디스크 미러링 시 영구적 고장 복구 기법의 설계 및 성능평가

Design and Performance Evaluation of Permanent Disk Error Recovery Schemes during Disk Mirroring

피 준 일* 홍 현 택** 송 석 일*** 유 재 수****
Jun-Il Pee Hyun-Taek Hong Seok-Il Song Jae-Soo Yoo

요 약

현대 저장 시스템 환경에서 데이터의 가용성과 신뢰성을 높이기 위한 방법으로 디스크 미러링 기법이나 Chained-declustering과 같은 방법을 사용한다. 이런 기법들을 사용할 때 만약 디스크에 고장이 발생하면 이를 효과적으로 복구하기 위한 방법이 필요하다. 이 논문에서는 디스크 미러링과 Chained-declustering에서 디스크 고장이 발생했을 때 가능한 정상상태에 가까운 디스크 I/O 성능을 보장하는 복구 기법을 제안한다. 또한, 시뮬레이션을 통해 기존방법과 제안하는 방법을 다양한 환경에서 성능평가 한다. 시뮬레이션 결과분석을 통해 I/O 응답시간 관점에서 제안한 방법들이 기존방법들에 비해 성능이 매우 향상됨을 보인다.

Abstract

In current storage system environment, we use disk mirroring or Chained-declustering scheme to improve data availability and reliability. If disk error is occurred when these schemes are used, we need efficient recovery schemes for these situations. In this paper, it is proposed that a recovery method of disk mirroring that guarantees I/O performance similar to normal state during disk failure. Also, we execute performance evaluation for our recovery scheme and existing schemes in various environments. We show through the simulation that our algorithm outperforms the classic schemes in terms of I/O response time.

1. 서 론

최근 인터넷의 발전과 함께 인터넷을 사용하는 사용자들의 수가 급증하고 있으며, 사용되는 데이터의 크기 또한 대용량화되고 있다. 따라서, 이를 효율적으로 관리하기 위한 저장시스템들에 대한 요구가 증가하고 있으며, 현재 SAN을 비롯한 여러 저장 시스템들이 연구되고 있다[1.2.3.4]. 이런 저장 시스템들은 데이터의 저장, 유지 및 관리와 같은 전통적인 저장 시스템의 기능뿐만 아니라, 다양하고 복잡한 사용자들의 요구를 어떠한 환경에서도 처리할 수 있어야 한다. 즉 저장시스템에서 유지하는 데이터의 가용성 및 신뢰성이 보장되어야 한다. 이를 보장하기 위해 미러링, 스트라이핑, 패리티를 이용한 스트라이핑등의 RAID 레벨을 제공하는 RAID(Redundant Array of Inexpensive Disks) [5,6,7]가 널리 사용된다.

특히 디스크 미러링은 두 개의 서로 다른 디스크에 같은 데이터를 중복 저장해 놓고 읽기 연산에 대해서 부하균등(load balancing)을 수행한다. 그리고, 한쪽 디스크가 고장이 발생하면 디스크를 교체하고 복구를 수행하면서 정상적인 서비스를 제공한다. 이 방법에서는 디스크 고장 발생 시 다른 한쪽 디스크로 부하가 모두 몰리는 단점이 있다. 이런 디스크 미러링의 문

* 정 회 원 : 충북대학교 정보통신공학과 박사과정
pji@netdb.chungbuk.ac.kr
** 비 회 원 : 충북대학교 정보통신공학과 석사과정
hongry@netdb.chungbuk.ac.kr
*** 비 회 원 : 충북대학교 정보통신공학과 박사과정
prince@netdb.chungbuk.ac.kr
**** 정 회 원 : 충북대학교 전기전자및컴퓨터공학부 부교수
yjs@cbucc.chungbuk.ac.kr

제점을 해결하고 좀더 높은 데이터의 가용성과 신뢰성을 제공하기 위하여 Chained-declustering[8,9,10, 11]과 같이 데이터를 분할하여 여러 디스크에 중복저장해서 신뢰도를 높이고 디스크 고장 발생 시에 여러 디스크에 적절히 부하균등을 하는 방법들이 제시되고 있다.

현재 대부분의 디스크 시스템에 대한 성능 분석은 주로 정상상태에서 이루어졌지만, 디스크 시스템이 실시간 응용 등에서 사용되는 경우에는 디스크 고장 시나 복구 시의 성능도 중요하다. 즉, 정상상태 뿐 아니라 디스크 고장 발생 시 그리고 복구 시에도 얼마나 정상상태에 가깝게 서비스를 제공하는 가도 중요하다. 디스크 시스템에 발생할 수 있는 고장은 일시적인 고장과 영구적인 고장으로 분류할 수 있다. 일시적인 고장은 디스크를 일시적으로 접근하지 못하여 발생하는 것으로 이로 인해 미러링에 사용되는 두 디스크간의 일관성이 깨질 수가 있다. 이와 달리 영구적인 고장은 디스크에 물리적인 고장이 발생하는 경우로 디스크의 교체가 불가피하며 디스크 전체를 복구해야 한다.

이 논문에서는 데이터를 중복저장해서 신뢰도를 높이는 디스크 미러링과 Chained-declustering에서 영구적 고장이 발생했을 때 되도록이면 정상상태와 같은 성능을 보장하면서 고장 복구를 수행하는 방법을 제안한다. 또한 기존의 복구방법과 제안하는 복구방법의 성능을 시뮬레이션을 통해 비교 평가한다.

이 논문의 구성은 다음과 같다. 2장에서 디스크 미러링과 Chained-declustering과 같은 고 가용성 및 신뢰성을 위한 방법과 이들에 대한 복구방법에 대해 기술한다. 3장에서는 디스크 미러링을 위한 보다 개선된 복구기법을 제안한다. 그리고 지금까지는 고려되지 않았던 Chained-declustering에 대한 복구방법에 대해서도 제안한다. 4장에서는 제안한 방법들과 기존의 방법들을 시뮬레이션을 통해서 비교 평가한 결과를 제시하고 분석한다. 마지막으로 5장에서 결론을 맺는다.

2 관련 연구

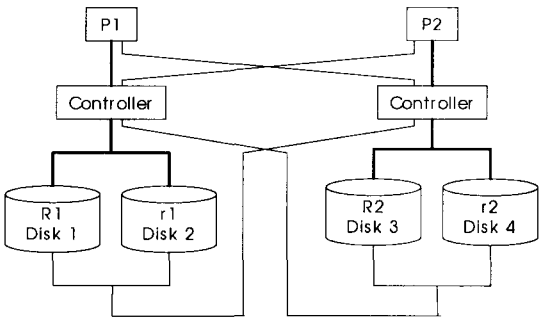
이 장에서는 디스크 고장 발생 시 이를 효과적으로 복구하기 위한 방법을 연구하기 위하여 디스크 미러링과 Chained-declustring 방법에 대하여 먼저 살펴본다. 또한 고장 복구 방법과 관련된 기존 연구에 대하여 살펴본다.

2.1 디스크 미러링

디스크 미러링[8,9,10,11]은 서로 다른 여러 디스크에 같은 데이터를 중복 저장하여 신뢰성과 가용성 그리고 읽기 연산에 대한 부하균등을 제공하기 위한 방법이다. 디스크 미러링의 읽기 연산은 항상 I/O를 수행할 대상이 되는 데이터가 하나 이상 존재하므로 성능 향상을 위한 부하 균등을 고려하여 가장 적절한 위치의 데이터를 선택하여 읽기 연산을 수행한다.

디스크 미러링의 구조는 그림 1과 같다. 각 I/O 컨트롤러는 두 개의 프로세서에 연결되어 있고, 각 디스크 드라이브들은 두 개의 I/O 컨트롤러에 연결되어 있으므로 두 개의 입출력 경로가 보장되게 된다. 하지만 I/O 컨트롤러나 프로세서에 고장이 발생하게 되면 다른 쪽에서 모든 요청을 처리해야만 하는 부담이 있다.

읽기 연산은 항상 I/O를 수행할 대상이 되는 데이터가 하나 이상 존재하므로 성능 향상을 위한 부하 균등을 고려하여 가장 적절한 위치의 데이



(그림 1) 디스크 미러링의 구조

Node	Cluster 0				Cluster 1			
	0	1	2	3	4	5	6	7
Primary Copy	R0	R1	R2	R3	R4	R5	R6	R7
Backup Copy	r0.0	r0.1	r0.2		r4.0	r4.1	r4.2	
	r1.2	r1.0	r1.1		r5.2	r5.0	r5.1	
	r2.1	r2.2	r2.0		r6.1	r6.2	r6.0	
	r3.0	r3.1	r3.2		r7.0	r7.1	r7.2	

(그림 2) Interleaved-declustering

터를 선택하여 읽기 연산을 수행한다. 소프트웨어 미러링 환경에서는 데이터가 저장된 디스크가 다양한 성능을 가질 수 있으므로 이를 고려하여 읽기 연산이 이루어져야 한다. 쓰기 연산의 경우에도 같은 데이터가 하나이상 존재하므로 미러링에 참여하고 있는 모든 디스크에 수행한다.

2.2 Interleaved-declustering

Interleaved-declustering[8,9,10]이란 여러 개의 노드(프로세서)들을 모아 클러스터를 구성하고 노드의 개수만큼 데이터를 분할하여 저장한다. 그림 2에서 보는 것처럼 데이터는 N(노드의 개수)개의 단편(fragment)으로 분할된다. 그리고, 각 분할된 단편은 다시 N-1개의 백업 단편으로 나뉘고 주 단편이 저장되는 노드를 제외한 N-1개에 나뉘어 저장된다. 그림 2는 클러스터가 2개 존재하며 데이터를 8개의 단편으로 분할하고 이를 두 클러스터에 존재한 노드들에 나뉘어 저장하는 상황을 보여준다. 이 방법에서는 하나의 노드가 고장나면 N-1개의 다른 노드에 균등하게 부하가 분산될 수 있다. 하지만 두 개 이상의 노드가 고장나면 더 이상의 I/O 요구에 대한 처리가 불가능하다.

2.3 Chained-declustering

Chained-declustering 방법에서는 여러 노드를 모아 클러스터를 만들고 각 노드에 데이터를 나누어 저장한다[8,9,10]. 주 데이터를 N(노드의 개수)개의 단편으로 나누어 각 노드에 저장하고, I 번째 노드의 주 데이터 복사본을 (I+1)%N번째에 중복해서 저장한다. 그림 3은 노드가 8개일 경우 Chained-

Node	0	1	2	3	4	5	6	7
Primary Copy	R0	R1	R2	R3	R4	R5	R6	R7
Backup Copy	r7	r0	r1	r2	r3	r4	r5	r6

(그림 3) Chained-declustering

Node	0	1	2	3	4	5	6	7
Primary Copy	R0	---	$\frac{1}{7}$ R2	$\frac{2}{7}$ R3	$\frac{3}{7}$ R4	$\frac{4}{7}$ R5	$\frac{5}{7}$ R6	$\frac{6}{7}$ R7
Bockup Capy	$\frac{1}{7}$ r7	---	r1	$\frac{6}{7}$ r2	$\frac{5}{7}$ r3	$\frac{4}{7}$ r4	$\frac{3}{7}$ r5	$\frac{2}{7}$ r6

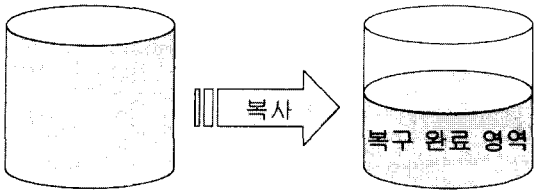
(그림 4) 노드 1 고장 시 부하분산

declustering 방법을 보여준다. Chained- declustering에서 정상 읽기 연산은 주 데이터 복사본에서 이루어지게 되고, 쓰기 연산은 디스크 미러링의 경우와 같이 원본 디스크와 백업 복사본(Backup Copy) 모두에서 이루어지게 된다.

그림 4는 클러스터의 노드들 중 하나가 고장났을 때 고장난 노드에 대한 부하를 어떻게 분산하는지 보여준다. 노드 1에 고장이 발생한 경우 노드 2에서 R1에 해당하는 데이터에 대한 연산을 수행하게 되고, 노드 2의 주 데이터 복사본 R2에 대한 연산은 1/7 만을 담당한다. 이러한 작업은 의무영역(responsible range)을 다시 계산함으로써 가능하다. 결국 한 개의 노드가 담당하는 부하는 8/7씩이 되어 노드 1의 고장으로 인해 발생하는 부하를 다른 7개의 노드들이 나누어 갖게된다. 하지만 인접한 두 개의 노드에 고장이 발생한 경우는 연산이 불가능하다.

2.4 영구적 고장에 대한 복구

기존에 디스크 미러링 시 고장이 발생하였을 경우를 위한 고장 복구 방법이 제시되었다[12]. 먼저 기존 복구 알고리즘들을 설명하기 전에 그림 5와 같은 상황을 가정한다. 두 개의 디스크에 미러링을 수행하는 도중 오른쪽의 디스크가 고장이 난 후 이를 다른 디스크로 교체하였다. 교체한 후 정상상태 디스크로 서비스를 계속 제공하면서



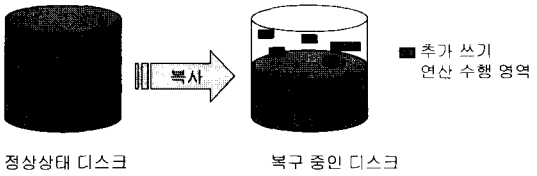
(그림 5) 디스크 교체 후 복구

정상상태 디스크의 내용을 차례로 복사하여 교체 디스크의 복구를 수행한다. 위와 같은 상황에서 첫 번째 고장 복구 방법은 다음과 같다. 모든 읽기 요청은 정상상태 디스크가 처리한다. 쓰기 요청은 복구 완료 영역에 대해서는 양쪽 모두 수행하고, 그 외의 영역에 대해서는 정상상태 디스크가 처리한다. 그리고 디스크 복구가 완료되면, 고장 발생 이전의 방법을 사용하여 요청을 처리한다. 이 방법은 간단하지만 복구하는 동안 정상상태 디스크에 대한 부하가 크다는 것이 단점이다.

이를 개선한 두 번째 방법에서는 읽기 요청에 대해서 복구 완료영역의 경우이면 두 디스크 중 부하가 적은 쪽으로 보내고, 그 외의 영역에 대한 요구이면 정상상태 디스크로 보낸다. 즉, 복구 완료 영역에 대해서는 부하균등을 수행한다. 쓰기 요청에 대해서는 첫 번째 방법과 동일하다. 이 방법에서는 복구하는 동안이라도 복구 완료된 영역에 대해서는 읽기 연산에 대한 부하균등을 수행할 수 있어 보다 높은 I/O 성능을 보장할 수 있다.

3. 영구적 디스크 고장 복구 기법

이 장에서는 제안하는 고장 복구 기법에 대해 설계한다. 앞서 디스크 미러링 시 영구적 고장이 발생하였을 경우 고장 복구 방법에 대한 기존 연구에 대해서 언급하였다. 여기서는 디스크 미러링 시 영구적 고장 복구 방법의 성능 개선을 위한 방법을 모색해보고, Chained-declustering의 경우 효율적인 고장 복구 기법에 대해서 설계한 내용을 언급한다. 제안하는 방법의 기본적인 목적은 고장이



(그림 6) 제안하는 복구 기법에서의 디스크 상태

발생하였을 경우 복구 시에도 되도록 정상상태에 근접한 디스크 I/O 성능을 낼 수 있도록 하는 것이다. 먼저 기존 디스크 미러링에 대한 복구방법보다 개선된 방법을 제안하고, Chained-declustering에 대한 복구방법에 대해서 기술한다.

3.1 디스크 미러링 시 고장 복구 기법

앞서 영구적 고장이 발생하였을 경우 디스크 미러링을 위한 기존 복구방법 중 마지막 방법은 완료된 영역에 대해서는 읽기 연산에 대해 부하균등을 허용한다. 그리고 쓰기에 대해서는 복구 완료된 영역에 한해서만 허용하고 있다. 제안하는 방법은 쓰기에 대해서 그런 제한을 두지 않고 양쪽 모두에 수행한다. 그러면 그림 6에서처럼 교체 디스크의 복구 완료된 영역 외에 다른 부분에도 데이터가 기록되게 된다. 이 경우 읽기 연산에 대해서 교체된 디스크의 복구영역뿐 아니라 쓰기 연산에 의해 기록된 데이터들에 대해서도 부하균등을 허용할 수 있다는 장점이 있다.

제안하는 복구 기법에서 정상적인 읽기 또는 쓰기 요청이 들어올 경우 수행과정은 그림 7과 같으며, 교체된 디스크의 복구 작업은 정상적인 I/O 요청이 없는 경우 수행한다. 즉, 복구 작업보다 정상적인 I/O 수행을 우선 시 한다.

그림 7에서처럼 모든 쓰기 요청은 모든 영역에 대하여 두 디스크에 수행을 한다. 이 경우 복구중의 디스크 영역은 이미 복구 완료된 영역, 쓰기가 수행된 영역, 그 이외의 영역으로 나눌 수 있다. 만약 읽기 요청이 들어온다면 그 영역이 세 영역 중 어느 영역에 속하는지 검사한 뒤 부하 균등을 이루도록 디스크에 읽기 연산을 배분한다.

```
읽기 연산의 알고리즘
( ReadB : 읽기 위한 블록,
RecoveryB :복구 완료영역 및 추가적 쓰기 연산으로 쓰여진 블록)
1. if (ReadB == RecoveryB)
2. 부하 균등을 통해 읽기 연산 수행
3. else
4. 정상상태 디스크를 통해 읽기 연산 수행

쓰기 연산의 알고리즘
1. 두 디스크의 모든 영역에 대해서 쓰기 연산 수행
```

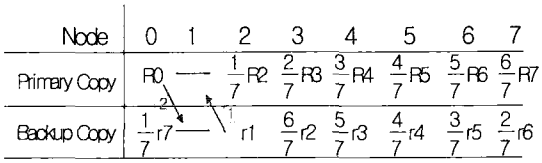
(그림 7) 제안하는 기법의 복구 알고리즘 (디스크 미러링)

이 방법의 장점은 복구 완료된 영역 외에 쓰기가 수행된 영역에 대해서도 부하 균등을 이룰 수 있다는 것이다. 하지만, 이를 위해서는 현재 어느 영역에 대해서 쓰기가 수행되었는지를 지속적으로 유지해야하는 부담이 있다. 그러나 이 부담은 얼마든지 구현에 의해서 경감할 수 있다.

또 다른 고려 사항은 쓰기 연산을 통해 쓰여진 부분에 대해서 복구 수행 중 정상 상태 디스크로부터 교체 디스크로 다시 복사를 할 것인가 하는 문제가 있다. 첫 번째는 이미 쓰기가 수행된 영역에 대해서도 역시 복구를 수행하는 방법이다. 이 경우 이미 쓰기가 수행된 영역에 대해서도 역시 복구를 한다면 불필요한 복구를 수행이라고 할 수 있다. 두 번째는 쓰기가 이미 수행된 부분에 대해서는 복구를 하지 않는 방법으로, 불필요한 복구 과정은 수행하지 않아도 되지만, 어느 블록이 쓰기가 수행되었는지를 판별해야하고 디스크 헤드를 다음 복구될 영역으로 이동시키기 위한 시간이 소요되는 문제점이 있다. 알려져 있다시피 디스크 접근 시 전체 수행 시간 중 디스크 헤드 이동시간이나 회전 지연 시간이 많은 부분을 차지한다. 두 방법 모두 장단점이 있지만, 제안하는 기법에서는 쓰기 완료 영역 역시 복구하는 기법을 채택하였다.

3.2 Chained-declustering 시 고장 복구 기법 1

Chained-declustering 방법은 복구를 수행하는 동안 고장난 노드의 부하를 여러 노드가 나누어 수행하므로 성능의 저하가 덜 하다. 하지만 이것은 Chained-declustering에 참여하는 노드들의 수가 충분히 많을 경우이다. 또한 노드의 수가 어느 정도 있다 하더라도 복구하는 동안 되도록 고장난 노드에 대한 부하 분산의 양을 줄일 수 있다면 보다 높은 디스크 I/O 성능을 보일 것이다.



(그림 8) Chained-declustering을 위한 복구 기법 1

```
Chained-declustering 복구 알고리즘(방법1)

1. 고장난 1번 노드에 노드 2의 r1을 복사해서 R1에 대한 복구를 수행
2. 복구가 완료되면 각 노드에 분산했던 부하를 다시 정상상태로 환원
3. 노드 1의 r0에 대한 복구를 수행
```

(그림 9) Chained-declustering을 위한 복구 기법 1의 알고리즘

이 논문에서는 Chained-declustering에 대한 복구 방법 두 가지를 제안한다. 그 중 첫 번째 방법을 이용하여 복구가 수행되는 상황을 그림 8에서 보여준다. 먼저 고장난 1번 노드에 노드 2의 r1을 복사해서 R1에 대한 복구를 수행한다. 이에 대한 복구가 모두 완료되면 각 노드에 분산했던 부하를 다시 정상상태와 같이 한다. 이 상태에서 다시 노드 1의 r0에 대한 복구를 수행한다. 이때 이미 복구가 끝난 영역에 대한 쓰기 요구는 주사본과 백업사본 모두에 이루어진다. 첫 번째 방법은 간단한 반면 복구된 영역에 대해서 전혀 부하균등을 수행하지 않는다.

Chained-declustering을 위한 첫 번째 복구 기법의 알고리즘은 그림 9와 같다.

3.3 Chained-declustering 시 고장 복구 기법 2

두 번째 방법은 고장난 노드 1의 R1을 복구하

각 노드의 부하 결감치

$\frac{1}{4} \times \frac{1}{7} = \frac{1}{28}$

Node	0	1	2	3	4	5	6	7
Primary Copy	R0	$\frac{7}{28}$ R1	$\frac{10}{28}$ R2	$\frac{13}{28}$ R3	$\frac{16}{28}$ R4	$\frac{19}{28}$ R5	$\frac{22}{28}$ R6	$\frac{25}{28}$ R7
Backup Copy	$\frac{3}{28}$ r7	—	$\frac{21}{28}$ r1	$\frac{18}{28}$ r2	$\frac{15}{28}$ r3	$\frac{12}{28}$ r4	$\frac{9}{28}$ r5	$\frac{6}{28}$ r6

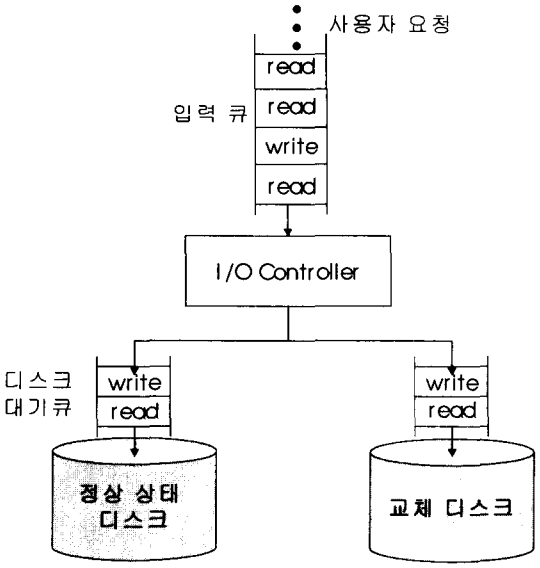
r10| ‡ 복구된 경우의 전체 의무 영역 재계산

(그림 10) Chained-declustering을 위한 복구 기법 2

Chained-declustering 복구 알고리즘(방법2)

1. 노드 2에 있는 r1중 1/4을 노드 1에 복사
2. r1의 복구가 1/4 만큼 수행되면 의무 영역 재계산
3. 노드 1에서도 복구 영역에 대해서는 서비스 재계
4. 위 1~3을 반복 수행하여 노드 1의 R1이 복구 완료되면 정상 서비스로 환원
5. 노드0의 R0를 노드1에 복사

(그림 11) Chained-declustering을 위한 복구 기법 2의 알고리즘



(그림 12) 디스크 미러링의 정상상태 시뮬레이션 모델

는 도중 주기적으로 클러스터의 각 노드에 할당된 의무 영역을 재 계산한다. 그림 10이 두 번째 방법을 이용해서 복구하는 모습을 보여준다.

의무 영역을 ¼ 만큼 복구가 될 때마다 재계산한다고 가정한다면, 그림에서 보듯이 노드 2에 있는 r1을 노드 1에 ¼ 만큼 복사했을 때 그림 10처럼 다른 노드의 의무영역을 재 계산하게 된다. 전체 7개의 노드가 있고 ¼만큼이 복구가 되었기 때문에 다른 7개의 노드는 각각 부하가 1/28 만큼씩 경감된다. 이 방법은 의무영역을 복구 과정 중에 재계산하여 보다 높은 부하균등을 제공할 수 있다. 하지만 주기적으로 의무영역을 재계산해야하는 부담과 얼마만큼의 주기로 재계산하는 것이 좋을 지 결정해야 한다.

Chained-declustering을 위한 첫 번째 복구 기법의 알고리즘은 그림 11과 같다. 그림 11은 재계산 주기를 1/4이 복구되었을 경우로 했을 때 Chained-declsutering 알고리즘을 나타낸다.

4 시뮬레이션 환경 및 성능 평가

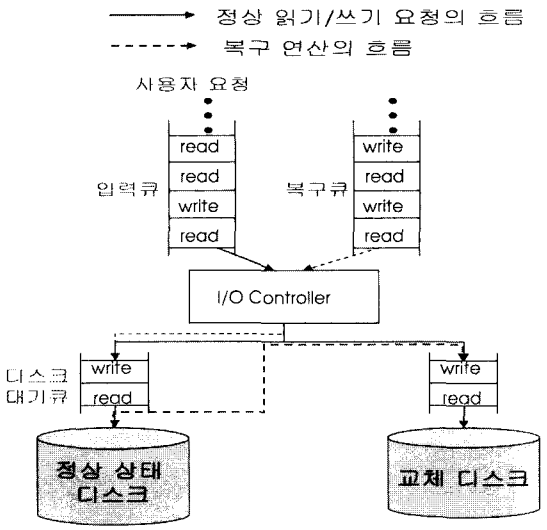
지금까지 디스크 미러링과 Chained-declustering을 위한 복구 방법에 대해서 자세히 기술하였다. 지금부터는 제안하는 방법들에 대하여 성능 평가를 수행하기 위한 시뮬레이션 환경 및 모델, 파라미터를 설명한다. 시뮬레이션에서는 영구적인 고장이 발생했을 경우 디스크 미러링을 위해 제안한 복구 방법과 기존의 디스크 미러링에 대한 복구 방법 2가지를 시뮬레이션 수행을 통해 비교 평가한다. 그리고 Chained-declustering을 위해 제안한 두 가지 방법을 시뮬레이션 하여 비교 평가한다.

4.1 시뮬레이션 환경

시뮬레이션에 사용된 컴퓨터는 펜티엄-III 1GHz에 256Mbytes의 주기억장치를 갖는다. 구현은 AweSim [13] 3.0 student version을 이용하였다. 디스크 미러링과 Chained-declustering에 참여하는 디스크들의 성능은 같다고 가정하였다. 시뮬레이션 시 디스크 성능 수치는 Quantum Atlas 10KIII [14]을 참조하였다.

4.1.1 시뮬레이션 모델

그림 12와 그림 13은 디스크 미러링의 시뮬레이션 모델을 나타낸다. 그림 12는 정상상태의 디



(그림 13) 디스크 미러링의 복구상태 시뮬레이션 모델

스크 미러링에 대한 시뮬레이션 모델이다. 그림에서 디스크의 수는 2개이지만 이것은 확장 가능하다. 주 디스크(Primary Disk)와 백업 디스크(Backup Disk) 각각에 읽기/쓰기 요구가 대기하는 큐가 존재한다. 그리고 I/O 컨트롤러에도 전체적인 디스크 읽기/쓰기 요구가 대기하는 큐가 존재한다. 최초에 사용자가 I/O 요구를 디스크 미러링 시스템에 보내면 I/O 컨트롤러 앞의 큐에서 일단 대기하게 된다. I/O 컨트롤러는 이들 I/O 요구들을 적절한 디스크로 보내게 된다. 이때 앞에서 설명했듯이 읽기 요구는 부하 균등을 이루기 위해 각 디스크에 할당된 큐에 대기하는 I/O요구의 수가 적은 쪽으로 보내지며, 쓰기 요구는 일관성을 유지하기 위해 양쪽 디스크 모두로 보내진다.

복구상태의 시뮬레이션 모델은 그림 13과 같다. 복구 상태 모델에서는 I/O 컨트롤러 앞에 두 개의 큐가 존재한다. 하나는 정상적인 I/O 요구들이 대기하는 큐이고, 다른 하나는 복구 I/O 요구들이 대기하는 큐이다. I/O 컨트롤러는 정상 I/O 요구들에 대해서 지장을 덜 주도록 하며, 두 큐로부터 나오는 I/O 요구를 적절하게 디스크에 분배한다. 어느 디스크에 전달하는지는 각각의 복구 알고리즘에 달려 있다. Chained-declustering에 대한 시뮬레이션 모델도 디스크 미러링과 유사하며, 클러스터에 참여하는 노드의 수에 따라서 디스크의 수를 확장하면 된다.

(표 1) 시뮬레이션 파라미터들

	파라미터	값
시스템 파라미터	I/O 시간	Average seek time : 4.5 ms Average latency time : 3.0ms Transfer rate 1. internal : 480 Mb/sec 2. buffer host : 320 Mb/sec
	블록의 크기	4 Kbytes
	전체데이터 크기	120 Mbytes
성능 파라미터	I/O 요구의 도착율	정상 요청:지수 분포 ([7,15]ms), 균일 분포 ([5,20]ms) 복구 요청 : 균일 분포 (50ms)
	디스크의 수	2 ~ 4
	쓰기/읽기의 비율	0 : 1 ~ 1 : 0

4.1.2 시뮬레이션 파라미터

실험에 사용된 시뮬레이션 파라미터들이 표 1에 있으며, 시스템 파라미터와 성능 파라미터로 나눠볼 수 있다. 먼저 시스템 파라미터에서 I/O시간은 Quantum Atlas 10KIII 모델을 참조하였다. I/O 시간은 디스크 읽기와 쓰기 시간을 동일하게 8.75 ms로 하였다. 또한 디스크 I/O 기본 블록 크기를 4 KByte로 하였으며, 전체 데이터의 크기를 120 MByte로 하였다. 실험을 위해 사용된 성능 파라미터로 3가지가 있다. 먼저 I/O 요구의 도착율은 정상 요청의 경우 지수 분포를 사용하였으며, 복구 요청의 경우 디스크 고장 후 복구 시 일정 시간마다 발생하므로 균일 요청을 사용하였다. 또한 정상 요청의 경우 역시 균일 분포에 대해서도 실험을 하였다. 실험에 사용된 디스크의 수는 디스크 미러링의 경우 2개, Chained- declustering의 경우 4개를 사용하였다. 마지막으로 읽기와 쓰기의 비율을 다양하게 변경시키면서 실험을 하였다.

정상 요청의 경우 지수 분포로 실험을 하였는

데, 지수 분포는 지속적인 현상을 설명하는데 흔히 쓰이는 확률 분포이며, 제안하는 기법을 시뮬레이션 하기 위한 환경에서도 적합한 분포라고 할 수 있다. 시뮬레이션에서는 식 1과 같은 지수 분포 확률 밀도함수를 사용하여 정상 I/O 요청 도착 간격이 평균 7~15 ms로 발생하도록 하였다. 정상 I/O 요청 도착 간격은 시스템의 부하를 변경시키는 요소가 되며, 도착 간격이 작을수록 시스템의 부하량이 많아지게 된다.

$$f(x) = \lambda e^{-\lambda x}, x = [0, \infty) \quad \text{(식 1)}$$

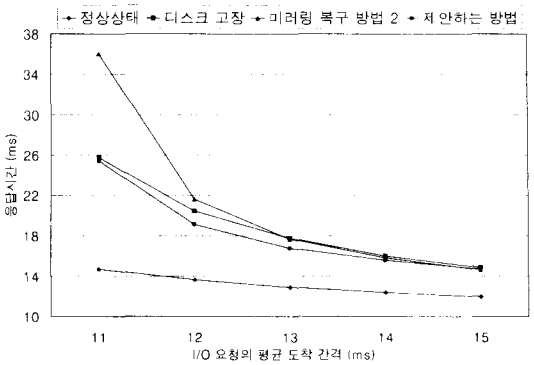
이 외에도 전체 데이터에 대한 접근형태는 균일 분포를 가정하였다. 출력 값은 정상상태에서 디스크 I/O의 평균응답시간과 복구중의 정상 I/O 요구에 대한 디스크 I/O의 평균응답시간을 측정하였고, 이 둘을 성능평가의 척도로 하였다.

4.2 성능 평가

AweSim 프로그램을 이용하여 제안하는 방법과 기존 방법을 시뮬레이션 한 결과를 설명한다. 먼저 디스크 미러링 시 고장 복구 성능 평가 결과를 설명하고, Chained-declustering시 제안한 두 가지 방법에 대해서 비교 평가하도록 하였다.

4.2.1 디스크 미러링

디스크 미러링 시 복구 기법에 따른 시뮬레이션은 정상 요청의 발생 간격에 따른 정상 요청의 응답 시간, 읽기와 쓰기 비율에 따른 응답시간, 정상 요청이 균일 분포일 경우 응답 시간을 측정하였다. 시뮬레이션 시 비교 대상은 정상 상태, 고장이 발생하였을 때, 기존 복구 방법 중 두 번째 방법으로 선정되었다. 기존 방법 중 첫 번째 방법은 복구 중 부하 균등을 수행하지 않기 때문에 다른 방법에 비해서 성능이 떨어지므로 성능이 더 나은 두 번째 방법에 대해서만 비교를 수행하였다.

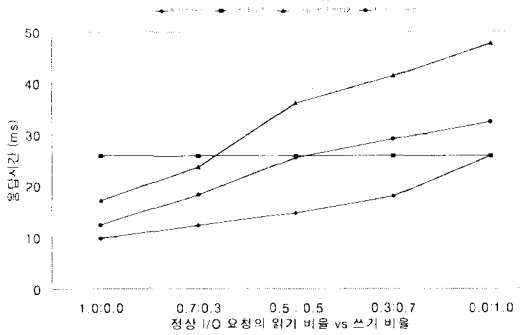


(그림 14) 정상 요청의 I/O 도착 간격에 따른 응답시간(디스크 미러링)

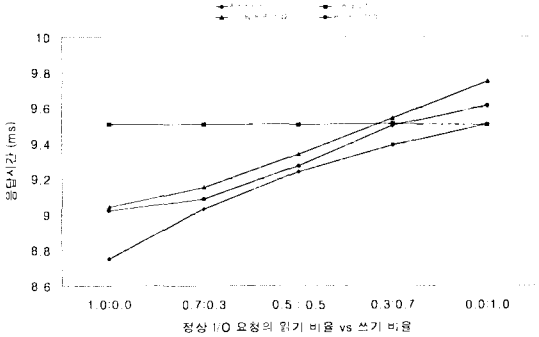
먼저 디스크 복구 시 시스템의 부하량에 따른 정상 요청의 응답 시간을 측정하였다. 즉, 지수 분포를 따르는 정상 요청의 도착 간격을 달리하여 시뮬레이션 하였으며, 평균 도착 간격을 11~15ms로 변경시키면서 시뮬레이션 하였다. 복구 요청의 경우 균일 분포로 50ms마다 주기적으로 발생하도록 하였다. 디스크 I/O의 읽기와 쓰기 비율은 1 : 1로 고정하였다. 시뮬레이션 수행 결과가 그림 14에 나타나있다.

그림 14 에서 보듯이 시뮬레이션을 통해 정상상태와 고장이 발생했을 때의 평균응답시간 차이는 5.84ms 이다. 기존 방법 중 두 번째 방법의 응답시간과 정상상태와의 차이는 8.73ms인 반면 제안하는 방법은 평균 5.1ms의 차이가 발생하였다. 제안하는 복구방법이 기존 방법에 비해서 3.63ms가 빨라졌으며, 상대적으로 약 40%의 성능개선이 있음을 알 수 있다. 이는 제안하는 방법이 쓰기 요청 시 두 디스크 모두에 수행되므로, 읽기 요청 수행 시 부하 균등을 통해 성능 향상을 가져온 것이다.

그림 15는 지수 분포일 때 읽기와 쓰기 비율에 따른 정상 요청의 응답 시간을 측정한 것으로, 평균 I/O 도착 간격은 11ms로 하였다. 읽기와 쓰기의 비율을 조정하면서 측정한 결과 그림에서 비율이 0.5 : 0.5 이전에는 제안하는 방법과 기존 방법이 디스크 고장 발생 시 보다 더 빠른 응답시간을 나타내었으며, 쓰기 비율이 높아질수록 디



(그림 15) 정상 요청의 읽기와 쓰기 비율에 따른 응답시간(지수 분포)



(그림 16) 정상 요청의 읽기와 쓰기 비율에 따른 응답시간(균일 분포)

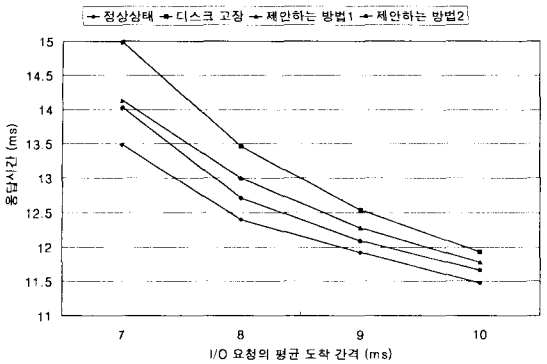
스크 고장 시보다 응답시간이 오래 걸렸다. 이것은 복구 수행으로 인하여 정상 I/O 요청이 지연되며, 쓰기 요청 역시 두 디스크에 모두 반영해야 하는 부담이 생기기 때문에 복구 중에 고장 시보다 좋지 않은 성능을 보였다. 하지만 읽기 비율이 높은 경우에는 부하 균등을 이루기 때문에 더 좋은 성능을 나타내었다. 정상 상태 시에는 쓰기와 읽기를 위해 수행되는 시간이 같다고 가정하였기 때문에 항상 약 25ms의 응답시간을 보였다. 이번 실험 역시 제안하는 방법이 기존의 디스크 복구 방법 2보다 우수하였다. 기존 방법 2의 경우 정상 상태와 평균 17ms의 차이를 보였으며, 제안하는 방법의 경우 약 7.5ms의 차이를 보였다. 즉, 제안하는 방법이 기존 방법 2에 비해 상대적으로 약 57% 정도의 성능 향상을 나타내었다.

그림 16은 입력 데이터의 분포가 5~20 ms 사이의 균일 분포를 따르는 경우에 대한 실험 결과이다. 비교 대상은 이전 실험과 동일하다. 실험 결과 지수 분포와 비슷한 형태를 나타내었는데 기존 방법의 경우 정상 상태와 평균 0.18ms 차이를 나타내었으며, 제안하는 방법의 경우 약 0.11ms의 차이를 나타내었다. 따라서 제안하는 방법이 기존 방법보다 평균 약 40% 우수함을 보였다. 균일 분포 역시 읽기 비율이 높을수록 제안하는 방법과 기존 방법이 디스크 고장 시보다 우수한 성능을 나타내었다. 하지만 쓰기 비율이 높아질수록 응답시간이 오래 걸렸다. 이것은 지수 분포처럼 복구와 정상 쓰기 요청의 추가 비용으로 인한 것이다.

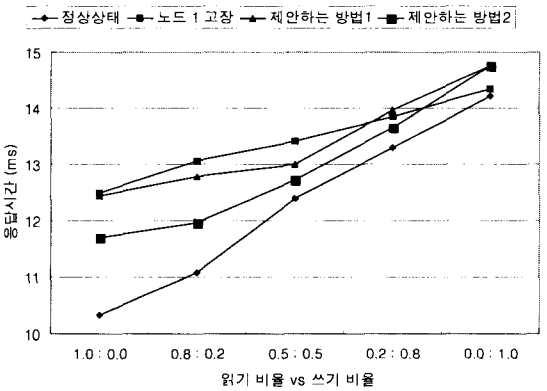
4.2.2 Chained-deculstering

Chained-declustering의 경우 디스크 고장 시에도 가용성을 높이기 위해서 제안된 방법으로 고장난 노드의 부하를 다른 노드에서 담당하여 처리한다. 따라서 고장 시에도 정상상태와 큰 차이를 보이지 않지만, 복구 중에 보다 나은 응답 시간 성능을 제공하기 위하여 본 논문에서 복구 알고리즘을 제안하였다. 그리고 제안하는 복구 알고리즘의 성능을 시뮬레이션을 통해 입증한다. 시뮬레이션에서는 정상상태에서의 디스크 I/O의 평균응답시간, 고장이 발생한 후의 평균응답시간, 그리고 제안하는 두 가지 방법을 이용해서 복구를 수행할 때의 I/O 평균응답시간을 측정하였다. 제안하는 방법 중 두 번째 방법에서 복구 중 의무 영역을 재계산하게 되는데, 의무 영역의 재계산 비용은 크지 않으므로 시뮬레이션 시에 무시하였다.

먼저 정상 I/O 도착 간격에 따른 I/O 응답시간을 측정한 결과가 그림 17과 같다. 디스크 I/O 요구의 도착 분포는 지수 분포를 따르며, 평균 도착 간격을 7~10ms로 변경시키면서 시뮬레이션 하였다. 복구 요청의 경우 균일 분포로 50ms마다 수행되도록 하였다. 실험 결과 고장이 발생했을 때는 응답시간이 약 0.9ms 정도 증가하였고, 첫 번째 방법을 이용해서 복구할 때는 0.47ms, 두 번째



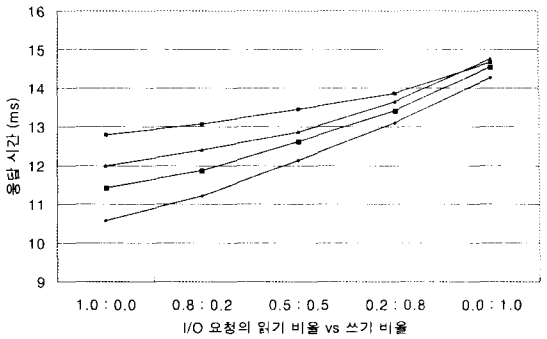
(그림 17) 정상 요청의 I/O 도착 간격에 따른 응답시간(Chained-declustering)



(그림 18) 정상 요청의 읽기와 쓰기 비율에 따른 응답시간(지수 분포)

방법에서는 0.3ms가 증가하였다. 두 번째 방법이 상대적으로 약 36%정도 성능 향상이 있었다. 이는 두 번째 방법이 디스크 복구 중에 주기적으로 의무 영역을 재계산하여 디스크 I/O요청에 대한 부하 균등을 할 수 있었기 때문이다.

그림 18은 Chained-declustering 시 디스크 입력과 출력 비율에 따른 응답시간을 측정한 결과이다. 정상 I/O 요청은 지수 분포를 따르며, 평균 도착 간격을 8ms로 고정하였다. 그림에서 보듯이 응답 시간 측면에서 제안하는 방법 1은 정상 상태와 약 1.1ms의 차이를 보였으며, 제안하는 방법 2는 0.7ms의 차이를 보였다. 따라서 제안하는 기법 중 방법 2가 방법 1보다 약 37% 정도 향상된 성능을 나타내었다. 또한 읽기의 비율이 높아지면



(그림 19) 정상 요청의 읽기와 쓰기 비율에 따른 응답시간(균일 분포)

복구 중에 의무 영역 재계산을 하는 방법 2가 더 우수함을 알 수 있었다. 모든 디스크 연산이 쓰기 요청일 경우에는 제안하는 방법들이 노드 고장 시보다 좋지 않은 결과를 나타내고 있는데, 이는 복구 연산과 두 디스크에 모두 쓰기를 반영해야 하는 부담이 생기기 때문이다.

마지막으로 Chained-declustering 시 정상 I/O 요청이 5~20ms 사이의 균일 분포로 발생하는 경우 읽기와 쓰기 요청 비율에 따른 응답시간을 측정한 결과가 그림 19이다. 시뮬레이션 결과 균일 분포로 디스크 I/O가 발생하는 경우 역시 지수 분포일때와 비슷한 결과를 나타내었다. 제안하는 방법 중 첫 번째 방법은 정상 상태에 비해서 0.87ms의 차이를 보였으며, 두 번째 방법은 약 0.52ms의 차이를 나타내었다. 따라서 두 번째 방법이 첫 번째에 비해 약 40%정도 향상된 것을 알 수 있다.

5. 결 론

현대 저장 시스템 환경에서는 데이터의 가용성과 신뢰성을 높이기 위한 방법으로 디스크 미러링이나 Chained-declustering 방법을 사용한다. 이 경우 디스크 미러링이나 Chained-deculstering에 참여하는 디스크에 고장이 발생할 수 있다. 만약 디스크에 고장이 발생하면, 복구를 수행하면서 얼마나 정상 상태와 유사한 성능으로 정상 I/O요구를 처리해주는 지의 여부가 중요하다. 이를 위해 본

논문에서는 디스크에 영구적 고장이 발생할 경우 이를 효과적으로 복구할 수 있는 방법들을 제안하였다. 또한 제안한 방법들과 기존방법들을 시뮬레이션을 통해 비교 평가하였다.

디스크 미러링이나 Chained-declustering시 발생할 수 있는 고장은 일시적인 고장과 영구적인 고장으로 분류해 볼 수 있다. 일시적 고장은 일시적으로 디스크를 사용하지 못할 경우로 두 디스크간의 일관성이 깨지는 경우를 말하며, 디스크의 일부를 영구적으로 접근하지 못하는 경우를 영구적 고장이라고 말한다. 이 경우 디스크의 교체가 필요하며, 고장 발생 시 정상상태에 비슷한 I/O 성능을 보장하면서 복구하는 방법에 대하여 기술하였다. 그리고 영구적인 고장 발생 시 제안하는 방법과 기존의 방법을 모두 시뮬레이션을 통해 비교평가 하였다. 시뮬레이션은 정상 요청을 지수 분포와 균일 분포로 하였으며, 정상 요청의 읽기와 쓰기 비율을 변경시키면서 수행하였다. 디스크 미러링 시 정상 요청을 지수 분포로 하고 읽기와 쓰기 비율을 1:1로 하였을 경우 제안하는 방법이 기존의 방법보다 약 40% 정도 나은 성능을 보였으며, 읽기와 쓰기 비율에 따른 시뮬레이션과 정상 I/O 요청이 균일 분포를 따를 때의 시뮬레이션 역시 각각 57%와 40%의 성능 향상을 보였다. Chained-declustering의 경우 두 가지 복구방법을 제안하였는데, 두 번째 방법이 첫 번째 방법보다 우수함을 알 수 있었다. 먼저 디스크 I/O 요청이 지수 분포를 따르는 경우 약 36% 정도의 성능 향상을 가져왔으며, 읽기와 쓰기 요청 비율에 따른 응답 시간측면에서도 약 37%의 성능 향상을 보였다.

참 고 문 헌

[1] 김정환, 강회일, 이동일, "SAN 기술 및 시장 동향," 전자통신동향분석, pp. 24~37, 2000.2.

[2] Barry Phillips, "Have Storage Area Networks Come of Age?," IEEE Computer volume 31 7, pp. 10~12, 1998.

[3] Thomas Ruwart, "Disk Subsystem Performance Evaluation: From Disk Drives to Storage Area Networks," IEEE Symposium on Mass Storage Systems, pp. 1~24, 2000.

[4] Molero, X., Silla, F., Santonja, V. and Duato, J., "Parallel and Distributed Systems," ICPADS, pp. 484~491, 2001.

[5] David A. Patterson, Garth A. Gibson and Randy H. Katz, "A Case for Redundant Arrays of Inexpensive Disks (RAID)," In Proceeding of SIGMOD Conference, pp. 109~116, 1988.

[6] Jai Menon, Jeff Riegel and James C. Wyllie, "Algorithms for Software and Low-Cost Hardware RAIDs," COMPCON, pp. 411~418, 1995.

[7] Edward K. Lee and Randy H. Katz, "The Performance of Parity Placements in Disk Arrays," IEEE Transactions on Computers, volume 42 6, pp. 651~664, 1993.

[8] Hui-I. Hsiao and D. J. DeWitt, "A performance study of three high availability data replication strategies," In Proceedings of ICPDIS, pp. 18~28, 1991.

[9] L. Golubchik, J. Lui and R. R. Muntz, "Chained declustering: load balancing and robustness to skew and failures," In Proceedings of Second International Workshop on Transaction and Query Processing, pp. 88~95, 1992.

[10] Hui-I Hsiao and D. J. DeWitt, "Chained Declustering: A New Availability Strategy for Multiprocessor Database Machines," In proceeding of ICDE, pp. 456~465, 1990.

[11] 김진표, 김종현, "디스크 디클러스터링을 이용한 입출력 시스템의 성능 분석," 정보과학회 논문지, pp. 203~213, 1996.2.

[12] H. H. Kari, H. K. Saikkonen, N. Park, and F. Lombardi, "Analysis of repair algorithms for mirrored-disk systems," IEEE Transactions on Reliability, Volume 46 2, pp. 193~200, 1997.

[13] Jean J. O'Reilly and William R Lilegdon, "Introduction to AWESIM," In Proceedings of the 1999 Winter Simulation Conference, pp. 196~200. 1999.
[14] http://www.quantum.com

◐ 저 자 소 개 ◑

피 준 일

1999년 충북대학교 컴퓨터공학과(공학사)
2001년 충북대학교 정보통신공학과(공학석사)
2001년~현재 : 충북대학교 정보통신공학과 박사과정
관심분야 : 고차원 색인 구조, 데이터 베이스 시스템, 메모리 상주형 데이터 베이스 시스템, 저장 시스템, 실시간 시스템, etc.
E-mail : pji@netdb.chungbuk.ac.kr

홍 현 택

2001년 충북대학교 정보통신공학과 (공학사)
2001년~현재 : 충북대학교 정보통신공학과 석사과정
관심분야 : 데이터 베이스 시스템, 정보 검색, 자료저장 시스템, XML, etc.
E-mail : hongry@netdb.chungbuk.ac.kr

송 석 일

1998년 충북대학교 정보통신공학과(공학사)
2000년 충북대학교 정보통신공학과(공학석사)
2000년~현재 : 충북대학교 정보통신공학과 박사과정
관심분야 : 데이터베이스 시스템, 트랜잭션, 저장 시스템, 멀티미디어 정보검색, XML, 정보 검색 프로토콜, etc.
E-mail : prince@netdb.chungbuk.ac.kr

유 재 수

1989년 전북대학교 컴퓨터공학과 졸업(공학사)
1991년 한국과학기술원 전산학과 졸업(공학석사)
1995년 한국과학기술원 전산학과 졸업(공학박사)
1995년~1996.8 목포대학교 전산통계학과 전임강사
1996.8~현재 : 충북대학교 전기전자 및 컴퓨터 공학부 부교수
관심분야 : 데이터베이스 시스템, 정보검색, 멀티미디어 데이터베이스, 분산 객체 컴퓨팅, etc.
E-mail : yjs@cbucc.chungbuk.ac.kr