, 15]. FI 객체는 해당하는 Feature들의 정보를 가지고 있다. 클라이언트는 FI의 멤버 함수인 get_geometry를 통해서 Geometry를 접근하는 인터페이스를 가지고 있다. get_geometry는 현재 포인터 위치의 Feature를 Geometry 타입으로 얻고 포인터를 증가시킨다. 이러한 이유로 클라이언트가 Feature들에 대한 접근을 수행할 때마다 추가 접근을 요구하게 된다.

이러한 이유로 이 논문에서는 (그림 6)처럼 FI가 새로운 Feature나 Geometry 객체를 생성하는 것이 아니고 WKSGeometry 타입으로 Feature의 값을 클라이언트에게 바로 전달함으로써 클라이언트가 다시 접근하게 하는 비용을 줄일



(그림 6) 개선된 접근 방법

수 있다. WKSGeometry에는 WKSCollectionType의 형태로 또 다른 WKSGeometry들이 존재한다. 그래서 클라이언트는 FI에 한번의 접근으로 해당하는 Feature들을 모두 읽어서 처리 할 수 있게 된다.

4. 인터페이스 설계

Open GIS simple feature specification for CORBA 인터페이스에는 FeatureIterator, FeaturePropertySetIterator, PropertyDefIterator, QueryResultSetIterator, 그리고 GeometryIterator, 총 5개의 Iterator 인터페이스가 존재한다. Iterator 인터페이스는 각각의 객체들이 여러 개가 존재하는 상황에서 각각의 객체들을 순차적으로나 비순차적으로 개개의 객체들에 접근하는 방법을 제공하는 인터페이스이다. 각 인터페이스 모두 CORBA에서 지원하는 점대점 통신을 통해, 한 객체씩 접근하는 방식이다. 그러나, 지리정보시스템의 대용량 공간 데이터를 전송하기에는 전송 속도 면에서 다른 미들웨어보다 많은 시간을 요구하는 문제를 가지고 있다. 따라서, 이 논문에서는 개방형 지리정보시스템에서 정의된 WKS(Well-Known Structure) 구조로 WKSGeometry 객체를 생성하고, 이를 이용하여 한번 연결 시에 모든 객체를 클라이언트 쪽에 전송하도록 인터페이스를 제시한다.

4.1 WKSGeometry 구조

이 절에서는 개방형 지리정보시스템의 WKSGeometry에 대해 기술한다.

Feature나 Geometry 객체는 CORBA 서버에 객체 형태로 존재하지만, WKS 타입은 실질적인 값을 갖는 구조체이다. 우리는 WKS 구조체를 이용하여 데이터 연결 시 한번에 모든 데이터를 WKSGeometry 형태로 전달한다.

```
struct WKSPoint {
    double x ;
    double y ;
} ;
typedef sequence<WKSPoint> WKSPointSeq ;
typedef sequence<WKSPoint> WKSLineString ;
typedef sequence<WKSLineString> WKSLineStringSeq ;
typedef sequence<WKSPoint> WKSLinearRing ;
typedef sequence<WKSLinearRing> WKSLinearRingSeq ;
struct WKSLinearPolygon {
    WKSLinearRing externalBoundary ;
    WKSLinearRingSeq internalBoundaries ;
} ;
typedef sequence <WKSLinearPolygon> WKSLinearPolyonSeq ;
enum WKSType {
    WKSPointType,WKSMultiPointType,
    WKSLineStringType,WKSMultiLineStringType,
    WKSLinearRingType, WKSLinearPolygonType,
    WKSMultiLinearPolygonType,
    WKSCollectionType
} ;
```

```
union WKSGeometry // near-equivalent to the
                  'CoordinateGeometry of the spec'
   switch (WKSType) {
          case WKSPointType :
                WKSPoint point ;
          case WKSMultiPointType :
                WKSPointSeq multi_point ;
          case WKSLineStringType :
                WKSLineString line_string ;
          case WKSMultiLineStringType :
                WKSLineStringSeq multi_line_string ;
          case WKSLinearRingType :
                WKSLinearRing linear_ring ;
          case WKSLinearPolygonType :
                WKSLinearPolygon linear_polygon ;
          case WKSMultiLinearPolygonType :
                WKSLinearPolygonSeq multi_linear_polygon ;
          case WKSCollectionType :
                sequence<WKSGeometry> collection ;
} ;
struct Envelope {
   WKSPoint minm ;
   WKSPoint maxm ;
} ;
```

(그림 7) WKS 구조의 IDL 명세

(그림 7)은 CORBA명세서의 WKS구조를 나타내는 IDL 파일의 일부이다. WKSPoint 타입은 x와 y의 좌표 값을 갖고, WKSLinearRing은 WKSPoint의 집합으로 이루어진다. WKSPolygon은 외부의 경계와 내부의 홀들로 구성된다. 또, WKSGeometry는 이러한 모든 구조체들을 갖는 타입으로, WKSGeometry 안에는 또 다른 WKSGeometry의 집합을 갖는 WKSCollectionType이 존재한다.

4.2 get_WKSGeometries 알고리즘

WKS 구조체로 모든 데이터를 한번에 전달하기 위한 get_WKSGeometries 함수에 대한 알고리즘이 (그림 8)에 나타나 있다.

get_WKSGeometries함수는 먼저 데이터베이스에서 읽은 데이터를 저장하기 위한 클래스를 할당하고, 데이터베이스 접속을 위한 초기 명령을 수행한다. 데이터베이스에 접속해서는 질의를 수행하여 전체 데이터 개수를 파악하고, 데이터베이스에 존재하는 데이터 개수만큼 필요한 공간을 확보한다. 공간이 확보되면 처음 데이터부터 읽으면서 필요한 데이터들을 저장하고, 전체 데이터를 반환하기 위한 WKS Geometry 객체를 만든다.

```
OGIS :: WKSGeometry* FeatureIteratorImpl :: get_WKSGeometries()
{
      데이터베이스에서 읽은 데이터를 임시 저장하기 위한 클래스 할당
      데이터의 크기를 기억 하기 위한 변수 선언
      데이터베이스 접속을 위한 초기 명령
      데이터베이스에 접속
      질의를 수행하여 현재 저장된 데이터의 개수를 파악
            try
            {
         질의 수행결과 갯수를 저장하기 위한 변수 선언
         결과 레코드의 처음으로 이동
         전체 갯수를 파악

         데이터 베이스에 존재하는 데이터 만큼의 공간을 확보한다.
            }
            catch (예외상황이 발생하면)
            {
                오류 메세지와 함께 종료.
            }

      전체 객체를 가져오기 위한 질의를 수행
      if (질의 수행이 정상적으로 수행하지 못한경우)
            {
                오류 메세지와 함께 종료
            }

            try
            {
            기하 객체의 갯수를 저장하기 위한 변수 선언
            기하 객체를 읽기 위한 데이터베이스 저장구조 선언
            값의 집합을 읽기 위한 데이터베이스 저장구조 선언
            점들의 값의 집합을 읽기 위한 데이터베이스 저장구조
            선언

            첫번째 레코드로 이동한다.

            while (마지막 레코드에 도달하면 종료)
                  {
                  질의 결과에서 각 ID, CODE, SHAPE, 좌표값
                    등에 해당하는 값을 읽어서 저장한다.
                  다음 레코드를 읽는다.
                  }
            }
            catch (예외상황이 발생하면)
            {
                  구현 객체의 참조를 넘기지 못한다
            }
      for( )
         {
         기하를 반환
         기하 데이터의 좌표 정보를 가져오기 위한 준비 작업수행
         각각의 좌표정보를 기하 타입으로 변환
         반환을 위한 WKSGeometry object 생성
         }
      }
}
```

(그림 8) get_WKBGeometries의 알고리즘

제안된 get_WKBGeometries인터페이스는 지리정보시스템에서 주로 요구되는 데이터의 검색 시 CORBA의 낮은 전송속도 문제를 극복하기 위해, WKSGeometry 형태로 모든 데이터들을 클라이언트에게 넘겨주도록 하여 성능 향상을 이룰 수 있게 한다.

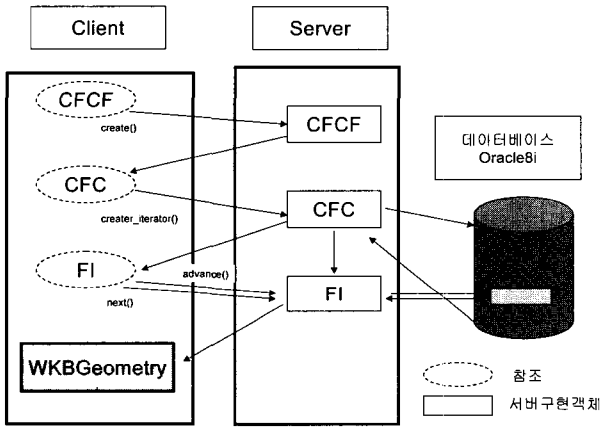
5. 구현 및 결과 분석

이 장에서는 4장에서 제안한 개선된 인터페이스를 적용한 시스템을 구현한다. 또한 구현된 시스템과 기존 인터페이스를 가진 시스템을 통한 데이터 접근 소요시간을 측정

하였다.

구현을 위한 시스템 환경은 데이터베이스로는 펜티엄 180Mhz, 램 40MB의 시스템과 운영체제 Alzza Linux 6.2상에서 오라클 8i를 사용하였으며, CORBA 서버로는 펜티엄 400Mhz, 램 128MB 시스템과 운영체제 Window 2000환경에서 Visibroker for cpp 3.3.3를 사용하였다. 그리고, 클라이언트는 Window 환경에서 실험을 하였다. (그림 9)는 개선된 인터페이스를 적용한 시스템을 나타낸다.

내부적인 인터페이스를 살펴보면 클라이언트가 바인딩을 통해 CFC를 생성 시켜주는 Factory 객체인 CFCF를 찾아 CFCF의 참조를 얻게 된다. 얻어진 참조를 통해 Feature의 타입과 속성에 대한 정보를 가지고 create를 호출하게 되면, CFCF는 CFC를 생성한 후 참조를 반환한다. 클라이언트가 create_iterator() 메시지를 CFC 서버 객체에 보내면, 그 서버 객체는 데이터베이스에 질의하여 해당하는 OID를 읽어오고 FeatureIterator 서버객체를 생성한 뒤 그 참조를 클라이언트에게 반환한다. 이렇게 넘겨진 참조를 이용하여 해당하는 객체들을 클라이언트에 WKSGeometry 형태로 넘겨준다.



(그림 9) 개선된 인터페이스를 적용한 시스템

실험 데이터는 도시의 건물과 도로를 나타내는 Polygon들을 대상으로 했으며, 각각의 튜플은 ID, CODE, SHAPE으로 구성되었다. 이때 객체내의 Private데이터는 가지고 있지 않고, Public데이터만 가지고 있는 상태이고, 데이터만을 전송한다. (그림 10)은 실험데이터의 일부분을 나타낸다.

```
ID CODE
---------- ----------
SHAPE(SDO_GTYPE, SDO_SRID, SDO_POINT(X, Y, Z),
SDO_ELEM_INFO, SDO_ORDINATES)
-----------------------------------------------------

        992 4112
SDO_GEOMETRY(2003, NULL, NULL, SDO_ELEM_INFO_ARRAY
(1, 1003, 1), SDO_ORDINATE_ARRAY(197991, 442085, 197991,
442096, 197994, 442096, 198001, 442088, 198001, 442085, 197991,
442085))

        993 4112
SDO_GEOMETRY(2003, NULL, NULL, SDO_ELEM_INFO_ARRAY
(1, 1003, 1), SDO_ORDINATE_ARRAY(197991, 442084, 197991,
442077, 198006, 442077, 198006, 442081, 198002, 442084, 197991,
442084))

        994 4112
SDO_GEOMETRY(2003, NULL, NULL, SDO_ELEM_INFO_ARRAY
(1, 1003, 1), SDO_ORDINATE_ARRAY(198016, 442098, 198023,
442094, 198019, 442087, 198012, 442092, 198016, 442098))

        995 4113
SDO_GEOMETRY(2003, NULL, NULL, SDO_ELEM_INFO_ARRAY
(1, 1003, 1), SDO_ORDINATE_ARRAY(198031, 442112, 198022,
442112, 198022, 442108, 198017, 442100, 198024, 442095, 198031,
442106, 198031, 442112))
                ...
```

(그림 10) 실험 데이터

(그림 11) 클라이언트에서 모든 객체를 검색한 화면

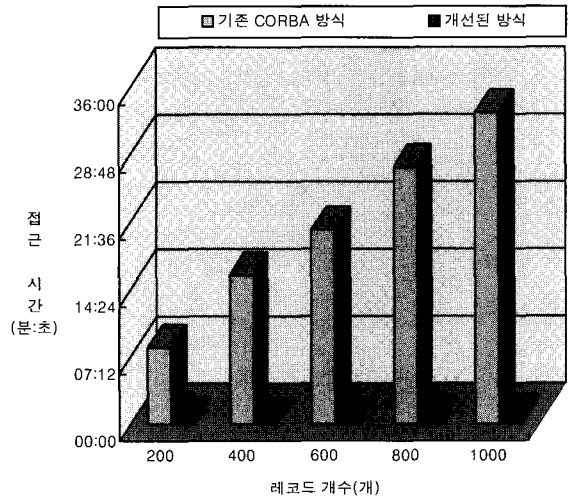
(그림 11)은 실험데이터를 모두 검색하여 화면에 출력한 예이다. 기존의 get_geometry 방식과 이 논문에서 제안한 get_WKSGeometries 방식으로 검색을 수행한 시간이 <표 1>이다. 이때 레코드의 개수를 200개, 400개, 600개, 800개, 1000개로 다르게 하여 실험하였다.

〈표 1〉 실험 결과

접근 방법 / 레코드 개수	기존 방법	개선 방법
	get_geometry()	get_WKSGeometries()
200	08 : 14	00 : 19
400	15 : 54	00 : 20
600	20 : 48	00 : 27
800	27 : 24	00 : 30
1000	33 : 15	00 : 43

(그림 12)는 기존의 인터페이스인 get_geometry와 이 논문에서 제안한 개선된 인터페이스인 get_WKBGeometries

에 대한 실험결과를 그래프로 표현한 것이다. 실험의 대상이 되는 레코드 개수는 200개에서 1000까지 200개씩 증가시켰고, 접근시간은 분과 초로 표현하였다. 그림에서 살펴보면 기존 방식을 사용한 경우는 개수에 비례해서 접근시간이 급하게 증가하고, 개선한 인터페이스를 이용하면 개수에 비례한 접근시간의 기울기가 완만한 것을 알수 있다. 또한 동일한 개수의 접근시간도 많은 차이가 나는 것을 보여준다.



(그림 12) 기존 방법과 개선된 접근 방법

기존의 모델과 새로 제안된 모델의 데이터 전송 시간을 분석하면 다음과 같다. 먼저 성능을 비교하기 위해 몇 가지 기호들을 정의한다.

α : 한 객체를 전송하는 시간

δ_1 : 순수 데이터 전송에 사용되는 시간을 제외한 모든 시간

δ_2 : 객체를 전송하기 위해 연결하는 시간

이렇게 할 경우, 기존의 접근 방법은 CORBA에서 지원하는 점대점 통신 방법을 사용하고 있기 때문에 각 Feature에 대한 접근 비용은 다음과 같이 계산 된다. 이때 피처 객체 x는 n개가 있다고 가정하고, 이때 기존의 방법을 사용한 소요시간을 T1으로 계산한다.

$$T1 = \delta_1 + \sum_{i=1}^{n} (\alpha + \delta_2) x_i$$

이 논문에서 제안된 방법을 사용한 소요시간을 T2라고 하면, T2 값은 다음과 같다.

$$T2 = \delta_1 + \delta_2 + \sum_{i=1}^{n} \alpha x_i$$

실험 결과를 수식에 대입해 보면 평균적으로 한 객체를 전송하는데 드는 시간(α)은 약 0.03초, 객체 전송만을 위한 시간(δ_2)은 약 2.02초가 소요된다. 객체를 전송하는 작업외의 모든시간(δ_1)은 약 7.78초가 소요된다. 이때 전송객체가 600개일 경우, T1은 7.78 + ((2.02 + 0.03)×600) ≒ 20분 48초가 되고, T2는 7.78 + 2.02 +(0.03×600) ≒ 27초가 된다.

이때 T1과 T2 모두 δ_1은 공통적으로 소요되나, 객체를 전송하기 위한 연결시간은 T2가 T1보다 δ_2을 적게 사용하는 것을 알 수 있다. 따라서 새로운 모델 방법은 기존의 방식보다 전송 시간면에서 기존의 방식과의 비교해서 훨씬 적은 시간으로도 데이터의 전송을 수행할 수 있음을 알 수 있다.

이 논문에서 제안한 FeatureIterator의 get_WKSGeometries 인터페이스는 CORBA 서버에 객체를 생성하지 않고 객체의 집합을 WKS 값으로 클라이언트에 전달하였다. 제안된 방법을 사용하면 CORBA의 전송 속도 비용을 줄일 수 있다.

6. 결 론

개방형 지리정보시스템을 위한 CORBA 서비스는 기하와 피처를 포함하는 지형공간 정보에 접근하고 관리할 수 있는 수단을 제공할 수 있고, 언어와 플랫폼 및 운영체제에 독립적인 서비스를 제공할 수 있다. 그러나 CORBA의 서비스는 점대점 통신만을 지원하고, 이로 인한 분산 시스템의 데이터 통신 비용을 많이 요구한다.

이 논문에서는 Open GIS simple feature for CORBA의 구현 명세에서 데이터를 효율적으로 전송할 수 인터페이스를 제안하고 구현하였다. 새로운 인터페이스를 설계하고 구현한 부분은 CORBA 인터페이스 중에 FeatureIterator이다. FeatureIterator는 current(), next(), next_n() 그리고 get_geometry()를 이용하여, Feature나 Geometry 객체를 생성하고 접근할 수 있게 하였다. 기존의 get_geometry는 하나의 객체를 CORBA 서버에 생성시키고, 클라이언트가 다시 접근하기 때문에 접근 비용이 많이 요구된다. 그러나, 이 논문에서 제안된 get_WKSGeometries 인터페이스는 기존의 방식처럼 CORBA 서버에 객체를 생성하지 않고, WKS의 값으로 클라이언트에게 직접 전달하여 접근 시간을 줄이도록 하였다.

제안한 모델을 검증하기 위한 실험에서는 새로 제안한 접근 방식의 소요시간이 기존 방식에 비해서 데이터 개수에 대한 소요 시간의 감소가 뚜렷하게 나타났다. 제안된 방식은 get_WKSGeometries 인터페이스를 이용하면 CORBA의 단점인 접근 비용을 줄일 수 있고, 효율적인 개방형 지리정보시스템을 구현할 수 있다.

참 고 문 헌

[1] 한국 전산원, Internet 지리정보시스템의 데이터 공유 표준연구, 1998.

[2] M. F. Worboys, "GIS : A Computing Perspective," Taylor & Francis, 1995.
[3] H. A. Jacobsen and A. Voisard, "CORBA-Based Interoperable Geographic Information Systems," Euro Parallel and Distributed Systems Conference, 1998.
[4] Open GIS Consortium, The OpenGIS Guide, 1998.
[5] A. Dogac, C. Dengi and M. T. zsu, "Building Interoperable Databases on Distributed Object Management Platforms," Communications of ACM, Vol.41, No.9, pp.95-103, 1998.
[6] Open GIS Consortium, The OpenGIS Abstract Specification Model, version 4, 1999.
[7] Open GIS Consortium, OpenGIS Simple Features Specification for CORBA, Revision 1.0, 1998.
[8] A. S. Gokhale, B. Natarajan, "GriT : A CORBA-Based GRID Middleware Architecture," HICSS, pp.319-329, 2003.
[9] A. S. Gokhale, D. C. Schmidt, B. Natarajan and N. Wang, "Applying model-integrated computing to component middleware and enterprise applications," CACM 45(10), pp.65-70. 2002.
[10] A. S. Gokhale and D. C. Schmidt, "Evaluating CORBA latency and scalability over high speed ATM networks," Proceedings of the International Conference of Distributed Computing Systems, 1997.
[11] P. Tuma and A. Buble, "Open CORBA Benchmarking," Proceedings of SPECTS, 2001,
[12] A. Buble, L. Bulej and P. Tuma, "CORBA Benchmarking : A Course With Hidden Obstacles, Proceedings of the IPDPS Workshop on Performance Modeling," Evaluation and Optimization of Parallel and Distributed Systems (PMEOPDS), pp.279-284, 2003.
[13] 장염승 외 2, "GEUS 기반 OpenGIS 서버의 설계 및 구현", 개방형GIS연구회논문지, 제2권 제1호, pp.5-16, 2000.
[14] 안경환 외 2, "CORBA를 이용한 OpenGIS 기반 미들웨어 구현", 개방형GIS연구회논문지, 제1권 제1호, pp.19-28, 1999.
[15] 강병극 외 4, "CORBA를 이용한 인터넷 GIS 통합 시스템 설계," 정보처리학회논문지D, 제8-D권, 제3호, pp.193-202, 2001.
[16] Open GIS Consortium, OpenGIS Simple Features Specification for SQL, Revision 1.0, 1998.
[17] Open GIS Consortium, OpenGIS Simple Features Specification for OLE/COM, Revision 1.1, 1999.
[18] http://www.opengis.org, 2003.
[19] http://www.omg.org, 2003.

김 상 호

e-mail : shkim@dblab.chungbuk.ac.kr
1997년 충북대학교 컴퓨터과학과
1999년 충북대학교 대학원 전산학과 석사
1999년~현재 충북대학교 대학원 전자계산학과 박사과정
관심분야 : 시공간 데이터베이스, Web Visualization, Component GIS

지 정 희

e-mail : jhchi@dblab.chungbuk.ac.kr
1998년 충주대학교 전자계산학과
2001년 충주대학교 대학원 전자계산학 전공(이학석사)
2001년~현재 충북대학교 전자계산학과 박사과정
관심분야 : 시공간 데이터베이스, Temporal GIS, 이동객체 관리 기법

류 근 호

e-mail : khryu@dblab.chungbuk.ac.kr
1976년 숭실대학교 전자계산학과
1980년 연세대학교 공학대학원 전산전공(공학석사)
1988년 연세대학교 대학원 전산전공(공학박사)
1986년 육군군수지원사전산실(ROTC 장교), 한국전자통신연구소(연구원), 한국방송통신대, 전산학과(조교수) 근무
1989~1991년 Univ. of Arizona 연구원(TempIS Project)
1986년~현재 충북대학교 전기전자 및 컴퓨터공학부 교수
관심분야 : 시간 데이터베이스, 시공간 데이터베이스, Temporal GIS, 객체 및 지식베이스 시스템, 지식기반 정보검색 시스템, 데이터 마이닝, 데이터베이스 보안 및 Bioinformatics