실시간 감시를 위한 학습기반 수행 예측모델의 검증

정 윤 석[†] · 김 태 완[††] · 장 천 현[†††]

요 약

실시간 시스템은 시스템이 적시성을 보장하는지 파악하기 위해 실시간 감시기법을 이용한다. 일반적으로 실시간 감시는 실시간 시스템의 현재 동작상태를 파악하는데 중점을 두는 기법이다. 그러나 실시간 시스템의 안정적인 수행을 지원하기 위해서는, 현재 상태를 파악하는 것뿐 아니라, 실시간 시스템 및 시스템상에서 동작하는 실시간 프로세스들의 수행도 예측할 수 있어야 한다. 그러나 기존 예측모델을 실시간 감시기법에 적용하기에는 몇 가지 한계가 있다. 첫째, 예측기능은 실시간 프로세스가 종료한 시점에서 정적인 분석을 통해 수행된다. 둘째, 예측을 위해 사전 기초 통계분석이 필요하다. 셋째, 예측을 위한 이전확률 및 클러스터 정보가 현재 시점을 정확하게 반영하지 못한다. 본 논문에서는 이러한 문제점들을 해결하고 실시간 감시기법에 적용할 수 있는 학습 기반의 수행 예측모델을 제안한다. 이 모델은 학습기법을 통해 불필요한 전처리 과정을 없애고, 현시점의 데이터를 이용해, 보다 정확한 실시간 프로세스의 수행 예측이 가능하도록 한다. 또한 이 모델은 실시간 프로세스 수행 시간의 증가율 분석을 통해 다단계 예측을 지원하며, 무엇보다 실시간 프로세스가 실행되는 동안 예측이 가능한 동적 예측을 지원하도록 설계하였다. 실험 결과를 통해 훈련집합의 크기가 10 이상이면 80% 이상의 판단 정확도를 보이며, 다단계 예측의 경우, 훈련집합의 크기 이상의 수행 횟수를 넘으면 다단계 예측의 예측 차는 최소화되는 것으로 나타났다. 본 논문에서 제안한 예측모델은 가장 단순한 학습 알고리즘을 적용했다는 점과, CPU, 메모리, 입출력 데이터를 다루는 다차원 자원공간 모델을 고려하지 못한 한계가 있어 향후에 관련 연구가 요구된다. 본 논문에서 제안하는 학습기반 수행 예측모델은 실시간 감시 및 제어를 필요로 하는 분야 및 응용 분야에 적용할 수 있다.

Verifying Execution Prediction Model based on Learning Algorithm for Real-time Monitoring

Yoon Seok Jeong[†] · Tae Wan Kim[††] · Chun Hyon Chang[†††]

ABSTRACT

Monitoring is used to see if a real-time system provides a service on time. Generally, monitoring for real-time focuses on investigating the current status of a real-time system. To support a stable performance of a real-time system, it should have not only a function to see the current status of real-time process but also a function to predict executions of real-time processes, however. The legacy prediction model has some limitation to apply it to a real-time monitoring. First, it performs a static prediction after a real-time process finished. Second, it needs a statistical pre-analysis before a prediction. Third, transition probability and data about clustering is not based on the current data. We propose the execution prediction model based on learning algorithm to solve these problems and apply it to real-time monitoring. This model gets rid of unnecessary pre-processing and supports a precise prediction based on current data. In addition, this supports multi-level prediction by a trend analysis of past execution data. Most of all, We designed the model to support dynamic prediction which is performed within a real-time process' execution. The results from some experiments show that the judgment accuracy is greater than 80% if the size of a training set is set to over 10, and, in the case of the multi-level prediction, that the prediction difference of the multi-level prediction is minimized if the number of execution is bigger than the size of a training set. The execution prediction model proposed in this model has some limitationthat the model used the most simplest learning algorithm and that it didn't consider the multi-regional space model managing CPU, memory and I/O data. The execution prediction model based on a learning algorithm proposed in this paper is used in some areas related to real-time monitoring and control.

키워드 : 실시간 감시(Real-time Monitoring), 실행시간 프로세스 모니터(Run-time Process Monitor), 학습 기반 수행 예측모델(Execution Prediction Model based on Learning Algorithm)

1. 서 론

여러 분야에서 실시간 시스템의 요구가 늘어나면서 시스템의 동작상태를 파악할 수 있는 실시간 감시기능이 요구되고 있다. 이에 따라 실시간 감시와 관련된 많은 연구가 수행되었으며, 주로 실시간 프로세스의 현재 동작상태를 감시하는 것에 초점을 맞추고 있다. 그러나 실시간 시스템의 안정적인 수행을 지원하기 위해서는 현재 상태를 파악하는 것뿐 아니라, 실시간 시스템 및 시스템 상에서 동작하는 실시간

† 준 회 원 : 건국대학교 대학원 컴퓨터공학과
†† 정 회 원 : 건국대학교 대학원 컴퓨터공학과
††† 종신회원 : 건국대학교 컴퓨터공학과 교수
논문접수 : 2004년 3월 17일, 심사완료 : 2004년 8월 6일

프로세스들의 움직임을 예측할 수 있어야 한다.

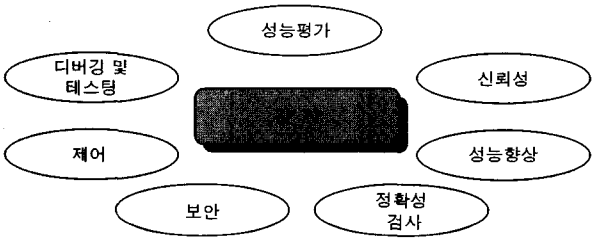
이를 위해 본 논문에서는 학습기반 수행 예측모델을 제안한다. 이 모델은 학습기법을 통해 불필요한 전처리 과정을 줄이고, 현재 데이터에 근거해 보다 정확한 예측을 지원한다. 또한 이 모델은 실시간 프로세스 실행 데이터의 증가율분석을 통한 다단계 예측을 지원하며, 실시간 프로세스가 실행되는 동안 예측이 가능한 동적 예측을 지원하도록 설계하였다.

본 논문의 2장에서는 기존 실시간 감시 및 예측 기능을 분석하여 문제점을 도출하였고, 3장에서는 본 논문에서 제안한 학습기반 수행 예측모델을 상세하게 설명하였다. 4장에서는 학습기반 수행 예측모델을 적용한 실행시간 프로세스 모니터의 구조를 제시하였다. 그리고 5장에서는 본 논문에서 제시한 학습기반 수행 예측모델을 검증하기 위한 실험 및 분석을 수행하였고, 향후 수행 예측모델을 적용하기 위한 기준을 제공하였다. 6장과 7장에서는 각각 논문의 한계점, 논문의 성과와 향후 연구 방향에 대해서 기술하였다.

2. 관련 연구

2.1 기존 감시기능

실시간 시스템이 산업 전반에 보급되면서 실시간 시스템을 감시하는 실시간 감시 기능에 대한 다수의 연구가 수행되었다. 예를 들어, 프로세스 차원의 감시, 시스템 차원의 감시, 하드웨어 차원의 감시, 네트워크 차원의 감시에 관한 연구가 이루어졌다[2-5, 9-11]. 이들 연구를 종합해 볼 때 감시란 '데이터를 수집하고, 이를 통해 각각의 어플리케이션 및 운영체제가 시간 별로 어떤 작업을 수행했는지를 파악하는 것'이며, 감시의 사용 목적 별로 (그림 1)과 같이 7가지로 나눌 수 있다[11].



(그림 1) 감시의 기본 사용 목적

감시 기능은 모니터가 담당한다. 모니터는 시스템의 상태를 감시하고, 이를 분석함으로 비정상적 행위를 찾아내고 이를 해결하는 소프트웨어이다[11]. 실시간 시스템에서 사용하는 모니터는 기본적으로 실시간 프로세스의 수행을 감시하도록 설계된다[2-5, 9, 12].

기존 실시간 감시는 모니터를 통해 실시간 프로세스의 현재 혹은 과거 상태를 파악하고, 수행분석, 정확성 검사 등 간단한 분석을 수행하는 것에 초점을 맞추고 있다[1-5].

2.2 기존 예측기능

예측과 관련된 몇 가지 연구 중에서, [8, 15]는 자원의 사용량을 예측할 수 있는 통계학적 방법에 기초한 자원 예측모델을 제시하고 있다. 자원 예측과정을 살펴보면, 사전에 시스템에서 동작하는 모든 프로세스에 대한 기초 통계분석을 수행하여 이를 근거로 클러스터링 분석을 수행한다. 그 후, 이전확률을 구하고, 특정 프로세스에 대한 상태-이전 모델을 작성한다. 상태-이전 모델과 특정 프로세스의 최근 실행값을 결합하여 프로세스의 향후 수행결과를 예측하게 된다.

이 모델은 프로세스의 자원소모량을 예측할 수 있는 체계를 제공해 주지만, 실시간 감시기법에 적용하기에는 몇 가지 한계가 있다. 먼저, 시스템 상의 모든 프로세스를 대상으로 기초 통계분석이라는 전처리 과정을 수행해야 하는 것과 통계 분석이 수행된 시점의 데이터를 기초로 예측을 수행하기 때문에 데이터 일관성이 결여된다는 문제점이 있다. 무엇보다도 실행시간에 실시간 프로세스를 분석할 수 있는 동적분석 체계를 제공하지 못하는 한계점을 가지고 있다.

2.3 문제점

기존에 수행된 감시기법과 예측에 관한 연구내용을 분석한 결과, 다음과 같은 한계를 지니고 있는 것으로 나타났다.

- 감시기능의 경우,
  - 실시간 프로세스의 현재 수행상태를 파악할 수 있지만, 실시간 프로세스가 어떻게 동작할 것인지는 예측할 수 없다.

- 예측기능의 경우,
  - 예측을 위해 모든 프로세스에 대한 기초 통계분석이라는 전처리 과정이 필요하다.
  - 기초 통계분석 시점과 예측 시점의 차이로 인해 데이터 일관성이 결여된다.
  - 기초 통계분석을 거치지 않고 처음 수행되는 실시간 프로세스에 대해서는 예측이 어렵다.
  - 다음 단계의 예측만을 지원하며, 다단계 예측을 지원하지 못한다.

이러한 문제점들을 해결하기 위해, 본 논문에서는 실행시간에 동적인 예측을 수행할 수 있는 학습기반 수행 예측모델을 제안한다.

3. 학습기반 수행 예측모델의 제안

본 장에서는 2.3절에서 파악한 문제점들을 해결하기 위해 학습 알고리즘과 단순 상태-이전 모델에 기반하여 실시간 프로세스의 동작을 예측하는 학습기반 수행 예측모델을 제안한다.

3.1 실시간을 지원하는 학습 알고리즘

기존의 예측 모델은 예측을 수행하기 위해 사전 처리 과

정을 거치는데, 사전 처리시점과 실제 예측하는 시점이 달라서, 현재 상황을 반영한 예측이 어렵다. 본 논문에서는 실시간을 지원하는 학습 알고리즘을 이용해 이러한 문제점을 해결하고, 최신 데이터에 기초한 예측을 지원한다.

실시간을 지원하는 학습 알고리즘은 기존의 학습 알고리즘과 달리, 다음의 두 가지 사항을 고려한다. 첫째, 실시간 프로세스의 상태 데이터를 기본 데이터로 사용한다. 실시간 프로세스의 상태를 예측하는데 있어, 주요 데이터인 실시간 프로세스의 실행 결과를 이용해 학습 과정을 수행해야 한다. 둘째, 훈련집합을 실시간으로 갱신한다. 정확한 예측을 위해서는 예측 당시의 프로세스 상태를 반영해야 한다. 이를 위해 실시간 프로세스 모니터가 추출하여 데이터 저장소에 저장한 상태 데이터를 지속적으로 훈련집합에 추가하여, 최신의 데이터를 유지해야 한다. 이들 두 가지 사항을 고려하며, 다음의 과정을 통해 학습을 수행한다.

먼저, 실시간 프로세스상태 데이터는 <표 1>과 같이 하나의 인스턴스이며, 훈련집합의 구성요소가 된다. 각 인스턴스는 크게 3개의 속성을 갖도록 설계하였다. Instance SN, Exec. Time 그리고 Classification은 인스턴스의 일련번호, 실시간 프로세스의 수행시간, 학습을 통해 도출한 판단 결과를 의미한다.

〈표 1〉 인스턴스 구조 예

Instance SN	Exec. Time	Classification
1	880	Safe
2	1,200	Unsafe

다음으로 전처리 과정을 통해 각 인스턴스의 속성 값을 정규화한다. 인스턴스의 속성 값을 정규화 하는 이유는 전체 인스턴스 중 몇 개의 인스턴스 속성 값이 결손되어도, 정확한 판단 결과를 도출하기 위해서이다. a_{max} 와 a_{min} 이 아직 처리되지 않은 a속성의 최대값과 최소값이라 할 때, 인스턴스 x의 속성 값 a를 정규화한 값 n_a는

$$n_a = \frac{x_a - a_{min}}{a_{max} - a_{min}}$$

이다. <표 2>는 <표 1>의 인스턴스를 정규화한 결과를 보여준다.

〈표 2〉 정규화된 인스턴스 구조 예

Instance SN	CPU Time	Classification
1	0	Safe
2	1	Unsafe

정규화된 데이터는 유사도를 결정하기 위해 이용된다. 훈련집합에 새로운 인스턴스가 추가되면 기존의 인스턴스들의 속성 값과 어느 정도 유사한지를 계산한다. 그 중 유사도가 가장 높은 인스턴스의 판단 결과를 취하여 자신의 판단 값으로 삼는다. 유사도는 두 개의 인스턴스 x, y의 유클리드 거리의 역으로 나타낸다. P가 모든 속성의 집합이라고 할 때, 유사도 S_i는

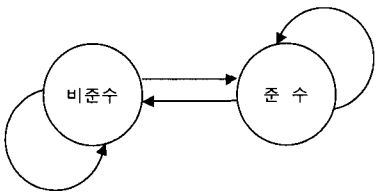
$$s_i = \frac{1}{\sqrt{\sum_{i=1}^{|P|}(x_i - y_i)}}$$

이다. 새롭게 입력된 데이터는 유사도 값과 비교하여 유사도가 더 높은 데이터에 가까운 것으로 판단한다. 판단 결과는 예측을 위해 사용된다.

3.2 단순 상태-이전 모델을 이용한 예측

[15]에서 제안한 상태-이전모델의 경우, 다수 프로세스에 대한 기초 통계분석 등의 전처리 과정을 거쳐야 하는 문제점과, 대상 프로세스와는 관계없는 기타 프로세스를 분석하고, 상태-이전 모델을 구성해야 하는 비효율성이 발생한다. 본 논문에서는 이러한 문제점을 해결하기 위해 단순 상태-이전 모델(Simple State-Transition Model)을 제안한다.

단순 상태-이전 모델은 실시간 프로세스가 취할 수 있는 가장 단순한 상태 모델로서 (그림 2)와 같다. 실시간 프로세스의 상태-이전 모델을 '준수'와 '비준수'의 상태로 나누는 이유는 기타의 프로세스와 달리 실시간 프로세스가 시간 제약조건 준수 여부에 초점을 맞추기 때문이다. 다시 말해, 실시간 프로세스는 시간 제약조건을 준수하고 정상적으로 동작하는 상태, 시간 제약조건을 어기고 비정상적으로 동작하는 상태, 두 가지로 나눌 수 있다. (그림 2)와 같이 실시간 프로세스가 '준수'와 '비준수'의 상태를 취하면, 상태이전이 일어나는 경우는, 비준수 → 준수, 준수 → 비준수, 준수 → 준수, 비준수 → 비준수의 네 가지 뿐이다.



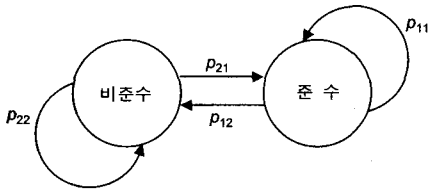
(그림 2) 단순 상태-이전 모델

상태를 단순화하면, 프로세스에 대한 사전 기초 통계분석 수행, 클러스터링, 상태-이전 모델 구성 등의 전처리 과정을 제거할 수 있다. 또한 기존에 수행된 적이 없는 프로세스에 대해서도 동일한 상태-이전 모델을 적용할 수 있다.

단순 상태-이전 모델에 근거해 실시간 프로세스의 상태 이전확률 계산을 위해서는 다음의 이전확률 공식을 사용한다. 이전확률 p_{ij}는

$$p_{ij} = \frac{\text{상태 } i\text{에서 상태 } j\text{로 이전한 횟수}}{\text{상태 } i\text{에서 이전한 횟수}}$$

이전확률 공식에 근거하여 계산된 이전확률은 (그림 3)과 같이 단순 상태-이전 모델과 결합하여 <표 3>과 같이 파악할 수 있다.



(그림 3) 이전확률을 적용한 단순 상태-이전 모델

〈표 3〉 이전확률 파악표

	준 수	비 준 수
준수	p_{11}	p_{12}
비준수	p_{21}	p_{22}

이전확률을 구한 다음, 프로세스의 수행을 예측한다. 이전확률이 $p_{lj}(j=1$ 또는 2, l = 이전 실행 시 속했던 상태)이고, 중심 값이 $d_j(j=1$ 또는 2)일 때, 예측 값 r을 도출하기 위한 공식은

$$r = \sum_{j=1}^{N} p_{lj} d_j$$

이다. 중심 값 d_j를 구하기 위해서는 학습모듈을 통해 도출한 판단결과를 이용한다. 판단 결과는 아래의 중심 값 공식을 통해 도출한다. 중심 값 $d_j(j=1$ 또는 2), 목적 속성 값 x_{ti}(t는 목적속성, $i=1,\cdots,T_1$), 목적속성이 'Safe'인 인스턴스의 개수 N_s, 목적속성이 'Unsafe'인 인스턴스의 개수가 N_u일 때, 중심 값을 구하는 공식은

$$d_j = \begin{cases} \frac{1}{n_s} \sum_{i=1}^{N_s} x_{ti} & \text{if } x_{ti}=0 \\ \frac{1}{n_u} \sum_{i=1}^{N_u} x_{ti} & \text{if } x_{ti}=1 \end{cases}$$

이다.

예측 값으로 실시간 프로세스가 향후 어떻게 동작을 할지 예측할 수 있다. 그러나 [8]과 같이 소프트웨어는 소프트웨어 노화현상으로 인한 자원소모량 증가 현상이 발생하므로, 일 단계 예측보다 실행 추이를 고려한 다단계 예측을 통해 보다 의미 있는 예측 정보를 도출할 수 있다.

이를 위해 본 논문에서는 다단계 예측 기법을 제안한다. 다단계 예측을 위해 현재 훈련집합에 저장되어 있는 인스턴스들의 증가율을 파악해야 한다. 평균 증가율 u는 훈련집합 내의 인스턴스 $s_i(i=1, 2,\cdots, N)$간의 증가율을 구하고, 이를 훈련집합의 인스턴스 개수인 T_1으로 나누어 구할 수 있다.

$$u = \frac{1}{T_1} \sum_{i=2}^{T_1} \frac{S_i - S_{i-1}}{S_{i-1}}$$

다단계 예측 값 $m_i(i=1, 2,\cdots, N)$는 예측 값 $r_i(i=1, 2,\cdots, N)$와 각 차원의 평균 증가율 u를 곱한 후, 예측 값을 더해 구할 수 있다. 다단계 예측 값을 구하기 위한 공식은 다음과 같다.

$$m_i = \begin{cases} ur_i + r_i & \text{if } i = 1 \\ um_{i-1} + m_{i-1} & \text{if } i > 1 \end{cases}$$

지금까지 설명한 학습기반 수행 예측모델의 체계를 요약하면 다음과 같다.

- 모수
  T_1 : 훈련집합의 크기
  T_2 : 예측을 위해 필요한 이전실행의 최소 횟수
  T_3 : 다단계 예측을 위한 예측단계
- 변수
  l : 이전 실행 시에 속했던 상태 번호
- 자료구조
  $[p_{ij}]$: 상태 이전 행렬, i=1 또는 2, j=1 또는 2
  $[s_i]$: 훈련집합, $i=1,\cdots,T_1, k$ = CPU
- 계산식

$$d_j = \begin{cases} \frac{1}{n_s} \sum_{i=1}^{N_s} x_{ti} & \text{if } x_{ti}=0 \\ \frac{1}{n_u} \sum_{i=1}^{N_u} x_{ti} & \text{if } x_{ti}=1 \end{cases}$$

$$r = \sum_{j=1}^{N} p_{lj} d_j$$

$$m_i = \begin{cases} ur_i + r_i & \text{if } i = 1 \\ um_{i-1} + m_{i-1} & \text{if } i > 1 \end{cases}$$

4. 학습기반 수행 예측모델을 적용한 실행시간 프로세스 모니터의 설계

4.1 실행시간 프로세스 모니터의 전체 구조

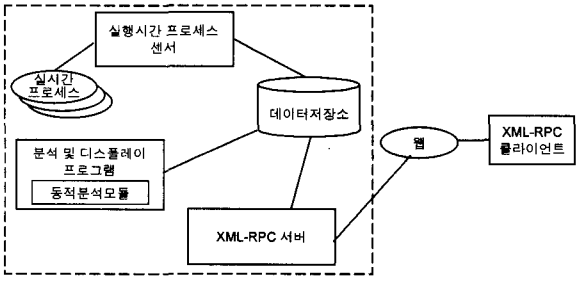
(그림 4)는 학습기반 수행 예측모델을 적용한 실행시간 프로세스 모니터의 전체 구조를 보여준다. 실행시간 프로세스 모니터는 크게 실행시간 프로세스 센서, 데이터 저장소, 분석 및 디스플레이 프로그램으로 구성되며, 각 요소의 기능은 다음과 같다.

- 실행시간 프로세스 센서 : 코드 형태로 실시간 프로세스 내에 삽입되어 실시간 프로세스의 상태 데이터를 수집한다.
- 데이터 저장소 : 실시간 프로세스의 상태 데이터를 저장

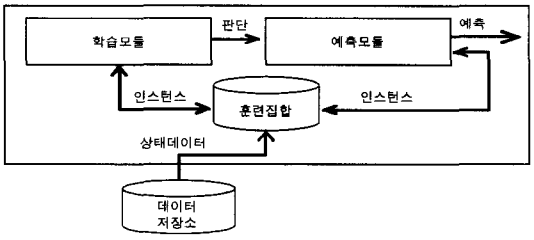
하고 관리한다. 데이터 저장소는 특정 어플리케이션이나 시스템에 대해 독립적이며, 저장된 데이터는 웹 등을 통해 원격에서 접근할 수 있다.

- 분석 및 디스플레이 프로그램 : 데이터 저장소로부터 데이터를 읽거나 분석하여 이를 출력한다.

본 논문에서는 [2-5]에서 제시한 실행시간 프로세스 모니터를 개선하여 분석 및 디스플레이 프로그램 내에 동적 분석 모듈을 추가하고, 예측 기능을 지원할 수 있도록 하였다. (그림 5)는 동적분석 모듈의 세부 구조를 보여준다. 동적분석 모듈은 크게 학습모듈과 예측모듈로 구성된다. 학습모듈은 학습 알고리즘을 이용하여 학습을 하며, 예측모듈은 학습모듈의 판단 결과와 단순 상태-이전 모델을 이용하여 예측한다.

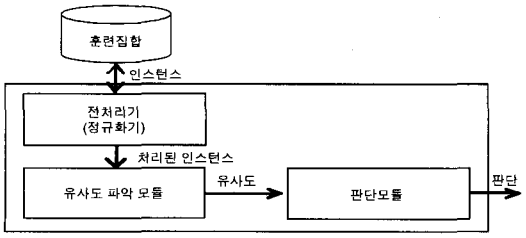


(그림 4) 실행시간 프로세스 모니터의 전체 구조



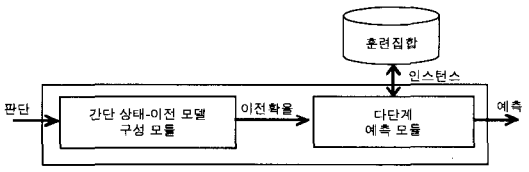
(그림 5) 동적분석 모듈의 구조

학습 모듈의 세부 구조는 (그림 6)과 같다. 실행시간 프로세스 센서는 실시간 프로세스의 상태 데이터를 수집하여 데이터 저장소에 저장한다. 학습모듈은 데이터 저장소에 저장된 프로세스의 상태 데이터를 실시간으로 가져다가 훈련집합의 인스턴스로 구성한다. 그 후, 정규화 및 유사도 파악 과정을 통해 학습 결과를 도출한다.



(그림 6) 학습모듈의 세부구조

(그림 7)은 예측모듈의 세부구조를 보여준다. 학습모듈에서 전달된 판단 데이터와 본 논문에서 제시한 간단 상태-이전 모델을 이용해 이전확률을 계산한다. 필요에 따라 이전 확률 및 기존 수행 결과를 이용해 다단계 예측을 수행한다. 최종적으로 예측모듈은 예측 결과를 출력한다.



(그림 7) 예측모듈의 세부구조

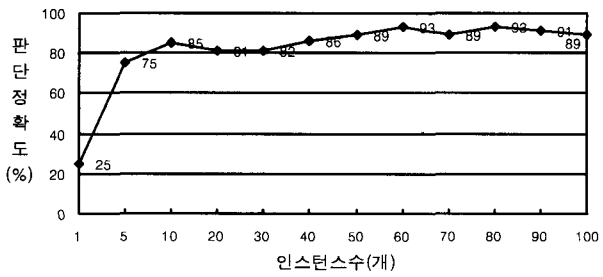
5. 학습기반 수행 예측모델의 실험 및 분석

학습기반 수행 예측모델의 성능을 측정하기 위해 5가지 실험을 수행하였다. 실험을 위해 실시간 모델인 TMO(Time-triggered Message-triggered Object) 모델을 이용하였다. 기본적인 실험 준비과정은 다음과 같다. 먼저, TMO모델에 기반하여 작성한 실시간 프로세스를 동작시키고, 본 논문에서 설계한 실시간 프로세스 모니터를 구현하여 실시간 프로세스 상태 데이터를 수집하였다. 수집한 데이터를 데이터 저장소에 저장한 후, 학습 및 예측의 기본데이터로 제공하였다.

5.1 실험 1 : 학습모듈의 판단정확성 파악 실험

본 실험은 학습모듈의 판단결과가 얼마나 정확한지 측정하는 실험이다. 실험을 위해 훈련집합 크기 T_1을 1에서 100으로 증가시키며 학습을 실시하였다. 학습 후, 학습 시 사용한 인스턴스와 독립적인 400개의 인스턴스를 추가 생성하여 예측모델에 제공하였다. 실험 후, 실제 생성된 데이터와 예측 데이터를 비교하여 판단정확성을 파악하였다.

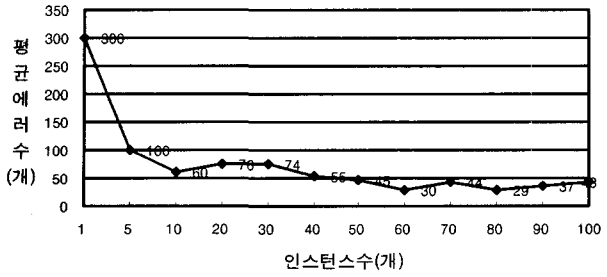
(그림 8)은 본 실험의 결과인 학습모듈의 학습곡선을 보여준다. 실험을 통해 학습에 사용된 인스턴스의 수가 증가할 때, 판단정확도 역시 증가하는 것을 알 수 있다. (그림 8)에서 보듯이, T_1 >= 10인 경우 80% 이상의 판단 정확도를 보이는 것으로 나타났다. 따라서 T_1은 최소 10 이상이어야 하며, 이를 통해 80% 이상의 정확도를 얻을 수 있다.



(그림 8) 학습모듈의 학습곡선

(그림 9)는 T_1을 증가시켰을 때의 평균 에러의 수를 측정한

것이다. 결과를 통해 T_1>=5 일 경우, 평균 에러 수는 100개 미만으로 낮아지는 것을 알 수 있다.



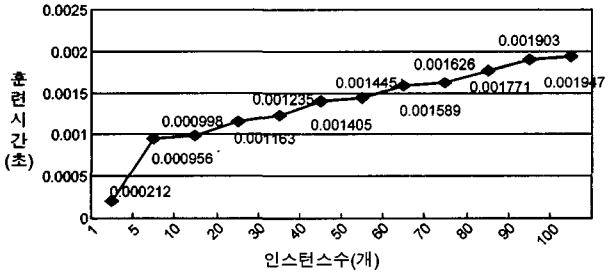
(그림 9) 평균 에러 그래프

5.2 실험 2 : 학습시간 파악 실험

본 실험은 학습 모듈이 학습하는데 어느 정도 시간이 걸리는지를 파악하기 위한 실험이다. 실시간 시스템에서는 처리시간에 따라 시스템의 성능이 결정되므로, 실시간 시스템 혹은 실시간 프로세스에 영향을 미칠 수 있는 훈련 시간을 파악할 필요가 있다.

실험을 위해 T_1을 1에서 100까지 증가시키며, 소요 시간을 측정하였다. 시간 측정은 학습이 시작되는 시점과 학습이 종료되는 시점을 측정하여 계산하였다.

(그림 9)는 학습을 위해 필요한 시간을 보여준다. 결과를 통해 인스턴스 수가 증가함에 따라 훈련시간도 증가하는 것을 알 수 있으며, 학습모듈의 판단 정확도가 80% 이상이기 위해서는 최소 0.001초 이상이 소요됨을 알 수 있다.



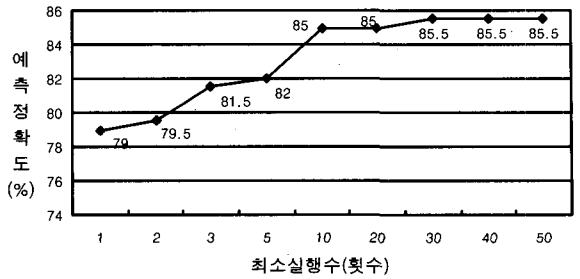
(그림 10) 학습시간 그래프

5.3 실험 3 : 최소 실행 수에 따른 예측정확도 파악 실험

본 실험은 학습기반 수행 예측모델에 기초해 예측을 수행할 때, 실시간 프로세스의 최소 실행 수를 얼마로 해야 하는지를 파악하기 위해 실시하였다.

실험 1에 근거해 86% 정확도를 보이도록 T_1=50으로 설정하고, 최근의 실시간 프로세스 실행 수 T_2를 1회에서 50회로 증가시켰다. 또한 학습 시 사용한 데이터와는 독립적인 실시간 프로세스 상태 데이터를 제공하여 학습 및 예측을 수행하였다. 실험 후, 실제 생성된 데이터와 예측 데이터를 비교하여 예측정확도를 측정하였다.

결과를 통해 예측정확도는 T_2=1일 경우 79%의 정확도를 보였으며, T_2=10회일 경우 85%의 정확도를 보였다. T_2>=10으로 할 경우, 예측정확도는 80%을 상회하는 것으로 나타났다.



(그림 11) 최소 실행 수에 따른 예측정확도 그래프

5.4 실험 4 : 예측품질 측정 실험

본 실험은 본 논문에서 제시한 학습기반 수행 예측모델을 이용하여 생성한 예측 값과 실시간 프로세스를 동작시킨 후 추출한 실제 값과의 유사성을 파악하고 예측품질을 측정하기 위해서 수행하였다.

본 실험을 위해 T_1 = 10으로 T_2 = 10으로 설정하였으며, 나머지 실험과정은 실험 3과 동일하며 실제 값과 도출된 예측값의 차를 구하였다. 예측품질을 파악하기 위해 도출한 결과값들에 대한 대응표본 t-검증(paired-t test)을 실시하였다. 이 실험은 두 개의 짝을 이루는 값들에 대한 평균차이를 측정하기 위해 이용한다. 실험을 실시하기 전에 다음과 같은 가설을 세웠다.

예측 값과 실제 값은 다르다.

. 가설검증을 위한 귀무가설과 대립가설은 다음과 같다. 여기서 D_0는 귀무가설(H_0)로 설정된 차이를 의미한다.

$$H_0 : D_0 = 0$$
$$H_1 : D_0 \neq 0$$

대응표본 t-검증의 결과는 다음과 같다.

〈표 4〉 대응표본 통계량

		평 균	N	표준편차	표준오차 평 균
대응1	예측값	365.68	130	.13	1.12E-02
	실제값	365.69	130	.46	4.06E-02

〈표 5〉 대응표본 검정

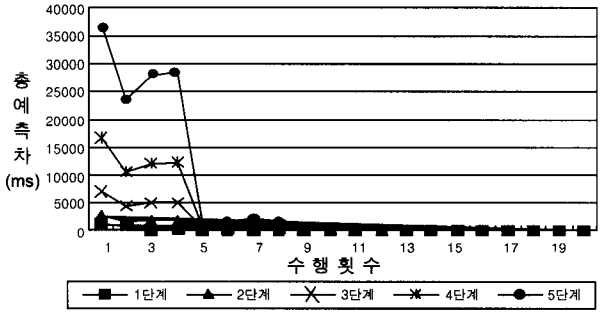
		대 응 차		t	자유도	유의확률
		95% 신뢰구간				
		하 한	상 한			
대응1	예측값 -실제값	-9.65E-02	7.35E-02	-.269	129	.789

<표 5>에서 나타난 것과 같이 양측검증에서 t-value는 -.269, p-value는 .789로 나타나 '$H_0 : D_0 = 0$' a=.05에서 지지된다. 따라서 예측값과 실제 값은 차이가 없는 것으로 나타났다.

5.5 실험 5 : 다단계 예측품질 측정 실험

본 실험은 본 논문에서 제시한 다단계 예측기법을 적용한 경우의 예측품질을 측정하기 위한 실험이다. 본 실험은 1단계에서부터 5단계 예측을 했을 때, 각 단계 예측 값과 실제 값 간의 차이를 비교하였다.

본 실험을 위해 $T_1 = 10$으로 $T_2 = 10$으로 설정하였으며, 각 단계별로 도출된 실제 값과 예측 값의 차를 구하였다. (그림 12)와 같이, 훈련집합의 값을 이용해 예측을 하는 경우, 실제 값과의 차이가 크게 나타나지만, T_1만큼의 인스턴스가 수집되는 수행횟수 10지점에서는 1~3ms만 차이가 나는 것으로 나타났다. 또한 본 실험에서 나타난 것과 같이 실시간 프로세스가 안정적으로 동작하는 경우에는 단계간 예측 차이가 크지 않은 것으로 나타났다.



(그림 12) 예측단계에 따른 예측차

5.6 실험의 결론

상기 제시한 실험 1에서 실험 5를 통해 다음과 같은 결론을 도출할 수 있다.

5.6.1 훈련집합의 크기는 얼마로 정해야 하는가?

훈련집합의 크기(T_1)가 미치는 영향도를 결정하기 위해, 5.1절에서 기술한 학습모듈의 판단정확성 파악 실험을 수행하였다. T_1을 1에서 100까지 증가시켰으며, 판단 결과를 도출하기 위해서는 훈련 시 사용한 인스턴스들과 독립적인 400여 개의 인스턴스를 입력하여 학습모듈이 판단결과를 생성하도록 하였다. 이러한 실험을 반복적으로 수행하였다.

결과를 통해 T_1 >=10이면, 80% 이상의 판단 정확도를 보이며, 판단 정확도를 높이기 위해 T_1 >= 10으로 설정해야 하는 것으로 나타났다.

5.6.2 최소 실행 수는 몇 회로 정해야 하는가?

최소 실행 수(T_2)가 미치는 영향 정도를 파악하기 위해서, 5.3절에서 기술한 최소 실행 수에 따른 예측정확도 파악 실험을 수행하였다. 실험을 위해 $T_1 = 50$으로 설정하였으며(평균 정확도 86%), 훈련 시 사용한 인스턴스와 독립적인 200개의 인스턴스를 입력하였다. T_2는 1회에서 50회로 증가시켜 가면서 반복적으로 수행하여, 예측의 정확도를 측정하였다.

결과를 통해 T_2 >= 10일 때, 84% 이상의 예측정확도를 보이며, T_2 >= 30이상이면, T_1 = 50일 경우 제공하는 86% 정확도에 근접하는 것으로 나타났다. 따라서 T_2 >= 10이상으로 설정해야 한다.

5.6.3 다단계 예측의 경우 몇 단계까지 해야 하는가?

다단계 예측에서 예측단계(T_3)가 미치는 영향도를 파악하기 위해서, 5.5절에서 다단계 예측품질 측정 실험을 수행하였다. 실험을 위해 $T_1 = 10$, $T_2 = 10$으로 설정하였고, 훈련 시 사용한 인스턴스와는 독립적인 150개의 인스턴스를 입력하였다. T_3는 1단계에서 5단계로 증가시키며 반복적으로 실험을 수행하였다.

결과를 통해 T_3에 관계없이 T_1만큼의 수행회수를 넘게 되면, 다단계 예측의 예측 차가 최소화되는 것으로 나타났다. 따라서 T_3는 어떤 값으로 설정하든지 예측품질에 큰 영향을 주지는 않지만, T_3이 클수록 예측 차가 크다는 점을 유의해야 한다.

6. 논문의 한계점

본 본문의 실시간 감시 체계에 적용할 수 있는 학습기반 수행 예측모델을 제안하였다. 하지만 몇 가지 한계점이 있었다.

1) 시스템 자원이 프로세스의 수행에 미치는 영향을 파악할 수 있도록 다차원 자원공간 모델을 지원하지 못하는 한계

본 논문의 설계 및 검증 과정에서는 프로세스의 실행 시간에 초점을 맞추었다. 하지만 실시간 프로세스는 CPU, 메모리, 입출력 등의 시스템 자원의 성능에 영향을 받게 된다. 실시간 프로세스의 수행 결과만 아니라, 실시간 프로세스가 받는 시스템 자원에 의해 받는 영향을 파악하기 위해서는 CPU, 메모리, 입출력 데이터를 다루는 다차원 자원공간 모델을 구성하고, 이를 학습기반 수행 예측모델에 적용하기 위한 연구가 필요하다.

2) 단순한 학습 알고리즘인 IB1만을 적용한 한계

학습 알고리즘 관련 연구는 다수 존재한다. 인스턴스 기반 학습 알고리즘의 경우, IB4까지 연구되었다. 따라서 향후에는 보다 고도화된 학습 알고리즘을 수행 예측모델에 적용하여, 예측 품질을 높일 수 있는 연구가 필요하다.

7. 결론 및 향후 방향

본 논문에서는 기존의 실시간 감시 체계가 가지고 있던 한계를 해결하기 위해 학습기반 수행 예측모델을 제안하였다. 이 학습기반 수행 예측모델은 실시간 프로세스가 동작하는 시점에서 동적분석을 통한 예측이 가능하며, 예측을 위

해 수행되는 사전 기초 통계분석 과정을 제거하였다. 또한 이전확률이나 상태 정보 등은 최신의 것을 유지할 수 있는 체계를 제공하며, 이를 기초로 다단계 예측이 가능하도록 하였다.

또한 본 논문에서는 학습 기반의 수행 예측모델을 적용하기 위한 실행시간 프로세스 모니터의 구조를 제시하여, 향후 실시간 감시 체계를 구현할 수 있는 체계를 설계하였다. 본 논문에서 제시한 학습기반 수행 예측모델과실행시간 프로세스 모니터의 설계는 실시간 감시 및 제어분야에서 활용될 수 있다.

향후 작업으로는 첫째, 특정 실시간 프로그래밍 모델(TMO (Time-triggered Message-triggered Object) 모델 등)을 이용해 본 논문에서 제시한 수행 예측모델을 적용한 실행시간 프로세스 모니터의 구현, 둘째, 다차원 자원 공간 모델을 이용한 다수의 자원에 대한 복합적 예측이 가능한 수행 예측모델의 개선하고자 한다.

참 고 문 헌

[1] "How To Enable Process Accounting on Linux," http://kldp.org

[2] 정윤석, 김태완, 장천현, "실행시간 프로세스 모니터를 위한 XML기반의 데이터 저장소의 설계", 정보처리학회논문지A, 제10-A권 제6호, Dec., 2003.

[3] 정윤석, 김태완, 장천현, "실행시간 프로세스 모니터를 위한 구조 설계", 정보처리학회 춘계학술발표대회논문집, 제10권 제1호 2003.

[4] Yoon Seok Jeong, Tae Wan Kim, Chun Hyon Chang, "Design and Implementation of a Run-time TMO Monitor on LTMOS," Proc. Embedded Systems and Applications, Jun., 2003.

[5] B. J. Min, et al:, "Implementation of a Run-time Monitor for TMO Programs on Windows NT," IEEE Computer Jun., 2000.

[6] Kim, J. G. and Cho, S. Y., "LTMOS : An Execution engine for TMO-Based Real-Time Distributed Objects," Proc. PDPTA '00, LasVegas, Vol.V, pp.2713-2718, Jun., 2000.

[7] S.H.Park, "LTMOS(LinuxTMO System)'s Manual," HUFS, Mar., 2000.

[8] Kalyanaraman Vaidyanathan, Kishor S. Trivedi, "A Measurement-Based Model for Estimation of Resource Exhaustion in Operational Software Systems," 10th International Symposium on Software Reliability Engineering, 1999.

[9] A. K. Mok and G. Liu, "Efficient Run-Time Monitoring of Timing Constraints," Proc. Real-Time Technology and Application, Jun., 1997.

[10] Hyung-Taek Lim, et al., "Monitor based Fault Management in a Distributed Environment," 1995.

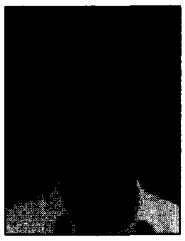
[11] B. A. Schroeder, "On-line Monitoring : A Tutorial," IEEE Computer, June, 1995.

[12] S. Sankar and M. Mandal, "Concurrent Runtime Monitoring of Formally Specified Programs," IEEE Computer Mar., 1993.

[13] Mike Loukides, "System Performance Tuning," O'Reilly, 1990.

[14] David W. Aha, "A Study of Instance-Based Algorithms for Supervised Learning Tasks," Technical Report 90-42, Nov., 1990.

[15] Murthy V. Devarakonda and Ravishankar K. Iyer, "Predictability of Process Resource Usage : A Measurement-Based Study on UNIX," IEEE Transactions on Software Engineering, Dec., 1989.

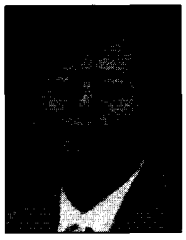


정 윤 석

e-mail : ysjeong@cse.konkuk.ac.kr
2000년 건국대학교 컴퓨터공학과 (공학석사)
2000년~현재 건국대학교 컴퓨터공학과 박사과정
관심분야 : 실시간 시스템 모니터링, 객체지향 프로그래밍, XML

김 태 완

e-mail : twkim@konkuk.ac.kr
1994년 건국대학교 전자계산학과(공학사)
1996년 건국대학교 전자계산학과 (공학석사)
1996년~2001년 현대중공업 기전연구소 연구원
2003년~2004년 경민대학 인터넷비즈니스과 겸임교수
1996년~현재 건국대학교 컴퓨터공학과 박사과정
2004년~현재 건국대학교 컴퓨터공학부 강의교수
관심분야 : 프로그래밍 언어, 실시간 프로그래밍, 자동화 소프트웨어, 산업기기 감시 진단 제어 시스템



장 천 현

e-mail : chchang@konkuk.ac.kr
1977년 서울대학교 계산통계(학사)
1979년 KAIST 전산학(석사)
1985년 KAIST 전산학(박사)
현재 건국대학교 컴퓨터 공학과 정교수
관심분야 : 프로그래밍 언어, 컴파일러, 실시간 시스템