분산 연성 실시간 시스템에서의 개선된 데드라인 할당

강경순°, 이금석
동국대학교 컴퓨터공학과

An Improved Deadline Assignment In Distributed Soft Real-Time System

Kyoung-soon Kang°, Keum-suk Lee
Dept. of Computer Engineering, Dongguk University

요 약

분산 연성 실시간 시스템에서의 태스크는 하나 이상의 노드에서 수행되는데, 하나 이상의 부태스크로 나누어져 할당되게 되며, 이때의 데드라인을 종단점 데드라인이라 한다. 종단점 데드라인을 노드에서 수행하는 부태스크의 가상 데드라인으로 할당하기 위한 정책은 태스크 클래스에 대한 공평성과 중단정책, 그리고 과부하 해결과 같은 시스템 자원의 효율적 사용을 위해 사용된다. 이러한 정책에서 가장 높은 성능을 갖는 것은 EQF 방법이나 전역 태스크와 지역 태스크 사이에서 실패 편향성을 갖는 단점이 있다.

본 논문에서는 최초 부태스크의 우선순위를 매우 높게 책정하고 뒤따르는 부태스크들이 가상 데드라인을 어기지 않은 경우 이전 부태스크의 가상 데드라인을 전체 데드라인에서 제외한다. 잔여 슬랙을 후위 부태스크들이 상속받지 않도록 하여 우선순위가 낮아지는 것을 방지하고, 가상 데드라인을 어긴 경우에만 EQF방법을 사용하였다. 동일한 도착율을 갖는 모의 실험에서 전역 태스크로만 구성한 경우 비슷한 실패율을 보였으나 전역 태스크와 지역 태스크가 존재하는 실험에서는 실패한 태스크의 수는 동일하지만 EQF방법보다 더 많은 전역 태스크를 생성하고 수행할 수 있었다.

1. 개 요

실시간 시스템은 연산의 정확성 측면에서 논리적 정확성과 함께 시간적 제약을 준수할 것을 요구하는 시스템으로 데드라인을 어겼을 때의 시스템의 상태에 따라 경성 실시간(hard real-time)과 연성 실시간(soft real-time)으로 구분된다. 분산 연성 실시간 시스템에서의 태스크는 하나 이상의 부태스크로 나누어져 독립적인 하나 이상의 노드에서 수행된다. 태스크는 데드라인을 갖게 되는데 이를 종단점 데드라인이라 한다. 종단점 데드라인을 노드에서 수행하는 부태스크의 가상 데드라인으로 할당하는 것은 태스크 클래스에 대한 공평성과 중단 정책, 그리고 과부하 상태를 해결하기 위해 사용되므로 시스템 자원의 효율적 사용을 위하여 매우 중요하다. 비록 태스크 정보와 노드 상태에 대한 정보를 알고 있다고 가정하여도 모든 태스크의 데드라인을 만족시키는 전역적인 최적 할당 방법은 NP 문제이므로 다양한 할당방법이 존재하게 된다.

본 연구는 가상 데드라인 할당 방법 중 가장 높은 성능을 갖는 EQF(Equal Flexibility)방법에서 전역 태스크와 지역 태스크간의 실패율과 편향성을 극복하기 위한 연구로써 노드에서의 기본적인 스케줄링 기법은 EDF(Earliest Deadline First)를 사용하였다. 최초 부태스크의 우선순위를 높여 먼저 수행될 수 있도록 하며, 가상 데드라인을 어기지 않은 부태스크에 대해서는 EQF와 같이 잔여 슬랙을 나머지 부태스크들이 상속받지 않도록 하였다. 이전 부태스크의 가상 데드라인을 전체 데드라인에서 제외시킴으로써 더 낮은 우선순위로 되는 것을 방지하였다. 데드라인을 어긴 태스크에 대해서는 부태스크의 도착시간을 적용하여 가상 데드라인을 할당해 줌으로써 전역 태스크의 실제적인 데드라인을 유지한 상태로 더 높은 우선순위를 갖도록 하여 지역 태스크와 전역 태스크간의 수행의 공평성과 더 높은 수행능력을 갖도록 한다.

2장에서는 관련 연구를 알아보며, 3장에서는 시스템 모델을, 4장에서는 성능 평가를 다루고, 5장에서 결론 및 향후 연구과제에 대하여 논한다.

2. 관련 연구

실시간 시스템에서의 스케줄링은 크게 정적인 방법과 동적인 방법으로 나눌 수 있으며 대표적인 방법이 RM(Rate Monotonic)과 EDF이다[4,5]. RM 방법은 각 태스크에 고정된 우선순위를 부여하는 방법으로 태스크의 수행시간이 짧을수록 높은 태스크 우선순위를 갖게 되며, EDF는 데드라인이 이르면 이를수록 높은 우선순위를 갖게 된다.

분산 연성 실시간 시스템에서의 태스크를 전역적 태스크와 지역적 태스크로 나누어 생각할 때, 전역적 태스크는 각 지역 스케줄러에게

낮은 우선순위로 인식되는 현상이 발생되어 우선순위 역전현상(Priority Inversion)이 발생할 수 있다. 하나의 전역 태스크가 여러 개(m)의 부태스크들로 이루어졌고 선행되는 태스크가 데드라인을 어겼을 경우 뒤따르는 태스크들이 연속적으로 데드라인을 어기는 현상이 발생될 수 있다[3].

종단점 데드라인에 대하여 각 부태스크가 갖는 임의의 데드라인을 가상 데드라인이라 하며, 공평성과 태스크의 중단 정책, 그리고 과부하 상태를 해결하기 위해 사용된다. 기존의 가상 데드라인 할당 정책으로는 UD(Ultimate Deadline), ED(Effective Deadline), EQF, EQS(Equal Slack)방법이 있다.

본 연구에서 사용된 태스크 T의 속성은 [표 1]과 같다.

속성	의미
$ar(T)$	태스크 T의 도착시간
$sl(T)$	태스크 T의 슬랙
$dl(T)$	태스크 T의 데드라인
$ex(T)$	태스크 T의 실제 실행시간
$pex(T)$	태스크 T의 예상 실행시간
T_i	전역 태스크 T의 i번째 부태스크

[표 1] 태스크 T의 속성

태스크의 데드라인을 정의하여 보면 다음과 같다.

$$dl(T) = ar(T) + ex(T) + sl(T)$$

UD는 전역 태스크의 부태스크에 대한 정보를 갖고 있지 않을 때 사용되며, 순차적으로 수행되는 전위 부태스크에 전역 태스크의 모든 슬랙 시간을 할당해 주는 방법으로 $dl(T_i) = dl(T)$로 표현된다. ED는 종단점 데드라인에서 후위 부태스크들이 수행될 실행시간을 뺀 나머지 시간을 부태스크의 데드라인으로 할당하는 방식으로 $dl(T_i) = dl(T) - \sum_{j=i+1}^{m} pex(T_j)$로 표현된다. UD나 ED는 전위 부태스크가 시스템의 다른 태스크에 비하여 낮은 우선순위를 갖게 되어 뒤따르는 후위 부태스크들이 실패할 확률이 높아지게 된다.

한편 EQS는 전역 태스크의 잔여 슬랙 시간을 후위 부태스크들에 동일하게 할당하는 방법으로 다음과 같으며 []은 잔여슬랙이다.

$$dl(T_i) = ar(T_i) + pex(T_i) + \left[dl(T) - ar(T_i) - \sum_{j=i}^{m} pex(T_j) \right] / (m - i + 1)$$

EQF는 전역 태스크의 잔여 슬랙 시간을 후위 부태스크들의 실행시간 비율에 따라 할당해주는 방법으로 아래와 같이 나타낼 수 있다.

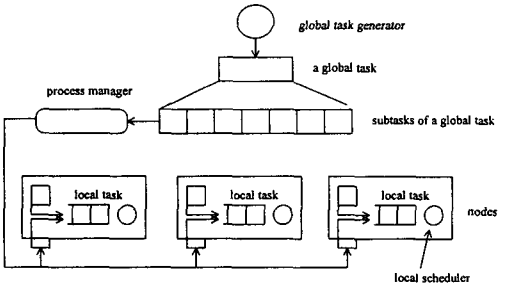
$$dl(T_i) = ar(T_i) + pex(T_i) + \left[dl(T) - ar(T_i) - \sum_{j=i}^{m} pex(T_j) \right] \cdot \left(pex(T_i) / \sum_{j=i}^{m} pex(T_j) \right)$$

UD, ED, EQS 순서로 성능이 향상되며, EQS와 EQF는 상당히 유사하다고 알려져 있다[3].

3. 시스템 모델

분산 실시간 시스템은 여러 개의 노드로 구성되며, 각 노드는 시스템 컴포넌트를 나타낸다. 각각의 노드는 하나 이상의 자원을 관리하며, 각각의 노드에서 독립적인 지역 스케줄러가 EDF방식을 통해 비선점형으로 태스크를 수행하게 된다.

태스크는 지역 태스크와 전역 태스크로 구분하였다. 지역 태스크는 단지 하나의 노드만을 방문하고 단 한번만 스케줄러에게 인식된다. 이에 반하여 전역 태스크는 하나 이상의 순차적인 부태스크들로 이루어져 있으며, 최종 부태스크가 종료되는 시간이 전역 태스크의 데드라인이 된다. 이때 각각의 부태스크들은 각 노드에서 실행된다. 본 연구에서 전역 태스크는 [그림 1]과 같이 프로세스 관리자에 의하여 감시 및 제어된다. 프로세스 관리자는 전역 태스크의 부태스크를 노드에 할당하여 지역 스케줄러의 우선순위에 따라 수행하게 한다. 또한 감시를 통하여 전역 태스크의 부태스크 수행이 완료되면 즉시 뒤따르는 부태스크를 노드에 할당하게 된다.



[그림 1] 시스템 모델

전역 태스크 T가 프로세스 관리자에 의하여 최초로 처리될 때, 데드라인인 $dl(T)$를 알고 있다고 가정하였으며, 예상 수행시간은 실제 수행시간과 동일하다고 가정하였다.

2장에서의 가상 데드라인 할당방법은 전위 부태스크의 실행이 후위 부태스크에 영향을 주어 연속적인 후위 부태스크의 실패를 가져올 수 있다. 본 연구에서 제안하는 방법은 맨 처음 수행되는 부태스크의 가상 데드라인을 짧게 함으로써 우선순위를 높여 먼저 수행되게 한다. 그리고 선행되는 부태스크가 데드라인을 어기지 않은 경우에는 선행되는 부태스크의 가상 데드라인을 전체 데드라인에서 뺀 나머지를 잔여슬랙으로 이용한다. 부태스크의 도착시간 대신 이전 부태스크의 가상 데드라인을 이용함으로써 부태스크의 우선순위가 낮아지는 것을 방지한다. 한편 선행되는 부태스크가 데드라인을 어긴 경우에는 해당 부태스크의 도착시간을 적용하여 가상 데드라인을 결정한다. 이를 통하여 지역 태스크와 전역 태스크간의 우선순위 경쟁에서 공평성을 갖도록 한다. 선행하는 부태스크가 데드라인 이전에 수행을 완료했을 경우 뒤따르는 부태스크의 데드라인은 아래와 같이 표현된다.

$$dl(T_i) = ar(T_i) + pex(T_i) + \left[dl(T) - dl(T_{i-1}) - \sum_{j=i}^{m} pex(T_j) \right] \cdot \left(pex(T_i) / \sum_{j=i}^{m} pex(T_j) \right)$$

4. 성능 평가

본 실험에서의 환경은 아래와 같이 설정하였다.

시스템에서는 k개의 노드가 존재하며, 각 노드에서는 비선점형 방식으로 태스크의 데드라인에 따르는 EDF방식을 사용하여 스케줄링하며, 만약 동일한 우선순위가 있는 경우 FIFO방식으로 스케줄링 된다.

지역 태스크는 평균 도착시간이 $1/\lambda_{local}$인 포아송 분포로 생성된다. k개의 노드가 있으므로 평균 도착율은 단위시간 당 $k \cdot \lambda_{local}$이 된다. 또한 지역 태스크의 수행시간은 평균 $1/\mu_{local}$을 갖는 지수분포로 생성되며 작업률은 $k \cdot \lambda_{local}/\mu_{local}$이 된다. 본 실험에서는 μ_{local}을 1로 설정하였으며, 지역 태스크의 슬랙은 $[S_{min}, S_{max}]$의 변위를 갖도록 균등하게 분포된다.

전역 태스크의 경우 평균 도착시간이 $1/\lambda_{global}$인 하나의 스트림을 갖는 포아송 프로세스를 통해 생성된다. 간략화된 실험 환경을 위하여 전역적 태스크는 m개의 부태스크로 이루어졌으며, 모든 부태스크는 수행시간이 평균 $1/\mu_{subtask}$를 갖는 지수분포를 이루도록 하였다. 그러므로 전역 태스크의 작업률은 $m \cdot \lambda_{global}/\mu_{subtask}$가 된다[3]. 부태스크의 수행에 필요한 노드의 선택 시 어느 특정 노드에 집중되는 것을 막기 위하여 임의로 태스크를 노드에 할당한다.

전역 태스크와 지역 태스크간의 유연성을 위하여 매개변수 rel_flex를 사용한다. 만약 전역 태스크의 슬랙 분포가 [a,b] 구간에서 일정하다면 a와 b를 아래와 같은 수식을 통하여 계산할 수 있다.

$$\left(\frac{a}{m/\mu_{subtask}}\right)/\left(\frac{S_{\min}}{1/\mu_{local}}\right) = rel\_flex, \quad \left(\frac{b}{m/\mu_{subtask}}\right)/\left(\frac{S_{\max}}{1/\mu_{local}}\right) = rel\_flex$$

rel_flex가 1인 경우 전역 태스크와 지역 태스크는 동일한 유연성을 갖도록 분포된다. 일반적으로 rel_flex가 1보다 큰 경우 전역 태스크가 지역 태스크보다 데드라인을 어기지 않게 된다.

시스템의 부하는 정규화된 부하를 갖는다. 이는 전체 시스템 처리 용량에 대한 생성된 작업의 비율로 나타낼 수 있다.

$$load = \left[m \cdot \frac{\lambda_{global}}{\mu_{subtask}} + k \cdot \frac{\lambda_{local}}{\mu_{local}} \right] / k$$

만약 안정한 시스템이라면 $0 \le load < 1$이 된다.

시스템 부하에 대한 지역 태스크의 비율을 위하여 매개변수 frac_local이 사용된다.

$$frac\_local = \left[k \cdot \frac{\lambda_{local}}{\mu_{local}} \right] / \left[m \cdot \frac{\lambda_{global}}{\mu_{subtask}} + k \cdot \frac{\lambda_{local}}{\mu_{local}} \right]$$ 로 정의되며 만약 지역 태스크가 하나도 없을 경우 frac_local은 0이 되며, frac_local이 1인 경우는 전역 태스크가 하나도 존재하지 않는 경우를 나타낸다. frac_local이 0.5인 경우는 지역 태스크의 수와 전역 부태스크의 수가 동일한 경우를 나타낸다.

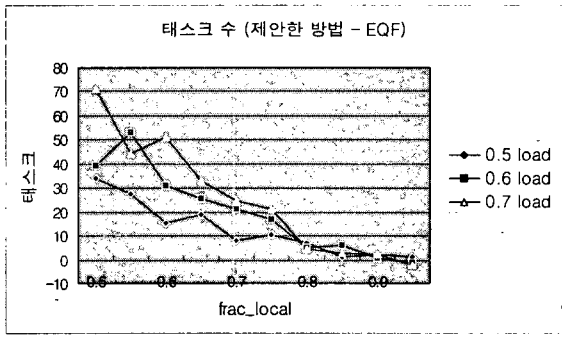
[표 2]는 모의 실험에서 사용된 매개변수이다.

매개변수	값
과부하 관리 정책	계속 수행
노드 스케줄링 기법	EDF
$\mu_{subtask}$	1.0
μ_{local}	1.0
k (노드 수)	6
m (전역 태스크의 부태스크 수)	6
$[S_{min}, S_{max}]$	[0.25, 2.5]
rel_flex	1.0
pex(X)/ex(X)	1.0

[표 2] 모의 실험 매개변수

본 연구에서는 EQF방식과 제안한 방법과의 성능을 모의 실험을 통하여 수행하였다. 실험 환경은 윈도우즈 NT 서버 기반의 펜티엄 II PC에 SMPL[4]을 사용하였으며, 컴파일러로 마이크로 소프트 Visual C++ 6.0을 사용하였다. 난수 생성을 위하여 프리웨어 버전인 RandNumGen[2]을 이용하였다.

제안한 방법과 EQF방법에서 부태스크의 가상 데드라인을 어긴 태스크의 수와 데드라인을 어긴 지역 태스크의 수는 제안한 방법이 많다. 이유는 제안한 방법이 전역 태스크의 초기 부태스크의 우선순위를 높게 책정함으로써 가상 데드라인을 어기게 되는 경우가 발생하기 때문이다. 도착율(λ)을 변화시키면서 태스크를 임의 생성한 경우의 결과는 [그림 2]와 같으며 태스크의 수는 제안한 방법과 EQF간의 차이다. 위 실험 결과에서 EQF는 전역 태스크의 잔여 슬랙들을 후속 부태스크들이 상속받게 되어 우선순위가 낮아지게 되며, 실패한 태스크는 계속해서 실패할 확률이 높아지게 된다. 초기 부태스크가 전역 태스크의 일정 슬랙을 갖게 되므로 우선순위가 높지 못해 지역 태스크에 대하여 우선적으로 수행될 수 없다.



[그림 2] 제안한 방법과 EQF간의 태스크 수행

제안한 방법은 이러한 지역 태스크와의 차이를 극복하기 위한 것으로 최초 부태스크에 높은 우선순위를 부여하였고, 부태스크가 가상 데드라인 이전에 완료된 경우 잔여 슬랙을 뒤따르는 부태스크가 가질 수 없도록 하였다. 전역 부태스크의 비율이 높고 시스템의 부하가 높은 경우 전역 태스크의 실패율은 같지만 더 많은 태스크를 생성하고 수행할 수 있었다.

5. 결론 및 향후 연구

분산 연성 실시간 시스템에 대한 스케줄링은 여러 가지 다양한 방법이 있을 수 있다. 본 연구는 다양한 분산 연성 실시간 시스템에서의 한 방법으로써 전역 태스크의 우선순위를 높이고 잔여슬랙을 후위 부태스크가 상속받지 않게 함으로써 추가적인 데드라인을 어기지 않고 EQF방법보다 더 많은 전역 태스크가 수행되도록 하였다.

향후 연구 과제는 다양한 분산 연성 실시간 시스템에서 사용될 수 있도록 본 연구에서 가정했던 부분에 대한 연구가 필요하다.

6. 참고문헌

[1] Randy Chow and Theodore Johnson, "Distributed Operating Systems & Algorithms," Addison-Wesley, 1997.

[2] Joseph A Fisher, Freeware version of RandNumGen, the C++ Random Number Generator, 1992. ftp.cis.ufl.edu/pub/simdigest/tools/RandNumGen.zip.

[3] B. Kao and Hector Garcia-Molina, "Deadline Assignment in a Distributed Soft Real-Time System," Proc. of the 13th International Conf. On Distributed Computing Systems, 1993.

[4] M. H. MacDougall, "Simulating Computer Systems Techniques and Tools," MIT Press, 1987.

[5] F. Panzieri and R. Davoli, "Real-Time : A Turorial," Technical Report UBLCS-93-33, 1993.