봉쇄문제를 축소한 비봉쇄 이단계 완료규약

안 인 순† · 김 경 창††

요 약

원자성 완료규약은 분산트랜잭션을 규약에 참여하는 사이트에서 일관되게 종료할 수 있게 한다. 사이트 나 통신의 고장에도 불구하고 참여자들이 트랜잭션을 정확하게 종료할 수 있다면 이것은 봉쇄문제를 제거한 규약이라 한다. 2PC 규약은 봉쇄 규약으로 유명하고, 반면에 3PC 규약은 봉쇄문제를 해결한 규약으로 알려졌다. 본 논문에서 제안된 봉쇄 해결 규약보다 봉쇄문제를 축소한 NB-2PC 규약을 제안한다. NB-2PC 규약은 2PC 규약과 3PC 규약을 변형한 규약이다. NB-2PC 규약은 선출단계에서 참여자들이 조정자에게서 Prepare 메시지를 받으면, 조정자와 여러 참여자들에게 응답 메시지를 보낸다. 참여자들은 결정 메시지를 기다리다 조정자의 고장으로 인해 봉쇄문제가 발생하면 Prepare 메시지를 보낸 참여자들 중에서 새로운 조정자를 선출한다. 새로운 조정자로 선출되는 참여자는 종료규약을 수행하여 봉쇄문제를 줄인다. 본 논문에서는 NB-2PC 규약의 기본적인 구조와 종료규약, 새로운 조정자 선출 방법을 제안한다. 또한 실험을 통하여 NB-2PC 규약이 3PC 규약보다 완료규약 수행시간이 우수하다는 것을 보인다.

Non-Blocking Two Phase Commit Protocol Reducing the Blocking Problem

Ihn-Soon Ahn† · Kyung-Chang Kim††

ABSTRACT

An atomic commitment protocol ensures that distributed transactions terminate consistently at participating sites. An atomic commitment protocol is said to be non-blocking if it permits transaction termination to proceed at correct participants despite of failure in the coordinator site and communication. It is well known that the famous two phase commit(2PC) is a blocking protocol, whereas the three phase commit(3PC) protocol is a non-blocking protocol. In this paper, we propose a non-blocking two phase commit(NB-2PC) protocol reducing the blocking problem than proposed non-blocking protocols. The NB-2PC protocol can be obtained through modifications of the 2PC protocol and 3PC protocol. After receiving Prepare message from coordinator in the NB-2PC protocol, participants respond to the coordinator and several participants in voting phase. While participants wait for decision message from the coordinator, the blocking occurs due to the failure of the coordinator site, participants elect new coordinator among several participants received response message. Despite of the coordinator site failures, participants consult new coordinator and follow termination protocols and achieve non-blocking property. We propose a basic structure of NB-2PC protocol and termination protocol and new coordinator election protocol. The NB-2PC protocol has non-bolcking property and reduces processing time of commit protocol than the 3PC protocol. Also, through simulation experiments, we propose the NB-2PC protocol exhibits better performance of processing time of commit protocol than 3PC protocol.

키워드 : 분산트랜잭션 처리(Distributed Transaction Processing), 원자성 완료규약(Atomic Commit Protocol), 봉쇄해결 규약(Non-blocking Protocol), NB-2PC 규약(NB-2PC Protocol), 종료규약(Termination Protocol).

1. 서 론

분산 데이터베이스 시스템에서는 통신 네트워크로 연결된 다수의 사이트에 데이터 아이템이 저장되어 있다. 분산 트랜잭션은 다른 사이트에 저장된 데이터 아이템을 접근하기 위하여 부트랜잭션으로 나눠서 실행된다. 이때 사이트와 통신은 독립적으로 고장이 발생할 수 있기 때문에 분산 트랜잭션의 원자성 유지는 중앙 집중 데이터베이스 시스템에서 보다 더욱 복잡하다. 즉 트랜잭션 수행에 참여하는 모든 사이트는 트랜잭션을 완료할 것인가, 철회할 것인가를 결정해야 한다. 이와 같이 동의와 결정을 확신하기 위하여 분산 데이터베이스 시스템에서는 원자성 완료규약(Atomicity Commit Protocol)을 수행한다[2]. 원자성 완료규약을 수행하는 분산 데이터베이스 시스템에서는 완료규약의 성능과 관련하여 다음과 같은 관점을 고려해야 한다.

첫 번째, 정상적인 수행 동안의 효율성이다. 고장이 발생하지 않을 경우 원자성을 제공하기 위해 발생하는 비용에

† 정 회 원 : 안동과학대학 정보처리학과 교수
†† 정 회 원 : 홍익대학교 컴퓨터공학과 교수
논문접수 : 2001년 6월 1일, 심사완료 : 2001년 8월 30일

관한 내용이다. 이것은 세 가지 비용을 가지고 측정된다[3]. 일관된 결정에 도달하기 위해 분산 트랜잭션 처리에 참여하는 사이트들간의 교환되는 메시지 수와 조정자 사이트와 참여자 사이트에 고장 발생 시 회복을 위해 저장하는 정보에 대한 로그 기록 회수, 그리고 참여자들이 결정에 도달하기 위해 요구되는 메시지 수와 단계 수(number of round)를 의미한다.

두 번째, 완료규약에 참여하는 사이트들의 고장에 대한 회복력이다. 완료규약을 수행하는 도중에 고장이 발생했을 경우에 작동 중인 사이트들은 고장의 영향에서 회복할 수 있어야 한다. 조정자 사이트와 참여자 사이트에 고장이 발생하더라도 항상 일관되게 모든 사이트는 고장에서 회복되어야 한다[4].

세 번째, 규약에 참여하는 사이트들의 독립적인 회복이다. 이것은 회복속도의 언급으로 데이터베이스를 회복하기 위해 요구되는 시간과 시스템 고장 후에 받아드리는 새로운 트랜잭션에 요구되는 시간을 말한다. 사이트는 고장 발생 시 자신의 로그에 관련된 정보가 저장되어 있다면 다른 사이트와 통신할 필요가 없이 독립적으로 회복할 수 있다.

기존 원자성 완료규약에 관한 연구로는 2가지 관점에서 연구되어졌다. 봉쇄규약에서 완료규약을 수행할 때 발생하는 메시지 교환 횟수와 로그 레코드 횟수를 줄이는 연구와 봉쇄문제를 줄이기 위한 연구가 진행되고 있다. 2PC(Two Phase Commit Protocol) 규약은 첫 번째로 제안된 가장 간단한 봉쇄완료규약이다[5]. 2PC 규약은 트랜잭션의 정상적인 처리에서 다수의 메시지 교환과 DT(Distributed Transaction) 로그 기록이 발생한다. 또한 사이트와 통신의 고장이 발생하여 이를 복구하는 동안 트랜잭션의 수행을 정지해야 하는 봉쇄문제가 발생한다. 완료규약을 수행하는 동안 메시지의 교환과 로그의 횟수를 줄이기 위해 연구된 2PC 규약의 변형은 PA 규약(Presumed Abort Protocol)과 PC 규약(Presumed Commit Protocol)이 있다[6]. 그리고 봉쇄문제를 줄이기 위한 완료규약은 3PC(Three Phase Commit Protocol)규약, BC(Backup Commit Protocol)규약, IBM-PrN(IBM's Presumed Nothing protocol)규약, DNB-AC(Decentralized Non-Blocking Atomic Commitment)규약 등이 있다.

본 논문에서 제안하는 새로운 완료규약은 봉쇄문제 축소하기 위한 완료규약이다. 따라서 기존에 제안된 비봉쇄 완료규약보다 메시지 교환 횟수와 로그 기록 횟수를 줄이고, 규약 수행시간을 줄였다.

본 논문은 제2절에서 기존의 비봉쇄 완료규약을 설명하고, 제3절에서는 본 논문에서 제안하는 봉쇄문제를 축소한 2단계 완료규약인 NB-2PC 규약의 기본 구조와 새로운 조정자 선출방법, 그리고 종료규약을 설명하고 제4절에서는 완료규약을 실험할 모델을 설정하고 제5절에서는 제안하는 NB-2PC 규약과 2PC 규약, 3PC 규약의 수행시간 비교를 통한 성능평가를 한다. 마지막으로 제6절에서는 결론을 맺는다.

2. 관련 연구

분산 데이터베이스 시스템에서 분산 트랜잭션의 수행은 트랜잭션이 제출된 사이트를 조정자(Coordinator) 사이트라 하고, 분산트랜잭션의 처리를 위해서 부트랜잭션으로 나누어 다른 사이트에 보내지게 되는데 이 사이트들을 참여자(Participant) 사이트라 한다. 조정자는 참여자들의 부트랜잭션 수행 후 참여자들과 메시지 교환을 통하여 완료규약을 수행하게 된다. 조정자의 고장으로 인해 참여자들이 조정자의 회복을 기다려야 하는 것을 봉쇄 완료규약이라 하고, 조정자의 회복을 기다리지 않고 완료규약을 완성하는 것을 비봉쇄 완료규약이라 한다. 이 절에서는 기존 제안된 봉쇄문제를 해결한 비봉쇄 완료규약의 특징을 설명한다.

3PC[4] 규약은 2PC 규약에 Pre-commit 단계를 추가하여 봉쇄문제를 해결하였다. Pre-commit 단계는 참여자들이 응답 메시지를 보내고, 조정자의 고장으로 인해 봉쇄문제가 발생하는 것을 방지하기 위해 이용된다. 조정자에 고장이 발생하면 참여자들은 참여자 중에서 새로운 조정자를 선출하여 종료 규약을 수행하여 트랜잭션의 철회와 완료를 할 수 있도록 하였다. 그러나 3PC 규약은 3단계로 규약이 수행되기 때문에 2PC 규약보다 더 많은 메시지 교환과 DT 로그 기록으로 인하여 규약 수행시간이 오래 걸리는 단점을 가진다.

BC[7] 규약은 2PC 규약에 예비(back up) 단계를 포함한다. 조정자는 선출단계에서 모든 참여자에게 응답 메시지를 받으면, 응답에 대한 결정을 예비 사이트에 보낸다. 예비 사이트는 이 결정을 받으면 조정자에게 결정에 대한 Ack 메시지를 보낸다. 조정자는 이 Ack 메시지를 받은 후 참여자들에게 결정 메시지를 보내게 된다. BC 규약은 조정자의 고장이 발생하는 경우 참여자들은 예비 사이트와 종료규약을 수행하여 결정에 도달한다. 이 규약의 단점은 규약이 3단계로 진행되고, 조정자와 예비 사이트가 동시에 고장이 발생하는 경우이다. 이 경우 참여자는 조정자 또는 예비 사이트의 회복을 기다리게 되는 봉쇄문제가 발생한다.

IBM-PrN[8, 9] 규약은 현실세계 응용에서 고장이나 통신지연 때문에 트랜잭션의 결과가 봉쇄되는 경우에 어떤 참여자는 결과를 알기 위해서 회복을 기다리지 않는 경우에 이용된다. 이 참여자들은 사업의 필요성에 따라서 조정자의 결정에 독단적으로 부트랜잭션을 완료하거나 철회한다. 참여자들의 결정은 휴리스틱 결정(heuristic decision)[18]을 허락하는 데 이 휴리스틱 결정은 트랜잭션과 데이터의 일관성에 피해를 준다. 그러나 사업의 기회 획득에 따라 이득이 생길 수 있는 장점을 가진다.

DNB-AC[10]은 2PC 규약의 기본적인 구조를 갖는다. 참여자들은 모든 메시지를 조정자와 다른 참여자들에게 동보(broadcast)하므로 써 봉쇄문제를 제거했다. 이 규약은 동기(synchronous) 시스템에서 활용되고 교환되는 메시지를 동

보 통신하는 경우에 규약의 수행시간은 향상시킬 수 있으나 점 대 점 통신의 경우에는 메시지 교환 횟수가 증가하여 규약의 수행시간을 늦어지게 하는 단점이 있다.

3. NB-2PC 규약

이 절에서는 본 논문에서 제안하는 NB-2PC 규약을 설명한다. NB-2PC 규약은 기존에 연구된 봉쇄해결 규약들의 단점을 보완하여 제안하는 완료규약이다. 제안하는 NB-2PC와 기존 연구된 규약과의 차이점은 다음과 같다.

BC 규약은 조정자와 예비 사이트가 동시에 고장이 발생했을 때 봉쇄문제가 발생하지만 NB-2PC 규약에서는 참여자들 중 새로운 조정자 역할을 할 참여자들을 여러 개를 둠으로써 BC 규약에 비해 봉쇄문제 발생 가능성을 줄였다. 그리고 BC 규약은 3단계로 규약이 진행되지만, NB-2PC 규약은 2단계로 규약이 진행된다. DNB-AC은 모든 참여자가 응답 메시지와 결정에 대한 Ack 메시지를 조정자와 다른 참여자에게 보냄으로써 메시지 교환 비용이 크게 증가한다. 2단계로 진행되어 동보통신인 경우는 효율적이나, 메시지 교환 비용 때문에 점 대 점 통신에는 비효율적이다. NB-2PC 규약에서는 동보통신과 점 대 점 통신을 고려하여 새로운 조정자 역할을 할 참여자에게만 모든 참여자가 응답 메시지를 보내게 하여 DNB-AC 보다 메시지 교환을 줄이고, 봉쇄문제를 축소하였다.

본 논문에서 제안하는 NB-2PC 규약의 주된 비교 대상은 3PC 규약이다. 기존 제안된 규약이 2PC 규약을 변형해서 봉쇄문제를 해결하려 하였지만, NB-2PC 규약은 2PC 규약과 3PC 규약의 장점을 가지고 설계된 완료규약이기 때문이다. NB-2PC 규약은 2PC 규약과 3PC 규약의 변형으로 3PC 규약보다 수행시간을 줄이고, 봉쇄문제를 제거하기 위해 제안하는 2단계 완료규약이다. 따라서 이 규약은 2PC 규약보다는 수행시간이 느리지만, 3PC 규약보다 수행시간을 줄이고 봉쇄 해결에 초점을 맞추어 연구하였다.

3.1 규약 테이블

조정자는 주기억장치에 규약 테이블을 유지한다[16, 17]. 규약 테이블에는 <표 1>과 같이 NB-2PC 규약을 진행하는 조정자와 참여자들의 규약 진행 상태를 저장한다. 참여자들이 규약 수행을 수행하는 도중에 발생하는 사이트의 고장 또는 교환되는 메시지를 잃어버렸을 경우 조정자에게 질의하면 조정자는 규약테이블에 있는 참여자들의 규약 진행상태를 확인하여 참여자들에게 알려준다. 또한 조정자도 사이트 고장이 발생할 수 있기 때문에 규약 테이블의 회복을 위해서 참여자들과 관련된 규약 진행상태를 DT 로그에 기록하고 안전장치에 옮겨야 한다.

규약 테이블에는 <표 1>과 같이 규약을 아직 완성하지 않은 조정자와 참여자들이 완료되거나 철회된 것뿐만 아니라 아직 진행하고 상태를 기록된다. 조정자는 분산 트랜잭션이 시작되면 규약 테이블 안에 분산 트랜잭션 식별자를 등록하고, 분산 트랜잭션의 엔트리에는 조정자와 참여자들의 집합과 조정자와 참여자들의 규약 진행 상태를 포함한다. 예를 들어 참여자의 Prepare 메시지에 대한 조정자의 응답여부(Commit-vote 또는 Abort-vote), 조정자의 결정 메시지에 대한 Ack 등을 기록한다. Ack 메시지는 조정자가 규약 테이블을 관리하는 데 도움을 주는데 모든 참여자들이 Ack 메시지를 보내면 조정자는 규약 테이블에서 분산 트랜잭션의 엔트리에서 참여자를 삭제한다.

〈표 1〉 규약 테이블

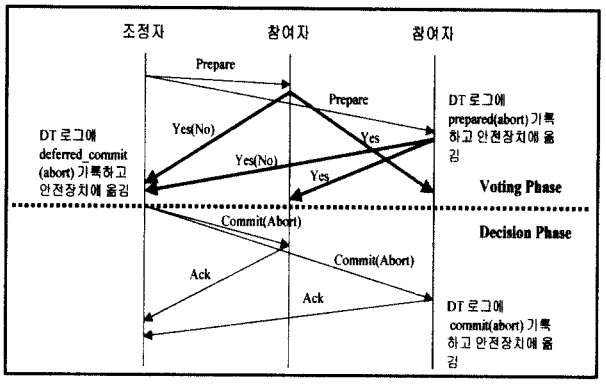
Trans. ID	Stable	State	Participant_id	Vote	Ack
	Yes No	Preparing Deferred_Commit Aborted Committed		None Abort - vote Read - only vote Commit - vote	Yes No

<표 1>에는 분산 트랜잭션의 식별자(identifier)를 Trans. ID로 표현하고, Stable의 의미는 분산 트랜잭션의 규약 진행 상태를 DT 로그에 기록했는지 여부를 나타낸다. 분산 트랜잭션의 State는 조정자가 Prepare 메시지를 보냈을 경우는 Preparing으로 기록한다. 분산 트랜잭션은 여러 개의 참여자를 가질 수 있다. 각각의 참여자는 Pid 또는 Participant id로 나타낸다. Prepare 메시지에 대한 응답은 Vote로 나타내고, 결정 메시지에 대한 응답 여부는 Ack로 표현한다.

3.2 NB-2PC의 기본구조

NB-2PC 규약의 기본 구조는 2PC 규약과 비슷하다. 2PC 규약은 봉쇄 규약이지만 NB-2PC 규약은 2PC 규약에서 발생하는 봉쇄문제를 제거하기 위해 제안하는 완료규약이다. NB-2PC 규약이 2PC 규약과 다른 특별한 특징은 조정자와 NBSet(p) 멤버들에게 모든 참여자들이 봉쇄문제를 제거하기 위하여 응답 메시지를 보낸다. NBSet(p) 멤버는 참여자 리스트 중에서 새로운 조정자 역할을 할 참여자의 집합이다. NB-2PC 규약은 동보통신인 경우에는 2PC 규약과 메시지 교환하는 데 걸리는 시간이 비슷하지만, 점 대 점인 경우에는 메시지 교환 횟수가 많아지는 단점을 가진다. 그러나 조정자의 고장이 발생하면 NBSet(p) 멤버들은 모든 참여자의 응답을 가지고 있기 때문에 새로운 조정자로 선출되어 봉쇄문제를 제거 할 수 있고, 3PC 규약보다 종료규약의 수행을 빠르게 할 수 있다.

NB-2PC 규약이 정상적으로 진행되는 경우 수행과정은 (그림 1)과 같으며 단계별로 설명한다.



(그림 1) NB-2PC 규약

[단계 1] : 조정자는 규약 테이블에 Preparing를 기록하고, Prepare 메시지를 모든 참여자들에게 보낸다. 이 메시지에는 조정자의 식별자와 참여자 리스트를 같이 보낸다. 참여자리스트에는 각 참여자에 대한 NBSet(p) 멤버의 수가 설정되어있다. NBSet(p) 멤버는 조정자의 고장이 발생하면 새로운 조정자 역할을 하는 참여자들이다.

[단계 2] : 참여자들은 Prepare 메시지를 받으면, DT 로그에 조정자 식별자와 참여자리스트, 그리고 prepared를 기록하고 안전장치로 옮기고, 조정자와 NBSet(p) 멤버에게 Yes 메시지를 보낸다. 또는 참여자의 응답이 No라면, DT 로그에 abort를 기록하고 안전장치에 옮기고, 조정자에게 No 메시지를 보내고 규약을 멈춘다.

[단계 3] : 조정자와 NBSet(p)는 참여자들에게서 Yes 또는 No 메시지를 받고, 조정자는 참여자들의 응답(Commit-vote 또는 Abort-vote)을 규약 테이블에 기록한다. 참여자들에게서 응답 메시지를 받은 NBSet(p) 멤버는 로그에 응답들을 유지한다. 조정자는 모든 참여자들에게서 Yes 메시지를 받으면, DT 로그에 deferred_commit를 기록하고 안전장치에 옮긴다. 그리고 규약 테이블에 Deferred_Commit를 기록하고, Commit 메시지를 모든 참여자에게 보낸다. DT 로그에 deferred_commit 기록을 하는 이유는 조정자의 고장이 발생하면, 선출된 새로운 조정자가 종료규약을 수행하게 된다. 이때 새로운 조정자와 참여자가 동시에 고장이 발생하면, 기존 조정자와 새로운 조정자의 결정이 서로 다른 결정을 초래할 수 있다. 그래서 이와 같은 서로 다른 결정의 발생을 방지하기 위해서 조정자의 결정을 참여자들이 결정을 한 후로 연기하는 것이다. 따라서 조정자는 완료 결정을 참여자들의 Ack 메시지를 받은 후로 연기한다.

또한 Deffered_commit 메시지의 의미는 3PC 규약에서의 Pre-commit 메시지와 다르다. Pre-commit 메시지는 참여자들이 완료 준비가 되었는지를 다시 한번 질의한다는 의미이다. 이 단계에서 조정자의 고장이 발생하면, 참여자들은 새로운 조정자와 종료규약을 수행하는 데 새로운 조정자와 참여자들이 Pre-commit를 받지 않은 경우는 abort 결정을 한다. 그러나 NP-2PC 규약에서는 NBSet(p)가 이미 참여자들의 Yes 결정을 가지고 있기 때문에 조정자가 고장이 발생하더라도 참여자들이 완료 결정을 할 준비가 되어있다고 판단하고 참여자들은 NBset(p) 중에서 새로운 조정자를 선출하여 결정에 도달하게 된다.

모든 참여자들이 Yes를 보내지 않았다면, DT 로그에 abort를 기록하고 안전장치에 옮기고, Yes 메시지를 보낸 참여자들에게 Abort 메시지를 보낸다.

[단계 4] : Yes 메시지를 보낸 참여자들은 Commit 또는 Abort 메시지를 기다린다. 결정 메시지를 받으면 결정에 따라 DT 로그에 commit 또는 abort를 기록하고 안전장치에 옮긴 후, 완료에 대한 Ack 메시지를 조정자에게 보낸다. 그러나 NB-2PC 규약은 조정자에게 참여자들이 Ack 메시지를 하나도 전달하지 않는 경우는 봉쇄문제가 발생한다. 이런 경우는 드문 경우이기 때문에 본 논문에서는 참여자들 중 하나 이상이 Ack 메시지를 조정자에게 보낸다는 가정을 둔다.

[단계 5] : 조정자는 모든 참여자들에게서 commit에 대한 Ack 메시지를 받으면, 완료를 결정하고 규약 테이블의 모든 내용을 삭제하고 규약을 완성한다.

3.3 시간종료(Time out) 행위

조정자와 참여자들의 고장으로 인해 메시지가 도착하지 않으면 메시지를 계속 기다려야 하는 경우가 발생한다. 이러한 경우를 피하기 위해 시간종료 행위를 이용한다. 시간지정종료 행위는 메시지가 시간 내에 도착하지 않으면 더 이상 조정자와 통신을 지속하지 않고 독단적으로 결정에 도달하기 위함이다.

NB-2PC 규약에서 시간지정종료 행위는 [단계 2], [단계 3], [단계 4], [단계 5]에서 발생할 수 있다.

[단계 2] : 조정자의 Prepare 메시지를 기다리다 시간종료가 발생하면, 이것은 참여자들이 응답을 하기 전이기 때문에 참여자들은 독단적으로 철회를 결정할 수 있다.

[단계 3] : 조정자가 모든 참여자에게서 Yes 또는 No 메시지를 기다리다 시간종료가 발생하면, 조정자는 아직 어떤 결정에도 도달하지 못한 상태이다. 따라서 조정자는 완료 결정을 할 수 없다. 조정자

는 철회 결정을 하고, Yes 메시지를 보낸 참여자들에게 Abort 메시지를 보낸다.

[단계 4] : 참여자가 응답 메시지를 보내고 조정자의 결정 메시지를 기다리다 시간종료가 발생하는 경우이다. 참여자들은 독단적으로 결정에 도달하지 못한다. 2PC 규약에서는 봉쇄가 발생하지만, NB-2PC 규약에서는 참여자들은 NB Set(p) 멤버 중에서 새로운 조정자를 선출하고, 새로운 조정자와 참여자들이 종료규약을 수행하여 결정에 도달하게 한다. 새로운 조정자 선출과정과 종료규약은 제3.4절, 제3.5절에서 설명한다.

[단계 5] : 조정자가 모든 참여자의 Ack 메시지를 기다리다 시간종료가 발생하는 경우이다. 모든 참여자가 Ack 메시지를 보내지 않았다면 NB-2PC 규약은 봉쇄가 발생한다. 왜냐하면 참여자들이 조정자에게 결정 메시지를 받았는지를 알 수 없기 때문이다. 따라서 이 논문에서는 하나 이상의 참여자가 결정에 대한 Ack 메시지를 보낸다고 가정했다. 조정자는 모든 참여자에게 Ack 메시지를 받으면 결정에 도달하고 규약 테이블에 있는 내용을 삭제하고 규약을 종료한다.

3.4 NB-2PC의 새로운 조정자 선출방법

완료규약 수행시 조정자의 고장이 발생하면, NB-2PC 규약에서는 새로운 조정자를 선출하여 봉쇄문제를 해결하여 결정에 도달할 수 있게 한다.

NB-2PC 규약이 2PC 규약과 차이점은 2PC 규약에서는 조정자의 고장이 발생하면, 새로운 조정자를 선출을 하지 않고 참여자들끼리 협동(Cooperative) 종료규약을 수행하는데, 협동종료규약은 참여자들끼리 서로의 규약 상태를 질의하는 것이다. 따라서 모든 참여자들이 조정자의 결정 메시지를 받지 못한 상태라면 참여자들은 봉쇄문제가 발생하게 된다. 이것은 트랜잭션의 완료 또는 철회를 할 수 없는 상태를 말한다. 그러나 NB-2PC 규약에서는 NBSet(p) 멤버 중에서 새로운 조정자를 선출해서 종료규약을 수행하여 봉쇄문제를 제거한다.

또한 3PC 규약과의 차이점은 3PC 규약에서 새로운 조정자의 선출은 참여자 리스트에 있는 순서대로 선출하지만 NB-2PC 규약에서는 참여자 리스트에 있는 참여자 순서는 의미를 가진다. 조정자와 참여자들의 통신지연시간을 계산하여 작은 시간을 갖는 참여자 순서대로 되어있다. 이것은 조정자와 참여자들간에 빠르게 메시지를 주고받는 순서대로 새로운 조정자를 선출하기 하므로 써 종료규약을 빠르게 수행할 수 있게 하기 위해서이다. NB-2PC 규약에서 새로운 조정자로 선출하기 위한 순서를 정하기 위하여 메시지 전송의 통신 지연 계산 방법은 다음과 같다.

본 논문에서는 조정자와 참여자 사이트에서 사용하는 지역시계는 모든 사이트에서 동기화 된다고 가정한다. 따라서 조정자와 참여자들은 주고받는 메시지의 전송지연을 측정하기 위하여 지역시계를 접근한다.

[단계 1] : 조정자는 부트랜잭션을 참여자들에게 보낼 때 메시지에 조정자의 지역 시계의 값을 포함하여 보낸다. 즉, 조정자의 지역 시계의 값, C1에서 메시지 전송을 시작한다. 참여자는 P1-C1에 전송된 메시지의 지연시간을 계산한다. P1은 참여자의 현재시간이다. 따라서 조정자가 참여자에게 메시지를 보내는 통신지연시간은 DT1 = P1 - C1이다.

[단계 2] : 마찬가지 방법으로 참여자들은 부트랜잭션의 수행을 끝내고 Ack 메시지를 보낼 때 DT1과 참여자의 지역 시계 값을 포함하여 보낸다. 즉, 참여자들은 지역시계의 값, P2에 메시지 전송을 시작한다. 조정자가 전송된 메시지를 받는 지연시간은 C2 - P2이다. C2는 조정자의 현재시간이다. 참여자가 조정자에게 메시지를 보내는 통신지연시간은 DT2 = C2 - P2이다.

[단계 3] : 조정자는 DT1과 DT2를 합해서 조정자와 참여자들 사이의 메시지의 통신지연시간을 계산하게 된다. 조정자는 작은 통신지연시간을 가지는 참여자 순서대로 집합 NewC에 저장하고 NB Set(p)멤버의 수를 설정하여 참여자들에게 Prepare 메시지를 보낼 때 함께 보낸다. 집합 NB Set(p) 멤버는 조정자의 고장으로 인하여 참여자들이 새로운 조정자를 선출할 때, 새로운 조정자의 순서로 활용된다.

(새로운 조정자 선출의 진행 예)

조정자 c로부터 메시지를 기다리다 시간종료에 의해 참여자 p가 조정자 c에 고장이 발생했다는 것을 알았을 경우 참여자 p는 참여자 리스트에서 조정자 c를 제거하고, 집합 NBSet(p) 멤버의 순서대로 새로운 조정자 역할을 할 참여자를 선택한다. 참여자 p가 NBSet(p) 멤버이라면, 참여자 p는 새로운 조정자로써의 역할을 담당하게 된다. 아닌 경우, 참여자 p는 NBSet(p) 멤버 중 첫 번째에 위치한 참여자 q에게 U_Elected 메시지를 보낸다. 이 메시지는 참여자 q가 새로운 조정자로 선출되었다는 것을 의미한다. 참여자 q가 U_Elected 메시지를 받으면, 참여자 q는 기존 조정자에 고장이 발생했다는 것을 알게되고, 자신이 새로운 조정자 역할을 해야 한다는 것을 알게된다. 따라서 참여자 q는 종료규약의 새로운 조정자 역할을 하게되고, 참여자 리스트에서 조정자와 자신보다 앞에 위치한 참여자들을 삭제한다. 참여자들을 삭제하는 이유는 자신이 새로운 조정자로 선출되었

다는 것을 의미한다.

참여자 q가 새로운 조정자로 선택되고, 어떤 참여자 x는 기존 조정자 c의 고장을 아직 발견하지 못하는 경우가 발생할 수 있다. 이때 참여자 x는 새로운 조정자 q로부터 결정 메시지 또는 State-Req 메시지를 받으면, 기존 조정자 c에 고장이 발생했다는 것을 알게 된다. State-Req 메시지는 새로운 조정자가 참여자들이 가지고 있는 DT 로그 기록에 대해 질의할 때 사용된다. 참여자 x는 참여자 리스트에서 기존 조정자 c와 새로운 조정자 q보다 앞에 위치한 참여자들을 제거하고 새로운 조정자 q와 종료규약을 수행한다. 이때 참여자 p는 새로운 조정자 q에게서 결정 메시지나 State-Req 메시지를 받지 못하면, NBSet(p) 멤버 중 다음 참여자 r에게 U_Elected 메시지를 보낸다. 이 경우는 새로운 조정자 q에 고장이 발생한 경우이다.

3.5 새로운 조정자에 의한 종료규약

종료 규약은 NBSet(p) 멤버 중에서 새로운 조정자가 선출되는 경우에 수행된다. 새로운 조정자가 선출되면 여러 가지 경우에 따라서 종료규약을 수행하게 된다.

[경우 1] : 새로운 참여자가 기존 조정자로부터 결정 메시지를 받은 경우, 새로운 조정자는 결정 메시지를 참여자들에게 보낸다.

[경우 2] : 새로운 조정자가 모든 참여자에게 받은 메시지가 Yes이고 자신도 Yes인 경우, 자신의 DT 로그에 deferred_commit 기록을 하고 안전장치로 옮긴 후, 다른 참여자들에게 Commit 메시지를 보낸다. 그렇지 않은 경우, 새로운 조정자는 DT 로그에 abort를 기록하고 안전장치로 옮긴 후, Yes 메시지를 보낸 참여자들에게 Abort 메시지를 보낸다.

[경우 3] : 새로운 조정자가 모든 참여자에게서 응답 메시지를 받지 못한 경우, 새로운 조정자는 어떤 참여자에게서 응답 메시지를 받지 못한 경우이기 때문에 모든 참여자의 상태를 알 수가 없다. 따라서 새로운 조정자는 DT 로그에 abort를 기록하고 안전장치로 옮긴 후 참여자들에게 Abort 메시지를 보낸다.

3.6 기존 조정자의 고장 회복

기존 조정자는 고장에서 회복되면 자신의 DT 로그 기록을 참조하여 결정에 도달하거나 새로운 조정자에게 질의를 보내 결정에 도달하게된다. DT 로그 기록이 abort이면, 철회 결정이 된 상태이다. 이 경우가 아니면, 새로운 조정자에게 질의를 보낸다. 새로운 조정자의 결정에 따라 완료 또는 철회를 결정하게 된다. 새로운 조정자가 종료규약 수행 중에 있다면 기존 조정자는 참여자로써 새로운 조정자의 결정을 기다려야 한다.

3.7 NB-2PC의 정확성

원자성 완료규약의 목적은 사이트의 고장이 발생할 지라도 모든 참여자들이 동의하여 트랜잭션을 완료 또는 철회되게 하는데 있다. 모든 참여자가 Yes 메시지로 조정자에게 응답한 경우에 전역 완료를 결정한다. 이것은 데이터 객체의 갱신을 확신한다. 모든 참여자가 Yes 메시지로 응답하지 않는 경우는 트랜잭션의 철회 결정을 해야 한다.

이 논문에서 제안하는 NB-2PC 규약의 정확성은 봉쇄해결 완료규약의 문제점을 해결하는 데 NBSet(p) 멤버를 이용한다. NB-2PC 규약은 정상적으로 규약이 수행되는 경우에는 2PC 규약에서와 같이 원자성을 보장한다. 또한 NB-2PC 규약에서는 조정자 고장으로 인해 종료규약을 수행하게 될 때, 새로운 조정자로 선출되는 NBSet(p) 멤버들은 모든 참여자의 응답을 가지고 있기 때문에 규약의 정확성을 보장할 수 있다. NB-2PC 규약의 정확성은 다음과 같이 정리할 수 있다.

[정리 1] NP-2PC 규약에서 결정이 완료이면, 모든 참여자는 Yes 메시지를 조정자와 NBSet(p) 멤버에게 보냈다.

(증명) : 어떤 참여자 pi가 완료를 결정하였다면, 조정자와 다른 참여자들은 완료 결정을 했다. pi와 다른 참여자들이 Yes메시지를 조정자와 NBSet(p) 멤버에게 보냈을 때 전역완료가 가능하다. 이 경우가 아니면 모든 참여자는 철회가 결정된다. 따라서 pi가 완료에 도달하기 위해서는 모든 참여자가 조정자와 NBSet(p) 멤버에게 Yes 메시지를 보냈을 때 가능하다. 조정자의 고장으로 인하여 새로운 조정자가 종료규약을 수행할 경우도 어떤 참여자 pj가 완료를 결정하였다면, 전역완료가 결정된다. 이것은 pj가 완료에 도달하기 위해서는 조정자와 NBSet(p) 멤버에게 모든 참여자가 Yes 메시지를 보냈거나 어떤 참여자가 완료가 결정된 상태에서 가능하다.

[정리 2] NP-2PC 규약에서 두 참여자는 서로 다른 결정을 하지 않는다.

(증명) : 정리 1에서 어떤 참여자가 조정자와 NBSet(p) 멤버에게 Yes 메시지를 보내지 않았다면, 참여자는 완료를 결정할 수 없고, 철회를 결정한다. 참여자들은 조정자에게서 Prepare 메시지를 받으면 자신들의 상태에 따라서 조정자와 NBSet(p) 멤버에게 Yes 또는 No 메시지를 보낼 수 있다. 이것은 조정자의 고장이 발생하더라도 NBSet(p) 멤버가 새로운 조정자로 선출되어 종료규약을 수행하기 때문에 모든 참여자의 응답을 알 수 있다. 따라서 새로운 조정자는 참여자들의 응답에 따라서 완료 또는 철회를 결정할 수 있다. 그래서 참여자들은 항상 동

일한 결정을 하게 된다. 그러나 네트워크 분할(network partition)이 발생하는 경우에는 참여자들은 서로 다른 결정에 도달할 수 있다.

[정리 3] NP-2PC 규약에서 고장이 없는 참여자는 결과적으로 결정에 도달한다.

(증명) : 적어도 하나의 참여자는 Yes 메시지를 보내지 않았거나, 모든 참여자는 Yes 메시지를 보낸 경우이다. 참여자들 중에 응답 메시지를 보내고 고장이 발생하더라도 문제가 되지 않는다. 왜냐하면 조정자는 모든 참여자에게 Yes 메시지를 받은 경우 deferred_commit 결정을 하고, 참여자들에게 Commit 메시지를 보낸다. 참여자가 고장에서 회복되면 조정자에게 질의를 하여 완료를 결정할 수 있기 때문이다. 또한 어떤 참여자가 No 메시지를 보내고, 고장이 발생한 경우에도 자신의 DT 로그 기록이 abort로 되어 있기 때문에 고장에서 회복되면 철회 결정을 한다. 새로운 조정자에 의한 종료규약 수행에서도 참여자의 고장이 발생하더라도 이 경우는 NBSet(p) 멤버가 모든 참여자들의 응답을 가지고 있기 때문에 참여자들은 고장 회복 후에 결정에 도달할 수 있다.

[정리 4] 모든 참여자가 No 메시지로 응답하지 않았고, 다른 참여자가 어떤 응답을 했는지 알지 못하여도 완료를 결정하여야 한다.

(증명) : 참여자가 철회를 결정하는 경우는 조정자에게 No 메시지로 응답했을 경우이다. 이 경우가 아니면 참여자들은 완료를 결정한다. 종료규약 수행 중에도 새로운 조정자가 가지고 있는 참여자의 응답이 No 메시지였다면 철회를 결정한다. 이러한 경우가 아니라면 참여자는 완료를 결정할 수밖에 없다.

4. 실험 모델

〈표 2〉 시뮬레이션 모델 파라메터와 설정치

파라메터	내 용	설 정 치
DBSize	데이터베이스에 있는 페이지 수	800page
NumSites	데이터베이스가 있는 사이트 수	10
PageSize	페이지 크기	4,096byte
TransSize	트랜잭션의 평균 참조 페이지 수	20 page
TransType	트랜잭션 형태(순서적 / 병렬)	순서적
DistDegree	분산의 degree(참여자 수)	n (규약참여사이트)
UpdateProb	페이지 갱신 가능성	1.0
NumCPUs	사이트 당 프로세서 수	1
NumDataDisks	사이트 당 데이터 디스크 수	1
NumLogDisks	사이트 당 로그 디스크 수	1
PageCPU	CPU 페이지 처리 시간	5ms
PageDisk	디스크 페이지 접근 시간	20ms
MsgCPU	메시지 송신/수신 시간	5ms

본 논문에서는 완료규약의 성능을 실험하기 위하여 분산 동시성제어의 성능을 연구하기 위하여 [11]에서 이용했던 파라메터를 가지고 성능평가에 활용한다. <표 2>는 시스템과 자원 관련 파라메터를 나타낸 것이다. 파라메터의 값 설정은 기본 설정값에 따른다.

5. 성능 평가

점 대 점 통신의 경우, 4절에서 제시한 파라메터를 이용하여 NB-2PC 규약과 3PC 규약, 그리고 2PC 규약에 대한 성능평가를 하였다. 성능평가는 두 가지 시험을 하였다.

첫째, NBSet(p) 멤버의 수와 참여자의 수에 따른 전체 규약 수행시간 비용을 측정하였고, 규약 진행 사이트의 수에 따라서 3PC 규약과 비교하여 NBSet(p)의 멤버의 수가 몇 개까지 완료규약 수행시간 비용이 적게 드는가를 측정하였다.

둘째, 정상적으로 트랜잭션이 처리되어 완료되는 경우의 전체 수행시간 비용과 조정자의 고장이 발생하여 종료규약을 수행할 경우의 전체 수행시간 비용을 계산하였다.

첫 번째 경우는 완료규약을 수행할 때 사용되는 DT 로그 기록 비용과 메시지 교환 시 발생하는 통신비용을 계산하였다. 두 번째 경우는 완료규약 수행 시 발생하는 비용뿐만 아니라, 트랜잭션 처리 시 발생하는 트랜잭션 I/O 비용과 CPU 접근 비용을 실험에 첨가하였다.

이 실험에 사용되는 데이터베이스와 트랜잭션에 관련된 파라메터는 [13]에서 이용한 것을 그대로 사용했다. 또한 메시지 교환 시간은 [14]에서 이용한 것이다. 메시지 교환에 관련된 디스크 접근 시간은 [1]에 따른다.

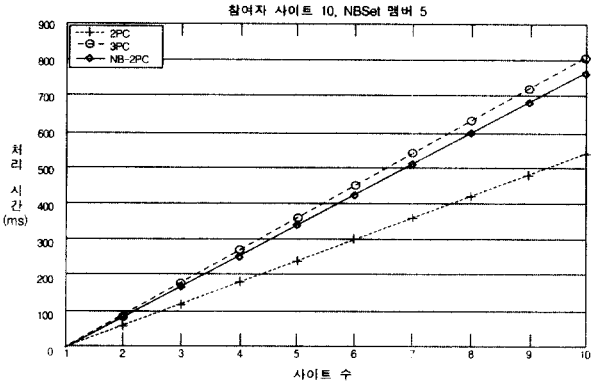
5.1 실험 1 : NBSet 멤버의 수에 따른 비용 계산

이 실험은 참여자의 수에 따라 NBSet(p) 멤버의 수가 몇 개일 때 3PC 규약에 비해 수행시간 비용이 작은가를 알아보는 실험이다. NBSet(p) 멤버의 수가 증가하면 메시지 교환 횟수가 많아져서 3PC 규약보다 수행시간이 느려질 수 있기 때문이다. 이 실험에서는 10, 25, 50, 100개의 참여자가 규약에 참여하는 경우, NBSet(p) 멤버의 수가 몇 개일 때까지 수행시간 비용이 3PC 규약에 비해 작은가를 실험했다. 완료규약 수행 시 발생하는 조정자와 (n - 1)참여자의 DT 로그 기록 횟수와 메시지 교환 횟수 비용은 <표 3>와 같다. n은 완료규약에 참여하는 사이트의 수를 말한다. <표 3>에서와 같이 NB-2PC 규약은 NBSet(p)멤버의 수가 증가하면 3PC 규약보다 메시지 교환 비용이 증가하지만, DT 로그 기록 횟수를 3PC 규약보다 줄였기 때문에 NBSet(p) 멤버의 수는 수행시간 비용을 계산하는 데 영향을 준다. 비용계산은 디스크 페이지 접근시간은 20ms하고, 메시지 전송시간은 5ms로 계산했다.

〈표 3〉 DT 로그 기록 횟수와 메시지 교환 횟수

규 약	DT 로그 기록 횟수	메시지 교환 횟수
2PC	2n - 1	4(n - 1)
3PC	3n - 1	6(n - 1)
NB - 2PC	2n - 1	4(n - 1) + (n - 1) (NBSet(p))

5.1.1 참여자 사이트 수가 10개인 경우(NBSet(p) 멤버의 수 : 5)

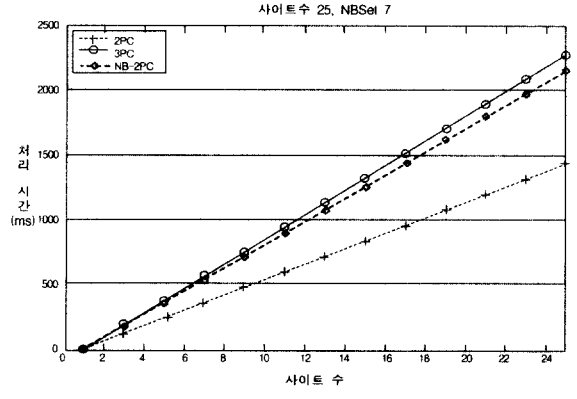


(그림 2) 사이트 수 : 10, NBSet(p) 멤버 수 : 5

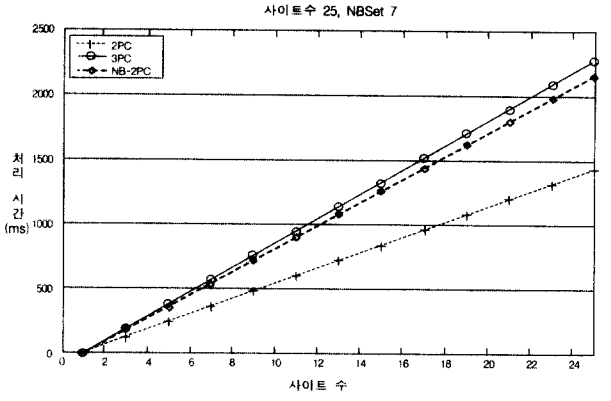
(그림 2)에서와 같이 실험결과, 참여자의 수가 10개일 때 3PC 규약 보다 규약 수행시간이 빠른 경우는 NBSet(p) 멤버의 수가 5개까지 가능했고, 6개에서는 거의 비슷한 수행시간이 나타났다. 그리고 NBSet(p) 멤버의 수가 7개인 경우에 3PC 규약보다 수행시간이 느려진다. 따라서 규약에 참여하는 사이트의 반이 NBSet(p) 멤버로서 규약에 참여하더라도 3PC 규약보다 수행시간은 빠르게 진행되었다.

5.1.2 참여자 사이트가 25개인 경우(NBSet(p) 멤버 = 5)

같은 방법으로 참여자 사이트가 25개, 50개, 100개인 경우, 수행시간을 실험한 결과 참여자 사이트 수가 10개일 때와 마찬가지로 나타났다. (그림 3), (그림 4)는 참여자 사이트가 25개인 경우, NBSet(p) 멤버의 수가 5개와 7개일 때 NB-2PC 규약의 수행시간을 나타낸 것이다.



(그림 3) 사이트 수 : 25, NBSet(p) 멤버 수 : 5인 경우



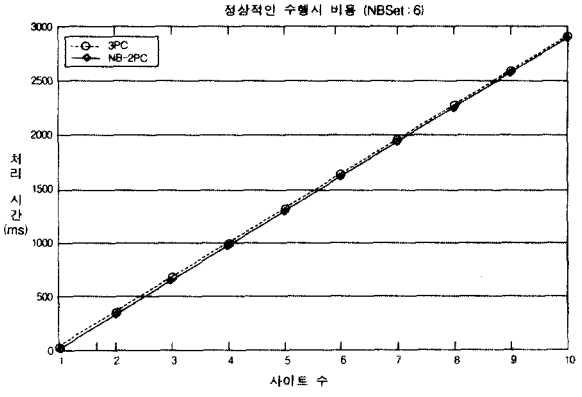
(그림 4) 사이트 수 : 25, NBSet(p) 멤버 수 : 7인 경우

결론적으로 참여자 사이트의 수와는 관계없이 항상 NB Set(p) 멤버의 수가 5개 이하이면 3PC 규약에 비해 완료규약 수행시간은 빠르게 나타났고, NBSet(p) 멤버의 수가 5개 이상이면 3PC 규약보다 수행시간이 느려졌다. 따라서 통신의 신뢰성에 따라 NBSet(p) 멤버의 수는 5개 이하로 한정하고, 최대한 NBSet(p) 멤버의 수를 줄이는 것이 좋다.

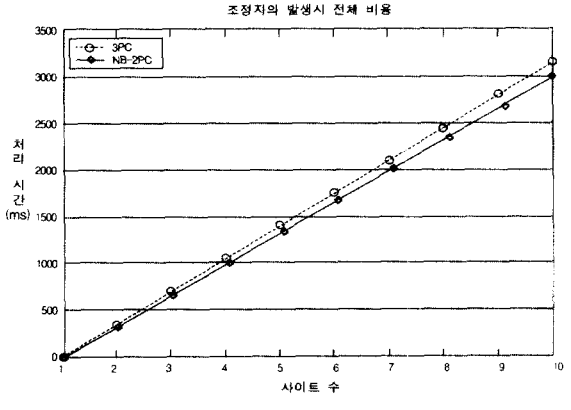
5.2 실험 2 : 정상적인 완료규약 수행시 비용

이 실험에서는 분산트랜잭션이 정상적으로 수행되었을 때 트랜잭션 처리시간과 완료규약 처리시간을 계산한다. 부트랜잭션을 참여자 사이트로 보낼 때 발생하는 통신비용과 참여자 사이트에서 부트랜잭션을 처리하고 Ack 메시지를 보내는 비용은 NB-2PC 규약과 2PC 규약, 그리고 3PC 규약이 동일한 개수를 교환하기 때문에 생략했다. 부트랜잭션을 처리하는 사이트의 수는 10개로 한정했고, NBSet(p) 멤버의 수는 7개로 하였다. 부트랜잭션 처리시 발생하는 I/O 비용은 참여자 사이트에서 발생하는 디스크 페이지 접근시간과 CPU 페이지 처리시간, 메시지 송신/수신 시간과 조정자 사이트에서 발생하는 디스크 페이지 접근시간과 CPU 페이지 처리시간을 계산했고, 완료규약을 수행할 때 발생하는 규약에 참여하는 사이트의 디스크 페이지 접근시간과 메시지 송신/수신 시간 비용을 계산했다.

정상적으로 트랜잭션이 수행되는 경우에 2PC 규약과 3PC 규약, 그리고 NB-2PC 규약의 전체 수행시간을 비교한 결과는 제5.1절에서와 마찬가지로 NBSet(p) 멤버의 수에 따라서 차이가 있었다. NB-2PC 규약을 수행할 경우는 NBSet(p) 멤버의 수를 6개 이하로 한 결과, (그림 5)에서와 같이 3PC 규약보다 빠른 수행시간을 보였다. 그리고 NB Set(p) 멤버의 수를 7개로 했을 경우는 3PC 규약보다 수행시간이 느려졌다. 이것은 트랜잭션을 처리할 때 트랜잭션이 참조하는 페이지 수와 기억장치 접근 횟수에 따라서 NB Set(p) 멤버의 수를 늘릴 수 있다는 것을 의미한다.



(그림 5) 정상적인 완료규약 수행시 비용. NBSet(p) 멤버 : 6



(그림 6) 조정자 고장 발생 시 완료규약 비용

5.3 실험 3 : 조정자 사이트의 고장발생시 완료규약 비용 비교

이 실험에서는 조정자의 고장 발생 시 참여자들이 종료규약을 수행할 때의 비용을 계산했다. 모든 참여자가 고장이 발생하지 않고, 새로운 조정자가 선출되어 종료규약을 수행하게 될 때 트랜잭션 처리의 전체비용을 계산한다. 계산방법은 종료규약 수행시 발생하는 메시지 교환 비용과 디스크 접근시간을 제5.2절의 실험에 첨가했다. NB-2PC 규약과 3PC 규약의 종료규약은 새로운 조정자가 참여자들에게 State_Req 메시지를 보내서 참여자들의 규약 진행 상태를 알아본다는 가정에서 실험했다. 이 경우에 NB-2PC 규약은 평균 4(n - 2)의 메시지 교환이 이루어지고, 3PC 규약은 6(n - 2)의 메시지 교환이 이루어진다. 이것은 NB-2PC 규약은 2단계로 종료규약을 수행하고, 3PC 규약은 3단계로 종료규약을 수행하기 때문이다. 실험 결과 (그림 6)에서와 같이 NB-2PC 규약은 종료규약 수행시 3PC 규약에 비해서 참여자 사이트가 많아질수록 트랜잭션 수행시간이 큰 폭으로 빨라졌다.

6. 결 론

본 논문에서는 조정자 사이트에 고장이 발생할 경우, 참여자들의 봉쇄문제를 축소한 NB-2PC 규약을 제안하였다. 기존 연구논문에서 BC 규약은 단순하게 조정자의 결정을 예비 사이트에 저장하여 조정자의 고장이 발생하면 참여자들은 예비 사이트와 통신하여 결정에 도달하는 종료규약을 수행하게 하였다. 따라서 조정자의 결정을 예비 사이트에 저장하기 위한 단계가 추가되고, 조정자와 예비 사이트가 동시에 고장이 발생하면 참여자들은 봉쇄되는 단점이 생긴다. DNB-AC에서는 참여자들의 응답 메시지를 조정자와 다른 참여자들에게 보내므로 써 메시지 교환 횟수가 증가하는 단점을 가졌다.

본 논문에서 제안한 NB-2PC 규약은 BC 규약과 DNB-AC의 단점을 보완한 2PC 규약과 3PC 규약을 변형한 완료규약이다. 따라서 BC 규약의 단점인 조정자와 예비 사이트가 동시에 고장났을 경우 봉쇄되는 것을 참여자들 중 새로운 조정자를 선출하여 해결하였고, ACP-UTRB 보다 메시지 교환 횟수를 줄여서 동보통신뿐만 아니라 점 대 점 통신에도 활용할 수 있게 하였다. 또한 3PC 규약과 비교하여 완료규약을 수행할 때 발생하는 메시지 교환횟수를 줄였다. 새로운 조정자 선출은 조정자와 참여자들 사이의 통신지연시간을 계산하여 통신지연시간이 적게 걸리는 순서대로 선출하기 때문에 3PC 규약에서 새로운 조정자 선출보다 종료규약을 수행하는 데 걸리는 시간을 줄일 수 있었다.

성능평가를 통하여 2PC 규약과 3PC 규약, 그리고 NB-2PC 규약의 수행시간을 실험하였다. NB-2PC 규약의 수행시간은 NBSet(p) 멤버의 수에 따라서 달라졌다. 참여자 사이트의 수에 관계없이 NBSet(p) 멤버의 수가 5개이하인 경우, 3PC 규약보다 NB-2PC 규약의 수행시간이 빠르게 나타났다. 또한 분산트랜잭션이 정상적으로 수행되는 경우와 조정자의 고장이 발생하여 종료규약을 수행할 경우, NBSet (p) 멤버의 수가 6개 이하인 경우 NB-2PC 규약이 3PC규약 보다 수행시간이 빠르게 나타났다.

참 고 문 헌

[1] J. Gray and A. Reuter, Transaction Processing : Concepts and Techniques. Morgan Kaufman, 1993.

[2] P. A. Bernstein and N. Goodman, "Concurrency Control in Distributed Database Systems," ACM Computing Surveys, Vol.13, No.2, pp.185-222, June 1981.

[3] P. Bernstein, V. Hadzilacos and N. Goodman, Concurrency Control and Recovery in Database Systems, Addison-Wesley, 1987.

[4] Skeen D, "Non-blocking Commit Protocols," Proc. of the ACM SIGMOD Int'l Conference on the Management of Data. pp.133-142, May 1981.

[5] Lampson B, Atomic Transactions. Distributed Systems : Architecture and Implementation-An Advanced Course, B.

Lampson(Ed.), Lecture Notes in Computer Science, Vol. 105, pp.246-265, Springer-Verlag, 1981.

[6] Y. J. Al-Houmaily, P. K. Chrysanthis and S. P. Levitan, "An Argument in Favor of the Presumed Commit Protocol," CS Technical Report 96-21, University of Pittsburgh, 1996.

[7] P. Krishna Reddy and Masaru Kitsuregawa, "Reducing blocking in two-phase commit protocol employing backup sites," In the Third IFCIS Conference on Cooperative Information Systems (CooIS '98), pp.406-415, August 20-22, 1998.

[8] Samaras, G., S. D. Nikolopoulos, "Algorithmic Techniques Incorporating Heuristic Decision in Commit Protocols," Proc. of the 25th Euromicro Conference, Sept. 1995.

[9] Systems Network Architecture, SYNC Point Services Architecture Reference, Document Number SC31-8134, IBM, Sept. 1994.

[10] O. Babaoglu and S. Toueg. "Non-Blocking Atomic Commitment," In Sape Mullender, editor, Distributed Systems, ACM Press, 1993.

[11] Ramesh Gupta, Jayant R. Haritsa, Krithi Ramamritham, "Revisiting Commit Processing in Distributed Database Systems," SIGMOD Conference. pp.486-497, 1997.

[12] M. Carey and M. Livny, "Distributed Concurrency Control Performance : A Study of Algorithms, Distribution, and Replication," Proc. of 14th Intl. Conf. on Very Large Data Bases, August 1988.

[13] M. Carey and M. Livny, "Conflict Detection Tradeoffs for Replicated Data," ACM Transactions on Database Systems, 16(4) : pp.703-746, 1991.

[14] M. Scott and A. Cox, "An Empirical Study of Message-Passing Overhead," In ICCC 7th International Conference on Distributed Computing Systems, pp.536-543, 1987.

[15] B. Lampson and D. Lomet, "A new presumed commit optimization for two phase commit," Proc. 19th VLDB Conference, Dublin, pp.630-640, 1993.

안 인 순

e-mail : aisoon@andong-c.ac.kr

1986년 홍익대학교 전자계산학과 졸업 (학사)

1991년 홍익대학교 대학원 전자계산학과 졸업(이학석사)

1995년 홍익대학교 대학원 전자계산학과 (박사과정 수료)

1991년~1992년 한국컴퓨그래피 주식회사 기술연구소

1994년~1999년 공주영상정보대학 전자계산과 조교수

1999년~현재 안동과학대학 정보처리학과 전임강사

관심분야 : 분산 데이터베이스, OLAP 및 데이터 웨어하우징, 멀티미디어 데이터베이스

김 경 창

e-mail : kckim@cs.hongik.ac.kr

1978년 홍익대학교 전자계산학과(학사)

1980년 한국과학기술원 전산학과(석사)

1990년 University of Texas at Austin 전산학과(박사)

1991년~현재 홍익대학교 정보컴퓨터공학부 부교수

관심분야 : 객체지향 데이터베이스, 주기억 데이터베이스, OLAP 및 데이터 웨어하우징, Web 데이터베이스