분산데이터베이스 시스템에서 연쇄 조정자를 이용한 2단계 완료규약

안 인 순*, 김 경 창**
*안동과학대학 정보처리학과
**홍익대학교 전자계산학과
aisoon@andong-c.ac.kr

Non-Blocking Two Phase Commit Protocol using Cascade Coordinator in Distributed Database System

Ihn-Soon Ahn*, Kyung Chang Kim**
* Dept. of Information Processing, Andong Science College
** Dept. of Computer Science, Hongik Univ.

요 약

분산 데이터베이스 시스템에서 데이터의 일치성을 유지하기 위해 원자성 완료 규약을 수행한다. 2PC 규약은 규약 수행시 조정자의 고장으로 인하여 참여자들이 블록킹이 될 수 있다. 참여자들이 블록킹되는 것은 자신들의 자원을 풀지 못하고 조정자가 고장에서 복구될 때까지 기다려야 한다. 블록킹을 해결하기 위해 제안된 방법은 3PC(Three Phase Commit Protocol)규약이 대표적이다. 이 규약은 블록킹은 해결하지만 2PC(Two Phase Commit) 규약보다 많은 메시지 교환과 로그 기록을 하게 되고 따라서 규약을 수행하는 데 걸리는 시간이 증가된다. 본 논문에서는 2PC 규약의 변형으로 블록킹을 해결할 수 있는 기법을 제안하므로써 기존의 3PC 규약보다 수행시간을 향상시킨다. 제안하는 완료규약의 특징은 새로운 조정자를 규약 전에 미리 선출하여 조정자 고장 발생시 종료규약을 수행하게 함으로써 블록킹을 해결하였다.

1. 서론

분산 데이터베이스 시스템에서는 사이트와 통신 고장에도 불구하고, 분산 트랜잭션의 원자성 완료 문제[1]를 해결하기 위해 완료규약(Commit Protocol)을 수행한다. 분산 트랜잭션의 수행은 트랜잭션이 제출된 사이트를 조정자(Coordinator)라 가정한다. 트랜잭션의 처리를 위해서 부트랜잭션으로 나누어 다른 사이트에 부트랜잭션의 수행을 요청할 수 있는 데 이 사이트들을 참여자(Participant)라 한다. 완료규약의 종류로는 2PC(Two Phase Commit)과 이와 유사한 규약들이 제안되었다[3]. 그러나 이 규약들은 블로킹 문제가 발생하고, 블로킹을 해결하기 위해 제안된 규약이 3PC(Three Phase Commit) 규약이 있다[4]. 3PC는 블로킹은 해결하였으나 2PC규약보다 메시지 비용과 로그 비용, 이에 따른 시간 비용이 더 많이 소모되고 복잡하며 이해하기 어렵다.

본 논문에서는 2PC규약을 변형하여 블로킹되는 것을 해결하는 NB-2PC를 제안한다. 제안하는 NB-2PC는 분산 트랜잭션의 연산이 끝남을 조정자에게 알리는 메시지에 미리 새로운 조정자를 선정하고 완료규약 수행시 모든 참여자가 새로운 조정자들을 알 수 있게 하므로써 조정자의 고장 발생시 참여자들의 블로킹을 방지하고 완료규약을 성공리에 완성할 수 있게 한다.

본 논문의 구성은 다음과 같다. 2장에서는 제안하는 규약의 분산 트랜잭션 모델의 실행과정을 보여주고 3장에서는 2PC의 변형으로 조정자의 고장 발생시 종료규약을 통해 블록킹을 해결할 수 있는 NB-2PC를 제안한다. 4장에서는 조정자와 참여자의 고장 회복 과정을 보여준다. 5장에서는 NB-2PC와 3PC의 비용 비교를 하고 6장에서는 결론과 향후 연구과제를 언급한다.

2. 분산 트랜잭션

분산트랜잭션은 다수의 사이트에서 데이터를 접근하는 프로그램을 실행하는 것이다. 고장이 없을 때 부트랜잭션의 분리된 실행은 일치된 상태에서 다른 부트랜잭션에게 데이터를 전송한다고 가정한다. 동시트랜잭션들의 논리적 분리는 동시 실행(Serializable execution)로써 형식화되고 동시성제어규약[2]를 통해서 성취된다. 이 논문에서는 고장에 대한 원자성 문제에 중점을 둔다. 부트랜잭션을 처리하는 사이트들을 참여자들이라고 하고, 이들 참여자들은 연산들을 수행한다. 각각의 참여자는 자신들의 지역사이트에서 데이터를 갱신한다.

트랜잭션을 끝내기 위해서 참여자들은 서로 갱신한 데이터가 영원히 안전장치에 저장할 수 있도록 조정되어야만 한다. 우리는 이 조정의 중앙집중 버전(centralized version)만 고려하는데, 이 행위를 지휘하도록 참여자들 중의 하나를 조정자로써 행위 하도록 한다. 우리는 각각의 부트랜잭션들에게 정확한 전역 확정자(identifier)를 할당한다고 가정한다.

그림1에서는 분산트랜잭션 실행을 관리하는 스키마를 설명하고 잇다. 이것은 원자성 완료문제를 해결하고 설명하기 위한 내

용으로써 제공된다. 트랜잭션은 조정자라고 불리는 하나의 참여자에서 시작된다. 조정자는 다른 참여자들에게 부트랜잭션 연산자들의 설명과 참여자들의 리스트, 그리고 전송시간을 포함하는 T_START를 보냄으로써 트랜잭션을 분산한다. 참여자는 T_START메시지를 받자마자 부트랜잭션을 처리한다.

```
  % Coordinator executes:
1          Coordnator send [T_START: subtransaction, Tp,
participants] to all participants
  % All participants (include coordinator) execute:
2            upon (receipt of [T_START: transaction, Tp,
             participants])
3       % Perform operations requested by transaction
4            if (willing and able to make updates permanent)
          then
5            vote := [YES : participant id, Tc,Tp]
6       else vote := [NO : participant id, Tc, Tp]
        % Decide COMMIT or ABORT according to atomic
          commitment protocol
7            Tcom = Tp+Tc ; 통신하는데 걸리는 비용
8       atomic_commitment(subtransaction, participants)
```

Tp : 참여자에게 보낸 시간
Tc : 조정자에게 보낸 시간

그림1. Distributed Transaction Execution Schema

참여자가 부트랜잭션에 의해 요구된 연산들을 수행한 후에 자신이 갱신연산을 수행할 수 있는지를 조정자에게 알리는 vote 메시지를 이용한다. YES vote는 지역에서 연산수행이 성공적으로 되고 참여자는 데이터의 갱신을 안전장치에 저장하고 영원하게 만들 의지가 있다는 것을 가리킨다. 다시 말해 갱신은 미래에 고장이 발생할지라도 안전장치에 새로운 값을 쓸 것이라는 것이다. NO vote는 여러 가지 이유(예. 저장장치 고장, deadlock, 동시성제어충돌 등) 때문에 참여자가 새로운 데이터 값을 트랜잭션의 결과로써 안전장치에 저장할 수 없는 상태라는 것을 가리킨다. 이 Ack 메시지를 보낼 때 우리는 조정자에서 참여자로 메시지를 보내는 시간과 참여자에서 조정자로 메시지를 보내는 시간을 계산한다. 이 시간들 중에서 가장 적게 걸리는 시간을 차기 조정자를 선출할 때 사용하기 위함이다. 마지막으로 참여자들은 원자성 완료규약의 실행에 의해서 트랜잭션의 결과를 결정한다.

3. NB-2PC 규약

```
procedure atomic_commitment(subtransaction, participants)
      %Task 1: Executed by the coordinator
1          send [Prepare: coordinator, participant list] to all
               participants
2          wait-for (receipt of[VOTE: vote] messages from
             all participants)
3            if (all votes are YES) then
4                broadcast (COMMIT, participants)
5            else broadcast (ABORT, participants)
6          timeout ;
7                broadcast (ABORT, participants)

8       %Task 2: Executed by all participants
9          wait-for (receipt of [Prepare] from coordinator)
10              send [VOTE: vote] to coordinator
11              if(vote = NO) then
12                  decide ABORT
13              else
14                  wait-for (delivery of decision message)
15                      if(decision message is ABORT) then
16                          decide ABORT
17                      else decide COMMIT
18                  timeout
19                   decide according to termination protocol()
20         timeout
21              decide ABORT
           end
```

그림2. NB-2PC: Non-Blocking Two Phase Commit

규약의 수행과정은 그림2에서 처럼 진행한다. 조정자는 참여자들에게 Prepare 메시지를 보낸다. 이때 Prepare 메시지에는 조정자 id와 참여자 리스트를 기록한다. 참여자 리스트는 2PC와 다른 점이 있다. 이 리스트는 새로운 조정자의 선출 순서와 같고 조정자 고장으로 인한 종료규약 수행시 이용된다. 새로운 조정자 순서는 분산 트랜잭션 수행시 통신시간에 의해 결정된다. 조정자는 참여자들의 시간을 검사하여 가장 작은 값을 갖는 참여자들 순서대로 새로운 조정자로 선출한다. 왜냐하면 새로운 조정자로 선출되는 참여자는 조정자에게서 가장 먼저 메시지를 받을 가능성을 이용하여 종료규약의 효율성을 증가시킨다. 참여자들은 Prepare 메시지를 받고 로그 레코드에 Prepared 기록과 조정자 id와 참여자 리스트를 기록하고 안전장치로 옮긴 후 참여자들은 Yes 메시지를 조정자에게 보낸다. 모든 참여자에게서 Yes메시지를 받은 조정자는 로그에 Commit를 기록하고 안전장치에 옮기고 참여자들에게 Commit 메시지를 보낸다. 따라서 새로운 조정자에 의한 종료규약을 수행한다. 참여자는 Commit 메시지를 받고나서 로그에 Commit를 저장하고 안전장치로 옮긴 후, Ack메시지를 조정자에게 보낸다. 조정자는 Ack메시지를 받으면 그때 규약을 완성하고 트랜잭션을 잊는다. 이때 만약 모든 참여자가 조정자의 고장으로 인해 Commit 메시지를 받지 못하면 규약을 완성하지 못하는 불확실한 상태에 빠지기 된다. 이와 같은 상황을 블록킹이라 한다.

3.1 종료 규약의 수행

제안하는 완료규약은 고장이 발생하지 않는 경우는 2PC와 같다. 그러나 제안하는 완료규약은 2PC에서 발생할 수 있는 조정자의 고장으로 인해 불확실한 상태에 빠져 조정자의 고장 회복을 기다리는 동안 모든 참여자가 블록킹되는 것을 방지한다. 다시말해서 NB-2Pc에서는 규약을 시작하기 전에 미리 새로운 조정자을 선출하므로써 2PC에서의 블록킹을 방지할 수 있다. 따라서 조정자가 모든 참여자에게 Prepare메시지를 보낼 때 새로운 조정자 리스트를 보내어 3PC에서 처럼 새로운 조정자를 선출할 필요 없이 종료규약을 수행한다. 이것은 메시지 비용면에서 2PC

또는 3PC와 마찬가지의 비용이 든다. 왜냐하면 다른 완료규약에서도 Prepare메시지를 보낼 때 참여자 리스트를 보내기 때문이다. 조정자 고장으로 인한 새로운 조정자의 종료규약은 다음과 같다. 조정자가 Prepare 메시지를 보낼 때 메시지에 함께 보낸 참여자 리스트의 순서에 따라서 종료규약을 수행하게된다. 새로운 조정자는 자신의 규약 진행상황을 로그 기록에서 알아낸다. 로그에 Abort가 있으면, 참여자들에게 Abort를 보내고 Commit가 있으면 Commit메시지를 보낸다. 그러나 Commit이나 Abort가 없으면 다른 참여자들에게 Vote_Request메시지를 보낸다. 참여자들은 이 Vote_Request 메시지를 받으면 자신들의 규약상황을 로그 기록을 통해 알아내고 참여자들의 상태를 새로운 조정자에게 보내게 된다. 참여자들의 상태가 Abort 또는 Commit이면 규약을 완성한다. 그리고 모든 참여자가 Prepared 상태라면 새로운 조정자는 Abort를 결정하고 참여자들에게 Abort메시지를 보낸다. 기존 조정자는 고장에서 회복되면 새로운 조정자의 상태에 따라서 Commit이나 Abort를 결정하게 된다. 따라서 기존 조정자는 자신의 로그 기록이 Commit라고 하더라도 참여자들로부터 Ack메시지가 없으면 규약이 완전히 끝난 것이 아니다. 고장으로 인한 불일치성이 발생할 수 있기 때문이다. 기존 조정자는 로그에 Abort기록이 있으면 독단적으로 Abort를 할 수 있다. 그러나 로그에 Commit 기록이 있고 Ack를 받지 못한 상태라면 새로운 조정자에게 규약을 결정할 수 있는 질의를 던진다. 새로운 조정자가 Commit를 결정했다면 Commit, Abort를 결정했다면 Abort를 결정한다. 또한 새로운 조정자가 종료 규약 수행 시 Commit메시지를 보낼 때 고장이 발생하면 새로운 조정자 리스트에 있는 다음 참여자가 계속 규약을 진행한다. 따라서 이규약은 전체 고장이 발생하지 않는 한 조정자 선출이 연쇄적으로 이루어지기 때문에 연쇄 조정자 한다.

4. 고장 회복

완료 규약을 수행하는 동안 조정자 또는 참여자에게 고장이 발생할 수 있다. 고장에서의 회복은 자신들이 가지고 있는 로그 기록으로 결정에 도달할 수 있다. Commit이나 Abort기록이 없는 경우는 질의를 통해 알 수 있다. 조정자의 고장이 발생하여 회복하는 경우는 다음과 같다. 조정자의 로그 기록에 Abort가 있는 경우는 독단적으로 Abort결정할 수 있다. 조정자 로그 기록에 아무것도 없는 경우는 새로운 조정자에게 결정을 질의한다. 새로운 조정자의 결정에 따라서 Abort 또는 Commit를 결정한다. 또한 로그에 Commit이 있고 참여자들의 Ack가 없는 경우에도 새로운 조정자에게 질의한다. 왜냐하면 종료규약 수행시 참여자들의 상태에 따라서 결정이 달라질 수 있기 때문이다.

참여자의 고장 회복은 다음과 같다. 참여자의 로그 기록에 Abort나 Commit가 있으면 Abort 또는 Commit를 결정할 수 있다. 그러나 로그에 Prepared만 있으면 조정자에게 질의한다. 조정자가 Abort를 보내면 Abort 결정하고 Commit를 보내면 Commit한다.

5. NB-2PC와 3PC의 비교

NB-2PC 규약이 정상적으로 수행될 때 3PC에서는 3단계로 규약을 수행하지만 제안하는 규약은 2단계로 이루어진다. 3PC에서는 조정자 선출 규약을 수행하기 위해 복잡한 알고리즘을 수행하지만 제안하는 규약은 조정자를 선출할 필요가 없다. 왜냐하면 미리 조정자를 선출했기 때문이다. 따라서 3PC와 비교하여 NB-2PC는 메시지 교환 비용이 크지 않다. 따라서 메시지 비용은 두 규약은 많은 차이가 있다. 종료규약을 수행하는데 최악인 경우에는 두 규약이 똑같으나 최악이 아닌 경우에는 NB-2PC규약이 비용면에서 좋은 결과를 얻었다. 자세한 내용은 표2에서 설명한다.

비용		NB-2PC	3PC
단계		2단계	3단계
메시지 수	정상규약	4n	5n
	종료규약	간단	복잡
로그 레코드		조정자: 1, 참여자 : 2	조정자 : 2 참여자 :3
조정자 선출 규약의 메시지		트랜잭션 수행 후 Ack메시지를 통해 선출	복잡

표2. 3PC와 NB-2PC의 비용 비교

6. 결론 및 향후 연구 방향

본 논문에서는 완료규약 수행 시 발생할 수 있는 참여자들의 블록킹 문제를 해결했다. 기존의 3PC에서는 메시지와 로그 기록에 대한 비용이 크다. 블록킹을 해결하기 위해 제안된 논문들은 대부분 메시지 비용이 많이든다. NB-2PC규약은 완료규약 수행 시 전송되는 메시지 비용과 조정자 선출규약, 그리고 종료규약의 수행시 발생하는 비용을 3PC와 비교하여 많이 줄일 수 있었다.

향후 연구과제로는 NB-2PC와 2PC,그리고 3PC를 메시지와 로그 레코드 비용, 그리고 수행시간 비교하는 모의실험을 해야하겠다. 또한 통신 분할이 발생했을 경우의 종료규약에 대하여 좀 더 연구가 이루어져야 한다.

참 고 문 헌

[1] G. Neiger and S. Toueg. Automatically Increasing the Fault-Tolerance of Distributed Algorithms. *Journal of Algorithms*, vol. 11, no. 3, September 1990, 374-419.

[2] P.A. Bernstein and N. Goodman. Concurrency Control in Distributed Database Systems. *ACM Computing Surveys*, vol. 13, no. 2, June 1981, 185-222.

[3] C. Dwork and D. Skeen. The Inherent Cost of Non-Blocking Commitment. In *Proc. of the 2nd ACM Symp. on Principles of Distributed Systems*, Montreal, Canada, August 1983, 1-11.

[4] D. Skeen. Determining the Last Process to Fail. *ACM Trans. on Computer Systems*, vol. 3, no. 1, February 1985, 15-30.

[5] T. Chandra and S. Toueg. Time and Message Efficient Reliable Broadcast. In *Proc. of the 4th International Workshop on Distributed Algorithms*, September 1990

[6] T. Chandra and S. Toueg. "Unreliable failure detectors for asynchronous systems", Proceedings of 10th ACM Symposium on Principles of Distributed Computing, 1991