실시간 위치 기반 서비스를 위한 확장 SLDS 설계 및 구현†

Design and Implementation of the Extended SLDS for Real-time Location Based Services

이승원*, 강홍구*, 홍동숙*, 한기준*

Seung-Won Lee, Hong-Koo Kang, Dong-Suk Hong, Ki-Joon Han

요 약 최근 이동 컴퓨팅 기술과 무선 측위 기술이 급속도로 발전하고 무선 인터넷이 보편화 됨에 따라 이동체의 위치 정보를 활용하는 위치 기반 서비스(LBS: Location Based Service)가 다양한 분야에서 제공되고 있다. 위치 기반 서비스를 제공하기 위해서는 이동체의 위치 정보를 주기적으로 저장하는 위치 데이타 서버가 필요하다. 기존에는 이동체의 위치 데이타를 저장하기 위해서 GIS 서버를 사용해 왔다. 하지만 GIS 서버는 정적 데이타를 기반으로 설계되었기 때문에 이동체의 위치 데이타를 저장하는데 적합하지 않다.

따라서, 본 논문에서는 이동체의 위치 데이타를 관리하기 위해 제안된 클러스터 기반 분산 컴퓨팅 구조를 갖는 GALIS(Gracefully Aging Location Information System) 아키텍처의 서브 시스템인 SLDS(Short-term Location Data Subsystem)를 확장하여 실시간 위치 기반 서비스를 위한 확장 SLDS를 설계 및 구현하였다. 확장 SLDS는 TMO(Time-triggered Message-triggered Object) 프로그래밍 기법을 이용하여 위치 데이타 처리의 실시간성을 보장하며, 이동체 데이타를 다수의 노드에 적절히 분산시킴으로써 대용량 위치 데이타를 효율적으로 관리할 수 있다. 그리고, 메인 메모리에서 위치 데이타를 관리하기 때문에 검색 및 갱신 오버헤드가 적다. 또한, 실험을 통하여 확장 SLDS는 기존에 제시된 SLDS 보다 더 효율적인 저장과 부하 분산을 수행한다는 것을 확인하였다.

Abstract Recently, with the rapid development of mobile computing, wireless positioning technologies, and the generalization of wireless internet, LBS(Location Based Service) which utilizes location information of moving objects is serving in many fields. In order to serve LBS efficiently, the location data server that periodically stores location data of moving objects is required. Formerly, GIS servers have been used to store location data of moving objects. However, GIS servers are not suitable to store location data of moving objects because it was designed to store static data.

Therefore, in this paper, we designed and implemented an extended SLDS(Short-term Location Data Subsystem) for real-time Location Based Services. The extended SLDS is extended from the SLDS which is a subsystem of the GALIS(Gracefully Aging Location Information System) architecture that was proposed as a cluster-based distributed computing system architecture for managing location data of moving objects. The extended SLDS guarantees real-time service capabilities using the TMO(Time-triggered Message-triggered Object) programming scheme and efficiently manages large volume of location data through distributing moving object data over multiple nodes. The extended SLDS also has a little search and update overhead because of managing location data in main memory. In addition, we proved that the extended SLDS stores location data and performs load distribution more efficiently than the original SLDS through the performance evaluation.

주요어 : 이동체, 위치 기반 서비스, GALIS, SLDS, TMO

Keywords : moving objects, LBS, GALIS, SLDS, TMO

† 본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터 육성.지원사업의 연구결과로 수행되었음.

* 건국대학교 컴퓨터공학부 {swlee, hkkang, dshong, kjhan}@db.konkuk.ac.kr

1. 서 론

최근에 이동 컴퓨팅 기술이 발전하고, 휴대 전화, PDA와 같은 이동 단말기가 보편화됨에 따라 이동체의 위치 데이타를 이용한 위치 추적 서비스, 교통 정보 서비스, 모바일 광고 서비스, 긴급 구조 서비스 등과 같은 위치 기반 서비스에 대한 수요가 급증하고 있다[1],[2]. 그리고, 다양한 위치 기반 서비스를 제공하기 위해서는 이동체의 위치 데이타를 실시간으로 저장하고 검색할 수 있는 위치 데이타 서버가 필수적으로 요구된다[1],[3],[4].

이동체의 위치 데이타는 이동체의 수가 많고 위치 획득 간격이 짧을수록 대용량이 되기 때문에 이동체의 위치 데이타 서버는 대용량의 데이타를 효율적으로 관리할 수 있어야 한다[5]. 또한, 이동체는 시간에 따라 위치가 계속 변경되는 시공간 객체이기 때문에 위치 데이타 서버는 연속적인 갱신 작업을 효율적으로 수행할 수 있어야 한다. 하지만, 위치 데이타 서버로써 주로 사용되는 GIS 서버는 이러한 두 가지 요구사항을 만족시키지 못한다. 대부분의 GIS 서버는 하나의 노드로 구성되기 때문에 대용량 데이타를 효율적으로 관리하기가 어렵고, 정적 데이타를 대상으로 하였기 때문에 연속적으로 위치를 변경하는 이동체를 관리하는데 적합하지 않다[6]. 이러한 문제를 해결하기 위해서 이동체 데이타를 위한 시스템 아키텍처인 GALIS가 제안되었다[7],[8],[9].

GALIS는 TMO[10] 프로그래밍 기법을 이용하여 실시간 처리를 지원할 수 있도록 하였으며, 다수의 노드로 구성된 분산 컴퓨팅 구조로 이루어져 있다. GALIS는 현재 위치 데이타를 처리하는 SLDS와 과거 위치 데이타를 처리하는 LLDS(Long-term Location Data Subsystem)로 구성되어있다. SLDS는 여러 개의 SDP(Short-term Data Processor)라고 불리우는 노드로 구성될 수 있으며, 이러한 분산 구조는 각 지역별 데이타를 여러 노드에 저장함으로써 위치 정보의 저장과 갱신 및 검색 질의에 대한 부하를 분산시키고 대용량의 데이타를 효율적으로 관리할 수 있다.

GALIS에서는 노드의 분할 및 합병을 위한 정책을 정의하였는데, SLDS는 GALIS의 분할 및 합병 정책에 따라 노드의 부하 분산을 수행한다. 그러나 SLDS에서 사용된 노드의 분할 및 합병 정책은 노드 경계를 이동하는 이동체가 많을수록 분할 및 합병의 횟수가 증가할 수 있다는 문제점을 가지고 있다. 이러한 문제는 물리적으로 분산되어 있는 SLDS에서 심각한 오버헤드를 초래한다. 또한, GALIS에서는 이동체의 위치 데이타를 색인하기 위해서 서비스 영역을 작은 Cell들로 분할하여 처리하도록 하고 있으며, SLDS는 이러한 색인 방법을 따른다. Cell을 이용하기 위해서는 Cell ID 결정 알고리즘이 필요하며, Cell ID 결정을 위해서 GALIS에서는 z-ordering 기법을 이용하는 것을 제안하였다[8],[9]. 그러나 Cell을 이용하는 것은 검색 시에는 유리하나 저장 시에는 Cell ID 결정을 위해 추가적인 연산 비용이 든다는 문제점을 가지고 있다.

지금까지 설명한 것과 같이 SLDS의 노드 분할 및 합병 정책은 많은 노드 분할 및 합병을 초래하며, Cell ID 결정을 위한 z-ordering 기법은 저장 시 추가적인 연산 비용이 든다는 문제점을 가지고 있다. 이러한 문제점을 해결하기 위해 본 논문에서는 기존의 SLDS 구조에 개선된 분할 및 합병 정책과 Cell ID 결정 알고리즘을 추가하여 확장 SLDS를 설계 및 구현하였으며, 또한 성능 평가를 수행하여 확장 SLDS가 기존의 SLDS 보다 더 우수하다는 것을 입증하였다.

본 논문의 구성은 다음과 같다. 제 2장에서는 관련 연구로 GALIS와 TMO프로그래밍 기법에 대해 살펴본다. 제 3장에서는 시스템의 설계 및 구현에 대해 설명하고, 제 4장에서는 성능 평가 결과를 분석한다. 마지막으로, 제 5장에서는 결론과 향후 연구과제를 언급한다.

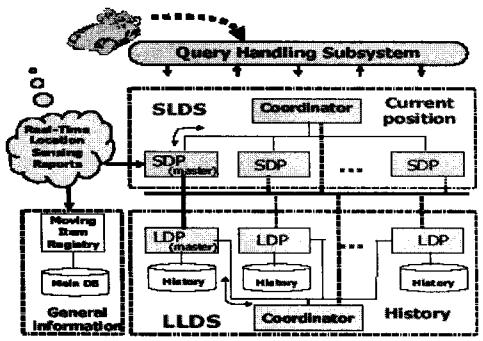
2. 관련 연구

본 장에서는 이동체의 위치 데이타를 관리하기 위해서 제안된 GALIS 아키텍처와 실시간성을 지원하는 확장 SLDS를 구현하기 위해 사용된 TMO 프로그래밍 기법에 대해서 설명한다.

2.1 GALIS 아키텍처

이동체의 위치 데이타를 관리하기 위해 제안된 GALIS는 다중 데이타 프로세서로 구성된 클러스터 기반 분산 컴퓨팅 구조로서 각 프로세서가 서로 다른 특정 영역 내에 있는 이동체의 움직임에 대한 처리를 수행한다. 위치 기반 서비스를 위해서 관리되어야 하는 전체 공간을 2차원 평면으로 볼 때, GALIS에서는 이 2차원 평면을 n개의 영역으로 나누며, 나뉘어진 영

역을 Macro-cell이라고 부른다. 각각의 Macro-cell 영역에 포함되는 이동체에 대한 위치 데이타는 하나의 데이타 프로세서가 관리하며, Macro-cell은 다시 100m×100m의 일정한 간격을 가진 Micro-cell이라는 영역으로 나뉘어진다. Micro-cell은 하나의 데이타 프로세서에서 이동체의 현재 위치 데이타를 색인하기 위해서 구성되며, z-ordering 기법을 이용하여 Cell ID를 계산한다. 또한, 이동체의 과거 데이타를 저장하는 경우에 있어서는 Time-zones 이라는 시간 구역을 두어 오래된 데이타일수록 이동체의 위치정보에 대한 정확도를 낮게 설정하는 방법을 채택하였다[7],[8],[11].



<그림 1> GALIS 전체 구조

GALIS의 전체 구조는 <그림 1>과 같으며, 현재 위치 데이타를 저장하기 위한 SLDS와 과거 위치 데이타를 저장하기 위한 LLDS로 구성된다. SLDS는 현재 위치 데이타의 주기적인 갱신을 효율적으로 수행하기 위해서 메인 메모리 기반 데이타베이스 관리시스템(MMDBMS: Main Memory Database Management System)을 사용하며 LLDS는 디스크 기반 데이타베이스 관리시스템을 사용한다.

SLDS 내부의 각 노드를 SDP라고 한다. 각각의 SDP는 하나의 Macro-cell의 영역 내에 포함된 이동체의 현재 위치 데이타를 관리한다. SDP 중 하나의 노드를 SDP Master라 하며, 다른 SDP들은 SDP Worker라 부른다. LLDS도 SLDS와 비슷하게 구성되며 LDP(Long - term Data Processor) Master와 LDP Worker로 이루어져있다. SLDS와 LLDS는 SDP와 LDP외에 부하 분산을 위한 Coordinator도 가지고 있다.

Coordinator는 분할 및 합병 정책에 따라 SDP 및 LDP를 분할하거나 합병함으로써 부하 분산을 수행한다[7],[9]. GALIS 아키텍처에서 제시된 SLDS의 분할 정책은 SDP가 담당하는 이동체 수가 미리 정해진 최대 이동체 수를 넘는 경우 두 개의 노드로 분할한다. 분할 축은 x축과 y축을 반복적으로 변경하면서 중간 지점을 기준으로 두 개로 분할을 수행하게 된다. 또한, SDP에서 처리하는 이동체 수가 최소 이동체 수 보다 적은 경우 두 개의 노드를 하나로 합병하고 있다.

2.2 TMO 프로그래밍 기법

TMO 프로그래밍 기법은 기존 객체 모델을 실시간 분산 컴퓨팅을 위해 확장한 모델이다. TMO는 프로그래머들이 약간의 노력만을 가지고 분산 실시간 응용을 설계하고 프로그래밍 할 수 있도록 하기 위하여 개발되었다[10]. TMO의 구성요소 및 주요 특징을 기술하면 다음과 같다.

첫째, 네트워크상의 다른 TMO 객체에 접근하는 방법을 제공하기 위해서 EAC(Environment Access Capability)를 제공하며 Gate와 RMMC (Real-time Multicast and Memory-Replication Channel)로 구성되어 있다. Gate는 원격 메소드 호출 형태의 메시지 전송 기능을 제공하며, RMMC는 메모리 복제 기반의 메시지 전송 기능을 제공한다. 둘째, ODS(Object Data Store)는 논리적인 데이타의 집합으로 이루어진 객체인 ODSS(Object Data Store Segment)들을 저장하는 공간이다. 이것은 하나의 TMO 객체 내에서 정의된 메소드들 간의 데이타를 공유하기 위해서 사용된다.

셋째, 다중 노드 상에 분산되어 있는TMO는 Gate를 이용하여 다른 TMO의 서비스 메소드(SvM : time-triggered Service Method)를 수행할 수 있다. 넷 째, TMO는 서비스 메소드 이외에도 새로운 유형의 시간구동 메소드(SpM : time-triggered Sontaneous Method)를 제공한다. SpM은 주기성을 띄거나 시간성을 갖는 서비스를 수행하기 위한 메소드이며, TMO에서 시간 속성을 정의하기 위한 명세인 AAC(Autonomous Activation Condition)에 의해서 실행된다.

다섯째, BCC(Basic Concurrency Constraints)라는 수행 규칙을 통해 적시에 SpM이 수행되는 것을 보장할 수 있다. 마지막으로, TMO는 메소드의 실행 및 출력 행위에 대해서 설계자가 시작 시각과 종료 시각을 줄 수 있다.

이러한 특징들은 실시간 분산 컴퓨팅을 위하여 필요한 기능 이므로, 프로그래머는 이러한 특징들을 활용하여 보다 쉽게 실시간 분산 응용을 개발할 수 있

다. 이러한 TMO 프로그래밍 기법의 특징들은 위치 데이타 처리의 실시간성을 요구하는 GALIS 아키텍처에 적용이 용이하며, 또한, 클러스터 기반의 분산 컴퓨팅 구조를 구성하는데 도움이 되기 때문에 GALIS 아키텍처에서 도입되었다.

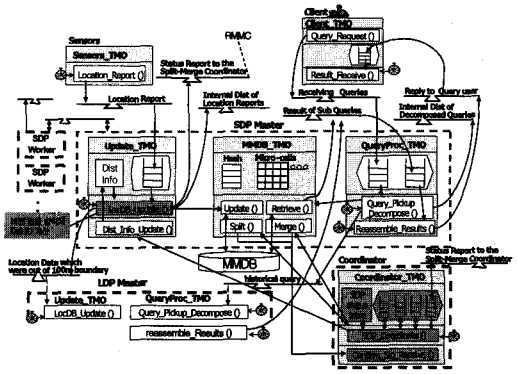
3. 시스템 설계 및 구현

본 장에서는 GALIS 아키텍처의 SLDS와 이것을 기반으로한 확장 SLDS의 구조에 관하여 비교 분석하고, 확장 SLDS의 핵심기능 세 가지를 설명한다. 확장 SLDS는 이동체를 시뮬레이션 하는 센서로부터 위치 데이타를 수신하여 저장하는 기능, 클라이언트의 질의를 처리하는 기능, 그리고 노드들의 부하 분산 기능을 가지고 있다.

3.1 확장 SLDS의 구조

본 논문에서 설계 및 구현한 확장 SLDS는 GALIS 아키텍처 중 현재 위치 데이타에 관한 처리를 담당하는 서브 시스템인 SLDS의 구조[8],[9]에 기반을 두고 있다. 확장 SLDS는 하나의 SDP Master와 다수의 SDP Worker 그리고 부하 분산 기능을 담당하는 Coordinator로 구성되어있다. SDP들은 서로 다른 영역에 대한 위치 데이타의 저장 및 검색을 책임지며, 세 개의 TMO 객체들(Update_TMO, MMDB_TMO, QueryProc_TMO)로 구성되어 있다. SLDS의 TMO 객체들은 서로 RMMC 채널을 이용하여 통신한다. SDP Master는 위치 데이타의 저장 및 검색 이외에 센서로부터 위치 데이타를 보고 받아 SDP Worker들에게 전송하는 역할, 클라이언트의 질의 수신과 SDP Worker들에게 전송하는 역할, 그리고 마지막으로 SDP Worker들로부터 수신된 질의 처리결과를 통합하는 역할을 갖고 있다. 이동체의 현재 위치는 매우 빈발적으로 갱신되기 때문에 SDP의 저장 장소로 MMDBMS를 사용하였다. 본 논문에서 설계한 확장 SLDS 구조는 <그림 2>와 같다.

확장 SLDS에는 Coordinator 모듈이 개선되었으며, 위치 데이타 저장 시의 오버헤드를 줄이기 위해 Update_TMO 객체의 LocDB_Update SpM에 비트 연결 방식이라는 알고리즘을 적용하였다. Coordinator 모듈은 분할되어야하는 SDP를 저장하는 분할 큐, 합병되어야하는 SDP를 저장하는 합병 큐, 스페어 노드를 저장하는 스페어 노드 큐, SDP들의 상태 정보를 임시로 저장하는 SDP 상태 큐를 가지고 있으며, 또한 SDP들의 상태 정보를 관리하기 위해 SDP 상태 테이블을 유지한다. 그리고 SDP들에게 분할 또는 합병 명령을 전달하기 위해 SDP_Coordinate SpM을 가지고 있으며, SDP들의 분할 또는 합병의 수행 결과를 보고받기 위해 Confirm_Sig_Receive SvM을 가지고 있다. 본 논문에서는 기존의 SLDS에서 사용된 분할 및 합병 알고리즘을 개선하여 더 효율적인 부하 분산을 가능하게 하였다. 지금까지 기술한 확장 SLDS의 비트 연결 방식과 Coordinator 모듈은 각각 3.2절과 3.4절에서 상세히 설명된다.



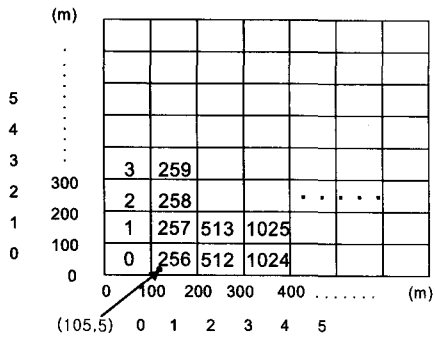
<그림 2> 확장 SLDS 구조

3.2 위치 보고 처리

센서로부터 RMMC를 통하여 실시간으로 들어오는 위치 데이타는 SDP Master의 Update_TMO 객체의 위치 데이타 큐에 저장되며, LocDB_Update SpM은 주기적으로 보고된 위치 데이타를 하나씩 가져와서 처리한다. LocDB_Update SpM의 실행 주기는 센서의 개수와 위치 보고 주기, 그리고 처리능력을 고려하여 설정되어야 한다. 예를 들어 위치를 보고하는 센서의 개수가 1,000개이고, 보고 주기가 30초이고, LocDB_Update SpM이 한번에 500개의 이동체를 처리하는 것이 가능하다고 할 때 실행주기는 15초가 되어야한다. 즉, 15초 마다 LocDB_Update SpM이 정확한 시간에 주기적으로 실행되어야 한다는 것을 의미한다.

위치 데이타 큐에서 가져온 위치 데이타는 몇 가지 과정을 거쳐서 처리된다. 먼저, 위치 데이타가 SDP Master에 의해서 처리되는 영역에 포함되는 경우, 현

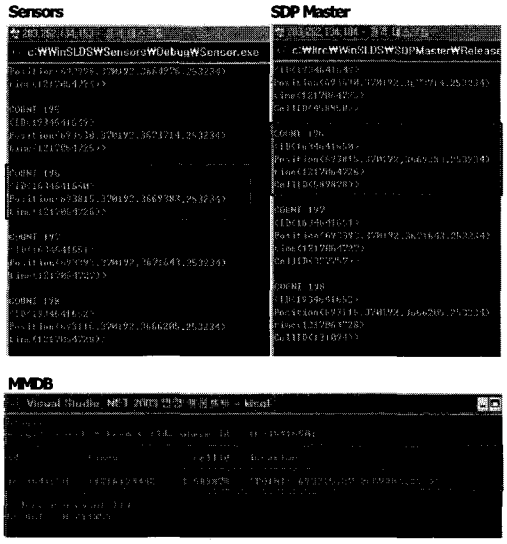
재 위치 데이타의 색인을 위해 위치 데이타의 Micro-cell ID를 계산하고 MMDB_TMO 객체의 Update SvM을 호출한다. Update SvM은 Micro-cell ID를 포함하는 위치 데이타를 MMDB에 갱신 또는 삽입하는 작업을 수행한다. 그렇지 않을 경우, SDP Master에 있는 Update_TMO 객체의 LocDB_Update SpM은 RMMC를 통하여 모든 SDP Worker들에게 위치 데이타를 브로드캐스트한다. 위치 데이타의 복사본은 모든 SDP Worker의 Update_TMO 객체의 내부에 있는 위치 데이타 큐에 저장되며, 모든 SDP Worker는 SDP Master와 동일한 방법으로 위치 데이타를 처리한다. 각각의 SDP Worker는 위치 데이타가 자신이 처리하는 영역에 포함되지 않는 경우 위치 데이타를 삭제하게 된다.



<그림 3> 비트 연결 방식을 이용한 Cell ID 계산

SDP Master나 SDP Worker에서 Micro-cell ID를 계산하기 위해 기존의 SLDS는 z-ordering 기법을 사용하였다. 하지만 z-ordering은 비트 단위 연산을 여러 번 사용하여 Cell ID 계산에 불필요한 오버헤드를 초래한다. 이와 같은 문제를 해결하기 위해서 본 논문에서는 새로운 비트 연결 방식을 고안하였다. 비트 연결 방식은 Micro-cell 영역이 100m 단위로 분할되어 있는 것을 이용하여 입력된 이동체의 좌표를 각각 100으로 나눈 후 나뉘어진 값의 비트를 연결하는 방법이다. <그림 3>에서는 (105,5)의 좌표를 갖는 위치 데이타가 들어왔을 때 비트 연결 방식을 적용하여 256의 값을 갖는 Cell ID를 결정한 예를 보여준다.

<그림 4>는 센서가 RMMC를 통하여 보고한 위치 데이타와 SDP Master에서 수신한 데이타, 그리고 MMDB에 저장된 데이타를 보여준다. 특히, SDP Master에서는 Cell ID가 추가되어서 MMDB에 저장된 것을 볼 수 있다.



<그림 4> 위치 보고 처리 결과

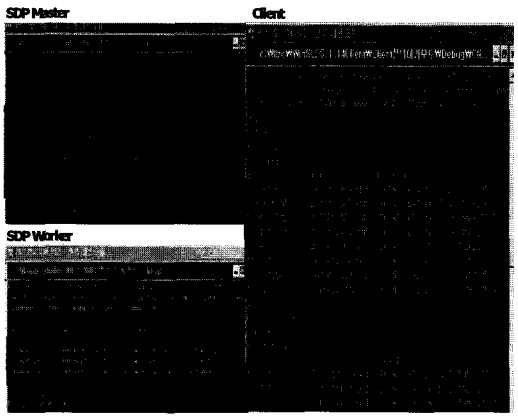
3.3 질의 처리

클라이언트가 RMMC를 통하여 질의를 SDP Master로 보내면, QueryProc_TMO 객체의 질의 큐에 저장된다. SDP Master에 있는 Query_Pickup_Decompose SpM은 주기적으로 질의를 분석하여 SDP Master가 처리하는 영역 내의 질의인 경우 MMDB_TMO 객체의 Retrieve SvM을 통해 질의를 처리한다.

이와 다르게 SDP Master가 처리하는 영역을 벗어난 질의가 수신된 경우 SDP Worker들에게 전송하고, 동시에 MMDB_TMO 객체에 있는 Retrieve SvM을 호출한다. SDP Worker는 자신이 처리하는 영역과 관련된 질의인 경우 SDP Master와 동일한 방법으로 질의를 처리하고, 그렇지 않을 경우 질의를 삭제한다. SDP Master 및 SDP Worker의 MMDB_TMO는 질의 처리 결과를 SDP Master에 있는 QueryProc_TMO 객체의 질의 처리 결과 큐에 전송하며, QueryProc_TMO의 Reassemble_Result SpM은 큐에 쌓인 질의 처리 결과를 하나로 조합한다. 조합된 질의 처리 결과는 RMMC를 통하여 해당 질의를 보낸 클라이언트에게 전송한다.

<그림 5>는 클라이언트가 수신한 질의 처리 결과와

두 SDP 노드에 저장되어있는 위치 데이타를 보여준다. <그림 5>에서 클라이언트는 SDP Master의 영역을 벗어난 질의를 요청하였고, SDP Master와 SDP Worker는 동시에 질의를 처리하게 된다. <그림 5>에서 SDP Master와 SDP Worker에서 처리된 각각의 질의 처리 결과는 통합되어 클라이언트에게 전송된 것을 볼 수 있다.



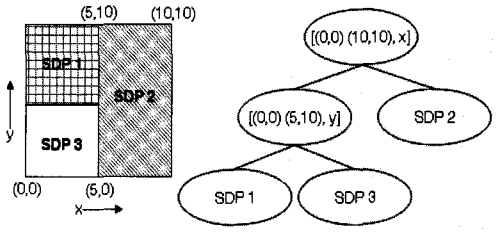
<그림 5> 질의 처리 결과

3.4 동적 부하 분산

확장 SLDS에서는 동적 부하 분산을 지원하는 새로운 분할 및 합병 알고리즘을 반영하여 Coordinator를 설계 및 구현하였다. SDP는 주기적으로 자신의 현재 상태를 Coordinator에게 보고하며, Coordinator는 보고된 SDP들의 상태 정보를 이용해 분할 및 합병을 판단한다.

Coordinator는 SDP들의 상태 정보를 관리하기 위해서 SDP 상태 테이블이라는 자료구조를 사용한다. SDP 상태 테이블은 해쉬 테이블과 상태 트리로 구성되며, 해쉬 테이블은 SDP를 식별하는 SDPID를 키(key)로 하여 생성된다. 해쉬 테이블을 사용함으로써 SDPID를 이용한 SDP 상태 테이블의 갱신을 신속히 처리할 수 있다.

SDP 상태 테이블은 SDP들의 분할 및 합병 정보를 저장하기 위해서 내부적으로 Adaptive kd-tree[12]에 기반한 상태 트리를 사용한다. 그림 6은 이러한 상태 트리의 구조를 보여준다. 상태 트리의 단말 노드는 각각 하나의 SDP의 상태를 저장하며, 단말 노드의 수는 현재 실행중인 SDP의 수를 의미한다. 상태 트리는 두 가지 특징을 가진다. 첫 번째는 단말 노드를 제외한 모든 노드는 자식 노드들의 합집합 영역과 동일한 사각형의 영역 정보 및 분할 축 정보를 저장하며 항상 두 개의 자식 노드를 가진다는 것이다. 두 번째는 루트 노드를 제외한 모든 노드들은 자신의 부모 노드에 대한 포인터를 가진다는 것이다. 이런 특징들은 GALIS에서 제시된 2분할 구조에 적합하다.



<그림 6> SDP 상태 트리

앞서 언급한 바와 같이 GALIS 아키텍처의 분할 및 합병 정책은 이동체가 노드 경계에서 이동하는 경우에 분할 및 합병의 횟수가 많이 발생할 수 있으며, 이것은 전체적인 시스템의 오버헤드를 초래한다. 이러한 문제를 해결하기 위해서 본 논문에서는 새로운 분할 및 합병 정책을 제시한다.

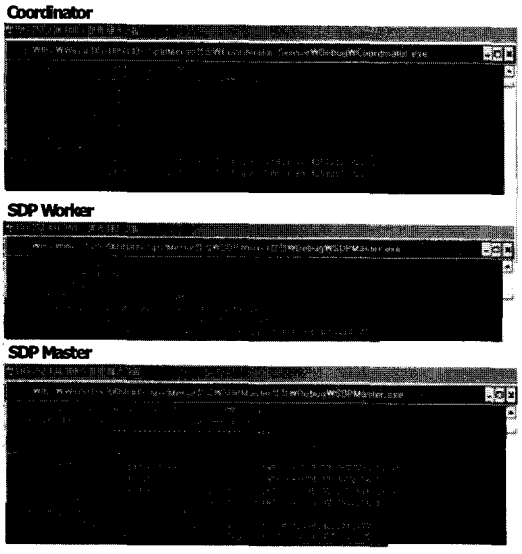
먼저 분할 정책에 개선된 내용은 다음과 같다. SDP가 처리 한계를 넘은 경우 오류 시간을 최소화하기 위해서 분할을 합병보다 높은 우선순위로 처리하게 하였으며, 분할될 노드의 부하를 분산시켜줄 스페어 노드가 없는 경우 합병 가능한 노드를 검색하여 합병 후 분할될 수 있도록 하였다.

합병 정책에 개선된 내용은 두 가지이며 다음과 같다. 첫 번째는 합병 임계치의 도입이다. 합병 임계치는 0에서 1사이의 값을 가지는 상수로 표현된다. 합병 임계치를 이용한 합병은 합병 후 SDP의 이동체 수가 (합병 임계치 × 최대 이동체 수)의 값을 넘지 않아야 한다는 합병 조건을 만족한 경우에만 합병을 수행하게 된다. 이것은 이동체가 노드 경계를 이동할 경우 분할 및 합병이 자주 발생되는 문제를 해결할 수 있다.

두 번째로 추가된 것은 스페어 노드의 유지 정책이다. 합병 임계치를 도입하면서 기존의 정책에서는 합병될 수 있는 노드가 합병하지 못하는 경우가 발생한다. 이것은 스페어 노드의 수를 감소시키며, 분할을 필

요로 하는 SDP가 분할을 시도하려고 하였을 때 분할되지 못하게 하는 문제를 야기시킬 수 있다. 이러한 문제를 해결하기 위해서 스페어 노드가 존재하지 않는 경우에는 합병 임계치를 사용하지 않는다. 즉, 합병 후 SDP의 이동체 수가 최대 이동체 수를 넘지 않은 경우에 합병을 수행한다.

Coordinator는 개선된 정책을 적용하여 SDP 및 스페어 노드에 분할 또는 합병 명령을 전송한다. Coordinator로부터 분할 또는 합병 명령을 전송 받은 SDP 및 스페어 노드는 분할 또는 합병 명령에 포함된 영역 정보를 이용하여 자신이 처리하는 영역을 변경한다. <그림 7>에서는 Coordinator가 전송한 분할 명령에 포함된 사각형 영역의 좌표, 그리고 분할 전과 후 SDP들이 처리하는 영역 정보를 보여준다.



<그림 7> Coordinator에 의해 분할 된 SDP

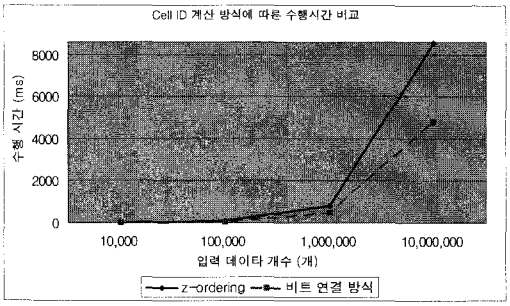
4. 성능 평가

본 장에서는 본 논문에서 개발한 확장 SLDS 대한 실험 결과를 분석한다. 실험은 두 가지 측면에서 수행되었다. 첫째, 본 논문에서 제시한 Micro-cell ID 계산 방법인 비트 연결 방식과 z-ordering을 비교하였다. 둘째, 본 논문에서 제안한 확장 SLDS의 SDP 분할 및 합병 알고리즘을 기존 SLDS의 분할 및 합병 알고리즘과 비교 평가하였다.

성능 평가에 사용되는 이동체의 위치 데이타를 생성하기 위해서 이동체 데이타 생성기인 GSTD[11]를 사용하였다. 실험에 사용된 하드웨어의 사양은 Intel CPU 1.6GHz, 786MB RAM이며, 운영체제는 Windows XP Professional을 사용하였다.

4.1 Micro-cell ID 계산 알고리즘 비교

기존의 SLDS에서 Micro-cell ID를 계산하기 위해서 사용한 z-ordering과 본 논문에서 제안한 비트 연결 방식과의 수행 시간을 비교하였다. 즉, 이동체의 위치 데이타를 1만 개에서 1,000만개 까지 증가시키면서 Cell ID를 계산하는 시간을 측정하였다.



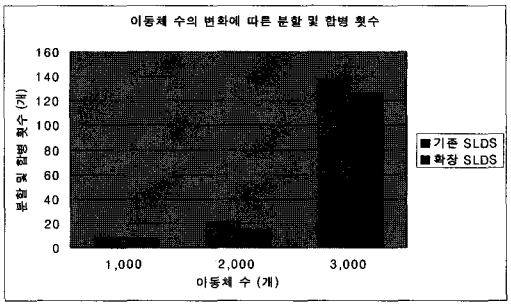
<그림 8> Cell ID 계산방식에 따른 수행시간

<그림 8>은 z-ordering과 비트 연결 방식의 수행시간을 비교한 것을 보여준다. 이 실험을 통해 비트 연결 방식이 z-ordering에 비해 약 2배 가량 빠르다는 것과 입력 데이타 개수가 증가함에 따라 z-ordering과 비트 연결 방식의 수행 시간의 차이는 점점 더 커진다는 것을 볼 수 있다. 따라서, 지속적이고 많은 데이타를 처리해야 하는 SLDS 시스템에서는 z-ordering보다 비트 연결 방식으로 Cell ID를 계산하는 것이 더 적합하다고 할 수 있다..

4.2 SDP 분할 및 합병 알고리즘 비교

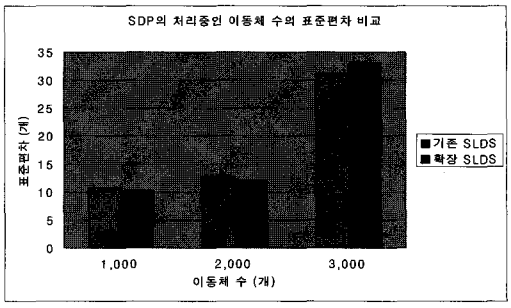
기존의 SLDS에서의 분할 및 합병 알고리즘과 본 논문에서 제시한 확장 SLDS의 분할 및 합병 알고리즘을 비교 평가하였다. 실험은 자체 제작한 시뮬레이터를 이용하여 진행되었으며, 실험에 사용된 확장 SLDS의 합병 임계치는 0.7로 설정하였다. SDP 노드는 20개로 설정하였으며, 하나의 SDP 노드의 최대와

최소 이동체 수는 각각 200개, 100개로 설정하였다. 이러한 SDP 노드 설정은 4,000개 보다 적은 이동체를 SLDS에서 처리할 수 있다는 것을 의미한다. 실험은 1,000~3,000개의 이동체에 대해 실험하였으며, 타임스탬프는 1에서 시작하여 100까지로 설정하였다. <그림 9>는 이동체 수의 변화에 따른 기존의 SLDS와 확장 SLDS가 수행한 분할 및 합병 횟수를 보여주며, 그림 10은 모든 타임스탬프에서 SDP 노드가 관리하는 이동체 수에 대한 표준편차를 보여준다.



<그림 9> 분할 및 합병횟수 비교

<그림 9>에서 확장 SLDS가 기존의 SLDS보다 분할 및 합병을 더 적게 수행한다는 것을 보여주며, 이동체 수가 증가할수록 분할 및 합병 횟수의 차이는 더 커진다는 것을 알 수 있다. 분할 및 합병 횟수가 적은 것은 동적 부하 분산에 걸리는 오버헤드가 더 적다는 것을 의미한다.



<그림 10> 디렉토리 구조

<그림 10>에서는 확장 SLDS의 SDP가 관리하는 이동체 수의 표준편차는 이동체 수가 2000개 이하일 때 기존 SLDS보다 더 작고, 3000개일 때는 그 반대이다. 실험 결과를 SLDS의 처리 한계와 관련지어 볼 때 이동체의 수가 처리 한계와 가깝지 않은 경우 기존 SLDS보다 확장 SLDS가 더 고르게 작업을 분산시키고 있다는 것을 알 수 있다.

<그림 9>와 <그림 10>을 통해서 확장 SLDS는 기존의 SLDS보다 더 적은 비용으로 부하 분산을 수행하며, 대부분의 경우 더 효율적인 부하 분산을 하고 있다는 것을 알 수 있다.

5. 결론 및 향후 연구과제

본 논문에서는 이동체의 위치 데이타를 효율적으로 관리하기 위한 GALIS 아키텍처를 기반으로 현재 위치 데이타를 실시간으로 처리하는 확장 SLDS를 설계 및 구현하였다. 본 논문에서 구현한 확장 SLDS는 TMO 프로그래밍 기법을 사용하여 위치 데이타의 저장 및 검색에 대한 실시간 처리를 지원하며, 메인 메모리에서 현재 위치 데이타를 관리하기 때문에 검색 및 갱신 오버헤드를 줄일 수 있다. 그리고, 본 논문에서 제안한 동적 부하 분산 알고리즘을 통하여 기존의 SLDS보다 더 적은 비용으로 부하 분산 처리를 수행한다는 것을 입증하였다. 또한, 현재 위치 데이타의 색인을 위해서 사용되는 z-ordering을 비트 연결 방식이라는 새로운 알고리즘으로 대체하여 잦은 갱신 작업에 더 효율적으로 대응할 수 있도록 하였다.

실시간 교통 정보 서비스와 같은 위치 기반 서비스는 도로위에 있는 많은 차량의 위치를 실시간으로 수신하고 저장되어야 사용자에게 질 높은 서비스가 가능하다. 또한 긴급 구조 서비스나, 친구 찾기 서비스, 모바일 광고 서비스 등의 다양한 위치 기반 서비스들은 모두 위치 데이타의 실시간 처리를 요구한다. 이러한 위치 기반 서비스에서 확장 SLDS는 위치 저장 플랫폼으로 활용될 수 있다. 향후 연구 과제로는 모든 상황에서 고르게 부하를 분산시킬 수 있는 분산 알고리즘, 위치 데이타의 필터링, 이동체의 위치를 이용한 트리거에 관한 연구가 필요하겠다.

참고문헌

[1] 양영규, "위치 기반 서비스(LBS: Location Based Service)기술 현황 및 전망," 한국정보처리학회 정보처리학회지, 8권 6호, 2001, pp.4-5.

[2] 한기준, "위치 기반 서비스(LBS) 표준화 연구 동향," 한국전산원 정보화 정책, 10권 4호, 2003, pp.3-17.

[3] Y. Theodoridis, "Ten Benchmark Database Queries for Location-based Services," The Computer Journal, Vol.46, No.6, 2003, pp.713-725.

[4] 류근호, 안윤애, 이준욱, 이용준, "이동 객체 데이타베이스와 위치 기반 서비스의 적용," 한국정보과학회 데이타베이스 연구, Vol.17, No.3, 2001, pp.57-74.

[5] C.S. Jensen, A. Friis-Christensen, T.B. Pedersen, D. Pfoser, S. altenis, N. Tryfona, "Location-Based Services: A Database Perspective," Proceedings of the 8th Scandinavian Research Conference on Geographical Information Science (ScanGIS), 2001, pp.59-68.

[6] 조대수, 남광우, 이재호, 민경욱, 장인성, 박종현, "대용량 위치 데이타 관리를 위한 정보 시스템," 한국 정보과학회 데이타베이스 연구, 18권 4호, 2002, pp.11-22.

[7] M.H. Kim, K.H.(Kane) Kim, Y.M. Nah, J.W. Lee, T.H. Wang, J.H. Lee, Y.K. Yang, "Distributed Adaptive Architecture for Managing Large Volumes of Moving Items," Society for Design and Process Science, IDPT-Vol.2, 2003, pp.737-744.

[8] Y.M. Nah, K.H.(Kane) Kim, T.H. Wang, M.H. Kim, J.H. Lee, Y.K. Yang, "A Cluster-based TMO-structured Scalable Approach for Location Information Systems," Proceedings of the 9th IEEE International Workshop on Object-oriented Real-time Dependable Systems (WORDS), 2003, pp.225-233.

[9] 나연묵, K.H.(Kane) Kim, 왕태형, 김문회, 이종훈, 양영규 "GALIS: LBS 시스템의 클러스터 기반 신축성소유 아키텍처," 한국정보과학회 데이타베이스 연구, 18권 4호, 2002, pp.66-80.

[10] K.H.(Kane) Kim, "Object-Oriented Real-Time Distributed Programming and Support Middleware," Proceedings of the 7th International Conference on Parallel & Distributed Systems (ICPADS), 2000, pp.10-20.

[11] Y. heodoridis, J. R. O. Silva, M. A. Nascimento," On the Generation of Spatiotemporal Datasets," Proceedings of the 6th International Symposium on Large Spatial Databases (SSD), 1999, pp.147-164.

[12] V. Gaede, O. Gnther, "Multidimensional Access Methods," ACM Computing Surveys, Vol.30, NO.2, 1998, pp.170-231.

[13] P. Rigaux, M. Scholl, A. Voisard, Spatial Databases: With Application to GIS, Morgan Kaufmann Publishers, 2001.

이승원

2004년 건국대학교 컴퓨터공학과 졸업 (공학사)

2004년 ~ 현재 건국대학교 대학원 컴퓨터공학과 석사과정

관심분야 : GIS, LBS, 텔레매틱스, MMDBMS, 분산 컴퓨팅, 실시간 컴퓨팅

강홍구

2002년 건국대학교 컴퓨터공학과 졸업 (공학사)

2004년 건국대학교 대학원 컴퓨터공학과 졸업(공학석사)

2004년 ~ 현재 건국대학교 대학원 컴퓨터공학과 박사과정

관심분야 : GIS, 공간데이타베이스, MMDBMS, MODB, LBS

홍동숙

1999년 건국대학교 컴퓨터공학과 졸업 (공학사)

2001년 건국대학교 대학원 컴퓨터공학과 졸업(공학석사)

2004년 ~ 2003년 쌍용정보통신 모바일/GIS 기술팀

2003년 ~ 현재 건국대학교 대학원 컴퓨터공학과 박사과정

관심분야 : LBS, 이동체 데이터베이스, 유비쿼터스 컴퓨팅

한기준

1979년 서울대학교 수학교육학과 졸업 (이학사)

1981년 한국과학기술원(KAIST) 전산학과 졸업 (공학석사)

1985년 한국과학기술원(KAIST) 전산학과 졸업 (공학박사)

1990년 Stanford 대학 전산학과 visiting scholar

1985년 ~ 현재 건국대학교 컴퓨터공학과 교수

2004년 ~ 현재 한국공간정보시스템학회 회장

2004년 ~ 현재 한국정보시스템감리사협회 회장

관심분야 : 공간데이타베이스, GIS, LBS, 텔레매틱스, 정보시스템 감리