펌 실시간 트랜잭션을 위한 효율적인 병행수행제어 기법

Efficient Concurrency Control Method for Firm Real-time Transactions

신재룡
광주보건대학 의약정보관리과

Jae-Ryong Shin(sjr@ghc.ac.kr)

요약

실시간 데이터베이스 시스템에서는 높은 우선순위를 갖는 트랜잭션의 선행 처리가 항상 보장되어야 한다. 비관적 병행수행제어 방법은 충돌 발생 시 낮은 우선순위 트랜잭션을 철회 또는 대기시킴으로써 충돌을 해결한다. 그런데 높은 우선순위 트랜잭션이 마감시간을 지키지 못하고 시스템에서 제거되는 경우, 낮은 우선순위 트랜잭션이 불필요하게 철회되거나 대기하는 문제가 발생된다. 본 논문에서 제안하는 방법은 마감시간을 초과할 트랜잭션을 미리 시스템에서 제거한다. 그래서 불필요한 자원 낭비를 막고 낮은 우선순위를 갖는 트랜잭션이 불필요하게 철회되거나 대기하는 문제를 해결한다. 성능평가를 통해 트랜잭션의 마감시간 초과 비율 측면에서 기존의 방법들 보다 우수함을 보인다.

■ 중심어 : | 실시간 데이터베이스 | 병행수행제어 | 우선순위 |

Abstract

It always must guarantee preceding process of the transaction with the higher priority in real-time database systems. The pessimistic concurrency control method resolves a conflict through aborting or blocking of a low priority transaction. However, if a high priority transaction is eliminated in a system because of its deadline missing, an unnecessary aborting or blocking of a low priority transaction is occurred. In this paper, the proposed method eliminates a transaction that is about to miss its deadline. And it prevents needless wastes of resources and eliminates unnecessary aborting or blocking of a low priority transaction. It is shown through the performance evaluation that the proposed method outperforms the existing methods in terms of the deadline missing ratio of transactions.

■ keyword : | Real-time Database | Concurrency Control | Priority |

I. 서 론

최근 들어 디스크 기반 실시간 데이터베이스 응용들 뿐만 아니라 주기억장치 기반의 다양한 실시간 데이터베이스 응용들이 보편화 되고 있다. 실시간 데이터베이스 시스템 내에서 수행되는 여러 실시간 트랜잭션들은 작업에 필요한 모든 자원을 획득하고 해당 데이터를 처리하는데 걸리는 시간의 합이 미리 정해진 마감시간을 초과하지 않아야 한다.

시스템 내에 존재하는 다양한 실시간 트랜잭션들은 마감시간 내에 처리되지 못하는 경우에 발생하는 영향에 따라 큰 재앙이나 막대한 손실을 가져오는 하드 실시간(hard real-time) 트랜잭션과 결과의 가치가 상실되어 버리는 펌 실시간(firm real-time) 트랜잭션, 그리

접수번호 : #100517-004
접수일자 : 2010년 05월 17일
심사완료일 : 2010년 6월 29일
교신저자 : 신재룡, e-mail : sjr@ghc.ac.kr

고 마감시간 초과 시부터 결과의 가치가 점차적으로 떨어지는 소프트 실시간(soft real-time) 트랜잭션으로 분류된다. 하드 실시간 트랜잭션은 반드시 마감시간 이내에 실행되어야 하고, 펌 실시간 트랜잭션은 가능한 많은 수의 트랜잭션이 마감시간 내에 처리될 수 있어야 한다[1-4].

시스템 내에서는 여러 트랜잭션들이 한정된 자원을 획득하기 위해 경쟁하면서 대기 시간이 증가되거나 재시작 되는 상황이 발생된다. 이러한 시간 지연 문제는 트랜잭션의 전체적인 수행 시간을 길어지게 만들고 결국 트랜잭션이 마감시간 내에 완료되지 못하게 하는 방해요인이 된다.

가장 일반적인 병행수행제어 기법인 2단계 잠금(two phase locking) 기법은 데이터 객체를 접근할 때마다 잠금을 설정하고 트랜잭션 종료 시에 모든 잠금을 한꺼번에 해제하는 방법으로 자원 경쟁을 해결한다. 디스크 기반 시스템에서 잠금 연산이 디스크 입출력 시간에 비해 매우 빠르게 수행되기 때문에 가장 일반적인 방법으로 알려져 있다. 그러나 시스템 부하가 증가하는 경우 잠금 설정 및 충돌 검사가 빈번하게 발생하게 되면서 연산 비용 및 복잡한 잠금 테이블을 유지하는 비용이 시스템에 부담으로 작용될 수 있다[1-4].

주기억장치 기반 실시간 데이터베이스 시스템은 모든 데이터를 주기억장치에 상주시키고 로그, 백업, 대용량 데이터 등 일부 데이터에 한하여 디스크를 사용한다. 디스크 기반 시스템에 비해 트랜잭션의 수행이 빠르게 이루어지면서 잠금 유지 시간도 짧아지게 되므로 충돌 발생 확률이 현저하게 낮아지게 된다. 그래서 이런 상황에서 2단계 잠금 기법을 사용하게 되면 잠금 연산 비용이 실제 데이터를 처리하는 비용보다 오히려 커지는 현상이 발생되기도 한다[5][6].

주기억장치 기반 실시간 데이터베이스 시스템에서 잠금 연산 비용을 감소시키면서 실시간 트랜잭션의 병행수행제어를 원활하게 수행할 수 있도록 제안된 것이 정적 잠금 기법이다. 데이터 객체를 접근할 때마다 잠금을 설정하는 2단계 잠금 기법과 달리 이 기법에서는 트랜잭션의 시작 시점에서 필요한 모든 잠금을 한꺼번에 요청하는 것이 특징이다. 트랜잭션은 접근하고자 하는 모든 데이터 객체에 대한 잠금을 모두 획득한 경우에만 수행이 시작된다. 그래서 일단 수행을 시작한 트랜잭션은 잠금 충돌로 대기하지 않고 수행을 완료할 수 있다는 장점을 갖는다.

수행중인 트랜잭션이 필요한 모든 데이터 객체에 대한 잠금을 획득하고 있으므로 다른 트랜잭션들이 동일한 데이터 객체를 획득하기 위해 대기하는 상황이 발생하더라도 교착상태(deadlock)가 발생될 가능성이 전혀 없다. 또한 트랜잭션의 시작 시점에서 한꺼번에 잠금 연산을 수행하므로 잠금 관리가 매우 단순해진다.

많은 실시간 멀티미디어 데이터베이스 응용들 중에는 마감시간 정보만을 우선순위에 반영하여 보다 많은 수의 트랜잭션이 마감시간 내에 처리되도록 하는 경우도 있으나, 트랜잭션 가치의 경중에 따라 최대 개수를 수행시키기 보다는 최고 가치를 만들고자 하는 응용들도 있다. 이와 같은 응용에서는 펌 실시간 트랜잭션들의 마감시간 초과비율을 최소화 시키는 문제와 더불어 중요도를 고려한 작업 순서 제어를 통해 전체적인 작업 수행 결과의 가치를 최대화 시키는 것이 필요하다. 이를 위해 중요도가 높은 트랜잭션을 분류하고 다른 일반 또는 낮은 중요도를 갖는 트랜잭션들보다 먼저 처리되도록 보장하여야 한다. 이렇게 마감시간 보다 중요도 레벨을 우선적으로 고려한 충돌 해결 방법을 사용하게 되면 결과에 대한 가치를 최대화하는 측면에서는 큰 이득을 볼 수 있으나 마감시간 초과 비율이 전체적으로 높아지는 결과를 감수해야 한다.

펌 실시간 트랜잭션의 마감시간 초과는 실행 결과의 무의미함과 더불어 자원의 불필요한 사용으로 다른 트랜잭션들에게 방해가 됨을 나타낸다. 따라서 마감시간을 지키지 못하는 펌 실시간 트랜잭션은 잠금 획득 과정에서 미리 발견되고 시스템에서 제거되어야 마땅하다. 이렇게 높은 우선순위를 갖는 펌 실시간 트랜잭션의 불필요한 실행을 막게 되면 낮은 우선순위를 갖는 트랜잭션들이 불필요하게 철회되거나 잠금 획득을 위해 오랜 시간 대기해야 하는 문제를 해결하고 보다 많은 수의 트랜잭션들이 마감시간 내에 처리를 완료할 수 있는 환경을 제공하게 된다.

주기억장치 기반 실시간 데이터베이스 시스템에서는

매우 짧은 시간에 트랜잭션의 수행 완료가 가능하다. 그러나 대용량 멀티미디어 데이터를 다루는 실시간 응용이 보편화 되면서 트랜잭션의 수행 시간도 길어지고, 제한된 시스템 자원 하에서 모든 실시간 트랜잭션들이 마감시간 내에 완료되지 못하는 상황이 발생된다. 이에 본 논문에서는 한정된 시스템 자원 하에서 보다 많은 수의 실시간 트랜잭션이 마감시간 내에 처리될 수 있도록 하는 새로운 병행수행제어 기법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 실시간 데이터베이스를 위한 다양한 병행수행제어 기법들을 살펴보고, 3장에서는 새로운 병행수행제어 기법을 제안한다. 4장에서 성능평가 결과를 살펴보고, 마지막 5장에서 결론 및 향후 연구방향을 제시한다.

II. 관련 연구

데이터 객체를 접근할 때마다 잠금을 설정하고 트랜잭션 종료 시에 모든 잠금을 한꺼번에 해제하는 방법으로 자원 경쟁을 해결하는 2단계 잠금 기법은 가장 일반적인 병행수행제어 기법이지만 실시간 트랜잭션의 우선순위를 전혀 고려하지 않는다. 그래서 낮은 우선순위를 갖는 트랜잭션에 의해 높은 우선순위를 갖는 트랜잭션이 대기하게 되는 우선순위 역전(priority inversion) 문제를 발생시킨다. 실시간 데이터베이스 시스템에 적합하도록 실시간 특성을 고려하여 2단계 잠금 기법을 개선한 방법이 2PL-HP(two phase locking with high priority)이다. 2PL-HP는 충돌 해결을 위해 항상 낮은 우선순위를 갖는 트랜잭션을 철회 또는 대기시키고, 높은 우선순위를 갖는 트랜잭션이 먼저 수행되도록 한다. 예를 들어, 낮은 우선순위 트랜잭션이 미리 획득한 자원을 높은 우선순위 트랜잭션이 나중에 요구하는 경우, 낮은 우선순위 트랜잭션의 수행을 철회시키고 높은 우선순위 트랜잭션이 선점하도록 한다. 반대로 높은 우선순위 트랜잭션이 갖는 자원을 낮은 우선순위 트랜잭션이 요구하는 경우, 낮은 우선순위 트랜잭션을 대기시키고 높은 우선순위 트랜잭션의 계속 수행을 보장한다. 이렇게 높은 우선순위를 갖는 트랜잭션을 항상 우선적으로 처리하기 때문에 우선순위 역전 문제가 해결되고 실시간 데이터베이스 응용에 적합하다.

그러나 펌 실시간 트랜잭션을 다루는 경우 2PL-HP 기법은 시스템의 효율을 저하시키는 문제점을 갖는다. 그것은 수행중인 낮은 우선순위 트랜잭션을 철회시키고 해당 자원을 획득한 높은 우선순위 트랜잭션이 마감시간 내에 완료되지 못하는 경우에 발생된다. 불필요하게 재 시작된 낮은 우선순위를 갖는 트랜잭션들까지 마감시간 내에 작업을 완료하지 못하거나 추가적인 시스템 자원 경쟁을 유발하면서 전체적인 효율을 떨어뜨리는 상황이 발생하는 것이다.

낮은 우선순위 트랜잭션이 불필요하게 철회되는 것을 방지하기 위해 제안된 것이 조건부 재시작 기법이다. 이 기법은 충돌 발생 시점에서 현재 실행중인 낮은 우선순위를 갖는 트랜잭션의 남아있는 수행시간을 계산하고 충돌을 야기한 높은 우선순위 트랜잭션의 여유시간을 판단하여 여유시간 내에 남은 수행을 완료할 수 있을 경우에 현재 수행중인 낮은 우선순위 트랜잭션의 계속적인 수행을 보장하는 것이다. 그래서 낮은 우선순위 트랜잭션의 남은 수행 시간과 대기 중인 높은 우선순위 트랜잭션의 여유 시간을 어느 정도 예측할 수 있어야 가능한 방법이다. 그리고 철회되지 않고 계속 수행되던 낮은 우선순위 트랜잭션이 또 다른 트랜잭션과 충돌하여 철회되는 경우에는 미리 대기하고 있던 트랜잭션이 해당 자원을 획득하지 못하고 결국 마감시간을 어기는 상황이 발생되기도 한다.

트랜잭션 철회 비용이 아주 큰 경우에는 앞서 수행중인 트랜잭션의 계속적인 수행을 보장해 주는 우선순위 상속 기법을 사용한다. 이 기법에서는 앞서 수행중인 낮은 우선순위 트랜잭션과 동일한 자원을 요청하는 높은 우선순위 트랜잭션 간에 충돌이 발생하였을 때 높은 우선순위를 갖는 자원 요청 트랜잭션을 대기시키고 이 트랜잭션의 우선순위를 수행중인 낮은 우선순위 트랜잭션에게 상속해준다. 그리고 낮은 우선순위 트랜잭션이 정상적으로 수행을 완료하면 대기하던 높은 우선순위 트랜잭션이 해당 자원을 획득하여 처리하게 된다.

지금까지 살펴본 실시간 병행수행제어 기법들은 모두 충돌 검사와 재시작이 충돌 발생 시점에 수행되는

방법들이다. 이와 달리 충돌 검사와 재시작이 트랜잭션 완료 직전에 수행되는 방법이 낙관적 병행수행제어 기법이다. 낙관적 병행수행제어 기법은 트랜잭션의 처리 과정을 읽기, 검사 그리고 쓰기 단계로 구성한다. 읽기 단계에서는 트랜잭션의 실행에 필요한 데이터를 로컬 영역으로 복사한 후 처리하고, 검사 단계에서는 완료 또는 재시작 여부를 결정하는 트랜잭션 충돌 검사를 실시한다. 그리고 쓰기 단계에서는 완료가 보장된 트랜잭션의 데이터를 데이터베이스에 반영한다. 충돌 검사는 현재 수행중인 트랜잭션들을 대상으로 충돌 여부를 판별하는 순방향 검사와 이미 완료된 트랜잭션들을 대상으로 충돌 여부를 판별하는 역방향 검사가 있다. 순방향 검사를 실시하는 경우에는 여러 가지 충돌 해결 방법에 따라 현재 수행중인 트랜잭션들을 모두 재시작 시키거나, 검사 단계 트랜잭션 하나만을 재시작 시킨다. 역방향 검사의 경우에는 충돌이 발생되면 이미 완료된 트랜잭션들의 데이터 일관성 보장을 위해 항상 검사 단계에 있는 트랜잭션이 철회되어 재시작 된다.

낙관적 병행수행제어 기법은 검사 단계에서 충돌 검사 및 해결이 이루어지므로 블로킹이 발생하지 않는다. 그러나 충돌을 재시작으로만 해결하기 때문에 블로킹과 재시작을 적절히 사용하는 2PL-HP 기법에 비해 불필요한 재시작이 많을 수밖에 없다. 따라서 낙관적 병행수행제어 기법은 자원 경쟁이 심하지 않은 경우에 유리하고, 2PL 기반의 비관적 병행수행제어 기법은 자원 경쟁이 심한 경우에 적합하다[1-5].

III. 제안하는 병행수행제어 기법

정적 잠금 기법에서는 트랜잭션이 잠금을 요청한 여러 테이블 중에서 어느 한 테이블이 잠금 충돌인 경우 모든 테이블의 잠금 요청 리스트에서 대기한다. 그러나 실시간 데이터베이스 시스템에서는 높은 우선순위 트랜잭션이 낮은 우선순위 트랜잭션을 철회시키고 잠금을 획득하도록 해야 한다. 다만 무분별한 잠금 획득으로 인해 낮은 우선순위를 갖는 트랜잭션이 곧바로 수행되지 못하고 불필요하게 대기하다가 결국 마감시간을 지키지 못하는 문제를 방지하여 보다 많은 수의 트랜잭션들이 마감시간을 지킬 수 있도록 해야 한다.

제안하는 병행수행제어 기법은 주기억장치 기반 실시간 데이터베이스 시스템을 위해 정적 잠금 기법을 기반으로 한다. 우선순위 정보에 따라 충돌 발생 시 높은 우선순위 트랜잭션의 선점을 보장한다. 그러나 자원을 선점하는 높은 우선순위 트랜잭션의 불필요한 수행을 사전에 제거하여 낮은 우선순위 트랜잭션들의 불필요한 대기로 수행시간이 증가되고 마감시간 초과 비율이 증가하는 것을 방지하고자 한다.

1. 개요

트랜잭션은 필요한 모든 테이블의 잠금을 일괄 획득한 후 수행을 시작한다. 잠금 획득 과정에서 충돌이 발생되면 자원을 점유하고 있는 트랜잭션(T_H)과 요청 트랜잭션(T_R)의 우선순위(P)를 비교하게 된다.

요청 트랜잭션의 우선순위가 더 높지 않은 경우($P(T_R) \leq P(T_H)$), 잠금 획득을 위해 해당 테이블의 대기 리스트에 추가된다. 반면에 더 높은 우선순위를 갖는 경우($P(T_R) > P(T_H)$), 실행가능성 검사를 수행하여 마감시간 내에 수행이 가능한 경우 자원을 선점하게 하고, 그렇지 않은 경우 시스템에서 제거시킨다.

실행가능성 검사는 마감시간까지 남아있는 시간($D(Ti) - t$)과 예상수행시간($E(Ti)$)을 비교하여 해당 트랜잭션(Ti)이 마감시간 내에 정상적으로 완료할 수 있는지 여부를 판단하는 것이다.

$$D(Ti) - t \geq E(Ti)$$

예상수행시간은 트랜잭션 타입별로 일정한 값으로 결정될 수 있으나 주변 환경 및 시스템 부하 변동 등에 따라 그 값이 조금씩 변할 수 있으므로 수행시간 변화 자료를 지속적으로 관찰하여 최적의 값을 예측해 내는 것이 필요하다.

2. 예상수행시간

각 트랜잭션 타입별로 예상수행시간을 계산하는 방

법은 다음과 같이 몇 가지로 나누어 볼 수 있다.

[방법1] 최초 수행 시점(t^0)에서부터 직전 수행 시점(t^{i-1})까지의 모든 수행기록의 평균값을 예상 수행시간으로 사용한다.

[방법2] 직전(t^{i-1}) 수행 시간을 트랜잭션의 예상수행 시간으로 결정한다.

[방법3] 최근 2번(t^{i-2}, t^{i-1}) 수행 시간의 평균값을 이용한다.

[방법4] 최근 3번(t^{i-3}, t^{i-2}, t^{i-1}) 수행 시간의 평균값을 이용한다.

[방법5] (t^{i-2})까지의 누적 평균과 직전(t^{i-1}) 수행 시간과의 중간 값을 이용한다.

예상수행시간 계산 방법에 따라 산출한 결과 값과 실제 수행시간을 나타낸 것이 [그림 1]이다. 여러 가지 방법들 간에 많은 차이를 보이는 것을 알 수 있다. 실제 수행시간과의 오차를 보다 정확하게 비교하기 위해 [그림 2]를 살펴보자.

방법1은 계산상의 복잡성과 현재의 시스템 부하변동률을 제대로 반영하지 못하는 것을 알 수 있다. 방법5도 방법1보다는 오차가 적으나 비슷한 단점을 갖는다. 방법2와 방법4는 실제 수행시간과 매우 유사하지만 방법3에 비해 오차가 다소 큼을 알 수 있다. 결과에서 보는 바와 같이 방법3은 실제수행시간과 약 3~5%정도의 근소한 오차를 보인다. 그리고 계산 방법도 간단하고 시스템 부하 변동에 따른 변화율도 제대로 반영하고 있으므로 예상수행시간 계산에 가장 적합하다.

실제 수행시간과의 오차는 상향 곡선인 경우와 하향 곡선인 경우로 구분하여 보완해주어야 한다. 상향과 하향을 구분하는 것은 t^{i-2} 시점의 수행시간과 t^{i-1} 시점의 수행시간의 크기를 비교하는 것으로 쉽게 구현 가능하다. 상향 곡선을 나타내는 경우(t^{i-2} 시점의 수행시간보다 t^{i-1} 시점의 수행시간이 길다.) 실제 수행시간보다 예상수행시간이 약간 작은 값을 갖기 때문에 이를 이용하여 마감시간 내에 처리 여부를 판단하는 것은 문제가 되지 않는다. 반면, 하향 곡선에서는 오히려 실제수행시간보다 예상수행시간이 약 3~5%정도 더 큰 값을 갖는다. 그래서 이 예상수행시간을 그대로 실행가능성 검사에 사용한다면 마감시간 내에 처리가 가능했던 일부 트랜잭션들이 실행불가능 판단을 받고 시스템에서 제거되는 문제가 발생한다. 따라서 이 문제는 트랜잭션의 수행시간 변화가 하향 곡선으로 판단된 경우에 한하여 예상수행시간을 5% 줄여서 실행가능성 판단에 사용하면 된다.

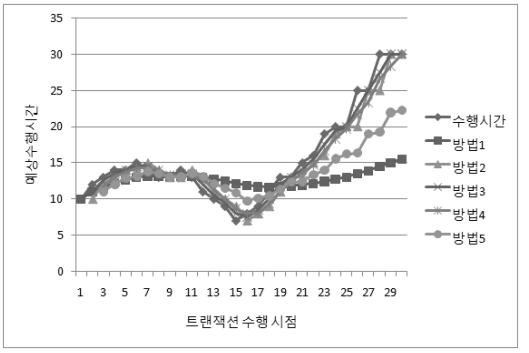

그림 1. 예상수행시간 계산 결과

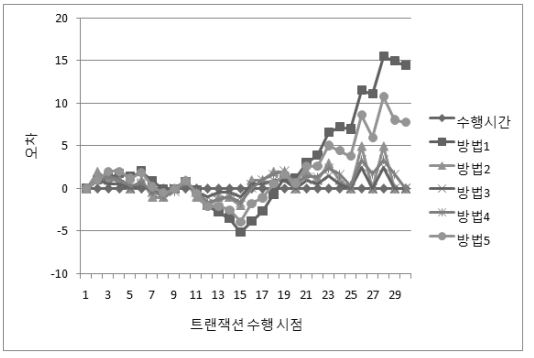

그림 2. 예상수행시간 계산 오차

IV. 성능 평가

성능평가 척도로 사용한 것은 트랜잭션의 마감시간 초과 비율이다. 여기에서 마감시간 초과 비율이란 전체 수행시킨 트랜잭션들 중에서 마감시간 내에 완료되지 못하고 철회된 트랜잭션의 개수를 비율로 나타낸 것이다. 성능 비교는 2단계 잠금 기반의 실시간 병행수행제어 기법인 2PL-HP와 실시간 주기억장치 데이터베이스

시스템을 위한 정적 잠금 기법을 대상으로 하였다.

성능평가에 사용된 컴퓨터는 인텔 펜티엄 IV 프로세서, 2GHz, 512MB이며, OS는 윈도우즈 2000, 컴파일러는 비주얼 C++을 사용하였다. 성능평가에 사용된 시스템 파라미터는 [표 1]과 같다. 성능평가를 위해 100가지 타입의 트랜잭션을 정의하였고, 우선순위 레벨을 10개로 구분하여 총 10,000개의 펌 실시간 트랜잭션을 수행시켜 보았다. 트랜잭션 수행 중에 접근하는 테이블 수는 평균 5개이고, 여유시간은 수행시간의 100~600%로 하였다.

표 1. 시스템 파라미터

파라미터	내 용	설정 값
DBSize	데이터베이스 크기	100 (relations)
TransSize	평균 접근 테이블 수	5 (relations)
CPUtime	평균 처리 시간	5 (ms)
UpdateRatio	쓰기 비율	50 (%)
NumLevels	우선순위 레벨 수	10 (개)
NumTrans	트랜잭션 수	10,000 (개)
NumTypes	트랜잭션 타입 수	100 (개)
Slack	트랜잭션의 여유시간	100~600 (%)

예상수행시간을 이용해 새롭게 수행되는 트랜잭션과 재시작 되는 펌 실시간 트랜잭션의 실행가능성을 판단하고 마감시간 내에 작업을 완료할 수 없는 경우 미리 시스템에서 제거시킨다.

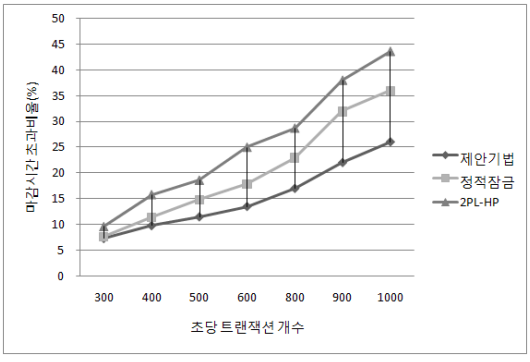

그림 3. 마감시간 초과비율

마감시간 초과비율을 비교한 결과는 그림 3과 같다. 초당 트랜잭션 도착율이 작은 경우에는 대부분의 트랜잭션이 충돌 없이 정상적으로 수행되기 때문에 마감시간을 초과하는 트랜잭션이 많지 않다. 그러나 초당 트랜잭션 도착율이 점점 커지면서 트랜잭션간의 충돌 발생 빈도가 높아지고 충돌 해결 과정에서 낮은 우선순위를 갖는 트랜잭션들은 철회되거나 대기시간이 증가하면서 마감시간을 지키지 못하는 상황이 자주 발생된다.

제안하는 기법은 정적 잠금 기법에 비해 10%, 2PL-HP 기법에 비해 18% 만큼 마감시간 초과 비율이 낮은 것을 알 수 있었다. 이것은 적절한 예상수행시간 계산을 통해 높은 우선순위 트랜잭션에게 자원을 할당하기 전에 실행가능성 검사를 수행하고, 높은 우선순위 트랜잭션이 불필요하게 수행되는 문제를 미리 제거했기 때문이다. 이를 통해 제거된 트랜잭션에 의해 다른 트랜잭션들이 불필요하게 철회되거나 대기하는 문제를 해결하면서 전체적인 시스템 처리 효율을 향상시키고 마감시간 내에 처리되는 트랜잭션의 수를 극대화시켰다.

2PL-HP 기법의 경우 자원을 소유하고 있던 낮은 우선순위 트랜잭션이 처리를 완료하지 못하고 철회되어 재시작 되는 일이 빈번하게 발생한다. 그런데 해당 자원을 획득한 높은 우선순위 트랜잭션이 마감시간 내에 완료가 불가능하여 결국 시스템에서 제거되는 경우, 불필요한 자원 낭비가 발생하게 되고 낮은 우선순위 트랜잭션은 불필요한 철회 때문에 피해를 볼 수밖에 없다. 반대로 높은 우선순위를 갖는 펌 실시간 트랜잭션이 자원을 소유하고 있는 상황에서 낮은 우선순위를 갖는 트랜잭션이 해당 자원을 요청하는 경우, 당연히 높은 우선순위를 갖는 트랜잭션의 계속적인 수행을 보장한다. 그런데 높은 우선순위 트랜잭션이 마감시간 내에 완료가 불가능하여 시스템에서 제거되는 경우 불필요한 자원의 낭비를 초래함은 물론 낮은 우선순위를 갖는 트랜잭션의 불필요한 대기로 인해 마감시간 초과 비율이 높아지는 문제가 발생된다. 정적 잠금 기법에서도 마감시간 내에 수행을 완료할 수 없는 높은 우선순위 트랜잭션이 데이터 객체에 대한 잠금을 획득하도록 허용함으로써 대기 중인 다른 트랜잭션의 대기 시간을 증가시키고 결국 많은 수의 트랜잭션들이 마감시간 내에 수행을

완료할 수 없게 하는 연쇄적인 마감시간 철회 문제를 야기한다.

V. 결 론

제안하는 병행수행제어 기법은 주기억장치 기반 실시간 데이터베이스 시스템을 위해 정적 잠금 기법을 기반으로 한다. 우선순위 정보에 따라 충돌 발생 시 높은 우선순위 트랜잭션의 선점을 보장한다. 그리고 마감시간 내에 수행을 완료할 수 있는 트랜잭션에게 자원을 할당한다. 이를 통해 높은 우선순위를 갖는 트랜잭션의 불필요한 수행을 막고 낮은 우선순위를 갖는 트랜잭션의 불필요한 작업 지연 및 철회 문제를 해결하여 보다 많은 수의 트랜잭션들이 마감시간 내에 처리될 수 있도록 한다.

제안된 기법의 우수성을 입증하기 위해 2PL-HP 기법과 정적 잠금 기법을 비교 대상으로 하였다. 성능평가를 통해 초당 트랜잭션 도착율이 높아질수록 제안하는 기법이 우수한 성능을 보임을 알 수 있었다.

참 고 문 헌

[1] J. Huang, J. A. Stankovic, K. Ramamritham, and D. F. Towsley, "Experimental Evaluation of Real-Time Optimistic Concurrency Control Schemes," in Proc. of 17th VLDB, pp.35-46, 1991.

[2] J. Lee and S. H. Son, "Using dynamic adjustment of serialization order for real-time database systems," in Proc. of Real-Time Systems Symposium, pp.66-75, 1993.

[3] J. R. Haritsa, M. J. Carey, and M. Livny, "On Being Optimistic about Real-Time Constraints," In Proceedings of the ACM Symposium on Principles of Database Systems, 1990.

[4] K. Y. Lam, S. L. Hung, and S. H. Son, "On Using Real-Time Static Locking Protocols for Distributed Real-Time Databases," Journal of Real-Time Systems, Vol.13, No.2, 1997.

[5] O. Ulusoy and A. Buchman, "A Real-Time Concurrency Control Protocol for Main-Memory Database Systems," Information Systems," Vol.23, No.2, 1998.

[6] A. Datta and S. H. Son, "Study of Concurrency Control in Real-Time Active Database Systems," IEEE Transactions on Knowledge and Data Engineering, Vol.14, No.3, 2002.

저 자 소 개

신 재 룡(Jae-Ryong Shin) 정회원

- 1996년 2월 : 충북대학교 정보통신공학과(공학사)
- 1998년 2월 : 충북대학교 정보통신공학과(공학석사)
- 2002년 8월 : 충북대학교 정보통신공학과(공학박사)
- 2003년 3월 ~ 현재 : 광주보건대학 의약정보관리과 교수

<관심분야> : 실시간데이터베이스, 내용기반검색