분산시스템에서 데드락 예방을 위한 데이터 우선순위 상속 알고리즘

노치환° ,서창석, 이병욱
경원대학교 전자계산학과

Data Priority-Inheritance Algorithm for Deadlock Prevention in Distributed Systems

ChiHawan Loe°, ChangSuk Seo, ByungWook Lee
Dept. of Computer Science, Kyungwon Univ.

요 약

본 연구의 목적은 트랜잭션 순서화 기법중 데이터 우선순위 알고리즘(DPLP: Data-Priority Based Locking Protocol)의 높은 재시작 비율을 낮추고, 시스템의 처리 효율을 향상 시키는 것이다. 이를 위해서 트랜잭션을 일정한 크기의 그룹으로 그룹핑하고, 같은 그룹에 있는 트랜잭션은 선점권을 인정함으로써 자원을 점유하고 있는 트랜잭션에게 우선순위를 상속하여 트랜잭션의 재시작 비율을 낮춘다. 낮은 트랜잭션이 데이터를 장시간 잠금(lock)하는 것을 방지하기 위해서 그룹 우선순위를 적용한다. 상속에 의한 데드락 방지를 위해서 충돌 전에 다른 데이터의 잠금여부를 검사한다. 성능을 평가하기 위해서 분산 데이타 베이스 시스템으로 시뮬레이션 환경을 구축하여 기존의 기법들과 성능을 비교하였다. 성능 비교 결과는 본 논문에서 제안하는 데이터 우선순위 상속 알고리즘이 기존의 상속 알고리즘과 데이터 우선순위 알고리즘 보다 성능이 우수함을 확인하였다.

Ⅰ. 서론

최근 컴퓨터 기술의 발달로 저렴한 가격의 컴퓨터 생산으로 인한 컴퓨터 보급의 확산과 통신 기술의 발달로 컴퓨터를 고속 네트워크로 상호 연결하여 자원을 공유하는 분산 시스템 환경이 조성되고 있다. 분산 시스템 환경에서 동시에[4] 다중 트랜잭션들이 데이터베이스에 접근하려고 하면 상호충돌(conflict)을 일으키거나, 데드락 상태(deadlock)를 발생시킬 수 있다. 충돌, 데드락 상태가 발생하면 시스템은 서비스 지연이나, 무한대기 상태에 빠진다. 분산환경에서 트랜잭션을 효율적으로 처리하지 못할 경우에는 분산환경에서 효율적인 자원 공유를 할 수 없다.

자료 공유의 개념이 확산되고 있는 현재는 트랜잭션 처리 문제는 더욱더 중요한 현안 문제로 대두되고 있다. 따라서 보다 효율적인 트랜잭션 처리를 위한 연구가 요청되고 있다.

본 논문에서는 기존의 알고리즘을 기반으로 효율적인 데드락 방지를 위한 알고리즘을 제안하고, 기존 알고리즘과 제안 알고리즘을 성공 비율, 재시작 비율의 관점에서 효율을 비교, 분석한다.

Ⅱ. 분산 시스템에서 병행제어 알고리즘

분산[3] 환경에서의 컴퓨터 통신은 기본 단위인 트랜잭션을 통해서 이루어지며, 특정 클라이언트가 요구하는 내용을 서버 및 다른 클라이언트가 처리하여 다시 그 클라이언트에게 보내는 구조로 되어 있다. 트랜잭션은 특정 클라이언트에서 서버 및 다른

클라이언트에 접근하는 파일시스템(file systems) 명령어와 데이터베이스 명령어 등을 처리하는 일련의 기본 조작들의 순서적 나열이며, 데이터베이스의 현재의 일치 상태(consistent state)를 새로운 일치 상태로 변환하는 연산 행위로, 트랜잭션이 성공적으로 끝나면, 모든 수정 사항은 완료(committed)되거나 취소(abort)된다.

서버는 동시에 다수의 클라이언트로부터 트랜잭션을 받아서 처리할 수 있도록 설계, 구현되어야 한다. 이러한 다중 트랜잭션들은 서버에 의해 조절되는 일정한 순서로 처리되어야 하는데, 이를 트랜잭션 스케줄링이라 한다. 트랜잭션의 스케줄링을 위해서 서버에는 트랜잭션 스케줄러가 존재한다. 트랜잭션 스케줄러는 트랜잭션 스케줄링[4] 알고리즘(TSA: Transaction Scheduling Algorithm)에 의거하여 트랜잭션들을 순차화(serialize)하여 충돌을 해결하고 데드락상태를 방지하는 역할을 수행한다. 트랜잭션을 순차화하기 위하여 스케줄러가 사용하는 알고리즘은 잠금 알고리즘(locking)[1], 타임스탬프 순서화 알고리즘(timestamp ordering), 낙관적 알고리즘(optimistic method)등이 있으며, 잠금 알고리즘에는 항시 블록 잠금 알고리즘(ABLP : Always Block Locking Protocol)[1], 우선순위에 기반 잠금 알고리즘(PBLP : Priority Based Locking Protocol), 우선순위 상속 잠금 알고리즘(PILP: Priority Inheritance Locking Priority)[2], 데이터 우선순위 기반형 알고리즘(DPBLP : Data Priority Based Locking Protocol) 등이 있다.

2.1 우선순위 상속 잠금 알고리즘

프로토콜 ABLP의 단점은 "우선순위 바뀜"[7] 문제이다. 우선순위 바뀜 문제는 높은 우선순위의 트랜잭션이 낮은 우선순위의 트랜잭션에 의해 블록킹되는 경우를 가리키는 것으로서 프로토콜 PILP는 이 문제를 해결하기 위해서 제안된 방법이다[2]. 이 프로토콜은, 한 트랜잭션이 높은 우선순위 트랜잭션을 블록킹할 때, 자신이 블록킹한 모든 트랜잭션의 우선순위중 가장 높은 우선순위를 상속받게 하는 것이다. 상속에 의해 잠금을 가진 트랜잭션은 가장 높은 우선순위를 가지게 되어 잠금을 유지할 수 있다. 프로토콜 PILP는 높은 우선순위 트랜잭션의 블록킹 시간을 단축시키는 효율을 가지고 있다.

우선순위를 상속받은 트랜잭션이 교착상태 때문에 포기될 때에, 그 트랜잭션은 원래의 우선순위로 환원된다. 잠금를 공유하고 있는 트랜잭션 그룹이 데이터 잠금을 보유하고 있으며, 데이터 항목에 대한 충돌 때문에 높은 우선순위 트랜잭션이 블록킹되어 있다면, 공유된 잠금 그룹에 존재하면서 블록킹된 트랜잭션 보다 낮은 우선순위의 트랜잭션은 블록킹되어있는 트랜잭션의 우선순위를 상속받는다. 표 2.1은 PILP 스케줄러 함수로서 변수 D는 데이터 항목, T는 트랜잭션, TR은 잠금을 가지고 있는 트랜잭션을 나타낸다.

표 2.1 PILP 스케줄러 함수

```
lock_request (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)
     }
}
```

2.2 데이터 우선순위 잠금 알고리즘

데이터 항목에 우선순위를 부여하여 교착상태가 발생하지 않도록 트랜잭션 스케줄링을 할 수 있는 개선된 잠금 프로토콜이다. 각 데이터 항목은 해당 데이터 항목에 접근할 수 있는 모든 트랜잭션들 중 가장 높은 우선순위와 같은 우선 순위를 갖는다. 새로운 트랜잭션이 시스템에 도달하면, 각 데이터 항목의 우선순위는, 자신이 해당 데이터 항목에 접근할 수 있는 트랜잭션의 우선순위보다 낮을 때에, 수정된다. 이 프로토콜은 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 고유의 우선순위로 바뀐다.

하나의 트랜잭션 리스트는 각각의 데이터 항목에 대하여 유지되며, 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. 리스트는 초기화시, 관련 트랜잭션이 실행 또는 포기될 때에 스케

줄러에 의해 수정된다. 트랜잭션 리스트가 비어 있는 데이터 항목에는 어떤 트랜잭션에 배정될 수 있는 우선순위 값보다 낮은 값이 배정된다.

표 2.2 DPLP에서 잠금 처리 요청 함수

```
lock_request (D, T) {

  if(priority(T) = Priority(D)){
        if(D was locked by a transaction TR)
        TR is aborted;
        Lock on D is granted to T;
        }
  otherwise
        T is blocked by the transaction that
determines the current
         Priority D;
}
```

표 2.3 DPLP에서 초기화 및 트랜잭션 종료 데이터 우선순위 처리 함수

```
lock_request (T) {

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 transactio
         which will access D);
}
```

표 2.3에서 제시한 절차는 트랜잭션의 잠금 요청이 프로토콜 DPLP에 의해서 처리되는 과정을 보여준다. 트랜잭션 T가 데이터 항목 D에 잠금을 요청한다고 하자, 잠금을 얻기 위해서는 트랜잭션 T의 우선순위가 D의 우선순위와 같아야 한다. 즉, D의 현재의 우선순위에 책임을 질 수 있는 트랜잭션이 되어야 하다는 것이다. T의 우선순위가 D의 우선순위보다 작으면, T는 블록킹된다. 데이터 항목 D는, 접근 리스트에 D를 포함하면서 D의 우선순위보다 높은 우선순위를 갖는 새로운 트랜잭션 T'가 시스템에 도달해서 D의 우선순위를 수정할 때에, 트랜잭션 T'에 의해서 잠금될 수 있다. 만약 T가 D에 접근할 시점에 D가 T'에 의해 잠금되어 있다면, T의 우선순위가 T'의 우선순위보다 높은 경우에, 낮은 우선순위의 트랜잭션 T'는 포기되고, T가 D에 잠금을 얻는다. 각 트랜잭션이 유일한 우선순위를 갖는다는 가정은, 높은 우선순위의 트랜잭션은 결코 낮은 우선순위의 트랜잭션에 의해 블록킹되지 않기 때문에, 프로토콜 DPLP의 교착상태를 방지한다.

III. 데이터 우선순위 상속 알고리즘

데이터 우선순위 알고리즘의 낭비적 재시작 문제를 선택적 상속에 의해서 재시작 비율을 낮추기 위해서 기존의 우선순위 상속 알고리즘을 이용하여 데이터 우선순위 알고리즘의 재시작 비율을 낮추고, 처리율을 높이기 위해서 트랜잭션을 그룹단위로 그룹핑한다. 트랜잭션의 그룹핑 단위는 10을 하나의 단위로 그룹핑한다. 그리고, 우선순위가 같은 그룹에 있는 트랜잭션은 선택적 선점권을 허락한다.

3.1 데드락 방지

기존의 우선순위 상속 알고리즘의 단점인 데드락 문제를 해결하기 위해서 만약 우선순위 높은 트랜잭션이 먼저 다른 데이터를 선점하고 있는 경우에는 같은 그룹의 트랜잭션이라도 선점을 허락하지 않고 트랜잭션의 우선순위에 따라서 수행함으로써 우선순위 상속 알고리즘의 단점인 교착 상태를 방지한다. 그림 3.1에서 트랜잭션 T_1, T_5의 두 개의 트랜잭션이 발생하고, 트랜잭션 T_1의 우선순위는 1, 트랜잭션 T_5의 우선순위는 5라고 가정한다. 그리고 트랜잭션 우선순위 간격 10을 하나의 그룹으로 그룹핑한다. 트랜잭션 T_1이 데이터 A을 잠금하고, 트랜잭션 T_5가 데이터 B를 잠금하고 있는 상황에서 트랜잭션 T_1이 데이터 B에 접근하면 트랜잭션 충돌이 일어난다. 그때 트랜잭션 관리기를 검사하여 우선순위 높은 트랜잭션 T_1이 이미 다른 자원을 잠금 하고 있으면, 트랜잭션 T_1의 우선순위를 트랜잭션 T_5에게 상속하지 않는다. 그리고, 트랜잭션 T_5는 재시작하고, 트랜잭션 T_1이 데이터 B에 대한 잠금 권한을 가진다. 따라서 우선순위 상속 알고리즘에서 상속에 의해 야기된 트랜잭션 교착 문제를 해결한다.

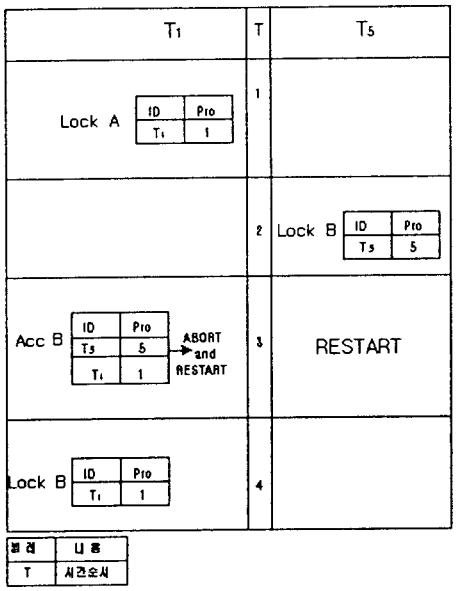

그림 3.1 데이터 우선순위 상속 알고리즘의 데드락 방지

3.2 트랜잭션의 그룹 우선순위 적용

데이터 우선순위 상속 알고리즘의 처리율을 높이기 위해서 그룹 우선순위를 적용한다. 서로 다른 그룹의 트랜잭션이 충돌할 경우에는 트랜잭션의 우선순위에 준하여 처리되게 함으로써 우선순위가 낮은 트랜잭션이 데이터를 장시간 잠금하는 것을 방지한다. 그림 3.2에서 트랜잭션 T_1의 우선순위는 1, 트랜잭션 T_{11}의 우선순위를 11라고 가정한다. 그리고 트랜잭션 우선순위 간격 10을 하나의 그룹으로 그룹핑하므로, T_1의 그룹 우선순위는 1, T_{11}의 그룹 우선순위를 2라고 가정한다. 트랜잭션 T_{11}이 데이터 B를 잠금하고, 트랜잭션 T_1은 데이터를 잠금 하지 않은 상황에서, 트랜잭션 T_1이 데이터 B에 접근하면 트랜잭션 충돌이 일어난다. 그때 트랜잭션 관리기는 스케줄러를 검사한다. 우선순위 높은 트랜잭션 T_1이 다른 자원을 잠금 하고있지 않다. 그러나 그룹 우선순위가 T_1은 그룹 우선순위가 1인데 반하여 T_{11}은 그룹 우선순위가 2이므로 트랜잭션 T_1의 우선순위를 트랜잭션 T_{11}에게 상속하지 않는다. 그리고 트랜잭션 T_{11}는 재시작 하고 트랜잭션 T_1이 데이터 B에 대한 잠금 권한을 가진다. 따라서 낮은 그룹의 트랜잭션이 장시간 데이터를 잠금하는 것을 방지함으로써 시스템 처리율을 높일 수 있다.

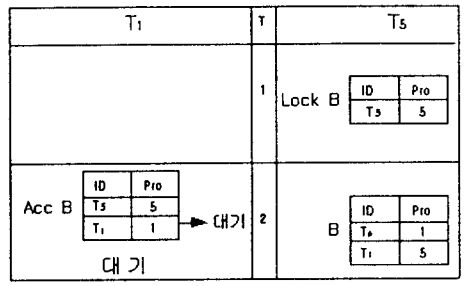

그림 3.2 데이터 우선순위 상속 알고리즘의 그룹 우선순위

3.3 트랜잭 우선순위 상속

동일 그룹의 트랜잭션이 충돌한 경우 우선순위 높은 트랜잭션이 먼저 다른 데이터에 대한 잠금을 하고 있지 않으면 선점을 허락하여 우선순위를 상속

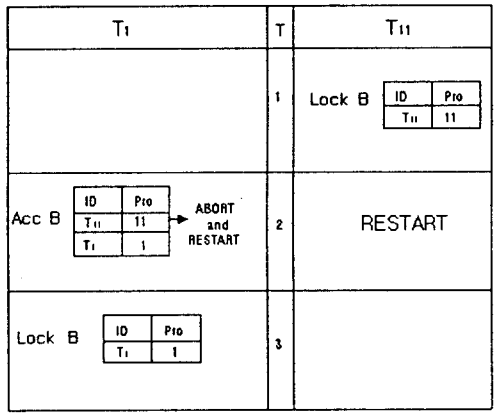

그림 3.3 데이터 우선순위 상속 알고리즘의 우선순위 상속

한다. 그림 3.3에서 동일 그룹 트랜잭션 T_5가 데이터 B를 잠금하고, 트랜잭션 T_1이 데이터를 잠금하지 않은 상황에서, 트랜잭션 T_1이 데이터 B에 접근하면 트랜잭션 충돌이 일어난다. 그때 트랜잭션 관리기를 검사하여 트랜잭션 T_1의 우선순위를 트랜잭션 T_5에게 상속하고, 트랜잭션 T_1은 트랜잭션 T_5의 우선순위를 상속 받고 대기한다. 그러므로 먼저 선점하고 있든 트랜잭션 T_5가 계속 B의 데이터를 잠금하고, 트랜잭션 T_1은 트랜잭션 T5의 우선순위를 상속 받아서 대기한다. 따라서 기존의 데이터 우선순위 알고리즘이 가지고 있는 낭비적 재시작 문제를 선택적 상속에 의해 감소시킴으로써 시스템의 낭비적 재시

작 비율을 줄인다..

IV. 성능 평가 및 결과

4.1 모의 실험 환경

트랜잭션 순서화 기법의 우선순위 상속 알고리즘, 데이터 우선순위 알고리즘, 데이터 우선순위 상속 알고리즘의 성능을 평가하기 위한 모의 실험 모델은 4개의[5] 구성 요소, 즉, 트랜잭션 발생기[6], 트랜잭션 관리기, 스케줄러, 자원관리기로 구성되어 있다. 이들 관계를 그림 4.1에 나타내었다.

트랜잭션 발생기는 트랜잭션의 도달을 모의 실험하고, 시스템 파라미터 값을 이용하여 각 트랜잭션의 형태, 마감시간, 데이터 접근 리스트를 정한다.

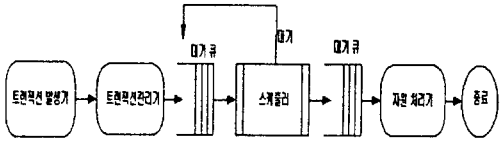

그림 4.1 모의 실험 환경

트랜잭션 관리기는 트랜잭션 고유번호를 발생시키는 역할과 트랜잭션들에게 우선순위를 부여하는 역할을 수행하며, 각 트랜잭션은 고유의 우선순위를 부여받으며, 짧은 트랜잭션이 그렇지 않은 트랜잭션보다 높은 우선순위를 갖는 방식을 채택한다. 스케줄러는 수행되는 기법에 근거해 데이터 접근을 순서화한다. 트랜잭션의 접근 요청은 트랜잭션의 우선순위에 기반하여, 승낙, 블록, 혹은 포기된다. 접근 요청이 승낙될 경우, 트랜잭션은 주기억장치에서 해당 데이터 항목을 읽으려 하고, 데이터가 주기억장치에 없다면, 트랜잭션은 데이터가 디스크로부터 주기억장치로 이전될 때까지 기다린다. 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. 그리고 데이터 우선순위 상속 알고리즘의 스케줄러 처리 절차를 그림 4.2에 나타내었다. 자원관리기는 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.

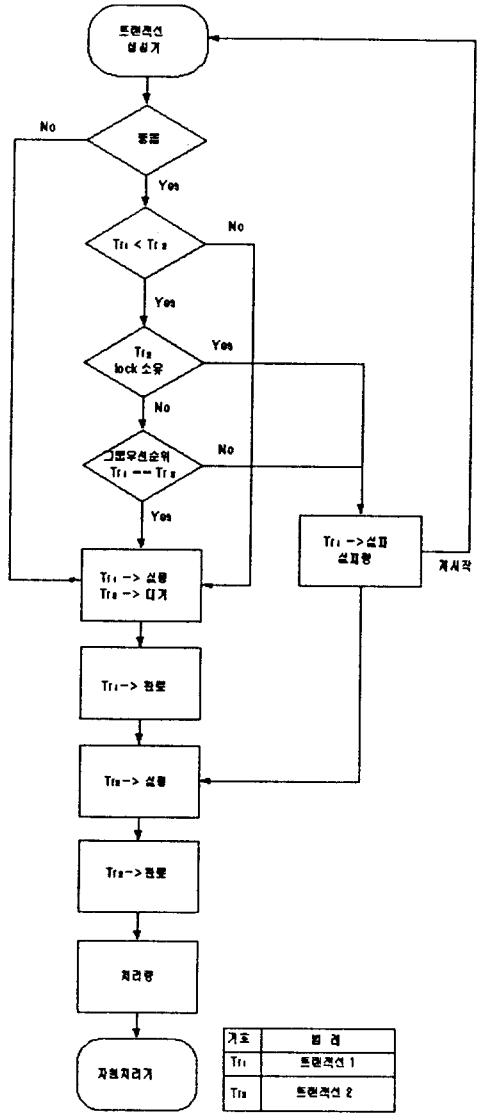

그림 4.2 스케줄러 처리 절차

4.2 모의 실험 평가 기준

우선순위 상속 알고리즘, 데이터 우선순위 기반형 알고리즘, 데이터 우선순위 상속 알고리즘을 아래의 2가지 측면에서 효율을 비교, 분석한다.

- 성공비율: 일정한 시간안에 수행된 트랜잭션 수와 총 트랜잭션 수의 비율
- 재시작 비율: 처리된 트랜잭션 수와 관찰된 재시작 트랜잭션 수의 비율

4.3 모의 실험 결과

우선순위 상속 알고리즘, 데이터 우선순위 알고리즘, 데이터 우선순위 상속 알고리즘을 성공비율과 재시작 비율의 측면에서 비교 분석 했다. 트랜잭션의 발생은 시간 간격은 EXPON(5)이고, 총 모의실험(simulation) 시간은 1000시간 이다.

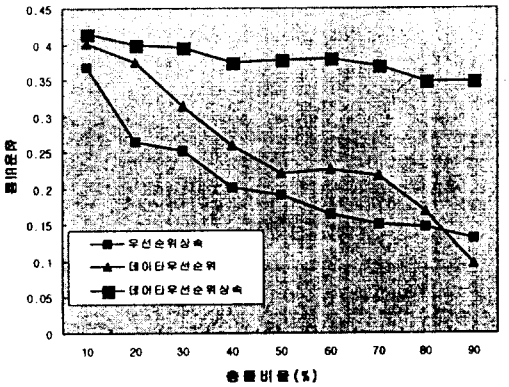

그림 4.3 충돌비 변화에 따른 성공 비율

표 4.1 충돌비에 따른 성공 효율 비교

데이터 우선순위 상속 알고리즘	효율단위(%)	비 고
우선순위 상속 알고리즘	17.10	효율 향상
데이터 우선순위 알고리즘	12.58	효율 향상

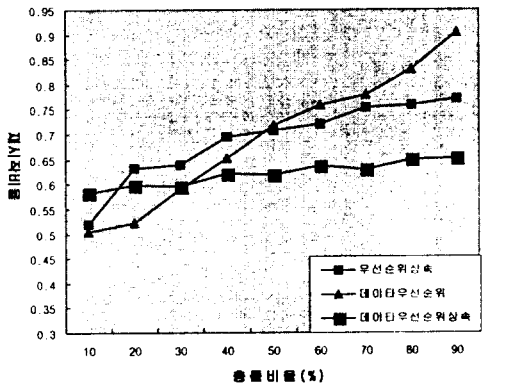

그림 4.4 충돌비 변화에 따른 재시작 비율

표 4.2 충돌비 변화에 따른 재시작 효율 비교

데이터 우선순위 상속 알고리즘	효율단위(%)	비 고
우선순위 상속 알고리즘	6.81	효율 향상
데이터 우선순위 알고리즘	7.49	효율 향상

V. 결 론

데이터 우선순위 잠금 알고리즘의 재시작 문제를 우선순위 상속에 의해서 해결하였다. 우선순위 상속에 의해 발생되는 교착상태 문제는 충돌한 트랜잭션이 충돌전에 다른 자원을 잠금하고 있는지 여부에 따라 상속 여부를 결정하는 선택적 상속 방법을 사용하여 해결했다. 처리율 향상을 위해서 낮은 트랜잭션이 자원을 장시간 잠금하는 것을 방지하기 위해 그룹 우선순위를 적용한 결과 성공 비율은 우선순위 상속 알고리즘 보다 17.10% 효율 향상되었으며, 데이터 우선순위 알고리즘 보다 12.5% 효율이 향상 되었다. 그리고 재시작 비율은 우선순위 상속 알고리즘보다 6.81% 효율 향상이 되었고, 데이터 우선순위 알고리즘 보다 7.49% 효율이 향상 되었다.

본 논문에서 제안하는 데이터 우선순위 상속 알고리즘은 처리 효율이 중시되는 분산 데이타 베이스 시스템에 적용시 시스템 성능 향상이 기대된다. 또한 최적의 그룹핑 단위를 구하면 분산환경에서 실시간 트랜잭션 처리를 요구하는 응용 분야에도 적용할 수 있으므로 이에 대한 향후 지속적인 연구가 필요하다.

참 고 문 헌

[1] L. Sha, R. Rajkumar, S. H. Son, C. H. Chang, "A Real-Time Locking Protocol", IEEE Transaction on Computers, Vol. 40, No. 7, 1991, pp. 793-900.

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

[3] L. Sha, R. Rajkumar, J. Lehoczky, "Concurrency Control for Distributed Real-Time Database", ACM SIGMOD Record, March 1988,

[4] 이병욱, "해제 일관성에 의한 분산 공유 메모리의 효율적 통신", 중앙대학교 전자계산학과 박사학위 논문, 1993

[5] 황규영(KAIST), 홍의경(서울시립대), 음두헌(덕성여대), 박영철(경북대), 김진호(강원대) 편역 "데이타베이스 시스템" 생능출판사.

[6] Ulusoy O, Belford G. G. 'Concurrency Control in Real-Time Database Systems', ACM 20th Annual Computer Science. Conf, March 1992,

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