이동 컴퓨팅 환경에 기반을 둔 데이터베이스 시스템에서 서버의 고장 회복 기법

조 정 란† · 황 부 현††

요 약

이동 컴퓨팅 환경은 무선 통신망 기술의 발전에 힘입어 사용자의 이동성을 지원할 수 있는 환경이다. 사용자들은 이동 트랜잭션을 수행함으로써 데이터베이스에 접근하고 원하는 결과를 얻는다. 이러한 이동 트랜잭션의 올바른 수행과 데이터베이스의 일관성을 유지하기 위해서는 트랜잭션들을 스케줄하기 위한 동시성 제어 기법과 캐쉬를 다루는 캐슁 기법, 그리고 고장에 견고한 시스템 구축을 위한 회복 기법이 필요하다. 이동 컴퓨팅 시스템은 기존의 분산 시스템을 기반으로 하고 있으나, 사용자의 이동성이나 무선 매체의 특성으로 인하여 기존의 분산시스템에서 사용하는 회복방법들을 그대로 사용할 수 없다. 따라서 본 논문에서는 고장에 견고한 이동 컴퓨팅 시스템 구축을 위한 회복 기법을 제안하고자 하는데, 특히 이동 컴퓨팅 환경에서 발생할 수 있는 고장의 유형 가운데 서버의 고장에 대한 회복 알고리즘을 개발하고 평가 분석한다.

A Recovery Mechanism for Server Failure in Database Systems based on Mobile Computing Environments

Jeong-Ran Cho† · Bu-Hyun Hwang††

ABSTRACT

A mobile computing environment is one that support user's mobility through the wireless communication technology. Users access the database and get results what they want by running mobile transactions. To run the mobile transaction correctly and to maintain the consistency in database, we need a concurrency control method to schedule transactions, a caching method to manage the cache, and a recovery method to construct a fault tolerant system. A mobile computing system is based on the existing distributed system, but we can't use recovery methods of the existing distributed system directly because of the user's mobility and the features of wireless media. So this paper presents a recovery mechanism to construct a fault tolerant mobile computing systems. Especially, we develop and analyze a recovery algorithm for server failure among types of failure which can arise in mobile computing environments.

1. 서 론

무선 통신망 기술의 발전과 컴퓨터 기술의 발전으로 이동 컴퓨팅이 가능해졌다. 이제 사용자들은 자신의 이동 컴퓨터를 사용하여 무선 네트워크를 통한 원거리 컴퓨터 시스템을 접근하거나, 이동하거나 여행 중에도 네트워크 연결을 계속 유지하면서 데이터베이스를 접근할 수 있게 되었다. 또한 무선 통신 기술의 발전으로 컴퓨터 통신을 위한 네트워크 선(line)을 설치하기 어려운 산간 고지대나 섬 지역에서도 쉽게 컴

※ 본 논문은 1997년도 한국과학재단 핵심전문연구비에 의하여 연구되었음.
† 정 회 원 : 광주여자대학교 전산학과 교수
†† 정 회 원 : 전남대학교 전산학과 교수
논문접수 : 1998년 3월 19일, 심사완료 : 1998년 10월 21일

서비를 이용하여 정보처리를 할 수 있게 되었다.

이동 컴퓨팅 환경이 사용자에게 많은 편리함을 제공하고 있지만 사용할 수 있는 이동 컴퓨터의 제한된 전력과 서로 다른 지역에서 제공되는 네트워크의 다양성 때문에, 이동 컴퓨터와 통신 네트워크간의 접속이 자주 중단되는 경우가 많고 항상 간헐적으로 통신이 이루어지는 문제점을 갖고 있다. 또한 이동 컴퓨터를 사용하는 이동 호스트는 데이터베이스를 접근하기 위해 무선 통신망을 사용하기 때문에 데이터 전송 속도가 느리고, 무선 통신망의 제한된 대역폭으로 인하여 많은 양의 데이터를 전송하기에 적합하지 않다. 그리고 사용자들이 이동하면서 트랜잭션을 제출하는 경우에는 이동 호스트의 이동에 따른 트랜잭션 관리가 요구된다. 트랜잭션이 수행되는 환경에서 트랜잭션의 관리 방법에는 동시성 제어 방법과 고장에 대한 회복 방법이 포함되어야 한다. 이동 컴퓨팅 환경에서도 사용자들은 원하는 작업을 수행하기 위해 트랜잭션을 제출하는데, 트랜잭션의 올바른 수행을 보장하기 위해서는 트랜잭션의 동시성 제어 방법과 회복 방법이 요구된다. 본 논문에서는 이동 컴퓨팅 환경에서 고장이 발생하였을 때 회복하기 위한 회복 알고리즘에 대하여 제안하고자 한다.

지금까지 제안된 대부분의 회복 기법들은 기존의 분산 환경을 기반으로 하고 있다. 하지만 무선 통신망의 좁은 대역폭, 이동 호스트의 잦은 단절, 트랜잭션의 이동성과 같은 이동 환경의 특성으로 인해 기존 분산 시스템에서의 회복 알고리즘을 이동 환경에 그대로 적용하기는 어렵다. 따라서 본 논문에서는 이동 환경에 효율적으로 적용할 수 있는 회복 방법을 제안하고자 한다. 특히 이동 컴퓨팅 환경에서 발생할 수 있는 고장의 유형 가운데 서버 고장에 대한 회복 방법을 제안한다. [15]에서 제안한 이동 컴퓨팅 환경에서의 회복 방법은 이동 호스트의 상태 정보가 다수의 서버에 중복 저장되어야 하기 때문에 중복 저장된 상태 정보를 일관성 있게 유지하기 위해서는 많은 메시지 교환과 디스크 공간을 필요로 한다. 본 논문에서는 이러한 문제점을 해결하기 위해 각 이동 호스트에 제출 리스트를 유지하며, 서버가 고장났을 때 제출 리스트의 정보를 사용하여 서버 고장으로부터 회복할 수 있다. 이동 호스트의 상태 정보는 현재 위치한 셀의 서버에만 유지되고 여러 서버에 중복 저장될 필요가 없기 때문에 많은 메시지 교환이나 디스크 공간을 요구하지 않는다.

본 논문의 구성은 다음과 같다. 2장에서는 기존 분산 시스템에서의 회복 방법이 이동 환경에 적합하지 않은 이유와 서버 고장을 다룬 연구에 대하여 기술한다. 3장에서는 본 논문에서 사용하는 이동 컴퓨팅 시스템 모델을 기술하고, 4장에서는 이동 컴퓨팅 시스템에서 발생하는 서버 고장에 대한 회복 정책을 제안한다. 5장에서는 제안한 회복 알고리즘을 분석하여 기존 알고리즘과 성능을 비교하고, 마지막으로 6장에서 결론을 기술한다.

2. 관련연구

[9, 10, 14, 16]에서 제시한 회복 방법들은 분산 환경에 기반을 두고 있다. 이 방법들은 이동 환경에 그대로 적용될 수 없는데, 그 이유는 이동 컴퓨팅 환경이 갖는 특성[2, 4, 8, 15]이 기존의 분산 환경과 다르기 때문이다. 무선 통신을 사용하는 이동 컴퓨팅 환경은 유선 환경에 비해 할당된 대역폭이 좁고 신뢰성이 낮으며 지연 시간이 길다. 또한 사용자의 이동성으로 인해 이동 컴퓨터의 위치가 동적으로 변경됨으로써 사용자 질의에 대해 일괄적인 답을 제공할 수 없다. 이동 컴퓨팅 시스템의 관리자는 사용자의 이동성을 지원할 수 있는 방법을 고안해야 한다. 그리고 네트워크 고장이나 한정된 배터리 사용 시간으로 인해 이동 컴퓨터는 서버와 자주 단절된다.

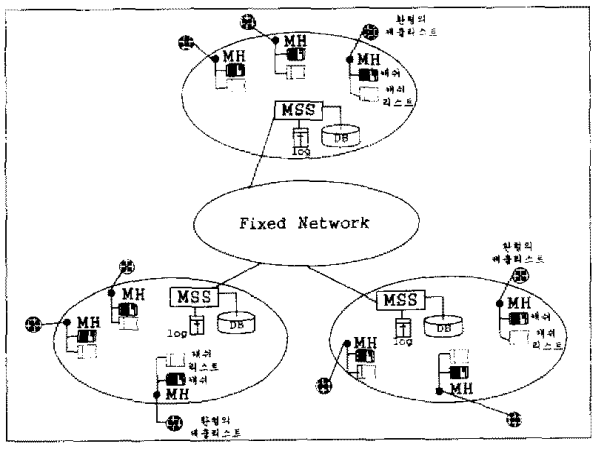
이동 컴퓨팅 환경에서도 여러 가지 유형의 고장이 발생할 수 있다. 이러한 고장으로부터 회복하기 위한 회복 기법은 데이터베이스 상태를 고장 전의 상태로 유지함으로써 데이터베이스를 접근하는 사용자에게 정확한 결과를 전달하기 위해 반드시 필요한 트랜잭션 관리 방법이다. 아직까지 이동 컴퓨팅 환경에서 발생가능한 고장에 대한 회복 알고리즘을 제안한 연구들은 그리 많지 않다. 본 논문에서는 이동 컴퓨팅 환경에서 발생할 수 있는 고장의 유형 중 서버 고장에 대한 회복 방법을 제안하고자 하는데, 지금까지 제안된 회복 방법들 중 서버 고장을 다룬 연구[15]에 대하여 살펴본다.

[15]에서는 동시에 k개의 서버가 고장났을 때 회복할 수 있는 방법을 제안하였다. 이 방법에서는, 한 이동 호스트의 정보를 현재 위치하고 있는 셀의 서버와 몇 개의 2차 서버(secondary server)에 중복 저장하였다. 이동 호스트의 2차 서버는 현재 있는 셀에 인접한 셀의 서버들로 구성된다. 이동 호스트가 위치하고 있

는 셀의 서버에서 고장이 발생하면, 이동 호스트는 인접한 셀로 이동하여 2차 서버 중 하나의 서버와 연결함으로써 연산을 계속한다. 이 방법에서는 이동 호스트의 상태 정보가 여러 서버에 중복 저장되기 때문에 많은 기억 공간을 필요로 한다. 또한 중복 저장된 상태 정보를 일관되게 유지하기 위해서는 서버들 사이에 부가적인 메시지 교환이 요구된다.

이동 컴퓨팅 환경에서 서버 고장이 발생하면 서버와 연결을 갖고 있는 이동 호스트들의 연산이 중지된다. 만약 서버가 고장 상태에 있는 기간이 길어져 회복하는데 많은 시간이 든다면 작동중인 이동 호스트의 지연 시간이 길어진다. 서버 고장 회복에 관한 기본적인 원칙은 서버 고장으로 인하여 이동 호스트의 작업이 지연되지 않도록 하는 것이다. 따라서 본 논문에서는 [15]에서 갖는 많은 메시지 교환과 디스크 공간의 필요의 문제를 해결하고, 서버 고장으로 인하여 이동 호스트의 작업이 방해받지 않는 회복 방법에 대하여 제안하고자 한다.

3. 이동 컴퓨팅 시스템 모델



(그림 1) 이동 컴퓨팅 시스템 모델
(Fig. 1) Mobile Computing System Model

본 논문에서 제안하는 회복 알고리즘을 위한 이동 컴퓨팅 환경의 시스템 모델은 그림 1과 같다. 이 모델은 [3]과 [4]에서 제안된 모델에 기본을 두고 있다. 시스템 모델은 고정 호스트(Fixed Host : FH)와 이동 호스트(Mobile Host : MH)로 구성된다. 고정 호스트는 기존의 유선 환경과 연결을 갖고 있는 호스트로, 고정 호스트 중 일부는 서버로서의 역할을 수행할 수 있다. 이동 호스트의 통신을 위해 필요한 기능을 갖고 있으며 이동 지원국(Mobile Support Station : MSS)이라 불리기도 한다. 이동 호스트는 서버와 무선 연결을 유지하면서 이동할 수 있는 호스트이다[7]. 한 개의 서버가 제어하는 지리적인 영역을 셀(cell)이라 한다. 이동 호스트가 다른 호스트와 통신하기 위해서는 서버를 경유해야 한다. 고정 호스트들 사이의 통신 채널은 이동 컴퓨팅 환경의 정적인 부분을 구성하며, 모든 이동 호스트들은 이동 컴퓨팅 환경의 이동 부분을 구성한다. 이동 호스트는 언제라도 한 셀에서 다른 셀로 이동할 수 있다.

4. 이동 컴퓨팅 시스템에서 회복 정책

본 장에서는 이동 컴퓨팅 환경에서 발생가능한 고장의 유형을 분류하고, 서버 고장에 대한 회복 알고리즘의 제안에 필요한 가정과 자료 구조, 그리고 구체적인 회복 알고리즘을 기술한다.

4.1 고장 유형

이동 컴퓨팅 환경에서 발생할 수 있는 고장은 트랜잭션 고장, 이동 호스트 고장, 서버 고장, 네트워크 고장 등으로 분류할 수 있다. 이와 같은 고장의 유형은 기존의 분산 환경에서 발생가능한 고장과 이동 컴퓨팅 환경에서 고유하게 발생할 수 있는 고장의 유형을 고려한 것이다. 트랜잭션 고장은 이동 호스트에서 제출한 트랜잭션이 철회되는 경우에 발생하는데, 이동 호스트에서는 두 종류의 트랜잭션이 발생한다. 하나는 판독 전용(read-only) 트랜잭션이고, 다른 하나는 갱신 트랜잭션이다. 판독 전용 트랜잭션은 이동 호스트에서 자체적으로 캐쉬를 사용하여 수행하며, 기록 연산을 포함하고 있는 갱신 트랜잭션은 서버에 제출하여 수행을 위임한다. 따라서 트랜잭션 고장은 판독 전용 트랜잭션의 철회와 갱신 트랜잭션의 철회를 포함한다. 이동 호스트 고장은 이동 컴퓨터의 고장으로 연산을 진행할 수 없는 경우에 발생한다. 서버 고장은 정전과 같은 이유로 서버가 작동하지 못하는 경우이며, 네트워크 고장은 통신망의 문제로 인하여 메시지를 전송할 수 없는 경우에 발생한다.

네 가지 고장의 유형에 대한 회복 방법을 살펴보면 다음과 같다. 첫째, 트랜잭션 고장은 판독 전용 트랜잭션이나 갱신 트랜잭션이 철회된 경우로 철회된 트랜잭

[illegible] 이동 호스트가 재제출함으로써 회복할 수 있다. 둘째, 이동 호스트 고장의 경우에는 서버가 이동 호스트로부터 제출받은 트랜잭션의 수행 결과를 이동 호스트가 회복될 때까지 자신의 임시 기억장소에 저장한 후 이동 호스트가 회복되었을 때 수행 결과를 전달함으로써 회복할 수 있다. 셋째, 서버 고장의 경우에는 그 셀에 있는 이동 호스트가 고장난 서버의 회복을 기다리지 않고 이웃한 셀로 이동하여 연산을 계속 수행할 수 있다. 하지만 이동 호스트가 인접한 셀로 이동하여 연산을 수행하기 위해서는 이동 호스트에서 수행중인 트랜잭션에 대한 상태가 유지되어야 하고 이 상태 정보에 따라 회복 절차를 수행해야 한다. 따라서 구체적으로 유지해야 할 상태 정보와 회복 방법을 고안할 필요가 있다. 넷째, 네트워크 고장의 경우는 이동 호스트 고장이나 서버 고장과의 구별이 불가능하다. 네트워크 고장이 발생하면 더 이상 이동 호스트와 서버와의 통신은 이루어지지 않기 때문에, 위에서 제안한 이동 호스트 고장 회복 알고리즘이나 서버 고장 회복 알고리즘을 적용함으로써 네트워크 고장으로부터 회복할 수 있다.

본 논문에서는 위에서 언급한 네 가지 고장의 유형 중 서버 고장에 대한 구체적인 회복 방법에 대하여 제안하고자 한다. 서버 고장을 제외한 다른 고장에 대한 회복 방법은 단순한 트랜잭션의 재제출이나 다른 유형의 회복 알고리즘을 적용하여 해결할 수 있기 때문에 서버 고장에 대한 자세한 알고리즘을 제안하고자 한다. 서버는 이동 호스트가 데이터를 접근할 수 있는 유일한 통로를 제공하며 이동 트랜잭션의 갱신 연산을 수행하기 때문에, 서버의 고장이 이동 호스트에 미치는 영향은 매우 크다. 또한 하나의 서버는 다수의 이동 호스트를 지원하기 때문에, 서버 고장으로부터 효율적으로 회복하는 방법은 시스템 전체의 성능에 영향을 미칠 수 있다.

4.2 가정

제안하는 회복 알고리즘을 위해 사용되는 가정들은 다음과 같다.

- 이동 호스트의 상태 정보는 현재 이동 호스트가 위치하고 있는 서버에만 저장한다. 이렇게 함으로써 상태 정보의 중복 저장에서 발생하는 많은 메시지 교환과 디스크공간 소모의 문제를 해결할 수 있다.
- 회복 알고리즘을 사용하여 회복되는 트랜잭션은 이동 호스트에서 제출되는 갱신 트랜잭션을 대상으로 한다. 서버에서 발생되는 트랜잭션들은 기존의 분산 환경에서 사용한 방법을 사용하여 처리하고, 본 논문에서는 이동 호스트에서 제출되는 트랜잭션의 회복 방법에 대하여 기술한다.
- 이동 호스트에서는 판독 연산만으로 구성되는 판독 전용 트랜잭션이 수행되고, 서버에서는 기록 연산을 포함하는 갱신 트랜잭션이 수행된다. 이동 호스트에서 발생하는 갱신 트랜잭션은 서버에 제출되어 수행된다.
- 서버 고장은 그 셀 내의 이동 호스트와 이웃한 서버에 의해 탐지될 수 있다. 서버고장은 고장난 서버의 셀에 있는 이동 호스트나 인접한 셀의 서버가 메시지를 보낸 후 일정 시간동안 응답이 없으면 고장으로 인식하거나 임의의 고장 탐지 방법에 의해 인식될 수 있다.
- 본 논문에서는 이동 컴퓨팅 환경에서 발생가능한 네 가지 고장 유형 가운데 서버 고장만을 다룬다.

4.3 자료 구조

본 논문에서 제안하는 회복 방법을 위해 유지되는 제출 리스트는 다음과 같다.

- **제출 리스트(Submission List)**

Tr_id	Tr_state
T_1	e
T_2	a
T_3	e
T_4	e

(그림 2) 제출 리스트
(Fig. 2) Submission List

제출 리스트는 그림 2와 같은 리스트 형태로 각 이동 호스트에 유지되며, 이동 호스트가 서버로 제출한 갱신 트랜잭션에 대한 정보를 유지한다. 제출 리스트의 각 엔트리는 이동 호스트가 서버로 제출한 트랜잭션의 식별자(Tr_id)와 제출한 트랜잭션의 상태(Tr_state)의 두 개의 필드로 구성된다. 일반적으로 트랜잭션의 상태는 수행중이거나 철회나 완료의 세 가지 상태 중

한 가지 상태를 갖는다. 제출 리스트에 유지되는 트랜잭션의 상태는 수행중(executive)을 의미하는 'e'와 서버로부터 철회 결정을 받았음을 의미하는 'a' (abort) 중 하나이다. 트랜잭션이 완료되면 'e' 상태에서 엔트리가 삭제되므로 완료 상태는 유지되지 않는다.

4.4 제안하는 회복 알고리즘

이동 컴퓨팅 환경에서 서버 고장이 발생하였을 때 회복 시간이 길어지면 고장난 서버의 제어 하에 있는 이동 호스트의 연산도 오랫동안 블럭(block)되므로 이런 문제를 해결하기 위해서는 새로운 알고리즘이 필요하다. 특히, 이동 호스트에서 서버로 많은 작업을 제출하여 수행하는 경우에 서버 고장은 전체적인 시스템 성능을 저하시키는 원인이 된다. 본 절에서는 제출 리스트를 이용하여 고장이 발생하지 않은 정상 상태와 고장이 발생한 상태에서 회복하는 알고리즘을 제안한다. 제안하는 방법을 RSL-MT(Recovery mechanism using Submission List for Mobile Transactions)라 하자.

4.4.1 정상 상태

서버 고장이 발생하지 않은 상태에서 제출 리스트를 사용한 트랜잭션 수행 방법을 살펴보기 위해 예제 1을 기술한다.

[예제 1] 정상 상태에서의 트랜잭션 수행

t_1	t_2	t_3	t_4	t_5	t_6	시간
↑	↑	↑	↑	↑	↑	
T_1	T_2	T_3	T_4	C_2	A_1	

● **이동 호스트의 트랜잭션 제출**

시간 t_4까지 이동 호스트가 서버에 갱신 트랜잭션 T_1, T_2, T_3, T_4를 순서대로 제출한 후의 제출 리스트는 그림 3과 같다.

Tr_id	Tr_state
T_1	e
T_2	e
T_3	e
T_4	e

(그림 3) 트랜잭션 제출
(Fig. 3) Submission of transactions

● **서버로부터 완료 결과 수신**

시간 t_5에서 서버로부터 T_2에 대한 완료 결정을 받았을 때 제출 리스트는 그림 4와 같다. 제출 리스트에서 완료된 T_2의 엔트리가 삭제되었음을 확인할 수 있다.

Tr_id	Tr_state
T_1	e
T_3	e
T_4	e

(그림 4) 완료 결과 수신
(Fig. 4) Reception of commit result

● **서버로부터 철회 결과 수신**

이동 호스트가 시간 t_6에서 서버로부터 T_1에 대한 철회 결정을 받게 되었을 때 제출 리스트의 상태는 그림 5와 같다. T_1에 대한 상태가 'e'에서 'a'로 변경되었음을 알 수 있다.

Tr_id	Tr_state
T_1	a
T_3	e
T_4	e

(그림 5) 철회 결과 수신
(Fig. 5) Reception of abort result

● **철회된 트랜잭션의 재제출**

이동 호스트는 서버로부터 철회 결과를 통보받은 트랜잭션들을 다시 제출하여 수행한다. 하지만 철회 결과를 받은 모든 트랜잭션을 재수행하지는 않는다. 트랜잭션의 철회 원인에 따라 재제출 여부가 결정되는데 재수행이 가능한 트랜잭션만 재제출되어 수행된다. 재수행이 가능한 트랜잭션의 예로는 다른 트랜잭션과의 충돌(conflict)로 인해 철회된 트랜잭션이 이 부류에 속한다. 재수행의 의미가 없는 트랜잭션의 예로는, 잔액이 부족한 구좌에서 현금을 인출하려다 철회된 트랜잭션을 들 수 있다. 따라서 이동 호스트는 철회된 트랜잭션의 의미에 따라 재수행 여부를 결정한다.

이동 호스트가 트랜잭션을 재제출하면 제출 리스트에 이 정보가 유지되는데, 그림 6과 같이 T_1이 철회되었을 때 철회된 트랜잭션의 상태 필드만을 'e'로 갱신하거나, 그림 7과 같이 철회된 트랜잭션의 엔트리를 삭제한 후 제출 리스트의 맨 마지막에 새로운 엔트리로 삽입하여 유지한다.

Tr_id	Tr_state
T_1	e
T_3	e
T_4	e

(그림 6) 철회된 트랜잭션의 재제출 I
(Fig. 6) Resubmission of aborted transaction I

Tr_id	Tr_state
T_3	e
T_4	e
T_1	e

(그림 7) 철회된 트랜잭션의 재제출 II
(Fig. 7) Resubmission of aborted transaction II

4.4.2 고장 상태

서버 고장은 고장난 서버의 셀에 있는 이동 호스트나 다른 서버에 의해 탐지가 가능하다고 가정하였다. 서버 고장이 발생했을 때 서버와 이동 호스트 각각에서 수행하는 절차는 다음과 같다.

서버 고장이 발생한 후 다시 회복하게 되면, 서버는 기존 분산 시스템에서의 회복 기법을 사용하여 회복할 수 있다. 고장 발생 후 서버가 다시 복구되면, 고장으로부터 회복되었음을 이동 호스트나 다른 서버들에게 알리고 이웃하는 서버와의 통신을 통해 자신의 고장 동안 발생한 데이터베이스 갱신을 반영하여 일관된 데이터베이스 상태를 유지한다.

서버 고장이 발생하였을 때 이동 호스트는 서버 고장이 탐지되면 이동 트랜잭션을 수행할 수 있는 서버를 찾아 이동해야 한다. 즉, 자신과의 연결을 갖고 있는 서버에 고장이 발생하였음을 감지한 이동 호스트는 자신의 위치에서 가장 가깝고 고장이 발생하지 않은 셀로 이동하여 그 셀의 서버와 연결한다. 그리고 제출 리스트에 기록되어 있는 트랜잭션들을 새로운 서버에게 다시 제출함으로써 계속적인 수행을 진행할 수 있다. 서버가 고장났을 때 제출 리스트에 기록된 트랜잭션들은 고장난 이전 서버에 의해 수행 중이었거나 (Tr_state : 'e'), 철회 결과를 받았지만 아직 재제출하지 못한 트랜잭션들(Tr_state : 'a')이다.

서버 고장은 트랜잭션의 수행 결과와 고장 시점에 따라 다음과 같이 두 가지 경우로 분류할 수 있다. 첫 번째는 이동 호스트가 서버로부터 철회 결정을 통보받은 후 재제출하기 전에 서버 고장이 발생한 경우이고, 두 번째는 서버에서 이동 호스트로 완료 결정을 전송한 후 이동 호스트에 도착하기 전에 서버 고장이 발생한 경우이다.

첫 번째 경우에 트랜잭션을 재수행하기 위해서 이동 호스트는 작동중인 이웃한 셀의 서버와 연결한 후 자신의 제출 리스트에 기록된 모든 트랜잭션을 새로운 서버에 재제출한다. 이 때 제출 리스트에 있는 트랜잭션의 상태 값을 'e'로 변경한다. 예를 들어, 이동 호스트가 그림 5의 제출 리스트 상태 즉, 철회 결과를 받은 T_1을 아직 재제출하지 않은 상태에서 서버 고장을 탐지하게 되면, 앞에서 설명한 것처럼 새로운 셀로 이동하여 제출 리스트에 기록되어 있는 트랜잭션들 T_1, T_3, T_4를 차례로 새로운 서버에게 제출하는데, 상태가 'a'인 트랜잭션 T_1은 그림 8과 같이 그 상태 값을 'a'에서 'e'로 변경한다.

Tr_id	Tr_state
T_1	e
T_3	e
T_4	e

(그림 8) 고장 후 트랜잭션 제출
(Fig. 8) Submission of transaction after failure

두 번째 경우에 서버로부터 완료 결정을 받은 트랜잭션에 대한 엔트리는 아직 이동 호스트의 제출 리스트에 유지되어 있다. 서버 고장을 감지한 이동 호스트는 이동하여 새로운 서버와 연결한 후 자신의 제출 리스트에 기록되어 있는 트랜잭션들을 새로운 서버에게

다시 재출한다. 하지만 재재출되는 트랜잭션가운데 이동 호스트의 제출 리스트에는 상태 값이 'e'로 남아 있지만 이미 이전 서버에서 수행을 마친 트랜잭션들은 새로운 서버에 의해 다시 수행됨으로써 중복 수행되어진다. 이 문제는 다음과 같이 해결할 수 있다. 이동 호스트는 서버의 고장을 탐지했을 때 곧바로 새로운 셀로 이동하지 않고, 일정시간 동안 기다린 후 이동한다. 즉, 최소한 서버가 보낸 메시지를 이동 호스트에서 받는데 걸리는 시간 동안은 기다림으로써 서버가 고장 발생 직전에 보낸 완료 메시지를 이동 호스트가 수신할 수 있다.

정상적인 상태에서 이동 호스트와 서버 각각에서 수행되는 알고리즘은 알고리즘 1과 같다.

(알고리즘 1) 정상 상태에서 서버와 MH 수행 알고리즘

Server's Algorithm

```
IF (MT is terminated)      // MT는 이동 트랜잭션
    THEN send the transaction's result to the
         corresponding MH;
```

MH's Algorithm

```
write tr_id to the submission list;
set the tr_state value to 'e';
submit MT to the server;
do until receive a result from the server
   wait;
IF (receive a result from the server)
    switch (result)
       commit : remove the entry of correspon
                ding MT from submission list;
       abort : set the tr_state value to 'a';
               resubmit the MT to the server;
```

서버 고장이 발생한 경우 회복 알고리즘은 알고리즘 2와 같다.

(알고리즘 2) 서버 고장에 대한 회복 알고리즘

MH's Algorithm

```
IF (server failure is detected)
    THEN search another server;
         move to the cell;
         resubmit all the transaction in the
         submission list;
         set the tr_state value to 'e';
         resume normal operation;
```

Server's Algorithm

```
IF (server is recovered from failure) THEN
    send the recover_M to all server and MHs;
      // recover_M은 서버가 고장으로부터 회복됐
         음을 알리는 메시지
    inquire the database state to the other
    server;
    resume operation;
```

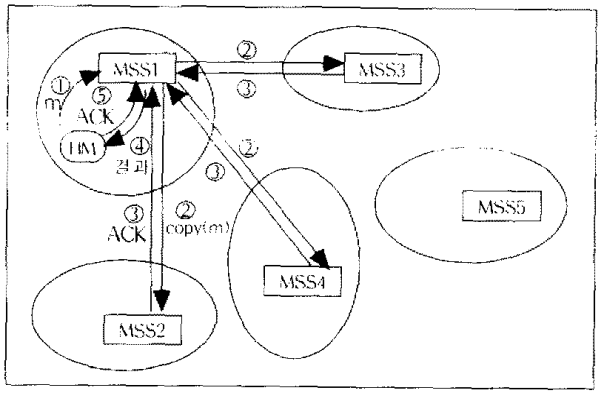
5. 제안하는 알고리즘의 분석 및 비교 평가

본 장에서는 본 논문에서 제안한 RSL-MT 알고리즘의 효율성 및 성능을 증명하기 위해 회복에 관한 연구를 제안한 [15] 방법과 비교 및 분석하였다. 두 알고리즘의 성능을 비교하기 위한 기준으로는 알고리즘을 처리하는데 요구되는 메시지 전송량과 알고리즘 수행을 위해 필요한 기억공간, 그리고 알고리즘 수행 시간을 사용하였다.

5.1 요구되는 메시지 전송량

1) [15] 알고리즘

그림 9에서 현재 이동 호스트가 위치하고 있는 서버를 의미하는 pri_mss = {MSS_1}이고, 인접한 셀의 서버 모임인 2차 서버 sec_mss = {MSS_2, MSS_3, MSS_4}라고 하자. 이동 호스트가 MSS_1에게 트랜잭션을 제출하면(①) MSS_1은 이의 사본을 sec_mss에게 보낸다(②). sec_mss가 메시지 수신 인지신호(③)를 보내면 이 때 MSS_1은 이동 호스트에게 ①에 대한 인지신호(④)를 보낸다. ②와 ③은 sec_mss내의 서버 수만큼



(그림 9) [15]에서 요구하는 메시지 전송량
(Fig. 9) Amount of message transmission of [15]

[illegible] sec_mss 집합의 서버 수를 N이라 하고, 수행되는 이동 트랜잭션의 수를 M이라 한다면, 필요한 총 메시지 양은 식 1과 같다.

(식 1) [15] 알고리즘에서의 메시지 전송량

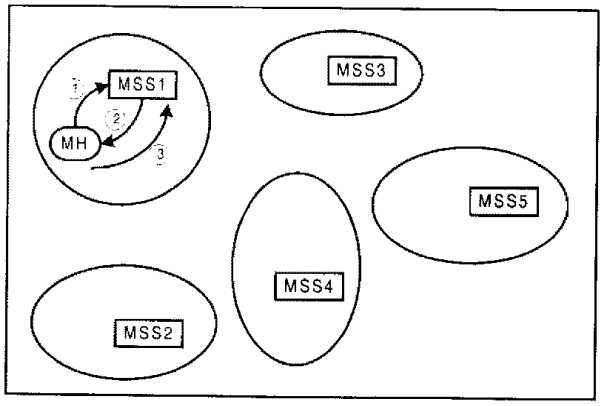
$$(2N+3)M \quad (N : 2차\ 서버\ 수,\ M : 이동\ 트랜잭션의\ 수)$$

2. RSL-MT

그림 10은 RSL-MT 알고리즘의 메시지 전송량을 표현하고 있다. 이동 호스트가 서버에게 트랜잭션을 제출(1)하고, 서버가 이동 호스트에게 트랜잭션 수행 결과를 통보(2), 그리고 이동 호스트가 서버에게 결과 수신 인식신호를 전송(3)하는 메시지가 필요하다. 서버의 수에 관계없이 수행되는 이동 트랜잭션의 수를 M이라 하였을 때 총 메시지 전송량은 식 2와 같다.

(식 2) RSL-MT 방법에서의 메시지 전송량

$$3M \quad (M : 이동\ 트랜잭션의\ 수)$$



(그림 10) RSL-MT에서의 메시지 전송량
(Fig. 10) Amount of message transmission of RSL-MT

5.2 알고리즘 수행에 필요한 기억 공간

① [15] 알고리즘

그림 9에서 이동 호스트의 상태 정보는 MSS_1뿐만 아니라 MSS_2, MSS_3, MSS_4에 중복저장되어 있다. 여기에서 서버가 유지하는 이동 호스트의 상태 정보로 로그 화일만을 고려한다면, 이 로그 화일이 sec_mss의 서버 수만큼 중복 저장된다. 한 이동 호스트가 제출한 트랜잭션들의 로그 화일 용량이 L byte라 하고, sec_mss의 서버 수를 N이라 한다면 필요한 총 메모리 공간은 식 3과 같다.

(식 3) [15] 알고리즘이 필요로 하는 기억 공간

$$L * (N + 1) \quad (L : 로그\ 화일의\ 용량,\ N : 2차\ 서버\ 수)$$

② RSL-MT

본 논문에서 이동 호스트의 상태 정보는 하나의 서버에만 저장되고 이동 호스트에는 제출 리스트가 유지된다. 제출 리스트에는 서버로 제출하는 트랜잭션의 목록만을 유지하므로, 제출 리스트가 차지하는 기억 공간은 서버에 있는 한 이동 호스트에 대한 상태정보인 로그 화일이 차지하는 공간보다 훨씬 덜 차지한다. 그래서 이동 호스트가 제출한 트랜잭션들의 로그 화일이 L byte 차지한다고 할 때, RSL-MT 알고리즘에서 필요로 하는 총 메모리 공간은 2L보다 훨씬 작다($<< 2L$). 그러므로 [15] 알고리즘의 식 3에서 sec_mss의 서버 수가 최소 1개라 하더라도 필요한 공간은 2L이므로 본 논문에서 필요로 하는 공간보다 많게 된다.

5.3 알고리즘 수행 시간

두 알고리즘의 성능을 비교하는 세 번째 기준으로 알고리즘 수행시간을 들 수 있는데, 이는 메시지 전송량에 비례한다고 볼 수 있다. 즉, 동일한 망 속도에서 메시지 전송량이 많으면 알고리즘을 수행하는데 소요되는 시간이 길어진다. 따라서 앞에서 설명한 메시지 전송량을 비교해 볼 때 [15]의 알고리즘이 본 논문에서 제시한 알고리즘보다 훨씬 많은 메시지를 필요로 하므로, 알고리즘 수행시간 또한 많이 소요된다고 할 수 있다.

본 논문에서 제시한 RSL-MT 방법에서는 이동 호스트의 상태 정보가 현재 이동 호스트가 위치한 서버에만 저장되도록 하였다. 이는 [15]에서 제시한 방법의 단점을 해결하였는데, [15]에서는 이동 호스트에 대한 상태 정보를 몇 개의 2차 서버에 중복 저장시켰다. 이로 인해 이동 호스트의 상태 정보를 저장하는데 많은 기억공간을 요구하게 되고, 이동 호스트의 상태 정보를 저장시킬 2차 서버를 선택해야 한다. 그리고 여러 서버에 중복 저장된 이동 호스트의 상태 정보를 일관성 있게 유지하기 위해 서버와 서버간, 서버와 이동 호스트간에 메시지를 주고받아야 하므로, 이에 대한 부가적인 메시지 교환이 필요하게 되고 메시지의 전달 시간 차이로 인한 지연시간이 발생하게 된다. 또한 서버가 고장났을 때 2차 서버 중 하나로 이동해야 하고,

더불어 2차 서버의 집합도 새롭게 결정되어야 한다. 결국 이동 호스트의 상태 정보를 중복 저장시킴으로써 많은 오버헤드(overhead)가 발생하게 된다.

6. 결 론

이동 컴퓨팅 환경은 이동 호스트의 이동성, 배터리의 한계, 무선 통신망의 낮은 신뢰성, 그리고 제한된 대역폭과 같은 특징을 갖고 있는 새롭게 대두되고 있는 환경이다. 따라서 해결해야 할 많은 문제들을 안고 있는데, 특히 회복 기법은 아직 많은 연구들이 이루어지지 않고 있다. 이동 컴퓨팅 환경에 있어 회복 기법은 시스템에서 발생할 수 있는 예측되지 않는 고장에 대해 데이터베이스의 상태를 일관성 있게 유지하고 고장에 견고한 시스템을 구축하기 위해 반드시 필요한 기능이다. 따라서 본 논문에서는 이동 컴퓨팅 환경에서 발생 가능한 고장의 유형 가운데 서버의 고장에 대한 회복 알고리즘을 제시하였다.

관련 연구에서 소개한 [15]와 본 논문과의 공통점은 이동 환경에서 서버 고장에 대한 회복 방법을 제시했다는 점이다. 두 방법 모두 서버의 고장으로 인해 이동 호스트의 수행이 멈추어지지 않게 하는 방법을 제안하였다. 하지만 [15]에서는 이동 호스트의 상태 정보를 여러 서버에 중복 저장시켜야 되므로 많은 기억 공간을 필요로 한다. 또한 중복 저장된 상태정보를 일관되게 유지하기 위해 추가적인 메시지 교환이 요구된다. 본 논문에서 제시한 회복 방법에서는 이동 호스트의 제출 리스트에 저장되어 있는 정보를 이용함으로써 회복을 위한 추가적인 기억 공간을 요구하지 않는다. 제안하는 회복 방법은 무엇보다도 이동 호스트의 이동성과 무선 통신망이 갖는 제약을 고려함으로써 이동 환경에 적용할 수 있는 효율적인 회복 알고리즘을 제안하였다.

참 고 문 헌

[1] Ada Fu, John C.S.Lui, M.H.Wong, Dynamic Policies in Selecting a Caching Set for a Distributed Mobile Computing Environment, CS-TR-95-06, 1995.

[2] Ahmed Elmagarmid, Jin Jing, Tetsuya Furukawa, Wireless Client/Server Computing for Personal Information Service and Application, SIGMOD Record, 1995.

[3] Arup Acharya and B.R.Badrimath, Delivering Multicast Messages in Networks with Mobile Hosts. In Proc. of the 13th Intl. Conf. on Distributed Computing Systems, May, 1993.

[4] Badrinath, B.R., Arup Acharyan, and Tomasz Imiellinski, Impact of Mobility on Distributed Computations. ACM Operating Systems Review, 27(2), April, 1993.

[5] Bernstein P.A., V.Hadzilacos, N.Goodman, Concurrency Control and Recovery for Database Systems, Reading, Mass., Addison Wesley, 1987.

[6] Hwang Buhyun, Three-level Transaction Scheduling in Multidatabase Systems, Ph.D. Thesis, KAIST, 1994.

[7] Ioannidis, J., Duchamp. D., and Maguire, G., IP Based Protocols for Mobile Internetworking. In Proceedings of ACM SIGCOMM Symposium on Communication Architecture and Protocols, pp.235-245, 1991.

[8] Jin Jing, Omran Bukhres, Ahmed Elmagarmid, Distributed Lock Management for Mobile Transactions, 1994.

[9] Johnson, D., and Zwaenepoel, W., Recovery in Distributed Systems Using Optimistic Message Logging and Checkpointing. Journal of Algorithms 11, 3, pp.462-491, 1990.

[10] Koo, R, and Toney, S., Checkpointing and Rollback-Recovery for Distributed Systems. IEEE Trans. Softw. Eng. SE-13, 1, pp.23-31, 1987.

[11] Man Hon Wong, Wing Man Leung, A Caching Policy to Support Read-only Transaction in a Mobile Computing Environment, CS-TR-95-07 May, 1995.

[12] Ravi Prakash, Makesh Singhal, Low-Cost Checkpointing and Failure Recovery in Mobile Computing Systems, 1995.

[13] Schlichiting R.D., and Schneider, F.B., Fail-Stop Processors : An Approach to Designing Fault-Tolerant Distributed Computing Systems. ACM Trans. Comput. Syst. 1, 3, pp.222-238, 1995.

[14] Sistla, V., and Welch, J., Efficient Distributed Recovery Using Message Logging. In Proceedings of ACM Symposium on Principles of Distributed Computing, pp.223-238, 1989.

[15] Sridhar Alagar, Ramki Rajagopalan, and S. Venkatesan, Tolerating Mobile Support Station Failures, Technical Report University of Texas at Dallas, U.S. November, 1993.

[16] Strom, R., and Yemini, S., Optimistic Recovery in Distributed Systems. ACM Trans. Comput. Syst. 3, 3, pp.204-226, 1985.

조 정 란

e-mail : jrcho@www.kwu.ac.kr

1987년 전남대학교 전산통계학과 (학사)

1989년 전남대학교 대학원 전산통계학과(석사)

1993년 전남대학교 대학원 전산통계학과(박사수료)

1994년~현재 광주여자대학교 전산학과 전임강사

관심분야 : 분산처리시스템, 이동 컴퓨팅, 트랜잭션 관리

황 부 현

e-mail : bhhwang@chonnam.chonnam.ac.kr

1978년 숭실대학교 전산학과 졸업 (학사)

1980년 한국과학기술원 전산학과 (공학석사)

1994년 한국과학기술원 전산학과 (공학박사)

1981년~현재 전남대학교 전산학과 교수

관심분야 : 데이타베이스 보안, 이동 컴퓨팅, 트랜잭션관리, 분산 처리 시스템 등