2001년 한국정보처리학회 춘계 학술발표논문집 제8권 제1호

분산 데이터베이스 시스템에서의 트랜잭션 제어기법 비교

이혜경*, 김희완**, 박동순**,김응모**
*경인여자대학 멀티미디어정보학부
**성균관대학교 전기전자컴퓨터공학부
e-mail : rheehk@dove.kyungin-c.ac.kr hwkim@syu.ac.kr dspark@tongwon.ac.kr
umkim@yurim.skku.ac.kr

A Comparison of Transaction Scheduling Schemes in Distributed Database Systems

Hae-Kyung Rhee*, Hee-Wan Kim**, Dong-Soon Park**, Ung-Mo Kim**
*Dept. of Multimedia Information Computing, Kyungin Women's College
**Dept. of Electrical and Computer Engineering, Sungkyunkwan University

요 약

데이터베이스에서 트랜잭션의 처리량을 향상시키기 위해서 기존의 문법 위주의 직렬성 이론으로는 단위시간당 처리량을 높이기는 힘든 형편이다. 이를 위하여 이타적 잠금 기법(altruistic locking: *AL*)은 트랜잭션이 객체를 사용한 다음 더 이상 그 객체를 요구하지 않을 때 다른 트랜잭션들이 그 객체를 로크할 수 있도록 미리 객체에 대한 로크를 해제함으로써 트랜잭션들의 대기시간을 줄이고 처리량을 향상시키기 위한 취지에서 제안된 것이다. 확장형 이타적 잠금(extended altruistic locking: *XAL*)기법은 *AL* 을 자취의 확장한 것으로 *AL* 이 근본적으로 안고 있는 반드시 기부된 객체만을 처리해야 한다는 제약을 완화한 기법이다. 본 논문에서는 분산 데이터베이스 환경하에서 장기 트랜잭션으로 인한 단기 트랜잭션의 대기현상을 최소화하도록 함으로써 동시성 제어의 정도를 높일 수 있는 새로운 확장형 이타적 잠금 기법인 양방향 기부기법(two-way donation locking: 2DL)을 제안하고 기존의 제어기법과 비교하였다. 모의실험에 의한 성능평가 결과 2DL 은 작업 처리율과 트랜잭션의 평균 대기시간 면에서 우수한 결과를 나타내었다.

1. Introduction

데이터베이스 기술이 넓은 범위의 응용에 적용됨에 따라 처리량의 증가를 위한 트랜잭션 처리 모델의 개발이 요구되고 있다. 전통적인 직렬성 표기를 기반으로 한 두 단계 잠금 기법(2PL)은 데이터의 정확성을 보장한다. 그러나 단기 트랜잭션들과 장기 트랜잭션들을 함께 처리할 때, 로크(lock)로 인하여 방해를 받아 동시성 정도가 떨어진다. 2PL에서 로크의 늦은 해제는 장기 트랜잭션으로 하여금 교착상태(deadlock) 상황으로 될 수 있는 가능성이 있어서 트랜잭션의 수행 자체가 실패할 가능성이 높다. 그리고 단기 트랜잭션은 장기 트랜잭션에 의해 기아현상이나 라이브로크(livelock)로 기다려야 한다. 이러한 현상을 줄이기 위해 이타적 잠금 기법(AL : altruistic locking)이 제안되었다[1]. AL은 장기 트랜잭션이 사용을 마친 객체를 즉각 반납함으로써 다른 트랜잭션으로 하여금 기대했던 것보다 조기에 이 기부된 객체를 사용하게 하는 기법이다. 확장형 이타적 잠금 기법(XAL : Extended Altruistic Locking)은 AL을 확장한 것으로 장기 트랜잭션에 의해 앞으로 로크 되지 않을 객체를 요청하여 사용할 수 있도록 허용하는 것이다.[1] 제안한 양방향 기부제어 기법(2DL : two-way donation locking)은 XAL을 확장한 것으로 분산 데이터베이스 시스템에서 동시성의 정도를 향상시켰다.

2. 관련 연구

이타적 잠금 기법(AL)은 여러 개의 트랜잭션들이 동시에 객체를 로크할 수 있는 두 단계 잠금 기법(2PL)을 보완한 기법이다. 이타적 잠금 기법은 로크(Lock), 해제(unloak)와 함께 기부(donation)라는 새로운 동시성 제어연산을 제공한다. 기부는 객체를 로크한 트랜잭션이 더 이상 그 객체를 필요로 하지 않을 때 데이터베이스 관리자에게 그 객체를 다른 트랜잭션이 사용할 수 있도록 허락하도록 요청한다. 기부된 객체가 사용될 때, 객체를 기부한 트랜잭션은 새로운 객체를 계속 로크할 수 있다. 랜잭션들은 현재까지 로크된 객체들만이 기부를 할 수 있다. 그러나 이미 기부된 객체에 대해서는 처리를 할 수 없다. 기부는 해제의 대체 방안이 아니며 정상적인 트랜잭션은 몇 개의 객체를 기부했던지 상관없이 로크된 객체에 대해서만 해제를 해야 한다. 기부연산은 어떠한 트랜잭션이 해제를 하기 전에 객체를 기부함으로써 다른 트랜잭션으로 하여금 기부된 객체를 로크할 수 있도록 함으로 유용하게 사용된다.

확장형 이타적 잠금(XAL)은 이타적 잠금 기법에서 반드시 기부된 객체만을 처리해야 한다는 제약 조건을 완화한 기법이다. XAL 에서는 트랜잭션에 대한 작업을 계속적으로 수행하기 위해 다른 트랜잭션의 자취에 속한 트랜잭션들만을 수행할 수 있도록 허락하기 위한 목적으로 제안된 것이다. XAL 의 가정은 단기 트랜잭션은 장기 트랜잭션이 자취에서 사용한 객체까지도 자취 의존적으로 계속 객체를 접근하도록 하는 것이다.[1] 이러한 가정은 자취리스트먼저/다른 객체는 후에(data-in-wake-list-first/other-data-later) 접근 방법이라 부른다. 그러므로 XAL 에서는 다른 객체 먼저/ 자취 리스트 후에(others-first/wake-later) 접근 이 필요할 때는 필연적으로 수행속도가 늦어지게 된다.

3. 2DL 프로토콜

3.1 가정

자취 확장 방법을 상세하게 기술하기 위해 다음의 트랜잭션 관리 원칙을 가정한다.

1(*Data Replication*): 지역 응용에서 사용된 트랜잭션은 다른 사이트에서도 복사되어 사용된다.
2(*Donation Privilege*): 장기 트랜잭션만이 기부연산을 사용할 특권을 가진다.
3(*Commit Policy*): 장기 트랜잭션은 결국 commit 된다.
4(*Deadlock Handling*): 어떤 트랜잭션이 교착상태에 빠지게 되면, 그 트랜잭션은 timeout 기법을 사용하여 제거한다.

3.2 알고리즘

양방향 기부 잠금 기법은 다음과 같이 Pseudo-code 로 나타낸다. (자취 확장 알고리즘).

```
Algorithm(Wake Expansion Rule of 2DL)
Input: LT1; LT2; ST
                    /* ST:short trans; LT1, LT2:long trans */
BEGIN
 FOREACH LockRequest
   IF(LockRequest.ST.data = Lock)   THEN
                    /* Locks being requested by ST already
granted to long trans other than LT1 and LT2 */
        Reply:=ScheduleWait(LockRequest);
   ELSE IF(LockRequest.ST.data = Donated) THEN
                    /* Locks being requested by ST donated
by long trans other than LT1 and LT2 */
   FOREACH (ST.wake ∈ LT1 OR LT2)
    IF(ST.wake = LT1) THEN   /* Donation conducted by
LT1? */
      IF(ST.data ∈ LT1.marking-set) THEN   /* Data
being requested by ST to be later accessed by LT1 ? */
          Reply:=ScheduleWait(LockRequest)
      ELSE
          Reply:=SecheduleDonated(LockRequest)
      ENDIF
   ELSE
   IF(ST.data ∈ LT2.marking-set) THEN      /* Data
being requested by ST to be later accessed by LT2 ? */
          Reply := ScheduleWait(LockRequest)
   ELSE
          Reply:= SecheduleDonated (LockRequest)
   ENDIF
  ENDIF
  ENDFOR
  ELSE
          Reply := SecheduleLock (LockRequest)
  ENDIF
   IF(Reply = Abort) THEN /* Lock request of ST aborted
*/
       Abort Transaction(Transactionid);
       Send(Abort);
       Return();
   ENDIF
 ENDFOR
END
```

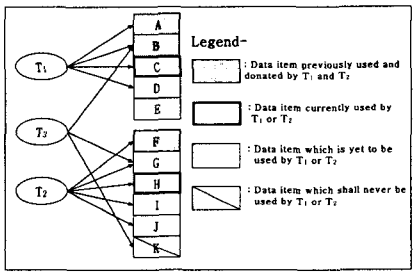
3.3 *2DL* 의 연산

XAL 을 사용하여 기부된 객체를 접근하고자 할 때, XAL 은 하나의 트랜잭션만이 객체를 요청할 수 있다. 그러나 2DL 에서는 단기 트랜잭션들은 하나의 기부 제약이 없음으로 기부된 객체를 더 자유롭게 접근할 수 있다. 단기 트랜잭션들은 다른 두 장기 트랜잭션에 의해 기부된 객체를 접근할 수 있다.

예 1 은 두개의 분산된 사이트의 장기 트랜잭션과 하나의 단기 트랜잭션을 가지고 2DL 을 적용한 것이다.

예 1(두 개의 동시에 수행되는 장기 트랜잭션에서 단기 트랜잭션의 처리를 허용하는 예): 장기 트랜잭션 *T1* 이 객체 *A, B, C, D* 와 *E* 를 차례로 접근한다고 가

정하자. *T1* 이 이미 객체 A, B 를 로크하였고, 성공적으로 기부했다고 하자. 현재 *T1* 은 객체 C 를 접근하고 있다고 하자. *T1* 을 따라서 동시에 두 개의 트랜잭션이 수행된다고 하자. 장기 트랜잭션인 *T2* 는 객체 *F, G, H, I* 와 *J* 를 순서적으로 접근하고자 하고, 단기 트랜잭션인 *T3* 는 *B, G, K* 를 순서대로 접근한다고 하자. *T2* 는 이미 로크되어 *F, G* 를 기부했다고 하자. 현재 *T2* 는 객체 *H* 를 접근하는 단계에 있다. (그림 1).



[그림 1] 두 개의 장기 트랜잭션과 단기 트랜잭션의 실행

XAL 을 적용하는 경우에, *T3* 가 객체 *B* 에 대한 로크는 허용되지만 객체 G 에 대한 로크는 이미 다른 장기 트랜잭션에 의해서 기부 되었기 때문에 허용되지 않는다. *T2* 가 Commit 된 후에만, G 는 *T3* 에 넘겨진다.

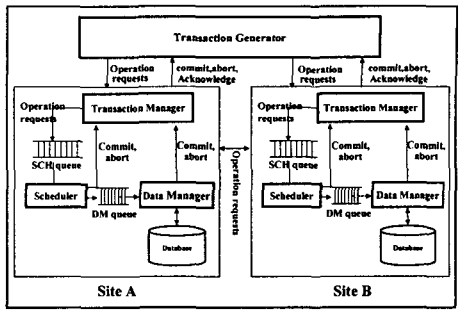
2DL 의 경우에는 *T3* 는 대기시간이 필요 없이 객체 G 를 접근할 수 있다. 이는 *T1* 의 자취 속에 *T2* 의 자취를 포함시켰기 때문에 가능하게 되는 것이다. 끝(예 1).

4. 성능 평가

4.1 모의실험 모델

모의실험 모델은 [그림 2]와 같이 각 사이트에는 다음과 같이 구성되어 있다. : 트랜잭션 생성기(TG: *transaction generator*), 트랜잭션 관리자(TM:*transaction manager*), 스케쥴러(SCH:*scheduler*), 데이터 관리자(DM: *data manager*), 데이터베이스(DB).

본 논문에서는 두 개의 분산된 사이트를 가지고 있으며, 각 사이트에는 각각의 시스템 모델을 가지고 있다. TG 는 하나씩 트랜잭션을 생성하여 일정시간 간격으로 TM 에게 보낸다. TM 은 SCH 에 연산 요청을 하고, DM 은 어느 객체를 사용할 것인지를 결정하기 위해 SCH 로부터의 연산을 분석한 후 요청된 객체를 저장할 디스크에 연산을 보낸다. 연산이 서버에서 완료되면, DM 은 요청된 연산이 완전히 끝났음을 알리는 메시지를 TM 에게 보낸다. 사이트 A 에서 트랜잭션의 처리가 완료되지 못하고 단절되면 그 트랜잭션은 사이트 B 에서 남은 객체를 처리한다.



[그림 2] 모의실험 모델

이 모델은 이산사건 모의실험(DEVS:discrete-event simulation) 언어인 *Scheme*[2]으로 구현했다. DEVS 에서는 기본 모델을 만들어서 더 큰 모델이 만들어지고, 이러한 모델들은 계층구조로 하나로 연결된다[4].

4.2 실험 방법

[표 1]은 실험에서 사용한 모델 인수들과 그 값들의 범위를 보여주고 있다. 인수의 값은 실세계의 컴퓨터 환경을 반영할 수 있도록 선택했다.

[표 1] 실험을 위한 인수들

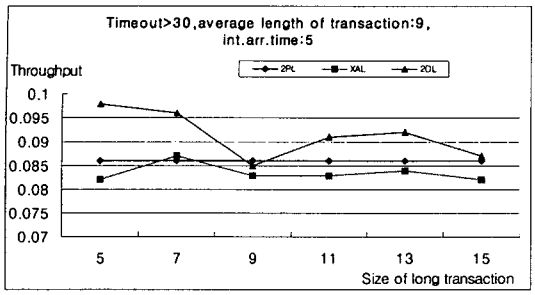
인수	값
사이트 수	2
데이터베이스 크기	100
CPU 수	2
Disk 수	4
단기 트랜잭션 크기	2, 3, 4, 5
장기 트랜잭션 크기	5, 7, 9, 11, 13, 15
실험시간 길이	300, 500, 700, 900, 1100, 1300, 1500

데이터베이스 크기는 충돌 정도에 영향을 준다. 데이터베이스 크기가 단기 트랜잭션과 장기 트랜잭션보다 크면, 충돌은 일어나지 않는다. *2PL, XAL* 과 *2DL* 의 성능을 평가하기 위해 트랜잭션에서 연산의 수를 표현하는 평균 트랜잭션 길이에 변화를 주었다.

CPU 와 Disk 의 수는 2 : 4 로 했으며, 이는 [3]에서 권장하는 1:2 의 비율에 따른 것이다.

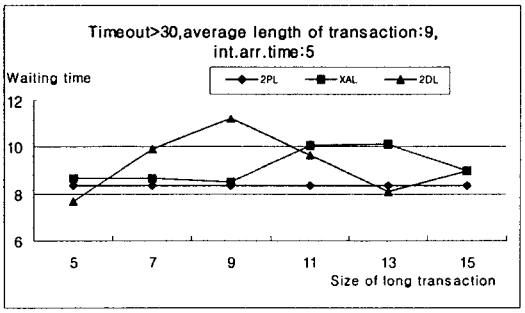
4.2 실험 결과

이 실험에서 *2DL* 이 일반적으로 2PL보다 평균 대기 시간면에서 우수함을 보였다. 또한 2DL은 처리량 성능면에서도 우수함을 보였고, XAL은 전체적으로 평균 대기 시간이 다른 제어기법 보다는 길게 나타났다.



[그림 3] 처리량

대부분의 경우에 2PL 에 대비하여 2DL 의 처리량은 101 ~ 113 %의 증가를 나타내었다. 2PL 에 대비하여 2DL 의 평균대기 시간은 트랜잭션 크기가 5 와 13 일 때 줄어들었으나, 나머지 경우에서는 오히려 늘어났음을 볼 수 있다. 이는 2DL 은 2PL 의 기법과 함께 장기 트랜잭션의 객체들에 대한 기부 연산 때문에 발생한 것으로 판단된다.



[그림 4] 평균 대기 시간

5 결론

2DL 은 장기 트랜잭션이 오버헤드를 야기할 때 다른 트랜잭션 제어기법에 비교할 때 성능이 우수하였다. 따라서 2DL 은 장기 트랜잭션들이 단기 트랜잭션들과 공존하는 환경에서 동시성 제어의 정도를 향상시키기 위해서 추천할 만한 제어기법이다. 본 논문에서는 두 개 사이트의 분산 데이터베이스 시스템에서 트랜잭션 제어기법들을 적용했지만, 동시성 제어를 위해 같은 방법으로 확장하여 적용 가능할 것이다. 향후 연구과제로는 모의실험의 방법과 인수의 값을 재조정하고 이를 기반으로 트랜잭션 제어기법에 대한 계속적인 연구가 필요할 것이다.

REFERENCES

[1] K. Salem, H. Garcia-Molina and J. Shands, "Altruistic Locking," ACM Transactions on Database Systems, Vol. 19, No. 1, pp. 117-169, March 1994.

[2] H. Bartley, C. Jensen and W. Oxley, " Scheme User's Guide and Language Reference Manual," Texas Instruments, Texas, U.S.A., 1988.

[3] R. Agrawal, M. J. Carey and M. Linvy, "Concurrency Control Performance Systems, Vol. 12, No. 4, pp. 609-654, December 1987.

[4] Zeigler, B.P., "Object-Oriented Simulation with Hierarchical, Modular Models: Intelligent Agents and Endomorphic Systems," Academic press, San Diego, CA, 1990.