공간 네트워크에서 이동객체의 위치정보 관리를 위한 동적 분산 그리드 기법

(Dynamic Distributed Grid Scheme to Manage the Location Information of Moving Objects in Spatial Networks)

김 영 창 † 홍 승 태 †
(YoungChang Kim) (SeungTae Hong)
조 경 진 † 장 재 우 ††
(KyungJin Jo) (JaeWoo Chang)

요 약 최근 공간 네트워크에서 대용량 이동객체의 위치정보를 관리하기 위한 DS-GRID(distributed S-GRID)가 제안되었다[1]. 그러나 DS-GRID는 균일 크기의 그리드 셀을 이용하기 때문에, 실제 응용에서 빈번히 발생하는 이동객체의 쏠림 현상을 효율적으로 관리하지 못하는 단점을 지닌다. 이를 해결하기 위해, 본 논문에서는 이동객체의 밀도에 따라 그리드 셀을 동적으로 분할하는 동적 분산 그리드 기법을 제안한다. 아울러 이를 위한 k-최근접 질의처리 알고리즘을 제안한다. 마지막으로 성능 평가를 통해 이동객체의 쏠림 현상이 발생하였을 경우, 제안하는 동적 분산 그리드 기법이 검색 및 업데이트 성능 측면에서 DS-GRID보다 우수함을 입증한다.

키워드 : 동적 분산 그리드 기법, 공간 네트워크, 이동객체

Abstract Recently, a new distributed grid scheme, called DS-GRID(distributed S-GRID), has been proposed to manage the location information of moving objects in a spatial network[1]. However, because DS-GRID uses uniform grid cells, it cannot handle skewed data which frequently occur in the real application. To solve this problem, we propose a dynamic distributed grid scheme which splits a grid cell dynamically based on the density of moving objects. In addition, we propose a k-nearest neighbor processing algorithm for the proposed scheme. Finally, it is shown from the performance analysis that our scheme achieves better retrieval and update performance than the DS-GRID when the moving objects are skewed.

Key words : Dynamic distributed grid scheme, spatial network, moving object

1. 서 론

최근 GPS 및 무선 이동 컴퓨팅 기술의 발달로 인해, 텔레매틱스(telematics) 및 위치기반 서비스(LBS) 응용이 활발하게 연구되고 있다. 위치기반 서비스의 주요 대상이 되는 이동객체는 이상적인 유클리디언(Euclidean) 공간 대신 실제 도로나 철도와 같은 공간 네트워크(sptial network)를 이동하기 때문에, POI(points of interets) 및 이동객체의 좌표를 이용한 유클리디언 거리가 아닌, 공간 네트워크에서 두 점을 잇는 최단 거리인 실제 네트워크 거리를 계산하여 질의 처리를 수행해야 한다. 공간 네트워크 상의 이동객체를 위한 기존 질의처리 알고리즘은 네트워크 거리 계산 비용을 감소시키기 위해, 검색대상이 되는 POI 및 노드사이의 거리를 미리 계산하여 저장하는 pre-computation 기법을 이용하였다[2-4]. 그러나 [2], [3]의 연구는 POI의 위치정보가 변경될 경우 POI와 노드사이의 거리를 재계산해야 하는 단점을 지니며, [4]의 연구는 시간의 흐름에 따라 연속적으로 변경되는 이동객체의 대용량 위치정보를 효율적으로 관리할 수 없다는 단점을 지닌다. 이를 위해 대용량의 이동객체의 위치정보 관리 및 k-최근접 질의를 효율적으로 처리하기 위해 균일 크기의 2차원 그리드 셀을 이용한 분산 그리드 기법이 연구 되었다. 그러나 기존 분산 그리드 기법은 균일한 크기의 그리드 셀 구조를 이용하기 때문에, 실제 응용에서 빈번히 발생하는 이동객체의 쏠림 현상이 발생할 경우, 해당 영역을 담당하는 특정 서버에 부하가 집중되어 해당 서버에서

*본 논문은 교육과학기술부와 한국산업기술재단의 지역혁신 인력양성사업으로 수행된 연구결과임
**이 논문은 2009년도 정부(교육과학기술부)의 재원으로 한국과학재단의 지원을 받아 수행된 연구임(No. 2009-0059417)
· 이 논문은 2009 한국컴퓨터종합학술대회에서 '공간 네트워크에서 이동객체를 위한 동적 분산 그리드 기법'의 제목으로 발표된 논문을 확장한 것임

† 학생회원 : 전북대학교 컴퓨터공학과
yckim@dblab.chonbuk.ac.kr
sthong@dblab.chonbuk.ac.kr
kj@dblab.chonbuk.ac.kr
†† 종신회원 : 전북대학교 컴퓨터공학과 교수
jwchang@chonbuk.ac.kr
(Corresponding author임)

논문접수 : 2009년 8월 13일
심사완료 : 2009년 10월 12일

의 검색 성능 저하 및 서버 간 부하 불균형 현상이 발생한다. 따라서 이를 해결하기 위해, 이동객체의 밀도에 따라 그리드 셀을 분할하여 부하를 분산시키는 연구가 필수적이다. 이를 위해 본 논문에서는 이동객체의 쏠림 현상에 따라 그리드 셀을 동적으로 분할하여, 서버에 집중되는 부하를 효과적으로 분산시키기 위한 동적 분산 그리드 기법을 제안한다. 아울러 이를 위한 k-최근접 질의처리 알고리즘을 제안한다.

본 논문은 구성은 다음과 같다. 2장의 관련 연구에서는 공간 네트워크 상의 대용량 이동객체의 위치정보 관리를 위한 DS-GRID를 소개한다. 3장에서는 이동객체의 쏠림 현상을 효과적으로 관리하기 위한 동적 분산 그리드 기법을 설계하고, 4장에서는 이를 위한 k-최근접 질의처리 알고리즘을 제안한다. 5장에서는 제안하는 동적 분산 그리드 기법을 위한 k-최근접 질의처리 알고리즘의 성능 평가를 수행한다. 마지막으로, 6장에서는 본 논문의 결론 및 향후 연구 방향에 대해서 기술한다.

2. 관련 연구

DS-GRID는 대용량 이동객체의 위치정보를 분산 처리하기 위해 공간 네트워크를 2차원의 n*n 그리드 셀로 나눈다. 이에 따라 에지 및 노드와 같은 공간 네트워크 데이터를 셀 단위로 저장하고, 그리드 셀별로 POI 및 이동객체를 위한 색인 구조를 구성한다. 각 그리드 셀은 셀 테이블(Cell Table), 경계점 테이블(BP Table), POI R-트리, MO Index, Vertex-Edge, Vertex-Border, Cell-Border의 6개 컴포넌트로 구성되며, 서버마다 하나의 그리드 셀이 할당되어 각 셀 영역에 포함되는 공간 네트워크 정보를 독립적으로 관리한다. DS-GRID에서는 k-최근접 질의처리를 위해 ICE(Incremental Cell Expansion) 알고리즘과 MCE(Multicasting-based Cell Expansion) 알고리즘을 제안하였다. ICE 알고리즘은 질의점이 위치한 셀과 인접 셀이 공유하는 모든 경계점 정보를 한 번의 질의 전송을 통해 전달함으로써, 인접 셀에서의 POI 재탐색 횟수를 감소시켜 검색 성능을 향상시켰다. 그러나 ICE 알고리즘은 k 개의 POI를 검색할 때 점진적으로 셀을 확장하기 때문에, k 값의 증가 및 POI 밀도의 감소에 따라 검색 성능이 저하되는 현상이 발생한다. 이를 해결하기 위해 MCE 알고리즘은 k 개의 POI를 검색하기 위한 셀 리스트를 생성하여, 이웃 셀에서의 POI 탐색을 병렬적으로 수행하여 검색 성능을 향상시켰다. 그러나 DS-GRID는 공간 네트워크를 균등한 크기의 2차원 그리드 셀 구조로 분할하였기 때문에, 특정 지역에서 이동객체의 쏠림 현상이 발생할 경우, 해당 영역을 담당하는 서버에 부하가 집중되어 검색 성능 저하 및 서버 간 부하 불균형 현상이 발생하는 단점을 지닌다.

3. 동적 분산 그리드 기법

본 절에서는 DS-GRID에서 이동객체의 쏠림 현상에 의해 발생하는 검색 성능 저하 및 서버간 부하 불균형 현상을 해결하기 위해, DS-GRID를 확장하여 이동객체의 밀도에 따라 그리드 영역을 동적으로 분할하는 동적 분산 그리드 기법을 제안한다. 시간이 흐름에 따라 이동객체의 위치 변경이 특정 지역에 편중되어 하나의 서버에 부하가 집중되는 현상을 해결하기 위해서는, 이동객체의 분포도에 따라 균일한 수의 이동객체를 포함하는 영역으로 공간 네트워크를 분할해야 한다. 이를 위해서는 서버가 수용 가능한 이동객체의 수를 임계값(threshold)으로 설정하고 특정 영역에 임계값을 초과하는 이동객체가 존재할 경우, 해당 영역을 균등한 수의 이동객체를 수용할 수 있는 그리드 셀로 분할하는 방법이 요구된다. 그러나 DS-GRID는 각 그리드 셀 영역에 포함되는 각 노드 사이의 거리 및 노드와 경계점 사이의 거리를 미리 계산하여 저장한다. 따라서 영역의 분할이 발생하여 두 개의 새로운 셀이 생성될 경우, 각 셀 내에 포함되는 공간 네트워크 자체의 정보를 재계산해야하는 문제점이 존재한다. 이를 해결하기 위해서는 셀의 분할 시 요구되는 네트워크 정보의 재계산 양을 줄이는 방법이 필요하다. 이를 위해 제안하는 동적 분산 그리드 기법은 공간 네트워크를 일정한 크기의 2차원 그리드 셀로 미리 나누어 저장하고, 서버에 임계값 이상의 부하가 집중될 경우 미리 나누어진 셀 단위로 영역을 분할한다. 따라서 미리 나누어 저장된 최소 크기의 셀 영역에 부하가 집중될 경우에만, 해당 셀을 분할하고 네트워크 정보를 재계산한다. 그림 1은 제안하는 동적 분산 그리드 기법의 전체적인 시스템 구조를 나타낸다. 그림에서 서버의 임계값을 2라 가정하였을 때, 4 개의 서버에 공간 네트워크 영역이 할당된 상황을 나타낸다. 이때 공간 네트워크를 4*4의 기본 그리드 셀 단위로 미리 계산하여 저장한다고 가정한다. 서버 1에는 그리드 셀 1, 5, 9, 13의 4개 셀 영역이 할당되고, 서버 2에는 9개의 셀 영역이 할당됨을 나타낸다. 한편, 셀 3 영역에는 임계값 이상의 이동객체가 밀집되어, 해당 셀을 분할한다. 따라서 서버 3과 4는 각각 셀 2, 3과 셀 4, 17이 할당됨을 나타낸다. 이때 dispatcher 서버는 전체 공간 네트워크에 대해 서버별 영역 할당 정보를 저장 관리한다. 이러한 이유는 균일한 크기의 그리드를 사용하는 DS-GRID는 이동객체의 좌표를 이용하여 해당 영역을 관리하는 서버를 검색할 수 있지만, 동적 분산 그리드 기법은 그리드 영역이 동적으로 분할되어 할당되기 때문에, 해당 영역을 관리하는 서버를 검색하기 위한 정보가 필요하기 때문이다.

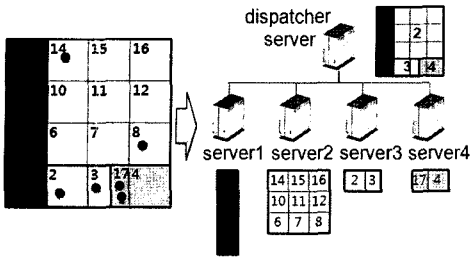

그림 1 동적 분산 그리드 기법의 시스템 구조

dispacher 서버는 이동객체의 위치 정보를 이용하여 해당 셀 관리 서버를 검색하기 위해, 2단계 그리드 구조의 그리드 셀 리스트를 관리한다. 그림 2는 2단계 그리드 구조를 이용한, 그림 1에 대한 그리드 셀 리스트를 나타낸다. 그림에서 CellID 3과 7을 포함하는 기본 그리드 셀 영역은 하나의 셀 영역에 이동객체의 쏠림현상이 발생하여 CellID 3과 17의 두 개의 셀로 분할된 후, 각각 서버 3, 4에 할당되어 관리됨을 나타낸다. 이를 통해 이동객체의 위치 정보를 이용하여 기본 그리드 셀 좌표를 계산하고, 해당 셀을 관리하는 서버를 검색할 수 있다.

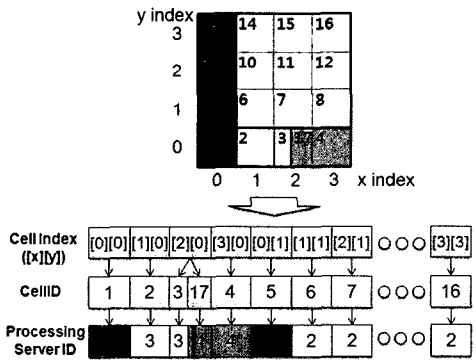

그림 2 2단계 그리드 구조를 이용한 그리드 셀 리스트

그리드 영역 분할은 해당 영역에 존재하는 이동객체의 수가 임계값을 초과할 경우 기본 그리드 셀 단위로 분할된다. 이상적인 그리드 영역 분할은 각 서버에 동일한 수의 이동객체가 할당이 되도록 영역을 분할하는 것이다. 그러나 이러한 영역 분할 조건은 기본 그리드 셀의 불필요한 분할로 네트워크 정보의 재계산 비용을 야기한다. 이를 위해 허용치 이내의 이동객체가 불균등하게 할당이 되도록 영역을 분할하기 위한 축을 다음과 같이 정의한다.

정책 1. 공간 네트워크가 n*n 개의 그리드 셀로 구성되고, 영역을 분할하기 위한 축 SplitAxis는 다음과 같다.

$$index_x=\{i|1<i<n, min(|\#\_MO_{leftregion}-\#\_MO_{rightregion}|)<\delta\}$$

$$index_y=\{i|1<i<n, min(|\#\_MO_{topregion}-\#\_MO_{bottomregion}|)<\delta\}$$

$$SplitAxis=\begin{cases} index_x & \text{if } |Area(left)-Area(right)| \\ & \quad <|Area(top)-Area(bottom)| \\ index_y & otherwise \end{cases}$$

이때, $Index_x$, Indexy는 각각 x, y축으로 그리드 영역을 분할할 때, 분할된 영역의 이동객체 수의 차이가 가장 작고, 허용치 δ값 이하인 셀 인덱스를 나타낸다. 아울러 Area(left), Area(right)은 $Index_x$를 기준으로 영역을 분할할 경우, Area(top), Area(bottom)은 $Index_y$를 기준으로 분할할 경우의 분할된 영역의 넓이를 나타낸다.

제안하는 셀 분할 기법은 영역 분할 후 각 서버에 할당될 이동객체 수의 차이가 허용치 δ값 이하일 경우, 셀의 분할 없이 기본 그리드 셀 단위로 영역을 분할하기 위한 기법이다. 그림 3은 셀 분할 기법을 이용한 그리드 영역 분할 알고리즘을 나타낸다. 첫째, 이동객체의 수의 차이가 가장 작게 나타나도록 분할하는 분할 인덱스를 x, y 축 방향에 대하여 각각 선택한다(1-2 라인). 둘째, x, y 축 방향의 분할 인덱스 중 분할 후의 영역의 차이가 보다 작은 축을 분할 인덱스로 선택한다(3 라인). 넷째, 선택된 분할 인덱스를 기준으로 분할될 두 영역의 이동객체 수의 차이가 임계값 δ 보다 클 경우, 추가로 셀을 분할하고, 분할된 셀에 대한 공간 네트워크 정보를 재 생성한다(4-8 라인). 마지막으로 그리드 셀 리스트에 분할된 영역 셀들의 담당 서버를 저장하고 생성된 영역을 반환한다.

```
Grid Region Split Algorithm(δ, Region)
newRegionA=∅, newRegionB=∅
01. Indexx=calculateIndex(numofCell, totalNumofMO, x)
02. Indexy=calculateIndex(numofCell, totalNumofMO, y)
03. splitIndex=compareRegion(Indexx, Indexy, Region)
04. splitRegion(splitIndex, newRegionA, newRegionB)
05. if ( diff(newRegionA, newRegionB)> δ )
06.    Cell = findCelltoSprit(newRegionA, newRegionB)
06.    SplitCells(RegionA, RegionB, Cell)
07.    regenerateNetworkData(RegionA)
08.    regenerateNetworkData(RegionB)
09. adjustGridCellList(RegionA, RegionB)
10. return newRegionA, newRegionB
```

그림 3 그리드 영역 분할 알고리즘

4. k-최근접 질의처리 알고리즘

본 절에서는 DS-GRID를 위한 두개의 k-최근접 질의처리 알고리즘 가운데 보다 우수한 성능을 나타내는 MCE 알고리즘을 확장하여 동적 분산 그리드 기법을 위한 k-최근접 질의처리 알고리즘인 D-MCE(Dynamic MCE) 알고리즘을 제안한다. DS-GRID를 위한 MCE 알고리즘은 각 셀로부터 일정 범위 안에 존재하는 이웃 셀의 Cell-Border 컴포넌트를 함께 저장한다. 이를 이용하여 k개의 POI를 검색하기 위해 방문할 셀 리스트를

생성하고, 셀 리스트에 포함된 셀을 관리하는 서버로 질의를 전달하여 외부 확장을 병렬적으로 수행한다. 그러나 동적 분산 그리드 기법에서는 영역을 동적으로 분할하기 때문에, 필요한 셀의 정보가 다른 서버에 존재하여 셀 리스트를 생성할 수 없는 경우가 발생한다. 이를 위해 D-MCE 알고리즘의 외부 확장 알고리즘은 질의를 전달받은 셀에서의 POI를 검색하기 전에 전달받은 경계점 정보를 이용하여 해당 서버에서 외부 확장을 위한 2차 셀 리스트를 생성한다.

D-MCE 알고리즘은 Cell-Border 컴포넌트를 이용하여 k 개의 POI를 검색할 때까지, 검색된 경계점을 공유하는 인접 셀에서 외부 확장을 반복적으로 수행한다. 이때, 외부 확장을 수행할 필요가 없는 인접 셀을 제거하기 위해 가지치기 정책을 다음과 같이 정의한다.

정책 2. 질의점과 검색된 k 번째 POI의 거리를 dmax, 외부 확장을 통해 검색된 경계점의 집합을 BPlist라 할 때, BPlist에 속한 임의의 경계점 중에서 다음 조건을 만족하는 경계점의 인접 셀들은 가치치기하여 외부확장 수행 대상에서 제외한다.

list of bp = {∀bp, bp∈BPlist and dist(Q, bpi)> dmax}

셀의 경계점은 에지의 두 노드가 다른 셀에 존재할 때 삽입되며, 두 노드 중앙에 위치한다. 따라서 k 번째 POI까지의 거리보다 먼 곳에 위치한 경계점이 존재할 경우, 이를 공유하는 인접 셀에 속한 POI는 당연히 k 번째 POI까지의 거리보다 더 먼 곳에 위치하기 때문에 해당 셀에서 외부확장을 수행할 필요가 없다.

그림 4는 D-MCE 알고리즘을 위한 외부 확장 알고리즘을 나타낸다. 첫째, 전달받은 인수인 flag가 true이면 외부 확장을 위한 2차 셀 리스트를 생성한다. 아울러 셀 리스트에 포함된 셀을 관리하는 프로세스로 질의를 전달하여 POI 검색을 병렬적으로 수행한다. 둘째, 외부 확장을 통해 검색된 POI를 Q_{dp}에, 경계점을 Q_v에 삽입하여 반환한다. 마지막으로 flag가 false일 경우, 해당 셀에서 외부 확장 알고리즘을 수행하고 결과를 반환한다.

```
DGRID-OuterExpansion Algorithm(q, k, BPlist, flag)
Qdp=∅, Qv=∅, Celllist=∅
01. if (flag==true)
02.     CreateCellList(q, k, Celllist)
03.     for each celli∈CellList
04.         sendQuery(celli, q, k, celli.BPlist)
05.     for each celli∈CellList
06.         getResult(celli, dplist, celli.BPlist)
07.         for each POIi∈dplist   Qdp.update(POIi)
08.         for each BPi∈BPlist    Qv.update(BPi)
09.     return POIs in Qdp
10. else
11.     return  OuterExpansion(q, k, BPlist)
```

그림 4 D-MCE 알고리즘을 위한 외부 확장 알고리즘

그림 5는 외부 확장 알고리즘을 이용한 D-MCE 알고리즘을 나타낸다. D-MCE 알고리즘은 질의를 전달하여 외부 확장을 수행하기 위한 셀 리스트 생성, 질의점이 위치한 셀 영역의 POI 검색을 위한 내부 확장 및 이웃 셀에서의 POI 검색을 위한 외부 확장의 세 단계로 구성된다. 첫째, k개의 POI를 검색하기 위해 방문해야할 셀 리스트를 생성하고, 해당 셀을 관리하는 서버로 질의를 전달하여 외부 확장을 수행한다(1-3 라인). 둘째, 내부 확장을 통해 자신의 셀에 포함된 POI를 검색한다(4 라인). 셋째, 외부 확장을 수행한 서버가 존재하면 해당 서버들로부터 검색된 POI를 전달받아 Q_{dp}에 삽입한다(6-9 라인). 넷째, 검색된 k 번째 POI 보다 작은 거리에 있는 경계점이 존재하면, 이를 이용하여 외부 확장을 수행할 새로운 셀 리스트를 생성하고 해당 경계점을 공유하는 이웃 셀에서 외부확장을 수행한다(10-14 라인). 마지막으로 검색된 결과를 사용자에게 반환한다.

```
D-MCE Algorithm(q, k, Query, Result)
Qv=∅, Qdp=∅, CellList=∅
01. CreateCellList(q, k, CellList)
02. for each celli∈CellList
03.     call OuterExpansion(q, k, celli.BPlist, true) in celli
04. InnerExpansion(q, k, Qv, Qdp, NULL)
05. while( CellList≠∅ )
06.     for each celli∈CellList
07.         integrateResult(celli, dplist, celli.BPlist)
08.         for each POIi∈dplist
09.             Qdp.update(POIi)
10.     dMax=Qdp.dist(k)
11.     for each vy∈celli.BPlist and dist(q,vy) < dMax
12.         CellList.InsertCell(vy)
13.     for each celli∈CellList
14.         call OuterExpansion(q, k, celli.BPlist, true) in celli
15. return POIs in Qdp
```

그림 5 D-MCE 알고리즘

5. 성능 분석

본 장에서는 제안하는 동적 분산 그리드 기법의 성능 평가를 수행한다. 성능 평가 환경은 HP ML 150 G3 서버, 인텔 Xeon 3.0 Ghz dual CPU, 2 GB 메모리, HP 250 GB SATA 7200 rpm HDD, BCM5703 Gigabyte Ethernet 이며, visual studio 2003을 이용하여 동적 분산 그리드 기법을 구현하였다. 성능 평가를 위한 공간 네트워크 데이터는 220,000 개의 에지와 170,000 개의 노드로 구성된 샌프란시스코 만 데이터를 이용하였으며, Brinkhoff 알고리즘[5]을 이용하여 1000개의 POI와 5000개의 이동객체를 생성하였다. 성능 비교 대상은 DS-GRID의 MCE 알고리즘과 제안하는 동적 분산 그리드 기법의 D-MCE 알고리즘이며, 성능 평가는 이동객체의 업데이트가 발생할 경우의 검색 성능 및 이동객

체의 위치 변화에 따른 업데이트 성능 비교를 수행한다.

첫째, 그림 6은 기본 그리드 셀의 개수가 50*50 일 때, 전체 이동객체 중에서 이동한 객체의 비율에 따라 업데이트가 발생할 경우의 검색 성능을 나타낸다. 그림에서 DS-GRID는 셀의 개수가 20*20 이상으로 증가할수록 검색시간이 선형적으로 증가한다. 이는 이동 객체의 셀의 크기가 작아질수록 쏠림 현상에 따른 업데이트 시간이 증가하기 때문이다. 그러나 제안하는 동적 분산 그리드 기법은 그리드 셀의 개수가 증가할수록 성능이 향상됨을 나타낸다. 이는 기본 셀의 크기가 작을수록 이동객체의 업데이트에 따른 셀 분할 횟수가 적어지기 때문에, 네트워크 데이터의 재계산 시간이 감소하기 때문이다.

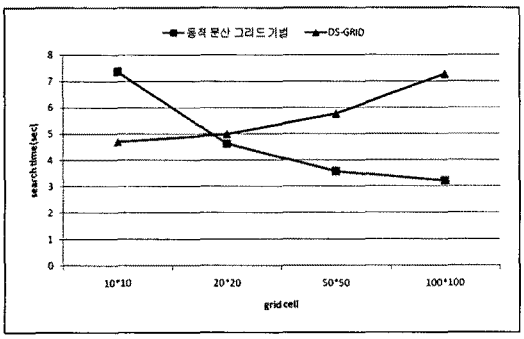

그림 6 이동 객체 업데이트 후의 검색 성능

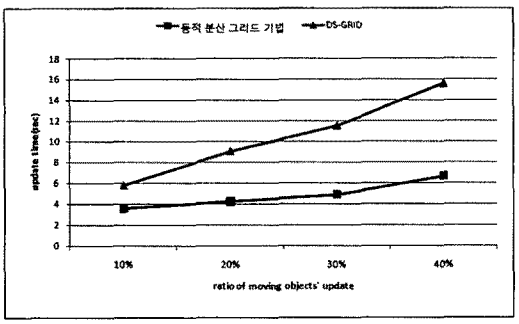

그림 7 이동 객체 업데이트 성능

둘째, 그림 7은 전체 이동객체 중 10%~40%의 이동객체에 대한 업데이트 성능을 나타낸다. 두 방법 모두 업데이트된 이동객체의 비율이 증가할수록 업데이트 시간이 선형적으로 증가한다. DS-GRID는 하나의 서버에 쏠림현상이 빈번하게 발생하기 때문에, 업데이트 시간의 증가 정도가 큰 반면, 제안하는 D-MCE 알고리즘은 동적 분할을 통해 각 서버에 집중되는 부하를 효과적으로 분산처리하기 때문에, 업데이트 시간의 증가 정도가 매우 완만함을 나타낸다. 전체 이동객체 중 업데이트된 이동객체의 비율이 10%, 20%, 30%, 40%일 때, D-MCE 알고리즘이 각각 37%, 53%, 57%, 57% 검색 성능이 우수함을 나타내었다. 제안하는 동적 분산 그리드 기법은 이동객체의 위치변경이 발생하지 않는 정적인 환경에서는 DS-GRID와 유사한 검색 성능을 나타내지만, 이동객체의 위치변화로 인해 데이터의 쏠림 현상이 발생하는 동적인 환경에서는 제안하는 기법이 매우 효율적임을 알 수 있다.

6. 결론 및 향후 연구

본 논문에서는 실제 응용에서 빈번히 발생하는 데이터 쏠림 현상(skewed data)을 효과적으로 지원하기 위해, DS-GRID를 확장한 동적 분산 그리드 기법을 제안하였다. 동적 분산 그리드 기법은 데이터의 쏠림 현상에 따라 그리드 셀을 동적으로 분할함으로써 서버에 집중되는 부하를 효과적으로 분산시킨다. 또한 셀 분할 지연 기법을 이용하여, 셀 분할에 따른 네트워크 데이터 재계산의 오버헤드를 감소시켰다. 아울러 제안하는 동적 분산 그리드 기법를 위한 D-MCE 알고리즘을 제안하였다. 성능평가를 통해 이동객체의 위치변화로 인해 데이터의 쏠림 현상이 발생하였을 경우, 제안하는 동적 분산 그리드 기법이 보다 우수한 검색 성능을 나타내었다.

향후연구로는 제안하는 동적 분산 그리드 기법을 실제 위치기반 서비스에 적용하여 제안하는 기법의 효율성을 입증하는 것이다.

참 고 문 헌

[1] Y.C. Kim, Y.J. Kim, J.W. Chang, "Distributed Grid Scheme using S-GRID for Location Information Management of a Large Number of Moving Objects," *Journal of Korea Spatial Information System Society*, vol.10, no.4, pp.11-19, Dec. 2008.

[2] M. Kolahdouzan, C. Shahabi, "Voronoi-based Nearest Neighbor Search for Spatial Network Databases," In *Proc. of 13th International Conference on Very Large Databases*, pp.840-851, 2004.

[3] X. Huang, C.S. Jensen, S. Saltenis, "The Islands Approach to Nearest Neighbor Querying in Spatial Networks," In *Proc. of 9th International Symposium on Advances in Spatial and Temporal Databases*, LNCS 3633, pp.73-90, 2005.

[4] X. Huang, C.S. Jensen, H. Lu, S. Saltenis, "S-GRID: A Versatile Approach to Efficient Query Processing in Spatial Networks," In *Proc. of 10th International Symposium on Advances in Spatial and Temporal Databases, LNCS 4605*, pp.93-111, 2007.

[5] T. Brinkhoff, "A Framework for Generating Network-Based Moving Objects," In *Proc. of Geo-Informatica*, pp.153-180, 2002.