공장 자동화를 위한 데이타베이스 시스템에서의 다중 버전 실시간 트랜잭션의 시뮬레이션

Simulation of Multiversion Real-time Transactions in Database Systems for Factory Automation

유 인 관*, 박 성 진*, 백 두 권*
In-Kwan Yu*, Seong-Jin Park*, Doo-Kwon Baik*

Abstract

In real-time database systems, transactions's commitment done before the given deadlines is more important than just getting the maximum throughput. Transactions missing the given deadlines are no longer meaningful in real-time applications.

Therefore, there is a need for new transaction processing models to meet the given deadlines in real-time database applications, because most conventional transaction models are not designed to meet deadlines.

In this paper we propose a new transaction models which uses multiple versions of a data item. The model uses read-from graphs and dynamic reorder of transactions to meet deadlines. A read-from graph contains the past read semantics of read operations and support the model to decide which database operation to be taken. Then, we show simulation results comparing the proposed model with other transaction models such as two phase locking model and the optimistic concurrency control model.

* 고려대학교 전산과학과

1. 서 론

일반적인 데이타베이스 시스템은 트랜잭션의 처리량을 극대화시키기 위해 트랜잭션을 스케쥴링한다. 스케쥴링은 직렬 가능성에 기초한 로킹, 타임스템프, 낙관적 병행 수행, 중포 트랜잭션 처리 모델 혹은 직렬성 완화 혹은 직렬성을 배제하는 처리 모델에 기초할 수 있다[4,6].

한편, 실시간 트랜잭션 처리에서는 직렬성 및 동시성 등의 일반적인 트랜잭션 특성이외에도 예측 가능성에 기반한 시간 제약 조건을 만족해야한다는 부가적인 특성을 가지며 정확한 처리 시간내에 처리되어야한다는 요구 사항이 가장 중요시 된다. 실시간 트랜잭션의 응용 분야로는 레이더 추적, 주식 매매, 좌석 예약 시스템, 고장 및 처리 과정 자동화, 핵반응 및 화학 반응 제어 등으로 광범위하다. 특히, 공장 자동화에서 사용되는 데이타베이스 연산은 공구의 선택이나 CNC 프로그램에 대한 읽기 전용 질의와 가공이나 처리 중인 중간 조립품에 관련된 데이타에 대한 읽기/쓰기 질의와 한 단계의 가공이나 처리가 완료되었을때, 그 상태를 기록하는 질의 등으로 구분할 수 있으며[8] 특히, 가공 및 처리 과정에는 순서가 정해져 있어 시간 제약을 지키지 못하면 불량품을 생산하게 되므로 엄격한 시간 제약 조건을 만족하는 것이 바람직하다.

실시간 트랜잭션은 기존 트랜잭션에서와 같이 읽기나 쓰기와 같은 한정된 정보만을 가지고는 처리할 수 없기때문에 대부분의 경우 추가적인 의미 정보(시간 제한, 수행 시간, 우선 순위, 필요한 데이타 객체의 집합, 읽기 전용, 쓰기 전용 등)를 포함하고, 이들을 각각의 유형으로 분류하여, 유형에 맞는 동시성 제어를 한다.[5,9]

한편, 실시간 트랜잭션의 특성을 만족하기 위해 여러가지 방법이 제안되었으며 이들은 크게 CPU, 데이타 객체, I/O를 각각 고려하여 적용할 수 있다. 본 논문에서는 데이타 객체에 따른 트랜잭션 처리 방법으로서 다중 버전을 이용한 실시간 트랜잭션 모델을 제안하였으며, 시뮬레이션을 통하여 대표적인 트랜잭션 처리 모델인 2단계 로킹 모델에 비하여 로킹 발생을 제거함에 따른 실시간 트랜잭션의 시간 제약 조건의 만족율이 향상됨을 보였다.

2. 실시간 트랜잭션 모델

본 장에서는 실시간 트랜잭션 모델에 관한 기존 연구와 본 논문에서 제안한 다중 버전 실시간 트랜잭션 모델에 대해 기술한다.

2.1 기존 연구

실시간 처리를 위한 트랜잭션 처리 모델은 다양하며 그 중 대부분은 전통적인 데이타베이스 시스템에서 사용하던 로킹, 타임스템프 그리고 낙관적 병행 수행 방법을 변형하거나, 적절히 혼합하여 사용하는 것이 일반적이다.

로킹을 이용한 트랜잭션의 처리는 트랜잭션이 읽거나 쓰게 되는 데이타에 로크(lock)를 걸어 두고 다른 트랜잭션은 로크된 데이터를 접근할 수 없게 한다. 로크는 대개 읽기와 쓰기에 연산의 유형에 따라 구분하여 읽기 로크 사이에는 충돌을 발생시키지 않고 여러 트랜잭션이 한 데이타를 읽을 수 있도록 한다. 로킹을 이용한 방법에는 데드록이 발생하거나 읽기/쓰기, 쓰기/쓰기 연산 사이에 발생하는 충돌로 인한 블록킹이 발생하는 단점이 있다.

실시간 트랜잭션에서는 HP(High Priority)[3]과 같은 방법을 이용하여 항상 높은 우선순위의 트랜잭션이 로크를 가질 수 있도록 하는 방법을 사용한다. 이것은 전통적인 로킹 방법에 비해 데드록이 발생하지 않으며 높은 우선순위의 트랜잭션의 블록킹이 없어진다.

낙관적 병행 수행 제어 방법은 트랜잭션 간의 충돌을 고려하지 않고 트랜잭션을 수행하고 나서, 트랜잭션이 완료하기 전에 트랜잭션 수행 중에 충돌이 발생했는지를 조사하는 방법이다. 이 방법은 데드록과 트랜잭션의 블록킹이 발생하지 않는다. 하지만 트랜잭션의 잦은 취소가 발생할 수 있다.

이러한 문제를 해결하기 위해 실시간 트랜잭션에서는 WAIT-50과 같은 방법을 사용한다[11]. WAIT-50는 충돌이 발생하는 트랜잭션들 중에 완료하려는 트랜잭션보다 우선 순위가 높은 트랜잭션이 50 퍼센트 이상이 되면 완료를 포기하고 자신이 취소를 하며, 그렇지 않은 경우 완료를 한다.

본 논문에서 제안하는 트랜잭션 처리 모델은 다중 버전을 사용함으로써 읽기/쓰기 연산을 줄이고 쓰기/쓰기 연산 사이의 충돌을 없앴다. 또한 우선 순위의 변경이 가능한 경우, 의존 관계 그래프를 이용하여 읽기/쓰기의 충돌도 최소화하여 불필요한 트랜잭션의 취소를 줄일 수 있다. 따라서 실시간 응용 프로그램의 시간 제약 조건을 최

대한 만족시킬 수 있도록 하는 새로운 트랜잭션 처리 모델이다.

2.2 다중 버전 실시간 트랜잭션 모델

제안된 다중 버전 실시간 트랜잭션 모델은 트랜잭션이 완료하기 전까지의 모든 버전을 유지하여 충돌을 줄이고 블록킹을 최소화한다. 즉, 쓰기/쓰기 연산 사이의 충돌은 전혀 존재하지 않으며, 쓰기 연산의 경우 의존 관계 그래프(Read From Graph)를 이용하여 직렬화를 위배하게 되는 경우만 취소함으로써 높은 동시성을 유지한다. 또한, 연산의 충돌을 줄여 블록킹을 줄이며, 높은 우선 순위를 가지는 트랜잭션이 먼저 수행될 수 있도록 함으로써 실시간 트랜잭션의 시간 제약 조건을 최대한 보장한다.

2.2.1 가정

다중 버전을 이용하는 트랜잭션의 처리에서 얼마나 많은 버전을 쓸 수 있을 것인지, 그리고 어느 버전을 읽을 것인지에 관한 문제는 직렬화를 보장하기 위한 중요한 문제중 하나이다. 어떠한 방법을 택하느냐에 따라 동시성의 증가와 단계적 취소(cascading abortion)의 감소가 상대적으로 결정된다. 실시간 데이터베이스 시스템에서는 동시성을 증가시켜 전체적인 시스템 처리율을 높이는 것보다도 어떻게 트랜잭션의 시간 제약 조건을 만족하며, 또 그것을 얼마나 안정적으로 보장할 수 있는가가 중요하다. 따라서, 본 논문에서는 데이터 객체에 따라 생성될 수 있는 버전의 수를 제한하지 않는다고 가정한다. 이러한 가정은 차후 단계적 취소 문제를 야기할 수도 있으며 시스템의 자원 낭비를 가져올 수 있으나, 계속적으로 하락하는 하드웨어 가격을 감안하고, 제안하는 트랜잭션 처리 모델에서 트랜잭션이 자발적으로 취소를 요구하지 않는 한 크게 문제되지는 않을 것으로 생각한다.

트랜잭션은 각기 고정된 우선순위를 할당받는다. 그리고, 같은 순위의 트랜잭션이라고 할지라도 스케쥴되는 순서에 따라 우선순위의 차이를 갖는다. 즉, 같은 우선순위의 트랜잭션은 트랜잭션 스케쥴러에 요구되는 순서에 따라 우선순위가 부여된다. 먼저 스케쥴되는 트랜잭션이 높은 우선순위를 갖는다. 이와 같이 부여 받은 트랜잭션의 각 우선순위는 일종의 타임 스템프로 생각할 수 있다.

본 논문에서 제안하는 트랜잭션 모델에서의 CPU의 스케쥴링은 EDF(ealiest deadline first) 방법을 사용한다[3]. 또한 읽기, 쓰기 연산을 수행할 때, 다른 트랜잭션이 기록한 버전을 한 번 읽고, 자신의 버전을 최종적으로 한 번 기록한다고 가정한다.

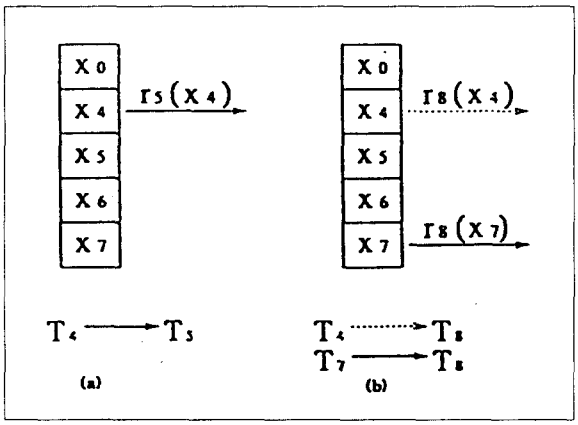
다음은 본 논문에서 트랜잭션 모델을 설명하기 위해 사용할 표기법이다.

- T_i : 우선순위 i를 가지는 트랜잭션을 나타낸다. i는 트랜잭션 식별자로 생각할 수 있다. i는 1보다 크거나 같다. 우선순위 값은 크기가 작을수록 우선순위가 높다.
- [a-z] : 알파벳 소문자의 집합은 데이터 객체를 나타낸다.
- x_i : x라는 데이터 객체 중 하나의 버전으로 T_i에 의해 기록된 버전을 나타낸다. i가 0인 경우는 이미 완료된 트랜잭션에 의해 기록된 값이다.
- A-closure(T_i) : 트랜잭션 T_i가 의존하는 트랜잭션의 집합이다.
- B-closure(T_i) : 트랜잭션 T_i에 의존하는 트랜잭션의 집합이다.
- $w_i(x)$: T_i의 쓰기 연산이다. x뿐만 아니라 그 버전인 x_i도 표시하여 사용할 수 있다.
- $r_i(x)$: T_i의 읽기 연산을 타나낸다. x뿐만 아니라 그 버전인 x_i도 표시하여 사용할 수 있다.

2.2.2 의존 관계 그래프

다중 버전을 고려할 경우 트랜잭션의 의존 관계(read-from)를 잘 따져야 한다. 직렬화 순서에 맞게 하기 위해서는 올바른 버전의 데이터 객체를 읽어야 하기 때문이다. 본 논문에서는 이를 위해 확장된 의존 관계 그래프를 제안했으며, 의존관계 그래프의 구성은 〈그림 1〉에 나타나 있다.

〈그림 1-a〉 에서 x_0는 이미 완료한 트랜잭션의 데이터 값(버전)이다. 그리고 x_4, x_5, x_6, x_7은 각각 T_4, T_5, T_6, T_7에 의해 기록된 x의 버전이다. 화살표는 T_5가 x_4에서 읽기 연산을 수행했다는 것을 의미한다. 이것을 의존관계 그래프를 이용하면 $T_4 \rightarrow T_5$와 같이 나타낼 수 있다. 각 데이터 객체에 대해 읽기 연산을 수행할 경우 완료하지 않은 트랜잭션 사이에는 계속적으로 의존 관계를 나타내는 링크가 생성된다. 〈그림 1-b〉에서 x_4에서 T_8이 읽기 연산을

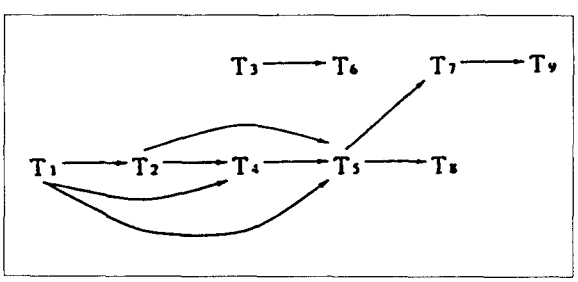


〈그림 1〉 의존관계 그래프의 구성

수행하는 것은 허용하지 않는다(점선은 불가능한 링크를 의미한다.). 즉 트랜잭션 자신의 우선순위와 같거나 자신의 우선 순위보다 한단계 바로 높은 버전만을 읽도록 한다. 따라서, T_8은 x_7만을 읽을 수 있는 것이다. 이렇게 자신 보다 바로 높은 우선순위의 버전만을 읽도록 하면 나중에 취소 비용을 줄일 수 있도록 하며, 직렬화를 보장한다.

트랜잭션은 읽기 연산을 수행할 때마다 자신이 읽은 데이터 객체의 트랜잭션 식별자를 의존 관계 그래프에 추가한다. 이 때, 의존 관계 그래프에 읽기 연산을 수행하는 트랜잭션이 의존하는 트랜잭션 중에 데이터 객체를 기록한 트랜잭션 식별자가 이미 존재하고 있다면 추가할 필요가 없다.

〈그림 2〉는 이와 같은 과정을 거쳐서 만들어진 트랜잭

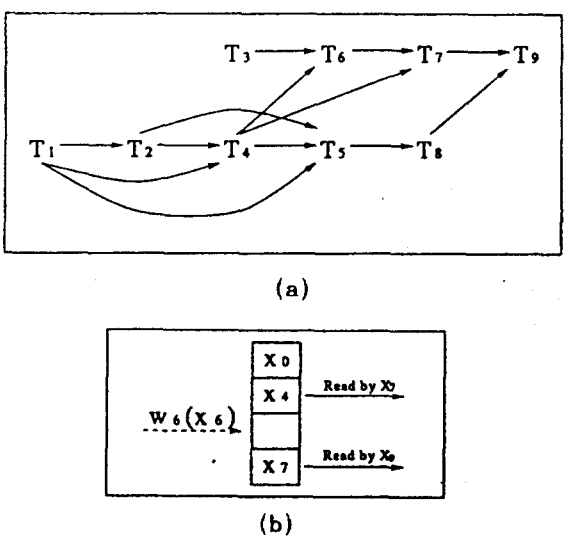


〈그림 2〉 구성된 2개의 의존관계 그래프

선의 의존 관계 그래프들이다. 이 그래프는 새로운 의존 관계가 발생할 때마다 새로운 링크가 생기게 되며, 데이터베이스 시스템 내부에는 이러한 그래프가 여러개 존재할 수 있다. 이것은 트랜잭션의 특정 그룹 사이에는 의존 관계가 있으나, 다른 트랜잭션들과는 의존 관계가 없는 트랜잭션의 그룹들이 존재할 수 있기 때문이다. 의존 관계 그래프에서 또 하나 볼 수 있는 특징은 DAG(directed acyclic graph)의 모양을 갖추고 있다는 것이다. 이것으로 미루어 의존 관계 사이에는 데드록이 없음을 알 수 있다.

2.2.3 제안된 트랜잭션 모델

트랜잭션이 읽기 연산을 수행할 때마다, 블록킹되지 않고 자신보다 바로 높은 우선순위(높은 순위의 버전이 없을 경우 완료된 트랜잭션의 버전)의 버전 값을 읽게 되며, 그 순간마다 의존 관계가 없던 트랜잭션 사이에는 의존 관계 그래프에서 링크를 생성하게 된다. (읽기 전용 연산인 경우 완료된 트랜잭션의 데이터 값만을 읽도록 하여 취소를 줄일 수도 있다.) 이 때, 트랜잭션의 기록이 트랜잭션의 우선 순위대로 수행이 된다면 아무런 문제가 발생하지 않는다. 다시 말해, $w_i(x)$, $w_j(x)$의 두 연산 사이에서 $i < j$의 관계가 있을 경우, 항상 $w_i(x) < w_j(x)$의 순서로 연산이 수행된다면 아무런 문제도 발생하지 않는다. 하지만, 특정 쓰기 연산의 경우 그 유효성을 잃을 수 있다. 예를 들어 〈그림 3〉에서와 같은 문제가 발생한다.



〈그림 3〉 쓰기 연산이 수행된 경우

〈그림 3-a〉의 의존 관계를 가지고 있는 데이터 객체(버

전)에 T_6이 〈그림 3-b〉와 같이 쓰기 연산을 수행할 경우 (반실선은 쓰기 연산을 나타내기 위한 것이다.), 다른 트랜잭션이 이미 T_5에서 읽기 연산을 수행하고 난 후이므로 삽입할 경우 단일 버전 직렬화가 유효성을 잃을 수 있게 된다. 단일 버전 직렬화를 만족하려면, 여러 버전이 있을 때, 읽기 연산은 반드시 바로 앞의 우선순위(타임스템프)의 버전을 읽어야 한다. 이것을 〈그림 3-b〉에서 살펴보면, x_6이 존재하고 있는데 T_7가 x_4에서 읽는 것과 같은 형태가 되어 단일 버전 직렬화를 잃게 된다. 트랜잭션은 쓰기 연산을 할 때 항상 이러한 현상이 발생하지 않도록 해야 한다. 즉, 다음 조건을 만족해야 한다. - $w_j(x_i)$, $i < k < j$일 때, x_k는 존재하지 않는다. - 따라서, 한 트랜잭션이 쓰기 연산을 수행할 수 있는 경우는 자신보다 우선순위가 적은 (타임스템프가 큰) 다른 트랜잭션이 데이터 객체에서 바로 상위의 우선순위를 가진 버전을 읽지 않은 경우에 가능하다. 하지만, 실제 시스템에서 각 버전마다 어느 트랜잭션이 읽었는지 기록하는 것은 비효율적일 것으로 예측된다. 따라서 어느 정도 의미 정보의 상실을 초래하더라도 읽기 연산이 수행될 때 플래그만을 설정한다. 트랜잭션은 기록해야 할 버전의 바로 상위 우선 순위의 버전에 플래그가 설정되어 있지 않다면 즉시 기록을 수행한다.

그렇지만 이 조건은 개선의 여지가 있다. T_7과 같이 다른 트랜잭션이 이미 읽기 연산을 수행하고 난 후에도 한 가지 조건을 더 이용하면 동시성을 증가시킬 수 있다. 그것은 앞서 정의한 B-closure(T_i)를 이용하여 트랜잭션들의 의존 관계를 고려하는 경우이다. 〈그림 3-a〉에서 B-closure (T_6)은 T_6에 이르는 의존 관계 DAG의 모든 노드를 말한다. 따라서 B-colusre(T_6)은 {T_1, T_2, T_3, T_4}이다. T_6가 읽은 버전들의 우선순위가 모두 4(즉 T_4)이상일 경우, T_6의 우선순위를 증가하여서 쓰기 연산을 수행할 수 있다. 즉 〈그림 3〉에서와 같이 T_6가 x에 대한 쓰기 연산을 수행하려는 경우, T_6를 현재 충돌을 일으키는 x_4의 우선순위 4보다는 큰 적절한 값을 선택하여, 예를 들어 $T_{3.5}$와 같이 바꾸어 주면 된다. 이것은 타임스템프를 바꾸어 준다고 볼 수도 있으며, 트랜잭션의 수행 순서를 바꾸어 주는 것과 같다. T_6에 의존하는 트랜잭션들도(다시 말해, A-closure (T_6)={T_7, T_9}) 똑같은 규칙을 적용하여 트랜잭션의 우선순위를 변경할 수 있다. 이 때 주의할 것은 의존 A-closure 관계에 있는 트랜잭션들은 T_6와 T_6보다 높은 순위, 즉 타임스템프가 작은 것을 읽은 경우에만 가능하며 그렇지 않은 경우, 그 트랜잭션과 그에 의존하는 트랜잭션들을 취소해야 한다. 그리고, 우선순위를 3.55와 같이 T_6가 바뀐 우선순위인 3.5보다 높게 해 주어야 한다.

이러한 방법은 트랜잭션의 선후 관계(precedence)가 정해지지 않는 경우와 트랜잭션의 취소에 따른 비용이 트랜잭션 처리 순서, 즉 타임스템프를 임의로 바꾸는 것보다 클 경우에 유리하다. 하지만 우선순위가 고정되어 있어 변경이 불가능한 경우에는 이러한 방법을 사용할 수 없다.

트랜잭션의 우선순위를 변경할 수 없을 때, 충돌을 일으키는 트랜잭션은 취소되어야 한다.

트랜잭션을 취소하는 경우에는 A-closure(T_i)를 이용하면 편리하다. A-closure(T_i)는 트랜잭션 T_i에 의존하는 모든 트랜잭션의 집합으로 T_i에서 나가는 모든 경로의 트랜잭션이 된다. 트랜잭션 T_i을 취소할 때 A-closure(T_i)를 함께 취소하여야 한다. A-closure(T_i)는 T_i에 의존하는 모든 트랜잭션이기 때문이다. 〈그림 3-a〉에서 A-closure(T_6)는 {T_7, T_9}이다. 따라서 T_6가 취소될 때, T_7, T_9도 취소해야 한다.

트랜잭션의 완료는 부분완료(precommit)와 완료(commit)로 구분하며, 트랜잭션이 수행을 완료하였으나 그 트랜잭션이 읽은 트랜잭션들이 완료하지 않은 경우에는 부분완료를 하고, 그렇지 않을 경우에 완료하도록 한다.

여기서 제안한 트랜잭션 모델은 단일 버전 직렬화 조건을 만족하며, 데드록이 발생하지 않는다. 이것은 다음 2가지 정리로 증명할 수 있다.

〈정리 1〉 의존 관계를 이용한 다중 버전 트랜잭션 처리는 단일 버전 직렬화 조건을 만족한다.

[증 명] 모든 데이터 객체에 대해 $r_j(x_i)$일 때, x_k는 존재하지 않는다. 이 때, $i < k < j$이며, T_i, T_j, T_k는 같은 의존 관계 그래프에 속하는 트랜잭션이다.

〈정리 2〉 의존 관계를 이용한 다중 버전 트랜잭션의 처리는 데드록이 발생하지 않는다.

[증 명] 높은 우선순위(낮은 타임스템프)의 트랜잭션이 먼저 수행되고 낮은 우선순위의 트랜잭션은 취소된다. 따라서 대기 상태가 있을 수 없으며, 대기 그래프에서 사이클은 발생할 수 없다.

3. 제안된 트랜잭션 모델의 시뮬레이션 모델링

본 장에서는 모델을 시뮬레이션하는데 필요한 가정들과 시뮬레이션 모델, 그리고 시뮬레이션 입력 매개 변수를 제시하였다. 또한, 모델의 비교 평가를 위한 기준이 되는 성능 지수를 설명하였다. 본 연구에서 시뮬레이션 프로그램은 SLAM Ⅱ 시뮬레이션 언어와 포트란 언어를 사용하여 구현하였으며 SunOS release 4.1.2로 운영되는 SPARC station Ⅱ에서 시뮬레이션을 수행하였다.

3.1 시뮬레이션 가정

일반적으로 실시간 트랜잭션 모델의 대부분은 2PL이나 낙관적 병행 수행 모델의 변형이 많다. 본 논문에서는 이미 널리 알려진 HP를 이용한 2PL 모델과 WAIT-50의 낙관적 모델을 본 논문에서 제안한 다중 버전 실시간 모델과 비교한다. 시뮬레이션하는데 있어 공통적으로 적용된 가정은 다음과 같다.

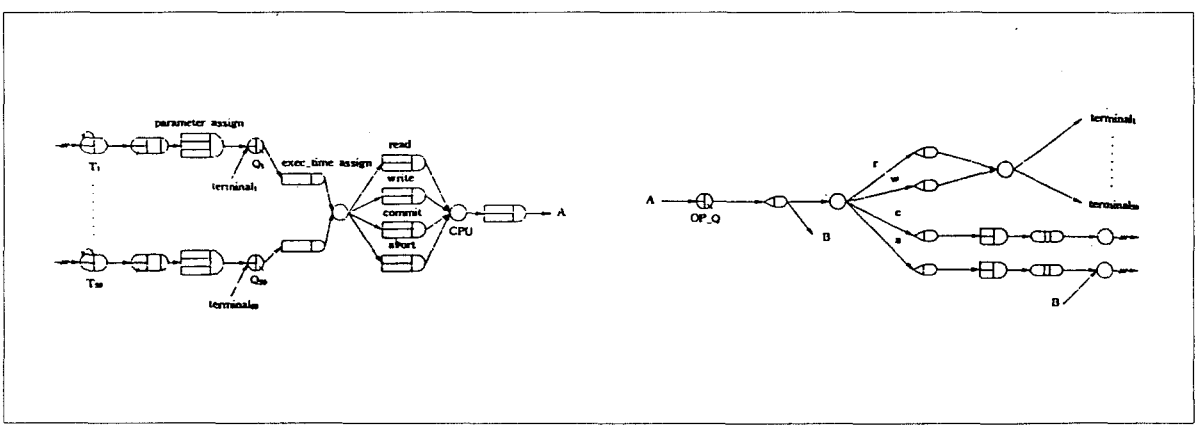
- 트랜잭션 오류 및 시스템 고장은 발생하지 않는다고 가정하였다.
- 2단계 로킹 모델을 사용할 경우, 교착 상태 방지를 위해 일반적인 time out 기법이 적용된다고 가정하였다.
- 액세스하는 데이타 항목은 주기억 장치에 항상 존재한다고 가정하였다[2]. 따라서, 디스크 액세스에 관련된 비용은 고려하지 않았다. 이는 시뮬레이션 하고자 하는 트랜잭션 모델들의 특성상 보조 기억 장치에 대한 액세스 비용이 무관함에 따른것으로 비교하는 모델 모두에 공통적으로 적용된다.
- 본 시뮬레이션은 분산이 아닌 중앙 집중형 데이타베이스 시스템에서의 트랜잭션이 처리되는 것으로 가정한다.

3.2 시뮬레이션 모델

본 절에서는 시뮬레이션을 위한 논리적 모델을 제시한다. 본 연구에서는 〈그림 4〉와 같이 SLAM Ⅱ 네트워크 모델을 사용하여 논리적 모델을 설계하였다[1,5].

〈그림 4〉에서 50개의 CREATE 노드가 주어진 시간 간격에 따라 트랜잭션을 생성시키고 있음을 알 수 있다. 이 모델에서 트랜잭션들은 SLAM Ⅱ 모델의 ENTITY들로 표현된다. 각 CREATE 노드에서 생성된 트랜잭션들은 대기 용량이 1인 RESOURCE를 위한 AWAIT 노드로 이동한다. 이는 특정 터미널에서 오직 하나의 트랜잭션만 생성될 수 있도록 하며, 트랜잭션은 RESOURCE를 할당받은 뒤 실행 시간, 우선 순위, 시간 제약 등의 속성들을 부여받는다.

일단 적절한 속성들이 부여되면 임의의 GOON 노드로 이동하는데 그곳에서 각각의 확률을 갖는 4개의 분기를



〈그림 4〉SLAM Ⅱ 네트워크 모델

따라 분기하게 되며 이것은 뒤따르는 ASSIGN 노드에 의해 트랜잭션의 읽기, 쓰기, 취소, 완료 등의 연산 형태를 부여받게 된다. 이후에 트랜잭션은 데이타베이스가 가용한 상태가 될때까지 연산 큐에서 대기하게 되며 데이타베이스 시스템이 준비 상태가 되면 연산 큐로부터 하나의 연산을 얻는다.

데이타베이스 시스템은 연산 형태에 따라 트랜잭션을 처리하며 4개의 연산 형태가 있기때문에 각 연산에 대한 처리 과정을 시뮬레이션하기 위해 4개의 분기가 사용되었다.

읽기 그리고 쓰기 연산들은 2단계 로킹 모델에서의 로킹 테이블, 낙관적 모델의 읽기, 쓰기 집합 혹은 다중 버전 트랜잭션 모델에서의 의존 관계 그래프의 수정에 영향을 미친다.

읽기/쓰기 연산들은 시스템내에서 충돌하는 다른 트랜잭션들을 취소시킬수 있으며, 성공적으로 수행될 경우엔 터미널을 의미하는 대기 용량 1을 갖는 QUEUE 노드로 되돌아간다.

터미널에서 약간의 지연후에 트랜잭션들은 다른 연산 형태를 가지고 연산큐로 이동하며 이때, 완료나 취소 연산 경우에는 ENTITY를 종료하고 그 트랜잭션이 시작될 때부터 보유하던 RESOURCE를 풀어줌으로써 터미널에서 다른 트랜잭션을 생성할 수 있도록 한다.

위 모델에서 각 연산들은 SLAM Ⅱ의 EVENT 노드에서 수행되며 각 EVENT 노드들은 데이타베이스 시스템의 상태를 기록하기 위해 SLAM Ⅱ와 FORTRAN 모델들과 밀접하게 관련이 된다.

본 논문에서는 2단계 로킹 모델, 낙관적 동시성 제어 모델 그리고 다중-버전 트랜잭션 모델들을 FORTRAN을 이용하여 시뮬레이션 모델로 표현하였다.

2단계 로킹 모델은 접근하는 데이타에 로크을 설정하며 트랜잭션이 완료되거나 취소될때 다른 트랜잭션의 읽기 혹은 쓰기를 위해 로크을 해제한다. 따라서, 2단계 로킹 모델은 읽기 로크 모드 및 쓰기 로크 모드의 2가지 로크 모드를 갖는다.

낙관적 동시성 제어 모델에서는 모든 트랜잭션들이 불일치성 검사없이 처리되며 특정 트랜잭션이 완료되려는 순간 타 트랜잭션과의 충돌 여부를 검사하여 충돌이 없으면 성공적으로 종료되고 그렇지않을 경우엔 충돌하는 트랜잭션들을 취소시킨다.

3.3 입력 매개 변수 및 성능 지수

다양한 시뮬레이션 환경을 만들기 위해 시뮬레이터의 입력으로 들어가는 매개변수를 살펴 보면 다음과 같다.

- tr—gen—rate : 트랜잭션 생성률
- w—ratio : 쓰기 연산 비율
- n—of—term : 터미널들의 수
- tr—ex—time : 트랜잭션 실행 시간
- slack : 제약 시간-트랜잭션 실행 시간
- n—of—data : 데이타 수
- tr—delay : 각 터미널에서의 트랜잭션 지연시간
- n—of—tr : 트랜잭션들의 수

〈표 1〉 입력 매개 변수

input parameter	scope
tr—gen—rate	1/4 - 30 trans/msec
w—ratio	50 % (0.5)
n—of—term	50 terminals
tr—ex—time	400 - 3000 msec
slack	5 - 100 msec
n—of—data	100 data
tr—delay	0 - 200 msec
n—of—tr	500 transactions

입력 매개 변수의 값은 대상 시스템에 따라서 달라지고 시뮬레이션 결과에 직접적으로 영향을 미친다. 본 연구에서는 다음과 같은 입력 매개 변수값을 설정하였다. 데이타의 수와 트랜잭션 발생 빈도등을 충돌이 잘 발생할 수 있도록 했으며, 따라서 각 모델이 충돌이 많은 상황에서 어떤 결과를 나타내는지 알 수 있다. slack값의 크기도 비교적 작게하여 블록킹이 적은 알고리즘을 더 쉽게 확인할 수 있도록 했다. 확률 분포는 일양분포를 사용하였다.

또한, 시뮬레이션에서의 각 모델에 대한 성능 평가 지수는 다음과 같다.

- n—tr—d : 트랜잭션의 시간제약조건 만족율
  = (제한시간내에 완료된 트랜잭션 수)
  / (실행 트랜잭션의 수)

일반적으로 데이타베이스 트랜잭션 모델에 대한 성능 평가 지수로써 트랜잭션의 응답 시간이나 시스템의 트랜잭션 처리 성공률 등을 사용한다. 본 연구에서는 실시간 트랜잭션의 가장 중요한 평가 지수로서 트랜잭션의 시간 제약 조건 만족률을 고려하여 평가하였다.

4. 시뮬레이션 결과 및 성능 분석

본 장에서는 2단계 동시성 제어 모델과 본 논문에서 제안한 다중 버전 실시간 동시성 제어 모델의 시뮬레이션 결과를 분석함으로써 두 모델의 성능을 비교 평가한다. 본 실험에서는 제 3.3절에서 기술한 입력 매개 변수들 중 쓰기 연산 비율, 시간 제약 조건, 데이타의 수를 〈표 1〉에 제시한 값에 따라 변환시켰을때의 각 트랜잭션들의 제약 시간 만족율을 관찰한다.

그리고, 편의상 우선 순위를 고려한 2단계 로킹 모델은 2PL로, 우선 순위를 고려한 낙관적 동시성 제어 모델은 OCC로, 본 논문에서 제안한 다중 버전 실시간 모델은 MV로 약칭한다.

시뮬레이션 결과에 대한 분석에서 가장 중요한 평가 기준은 제약 시간안에 완료된 트랜잭션들의 수로써 이는 제약 시간내에 완료되지 못한 트랜잭션은 공장 자동화 환경에서 무의미하기 때문이다.

4.1 쓰기 연산 비율에 따른 제약 시간 만족율

이 시뮬레이션에서는 취소 연산을 제외한 읽기, 쓰기, 완료 연산만을 사용하여 읽기, 쓰기 연산 사이의 비율 관계만을 살펴보았다. 3가지 모델 모두 쓰기 연산의 비율이 높아질수록 즉, 쓰기 연산이 빈번하게 발생할 수록 시간 제약 만족률이 전체적으로 감소함을 볼 수 있다. MV의 경우 읽기/읽기, 쓰기/쓰기 연산 사이의 충돌이 존재하지 않으므로 그래프의 양 끝이 거의 100%의 제약 시간 만족률을 보이는 것을 알 수 있다. 동일한 쓰기 연산 비율에서는 쓰기 로크로 인해 충돌 발생률이 가장 높은 2PL 모델이 가장 낮고 본 논문에서 제안한 MV 모델이 가장 높게 제약 시간을 만족할 수 있음을 보여준다. 이는 2PL 모델이 트랜잭션의 실행 중 갱신하려고 하는 데이타 항목에 대해 배타적 로크를 설정함에 의해서 로크가 설정된 데이타 항목을 액세스하려는 다른 트랜잭션들이 블록킹됨에 따라 평균 트랜잭션의 완료 시간이 길어지기 때문이다. 이에 반해 다중 버전을 이용한 MV 모델은 블록킹을 제거함에 의해 평균 트랜잭션 응답 시간이 짧아지기때문에 제약 시간을 만족하는 트랜잭션의 갯수가 많음을 알 수 있다.

4.2 slack 간격에 따른 제약 시간 만족율

2PL, OCC, MV 모델 모두가 deadline이 증가함에 따라 트랜잭션의 시간 제약 만족율이 증가함을 알 수 있다. 이것은 블록킹이 되는 모델의 경우라도 deadline이 커질수록 여유 시간인 slack이 커지게되어 시간 제약 만족율이 증가하게 되는 것이다. MV 모델이 다른 두 모델보다 뛰어난 성능을 보인다는 것을 알 수 있다.

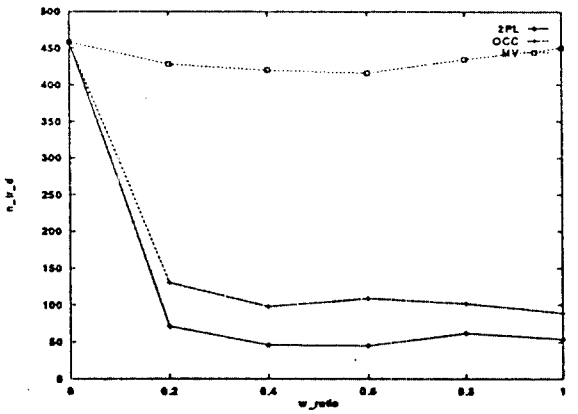
4.3 데이타 수에 따른 제약 시간 만족율

〈그림 5-c〉를 통하여 2PL 모델이 실시간 환경에 가장 부적합함을 알 수 있으며 이는 로크에 따른 블록킹이 충돌이 발생한 트랜잭션들이 제약 시간을 만족할 수 없도록 하기 때문이다. OOC 모델은 2PL 모델보다는 제약 시간 만족율이 높으나 충돌이 빈번하게 발생하는 환경에서는 마찬가지로 만족율이 낮음을 볼 수 있다.

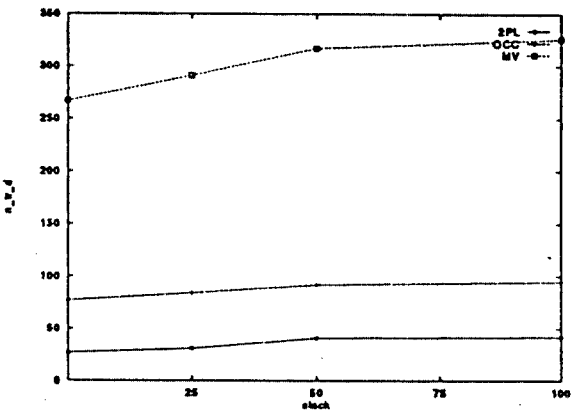
결론적으로 전체적으로 실시간 트랜잭션의 가장 중요한 고려 사항인 각 트랜잭션의 제약 시간 만족율을 살펴보면 제안한 모델 MV이 3 종류의 시뮬레이션한 모델중에서 제약 시간 만족율이 가장 높으며 이는 제안한 모델이 블록킹을 허용하지않고, 트랜잭션의 순서를 재조정하기 때문이다. 따라서, 전체적으로 2PL 모델보다는 OCC 모델이, OCC 모델보다는 MV 모델이 실시간 트랜잭션 모델에 적합하다는 결론을 내릴 수 있다.

5. 결 론

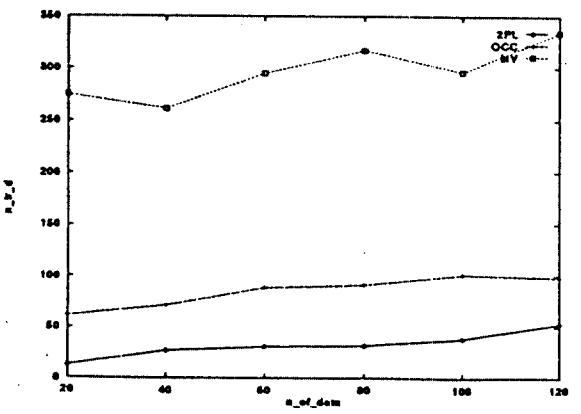
본 논문에서는 공장 자동화 데이타베이스 시스템의 트랜잭션 처리 요구 사항중 실시간성을 만족시키기위해 시간 제약 조건을 우선적으로 만족시키는 다중 버전 실시간 트랜잭션 모델을 제안하였다. 그리고, 이에 대한 성능 평가로써 시뮬레이션 모델을 구성하였으며, 대표적인 트랜잭션 모델인 로킹에 기반한 H2PL 그리고, OCC 모델과의



〈그림 5-a〉 write 연산비율에 따른 제약시간 만족율



〈그림 5-b〉 slack에 따른 제약시간 만족율



〈그림 5-c〉 데이타 수에 따른 제약시간 만족율

시뮬레이션을 통한 성능을 비교하여 제시하였다.

결과적으로 본 논문에서 제안한 다중 버전 실시간 모델은 다중 버전을 이용하여 동시성을 증가하였으며, 블로킹을 최소화하여 실시간 트랜잭션의 예측 가능성을 높였다. 제안된 모델은 읽기 연산의 블로킹이 전혀 발생하지 않으며, 쓰기 연산의 경우에도 블록킹을 최소화하여 2PL, OCC에 비해 동시성과 시간 제약 만족성이 향상됨을 시뮬레이션 결과를 통해 알 수 있다.

추후 연구과제로는 제안한 다중 버전 실시간 트랜잭션 모델을 다중 버전에 기반한 다른 실시간 트랜잭션 모델을 포함한 보다 많은 실시간 트랜잭션 모델들과의 성능 비교가 이루어져야할 것으로 생각된다.

참고문헌

[1] A. Alan B. Pritsker, Introduction to Simulation and SLAM Ⅱ, 3rd edition, Wiley, 1986

[2] Abbott, Robert and Hector Garcia-Molina, "Scheduling Real-time Transactions," ACM SIGMOD Record, 1988.

[3] Abbott, Robert and Hector Garcia-Molina, "Scheduling Real-Time Transactions: A Performance Evaluation," ACM TODS, 1992.

[4] Bernstein, P.A. and Hadzilacos, V. and Goodman, N., "Concurrency Control and Recovery in Database Systems," Addison-Wesley, 1987.

[5] B. Pritsker, SLAM Ⅱ Quick Reference Manual, Pritsker Corporation, 1990

[6] Gray, Jim and Reuter Andeas, Transaction Processing: Concetps and Techniques, Morgan Kaufmann, 1993.

[7] Huang, Jaindong and Stankovic, J.A., "Experimental Evaluation of Real-Time Optimistic Concurrency Control Systems," VLDB, 1991.

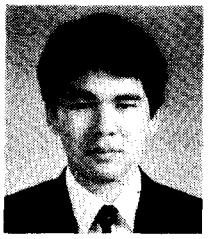
[8] Ranky, G. Paul, Manufacturing Database Management and Knowledge Based Expert Systems, CIMware Limited, 1990.

[9] Son, Sang Hyuk and Kim, Young-Kuk, "Predictability and Consistency in Real-Time Database Systems," Porceedings of InfoScience, 1993.

[10] Son, S.H. and Park, S, "Scheduling and Concurrency Control for Real-Time Database Systems," ADSFAA,

1993. Transactions," ACM SIGMOD Record, 1988.
[11] Stankovic, J.A. and Wei Zhao, "On Real-Time

● 저자소개 ●



유인관
1992년 고려대학교 철학과 졸업
1992년~현재 고려대학교 전산과학과(석사과정 재학중)
관심분야: 데이타베이스, 트랜잭션 처리 기법, 시뮬레이션

박성진
1991년 고려대학교 전산학과 졸업
1993년 고려대학교 전산과학과(석사학위 취득)
1993년~현재 고려대학교 전산과학과(박사과정 재학중)
관심분야: 분산데이타베이스, 모델링 및 시뮬레이션, 공학 데이타베이스

백두권
1974년 고려대학교 수학과 졸업
1977년 고려대학교 산업공학과(석사학위 취득)
1977년~1980년 대전초급대학 및 전남대학교 공과대학 전임강사
1983년 Wayne State Univ. Computer Science 석사
1986년 Wayne State Univ. Computer Science 박사
1986년~현재 고려대학교 전산과학과 교수
관심분야: 지식베이스, 모델링과 시뮬레이션, 분산 시스템, 소프트웨어 공학, 데이타베이스 등