DOI: 10.3745/KIPSTA.2008.15-A.2.111

이동 기기에 적합한 소프트웨어 에이전트 기반의 효율적 체크포인팅 기법

임 성 채†

요 약

이동 통신 시스템의 발전과 함께 여러 대의 이동 기기에서 동작하는 분산 응용의 필요성이 점차 커지고 있다. 모바일 기기의 고장이나 통신망 단절이 기존 고정 통신망에 비해 자주 발생하는 환경을 고려할 때 모바일 응용을 위한 복구 기법이 매우 중요하며, 중단된 응용의 재시작을 위해 체크포인팅이 널리 사용되고 있다. 본 논문에서도 이런 분산 응용의 복구를 위한 효율적 체크포인팅 기법을 제안한다. 제안된 기법에서는 MSS(Mobile Support Station)에서 동작하는 체크포인팅 에이전트라는 소프트웨어 에이전트를 사용한다. 이 에이전트는 R-distance (rollback-distance) 개념을 지원하며, 이를 통해 복귀되는 지역 체크포인트의 최대 개수를 한정할 수 있다. 제안된 방식은 기존의 문제점이었던 도미노 현상이나 체크포인트 유지에 필요한 추가 비용을 크게 줄이면서도 매우 유연한 방식의 체크포인트 생성을 지원할 수 있다.

키워드 : 모바일 통신, 분산 체크포인트, 복구기법

An Efficient Checkpointing Method for Mobile Hosts via the Software Agent

Lim Sung Chae†

ABSTRACT

With the advance in mobile communication systems, the need for distributed applications running on multiple mobile devices also grows gradually. As such applications are subject to H/W failures of the mobile device or communication disruptions, compared to the traditional applications in fixed networks, it is crucial to develop any recovery mechanism suitable for them. For this, checkpointing is widely used to restart interrupted applications. In this paper, we devise an efficient checkpointing method that adopts the software agent executed at the mobile support station. The agent, called the checkpointing agent, is aimed at supporting the concept of rollback-distance (R-distance) that bounds the maximum number of roll-backed local checkpoints. By means of the R-distance, our method can prevent undesirable domino effects and heavy checkpoint overhead, while providing high flexibility in checkpoint creation.

Keyword : Mobile Communication, Distributed Checkpoints, Recovery

1. 서 론

무선 통신망을 통한 인터넷 사용이 점차 일반화되고, 이를 지원할 수 있는 모바일 기기(host)의 형태도 다양화 고성능화되어 가고 있다. 이런 무선 통신 환경의 발전과 함께 모바일 기기를 사용한 분산 응용의 중요성도 증가하게 되었다[1-4]. 분산 응용이라 함은 여러 대의 모바일 기기에서 동작하는 AP(Application Process)들이 서로 통신을 하고 이들 간에 전송된 데이터에 의해 수행 결과가 결정되는 응용을 말한다. 분산 응용이 일정 시간 이상 수행될 때 고려되어야 할 것이 모바일 기기의 고장 및 무선 통신 해제에 따른 복구(recovery) 방식이다. 이를 위해 주로 사용되는 기법이 체크포인팅(checkpointing) 기법이며, 체크포인트 레코드를 안전한 기억 장소에 저장해 둠으로써 고장 발생시 복구 작업을 수행할 수 있다[5-13].

본 논문에는 이와 같은 무선 통신 환경에서의 분산 체크포인팅 기법에 대한 연구를 수행한다. 제안된 기법에서는 기존의 비동기식 체크포인팅 기법의 약점인 도미노(domino) 현상 및 낮은 유연성 문제를 방지하기 위해 체크포인팅 에이전트(agent)를 제안한다. 에이전트를 통해 각 AP들이 비동기적으로 수행하는 지역(local) 체크포인트 들이 일정한 개수 이상 복귀(rollback)되어 무용화되는 것을 방지할 수 있고, 에이전트 간의 유선 통신망을 사용하여 체크포인팅에 필요한 통신 추가비용을 최소화할 수 있다.

제안된 체크포인팅 에이전트는 분산 응용에 대해 주어진

※ 이 논문은 2006년도 동덕여자대학교 학술연구비 지원에 의하여 수행된 것임.
† 정 회 원 : 동덕여자대학교 컴퓨터전공 조교수
논문접수 : 2008년 1월 18일, 심사완료 : 2008년 2월 23일

R-distance(rollback-distance)을 지키도록 동작한다. 여기서 R-distance가 유지된다고 함은 각 지역 체크포인트가 특정 개 이상 무용화될 수 없음을 의미한다. 이런 최대 복귀 허용 거리를 정하는 R-distance라고 하는 개념을 도입함으로써 비동기식 체크포인팅 기법의 문제점인 도미노 현상을 피하면서도 각 AP들이 응용의 의미성에 따라 매우 유연한 방식으로 일관성 전역(global) 체크포인트를 생성할 수 있도록 했다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구로서 전역 일관성 상태 및 무선 통신 환경에 대해서 알아본다. 3장에서는 연구 동기 및 제안 기법의 세부 알고리즘을 기술한다. 4장에서는 제안된 기법의 성능 특성에 대해 알아보고, 5장에서 결론을 맺는다.

2. 관련 연구

2.1 전역 일관성 상태

체크포인팅 기법을 통한 복구 및 재시작이란, 서로 다른 모바일 기기의 AP들이 각자 자신의 작업 상태를 체크포인트 레코드로 기록해 두었다가 고장 발생 시에 기록된 내용을 사용하여 고장 시점 이전의 특정 상태에서부터 다시 빠르게 분산 응용을 수행할 수 있음을 의미한다[7,14-16]. 여기서 재시작 시점은 분산 응용의 관점에서 전역 일관성 상태(CGS: Consistent Global State)를 가지고 있어야 한다.

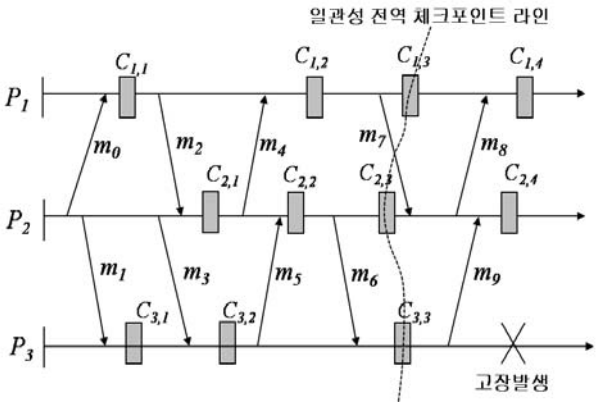
CGS를 정함에 있어서 Lamport[17] 제안한 이벤트 간 순서 결정 방식이 사용된다. 분산 응용에 참가한 AP들은 다른 AP와의 메시지 전송 및 수신이벤트, 그리고 자발적인 작업 수행 이벤트의 세 가지 이벤트를 수행하는 능동적 오브젝트로 모델링된다. 그리고 이들 이벤트 간의 발생 순서는 "happens before"에 따른다. 아래는 이런 HBR(Happens-Before Relation)에 대한 정의이다.

〖HBR 정의〗 임의의 두 이벤트 e_1과 e_2 사이에 다음의 어느 한 조건이 만족된다면 e_1 "happens before" e_2의 관계가 성립한다고 정의한다.

1. e_1과 e_2가 동일한 AP에서 발생한 이벤트이고 e_1이 e_2에 앞서 발생하였다.
2. 서로 다른 두 AP P_1과 P_2가 존재하고, P_1이 P_2로 메시지 m을 전송(e_1)하였고 P_2가 메시지 m을 수신(e_2)하였다.

Lamport는 공통의 타이머가 없는 분산 응용에서 HBR를 사용하여 이벤트 간 발생 순서를 정했으며, HBR의 이행성(transitive) 성질에 따라 이벤트들 간에 부분 순서가 정해진다.

CGS는 이런 HBR을 사용하여 정의할 수 있다. 즉, 임의의 분산 응용에 참여한 각 AP들이 생성한 지역 체크포인트의 집합이 전역 일관성을 갖기 위해서는 체크포인트에 기록된 모든 이벤트 e_j에 대해서, e_i "happens before" e_j 관계를 가지는 모든 이벤트 e_i 역시 체크포인트로 기록되어야 한다.



(그림 1) 분산 응용에서의 고장 및 전역 체크포인트의 예.

이런 전역 일관성을 가지는 지역 체크포인트들의 집합을 일관성 전역 체크 포인트라 부른다. *논문에서는 이런 일관성 전역 체크포인트를 간단히 줄여 전역 체크포인트라고 부르겠다.*

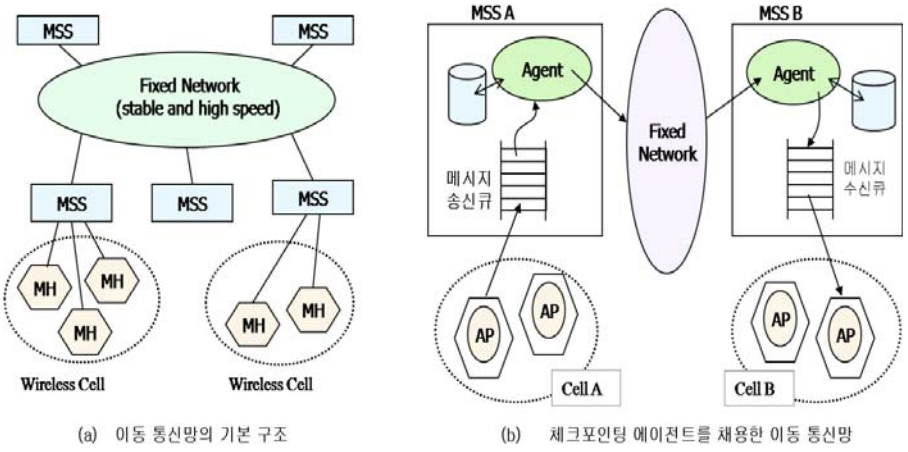
(그림 1)은 세 개의 AP P_1, P_2, P_3이 수행한 응용 및 관련된 전역 체크포인트를 보인 예이다. AP들 간의 메시지는 m_0에서 m_9로 표시되며, P_j가 생성한 지역 체크포인트는 순서에 따라 $C_{j,i}$로 표시했다. 그림은 P_3가 있는 기기에서 고장이 발생한 상황을 보이며, 이 예에서 가장 최근의 전역 체크포인트는 $C_{1,3}$, $C_{2,3}$, $C_{3,3}$를 포함한다. 그림에서 $C_{2,4}$가 제외된 것은 m_9의 전송 이벤트가 P_3의 지역 체크포인트에 포함되지 못했기 때문이다. $C_{2,4}$가 포함되지 못함으로써 m_8 메시지의 수신 이벤트를 기록한 $C_{1,4}$ 역시 전역 체크포인트에서 제외되었다.

2.2 무선 통신 환경에서의 체크포인팅 기법

2.2.1 이동 통신 환경

이동 통신 환경은 무선 셀(cell) 안의 모바일 기기인 MH(Mobile Host)와 무선 연결을 관리하고 통신하는 MSS(Mobile Support Station)와 MSS 간의 유선 통신망으로 구성된다[1,3,4]. MSS는 무선 통신 인터페이스를 통해 자신의 셀 안에 있는 MH들과 무선 통신 연결을 관리한다. (그림 2)(a)는 이런 일반적인 이동 통신 환경을 보인다. 또한 MSS는 MH들과의 통신을 위해 메시지 수신큐(inbound queue)와 송신큐(outbound queue)를 가지고 있다. MSS와 MH는 전체 시스템 범위에서 유일한 식별자를 가지며, 각 AP의 식별자는 자신이 동작하는 MH 식별자와 MH 내의 식별자 조합으로 구성된다.

논문의 시스템 역시 (그림 2)(a)와 같은 구조를 가정하며, 다른 점은 각 MSS에 체크포인팅 기능을 위한 별도의 소프트웨어 에이전트를 둔다는 점이다. 이를 (그림 2)(b)에 나타냈으며 그림에서는 셀 A에 있는 AP가 셀 B의 다른 AP로 메시지를 보내는 상황을 보인다. MSS에 존재하는 에이전트는 로깅 수행 및 전송되는 메시지에 체크포인팅 작업에 필요한 데이터를 덧붙이거나 빼는 작업을 수행한다. 이에 대



(a) 이동 통신망의 기본 구조 (b) 체크포인팅 에이전트를 채용한 이동 통신망

(그림 2) 논문에서의 이동 통신망 구조

한 내용은 3 장에서 기술된다.

2.2.2 기존 무선 체크포인팅 기법

분산 응용을 위한 체크포인팅 기법은 크게 동기식과 비동기식 기법으로 나눌 수 있다. 먼저 동기식 기법은 각 AP의 체크포인팅 요청 시점마다 전역 체크포인트가 생성되도록 강제하는 방식이다. 이를 위해서 응용에 참가하고 있는 모든 AP들로 지역 체크포인트 생성을 강제하는 메시지를 보내며 이에 대한 수행이 끝나면 다시 작업을 속개한다. 이런 동기식 알고리즘은 무선 통신망에서와 같이 통신 비용이 높고 AP들이 임시로 통신 연결을 해제할 수 있는 무선 통신 환경에서는 적용이 어렵다[5,6,8,12].

이런 이유로 무선 통신망에서는 비동기식 체크포인팅 기법이 선호된다. 이 기법에서는 AP의 체크포인트 생성 시에 의무적인 전역 체크포인트 생성은 없다. 따라서 복구 과정에서 흩어져 있는 지역 체크포인트 들을 분석하여 CGS를 이루는 전역 체크포인트를 정해야 하는 추가 비용이 있다. 또한 도미노(domino) 현상으로 인해 최근 상태를 반영한 전역 체크포인트를 정할 수 없는 문제도 있다[6,11,15].

이런 도미노 현상을 이해하기 위해 (그림 1)을 다시 사용한다. 이 그림에서 지역 체크포인트 $C_{2,3}$가 없다고 가정해 보자. 그러면 전체 10개의 지역 체크포인트 중에서 도미노 현상으로 인해 일관성 전역 체크포인트를 생성하는 집합을 구할 수 없다. 따라서 응용의 초기 상태에서부터 다시 재시작해야 하는 문제가 발생한다. 이런 도미노 현상을 막기 위해 무선 환경에서는 메시지 유도(message-induced)에 의한 체크포인트 생성 프로토콜이 사용되기도 한다[5,6,13].

이 방법의 아이디어는 다음과 같다. (그림 1)의 예에서 $C_{2,3}$이 없는 경우 도미노 현상이 발생하는데, 그 원인은 메시지 수신이나 송신 이벤트 사이에 체크포인트가 생성되지 않았기 때문이다. 이런 점에 착안하여 다른 AP로부터 메시지를 수신하는 경우 특정 조건에 따라 지역 체크포인트를 강제로 생성하는 방식이 메시지 유도 방식이다[6,15].

메시지 유도 방식에서 AP의 동작 상태를 SEND와 RECEIVE의 2-phase로 구분한다. 이 두 상태는 메시지의 전송 및 수신 이벤트에 의해 설정되며 SEND 상태에서의 메시지 전송 이벤트나, RECEIVE 상태에서의 메시지 수신 이벤트는 현재 상태를 그대로 유지시킨다. 강제적인 체크포인트 생성은 SEND 상태에서 메시지 수신 이벤트가 발생했을 때이다. 이런 메시지 유도 방식을 사용함으로써 도미노 현상의 문제를 피할 수 있다.

3. 제안하는 에이전트 기반 체크포인팅 기법

3.1 연구 동기

앞서 기술했듯이 무선 통신망에서는 비동기식 체크포인팅 기법이 일반적이며 메시지 유도에 따른 강제적 체크포인팅 프로토콜이 유용하다. 하지만 이 기법의 가장 큰 단점은 메시지가 수신되는 상황과 지역 체크포인트 생성이 연결되기 때문에 분산 응용의 진행 의미성과(semantics) 관련이 없이 체크포인트가 생성된다는 점이다.

각 AP 마다 중요한 작업을 수행하는 시점이 있고, 그런 중요 시점은 AP의 작업 진행 의미성에 의해 결정된다. 의미성을 반영한 체크포인팅이 되기 위해서는 AP들이 응용 수행 중에 중요 작업이 진행 중이라고 판단하면 자신의 작업 내용이 전역 체크포인트에 반영될 것을 강제 할 수도 있어야 하고, 그렇지 않은 경우라면 지역 체크포인트가 취소되는 것을 용인할 수도 있어야 한다. 이런 의미성을 고려한 체크포인팅은 기존의 메시지 유도에 의한 강제적 체크포인팅 기법에서는 지원할 수 없는 특성이다.

본 논문에서는 분산 응용의 진행 의미성을 유연하게 반영할 수 있는 체크포인팅 기법에 대해 연구한다. 제안된 방법은 체크포인팅 에이전트(Checkpointing Agent, 이하 CA)로 불리는 프로그램과 R-distance로 표현되는 최대 복귀 허용 거리 개념을 사용한다. CA는 자신과 연관된 AP가 지역 체크포인트를 생성할 때 마다 응용에 이미 부여되어 있는 R-distance의 값을 참조하여, 복귀될 수 있는 지역 체크포인트의 개수가 R-distance 값보다 작은지를 체크한다. 만약 이를 충족시키지 못한다면 R-distance를 만족 시킬 수 있도록

현재의 전역 체크포인트 시점을 좀 더 최근 것으로 앞당긴다. 그렇지 않고 조건을 충족하는 상황이라면 지역 체크포인트 생성을 알리는 메시지만을 다른 CA로 방송한다. 유연성 있는 전역 체크포인트 생성 기법을 통해 체크포인트 생성에 드는 비용을 최소화 하면서, AP들의 의미성에 따른 체크포인트 생성을 지원하고자 한다.

3.2 에이전트를 이용한 체크포인팅

제안하는 체크포인팅 기법은 응용의 중요도에 따라 응용을 시작한 AP가 R-distance로 양의 정수 값을 지정할 수 있다. R-distance는 고장 발생 시 복귀될 수 있는 지역 체크포인트의 최대 개수를 한정하는 값이며, 이 값이 작다면 진행된 작업의 중요도를 높게 보고 좀 더 최근 시점을 반영하는 전역 체크포인트가 생성된다.

(그림 3)은 위와 같은 특성을 지원하기 위한 CA 알고리즘이다. 알고리즘은 분산 응용이 시작된 후의 CA 동작을 나타내며, 편의상 알고리즘을 수행하는 CA는 CA_i로, 다른 CA들은 CA_j로, CA_i가 있는 MSS와 연결된 AP는 P_k로 표현한다. 체크포인트 요청시에 P_k에 의해 생성될 지역 체크포인트의 시리얼 번호는 $Serial_{P_k}$이며, 이에 대한 데이터는 메모리 상에서 관리된다. 물론, $Serial_{P_k}$ 보다 작은 시리얼 번호의 지역 체크포인트 들은 디스크에 저장되어 있다.

알고리즘은 크게 두 부분으로 구성된다. 첫 번째 부분인 1행에서 12행까지는 송신 (outbound) 메시지 처리이며, 이 메시지는 데이터 전송 메시지이거나 체크포인트 생성 요청 메시지 중 하나이다. 메시지가 데이터 전송을 위한 것이라면 3행과 4행에서와 같이 메모리에서 관리하는 체크포인트 레코드를 수정하고 전역 체크포인트 관리에 필요한 추가적인 필드를 더해서 상대편 CA_j로 메시지를 전송한다. 이때 사용되는 추가적인 컨트롤 필드들에 대해서는 3.3.1절에 기술된다. 또 알고리즘에 사용한 루틴들은 3.3.2절에서 기술한다.

체크포인트 요청의 경우라면 6행에서와 같이 *CalcCGSSerial*()을 사용하여 최근 전역 체크포인트에 속하는 지역 체크포인트의 시리얼을 구하고 이 값과 $Serial_{P_k}$ 값을 비교하여 R-distance가 유지되고 있는지를 조사한다. 만약 R-distance가 유지되지 않는 경우라면 8행에서 *ForcedCGS*()를 수행하여 생성 가능한 전역 체크포인트의 시점을 앞당긴다.

알고리즘의 두 번째 부분인 13행에서 22행에서는 수신 (inbound) 메시지의 종류에 따라 세 가지 작업 중 하나가 수행된다. 앞서 언급된 바와 같이 다른 AP로 부터의 메시지는 CA_j를 통해 CA_i로 수신된다. 14행에서와 같이 응용 데이터 전송을 위한 메시지인 경우에는 메시지에 첨가된 체크포인트 관련 정보를 반영하기 위해 *UpdateChptRec*()를 수행한다. 그리고 체크포인트 관련 필드를 삭제하고 갱신된 ***M***을 메시지 수신큐에 입력함으로써 P_k로 전송한다.

다음으로 메시지가 지역 체크포인트를 알리기 위해 방송된 메시지인 경우는 이를 반영하기 위해서 *UpdateChptRec*()를 수행한다. 이때의 메시지는 CA들에만 필요한 메시지이므로 P_k로는 전송되지 않는다. 마지막으로 CA_j가 체크포인트 생성을 강제하는 메시지를 보낸 경우라면 이를 P_k에게

```
1.   if( P_k로부터 메시지 M을 수신 )
2.     if( M은 데이터 전송을 위한 메시지 )
3.       P_k의 체크포인트 레코드 수정을 위해 UpdateChptRec(M)을 수행.
4.       메시지 M 에 체크포인팅 관련 필드를 추가하고, 수정된 메시지 M을 상대편 CA_j로 전송.
5.     else  /* M은 지역 체크포인트 생성을 요청하는 메시지 */
6.       gchpt = CalcCGSSerail(). /* 전역 체크포인트를 계산 */
7.       if( R-distacne < Serial_{P_k} − gchpt )
8.         ForceCGS()을 수행하여 전역 체크포인트를 생성.
9.       시리얼 번호 Serial_{P_k}를 가지는 지역 체크포인트를 디스크에 기록하고  Serial_{P_k}의 값을 1증가시킴.
10.      요청이 처리되었음을 알리는 응답메시지를 P_k로 보내고, 관계된 모든 CA들로 새로운 지역 체크포
         인트 시리얼을 저장한 메시지를 방송.
11.    end_if
12.  end_if
13.  if( CA_j로부터 메시지 M을 수신 )
14.    if( M은 데이터 전송을 위한 메시지 )
15.      P_k의 체크포인트 레코드 수정을 위해 UpdateChptRec(M)을 수행.
16.      메시지 M에서 체크포인트 관련 필드를 삭제하고, 수정된 M을 P_k로 보냄.
17.    else if( M은 지역 체크포인트 생성을 알리는 메시지 )
18.      P_k의 체크포인트 레코드 갱신을 위해 UpdateChptRec(M)을 수행.
19.    else /* M은  지역 체크포인트 생성을 강제하는 메시지 */
20.      P_k로 지역 체크포인트를 생성을 요청하는 메시지를 전송.
21.    end_if
22.  end_if
```

(그림 3) 메시지 M을 수신한 CA_i의 동작 알고리즘

알린다. 이에 따라 P_k는 자신의 상태 정보를 담은 체크포인팅 요청 메시지를 보내게 되므로, 이후 (그림 3)의 6행에서부터 수행될 것이다.

CA와 R-distance 개념을 사용한 체크포인팅 기법의 장점은 다음과 같다. 첫째 전역 체크포인트 생성에 있어 유연성이 확보된다. 전역 체크포인트에 포함될 각 AP들의 지역 체크포인트들은 R-distance내에서 자유롭게 선택될 수 있다. 이런 이유로 전역 체크포인트 생성 시에 시간 지연 및 비용을 최소화시킬 수 있는 전역 체크포인트의 선택이 가능하다. 만약, 특정 AP의 작업 내용이 매우 중요하여 반드시 전역 체크포인트에 포함시키고 싶다면, R-distance 개의 지역 체크포인트를 연속적으로 생성함으로써 이를 실현할 수 있다. 이때도 다른 AP들의 경우 가장 최근 지역 체크포인트가 포함될 필요는 없기에 유연성은 그대로 유지된다.

둘째는 유선 통신망을 최대한 이용할 수 있다는 점이다. CA들은 유선 통신망으로 연결되어 있기에 이들 간의 데이터 통신 비용은 상대적으로 매우 낮다고 할 수 있다. 이점을 이용하여 이들 간의 메시지 전송을 통해 전역 체크포인트를 구한다. 또한 체크포인팅에 필요한 추가 데이터들이 MSS와 MH간의 무선 통신에서는 사용되지 않기 때문에 무선 데이터 통신량을 크게 줄일 수 있다.

3.3 제안하는 알고리즘

3.3.1 메시지 형식(format)

세부 알고리즘을 위해서는 메시지 형식과 CA가 생성하는 체크포인트 레코드에 대한 설명이 필요하다. 아래는 메시지 ***M***을 정의하는 필드 이름과 간단한 설명이다. 아래 필드 중 뒤의 세 가지는 CA들 사이에서만 사용되는 필드들이며, 이 필드에는 체크포인팅에 사용되는 제어(control) 데이터가 저장된다. CA가 다른 CA로 메시지를 전송할 때 추가되었다가, 수신한 CA측에서 수신큐로 보내기 전에 제거됨으로 AP와 MSS간의 무선 통신에는 사용되지 않는다.

아래의 설명에서 편의상 AP P_s에서 AP P_r로 메시지가 전송됨을 가정한다.

- ***M***.mesg_type: 메시지 종류를 표시.
- ***M***.send_id: P_s의 식별자.
- ***M***.recv_id: P_r의 식별자.
- ***M***.data: P_s가 전송하는 응용 데이터.
- ***M***.join_ap[1,2, .., *N*]: *N*은 분산 응용에 참가한 AP 개수. join_ap[*j*]는 *j*번째 AP의 식별자 값.
- ***M***.chpt_serial[1, 2, ..., *N*]: AP들의 지역 체크포인트 생성 정보. chpt_serial[*j*]는 *j*번째 AP의 가장 최근 지역 체크포인트 시리얼 번호.
- ***M***.chpt_dep[1,2, .., *N*]: 체크포인트 의존관계 벡터.

아래는 임의의 AP P_k에 대해 생성되는 체크포인트 레코드에 대한 설명이다. 아직 디스크에 기록되지 않은 레코드는 메모리에 존재하며, 지역 체크포인트 생성시점에 디스크로 기록된다. 기록된 레코드들에 접근할 수 있도록 가장 뒤쪽 필드인 prev_chpt가 존재하고 이 필드를 통해 시간 역순으로 접근이 가능하다.

- ***CHPT***.join_ap[1, 2, ..., *N*]: join_ap[*j*]는 *j*번째 AP의 식별자 값.
- ***CHPT***.id: P_k의 식별자.
- ***CHPT***.serial: 저장된 체크포인트 레코드의 시리얼 번호.
- ***CHPT***.R-distance: 응용에 정해진 최대 복귀 허용 거리.
- ***CHPT***.chpt_serial[1, 2, ..., *N*]: AP들의 지역 체크포인트 생성 정보.
- ***CHPT***.chpt_dep[1, 2, ..., *N*]: 체크포인트 의존관계 벡터.
- ***CHPT***.sent_mesg[]: 이전 지역 체크포인트 생성 시점 이후 AP_k가 보낸 메시지 정보.
- ***CHPT***.sent_mesg_no: send_mesg[]에 기록된 메시지 개수.
- ***CHPT***.prev_chpt: 이전 지역 체크포인트 레코드의 디스크 저장 주소.

위의 chpt_dep[] 데이터는 전역 체크포인트를 결정할 때 반드시 필요한 정보로서 AP들의 생성하는 각 지역 체크포인트 간의 의존성을 나타낸다[6,9,13,15]. 예를 들어, chpt_dep[*j*]의 값이 *n*이라고 한다면, P_k의 현재 지역 체크포인트가 복귀되지 않기 위해서는 *j* 번째 AP의 시리얼 번호가 *n*인 지역 체크포인트가 디스크에 기록되어 있어야 함을 의미한다. 이 데이터는 기존 알고리즘에서 일반적으로 이용된 제어 데이터이며 논문에서는 이를 갱신하는 루틴으로 *UpdateChptRec*() 루틴을 사용하였었다.

3.3.2 세부 수행 알고리즘

본 절에서는 (그림 3)의 CA 알고리즘에서 이용한 루틴들에 대해 설명한다. 본 절에 기술된 알고리즘 외에도 AP의 메시지 송수신 루틴도 필요하지만 이에 대한 사항은 쉽게 유추될 수 있고 기존 논문[5,6,9]들에 서술되었기에 생략하기로 한다.

(그림 4)의 *CalcSerialCGS()*는 (그림 3)의 6행에서 사용되었으며 지역 체크포인트 중 가장 최근 전역 체크포인트에 포함된 것을 알려주는 루틴이다. 이 루틴을 통해 반환된 값과 현재의 지역 체크포인트 시리얼 값을 비교하여 복귀 거리를 판단할 수 있다. *Current*에는 디스크에 기록되기 전의 체크포인트 레코드가 저장되며 이 정보가 이후 시리얼 $Serial_{P_k}$을 가지는 체크포인트 레코드로 디스크에 저장된다. *CHPT*[2]에서 *CHPT*[*r_distance*]까지의 레코드는 이전에 생성한 지역 체크포인트를 디스크에 읽은 것이며, *CHPT*[1]은 *Current*와 동일한 데이터이다. 이들 체크포인트 레코드들을 사용하여 CGS 라인을 4행에서 11행까지 구하고 있다. 이때 사용되는 정보가 체크포인트 의존성이다. CGS 라인이 구해지지 않는 경우라면 12행이 수행되어 전역 체크포인트 생성이 강제될 것이다.

다음 (그림 5)에서는 주어진 R-distance를 강제하기 위한 *ForcedCGS*()의 알고리즘이다. 여기서도 (그림 4)에서의 기호를 사용한다. 이 루틴과 기존 기법과의 차이는 가장 최근

Return value: serial number of P_k's local checkpoint included in the most recent CGS line.

```
1.  r_distance = Current.R-distance. /* application-specific recover distance */
2.  Let CHPT[1], CHPT[2], ..., CHPT[r_distance] be the checkpoint records stored in the disk,
    which have serial numbers of Serial_Pk, Serial_Pk − 1,.., Serial_Pk − r_distance − 1, respectively.
3.  ap_num = number of APs in Current.join_ap[].
4.  for j = 1 to r_distance do
5.      cgs_exist = yes.
6.      for ap = 1 to ap_num
7.          if ( CHPT[j].chpt_dep[ap] > Current.chpt_serial[ap] )
8.              cgs_exist = no.
9.      end_for
10.     if( cgs_exist == yes ) then return Serial_Pk − j+1
11. end_for
12. return Serial_Pk − r_distance.
```

(그림 4) 루틴 *CalcSerialCGS*()를 위한 알고리즘

```
1.  r_distance = Current.R-distance. /* application-specific maximum rollback distance */
2.  Let CHPT[1], CHPT[2], ..., CHPT[r_distance] be the checkpoint records that have serial num-
    bers of Serial_Pk, Serial_Pk − 1, .., Serial_Pk - r_distance -1, respectively.
3.  ap_num = number of APs in Current.join_ap[].
4.  for j = 0 to r_distance
5.      needed_chpt_num[j] = 0.
6.      for ap = 1 to ap_num
7.          if( CHPT[j].chpt_dep[ap] > Current.chpt_serial[ap] )
8.              needed_chpt_num[j] = needed_chpt_num[j] + 1.
9.      end_for
10. end_for
11. cgs_line = 1.
12. for j = 1 to r_distance
13.     if( needed_chpt_num[j] + j < needed_chpt_num[cgs_line] + cgs_line )
14.         cgs_line = j.
15. end_for
16. forall ap having CHPT[cgs_line].chpt_dep[ap] > Current.chpt_serial[ap]
17.     Send a checkpoint creation message to a corresponding CA managing ap.
18. Wait until the checkpointing requests above are all served.
```

(그림 5) 루틴 *ForcedCGS*()을 위한 알고리즘

Routine *UpdateChptRec*(message ***M***)

```
1.  if( M is a message toward P_k ) /* inbound message */
2.      forall ap in Current.join_ap[]
3.          Current.chpt_serial[ap]=max(M.chpt_serial[ap], Current.chpt_serial[ap]).
4.          Current.chpt_dep[ap]=max(M.chpt_dep[ap], Current.chpt_dep[ap]).
5.      end_forall
6.  else /* M is a message from P_k */
7.      Write the content of M into a free entry of Current.sent_mesg[].
8.      Current.sent_mesg_no = Current.sent_mesg_no + 1.
9.  endif
```

(그림 6) 루틴 *UpdateChptRec*()를 위한 알고리즘

의 지역 체크포인트가 CGS에 포함되도록 강제하지는 않는다는 점이다. 즉, $[Serial_{P_k} - R\text{-}distance + 1, Serial_{P_k}]$의 범위 내에서 적절한 전역 체크포인트를 선택한다.

이런 선택 시에 고려해야 할 두 요소로, 생성해야할 강제 지역 체크포인트 개수와 포함될 지역 체크포인트의 시리얼 번호를 본다. 이 두 요소를 고려한 것이 12 행에서 15 행까지의 알고리즘이다. 알고리즘에서는 지역 체크포인트 시리얼을 하나 더 증가시킴으로써 얻는 이익과 강제적인 체크포인트 생성 요청을 줄임으로써 얻는 이익을 동등하게 보았다. 이 둘 간의 중요도 비중은 응용의 특성마다 다르게 둘 수는 있다.

마지막으로 (그림 6)은 *UpdateChptRec*()의 알고리즘을 보인다. 이 알고리즘을 통해 메모리에 있는 체크포인트 레코드가 갱신된다. 이후 디스크로 저장될 때 저장 주소는 *Current* 레코드의 prev_rec 필드에 저장됨으로써 필요시 저장 역순으로 읽어 들일 수 있다. *Current* 데이터는 디스크로 기록된 후 일부 필드가 초기화되거나 갱신됨으로써 다음 체크포인트 레코드 기록에 다시 사용된다.

4. 성능 특성 분석

체크포인팅 알고리즘의 성능에 영향을 미치는 요소는 매우 다양하지만 크게 고장 발생 시의 지역 체크포인트 평균 취소 개수와 전역 체크포인트 생성 비용의 2 가지가 중요하게 고려된다.

먼저 고장 발생 시 취소되는 지역 체크포인트의 개수는 일관성을 갖는 전역 체크포인트가 얼마나 가까운 시점을 반영하는가와 관계가 있다. 제안한 기법의 경우 분산 응용에 참여한 AP의 개수를 N이라할 때, 각 AP들의 지역 체크포인트의 최대 복귀 개수는 R-distance - 1개이다. 따라서 (R-distance -1)×N 이 이론적 최대 복귀 개수이고, 확률적으로는 이 값의 50% 이하로 복귀된다고 할 수 있다. 또한 중요 작업이 수행된 후에는 체크포인트 생성을 연속적으로 요청함으로써 복귀 가능 개수를 줄일 수 있다.

전역 체크포인트 생성에 따른 비용은 각 AP가 자신의 지역 체크포인트들을 생성할 때까지 기다려야 하는 시간 지연 및 추가 비용이다. 시간 지연은 몇 개의 AP들에게 지역 체크포인트 생성을 강제하는 메시지를 보내야 하는 가와 연관된다. 기존 일관성 전역 체크포인트 생성 알고리즘에서는 현재 상태를 반영하는 전역 체크포인트의 생성을 위해 대부분의 AP들로 강제 체크포인트 생성 요청을 해야 했다. 이에 반해 제안된 논문에서는 R-distance 개의 지역 체크포인트 중에서 시간 지연이 작은 체크포인트가 선택될 수 있다. 이로 인해 많은 경우 강제적 생성 요청이 없이도 일관성 전역 체크포인트를 선택할 수 있다. 이런 특성은 시간이 흐름에 따라 다른 AP들의 지역 체크포인트 시리얼이 증가하고, 이에 따라 자연스럽게 일관성 전역 체크포인트의 시점도 함께 갱신되기 때문이다. 즉 R-distance를 사용한 유연한 전역 체크포인트 선택이 가능함으로써 이를 생성하기 위한 시간 지연의 평균값을 크게 줄일 수 있다.

또한 제안한 방법은 알고리즘 수행에 필요한 제어 데이터는 유선 통신망을 사용하여 전송된다. 즉, CA들 간의 통신에서만 제어 데이터가 이동하며, 무선 통신망에서는 보이지 않도록 한다. 이와 함께, 체크포인트 레코드의 저장 및 관리가 서버에서 동작하는 CA에 의해 수행된다. 이런 특성은 무선 통신 비용이 상대적으로 크고 이동 기기의 경우 메모리나 디스크 저장 비용이 큰 것을 감안할 때 체크포인팅 작업에 드는 추가 비용을 줄일 수 있는 특징이 있다.

5. 결 론

본 논문에서는 이동 통신 환경에서 분산 응용을 수행하려 할 때 필요한 체크포인팅 기법에 대해서 연구하였다. 제안된 기법은 MSS에 체크포인팅 에이전트라는 소프트웨어를 둠으로써 무선 통신 비용을 크게 줄이고 일관성 전역 체크포인트 생성에 예상되는 시간 지연을 줄일 수 있다. 이를 위해 최대 복귀 허용 거리를 의미하는 rollback distance (R-distance)라는 개념을 고안하여 일정한 시간 범위 내에 전역 체크포인트가 유지될 수 있게 했다. R-distance를 통해 각 응용 프로세스들은 자신의 작업 진행 과정의 의미성에 따라 유연한 방식으로 체크포인트를 생성할 수 있으며, 작업 진행의 중요도에 따라 일관성 전역 체크포인트의 시점을 임의로 앞당길 수 있다. 이러한 특성을 지원하기 위해 체크포인팅 에이전트 간에 제어 데이터가 전송되어야 하는데, 이런 데이터 전송에는 유선 통신망을 사용함으로써 무선 통신망 사용에 따른 통신 추가비용을 없앴다.

참 고 문 헌

[1] T. Imielinski and B. R. Badrinath, Mobile Wireless Computing: Challenges in Data Management, *Communications of the ACM*, pp.19-28, Vol.37, No.10, October, 1994.

[2] Yi-Bing Lin, Failure Restoration of Mobility Databases for Personal Communication Networks, Wireless Networks, Vol.1, No.3, 1995.

[3] Sashidhar Gadiraju and Vijay Kumar, Recovery in the Mobile Wireless Environment Using Mobile Agents, *IEEE Trans. on Mobile Computing*, Vol.3, No.2, April, 2004.

[4] Ricardo Baratto, Shaya Potter, Gong Su, and Jason Nieh, MobiDesk: Mobile Virtual Desktop Computing, In *Proc. of the 10th International Conference on Mobile Computing and Networking*, pp.1-15, 2004.

[5] Dhiraj K. Pradhan, P. Krishna, and Nitin H. Vaidya, Recovery in Mobile Wireless Environment: Design and

Trade-off Analysis, In *Proc. of the 26th International Symposium on Fault-Tolerant Computing*, pp.16-25, 1996.

[6] Arup Acharya and B. R. Badrinath, Checkpointing Distributed Applications on Mobile Computers, In P*roc. of the 3rd International Conference on Parallel and Distributed Information Systems*, pp.73-80, 1994.

[7] Y. M. Wang, Consistent Gobal Chckpoints That Contain a Given Set of Local Checkpoints, IEEE Trans. on Computers, Vol.46, No.4, pp.456-468, 1997.

[8] Tongchit Tantikul and D. Manivannan, Communication-Induced Checkpointing and Asynchronous Recovery Protocol for Mobile Computing Systems, In *Proc. of the 6th International Conference on Parallel and Distributed Computing Applications and Technologies*, pp.70-74, 2005.

[9] Taesoon Park and Heon Y. Yeom, An Asynchronous Recovery Scheme based on Optimistic Message Logging for Mobile Computing Systems, In *Proc. of the 20th International Conference on Distributed Computing Systems*, pp.436-443, 2000.

[10] R. E. Strong and S. Yemini, Optimitic Recovery in Distributed Systems, *ACM Trans. on Computer Systems*, Vol.3, No.3, August, 1985.

[11] D. Manivannan and Mukesh Singhal, Quasi-Synchronous Checkpointing: Models, Characterization, and Classification, *IEEE Trans. on Parallel and Distributed Systems*, Vol.10, No.7, July, 1999.

[12] Cheng-Min Lin and Chyi-Ren Dow, Efficient Checkpoint-based Failure Recovery Techniques in Mobile Computing Systems, *Journal of Information Science and Engineering*, pp.549-573, Vol 17, No.4, 2001.

[13] Lorenzo Alvisi, E. N. Elnozahy, Sriram Rao, Syed Amir Husain and Asanka De Mel, An Analysis of Communication Induced Checkpointing, In *Proc. of the Symposium on Fault-Tolerant Computing Symp.*, pp.242-249, 1999.

[14] Franco Zambonelli, On the Effectiveness of Distributed Checkpoint Algorithms for Domino-Free Recovery, In *Proc. of High Performance Distributed Computing*, pp.124-131, 1998.

[15] Mootaz Elnozahy, et. al., A Survey of Rollback-Recovery Protocols in Message-Passing Systems, *Technical Report: CMU-CS-99-148*, June, 1999.

[16] Yi-Min Wang and W. Kent Fuchs, Lazy Checkpointing Coordination for Bounding Rollback Propagation, In *Proc. of the International Symposium on Reliable Distributed Systems*, pp.78-85, 1993.

[17] Lapmport, Time, clocks, and the Ordering of Events in a Distributed System, Communication of ACM, Von. 21, No.7, pp.558-565, 1978.

임 성 채

e-mail : sclim@dongduk.ac.kr

1992년 서울대학교 컴퓨터공학과(학사)

1994년 한국과학기술원 대학원 전산학과 (이학석사)

2004년 한국과학기술원 대학원 전산학과 (공학박사)

2000년~2000년 서울정보시스템 기술부 부장

2000년~2005년 2월 코리아와이즈넛 연구소 수석연구원 및 이사

2005년 3월~현 재 동덕여자대학교 컴퓨터전공 조교수

관심분야 : Web IR, 멀티미디어 시스템, 고성능 색인 기법