다중프로세서 시스템에서 고장 감내 실시간 스케줄 기법

구현우° 홍영식

동국대학교 컴퓨터공학과

windole9°@hotmail.com, hongys@dongguk.edu

Fault tolerant real-time task scheduling approach in Multiprocessor Systems

Hyun-Woo Koo° ,Young-Sik Hong

Dept. of Computer Engineering, Dongguk University

요 약

많은 산업 분야에서 사용되는 실시간 시스템은 논리적 정확성뿐만 아니라 시간적 정확성을 요구한다. 그래서 실시간 시스템에서 동작하는 작업들은 항상 마감시간을 지키기 위해 작업에 대한 스케줄링이 매우 중요한 요소가 된다. 그리고 시스템의 신뢰도를 높이기 위해서는 고장 감내가 반드시 필요하다. 특히, 현대 기술의 발달로 일부 분야에서 사용되어 지던 실시간 시스템이 실시간 내장형 시스템 형태로 다양한 분야에 사용되어 진다. 내장형 시스템을 개발하는데 있어 자원의 절약 또한 하나의 중요한 요소이다. 따라서 본 논문에서는 실시간 시스템에서의 작업들이 마감 시간을 만족하며 고장 감내를 보장하는 시간 중복 기법과 백업 기법을 혼합하여 주기적으로 동작하는 작업들의 신뢰도와 자원의 효율 및 절약을 높이는 스케줄링 기법을 제안하고 실험한다.

1. 서 론

실시간 시스템은 통신 시스템, 비행기 운행 관리, 원자력 발전소의 원자로 제어장치 그리고 군사용 미사일 시스템과 같은 특수 환경을 위한 시스템에 사용되어 졌다. 점차 현대의 기술이 발전하고 사용자의 요구사항이 복잡해지면서 실시간 시스템이 사용되어진 특수한 시스템뿐만 아니라 다양한 분야에서 실시간 시스템을 적용하고 있으며 실시간 내장형 시스템도 개발되었다.

이러한 실시간 시스템은 논리적 정확성에 의해 올바른 결과를 도출할 뿐만 아니라 시간적 정확성을 지켜야 한다. 시간적 정확성을 위해서는 실시간 시스템에서 동작되는 모든 작업은 시간 제약 조건, 마감시간을 만족해야만 한다. 그러므로 실시간 시스템을 개발하기 위해서 효율적인 작업 스케줄링 기법 연구가 필요하다. [7]

그리고 실시간 시스템이 사용되는 많은 분야는 위험성을 내포하고 있기 때문에 시스템의 신뢰도 또한 중요한 요소이다. 시스템의 신뢰도를 높이기 위해서는 고장이 발생 했을때 빠른 대처를 하여야 한다. 또한 실시간 시스템은 시간의 제약 조건을 반드시 지켜야 하는 특징을 지니고 있다. 이 두 가지 요소를 바탕으로 실시간 시스템의 신뢰도를 중요하게 생각한다.

실시간 내장형 시스템의 개발에서 또한 중요한 요소는 자원의 효율적인 사용을 들 수 있다. 이에 본 논문에서는 자원의 효율적인 사용과 시스템의 신뢰도를 보장할 수 있는 기법을 제안하고 실험을 통해 기존의 고장 감내를 위한 액티브 백업 기법보다 자원 절약 및 더 나은 성능을 보여준다.

2. 관련 연구

실시간 시스템의 작업을 스케줄링 하기 위해 널리 사용되는 스케줄링 알고리즘은 Liu와 Layland가 제안한 RMS(Rate Monotonic Scheduling)알고리즘과 EDF(Earnest Deadline First)알고리즘이다. RMS는 $U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq n(2^{1/n} - 1)$에서 최적의 알고리즘임이 증명 되었다.[5]

그 후 많은 연구는 RMS와 EDF를 변환하고 확장하는 알고리즘들이 소개되어 졌다. 단순히 스케줄 알고리즘을 변환하고 확장하는 것은 시스템의 스케줄 가능성을 높이고 높은 효율을 보장하지만 시스템의 신뢰도를 높이는 방법이 되지 않는다. 따라서, 시스템의 신뢰도를 높이기 위해서는 고장 감내가 필수적이다.

실시간 시스템에서 고장 감내를 위해 사용되는 기법으로는 크게 가중치 값을 기초로 하여 재 스케줄링하는 기법과 중복(Redundancy) 기법이 널리 이용되어 지고 있다. 가중치 값을 기초로 하는 스케줄링 기법은 고장이 발생한 시점에서 고장에 영향을 받는 작업들의 가중치 값을 계산하여 각각의 작업에 새로운 우선순위를 할당하여 재 스케줄링하는 방법이다.

고장 감내를 위한 두 번째 기법인 중복기법은 크게 세 가지로 구분 된다. 하드웨어, 소프트웨어 그리고 시간 중복이다. 하드웨어 중복 기법은 시스템과 유사한 또 다른 시스템(Backup Systems)을 두어 고장에 대해 대비를 하는 기법을 말한다.

그리고 소프트웨어 중복 기법은 작업에 대한 백업 복사본(Backup Copy)를 두어 일시적 또는 영구적 고장으로 인해 작업이 시간의 제약조건 내에 올바르게 실행치 못할 경우 백업 복사본을 사용한다. 작업이 실행되고 있는 프로세서가 아닌 다른 프로세서에 백업 복사본을 할당하여 실행시킴으로써 고장 감내를 보장하는 기법을 말한다. 소프트웨어 중복 기법은 백업 기법이라고도 칭한다. 이 백업 기법은 작업의 백업 복사본이 병렬처리 되는 액티브 백업 기법

과 작업이 고장으로 인해 올바른 동작을 방해 받게 되면 백업 복사본을 실행시키는 패시브 백업 기법으로 나뉜다. 액티브 백업 기법은 구현이 쉽지만 작업들을 실행하기 위해서는 두 배의 프로세서 수가 필요한 단점을 지니고 패시브 백업 기법은 액티브 백업 기법보다 적은 수의 프로세서를 필요로 하지만 작업의 마감시간을 만족하기 위해서 작업들의 실행시간이 마감시간의 1/2보다 작아야 하는 단점을 가지고 있다. [8] 백업 복사본을 오버로딩하여 고장 감내를 보장하면서 작업을 동작하게 하는 프로세서의 수를 줄이는 방법도 제안되었다.[6] 이러한 오버로딩 기법은 여러 프로세서에서 동시에 일어나는 고장은 고려하지 않고 있다.

시간 중복 기법은 하나의 프로세서에서 동작되는 작업들에 의해 사용되지 않는 시간을 재사용하는 기법이다. 시간 중복을 이용하면 일시적인 고장에 대해서 재실행하여 시스템의 신뢰도를 보장하면서 시간 제약조건을 만족할 수 있다 [1],[2]

3. 제안 모델

본 논문에서의 실시간 시스템은 여러 개의 주기적인 작업 집합으로 구성되어 진다. 각각의 작업은 도착시간(A_i), 실행시간(C_i), 주기(P_i) 그리고 마감시간(D_i)의 파라미터를 가진다.

$$T = \{t_i \mid t_i = (A_i, C_i, D_i, P_i), 1 \leq i \leq n\} \quad (1)$$

그리고 각각의 작업은 서로 독립적이고 우선순위에 의해 서로 선점이 가능하며, 주기와 마감시간이 같다고 가정한다.

실시간 시스템에서 고려해야할 고장은 크게 영구적인 고장과 일시적인 고장으로 나눌 수 있다. 영구적인 고장은 하드웨어의 고장 또는 그 외 다른 이유로 작업이 할당된 프로세서가 사용이 불가능한 경우이고, 일시적인 고장은 소프트웨어 버그로 인해 프로세서에 할당된 하나의 작업에 대한 결과가 올바르지 않게 나오는 경우를 말한다. 이러한 일시적인 고장은 작업의 마지막에 감지할 수 있다. 본 논문에서는 일시적인 고장을 고려하며 고장이 발생한 시점에서 하나의 작업만 고장에 영향을 받는 것으로 가정한다. 즉 고장에 대한 다른 작업으로의 전이가 일어나지 않는다.

실시간 시스템의 높은 신뢰도를 보장하고 시스템 자원의 재활용 및 절약을 위해 기존에 연구되어진 시간 중복을 통한 고장 감내 기법과 백업을 통한 고장 감내 기법을 혼용하여 신뢰도를 높이는 방안을 제안한다. 기존에 연구 되어진 시간 중복을 통한 고장 감내 스케줄링 기법들은 작업의 실행 시간이 작업의 주기 내에 두 번 이상 실행이 가능할 만큼 적은 것을 가정하거나 작업을 중요한 동작 부분과 선택적인 동작 부분으로 나누어 연구되어 왔다.[3] 본 논문에서는 이러한 작업에 대한 가정을 두지 않고 고장 감내를 가능케 하는 스케줄링 기법을 살펴보도록 하겠다.

하나의 프로세서에 할당되는 작업을 다음의 공식과 같이 크게 두개의 그룹으로 나누어 각각의 작업에 다른 고장 감내 기법을 적용한다. 작업의 그룹화 기법은 간단하게 실행시간 대 마감시간의 비율이 2보다 작은 작업과 2보다 크거나 같은 작업으로 나눈다.

$$T_L = \{t_i \mid D_i/C_i > 2 (1 \leq i \leq n)\}$$
$$T_H = \{t_i \mid D_i/C_i \leq 2 (1 \leq i \leq n)\} \quad (2)$$

프로세서에 각각의 작업들을 할당하기 위해서 스케줄링 가능성을 검사하기 위해 RMS에서 사용된 한정공식을 이용하고 또한 고장을 고려한 스케줄링 가능성을 동시에 검사한다. 고장을 고려한 한정값은 고장이 발생했을 시 작업 집합 T_L에 포함되는 작업일 경우 작업의 실행이 한 주기에 두 번 동작하는 것으로 가정하여 계산한다. 두 조건을 만족하면 해당 프로세서에 작업을 할당하고 EDF 기법을 이용하여 마감시간이 짧은 작업에 높은 우선순위를 주어 작업을 스케줄링 한다.

$$U_{arv} \leq m(2^{1/m} - 1) \text{ and } U_f \leq 1 \quad (3)$$

$$U_{arv} = \sum_{i=1}^{n} \frac{C_i}{P_i} \quad (4)$$

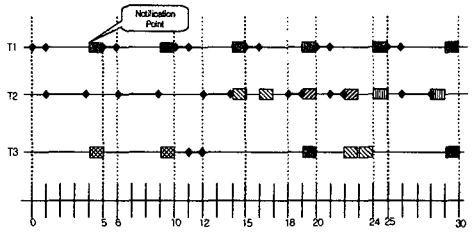
$$U_f = \sum_{i \in \{T_L\}} \frac{(C_i + f_r C_i)}{P_i} + \sum_{i \in \{T_H\}} \frac{C_i}{P_i}, \; f_r : fault\ ratio$$
$$= \sum_{i=1}^{m} \frac{C_i}{P_i} + \sum_{i \in \{T_L\}} \frac{f_r C_i}{P_i} \quad (5)$$

프로세서에 작업이 할당되어 지고 나면 낮은 실행시간 대 마감시간 비율을 가지는 작업 즉, 작업 집합 T_L에 포함되는 작업에 대해서 시간 중복 고장 감내를 보장키 위해 NP(Notification Point)를 계산한다. NP는 Backward RM 기법을 이용한다.[4]

$$N_i = \{n_{ij} \mid n_{ij} = j \times D_i - C_i, \; (1 \leq j \leq \frac{LCM}{D_i}), i \in T_L\} \quad (6)$$

구해진 NP는 해당 작업이 고장에 의해 올바르게 동작치 못할 경우에만 사용되어 진다.

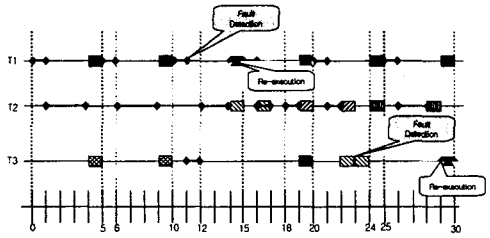
만약 프로세서에 할당되어진 작업들 중에 하나의 작업이 일시적인 고장으로 인해 올바르게 동작치 못하면 고장에 영향을 받은 작업이 T_L그룹에 속하면 해당 NP가 활성화 되어 NP 시점에서 다른 작업보다 가장 높은 우선순위를 가져 만약 실행중인 다른 작업이 있다면 작업을 선점하여 마감시간을 지키게 된다. 그렇지 않고 T_H그룹에 속하면 백업 기법에 의해 다른 프로세서에서 액티브하게 동작중인 백업 복사본의 결과를 취하게 된다. T_L 그룹 내의 작업이 마감시간 전에 올바른 동작 완료하면 해당 NP는 무시되고 일시적인 고장 감내를 위해 할당되어진 시간은 다른 작업의 실행을 위해 할당된다.



[그림 1] Notification point 계산 결과

예를 들어 프로세서에 할당될 작업이 t_1=(0, 1, 5, 5), t_2=(0, 3, 6, 6), t_3 = (0, 1, 10, 10) 세 개가 존재하면 공식 (2)에 의해 T_L={t_1, t_3}, T_H={t_2}로 분류되어 진다 그리고 공식 (4)와 공식 (5)에 의해 U_{arv}=0.8 이고 U_f =

0.9가 되어 공식 (2)의 조건을 만족하므로 스케줄이 가능하다 그리고 t_1과 t_3의 NP를 공식 (6)으로 구하면 N_1={4, 9, 14, 19, 24, 29}, N_3={9, 19, 29}가 된다. [그림 1]은 작업들을 EDF기법을 이용하여 스케줄링 하고 계산되어진 NP를 표시한 그림이다. t_2의 스케줄링 형태를 보면 빗금이 쳐진 각각의 두개 쌍의 블록을 볼 수 있다 이러한 블록은 t_1이나 t_3에 의해 앞에 존재하는 블록이 선점 당하게 되면 뒤의 블록에서 작업을 실행함을 뜻한다.

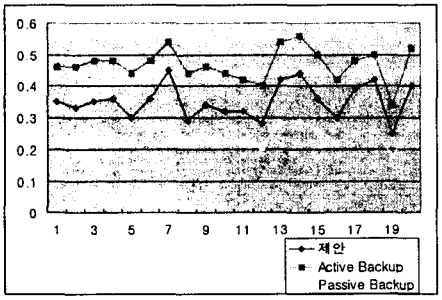


[그림 2] 고장 발생시 재실행 결과

[그림 2]는 시점 11과 22에서 일시적인 고장이 발생하였을 시의 고장 감내를 보여주고 있다. 시점 11에서 고장에 의해 t_1의 작업이 올바르게 실행되지 못하였으므로 현 작업의 NP인 시점 14에서 t_2의 실행을 선점하여 재 실행 되고 t_2는 시점 16에서 나머지 작업이 실행되어 두 작업 모두 마감시간을 지키는 것을 볼 수 있다.

4. 실험 및 분석

제안된 스케줄링 기법을 자원의 재활용과 절약의 측면에서 액티브 백업 기법 및 패시브 백업 기법과 비교 실험하였다. 작업 주기는 1에서 100사이로 랜덤하게 설정하였고, 실행 시간은 해당 주기의 2/3이하의 랜덤 수로 설정하였으며 고장 비율은 0.05%로 하였다.

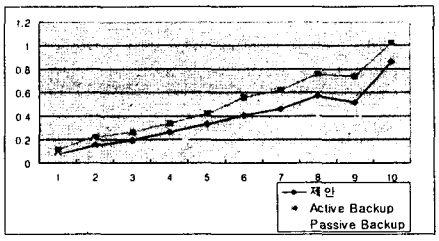


[그림 3] 프로세서 사용률 비교(n=50, f_t=0.05)

[그림 3]은 50개의 작업을 가지는 20개의 작업 집합으로 고장 감내를 보장하고 스케줄링 가능한 프로세서 사용률을 비교한 자료이다. [그림 4]는 작업의 수에 따른 프로세서 사용률을 나타내는 그래프이다.

두 실험 결과를 보면 액티브 백업 기법 보다 제안 기법이 약 25%의 프로세서 절약을 보이고 패시브 백업 기법 보다는 약 10% 이상의 프로세서가 더 사용되어 지는 것을 볼 수 있다.

하지만 패시브 백업 기법은 여러 프로세서에서 동시에 발생하는 고장에 대해서는 감내를 하지 못한다. 또한 작업 모델을 설계할 때 마감시간이 실행 시간보다 두 배 이상 긴 시간을 가져야 고장 감내가 가능하다는 제약 조건이 따르게 된다.



[그림 4] 프로세서 사용률 비교(n=10:100, f_t=0.05)

5. 결론 및 향후 연구 과제

실시간 시스템에서 작업의 시간 제약 조건을 만족하면서 시스템의 신뢰도를 높일 수 있는 방안을 살펴보고 기존의 백업 기법을 이용한 고장 감내 스케줄링과 스케줄링 가능성을 만족하기 위해 필요한 프로세서의 수를 비교하여 자원 절약의 효과를 볼 수 있었다. 향후 과제로는 스케줄링에 의해 발생하는 프로세서의 휴지 시간을 비주기적인 작업이 프로세서에 도착했을 때 사용케하는 방안을 연구해 봄직하다. 그리고 스케줄링 가능성에 사용된 한정 값을 높여 프로세서 효율을 높이면서 자원 절약의 효과를 볼 수 있는 방안도 연구하고자 한다.

6. 참고문헌

[1] Y. Chen, X. Yu and G. Xiong. Fault-Tolerant Earliest Deadline First Scheduling with Resource Reclaim, ICA3PP'02, 2002

[2] S. Ghosh, R. Melhem, D. Mosse and J. S. Sarma, Fault-Tolerant Rate-Monotonic Scheduling, J. Real-Time Systems, 1998

[3] C. C. Han, K. G. Shin and J. Wu, A Fault-Tolerant Scheduling Algorithm for Real-Time Periodic Tasks with Possible Software Faults, IEEE Transcations on Computers, Vol 52, NO. 3, March 2003

[4] S. Lauzac, R. Melhem and D. Mosse. An Improved Rate-Monotonic Admssion Control and Its Applications, IEEE Transactions on Computers, VOL. 52, NO. 3, March, 2003

[5] C. L. Liu and J. Layland. Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment. Journal of the ACM, 20(1):46-61, 1973

[6] R. A. Omari, A. K. Somani, G. Manimaran, A New Fault-Tolerant Technique for Improving Schedulability in Multiprocessor Real-time Systems, IPDPS'01, April, 2001

[7] K. Ramamritham and J. A. Stankovic, Scheduling Algorithm and Operating Systems Support for Real-Time Systems, Proceedings of the IEEE, Vol 82, No 1, pp.55-67, January 1994

[8] C. Yang and G. Deconinck. A Fault-Tolerant Reservation-Based Strategy for Scheduling Aperiodic Tasks in Mutlprocessor Systems. EUROMICRO-PDP'02, 2002