실시간 제어 시스템의 결함 허용성을 위한 적응형 체크포인팅 기법

An Adaptive Checkpointing Scheme for Fault Tolerance of Real-Time Control Systems

류 상 문*
(Sang-Moon Ryu)

Abstract: The checkpointing scheme is a well-known technique to cope with transient faults in digital systems. This paper proposes an adaptive checkpointing scheme for the reliability improvement of real-time control systems. The proposed adaptive checkpointing scheme is based on the previous work about the reliability problem of an equidistant checkpointing scheme. For the derivation of the adaptive scheme, some conditions are introduced which are to be satisfied for the reliability improvement by exploiting an equidistant checkpointing scheme. Numerical data show the proposed adaptive scheme outperforms the equidistant scheme from a reliability point of view.

Keywords: real-time control system, fault tolerance, adaptive checkpointing

I. 서론

실시간 제어 시스템은 제어 작업을 수행하기 위한 특정 목적의 컴퓨터 시스템으로서, 제어 작업 수행에 있어 요구되는 시간 제약 조건을 충실히 만족해야 한다. 그리고 종종 열악하고 급변하는 환경 속에서 중단 없이 오랜 시간 지속적으로 동작하도록 요구되기 때문에 높은 신뢰도 조건을 만족하여야 한다.

실시간 제어 시스템을 포함한 모든 컴퓨터 시스템은 일시적 결함(transient faults)이 유발하는 일시적 오류(transient errors)로 인하여 오동작할 수 있다[1-3]. 컴퓨터 시스템에서 일시적 오류를 극복하기 위한 방안은 계속 연구되어 왔으며 대표적인 방안으로 체크포인팅(checkpointing) 기법[4]을 꼽을 수 있다.

체크포인팅(checkpointing) 기법은 컴퓨터 시스템에서 실행되는 태스크(task)의 상태를 안정성이 높은 저장 장치에 주기적으로 저장하는 것을 의미한다. 이렇게 저장된 태스크의 상태를 체크포인트(checkpoint)라고 하며, 일시적 오류가 감지되었을 때 최근에 생성된 체크포인트를 이용하여 태스크를 오류 발생 이전 상태로 되돌리는 것을 롤백 복구(rollback recovery)라 한다.

체크포인팅 작업이 일정 시간 간격으로 수행되는 경우를 등간격 체크포인팅(equidistant checkpointing) 기법이라 하고, 체크포인팅 작업의 수행 시간 간격이 변화하는 경우를 적응형 체크포인팅(adaptive checkpointing) 기법이라 한다.

대부분의 등간격 체크포인팅 기법에 관한 이전 연구는 데이터베이스 시스템이나 범용 컴퓨터 시스템에서의 시스템의 가용성 최대화나 성능 평가, 프로그램의 평균 실행 시간 최소화, 체크포인팅 부담 최소화 등이 주요 주제였으며[6,5], 실시간 시스템에 대해서는 태스크 평균 실행 시간이나 응답 시간 분석 등에 관한 연구[7,8]가 주요 주제였었다[9].

* 책임저자(Corresponding Author)
논문접수: 2009. 1. 29., 채택확정: 2009. 4. 19.
류상문: 군산대학교 제어로봇시스템공학과(smryu@kunsan.ac.kr)
※ 이 논문은 2007학년도 군산대학교 신임교수 연구비 지원에 의하여 연구되었음.

적응형 체크포인팅 기법에 관한 이전 연구 [10-13]는 실시간 제어 시스템과는 성격이 다른 시스템들을 대상으로 이루어졌다. 실시간 제어 시스템에 대해 적용할 만한 적응형 체크포인팅 기법에 관한 이전 연구는 [14-16]가 있으며, 이들은 실시간 태스크의 실행에 소요되는 평균 시간을 최소화하여 요구되는 시간 제약을 만족할 수 있는 확률을 최대로 하는 기법을 제안하였다.

본 논문에서는 실시간 제어 시스템의 신뢰도 향상을 목표로 하는 등간격 체크포인팅에 관한 이전 연구 [9]의 결과를 바탕으로 하여, 실시간 제어 시스템의 신뢰도를 더욱 향상할 수 있는 적응형 체크포인팅 기법에 대해 제안하고 그 효과를 보인다. 이전 연구 [9]에서와 마찬가지로, 실시간 제어 시스템이 일시적 오류를 극복하면서 주어진 시간 조건을 만족하며 센서로부터의 입력에 따른 구동장치를 위한 출력 계산을 완료할 확률, 즉 주기적 실시간 제어 태스크(이하 태스크)의 1회 실행 성공 확률을 성능 지표로 정하고 이를 통하여 제안하는 적응형 체크포인팅 기법의 효과를 검토한다.

II 장에서는 대상 시스템의 모델 및 가정에 대해 기술하고, III 장에서는 적응형 체크포인팅 기법 유도를 위해 필요한 등간격 체크포인팅에 관한 몇 가지 고찰 결과를 소개한다. 그리고 IV 장에서 적응형 기법을 제안하고 V 장에서 그 효과를 보인 후 VI 장에서 결론을 맺는다.

II. 가정 및 시스템 모델

본 연구에서 적용한 가정과 시스템 모델은 체크포인팅 작업 수행의 시간 간격이 고정되어 있지 않고 적응형으로 변화한다는 것을 제외하고는 이전 연구 [9]과 동일하며, 간단히 정리하면 다음과 같다.

일시적 오류는 평균값 λ 를 갖는 포아송(poisson) 분포에 따라 발생한다고 가정한다[5-8,10,11,14-16]. 그리고 발생한 일시적 오류는 그림 1과 같은 이중화 구조와 비교기를 통해서 체크포인팅 작업 수행 직전과 직후 그리고 롤백 복구 작업 수행 직후에 완벽하게 검출된다고 가정한다. 실제로는 완벽할 수 없는 오류 검출 기법의 성능을 고려하려면 적용된 검

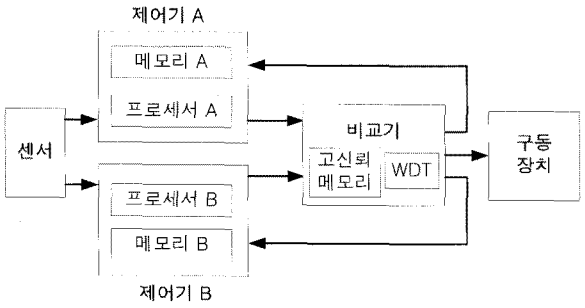


그림 1. 오류 검출을 위한 이중화 구조 제어 시스템.
Fig. 1. DMR structured control system for error detection.

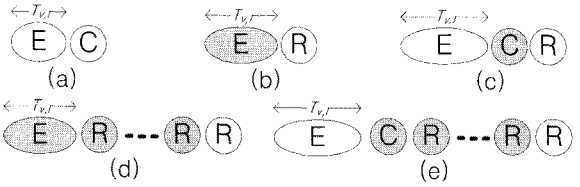


그림 2. 체크포인팅 기법이 적용된 태스크의 가능한 수행 패턴.
Fig. 2. Possible execution patterns of a task with checkpointing.

그림 3. 태스크의 1회 수행 패턴의 예.
Fig. 3. An execution pattern of a release of a task.

출 기법의 검출 범위(detection coverage) [1]를 분석된 시스템의 신뢰도에 적용하면 된다.

그림 2는 적응형 체크포인팅 기법이 적용된 경우에 태스크의 수행 중 나타날 수 있는 패턴을 보여준다. 그림에서 E, C와 R은 각각 태스크의 실행 구간, 체크포인팅 작업 그리고 롤백 복구 작업을 의미하며, 회색으로 표현된 것은 해당 구간에서 일시적 오류가 발생하였음을 나타낸다.

적응형 체크포인팅 기법에서는 체크포인팅 작업 수행 시점이 태스크가 활용할 수 있는 잔여 여유 시간과 태스크의 한 주기 수행을 완료하기 위해 요구되는 시간에 따라서 동적으로 정해진다. 그림 3는 태스크의 한 주기 실행 시작 후 $T_{v,1}$에 체크포인팅 작업이 수행되고, $T_{v,2}$후에는 일시적 오류의 발생이 감지되어 체크포인팅 작업 대신 롤백 복구가 수행되고, $T_{v,3}$후에는 체크포인팅 작업 수행 중 오류가 검출되어 롤백 복구 작업이 수행되고, $T_{v,3}$후에는 정상적으로 태스크의 실행 구간이 수행되고 체크포인팅 작업이 수행된 경우를 보여준다. 체크포인팅 수행 시점 $T_{v,l}, l=1,2,3,...,$ 을 정하는 것이 적응형 체크포인팅 기법의 핵심이며 다음 장에서 상세히 기술된다.

본 논문에서 사용되는 주요 기호는 이전 연구[9]에서 사용된 것과 동일하며, 적응형 기법을 표현하기 위해 필요한 기호들은 추가로 정의될 것이다.

III. 등간격 체크포인팅 기법에 대한 고찰

각 제어기에서 동일하게 실행되는 주기적 실시간 제어 태스크가 센서의 값을 바탕으로 제어 출력을 계산하는데 소요되는 시간을 실행 시간(execution time)이라 하고, 1회의 제어 출력 계산에 허용된 시간을 상대적 마감 시간(relative deadline) 이라고 한다[17]. 상대적 마감 시간과 실행 시간의 차이를 여유 시간(slack time)이라 하며, 이 여유 시간을 활용해서 주기적인 체크포인팅 작업과 필요한 롤백 복구를 수행하여 일시적 결함의 극복할 수 있는 것이다.

이전 연구 [9]의 결과에 의하면 일시적 오류가 평균값 λ를 갖는 포아송 분포에 따라 발생하는 환경에서 실행 시간이 T_e, 여유 시간이 T_s, 체크포인팅 작업 소요 시간이 T_c, 롤백 복구 소요 시간이 T_r 인 실시간 제어 태스크의 1회 실행에 N_c 회의 체크포인팅 작업을 수행하면 이 제어 태스크가 발생할 수 있는 일시적 오류를 극복하고 주어진 시간 조건을 만족하여 수행 완료될 확률 P_s 는 식 (1)처럼 표현될 수 있다.

$$P_s = \sum_{i=0}^{N_i} \sum_{j=0}^{N_j} \Pr\{\langle i,j \rangle\} \tag{1}$$

여기서,

$$N_i = \left\lfloor \frac{T_s - N_c T_c}{\frac{T_e}{N_c} + T_r} \right\rfloor \tag{2}$$

$$N_j = \left\lfloor \frac{T_s - N_c T_c - i\left(\frac{T_e}{N_c} + T_r\right)}{\frac{T_e}{N_c} + T_c + T_r} \right\rfloor \tag{3}$$

$$\begin{aligned} \Pr\{\langle i,j \rangle\} = & \binom{N_c - 1 + i + j}{i} \binom{N_c - 1 + j}{j} \\ & \cdot \left(1 - e^{-\lambda \frac{T_e}{N_c}}\right)^i e^{-i\lambda T_r} \\ & \cdot e^{-j\lambda \frac{T_e}{N_c}} (1 - e^{-\lambda T_c})^j e^{-j\lambda T_r} \\ & \cdot e^{-\lambda\left(\frac{T_e}{N_c} + T_c\right) N_c} \cdot \sum_{k=0}^{N_k} \frac{(i+j)^k}{k!} (1 - e^{-\lambda T_r})^k \end{aligned} \tag{4}$$

$$N_k = \begin{cases} 0, & \text{when } i = j = 0 \\ \left\lfloor \dfrac{T_s - N_c T_c - i\left(\frac{T_e}{N_c} + T_r\right) - j\left(\frac{T_e}{N_c} + T_c + T_r\right)}{T_r} \right\rfloor, & \text{otherwise} \end{cases} \tag{5}$$

그리고 $\lfloor \cdot \rfloor$ 와 $\lceil \cdot \rceil$ 는 각각 주어진 값보다 크지 않은 최대의 정수와 주어진 값보다 작지 않은 최소의 정수를 의미한다.

적응형 체크포인팅 기법의 도입을 위하여 등간격 체크포인팅 기법에 대한 이전 연구 [9]의 결과에 대해 다음의 세 가지 사항을 고찰한다.

1. 최소 필요 여유 시간

등간격 체크포인팅을 수행하는 것이 실시간 제어 시스템의 신뢰도를 향상시키기 위해서 요구되는 최소한의 여유 시간을 '최소 필요 여유 시간', T_s^m 라고 정의하고 이를 구한다. 신뢰도가 향상되기 위해서는 태스크 실행 구간에서 발생할

수 있는 일시적 오류를 한 번이라도 극복할 수 있어야 하며, 여유 시간 T_s 는 이를 허용할 수 있어야 한다. 이를 수식으로 표현하면 식 (6)와 같다.

$$T_s > T_c N_c + \frac{T_e}{N_c} + T_r \tag{6}$$

식 (6)의 우측 부분을 최소화하는 정수 N_c 를 구해보면 $\left\lfloor \sqrt{\frac{T_e}{T_c}} \right\rfloor$ 와 $\left\lceil \sqrt{\frac{T_e}{T_c}} \right\rceil$ 중에서 식 (6)의 우측을 더 작게 만들어 주는 값임을 알 수 있다. 따라서 최소 필요 여유 시간 T_s^m 는 식 (7)과 같이 표현될 수 있으며, 등간격 체크포인팅 기법 적용이 실시간 제어 시스템의 신뢰도를 향상시킬 수 있으려면 T_s 는 식 (8)를 만족하여야 한다.

$$\begin{aligned} T_s^m = \min(&T_c \left\lfloor \sqrt{\frac{T_e}{T_c}} \right\rfloor + \frac{T_e}{\left\lfloor \sqrt{\frac{T_e}{T_c}} \right\rfloor} + T_r, \\ &T_c \left\lceil \sqrt{\frac{T_e}{T_c}} \right\rceil + \frac{T_e}{\left\lceil \sqrt{\frac{T_e}{T_c}} \right\rceil} + T_r) \end{aligned} \tag{7}$$

$$T_s \geq T_s^m \tag{8}$$

2. 체크포인팅 수행 조건

식 (1)에서 크기가 우세한 세 항만을 추출하여 정리하면 식 (9)과 같다.

$$\begin{aligned} P_s \approx\ & e^{-\lambda\left(\frac{T_e}{N_c}+T_c\right)N_c} + N_c e^{-\lambda\frac{T_e}{N_c}}\left(1-e^{-\lambda T_c}\right)e^{-\lambda\left(\frac{T_e}{N_c}+T_c\right)N_c} \\ &+ N_c\left(1-e^{-\lambda\frac{T_e}{N_c}}\right)e^{-\lambda T_r}e^{-\lambda\left(\frac{T_e}{N_c}+T_c\right)N_c} \\ =& \left\{1+N_c e^{-\lambda T_r}\left[\left(1-e^{-\lambda\frac{T_e}{N_c}}\right)+e^{-\lambda\frac{T_e}{N_c}}\left(1-e^{-\lambda T_c}\right)\right]\right\} \\ &\cdot e^{-\lambda(T_e+T_c N_c)} \end{aligned} \tag{9}$$

체크포인팅 적용으로 신뢰도가 향상되기 위해서는 식 (9)의 결과가 체크포인팅을 적용하지 않았을 때의 태스크의 성공적 실행 확률 $e^{-\lambda T_e}$ 보다 커야 하므로 식 (10)가 성립해야 한다.

$$e^{\lambda T_c N_c} - 1 < N_c e^{-\lambda T_r}\left(1-e^{-\lambda\left(\frac{T_e}{N_c}+T_c\right)}\right) \tag{10}$$

$0 < a < 1$ 을 만족하는 실수 a 에 대해 $e^a - 1 < 2a$, $1-a < e^{-a}$ 그리고 $a - \frac{1}{2}a^2 < 1 - e^{-a}$ 이 성립하고 λ 의 값은 매우 작으므로 식 (10)의 각 항에 대해 식 (11), (12) 그리고 (13)을 얻을 수 있다.

$$e^{\lambda N_c T_c} - 1 < 2\lambda T_c N_c \tag{11}$$

$$1 - \lambda T_r < e^{-\lambda T_r} \tag{12}$$

$$\lambda\left(\frac{T_e}{N_c}+T_c\right) - \frac{1}{2}\lambda^2\left(\frac{T_e}{N_c}+T_c\right)^2 < 1 - e^{-\lambda\left(\frac{T_e}{N_c}+T_c\right)} \tag{13}$$

식 (11), (12), (13)을 식 (10)에 적절히 대입하면 다음의 식 (14)을 얻을 수 있다.

$$\begin{aligned} 2T_c < &\frac{T_e}{N_c} + T_c - \lambda T_r\left(\frac{T_e}{N_c}+T_c\right) - \frac{\lambda}{2}\left(\frac{T_e}{N_c}+T_c\right)^2 \\ &+ \frac{\lambda^2}{2}T_r\left(\frac{T_e}{N_c}+T_c\right)^2 \end{aligned} \tag{14}$$

식 (14)는 등간격 체크포인팅 기법이 적용될 때 신뢰도 증가를 위해 만족되어야 할 조건이 된다. 만일 식 (14)에서 λ 가 포함되어 있지 않은 우세항만을 남겨놓으면 식 (15)를 얻게 된다.

$$T_c < \frac{T_e}{N_c} \tag{15}$$

그리고 등간격 체크포인팅에서 태스크의 한 주기당 필요한 최소 체크포인팅 작업 수는 2 이상이므로 식 (15)로부터 최소한의 체크포인팅 작업이 적용될 때에도 만족되어야 하는 조건을 의미하는 식(16)을 얻을 수 있다.

$$2T_c < T_e \tag{16}$$

3. 부최적(Suboptimal) 등간격 체크포인팅 수행 방안

식 (8)과 식(15)의 조건이 만족되면 등간격 체크포인팅 기법 적용으로 인해 신뢰도가 향상된다고 기대할 수 있다. 신뢰도 관점에서의 최적 등간격 체크포인팅 방법은 식 (1)의 P_s 를 최대화하는 N_c 를 구하여 이를 적용하는 것이다. 그러나 이를 수식으로 유도하기는 매우 어렵다.

이전 연구 [9]의 표 1과 그림 4를 보면 식 (1)의 P_s 가 식 (2)의 N_i 에 비례하는 것을 알 수 있다. 이것은 N_i 의 값이 일시적 오류를 포용할 수 있는 실행구간의 수를 의미하기 때문이다. 따라서 N_i 의 값을 최대로 만들어주는 N_c 를 구하면 부최적 체크포인팅을 수행할 수 있게 된다.

N_i 를 최대로 하여주는 N_c 의 값은

$$\left\lfloor \frac{-T_e + \sqrt{T_e(T_e + T_s T_r / T_c)}}{T_r} \right\rfloor \text{와} \left\lceil \frac{-T_e + \sqrt{T_e(T_e + T_s T_r / T_c)}}{T_r} \right\rceil$$

중 하나인데 태스크의 한 주기 실행 동안에 수행되는 체크포인팅 작업 수가 작을수록 태스크가 일시적 오류에 적게 노출될 것이므로 부최적 등간격 체크포인팅 작업 회수 N_c^o 를 식 (17)과 같이 정할 수 있다.

$$N_c^o = \left\lfloor \frac{-T_e + \sqrt{T_e(T_e + T_s T_r / T_c)}}{T_r} \right\rfloor \tag{17}$$

즉, 식 (17)의 N_c^o 를 적용하여 등간격 체크포인팅을 수행하면 적은 체크포인팅 작업 수행 부담으로 높은 신뢰도를 얻을 수 있다는 것이다.

IV. 적응형 체크포인팅 기법

적응형 체크포인팅 기법은 현재 태스크의 수행 상황에 따라 태스크의 한 주기가 시작하는 시점과 체크포인팅 또는 롤백 복구 작업이 정상적으로 완료되는 시점에서 $l(=1,2,3,\cdots)$ 번째 체크포인팅 작업 수행 시점 $T_{v,l}$을 결정하여 이를 적용하는 것이다. $T_{v,l}$을 계산하는 시점에서 태스크가 활용할 수 있는 '잔여 여유 시간'을 $T_{s,l}$, 태스크의 한 주기 수행을 완료하기 위한 '잔여 실행 시간'을 $T_{e,l}$이라고 하면, 이들은 다음과 같이 계산이 된다.

첫 번째 ($l=1$) 실행 구간 수행 전:

$$\begin{aligned} T_{e,l} &= T_e \\ T_{s,l} &= T_s \end{aligned} \tag{18}$$

그림 2(a)처럼 l 번째 실행 구간 다음의 체크포인팅 작업이 오류 없이 수행된 경우:

$$\begin{aligned} T_{e,l+1} &= T_{e,l} - T_{v,l} \\ T_{s,l+1} &= T_{s,l} - T_c \end{aligned} \tag{19}$$

그림 2(b)와 (d)처럼 l 번째 실행 구간에서 오류가 발생하고 $m(=0,1,2,\cdots)$ 회의 롤백 복구 작업 중 오류가 발생 한 후 1회의 롤백 복구 작업이 오류 없이 수행된 경우:

$$\begin{aligned} T_{e,l+1} &= T_{e,l} \\ T_{s,l+1} &= T_{s,l} - T_{v,l} - (m+1)T_r \end{aligned} \tag{20}$$

그림 2(c)와 (e)처럼 l 번째 실행 구간 다음의 체크포인팅 작업 중 오류가 발생하고 $n(=0,1,2,\cdots)$ 회의 롤백 복구 작업 중 오류가 발생 한 후 1회의 롤백 복구 작업이 오류 없이 수행된 경우:

$$\begin{aligned} T_{e,l+1} &= T_{e,l} \\ T_{s,l+1} &= T_{s,l} - T_{v,l} - T_c - (n+1)T_r \end{aligned} \tag{21}$$

앞서 기술된 고찰의 결과를 바탕으로 제안되는 적응형 체크포인팅 기법은 l 번째 실행 구간이 수행되기 직전에 l 번째 실행 구간의 길이 $T_{v,l}$을 다음의 규칙을 이용하여 계산한다.

식 (8)와 (16)의 조건이 만족되는 경우(주석문 A)에는 앞서 기술된 부최적 체크포인팅 방안을 적용하여 $T_{v,l}$을 결정한다. 여기서 $N_{c,l}^o$는 식 (17)을 이용하여 식 (22)처럼 표현된다.

```
if (T_{e,l} > 2T_c) and (T_{s,l} ≥ T_s^m) then // A
    T_{v,l} = T_{e,l} / N_{c,l}^o
else if ( (2T_c + T_r) ≤ T_{s,l} < (T_{e,l} + T_c) ) then // B
    T_{v,l} = T_{s,l} - T_c - T_r
else    // C
    T_{v,l} = T_{e,l}
end if
```

그림 3. 제안된 적응형 체크포인팅 기법에서의 $T_{v,l}$ 계산.

Fig. 3. $T_{v,l}$ in the proposed adaptive checkpointing scheme.

$$N_{c,l}^o = \left\lfloor \frac{-T_{e,l} + \sqrt{T_{e,l}(T_{e,l} + T_{s,l}T_r / T_c)}}{T_r} \right\rfloor \tag{22}$$

잔여 여유 시간이 일시적 오류를 극복하는데 필요한 1회의 롤백 복구 작업과 2회의 체크포인팅 작업을 허용할 수 있고, 잔여 실행 시간과 1회의 체크포인팅 작업 시간의 합 보다 작을 때(주석문 B)에는 잔여 여유 시간에서 각 1회의 체크포인팅 작업과 롤백 복구 작업 시간을 제한 것으로 $T_{v,l}$을 결정한다.

마지막으로 발생한 오류를 극복할 수 있는 여유 시간이 확보되지 않은 경우(주석문 C)에는 잔여 실행 시간으로 $T_{v,l}$을 결정하여 태스크의 수행 완료를 도모한다.

V. 성능 비교 및 고찰

그림 4와 표 1은 이전 연구 [9]의 등간격 체크포인팅 기법과 본 논문에서 제안되는 적응형 체크포인팅 기법의 성능 비교를 보인 것이다. $T_e=100$, $T_c=1$, $T_r=2$ 인 태스크가 $\lambda=0.001$ 인 환경에서 수행되는 상황을 가정하였으며, 주어진 여유시간 T_s를 활용하여 등간격 체크포인팅 기법을 적용하여 얻을 수 있는 최대의 태스크 성공적 실행 확률 P_s^e과 앞서 제안된 적응형 체크포인팅 기법을 적용하여 얻을 수 있는 태스크 성공적 실행 확률 P_s^a를 함께 나타낸 것이다. P_s^e는 이전 연구의 결과인 식 (1)을 적용하여 얻었고, P_s^a는 태스크를 10^7회 반복 수행한 시뮬레이션을 통하여 얻었다.

표 1과 그림 4를 보면 적응형 기법이 등간격 기법에 비해 신뢰도 성능면에서 뛰어남을 알 수 있다. 특히 $10 \le T_s \le 21$ 구간에서는 등간격 기법이 신뢰도 개선 효과를 발휘하지 못한다. 이것인 등간격 체크포인팅 기법이 신뢰도를 개선하는데 요구되는 식 (7)의 최소 여유 시간 22가 만족되지 못하기 때문이다. 하지만 적응형 체크포인팅 기법은 주어진 여유 시간을 활용하여 발생할 수 있는 일시적 오류를 극복할 수 있기 때문에 신뢰도를 개선할 수 있다.

등간격 기법은 활용할 수 있는 여유 시간이 증가함에 따른 신뢰도의 개선 추이가 계단 형태로 나타나지만, 적응형 기법은 여유 시간의 증가에 비례하여 신뢰도가 개선되는 것을 알

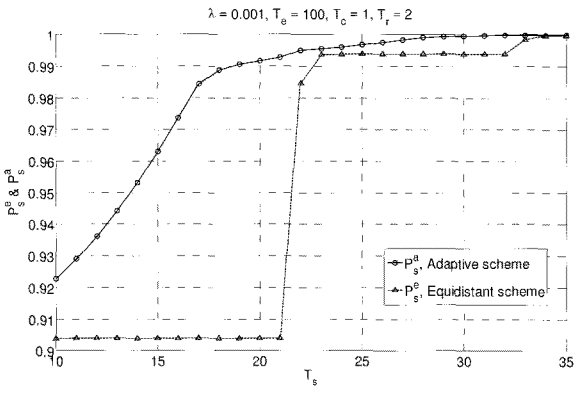


그림 4. 등간격 기법과 적응형 기법의 성능 비교.

Fig. 4. Performance comparison between the equidistant scheme and the adaptive scheme.

표 1. 수치 데이터.
Table 1. Numerical data.

T_s	P_s^e	P_s^a	
		평균	95% 신뢰 구간
10	0.90393	0.92285	0.92270 - 0.92301
11	0.90393	0.92900	0.92887 - 0.92913
12	0.90393	0.93622	0.93607 - 0.93637
13	0.90393	0.94434	0.94422 - 0.94446
14	0.90393	0.95308	0.95295 - 0.95321
15	0.90393	0.96294	0.96283 - 0.96305
16	0.90393	0.97358	0.97351 - 0.97365
17	0.90393	0.98431	0.98421 - 0.98442
18	0.90393	0.98883	0.98875 - 0.98890
19	0.90393	0.99069	0.99060 - 0.99078
20	0.90393	0.99176	0.99169 - 0.99182
21	0.90393	0.99284	0.99279 - 0.99290
22	0.98479	0.99505	0.99499 - 0.99511
23	0.99364	0.99547	0.99544 - 0.99551
24	0.99381	0.99609	0.99604 - 0.99614
25	0.99390	0.99676	0.99673 - 0.99680
26	0.99392	0.99753	0.99750 - 0.99756
27	0.99393	0.99833	0.99830 - 0.99837
28	0.99393	0.99901	0.99898 - 0.99905
29	0.99393	0.99934	0.99931 - 0.99937
30	0.99393	0.99950	0.99948 - 0.99952
31	0.99393	0.99967	0.99966 - 0.99968
32	0.99393	0.99977	0.99976 - 0.99978
33	0.99848	0.99981	0.99980 - 0.99982
34	0.99963	0.99984	0.99983 - 0.99985
35	0.99972	0.99989	0.99988 - 0.99989

수 있다. 그리고 여유 시간이 일정 값 이상($T_s \geq 32$) 제공되면 등간격 기법과 적응형 기법 사이의 성능 차는 현격히 줄어들지만 여전히 적응형 기법의 신뢰도 성능이 우월함을 알 수 있다.

VI. 결론

이전 연구 [9]에서는 일시적 오류에 대한 실시간 제어 시스템의 신뢰도를 개선할 수 있는 등간격 체크포인팅 기법에 대해서 고찰하였다. 본 연구에서는 이전 연구의 결과를 바탕으로, 체크포인팅 작업 수행 시점이 상황에 따라 변하는 적응형 체크포인팅 기법을 제안하고 기존의 방안과 그 성능을 비교하였다.

제안된 적응형 기법은 체크포인팅 작업 수행 시점을 결정할 때마다 태스크의 잔여 여유 시간과 잔여 실행 시간을 각각 등간격 체크포인팅 기법에서의 태스크 여유 시간과 실행 시간으로 보고 이를 고려한 부최적 체크포인팅 기법을 적용하는 것이다. 그리고 부최적 체크포인팅 기법 적용에 필요한 조건이 만족되지 않을 때에는 잔여 여유 시간이 오류의 극복에 활용될 수 있도록 체크포인팅 작업 수행 시점을 정하는 것이다.

제안된 기법은 과거의 오류 발생 상황을 고려하기 때문에 태스크에게 주어진 여유 시간을 더욱 효율적으로 사용할 수 있다. 따라서 기존의 등간격 기법보다 신뢰도 성능을 더욱 향상할 수 있다. 하지만 태스크의 수행이 완료될 때까지 체크포인팅 수행 시점을 여러 번 결정해야 하기 때문에 기존의 등간격 기법 보다 계산량이 조금 증가한다. 증가된 계산량은 체크포인팅 작업 소요 시간과 롤백 복구 소요 시간의 증가로 이어질 수 있으므로 유사한 성능을 내면서 계산량을 줄이는 방안을 강구해야 할 필요가 있다.

참고문헌

[1] B. W. Johnson, *Design and Analysis of Fault-Tolerant Digital Systems*, Addison-Wesley, 1989.

[2] D. P. Siewiorek, *Reliable Computer Systems: Design and Evaluation*, A K Peters, 1998.

[3] E. Dupont, M. Nicolaidis, and P. Rohr, "Embedded robustness IPs for transient-error-free ICs," *IEEE Design & Test of Computers*, vol. 19, pp. 56-70, May-Jun. 2002.

[4] B. Randell, "System structure for software fault tolerance," *IEEE Trans. Software Engineering*, vol. 1, no. 2, pp. 220-232, June 1975.

[5] Y. Ling, J. Mi, and X. Lin, "A variational calculus approach to optimal checkpoint placement," *IEEE Trans. Computers*, vol. 50, no. 7, pp. 699-708, Jul. 2001.

[6] A. Ziv and J. Bruck, "An on-line algorithm for checkpoint placement," *IEEE Trans. Computers*, vol. 46, no. 9, pp. 976-985, Sep. 1997.

[7] K. G. Shin, T.-H. Lin, and Y.-H. Lee, "Optimal checkpointing of real-time tasks," *IEEE Trans. Computers*, vol. C-36, no. 11, pp. 1328-1341, Nov. 1987.

[8] S. Punnekkat, A. Burns, and R. Davis, "Analysis of checkpointing for real-time systems," *The Int'l Journal of Time-Critical Computing Systems (Real-Time Systems)*, vol. 20, no. 1, pp. 83-102, Jan. 2001.

[9] 유상문, "실시간 제어 시스템의 결함 극복을 위한 이중화 구조와 체크포인팅 기법의 성능 분석," 제어 · 로봇 · 시스템학회 논문지, 제14권 제4호, pp. 376-380, Apr. 2008.

[10] C.-M. Lin and C.-R. Dow, "Efficient techniques for adaptive independent checkpointing in distributed systems," *IEICE Trans. Information & Systems*, vol. E83-D, no. 8, pp. 1642-1653, Aug. 2000.

[11] N. Chen and S. Ren, "Architecture support for behavior-based adaptive checkpointing," *Journal of Software*, vol. 3, no. 2, pp. 61-68, Feb. 2008.

[12] Y. Gao, C. Deng, and Y. Che, "An adaptive index-based algorithm using time-coordination in mobile computing," *Proc. 2008 International Symposiums on Information Processing*, pp. 578-585, May 2008.

[13] M. Chtepen, F. Claeys, B. Dhoedt, F. Turck, P. Vanrolleghem, and P. Demeester, "Providing fault-tolerance in unreliable grid systems through adaptive checkpointing and replication," *Lecture Notes in Computer Science*, vol. 4487, pp. 454-461, 2007.

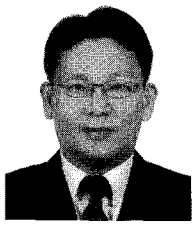
[14] Z. Li, H. Chen, and S. Yu, "Performance optimization for energy-aware adaptive checkpointing in embedded real-time

systems," *Proc. the conference on Design, Automation and Test in Europe*, pp. 678-683, Mar. 2006.

[15] Y. Xiang, Z. Li, and H. Chen, "Optimizing adaptive checkpointing schemes for grid workflow systems," *Proc. the Fifth International Conference on Grid and Cooperative Computing Workshops*, pp. 181-188, Oct. 2006.

[16] Y. Zhang and K. Chakrabarty, "Dynamic adaptation for fault tolerance and power management in embedded real-time systems," *ACM Trans. Embedded Computing Systems*, vol. 3, no. 2, pp. 336-360, May 2004.

[17] J. W. S. Liu, *Real-Time Systems*, Prentice-Hall, 2000.



류 상 문

1992년 금오공과대학교 전자공학과 졸업. 1995년 한국과학기술원 전기및전자공학과 석사. 2006년 동 대학원 전자전산학과 박사. 1995년~2000년 LG전자(주). 2000년~2004년 한국과학기술원. 2006년~현재 군산대학교 제어로봇시스템공학과 조교수. 관심분야는 임베디드 제어 시스템, 실시간 제어 시스템, 결함허용 임베디드 시스템.