# 데이터베이스 클러스터의 가용성 향상을 위한 온라인 확장 기법

이 충 호[†] · 장 용 일[††] · 배 해 영[†††]


## 요 약

온라인 확장 기법은 비공유 데이터베이스 클러스터에서 온라인 상태에서 새로운 노드를 추가하고 데이터 재조직을 수행함으로써 작업 부하를 분산시키거나 전체 트랜잭션 처리량을 늘리기 위한 기법이다. 그러나, 기존의 온라인 확장 기법에서는 과부하 상태의 노드에 데이터 전송과 일관성 유지에 대한 추가적인 부하가 발생됨으로써, 전체 시스템의 응답속도가 느려지고 노드의 결함 발생 가능성이 높아진다. 또한, 확장 연산을 하나의 트랜잭션으로 처리하여 확장 수행 중에 결함이 발생한 경우, 전체 작업을 취소 시키므로 시스템의 가용성을 감소시키는 문제점이 있다. 본 논문에서는 비공유 데이터베이스 클러스터에서 높은 가용성을 위한 데이터 확장 기법을 제안한다. 제안된 온라인 확장 기법은 확장 연산 수행 중에 발생되는 노드의 추가적 부하를 병렬 데이터 전송 과정과 복제본의 완성 과정을 통해 분산시키고, 확장 중에 발생한 결함에 대해서 효율적인 회복을 수행함으로써 데이터베이스 클러스터의 가용성을 향상시킨다. 즉, 원본 노드의 데이터를 각 복제본이 저장된 노드들에서 동시에 전송함으로써 데이터 전송을 병렬화하고, 전송 영역을 서로 분배하여 원본 노드의 부하와 다른 트랜잭션에 대한 간섭을 줄인다. 또한, 온라인 확장 기법에서의 노드 결함에 대해 빠른 회복을 수행한다. 본 논문에서는 성능평가를 통해 제안 기법이 기존 기법에 비해 노드의 부하를 감소시켜 결함 발생 가능성을 낮추고, 온라인 확장 연산에 대한 회복 처리 시간을 단축하여 데이터베이스 클러스터의 가용성을 향상시킴을 보인다.


# An Online Scaling Method for Improving the Availability of a Database Cluster

Chung-Ho Lee[†] · Yong-Il Jang[††] · Hea Young Bae[†††]


## ABSTRACT

An online scaling method adds new nodes to the shared-nothing database cluster and makes tables be reorganized while the system is running. The objective is to share the workload with many nodes and increase the capacity of cluster systems. The existing online scaling method, however, has two problems. One is the degradation of response time and transactions throughput due to the additional overheads of data transfer and replica's consistency. The other is an inefficient recovery mechanism in which the overall scaling transaction is aborted by a fault. These problems deteriorate the availability of a database cluster. This paper presents an advanced online scaling method that solves these problems and improves the availability of a shared-nothing database cluster. To avoid the additional overheads throughout the scaling period, our scalingmethod consists of two phases : a parallel data transfer phase and a combination phase. The parallel data transfer phase reduces the size of data transfer by dividing the data into the number of replicas. The combination phase combines the transferred datausing resources of spare nodes. Also, our method presents an efficient and fast recovery mechanism in the case of a fault during scaling period. Our experiments show that the proposed scaling method reduces the possibility of failure throughout the scaling period and improves the availability of the database cluster.


**키워드 : 온-라인 확장(On-line Scaling), 온-라인 재조직(On-line Reorganization), 데이터베이스 클러스터 시스템(Database Cluster System), 고 가용성(High Availability)**

## 1. 서 론

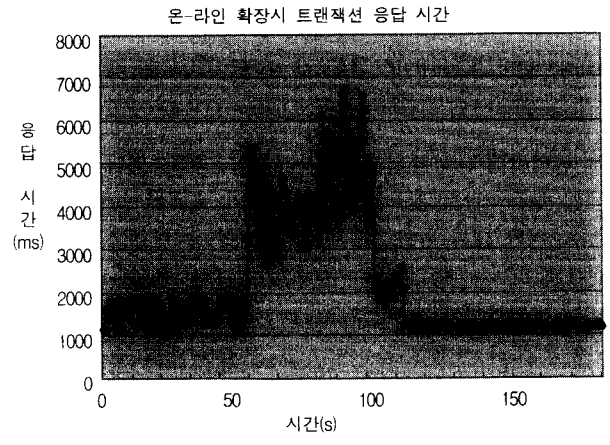
최근 인터넷 사용자와 다양한 웹 기반 정보 시스템의 폭발적인 증가로, 수많은 사용자 그룹에게 무정지 서비스를 제공하고, 갑작스러운 과부하 문제에 대처하기 위한 기술로써 데이터베이스 클러스터 기술이 사용되고 있다. 또한 이러한 클러스터 기술 중에서 비용대비 효율이 높은 해결책으로, 여러 대의 워크스테이션이나 개인용 컴퓨터들을 빠른 네트워크로 연결하여 클러스터를 구성한 비공유 데이터베이스 클러스터가 많이 활용되고 있다[1, 5]. 비공유 구조는 공유 메모리 구조와 공유 디스크 구조에 비해 우수한 가용성을 갖지만, 비공유 데이터베이스 클러스터는 다음과 같은 문제점이 있다[5].

첫째, 인터넷과 같은 동적 환경에서는 사용자의 질의 패턴

※ 본 연구는 정보통신부의 대학 S/W 연구센터 지원사업의 연구 결과임.
† 준 회 원 : 인하대학교 지능형 GIS 연구센터 연구원
†† 준 회 원 : 인하대학교 대학원 컴퓨터공학부
††† 종신회원 : 인하대학교 전자계산공학과 교수
논문접수 : 2002년 11월 8일, 심사완료 : 2003년 6월 25일

변화로, 최초 구성된 데이터베이스의 구성이 항상 최적이 아니므로 시간의 경과에 따라 시스템성능이 저하된다. 둘째, 분할된 데이터 중에서 특정 부분(hot spot area)에 대해 사용자 질의가 집중되는 경우, 시스템의 결함 및 성능 저하가 발생한다. 셋째, 사용자가 증가하여 시스템의 처리량이 한계에 도달하는 경우, 서비스의 중지 없이 전체 처리량을 확장하는 온라인 확장 기법(Online scaling)이 필수적인데, 온라인 확장 연산은추가적인 시스템 부하를 발생시켜 오히려 결함 발생 가능성을 높인다. 이러한 문제점들은 모두 데이터베이스 클러스터의 가용성과 밀접한 관련이 있으며, 가용성 저하의 요인이 된다[1, 2, 9, 10, 17, 19, 23]. 따라서, 본 논문에서는 비공유 데이터베이스 클러스터에서 가용성을 보다 향상 시키기 위한 온라인 확장 기법을 중점적으로 다룬다.

비공유 데이터베이스 클러스터 환경에서 온라인 확장 기법은 온라인 상태의 시스템에 물리적으로 새로운 노드를 추가하고 데이터 재조직을 수행함으로써 작업 부하를 분산시키거나 전체 트랜잭션 처리량을 늘리기 위한 기법이다[1, 6, 24]. 온라인 확장 기법은 과중한 작업 부하에 의한 시스템 결함 문제를 해결하고 시스템의 가용성을 높인다. 그러나, 기존의 온라인 확장 기법은 현재 진행중인 트랜잭션들로 인해 이미 과부하 상태의 원본 노드(source node)에서 목적 노드(destination node)로 온라인 확장 연산을 수행함으로써, 원본 노드의 다른 트랜잭션 처리를 방해하고, 확장 연산의 추가적 부하로 인해 결함 발생 가능성을 높인다. 또한, 기존의 온라인 확장 기법은 확장 연산을 하나의 트랜잭션 단위로 처리함으로써, 확장 연산 수행 중에 발생한 하나의 결함에 대해 확장 연산 전체를 취소하고 재수행을 하는 문제점이 있다. (그림 1)은 온라인 확장 연산의 수행 중, 트랜잭션 응답 시간의 변화 예를 보이고 있다. 확장 연산을 수행함으로써 수행 전에 비해 수행 후가 시스템의 응답속도가 눈에 띄게 향상되지만, 확장 연산의 수행 중에는 추가적인 데이터 및 로그 전송 부하로 응답시간이 크게 느려짐을 볼 수 있다[1].




(그림 1) 온라인 확장 수행 중 트랜잭션 응답 시간의 변화 예

따라서, 본 논문에서는 온라인 확장 연산의 비용을 줄이고, 확장 연산 수행 중의 결함 발생에 대해 효율적인 회복 과정을 수행하는 높은 가용성을 보장하는 온라인 확장 기법을 제안한다. 제안된 기법은 기존 기법과 비교해서 다음과 같은 차이점을 갖는다.

① 온라인 확장 연산의 비용을 줄임으로써 결함 발생 가능성을 줄인다. 확장 연산의 수행 중에 원본 노드(source node)에 집중되는 부하를 확장에 참여하는 예비 노드(spare node)의 여유 자원을 통해 해결한다. 즉, 하나의 원본 노드가 아닌 복제본(replica)을 갖는 하나 이상의 원본 노드들을 이용해 원본 노드가 전송하게 되는 영역을 분담시킴으로써 원본 노드별 데이터 전송량을 줄이고, 영역별 전송된 데이터에 대한 복제본을 예비 노드에서 추후 완성함으로써 원본 노드의 부하를 줄이고 현재 진행 중인 트랜잭션에 대한 간섭을 줄인다.

② 온라인 확장 연산 수행 중 결함 발생에 대한 효율적인 회복 과정을 수행한다. 확장 데이터의 영역별 분배 및 관리로 노드의 결함 발생시 효율적인 회복 처리가 가능하도록 한다. 즉, 각각의 예비 노드에서 전송 영역을 독립적으로 관리하면서 복제본을 완성함으로써, 확장에 참여하는 노드들에 결함이 발생하더라도 전송 영역의 재분배와 예비 노드간 손실 데이터의 복사 과정을 통해 효율적인 회복 처리가 가능하도록 한다.

본 논문에서는 실험 모형에 의한 성능 평가를 통해 제안된 기법이 기존 온라인 확장 기법에 비해 복제본의 수가 늘어날수록 원본 노드의 부하가 감소함을 보이고, 회복 처리에서도 기존 기법에 비해 처리 성능이 개선됨을 보인다.

본 논문의 구성은 다음과 같다. 2절에서 데이터베이스 클러스터에 대한 선행 연구와 온라인 확장 기법에 대해 살펴본다. 3절에서는 고 가용성을 위한 온라인 확장 기법의 기본 구조 및 수행 과정을 제안하고, 영역의 재분배, 일관성 유지 방법에 대해 세부적으로 설명한다. 4절에서는 온라인 확장 기법의 회복 처리를 위한 확장 정보의 관리, 동기화 과정, 확장 관리자 등의 내용을 다룬다. 5절에서는 제안 기법과 기존 온라인 확장 기법의 성능 평가를 보이고, 마지막으로 6절에서 결론을 맺는다.

## 2. 관련 연구

본 절에서는 먼저 비공유 데이터베이스 클러스터의 확장성과 가용성을 다루고, 온라인 확장 기법에 대한 기존 연구에 대해 설명한다.

### 2.1 비공유 데이터베이스 클러스터의 확장성 및 가용성

데이터베이스 클러스터 시스템의 목적은 고성능과 높은 가용성, 확장성이다. 그 중에서 확장성은 기존의 시스템에

새로운 노드를 추가함으로써 트랜잭션 처리 능력을 향상시키거나 전체 시스템의 성능을 향상시킬 수 있는 능력을 의미한다. 과거의 확장성은 단일 시스템인 SMP(Symmetric Multiprocessor)에 메모리 또는 프로세서를 추가해 병렬성 위주의 해결을 하였기 때문에 비공유 구조의 클러스터 시스템에서의 확장성과는 차이가 있다. SMP 시스템은 시스템의 확장을 수행하기 위해 서비스를 중단하여야 했고, 또한, 일부 문제가 발생할 경우 전체 서비스의 중단으로 이어져 가용성에 있어서 큰 문제가 있었다. 일반적으로 데이터베이스의 확장성과 가용성을 서로 독립적인 특성으로 보기 쉽지만 가용성에 있어서 확장성은 매우 중요한 요소이다. 즉, 데이터베이스 시스템은 높은 가용성을 유지하기 위해 급격한 작업량의 증가에도 서비스가 중지되지 않아야 하는데, 확장성은 시스템의 전체 작업 처리량을 늘리고 동시에 부하 집중 문제를 해결함으로써 높은 가용성을 보장한다[5]. 비공유 구조의 데이터베이스 클러스터 시스템에서는 가용성과 확장성을 향상 시키기 위해 데이터 분할(partition)과 복제(replication)라는 두 가지 정책을 주로 사용한다[1, 21, 22].

비공유 구조의 데이터베이스 클러스터에서 임의의 노드에 결함이 발생한 경우 해당 노드는 더 이상의 트랜잭션 처리가 불가능하며, 해당 노드의 복제본을 가지고 있는 다른 노드들은 결함 발생 노드를 제외시키고 트랜잭션 처리를 수행한다. 따라서 결함이 발생되더라도 트랜잭션 처리는 지속될 수 있다. 회복 과정을 통해 해당 노드는 다시 시스템에 참여할 수 있지만 복구를 위해 자기 복구 과정, 데이터 전송 과정, 동기화 과정을 거치게 된다. 자기 복구 과정은 결함 발생에 의해 데이터베이스의 내용이 일관성을 잃는 문제를해결한다. 대부분의 데이터베이스 시스템은 noforce/steal 정책을 사용한다. noforce 정책에 의해 완료(commit)된 트랜잭션이 디스크에 반영되지 않았을 수 있으며, 그러나, steal정책에 의해 아직 완료되지 않은 데이터가 디스크에 반영되는 경우가 발생할 수 있다. 따라서 자기 복구 과정에서는 무효화(undo) 과정과 재수행(redo) 과정의 두 단계를 수행하여 데이터베이스의 일관성을 복구한다. 데이터 전송 과정은 결함 발생 후 복구 처리의 시작까지 다른 복제본들을 통해 수행된 트랜잭션의 갱신 정보를 해당 노드로 전송하여 재수행 과정을 거친다. 이때, 데이터 전송 크기를 줄이기 위해 다른 노드들은 재수행 로그와 같은부가적인 자료구조를 관리한다. 동기화 과정은 복구 완료된 노드를 다른 노드와 함께 트랜잭션 처리에 참여 시키는 과정에 속한다. 데이터 전송 과정 도중에 발생되는 트랜잭션 처리에 의해 다른 노드는 반복적인 데이터 전송 과정을 수행하게 되며, 이러한 문제를 해결하기 위해 복구 노드는 데이터 전송 과정이 거의 마무리되는 일정 순간부터 발생되는 트랜잭션을 큐에 저장하고 데이터 전송 과정이 끝난 후 큐에 쌓인 트랜잭션을 일괄적으로 처리함으로써 다른 노드들과의 동기화 과정을 수행한다[8, 9].

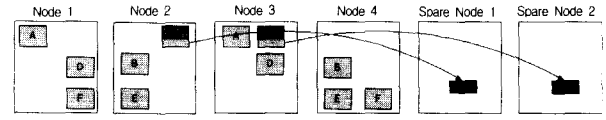
### 2.2 온라인 확장 기법

온라인 확장 기법은 물리적으로 현 시스템에 활성 노드를 추가하고 부하가 집중되는 데이터를 분할하여 새로운 노드로 이동시켜 전체적인 처리량을 늘리는 동시에 부하 분산을 하는 기법이다[1]. 일반적으로 온라인 확장 기법은 온라인 재조직의 한 부분으로 분리될 수 있으며, 온라인 재조직에서의 복사본 간의 데이터 이동과는 달리 온라인 확장 기법에서는 새로운 복제본을 생성하고 해당 복제본으로 기존의 복제본의 데이터를 이동시키는 방법을 사용한다[22]. 온라인 확장 기법은 확장 시 소요되는 처리 시간을 줄이고, 실시간 트랜잭션에 대한 영향을 최소화해야 하며, CPU · 네트워크 부하를 최소화하며, 데이터베이스의 확장성을 향상시키는 동시에 확장 후 질의 처리 성능을 향상시킬 수 있어야 한다[11, 16].

온라인 확장 기법의 처리 과정은 크게 준비과정, 데이터 전송 과정, 완료 과정으로 나뉜다. 준비 과정에서 온라인 확장 기법은 확장의 목적이 되는 노드의 선정, 새롭게 생성되는 복제본에 대한 카탈로그 정보의 생성 등의 처리를 하며, 데이터 전송 과정을 통해 데이터와 데이터 전송 도중에 발생하는 로그를 전송한다. 이 때, 확장 도중에 입력되는 트랜잭션을 고려하여 데이터 전송 과정은 데이터를 일정 크기의 블록으로 나누고, 데이터 전송 프로세스의 우선 순위를 최하로 두거나 일정한 시간 간격으로 데이터 블록을 전송하는 지연 전송 기법을 사용한다. 데이터의 전송이 모두 완료되면 완료 과정을 수행하는데, 이 과정에서는 원본 데이터와 확장 데이터의 동기화 과정, 데이터베이스 카탈로그 정보의 변경, 원본 데이터 내의 확장 부분 삭제 등의 처리를 한다.

클러스터를 구성하고 있는 여러 노드들을 역할에 따라서 두 가지 타입으로 나누는데 실시간 클라이언트 질의를 수행하고 있는 노드를 활성 노드(active node)라 하고 시스템의 확장을 위해서 사전에 등록되어 있는 노드를 예비 노드(spare node)라 한다. 온라인 확장 기법의 처리를 위해 예비 노드는 항상 준비되어 있어야 한다.

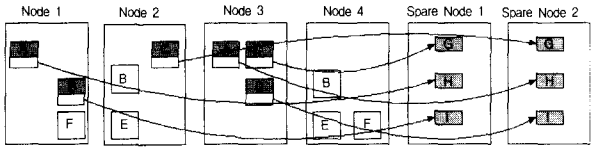
(그림 2)는 단일 복제본에 대한 온라인 확장 기법의 적용 내용을 나타낸다. 예를 들어, 복제본 C가 범위 분할 기법을 기반으로 구성이 되어 있으며, 801부터 1000까지의 데이터를 가지고 있다면 온라인 확장에 의해 기존의 복제본 C는 801부터 900까지의 데이터를 가지고 새롭게 확장된 복제본 G는 기존의 복제본 C의 901부터 1000까지의 데이터를 가지게 된다.




(그림 2) 단일 복제본에 대한 온라인 확장 기법

(그림 3)은 노드에 대한 부하 집중의 경우 다수의 복제본을 동시에 확장시키는 과정을 나타낸다. 각각의 과정은 온라인 확장 기법에서 쓰레드로 구현이 되어 병렬적으로 수행한

다. 온라인 확장 과정은 여러 프로세스 집합의 상호 연관관계를 통해 동작된다. 이러한 프로세스의 집합에는 전송 데이터를 읽는 프로세스, 분할 프로세스, 데이터 전송 프로세스, 갱신 데이터의 반영 프로세스 등이 있다. 각각의 과정에서 온라인 확장 연산은 마치 하나의 트랜잭션 단위로 관리가 되지만 몇몇 프로세스에서는 자신의 수행 결과에 대한 로그를 남기지 않아 결함이 발생할 경우 지금까지 진행해온 확장에 관련된 내용들을 모두 취소하게 된다. 그러나, 데이터베이스의 무결성을 보장하기 위해 온라인 확장 연산에서 쓰여진 확장 공간이나 네트워크 채널 등의 자원을 원 상태로 돌리는 작업을 수행한다. 따라서, 온라인 확장 기법은 처리 과정을 트랜잭션 단위로 관리하여 네트워크 단절이나 노드의 결함 발생 등의 문제에 대해서는 전체 확장 연산을 중지시키고 데이터베이스를 확장 연산 이전의 상태로 돌려놓는다.



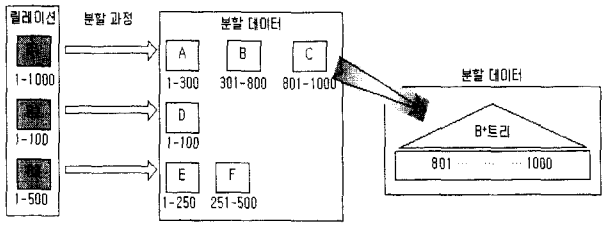

(그림 3) 노드에 대한 확장 기법

## 3. 고가용성을 위한 온라인 확장 기법

본 절에서는 고 가용성을 위한 온라인 확장 기법을 제안한다. 먼저, 제안 기법의 기반 환경과 온라인 확장 기법의 고려사항을 알아보고, 제안 기법의 기본 수행 모델을 제안한다.

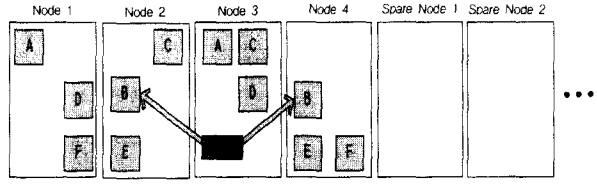
### 3.1 기반환경 및 고려사항

제안 기법의 기반환경은 비공유 구조의 데이터베이스 클러스터에서 부분복제(partial replication)된 데이터를 관리한다. 임의의 릴레이션 전체 또는 분할된 데이터를 가용성을 위해 두 개 이상의 복제본을 만든다. 각각의 복제본들은 비균등 범위 분할 기법에 의해 분할된 상태이며 2계층 B+-트리 색인을 구성한다[11]. 데이터베이스 클러스터는 확장 가능하며, 이를 위해 예비 노드(spare node)를 둔다. 예비노드는 확장 연산의 목적 노드(destination node)로 사용된다. 예비 노드의 수는 관리자에 의해 결정된다[1, 6]. (그림 4)는 릴레이션 R1, R2, R3에 대해서 B+-트리 색인에 기반 한 비균등 범위 분할이 된 상태이다.




(그림 4) 데이터의 분할

(그림 5)는 (그림 4)에서 분할된 데이터가 클러스터를 구성하는 노드에 가용성을 위해 부분 복제된 상태이다.




(그림 5) 데이터의 복제

온라인 확장 기법은 데이터베이스의 서비스를 온라인 상태로 유지하면서 새로운 노드를 데이터베이스 클러스터에 포함시키기 때문에 온라인 확장 연산의 수행에 있어서 가장 큰 부하를 갖는 과정은 데이터 이동 과정이다. 기존의 온라인 확장 기법은 확장에 참여하는 모든 원본 노드(source node)들이 각각의 목적 노드(destination node)로 동일 데이터를 이동시키게 되면서 데이터 전송에 의한 부하가 원본 노드로 그대로 부과되며, 현재 처리중인 트랜잭션을 방해하게 된다. 이동 데이터를 네트워크의 환경에 최적화된 크기의 블록 단위로 나누고 일정 시간 간격으로 지연시키면서 전송하여 노드의 부하를 줄이는 지연 전송 기법은 이러한 문제를 해결하기 위해 사용되지만 데이터의 크기가 커질수록 온라인 확장 연산에 소요되는 시간이 크게 증가되어 확장성을 저하시킨다. 또한, 서비스를 지속적으로 제공하면서 온라인 확장을 처리하기 위해서는 확장 영역에 포함되는 원본 데이터와 확장 데이터간에 일관성 유지가 반드시 필요하다. 그리고, 온라인 확장 연산을 수행하는 도중에 발생하는 시스템 고장 또는 운영체제의 오류와 같은 결함의 발생에 대해 기존의 기법은 확장 연산 전체를 무효화하거나 처음부터 재수행 하는 문제점을 갖는다. 따라서, 본 논문에서 제안하는 온라인 확장 기법은 다음의 사항들을 고려한다.

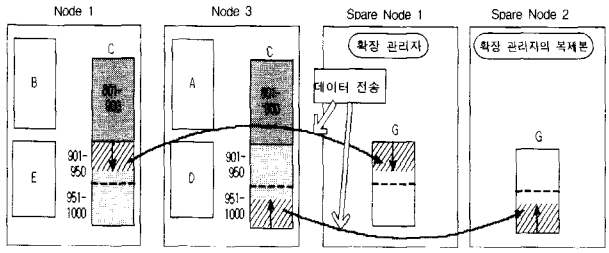
첫째 : 온라인 확장 연산에 의해 원본 노드에 추가되는 부하의 크기를 최소화한다. 이를 위해 온라인 확장 연산을 의해 선택되는 목적 노드들의 자원을 최대한 활용한다.

둘째 : 온라인 확장 연산의 수행 중에 노드의 결함 발생 문제를 허용하면서 온라인 확장 연산을 무효화나 재수행 없이 완료하도록 한다.

### 3.2 고가용성을 위한 확장 기법

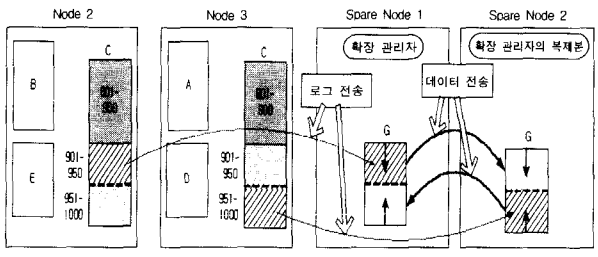
앞에서 언급한 고려사항을 기반으로 제안된 온라인 확장 기법은 두 단계로 구성되는데, 병렬 데이터 전송 단계와 복제본의 완성 단계이다. 즉, 원본 데이터가 있는 원본 노드에서 목적 노드로 병렬 전송 후 목적 노드에서 복제본의 완성 단계를 수행한다. 데이터의 병렬 전송을 통해 데이터 이동에 의한 원본 노드의 부하를 낮추고, 각 노드의 전송

영역을 논리적으로 분배함으로써 특정 노드의 결함 발생 시 회복을 효율적으로 수행한다. 확장 관리자는 온라인 확장 연산을 수행하는데 필요한 정보를 가지고 확장 연산을 주관한다. 또한, 복제본의 완성 과정은 예비 노드의 자원들을 최대한 활용하여 원본 노드의 부하를 크게 줄인다.




(그림 6) 병렬 데이터 전송

(그림 6)은 병렬 데이터 전송 과정을 나타낸다. 예를 들어 온라인 확장 연산을 위해 복제본 C가 선택되고 예비노드 1과 예비노드 2로 확장이 되는 경우, 노드 2와 노드 3은 원본 노드가 되고 예비노드 1과 예비노드 2는 목적노드가 된다. 이때, 복제본 C는 전송 영역을 2등분하여 원본 노드들이 서로 다른 영역을 각각 목적 노드로 전송한다. 이때, 온라인 확장에 참여하는 원본 노드의 수가 N이라면 각각의 노드가 전송하여야 할 데이터의 영역은 전체 전송 영역의 1/N이 된다. 결국 병렬 데이터 전송 과정을 통해 각 원본 노드의 데이터 전송 크기가 줄어드는 효과를 얻을 수 있다. 또한, 데이터 전송 도중에 입력되는 트랜잭션에 의해 이미 전송된 영역의 데이터가 바뀌는 경우, 제안 기법은 트랜잭션의 갱신 영역에 대한 재수행 정보만을 담는 재수행 로그를 생성하여 자신이 데이터를 전송하는 목적 노드로 다음 데이터 전송시 함께 전송한다. 그리고, 복제본의 완성과정에 필요한 목적 노드들간의 데이터 전송시에는 지연 전송 기법을 사용할 필요가 없고 빠른 데이터 전송이 가능하다. 결과적으로 저속에서의 데이터 전송을 축소시키고 고속의 데이터 전송을 활성화하여 온라인 확장 처리 시간에 대해서 이득을 얻을 수 있다.




(그림 7) 복제본의 완성 과정

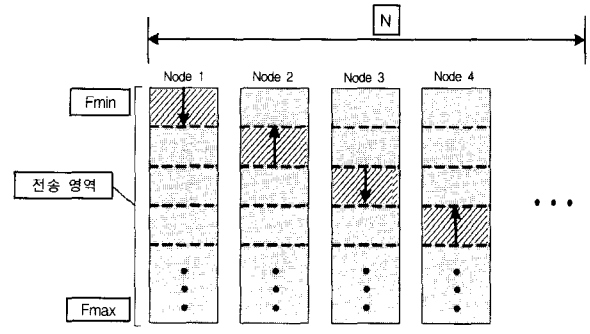
(그림 7)은 원본 노드로부터 각각의 목적 노드로의 데이터 전송이 완료된 후, 목적 노드에서 수행되는 복제본의 완성 과정을 나타낸다. 데이터 전송이 완료된 후 목적 노드들은 서로 독립적인 데이터를 가지게 되며, 목적 노드들 간에 서로 데이터를 전송함으로써 온라인 확장 과정을 마치게 된다. 이때, 복제본의 완성 과정 도중에 원본 노드에서 발생되는 트랜잭션에 의해 전송되는 로그 정보는 목적 노드에서 현재 자신의 전송 영역에 따라 여과되어 다른 목적 노드로 전송된다. 결과적으로 복제본의 완성 과정을 통해 예비 노드의 자원을 효율적으로 활용하게 된다.

제안 기법은 병렬 데이터 전송과 복제본의 완성 과정을 제외한 부분에서 기존의 온라인 확장 기법의 기본적인 수행 과정과 동일하게 동작한다. 그러나, 확장 관리자를 새롭게 생성하여 관리함으로써 온라인 확장 기법의 전체적인 관리를 처리한다. 확장 관리자는 온라인 확장 이벤트가 발생된 이후에 예비 노드에서 동적으로 구동이 되며, 확장이 완료되면 자동으로 삭제된다. 또한, 확장 관리자는 자신의 정보를 다른 예비 노드에 중복 복제하여 관리함으로써 온라인 확장 기법의 가용성을 높이도록 한다. 두 번째 과정인 확장 정보의 생성 과정에서는 확장 이벤트를 통해 확장의 종류 및 영역을 설정하고 확장 관리자를 통해 각 노드의 확장 정보를 생성하여 전송한다. 전송된 확장 정보를 통해 세 번째 단계에서 각 노드의 목적에 맞게 프로세스를 구동한다. 이후, 확장의 완료 과정에서는 원본 노드의 데이터와 목적 노드의 데이터를 동일한 상태로 만든 후 확장 과정을 마치게 된다.

### 3.3 자료구조 및 세부동작 과정

#### 3.3.1 데이터의 병렬 전송

제안 기법은 데이터 전송의 효율을 높이기 위해 동일 복제본을 가지고 있는 원본 노드들이 전송해야 할 데이터의 영역을 독립적으로 분배하고, 각 노드가 전송해야 할 목적 노드를 서로 다르게 하여 원본 노드의 부하와 현재 진행중인 트랜잭션 처리에 대한 간섭을 최소화한다.




(그림 8) 데이터의 병렬 전송 구조

(그림 8)에서 Fmin과 Fmax는 각각 전송 영역에 포함되는 레코드의 기본키 값의 최하 값과 최고 값을 나타낸다. N은 원본 노드의 수를 나타내며, 각 노드의 전송 영역은 N만큼 나뉘어진다. 이때, 각 노드의 데이터 전송 영역 minKey

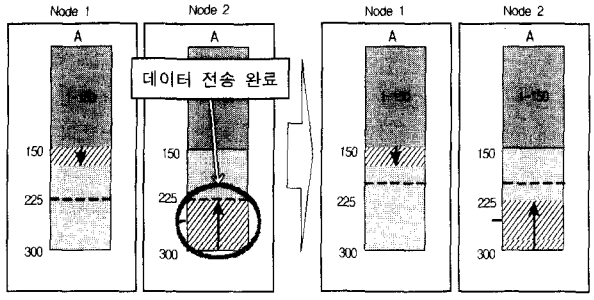
와 maxKey, 그리고 전송 방향에 해당하는 Order는 다음의 식 (1), 식 (2), 식 (3)에 의해 각각 결정된다.

$$\min Key \leftarrow ((F\max - F\min)/N)\times(n-1) \quad (1)$$

$$\max Key \leftarrow ((F\max - F\min)/N)\times n \quad (2)$$

$$order \leftarrow (n \bmod 2 = 1)?\, ASC : DESC \quad (3)$$

데이터의 전송 영역이 논리적으로 나뉘는 기준은 항상 현재 전송 영역과 다음 전송 영역이 서로 반대가 되도록 하고, 서로 전송 방향이 마주보게 되는 노드를 이웃 노드라 한다. 이러한 방식은 각 노드의 전송 속도에 따라 데이터 전송의 완료 시간이 달라지는 경우 전송 영역의 재분배를 보다 수월하게 수행하기 위해 적용된다.




(그림 9) 전송영역의 재분배

(그림 9)는 노드 2의 데이터 전송이 노드 1보다 먼저 끝났을 경우 노드 1과의 재분배 과정을 간단하게 처리할 수 있음을 보인다. 임의의 노드가 데이터 전송을 완료하게 되면, 해당 노드는 확장 관리자로 전송 완료 메시지를 전달한다. 확장 관리자는 해당 노드의 전송 방향에 있는 이웃 노드가 아직 데이터를 전송하는 중이라면, 이웃 노드와 데이터 전송을 완료한 노드 간에 일시적으로 데이터 전송을 중지하고 전송 영역의 재분배 과정을 수행한다.

3.3.2 복제본의 완성

원본 노드로부터의 데이터 전송이 완료되는 시점부터 목적 노드는 자신과 다른 영역을 전송 받은 목적 노드들과 서로 데이터를 주고 받는다. 이때, 원본 노드들은 더 이상의 데이터 전송은 하지 않으며, 갱신 트랜잭션이 발생하여 변경된 데이터에 대한 재수행 로그만을 전송한다. 이러한 과정을 통해 각각의 목적 노드들은 자신이 전송 받은 데이터 영역 이외의 데이터를 전송 받음으로써 완전한 복제본을 구성한다. 또한, 목적 노드들은 트랜잭션 처리에 대한 간섭을 고려할 필요가 없어 원본 노드에서 사용하는 지연 전송 기법을 사용할 필요가 없게 되며, 보다 고속의 데이터 전송이 가능하다. 이러한 특징은 원본 노드의 저속의 데이터 전송 부분을 줄이고 목적 노드의 고속의 데이터 전송 부분을 확장하여 원본 노드와 목적 노드의 데이터 전송 속도의 차이가 클수록 시간적으로 빠른 확장 처리가 가능하게 된다.

3.3.3 노드간 데이터 전송

제안 기법은 데이터의 전송 영역을 각 노드에 분배하게 되면서 기존의 온라인 확장 기법과는 다르게 전송 영역과 전송 방향과 같은 부가적인 정보가 필요하게 되었다. 또한, 임의의 원본 노드와 해당 원본 노드로부터 데이터를 전송 받는 목적 노드는 다른 노드와는 독립적으로 데이터의 송수신을 수행할 수 있어야 한다. 이를 위해 각 노드에는 데이터 전송과 수신을 위한 각기 다른 정보가 생성된다. 확장 정보는 동적으로 구성되어 메인 메모리 내에서 관리되며, 데이터 전송이 완료되면 자동으로 삭제된다. (그림 10)은 데이터를 전송하는 노드와 수신하는 노드가 가지는 전송 정보와 수신 정보, 그리고 확장 관리자가 가지는 정보를 나타낸다. 데이터의 송수신 정보와는 다르게 확장 관리자의 정보는 테이블로 관리된다.

데이터 전송 정보

| DestNodeNum | minKey | maxKey | HighKey | Order | HighLSN | CopyMode |
|---|---|---|---|---|---|---|
| 대상 노드 | 전송범위 (최소) | 전송범위 (최소) | 최근전송 데이터 값 | 진행 순서 | 최근 로그 번호 | 확장 노드의 데이터 이동 |

데이터 수신 정보

| SrcNodeNum | minKey | maxKey | HighKey | Order |
|---|---|---|---|---|
| 원본 노드 | 전송범위 (최소) | 전송범위 (최소) | 최근 전송 데이터 값 | 진행 순서 |

확장 관리자의 정보 테이블

| SrcNodeNum | DestNodeNum | minKey | maxKey | Order | CopyMode | TargetReplID |
|---|---|---|---|---|---|---|
| 원본 노드 | 대상 노드 | 전송범위 (최소) | 전송범위 (최소) | 진행 순서 | 확장 노드의 데이터 이동 | 대상 복제본의 번호 |

(그림 10) 확장 정보

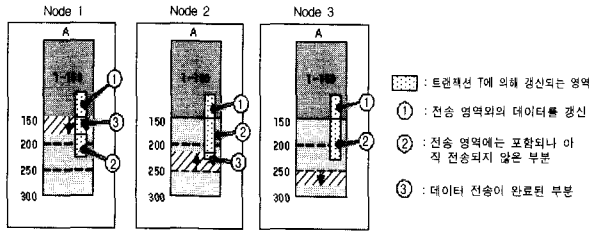
각각의 자료구조에 대한 설명은 다음과 같다.

DestNodeNum : 전송의 목적지가 되는 노드의 번호
minKey : 데이터 전송 영역에 포함되는 레코드에 속하는 기본키의 최하값
maxKey : 데이터 전송 영역에 포함되는 레코드에 속하는 기본키의 최고값
HighKey : 현재 전송 영역의 내에서 가장 최근에 전송한 데이터의 최고값
RedoLSN : 현재 전송되어야 할 재수행 로그의 최하값
HighLSN : 현재 저장되어 있는 재수행 로그의 최고값
CopyMode : 복제본의 완성 과정에서의 데이터 전송에 대한 구별값
Order : 데이터 전송의 순서. 각각 ASC와 DESC 값을 갖는다.
SrcNodeNum : 데이터를 전송하는 원본 노드의 번호
TargetReplID : 목적 복제본의 번호

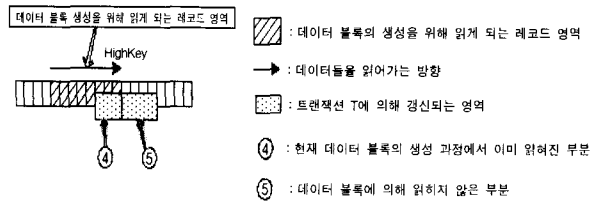
3.3.4 확장 데이터의 일관성 유지

온라인 확장 기법은 다른 트랜잭션의 처리를 수행하면서 동시에 확장 연산 과정을 완료하여야 한다. 만일, 갱신 트랜잭션이 발생하게 되면 원본 데이터의 값이 변경이 되며, 이

미 전송된 확장 데이터는 일관성을 상실하는 경우가 발생한다. 따라서, 일관성 유지를 위해 온라인 확장 중에 발생되는 트랜잭션에 대해서는 해당 갱신 데이터에 대한 정보를 확장 데이터로 보내어 최신의 값으로 반영할 수 있어야 한다. 즉, 온라인 확장 기법은 데이터 블록의 전송 시 로그 정보도 함께 보내게 되는데, 이러한 과정에서 갱신 영역에 대한 재수행 로그의 여과 과정이 필요하다.




(a) 트랜잭션의 갱신 영역별 처리

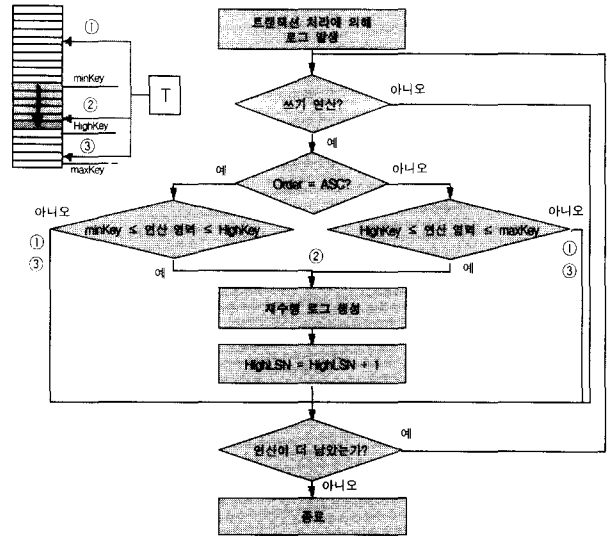



(b) 데이터 블록의 생성 과정과 트랜잭션 처리

(그림 11) 트랜잭션 처리와 일관성 유지

(그림 11)(a)는 트랜잭션의 갱신 영역에 대해 분류하고 있다. ①은 현재 온라인 확장에 의해 전송되는 데이터 영역이 아니다. 따라서 고려할 필요가 없다. ②는 전송 영역에는 포함되지만 아직 전송을 하지 않았거나, 해당 노드의 전송 영역이 아닌 경우이므로 고려할 필요가 없다. 마지막으로 ③은 현재 목적 노드로 데이터가 전송된 영역에 포함된다. 따라서 갱신 정보를 포함하는 로그를 생성하여 데이터 블록의 전송 시 함께 전송하여야 한다. (그림 11)(b)는 트랜잭션에 의해 갱신되는 영역이 현재 데이터 블록의 생성 과정에서 읽히는 레코드 영역과 겹치는 경우를 나타낸다. 두 가지 과정이 동시에 일어나게 되지만 HighKey 값을 순차적으로 갱신하는 데이터 블록의 생성 과정에 따라서 (그림 11)(a)의 예제와 마찬가지로 전송 영역의 ④와 미 전송 영역의 ⑤로 구분하여 관리한다.

재수행 로그는 노드의 메인 메모리가 충분히 크다는 가정에 의해 메인 메모리 내에서 관리된다. 데이터 블록의 전송 시 전송 부분에 대해서는 자동으로 메모리에서 삭제된다. 재수행 로그는 갱신트랜잭션에 대해서 현재 이미 전송된 데이터 영역에 대해서만 생성하도록 여과 과정을 수행한다. 예를 들어, (그림 12)에서 트랜잭션 T에 의해 갱신되는 영역 ①과 영역 ②은 전송 영역에서 벗어나므로 재수행 로그를 생성치 않는다.




(그림 12) 재수행 로그의 수행 과정

## 4. 온라인 확장 기법의 결함 허용

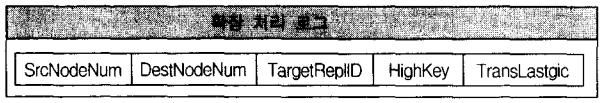
본 절에서는 온라인 확장 연산의 수행 중에 발생할 수 있는 노드의 결함과 그에 대한 해결책을 다룬다.

### 4.1 온라인 확장 연산을 위한 회복 기법

데이터베이스 시스템의 회복 처리와 온라인 확장 연산의 회복 처리는 서로 독립적으로 수행된다. 온라인 확장 연산의 회복 처리는 결함 발생 노드의 회복 과정이 모두 끝난 후부터 동작된다.

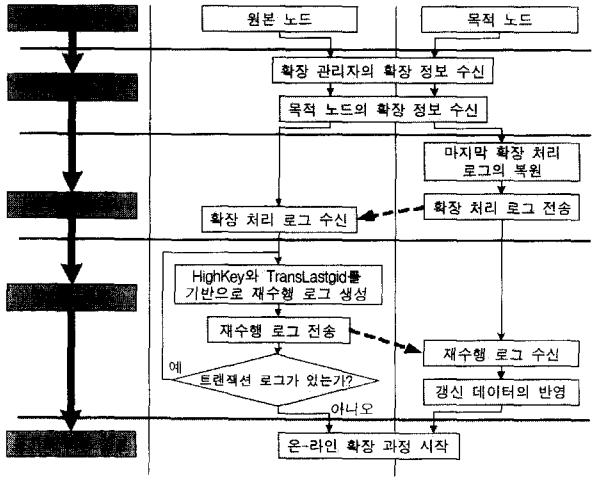
확장 정보들은 모두 메인 메모리에서 관리되기 때문에 온라인 확장 연산의 처리 도중에 노드에 결함이 발생하여 시스템이 중지되는 경우에는 온라인 확장의 진행이 불가능해진다. 온라인 확장 기법은 회복된 노드가 다시 온라인 확장에 참여할 수 있도록 확장 관리자와 자신이 원래 데이터를 전송 또는 수신하던 노드로부터 확장 정보들을 받아서 온라인 확장 과정을 재개한다. 확장 관리자를 포함하는 노드에 결함이 발생할 경우, 제안 기법에서는 확장 관리자의 복제본을 활성화하여 온라인 확장 처리를 지속시킨다. 그러나, 데이터베이스 정보와 확장 정보를 복구하더라도 결함 이전으로의 완벽한 복원이 불가능한 문제가 발생한다. 원본 노드에서 결함이 발생한 경우, 회복이 완료되는 즉시 자신의 목적 노드로 지금까지 발생한 트랜잭션에 대한 재수행 로그를 전송해야 한다. 그러나, 결함 이전의 메인 메모리에 저장되었던 재수행 로그가 파괴되어 정확한 정보의 전송이 힘들게 된다. 따라서, 원본 노드는 지금까지 전송된 영역에 대해 초기화 시키고 다시 데이터를 처음부터 보내게 된다. 목적 노드에서 결함이 발생한 경우 확장 정보가 복원이 되더라도 원본 노드와 마찬가지로 확장 정보를 메인 메모리에서 관리하기 때문에 결함 직전까지 자신이 처리한 확장 데이터에 대한 정보를 잃게 되어 원본 데이터와의 일관성

유지가 불가능하게 된다. 따라서, 결함 발생 후에도 일관성을 유지하기 위해서는 모든 과정에서 지금까지 처리된 트랜잭션 번호와 데이터베이스에 반영된 레코드의 기본키 값을 기록하고 있어야 한다. 이러한 정보를 통해 원본 노드는 자신이 보내야 할 정확한 재수행 로그를 생성할 수 있으며, 목적 노드는 정확한 갱신 연산을 수행할 수 있다.




(그림 13) 확장 처리 로그의 구조

(그림 13)은 회복 처리를 위해 새롭게 정의된 확장 처리 로그의 구조를 나타낸다. 확장 처리 로그는 데이터 블록의 수행 완료시 한 번씩 데이터베이스에 로그로 기록이 된다. 이 정보는 결함 발생시 목적 노드로부터 원본 노드로 전송되어 원본 노드에서의 동기화 작업을 수행하는데 이용이 된다. TransLastgid는 데이터베이스 클러스터에서 각 트랜잭션마다 고유하게 할당되는 gid값을 이용해 원본 데이터와 확장 데이터의 일관성 유지를 위해 사용된다. 원본 노드로부터 전송되는 재수행 로그에 해당 로그를 수행한 트랜잭션의 gid값을 포함시켜 데이터 블록의 처리가 끝날 때마다 마지막으로 수행한 재수행 로그의 gid값을 TransLast gid에 대입한다. 또한, 확장 처리 로그의 HighKey값은 마지막으로 노드에 반영된 레코드의 HighKey값을 대입한다. 노드의 결함이 발생한 후 두 가지 값은 원본 노드로 전송되어 동기화 작업에서의 데이터 전송 영역과 갱신 데이터 전송을 위한 지침이 된다. 그리고, SrcNodeNum, DestNode Num, TargetReplID를 통해 해당 확장 처리 로그의 확장 처리 프로세스를 구분한다.

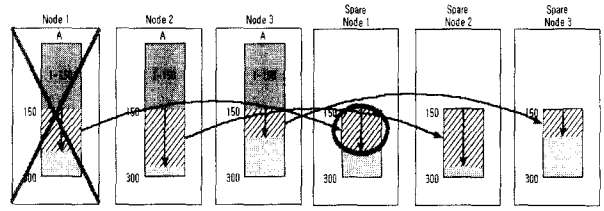



(그림 14) 온라인 확장 기법의 동기화 과정

(그림 14)는 온라인 확장 기법의 동기화 과정을 나타낸다. 온라인 확장 기법에서의 동기화 과정은 확장 처리 로그를 기준으로 결함 발생에 의해 온라인 확장의 처리가 중지된 이후 현재까지 진행된 트랜잭션을 결함 발생 노드에 반영시켜 온라인 확장의 다음 과정을 진행할 수 있도록 준비하는 과정을 뜻한다. 동기화 과정은 재수행 로그의 생성 과정을 그대로 이용하는 대신 현재 진행중인 트랜잭션이 아닌 데이터베이스에 남겨진 로그를 읽는다. 동기화 과정의 진행 도중에도 트랜잭션 로그는 계속 쌓이는데 동기화 과정이 완료 단계에 이르러 현재 진행중인 트랜잭션에 대해 실시간 처리가 충분히 가능하다고 판단되면 동기화 과정을 마치고 결함 발생 이전에 수행하던 온라인 확장의 처리 과정을 원래대로 수행한다.

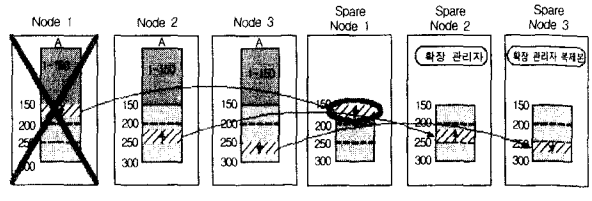
### 4.2 노드의 결함 발생에 따른 회복

일반적으로 원본 노드는 데이터베이스 시스템의 운영과 각종 서비스 제공으로 인해 자원의 사용량이 크므로 목적 노드에 비해 결함 발생 가능성이 높다. 원본 노드에 결함이 발생하게 되어 온라인 확장이 중지되면 목적 노드로의 데이터 전송이 중단되어 해당 원본 노드로부터 데이터 전송을 받던 목적 노드는 처리할 데이터가 없게 되므로 기존 온라인 확장 기법에서는 정체 현상이 발생한다.



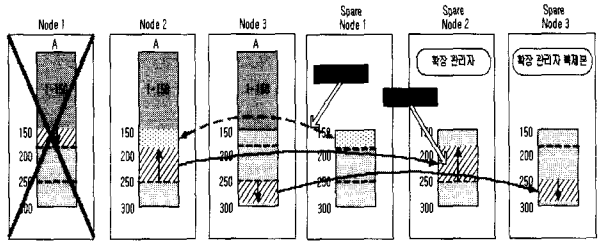

(그림 15) 기존 온라인 확장 기법에서 원본 노드의 결함 발생

(그림 15)는 노드 1에 결함이 발생하여 더 이상의 데이터 전송이 불가능한 경우를 나타낸다. 이 예제는 예비 노드 2와 예비 노드 3이 예비 노드 1의 확장이 완료될 때까지 기다리게 된다. 노드 1의 복구 완료까지 걸리는 시간이 상대적으로 길어진다면 온라인 확장의 완료 시간 또한 그만큼 길어지게 된다. 그러나, 노드 2와 노드 3은 자체적으로 예비 노드 2와 예비 노드 3으로 현재 데이터 전송과 로그 전송을 수행하는 중이므로 예비 노드 1에 대한 직접적인 데이터 전송이 불가능하다. 예비 노드 2와 예비 노드 3에서도 제안 기법과 같이 목적 노드 간에 재전송 기법이 존재하지 않는 이상 예비 노드 1로의 데이터 전송이 불가능하다.




(그림 16) 제안 기법에서 원본 노드의 결함 발생

(그림 16)은 본 논문에서 제안된 온라인 확장 기법에서 원본 노드에 결함이 발생된 경우를 나타낸다. 제안 기법은 각 원본 노드의 전송 영역을 서로 분할하여 전송하므로 그림에서와 같이 노드 1에 결함이 발생하면 예비 노드 1부터 예비 노드 3까지 노드 1의 전송 영역에 해당하는 데이터 영역은 데이터 전송을 받지 못하게 된다. 그러나, 제안 기법은 전송 영역의 재분배 기능을 이용해 노드 1에서 아직 전송하지 못한 영역에 대해 노드 2에서 전송이 가능하다. 결국 노드 1에서 복구 과정이 완료되었을 때 노드 2에서 노드 1의 나머지 부분까지 모두 전송한 경우 노드 1은 예비 노드 1과의 동기화 과정만 마치면 된다. 결과적으로 제안 기법은 기존 기법에 비해 결함 발생에 대해 보다 능동적인 대처가 가능하다. 노드 1의 복구 과정에 걸리는 시간보다 노드 2와 같은 다른 원본 노드에서 결함 발생 노드의 전송 영역에 대해 동기화 과정을 처리하는 것이 보다 효율적일 가능성이 존재한다. 이러한 점을 고려하여 이웃 노드를 통해 결함 발생 노드의 데이터 전송의 지연 문제를 해결하는 기법의 제안이 가능하다. 노드 2는 영역 재분배 과정을 거치면서 노드 1에 의해 이미 전송된 영역 외의 모든 영역에 대해서 전송이 가능하다. 이웃 노드에 의해 결함 발생 노드의 전송 영역 외의 모든 영역이 전송 완료된 후 확장 관리자는 결함 발생 노드를 원본 노드에서 제거하고 해당 이웃 노드의 전송 영역에 결함 발생 노드의 전송 영역까지 포함시켜 동기화 과정을 수행토록 한다.

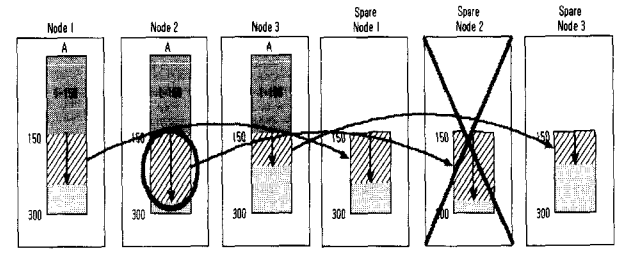



(그림 17) 이웃 노드에 의한 결함 영역의 동기화 과정

(그림 17)은 노드 1의 이웃 노드에 해당하는 노드 2에 의해 동기화 과정이 완료되는 과정을 나타낸다. 이후부터 노드 2는 두 개의 예비 노드 1에 해당하는 데이터 전송 정보와 예비 노드 2의 원래의 전송 정보를 갖게 된다. 복제본의 완성 과정 도중에 발생되는 원본 노드의 결함은 본 절에서 제안한 기법에서 영역의 재분배 과정을 제외한 모든 과정을 동일하게 적용하여 해결할 수 있다. 또한, 본 기법은 디스크의 고장, 바이러스에 의한 정보의 파괴 등의 문제로 인해 원본 노드의 복구가 불가능할 경우에도 온라인 확장을 성공적으로 완료할 수 있다.

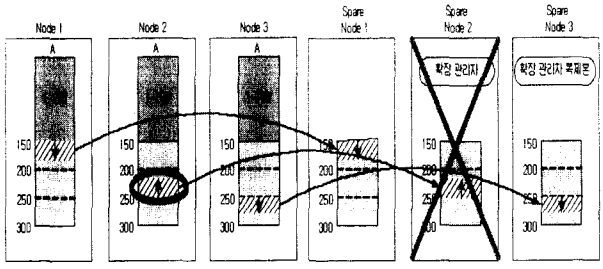
4.2.1 목적 노드의 결함

목적 노드에 결함이 발생하는 경우 원본 노드에서의 문제와는 달리 노드의 복구가 불가능한 경우를 고려하여야 한다. 원본 노드에서 결함이 발생하면 데이터베이스 클러스터는 자체적으로 내장하고 있는 회복 처리기를 통해 노드의 복구 과정을 수행한다. 임의의 노드에 대해 복구가 불가능할 경우 복제본을 이용해 서비스를 지속적으로 제공하면서 새로운 노드를 추가하고 다른 복제본들에서 새롭게 추가된 노드로 기존 노드에 있던 복제본들을 복사한다. 그러나, 온라인 확장 도중에 목적 노드에서 결함이 발생하는 경우 아직 완전히 데이터베이스 클러스터에 통합되어 운영되는 노드가 아닌 이상 이러한 복구 과정을 적용시킬 수 없다. 제안하는 온라인 확장 기법은 이러한 경우를 고려하여 두 가지 회복 기법을 제안한다. 목적 노드에서 결함이 발생하게 되면 확장 관리자는 시스템의 환경에 따라 임의로 조정되는 시간을 기준으로 회복 가능, 불가능 판단을 한다. 회복이 가능한 경우 4.2절에서와 마찬가지로 해당 노드는 확장 관리자와 자신에게 데이터를 전송하던 원본 노드로부터 확장 정보를 받게 된다. 그리고, 마지막에 저장된 TransLastgid 값을 원본 노드로 전송하여 동기화 과정을 수행한다. 그러나, 회복이 불가능한 것으로 판단이 되면 온라인 확장 기법은 복제본의 수가 셋 이상이 되는 경우 결함 발생 노드에 대해 무시하고 온라인 확장을 완료한 후에 복제본의 수를 늘리는 재조직 과정을 수행한다. 그러나, 이 기법은 온라인 확장에 소요되는 시간은 단축될 수 있으나 재조직 과정의 부가적인 과정을 필요로 한다. 다른 방법으로 새로운 예비 노드를 목적 노드로 추가하여 해당 노드에 대해 처음부터 데이터를 전송하는 방법이 있다. 그러나, 이 기법은 온라인 확장의 처리 시간을 증가시키는 문제가 있다.



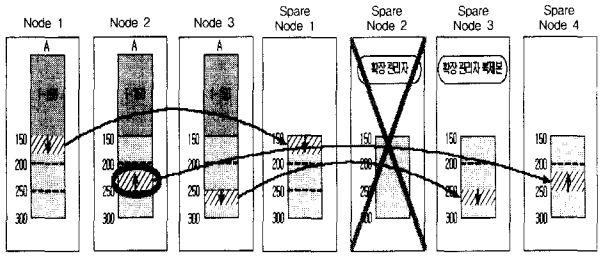

(그림 18) 기존 온라인 확장기법에서 목적노드의 결함 발생

(그림 18)은 목적 노드인 예비 노드 2에 결함이 발생한 경우를 나타낸다. 노드 2는 자신의 확장 목적이 되는 예비 노드 2가 더 이상의 데이터 전송을 받을 수 없게 되어 예비 노드 2의 복구 과정이 완료될 때까지 기다리게 된다. 결국 예비 노드 2의 복구 과정이 완료되기까지 전체적인 온라인 확장은 지체되게 되며, 처리 시간은 그만큼 지연되게 된다. 예비 노드 2의 복구가 불가능한 경우 (그림 17)과 같이 복제본의 수가 셋 이상이 되면 하나의 노드에 온라인 확장을 못하게 되더라도 두 개 이상의 노드가 데이터 전송을 끝마칠 수 있으며, 결함 발생 노드에 대해 무시하고 나머지 노드를 통해 확장을 완료할 수 있다.



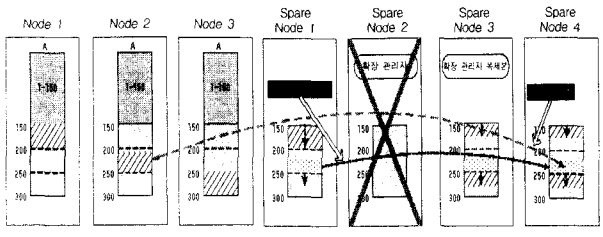

(그림 19) 제안 기법에서 목적노드의 결함 발생

(그림 19)는 목적 노드인 예비 노드 2에 결함이 발생한 경우이다. 기존 기법에서와 마찬가지로 원본 노드는 더 이상의 데이터 전송을 할 수 없게 된다. 그러나, 제안 기법은 4.2절과 마찬가지로 이웃 노드의 영역 재분배를 통해 복구 과정 중에도 결함 발생 노드의 전송 영역까지 데이터를 전송하여 목적 노드의 회복 후에는 기존 온라인 확장 기법보다 빠른 처리가 가능하다.




(그림 20) 새로운 예비 노드를 온라인 확장에 참여 시켜 처리하는 과정

(그림 20)은 목적 노드의 회복이 불가능한 경우 새로운 예비 노드를 온라인 확장에 참여시키고 원본 노드의 목적 노드를 새롭게 추가된 노드로 지정하여 데이터 전송을 재개하는 과정을 나타낸다. 이 기법은 원본 노드의 데이터 전송 크기가 복제본의 수에 비례해 줄어들게 되므로 기존 온라인 확장 기법에서 새로운 노드를 추가할 때에 비해 비용이 적게 든다. 또한, 복제본의 완성 과정에서 결함이 발생한 경우에 대해서는 (그림 21)에서와 같이 이미 다른 목적 노드로 전송된 예비 노드 2의 전송 영역에 대해서 예비 노드 1로부터 전송 받아 동기화 과정만을 수행하여 원본 노드에서의 불필요한 데이터 전송을 회피함으로써 빠른 회복 처리가 가능해진다.



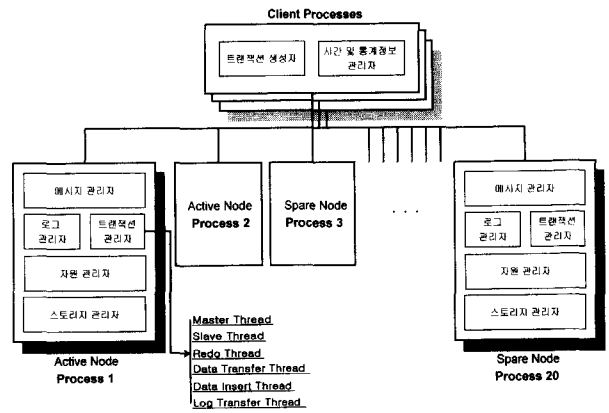

(그림 21) 목적 노드간에 데이터 복구 과정과 동기화 과정

## 5. 성능 평가

본 절에서는 실험모형을 통하여 본 논문에서 제안된 온라인 확장 기법과 기존의 온라인 확장 기법[1]의 성능을 비교 평가한다.

### 5.1 성능 평가 환경 및 방법

실험환경으로 3.1절에서 제시한 기반환경과 동일한 성능 평가 모형을 구성하였다(그림 22). 모형 시스템은 실제 구현 시스템에서 갖는 노드 수의 제한을 받지 않도록 CSIM 라이브러리를 이용하여 구현되었다[18, 20].




(그림 22) 성능 평가 모형

(그림 22)에서 클러스터내의 20개 노드에 해당하는 프로세스를 생성하며, 클라이언트 프로세스를 통해 일정 주기 간격(Wait Time)으로 트랜잭션을 생성하고 해당 트랜잭션의 수행 과정을 모니터링 한다. 각 노드 프로세스는 해당 노드의 역할에 따라 활성(Active)과 예비(Spare) 노드로 나뉘며 초기에는 활성 노드를 2개로 시작하여 10개까지 확장하는 모의 실험을 수행한다. 클라이언트 프로세서에 의해 생성된 트랜잭션의 실질적인 실행은 트랜잭션 관리자에 의해서 수행되는데, 트랜잭션 관리자는 수행하는 역할에 따라 6개 타입의 쓰레드 풀(Thread Pool)을 구성한다. 쓰레드 풀에는 원본 데이터에 발생되는 트랜잭션을 처리하는 마스터 쓰레드(Master Thread)와 복제본에 대한 전파 트랜잭션을 처리하는 슬레이브 쓰레드(Slave Thread), 온라인 확장 연산 수행시에 데이터 이동을 수행하는 데이터 이동 쓰레드(Data Transfer Thread), 로그 이동 쓰레드(Log Transfer Thread), 이동된 데이터의 삽입을 수행하는 데이터 삽입 쓰레드(Data Insert Thread), 이동된 로그를 재수행 하는 재수행 쓰레드(Redo Thread) 등으로 구성된다. 메시지 관리자와 로그 관리자, 자원 관리자, 스토리지 관리자는 FIFO 큐를 이용하여 I/O 요청 및 로크 요청 등을 순서대로 처리한다.

모의 실험에서 사용된 구체적인 매개변수와 설정 값들은 <표 1>과 같다. 주로 기존의 온라인 확장 기법에서 사용된

값을 참조하였다[1]. <표 1>에서 실험에 사용된 데이터의 크기는 273Mbyte이며, 각 레코드의 평균 크기는 4K byte이다. 트랜잭션 생성자에서 발생되는 트랜잭션은 읽기, 갱신, 삽입, 삭제 연산으로 균등하게 구성되며, 트랜잭션 당 접근되는 레코드의 수는 1~5개이다. 이러한 트랜잭션은 0ms과 20ms 내에서 임의의 간격으로 발생된다. 트랜잭션 관리자에 속하는 타입별 쓰레드의 개수는 해당 노드의 역할에 따라 차이를 갖는다. 이는 전체 쓰레드의 개수를 150개로 가정하여, 예비노드의 경우 온라인 확장에 보다 많은 자원을 할당하게 된다.

〈표 1〉 모의 실험을 위한 매개변수와 설정 값

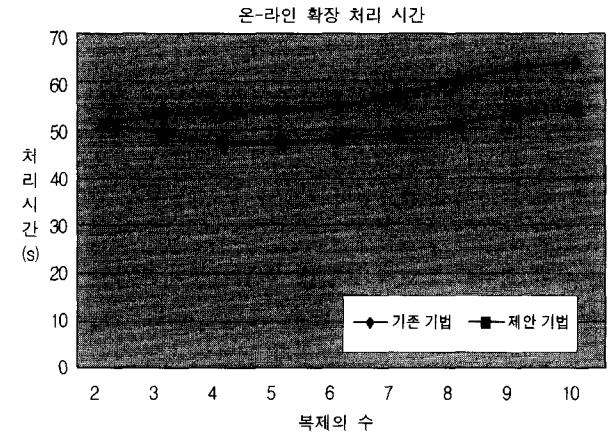
| 데이터베이스 | | |
|---|---|---|
| Records | 레코드 수 | 70,000 |
| RecordAvgSize | 레코드 평균 크기 | 4KB |
| Replicas | 복제본의 수 | 2~10 |
| KeyRanges | 기본키 값의 범위 | 1~100,000 |
| TransSize | 트랜잭션 당 접근 레코드 수 | 1~5 records |
| CommadType | 질의 타입 | Read, Update, Insert, Delete |
| CommadTypePerc | 질의 타입의 분포 | 균등분포 |
| WaitTime | 트랜잭션 발생 간격 | 0~20ms |
| RecThresholdT | 회복 한계 시간 | 30s |
| 시 스 템 | | |
| NetBandWidth | 노드간 데이터 전송 속도 | 1Gbps |
| NetDelay | 네트워 지연 시간 | 0.1ms |
| BlockSize | 데이터 블록의 크기 | 64KB |
| ActiveNodes | 활성 노드의 수 | 2~10 |
| SpaceNodes | 예비 노드의 수 | 2~10 |
| ClientNum | 클라이언트의 수 | 10 |
| CPUSpeed | 노드의 CPU 속도 | 1.5GHz |
| DiskAccess | 디스크 접근 속도 | 10ms |
| RecordAccess | 레코드 접근 속도 | 0.2ms |
| BufferHitRatio | 버퍼 적중률 | 80% |
| MasterThread | 마스터 쓰레드의 수 | 50 |
| SlaveThread | 슬레이브 쓰레드의 수 | 20 |
| DataTranThread | 데이터 이동 쓰레드의 수 | Active : 25, Spare : 50 |
| DataInsThread | 데이터 삽입 쓰레드의 수 | Active : 5, Spare : 20 |
| LogTranThread | 로그 이동 쓰레드의 수 | Active : 5, Spare : 20 |
| RedoThread | 재수행 쓰레드의 수 | Active : 5, Spare : 20 |

본 실험에 사용된 성능 평가 지수는 온라인 확장 연산의 완료시간과 트랜잭션 응답 시간이다. 즉, 복제본의 수에 따른 온라인 확장 연산의 처리 완료 시간과 온라인 확장 연산 수행 중의 트랜잭션 응답 시간, 온라인 확장 연산 수행 중의 결함 발생에 대한 회복 완료 시간이다.

### 5.1 실험 결과 평가

#### 5.1.1 복제본의 수에 따른 온라인 확장 연산의 처리 완료 시간

제안 기법과 기존 온라인 확장 기법의 처리 시간을 비교하기 위해 복제본의 수를 2개 노드에서 10개 노드까지 늘려가면서 처리 완료 시간을 측정하였다. 질의 처리의 시작 시점부터 20초 후에 온라인 확장을 수행하였으며, 실험의 신뢰성을 높이기 위해 20개의 서로 다른 seed를 이용하여 생성된 트랜잭션들을 수행한 결과들의 평균값으로 산출하였다.



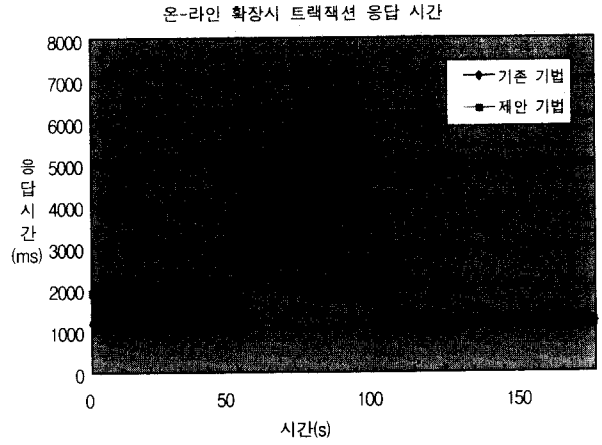

(그림 23) 온라인 확장 처리 시간

(그림 23)은 제안 기법이 기존 기법에 비해 평균 약 11%의 처리 시간 감소를 보였다. 이는 원본 노드로부터 전송되는 데이터의 크기를 줄임으로써 지연 전송에 의한 데이터 전송 크기를 줄이고, 확장 노드에서 복제본을 완성하면서 데이터 전송에 노드의 자원을 최대한 활용할 수 있게 되어 전체적인 온라인 확장의 처리 시간이 감소된 것이다. 즉, 데이터와 로그의 이동에 보다 많은 쓰레드를 할당함으로써 온라인 확장 연산의 완료 시간이 감소하였고, 복제본의 수가 많아질수록 처리 시간의 차이가 많이 나는 것은 병렬 데이터 전송에 의한 효과이다.

#### 5.1.2 온라인 확장 연산 수행 중의 평균 트랜잭션 응답 시간

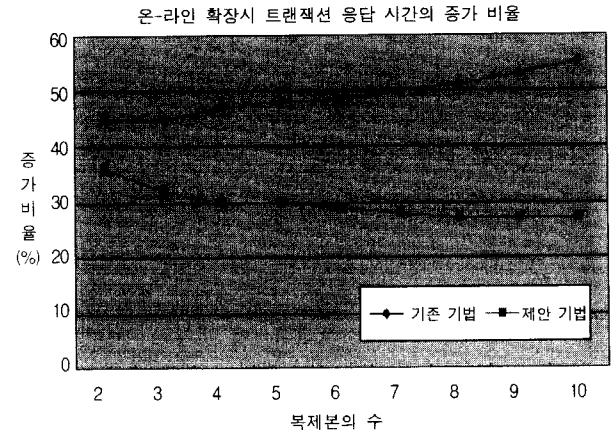
첫 번째 실험에 추가적으로, 기존 기법과 제안 기법에서 온라인 확장 연산이 다른 트랜잭션에 미치는 영향을 분석하였다. 이를 위해, 온라인 확장 연산의 수행 중에 원본 노드의 트랜잭션의 평균 응답 시간을 측정하였다. 첫번째 실험과 마찬가지로 복제본의 수를 2개 노드에서 10개 노드까지 늘려가면서 각 확장시의 트랜잭션 평균 응답시간을 측정하였다. (그림 24)는 복제본이 4개 노드에 있을 때 수행된 온라인 확장 처리 도중의 트랜잭션의 응답 시간이다.

(그림 24)는 기존 기법에 비해 제안 기법의 트랜잭션 응답 시간이 향상됨을 나타낸다. 특히 온라인 확장 연산의 중반 이후에 보다 향상된 결과를 보였다. 이러한 결과는 원본




(그림 24) 온라인 확장시 원본 노드의 트랜잭션 응답 시간

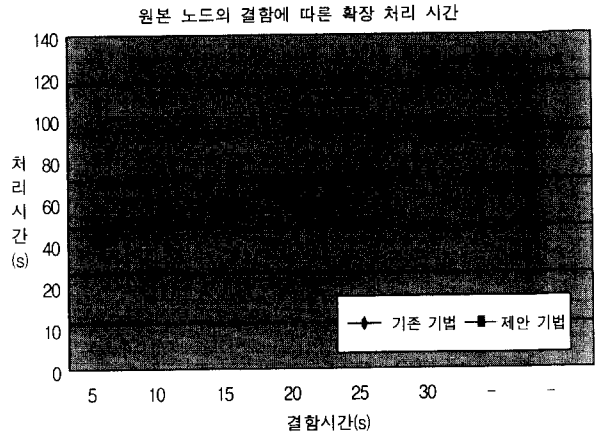
노드의 데이터 전송 영역을 노드의 수만큼 줄임으로써 일관성 유지를 위해 필요했던 재수행 로그의 생성 작업과 로그 전송 부담이 줄어들게 되었다. (그림 25)는 앞에서 수행된 실험 결과를 트랜잭션 응답 시간의 증가 비율로 나타낸 그림이다. 기존 기법은 복제본의 개수가 많을수록 원본 노드의 트랜잭션 응답 시간의 증가 비율이 상승하는데 비해 제안 기법에서는 복제본의 개수가 많을수록 트랜잭션 응답 시간의 증가 비율이 오히려 감소하였다. 즉, 제안 기법은 복제본의 개수에 상관없이 온라인 확장 연산이 원본 노드의 트랜잭션 응답시간에 크게 영향을 미치지 않았다. 따라서, 첫 번째와 두 번째 실험을 통해 제안 기법은 온라인 확장 연산의 수행 중에 발생되는 노드의 부하를 감소시켜 결함발생 가능성을 낮추고 데이터베이스 클러스터 시스템의 가용성을 향상 시켰다.



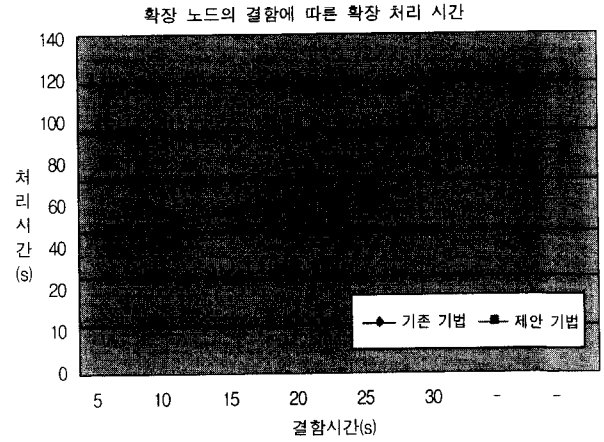

(그림 25) 온라인 확장 연산 수행 중 트랜잭션 응답시간의 증가율

5.1.3 온라인 확장 연산 수행 중의 결함 발생에 대한 회복 완료 시간

4장에서 다루었던 제안된 기법의 가용성 측면을 평가하기 위해, 본 실험은 온라인 확장 연산의 수행 중에 임의의 노드에 강제적인 결함을 발생시키고 이러한 결함에서 회복하는 회복 완료 시간을 측정하였다. 즉, 복제본의 복제 수 4대를 기준으로 온라인 확장을 수행하고 온라인 확장의 시작 시점으로부터 20초 후에 결함을 발생시켰다. 결함 발생 노드가 원본 노드인 경우와 확장 노드인 경우를 나누어서 회복 완료 시간을 측정하였다. (그림 26)는 원본 노드에서의 결함 시간에 따른 기존 기법과 제안 기법의 온라인 확장 처리 시간이고, (그림 27)은 확장 노드에서의 결함 발생에 따른 온라인 확장 기법의 처리 시간이다.




(그림 26) 원본 노드의 결함에 따른 온라인 확장 처리 시간




(그림 27) 확장 노드의 결함에 따른 온라인 확장 처리 시간

(그림 26)과 (그림 27)은 각각 원본 노드와 확장 노드에서 강제적으로 노드의 서비스를 중지시킨 상태에서 동일 복제본의 복제 수를 가지고 데이터베이스 클러스터의 결함 발생 문제에 대해 실험한 결과이다. 결함 발생 후 시스템은 30초 내에 결함 발생 노드로부터 아무런 응답이 없으면 회복 불가 판정을 내리고 다른 회복 기법을 적용한다. 본 실험에서는 데이터베이스 클러스터에서의 회복은 무시하고 온라인 확장의 처리 시간에 대해서만 다루었다. (그림 26)에서는 결함 시간이 15초가 될 때까지는 기존 기법과 별다른 차이가 없으며 20초 이후부터 기존 기법에 비해 처리

시간이 크게 증가되지 않음을 보였다. 이는 15초부터 20초 사이에 제안 기법은 영역의 재분배 과정을 거치면서 회복 처리 시간을 앞당기고, 30초부터는 결함 발생 노드에서의 전송 영역까지 다른 노드를 통해 완성하는 과정을 진행함으로써 확장 시간이 크게 증가하지만 그 이후부터 더 이상의 처리 시간 증가가 없게 된다. (그림 27)은 기존 기법과 제안 기법은 회복 불가 판정이 내려지는 30초 후부터 새로운 예비 노드를 온라인 확장에 포함시키면서 30초 이후의 처리시간이 크게 증가하였다. 기존 기법에서는 온라인 확장을 불완전하게 끝낸 후 다시 복제본 확장 과정을 완료하기까지의 처리 시간을 합쳤다. 실험의 결과로써, 제안된 기법은 영역의 재분배와 이웃 노드에서의 동기화 과정, 예비 노드 간의 데이터 복사 과정, 확장 관리자의 중복 복제 등의 기법을 이용함으로써, 기존 기법에 비해 보다 효율적인 회복 처리를 할 수 있음을 알 수 있었다.

## 6. 결 론

온라인 확장 기법은 비공유 데이터베이스 클러스터에서의 질의 패턴의 변화, 부하 집중, 처리량의 한계 도달 등의 문제점을 해결하기 위한 기법이다. 온라인 확장 기법은 온라인 상태의 시스템에 물리적으로 새로운 노드를 추가하고 데이터 재조직을 수행함으로써 작업 부하를 분산시키거나 전체 트랜잭션 처리량을 늘리기 위한 기법이다. 확장 가능한 예비 노드를 통해 확장 과정을 진행함으로써 비교적 빠르고 안정적인 부하 분산이 가능하며, 전체적인 시스템의 처리량을 개선시키는 방법이기에 다른 기법들에 비해 비공유 데이터베이스 클러스터에서 발생하는 문제점들의 근본적인 해결이 가능하다.

그러나, 기존의 온라인 확장 기법은 대량의 데이터 전송과 일관성 유지에 대한 비용이 원본 노드에 크게 작용함에 따라 이미 트랜잭션 처리로 인해 과부하 상태에 있는 노드에 부가적인 처리량을 주게 되어 트랜잭션 처리량을 감소시켰다. 이는 동일 복제본을 수용하는 각각의 원본 노드에서 같은 데이터를 중복하여 전송하는 문제로 인해 발생한다. 또한, 원본 노드의 부하 증가로 인해 노드의 결함 발생 가능성이 높아지게 되었지만, 기존 기법은 결함 발생의 경우 온라인 확장을 완전히 취소하여 가용성에 있어서 큰 문제를 가지고 있었다. 즉, 기존의 온라인 확장 기법은 확장 연산 전체를 하나의 트랜잭션으로 간주하기 때문에 확장 시 발생된 문제에 대해서 확장 과정 중에 생성된 모든 정보를 확장 이전의 상태로 돌려놓는 철회(undo) 과정을 수행한다.

본 논문에서는 비공유 데이터베이스 클러스터에서 고 가용성을 위한 온라인 확장 기법을 제안하였다. 제안 기법은 데이터 전송을 병렬로 수행하면서 각 원본 노드의 전송 영역의 크기를 줄여 노드의 부하를 줄이고 트랜잭션에 대한 간섭을 최소화하면서 예비 노드에서 복제본 구성을 완료하도록 처리하여 원본 노드의 부하를 줄이는 동시에 예비 노드의 자원을 최대한 활용하였다. 또한, 온라인 확장 도중에 발생하는 노드의 결함에 대한 회복 기법을 제안하고 이 회복 기법이 병렬 데이터 전송 과정을 거치는 본 논문의 제안 구조에서 영역의 재분배, 이웃 노드에서의 동기화 과정, 예비 노드간의 데이터 복사 과정, 그리고 확장 관리자의 중복 복제 등의 기법을 이용함으로써 보다 효율적인 회복을 수행하도록 하였다. 본 논문에서는 실험 모형에 기반 한 성능 평가를 통해, 제안된 기법이 원본 노드의 부하를 줄여 트랜잭션 처리 속도가 기존 기법에 비해 향상됨을 보였으며, 기존의 기법에 비해 개선된 회복 처리 성능을 보였다. 따라서, 본 논문에서 제안된 온라인 확장 기법은 비공유 데이터베이스 클러스터에서 보다 높은 확장성과 가용성을 보장하는데 기여하였다.

## 참 고 문 헌


[1] S. E. Bratsberg and R. humborstad, "Online Scaling in a Highly Available Database," Proceedings of the 27th International Conference on Very Large Databases, pp.451-460, Sept., 2001.

[2] Y. Breitbart and H. F. Korth, "Replication and consistency : Being lazy helps sometimes," Proceedingsof the 16th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pp.173-184, May, 1997.

[3] J. Gray, P. Helland, P. O'Neil and D. Shasha, "The dangers of replication and a solution," Proceedingsof the ACM SIG MOD International Conference on Management of Data, pp. 173-182, Jun., 1996.

[4] Informix, "Informix Extended Parallel Server 8.3," Informix, http://www.Informix.com.

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

[6] Y. Jang, C. Lee, J. Lee and H. Bae, "Improved On-line Scaling Scheme in a Scalable and Highly Available Database," Proceedings of the International Conference PDPTA '02, Las Vegas, Nevada, USA, pp.1345-1351, Jun., 2002.

[7] B. Kemme and G. Alonso, "Don't be lazy, be consistent : Postgres-R, a new way to implement database replication," Proceedings of the International Conferenceon Very Large Databases, Cairo, Egypt, pp.134-143, Sept., 2000.

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

[9] B. Kemme, A. Bartoli and O. Babaoglu, "Online reconfiguration in replicated databases based on group communication," Proceedingsof the International Conference on Dependable Systems and Networks, Goteborg, Sweden, pp. 117-126, Jul., 2001.

[10] B. Kemme and G. Alonso, "A New Approach to Developing

and Implementing Eager Database Replication Protocols," ACM Transactions on Databases Systems, Vol.25, No.3, pp.333-379, Sept., 2000.

[11] M. L. Lee, M. Kitsuregawa and B. C. Ooi, "Towards Self-Tuning Data Placement in Parallel Database Systems," Proceedings of ACM SIGMOD International Conference on Management of Data, Dallas, TX USA, pp.225-236, May, 2000.

[12] M. Patino Martnez, R. Jimenez-Peris, B. Kemme and G. Alonso, "Scalable replication in database clusters," Proceedings of the 14th International Symposium on Distributed Computing, Toledo, Spain, pp.315-329, Oct., 2000.

[13] M. T. Ozsu and P. Valduriez. "Principles of Distributed Database System," Prentice-Hall, 1999.

[14] E. Pacitti, P. Minet and E. Simon, "Fast algorithms for maintaining replica consistency in lazy master replicated databases," Proceedings of International Conference on Very Large Database (VLDB 1999), Edinburgh, pp.126-137, Sept., 1999.

[15] R. Jimnez-Peris, M. Patio-Martnez, B. Kemme and G. Alonso, "Improving the Scalability of Fault-Tolerant Database Clusters," Proceedings of the 22rd International Conference on Distributed Computing Systems, pp.477-484, Jul., 2002.

[16] N. Ponnekanti and H. Kodavalla, "Online Index Rebulid," Proceedings of ACM SIGMOD International Conferenceon Management of Data, Dallas, TX USA, pp.529-538, May, 2000.

[17] G. H. Sockut and B. R. Iyer, "A survey of online reorganization in IBM products and research," IEEE Data Engineering Bulletin, Vol.19, No.2, pp.4-11, Mar., 1996.

[18] H. Schwtman, "CSIM User Guide : CSIM 19 Simulation Engine(C, C++ Version)," Mesquite Software, Inc., 2001.

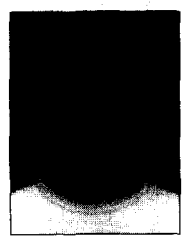
[19] J. Wang, M. Miyazaki, H. Kameda and J. Li, "Improving Performance of Parallel Transaction Processing Systems by Balancing Data Load on Line," Proceedings of 7th International Conference on Parallel and Distributed Systems, Iwate, pp.331-338, Japan, Jul., 2000.

[20] K. Watkins, "Discrete event simulation in C," McGraw-Hill, 1993.

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

[22] R. Vandewall, "Database Replication Prototype," Masters thesis, Department of Mathematics and Computer Science, University of Groningen, Netherlands, 2000.

[23] C. Zou and B. Salzberg, "On-line reorganization of sparsely-populated B+-trees," Proceedings of ACM SIGMOD International Conference on Management of Data, Montreal, Canada, pp.115-124, Jun., 1996.



**이 충 호**

e-mail : chlee@dblab.inha.ac.kr

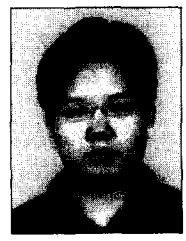
1997년 인하대학교 전자계산공학과(공학사)

1999년 인하대학교 대학원 전자계산공학과 (공학석사)

2003년 인하대학교 대학원 전자계산공학 (공학박사)

2003년~현재 인하대학교 지능형 GIS 연구센터 연구원

관심분야 : 데이터베이스 클러스터, 분산 데이터베이스, 공간 데이터베이스 등



**장 용 일**

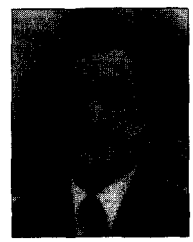
e-mail : himalia@dblab.inha.ac.kr

1997년 인하대학교 전자계산공학과(공학사)

2001년 인하대학교 컴퓨터공학부(공학석사)

2003년~현재 인하대학교 컴퓨터공학부 박사과정

관심분야 : 웹 & 모바일 GIS, 공간 데이터베이스 클러스터, 위치기반 서비스



**배 해 영**

e-mail : hybae@inha.ac.kr

1974년 인하대학교 응용물리학과(공학사)

1978년 연세대학교 대학원 전자계산학과 (공학석사)

1989년 숭실대학교 대학원 전자계산학과 (공학박사)

1985년 Univ. of Houston 객원 교수

1992년~1994년 인하대학교 전자계산소 소장

1982년~현재 인하대학교 전자계산공학과 교수

1999년~현재 지능형 GIS 연구센터 소장

2000년~현재 중국 중경우전대학교 대학원 명예 교수

관심분야 : 분산 데이터베이스, 공간 데이터베이스, 지리정보 시스템, 멀티미디어 데이터베이스 등