병렬적 로깅 기법을 통한 주 메모리 DBMS의 병목 현상 해소

이주창, 이상호, 차상균
{juch, sangho, chask}@kdb.snu.ac.kr
서울대학교 지식 및 정보 넷 연구실

Solving Bottleneck in MMDB using Parallel Logging method

Juchang Lee, Sangho Lee, Sang K. Cha
Knowledge and Information Net for Sharing Lab., Seoul National University

요 약

주 메모리 DBMS에서의 회복 시스템은 디스크에의 접근을 요하는 유일한 경우이기 때문에 시스템 전체 성능의 병목(bottleneck)이 된다. 본 논문에서는 회복 시스템에서 발생하는 주 메모리 DBMS의 병목 현상을 해소하기 위하여 복수 개의 로그 디스크를 이용하는 병렬적 로깅 기법을 제안한다. 또한 병렬적 로깅 기법을 사용하는 경우 로그 레코드들이 여러 디스크에 흩어짐으로써 시스템 재시작 시 이들간의 순서를 재규명 해야 하는 오버헤드가 생기는데 이를 해결하기 위해서 병렬적 재시작 알고리즘을 제안한다.

1. 서론

주 메모리 상주형 DBMS에서의 회복 시스템은 디스크에의 접근을 요하는 유일한 경우이기 때문에 시스템 전체 성능상 병목이 된다. 본 논문에서는 회복 시스템에서 발생하는 MMDB의 병목 현상을 해소하기 위하여 복수 개의 로그 디스크를 이용한 병렬적 로깅 기법을 제안한다. 기본적인 생각은, 한 시스템이 여러 개의 디스크를 보유하는 것은 아주 보편적인 경우이므로 이런 경우에 시스템의 자원을 최대한으로 활용하자는 것이다. 그러나 하나의 디스크에 모든 로그를 저장하는 경우에는 로그 파일 내에서의 로그 레코드 저장 순서가 로그 레코드의 생성 순서와 일치하지만, 여러 개의 디스크에 한 시스템에서 발생되는 로그를 분산시켜 저장하는 경우 로그 레코드들의 생성 순서에 관한 정보를 별도로 저장해야 할 뿐만 아니라 여러 개의 로그 디스크로부터 로그 레코드들의 생성 순서를 재구성해야 하는 오버헤드가 존재한다. 시스템 재시작 성능은 시스템의 availability와 깊은 관련이 있는 중요한 요소이므로 본 논문에서는 시스템 재시작 시간 단축을 위한 병렬적 재시작 알고리즘을 제시한다.

본 논문의 구성은 아래와 같다. 우선 2장에서는 관련 연구에 대해 기술하고 3장에서 본 논문이 전제하고 있는 회복 기법을 설명한다. 4장에서는 복수 개의 로그 디스크가 존재할 때에 가능한 로깅 구조를 열거하고 각각의 장단점을 분석한다. 그리고 5장에서는 병렬적 로깅 기법 사용 시 발생하는 분산된 로그 레코드 간의 우선 순위 문제를 기술하고 이를 해결할 수 있는 병렬적 재시작 알고리즘을 제시한다.

2. 관련 연구

[1]에서 이미 복수 개의 디스크를 사용하여 회복 시스템의 성능을 향상시킨다는 생각이 언급된 바 있으나 구체적인 설계나 재시작 알고리즘에 대한 연구는 없었다. 또한, [2]에서는 하나의 데이터베이스를 공유하는 여러 개의 서버 객체들이 동작하고 있는 분산 환경 하에서 각 서버마다 개별적인 로그 디스크를 할당하는 방법을 제안하고 있다. 그러나, [2]의 구조는 본 논문이 고려하고 있는 하나의 서버 객체에 여러 개의 로그 디스크를 부착하는 구조와는 근본적인 차이가 있어서, [2]에서 제시된 재시작 알고리즘을 본 논문의 구조에 직접적으로 적용할 수 없다.

3. Xmas의 회복 기법

Xmas(eXtensible MAin memory System)는 서울대학교 지식 및 정보 넷 연구실에서 개발된 주 메모리 상주형 저장 시스템으로서 회복 관리, 동시성 제어, 데이터 접근 경로 제공의 기본 기능 외에도 공간 데이터 관리 등의 기능을 지원하고 있다[3]. 본 논문의 회복 시스템에 관한 연구는 Xmas의 회복 시스템을

토대로 이루어 졌는데, Xmas 의 회복 시스템은 ARIES 회복 기법[4]을 바탕으로 하되 이를 주 메모리 환경에 적합하도록 개선하여 사용하고 있다.

가장 잘 알려져 있는 회복 기법 중 하나인 ARIES 는 기본적으로 로깅과 체크 포인팅을 통하여 데이터베이스 시스템의 일관성(consistency)를 유지한다. ARIES 는 물리적 로깅과 논리적 로깅을 모두 지원할 수 있도록 되어 있는데, Xmas 는 그 가운데서 물리적 로깅만을 지원한다. 체크 포인팅의 경우에는 ARIES 와 Xmas 모두 트랜잭션의 수행과 독립적으로 진행될 수 있도록 하는 퍼지 체크포인팅을 사용한다. 한편, 시스템 재시작 시 Xmas 는 ARIES 와 다르게 analysis 과정 없이 redo 와 undo 만의 수행으로 가능한데, 이는 주 메모리 DBMS 의 경우 메모리 상의 데이터베이스 내용이 언제 디스크에 반영되는지 확실히 알 수 있기 때문이다.

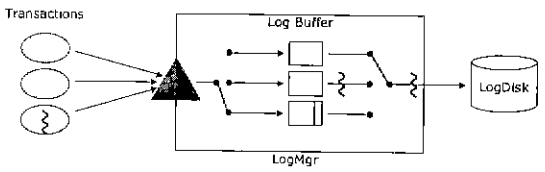

그림 1 기존 Xmas 회복 시스템의 구조

그림 1 은 하나의 로그 디스크를 사용하고 있을 때의 Xmas 회복 시스템의 구조이다. 하나의 로그 관리자 내에는 여러 개의 로그 버퍼들이 존재하고 있는데, 한 로그 버퍼가 디스크에 flush 되는 도중에도 다른 트랜잭션의 로그 레코드들이 계속해서 로그 버퍼에 쓰일 수 있도록 하기 위해서는 최소한 활동 중인 트랜잭션 수 이상의 로그 버퍼가 존재해야 한다. 한편, 로그 flush 는 트랜잭션 commit, 버퍼 full, 그리고 체크 포인팅 종료 직전에 발생한다.

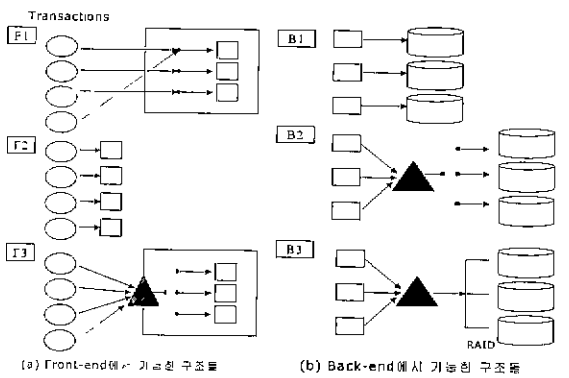

그림 2 구성 가능한 병렬적 로깅 모델들

4. 병렬적 로깅 모델들과 이들 간의 특성 비교

먼저 활동 중인 트랜잭션과 로그 버퍼의 가능한 관계(front-end), 로그 버퍼와 로그 디스크와의 가능한 관계(back-end)를 그림 2 와 같이 정리해 볼 수 있다. F1 은 로그 버퍼의 갯수가 초기화되어 있어 버퍼가 부족하여 트랜잭션이 봉쇄될 수 있는 가능성이 있지만, F2 의 경우는 dynamic 하게 버퍼 수가 결정되므로 그러한 단점은 없다. 물론, F2 의 경우 시스템 자원을 많이 요구하게 된다는 단점이 있다. 그리고 F3 와 같이 로그 버퍼와 트랜잭션이 바인드되어 있지 않은 경우도 생각할 수 있다. F1 이나 F2 의 경우 한 트랜잭션의 로그들은 동일한 로그 버퍼에 쌓이게 되지만, F3 에서는 여러 개의 로그 버퍼에 흩어질 수 있다.

B1 은 로그 버퍼와 로그 디스크가 바인드되어 있는 경우이고 B2 는 dynamic 하게 flush 할 로그 디스크를 결정하는 경우이다. B1 의 경우 한 디스크에만 flush 요청이 몰려 load balancing 제대로 되지 않을 우려가 있지만, B2 의 경우에는 한 로그 버퍼에 담겨 있는 내용이 여러 디스크에 흩어지는 경우가 발생한다.

한편, B3 와 같이 RAID 시스템을 로그 디스크로 사용하는 경우도 생각해 볼 수 있다. 이 경우 load balancing 이나 5 장에서 보게 될 분산된 로그 레코드들 간의 우선 순위 문제와 무관하기 때문에 하나의 로그 디스크가 존재하는 경우와 동일한 재시작 알고리즘을 적용할 수 있다. 그러나 RAID 시스템 자체의 부가적인 장치 비용 뿐만 아니라, 회복 시스템에서는 소량의 디스크 write 가 자주 발생하기 때문에 B1 이나 B2 에 비해 오히려 로깅 성능이 저하될 수 있다.

5. 병렬적 로깅 모델 하에서의 재시작 알고리즘

4 장에서 제시한 front-end 구조와 back-end 구조를 조합하면, 여러 가지 로깅 모델을 생각해 볼 수 있다. 일반적으로 하나의 로그 디스크만을 사용할 때에는 로그 레코드의 생성 순서에 따라 로그 레코드들이 저장되기 때문에 시스템 붕괴 후 재시작 시 순차적으로 로그 파일을 읽어 가며 redo 를 할 수 있다. 그러나, 병렬적 로깅 모델의 경우에는 여러 디스크에 로그 레코드들이 흩어지게 되므로 재시작 시 여러 디스크에 존재하는 로그 레코드들로부터 생성 순서를 재구성해야 한다. 특히 F1B1, F2B1 모델을 제외한 경우에는 한 트랜잭션의 레코드들도 여러 다른 디스크에 저장될 수 있다. 이러 분산된 로그 레코드들 간

의 우선 순위 규명을 위해서는 각 commit log 또는 모든 log 마다 global LSN(gLSN)를 두어야 할 뿐만 아니라 시스템 재시작 시에도 우선 순위를 규명하기 위한 analysis 과정과 정렬 과정이 필요하기 때문에 시스템 재시작 시간이 길어질 수 있다

이러한 오버헤드를 없애기 위해서는 로그 레코드의 생성 순서와 무관하게 시스템 붕괴 직전 상태로 데이터베이스를 회복할 수 있는 기법이 필요한데, gLSN을 이용하여 회복 연산 중에도 데이터베이스를 locking 하는 방법을 통해 가능하다. 물리적 로깅을 하는 시스템은 변경 전 이미지와 변경 후 이미지를 로그에 기록하기 때문에, 동일한 레코드에 대해 시간적으로 후에 생성된 로그를 redo 했다면 그 레코드에 대해 먼저 발생했었던 로그를 다시 redo 할 필요가 없어진다. 특히 이러한 회복 기법 사용 시 각 디스크 별로 별도의 쓰레드를 두어 병렬적으로 redo 를 할 수 있기 때문에 회복 속도를 크게 향상 시킬 수 있다

6. 성능 평가

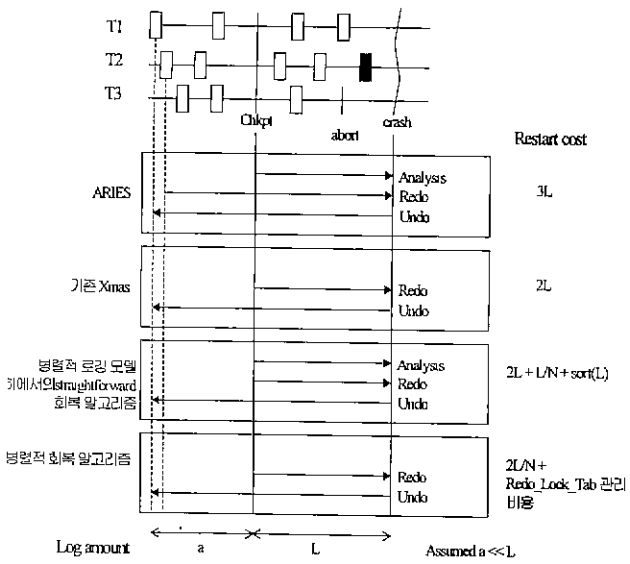

그림 3: 시스템 재시작 성능 비교

그림 3 은 ARIES 의 회복 알고리즘, 하나의 로그 디스크만 사용하던 기존 Xmas 의 회복 알고리즘, 그리고 본 논문에서 새롭게 제안된 병렬적 회복 알고리즘 간의 재시작 성능을 analytic 하게 비교한 것이다 N 은 로그 디스크의 개수이고 L 은 가장 최근에 성공한 체크 포인트 이후에 쌓인 로그 레코드의 양이다. 그림의 우측의 수식은 각 알고리즘 별로 재시작 비용을 계산한 것인데 기존 Xmas 의 경우는 순차적으로 디스크를 읽어가며 redo 와 undo 를 해야하므로 2L 의 cost 가 든다. 또한, 단순한 회복 알고리즘은 undo 시에만 병렬적으로 진행할 수 있고(2PL 동시성 제어 기법을 사용한다는 가정하에서 가능) 로그 레코드들의 gLSN 을 파악하기 위한 analysis pass 와 이들로부터 순서를 정렬하는 과정이 필요기 때문에 2L+L/N+sort(L)의 비용이 든다 그러나, 본 논문에서 제안한 병렬적 회복 알고리즘은 analysis pass 없이 redo 와 undo 를 병렬적으로 수행할 수 있기 때문에 2L/N+(Redo_Lock_Tab 관리비용) 만큼의 비용만이 요구된다. 뿐만 아니라, Redo_Lock_Tab 을 관리할 때에는 디스크에의 접근이 필요없고 각 object 마다 lock bit 를 두는 방법도 고려할 수 있으므로 Redo_Lock_Tab 관리로 인한 오버헤드는 상대적으로 작다

7. 결론

본 논문에서는 복수 개의 로그 디스크가 존재하는 환경 하에서 병렬적 로깅 기법을 제안하고 이에 뒤따를 수 있는 분산된 로그 레코드들 간의 우선 순위 문제를 병렬적 회복 알고리즘을 제안함으로써 해결하였다. 향후에는 본 논문에서 제안한 병렬적 로깅 모델 및 회복 알고리즘의 안정성을 실험적으로 검증할 계획이다.

8. 참고 문헌

[1] Hector Garcia-Molina, Kenneth Salem "Main Memory Database Systems. An Overview," in IEEE Transactions on Knowledge and Data Engineering, 1992

[2] C. Mohan, "Data Base Recovery in Shared Disks and Client-Server Architecture," in IEEE international conference on Distributed Computing Systems 1992.

[3] 차상균, 박장호, 이성직, 박병대, "확장 용이한 주메모리 실시간 저장 시스템의 구조," 한국 정보 과학회 논문지(B), 제 24 권 제 6 호, 1997 년 6 월.

[4] C. Mohan, Don Haderle, Bruce Lindsay, "ARIES: A Transaction Recovery Method Supporting Fine-Granularity Locking and Partial Rollbacks Using Write-Ahead Logging," in ACM Transaction on Database Systems, pages 94-162, 1992