고성능 병렬화일 시스템을 위한 디스크 할당 방법

박 기 현†

요 약

최근, 멀티미디어 정보처리와 같은 대규모 데이터 처리에 필수적인 입출력 시스템의 성능을 높히기 위하여 많은 관심이 집중되고 있으며, 고성능 병렬화일 시스템에 관한 연구도 이런 노력에 속한다. 본 연구에서는 고성능 병렬화일 시스템을 위한 효율적인 디스크 할당 방법을 제안한다. 즉, 병렬화일의 자료 분산(data declustering) 특성을 이용하여 병렬화일에 대한 병렬도 개념을 정의하고, 이를 기반으로 여러 병렬화일들이 동시에 처리되는 경우에, 최대의 작업처리량(throughput)을 얻기 위한 각 병렬화일에 적합한 디스크상의 자료 분산 정도를 계산하는 방법을 제안한다. 또한 동시에 처리되는 병렬화일들이 많이 늘어날수록, 최대의 작업처리량을 얻기 위한 계산이 너무 복잡해지므로, 효율적인 근사 디스크 할당 알고리즘도 아울러 제안한다. 제안된 근사 알고리즘은 계산이 간단하고, 특히 입출력 작업부하(workload)가 높은 환경에서는 매우 효율적임을 보여준다. 또한 입출력 요구들의 도착 비율이 무한대일 경우, 근사 알고리즘만을 이용하여도 최대 작업처리량을 위한 최적 디스크 할당을 얻을 수 있음을 증명하였다.

A Disk Allocation Scheme for High-Performance Parallel File Systems

KeeHyun Park†

ABSTRACT

In recent years, much attention has been focused on improving I/O devices' processing speed which is essential in such large data processing areas as multimedia data processing. And studies on high-performance parallel file systems are considered to be one of such efforts. In this paper, an efficient disk allocation scheme is proposed for high-performance parallel file systems. In other words, the concept of a parallel disk file's parallelism is defined using data declustering characteristic of a given parallel file. With the concept, an efficient disk allocation scheme is proposed which calculates the appropriate degree of data declustering on disks for each parallel file in order to obtain the maximum throughput when more than one parallel file is used at the same time. Since, calculation for obtaining the maximum throughput is too complex as the number of parallel files increases, an approximate disk allocation algorithm is also proposed in this paper. The approximate algorithm is very simple and especially provides very good results when I/O workload is high. In addition, it has shown that the approximate algorithm provides the optimal disk allocation for the maximum throughput when the arrival rate of I/O requests is infinite.

1. 서 론

빠른 계산 능력을 가진 고성능 컴퓨터와 고속 데이터 전송능력을 가진 네트워크의 출현으로 인하여, 이러한 고속 정보처리 요구를 만족시키기 위한 컴퓨팅 환경이 조성되고 있다. 그러나 컴퓨터 시스템의 모든 부분이 고속 정보처리 요구를 만족시킬 수 있도록 성능이 향상되지는 않았다. CPU와 메모리의 처리속도는 최근에 급속하게 증가되었는데, CPU는 약 1년 반을

※ 이 연구는 계명대학교 비사연구비(1996)에 의해서 조성되었습니다.
† 정 회 원 : 계명대학교 컴퓨터 및 전자공학부 교수
논문접수 : 2000년 3월 15일, 심사완료 : 2000년 7월 27일

주기로 처리속도가 2배씩 증가하여 왔으며, 메모리는 2년을 주기로 집적도가 4배 이상씩 증가하여 왔다. 향후 이런 추세는 계속될 것이라는 전망이 우세하다 [1, 10].

따라서 입출력 서브 시스템에서의 병목현상을 줄이고 전체 시스템의 성능을 향상시켜서 멀티미디어 정보처리를 원활히 수행할 수 있도록, 입출력 속도 향상에 대한 연구가 최근 활성화되고 있다 [4, 8-9, 14, 16-17, 20-25]. 고성능 병렬화일 시스템 등에 대한 연구도 그 중의 하나이다. 병렬화일 시스템은 파일을 여러 보조 기억장치(주로 디스크)들에 분산 저장하는 방법으로서, 화일 입출력을 병렬로 할 수 있기 때문에 기존의 순차적 입출력 방법보다 훨씬 나은 성능을 가져온다. 즉, 고성능 병렬화일 시스템에서는 디스크 스트라이핑(disk striping) 혹은 자료 분산(data declustering) 등의 방법을 통하여 데이터를 분산 저장함으로써 동시에 여러 입출력 데이터를 처리할 수 있게 한다. 이런 방법은 현재 디스크 배열 혹은 RAID (Redundant Arrays of Inexpensive (Independent) Disks)를 매개체로 하여 구현되고 있다[11, 19].

1.1 관련 연구

현재까지 개발된 고성능 병렬화일 시스템들 중에서 대표적인 것은 다음과 같다:

(1) Bridge 병렬화일 시스템 : Bridge 병렬화일 시스템은 BBN Butterfly 병렬처리 시스템을 위하여 개발되었다 [5]. 데이터는 interleaved 형식으로 여러 디스크에 분산 저장되고 LFS(Local File System)에 의해 관리된다.

(2) sfs(scalable file system) 병렬화일 시스템 : sfs 병렬화일 시스템은 CM-5 병렬처리 시스템을 위하여 개발되었다 [15]. UFS(UNIX File System)을 CM-5에 적합하도록 수정하여 사용하기 때문에 UNIX 화일과 호환된다. 이 시스템은 SDA(Scalable Disk Array)를 사용하여 데이터를 분산 저장한다.

(3) Vesta 병렬화일 시스템 : Vesta 병렬화일 시스템은 대규모 병렬처리 시스템인 Vulcan에 사용되기 위하여 개발되었다 [4]. 입출력 노드를 따로 두어서 입출력 부분의 병목현상을 줄이려고 노력하였다. 사용자가 화일을 병렬로 분할 저장할 수 있도록 하는 기능이 있으며, 물리적으로는 입출력 노드로 분할된다. Vesta에서는 물리적 분할 수 및 레코드 크기 등을 제어할 수 있으며, 화일의 동적인 분할도 가능하다.

화일 시스템의 대표적인 매개체는 디스크이다. 따라서 시스템 측면에서 접근하여 디스크의 성능 향상을 꾀하는 연구들이 진행되었고, 그들 중에서 몇 가지 중요 연구들을 살펴보면 아래와 같다:

(1) 동기화된 디스크 인터리빙(synchronized disk interleaving) 시스템 : 디스크 인터리빙은 하나의 데이터 블록을 여러 개의 서브 블록들로 나누어서 각 서브 블록을 별개의 디스크에 저장한다. 이 방법은 대규모 데이터베이스 시스템이나 과학적인 응용을 위해서 디스크의 대역폭을 증가시키기 위한 수단으로 혹은 다중처리기 시스템에서 고성능 입출력 시스템 구조로써 연구되었다.

동기화된 디스크 인터리빙은 하나의 화일을 디스크들 사이에 바이트 단위로 인터리브하고 각 디스크의 헤드들은 같은 위치에 위치되도록 디스크들을 동기화시킨다. 모든 디스크들은 n배의 용량을 가진 하나의 큰 디스크처럼 작용하며 탐구시간(seek time)과 지연시간(latency time)은 동일하다. 모든 디스크들이 서로 연결되었기 때문에, 각 디스크는 같은 서비스 요구 비율(request rate)을 가지므로 각 디스크에 대해서 작업부하가 균형을 이룬다 [11, 12].

(2) 자료 분산(data declustering) 시스템 : 자료 분산 혹은 디스크 스트라이핑은 한 화일의 여러 블록들을 다른 디스크들에 저장하는 기법이다. 이 방법에서는 같은 화일의 여러 블록들이 병렬로 읽거나 쓰여질 수 있도록 블록 단위 인터리빙을 수행한다. 한 개의 화일을 여러 개의 블록들로 나누어서 다른 디스크들에 분산 저장하는 시스템을 자료 분산 시스템이라 한다 [12, 22]. 디스크 동기화 방법은 하드웨어적으로 동기화가 가능한 시스템에서만 이루어질 수 있으며, 현실적인 기술의 한계 때문에 많은 디스크들을 동기화 하기 어려운데 반해서, 자료 분산 시스템은 이런 제약이 비교적 적다.

1.2 논문의 초점

최근까지 입출력 서브 시스템의 성능을 향상시키고

자 하는 노력은 계속되어 왔으나, 디스크의 병렬화일 시스템에 관한 연구는 아직까지 초기 단계에 머무르고 있어서 보다 깊은 연구를 필요로 한다. 만약 병렬화일의 병렬도(degree of parallelism)를 구체적으로 정의하고, 이를 이용하여 각 병렬화일에 적합한 디스크 할당 방법이 체계적으로 개발된다면 입출력 시스템의 성능을 상당히 높일 수 있을 것이다. 그 근거로서, 프로세서의 계산속도에 대한 병렬도는 이미 그 개념이 정립되어 효율적인 프로세서 할당에 기여하고 있다[2, 6, 7, 18].

따라서, 본 연구에서는 고성능 디스크 병렬화일 시스템을 위한 효율적인 디스크 할당 방법을 제안한다. 즉, 병렬화일의 자료 분산(data declustering) 특성을 이용하여 병렬화일에 대한 병렬도 개념을 정의하고, 이를 기반으로 여러 병렬화일들이 동시에 처리되는 경우에, 최대의 작업처리량(throughput)을 얻기 위한 각 병렬화일에 적합한 디스크상의 자료 분산 정도를 계산하는 방법을 제안한다. 동시에 처리되는 병렬화일들이 많이 늘어날수록, 최대의 작업처리량을 얻기 위한 계산이 너무 복잡해지므로, 효율적인 근사 디스크 할당 알고리즘도 아울러 제안한다. 제안된 근사 알고리즘은 계산이 간단하고, 특히 입출력 작업부하(workload)가 높은 환경에서는 매우 효율적임을 보여준다. 또한 입출력 요구들의 도착 비율이 무한대일 경우, 근사 알고리즘만을 이용하여도 최대 작업처리량을 위한 최적 디스크 할당을 얻을 수 있음을 증명하였다.

본 연구의 구성은 다음과 같다: 병렬화일에 대한 병렬도 개념 및 이를 이용한 디스크 할당 방법들은 2절과 3절에서 각각 제안한다. 4절에서는 제안된 방법에 대한 성능 평가를 다루며, 마지막으로 5절에서는 결론 및 향후 연구 과제를 토의한다.

2. 병렬화일의 병렬도

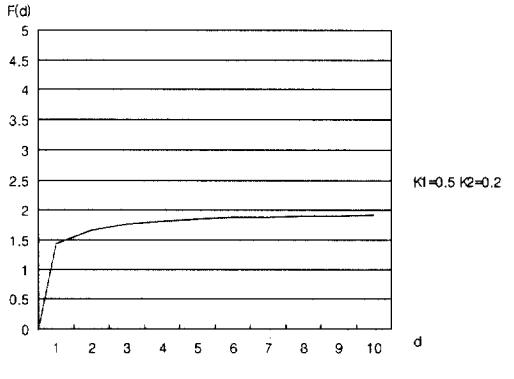
컴퓨터 시스템의 성능을 효율적으로 분석하기 위해서는 해당 시스템에 접수되는 작업부하의 특성을 파악하고, 이를 체계적으로 표현하여야 한다. 이 절에서는, 본 연구가 제안하는 병렬화일에 대한 병렬 특성 표현을 설명한다. 원래 병렬도는 프로세서 할당을 계산할 때 병렬 프로그램의 실행 속도 혹은 실행 형태를 나타내기 위하여 사용되었다 [6, 18].

본 연구에서는 병렬화일의 병렬도 개념을 정립하여 효율적인 고성능 병렬화일 디스크 할당 방법을 개발하는데 유용하게 사용한다. 병렬도는 병렬화일이 자료 분산 등의 방법을 통한 데이터의 분산 저장 형태를 규정하는 척도가 된다. 병렬도를 단순하지만 효율적으로 표현하는 자체가 좋은 할당 방법을 개발하는 지름길이라는 것이 일반적인 이치이므로, 본 연구에서는 병렬도의 단순성과 효율적인 표현 능력에 관심을 집중한다. 또한 본 연구에서는, 여러 병렬화일 입출력 요구들이 대기하고 있을 때, 각 병렬화일의 병렬도를 이용한 효율적인 디스크 할당 방법을 제시한다.

본 연구에서는 병렬화일의 병렬도를 해당 병렬화일에 대한 입출력 요구 처리 비율로 표현한다. 병렬화일 i의 입출력 병렬도 $F_i(d_i)$는 식 (1)과 같이 표현한다.

$$F_i(d_i) = \frac{d_i}{k_{i1} \cdot d_i + k_{i2}} \qquad (1)$$

여기서, d_i는 병렬화일 i에 대한 자료 분산 정도(degree of data declustering, dd)를, k_{i1} 및 k_{i2}는 병렬화일 i에 대한 실행 상수들이다. 자료 분산 정도는 하나의 병렬화일에 대해 가능한 병행 엑세스들의 수라고 정의하며, 자료를 분산 저장하기 위하여 필요한 디스크들의 개수를 의미한다. 만약 dd = m이면, 해당 병렬화일이 m개의 디스크들에 분산 저장되며 m개의 블록들이 동시에 엑세스될 수 있다. 실행 상수 k_{i1}, k_{i2}는 병렬화일 i를 d_i 만큼의 디스크들에 분산 저장하고 입출력 요구를 단독으로 처리하였을 때 얻을 수 있는 상수이다. 만일 k_{i1} = 0.5이고 k_{i2} = 0.2이면, 병렬화일 i의 입출력 병렬도 $F_i(d_i)$는 (그림 2)와 같다.

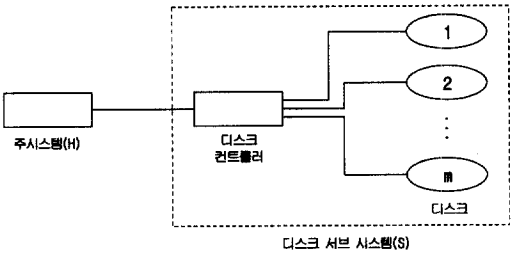


(그림 2) 병렬화일 i의 입출력 병렬도 $F_i(d_i)$(예)

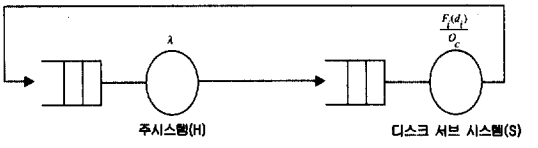
(그림 2)에서 보듯이, $F_i(d_i)$는 d_i의 값이 증가함에 따라 증가하지만, 어느 정도를 지나면, 증가 효과를 얻지 못하는 병렬화일의 특성을 나타낸다. 실행 상수들은 여러 가지 방법으로 얻을 수 있다. 예를 들면, d_i 만큼의 자료 분산을 한 병렬화일 i의 입출력 요구를 단독 처리하여 얻은 실행 비율들과 curve fitting 방법 [3]을 이용하여 상수 값들을 구할 수 있다.

3. 병렬화일의 디스크 할당

본 연구에서 고려하는 입출력 서브 시스템은 (그림 3)과 같다. 주 시스템(H)에서 디스크 입출력 요구가 발생되면, 디스크 컨트롤러를 통해서 디스크 서브 시스템(S)으로의 접근이 이루어진다. 병렬화일은, 병렬 검색이 가능하도록, 자료 분산 정도에 의거해서 m개의 디스크들에 분산 저장된다. (그림 3)의 시스템을 분석하기 위한 closed queueing 시스템은 (그림 4)와 같이 표현될 수 있다. $F_i(d_i)$의 병렬도 특성을 가진 각 병렬화일에 대한 입출력 요구들이 주 시스템에서의 대기(waiting)를 거쳐서 λ의 비율로 디스크 서브 시스템으로 들어가면, 디스크 서브 시스템에서는 전달된 입출력 요구를 $\frac{F_i(d_i)}{O_c}$의 비율로 서비스한다. 병렬화일들을 m개의 디스크들에게 골고루 분산 배치할 경우, 한 개의 디스크에 함께 저장될 병렬화일들의 최대 개수(즉, 최대 중첩도) $c = \lfloor \frac{\sum_{i=1}^{n} d_i}{m} \rfloor$이다. O_c는 병렬화일 i의 최대 중첩도가 c일 경우에 생길 수 있는 검색 지연 오버헤드이다. 이 절에서는, 주어진 시스템의 환경에서 가장 높은 작업처리량(throughput)을 얻을 수 있도록 각 병렬화일 i에 대한 d_i를 결정하는 방법을 제안한다.



(그림 3) 시스템 모델



(그림 4) Queueing 시스템

(그림 4)의 queueing 시스템에 대한 Markov diagram은 (그림 5)와 같다. 여기서, $P(H_1, H_2, \ldots, H_n)$는 모든 병렬화일에 대한 입출력 요구들이 H에 있는 상태를 의미하며, $P(H_1, \ldots, H_{k-1}, S_k, H_{k+1}, \ldots, H_n)$는 병렬화일 k에 대한 입출력 요구만 S에 있고, 나머지는 H에 있는 상태를 의미한다.

이 diagram에 관한 flow balance equation [13]들은 다음과 같다.

$$\lambda \cdot P(\varnothing) = \sum_{k=1}^{n} F_k(d_k) \cdot P(\{k\})$$

$$(\lambda + F_k(d_k)) \cdot P(\{k\}) = \frac{1}{O_c} \sum_{j=1}^{n} F_j(d_j) \cdot P(\{k,j\}), \quad \textit{where } k \neq j$$

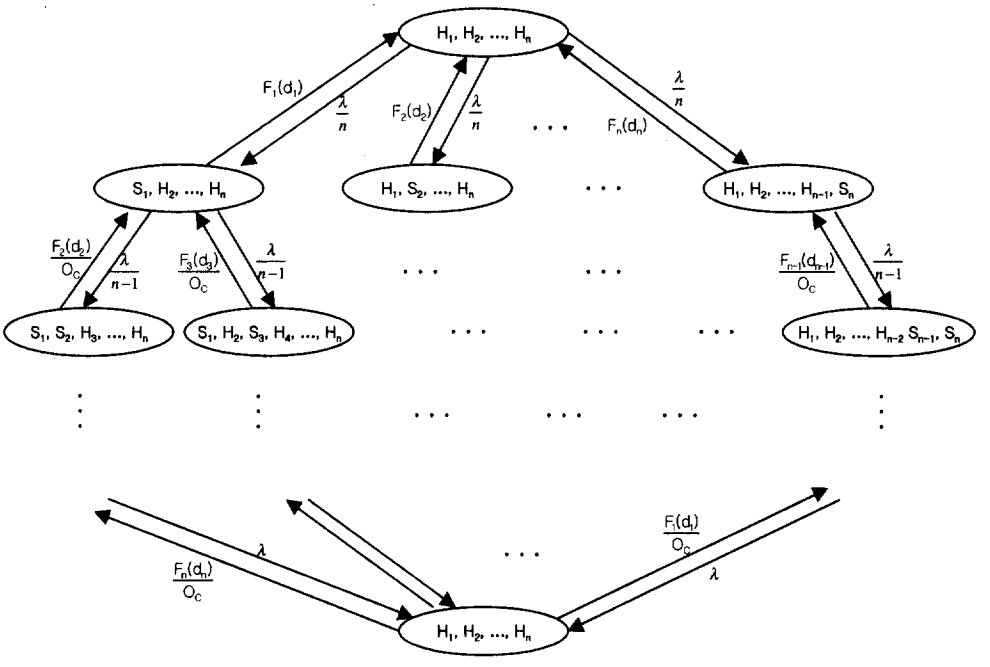
$$(\lambda + \frac{1}{O_c} \sum_{k=1}^{n} F_k(d_k)) \cdot P(\{A\}) = \frac{1}{O_c} \sum_{j=1}^{n} F_j(d_j) \cdot P(A \cup \{j\}), \quad \text{for } \textit{every } A \text{ and } k \neq j$$

$$\frac{1}{O_c} \sum_{k=1}^{n} F_k(d_k)) \cdot P(\{1,2,\ldots,n\}) = \lambda \cdot \sum_{k=1}^{n} P(\{1,2,\ldots,n\} - \{k\})$$

$$\sum_{P} = 1$$

$P(\{A\})$는 A 집합에 있는 병렬화일들에 대한 입출력 요구들만 S에 있고, 나머지는 H에 있는 상태를 의미한다. $P(\{\varnothing\})$는 모든 병렬화일들의 입출력 요구들이 H에 있는 상태를 의미하고, $P(\{1,2,\ldots,n\})$은 모든 병렬화일들의 입출력 요구들이 S에 있는 상태를 나타낸다. λ는 디스크 서브 시스템에로의 입출력 요구 도착 비율이다.

위에서 설명한 flow balance equation을 이용하여 연립방정식을 계산하면, 전체 시스템의 작업처리량을 최대화할 수 있는 각 병렬화일의 최적 자료 분산 정도를 구할 수 있다. 그러나, 이 방법은 최적의 자료 분산 정도를 구할 수는 있으나, 병렬화일들의 수가 증가할



(그림 5) Markov diagram

수록 수식이 복잡해지기 때문에 시스템이 복잡하게 되면 실용적일 수는 없다.

따라서 본 연구에서는, 위에서 언급한 방법 이외에 근사적인 방법도 아울러 제안한다. 이 방법은, 비록 최적의 값을 구할 수는 없으나, 근사값을 쉽게 구할 수 있으므로 실용적이다. (그림 6)은 근사 알고리즘을 보여준다. (그림 6)에서 $\Delta_i(d_i)$는 식 (2)와 같이 표현된다. $\Delta_i(d_i)$는 (d_i-1) 만큼의 자료 분산 저장을 한 병렬화일 i가 한 개의 디스크를 더 늘려서 자료 분산 저장을 할 경우에 향상될 수 있는 처리비율 증가를 나타낸다.

$$\Delta_i(d_i) = \frac{F_i(d_i)}{O_c} - \frac{F_i(d_i-1)}{O_c} \qquad (2)$$

(그림 6)의 근사 알고리즘은 각 병렬화일이 m개의 디스크들 중에서 몇 개를 할당받아서 분산 저장하느냐를 결정한다. 즉, 각 병렬화일의 적합한 자료 분산 정도를 계산한다. 근사 알고리즘은 디스크를 한 개씩 순차적으로 할당하며, 식 (2)를 이용하여 디스크를 할당하였을 경우에 해당 병렬화일의 병렬도 기준으로 가장 높은 성능 향상을 이룰 수 있는 병렬화일을 선정한다. m을 할당할 전체 디스크 수라고 하면, 근사 알고리즘의 시간 복잡도(time complexity)는 $O(m)$에 불과하다.

```
Let Ndisks be the total number of disks to be assigned
assign one disk to every parallel file i
 di = 1 for every parallel file i
Ndisks = Ndisks - Σ_{i=1}^{n} di
calculate Δi(di) for every parallel file i
while (Ndisks > 0)
begin
    let k be the parallel file which has biggest Δi(di)
      among all parallel files
    assign one more disk to parallel file k
     dk = dk + 1
    Ndisks = Ndisks - 1
    calculate new Δi(di) for every parallel file i
end
```

(그림 6) 근사 알고리즘

[정리 1] n개의 병렬화일들을 m개의 디스크들에게 할당하는 closed queueing system 환경에서 $\lambda=\infty$이면, (그림 6)의 근사 알고리즘에서 계산한 $d_k\ (k = 1, \cdots, n)$들은 최적 자료 분산 정도이다.

[증명] $\lambda=\infty$이기 때문에 모든 병렬화일들의 I/O 요구들이 동시에 처리되고 있다는 가정이므로, (그림 6)의 근사 알고리즘에서 구한 d_k들이 최적 자료 분산 정도임을 증명하기 위해서, $T=\frac{1}{O_c}\sum_{k=1}^{n}F_k(d_k)$, $(k=1,2,\cdots,n, \sum_{k=1}^{n}d_k=m,\ 0\le d_k\le m.)$이 최대 작업처리량임을 보이면 된다. OPT_i를 병렬화일 i의 최적 자료 분산 정도라고 하고 d_i를 (그림 6)의 근사 알고리즘에서 구한 근사값이라고 하면, 모든 i에 관해서 $OPT_i=d_i$임을 보이면 된다.

우선 n개의 병렬화일들을 2개의 그룹, 즉 1개의 병렬화일 그룹과 나머지 $n-1$개의 병렬화일 그룹으로 나눈다. 1개의 병렬화일 그룹에 속한 파일을 병렬화일 1이라 표기하자. F_1은 병렬화일 1의 I/O 요구 처리 비율이고 $\widehat{F_{n-1}}$는 $n-1$개 병렬화일들의 합성 I/O 요구 처리 비율을 나타낸다. $\widehat{F_{n-1}}$는 F_2, F_3, …, F_n로 부터 다음과 같이 합성하여 만든다:

(1) 각 병렬화일 i (i = 2, 3, …, n)에 관해서, $\Delta_i(1)$, $\Delta_i(2)$, …, $\Delta_i(m)$을 구한다.

(2) 위에서 구한 $(n-1)m$개의 Δ_i들을 내림차순으로 배열한다. 이 배열을 $\widehat{\Delta_{n-1}}(1)$, $\widehat{\Delta_{n-1}}(2)$, …, $\widehat{\Delta_{n-1}}$ $((n-1)m)$으로 표현한다.

(3) $\widehat{F_1}(i) = -1$, $(i=m+1,m+2,\cdots,(n-1)m)$

(4) $\widehat{F_{n-1}}(0) = 0$, $\widehat{F_{n-1}}(i+1)=\widehat{F_{n-1}}(i)+\widehat{\Delta_{n-1}}$, $(i+1)$, $(i=0,1,2,\cdots,(n-1)m-1)$

$\widehat{F_{n-1}}((n-1)m)=\sum_{k=2}^{n}F_k(m)$이고 $\widehat{F_{n-1}}$은 increasing concave 형태의 함수가 된다. (3)항은 근사 알고리즘이 병렬화일 1에 m개 이상의 디스크를 배정할 수 없도록 한다.

$OPT_1=d_1+t$ $(1\le t\le m-d_1)$라고 가정하자. 즉, 병렬화일 1은 t개의 디스크를 더 할당받아야 최적이 된다고 가정한다. 또한, 나머지 n-1개의 병렬화일 그룹의 최적값은 $\widehat{D_{n-1}}$이라고 가정한다 ($OPT_1+\widehat{D_{n-1}}=m$). 따라서,

$$T_{\max} = \widehat{T_1}+\frac{1}{O_c}\Big(\sum_{i=1}^{t}\Delta(d_1+i) - \sum_{i=1}^{t}\widehat{\Delta_{n-1}}(\widehat{D_{n-1}}-i)\Big) \le \widehat{T_1}+\frac{1}{O_c}(t\Delta_1(d_1+1) - t\widehat{\Delta_{n-1}}(\widehat{D_{n-1}}))$$

여기서 $T_{\max}$는 시스템의 최대 작업처리량이고, $\widehat{T_1}=\frac{1}{O_c}(F_1(d_1)+\widehat{F_{n-1}}(\widehat{D_{n-1}}))$이다. 위에서 언급한 가정에 의해서 $T_{\max}>\widehat{T_1}$이므로, 위의 식에서 $\Delta_1(d_1+1)>\widehat{\Delta_{n-1}}(\widehat{D_{n-1}})$이 된다. 그런데, 이럴 경우, 근사 알고리즘에 의해서 병렬화일 1은 적어도 1개의 디스크를 더 할당받을 수 있게 되므로, 모순이다. $OPT_1=d_1-t$ $(1\le t<d_1)$일 경우에도 비슷한 논리도 모순을 유도할 수 있다. 따라서 $OPT_1=d_1$이고 d_1은 병렬화일 1에 대한 최적 자료 분산 정도이다.

다음, n-1개의 나머지 병렬화일 그룹들을 다시 두 그룹(병렬화일 2 그룹 및 n-2개의 나머지 병렬화일 그룹)으로 분리한다. 위에서 언급한 논리와 비슷한 방법으로 $OPT_2=d_2$임을 증명할 수 있다. 이런 식으로 병렬화일 n-1 그룹과 병렬화일 n 그룹으로 나누어질 때까지 계속하여 $OPT_{n-1}=d_{n-1}$ 및 $OPT_n=d_n$임을 증명한다. 그러므로, 전체 작업처리량 $T=\frac{1}{O_c}\sum_{k=1}^{n}F_k(d_k)$가 최대이다. ■

4. 성능 평가

(그림 6)에서 제안한 근사 알고리즘의 성능을 평가하기 위하여 2개 및 3개의 병렬화일들에 대한 입출력 요구들이 동시에 처리되는 환경((그림 5)에서 n = 2, 3)을 생각한다. 할당될 디스크 수는 8개로 한정했다. Flow balance equation을 이용하여 계산한 최적값과 (그림 6)의 근사 알고리즘에서 구한 근사값들을 비교하면 표 1과 같다. 여기서 O_c는 1.2로 설정했다. 표 1에서 볼 수 있듯이, 최적 및 근사 작업처리량(TPUT)들을 비교했을 때, 모두 4% 미만의 상대 오차를 보여준다. 특히, λ = 5.0, 10.0 및 30.0 일 때에는 병렬화일들의 특성에 관계없이, 최적값과 동일한 근사값을 얻을 수 있음을 보여준다. 이것은 앞에서 언급한 정리 1의 내용을 부분적으로 실증하는 예가 될 수 있다. 표 2는 O_c를 1.5로 설정했을 경우의 최적값 및 근사값들에 대한 비교 결과를 보여준다.

<표 3> 및 <표 4>는 3개의 병렬화일들의 입출력 요구들이 동시에 처리될 경우에 최적값 및 근사값들에 대한 비교 결과를 보여준다. <표 1> 및 <표 2>에서와 마찬가지로, 입출력 요구들의 도착 비율이 높을수록, 근사값들이 최적값들에 가까이 접근함을 알 수 있다. 또한 입출력 요구들의 도착 비율이 높을수록, 병렬도 $F_i(d_i)$의 기울기가 높은 요구에게 많은 디스크 할당하는 것이 최적에 가까운 작업처리량을 얻을 수 있음을 알 수 있다.

〈표 1〉 최적값 및 근사값 비교표
(병렬화일들의 수 = 2, O_c = 1.2)

(k11,k12) (k21,k22)	도착 비율	최적 (d1,d2)	최대 TPUT	근사 (d1,d2)	근사 TPUT	상대오차 (TPUT)
(0.5,0.2) (0.2,0.5)	0.1	(2,6)	0.100	(2,6)	0.100	0.0%
	0.3	(3,5)	0.298	(2,6)	0.298	0.0%
	1.0	(2,6)	0.945	(2,6)	0.945	0.0%
	10.0	(2,6)	3.469	(2,6)	3.469	0.0%
	30.0	(2,6)	4.031	(2,6)	4.031	0.0%
(0.5,0.2) (0.2,50.0)	0.1	(2,6)	0.098	(5,3)	0.096	2.0%
	0.3	(4,8)	0.274	(5,3)	0.263	4.0%
	1.0	(4,5)	0.688	(5,3)	0.665	3.3%
	10.0	(5,3)	1.387	(5,3)	1.387	0.0%
	30.0	(5,3)	1.518	(5,3)	1.518	0.0%
(0.2,0.5) (0.2,50.0)	0.1	(2,6)	0.099	(7,1)	0.098	1.0%
	0.3	(5,8)	0.282	(7,1)	0.280	0.7%
	1.0	(8,6)	0.802	(7,1)	0.794	1.0%
	10.0	(7,1)	2.711	(7,1)	2.711	0.0%
	30.0	(7,1)	2.802	(7,1)	2.802	0.0%

〈표 2〉 최적값 및 근사값 비교표
(병렬화일들의 수 = 2, O_c = 1.5)

(k11,k12) (k21,k22)	도착 비율	최적 (d1,d2)	최대 TPUT	근사 (d1,d2)	근사 TPUT	상대오차 (TPUT)
(0.5,0.2) (0.2,0.5)	0.1	(2,6)	0.100	(2,6)	0.100	0.0%
	0.3	(3,5)	0.298	(2,6)	0.298	0.0%
	1.0	(2,6)	0.945	(2,6)	0.945	0.0%
	10.0	(2,6)	2.982	(2,6)	2.982	0.0%
	30.0	(2,6)	3.314	(2,6)	3.314	0.0%
(0.5,0.2) (0.2,50.0)	0.1	(2,6)	0.098	(5,3)	0.096	2.0%
	0.3	(2,6)	0.273	(5,3)	0.263	3.7%
	1.0	(4,5)	0.688	(5,3)	0.665	3.3%
	10.0	(5,3)	1.142	(5,3)	1.142	0.0%
	30.0	(5,3)	1.227	(5,3)	1.227	0.0%
(0.2,0.5) (0.2,50.0)	0.1	(2,6)	0.099	(7,1)	0.098	1.0%
	0.3	(5,8)	0.282	(7,1)	0.280	0.7%
	1.0	(7,1)	0.794	(7,1)	0.794	0.0%
	10.0	(7,1)	1.987	(7,1)	1.987	0.0%
	30.0	(7,1)	2.284	(7,1)	2.284	0.0%

〈표 3〉 최적값 및 근사값 비교표
(병렬화일들의 수 = 3, O_c = 1.2)

(k11,k12) (k21,k22) (k31,k32)	도착 비율	최적 (d1,d2)	최대 TPUT	근사 (d1,d2)	근사 TPUT	상대오차 (TPUT)
(0.5,0.2) (0.2,0.5) (0.2,50.0)	0.1	(1,1,6)	0.080	(5,2,1)	0.064	20.0%
	0.3	(2,1,5)	0.192	(5,2,1)	0.170	11.5%
	0.6	(2,2,4)	0.322	(5,2,1)	0.308	4.3%
	1.0	(3,2,3)	0.467	(5,2,1)	0.461	1.3%
	10.0	(5,2,1)	0.988	(5,2,1)	0.988	0.0%
	20.0	(5,2,1)	0.621	(5,2,1)	0.621	0.0%
	30.0	(5,2,1)	0.377	(5,2,1)	0.377	0.0%

〈표 4〉 최적값 및 근사값 비교표
(병렬화일들의 수 = 3, O_c = 1.5)

(k11,k12) (k21,k22) (k31,k32)	도착 비율	최적 (d1,d2)	최대 TPUT	근사 (d1,d2)	근사 TPUT	상대오차 (TPUT)
(0.5,0.2) (0.2,0.5) (0.2,50.0)	0.1	(1,1,6)	0.080	(5,2,1)	0.064	20.0%
	0.3	(2,1,5)	0.191	(5,2,1)	0.170	11.0%
	0.6	(2,2,4)	0.319	(5,2,1)	0.307	3.8%
	1.0	(3,2,3)	0.458	(5,2,1)	0.454	0.9%
	10.0	(5,2,1)	0.801	(5,2,1)	0.801	0.0%
	20.0	(5,2,1)	0.431	(5,2,1)	0.431	0.0%
	30.0	(5,2,1)	0.242	(5,2,1)	0.242	0.0%

5. 결론 및 향후 연구과제

본 연구는 병렬화일들을 효율적으로 디스크 시스템에 저장 검색하기 위한 디스크 할당 방법을 제안하였다. 이를 위하여 병렬화일의 병렬도 개념 및 이를 기반으로 한, 자료 분산 정도 결정 방법을 제안하였다. 최적의 자료 분산 정도를 구할 수도 있으나, 병렬화일들의 수가 많아질수록 계산이 복잡하여지기 때문에, 근사적인 알고리즘도 아울러 제안하였다. 특히 디스크 입출력 요구들의 도착 비율이 무한대인 환경에서는, 제안된 근사 알고리즘이 최대 작업처리량을 위한 최적의 디스크 할당을 계산할 수 있음을 증명하고, 이를 부분적으로 실증하였다. 또한 계산을 통하여, 입출력 요구들의 도착비율이 높을수록, 근사값들이 최적값들에 가까이 접근함을 알 수 있었다. 또한 입출력 요구들의 도착 비율이 높을수록, 병렬도 $F_i(d_i)$의 기울기가 높은 요구에게 많은 디스크 할당하는 것이 최적에 가까운 작업처리량을 얻을 수 있음을 알 수 있었다.

그러나, 만약에 시스템의 도착비율에 대한 고려도 근사 알고리즘에서 이루어 질 수 있다면 좀 더 나은

방법을 제안할 수 있을 것이라고 생각되므로, 이에 대한 연구가 앞으로 이루어져야 할 것이다. 또한, 시뮬레이션이나 실제 시스템에서의 구현에 의한 근사 알고리즘의 성능 평가도 향후 이루어져야 할 것이다.

참 고 문 헌

[1] P. M. Chen, *et. al.*, "An evaluation of redundant arrays of disks using an Amdahl 5890," *Proc. of SIGMETRICS Conf. on Measurement and Modelling of Computer Systems*, pp.74-85, May 1990.

[2] J. G. Choi and K. H. Park, "Performance comparisons of processor allocation schemes using simulation techniques," *Journal of the Korea Society for Simulation*, Vol.3, No.1, pp.43-54, July 1994.

[3] S. D. Conte and C. Boor, *Elementary Numerical Analysis : An Algorithmic Approach*, 3rd. ed., McGraw-Hill, 1980.

[4] P. F. Corbett, "Overview of the Vesta parallel file system," *Computer Architecture News*, Vol.21., No.5, pp.7-14, 1993.

[5] P. C. Dibble, M. L. Scott and C. S. Eliss, "Bridge : a high-performance file system for parallel processing," *8th. Int. Conf. on Dist. Comput. Syst.*, pp. 154-161, 1988.

[6] L. W. Dowdy and M. R. Leuze, "On modeling partitioned multiprocessor systems," *Int. Journal of High Speed Computing*, Vol.5, Sept. 1993.

[7] D. L. Eager, J. Zahorjan, and E. D. Lazowska, "Speedup versus efficiency in parallel systems," *IEEE Trans. on Computers*, Vol.38, pp.408-423, Mar. 1989.

[8] G. R. Ganger, *et. al.*, "Disk arrays : high-Performance, high-reliability storage subsystems," *IEEE COMPUTER*, Vol.27, No.3, pp.30-36, Mar. 1994.

[9] R. Karedla, J. S. Love, and B. G. Wherry, "Caching strategies to improve disk system performance," *IEEE COMPUTER*, Vol.27, No.3, pp.38-46, Mar. 1994.

[10] R. H. Katz, *et. al.*, "Disk system architectures for high performance computing," *Proc. of the IEEE*, Vol.77, No.12, pp.1842-1858, Dec. 1989.

[11] S. Kenneth and M. H. Garcia, "Disk striping," *Proc. of the IEEE Data Engineering Conf.*, Los Angeles, CA, pp.336-342, Feb. 1986.

[12] M. Y. Kim, "Synchronized disk interleaving," *IEEE Trans. on Computers*, C-35(11), pp.978-988, Nov. 1986.

[13] L. Kleinrock, *Queueing Systems Vol.1 : Theory*, John Wiley & Sons, New York, 1975.

[14] E. K. Lee, "Software and performance issues in the implementation of a RAID prototype," *Tech. Report UCB/CSD 93*, Mar. 1993.

[15] S. J. LoVerso, M. Isman, and A. Nanopoulos, "sfs : a parallel file system for the CM-5," *Proc. of 1993 Summer USENIX*, pp.291-304, 1993.

[16] E. L. Miller and R. H. Katz, "RAMA : A File System for Massively-Parallel Computers," *12th. IEEE Symp. on Mass Storage Systems*, pp.163-198, 1993.

[17] N. Nieuwejaar and D. Kotz, "The Galley Parallel File System," *ACM ICS '96, Philadelphia, USA*, pp.374-381, 1996.

[18] K. H. Park, *Dynamic processor partitioning for multiporgrammed multiprocessor systems*. Ph.D. Thesis, Vanderbilt Univ., Nashville, TN, 1990.

[19] D. A. Patterson, G. Gibson and R. H. Katz, "Introduction to redundant arrays of inexpensive disks (RAID)," *Proc. of Spring COMCON*, pp.112-117, Feb. 1989.

[20] P. Pierce, "A concurrent file system for a highly parallel mass storage subsystem," *4th. Conf. on Hypercubes, Concurrent Computers and Applications*, pp.161-166, Mar. 1989.

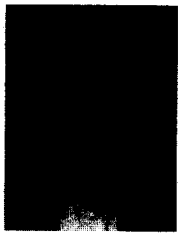
[21] T. W. Pratt, *et. al.*, "A comparison of the architecture and performance of two parallel file systems," *4th. Conf. on Hypercubes, Concurrent Computers and Applications*, pp.161-166, 1989.

[22] A. N. Reddy and P. Banerjee, "An evaluation of multiple-disk I/O systems," *IEEE Trans. on COMPUTERS*, Vol.38, No.12, pp.1680-1690, Dec. 1989.

[23] A. N. Reddy and J. C. Wyllie, "I/O issues in a multimedia system," *IEEE COMPUTER*, Vol.27, No.3, pp.69-74, Mar. 1994.

[24] J. Rosario and A. N. Choudhary, "High-performance I/O for massively parallel computers : problems and prospects," *IEEE COMPUTER*, Vol.27, No.3, pp. 59-68, Mar. 1994.

[25] K. Salem and G. Garcia-Molina, "Disk striping," *Proc. of 2nd. IEEE Data Engr. Conf*, pp.336-342, Feb. 1986.



박 기 현

e-mail : khp@kmucc.keimyung.ac.kr

1979년 경북대학교 전자공학과 (컴퓨터 전공) 졸업(학사)

1981년 한국과학기술원 전자계산학과 졸업(석사)

1990년 미국 Vanderbilt 대학교 전자계산학과 졸업(박사)

1981년~현재 계명대학교 컴퓨터 및 전자공학부 교수

관심분야 : 병렬/분산처리 시스템, 운영체제, 성능분석 등