실시간 시스템를 위한 피드백 스케쥴러의 설계

심재환° 김진현 최진영
고려대학교 컴퓨터학과 정형기법 연구실
{jhsim°, jhkim, choi}@formal.korea.ac.kr

Design of Feedback Scheduler for Real Time System

Jae-Hwan SIM° Jin-Hyun KIM, Jin-Young CHOI
The Computer Theory and Formal Methods Lab, Dept. Computer Science, Korea University

요 약

실시간 시스템에서 스케쥴링 가능성 여부는 매우 중요하다. 그래서 실시간 시스템이 주어진 시간적 제약 사항을 만족 시킬 수 있도록 스케쥴러 등 많은 연구들이 진행되고 있다. 그런데 기존의 스케쥴링 방식은 외부 환경이나, 태스크에 대한 정확한 정보를 요구한다. 하지만 이런 정보를 정확히 예측하는 것이 매우 힘들다. 그래서 이런 정보들에 따라 시스템의 성능이 저하 되거나, 아니면 오류를 야기 할 수 있다. 그래서 본 논문에서는 실시간 시스템의 이론에 제어 이론을 접목하여 시스템에 외부 환경에 대해 강인 하면서 높은 효율을 보일 수 있는 스케쥴러를 설계하고 제시 한다.

1. 서 론

현재 자동차, 항공, 로봇, 원자력, 철도 등의 많은 분야에서 실시간 시스템이 적용되고 있다. 이런 실시간 시스템을 개발하기 위해 실시간 운영체제가 도입이 되었고, 실시간 운영체제에 적합한 EDF(Earliest Deadline First), RM(Rate Monotonic) 등의 실시간 스케쥴러들이 적용되고 있다. 많은 실시간 시스템들이 개방형 시스템으로서 외부와의 상호작용을 하게 되어있다. 그러나 이런 시스템들은 외부에서의 작업부하나 자원 사용에 대해 많은 불확실성을 가지고 있고, 불확실성을 가진 상황에서 전통적인 실시간 스케쥴링 방식은 강인함을 가지지 못한다.

이에 본 논문에서는 실시간 시스템이 외부 환경에 대해서 강인함을 가지게 하기 위해서 실시간 스케줄링 이론에 제어 이론을 도입하고자 한다. 제어 이론은 로봇, 원자력 등의 분야에서 이미 수십 년간 연구 되어 왔고 주로 물리적인 제어분야에서 이용되어 왔다. 그리고 최근에는 제어 이론을 소프트웨어에 적용 시키려는 연구들이 활발히 진행되어 왔다. 제어 이론을 소프트웨어에 적용 시키려는 이유는 다음과 같다.

- 제어 이론은 환경의 불확실성, 비선형성, 시간에 따른 변화 등에 대해서 강인한 성능을 보장해 준다. - 많은 실시간 시스템은 개방형 시스템이고 이 시스템이 주변 환경과의 반응에 따른 시스템의 작업 부하 및 스케줄링 가능성 등을 정량적으로 정확히 예측하기 힘들다. 따라서 실시간 시스템의 이론에 제어 이론을 적용 시키면 불확실한 환경에 대해서 강인한 시스템을 구축할 수 있다.
- 제어 이론은 정확한 시스템 모델을 필요로 하지 않는다. - 실시간 시스템의 스케쥴러가 정확한 동작을 하기 위해선 태스크들에 대한 정확한 정보를 필요로 한다. 또한 시스템이 태스크들에 대해 알아야 할 정보들이 많이 존재한다. 태스크의 집합, 태스크의 도착시간, 태스크의 주기, 태스크의 데드라인, 태스크의 수행 시간, 시스템 자원의 사용 여부 등이 그것이다. 하지만 태스크 들의 이러한 정보를 정확하게 예측하는 것은 매우 어려운 일이다. 특히 태스크의 수행시간 같은 것은 시스템이 파이프라이닝을 사용하는지, 캐쉬를 사용하는지, 슈퍼스카라 구조인지 등의 경우에 따라 매우 예측하기 힘들며 WCET(Worst Case Execution Time)를 구해서 스케줄링을 하게 된다. 그러나 이 경우는 시스템의 사용 율이 매우 낮아지게 (Under Utilization) 되는 요인이 된다. 하지만 스케쥴러에 제어이론을 적용하게 되었을 경우는 이러한 정확한 태스크의 정보가 없더라도 시스템의 사용율은 극대화 하면서 태스크의 데드라인을 놓치는 태스크의 비율은 극소화 시킬 수 있다.
- 피드백 제어 이론을 이용하여 시스템의 효율을 향상 시킬 수 있다. - 피드백 제어 이론은 시스템에 어떤 Set Point가 주어지면 시스템이 그 Set Point에 근접하도록 해준다. 따라서 이런 과정을 통해 시스템의 효율을 극대화 시킬 수 있다는 장점이 있다.

위와 같은 이유로 본 논문에서는 실시간 스케쥴링 이론에 피드백 제어 이론을 접목 시키려 한다. 앞으로 제시될 내용은 다음과 같다. 2장과 3장에서는 배경지식으로 실시간 스케쥴링과 제어 이론에 대해 언급 될 것이다. 4장에서는 제시 하려는 피드백 스케쥴러의 구조와 설계에 대해서 언급이 될 것이다. 5장에서는 간단한 실험 내용을 언급하고, 6장에서 결론을 짓겠다.

2. 실시간 스케줄링

실시간 시스템에 있어서 가장 중요한 문제는 시스템이 주어진 시간적 제약을 만족하느냐 이다. 그래서 이런

시간적 제약을 엄격히 관리하기 위해 나온 운영체제가 실시간 운영체제 이고, 이 운영체제의 스케쥴러가 RM, EDF와 같은 스케쥴러 이다. 그러나 이러한 스케쥴러 들은 작업 부하 나 태스크의 실행 시간 등을 정확히 알고 있어야만 잘 동작할 수 있다. 예측 불가능한 작업 부하 등에 대해서는 좋은 않은 효율을 내거나 잘 동작하던 시스템의 스케쥴링이 깨어질 수 있다. 스케쥴링이 깨어지지 않게 하기 위해서 CPU의 여유도를 크게 하여 시스템을 설계할 수 있겠지만 이 때 CPU는 매우 낮은 Utilization을 가지게 된다. 대부분의 실시간 내장형 시스템은 넉넉하지 않은 자원을 가지고 있기고, 자원은 비용과 연관되기 때문에 CPU Utilization이 낮아지는 것은 매운 바람직하지 못한 상황이다. 따라서, 스케쥴링이 깨어지지 않는 범위에서 CPU Utilization을 극대화 시킬 수 있고, 불확실한 외부 작업 부하에 대해 강인하게 동작할 수 있는 시스템이 필요하다. 그래서 실시간 시스템 이론에 제어 이론을 접목하려 하는 것이다.

3. 제어 이론

아래의 그림 1은 일반적인 제어 시스템의 구조를 나타낸 그림이다.

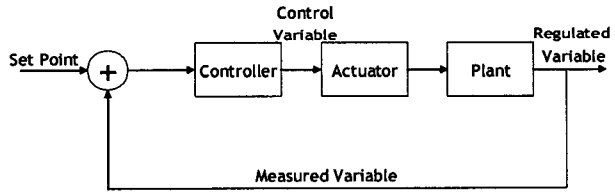

그림 1 제어 시스템의 구조

위의 그림에서 Plant는 실제 제어할 대상이다. 예를 들면 일반적인 로봇제어시스템에서의 로봇 팔이나 로봇의 바퀴 물리적인 파트이다. Actuator는 Plant를 동작 시켜 주는 부분이다. 일반적인 제어 시스템에서는 Motor등이 좋은 예이다. Controller(제어기)는 입력받은 Set Point와 피드백 받은 Variable로 계산된 값에 의해 Plant를 적절히 제어해 주는 부분이다. PID Controller, PD Controller, PI Controller 등등 수많은 Controller가 존재하지만, 90% 이상의 제어 시스템에서 PID Controller가 사용되고 있다. 본 논문에서도 PID Controller를 이용하여 실시간 시스템의 스케쥴러를 제어한다. 본 논문에서 Set Point는 Desired Miss ratio이고, Control Variable은 CPU Utilization 이다. 그리고, Regulated Variable은 실제 계산된 Miss ratio가 된다. 구간 $(th,\ (t+1)h)$ 에서의 Miss ratio를 $M(t)$, CPU Utilization을 $U(t)$라 하면, $M(t)$ 와 $U(t)$는 다음과 같이 정의 된다.

- $M(t) = \dfrac{No.\ of\ deadline\ missed\ tasks}{No.\ of\ tasks(complete\ a\ aborted)}$
- $U(t) = \dfrac{h-(CPU\ Idle\ Time)}{h}$

위와 같은 구조에서 구간 $(th,\ (t+1)h)$에서의 $M(t)$와 Desired Miss ratio M_s를 제어기의 입력으로 하여 입력에 따라 제어기가 태스크들의 Period를 조정해 줌에 따라 $U(t)$를 조절할 수 있다. 쉽게 말하자면, Miss ratio가 증가 하려고 하면 Utilization을 낮추고 Miss ratio 가 감소 하려 하면 Utilization을 높여서 Miss ratio가 거의 0인 상태에서 Utilization을 극대화 시킬 수 있다.

4. 피드백 스케쥴러의 구조 및 설계

다음의 그림 2는 피드백 스케쥴러의 구조를 나타내는 그림이다.

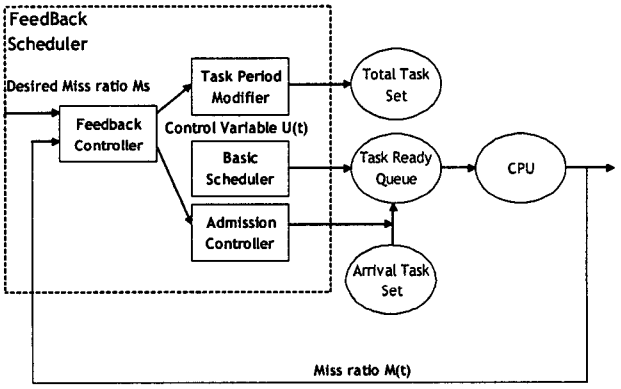

그림 2 피드백 스케쥴러의 구조

피드백 스케쥴러의 구조는 위의 점선으로 표시된 영역과 같다. 피드백 스케쥴러의 입력으로 Set Point인 Desired Miss ratio M_s가 들어가고 그 결과로 실제 Miss ratio인 $M(t)$가 나오게 된다. 이 때 M_s와 $M(t)$의 차이를 $E(t)$라 하면, $E(t) = M_s - M(t)$가 되고, PID 제어기에 의해 제어 변수인 ΔCV(Control Variable)값이 정해진다.

$\Delta CV = G_P E(t) + G_I \int E(t)dt + G_D \dfrac{dE(t)}{dt}$ 에 의해 계산된다. 여기서 G_P, G_I, G_D값은 PID 제어기의 제어 이득값인데, 근궤적법에 의해 시스템이 안정화 되어 지는 이득값을 얻어 질 수 있다. 이렇게 계산 된 ΔCV에 의해서 적절한 Utilization을 갖도록 태스크들의 주기를 수정할 수 있다. 이 때 위의 구조를 갖는 실시간 시스템을 아래와 같이 가정한다.

가정) 위의 피드백 스케쥴러는 태스크들의 데드라인이 엄격하게 어느 한 순간으로 고정되어 있지 않는 시스템에 사용된다.

본 논문의 피드백 스케쥴러는 CPU Utilization이 낮을 때는 태스크가 태스크 준비 리스트에 들어가는 주기를 빨리하여 CPU Utilization을 높이고, CPU Utilization이

높을 때는 태스크가 태스크 준비 큐에 들어가는 주기를 느리게 하여 CPU Utilization을 낮추는 방식을 이용하게 되는데 이처럼 태스크의 주기를 낮추게 되면 태스크의 데드라인도 조금 씩 바뀌게 될 수 있으므로, 데드라인이 아주 엄격히 정해진 시스템에서는 사용 될 수 없다. 따라서 이 시스템에서는 데드라인을 유연하게 정하고, 시스템에 문제가 생기지 않는 선에서 태스크의 주기를 늘리거나 줄이거나 하여 Miss Ratio를 원하는 수준에 맞추면서 CPU Utilization을 극대화 시킨다. Task Period Modifier에서 더 이상 태스크의 주기를 늘릴 수 없을 때에는 Admission Controller에 의해 더 이상 태스크가 준비 리스트에 들어가는 것을 막는다. 그리고, Basic Scheduler는 기본적인 실시간 스케쥴링 정책에 따라 준비 리스트에 있는 태스크들을 스케쥴링 해준다. Basic Scheduler로는 Fixed Priority Scheduler를 사용 하였다.

5. 시뮬레이션

시뮬레이션은 기존의 Fixed Priority Scheduler만 사용 한 것과 Fixed Priority Scheduler에 제어 이론을 접목한 피드백 스케쥴러를 비교 하였다. 실제 시스템에 탑재하여 실험한 것이 아니라, MATLAB을 이용하여 시뮬레이션을 실행 하였다.

- 시스템의 모델링
- CPU의 1클럭 Tick 시간 : 10ms
- 주기적인 태스크의 개수 : 50개
- 주기적인 태스크 들의 주기 : 5Tick
- 각 태스크들의 수행 시간 : 0.3 - 0.5ms
- 성능 비교 방법 : 처음 10Tick은 외부에서 주어 지는 작업 부하 없이 동작 시키고, 10-20 Tick 사이에서 작업 부하를 가한다.

위와 같은 조건으로 시뮬레이션 한 결과는 아래와 같다.

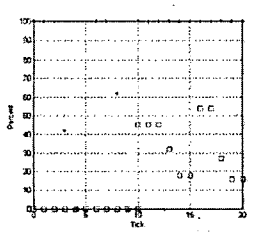

그림 3 Fixed Priority Scheduler 결과

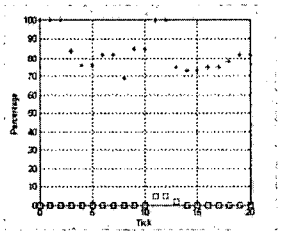

그림 4 Feedback Scheduler 결과

위의 그림 3은 Fixed Priority Scheduler만을 이용하여 시뮬레이션을 한 결과 이고, 그림 4는 Feedback Scheduler를 이용하여 시뮬레이션을 한 결과이다. 위 의 그래프에서 *는 CPU Utilization을 의미 하고, 네모 표시는 Deadline Miss Ratio를 의미한다. 그림 3의 결과에서 알 수 있는 것은 기본적인 실시간 스케쥴러만을 사용 할 때에 외부에서의 작업 부하가 없는 0에서 10 Tick 사이에서는 잘 동작되지만 예측 못한 작업부하를 가한 10 Ti 에서 20Tick에서는 태스크들이 많은 수 Deadline을 넘은 결과를 알 수 있었다. 또한 작업을 다 처리한 3Tick과 5Tick 사이와 8Tick과 10Tick 사이에서는 CPU가 Under Utilization 됨을 알 수 있었다. 이에 반해서 피드백 스케쥴러에서는 외부 작업 부하에 대해 강인함을 알 수 있고, 또한 지속적으로 일정 범위 내에서 CPU Utilization이 유지 됨을 알 수 있었다.

6. 결론 및 향후 연구

6장에서의 시뮬레이션의 결과에서 알 수 있듯이 기본적인 스케쥴러 만을 사용하였을 때, 예측하기 어려운 작업 부하가 없었을 때에는 Deadline Miss 없이 잘 동작하였으나 CPU의 Utilization이 낮은 것을 알 수 있었고, 작업부하를 늘리게 되면 CPU Utilization이 증가 하지만, 많은 수의 태스크 들이 Deadline을 Miss 하는 것을 알 수 있었다. 본 논문에서 제시하는 방법인 태스크의 주기를 조정하는 방법은 엄격히 제한 된 데드라인을 가진 실시간 시스템에서 문제점이 있지만, 시스템이 인내할 수 있는 범위에서의 유연한 데드라인을 가지고 Deadline Miss를 최소화 하며 CPU Utilization을 극대화 할 수 있음을 알 수 있었다. 또한 과도한 양의 작업 부하에 대해서도 기본적인 스케쥴러 만을 사용한 실시간 시스템에 비해 강인함을 나타냄을 알 수 있었다. 향 후 연구 과제로는 설계와 시뮬레이션에 그친 피드백 스케쥴러를 Fixed Priority Scheduler를 가진 uCOS-II에 구현하여 성능 분석 및 실시간적인 요소를 만족하는 지에 대해 연구할 예정이다. 이를 통해 이 스케쥴러의 실제 사용 가능성에 대한 분석이 가능할 것이다.

7. 참고 문헌

[1] David C. Steere, Ashvin Goel, Joshua Gruenberg, Dylan McNamee, Calton Pu, and Jonathan Walpole, "A Feedback-driven Proportion Allocatior for Real-Rate Scheduling", In the Proceedings of the 3rd Symposium on Operating Systems Design and Implementation 1999

[2] C. Lu, J. Stankovic, G. Tao and S. Son, Feedback Control Real-Time Scheduling: Framework, Modeling and Algorithms, special issue of RT Systems Journal on Control-Theoretic Approaches to Real-Time Computing, Vol. 23, No. 1/2 July/September, 2002, pp. 85-126.