분산 이동객체 데이터베이스를 위한 k-NN질의 처리

한종형[0] 이준우, 나연묵
단국대학교 전자컴퓨터공학과
{jhhan[0],jwlee,ymnah}@dblab.dankook.ac.kr

k-NN Query Processing for Distributed Moving Object Databases

Jonghyeong Han[0] , Joonwoo Lee, Yunmook Nah
Dept. of Electronics & Computer Engineering, Dankook University

요 약

GIS분야와 유비쿼터스 환경의 진보로 언제 어디서나 유무선으로 정보를 주고 받는 환경의 계선에 대한 발전이 계속 되어 왔다. 이런 환경에서 이동객체의 이용도가 증대됨에 따라 대용량의 객체 처리를 위해 분산 처리방식이 적용 되었다. 기존 연구의 k-NN질의는 단일 노드에서 질의 처리 비용의 절감에 중점을 두어 분할된 노드에서의 질의처리에 관련된 연구가 부족하였다. 분할된 노드에서 질의를 처리하기 위해서 고비용이 요구되는 k-NN질의를 위하여 본 논문에서는 Hybrid k-NN질의처리 방식을 제안한다. 제안방식은 k-NN질의와 범위질의 특성을 결합한 형태로 분할된 노드에 질의처리를 가능하게 하고, 질의처리 시 k-NN질의와 범위질의의 혼합으로 k-NN질의의 고비용을 절감하는 방법이다. 이 방법은 GALIS 프로토타입의 SLDS의 질의 처리 부분을 개선에 활용할 수 있다.

1. 서 론

모바일, 유비쿼터스 환경등의 다변화를 추구하는 현재의 GIS분야는 언제 어디서나 GPS로 위치 측위기술과 무선통신 기술의 발전으로 인해 이동 단말기의 대중화로 위치 기반 서비스(LBS: Location Based Service)에 대한 관심이 급증하고 있다. 대부분의 연구가 다중 노드를 대상으로 휴대폰 사용자 위치 추적 같은 위치 정보 갱신이 빈번하면서 최소 백만 단위 이상의 대용량 이동 객체를 처리하고 있다.

단일노드일 때의 문제를 해결하기 위하여 제안된 GALIS(Gracefully Aging Location Information System) 아키텍쳐는 클러스터 기반 분산 컴퓨팅 구조로 설계되어 각 지역별 데이타를 여러 노드에 저장함으로써 위치 정보의 저장과 갱신 및 질의에 대한 부하를 분산시켜 대용량의 데이터를 처리할 수 있게 되었다[1,4,5].

다중 노드로 구성된 노드내의 현재객체 정보는 R트리로 저장 관리한다[6,7]. 저장된 객체에 대해서 k-NN질의 및 객체 위치 질의를 할 수 있다[8,9,10,11]. 지역적으로 분할된 노드들의 k-NN(k-Nearest Neighbor)질의에 대해서 처리 비용 감소에 대해서 개선방안을 제시한다.

k-NN질의처리 시 질의대상이 되는 모든 노드에 대한 질의를 하는 방법과 k-NN질의와 범위(Range)질의를 혼합하여 질의를 하는 Hybrid k-NN질의 방법을 제시한다.

GALIS의 구조에 대해 2장에서 설명하고, 3장에서는 분산 환경에서의 k-NN질의를 개선한 Hybrid k-NN질의를 제시하며, 4장에서는 본 논문에서 제안한 방법의 효율성을 실험을 통해 증명하고, 끝으로 5장에서는 결론 및 향후 연구 대해 제시한다.

2. 관련연구

2.1 GALIS의 구조

GALIS의 전체적인 구조는 그림 1과 같다.

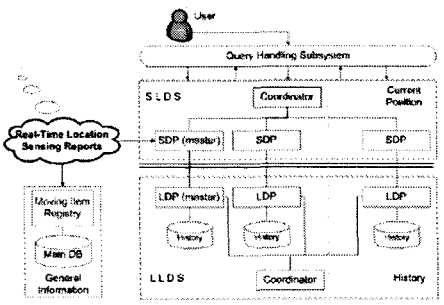

그림 1 GALIS 전체구조

Moving Item Registry는 main DB에 이동 객체의 프로필을 등록하게 된다. GALIS는 크게 객체의 현재 위치 정보를 처리하는 SLDS와 과거 위치 정보를 처리하는 LLDS(Long-term Location Data Subsystem)으로 나눌 수 있다[2,3]. SLDS는 메인 메모리 데이타베이스를 사용함으로써 디스크기반 데이타베이스에 비해 빠른 처리 시간으로 대용량 객체의 빈번한 위치정보 갱신에 대응할 수 있다.

SLDS내의 각 노드를 SDP(Short-term Data Processor)라고 한다. SDP는 macro-cell로 명명된 일정 지역 내의 객체의 정보를 담당하며. 다수의 SDP가 다른 객체들의 현재 위치 정보를 갱신하기 위해 동시에 동작하게 된다. SDP master는 실시간으로 측위된 이동 객체의 위치 정보를 받아 다른 SDP worker 및 LDP(Long-term Data Processor) master로 전달하는 역할을 한다. 각각의 SDP, LDP는 지역적으로 분할된 노드의 객체 정보를 유지한다.

coordinator node는 각 노드에 할당된 객체 수를 모니터 하여 한 노드에 객체가 편중되지 않도록 노드들을 분할하거나 합병함으로써 동적 부하 균형을 가능하도록 한다.

3.k-NN질의처리

k-NN질의는 질의 위치로부터 k값 만큼의 최 근접 객체를 검출하여 결과로 반환하는 질의이다. k-NN질의는 k값을 찾기 위해 k값보다 많은 후보 객체를 선출하여 순위를 결정짓고 순위에 대해 k값의 객체를 반환한다.

분산 환경에서의 k-NN질의 시 고려될 점으로 인접 노드에 대한 질의 형태는 그림 2와 같다.

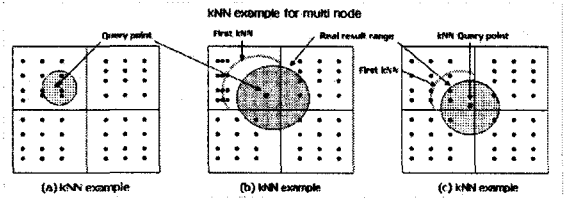

그림 2 다중노드에 대한 k-NN질의 예

단일 노드에 대한 질의는 선출된 객체를 결과로 반환하면 된다. 그러나 지역적으로 연결되어 있는 다중 노드의 경우 인접 노드 내에 존재하는 객체의 위치도 고려해야 한다.

그림 2(a)는 다중 노드에 대해 k-NN질의의 예이다. 질의 결과로 단일 노드에 질의를 한 것과 동일하며, 질의대상 노드에서 k-NN질의 객체를 모두 검출한 형태이다. 그림 2(b)는 질의 대상 노드(좌측상단)에서 질의 객체를 검출하였다. 격자 무늬의 외각에 위치한 반 호의 범위가 처음 k-NN질의를 하여 검출된 객체가 포함된 범위이고, 격자 무늬범위에 해당하는 객체가 최종결과로 검출된 객체이다. k-NN질의 좌표가 해당하는 노드의 객체보다 인접 노드의 객체가 더 가까워 결과적으로 k-NN질의 노드의 객체는 제외되고 인접 노드의 객체가 검출된 형태이다. 그림 2(c)의 각 노드는 객체가 고르게 분포되어 있다. k-NN질의처리 후 격자무늬내의 객체가 최종 결과이며, 각 노드의 객체가 균등하게 검출된 형태이다.

3.1 질의 처리 과정

k-NN질의 처리 과정은 그림 3과 같다. Tree creator로 노드의 객체를 이용하여 인덱스 구조를 형성하고, 객체의 삽입 삭제를 한다.

질의를 할 노드는 Query Analyzer를 통하여 질의 대상이 되는 노드를 찾아서 질의를 처리한다. 처리된 k-NN질의 결과값 중 k번째의 객체의 위치와 k-NN질의 위치로 인접 노드에 대해 질의처리 여부를 Query Checker를 이용하여 질의할 노드를 판단한다.

질의대상 노드가 존재 하지 않을 때는 최초결과 값을 반환하고, 질의 여부가 결정되면 Query Creator를 이용하여 질의를 생성한다. 생성된 질의로 질의 대상 노드에 질의를 하고, 질의 결과를 이용하여 R트리를 새로 만든 후에 트리에 최종 k-NN질의를 한다. 최종 k-NN질의는 최초 질의 시 사용된 k-NN질의를 이용한다.

최종 질의 처리 후 반환된 결과는 지역적 분할이 되지 않는 노드에 k-NN질의를 한 것과 동일한 결과를 얻게 된다.

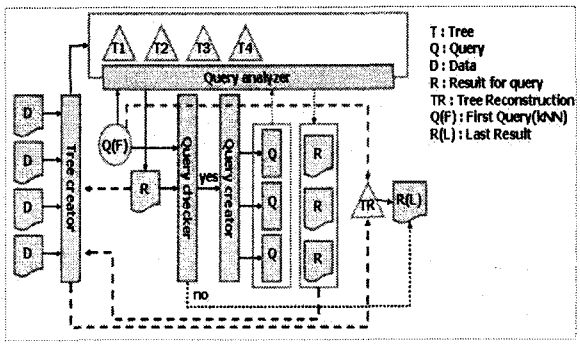

그림 3 다중 노드에 대한 k-NN질의 처리 과정

3.2 인접 노드에 대한 질의 여부 판단

분산 노드에 k-NN질의 시 인접 노드에 대해 질의여부 판단이 고려되어야 한다. 그림 2(b), 2(c) 와 같이 처음 질의 대상 노드 외에 인접 노드의 객체를 검사하여 정확한 최종 결과를 반환해야 한다.

그림 4(a)의 인접 노드(E, S)에 질의 여부판단을 위해 처음 k-NN질의를 마친 후 결과값 k의 객체와 k-NN질의 좌표를 이용하여 질의여부를 확인한다.

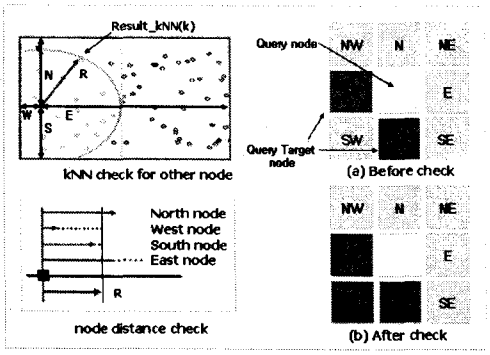

그림 4 질의 대상 노드 확인

처음 질의 대상이 되었던 노드에 질의를 마치고, k-NN질의와 결과 K값의 좌표간의 거리 R(radius)을 계산한다. 그림 4(좌측)은 먼저 계산되어 있는 R값과 노드의 각 방향에 대한 경계선과의 거리를 비교한다. R값과의 비교대상인 N, S, E, W값을 비교하여 길이가 R보다 작으면 질의 대상 노드로 선정하고 질의를 한다.

비교 과정에서 대각선에 위치해 있는 노드(그림 4(a)의 SE)에 대해서는 대각선 인접 노드와 처음 질의 노드의 인근 2개의 노드(E, S)에 대한 확인과정에서 그림 4(a)의 E, S 노드가 질의 대상이 되었다면, 그림 4(b)의 대각선에 위치해 있는 노드(SE)는 질의 대상이 된다.

3.3 Hybrid k-NN질의

다중 노드에 대해 k-NN질의는 질의대상 노드에 대한 모든 질의 처리 후 결과를 통합한다. 통합된 결과에 대해서 다시 한번 k-NN질의를 수행하여 결과를 반환한다. 노드 내에 포함되는 객체 수에 따라 동일한 k값의 NN질의 처리 시간은 객체 수에 비례한다. 이러한 고 비용의 처리시간을 감소시키기 위해서 본 논문에서는 제안한 방법은 그림 5(c)에서 보듯이 k-NN질의 대상 노드 외에 나머지 노드에 대해서는 k-NN질의 대신 범위 질의를 수행하는 방법을 제시한다.

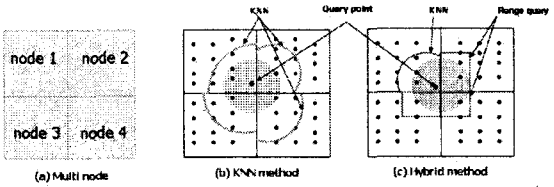

그림 5 다중 노드에 대한 질의 처리 방법

그림 5(b)는 질의 대상이 되는 노드 1, 2, 3, 4에 대해 k-NN질의를 한 것이다. 그림 5(c)는 노드 1에 k-NN질의를 하고 나머지 노드 2, 3, 4는 범위질의를 한 것이다. k-NN질의는 특성상 후보자 선출과 후보자중 k값의 객체를 선출에 고 비용이 소요된다.

범위 질의는 질의 범위에 대해서 처리시간이 증가하는 특성을 띤다. 다른 노드에 대한 범위질의의 영역을 그림 6의 영역으로 제한하기 위해서 그림 4에서 계산되었던 R값을 기준으로 질의범위를 제한한다.

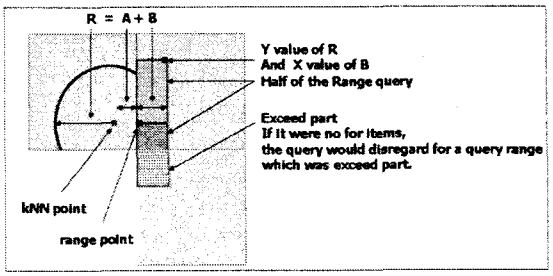

그림 6 질의대상 노드에 대한 질의범위 설정

R값의 길이는 A+B값과 같다. 질의대상 노드의 질의위치는 k-NN질의 좌표로부터 인접 노드와 가장 가까운 위치를 범위질의의 중심점으로 선택한다. 이때 방향 값을 제외시킨 R값의 X, Y좌표 값은 범위질의의 중심점으로부터 사각형을 이루는 X 또는 Y좌표로 사용한다. B값은 범위질의의 두께가 되며 B값의 X, Y값은 R값에서 참조되지 않은 X 또는 Y값으로 사용되어 질의 범위의 사각형을 만들게 된다. R(x, y)와 B(x, y)값은 질의 대상 노드의 위치(상하, 좌우)에 따라 좌표 점을 선택하여 사용한다.

범위질의로 만들어진 사각형의 질의는 실제로 처리해야 할 질의의 절반을 이루기 때문에 동일한 크기의 범위를 더해서 실제질의 크기의 범위를 형성한다.

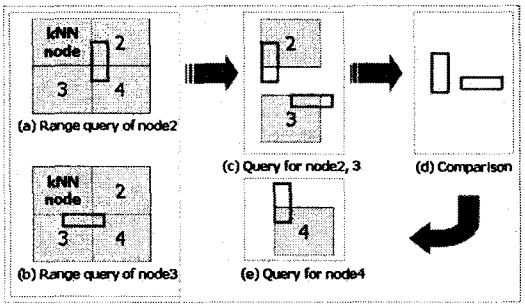

그림 7 범위질의 사용

생성된 범위질의로 질의대상 노드에 질의를 한다. 질의 노드로 부터 대각선에 위치한 그림 7(e)노드에 대해서 별도의 질의를 만들지 않고 범위질의의 크기(B값)가 큰 범위질의를 사용한다.

4. 실험 및 결과

다중 노드에 k-NN질의를 할 때, 질의대상 노드에 대해서 각각의 질의처리 시간을 비교한다. k-NN질의 점 위치는 4개의 노드로 구성된 노드에 모두 질의될 수 있도록 최초 질의 노드(좌측상단)의 우측하단 경계선 부근으로 위치 한다. 실험은 Fedora core 4 Linux환경에서 CPU 3.0GB, 메모리 512MB의 단일 노드로 실험을 하였다. 실험의 편의를 위해 단일 프로세서에 네 개의 노드를 구성하였다.

비교 실험을 위해 4개의 노드를 R트리로 구성하였으며, 노드별로 4천, 4만, 8만, 40만 개의 이동 객체를 네 개의 노드에 균등하게 분포시킨 4개의 노드를 구성하였다.

4.1 k-NN질의

그림 8은 분할된 4개의 노드에 각각 k-NN질의를 하였을 때 나타난 결과의 그래프이다. 분할된 노드는 균등한 객체 수를 가지고 있어 처리시간이 고른 분포를 보이는 것을 알 수 있다.

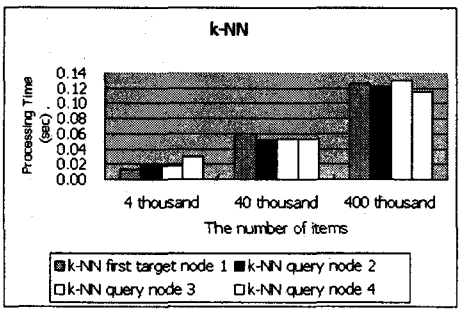

그림 8 전체 노드에 k-NN질의처리 시간

k-NN질의는 동일한 k값의 질의를 하였을 때, 객체 수에 비례하여 처리 시간이 증가하는 것을 보인다. k-NN질의 처리시 후보자를 검출하고, 검출된 후보자중 k값에 해당하는 객체를 선출하기 때문에 객체의 비교처리 시간이 많이 소요된다. 만약 실제 상황에서 4개의 노드로 구성된 시스템에서 이동 객체가 전체 객체 수는 동일하나 분포 상태는 고르게 나타내지 않을 때 객체의 처리시간이 다르게 나올 수 있고, 객체의 분포와 객체 수에 따라 처리비용이 높아질 수 있다.

4.2 Hybrid k-NN질의 실험

그림 9은 처음 질의 노드에 대해서는 k-NN질의를 하였고, 나머지 질의 대상이 된 노드에 대해서는 범위 질의를 수행한 실험결과를 그래프로 표현한 것이다

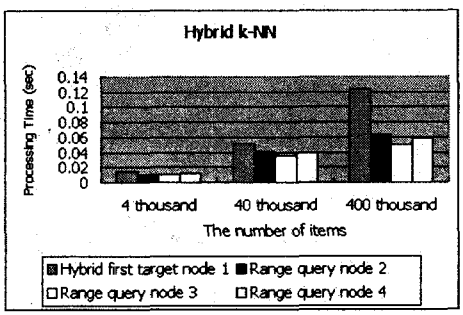

그림 9 Hybrid k-NN질의를 이용한 노드별 질의처리시간

앞에서 수행한 k-NN질의만을 수행한 실험보다 Hybrid k-NN질의는 질의처리 시간이 단축된 것을 실험을 통해 살펴볼 수 있다.

또한 Hybrid k-NN질의를 이용한 실험에서 최종 결과로 반환받은 결과 객체와 k-NN질의만을 수행한 실험의 최종 반환 결과 객체와 상호 비교에서 일치된 결과를 보였다.

4.3 실험 비교

본 논문에서 제시한 Hybrid k-NN질의의 효용성을 증명하기 위해 각각의 노드에 이동객체를 2만개씩 총 8만개의 객체를 갖는 분산 노드에서 발생하는 질의처리 시간을 측정하고 그림 10의 그래프로 표현하였다.

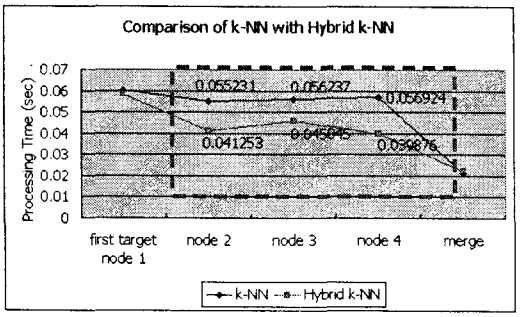

그림 10 k-NN질의 비교

처음 질의는 두 가지 모두 k-NN질의를 수행하므로 유사한 처리 시간을 보였으며, 중앙의 점선 사각형내의 위쪽 그래프는 k-NN질의 처리 시간이고 아래 그래프는 범위질의 처리 시간이다. Hybrid k-NN질의를 이용해 범위 질의를 수행한 노드에서 보다 빠르게 질의를 수행하게 된다.

그림 10의 오른쪽 merge 부분은 처리된 결과를 병합하여 R트리의 재구성 시간과 구성된 트리에 k-NN질의를 하여 최종 결과를 처리한 시간이다.

본 논문에서 제시한 개선 방법의 효율을 증명하기 위해 병렬처리 수행시간을 측정하였다. 그림 11의 누적 그래프를 통해 제시한 방법의 효용성을 다시 한번 확인할 수 있다.

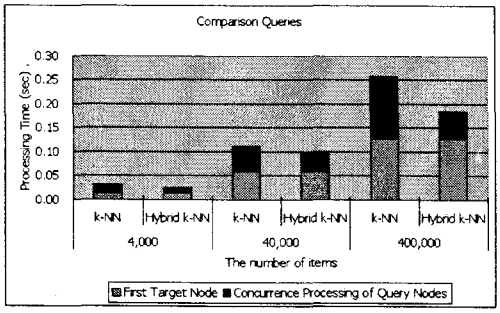

그림 11k-NN질의 전체 수행 시간 비교

5. 결론 및 향후 연구

본 논문에서는 GALIS 기반의 SDP의 지역 분할된 노드에 대한 k-NN질의를 제안하였다. 첫 번째 k-NN질의의 대상이 된 노드에 k-NN질의를 하고, 나머지 노드 중 질의 대상이 된 인접 노드에 대해서 k-NN질의를 하는 방법과 범위질의를 하는 방법이다. 두 가지 질의 방법을 상호 비교 실험을 통하여 분할된 노드에 대한 질의 성능이 범위 질의를 혼합한 형태가 k-NN질의만을 한 형태보다 효과적임을 증명 하였다. k-NN질의 특성과 범위질의 특성을 조합한 질의가 질의특성 별로 단점을 보완하여 보다 나은 질의 성능을 나타냈다.

향후 연구과제로 분산 환경에서의 효율적인 질의를 수행하는 인덱스 구조 연구와 여러 형태의 질의를 개선하는 연구가 필요하다.

참고문헌

[1] Yunmook Nah, K.H. (Kane) Kim, Taehyung Wang, MoonHae Kim, Jonghoon Lee, Young Kyu Yang, "GALIS: A Cluster-based Scalable Architecture for Location-Based Service Systems," Database Research, 18(4), KISS SIGDB, December 2002, pp.66-80.

[2] Yunmook Nah, K.H.(Kane) Kim, Taehyung Wang, MoonHae Kim, Jonghoon Lee, Young Kyu Yang, "A Clusterbased TMO-structured Scalable Approach for Location Information Systems," in Proc. WORDS 2003 Fall, IEEE CS Press, October 2003, pp.225-233.

[3] Moon Hae Kim, K.H.(Kane) Kim, Yunmook Nah, Joonwoo Lee, Taehyung Wang, Jonghoon Lee, Young Kyu Yang, "Distributed Adaptive Architecture for Managing Large Volumes of Moving Items," IDPT-Vol.2, 2003, pp.737-744.

[4] S. Saltenis, C. S. Jensen, S. T. Leutenegger, and M. A. Lopez, "Indexing the Positions of continuously Moving Objects," SIGMOD, 2000, pp.331-342.

[5] Zhang, j., Zhu, m., Papadias, D., Tao, Y., Lee, D, "Location-Based Spatial Queries. To appear in proceedings of ACM conference on Managerment of Data," SIGMOD, June 9-12, 2003, pp 467-478.

[6] A. Guttman, "R-Trees: A Dynamic Index Structure for Spatial Searching," ACM SIGMOD, 1984, pp.47-57.

[7] N. Beckmann, H. Kriegel, R.Schneider, and B.Seeger, "The R*-Tree : An Efficient and Robust Access Method for Points and Rectangles," ACM SIGMOD, 1990, pp.322-331.

[8] Gisli R. Hjaltason, Hanan Samet, "Ranking in Spatial Databases," 4th Symposium on Spatial Databases, Portland, 1995, pp.83-95.

[9] King Lum Cheung, Ada Wai-chee Fu, "Enhanced nearest Neighbor search on the R-Tree," SIGMOD, 1998, pp.16-21.

[10] G.S.Iwerks, H.Samet and K.Smith, "Continuous k-nearest neighbor queries for continuously moving points with updates," VLDB, 2003.

[11] 박원순, 전세길, 나연묵, "대용량 이동 객체의 위치 정보 인덱싱," 한국정보과학회 2002 추계 학술발표 논문집(I), 29(2), pp.49-51.

[12] 이준우, 전세길, 나연묵, "TMO 기반 분산 이동 객체 데이타베이스의 설계 및 구현," 한국정보과학회 2003 춘계 학술발표 논문집, VOL.30 NO.01, pp.764-766.