特輯

고성능 입출력시스템 구조의 기술 동향

金 宗 鉉
延世大學校 電算學科

Ⅰ. 서 론

지난 수년간 프로세서의 속도가 계속 상승함에 따라 컴퓨터시스템의 프로세싱 속도(processing speed)가 급속히 상승하여 왔으나, 입출력 성능은 그와 보조를 맞추지 못하고 있다. 즉, 입출력 서브시스템의 성능과 프로세싱 속도간에는 매우 큰 불균형이 존재하고 있다. 프로세싱 속도와 입출력 속도간의 그러한 차이때문에, 시스템이 작업을 처리하는 능력은 궁극적으로 입출력이 얼마나 빨리 수행될 수 있느냐에 따라 결정된다. 비록 프로세서들이 작업들을 상당히 빨리 처리한다고 하더라도, 만약 입출력 동작들이 순차적으로(serially) 일어날 수 밖에 없다면 전체 프로세싱 능력은 이러한 입출력 병목에 큰 영향을 받을 수 밖에 없다는 것이다.[1] 특히, 다수의 프로세서들이 포함되는 병렬컴퓨터나 대규모의 데이타들이 저장되는 멀티미디어 정보처리시스템들에 있어서는 입출력 성능이 전체 시스템 성능에 주는 영향은 더욱 커지게 된다.

입출력 대역폭(I/O bandwidth)과 프로세싱 속도간의 균형의 중요성은 Kung[2] 에 의하여 지적된 바가 있는데, 그의 논문에 따르면 몇몇 응용문제들의 처리에 있어서는 입출력 속도때문에, 프로세서에 기억장치가 추가되더라도 전체 성능이 개선될 수 없다는 것이다. 그러한 응용들을 위하여는 입출력 능력이 시스템의 프로세싱 능력만큼 향상되어야 한다는 것은 필수적이다. 입출력 서브시스템의 성능이 그 이외의 시스템 요소들(프로세서, 기억장치 등)의 속도와 같은 비율로 고속화되지 않고는 선형적인 속도향상(linear speedup)을 얻을 수가 없다. 그러므로 시스템 전체의 성능 향상을 얻기 위하여는 입출력 서브시스템의 성능도 반드시 개선되어야 한다.

입출력 서브시스템은 대부분의 경우에 시스템 내에서 가장 느린 요소이다. 만약 이러한 경향이 계속된다면, 입출력 서브시스템은 새로운 구조를 가진 대부분의 시스템에서 성능 병목이 될 수 밖에 없게 될 것이다. 특히, 입출력 병목은 고성능 컴퓨터시스템에서 빈번히 나타나는데, 이러한 시스템에서는 프로세서와 기억장치 기술은 이미 거의 한계에 도달했다고 볼 수 있다. 최근에는 중형급 및 대형시스템에서도 입출력 속도와 계산 속도의 비율이 급속히 높아져 감에 따라 시스템 균형을 위한 설계자들의 관심이 입출력으로 많이 옮겨져 오고 있다.

입출력 장치들 중에서도 프로세서의 프로그램 실행 속도에 직접적으로 영향을 주는 장치는 디스크 드라이브이다. 반도체 기억장치와는 달리, 디스크 드라이브 에는 기계적 장치들이 많이 포함되어 있기 때문에 속도의 개선은 반도체 기술이 나 전자회로의 발전 속도에 비하여 현저히 느리게 발전해 오고 있다. 수치적으로 보면, 지난 30년간 CPU와 주기억장치의 속도는 수백배 증가하였지만, 디스크 액세스 시간은 수십배 정도 밖에 개선되지 않았다. 현재 디스크들은 주기억장치 속도의 1/10000 정도이며, 그 차이는 점차 더 커질 것으로 예상된다. 이러한 문제는 CPU의 계속적인 속도 향상때문에 더욱 심각해지고 있으며, 이에 따라 디스크 서브시스템 성능을 개선해야 할 필요성이 더욱 높아지고 있다.

본 稿에서는 최근 다중처리 구조를 가진 중형급 이상의 컴퓨터시스템에서 입출력 성능을 향상시키기 위하여 연구되고 있는 기술적인 사항들을 조사하고 분석하고자 한다. 제2장에서는 고성능 입출력시스템을

위한 기술에 대하여 전반적으로 논하고, 그들 중에서 가장 널리 채택되고 있는 기술인 디스크 배열 기술과 RAID 구조에 대하여 제3장에서 설명하며, 제4장에서는 그외의 최근 연구 동향에 대하여 소개하고자 한다. 이러한 제반 기술에 대한 전망과 결론은 제5장에서 논한다.

Ⅱ. 입출력 성능 향상을 위한 기술

이 장에서는 먼저 입출력 성능을 향상시키기 위하여 현재 고성능 시스템들(특히, 다중프로세서 시스템 및 슈퍼컴퓨터)에서 채택되었거나 연구되고 있는 여러 가지 방법들을 전반적으로 살펴보고자 한다.

1. 입출력 시스템의 구성

일반적으로 다중처리 구조를 가진 시스템에서 입출력 병목의 요인들로는 입출력 채널의 속도, 입출력 버스의 대역폭 및 디스크 드라이브의 속도 등으로 볼 수 있다. 최근 중형급 컴퓨터에서 일반적으로 사용되는 입출력 구조는 그림 1과 같다. 이와 같은 구조에서는 입출력 버스의 병목 현상이 발생하기 때문에 여러개의 입출력 프로세서와 입출력 버스들이 접속되는 것이 보통이다.

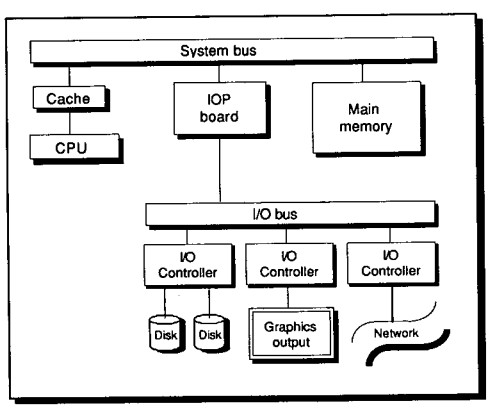


그림 1. 중형급시스템의 입출력 시스템 구조

비교적 많은 수의 입출력 장치들이 연결되는 메인프레임급들 중에서 IBM 3090 시스템의 대략적인 입출력 구조에서는 그림 2와 같이 복잡한 상호연결 방식이 사용되고 있다. IBM 시스템에서는 입출력장치와 주기억장치 사이에 데이타와 제어 정보를 전송하는 채널(channel)이 여러개 있는데, 각 채널은 프로세서, 버스 인터페이스 회로 및 케이블들로 구성되며 속도 정합 버퍼(speed-matching buffer)라고 불리기도 한다. 같은 통로를 사용하는 디스크들의 집합은 스트링(string)이라고 하며, 이들의 동작은 스트링 제어기(string controller)에 의하여 제어된다.

채널은 CPU로부터 보내진 입출력 요구를 해독하여 해당 storage director로 명령을 보낸다. 이때 채널은 입출력 명령어 코드를 시스템 공유 기억장치로부터 읽어온다. Storage director(device controller)는 디스크의 액세스 동작을 제어하 고 데이타 전송을 관장한다. 그 외에도 storage dierctor는 오류 검출및 정정 동작과 직렬/병렬 데이타 형식의 전환 동작도 수행하기도 하는데, 이 기능을 가지지 않은 경우에는 스트링 제어기가 대신 그 기능을 수행한다.

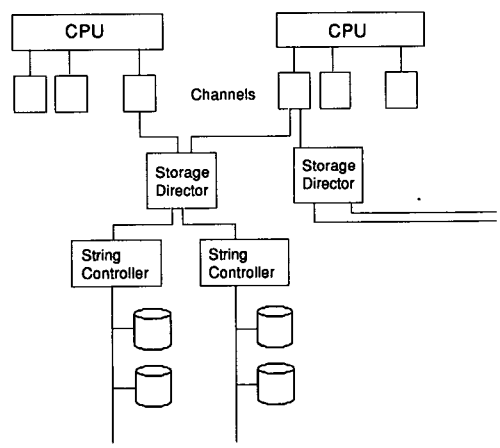


그림 2. IBM 메인프레임의 입출력 구조

입출력이 빈번하지는 않으나 대규모의 데이타 액세스가 발생하는 슈퍼컴퓨터의 입출력 구조의 한 예로서, 그림 3은 CRAY Y-MP의 경우를 보여주고 있다. CRAY I/O system(IOS)은 시스템 모델에 따라 약간의 차이는 있지만, 일반적으로 2개 내지 4개의 입출력 프로세서(IOP)들을 가지고 있다. 그중에서 MIOP는 시스템 운영자 터미날에 접속되어, 주로 시스템 유지보수 기능을 수행한다. XIOP는 블럭 멀티플렉싱(block multilexing)을 지원하며, 비교적 저속의 장치들의 제어를 담당한다. BIOP와 DIOP는

디스크와 같은 고속 장치들의 제어에 적합하도록 설계되어 있으며, 이들 각각에는 최대 4개씩의 디스크들이 디스크 제어 유니트(DSU)들을 통하여 접속된다. BIOP와 DIOP는 각각 3개씩의 DCU가 연결될 수 있으므로 IOS 當 최대 24개의 디스크들이 접속될 수 있고, Y-MP 시스템은 2개까지의 IOS들이 연결될 수 있으므로 전체적으로 48개의 디스크들을 가질 수 있다.

각 IOP는 지역 메모리를 가지고 있고 버퍼 메모리를 다른 IOP들과 공유한다. IOP는 IBM 시스템에서의 채널과 비슷한 기능을 하지만, 자신이 수행할 입출력 프로그램 코드들을 지역 메모리에 가지고 있는 것이 다르다. IOP의 지역 메모리는 고속 통신 인터페이스(CRAY에서는 이것을 채널이라고 함)를 통하여 DCU에 접속된다. 따라서, 데이타는 지역 메모리와 100MB/s 속도의 채널을 통하여 디스크와 주 기억장치 사이에 전송된다.

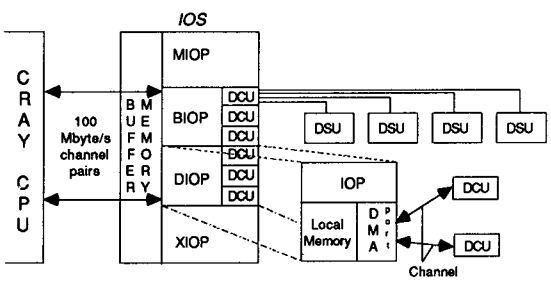


그림 3. CRAY Y-MP의 입출력시스템 구조

2. 요소 기술들

시스템 차원에서 여러개의 채널들과 IOP들이 접속되더라도, 입출력 요구의 수가 증가하고 디스크의 속도가 제한되면 입출력 성능 향상도 한계에 이르게 된다. 따라서, 주기억장치의 용량을 증가시켜 입출력 요구의 횟수를 감소시키는 것도 최근의 반도체 기술 발전 동향으로 보아서 어느 정도 가능한 것이라고 할 수는 있지만, 근본적인 해결책으로 볼 수는 없다. 현재 고려되고 있는 방법들로는 기존의 디스크 구조에 다중-헤드(multiple-head)를 부착하는 방법과 병렬 전송 채 널을 구성하거나 디스크의 저장 밀도를 증가시켜서 헤드가 움직이는 거리를 단축 시키는 방법 등이 있다. 또 다른 방법으로는 비교적 느리지만 가격이 저렴한 메모리 칩들을 이용한 solid state disk를 사용하는 방법도 사용되고 있다 (예: CRAY 슈퍼컴퓨터).

이와 같이 디스크 재질이나 디스크서브시스템내의 구조를 개선하는 방법 이외에도 디스크와 주기억장치 사이에 디스크 캐쉬(disk cache)를 삽입하여 디스크 액세스 요구들 중의 일부분에 대하여는 실제로 디스크가 구동되지 않고 데이타를 전송해줄 수 있도록 하는 방법도 널리 연구되고 있다. 디스크 캐쉬는 IOP 내 또는 디스크 제어기내에 위치한다. 또한 큐에서 대기하는 디스크 요구들 중에서 현재의 디스크 헤드의 위치 변동을 최소화할 수 있는 요구를 선별하여 액세스 순서를 정해주는 디스크 스케줄링(disk scheduling) 기법도 사용될 수 있다. [5]

최근 가장 관심을 가질만한 것은 디스크 제어기 또는 입출력 버스에 여러개의 디스크들을 연결하고, 저장될 데이타 블럭들을 분산 저장시켜서 비동기적으로 동시 액세스(concurrent access)가 가능하도록 하는 디스크 인터리빙(disk interleaving) 즉, 디스크 스트라이핑(disk striping) 방식도 많이 연구되고 있다. 이와 유사한 방법으로서 저렴한 가격의 디스크들 여러개가 배열(array) 구조 로 연결되어 한개의 유니트내에 장착된 RAID(Redundant Array of Inexpensive Disks)가 출현하여 많은 관심을 끌고 있다. RAID에서는 모든 디스크들이 동기적 으로 액세스(synchronous access)되고, 데이타 전송도 병렬로 이루어진다. 또한 여분의 디스크들을 사용하여 신뢰도와 신속한 오류회복 기능을 가지고 있는 것이 특징이다.

디스크 결함에 의한 데이타 손실 또는 시스템 안정성의 보완을 위한 기술로는 데이타 블럭을 여러개의 디스크들에 중복 저장(복사)하는 디스크 샤도윙(disk shodowing) 방법과 두개의 디스크들에 동시에 저장하는 디스크 미러링(disk mirroring) 방법 등이 있다. 이 기술들은 결함허용 능력을 중요시하는 시스템에 서 독립적인 디스크들을 이용하여 구현되기도 하고, RAID와 같이 한개의 유니트 내에서 구현되기도 한다.

Ⅲ. 디스크시스템의 성능 향상 기술

제2장에서 개략적으로 설명된 기술들 중에서 시스

템 성능에 가장 큰 영향을 주는 부분은 디스크시스템이기 때문에 이에 대하여는 좀더 상세히 설명하기로 한 다. 여기에서는 데이타를 디스크에 저장할 때 인터리빙(또는 디클러스터링)시킨 구조를 디스크 배열(disk array)이라고 부르며, 그 범주에 들면서도 특수한 경우로서 한개의 유니트로 패키지화된 RAID와는 구분하여 별도로 설명하기로 한다.

1. 디스크 배열

디스크 배열은 여러개의 디스크들을 하나의 그룹으로 구성하여 액세스 병렬성을 이용하면서도 사용자에게는 한개의 논리적 디스크 유니트로 보이게 하는 기술 이다. 이 방법의 장점은 한개의 입출력 요구에 대하여 몇배의 디스크 대역폭을 얻을 수 있다는 것과 여러개의 입출력 요구들을 병렬로 처리할 수 있다는 점이다.

디스크 배열은 연결 구조와 운용 방식에 따라 몇가지 종류로 분류될 수 있으며, 비교적 새로운 개념이기 때문에 용어(terminology)상의 혼란도 있다. Kim[3] 은 디스크 배열 구조를 동기식(synchronous)과 비동기식(asynchronous)으로 구분하여 설명하고 있고, Salem[4] 은 디스크 스트라이핑(disk striping)이라는 용어를 사용하였다. 또한, Livny et al[5] 은 디클러스터링(declustering)이라는 표 현을 쓰고 있고, Reddy[1] 는 인터리브드 디클러스터링(interleaved declustering)이라는 새로운 구조를 소개하였다.

인터리빙 방식에서는 각 데이타 블럭이 여러 조각으로 나누어져서 연속적으로 배열되어 있는 디스크들에 순서대로 저장된다. 따라서, 여러개의 디스크들에서 동시 액세스(seek 및 rotation)가 가능해지기 때문에 한개의 블럭을 액세스하는 데 걸리는 시간을 줄일 수 있을 뿐만 아니라 병렬 전송이 가능해짐에 따라 데이타 전송 속도도 향상시킬 수 있다. 이 방식은 이미 몇몇 슈퍼컴퓨터에서도 사용된 바가 있으며, 스트라이핑이라는 용어와 동일한 의미를 가진다.

반면에, 디클러스터링은 한개의 화일을 구성하고 있는 블럭들을 여러개의 디스 크들에 분산 저장하는 방법이다. 이 방법의 목적은 블럭 단위의 동시 입출력 (concurrent I/O)을 가능하게 해주는 것과 다중-블럭(multi-block) 입출력 요구를 병렬로 처리하는 것이다. 다시 표현하면, 분산 저장의 단위가 인터리빙에서는 서브-블럭(sub-block), 바이트 또는 비트 단위이지만, 디클러스터링에서는 블럭 단위라는 점이다.

이러한 디스크 배열 조직들을 구분하는데 도움이 되는 세가지 기술적인 사항들을 보면 다음과 같다.[9]

1) 인터리빙의 단위(degree of interleaving): 배열내의 각 디스크들에 분산 저장되는 단위가 비트, 바이트, 블럭, 트랙(track) 또는 실린더(cylinder) 중의 어느 것인가 ?

2) 디스크 팔(disk arm)의 독립성: 배열을 구성하고 있는 디스크들의 구동장치(actuator: arm의 위치를 조정)들이 하나의 단위로서 동시에 움직이는가 혹은 서로다른 위치의 트랙으로 독립적으로 이동하는가 ?

3) 회전(Ratation)의 독립성: 디스크의 회전축(spindle)들이 동시에 회전하여 서 모든 디스크들의 헤드 아래에 동일한 번호의 섹터들이 위치하게 되는가 혹은 회전축들이 서로 독립적으로 회전하는가 ?

그림 4는 위의 세가지 사항들에 따라 디스크 배열 방식들을 구분한 것이다.

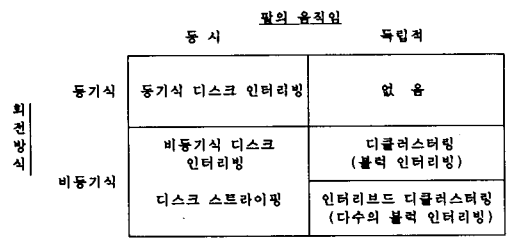


그림 4. 디스크 배열의 분류

이와 같은 배열 방식에 따라 디스크 서브시스템을 구성하는 방법들을 종합하면, 디스크들을 동기화(synchronization)시키는 것과 데이타를 인터리빙하는 것으로 요약될 수 있다. 이 두가지가 적절히 조합되면 여러가지 구성이 가능하며, 그들간의 상관 관계와 그에 따른 특성들을 분석해보면 다음과 같다.

디스크를 동기화시키는 것은 화일 읽기 시간(file read time)이 I/O 서비스 시간의 많은 부분을 차지하는 경우에 유용하다.[3] 이 방식에서 각 화일은 디스크들 간에 바이트 단위(bytewise)로 인터리브된다. 또한, 모든 디스크들의 헤드들 이같은 장소에 위치하도록 동기화된다. 따라서, 모든 디스크들이 같이 동작하여 하나의 큰 디스크로서의 기능을 하게 됨으로써, 전송율(transfer rate)이 m배가 되고, 용량은

한 디스크 용량의 m배가 된다. 이 경우에 seek time과 latency time은 변하지 않는다 (즉, 한개의 디스크에 대한 seek 및 latency time만 소요된다.) 전송시간이 I/O 서비스 시간의 많은 부분을 차지하는 경우에 이 방식은 높은 성능을 보여준다. 또한, 모든 디스크들이 같이 동작하기 때문에(상호 연관 되어 있기 때문에) 각 디스크는 동일한 요구율(request rate)을 가지며, 따라서 모든 디스크들에 균형된 부하가 걸리게 된다. 각 디스크의 이용율은 더 높아지는 데, 그 이유는 각 요구를 모든 디스크들이 함께 처리하기 때문이다. 따라서, 이 방식은 큰 화일들이 전송될 필요가 있을 때 유용하다. 본 고에서는 이 방식을 채택하는 시스템을 동기화된 시스템(synchronized system)이라고 정의한다. 현재 상용화된 몇가지 디스크 시스템들이 이 구조를 가지고 있다. [6,7]

여기서 한개의 유니트를 구성하고 있는 동기화된 디스크의 수를 동기화 정도(degree of synchronization: ds)라고 정의한다. 예를 들어, 16개의 디스크들을 가지며 ds=2인 시스템은 각각 2개의 동기화된 디스크들을 가진 8개의 유니트들로 구성된다. 만약 디스크 시스템이 m개의 디스크들로 구성되어 있다면, ds는 1부터 m까지 변할 수 있다. Kim [3] 은 ds = m인 경우에 대하여 성능을 연구하였고, Reddy [1] 는 여러가지 ds값에 대하여 특성을 분석하였다. 비교를 위하여, 모든 디스크들이 독립적인 유니트들로 구성되고, 각 화일이 한개의 디스크에 위치하는 (각 화일이 나누어져 여러 디스크에 분산 저장되지 않는) 시스템을 여기서는 전통적인 시스템(traditional system)이라고 정의한다.

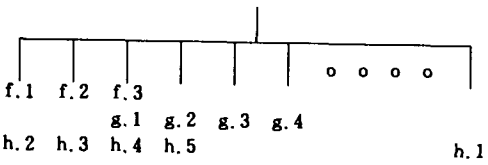
디스크 스트라이핑 [4] 또는 디스크 디클러스터링 [5] 에서는 화일을 블럭 단위로 인터리빙함으로써 한 화일내의 여러 블럭들을 여러개의 디스크들로부터 병렬로 읽거나 쓸 수 있다. 화일을 인터리빙하는 한가지 방법은 다음과 같다: 만약 블럭(i)가 디스크(j)에 저장되어 있다면, 블럭(i+1)은 디스크(j+1)에, 블럭(i+2) 는 디스크(j+2)에 각각 저장된다. 만약 I/O 요구가 한개의 블럭에 대한 것이라면, 그 요구는 그 블럭을 가진 디스크의 큐로 들어간다. 만약 다중-블럭 요구 (multiple-block request)가 디스크 시스템에 도착한다면, 요구는 여러개의 단일-블럭 요구(single-block request)들로 분리되어서 해당되는 디스크들의 큐로 들어간다. 이 결과로 디스크의 동시 액세스(concurrent access)가 가능해진다. 이 구조는 여러개의 단일-블럭 요구들도 동시에 처리되도록 해준다. 데이타를 인터리빙함으로써, 디스크 시스템이 다양한 요구율에 대하여 계속적으로 균형이 유지 될 수 있다. 이 구조에서 seek 및 latency 오버헤드들은 전송되는 각 블럭에 대하여 발생하기 때문에 서비스 시간은 결과적으로 증가된다. [5] 의 연구 결과에 따르면, 디클러스터링 방식은 전통적인 시스템에서의 대기시간(queueing time)의 변화를 줄여줌으로써 서비스 시간을 효과적으로 조절하여 고성능을 얻게 해준다. 한개의 화일을 여러개의 블럭들로 나누어서 여러개의 디스크들에 저장하는 시스템을 여기서는 디클러스터링 시스템이라고 정의한다.

또한, 한개의 화일에 대하여 이루어질 수 있는 동시 액세스의 수를 디클러스터링 정도(degree of declustering: dd)라고 정의한다. 만약 dd=1이면, 어떤 주어진 시간에 한 화일내의 한개의 블럭만 액세스될 수 있다. dd=m이라면, 화일의 m개 블럭이 동시에 읽혀질 수 있다. 여기서는 그러한 시스템을 완전 디클러스터된 시스템(fully declustered system)이라고 부른다.

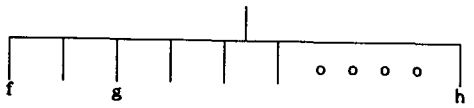
비록 데이타 디클러스터링과 디스크 동기화는 같은 목표를 달성하기 위한 두가지 방법이지만, 몇가지 근본적인 차이가 있다. 동기화된 시스템은 화일을 한개의 유니트에 저장하며, 따라서 한 화일내의 여러 블럭들에 대한 동시 액세스(읽기/ 쓰기)는 허용하지 않는다. 두 시스템들의 요구 응답시간은 대기시간상의 차이와 주어진 요구에 대한 서비스에 있어서의 동시성때문에 전통적인 시스템 보다는 더 우수하다. 동기화된 시스템(synchronized system)은 화일 액세스의 지역성(locality)을 이용한다. 즉, 큰 블럭의 전송시에는 seek와 latency 오버헤드가 한번만 포함된다. 디클러스터된 시스템(declustered system)에서는 큰 데이타 블럭에 대한 액세스를 여러개의 작은 데이타 블럭에 대한 많은 액세스들로 분산시 킴으로써 디스크 시스템에서의 화일 액세스의 지역성을 배제시킨다. 완전 동기화 된 디스크시스템(fully synchronized disk system)은 SIMD 방식에 대응되고, 완전 디클러스터된 시스템(fully declustered system)은 MIMD 방식에 대응된다.

이 개념들을 이용하여 구성할 수 있는 시스템 구조들에는 (1) 여러가지 인터리빙 정도(degrees of interleaving)를 가진 동기화된 시스템, (2) 여러가지 디클러스터링 정도(degrees of declustering)를 가진 디클러스터된 시스템, 및 (3) 여러가지 동기화

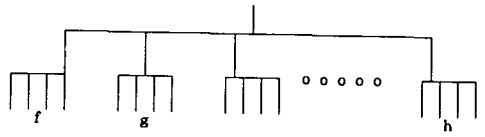
및 디클러스터링 정도들을 가진 디클러스터-동기화된 시스템 (declustered-synchronized system)이 있다. 이 구조들에는 완전 디클러스터된 시스템, 완전 동기화된 시스템, 및 전통적인 시스템도 포함될 수 있다. 이들 중 의 몇가지는 [8] 에서 연구된 바가 있다. 몇가지 디스크 조직들이 그림 5에 보여져 있는데, f.m은 화일 f의 m 번째 블럭을 가르키고, f는 전체 화일을 나타낸다. 그림 5에서 사용된 표기 방식을 따르면, 디스크 조직은 세 변수들 (dd,ds) 및 m에 의하여 표현될 수 있다. 이 표기를 이용하면, 전통적인 시스템은 (1,1)로, 완전 디클러스터된 시스템은 (m,1)로, 완전 동기화된 시스템은 (1,m)으로 각각 표현할 수 있으며, 여기서 m은 시스템 내의 디스크의 수를 나타낸다. 세 변수들 dd, ds 및 m을 이용하면 어떤 조직도 표현할 수 있다.



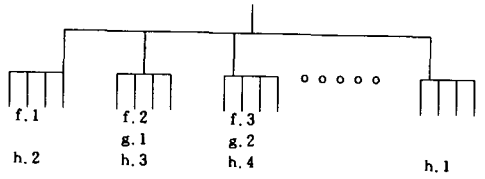
(a) 클러스터된 시스템: (16,1)



(b) 전통적 시스템: (1,1)



(c) 동기화된 시스템: (1,4)



(d) 혼합 시스템: (4,4)

그림 5. 디스크 서브시스템의 구성도

2. RAID 기술

이 절에서는 많은 수의 값싼 디스크들을 배열 구조로 연결하여 신뢰도와 전송 대역폭을 향상시킨 RAID 기술에 대하여 설명하고자 한다.[8,9] 이 기술은 최근에 출현하여 중형급 컴퓨터시스템에서 사용이 급속히 증가하고 있다. 여기서는 먼저 RAID의 출현 배경에 대하여 알아보며, 여러가지 RAID 구조들을 분석하고 각각의 특징을 설명한다.

1) RAID 출현의 배경

1956년에 개발된 RAMAC 디스크 드라이브로부터 시작된 하드 디스크 기술은 오늘날의 Winchester 드라이브로까지 발전해 오는 동안에 플레터(platter)의 크기가 계속하여 줄어들고 있다. 메인프레임급에서는 14인치 및 10.5인치 디스크들이 주로 사용되고 있고, 미니급 컴퓨터에서는 8인치 디스크들이, 워크스테이션 및 PC에서는 5.25인치와 3.5인치 디스크들이 사용되고 있다. 크기가 계속 감소함에 따라 메가바이트당 가격이 떨어지고 시스템내의 디스크 용량이 급속히 증가하고 있다. 이러한 추세는 소형 디스크에서 특히 두드러지게 나타나고 있고 낮은 전력 소모도 그 요인이 되고 있다. 그림 6은 지난 10년간의 디스크 가격 동향을 보여 주고 있는데, 소형 디스크들의 가격이 대형 디스크들 보다 더 급속히 떨어지고 있음을 볼 수 있다. 1988년도에는 5.25인치 디스크가 가격을 주도하였으나, 1992년도에는 3.5인치 디스크가 주도하기 시작하였다.

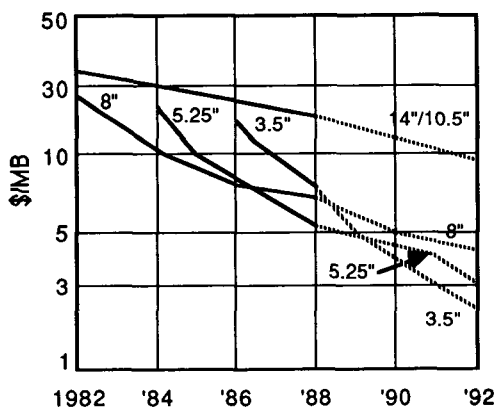


그림 6. 디스크 가격의 동향

디스크 입출력의 고성능화를 위한 핵심 요소는 액

세스 시간을 줄이는 것과 시스템과 장치들간의 데이타 전송 대역폭을 높이는 것이다. 이것은 많은 수의 값싼 디스크들을 배열 구조로 연결하여 동기적 혹은 비동기적 액세스가 가능하도록 하고, 여러개의 전송 채널들을 제공함으로써 가능해진다. 그러나, 이 방법의 주요 단점은 결함 허용 능력이 낮아진다는 것이다. 즉, 데이타 블럭을 여러개의 디스크들로 분산 저장함에 따라 그 중의 어느 한 디스크에만 결함이 발생하여도 전체 데이타 블럭이 손상되고 전체 디스크 유니트의 사용이 불가능해진다. 일반적으로 소형 디스크가 더 큰 디스크와 비교할 때 비슷한 수준의 신뢰도를 가지고 있지만, 여러개가 같이 연결되면 신뢰도가 떨어진다. 결함율(failure rate)이 일정하다고 가정할 때, 디스크 배열의 MTTF(mean time to failure)은 다음과 같다.

$$\frac{\text{단일 디스크의 MTTF}}{\text{배열내의 디스크의 수}}$$

예를 들어, MTTF가 30000 시간인 디스크 100개를 배열로 구성한 경우에 MTTF는 30000/100 = 300 시간이 된다. 즉, 3년 정도에 한번씩 결함이 발생하는 디스크가 배열로 연결되어 하나의 유니트로서 동작하는 경우에는 2週日 이내에 결함이 발생하게 된다. 만약 1000개의 디스크들로 배열을 구성하면 30시간, 즉 하루정도에 한번씩 결함이 발생한다는 계산이 나온다.

이러한 문제를 보완하기 위하여 결함 허용 능력을 높이면서도 데이타 액세스 속도를 그대로 유지시키기 위한 노력들이 많이 이루어지고 있다. 그에 따라 여러 가지의 RAID 조직들이 제안되었으며 각각은 용도와 신뢰도 요구 정도에 따라 적절히 선택될 수 있다. 신뢰도 향상을 위하여 일반적으로 사용되고 있는 방법으로는 결함 발생시에 원래의 데이타를 회복시키는 데 필요한 정보를 저장하고 있는 별도의 디스크와 결함이 발생한 디스크를 대치시킬 여분의 디스크들을 추가적으로 포함하는 것이다. 이에 따라 입출력 대역폭과 중복 정보의 저장 공간 사이에 조정(tradeoff)이 필요하며, 각 RAID 조직들은 조정의 정도가 서로 다르다.

이와 같이 신뢰성 보장을 위한 정보를 저장하기 위하여 추가되는 디스크를 "검사 디스크(check disk)"라고 부르며, 동작 원리는 다음과 같다. 배열내의 한 디스크에 결함이 발생하면, 그 디스크의 사용은 중단되고 검사 디스크에 저장된 정보를 이용하여 원래의 정보가 신속히 재구성되어 여분의 새로운 디스크에 저장된다. 이 과정에 걸리는 시간은 MTTR(mean time to repair)이라고 부른다. 운용 시간(run time)동안에 결함 디스크가 여분의 다른 디스크로 대치되는 것은 전자 회로에 의하여 자동적으로 일어나며, 주기적으로 시스템 운용자(human operator)가 모든 결함 디스크들을 새로운 디스크들로 대체시켜 준다.

RAID에서 전체 디스크들을 모두 하나의 배열로서 구성하는 것은 신뢰도 측면에서 바람직하지 못하므로, 몇개의 디스크들로 구성된 여러개의 그룹으로 나누어진다. 이와 관련하여, RAID에 관한 앞으로의 설명에서는 다음과 같은 표기들을 사용하기로 한다.

D = 데이타가 저장되는 디스크들의 전체 수 (검사 디스크는 제외).
G = 한 그룹내의 데이타 디스크들의 수 (검사 디스크는 제외).
C = 한 그룹내의 검사 디스크들의 수
n_G = D/G = 그룹들의 수

RAID 조직들을 내부 구조와 운용 방식에 따라 분류하고 각각의 특성을 분석해 보면 다음과 같다.

2) RAID-1

이 조직은 디스크의 신뢰성을 높이기 위하여 전통적으로 사용되던 것으로서, 그림 7과 같이 구성된다. 이 방법은 가장 비용이 많이 드는데, 그 이유는 각 그룹이 한개의 데이타 디스크와 그 내용이 중복 복사되는 검사 디스크로 구성되며 (즉, G = 1, C = 1), 모든 쓰기 동작은 데이타 디스크와 검사 디스크로 동시에 이루어지기 때문이다. 따라서, 더 높은 신뢰도를 얻기 위하여 입출력 대역폭과 저장 용량의 희생이 매우 커진다.

즉, 전체 디스크들의 50%는 중복된 데이타 저장에 사용된다. 그러나 제어기와 입출력 통로(I/O pathway)가 각 디스크에 대하여 별도로 제공된다고 가정할 경우에, 읽기 동작의 대역폭은 쓰기 동작의 대역폭의 2배가 된다. 이 방식은 Tandem 컴퓨터에서 많이 사용되었는데, 이 컴퓨터에서 는 제어기들(controllers)과 입출력 통로들도 신뢰성을 위하여 중복되어 있다.

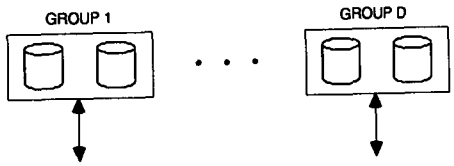


그림 7. RAID-1의 구성도

3) RAID-2

칩당 한비트씩 저장되는 DRAM 배열 구조에서 데이타의 오류 검출 및 정정을 위하여 해밍 코드를 사용하는 것과 마찬가지로, 디스크 배열에서도 비트 단위 인터리빙이 된 경우에는 충분한 수의 여분의 디스크들을 추가하여 비트 오류를 검출하고 정정할 수 있다. 이 개념이 사용된 구조가 RAID-2이다 (그림 8). 이러한 구조에서 데이타 디스크의 수를 G, 검사 디스크의 수를 C라 하면, G=10인 디스크 그룹에 대하여 C=4가 되며(40% 오버헤드), G=25인 경우에는 C=5(20% 오버헤드)가 된다. 이 경우에 검사 디스크에 따른 오버헤드는 그룹의 크기가 커질수록 감소한다는 것을 알 수 있다. RAID-2의 단점은 코드의 오류 정정 동작 때문에 그룹당 단한개의 입출력만 가능하다는 점이다.

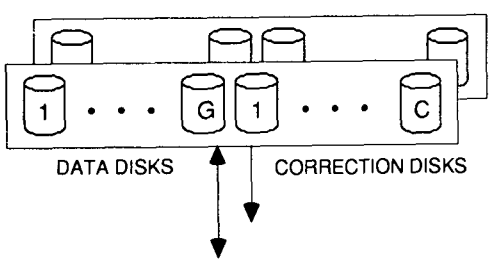


그림 8. RAID-2의 구성도

4) RAID-3

RAID-2에서 사용된 검사 디스크들은 오류 비트의 위치를 검출하기 위한 것인데, 사실상 이러한 오류는 디스크 인터페이스 회로에 의하여 제공되는 특수 신호 들이나 디스크에 저장된 추가적인 정보를 이용하여 검출이 가능한 경우가 많다. 다시 표현하면, 한 섹터내에서 발생하는 오류 비트는 디스크 제어기가 검출할 수 있다는 것이다. 일단 오류가 검출되기만 하면, 단한개의 패리티 비트만으로도 오류 비트 정정이 가능하다. 따라서, 검사 비트는 그룹당 한개만 있으면 되므로 오버헤드는 앞에서 본 예의 경우에 각각 4%와 10%로 줄어들게 된다. 이와 같이 RAID-2 구조를 그룹당 한개의 검사 비트로 변경한 것이 RAID-3이다.

RAID-3는 일반적으로 트랜잭션 처리 응용 보다는 과학계산 응용에 더 적합한 것으로 알려져 있다. [9] 상용화된 RAID-3로는 Maxtor社와 Micropolis社의 제품들이 나와있는데, 모두 동기식 5.25인치 SCSI 디스크로서 G=4, C=1의 조직을 가지 고 있다. [10] 이러한 유니트는 호스트 컴퓨터에게는 한 개의 논리적 디스크로 보이지만, 전송 대역폭과 용량이 각각 4배로 증가하였고 신뢰도도 높아졌다.

5) RAID-4

앞에서 소개된 조직들에서는 한 그룹당 한개의 입출력 채널만이 제공되기 때문에 전송 시간상의 잇점을 얻을 수가 없다. 또한 디스크들이 동기화되지 않은 경우에는 검색 및 회전 지연 시간을 감축하는 것도 불가능해진다. RAID-4에서는 그룹당 여러개의 입출력이 가능하도록 함으로써, 즉 전송 동작에 병렬성을 추가함으로써 전송율을 개선시켰다. 이 경우에는 논리적 전송 블럭을 여러개의 디스크들에 분산 저장시키지 않고 한개의 디스크에 저장한다. 이렇게 하면 읽기 동작시에 다른 디스크들은 액세스하지 않고도 오류 검출이 가능해진다. 즉, 데이타 저장 방식에 있어서 RAID-4가 RAID-3와 다른 점은 비트 혹은 바이트 단위의 인터 리빙이 아닌 섹터 단위의 인터리빙을 한다는 것이다.

이 경우에 쓰기 동작에서 검사(패리티) 정보를 다시 계산하는 과정이 더 복잡해 보이지만, 사실은 더 간단해진다. 새로운 패리티의 계산은 원래의 데이타 블럭과 패리티 블럭 및 새로운 데이타 블럭들간에 exclusive-OR 동작을 수행하면 된다. RAID-4에서는 쓰기 동작에 두개의 디스크를 사용하며 네번의 액세스(데이타와 패리티의 읽기 및 쓰기)가 필요하다. 따라서, 단일 디스크의 읽기 동작에는 병렬성을 구현하였지만, 쓰기 동작에 있어서는 패리티 디스크를 읽고 쓰는 동작이 필요하기 때문에 여전히 그룹당 한번으로 제한된다. 이 조직은 Salem과 Garcia-Molina에 의하여 제안되었다. [4]

6) RAID-5

RAID-5는 RAID-3를 개선한 것으로서, 검사 정보를 그룹내 모든 디스크들로 분산 저장하는 방식을 이용하였다 (그림 9). 그림에서 같은 行(column)에

속해 있는 디스크들이 하나의 그룹을 형성하고, 각 列(row)은 디스크 내의 섹터들을 나타낸다. 패리티 블럭들은 (G+1)번째 디스크에 저장되는 것이 아니라 그룹내의 모든 디스크들로 분산된다. 이때 배치 방법은 각 섹터 列에 한개의 패리티 블럭씩만 저장되도록 하는 것이다. 디스크의 수보다 디스크當 섹터의 수가 더 많은 경우에는 이러한 배치 방식이 여러번 반복될 것이다. 이 조직에서 列에 속한 데이 타와 패리티 블럭이 서로 다른 行에 저장된 경우에는 두번의 쓰기 동작이 병렬로 일어날 수 있다. 따라서, 최대 (G+1)/2번의 쓰기 동작이 동시에 수행될 수 있게 된다. 이 조직은 요구된 데이타의 크기에 관계없이, 또한 입출력 요구의 패턴 에 관계없이 좋은 성능을 보여준다.

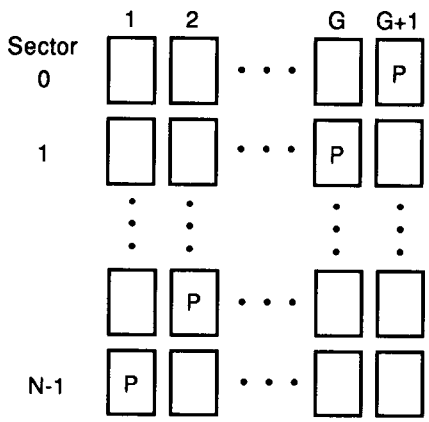


그림 9. RAID-5의 구성도

7) RAID-6

이 조직은 RAID-5의 디스크 배열을 3차원으로 구성하여 列에 대한 패리티 외에 行에 대한 패리티를 추가하고, 병렬쓰기가 가능해지도록 인터리빙한 것이다. 따라서, 2개 이상의 디스크 결함이 발생하여도 회복이 가능해진다. 그러나 이 조직에서는 한번의 쓰기 동작에 6번의 액세스 동작이 필요하다. 원래의 데이타와 행 및 열 패리티의 읽기, 새로운 데이타와 행 및 열 패리티의 쓰기. 이러한 오버헤드도 매우 높은 결함허용이 요구되는 응용에 대하여는 감수할만 한 것이다.

지금까지 설명된 RAID들 중에서 입출력 요구의 블럭 크기에 관계없이 좋은 성능을 가지는 조직은 RAID-1과 RAID-5이다. 이 두 조직의 성능 특성을 동일한 수의 디스크들에 대하여 비교한 결과를 보면 다음과 같다. 먼저, 읽기의 경우에는 미러링을 이용한 RIAD-1의 조직에서도 2배의 읽기 대역폭을 가질 수 있으므로 RAID-5와 마찬가지로 100%의 이용율을 보여준다. 그러나 쓰기 동작에 있어서는 다른 결과를 보여준다. RAID-1은 쓰기 동작이 두개의 디스크에 동시에 이루어지기 떠문에 대역폭이 절반(50%)으로 줄어든다. RAID-5에 있어서는 큰 블럭 쓰기의 경우에 G/(G+1) 퍼센트의 대역폭 이용율을 가질 수 있지만, 작은 블럭의 쓰기에서는 每 쓰기 동작에서 요구되는 4번의 입출력 때문에 대역폭의 25%만 유용하게 된다. 따라서, 작은 블럭의 쓰기에서는 RAID-1이 오히려 더 높은 대역폭을 가진다.

비교 결과를 전체적으로 정리하면, 작은 블럭의 읽기와 쓰기가 많은 환경에서는 RAID-1이 더 높은 성능을 보여준다. 그러나 용량과 비용을 중요시하는 응용이나 큰 블럭의 요구가 많은 경우에는 RAID-5가 더 높은 성능을 보이며, 특히 가격대 성능비 측면에서 더 우수하다.

Ⅳ. 입출력 관련 연구의 동향

앞에서 설명한 내용들은 이미 개발이 완료되었거나 제품 생산에 반영된 기술들에 대한 것들이었다. 이 장에서는 입출력 성능과 신뢰도를 더 향상시키기 위한 최근 연구 동향들에 대하여 간략히 소개하고자 한다.

디스크 액세스 시간에서 가장 큰 요소는 검색 시간(seek time)이다. 이것은 특히 직전에 액세스된 실린더와 다음번 실린더 사이의 거리가 먼 경우에는 매우 큰 지연을 유발하게 된다. Livny [5] 은 디스크 큐에 대기중인 요구들의 실린더 주소를 참조하여 처리 순서를 재배열함으로써 디스크 헤드의 운동거리를 줄일 수 있는 스케쥴링(scheduling) 알고리즘에 관하여 조사하였다. 고려된 알고리즘들은 FIFO(first-in first-out), SSF(shortest seek first), 및 FFBF (fastest fitting block first) 등이었다. 시뮬레이션 결과에 따르면, FIFO는 방법이 간단하며 이 용율이 낮은(underutilized) 시스템의 경우에 높은 성능을 보여주었다. SSF는 디스크 액세스에 있어서 지역성(locality)의 잇점을 최대화할 수 있었지만, 어떤 요구들은 매우 오랫동안 기다리게 되는 문제점을 가지고 있다. FFBF 방식은 크기가 서로 다른 다중블럭(multiblock) 요구들이 혼합되어 있는 경우에, 큰 블럭 요구가 먼저 들어와서 처리되고 있는 중에도 그

요구의 처리 시간에 영향을 주지 않는 한 나중에 들어온 작은 블럭 요구를 먼저 처리해 줄 수 있도록 조정해주는 방식이다. 따라서, 이 방식이 사용되면 적은 수의 블럭을 액세스하는 요구들이 빨리 처리되는 잇점을 보여준다.

Gibson과 Patterson [11] 은 디스크 배열의 신뢰도를 더 높일 수 있는 패리티 엔코딩(parity endoding) 방법을 제안하였다. 이 연구는 특히 신뢰도를 높이는데 따른 중복 데이타의 양과 여분의 디스크의 수를 최적화하고 오류 복구를 위한 오버헤드를 최소화시키는데 촛점을 두고 있다.

Reddy et al [12] 은 결함-허용 디스크 배열을 설계하는데 있어서 패리티 방식의 장점과 二重 복사(dual copy) 방법의 장점을 혼합한 새로운 기술을 개발하였다. 결과적으로 사용자는 원하는 결함허용의 정도에 따라 다양한 사양들 중에서 선택할 수 있도록 하였다. Chen [13] 은 RAID-5와 패리티 스트라이핑을 이용한 디스크 배열 조직을 위한 쓰기 동기화(write synchronization) 방식을 제안하고, 트랜잭션 처리와 UNIX 환경에서 얻은 작업부하를 이용하여 각 조직들의 성능을 비교하였다. Mourad et al [14] 은 시스템 실패(system crash) 또는 트랜잭션 중단(transaction abort) 등이 발생했을 때 데이타베이스를 신속히 복구하는 데 사용될 수 있는 새로운 RAID 조직을 제안하였다. Menon et al [15] 은 디스크 배열의 쓰기 동작에서 필요한 네번의 액세스를 세번 혹은 두번의 액세스로 개선할 수 있는 부동 패리티(floating parity) 방식을 제안하였다.

소결합 구조(loosely-coupled structure)를 가진 하이퍼큐브 시스템을 위한 디스크 서브시스템에 관한 새로운 연구도 진행되고 있다. Ghosh et al [16] 은 네트워크상에서 일반적인 데이타의 통신과 입출력 데이타 통신 사이의 상호 방해를 줄이기 위하여 입출력 노드들을 위한 네트워크를 추가하는 방법을 제안하였다. 분석적 방법과 시뮬레이션을 이용한 연구 결과에 따르면, 이 네트워크를 추가함으로써 통신 지연 시간을 줄이고 입출력 전송 대역폭을 높일 수 있음이 입증되었다.

V. 결 론

본 稿에서는 고성능 컴퓨터시스템에서 성능 병목의 주요 요인이 되고 있는 입출력 서브시스템의 성능 향상을 위하여 연구 개발되고 있는 제반 기술 동향에 대 하여 조사하였다. 입출력 성능을 향상시키기 위하여는 여러개의 입출력 프로세서와 채널을 제공하는 방법이나 주기억장치의 용량을 증가시켜 입출력 요구의 수를 감소시키는 방법 등도 중요하지만, 입출력 서브시스템내에서 가장 속도가 느린 요소인 디스크시스템의 속도를 향상시키는 것이 가장 핵심이 되고 있다. 따라서 여기서는 디스크시스템의 내부 구조와 저장 방식의 개선에 관한 기술들에 촛점을 두고 설명하였다.

디스크 드라이브는 기계적 장치이기 때문에 반도체 전자회로들과는 달리 자체의 속도를 향상시키기가 매우 어렵다. 따라서, 여러개의 디스크들에 데이타를 분산 저장하여 동시 액세스가 가능하도록 병렬성을 부가하는 기술이 중점적으로 연구되고 있다. 세부적으로 보면, 비트 또는 바이트 단위의 인터리빙 기술과 블럭 단위의 디클러스터링 기술이 어느 정도 정립되어 있고, 그러한 디스크 배열 구조를 바탕으로 하여 신뢰도를 향상시킨 RAID 기술 개발이 급속히 진전되어 많은 제품도 생산되고 있다. 그러나 신뢰도와 저장 효율 및 액세스 시간 사이에 존재하는 연관성 때문에 아직도 많은 보완이 필요하다. 또한, 이러한 새로운 디스크 조직들은 응용의 입출력 패턴이나 데이타 지역성에 따라 성능이 크게 달라질 수 있기 때문에 어떠한 조직이 최적이라고 말하기는 어려운 일이다.

현재의 전반적인 기술 발전 동향을 보면, 입출력 채널이나 디스크 서브시스템에 병렬성이 추가됨으로써 입출력 속도가 향상될 수 있는 것은 분명한 것이며, 이에 따른 보완을 위한 여러가지 연구가 계속되고 있으므로 많은 진전이 있을 것으로 기대된다. 그러나 프로세서와 기억장치의 속도 향상, 데이타베이스의 규모 증가, 멀티미디어 정보 처리 및 시스템내의 프로세서 수의 증가 등과 같은 흐름을 볼 때, 입출력 성능 향상을 위한 연구는 컴퓨터시스템 개발에 있어서 가장 중요시되어야 할 과제임에 틀림없다.

參 考 文 獻

[1] A. Reddy, "An evaluation of multiple-

disk I/O systems," *IEEE Trans. on Computers*, vol.38, no.12, pp. 1680-1690, Dec. 1989.

[2] H. Kung, "Memory requirements for balanced computer architectures," *in Proc. 13th Annu. Int. Symp. Computer Architecture*, pp. 49-54, 1986.

[3] M. Kim, "Synchronized disk interleaving," *IEEE Trans. on Computers*, vol. C-35, no. 11, pp. 978-988, Nov. 1986.

[4] K. Salem and H. Garcia-Milina, "Disk striping," *in Proc. Int. Conf. Data Engineering*, pp. 336-342, 1986.

[5] M. Livny, S. Khoshafian, and H. Boral, "Multi-disk management algorithms," *in Proc. ACM SIGMETRICS*, pp. 69-77, May 1987.

[6] Fujitsu America, *M2360A Parallel Transfer Disk Engineering Specifications*, San Jose, CA, 1986.

[7] Cray Research, *Cray X-MP and Cray-1 Computer Systems: Disk Systems Hardware Reference Manual*, Vol. H0077, Mendota Heights, MN, 1985.

[8] D. Patterson, G. Gibson, and R. Katz, "A case for redundant arrays of inexpensive disks (RAID)," *in Proc. ACM SIGMOD Conf.*, June 1988.

[9] R. Katz, G. Gibson, and D. Patterson, "Disk system architectures for high performance computing," *Proc. of IEEE*, vol. 77, no. 12, pp. 1842-1858, Dec. 1989.

[10] N. maginnis, "Store more, spend less: Mid-range abound," *Computerworld*, p. 71, Nov. 16, 1987.

[11] G. Gibson and D. Patterson, "Designing disk arrays for high data reliability," *J. of Parallel and Distributed Computing 17*, p. 4-27, 1993.

[12] A. Reddy and J. Banerjee, "Design and evaluation of gracefully degradable disk arrays," *J. of Parallel and Distributed Computing 17*, p. 28-40, 1993.

[13] S. Chen and D. Towsley, "The design and evaluation of RAID 5 and parity striping disk array architectures," *J. of Parallel and Distributed Computing 17*, p. 58-74, 1993.

[14] A. Mourad et al., "Recovery issues in databases using redundant disk arrays," *J. of Parallel and Distributed Computing 17*, p. 75-89, 1993.

[15] J. Menon et al., "Floating parity and data disk arrays," *J. of Parallel and Distributed Computing 17*, p. 129-139, 1993.

[16] J. Ghosh et al., "Performance evaluation of a parallel I/O subsystem for hypercube multicomputers," *J. of Parallel and Distributed Computing 17*, p. 90-106, 1993.

筆 者 紹 介

金 宗 鉉
1952年 1月 15日生
1976年 2月 연세대학교 전기공학과 졸업 (공학사)
1981年 2月 연세대학교 대학원 졸업 (공학석사)
1988年 1月 Arizona State University 전산공학과(Ph.D)

1976年 3月 ~ 1982年 2月 국방과학연구소 근무
1988年 2月 ~ 1990年 2月 한국전자통신연구소 프로세서구조연구실장
1990年 3月 ~ 현재 연세대학교(원주캠퍼스) 전산학과 부교수

주관심분야: 컴퓨터 구조, 병렬 알고리즘, 시뮬레이션