동적 부하 분산을 지원하는 확장 SLDS의 설계 및 구현

Design and Implementation of the Extended SLDS Supporting Dynamic Load Balancing

이승원*, 홍동숙, 강홍구, 한기준
건국대학교 컴퓨터정보통신공학과
{swlee*, dshong, hkkang, kjhan}@db.konkuk.ac.kr
Seung-Won Lee*, Dong-Suk Hong, Hong-Koo Kang, Ki-Joon Han
Dept. of Computer Information & Communication Engineering, Konkuk University

요 약

최근 들어 개인 휴대 단말기가 보편화되고 GPS와 같은 위치 측정 기술이 발달하면서 이동체의 위치 데이타를 활용한 위치 기반 서비스에 대한 관심이 급증하고 있다. 이러한 위치 기반 서비스에서 이용되는 위치 데이타를 관리하기 위해서 일반적으로 단일 노드의 디스크 기반 GIS 서버를 사용한다. 그러나, 이동체의 경우 위치의 변화가 매우 빈번하며 대용량이기 때문에 기존의 GIS 서버로는 관리가 어렵다. 그러므로 위치 기반 서비스에서는 이동체의 대용량 위치 데이타를 효율적으로 관리 할 수 있는 위치 데이타 관리 시스템이 요구된다. 따라서 본 논문에서는 이동체의 위치 데이타를 관리하기 위해 제안된 클러스터 기반 분산 컴퓨팅 구조를 갖는 GALIS 아키텍처의 서브 시스템인 SLDS를 확장하여 동적 부하 분산을 지원하는 확장 SLDS를 설계 및 구현하였다. 또한, 실험을 통하여 확장 SLDS가 기존 SLDS에 비하여 더욱 효율적으로 부하 분산을 수행한다는 것을 검증하였다. 본 논문에서 구현한 확장 SLDS는 노드들을 주기적으로 감시하여 위치 데이타를 다수의 노드에 적절히 분산시킴으로써 대용량의 데이타를 효율적으로 관리할 수 있고 시스템의 가용성을 높일 수 있다.

1. 서 론

최근 들어 무선 인터넷 시장이 형성되면서 이동성을 기반으로 하는 서비스에 대한 관심이 급증하고 있다. 그리고, 개인 휴대 단말기의 사용이 일반화되고 GPS 등과 같은 위치 측정 기술이 발달하면서 이동체의 위치 데이타를 활용한 위치 기반 서비스는 크나큰 관심을 불러들이고 있다. 이러한 서비스를 제공하기 위해서는 이동체의 위치 데이타를 효율적으로 관리하는 위치 데이타 관리 시스템의 기술 개발이 절대적으로 필요하게 되었다[8].

이동체는 시간에 따라 위치 정보가 계속 변경되는 공간 객체를 의미한다. 이러한 공간 객체를 저장하기 위해서 일반적으로 GIS 서버를 사용해 왔다. 그러나 대부분 단일 노드로 구성되는 디스크 기반 GIS 서버는 대용량 이동체 데이타를 관리하는 것이 어렵고, 정적 데이타를 대상으로 하였기 때문에 연속적으로 위치를 변경하는 이동체를 관리하는데 적합하지 않다[7]. 이러한 문제를 해결하기 위해서 GALIS(Gracefully Aging Location Information System)와 같은 아키텍처가 제안되었다[3,4,6].

GALIS는 다수의 노드로 구성된 분산 컴퓨팅 구조를 기반으로 한다. 이러한 분산 구조는 각 지역별 데이타를 여러 노드에 저장함으로써, 위치 정보의 저장과 갱신 및 검색 질의에 대한 부하를 분산시켜 대용량

의 데이타를 효율적으로 처리할 수 있다.

GALIS에서는 노드의 분할 및 합병 정책에 따라 노드를 분할 또는 합병함으로써 동적 부하 분산을 가능하게 한다. 그러나 GALIS에서 제안한 분할 및 합병 정책은 노드 경계를 이동하는 이동체가 많을수록 분할 및 합병의 횟수가 많아질 수 있으며, 물리적으로 분산되어 있는 노드들간의 분할 및 합병 횟수가 증가하는 것은 시스템의 성능을 현저히 떨어뜨리는 원인이 된다. 이러한 문제를 해결하기 위해서 본 논문에서는 GALIS의 아키텍처에서 제시된 분할 및 합병 정책을 개선하였고, 개선된 정책을 기반으로 GALIS의 서브시스템 중 이동체의 현재 위치 데이타를 관리하는 SLDS(Short-term Location Data Subsystem)를 확장하여 확장 SLDS를 구현하였다.

본 논문의 구성은 다음과 같다. 제 2장에서는 관련 연구로 GALIS와 TMO 프로그래밍 스킴에 대해 살펴본다. 제 3장에서는 확장 SLDS의 설계 및 구현에 대해 설명하고, 제 4장에서는 실험 및 성능 평가 결과를 분석한다. 마지막으로, 제 6장에서는 결론과 향후 연구과제를 언급한다.

2. 관련 연구

본 장에서는 이동체의 위치 데이타를 관리하기 위해서 제안된 GALIS 아키텍처에 대해서 설명하고, 실시간성을 지원하는 SLDS를 구현하기 위해 사용된 TMO 프로그래밍 스킴에 대해서 기술한다.

2.1 GALIS

위치 기반 서비스를 위해 제안된 GALIS는 다중 데이타 프로세서로 구성된 클러스터 기반 분산 컴퓨팅 구조로서 각 프로세서가 서로 다른 특정 영역 내에 있는 이동체의 움직임에 대한 처리를 수행할 수 있다.

위치 기반 서비스를 위해서 관리되어야 하는 전체 공간을 2차원 평면으로 볼 때, GALIS에서는 이 2차원 평면을 n개의 영역으로 나누며, 나뉘어진 영역을 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 인덱싱 하기 위해서 구성되며, z-ordering 기법을 이용하여 Cell ID를 계산한다[4,6].

GALIS는 현재 위치 데이타를 저장하기 위한 SLDS와 과거 위치 데이타를 저장하기 위한 LLDS(Long-term Location Data Subsystem)로 구성된다. SLDS는 현재 위치 데이타의 주기적인 갱신을 효율적으로 수행하기 위해서 메인 메모리 기반의 데이타베이스(MMDB)를 사용하며, LLDS는 디스크 기반의 데이타베이스를 사용한다.

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

Coordinator는 분할 및 합병 정책에 따라 SDP 및 LDP를 분할하거나 합병함으로써 부하 분산을 수행한다[3,4]. GALIS 아키텍처에서 제시된 SLDS의 분할 정책은 SDP가 담당하는 이동체 수가 미리 정해진 최대 이동체 수를 넘는 경우 두 개의 노드로 분할한다는 것이다. 분할 축은 x축과 y축을 반복적으로 변경하면서 중간지점을 기준으로 두 개로 분할을 수행하게 된다. 또

한, SDP에서 처리하는 이동체 수가 최소 이동체 수 보다 적은 경우 두 개의 노드를 하나로 합병한다.

2.2 TMO 프로그래밍 스킴

TMO(Time-triggered Message-triggered Object)는 기존 객체 모델을 실시간 분산 컴퓨팅을 위해 확장한 모델이다. TMO는 SvM(message-triggered Service Method)과 SpM (time-triggered Spontaneous Method)이라는 두 가지 형태의 메소드를 제공한다.

TMO는 네트워크상의 다른 TMO와 통신하는 방법을 제공하기 위해서 메시지 기반 원격 메소드 호출 인터페이스인 Gate와 메모리 복제 기반의 통신 채널인 RMMC (Real-time Multicast and Memory-Replication Channel)를 가지고 있다.

SvM은 외부 TMO로부터 전달 받은 메시지에 의해서 수행되는 메소드이다. 다중 노드 상에 분산되어 있는 TMO들은 Gate를 이용하여 다른 TMO의 서비스 메소드를 수행할 수 있다. SpM은 주기성을 띄거나 시간성을 갖는 서비스를 수행하기 위한 메소드이며, 실행 시간에 대한 조건인 AAC (Autonomous Activation Condition) 명세에 의해서 실행된다.

3. 시스템 설계 및 구현

본 장에서는 GALIS 아키텍처의 SLDS를 기반으로한 확장 SLDS의 구조에 관하여 간략히 설명하고, 확장 SLDS의 핵심기능을 세 가지로 나누어서 설명한다.

3.1 확장 SLDS의 구조

본 논문에서 설계한 확장 SLDS는 그림 1에서 보여주고 있으며, GALIS 아키텍처 중 현재 위치 데이타에 관한 관리를 담당하는 SLDS 구조에 기반을 두고 있다. 확장 SLDS는 하나의 SDP Master, 다수의 SDP Worker, Coordinator로 구성되어 있다. 그리고, 확장 SLDS를 테스트하기 위하여 Sensors와 Client를 설계 및 구현하였다.

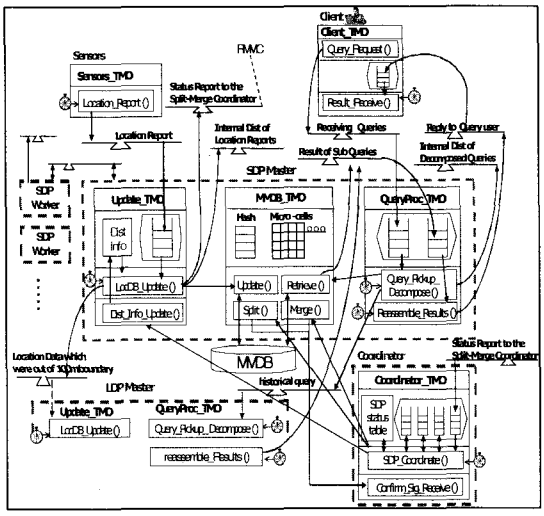

그림 1. 확장 SLDS 구조

Sensors는 이동체의 위치 데이타를 생성 및 보고하고, Client는 확장 SLDS에게 검색 질의를 요청한다. SLDS의 핵심인 SDP들은 세 개의 TMO 객체들(Update_TMO, MMDB_TMO, Query_Proc _TMO)로 구성되어 있다. SDP Master는 위치 데이타의 저장 및 검색 이외에 Sensors로부터 위치 데이타를 보고받고 Worker들에게 전송하는 역할, Client의 질의 수신과 Worker들에게 전송하는 역할, 그리고 Worker들로부터 수신된 질의 처리결과를 통합하는 역할을 수행한다. Coordinator는 SDP의 분할 및 합병을 책임지며, SDP들의 상태를 감시하기 위해 내부적으로 SDP 상태 테이블(SDP Status Table)을 가지고 있다.

3.2 위치 보고 처리

Sensors로부터 RMMC를 통하여 들어오는 위치 데이타는 Update_TMO의 위치 데이타 관련 큐에 저장되며, LocDB_ Update SpM은 주기적으로 보고된 위치 데이타를 가져와서 처리한다. 만약 위치 데이타가 SDP Master에 의해서 처리되는 영역에 포함되는 경우, 현재 위치 데이타의 인덱싱을

위해 위치 데이타의 Micro-cell ID를 계산하고 MMDB_TMO의 Update SvM을 호출한다.

Update SvM은 Micro-cell ID를 포함하는 위치 데이타를 MMDB에 갱신 또는 삽입하는 작업을 수행한다. 그렇지 않을 경우, Update_TMO의 LocDB_Update SpM은 RMMC를 통하여 모든 SDP Worker들에게 위치 데이타를 브로드캐스트한다. 위치 데이타의 복사본은 모든 SDP Worker의 Update_TMO 내부에 있는 위치 데이타 관련 큐에 저장된다. 모든 SDP Worker는 SDP Master와 동일한 방법으로 위치 데이타를 처리하며, 만약 자신이 처리하는 영역에 포함되지 않는 경우 데이타를 삭제한다. 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 시스템에서는 새롭게 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 2는 비트 연결 방식을 사용하여 Cell ID를 계산한 예제이다.

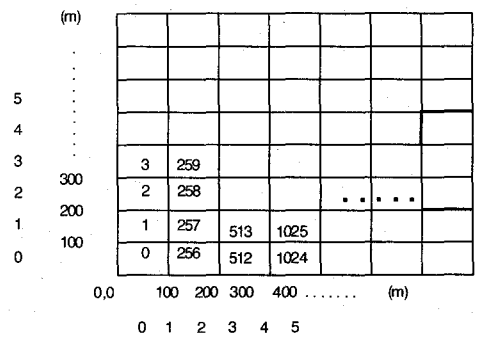

그림 2. 비트 연결 방식을 이용한 Cell ID 계산

그림 3은 Sensors가 RMMC를 통하여 보고한 위치 데이타와 SDP Master에서 수신한 데이타, 그리고 MMDB에 저장된 데이타를 보여준다.

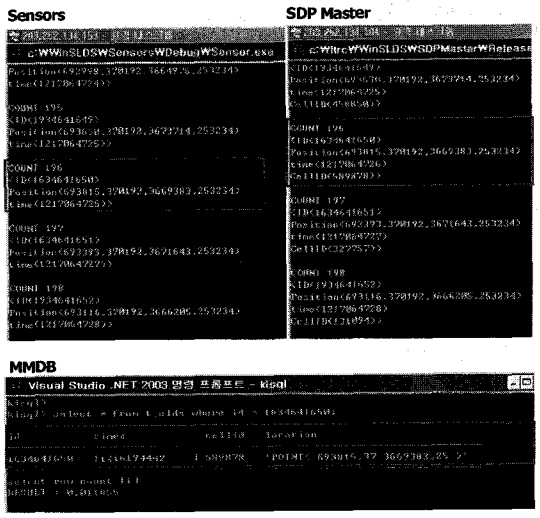

그림 3. 위치 보고 처리 결과

3.3 질의 처리

Client가 RMMC를 통하여 질의를 SDP Master로 보내면, QuerProc_TMO의 질의 관련 큐에 저장된다. SDP Master에 있는 Query_Pickup_Decompose SpM은 주기적으로 질의를 분석하여 SDP Master가 처리하는 영역 내의 질의인 경우 MMDB_ TMO의 Retrieve SvM을 통해 질의를 처리한다. 이와 다르게 SDP Master가 처리하는 영역을 벗어난 질의가 수신된 경우 SDP Worker들에게 전송하고, 동시에 MMDB_TMO에 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 Master의 QueryProc_TMO의 내부 큐에 전송하며, QueryProc TMO의 Reassemble_Result SpM은 큐에 쌓인 질의 처리 결과를 하나로 조합한다. 조합된 질의 처리 결과는 RMMC를 통하여 해당 질의를 보낸 Client에게 전송한다.

그림 4는 Client가 수신한 질의 처리 결과와 두 SDP 노드에 저장되어있는 위치 데이타를 보여준다.

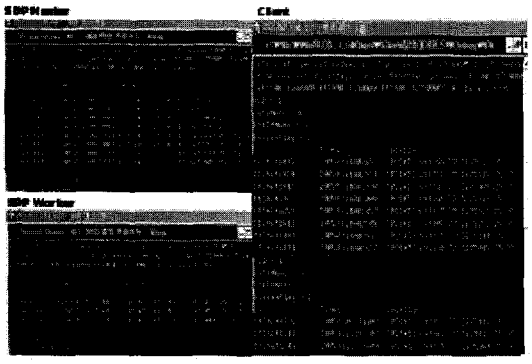

그림 4. 질의 처리 결과

3.4 동적 부하 분산

확장 SLDS에서는 동적 부하 분산을 지원하는 새로운 분할 및 합병 알고리즘을 반영하여 Coordinator를 설계 및 구현 하였다. 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 이용하여 SDP 상태 테이블을 갱신할 때 신속히 처리할 수 있다.

SDP 상태 테이블은 SDP들의 분할 및 합병 정보를 저장하기 위해서 내부적으로 Adaptive kd-tree[1]에 기반한 상태 트리를 사용한다. 그림 5는 이러한 상태 트리의 구조를 보여준다. 상태 트리의 단말 노드는 각각 하나의 SDP의 상태를 저장하며, 단말 노드의 수는 현재 실행중인 SDP의 수를 나타낸다. 상태 트리는 두 가지 특징을 가진다. 그 중 하나는 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. 다음으로 루트 노드를 제외한 모든 노드들은 자신의 부모 노드에 대한 포인터를 가진다. 이런 특징들은 GALIS에서 제시된 2분할 구조에 적합하다.

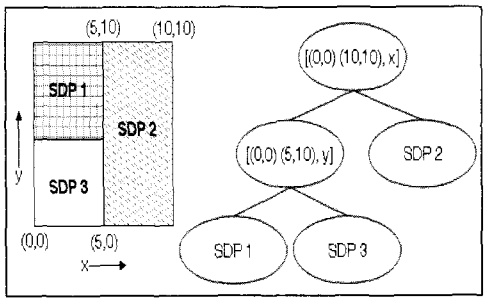

그림 5. SDP 상태 트리

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. 이 문제를 해결하기 위해서 본 논문에서는 새로운 분할 및 합병 정책을 제시한다.

먼저 분할정책에 추가된 내용은 다음과 같다. SDP가 처리한계를 넘은 경우 오류시간을 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.

합병정책에 추가된 내용은 두 가지 이며 다음과 같다. 첫 번째는 merge threshold의 도입이다. merge threshold는 0에서 1 사이의 값을 가지는 상수로 표현된다. merge threshold를 이용한 합병은 합병 후 SDP의 이동체 수가 (merge threshold×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. merge threshold를 도입하면서 기존의 정책에서는 합병될 수 있는 노드가 합병하지 못하는 경우가 발생한다. 이것은 스페어 노드의 수를 감소시키며, 분할을 필요로 하는 SDP가 분할을 시도하려고 하였을 때 분할되지 못하게 하는 문제를 야기시킬 수 있다. 이 문제를 해결하기 위해서 스페어 노드가 존재하지 않는 경우에는 merge threshold를 사용하지 않는다. 즉, 합병 후 SDP의 이동체 수가 최대 이동체 수를 넘지 않은 경우에 합병을 수행한다.
그림 6은 Coordinator가 SDP들에게 보낸 영역 분할 명령과 분할을 수행한 후의 SDP들의 영역 정보를 보여준다.

4. 실험 및 성능 평가

본 장에서는 본 논문에서 구현한 확장 SLDS 대한 실험 결과를 분석한다. 실험은 두 가지 측면에서 수행되었다. 첫째, 본 논문에서 제시한 Micro-cell ID 계산 방법

그림 6. Coordinator에 의해 분할 된 SDP

인 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 GSTD를 사용하였으며, 하드웨어 사양은 Intel CPU 1.6GHz, 786MB RAM 이며, 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. 이동체의 위치 데이타를 1만 개에서 1000만 개까지 증가시키면서 Cell ID를 계산하는 시간을 측정하였다.

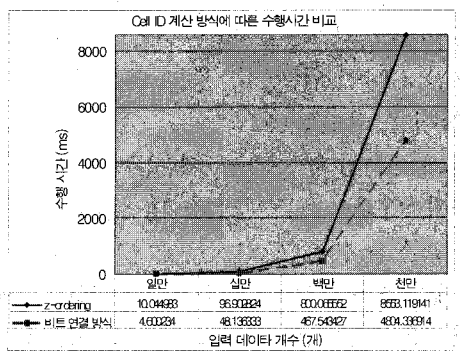

그림 7. Cell ID 계산방식에 따른 수행시간

그림 7은 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 진행되었다. 실험에 사용된 확장 SLDS의 merge threshold는 0.7이다. SDP 노드는 20개로 설정하였으며, 하나의 SDP 노드의 최대와 최소 이동체 수는 각각 200개, 100개로 설정하였다. 이러한 SDP 노드 설정은 4000개 보다 적은 이동체를 SLDS에서 처리할 수 있다는 것을 의미한다. 실험은 1000~3000개의 이동체에 대해 실험하였으며, 타임스탬프는 1에서 시작하여 100까지로 설정하였다. 그림 8은 이동체 수의 변화에 따른 기존의 SLDS와 확장 SLDS가 수행한 분할 및 합병 횟수를 보여주며, 그림 9는 모든 타임스탬프에서 SDP 노드가 관리하는 이동체 수에 대한 표준편차를 보여준다.

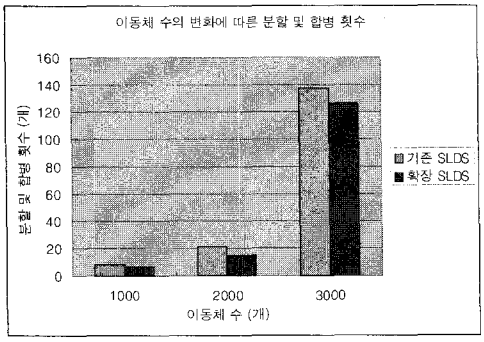

그림 8. 분할 및 합병횟수 비교

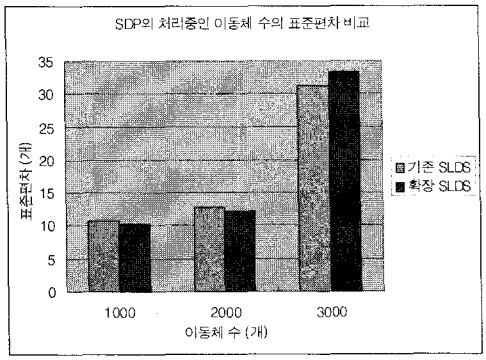

그림 9. SDP가 관리하는 이동체 수의 표준편차 비교

그림 8에서 확장 SLDS가 기존의 SLDS보다 분할 및 합병을 더 적게 수행한다는 것을 보여주며, 이동체 수가 증가할 수록 분할 및 합병 횟수의 차이는 더 커진다. 분할 및 합병 횟수가 적은 것은 동적 부하 분산에 걸리는 오버헤드가 더 적다는 것을 의미한다.

그림 9에서는 확장 SLDS의 SDP가 관리하는 이동체 수의 표준편차는 이동체 수가 2000개 이하일 때 기존 SLDS보다 더 작고, 3000개일 때는 그 반대이다. 실험 결과를 SLDS의 처리한계와 관련 지어 볼 때 이동체의 수가 처리한계와 가깝지 않은 경우 기존 SLDS보다 확장 SLDS가 더 고르게 작업을 분산시키고 있다는 것을 알 수 있다.

그림 8과 9를 통해서 확장 SLDS는 기존의 SLDS보다 더 적은 비용으로 부하 분산을 수행하며, 이동체 수가 많지 않은 경우 더 효율적인 부하 분산을 하고 있다는 것을 알 수 있다.

4. 결론 및 향후 연구과제

본 논문에서는 이동체의 위치 데이타를 효율적으로 관리하기 위한 GALIS 아키텍처를 기반으로 현재 위치 데이타를 처리하는 확장 SLDS를 설계 및 구현하였다.

본 논문에서 구현한 확장 SLDS는 메인 메모리에서 현재 위치 데이터를 관리하기 때문에 검색 및 갱신 오버헤드를 줄일 수 있으며, TMO프로그래밍 스킴을 사용하여 위치 데이타의 저장 및 검색에 대한 실시간 처리를 지원한다. 그리고, 본 논문에서 제안한 동적 부하 분산 알고리즘을 통하여 기존의 SLDS보다 더 적은 비용으로 부하 분산 처리를 수행한다는 것을 입증하였다. 또한, 현재 위치 데이타의 인덱싱을 위해서 사용되는 z-ordering을 비트 연결 방식이라는 새로운 알고리즘으로 대체하여 잦은 갱신 작업에 더 효율적으로 대응할 수 있도록 하

였다.

향후 연구 과제로는 위치 데이타의 필터링에 대한 연구와 이동체의 위치를 이용한 트리거에 관한 연구가 필요하겠다.

참 고 문 헌

1. V. Gaede, O. Gunther, "Multidimensional Access Methods," ACM Computing Surveys, Vol.30, NO.2, 1998, pp.170-231.
2. K.H.(Kane) Kim, "Object-Oriented Real-Time Distributed Programming and Support Middleware," 7th Int'l Conf. on Parallel & Distributed Systems (ICPADS), 2000, pp.10-20.
3. 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.
4. 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.
5. P. Rigaux, M. Scholl, A. Voisard, Spatial Databases: With Application to GIS, Morgan Kaufmann Publishers, 2001.
6. 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.
7. 조대수, 남광우, 이재호, 민경욱, 장인성, 박종현, "대용량 위치 데이타 관리를 위한 정보 시스템," 한국 정보과학회 데이타베이스연구, 18권 4호, 2002, pp.11-22.
8. 한기준, "위치 기반 서비스(LBS) 표준화 연구 동향," 한국전산원 정보화 정책, 10권 4호, 2003, pp.3-17.