# GALIS 구조 기반 실시간 분산 위치 데이타 서버 구현†

# An Implementation of distributed Real-time Location Data Server based on the GALIS Architecture

이준우*, 이운주**, 이호**, 나연묵***

Joon-Woo Lee, Woon-Ju Lee, Ho Lee, Yun-Mook Nah

**요 약** LBS 시스템 분야의 도전할 만한 과제는 이동 객체를 다루는 수준부터 수백만 개의 이동 객체를 처리할 수 있는 높은 신뢰도의 시스템 아키텍처를 구현하는 것이다. GALIS로 명명된 아키텍처는 각각 다른 지리적 영역과 시간적 영역에 연관된 레코드를 유지하는 다수의 프로세서로 구성된 클러스터 기반 분산 컴퓨팅 시스템 아키텍처이다. 이 논문에서는 실시간 분산 객체 프로그래밍과 실시간 분산 컴퓨팅 시스템 디자인을 지원하는 미들웨어 실행 엔진을 포함하는 TMO 프로그래밍 기법을 적용하여 GALIS의 주요 요소를 구성하는 위치 데이타 서버의 프로토타입을 구현했다. 본 논문에서는 실질적으로 위치 측위 정보가 발생하는 과정과 이런 위치 정보와 위치 관련 질의가 어떻게 처리되는지도 기술하였다. 몇 가지 실험은 분산을 통해 프로세서의 부하를 줄여주어 성능향상이 있음을 살펴볼 수 있었다.

***Abstract*** A challenging task in the LBS system engineering is to implement a highly scalable system architecture which can manage moderate-size configurations handling thousands of moving items as well as upper-end configurations handling distributed computing system architecture that consists of multiple data processors, each dedicated to keeping records relevant to a different geographical zone and a different time zone. In this paper, we explain a prototype location data server structuring major components of GALIS by employing the TMO programming scheme, including the execution engine middleware developed to support real-time distributed object programming and real-time distributed computing system design. We present how to generate realistic location sensing reports and how to process such location reports and location-related queries. Some experimental results showing performance factors regarding distributed query processing are also explained.

**주요어** : 위치기반서비스, 위치 데이타 서버, GALIS, 지리정보시스템

**Keywords** : Location-Based Service, Location Data Server, GALIS, GIS

## 1. 서 론

위치 기반 서비스(Location Based Service)는 위치 측위 기술, 무선 통신기술, 데이타베이스 서버 기술등 급속도로 발전하는 기술이 집약된 컴퓨팅 응용(application)의 새로운 분야이다. LBS 시스템은 다양

† 이 연구는 2004학년도 단국대학교 대학연구비의 지원으로 연구되었음

* 단국대학교 컴퓨터공학과 박사과정 jwlee@dblab.dankook.ac.kr

** 단국대학교 컴퓨터공학과 석사과정 {wjlee, hlee}@dblab.dankook.ac.kr

*** 단국대학교 컴퓨터 공학과 교수 ymnah@dblab.dankook.ac.kr

한 형태의 위치 관련 정보를 제공한다. 예를 들어 시스템의 현재 고객 또는 잠재적인 미래의 고객의 현재와 과거의 위치 정보, 관심 있는 시설의 현재와 과거의 위치 정보 등이 있다. 이는 또한 다음과 같은 질의에 응답하기에 적합해야 한다: "특정인의 반경 100m 범위에 지난 6시간 동안 어떤 사람들이 있었는가?", "나의 현재 위치의 반경 1Km범위의 별 네 개 등급의 호텔은 어떤 것이 있나?" 등.

과거 연구에서 위치 정보의 저장 구조에 대한 다양한 접근 방법들이 소개되었다. RT-tree, 3D R-tree, STR-tree, TB-tree, hashing based index, TPR-tree 등 다양한 인덱스 구조들이 제안되었다[1][2][3][4][5][6]. 그러나 대부분의 최근 연구활동 들은 단일 노드 구조의 LBS 시스템에 집중되어 있다. LBS 시스템 공학의 주요 과제중의 하나는 높은 신뢰성을 지원하며 백만 단위의 이상의 이동 객체의 위치 정보를 저장하는 것이다. 이 분야에서 단일 노드를 다루는 과거와 현재의 연구 활동들과는 달리, 본 논문에서는 다수의 휴대폰 사용자를 다루는 대용량 이동 객체를 처리하기 위한 방법을 고안한 새로운 분산 컴퓨팅 아키텍처를 연구하였다. 우리는 만족할 만한 수준의 신뢰도로 저장하기 위해서는 필연적으로 데이타베이스를 중심으로 한 컴퓨팅 노드 클러스터의 사용을 기반으로 한 분산 병렬 처리가 필요하다고 생각했다.

GALIS(Gracefully Aging Location Information System)로 명명된 이 구조는 2002년도부터 진행되어 왔다[7][8][9]. 이는 각각 다른 지리적 영역과 시간적 영역에 연관된 레코드를 유지하는 다수의 프로세서로 구성된 클러스터 기반 분산 컴퓨팅 시스템 구조이다. 이 구조는 서로 다른 지리적 영역을 통과해 이동하는 객체가 있는 경우 해당 영역을 담당하는 프로세서들 간의 레코드들에 대한 동적 재배치를 지원하며, 과거 위치 레코드들의 나이(age)가 계속 증가하도록 한다. 또한 프로세서에 할당된 지리적 영역은 해당 영역에 포함된 객체의 수가 미리 정해놓은 상한선을 넘거나 하한선보다 적어지는 경우, 부하 분산을 위해 동적으로 변화할 수 있다. 효율적인 병행 수행 기법은 GALIS에서 제공하는 중요한 요소이다. 센서들(또는 릴레이 스테이션)에서 끊임없이 발생하는 이동 객체의 새로운 위치 정보와 다양한 클라이언트에서 오는 위치 관련 질의의 효율적 병행 처리를 순조롭게 하는 것이 특히 중요하다. 이러한 데이타 처리는 데이타베이스 중심의 분산 컴퓨팅 노드들의 협력 작업을 자주 수반하고 때로는 노드들 내에서 데이타베이스의 협력적재 구성이 일어나기도 한다.

이 논문에서는 TMO(Time-triggered Message-triggered Object) 프로그래밍 기법을 적용하여 GALIS의 주요 요소를 구성하는 위치 데이타 서버의 프로토타입을 구현했다. 우리는 실시간 분산 객체 프로그래밍과 실시간 분산 컴퓨팅 시스템 디자인[10][11][12]을 위한 TMO 기법이 효율적인 병행 수행을 지원하는 고성능의 미들웨어 구현에 상대적으로 용이하다고 판단했다. 그리고 위치 데이타의 저장을 공간 데이터베이스 엔진인 GMS(Geomania Millennium Server)를 이용하여 다른 관계 데이터베이스 시스템을 이용한 것보다 향상된 성능을 보여주었다. 본 논문의 실험에서 실 세계 사용자들의 실제 이동 패턴과는 다른 특정 지역에서 랜덤하게 생성된 데이타를 사용하였고 위치 데이타에 대한 저장과 위치 관련 질의를 어떻게 처리하는지를 본 논문에 기술하였다. 또한 데이타 저장과 분산 질의 처리에 관련된 몇 가지 성능평가를 하였다.

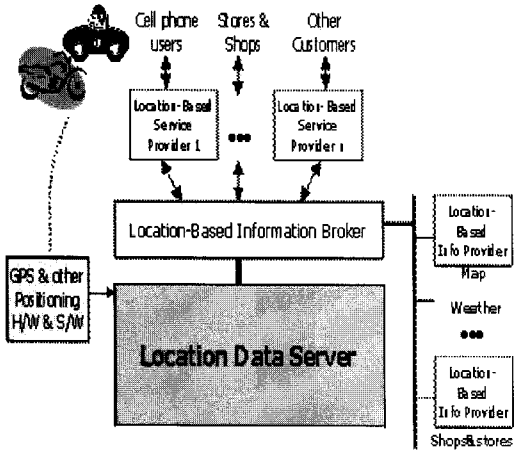
본 논문의 구성은 다음과 같다. 2장에서는 LBS 시스템과 GALIS 구조의 기본 구성에 대해 기술하였다. 3장에서는 위치 데이타 서버의 TMO기반 구현 접근 방법에 대하여 설명 하였으며 4장에서는 분산 컴퓨팅의 이점을 보여주는 실험 결과에 대해 기술했으며 마지막으로 5장에서는 이 논문의 결론을 내린다.

## 2. 관련 연구

### 2.1 LBS 시스템

LBS 시스템의 기본 구성은 그림 1과 같다. 측위 시스템은 이동 객체의 현재 위치를 획득하기 위해 사용된다. 위치 데이타 서버(LDS : Location Data Server)는 메인 메모리 기반과 디스크 기반의 데이타베이스 시스템 모두를 사용하여 이동 객체의 현재와 과거 위

치를 저장하고 인덱싱 하는데 사용된다. 위치 기반 정보 중개자(LBIB; Location-Based Information Broker)는 위치 서비스 제공자(LSPs : Location-Based Service Providers)의 요청을 받아 LDS에서 위치 정보를 획득하여, 이를 위치 기반 정보 제공자(LBIP : Location Info Provider)에서 제공하는 지리적 영역과 연관된 정보와 조합한다.

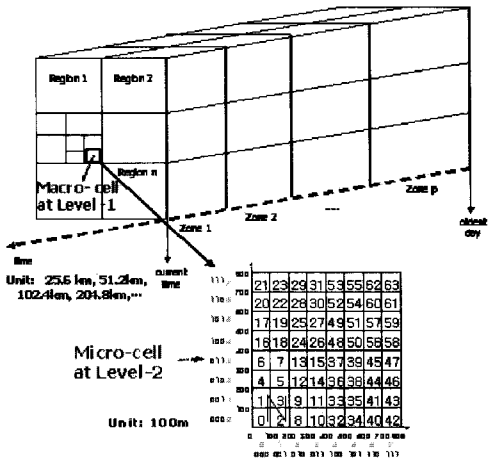



<그림 1> LBS 시스템 구조

고객들은 휴대폰 사용자나 상점 등이 될 수 있다. 이 플랫폼에서 제공하는 위치 기반 서비스를 이용할 때의 대표적인 시나리오는 아래와 같은 것이 있다. 휴대폰 사용자가 자신의 위치로부터 반경 1Km 내의 친구를 찾고 싶을 때, 해당 서비스를 제공하는 LSP에 요청을 보내고, 해당 LSP는 LBIB-LDS의 서비스를 이용하여 결과를 돌려준다.

### 2.2 시공간 모델링

<그림 2>에서처럼 GALIS에서 2차원 공간을 n개의 공간 영역으로 분할하고 1차원의 시간 축을 p개의 시간 영역으로 분할한다. 공간적인 관점에서 LBS 시스템에 의해 다루어지는 지리적 영역내의 한 영역을 macro-cell이라고 한다. 각 macro-cell은 한 변의 기본 길이가 25.6Km이고 그 배수로 증가할 수 있는 정사각형의 영역을 담당한다. 여러 개의 macro-cell들이 있을 때 그 크기는 각자 다를 수 있다. 각 macro-cell은 고정된 크기의 정사각형 cell로 균등하게 분할된다. 이 고정된 크기의 cell을 micro-cell이라 한다. 현재 micro-cell의 기본 크기는 100m X 100m로 설정되어 있다.

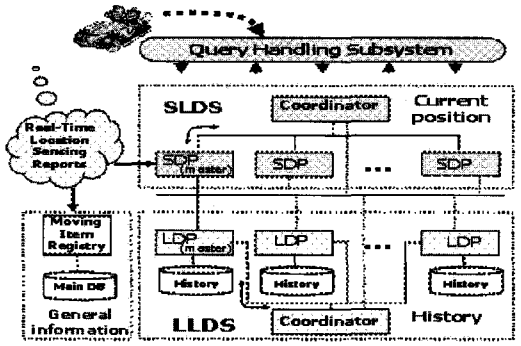



<그림 2> Spatio-Temporal modeling[7]

시간적인 관점에서, 최근의 과거 시간을 네 개의 시간 영역으로 분할한다. Time-zone 1 은 최근 한 시간 동안의 영역을 담당한다. 과거 기록 중 100m X 100m cell의 외부로 이동하는 기록만 Time-zone 1 데이타 서브 시스템에 유지된다. Time-zone 2는 현재부터 네 시간 전과 현재부터 한 시간 전 시점 사이의 영역을 담당한다. 여기서는 400m X 400m cell의 외부로 이동하는 기록만 Time-zone 2 데이타 서브시스템에 유지한다. 마찬가지로, Time-zone 3는 오늘 AM 0:00부터 Time-zone 2의 시작 시점까지의 시간 영역을 담당하고, 3200m X 3200m cell의 외부로 이동하는 기록만을 유지한다. 마지막으로 Time-zone 4는 LDS 시스템이 구동된 시점부터 Time-zone 3의 시작 시점까지의 시간 영역을 담당하고, 3200m X 3200m cell을 기준으로 삼는다. 이런 방식으로, 우리는 오래된 위치 기록일수록 자세한 세부 기록이 필요하지 않다는 관점을 반영하였다.

### 2.3 GALIS 구조

GALIS 구조는 위치 데이타 서버를 제안한 것으로, 다중 데이타 프로세서로 구성된 클러스터 기반 분산 컴퓨팅 구조로서 지역적인 분할과 time-zone을 통한 시간적인 분할을 통해 정보를 가지고 있는다. 아래의 그림 3은 GALIS 구조를 전체적으로 표현한 것이다. MIR(Moving Item Registry)은 메인 데이타베이스로 이동 객체의 프로필을 정보를 제공한다. GALIS 구조는 크게 SLDS와 LLDS로 구성되어 있다.




<그림 3> GALIS 구조[7]

SLDS(Short-term Location Data Subsystem)는 메인 메모리 데이타베이스를 이용하여 이동 객체들의 현재 위치 정보를 주관하게 된다. 또한 SLDS는 모든 이동객체의 현재 위치 정보를 빈번하게 갱신할 수 있어야 한다. SLDS내에서 각각의 분산 컴퓨팅 노드들은 SDP(Short-term Data Processor)라고 정의한다. SDP는 macro-cell로 명명된 일정 지역 내의 객체의 정보를 담당하며, 다수의 SDP는 다른 객체들의 현재 위치 정보를 갱신하기 위해 동시에 동작하게 된다. Coordinator node는 각 노드에 할당된 객체 수를 모니터하여 한 노드에 객체가 편중되지 않도록 노드들을 동적으로 분할 하거나 합병함으로써 노드간의 균형을 조절하게 한다.

SLDS와 비슷하게 LLDS(Long-term Location Data Subsystem)는 디스크 기반 데이타베이스 서버를 이용하는LDP(Long-term Data Processor)라고 불리는 분산 컴퓨팅 노드의 클러스터로 이루어져 있다. LLDS는 이동 객체의 과거 위치 정보를 경과된 시간대에 따라 네 개의 Time-zone에 나누어 유지하게 된다. 만약 한 객체가 100m x 100m의 micro-cell 내에서 이동을 했다면 SLDS에서는 현재 위치 정보를 갱신하는 반면 LLDS에서는 갱신하지 않는다. 이는 SLDS에 비해 LLDS에서의 위치 정보 갱신 빈도가 낮음을 의미한다.

## 3. TMO 기반 위치 데이타 서버 프로토타입

### 3.1 TMO 구조

다수의 SDP(또는 LDP)들의 분산 컴퓨팅 동작은 클러스터 기반의 분산 컴퓨팅 접근방식의 잠재적인 장점을 이끌어 내기 위해 잘 조화되어야 한다. 전역 시간 기반 코디네이션(global time based coordination)의 매우 경쟁력 있는 실행 잠재력에 대해 최근 몇 년간 분산 컴퓨팅 연구에서 의해 점점 더 많이 연구되고 있다[13]. 실시간 분산 객체 프로그래밍이 사용된 TMO 기법이 적용된다면 전역 시간 기반 코디네이션의 구현은 상당히 용이해진다. 현재 사용 가능한 TMO 프로그래밍 도구들은 일반적인 분산 컴퓨팅 프로그래밍에 사용하기에 가장 쉬운 도구이기도 하다. 그래서 GALIS의 각 서브시스템은 TMO network로 구성되어있다.

TMO는 잠금 가능한 데이타 멤버(C++ object의 데이타 멤버와 같은)들의 그룹으로 구성된 ODS(Object Data Store)를 포함한다. 각 잠금 가능한 그룹은 ODSS(ODS segment)라고 하며, TMO가 소멸될 때까지 존재한다. 또한 SvMs(Service Methods)라고 하는 클라이언트 TMO의 호출에 의해 구동되는 원격 메서드 호출 메카니즘을 제공한다. SvM의 실행은 동일 노드나 다른 노드에 위치할 수 있는 클라이언트 TMO로부터의 서비스 요청을 전달하는 메시지를 받음으로써 이루어진다. 클라이언트 TMO와 서버 TMO의 고도의 병행 수행을 용이하게 하기 위해 Non-Blocking 타임의 호출이 원격 SvMs으로 만들어질 수 있다.

위에 언급된 특성들은 지난 십년여간 산업계에서 기본 객체(C++과 Java program의 객체들과 같은)로

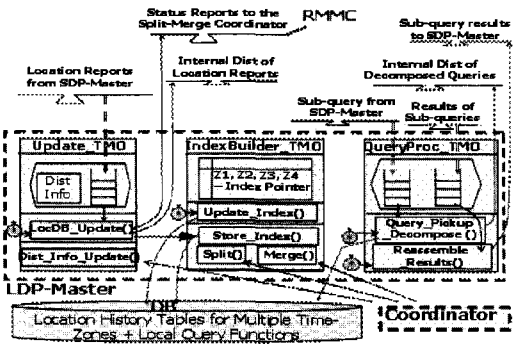
부터 확장된 모든 분산 컴퓨팅 객체 특성을 반영한 비교적 작은 변형이다. 추가로 TMO는 두 개의 독특한 확장을 보여준다. 하나는 클라이언트 TMO의 직접 호출에 의해 구동할 수 없는 대신 실제 시간에 미리 명시된 범위(예를 들어 "for t =from 10am to 10:50am every 30min start-during(t, t+5min)")에 들어왔을 때 구동되는 Time-Triggered Methods 라고도 불리는 SpMs(Spontaneous Methods)의 제공이다. 다른 하나는 어떤 ODSSs에 SpM 실행과 SvM 실행 시 동시에 접근하려 하여 충돌이 발생했을 때 SpM에게 먼저 ODSSs를 사용하는 권한을 주는 기본 병행 수행 제약(Basic Concurrency Constraint)이란 제약 조건을 두어 이를 방지하였다. 이 병행 수행 제약은 설계자에게 요구되는 설계 시점에서의 TMO의 시간적인 서비스 성능 보장에 대한 노력을 크게 줄여준다.

그리고 원격 메소드 호출에 의한 상호 작용 모드에 더하여, TMO는 포함된 객체의 데이타 멤버로 명시적으로 명시된 access gate인 논리적 메시지 채널을 통해 교환될 수 있는 다른 상호 작용 모드를 사용할 수 있다. TMO에서 도입한 발전된 형태의 이런 채널 구조를 RMMC(Real-time Multicast and Memory-replication Channel)라고 한다.

### 3.2 이동객체 위치정보 저장

Master SDP는 RMMC로 연결되어 있는 Update_TMO, MMDB_TMO 그리고 Query_Proc_TMO의 세 개의 TMO로 구성되어 있고, Slave SDP 역시 같은 구성으로 되어 있다. Simulator를 통해 전달 되어지는 위치정보는 RMMC를 통해 Update_TMO내의 큐에 쌓이게 되고, Update_TMO내에 있는 Location_DB_Update SpM은 버퍼에 쌓여있는 자료들을 주기적으로 가지고 온다. 이렇게 가지고 온 위치 데이타가 Master SDP에 담당하는 지역에 해당한다면 MMDB_TMO의 Update_SvM을 이용해 MMDB_TMO로 전달하고 메인 메모리 데이타베이스에 저장하게 된다. Master SDP가 담당하는 지역에 해당하지 않는 데이타들은 Location_DB_Update SpM가 또 다른 RMMC를 통해 모든 Worker SDP로 자료를 전송하게 된다. 이렇게 전송된 데이타들은 Worker SDP내의 Update_TMO의 큐에 쌓이게 되고, Worker SDP의 TMO들은 도착한 이동객체의 정보가 자신의 영역에 해당하고 있는 정보만을 필터링하여 메인 메모리 데이타베이스에 저장하고, 영역 밖의 데이타는 버리게 된다.

Mater LDP는 아래의 그림 4처럼 RMMC로 연결되어 있는 세 개의 Update_TMO, IndexBuild_TMO, 그리고 Query_Proc_TMO로 구성되어 있다.




<그림 4> TMO 네트워크로 구성된 LDP Master

Master LDP는 micro-cell의 100m x 100m를 벗어난 데이타에 대한 위치 정보만을 SDP Master로부터 전달 받아 Update_TMO의 버퍼에 쌓아 놓게 된다. 이렇게 쌓여진 정보는 Update_TMO의 Location_DB_Update SpM이 주기적으로 버퍼에 접근하여 가지고 오게 된다. 만약 이렇게 접근하여 가지고 온 위치 정보가 Master LDP의 지역에 해당하는 위치 정보라면 IndexBuilder_TMO 내에 있는 Store_Index SvM의 동작에 의해 상용 데이타베이스 시스템에 위치 자료의 시간 값을 포함하는 균등 그리드 형태로 표현되는 과거 위치정보 테이블에 저장 된다. 만약 Master LDP가 담당하지 않는 지역에 해당하는 데이타들은 Update_TMO에 있는 Location_DB_Update SpM에 의해 모든 worker LDP로 또 다른 RMMC를 통해 위치 정보를 전송하게 된다. 이렇게 전송된 데이타들은 Worker LDP내의 Update_TMO의 큐에 쌓이게 되고, Worker LDP의 TMO들은 도착한 이동객체

의 정보가 자신의 영역에 해당하는 데이타들은 IndexBuild_TMO의 Stored_Index SvM을 통해 과거 위치정보를 저장하는 테이블에 저장된다.

### 3.3 질의 처리 과정

Query_Proc_TMO는 사용자로부터 질의를 입력받고, 각 질의를 수행한 후 결과를 사용자에게 전달하는 역할을 한다. Query_Proc_TMO의 Query_Pickup_Decompose SpM은 주기적으로 버퍼에 쌓여있는 질의를 하나씩 가지고 온다. 이렇게 가지고 온 질의는 Query_Pickup_Decompose SpM이 분해하고 분석하여 worker SDP에 해당하는 부분 질의 형태로 분산시켜 주며, 과거의 위치와 관련된 질의는 Master LDP로 질의를 전송하는 역할을 해준다. Master SDP나 Worker SDP가 수행한 질의의 결과는 Query_Proc_TMO가 접근 가능한 Master SDP 내의 버퍼로 전송되는데, Query_Proc_TMO의 Reassemble_Result SpM는 주기적으로 버퍼에 있는 결과를 꺼내서 종합하여 질의를 한 사용자에게 다시 전달을 해주게 된다.

Master SDP로부터 RMMC를 통해 전달된 과거와 관련된 부분 질의는 Master LDP의 Query_Proc_TMO의 버퍼에 쌓이게 된다. 이렇게 버퍼에 쌓인 질의는 Query_Proc_TMO의 Query_Pickup_Decompose SpM가 분석하고 worker LDP로 전송 되야 하는 질의는 부분 질의 형태로 분산되어진다. Worker LDP에서 수행된 부분 질의는 Master LDP내에 있는 또 다른 버퍼로 보내 Master LDP가 모으게 된다. Query_Proc_TMO의 Reassemble_Result SpM은 버퍼에 수집된 결과를 종합하여 RMMC를 통해 Master SDP로 전송한다.

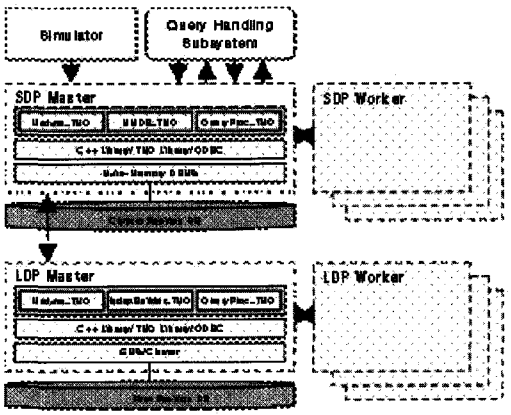
## 4 구현 및 성능평가

### 4.1 구현 및 실험 환경

프로토타입의 구현은 4개의 노드로 구성되어 있고, 각각의 노드를 구성하는 컴퓨터는 P4 2.4GHz CPU, 1Gbyte RAM, MS Windows 2003 Server OS 그리고 DBMS는 GMS와 Altibase를 이용하여 구성하였다.

시스템 개발도구는 Microsoft사의 Visual Studio .Net 의 Visual C++ 7.0을 이용해 개발하였다. 위치 정보를 저장할 데이타베이스 엔진으로는 Altibase사의 Altibase와 지오매니아사의 GMS를 사용하였다. Altibase와 연동을 위해선 ODBC API를 이용하였고, GMS와의 연동은 GMS에서 제공하는 CLI를 이용해 연동하였다.

<그림 5> 에서와 같이 SLDS는 1개의 Master SDP와 3개의 Worker SDP로 구성되고, 위치 정보의 저장은 메인 메모리기반 데이타베이스인 Altibase에 저장한다. 또한 LLDS는 1개의 Master LDP와 3개의 Worker LDP로 구성되며 위치 정보의 저장은 디스크 기반 데이타베이스인 GMS를 이용해 저장한다. 각 노드의 구성은 분할된 지역을 담당하는 하나의 SDP와 하나의 LDP로 구성된다. 또한 이동객체의 움직임을 별도의 인터페이스를 통해 모니터링 할 수 있다.



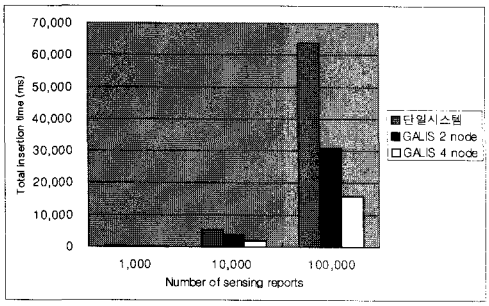

<그림 5> 실험 환경

### 4.2 이동객체 위치 정보 생성 및 저장

이동객체의 위치 정보 생성은 Simulator를 통해 전체 지역에 고르게 분포한 데이타를 생성하게 하였고, 이동객체의 움직임은 특정 지역 밖을 벗어나지 않는 범위 내에서 랜덤하게 움직이게 하였다. 이렇게 생성되어 움직이는 이동객체의 위치 데이타는 5초단위로 GALIS에 계속적으로 알리게 된다.

TMO 프로그래밍 기법을 통해 구성된 GALIS 아키텍처의 데이타 분산처리 능력을 평가하기 위해 시스템 구성을 분산의 기능이 없는 단일 시스템, GALIS 아키텍처를 적용하여 분산 처리하는 2 node 와 4 node system으로 구성하였고, 데이타 수를 증가시켜 처리 시간을 측정하였다.

실험 결과 GALIS 아키텍처를 적용한 분산처리 시스템이 성능 면에서 앞서는 것을 그림 6을 통해 살펴볼 수 있다. 분산기능이 없는 단일 시스템에서의 데이타 처리 시간은 2 node로 구성된 GALIS 시스템 보다 약 2배 이상의 처리 시간을 소요하는데, 시뮬레이터에서의 위치 정보가 전체 영역에서 고르게 분포하는 상황으로 발생시키기 때문에 단일 시스템에서 처리해야 하는 데이타의 수가 2 노드로 구성된 GALIS 시스템보다 2배 이상의 데이타를 처리하기 때문이다. 만약 데이타가 특정 노드로 집중하는 상황이 발생하면 아래의 그림과 같은 결과를 기대 할 수 없다. 하지만 GALIS 아키텍처의 Coordinator에서 데이타 집중현상이 발생하는 노드를 동적으로 분할하여, 처리해야 하는 데이타를 줄여주기 때문에 분산이 없는 단일 시스템과 같은 성능저하 현상을 피할 수 있다.




<그림 6> 위치 정보 처리 시간

만약 대량의 이동 객체 위치 데이타가 계속적으로 입력이 되고 있을 때 질의가 발생할 경우, 단일 시스템에서는 대용량의 위치 데이타를 입력에 따른 입력 지연 시간이 발생하게 되고 질의에 대한 응답은 최신 데이타를 반영하지 못하는 결과를 가져오게 된다. 본 논문에서 구현한 GALIS 아키텍처는 이러한 문제점을 줄여줄 수 있다.

GALIS 아키텍처에서 데이타를 처리하는데 있어서 또 다른 장점은 시공간 모델링을 통해 데이타의 수를 줄여줄 수 있다는 것이다. 아래의 표 1은 300개의 이동객체가 움직이며 시간 경과에 따라 셀 필터링을 통해 SLDS에서 LLDS로 전송되는 객체의 수를 표로 나타낸 것이다

<표 1> 필터링된 데이타 수

| | 100m |
|---|---|
| 1분 | 246 |
| 5분 | 246 |
| 10분 | 244 |

<표 1> 에서 나타난 것처럼 전송되는 객체의 수는 줄어들고 LLDS에 누적되는 데이타의 양이 적어져 데이타 처리시간 및 저장 공간에 활용에 있어 성능향상이 일어날 수 있다. 이는 LLDS에서 제공하는 Time-zone 이동에 따른 필터링에서도 비슷한 성능향상을 제공한다. 하지만 세부 데이타의 손실이 발생하여 저장된 위치정보의 정확성이 떨어질 수 있는 단점이 발생하고, 이러한 필터링은 GALIS가 적용되어야 하는 분야의 요구사항에 따라 시공간 모델링의 세부조건을 유연하게 적용하면 문제를 해결할 수 있다.

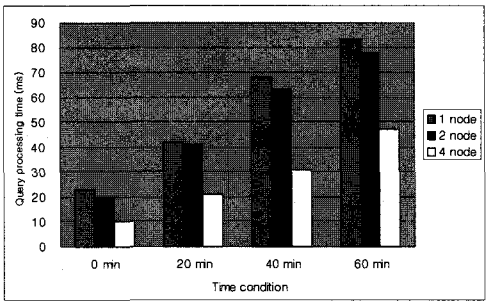
### 4.3 질의 처리 시간

질의를 처리하는 시간은 노드들 간의 전송시간과 전달 받은 질의를 유형에 따라 Altibase 혹은 GMS에서 처리하는 시간, 그리고 이 결과를 전송하는 시간으로 구성된다. 질의가 노드들에 전송되는 위치 데이타를 전송하는 시간에 비해 매우 짧은 시간에 이루어지는 반면에, 검색된 결과는 데이타의 수에 따라 전송시간이 결정되기 때문에 질의처리 시간의 결정은 질의 결과 데이타 집합의 크기에 따라 결정된다.

정확한 질의 성능평가를 위해 샘플 데이타를 구성하였다. 샘플 데이타는 이동 객체 800개가 5초 간격으로 한 시간 동안 발생시킨 좌표로 구성된 데이타며 이때 총 발생한 데이타의 수는 576,000개이다. 또한 같은

질의를 3회 하여 결과값의 평균을 내었다. 이때 이동 객체를 저장하는 테이블의 인덱스는 객체를 식별하는 ID와 시간으로 정해 주었다. 본 논문에서 평가한 질의 처리 성능평가는 아이템 관련 질의와 영역질의 두 가지 질의 유형을 테스트하였다.

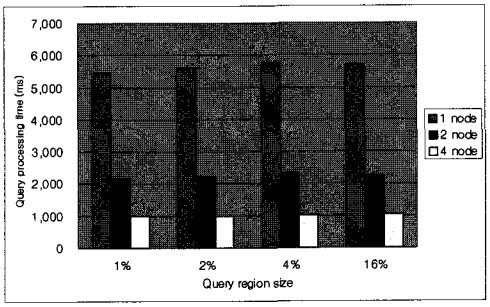
<그림 7>은 노드 수에 따른 아이템 관련 질의이며, 특정 아이디의 위치정보를 시간조건을 다르게 하여 질의하였을 때 그 처리 시간을 그래프로 나타낸 것이다.




<그림 7> Temporal query processing time

시스템이 하나의 노드 보다는 네 개의 노드로 구성되었을 때 보다 빠른 검색을 하며, 하나의 노드와 두 개의 노드로 구성된 시스템간의 결과가 크게 차이가 나지 않는 이유는 질의 조건이 인덱스가 설정되어 있는 상황이기 때문이라고 생각되고, 만약 샘플 데이타의 수가 많아지게 되면, 하나의 노드 시스템과 두 개의 노드 시스템간의 차이도 확실하게 보여질 수 있다.

<그림 8>은 사각형 형태인 범위 질의를 하였을 때 소요되는 질의처리 시간을 보여주고 있다. 샘플 데이타의 구성은 위의 실험 시 구성한 데이타와 동일하며, 범위 질의의 형태는 전체 영역의 1%, 2%, 4%, 16%를 검색하는 질의이다. 시스템의 구성 그리고 영역의 범위와 상관 없이 3가지 시스템 모두 비슷한 결과값을 보이는데, 이는 영역질의는 데이타베이스 전체를 검색해야 하기 때문에 이런 결과가 나타났다. 하지만 3가지 시스템 모두 네 개의 노드일 때 가장 좋은 성능을 보이는데, 이는 전체를 검색해야 하는 데이타베이스 사이즈가 작기 때문에 생긴 결과이다.




<그림 8> Spatial query processing time

아이템 관련 질의 및 범위 관련 질의를 처리하는 경우를 테스트하여 본 결과 노드가 늘어날 경우 검색해야 하는 데이타베이스의 크기가 줄어들기 때문에 보다 효율적으로 질의를 수행 할 수 있는 것을 살펴 볼 수 있다.

## 5. 결론 및 향후 연구

본 논문에서는 대용량의 현재 위치와 과거 정보를 다루기 위한 GALIS 아키텍처라는 실시간 분산 위치 데이타 서버를 제안하였다. GALIS의 구조는 다수의 프로세서로 구성된 클러스터 기반 분산 컴퓨팅 시스템이다. 이 구조는 프로세서를 지역적으로 분할하여 해당 지역을 담당하는 구조로 분할하였고, Time-zone이라는 발생한 위치정보를 시간적으로 분할하여 위치정보를 관리하도록 구성되어 있다. 이러한 GALIS 구조는 계속적으로 발생되는 이동객체의 위치 데이타를 효율적인 방법으로 지속적인 관리가 가능하게 하였고, 이때 클라이언트로부터 발생되는 여러 형태의 질의를 동시에 수행할 수 있는 구조로 설계 되었다. GALIS 구조에 적용된 여러 가지 기법은 하나의 노드에서 처리해야 하는 정보를 줄여주는 역할을 하여 노드에 걸리는 여러 형태의 부하를 줄이는 장점이 있다.

GALIS는 TMO 프로그래밍 기법을 적용하여 여러 프로세서들간의 연결과 통신이 자유롭게 하였으며 효율적이고 유연한 확장 능력을 가지게 되었다. 또한 분산 처리환경을 쉽고 효율적으로 구축할 수 있어 프로그래머들의 노력과 비용을 줄여주는 장점이 있다. 그

리고 공간 데이타베이스 엔진인 GMS를 이용하여 MS_SQL과 같은 관계 데이타베이스 엔진을 사용할 때 보다 효율적인 저장과 GMS에서 제공하는 공간질의 연산을 이용할 수 있어 보다 효율적으로 질의를 처리할 수 있었다.

본 논문에서 평가한 실험들을 통해서 GALIS에서 제공하는 시공간 모델링 기법이나 분산 컴퓨팅 환경을 통해 지속적으로 처리해야 하는 대용량 데이타를 보다 효율적으로 처리할 수 있음을 살펴볼 수 있었다.

향후 연구로는 GALIS에서 제시한 시공간 모델링을 통해 발생하는 몇 가지 단점을 해결하는 방법이나 이동객체의 위치정보를 발생하는 시뮬레이터의 위치정보가 보다 실 세계의 특성이 더 반영된 데이타를 만들어내는 분야의 연구와 시공간 질의를 효율적으로 처리할 수 있는 질의처리 알고리즘의 분야의 연구가 필요하다.

## 참고문헌


[1] Xu, X., Han, J., and Lu, W., "RT-tree: An Improved R-tree Index Structure for Spatiotemporal Databases", Proc. 4th Int'l Symp. on Spatial Data Handling(SDH), 1990.

[2] Theodoridis, Y., Vazirgiannis, M., and Sellis, T., "Spatio-Temporal Indexing for Large Multimedia Applications", Proc. 3rd IEEE Conf. on Multimedia Computing and Systems(ICMCS), 1996.

[3] Nascimento, M.A..,and Silva, J.R.O., "Towards Historical R-Trees", Proc. ACM Symp. on Applied Computing (ACM-SAC), 1998.

[4] Pfoser, D. Jensen, C.S., and Theodoridis, Y., "Novel Approaches to the Indexing of Moving Object Trajectories, " Proc.VLDB 2000, pp.395-406.

[5] Song, Z. and Roussopoulos, N., "Hashing Moving Objects," Proc. Int'l Conf. on Mobile Data Management (MDM), 2001.

[6] Saltenis, S., et al, Jensen, C., Leutenegger, S., Lopez, M., "Indexing the Positions of Continuously Moving Objects," Proc. ACM SIGMOD, 2000.

[7] Nah, Y., Wang, T., Kim, K.H., Kim. M.H., and Yang, Y.K., "TMO-structured Cluster-based Real-time Management of Location Data on Massive Volume of Moving Items," in Proc. STFES 2003, IEEE Press, Hakodate, Japan, May 2003, pp.89-92.

[8] Nah, Y., Kim, K.H., Wang, T., Kim, M.H., Lee, J., and Yang, Y.K., "A Cluster-based TMO-structured Scalable Approach for Location Information Systems," in Proc. WORDS 2003 Fall, IEEE CS Press, Capri Island, Italy, October 2003, pp.225-233.

[9] Kim, M.H., Kim, K.H., Nah, Y., Lee, J., Wang, T., Lee, J. and Yang, Y.K., "Distributed Adaptive Architecture for Managing Very Volume of Moving Objects," in Proc. 7th World Conference on Integrated Design and Process Technology (IDPT), Austin, Texas, December 2003, pp.737-744.

[10] Kim, K.H., "Object Structures for Real-Time Systems and Simulators", IEEE Computer, Aug. 1997, pp.62-70.

[11] Kim, K.H., "APIs for Real-Time Distributed Object Programming", IEEE Computer, June 2000, pp.72-80.

[12] Kim, K.H., "Commanding and Reactice Control of Peripherals in the TMO Programming Scheme", Proc.
ISORC 2002 (5th IEEE CS Int'l Symp. on OO Real-time distributed Computing), Crystal City, VA, April 2002, pp.448-456.

[13] H. Kopetz, Real-Time Systems - Design Principles for Distributed Embedded Applications, Kluwer
Academic Publishers, Chap. 3, pp.45-70, 1997.

이준우

2000년 단국대학교 생물학과(이학사)
2003년 단국대학교 대학원 컴퓨터공학과 (공학석사)
2003년 ~ 현재 단국대학교 대학원 컴퓨터공학과 박사과정
관심분야 : 이동객체 데이터베이스, GIS, 분산컴퓨팅, 바이오 인포매틱스

이운주

2002년 단국대학교 컴퓨터공학과 (공학사)
2005년 단국대학교 컴퓨터공학과 (공학석사)
관심분야 : 이동객체 데이타베이스, LBS, GIS

이호

2004년 단국대학교 컴퓨터공학과 (공학사)
2004년 ~ 현재 단국대학교 컴퓨터공학과(석사과정)
관심분야 : 이동객체 데이터베이스, 공간데이타베이스, XML

나연묵

1986년 서울대학교 컴퓨터공학과 (공학사)
1988년 서울대학교 대학원 컴퓨터공학과(공학석사)
1993년 서울대학교 대학원 컴퓨터공학과(공학박사)
1991년 IBM T. J. Watson 연구소 객원연구원
2001년 ~ 2002년 University of California, Irvine 객원교수
1993년 ~ 현재 단국대학교 전기전자컴퓨터공학부 전기전자컴퓨터공학 전공 부교수
관심분야 : 데이타베이스, 데이타 모델링, 객체지향 데이타베이스, 멀티미디어 데이타베이스, 멀티미디어 정보 검색