고 가용성 분산 시스템을 위한 효율적인 하이브리드 복제 프로토콜

윤 희 용[†] · 최 성 춘[††]

요 약

분산 시스템에서 가용성을 높이고 전체 시스템의 성능을 향상시키기 위해 데이터는 여러 노드에 중복하여 저장된다. 여기서는 전역적 접근 제어를 위해서 읽기/쓰기 동작을 수행하는데 필요한 노드의 집합을 정의하는 Quorum 프로토콜이 존재한다. Quorum 프로토콜을 사용하는 대표적인 복제 프로토콜인 Tree Quorum 프로토콜은 트리의 높이가 증가할수록 노드의 수가 기하급수적으로 증가하고, Grid 프로토콜은 노드에 장애가 발생하지 않아도 언제나 같은 읽기/쓰기 비용을 갖는다는 단점을 갖고 있다. 따라서, 본 논문에서는 기존 프로토콜의 장점을 가지면서 단점을 해결할 수 있는 새로운 하이브리드 프로토콜을 제안한다. 제안된 하이브리드 프로토콜은 전체적으로는 트리 구조를 가지면서 각 레벨에서는 그리드의 열과 같은 구조를 가짐으로써 노드에 장애가 없을 때에는 Tree Quorum 프로토콜과 같이 적은 동작 비용을 요구하며, 노드에 장애가 존재할 경우에도 기존 프로토콜에 비해 상대적으로 적은 동작 비용과 높은 가용성을 보인다. 그러므로 높은 데이터 가용성이 요구되는 서바이벌 스토리지 시스템에 효율적으로 적용 가능하다. 본 논문에서는 수학적 모델링을 통하여 제안된 프로토콜의 비용과 가용성을 평가하고, 시뮬레이션을 통해 응답시간과 처리율을 기존의 Tree quorum프로토콜과 비교한다.

An Efficient Hybrid Replication Protocol for High Available Distributed System

Hee Yong Youn[†] · Sung Chune Choi[††]

ABSTRACT

In distributed systems data are replicated and stored at several nodes to increase the availability and overall performance. Here Quorum protocol defining a certain set of replicas required for read/write operation exists for global concurrency control. One of the representative replication protocols - the Tree Quorum protocol - has a drawback of rapidly growing number of replicas as the level increases, while the Grid protocol requires the same operation cost even without any failure. In this paper, thus, we propose a new replication protocol called hybrid protocol which capitalizes the merits of the existing protocols and solves the problems of them at the same time. The proposed hybrid protocol has very low operation cost in the absence of failure like the tree quorum protocol, and has relatively lower operation cost and higher availability than existing protocols when failure occurs by employing tree architecture as the overall organization while each level of the tree is organized as a row of a grid architecture. It is thus effective to be applied to survival storage system. We conduct cost and availability analysis of the proposed protocol through mathematical modeling, and response time and throughput are compared with those of the Tree Quorum protocol through computer simulation.

키워드 : 복제 프로토콜, tree quorum, grid, 분산 시스템, 가용성

1. 소 개

오늘날 대용량 분산 컴퓨팅 환경에서 데이터와 서비스의 복제는 데이터의 가용성을 높이고, 전체 시스템의 성능을 향상시키기 위해 필수적인 기술이다 [1]. 또한 다중 노드를 사용함으로써 단일 노드의 사용으로 인한 병목현상 문제를 해결할 수 있다. 그러나 노드의 수가 증가할수록 통신비용은 증가하게 되므로, 전체 시스템을 구성하는 노드 중에 읽기/쓰기 동작을 수행하는 노드의 수는 가능하면 적은 수로 유지되어야 한다[2][3].

분산 시스템에서 모든 데이터는 여러 노드들에 중복하여 저장된다. 다수의 노드에 데이터를 중복 저장하는데 있어 중요한 문제는 노드들에 각기 동일하게 저장된 데이터의 일관성을 유지하는 것이다. 일관성 제어란, 읽기 동작을 수행했을 때 마지막 쓰기 동작에 의해 쓰여진 최신 데이터를 사용자에게 제공하는 것과, 읽기/쓰기 동작과 두개의 쓰기 동작이 동시에 수행되지 않도록 제어하는 것을 의미한다. 이

† 종신회원 : 성균관대학교 정보통신공학부 교수
†† 준 회 원 : 성균관대학교 정보통신공학부 박사과정
논문접수 : 2003년 10월 20일, 심사완료 : 2005년 3월 4일

러한 데이터의 일관성을 유지하기 위하여 일관성 제어 프로토콜은 사용자 동작을 동기화 할 필요가 있다 [4][5][6]. 예를 들어, 각기 다른 사용자로부터 수행되는 쓰기 동작은 다른 노드에 복제된 동일한 데이터를 동시에 변경하도록 허용하지 않도록 해야 한다. 각 노드는 저장장치에 저장된 데이터에 대한 여러 노드들의 접근을 동기화 하기 위하여 중앙 집중된 일관성 제어 기법을 사용하고, 여러 노드의 중복 저장된 데이터에 대한 전역적인 접근 제어를 위해 복제 제어 프로토콜을 사용한다. 락킹 기법은 기본적인 중앙 집중된 일관성 제어 기법으로 사용되며, 복제 제어 프로토콜을 위해 각기 다른 목적으로 다양한 복제 프로토콜이 존재 한다. 전역적인 접근 제어를 위해 사용되는 기존의 대표적인 방식은 읽기/쓰기 동작을 수행하는데 필요한 노드의 집합을 정의하는 Quorum이 존재한다. Quorum은 읽기 동작을 위한 *RQ*(read quorum)과 쓰기 동작을 위한 *WQ*(write quorum)으로 정의된다. 여기서 *WQ*는 *RQ*중에서 적어도 하나 이상의 노드를 중복하여 포함해야 한다.

Read-One/Write-All 프로토콜[7]은 최소 읽기 비용과 최대 읽기 가용성의 장점을 가지는 반면 쓰기 동작을 위해서 최대 통신 비용을 가지는 단점을 가진다. Quorum Consensus [9]와 Dynamic Voting[5] 프로토콜은 우수한 읽기와 쓰기 가용성을 가지는 반면 높은 읽기 비용을 가지는 단점을 보인다. 낮은 읽기 비용을 가지는 복제 프로토콜의 설계는 결국 상대적으로 높은 쓰기 비용을 가지게 되며, 이러한 복제 기법들은 $O(n)$의 동작 비용을 가진다. 이것은 읽기/쓰기 비용이 복제를 위해 존재하는 노드의 수에 선형적으로 의존한다는 것을 의미한다. 이러한 문제를 해결하고, 동작에 필요한 통신 비용을 최소화 하기 위하여 논리 구조의 노드 구성을 이용한 복제 프로토콜이 등장하게 되었다.

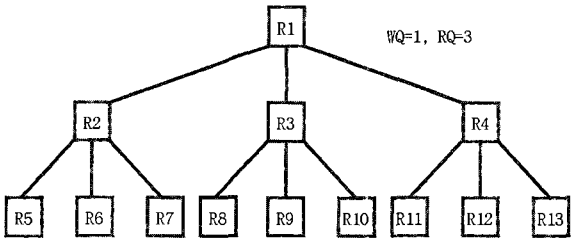
초기의 논리 구조를 이용한 복제 프로토콜은 Multi-Level Voting 프로토콜[10]과 Weighted Voting 프로토콜[9]이 있다. 이러한 프로토콜은 노드의 수가 증가할 때 상대적으로 높은 동작 비용을 가지는 단점을 가진다. 기존의 프로토콜의 단점을 보안하기 위해 대표적인 복제 프로토콜은 Tree Quorum 프로토콜[13]과 Grid 프로토콜[11]이다. Tree Quorum 프로토콜은 트리의 루트 노드를 이용하여 루트 노드의 장애가 발생하지 않을 경우 1의 적은 읽기 비용을 가지는 반면 트리의 높이가 증가할수록 노드의 수가 기하급수적으로 증가한다는 단점을 가지고 있다. 그러므로 만약 노드 장애 발생 확률이 증가한다면, 더 많은 노드를 접근해야 하기 때문에 읽기 비용이 노드의 개수만큼 증가하게 된다. Grid 프로토콜은 노드를 행과 열로 구성하여 논리적인 그리드(grid) 형태로 구성된다. Grid 프로토콜은 Tree Quorum 프로토콜에 비해 높은 가용성을 보이는 반면 노드의 장애가 발생하지 않는 환경에서도 항상 같은 읽기/쓰기 비용을 갖는다는 단점을 가지고 있다. 따라서 본 논문에서는 기존의 Tree Quorum 프로토콜과 Grid 프로토콜이 가지는 장점을 모두 가지면서 기존 프로토콜의 단점을 해결할 수 있는 새로운 하이브리드 복제 프로토콜을 제안한다. 제안된 하이브리드 복제 프로토콜은 노드의 장애가 없을 경우 Tree Quorum프로토콜과 같이 우수한 동작 비용을 가지며, 노드의 장애가 존재할 경우에도 기존의 프로토콜에 비해 상대적으로 적은 동작 비용을 가진다. 본 논문에서는 수학적 모델링을 통하여 제안된 프로토콜의 비용과 가용성을 평가하고, 시뮬레이션을 통해 응답시간과 처리율을 기존의 Tree Quorum프로토콜과 비교한다. 제안된 하이브리드 복제 프로토콜은 더 적은 노드의 구성으로 기존의 Tree Quorum 프로토콜에 비해 우수한 읽기 동작 비용 및 가용성을 가지며, 월등히 뛰어난 응답시간과 처리율을 나타낸다. 따라서 최근 데이터의 가용성을 높이기 위해 연구중인 서바이벌(survival) 저장 시스템에 효율적으로 적용 가능하다.

본 논문의 구성은 다음과 같다. 2장에서는 기존 복제 프로토콜에 대하여 설명하고, 3장에서는 본 논문에서 제안하는 하이브리드 복제 프로토콜을 소개한다. 4장에서는 기존 프로토콜과의 읽기/쓰기 비용, 가용성, 응답시간, 그리고 처리율을 비교하고, 마지막으로 5장에서는 논문의 결론을 제시한다.

2. 이전 작업

2.1 Logarithmic 복제 프로토콜

이번 절에서는 Tree Quorum 프로토콜 중 Logarithmic 프로토콜에 대하여 설명하려 한다. Tree Quorum 프로토콜은 (그림 1)과 같이 트리 구조의 논리적인 구성을 이용한다. 높이 h의 트리로 구성된 n개의 노드가 있다고 가정하면, 잎 노드를 제외한 각 노드 R_i는 S_{R_i}수의 자식 노드를 갖는다. 각 노드들을 위한 읽기와 쓰기 집합을 정의하기 위해 rq_{R_i}와 wq_{R_i}을 정의한다. Tree Quorum 프로토콜은 다양한 rq_{R_i}와 wq_{R_i}값을 가질 수 있으며, 그에 따라 다른 성능을 보인다. 여기서 rq_{R_i}와 wq_{R_i}는 읽기 및 쓰기 동작을 수행할 때, 반드시 읽어야 하는 자식의 수를 의미한다. 이때 $rq_{R_i}=S_{R_i}$의 값과 $Wq_{R_i}=1$의 값을 가지는 프로토콜을 Logarithmic 프로토콜이라 하며, Tree Quorum 프로토콜에서 가장 좋은 성능을 나타낸다.



(그림 1) 13개의 노드를 갖는 높이 3의 트리.

알고리즘

읽기 동작 : 루트 노드 또는 만약 루트 노드가 장애가 발생하면 루트의 rq_{R_i}를 읽게 된다. 루트의 자식

노드는 다시 루트 노드와 같이 동작하게 된다.

쓰기 동작 : 루트 노드 그리고 루트의 wq_R를 쓰게 된다. 선택된 자식 노드는 다시 루트 노드와 같이 동작하게 된다.

읽기 동작을 위해 가능한 노드들의 집합은 {R1}, {R2, R3, R4}, {R3, R4, R5, R6, R7}, 그리고 {R3, R5, R6, R7, R11, R12, R13}이 된다. 쓰기 동작을 위해 필요한 노드들의 집합은 {R1, R2, R6}, {R1, R3, R8}, 그리고 {R1, R4, R11}이 된다.

복제된 데이터들의 일관성 유지를 위해, 선택된 *RQ*와 *WQ*은 다음의 몇 가지 조건을 만족해야 한다. 같은 데이터에 대한 읽기 동작과 쓰기 동작이 동시에 발생하지 않도록 읽기/쓰기 충돌을 인지하기 위해, *RQ* 집합의 노드 중 반드시 하나 이상의 노드는 WQ 집합에 포함되어야 한다. 이 조건은 $rq_R + wq_R > V_R$에 의하여 만족될 수 있으며, 여기서 V_R는 자식의 수를 의미한다. 두개의 쓰기 동작의 충돌을 인지하기 위해, 두개의 *WQ* 집합의 노드 중에 최소한 하나의 노드는 두 *WQ*집합 모두에 포함되어 있어야 한다. 이것은 쓰기 동작을 위한 *WQ*이 항상 루트 노드를 포함하기 때문에 만족할 수 있다.

2.2 Grid 프로토콜

Grid 프로토콜은 (그림 2)와 같이 행과 열의 논리적인 Grid 형태로 구성된다. 읽기와 쓰기 동작들은 읽기/쓰기와 쓰기/쓰기와 같은 동작간의 충돌을 감지하기 위해 행과 열에 대한 *RQ*과 *WQ*을 요구한다. Grid 프로토콜에서 *RQ*은 각 열에서 임의의 하나의 노드를 선택하여 구성되고, *WQ*은 각 열에서 임의의 하나의 노드와 임의의 하나의 전체 열로 구성된다. (그림 2)는 4행 4열의 16개 노드를 갖는 그리드 네트웍 구조를 보여준다.

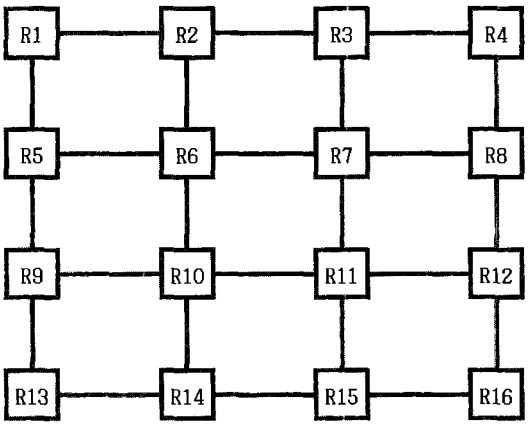
알고리즘

읽기 동작 : 각 행의 임의의 하나의 노드를 읽는다. 여기서 각 행에서 선택된 노드의 집합을 C-cover라고 부른다.

쓰기 동작 : C-cover와 임의의 열 전체에 대하여 쓰기 동작을 수행한다.

(그림 2)에서 읽기 동작을 위해 가능한 노드들의 집합은 {R1, R6, R3, R12}와 {R1, R6, R7, R8}이 된다. 쓰기 동작을 위해 필요한 노드들의 집합은 {R1, R10, R3, R4, R2, R6, R14}와 {R9, R6, R3, R8, R7, R11, R15}이다.

여기서 쓰기 동작은 각 열에서 하나의 노드와 임의의 전체 열에 대하여 동작하게 되므로 두 개의 쓰기 동작은 동시에 수행될 수 없다. 만약 하나의 쓰기 동작이 전체 열에 대하여, 즉 C-cover를 사용되고 있다면 다른 쓰기 동작은 그 열에 접근할 수 없으므로 쓰기 요청은 대기 상태가 된다. 물론 이것은 쓰기와 읽기 동작이 동시에 수행될 수 없는 이유가 된다. Grid 프로토콜은 장애의 발생 여부에 상관없이 *RQ*과 *WQ*의 수가 항상 동일하므로 고정된 동작 비용을 가지게 된다. 하지만, Grid 프로토콜은 Logarithmic 프로토콜에 비해 높은 가용성을 가진다. 다음 장에서는 Logarithmic 프로토콜의 단점을 해결하고, Grid 프로토콜과 같이 보다 안정된 성능을 보이는 제안하는 대칭 트리 프로토콜에 대하여 설명한다.



(그림 2) 16개의 노드를 갖는 그리드 네트워크.

3. 제안된 하이브리드 프로토콜

3.1 동기

기존의 대부분의 복제 프로토콜은 읽기와 쓰기 동작을 위해 복사본을 가지고 있는 대부분의 노드를 접근해야 한다. 예를 들어, 16개의 노드를 갖는 Tree Quorum Consensus 프로토콜의 경우, 읽기 동작과 쓰기 동작을 위해 필요한 노드 수의 합은 전체 노드의 개수 보다 많은 노드를 필요로 한다. 하지만, 만약 노드를 논리적 형태로 구성한다면, 동작을 위해 접근하는 노드의 수는 더욱 줄어들게 된다. 예를 들어, 16개의 노드를 갖는 Grid 프로토콜의 경우, 읽기 동작을 위해 필요한 노드의 수는 4이고, 쓰기 동작을 위해 필요한 노드의 수는 7이 된다. 하지만 Grid 프로토콜은 장애의 발생 여부에 상관없이 *RQ*과 *WQ*의 수가 항상 동일하므로 고정된 동작 비용을 가지게 된다. 그러므로 장애가 발생하는 않는 환경을 위해 더욱 효과적인 다른 구조를 필요로 하게 되었다. 이를 위해 제안된 프로토콜이 Tree Quorum 프로토콜이다. 13개의 노드를 갖는 Tree Quorum 프로토콜은 최선의 경우 읽기 비용으로 1을 가지며, 쓰기 비용으로 3을 가진다. 하지만 Tree Quorum 프로토콜은 트리의 레벨이 증가함에 따라 노드의 수가 급격하게 증가한다는 단점을 가지고 있다. 예를 들어, 5레벨의 Logarithmic 프로토콜의 경우는 최악의 경우 읽기 비용으로 81개의 노드를 필요로 하게 된다. 또한 Grid 프로토콜에 비해 낮은 가용성을 가지게 된다.

그러므로 본 논문에서는 이전에 언급한 두개의 대표적인

프로토콜이 가지는 문제점을 해결할 수 있는 새로운 하이브리드 프로토콜을 제안한다. 제안된 프로토콜은 Tree Quorum 프로토콜과 달리 트리의 레벨이 증가함에 따라 노드의 수가 급격하게 증가하지 않는다. 그 결과 평균 읽기 비용은 *Logarithmic* 프로토콜에 비해 더욱 적게 되고, 쓰기 가용성은 더욱 높아지게 된다.

3.2 제안된 하이브리드 프로토콜

제안된 하이브리드 프로토콜은 Tree Quorum프로토콜과 Grid 프로토콜의 조합에 의하여 구성될 수 있다. 그러므로 제안된 하이브리드 프로토콜은 Tree Quorum 프로토콜과 같이 최선의 경우 최소 읽기 동작 비용의 장점을 가지며, Grid 프로토콜과 같이 적은 수의 노드를 이용하여 전체 시스템을 구성할 수 있는 장점을 가진다. 제안된 프로토콜의 노드 구성은 (그림 3)과 같이 삼각형 구조로 구성된다. 첫번째 열은 하나의 노드로 구성되고, 두 번째 열은 세 개의 노드로 구성되므로, n번째 열은 2n+1개의 노드로 구성된다.

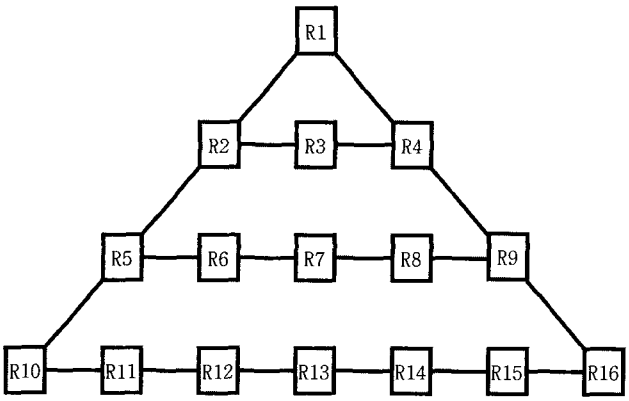
알고리즘

읽기 동작 : 읽기 동작은 임의의 열의 전체 노드에 대하여 수행하고, 만약 노드의 장애가 발생할 경우 다음 레벨로 이동하여 같은 동작을 수행하게 된다. 읽기 동작은 마지막 레벨에 도달할 때까지 계속 된다.

쓰기 동작 : 쓰기 동작은 첫번째 레벨부터 시작하여 마지막 레벨까지 각 레벨에서 임의의 하나의 노드를 선택하여 수행하게 된다.

(그림 3)에서 읽기 동작을 위해 가능한 노드들의 집합은 {R1}, {R2, R3, R4}, 그리고 {R5, R6, R7, R8, R9}가 된다. 쓰기 동작을 위해 필요한 노드들의 집합은 {R1, R3, R5, R13}과 {R1, R2, R9, R10}이다.

제안된 프로토콜은 이해와 동작이 쉽다는 장점을 가지는 반면 모든 쓰기 동작과 대부분의 읽기 동작이 루트 노드부터 시작하므로 루트 노드가 병목현상이 발생할 수 있다는 단점을 가질 수 있다. 이러한 루트 노드에 대한 병목현상 문제는 임의의 레벨에서 읽기 동작을 수행할 수 있도록 프로토콜의 알고리즘을 바꿈으로써 해결될 수 있다. 실제로, 읽기 동작은 임의의 레벨에서 전체 노드에 대하여 수행하고, 쓰기 동작은 각 레벨에서 하나의 노드에 대하여 수행하므로 알고리즘의 변경에 따른 문제가 발생하지 않는다.



(그림 3) 16개의 노드를 갖는 제안된 하이브리드 프로토콜을 위한 네트워크

3.3 제안된 하이브리드 프로토콜의 동작 예

(그림 4)는 제안된 하이브리드 프로토콜의 동작 예를 보여준다. 이 예는 16개의 노드를 갖는 레벨 4의 구조를 사용한다. (그림 4.a)에서 보는 것과 같이 쓰기 동작은 각 레벨의 하나의 노드에 대하여 쓰기를 수행한다. 읽기 동작을 위해, 만약 루트 노드가 읽기 가능하면, 오직 루트 노드만 읽기를 수행한다. (그림 4.b)에서 보는 것과 같이 만약 노드-0이 가능하지 않으면, 노드-0 대신에 노드-1, 노드-2, 그리고 노드-3에 대하여 읽기 동작을 수행한다. 이전 절에서 말한 것과 같이 읽기 동작은 만약 각 레벨에서 하나 이상의 노드에 장애가 발생하면 다음 레벨로 이동하여 읽기 동작을 수행하게 되고, 마지막 레벨까지 계속된다.

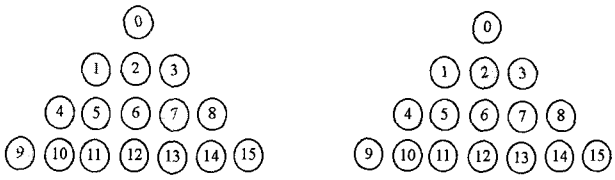
다음의 정의는 제안된 하이브리드 프로토콜의 동작의 정확성을 보증한다.

정의 1: 읽기 동작은 항상 마지막에 쓰기 동작에 의해 쓰여진 최신 결과를 반환한다.

증명: (그림 4)에서 보인 것과 같이, 임의의 RQ와 WQ를 정할 때 $RQ \cap WQ \neq 0$이다. 그러므로, 제안된 프로토콜은 임의의 RQ는 항상 최신의 데이터를 갖는 노드를 포함하도록 보증한다. □

정의 2: 읽기/쓰기 또는 쓰기/쓰기 동작은 동시에 수행될 수 없다.

증명: 데이터에 대한 일관성은 선택된 RQ와 WQ가 읽기/쓰기와 쓰기/쓰기 충돌을 감지할 수 있기 때문에 유지될 수 있다. 제안된 하이브리드 프로토콜에서 읽기/쓰기 충돌은 읽기 동작은 하나의 레벨 전체에 대하여 수행하고 쓰기 동작은 최소한 각 레벨의 하나의 노드에 대하여 수행하기 때문에 읽기 동작을 수행하는 레벨에서 충돌을 검출할 수 있다. 쓰기/쓰기 충돌은 모든 쓰기 동작이 반드시 루트 노드에서부터 시작해야 하기 때문에 검출 가능하다. 그러므로 하나의 노드가 쓰기 동작을 수행하면 다른 노드는 루트 노드에 대한 접근이 불가능하게 되어 두 개의 쓰기 동작이 동시에 수행될 수 없다. □



(a) WQ={0, 1, 7, 9} (b) RQ={1, 2, 3}

(그림 4) 읽기/쓰기 동작의 예

4. 성능 분석

이번 절에서는 제안된 프로토콜의 읽기 및 쓰기 비용과 가용성에 대하여 기존의 Logarithmic 프로토콜과 비교한다. 정확한 성능평가를 위하여 다음과 같은 전제를 사용한다.

- 링크의 실패는 발생하지 않는다.
- 노드의 장애는 발생할 수 있다.
- 노드들의 실패는 독립적이며 실패율은 일정하다.
- 모든 노드는 확률 p의 가용성을 갖는다.
- Logarithmic 프로토콜에서 잎 노드를 제외한 모든 노드는 동일한 S개의 자식 노드를 갖는다.

4.1 비용 분석

비용 분석을 위하여, 읽기 비용과 쓰기 비용을 따로 고려한다. 분산 프로토콜에서 비용이란, 동작의 요청에 따라 동작을 수행하는 노드의 수에 의하여 계산될 수 있다. 만약 루트 노드가 가능하다면, 읽기 동작은 매우 효율적으로 루트 노드만으로 수행될 수 있다. 그러나 만약 모든 동작이 루트노드를 시작으로 수행하게 된다면, 루트 노드는 병목현상이 발생하게 된다. 그러므로 이전 장에서 말한 것과 같이 임의의 레벨에서부터 읽기 동작을 수행할 수 있도록 알고리즘의 수정이 필요하다. 읽기 동작을 위한 레벨을 선택하기 위해 [0,1] 범위 내에서 매개변수 f를 선택하고, 난수 발생에 의하여 변수 x를 만든다. 여기서 f는 레벨이 선택될 확률을 의미한다. 예를 들어 루트 노드가 선택될 확률이 f라면 선택되지 않을 확률은 $(1-f)$가 된다. 이러한 조건에 따라 만약 $x \le f$라면, 루트 노드는 선택되어 지고 읽기 동작을 수행한다. 그렇지 않을 경우 새로운 변수 x를 만들고, 만약 $x \le f$라면 다음 레벨이 선택되어 진다.

4.1.1 Logarithmic 프로토콜

h+1레벨로 구성된 트리 구조가 있다고 가정할 때, 평균 읽기 비용은 다음과 같이 계산될 수 있다. 여기서 f는 각 레벨이 선택될 확률에 의해 결정된다.

$$\begin{aligned} C_{read} &= f + (1-f)\cdot f \cdot RQ + (1-f)^2 \cdot f \cdot RQ^2 + \cdots + (1-f)^{h-1} \cdot f \cdot RQ^{h-1} \\ &\quad + (1-(f + (1-f)\cdot f + (1-f)^2 \cdot f + \cdots + (1-f)^{h-1} \cdot f)) \cdot RQ^h \\ &= f\sum_{k=0}^{h-1}(1-f)^k \cdot RQ^k + \left(1 - f\sum_{k=0}^{h-1}(1-f)^k\right) \cdot RQ^h \\ &= \begin{cases} f \cdot h + 1 & \text{if } (1-f)\cdot RQ = 1 \\ f\dfrac{(1-f)^h RQ^h - 1}{(1-f)RQ - 1} + (1-f)^h \cdot RQ^h & \text{else} \end{cases} \end{aligned}$$

위의 식에서 Logarithmic 프로토콜은 $rq_{R_i} = S_{R_i}$의 값과 $wq_{R_i} = 1$의 값을 가지므로 평균 읽기 비용은 다음과 같이 간략화 될 수 있다.

$$C_{read} = f\frac{((1-f)S)^h - 1}{(1-f)S - 1} + ((1-f)S)^h$$

최소 읽기 비용은 루트노드에 대한 읽기 동작이 성공한 경우이므로 다음과 같다.

$$C_{read(\min)} = 1$$

Logarithmic 프로토콜을 위한 쓰기 비용은 각 레벨에서 하나의 노드에만 쓰기를 수행하기 때문에 다음의 식과 같이 레벨에 의존하게 된다.

$$C_{write} = h + 1$$

4.1.2 하이브리드 프로토콜

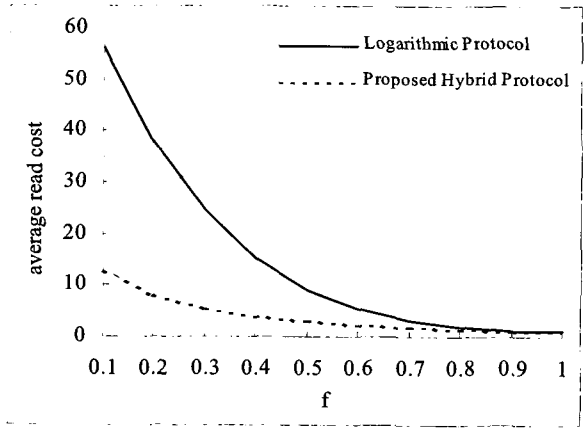
h+1레벨로 구성된 트리 구조가 있다고 가정할 때, 평균 읽기 비용은 다음과 같이 계산될 수 있으며, 최소 읽기 비용은 Logarithmic과 같은 1이 된다.

하이브리드 프로토콜의 쓰기 비용은 Logarithmic 프로토

$$\begin{aligned} C_{read} &= f + (1-f)\cdot f \cdot 3 + (1-f)^2 \cdot f \cdot 5 + \cdots + (1-f)^{h-1} \cdot f \cdot (2(h-1)+1) \\ &\quad + (1-(f + (1-f)\cdot f + (1-f)^2 + \cdots + (1-f)^{h-1} \cdot f)) \cdot (2h-1) \\ &= f\sum_{k=0}^{h-1}(1-f)^k \cdot (2k+1) + \left(1 - f\sum_{k=0}^{h-1}(1-f)^k\right) \cdot (2h-1) \end{aligned}$$

콜과 동일하게 각 레벨에서 하나의 노드에 대하여 쓰기를 수행하므로 같은 값을 가지게 된다.

$$C_{write} = h+1$$



(그림 5) 121개의 노드 구성에서의 평균 읽기 비용 비교

(그림 5)는 121개의 노드를 갖는 두 프로토콜의 평균 읽기 비용을 비교한 결과이다. 평균 쓰기 비용은 두개의 프로토콜 모두 레벨에 의존하게 된다. Logarithmic 프로토콜의 경우는 121개의 노드를 구성하기 위해 5레벨로 구성되므로 쓰기 비용으로 5의 값을 가지고, 제안된 하이브리드 프로토콜 경우는 11의 값을 가지게 된다. (그림 5)에서 보는 것과 같이 같은 수의 노드를 가질 때 평균 읽기 비용은 제안된 프로토콜이 Logarithmic 프로토콜에 비해 훨씬 우수한 것으로 나타난다. 이러한 결과를 보이는 이유는, 121개의 노드를 가지기 위해서 Logarithmic 프로토콜의 경우는 마지막 레벨의 노드의 수가 81개로 구성되지만, 제안된 프로토콜의 경우는 마지막 레벨이 21개의 노드만으로 구성 되기 때문에, 최악의 경우 읽기 비용이 Logarithmic 프로토콜 보다 적게 된다. 또한 f의 값이 1일 경우에는 루트 노드가 선택되어질 최선의 경우이므로 1의 값을 가지는 것을 볼 수 있다.

4.2 가용성 분석

4.2.1 Logarithmic 프로토콜

높이 l로 구성된 트리에서 Tree Quorum 프로토콜에 의해 수행된 읽기 동작의 가용성은 다음의 식과 같이 구해질 수 있다. 읽기 동작은 상위레벨에서 하위레벨로 노드의 실패 여부에 따라 수행되기 때문에 재귀적으로 계산되어 진다.

$$\wp_{read}^{(l)} = p + (1-p)\sum_{k=RQ}^{S}\binom{S}{k}\left(\wp_{read}^{(l-1)}\right)^{k}\left(1-\wp_{read}^{(l-1)}\right)^{S-k}$$

여기서 $\wp_{read}^{(0)}$은 p가 되며 루트 노드의 가용성을 나타낸다. Logarithmic 프로토콜은 RQ의 값이 S로 고정되므로 읽기 동작의 가용성은 다음 식과 같다.

$$\wp_{read}^{(l)} = p + (1-p)\left(\wp_{read}^{(l-1)}\right)^{S}$$

Tree Quorum 프로토콜의 쓰기 가용성은 다음 식과 같이 나타낼 수 있으며, $\wp_{write}^{(0)}$은 p이다..

$$\wp_{write}^{(l)} = p\sum_{k=WQ}^{S}\binom{S}{k}\left(\wp_{write}^{(l-1)}\right)^{k}\left(1-\wp_{write}^{(l-1)}\right)^{S-k}$$

위의 식으로부터 Logarithmic 프로토콜의 쓰기 가용성을 다음과 같이 구할 수 있다.

$$\wp_{write}^{(l)} = p\left(1-\left(1-\wp_{write}^{(l-1)}\right)^{S}\right)$$

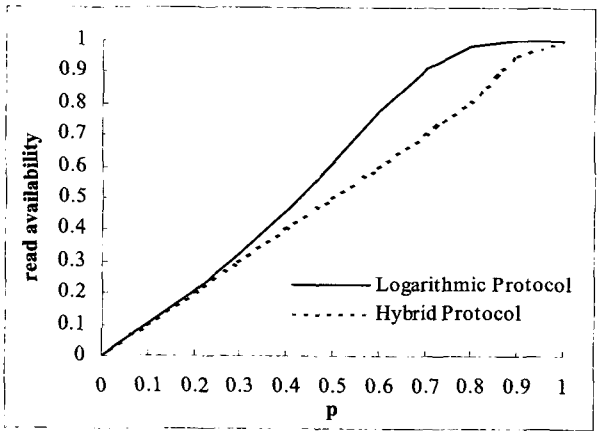
4.2.2 하이브리드 프로토콜

제안된 하이브리드 프로토콜의 읽기 가용성은 다음의 식과 같다. 여기서 S는 각 레벨에서의 노드 수를 의미하므로 $2l+1$이 된다.

$$\wp_{read}^{(l)} = p + (1-p)\left(\wp_{read}^{(l-1)}\right)^{S}$$

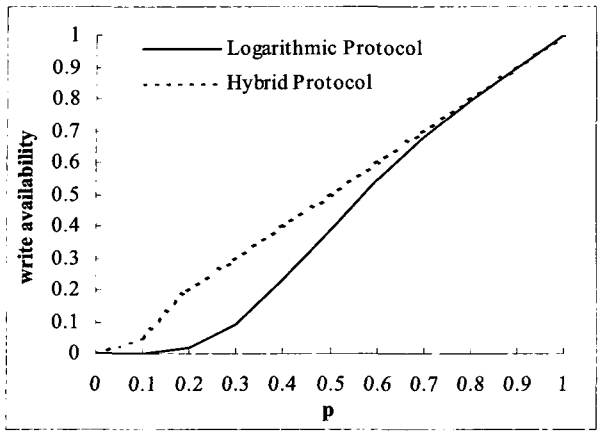
쓰기 가용성은 다음의 식과 같으며, S는 읽기 가용성에서와 같이 레벨의 노드 수를 나타낸다.

$$\wp_{write}^{(l)} = p\left(1-\left(1-\wp_{write}^{(l-1)}\right)^{S}\right)$$



(그림 6) 읽기 가용성 비교

(그림 6)과 (그림 7)은 두 프로토콜의 읽기 가용성과 쓰기 가용성을 위의 수식적인 분석에 의해 비교한 결과이다. 두 프로토콜의 가용성은 비슷한 결과를 가져오는 것을 볼 수 있다. 읽기 가용성의 경우, Logarithmic 프로토콜은 노드



(그림 7) 쓰기 가용성 비교

의 장애가 발생할 경우 같은 수의 자식 노드에 대하여 읽기 동작을 수행하는 반면 제안된 하이브리드 프로토콜은 읽기를 수행하는 노드의 수가 증가하므로 Logarithmic 프로토콜에 비해 비교적 낮은 가용성을 가지게 된다. 하지만 읽기 비용이 Logarithmic 프로토콜에 비해 적으므로 실제 가용성은 수식적 분석보다 더욱 높게 된다. 쓰기 가용성의 경우, 제안된 하이브리드 프로토콜의 경우 레벨이 증가할수록 선택될 수 있는 노드의 수가 증가하므로 Logarithmic 프로토콜에 비해 높은 가용성을 갖게 된다.

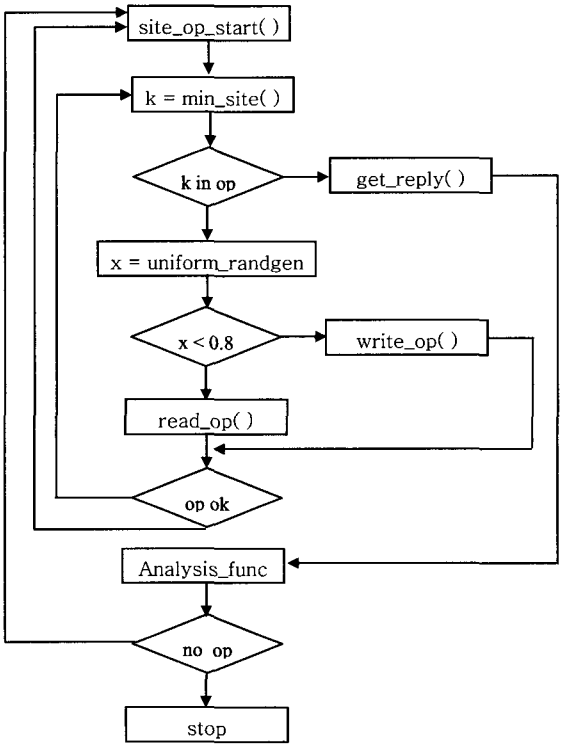
4.3 응답시간과 처리율

제안된 하이브리드 프로토콜과 Logarithmic 프로토콜의 응답시간과 처리율을 비교하기 위해 시뮬레이션을 수행한다. 응답 시간은 노드에 동작이 수행되기 시작한 시간부터 동작이 끝나는 시간을 의미하고, 처리율은 단위 시간동안에 성공한 동작의 수를 의미한다. 시뮬레이션의 정확성을 위해 다음의 가정이 주어진다.

가정

- 노드들은 인터넷에 의해 연결되고, 노드들 간에 통신은 브로드캐스트 된다.
- 각 사이트의 요청 시간은 지수 분포 함수에 따라 발생한다.
- 동작의 시작은 매체에 접근한 시간부터 시작된다.
- 모든 노드는 전체 노드의 수와 동일한 크기의 큐를 갖는다. 이것은 각 노드는 다른 모든 노드들로부터 요청을 받을 수 있음을 의미한다.
- 모든 노드는 큐의 순서에 기초하여 다른 노드들로부터 요청에 응답한다.
- 노드가 다른 노드에 요청을 보냈을 때, 동작에 참여하는 다른 모든 노드에 요청이 같은 시간에 수신된다.
- 락 요청 통신 시간 = 0.0001, 락 처리 시간 = 0.00002, 요청 처리 시간 = 0.0003, 응답 통신 시간 = 0.0003, 그리고 동작 시간 = 0.00005

(그림 8)은 제안된 하이브리드 프로토콜의 응답시간과 처리율을 구하기 위한 시뮬레이션 흐름도를 보여준다. 시뮬레이션은 세 부분으로 나뉘어진다. min_site는 전체 노드 중에 가장 작은 동작 시간을 가지고 있는 노드를 찾는 동작을 수행한다. 동작 시간은 각 노드가 동작을 시작하는 시간을 의미한다. read_op와 write_op는 실제 동작을 수행하기 위해 구성된 부분이다. 마지막으로 anaysis_func는 노드의 타임스탬프를 관리하고, 시뮬레이션 동안 응답시간과 처리율을 계산한다. 각 노드의 동작 시작 시간은 지수 분포 함수에 의하여 결정되고, 시뮬레이션의 초기화 과정에서 전체 노드에 할당되며, 동작을 수행하고 끝날 때마다 현재 시간의 이후 시간으로 다시 할당된다. 각 노드의 동작 유형(읽기와 쓰기)은 균일 분포 함수에 의하여 결정된다.



(그림 8) 하이브리드 복제 프로토콜의 시뮬레이션을 위한 흐름도

```
read(min_node)
   lev = uniform_randgen( );
   WHILE(lev < last_level)
     IF( avail_site())
/* all sites of level(lev) available */
         get reply times for quorum;
         RETURN(reply time);
     ELSE
         lev = lev + 1;
     ENDIF
   ENDWHILE
   /* get next time for next operation*/
   RETURN(exponential_randgen( ))
```

```
write()
  l=0;
  WHILE(l>no_of_lev)
    IF(avail_site())
    /* at least one site available */
     l=l+1;
    ELSE
  /* get next time for next operation*/
     RETURN(not OK)
    ENDIF
  ENDWHILE
  get reply times for quorum;
RETURN(reply time);
```

```
avail_site(i)
  IF(site i is available)
    RETUN(OK)
  ELSE
IF(site(i) is not last level)
      k=0;
      FOR(all sites j of level)
        A[k] = avail_site(j)
        k=k+1;
      ENDFOR
    ELSE
      RETURN(not OK);
    ENDIF
  ENDIF
  IF(all sites are OK)
    RETUN(OK)
  ELSE
    RETURN(not OK)
  ENDIF
```

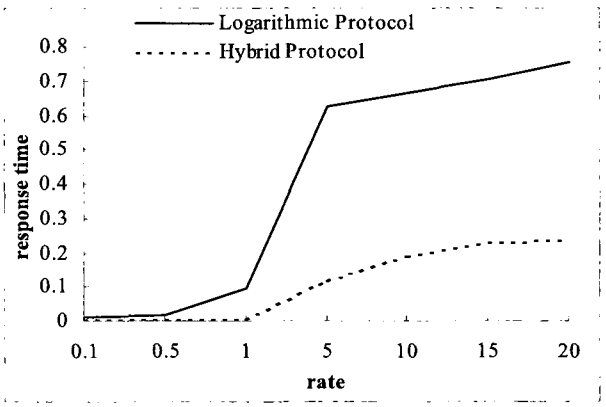
(그림 9) 읽기와 쓰기 동작을 위한 알고리즘

시뮬레이션을 위한 읽기와 쓰기 동작을 위한 의사코드는 (그림 9)와 같다. 복제된 데이터의 일관성 유지를 위해, 락킹 동작은 읽기와 쓰기 동작이 수행되기 전에 반드시 수행되어야 한다. RQ와 WQ에 포함된 노드들은 avail_site 함수에 의하여 락킹 동작을 수행하게 된다. 시뮬레이션 결과는 노드의 장애 발생과 동작 시작 시간의 임의성으로 인하여 100,000,000번의 시뮬레이션을 수행하여 평균치로 계산된다.

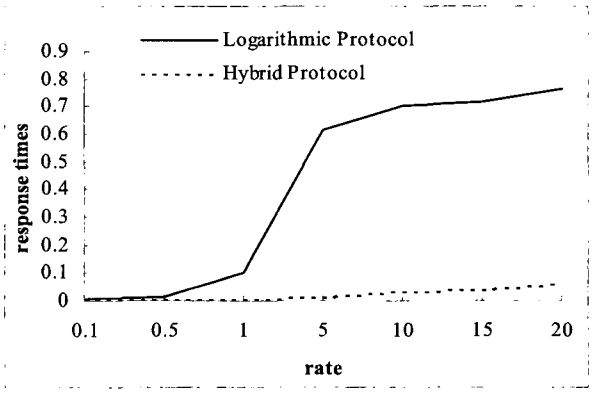
4.3.1 응답시간

(그림 10)은 121개의 노드를 갖는 두 프로토콜의 응답시간을 나타낸다. 제안된 프로토콜의 읽기와 쓰기 응답 시간은 Logarithmic 프로토콜에 비해 매우 적게 나타난다. 동작 요청율이 증가함에 따라 두 프로토콜간의 차이는 더욱 커지게 된다. 또한 노드의 수가 증가할수록, 제안된 하이브리드 프로토콜의 응답 시간은 Logarithmic 프로토콜에 비해 더욱 작게 나타난다. 이것은 제안된 하이브리드 프로토콜이 Logarithmic 프로토콜에 비해 동작을 수행하기 위해 필요한 전체 노드의 개수가 더욱 작기 때문에 이러한 결과를 예측할 수 있다. 동작 요청율은 단위 시간동안 얼마나 많은 동작이 요청되었는지를 나타내며, 값이 커질수록 더욱 많은 요청이 요구되는 것을 의미한다. 동작 요청율이 증가할수록 Logarithmic 프로토콜은 각 노드에서 더욱 많은 요청이 큐에 들어가게 되고 각 요청의 대기시간이 증가하게 되므로 응답시간의 차이는 더욱 커지게 된다.

쓰기 동작에서 Logarithmic 프로토콜의 응답시간이 급격하게 증가하는 이유는, Logarithmic 프로토콜은 자신의 자식 노드에 대해서만 쓰기 동작을 수행한다. 따라서 자신의 자식 노드가 다른 요청을 모두 마치기 전까지 쓰기 동작을 수행할 수 없으므로 전체 대기 시간이 증가하게 된다. 이것은 Logarithmic 프로토콜의 레벨이 증가하거나 요청율이 증가할 경우 더욱 커지게 된다. 하지만 제안하는 하이브리드 프로토콜의 쓰기 동작은 레벨마다 하나의 노드에 대하여 수행하므로, 해당 레벨의 임의의 노드가 작업이 끝나면 즉시 쓰기 동작을 수행할 수 있다. 따라서 쓰기 동작에 필요한 노드의 수는 증가하지만 하위 레벨로 내려갈수록 선택 가능한 노드의 수가 증가하므로 적은 대기 시간을 가지게 된다.



(그림 10) 읽기 응답시간 비교

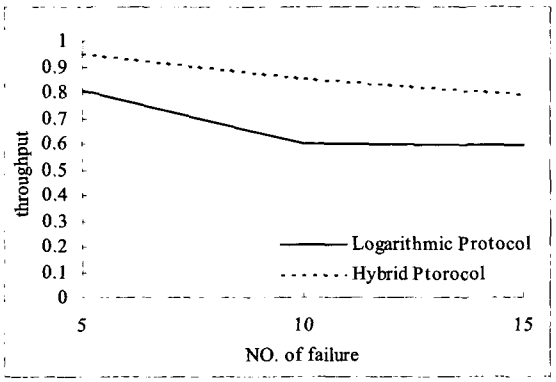


(그림 11) 쓰기 응답시간 비교

4.3.2 처리율

(그림 12)는 장애가 발생한 노드의 수에 따른 두 프로토콜의 처리율을 비교한 결과이다. 같은 수의 장애가 발생하였을 때, 제안된 하이브리드 프로토콜의 성공한 동작의 전체 수는 Logarithmic 프로토콜에 비해 더욱 많으므로 처리

율은 높게 나타난다. 이것은 Logarithmic 프로토콜이 제안된 하이브리드 프로토콜에 비해 더욱 적은 레벨 수를 가지고 있으므로 장애가 발생할 경우 전체 동작의 장애 발생확률도 증가하기 때문에 나타나는 결과이다. 즉 제안된 하이브리드 프로토콜은 Logarithmic 프로토콜에 비해 동작을 성공시키기 위하여 더욱 많은 노드들의 조합이 가능하다는 것을 의미하게 된다.



(그림 12) 다양한 노드 장애 발생 수에 따른 처리율 비교

5. 결론 및 향후 과제

본 논문은 기존의 Logarithmic 프로토콜에 비해 더욱 낮은 비용, 적은 응답시간, 높은 가용성, 그리고 높은 처리율을 갖는 새로운 하이브리드 복제 프로토콜을 제안한다. 제안된 하이브리드 프로토콜의 주요 장점은 Logarithmic 프로토콜에 비해 노드의 수가 급격하게 증가하지 않고, 구현 및 이해가 쉽다는 것이다.

복제 프로토콜의 선택은 환경에 따라 필요한 주요 성능 요소에 의존하여 사용하게 된다. 제안된 프로토콜은 낮은 비용과 높은 쓰기 가용성을 요구하는 환경에 특히 유용하게 사용된다. 불행하게도, 노드의 논리 구조는 노드들의 특정한 조합들이 사용된 구조에 의존하여 동작에 필요한 노드를 선택해야 하므로 전체 통신 비용을 낮추는 결과 뿐만 아니라 가용성을 낮추는 결과를 가져왔다. 그러므로, 가용성에 관련된 단점은 동작 비용을 감소시킨다는 본질적인 이득에 대하여 무시 될 수 있다. 또한 제안된 프로토콜은 최근 데이터의 가용성을 높이기 위해 연구중인 서바이벌(survival) 저장 시스템에 효율적으로 적용 가능하다.

향후 연구과제는 동작 비용과 함께 가용성을 높여 성능을 최대화할 수 있는 새로운 프로토콜을 연구하는 것이다.

참 고 문 헌

[1] C. Amza,, A.L. Cox, W. Zwaenepoel, **Data replication strategies for fault tolerance and availability on commodity clusters,** *Proc. Int'l Conf on Dependable Systems and Networks (DSN)*, 2000, 459-467

[2] K. Arai, K. Tanaka, M. Takizawa, **Group protocol for quorum-based replication,** *Proc. Seventh Int'l Conf on Parallel and Distributed Systems*, 2000, 57-64.

[3] G. Alonso, Partial Database Replication and Group Communication Primitives, *Proc. of the 2nd European Research Seminar on Advances in Distributed Systems (ERSADS'97)*, March 1997, 171-176.

[4] R.H. Thomas. A Majority Consensus Approach to Concurrency Control for Multiple Copy Data-based, *ACM Trans on Database Systems*, 4(2),1979,180-207.

[5] D. Davcev, A Dynamic Voting Scheme in Distributed Systems. *IEEE Trans on Software Engineering*, 15(1), 1989, 93-97.

[6] D. Saha, S. Rangarajan, S.K. Tripathi, An Analysis of the Average Message Overhead in Replica Control Protocols, *IEEE Trans on Parallel and Distributed Systems*, 7(10), Oct. 1996, 1026-1034.

[7] P.A. Bernstein and N. Goodman, An Algorithm for Concurrency Control and Recovery in Replicated Distributed Databases, *ACM Trans on Distributed Systems*, 9(4), 1984, 596-615.

[8] B. Freisleben, H.H. Koch, and O. Theel, Designing Multi-Level Quorum Schemes for Highly Replicated Data. *Proc. of the 1991 Pacific Rim Int'l Symp on Fault Tolerant Systems, IEEE*, 1991, 154-159.

[9] D.K. Gifford, Weighted Voting for Replicated Data, *Proc. of the 7th ACM Symp on Operating Systems Principles*, 1979, 150-162.

[10] A. Kumar, Hierarchical quotum consensus : A new algorithm for managing replicated data", IEEE Transaction on Computer, 40(9), September 1991, 996-1004.

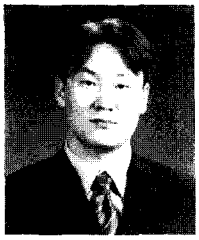
[11] S. Cheung, M. Ammar, and M. Ahamad, The Grid Protocol: A High Performance Scheme for Maintaining Replicated Data, *Proc of the 6th Int'l Conf on Data Engineering*, 1990, 438-445.

[12] T. Anderson, Y. Breitbart, H. Korth, A. Wool, Replication, Consistency, and Practicality: Are These Mutually Exclusive?, *ACM SIGMOD Int'l Conf on Management of Data*, June 1998, 484-495.

[13] D. Agrawal and A. El Abbadi, The tree Quorum protocol: An Efficient Approach for Managing Replicated Data, Proc of the 16th Very Large Databases (VLDB) Conf, 1990, 243-254.

윤 희 용

e-mail : youn@ece.skku.ac.kr
1977년 서울대학교 전자공학과(학사)
1979년 서울대학교 전자공학과(석사)
1988년 Univ. of Massachusetts 컴퓨터공학과(박사)
1988년~1991년 Univ. of North Texas. 조교수
1991년~1999년 Univ. of Texas Arlington 부교수
2000년~현재 성균관대학교 정보통신공학부 교수
관심분야 : 모바일 컴퓨팅, 분산처리, 유비쿼터스컴퓨팅 등



최 성 춘

e-mail : choisc@ece.skku.ac.kr
2001년 남서울대학교 정보통신공학부(학사)
2003년 성균관대학교 정보통신공학부(석사)
2003년~현재 성균관대학교 정보통신공학부(박사과정)
관심분야 : 분산컴퓨팅, 모바일컴퓨팅 등