삽입기법과 양방향 스캔 기법에 기반한 실시간 디스크 스케줄링 알고리즘

정회원 이 명 섭*, 박 창 현**

The real-time scheduling algorithms based on the Insertion technique and Two-way SCAN technique

Myung Sub Lee*, Chang Hyeon Park** ***Regular Members***

요 약

최근 들어, 실시간 디스크 스케줄링에서 단위 시간당 처리량을 최대화하기 위한 다양한 실시간 스케줄링 알고리즘들이 연구되고 있다. 특히, 실시간 디스크 스케줄링 알고리즘을 대표하는 EDF(Earliest Deadline First)에 스캔기법을 적용한 많은 알고리즘들이 연구되고 있다. 그러나 제안된 디스크 스케줄링 알고리즘들은 스캔 그룹을 생성할 때 계속되는 I/O 요구들을 고려해야 하기 때문에 많은 제약을 가지고 있다. 또한, 스캔 기법에서 서비스 방향의 고정으로 인하여 디스크의 효율성을 떨어뜨리는 결과를 초래하고 있다. 본 논문에서는 기존의 경성 실시간 시스템에서 사용되는 실시간 디스크 스케줄링 알고리즘들의 문제점을 해결하기 위해 삽입 기법과, 양방향 스캔기법을 기반으로 한 새로운 실시간 디스크 스케줄링 알고리즘을 제안하고 실험을 통해 제안 시스템의 시간당 처리량과 서비스 가능한 I/O 요구가 우수함을 증명한다.

Key Words : Real-time disk scheduling, EDF, Insertion method, two-way scan technique

ABSTRACT

Recently, to increase throughput per hour on real-time disk scheduling, a lot of algorithms that apply SCAN technique to EDF(Earliest Deadline First) that is representative real-time disk scheduling algorithm are studied. However, existing disk scheduling algorithms have several limitations because they consider continuous I/O requests when create SCAN group. Also, because SCAN technique was fixed direction, the existing algorithms have shortcoming that there are a lot of time damages. This paper proposes a new real-time disk scheduling algorithm based on the insertion technique and the two-way SCAN technique to solve the problems of the exiting real-time disk scheduling algorithms in hard real-time system. The simulation result shows that, when using our techniques, the disk throughput and the number of serviceable I/O requests are enhanced.

Ⅰ. 서 론

디스크 스케줄링 알고리즘이란 대기 중인 디스크 입출력 요구의 처리순서를 결정하는 기법으로서, 목적은 디스크 시스템의 성능향상에 있다. 디스크 시스템의 성능향상을 위해서는 크게 3 가지 측면을 고려해야한다. 첫째, 단위 시간당 얼마나 많은 디스크 입출력 요구를 서비스하는가를 측정하는 단위 시간당 처리량(Throughput), 둘째, 각 디스크 입출력 요구에 대해 얼마나 빠른 시간 내에 서비스하는

* 영남대학교 컴퓨터공학과 인공지능 및 지능정보 시스템 연구실 (skydream@yu.ac.kr)
** 영남대학교 컴퓨터공학과 부교수 (park@yu.ac.kr) 교신저자
논문번호 : KCS2004-11-276, 접수일자 : 2005년 02월 01일

가를 측정하는 평균 응답시간(response time), 셋째, 예측성 판단을 위한 요소로서 응답시간의 분산(Variance)에 사용되는 응답시간의 예측성(Predictability)등의 측면을 고려한 여러 가지 디스크 스케줄링 알고리즘이 제안 되었다[1]. 그러나 이러한 알고리즘들은 디스크의 효율성만을 고려하고 실시간 디스크 입출력 요구의 종료시간은 고려하지 않기 때문에 실시간 시스템에서 이용하는 데에는 부적합한 면이 있다. 따라서 이러한 문제점을 해결하기 위한 여러 가지 실시간 디스크 스케줄링 알고리즘이 제안 되었다.

실시간 디스크 스케줄링 알고리즘은 종료시간 이내에 디스크 입출력 요구를 서비스하면서 또한 디스크를 효율적으로 이용할 수 있도록 디스크 입출력 요구의 순서를 결정한다. 기존의 실시간 디스크 스케줄링 알고리즘에는 EDF(Earliest Deadline First)[5, 6], EDF에 SCAN기법을 추가한 P-SCAN(Priority SCAN)[7] SCAN-EDF[8], 큐를 이용한 SSEDO/SSEDV(Shortest Seektime and Earliest Deadline by Ordering/Value)[9], 디스크 입출력 요구들의 긴급도에 기반 한 UG-SSTF(Urgent Group and Shortest Seek Time First)[10], SCAN그룹을 확장시킨 MSG (Maximum Scannable Group)[11], 그리고 MSG를 이용하여 종료시간을 수정한 DMS(Deadline Modification SCAN)등이 있다[12]. 그러나 이들 실시간 디스크 스케줄링 알고리즘들은 종료시간의 순서대로 디스크 입출력 요구를 서비스함으로써 디스크의 효율성이 나빠지는 결과를 초래하고 있다. 이를 해결하기 위해 SCAN기법을 이용한 스케줄링 알고리즘으로 디스크의 효율성을 개선 시켰지만, 여전히 연속된 요구들만 SCAN그룹으로 묶을 수 있다는 제한을 가진다. 뿐만 아니라 SCAN기법의 서비스방향 또한 고정되어 있어서 결과적으로 디스크의 효율성이 떨어진다.

본 논문에서는 이런 문제점을 해결하기 위하여 연속되지 않은 실시간 요구들을 효과적으로 SCAN 그룹에 삽입할 수 있는 삽입 기법과, 연속하는 SCAN그룹들을 합병할 수 있는 SCAN합병 기법, 그리고 SCAN기법을 적용하여 서비스 할 때 성능을 개선할 수 있는 양방향 SCAN기법을 기반으로 하는 새로운 실시간 디스크 스케줄링 알고리즘을 제안한다.

본 논문의 2장에서는 관련연구를 기술하고, 3장에서는 시간당 처리량을 높일 수 있는 새로운 실시간 디스크 스케줄링 알고리즘, 즉 삽입 기법과 양방향 SCAN기법을 기반으로 하는 디스크 스케줄링 알고리즘을 기술한다. 4장에서 제안알고리즘의 성능평가를 보이고, 5장에서 결론을 기술한다.

Ⅱ. 관련 연구

기존의 SCAN[2]이나 SSTF(Shortest Seek Time First)[3, 4]같은 비 실시간 디스크 스케줄링 알고리즘은 서비스 순서를 결정할 때 각각 입출력 요구가 요청하는 데이터의 물리적 위치만을 고려하기 때문에 실시간적인 특성을 갖는 입출력 요구, 즉 종료시한 이내에 서비스를 받아야 하는 디스크 입출력 요구들을 서비스 하는 데에는 적합하지 않다. 이러한 문제점을 해결하기 위하여 여러 가지 실시간 디스크 스케줄링 알고리즘이 제안되었고 그것에 관한 많은 연구 결과가 발표 되었다. 그중 대표적인 실시간 디스크 스케줄링 알고리즘을 살펴보면 다음과 같다.

EDF(Earliest Deadline First)는 가장 잘 알려진 실시간 디스크 스케줄링 알고리즘으로 종료시간에 가장 가까운 요구를 먼저 처리해 주는 알고리즘이다. 즉, 종료시간이 가장 빠른 요구가 가장 높은 우선순위를 갖고 서비스를 받는다. EDF는 디스크 입출력 요구들의 서비스 시간이 미리 알려져 있다면 최적이라고 알려져 있다. EDF 알고리즘의 경우 디스크 입출력 요구를 서비스 하는데 있어서 디스크 헤드의 탐색시간을 줄이기 위한 방법을 고려하지 않기 때문에 디스크의 효율성이 상당히 떨어지는 단점을 가진다.

P-SCAN(Priority SCAN)은 디스크 큐 안에 있는 모든 입출력 요구들을 몇 개의 우선순위 단계로 분류하는 전략에 기반을 두고 있다. 그리고 각각의 우선순위 단계에서는 SCAN을 이용하여 입출력 요구를 서비스 해 준다. 가장 우선순위가 높은 단계에 속해 있는 입출력 요구를 모두 서비스하면 다음으로 우선순위가 높은 단계에 속해 있는 입출력 요구들을 서비스 해 준다. 각각의 입출력 요구를 서비스한 후에 디스크 스케줄러는 더 우선순위가 높은 입출력 요구가 서비스를 기다리고 있는지 확인한 후 있으면 우선순위 단계를 높여서 서비스를 해 준다. 이 알고리즘의 단점은 수용제어 전략을 채택하고 있지 않기 때문에 우선순위가 낮은 요구들이 오랜 시간 서비스를 받지 못해서 종료시한을 초과하는 경우가 발생 할 수 있다는 것이다.

SCAN-EDF는 EDF에서 비슷한 종료기한을 가진

요구들을 SCAN방식으로 처리하여 탐색시간을 줄인 알고리즘이다. 디스크에 대한 입출력 요구들이 도착하면 종료시한에 따라 큐 안에 정렬해 넣고, EDF와 마찬가지로 종료기한이 가장 빠른 것부터 서비스한다. 이때 종료시한이 같은 입출력 요구가 있다면 SCAN의 서비스 순서대로, 즉 요청하는 데이터가 있는 트랙을 순차적으로 탐색하면서 서비스해 준다. 그러나 실제 디스크에 대한 입출력 요구가 같은 종료시한을 갖는 경우가 극히 드물다. 따라서 일반적으로 SCAN-EDF는 EDF처럼 동작하는 경우가 대부분이다. 여전히 EDF처럼 디스크의 효율성이 떨어지는 단점을 가진다.

SSEDO/SSEDV(Shortest Seektime and Earliest Deadline by Ordering/Value)는 Chen에 의해 제안된 알고리즘으로 디스크 큐 속에 있는 입출력 요구들을 종료시한에 따라 정렬하고, 알고리즘의 수행시간을 줄이기 위해 m개의 선두 입출력 요구들로 정의된 윈도우에서 스케줄링 결정을 내린다. 이 두 알고리즘은 각기 다른 방법을 이용하여 윈도우 내의 입출력 요구들에게 우선순위 값을 부여하고 우선순위 값이 가장 작은 것부터 서비스한다. 큐 안에 정렬되어 있는 입출력 요구들 중에 i번째 요구의 종료시한을 d_i이라 하고 현재 헤드의 위치로부터의 탐색거리를 l_i이라 하면, 우선순위 P_i는 다음과 같이 정의 된다.

식 (1)의 방법에 따라 윈도우 내의 입출력 요구들의 우선순위 값을 정한 후 그 값이 가장 작은 것부터 서비스한다. 즉 탐색거리가 짧고 종료시한이 빠를수록 높은 우선순위를 갖는다. SSEDO/SSEDV는 종료시간을 초과한 입출력 요구라도 빠른 시간 이내에 서비스를 해 주어야 평균 종료시한 초과시간을 줄일 수 있기 때문에, 종료시한을 초과한 입출력 요구를 서비스할 필요가 없는 경성 실시간 시스템에 적용하기에는 무리가 있다.

UG-SSTF(Urgent Group and Shortest Seek Time First)는 디스크 입출력 요구들을 긴급도의 정도에 따라 세 개의 그룹으로 구분하고 같은 그룹에 속한 입출력 요구들을 탐색시간이 작은 것부터 서비스를 해 주는 알고리즘이다. 이미 종료시한을 초과한 입출력 요구들은 MG(Miss Group)에 속하게 되고 긴급한 입출력 요구들은 UG(Urgent Group), 그리고 종료시한까지 시간이 많이 남아있는 입출력 요구들은 DG(Deferable Group)에 속하게 된다. 이 알고리즘은 입출력 요구들을 세 개의 그룹으로 나누어서 서비스함으로써 낮은 종료시한 실기 율을 달성하고 연성 실시간 시스템의 전형적인 성능 요구 사항인 실기율과 종료시한 초과 시간 간의 이해득실을 제어하는 메커니즘을 제공한다. 하지만 UG-SSTF 역시 앞에서 살펴본 SSEDO/SSEDV와 같이 경성 실시간 시스템에 적용하기에는 무리가 있다.

MSG(Maximum Scannable Group)는 SCAN-EDF가 비슷한 종료시한을 가진 요구들만 SCAN방식으로 처리하는 문제점을 개선한 확장 SCAN그룹 정의한다. SCAN그룹이란 종료시한을 초과하지 않는 범위에서 SCAN기법으로 서비스될 수 있는 요구들의 집합을 의미한다. EDF로 스케줄링 된 디스크 요구들의 집합 $R_{EDF}=\{R_{EDF(1)}, R_{EDF(2)}, \cdots, R_{EDF(n)}\}$이 존재할 때, 이 요구들의 집합의 확장 SCAN그룹 MSG을 만들면 i번째 확장 SCAN그룹 MSG_i는 식 (2)와 같이 정의한다. MSG는 연속된 디스크 요구들만 SCAN그룹으로 묶을 수 있는 제약을 가진다.

DMS(Deadline Modification SCAN)는 먼저 MSG에 포함된 요구들 중에서 종료시한이 MSG가 끝나는 시간, 즉 MSG의 종료시간보다 큰 요구들의 종료시한을 MSG의 종료시간으로 변경한다. 그리고 각각의 요구들의 종료시한을 비교해서, 먼저 서비스되는 요구들이 나중에 서비스되는 요구들보다 항상 종료시한이 작거나 같게 유지한다. 이 결과 더 많은 디스크 요구들을 서비스 할 수 있다. 그러나 DMS 알고리즘 역시 MSG가 가지는 연속된 요구들만 SCAN그룹으로 묶을 수 있다는 단점을 가진다. 또한 SCAN그룹과 SCAN그룹 사이에 방향이 고정되어 있어서 시간적 손실이 생길 수 있다.

이상에서 살펴본 실시간 디스크 스케줄링 알고리즘에서 SCAN-EDF나 MSG, DMS등을 이용해서 서비스를 하면 모든 실시간 요구를 종료시한 이내에 서비스해 줄 수 있을 뿐만 아니라 요구의 응답

$$\begin{aligned}
&\text{SSEDO:} \quad P_i = w_i \times d_i, i = 1, 2, ..., m \\
&\qquad\qquad where\ w_i = \beta^{i-1}(\beta \geq 1), i = 1, 2, ..., m. \\
&\text{SSEDV:} \quad P_i = \alpha \times d_i + (1-\alpha) \times l_i\ (0 \leq \alpha \leq 1), i = 1, 2, ..., m.
\end{aligned} \tag{1}$$

$$MSG_k = R_{EDF(i)}, R_{EDF(i+1)}, \cdots, R_{EDF(i+m)}$$
$$which\ satisfies: \quad f_k \leqq d_i\ and\ r_k \leqq e_i$$
$$for\ k = i\ to\ i+m. \tag{2}$$

시간도 상당히 줄일 수 있다. 그러나 여전히 앞에서 살펴본 SCAN그룹 결정문제나 SCAN의 서비스 방향문제가 존재한다.

본 논문에서는 이러한 SCAN그룹 결정문제나 SCAN의 서비스 방향문제를 해결해서 디스크의 효율성을 높임으로써 단위 시간당 더 많은 수의 실시간 입출력 요구를 서비스 해줄 수 있고 디스크의 단위 시간당 처리량을 높일 수 있는 삽입기법과 양방향 SCAN기법을 기반으로 하는 새로운 실시간 디스크 스케줄링 알고리즘을 제안한다.

Ⅲ. 제안알고리즘

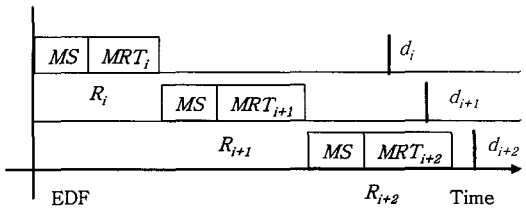
본 논문에서 제안하는 실시간 디스크 스케줄링 알고리즘은 SCAN그룹을 결정할 때 연속하지 않는 요구일지라도 적절하게 SCAN그룹에 삽입할 수 있는 삽입 기법, 연속된 SCAN그룹을 합병할 수 있는 SCAN그룹 합병 기법, 마지막으로 SCAN방향을 효과적으로 결정할 수 있는 양방향 SCAN기법으로 구성된다.

3.1 문제 정의

디스크 입출력 요구의 디스크 접근시간은 식 (3)과 같이 탐색시간(seek time), 회전 지연시간(rotational latency time), 데이터 전송시간(data transfer time)의 합으로 표현한다.

$$\text{access time} = \begin{matrix}\text{seek time} \\ + \text{ rotational latency time} \\ + \text{ data transfer time}\end{matrix} \tag{3}$$

이때, 탐색시간은 접근하려는 트랙으로 디스크 암(arm)을 이동 시키는데 걸리는 시간이고, 회전 지연시간은 접근될 섹터가 판독 헤드 아래에 오는데 걸리는 시간이며, 데이터 전송시간은 전송 섹터 수에 선형적으로 비례한다. 실시간 디스크 요구 $R_i\{r_i, d_i, a_i, b_i\}$에서 데이터의 트랙위치 a_i는 탐색시간, 데이터용량 b_i는 데이터 전송시간과 각각 관련이 있다. 전송시간을 T_i(data Transfer time), 최대 회전 지연시간을 MR(Maximum Rotational latency time),



MRT_i= MR (Maximum Rotational latency time) + T_i (Transfer time)

그림 1. EDF 스케줄링

실제 탐색시간을 S_i, 최대 탐색시간을 MS(Maximum Seek time)로 각각 정의한다. 그리고 MR과 T_i의합을 MRT_i로 정의한다.

그림 1에서 실시간 디스크 요구의 집합 $R = \{R_i, R_{i+1}, R_{i+2}\}$을 EDF기법으로 스케줄링 한 결과를 보인다. 여기서 MSG_i는 $\{R_i, R_{i+1}, R_{i+2}\}$이다.

그림 1에 보이는 것과 같이 MRT_i의 T_i부분은 줄일 수 없고, MR을 줄인다 하더라도 크게 디스크의 성능 개선을 기대할 수 없다. 결과적으로 디스크 요구마다 MS가 존재하고 요구에 독립적이기 때문에, MS의 크기 및 개수를 줄임으로써 디스크의 성능을 향상 시킬 수 있다. 표 1에서 본 논문에서 사용된 파라메타를 보인다.

표 1. 실시간 스케줄링 관련 파라메타

R	a set of requests $R = \{R_0, R_1, ..., R_n\}$
R_i	a real- time disk requests, $R_i = \{r_i, d_i, a_i, b_i\}$
r_i	release time of R_i
d_i	deadline of R_i
a_i	the track location of R_i
b_i	the data capacity of R_i
c_{ji}	the service time of R_i when it's served after R_j
e_i	the start-time of R_i
f_i	the fulfill-time of R_i
R_Z	the schedule result of R by the algorithm Z, $R_Z = R_{Z(0)}, R_{Z(1)}, ..., R_{Z(n)}$.
$f_{Z(n)}$	the schedule fulfill-time of R_Z
MSG_i	the maximum-scannable-group started from R_i
m_i	the count of request of MSG_i
T_i	the data Transfer time of R_i
S_i	the Seek time of R_i
MS	the Maximum Seek time
MR	the Maximum Rotational latency time
MRT_i	$MR + T_i$
α	the sum of time for stop a disk head and accelerate a disk head.

3.2 삽입 기법

본 논문에서는 SCAN그룹을 결정할 때, SCAN그룹의 종료시간을 초과하지 않는 범위에서 연속되지 않은 적당한 요구를 삽입할 수 있는 삽입 기법을 제시한다. 각각의 실시간 디스크 요구들은 독립적이고, i번째 MSG가 $MSG_i = \{R_i, ..., R_n\}$이며, $i \leq n$일 때, MSG의 탐색시간은 최대 $MS + (\alpha \times m_i)$이다. 이때, α는 디스크헤드를 멈추는 시간과 헤드를 가속하는 시간의 합이며, m_i는 MSG_i에 속한 요구들의 개수이다. 또한 MSG는 SCAN방식으로 정렬된 요구의 그룹이기 때문에 $a_i < a_n$이며, MSG_i에 속한 모든 요구들의 데이터 트랙은 a_i와 a_n사이에 위치하게 된다. MS는 디스크헤드의 최대 탐색시간이므로, MSG_i에 속한 모든 요구들의 탐색시간(S)의 합은 식 (4)와 같이 MS보다 작거나 최대 같다.

$$MS \geq \sum_{k=i}^{n} S_k \tag{4}$$

그러나 실제 탐색시간은 α의 시간이 각각의 요구마다 존재하기 때문에, 식 (4)를 식 (5)와 같이 수정한다.

$$MS + (\alpha \times m_i) \geq \sum_{k=i}^{n} S_k \tag{5}$$

MSG_i의 실제 탐색시간은 $MS + (\alpha \times m_i)$이지만 EDF에서 고려하는 MSG의 탐색시간은 $m_i \times MS$이기 때문에, MSG_i의 모든 요구들을 서비스하고 나면 $m_i \times MS$에서 $MS + (\alpha \times m_i)$을 뺀 차이만큼 시간적 여유가 생긴다. 즉, 이 여유시간 만큼 MRT를 가진 요구를 삽입해도 MSG에 속하는 모든 요구들은 종료시간을 어기지 않는다. 그러나 MSG와 다음 MSG 사이에 헤드가 이동하는 시간이 존재하기 때문에, 한 번 더 MS을 더해 줘야한다. 또한, α의 값은 아주 작기 때문에 $\alpha \times m_i$는 MS보다 작기 때문에 결과적으로 식 (5)의 $MS + (\alpha \times m_i)$는 $3 \times MS$로 변경한다. 삽입 가능한 요구의 MRT값은 식 (6)에서 보이는 것과 같이, MS와 $(m_i - 3)$을 곱한 값이 된다.

$$\begin{aligned} &\text{time of insert request} \\ &= MS \times m_i - 3 \times MS \\ &= MS \times (m_i - 3) \\ &where\ MSG_i = \{R_i, ..., R_n\}\ and\ i < n \end{aligned} \tag{6}$$

3.3 SCAN그룹 합병 기법

식 (6)에서와 같이 m_i의 값이 크면 클수록 더 좋은 성능을 보일 수 있다는 것을 알 수 있다. 즉, 서로 다른 MSG를 합병시킬 수 있는 알고리즘을 이용하면 더욱 디스크 성능을 개선시킬 수 있다는 것이다. 연속된 MSG를 하나의 SCAN그룹으로 합병시킬 때, 합병된 SCAN그룹의 m값은 합병되는 MSG의 m값의 합이 된다. 본 논문에서는 삽입 기법과 MSG에 의해 생성된 SCAN그룹의 마지막 요구의 트랙위치가 연속된 다음 SCAN그룹의 처음 요구의 트랙위치보다 뒤이면 두 그룹을 합병하는 전략을 사용한다. 즉, MSG_i의 마지막 요구(R_j)의 트랙위치 a_j 크기가 연속하는 MSG_k 처음 요구(R_k)의 트랙위치 a_k 보다 앞이면, 두 MSG는 합병할 수 있다. 그림 2에서 SCAN그룹 합병 알고리즘을 보인다. 각각의 SCAN그룹들은 이미 SCAN방식으로 정렬되어 있기 때문에 합병된 SCAN그룹을 추가로 정렬할 필요는 없다.

```
which satisfies : MSG_i = {R_i, ..., R_j} and MSG_k = {R_k, ..., R_n}
                  MSG_k is MSG next to MSG_i
i = 1
while( MSG_k exists )
{
        // R_j is last request of MSG_i
        // R_k is start request of MSG_k
         if(a_j < a_k)
                merge MSG_i with MSG_k
        MSG_i is next MSG
}
```

그림 2. SCAN그룹 합병 알고리즘

```
which satisfies : MSG_i = {R_i, ..., R_j} and MSG_k = {R_k, ..., R_n}
                  MSG_k is MSG next to MSG_i
i = 1
while( MSG_k exists )
{
        // R_i is start request of MSG_i
        // R_j is last request of  MSG_i
        // R_k is start request of MSG_k
        // R_n is last request of  MSG_k

        if( distance from a_j to a_k > distance form a_j to a_n )
                deal with a both direction SCAN
        else
                fixed SCAN
        MSG_i is next MSG
}
```

그림 3. 양방향 SCAN기법의 알고리즘

3.4 양방향 SCAN기법

$MSG_i = \{R_i, ..., R_j\}$일 때, R_i으로부터 R_j로 서비스 하거나, 역방향 R_j으로부터 R_i로 서비스 하던지 전체 실행시간($f_{MSG(n)}$)은 변화가 없다. MSG_i 다음 연속된 MSG가 $MSG_k = \{R_k, ..., R_n\}$일 때, 고정된 SCAN기법은 a_i으로부터 a_j, a_k, a_n의 순서로 헤드를 이동하면서 서비스 한다. 이때 a_n에서 a_j의 거리가, a_k에서 a_j의 거리보다 가깝다면, a_j 다음에 a_n로 서비스 하는 것보다 a_k로 서비스 하는 것이 더 효과적이다. 즉, 고정된 SCAN기법 보다 가변적인 양방향 SCAN기법이 디스크 성능을 더 향상 시킬 수 있다. 그림 3에서 양방향 SCAN 알고리즘을 보인다.

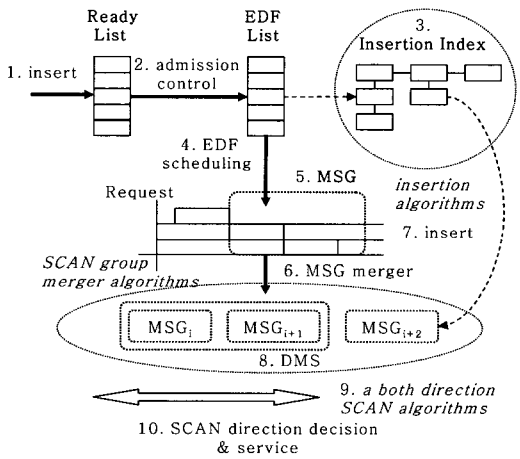


그림 4. 제안 알고리즘 전체 구성도 및 수행순서

3.5 제안 알고리즘의 수행순서

그림 4는 제안 알고리즘의 전체 구성도이며 수행순서는 다음과 같다.

먼저, 실시간 디스크 요구가 들어오면 수용제어를 실시한다. 수용제어가 완료된 요구들의 각각의 삽입 인덱스를 구성하고, EDF기법으로 스케줄링 한다. 스케줄링 된 요구들의 MSG를 구성한다.

그리고 MSG를 SCAN합병기법을 이용하여 적당한 MSG들을 합병하고, 삽입 인덱스를 이용한 삽입기법으로 수정된 MSG를 구성한다. 수정된 MSG의 요구들의 종료시간을 DMS기법을 이용하여 수정하고, 마지막으로 양방향 SCAN기법을 이용하여 스캔의 방향을 결정한다. 이를 단계별로 자세하게 기술하면 다음과 같다.

① 대기 리스트 삽입: 도착하는 모든 입출력 요구들(실시간 입출력 요구)을 대기 리스트에 넣는다.

② 수용 제어: 대기 리스트에 있는 요구들 중에서 현재 스케줄링 알고리즘에서 제공하고 있는 수용제어 방식을 적용하여 만족하는 요구들만 EDF리스트에 삽입한다. 처음에는 EDF와 비슷한 수의 요구를 수용하지만, 삽입 실시간 디스크 스케줄링과 종료시간을 수정하는 DMS기법을 적용하기 때문에, 다음 수용제어를 할 때 더 많은 수의 요구를 수용 할 수 있다.

③ 삽입 인덱스 생성: EDF리스트에 속한 요구의 집합이 $R = \{R_1, ..., R_n\}$이고, i가 $0 \leq i \leq n$

이면, 요구(R_i)의 MRT_i을 MS와 비교하여 MS의 배수에 따라 삽입 인덱스를 생성한다. 삽입 인덱스는 삽입 삭제가 용이한 링크 더리스트로 구성하며, 이때 내부리스트는 요구(R_i)의 EDF리스트의 위치를 나타낸다. 삽입 인덱스는 쓰레드로 작성하여 스케줄링이 실행되는 중에도 요구가 들어오면 계속 인덱스를 생성하고, 또한 요구가 MSG에서 포함될 때 요구의 인덱스를 삭제한다. 그림 5에서 삽입 인덱스의 구성을 보인다.

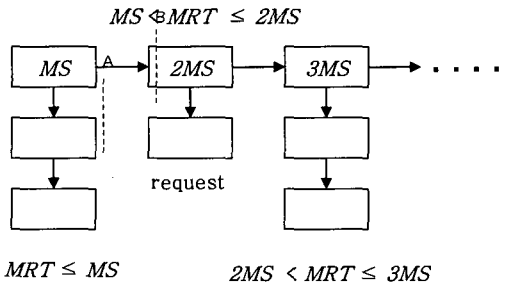


그림 5. 삽입 인덱스의 구성

④ EDF 스케줄링: 종료시한이 가장 빠른 요구 순으로 EDF리스트에 있는 요구들을 EDF로 스케줄링 한다. 여기서 EDF리스트는 아직 수용제어만 행해진 리스트이다. EDF로 스케줄링 할 때 종료시한 순으로 스케줄링이 된다.

⑤ SCAN그룹(MSG) 생성: EDF로 스케줄링이 된 요구들을 가지고 MSG를 생성한다. 요구가 MSG에 포함될 때 삽입 리스트에서 요구의 인덱스를 삭제함으로써 스케줄링이 된 요구가 다시 삽입되는 것을 방지한다. MSG는 순차적으로 생성하며, 두개이상의 MSG가 생성되면 SCAN그룹 합병기법을 적용시킨다.

⑥ SCAN그룹 합병 기법: 두개이상의 MSG가 생성되면 SCAN그룹 합병 기법을 적용하여 가능한 MSG들을 합병한다.

⑦ 삽입 기법: MSG들을 합병 하고나면, 삽입인덱스에서 적당한 요구를 삽입기법으로 삽입한다. 삽입할 때는 먼저 가능한 가장 큰 요구부터 삽입한다. 이때 삽입가능 한 태스크의 MRT크기는 최대 $MS\times(m_i-3)$이기 때문에, MSG에 속한 요구의 개수가 3개 이하이면 삽입기법을 적용하지 않는다. 마지막으로 삽입 인덱스에서 삽입된 요구의 인덱스를 삭제한다.

⑧ DMS: 합병 삽입된 MSG에 DMS기법을 적용시켜 EDF스케줄링 순서를 유지시킨다.

⑨ 양방향 SCAN기법: DMS기법으로 마감시간이 수정된 연속된 MSG들의 SCAN방향을 결정한다.

⑩ 서비스: 마지막으로 양방향 SCAN기법에 의해 결정된 방향대로 MSG를 서비스하고 EDF리스트에 요구가 존재하면 ④번부터 다시 반복한다.

Ⅳ. 실험 및 성능평가

본 논문에서는 제안알고리즘의 성능을 평가하기 위해서 시뮬레이션 도구로 CSIM18 시뮬레이션 엔진과 MFC6.0을 이용한다. 실험에서 실시간 입출력 요구는 주기적인 특성을 갖지 않고 무 작위적으로 디스크 블록을 요청한다고 가정한다. 또한 실시간 입출력 요구는 한번에 8KByte씩의 데이터를 요청한다고 가정하고 실험을 수행한다. 실험에 이용한 환경설정[13, 14]은 표 2와 같다.

표 2. 실험에 이용한 환경설정

실린더 개수	6176
트랙 수 /실린더	4
섹터 수/트랙	126
회전 시간	8.3×10-3
최대 탐색시간	15×10-3
평균 탐색시간	8×10-3
디스크 드라이브의 전송속도	4 MByte/sec

실시간 디스크 스케줄링 알고리즘의 성능을 평가하는 가장 중요한 요소 중 하나는 종료시간을 초과하지 않는 범위 내에서 단위 시간당 서비스 가능한 입출력 요구가 몇 개인가 하는 것이다. 먼저, 요구를 전송하는 전송시간을 계산하면 식 (7)과 같다.

$$T(data\ transfer\ time) = \frac{8KBytes}{4MBytes/sec} = 2ms \tag{7}$$

먼저, 본 논문에서 제안한 알고리즘을 이용했을 때의 디스크 효율성을 평가하기 위해 정해진 시간에 얼마나 많은 입출력 요구의 수를 서비스하는가를 평가하였다. 그림 6은 입출력 요구 개수를 변화시키면서 처리된 요구의 개수를 측정한 것이다.

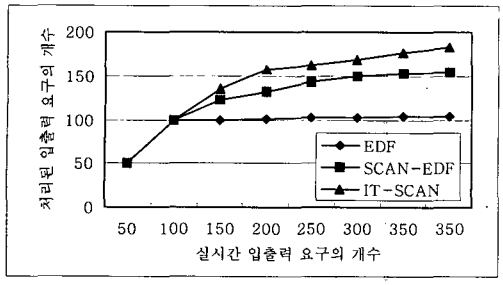


그림 6. 정해진 시간에 처리된 입출력 요구의 개수

그림 6에서 "A"지점에서 "B"지점까지는 세 개의 알고리즘 모두 처리된 입출력 요구의 개수가 비슷하다. 이는 입출력 요구의 개수가 적을 경우 입력 요구를 세 알고리즘 모두 처리할 수 있기 때문이다. B지점 이후에서 EDF가 처리하는 요구의 개수가 고정되어 조금 증가하는 것은 입출력 요구의 탐색시간을 최대 탐색시간으로 처리하여 디스크의 효율성이 떨어지기 때문이다.

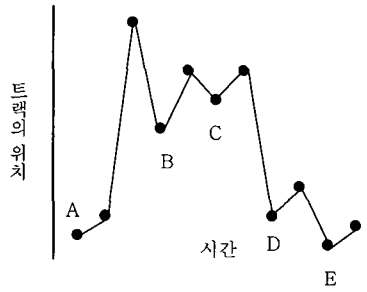
반면, SCAN-EDF나 본 논문에서 제안한 IT-SCAN(Insertion technique and Two-way SCAN technique base real-time disk scheduling)의 경우에는 적당한 요구들을 SCAN그룹으로 묶어서 처리하기 때문에 처리되는 입출력 요구의 개수는 들어온 요구의 개수에 따라 증가한다. IT-SCAN의 증가가 SCAN-EDF보다 큰 것은 삽입기법을 이용하여 연속하지 않은 요구라도 SCAN그룹으로 묶을 수 있어 보다 큰 SCAN그룹을 만들 수 있기 때문이다.

또한, 양방향 SCAN기법으로 처리하여 디스크 효율성을 높였기 때문이다.

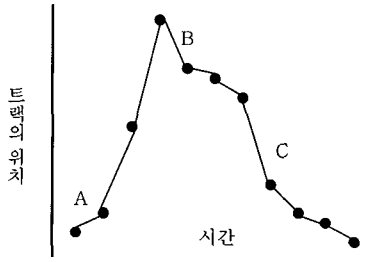
그림 6의 실험에서 알 수 있듯이 정해진 시간에 요구되어지는 입출력 요구의수가 적을 때는 EDF나 SCAN-EDF, IT-SCAN 모두 처리되는 요구의수가 비슷하지만, 입출력 요구수가 조금씩 증가함에 따라 IT-SCAN이 EDF나 SCAN-EDF보다 처리되는 요구의 개수가 많아지는 것을 알 수 있다. 이는 IT-SCAN이 디스크를 보다 효율적으로 이용한다는 것을 나타낸다.

그림 7은 SCAN-EDF와 IT-SCAN을 이용할 때 처리된 요구의 개수가 차이나는 원인을 보이고 있다. 그림 7의 (a)와 (b)는 각각 SCAN-EDF와 IT-SCAN을 이용해서 임의의 10개의 실시간 입출력 요구를 서비스 할 경우 디스크 헤드의 움직임을 보인다.

각각의 스캔 그룹의 시작점은 (a)의 경우는 "A, B, C, D, E"이며 (b)의 경우는 "A, B, C"이다. 즉



(a) SCAN-EDF의 경우



(b) IT-SCAN의 경우

그림 7. SCAN-EDF와 IT-SCAN의 헤드의 움직임 비교

SCAN-EDF을 이용하면 SCAN그룹이 5개가 되지만 IT-SCAN의 경우 삽입기법을 이용하여 3개의 SCAN그룹만으로 묶을 수 있다. "A"의 SCAN그룹에서 "B"의 SCAN그룹으로 서비스할 경우 SCAN-EDF의 경우는 SCAN그룹사이의 SCAN방향이 고정되었기 때문에 "A"의 끝에서 "B"의 끝으로 서비스 하지 못한다.

이와 달리 IT-SCAN은 양방향 SCAN의 지원하기 때문에 B의 SCAN그룹은 역방향으로 서비스 할 수 있다. 결과적으로 헤드의 진행 방향의 움직인 거리를 보면 SCAN-EDF가 IT-SCAN에 비하여 많으므로 그만큼 디스크의 효율성이 떨어지기 때문에 처리된 요구의 개수가 차이가 나게 된다.

그림 8은 EDF와 SCAN-EDF, IT-SCAN을 이용하여 실시간 입출력 요구를 서비스했을 때 총 수행시간을 보이고 있다. 여기서 총 수행시간이란 실시간 입출력 요구들의 마감시간을 매우 크게 설정했을 때 걸리는 총 실행 시간을 측정한 것이다.

그림 8을 보면 세 개의 알고리즘 모두 요구된 실시간 입출력 요구의 개수에 비례하여 증가하는 것을 볼 수 있다. 그러나 EDF보다는 SCAN-EDF가 총 수행 시간이 적은 이유는 요구의 개수가 증가할수록 SCAN-EDF의 경우 비슷한 종료시한을 가진 요구수가 증가 하고, 따라서 SCAN기법으로 처리되는 요구의 수도 증가하기 때문이다.

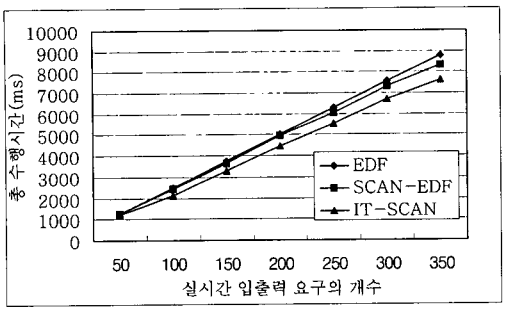


그림 8. 총 수행시간

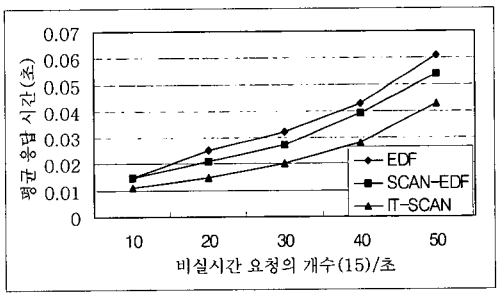


그림 9. 비실시간 I/O 요구의 응답시간 비교 (15)

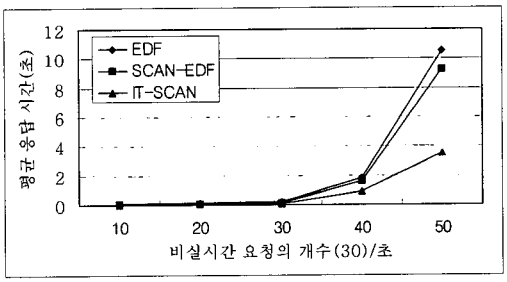


그림 10. 비실시간 I/O 요구의 응답시간 비교 (30)

SCAN-EDF보다 IT-SCAN의 총 수행 시간이 적은 이유는 삽입기법을 이용하여 SCAN-EDF보다 더욱 큰 SCAN그룹을 생성시킬 수 있고 양방향 SCAN 기법으로 SCAN그룹 SCAN그룹사이의 처리방향을 효과적으로 결정할 수 있기 때문이다. 종료시한이 아주 큰 입출력 요구는 비 실시간 요구의 특성을 가지기 때문에 결과적으로 비 실시간 입출력 요구의 디스크 스케줄링에도 충분히 효과적이라는 것을 보여준다.

그림 9와 그림 10은 EDF와 SCAN-EDF, IT-SCAN을 이용하여 서비스했을 경우 비 실시간 입출력 요구의 응답 시간을 비교한 실험 결과이다. IT-SCAN의 경우 비 실시간 요구를 SCAN그룹에 우선적으로 삽입한다. 그림 9는 초당 발생하는 실시간 입출력 요구의 수가 15개일 때 비 실시간 입출력 요구의 응답 시간을 나타낸 것이다.

EDF나 SCAN-EDF는 실시간 입출력 요구를 우선적으로 서비스하고 비 실시간 입출력 요구를 서비스하기 때문에 비 실시간 요구가 삽입될 수 있는 IT-SCAN에 비하여 응답 시간이 더 크게 나온다.

그림 10에 보인바와 같이 실시간 입출력 요구의 개수를 30으로 증가시키면 그 차이는 더욱 두드러진다. 이 실험 결과를 보면 실시간 입출력 요구의 수가 적을 때에는 비 실시간 입출력 요구의 응답 시간의 차이가 작지만 실시간 입출력 요구의 수가 많아지면 비 실시간 입출력 요구의 응답 시간의 차이는 상당히 커지게 됨을 알 수 있다.

삽입 기법과 양방향 SCAN기법 기반 실시간 디스크 스케줄링 알고리즘을 EDF나, SCAN-EDF와 비교했을 때 문제가 되는 부분은 알고리즘의 복잡도이다. 제안 알고리즘은 실시간 입출력 요구가 새로 들어오거나 서비스될 때마다 추가적인 계산이 필요하다. 하지만 시간이 흐름에 따라 프로세서의 속도는 빠른 속도로 발전하고 있는 반면에 디스크의 상대적으로 발전 속도가 느린 현실을 감안해 보면 알고리즘의 복잡도로 인한 성능 저하는 큰 문제가 되지 않는다.

V. 결 론

본 논문에서는 기존의 경성 실시간 시스템에서 적용 가능한 실시간 디스크 스케줄링 알고리즘들의 문제점을 해결하기 위해 삽입 기법과, 양방향 SCAN 기법을 기반으로 한 새로운 실시간 디스크 스케줄링 알고리즘을 제안하였고 성능을 평가하기 위하여 대표적인 경성 실시간 디스크 스케줄링 알고리즘인 EDF, SCAN-EDF와 각각 비교평가 하였다. 실험 결과에 따르면 삽입 알고리즘과 양방향 SCAN알고리즘은 단위 시간당 서비스 가능한 입출력 요구의 수가 EDF나 SCAN-EDF보다 월등한 성능을 보이고 있다. 또한, 실시간 입출력 요구의 총 수행시간을 보면 EDF나 SCAN-EDF보다 성능이 월등하게 좋다는 것을 알 수 있었다. 실시간 입출력 요구 뿐만아니라 비 실시간 입출력 요구의 평균 응답시간 또한 다른 알고리즘보다 성능이 월등하며, 요구의 수가 늘어나게 되면 이러한 편차는 더욱 커지는 현상이 있었다. 그 이유는 삽입기법을 이용하여 SCAN-EDF보다 더욱 큰 SCAN그룹을 생성시킬 수 있고 양방향 SCAN기법으로 SCAN그룹 SCAN그룹사이의 처리방향을 효과적으로 결정할 수 있기 때문이다.

참 고 문 헌

[1] P. J. Denning, "Effect of Scheduling on File Memory Operations," In Proceedings of AFIPS SJCC, volume 30, pages 9-21, 1967.

[2] T. J. Teorey, "Properties of Disk Scheduling Policies in Multi programmed Computer Systems," In Proceedings of AFIPS SJCC, volume 41, pages 1-11, 1972.

[3] N. C. Wilhelm, "An Anomaly in Disk Scheduling: a Comparison of FCFS and SSTF Seek Scheduling Using an Empirical Model for Disk Accesses," CACM, 19(1), pages 13-17, 1972.

[4] M. Hofri, "Disk Scheduling: FCFS vs. SSTF Revisited," CACM, 23(11) pages 9-21, 1967.

[5] B. Kao and R. Cheng, "Disk Scheduling," In REAL-TIME DATABASE SYSTEMS, pages 97-107, Kluwer, 2002.

[6] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment," Journal of ACM, pages 46-61, 1973.

[7] M. J. Carey, R. Jauhari, and M. Livny, "Priority in DBMS Resource Scheduling," In Proceedings of the 15th VLDB Conf, pages 397-410, 1989.

[8] A. L. N. Reddy and J. Wyllie, "Disk Scheduling in a Multimedia I/O System," In Proc. of the first ACM International Conference on Multimedia, pages 225-233, Anheim, CA, 1993.

[9] S. Chen, J. A. Stankovic, J. F. Kurose and D. Towsley, "Performance Evaluation of Two New Disk Scheduling Algorithms for Real-Time Systems," The Journal of Real-Time Systems, 3(3), 1991.

[10] K. Hwang and H. Shin, "Real-time disk scheduling based on urgent group and shortest seek time first," in Proceedings of 5th EUROMICRO Workshop on Real-Time Systems, page 124-130, 1993.

[11] H. P. Chang, R. I. Chang, W. K. Shih and R. C. Chang, "Enlarged-Maximum- Scannable-Groups for Real-Time Disk Scheduling in Multimedia System," IEEE COMPSAC, pages 383-388, 2000.

[12] R. I. Chang, W. K. Shih and R. C. Chang, "Deadline-Modification-SCAN with Maximum-Scannable-Groups for Multimedia Real-Time Disk Scheduling," Proc. IEEE RTSS, pages 40-49, 1998.

[13] C. Ruemmler and J. Wilkes, "An introduction to disk drive modeling," IEEE Computers, pages 16-28, 1994.

[14] R. P. King, "Disk arm movement in anticipation of future requests," ACM Trans. Computer Systems, volume 8, no. 3, pages 214-229, 1990.

이 명 섭 (Myung Sub Lee) 정회원

1998년 2월 경일대학교 컴퓨터공학(공학사)

2000년 2월 영남대학교 대학원 컴퓨터공학과(공학석사)

2003년 8월 영남대학교 대학원 컴퓨터공학과(공학박사)

2003년 3월~현재 영남대학교 전자정보공학부 객원교수

<관심분야> 망관리, 에이젼트, QoS

박 창 현 (Chang Hyeon Park) 정회원

1986년 경북대학교 전자공학과(공학사)

1988년 서울대학교 계산통계학과 전산학전공(이학석사)

1992년 서울대학교 계산통계학과 전산학전공(이학박사)

1992년~1993년 서울대학교 컴퓨터신기술공동연구소 특별연구원

1998년~1999년 University of Maryland, Institute of Advanced Computer Systems, Visiting Researcher

1993~현재 영남대학교 컴퓨터공학과 부교수

<관심분야> 인공지능, 에이젼트, 지능형 망관리