확장 가능한 고가용 데이터베이스에서 개선된 온-라인 확장 기법

°장용일* 이충호* 이재동** 배해영*
*인하대학교 전자계산공학과
**단국대학교 전산과
himalia@orgio.net

An Improved On-line Scaling Schema in a Scalable and Highly Available Database

°Yong-Il Jang* 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

데이터베이스의 활용도가 크게 증가되고, 사용자가 증가되면서 데이터베이스의 가용성과 확장성이 중요시되고 있다. 이에 따라 확장 가능한 데이터베이스는 실시간 트랜잭션의 처리를 위해 온-라인 상태로 중단 없이 동작해야 한다. 사용자의 증가에 따른 질의의 집중 현상을 해결하기 위해 데이터베이스는 사용자의 질의를 처리하면서도 확장이 가능해야 한다. 또한, 온-라인 확장은 확장 가능한 고가용 데이터베이스에서 트랜잭션의 결과 응답 시간에 영향을 미치지 않고, 트랜잭션의 처리량의 저하가 없어야 한다.

본 논문에서는 질의 집중 현상을 해결하기 위해 기존의 데이터베이스에서 제안된 기법들을 살펴보고 온-라인 확장에 대한 기존 연구에서의 문제점을 보이며, 개선된 온-라인 확장 기법을 제안한다. 제안되는 기법은 불필요한 확장 영역을 축소시키고 확장되는 노드에 대한 정책을 변형하여 내부 네트웍 사용을 줄임과 동시에, 데이터 복사의 병렬성을 향상시킨다. 본 연구를 통해 개선된 확장 기법은 온-라인 확장 시 데이터베이스의 처리량과 트랜잭션 응답 속도를 향상시키고 확장성을 유지한다.

1. 서 론

인터넷이 보편화 되면서 많은 사용자들이 인터넷을 통해 정보를 얻어가고 있다. 그 정보의 영역도 전자상거래, 인터넷 경매, 지번 검색 등 매우 다양해지고 있으며, 이러한 서비스가 24시간 지속되길 사용자들은 원한다. 이러한 요구를 수용하기 위해서는 각종 서비스에 사용되는 데이터베이스가 고확장성, 고가용성을 지원해야 한다. 그러나, 다양해지는 사용자의 요구는 사용자 질의가 동적으로 바뀔 수 있음을 의미하며, 이를 위해 데이터베이스는 서비스를 중지하지 않는 상황에서도 변화되는 사용자의 질의를 적절히 수용할 수 있게끔 확장 가능해야 한다.

즉, 데이터베이스는 사용자의 질의가 임의의 순간 집중되는 경우를 고려해 온-라인 상태에서 실시간 트랜잭션의 처리를 중지하지 않고 데이터베이스를 확장한다.

지금까지의 연구는 데이터베이스를 구성하는 노드에 질의가 집중되는 경우, 질의의 집중을 발생시키는 테이블의 데이터를 분할하여 분산시키는 방법을 통해 질의의 집중 현상을 해결하였다. 이는 데이터베이스의 구성 내용에는 변형이 없이 오직 데이터들의 분포만 바꿈으로써 해결하는 방법이다. 그러나, 이러한 경우 전체적인 사용자의 질의가 증가되는 경우를 위해서는 완벽한 해결방법이 될 수 없다.

온-라인 확장이란 임의의 노드 또는 테이블에 질의가 집중되는 경우 트랜잭션 처리를 중지하지 않고, 새로운 노드를 현재 데이터베이스에 포함시키고 새롭게 포함된 노드로 데이터를 분할하여 질의를 분산시키는 방법이다. 온-라인 확장은 데이터베이스의 범위 확장(Scale-Up)을 통해 질의 분산을 하므로 사용자의 전체적인 증가에 대해서도 해결이 가능하다.

이러한 질의의 집중으로 인한 문제를 해결하기 위한 연구로 데이터의 분산에 대한 연구가 계속 진행되어 왔지만, 직접적인 클러스터 노드의 확장을 이용한 연구는 아직 진행되지 않아왔다. 최근 [1]에서 비로소 노드의 확장을 통한 질의 분산 기법을 제안하였는데, 이 방법은 주 데이터와 복사본의 중복 복사로 인한 네트웍 과부하와 확장 영역이 해당 테이블의 전 영역에 걸치기에 노드의 수가 많을수록 수행 속도가 떨어진다.

본 논문에서는 온-라인 확장 시 성능을 개선하기 위한 새로운 클러스터 구조와 확장 기법을 제안한다. 새롭게 제시되는 클러스터의 구조는 복사본을 저장하는 노드를 가능한 주 데이터가 없는 노드로 선정하여 해당 데이터의 확장 시 불필요한 복사를 최소화 하며, 노드 내의 지역 복사의 활용으로 실시간 트랜잭션 처리 속도의 저하를 줄인다. 또한, 본 정책은 부가적으로 확장 처리의 병렬성을 높이는 장점을 갖는다.

본 논문의 내용 구성은 다음과 같다. 2장에서 관련연구를 다루고, 3장에서 바탕으로 삼는 시스템의 개요를 기술한다. 그리고 4장에서 확장된 온-라인 확장 구조 및 기법을 제시하고, 그에 따른 개선 및 문제점을 기술한다. 마지막으로, 5장에서 성능 평가를 하며, 6장에서 결론을 내린다.

2. 관련 연구

본 장에서는 온-라인 확장 이전의 질의 분산에 관한 연구 내용과 온-라인 확장의 기본 수행 단계에 대해 다룬다.

2.1 질의 분산을 위한 연구

[2]에서는 온-라인 확장을 위해 노드 간에 완전 복제를 수행하여 시스템의 실패에 대해 복구를 수행한다. 연구 내용은 주로 회복에 대해서 다루고 있는데, 온-라인 확장 중의 회복을 마치 하나의 트랜잭션처럼 관리한다.

[3]은 B+트리를 이용한 기법을 제시하였다. 이 기법은 B+트리의 특징인 완전한 트리의 구성을 각 노드의 전체 인덱스로 활용하여 특정 노드로 질의가 집중되는 경우를 막을 수 있다. 이는 다시 [4]를 통해서 비공유 클러스터에서의 메인 메모리 기반의 인덱스 구조로 확장이 되었다.

IBM에서는 쉐도우 복사를 이용, 온-라인 데이터 복사를 수행하는 연구를 하였는데, 이 연구의 주된 문제점은 두 가지 버전의 레코드를 일치시키는데 있다[5].

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

2.2 온-라인 확장

온-라인 확장 과정은 크게 세 단계로 나뉜다.

제 1 단계
- A. 사전 정보의 변경에 대한 락을 건다.
- B. 새로운 사전 정보를 생성한다.
- C. 해당 노드에 데이터 복사를 위한 대상 설정을 한다.
- D. 원본과 해당 복사본과의 논리적 연결을 생성한다.

제 2 단계
- A. 실시간 트랜잭션에 대해 로그 전송을 한다.
- B. 동시에 데이터 복사를 수행한다.

제 3 단계
- A. 질의 입력을 멈추고, 수행중인 질의가 모두 완료된다.
- B. 데이터 복사와 실시간 트랜잭션의 로그처리까지 끝낸다.
- C. 새로운 사전 정보로 사전 정보를 교체한다.
- D. 질의 수행을 재개한다.

3. 시스템 개요

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

3.1 비공유 구조

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

3.2 데이터 분할

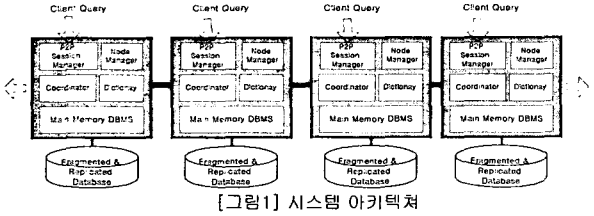
본 시스템의 데이터는 분할(Fragmentation)과 복제(Replication)라는 두 가지 정책을 혼합 사용하여 각 노드의 저장장치에서 관리된다. 분할의 방법으로 테이블의 기본키를 통해 범위에 의하거나 해쉬 함수를 통해 각 노드에 저장한다. 또한 테이블의 레코드 수에 따라서 분할에 참여하는 노드의 수가 결정되고 각 노드별로 해당되는 부분만이 저장된다[6]. 복제는 시스템의 가용성을 높이기 위한 정책으로 분할된 데이터를 다른 노드에 복사본을 저장하는 방법이다. 복사본의 경우 두개가 가장 이상적이라는 연구 결과를 따라서 하나의 주 데이터(Master Data)에 대해서 하나의 복사본(Backup Data)를 구성한다[7].

3.3 노드의 온-라인 확장

시스템의 가용성을 높이기 위한 방법으로 온-라인 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 데이터를 재 배치하고 인덱스를 재구성함으로써 시스템의 전체 트랜잭션 처리량을 증가시키고 평균 응답 시간을 줄이도록 한다.

각 노드들의 집합은 그룹으로 관리되며 그룹 내에는 활성화 노드, 유휴 노드, 복사 노드의 구분을 가지고 역할을 분담하여 시스템의 기능을 수행한다. 또한, 이는 온-라인 확장 시 정책 결정에 영향을 미친다.

3.4 주요 컴포넌트



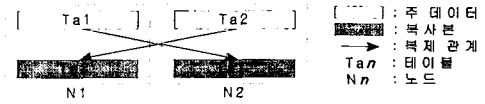
[그림1] 시스템 아키텍쳐

[그림1]과 같이 본 시스템을 구성하는 주요 컴포넌트로 피어투피어 세션 관리자(P2P Session Manager), 노드 관리자(Node Manager), 조정자(Coordinator), 데이터사전(Dictionary), 메모리상주 DBMS(Main-memory DBMS)로 나눌 수 있다. 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+트리가 내부적으로 구성된다[8].

4. 개선된 온-라인 확장 기법

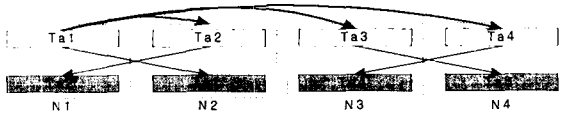
본 장에서는 기존의 온-라인 확장 기법의 문제점을 개선된 온-라인 확장 기법에서 제안하는 정책으로 해결됨을 보이고 제안된 기법이 기존 기법과의 다른점과 문제점 해결 방법을 다룬다.

4.1 기존 온-라인 확장 기법의 문제점



[그림 2] 1 : 1 복제 구조

[그림2]는 1:1 대칭 구조의 복제 형태를 갖는다. N1과 N2는 각각 노드를 가리키며, 그 안에 테이블 Ta을 분할한 Ta1과 Ta2의 주 데이터와 Ta1' 과 Ta2' 의 복제본이 존재한다.



[그림3] 온-라인 확장

[그림3]은 위의 Ta테이블을 확장하는 경우의 예를 든 것이다. Ta테이블은 노드 3, 4번에 새로운 테이블을 만들고 Ta1과 Ta2 모두로부터 데이터를 복사받는다. 그러나, 이러한 방식으로 확장을 할 경우 재확장 시 새롭게 생기는 두 노드는 기존의 4개의 테이블로부터 각각 데이터를 복사하는데 경우의 수를 따지면 복제본에 대한 복사까지 기존 N노드에 대해서 단일 노드에서의 확장을 제외하고는 모두 $2N^2+2N$번의 데이터 복사과정을 수행한다. 이는 $O(N^2)$의 수행 성능을 나타낸다.

기존의 클러스터의 구조는 위의 예에서 보았듯이 노드의 증가 수에 비해 제곱에 비례하는 데이터 복사를 필요로 한다. 기존 기법의 문제점을 열거하면 다음과 같다.

- 데이터 복제 횟수 : $O(N^2)$
- 복제본의 중복 복제
- 특정 영역의 질의 집중에 대한 고려 미비

4.2 개선된 온-라인 확장 기법

앞 절에서 다룬 기존 기법의 문제점을 해결하기 위해 본 논문에서는 클러스터 시스템에 있어서 복제본을 저장할 노드 선정을 위한 새로운 정책을 제안한다.

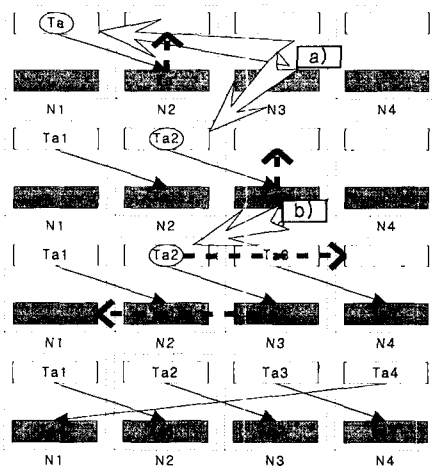
" 테이블은 논리적 그룹 내에 분포되며, 복제본 생성 시 선택되는 노드에는 같은 테이블의 주 데이터가 없도록 한다. "

[그림4]는 테이블 Ta를 확장하는 내용이다. 테이블 Ta는 처음에 단일 노드에 분포되어 있으며, 이후 질의의 집중으로 인해 Ta1과 Ta2로 분할되고, 최종적으로 모든 노드에 분산 저장되게 된다. 이 때, 각각의 경우를 살펴보면 다음과 같다.

a)의 경우 복사본에 주 데이터가 분포하지 않는다. 따라서 복사본을 이용해 주데이터를 구성하며, 다른 주 데이터가 없는 노드를 선택해 새롭게 구성되는 복사본을 저장한다. b)는 복사본 노드에 이미 주 데이터가 분포하는 경우로 이럴 때는 가능한 테이블이 분포되지 않는 노드를 선택하여 주 데이터와 복사본을 각각 분산시킨다. 이 때, 모든 확장은 단일 노드에 대해서만 한정지어 테이블 내의 다른 데이터에 영향을 주지않는다.

알고리즘은 질의가 집중되는 노드와 해당 테이블을 입력받아 해당 테

이블이 분산되어야 할 새로운 주 데이터 노드와 복사본 노드를 반환한다.



[그림4] 새로운 구조에 의한 확장 과정

개선된 온-라인 확장 기법을 위한 알고리즘은 다음과 같다.

제 1 단계 : 해당 테이블의 복사본 노드를 찾아 아이디를 반환한다.

제 2 단계 : 반환된 노드에 주 데이터가 분포되지 않았다면 ScalingLocal 함수를 이용해 복사본으로 주 데이터를 구성하고, 새로이 찾아진 복사본 노드로 데이터 복사를 수행한다.

제 3 단계 : 주 데이터가 분포되어 있는 경우 주 데이터와 복사본이 분포되지 않은 두 개의 노드를 찾아 분산 과정을 수행한다. 이 때 데이터 복사는 2번 일어나며, 연산의 수행은 병렬로 처리된다.

```
input: Scaling Event
variables :
   NodeID: 확장이 수행될 노드
   TableID: 확장의 대상이 되는 테이블
   TargetMasterID: 테이블이 확장될 대상 노드
   TargetBackupID: 확장된 테이블의 복사본 노드
Begin
        // 제 1 단계
  TargetMasterID ← FindBackupNode(NodeID, TableID);
        // 제 2 단계
  if IsExistMasterData(TargetMasterID) = false then
     TargetBackupID ← GetEmptyBackupNode();
     ScalingLocal(TargetMasterID, TargetBackupID);
  else  // 제 3 단계
     TargetMasterID ← GetEmptyMasterNode();
     TargetBackupID ← GetEmptyBackupNode();
     ScalingTwoCopy(TargetMasterID, TargetBackupID);
endif
```

[알고리즘1] 제안된 구조를 위한 알고리즘

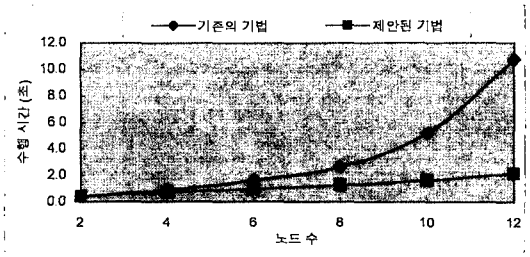
4.3 개선된 온-라인 확장 기법의 특징

제안된 기법의 가장 큰 특징은 데이터 복사의 횟수에 있다. [그림4]와 [알고리즘1]에서 보는 바와 같이 평균 데이터 복사의 횟수는 1회이며, 최악의 경우라 하더라도 2회의 복사 연산으로 온-라인 확장을 종료한다.

개선된 구조에 의한 확장의 특징은 다음과 같다.

- 데이터 복제 횟수 : 최고 O(2) → 복사본 노드에서 주 데이터를 복사하여 네트웍 부하를 줄이며, 보다 빠른 수행이 가능
- 확장에 참여하는 영역 : 단일 노드 → 확장의 범위 축소로 시스템의 실시간 트랜잭션 처리량에 미치는 영향 최소
- 그룹을 이용한 데이블 분할 영역의 관리 → 온-라인 확장 시의 영역을 그룹별로 통제가 가능
- 병렬성 향상 → 주 데이터와 복사본의 데이터 복사는 서로 다른 노드에서 수행되므로 병렬성이 향상됨
- 확장성 유지 → 확장 연산에 의해 선택되는 노드는 재 확장을 고려하여 선택되므로 확장성이 유지됨

5.성능평가



[그림5] 노드 수에 따른 온-라인 확장 시간 비교

각 노드는 Pentium IV 1.4GHz CPU와 768MB메모리를 탑재한 워크스테이션이다. 패스트이더넷을 통해 각 노드는 연결되었으며, 각각의 수치는 모두 5번씩 반복수행 후 얻어진 값들이다.

기존의 기법에서 수행시간의 증가는 네트웍의 부하보다는 제곱에 비례하는 확장의 복잡성에 기인한다. 시간의 증가는 제안된 기법에서도 발견된다. 제안된 기법에서 발생되는 시간 증가의 이유는 노드 수가 많아질수록 사용자 질의 처리를 위한 네트웍 사용이 증가하는데 있다.

6. 결론

본 연구는 방대한 데이터의 복사에 의한 내부 네트웍의 부하와 주 데이터와 복사본에 의해 중복된 데이터의 복사로 인한 불필요한 자료 전송, 특정 노드에 질의가 집중되더라도 확장의 대상이 노드 전체가 되는 기존 온-라인 확장 기법의 문제점을 해결하기 위해 개선된 온-라인 확장 기법을 제안하였다.

제안된 기법은 중복 데이터에 대한 불필요한 연산과 네트웍 사용을 줄였으며, 확장의 대상이 되는 노드를 질의가 집중되는 단일 노드로 국한시켰고, 복사본 노드 선택 정책에 의해 확장성을 지속적으로 유지한다.

개선된 온-라인 확장 기법은 확장 연산에 의한 실시간 트랜잭션의 부하를 최대한 줄였으며, 기본 기법의 $O(N^2)$의 복사 횟수에 비해 O(2)번에 해당하는 데이터 복사 횟수를 보인다. 온-라인 확장에서 가장 큰 부하를 일으키는 부분은 방대한 자료의 데이터 복사에서 일어난다. 개선된 온-라인 확장 기법은 이러한 데이터 복사를 줄임으로써 확장 수행 시간을 크게 단축시켰다.

7. 참고문헌

[1] Svein Erik Bratsberg, Rune Humborstad, "Online Scaling in a Highly Available Database", In Proceedings of the 27th VLDB Conference, Roma, Italy, 2001

[2] Svein Erik Bratsberg, Oystein Grovlen, Svein-Olaf Hvasshovd, Bjorn P. Munch, and Oystein Torbjornsen, "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

[3] Chendong Zou, Betty Salzberg, "On-line reorganization of sparsely-populated b+-trees", In Proceedings of ACM/SIGMOD, p115-124, June 1996.

[4] Mong Li Lee, Masaru Kitsuregawa, Beng Chin Ooi, "Towards Self-Tuning Data Placement in Parallel Database Systems", ACM SIGMOD, Dallas, Texas, 2000

[5] Gary H. Sockut, Balakrishna R. Iyer, "A survey of online reorganization in IBM products and research", IEEE Data Engineering Bulletin, 19(2):4-11, 1996

[6] Bettina Kemme, Gustao alonso, "A New Approach to Developing and Implementing Eager Database Replication Protocols", ACM Transactions on Databases Systems, Vol. 25, No.3, Pages 333-379, September 2000

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

[8] Jun Rao, Kenneth A.Ross, "Making B+-Trees Cache Conscious in Main Memory", ACM MOD, Dallas, TX USA, 2000