다중처리기 상의 실시간 스케줄링 알고리즘의 우월 관계 및 성능

(Dominance and Performance of Real-time Scheduling Algorithms on Multiprocessors)

박 민 규 † 한 상 철 †† 김 희 헌 ††† 조 성 제 †††† 조 유 근 †††††
(Minkyu Park) (Sangchul Han) (HeeHeon Kim) (Seongje Cho) (Yookun Cho)

요 약 하드웨어 기술이 발전하고 실시간 시스템들의 작업 부하가 커지면서 다중처리기를 실시간 시스템에 사용하는 것이 요구되고 있지만, 단일처리기와는 달리 다중처리기 실시간 스케줄링 문제는 대부분 효율적인 해결 방안이 알려져 있지 않다. 따라서 단일처리기 스케줄링 알고리즘을 다중처리기에 그대로 적용하는 연구와 단일처리기 스케줄링 알고리즘을 변형한 다중처리기 스케줄링 알고리즘에 관한 연구가 활발히 이루어지고 있다. 대표적인 알고리즘으로는 EDF(Earliest Deadline First), LLF(Least Laxity First), EDF-US[m/(2m-1)], EDZL(Earliest Deadline Zero Laxity) 알고리즘 등이 있으며, 이들 간의 비교 연구가 필요하다. 본 논문에서는 스케줄 가능성 측면에서 이 알고리즘들 사이의 우월(dominance) 관계를 밝혔다. EDF, LLF, EDF-US[m/(2m-1)] 간에는 우월 관계가 없으나, EDZL은 EDF보다 우월함을 증명하였다. 또한 모의실험을 통하여 EDZL은 선점을 적게 유발하고 처리기 이용률이 높음을 보였다.

키워드 : 우월 관계, 다중처리기, 실시간 스케줄링 알고리즘, 여유시간, 마감시간

Abstract Multiprocessor architecture becomes increasingly common on real-time systems as computer hardware technology rapidly progresses and the workload of real-time systems increases. However, efficient solutions for many real-time multiprocessor scheduling problems are not known. Hence many researchers apply uniprocessor scheduling algorithms to multiprocessor scheduling or devise new algorithms based on these algorithms. Such algorithms are EDF (Earliest Deadline First), LLF (Least Laxity First), EDF-US[m/(2m-1)], and EDZL (Earliest Deadline Zero Laxity), and comparative studies on them are necessary. In this paper, we show the dominance relation of these algorithms with respect to schedulability, and we prove EDZL strictly dominates EDF. The simulation results show that EDZL has high processor utilization and it causes a small number of preemptions.

Key words : dominance, multiprocessor, real-time scheduling algorithm, laxity, deadline

1. 서 론

하드웨어 기술이 발전하고 실시간 태스크들의 작업부하가 커짐에 따라, 다중처리기를 실시간 시스템에 사용하는 것이 점차 일반화되고 있다. 최근에 다중처리기 시스템에서 주기 태스크 스케줄링에 대한 연구가 많이 진행되고 있으며, 그 중 단일처리기 스케줄링 알고리즘을 다중처리기에 그대로 적용하는 연구와 단일처리기 스케줄링 알고리즘을 변형한 다중처리기 스케줄링 알고리즘에 관한 연구가 활발히 이루어지고 있다.

EDF(Earliest Deadline First) 알고리즘은 대표적인 단일처리기 스케줄링 알고리즘으로서 단일처리기 상에서 최적(optimal)의 알고리즘이다[1,2]. EDF는 문맥교환과 선점 횟수가 작업의 개수에 한정되고 알고리즘이 단순하여 효율적인 구현이 가능하다. 그러나 EDF는 다중처리기 상에서는 최적이 아니며[2,3] 이용률이 높은 태스크와 낮은 태스크들이 혼합되어 있는 태스크 집합을

· 본 연구는 두뇌한국21 사업과 서울대학교 컴퓨터연구소의 지원으로 수행되었습니다.

† 정 회 원 : 서울대학교 컴퓨터공학과
mkpark@ssrnet.snu.ac.kr
†† 학생회원 : 서울대학교 전기. 컴퓨터공학부
schan@ssrnet.snu.ac.kr
††† 비 회 원 : 서울대학교 전기. 컴퓨터공학부
hhkim@ssrnet.snu.ac.kr
†††† 정 회 원 : 단국대학교 정보컴퓨터학부 교수
sjcho@dku.edu
††††† 종신회원 : 서울대학교 전기. 컴퓨터공학부 교수
cho@cse.snu.ac.kr

논문접수 : 2004년 4월 20일
심사완료 : 2005년 4월 14일

잘 스케줄 하지 못하여 처리기 이용률(processor utilization)이 비교적 낮다[4].

LLF(Least Laxity First) 알고리즘도 최적의 단일처리기 스케줄링 알고리즘이지만[5], 역시 다중처리기 상에서는 최적이 아니다[2,3]. LLF는 EDF보다 처리기 이용률이 높은 것으로 알려져 있으나 LLF의 선점 횟수는 예측 불가능하여 매우 커질 수 있다[6].

EDF-US[m/(2m-1)] 알고리즘은 EDF 기반 다중처리기 스케줄링 알고리즘이다. EDF-US[m/(2m-1)] 알고리즘은 태스크의 이용률이 높은 태스크에게 최고 우선순위를 부여하고 이용률이 낮은 태스크는 EDF에 따라 우선순위를 부여함으로써 EDF의 단점을 보완하였다[7].

EDZL(Earliest Deadline Zero Laxity) 알고리즘은 EDF 기반 다중처리기 스케줄링 알고리즘이다. EDZL은 여유시간(laxity)이 0인 작업이 발생하면 그 작업에게 가장 높은 우선순위를 부여하고, 여유시간이 양수인 작업들은 EDF에 따라 우선순위를 부여한다. Cho 등은 비주기 작업 스케줄링에서 EDZL의 문맥교환이 적고 스케줄 성공률이 높음을 보였다[8].

본 논문은 주기 태스크 집합에 대해서, EDF와 LLF, EDF-US[m/(2m-1)], EDZL 알고리즘 간의 우월(dominance) 관계를 밝혔다. LLF와 EDF-US[m/(2m-1)] 알고리즘은 EDF 알고리즘보다 우월하지 않지만 EDZL 알고리즘은 EDF 알고리즘보다 우월하다. 즉, EDZL 알고리즘은 EDF 알고리즘이 스케줄 할 수 있는 모든 태스크 집합을 스케줄 할 수 있다. 또한 모의실험을 통하여 스케줄 가능한 태스크 집합의 비율, 스케줄 보장 이용률(schedulable utilization bound), 선점 횟수 측면에서 위의 스케줄링 알고리즘들을 비교하였다. 실험 결과, EDZL 알고리즘은 스케줄 가능한 태스크 집합과 스케줄 보장 이용률 측면에서 LLF 알고리즘에 필적하였으며, 선점 횟수는 EDF 알고리즘과 비슷하였다.

본 논문의 구성은 다음과 같다. 2장에서는 태스크 모델과 비교 대상 스케줄링 알고리즘들의 특성을 기술한다. 3장에서는 EDZL 알고리즘을 설명하고 알고리즘간의 우월 관계를 정의하고 알고리즘 간의 우월 관계를 보인다. 4장에서는 모의실험을 통한 성능 평가 결과를 제시하고, 5장에서 결론을 내리고 향후 연구 과제를 설명한다.

2. 관련 연구

2.1 시스템 모델

본 논문은 m개의 동일한 처리기(identical processor) 상에서 주기 태스크 집합을 대상으로 하는 선점 가능(preemptive) 경성 실시간(hard real-time) 스케줄링을 다룬다. 태스크 집합 $\tau = \{\tau_1, \tau_2, ..., \tau_n\}$은 n개의 주기 태스크로 구성되며($n > m$), 각 태스크 τ_i는 (R_i, C_i, D_i, P_i)로 표현되는데, 이는 각각 태스크의 도착 시간(release time), 수행시간(worst-case execution time), 상대적 마감시간(relative deadline), 주기(period)를 의미한다. 각 태스크는 P_i의 정수 배 시간, $R_i + (k-1) \cdot P_i$, (k = 1, 2, ...) 마다 하나의 작업(job)을 생성하며, 이 작업은 마감시간, $R_i + (k-1) \cdot P_i + D_i$까지 완료되어야 한다. 각 태스크의 도착 시간이 고정되어 있고 모든 태스크의 도착 시간이 일정할 때($R_1 = R_2 = \cdots = R_n = 0$) 동기적(synchronous) 태스크 집합이라고 한다. 또한 각 태스크 마감시간이 주기와 일치하면($D_i = P_i$) 묵시적 마감시간(implicit deadline)을 가진다고 한다. 태스크 집합이 동기적이고 묵시적 마감시간을 가진 경우 τ_i 는 (C_i, D_i)로 표현한다. 각 태스크 τ_i의 이용률(utilization)은 $u_i = C_i/D_i$로 주어지며, 태스크 집합 τ의 전체 이용률은 $U(\tau) = \sum_{i=1}^{n} u_i$ 이다.

작업 J_i는 (c_i, d_i)로 표시되며, 이는 각각 그 작업의 남아있는 수행 시간(remaining execution time)과 마감시간까지 남은 시간(remaining time to deadline)을 의미한다. 작업이 처음 도착할 때 c_i는 그 작업이 속한 태스크의 수행시간으로 주어지고, d_i는 상대적 마감시간으로 주어진다. 작업 J_i의 여유시간은 $l_i = d_i - c_i$로 정의되며, 여유시간은 작업의 긴급한 정도를 나타낸다. 여유시간이 음수이면 작업의 마감시간을 충족할 수 없음을 의미하고, 여유시간이 0이면 즉시 그 작업을 수행해야 마감시간을 충족할 수 있다.

태스크 집합의 어떤 스케줄이 각 태스크의 모든 작업의 마감시간을 충족할 때 그 스케줄을 유효(valid)하다고 한다. 태스크 집합이 m개의 처리기 상에서 유효 스케줄을 가질 때 그 태스크 집합은 m개 처리기 상에서 실행 가능(feasible)하다고 한다. 어떤 스케줄링 알고리즘이 m개의 처리기 상에서 실행 가능한 모든 태스크 집합에 대해 유효한 스케줄을 생성해 낼 수 있으면, 그 스케줄링 알고리즘은 m개 처리기 상에서 최적(optimal)이라고 한다. 스케줄링 알고리즘 Q가 주어진 태스크 집합 τ에 대해 m개의 처리기 상에서 유효한 스케줄을 생성해 낼 수 있으면, 태스크 집합 τ는 스케줄링 알고리즘 Q에 의해 m 개 처리기 상에서 스케줄 가능(schedulable)하다고 한다. 스케줄링 알고리즘 Q가 전체 이용률이 어떤 실수 이하인 모든 주기 태스크 집합에 대해 유효 스케줄을 생성할 수 있으면, 그 실수를 스케줄링 알고리즘 Q의 스케줄 보장 이용률(schedulable utilization) 또는 스케줄 보장 한계(schedulable utilization bound)라고 한다.

본 논문에서는 태스크 집합이 동기적이고, 모든 태스

크들이 묵시적 마감시간을 가지는 경우를 대상으로 한다. 또한, 각 작업은 독립적이라고 가정한다. 즉, 각 작업은 같은 태스크에 속하거나 다른 태스크에 속한 작업과 데이타 의존성(data dependency), 선행제약(precedence constraints) 등을 가지지 않는다.

2.2 다중처리기 상의 스케줄링 알고리즘

현재 시간에 준비된 작업들에 대해 마감시간이 빠른 순서로 높은 우선순위를 부여하는 EDF 알고리즘은 단일처리기에서는 최적이나 다중처리기에서는 최적이 아니며, 스케줄 보장 이용률이 낮다. 예를 들어, m개의 처리기를 갖는 시스템에서 주기가 1이고 수행시간이 2ε인 태스크가 m개, 주기가 $(1+\varepsilon)$ 이고 수행시간이 1인 태스크가 1개 있는 태스크 집합이 있다고 하자. EDF에 따라 스케줄하면 주기가 $(1+\varepsilon)$ 인 태스크는 마감시간을 충족할 수 없다. ε가 0에 수렴하면 이때의 이용률 U은 1에 수렴한다. 즉, m개의 처리기가 존재하더라도 다중처리기 상에서 EDF의 스케줄 보장 이용률은 1이다[4]. 그럼에도 불구하고, EDF가 다중처리기에서 계속 연구되는 이유는 선점과 처리기 간 이주 횟수가 작업의 개수에 한정되어 매우 효율적인 구현을 할 수 있기 때문이다[7,9].

위의 예와 같이 EDF는 이용률이 큰 태스크와 이용률이 작은 태스크가 혼합되어 있는 태스크 집합에 대해 스케줄 실패할 확률이 높다. EDF 기반 스케줄링 알고리즘인 EDF-US[m/(2m-1)] 알고리즘은 태스크의 이용률에 따라 우선순위 부여 방식을 다르게 함으로써 이러한 EDF의 단점을 보완하였다. EDF-US[m/(2m-1)]은 이용률이 $u_i > m/(2m-1)$인 모든 태스크 τ_i의 모든 작업에게 가장 높은 우선순위를 동일하게 부여하고, 이용률이 $u_i \le m/(2m-1)$인 모든 태스크 τ_j의 모든 작업에게는 EDF 방식으로 우선순위를 부여한다. EDF-US[m/(2m-1)]은 전체 이용률이 $m^2/(2m-1)$ 이하인 임의의 주기 태스크 집합을 스케줄 할 수 있다[7].

LLF는 현재 시간에 준비된 작업들 중 여유시간이 작은 작업에게 더 높은 우선순위를 부여한다. LLF 스케줄링에서는 시간이 흐름에 따라 우선순위가 계속 바뀌며, 우선순위가 같은 작업들로 인해 선점 횟수가 매우 많아지므로 선점 횟수의 예측이 불가능하다[6].

3. EDZL 알고리즘과 스케줄 가능성 비교

본 장에서는 EDZL 알고리즘을 기술하고, 스케줄 가능성 측면에서 EDF, LLF, EDF-US[m/(2m-1)], EDZL 알고리즘을 비교한다.

3.1 EDZL 알고리즘

EDZL 알고리즘은 여유시간이 0보다 큰 작업들은 EDF에 따라 스케줄 한다. 여유시간이 0인 작업이 발생하면(이 작업은 당장 수행하지 않으면 마감시간을 충족할 수 없다) 그 작업에게 가장 높은 우선순위를 부여한다. 예를 들어, 태스크 집합 {(1,2), (1,2), (5,6)}을 2개의 처리기 상에서 스케줄 할 때 EDF 스케줄과 EDZL 스케줄을 살펴보자. 그림 2의 (a)와 (b)는 각각 EDF 알고리즘과 EDZL 알고리즘에 따른 스케줄을 보여준다. EDF 스케줄과 EDZL 스케줄은 τ_3의 여유시간이 0이 되는 시간 2에서 달라진다. EDF 알고리즘은 마감시간이 가장 빠른 τ_1과 τ_2에게 가장 높은 우선순위를 부여하기 때문에 τ_3은 마감시간을 충족할 수 없다. 이와는 달리, EDZL 알고리즘은 여유시간이 0인 τ_3에게 가장

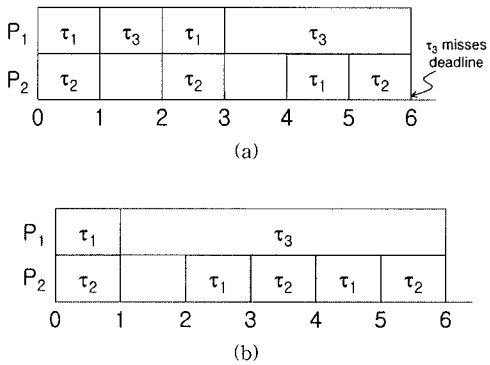

그림 2 태스크 집합 {(1,2),(1,2),(5,6)}의 (a) EDF 스케줄 (b) EDZL 스케줄

EDZL 알고리즘은 다음 규칙에 의해 각 작업에게 우선순위를 부여하고, 우선순위가 가장 높은 m개의 작업에게 처리기를 할당한다. 우선순위가 같은 작업들의 실행 순서는 임의로 결정한다.

i) 여유시간이 0이면, 가장 높은 우선순위를 부여한다.

ii) 여유시간이 0보다 크면, EDF에 따라 우선순위를 부여한다.

iii) 여유시간이 0보다 작으면, 시스템 실패(system failure) 신호를 발생시킨다.

그림 1 EDZL 알고리즘

높은 우선순위를 부여하기 때문에 모든 태스크가 마감시간을 충족할 수 있다.

Cho 등은 EDZL이 준최적(suboptimal)임을 증명하고, 모의실험을 통하여 비주기 작업 스케줄링에서 스케줄 성공률과 문맥교환 횟수를 비교하였다[8]. 그러나 주기 태스크 스케줄링 알고리즘인 EDF-US[m/(2m-1)]과 성능을 비교하지 않았으며, 스케줄 가능한 태스크 집합의 측면과 스케줄 보장 이용률의 측면에서 EDZL이 EDF보다 우수함을 보이지 못하였다.

3.2 스케줄 가능성 비교

스케줄링 알고리즘의 성능을 비교할 때 스케줄 보장 이용률이 하나의 척도가 되지만, 다중처리기에서 스케줄 보장 이용률이 정확히 알려진 알고리즘은 거의 없다. 뿐만 아니라 스케줄 보장 이용률을 정확히 알아도 한 알고리즘이 다른 알고리즘보다 항상 우수하다고 말할 수 없다. 왜냐하면 스케줄 보장 이용률이 높은 알고리즘이 스케줄 할 수 없는 태스크 집합을 스케줄 보장 이용률이 낮은 알고리즘이 스케줄 할 수도 있기 때문이다.

정의 1. 알고리즘 Q_1에 의해 스케줄 가능한 임의의 태스크 집합 τ가 알고리즘 Q_2에 의해 항상 스케줄 가능하면, Q_2는 Q_1보다 우월(*dominate*)하다고 한다. 또한 알고리즘 Q_2가 Q_1보다 우월하고 Q_1에 의해 스케줄 할 수 없지만 Q_2에 의해 스케줄 가능한 태스크 집합이 존재하면 Q_2는 Q_1보다 강 우월(strictly dominate)하다고 한다.

우월 관계는 스케줄 가능성 측면에서 스케줄링 알고리즘의 성능을 비교하는 좋은 기준이다. 알고리즘 Q_2는 알고리즘 Q_1이 스케줄 할 수 있는 모든 태스크 집합을 스케줄 할 수 있으므로, 스케줄링 오버헤드(scheduling overhead)를 무시할 때, 스케줄 가능성 측면에서 더 우수하다고 말할 수 있다. 또한 알고리즘 Q_2의 스케줄 보장 이용률은 알고리즘 Q_1의 스케줄 보장 이용률보다 높거나 같음이 명백하다.

보조 정리 1. LLF는 EDF 및 EDF-US[m/(2m-1)], EDZL보다 우월하지 않다.

증명. 태스크 집합 τ = {(2,4), (2,4), (2,4), (4,8)}은 2개의 처리기 상에서 EDF, EDF-US[m/(2m-1)], EDZL에 의해 스케줄 가능하지만, LLF에 의해 스케줄 가능하지 않다. 그림 3의 (a)와 (b)는 각각 τ에 대한 LLF 스케줄과 EDF 스케줄이다. EDF 스케줄에서 모든 태스크는 마감시간을 충족한다. 하지만, LLF 스케줄에서 시간 7에 τ_1, τ_3, τ_4가 모두 여유시간이 0이므로 이 중 하나의 태스크는 처리기를 할당받지 못해 마감시간을 충족하지 못한다. 그림 3(a)의 LLF 스케줄에서는 우선순위가 동일한 작업들이 처리기를 공유하지 않았지만, 처리기를 공유하더라도 τ는 스케줄 가능하지 않다.

m = 2일 때 EDF-US[m/(2m-1)]은 이용률이 2/3보다 큰 태스크에게 가장 높은 우선순위를 부여하고, 이용률이 2/3보다 작거나 같은 태스크들은 EDF로 우선순위를 부여한다. $u_1 = u_2 = u_3 = u_4 = 1/2$ 이므로 EDF-US[m/(2m-1)] 스케줄은 EDF 스케줄과 동일하다. 또한 그림 3 (b)에서 여유시간이 0인 작업들(시간 2, 3에서의 τ_3과 시간 6, 7에서의 τ_3, τ_4)은 마감시간이 가장 빠른 작업들이기 때문에 EDZL 스케줄은 EDF 스케줄과 동일하다. □

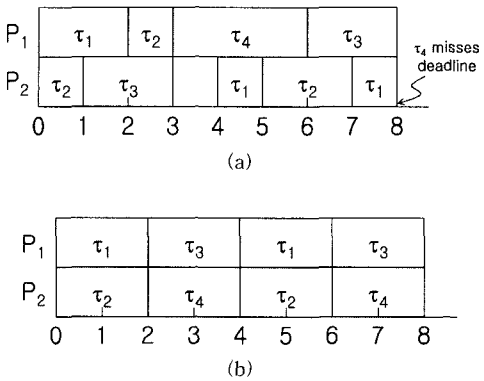

그림 3 태스크 집합 {(2,4),(2,4),(2,4),(4,8)}의 (a) LLF 스케줄 (b) EDF, EDF[m/(2m-1)], EDZL 스케줄

Kalyanasundaram 등은 비주기 태스크 스케줄링에서 LLF가 EDF보다 우월하지 않음을 보였다[10]. 보조 정리 1의 예는 주기 태스크이면서 $U(\tau) \le m$ 인 경우에도 LLF가 EDF보다 우월하지 않음을 보인다.

보조 정리 2. EDF는 LLF 및 EDF-US[m/(2m-1)], EDZL보다 우월하지 않다.

증명. 태스크 집합 τ = {(1,2), (1,2), (4,5)}는 2개의 처리기 상에서 LLF, EDF-US[m/(2m-1)], EDZL에 의해 스케줄 가능하지만 EDF에 의해서는 스케줄 가능하지 않다. 그림 4(a)의 시간 2에서 τ_3은 여유시간이 0이지만 τ_1과 τ_2의 마감시간이 더 빠르므로 τ_1과 τ_2에게 처리기가 할당되어 τ_3은 마감시간을 충족하지 못한다. □

보조 정리 3. EDF-US[m/(2m-1)]는 EDF 및 LLF, EDZL보다 우월하지 않다.

증명. 태스크 집합 τ = {(1,3), (3,4), (3,4)}는 2개의 처리기 상에서 EDF, LLF, EDZL에 의해 스케줄 가능하지만, EDF-US[m/(2m-1)]에 의해 스케줄 가능하지 않다. 그림 5(a)에서 $u_2 = u_3 = 3/4 > 2/3$ 이므로 τ_2와 τ_3은 항상 가장 높은 우선순위를 부여 받는다. 따라서 τ_1은 처리기를 할당받지 못한다. □

보조 정리 4. EDZL은 LLF 및 EDF-US[m/(2m-1)]보다 우월하지 않다.

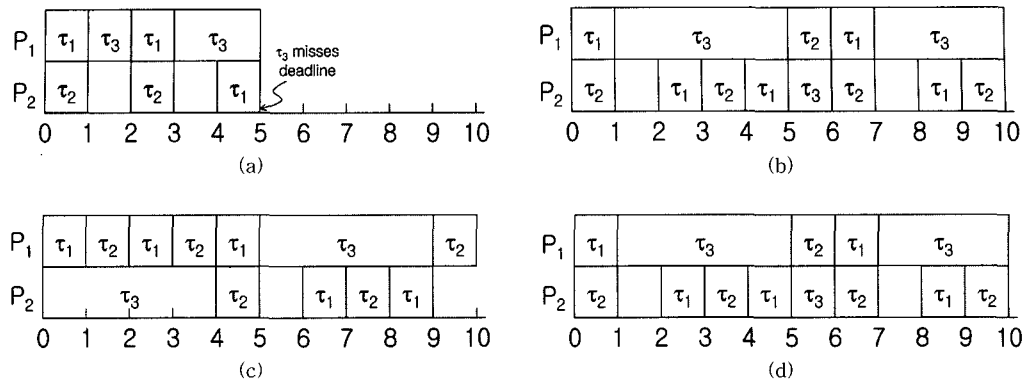

그림 4 태스크 집합 {(1,2),(1,2),(4,5)}의 (a) EDF 스케줄 (b) LLF 스케줄 (c) EDF-US[m/(2m-1)] 스케줄 (d) EDZL 스케줄

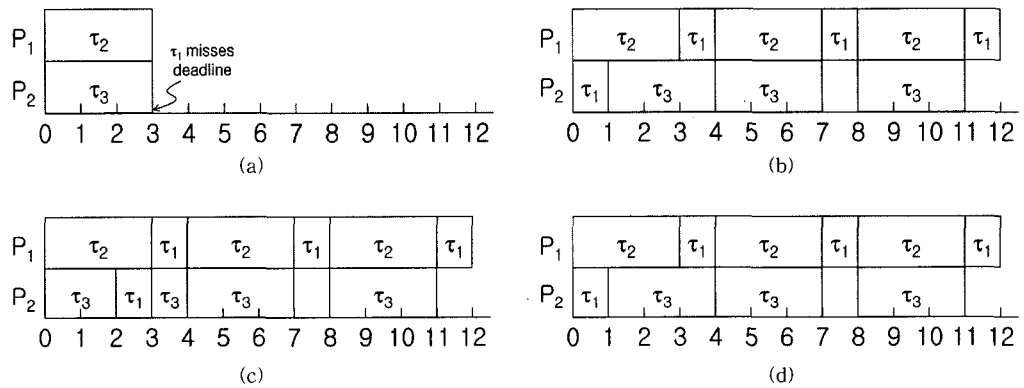

그림 5 태스크 집합 {(1,3), (3,4), (3,4)}의 (a) EDF-US[m/(2m-1)] 스케줄 (b) EDF 스케줄 (c) LLF 스케줄 (d) EDZL 스케줄

증명. 태스크 집합 τ = {(1,2), (1,2), (1,2), (3,5), (8,10)}은 3개의 처리기 상에서 LLF, EDF-US[m/(2m-1)]에 의해 스케줄 가능하지만, EDZL에 의해 스케줄 가능하지 않다. 그림 6(a)의 시간 9에 τ_2, τ_3, τ_4, τ_5가 모두 여유시간이 0이므로 이 중 하나의 태스크는 처리기를 할당받지 못해 마감시간을 충족하지 못한다. □

태스크 집합 $\tau = \{\tau_1, \tau_2, \ldots, \tau_n\}$을 m개의 처리기 상에서 스케줄 할 때 시간 t에 수행 가능한 p개의 작업($p \leq n$) 중 알고리즘 Q가 부여한 우선순위 순서대로 나열한 작업의 집합을 $\sigma_Q(t) = \{J_1, J_2, \ldots, J_k \mid k = \min(p,m)\}$라고 하자(i < j이면 J_i의 우선순위가 J_j의 우선순위보다 높다). $\sigma_Q(t)$는 알고리즘 Q에 의해 시간 t에 처리기가 할당될 작업의 집합을 나타낸다.

정리 1. 태스크 집합 τ가 EDF에 의해 스케줄 가능하면, $\sigma_{EDF}(t) = \sigma_{EDZL}(t)$이다.

증명. 시간 t에 p개의 작업 J_1, ..., J_p가 수행 가능하고 마감시간이 빠른 순서라고 하자. (J_1의 마감시간이

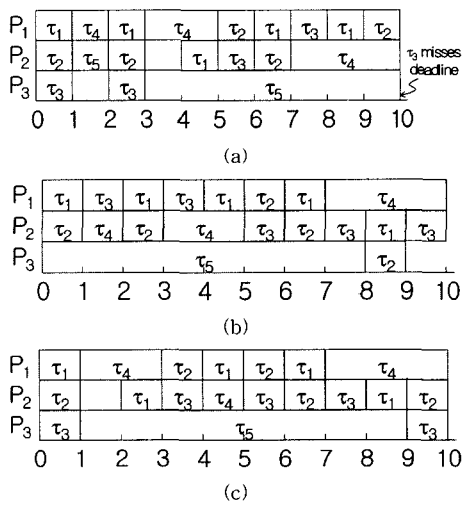

그림 6 태스크 집합 {(1,2), (1,2), (1,2), (3,5), (8,10)}의 (a) EDZL 스케줄 (b) EDF-US[m/(2m-1)] 스케줄 (c) LLF 스케줄

가장 빠르고 J_p의 마감시간이 가장 늦다.) EDF는 마감시간이 가장 빠른 작업에게 처리기를 할당하므로 $\sigma_{EDF}(t) = \{J_1, \ldots, J_k\}$이다. $\sigma_{EDF}(t)$에 속한 작업 중에 여유시간이 0보다 큰 작업을 $J'_1, J'_2, \ldots, J'_s$ ($0 \le s \le k$)라고 하고 마감시간이 빠른 순서라고 하자. 그러면, $\sigma_{EDF}(t) - \{J'_1, J'_2, \ldots, J'_s\}$는 $\sigma_{EDF}(t)$에 속한 작업 중에 여유시간이 0인 작업의 집합이며, $|\sigma_{EDF}(t) - \{J'_1, J'_2, \ldots, J'_s\}| = k-s$ 이다.

EDZL은 여유시간이 0인 작업에게 가장 높은 우선순위를 부여하므로 $\sigma_{EDF}(t) - \{J'_1, J'_2, \ldots, J'_s\}$에 속한 $(k-s)$개의 작업들에게 가장 높은 우선순위를 부여한다. 그리고 여유시간이 0보다 큰 작업들은 EDF에 따라 우선순위를 부여하므로 $J'_1, J'_2, \ldots, J'_s, J_{k+1}, J_{k+2}, \ldots, J_p$ 중에서 마감시간이 가장 빠른 s개의 작업에게 처리기를 할당한다. (τ가 EDF에 의해 스케줄 가능하므로 $J_{k+1}, J_{k+2}, \ldots, J_p$ 중에는 여유시간이 0인 작업이 존재할 수 없다.) 따라서, $\sigma_{EDZL}(t) = (\sigma_{EDF}(t) - \{J'_1, J'_2, \ldots, J'_s\}) \cup \{J'_1, J'_2, \ldots, J'_s\} = \sigma_{EDF}(t)$. □

정리 2. EDZL은 EDF보다 우월하다.

증명. 정리 1에 의해 EDF에 의해 스케줄 가능한 태스크 집합은 EDF 스케줄과 EDZL 스케줄이 동일하므로, 그 태스크 집합은 EDZL에 의해 스케줄 가능하다. □

보조정리 1~4에서 보인 바와 같이, EDF에 의해 스케줄 가능하지만 LLF 또는 EDF-US[m/(2m-1)]에 의해 스케줄 가능하지 않은 태스크 집합이 존재하므로, LLF와 EDF-US[m/(2m-1)]은 EDF보다 우월하지 않다. 또한, LLF와 EDF-US[m/(2m-1)], EDZL도 상호 우월 관계가 없다. 그러나 보조정리 2와 정리 2에 의해 EDZL은 EDF보다 강 우월하므로, 스케줄링 오버헤드를 무시할 때, 스케줄 가능성 측면에서 EDF보다 우수하다.

4. 모의실험

본 장에서는 모의실험을 통해 EDF, LLF, EDF-US[m/(2m-1)], EDZL 알고리즘의 성능을 비교 및 평가한다. 실험 I 에서는 무작위로 생성한 태스크 집합을 대상으로 실험하였고 실험 II에서는 일정 범위 안에서 생성 가능한 모든 태스크 집합을 대상으로 실험하였다.

4.1 실험 I

본 실험에서는 무작위로 생성한 태스크 집합에 대하여 성능을 평가하였다. 태스크 집합은 5개의 군(群)으로 나누어 생성하였으며, 각 군은 이용률이 $(u, u+1]$인 1600개의 태스크 집합으로 구성하였다(u = 1, 2, ..., 5). 각 태스크의 주기는 [10, 300]의 범위에서 평균이 50, 표준편차가 25인 정규분포(normal distribution)에 따라 생성하고, 수행시간은 [1, 40]의 범위에서 균일분포(uniform distribution)에 따라 생성하였다. 마감시간은 주기와 같고, 모든 태스크의 시작시간은 같다. 단, 시간과 각 태스크의 주기 및 수행시간은 정수이다.

그림 7은 처리기 수에 따른 전체 태스크 집합에 대한 스케줄 가능한 태스크 집합의 비율이다. 실험 범위의 태스크 집합에 대해, EDZL은 LLF와 대등한 성공률을 보여준다. 전체 8000개의 태스크 집합 중 EDZL로 스케줄 가능한 태스크 집합의 수와 LLF로 스케줄 가능한 태스크 집합의 수의 차이는 4개 이하였다. 반면에 EDF-US[m/(2m-1)]은 EDF보다는 우수하지만 EDZL보다 성공률이 낮다. 처리기가 4개일 때 EDF와 비교하여 EDZL과 LLF는 약 28%, EDF-US[m/(2m-1)]은 약 8% 더 많은 태스크 집합을 스케줄 할 수 있다.

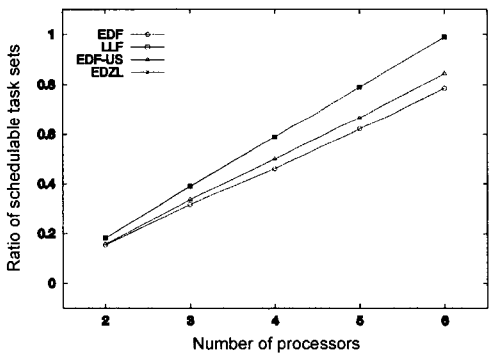

그림 7 스케줄 가능한 태스크 집합의 비율

그림 8은 처리기 수에 따른 스케줄 보장 이용률이다. LLF와 EDZL의 스케줄 보장 이용률은 이론적으로 알려지지 않았지만 본 실험 결과 EDF와 EDF-US[m/(2m-1)]보다 높음을 알 수 있다. EDZL의 스케줄 보장 이용률은 LLF와 대등하며, 두 알고리즘 사이의 우열관계는 명확하지 않다. EDF-US[m/(2m-1)]의 경우 실험 범위의 태스크 집합에 대한 스케줄 보장 이용률과 이론적으로 알려진 값과는 다소 차이가 있다. 처리기가 6개일 때 EDF-US[m/(2m-1)]의 이론적 스케줄 보장 이용률은 3.27이며, 실험 범위의 태스크 집합에서는 3.85이다. 그 이유는 이론적인 스케줄 보장 이용률은 최악의 경우를 고려했을 때의 값인데 실험 대상 태스크 집합에는 최악의 경우가 포함되지 않았을 가능성이 높기 때문이다.

그림 9는 EDF, LLF, EDF-US[m/(2m-1)], EDZL 모두에 의해 스케줄 가능한 태스크 집합에 대해 처리기가 3개 일 때 태스크 개수에 따른 평균 선점 횟수이다.

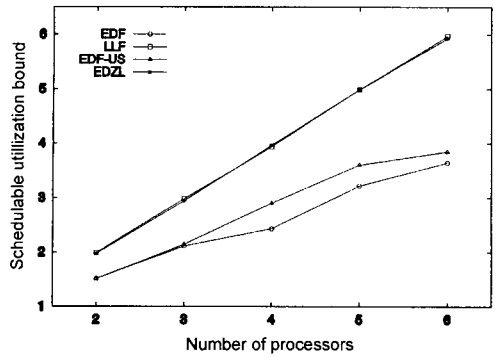

그림 8 스케줄 보장 이용률

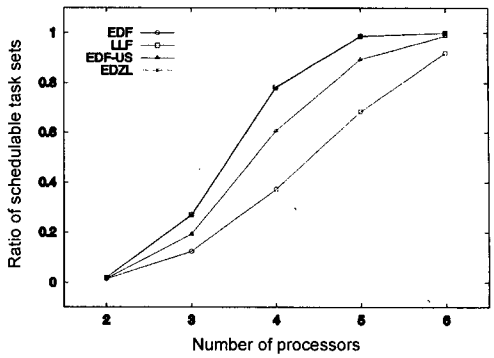

그림 10 스케줄 가능한 태스크 집합의 비율

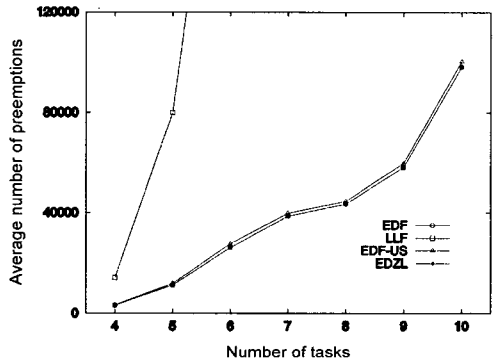

그림 9 평균 선점횟수 (처리기 3개일 때)

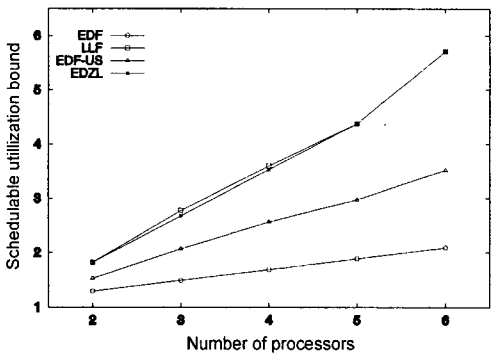

그림 11 스케줄 보장 이용률

LLF의 경우, 여유시간의 변화에 따라 우선순위가 지속적으로 변경되므로 많은 선점이 발생한다. 실험 결과, 태스크 수가 8개일 때 LLF의 평균 선점 횟수는 EDF, EDF-US, EDZL의 평균 선점 횟수의 약 10배이다. EDF-US[m/(2m-1)]의 경우, 평균 선점 횟수가 EDF보다 다소 높다. 그 이유는 마감시간이 늦지만 이용률이 높은 작업이 이용률은 낮지만 마감시간이 빠른 작업을 선점하는 경우가 발생하기 때문이다. EDF와 EDZL의 평균 선점횟수는 일치한다. 그 이유는 정리 1에 의해 EDF로 스케줄 가능한 태스크 집합에 대해서 EDF 스케줄과 EDZL 스케줄이 동일하므로 선점횟수가 동일하기 때문이다.

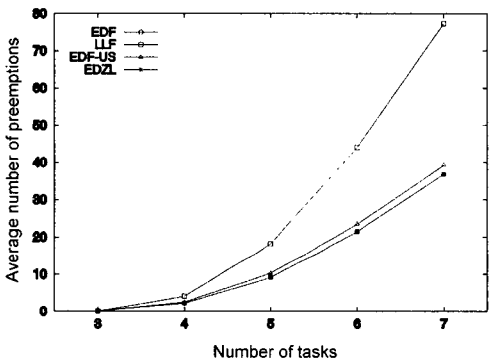

그림 12 평균 선점 횟수 (처리기 3개 일 때)

4.2 실험 II

본 실험의 목적은 주기와 태스크의 수를 제한했을 때 주어진 범위에서 최악의 경우를 포함한 모든 경우에 대해 성능을 비교하는 것이다. 일반적으로 무작위로 생성한 태스크 집합을 대상으로 하는 실험은 최악의 경우가 포함되지 않을 수 있으며, 또한 다중처리기 상에서는 최악의 경우를 알기 어렵다.

각 태스크 집합은 독립적인 3~7개의 태스크로 구성된다. 각 태스크의 주기는 2~10이며 수행시간은 1~(주기-1)이다. 이 범위에서 가능한 모든 태스크 집합을 생성하여 모의실험을 수행하였다. 두 개 이상의 작업이 같은 우선순위를 가질 때는 선점이 발생하지 않는 작업을 우선적으로 스케줄 하였다.

그림 10~12에서 보듯이 위 범위의 모든 태스크 집합에 대한 실험 결과도 실험 I 의 결과와 유사한 경향을 보인다. EDZL은 스케줄 할 수 있는 태스크 집합의 비율과 스케줄 보장 이용률의 측면에서 LLF와 대등하며

EDF와 EDF-US[m/(2m-1)]보다 우수한 성능을 보인다. 또한 선점 횟수의 측면에서 EDZL은 EDF 및 EDF-US[m/(2m-1)]과 비슷하다.

일정 범위 안에서 가능한 모든 태스크 집합에 대한 실험 결과가 무작위로 생성한 태스크 집합에 대한 실험 결과와 부합하므로 EDZL은 임의의 태스크 집합에 대해서도 유사한 성능을 보일 것으로 예상된다.

5. 결 론

본 논문은 다중처리기 상에서 스케줄 가능성 측면에서 알고리즘 간의 우월 관계, 스케줄 보장 이용률, 스케줄 가능한 태스크 집합의 비율, 선점 횟수의 관점에서 EDF, LLF, EDF-US[m/(2m-1)], EDZL 알고리즘을 비교하였다. 스케줄 가능성 측면에서 EDZL은 EDF보다 우월하였다. 즉, EDF로 스케줄 가능한 모든 태스크 집합은 EDZL로 스케줄 가능하다. 또한 무작위로 생성한 태스크 집합을 대상으로 한 모의실험 결과, EDZL의 스케줄 보장 이용률과 스케줄 가능한 태스크 집합의 비율은 EDF-US[m/(2m-1)]보다 높고 LLF와 대등하였으며, EDZL의 선점 횟수는 LLF보다 적고 EDF-US[m/(2m-1)]과 거의 동일하였다. 일정 범위에서 생성 가능한 모든 태스크 집합을 대상으로 한 모의실험에서도 유사한 경향을 보였다. 향후 연구과제로서 EDZL의 스케줄 가능성(feasibility) 검사를 위한 스케줄 보장 이용률, 예측가능성(predictability), 과부하에서의 특성, 자원 공유를 위한 프로토콜 등에 대한 연구가 필요하다.

참 고 문 헌

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

[2] Liu, J. W., Real-Time Systems, p.70, Prentice Hall, 2000.

[3] Dertouzos, M. L. and Mok, A. K., "Multiprocessor On-line Scheduling of Hard Real-Time Tasks," IEEE Trans. on Software Eng., Vol.15, No.12, pp. 1497-1506, 1989.

[4] Dhall, S. K. and Liu, C. L., "On a Real-Time Scheduling Problem," Operations Research. Vol.26, No.1, pp. 127-140, 1978.

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

[6] Carpenter, J., Funk, S., Holman, P., Srinivasan, A., Anderson, J., and Baruah, S., "A Categorization of Real-time Multiprocessor Scheduling Problems and Algorithms," In Handbook of Scheduling: Algorithms, Models, and Performance Analysis, Joseph Y-T Leung (ed). Chapman Hall/ CRC Press. 2004.

[7] Srinivasan, A. and Baruah, S., "Deadline-based Scheduling of Periodic Task Systems on Multiprocessors," Information Processing Letters, Vol. 84, No.2, pp. 93-98, 2002.

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

[9] Mok, A., "Task Management Techniques for Enforcing ED Scheduling on a Periodic Task Set," Proceedings of the 5th IEEE Workshop on Real-Time Software and Operating Systems, Washington, DC, pp. 42-46. May 1988.

[10] Kalyanasundaram, B., Pruhs, K. P., and Torng, E., "Errata: A New Algorithm for Scheduling Periodic, Real-Time Tasks," Algorithmica, Vol.28, pp. 269-270, 2000.

박 민 규

1991년 서울대학교 컴퓨터공학과 졸업(공학사). 1993년 서울대학교 컴퓨터공학과 졸업(공학석사). 1993년~현재 서울대학교 컴퓨터공학과 박사과정. 관심분야는 스케줄링 알고리즘, 실시간시스템, 분산시스템

한 상 철

1998년 연세대학교 컴퓨터과학과 졸업(공학사). 2000년 서울대학교 컴퓨터공학과 졸업(공학석사) 2000년~현재 서울대학교 컴퓨터공학과 박사과정. 관심분야는 실시간 시스템, 스케줄링 알고리즘

김 희 헌

1999년 고려대학교 컴퓨터과학과 졸업 2000년~현재 서울대학교 전기·컴퓨터공학부 석박사 통합과정. 관심분야는 다중처리기상에서의 실시간 스케줄링 분산시스템

조 성 제

1989년 서울대학교 컴퓨터공학과(학사). 1991년 서울대학교 대학원 컴퓨터공학과(공학석사). 1996년 서울대학교 대학원 컴퓨터공학과(공학박사). 1996년~1997년 서울대학교 컴퓨터신기술연구소 객원연구원. 2001년 미국 University of

Californmia, Irvine 객원 연구원. 1997년~현재 단국대학교 정보컴퓨터학부 컴퓨터과학전공 조교수. 관심분야는 컴퓨터 보안, 시스템 소프트웨어, 실시간 시스템, 분산 시스템 등

조 유 근
정보과학회논문지 : 시스템 및 이론
제 32 권 제 3 호 참조