모바일 컴퓨팅 시스템에서 과부하를 줄이기 위한 경량 체크포인팅 기법

이창엽[O] 최창열 김성수
아주대학교 정보통신전문대학원
{cylee[O], clchoi, sskim}@ajou.ac.kr

Light-weight Checkpointing Mechanism for Reducing Overhead in Mobile Computing Systems

Changyup Lee[O], Changyeol Choi, Sungsoo Kim
Graduate School of Information and Communication, Ajou University

요 약

최근 모바일 환경에서 모바일 기기가 결함에 쉽게 노출될 수 있다는 특성 때문에 모바일 컴퓨팅 시스템에서의 결함 허용에 대한 관심이 높아지고 있다. 결함 허용을 제공하기 위한 기법 중 하나로 체크포인팅을 들 수 있는데, 이를 모바일 환경에 적용하기 위해서는 체크포인트의 실행으로 인해 모바일 기기에 가해지는 과부하를 줄이는 것이 중요하다. 따라서 본 논문에서는 각각의 프로세스가 독립적으로 실행하는 BASIC 체크포인트를 없앰으로써 과부하를 줄이기 위한 체크포인팅 기법을 제안한다.

1. 서론

최근 모바일 환경에서 모바일 기기가 결함에 쉽게 노출될 수 있다는 특성 때문에 모바일 컴퓨팅 시스템에서의 결함 허용에 대한 관심이 높아지고 있다. 이런 특성의 원인으로는 한정된 모바일 기기의 자원, 낮은 대역폭, 잦은 네트워크와의 연결과 이탈, 그리고 배터리 소비 등이 있다[1]. 결함 허용을 제공하기 위한 기법 중 하나로 체크포인팅을 들 수 있다. 체크포인팅 기법은 프로세스의 상태를 주기적으로 안정된 저장소에 저장함으로써, 결함이 발생하였을 때 체크포인트를 실행한 지점부터 시스템을 재시작할 수 있도록 한다. 체크포인트는 각각의 프로세스가 독립적으로 실행하는 BASIC 체크포인트와, 도미노 효과를 막기 위해 실행하는 FORCED 체크포인트로 나눌 수 있다. 기존의 체크포인팅 기법들은 BASIC 체크포인트를 실행시키면서 FORCED 체크포인트의 수를 최대한 줄이기 위해 노력하였다. 하지만 BASIC 체크포인트를 실행하기 위해서는 시스템 자신의 상태에 대한 감시가 필요하고, 이 과정에서 과부하가 발생하게 된다. 특히 모바일 기기에서 발생하는 과부하는 배터리 소비 문제로 이어지기 때문에 모바일 기기에 가해지는 과부하를 줄이는 것이 중요하다. 따라서 본 논문에서는 각각의 프로세스가 독립적으로 실행하는 BASIC 체크포인트를 없앰으로써 과부하를 줄이는 체크포인팅 기법을 제안한다.

본 연구는 21세기 프론티어 연구개발사업의 일환으로 추진되고 있는 정보통신부의 유비쿼터스컴퓨팅 및 네트워크 원천기반기술개발 사업의 지원에 의한 것임.

2. 관련연구

J. Helary [2] 등이 제안한 HMNR은 분산 환경에 적용되는 통신에 의한 체크포인팅 (CIC) 프로토콜로, 도미노 효과를 막으면서 FORCED 체크포인트의 수를 최대한 줄인다. 하지만 체크포인트의 수를 줄이기 위해 필요한 계산과, 메시지에 실어야 할 제어 정보의 양이 많아지게 되었다. 본 논문에서는 BASIC 체크포인트를 실행하지 않음으로써 HMNR보다 전체 체크포인트의 수, 계산의 양과 제어 정보의 크기를 줄이는 경량의 알고리즘을 제안한다.

A. Agbaria [3] 등은 과부하 비율이라는 단위를 소개하여 체크포인팅 프로토콜의 과부하를 측정하였다. 과부하 비율은 체크포인트를 실행시킬 때 생기는 과부하와 제어 정보의 크기뿐만 아니라 오류가 발생하였을 때의 상태까지 고려하였다. 이 과부하 비율을 사용하여 본 논문에서 제안한 기법과 HMNR의 과부하를 비교한다.

3. 시스템 모델

모바일 컴퓨팅 시스템은 모바일 호스트 (MH)와 모바일 지원 스테이션 (MSS)으로 이루어진다. 하나의 MSS에 의해 서비스가 제공되는 셀 내에 존재하는 MH는 MSS와 직접적으로 통신할 수 있다. 모든 MH는 한 시점에서 하나의 셀에만 속할 수 있으며, MSS들은 유선 네트워크에 의해 안정적으로 통신할 수 있다. 그림 1은 전형적인 모바일 컴퓨팅 시스템의 예를 보여준다.

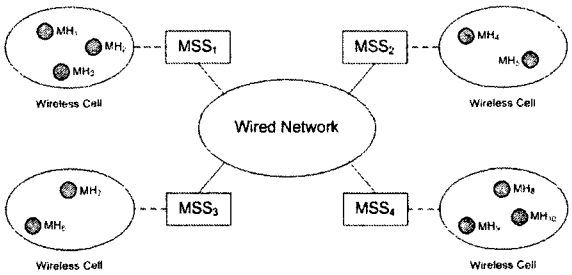

그림 1. 모바일 컴퓨팅 시스템의 예

본 논문에서 고려한 환경은 n 개의 프로세스들로 이루어진 모바일 컴퓨팅 시스템으로, 프로세스들은 하나의 MSS와 그것에 의해 서비스를 받고 있는 셀 안의 $n-1$ 개의 MH 내에서 동작하고 있다. 또한 메시지를 교환하기 위해 무선 통신을 사용하며, 모든 메시지는 체크포인트에 필요한 제어 정보를 포함하고 있다.

로컬 체크포인트는 프로세스의 상태를 저장한 것이고, 글로벌 체크포인트는 로컬 체크포인트들의 집합이다. 한 쌍의 체크포인트에서, 한 프로세스가 메시지를 받았는데 다른 프로세스에서 보낸 기록이 없다면 그 메시지는 ORPHAN 메시지라고 하며, 해당 메시지가 없는 체크포인트 쌍을 일관되었다고 한다. 또한 글로벌 체크포인트를 이루고 있는 모든 로컬 체크포인트가 일관되었다면 그 글로벌 체크포인트를 일관되었다고 한다.

일관된 글로벌 체크포인트를 구축하기 위해서는 지그재그 경로를 없애는 것이 중요하다. [4]에서는 지그재그 경로에 대해 설명하였으며, 서로 다른 프로세스에 속해 있는 체크포인트의 집합이 지그재그 경로를 가지고 있지 않으면 일관된 글로벌 체크포인트를 이룬다는 것을 증명하였다. 본 논문에서 제안한 체크포인팅 기법은 이 증명에 따라 지그재그 경로를 없앰으로써 일관된 글로벌 체크포인트를 구축한다.

4. FORCED 체크포인트만을 이용한 체크포인팅 기법 (CSF)

본 논문에서 제안한 체크포인팅 기법은 BASIC 체크포인트를 제외시키고, FORCED 체크포인트만을 실행시킨다. 이 기법으로 인해 시스템을 감시할 때 생기는 과부하를 MSS에만 부과함으로써 모바일 기기에 가해지는 부담을 줄일 수 있다. CSF의 기본적인 개념은, MSS가 체크포인트를 실행한 다음에 보낸 메시지를 받은 모든 프로세스들은 체크포인트를 실행해야 한다는 것이다. 메시지를 받은 프로세스는 그 메시지를 접수하기 전에 체크포인트를 실행해야 하는데, 이를 통해 지그재그 경로가 생기는 것을 막을 수 있다. 쓸모 없게 되는 체크포인트의 수를 줄이기 위해서, 각 프로세스는 일관된 글로벌 체크포인트 구축 시에 단지 한 번의 체크포인트만을 실행한다. 일단 일관된 글로벌 체크포인트가 구축되면 모든 프로세스는 MSS가 체크포인트를 실행시킬 때까지 체크포인트를 실행하지 않는다. 이 프로토콜은 MH와 MSS에 각각 사용되

MH에 의해 사용되는 알고리즘에서 각 프로세스는 현재 자신이 실행한 체크포인트의 수에 대한 정보를 가지고 있고, 제어 정보에는 모든 프로세스의 체크포인트 실행 횟수가 담겨 있다. 하나의 프로세스가 다른 프로세스로부터 받은 메시지에 실려온 제어 정보를 통해 자신보다 더 많은 체크포인트를 실행한 프로세스가 있다는 것을 발견하면, 그 프로세스는 일관된 글로벌 체크포인트를 구축하기 위해 강제적으로 체크포인트를 실행해야 한다. 만약 모든 프로세스가 실행한 체크포인트의 수가 같다면 이미 일관된 글로벌 체크포인트 상태이기 때문에 체크포인트를 실행하지 않고 메시지를 받는다.

일관된 글로벌 체크포인트의 구축은 MSS에서부터 시작되기 때문에 MSS에 의해 사용되는 알고리즘에는 두 가지 정보가 더 필요하다. 우선 그룹 내에 있는 모든 프로세스에서 실행된 체크포인트들이 일관되어 있는지를 확인하고, 일관되지 않았다면 체크포인트를 실행하지 않는다. 만약 메시지를 받았을 때, 모든 프로세스에서 실행한 체크포인트의 수가 같다면 일관된 글로벌 체크포인트가 구축되었다는 것을 의미하며, 그 후부터 받은 메시지의 수를 계산한다. MSS 내의 프로세스가 받은 메시지의 수가 미리 정해둔 메시지의 수와 같아지면, 그 프로세스는 체크포인트를 실행함으로써 일관된 글로벌 체크포인트 구축 과정을 시작한다.

그림 2는 CSF를 사용하여 일관된 글로벌 체크포인트를 구축하는 예이고, 표 1은 그림 2에서 시간에 따른 프로세스 정보의 상태 변화를 나타낸다. *ck_MSS*와 *ck_MH1*, *ck_MH2*는 각각의 프로세스가 가지고 있는 자신이 실행한 체크포인트의 수이고, *ckpt*[*n*]은 각 메시지에 실려

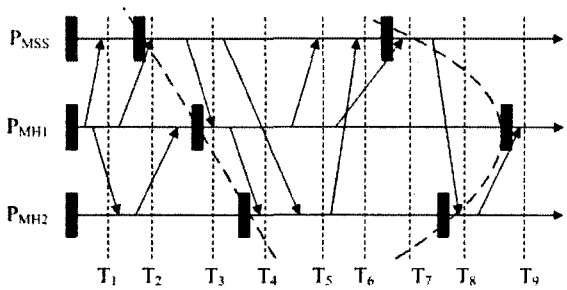

그림 2. CSF를 사용한 일관된 글로벌 체크포인트 구축

표 1. 시간에 따른 프로세스 정보의 상태 변화

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9
ck_MSS	0	1	1	1	1	1	2	2	2
ck_MH1	0	0	1	1	1	1	1	1	2
ck_MH2	0	0	0	1	1	1	1	2	2
ckpt[0]	0	0	1	1	1	1	1	2	2
ckpt[1]	0	0	0	1	1	1	1	1	1
ckpt[2]	0	0	0	0	0	1	0	1	2
consistent	T	F	F	F	F	T	F	F	F
count	1	0	0	0	0	1	0	0	0

는 두 가지 알고리즘이 있다.
가는 제어 정보로, 프로세스들이 실행한 체크포인트의 수를 나타낸다. *consistent*와 *count*는 MSS가 가지고 있는 정보로, *consistent*는 일관된 글로벌 체크포인트가 구축되었는지에 대한 정보이고, *count*는 *consistent*가 True 상태일 때 이후에 MSS가 받은 메시지의 수이다. 위의 예에서 MSS는 *count*가 1일 때 체크포인트를 실행한다. 일단 MSS가 체크포인트를 실행한 후에, MSS로부터 메시지를 받은 프로세스도 체크포인트를 실행하게 된다. T_3에서 P_{MH1}은 P_{MSS}로부터 메시지를 받고, *ck_MH1*과 *ckpt*[*n*]을 비교한다. *ckpt*[0]이 *ch_MH1*보다 크기 때문에 다른 프로세스에서 체크포인트가 실행된 것을 의미하므로 MH1도 체크포인트를 실행한다. T_4에서 P_{MH2}는 P_{MH1}로부터 받은 메시지를 통해서 체크포인트를 실행하였고, 그 결과 일관된 글로벌 체크포인트를 구축하게 되었다. T_5에서 P_{MH1}이 보낸 메시지에는 아직 P_{MH2}가 실행한 체크포인트에 대한 정보가 없기 때문에 P_{MSS}의 *consistent*는 False이다. 하지만 P_{MSS}는 T_6에서 P_{MH2}가 보낸 메시지를 통해 일관된 글로벌 체크포인트가 구축되었다는 것을 알게 되었다. 따라서 *consistent*를 True로 수정하고 *count*를 1 증가시킴으로써 다음에 메시지를 받았을 때 체크포인트를 실행하게 된다.

5. 과부하 비율 분석

본 논문에서는 [3]에서 정의한 과부하 비율 공식을 사용하여 CSF의 과부하를 계산하였으며, 그 공식은 다음과 같다.

$$v = \frac{\Gamma^*(F+1)}{T - O^*(F+1)} - 1,\ v \geq 0.$$

T는 하나의 프로세스에서 두 개의 연속된 체크포인트가 실행될 때의 시간 간격이고, F는 T 시간 동안 결함이 발생되지 않았을 경우에 실행되는 FORCED 체크포인트의 수이다. O는 전체 체크포인트 과부하로, 체크포인트로 인해 생기는 과부하와 제어 정보로 인해 생기는 과부하의 합이다. Γ는 하나 이상의 결함이 발생하였을 때 예상되는, 체크포인트 간격 사이에서의 프로세스 실행 시간이다. 위의 공식을 사용하여 CSF와 HMNR의 과부하 비율을 분석한 결과가 그림 3에 표현되었다. CSF에서 MSS는 T 시간마다 체크포인트를 실행하며, 그에 따라 MH도 T 시간 주기로 체크포인트를 실행하게 된다. HMNR에서는 모든 프로세스들이 T 시간마다 BASIC 체크포인트를 실행하기 때문에 HMNR에서 실행한 BASIC 체크포인트의 수는 CSF에서 실행한 모든 체크포인트의 수와 같다. 하지만 HMNR에서는 BASIC 체크포인트 외에 FORCED 체크포인트를 실행함으로써, 총 체크포인트 수가 CSF에서의 체크포인트 수보다 더 많아지게 된다. T는 500으로 정했으며, 각 단위 시간 동안의 과부하 비율은 크게 차이가 나지 않지만 과부하가 누적되면서 그 차이가 점점 커지는 것을 볼 수 있다.

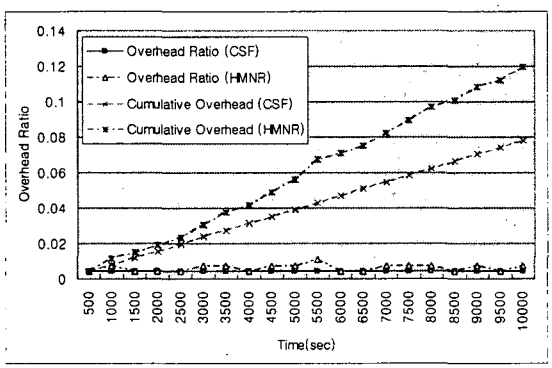

그림 3. CSF와 HMNR의 과부하 비율 분석

6. 결론

모바일 컴퓨팅 시스템은 기존의 분산 컴퓨팅 시스템과 달리 모바일 기기가 쉽게 결함에 노출된다는 특징을 가지고 있다. 이를 해결하기 위한 결함 허용 기법 중 하나로 체크포인팅 기법이 있지만, 체크포인트의 실행으로 인해 모바일 기기에 과부하가 가해지게 된다. 따라서 본 논문에서는 모바일 기기에 가해지는 과부하를 줄임으로써 모바일 컴퓨팅 시스템에 사용될 수 있는 경량 체크포인팅 기법을 제안하였다. CSF는 FORCED 체크포인트만을 실행하기 때문에 BASIC 체크포인트를 실행할 때 생기는 과부하를 없앨 수 있다. 또한 CSF를 사용함으로써 모든 프로세스가 실행하는 전체 체크포인트의 수를 줄일 수 있고, 이로 인해 전체 시스템의 과부하 비율을 낮게 유지할 수 있다. 하지만 시스템 내에서 발생하는 메시지의 수와 프로세스에서 실행하는 체크포인트의 수에는 밀접한 관련이 있다. 따라서 메시지의 수와 체크포인트로 인한 과부하의 상관관계에 대해 분석하여 CSF의 효율성을 최대화할 수 있는 체크포인트의 주기를 구하는 것이 중요하다.

참고문헌

[1] F. Quaglia, B. Ciciani and R. Baldoni, "Checkpointing Protocols in Distributed Systems with Mobile Hosts: a Performance Analysis," Proceedings of 3rd Workshop on Fault-Tolerant Parallel and Distributed Systems, Vol. 1388, pp. 742-755, Apr. 1998.

[2] A. Mostefaoui, J. Helary, R. Netzer and M. Raynal, "Communication-Based Prevention of Useless Checkpoints in Distributed Computations," Distributed Computing, Vol. 13, No. 1, pp. 29-43, Jan. 2000.

[3] A. Agbaria, A. Freund and R. Friedman, "Evaluating Distributed Checkpointing Protocols," Proceedings of 23rd IEEE International Conference on Distributed Computing Systems, pp. 266-273, May 2003.

[4] R. Netzer and J. Xu, "Necessary and Sufficient Conditions for Consistent Global Snapshots," IEEE Transactions on Parallel and Distributed Systems, Vol. 6, No. 2, pp. 165-169, Feb. 1995.