제 18 회 한국정보처리학회 추계학술발표대회 논문집 제 9 권 제 2 호 (2002.11)

iBASE/Cluster-MiDAS 잠금 관리 모듈의 설계 및 구현

이훈순, 김영철, 김준
한국 전자통신연구원
e-mail : {hunsoon,kimyc,jkim}@etri.re.kr

Design and Implementation of Lock Management Module in iBASE/Cluster-MiDAS

Hun Soon Lee, Young Cheol Kim, June Kim
Electronics and Telecommunications Research Institute

요 약

iBASE/Cluster-MiDAS 는 기 개발된 바다-IV 를 공유 디스크 기반 클러스터 환경에서 동작하도록 확장한 iBASE/Cluster 의 하부 구조로써 다중 사용자용 자료 저장시스템이다. iBASE/Cluster-MiDAS 에서는 하나의 노드에서 동시에 수행되는 여러 트랜잭션들 사이에의 영향뿐 아니라 여러 노드에서 동시에 수행되는 여러 트랜잭션들 사이의 영향으로부터 데이터의 일관성을 유지해야 한다.

본 논문에서는 여러 노드에서 동시에 수행되는 여러 트랜잭션들 간의 동시성 제어를 위해 바다-IV/MiDAS 의 잠금 관리 모듈을 확장한 iBASE/Cluster-MiDAS 잠금 관리 모듈의 내부 구조와 구현 시 사용된 기법을 설명한다. iBASE/Cluster-MiDAS 의 잠금 관리 모듈은 잠금 캐싱을 사용하는 중앙 집중 잠금 관리 기법에 기반하고 있으며, 잠금 관리 모듈 수준에서 오류에 유연하게 대처한다.

1. 서론

컴퓨터가 등장한 이후로 컴퓨터의 성능을 높이기 위한 방법으로 마이크로 프로세서의 성능 향상이나 시스템 버스의 처리 속도 개선, 입출력 처리 능력 확장 등에 그 초점이 맞추어져 있었다. 하지만 단일 컴퓨터 시스템 만으로는 대규모 자료 처리 능력에 한계가 있으며 컴퓨터 시스템에 장애가 발생할 경우에 적절히 대처할 수가 없다.

인터넷에서 정보 제공자는 시·공간적 제약없이 끊임없는 서비스를 제공해 주어야 한다. 만약 서버의 고장으로 잠시 동안 서비스가 중단된다면 정보 제공자들은 사용자들로부터 신뢰를 잃게 되어 막대한 손실을 입게 될 것이다.

이러한 인터넷 환경에서 저비용으로 응용 서비스를 중단없이 제공하기 위한 방법으로 하나의 커다란 컴퓨팅 파워를 가진 값 비싼 단일 시스템을 이용하는 것보다는 작은 컴퓨팅 파워를 가진 여러 대의 컴퓨터들을 서로 연결하여 마치 하나의 컴퓨터처럼 사용하는 클러스터로 구성된 시스템이 많이 이용되고 있다.

따라서 모든 응용 서비스의 근간이 되는 데이터베이스 관리시스템 (DataBase Management System, 이하 DBMS) 도 이러한 요구 사항들을 만족 시켜 줄 수 있어야 한다. 즉, DBMS 가 기본 DBMS 의 기능뿐 아니라 클러스터 시스템이 가지는 고유 특징인 고 가용성과 고 확장성을 지원해야 한다.

고 가용성과 고 확장성을 위한 클러스터 환경에서 대용량 데이터의 효율적인 저장 및 검색을 지원해 주는 DBMS 를 클러스터 기반 DBMS 라고 한다.

상용 클러스터 기반 DBMS 로는 공유 디스크 (shared disk) 환경의 Oracle 9i-RAC[Oracle]와 비공유 (shared nothing) 환경의 Informix Extended Parallel Server 8.3[Informix], DB2 Universal Database Extended Enterprise Edition[DB2] 등 이 있다.

본 논문에서는 기 개발된 바다-IV 가 클러스터 DBMS 로써의 기능을 수행할 수 있도록 확장한 iBASE/Cluster 의 하부 저장 시스템인 iBASE/Cluster-MiDAS (이하 MiDAS/Cluster) 에서 데이터 일관성이 유지되도록 동시성 제어를 위한 방법으로 사용한 잠금 관리 기법에 대해 살펴본다.

본 논문의 구성은 다음과 같다. 2 장에서는 문제 정의를, 3 장에서는 관련 연구에 대해, 4 장에서 잠금 관리 모듈의 설계 및 구현에 대해 설명하고, 5 장에서 결론을 맺는다.

2. 문제 정의

여러 트랜잭션이 동시에 수행되는 경우에는 트랜잭션들 간의 영향으로 데이터의 일관성이 유지되지 않을 수도 있다. 따라서 동시에 수행되는 여러 트랜잭션들의 자료 접근에 대한 순서화를 하는 동시성 제어 (concurrency control) 기능이 필요하다.

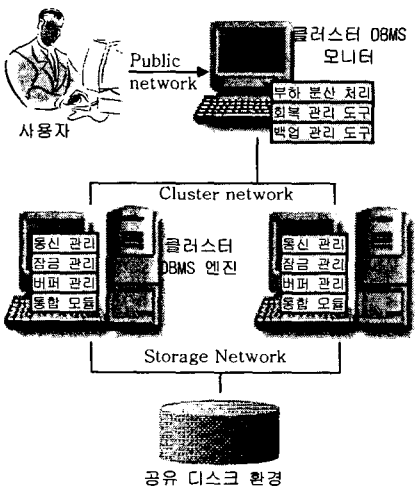
단일 시스템 환경이나 비공유 환경에서는 동일한 데이터에 접근하는 트랜잭션들은 하나의 시스템(이하 노드)에서만 존재하므로 단순히 하나의 시스템에서

동작하는 여러 트랜잭션 사이에의 영향만을 고려하면 되었다. 하지만 공유 디스크 환경에서는 클러스터를 구성하는 여러 노드에서 동작하는 여러 트랜잭션들이 공유 디스크 상의 공유 데이터에 접근하므로 한 노드에서 동작하는 여러 트랜잭션들 사이에의 영향뿐 아니라 클러스터를 구성하는 여러 노드 사이에의 영향을 고려해야 한다. 따라서 잠금을 이용한 동시성 제어를 한다면 하나의 노드에서 동작하는 트랜잭션들이 획득한 잠금에 대한 관리뿐 아니라 클러스터를 구성하는 노드들 전체에서 동작하는 트랜잭션들이 획득한 잠금에 대한 관리가 필요하다.

고 가용성을 가지는 클러스터 기반 DBMS 는 클러스터를 구성하는 하나의 노드 오류가 발생해도 다른 노드에서는 데이터의 일관성을 유지하면서 서비스를 계속 진행할 수 있어야 한다. 시스템의 고 가용성을 위해서는 데이터의 일관성 보존을 위한 모듈인 잠금 관리 모듈이 오류에 유연하게 대처할 수 있어야 한다.

3. 관련 연구

3.1 iBASE/Cluster



[그림 1] iBASE/Cluster 개념도

iBASE/Cluster 는 공유 디스크 기반 클러스터 환경에서 클러스터를 구성하는 여러 노드에 DBMS 엔진을 탑재하여 논리적으로 하나의 DBMS 처럼 동작하며, 고 가용성과 고성능 트랜잭션 처리를 지원하는 DBMS 이다.

iBASE/Cluster 는 기 개발한 바다-IV 를 테스트 베드로 사용하여 개발하고, 클러스터 시스템이 가지는 고유 특징들을 지원함과 동시에 바다-IV 에서 동작하는 응용 프로그램이 그대로 동작되도록 호환성을 지원한다. 바다-IV 중 트랜잭션 처리와 관련된 바다-IV/MiDAS 는 클러스터 특성에 관련된 기능을 위해 재설계하여 구현했으며, 다른 부분들은 그대로 활용했다.

3.2 개념

가. 중앙 집중 잠금 관리와 분산 잠금 관리

중앙 집중 잠금 관리 (central lock management) 란, 한 곳에서 모든 자원에 대한 잠금 관리를 하는 것이다[Rahm91]. 한 곳에서 모든 자원에 대한 잠금 관리를 한다는 것은 잠금 승인(할당)과 반납에 대한 권한이 한 곳에 있다는 것이다. 분산 잠금 관리 (distributed lock management) 란, 잠금에 대한 관리를 여러 노드에서 나누어서 하는 것이다[Rahm91].

나. 전역 잠금 관리와 지역 잠금 관리

클러스터 환경에서 특정 잠금 자원에 관련된 잠금 정보를 관리하는 것을 전역 잠금 관리 (global lock management) 라고 하고, 특정 노드와 관련된 잠금 정보를 관리하는 것을 지역 잠금 관리 (local lock management) 라 한다. 따라서 전역 잠금 관리 정보에는 특정 잠금 자원에 대해 모든 노드에서 획득/요청한 잠금에 대한 관리가 이루어지고, 지역 잠금 관리 정보에는 특정 노드의 트랜잭션이 획득/요청한 잠금에 대한 관리가 이루어진다.

다. 잠금 캐싱

잠금 캐싱 (lock caching) 이란, 서비스 노드에서 한 번 획득한 잠금에 대해 해당 잠금을 요청한 트랜잭션이 종료되었음에도 불구하고 해당 객체에 대한 잠금을 관리하는 곳으로 반환하지 않고 서비스 노드 수준에서 관리하고 있다가, 해당 노드의 다른 트랜잭션에서 잠금을 요청하면, 해당 객체에 대한 잠금을 관리하는 곳에 문의하지 않고 해당 노드 수준에서 잠금에 대한 할당을 할 수 있도록 하는 것을 말한다.

라. 논리 잠금과 물리 잠금

공유 디스크 클러스터 환경에서 논리적 일관성을 위해 획득하는 잠금을 논리 잠금 (logical lock) 이라 하는데 파일과 레코드 잠금이 이에 해당되고, 물리적 일관성을 위해 획득하는 잠금을 물리 잠금 (physical lock) 이라 하는데 페이지 잠금이 이에 해당한다 [Moh91]. MiDAS/Cluster 에서는 논리 잠금을 트랜잭션 잠금, 물리 잠금을 버퍼 잠금이라 칭한다.

3.3 Oracle 9i에서의 잠금 관리

Oracle9i-RAC [Oracle] 에서는 공유 데이터베이스와 데이터베이스에 있는 공유 자원에 대한 동시 접근시 데이터 일관성을 제공할 수 있도록 하기 위해서 전역 캐시 서비스 (Global Cache Service, GCS) 와 전역 인큐 서비스 (Global Enqueue Service, GES) 라는 것을 제공한다. GCS 와 GES 에서는 자원과 자원을 획득하고 있는 인큐에 대한 정보를 기록하기 위해 전역 자원 디렉토리 (Global Resource Directory, GRD) 를 유지한다. 이 GRD 는 클러스터를 구성하는 모든 노드의 메모리에 중복/분산 시켜서 관리한다. 즉, GCS 와 GES 는 분산된 구조를 가지며 고장 감내(fault tolerance)를 한다.

4. 잠금 관리 모듈의 설계 및 구현

본 장에서는 읽기 연산이 많은 응용 -- 예를 들면 바이오 인포메틱스 (Bio-Informatics) 나 데이터 웨어하

우스 (Data Warehouse) 응용 -- 에 적합하도록 설계 및 구현한 잠금 관리 모듈에 대해 살펴 본다.

4.1 잠금 관리기 구조

가. 잠금 관리 방법

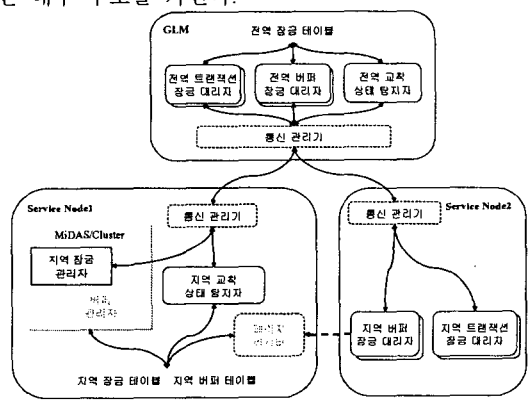
중앙 집중 잠금 관리 방법은 잠금 관리의 부하가 한 곳으로 집중되어 성능 저하를 가져 올 수 있다. 분산 잠금 관리 방법은 동적인 노드 추가(삭제) 시에 잠금 자원 관리 권한 분배의 문제가 있고, 잠금 획득(반환) 시에 잠금에 대한 관리 권한이 어느 곳에 있는지를 알아내는 절차가 필요하다. 특정 노드에서 잠금 자원 관리 권한 정보를 관리하고 있다면 이 또한 부하가 될 수 있다.

읽기 연산이 많은 응용에서는 잠금 충돌이 많이 발생하지 않으므로 잠금 캐싱 기법을 사용하는 것이 성능을 올리는 한 방법이 될 수 있다. 또한 캐싱된 잠금의 경우에는 잠금 자원 관리를 담당하는 곳으로 부하가 집중되는 것을 어느 정도 줄일 수 있다.

이상을 종합하여 MiDAS/Cluster 는 잠금 자원에 대한 관리 편의를 위해 잠금 자원에 대한 관리 권한은 한 곳에서 가지고 있으며, 읽기 연산이 많은 응용에 적합하도록 하기 위해 잠금 캐싱을 지원하는 구조, 즉 잠금 캐싱을 지원하는 중앙 집중 전역 잠금 관리 (centralized global lock management with lock caching) 구조를 가지도록 한다.

나. 내부 구조

MiDAS/Cluster 의 잠금 관리 모듈은 [그림 2]와 같은 내부 구조를 가진다.



[그림 2] MiDAS/Cluster 잠금 관리 모듈 내부 구조

클러스터 전체에서 노드 수준의 잠금 할당에 대한 관리를 전역 잠금 관리자 (Global Lock Manager, GLM)가 담당하고, 각 서비스 노드에서 캐싱된(노드에 할당된) 잠금에 대해 진행 중인 트랜잭션 수준의 잠금 할당에 대한 관리는 지역 잠금 관리자 (Local Lock Manager, LLM)가 담당한다. 전역 잠금 관리자 수준에서 관리하는 잠금 정보는 전역 잠금 테이블에, 지역 잠금 관리자 수준에서 관리하는 잠금 정보는 지역 잠금 테이블에 유지된다.

각 잠금 관리자들은 그 하는 일에 따라 트랜잭션 (파일,레코드) 잠금 처리를 지원하기 위한 트랜잭션 잠금 대리자 (transaction lock assistant agent) , 버퍼 잠금을 위한 버퍼 잠금 대리자 (buffer lock assistant agent) 와 교착 상태 탐지 및 해결을 위한 교착 상태 탐지자 (deadlock detector)로 나누어진다. 동시성 향상을 위해 잠금 테이블을 다수의 파티션으로 나누었는데 각 잠금 대리자들은 잠금 테이블 파티션 수만큼 존재하고, 각각은 지정된 파티션만을 담당한다.

전역 잠금 관리 노드로부터의 잠금 승인과 반환 관련 모든 메시지는 잠금 대리자가 받아서 처리한다. 만약 트랜잭션이 잠금 승인 메시지를 받는다면 프로세스 스케줄링 정책에 의해 문맥 교환 (context switching)이 일어나는 경우에 잠금에 대한 승인 처리도 아직 완료되지 않았는데 반환 요청에 대한 처리가 먼저 진행되는 등의 문제가 발생할 수 있다. 전역 잠금 관리자에서도 같은 이유로 각 파티션 별 잠금을 담당하는 대리자를 하나씩만 둔다.

통신 관리기는 전역 잠금 관리자와 여러 지역 잠금 관리자 간의 메시지 교환을 담당하는 역할을 하고, 이를 통해 여러 노드에 존재하는 잠금 관리자들을 하나의 커다란 잠금 관리 모듈로 보이게 한다.

4.2 잠금 연산

MiDAS/Cluster 에서는 서비스 노드에서 캐싱된 잠금을 가능한 오랫동안 사용할 수 있는 방향으로 잠금 연산이 처리된다.

서비스 노드에서 트랜잭션의 잠금 요청을 무조건 전역 잠금 관리 노드로 올리는 것이 아니라 캐싱되면 잠금이 허용 가능한 경우에만 전역 잠금 관리 노드에 잠금 요청을 올린다. 즉, 서비스 노드에 캐싱되어도 진행 중인 트랜잭션들이 획득한 잠금과 호환성이 없어서 요청한 트랜잭션이 잠금을 할당 받아 진행할 수 없는 경우에는 다른 캐싱된 노드에서 잠금을 좀 더 오랫동안 사용할 수 있도록 한다. 또한 동일 트랜잭션의 잠금 변환 요청이 지역 잠금 관리자 수준에서 승인이 불가한 경우에도 이미 다른 트랜잭션에 의해 잠금 요청이 올라갔으나 승인 메시지를 받지 못한 것이 있는 경우라면 그 요청 이후에 처리되도록 한다. 이렇게 함으로써 전역 잠금 관리 노드의 특정 객체 관련 잠금 테이블 엔트리에 특정 노드 관련 정보는 승인 그룹과 대기 그룹에 최대 각각 하나씩만 존재하게 되고, 잠금 요청 메시지를 줄이기 위해 전역 잠금 관리 노드에 이미 요청한 잠금 모드에 대한 추적을 하지 않아도 된다.

전역 잠금 관리 노드로부터 잠금 반환 요청 메시지를 서비스 노드가 받았을 때 캐싱된 잠금에 대한 반환은 즉시 하는 것이 아니라 현재 캐싱된 잠금 모드로 승인 가능한 대기 중인 트랜잭션들에 대한 처리를 마친 후에 잠금 반환이 일어나도록 했다.

4.3 교착 탐지

잠금 기법에 기반한 동시성 제어 방식은 교착 상태가 발생할 수 있으므로 교착 상태에 대한 탐지 기법을 제공한다. 트랜잭션 수준의 잠금 정보는 서비스 노드의 지역 잠금 테이블에 분산되어 존재한다. 교착 상

태 탐지를 위해 전역 잠금 관리 노드에 전역 교착 상태 탐지자 (Global Deadlock Detector, GDD) 프로세스와 서비스 노드에 지역 교착 상태 탐지자 (Local Deadlock Detector, LDD) 프로세스를 둔다. 전역 교착 상태 탐지는 GDD 가 주기적으로 LDD 에게 잠금 정보를 요청하여 잠금 정보를 받아서 탐지한다. LDD 는 GDD 에게 잠금 정보를 전송하기 전에 해당 지역 노드 수준의 교착 상태를 탐지해서 해결한 후에 전송한다. 교착 상태 탐지는 바다-IV/MiDAS 에서와 같이 Jiang 이 제안한 알고리즘[Jia88]을 이용한다.

교착 상태 탐지 결과 희생자로 선정된 트랜잭션이 요청한 전역 잠금 관리자로 전송된 잠금 요청에 대한 철회는 하지 않는다.

래치와 마찬가지로 버퍼 잠금은 교착 상태를 방지하는 방법 (deadlock prevention) 으로 요청하고 획득하므로 교착 상태 탐지의 대상에서 제외된다.

4.4 오류 대처

잠금 관리 모듈 수준에서 오류에 대한 대처는 데이터의 일관성을 해치는 상황을 방지하면서 고 가용성을 지원하는 것이다.

전역 잠금 관리 노드에서는 서비스 노드 오류시 회복을 수행해야 하는 자원과 관련된 잠금 서비스는 허용하지 않는다. 하지만 회복과 무관한 자원에 대한 잠금 서비스는 허용한다. 따라서 오류가 발생하지 않은 노드에서 오류와 무관한 자원에 대해 접근하는 트랜잭션은 중단없이 계속 진행된다.

전역 잠금 관리 노드 오류시에는 각 서비스 노드에서 유지하고 있던 잠금 정보를 받아서 전역 잠금 정보를 구축하는데 빠른 복구를 위해 각 파티션 별로 병렬로 진행한다. 이때 데이터의 일관성을 해치는 상황을 막기 위해 전역 잠금 정보에 대한 구축이 완료될 때 까지는 어떠한 전역 잠금 서비스도 허용하지 않는다.

서비스 노드 오류를 포함한 경우에는 로그에 기반한 회복이 진행되는데 회복의 진행 단계에 따라 잠금에 대한 점진적 허용을 한다. 재수행 연산이 종료되면 버퍼 잠금에 대한 허용을 하고, 복귀 연산이 종료되면 트랜잭션 잠금에 대한 허용을 한다.

오류와 잠금 서비스 허용과의 관계는 [표 1]과 같다.

4.5 구현

MiDAS/Cluster 잠금 관리 모듈은 Linux 환경에서 C (gcc 2.96) 를 사용하여 대략 30,000 라인의 소스 코드로 구현했다.

테스팅을 위해 RedHat 7.1 버전이 설치된 4 대의 컴퓩 머신을 기가비트 스위치로 클러스터 네트워크를 구축했다. 하나의 노드에는 전역 잠금 관리자와 지역 잠금 관리자를, 다른 세 노드에는 지역 잠금 관리자를 설치 후에 자체 개발한 테스트 프로그램으로 잠금 관리 모듈에 대한 간단한 테스팅을 마쳤고, 현재에는 CodeScroll[Code]이라는 테스팅 도구를 이용하여 집중적으로 테스팅을 하고 있다.

5. 결론

본 논문에서는 클러스터 기반 DBMS 인 iBASE/Cluster 의 하부 저장 구조인 MiDAS/Cluster 의 잠금 관리 모듈의 설계 및 구현에 대해 살펴 보았다.

잠금 자원에 대한 관리 권한은 한 곳에서 가지고 있으며 읽기 연산이 많은 응용에 적합한 잠금 캐싱을 지원하는 구조, 즉 잠금 캐싱을 지원하는 중앙 집중 전역 잠금 관리 구조를 MiDAS/Cluster 의 잠금 관리 모듈 구조로 제안했다. 또한 잠금 연산시 서비스 노드에서 캐싱된 잠금을 가능한 오랫동안 사용할 수 있게 함으로써 잠금 캐싱 효과를 최대한 누릴 수 있고, 잠금 획득/잠금 반환 절차가 단순해지도록 했다. 고 가용성 지원을 위해 오류시 잠금 모듈 수준의 대처 방법을 제안했다.

향후에는 GLM 오류시 좀더 빠른 회복을 지원할 수 있는 예비 GLM(backup GLM)을 두는 방법에 대한 연구가 이어져야 한다.

[표 1] 오류와 잠금 서비스

가. 오류 직후에 잠금 서비스

오류 / 오너	서비스 노드	GLM 노드	복합 오류
정상 노드	O	X	X
디스크	O	X	X
오류 노드	X	X	X

O : 허용 X : 불허

나. 오류 후 잠금 허용 시점

오류 / 오너	서비스 노드	GLM 노드	복합 오류 (SN + GLM)
정상 노드	이미 허용	전역 잠금 정보 구축 후	전역 잠금 정보 구축 후
디스크			BL : 재수행 후 TL : 복귀 후
오류 노드	BL:재수행 후 TL:복귀 후		

BL : 버퍼 잠금 TL : 트랜잭션 잠금

참고문헌

[Code] CodeScroll http://www.suresofttech.co.kr/

[DB2] DB2 Universal Database http://www-4.ibm.com/software/data/db2/udb/

[Informix] Informix Extended Parallel Server 8.3 http://www-3.ibm.com/software/data/informix/xps/

[Jia88] B. Jiang, "Deadlock Detection is Really Cheap," ACM SIGMOD RECORD, Vol.17,No.2,June 1988.

[Moh91] Mohan, C., Narang, I., "Recovery and Coherency-Control Protocols for Fast Intersystem Page Transfer and Fine-Granularity Locking in a Shared Disks Transaction Environment," Proc. 17th Int'l Conf. on VLDB, Barcelona, September 1991

[Oracle] Oracle 9*i* Database : Real Application Cluster http://www.oracle.com/jp/deploy/database/oracle9i/index.html?rc_homc.html

[Rahm91] Rahm, E. "Concurrency and Coherency Control in Database Sharing Systems", Technical Report ZRI 3/91, Dec. 1991, Univ. of Kaiserslautern, Dept. of Computer Science.