Real-Time 시스템에서 abnormal task 자동 검출 방안

정창수
LG전자 CDMA시스템 연구소
sideh2o@lge.com

Automatic detection methods of abnormal task in real-time systems

Chang-su Jung
CDMA System Research Lab. LG Electronics Inc.

요 약

본 논문에서는 real-time환경에서 abnormal task를 자동 검출하여 시스템 overload 및 오 동작을 사전에 검출할 수 있는 방안을 제안한다. 본 논문에서 제안한 방안은 context switching이 발생하는 시점에서 각 task들의 cpu 점유율 및 context switching 횟수를 분석하여 비정상적으로 높은 cpu점유율을 가지는 task와 과도한 context switching을 일으켜 시스템에 overload를 주는 task들을 자동으로 검출한다. 이들 이용하여 신뢰성 있는 real-time 시스템 설계 및 구현을 지원할 수 있다.

1. 서론

현재 네트워크 장비 및 이동통신 분야를 포함한 다양한 분야에서 엠베디드 시스템에 대한 요구와 중요성이 증가하고 있다. 특히 시스템이 복잡해지고 처리해야 할 작업들이 많아지면서 멀티태스킹의 중요성 및 각 task의 안정성과 신뢰성에 대한 관심이 높아지고 있다. 특히 real-time 환경에서 동작하는 각 task들은 작업처리 시간에 더욱 민감하게 동작하여야 하며 결함발생으로 인해 시스템이 오 동작을 유발시키지 않도록 하기 위해 task들의 신뢰성이 절실히 필요한 상황이다[1].

Real-time 환경에서 특정 task의 결함으로 인해 정상적인 동작을 수행하지 못하거나 전체 시스템이 멈춰 버릴 경우 그 피해는 심각하다고 할 수 있다[2]. 따라서 real-time 환경에서 동작하는 task들이 자체 결함이나 오 동작으로 정상적인 업무 수행을 못할 경우를 미리 감지하고, 만일 결함이 발생하였다면 최소한의 피해로 전체 시스템을 정상상태로 복구 시킬 수 있는 기능이 필수적이다.

본 논문에서는 real-time환경에서 abnormal task를 자동 검출하여 신뢰성 있는 task관리 기능을 제공하는 방안을 제안하며, 이를 위해 context switching 과정을 이용하여 cpu 점유율 및 시스템 overload를 분석하여 abnormal task를 자동으로 검출하는 방안을 제공한다.

본 논문의 구성은 2장에서는 real-time환경에서 task scheduling 및 context switching과정에서 발생하는 시스템 overload에 대해 기술하였으며, 3장에서는 본 논문에서 제안한 검출구조 및 검출 방안에 대해 기술하였고, 4장에서는 결론 및 향후 연구과제를 기술하였다.

2. 관련 연구

Real-Time환경의 각 task들은 정해진 시간 안에 정해진 업무를 수행해야 하기 때문에 엄격한 시간제약 조건을 가지게 된다[3]. 이런 제약 조건을 만족시키기 위해 RTOS(Real-Time Operating System)는 짧은 context switching time과 우선순위에 기초한 선점형 스케줄링, 외부 인터럽트에 대한 빠른 응답 등을 제공해야 한다[4]. 멀티태스킹 환경에서는 스케줄러에 의해 context switching이 발생하며, 이 과정 동안 다른 작업은 수행 되지 않는다. 특히, 수행시간은 짧고 높은 우선순위를 가지는 task는 빈번한 context switching을 초래하여 시스템에 overload를 발생시키게 된다[5].

각 task들은 OS에서 제공하는 스케줄러에 의해 context switching이 발생하게 되며, 선점형 커널을 사용하는 대부분의 RTOS환경에서는 우선순위에 따라 cpu를 할당 받게 된다. 그러나 높은 우선순위를 가지는 task가 수행 도중에 내부 결함이나 하드웨어 결함 등으로 인해 *suspend*상태에 빠지거나 계속 cpu를 점유하는 경우가 발생될 경우, 다른 task들은 cpu를 할당 받지 못해 업무를 처리하지 못하게 된다. 이런 문제점들을 해결하기 위해 본 논문에서는 일정 시간 간격마다 context switching 시 분석한 task들의 cpu 점유율 및 context switching 횟수를 이용하여 비정상적으로 높은 cpu점유율을 가지는 task와 과도한 context switching을 일으키는 task들을 자동 검출하는 방안을 제안한다.

본 논문에서 제안한 방안은 결함 검출을 위해 전용 processor를 사용하지 않고 application 내부에서 투명하게 수행될 수 있는 구조이다. 본 방안은 context를 저장하거나 복귀시키는데 소요되는 overload는 고려하지 않았으며, 시스템에서 동작하는 모든 task는 지속적이고 주기적인 경우로 제한한다.

3. Abnormal task 자동 검출 구조

각 task들은 초기 생성시 TCB와 stack영역을 memory pool로부터 할당 받고 정해진 동작을 수행하게 된다. Task생성시 할당 받는 TCB에는 태스크이름, 우선순위, stack point, status 정보, register 정보 등을 각각 독립적으로 가진다. 이런 정보를 이용하여 각 task별 상태관리와 cpu 사용현황을 분석하여 과도한 cpu 사용 task를 구분하는데 이용한다. 또한 본 논문에서 제안한 방안은 기존 OS에서 제공하는 TCB 구조에서 사용하지 않는 reserved 필드를 재정의하여 cpu 점유율과 context switching 횟수를 저장하도록 설계하였다[6]. Abnormal task 자동 검출을 위한 전체 구조는 그림 1과 같이 세 부분으로 나누어 설계 하였다.

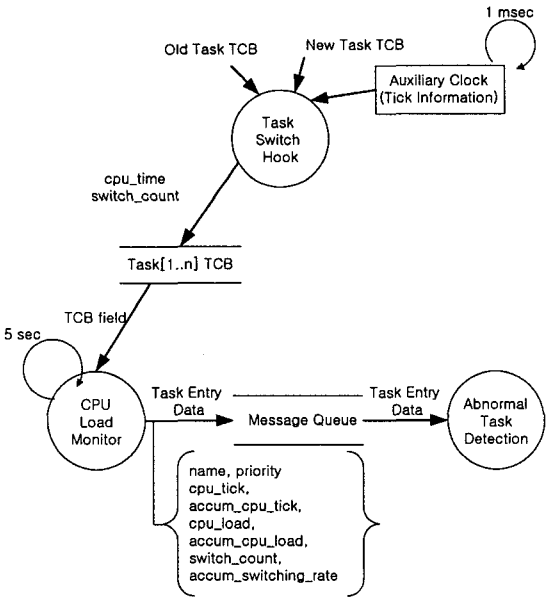

그림 1. 전체 구조도

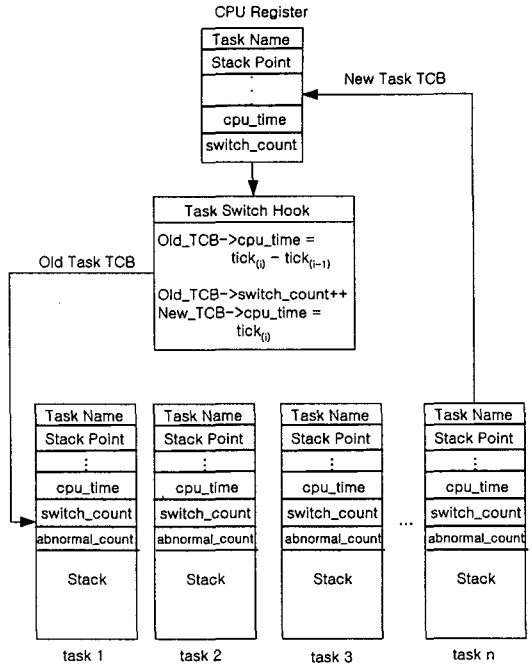

그림 2. Task Switch Hook 과정

Task Switch Hook 과정은 context switching이 발생할 때마다 task의 cpu 사용 시간 및 context switch count를 TCB에 저장하며, 정확한 시간 정보를 위해 보조 클럭을 사용한다. CPU Load Monitor 과정은 일정 시간 간격(5초)마다 TCB정보를 분석하여 task의 cpu 점유율 및 switching overload를 계산하고 이 정보를 메시지 큐로 전송한다. Abnormal Task Detection과정은 메시지 큐로부터 정보를 읽어 각 task의 cpu 점유율 및 switching overload를 분석하고 abnormal task를 검출하여 해당 task를 삭제 및 재시동 시키는 기능을 수행한다.

CPU Load Monitor 과정과 Abnormal Task Detection과정은 각각 application task로 설계하였다. context switching 과정은 수 msec이내에 수행되어 그 과정에서 많은 연산이 수행될 경우 시스템의 부하를 증가시킬 수 있기 때문에 그림 1과 같이 cpu time 및 context switching count를 계산하는 task와 수집된 정보를 이용하여 task의 결함발생 유무를 분석하는 task로 나누어 각 기능의 부하를 최소화하고 독립된 동작 환경을 제공한다.

3.1 Task switch Hook

각 task들의 context switching 과정은 이전에 cpu를 할당 받아서 수행중인 task(Old_TCB)와 새롭게 cpu를 할당 받아서 실행될 task(New_TCB)로 나누어 진다. 그림 2와 같이 task n에게 cpu를 넘겨주게 되는 task 1은 자신의 context를 TCB나 stack에 저장하게 되며 이 시점에서 이전 context switching때 저장된 시간과 현재 시간과의 차를 cpu_time 필드에 누적한다. Context switching 횟수는 switch_count 필드에 저장하여 각 task의 context switching overload를 분석하는데 사용한다.

이 과정에서 각 task의 정확한 cpu 사용시간을 계산하기 위해 보조 클럭을 사용한다. 시스템에서 제공하는 시스템 클럭(1 tick=10 msec)을 사용할 경우 몇 msec 또는 usec단위로 수행되는 context switching과정을 정확하게 측정하기 어렵기 때문에 시스템 클럭보다 정밀한 보조 클럭(1 msec)을 사용하여 정확한 시간을 계산한다. cpu_time은 그림 3과 같이 cpu를 할당 받은 시간(c_{k-1})과 cpu를 반납한 시간(c_k)의 차이를 사용하며, context를 저장하거나 복귀시키는데 소요되는 시간은 고려하지 않는다.

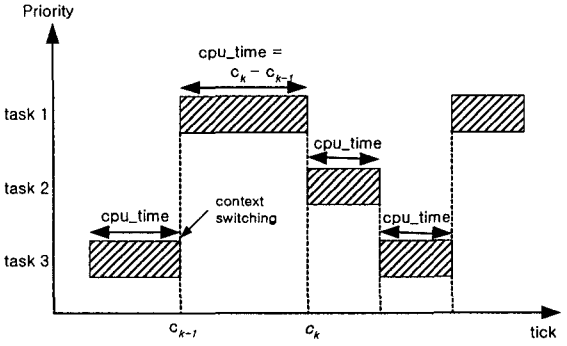

그림 3. cpu_time

3.2 CPU Load Monitor

각 task의 TCB에 저장된 cpu_time 및 switch_count 값을 5초 간격으로 분석한 다음 그 결과값을 Task Entry Data 형태로 변환하여 메시지 큐로 전송한다. Task Entry Data는 {task_name, task_id, priority, cpu_time, switch_count, cpu_usage, accum_cpu_usage}로 구성한다. 현재 n개의 task가 수행중인

경우 cpu_time, cpu 점유율, 누적 cpu 점유율 및 전체 시스템 cpu overload는 식(1)과 같다.

. CPU time (tick): $T[i]_{tick} = c_k - c_{k-1}$

(c_k : Current context switching time,

c_{k-1} : Previous context switching time)

. CPU usage (%): $T[i]_{cpu\_usage} = \frac{T[i]_{tick}}{t_{interval}} \times 100$

($t_{interval}$: Check interval: 5sec)

. Accumulated CPU usage (%):

$$T[i]_{accum\_cpu\_usage} = \sum_{k=1}^{n} \frac{c_k - c_{k-1}}{t_{elapsed\_tick}} \times 100 \qquad (1)$$

. Total CPU load (%): $C_{total\_load} = \sum_{i=1}^{n} T[i]_{accum\_cpu\_usage}$

이렇게 계산된 각 데이터는 Task Entry Data형태로 메시지 큐에 저장한다.

3.3 Abnormal Task Detection

Abnormal task 검출 방안은 각 task별 CPU 점유율과 context switching 횟수가 미리 정의한 임계 값을 연속해서 3회 이상 초과할 경우, 전체 시스템에 overload를 많이 주어 시스템 성능 저하 및 결함 발생 가능성이 높은 task로 분류하여 삭제 및 재시동 시킨다.

3.3.1 CPU 점유율을 이용한 검출

Task Entry Data에서 각 task별 cpu 점유율 및 누적 cpu 점유율을 분석하여 임계 값을 넘는 task가 발견되면, 해당 TCB의 abnormal_count값을 증가 시켜서 기준 값을 초과할 경우 해당 task를 삭제 및 재시동하는 방안이다. Abnormal_count는 각 task별로 TCB내에 별도의 필드를 두어서 그 횟수를 저장하며 기준 값은 3회로 설정하였다. Task가 특정 시점에서 burst한 속성을 지닐 경우 그 특성을 정확하게 파악하지 못하고 삭제 시키는 오류를 피하기 위해 연속 3회 이상 임계 값을 넘는 경우 해당 task를 삭제 및 재시동 한다. 이 방안은 각 task의 정확한 cpu 사용 시간 및 점유율을 구할 수 있어서 cpu 독점 task를 검출하는데 유용하다.

3.3.2 Context Switching 발생 빈도를 이용한 검출

특정 task가 빈번한 context switching을 유발할 경우 cpu가 context를 저장하는데 소요되는 시간 동안 task들이 동작을 멈추게 되고 이로 인해 시스템 overload가 증가한다. 따라서 일정 시간 동안 발생한 전체 context switching 횟수에 대해 일정 비율 이상을 차지하는 task를 abnormal task로 분류한다. 이 방법은 우선순위가 높은 task가 짧은 수행시간을 가질 경우 cpu점유율을 구하기 어려운 문제점을 보완할 수 있다.

두 가지 검출방안을 이용한 abnormal task를 검출하는 과정은 아래와 같으며, 장단점 분석은 표1과 같다.

```
if((T[i]accum_cpu_usage > CPU_THRESHOLD) or
   (T[i]switch_count > TASK_SWITCH_THRESHOLD) then
     T[i]abnormal_count++;
     if(T[i]abnormal_count > 3 )
          Delete & Restart T[i];
else
     T[i]abnormal_count--;
```

표 1. 결함 검출 방안 장단점 비교

	CPU 점유율을 이용한 검출	Context Switching 발생 빈도를 이용한 검출
장점	. task의 cpu 사용 시간을 계산하여 cpu 독점 task를 검출하기에 용이함	. 짧은 수행시간과 높은 우선순위를 가지는 task의 cpu overload를 검사하기가 용이함
단점	. 사용 된 보조 클럭의 정밀도에 따라 cpu 사용 시간에 오차가 발생할 수 있음	. cpu 점유율이 낮지만 빈번한 context switching을 일으키는 task의 경우 heavy load task로 분류를 하는 문제점 발생

4. 결론

본 논문에서는 real-time환경에서 abnormal task를 자동 검출하여 시스템 overload 및 오 동작을 사전에 검출할 수 있는 방안을 제안하였다. 이를 위해 context switching이 발생하는 시점에서 각 task들의 cpu 점유율 및 context switching 횟수를 분석하여 비정상적으로 높은 cpu점유율을 가지는 task와 과도한 context switching을 일으키는 task들을 자동으로 검출하는 방안을 제안하였다. 본 논문에서 제안한 방안은 context를 저장하거나 복귀시키는데 소요되는 overload는 고려하지 않았으며, 주기적이고 지속적인 task의 경우로만 한정하였다. burst한 특성을 가지는 task는 특정 시간동안에만 cpu를 사용한 후 반납하기 때문에 지속적인 cpu 점유율을 측정하기에 어려운 문제점이 있다.

향후 연구에서는 검출된 task를 자동 복구 시켜서 정상적인 동작 수행을 보장할 수 있는 방안 및 복구과정에서 다른 task 및 시스템에 미치는 영향을 최소한으로 줄일 수 있는 연구가 필요하다.

[참고문헌]

[1] Sunondo Ghosh, Rami Melhem, Daniel Mosse, "Fault-Tolerance Through Scheduling of Aperiodic Tasks in Hard-Real-Time Multiprocessor Systems", IEEE Trans. on parallel and distributed systems, vol.8 March, 1997.

[2] Dieter Haban, Kang G. Shin, "Application of Real-Time Monitoring to Scheduling Task with Random Execution Times", IEEE Trans. On software engineering, vol. 16 Dec., 1990.

[3] K. Ramamritham, John A. Stankovic, "Scheduling Algorithms and Operating Systems Support for Real-Time Systems", invited paper, Proceedings of the IEEE, 1994.

[4] John A. Stankovic, "Misconceptions About Real-Time Computing", IEEE Computer, 1988.

[5] A. Silberschatz, P.B. Galvin, G.Gagne, "Operating System Concepts", 6th Edition, Wiley, 2003.

[6] "VxWorks Programmer's Guide 5.4", WindRiver, 1999.