GMS/Cluster 설계 및 구현†

유병섭*, 김명근*, 김재홍**, 배해영*
*인하대학교 전자계산공학과
**영동대학교 컴퓨터공학과
e-mail:subi@dblab.inha.ac.kr

Design and Implementation of GMS/Cluster

ᵘByeong-Seob You*, Myeong-Keun Kim*, Jae-Hong Kim**, Hyung-Young Bae*
*Dept of Computer Science, Inha University
*Dept of Computer Engineering, Youngdong University

요 약

최근 모바일 장치의 발전으로 인하여 이동 객체의 위치를 기반으로 하는 위치 기반 서비스(LBS:Location Based Service)가 발전되고 있다. 위치 기반 서비스에서는 이동 객체의 위치 정보가 시간의 흐름에 따라 그 변화량이 방대하게 증가되기 때문에 대용량 데이터를 관리하기 위한 데이터베이스의 필요성이 요구되고 있다. 기존의 데이터베이스 시스템은 하나의 서버가 모든 처리를 담당하므로 동시 사용자 수가 제한되고, 사용자의 수가 증가할수록 성능이 저하된다. 특히, 사용자 질의의 급증에 대해 유연하게 대처하지 못해 시스템이 정지하여 서비스를 제공하지 못하게 된다.

본 논문에서는 대량의 데이터를 관리하고, 사용자 질의가 급증하여도 서비스가 계속되는 GMS/Cluster를 제안한다. 제안한 GMS/Cluster는 비공유 데이터베이스 클러스터로 독립적으로 질의를 처리할 수 있는 노드와 노드들의 묶음인 그룹으로 구성된다. GMS/Cluster는 분할 정책의 사용으로 대량의 데이터를 효율적으로 관리하여 성능이 향상되고, 복제 정책의 사용으로 서비스를 하던 노드가 정지되어도 복제본을 가진 노드가 대신 서비스를 하여 가용성이 높아지며, 서비스를 중지하지 않고 실시간으로 온라인 확장을 하여 급증하거나 접근 패턴이 변경하는 사용자 질의에 대처할 수 있다.

1. 서론

최근 무선 인터넷과 PDA, HPC와 같은 모바일 장치의 발전으로 인하여 이동 객체의 위치를 기반으로 하는 위치 기반 서비스가 발전되어 가고 있다. 위치 기반 서비스에서 가장 기본적으로 선행되어야 할 사항이 바로 이동 객체의 위치 정보를 효과적으로 관리하는 방법에 대한 연구이다.

†본 연구는 정보통신부의 대학 S/W 연구센터 지원사업의 연구 결과임.

특히, 이동 객체의 위치 정보는 시간이 흐름에 따라 그 변화량이 방대하게 증가되기 때문에 대용량 데이터를 관리하기 위한 데이터베이스의 필요성이 요구되고 있다[1].

기존의 데이터베이스 시스템은 하나의 서버가 모든 처리를 담당하므로 동시 사용자 수가 제한되고, 사용자의 수가 증가할수록 성능이 저하된다. 특히, 사용자 질의의 급증에 대해 유연하게 대처하지 못해 시스템이 정지하여 서비스를 제공하지 못하게 된다.

기존의 데이터베이스 문제점을 해결하기 위해 확장 가능한 데이터베이스들에 대한 연구 및 개발이 진행되었고, 그 중에서도 비공유 구조(Shared-nothing)의 데이터베이스 클러스터는 비용대비 성능이 우수하다는 장점으로 최근까지 연구가 활발히 진행되고 있다[2, 3, 4].

본 논문에서는 비공유 데이터베이스 클러스터인 GMS/Cluster를 제안한다. 제안한 GMS/Cluster는 독립적으로 질의를 처리할 수 있는 노드와 클러스터 구성을 위한 노드들의 묶음인 그룹으로 구성된다. GMS/Cluster는 분할 정책을 사용하여 대량의 데이터를 효율적으로 분할하여 관리하므로 성능이 향상되고, 복제 정책을 사용하여 중복된 데이터를 유지하므로 서비스를 하던 노드가 정지되어도 복제본을 가진 노드가 대신 서비스를 하여 가용성이 높아진다. 또한, 서비스를 중지하지 않고 실시간으로 온라인 확장을 하여 급증하거나 접근 패턴이 변경하는 사용자 질의에 따른 작업부하 편중에 대처할 수 있다[2, 9, 10].

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구를 다루고, 3장에서는 제안한 GMS/Cluster에 대해 언급한다. 그리고 4장에서 결론을 맺는다.

2. 관련연구

2.1 데이터베이스 클러스터의 종류

데이터베이스 클러스터의 종류는 그 쓰임새와 특성에 따라 공유 디스크 구조, 공유 메모리 구조, 비공유 구조의 세가지 구조가 존재한다. 공유 디스크 구조는 모든 노드가 독립된 프로세서와 메모리를 가지고 공통의 디스크를 직접 엑세스하여 트랜잭션을 처리하는 구조이고, 공유 메모리 구조는 모든 노드가 독립된 프로세서를 가지고 공통의 메모리와 디스크를 직접 엑세스할 수 있는 구조이다. 비공유 구조는 모든 노드가 독립된 시스템으로 구성되고, 이들의 통신을 위해 각 노드들을 고속의 네트워크에 연결하는 구조이다[11].

2.2 분할 및 복제

데이터베이스 클러스터에서는 빠른 응답시간과 가용성을 가지기 위하여 분할과 복제를 이용한다.

분할은 테이블을 튜플단위로 분할하는 수평분할과 필드를 기준으로 분할하는 수직분할이 있으며, 필요에 따라 각 분할방법을 사용한다. 수평분할 방법에는 기본키에 대해 해쉬함수를 이용하여 분할하는 해쉬분할과 기본키에 대해 일정한 범위로 나누어 분할하는 범위분할이 있다. 이러한 분할을 하면 많은 갱신 질의를 동시에 처리할 수 있으므로 빠른 응답시간을 갖는다[5, 6].

복제는 한 노드의 테이블을 똑같이 복사하여 다른 노드에 두는 것으로 서비스를 하던 노드가 정지 되었을때 복제본을 가진 노드가 대신 서비스를 함으로써 가용성을 높인다[12].

2.3 온라인 확장

온라인 확장은 사용자의 질의가 급증하여 한 노드 또는 한 그룹에 집중될 때 부하를 줄이기 위하여 유휴 노드를 이용하여 해당 노드를 분할 또는 복제하여 확장하는 것이다. 이를 위하여 데이터베이스 클러스터는 실제로 질의를 처리하는 노드 이외에 사전에 등록된 유휴 노드를 항상 갖고 있다[7, 8, 9, 10].

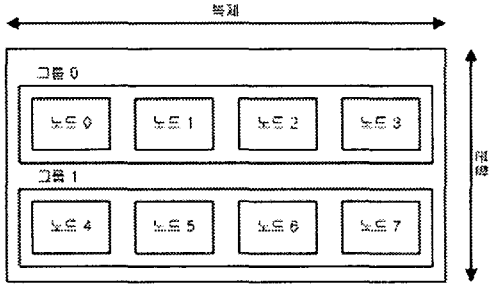
3. GMS/Cluster

3.1 GMS/Cluster 시스템 구성

3.1.1 GMS/Cluster의 클러스터 구성

GMS/Cluster의 클러스터 구성은 <그림 1>과 같다. 각 노드는 독립적으로 질의를 처리할 수 있는 GMS/Cluster 시스템으로 구성되며, 2 ~4개의 노드들을 묶어 하나의 그룹을 구성한다. 각 그룹에는

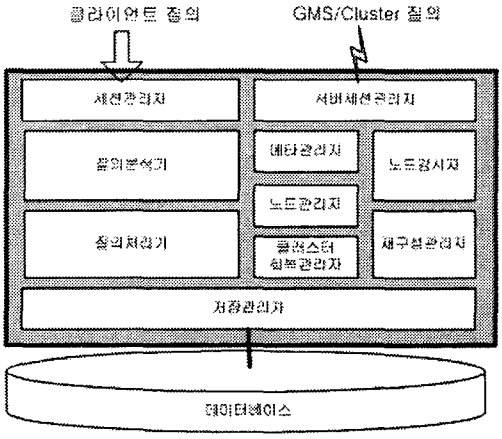
테이블에 대하여 하나의 노드에 마스터가 존재하고 나머지 노드들에 해당 테이블의 복제본들이 존재하게 된다. 또한, 하나의 그룹에는 여러개의 테이블이 존재할 수 있고, 각 테이블의 마스터는 그룹내 한 노드로 테이블 생성시 사용자가 지정할 수 있으며, 분할된 마스터들은 구분을 위한 고유의 번호를 갖는다. 분할은 그룹 사이에 하게 되며, 복제는 그룹내의 모든 노드들에 하게 된다. 각 노드들은 노드간의 정보 전송을 위해 고속의 네트워크로 연결되어 있다.



<그림 1> GMS/Cluster의 클러스터 구성

3.1.2 시스템 구성

GMS/Cluster의 구성을 간단히 살펴보면 <그림 2>와 같다. 세션관리자는 클라이언트의 질의를 받고 처리 결과를 클라이언트에 보내는 역할을 담당한다. 질의분석기는 세션관리자로부터 받은 질의를 파싱하고 파싱된 질의를 메타정보를 이용하여 분석하고, 질의처리기는 분석된 질의를 노드관리자를 통해 해당 노드에서 처리하도록 한다. 메타관리자는 클러스터 구성정보와 테이블 정보를 관리하고, 노드관리자는 클러스터로 구성된 노드 및 그룹 등의 정보를 관리한다. 클러스터회복관리자는 노드간 질의 전송 및 처리에 대한 회복을 관리한다. 노드감시자는 노드의 질의 패턴 및 테이블의 질의처리량을 감시하여 온라인 확장 여부를 결정하고, 재구성관리자는 클러스터에 구성되어 있는 테이블들의 재구성에 대한 처리를 관리한다.

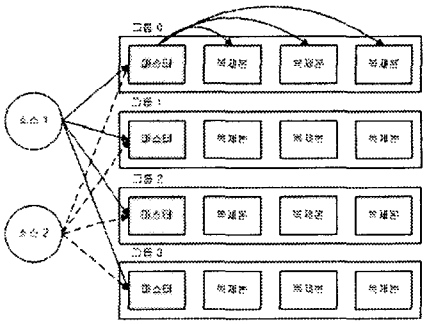


<그림 2> GMS/Cluster 구성

3.2 질의 처리

3.2.1 일반적인 질의 처리

GMS/Cluster는 클러스터로 구성되어 있어 단일 DBMS와는 다른 방법으로 질의가 처리된다. 여러 노드에 걸쳐 테이블이 분할 및 복제 되어 있으므로 이에 따른 일관성 유지와 동시성 제어가 중요하다. 따라서 GMS/Cluster에서는 <그림 3>과 같은 방법으로 일관성 유지 및 동시성 제어를 한다.



<그림 3> 일관성 유지를 위한 질의 처리 모형

(1) 소스와 마스터간 질의 처리

사용자의 질의를 받는 노드를 소스라 하면 소스는 질의를 처리할 해당 마스터들을 찾은후 마스터의 고유 번호 순서에 따라 질의를 처리한다. 모든 마스터가 성공 메시지를 보내면 사용자에게 성공 메시지를 보

내고, 어느 하나의 마스터라도 실패 메시지를 보내면 이미 실행된 마스터에 수행된 질의를 취소하고 사용자에게 실패 메시지를 보낸다.

고유번호에 따라 마스터들에게 직렬적으로 질의를 전송하므로 질의처리의 직렬성이 보장되므로 동시성 제어가 가능하고 일관성이 유지된다.

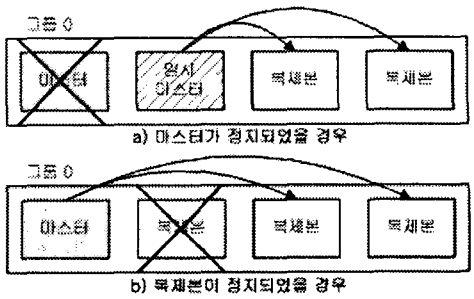
(2) 마스터와 복제본간 질의 처리

각 마스터에서는 소스로부터의 질의를 받게되면 자신의 노드에서 먼저 질의 처리를 한 후 성공이면 자신의 복제본에게 전파한다. 만약 자신의 노드에서 실패이면 소스에게 실패 메시지를 보낸다. 복제본을 갖는 모든 노드에서 성공메시지를 보내면 마스터는 소스에게 성공 메시지를 보내고, 어느 한 복제본 노드라도 실패 메시지를 보내면 이미 수행된 노드의 질의를 취소하고 마스터는 소스에게 실패 메시지를 보낸다.

그룹내의 노드들 사이에는 마스터를 통해 처리하고 마스터가 복제본을 전파하므로 직렬성이 보장되므로 동시성 제어가 가능하고 일관성이 유지된다.

3.2.2 동작하고 있는 노드의 시스템 정지시 질의 처리

특정 노드의 시스템이 정지 되었을 경우 해당 노드에 존재 하는 각 테이블에 대해 마스터인 경우와 복제본인 경우에 따라 질의 처리가 달라진다. <그림 4>에서 보듯이 마스터 테이블이 존재하는 노드가 정지되었을 경우에는 그룹내의 다른 노드의 복제본을 임시 마스터로 선정하고 질의 처리를 한다. 그렇지 않고 복제본 테이블이 존재하는 노드가 정지되었을 경우에는 기존 질의 처리와 똑같이 처리하고 마스터는 정지된 복제본에만 전파를 하지 않는다. 정지된 노드가 재개되는 과정은 회복에서 언급한다.



<그림 4> 특정 노드 정지시 질의 처리

3.3 로그 및 회복

3.3.1 로그

로그에는 노드 자체의 회복을 위한 로그와 클러스터 구성의 회복을 위한 클러스터로그가 존재한다. 노드 자체의 회복을 위한 로그는 기존의 데이터베이스 시스템과 같은 구성을 갖는다. 클러스터 구성에서의 회복을 위해 추가된 클러스터로그는 클러스터간 질의 전송이나 메타정보 전송에 대해 실패할 경우 회복을 위해 존재하는 로그로 마스터 테이블에 대한 질의와 온라인 재조직, 온라인 확장에 대한 것만 저장한다.

3.3.2 회복

클러스터에서의 회복은 자신의 데이터베이스를 회복하는 것 뿐만 아니라 클러스터 구성에 따른 회복도 해야 한다. 시스템이 정지된 노드는 재시작이 가능한 경우와 불가능한 경우가 있다. 재시작이 가능한 경우 자신의 데이터베이스를 회복한 후 클러스터 구성에 대한 회복을 하고, 그렇지 않은 경우에는 유휴노드를 정지된 노드가 속한 그룹에 추가를 함으로써 회복한다. 정지된 노드가 재시작이 가능한 경우의 클러스터 노드의 회복 순서는 다음과 같다.

(1) 시스템이 재시작되면 노드 자신에 대한 회복을 한다.

(2) 자신의 회복이 완료되면 그룹내의 다른 노드들에서 자신에 해당하는 클러스터로그를 이용하여 그동안 변경된

테이블 정보들을 전송받아 처리한다.

(3) 클러스터 구성에 따른 데이터와 메타 정보의 동기화를 맞춘다.

3.4 온라인 재조직

GMS/Cluster는 사용자의 질의패턴이 변경되어 이미 구성된 테이블을 재조정하기 위하여 온라인 재조직을 지원한다. 온라인 재조직에는 테이블 이동, 복사, 분할, 합병 등이 있다. 테이블 이동은 특정 그룹에 속한 테이블을 해당 테이블이 존재하지 않는 다른 그룹으로 이동하는 것이고, 테이블 복사는 특정 노드에 속한 테이블을 해당 테이블이 존재하지 않는 노드로 복사하는 것으로 다른 온라인 재조직 연산을 위한 것이다. 테이블 분할은 특정 그룹에 속한 테이블에 대해 해당 테이블이 존재하지 않는 다른 그룹에 해당 테이블의 일부 튜플들을 옮기는 것이고, 테이블 합병은 두 그룹에 존재하는 같은 테이블을 하나의 그룹에 합치는 것이다.

3.5 온라인 확장

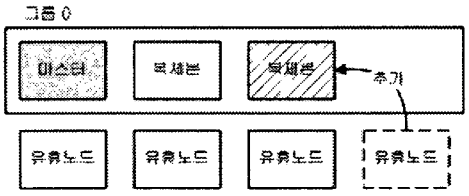
GMS/Cluster는 급증하는 사용자 질의에 대한 부하를 줄이기 위하여 온라인 재조직 중 하나인 온라인 확장을 지원한다. 특히 온라인 확장은 실시간으로 이루어지므로 사용자의 질의가 중지되지 않는다.

온라인 확장을 위해 각 노드에서는 사용자 질의의 유형 및 양을 주기적으로 검사하여 온라인 확장여부를 판단하는 감시자가 존재한다. 그리고, 시스템의 확장을 위해 사전에 등록되어 있는 노드를 유휴노드라 하는데, 이는 온라인 확장을 위해 항상 준비되어 있어야 한다.

온라인 확장에는 특정 그룹에 유휴 노드를 추가하여 복제본을 추가하는 그룹내 노드 추가와 유휴 노드들을 하나의 그룹으로 구성하고 이를 추가하여 특정 그룹의 테이블들을 확장하는 그룹 추가가 있다.

3.5.1 그룹내 노드 추가

그룹내 노드 추가는 특정 그룹에 유휴 노드를 추가하여 복제본을 추가하는 것으로 특정 그룹에 검색 질의에 의해 집중되는 부하를 분산시켜주는 것이다. 그룹내 노드 추가는 <그림 5>와 같으며 순서는 다음과 같다.



<그림 5> 그룹내 노드 추가

(1) 유휴노드가 존재하는지 검색하고, 존재하면 하나를 선정하여 그룹에 추가한다.

(2) 그룹에 존재하는 모든 테이블을 검색한다.

(3) 각각의 테이블에 대하여 복제본을 생성한다.

① 복제할 테이블의 복제본이 존재하는 노드를 찾는다.

② 복제본이 존재하는 노드에서 테이블 정보를 읽어와 새로 추가한 노드에 테이블을 생성한다.

③ 복제본이 존재하는 노드에 해당 테이블에 대한 질의를 쌓아놓을 큐를 생성하고 해당 토드의 해당 테이블에 대한 질의 처리를 중지 시킨다.

④ 복제본이 존재하는 노드에서 해당 테이블의 모든 튜플을 읽어, 새로 추가한 노드의 테이블에 복사한다.

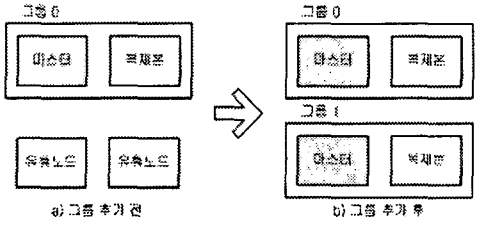
⑤ 복제본이 존재하는 노드에서 큐에 쌓인 질의를 순서대로 자신과 추가 노드에 처리한다.

⑥ 큐에 쌓인 질의를 모두 처리하면 마스터에 질의 처리 중지를 요청한 후 그룹의 동기화를 맞춘다.

⑦ 전체 노드에 대해 메타정보를 변경한 후 질의처리를 재개한다.

3.5.2그룹 추가

그룹 추가는 유휴 노드들을 하나의 그룹으로 구성하고 이를 추가하여 특정 그룹의 테이블들을 확장하는 것으로 특정 그룹에 갱신 질의가 집중되는 부하를 분산시켜주는 것이다. 그룹 추가는 <그림 6>과 같으며 순서는 다음과 같다.



<그림 6> 그룹 추가

(1) 확장하려는 그룹의 노드 개수만큼 유휴 노드가 존재하는지 검색하고, 존재하면 새 그룹으로 구성하여 그룹을 추가한다.
(2) 그룹에 존재하는 모든 테이블을 검색한다.
(3) 각각의 테이블에 대하여 다음과 같이 확장한다.
  ① 테이블의 분할부분을 선택하여 추가한 그룹에 복사한다.
  ② 복사한 데이터를 원본 그룹의 테이블에서 삭제한다.
  ③ 메타정보를 동기화한다.

4. 결론 및 향후연구

본 연구는 GMS/Cluster의 설계 및 구현에 대해 알아보았다. GMS/Cluster는 독립적인 시스템으로 동작할 수 있는 노드들이 고속의 네트워크로 연결되어 있다. 노드들의 효율적 관리와 간편한 클러스터 구성를 위해 2 ~ 4개의 노드들을 하나의 그룹으로 구성한다. 그룹과 그룹 사이에는 테이블을 수평분할하여 각 그룹이 분할된 테이블 하나를 각각 유지하고, 그룹내의 노드들 사이에는 테이블의 복제본을 유지한다. 분할과 복제에 의해 질의 처리의 가용성을 높인다.

동시성 제어와 일관성 유지를 위하여, 그룹간에 분할된 테이블들 사이에는 고유번호가 주어져 그룹간 질의 처리의 직렬성이 보장되고, 각 그룹의 노드들 사이에는 테이블에 대해 복제본들을 관리하는 하나의 마스터가 존재하여 그룹내 노드들간 질의 처리의 직렬성이 보장된다.

클러스터로그는 클러스터간의 질의 전송 및 메타정보 전송에 대해 로그를 기록하는 것으로 질의 처리 노드가 정지후 재개될 경우 일관성 유지를 위한 정보를 갖는다. 클러스터회복은 노드 자신의 회복 이후 클러스터로그를 이용하여 클러스터 구성에 대한 회복을 한다.

온라인 재조직은 이미 구성되어 있는 테이블을 재구성하여 사용자 질의 패턴 변경에 따른 테이블의 질의 처리 효율성을 높인다. 온라인 확장은 그룹내 노드 추가와 그룹 추가 등이 있는데, 그룹내 노드 추가는 특정 그룹에 유휴 노드를 추가하여 집중되는 검색질의에 대한 부하를 줄이고, 그룹 추가는 특정 그룹의 테이블들을 유휴 노드들로 새로 구성한 그룹에 분할확장 하는 것으로 급증하는 사용자 질의에 대한 부하를 줄인다.

향후연구로는 다른 데이터베이스 클러스터와 성능비교를 하는 것이다.

참고문헌

[1] R. H. Guting, and et. al, "A Foundation for Representing and Querying Moving Objects", ACM Transactions on Database Systems, Vol. 25, No. 1, pp. 1-42, 2000

[2] Svein Erik Bratsberg, "Online Scaling in a Highly Available Database", Clustra, 2001

[3] Roger Bamford, Rafiul Ahad, Angelo Pruscino, "A Scalable and Highly Available Networked Database Architecture",

Proceedings of the 25th VLDB Conference, Edinburgh, Scotland, 1999

[4] Svein-Olaf Hvasshovd, Oystein Torbjornsen, Svein Erik Bratsberg, "The ClustRa Telecom Database: High Availability, High Throughput, and Real-Time Response", Proceedings of the 21st VLDB Conference, Zurich, Switserland, 1995

[5] B. Kemme, "Database Replication for Clusters of Workstations", PhD thesis, Department of Computer Science, ETH Zurich, Switzerland, 2000

[6] Abraham Silberschatz, Henry F. Korth, S. Sudarshan, DATABASE SYSTEM CONCEPTS Third Edition, McGraw-Hill, 588-593, 1997

[7] B. Kogan and S. Jajodia, "Concurrency Control in Multilevelsecure Databases Using Replicated Architecture", Proceedings of the ACM/SIGMOD International Conference on the Management of Data, 153-162, 1990

[8] Oystein Torbjornsen, "Multi-Site Declustering Strategies for Very High Database Service Availability", PhD thesis, The Norwegian Institute of Technology, University of Trondheim, 186, January 1995

[9] Yong-Il Jang, Chung-Ho Lee, Jae-Dong Lee and Hae-Young Bae, "Improved On-line Scaling Scheme in a Scalable and Highly Available Database", In Proceedings of the PDPTA'02 Conference, Las Vegas, Nevada, USA, 2002

[10] M. Wiesmann, F. Pedone, A. Schiper, B. Kemme and G. Alonso, "Understanding Replication in Databases and Distributed Systems", In Proceedings of the 20th International Conference on Distributed Computing Systems, pages 464-474, April 2000

[11] David J. DeWitt and Jim Gray, "Parallel Database Systems : The Future of Database Processing or a Passing Fad?", Microsoft, http://research.microsoft.com/~gray/CacmParallel-DB.doc

[12] Roel Vandewall, "Database Replication Prototype", Masters thesis, Department of Mathematics and Computer Science, University of Groningen, The Netherlands, 2000