Thomas 기록 규칙을 이용한 실시간 낙관적 동시성 제어

(Real-Time Optimistic Concurrency Control using Thomas' Write Rule)

김 말 희 † 박 석 ††
(Mal-Hee Kim) (Seog Park)

요 약 낙관적 기법은 실시간 데이타베이스 시스템을 위한 동시성 제어로서 적합하다. 특히, 종료시한을 초과한 트랜잭션이 시스템으로부터 제거되는 firm 실시간 데이타베이스 시스템에서 낙관적 기법은 잠금 기법보다 우수한 성능을 보인다. 그러나 낙관적 기법은 낭비적 수행과 과도한 재시작의 문제를 안고 있다. 종료에 가까운 트랜잭션의 재시작은 시스템 자원의 낭비는 물론 종료시한 초과 가능성을 높인다. 발생되는 재시작의 수를 줄이기 위해서 충돌하는 트랜잭션들간의 직렬화 순서를 동적으로 조정하는 방법이 이용되었다. 그러나 직렬화 순서의 동적 조정 기법을 이용함에도 불구하고 불필요한 재시작이 발생된다. 본 논문에서는 기존의 타임스탬프 기반 동시성 제어에서 이용되던 Thomas 기록 규칙을 이용하여 이러한 불필요한 재시작을 제거한 개선된 실시간 낙관적 동시성 제어 기법을 제안한다. 제안된 방법은 요구되는 데이타베이스 일관성을 보장하면서도 발생되는 재시작 수를 줄임으로써 성능을 향상시킨다.

Abstract Optimistic concurrency control is well suited to real-time database systems. According to several performance evaluation results, especially, in a firm real-time database system that discards tardy transactions, optimistic concurrency control outperforms locking. But optimistic concurrency control has wasted execution problem. In addition, optimistic concurrency control causes heavy restarts. The restarts of near-to-complete transactions significantly increase resources wastes and deadline missing probability of transactions. In order to reduce the number of restarts, the scheme of dynamically adjusting the serialization order of the conflicting transactions has been used. Despite of using dynamic adjustment of serialization order, unnecessary restarts occur. In this thesis, we propose enhanced real-time optimistic concurrency control removing such unnecessary restarts. In the proposed method, we remove unnecessary restarts which are caused by late write operations, using Thomas' write rule that is used in concurrency control based on conventional timestamp ordering scheme.

1. 서 론

실시간 시스템의 가장 큰 특징은 종료시한(deadline)과 같은 시간 제약사항(timing contraints)이다. 이러한 실시간적인 특징과 전통적인 데이타베이스 시스템이 결합된 것이 바로 실시간 데이타베이스 시스템(real-time database systems)이다. 따라서 실시간 데이타베이스 시스템에서의 트랜잭션 처리는 실시간적인 일관성과 데이타베이스 일관성을 유지해야만 한다. 즉, 종료시한으로 표현되는 시간 제약사항을 준수해야 하는 동시에 데이타베이스의 일관성(database consistency)을 유지해야만 한다[7,8,13].

실시간 데이타베이스 시스템에서는 트랜잭션 처리의 적시성 수준(timeliness level)[1] 으로 성능을 평가한다.

· 이 연구는 1998년 한국 과학재단 핵심 전문연구비 지원에 의한 결과임 (과제번호 981-0912-110-2)
† 정 회 원 : 한국전자통신연구원 정보보호기술연구본부 연구원
marickim@etri.re.kr
†† 종신회원 : 서강대학교 컴퓨터학과 교수
spark@dblab.sogang.ac.kr
논문접수 : 1998년 9월 16일
심사완료 : 2000년 10월 30일

1) 적시성 수준 : 트랜잭션이 사용한 데이타와 트랜잭션이 수행한 작업이 시간적으로 얼마나 적절하게 처리되었는가를 나타

즉, 실시간 데이타베이스 시스템의 목적은 요구되는 일관성 조건을 만족하면서 종료시한을 초과하는 트랜잭션의 수를 최소화하는 것이다. 연구 결과[3]에 따르면 다양한 동시성 제어 기법 중에서 펌 실시간 데이타베이스 시스템(firm real-time database systems)에서는 잠금(locking)보다 낙관적 동시성 제어(optimistic concurrency control)의 성능이 우수하다[3]. 펌 실시간 데이타베이스에서는 종료시한이 초과된 트랜잭션은 시스템으로부터 제거된다. 낙관적 동시성 제어는 검증 단계에 충돌을 해결하므로 낭비되는 재시작(wasted restart)[2] 없이 결국 완료될 트랜잭션에 의해서만 다른 트랜잭션이 재시작되기 때문이다[2,3]. 그러나 낙관적 동시성 제어는 과도한 재시작의 문제를 갖는다[1,2,3]. 재시작은 종료시한 만족을 어렵게 한다. 따라서 재시작을 최소화하기 위해서 직렬화 순서를 동적으로 조정하는 낙관적 동시성 제어가 제안되었다[1,3].

잠금과 낙관적 동시성 제어를 통합하는 새로운 동시성 제어 기법들 또한 제안되었다[9,10,11,12,14]. 제안된 기법들은 가능한 모든 직렬화 순서에 대응되도록 동일한 트랜잭션에 대해서 다수의 트랜잭션 버전을 생성함으로써 종료시한 만족을 최대화한다. 이러한 동시성 제어에는 다음과 같이 두가지로 분류된다. 낙관적 동시성 제어에 기초하여 잠금을 통합시킨 투기적인 동시성 제어(speculative concurrency control)와 잠금에 기초하여 낙관적 동시성 제어를 통합한 대체 버전 동시성 제어(alternative version concurrency control)가 그것이다. 하지만 이러한 동시성 제어 기법들은 버전의 갯수가 많아질수록 버전 관리의 복잡성과 많은 자원을 필요로 한다. 따라서, 사용가능 자원이 제한적인 펌 실시간 데이타베이스 시스템을 위한 동시성 제어로는 부적합하다.

본 논문에서는 제안하는 기법은 사용가능 자원이 제한적인 펌 실시간 데이타베이스 시스템에서 효과적인 제어방법이다. 즉, 이미 성능의 우수성이 입증된 직렬화 순서의 동적 조정 기법을 이용한 실시간 낙관적 동시성 제어에, 이 방법이 안고있는 문제점인 불필요한 재시작을 Thomas 기록 규칙을 적용하여 제거함으로써 성능의 향상을 가져온 개선된 실시간 낙관적 동시성 제어 기법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 본 논문이 기초하고 있는 직렬화 순서의 동적 조정 기법을 이용한 실시간 낙관적 동시성 제어에 대해서 살펴본다. 3장에서는 Thomas 기록 규칙을 적용한 개선된 실시간 낙관적 동시성 제어기법을 제안하고 타당성을 보인다. 4장에서는 제안한 동시성 제어 기법의 성능을 평가하고, 5장에서는 결론을 내린다.

2. 낙관적 동시성 제어

실시간 데이타베이스 시스템을 위한 동시성 제어 기법으로서는 낙관적 동시성 제어가 적합하다.

낙관적 동시성 제어에서 트랜잭션 수행은 판독 단계, 검증 단계, 기록 단계의 세단계로 진행되며 중지되거나 완료될 때까지 방해받지 않고 진행된다. 직렬성을 보장하기 위한 기본적인 원리는 다음과 같이 모두 같다.

낙관적 동시성 제어 기법에서 직렬화 순서상 트랜잭션 T_i가 트랜잭션 T_j 앞에 오기 위해서는 다음의 두 가지 조건이 만족되어져야 한다[1,2].

조건 1 : T_i의 기록이 T_j의 기록을 덮어쓸 수 없다.

조건 2 : T_i의 기록이 T_j의 판독 단계에 영향을 줄 수 없다.

그러나 실시간 낙관적 동시성 제어는 과도한 재시작의 문제를 안고 있다. 이러한 과도한 재시작 수를 줄이기 위해서 직렬화 순서의 동적 조정을 이용한 실시간 낙관적 동시성 제어 기법이 제안되었다[1,2]. 성능 평가[3] 결과에 의하면 직렬화 순서의 동적 조정을 이용한 낙관적 동시성 제어가 다른 동시성 제어 방법보다 펌 실시간 데이타베이스 환경에서 성능이 가장 우수하다.

예 1은 두 조건을 모두 만족하면서 직렬화 순서를 동적으로 조정하여 낙관적 기법에서 불필요하게 발생되는 재시작을 제거한다[1,2].

예 1 : 다음과 같은 트랜잭션 T_1, T_2, T_3와 그림 1과 같은 수행기록을 가정한다.

T_1 : $R_1[x]$ $W_1[x]$ $R_1[y]$ $W_1[y]$ V_1

T_2 : $R_2[x]$ $W_2[x]$...V_2

T_3 : $R_3[y]$...V_3

R은 판독 연산,W는 기록 연산, V는 검증,C는 완료를 나타낸다.

	1	2	3	4	5	6	7	8
T_1	$R_1[x]$	$W_1[x]$				$R_1[y]$	$W_1[y]$	V_1/C_1
T_2			$R_2[x]$		$W_2[x]$			A_2
T_3				$R_3[y]$				A_3

그림 1 예 1의 수행기록

기본적인 낙관적 동시성 제어 기법에 따라 수행하게 되면 직렬화 순서는 $T_1 \rightarrow T_2$, $T_1 \rightarrow T_3$가 되고,

---

내는 기준

2) 낭비되는 재시작 : 다른 트랜잭션을 재시작시킨 트랜잭션이 종료시한 초과로 취소되는 경우

조건 2의 판독 의존 관계로 T_2, T_3 모두 재시작된다.

그러나, 예 1의 T_1이 검증 단계에 있을 때 T_3는 T_1에 대해서 판독-기록 충돌만을 발생시킨다. T_3와 같은 경우 직렬화 순서를 T_1의 검증 단계동안에 $T_3 \rightarrow T_1$으로 조정한다면 판독 의존 관계가 성립되지 않으므로 재시작되지 않아도 된다. 검증 단계에 있는 트랜잭션에 대해서 판독-기록 충돌만을 발생시키는 이러한 충돌을 조정 가능한 충돌이라고한다. 이러한 충돌은 발견 즉시 재시작시키지 않는다. 판독-기록 충돌 이외에 기록-기록 충돌까지 포함하는 T_2와 같은 트랜잭션은 조정 불가능한 충돌이라고 한다. 이러한 충돌은 발견 즉시 재시작한다[1,2]. 직렬화 순서를 유도할 수 있는 충돌에는 다음 세가지가 있다[1,2]. 단, RS(T)와 WS(T)는 각각 트랜잭션 T의 판독 집합과 기록 집합, 그리고 T_v는 검증 단계의 트랜잭션, T_a는 판독 단계의 트랜잭션을 나타낸다.

i) 판독-기록 충돌($RS(T_v) \cap WS(T_a) \neq \varnothing$)

전방향 조정(forward adjustment)으로 직렬화 순서 $T_v \rightarrow T_a$를 유도한다. T_a의 기록은 T_v의 판독 단계에 영향을 주지 않는다.

ii) 기록-판독 충돌($WS(T_v) \cap RS(T_a) \neq \varnothing$)

후방향 조정(backward adjustment)으로 직렬화 순서 $T_a \rightarrow T_v$를 유도한다. T_v의 기록은 T_a의 판독 단계에 영향을 주지 않는다.

iii) 기록-기록 충돌($WS(T_v) \cap WS(T_a) \neq \varnothing$)

전방향 조정(forward adjustment)으로 직렬화 순서 $T_v \rightarrow T_a$를 유도한다. T_v의 기록은 T_a의 기록을 덮어 쓰지 않는다.

직렬화 순서의 동적 조정을 이용한 실시간 낙관적 동시 제어 기법이 불필요하게 재시작되어지는 트랜잭션의 수를 줄였지만 여전히 트랜잭션의 불필요한 재시작이 있을 수 있는 문제를 가지고 있다. 직렬화 순서의 동적 조정을 이용한 실시간 낙관적 동시 제어 기법에서는 후방향 조정된 트랜잭션이 전방향 조정을 필요로 하게 되면 조정 불가능 충돌로 재시작될 수 밖에 없다. 그런데 이러한 트랜잭션의 재시작 중 불필요한 재시작이 있을 수 있다. 다음 그림 2는 직렬화 순서의 동적 조정을 이용한 실시간 낙관적 동시 제어 기법에서 발생할 수 있는 불필요한 재시작의 경우의 수행 기록이다.

직렬화 순서의 동적 조정을 이용하면 그림 2의 수행

	1	2	3	4	5
T_1	$R_1[x]$			$W_1[x]$	V_1/C_1
T_2		$W_2[x]$	V_2/C_2		

그림 2 불필요한 재시작이 발생될 수 있는 수행기록

```
If WS(Tv) ∩ RS(Ta) ≠∅ then
    Backward adjust serialization order(Ta → Tv)
endif
For each backward adjusted transaction Ta
    If RS(Tv) ∩ WS(Ta) ≠∅ or WS(Tv) ∩ WS(Ta) ≠
       ∅ then
            Restart Ta
    endif
enddo
```

그림 3 직렬화 순서의 동적 조정을 이용한 실시간 낙관적 동시성 제어

기록은 $T_1 \rightarrow T_2$의 직렬화 순서를 유지하면서 $T_2 \rightarrow T_1$의 완료 순서를 갖게 된다. 따라서, 후방향 조정된 T_1이 T_2에 대해서 판독 의존 관계나 덮어 쓰기를 시도하게 되면 직렬성이 위반되므로 T_1은 $W_1[x]$ 연산에 의해서 재시작된다.

그러나 그림 2의 경우 뷰 동등[3] 에 기반한 직렬성을 보장하는 기법을 이용한다면 T_1을 재시작시키지 않고 완료시킬 수가 있다. 그림 2의 시간 5에서의 $W_1[x]$을 판독하는 트랜잭션은 존재하지 않는다. 따라서 $W_1[x]$을 무시하면 $T_1 \rightarrow T_2$의 직렬화 순서에 의한 수행 결과와 같아지게 된다. 따라서 T_1을 재시작시키지 않고도 직렬성을 보장하면서 T_1,T_2 모두 완료 시킬 수가 있다.

그림 3은 직렬화 순서의 동적 조정을 이용한 실시간 낙관적 동시성 제어를 나타낸다[1,2].

3. Thomas 기록 규칙을 적용한 실시간 낙관적 동시성 제어

직렬화 순서의 동적 조정을 이용한 낙관적 동시성 제어 기법은 트랜잭션의 완료 순서와 직렬화 순서가 일치하지 않는다는 점에서 기존의 낙관적 동시성 제어 기법과는 차별화된다[1,2].

직렬화 순서의 동적 조정을 이용한 실시간 낙관적 동시 제어 기법에서는 후방향 조정된 트랜잭션이 전방향 조정을 필요로 하게 되면 조정 불가능 충돌로 재시작될 수 밖에 없는데, 후방향 조정된 트랜잭션의 이러한 직렬성 위반 연산 검사 메커니즘으로 타임스탬프 순서화 기법을 이용한다. 각 트랜잭션은 검증 단계의 마지막 단계에 동적으로 타임스탬프를 할당받고 판독 단계에는 가능한 최대 완료 타임스탬프를 유지한다. 즉 후방향 조정

3) 뷰 동등 : 수행기록 H와 H'가 같은 트랜잭션 집합과 같은 연산에 대해서 정의되고, 모든 판독 연산이 동일한 트랜잭션에 의한 기록에 대한 것이고, 모든 데이타에 대한 기록 연산이 동일한 트랜잭션에 의한 것일 때, H와 H'은 뷰 동등하다.

된 트랜잭션의 경우 자신을 후방향 조정시킨 트랜잭션보다는 작은 가능한 최대 완료 타임스탬프를 갖게 된다. 이들간의 관계를 이용하여 후방향 조정된 트랜잭션의 이미 완료된 트랜잭션에 대한 직렬성을 보장한다[1].

그러나 앞의 그림 2의 경우와 같이 후방향 조정된 트랜잭션의 불필요한 재시작을 줄이기 위하여 뷰 동등에 기반한 직렬성을 보장하는 Thomas 기록 규칙을 적용한 타임스탬프 순서화 기법을 이용한다면 후방향 조정된 트랜잭션을 재시작시키지 않고 완료시킬 수도 있다.

3.1 Thomas 기록 규칙(TWR)

아래 그림 4의 스케줄은 기본적인 타임스탬프 순서화를 이용하여 모두 완료할 수 없다. $T_1 \rightarrow T_2$, $T_2 \rightarrow T_1$의 어떤 직렬 수행과도 같은 결과를 내지 못하기 때문이다.

	1	2	3	4	5
T_1	$R_1[x]$			***$W_1[x]$***	C_1
T_2		$W_2[x]$	C_2		

그림 4 Thomas 기록 규칙의 개념

그림 4에서 수행 순서에 따라 $TS(T_1) < TS(T_2)$이고, $W_1[x]$이 수행되려고 할 때 이미 T_2가 $W_2[x]$를 수행하고 완료했다. 따라서, $TS(T_2) > TS(T_1)$이므로 T_1은 재시작된다. $TS(T_i) < TS(T_2)$인 T_i의 경우 x에 대한 판독을 시도하게 되면 T_1과 같은 이유로 재시작되고, $TS(T_j) > TS(T_2)$인 T_j의 경우 x에 대한 판독을 시도하면 T_1이 아닌 T_2가 기록한 값을 판독해야 한다. 즉, $W_1[x]$를 판독하는 트랜잭션은 존재하지 않으므로 무시되어도 직렬성이 보장된다. 만약 트랜잭션 T_i가 $W_i[x]$를 발생시켰다면 그림 5의 규칙에 따른다[7].

```
if TS(Ti) < RTS(x)   then  restart Ti
if TS(Ti) < WTS(x)   then  ignore Wi[x]
else , execute Wi[x]  and  WTS(x):= max{WTS(x), TS(Ti)}
```

* *RTS(x)* : x에 대한 판독을 수행한 완료된 트랜잭션 중 가장 큰 타임스탬프

* *WTS(x)* : x에 대한 기록을 수행한 완료된 트랜잭션 중 가장 큰 타임스탬프

그림 5 Thomas 기록 규칙

3.2 Thomas 기록 규칙의 적용

Thomas 기록 규칙을 적용할 수 있는 트랜잭션은 후방향 조정된 트랜잭션으로 후방향 조정에 관계된 판독 연산과 Thomas 기록 규칙 적용 대상이 되는 기록 연산을 모두 갖는 갱신 트랜잭션이다. 직렬화 순서의 동적 조정 기법을 이용한 실시간 낙관적 동시성 제어에서 발생되는 불필요한 재시작을 판독단계에 Thomas 기록 규칙을 적용하여 제거할 수 있다.

자료 구조	정 의
WTS(X)	데이타X에 기록 연산을 수행한 완료된 트랜잭션들 중 가장 큰 타임스탬프
RTS(X)	데이타X에 판독 연산을 수행한 완료된 트랜잭션들 중 가장 큰 타임스탬프
MaxTS(T_i)	T_i 가 가질 수 있는 가장 큰 완료 타임스탬프
TS(T_i)	T_i 의 최종 타임스탬프
WTS_R(T_i,X)	T_i 가 X를 판독하려고 하는 순간의 WTS(X)
Time_W(T_i,X)	T_i 가 X를 자신의 작업 영역에 기록한 시간
RS(T_i) / WS(T_i)	T_i 의 판독 집합/ 기록 집합
BeforeSet(T_i)	후방향 조정에 의해서 T_i보다 앞서는 직렬화 순서를 갖는 활성 트랜잭션들의 리스트

그림 6 실시간 낙관적 병행수행 관련 자료 구조

다음 그림 6은 직렬화 순서의 동적 조정 기법을 이용한 실시간 낙관적 동시성 제어에 이용되는 자료구조이다.

직렬화 순서의 후방향 조정을 경험한 트랜잭션(T_v)이 검증 단계에 도달했을 때 현재 판독 단계의 충돌하는 활성 트랜잭션의 직렬화 순서를 조정하기에 앞서서 이미 완료한 트랜잭션에 대한 직렬성 검사를 하게 된다[1]. 이 단계에 Thomas 기록 규칙을 적용한다. 즉, 기록 단계에 Thomas 기록을 적용한 [14]와는 달리 본 논문은 검증 단계에 적용한다. 그림 7은 Thomas 기록 규칙을 적용하는 프로토콜을 기술한다.

```
foreach X in RS(Tv)
if  WTS_R(Tv,X) > MaxTS(Tv)  then  restart(Tv)  endif
enddo
foreach X in WS(Tv)
if Time_W(Tv,X)  >  MaxTS(Tv) then
if MaxTS(Tv) < RTS(X) then restart(Tv) endif
if WTS_R(Tv,X) > MaxTS(Tv) then ignore Wv[x]
endif (Thomas 기록 규칙을 적용)
endif
enddo
```

그림 7 Thomas 기록 규칙을 적용한 실시간 낙관적 동시성 제어

3.3 타당성의 증명

후방향 조정된 트랜잭션이 직렬화 순서상 뒤에 오는 이미 완료된 트랜잭션과 동일한 데이타에 대하여 기록 연산을 수행하려고 할 때, 이러한 기록에 대한 판독을

수행하는 트랜잭션이 존재하지 않음을 보인다. 따라서 직렬성을 위반하는 기록 연산을 무시하고 Thomas 기록 규칙을 적용하게 되면 뷰 동등에 기반한 직렬성이 보장되고[4] 트랜잭션은 완료될 수 있다.

직렬화 순서의 동적 조정을 이용한 낙관적 동시성 제어에서의 직렬화 순서는 검증 단계에 할당받는 TS값과 판독 단계에 유지하게 되는 가능한 최대 완료 타임스탬프에 의해서 정해진다. 완료 순서는 검증 단계에 도착한 순서가 되지만 직렬화 순서는 완료 순서와 다를 수 있다[1,2]. 또한 후방향 조정된 트랜잭션은 그렇지 않은 트랜잭션보다 직렬화 순서상 뒤에 올 수 없다.

T를 T_v가 검증 단계에 들어갔을 때 활성 트랜잭션의 집합이라고 하자. 그렇다면, $WS(T_v) \cap RS(T_a) \neq \varnothing$ ($T_a \in T$) 인 트랜잭션 T_a 에 대해서 $T_a \rightarrow T_v$의 직렬화 순서가 만들어지고, T는 그림 8과 같이 이분된다.

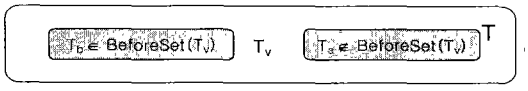

그림 8 활성 트랜잭션 집합 T의 분리 (1)

다음으로 T_b ($T_b \in$ BeforeSet(T_v))가 검증 단계에 도착했다고 하자. 그렇다면 BeforeSet(T_v)가 또다시 T_b를 중심으로 그림 9와 같이 이분된다.

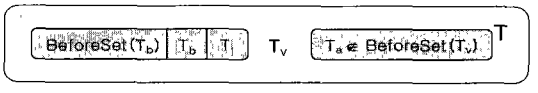

그림 9 활성 트랜잭션 집합 T의 분리 (2)

후방향 조정된 트랜잭션은 검증 단계에서의 직렬화 순서의 조정에 앞서 이미 완료된 트랜잭션들에 대한 직렬화 가능성을 검사해야 한다. 다음 그림 10-12들은 T_v가 검증 단계에 있을 때 T_b가 후방향 조정되고 이어서 T_b가 검증 단계에 들어간다.

그림 10은 판독 연산 $R_b[y]$에 대한 직렬성 검사 과정을 나타낸다. 수행기록 H가 H = $R_b[x]$ $W_v[x]$ $W_v[y]$ V_v C_v ***$R_b[y]$*** V_b와 같을 때, $MaxTS(T_b) < WTS_R(y) = TS(T_v)$이므로 T_b는 재시작된다.

	1	2	3	4	5	6
T_b	$R_b[x]$				***$R_b[y]$***	V_b/C_b
T_v		$W_v[x]$	$W_v[y]$	V_v/C_v		

그림 10 직렬성을 위반하는 판독 연산

그림 11은 기록 연산 $W_b[y]$에 대해서 직렬성 검사를 한다. 수행기록 H가 H = $R_b[x]$ $R_v[x]$ $W_v[y]$ V_v C_v ***$W_b[y]$*** V_b와 같을 때, $MaxTS(T_b) < Time\_W(T_b,y)$ 이고 $MaxTS(T_b) < RTS(y)=TS(T_v)$이므로 T_b은 재시작된다.

	1	2	3	4	5	6
T_b	$R_b[x]$				***$W_b[y]$***	V_b/C_b
T_v		$R_v[y]$	$W_v[x]$	V_v/C_v		

그림 11 직렬성을 위반하는 기록 연산(1)

그림 12는 기록 연산 $W_b[x]$에 대해서 직렬성 검사를 한다. 수행기록 H가 H = $R_b[x]$ $W_v[x]$ V_v C_v ***$W_b[x]$*** V_b와 같을 때, $MaxTS(T_b) < Time_W(T_b,x)$이고 $MaxTS(T_b) < WTS(x)=TS(T_v)$이다. 따라서 T_b는 재시작된다. 그러나 T_b에 Thomas 기록 규칙을 적용하여 기록 연산 $W_b[x]$을 무시한다면 T_b는 성공적으로 완료될 수 있다.

	1	2	3	4	5
T_b	$R_b[x]$			***$W_b[x]$***	V_b/C_b
T_r		$W_v[x]$	V_v/C_v		

그림 12 직렬성을 위반하는 기록 연산(2)

직렬성을 위반하는 기록 연산(2)의 경우를 자세히 살펴본다. 그림 9에서 T_j는 T_v에 의해서 후방향 조정되고 T_b와는 충돌이 없거나, 기록-기록 충돌이나 판독-기록 충돌이 있는 트랜잭션이다. 다음의 가능한 모든 경우(그림 13-15)들을 통해서 T_b가 재시작되거나 Thomas 기록 규칙을 적용하여 완료되거나 모두 $R_j[z]$에 대해서 $WTS(z) > MaxTS(T_j)$ 이므로 이미 완료된 트랜잭션 T_v의 영향으로 T_j는 재시작된다.

	1	2	3	4	5	6	7	8	9	10	11
T_b	$R_b[x]$		$R_b[y]$					***$W_b[z]$***	V_b/?		
T_j		$R_j[x]$				$W_j[y]$				$R_j[z]$	V/A
T_v				$W_v[x]$	***$W_v[z]$***		V_v/C_v				

그림 13 T_j와 T_{ab}는 판독-기록 충돌

	1	2	3	4	5	6	7	8	9	10	11
T_b	$R_b[x]$		$W_b[y]$					***$W_b[z]$***	V_b/?		
T_j		$R_j[x]$				$W_j[y]$				$R_j[z]$	V/A
T_v				$W_v[x]$	***$W_v[z]$***		V_v/C_v				

그림 14 T_j와 T_{ab}는 기록-기록 충돌

	1	2	3	4	5	6	7	8	9	10
T_b	$R_b[x]$		$W_b[y]$				***$W_b[z]$***	V_b/?		
T_i		$R_i[x]$							$R_i[z]$	V_i/A
T_v				$W_v[x]$	***$W_v[z]$***	V_v/C_v				

그림 15 T_j와 T_{ab}는 충돌이 없는 경우.

그림 13-15을 통해서 $TS(T_b) < MaxTS(T_i) < TS(T_v)$ 인 트랜잭션 T_j가 z에 대한 판독 연산을 갖는다면 재시작됨을 보였다. 또한 $MaxTS(T_a) > TS(T_v)$ 인 트랜잭션 T_a가 z에 대한 판독 연산을 갖는다면 T_v의 기록을 판독해야 한다. 따라서, T_b의 z에 대한 기록 $W_b[z]$을 판독하는 트랜잭션은 존재하지 않는다. 따라서, $W_b[z]$을 무시한다면 직렬 수행 $T_b \rightarrow T_v$ 과 뷰 동등하므로 직렬성이 보장되고 T_b는 성공적으로 완료될 수 있다.

직렬화 순서의 동적 조정을 이용한 실시간 낙관적 동시성 제어에서 후방향 조정되지 않은 트랜잭션은 기본적인 검증 기법으로 충분히 직렬성을 보장할 수 있다. 하지만 후방향 조정된 트랜잭션에 대해서는 기본적인 검증 기법으로는 직렬성을 보장하지 못하므로 타임스탬프 기법을 이용한다. 타임스탬프를 이용한 동시성 제어에서 판독되지 않는 늦은 기록 연산에 대해서는 Thomas 기록 규칙을 적용하여 무시해도 뷰 동등에 기반한 직렬성이 보장되므로 수행을 완료할 수 있다.

이상으로 폼 실시간 데이타베이스 시스템에서 가장 우수한 성능을 보이는 직렬화 순서의 동적 조정을 이용한 실시간 낙관적 동시성 제어에 Thomas 기록 규칙을 적용하여 재시작 수를 줄이는 방법과 효과 및 그 타당성을 보였다.

4. 성능 평가

성능 평가의 대상은 직렬화 순서의 동적 조정을 이용한 실시간 낙관적 동시성 제어[2]와 본 논문에서 제안하는 Thomas 기록 규칙을 적용한 개선된 실시간 낙관적 동시성 제어이고, 평가의 기준은 트랜잭션의 재시작 비율과 종료시한 만족 비율이다. 본 논문에서 제안하는 알고리즘의 성능을 평가하기 위해 그림 16의 시스템 모델을 사용하였고 시뮬레이션 툴로는 AweSim[15]을 이용해서 수행되었다.

가정하는 시스템 환경은 한 개의 CPU와 한 개의 디스크를 가진다. 트랜잭션에 대해서는 다음과 같은 식을 이용하여 종료시한과 우선순위를 결정하였다.

종료시한 = 도착시간 + 트랜잭션 연산 수 * OP_TIME * SLACK

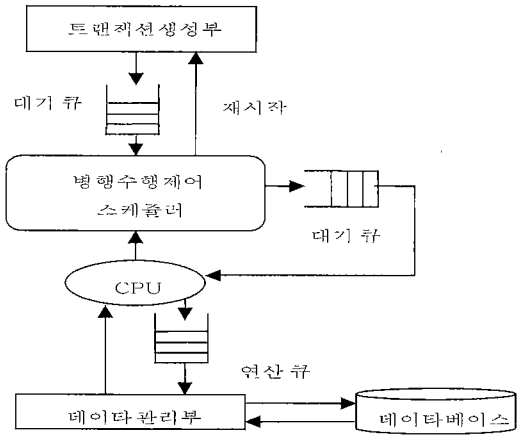

그림 16 시뮬레이션 모델

우선순위 = 종료시한 / 100000.0 (earliest-deadline-first policy)

여유 지연 시간인 SLACK 시간은 UNIFORM(10.0, 20.0)함수를 이용하여 균일하게 할당하고, 연산을 수행하기 위해서 소요되는 시간인 OP_TIME은 연산에 대한 CPU 서비스 시간을 의미한다.

다음 그림 17은 성능 평가를 위해 가정한 매개변수들이다. 매개변수는 [1]의 시뮬레이션에서 사용한 값을 이용하였으며 판독 전용 트랜잭션대 갱신 트랜잭션의 비율과 기록 확률은 본 논문이 제안하는 상대적 시간 일관성 및 데이타 종료시한의 성능 평가를 위해서 조정된 값이다.

매 개 변 수	값
데이타베이스 크기	200 table
시간 데이타의 비율	5%
센서 트랜잭션의 주기	50.0~200.0 ms
연산에 대한 CPU 서비스 시간	3ms
디스크 입출력 시간	25ms
트랜잭션 취소시 소요 시간	20ms
데이타가 버퍼에 있을 확률	0.5
최대 트랜잭션 수	250
종료시한 수식에서 최대 SLACK	20.0
종료시한 수식에서 최소 SLACK	10.0
판독 전용 트랜잭션:일반 트랜잭션	55 : 45
판독 전용 트랜잭션중 RTC 판독전용 트랜잭션 비율	60%
갱신 확률	80%

그림 17 성능 평가를 위해 가정된 매개변수들

- table : lock을 걸 수 있는 최소 단위

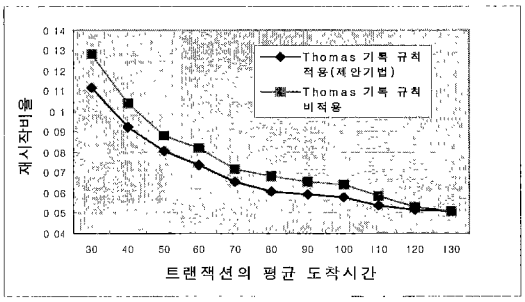

그림 18 재시작 비율 비교

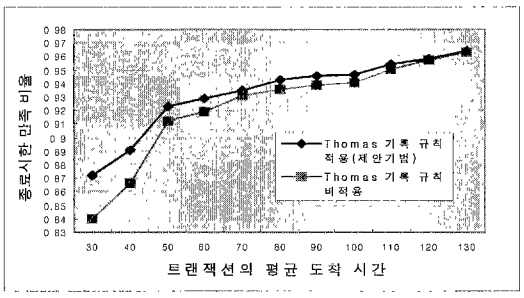

그림 19 종료시한 만족 비율 비교

그림 18과 19는 실시간 낙관적 동시성 제어 기법에 Thomas 기록 규칙을 적용한 경우와 적용하지 않은 경우의 재시작 비율과 종료시한 만족 비율을 평가한 그림이다. 우선 그림 18은 트랜잭션의 도착 간격시간이 변화함에 따라, Thomas 기록 규칙을 적용한 경우와 적용하지 않은 경우의 재시작 비율의 비교를 나타낸다. 트랜잭션의 도착 간격시간이 적어짐에 따라 트랜잭션 사이의 데이타 충돌이 발생하는 확률이 높아지기 때문에 전체적으로 재시작 비율이 높아지는 곡선을 그리게 되고, 데이타 충돌이 발생하는 확률이 높아짐에 따라 Thomas 기록 규칙을 적용하는 확률도 높아져 재시작 되는 비율이 줄어들게 된다.

그림 19는 Thomas 기록 규칙을 적용한 경우와 적용하지 않은 경우의 종료시한 만족 비율의 비교를 나타낸다. 트랜잭션의 재시작 비율이 낮아짐에 따라 트랜잭션의 처리시간이 빨라질 것이고, 트랜잭션의 처리시간이 빨라지면 그 만큼 트랜잭션이 자신의 종료시한 내에 수행을 완료하는 확률이 높아지기 때문에 제안하는 기법이 트랜잭션의 종료시한을 만족하는 비율이 높아지게 된다.

5. 결 론

기존의 실시간 데이타베이스 시스템에서의 동시성 제어 기법들은 데이타베이스의 일관성을 만족하면서 가능한 많은 트랜잭션들이 종료시한을 만족하도록 하는 것에 중점을 두어 왔다. 종료시한 만족을 최대화하기 위해서 기본적인 동시성 제어 방법인 잠금과 낙관적 동시성 제어를 통합하거나, 동일한 트랜잭션에 대해서 여러 개의 버전을 생성하거나, 직렬화 순서를 동적으로 조정하는 기법들을 이용해왔다. 연구 결과에 따르면 사용 가능 자원이 제한적인 펌 실시간 데이타베이스 시스템에서는 직렬화 순서의 동적 조정을 이용한 실시간 낙관적 동시성 제어 기법이 우수한 것으로 밝혀졌다[2].

본 논문에서는 직렬화 순서의 동적 조정을 이용한 실시간 낙관적 병행수행 기법에 Thomas 기록 규칙을 적용하여 재시작 수를 줄이는 개선된 실시간 낙관적 동시성 제어 기법을 제안하고 타당성을 증명했다. 제안된 방법은 뒤늦은 갱신 연산에 의한 재시작을 제거함으로써 갱신 비율이 높은 환경에서의 실시간 낙관적 동시성 제어 기법의 성능을 향상시킨다. 즉, 직렬화 순서상 뒤에 오는 트랜잭션에 의해 이미 기록 연산이 수행된 동일한 데이타에 대한 기록 연산을 수행하려는 트랜잭션에 Thomas 기록 규칙을 적용하여 재시작시키지 않고 완료시킨다. 따라서 재시작 수가 감소하고 종료시한을 만족하는 트랜잭션의 수를 증가시킬 수가 있다. 이러한 성능상의 향상은 성능 평가를 통해서 보였다.

참 고 문 헌

[1] K.W. Lam, K.Y. Lam, and S.L. Hung, Real-Time Optimistic Concurrency Control Protocol with Dynamic Adjustment of Serialization Order, *Proceedings of IEEE Real-Time Technology and Application Symposium*, pp. 174-179, May 1995 .

[2] J. Lee and S.H.Son, Using Dynamic Adjustment of Serialization Order for Real-Time Database Systems, *Proceedings of the 14th IEEE Real-Time Systems Symposium*, pp. 66-75, December 1993.

[3] J. Lee and S.H.Son, *Performance of Concurrency Control Mechanisms in Centralized Database Systems*, pp. 429-460 Prentice-Hall.

[4] P. A. Berstein, V. Hadzilacos, and N. Goodman, *Concurrency Control and Recovery in Database Systems*, pp. 25-46, Addison-Wesley Publishing Company.

[5] H.F. Korth and A. Silberschatz, *Database System Concepts*, pp. 347-394 McRaw-Hill Computer Science Series, 2nd Edition.

[6] B. Kao and H. Garcia-Molina, *Advances in*

Real-Time Systems, chapter 19. Prentice Hall
[7] S.H. Son, S. Park and Y. Lin, An Integrated Real-Time Locking Protocol, *Proceedings of the 8th International Conference on Data Engineering*, pp. 527-534 February 1992
[8] L.C. Shu and M. Young, *Correctness Criteria and Concurrency Control for Real-Time System : A Survey*, Technical Report No. SERC-TR-131-P, Univ. of Purdue, November 1992.
[9] A. Bestavor, *Speculative Concurrency Control for Real-Time Databases*, Technical Report 93-002, Univ. of Boston, January 1993.
[10] S. Braoudakis, *Concurrency Control Protocols for Real-Time Databases*, Technical Report BU-CEIS-95-14, Univ. of Boston , 1995 .
[11] D. Hong, S. Chakravarthy and T. Johnson, *Alternative Version Concurrency Control(AVCC) for firm real-time database system*, Technical Report UF-CIS-TR-95-031, univ. of Deiaware, 1995.
[12] D. Hong, S. Chakravarthy and T. Johnson, Locking Based Concurrency Control for Integrated Real-Time Database System, *Online-Proceedings of the 1st International Workshop on Real-Time Databases,* pp. 138-143, March 1996.
[13] K. Ramamritham, Real-Time Databases, Distributed and Parallel Databases, Vol.1, No 1, pp. 199-226, January 1993.
[14] 윤 인호, 박석, 실시간 데이타베이스 시스템의 혼합 트랜잭션 환경에서 직렬화 순서의 동적 조정 방식을 이용한 동시성 제어, 동계 데이타베이스 학술대회 논문집 13권 1호, pp. 151-156, 1997.2.
[15] A. Alan and B. Pritsker, Simulation with Visual Slam *and Awesim* , System Publishing Co., 1997

김 말 희
1996년 서강대학교 전자계산학과 (공학사). 1998년 서강대학교 공과대학원 전자계산학과 졸업(공학석사)-세부전공 데이타베이스. 1998년 2월 ~ 2000년 10월 삼성전자 통신연구소 근무(IMT 2000 단말기호처리 s/w 개발). 2000년 11월 ~ 현재 한국전자통신연구원 정보보호기술연구본부 무선인터넷 정보보호연구팀 근무. 관심분야는 실시간 데이터베이스 시스템, 병행수행 제어기법

박 석
정보과학회논문지 : 데이타베이스
제 27 권 제 1 호 참조