韓國 컴퓨터情報學會 論文誌
第8卷 第3號, 2003. 9.

2003-8-3-1-4

분산 데이터베이스 시스템에서의 교착상태 탐지기법의 성능평가

이 원 섭*, 이 상 희**

Performance Comparision of Deadlock Detection Schemes in Distributed Database Systems

LEE, WONSUP*, LEE, SANGHEE**

요 약

Choudhary의 선분추적 교착상태 탐지 알고리즘에서는 존재하는 교착상태를 찾지 못하는 경우가 있다. 이 문제를 수정한 알고리즘을 제안했었다. 본 논문에서는 수정 알고리즘과 트랜잭션-자원 그래프(TR graph)를 사용하는 Tsai의 알고리즘과의 성능을 비교하였다.

Abstract

The edge-chasing deadlock detection algorithm of Choudhary fails to remove the existing deadlocks after committing the transaction whose priority is lowest on the transaction wait-for path. We proposed a modified algorithm that solves this problem. In this thesis, the performance of the modified algorithm is compared with that of the Tsai's deadlock detection algorithm that uses transaction-resource graph(TR graph) using simulation approach.

▸ Keyword : 분산데이터베이스, 분산교착상태탐지, 선분 추적, 트랜잭션 자원 그래프, 성능평가

* 인덕대학 전자과 조교수
** 청강문화산업대학 컴퓨터소프트웨어과 조교수

Ⅰ. 서 론

데이터베이스 관리 시스템(database management system)은 동시에 여러 트랜잭션이 실행되는 환경을 제공하므로 데이터베이스의 정확성을 유지하기 위해서는 동시성 제어가 필요하다. 동시성제어기법에는 로킹기법[1], 타임스탬프 순서기법[2], 그리고 낙관적 기법[3]이 있으나 일반적으로 사용되는 기법은 로킹이다. 그러나 로킹은 자원의 독점을 허용하기 때문에 교착상태(deadlock)가 발생할 수 있으므로 교착상태에 대한 처리가 필요하다.

교착상태의 처리기법에는 교착상태 방지, 교착상태 회피, 교착상태 탐지가 있다. 교착상태 방지와 회피 기법은 분산 데이터베이스와 같은 복잡한 환경에는 적용할 수 없으므로 교착상태 탐지 기법이 널리 사용된다.

분산 교착상태 탐지에는 트랜잭션 자원 그래프(transaction resource graph) 유지 기법, 선분추적(edge-chasing)기법, 타임 아웃(time out) 기법이 있으나, 정확하게 교착상태를 탐지하는 기법은 트랜잭션 자원 그래프유지 기법, 선분추적 기법이다.

대표적인 두 교착상태 탐지기법의 성능과 특성을 비교하기 위해 성능평가를 하였다.

분산데이터베이스 환경에서 교착상태 탐지 알고리즘의 성능평가를 한 논문으로 [11], [12]가 있다. [11]은 이질형 데이터베이스 환경을 가정하여 교착상태 알고리즘의 성능평가에 관한 연구이고, [12]는 멀티데이터베이스 환경에서 타임아웃 기법 중심으로 성능평가를 하였다.

본 논문은 분산데이터베이스 환경하에서 성능평가의 대상으로 트랜잭션자원 그래프 기법의 대표적 알고리즘인 [6]과 선분추적 기법의 대표적인 알고리즘인 [4]에 대한 성능을 시뮬레이션을 통해 평가하던 중에 [4]의 선분 추적을 기반으로 한 교착상태 탐지 알고리즘에서 오류를 발견하고 그 오류를 수정한 새로운 알고리즘을 [5]에서 제시했으며, [6]의 트랜잭션 자원 그래프 유지방법과 [5]에서 제안한 수정 알고리즘의 성능 평가하겠다.

Ⅱ. 관련연구

[6]의 알고리즘은 교착상태 결정과 사이트간의 메시지 교환 전략으로 구성된다. 교착상태 결정과정은 트랜잭션과 자원사이의 요청과 배당 관계를 표현하는 축소된 트랜잭션 자원 그래프(Reduced Transaction Resource Graph : RTR Graph)에서 사이클의 존재확인으로 결정된다. 사이트간의 메시지 교환전략은 도달쌍 메시지의 발생과 특정한 메시지에 붙이는 타임스탬프의 이용부분으로 구성된다. 도달쌍 (T,R)은 트랜잭션 T가 자원 R을 이행적으로 기다리는 것을 의미한다. Tsai알고리즘에서는 요청 선분과 배당 선분 외에 TWF(Transaction Wait-For)선분, 자원 도달 선분(Resource reaching edge), 트랜잭션 도달 선분(Transaction reaching edge)이 부가적으로 사용된다. [6] 알고리즘에 대해 알아보자.

사이트 Sk에서 트랜잭션 T가 요청한 자원 R에 대해서 로크 허용이 불가능하여 요청선분(R, T)가 첨가되거나, 다른 사이트로부터 받은 도달쌍(T, R)을 그래프 상에 첨가함으로 인해 TWF선분 (T, T1)이나 트랜잭션 도달선분(T, T1)이 발생했을 때만 도달쌍에 대한 고려가 필요하다. 여기서 T1은 자원 R을 이미 배당받은 트랜잭션이다.

선분(R, T)와 (T, T1)의 첨가로 인해서 T, T1, . . ., Tn으로 이어지는 경로가 구성될 때 Tn에서 Rn으로 나가는 선분 즉 요청선분이 있으면 X = Rn이 되고 Tn으로 나가는 선분이 없으면 X = R이 된다. T를 거쳐서 X에 도달하는 그래프 상의 모든 Ti에 대해서 다음을 수행한다.

여기서 Sorg(T)는 T가 제기된 사이트를 의미하고, Sorg(R)은 R이 있는 사이트를 의미한다.

① Ti가 Sorg(Ti) ≠ Sorg(R')인 자원 R'를 가지고 있으며, Sorg(R') ≠ Sk면 도달메시지 (Ti, X)를 Sorg(R')으로 전송한다.

② Sorg(Ti) ≠ Sk고 Ti ≠ T인 경우 (Ti, X)를 Sorg(Ti)로 전송한다.

③ Sorg(Ti) ≠ Sk이고 Ti = T이고 X = Rn인 경우는 (Ti, X)를 Sorg(Ti)로 전송한다.

선분 추적 기법에 의한 교착상태 탐지의 기본 개념은 실제 트랜잭션 자원 그래프의 유지 없이 그래프의 선분에 해당하는 경로를 특수한 메시지인 probe가 추적하여 사이클의 형성 유무를 확인한다. Sinha[7]는 타임 스탬프를 사용하여 프로세스간에 교환되는 메시지 수를 감소시켰다. 시간의 관점에서 볼 때 항상 타임 스탬프가 큰, 즉 우선순위가 낮은 트랜잭션이 타임 스탬프가 작은, 우선순위가 높은 트랜잭션을 기다리면 교착상태는 절대 발생하지 않는다.

만일 타임스탬프가 작은 트랜잭션이 타임스탬프가 큰 트랜잭션을 기다리는 상황을 역충돌(antagonistic conflict)이라 하고 이때 probe를 발생시킨다. Sinha는 타임스탬프를 도입하여 메시지 수는 줄였으나, [4]에서 지적한 바와 같이 탐지하지 못하는 교착상태와 실제로 존재하지 않는 교착상태를 탐지하는 거짓 교착상태(false deadlock) 문제를 해결하지 못한다.

[4]에서는 자원에 대한 로크를 가진 트랜잭션을 소유자라 하고 요청 큐에서 기다리는 트랜잭션을 요청자라 한다. Probe는 (발생자, 주니어)로 구성된다. 발생자는 역충돌 발생시 probe를 발생시킨 트랜잭션의 번호이다. 주니어는 probe가 지나가는 경로에 있는 트랜잭션들 중에서 우선순위가 가장 낮은 트랜잭션으로 교착상태가 발생할 경우 희생자가 된다.

[4]에서 제안한 알고리즘에도 존재하는 교착상태를 탐지하지 못하거나, 교착상태를 탐지해도 제거하지 못하는 오류가 발견되었다. [5]에서 오류를 수정한 새로운 알고리즘을 다음에 제시하였다. 아래에 [4]의 알고리즘을 수정한 알고리즘이다. 수정된 부분은 고딕체로 표기하였다.

1. 데이터 관리자가 아래 메시지를 받을 때 :

(1) 로크 요청 메시지

로크 허용이 가능하면 요청한 트랜잭션에 로크를 허용한다. 로크 허용이 불가능하면 요청자〉소유자인 경우에 probe를 발생시켜 소유자에게 전송한다.

(2) 로크 해제 메시지

요청 큐에 요청자가 있으면 로크를 허용한다. 요청 큐에 요청자가 남아 있으면 요청자〉소유자인 경우 probe를 발생시킨다. R의 로크를 해제한 트랜잭션 T가 희생자 후보인지를 확인한다. T가 희생자 후보이면 요청 큐의 나머지 요청자들에게 각 요청자의 probe큐의 사본을 다시 보낼 것을 요청한다.

(3) probe

소유자〉발생자이면 받은 probe를 없애고, 소유자〈발생자이면 probe를 소유자에게 전송하며, 소유자 = 발생자이면 철회 메시지를 희생자에게 전송한다.

(4) clean 메시지

Clean 메시지를 소유자에게 전송하고, 각 요청자에 대해서 요청자〉소유자인 경우 probe를 재 발생시킨다. 요청 큐안의 각 요청자들에게 요청자의 probe 큐내의 probe들의 사본을 재전송할 것을 요청한다.

단, 요청자가 희생자인 경우는 probe 큐의 사본을 요청하지 않는다.

2. 트랜잭션 관리자가 아래 메시지를 받으면:

(1) probe

쥬니어〉T이면 쥬니어를 T로 바꾸고 probe 큐에 저장한다. T가 대기상태이면 저장된 probe의 사본을 T가 기다리는 자원을 관리하는 데이터관리자에게 전송한다.

(2) 대기 메시지

T는 probe 큐에 있는 probe들의 사본을 T가 기다리는 자원을 관리하는 데이터 관리자에게 전송한다.

(3) 철회 메시지

Clean 메시지를 T가 기다리는 데이터 관리자에게 전송하고 T의 probe 큐를 T가 기다리는 데이터 관리자에게 전송한다.

(4) clean 메시지

Probe 큐의 모든 probe를 전부 없앤다. T가 대기상태이고 희생자가 아니면 clean 메시지를 T가 배당되기를 기다리는 자원을 관리하는 데이터 관리자에게 전송한다. T가 배당 받은 자원이 가진 요청 큐안의 트랜잭션들 중에서 요청자〉소유자인 경우 다시 probe(요청자, 소유자)를 발생시키고, 각 요청 자에게 요청자가 가진 probe 큐의 사본을 요청한다. T가 대기상태이고 희생자이면 철회 단계에 들어간다. T는 모든 로크를 해제하고 자원에 대한 요청을 취소한다. 철회 단계 동안에는 T가 받은 어떤 메시지도 버린다.

(5) 배당 메시지

실행시킬 연산이 남아 있으면 연산에 필요한 자원 R에 대한 로크 요청 메시지를 R을 관리하는 데이터 관리자에게로 전송한다. 트랜잭션의 실행이 완전히 끝났으면 T가 희생자후보 즉 T의 probe 큐의 probe들의 주니어인지를 확인한다. T가 쥬니어가 아니면 자원을 해제할 때 트랜잭션은 probe 큐의 모든 probe를 복사하여 로크를 해제하는 자원을 관리하는 데이터 관리자로 전송한다.

T가 주니어면 로크를 해제할 자원을 관리하는 데이터 관리자로 로크 해제 메시지를 전송할 때 T가 희생자 후보라는 정보를 같이 전송한다.

좀 더 자세한 알고리즘은 [6], [4]을 참조하기 바란다.

III. 시뮬레이션 환경

1. 가정

통신망에 존재하는 각 사이트는 유일한 사이트 번호를 가지고 있으며, 모든 사이트들 간에는 직접 통신이 가능하다. 통신망은 무결함하며 메시지는 보내준 순서대로 받으며 제한된 시간 내에 목적지에 도착한다.

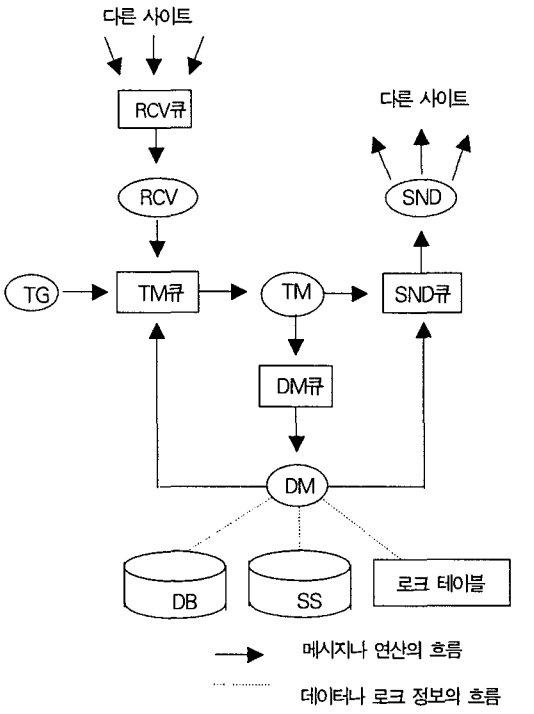
동시성 제어 기법으로는 write 로크만을 이용하는 B2PL(basic two phase locking)을 사용하고 완료프로토콜은 2PC(two phase commit)를 사용한다[8][9][10]. 모든 데이터는 전혀 중복이 없으며 데이터베이스와 로크 테이블은 보조 기억 장치에, 트랜잭션-자원 그래프는 주기억장치에 상주한다.

트랜잭션 번호는 고유하게 부여되며, 한 데이터 항목을 기록하기 위해서는 쓰기를 하기 전에 반드시 읽기를 해야 한다. 특정 자원에 대해서 두 번 로크를 요청하지 않으며 각 트랜잭션은 요청한 자원에 대한 로크가 허용되기 전에는 다른 자원에 대해 로크를 요청하지 않는다.

2. 트랜잭션 처리 모델

본 논문에서 [6][4]의 성능 평가에 [1]에서 제안한 모델을 기반으로 했다.

<그림 1>의 트랜잭션 처리 모델에서 타원은 처리기(proceesor로서 각각 자신의 큐를 가지고 있으며, 각 큐에 잇는 메시지들은 FIFO(first in first out) 순서로 하나씩 처리된다.



약어 설명	
	트랜잭션 생성기(transaction generator : TG)
	트랜잭션 관리자(transaction manager : TM)
	데이터 관리자(data manager : DM)
	메시지 송신자(message sender : SND)
	메시지 수신자(message receiver : RCV)
	데이터베이스(database : DB)
	로크 테이블(lock table)
	안전 기억장치(secure storage : SS)

그림 1. 트랜잭션 처리 모델

트랜잭션 생성기는 지수 함수 분포를 갖는 평균 트랜잭션 도착 시간 (mean transaction interarrival time)당 하나의 트랜잭션을 만들어서 그 트랜잭션을 지역 트랜잭션 관리자에게 보낸다.

트랜잭션은 시작(T_begin), 읽기(read), 쓰기(write), 종료(T_end)의 조합으로 구성된다. 한 트랜잭션 내에서는 같은 자원에 두 번 로크를 요청하지 않는다. 트랜잭션 관리자는 트랜잭션 생성기로부터 트랜잭션을 받으면 그 트랜잭션을 트랜잭션 리스트에 첨가한다. 트랜잭션 관리자는 트랜잭션 연산에 필요한 자원에 대한 로크 요청 메시지를 데이터 관리자에 보내고 그 트랜잭션을 대기 상태로 만든다. 트랜잭션이 실행 도중에 교착상태의 희생자로 선정되어 철회 메시지를 받으면 그때까지 배당 받은 모든 자원을 해제한 후에 그 트랜잭션을 처음부터 다시 시작한다.

데이터 관리자는 실제로 지역 데이터베이스에 액세스하고 갱신하는 기능과 지역 트랜잭션 관리자와 메시지 수신자로부터 받은 메시지를 내용에 따라, 로크 테이블, 트랜잭션 자원 그래프를 관리하고 교착상태를 탐지하는 기능도 함께 한다. 메시지 송신자와 메시지 수신자는 사이트들 간에 메시지를 통신 시간(communication time) 동안 지연시킨 후에 다른 사이트의 메시지 수신자에게 보내며, 메시지 수신자는 받은 연산이나 메시지를 지역 트랜잭션 관리자 혹은 데이터 관리자로 보낸다.

3. 입력 매개 변수 및 성능 지수

본 논문에서는 시뮬레이션을 통해서 성능을 비교하기 위해서 사이트의 수, 사이트 당 실행된 트랜잭션의 수, 데이터베이스의 크기, 트랜잭션의 최대 길이, 디스크 입출력 시간, 통신 지연 시간, 평균 트랜잭션의 도착 시간, 전역 자원 요청 비율을 입력 매개변수로 사용하였다.

사이트의 수는 분산 시스템을 구성하는 사이트의 수로 본 성능 평가에서는 3개로 고정하였고, 사이트 당 수행되는 트랜잭션의 수는 각 사이트 당 1000개로 제한하였다. 데이터베이스의 크기는 데이터베이스 내에 존재하는 데이터 항목 수로 500으로 고정했으며, 트랜잭션의 최대 길이는 8로 평균 트랜잭션 길이는 4이다. 디스크 입출력 시간은 보조기억 장치에 저장되어 있는 한 데이터 항목을 읽거나 갱신하는데 소요되는 시간으로서 30ms로 하였다.

통신 지연 시간은 하나의 메시지가 다른 사이트로 전송되는데 소요되는 시간으로 본 연구에서는 30ms, 60ms, 120ms로 하였다. 트랜잭션의 평균 도착 시간은 본 연구에서는 트랜잭션 평균 도착 시간을 500ms, 1000ms, 1500ms, 2000ms로 정했으며, 실제 시스템과 달리 동시에 처리할 수 잇는 트랜잭션의 수가 제한되지 않으므로 트랜잭션의 평균 도착 시간이 작아지면 충돌 가능성이 급격하게 증가한다. 전역 자원 요청 비율(global resource request ratio)은 전역 자원 요청 수를 전체 자원 요청 수로 나눈 것으로 정의한다.

전역 자원 요청이란 트랜잭션이 제기된 사이트가 아닌 다른 사이트에 존재하는 자원으로 요청하는 것이다. 전역 자원 요청 비율은 0, 0.25, 0.5, 0.75, 1로 변화시키면서 실험하였다.

교착상태 탐지 기법의 성능을 비교하기 위해서 측정되는 성능 지수로는 평균 트랜잭션 응답 시간을 선택하였다. 평균 트랜잭션 응답 시간은 트랜잭션이 제기되어서 끝날 때까지 소요된 시간의 평균이다.

이 지수는 두 교착상태 탐지 기법의 성능을 비교하는 가장 중요하고 총괄적인 성능 지수이다. 이외에도 평균 트랜잭션 충돌율, 교착상태 탐지를 위해 사이트 간에 교환된 메시지의 수, 교착상태 제거하기 위해 사이트 간에 교환된 메시지의 수, 메시지 당 평균 큐잉 지연 시간, 메시지 당 평균 통신 지연 시간 등의 성능지수를 측정하였으나 본 논문에서는 지면의 제약으로 생략하겠다.

Ⅳ. 시뮬레이션 결과 및 분석

본 장에서는 4장에서 설명한 트랜잭션 처리 모델과 입력 매개 변수를 사용하여 [6]와 본 논문에서 수정한 Choudhary의 알고리즘에 대하여 전역 자원 요청 비율, 통신 지연 시간, 평균 트랜잭션 도착 시간의 변화에 따른 트랜잭션의 응답 시간의 변화를 분석한다. 시뮬레이션 프로그램은 Path Pascal로 작성하였으며 Solaris가 설치된 SUN 워크스테이션에서 실행하였다.

1. 전역 자원 요청 비율 변화에 따른 트랜잭션 평균 응답 시간의 변화

전역 자원 요청 비율이 증가함에 따라 트랜잭션 평균 응답 시간의 변화를 조사하기 위해서 시뮬레이션으로부터 얻은 결과들 중에서 [6]와 수정된 알고리즘의 성능을 명확히 나타낼 수 있는 평균 트랜잭션 도착 시간 = 1000ms, 통신 지연 시간 = 60ms인 경우를 선택하여 〈그림 2〉에 수록하였다.

〈그림 2〉에 의하면 전역 자원 요청 비율이 증가함에 따라 수정된 알고리즘과 [6]의 트랜잭션 응답 시간은 모두 증가하는 추세를 보이고 있다.

전역 자원 요청 비율이 0일 때, 즉 각 사이트에 제기된 트랜잭션은 트랜잭션이 제기된 사이트의 자원만을 요청할 경우, 수정된 알고리즘은 트랜잭션 평균 응답 시간이 1711ms이고, Tsai알고리즘은 1182ms로 Tsai 알고리즘이 좋다.

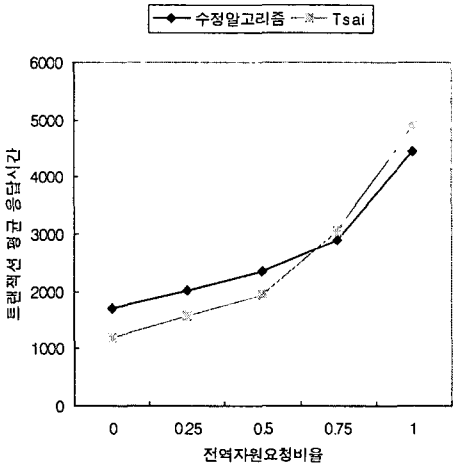


그림 2. 전역 자원 요청 비율 변화에 따른
트랜잭션 평균 응답 시간의 변화
(평균트랜잭션도착시간 = 1000ms, 통신지연시간 = 60ms)

전역 자원 요청 비율이 1인 경우 트랜잭션의 평균 응답 시간은 Tsai 알고리즘이 4899ms이고 Chouodhary 알고리즘이 4448ms로 전역 자원 요청 비율이 0일 때와 결과는 반대된다. 그 이유는 전역 자원 요청 비율이 크면 사이트 간에 교환되는 메시지의 수가 많아져서 통신 지연 시간이 증가한다. 그리고 전역 자원 요청 비율이 크면 지역 사이트에서 처리되는 연산의 수가 줄어들고 다른 사이트로 보낸 메시지에 대한 응답이 통신 지연 시간의 증가로 지연됨에 따라 지역 사이트의 프로세스 큐 내의 메시지 수는 감소하여 큐잉 지연 시간이 감소하게 된다. 앞의 두 이유로 인하여 통신 지연이 큐잉 지연 시간보다 커짐에 따라 큐잉 지연 시간이 트랜잭션 응답 시간에 미치는 영향도 감소한다. 즉 앞에서 설명한 정보 저장 장소로 인하여 수정된 알고리즘의 불리한 점이 감소한다. 또한 Tsai 알고리즘은 수정된 알고리즘보다 교착상태 탐지를 위해서 교환되는 정보의 흐름에 있어서도 불리하다.

수정된 알고리즘의 경우 트랜잭션 T가 다른 사이트에 있는 자원 R에 대한 로크를 요청한 경우 트랜잭션 관리자는 요청 메시지를 자원 R을 관리하는 데이터 관리자에게로 보내고 데이터 관리자는 로크 허용이 가능하면 배당 메시지를, 허용이 불가능하면 대기 메시지를 트랜잭션 관리자로 전송한다. 그러나 Tsai의 알고리즘에서는 요청메시지를 우선 지역 데이터 관리자에게 보내서 지역 데이터 관리자가 유지하는 그래프에 T에서 R로의 요청 선분을 첨가 후 요청 메시지를 자원 R이 있는 사이트의 데이터 관리자에게 전송한다. 데이터 관리자는 자원 R이 배당 가능하면 그래프를 갱신하고 배당 메시지를 트랜잭션 T가 제기된 사이트의 데이터 관리자로 전송한다. 배당 메시지를 받은 데이터 관리자는 T와 R 사이의 요청 선분을 배당 선분으로 대체한 후 배당 메시지를 트랜잭션 관리자로 보낸다. 이것을 수정된 알고리즘과 비교하면 사이트 간에 교환되는 메시지의 수는 같으나 지역 사이트 내에서 교환되는 메시지 수가 2개 더 많은 것이다. 트랜잭션의 평균 길이가 4이고 전체 실행된 트랜잭션의 수가 200개이다. 전역 자원 요청 비율이 1인 경우는 Tsai의 알고리즘은 수정된 알고리즘보다 지역 프로세스 사이에 1600개의 메시지가 더 교환된다. 따라서 전역 자원 요청 비율이 증가하면 Tsai 알고리즘은 불리하게 된다. 그림 2에서 두 곡선이 교차하게 되는 것은 전역 자원 비율이 낮으면 통신 지연 시간이 트랜잭션 응답 시간에 큰 영향을 미치지 못하므로 큐잉 지연 시간이 짧은 Tsai 알고리즘이 유리하고, 전역 자원 비율이 큰 경우는 큐잉 지연 시간과 통신 지연 시간이 비슷하여 지역 사이트 내의 프로세스 사이의 메시지 교환 정책에서 불리한 Tsai 알고리즘이 불리하기 때문에 발생한다.

2. 평균 트랜잭션 도착 시간 변화에 따른 트랜잭션 평균 응답 시간의 변화

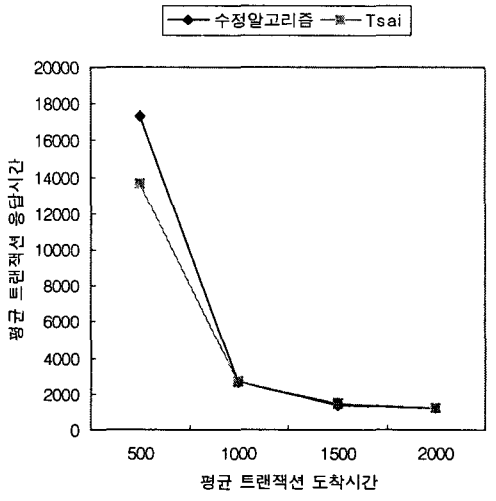


그림 3. 평균 트랜잭션 도착 시간 변화에 따른 트랜잭션 평균 응답 시간의 변화
(전역자원요청비율 = 0.25, 통신지연시간 = 120ms)

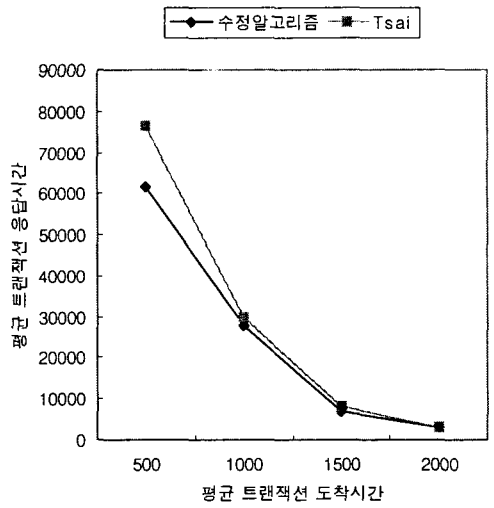


그림 4. 평균 트랜잭션 도착 시간 변화에 따른 트랜잭션 평균 응답 시간의 변화
(전역자원요청비율 = 0.75, 통신지연시간 = 120ms)

〈그림 3과 4〉는 트랜잭션의 평균 도착 시간에 대한 [6]와 수정된 알고리즘의 트랜잭션 응답 시간의 변화를 그래프로 표현한 것이다. 〈그림 3〉은 전역 자원 요청율이 0.25, 그림 4는 0.75 이고 통신 지연 시간은 120ms이다. 〈그림 3과 4〉의 그래프에서 일반적인 예측과 같이 트랜잭션의 도착시간이 짧을수록 응답시간이 길어지는 것을 알 수 있다. 하지만 그림 3에서는 [6] 알고리즘이 〈그림 4〉에서는 수정 알고리즘의 성능이 좋은 것으로 나타난다. 이것은 그림 2에서와 같은 이유이다.

V. 결 론

본 논문에서는 분산 데이터베이스 시스템에서 교착상태 탐지에 사용되는 대표적인 두 가지 기법, 트랜잭션 자원 그래프를 이용한 기법과 선분 추적 기법의 대표적인 알고리즘인 [6]와 [4]의 수정 알고리즘의 성능을 시뮬레이션을 통해서 평가하였다. 성능 평가는 전역 자원 요청, 비율 평균 트랜잭션 도착 시간과 통신 지연 시간의 변화에 따른 트랜잭션 평균 응답 시간을 측정하고 비교하였다.

전역 자원 요청 비율의 변화에 따른 평균 응답 시간의 변화는 전역 자원 요청 비율이 작을 때는 [6]의 트랜잭션 평균 응답 시간이, 전역 자원 요청 비율이 클 때는 수정된 알고리즘의 트랜잭션 평균 응답 시간이 좋은 것으로 나타났다. 평균 트랜잭션 도착 시간은 시스템내의 트랜잭션 수에 영향을 주고, 시스템내의 트랜잭션의 수는 충돌율과 관계에 있으므로 트랜잭션 도착 시간은 트랜잭션 응답 시간에 영향을 준다.

통신 지연 시간이 증가하면 모든 경우에서 트랜잭션의 평균 응답 시간은 증가한다. 또한 전역 자원 요청 비율이 증가하면 각 트랜잭션은 다른 사이트와 메시지 교환수가 증가하게 되며 그 영향으로 트랜잭션 응답 시간도 증가한다.

참고 문헌

[1] Bernstein, P. A., and Goodman N., "Concurrency Control in Distributed Database System," ACM Computing Surveys, vol. 13, no. 2, pp. 185- 221, Jun. 1981.

[2] Bernstein, P. A., and Goodman N., "Timestamp Based Algorithm for Concurrency Control in Distributed Database Systems," Proceedings of Very Large Database Systems, pp. 285-300, Oct. 1980.

[3] Kung, H. T. and Robinson, J. T., "On Optimistic Methods for Concurrency Control. "ACM Transactions on Database Systems, vol.6, no. 2, pp. 213-226, Jun.1981.

[4] Choudhary, A. N., Kohler W. H., Stankovic J. A., and Towsley D., "A Priority Based Prove Alogorithm for Distributed Deadlock Detection and Resolution," The 7th international Conference on Distributed Computing Systems, pp. 162-168. Sept. 1987.

[5] 이원섭, "분산 데이터베이스 시스템에서의 교착상태 탐지기법," 한국컴퓨터정보학회 논문지, 6권 2호, 2001년 6월

[6] Tsai, W. C., "Distibuted Deadlock Detection in Distributed Database Systems," Ph. D. Thesis, University of Illinois at Urbana-Champain, 1982.
[7] Sinha, Mukul K. and Natarajan, N., "A Priority Based Distributed Deadlock Detection Algorithm," IEEE Trans.of Software Engineering, Vol. SE-11, pp. 67-80, Jan. 1985.
[8] Ceri, S., and Pelagatti G., Distributed Databases : Principles and Systems, McGraw-Hill Inc., 1984
[9] Moon, S. C., "Performance of Concurrency Control Methods Distributed Database Management Systems," Ph. D. Thesis, University of Illinois at Urbana-Champain, 1985.
[10] Singhal, M. and Agrawala, A. K., "Synchronization Techniqies in Distributed Database Systems," TR-1396, University of Maryland, May 1984.
[11] 남홍우, "이질형 데이타베이스 체계의 교착상태 검출," 석사논문, 한국과학기술원, 1992.
[12] 이인선, " 멀티 데이타베이스 시스템에서의 교착상태 해결방안의 성능평가," 석사논문, 한국과학기술원, 1996.

저 자 소 개

이 원 섭
1999 ~ 현재
인덕대학 전자과 교수
〈관심분야〉 분산데이터베이스

이 상 희
1996 ~ 현재
청강문화산업대학 컴퓨터 소프트웨어과 교수
〈관심분야〉 분산데이터베이스, 데이터웨어하우스