Journal of Digital Contents Society Vol. 9 No. 4 Dec. 2008(pp. 561-568)

비주기 태스크를 고려한 흐름공정 모델의 실시간 스케줄링

문석환,* 김인국**

요 약

이제까지 흐름공정 모델에 관한 연구에서는 주로 주기 태스크들에 대한 스케줄링 방법들을 제시하였지만, 본 논문에서는 합성 이용율을 이용하여 흐름공정 모델에서 비주기적 지역 태스크와 선행 관계를 갖는 비주기적 종단 태스크가 n개의 노드에서 함께 실행되는 경우에 대해서, 스케줄링 가능성 여부를 판단할 수 있는 알고리즘을 제시하였다. 이전에 제시된 여러 단계의 파이프라인에서 실행되는 비주기 종단 태스크 처리방법을 흐름공정 모델에 적용하면, 실제로는 스케줄링이 불가능한 태스크가 스케줄링 가능하다고 판정되고, 이로 인해 실제로는 스케줄링이 가능한 태스크들이 스케줄링 불가능하다고 판정되는 문제가 발생한다. 본 논문에서 제시한 알고리즘은 이러한 문제를 해결하였고, 모의실험을 통해 스케줄링 가능성이 10% 증대되었음을 확인하였다.

Real-Time Scheduling in Flow Shop Model Considering Aperiodic Tasks

Seok-Hwan Moon*, In-Guk Kim**

Abstract

Research on the flow shop model has mainly been centered around periodic tasks scheduling. In this paper, we present an algorithm using synthetic utilization that can check the schedulability of aperiodic local tasks and aperiodic end-to-end tasks with precedence relation in the flow shop model. If the scheduling algorithm for aperiodic end-to-end tasks executed in the multiple stage pipeline is applied to the flow shop model, sometimes the actually schedulable tasks are decided to be not schedulable because of the fact that the actually unschedulable tasks are decided to be schedulable. The algorithm presented in this paper solves the problem, and the simulation shows that the schedulability increases 10%.

Keywords : Real-Time Scheduling, Aperiodic Tasks, Distributed Real-Time System, Flow Shop Model

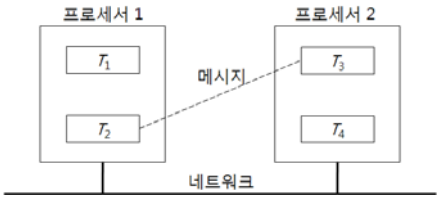
1. 서론

분산 실시간 시스템은 지리적으로 분산되어 있는 시스템들을 네트워크로 연결하여 다수의 프로세서 노드로 구성되며, 분산 실시간 시스템에서 각 프로세서 노드에서 실행되는 태스크들은 지역 태스크(local task)와 종단 태스크(end-to-end task)로 구분할 수 있다.

지역 태스크는 어떤 임의의 노드에서만 실행되는 태스크로서, 다른 노드에 있는 태스크와 메시지 전달에 의한 통신을 하지 않는다. 반면에 종단 태스크는 서로 다른 노드에서 수행되는 서브 태스크들로 구성되며, 메시지 전송을 통해 서브 태스크들 사이의 실행 순서가 정해지는 선행관계를 갖는다.

선행관계란 첫 번째 서브 태스크가 실행 후, 두 번째 서브 태스크 부터는 이전의 태스크가 실행을 완료한 후에 보내는 메시지 도착에 의해서 실행되는 형태를 말한다. (그림 1)은 선행관계를 갖는 분산 실시간시스템의 예이다. (그림 1)에서 보면 T_3은 항상 T_2가 먼저 수행 완료되고, 메시지가 도착한 후에 실행된다.

※ 제일저자(First Author) : 문석환
접수일:2008년 10월 24일, 완료일:2008년 12월 05일
* 영동대학교
shmoon@youngdong.ac.kr
** 단국대학교



(그림 1) 분산 실시간 시스템

주기 태스크인 경우 지역 태스크의 응답시간은 임의의 노드에서 태스크가 완료되는 시점까지 소용되는 시간을 의미하고, 종단 태스크의 응답시간은 첫 번째 서브 태스크가 시작하여 마지막 서브 태스크의 완료시점까지 소요되는 시간을 의미하는데, 이것을 종단 응답시간(end-to-end response time)이라 한다.

분산 실시간 시스템에서 경성종료시한을 갖는 주기적 태스크인 지역 태스크와 종단 태스크들에 관한 스케줄링 방법은 많은 연구가 시행되어 왔다[1, 2, 3, 4]. 고정우선순위기반의 선점형 스케줄링 방식인 RM, DM [5, 6] 방식을 사용 하는 지역 태스크와 종단 태스크들은, 경성종료시한을 가지므로 태스크 스케줄링 알고리즘은 최악의 경우 응답시간이 종료시한보다 항상 작거나 같아야 함을 이용해서 스케줄링 분석을 한다. 이러한 종단 태스크들은 흐름공정(flow shop) 모델[7]에서도 흔히 나타난다.

다중 프로세서 시스템이나 분산 시스템에서 태스크들은 서브태스크들로 나뉘어 하나 이상의 프로세서에서 처리되는 경우가 있다. 이때 각 태스크를 구성하는 서브태스크의 수와 그들이 프로세서들에서 수행되는 순서가 모든 태스크들에 있어서 같은 경우 흐름 공정 문제라 부른다. 이러한 흐름 공정은 다중 프로세서 시스템이나 분산 시스템에서 프로세서들이 기능별로 전담된(functionally dedicated) 경우의 모델로 사용할 수 있다. 흐름 공정 문제에서 프로세서의 수가 2개 이상인 경우의 거의 모든 문제들은 NP-complete 이다[8].

흐름 공정 문제를 구성하는 태스크들이 주기적인 태스크들인 경우 주기적인 흐름 공정 문제라 한다. 예를 들면 어떤 실시간 컨트롤 시스템이 입력 프로세서(input processor), 처리 프로세서(computation processor), 그리고 출력 프로세서(output processor)로 구성되어 있다고 하면 입력 프로세서는 센서들의 값을 입력받고(read), 처리 프로세서는 입력된 센서값을 처리하여 그에 맞는 적절한 명령을 발생한다. 그리고 출력 프로세서는 그 명령을 시스템이 제어하는 구동기(acturator)로 전송하는 일을 하게 된다. 서브태스크들은 반드시 입력 프로세서에서 가장 먼저 실행 완료된 다음에, 처리 프로세서에서 실행되고 마지막으로 출력 프로세서에서 실행된다. 즉, 각 서브 태스크들은 선행관계를 갖는다. 이러한 3개의 프로세서를 이용하여 어떤 작업을 처리하는 시스템은 분산 실시간 시스템과 흐름공정 모델의 기본적인 예로서 설명할 수 있다.

분산 실시간 시스템에서 다루는 태스크들은 이제까지는 주로 주기적인 태스크들에 한정되었다. 또한 Kim등[9]의 흐름공정에서 스케줄링 분석도 주기적 태스크에 관한 문제를 다루고 있다.

본 논문에서는 분산 실시간 시스템에서의 비주기 태스크들이 포함된 흐름공정모델에서의 스케줄링 분석 방법을 제시하고자 한다. 2장에서는 주기 태스크들에 대한 흐름공정 모델에 대해서 기술하고 3장에서는 비주기적 태스크들에 대한 흐름공정 모델에서의 스케줄링 분석방법을 제시한다. 4장에서는 모의실험 결과를 보이고, 마지막으로 5장에서는 결론 및 향후 연구 과제에 대해서 기술한다.

2. 주기적 흐름공정 스케줄링

2.1 주기적 흐름공정 모델

Bettati와 Liu[8]는 주기적 흐름 공정 모델에서의 스케줄링 방법을 제시하였다. 주기적인 흐름공정 모델에서 프로세서의 집합은 $P=\{P_1,P_2,\cdots,P_m\}$, 태스크의 집합은 $\tau=\{\tau_1,\tau_2,\cdots,\tau_n\}$ 이고, 각 태스크는 m개의 서브태스크들로 구성되며, i번째 태스크 τ_i의 서브태스크들은 각각 $\tau_{i,1}$, $\tau_{i,2},\ldots,\tau_{i,m}$으로 표기한다. S_j는 집합 τ내의 모든 태스크들의 j번째 서브태스크들의 집합이다.

하나의 태스크를 구성하는 서브태스크들 사이에는 선행관계가 존재해서, 각 서브태스크 $\tau_{i,j}$는 바로 앞의 서브태스크 $\tau_{i,j-1}$의 실행이 종료되어야만 실행을 시작할 수 있다. i번째 태스크 τ_i의 주기는 T_i, 종료시한은 D_i로 표기하고, 이때

$D_i \leq T_i$라 가정한다. 각 서브태스크 $\tau_{i,j}$의 실행 요구시간(computation requirement)은 $C_{i,j}$로 표시하며, $C_i = C_{i,1} + C_{i,2} + \cdots + C_{i,m}$이라 하자. 태스크 τ_i가 첫 번째 주기에서 실행된 것을 τ_i의 첫 번째 job이라 부르고, j번째 주기에서 실행된 것을 j번째 job이라 부르기로 하자. 태스크들은 주기가 작은 것부터 큰 순으로 배열되어 τ_n의 주기가 가장 큰 것으로 가정한다.

단일 프로세서 환경에서 RM 스케줄링 알고리즘을 사용하는 경우, 태스크 집합의 스케줄링 가능성은 프로세서 이용율(utilization) U가 $n(2^{1/n}-1)$보다 작거나 같으면 확인되며[5], 이에 의해 확인되지 못하는 경우는 시점 0에서 동시에 시작된 각 태스크의 첫 번째 job의 응답시간(response time)을 구하여, 이 값이 종료시한보다 작은가의 여부로 확인할 수 있다[10].

그러나 주기적인 흐름 공정 모델의 환경에서는 Lehoczky등[10]이 제시한 방법이 적용 불가능한 경우가 발생한다. 주기적인 흐름 공정 문제의 경우 각 프로세서 P_j에서 S_j가 RM 스케줄링 알고리즘을 이용하여 독립적으로 스케줄링이 가능하다고 해서, 전체가 RM 스케줄링 알고리즘을 이용하여 스케줄링이 가능한 것은 아니다. 또한 주기적인 흐름 공정 문제에서 RM 스케줄링 알고리즘을 이용하는 경우, 임계 순간에서 시작된 각 태스크의 응답시간은 최악의 경우의 응답시간이 아닐 수도 있다.

Bettati와 Liu[8]는 각 프로세서 P_j에서 실행되는 S_j에 속한 모든 서브태스크들에 대해 $wrt_{i,j} \leq \triangle T_i$를 만족하는 $\triangle$를 구하여 위상을 조정함으로서, 각 프로세서에서 독립적으로 스케줄링 가능성 검사 및 스케줄의 구성이 이루어지도록 하였다. 그러나 이 경우 각 태스크의 서브태스크의 위상이 일률적인 값을 이용해서 조정되므로, 스케줄링이 가능하다고 판정되는 태스크 집합의 비율이 상당히 낮아지는 현상을 보인다.

2.2 개선된 주기적 흐름공정 모델

Kim등[9]은 Bettati와 Liu의 방식을 개선한 방법을 제안하였다. 주기적인 흐름 공정 문제에서 RM 스케줄링알고리즘이 각 프로세서에서 사용된다고 가정한다. P_j에서 S_j가 독립적으로 스케줄링 될 때 $wrt_{i,j}$를 Lehoczky등[10]의 방법에 의해 계산된 $\tau_{i,j}$의 최악의 경우의 응답시간이라고 하고, $wrt_i = wrt_{i,1} + wrt_{i,2} + \cdots + wrt_{i,m}$이라 하자. $\tau_{i,j}$의 시작 시간은 $\tau_{i,1}$의 경우에는 0이고, 그 외의 경우에는 $wrt_{i,j-1}$로 놓는다. 이때 모든 i에 대해 $wrt_i \leq D_i$ $(1 \leq i \leq n)$이면 주어진 태스크 집합은 스케줄링이 가능하다.

이 사실은 다음과 같이 증명될수 있다. 임의의 태스크 τ_i에 대해 $\tau_{i,1}$이 시점 0에서 시작되었을 때, 어떤 위상하에서도 $wrt_{i,1}$이내에 끝마쳐질 수 있음은 명백하다. 이때 $wrt_{i,1}$은 $\tau_{i,2}$의 시작 시점으로 간주되며, $\tau_{i,2}$는 또한 어떤 위상하에서도 시작 시점으로부터 $wrt_{i,2}$이내에 끝마쳐 질 수 있다. 이러한 추론은 τ_i의 모든 서브태스크들에 대해 적용 가능하며, 따라서 τ_i는 어떠한 위상하에서도 wrt_i이내에 끝마쳐질 수 있다. 그러므로 $wrt_i \leq D_i$이면 τ_i는 스케줄링이 가능하다. 또한 모든 i에 대해 τ_i이면 주어진 태스크 집합은 스케줄링이 가능하다.

3. 비주기적 흐름공정 스케줄링

3.1 합성 이용율

Abdelzaher등[11]에 의해 제시된 합성 이용율 방법은, 경성 비주기 태스크 집합에 대하여 스케줄링 가능한 합성 이용율의 상한을 제시하였다. 또한 이를 이용하여 비주기 태스크가 도착했을 때, 합성 이용율을 상한과 비교하여 도착한 비주기 태스크의 수락여부를 인정할 것인지, 거절할 것인지를 결정하게 된다.

비주기 태스크 집합 T_a를 $\{T_1, T_2, T_3 \cdots T_i \cdots\}$라 할 때, T_1은 T_2보다 먼저 도착(arrival)된 태스크라 가정한다. 즉 T_i는 T_{i+1}보다 먼저 도착한 비주기 태스크이다. 비주기 태스크 T_i의 수행시간(execution time)을 $C_i(>0)$, 도착시간(arrival time)을 A_i, 상대적 종료시간(relative deadline)을 $D_i(>0)$ (여기서 절대적 종료시간(absolute deadline) d_i는 $A_i + D_i$로서 정의된다.)라 할 때 임의의 시점 t에서의 합성 이용율은 다음과 같이 정의된다.

$$U_{(t)} = \sum_{T_i \in S_{(t)}} \frac{C_i}{D_i} \quad (1)$$

식 (1)은 Liu와 Layland가 정의한 주기태스크들의 스케줄링 알고리즘인 RM 스케줄링 알고리즘을 비주기 태스크에 확장하여 정의하였다. 식 (1)에서 $S_{(t)}$는 임의의 시점 t를 기준으로 t이전에 도착한 태스크 중 아직 종료시한이 지나지 않은 태스크 집합을 나타내며, $S_{(t)} = \{T_i | A_i \le t < A_i + D_i\}$ 로 표현된다. 또한 임의의 시점 t에서의 합성 이용율의 상한은 식 (2)와 같이 정의된다.

$$\begin{cases} UB_{(n)} = \frac{1}{2} + \frac{1}{2n} & n < 3 \\ UB_{(n)} = \frac{1}{1 + \sqrt{\frac{1}{2}(1 - \frac{1}{n-1})}} & n \ge 3 \end{cases} \quad (2)$$

여기서 n은 현재요청집합(current invocation)에 속하는 비주기 태스크의 수이며, n이 무한히 증가하게 되면 합성 이용율의 상한은 $\frac{1}{1+\sqrt{1/2}} \approx 0.59$에 수렴하게 된다.

Abdelzaher등[11]은 임의의 시점 t에서의 합성 이용율 $U_{(t)} = \sum_{T_i \in S_{(t)}} \frac{C_i}{D_i}$이 상한인 0.59를 넘지 않으면 비주기 태스크가 스케줄링 가능하다는 것을 증명하였다. 또한 합성 이용율을 이용한 스케줄링 분석은 $O(1)$에 수행할 수 있어 수락제어를 수행할 때 적합하다. 또한 Abdelzaher등[12]은 여러 단계(multiple stage)를 거쳐 실행되고, 종단 종료시한(end-to-end deadline)을 갖는 파이프라인(pipeline)에서의 비주기 태스크 처리에 관한 방법을 제시하였다. Hawkins와 Abdelzaher[13]는 이를 확장하여 고정 우선순위 스케줄링 정책을 이용하여, 분산 실시간 시스템에서 비주기 태스크들의 스케줄링 방법을 제시하였으며, 각 노드에서 비주기 태스크들의 합성 이용율을 다음과 같이 정의하고 있다.

$$U_j = \sum_{T_i \in S_j(t)} \frac{C_{ij}}{D_i} \quad (3)$$

여기서 j는 분산 실시간 시스템에서의 각 노드라고 할 수 있다. $D_{\max}$를 노드에서 우선순위가 제일 높은 태스크의 종료시한이라 할 때, 식 (3)을 이용하여 각 노드에서 비주기 태스크가 소비한 시간을 다음 식 (4)와 같이 정의하였다.

$$F_j = \frac{U_j(1 - \frac{U_j}{2})}{1 - U_j} D_{\max} \quad (4)$$

임의의 비주기 태스크가 실행되는 노드의 수가 N개 이고, 종단응답 시간을 D_n 이라하면 다음 식 (5)를 만족하면 임의의 비주기 태스크는 스케줄링 가능하다.

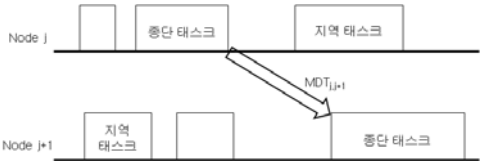
$$\sum_{j=1}^{N} F_j \le D_n \quad (5)$$

본 논문에서는 이러한 특성을 이용하여 분산 실시간 환경에서 비주기적 지역 태스크와 비주기적 종단 태스크들의 흐름공정 모델에서의 스케줄링 방법에 관해 연구하고자 한다.

3.2 합성 이용율을 이용한 비주기적 흐름 공정

본 논문에서의 시스템 모델은 다음과 같다. 먼저 비주기적인 흐름 공정이 m개의 프로세서로 구성되어 있고, n개의 태스크들은 각각 m개의 서브태스크들로 구성된다. 각 태스크의 i번째 서브태스크는 i번째 프로세서에서 수행된다($1 \le i \le m$). 이때 각 태스크의 서브태스크들 사이에는 선행관계가 존재하여 i-1번째 서브태스크의 실행이 종료되어야만 i번째 서브태스크가 실행을 시작할 수 있다.

태스크들은 모두 비주기적이고 각 노드에서의 태스크에게는 종료시한 D_m이 주어지며, 태스크의 전체 종료시한D_n은 해당 태스크가 전체노드를 거쳐 실행될 때 적용되는 실행 종료시한이다. 또한 종단 태스크인 경우 노드 j에서 노드 j+1로 전달되는 메시지를 통해 실행되는데, 이때 발생되는 메시지 전송 시간은 $MDT_{j,j+1}$ 이다. 다음 (그림 2)는 비주기적 흐름공정 모델의 예이다.



(그림 2) 비주기적 흐름공정 모델

비주기적인 흐름공정 모델에서는 합성 이용율 방법을 통해서, 각 노드에서 비주기 태스크에 대한 스케줄링 가능성 여부를 판단하게 된다. 임의의 태스크가 지역 태스크라면 합성 이용율을 이용한 식 (3)을 이용하여 스케줄링 가능성 여부를 판단하게 되고, 종단 태스크라면 식 (5)를 이용하여 스케줄링 가능성 여부를 판단하게 된다. 종단 태스크에서는 메시지 전송 시간이 포함되어야 하므로, 비주기 흐름공정 모델에서의 스케줄링 가능성 여부를 판단하는 방법은 다음과 같이 정의할 수 있다.

$$\sum_{j=1}^{N} F_j + MDT_{wc} \le D_n \qquad (6)$$

즉, 비주기적 흐름공정 모델에서의 종단 비주기 태스크들은, 도착한 노드에서 합성 이용율을 이용한 수락제어를 통해 노드에서의 스케줄링 가능성 여부를 판단한 다음, 해당 노드에서 소비한 값 F_j를 계산하여 노드에서의 종료시한D_m과 비교하여, F_j가 D_m보다 작거나 같음을 만족하면 도착한 노드에서는 스케줄링이 가능하다고 판단할 수 있다. 다음 노드에서는 선행 관계에 의해 메시지를 받아 실행되는 종단형 비주기 태스크들은 이전 노드에서 전달되는 메시지의 지연 시간까지 고려해야한다.

전체 노드에서 발생되는 최악의 경우 메시지 전송시간을 MDT_{wc}라 하면, 각 노드에서의 소비한 값 F_j의합과 MDT_{wc} 값을 합한 값이 임의 태스크의 전체 종료시한보다 작거나 같다면, 임의 태스크는 스케줄링 가능하다고 판단할 수 있다.

<표 1>은 이러한 비주기적 흐름공정 모델에서의 스케줄링 알고리즘이고, (그림 3)은 알고리즘 <표 1>의 블록도이다. <표 2>는 지역태스크와 종단 태스크가 혼합된 비주기 흐름 공정 모델의 예이다. 여기서 프로세서 이용율은 각 노드에서의 지역 태스크와 종단 태스크의 합성 이용율이며, 태스크 T_1, T_2, T_3은 종단 태스크이다. D_1, D_2, D_3은 각 노드에서 종단태스크들의 종료시한이며 이 값은 노드에서 서브 태스크들의 실행시간이 종료되는 시점과 동일하다. 즉, 서브태스크들의 각 노드에서의 종료시한의 합은 전체 종료 시한 D_n과 동일하다. $D_{\max}$는 노드에서 우선순위가 제일 높은 태스크의 종료시한이다.

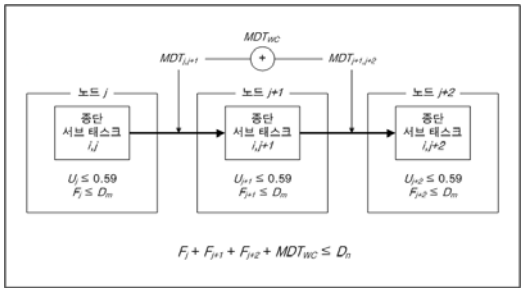
노드 P_1에서 태스크 T_2는 Kim등[9]과 Lehoczky등[10]이 제시한 방법을 적용하면, 각 노드의 F_j값을 합한 F_n 값 23.47이 전체 종료시한 40 보다 작기 때문에 스케줄링 가능하다고 판단되지만, 흐름공정 모델에 적용하면 노드 P_1에서의 F_i의 값 10.14가 종료시한 D_1의 값 10보다 크기 때문에, 스케줄링이 불가능한 것으로 판단될 수 있다.

<표 1> 스케줄링 알고리즘

```
m : 프로세서 개수
n : 태스크 개수
F_j : 노드 j에서 소비한 값

for i=1 to n do
 begin
   F_j = 0
     for j=1 to m do
       begin
           if ( U_j = Σ_{T_i ∈ S_j(t)} C_ij / D_i ≤ 0.59) then
             print ("accept")
           else print ("reject") exit
           end if

       if ( F_i,j = U_j(1 − U_j/2) / (1 − U_j) D_max ≤ D_m ) then
             print ("schedulable at node j")
             F_j = F_j + F_i,j
       else print ("not schedulable at node j")
           end if
       end
       if F_j + MDT_wc ≤ D_n
         print ("schedulable")
       else print ("not schedulable")
       end if
 end
```



(그림 3) 알고리즘 블럭도

<표 2> 비주기적 흐름공정 모델의 스케줄링

		T_1	T_2	T_3
$P1$	U_1	0.45	0.59	0.32
	D_1	8	10	7
	D_{max}	3	10	5
	F_1	1.90	10.14	1.97
$P2$	U_2	0.52	0.43	0.35
	D_2	8	10	15
	D_{max}	6	9	15
	F_2	4.81	5.32	6.66
$P3$	U_3	0.38	0.4	0.57
	D_3	8	20	13
	D_{max}	5	15	13
	F_3	2.48	8.0	12.32
F_n		9.19	23.47	20.96
$MDTwc$		2	2	2
D_n		24	40	35

Abdelzaher등[12][13]의 방법을 이용하면 스케줄링 가능한 태스크 수는 늘어날 수 있지만, 각 노드에서 종료시한을 넘기는 태스크들에 대한 처리를 하지 못하게 되는 경우가 발생하게 되며, 스케줄링 가능한 다른 지역 태스크나 종단 태스크들에게 영향을 미치게 된다. 다음 장에서는 <표 1>의 알고리즘을 이용한 비주기 흐름공정 모델에서 모의실험을 통해 비주기 종단 태스크들의 스케줄링 가능성 여부를 판단해 본다.

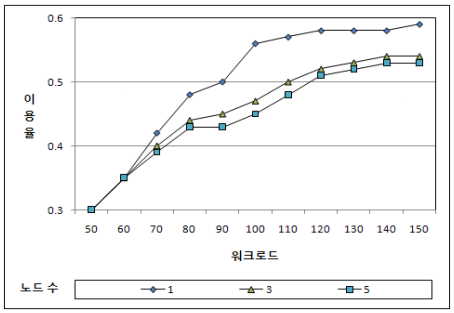
4. 실험 및 결과

4.1 모의실험 방법

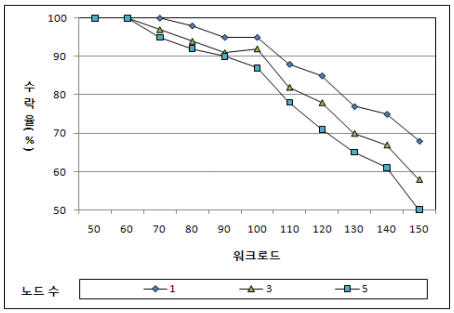
모의실험에서는 비주기적인 지역태스크와 종단태스크로 나누어 1000개를 랜덤하게 생성하고, 노드의 개수에 따른 합성 이용율 변화와 1~5개의 노드를 거쳐 실행되는 태스크들의 워크로드 변화에 따른 스케줄링 비율을 보여준다. 입력 워크로드는 모든 도착된 태스크들의 수행시간의 합과 태스크들이 도착한 시간구간의 비율로 표현할 수 있는데, 실험 범위는 50% ~ 150%까지 10%씩 증가되는 시점에서 이용율을 측정하였다.

4.2 실험 결과

(그림 4)는 비주기 흐름 공정 모델에서 노드의 수에 따른 이용율의 변화를 나타낸 것이다. 각 노드에는 지역 태스크와 종단 태스크가 함께 처리된다. 노드의 수가 많아질수록 이용율이 조금씩 낮아지는데, 그 이유는 노드에서 순차적으로 처리되는 종단태스크들이 노드에서 수행될 때마다 실행 시간이 줄어들기 때문이다. 하지만 본 실험에서는 지역 태스크도 고려하였기 때문에 노드의 수에 따른 이용율이 큰 차이를 보이지는 않는다. 또한 단일 노드에서와 마찬가지로 워크로드가 증가하면서 이용율이 증가하는 것을 볼 수 있다.

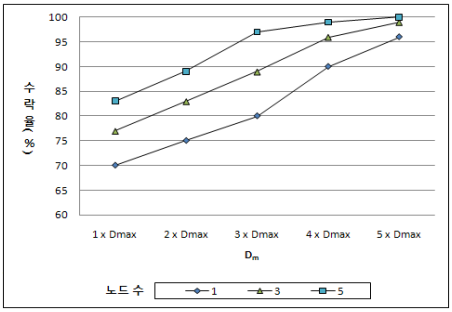


(그림 4) 비주기적 흐름공정 모델의 이용율



(그림 5) 워크로드별 태스크 수락 비율

(그림 5)에서는 각 노드에 도착하여 수락되는 비주기 태스크들의 워크로드별 비율을 보여주고 있다. 비주기 태스크의 합성이용율을 이용한 수락제어는 이용율이 높아지게 되면, 입력되는 비주기 태스크들의 수락율이 그만큼 낮아지게 되는데, 노드의 수가 증가되더라도 입력 워크로드가 증가하게 되면 수락율은 낮아지게 된다.



(그림 6) 비주기적 흐름공정 모델의 태스크 수락 비율

(그림 6)은 각 노드에서 비주기 태스크의 소비되는 시간 F_j를 구할 때 $D_{\max}$에 따른 노드에서의 태스크 수락율을 보여준다. <표 2>에서 볼 수 있듯이 $D_{\max}$는 노드에서 우선순위가 제일 높은 태스크의 종료시한을 의미한다. 우선순위가 높다는 의미는 종료시한이 짧다는 것을 의미 하며, 이것은 F_j의 값에 영향을 미치게 된다. 비주기 흐름공정 태스크의 임의의 노드에서 종료시한 $D_m = D_{\max}$인 경우, 해당 노드에서의 종료시한을 넘기게 되어 스케줄링이 불가능한 경우가 발생될 수 있다. D_m과 $D_{\max}$가 3배의 차이를 갖는다면 약 10%정도의 수락율이 증가하게 된다. 마찬가지로 5배의 값을 갖는 D_m이 존재하게 되면 수락율은 더욱 증가하게 된다. 단일 노드에서 비주기 태스크의 합성이용율은 실행 시간이 짧고, 종료시한이 커지면 프로세서 이용율이 하락하여 수락율이 높아지게 되는데, 마찬가지로 비주기 적 흐름공정 모델의 노드에서 소비되는 시간을 구하는 경우에도, 종료시한이 커지면 수락율은 증가하게 된다. 또한 노드의 수가 증가할수록 수락율도 증가하게 된다.

5. 결론

본 논문에서 제시한 비주기적 흐름공정 모델에서의 스케줄링 방법은, 합성 이용율을 이용하여 노드에 도착한 서브태스크의 소비되는 시간을 계산하여 스케줄링 가능한지를 판단하는 방법이다. 각 노드에서 소비되는 시간은 $D_{\max}$의 값에 비례하여 변하는데, 각 노드에서 지역 태스크들과 종단 태스크들이 함께 존재할 때, 선행관계를 갖는 종단 태스크들의 종료시한은 흐름공정의 특성 상, 실행 시간이 끝나는 시점이 종료시한과 같게 된다. D_m과 $D_{\max}$가 3배의 차이를 갖는다면 태스크 스케줄링 가능성이 약 10% 정도 증가하게 된다. 즉, 태스크들의 종료시한이 클수록, 노드의 수가 많을수록 종단 태스크들의 스케줄링 가능성은 높아지게 된다.

향후 연구로는 메시지 전송 시간이 가변적인 경우와, 주기 태스크와 비주기 태스크가 혼합된 형태의 흐름공정 모델에서의 스케줄링 방법에 대한 연구가 필요하다.

참 고 문 헌

[1] J. J. G. Garcia and M. G. Harbour, "Optimized Priority Assignment for Tasks and Message in Distributed Hard Real-Time Systems," Proc. of Workshop on Parallel and Distributed Real-Time Systems, pp. 124-132, 1995.

[2] D. T. Peng, K. G. Shin, and T. F. Abdelzaher, "Assignment and Scheduling Communicating Periodic Tasks in Distributed Real-Time Systems," IEEE Transactions on Software Engineering, Vol. 23, No. 12, pp. 745-758, 1997.

[3] K. W. Tindell, A. Burns, and A. J. Wellings, "Allocating Real-Time Tasks. An NP-Hard Problem Made Easy," Real-Time Systems, Vol. 4, No. 2, pp. 145-166, 1992

[4] J. Sun and J. Liu, "Synchronization Protocols in Distributed Real-Time Systems," Proc. of International Conference on Distributed Computing Systems, pp. 38-45, 1996.

[5] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in hard real time environment", Journal of the ACM, Vol. 20, pp. 46-61, Jan. 1973.

[6] N. Audsley, A. Burns, M. Richardson, and A. J. Wellings, "Hard Real-Time Scheduling: The Deadline Monotonic Apporoach," Proc. of IEEE Workshop on Real-Time Operating Systems and Software, pp. 133-137, 1991.

[7] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, "Sequencing and Scheduling: Algorithms and Complexity," Centre for Mathmatics and Computer Science, Amsterdam, 1989.

[8] R. Bettati and J. W. S. Liu, "Algorithms for end-to-end scheduling to meet deadlines," Proc. of the 2nd Conf. on Parallel and Distributed processing, Dec. 1990.

[9] I. G. Kim, K. H. Choi, S. K. Park, D. Y. Kim, M. P. Hong, and M. J. Lee, "Scheduling in periodic flow shop using worst-case response time," proc. of 1st International Workshop on Real-Time Computing Systems and Applications, pp. 9-16, Dec. 1994.

[10] J. P. Lehoczky, L. Sha and Y. Ding, "The rate monotonic scheduling algorithm: exact characterization and average case behavior," Proceedings of the 10th IEEE Real Time Systems Symposium, Dec. 1989.

[11] T. F. Abdelzaher, V. Sharma, and C. Lu, "A Utilization bound for aperiodic tasks and priority driven scheduling", IEEE Transactions on Computers, Vol. 53, No. 3, pp. 334-350, Mar. 2004

[12] T. F. Abdelzaher, G. Thaker, and P. Lardieri, "A feasible region for meeting aperiodic end-to-end deadlines in resource pipelines," International Conference on Distributed Computing Systems, Tokyo, Japan, March. 2004

[13] W. Hawkins, and T. F. Abdelzaher, "Towards Feasible Region Calculus: An End-to-end Schedulability Analysis of Real-Time Multistage Execution," Proceedings of the 26th IEEE Real Time Systems Symposium, Dec. 2005.

문 석 환

2001년 : 단국대학교대학원 전자계산학과(이학석사)
2006년 : 단국대학교대학원 전자계산학과(박사 수료)
2006년～현 재 : 영동대학교 임베디드소프트웨어학과 전임강사
관심분야 : 운영체제, 실시간시스템, 임베디드시스템

김 인 국

1982년 : 단국대학교(학사)
1985년 : 미국 에모리대학교(석사)
1995년 : 아주대학교(박사)
1986년～현재 : 단국대학교 컴퓨터학부 컴퓨터과학전공 교수
2001년～2003년 : 미국 뉴멕시코공과대학 방문교수
관심분야 : 운영체제, 실시간시스템, 임베디드시스템