다중프로세서를 위한 혼합-임계 EDZL 알고리즘 개발

정남용O, 이진규‡
성균관대학교 컴퓨터공학과
goswlsqkqh@skku.edu, jinkyu.lee@skku.edu

Development of Mixed-Criticality EDZL Algorithm for Multiprocessors

Namyong JungO, Jinkyu Lee‡
Dept. of Computer Science and Engineering, Sungkyunkwan University

요 약

최근에 실시간 시스템 연구에서는 실행시간의 정확한 측정의 어려움으로 인한 효율 감소를 해소하고자 혼합-임계(Mixed-Criticality)라는 개념이 새롭게 대두되었다. 하지만 다중프로세서 상에서의 혼합-임계 스케줄링 알고리즘에 대한 연구는 아직 초기 단계이다. EDZL(Earliest Deadline first until Zero Laxity) 스케줄링 알고리즘은 단일-임계(Single-Criticality) 환경에서의 다중프로세서 상에서 성능이 좋은 알고리즘으로 알려져 있지만, 이를 혼합-임계 환경에 적용한 연구는 아직까지 존재하지 않았다. 본 논문에서는 혼합-임계 환경에서의 최소여유시간(Laxity)를 새롭게 정의하여 혼합-임계 환경을 위한 EDZL 스케줄링 알고리즘을 개발하였다. 이를 EDF(Earliest Deadline First) 스케줄링 알고리즘과의 성능 비교를 통해, 본 연구에서 제안하는 EDZL 알고리즘이 혼합-임계 환경에서의 다중프로세서 상에서 우수한 성능을 보임을 확인하였다.

1. 서 론

실시간 시스템의 스케줄링은 주어진 작업들의 시간 제약을 만족시키기 위해 작업들의 우선순위를 정해주는 것이다. 특정 스케줄링 알고리즘 하에서 시간 제약이 만족됨을 보장하기 위해 최대수행시간(Worst-Case Execution Time)은 꼭 필요한 매개변수이지만, 최대수행시간을 정확하게 구하는 것은 매우 어려운 일이다. 따라서, 매우 엄격한 인증기관에서는 프로그램 코드 분석을 통해 최대수행시간의 상한선을 비관적으로 계산하여 매우 큰 상한값을 갖게 되며, 덜 엄격한 인증기관에서는 서로 다른 입력에 대해 실험을 반복해서 최대수행시간의 상한선을 구하게 되는데, 이는 이론적으로 보장된 상한선은 아니다.

따라서 이론적으로 보장된 상한선을 가지고 스케줄링을 하게 되면 상한선보다 훨씬 적은 평균 수행시간 때문에 실시간 시스템의 이용률이 저하되며, 반대로 경험적인 상한선을 가지고 스케줄링을 하게 되면 매우 적은 확률이지만 실제 수행시간이 상한선보다 커져서 실시간 시스템의 시간 제약을 만족시키지 못할 수 있다. 이러한 문제를 해결하기 위하여 2007년 Vestel은 최초로 혼합-임계(Mixed Criticality)라는 개념을 제안하였다[1]. 이는 각 작업마다 중요도를 매겨놓고 평소에는 경험적인 상한선을 이용하여 스케줄링을 하고, 중요도가 높은 작업의 실제 수행시간이 경험적인 상한선 보다 커지는 경우 중요도가 낮은 작업은 포기하고 중요도가 높은 작업만 시간 제약을 만족시키는 방법이다.

최근에 다중프로세서를 탑재한 실시간 시스템의 등장으로 인해, 혼합-임계 환경에서의 다중프로세서 상에서의 전역 선점형(Global Preemptive) 스케줄링에 관한 연구가 2012년부터 시작되었다[2, 3]. Pathan은 고정 우선순위(Fixed Priority) 알고리즘을 고려하였으며[2], Li와 Baruah는 EDF-VD와 fpEDF를 결합한 새로운 스케줄링 알고리즘을 제시하였다[3]. 하지만 단일프로세서에서의 혼합-임계 스케줄링에 대한 많은 연구가 진행된 사실과는 달리 다중프로세서 상에서의 혼합-임계 스케줄링에 대한 연구는 초기 단계이며, 몇몇 알고리즘에 대해서 한정적으로 연구가 되어 왔다.

실시간 시스템에서 가장 많이 연구되는 알고리즘 중에 하나인 EDF(Earliest Deadline First) 알고리즘은 단일프로세서 상에서는 최적이지만 다중프로세서 상에서는 비효율적이라는 것이 알려져 있다[4]. 이에 새롭게 등장한 스케줄링 알고리즘이 EDZL(Earliest Deadline first until Zero Laxity) 알고리즘이다[5][6]. 어떤 작업의 최소여유시간(Laxity)는 그 작업이 마감시간 초과를 내지 않기 위해 기다릴 수 있는 최대한의 시간을 의미한다. EDZL은 EDF와 동일하게 마감시간이 빠를수록 높은 우선순위를 부여하지만, 최소여유시간이 0인 작업은 가장 높은 우선순위를 부여하는 방식으로 동작한다. 단일-임계(Single-Criticality) 환경에서 EDF로 스케줄링이 가능하면 EDZL로도 항상 스케줄링이 가능하다는 성질이 알려져 있으며, EDZL은 EDF와는 달리 다중프로세서 상에서 효율적인 것으로 알려져 있다[7]. 하지만 혼합-임계 환경에서 EDZL이 아직 적용된 연구는 존재하지 않았다.

이 논문에서는 혼합-임계 환경에서 EDZL이

‡ 교신저자 (Corresponding author)

효율적인지 알아보는 것을 목표로 하며, 이를 위해 최소여유시간을 혼합-임계의 관점에서 재해석하여, 혼합-임계 환경에서의 EDZL을 개발 하는 것을 목표로 한다. 이후에, 임의의 태스크 집합들을 생성하여 기존에 있던 혼합-임계 환경에서의 EDF와의 시간 제약 만족 여부를 비교함으로써 혼합-임계 환경에서의 다중프로세서 상에서 제안하는 EDZL이 우수함을 입증할 것이다.

2. 시스템 모델

이 논문에서는 혼합-임계 산발성(Sporadic) 태스크 시스템을 고려한다. 우리는 임계의 가짓수가 총 2개(LO, HI)인 혼합-임계 태스크 집합 τ를 고려한다. 각 태스크 $\tau_i \in \tau$는 LO 또는 HI의 임계를 가진다. 임계가 LO인 태스크는 임계가 LO인 작업을 적어도 주기 이상의 시간 간격으로 생성하고 임계가 HI인 태스크는 임계가 HI인 작업을 적어도 주기 이상의 시간 간격으로 생성한다. 임계가 LO인 τ_i의 작업 j_i는 실행시간 C_i^{LO}를 가지고, 임계가 HI인 τ_i의 작업 j_i는 두 개의 실행시간 $< C_i^{LO}, C_i^{HI} >$ (단, $C_i^{LO} \le C_i^{HI}$)을 가진다. C_i^{LO}는 LO-임계에서의 최대수행시간, C_i^{HI}는 HI-임계에서의 최대수행시간이라고 지칭한다. 어떤 작업이 C_i^{LO}가 넘기 전에 실행 종료 신호를 보내면 LO-임계 행동을 보인다고 정의한다. 만약에 C_i^{LO}를 넘기고 C_i^{HI}전에 실행 종료 신호를 보내면 HI-임계 행동을 보인다고 정의한다.

혼합-임계 산발성 모델의 어떤 알고리즘이 다음의 2가지 조건을 만족하면 올바르다고 정의한다.

- 모든 작업들이 LO-임계 행동을 보이는 동안 알고리즘은 그 작업들이 마감시간 전까지 LO-임계에서의 최대수행시간으로 실행될 충분한 시간을 확보한다는 것을 보장한다.
- 만약 어떤 작업이 HI-임계 행동을 보인다면 알고리즘은 임계가 HI인 모든 작업들이 마감시간 전까지 HI-임계에서의 최대수행시간으로 실행될 충분한 시간을 확보한다는 것을 보장한다.

3. 단일-임계 환경에서의 기존 EDZL 알고리즘과 혼합-임계에서의 새로운 EDZL 알고리즘

3.1. 단일-임계 환경에서의 기존 EDZL 알고리즘

최소여유시간이란 작업이 마감시간 초과를 일으키지 않기 위해 실행되지 않고 대기할 수 있는 최대한의 시간을 의미한다. EDZL 알고리즘은 EDF와 비슷하게 마감시간이 더 빠를수록 더 높은 우선순위를 부여하지만, 최소여유시간이 0인 작업에 가장 높은 우선순위를 부여한다는 점에서 다르다.

3.2 혼합-임계에서 고려되어야 하는 점

혼합-임계에서는 실행 시간이 두 종류가 있다. 따라서 기존의 단일-임계 환경에서의 최소여유시간의 개념을 그대로 적용할 수는 없다. 만약에 LO-임계에서의 최대수행시간을 기준으로 실행되지 않고 대기할 수 있는 최대한의 시간을 혼합-임계에서의 최소여유시간이라고 가정해보자. 어떤 작업이 최소여유시간이 0인 상태에 있다가 다른 작업에 의해 HI-임계 행동을 보여야 하는 상황이 올 경우 HI-임계에서의 최대수행시간이 더 길다면 마감시간 초과가 일어나게 된다.

3.3 혼합-임계에서의 EDZL 알고리즘

3.2절에서와 같은 상황을 피하기 위해 우리는 혼합-임계 환경에서의 최소여유시간을 가장 높은 임계에서의 최대수행시간을 기준으로 마감시간 초과를 일으키지 않고 대기할 수 있는 최대한의 시간이라고 정의한다. 그리고 단일-임계에서의 EDZL과 같이, 마감시간이 빠를수록 더 높은 우선순위를 부여하되 혼합 임계에서의 최소수행시간이 0인 작업에 대해서 가장 높은 우선순위를 부여하게 된다. 시스템이 작동될 때 그림1의 알고리즘에 따라 작업들이 실행되게 된다. 첫 번째 줄에서는 임계 지표 β를 LO로 초기화시킨다. 이후 β가 LO인 동안 EDZL로 작업들의 우선순위를 정하며 만약 하나의 작업이라도 LO에서의 최대수행시간을 넘어가게 되면 5번째 줄에서 임계 지표가 HI로 바뀐 다음 11번째 줄에 있는 루프로 들어가게 된다. β가 HI인 동안에는 임계가 HI인 작업들에 대해서만 EDZL로 우선순위를 정하게 되며 하나의 작업이라도 HI-임계에서의 최대수행시간을 넘어가게 되면 14번째 줄에서 스케줄링이 실패했다고 반환한 뒤 끝나게 되고 그런 경우가 발생하지 않았다면 17번째 줄에서 스케줄링이 성공했다고 반환한 뒤 알고리즘을 종료하게 된다.

Algorithm 1 EDZL scheduling algorithm in MC

```
1:  Criticality indicator β is initialized to LO.
2:  while β =LO do
3:      The priorities of jobs are determined by EDZL policy.
4:      if Any job exceeds its LO-execution time then
5:          β becomes HI.
6:      end if
7:      if Any job misses its deadline then
8:          return False.
9:      end if
10: end while
11: while β =HI do
12:     The priorities of HI-criticality jobs are determined by EDZL policy.
13:     if Any HI-critality job exceeds its HI-execution time or misses its deadline then
14:         return False.
15:     end if
16: end while
17: return True.
```

그림 1. 혼합-임계에서의 EDZL 알고리즘

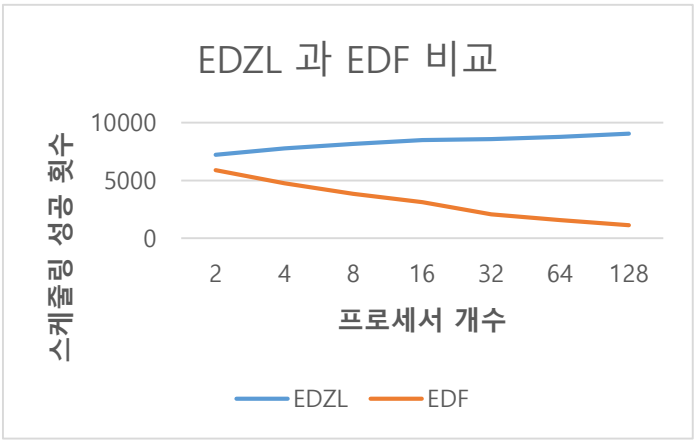

그림 2. EDZL과 EDF 성능 비교(제한된 마감시간 태스크)

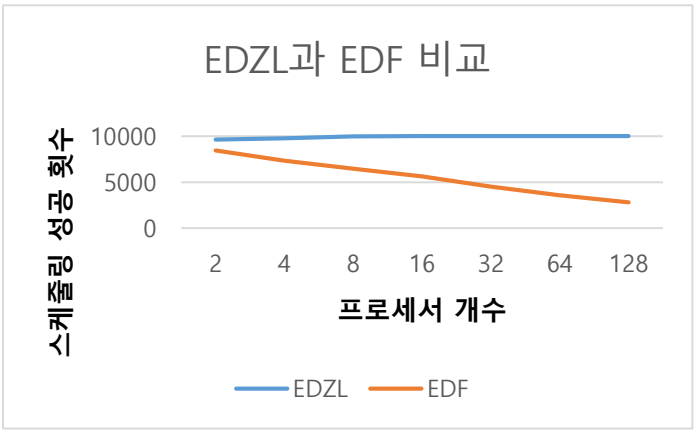

그림 3. EDZL과 EDF 성능 비교(함축적 마감시간 태스크)

4. 성능 평가 및 토의

우리는 혼합-임계에서 EDZL과 EDF의 성능을 비교하여 EDZL이 유용한지를 평가하였다. 임의의 제한된 마감시간(constrained-deadline)을 가지는 산발성 작업과 함축적 마감시간(implicit-deadline)을 가지는 임의의 작업 집합들을 각각 생성한 다음, 임의의 작업이 LO-임계에서의 최대수행시간을 채울 때마다 0.5%의 확률로 HI-임계 행동을 보이게 하는 모의 실험을 수행하였다. 프로세서의 개수는 2, 4, 8, 16, 32, 64, 128개로 하여 각각의 경우에 대해 10,000개의 작업 집합들로 실험해 보았다. 그림 2와 3에서 가로축은 프로세서의 개수, 세로축은 마감시간 초과를 하지 않고 올바르게 끝난 경우의 수를 나타낸다.

제한된 마감시간을 가지는 산발성 태스크들로 모의 실험한 결과, 그림 2에서 볼 수 있듯이 EDZL이 항상 우월한 성능을 보이며, 프로세서의 개수가 늘어날수록 보다 향상의 정도가 커짐을 보임을 알 수 있다. 함축적 마감시간을 가지는 산발성 작업들로 모의 실험한 결과도 마찬가지로 EDZL이 더 좋은 성능을 보였다. 이에 대한 결과는 그림3에 나타나 있다.

단일-임계 환경에서는 EDF에 비해 EDZL이 스케줄링 가능(Schedulability) 능력에서 우위를 보일 지라도 계속해서 남은 실행시간과 마감시간을 확인해야 하기 때문에 오버헤드가 클 수 밖에 없었다. 하지만 혼합-임계에서는 EDF 스케줄링 알고리즘을 사용하더라도 임계가 올라가는 상황이 올 것을 고려해서 지속적으로 남은 실행시간을 확인해야 한다. 따라서 혼합-임계에서는 EDZL와 EDF의 오버헤드 차이가 크지 않다.

5. 결론 및 향후 연구

혼합-임계에서 다시 정의한 최소여유시간이 합리적임을 알 수 있었으며, 혼합-임계에서도 EDZL이 EDF보다 더 우수한 성능을 보임을 알 수 있었다.

향후 혼합-임계에서 EDZL 스케줄링 분석을 통해 태스크 집합이 스케줄 가능한지 검증할 수 있는 방법을 개발할 예정이다.

Acknowledgements

본 연구는 미래창조과학부 및 정보통신기술진흥센터의 서울어코드활성화지원사업의 연구결과로 수행되었음 (IITP-2015-R0613-15-1062). 또한 본 연구는 2014년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(NRF-2014R1A1A1035827). 본 연구는 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”, Euromicro Conference on Real-Time Systems, pages 309-320, 2012.

[3] S.Baruah, H.Li, “Global mixed-criticality scheduling on multiprocessors”, 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.