소프트 실시간 데이타베이스 시스템에서 이중 록킹을 이용한 트랜잭션 스케쥴링 기법

최 의 인† · 고 병 오††

요 약

컴퓨터의 응용 영역이 확대됨에 따라 일정 시간 이내에 자료를 신속히 처리해야하는 실시간 응용들이 대두되기 시작했다. 기존의 디스크 기반 데이타베이스 시스템은 디스크 입출력으로 인한 성능 저하로 마감시간을 지닌 실시간 트랜잭션 처리에는 부적당하다. 또한, 최근 마감시간 우선 기법을 사용하는 실시간 트랜잭션 스케쥴링 기법은 트랜잭션이 과다하게 적재되는 경우 연쇄적인 마감시간 초과로 인하여 성능이 저하된다.

이러한 문제점을 해결하기 위해 이 논문에서는 첫째, 주기억 데이터베이스 환경하에서 우선순위를 기반으로 한 이중 록킹 기법을 제안하여 실시간 트랜잭션 스케쥴링 기법의 성능을 향상시켰다. 그리고, 본 연구에서 제안한 우선순위를 기반으로 한 이중 록킹 기법의 성능을 평가하기 위해 모델링하고, 주기억 데이터베이스 환경하에서 이산적 사건 모델을 지원하는 SLAM II로 시뮬레이션하여 타 연구와의 성능을 비교 분석하였다.

Transaction Scheduling Technique Using Double Locking in a Soft Real-Time Database System

Eui In Choi† · Byoung Oh Koh††

ABSTRACT

As the areas of computer application are expanded, the real-time application environments that must process as many transactions as possible within their deadlines have been increased recently. Conventional disk based database system is not appropriate in real-time transaction processing due to delaying time for disk I/O processing. When the system is overloaded, the performance of transaction scheduling technique using earliest deadline first deteriorates rapidly because it can assign the highest priority to a transaction that has already missed or is about to miss its deadline.

Therefore, the performance of suggested transaction secheduling technique is made to be improved by proposing the double locking mechanism based on priority. Finally, in order to evaluate the performance of the proposed priority-based double locking techniques under single processor and main memory database system environments, the simulation model was developed using the SLAM II language.

※이 논문은 1996년도 한남대학교 학술연구 조성비 지원에 의하여 연구되었음.

† 정 회 원:한남대 컴퓨터공학과
†† 비 회 원:세명대 정보처리학과

논문접수:1996년 8월 29일, 심사완료:1997년 1월 15일

1. 서 론

실시간 시스템은 트랜잭션의 시간적 제약조건인 마감시간(deadline)을 반드시 보장하여야 결과에 대한 가치가 의미있는 하드 실시간 시스템(hard real-time system)과 트랜잭션의 마감시간이 초과하여도 가치는 떨어지지만 취소되기를 원하지 않는 소프트 실시간 시스템(soft real-time system)이 존재한다. 이러한 실시간 시스템에서 대량의 데이타를 취급하기 위하여 구축한 실시간 데이타베이스 시스템(Real-Time DataBase System;RTDBS)은 트랜잭션에 의하여 공유된 데이타의 일관성을 유지하면서 마감시간을 만족하는 트랜잭션의 수가 많도록 트랜잭션을 스케줄링하여야 한다[2, 6, 11]. 즉, 실시간 데이타베이스 시스템에서 트랜잭션 처리는 평균 응답 시간을 단축하는 빠른 계산보다는 규정된 시간 이내에 종료하여 트랜잭션의 마감시간을 만족시키는 것이다. 이런 트랜잭션들을 기존의 디스크 기반 데이타베이스 시스템에서 처리할 경우에 첫째, 사용자에게 서비스 제공시 수시로 디스크 입출력을 수행하여 많은 지연 시간을 초래하기 때문에 마감시간이내에 트랜잭션의 마감시간을 대부분 만족시킬 수 없다. 둘째, 각 트랜잭션이 접근하려는 데이타에 대해 일관성을 보존해야 하는 일관성 제약조건은 트랜잭션을 어느 시점에서 블록시키거나 교착상태를 발생시키므로써 트랜잭션의 예상 수행 시간을 예측할 수 없어 마감시간을 만족하기가 어렵다.

이러한 문제점을 해결하여 트랜잭션의 마감시간을 만족하는 트랜잭션의 수가 가능한 많도록 기존의 기법을 개선하는 것이 필요하다. 첫째, 트랜잭션 스케줄링 기법에 의하여 스케줄링 되더라도 트랜잭션이 어느 시점에서 블록되거나 교착상태가 발생되면 트랜잭션들의 예상 수행 시간을 예측하기가 어려워 마감시간을 만족하기가 어렵다. 그러므로 마감시간을 고려하여 트랜잭션을 제어하는 동시성 제어 기법이 필요하다. 둘째, 기존의 디스크 데이타베이스 시스템에서 실시간 시스템이 요구하는 트랜잭션을 처리 할 경우 많은 디스크 입출력으로 지연 시간이 발생하기 때문에 트랜잭션의 마감시간을 보장하기가 어렵다. 그러므로 이를 개선한 데이타베이스 시스템이 필요하다. 또한 트랜잭션 처리시 마감시간 이내에 처리되는 트랜잭션의 수가 많도록 하는 실시간 데이타베이스 시스템을 설계하는 방법은 디스크 접근 지연 같은 병목현상을 해결 할 수 있도록 기존의 데이타베이스 시스템을 다시 설계하거나 기능을 축소시키는 방법이다[4, 7, 9, 10].

2. 관련 연구

2.1 우선순위 할당 정책

우선순위 할당 정책은 실시간 트랜잭션 스케줄링에서 트랜잭션의 마감시간을 고려하여 우선순위를 산출하고 이를 트랜잭션에 할당한다. 이러한 우선순위 할당 정책에는 FCFS(First Come Frist Service) 기법, LS(least Slack) 기법, EDF 기법 등이 있다[1, 5].

가장 먼저 도착한 트랜잭션에 가장 높은 우선순위를 할당해 주는 FCFS 기법은 마감시간 정보를 고려하지 않아 긴급한 마감시간을 지닌 트랜잭션보다는 긴급하지 않은 마감시간을 가지고 먼저 들어온 트랜잭션이 우선적으로 종료된다. 따라서, 마감시간을 고려하지 않았기 때문에 FCFS 기법은 실시간 처리 시스템에서는 부적합하다는 단점이 있다. 여유 시간(slack time)이 가장 작은 트랜잭션에 높은 우선순위를 할당하는 LS 기법은 트랜잭션들 간에 충돌이 발생하는 경우에 고 우선순위를 지닌 트랜잭션의 여유시간 이내에 저 우선순위를 지닌 트랜잭션을 완료할 수 있다면 저 우선순위를 지닌 트랜잭션을 취소시키지 않고 실행함으로써 자원의 낭비를 줄일 수 있고, 필요할 때 마다 트랜잭션에 대한 여유시간을 계산할 수 있으므로 스케줄링의 예측이 가능하다는 장점이 있다. 반면에, 여유시간 계산에 따른 오버헤드가 발생한다는 단점이 있다. 트랜잭션의 마감시간이 가까운 트랜잭션에 고 우선순위(high priority)를 할당하는 EDF 기법은 적정 수준에서는 성능이 양호하다는 장점이 있는 반면, 마감시간을 막 초과하려는 트랜잭션과 마감시간에 근접한 트랜잭션에 고 우선순위를 할당하기 때문에 연쇄적인 마감시간 초과가 발생되어 성능이 저하된다는 단점이 있다[1, 8].

본 논문에서는 이런 문제점 해결을 위해 트랜잭션 분류, 적중 큐, 초과 큐 마감시간 초과 예측, 짧은 트랜잭션 우선 기법 등 새로운 개념을 도입한 트랜잭션 스케줄링 기법을 제안하였다.

2.2 동시성 제어 정책

트랜잭션들이 우선순위 역행이나 교착상태가 발생하면 트랜잭션의 마감시간을 만족할 수 없을 뿐만 아니라 처리율도 크게 저하된다. 따라서, 트랜잭션의 마감시간을 고려하여 충돌하는 트랜잭션들을 동적으로 제어함으로써 시스템에 도착하는 트랜잭션 수와 마감시간 이내에 처리되는 트랜잭션 수의 비인 적중률을 향상시키도록 하는 동시성 제어 기법이 반드시 필요하다.

충돌이 발생하는 경우 저 우선순위를 지닌 트랜잭션은 취소시키고 고 우선순위를 지닌 트랜잭션을 실행시키는 고 우선순위(high priority; HP) 기법은 구현이 용이하고 우선순위 역행을 해결할 수 있다는 장점이 있는 반면에, 취소되는 트랜잭션이 많아 자원이 낭비된다. 요구한 트랜잭션이 점유한 트랜잭션보다 우선순위가 높을 때 요구한 트랜잭션의 우선순위를 점유한 트랜잭션에 상속시키는 우선순위 상속(priority inheritance; PI) 기법은 우선순위 역행을 해결할 수 있다는 장점은 있으나 교착 상태가 발생된다는 단점이 있다[Sha88]. 우선순위 상속 기법의 확장 개념인 우선순위 상한(priority ceiling; PC) 기법은 록된 데이타 항목들 중 가장 높은 데이타 항목의 우선순위 상한 값보다 더 높은 우선순위를 지닌 트랜잭션에게만 록을 허락함으로써 교착상태의 문제점을 해결할 수 있다는 장점이 있는 반면에 다른 데이타 항목에 종속적으로 스케쥴링되기 때문에 대부분의 트랜잭션이 블록된다는 단점이 있다[8, 14, 15].

이 논문에서는 구현이 용이하고 기존의 데이타베이스 시스템에서 사용하는 2단계 록킹 기법에 고 우선순위, 우선순위 상속, 그리고 우선순위 상한 기법의 문제점을 개선한 우선순위를 기반으로 한 이중 록킹 기법을 제안한다.

3. 개선된 동시성 제어 기법

3.1 동시성 제어 기법

트랜잭션 스케쥴링 기법의 목적은 마감시간을 만족하는 트랜잭션의 수가 가능한 한 많도록 하는 것이다. 따라서, 시간적 개념을 무시한 기존의 동시성 제어 기법들은 2 단계 록킹 기법이나 낙관적인 동시성 기법 등을 중심으로 연구가 진행되어왔으나 실시간 데이타베이스 시스템 환경에서의 동시성 제어 기법은 기존의 기법을 보완하거나 새로운 기법이 필요하다. 본 연구에서는 록 관리를 용이하게 하면서 우선순위 역행과 교착상태 문제를 해결한 우선순위를 기반으로 한 이중 록킹(Priority-based-Double-Locking; PDL) 기법을 제안한다.

3.1.1 우선순위를 기반으로 한 이중 록킹 기법

트랜잭션 스케쥴링 기법에 의하여 트랜잭션들이 잘 스케쥴링된 경우에도 트랜잭션들 간에 높은 우선순위를 지닌 트랙잭션이 낮은 우선순위를 지닌 트랜잭션을 기다리는 우선순위 역행(priority inversion)이나 교착상태가 발생하는 경우에는 트랜잭션의 마감시간을 보장할 수 없게 된다. 따라서, 본 논문에서 제안한 기법의 특징은 록 단위를 이중 록 모드 즉, 록 관리가 용이한 테이블 록과 공유도를 향상시킬 수 있는 페이지 록 단위를 병행해서 사용하였다. 그리고 각 트랜잭션이 접근하려는 테이블에 트랜잭션의 우선순위를 할당한 뒤 이 테이블의 우선순위를 이용하여 충돌하는 트랜잭션을 해결하므로써 취소되는 트랜잭션의 수를 줄였다.

3.1.1.1 테이블 우선순위 관리

발생된 트랜잭션은 접근하려는 테이블에 트랜잭션 자신의 우선순위를 할당한다. 그러나, 접근하려는 트랜잭션이 이미 우선순위를 할당받은 테이블에 접근하는 경우는 할당받은 테이블의 우선순위와 비교한다. 비교 결과 테이블의 우선순위보다 낮은 우선순위를 지닌 트랜잭션은 무시되고, 접근하는 트랜잭션의 우선순위가 보다 높은 경우에만 테이블의 우선순위 변경한다. 한편, 완료되는 트랜잭션의 우선순위를 가지고 있는 테이블은 다시 그 테이블에 접근하는 트랜잭션들 중 가장 높은 우선순위를 지닌 트랜잭션의 우선순위로 변경한다. 테이블에 우선순위를 할당하고 변경하는 테이블 우선순위 관리 알고리즘은 알고리즘 3.1과 같다.

알고리즘 3.1 테이블 우선순위 관리 알고리즘

```
Table_Priority_Handling(TAi, Ti) /*트랜잭션 Ti가 접근하려는
테이블 TAi에 대한 우선순위 관리*/
Input:테이블 TAi을 접근하려는 트랜잭션 Ti
```

```
Output:테이블 TAi에 트랜잭션 Ti의 우선순위 할당
Algorithm:
High-Priority:The Highest Priority of transactions Ti that will
              access table TAi
1 if(The transaction Ti is a initializing transaction)
2   { for (;;)/*트랜잭션 Ti가 접근하려는 모든 테이블 TAi에 우선
      순위 할당*/
3     { if(The table TAi is not assigned Priority(Ti) yet)
4         Assign Priority(Ti) to table TAi;
5       else
6         if(Priority(Ti)>Priority(TAi)) Assign Priority(Ti) to
          table TAi;
7     }/*For 문에 대한 End*/
8   }
9 else/*The transaction Ti is a committing or aborting trans-
  action)*/
10  { for (;;)/*트랜잭션 Ti가 접근하려는 모든 테이블 TAi에서 우
      선순위를 해제*/
11    if(Priority(Ti)>Priority(TAi)) Assign High-Priority to
      table TAi;
12  }
```

3.1.1.2 록 모드 변환 기법

이 논문에서는 록 관리를 위해 테이블 록 모드와 페이지 록 모드를 병행하여 사용하는데, 이 두 록 모드는 필요에 따라 동적으로 변환된다. 즉, 록 모드 변환에는 테이블 록 모드에서 페이지 록 모드로 변환하는 록 모드 축소 기법과 페이지 록 모드에서 테이블 록 모드로 변환하는 록 모드 확대 기법이 있다. 일반적으로는 록 관리가 용이한 테이블 록 모드만을 사용는데 어떤 트랜잭션에 의하여 이미 점유된 테이블에 록을 요구할 경우는 요구한 트랜잭션의 우선순위와 테이블의 우선순위를 비교한다. 이때 테이블의 우선순위가 요구한 트랜잭션의 우선순위보다 큰 경우에는 요구한 트랜잭션은 블록시키고, 이미 록을 보유하고 사용중에 있는 트랜잭션이 계속 실행된다. 그러나, 테이블의 우선순위가 요구한 트랜잭션의 우선순위보다 작은 경우에는 테이블 충돌이 발생된다. 이때 테이블 충돌을 해결하기 위해 테이블 록 모드에서 페이지 록 모드로 변환하여 페이지마저 충돌이 발생했는지 검사한다. 검사 결과 페이지마저 충돌이 발생했다면 테이블을 이미 점유하고 있는 트랜잭션은 취소시키고 새로 요구한 트랜잭션을 실행한다. 그러나, 페이지 충돌이 발생하지 않았으면 테이블을 이미 점유하고 있는 트랜잭션을 대기시킨 후 요구한 트랜잭션을 먼저 실행한다.

3.1.1.3 충돌 트랜잭션 관리 알고리즘

테이블에 트랜잭션의 우선순위를 할당하고 록 단위를 테이블 록 모드에서 페이지 록 모드로 또는 페이지 록 모드에서 테이블 록 모드로 변환하는 록 모드 변환 기법을 이용하여 록 요구시 충돌하는 트랜잭션을 관리하는 알고리즘은 3.2와 같다.

알고리즘 3.2 우선순위를 기반으로 한 이중 록킹 기법에 의한 충돌 관리 알고리즘

```
Table_Lock_Request_Handling(TAi, Ti)/*테이블 TAi에 대한 트
랜잭션 Ti의 록 요구 관리*/
Input:테이블 TAi를 접근하려는 트랜잭션 Ti
Output:테이블 TAi에 대한 트랜잭션 Ti의 록 인정 및 취소
Algorithm:
1 if(The table TAi was not locked by any Transaction Ti')
2   {The Lock on the table TAi is granted to the transaction Ti;}
3 else {if(Priority(TAi)>Priority(Ti))}
4       {The transaction Ti is blocked;
5        The transaction Ti' runs continuously;
6       }
7     else /*테이블 충돌 발생→록 모드 변환 기법 적용*/
8     {Convert the table-lock-mode into the page-lock-mode;
       /*록 모드 축소 변환 기법*/
9        if(The transaction Ti and transaction Ti' conflict on
           any Page)
           /*페이지 마저 충돌이 발생한 경우*/
10         {The transaction Ti runs;
11          The transaction Ti' is aborted;}
12       else/*페이지는 충돌이 발생하지 않은 경우*/
13         {The transaction Ti runs;
14          The transaction Ti' is waited;}
         /*테이블 충돌을 해결한 후 록 모드 확대 변환 기법 적용*/
15       Convert the page-lock-mode into the table-lock-mode;
16    }/*8 line의 {에 대한}*/
17  }/*2 line의 {에 대한}*/
```

본 기법의 특징은 첫째, 일반적으로는 테이블 록 모드를 사용하는 도중 충돌이 발생하면 충돌을 해결하기 위해 테이블 록 모드에서 페이지 록 모드로 변환하였고, 충돌이 해결된 뒤에는 록 관리를 쉽게하기 위하여 페이지 록 모드에서 테이블 록 모드로 변환하는 이중 록 모드를 사용하였다. 둘째, 충돌이 발생된 트랜잭션들의 우선순위를 비교하는 것이 아니라 새로운 트랜잭션이 사용하려는 테이블 우선순위와 점유한 트랜잭션의 테이블 우선순위를 비교하여 록 변환 모드를 적용하므로써 취소되는 트랜잭션의 수를 줄였다. 셋째, 높은 우선순위를 지닌 트랜잭션이 낮은 우선순위를 지닌 트랜잭션을 기다리는 우선순위 역행과 교착상태를 제거하여 마감시간을 만족하는 트랜잭션의 수를 향상시켰다.

3.1.2 트랜잭션 스케쥴링 수행 예

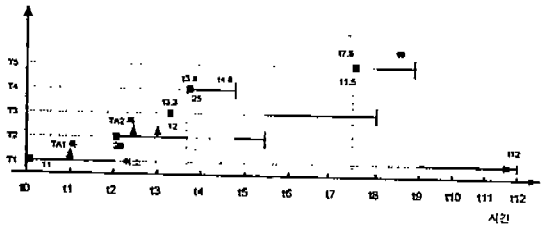
트랜잭션 T1, T2, T3, T4, 그리고 T5는 〈표 3.1〉처럼 마감시간, 예측 실행 시간, 그리고 발생 시간을 갖는다고 가정한다.

〈표 3.1〉 트랜잭션의 시간 정보
〈Table 3.1〉 Time information for transaction

트랜잭션 종류	마감시간	예측 실행 시간	발생시간	우선순위
T1	9	3	t0	11
T2	5	2.5	t2	20
T3	7	2.5	t3.3	12
T4	4	1	t3.8	25/−1
T5	8.5	1	t7.5	11.5

트랜잭션 T1과 T2는 테이블 TA1과 TA2를 접근하고, 트랜잭션 T3, T4, T5는 각각 테이블 TA3, TA4, 그리고 TA5를 접근한다고 가정한다. 트랜잭션에 대한 우선순위 할당은 1/D(t)*100으로 계산한다(단, D(t)는 마감시간). 즉, EDF 기법을 적용한다. 〈표 3.1〉에서 가정한 트랜잭션의 시간 정보를 근거로 EDF 기법과 기존의 HP 기법으로 처리한 결과와 이 논문에서 제안한 스케쥴링 기법과 PDL 기법으로 처리한 결과를 마감시간 초과 및 적중하는 트랜잭션의 갯수를 비교하여 (그림 3.1)에서 (그림 3.2)까지 보여준다.

EDF 기법과 HP 기법으로 처리하면, (그림 3.1)과 같이 우선순위 11을 지닌 트랜잭션 T1은 시간 to에서 시작하여 시간 t1에서 테이블 TA1에 록을 걸고, 트랜잭션 T2가 들어올 때인 시간 t2까지 계속 실행한다. 우선순위 20을 지닌 트랜잭션 T2는 시간 t2에서 트랜잭션 T1을 선점한 후 시간 t2.5에서 테이블 TA2에 록을 걸고 계속 실행한다. 시간 t3에 도착했을때 트랜잭션 T2는 이미 트랜잭션 T1에 의하여 록된 테이블 TA1에 록을 요청하므로써 충돌이 발생한다. 이때 우선순위가 낮은 트랜잭션 T1은 취소되고, 우선순위가 높은 트랜잭션 T2는 우선순위 25를 지닌 트랜잭션 T4가 발생될때인 시간 t3.8까지 계속 실행된다. 한편, 트랜잭션 T3는 시간 t3.3에서 발생하나 우선순위가 낮기 때문에 블록된다. 트랜잭션 T4는 우선순위 25를 가지고 시간 t3.8에서 발생하여 테이블 TA4에 록을 걸고 시간 t4.8까지 실행하여 완료한다. 이때, 트랜잭션 T2는 시간 t4.8에서 다시 시작하여 시간 t5.5까지 실행하여 완료한다.

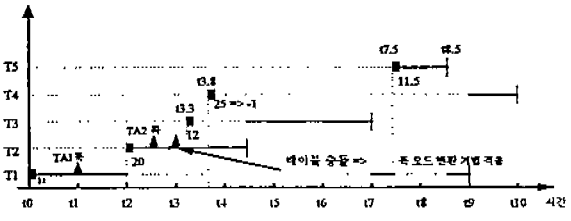


(그림 3.1) HP 기법
(Fig. 3.1) High priority technique

우선순위 12인 트랜잭션 T3는 시간 t5.5에서 테이블 TA3에 록을 걸고 실행하기 시작하여 시간 t8에서 종료된다. 우선순위 11.5를 지닌 트랜잭션 T5는 t7.5에서 발생되었으나 트랜잭션 T3보다 우선순위가 낮아 블록되고, 시간 t8에서 테이블 TA5에 록을 걸고 실행을 시작하여 시간 t9에서 완료한다. 취소되었던 트랜잭션 T1이 t9에서 실행을 시작하여 시간 t12에서 종료된다. 이 방법으로 처리한 결과 트랜잭션 T1, T2, T3, T4, 그리고 T5는 모두 마감시간을 초과하였다.

본 논문에서 제안한 기법으로 처리하면, (그림 3.2)와 같이 우선순위 11을 지닌 트랜잭션 T1은 시간 to에서 시작하여 시간 t1에서 테이블 TA1에 록을 걸고 트랜잭션 T2가 들어올때인 시간 t2까지 계속 실행한

다. 우선순위 20을 지닌 트랜잭션 T2는 시간 t2에서 트랜잭션 T1을 선점한 후 시간 t2.5에서 테이블 TA2에 록을 걸고 계속 실행한다. 시간 t3에 도착했을때 트랜잭션 T2는 트랜잭션 T1에 의해 이미 록된 테이블 TA1에 록을 요청하나 테이블 TA1의 우선순위 20과 같기 때문에 충돌이 발생한다. 이런 충돌을 해결하기 위하여 테이블 록 단위에서 페이지 록 단위로 변환하여 충돌을 해결한 후 시간 t4.5까지 실행하여 완료한다. 이때, 트랜잭션 T1과 T2는 페이지 충돌이 발생되지 않아 우선순위가 낮은 트랜잭션 T1을 취소시키지 않고 대기시킴으로써 자원의 낭비를 줄였다. 한편, 충돌을 해결한 후에는 록 관리를 용이하게 하기위해 페이지 록 단위에서 테이블 록 단위로 변환한다. 시간 t3.3에서 발생했으나 우선순위가 낮아 블록된 우선순위 12를 지닌 트랜잭션 T3는 시간 t4.5에서 테이블 TA3에 록을 걸고 실행을 시작하여 시간 t7에서 완료한다. 트랜잭션 T1이 다시 시간 t7에서 시간 t7.5까지 실행한다. 우선순위 11.5를 지닌 트랜잭션 T5는 시간 t7.5에서 시작하여 테이블 TA5에 록을 걸고 시간 t8.5에서 완료한다. 트랜잭션 T1이 다시 시간 t8.5에서 시작하여 시간 t9에서 완료한다. 우선순위 −1을 지닌 트랜잭션 T4는 시간 t9에서 테이블 TA4에 록을 걸고 시작하여 시간 t10에서 완료한다.



(그림 3.2) 제안한 스케줄링 기법과 PDL 기법
(Fig. 3.2) Proposed scheduling and double locking technique

기존의 방법에서는 EDF 기법을 적용하여 트랜잭션 T4에 우선순위 25를 할당하였으나 이 논문에서는 마감시간 이내에 처리 가능한지 여부를 검사한 결과 마감시간 이내에 처리하지 못하므로써 짧은 트랜잭션에 높은 우선순위를 할당하는 기법인 짧은 트랜잭션 우선 기법을 적용하였다. 즉, 예측 실행 시간에 마이너스(−)를 붙여 −1을 우선순위로 할당하였다. 이 방법으로 처리한 결과 트랜잭션 T4는 마감시간을 초과하였고, 트랜잭션 T1, T2, T3, 그리고 T5는 모두 마감시간 이내에 처리하였다. 종합적으로 분석하면, EDF 기법과 HP 기법으로 처리하면, 트랜잭션 T1, T2, T3, T4 그리고 T5는 모두 마감시간을 초과하였다. EDF 기법과 PI기법으로 처리하면, 트랜잭션 T1만 마감시간 이내에 처리 하였고, 나머지 트랜잭션 T2, T3, T4, 그리고 T5는 모두 마감시간을 초과하였다. 제안한 기법의 경우는 트랜잭션 T4만 마감시간을 초과하였고, 나머지 트랜잭션 T1, T2, T3, 그리고 T5는 모두 마감시간 이내에 처리 하였다. 그러나 트랜잭션 T4는 발생때부터 마감시간 이내에 처리할 수 없는 트랜잭션이었다.

4. 성능 평가 및 분석

본 논문에서 제안한 PDL 기법의 성능을 평가하기 위해 이산적 사건 모델을 지원하는 SLAM II 시뮬레이션 언어을 이용하여 시뮬레이션을 수행하였다[3].

4.1 시뮬레이션 모델

시뮬레이션의 모델의 주요 구성 요소는 터미널, 트랜잭션 처리기, 트랜잭션 관리기, 동시성 관리기, 그리고 데이타 관리기로 구성된다. 터미널에서 지수적 분포로 발생시킨 트랜잭션은 시스템에 들어갈 때 실행할 준비가 되어있다. 트랜잭션 처리기는 각 트랜잭션에게 마감시간을 할당하고, 마감시간과 예측 및 실행 시간 정보를 근거로 각각의 트랜잭션에게 우선순위를 할당한다. 이때 마감시간은 다음과 같은 방법으로 계산한다. 마감시간＝시작 시간＋예측 실행 시간＋Slack_factor. 이때, Slack_factor는 마감시간의 긴급함(tightness)과 느슨함(looseness)을 제어하는 매개변수이다. 이 마감시간과 예측 실행 시간에 의해 할당된 우선순위에 따라 적중 큐와 초과 큐에 대기시킨다. 트랜잭션 관리기는 큐에서 가장 높은 우선순위를 지닌 트랜잭션을 선택하여 트랜잭션이 마감시간을 초과했는지 여부를 결정하고, 록 요구와 각 트랜잭션에 대한 데이타베이스 연산을 수행한다. 동시성 관리기는 프로토콜 명세에 따라 록 요구를 스케줄한다. 요구한 트랜잭션이 테이블 우선순위보다 작다면 정렬된 EDF

QUEUE 또는 STF(Short Transaction First) QUEUE에 대기된다. 요구한 트랜잭션이 테이블 우선순위보다 크다면 테이블 록 모드에서 페이지 록 모드로 변환하여 페이지마저 충돌이 발생했는지 검사한다. 페이지 충돌이 발생하면 점유하고 있는 트랜잭션을 취소시키고 새로운 트랜잭션에게 록을 부여한다. 지연된 록 요구는 충돌했던 연산들이 그들의 록을 해제할 때만 다시 스케쥴된다. 록 요구가 부여될 때 동시성 관리기는 록이 부여되었다는 신호를 트랜잭션 관리기에게 보낸다. 이때, 트랜잭션 관리기는 데이타 관리기에게 데이타베이스 연산을 수행하도록 한다. 데이타 관리기는 데이타베이스 연산들을 실행한다.

4.2 실험 환경

시뮬레이션 매개 변수는 우선적으로 CPU의 갯수를 나타내는 CPU_num의 값에 1을 주었고, 주기억 데이타베이스 환경이기 때문에 데이타베이스에 저장된 페이지의 수와 주기억 장치에 저장할 수 있는 페이지의 수가 같다고 가정한다. 따라서 DB_size와 MEM_size에 동일한 값인 500 pages을 주었다. 한편 고장이 없는 시스템으로 가정했기 때문에 디스크의 수를 나타내는 매개변수 DISK_num의 값에 0을 주었다. 평균 도착율 Arr_Rate을 가지고 시스템에 들어오는 트랜잭션에 의해 접근될 페이지는 평균 Pages를 가지고 정규 분포로 선택된다. 이때 각 페이지는 갱신된 확률을 가지고 변경된다. 트랜잭션에 의해 변경될 확률의 값을 1로 한 이유는 모든 록을 독점 록으로 한정하여 충돌이 두 트랜잭션들 사이에서만 발생하도록 하였기 때문이다. 한 트랜잭션에 대한 CPU 요구를 R이라 가정하면, R은 특정한 트랜잭션에 의해 접근된 실제 페이지의 수에 접근된 페이지 당 CPU 서비스 시간을 곱한 값이다. R에 대한 트랜잭션의 예측 실행 시간 E(t)에 대한 정확도는 예측 실행 시간의 오류를 나타내는 매개변수 Est_Err에 의해 제어된다. 즉 E(t) =R +Est_Err이다. 이때 Est_Err의 값이 0이면 트랜잭션의 예측 실행 시간 E(t)가 정확한 경우이다. 타 연구와 비교하기 위해 Est_Err의 값을 0에서부터 4까지 변화시키면서 성능을 측정하였다. 매개변수 Rebort는 한 트랜잭션을 재시작 또는 취소하기 위해 필요한 CPU의 시간을 제어한다. 한 트랜잭션의 취소는 적합성 결정 정책에 의하여 이루어지고 시스템으로 부터 제거된다. 따라서 트랜잭션은 재실행되지 않는다. 한편 록 충돌로 인하여 트랜잭션이 취소될때 트랜잭션은 재수행되고 준비 큐에 다시 놓인다. 마감시간의 할당은 트랜잭션의 여유 시간에 대한 하한 값과 상한 값을 나타내는 매개 변수 Min_Slack과 Max_Slack에 의해 조정된다. 즉 마감시간=시작 시간 +트랜잭션의 예측 실행 시간 +Min_Slack or Max_Slack이다. 이때 Min_Slack과 Max_Slack은 마감시간의 긴급함과 느슨함을 제어하는 매개변수이다. 이 매개변수의 값은 0에서 3까지 변화를 주며 성능을 검사하였다. 매개변수에 할당된 값은 성능 평가에서 일반적으로 많이 사용되는 값을 기초로 하였다[1].

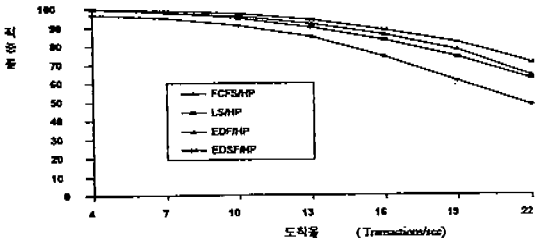
4.3 성능 비교 분석

시뮬레이션 비교 분석은 먼저 우선순위 할당 정책으로는 FCFS 기법, LS 기법, EDF 기법, 그리고 본 연구에서 제안한 EDSF(Earliest Deadline and Short Transaction First) 기법을 선택하였고, 동시성 제어 정책으로는 HP 기법, PI 기법, PC 기법, 그리고 본 연구에서 제안한 PDL 기법을 선택하였다. 이 경우 16개의 알고리즘이 생성되나 그 중 도착률 실험과 예측 실행 시간(estimate run time) 실험을 통해 성능을 비교 분석 한다.

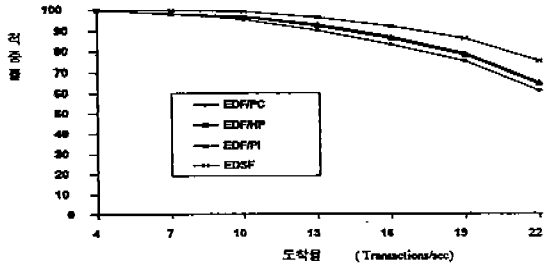
4.3.1 도착률 실험

우선순위 할당 정책 FCFS, LS, EDF, EDSF와 동시성 제어 기법 HP를 각각 적용시켜 도착률 변화에 따른 성공한 트랜잭션의 백분율을 시뮬레이션한 결과가 (그림 4.1)의 그래프와 같다. (그림 4.1)의 그래프를 분석해보면 낮은 적재하에서는 FCFS/HP, LS/HP, EDF/HP, 그리고 EDSF 모두 마감시간이내에 수행된다. 그러나, 도착률이 점점 증가하여 초당 22개가 발생하는 과다적재시에는 본 연구에서 제안한 기법인 EDSF/HP는 (그림 4.1)에서 FCFS/HP 보다 23%, LS/HP 보다 9%, EDF/HP 보다 7%가 성능이 향상됨을 알 수 있다. (그림 4.1)의 특징은 EDF를 사용한 스케쥴링 기법들이 적정 수준에서는 성능이 양호하나, 과다적재시에는 다른 스케쥴링에 비해 급격히 성능이 저하됨을 볼수 있다. 우선순위 할당 정책 EDF에 동시성 제어 정책 PC, HP, PI와 본 연구에서 제안한 EDSF 기법에 PDL 기법을 적용시켜 도착률 변화

에 따른 성공한 트랜잭션의 백분율을 시뮬레이션한 결과는 (그림 4.2)와 같다. (그림 4.2)의 그래프를 분석해보면 낮은 적재하에서는 EDF/PC, EDF/HP, EDF/PI, 그리고 EDSF가 모두 마감시간을 만족한다. 그러나, 도착률이 점점 증가하여 초당 22개가 발생하는 과다적재시에 본 연구에서 제안한 기법이 (그림 4.2)에서 보는바와 같이 EDF/PC 보다는 14%, EDF/HP 보다는 11%, EDF/PI 보다는 10% 성능을 향상시켰다.



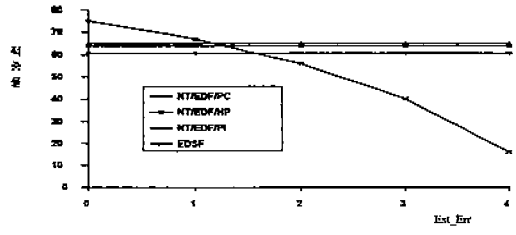
(그림 4.1) 도착률 변화에 따른 트랜잭션의 적중률(FCFS vs LS vs EDF vs EDSF/HP)
(Fig 4.1) Transaction hit rate(FCFS vs LS vs EDF vs EDSF, HP)



(그림 4.2) 도착률 변화에 따른 트랜잭션의 적중률(EDF, PC vs HP vs PI vs EDSF)
(Fig 4.2) Transaction hit rate(EDF, PC vs HP vs PI vs EDSF)

4.3.2 예측 실행 시간 실험

예측 실행 시간 E(t)는 스케줄링 구성 요소인 적합성 결정 정책, 우선순위 할당 정책, 그리고 동시성 제어 정책들 중 일부에서만 사용한다. 세 가지의 적합성 결정 정책중 FDE만 예측 실행 시간 E(t)를 사용하므로 NTE를 사용하는 스케줄링에는 전혀 영향을 미치지 않는다. 그러나, 본 논문에서는 첫째, NTE를 사용하면서 마감시간 이내에 처리가 가능한 트랜잭션과 마감시간 이내에 처리가 불가능한 트랜잭션을 분류할 때 예측 실행 시간 E(t)을 사용한다. 둘째, 연쇄적인 마감시간 초과로 인하여 성능이 저하되는 문제점을 제거하기 위하여 적중 류에서 선택한 트랜잭션에 대해 실행 직전에 다시 한번 마감시간 초과를 검사할 때 예측 실행 시간 E(t)를 사용한다. 그러므로 예측 실행 시간의 정확도가 트랜잭션 스케줄링에 큰 영향을 미친다. (그림 4.3)에서 보는 바와 같이 NTE/EDF/PC, NTE/EDF/HP, 그리고 NTE/EDF/PI는 오류가 낮을 때(Est_Err = 0)와 오류가 높을 때(Est_Err = 4) 사이의 성능은 일정하다. 그러나, EDSF는 예측 실행 시간에 대한 오류가 낮을 때는 NTE/EDF/PC 보다 14%, NTE/EDF/HP 보다 11%, 그리고 NTE/EDF/PI 보다 10% 성능이 좋으나 오류가 높을수록 성능은 급격이 저하되며 오류가 가장 높을 때(Est_Err = 4) 성능은 NTE/EDF/PC 보다 45%, NTE/EDF/HP 보다 48%, 그리고 NTE/EDF/PI 보다 49% 성능이 저하된다. 이 성능 평가를 토대로 분석한 결과 본 논문에서 제안한 기법은 예측 실행 시간이 정확한 경우에만 좋은 성능을 기대할 수 있다.



(그림 4.3) 예측 실행 시간 오류에 따른 트랜잭션의 적중률(EDF, PC vs HP vs PI vs EDSF)
(Fig 4.3) Transaction hit rate(EDF, PC vs HP vs PI vs EDSF)

5. 결 론

마감시간 이내에 자료를 처리해야하는 실시간 데이타베이스 시스템을 기존의 데이타베이스 시스템에서 사용된 기법을 그대로 적용할 경우 다음과 같은 문제점이 발생한다. 각 트랜잭션이 접근하려는 데이타에 대해 일관성을 보존해야하는 일관성 제약 조건

은 트랜잭션을 어느 시점에서 블록시키거나 교착상태를 유발시키므로 트랜잭션의 마감시간을 만족하기가 어렵다. 따라서, 본 연구에서는 위의 문제점에 대한 해결책으로 마감시간 이내에 처리가 가능한 트랜잭션들은 EDF 기법으로 우선순위를 할당하여 적중 큐에 대기시키고, 마감시간 이내에 처리가 불가능한 트랜잭션들에 대해서는 짧은 트랜잭션 우선 기법으로 우선순위를 할당하여 초과 큐에 대기시켰다. 그리고 적중 큐에 있는 트랜잭션에 대해 실행 바로 직전 마감시간 초과 여부를 동적으로 검사하여 마감시간 이내에 처리하지 못하는 트랜잭션에 대해서는 짧은 트랜잭션에 높은 우선순위를 할당하는 방법으로 다시 우선순위를 할당하여 적중 큐에서 초과 큐에 보내는 큐 이동 등 새로운 개념을 도입하여 과다적재시 연쇄적인 마감시간 초과로 인하여 성능이 저하되는 EDF 기법의 문제점을 해결하여 성능을 향상시킨 동시성 제어 기법을 제안하였다. 끝으로, 이 논문에서 제안한 동시성 제어 기법과 PDL 기법의 성능을 타 연구와 비교 평가하기 위해 모델링하고 이산적 사건 모델을 지원하는 SLAM II 언어를 이용하여 시뮬레이션을 수행하였다. 추후 연구 방향으로는 첫째, 제안한 PDL 동시성 제어 기법의 구현과 빠른 응답, 그리고 트랜잭션 처리율의 향상을 위해 주기억 데이타베이스 시스템을 개발하기 위한 프로토타입을 설계하여 현재 구현중에 있으며 이를 정당화하기 위한 성능 분석이 요구된다. 둘째, 주기억 데이타베이스 시스템에서는 전체 데이타베이스를 주기억 장치 내에 상주시킨 상태에서 각종 데이타베이스 연산을 수행하기 때문에 시스템 전원이나 주기억 장치 칩등에 이상이 발생할 경우 데이타베이스 전체가 손실된다. 이를 위해서는 로깅, 검사점(checkpointing) 실행, 로그 압축(log compaction) 방법 등을 고찰하여 주기억 장치 데이타베이스 회복 기법에 대한 연구가 필요하다.

참 고 문 헌

[1] R. Abbott, H. Garcia-Molina, "Scheduling Real-Time Transactions: A Performance Evaluation", Proc. of VLDB, vol. 14, p.1-12, 1988.

[2] A. Agrawal, A. EL Abadi and R. Jeffers, "Using Delayed Commitment in Locking Protocols for Real-Time Databases", ACM SIGMOD RECORD, vol. 21, no. 1, p.104-113, 1992.

[3] A. Alan, B. Pritsker, "Introduction to Simulation and Slam II", A Halsted Press Book, John Wiley & Sons, Third Edition, 1986.

[4] G. von Bultzingslowen, et al, "KARDAMOM-A Dataflow Database Machine for Management Systems", Proc. of VLDB, vol. 14, p.239-250, 1988.

[5] Haritsa, J., Carey, M., and Livny, M. "On being optimistic about real-time constraints", In Proceedings of the ACM Symposium on Principles of Database Systems, p.331-343, 1990.

[6] J. R. Haritsa, M. Linvy and M. J. Carey, "Earliest Deadline Scheduling for Real-Time Database Systems", in Proc. of Real-Time Systems Symposium, IEEE, p.232-242, 1991.

[7] L.Sha, R. Rajkummar, and J.P.Lehoczky, "Concurrency Control for Distributed Real-Time Databases", ACM SIGMOD RECORD, vol. 17, no. 1, p. 82-98, 1988.

[8] L. Sha, R. Rajkumar, J. Lehoczky "Priortiy Inheritance Protocols: An Approach to Real-Time Synchronization", IEEE Transaction on Computers, vol. 39, no. 9, p.1175-1185, 1990.

[9] Mukesh Singhal, "Issues and Approaches to Design of Real-Time Database Systems" ACM SIGMOD RECORD, vol. 17, no. 1, 1988.

[10] Sang H. Son, "Real-Time Database Systems: Issues and Approaches" ACM SIGMOD RECORD, vol. 17, no. 1, 1988.

[11] S. H. Son, S. Park and Yi Lin, "An Integrated Real-Time Locking Protocol" Proc. of Data Eng., vol. 8, p.527-534, 1992.

[12] ByoungOh Koh and HaeChull Lim, "Transaction Scheduling and Priority-based Locking Techniques of Main Memory Database System in Soft Real-Time Environments" proceedings of ITC-CSCC'96., p.1254-1257, 1996.

[13] ByoungOh Koh, DonJe Cho and HaeChull Lim, "Soft Real-Time Transaction Scheduling in Main Memory Database System" Proceedings of The

22nd KISS Fall Conference, p.297-300, 1995.

[14] Kwok-wa Lam, Kam-yiu Lam and Sheung-lun Hung, "Real-time Optimistic Concurrency Control Protocol with Dynamic Adjustment of Serialization Order", Proceedings of the IEEE, pp. 174-179, 1995.

[15] S. H Son and Seok Park, "Scheduling and Concurrency Control for Real-Time Database Systems" Proc. of DASFA, 1993.

최 의 인

1982년 숭전대학교 계산통계학과 졸업(학사)
1984년 홍익대학교 전자계산학과 졸업(이학석사)
1995년 홍익대학교 전자계산학과 졸업(이학박사)
1985년~1988년 공군 교육사 전산실장
1992년~1996년 명지전문대학 전자계산과 조교수
1996년~현재 한남대학교 컴퓨터공학과 조교수
관심분야:실시간데이타베이스, 주기억데이타베이스, 클라이언트/서버 데이타베이스

고 병 오

1986년 충남대학교 계산통계학과 졸업(학사)
1989년 홍익대학교 전자계산학과 졸업(이학석사)
1996년 홍익대학교 전자계산학과 졸업(이학박사)
1993년 영동전문대학 전자계산과 전임강사
1994년~현재 세명대학교 정보처리학과 조교수
관심분야:실시간데이타베이스, 객체지향데이타베이스