RAID 5의 성능 향상을 위한 부하 균등 destage 기법

(Load-Balancing Destage Algorithm for the Performance Enhancement of RAID 5)

장 윤 석 † 김 종 상 ††
(Yun Seok Chang) (Chong Sang Kim)

요 약 디스크 캐쉬를 사용하는 RAID 5에서 읽기 요구들에 대한 응답 시간은 쓰기 요구들을 디스크 캐쉬로부터 디스크로 스케줄하는 반출 알고리즘에 따라서 많은 영향을 받는다. RAID 5는 다수의 디스크들로 디스크배열을 이루고 있으므로 호스트 입출력 요구, 특히 읽기 요구들이 효과적으로 처리되기 위해서는 개개의 디스크의 입출력 성능 향상도 중요하지만 전체적인 디스크배열 차원에서의 성능 향상도 매우 중요하다. RAID 5제어기는 하나의 호스트 입출력 요구에 대하여 여러 디스크로 분산된 입출력 요구들을 발생시키고 이들이 각 디스크에서 모두 처리되어야만 입출력 처리가 종료되므로, 호스트 입출력 요구에 대한 응답 시간은 디스크배열을 이루는 디스크들 중 가장 부하가 큰 디스크에 의하여 결정되어진다. 따라서 일부 디스크에 부하가 집중되어 해당 디스크에서의 입출력 처리 시간이 증가되면 다른 디스크에서의 입출력 처리 시간이 아무리 짧더라도 전체적인 호스트 입출력 시간은 증가된다. 그러나 기존의 반출 알고리즘들은 개개의 디스크에서의 입출력 최적화 방법에 관한 문제만을 다루고 있으므로 디스크배열의 관점에서 디스크들간의 부하 불균형에 의하여 응답 시간이 증가되는 문제를 해결하기는 어렵다. 본 연구에서는 기존의 반출 알고리즘에 디스크배열 차원에서의 부하 균등 기법을 적용한 새로운 반출 알고리즘을 제안하고 모의 실험 도구를 이용하여 성능을 평가함으로써 새로운 알고리즘이 기존의 알고리즘에 비하여 우수한 성능을 보임을 입증하였다.

Abstract Write requests are written from the disk cache to the disks by a destage algorithm, and the response time of host read request dominates the performance of the disk. Since RAID is composed of multiple disks, although the performance at the disk level is important to service requests effectively, the performance at the disk array level is more important. In RAID, a host request can not be completed until all the striped requests are completed and the response time of the host request is dependent on the response time of the disk with the heaviest load. However, previous destage algorithms do not take into consideration the overall performance of all the disks in the RAID but destsge write requests for optimal performance at an individual disk, and it may cause overload on a few disks. Thus, it increases the service time of some striped requests, and therefore, the response time of host request increases. This paper suggests new Load-Balancing Destage(LBD) algorithm adopted at the disk array level, and shows that the LBD algorithm has higher performance than previous destage algorithms by evaluating their performance using a simulator.

1. 서 론

일반적으로 컴퓨터 시스템은 전자적인 요소들과 기계적인 요소들이 결합되어 구성된다. 이들 중 기계 요소들의 동작 속도는 전자적인 요소들에 비하여 매우 느린 편이다. 특히, 디스크 시스템은 운영 체제와 같은 컴퓨터 시스템의 기본적인 동작으로부터 사용자 데이타 관리에까지 컴퓨터 운영에 전반적으로 깊이 관련되어 있으므로 기계 요소 중에서는 컴퓨터 시스템의 성능에 가장 큰 영향을 끼친다. 따라서 디스크 시스템의 성능을

† 정 회 원 대진대학교 컴퓨터공학과 교수
†† 종신회원 : 서울대학교 컴퓨터공학과 교수
논문접수 : 1997년 5월 28일
심사완료 1997년 9월 19일

향상시키기 위하여 여러 가지 각도에서 많은 연구들이 수행되어 왔다. 디스크 시스템에 관련된 연구는 크게 2가지로 분류될 수 있다. 첫째로는 디스크 메커니즘의 개선이나 디스크 스케줄링 알고리즘의 개선, 그리고 디스크 캐쉬의 성능 개선에 관한 연구 등으로 이들은 개개의 디스크 시스템에 관한 연구들로 분류될 수 있다[2, 6, 14, 15, 17] 둘째로는 RAID와 같이 다수의 디스크를 사용하는 디스크 배열에 관한 연구가 있다[3, 5, 7, 10, 11, 16].

최근의 대형 컴퓨터 시스템에서는 단일의 대용량 디스크를 사용하기보다는 여러 개의 디스크들을 연동하는 디스크배열(disk array)을 이용하는 추세에 있다. 그러나 디스크 배열은 보통 다수의 대용량 디스크들로 구성되므로 평균 고장 발생 시간(MTTF:Mean Time To Failure)이 단일 디스크에 비하여 짧기 때문에 대용량의 데이타들을 안정적으로 저장하기 어렵다. 따라서 최근에는 디스크 배열에 결함 허용 특성을 부가한 RAID(Redundant Arrays of Inexpensive Disks)가 점차로 많이 사용되고 있다. RAID는 데이타의 결함을 제어하기 위한 데이타 기록 방법에 따라서 레벨 0, 1, 2, 3, 4, 5로 분류된다[4]. 현재 대부분의 상용 RAID 시스템에서는 RAID 레벨 5(RAID 5)가 우수한 응답 성능을 보이므로 많이 이용된다. 또한 대부분의 대용량 디스크 시스템들은 입출력 수행 시간을 효과적으로 줄이기 위하여 디스크 캐쉬를 사용한다[15]. 디스크 시스템에 디스크 캐쉬를 사용하면 읽기 요구에 대한 입출력 수행 시간에 대하여 많은 이득을 얻을 수 있다. 또한 쓰기 캐쉬를 읽기 캐쉬와는 별도로 NVRAM(Non-Volatile RAM)으로 구성하면 쓰기 요구에 대해서도 상당한 이득을 얻을 수 있다[1, 12]. RAID 5 에서도 제어기 내에 디스크 캐쉬를 설치하여 쓰기 캐쉬를 NVRAM으로 구성하고 적절한 반출(destage) 기법을 사용하여 디스크로의 쓰기 동작을 수행하면 입출력 수행 시간을 효과적으로 줄일 수 있다[8, 9, 17]. 따라서 많은 상용 RAID 5 시스템들은 제어기내에 대용량의 디스크 캐쉬를 채택하고 있다. 이와 같은 RAID 5 시스템에서 디스크 캐쉬에 저장된 쓰기 요구들은 반출 알고리즘에 따라서 디스크로 쓰여지며, 반출 알고리즘은 쓰기 캐쉬 내의 데이타들을 디스크로 스케줄하는 순서와 방법을 결정한다.

RAID 5에서는 하나의 입출력 요구가 발생되면 여러 디스크로 입출력 요구가 분산되어 처리되기 때문에 일부 디스크들에서의 부하가 증가되면 평균 대기 지연 시간이 증가되므로 다른 디스크들의 부하가 작더라도 호스트 관점에서의 입출력 처리 시간, 특히 읽기 요구에 대한 처리 시간이 증가된다. 반대로 전체 부하의 크기가 동일한 경우에, 부하가 각 디스크에 고루 분산되면 각 디스크에서의 평균 대기 시간이 줄어들게 된다. 따라서 다수의 디스크가 사용되는 RAID 5에서는 개개의 디스크의 성능 향상도 중요하지만 디스크간의 적절한 부하 조절 기법도 역시 중요하다. 특정한 디스크에 부하가 증가되어 읽기 요구들에 대한 디스크 입출력 수행 시간이 지연되는 경우를 방지하기 위해서는 디스크 캐쉬로부터 디스크들로 쓰여지는 쓰기 요구들을 결정할 때 디스크들의 부하 크기에 따라서 쓰기 요구들의 처리 순서를 결정하여야 할 필요가 있다. 이는 디스크 캐쉬 제어기에서 디스크로 쓰여질 쓰기 요구들에 대한 반출 동작을 적절히 제어함으로써 수행될 수 있다. 그러나 기존에 연구된 바 있는 반출 알고리즘들은 디스크배열을 구성하는 디스크들을 전체적으로 고려하는 것이 아니라 개개의 디스크별로 최적의 반출 동작을 수행하도록 되어 있으므로 특정 디스크들에 부하가 불균등하게 부가될 수 있다.

본 연구에서는 쓰기 요구들을 디스크에 반출할 때, 디스크의 부하 조절을 위해서 디스크 캐쉬 내의 쓰기 요구들을 각 디스크의 큐잉 길이에 따라서 선별하여 반출하는 새로운 부하 균등 반출(Load-Balancing Destage : LBD) 알고리즘을 제안하였다. 이는 디스크의 부하를 큐 길이로 설정하고 디스크 캐쉬로부터 반출될 쓰기 요구들을 스케줄링할 때 디스크 큐 길이의 역순으로 스케줄링하여 가장 부하가 적은 디스크에 해당되는 반출 요구들을 먼저 디스크에 큐잉 되도록 함으로써 결과적으로 디스크배열을 이루는 디스크들의 부하를 균등하게 조절할 수 있다. 이러한 부하 균등 반출 알고리즘에 대한 성능 평가를 위하여 본 연구에서는 디스크 캐쉬를 포함하는 RAID 5 모델을 구성하고 모의 실험 도구를 사용하여 부하 균등 반출 알고리즘을 사용한 RAID 5의 입출력 성능을 기존의 반출 알고리즘과 비교 평가하였다.

2. 부하 균등 반출 알고리즘

단일 디스크 시스템과 마찬가지로 RAID 5 에서도 호스트로부터 입출력 요구가 발생되면 RAID 제어기 내의 디스크 캐쉬에서 먼저 처리된다. 그러나 RAID 5의 디스크 캐쉬는 하나의 디스크에 대한 데이타 열을 취급하기보다는 다수의 디스크들에 대한 데이타 열들을 취급하기 때문에 단일 디스크와는 다른 응답 특성을 가진다. 단일 디스크 시스템에서는 호스트로부터 읽기 요구가 발생되었을 때, 요구된 데이타 블록들이 모두 캐쉬에서

적중되면 이들은 즉시 호스트로 전송된다. 반면에 읽기 요구에서 요구된 데이타 블록들의 전부, 또는 일부가 캐쉬에서 적중되지 않은 경우에 디스크 제어기는 이들에 대한 읽기 요구를 발생시킨다. 호스트로부터의 쓰기 요구가 발생된 경우에는 데이타 블록들은 일단 디스크 캐쉬에 저장된 다음에 적절한 시기에 반출 알고리즘에 따라서 각 디스크로 스케줄된다. 그러나 RAID 5에서 입출력 요구들이 디스크 캐쉬에서 적중되지 않을 경우 이에 대한 처리 과정은 단일 디스크 시스템에 비하여 좀 더 복잡하다. 호스트로부터의 읽기 요구가 디스크 캐쉬에서 적중되지 않은 경우, 요구된 데이타의 크기가 RAID의 분할 단위(striping unit)보다 크면 RAID 5 제어기는 데이타를 분할 단위로 나누고 해당 디스크들로의 개별적인 읽기 요구들을 발생시킨다. 이들 개별적인 읽기 요구들은 해당 디스크에 각각 스케줄되어 디스크 스케줄링 알고리즘에 따라서 처리된다. 각 디스크들은 독립적인 입출력 큐를 사용하여 입출력 요구들을 처리하므로 분할된 개개의 읽기 요구들에 대한 처리 시간은 각 디스크의 부하와 디스크 헤드의 물리적인 위치에 따라서 달라지게 된다. 디스크 i 에서 읽기 요구가 처리되는 시간 $T_read(i)$는 다음과 같이 계산될 수 있다.

$$T\_read(i) = T\_queueing(i) + T\_seek(i) + T\_transfer(i)$$

i : 디스크 번호, $0 < i < k$

k : 디스크 배열의 크기

호스트로부터 발생된 읽기 요구는 분할된 개별적인 입출력 요구들에 대한 처리가 모두 완료되어야만 종료될 수 있다. 그러므로 개별적인 읽기 요구에 대한 처리 시간 $T\_read(i)$ 중 가장 큰 값이 호스트로부터의 읽기 요구에 대한 처리 시간 T_host_read 가 된다.

$$T\_host\_read = MAX(T\_read(0), T\_read(1), ..., T\_read(k))$$

디스크에서의 읽기 요구에 대한 탐색 시간 $T_seek(i)$와 $T_transfer(i)$는 디스크와 데이타에 대한 물리적인 고정 인자에 의하여 결정된다. 그러나 큐잉 지연 시간 $T_queueing(i)$은 개개의 디스크 큐에 대기하고 있는 입출력 요구들의 수, 즉 큐 길이에 영향을 받는다. 비록 큐잉된 입출력 요구들은 탐색 시간이나 기타 여러 요소들에 의하여 처리 순서가 재배치될 수 있지만, 평균적으로 큐 길이가 길면 입출력 요구에 대한 처리 시간이 길어진다. 디스크 큐에는 다음과 같은 두 종류의 입출력 요구들이 큐잉된다. 첫째로는 호스트로부터 요구되는 읽기 요구(demand read) 중 캐쉬에서 적중되지 못한 읽기 요구들이 큐잉된다. 이들에 대한 처리 시간은 호스트 시스템의 입출력 처리 시간에 큰 영향을 끼치게 되므로 가능한 한 빨리 처리되어야만 한다. 따라서 보통, 호스트로부터의 읽기 요구는 제어기 레벨에서 다른 입출력 요구들에 대하여 우선 순위를 가진다. 둘째로는 반출 요구들이 큐잉된다. 이는 호스트로부터 디스크에 쓰여질 데이타에 대한 패리티를 계산하는 데에 필요한 이전 데이타와 패리티에 대한 읽기 요구들과, 반출 알고리즘에 의하여 디스크 캐쉬로부터 방출되는 쓰기 데이타 및 패리티 데이타에 대한 쓰기 요구들이 포함된다. 비록 디스크 캐쉬를 사용하는 디스크 시스템에서 쓰기 요구들은 백그라운드로 처리되기는 하지만 쓰기 요구들이 디스크에서 처리되고 있는 동안에 호스트로부터 요구된 읽기 요구들은 이들 쓰기 요구들에 의하여 처리가 지연될 수 있다. 따라서 일부 디스크 제어기는 호스트로부터의 읽기 요구에 대한 큐와 쓰기 요구에 대한 반출 큐를 별도로 운영하여 호스트로부터의 읽기 요구에 대한 큐에 우선 순위를 주는 방식을 사용하기도 한다[17]. 이와 같은 방법은 디스크의 부하가 비교적 낮은 경우에 호스트 읽기 요구들에 대한 응답 시간을 효과적으로 줄일 수 있다.

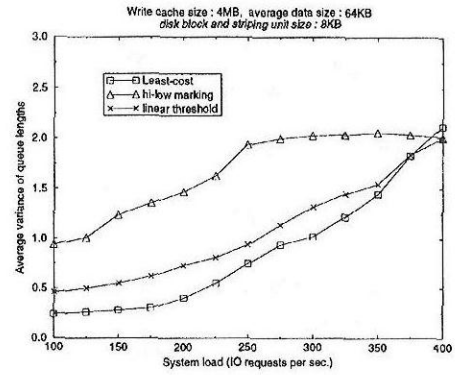

그림 1 부하에 따른 디스크별 큐 길이의 분산

그러나 읽기 요구에 무조건 우선 순위가 부여될 경우, 호스트로부터의 입출력 부하가 증가되면 디스크 캐쉬에 저장된 쓰기 데이타들을 디스크로 반출할 수 있는 기회가 줄어들고, 이로 인하여 디스크 캐쉬가 쓰기 요구들로 포화상태가 될 확률이 높아진다. 디스크 캐쉬가 포화상

태가 되면, 디스크 캐쉬 내에 빈 공간을 생성하기 위하여 반출 요구들은 읽기 요구에 대해서도 우선하여 강제적으로 디스크에 쓰여져야만 한다. 입출력 부하가 매우 커지면, 이러한 강제적인 반출 요구들이 디스크에 많이 큐잉되므로 이들을 처리하는 동안 호스트로부터 발생되어진 읽기 요구들에 대한 처리가 지연된다. 만약에 반출 요구에 의하여 발생되는 부하가 RAID 5 의 디스크배열을 구성하는 모든 디스크에 대하여 동일하고, 호스트로부터의 읽기 요구가 모든 디스크에 동등하게 분할되어 할당되어진다면 RAID 5에서의 읽기 요구에 대한 처리 시간은 단일 디스크의 경우와 동일하게 계산될 수 있다. 그러나 제 4장의 모의 실험 도구를 이용하여 RAID 5 에서의 디스크별 부하 분포를 조사해 보면 부하 분포는 각 디스크에 대하여 균일하지 않고 큰 차이를 보이며 이는 반출 알고리즘에 따라서도 차이를 보인다. 본 연구에서는 기존에 연구되어진 반출 알고리즘으로 최소 부하 스케쥴링 알고리즘(least-cost scheduling algorithm), 상하 제한 알고리즘(hi-low marking algorithm), 그리고 선형 한계 스케쥴링 알고리즘(linear threshold scheduling algorithm)의 3가지 반출알고리즘을 비교하였다. 그림 1은 디스크 배열을 구성하는 디스크들에 대한 개별적인 부하, 즉 큐 길이의 편차의 평균 V_{avg}를 호스트로부터의 부하에 따라서 반출 알고리즘별로 모의 실험한 결과를 나타낸다. V_{avg}는 다음과 같이 계산되었다.

$$V_{avg} = \frac{\sum_{i=1}^{k} V(i)}{k} \quad \text{and} \quad V(i) = \frac{\sum_{t=0}^{T_{end}} (Qi - Qm)^2}{T_{end}}$$

여기서 Qi는 각 디스크 i의 큐 길이를, Qm은 각 시간 t에서 각 디스크들의 큐 길이의 평균값을 나타내고, T_{end}는 모의 실험이 종료되는 시간을 초 단위로 나타낸다..

그림에서 V_{avg}는 어떤 반출 알고리즘에 대해서도 비교적 큰 값을 나타낸다. 이는 각 디스크간 큐 길이의 분포가 매우 불균일하다는 사실을 나타낸다. 실험에서 디스크별로 부하의 차이가 크게 나타나는 원인은 기존의 반출 알고리즘이 디스크간 부하 분포에는 관계없이 개개의 디스크의 조건에 의해서만 동작하도록 설계되어 있기 때문이다. 이 때문에 각 디스크별 큐 길이에 차이가 발생되고, 따라서 개별적인 디스크 내에서는 입출력 요구 처리 시간의 최적화가 이루어질 수 있어도 디스크 배열의 차원에서는 입출력 처리 시간이 최적화 되지 못하는 결과를 나타낸다.

이와 같은 문제점을 해결하기 위해서는 기존의 반출 알고리즘에 디스크간의 부하 분포에 대한 요소들을 포함하여야 할 필요가 있다. 이는 다음과 같은 매우 간단한 알고리즘을 기존의 반출 알고리즘에 포함시킴으로써 구현될 수 있다.

< 부하 균등 반출 알고리즘 >

1) 디스크 캐쉬로부터 반출되어질 쓰기 요구들은 기존의 반출 알고리즘에 따른 순서대로 배열한다.
2) 디스크들 중 큐 길이가 가장 작은 디스크에 해당되는 반출 요구를 배열에서 탐색한다.
3) 탐색된 반출 요구를 디스크로 스케줄한다.
4) 반출 동작이 가능할 때마다 위의 동작을 반복한다.

이와 같이 반출될 쓰기 요구들을 디스크 캐쉬로부터 디스크로 반출할 때 디스크의 큐 길이가 가장 짧은 디스크에 해당되는 반출 요구를 먼저 반출하면 전체적인 디스크들의 큐 길이를 균등하게 유지할 수 있다. 부하 균등 반출 알고리즘은 매우 간단한 방법으로 일부 디스크에 부하가 집중되는 것을 방지하고, 호스트로부터의 읽기 요구에 의하여 생성된 읽기 요구들에 대한 최대 처리 시간 *T_host_read*를 줄이는 효과를 나타낼 수 있다.

3. RAID 5 모의 실험 도구

기존의 반출알고리즘들과 제안된 부하 균등 반출 알고리즘의 성능을 평가하기 위한 모의 실험 도구로 본 연구에서는 RAID 5 시뮬레이터를 구현하였다. 이는 Berkeley와 CMU에서 개발된 RAID 시뮬레이터인 raidSim을 기본으로 하고 RAID 제어기에 디스크 캐쉬를 구현하여 재구성되었다. 디스크 캐쉬는 기존의 여러 연구에서 제안된 디스크 캐쉬 운영 알고리즘들을 포함하고 기존의 반출 알고리즘과 제안된 부하 균등 반출 알고리즘을 모두 포함하여 구현되었다. 시뮬레이터는 다음과 같은 부분들로 구성되어 있다

- 합성 부하 생성기(synthetic workload generator)
- RAID 5 시뮬레이터
- 디스크 캐쉬 시뮬레이터
- 디스크 시뮬레이터

이 중 합성 부하 생성기와 RAID 시뮬레이터는 raidSim을 기본으로 구성하였다. 디스크 캐쉬 시뮬레이터는 dinero-III를 기본으로 하여 본 연구에서 별도로

구현되었으며 디스크 시뮬레이터는 HP의 디스크 모델링 방법에 의하여 구현하였다.

3.1 합성 부하 생성기

다양한 입출력 패턴과 부하에 대한 RAID 5의 성능을 평가하기 위해서는 시뮬레이터의 입력은 입출력의 종류, 크기, 분포, 부하 등의 여러 입출력 요소들을 반영할 수 있어야 한다. 그러나 실제 디스크 시스템으로부터 이와 같은 입출력 트레이스를 얻기는 매우 어렵다. 따라서 대부분의 RAID 관련 연구에서는 실제적인 입출력 트레이스를 사용하기보다는 인위적인 부하 생성기를 구성하여 RAID 5 시뮬레이터의 입력으로 사용하고 있다. 이는 다음과 같은 요소들을 조절하여 필요한 조건의 부하를 발생시킬 수 있도록 하고 있다.

- 입출력의 종류 (read/write) 및 발생 확률 (probability)
- 입출력 데이타 및 데이타 정렬 크기
- 입출력 크기의 분포 방법(deterministic / exponential)
- 지역성(locality)을 발생시키기 위한 지역 공간의 위치와 크기

합성 부하 생성기는 디스크배열상의 모든 데이타 블록에 대한 입출력 요구들을 임의로 발생시키기보다는 다중 프로세스 그룹들에 의하여 입출력 요구의 일부가 지역성을 가지도록 하고 있다. 각 프로세스 그룹은 입출력 요구 발생률(request rate), 읽기/쓰기의 비율, 지역성의 정도와 같은 요소들을 임의로 조정할 수 있다. 각 프로세스 그룹은 1 ~ 30개의 프로세스들로 구성된다.

3.2 RAID 5 시뮬레이터

RAID 5 시뮬레이터는 호스트로부터 입출력 버스를 통하여 요구되거나 전송된 입출력 데이타가 RAID 5의 분할 단위보다 큰 경우 이를 분할 단위로 나누어 여러 개의 입출력 요구들로 변환하고 이들을 디스크배열로 전송한다 일단 개개의 디스크로 입출력 요구들이 전송되면 입출력 요구들은 각 디스크에서 독립적으로 처리되며 모든 입출력 요구들에 대한 처리가 종료되어야만 호스트 입출력 요구가 완료된 것으로 간주한다. 또한 시뮬레이터는 입출력 요구를 처리하는 도중에 에러가 발생된 경우를 시뮬레이션 하는 기능도 사용할 수 있다. 에러가 발생되었다고 가정되면 RAID 5 시뮬레이터는 내부적으로 패리티 데이타와 나머지 패리티 그룹에 속하는 디스크들의 데이타로부터 손상된 데이타를 복원하는 기능을 수행한다. 또한 쓰기 요구에 대한 처리가 수행될 때 필요한 패리티 계산 과정도 수행한다.

3.3 디스크 캐쉬 시뮬레이터

디스크 캐쉬는 읽기 캐쉬와 쓰기 캐쉬의 두 부분으로 구현되었다. 읽기 캐쉬에는 호스트로부터 발생된 읽기 요구나 RAID 제어기로부터 발생되는 반출 읽기 요구에 의하여 디스크로부터 전송된 데이타들을 저장한다. 쓰기 캐쉬에는 호스트로부터 전송된 쓰기 데이타들과 이들에 대한 패리티 데이타들을 저장한다. 이 때 쓰기 캐쉬에 저장되어 있는 데이타와 디스크 상에 저장되어 있는 데이타간의 무결성을 보장하기 위하여 쓰기 캐쉬는 NVRAM으로 구성되어 있다고 가정한다. 두 캐쉬는 모두 호스트 버스 인터페이스와 디스크배열 사이에 위치한다. 따라서 이 디스크 캐쉬는 각 디스크 내에 위치하는 트랙 버퍼나 캐쉬와는 구별된다. 호스트로부터 발생되는 입출력 요구들은 먼저 디스크 캐쉬 제어기에 큐잉되고 FCFS 순서에 따라서 처리된다.

호스트 읽기 요구가 발생되면 디스크 캐쉬 제어기는 읽기 캐쉬와 쓰기 캐쉬를 동시에 탐색한다. 만약 요구되는 모든 데이타들이 캐쉬에서 처리될 수 있다면 데이타는 입출력 버스로 즉시 전송된다. 요구된 데이타중 일부가 디스크 캐쉬에서 처리되지 못하면 디스크 캐쉬 제어기는 해당 디스크들로 읽기 요구들을 발생시킨다. 이 때 발생되는 읽기 요구들은 개개의 디스크에 큐잉되고 이는 반출 요구들에 대하여 우선 순위를 가질 수 있다. 디스크로의 개별적인 읽기 요구들이 모두 완료되어 데이타들이 읽기 캐쉬로 모두 전송되면 데이타들은 하나의 단위로 호스트로 전송된다. 따라서 호스트 읽기 요구가 디스크 캐쉬에서 모두 처리되지 않은 경우의 읽기 요구 처리 시간은 개개의 디스크로 발생되어지는 읽기 요구와 각 디스크에서의 부하에 따라서 달라지게 되고 이에 따라서 입출력 요구의 처리 순서가 달라질 수 있게 된다

호스트로부터 쓰기 요구가 발생되면 데이타는 먼저 쓰기 캐쉬에 저장된다. 이 때 요구된 데이타와 같은 디스크 논리 주소를 가지는 데이타가 쓰기 캐쉬 내에 이미 저장되어 있을 경우에는 여러 가지 방법으로 처리된다. 첫째로 쓰기 캐쉬 내의 데이타에 대한 패리티가 아직 계산되어 있지 않다면 디스크 캐쉬 제어기는 새로운 데이타를 그대로 덮어쓴다. 둘째로 쓰기 캐쉬 내의 데이타에 대한 패리티 값이 이미 계산되어 있고 이 데이타가 아직 디스크에 쓰여지지 않았다면, 디스크 캐쉬 제어기는 새로운 데이타를 위한 캐쉬 공간을 새로이 할당한다. 이미 캐쉬 내에 존재하고 있는 이전 데이타는 반출 알고리즘에 의하여 디스크로 쓰여지도록 스케줄링되거나, 스케줄링되기 전에 새로이 할당된 데이타에 대한 패

리티가 다시 계산되면 캐쉬 내에서 삭제될 수 있다.

디스크 캐쉬 제어기는 쓰기 캐쉬에 새로이 저장된 데이타에 해당되는 패리티값을 계산하기 위한 데이타들이 디스크 캐쉬에서 모두 처리되지 않을 경우, 패리티값 계산에 필요한 데이타들에 대한 입출력 요구들을 발생하여야 한다. 또한 패리티값 계산이 완료되면 제어기는 최종적으로는 데이타 값과 패리티값들을 디스크에 기록하기 위한 쓰기 요구를 발생하여야 한다. 이들 반출 입출력 요구들을 디스크 캐쉬로부터 디스크로 스케줄링하는 방법은 반출 알고리즘에 따라서 정해진다. 본 연구에서는 이와 같은 반출 입출력 요구에 대한 여러 스케줄링 알고리즘을 비교 분석하기 위하여 기존에 제안된 여러 반출 알고리즘들과 본 연구에서 제안된 부하 균등 반출 알고리즘을 디스크 캐쉬 제어기에 구현하였다. 따라서 반출 요구들은 반출 알고리즘에 따라서 이면적으로 수행된다. 부하 균등 반출 알고리즘을 구현하기 위하여 본 연구에서는 각 디스크별 큐 길이를 동적으로 파악하는 자료 구조인 리스트 체인(list chain)을 이용하였다. 리스트 체인은 각 디스크에서 입출력을 수행할 때마다 갱신된다. 부하 균등 반출 알고리즘은 이 리스트 체인을 참조하여 쓰기 캐쉬로부터 디스크배열로 스케줄될 반출 요구를 결정한다.

3.4 디스크 시뮬레이터

디스크 시뮬레이터는 크게 디스크 메커니즘과 트랙 버퍼, 그리고 입출력 버스 인터페이스의 세 부분으로 구성되었다. 여기에서 사용된 디스크 모델은 HP 97560 SCSI 디스크를 기본으로 하고 있다. 이 디스크에 대한 기본 요소인 탐색 시간, 회전 지연 시간, 데이타 전송 시간은 3.1의 디스크 모델에 따라서 구현되었다. 입출력 버스 인터페이스는 20MBps의 속도를 가지는 SCSI 인터페이스를 시뮬레이션 하도록 구현되었다. 버스 상에서 전송되어지는 입출력 요구들은 FIFO 순서로 큐잉된다. 입출력 요구들이 버스 상에서 데이타를 전송할 때에도 FIFO 순으로 버스를 먼저 예약하여야 하므로 버스 상에서의 충돌 지연은 발생되지 않는다고 가정한다.

4. 모의 실험 및 결과

4.1 시스템 구성

본 실험에서 구현된 RAID 5 시뮬레이터의 디스크 배열은 모두 20개의 HP 97560 디스크를 사용하여 2차원 배열로 구성되었다(그림 2). 이는 2개의 열로 구성되며 각 열은 10개의 디스크들을 포함한다. 디스크 데이타는 4KB의 기본 블록크기를 가지며 모든 디스크 입출력은 기본 블록크기의 배수로 수행된다. 분할 단위의 크기는 대부분의 OLTP 응용에 적합하도록 8 ~ 128KB로 설정하였다. 전체 디스크 공간이 20GB를 넘는 크기이므로 읽기 캐쉬의 크기는 전체 디스크의 0.1%정도인 16MB로 설정하였다. 쓰기 캐쉬의 크기는 1MB로부터 16MB까지 변화할 수 있도록 하였다. 두 캐쉬는 모두 기본적으로는 LRU 교체 전략을 사용하며 캐쉬 블록의 크기도 4 ~ 32KB 내에서 변화되어질 수 있으나 본 실험에서는 주로 8KB 블록으로 설정하여 사용하였다.

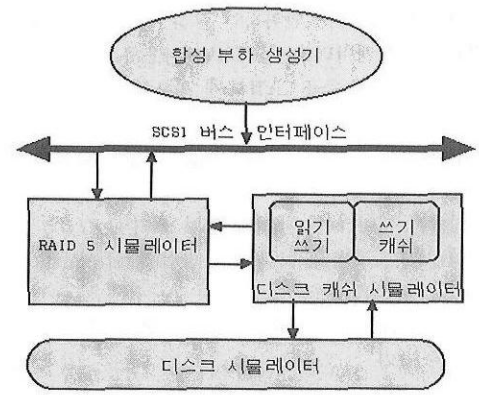

그림 2 시뮬레이션 모델의 구성

합성 부하 생성기로부터 발생되는 전체 입출력 요구 중 읽기 요구는 40%, 쓰기 요구는 60%의 비율로 생성된다. 입출력 데이타의 크기는 평균 64KB로 8KB 단위로 정렬된다. 입출력 발생 분포는 지수분포이다. 또한 입출력 요구들 중 일부는 지역성을 가지도록 하기 위하여 전체 입출력 요구 중 70%가 전체 디스크 주소 공간의 5% 이내에서 집중적으로 발생되도록 하였다.

4.2 모의 실험 측정 요소

반출 알고리즘에 따른 RAID 5의 성능은 기존에 연구된 최소 비용 스케줄링 알고리즘, 상하 제한 알고리즘, 선형 한계 스케줄링 알고리즘과 본 연구에서 제안된 부하 균등 반출 알고리즘에 대한 성능을 비교함으로써 평가될 수 있다. 이를 위해서는 반출 알고리즘이 RAID 5의 성능에 미치는 요소들을 정의하고 이들을 정량적인 값으로 표현할 수 있어야 한다. 본 논문에서는 다음과 같은 요소들에 의하여 RAID 5의 성능을 평가하였다.

1) 읽기 요구에 대한 응답 시간(Average Host Read Response Time)

디스크캐쉬를 사용하는 디스크 시스템에서 가장 중요한 성능 평가 요소는 호스트 읽기 요구에 대한 응답 시간이다. 호스트 읽기 요구가 디스크 캐쉬에서 적중되지

못하면 이들은 분할 단위의 데이타 크기를 가지는 여러 읽기 요구들로 분산되어 직접 각 디스크에 큐잉되지만 디스크 캐쉬 내에 저장되어 있는 반출 요구들은 반출 알고리즘에 따라서 각 디스크로 큐잉된다. 그러므로 각 디스크의 큐 길이는 이전에 큐잉된 호스트 읽기 요구들과 반출 요구들에 의하여 결정되며, 분산된 호스트 읽기 요구들은 각 디스크의 큐 길이에 따라서 그 처리 시간이 달라지게 된다. 따라서 읽기 요구에 대한 응답 시간은 반출 알고리즘이 읽기 요구들의 처리에 끼치는 영향을 잘 반영할 수 있다.

2) 최대 처리량(Maximum Throughput)

최대 처리량은 단위 시간당 최대로 처리될 수 있는 입출력 데이타의 양으로 정의된다. RAID 5에서의 입출력 처리량을 높이기 위해서는 개개의 입출력 요구들이 제어기에서 대기 지연되는 시간을 가능한 한 최소화하여야 한다. 디스크 캐쉬를 사용하는 RAID 5에서 대기 지연 시간은 각 디스크의 큐 길이와 디스크 스케줄링 알고리즘에 따라서 달라지고 큐 길이는 반출 알고리즘에 따라서 달라지며, 각 디스크에서의 대기 지연 시간이 줄어들수록 단위 시간당 처리될 수 있는 입출력 요구의 처리량은 증가될 수 있다. 따라서 최대 처리량은 부하의 변화에 따른 반출 알고리즘의 성능을 반영할 수 있는 요소가 될 수 있다.

3) 디스크별 큐 길이의 분산(Variance of Disk Queue Length)

호스트로부터 발생된 읽기 요구들이 각 디스크에 분산되어 처리될 때, 각 디스크에서의 처리 시간은 디스크의 큐 길이에 직접적으로 영향을 받는다. 단일 디스크 시스템에서는 평균 큐 길이가 길어지면 대기 지연 시간이 증가되므로 평균적으로 디스크에서의 입출력 처리 시간이 증가된다. RAID 5에서 호스트 읽기 요구는 여러 디스크에서 분산 처리되므로 응답 시간은 최대 처리 시간을 가지는 디스크의 처리 시간에 의하여 결정된다. 따라서 동일한 부하에서는 각 디스크별 큐 길이의 편차가 작을수록 디스크배열 관점에서의 최대 처리 시간 *T_host_read*가 줄어들게 된다. 이러한 큐 길이의 편차는 시스템의 부하와 반출 알고리즘에 의하여 달라질 수 있으므로 읽기 요구에 대한 처리 시간, 즉 응답 시간과 직결되는 성능 측정 요소가 된다.

본 연구에서 구성된 모의 실험 도구는 이와 같은 RAID 5의 여러 성능 측정 요소들에 대한 정량적인 결과를 생성하여 준다. 모의 실험은 기존의 반출 알고리즘과 본 논문에서 제안된 부하 균등 반출 알고리즘에 대하여 시스템 부하의 크기와 쓰기 캐쉬의 크기를 변화시키면서 수행되었다. 시스템의 부하는 합성 부하 생성기에 의하여 단위 시간당 발생되는 입출력 요구의 수 (IO requests per second)로 표현된다. 그리고 어떤 경우에도 디스크 캐쉬 폭주는 발생되지 않는 것으로 가정한다.

4.3 실험 결과

본 논문에서 제안한 부하 균등 반출 알고리즘과 기존에 연구되어진 반출 알고리즘들의 성능을 비교 평가하기 위하여 먼저 반출 알고리즘별로 성능 평가 요소들을 측정하였다. 여기에서 쓰기 캐쉬의 크기는 16MB이고 분할 단위의 크기는 16KB로 하고, 디스크 캐쉬 블록 크기는 4KB 단위를 사용하였다.

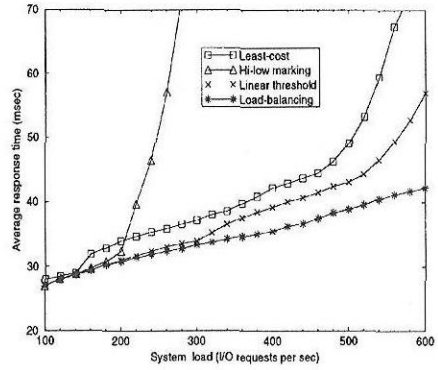

그림 3 시스템 부하에 따른 응답 시간의 변화

성능 평가 요소 중 컴퓨터 시스템의 성능에 가장 큰 영향을 주는 요소는 읽기 요구에 대한 응답시간이다. 그림 3은 시스템의 부하의 변화에 따른 응답 시간의 변화를 반출 알고리즘별로 나타낸다. 여기서 부하 균등 반출 알고리즘은 최소 비용 스케줄링 알고리즘이나 상하 제한 알고리즘에 비해서는 우수한 성능을 나타내지만 낮은 부하에서는 선형 한계 스케줄링 알고리즘과 거의 비슷한 성능을 나타낸다. 이는 본 연구에서 제안된 반출 알고리즘이 기본적으로 선형 한계 스케줄링 알고리즘을 기본으로 적용하고 있기 때문이다. 선형 한계 스케줄링 알고리즘은 비교적 부하가 작은 경우에 시스템의 부하 증가에 따른 쓰기 캐쉬 점유율을 능동적으로 조절하여 높은 캐쉬 점유율을 유지할 수 있도록 설계되어 있으므로 읽기 요구들이 쓰기 캐쉬에서 적중되거나 쓰기 요구들이 쓰기 캐쉬에 재 저장(re-write)될 확률이 다른 알고리즘에 비하여 높다. 또한 기존에 제안된 반출 알고리즘들은 시스템의 부하가 증가됨에 따라서 응답 시간이 급격히 증대되는 데에 비하여 부하 균등 반출 알고리즘은 부하를 여러 디스크들로 분산시키므로 응답 시간이

비교적 천천히 증가된다. 따라서 시스템의 부하가 큰 경우에 부하 균등 반출 알고리즘은 다른 반출 알고리즘에 비하여 우수한 성능을 보인다.

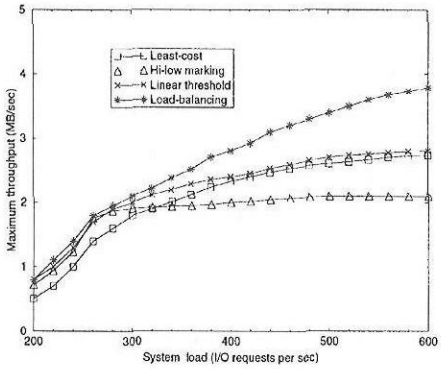

그림 4 시스템 부하에 따른 최대 처리량의 변화

그림 4는 부하의 변화에 따른 최대 처리량의 변화를 나타낸다. 부하 균등 반출 알고리즘은 다른 알고리즘들에 비하여 비교적 우수한 처리 성능을 보이고 있음을 알 수 있다. 시스템의 부하가 증가되면 기존에 제안된 반출 알고리즘들은 최대 처리량의 증가율이 거의 정체되는 경향을 보이고 있다. 이는 각 디스크로 분산된 호스트 입출력 요구에 대한 처리 시간이 입출력 큐 길이가 큰 디스크에서는 큰 값을 가지기 때문이다. 이 경우 해당 디스크에서 대기 지연이 발생되고, 이로 인하여 하나의 호스트 입출력 요구의 처리에 걸리는 시간이 증가되므로 단위 시간당 평균적으로 처리될 수 있는 입출력 요구들의 수는 제한된다. 그러나 부하 균등 반출 알고리즘은 디스크배열을 이루는 개개의 디스크들의 부하가 모두 공평하게 커질 때까지는 부하가 증가되더라도 최대 처리량이 계속 증가될 수 있다.

반출 알고리즘별로 호스트 읽기 요구에 대한 응답시간과 처리량의 증가율에 차이를 나타내는 원인은 디스크배열을 이루는 각 디스크에서의 시스템 부하에 따른 큐 길이 변화로 분석될 수 있다. 그림 5는 시스템의 부하에 따른 개별적인 디스크 큐 길이의 편차 V_{avg}을 나타낸다. 그림 1에서 나타난 바와 같이 부하 균등 반출 알고리즘을 제외한 다른 반출 알고리즘들의 큐 길이는 큰 편차를 나타내고 있으며, 이는 부하가 증가함에 따라서 더욱 커지는 경향을 보이고 있다. 이에 반하여 부하 균등 반출 알고리즘은 부하를 동적으로 여러 디스크로 분산시키므로 다른 알고리즘들에 비하여 비교적 작은 편차를 나타내고 있다.

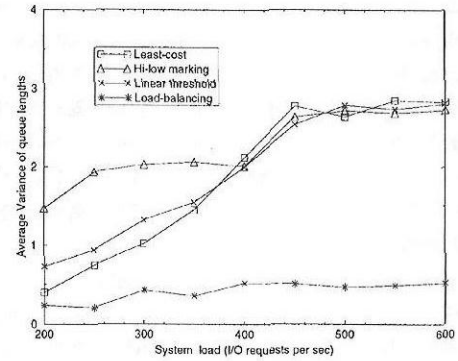

그림 5 시스템 부하에 따른 큐 길이의 분산

그림 6은 각 디스크의 평균 큐 길이를 변화시켰을 때, 호스트 읽기 요구에 대한 응답 시간의 변화를 나타낸다. 부하 균등 반출 알고리즘을 제외한 다른 반출 알고리즘들은 큐 길이 1을 기준으로 하여 1이하인 경우에 응답 시간은 부하 균등 반출 알고리즘과 비교하여 큰 차이가 없다. 그러나 큐잉 길이가 1을 초과하게 되면 선형 한계 스케줄링 알고리즘이나 부하 균등 반출 알고리즘에서는 큐 길이가 증가하여도 응답 시간이 비교적 천천히 증가되는 반면, 다른 알고리즘들은 급격하게 응답 시간이 증가됨을 알 수 있다.

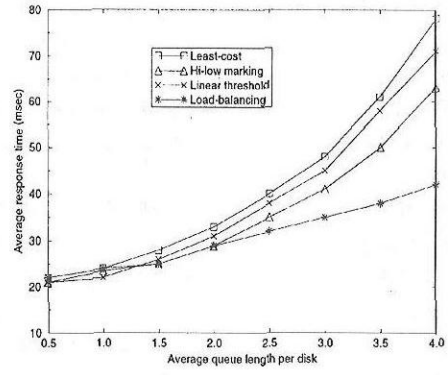

그림 6 평균 큐 길이에 따른 응답 시간의 변화

위와 같은 결과들을 분석하여 보면, 시스템의 부하가 비교적 적을 때에는 여러 반출 알고리즘들의 입출력 성능에는 큰 차이가 없다. 그러나 시스템의 부하가 일정한도 이상으로 증가되면 부하 균등 반출 알고리즘이 다

른 알고리즘에 비하여 우수한 성능을 보임을 알 수 있다. 특히 부하 균등 반출 알고리즘은 디스크 캐쉬에서도 비교적 우수한 성능을 보이고 있다. 그림 8은 쓰기 캐쉬의 크기 변화에 따른 읽기 요구에 대한 쓰기 캐쉬에서의 적중률을 나타낸다. 디스크 캐쉬만을 고려할 경우, 읽기 캐쉬의 크기와 캐쉥 알고리즘은 고정되어 있으므로 쓰기 캐쉬에서의 적중률은 읽기 요구에 대한 응답 시간에 영향을 미칠 수 있으며 이는 쓰기 캐쉬의 평균 점유율과 관계가 있다. 부하 균등 반출 알고리즘에서 쓰기 캐쉬로부터 디스크로 반출 요구를 발생시키는 알고리즘은 선형 한계 스케줄링 알고리즘과 동일한 방법을 사용하고 있다. 따라서 그림 7에서 부하 균등 반출 알고리즘과 선형 한계 스케줄링 알고리즘은 거의 동일한 캐쉬 적중률을 보이고 있다. 또한 부하에 따른 평균 쓰기 캐쉬 점유율에서도 두 알고리즘은 거의 같은 경향을 보이고 있다 (그림 8).

그림 9, 10은 부하 균등 반출 알고리즘에서 쓰기 캐쉬의 크기에 따른 응답 시간과 처리량의 변화를 나타낸다. 쓰기 캐쉬의 크기가 증가하면 읽기 요구들이 쓰기 캐쉬에서 적중될 확률이 증가되고, 이에 따라서 평균 응답 시간이 줄어들게 된다. 쓰기 캐쉬에 저장되는 반출 요구들이 증가되면 반출되기에 적절한 요구들이 많아지게 된다. 따라서 최적의 반출 요구들을 선별하여 스케줄할 수 있는 확률이 증가되므로 디스크별 평균 대기시간이 줄어들게 된다. 또한 응답 시간은 최대 처리량과도 관련이 있다. 그러나 응답 시간은 캐쉬의 크기에 따라서 차이를 나타내는 데에 비하여 최대 처리량은 캐쉬의 크기가 증가하더라도 응답 시간에서만큼 차이를 보이지는 않는다. 이는 부하 균등 반출 알고리즘이 기본적으로 선형 한계 스케줄링 알고리즘을 사용하기 때문에 시스템의 부하가 커지면 캐쉬의 크기가 증가하더라도 디스크로의 입출력 요구의 수는 일정비율로만 증가되기 때문이다.

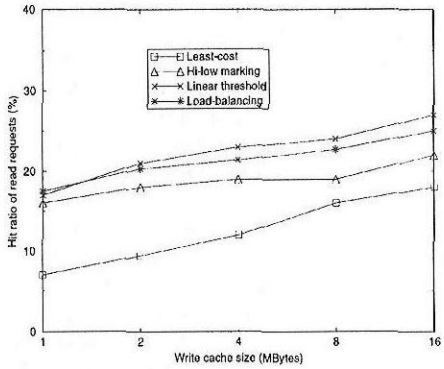

그림 7 쓰기 캐쉬의 크기에 따른 읽기 요구의 적중률

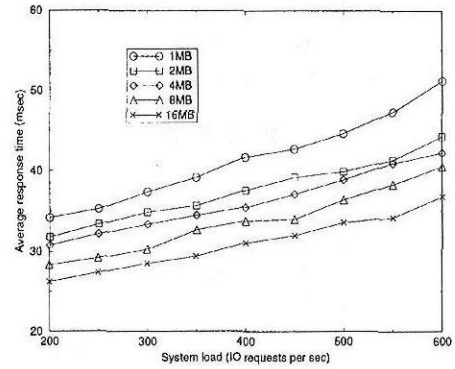

그림 9 쓰기 캐쉬의 크기에 따른 응답 시간의 변화

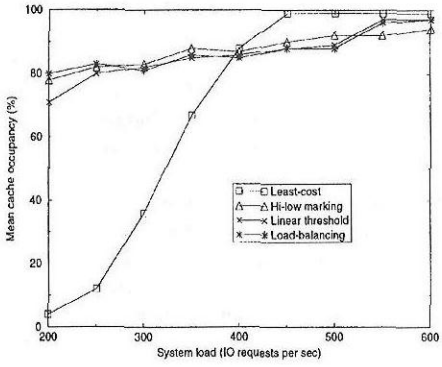

그림 8 부하에 따른 캐쉬 점유율

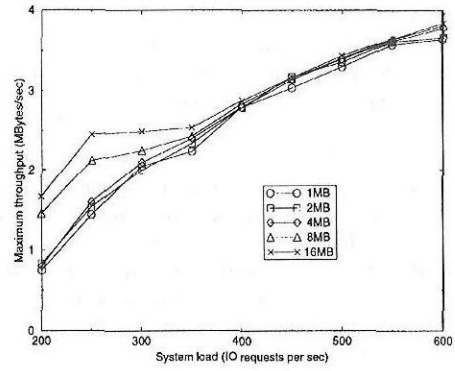

그림 10 쓰기 캐쉬의 크기에 따른 최대 처리량

5. 결 론

부하 균등 반출 알고리즘은 RAID 5의 디스크배열을 이루는 각 디스크로의 입출력 부하가 불균등하게 분포되는 것을 방지하고, 디스크간의 큐 길이의 편차를 줄임으로써 부하가 집중된 일부 디스크에서의 대기 지연으로 인하여 호스트 입출력 요구들의 처리시간이 증대되는 문제를 줄일 수 있다 본 논문에서는 기존의 반출 알고리즘에 부하 균형 요소를 부가한 새로운 RAID 5 모델을 구성하고 모의 실험을 통하여 부하 균등 반출 알고리즘이 RAID 5의 성능을 효과적으로 증대시킬 수 있음을 증명하였다 특히 기존의 디스크 성능 향상에 관련된 연구들이 대부분 디스크 캐싱 알고리즘이나 디스크 스케줄링 알고리즘들을 단일 디스크 관점에서 접근하였던 것에 비하여 부하 균등 반출 알고리즘은 기존의 성능 향상 기법들을 디스크배열 차원에서 접근하였다. 따라서 부하 균등 반출 알고리즘은 단일의 디스크 문제에서는 고려되지 않았던 디스크간 부하 균등화 문제를 주로 다루었으며 그 결과 RAID 5의 입출력 성능을 상당히 향상시킬 수 있음을 보였다.

참 고 문 헌

[1] M G. Baker, et al, "No-Volatile Memory for Fast, Reliable File Systems", *Proceedings of the 5th International Conference on Architectural Support for Programming Languages and Operating Systems*, Oct., 1992.

[2] P. Biswas, K. K. Ramakrishnan, and D. Towsley, "Trace-Driven Analysis of caching Policies for Disks", *Proceedings of the 1993 ACM SIGMETRICS conference on Measurement and Modeling of Computer Systems*, May 1993, pp. 13-23.

[3] P. M. Chen, "Striping in a RAID Level 5 Disk Array", *Proceedings of the 1995 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, May 1995, pp. 136-145.

[4] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz and D. A. Patterson, "RAID : High Performance, Reliable Secondary Storage", *ACM Computing Surveys*, Vol. 26, No 2, June 1994, pp. 145-188.

[5] G. R. Ganger, B. R. Worthington, R. Y. Hou, and Y. N. Patt, "Disk Arrays : High-Performance, High Reliability Storage Subsystems", *IEEE Computer*, Vol. 27, No. 3, March 1994, pp. 30-36.

[6] R. P. King, "Disk Arm Movement in Anticipation of Future Requests", *ACM Transactions on Computer Systems*, Vol. 8, No. 3, Aug. 1990, pp. 214-229.

[7] J. Menon and D. Mattson, "Performance of Disk Arrays in Transaction Processing Environments", *Proceedings of the 12th International Conference on Distributed Computing Systems*, June 192, pp. 302-309.

[8] J. Menon and J. Cortney, "The Architecture of a Fault-Tolerant Cached RAID Controller", *Proceedings of the 20th Annual International Symposium on Computer Architecture*, May 1993, pp.76-86.

[9] J. Menon, "Performance of RAID 5 Disk Arrays with Read and Write Caching", *Distributed and Parallel Databases*, Vol 17, No 1-2, 1993, pp. 129-139.

[10] D. A. Patterson, P. M. Chen, G. A. Gibson and R. H. Katz, "Introduction to Redundant Arrays of Inexpensive Disks (RAID), *IEEE CompCon '89*, Feb. 1989, pp. 112-117.

[11] D. A. Patterson, G. A. Gibson, and R. H. Katz, "A Case for Redundant Arrays of Inexpensive Disks (RAID)", *Proceedings of ACM SIGMOD*, June 1988, pp. 109-116.

[12] C. Ruemmler and J. Wilkes, "UNIX Disk Access Patterns", *Proceedings of the Winter 1993 USENIX Conference*, Jan. 1993, pp. 405-420.

[13] C. Ruemmler and J. Wilkes, "An Introduction to Disk Drive Modeling", *IEEE Computer*, Vol. 27, No. 3, March 1994, pp. 17-28.

[14] M. Seltzer, P. M. Chen, and J. Ousterhout, "Disk Scheduling Revisited", *Proceedings of the Winter 1990 USENIX Conference*, Jan. 19990, pp. 313-324.

[15] A. J. Smith, "Disk Cache - Miss Ratio Analysis and Design Considerations", *ACM Transactions on Computer Systems*, Vol. 3, No. 3, Aug. 1985, pp. 161-203.

[16] J. Wilkes, R. Golding, C. Staelin, and T. Sullivan, "The HP AutoRAID hierarchical storage system", *15th Symposium on Operating System Principles*, Dec. 1995.

[17] A. Varma and Q. Jacobson, "Destage Algorithms for Disk Arrays with Non-Volatile Caches", *Proceedings of the 22th Annual International Symposium on Computer Architecture*, 1995, pp. 83-95.

[18] B. R. Worthington, G. R. Ganger, and Y. N. Patt, "Scheduling Algorithms for the Modern Disk Drives", *Proceedings of the 1994 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, May 1994, pp. 241-251.

장 윤 석

1988년 서울대학교 자연과학대학 물리학과 학사. 1990년 서울대학교 대학원 컴퓨터공학과 공학석사. 1992년 서울대학교 대학원 컴퓨터공학과 박사과정 수료. 1994년 ~ 현재 대진대학교 컴퓨터공학과 전임강사. 관심분야는 컴퓨터 구조, 마이크로컴퓨터, 성능 평가.

김 종 상

1960년 서울대학교 공과대학 전자공학과 학사. 1965년 서울대학교 공과대학 전자공학과 석사. 1975년 서울대학교 공과대학 전자공학과 박사. 1979년 ~ 현재 서울대학교 공과대학 컴퓨터공학과 교수. 1986년 ~ 1988년 한국정보과학회 회장. 1992년 ~ 현재 서울대학교 컴퓨터신기술 연구소 소장.