다중프로세서에서의 혼합-임계 EDZL 알고리즘에서 임계-상승이 일어나기 전의 스케줄-가능성 분석

정남용 ○, 이진규‡

성균관대학교 컴퓨터공학과

goswlsqkqh@skku.edu, jinkyu.lee@skku.edu

Mixed-criticality EDZL scheduling analysis on multiprocessors considering scenarios before criticality transition

Namyong Jung○, Jinkyu Lee‡

Dept. of Computer Science and Engineering, Sungkyunkwan University

요 약

혼합-임계(Mixed-Criticality)는 실시간 시스템 상에서 최대수행시간의 정확한 측정에 대한 어려움으로 인해 발생하는 비효율 문제를 해결하기 위해 대두된 개념이다. 혼합-임계 EDZL(Earliest Deadline First until Zero laxity) 스케줄링 알고리즘은 단일-임계 EDZL 알고리즘의 뛰어난 효율에 영감을 받아 고안된 알고리즘이지만, 이에 대한 스케줄-가능성 연구는 아직 이루어지지 않은 상태이다. 본 논문에서는 혼합-임계 EDZL에서 임계가 올라가지 않았을 때의 스케줄-가능성을 연구했다. 그리고 이 연구를 더 개량한 연구까지 제시한 다음, 두 스케줄 가능성의 성능을 비교하였다.

1. 서 론

최근 드론, 무인자동차 등 실시간 시스템을 기반으로 한 기술들이 각광받고 있다. 이런 실시간 시스템에서는 작업들이 마감시간 제한을 어기지 않도록 스케줄링 하는 것이 중요하다. 실시간 스케줄링에서 작업의 최대수행시간을 아는 것은 필수적이지만 이를 정확하게 아는 것은 불가능하다. 이에 대해 시스템의 안정성을 중요시하는 인증 기관들은 이론들을 동원해 보다 부정적으로 최대수행시간의 상한선을 계산한다. 이 때는 실행시간의 상한값이 매우 크게 정해지게 된다. 반면에 시스템 설계자는 최대수행시간을 보다 긍정적으로 계산하고 싶어한다. 그래서 이들은 경험에 의거한 실행시간 중 최대시간 등을 이용해 실행시간의 상한값을 보다 작게 정한다. 이론을 통해 구한 부정적인 실행시간의 상한선으로 스케줄링을 하게 되면 평균적인 실시간 시스템의 이용률이 저하될 수 밖에 없다. 반대로 경험에 의거한 실행시간의 상한선을 이용해 스케줄링을 하면 실행시간이 상한선을 넘겨서 시간 제약을 만족시키지 못하는 경우가 발생할 수 있다.

혼합-임계(Mixed Criticality)라는 개념은 이런 문제를 해결하기 위해 등장했다 [1]. 혼합-임계에서는 작업들에 중요도를 정해놓고 평상시에는 경험에 의거한 상한선을 이용해서 스케줄링을 하다가, 중요도가 높은 작업의 실제 수행시간이 경험적인 상한선을 넘어가는 경우가 발생한다면 중요도가 낮은 작업들의 마감시간 준수를 어기는 한이 있더라도 중요도가 높은 작업들은 이론적인 상한선으로 마감시간 제약을 지키게 하는 방법이다. 혼합-임계에서의 스케줄링 알고리즘에 대한 연구도 활발하게 이루어지고 있다 [2, 3].

단일-임계에서 가장 대표적인 스케줄링 알고리즘인 EDF(Earliest Deadline First)는 단일프로세서(Uniprocessor)상에서는 최적(Optimal)이라는 것이 증명되어 있지만 다중프로세서상에서는 효율이 낮다 [4]. 이를 보완하기 위해 EDF를 개조한 EDZL(Earliest Deadline until Zero Laxity) 스케줄링 알고리즘이 대두되었다 [5, 6]. 작업의 최소여유시간(Laxity)이란 그 작업이 실행되지 않은 상태를 유지하면서 시간 제약을 어기지 않는 최대한의 시간을 의미한다. EDZL 스케줄링 알고리즘은 마감시간이 빠를수록 높은 순위를 부여하되 최소여유시간이 0인 작업에 대해서 가장 높은 우선순위를 부여하게 된다. 최근 단일-임계일 때 다중프로세서상에서 EDZL의 효율성에 착안하여 이를 혼합-임계까지 확장한 알고리즘이 제시되었다 [7, 8]. 하지만 아직까지 혼합-임계 EDZL 스케줄링 알고리즘에 대한 스케줄-가능성 분석(Scheduling Analysis)에 대한 연구는 존재하지 않았다.

이 논문에서는 단일-임계에서의 EDZL 스케줄-가능성 분석을 응용해 시스템 임계가 올라가기 전 상황에 대한 혼합-임계 EDZL의 스케줄-가능성 분석을 할 것이다. 이후, 더 엄격하게 스케줄-가능성을 높이는 방법을 적용해서 이를 적용하지 않았을 때와 비교하여 얼마나 향상 되었는지를 비교함으로써 본 논문에서의 분석이 우수하다는 것을 보일 것이다.

2. 시스템 모델

이 논문에서 우리는 혼합-임계(Sporadic) 산발성 시

‡ 교신저자 (Corresponding author)

스템을 고려한다. 이 시스템은 m개의 동등한 프로세서들로 이루어져 있다. 이 시스템에서 스케줄링 될 혼합-임계 태스크 집합 τ은 임계의 가짓수가 두 개(LO, HI)이다. 각 태스크 $\tau_i \in \tau$는 다음과 같이 다섯 개의 변수로 이루어져 있다; $\tau_i = (T_i, D_i, \chi_i, C_i^{LO}, C_i^{HI})$. T_i는 태스크가 방출한 작업들 간의 최소한의 시간 간격이다. D_i는 상대적 마감시간이다. $D_i \le T_i$이다. χ_i는 태스크의 임계이다. $\chi_i \in \{LO, HI\}$이다. C_i^{LO}는 LO-실행시간이라고 지칭한다. C_i^{LO}은 양의 실수이다. $C_i^{LO} \le D_i$이다. C_i^{HI}는 HI-실행시간이다. C_i^{LO}은 양의 실수이다. $\chi_i = \mathrm{LO}$면 $C_i^{LO} = C_i^{HI}$이고 $\chi_i = \mathrm{HI}$이면 $C_i^{LO} \le C_i^{HI}$이며 $C_i^{HI} \le D_i$이다.

각 태스크 τ_i는 최소 T_i의 시간 간격마다 작업을 방출한다. τ_i의 k번째 작업을 j_i^k라고 하며 다음과 같은 다섯 개의 변수로 이루어져 있다; $j_i^k = (r_i^k, d_i^k, \chi_i, C_i^{LO}, C_i^{HI})$. 여기서 r_i^k는 j_i^k가 방출된 시간이다. d_i^k는 j_i^k의 절대적 마감시간, 다시 말해 $d_i^k = r_i^k + D_i$이다. $\chi_i, C_i^{LO}, C_i^{HI}$는 각각 작업 j_i^k의 임계, LO-실행시간, HI-실행시간으로 τ_i의 값과 같다.

혼합-임계 환경에서의 최소여유시간은 HI-실행시간을 기준으로 마감시간을 초과하지 않는 한에서 실행되지 않고 기다릴 수 있는 최대한의 시간이라고 정의된다 [8]. 즉, 어떤 시점 t가 있을 때, 임의의 작업 j_i^k가 HI-실행시간을 다 실행하기 위해 남은 시간이 c라면 최소여유시간은 $d_i^k - (t + c)$이다. 혼합-임계에서의 EDZL 스케줄링 알고리즘을 사용하면 마감시간이 빠를수록 높은 우선순위를 부여하되 최소여유시간이 0인 작업에 대해서는 가장 높은 우선순위를 부여하게 된다.

시스템은 초창기에 임계가 LO이며 어떤 $\chi_i = HI$인 작업 j_i^k이 LO-실행시간을 넘어도 실행을 끝내지 못하면 임계가 HI로 상승한다. 이 논문에서는 시스템의 임계가 LO일 때만 고려할 것이다. 이 논문에서는 어떤 알고리즘이 다음과 같은 조건을 만족하면 올바르다고 정의한다. 이 정의는 Pathan의 연구에서 제시한 임계 상승 전에 작업들이 만족해야 하는 조건과 부합한다 [2]. 알고리즘은 $\chi_i = LO$인 작업에 대해 절대적 마감시간 전에 실행을 끝낸다는 것을 보장하고 $\chi_i = HI$인 작업은 절대적 마감시간에서 $C_i^{HI} - C_i^{LO}$만큼 빠른 시점 전에 끝낸다는 것을 보장한다. 시스템의 임계가 HI일 때는 알고리즘이 $\chi_i = HI$인 작업에 대해 절대적 마감시간 전에 실행을 끝낸다는 것을 보장한다.

구간 $[x, y)$가 있을 때 이 구간에서 특별히 τ_i의 작업들의 높은 우선순위에 밀려 τ_k의 작업이 실행되지 못할 때 방해를 받는다고 표현하고 방해를 받는 시간의 최댓값을 $I_k^i(x, y)$라고 정의한다.

3. 시스템이 LO-임계일 때 스케줄-가능성 분석

이 절에서는 혼합-임계 EDZL 스케줄-가능성 분석에 필요한 기존의 연구를 먼저 소개하고 이를 혼합-임계에 맞게 수정한 다음, 방해하는 태스크의 임계를 고려하여 스케줄-가능성을 분석하는 방법을 보일 것이다.

3.1 기존의 연구

Cirinei는 단일-임계 환경에서 EDZL 스케줄링 알고리즘을 사용할 때 작업들이 최소여유시간이 0인 상태에 도달하지 않을 조건을 연구하였다 [9]. 어떤 단일-임계 제한적 마감시간(constrained-deadline) 태스크 τ_k가 있고 이 태스크의 상대적 마감시간을 D_k, 실행시간을 C_k라고 하자. 최소여유시간이 0인 상태에 도달하지 않을 조건은 다음과 같다.

$$\sum_{i \ne k} I_k^i(r_k^j, d_k^j) < m(D_k - C_k)$$

그리고 최소여유시간이 0인 상태에 도달하지 않는 태스크의 개수가 m개 이하라면 단일-임계에서 EDZL 스케줄링 알고리즘으로 스케줄-가능(schedulable)함을 보였다.

3.2 혼합-임계 EDZL 스케줄-가능성 분석

우리는 단일-임계 EDZL 스케줄-가능성 분석을 같이 혼합-임계 환경에 맞게 변형시켜서 다음의 정리로 나타내었다.

정리 1. 어떤 혼합-임계 태스크 τ_k가 존재하고 혼합-임계 EDZL을 사용할 때, 다음 조건을 만족하면 시스템의 임계가 LO로 유지되는 상황에서는 최소여유시간이 0이 상태에 도달하지 않는다:

$$\sum_{i \ne k} I_k^i(r_k^j, d_k^j) < m(D_k - C_k^{HI}) \cdots\cdots (1)$$

태스크 τ_k의 임계가 LO라면 $C_k^{HI} = C_k^{LO}$라고 했으므로 위의 부등식을 만족하면 마감시간까지 남은 시간이 남은 LO-실행시간보다 많다는 것을 보장한다. 반대로 태스크 τ_k의 임계가 HI라면 현재 시점에서 마감시간에서 $C_k^{HI} - C_k^{LO}$만큼 이른 시점까지의 시간이 남은 LO-실행시간보다 많다는 것을 보장한다. 따라서 혼합-임계에서 위

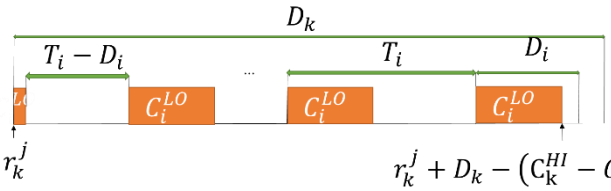

그림 1 모든 τ_i의 작업들이 방해량이 될 때

의 부등식을 만족하면 최소여유시간이 항상 0보다 크다. 그리고 위의 부등식을 만족하지 않는 태스크의 개수가 m개 이하이고 시스템의 임계가 LO로 유지될 때 혼합-임계 EDZL로 스케줄-가능하다. 하지만 혼합-임계 EDZL에서는 단일-임계에서와 달리 χ_i가 LO인지 HI인지에 따라 τ_i의 작업이 방해할 수 있는 구간이 달라지며 이를 고려하여 $I_k^i(r_k^j, d_k^j)$를 구해야 한다.

우선적으로 생각할 수 있는 경우는 구간 $[r_k^j, d_k^j)$의 끝에서 $C_k^{HI} - C_k^{LO}$만큼 이른 시점에서 τ_i의 LO-실행시간만큼 이른 시간에 τ_i의 작업이 방출되었고 이전 시점들에는 T_i 간격으로 τ_i의 작업들이 계속 방출된 상황이다.

만약 이 경우에 그림 1과 같이 구간 $[r_k^j, d_k^j)$에서 마지막에 방출된 τ_i의 작업의 마감시간이 $r_k^j + D_k$보다 빠르다고 가정해보자. 이 경우를 부등식으로 표현하면 다음과 같다.

$$r_k^j + \{D_k - (C_k^{HI} - C_k^{LO}) - C_i^{LO} + D_i\} < r_k^j + D_k$$

이 때, 구간 $[r_k^j, d_k^j)$에서 방출된 τ_i의 작업들이 모두 j_k^j보다 우선순위가 높으므로 이 경우에 $I_k^i(r_k^j, d_k^j)$가 최대이다. 그리고 이 때의 $I_k^i(r_k^j, d_k^j)$는 다음과 같다. 우선

$[r_k^j, r_k^j + D_k - (C_k^{HI} - C_k^{LO}) - C_i^{LO})$에서 방출되는 τ_i의 작업들의 개수를 N_k^i라고 하면

$$N_k^i = \left\lfloor \frac{D_k - (C_k^{HI} - C_k^{LO}) - C_i^{LO}}{T_i} \right\rfloor$$

와 같고, $I_k^i(r_i^j, d_i^j)$는 다음과 같은 식으로 구할 수 있다.

$$I_k^i(r_i^j, d_i^j) = C_i^{LO} + N_k^i * C_i^{LO} + \min(\max(D_k - (C_k^{HI} - C_k^{LO}) - C_i^{LO} - N_k^i * T_i - (T_i - D_i), 0), C_i^{LO})$$

하지만 그림 2와 같이 구간 $[r_k^j, d_k^j)$에서 마지막에 방출된 τ_i의 작업의 마감시간이 $r_k^j + D_k$와 같거나 더 나중인 경우를 생각해보자. 이 경우를 부등식으로 표현하면 다음과 같다.

$$r_k^j + \{D_k - (C_k^{HI} - C_k^{LO}) - C_i^{LO} + D_i\} \geq r_k^j + D_k$$

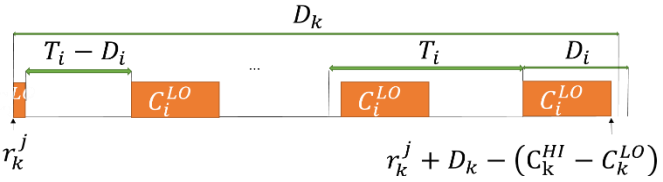

그림 2 τ_i의 마지막 작업이 방해를 하지 못할 때

이 때는 $r_k^j + D_k - (C_k^{HI} - C_k^{LO}) - C_i^{LO}$ 시점에 방출된 τ_i의 작업의 우선순위가 j_k^j보다 낮아져 $I_k^i(r_k^j, d_k^j)$가 최대가 될 수 없다. 따라서 τ_i의 작업의 우선순위가 더 높아질 때까지 방출시간을 당겨야 한다. 그러면 그림 3과 같이 j_k^j의 마감시간과 $[r_k^j, d_k^j)$에서 τ_i가 마지막으로 방출한 작업의 마감시간이 일치하게 된다. 이를 구하는 수식은 다음과 같다.

$$I_k^i(r_i^j, d_i^j) = C_i^{LO} + \left\lfloor \frac{D_k - D_i}{T_i} \right\rfloor * C_i^{LO} + \min\left(\max\left((D_k - D_i) - \left\lfloor \frac{D_k - D_i}{T_i} \right\rfloor * T_i - (T_i - D_i), 0\right), C_i^{LO}\right)$$

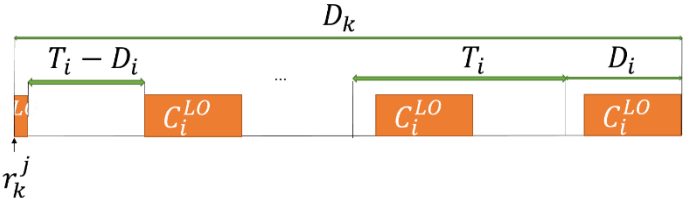

그림 3 τ_i의 마지막 작업의 방출 시간을 당겼을 때

Cirinei는 단일-임계 EDZL에 대해 방해량을 (상대적 마감시간-실행시간)으로 상한값을 정해서 더 엄격하게 스케줄-가능성을 분석하는 방법을 제시했다 [9]. 혼합-임계 EDZL에 대해서도 비슷한 방법을 적용할 수 있다. 혼합-임계에서는 최소여유시간이 HI-실행시간을 기준으로 정해지므로 상한값을 (상대적 마감시간-(HI-실행시간))으로 바꿔주면 된다.

정리 2. 혼합-임계 EDZL을 사용할 때, 임의의 태스크 $\tau_k \in \tau$는 다음의 부등식을 만족하면 시스템의 임계가 LO로 유지되는 상황에서는 최소여유시간이 0인 상태에 도달하지 않는다.

$$\sum_{i \neq k} \min(I_k^i(r_k^j, d_k^j), D_k - C_k^{HI}) < m(D_k - C_k^{HI}) \cdots\cdots\cdots (2)$$

4. 평가

우리는 3절에서 만든 스케줄-가능성 분석의 성능을 평가하기 위해 우선 프로세서의 개수가 2, 4개일 때 주기가 1000 이하의 자연수이고 다음의 부등식을 만족하는 임의의 태스크 집합을 각각 10만 개씩 생성했다.

$$\sum_{\tau_i \in \tau} \frac{C_i^{LO}}{T_i} \leq m, \sum_{\tau_i \in \tau \wedge \chi_i = HI} \frac{C_i^{HI}}{T_i} \leq m$$

그리고 3절의 스케줄 가능성 분석 (1), (2)를 통과하는 태스크 집합의 개수를 각각 그림 4에 그래프로 나타내었다. 프로세서의 개수가 늘어날수록 향상된 스케줄-가능성 분석 (2)가 기존 분석 (1)보다 더 우수한 성능을 보인다는 것을 알 수 있다.

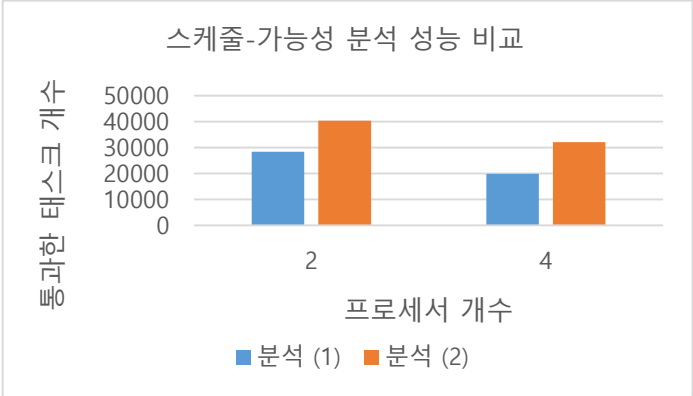

그림 4 스케줄-가능성 분석 성능 비교

5. 결론

우리는 혼합-임계 EDZL에서 시스템 임계가 올라가기 전 상황에 대한 스케줄-가능성 분석을 하였다. 우선, 혼합 단일-임계 EDZL 스케줄-가능성 분석을 혼합-임계 EDZL에 맞게 수정한 다음, 혼합-임계에 맞게 방해량을 구하는 방법들을 설명하였다. 이 연구는 차후 혼합-임계 EDZL 알고리즘에서 시스템 임계 상승 이후 상황까지 고려한 스케줄-가능성 분석의 기초가 될 것으로 기대한다.

Acknowledgement

이 논문은 2014년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(NRF-2014R1A1A1035827). 이 논문은 또한 미래창조과학부 및 정보통신기술진흥센터의 SW중심대학지원사업의 연구결과로 수행되었음(R2215-16-1005). 이 논문은 또한 미래창조과학부 및 정보통신기술진흥센터의 정보통신·방송 연구개발사업의 일환으로 수행하였음. [No. R0190-15-2071, (2세부) 클라우드 맵 기반의 자율이동 서비스 다양성 지원을 위한 개방형 PnP형 플랫폼 기술 개발].

참고문헌

[1] S. Vestel, "Preemptive Scheduling of Multi-criticality Systems with Varying Degrees of Execution Time Assurance", In Proc of RTSS, pages 239-243, 2007.

[2] R. M. Pathan, "Schedulability Analysis of Mixed-criticality Systems on Multiprocessors", In Proc of Euromicro Conference on Real-Time Systems, pages 309-320, 2012.

[3] H. Li, S. Baruah, "Global mixed-criticality scheduling on multiprocessors", In Proc of Euromicro Conference on Real-Time Systems, pages 166-175, 2012.

[4] C. Liu and J. Layland, "Scheduling Algorithms for MultiProgramming in a Hard-Real-Time Environment," J. ACM, vol. 20, no. 1, pp. 46-61, 1973.

[5] S. K. Lee, "On-line multiprocessor scheduling algorithms for real-time tasks," in IEEE Region 10's Ninth Annual International Conference, pp. 607-611, 1994.

[6] S. Cho, S.-K. Lee, S. Ahn, and K.-J. Lin, "Efficient real-time scheduling algorithms for multiprocessor systems," IEICE Trans. on Communications, vol. E85-B, no. 12, pp. 2859-2867, 2002.

[7] M. Park, S. Han, H. Kim, S. Cho, and Y. Cho, "Comparison of deadline-based scheduling algorithms for periodic real-time tasks on multiprocessor," IEICE Transaction on Information and Systems, vol. E88-D, pp. 658-661, 2005.

[8] 정남용, 이진규, "다중프로세서를 위한 혼합-임계 EDZL 알고리즘 개발", 한국정보과학회 2015년 동계학술발표회 논문집, 2015.12, 1579-1581.

[9] M. Cirinei, T. Baker, "EDZL Scheduling Analysis", In Proc of Euromicro Conference on Real-Time Systems, pages 9 -18, 2012.