현재 위치 데이타 관리를 위한 그리드 기반 노드 분할 알고리즘†

A Grid-based Node Split Algorithm for Managing Current Location Data

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

요지 최근 이동체의 위치 데이타를 활용하는 위치 기반 서비스에 대한 관심이 급증하고 있다. 이러한 위치 기반 서비스에서 이용되는 대용량 위치 데이타를 효율적으로 관리하기 위한 아키텍처로서 클러스터 기반 분산 컴퓨팅 구조를 갖는 GALIS(Gracefully Aging Location Information System) 아키텍처가 제안되었다. GALIS는 비균등 2-단계 그리드를 사용하여 노드들의 부하 분산 및 색인을 수행한다. 하지만 비균등 2-단계 그리드의 분할 알고리즘은 이동체가 특정 지역에 편중되는 경우 불필요한 노드를 생성하는 문제를 가지고 있다.

따라서 본 논문에서는 이동체의 다양한 분포에 대하여 더욱 효율적인 노드 분할 알고리즘을 제시한다. 본 논문에서 제시한 노드 분할 알고리즘은 이동체의 현재 위치에 따른 공간적 분포를 고려하기 때문에 이동체가 특정 지역에 편중되는 경우에도 불필요한 노드를 생성하지 않고 효율적인 부하 분산을 수행할 수 있으며, 분산 시스템에서 중요시 되는 균형있는 부하 분산을 수행할 수 있다. 또한, 가상 노드 분할 시뮬레이터를 구현하여 다양한 이동체 데이타 분포 형태에 대해 실험하였으며, 이러한 실험을 통하여 기존의 알고리즘보다 더욱 효율적으로 노드를 분할하는 것을 검증하였다.

1. 서 론

최근 들어 무선 이동통신 기술의 발달로 휴대폰과 PDA 등의 무선 이동 기기가 보편화되고 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 시스템이 필요하다[5].

위치 기반 서비스를 위해 제안된 GALIS 아키텍처는 대용량의 이동체 데이타를 처리하기 위해서 클러스터 기반 분산 컴퓨팅 구조로 설계되었다[1,2,4]. GALIS는 현재 위치 데이타를 관리하기 위한 SLDS(Short-term Location Data Subsystem)와 과거 위치 데이타를 관리하기 위한 LLDS(Long-term Location Data Subsystem)로 구성되어 있다. 이 중 SLDS는 전체 서비스 영역을 비균등 2-단계 그리드 구조에 따라 더 작은 영역으로 분할하며, 각 노드는 서로 다른 영역내에 있는 이동체의 움직임에 대해 처리한다.

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

GALIS에서 정의한 비균등 2-단계 그리드 구조는 이동체 데이타의 부하 분산과 색인을 위해 제안된 방법이며, 부하 분산을 위해 특정 노드에서 처리하는 이동체의 수가 증가하여 상한을 초과하는 경우 두 개의 노드로 분할하고 있다.

비균등 2-단계 그리드 구조는 기본적으로 2분할 구조를 가지고 있으며, 분할하는 영역의 중간지점을 기준으로 노드 분할을 수행한다[4]. 이러한 노드 분할 방법은 이동체의 현재 위치 데이타에 대한 분포를 고려하지 않았기 때문에 불필요한 노드를 생성하며, 노드간 부하 불균형의 문제점을 가지고 있다.

따라서 본 논문에서는 SLDS에 적합한 효율적인 노드 분할 알고리즘을 제시한다. 본 논문에서 제안한 알고리즘은 분할된 후의 두 노드의 이동체 수 및 영역의 크기를 고려하기 때문에 불필요한 노드를 생성하지 않으며, 노드들의 부하 불균형 문제도 해결할 수 있다.

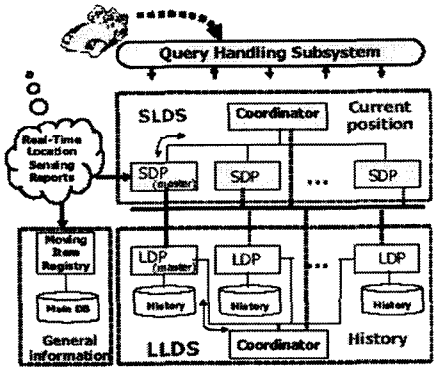
본 논문의 구성은 다음과 같다. 제 2장에서는 관련 연구로 GALIS 아키텍처와 비균등 2-단계 그리드 구조에 대해 살펴본다. 제 3장에서는 비균등 2-단계 그리드의 문제점에 관하여 예를 들어 설명하고, 제 4장에서는 본 논문에서 제시하는 노드 분할 알고리즘에 대해 설명한다. 제 5장에서는 실험 및 성능 평가 결과를 분석하고, 마지막으로 제 6장에서는 결론과 향후 연구과제를 언급한다.

2. 관련 연구

본 장에서는 이동체의 위치 데이타를 관리하기 위해서 제안된 GALIS 아키텍처와 비균등 2-단계 그리드에 대해 설명한다.

2.1 GALIS

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



<그림 1> GALIS 전체 구조

GALIS 아키텍처의 전체 구조는 <그림 1>에서 보여준다. GALIS는 현재 위치 데이타를 관리하기 위한 SLDS와 과거 위치 데이타를 관리하기 위한 LLDS로 구성된다.

SLDS는 현재 위치 데이타의 주기적인 갱신을 효율적으로 처리하기 위해서 메인 메모리 기반의 데이타베이스를 사용하며, LLDS는 디스크 기반의 데이타베이스를 사용한다[1,2,4]. SLDS 내부의 각 노드를 SDP(Short-term Data Processor)라고 하며, 각각의 SDP는 하나의 Macro-cell의 영역 내에 포함된 이동체의 현재 위치 정보를 관리한다. SDP 중 하나의 노드를 SDP Master라 하며, 다른 SDP는 SDP Worker라 부른다.

LLDS도 SLDS와 비슷한 형태이며 LLDS의 내부의 노드는 LDP (Long-term Data Processor) Master와 LDP Worker로 구성된다. SLDS와 LLDS는 SDP와 LDP외에 부하 분산을 위한 Coordinator도 가지고 있다. SDP Master는 실시간으로 보고되는 이동체의 위치 정보를 수신하여 다른 SDP 및 LDP Master로 전달하는 역할을 한다. Coordinator 노드는 SDP 또는 LDP에서 처리하고 있는 이동체의 수를 관찰하며, 비균등 2-단계 그리드 구조를 이용하여 동적 부하 분산을 수행한다[4].

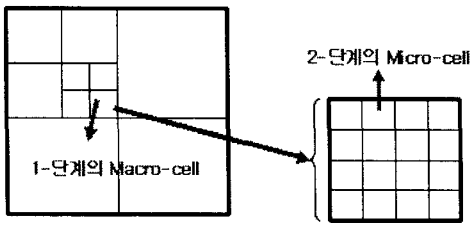
2.2 비균등 2-단계 그리드

위치 기반 서비스를 위해서 처리되어야 하는 전체 공간을 2차원 평면으로 볼 때, GALIS에서는 이 2차원 평면을 n개의 영역으로 나누며, 나뉘어진 각각의 영역을

Macro-cell 이라고 부른다.

Macro-cell은 서로 다른 크기의 사각형 영역으로 구성될 수 있다. Macro-cell의 분할은 하나의 Macro-cell에 포함된 이동체의 수가 미리 정해놓은 최대 이동체 수를 넘는 경우 수행되며, 분할 방법은 x축과 y축을 번갈아 가면서 중간 지점으로 분할한다[1,2,4]. 또한 특정 Macro-cell에 포함된 이동체 수가 미리 정해놓은 최소 이동체 수 보다 작은 경우에는 인접 노드와 합병을 수행한다. 각각의 Macro-cell 영역에 포함되는 이동체에 대한 위치 정보는 하나의 노드가 담당해서 처리하며, Macro-cell은 다시 100mX100m의 일정한 크기를 가지는 Micro-cell이라는 영역으로 나뉘어진다.

Micro-cell은 하나의 노드에서 이동체의 현재 위치 정보를 색인하기 위해서 구성되며, z-ordering 기법을 사용한다[4]. z-ordering 기법을 이용하여 이동체 데이타를 입력하게 되면, 시스템은 특정 Micro-cell에 포함된 이동체 수를 알 수 있다.

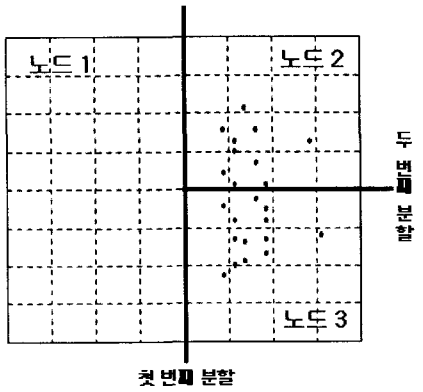


<그림 2> 비균등 2-단계 그리드 구조

<그림 2>의 왼쪽 그림은 전체 서비스 영역을 10개의 Macro-cell로 분할한 것을 보여주며, 오른쪽의 그림은 하나의 Macro-cell이 16개의 일정한 크기를 가지는 Micro-cell로 구성된 것을 보여준다.

3. 문제 정의

비균등 2-단계 그리드는 노드간 부하 불균형 문제와 불필요한 노드 생성의 문제를 가지고 있다. 불필요한 노드 생성은 특정 노드가 분할이 필요할 때 분할을 하지 못하게 하는 원인이 될 수 있다. 또한, 부하 불균형 문제는 시스템 자원을 적절히 사용하지 못하고 있다는 것을 의미한다.



<그림 3> 비균등 2-단계 그리드의 분할

<그림 3>에서는 비균등 2-단계 그리드에서 제안한 분할 방법에 의해 분할된 노드상태를 보여준다. <그림 3>에 대한 설명을 하기 전에 <그림 3>에 해당하는 전체 영역을 서비스하고 있는 시스템은 최대 3대의 노드를 사용할 수 있다고 가정한다.

<그림 3>에서는 이동체가 우측 영역에 몰려있음에도 불구하고 첫 번째 분할이 x축의 중간지점으로 발생하기 때문에 실제로 한번의 분할을 통해 이동체를 두 노드에 분산시킬 수 없다. 그래서, 두 번째 분할을 통하여 노드 2와 노드 3에 이동체가 분산되도록 하게 된다. 이와 같은 상황에서 노드 1,2,3의 이동체 수는 균형을 이루지 못하고 있다. 노드 1의 경우 실제로 부하가 전혀 없음(이동체 수=0)에도 불구하고 특정 영역에 대해 처리하고 있으며, 노드 2에 있는 이동체가 노드 3으로 이동하여 분할을 해야 할 경우 노드 부족으로 분할을 수행할 수 없는 문제가 발생한다.

4. 노드 분할 알고리즘

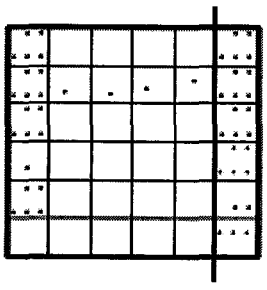
본 장에서는 새로운 노드 분할 알고리즘에 대해 설명한다. 노드 분할 알고리즘은 후보 분할 지점 결정과 최종 분할 지점 결정에 대한 두 가지 세부 알고리즘으로 나뉘어진다.

4.1 후보 분할 지점 결정

노드 분할 알고리즘에서 고려하는 것으로 분할 후 이동체 수가 있다. 노드 분할 알고리즘은 분할된 후의 노드가 가지는 이동체 수를 고려하여 분할 지점을 결정하기 때문에 알고리즘을 수행하는 도중 분할이 된 후 하나의 노드가 가져야 하는 이동체 수의 최적 값을 알아야 한다. 이 최적 값은 현재 노드의 이동체 수를 반으로 나눈 값과 같다. 이 값을 분할 후 최적 이동체 수라고 한다.

분할 지점을 결정할 때 이동체 수만을 고려할 경우 노드가 처리하는 영역에 대한 불균형이 생기게 된다. <그림 4>에서 영역의 크기를 고려하지 않은 경우에 발생할 수 있는 분할의 예를 보여주고 있다.

<그림 4>에서 이동체 수는 적절하게 나뉘어 졌지만 영역의 크기를 고려하지 않았기 때문에 한 노드의 크기가 매우 작으며 이러한 경우 영역의 크기가 작은 쪽에 포함된 이동체들이 하나의 셀 크기만큼 다른 노드 영역으로 이동하는 경우 또 다시 분할이 필요하게 된다. 이러한 문제를 해결하기 위해서 분할 후 최적 이동체 수를 최소 값과 최대 값으로 표현되는 일정 범위 값으로 다시 계산한다. 이 값을 느슨한 최적 값이라고 정의한다. 느슨한 최적 값을 이용해 분할 지점의 후보들을 선출한다.



<그림 4> 영역의 크기 비고려 분할 예

후보 분할 지점을 결정하는 알고리즘은 셀에 포함된 이동체 수를 값으로 가지는 정수형 이차원 배열(cells)을 입력으로 하며, cellij는 x축의 i번째 y축의 j번째 셀에 포함된 이동체 수를 의미한다. SLDS의 특징상 특정 Micro-cell에 포함된 이동체의 수를 알 수 있기 때문에 알고리즘에서 이용하였다. 그 외에 최적 값 변동계수(CV)와 현재 노드가 처리하고 있는 이동체 수(MONum)를 입력으로 받는다.

분할 지점 결정 방법은 분할 후 최적 값과 느슨한 최적 값의 최소(MinLOPT), 최대(MaxLOPT)를 구하는 것으로 시작된다. <그림 5>에 이러한 값들을 구하는 알고리즘이 기술되어있다. 다음으로 입력받은 분할 축 정보에 따라 x축인 경우, 첫 열에 포함된 이동체 수를 합산하여 일차원 배열에 저장하고 다음 열은 앞 열에서 저장된 값과 현재 열의 이동체 수를 모두 합하여 누적된 값을 저장한다. y축의 경우에는 행에 대해서 x축과 동일하게 진행한다. 이렇게 구해진 일차원 배열은 누적배열(CA)이라고 정의하며, x축의 경우 CAx, y축인 경우 CAy로 표기하고, CA는 CAx의 i번째 원소를 나타낸다. CA는 누적된 값이기 때문에 자동적으로 정렬된다.

```
알고리즘: 후보 분할 지점 결정(cells, CV, MONum)
OPT ← MONum / 2
MinLOPT ← OPT - (OPT * (CV/100))
MaxLOPT ← OPT + (OPT * (CV/100))
// x, y축에 대한 누적 배열(CA) 계산
for i from 1 to x축 셀 수 do
    for j from 1 to y축 셀 수 do
        CA^x_i ← CA^x_i + cells_ij
    end for
end for
for i from 1 to y축 셀 수 do
    for j from 1 to x축 셀 수 do
        CA^y_i ← yCA^y_i + cells_ij
    end for
end for
// 후보 분할 지점 선택 CSP(Candidate Split Points)
i ← 1
while CA^x_i < MaxLOPT
    if CA^x_i > MinLOPT
        CSP^x ← CSP^x + i
    end if
    i ← i + 1
end while
i ← 1
while CA^y_i < MaxLOPT
    if CA^y_i > MinLOPT
        CSP^y ← CSP^y + i
    end if
    i ← i + 1
end while
return CSP
```

<그림 5> 후보 분할 지점 결정 알고리즘

후보 분할 지점을 선택하기 위해서 CA와 느슨한 최적 값을 이용한다. 정렬된 각 CA에서 느슨한 최적 값이 삽입될 수 있는 위치를 찾아서 후보 분할 지점의 집합인 CSP에

저장한다. x축을 기준으로 하는 후보 분할 지점들은 CSPx로 표기하며, y축을 기준으로 하는 후보 분할 지점들은 CSPy로 표기한다. CSP에 저장된 값은 해당 축을 기준으로 몇 번째 Micro-cell에서 분할될 수 있는지 나타내는 정수 값이다.

4.2 최종 분할 지점 결정

후보가 되는 분할 지점들이 CSP에 저장되면 최종 분할 지점을 결정한다. 최종 분할 지점을 선택하는 중요한 요소인 DF는 분할 후의 두 노드가 가지는 이동체 수와 영역의 비에 대한 차이를 나타내며, 이 차이가 가장 적은 후보 분할 지점을 최종 분할 지점으로 선택한다. <그림 6>에 나타난 식은 DF를 구하는 방법이다. 후보 분할 지점을 기준으로 분할을 하였을 경우 분할된 두 개의 노드를 노드 1과 2라고 하며, MONum1과 MicroNum1은 각각 노드 1이 처리하는 이동체 수와 Micro-cell의 수를 나타낸다.

$$DF = | MONum_1/MicroNum_1 - MONum_2/MicroNum_2 |$$

<그림 6> 두 노드의 이동체 수와 영역의 비 차이

<그림 7>은 각각의 후보 분할 지점에 대한 DF를 구하면서 최소 DF 값을 가지는 후보 분할 지점을 선택하는 알고리즘을 보여준다.

```
알고리즘: 최종 분할 지점 결정(CSP, MONum)
xSize ← x축 셀 수
ySize ← y축 셀 수
MinDF ← MONum //RP의 최대 값
for each i ∈ CSP^x do
  DF ← |CA^x_i /(ySize*i)-(MONum-CA^x_i)/(ySize*(xSize- i)|
  if DF < MinDF then
    T ← ∅
    T ← T + i
    MinDF ← DF
  else if DF = MinDF then
    T ← T + i
  end if
end for
for each i ∈ CSP^y do
  DF ← |CA^y_i /(xSize*i) - (MONum-yCA^y_i)/(xSize*(ySize-i)|
  if DF < MinDF then
    T ← ∅
    T ← T + i
    MinDF ← DF
  else if DF = MinDF then
    T ← T + i
  end if
end for
return T
```

<그림 7> 최종 분할 지점 결정 알고리즘

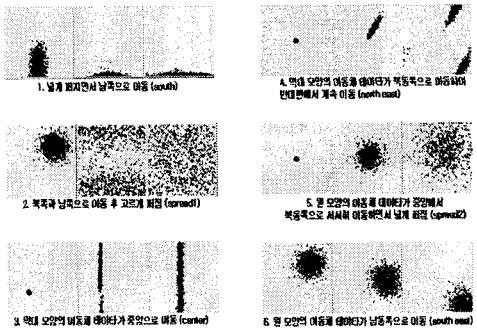
만약 선택된 최종 분할 지점이 하나 이상 선택된 경우에는 선택된 지점들 중 분할되는 지점에 인접한 셀들의 이동체 수가 가장 적은 지점을 선택한다. 이 경우에도 하나 이상의 지점이 선택될 수 있다. 이 경우, 다시 선택된 지점들 중 중간지점에 가장 가까운 분할 지점을 선택한다.

5. 실험 및 성능 평가

본 장에서는 실험에 사용된 이동체 데이타와 두 가지 알고리즘의 성능 평가를 위해 구현된 가상 노드 분할 시뮬레이터의 설정에 관하여 설명한다. 마지막으로 개선된 노드 분할 알고리즘과 비균등 2-단계 그리드의 분할 알고리즘을 노드 분산을 위해 사용한 노드 수와 분할 및 합병 횟수 그리고 분산 정도의 세 가지 측면에서 비교한다.

5.1 이동체 데이타

실험에 사용된 이동체 데이타는 GSTD[3]를 이용하였으며, 이동체 수 1000개와 타임스탬프 수는 10개로 하였다. 또한, 다양한 경우의 이동체 유형에 관하여 실험하기 위해 6가지의 분포 유형을 가지는 데이타를 생성하였다. <그림 8>에 6가지 유형의 이동체 데이타의 생성된 모습이 나타나있다.



<그림 8> 6가지 유형의 이동체 데이타

5.2 가상 노드 분할 시뮬레이터 설정

제안된 알고리즘을 평가하기 위해 가상 노드 분할 시뮬레이터를 구현하였다. 시뮬레이

터는 비균등 2-단계 그리드 구조를 이용해서 실행되는 것과 본 논문에서 제안한 개선된 분할 알고리즘을 이용해서 실행되는 두 가지가 있다. 두 가지 시뮬레이터의 설정 변수를 <표 1>에 정리하였다.

<표 1> 시뮬레이터의 공통 설정 변수

변수 명	설명
MAXMONUM	한 노드에서 처리 가능한 최대 이동체 수
MINMONUM	노드 합병을 위한 최소 이동체 수
NODENUM	전체 노드의 수
TSNUM	입력 데이타의 타임스탬프 수
MONUM	입력 데이타의 이동체 수
CV	최적 값 변동 계수

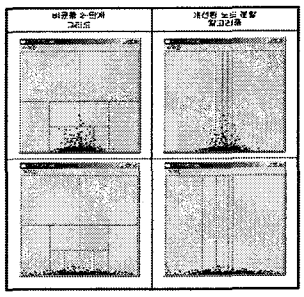
<표 2>에서는 실험에 사용된 설정 값을 보여준다. 두 시뮬레이터의 설정은 모두 동일하다. <표 2>에서 CV는 개선된 노드 분할 알고리즘을 이용한 시뮬레이터에서 사용되는 추가적인 설정 변수로 최적 값 변동 계수이며 비균등 2-단계 그리드를 이용한 시뮬레이터에서는 사용되지 않는다.

<표 2> 시뮬레이터 설정

알고리즘 / 설정 변수	비균등 2-단계 그리드	개선된 노드 분할 알고리즘
MAXMONUM	100	100
MINMONUM	50	50
NODENUM	30	30
TSNUM	10	10
MONUM	1000	1000
CV	N/A	10

5.3 성능 평가 및 분석

성능 평가는 6개의 데이타 유형 별로 부하 분산을 위해 사용한 평균 노드 수, 분할 및 합병 횟수, 그리고 부하 분산의 균형 정도를 비교하는 것을 통해 이뤄졌다.

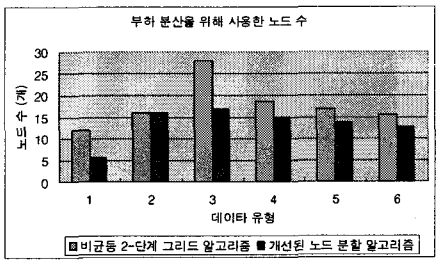


<그림 9> 시뮬레이터에 의한 분할 모습

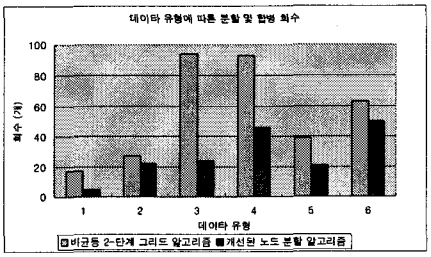
시뮬레이터를 수행한 결과는 타임스탬프별로 분할된 모습을 볼 수 있으며, <그림 9>에서는 실험 데이타 유형 중 첫 번째인 남쪽으로 퍼지면서 이동하는 이동체 유형에 대한 분할 모습을 보여준다.

<그림 9>의 위 부분의 그림은 세 번째 타임스탬프에서 분할 된 모습이며, 아래 부분은 마지막 타임스탬프에서 나타난 분할 모습이다. 이 그림을 통해 알 수 있는 것은 비균등 2-단계 알고리즘은 분할 시 부하가 전혀 없는 불필요한 노드를 생성하고 있다는 것과 개선된 알고리즘에서는 모든 노드의 부하가 균형 있게 분산되는 것을 볼 수 있다.

<그림 10>은 부하 분산을 위해 사용한 노드 수를 나타낸다. 고르게 퍼지는 데이타인 2번 유형의 경우는 동일한 노드 수로 부하 분산을 수행하였고 그 외에는 개선된 알고리즘이 더 적은 노드를 사용하고 있다는 것을 볼 수 있다.

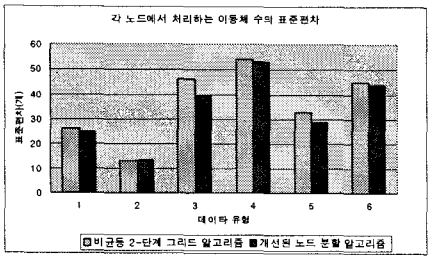


<그림 10> 부하 분산을 위해 사용한 노드 수



<그림 11> 데이타 유형에 따른 분할 및 합병 횟수

<그림 11>에서는 부하 분산 시의 오버헤드 정도를 알아보기 위해 두 알고리즘을 사용한 시뮬레이터에서 발생한 분할 및 합병 횟수를 측정해 보았다. <그림 11>에서 보는 바와 같이 대부분의 경우에 개선된 노드 분할 알고리즘이 분할 및 합병 횟수가 적으며, 이것은 분할이나 합병을 더 적게 수행하면서 부하 분산을 처리할 수 있다는 것을 보여준다.



<그림 12> 각 노드에서 처리하는 이동체 수의 표준편차

<그림 12>는 두 분할 알고리즘 중 어느 것이 더 균등한 부하 분산을 수행하고 있는지 알아보기 위해 노드들이 처리하는 이동체 수의 표준 편차를 측정하였다. 2번 데이타 유형을 제외한 모든 경우 표준 편차는 개선된 노드 분할 알고리즘에서 더 작게 나타났으며, 2번 유형의 경우에는 거의 동일하게 나타났다. 이것은 2번 유형에 대해서는 두 알고리즘 모두 균형 있게 부하를 분산시키고 있으며, 나머지의 경우에는 개선된 노드 분할 알고리즘이 더 균형있는 부하 분산을 수행하고 있다는 것을 나타낸다.

6. 결론 및 향후 연구과제

본 논문에서는 위치 기반 서비스를 위해 제안된 GALIS 아키텍처 중 현재 위치를 관리하는 SLDS에 적합한 노드 분할 알고리즘을 제안하였다. 제안된 알고리즘은 비균등 2-단계 그리드 알고리즘에서 발생하는 불필요한 노드를 생성하지 않는다. 또한, 적은 분할 및 합병을 통해 더 고르게 부하를 분산시키는 것을 확인하였다.

지금까지의 부하 분산은 미리 정해진 전역적인 최대와 최소 이동체 수를 기준으로 이루어 졌으나, 향후에는 시스템 성능에 따라 적합한 최대와 최소 이동체 수를 적용하는 방안에 대한 연구가 필요하다.

< 참 고 문 헌 >

[1] 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.

[2] 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.

[3] Y. Theodoridis, 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.

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

[5] 이승원, 강홍구, 홍동숙, 한기준, "실시간 위치 기반 서비스를 위한 확장 SLDS의 설계 및 구현", 한국공간정보 시스템학회 논문지, 7권 2호, 2005, pp.47-56.