비공유 공간 데이터베이스 클러스터에서 최신버전의 클러스터 로그를 이용한 회복기법

[0]장일국*, 장용일*, 박순영*, 배해영*
*인하대학교 전자계산학과
e-mail: j98221042@dblab.inha.ac.kr

Recovery Method Using Recently Version Based Cluster Log in Shared-Nothing Spatial Database Cluster

[0]Il-Kook Jang*, Yong-Il Jang*, Soon-Young Park*, Hae-Young Bae*
*Dept. of Computer Science and Engineering, Inha University

요 약

회복기법은 비공유 공간 데이터베이스 클러스터에서 고가용성을 위해 매우 중요하게 고려되고 있다. 일반적으로 데이터베이스 클러스터의 회복기법은 노드의 오류가 발생한 경우 로컬 로그와는 별도로 클러스터 로그를 생성하며, 이를 기반으로 해당 노드에서의 회복과정을 수행한다. 그러나, 기존의 기법은 하나의 레코드를 위해 다수의 갱신정보를 유지함으로써 클러스터 로그의 크기가 증가되고, 전송비용이 증가된다. 이는 회복노드에서 하나의 레코드에 대해 여러 번의 불필요한 연산을 실행하여 회복시간이 증가되고, 전체적인 시스템의 부하를 증가시키는 문제를 발생시킨다.

본 논문에서는 비공유 공간 데이터베이스 클러스터에서 최신버전의 클러스터 로그를 이용한 회복기법을 제안한다. 제안기법에서의 최신버전의 클러스터 로그는 레코드의 변경사항과 실제 데이터를 가리키는 포인터 정보로 구성되고, 하나의 갱신정보를 유지함으로써 클러스터 로그의 크기가 감소하며, 전송비용이 감소한다. 회복노드에서는 하나의 레코드에 대해 한번의 갱신연산만 실행하므로 빠른 회복이 가능하며, 시스템의 가용성을 향상시킨다.

1. 서론

최근 무선 인터넷과 모바일 장치가 발전하고 대중화됨에 따라 지리정보와 같은 공간 데이터를 제공하는 서비스가 증가 하였다. 공간 데이터의 특성상 대용량의 데이터를 관리하고 빠르게 처리할 수 있는 데이터베이스가 요구되었으며, 이러한 서비스를 이용하는 사용자의 급증에 따른 높은 동시처리량과 고가용성이 요구되었다 [1]. 비공유 구조의 공간 데이터베이스 클러스터는 독립적으로 동작하는 노드들을 고속의 네트워크로 연결시키는 구조를 가지며 분할 정책과 복제 정책의 사용으로 사용자에 대한 질의응답 속도를 빠르게 제공하는 고성능, 예측할 수 없는 사용자를 수용하기 위한 고확장성, 급격한 사용자 증가에 장애가 발생하여도 지속적인 서비스를 제공하는 고가용성을 제공한다 [2,3]. 특히 고가용성을 위하여 비공유 공간 데이터베이스 클러스터에서 회복기법은 중요하게 고려되고 있다. 일반적으로 데이터베이스 클러스터의 회복기법은 노드의 오류가 발생한 경우 노드 자신의 회복을 위한 로컬 로그와 클러스터 구성에 대한 회복을 위해 클러스터 로그를 생성하며, 이를 기반으로 해당 노드에서의 회복과정을 수행한다. 기존의 회복 기법은 하나의 레코드를 위해 다수의 갱신정보를 클러스터 로그에 유지함으로써 클러스터 로그의 크기가 증가되고, 전송비용이 증가된다. 이는 회복노드에서 하나의 레코드에 대해 여러 번의 불필요한 갱신연산을 수행하여 회복시간이 증가되고, 전체적인 시스템의 부하를 증가시키는 문제를 발생시킨다.

본 논문에서는 비공유 공간 데이터베이스 클러스터에서 최신버전의 클러스터 로그를 이용한 회복기법을 제안한다. 제안기법에서의 최신버전의 클러스터 로그는 레코드의 변경사항과 실제 데이터를 가리키는 포인터 정보로 구성되고, 하나의 갱신정보를 유지함으로써 클러스터 로그의 크기가 감소하며, 전송비용이 감소한다. 회복노드에서는 하나의 레코드에 대해 한번의 갱신연산만 실행하므로 빠른 회복이 가능하며, 시스템의 가용성을 향상시킨다.

본 논문의 내용 구성은 다음과 같다. 2장에서 관련연구를 다룬다. 3장에서 최신버전의 클러스터 로그의 구조, 기록, 전송 등 각 구성에 대하여 설명하고, 4장에서는 각 구성을 기반으로 최신버전의 클러스터 로그를 이용한 회복기법을 기술한다. 마지막으로 5장에서 결론을 내린다.

2. 관련연구

본장에서는 ClustRa의 회복기법을 설명하고, 본 논문의 기반이 되는 Cluster의 전체적인 시스템 구조와 회복기법에 대하여 설명한다.

2.1 ClustRa의 회복기법

ClustRa는 독립적인 질의 처리가 가능한 노드를 고속의 네트워크로 연결한 메인 메모리 기반의 DBMS이다. ClustRa의 그룹에는 마스터 노드와 백업노드가 존재하며, 그룹 내에 적용되는 분할정책과 복제정책의 사용으로 고성능, 고가용성을 지원한다 [5,6]. ClustRa에서의 회복기법은 노드 자체의 일관성 유지를 위한 내부로그와 노드 간 일관성 유지를 위한 분산로그를 이용하여 회복을 수행한다. 노드 자체의 회복이 완료되기 전까지 빈번한 갱신연산이 발생할 경우 분산로그의 크기가 기하급수적으로 증가하고, 분산로그를 회복노드에 전송하는 비용이 증가한다. ClustRa에서의 회복과정은 내부

1) 본 연구는 대학 IT연구센터 육성·지원사업의 연구결과로 수행되었음

로그를 이용하여 자체회복을 수행한 후 분산로그를 이용하여 클러스터 구성을 위한 회복을 수행한다. 그러나 자체회복이 오래 걸릴 경우 분산로그의 크기가 증가하게 되고, 분산로그를 순차적으로 전송하므로 회복시간이 느려지게 된다.

2.2 Cluster의 개요

본 논문에 기반이 되는 시스템은 비공유 공간 데이터베이스 클러스터인 GMS/Cluster 이며, GMS/Cluster의 구조는 [그림 1]과 같다.

[그림 1] GMS/Cluster의 구조

비공유 구조의 공간 데이터베이스 클러스터는 기본적으로 노드와 그룹, 유휴 노드로 구성된다. 노드는 독립적인 DBMS 기능을 수행하며, 각각의 노드들은 고속의 네트워크로 연결되어 있다 [4,9]. 그룹은 2~4개의 노드로 구성되며, 그룹 내에서는 하나의 마스터 노드와 여러 개의 복사본 노드가 존재 한다. 유휴 노드는 아무런 처리도 하지 않는 노드로 활동 노드의 붕괴 시 대체노드나 온라인 확장 등에 사용된다. 그룹 내에 적용되는 복제정책은 각 데이터에 대한 복사본을 다른 노드에서 유지하는 것으로 질의를 처리하던 노드에 오류가 발생할 경우 해당 노드의 데이터 복사본을 유지하는 다른 노드가 서비스를 지속할 수 있는 고가용성을 제공한다. 그룹 간에 적용되는 분할정책은 하나의 데이터를 몇 개의 작은 데이터로 분할하여 각각 다른 노드에서 관리하는 것으로 대용량 데이터의 효율적인 관리와 데이터의 갱신연산에 대하여 동시 처리량을 향상시킴으로서 고성능을 제공한다 [7,8]. 노드에 부하가 집중되는 경우 유휴 노드가 클러스터에 추가되어 활성화 상태가 되고 기존에 분할 및 복제되어 있던 데이터를 재배치함으로써 예측할 수 없는 사용자를 수용할 수 있는 고확장성을 제공한다 [7,9].

2.3 Cluster의 회복기법

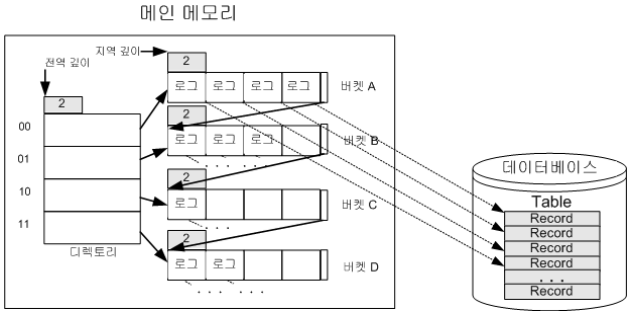
GMS/Cluster는 클러스터 로그를 이용함으로써 결함 허용과 오류가 발생한 노드의 회복을 지원한다 [9]. 특정 노드의 오류를 즉시 감지하기 위하여 그룹을 형성할 때 노드 간 망형 연결을 구축하고, 특정 노드에서 오류를 발견하면 그룹 내의 나머지 노드들은 이를 바로 알게 되어 오류가 발생된 노드의 대체노드를 결정한다. 그룹 내 나머지 노드들은 클러스터 로그를 기록하기 시작하고, 오류가 발생한 노드가 재 시작을 하면 나머지 노드들은 자신이 갖고 있는 클러스터 로그를 전송함으로써 오류가 발생한 노드의 회복을 지원한다. 오류가 발생한 노드의 회복과정은 오류가 발생한 노드가 회복되어 다시 그룹에 참여하는 경우와 유휴노드가 그룹에 참여 하는 경우가 있다 [9]. 오류가 발생한 노드가 회복되어 그룹에 참여하는 경우는 로컬로그를 이용하여 자체회복을 수행한 다음 클러스터 구성의 회복을 위하여 그룹 내 다른 노드에게 클러스터 로그를 전송 받아 회복을 수행한다. 유휴노드가 그룹에 참여하는 경우는 오류가 발생한 노드를 대신하여 유휴 노드중 하나를 선택하여 그룹의 일원으로 대체 시키고, 유휴 노드에게 데이터베이스의 내용을 완전 복제 시킨 후 클러스터 그룹의 일원으로 서비스를 시작하게 된다. 오류가 발생한 노드는 시스템 관리자에 의해 데이터베이스 초기화를 수행하여 유휴노드로 재시작을 하게 된다.

3. 최신버전의 클러스터 로그 기록 및 전송

본장에서는 최신버전의 클러스터 로그 구조를 설명하고, 최신버전의 클러스터 로그 기록 및 전송에 대하여 설명한다.

3.1 최신버전의 클러스터 로그 구조

특정 노드에서 오류가 발생한 경우 로컬 로그와는 별도로 최신버전의 클러스터 로그를 생성하며, 이를 기반으로 오류가 발생한 노드에서 회복을 수행한다. [그림 2]는 최신버전의 클러스터 로그를 효율적으로 관리하기 위하여 확장성 해싱을 이용한 클러스터 로그 관리 구조이다.



[그림 2] 확장성 해싱을 이용한 클러스터 로그 관리 구조

확장성 해싱을 이용한 최신버전의 클러스터 로그는 메인 메모리에서 관리 되고, 구성되는 정보는 다음과 같다. 전역깊이는 디렉토리에 대한 인덱스로써 현재 디렉토리의 크기를 나타내고, 지역깊이는 해당 버켓의 오버플로우 발생 여부를 나타내기 위해 유지된다. 디렉토리는 버켓을 가리키는 포인터를 저장하고, 버켓은 최신버전의 클러스터 로그를 유지하며, 버켓간 연결 리스트를 이용하여 순차적인 접근이 가능하다. 기록된 최신버전의 클러스터 로그는 레코드의 변경사항과 데이터베이스에 유지되어 있는 실제 데이터를 가리키고, 데이터베이스는 연산이 발생하는 실제 데이터를 관리한다. 최신버전의 클러스터 로그 관리는 연산이 발생한 해당 레코드의 RID를 해쉬함수에 적용한 결과의 이진표현으로 디렉토리 주소를 나타내고, 디렉토리에 저장되어 있는 포인터가 가리키는 버켓에 최신버전의 클러스터 로그를 유지한다. 여러 번의 연산이 발생할 경우 디렉토리는 증가하지만 감소하지 않으며, 버켓은 로그에 따라 증가와 감소하는 구조를 가진다. 다음 [그림 3]은 최신버전의 클러스터 로그의 구조이다.

연산	클러스터 로그	
삽입(INSERT)	I	RID

연산	클러스터 로그		
갱신(UPDATE)	U	RID	Old PK

연산	클러스터 로그	
삭제(DELETE)	D	PK

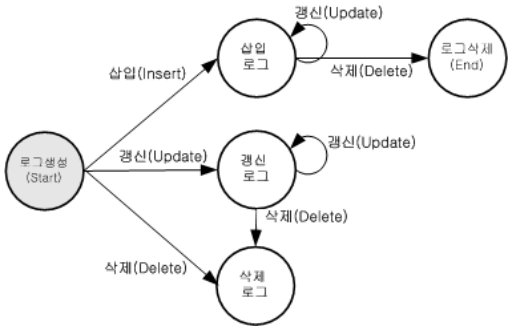
I : Insert U : Update
D : Delete PK : Primary Key

[그림 3] 최신버전의 클러스터 로그 구조

발생하는 연산에 따라 최신버전의 클러스터 로그에 유지되는 정보가 달라진다. Insert 연산이 발생한 경우 레코드의 식별자인 RID와 변경사항을 나타내는 Insert 플래그를 유지한다. Update 연산이 발생한 경우 새로운 갱신 레코드의 식별자인 RID와 변경사항을 나타내는 Update 플래그와 갱신 대상의 Old Primary Key를 유지한다. Delete 연산이 발생한 경우 변경사항을 나타내는 Delete 플래그와 삭제 대상이 되는 Old Primary Key를 유지한다.

3.2 최신버전의 클러스터 로그 기록

최신버전의 클러스터 로그 기록은 다수의 갱신 연산이 발생하여도 최신의 변경사항 하나만을 유지한다. 다음 [그림 4]는 클러스터 로그 생성 후 다수의 연산이 발생하여도 최신의 변경사항 하나만을 유지하는 최신버전의 클러스터 로그를 나타낸다.



a) 클러스터 로그 생성 후 발생하는 연산

	기존에 기록한 클러스터 로그			
새로운 로그	–	삽입 로그	갱신 로그	삭제 로그
삽입 로그	삽입 로그	X	X	※
갱신 로그	갱신 로그	삽입 로그	갱신 로그	X
삭제 로그	삭제 로그	로그 삭제	삭제 로그	X

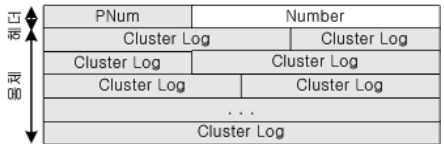
b) 최신버전의 클러스터 로그 내용

[그림 4] 최신버전의 클러스터 로그

클러스터 로그 생성 후 관리되는 연산과정은 Insert, Update, Delete 연산이 발생할 수 있으며, 시작은 클러스터 로그의 생성을 나타낸다. Insert 연산 이후 발생하는 Update 연산은 Insert 로그를 유지하고, 삭제연산은 기록된 로그를 삭제한다. Update 연산 이후 발생하는 Update 연산은 Update 로그를 유지하고, Delete 연산은 Delete 로그를 유지한다. Delete 연산이 발생한 후 더 이상 연산이 발생하지 않는다.

3.3 최신버전의 클러스터 로그 전송

오류가 발생한 노드는 클러스터 구성에 대한 회복을 위하여 그룹 내 다른 노드들과 연결하여 클러스터 로그를 요청한다. 최신버전의 클러스터 로그의 전송은 고속의 네트워크를 통해 패킷 단위로 전송하며, 최신버전의 클러스터 로그 전송 자료구조는 다음 [그림 5]와 같다.



[그림 5] 최신버전의 클러스터 로그 전송 자료구조

최신버전의 클러스터 로그 전송 자료구조는 헤더와 몸체로 구성된다. 헤더는 클러스터 로그를 순차적으로 전송하는데 필요한 패킷번호를 유지하며, 몸체는 변경된 실제 데이터를 유지하고 있는 최신버전의 클러스터 로그를 순차적으로 패킷의 몸체 크기만큼 복사한다. 헤더의 <PNum>는 클러스터 로그 전송 패킷번호를 나타내는 것으로 패킷의 직렬성을 보장하기 위하여 사용하고, 전송되는 Cluster Log는 발생되는 연산에 따라 전송되는 정보가 달라진다. 연산에 따른 최신버전의 클러스터 로그 전송 정보는 다음 [그림 6]과 같다.

기록된 로그	전송 정보
삽입(INSERT)	Data

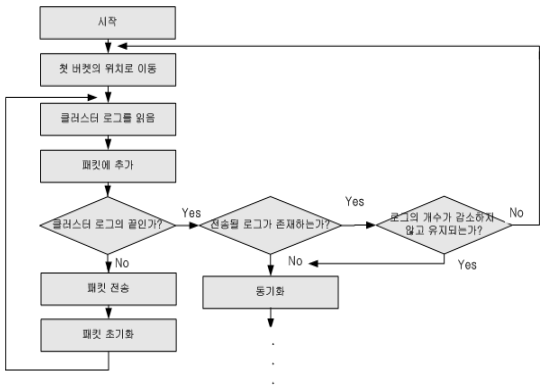
기록된 로그	전송 정보	
갱신(UPDATE)	Old PK	Data

기록된 로그	전송 정보
삭제(DELETE)	PK

PK : Primary Key

[그림 6] 연산에 따른 최신버전의 클러스터 로그 전송 정보

전송되는 Data는 최신버전의 클러스터 로그에 기록된 RID가 가리키는 실제 Data와 Primary Key를 유지한다. 삽입연산에 따른 최신버전의 클러스터 로그는 Data를 전송하고, 갱신연산에 따른 최신버전의 클러스터 로그는 이전 데이터의 Old Primary Key와 갱신된 Data를 전송하며, 삭제연산에 따른 최신버전의 클러스터 로그는 삭제시킬 데이터의 Primary Key를 전송한다. 다음 [그림 7]은 최신버전의 클러스터 로그 전송과정을 나타낸다.



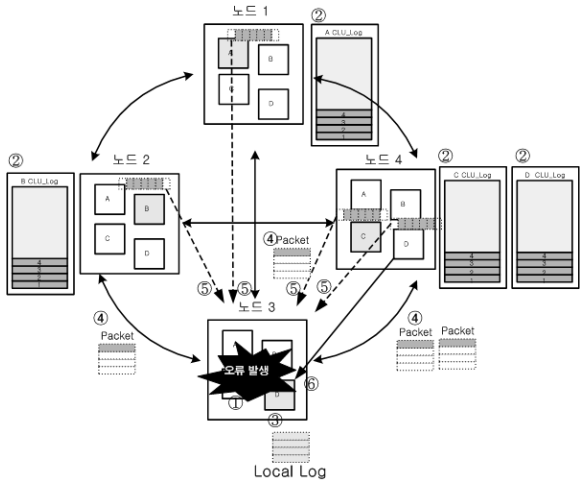
[그림 7] 최신버전의 클러스터 로그 전송

마스터 노드에서 각 버켓에 저장되어 있는 최신버전의 클러스터 로그를 연결 리스트를 이용하여 순차적으로 읽어 회복노드에게 패킷단위로 전송한다. 회복노드로부터 확인응답을 받은 패킷은 초기화를 수행하고, 전송 중에 발생하는 트랜잭션은 해당 마스터 테이블에서 처리한 후 최신버전의 클러스터 로그를 남긴다. 패킷의 끝에 도착하면 전송되어야할 최신버전의 클러스터 로그의 개수를 검사하여 전송되어야할 최신버전의 클러스터 로그가 존재한다면 첫 버켓의 위치로 돌아가 최신버전의 클러스터를 패킷단위로 전송한다. 전송되어야할 최신버전의 클러스터 로그가 더 이상 없거나, 최신버전의 클러스터 로그의 개수가 더 이상 줄어들지 않고 일정한 수치를 유지 하게 되면 동기화 과정을 수행한다.

4. 최신버전의 클러스터 로그를 이용한 회복

특정 노드에서 오류가 발생한 경우 최신버전의 클러스터 로그를 기록하고, 자체회복과 클러스터 구성에 대한 회복을 수행한다. 동기화 과정을 수행하여 일관성을 유지하고, 오류발생 이전 상태로 전환하여 정상적인 서비스를 재개한다. 다음 [그림 8]은

최신버전의 클러스터 로그를 이용한 회복과정을 나타낸 것이다.



[그림 8] 최신버전의 클러스터 로그를 이용한 회복

노드의 오류 발생

Cluster는 특정 노드의 오류를 즉시 감지하기 위하여 그룹 간 망형 연결을 구축한다. 하나의 노드에서 오류가 발생한 경우 오류가 발생한 노드는 서비스가 중단되지만, 복사본 노드가 역할을 대신함으로써 지속적인 서비스를 제공한다.

최신버전의 클러스터 로그 기록

자신의 노드에 해당하는 마스터 테이블에서 독립적으로 최신버전의 클러스터 로그를 관리한다. 또한, 오류가 발생한 노드에 존재하던 마스터 테이블에 대하여 임시 마스터 테이블을 설정하는데, 임시 마스터 테이블에 대하여 최신버전의 클러스터 로그를 남긴다. 최신버전의 클러스터는 특정 노드에서 오류가 발생한 시점부터 트랜잭션에 따라 기록되며, 확장성 해싱을 이용하여 메인 메모리에 유지한다.

오류가 발생한 노드의 자체회복

오류가 발생한 노드는 로컬로그를 이용하여 자체회복을 수행한다. 로컬로그는 노드가 가진 데이터에 대해 연산이 발생한 경우 기록되는 로그이며, 기존의 단일 데이터베이스의 회복과 같은 방법을 사용한다. 노드의 자체회복이 끝나게 되면 자신의 데이터에 대해 일관성을 유지한 상태가 되고, 클러스터 구성에 대한 회복을 위해 그룹 내 다른 노드들에게 최신버전의 클러스터 로그를 요청한다.

최신버전의 클러스터 로그 전송

그룹 내 나머지 노드들은 오류가 발생한 노드가 복원된 것을 감지하고 각 버켓에 저장되어 있는 최신버전의 클러스터 로그를 연결 리스트를 이용하여 순차적으로 읽어 패킷단위로 전송한다. 전송되어야할 최신버전의 클러스터 로그가 존재하지 않거나, 최신버전의 클러스터 로그의 개수가 줄어들지 않고 일정한 수치를 유지한다면 동기화 과정을 수행한다.

동기화 과정

동기화 과정은 마지막 클러스터 로그 패킷을 전송한 후부터 모든 클러스터 로그의 반영이 끝날 때 까지 발생하는 트랜잭션에 대해 일관성을 유지시키는 과정이다. 동기화 과정동안 마스터 노드는 트랜잭션에 대한 처리를 잠시 멈추는 일시적인 트랜잭션 대기상태가 되고, 대기상태 동안 발생하는 모든 트랜잭션은 마스터 노드의 큐(Queue)에 유지된다. 마지막 클러스터 로그 패킷을 회복노드로 전송하고, 트랜잭션 대기 상태 동안 큐에 유지되어 있는 트랜잭션을 회복노드로 전송하여 그룹 내 다른 노드들과 일관성을 유지하게 된다.

정상적인 서비스 재개

모든 최신버전의 클러스터 로그를 반영한 후 동기화 과정을 수행한 테이블은 오류 발생 이전의 상태로 돌아가야 한다. 오류 발생 이전의 테이블 구성으로 전환하는 데에는 오류가 발생하기 이전의 테이블 역할에 따라 나누어진다. 마스터 테이블에서 오류가 발생한 경우 클러스터 로그를 이용한 회복을 수행한 후 동기화 과정을 거쳐 자신이 마스터 테이블로서 모든 트랜잭션에 대해 정상적인 서비스를 재개 한다. 복사본 노드에서 오류가 발생한 경우 클러스터 로그를 이용한 회복을 수행한 후 동기화 과정을 거쳐 백업 노드로써 마스터 노드에서 전파되는 질의를 수행한다.

5. 결론

본 논문에서는 비공유 데이터베이스 클러스터에서 최신버전의 로그를 이용한 회복기법을 제안하였다. 제안기법에서의 최신버전의 클러스터 로그는 레코드의 키값을 이용한 확장성 해싱을 기반으로 해당 클러스터 로그의 위치를 빠르게 검색하고, 레코드의 변경사항과 실제 데이터를 가리키는 포인터 정보로 구성된다. 하나의 레코드를 위해 최신버전의 클러스터 로그 하나만을 유지하므로 클러스터 로그의 크기가 감소하고, 전송비용이 감소한다. 회복노드에서는 하나의 레코드에 대해 한번의 갱신연산만 실행하므로 빠른 회복이 가능하며, 시스템의 가용성을 향상시킨다.

향후연구는 제안된 최신버전의 클러스터 로그에 대한 성능 평가가 필요하다.

참고문헌

[1] R. H. Guting, and et. al, "A Foundation for Representing and Queryin g Moving Objects", ACM Transactions on Database Systems, Vol. 2 5, No. 1, pp. 1-42, 2000

[2] Roger Bamford, Rafiul Ahad, Angelo Pruscino, "A Scalable and Hi ghly Available Networked Database Architecture", Proceedings of t he 25th VLDB Conference, 1999

[3] Roel Vandewall, "Database Replication Prototype", Masters thesis, D epartment of Mathematics and Computer Science, University of Gron ingen, The Netherlands, 2000

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

[5] Svein-Olaf Hvasshovd, Oystein Torbjornsen, Svein Erik Bratsberg, "The ClustRa Telecom Database: High Availability, High Throughp ut, and Real-Time Response", Proceedings of the 21st VLDB Co nference, 1995

[6] Oystein Torbjornsen, Svein-Olaf Hvasshovd, Young-Kuk Kim, "To wards Real-Time Performance in a Scalable, Continuously Availab le Telecom DBMS", ClustRa, 2001

[7] 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, 2002

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

[9] 유병섭, 김명근, 김재홍, 배해영, "GMS/Cluster 설계 및 구현", 개방형 지리정보 시스템 학회, p. 225-231, 2003