트랜잭션을 위한 데이터 우선순위 기반형 시간소인 순서화 기법

윤 석 환† · 김 평 중† · 박 지 은† · 이 재 영† ·
이 동 현† · 궁 상 환†

요 약

트랜잭션 순서화 기법중 기존의 시간소인 순서화 기법(Timestamp-Ordering Protocol)은 시스템에 진입하는 트랜잭션에 시간소인을 배정하여 이를 기준으로 트랜잭션을 순서화 함으로서 우선순위가 높은 트랜잭션이 나중에 처리되는 우선순위 바뀜 현상이 발생할 수 있다. 이를 방지하기 위하여 트랜잭션들을 그들의 시스템 진입 시점에 따라 일정한 시간간격으로 범주화 한 후, 같은 범주내에 있는 트랜잭션들에 대해서는 그들의 우선순위에 따라 순서화하는 데이터 우선순위 기반형 시간소인 순서화 기법을 제안한다. 이 기법의 성능을 평가하기 위하여 실시간 데이터베이스 시스템으로 시뮬레이션 환경을 구축하여 기존의 기법들과 성능을 비교하였으며, 제안하는 기법이 높은 부하와 높은 데이터 충돌의 조건하에서 기존의 시간소인 순서화 기법보다 성능이 우수함을 확인하였다.

Data Priority-Based Timestamp-Ordering Protocol for Transactions

Seokhwan Yoon† · Pyoungjung Kim† · Jieun Park† · Jaeyoung Lee† ·
Tonghyun Lee† · Sanghwan Kung†

ABSTRACT

Timestamp-Ordering Protocol among transaction scheduling algorithms can cause the priority reversion that a transaction with higher priority is processed after the transaction with lower priority by assigning timestamp to transactions entering system and scheduling them based on the timestamp. To prevent this reversion, we suggest a data priority-based timestamp ordering protocol to schedule transactions based on their priority within the same timestamp group after grouping transactions into constant time interval based on entering points. To evaluate the performance of this protocol, we compared the performance of this protocol with that of others after constructing the simulation environment with real time database system. We verified that the performance of proposed protocol is superior to that of timestamp ordering protocol under the condition of high load and high data conflicts.

† 정 회 원 : 한국전자통신연구원 컴퓨터연구단
논문접수 : 1997년 1월 31일, 심사완료 : 1997년 4월 17일

1. 서 론

'90년대에 들어 종래의 메인 프레임 위주의 컴퓨팅 환경이 분산 환경으로 바뀌어 가면서 온라인 트랜잭션 처리(OLTP:On-Line Transaction Processing) 방식에 대한 관심이 고조되고 있다. 이러한 방식은 이기종간의 인터페이스까지 포함하는 분산처리 능력이 강화되어 원격지간의 컴퓨터 통신 및 공동작업 등을 가능하게 하고 있다. 이는 클라이언트인 단말의 기능이 강화되고 네트워크가 빠르고 안정적으로 발전하여 클라이언트와 서버간의 통신 및 클라이언트간의 통신이 보다 용이해졌기 때문이다. 분산 환경에서의 컴퓨터 통신은 기본단위인 트랜잭션을 통해서 이루어지며, 특정 클라이언트가 요구하는 내용을 서버 및 다른 클라이언트가 처리하여 다시 그 클라이언트에게 보내 주는 구조로 되어있다.

트랜잭션이란 특정 클라이언트에서 서버 및 다른 클라이언트에 접근하는 하일 시스템 명령어와 데이터베이스 명령어 등을 처리하는 일련의 기본 조작들의 순서적 나열이며, 데이터베이스의 현재의 일치 상태를 새로운 일치 상태로 변환하는 연산 행위이다. 데이터베이스에 접근하는 동시 다발적인 트랜잭션들은 데이터베이스 관리 시스템에 존재하는 트랜잭션 처리기에 의해 블럭되는 일정한 순서로 처리되어야 한다. 트랜잭션 처리기는 트랜잭션 순서화 기법(TSA :Transaction Scheduling Algorithm)에 의거하여 트랜잭션들을 순차화 한다.

트랜잭션 순서화 기법은 일반적으로 잠금 기법, 시간소인 순서화 기법, 낙관적 기법으로 분류할 수 있으며, 잠금 기법은 구현 방법에 따라 다시 항시 블럭 잠금 기법, 우선순위 상속 잠금 기법, 우선순위에 기반한 잠금 기법, 우선순위 상한 잠금기법으로 세분된다. 이를 기반으로 기존의 연구 결과를 크게 3그룹 즉, 잠금 기법, 시간소인 순서화 기법, 낙관적 기법으로 분류하여 각각의 특징을 분석하였다.

잠금 기법은 트랜잭션이 시스템에 진입하면 트랜잭션의 우선순위에 따라 트랜잭션들을 순서화하는 방식이어서 트랜잭션들간에 서로가 서로의 선처리를 기다리는 교착 상태가 발생할 수 있다는 문제점이 있다. 시간소인 순서화 기법은 트랜잭션의 우선순위를 전혀 고려하지 않고 시간소인에 의해서만 순서화 하기 때문에 우선순위가 높은 트랜잭션을 빨리 처리하지 못하는 단점이 있다. 그리고 낙관적 기법은 읽기 연산, 확인 연산, 쓰기 연산의 3단계 과정을 거침으로서 트랜잭션들 간의 충돌 및 오버헤드 비용 발생 가능성이 높다는 단점이 있다.

이러한 문제점들은 다음의 방법으로 해결될 수 있다. 잠금 기법에서의 교착상태 발생 가능성을 방지하기 위해서는 트랜잭션과 트랜잭션이 접근하려는 데이터 항목 모두에 같은 우선순위를 부여하여 데이터 항목 입장에서의 가장 높은 우선순위에 대응하는 트랜잭션을 먼저 처리하도록 하는 것이다. 그리고 시간소인 순서화 기법에 대해서는 트랜잭션들을 그들의 도착 시점에 따라 일정한 시간간격에 의한 시간그룹으로 분류한후, 같은 시간그룹 내의 트랜잭션 순서화를 위해서는 앞에서의 교착상태 방지를 위한 방식을 활용하는 것이다.

제안하는 트랜잭션 순서화 기법들은 다음과 같다. 첫째는 데이터 우선순위 기반형 잠금 기법(DPLP:Data Priority-Based Locking Protocol)으로서 기존 잠금 기법들의 단점인 트랜잭션 상호간의 교착상태 발생 가능성을 예방하기 위하여 트랜잭션의 우선순위를 트랜잭션과 접근하려는 데이터 항목 모두에 부여한 후, 데이터 항목 입장에서 가장 높은 우선 순위에 대응하는 트랜잭션을 먼저 처리하도록 하는 것이다. 둘째는 데이터 우선순위 기반형 시간소인 기법(DPTOP:Data Priority-Based Timestamp Ordering Protocol)으로서 기존 시간소인 순서화 기법에서 트랜잭션의 우선순위를 고려하지 않는 문제점들을 보완하기 위하여 트랜잭션들의 도착 시점에 따라 일정한 시간소인 그룹으로 분류한 후, 같은 시간소인 그룹내의 트랜잭션 순서화에 대해서는 앞에서 제안한 DPLP 기법을 적용하는 것이다.

그리고 제안하는 기법의 성능을 평가하기 위해서는 일정한 크기의 데이터베이스를 대상으로 기존의 기법들과 함께 시뮬레이션 한 후, 각각의 성능을 비교하였다. 기존의 기법들과 제안하는 기법들 각각을 C 언어로 코딩하여 이를 SUN 워크스테이션 환경하에서 제 4장의 시뮬레이션 모델 및 변수에 따라 시뮬레이션 하였다. 성능 평가를 위한 기준으로는 성공 비율, 평균 지연, 충돌 비율, 재시작 비율을 이용하였다.

2. 이론적 배경

2.1. 잠금 기법

2.1.1 항시 블럭 잠금 기법(ABLP : Always Block Locking Protocol)

항시 블럭 잠금 기법은 엄격한 2단계 잠금 구조에 근거를 두고 있다[15]. 잠금 요구가 충돌을 일으키면, 잠금을 요구하는 트랜잭션들은 충돌하는 잠금을 유지하고 있는 스케줄러에 의해 블럭된다. 잠금요청을 순서화 할 때에 트랜잭션들의 실시간 특성은 고려하지 않는다. 즉, 잠금 요청들은 선입선출(FIFO : First-in First-out) 순서에 의해 처리된다. 데이터 접근 순서화 시 우선순위를 고려하는 잠금 기법의 성능 이득을 측정할 때에, 항시 블럭 잠금 기법을 기준으로 이용한다. ABLP 기법을 포함한 모든 기법에 대해서, 트랜잭션의 중앙처리장치 요구는 실시간 우선 순위에 기반하여 순서화한다. 중앙처리장치를 요구하는 높은 우선 순위의 트랜잭션은 보다 낮은 우선 순위의 트랜잭션 수행을 선점할 수 있다. 〈표 2-1〉는 ABLP 기법에서의 스케줄러 함수를 보여 준다.

〈표 2-1〉 ABLP기법에서의 스케줄러 함수
〈Table 2-1〉 scheduler function in ABLP

```
TR          lock-holding transaction

lock_request_handling(D, T) {
   if ( D is locked) {
  block T
   }
    otherwise {
  Lock on D is granted to T
     }
   }
```

2.1.2 우선순위에 기반한 잠금 기법(PBLP : Priority-Based Locking Protocol)

이 기법에서는 높은 우선순위의 트랜잭션이 언제나 충돌시의 승자가 된다[3]. 필요할 때는 언제든지 낮은 우선순위의 트랜잭션을 취소함으로서 우선순위 바뀜 현상을 예방한다. 잠금 충돌을 해결하기 위해서, 잠금 요청 트랜잭션의 우선순위가 잠금을 가진 트랜잭션의 우선순위보다 높다면, 잠금을 가졌던 트랜잭션은 포기되고, 잠금 요청 트랜잭션이 잠금을 부여받는다. 반대로 잠금 요청 트랜잭션의 우선순위가 보다 낮다면, 잠금 요청 트랜잭션은 높은 우선순위의 트랜잭션에 의해 블럭된다.

만약 데이터 항목이 트랜잭션 그룹에 의해 읽기 잠금 되어 있다면, 그 항목에 쓰기잠금을 요청하는 트랜잭션 T는, 잠금을 공유하는 다른 트랜잭션이 T의 우선순위보다 높을 경우에 블럭된다. 만약, 트랜잭션 T의 우선순위가 잠금을 공유하는 모든 트랜잭션의 우선순위보다 높다면, 잠금 공유 그룹에 존재하는 모든 트랜잭션은 포기된다(〈표2-2〉).

〈표 2-2〉 PBLP기법에서의 스케줄러 함수
〈Table 2-2〉 scheduler function in PBLP

```
TR   lock-holding transaction

lock_request_handling(D, T) {
   if ( D is not locked) {
  Lock on D is granted to T
   }
    else if ( Priority(T) > Priority(TR)){
  Lock on D is granted to T
   }
    otherwise {
  T is blocked
   } }
```

2.1.3 우선순위 상속 기법(PILP : Priority Inheritance Protocol)

ABLP 기법의 단점은 "우선순위 바뀜" 문제이다. 우선순위 바뀜 문제는 높은 우선순위의 트랜잭션이 낮은 우선순위의 트랜잭션에 의해 블럭되는 경우를 가리키는 것으로서 PILP 기법은 이 문제를 해결하기 위해서 제안된 방법이다[4]. 이 기법은, 한 트랜잭션이 높은 우선순위의 트랜잭션을 블럭할 때, 자신이 블럭한 모든 트랜잭션들의 우선순위중 가장 높은 우선순위를 상속받게 하는 것이다. 상속에 의해 잠금을 가진 트랜잭션은 가장 높은 우선순위를 가지게 되어 잠금을 유지할 수 있다. PILP 기법은 높은 우선순위 트랜잭션의 블럭 시간을 단축시키는 목적을 지니고 있다.

우선순위를 상속받은 트랜잭션이 교착상태 때문에 포기될 때에, 그 트랜잭션은 원래의 우선순위로 환원된다. 잠금을 공유하고 있는 트랜잭션 그룹이 데이터 잠금을 보유하고 있으며, 데이터 항목에 대한 충돌 때문에 높은 우선순위의 트랜잭션이 블럭되어 있다

면, 공유된 잠금 그룹에 존재하면서 블럭된 트랜잭션보다 낮은 우선순위의 트랜잭션은 블럭되어 있는 트랜잭션의 우선순위를 상속받는다. 〈표 2-3〉는 PILP 기법에서의 스케줄러 함수를 보여준다.

2.1.4 우선순위 상한 기법(PCLP : Priority-Ceiling Locking Protocol)

L.Sha et.al.에 의해 제안된 우선순위 상한 기법은 우선순위 상속 기법(PILP)을 확장한 것이다[5, 9]. 이는 PILP 기법에서의 교착상태 문제를 제거하였으며, 높은 우선순위 트랜잭션의 블럭 시간을 줄이고자 하는 것이다. 데이터 항목의 우선순위 상한은 그 항목에 잠금을 가질 수도 있는 가장 높은 우선순위의 트랜잭션의 우선순위이다. 데이터 항목에 잠금을 얻으려면, 데이터 항목의 우선순위 보다 높은 우선순위를 가져야 한다. 그렇지 않으면 블럭된다. 잠금을 가진 트랜잭션이 수행되고 영구 기억장치에 그 결과가 수록되면 데이터 항목의 우선순위 상한값은 재조정된다.

〈표 2-3〉 PILP 기법에서의 스케줄러 함수
〈Table 2-3〉 scheduler function in PILP

```
TR          lock-holding transaction

lock_request_handling(D, T) {
    if (TR = NULL) {
    Lock on D is granted to T
    }
    else {
    T is blocked
    if ( Priority(T) > Priority(TR)){
        Priority(TR)=Priority(T)
    }
}
```

〈표 2-4〉 PCLP 기법에서의 잠금요청 처리 과정
〈Table 2-4〉 lock request handling process in PCLP

```
lock_request_handling(D, T) {
    /* Transaction T requests a lock on data item D */
    if (priority(T) > MAX_PCLP) {
            Lock on D is granted to T;
    }
    else {
            T is blocked by TR;
            if ( priority(T) > priority(TR))
                priority(TR) = priority(T);
            }
        }
```

스케줄러는 시스템내의 데이터 항목 각각에 대하여 트랜잭션 리스트를 관리하여야 한다. 이 리스트에는 데이터 항목에 접근하는 트랜잭션과 접근 예정인 트랜잭션의 고유번호와 우선순위가 포함된다. 〈표 2-4〉에서는 집합 프로세스의 잠금 요청을 스케줄러가 어떻게 처리하는지를 보여 준다.

2.2 시간소인 순서화 기법(TOP : Timestamp-Ordering Protocol)

이 기법은 시간소인 순서화에 의한 동시성 제어 기법의 기본적인 버젼이다[14]. 각 트랜잭션은 시스템에 진입할 때 시간소인을 배정받는다. 충돌 연산들은 트랜잭션의 시간소인에 의해 순서가 정해진다. 트랜잭션의 각 연산은 트랜잭션의 시간소인을 갖는다. 각 데이터 항목에 대하여, 그 항목에 대한 읽기 연산의 가장 큰 시간소인과 쓰기 연산의 가장 큰 시간소인이 그 항목의 읽기/쓰기 시간소인으로서 각각 유지된다. 데이터 항목에 대한 읽기 요청은, 그 항목을 수정한 보다 더 큰 시간소인을 가진 트랜잭션이 없을 경우, 채택된다. 쓰기 요청은, 그 항목을 이미 읽기했거나 쓰기했던 보다 더 큰 시간소인을 가진 트랜잭션이 없을 경우, 채택된다. 〈표 2-5〉에서는 시간소인 순서화 기법에서의 스케줄러 함수를 보여 준다.

〈표2-5〉 TOP 기법에서의 스케줄러 함수
〈Table 2-5〉 scheduler function in TOP

```
transation_conflict_handling(D, T) {
    if ( D is not locked) {
            Lock on D is granted to T
    }
    else if ( Priority(T) > Priority(TR)){
            Lock on D is granted to T
    }
    else {
            T is blocked
    } }
```

2.3 낙관적 기법 (OP : Optimistic Protocol)

이 기법은 H.T.Kung과 J.T.Robinson이 제안한 것으로서 순차적 확인을 하는 낙관적 순서화 기법이다[7]. 각 트랜잭션의 실행은 3단계-읽기 단계, 확인 단계, 쓰기 단계-로 구성된다. 읽기 단계에서, 실행 트랜잭션은 다른 어떤 트랜잭션에 의해서도 블럭되지 않

으면서 모든 읽기/쓰기 연산을 수행하며, 데이터 수정은 데이터 항목의 임시 저장 공간(local copy)상에서 수행한다.

〈표 2-6〉 OP 기법에서의 스케줄러 함수
〈Table 2-6〉 scheduler function in OP

```
t of T_i :  transaction number-ascending integer-assigned
            at the beginning of the validation phase of the transaction.

validation_request_handling( T) {
for T_i=T_j, min to T_i, max {
    if (read set of T_j and write set of T_i intersect) {
      fail
    }
}
return valid
```

이 때의 데이터 항목들을 다른 트랜잭션이 접근하지는 못한다. 확인 단계에서는 트랜잭션 수행 내용이 데이터베이스 상의 어떤 데이터 비일관성을 초래하였는지를 체크한다. 트랜잭션 T에 의해 읽혀진 어떤 데이터 항목이 다른 트랜잭션 T′에 의해 수정되었다면 그것은 트랜잭션 T의 수행시간동안 유지되며, 트랜잭션 T는 다시 시작된다. 그렇지 않으면, 트랜잭션 T는 데이터베이스에서 수행한 모든 수정사항들을 반영하기 위해서 쓰기 단계로 진입하여 트랜잭션을 수행한다. 〈표 2-6〉에서는 낙관적 제어 기법에서의 스케줄러 함수를 보여 준다.

3. 데이터 우선순위 기반형 트랜잭션 순서화 기법

3.1 데이터 우선순위 기반형 잠금 기법(DPLP : Data-Priority-Based Locking Protocol)

본 절에서는 트랜잭션 및 데이터 항목 모두에 우선순위를 부여하여 교착상태가 발생하지 않도록 트랜잭션을 순서화 할 수 있는 개선된 잠금 기법을 제안한다. 각 데이터 항목은 해당 데이터 항목에 접근할 수 있는 모든 트랜잭션 중 가장 높은 우선순위와 같은 우선순위를 갖는다. 새로운 트랜잭션이 시스템에 도달하면 접근되는 각 데이터 항목의 우선순위는, 그 항목의 우선순위가 해당 트랜잭션의 우선순위보다 낮을 때에 수정된다. 이 기법은 접근되는 데이터 항목의 리스트가 도달하는 트랜잭션에 의해 스케줄러에게 통보된다고 가정한다. 트랜잭션이 종료될 때, 그 트랜잭션의 우선순위를 가졌던 각 데이터 항목은 그 항목에 접근하려 하는 나머지 트랜잭션들 중 가장 높은 우선순위를 가진 트랜잭션의 우선순위로 수정되어, 이것이 고유의 우선순위로 바뀐다. 또한, 각 트랜잭션은 유일한 우선순위를 갖는다고 가정한다.

〈표 3-1〉은 DPLP에서의 잠금 요청 처리 과정의 내용을 보여준다. 하나의 트랜잭션 리스트는 각각의 데이터 항목에 대하여 유지된다. 이에는 특정 데이터 항목에 접근을 요청하는 트랜잭션의 고유번호와 우선순위가 포함되어 있다. 이 리스트는 트랜잭션 우선순위에 기반하여 정렬되어 있으며, 가장 높은 우선순위의 트랜잭션이 데이터 항목의 우선순위를 결정한다. 이 리스트는 초기화시와 관련 트랜잭션이 실행 또는 포기될 때에 스케줄러에 의해 수정된다. 트랜잭션 리스트가 비어 있는 데이터 항목에는 어떤 트랜잭션에 배정될 수 있는 우선순위 값보다 낮은 값이 배정된다.

〈표 3-1〉 DPLP 기법 : 잠금 처리 요청 과정
〈Table 3-1〉 lock request processing in DPLP

```
lock_request_handling(D,T) {
/* Transaction T requests a lock on data item D */
if(priority(T) = priority(D)) {
if (D was locked by a transaction T' )
T' is aborted;
Lock on D is granted to T;
}
otherwise
T is blocked by the transaction that determines the current priority D;
}
```

〈표 3-2〉에 제시한 절차는 트랜잭션의 잠금 요청이 DPLP 기법에 의해서 처리되는 과정을 보여 준다. 트랜잭션 T가 데이터 항목 D에 잠금을 요청한다고 하자. 그 잠금을 얻기 위해서는 트랜잭션 T의 우선순위가 데이터 항목 D의 우선순위와 같아야 한다. 즉, D의 현재의 우선순위에 책임을 질 수 있는 트랜잭션이 되어야 한다는 것이다. T의 우선순위가 D의 우선순위보다 낮으면, T는 블럭된다.

데이터 항목 D는, 접근 리스트에 D를 포함하면서도 D의 우선순위보다 높은 우선순위를 갖는 새로운 트랜잭션 T′가 시스템에 도달해서 D의 우선순위를 수정할 때에, 트랜잭션 T′에 의해서 잠금되어질 수 있다. 만약 T가 D에 접근할 필요가 있을 시점에 D가

〈표 3-2〉 DPLP 기법 : 초기화 및 트랜잭션 종료시 데이터 우선순위 처리 과정

〈Table 3-2〉 Protocol DPLP : data priority processing for transaction initiation and termination

```
data_priority_handling(T) {
/* Transaction T is being initialized or terminated */
if T is a new transaction being initialized
  for each data item D in T's access list
        if(priority(T) > priority(D))
              priority(D) = priority(T);
otherwise /* T is being committed or aborted */
  for each data item D accessed by T
        if(priority(T) = priority(D))
              priority(D) = priority(highest priority
                          active transaction which will access D);
}
```

T′에 의해 잠금되어 있다면, T의 우선순위가 T′의 우선순위보다 높은 경우에 낮은 우선순위의 트랜잭션 T′는 포기되어지고, T가 D에 잠금을 얻는다. 각 트랜잭션이 유일한 우선순위를 갖는다는 가정은, 높은 우선순위의 트랜잭션은 결코 낮은 우선순위의 트랜잭션에 의해 블럭되지 않기 때문에, DPLP 기법은 트랜잭션들간의 교착상태를 방지한다.

DPLP 기법은 읽기/쓰기 잠금 의미론으로 확장될 수 있다. 이러한 확장에 대하여, 각 데이터 항목은 두 개의 우선순위 값, 하나는 읽기 접근에 대한 값, 하나는 쓰기 접근에 대한 값에 연관되어 있다. 어떤 데이터 항목 D에 읽기 잠금을 얻기 위해서는, 트랜잭션 T의 우선순위 값이 D의 쓰기 우선순위 값보다 크거나 같아야 한다. 데이터 항목 D에 대한 트랜잭션 T의 쓰기 잠금 요청은, T의 우선순위가 D의 쓰기 잠금 우선순위와는 같으며 D의 읽기 우선순위보다는 크거나 같은 경우에 받아들여진다. 트랜잭션 T가 D에 대한 쓰기 잠금을 얻을 때, 만약 D가 트랜잭션 그룹에 의

〈표 3-3〉 DPLP 기법에서 읽기/쓰기 요청을 처리하는 절차

〈Table 3-3〉 Read/Write request processing in DPLP

```
Handling READ LOCK requests
        read_lock_request_handling(D,T) {
         /* Transaction T requests a read lock on data item D */
         if(priority(T) >= write-priority(D)) {
           if(D was write-locked by a transaction T')
             T' is aborted;
           Read lock on D is granted to T;
         }
         otherwise
           T is blocked by the transaction that has assigned the write priority of D;
        }

        write_lock_request_handling(D,T) {
        /* Transaction T requests a write lock on data item D */
         if(priority(T) = write-priority(D) and priority(T) >= read-priority(D)) {
           if(D was read or write locked by any  transaction T')
             T' is aborted;
           Write lock on D is granted to T;
         }
         otherwise
           T is blocked by the transaction that has assigned
           the maximum of read and write priorities of D;
        }
```

해 이미 쓰기 잠금되어 있다면, 읽기 잠금 그룹에 있는 모든 트랜잭션들은 포기된다. 읽기/쓰기 요청을 처리하는 절차가 〈표 3-3〉에 상세히 표현되어 있다.

3.2 데이터 우선순위 기반형 시간소인 순서화 기법 (DPTOP : Data-Priority-Based Timestamp-Ordering Protocol)

기본적인 시간소인 순서화 기법에서, 충돌연산을 위한 순서화 의사결정은 모두 시작시점에서 트랜잭션에 배정되는 시간소인 값에 근거를 두고 있다. 각 트랜잭션에는 시스템에 진입(또는 재진입)하는 시점에 근거하여 시간소인이 배정된다. 우선순위가 부여된 트랜잭션 실행 환경에서, 트랜잭션 T의 시간소인보다 높은 시간소인을 지닌 보다 낮은 우선순위의 트랜잭션 T′이 그 데이터 항목에 이미 접근 했을 때에, 높은 우선순위의 트랜잭션 T는 그 데이터 항목에 대한 접근에서 포기될 수도 있다. 이러한 문제를 시간소인 순서화 기법에서의 우선순위 바뀜현상이라고 한다.

본 논문에서는 이러한 우선순위 바뀜 현상을 극복하기 위한 새로운 기법을 제시한다. 순서화 과정에서 트랜잭션의 실시간 우선순위를 이용하는 한가지 가능한 방법은 시간소인 배정절차에 실시간 제약조건을 포함하는 것이다. 본 논문에서 제시하는 DPTOP 기법은 트랜잭션들을 그들의 진입 시점에 근거해서 시간소인 그룹으로 범주화한다. 시간소인을 어떤 길이의 시간간격으로 나누고, 같은 시간 구간에 시스템에 진입하는 트랜잭션들을 같은 시간소인 그룹에 위치시킨다. 기본적인 아이디어는 그들의 실시간 우선순위에 따라 같은 시간소인 그룹의 트랜잭션들을 순서화하는 것이다.

각 트랜잭션은 그룹 시간소인과 실시간 시간소인으로 구성된 2 수준의 시간소인을 배정받는다. 같은 시간소인 그룹내에 있는 트랜잭션들은 그 그룹에서의 첫번째 트랜잭션의 진입시간과 같은 그룹 시간소인을 배정받는다. 같은 그룹내 트랜잭션들의 실시간 시간소인은 트랜잭션들의 실시간 우선순위에 기반해서 결정된다. 가장 높은 우선순위의 트랜잭션이 가장 빠른 실시간 시간소인을 얻는다. 그래서 그것은 데이터 접근 충돌의 경우에 같은 그룹에 있는 어떤 다른 트랜잭션에 의해서도 포기되지 않는다. 서로 다른 그룹으로부터의 트랜잭션들을 순서화할 때 그룹 시간소인을 이용하는 반면에, 같은 그룹으로부터의 트랜잭션들의 접근 요청을 순서화할 때에는 실시간 시간소인을 이용한다. 어떤 2개의 시간소인 그룹사이에서, 먼저 형성된 것이 보다 더 빠른 값의 그룹 시간소인을 갖는다. 서로 다른 그룹에 속한 트랜잭션들에 의한 데이터 항목 접근은, TOP에서 처럼 우선순위를 고려하지 않고, 트랜잭션이 시스템에 도착한 시간에 기반해서 순서화된다.

각 데이터 항목은 그 항목에 대한 접근을 수행하는 가장 최근 트랜잭션의 그룹 시간소인과 실시간 시간소인 모두와 관계되어 있다. DPTOP 기법하에서 트랜잭션들의 접근요청이 어떻게 처리되는지를 〈표 3-4〉에 정리하였다. 트랜잭션 T가 데이터 항목 D에 접근을 요청하고 있다고 하자. 이 경우, 트랜잭션 T의 그룹 시간소인이 데이터 항목 D의 시간소인과 비교된다. 만약 트랜잭션 T가 보다 더 빠른 그룹 시간소인을 갖고 있다면 그 접근요청은 받아들여지고, 데이터 항목 D의 그룹 시간소인과 실시간 시간소인은 수정된다. 반대의 경우에는 트랜잭션 T가 포기된다. 만약 둘 다 같은 그룹 시간소인을 갖고 있다면, 실시간 시간소인 값들이 비교된다. 트랜잭션 T가 성공적이기 위해서는 보다 더 빠른 실시간 시간소인을 가져야 한다. 그렇지 않으면 트랜잭션 T는 포기된다. 포기된 트랜잭션은, 다시 시작될 때에, 새로운 그룹 시간소인과 실시간 시간소인을 얻는다. 같은 시간소인 그룹의 트랜잭션들이 데이터 충돌을 일으킬 경우 DPTOP 기법은 TOP 기법과는 다르게 처리한다.

읽기/쓰기 접근 의미론은 각 데이터 항목에 대한

〈표 3-4〉 DPTOP 기법에서의 데이터 접근 처리과정
〈Table 3-4〉 data access handling process in DPTOP

```
data_access_handling(D,T) {
    /* Transaction T requests to access data item D */
    if(group_timestamp(T) > group_timestamp(D)) {
        T accesses D;
        group_timestamp(D)=group_timestamp(T);
    }
    else if(group_timestamp(T)=group_timestamp(D)
        & real_time_timestamp(T) > real_time_timestamp(D)) {
        T accesses D;
        real_time_timestamp(D)= real_time_timestamp(T);
    }
    else {
        T is aborted;
    }
```

읽기와 쓰기 접근 둘다의 최대 그룹/실시간 시간소인을 유지함으로서 기법에 단순히 포함될 수 있다. 읽기 요청에 대하여, 요청하는 트랜잭션의 그룹/실시간 시간소인은, 쓰기 요청에 대한 것과 마찬가지로, 데이터 항목의 그룹/실시간 쓰기 시간소인에 대비하여 체크되며, 트랜잭션 시간소인은 그 데이터 항목의 읽기/쓰기 시간소인 모두에 대하여 체크된다.

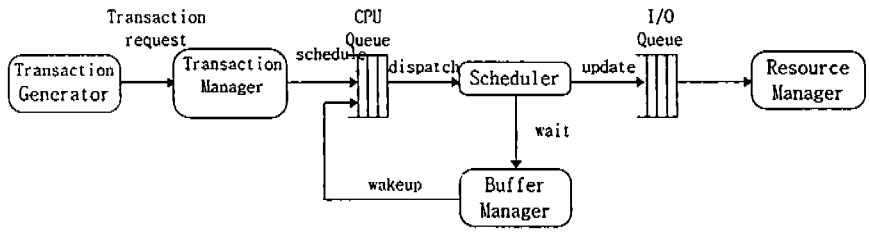
트랜잭션들에 대한 시간소인 그룹을 구성함에 있어 주요 이슈는 시간간격(Δ)의 선택이다. DPTOP 기법의 성능에 대한 시간간격 값의 효과는 연구되어 왔으며, 그 결과는 제 V 장에 제시한다. 시간간격 값은 너무 작아도 안 되고, 너무 커도 안 된다. 시간간격값이 너무 작으면 DPTOP 기법은 TOP 기법과 유사하다. 즉, 각 트랜잭션 그룹은, 트랜잭션 우선순위가 순서화 의사결정에 거의 이용되지 않는 상황으로 유도되면서, 보통 하나의 트랜잭션만을 포함한다. 시간간격 값이 너무 크면, DPTOP 기법은, 트랜잭션 우선순위에 근거해서 해결된 데이터 충돌의 갯수가 보다 더 큰 그룹 크기와 더불어 증가한다고 하더라도, TOP 기법보다 더 나쁜 성능을 나타내는 것으로 판명되었다. 이러한 결과에 대한 하나의 이유는 소위 말하는 "궁핍 현상" 즉, 만약 어떤 트랜잭션이 재시작될 때에도 전과 같은 트랜잭션 그룹에 위치한다면(그래서 전과 똑같은 시간소인을 배정받는다면) 같은 데이터 항목에 대한 매번의 시도에서 그 트랜잭션은 반복적으로 포기된다는 것이다. TOP 기법보다 성능이 나쁜 또다른 이유는 그룹핑 아이디어의 결과로서 트랜잭션 포기가 불필요하게 발생하는 것이다.

4. 시뮬레이션 모델 및 환경

4.1 시뮬레이션 모델 구성

트랜잭션 순서화 기법의 성능을 평가하기 위한 시뮬레이션 모형은 5개의 구성요소, 즉 트랜잭션 발생기, 트랜잭션 관리기, 스케줄러, 버퍼 관리기, 자원관리기로 구성하였다. 이들의 관계를 (그림 4-1)에 나타내었다.

트랜잭션 발생기는 트랜잭션의 도달(진입)을 시뮬레이션하고, 시스템 파라미터 값을 이용하여 각 트랜잭션의 형태, 마감시간, 데이터 접근 리스트를 정한다. 트랜잭션 관리기는 트랜잭션 고유번호를 발생시키는 역할과 트랜잭션들에게 실시간 우선순위를 부여하는 역할을 수행한다. 시스템에 진입된 각 트랜잭션은 마감시간 형태의 실시간 제약조건에 연관되어 있다. PCLP 기법과 DPLP 기법을 제외한 모든 트랜잭션 순서화 기법들에 대해서는, 순서화 의사결정에 이용하기 위한 유일한 정보로서 마감시간이 이용되는 데, 이는 시스템에 진입하는 트랜잭션에 의해 제공된다. 각 트랜잭션은 고유의 마감시간에 의해서 유일한 우선순위를 부여받는다. 별다르게 지정되지 않는 한, 가장 급한 마감시간을 가진 트랜잭션에 우선순위를 주는 배정방식 즉, 보다 급한 마감시간을 가진 트랜잭션이 그렇지 않은 트랜잭션보다 높은 우선순위를 갖는 방식을 채택한다. 같은 마감시간을 갖는 2개의 트랜잭션에 대해서는 시스템에 좀 더 일찍 진입한 트랜잭션이 보다 높은 우선순위를 배정받는다. 트랜잭션 마감시간은 신축적이다. 즉, 각 트랜잭션은 자기의 마감시간을 넘긴다고 하더라도, 완성될 때까지 수행된다. 각 트랜잭션은 특정한 데이터 항목에 대하여 하나 이상의 데이터베이스 연산(읽기/쓰기)을 수행한다.



(그림 4-1) 시뮬레이션 모델및 환경
(Fig 4-1) simulation model and environment

트랜잭션들의 동시적 데이터 접근 요청은 스케줄러에 의해 블럭된다. 스케줄러는 수행되는 기법에 근거해 데이터 접근을 순서화한다. 트랜잭션의 접근 요청은 트랜잭션의 실시간 우선순위에 기반하여, 승낙, 블럭, 혹은 포기 된다. 접근 요청이 승낙될 경우, 트랜잭션은 주기억장치에서 해당 데이터 항목을 읽으려 한다. 데이터가 주기억장치에 없다면, 트랜잭션은 데이터가 디스크로부터 주기억장치로 이전될 때까지 기다린다. 디스크와 주기억장치 사이의 데이터 이전은 버퍼 관리기에 의하여 이루어진다. 저장장치 버퍼의 관리를 위해서는 선입선출 페이지 대체 전략(FIFO page replacement strategy)이 이용된다. 쓰기 연산은 수정된 데이터를 디스크에 쓰기 위한 입출력 활동을 제외하고는 읽기 연산과 유사하게 처리된다. 만약 트랜잭션이 수행중 하나 이상의 데이터 항목을 수정하였다면, 수정사항을 데이터베이스에 쓰기 위해서 입출력 대기행렬에 들어간다.

자원 관리기는 데이터 항목의 읽기/쓰기를 위해서는 입출력 서비스를, 데이터 항목을 처리하고 동시성 제어 연산(충돌 체크, 잠금 등)을 수행하기 위해서는 중앙처리장치 서비스를 제공하는 역할을 한다. 중앙처리장치 행렬과 입출력 행렬은 둘 다 트랜잭션의 실시간 우선순위에 기반하여 구성되어 있다. 선점 예약 우선순위 순서화는 중앙처리장치에 의해 이용된다. 즉, 보다 높은 우선순위의 프로세스는 보다 낮은 우선순위의 프로세스를 선점하며, 보다 낮은 우선순위의 프로세스는 중앙처리장치를 기다리는 보다 높은 우선순위의 프로세스가 없을 때에 다시 시작될 수 있다. 선점 현상 이외에도, 중앙처리장치는 잠금 충돌의 결과로서 혹은 입출력을 위한 트랜잭션에 의해서 풀릴 수 있다.

4.2 시뮬레이션 환경

시스템 형상과 작업부하를 지정하기 위한 변수들은 다음과 같이 정의하였다. 데이터베이스의 크기(db_size)는 데이터베이스에 저장되어 있는 데이터 항목의 갯수에 관련되며, 메모리의 크기(mem_size)는 주기억장치에 유지될 수 있는 데이터 항목의 갯수이다. IAT(inter arrival time)는 트랜잭션들의 평균 도착 시간 간격이다. 도착률은 포아송 분포를 가정한다. 트랜잭션의 형태(읽기 또는 수정)는 수정 형태 확률을 지정하는 파라미터 tr_type_prob을 이용해서 랜덤하게 결정된다. 트랜잭션에 의해서 접근되는 데이터 항목의 갯수는 파라미터 access_mean에 의해서 결정된

〈표 4-1〉 성능 평가에 사용된 변수
〈Table 4-1〉 parameters used for performance evaluation
Configuration Parameters

Configuration Parameters		
db_size	데이타베이스 크기	200
mem_size	메모리 크기	50
cpu_time	CPU 소요 시간	12msec(constant)
io_time	IO 소요 시간	12msec(constant)
max_tr_nr	최대트랜잭션수	20
pri_assign_cost	초기배정비용	1msec(constant)
basic_op_cost	기본연산비용	0.1msec(constant)
Transaction Parameters		
iat	트랜잭션도착시간차	100msec(exponential)
tr_type_pro	트랜잭션형태비	0.5
access_mean	접근평균	6(exponential)
data_update_prob	데이타갱신비	0.5
slack_rate	여유시간비율	5(exponential)

다. 데이터 항목의 갯수의 분포는 지수분포를 이룬다. 수정 트랜잭션의 각 데이터 접근에 대하여, 데이터 항목이 수정될 확률은 파라미터 data_update_prob에 의해서 정해진다. 각 새로운 트랜잭션에 대하여 그 트랜잭션에 유일한 실시간 우선순위를 배정하는 초기의 중앙처리장치 비용이 발생한다.

이러한 처리비용은 파라미터 pri_assign_cost에 의해서 시뮬레이션된다. 트랜잭션들에 대한 중앙처리장치와 입출력 시간은 중앙처리장치-입출력장치 겹침을 달성하기 위한 목적으로 분리되어 고려될 수 있다. 하나의 데이터 항목을 처리하는 중앙처리장치 시간은 파라미터 cpu_time에 의해서 지정되며, 디스크에 있는 데이터 항목에 접근하는 시간은 파라미터 io_time에 의해서 정해진다. 이러한 파라미터들은 일정한 서비스 시간 요구를 나타낸다. 시스템에서 트랜잭션 오버헤드의 가능성을 예방하기 위해 시스템내의 능동적 트랜잭션의 총 갯수가 파라미터 max_tr_nr에 의해서 제한된다. 이러한 제한이 달성되었을 때 도달된 트랜잭션은 일시적으로 이전된다. 프로토콜들을 공정하게 평가하기 위해서는, 여러 동시성 제어 연산들을 수행하는 오버헤드가 고려되어야 한다((표 4-1)).

5. 성능 평가

실시간 데이터베이스 시스템에 근거한 시뮬레이션 모델은 트랜잭션 순서화 기법들의 실시간 성능을 평가하기 위함이다. 프로그램은 주기억장치에 상주하고 있는 데이터 항목들의 리스트를 추적한다. 수행된 시뮬레이션의 각 런은 500 트랜잭션이 수행될 때까지 계속되었다. 시뮬레이션 결과를 검증하기 위하여 독립적 반복 방법을 이용하였는 데, 이는 각 형상 파라미터를 서로 다른 난수 초기화 변수로 25회씩 런한 후, 복제 평균의 평균치를 최종 추정치로 활용하는 것이다. 독립적 관찰의 가정하에 성립된 결과에 대하여 95%의 신뢰구간을 구하였다. 다음의 절에서는 오로지 통계적으로 유의한 성능 결과를 논한다. 각 통계적 데이터 포인트의 신뢰구간의 넓이는 점추정치의 3% 이하이다. 그려진 그래프에는 성능결과들의 평균값만을 그래프에 표시(plotting) 하였다.

성능 평가 기준으로는 성공비율(success-ratio) , 평균 지연(average lateness), 충돌 비율(conflict ratio), 재시작 비율(restart ratio) 을 이용하였다. 성공 비율은 마감시간 제약조건을 만족시킨 트랜잭션 수와 총 트랜잭션 수와의 비율이며, 평균 지연은 마감시간을 지키지 못한 트랜잭션들의 평균 지연 시간이며, 충돌 비율은 관찰된 충돌의 수와 처리된 트랜잭션 수와의 비율이며, 재시작 비율은 관찰된 재시작 트랜잭션 수와 처리된 트랜잭션의 수와의 비율이다.

5.1. 트랜잭션 도착시간 차(IAT : inter arrival time)의 변화에 따른 성능 평가

5.1.1. 잠금 기법의 경우

본 시뮬레이션에서의 결과를 분석해 보면, PILP 기법이 ABLP 기법보다는 상당한 진전이 있다는 사실을 알게 된다. 이러한 진전은 우선순위 상속 방법을 적용하여 높은 우선순위 트랜잭션들의 블럭시간을 줄인 때문이다. 그러나, PILP 기법의 성능은 PBLP 기법의 성능 수준에는 미치지 못한다. PBLP 기법은 보다 높은 우선순위 트랜잭션들을 결코 블럭하지 않으며, 필요할 때는 낮은 우선순위 트랜잭션들을 포기한다는 사실이다. PBLP 기법은 또한 교착상태의 가능성과 비용을 제거한다. 본 논문에서 제안하는 DPLP 기법의 성능은, 특히 트랜잭션 부하가 높을 경우, PBLP 기법의 성능보다 우수하다. PBLP 기법에서 시뮬레이션된 얼마간의 트랜잭션 포기와 그에 따르는 자원 낭비는 접근 순서화시 자료 요구사항에 대한 사전 지식을 채택하는 DPLP 기법에 의해서 예방된다. DPLP 기법에서는 어떤 두개의 충돌하는 트랜잭션 사이에서 낮은 우선순위의 트랜잭션은 높은 우선순위 트랜잭션이 시스템에 진입하기 이전에, 그 데이터 항목의 우선순위가 높은 우선순위 트랜잭션의 진입에 의해서 수정되기 이전에, 그 데이터 항목을 접근하는 경우에 한하여 포기의 가능성을 제한한다. 이러한 특별한 경우에 대해서, 낮은 우선순위 트랜잭션의 수행 이전에 높은 우선순위 트랜잭션이 그 데이터 항목을 접근할 때 그리고 접근한다면, 낮은 우선순위 트랜잭션은 포기된다. 결론적으로, 잠금 기법의 경우에 제안하는 DPLP 기법은 기존의 기법들보다 본 시뮬레이션 환경하에서는 성능이 우수함을 이미 확인한 바 있다[2].

5.1.2 시간소인 순서화 기법과 낙관적 기법의 경우

데이터 우선순위 기반형 시간소인 순서화 기법 DPTOP의 성능을 다른 기법들의 성능과 비교하기 위한 시뮬레이션을 수행하기 이전에 시간간격(Δ)에 대한 합리적인 값을 선정하기 위한 일련의 테스트를 수행하였다. TOP에 대한 DPTOP의 성능 향상은 시간간격과 트랜잭션 도착 시간차에 관계가 많기 때문에 이에 대한 영향을 조사하였다.

시간간격이 크면 같은 그룹 시간소인내에 포함되는 트랜잭션의 갯수가 많아지기 때문에 시간소인에 따라 그룹핑하는 의미가 없어져서 TOP 기법보다 성능이 좋지 않을 것이다. 반대로, 시간간격이 작으면, 같은 그룹 시간소인내에 포함되는 트랜잭션의 갯수가 작아지기 때문에 트랜잭션 및 데이터 항목의 실시간 우선순위와는 관계없이 거의 실시간 시간소인에 의해서만 순서화 되므로 TOP 기법과 거의 같은 결과를 준다. 실제로 시뮬레이션을 수행해 본 결과, 시간간격이 작은 경우에는 트랜잭션 도착 시간차에 관계없이 TOP에 대한 DPTOP의 성능 향상이 없었다. 시간간격이 2000msec가 되면 DPTOP보다는 TOP의 성능이 좋은 것으로 나타났다. 시간간격이 커질 경우, TOP에서는 트랜잭션의 우선순위에 관계없이 시간소인에 의해서만 순서화 하는 반면, DPTOP에서는 그룹내에서의 실시간 우선순위에 따라 순서화하기 때문에 TOP의 성능이 좋은 것이다. 트랜잭션 도착 시간차가 큰 경우에도 시간간격에 관계없이 TOP에 대한 DPTOP의 성능 향상이 없었다.

TOP에 대한 DPTOP의 성능 향상은 시간간격이 100-1000 msec이며, 트랜잭션 도착 시간차가 75-115 msec일때 달성되었다. 〈표 5-1〉는 백분율에 의한 성공비율의 관점에서, 시간간격 값의 범위에 대하여 TOP 기법에 대한 DPTOP 기법의 향상도를 보여 준다. 각 시간간격 값은 트랜잭션 부하의 서로 다른 수준에 대응하는 서로 다른 트랜잭션 도착 시간차 값에 대하여 테스트 하였다. 파라미터 설정에 대하여 DPTOP 기법의 가장 좋은 성능 결과는 시간간격 값이 500msec이고, 트랜잭션 도착 시간차가 75msec일때 달성되었다.

시간소인 순서화 기법은 시뮬레이션에서 실험한 대부분의 파라미터 범위에 대하여 다른 순서화 기법들과 비교할 때 성능이 좋지 않았다. 본 논문에서 제안한 데이터 우선순위 기반형 시간소인 순서화 기법(DPTOP : Data Priority-Based Timestamp-ordering Protocol)을 채택함으로서 TOP 기법에 실시간 우선순위를 포함하는 효과를 조사하였다. 본 논문에서 관찰한 것은 TOP 기법의 실시간 제약조건을 트랜잭션들에게 시간소인 배정시 실시간 우선순위로 이용함으로서 특히 높은 부하와 높은 데이터 충돌 조건하에서 성능을 향상시키는 것이 가능하다는 점이다.

〈그림 5-1〉 트랜잭션 도착 시간차와 시간간격(Δ)에 따른 DPTOP의 성능향상(%)

〈Table 5-1〉 performance improvement of DPTOP depending upon IAT and time interval(%)

시간간격(Δ) / 도착 시간차	20	50	100	250	500	750	1000	2000
75	0	0	5.5	10.9	12.7	9.1	1.8	-3.6
85	0	0	2.7	5.5	6.9	6.9	1.4	-1.4
95	0	0	1.2	3.7	4.9	3.7	0	0
105	0	0	1.2	3.5	3.5	4.7	3.5	-1.2
115	0	0	0	1.1	1.1	2.2	1.1	-1.1
125	0	0	0	0	0	0	0	-1.1
135	0	0	0	0	0	0	0	0

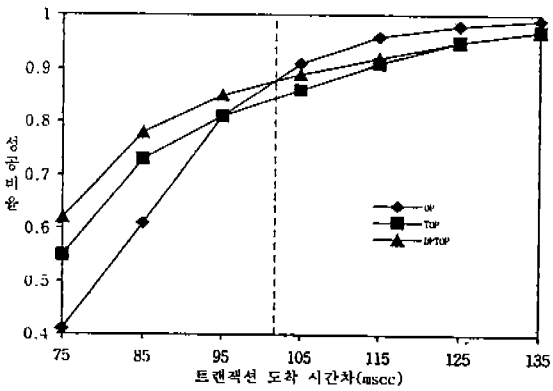
또한 DPTOP 기법은 높은 수준의 트랜잭션 충돌 상황에서 성능이 급격히 떨어지는 시간소인 순서화 기법 및 낙관적 기법에 비해 트랜잭션 충돌 수준의 변화에 따른 일정한 성능을 확인하였다.

(그림 5-1)과 (그림 5-2)은 시간소인 순서화 기법과 낙관적 기법의 성능 결과를 함께 보여준다. 낙관적 기법은 트랜잭션의 도착시간차 값이 큰 경우에 즉, 시스템에 약간의 부하가 있을 때에 매우 좋은 성능을 보여준다. 그러나, 부하 수준이 높을 경우에는, OP 기법의 성능이 다른 기법보다 못하다. 이는 트랜잭션 사이의 충돌 갯수가 항상 늘어남으로 인하여 재시작의 갯수가 증가함에 의하여 설명될 수 있다. 재시작의 갯수가 큰 것에 기인한 낭비된 수행시간은 실질적으로 마감시간에 늦은 갯수와 늦은 트랜잭션의 도착지연 갯수를 증가시킨다.

TOP 기법은 시스템의 부하가 높을 때 OP 기법보다 성능이 좋다. 그러나, 낮은 부하 수준하에서는 상황이 달라진다. 제안하는 DPTOP 기법은 순서화 의사결정시 실시간 우선순위를 포함함으로서 TOP 기법의 실시간 성능을 개선하기 위하여 설계되었다. 높은 트랜잭션 부하에 대해서, DPTOP 기법은 TOP 기법의 성능에 대하여 상당한 개선을 제공한다.

만약 우리가 모든 형태의 기법들을 함께 비교한다면, DPLP 기법이 모든 기법 중에서 가장 성능이 좋다. 높은 트랜잭션 부하하에서 시간소인 순서화 기법 DPTOP의 성능은 DPLP 기법의 성능과 비슷하다. 낙관적 기법은 낮은 트랜잭션 부하의 경우에서만 성능이 좋다.

이하에서는 성능이 가장 우수한 3가지 기법(DPLP, DPTOP, OP)에 대한 성능 평가 결과를 제시한다. DPLP 기법은 잠금 기법 중에서, DPTOP 기법은 시간소인 순서화 기법중에서 다른 기법들보다 여러 기준에서 성능이 우수한 것으로 확인되었기 때문이다.



(그림 5-1) 시간소인및 낙관적 기법의 트랜잭션 도착 시간차 변화에 따른 성공비율

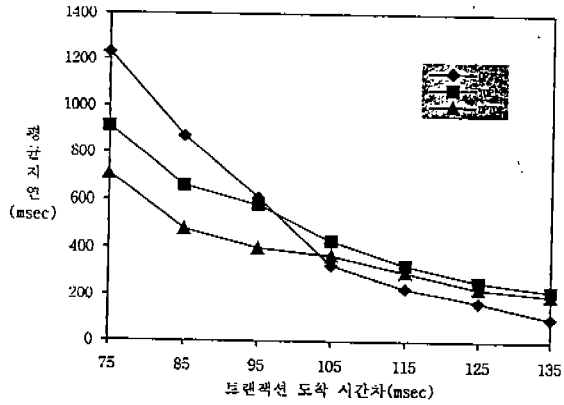
(Fig 5-1) success ratio for IAT of timestamp ordering and optimistic protocol

5.2 데이터베이스 크기 변화에 따른 성능 평가

본 논문에서 제안하는 기법의 시뮬레이션시 데이터베이스의 크기는 앞 장에서 서술된 모든 시뮬레이션에서 200개의 데이터 항목으로 고정되었다. 매우 작은 데이터베이스 크기를 가정하는 목적은 트랜잭션들 사이의 높은 데이터 충돌 횟수를 구하기 위함이다. 데이터베이스의 약 3%가 각 트랜잭션에 의해 접근되어 여러 충돌 해결 전략의 상대적 효율성을 관찰하기에 충분히 높은 충돌 비율을 제공하였다. 또한 우리의 실험에서 이용된 작은 데이터베이스는 보다 더 큰 데이터베이스의 접근이 잦은 부분(hot spot)로서 고려된다.

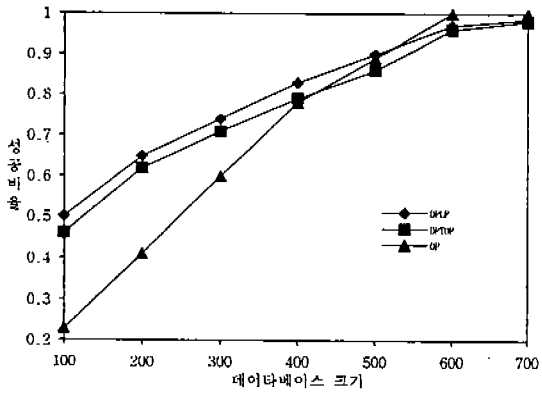
서로 다른 수준의 데이터 충돌에서 기법들의 성능을 보다 완전하게 연구하기 위해서, 데이터베이스의 크기(db_size)는 100-1000 데이터 항목의 범위에서 변화되었다. 이 값들은 각 트랜잭션에 의해 접근되는 데이터베이스의 6%-0.6%까지에 대응된다.

(그림 5-3)은 DPLP, DPTOP, OP 기법에 의한 성공비율 결과를 보여 준다. 데이터베이스가 클 경우, 트랜잭션들은 어떠한 순서화 기법하에서도 마감시간을 만족시키는 데에 문제가 없다. 각 트랜잭션의 데이터 접근이 데이터베이스에 대하여 균등하게 분포되어 있으므로, 동시적 트랜잭션들의 접근 집합들은, 많은



(그림 5-2) 시간소인및 낙관적 기법의 트랜잭션 도착시간차 변화에 따른 평균 지연

(Fig 5-2) average lateness for IAT of timestamp ordering and optimistic protocols



(그림 5-3) 시간소인및 낙관적 기법의 데이터베이스 크기 변화에 따른 성공 비율

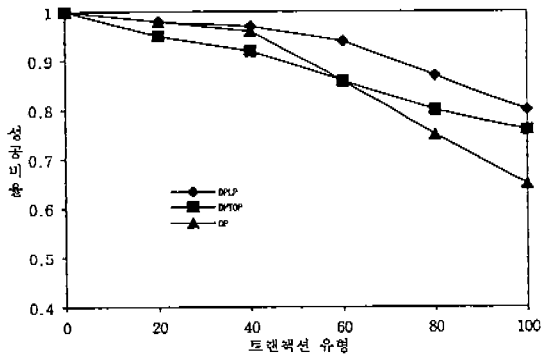
(Fig 5-3) success ratio for DB size of timestamp ordering and optimistic protocols

수의 데이터 항목이 구축되어 있는 데이터베이스에 저장되어 있을 때, 즉, 시스템의 데이터 충돌 수준이 낮을 때, 많은 공통의 항목을 갖지 않는다.

5.3 트랜잭션 유형의 변화에 따른 성능 평가

이번 시뮬레이션에서는, 읽기만 하는 트랜잭션(read-only)와 수정 트랜잭션의 서로 다른 조합하에서 기법들의 성능을 조사하기 위하여, tr_type_prob는 0.0에서 1.0까지 0.2의 단계로 변화하였다. 수정 트랜잭션의 작은 비율에 비해서, 모든 프로토콜들의 성능은 매우 유사하다. 충돌 비율이 낮기 때문에, 대부분의 트랜잭션들은 실시간 제약조건을 만족한다. 트랜잭션들 사이의 데이터 충돌 횟수는 쓰기 연산의 수가 증가함에 따라, 쓰기 연산은 읽기와 쓰기 연산 모두와 충돌하기 때문에, 증가한다.

많은 수의 수정 트랜잭션이 실시간 성능에 주는 또 하나의 효과는 수정된 데이터 항목을 디스크에 쓰는 입출력의 무거운 부하에 기인한 트랜잭션 평균 수명 시간의 증가이다. (그림 5-4)은 DPLP, DPTOP, OP에 대한 평균 수정 트랜잭션 백분율의 함수로서 성공 비율을 보여 준다. DPLP 기법의 성능은 대부분의 트랜잭션이 수정일 때 DPTOP 기법의 성능보다 약간 좋다. 낙관적 기법은 많은 수의 수정이 있을 때에, 자료 충돌의 수가 증가하기 때문에, 다른 기법보다 성능이 나쁘다.



(그림 5-4) 시간소인및 낙관적 기법의 트랜잭션 유형 변화 대 성공비율

(Fig 5-4) success ratio for transaction type of timestamp ordering and optimistic protocols

6. 결 론

시간간격이 작은 경우에는 트랜잭션 도착 시간차에 관계없이 TOP에 대한 DPTOP의 성능 향상이 없었다. 시간간격이 2000msec가 되면 DPTOP보다는 TOP의 성능이 좋은 것으로 나타났다. 시간간격이 커질 경우, TOP에서는 트랜잭션의 우선순위에 관계없이 시간소인에 의해서만 순서화하는 반면, DPTOP에서는 그룹내에서의 실시간 우선순위에 따라 순서화하기 때문에 TOP의 성능이 좋은 것이다. 트랜잭션 도착 시간차가 큰 경우에도 시간간격에 관계없이 TOP에 대한 DPTOP의 성능 향상이 없었다.

TOP에 대한 DPTOP의 성능 향상은 시간간격이 100-1000 msec이며, 트랜잭션 도착 시간차가 75-115 msec일때 달성되었다. 각 시간간격 값은 트랜잭션 부하의 서로 다른 수준에 대응하는 서로 다른 트랜잭션 도착 시간차 값에 대하여 테스트한 결과, DPTOP 기법의 가장 좋은 성능 결과는 시간간격 값이 500msec이고, 트랜잭션 도착 시간차가 75msec일때 달성되었다.

시간소인 순서화 기법은 시뮬레이션에서 실험한 대부분의 파라미터 범위에 대하여 다른 순서화 기법들과 비교할 때 성능이 좋지 않았다. 본 논문에서는 제안하는 데이터 우선순위 기반형 시간소인 순서화 기법(DPTOP : Data-Priority-Based Timestamp-ordering Protocol)을 채택함으로서 기존의 시간소인 순서화 기법에 실시간 우선순위를 포함하는 효과를 조사하였다. 시뮬레이션 결과, TOP 기법의 실시간 제약조건을 트랜잭션들에게 시간소인 배정시 실시간 우선순위를 이용함으로서 특히 높은 부하와 높은 데이터 충돌 조건하에서 성능을 향상시키는 것이 가능함을 확인하였다.

본 논문에서 제안하는 트랜잭션 순서화 기법들은 여러 응용을 위한 시스템 구축시의 데이터베이스 관리 시스템에 그대로 적용되어 시스템의 성능 향상을 도모할 수 있을 것이며, 분산 환경에서 실시간 온-라인 트랜잭션 처리를 해야 하는 경우에도 활용이 가능할 것이므로 이에 대한 지속적인 연구가 필요하다고 본다.

참 고 문 헌

[1] 윤석환, 이재영, 박치항, 신용백, "On the Concurrency Control and the Interface Design for Collaboration-Aware Applications," 정보처리논문지, 한국정보처리학회, Vol. 3, No. 3, pp. 631-639, 1996.

[2] 윤석환, 이재영, 박치항, "데이터 우선순위에 기초한 트랜잭션 순서화 기법의 제안 및 실시간 DBMS에서의 성능 비교 연구," 정보처리논문지, 한국정보처리학회, Vol. 3, No. 4, pp. 830-816, 1996.

[3] R. Abbott, and H. Garcia-Molina, "Scheduling Real-Time Transactions: A Performance Evaluation," 14th International Conference on Very Large Data Bases, pp. 1-12, 1988.

[4] L. Sha, R. Rajkumar, J. Lehoczky, "Priority Inheritance Protocols: An Approach to Real-Time Synchronization," IEEE Transactions on Computers, Vol. 39, No. 9, pp. 1175-1185, 1990.

[5] L. Sha, R. Rajkumar, J. Lehoczky, "Concurrency Control for Distributed Real-Time Databases," ACM SIGMOD Record, pp. 82-98, 1988.

[6] R. Agrawal, M. J. Carey, M. Livny, "Concurrency Control Performance Modelling: Alternatives and Implications," ACM Transactions on Database Systems, Vol. 12, No. 4, pp. 609-654, 1987.

[7] H. T. Kung and J. T. Robinson, "On Optimistic Methods for Concurrency Control," ACM TODS, Vol. 6, No. 2, pp. 213-226, 1981.

[8] J. Huang, J. A. Stankovic, K. Ramamritham, D. Towsley, "Experimental Evaluation of Real-Time Optimistic Concurrency Control Schemes," 19th International Conference on Very Large Data Bases, pp. 35-46, 1991.

[9] L. Sha, R. Rajkumar, S. H. Son, C. H. Chang, "A Real-Time Locking Protocol," IEEE Transactions on Computers, Vol. 40, No. 7, pp. 793-800, 1991.

[10] K. G. Shin, "Introduction to the Special Issue on Real-Time Systems," IEEE Transactions on Computers, Vol. 36, No. 8, pp. 901-902, 1987.

[11] Y. Tay, N. Goodman, R. Suri, "Locking Performance in Centralized Databases," ACM Transactions on Database Systems, Vol. 10, No. 4, pp. 415-462, 1985.

[12] P. S. Yu, D. M. Dias, "Analysis of Hybrid Concurrency Control Schemes For a High Data Contention Environment," IEEE Transactions on Software Engineering, Vol. 18, No. 2, pp. 118-129, 1992.

[13] M. Knister , A. Prakash, "Issues in the design of a toolkit for supporting multiple group editors," Computing Systems (The Journal of the Usenix Association), Vol. 6, No. 2, pp. 135-166, 1993.

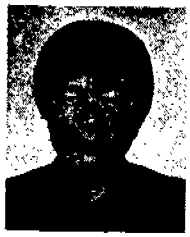
[14] P. Bernstein, V. Hadzilacos, N. Goodman, "Concurrency Control in a System for Distributed Databaseds (SDD-1)," ACM Trans. on Database Systems, Vol. 5, No. 1, pp. 18-51, 1980.

[15] P. Bernstein, V. Hadzilacos, N. Goodman, 'Concurrency Control and Recovery in Database Systems,' Reading MA, Addison-Wesley, 1987.

[16] J. Huang, J. A. Stankovic, K. Ramamritham, and D. Towsley, "On Using Priority Inheritance In Real-Time Databases," 12th Real-Time Symposium, pp. 210-221, 1990.

[17] J. R. Haritsa, M. J. Carey and M. Livny, "On Being Optimistic About Real-Time Constraints," ACM SIGACT-SIGMOD-SIGART, pp. 331-343, 1990.

[18] J. R. Haritsa, M. J. Carey and M. Livny, "Dynamic Real-Time Optimistic Concurrency Control," 11th Real-Time Systems Symposium, pp. 94-103, 1990.



윤 석 환

1982년 2월 아주대학교 산업공학과(공학사)

1984년 2월 건국대학교 산업공학과(공학석사)

1996년 8월 아주대학교 산업공학과(공학박사)

1992년 8월 품질관리 기술사 자격 취득

1995년 1월~현재 한국정보처리학회지 편집위원
1986년 1월~현재 한국전자통신연구소 책임연구원 (컴퓨터연구단 멀티미디어연구부)
관심분야:그룹웨어, S/W 공학, 생산정보시스템, 개발방법론

김 평 중

1985년 충남대학교 계산통계학과(이학사)
1995년 KAIST 전산학과(공학석사)
1995년 정보처리기술사 취득
1987년~1988년 포항종합제철(주)전산시스템부 4급
1988년~현재 한국전자통신연구소 멀티미디어연구부 선임연구원
관심분야:컴퓨터네트워크, 프로토콜 공학, 그룹웨어, 분산 멀티미디어 등

박 지 은

1991년 안동대학교 전산통계학과(이학사)
1993년 경북대학교 대학원 전자계산학과(이학석사)
1993년~현재 한국전자통신연구원 연구원
관심분야:CSCW, Visual Programming, 멀티미디어

이 재 영

1988년 2월 서울대학교 물리학과(이학사)
1990년 5월 The Johns Hopkins Univ.(이학석사)
1994년 8월 The Johns Hopkins Univ.(이학박사)
1994년 12월~현재 한국전자통신연구소 시각언어연구실
관심분야:그룹웨어, 분산처리, S/W 공학

이 동 현

1988년 서울대학교 자원공학과(학사)
1990년 과학원 전산학과(석사)
1990년~현재 한국전자통신연구소 시각언어 연구실 선임연구원
관심분야:Distributed Object-Oriented System, Object-Oriented Visual Programming, Event-based Programming, Machine Learning

궁 상 환

1977년 2월 숭실대학교 전산학과 졸업
1977년~1983년 육군 제2군수지원사령부 전산장교
1983년 고려대학교 경영대학원 전산정보 석사
1991년~현재 충북대학교 대학원 박사과정
1982년~현재 한국전자통신연구소 책임연구원(컴퓨터연구단 멀티미디어연구부 시각언어연구실)
관심분야:multicast transport, distributed system architecture등