스트라이핑 시스템에서 동적 디스크 수를 지원하는 계산에 의한 매핑 방법

박유현○*, 김창수*, 김영호*, 강동재*, 신범주*
한국전자통신연구원 컴퓨터·소프트웨어연구소*

The method for adding disk with supporting dynamic number of disks in striping system

BAK,Yuhyeon○* KIM,Changsoo* KIM, Youngho* KANG, Dongjae* SHIN,Bumjoo*
ETRI Computer & Software Laboratory

요약

RAID 레벨 중에서 스트라이핑을 하는 시스템에서는 운영중에 디스크를 추가하려면, 기존의 디스크에 저장되어 있는 데이터들은 새로 추가된 디스크를 고려하여 스트라이핑 하기 위해 재구성하는 과정이 필요하다. 이러한 동적 디스크 수의 변화에 유연하게 적응할 수 있도록 매핑 테이블을 사용하는 방법들이 제안되고 있으나 관리해야 할 데이터의 양이 디스크 용량에 비례하여 증가하고 메인 메모리에서 모두 관리할 수 없기 때문에 성능이 떨어지는 문제가 발생한다. 이 논문에서는 스트라이핑을 하는 RAID 시스템에서 적은 양의 메타데이터(SZIT)를 관리함으로써 동적 디스크 수를 지원하는 계산에 의한 매핑 방법을 제안한다.

1. 서론

RAID[1]는 시스템의 입출력 성능 개선 및 신뢰성을 위해 제안된 개념으로, 응용 특성에 따라 여러 단계를 제공한다. 그 중에서 0, 4, 5 단계는 디스크 배열을 구성하는 장치들에 데이터를 분산하여 저장하는 방법이다. 이러한 단계들은 기본적으로 스트라이핑을 하는데, 스트라이핑을 하게 되면 디스크에 대한 I/O를 동시에 여러 디스크에 분산시켜 성능을 높일 수 있다.

RAID 0, 4, 5와 같이 스트라이핑을 하는 시스템에서 데이터를 디스크에 저장할 때는 디스크 수로 모듈러 연산을 수행하여 저장할 위치 디스크를 결정하는 것이 보편적인데, 사용하고 있는 도중에 새로운 디스크를 추가하고자 할 때는 디스크의 수가 변경되기 때문에, 저장되어 있는 데이터를 전체 디스크를 고려하여 균등하게 분배하는 재배치를 수행하여야 한다. 대부분의 경우 재배치를 할 경우에는 데이터를 백업하고 데이터를 새로 추가된 디스크를 고려하여 다시 분배하기 때문에 이러한 연산을 수행하고 있는 동안에는 다른 디스크 입출력 서비스를 제공하지 않는 경우가 많다. 이러한 문제를 해결하기 위해 논리주소에서 물리주소로의 매핑을 수식에 의해 수행하지 않고, 매핑정보를 테이블로 유지하여 동적인 디스크 수에 관계없는 유연성을 가지는 시스템들이 제안되고 있다.

하지만, 매핑 테이블을 사용하는 방법은 모든 디스크 I/O 때마다 테이블 등의 메타데이터를 참조해야 하기 때문에 전체적인 성능이 떨어지는 단점이 있다.

이 논문에서는 스트라이핑 방법으로 데이터를 저장하는 시스템에서 디스크를 추가할 때 메인 메모리에 모두 올려놓을 수 있을 정도의 적은 메타데이터(SZIT : Striping Zone Information Table)를 통하여 동적인 디스크 수를 지원하는 계산에 의한 매핑 방법을 제안한다.

2. 관련연구

2.1 스트라이핑 시스템에서 논리주소와 물리주소의 매핑 방법

스트라이핑 시스템에서 논리적 주소와 물리적 주소의 매핑은 일반적으로 계산식을 통해 이루어진다[2]. 즉, 모듈러 연산과 같은 수식을 통해 논리주소를 물리주소로 변환하는데 이러한 방법은 디스크 수의 동적인 변화가 발생하면 수식 자체가 바뀌게 되어 기존 데이터의 이동이 필요하다.

이러한 문제점을 해결하기 위해서 논리주소와 그에 대응하는 물리주소를 [표 1] 같이 매핑 테이블에 저장하여 매핑하는 방법[3]이 있으나 정보를 저장하기 위한 매핑 테이블의 크기가 매우 큰 단점을 가진다.

[표 1] 매핑 테이블의 구조

논리주소	디스크 번호	물리주소
0K	1	0K
1K	2	0K
2K	3	0K
3K	1	1K
4K	2	1K
5K	3	1K
:	:	:

2.2 스트라이핑 시스템에서 디스크 추가 방법

일반적으로 스트라이핑 시스템에서 디스크를 추가하는 방법으로는 열추가 방법과 행추가 방법이 있다[5]. 열추가(add column) 방법은 기존의 시스템을 구성하는 디스크의 수를 고려하지 않고 디스크를 추가하기 때문에 이미 시스템에 들어있는 데이터들을 새로 재구성 해야 한다. 행추가(add row) 방법은 이미 구성하고 있는 디스크 수를 고려하여 그 배수만큼의 디스크를 논리적으로 연결하는 방법이다.

3. SZIT를 이용한 매핑 방법

3.1 SZIT(Striping Zone Information Table)

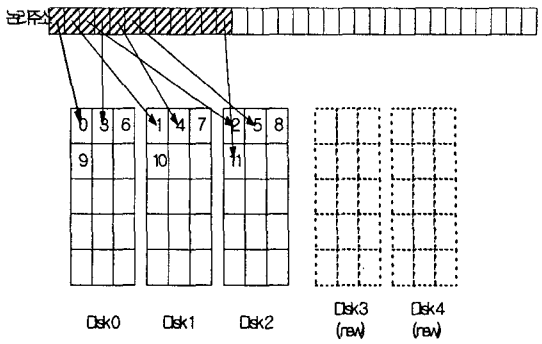
이 논문에서 제안하는 SZIT를 이용한 매핑 방법은 디스크가 추가되는 시점 이후로는 추가된 디스크들에만 스트라이핑으로 쓰기 연산을 수행하여 기존 디스크들의 데이터 보유량까지 진행한다. 모든 디스크의 데이터 보유량이 동일해 지면 그 이후로는 모든 디스크를 대상으로 스트라이핑을 수행한다. SZIT의 구성은 다음과 같다.

[표 2] SZIT의 구성

zone	스트라이핑 지역 번호
tot_disk	전체 디스크 수
par_disk	참여 디스크 수
s_phy	첫 물리 블록 번호
e_pyh	끝 물리 블록 번호
s_log	첫 논리 블록 번호
e_log	끝 논리 블록 번호

3.2 디스크 수 변화에 따른 SZIT

3개의 디스크로 스트라이핑하는 과정은 [그림 1]에서와 같이 되고, 이 시점에서 SZIT는 [표 3]과 같이 된다.

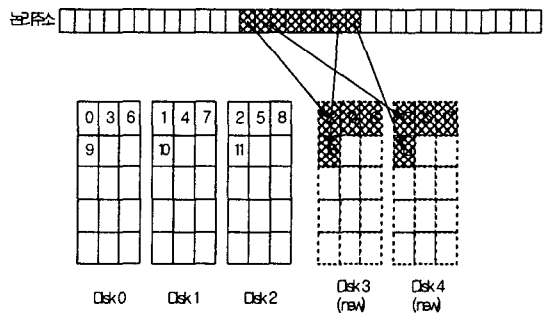


[그림 1] 단계 1

[표 3] 저장매체 추가 이후의 SZIT

zone	tot_disk	par_disk	s_phy	e_phy	s_log	e_log
0	3	3	0	14	0	44

새로운 디스크가 추가되면 SZIT는 [표 4]와 같이 변경되며 이후로는 [그림 2]와 같이 매핑된다.

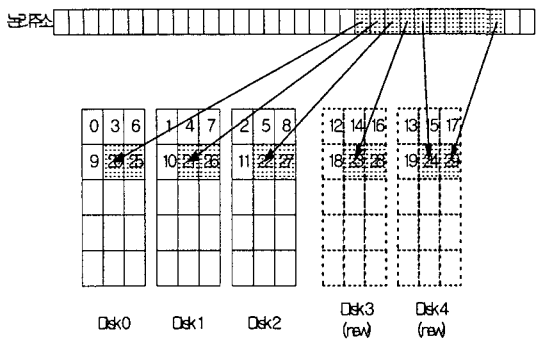


[그림 2] 단계 2

[표 4] 저장매체 추가 이후의 SZIT

zone	tot_disk	par_disk	s_phy	e_phy	s_log	e_log
0	3	3	0	3	0	11
1	5	2	0	3	12	19
2	5	5	4	14	20	74

Zone 1의 모든 영역에 대한 매핑이 끝나면 [그림 3]와 같이 Zone 2 영역에 대해 모든 디스크를 대상으로 스트라이핑이 수행된다.



[그림 3] 단계 3

3.3 SZIT를 이용한 매핑

[표 5] SZIT를 이용한 매핑 계산식

targetDevice = ((logBlkNo) % parDevice) + (totalDevice-parDevice) targetPhyBlk = (logBlkNo-logStartNo) / parDevice + phyStartNo	
targetDevice	논리블록 n을 저장하는 디스크
targetPhyBlk	논리블록 n에 매핑 되는 디스크 내의 물리논리 블록 번호
logBlkNo	논리블록 n의 주소
phyStartNo	logBlkNo의 물리블록 번호
parDevice	참여하고 있는 디스크의 수
totalDevice	전체 디스크 수

3.4 SZIT의 저장

SZIT를 메모리에서만 관리하게 되면 시스템의 전원이 나간 후 다시 시작하면 이러한 정보를 잃게 되므로, 이 정보는 디스크에 직접 저장할 필요가 있다. 따라서 SZIT의 변경이 일어날 경우에는 메모리 상의 정보를 수정한 후, 각 디스크의 레이블 정보를 저장하는 지역에 정보를 중복해서 기록한다. 또한 디스크 추가가 발견되면 SZIT를 수정하고 즉시 디스크에 반영시킨다. 시스템의 재동작시에는 디스크의 첫 부분에 저장된 스트라이핑 지역 정보를 읽어서 전체적인 시스템의 스트라이핑 지역 정보를 만들게 된다.

5. 매핑 방법 비교

이 장에서는 다른 매핑방법과 SZIT를 이용한 매핑방법을 비교한다.

5.1 매핑 방법들간의 성능 비교

계산에 의한 매핑은 논리블록을 물리블록으로 변환하는데 있어서 수식을 사용하기 때문에 따로 저장해야할 매핑 데이터는 전혀 없다. 이에 반해서 매핑 테이블에 의한 방법은 논리블록을 물리블록으로 변환하기 위한 테이블이 존재하는데, 이 테이블은 (논리블록의 수 × 테이블에서의 한 튜플의 크기) 가 된다. 디스크의 크기가 클수록 이 수는 증가한다. 이 논문에서 제안하는 SZIT를 통한 매핑에서도 매핑 테이블처럼 추가의 매핑 정보를 관리하는데, 이 정보의 크기는 (디스크 추가 회수 × 테이블에서의 한 튜플의 크기) 가 되므로 매핑 테이블에서 관리하는 매핑 데이터에 비해서 현저히 적은 데이터를 관리하게 된다. 이에 대한 자세한 비교는 5.2에서 살펴본다.

계산에 의한 매핑 방법은 디스크 수의 변화에 유연하지 못하기 때문에 디스크를 추가할 경우에는 반드시 데이터 재배치를 하여야 하지만, 매핑 테이블에 의한 매핑, SZIT에 의한 매핑 방법은 반드시 재배치를 필요는 없다.

매핑 연산의 속도면에서 보면 수식 정보가 메모리에 올려져 있는 계산에 의한 매핑과 SZIT에 의한 매핑 방법은 빨리 수행된다. 하지만, 매핑 테이블의 경우에는 저장할 데이터의 양이 매우 많기 때문에 모두 메모리에 올려둘 수가 없다. 또한 논리주소와 물리주소간의 상관관계가 존재하지 않기 때문에 저장장치의 블록 할당여부를 관리하기 위해서 비트맵과 같은 관리기법이 필요하다. 따라서 메모리에 없는 매핑 정보가 필요할 때와 할당여부 정보의 접근을 위해 디스크 I/O가 수행되어야 하기 때문에 연산속도가 느려진다.

계산에 의한 매핑과 매핑 테이블에 의한 매핑은 항상 모든 디스크를 대상으로 스트라이핑을 수행한다. 하지만 SZIT에 의한 매핑에서는 부분 스트라이핑을 수행하는 영역이 존재한다. 디스크를 추가할 때 여러 개의 디스크가 추가된다면 스트라이핑의 효과를 충분히 볼 수 있겠지만, 1개의 디스크만을 추가할 경우에는 스트라이핑 효과를 전혀 볼 수 없는 영역이 존재한다.

5.2 디스크 수의 변화에 따른 매핑 정보의 수 비교

매핑을 블록단위로 하고 블록의 크기를 1KB라고 할 때, 시스템에서 저장해야 할 매핑정보의 크기는 [표 11]과 같다. 여기에서 매핑 테이블에서의 한 튜플 정보의 크기는 논리블록주소 (4 Byte), 디스크 번

호 (1 Byte), 물리블록번호 (4 Byte) 이며, 모든 디스크의 크기는 20 GB이며, 디스크 추가는 2개씩 이루어진다고 가정한다. 또한 SZIT의 한 튜플의 크기는 스트라이핑 지역번호 (1 Byte), 전체 디스크 수 (1 Byte), 참여 디스크 수 (1 Byte), 첫 물리블록번호 (4 Byte), 끝 물리블록번호 (4 Byte), 첫 논리블록번호 (4 Byte), 끝 논리블록번호 (4 Byte)라고 가정한다. 따라서 매핑 정보 테이블의 한 튜플의 크기는 9 Byte가 되며, SZIT의 한 튜플의 크기는 19 Byte가 된다.

계산에 의한 매핑방법은 어떤 경우에도 추가로 유지하는 매핑 정보가 없다. 하지만 매핑 테이블을 사용하는 경우에서는 용량이 커질수록 유지하는 매핑 정보가 증가한다. 초기 상태에서 20 GB 디스크 3개로 구성되어 있는 시스템에서 전체 용량 60 GB의 1.14%인 540 MB의 공간이 매핑 테이블을 위해 할당되어야 한다. 140 Giga Byte 정도의 용량을 가지는 시스템에서 1.2 Giga byte의 메타데이터를 메인 메모리상에 유지하는 것은 힘들기 때문에 디스크 I/O 횟수가 많아지고 이에 따라 성능이 떨어지게 된다. 이에 반해 SZIT를 이용한 매핑 방법은 그보다 훨씬 작은 19 Byte의 공간만 필요하며, 디스크의 추가되는 시점에 따라 스트라이핑 지역 정보가 조금씩 달라질 수 있지만, 데이터 채우기가 끝나고 전체 디스크를 대상으로 스트라이핑을 하고 있을 때 디스크가 추가된다면, 첫 번째 추가 시에는 57 Byte, 두 번째 추가 시에는 95 Byte의 정보만을 유지하면 된다.

[표 6] 매핑 방법의 비교

	수식에 의한 매핑	매핑 테이블에 의한 매핑	SZIT에 의한 매핑
저장할 매핑 데이터 양	불필요	많음	적음
디스크 추가에 대한 데이터 재배치 필요 여부	반드시 필요	필요 없음 (선택 사항)	필요 없음 (선택 사항)
매핑 연산 속도	빠름 (in-memory 연산)	느림 (디스크 입/출력 필요)	빠름 (in-memory 연산)
스트라이핑 대상 디스크	항상 모든 디스크	일부 디스크에만 스트라이핑 하는 경우 존재	일부 디스크에만 스트라이핑 하는 경우 존재

[표 7] 디스크 수의 변화에 따른 매핑 정보의 수

	초기 디스크	첫 번째 추가	두 번째 추가
수식에 의한 매핑	0	0	0
매핑테이블에 의한 매핑	540 MB	900 MB	1260 MB
SZIT를 이용한 매핑	19 B	57 B	95 B
전체 디스크 용량	60 GB	100 GB	140 GB

6. 결론

이 논문에서는 스트라이핑 시스템에서 적은 양의 메타데이터(SZIT)를 관리함으로써 동적 디스크 수를 지원하는 계산에 의한 매핑 방법을 제안하고 기존의 순수한 계산의 의한 매핑법, 매핑 테이블을 이용한 매핑법과 비교를 하였다.

제안하는 방법은 디스크가 추가될 때마다 달라지는 스트라이핑 대상 디스크의 수에 대한 정보를 테이블로 유지하여 이 정보와 수식을 사용하여 논리블럭과 물리블럭을 매핑한다. 즉, 디스크가 새로 추가될 때, 새로 추가된 디스크만을 통해 스트라이핑으로 데이터를 저장하여 기존 디스크의 용량만큼 데이터가 채워지면 그때부터 전체 디스크를 대상으로 스트라이핑을 수행한다.

제안하는 방법은 다양한 크기의 디스크를 지원하는 방법, 온라인 백업을 위한 스냅샷, SZIT 매핑법에서의 재구성 방법 등의 추가 연구가 필요하다.

참고문헌

[1] D.A. Patterson, J.L.Hennesy, R.H.Katz, "A case for redundant arrays of inexpensive disk(RAID)", International Conference of Management of Data(SIGMOD), pp.109-116, 1988.

[2] Steven R. Soltis, Thomas M. Ruwart, Matthew T. O'Keefe, "The Global File System", Conference on Mass Storage Systems and Technologies, Sept 17-19, 1996.

[3] Edward K. Lee, Chandramohan A. Thekkath, "Petal:Distributed Virtual Disks", In Proceedings of International Conference on ASPLOS, 1996.

[4] "Features of veritas volume manager and veritas file system", http://www.veritas.com