영역 질의의 효율적 처리를 위한 버퍼링 기법†

김상우° 이준우 전세길* 나연묵
단국대학교 전자컴퓨터공학과, 도로교통기술원*
{swkim°, jwlee}@dblab.dankook.ac.kr, sgjeon@freeway.co.kr*, ymnah@dku.edu

Buffering Strategy for Efficient Processing of Range Queries

Sangwoo Kim°, Joonwoo Lee, Segil Jeon*, Yunmook Nah
Dept. of Electronics and Computer Engineering, Dankook University
Highway & Transportation Technology Institute*

요 약

휴대전화, PDA, GPS 등 모바일 기기들의 발전과 보급으로 인하여 위치 기반 서비스에 대한 관심이 크게 증가하고 있다. 휴대폰 사용자 위치 추적과 같은 대용량의 객체를 처리하기 위해서 기존의 단일 노드 기반 시스템으로는 어려움이 있어, 클러스터 기반 분산 컴퓨팅 구조로 GALIS 아키텍처가 제안되었다. 본 논문에서 제안하는 질의 처리 성능 개선을 위한 버퍼링 기법은 GALIS 의 질의 처리 서브 시스템 성능 향상을 위해 질의 처리 결과를 버퍼링하고 연속된 질의 처리시 발생할 수 있는 중첩된 질의 영역을 관리하게 설계 하였다. 버퍼링 기법을 통해 수행되는 질의는 중첩된 질의 영역을 제외한 부분만을 수행하기 때문에 결과 셋의 크기를 줄여주는 역할을 하고, 질의 처리 시간 감소에 큰 영향을 미친다.

1. 서론

최근의 GPS 로 대표되는 위치 측위 기술과 무선 통신 기술의 비약적인 발전, 그리고 이동 단말기의 대중화로 인하여 위치 기반 서비스(LBS: Location Based Service)에 대한 관심이 크게 증가하고 있다.

GALIS(Gracefully Aging Location Information System) 아키텍처는 클러스터 기반 분산 컴퓨팅 구조로 설계되어 각 지역별 데이터를 여러 노드에 저장함으로써 위치 정보의 저장과 갱신 및 질의에 대한 부하를 분산 시켜 대용량의 데이터를 처리할 수 있다[1,2,3].

GALIS 아키텍처를 기반으로 한 효율적인 범위 질의 처리 기법으로는 비균등 2 단계 고정 그리드 인덱싱하에서 시공간 필터링을 이용해 질의 처리 시간을 줄이는 연구가 있으며, 질의 큐에 적재되어 있는 질의들 간의 시간과 공간적 관련성을 이용하여 처리 순서를 재배치하여 질의 처리 효율을 높이는 시공간 질의 스케쥴링 기법이 있다[4,5]. 시공간 필터링의 경우 필터링을 위해 시공간 데이터 삽입 시 별도의 처리 과정이 필요하며, 시공간 질의 스케쥴링은 실시간 질의 처리에 활용하는데 문제점이 있다. 데이터 스트림 관리 시스템(DSMS: Data Stream Management System)에서 질의 처리 시간 감소를 위한 OSCAR(Overload-sensitive Stream Capture and Archive Reduction) 접근 방법은 현재데이터를 과거데이터로 관리할 때 질의 처리 성능을 개선하기 위해 데이터의 크기를 무작위로 줄이는 문제점이 있다[6]. 연속적으로 동시 발생하는 이동 객체에 대한 질의들을 모아서 처리하여 중복된 질의를 피하는 SINA(Scalable INcremental hash-based Algorithm)알고리즘이 있다[7]. 질의 연산의 결과를 활용하는 기법으로는 하드웨어 레벨에서 그래픽 카드에 있는 공간질의에 대한 결과를 활용한 연구와 질의하지 않는 영역에 대한 처리를 피하고 중복된 질의 영역을 공유하는 precision sharing 기법은 영역 질의를 위한 개선이 필요하다[8,9]. 캐시 메모리 관리를 통한 실시간 트랜잭션 처리기법 STEPS(Synchronized Transactions through Explicit Processor Scheduling)를 도입하여 성능 향상을 시킬 수 있다[10].

본 논문에서는 클러스터 분산 컴퓨팅 구조로 제안된 GALIS 아키텍처의 질의 처리 서브시스템의 성능 향상을 위한 버퍼링 기법을 이용한 질의 처리 기법을 제안한다.

제안된 버퍼링 기법은 데이터베이스의 질의 처리 수행시간 중 디스크 접근 시간을 줄여주는 역할을 해주어 GALIS 에서 발생될 수 있는 질의를 보다 효율적으로 할 수 있게 한다.

본 논문의 구성은 다음과 같다. 2 장에서 영역 질의의 버퍼링 기법을 이용 한 질의 처리 기법을 제안하고, 3 장에서 질의 처리 성능 실험 과정과 결과를 보여주며, 4 장에서는 결론을 맺는다.

2. 영역 질의의 버퍼링 기법

이동객체 정보를 관리하는 시스템에는 관심 있는 영역에 집중된 질의 요청이 지속적으로 이루어진다. 본 논문에서 제안한 버퍼링 기법은 이동객체의 정보를 관리하는 시스템에 여러 개의 영역 질의가 수행될

† 본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT 연구센터 육성지원사업의 연구결과로 수행되었음

때 발생될 수 있는 질의 영역간 중첩되는 영역을 버퍼링을 이용하여 관리하는 기법이며, 이를 통해 질의 처리 성능을 향상 시킬 수 있다.

그림 1 의 영역 ①,②같은 영역 질의가 요청 된다고 가정하면, 두 질의는 중첩되는 부분이 발생하게 된다. 버퍼링 기법은 먼저 요청된 질의 영역 ①에 대한 질의를 먼저 수행하고, 결과는 메모리에 버퍼링 시킨다.

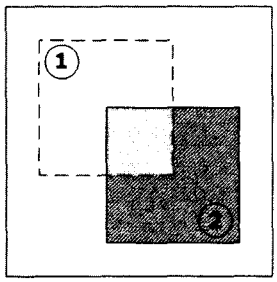


그림 1. 영역 질의 예제

이후에 수행되는 질의 영역 ②에 대한 질의를 수행할 때는 이전 질의 영역 ①과 중첩이 되는지 판단하고 중첩되는 영역의 비율이 버퍼링 기법의 수행 판단 계수보다 큰지 판단한다. 두 조건에 만족할 경우 중첩되지 않는 영역을 시스템에 질의하고 중첩 되는 영역에 대한 질의는 버퍼링 된 메모리에서 구해서 두 질의 처리 결과를 합한다. 마지막으로 다음 영역 질의를 위해 시스템으로부터 구한 결과를 버퍼링 해둔다(그림 2).

```
procedure ImprovedQuery(RANGE cQ)
// cQ : current Query
// pQ : previous Query
// cQ separate bQ(buffer) and iQ(improved)
// Cr : coefficient of overlapped region ratio
begin
  if (Overlap(pQ,cQ) is true) then
    if (GetOverlappedRegion(cQ) > Cr) then
      SeparateQuery(cQ, bQ, iQ);
      resultImp = QueryToDB(iQ);
      resultBuf = QueryToBuf(bQ);
      resultSet = resultImp + resultBuf;
      StoreResultToBuffer(resultImp);
      return
    endif
  endif
  // curQry does not intersect with Buffer
  resultSet = QueryToDB(cQ);
  StroreResultToBuffer(resultSet);
end
```

그림 2. 버퍼링 기법의 의사 코드

이와 같이 버퍼링 기법을 통해 수행되는 질의들은 시스템에 전달할 질의 영역의 크기를 줄여주고, 반환해야 하는 질의 결과 셋의 크기를 감소시켜 디스크 접근 시간을 줄여주게 된다.

일반적으로 그림 1 의 영역 ②에 대한 질의는 그림 3 과 같이 수행되었으나, 제안한 버퍼링 기법은 그림 4 와 같이 질의 영역을 분할하여 버퍼링된 메모리로부터 구할 수 있는 질의 영역(bQ)과 시스템으로부터 구할 수 있는 질의 영역(iQ)으로 분할한다. 그림 5 는 영역 ②를 버퍼링 기법을 사용하여 개선된 질의어 예시이다.

```
SELECT pos_x, pos_y
FROM galis
WHERE Contains(POLYGON((2464 3464,7464 3464,
7464 8464,2464 8464)),OBJ);
```

그림 3. 질의 영역 ②에 대한 질의어 예시

```
procedure SeparateQuery
 (RANGE cQ, RANGE bQ, RANGE iQ)
// seg_cQ : segment of current Query
// seg_pQ : segment of previous Query
// listPoint : list of point
begin
  for (each segment of pQ) do
    for (each segment of cQ) do
      seg_pQ = GetSegmentFromRange(pQ);
      seg_cQ = GetSegmentFromRange(cQ);
      point = GetIntersectionPoint(seg_pQ, seg_cQ);
      if (point is validate) then
        Add point to listPoint;
      endif
    end for
  end for
  bQ = ChangePointsToRange(listPoint);
  for (each vertex of cQ) then
    point = vertex;
    if (point is not contain from bQ) then
      Add point to listPoint;
    endif
  end for
  iQ = ChangePointsToRange(listPoint);
end
```

그림 4. 질의 영역을 분할하는 의사코드

```
SELECT pos_x, pos_y
FROM galis
WHERE Contains(POLYGON((6000 7000,6000 3464,
7464 3464,7464 8464,2464 8464,2464 7000)),OBJ);
```

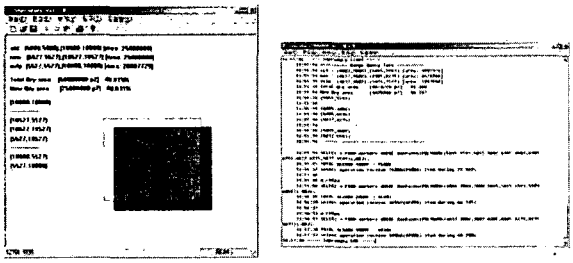
그림 5. 개선된 질의 영역에 대한 질의어 예시

3. 질의 처리 성능 실험

시스템 환경은 Intel Pentium 4 CPU 3.0GHz 및 1GB memory 에 운영체제는 마이크로소프트사의 Windows Server 2003, 데이터베이스 관리 시스템은 지오매니아사의 GMS(Geomaina Millennium Server)를 사용하여 단일 노드 환경으로 구성하고 C++로 구현하였다. 성능 평가를 측정하기 위해 사용된 이동객체 데이터는 약 30 만 건으로 서울시에 주요도로를 기준으로 이동객체 시뮬레이터인 Network-based Generator 를 이용하여

생성하였다[11].

그림 6(a)는 본 논문에서 제안한 버퍼링 기법을 위한 시스템 도구이다. 이 프로그램은 두 개의 영역 질의에 대한 상세 정보를 나타내고, 겹치는 영역에 대한 정보 및 중첩된 부분을 제외한 영역의 좌표를 추출하여 표시하게 된다. 또한 추출된 좌표 정보를 통해 질의를 수행한다. 이렇게 수행된 결과는 그림 6(b)의 텍스트 파일을 통해 저장하고 결과에 대한 자세한 정보를 확인할 수 있다.



(a) (b)

그림 6. 질의 실행 화면

본 논문에서는 제안한 버퍼링 기법의 성능 평가를 위하여 두 가지 실험을 하였다.

첫 번째 실험은 두 개의 영역 질의가 수행될 때, 질의 결과가 버퍼링되는 첫 번째 질의 영역을 동일하게 하고 두 번째 질의 영역의 위치를 조정하여 중첩된 영역 비율을 25%, 50%, 75%로 다르게 한 실험이다. 그림 7 는 실험을 통해 나타난 결과를 그래프로 표현한 것으로서, 중첩되지 않은 영역을 다각형 형태의 질의로 실험 한 것이다.

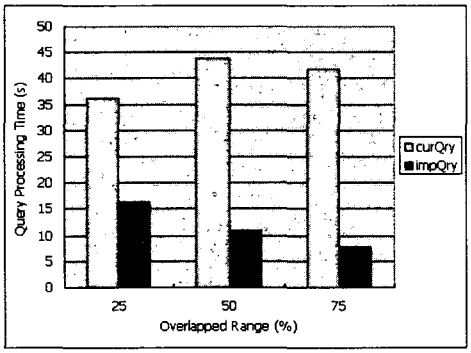


그림 7. 버퍼링 기법을 이용한 질의 처리 시간

결과를 살펴보면 중첩된 영역이 클수록 질의 처리 수행 시간이 빠른 것을 알 수 있다. 이는 중첩된 영역이 커질수록 검색해야 하는 질의 영역이 작아지므로 디스크로부터 가지고 와야 하는 데이터의 수가 감소하기 때문에 나타나는 결과이다.

4. 결론

본 논문은 이동객체 데이터베이스 시스템에서 발생할 수 있는 질의를 보다 효율적으로 수행하기 위해 버퍼링을 이용한 질의 처리 기법을 제안하였다. 본 논문에서 제안한 버퍼링 기법은 GALIS 서브 시스템의 질의 처리에 적용하여 보다 효율적인 질의를 수행할 수 있으며, 다른 이동객체 데이터베이스 시스템에도 적용시킬 수 있다.

버퍼링 기법은 여러 개의 영역 질의가 수행될 때 질의 영역이 중첩된 부분을 제외한 영역만을 수행하여 질의 처리 성능을 향상시킨다.

향후 본 연구에서 제안한 버퍼링 기법을 단일 노드가 아닌 다중 노드에 적용시켜 성능 향상을 시키는 연구가 필요하다.

참고문헌

[1] Yunmook Nah, K.H. (Kane) Kim, Taehyung Wang, Moon Hae 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, Joonwoo Lee, Seungyong Park, Ho Lee, Sangwoo Kim, Moon Hae Kim, Ki-Joon Han, "TMO-structured Distributed Location Information System Prototype," in Proc. ISORC 2005, IEEE CS Press, May 16-20, 2005, Seattle, Washington, USA, pp.321-328.

[3] 김상우, 전세길, 나연묵, "분산 이동 객체 데이터베이스를 위한 실시간 모니터링 시스템", 한국정보처리학회, 춘계학술발표 논문집 B1, 2006. (게재예정)

[4] Segil Jeon, Chung-Woo Lee, Yunmook Nah, Moon-Hae Kim, Ki-Joon Han, "Distributed Location-based Query Processing on Large Volumes of Moving Items," in Proc. The 2nd Asian Symposium on Geographic Information Systems from Computer Science & Engineering View, May 26-28, 2004, Chongqing, China, pp.208-217.

[5] Segil Jeon, Yunmook Nah, "Range Query Processing for Distributed Real-time Moving Object Databases," in Proc. the 3rd Asia Pacific Int'l Symposium on Information Technology, Jan. 13-14, 2004, Istanbul, Turkey, pp.422-427

[6] Sirish Chandrasekaran, Michael J. Franklin, "Remembrance of Streams Past: Overload-Sensitive Management of Archived Streams," VLDB 2004, pp.348-359.

[7] Mohamed F. Mokbel, Xiaopeng Xiong, Walid G. Aref, "SINA: Scalable Incremental Processing of Continuous Queries in Spatio-temporal Databases," SIGMOD Conference 2004, pp.623-634.

[8] Nagender Bandi, Chengyu Sun, Amr El Abbadi, Divyakant Agrawal, "Hardware Acceleration in Commercial Databases: A Case Study of Spatial Operations," VLDB 2004, pp.1021-1032.

[9] S. Krishnamurthy, M. J. Franklin and Garrett Jacobson., "The Case for Precision Sharing," VLDB 2004.

[10] Stavros Harizopoulos, Anastassia Ailamaki, "STEPS towards Cache-resident Transaction Processing," VLDB 2004, pp.660-671.

[11] Network-based Generator of Moving Objects, http://www.fh-oow.de/institute/iapg/personen/brinkhoff/generator/