하드와 소프트 실시간 트랜잭션을 위한 통합된 동시성제어 기법

Integrated Concurrency Control Protocol for Hard and Soft Real-Time Transactions

홍석희
경성대학교 컴퓨터정보학부

Seok Hee Hong(shong@ks.ac.kr)

요약

대부분의 실시간 동시성제어 기법은 주로 한 가지 종류의 실시간 트랜잭션들을 위한 데이터 충돌 해결에 적용되어왔다. 다양한 응용 프로그램 지원과 하드웨어 성능향상으로 실시간 데이터베이스 시스템에서 여러 종류의 실시간 트랜잭션들을 스케줄 할 필요성이 증가하고 있다. 본 논문에서는 하드와 소프트 실시간 트랜잭션들 사이의 데이터 충돌을 해결하는 통합된 동시성제어 기법을 제안한다. 기존에 연구된 PCP(Priority Ceiling Protocol)와 MVPR(Multiversion with Precedence Relationship) 기법을 기반으로 하드 실시간 트랜잭션이 종료시한 내에 작업을 종료하도록 보장하며 종료시한을 만족하는 소프트 실시간 트랜잭션의 비율을 향상시키도록 한다. 제안한 통합된 동시성제어 기법이 직렬화가능 스케줄을 만족시키고 교착상태를 발생시키지 않음을 증명하였다. 모의실험을 통하여 다른 동시성제어 기법과 성능평가 비교를 하였다.

■ 중심어 : | 실시간 데이터베이스 시스템 | 동시성제어 기법 | 우선순위 | 종료시한 |

Abstract

Most concurrency control protocols have been devised for resolving data conflicts among real-time transactions of a single type. Recent real-time database systems should support various types of real-time transactions due to needs of many different types of applications and steady improvement of hardware. In this paper, we propose integrated concurrency control protocol to resolve data conflicts among hard and soft real-time transactions. Our proposed protocol, based on PCP(Priority Ceiling Protocol) for a hard real-time transactions and MVPR(Multiversion with Precedence Relationship), guarantees that hard real-time transactions meet their deadline, and decreases the deadline miss ratio of soft real-time transactions. We also proved that the proposed protocol guarantees serializable schedules and no deadlocks. The performance of the proposed protocol has been compared with other real-time concurrency protocols.

■ keyword : | Real-Time Database Systems|Concurrency Contorl|Priority|Deadline |

* 본 연구는 2005학년도 경성대학교 학술지원연구비에 의하여 연구되었습니다.

접수번호 : #080124-001
접수일자 : 2008년 01월 24일
심사완료일 : 2008년 02월 18일
교신저자 : 홍석희, e-mail : shong@ks.ac.kr

I. 서 론

실시간 데이터베이스 시스템의 트랜잭션은 주어진 시간 내에 작업을 완료해야 하는 제약을 가진다. 이와 같은 시간 제약을 종료시한(deadline)이라고 한다[1-3]. 실시간 데이터베이스 시스템의 트랜잭션들은 종료시한을 지키지 못한 경우 발생하는 결과에 따라 세 가지로 분류될 수 있다. 하드 실시간 트랜잭션(hard real-time transaction)은 종료시한을 지키지 못한 경우 시스템이 파괴되는 수준의 최악의 결과를 낸다. 따라서 하드 실시간 트랜잭션은 반드시 종료시한 내에 작업을 완료해야 한다. 펌 실시간 트랜잭션(firm real-time transaction)은 종료시한 이후에도 작업을 계속하는 경우 실행 결과가 무의미해지게 된다. 따라서 펌 실시간 트랜잭션의 경우 종료시한을 넘는 경우 즉시 실행을 철회해야 한다. 마지막으로 소프트 실시간 트랜잭션(soft real-time transaction)은 종료시한 이후까지 작업을 종료하지 못하는 경우 실행 결과가 점차 가치가 없어지게 된다.

실시간 트랜잭션들 사이의 동시성제어에 대한 연구는 실시간 데이터베이스 시스템 개발에 핵심적인 기술로 많은 연구 성과를 보여주었다. 대부분의 실시간 동시성제어 기법은 한 가지 종류의 실시간 트랜잭션을 지원하기 위해서 연구되어 왔다. 그러나 실시간 데이터베이스 시스템이 운영되는 환경은 다양한 사용자 요구사항과 복잡한 응용 프로그램을 지원하기 때문에 한 가지 종류의 실시간 트랜잭션만을 지원할 수가 없다. 따라서 하드, 펌, 소프트 실시간 트랜잭션들이 동시에 실행될 수 있는 환경을 지원하는 실시간 데이터베이스 시스템의 개발은 필수적이다. 다양한 실시간 트랜잭션을 위한 통합된 실시간 동시성제어 기법은 종료시한에 대한 여러 가지 요구조건을 만족시키면서 시스템의 성능을 향상시켜야 한다. 통합된 실시간 동시성제어 기법은 하드 실시간 트랜잭션들이 종료시한 내에 작업을 완료하도록 보장해야 하며 가능한 많은 소프트와 펌 실시간 트랜잭션들이 종료시한 이내에 작업을 완료하도록 해야 한다. 본 논문은 하드 실시간 트랜잭션과 소프트 실시간 트랜잭션을 효율적으로 스케줄링하기 위한 통합된 동시성제어 기법을 제안하고 모의실험을 통해서 다른 기법과 비교하고자 한다.

본 논문의 구성은 다음과 같다. 2장에서 기존에 연구된 동시성제어 기법에 대해서 기술하고 3장에서 제안하는 동시성제어 기법을 위한 트랜잭션 모델을 제시한다. 4장에서 통합된 동시성제어 기법을 제안하고 5장에서 모의실험을 통한 성능평가를 수행하고 마지막으로 6장에서 결론을 맺는다.

II. 관련 연구

일반 데이터베이스 시스템에서 동시성제어 기법의 성능평가 척도는 단위 시간 내에 실행된 트랜잭션들의 평균 응답시간(response time)이지만 소프트 실시간 데이터베이스 시스템은 단위 시간 내에 실행된 트랜잭션들이 종료시한 내에 작업을 완료한 비율로 성능을 평가한다. 많은 연구성과를 낸 소프트 실시간 동시성제어 기법들은 트랜잭션의 종료시한을 우선순위(priority)로 변환하여 우선순위에 기반한 동시성제어 기법을 연구해왔다. 일반 데이터베이스 시스템을 위한 동시성제어 기법으로 많은 연구가 이루어진 2단계 잠금기법(2PL: 2 phase locking)이나 낙관적기법(optimistic) 등이 실시간 동시성제어 기법에 적용되어 많은 연구성과를 보여주었다[1][4-6]. 트랜잭션들 사이에 데이터 충돌이 발생하는 경우 높은 우선순위의 트랜잭션을 선호하도록 데이터 충돌을 해결한다. 또한, 다중버전 잠금기법을 적용한 실시간 동시성제어 기법들에 대한 연구가 있었다. 특히, [6]의 연구에서 제안한 MVPR(Multiversion with Precedence Relationship)은 다중버전 2PL 기법을 기반으로 실시간 트랜잭션들에 대한 사전정보 없이 높은 우선순위의 트랜잭션을 선호하여 데이터 충돌을 해결하며 낮은 우선순위의 트랜잭션이라도 작업성과에 따라서 선택적으로 철회 또는 계속 실행하도록 한다.

많은 연구성과가 있었던 소프트 실시간 동시성제어 기법에 비해서 하드 실시간 동시성제어 기법의 경우 연구성과가 많지 않았다. 하드 실시간 트랜잭션은 종료시한을 반드시 만족해야 하는 엄격한 요구조건 때문에 실

시간처리 시스템의 프로세스와 유사한 작업 환경을 적용 받는다. 하드 실시간 트랜잭션은 주기적인 작업을 정해진 주기(period)로 반복적으로 실행하며 사용할 데이터를 미리 파악할 수 있다고 가정한다. [7]의 연구는 하드 실시간 트랜잭션의 실행시간, 사용할 데이터, 주기 등의 정보를 미리 파악할 수 있다고 가정하여 데이터 충돌을 해결한다. 하드 실시간 프로세스의 스케줄링 기법을 많이 연구된 주기단조(rate monotonic) 기법과 우선순위 실링 프로토콜(PCP:Priority Ceiling Protocol)을 하드 실시간 동시성제어 기법에 적용하였다. PCP 기법은 데이터 충돌시 가장 높은 우선순위의 트랜잭션을 선호하여 데이터 충돌을 해결하며 우선순위 상속(priority inheritance) 기법을 적용하여 록을 소유한 낮은 우선순위의 트랜잭션을 일시적으로 높은 우선순위를 할당하여 좀 더 능동적으로 종료시한을 만족시킬 수 있게 한다. 기존에 연구된 실시간 동시성제어 기법을 적용하여 여러 종류의 실시간 트랜잭션을 위한 통합된 동시성제어 기법에 대한 연구가 있었다. [8]의 연구는 일반 트랜잭션, 하드와 소프트 실시간 트랜잭션들 사이의 데이터 충돌을 해결하는 통합된 동시성제어 기법을 제안하였다. 하드 실시간 트랜잭션을 위한 동시성제어 기법으로는 PCP, 소프트 실시간 트랜잭션을 위한 동시성제어 기법으로는 낙관적기법인 OCC-W50을 적용하였고 일반 트랜잭션을 위해서는 2PL을 활용하였다. 세 가지 동시성제어 기법을 통합하기 위해 다단계 방식의 동시성제어 기법을 고안하였다. 다른 종류의 트랜잭션들 사이의 데이터 충돌을 해결하기 위해서 상위 단계에 통합된 트랜잭션 스케줄러를 두었으며 같은 종류의 트랜잭션들 사이의 데이터 충돌을 위해서 하위 단계에 세 가지 종류의 트랜잭션 스케줄러를 두었다. [9]의 연구는 일반 트랜잭션을 제외하고 하드와 소프트 실시간 트랜잭션을 위한 유사한 방법으로 통합된 동시성제어 기법을 제안하였다.

펌 실시간 트랜잭션과 일반 트랜잭션 사이의 데이터 충돌을 다단계 스케줄러를 기반으로 해결하고자 한 연구도 있었다[10]. 이 연구에서는 펌 실시간 트랜잭션들 위한 동시성제어 기법으로 우수한 성능을 보인 낙관적 기법으로 OPT-50을 기반으로 하였고 일반 동시성제어 기법으로는 2PL-HP를 사용하였다. [8-10]에서 제안한 여러 종류의 실시간 트랜잭션과 일반 트랜잭션을 위한 통합된 동시성제어 기법들은 단순히 기존에 연구된 각각의 동시성제어 기법들을 다단계로 통합하였기 때문에 다른 종류의 트랜잭션들 사이의 데이터 충돌을 효율적으로 해결하기에는 미흡하다. 또한, [8]과 [9]의 연구에서 기반으로 하고 있는 낙관적인 기법은 실시간 데이터베이스 시스템의 트랜잭션들의 수가 증가 할수록 시스템의 성능이 크게 저하되는 문제점이 있다. 본 논문에서 제안하는 통합된 동시성제어 기법은 기존에 연구된 PCP 기법과 [3]의 연구에서 실시간 트랜잭션의 수가 증가하여도 시스템의 성능이 크게 저하되지 않는 MVPR 기법을 기반으로 한다. 또한, PCP 기법과 MVPR 기법을 유기적으로 통합시켜 하드와 소프트 실시간 트랜잭션들 사이의 데이터 충돌도 유연하고 효율적으로 해결하고자 한다.

III. 트랜잭션 모델

1. 트랜잭션 모델

트랜잭션은 데이터에 대한 일련의 읽기와 쓰기 연산들로 표현되며 데이터를 사용하기 전에 해당하는 록을 소유하도록 요청한다. 요청 한 록이 호환되지 않는 록 모드를 다른 트랜잭션이 소유하고 있는 경우 트랜잭션 관리자는 이 데이터 충돌을 해결해야 한다. 제안하는 통합된 동시성제어 기법은 일반적인 실시간 데이터베이스 환경에서 상대적으로 중요도가 낮은 펌 실시간 트랜잭션을 제외하고 하드와 소프트 실시간 트랜잭션 사이의 데이터 충돌을 해결한다. [그림 1]은 본 논문에서 가정하는 트랜잭션 모델을 나타낸다. 트랜잭션 관리자(transaction manager)는 실시간 트랜잭션들 사이의 데이터 충돌을 해결한다. 트랜잭션 관리자는 같은 종류의 트랜잭션들 간의 데이터 충돌인 경우 하부의 해당 스케줄러에서 해결하게 하고 하드와 소프트 실시간 트랜잭션들 사이의 데이터 충돌인 경우 직접 해결한다. 하드 실시간 트랜잭션은 주기적으로 실행하며 실행시간과 사용할 데이터에 대한 정보가 미리 트랜잭션 관리자에

게 알려진다.

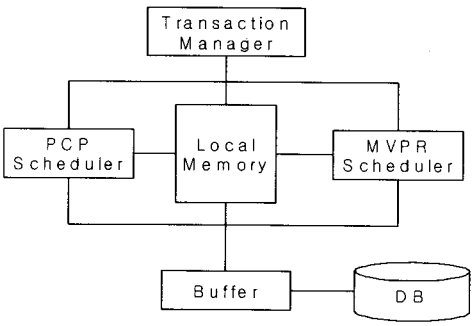

그림 1. 트랜잭션 모델

하드 실시간 트랜잭션이 실행을 시작할 때 필요한 모든 데이터를 데이터베이스에서 지역 기억장치(local memory)로 읽어오기 때문에 하드 실시간 트랜잭션은 디스크 I/O 없이 효율적으로 실행될 수 있다. 반면에 소프트 실시간 트랜잭션의 경우 실행시간, 종료시한, 사용할 데이터 등에 대한 정보를 트랜잭션 관리자에게 미리 제공할 수 없다. 따라서 소프트 실시간 트랜잭션은 필요로 하는 데이터를 실행 중간에 데이터베이스로부터 디스크 I/O를 통해 버퍼로 읽어야 한다. 또한, 본 논문에서 트랜잭션의 쓰기연산은 덮어쓰기(overwrite)의 의미를 가지기 때문에 쓰기연산에 읽기연산은 포함되지 않는다.

2. PCP 기법

하드 실시간 트랜잭션의 종료시한을 만족시키기 위한 PCP 기법은 록을 사용하여 데이터 일관성을 유지하며 주기단조 방식으로 우선순위를 할당한다. PCP 기법에서 하드 실시간 트랜잭션이 사용하는 록은 읽기록과 쓰기록 등이다. 록을 기다리는 다른 트랜잭션의 대기시간을 적정 수준으로 제한하기 위해서 록을 소유한 트랜잭션들 중 가장 높은 우선순위를 이용한다. 각 데이터는 우선순위 실링값을 가지며 이 실링값에 근거하여 데이터 충돌을 해결한다. 록을 요청한 트랜잭션의 우선순위가 다른 트랜잭션에 의해서 소유된 데이터 중 가장 높은 우선순위 실링값보다 크지 않다면 록을 요청한 트랜잭션은 대기 상태가 된다. PCP 기법은 실행될 하드 실시간 트랜잭션의 주기와 실행시간에 대한 정해진 조건이 유지된다면 모든 하드 실시간 트랜잭션이 종료시한을 만족할 수 있음을 보장한다.

3. MVPR 기법

소프트 실시간 트랜잭션을 위한 동시성제어 기법으로 제안된 MVPR은 다중버전 2PL을 실시간 데이터베이스 환경에 적용시킨 기법이다. 트랜잭션들 사이의 직렬화가능 순서(serialization order)를 트랜잭션들 사이의 선행관계(precedence relationship)로 표현하여 우선순위를 기반으로 데이터 충돌을 해결한다[6]. MVPR은 읽기(read), 쓰기(write), 검증(certify) 등의 세 가지 록 모드를 사용한다. 읽기록은 데이터베이스로부터 데이터를 읽기 위해서 사용하며 쓰기록은 지역 기억장치에 읽어드린 데이터를 변경시킬 때 사용한다. 트랜잭션이 종료하는 시점에 소유하고 있는 모든 쓰기록은 검증록으로 변환되어야 하고 그 후 해당 데이터를 데이터베이스에 저장한다.

IV. 통합된 동시성제어 기법

1. 데이터 충돌 해결 방법

하드와 소프트 실시간 트랜잭션이 동시에 실행되기 때문에 트랜잭션 종류에 따른 데이터 충돌은 다음과 같이 분류된다. HRT는 하드 실시간 트랜잭션을 SRT는 소프트 실시간 트랜잭션을 의미한다.

- HRT 간의 데이터 충돌 : PCP 기법으로 해결.
- SRT 간의 데이터 충돌 : MVPR 기법으로 해결.
- HRT와 SRT 사이의 데이터 충돌 : 통합된 록 호환성표(lock compatibility matrix)를 참조하여 해결.

같은 종류의 트랜잭션들 사이의 데이터 충돌은 PCP나 MVPR로 해결되지만 서로 다른 종류의 실시간 트랜잭션들 사이의 데이터 충돌은 PCP나 MVPR 기법을 직

접 적용할 수가 없다. HRT와 SRT 사이에 데이터 충돌이 발생하는 경우는 다음과 같이 두 가지 상황으로 설명된다.

1. HRT가 소유한 록과 호환되지 않는 록을 SRT가 요청하는 경우.
2. SRT가 소유한 록과 호환되지 않는 록을 HRT가 요청하는 경우.

HRT의 종료시한은 반드시 지켜야 하기 때문에 항상 SRT보다 유리하게 데이터 충돌을 해결한다. 따라서 데이터 충돌에 포함된 SRT는 실행 대기 또는 철회 등으로 HRT를 선호하는 방식으로 데이터 충돌을 해결한다. 1번 상황에서는 HRT가 이미 록을 소유한 후에 SRT가 록을 요청했으므로 SRT를 HRT가 록을 해제할 때까지 대기시킨다. 2번 상황의 경우 반대로 SRT가 소유한 록을 HRT가 요청하므로 록이 해제될 때까지 대기하거나 SRT를 철회시키는 방식으로 데이터 충돌을 해결할 수 있다. 그러나 이와 같은 두 가지 방식이 단순하게 적용될 수는 없다. 예를 들어, HRT를 일방적으로 대기시킨다면 HRT가 종료시한을 만족시키지 못 할 수 있고 SRT를 항상 철회시키는 경우 SRT가 종료시한에 가까웠고 HRT는 종료시한까지 여유가 있는 경우 SRT의 철회는 SRT가 사용한 자원을 낭비하기 때문에 시스템 성능을 저하시키게 된다.

2. 통합된 록 호환성표

2.1 HRT와 SRT의 선행관계

HRT와 SRT 사이의 데이터 충돌을 해결하기 위한 록 호환성표는 MVPR에서 사용한 록 호환성표의 개념을 기반으로 하기 때문에 HRT와 SRT 사이의 선행관계를 고려해야 한다. PCP 기법은 우선순위가 간접적으로 트랜잭션들 사이의 직렬화가능 순서를 암시하기 때문에 명시적으로 선행관계에 해당하는 정보를 사용하지 않는다. 그러나 MVPR 기법은 데이터 충돌이 처음으로 발생한 순간 데이터 충돌을 발생시킨 SRT들 사이에 선행관계인 Before_Set과 After_Set을 설정한다[6]. 따라서 HRT와 SRT 사이의 데이터 충돌을 해결하기 위해서는 이와 같은 선행관계가 사용되어야 한다. HRT는 읽기록(R)과 쓰기록(W)을 SRT는 읽기록, 쓰기록, 검증록(C) 등을 사용한다. HRT의 쓰기록은 SRT의 검증록과 기능 상 동일하기 때문에 MVPR에서 사용한 선행관계 설정 방법을 HRT와 SRT 사이에 그대로 적용할 수 있다. 다음은 HRT와 SRT 사이에 데이터 충돌을 발생시키는 상황에 대해서 선행관계를 설정하는 방법을 제시한다. RSRT[x]⇒WHRT[x]는 데이터 x에 대해서 SRT가 읽기록을 소유한 후에 HRT가 쓰기록을 요청하는 상황을 표시한다. SRT<HRT는 선행관계에서 SRT가 HRT를 앞서는 것을 나타낸다.

- RSRT[x]⇒WHRT[x] : HRT의 쓰기연산 전에 SRT가 읽기연산을 했으므로 SRT<HRT로 설정한다.
- WSRT[x]⇒RHRT[x] : SRT의 쓰기연산은 아직 데이터베이스에 반영되지 않았기 때문에 HRT의 읽기연산은 SRT의 쓰기연산 전의 데이터를 읽는다. 따라서 HRT<SRT로 설정한다.
- CSRT[x]⇒RHRT[x] : 검증록은 데이터베이스에 데이터를 저장하기 위한 검증연산을 위한 것이므로 SRT<HRT로 설정한다.
- CSRT[x]⇒WHRT[x] : 이미 SRT가 검증을 했으므로 SRT<HRT로 설정한다.
- 나머지 HRT가 소유한 록을 SRT가 요청하는 상황에서는 HRT가 유리하도록 HRT<SRT로 설정한다.

일단 HRT와 SRT 사이에 선행관계가 설정되었다면 이후에 발생하는 모든 데이터 충돌에 이 선행관계를 활용하여 데이터 충돌을 해결한다. 선행관계는 이행규칙(transitive rule)에 의해서 더 많은 선행관계로 확장될 수 있다. 예를 들어, $T_1<T_2$와 $T_2<T_3$가 성립하면 이행규칙에 의해서 $T_1<T_3$가 유도된다. PCP 기법을 적용하는 HRT들 사이에는 선행관계가 성립하지 않는다. 따라서, HRT들과 SRT 사이의 선행관계가 적절하게 활용되기 힘들 수도 있다. 예를 들어, SRT<HRT_1이 성립하고 HRT_2가 HRT_1에 의해서 대기상태가 되었다고 하

자. 만일 SRT와 HRT_2와 첫 번째 데이터 충돌이 발생하여 HRT_2<SRT의 선행관계가 설정했다고 하자. PCP 기법은 2PL을 사용하기 때문에 HRT_1<HRT_2의 직렬화 가능 순서를 유추할 수 있다. 따라서 이행규칙에 의해서 SRT<HRT_1<HRT_2<SRT의 순환적인(cyclic) 선행관계가 성립하기 때문에 데이터 일관성을 위반하게 된다. HRT들 사이의 선행관계를 명시적으로 유지하지는 않지만 HRT들의 대기 상태에 따라서 이와 같은 선행관계를 유추하여 SRT와의 선행관계를 구성한다.

2.2 HRT와 SRT 사이의 록 호환성표

SRT와 HRT가 데이터 충돌을 발생시키면 트랜잭션 관리자는 두 트랜잭션 사이의 선행관계에 따라서 해당하는 록 호환성표를 참조하여 데이터 충돌을 해결한다. 록 호환성표는 록을 요청한 트랜잭션이 SRT 또는 HRT인지 따라서 [표 1]과 [표 2]로 분류된다. [표 1]은 SRT가 록을 소유하고 있을 때 HRT가 록을 요청하는 경우에 필요한 록 호환성표이다. [표 2]는 HRT가 록을 소유하고 있을 때 SRT가 록을 요청하는 경우에 필요한 록 호환성표이다. 선행관계에 따라서 두 부분으로 구분된다.

표 1. SRT가 소유한 록을 HRT가 요청시의 록호환표

SRT \ HRT	SRT<HRT		HRT<SRT	
	R	W	R	W
R	Y	Y	Y	A_{SRT}
W	$A_{SRT}(B^*)$	Y	Y	Y
C	B^*	Y	Y	Y

표 2. HRT가 소유한 록을 SRT가 요청시의 록호환표

HRT \ SRT	SRT<HRT			HRT<SRT		
	R	W	C	R	W	C
R	Y	A_{SRT}	A_{SRT}	Y	Y	B
W	Y	Y	Y	B	Y	B

Y는 트랜잭션이 요청한 록이 허용됨을 의미하고 A_{SRT}는 록 소유자인 SRT를 철회시킴을 의미한다. B는 록을 요청한 SRT를 대기시키는 것이다. B^*는 록을 요청한 HRT가 정해진 조건이 만족되는 동안 록이 해제되기를 기다린 후에 SRT를 철회시키는 것이다. 괄호 안의 B^*는 SRT가 다음에 기술할 종료단계(commit phase)에 있는 경우 B^*로 해결함을 의미한다. 이와 같이 HRT를 대기시키면 우선순위 반전(priority inversion) 문제가 발생하여 반드시 만족시켜야 할 종료시한을 넘길 수도 있다. 그러나 제안하는 동시성제어 기법에서는 정해진 시간제약을 만족시키는 동안만 대기시키기 때문에 HRT가 종료시한 내에 작업을 완료할 수 있게 한다. B^*의 경우 다음 부등식 (1)이 참(true)인 동안 HRT를 대기시키도록 한다. 부등식 (1)에서 α는 시스템 부하에 따른 가중치로 0에서 1사이의 값이다. d_i는 HRT_i의 종료시한이며 t는 현재 시간, c_i는 HRT_i의 총 실행시간,

$$\alpha \times (d_i - t) \geq c_i - t_i \quad (1)$$

ti는 HRTi가 현재까지 대기시간을 제외한 실행시간이다. 부등식 (1)의 의미는 가중치 α에 따른 종료시한까지의 여유시간(slack time)이 남은 실행시간보다 커야 한다는 것이다. HRTi는 여유시간이 남은 실행시간보다 같거나 적게 남은 순간 HRTi의 종료시한을 만족시키지 못하기 때문에 록을 소유한 SRT를 철회시키고 요청한 록을 소유한다. 시스템 부하에 따라서 변경되는 α를 1에 가깝게 할 수 록 HRTi의 대기시간이 길어지고 α를 0에 가깝게 할 수 록 HRTi의 대기시간이 짧아진다. 결과적으로 시스템의 부하가 심할 수 록 α를 0에 가깝게 해야 HRTi가 종료시한을 넘기지 않게 할 수 있다.

2.3 종료 단계(commit phase)

실시간 트랜잭션이 모든 작업을 완료한 후 마지막으로 [그림 2]에서와 같은 종료단계(commit phase)를 실행해야 한다. SRT의 종료단계에서는 소유한 모든 쓰기 록을 검증록으로 변환하고 선행관계에서 앞에 오는 모든 트랜잭션들이 종료할 때까지 대기한다. 또한, HRT의 경우 $\alpha \times (d_i - t) \geq c_i - t_i$의 조건이 만족되는 동안 선행관계에서 앞에 오는 SRT들이 종료할 때까지 대기한다. HRT의 경우 SRT들이 종료할 때까지 장기간 대기할 수 없기 때문에 부등식(1)의 조건을 적용하

여 HRT의 여유시간 내에서 대기하도록 한다. 마지막으로 임계영역(critical section)에서 단 하나의 실시간 트랜잭션만의 종료단계를 계속한다. 변경된 데이터를 데이터베이스에 저장하고 소유한 모든 록들을 반환한 후 선행관계를 수정한다. 종료단계에서 소유한 록들을 한 번에 해제함으로써 엄정한 스케줄(strict schedule)을 보장하여 불필요한 연속적인 철회(cascading aborts)를 방지할 수 있다[11].

```
Commit Phase(T)

if T is SRT then
    convert all Write locks into Certify locks;
    wait until Before_Set*[T] = 0;
else        /* T is HRT */
    wait all SRTs in Before_Set*[T] to commit
          while  α×(d_i − t) ≥ c_i − t_i   ;
    abort all SRTs in Before_Set*[T];
endif
Critical Section Begin
    write all modified data on database;
    unlock all locks;
    for all transactions K ∈ After_Set*[T],
          Before_Set[K] = Before_Set[K] − {T};
Critical Section End
```

그림 2. 종료 단계

3. 통합된 동시성제어 기법의 특성

제안하는 통합된 동시성제어 기법은 HRT들 사이에 PCP 기법을 적용하였고 SRT들 사이에 MVPR 기법을 사용하였다. HRT와 SRT 사이의 데이터 충돌은 통합된 록 호환성표를 기반으로 HRT의 종류시한을 만족시키면서 여유시간을 활용하여 SRT들을 스케줄 하고자 하였다. 따라서 통합된 동시성제어 기법은 PCP와 MVPR의 이론적 특성을 함께 가진다. 본 절에서는 제안하는 통합된 동시성제어 기법이 직렬화가능 스케줄을 보장하고 교착상태(deadlock)를 발생시키지 않음을 증명한다.

- 정리 1 : 통합된 동시성제어 기법은 HRT와 SRT에 대해서 직렬화가능 스케줄을 생성한다.
- 증명 : 세 가지 경우로 나누어 증명한다. (1) HRT 사이에 데이터 충돌이 발생하는 경우 2PL을 사용하는 PCP 기법의 특성에 의해서 직렬화가능 스케줄을 생성한다. (2) SRT 사이에 데이터 충돌이 발생하는 경우 MVPR 기법의 특성에 의해서 직렬화가능 스케줄을 보장한다. MVPR은 SRT들 사이에 선행관계를 순환 관계가 성립하지 않도록 록 호환성표를 구성하였으며 종료단계에서 선행관계에서 앞서는 SRT들만을 기다린다. (3) HRT와 SRT 사이에 데이터 충돌이 발생하는 경우 록 호환성표에 의해서 순환적인 선행관계가 성립하지 않도록 데이터 충돌을 해결한다. HRT와 SRT 사이에 이미 설정된 선행관계가 순환적이 되지 않도록 [표 1]과 [표 2]를 구성하였기 때문에 (2)의 경우와 동일하게 순환적인 선행관계는 생성될 수 없다. 또한, 통합된 동시성제어 기법의 종료단계에서도 선행관계에서 앞서는 트랜잭션들을 기다리기 때문에 직렬화가능 스케줄을 보장한다.

- 정리 2 : 통합된 동시성제어 기법은 교착상태를 발생시키지 않는다.
- 증명 : PCP와 MVPR은 해당 논문에서 교착상태를 발생시키지 않음을 이론적으로 증명하였다. 통합된 동시성제어 기법 역시 이와 같은 PCP와 MVPR 기법을 기반으로 했기 때문에 교착상태를 발생시키지 않는다. 그러나 HRT와 SRT 사이에 교착상태가 발생하지 않는지는 증명하고자 한다. 교착상태가 발생하려면 트랜잭션들 사이에 순환적인 대기(circular waiting)가 존재해야 한다. 통합된 동시성제어 기법에서 HRT와 SRT 사이에 대기 상태가 존재하는 경우는 선행관계에서 앞서는 트랜잭션을 선행관계에서 뒤에 오는 트랜잭션이 기다리는 경우에만 가능하다. 정리 1에 의해서 HRT와 SRT 사이에는 순환적인 선행관계가 성립할 수 없기 때문에 통합된 동시성제어 기법에 의해서 HRT와 SRT를 스케줄 하는 경우 교착상태는 발생할 수 없다.

V. 성능 평가

본 장에서는 통합된 동시성제어 기법을 다른 동시성 제어 기법들과 모의실험을 통해서 성능 평가를 하고자 한다. 모의실험을 통한 성능 평가는 프로세스 기반의 모의실험 라이브러리인 C++SIM[12]을 사용하여 수행한다.

1. 모의실험 모델

모의실험에 사용할 실시간 데이터베이스 시스템은 하드와 소프트 실시간 트랜잭션이 동시에 스케줄 된다고 가정한다. 하드와 소프트 실시간 트랜잭션들을 위한 동시성제어 기법인 MCC(mixed concurrency control) 기법[9]과 하드 실시간 동시성제어 기법인 PCP (priority ceiling protocol)[7] 등과 제안하는 통합된 동시성제어 기법(integrated concurrency control: ICC)을 비교한다. PCP 기법은 HRT를 위한 동시성제어 기법이지만 SRT에 비해서 하드 실시간 트랜잭션의 스케줄링에 우수한 성능을 보이기 때문에 비교대상으로 선택하고자 한다. 모의실험에 사용된 PCP 기법은 SRT들 사이의 데이터 충돌을 해결하기에 적합하지 않기 때문에 2PL-HP[1] 기법을 기반으로 SRT를 위한 스케줄링이 가능하도록 PCP 기법을 변형하였다. 실시간 트랜잭션은 종료시한을 가지며 데이터 충돌을 해결하기 위해서 우선순위로 변환된다. HRT의 우선순위는 하드 실시간 시스템에서 일반적으로 우수한 성능을 보여주는 주기 단조 알고리즘을 사용하여 할당한다. SRT의 경우 소프트 실시간 시스템에서 일반적으로 사용되는 EDF (earliest deadline first) 방식을 사용하여 우선순위를 할당한다. HRT의 종료시한은 실행시간과 주기를 이용해서 결정하며 SRT의 종료시한은 트랜잭션 내의 연산의 수에 가중치인 여유비율(slack factor)을 곱하여 결정한다.

2. 모의실험 변수 설정

다음 [표 3]은 모의실험에 사용할 변수의 의미와 설정 값을 기술한다. HRT는 필요한 데이터를 데이터베이스에서 지역 기억장치로 읽어온 후 작업을 시작하기 때문에 HRT 종료단계 이전에는 디스크 I/O를 하지 않는다. 그러나 SRT는 데이터를 사용할 시점에 데이터베이스에서 디스크 I/O를 통해 지역 기억장치에 읽어 들인다. 일반 데이터베이스 시스템에서 성능평가 척도는 평균 응답시간이나 단위시간 동안 처리된 트랜잭션의 수로 평가된다. 그러나 실시간 데이터베이스 시스템의 경우 종료시한을 초과한 트랜잭션의 비율인 MissRatio가 성능평가 척도로 사용된다. MissRatio는 100×(종료시한을 넘긴 트랜잭션 수/총 트랜잭션 수)로 표현된다.

표 3. 모의실험 변수 설정

모의실험 변수	설정 값
DB 크기	2000
HRT의 트랜잭션 크기	2~7개 연산(균등분포)
SRT의 트랜잭션 크기	5~10개 연산(균등분포)
SRT의 도착비율	초당 100~500개(균등분포)
쓰기연산 비율	30~50%(균등분포)

3. 성능평가 결과

실제 환경의 실시간 데이터베이스 시스템에서 HRT가 종료시한을 초과하는 경우 심각한 시스템 실패가 발생 할 수 있다. 따라서, 실제 환경에서 HRT는 0% 이상의 MissRatio는 될 수가 없다. 그러나 [그림 3]과 [그림 4]에서 HRT의 MissRatio가 0% 이상이 될 수 있음을 볼 수 있다. 이와 같은 MissRatio는 실제 환경에서는 불가능한 수치이나 HRT의 스케줄링 성능을 다른 동시성 제어 기법과 비교 평가하기 위해서 0% 이상의 MissRatio가 가능하도록 모의실험의 트랜잭션 도착비율을 과도하게 증가시켰다. [그림 3]은 시스템에서 실행되는 HRT와 SRT의 비율이 50%:50%의 경우 HRT와 SRT의 MissRatio를 나타낸다. [그림 3]에서 나타난 바와 같이 HRT의 MissRatio가 트랜잭션 도착비율이 증가함에 따라 SRT의 MissRatio는 HRT 위주의 동시성 제어 기법인 PCP가 최소 30%에서 최대 100%의 좋지 않은 MissRatio를 보여주는데 반해 ICC는 15%에서 56% 사이의 비교적 양호한 MissRatio를 보여준다. 그러나 MCC의 경우 SRT를 위해서 낙관적인 기법인 OCC를 사용했기 때문에 트랜잭션 도착비율이 높아질 수 록 MissRatio가 100%에 근접해가는 것을 볼 수 있다.

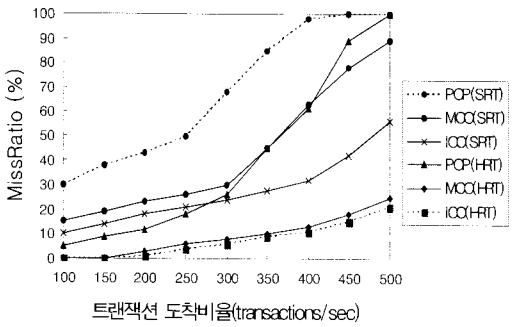

그림 3. MissRatio(HRT 비율=50%)

HRT의 MissRatio의 경우 PCP는 시스템 부하가 심해질수록 100%대로 빠르게 상승하는 것을 볼 수 있다. 이는 50%의 비율을 차지하는 SRT와의 데이터 충돌로 인해 PCP의 강점인 HRT의 데이터 충돌을 적절하게 해결하지 못했기 때문으로 보인다. [그림 4]의 경우 HRT대 SRT의 비율이 80%:20%로 HRT의 비율이 압도적으로 많은 경우의 MissRatio를 나타낸다. HRT의 비율이 높기 때문에 SRT를 위한 스케줄링 기회가 감소한다.

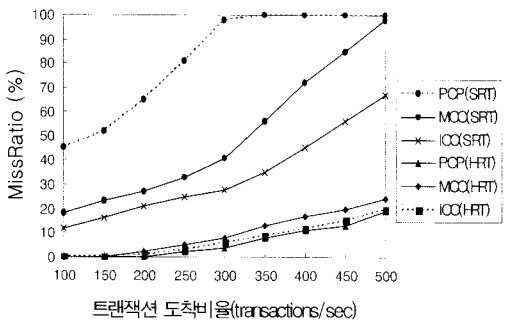

그림 4. MissRatio(HRT 비율=80%)

따라서 HRT의 MissRatio는 세 기법이 거의 비슷하게 나온 것을 볼 수 있다. 그러나 SRT의 경우 OCC 기법을 사용한 MCC의 경우 시스템 부하가 심해지면서 ICC에 비해서 MissRatio가 상승하는 것을 볼 수 있다. 성능평가 결과 HRT의 비율이 SRT와 비슷한 경우 PCP와 MCC의 성능은 좋지 않지만 ICC는 시스템 부하가 증가하여도 SRT의 MissRatio가 심하게 악화되지는 않는다. HRT의 MissRatio의 경우 PCP 기법을 기반으로 하기 때문에 ICC는 MCC에 비해서 근소한 차이로 낮은 MissRatio를 보여주었다. 그러나 HRT의 비율이 SRT에 비해서 많아질수록 세 기법의 HRT의 MissRatio는 비슷한 수준에서 낮게 유지됨을 볼 수 있었다.

VI. 결론

최근의 다양한 실시간 응용 프로그램에 대한 요구와 하드웨어 성능 향상으로 인해서 실시간 데이터베이스 시스템에서 여러 종류의 실시간 트랜잭션들을 스케줄링할 필요가 요구되고 있다. 본 논문에서는 하드와 소프트 실시간 트랜잭션들 사이의 데이터 충돌을 해결하기 위해서 기존의 기법들인 PCP와 MVPR을 기반으로 통합된 동시성제어 기법을 제안하였다. 제안한 통합된 동시성제어 기법이 직렬화가능 스케줄을 보장하고 교착상태를 발생시키지 않음을 증명하였고 다른 동시성제어 기법들과 성능평가를 통해서 제안한 기법과 비교하였다.

통합된 동시성제어 기법은 HRT가 종료시한을 만족할 수 있는 대기여유 시간을 이용하여 SRT를 스케줄링함으로써 많은 SRT가 종료시한을 만족할 수 있도록 데이터 충돌을 해결한다. 또한, SRT가 HRT와 데이터 충돌을 발생시키더라도 HRT의 종료시한까지의 여유시간을 최대한 활용하여 SRT가 HRT에 의해서 철회되어 시스템 자원을 낭비하지 않도록 한다. 향 후 연구 계획으로 펌 실시간 트랜잭션과 일반 트랜잭션과 같은 좀 더 다양한 종류의 실시간 트랜잭션들 사이의 데이터 충돌을 해결할 수 있는 동시성제어 기법을 연구하고자 한다.

참 고 문 헌

[1] R. Abbott, "Scheduling real-time transactions: A performance evaluation," ACM Trans, on

Database Systems, Vol.17, No.3, pp.513-560, 1992.

[2] M. Singhal, "Issues and approaches to design real-time database systems," ACM, SIGMOD Record, Vol.17, No.1, pp.19-33, 1988.

[3] S. H. Son, "Real-time database systems : A new challenge," Data Engineering, Vol.13, No.4, pp.51-57, 1990.

[4] J. R. Haritsa, M. J. Carey, and M. Linvy, "On being optimistic about real-time constraints," Prooceedings of the 1990, ACM PODS Symposium, pp.331-343, 1990.

[5] J. Lee and S. H. Son, "Using dynamic adjustment of serialization order for real-time database systems," Prooceedings of the 11th Real-Time Systems Symposium, pp.66-75, 1993.

[6] S. H. Hong and M. H. Kim, "Resolving data conflicts with multiple versions and precedence relationships in real-time databases," Information Processing Letters, Vol.61, No.3, pp.149-156, 1997.

[7] L. Sha, R. Rajkumar, S. H. Son, and C. H. Chang, "A Real-time locking protocol," IEEE Transactions on Computers, Vol.40, No.7, pp.793-800, 1991.

[8] K. Y. Lam, T. W. Kuo, and T. S. H. Lee, "Strategies for Resolving Inter-Class Data Conflicts in Mixed Real-time Database Systems," Journal of Systems and Software, Vol.61, pp.1-14, 2002.

[9] K. Y. Lam, T. W. Kuo, W. H. Tsang, and G. C. K. Law, "The Reduced Ceiling Protocol for Concurrency Control in Real-time Databases with Mixed Transactions," The Computer Journal, Vol.43, No.1, pp.66-80, 2000.

[10] S. Thomas, S. Seshadri, and J. R. Haritsa, "Integrating Standard Transactions in Firm Real-Time Database Systems," Information Systems, Vol.21, No.1, pp.3-28, 1996.

[11] A. Bernstein, A. Philip, V. Hadzilacos, and N. Goodman, "*Concurrency Control And Recovery In Database Systems*," Addison-Wesley, 1987.

[12] http://cxxsim.ncl.ac.uk/

저 자 소 개

홍 석 희(Seok Hee Hong) 정회원

- 1989년 2월 : 홍익대학교 전자계산학과(공학사)
- 1991년 2월 : 한국과학기술원 전산학과(공학석사)
- 1997년 2월 : 한국과학기술원 전산학과(공학박사)
- 1997년 3월 ~ 8월 : 한국전자통신연구원(ETRI) 박사후 연수과정
- 1997년 9월 ~ 현재 : 경성대학교 컴퓨터정보학부 부교수

<관심분야> : 실시간 데이터베이스, 소프트웨어 테스팅, 트랜잭션 처리 시스템