확장 가능한 고가용 데이터베이스에서 네트워크 비용을 줄이기 위한 변형된 분할기법

°유병섭* 이충호* 이재동** 배해영*
*인하대학교 전자계산공학과
**단국대학교 전산과
ysubi@hanmail.net

A Modified Fragmentation Technique for Reducing Network Cost in A Scalable and Highly Available Clustered Database

°Byeong-Seob You* Chung-Ho Lee* Jae Dong Lee** Hae-Young Bae*
*Dept. of Computer Science and Engineering, Inha University
**Dept. of Computer Science, Dankook University

*요 약

최근 전자상거래와 같은 웹 기반 응용프로그램에서는 높은 가용성과 확장성을 가지며 빠른 응답시간을 갖는 데이터베이스에 대한 필요성이 대두되고 있다. 이러한 요구에 대한 해결책의 하나로 비공유 구조의 클러스터 시스템을 구성하고 분할과 복제정책을 사용한다. 즉, 해쉬함수나 범위값에 의해 분할하여 여러 노드에 분산 시키고 서로 다른 노드에 마스터와 백업을 두어 가용성을 높이고 있다. 그러나 기존의 방법은 하나의 갱신 질의에 대해서 마스터와 백업에 각각 질의를 보내주어야 하고 온라인 확장시에는 모든 마스터와 백업의 데이터가 재구성되어야 하므로 네트워크 비용이 크다는 문제점이 있다.

따라서, 본 논문에서는 이러한 네트워크 비용을 줄이기 위한 변형된 분할 기법을 제안한다. 제안된 기법에서 마스터는 기존의 기법과 동일한 방법으로 저장하나 백업은 네트워크를 통해 지정된 노드로 포워딩을 하지 않고 질의를 받은 서버에 그대로 저장함으로써 클러스터를 구성하는 노드 사이에 통신 비용을 줄인다. 또한 온라인 확장에서는 기존의 기법과 달리 백업데이터는 같은 서버의 마스터데이터와 중복되는 것만 이동시킴으로써 데이터 이동비용을 줄이며, 전체 트랜잭션 처리량을 높인다.

1. 서 론

오늘날 인터넷 상에서 관리되는 데이터의 폭발적인 증가와 전자상거래와 같은 웹 기반 응용 프로그램 보급의 증가로 인해 대량의 데이터 관리와 클라이언트의 질의에 대한 빠른 응답시간이 중요한 문제로 대두되고 있다. 특히, 웹은 사용자 수의 급속한 증가나 접근 패턴의 변경에 따른 작업부하의 편중이 발생될 수 있는 동적인 환경이므로 높은 확장성과 빠른 응답시간을 갖는 데이터베이스에 대한 필요성이 요구되고 있다[1]. 그 결과 확장 가능한 데이터베이스들에 대한 연구 및 개발이 진행되었고, 그 중에서도 비공유 구조(Shared-nothing)는 비용대비 성능이 우수하다는 장점으로 최근까지 연구가 활발히 진행되고 있다[2, 3]. 기존의 비공유 구조의 확장 가능한 클러스터링 데이터베이스에서는 클라이언트 질의에 대한 응답시간을 줄이고 가용성을 높이기 위해 복제(Replication)와 분할(Fragmentation) 정책을 사용한다[4, 5]. 즉, 해쉬함수나 범위값에 의해서 분할하여 여러 노드에 분산 시키고 서로 다른 노드에 마스터와 백업을 두어 가용성을 높이고 있다. 이때, 복제는 갱신연산에 대한 일관성 유지비용을 최소화하기 위해서 원본데이터를 갖는 마스터 하나와 서버의 붕괴시 데이터 복구에 필요한 백업 하나를 각각 다른 서버에 두는 방법을 사용하고 있다[6, 7]. 또한 갑자기 한 데이터에 질의가 집중 되는 문제를 해결하기 위해 새로운 노드를 추가하여 데이터를 분할하는 온라인 확장을 이용함으로써 질의를 분산시킨다. 이때 서버가 멈추지 않은 상태에서 새로운 노드를 추가함으로써 온라인 확장 중에도 클라이언트의 실시간 트랜잭션을 처리하도록 하고 있다[8, 9].

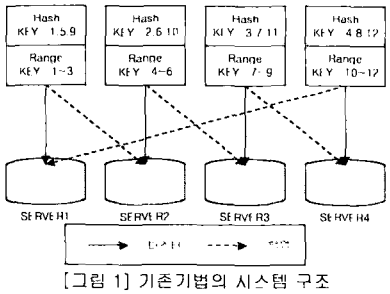
그러나, 기존의 분할 기법에서는 분할 데이터의 삽입질의가 마스터나 백업이 존재하지 않는 노드로 들어오면 질의를 두 서버에 보내줘야 하고, 온라인 확장의 경우 모든 마스터와 백업의 데이터가 재구성되어 이동해야 하므로 네트워크 비용이 증가되어 전체적인 성능 저하를 초래하게 되는 문제점이 있다[2].

* 본 연구는 정보통신부의 대학 S/W 연구센터 지원사업의 연구 결과임

따라서, 본 논문에서는 기존의 분할기법을 변형하여 네트워크 비용의 문제점을 해결할 수 있는 방법을 제안한다. 제안된 기법에서는 삽입질의에 대해서 백업 질의를 받은 서버에 저장하는 기법을 이용하여 백업에 대한 네트워크 비용을 줄이고, 온라인 확장시 백업데이터가 같은 서버의 마스터데이터와 중복되는 데이터만 이동하는 기법을 이용하여 네트워크 비용을 줄임으로써 빠른 삽입질의 수행과 온라인 확장을 하게 한다.

본 논문의 구성은 2장에서 관련 연구를 다루고, 3장에서는 제안된 기법의 시스템 환경과 배경에 대해서 언급한다. 그리고 4장에서 네트워크 비용을 줄이기 위한 변형된 분할기법을 설명하고 5장에서 성능평가를 한다. 마지막으로 6장에서 결론을 맺도록 한다.

2. 관련 연구



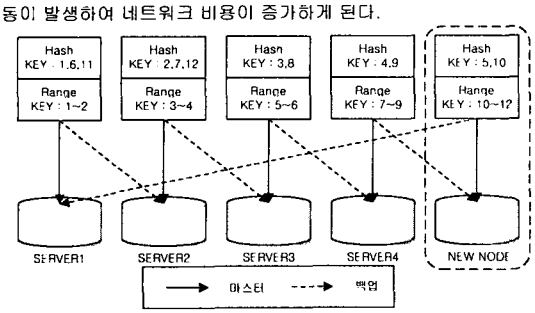
[그림 1] 기존기법의 시스템 구조

기존 분할방법은 분할된 데이터를 각각 마스터로 놓고 각 마스터의 복사본을 다른 서버에 백업으로 둔다. 이때 데이터를 분할하는 기법에는 해쉬함수와 범위를 사용하는데 [그림 1]의 시스템 구조로 이루어진다.

[그림 1]에서 해쉬함수의 경우를 보면 1 ~ 12의 키값을 갖는

데이터를 받았을 때 각 키값에 해당하는 마스터와 백업의 위치를 보여준다. 여기서 클라이언트가 키값이 1인 질의를 서버4에 보내면 서버4는 그 질의를 마스터는 서버1에 백업은 서버2에 보내어 처리하고 그 결과를 클라이언트에게 다시 보내게 된다. 이 경우 마스터와 백업 모두 네트워크로 보내야 하기 때문에 네트워크 비용이 증가하게 된다.

해쉬기반에서 온라인 확장시 새로운 노드가 추가되면 [그림 2]와 같은 구조가 되며 이때 각 서버의 데이터는 해쉬함수에 의한 데이터 구성이 되어야 하므로 모든 마스터데이터가 재구성되고 이에 따라 백업데이터도 모두 바뀌게 된다. 따라서 모든 서버에서 마스터데이터와 백업 데이터의 이동이 발생하여 네트워크 비용이 증가하게 된다.



[그림 2] 기존기법의 시스템의 온라인 확장시 구조

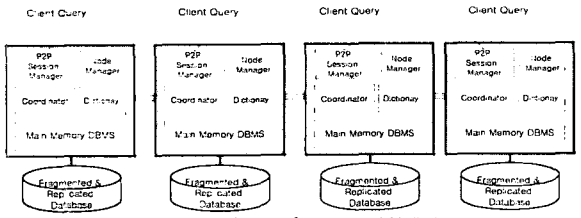
3. 시스템 개요

본 장에서는 본 논문의 전개상 미리 언급되어야 할 전체 시스템 구조와 환경 및 배경에 대해서 설명한다.

3.1 시스템 구조와 주요 컴포넌트

본 시스템의 구조는 비공유 환경을 가정한다. 즉 높은 대역폭과 네트워크 지연이 거의 없는 고속의 랜(LAN) 망을 통해 지리적으로 가까운 여러 개의 노드가 서로 연결되어 클러스터를 구성하고 각 노드는 독립적인 주기억장치와 메모리, 저장장치를 갖는 워크스테이션급으로 구성된다.

이러한 구조를 구현하기 위해 본 시스템은 [그림 3]과 같이 몇 개의 주요 컴포넌트로 구성된다. 피어투피어 세션 관리자는 각 노드간의 데이터와 로그정보, 제어 메시지 전송을 위한 피어채널(Peer Channel)과 클라이언트의 직접적인 질의를 처리하는 클라이언트 채널로 구성된다. 노드관리자는 각 노드의 시스템 정보, 질의 패턴, 접근되는 데이터에 대한 통계정보를 관리함으로써 자신의 노드에 걸리는 부하를 체크하고 온라인 확장의 실행 여부를 결정한다. 조정자는 두개 이상의 노드를 참조하는 트랜잭션을 처리하면서 임시적 데이터 공간을 관리한다. 데이터사전은 데이터의 분할과 복제 정보를 저장 관리하면서 트랜잭션 처리시에 참조된다. 각 트랜잭션은 디스크기반 DBMS에 비해 보다 빠른 질의 응답 시간을 갖는 메모리 상주 DBMS를 통해 처리되며 빠른 데이터 접근을 위한 메모리 기반 색인으로 CSB+트리(cache Sensitive B+tree)가 내부적으로 구성된다[10].



[그림 3] 시스템 아키텍쳐

3.2 데이터 분할 기법과 온라인 확장

본 시스템의 데이터는, 병렬처리를 높이기 위해 테이블의 기본키에 해쉬나 범위의 기법을 사용하는 분할(Fragmentation)과 가용성을 높이기 위해 분할된 데이터를 다른 노드에 복사본으로 저장하는 복제(Replication)라는 두 가지 정책을 혼합 사용하여 각 노드의 저장장치에서 관리된다. 복사본의 경우 두개가 가장 이상적이라는 연구 결과를 따라서 하나의 마스터(Master)에 대해서 하나의 백업(Backup)을 구성한다[7].

또한, 시스템의 가용성을 높이기 위한 방법으로 온라인 확장을 지원한다. 즉, 시스템이 가동 중에 특정 노드가 고장을 일으킨 경우나 부하가 집중되는 경우 예비로 등록되어 있던 새로운 노드가 클러스터에 추가되어 활성화 상태가 되고 기존에 분할 및 복제되어 있던 데이터와 인덱스를 재구성함으로써 시스템의 전체 트랜잭션 처리량을 증가시키고 평균 응답 시간을 줄이도록 한다.

4. 변형된 분할 기법

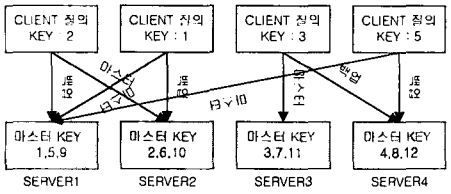
본 장에서는 시스템 구현에서 나타난 분할 기법의 문제점을 해결하기 위한 방법인 변형된 분할 기법의 기본적인 가정과 각 연산에 대한 알고리즘을 설명한다.

4.1 가정

- 동시성 제어를 위한 기법으로 2PC를 사용한다.
- 네트워크 비용이 로컬의 처리비용보다 훨씬 많이 든다.
- 모든 질의는 스위치 허브를 통해 라운드 로빈(Round Robin) 방식으로 각 노드에 전달된다.
- 마스터의 분할 방법은 해쉬로 한다.(다른 방법을 사용하는 경우에는 마스터 분할 방식에 대한 알고리즘을 변경하면 된다.)

4.2 변형된 분할 기법의 시스템 구조

기존에는 마스터와 똑 같은 데이터를 갖는 백업을 한 곳에 두었지만 여기서는 마스터에 대한 백업을 한 서버로 고정하지 않는다. 즉, 백업은 질의가 들어온 서버에서 처리하도록 한다. 이때 분할방식에 의해 마스터와 백업이 한 서버에 이루어 지게 되는 경우는 백업을 다른 서버로 넘겨 처리한다. 갱신연산의 경우, 백업데이터를 찾아야 하고 회복시 붕괴된 시스템의 마스터데이터를 찾아야 하므로 각 테이블에는 사용자에게 보이지 않는 마스터/백업의 서버 주소 필드를 하나 추가한다



[그림 4] 확장된 분할방법의 시스템 구조

[그림 4]을 보면 서버1에 키값이 2를 갖는 질의가 들어왔을 경우 마스터는 서버2에서 처리되고 백업은 서버1에서 처리된다. 이때 마스터는 백업의 서버 주소를, 백업은 마스터의 서버 주소를 갖이 저장한다. SERVER2에 KEY값이 1을 갖는 질의가 들어왔을 경우와 SERVER4에 KEY값이 5를 갖는 질의 또한 같은 방법으로 저장된다. 하지만 SERVER3에 KEY값이 3을 갖는 질의가 들어왔을 경우 마스터와 백업이 같은 서버과 된다.이때 백업은 SERVER1나 SERVER2, SERVER4중 한곳에 보낸다.

4.3 연산자별 질의 수행 방법 및 알고리즘

4.3.1 삽입연산

```
INPUT : INSERT QUERY
variables
   KeyValue : KEY값
   MasterIP, BackupIP : 마스터와 백업 IP주소
   MasterQry, BackupQry : 마스터와 백업 쿼리
begin
   KeyValue ← 입력받은 질의의 KEY값;
   MasterIP ← 마스터의 IP 주소를 넣음;
   BackupIP ← 로컬 IP 주소를 넣음;
   if MasterIP = BackupIP then
      BackupIP = FindBackupIP(). // 다른 백업서버의 주소를 찾아옴
   endif
   MasterQry = ChangeQry( 입력받은 쿼리, BackupIP ); // 마스터 쿼리에 백업서버 주소 추가
   BackupQry = ChangeQry( 입력받은 쿼리, MasterIP ); // 백업 쿼리에 마스터서버 주소 추가
   if InsertQry( MasterQry, MasterIP ) = true then // 마스터 쿼리를 마스터 서버에 입력
      if InsertQry( BackupQry, BackupIP ) = false then // 백업 쿼리를 백업 서버에 입력
         // 백업서버에서 ROLLBACK되었을 경우 마스터도 ROLLBACK
         RecoverQry( MasterQry, MasterIP );
         return ROLLBACK;
      endif
   else
      return ROLLBACK;
   endif
   return COMMIT
end
```

[알고리즘 1] 삽입 알고리즘

질의를 받은 서버는 키에 해당하는 마스터의 주소를 알아온다. 이를 백업 주소인 로컬주소와 비교하여 같을 경우 백업의 주소에 다른 백업 서버의 주소를 찾아와 넣는다. 그 다음 마스터질의에 백업주소를 추가하고 백업질의에 마스터 주소를 추가한다. 그리고 마스터에 질의를 보낸다. 이때 연산이 실패하면 클라이언트에게 실패 메시지 보내고, 성공하면 백업에 질의를 보낸다. 백업에서 성공하면 클라이언트에게 성공 메시지를 보내고, 실패하면 마스터에 이전 데이터로 회복한 후 실패 메시지를 보낸다.

따라서 질의를 받은 서버가 백업이 되므로 백업에 질의를 보내야 하는 네트워크 비용이 감소하게 되어 질의 수행시간이 빨라지게 된다.

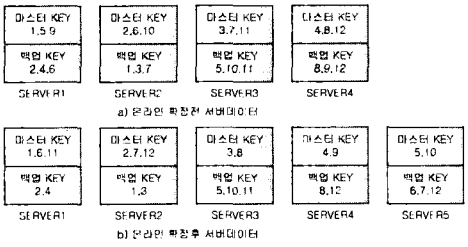
4.3.2 삭제, 갱신, 검색 연산

삭제연산과 갱신연산은 먼저 질의를 수행할 마스터를 찾는다. 그 다음 마스터에서 백업의 주소를 알아내고 질의를 수행한다. 수행 후 백업서버의 주소로 백업의 쿼리를 보내고 이를 수행한다. 모두 성공적으로 처리되었으면 성공 결과를 전송한다. 하지만 수행 중 어느 하나라도 실패하면 데이터를 이전의 값으로 회복시킨 뒤 실패 결과를 전송한다.

검색연산의 경우는 백업이 필요 없으므로 마스터에서 질의를 수행한 후 그 결과를 전송하면 된다. 따라서 질의를 수행할 해당 마스터를 찾아서 질의를 수행하고 그 결과 레코드를 모두 합하여 보낸다.

4.3.3 온라인 확장 연산

Hash기반에서 온라인 확장을 하는 방법에는 새로운 노드의 개수를 이전의 노드 개수와 같게 하는 방법이 있는데 이는 각 서버에서 새로운 노드하나에 데이터를 보내주면 끝나기 때문에 노드를 하나만 확장하는 것보다 네트워크비용에서 효율적이다. 하지만 새로운 노드가 되기 위한 서버의 개수가 기하급수적으로 필요하게 되므로 여기서는 노드 하나만 확장하는 것에 대해서 설명한다.



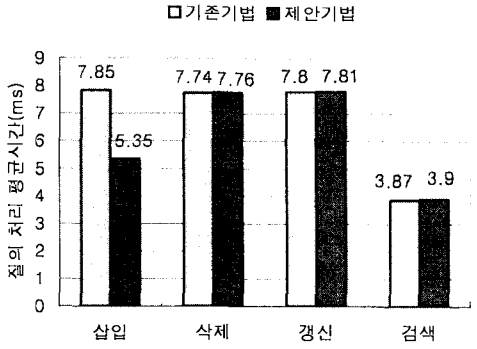
[그림 5] 온라인 확장전후의 구조

먼저 마스터를 기존의 확장방법과 같은 방법으로 확장을 한다. 그 다음 같은 서버에 마스터데이터와 일치하는 백업데이터를 찾아 새로 확장한 서버의 백업서버로 옮긴다.이러한 방법을 이용하면 온라인 확장 전 [그림 5]의 a)에서 b)와 같이 확장하게 된다. 이때 백업 데이터를 전부 이동지 않기 때문에 기존의 기법보다 네트워크 비용이 훨씬 감소하게 되어 성능이 향상되게 된다.

5. 성능평가

본 장에서는 기존의 분할 기법과 변형된 분할 기법을 적용한 두 타입에 대한 성능평가를 하였다. 테스트 환경은 4개의 노드(1.4GHz, 768MB, 랜카드 2개인 서버 3대와 700MHz, 512MB, 랜카드 2개인 서버 1대로 구성)와 한대의 클라이언트(700MHz, 512MB), 스위치 허브로 구성한다. 이때 각 노드끼리의 내부 네트워크는 스위치허브(100Mbps)를 이용하고 클라이언트와의 연결인 외부 네트워크는 스위치허브(100Mbps) 3개를 거친다. 질의는 모든 연산을 포함하며 Round Robin방식을 이용하여 랜덤하게 각 서버로 보낸다. 검색은 하나의 레코드에 대한 연산만 한다. 질의 처리시간은 시스템에서 하나의 질의를 연산하는 시간을 말한다.

테스트 결과 [그림 6]과 같이 삭제, 갱신, 검색 연산에 대해서는 비슷한 성능을 보였고 삽입 연산에 대해서는 기존 기법보다 처리시간이 약 30% 정도 빨랐다. 온라인 확장의 경우 기존의 기법은 확장할 노드의 개수가 늘어남에 따라 네트워크 비용이 크게 증가하였으나 확장된 분할 기법을 이용할 경우 네트워크 비용이 작게 증가하였다.



[그림 6] 삽입 질의 처리시간

6. 결론

본 연구는 기존 분할 기법에서의 네트워크 비용의 문제점을 보완하여 성능을 향상시키는 것에 초점을 맞추었다. 기존 기법에서 삽입 질의는 질의를 받은 서버가 아닌 마스터와 백업에 질의를 보내야 하기 때문에 네트워크 비용이 증가 하였으나 변형된 분할 기법을 사용하는 시스템에서는 하나의 서버에만 질의를 보내고 다른 하나는 질의를 받은 서버에서 처리하므로 네트워크 비용이 감소하게 되었다.또한 온라인 확장시 마스터의 데이터만 이동하고 백업데이터는 같은 서버의 마스터데이터와 같은 것만 이동하므로 네트워크비용이 감소하게 되었다

향후 연구과제로는 제안한 분할 기법에서의 보다 효율적인 조인 연산에 대한 연구가 필요하다.

7. 참고문헌

[1] *mong Li Lee, Masaru Kitsuregawa, Beng Chin Ooi, Kian-Lee Tan* Anirban Mondal, " Towards Self-Tuning Data Placement in Parallel Database Systems" , ACM/SIGMOD, 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] mong Li Lee, Masaru Kitsuregawa, Beng Chin Ooi, Kian-Lee Tan Anirban Mondal, " Towards self-tuning data placement in parallel database systems" , In Proceedings of ACM/SIGMOD(Management of Data), 225-236, May 2000

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

[6] 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

[7] *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

[8] Chendong Zou, " Safely and efficiently updating references during on-line reorganization" , In Proceedings of the 24th International Conference on Very Large Databases, New York City, New York(VLDB '98), 512-522, September 1998

[9] Svein Erik Bratsberg, " Providing a highly available database by replication and online self-repair" , International Journal of Engineering Intelligent Systems for Electrical Engineering and Communications, Special issue on Databases and Telecommunications, 4(3):131-139, September 1996

[10] jun Rao, Kenneth A. Ross, " Making B+-Trees Cache Conscious in Main Memory" , ACM, 2000