□특 집□

실시간 트랜잭션 스케쥴링에서 교착상태 해결

동국대학교 전자계산학과 변 정 용**
홍익대학교 전자계산학과 김경창* · 임해철*

목 차

Ⅰ. 서 론
Ⅱ. 교착상태의 특성
Ⅲ. 실시간 교착상태의 해결법
Ⅳ. 요약 및 결론

Ⅰ. 서 론

여러가지 목적의 자료 접근에 대한 시간제약 요구를 만족하는 데이타베이스 시스템을 설계 구현하는 방법은 실시간 시스템의 성공에 중요한 역할을 한다. 실시간 데이타베이스는 명시적 시간제약을 가진 트랜잭션을 취급하기 위하여 설계된 트랜잭션 처리 시스템이다. 이러한 실시간 트랜잭션 관리가 중요시되고 있는 응용분야로는 증권거래 프로그램, 레이다 추적 시스템, 항공제어 시스템, 군명령통제 시스템, 통합제조 시스템, 로보트 공학, 원자력 발전소 등을 들 수 있다[4, 6, 8, 10, 13]. 이러한 응용분야들은 결과의 시간적 의미가 정확성 범주의 한 부분이 되고 있다. 이들은 일정한 시간제약 아래 데이타베이스 일관성을 만족시켜야 하고, 시스템이 과다적재 상태에 이르더라도 시간제약 내에 완료되는 트랜잭션의 수를 최대로 할 수 있어야 한다. 여기서 시간제약이란 어떤 트랜잭션의 실행이 종료되는 예상시간을 말하며 이를 규정시간(deadline)이라 한다.

기존의 데이타베이스 관리 시스템은 시간제약에 관계없이 도착된 모든 트랜잭션을 공평하게 처리하면서 그 시간을 최소로 하는 것을 목표로 하는 반면, 실시간 데이타베이스 시스템에서 트랜잭션 관리의 주된 목적은 직렬성의 순서로 대표되는 자료의 일관성을 유지하면서 각 트랜잭션과 결합된 규정시간에 적중하는 트랜잭션의 수가 최대가 되도록 스케쥴링 하는 것이다. 대부분의 실시간 병행수행 제어는 2단계 록킹법(2PL: 2Phase Locking)에 기반한다. 2PL은 접근하려는 모든 자료객체에 대하여 록킹을 하는 증대단계(growing phase)와, 실행이 완료(commit)되었을 때 모든 록크를 풀어주는 수축단계(shrinking phase)로 구성되어 있다. 일단 수축단계에 접어들면 록킹장치는 더 이상 이행되지 않는다. 하지만 2PL은 트랜잭션들이 병행으로 자료항목에 접근하려 할 때 길고 예측불가한 블럭킹 시간과, 이러한 상황이 발전되었을 때 자료 접근을 시도하는 트랜잭션들 상호간에 점유하고 있는 자료객체를 무한히 기다리게 하는 교착상태(deadlock)와 같은 본질적인 문제의 발생가능성을 가지고 있다. 실시간 트랜잭션 스케쥴링에서 교착상태에 효율적으로 대처할 수 있는 시간적 특성을 고려한 해결법이 요구된다.

본 논문은 2PL에서 트랜잭션의 스케쥴러가 관리하고 사용하는 록킹장치와 교착상태간에 관련된 특성을 파악하고, 기존하는 교착상태 해결법인 탐지 및 회복법, 예방법, 회피법에 대하여 실시간 트랜잭션 스케쥴링에서 고려하여야 할 사항들을 검토하고, 지

*종신회원
**정회원

금까지의 연구[10, 11, 15, 16]에서 제안된 방법의 특성을 점유하고 있는 자료객체를, …, T_{n-1}은 T_n이 점유하고 소개한다. 본 논문의 구성은 II에서 교착상태의 필요조건과 2PL에서 발생하는 상황 등의 특성을 소개하고, III은 실시간 트랜잭션 스케쥴링에서 교착상태의 해결법을 소개하고, 고려 사항을 토의한다. 그리고 IV에서 요약 및 결론과 앞으로 할 일의 소개로 되어 있다.

II. 교착상태의 특성

2PL에서 바람직하지 않지만 중요한 특성은 교착상태가 발생한다는 점이다. 교착상태에서 미완료 상태의 트랜잭션이 자료객체들을 묶어둠으로써 더 이상 진행이 차단됨은 물론 그 자원에 접근하려는 다른 트랜잭션들을 무한정 기다리게 하여 시스템의 성능을 저하시킬 수 있다. 먼저 이러한 교착상태가 발생하게 되는 필요조건[11, 16]을 보고, 트랜잭션 스케쥴링에 있어서 대표적인 두 가지의 교착상태가 발생하는 상황[2]을 소개한다.

2.1 필요조건

교착상태가 발생하려면 다음 네 가지 조건이 동시에 만족되어야 하며, 이들 조건은 상호 완전 독립되어 있지 않다. 이들은 운영체제의 정의를 빌어서 데이타베이스 트랜잭션 스케쥴링에 적용한 것이다. 스케쥴링의 단위가 운영체제에서 프로세스 또는 타스크인 반면 데이타베이스에선 트랜잭션이고, 경쟁하는 대상을 프로세스 CPU, 입출력장치, 주기억장치 등의 자원으로 하고 있는 반면 트랜잭션은 자료객체로 하고 있다.

2.1.1 상호배제

적어도 하나 이상의 자료객체는 비공유(exclusive lock)되어야 한다. 한번에 한 트랜잭션만이 이 자료객체를 사용할 수 있음을 말한다. 이때 다른 트랜잭션이 그 자료객체에 대한 접근을 요청한다면 요청 트랜잭션은 그 자료객체가 해제될 때까지 기다려야 한다.

2.1.2 부분할당

한 트랜잭션이 적어도 하나의 자료객체를 점유한 상태로 현재 다른 트랜잭션이 점유하고 있는 자료객체를 얻으려고 기다리고 있어야 한다.

2.1.3 비선점(non-preemption)

어떤 트랜잭션이 소유하고 있는 자료객체를 다른 트랜잭션이 강제로 빼앗을 수 없다. 반드시 점유하고 있는 자료객체는 트랜잭션이 완료되어야 해제될 수 있다.

2.1.4 환형대기

대기 상태인 트랜잭션의 집합 {T0, T1, T2, …, Tn}이 있을 때 T0은 T1이 점유하고 있는 자료객체를, T1은 T2가 점유하고 있는 자료객체를, T2는 T3이 점유하고 있는 자료객체를, …, T_{n-1}은 T_n이 점유하고 있는 자료객체를, Tn은 T0이 점유하고 있는 자료객체를 얻으려고 각각 기다린다. 점유와 대기는 본 조건에도 내포되어 있다.

2.2 고전적 상황

2PL 스케쥴러는 트랜잭션 T1, T3을 처리하고 있다고 가정하고 가장 일반적인 교착상태의 발생 상황에 관한 예를 들어 본다. 단, 트랜잭션 처리 모델은 트랜잭션 관리기(TM)가 요청을 하면 트랜잭션 스케쥴러가 록크얻기(get－lock)를 시도하고, 록크를 얻으면 디스크 관리기(DM)에게 처리를 요청한다.

T1 : r1[x]→w1[y]→c1
T3 : w3[y]→w3[x]→c3

여기서 x, y는 접근하려는 자료객체이고, 이들 x, y에 대한 읽기 및 쓰기 요청은 r[x], w[x]이고, 읽기록크(read-lock)와 쓰기록크(write-lock)을 표시하기 위하여 rl[x], rl[y], wl[x], wl[y]를 사용한다. 그러면 다음과 같은 연속된 사건의 결과로 교착상태가 발생함을 보자.

(1) 처음 트랜잭션은 어떠한 록크도 소지하고 있지 않다.

(2) 스케쥴러는 TM로부터 r1[x]를 받아서 rl1[x]를 DM에게 제시한다.

(3) 스케쥴러가 TM으로부터 w3[y]를 받아서 wl3[y]를 얻은 다음 w3[y]를 DM에게 제시한다.

(4) 스케쥴러가 TM으로부터 w3[x]를 받아서 wl3[x]를 얻으려 할 때 이미 록크가 되어 있는 rl1[x]과 충돌하므로 록크를 얻지 못한다. 이래서 wl3[x]는 진행이 지연된다.

(5) 스케쥴러는 TM으로부터 w1[y]를 받는다. (4)

에서처럼 wl[y]는 진행이 지연되어야 한다.

스케쥴러가 2PL의 규칙 그대로 행동했다 하더라도 T1과 T3 어느 트랜잭션도 이들 규칙중에 하나를 위반함이 없이 완료할 수는 없다. 즉 (4) (5)에서 스케쥴러가 wl3[x]나 wl1[y]를 얻음이 없이 w3[x]나 wl[y]를 DM에게 보낸다면 (1)을 위반하게 된다. wl3[y]를 해제한다면 wl1[y]를 얻을 수 있으므로 DM에게 w1[y]를 보내는 것이 허용된다. 이 때 스케쥴러는 wl3[x]를 결코 얻을 수 없게 된다. 만약 할당하게 된다면 규칙을 위반하게 된다. 이러한 상황을 고전적 교착상태라 한다. 이러한 교착상태에서 두 트랜잭션 가운데 어느 것이 진행되려면 먼저 상대가 진행에 필요로 하는 자료객체를 포기하고 풀어주어야만 한다.

2.3 록크 변환

트랜잭션이 읽기록크를 쓰기록크로 강화하려고 할 때 교착상태가 발생한다. 트랜잭션 Ti가 자료객체 x를 읽은 다음 그것을 쓰려고 하는 상황을 가정하자. Ti는 스케쥴러에게 ri[x]를 보내고 스케쥴러는 rli[x]을 얻는다. 이때 Ti가 스케쥴러에게 wi[x]를 보내면 스케쥴러는 rli[x]를 wli[x]로 록크의 격을 강화하여 바꿔 주어야 한다. 이러한 록크의 강화를 록크변환(Lock conversion)이라 한다. 2PL에 따르기 위하여 스케쥴러는 rli[x]를 해제하여서는 안된다. 같은 트랜잭션에 의한 록크얻기는 다른 트랜잭션과의 충돌이 아니기 때문에 문제가 되지 않는다. 그러나 두 트랜잭션이 병행으로 그들의 자료객체에 관한 읽기록크를 쓰기록크로 바꾸려고 할 때 결과는 교착상태가 된다.

예를 들어서 트랜잭션 T4, T5가 2PL 스케쥴러에게 다음과 같은 연산을 발생시킨다고 가정하자.

T4 : r4[x]→w4[x]→c4
T5 : r5[x]→w5[x]→c5

스케쥴러는 다음 사건의 결과에 직면하게 될 것이다.

(1) 스케쥴러는 r4[x]를 받아서 rl4[x]를 얻은 다음 r4[x]를 DM에게 보낸다.
(2) 스케쥴러는 r5[x]를 받아서 rl5[x]를 얻은 다음 r5[x]를 DM에게 보낸다.
(3) 스케쥴러는 w4[x]를 받는다. wl4[x]는 rl5[x]와 충돌하기 때문에 연산을 지연시켜야 한다.
(4) 스케쥴러는 w5[x]를 받는다. wl5[x]는 rl4[x]와 충돌하기 때문에 연산을 지연시켜야 한다.

여기서 두 트랜잭션 가운데 어느 것도 자신이 소유하고 있는 r1[x]를 해제할 수 없기 때문에 트랜잭션은 교착상태가 된다. 이와같은 교착상태는 트랜잭션이 대량의 자료 가운데 어떤 값을 가진 자료객체를 찾아서 그 값을 갱신하려고 할 때 공통적으로 발생한다. 즉 먼저 각 자료객체에 대하여 읽기록크를 얻고난 다음 해당자료를 갱신하려고 하였을 때 읽기록크를 쓰기록크로 바꿔주게 된다.

III. 실시간 교착상태의 해결법

실시간 데이타베이스에서 트랜잭션의 종료는 규정시간에 맞추어야 의미를 갖는다. 트랜잭션 스케쥴러는 규정시간에 적중하는 트랜잭션 수의 최대화를 목표로 한다. 이를 위하여 블럭킹이나 교착상태와 같은 무한정으로 기다리는 시간요인들을 제거하거나 그러한 결과를 해결하여야 한다. 그러한 실시간 트랜잭션 스케쥴링의 목적에 부합하려면 트랜잭션과 결합된 시간적 제약에 가장 심대한 문제를 일으킬 교착상태 해결법이 요구된다. 교착상태 해결법은 교착상태가 발생하지 않도록 조건을 억제하는 예방법(prevention), 트랜잭션에 관한 사전지식에 의존하여 불안정한 자료 요청을 제약하는 회피법(avoidence), 그리고 이러한 방법을 쓰지 않고, 교착상태가 발생하였을 때 이를 탐지해서 회복하는 탐지 및 회복법(detection and recovery) 등이 있다. 본장은 이러한 해결법에 시간적 특성을 고려하여 확장 가능성을 검토하고, 이들에 관한 기존의 연구결과를 소개한다.

3.1 탐지 및 회복법

앞에서 이미 언급되었지만 트랜잭션의 병행수행 제어법으로 가장 일반적이며 널리 쓰이는 프로토콜이 2PL이다. 이 2PL의 록킹장치가 실시간 환경에서 블럭킹과 교착상태 발생의 원인이 되고 있다[2]. 이러한 문제의 해결이 OCC 또는 OCCL(Optimistic Concurrency Control Locking)에서 추구되고 있다[10, 13]. 그러나 실제 구현 상황을 고려한 실험결과[10]에 따르면 OCCL은 단지 자료객체에 대하여 경쟁이 낮을 때 2PL 보다 유리하며, 경쟁이 높을 때는 2PL보다 실행성능이 떨어진다고 한다. 이러한 점에서 2PL은 일반적으로 가장 많이 사용한다는 사실 이외에도 그 사용에 대한 정당성을 가지고 있다고 보아야 한다. 이러한 근거에 따라 2PL을 사용하면 교착상태를 피할

수 없으며, 실시간 요구에 만족하도록 교착상태를 해결하는 전략이 필요하다.

3.1.1 탐지법

스케쥴러는 어떠한 트랜잭션도 영원히 블럭되지 않도록 교착상태를 탐지해 낼 수 있는 전략이 필요하다. 그리고 실시간 환경에서 교착상태에 처하여 있는 트랜잭션이 규정시간을 놓치지 않도록 탐지 전략은 탐지 전략은 효율적이어야 한다. 여기서 교착상태 발생빈도와 이를 실시간에 탐지하는 전략을 살펴본다.

(1) 한계시간(timeout)

스케쥴러는 어떤 트랜잭션이 록크 요청에 대하여 너무 오랫동안 기다렸다고 판단하면 그 트랜잭션이 개입된 교착상태가 있음을 추측하고 그 트랜잭션을 중단시킨다. 그런데 실제로 교착상태가 아닌 긴 트랜잭션에 의하여 점유된 자료를 기다리는 경우일 수도 있다. 이러한 사실에 기반하여 한계시간의 길이를 트랜잭션의 특성에 따라서 조절하므로써 교착상태가 아닌 트랜잭션의 중단을 막을 수 있다. 여기서 특성에는 시간제약도 포함된다.

시간제약을 고려한 방법으로는 트랜잭션의 규정시간에 따라서 한계시간의 길이를 유동적으로 부여하는 것이다. 다시 말해서 임박한 규정시간 우선법(EDF)[8]과 같은 방법으로 규정시간에 임박한 트랜잭션일수록 록크요청을 한 다음 기다리는 한계시간을 짧게 가지는 것이다. 그렇게 하므로써 규정시간에 임박한 트랜잭션이 다른 대응조치를 취할 수 있게 한다. 그러므로 시스템 성능 저하를 초래할 수 있다.

$$\text{한계시간(T)} = \text{표준한계시간} - n/(\text{규정시간(T)} - \text{현재시간}) \quad \langle\text{식-1}\rangle$$

〈식-1〉은 주어진 표준 한계시간을 기준으로 하여 규정시간에 임박할수록 한계시간의 값이 작아짐을 나타낸다. n은 한계시간의 길이를 일정비율로 조절할 수 있는 값으로 n이 클수록 한계시간은 더욱 작아진다. 물론 이 식은 한계시간 평가의 한 예이며 보다 정확한 시스템 상황을 반영할 수 있는 식을 구하기 위한 추후 연구가 필요하다고 본다.

(2) CPU의 한계효율

일정값 이하로 CPU 효율이 떨어졌을 때 교착상태의 가능성을 예견하고 교착상태 탐지 작업에 돌입한다. 여기서 CPU 효율은 스케쥴러가 트랜잭션의 규정시간 내에 이룩한 실행완료율에 대응시킬 수 있다. 그러므로 CPU의 한계효율은 트랜잭션의 스케쥴러 한계효율로 할 수 있다.

(3) 탐지 빈도

위의 두 가지 탐지 전략을 간접적인 상황에 의한 교착상태의 예측법이라 한다면, 이에 대응하는 또 하나의 다른 접근법은 정확한 탐지법이다. 이러한 정보를 제공하기 위하여 스케쥴러는 트랜잭션의 이름이 부착된 노드와 연결선으로 구성된 대기 그래프(WFG: waits-for graph)를 유지한다. 대기 그래프는 G=(V, E)로 표시된다. 트랜잭션 Ti는 트랜잭션 Tj가 점유하고 있는 록크가 해제되기를 기다리고 있다면 두 트랜잭션 Ti, Tj는 Ti→Tj로 표시된다. 대기 그래프의 유지는 Ti에 의한 록크 요청이 Tj에 의하여 점유된 록크와의 충돌로 Ti가 블럭될 때 스케쥴러가 연결선 Ti→Tj를 WFG에 추가하고, Ti가 요청한 록크를 블럭시킨 Tj가 소유하고 있는 록크가 해제될 때 WFG에서 Ti→Tj를 제거한다. 이러한 과정에서 대기 그래프가 T0→T1→⋯→Tn→T0처럼 사슬(cycle)을 이루게 될 때 T0는 결국 자신을 기다리면서 블럭상태에 있게 되어 교착상태가 된다. 탐지 전략이란 스케쥴러가 WFG에서 사슬을 검사하여 교착상태를 탐지해 내는 것이다.

시간제약과 관련하여 스케쥴러가 WFG에서 사슬을 검사하는 방법을 검토해 보자. 새로운 연결선이 추가될 때마다 그 연결선을 포함하여 사슬을 찾는 것이다. 이것은 부담이 아주 크다. 왜냐하면 드물게 나타나는 교착상태를 검사하기 위하여 매번 검사하는 것은 노력의 낭비일 수 있다. 대안으로 몇개의 연결선이 추가될 때까지 또는 일정 한계시간 동안 기다리는 방법을 사용할 수 있다. 검사 횟수를 줄인다 하여도 이미 존재하고 있는 교착상태는 모두 탐지 가능하기 때문이다. 그러나 시간제약을 고려할 때 이들 검사시간의 간격은 트랜잭션과 결합된 규정시간과 밀접한 관계에 있다. 여기서 연결선을 WFG에 추가할 때 트랜잭션의 중요도(criticalness), 우선순위, 규정시간 등을 고려하여, WFG에 대한 사슬의 존재 여부 검사는 탐지전략에 따라서 연결선을 WFG에 추가할 때, 해당 트랜잭션이 가진 중요도나 우선순위의 크기 또는 규정시간의 임박함 정도에 따라서 결정할 수 있다. 예를 들어서 어떤 트랜잭션의 규정시간 임박 정도가 10 ms←ret(T)(남은 실행시간)가 되는 트랜잭션은 WFG에 연결선을 추가할 때마다 검사하도록 하고, 100 ms←ret(T)←200 ms 인 트랜잭션은 20 ms, 200 ms←ret(T)이면 30 ms의 한계시

간이 소요된 다음에 록크얻기를 실패했을 때 WFG에서 사슬을 검사하게 하는 전략을 취할 수 있다.

3.1.2 회복법

교착상태가 탐지되면 WFG에서 사슬을 끊기 위하여 사슬의 형성에 개입된 트랜잭션 가운데 하나를 선택하여 중단시켜야 한다. 실시간 트랜잭션들에 대하여 희생자 선정에는 두 가지 사항을 중점적으로 고려하여야 한다. 첫째는 희생자를 제외한 나머지 트랜잭션들의 시간제약이 가능한 많이 적중될 수 있도록 하여야 한다. 둘째 중단에 소요되는 비용이 최소가 되어야 한다. 희생자 선정에 고려되는 일반적인 사항을 나열하면 다음과 같다.

- 트랜잭션이 투자한 노력의 양
- 트랜잭션의 중단 비용
- 트랜잭션의 실행 완성에 소요되는 노력의 양
- 트랜잭션을 포함하는 사슬의 수

여기서 교착상태를 회복하기 위하여 트랜잭션의 특성으로써 이상의 사항을 포함하여, 시간적 특성과 중단 연산의 비용 그리고 방법의 복잡성을 고려한 다섯가지 해결책[11]을 소개한다. 단 여기서 zt는 제로 포인터라 하며, 중요도를 가진 트랜잭션이 규정시간 dt을 초과한 다음에 결과의 의미가 완전히 없어지는 지점을 말한다. 이것은 유연한 규정시간(soft deadline)이 고려될 수 있음을 의미한다.

[교착상태 해결책 1(DRP1)] 교착상태 탐지에 개입하고 있는 트랜잭션을 항상 중단한다. 이 정책은 교착상태 사슬에 관련된 트랜잭션들에 관한 어떠한 정보도 필요로 하지 않는다는 점에서 간단하고 효율적이다.

[교착상태 해결책 2(DRP2)] 교착상태 사슬을 추적해서 트랜잭션의 결과와 의미가 완전히 없어진 트랜잭션들 가운데 즉 t>zt인 첫번째 트랜잭션을 중단한다. 그렇지 않으면 가장 긴 규정시간을 가진 트랜잭션을 중단한다.

[교착상태 해결책 3(DRP3)] 교착상태 사슬을 추적해서 트랜잭션의 결과의 의미가 완전히 없어진 트랜잭션들 가운데 즉 t>zt인 첫번째 트랜잭션을 중단한다. 그렇지 않으면 규정시간에 가장 임박한 트랜잭션을 중단한다.

[교착상태 해결책 4(DRP4)] 교착상태 사슬을 추적해서 트랜잭션의 결과와 의미가 완전히 없어진 트랜잭션들 가운데 즉 t>zt인 첫번째 트랜잭션을 중단한다. 그렇지 않으면 가장 낮은 중요도를 가진 트랜잭션을 중단한다.

DRP1은 트랜잭션의 특성 보다는 가장 최근에 교착상태를 유발한 트랜잭션을 중단하므로써 해결한 방법이다. 여기에 교착상태를 유발한 트랜잭션이 우선순위가 매우 높을 경우 대기 그래프의 사슬에서 바로 전 트랜잭션을 희생자로 선정하는 방법이 있을 수도 있다. DRP2는 기아상태 발생 가능이 있고, DRP3는 문제가 없겠으나 그 결과는 회복하는데 비용이 많이 들 수 있다. DRP4는 일반적인 방법이다.

$$time\_needed.T(t) = ((t - st.T) \times (R\_total.T - R\_accessed.T(t))) / R\_accessed.T(t) \quad \langle 식\text{-}2 \rangle$$

단, R_total.T는 트랜잭션 T가 접근해야 할 전체 레코드 수이고, st.T는 트랜잭션 T가 도착한 시간이다. R_accessed.T(t)는 트랜잭션 T가 현재시간 t까지 접근 완료한 레코드의 수이고, time_needed.T(t)는 트랜잭션 T가 현재시간 t에서 접근해야할 남은 레코드 수이다.

[교착상태 해결책 5] 자료접근 충돌 해결에 사용하였던 트랜잭션의 남은 실행시간 평가식인 〈식-2〉에서 인용한 트랜잭션 T을 완성하기 위하여 필요한 시간인 time_needed.T(t)를 사용한다. 만약 (time_needed.T(t)+t)<dl.T이면 트랜잭션 T는 "적합(feasible)"이라 하고, 그렇지 않으면 "완만(tardy)"이라고 한다. 이 해결책은 완만한 트랜잭션 가운데 최소의 중요도를 갖는 트랜잭션을 중단시킨다. 그렇지 않으면 최소의 중요도를 가진 적합한 트랜잭션을 중단한다. 알고리즘[11]은 다음과 같다.

```
S1: set tardy_set to empty
    set feasible_set to empty
S2: trace deadlock cycle
    for each T in the cycle do
    if t>zt then abort T; return
    else if T is tardy then add T to tardy_
        set
        else add T to feasible_set
        end if
    end if
S3: if tardy_set is not empty
        then search tardy_set for T wih the
        least criticalness
```

```
        else search feasible_set for T with the
        least criticalness
    end if
    abort T
    return
```

그리고 간접적으로 해결되는 예로는 실시간 트랜잭션 스케쥴링에서 고순위의 트랜잭션이 록크요청에서 실패하여 블럭될 때 발생되는 우선순위역행(priority inversion) 현상을 해결할 때에 우선순위 중단법(PA: priority abort)[9]에 의하여 낮은 우선순위를 가진 트랜잭션을 중단시킴에 따라서 이 때 교착상태가 존재한다면 함께 해결될 수도 있을 것이다.

3.2 예방법

II에서 언급한 교착상태의 4가지 필요조건 가운데 적어도 하나를 거부하므로써 교착상태가 일어나는 것을 방지하는 수단이다. 이들은 교착상태가 중대한 손실이나 사태를 초래하는 경우에 적합하다. 예를 들어서 화학, 핵처리 등을 제어하거나 병원의 중환자를 감시하고 제어하는 시스템 등이다.

실시간 데이타베이스에서 예방법을 직접 연구한 결과는 없다. 그런데 예방법은 실시간 트랜잭션 스케쥴링에서 목적하는 바와 일맥상통한다. 그 목적하는 바란 바로 교착상태의 필요조건 가운데 하나인 자료객체에 접근할 때 상호배제를 위하여 사용하는 록킹장치(Locking)를 제거하는 것이다. 자료접근의 상호배제적 특성을 제공하는 록킹장치는 병행수행을 위하여 사용하는 가장 일반적인 방법인 2PL에 의하여 설치된 장치이다. 실시간 트랜잭션 스케쥴링에서 록킹장치를 제거하려는 것은 자료객체의 접근 요청에서 록크 얻기를 실패하면 블럭킹상태에서 해제가 될 때까지 기다리는 시간이 불확정적이고 예측불가하다는 특성 때문이다. 이러한 사실은 실시간 스케쥴러의 입장에서 트랜잭션이 규정시간에 맞게 완료를 할 수 없게하는 요인을 제공한다. 교착상태는 불확정적 예측불가성을 가지고 있으며, 또한 블럭킹 상태에 있는 트랜잭션이 대기 그래프에서 사슬을 가지는 특수한 경우에 해당한다. 따라서 상호배제적 특성을 완화하거나 록킹장치를 제거하므로써 4가지 필요조건 가운데 하나를 거부하는 것이 된다. 이러한 연구로는 낙관적 접근법 (OCCL)[10]을 비롯하여 2PL-OS[1], 통합실시간록킹법[13, 14] 등이 있다. 이들은 록킹장치를 일부 또는 전혀 사용하지 않거나, 상호배제 특성을 최대로 완화시키는 전략을 취하고 있다. 또한 운영체제에서 제기하고 있는 세 가지 중요한 방지법인 총체적 요청법(collective requests), 순서적 요청법(ordered requests), 선점법(preemption) 등을 실시간 트랜잭션 스케쥴링에서 적용하는 방법을 검토해 볼 필요가 있다고 보며 추후 연구과제로 미룬다. 다음은 앞에서 언급한 상호배제 특성 거부와 관련한 방법의 특성을 소개한다.

3.2.1 OCCL

Haung[1992]은 OCC의 구현상의 정확성을 보장하기 위하여 록킹장치를 부분적으로 도입한 OCCL[10]을 제안하였다. 이것은 비록 R_lock이 사용되고 있지만 유효성 검사 트랜잭션이 임계구역(critical section)에서 V-lock를 얻도록 하므로써 교착상태가 없고, 고도의 잠재적 병행성을 띄는 특성을 가진다. 그것은 자신이 요청한 모든 V_lock을 부여받은 트랜잭션은 임계구역을 떠난 다음에는 어떤 록크도 요청하지 않기 때문에 다른 트랜잭션을 기다리지 않게되고, 따라서 대기사슬을 형성하지 않는다는 사실에 기인한다. 그리고 OCCL은 논리적 수준에서 2PL과 다르지만 구현에 있어서 비슷한데, 둘 다 록킹장치를 사용하고, 해시 데이블 및 록크 테이블의 관리에 개입하고 있다. 그런데 2PL은 교착상태 탐지를 하여야 하고, OCCL은 그럴 필요가 없다. 그러나 교착상태의 탐지는 심각한 부담을 주지 않는다는 연구 결과가 있다. 한편 OCCL은 개별 트랜잭션의 읽기, 쓰기 연산 집단의 관리에 많은 비용이 든다고 한다. 또한 양자는 록킹장치를 쓰므로써 똑같이 우선순위 역행이 발생한다. 결국 [10]의 실험 결과에서 OCCL이 2PL보다 더 큰 부담이 있다고 지적한 사실에서 장단점을 알 수 있다.

3.2.2 2PL-OS

2PL-OS는 경직된 규정시간을 가진 실시간 데이타베이스에서 가장 일반적인 록킹장치 접근법인 2PL에 "순서적 공유법(OS: Ordered Sharing)"을 결합한 방법이며, 읽기와 쓰기 연산에서 일어나는 블럭킹을 없앨 목적을 가지고 있다[1]. 이것은 트랜잭션들이 완료단계에서 지연되는 특성을 가졌다. 블럭킹에는 읽기-쓰기, 쓰기-읽기, 쓰기-쓰기 등 세 가지 유형이 있고, 이들이 제거 대상이 된다. 예를 들어서 읽기-쓰기 블럭킹을 제거하려면 트랜잭션 Ti가 어떤 객체에

관한 읽기록크를 점유하고 있더라고 트랜잭션 Tj가 같은 객체에 대한 쓰기록크를 부여받을 수 있다는 것이다. 이 때 Ti의 읽기록크로부터 Tj의 쓰기록크로 "순서적 공유 관계(ordered shared relationship)"가 있다고 한다. 2PL-OS는 직렬성을 보장하려고 "순서적 공유 규칙"을 준수한다. 그러나 2PL-OS가 가지고 있는 완료단계에서 발생하는 지연은 읽기와 쓰기 연산을 실행함으로써 다른 트랜잭션을 블럭시키지 않는다. 그리고 실시간 데이타베이스에서 트랜잭션의 시의성이 응답시간 보다 더 크므로 지연은 성능 저하를 초래하지 않는다. 2PL-OS의 문제는 연쇄중단(CA: Cascaded Abort)인 데, 일차적으로 쓰기-읽기 블럭킹을 유지하므로써, 그라고 이차적으로 객체들의 이전 상태 자료(BI: Before-Image)를 이용하므로써 연쇄중단을 회피할 수 있다. 전자를 ACA 2PL-OS, 후자를 2PL-OS/BI라 한다. 여기서 록킹장치를 제거하므로써 상호배제 조건을 거부하였다. 그러므로써 교착상태를 방지할 수 있다.

3.2.3 통합실시간 록킹법

손상혁[1991, 1992]은 낙관적 접근법과 우선순위 의존 록킹장치를 통합한 병행수행 제어에서 시간을 중시하는 통합 실시간 록킹법 (integrated real-time protocol)이라는 새로운 방법[13, 14]을 제시하였다. 여기서 이 알고리즘이 갖는 전체적인 특성을 요약 소개한다.

트랜잭션의 데이타 요구나 실행시간에 관한 어떤 정보도 사용하지 않는다는 특성이 있다. 또한 트랜잭션의 쓰기연산을 지연시키므로써 트랜잭션의 시의성과 중요성에 따라서 트랜잭션간의 직렬성 순서를 다양하게 조정하도록 사용한 것으로 직렬성 순서에 관한 기존의 제약을 완화하였다. 이는 시스템의 병행성 수준을 줄이거나 현저하게 재시동율을 증가시킴이 없이 가능한 많은 트랜잭션이 시간제약을 만족할 수 있도록 하였다. 또한 높은 우선순위 트랜잭션은 완료되지 않은 낮은 우선순위 트랜잭션에 의하여 결코 블럭되지 않으며, 저순위 트랜잭션도 연산충돌에 직면하여 고순위 트랜잭션에 의하여 중단되지 않을 수 있다. 따라서 직렬성 순서의 조정은 시간중시 스케쥴링을 지원하는 장치로 볼 수 있다.

트랜잭션의 처리 과정을 간단히 요약하면 트랜잭션 관리기에 제시된 각 트랜잭션은 읽기단계, 대기단계, 쓰기단계를 거쳐서 종료된다. 읽기단계 동안 트랜잭션은 데이타베이스로부터 자료를 읽어서 지역작업공간에 저장하고, 이 단계가 완성된 후에는 대기단계에서 완료할 기회를 기다린다. 완료가 되면 쓰기단계로 연결되고 변경된 모든 내용은 데이타베이스에 영구상태로 저장된다. 트랜잭션이 세 단계의 어떤 상태에 있을 때 활동중(active)이라 한다. 여기서 제안된 알고리즘은 읽기-쓰기 충돌에 대하여 2PL을 그리고 쓰기-쓰기 충돌에 대하여 TWR(Thomas's Write Rule)[2]을 사용하는 스케쥴러들의 특성을 통합한 것이다. 블럭킹을 없애므로써 교착상태를 방지한다.

3.3 회피법

트랜잭션의 앞으로의 행동에 관한 일부 지식에 의존하여 교착상태가 발생할 가능성이 있는 자료객체의 할당 방식을 미리 제약하므로써 그 발생을 피하는 방법이다. 이 해결책은 자료를 비롯한 자원의 이용율의 저하가 불가피하다. 특히 교착상태로 비싼 댓가를 치르는 자료객체인 경우에 이 방법으로 관리된다[8]. 여기서 트랜잭션의 앞으로의 행동에 관한 정보를 미리 안다는 것은 몇몇 예에서는 쉽지만 다른 경우에는 쉽지 않다. 실시간 데이타베이스인 경우 교착상태의 회피법에 관하여 연구된 내용은 아직 없다. 운영체제에서 사용하는 대표적인 회피 알고리즘인 "뱅커스 알고리즘(Banker's algorithm)"은 프로세스가 필요로 하는 자원의 최대 수를 미리 알아서 이를 만족할 때는 안전 상태라하여 진행시키고, 그렇지 못하면 자원이 해제될 때까지 기다린다. 이 알고리즘은 Dijkstra(1969)에 의하여 다수 자원으로 확장되었다. 이러한 회피 알고리즘이 실시간 트랜잭션 스케쥴링에 적합성을 갖는지에 대하여 연구가 요구된다.

실시간 트랜잭션 스케쥴링에서 트랜잭션의 미래의 행동이란 트랜잭션이 접근하려는 자료객체의 수를 비롯하여, 규정시간 초과 및 고우선순위 트랜잭션에 의하여 취소될 저순위 트랜잭션에 대한 예측취소 등이 있다[17, 18]. 이러한 예측 취소는 직접적인 교착상태 해결법이라기보다는 간접적으로 자원의 사용가능성을 높여 주고, 블럭상태에서 자료객체를 기다리는 트랜잭션의 수를 줄여 주므로써 교착상태의 발생가능성을 간접적으로 줄여준다.

여기서 마애가와(1987)가 제안한 운영체제에서 발생하는 교착상태 회피 알고리즘을 소개하고, 이를

실시간 트랜잭션 스케쥴링에 대하여 검토해 본다.

[회피 알고리즘]

(1) 요청 연결선을 할당 연결선으로 변경하여 미래의 상태를 예측해 본다.

(2) 이 상태에 대한 최대 수요 그래프를 구성하고 교착상태에 대하여 분석한다. 만약 교착상태가 존재한다면 그 요청 받기를 지연하고, 그렇지 않으면 그 요청을 받는다.

위의 알고리즘에서 (1)은 트랜잭션의 우선순위를 고려하여 예측할 수 있고, (2)의 경우에는 트랜잭션의 특성에 따라서 고순위에 중요도가 높으면 선점까지도 고려해 볼 수 있다.

3.4 록킹의 단위

록킹 단위가 크면 트랜잭션의 대기가 빈번해지고 그러므로써 교착상태가 발생할 가능성이 높아진다. 그러나 작은 록킹 단위는 갱신 연산에서 특히 중요한 테이블이나 인덱스의 록킹을 최소화하므로써 병행수행을 최대화한다. 이처럼 작은 록킹 크기, 클러스터링, 분할 등은 트랜잭션의 대기를 최소화하게 되어 교착상태의 4가지 필요조건 가운데 하나인 "점유와 대기"라는 조건을 약화시키므로써 교착상태의 확률을 줄이는데 간접적으로 기여할 수 있다.

3.5 기아상태

교착상태가 아니면서 그와 같은 모습을 보이는 것이 기아상태(starvation problem)이다. 긴 트랜잭션은 다른 트랜잭션과 충돌할 확률이 높기 때문에 반복해서 재출발(repeated restart)하기 쉽고, 그래서 짧은 트랜잭션 보다 규정시간에 적중할 기회가 적다. 이러한 현상은 실시간 트랜잭션 스케쥴링에서 규정시간 적중을 어렵게 만든다. OCCL은 전통적인 재출발 횟수 제한 등의 방법 대신에 트랜잭션의 길이와 규정시간에 민감한 우선순위 할당법을 사용하여 이 문제에 대처하고 있다. OCCL은 가중화된 우선순위 스케쥴링법과 통합하여 2PL보다 기아상태의 대처에 좀 더 큰 융통성을 갖는다[10, 11, 16]고 한다.

IV. 결 론

교착상태의 필요조건을 포함한 본질적 특성을 알아보고, 교착상태의 일반적인 해결법인 탐지 및 회복법, 예방법, 회피법에 대하여 실시간 데이타베이스의 트랜잭션 스케쥴링에서 고려하여야 사항을 논의하였다.

교착상태는 2PL에서 본질적으로 발생하는 블럭킹의 특수한 한 형태이며, 실시간 트랜잭션 스케쥴링에서는 매우 중요하다. 탐지법에서 트랜잭션과 결부되어 있는 시간적 특성인 규정시간, 우선순위와 중요도를 고려하여 한계시간, 한계효율, 탐지빈도를 결정할 수 있음을 보았고, 회복법에서 대기 그래프(WFG)에 있는 사슬을 끊어 주기 위하여, 사슬을 찾아내고 그 트랜잭션들 가운데 시간 및 비용에 관해 효과적인 최적의 희생자를 선정하여 중단시키고, 그 트랜잭션을 이전의 상태로 복귀시켜 주는 후속 작업을 전체적인 과정에서 트랜잭션의 시간적 특성을 고려하여 검토하였다. 예방법은 네 가지 필요조건 가운데 적어도 하나를 거부하는 것으로, 그 가운데 2PL의 상호배제 특성을 제거한 낙관적 접근법(OCC, OCCL)과 2PL-OS, 통합실시간록킹법 등을 소개하였다. 실험적 결과에 따르면 OCCL은 자료의 경쟁이 낮을 때 우수하고 2PL은 높을 때 우수하며 구현상에 있어서 OCCL이 2PL에 비하여 결코 우수하지 못하다. 이러한 새로운 방안은 교착상태가 블럭킹의 특수한 한 유형이라는 점에서 시간제약에 중요한 요소인 블럭킹을 제거함으로 인하여 블럭킹과 함께 교착상태도 해결하고 있다. 회피법은 트랜잭션의 행동을 미리 알아내므로써 교착상태를 피하는 방법으로 아직 연구가 없는 편이지만 운영체제의 "뱅커스 알고리즘" 등이 적용 가능할 것으로 본다. 그리고 록킹장치를 할 자료의 크기가 병행성에 영향을 미치며, 이것은 간접적으로 블럭킹을 줄여 주기 때문에 그에 비례하여 교착상태의 가능성이 줄어든다. 그러나 아직 그 단위에 대한 모델이 없다는 점에서 추후 연구가 필요할 것으로 본다. 그리고 교착상태와 유사한 현상을 갖는 기아상태를 예방하는 방법도 실시간 차원에서 고려한 연구 결과를 소개하였다.

본 논문은 운영체제에서 심도있게 논의된 교착상태를 실시간 트랜잭션 스케쥴러의 차원에서 검토하였다. 핵심은 실시간 시스템이 추구하는 목적이 규정시간에 적중하는 트랜잭션의 수를 최대로 하기 위하여 병행수행 환경에서 지연을 최소화하는 데 역점을 두었다. 그러나 이러한 사항은 부분적으로 연구가 이루어지긴 하였으나 전반적으로 체계를 갖추어 논의되지 않고 있다. 앞으로 연구가 요구되며, 특히 실시간 제어를 필요로 하는 분야에서 중요한 관심사가

될 것으로 본다.

참 고 문 헌

1. A. Agrawal, A. EL Abadi and R. Jeffers, "Using Delayed Commitment in Locking Protocols for Real-Time Databases", ACM SIGMOD, pp. 104~113, 1992.
2. P. A. Bernstein, V. Hadzilacoos and N. Goodman, "Two Phase Locking" in Concurrency Control and Recovery in Database System, Addison-Wesley Pub., pp. 47~11, 1987.
3. P. A. Bernstein, V. Hadzilacoos and N. Goodman. "Non-Locking Scheduler" in Concurrency Conrol and Recovery in Database System, Addison-Wesley Pub., pp. 113~142. 1987.
4. A. P. Buchmann, D. R. McCarthy, M. Hsu and U. Dayal, "Time_Critical Database Scheduling: A Framework For Integrating Real-Time Scheduling and Concurrency Control, Data Engineering, IEEE, 1989.
5. Sharma Chakravarthy, "Rule Management and Evaluation: An Active DBMS Perspective", in SIGMOD RECORD, Vol. 18, No. 3, 1989.
6. S. E. Chodrow and F. J. Marc Donner, "Run-Time Monitoring of Real-Time System, in Proc. REAL_TIME SYSTEMS SYMPOSIUM. IEEE, DEC., 1991.
7. C. C. fleming and B. V. Halle, Handbook of Relational Database Design. Addison-wesley Pub., pp. 534~535, 1989.
8. J. R. Haritsa, M. Linvy and M. J. Carey. "Earliest Deadline Scheduling For Real-time Database Systems". in Proc. REAL_TIME SYSTEMS SYMPOSIUM, IEEE, DEC. 1991.
9. Jiandong Huang and John A. Stankovic, "On Using Priority Inheritance in Real-Time Databases" in Proc. REAL-TIME SYSTEMS SYMPOSIUM. IEEE, DEC. 1991.
10. J. Haung and J. A. Stankovic, "Experimental Evaluation of Real-Time Optimistic Concurrency Control Schemes", Conference on VLDB, 1991.
11. M. Maegawa, A. E. Oldehoeft and R. R. Oldehoeft, "Deadlock" in Operating Systems: Advanced Concept, The Benjamin/Cummings Pub., pp. 101~131, 1987.
12. N. E. Miller, "Blocking and Buffering" in File Structures Using PASCAL", The benjamin/cummings Pub. co., Inc.. pp. 72~95, 1987.
13. S. H. Son, Y. Lin, and R. P. Cook. "Concurrency Control in Real-Time Database System in Foundations of real-time Computing: Scheduling and Resource Management, Kluwer academic Pub., 1991.
14. S. H. Son, S. Park, and Yi Lin, "An Integrated Real-time Locking Protocol", proc. of Data Engineering, pp. 527~534, 1992.
15. J. A. Stakovic, K. Ramamritham and D. Towsley, "Scheduling In Real Time Transaction Systems" in Foundations of real-time Computing: Scheduling and Resource Management, Kluwer acaemic Pub., 1991.
16. 김길창, 운영체제개념, 정익사, pp. 304~340, 1988.
17. 김정기, 장재우, 예측취소 기법을 이용한 실시간 데이타베이스의 트랜잭션 스케줄링, '92 정보과학회 가을 학술발표 논문집, Vol. 19, No. 2, pp. 109~112, 1992.
18. 변정용, 김경창, 임해철, 실시간 트랜잭션 처리에서 규정시간 초과 예측, '93 동계 데이타베이스 연구회 학술발표 논문집, 1993.

변 정 용

1980 동국대학교 전자계산학과 학사
1983 동국대학교 전자계산학과 석사
현재 동의대학교 전자계산학과 박사과정 재학중
1982 ~1987 한국전자통신연구소 컴퓨터기술연구단 연구원
1988 ~현재 동국대학교(경주) 전자계산학과 조교수
관심 분야 : 데이타베이스(실시간, 능동, 객체지향), 한글공학

김 경 창

1978 홍익대학교 전자계산학과 학사
1980 한국과학기술원 전산학 석사
1990 미국 택사스대학(오스틴) 전산학 박사
1980 ~1983 경제기획원 조사 통계국 사무관
1990 ~1983 미국 해군대학원 객원 교수
관심 분야 : 객체지향 데이타베이스, 멀티미디어 시스템, 분산 시스템, 소프트웨어 공학

임 해 철

1976 서울대학교 계산통계학과 학사
1978 한국과학기술원 전자계산학과 석사
1988 서울대학교 컴퓨터공학과 공학박사
1978 ~1981 현대 엔지니어링 대리
1989 ~1990 University of Florida Post-Doc
1989 ~현재 홍익대학교 공과대학 컴퓨터공학과 부교수
관심 분야 : 객체지향 데이타베이스, 실시간 데이타베이스, 분산 데이타베이스 불완전 데이타베이스