Journal of KIIT. Vol. 20, No. 1, pp. 41-48, Jan. 31, 2021. pISSN 1598-8619, eISSN 2093-7571 41
http://dx.doi.org/10.14801/jkiit.2022.20.1.41

대규모 분산 시스템을 위한 적응적 송신자 기반 메시지 로깅 및 검사점 기록 프로토콜

안 진 호*

Adaptive Sender-based Message Logging and Checkpointing Protocol for Large-Scale Distributed Systems

Jinho Ahn*

이 논문은 2019학년도 경기대학교 연구년 수혜로 연구되었음(과제번호: 2019-024-001).

요 약

최근에 강제적 검사점 수를 상당히 줄일 수 있는 두 개의 상호보완적인 통신 유도 검사점 기록 프로토콜이 개발되었다. 첫 번째 프로토콜은 메시지를 송신한 각 프로세스가 추가 메시지 발생 없이 기존 기법들에 비해 메시지 수신자의 최신 타임스탬프 값을 보다 이른 시점에 알 수 있도록 하는 개선된 송신자 기반 메시지 로깅 기법을 포함한다. 두 번째 프로토콜은 각 수신자가 메시지 송신자의 회복가능성을 확인할 수 있게 함으로써 부분 결정적 모델을 가정하지 않고 메시지 로깅을 활용할 수 있다. 본 논문에서는 이러한 두 프로토콜의 주요 비용경감기법들을 결합하여 어떠한 필요 없는 기본 검사점도 허용하지 않으며 그 비용을 크게 떨어뜨릴 수 있는 효율적인 프로토콜을 제안한다. 시뮬레이션은 강제적 검사점 수의 측면에서 애플리케이션 통신패턴에 관계없이 최근 기법에 비해 두 기법의 시너지로부터 얻어지는 비용절감효과가 1.4배~6.2배임을 보여준다.

Abstract

Recently, two complementary communication-induced checkpointing protocols have been developed that may considerably decrease the number of forced checkpoints. The first includes an enhanced sender-based message logging to allow each process sending a message to know the most recent timestamp value of its receiver much earlier without any extra message compared with the existing ones. The second can exploit message logging with no piecewise-deterministic model assumption by enabling each receiver to check recoverability of its sender. This paper proposes an efficient protocol to combine the primary overhead alleviation techniques of the two protocols to greatly drop the overhead while permitting no useless basic checkpoint. Simulation shows that, in terms of the number of forced checkpoints, the cost-saving effect gained from the synergy of the two techniques is 1.4 to 6.2 times compared with the recent one, regardless of communication patterns of applications.

Keywords
distributed system, fault-tolerance, checkpointing, message logging

* 경기대학교 AI컴퓨터공학부 교수
- ORCID: http://orcid.org/0000-0001-8776-5185

· Received: Nov. 02, 2021, Revised: Jan. 14, 2022, Accepted: Jan. 17, 2022
· Corresponding Author: Jinho Ahn
School of AI Computer Science and Engineering, Kyonggi University,
154-42 Gwanggyosan-ro, Yeongtong-gu, Suwon-si, Gyeonggi-do 16227, Korea,
Tel.: +82-31-249-9674, Email: jhahn@kgu.ac.kr

Ⅰ. 서 론

검사점 기록 기반 복귀회복 기법은 여러 개의 프로세스들로 구성된 분산시스템을 위한 결함포용성을 제공하는 잘 알려진 경량의 도구이다[1]. 초창기에 개발된 두 종류의 기법은 비조정형 검사점 기록(Uncoordinated checkpointing)과 조정형 검사점 기록(Coordinated checkpointing) 기법이다[2]. 조정형 기법은 검사점을 취할 때마다 전역 검사점(Global checkpoint)의 일관성을 보장함으로써 정상 수행 시 발생하는 비용이 높으나, 회복 과정이 간단하고 연속적인 복귀(Cascading rollback)[1]가 발생하지 않는다.

반면 비조정형 기법은 각 프로세스가 검사점 취할 때 전역 검사점 일관성을 위한 다른 프로세스와의 동기화가 필요 없어 정상 수행 시 발생 비용이 적으나, 회복 시 복귀할 일관된 전역 검사점 검색 시간 및 비용이 높고 도미노 현상(Domino effect)[2]이 발생할 수 있다. 이후 이들의 장단점을 보완하기 위해 통신 유도 검사점 기록(Communication-induced checkpointing) 기법이 개발되었다[1]-[5]. 이는 이미 독립적으로 취해진 기본 검사점(Basic checkpoint)이 해당 프로세스 고장 시 하나 이상의 일관된 전역 검사점에 포함되어 필요 없지 않도록 강제적 검사점(Forced checkpoint)을 취하면서 검사점 기록에 대한 베스트-에포트(Best-effort) 자취성을 가진다.

이 기법에 대한 대부분의 이전 연구들은 강제적 검사점 수를 최소화하려는 궁긍적 목표를 달성하기 위해 진행되어왔다. 대표적으로 HMNR 프로토콜[2]은 각 송신 메시지 내에 제어 정보 벡터를 포함시켜 이를 이용하여 불필요한 강제적 검사점 수를 가능한 줄이고자 하였다. 그 이후 게으른 인덱싱(Lazy indexing) 기법을 사용하여 일부 특정한 경우에 발생할 수 있는 많은 강제적 검사점 수를 줄일 수 있는 개선된 HMNR 프로토콜 LazyHMNR[3]이 제안되었다. 그러나 이러한 HMNR 계열 프로토콜에서도 한 애플리케이션 수행 시 각 프로세스가 취하는 추가적인 강제적 검사점 수가 평균적으로 기본 검사점 수의 두 배 이상이 될 수 있다[1].

기존 프로토콜의 이러한 성능적 한계는 다음과 같이 두 가지 문제점에 기인한다. 첫 번째로 이들은 메시지를 수신하는 각 프로세스가 비인과형(Non-causal) Z-경로[1]에 대한 충분한 관련 정보를 신속하게 얻기 어렵다. 두 번째로 메시지를 수신한 각 프로세스가 해당 메시지 송신자에 대한 회복가능성을 알 수 없다. 최근에 이러한 문제점을 각각 해결하여 강제적 검사점 수를 상당히 줄일 수 있는 두 개의 상호보완적인 통신 유도 검사점 기록 프로토콜이 개발되었다. 첫 번째 프로토콜[4]은 메시지를 송신한 각 프로세스가 추가 메시지 발생 없이 기존 기법들에 비해 메시지 수신자의 최신 타임스탬프 값을 보다 이른 시점에 알 수 있도록 하는 개선된 송신자 기반 메시지 로깅(Sender-based message logging) 기법을 포함한다. 두 번째 프로토콜[5]은 각 수신자가 메시지 송신자의 회복가능성을 확인할 수 있게 함으로써 부분 결정적(PWD, PieceWise-Deterministic) 모델을 가정하지 않고 메시지 로깅을 활용할 수 있다.

본 논문에서는 이러한 두 프로토콜의 주요 비용 경감기법들을 결합하여 어떠한 필요 없는 기본 검사점도 허용하지 않으며 그 비용을 크게 떨어뜨릴 수 있는 효율적인 프로토콜을 제안한다.

Ⅱ. 관련연구

FINE 프로토콜[6]은 HMNR 프로토콜이 보유한 변수만으로 완화된 검사점 일관성 조건을 적용하여 HMNR보다 적은 수의 강제적 검사점을 취할 수 있게 개발되었다. 또한 LazyFINE 프로토콜[7]은 게으른 인덱싱 기법을 FINE에 적용하여 일부 프로세스들의 타임스탬프 값의 빠른 증가를 억제함으로써 불필요한 강제적 검사점 수를 줄일 수 있다. 그러나 이후 문헌에서 두 프로토콜은 모두 Z-일관성 타임스탬핑 규칙을 준수하지 못할 수 있다는 것이 증명되었다[1]. 즉, 이미 취해진 기본 검사점들 중 일부가 어떠한 일관된 전역 검사점 집합에도 포함되지 않아 회복 시 불필요한 기본 검사점이 될 수 있다.

MP 프로토콜[8]은 HMNR을 분산 DBMS에 적합하게 변형한 것으로 완전 커밋 트랜잭션들의 상태만을 안전한 저장소에 기록함으로써 추가적인 강제적 검사점 수를 줄일 수 있다.

VMMP 프로토콜[9]은 HMNR을 자율형 웹 서비스[10]에 적합하게 변형한 것으로 서비스 품질 파라메타와 현재 적용되는 정책에 따라 강제적 검사점

기록 여부를 결정할 수 있도록 개발되었다.

그러나 이러한 모든 프로토콜은 앞 절에서 언급한 HMNR의 성능적 한계를 동일하게 내포하고 있다.

III. 제안 프로로콜

본 논문에서 제안하는 통신 유도 검사점 기록 프로토콜 SynergyCIC는 HMNR 계열 프로토콜에 비해 추가적인 애플리케이션 메시지 발생비용 없이 다음과 같은 특징을 보유하도록 설계하였다:

1. 메시지 수신으로 인해 이미 취해진 임의의 기본 검사점이 어떠한 일관된 전역 검사점에 포함되지 않을 수 있는 경우에 강제적 검사점을 취한다.
2. 하나의 메시지 송신자가 해당 메시지에 대한 송신자 기반 메시지 로깅과정에서 수신자의 현재 타임스탬프 값을 수집하게 한다.
3. 송신자 기반 메시지 로깅을 포함하고 있지만 PWD 모델에 대한 제약을 받지 않게 한다.
4. HMNR과 LazyHMNR이 하나의 메시지를 수신한 프로세스가 강제적 검사점을 취해야 한다고 결정하는 경우에 대해서도, 그 메시지 생성 직전의 송신자 상태가 회복 가능한 것으로 식별된다면 강제적 검사점을 취하지 않는다.

먼저, 특징 1을 만족하기 위해 HMNR(LazyHMNR 포함) 프로토콜에서 정의한 다음과 같은 동일한 변수를 포함한다.

- ts_p: 프로세스 p의 현재의 타임스탬프 값을 유지하는 변수이다. 초기값은 0이며 p가 검사점을 취할 때마다 이 변수 값은 1씩 증가하고, 해당 검사점의 타임스탬프 값으로 설정된다. 각 메시지 m이 송신 될 때, 이 변수 값은 m에 포함되어 전달된다. p가 메시지 m'을 수신할 때 (ts_p<m'.ts)이면 ts_p를 m'.ts로 갱신한다.
- $greater_p$: p가 알고 있는 q의 타임스탬프 최근 값이 ts_p보다 작은가('T')를 나타내기 위한 부울형 변수 $greater_p[q]$로 구성된 벡터이다. $greater_p[p]$는 'F'로, $\forall q(p\neq q)$:$greater_p[q]$는 'T'로 초기화된다. p가 검사점을 취할 때마다 $\forall q(p\neq q)$:$greater_p[q]$는 'T'로 재설정된다. p가 메시지 m을 송신할 때, 이 벡터는 m에 포함되어 전달된다. p가 m'을 수신할 때 (ts_p<m'.ts)이면 $\forall q(p\neq q)$:$greater_p[q]$를 m'.greater[q]로 갱신한다. 만약 (ts_p=m'.ts)이면 $\forall q(p\neq q)$: $greater_p[q]$를 ($greater_p[q]\wedge$m'.greater[q])로 갱신한다.
- $sent_to_p$: p로부터 q로의 비인과형 Z-경로를 감지하기 위한 부울형 변수 $sent_to_p[q]$로 구성된 벡터이다. $\forall q$:$sent_to_p[q]$는 'F'로 초기화되고, p가 최신 검사점을 취한 직후 q에게 첫 번째 메시지를 송신한다면 'T'로 설정된다. p가 검사점을 취할 때마다 $\forall q(p\neq q)$:$sent_to_p[q]$는 'F'로 재설정된다. p가 메시지 m'을 수신할 때 $sent_to_p[q]$가 'T'인 어떤 q가 존재한다면, p의 최신 검사점으로부터 q를 거쳐 m'에 이르는 비인과형 Z-경로[1]가 있다는 것을 의미한다. 이때 조건 C1:$\exists q(1\leq q\leq n)$:$sent_to_p[q]\wedge$m'.greater[q]$\wedge$(m'.ts>ts_p))을 만족한다면, 그 비인과형 Z-경로가 하나 이상의 Z-사이클[1]을 포함할 수 있다는 것을 의미한다. 이 경우 p는 강제적 검사점을 취하여 특징 1을 만족시킨다.
- $ckpt_p$: p가 알고 있는 q의 현재까지 검사점 기록 수를 저장하는 변수 $ckpt_p[q]$로 구성된 벡터이다. $\forall q$:$ckpt_p[q]$의 초기값은 0이다. p가 검사점을 취할 때마다 $ckpt_p[p]$는 1씩 증가하고, 메시지 m을 송신할 때, 이 벡터는 m에 포함되어 전달된다. p가 m'을 수신할 때 ($ckpt_p[p]$=m'.ckpt[p])이면, p의 현재 검사점 이후에 p가 송신한 하나의 메시지부터 m' 수신으로 연결되는 인과형 Z-경로가 하나 이상 존재함을 의미한다. $\forall q(p\neq q)$:($ckpt_p[q]$<m'.ckpt[q])이면 $ckpt_p[q]$가 m'.ckpt[q]로 갱신된다.
- $taken_p$: p가 알고 있는 q의 최신 검사점으로부터 p의 다음 검사점에 이르는 인과형 Z-경로를 감지하기 위한 부울형 변수 $taken_p[q]$로 구성된 벡터이다. $taken_p[p]$는 'F'로 초기화되고 p가 검사점을 취할 때마다 $\forall q(p\neq q)$:$taken_p[q]$는 'T'로 설정된다. 메시지 m을 송신할 때, 이 변수는 m에 포함되어 전달된다. p가 m'을 수신할 때 조건 C_2:($ckpt_p[p]$=m'.ckpt[p])$\wedge$m'.taken[p]='T')을 만족한다면, p의 다음 검사점으로의 인과형 Z-경로가 하나 이상의 Z-사이클을 포함한다는 의미다. 이 경우 p는 강제적 검사점을 취하여 특징 1을 만족시킨다. $\forall q(p\neq q)$:($ckpt_p[q]$<m'.ckpt[q])이면 $taken_p[q]\leftarrow$m'.taken

[q]이고, $\forall q(p \neq q)$:($ckpt_p[q]=m'.ckpt[q]$)이면 $taken_p[q] \leftarrow taken_p[q] \wedge m'.taken[q]$이다.

따라서 SynergyCIC는 식 (1)에서 정의된 조건 C_{FC}를 시행함으로써 특성 1을 만족한다.

$$C_{FC} = C_1 \vee C_2 \quad (1)$$

3.1 개선된 송신자 기반 메시지로깅(AdvSBML)

프로토콜 특징 2를 만족하기 위해 설계된 기법 AdvSBML은 다음과 같이 동작 한다:

1. 프로세스 p가 메시지 m을 다른 프로세스 q에게 송신하고자 할 때, p는 자신이 송신한 최근 메시지의 송신일련번호(SSN: Send Sequence Number) 변수 SSN_p 값을 1 증가시킨다.
2. p는 m의 수신자 식별자(RID) 및 SSN, 수신일련번호(RSN: Receive Sequence Number), 데이터(data)로 구성된 m을 위한 로그원소 lge_m을 자신의 휘발성 메모리에 저장한다. 이 때, m.RSN은 유효하지 않은 값 0으로 설정한다. 또한 p는 m에 (SSN_p, data, ts_p)를 포함시켜 q에게 송신한다.
3. q는 자신이 수신한 최근 메시지의 RSN 변수 RSN_q 값을 1 증가시킨 후 m의 RSN으로 설정한다. 이때, (ts_q<m.ts)인 경우 ts_q를 m.ts로 갱신한다. 또한 응답 제어메시지 ACK_m에 (RSN_q, ts_q)를 포함시켜 p에게 전달한다. 이후 호출되는 메시지 송신 오퍼레이션은 버퍼에 대기시킨다.
4. p가 ACK_m을 수신할 때, lge_m.RSN을 ACK_m.RSN으로 설정한다. 또한 ts_p < ACK_m.ts인 경우 ts_p를 ACK_m.ts로 갱신한다. 이후 확인 제어메시지 CFM_m에 ACK_m.RSN을 포함시켜 q에게 송신한다.
5. CFM_m을 전달받은 q는 p의 타임스탬프 값이 자신의 타임스탬프 값 이상이라고 표기한다($greater_q[p] \leftarrow F$). 또한 m의 수신 직후부터 다음 수신 메시지 m_α 이전까지 버퍼에 대기한 모든 메시지 송신 오퍼레이션을 실행가능하도록 한다.

AdvSBML이 기존 프로토콜보다 어떻게 강제적 검사점 수를 줄일 수 있는지를 그림 1의 예로 설명하고자 한다. 그림 1에서 p가 r로부터 m_α를 수신한 경우, m_α.ts(=4)<ts_p(=5)인 상태이다.

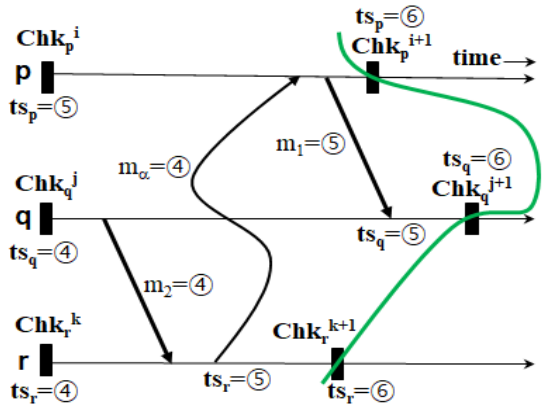


그림 1. AdvSBML이 강제적 검사점 기록을 생략하는 예
Fig. 1. Example of AdvSBML omitting forced checkpointing

p는 $ACK_{m\alpha}$에 (RSN_p, ts_p)를 포함시켜 r에게 전달한다. 이때, ($ACK_{m\alpha}$.ts > ts_r(=4))이기에 r은 $ts_r \leftarrow ACK_{m\alpha}$.ts(=5)과 $greater_r[p] \leftarrow$'F', $greater_r[q] \leftarrow$'T'를 수행한다. r로부터 $CFM_{m\alpha}$을 전달받은 p는 $greater_p[r]$을 'F'로 갱신한다. q가 p로부터 m_1을 수신할 때, $sent_to_q[r]$='T'와 m_1.ts(=5)>ts_q(=4)이지만 HMNR 계열 프로토콜과 달리 m_1.greater[r]='F'이기 때문에 C_1을 만족하지 않아 강제적 검사점을 취하지 않는다. r이 (k+1)번째 검사점 Chk_r^{k+1}을 취하더라도 AdvSBML에 의해 검사점의 타임스탬프 값이 6이 되어 그림 1과 같이 p와 q가 각각 m_1의 송신과 수신 이후에 취해진 Chk_p^{i+1}와 Chk_q^{j+1}과 함께 일관적인 회복라인(Consistent recovery line)[11]을 구성하게 된다.

3.2 회복가능성 및 비결정성 확인(RNDCheck)

한 프로세스는 처음에 일정기간 동안 결정적 계산을 수행하다가 비결정적(Non-deterministic) 이벤트가 발생한 후 비결정적 수행구간으로 전환된다. 비결정적 이벤트는 크게 두 가지 즉, 로깅 가능한 형태와 불가능한 형태로 나누어진다. 첫 번째는 프로세스 회복 시 고장 전과 동일한 위치에서 재연이 가능한 이벤트이다. 이러한 예로는 메시지 수신 및 소프트웨어 인터럽트, 시그널 등이 있다. 두 번째는 프로세스 고장 시 동일하게 반복적인 수행을 지원하지 못하는 이벤트이다.

특징 3과 4를 만족하기 위해 두 번째 기법 RNDCheck를 위해 각 프로세스마다 유지해야 변수는 다음과 같다.

- $NDEvInfo_p$: p가 알고 있는 q의 최근 송신 메시지의 SSN과 로깅 불가능한 비결정적 내부 이벤트 발생여부(ND)를 함께 저장하는 변수 $NDEvInfo_p[q]$로 구성된 벡터이다. ∀q:$NDEvInfo_p[q]$의 두 원소는 (0,'F')로 초기화되고, m을 송신할 때 $NDEvInfo_p[p].SSN$은 1 증가한 후 벡터를 m에 포함시켜 전달한다. p가 검사점을 취할 때마다 $NDEvInfo_p[p].ND$를 'F'로 재설정한다. p가 q로부터 메시지 m'을 수신할 때 ($NDEvInfo_p[q].SSN<m'.NDEvInfo[q].SSN$)이면, $NDEvInfo_p[q]$를 m'.NDEvInfo[q]로 갱신한다.
- $ExMod_p$: p를 포함한 임의의 프로세스가 자신의 최신 검사점 이후에 최소한 하나의 로깅 불가능한 비결정적 이벤트를 수행했는지를 표기하기 위한 부울형 변수이다. 'F'로 초기화되고, m을 송신할 때 m에 포함시켜 전달한다. ($ExMod_p$='T')이고 ∀q:($NDEvInfo_p[q].ND$='F')이면 $ExMod_p$를 'F'로 재설정한다.

RNDCheck는 이러한 프로세스별 변수를 활용하여 다음과 같은 방식으로 동작 한다:

1. 프로세스 p는 최초 검사점 Chk_p^0를 취하고 ($NDEvInfo_p[p].ND$←F) 결정적 모드($ExMod_p$←F)로 수행을 시작한다. p에서 임의의 로깅 불가능한 내부 이벤트가 발생한다면($NDEvInfo_p[p].ND$←T), 비결정적 모드($ExMod_p$←T)로 전환된다. 이후에 p가 다음 검사점 Chk_p^1을 취한다면($NDEvInfo_p[p].ND$←F) 다시 결정적 모드로 전환된다($ExMod_p$←F).
2. p가 검사점 Chk_p^1이후에 현재 비결정적 모드인 프로세스 q로부터 메시지 m을 수신한다면 (m.NDEvInfo[q].ND=T, m.ExMod=T), p는 q에서 임의의 로깅 불가능한 내부 이벤트가 발생한 것으로 간주하고($NDEvInfo_p[q].ND$←T) 비결정적 모드($ExMod_p$←T)로 접어든다. 이후에 p가 다음 검사점 Chk_p^2를 취한다 할지라도($NDEvInfo_p[p].ND$←F) q의 상태가 재연 가능한 상태인지 p가 알 수 없기 때문에($NDEvInfo_p[q].ND$=T) 자신의 모드를 비결정적으로 유지한다($ExMod_p$=T). 만약 q가 새로운 검사점을 취하고($NDEvInfo_q[q].ND$←F) 결정적 모드로 전환한 후($ExMod_q$←F) 다른 메시지 m'을 p에게 전달한다면, p는 q의 상태를 갱신하고 ($NDEvInfo_p[q].ND$←F) 자신의 모드를 결정적으로 전환한다($ExMod_p$←F).
3. p의 검사점 기록 및 통신패턴에 따른 실행이 완료될 때까지 1~2에서 언급한 수행모드변환규칙이 시행된다.

RNDCheck이 기존 프로토콜보다 강제적 검사점 수를 줄일 수 있는 방법을 그림 2의 예로 설명하고자 한다. 그림 2에서 세 프로세스 p, q, r의 각 검사점 (Chk_p^i,Chk_q^j,Chk_r^k) 직후 SSN이 a, b, c라고 가정한다. 이후 q가 r에게 m_2를 송신할 때, q의 수행모드는 결정적이고($NDEvInfo_q[q].ND$=F,$ExMod_q$=F),$NDEvInfo_q[q].SSN$가 1 증가하여 m_2의 SSN은 (b+1)이 된다. 또한 q는 AdvSBML의 로깅과정을 수행한다. 이때 $ExMod_q$과 $NDEvInfo_q$는 m_2에 포함되어 r에게 전달된다. r은 ($ExMod_r$←$ExMod_r$∨m_2.ExMod)를 수행하고 ∀j(j≠r):($NDEvInfor[j].SSN<m_2.NDEvInfo[j]$. SSN)이면 NDEvInfor[j]를 m_2.NDEvInfo[j]로 갱신한다. p가 q에게 m_1을 송신할 때, ($NDEvInfo_p[p]$= (a+1,F))이고 ($ExMod_p$=F)이다. q가 m_1을 수신할 때, C_1=sent_$to_q[r]$∧m_1.greater[r]∧($m_1.ts>ts_q$)=T이지만 m_1.ExMod=F이므로 q는 p가 회복 시 AdvSBML에 의해 m_1을 동일하게 재연할 수 있다는 것을 확인하고 강제적 검사점을 취하지 않는다. 이때 Chk_r^{k+1}는 (Chk_p^i,Chk_q^j)과 함께 일관적인 회복라인을 구성하게 된다.

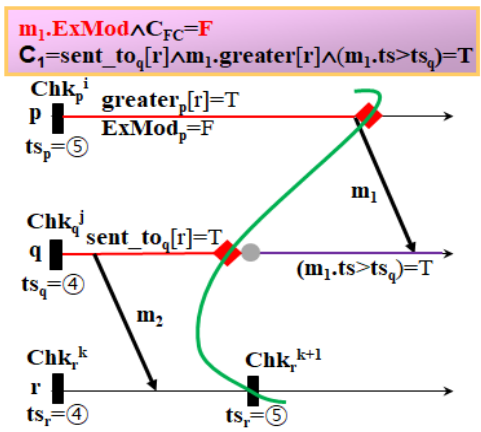


그림 2. RNDCheck이 강제적 검사점 기록을 생략하는 예
Fig. 2. Example of RNDCheck omitting forced checkpointing

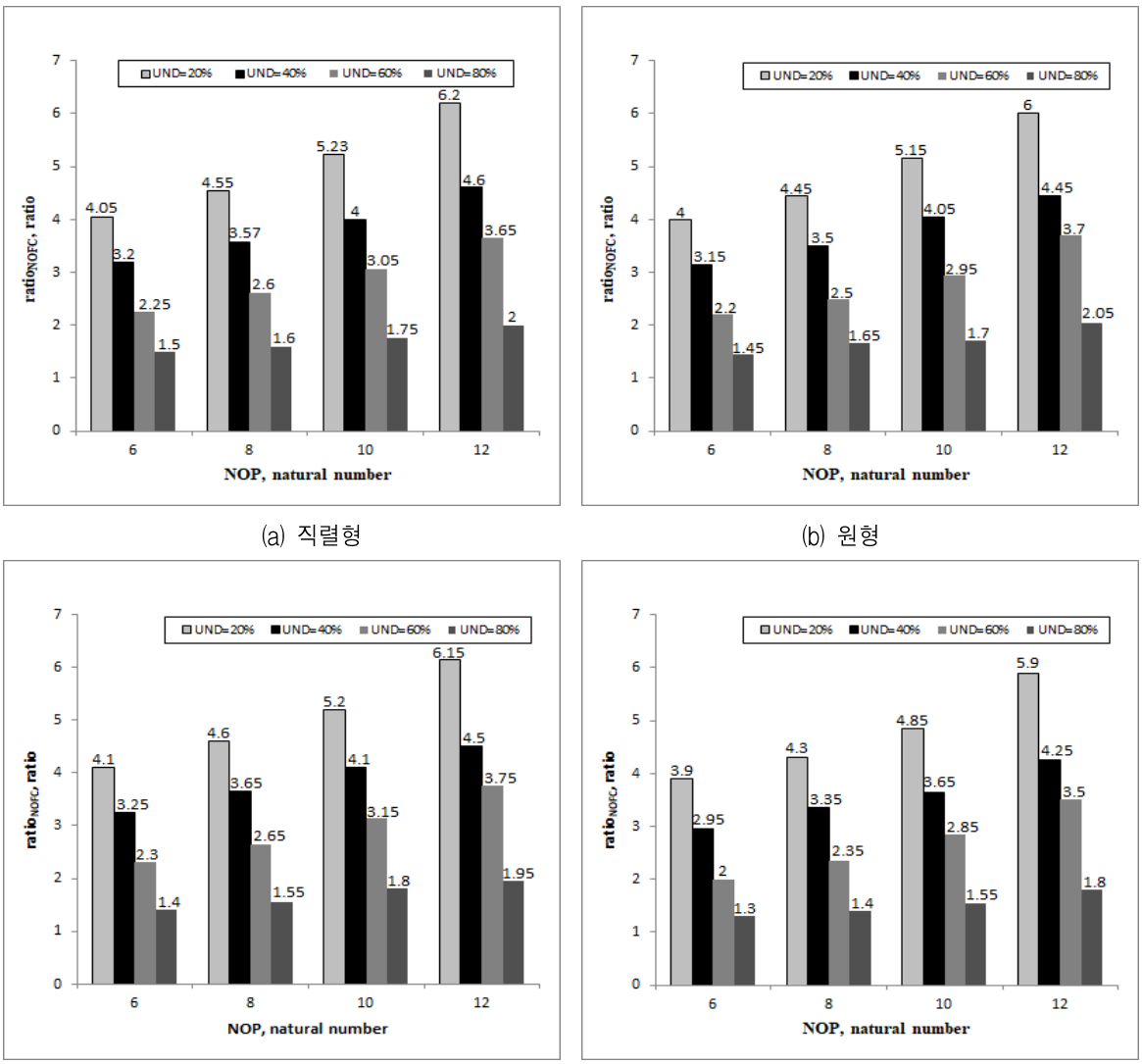
Ⅳ. 성능평가

본 절에서는 이산이벤트 시뮬레이션 언어[12]를 사용하여 HMNR 계열 프로토콜 중 특징 1을 만족하는 현재까지 가장 효율적인 프로토콜인 LazyHMNR과 본 논문에서 제안한 프로토콜 SynergyCIC의 성능을 비교하고자 한다. 성능평가를 위한 주요 성능 지표는 SynergyCIC가 발생시키는 강제적 검사점 수에 대한 LazyHMNR의 강제적 검사점 수의 비율인 $ratio_{NOFC}$이다.

시뮬레이션되는 시스템은 하나의 브로드캐스트 네트워크에 연결된 N개의 노드들로 구성되어 있다. 각 노드마다 하나의 프로세스가 수행되고, 모든 프로세스의 수행이 함께 시작되고 완료되도록 설계되었다. 한 프로세스로부터 송신되는 각 애플리케이션 메시지의 목적지는 단일 프로세스이다.

네트워크 링크 대여폭은 100Mbps이고 전송지연시간은 1ms이다. 애플리케이션 메시지 크기의 범위는 1KB~100KB이다. 각 프로세스는 지수분포에 따라 평균 300초 간격으로 기본 검사점 기록과정을 수행한다. 또한, 각 메시지는 지수분포에 따라 평균 3초 간격으로 임의로 선택된 프로세스로부터 송신된다.

시뮬레이션을 위해 네 가지 통신패턴인 직렬형, 원형, 계층형, 불규칙형 분산애플리케이션[13]이 사용되었다.



(a) 직렬형 (b) 원형

(c) 계층형 (d) 불규칙형

그림 3. $Ratio_{NOFC}$에 대한 성능 평가 결과

Fig. 3. Performance evaluation results on $ratio_{NOFC}$, (a) Serial, (b) Circular, (c) Hierarchical, (d) Irregular

그림 3은 한 프로세스에서 발생하는 이벤트 중 로깅 불가능한 비결정적 이벤트 비율(UND)이 20%, 40%, 60%, 80%이고 프로세스 수(NOP)의 범위가 6~12일 때, 네 가지 통신패턴 애플리케이션 수행에 따른 각각의 $ratio_{NOFC}$에 대한 실험결과이다.

NOP가 증가함에 따라 C_1 혹은 C_2를 만족하는 비인과형 Z-경로들이 형성될 가능성이 비례적으로 높아지기 때문에, $ratio_{NOFC}$는 네 가지 모든 통신패턴에서 상승한다. 또한 프로세스별 로깅 불가능한 비결정적 이벤트 발생이 적을수록 LazyHMNR의 강제적 검사점 기록 비용을 크게 절감할 수 있다.

이러한 결과는 다음과 같이 두 가지로 분석된다. 첫 번째는 AdvSBML의 수행을 통해 SynergyCIC는 각 프로세스가 시작과 마지막 검사점의 타임스탬프 값을 LazyHMNR보다 신속히 획득함으로써 강제적 검사점 기록과정을 생략하는 횟수가 증가하기 때문이다. 두 번째는 프로세스마다 증가하는 결정적 수행구간에서 형성되는 C_1 혹은 C_2를 만족하는 비인과형 Z-경로들을 RNDCheck가 식별함으로써 LazyHMNR과 달리 강제적 검사점 기록과정을 생략하는 횟수가 많아지기 때문이다.

이러한 두 기법은 통신패턴에 관계없이 강제적 검사점 수의 감소에 큰 효과가 있는 것을 알 수 있다. 다만, 불규칙형 통신패턴에서 통신의 불칙성으로 인해 Z-사이클의 형성 가능성이 수행 시마다 달라질 수 있기 때문에 그 효율성의 등락이 발생할 수 있다.

V. 결 론

본 논문에서는 두 기법 AdvSBML과 RNDCheck을 결합하여 어떠한 필요 없는 기본 검사점도 허용하지 않으며 강제적 검사점 기록 비용을 크게 떨어뜨릴 수 있는 통신 유도 검사점 기록 프로토콜 SynergyCIC를 제안하였다.

AdvSBML은 메시지를 송신한 각 프로세스가 추가 메시지 발생 없이 기존 기법들에 비해 메시지 수신자의 최신 타임스탬프 값을 보다 이른 시점에 알 수 있게 한다. RNDCheck는 각 수신자가 메시지 송신자의 회복가능성을 확인할 수 있게 함으로써 PWD 모델을 가정하지 않고 AdvSBML을 활용할 수 있다.

시뮬레이션은 강제적 검사점 수의 측면에서 애플리케이션 통신패턴에 관계없이 최근 기법에 비해 두 기법의 시너지로부터 얻어지는 비용절감효과가 1.4배~6.2배임을 보여준다.

References

[1] I. C. Garcia, G. M. D. Vieira, and L. E. Buzato, "A rollback in the history of communication-induced checkpointing", submitted(arXiv:1702.06167 [cs.DC]), Jun. 2019.

[2] J. M. Helary, A. Mostefaoui, R. H. B. Netzer, and M. Raynal, "Communication-based prevention of useless checkpoints in distributed computations", Distrib. Comput., Vol. 13, No. 1, pp. 29-43, Jan. 2000. http://dx.doi.org/10.1007/s004460050003.

[3] J. Tsai, "Applying the fully-informed checkpointing protocol to the lazy indexing strategy", J. Inf. Sci. Eng., Vol. 23, No. 5, pp. 1611-1621, Sep. 2007.

[4] J. Ahn, "Enhanced Sender-Based Message Logging for Reducing Forced Checkpointing Overhead in Distributed Systems", IEICE TRANSACTIONS on Information and Systems, Vol. E104-D, No. 9, pp. 1500-1505, Sep. 2021. http://dx.doi.org/10.1587/transinf.2021EDL8027.

[5] J. Ahn, "Communication-Induced Checkpointing with Message Logging beyond the Piecewise Deterministic (PWD) Model for Distributed Systems", Electronics, Vol. 10, No. 12, pp. 1-16, Jun. 2021. https://doi.org/10.3390/electronics10121428.

[6] Y. Luo and D. Manivannan, "FINE: A Fully Informed aNd Efficient communication-induced checkpointing protocol for distributed systems", J. Parallel Distrib. Comput., Vol. 69, No. 2, pp. 153-167, Feb. 2009. https://doi.org/10.1016/j.jpdc.2008.07.012.

[7] Y. Luo and D. Manivannan, "Theoretical and experimental evaluation of communication-induced checkpointing protocols in FE and FLazy−E", Perform. Eval., Vol. 68, No. 5, pp. 429–445, May 2011. https://doi.org/10.1016/j.peva.2011.01.005.

[8] H. Mansouri and A. Pathan, "A communication-induced checkpointing algorithm for consistent-transaction in distributed database systems", In Proceedings of the International Symposium on Security in Computing and Communication, Chennai, India, pp. 21-32, Oct. 2020. http://dx.doi.org/10.1007/978-981-16-0422-5_2.

[9] M. Vargas-Santiago, L. Morales-Rosales, R. Monroy, S. Pomares-Hernandez, and K. Drira, "Autonomic web services based on different adaptive quasi-asynchronous checkpointing techniques", Appl. Sci., Vol. 10, No. 7, pp. 2495-2515, Apr. 2020. https://doi.org/10.3390/app10072495.

[10] S. Lee and Y. Lee, "Implementation of Responsive Web Application for Location-based Semantic Search", Journal of KIIT, Vol. 17, No. 5, pp. 1-12, May 2019. http://dx.doi.org/10.14801/jkiit.2019.17.5.1.

[11] H. Chin and J. Ahn, "Duplicated Control Data Purging Algorithms for SBML Protocol Tolerating Temporary Communication Errors", Journal of KIIT, Vol. 17, No. 5, pp. 21-27, May 2019. http://dx.doi.org/10.14801/jkiit.2019.17.5.21.

[12] R. Bagrodia, R. Meyer, M. Takai, Y. Chen, X. Zeng, J. Martin, and H. Y. Song, "Parsec: a parallel simulation environments for complex systems", Comput J., Vol. 31, No. 10, pp. 77-85, Oct. 1998. https://doi.org/10.1109/2.722293.

[13] G. R. Andrews, "Paradigms for process interaction in distributed programs", ACM Comput. Surv., Vol. 23, No. 1, pp. 49-90, Mar. 1991. https://doi.org/10.1145/103162.103164.

저자소개

안 진 호 (Jinho Ahn)

1997년 2월 : 고려대학교 컴퓨터학과(이학사)
1999년 2월 : 고려대학교 컴퓨터학과(이학석사)
2003년 2월 : 고려대학교 컴퓨터학과(이학박사)
2003년 3월 ~ 현재 : 경기대학교 AI컴퓨터공학부 교수
관심분야 : 분산 및 병렬처리시스템, 지능형 사물인터넷, 빅데이터 처리 플랫폼, 자율 컴퓨팅