DOI: 10.3745/KIPSTA.2010.17A.6.259

다중프로세서 시스템상의 공유 자원을 포함하는 태스크를 위한 실시간 스케줄링 알고리즘

이 상 태[†] · 김 용 석[††]

요 약

다중 프로세서 상의 공유 자원을 포함하는 태스크를 스케줄링 하는데 있어서 마감 시간을 기준으로 스케줄링 하는 EDF (Earliest Deadline First) 를 그대로 적용한 GEDF (Global EDF) 알고리즘은 공유 자원을 가지는 태스크에 대한 처리가 없어서 스케줄링 성공률이 떨어지게 된다. 본 논문에서는 공유 자원을 포함하는 태스크를 위해 태스크를 공유 자원을 접근하는 임계영역을 포함하는 부분과 그렇지 않은 부분으로 각각 나누어 개별적인 마감 시간을 부여해 처리하는 EDFP (Earliest Deadline First with Partitioning) 알고리즘을 제안한다. 시뮬레이션을 통한 평가 결과 공유 자원을 포함하는 태스크들에 대하여 EDFP는 시스템 이용률이 높아지고 프로세서의 개수가 많이 질수록 GEDF 보다 나은 성능을 보인다.

키워드 : 다중프로세서 시스템, 실시간 태스크, 주기적 태스크, 공유 자원

A Real-Time Scheduling Algorithm for Tasks with Shared Resources on Multiprocessor Systems

Sang tae Lee[†] · Young seok Kim[††]

ABSTRACT

In case of scheduling tasks with shared resources in multiprocessor systems, Global Earliest Deadline First (GEDF) algorithm, equally applied Earliest Deadline First (EDF) which runs scheduling with deadline criterion, makes schedulability decline because GEDF typically does not have a specific process in order to handle tasks with shared resources. In this paper, we propose Earliest Deadline First with Partitioning (EDFP) for tasks with shared resources which partitions a task into two kinds of subtasks that include critical sections to access to shared resources, gives their own deadline respectively and manages them. As a result of simulations, EDFP shows better performance than GEDF for tasks with shared resources since system load goes up and the number of processor increases.

Keywords : Multiprocessor System, Realtime Task, Periodic Task, Shared Resource

1. 서 론

하드웨어의 급격한 발전과 더불어 실시간 처리 시스템이 점점 복잡해져감에 따라 다중 프로세서의 필요성이 증대되고 있다. 더욱이 하나의 칩에 다수의 코어를 가지는 프로세서가 보편화됨에 따라 다중 프로세서 환경 또한 대중화 되고 있다. 이에 시스템에 좀 더 효율적인 실시간 태스크의 처리를 위하여 스케줄링 알고리즘에 관한 연구도 활발하게 진행되고 있다. 단일 프로세서의 환경에 대한 실시간 처리 시스템의 연구는 이미 상당히 연구되어 있지만 다중 프로세서 환경에 대해서는 아직 많은 연구가 필요한 실정이다. 다중 프로세서의 환경은 어떠한 프로세서를 선택하여 처리하며 프로세서와 프로세서간의 태스크 이주 현상에 대한 정책에 따라 스케줄링 성공률이 영향을 받게 되어 단일 프로세서에서의 연구를 그대로 적용 시킬 수 없다. 더욱이 다중 프로세서 상에서 공유 자원을 포함하는 스케줄링 정책에 대해서는 연구가 미흡한 실정이다.

가장 널리 알려진 EDF(Earliest Deadline First)를 다중 프로세서 시스템에 적용한 것이 GEDF(Global EDF) 이다. 본 논문에서는 EDF를 기초로 하여 공유 자원 처리에 초점을 맞춘 EDFP(Earliest deadline first with Partitioning) 알고리즘을 제안하고 그에 대한 성능을 평가하였다. EDFP는

† 정 회 원 : 강원대학교 컴퓨터정보통신공학부 석사과정
†† 정 회 원 : 강원대학교 컴퓨터학부 교수(교신저자)
논문접수 : 2010년 8월 30일
수 정 일 : 1차 2010년 11월 8일
심사완료 : 2010년 11월 9일

공유 자원을 가지는 태스크에 대해서 공유 자원을 접근하는 임계영역을 포함하는 부분과 그렇지 않은 부분을 별개로 생각하여 마치 한 개의 태스크가 여러 개의 부분 태스크들로 이루어 진 것처럼 간주하고 그들에 대해서 별도의 마감 시간을 적용하여 스케줄링 하는 방법을 취하였다. 이것은 공유 자원에 대한 처리로 공유 자원들이 임계 구역을 사용할 때 최대한 긴 시간 동안 여유 있게 임계 구역을 사용하게 하여 태스크들 간의 임계 구역 사용의 중첩 현상을 최소화 하는 방법을 통해 그 스케줄링 성공률을 높이도록 하였다.

본 논문에서는 3장에서 EDFP 알고리즘의 도출 배경 및 세부적인 알고리즘의 내용에 대해 설명하고 4장에서 GEDF와 EDFP를 시뮬레이션을 통해 시스템 로드와 프로세서의 개수에 따라 성능 평가를 한다.

2. 관련 연구

다중 프로세서의 환경은 단일 프로세서 환경과는 다르게 어느 프로세서에서 실행되는가와 우선 배정된 프로세서에서 다른 프로세서로의 태스크 이주 현상과 같은 새로운 고려 사항이 적용되어야 한다. 그렇기 때문에 단일 프로세서에서 최적인 알고리즘이라 할지라도 다중 프로세서의 환경에서는 좋지 못한 성능을 보여주는 경우가 있다. 이에 Carpenter등은 위의 사항들을 고려하여 다중 프로세서 환경에서의 스케줄링 알고리즘을 분류하고 그들 간의 관계를 새로이 정리하였다[1].

일반적으로 자원이라 함은 자료 구조, 변수의 모음, 주 메모리 영역, 파일, 혹은 주변 기기의 레지스터의 모음 등이 될 수 있다. 그러한 자원 중에 하나 이상의 태스크에 사용될 수 있는 것을 공유 자원이라고 하여 공유 자원이 상호 독점적으로만 사용 되어야 한다면 동시 다발적인 프로세서의 접근에 대해 뮤택스 등을 통하여 적절히 보호된다[2].

실시간 스케줄링 알고리즘들 중에 가장 널리 알려져 있으며 보편적으로 사용되고 있는 EDF 알고리즘은 태스크의 마감 시간이 빠른 순서로 태스크들을 스케줄링 하는 알고리즘으로서 이는 단일 프로세서 상에서는 최적이다[3,4]. 마감 시간에 따라 태스크의 순위가 결정되므로 문맥 교환 횟수가 매우 낮으며 알고리즘이 단순함에 따라 다른 스케줄링 알고리즘들에 비해 상대적으로 구현이 용이하다 할 수 있다. 하지만 알고리즘 상에 공유 자원에 관한 처리가 없을 뿐만 아니라 다중 프로세서 환경에서의 특수성 또한 고려하지 않아 적절한 스케줄링이 되지 않는 경우도 존재한다.

EDF와 마찬가지로 LLF(Least Laxity First) 역시 단일 프로세서 환경에서 최적으로 알려진 알고리즘이다[5]. LLF는 태스크들의 여유 시간을 기준으로 여유 시간이 가장 짧은 태스크를 먼저 실행하는 방식을 취하고 있다. 다중 프로세서 환경에서는 스케줄링 성공률에 있어 마감 시간을 기준으로 한 알고리즘 보다 여유 시간을 기준으로 한 알고리즘이 일반적으로 유리하다고[6] 알려진 만큼 다중 프로세서 환경에서도 우수한 스케줄링 성공률을 보여준다. 하지만 여유 시간은 고정 되어있는 마감 시간과는 달리 값이 유동적으로 변하므로 각 태스크들의 상대적인 여유 시간의 차이는 무수히 바뀌게 되며 이에 따라 문맥 교환 횟수가 지나치게 높아지게 되어 현실적인 적용에는 무리가 있다[1]. 뿐만 아니라 여기서의 여유 시간은 태스크의 마감 시간과 계산 시간에 따른 것으로 공유 자원에 대한 고려가 없이 적용되는 것으로 공유 자원에 의해 발생하는 임계 구역의 사용에 대한 처리가 없어 스케줄링상의 불리함을 가지게 된다.

EDZL은 EDF방식을 따름과 동시에 여유 시간이 0 이 되는 태스크에 대한 처리를 더해 성공률을 높인 알고리즘이다[7]. 대기 큐에 여유 시간이 0 인 태스크가 존재하지 않으면 EDF의 스케줄링 정책에 따라 마감시간에 빠른 순서대로 태스크를 처리한다. 하지만 여유 시간이 0 인 태스크가 발생하게 되면 현재 실행중인 태스크들 중 여유 시간이 0 보다 크면서 마감시간이 가장 늦은 태스크와 문맥교환을 하여 실행 할 수 있게 한다. 여유 시간이 0 인 태스크는 바로 실행되지 못하면 마감 시간을 충족시키지 못하기 때문이다. 즉 여유 시간이 0 인 태스크가 발생하지 않는 태스크 집합에 대해서는 EDF와 동일한 스케줄링을 하며 그렇지 않은 태스크 집합에서는 여유 시간이 0 인 태스크가 발생하면 다른 태스크들보다 높은 우선순위를 할당하여 처리하게 된다. 그에 따라 EDF보다 높은 스케줄링 성공률을 가지게 된다. 이와 같은 방식으로 여유 시간이 0 이 되는 태스크에 대한 처리에 LLF방식을 따라 높은 성공률을 유지함과 동시에 문맥교환 횟수를 큰 폭으로 줄인 것이 LLZL 알고리즘이다[8]. LLZL은 LLF가 가지는 스케줄링 성공률의 장점을 크게 감소시키지 않으면서 EDF와 유사한 낮은 문맥교환율을 보인다. 또한 EDZL보다도 우수한 성능을 보여주었다. 하지만 EDZL 과 LLZL 역시 공유 자원을 사용하는 태스크에 대한 처리는 해주지 않고 있다.

PIP(Priority Inheritance protocol)는 우선순위 스케줄링에서 공유 자원의 사용을 고려해 준 우선순위 상속 프로토콜이다. PIP는 공유 자원을 사용하는 여러 개의 태스크가 스케줄링 될 때에 공유 자원에 의한 우선순위 역전 현상을 방지해 준다[9]. 우선순위가 낮은 태스크가 공유 자원의 사용으로 우선순위가 높은 태스크보다 먼저 공유 자원을 사용하는 동안 그 우선순위를 상속 받아 공유 자원을 사용하는 기간 동안만 높은 우선순위를 가지고 공유 자원의 사용이 끝나면 다시 우선순위를 돌려주는 방법으로 전체 태스크에 대한 우선순위 역전 현상을 방지하는 기법이다. 하지만 이는 단일 프로세서 환경을 위한 프로토콜이며 그 초점이 전체 태스크의 스케줄링 성공률이 아닌 우선순위 역전 현상의 방지에 있다.

3. EDFP 알고리즘

본 논문은 M개의 프로세서 상에서 주기적인 태스크의

집합을 대상으로 하는 경성 실시간 스케줄링 기법을 다룬다. 태스크의 집합은 N개의 선점 가능한 주기적인 태스크로 구성되며 태스크는 공유 자원 사용 구간, 마감 시간 그리고 계산 시간 정보를 가지고 있으며. 공유 자원 사용 구간은 계산 시간의 일정 부분을 차지하게 된다.

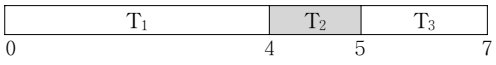
EDFP 알고리즘은 EDF를 기반으로 하여 태스크들에서 공유 자원 사용 구간을 빠르게 처리하는 방식을 통해 전체적인 스케줄링 성공률을 높이는 스케줄링 알고리즘이다. 다중 프로세서 시스템 상에서 EDF와 같이 기존의 공유 자원을 가지지 않는 일반적인 태스크를 처리하는 스케줄링 알고리즘으로는 공유 자원을 처리 해 줄 시에 임계 구역의 사용이 빈번히 발생하게 되어 처리해야 할 태스크가 있는 상황에서도 필요한 태스크가 처리되지 못하고 태스크가 밀리는 현상이 발생하게 된다. 그에 따라 전체적인 스케줄링 성공률이 떨어짐을 막을 수 없게 된다.

이를 극복하기 위하여 EDFP는 태스크 처리에 있어 공유 자원 사용 구간에 최 우선권을 주어서 처리해야 할 공유 자원 사용이 필요한 경우에는 항상 빠르게 임계 구역을 사용하게 한다. 이를 통해 공유 자원 처리를 위한 시간을 좀 더 여유 있게 만들어 이 후에 등장할 공유 자원 사용 구간과의 중첩을 예방해준다. 또한 이를 통한 지속적인 공유 자원 사용 구간의 처리로 임계 구역 사용의 중첩 현상을 최소화한다.

EDFP에서는 각각의 태스크를 공유 자원을 접근하는 임계영역을 포함하는 부분과 그렇지 않을 부분으로 나누고 이들을 독립적인 태스크처럼 EDF를 이용해 처리 하는 방식을 취한다. 이를 위해서 나누어진 태스크들은 각각 별도의 마감 시간을 갖는데 기존 태스크의 마감시간 및 공유 자원 사용시간을 적용하여 결정한다.

(그림 1)에서 태스크 T는 마감 시간 D = 8 를 가지며 T_1 과 공유 자원 사용 구간인 T_2, 그리고 T_3 세 개의 서브 태스크로 나누어져 각각 C_1 = 4, C_2 = 1, C_3 = 2 의 계산 시간을 가진다. T_3는 태스크의 마감 시간인 D 를 그대로 사용, D_3 = D 가 되며 공유 자원 사용 구간인 T_2의 마감 시간 $D_2 = D_3 - C_3$ 로 구하게 된다. 마찬 가지로 T_1의 마감 시간은 태스크의 마감시간에서 T_1이전의 계산 시간을 제외한 $D_1 = D_2 - C_2$ 가 된다. 그리하여 각각 T_1, T_2, T_3는 마감시간 5, 6, 그리고 8 을 갖게 된다.

GEDF와 EDFP를 비교하기 위하여 (그림 2)의 태스크 T_1, T_2, T_3 이 동시에 도착하여 두 개의 프로세서에서 실행한다. GEDF에서 T_1 과 T_2 는 마감 시간을 충족하지만 (그림 2-(a)) 의 T_3 의 (*) 부분처럼 마감 시간을 충족시키지 못하는 태스크가 발생한다. 하지만 (그림 2-(b)) 와 같이 EDFP의 경우는 모든 태스크에 대하여 마감시간을 충족시킬 수 있게 된다.

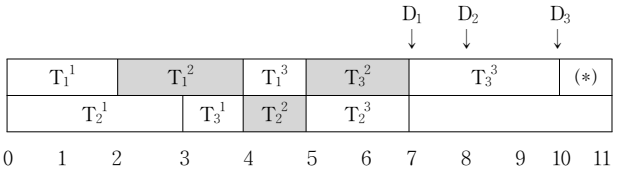


(그림 1) 태스크 분할

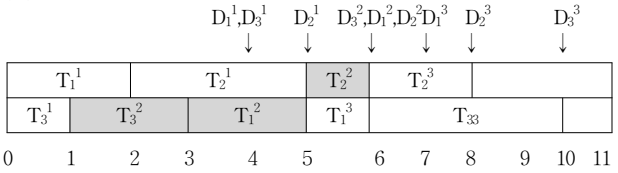
	마감 시간	C_1	C_2	C_3
T_1	7	2	2	1
T_2	8	3	1	2
T_3	10	1	2	4

	마감 시간		마감 시간		마감 시간
T_1^1	4	T_1^2	6	T_1^3	7
T_2^1	5	T_2^2	6	T_2^3	8
T_3^1	4	T_3^2	6	T_3^3	10

(a) GEDF



(b) EDFP



(그림 2) GEDF 와 EDFP의 비교 예

EDFP의 스케줄링 방법은 공유 자원을 사용하는 경우와 공유 자원을 사용하지 않는 경우 두 가지 경우로 나누어 볼 수 있다. 우선 공유 자원을 사용하는 경우에는 항상 공유 자원을 사용하는 태스크를 최우선으로 프로세서에 할당하여 실행 하게 해 주며 나머지 프로세서에서 일반 태스크들을 처리하는 방법을 취한다. 이 때 복수 태스크들이 공유 자원을 사용하려는 경우가 발생하게 된다. 이럴 경우에는 공유 자원을 사용하는 태스크 별개의 마감 시간에 따라서 가장 급한 마감 시간을 가진 공유 자원을 사용하는 태스크를 먼저 프로세서에 할당 되게 된다. 반면 공유 자원을 사용하지 않는 태스크들에 대해서는 상위 우선순위를 가지는 공유 자원을 사용하는 태스크가 존재하지 않게 되므로 EDF와 동일한 방법으로 프로세서에 할당 되게 된다. 다만 위에서 언급한 바와 같이 일반 태스크는 공유 자원을 사용하는 태스크의 처리 순서에 영향을 미칠 뿐만 아니라 직접적인 태스크 마감 시간에도 영향을 미쳐 태스크 성공률을 좌우하게 하므로 일반 태스크들은 전체 태스크의 마감 시간을 이용하지 않고 개별적으로 구해진 마감 시간을 이용하여 프로세서에 할당 되게 된다.

(그림 3)은 EDFP에 대한 알고리즘을 설명하고 있다. 새로운 태스크가 도착하게 되면 이 태스크를 공유 자원을 사용하는 부분과 아닌 부분으로 나누어 진 것으로 간주하고 이를 큐에 삽입하게 된다. 이때 분할된 서브 태스크들은 원래 태스크에서의 순서 정보를 가지게 되어 자신 이전의 서브 태스크가 실행되지 않으면 자신이 실행 될 수 없게 된다. 또한 공유 자원을 사용하는 서브 태스크와 그렇지 않은

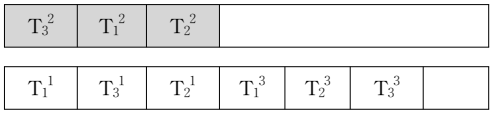
```
if(a new task T has arrived)
  insert T with partitioning into the ready queue
  if(an idle processor exists)
    if(a task with shared resource can be run)
      select the task earliest deadline from
        the ready queue
      assign the task to the idle processor
    else
      select the normal task earliest deadline
        from the ready queue
      assign the normal task to the idle processor

if(a task on processor P is finished)
  if(a task with shared resource can be run)
    select the task earliest deadline from
      the ready queue
    assign the task to the processor P
  else
    select the normal task earliest deadline from
      the ready queue
    assign the normal task to the processor P
```

(그림 3) EDFP 알고리즘

서브 태스크는 서로 다른 큐에 따로 삽입되어 EDF에 따라 정렬되게 된다. (그림 4)는 (그림 2)의 서브 태스크들이 서로 다른 큐에 삽입된 상태를 보여준다. 큐에 삽입이 끝난 이후에 현재 사용되지 않고 있는 프로세서가 있다면 우선 공유 자원을 사용하는 태스크의 레디 큐를 확인해 태스크를 프로세서에 할당 해 준다.

이때 여러 개의 공유 자원을 사용하는 태스크가 사용 될 수 있는 경우 사전에 주어진 별도의 마감 시간을 고려하여 가장 마감시간이 빠른 태스크를 지정해 준다. 공유 자원을 사용하는 태스크가 없다면 일반 태스크의 큐를 확인하여 마감 시간이 가장 빠른 태스크를 프로세서에 할당 해 준다. 또한 어느 프로세서에서 한 태스크가 종료 되었을 경우 이후에 처리될 태스크의 지정을 위해 위와 마찬가지로 공유 자원을 사용하는 태스크의 레디 큐를 확인해 주어진 별도의 마감 시간에 따라 할당 해 주고 그렇지 않은 경우 일반 태스크의 레디 큐를 확인해 역시 마감 시간이 가장 빠른 태스크를 프로세서에 지정해 준다.

GEDF의 경우는 공유 자원에 대한 어떠한 처리 방침 없이 태스크 하나에 대한 마감 시간만을 가지고 스케줄링을 하게 되므로 공유 자원이 복수의 태스크에서 발생하는 경우 임계 구역 사용의 중첩 현상이 발생하게 된다. 그에 따라 전반적인 태스크들이 차례대로 밀리게 되어 마감 시간까지 처리 되지 못하고 태스크 성공률이 떨어지게 된다. 그러나 EDFP는 GEDF와는 다르게 공유 자원을 사용하는 태스크와 일반 태스크를 구분하여 각각의 마감 시간을 주며 둘 사이에 우선순위를 두어 공유 자원을 가장 먼저 처리해 주는 방식을 통해 임계 구역 사용의 중첩 현상을 완화하여 좀 더 효율적인 스케줄링 성공률을 가지게 된다.



(그림 4) 서로 다른 큐의 서브 태스크 삽입

특별한 경우에 한해 태스크들이 여러 종류의 공유 자원을 계속적으로 이용하는 경우가 발생하여 급하게 처리해야 될 일반 태스크가 공유 자원에 의해 밀려나는 경우가 발생할 수 있다. 하지만 현실적인 적용에 있어 다수의 공유 자원이 사용된다 할지라도 그 양은 전체 태스크에서 매우 적은 부분일 것이다. 그러므로 일반 태스크가 공유 자원에 의해 밀려나는 경우가 발생하여도 그 수치는 매우 미미할 뿐만 아니라 그로 인한 스케줄링 성공률의 변화 또한 매우 적다고 할 수 있다.

4. 성능 평가

4.1 시뮬레이션 모델

성능 평가를 위해 시뮬레이션을 통하여 EDFP을 기존에 가장 많이 사용되고 있는 GEDF와 5가지 면에서 비교 및 평가를 하였다. 각각 시스템의 부하, 프로세서의 개수, 태스크 내의 공유 자원의 비율 그리고 태스크의 형태에 변화를 주어 연속적인 공유 자원이 등장하게 한 태스크에 대한 평가와 서로 다른 공유 자원이 사용 될 경우에서의 평가를 시행하였다.

시뮬레이션을 위한 태스크 모델로 태스크 T_i를 (C_i^1, C_i^2, C_i^3, D_i, A_i)로 구성하였다. 각각은 공유 자원 이전의 계산 시간, 공유 자원의 계산 시간, 공유 자원 이후의 계산 시간, 마감 시간, 그리고 도착 시간을 의미한다. 각각의 계산 시간은 일정한 크기 한도 내에서 랜덤하게 주어지며 이들의 합이 한 태스크의 계산 시간이 된다. 도착 시간을 가리키는 A_i는 각 태스크의 주기에 따라 일정한 시간 간격을 가지게 된다. 각각 나누어진 태스크의 마감 시간으로 공유 자원 이후의 마감 시간 $D_i^3 = D_i$, 공유 자원의 마감 시간 $D_i^2 = D_i^3 - C_i^3$, 그리고 공유 자원 이전의 마감 시간 $D_i^1 = D_i^2 - C_i^2$로 설정되었다.

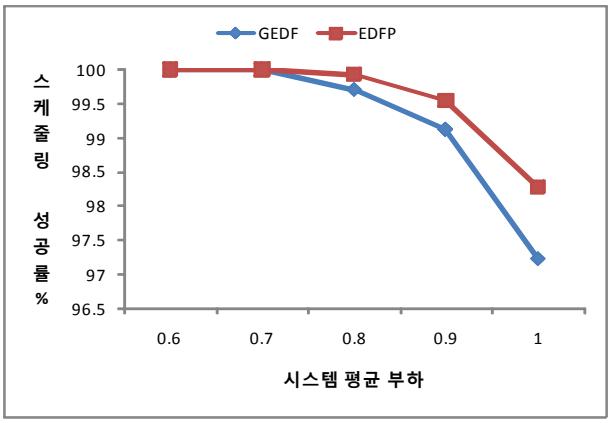
공유 자원 사용 시간은 한 태스크의 크기 대비 일정한 비율로 지정하며 태스크의 크기는 주기의 길이에 따른 시스템 평균 부하에 따라 결정되게 이루어져있다. 시스템의 평균 부하는 $\sum_{i=1}^{N} \frac{C_i}{P_i} / M$ 이다. M은 프로세서의 개수, N은 태스크의 개수, C_i 및 P_i 는 태스크 T_i 의 계산 시간 및 주기를 의미한다.

성능 평가는 스케줄링에 따른 태스크의 성공률로 비교하였다. 여기서의 스케줄링 성공률이란 실시한 모든 태스크 인스턴스의 개수 대비 성공적으로 스케줄링이 이루어진 태스크 인스턴스의 개수의 비율을 의미한다.

4.2 시스템의 부하에 따른 평가

(그림 5)는 시스템 부하의 변화에 따른 두 스케줄링의 성공률을 비교하여 보여주고 있다. 시스템의 부하는 0.6 부터 1.0 까지의 변화로 적용되었다. EDFP의 경우가 GEDF의 경우 보다 더 우수한 성능을 보여주었으며 그 성능의 차이는 시스템 부하가 증가함에 따라 더욱 큰 성능 차이를 나타내었다. 시뮬레이션에 적용된 파라미터로는 M = 3, N = 6, 공유 자원의 사용시간 비율은 10% 이다. 각 시뮬레이션의 시행에 있어 GEDF와 EDFP 두 개의 스케줄링 알고리즘에는 동일한 태스크 집합을 적용하였다.

시스템의 부하가 낮은 구간에서는 GEDF와 EDFP 양쪽 다 높은 성공률을 보여주며 차이를 나타내지 않고 있지만 로드가 점점 높아질수록 그 성능 차이가 점점 크게 나타나는 추세를 보여주고 있다. 이는 시스템의 로드가 높으면 높아 질수 록 공유 자원들의 임계 구역 사용이 잦아지고 그에 따라 GEDF는 경우는 태스크의 밀림 현상이 늘어나지만 EDFP의 경우는 공유 자원을 고려, 우선적으로 처리해 임계 구역을 폭 넓게 사용하게 하여 태스크의 성공률을 높임과 동시에 공유 자원들의 임계 구역 사용시도의 복수 접근을 완화시키기 때문이다.

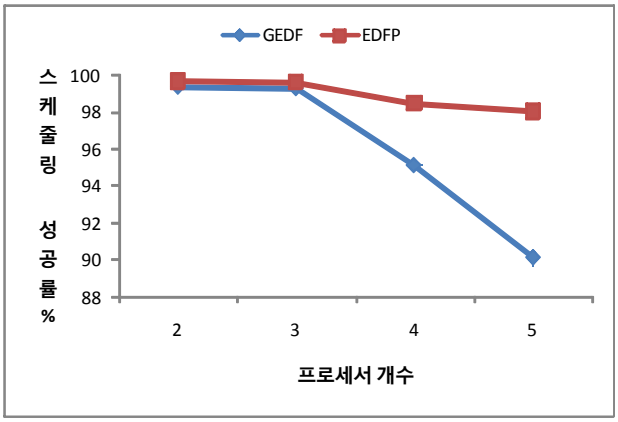


(그림 5) 시스템 평균 부하에 따른 평가

4.3 프로세서 개수에 따른 평가

(그림 6)은 프로세서의 개수에 따른 GEDF와 EDFP의 스케줄링 성공률을 보여주고 있다. 프로세서의 개수를 2개부터 5개까지의 변화로 적용하였다. 이 경우에는 시스템 부하에 따른 성능의 차이보다 더욱 큰 폭의 성능 차이를 보여주었다. 시뮬레이션에 적용한 파라미터로는 N = 6, 시스템 평균 부하 0.9, 공유 자원의 비율은 10% 이다. 위와 같이 각 시뮬레이션에 있어서 두 스케줄링 알고리즘에 동일한 태스크 집합이 적용되었다.

프로세서가 2개 일 경우에는 두 스케줄링 알고리즘의 차이가 미미하지만 프로세스의 개수가 늘어남에 따라 GEDF의 경우는 큰 폭의 하락세를 나타내는 반면 EDFP의 경우 성공률이 하락하기는 하지만 매우 미미한 것을 알 수 있다. 이는 프로세서의 개수가 늘어남에 따라 공유 자원의 처리



(그림 6) 프로세서 개수에 따른 평가

가 더욱 중요해 짐을 나타낸다. GEDF의 경우 프로세서가 늘어나게 되어도 공유 자원을 우선적으로 처리하는 절차가 없기 때문에 공유 자원을 위한 여유 시간을 사전에 만들어 놓지 못할 뿐만 아니라 공유 자원이 임계 구역 사용하게 되면 그 기간 동안 일반 태스크의 처리 역시 이 후의 공유 자원에 대한 대비 없이 이루어지기 때문에 자연스럽게 스케줄링 성공률이 떨어지는 것이다. 반면 EDFP의 경우는 공유 자원에 우선권을 주기 때문에 공유 자원 이전의 일반 태스크의 마감 시간을 통해 공유 자원 이전 일반 태스크의 처리를 하여 공유 자원 처리를 위한 여유 시간을 길게 만들 수 있게 하여 GEDF 보다 빠르게 공유 자원의 처리를 시작 할 수 있게 된다. 또한 공유 자원이 임계 구역을 사용하더라도 이후의 공유 자원에 대비로 그에 관한 마감 시간을 가지고 일반 태스크를 처리해 주기 때문에 GEDF보다 높은 스케줄링 성공률을 보여주게 된다.

각 태스크들이 여러 개의 임계 구역을 실행하는 경우와 여러 개의 공유 자원들이 사용되는 경우에 대한 시뮬레이션 결과도 (그림 5) 및 (그림 6)과 유사하게 나타났다. 또한 태스크에 대한 공유 자원 비율에 변화를 주고 시뮬레이션 한 결과 역시 기존의 결과와 비슷한 성능 차이를 나타내 주었다.

5. 결 론

다중 프로세서 환경에서 공유 자원의 사용을 반영한 스케줄링 알고리즘을 위하여 본 논문에서는 EDFP 알고리즘을 제시하고 여러 가지 스케줄링 알고리즘 중에 가장 널리 알려져 사용되고 있는 GEDF 알고리즘과 성능을 비교 하였다.

EDFP는 공유 자원을 가지는 태스크들의 성공률을 높이기 위하여 태스크를 공유 자원 사용 부분과 비 공유 자원 사용 부분으로 나누어 각각의 마감 시간을 가지고 그에 따라 처리하게 하였으며 공유 자원 사용 부분에는 일반 태스크 보다 우선순위를 주어 처리하게 하였다. 또한 공유 자원들에 의한 크리티컬 섹션 발생 시에도 이후의 공유 자원을 고려한 일반 태스크 처리 방식을 이용하여 전반적인 성공

률을 높이도록 하였다. 시뮬레이션을 통한 성능 평가 결과 EDFP는 GEDF에 비해 우수한 성능을 보였으며 시스템 부하가 늘어나거나 프로세서의 개수가 늘어남에 따라 더욱 큰 성능 차이를 나타내었다.

참 고 문 헌

[1] J. Carpenter, S. Funk, P. Holman, A. Srinivasan, J. Anderson, and S. Baruah, "A categorization of real-time multiprocessor scheduling problems and algorithms," Handbook on Scheduling Algorithms, Methods, and Models, pp.30.1-30.19, 2004.

[2] Giorgio C. Buttazzo, 'Hard real-time compuing systems', 2nd Ed., Springer, 2005.

[3] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment," Journal of the ACM, Vol.20, No.1, pp.46-61, 1973.

[4] C. L. Liu and J. W. Layland, 'Real-Time Systems', Prentice Hall, 2000.

[5] J. Hong, X. Tan, D. Towsley, "A performance analysis of minimum laxity and earliest deadline scheduling in a real-time system," IEEE Transactions on Computers, Vol.38, No.12, pp.1736-1744, Dec. 1989.

[6] Joseph Y. Leung, "A New Algorithm for Scheduling Periodic Real-Time Tasks," Algorithmica, Vol.4, pp.209-219, 1989.

[7] S. Cho, S. Lee, S. Ahn and K. Lin, "Efficient Real-Time Scheduling Algorithms for Multiprocessor Systems," IEICE Trans Commun., Vol. E85-B, No.12, pp.2859-2867, 2002.

[8] 조규억, 김용석, "다중프로세서 시스템을 위한 여유시간 기반의 온라인 실시간 스케줄링 알고리즘", 정보처리학회논문지 A, 제16-A권 제6호, Dec. 2009.

[9] L. Sha, R. Rajkumar, and J. P. Lehoczky, "Priority inheritance protocols: An approach to real-time synchronization," IEEE Transactions on Computer, Vol.39, No.9, Sep. 1990.

이 상 태

e-mail : finangels@hotmail.com

2009년 강원대학교 컴퓨터정보통신공학부(학사)

2009년~현 재 강원대학교 컴퓨터정보통신공학부 석사과정

관심분야 : 실시간 시스템, 운영체제

김 용 석

e-mail : yskim@kangwon.ac.kr

1984년 서울대학교 해양학과(학사)

1986년 KAIST 전기및전자공학과(공학석사)

1989년 KAIST 전기및전자공학과(공학박사)

1990년~1995년 한국생산기술연구원 및 전자부품연구소 선임연구원

1995년~현 재 강원대학교 컴퓨터학부 교수

관심분야 : 운영체제, 실시간 시스템, 임베디드 시스템