모바일 호스트의 안정적인 복구를 위한 메시지 로그 기법들의 성능비교

최가현, 황병연
가톨릭대학교 컴퓨터공학과
e-mail : {namu25,byhwang}@catholic.ac.kr

Performance Comparison of Message Log Methods for Stable Recovery of Mobile Host

Ga-Hyun Choi, Byung-Yeon Hwang
Dept. of Computer Engineering, The Catholic University of Korea

요 약

모바일 컴퓨팅 환경은 네트워크가 손상되기 쉬우며, MH(Mobile Host)를 지원하는 MSS(Mobile Support Station)가 안정적이지 못하기 때문에, 이런 상황을 고려하여 시스템 환경을 모델링 한다. 메시지 전송 시 에러가 발생했을 경우 복구 기법으로 체크포인팅 기법과 메시지 로깅 기법을 추가적으로 사용하게 된다. 본 논문에서는 움직임을 기반으로 한 모바일 시스템 환경에서 체크포인트와 함께 메시지 로그를 사용하는 기법의 성능을 비교하고자 한다.

1. 서론

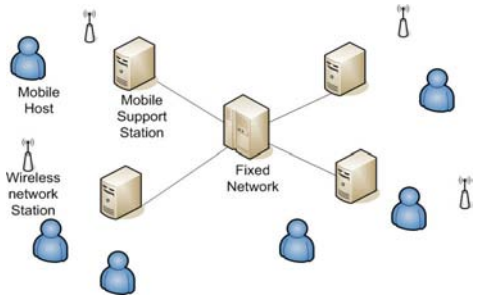
현재 분산 처리 시스템은 무선 상에서 필요한 서비스들의 요구가 증대됨에 따라, 그 영역을 다양한 분야로 확대시켜 나가고 있다. 무선 서비스의 불안정한 요소 중 하나는 오류가 발생할 가능성이 많다는 것이다. 따라서 이 점을 보안하기 위해, 체크포인트(checkpoint)를 사용해서 복구하는 기법들이 널리 사용되고 있다. 체크포인트를 사용한 방법은 안정적이지 못한 MH 와 손상되기 쉬운 네트워크 환경에 적용될 수 있기 때문에, 현실 세계에 적합하다. MH 는 MSS 에 연결되어 있으면서, 자신들의 변경 사항을 MSS 에 보고하게 된다. 이런 MH 는 셀(cell)이라고 하는 영역으로 묶여 있어서 이동 시 그 정보를 MSS 에 보고하게 된다. 이런 과정을 통해, 데이터의 무결성과 위치 정보의 투명성을 유지해 나간다. 모바일 환경을 구성하기 위해서는 다음의 네 가지 차원의 특성을 고려하여야 한다[1]. (그림 1)은 일반적인 이동 통신 환경에서의 네트워크 구조이다.

1) 낮은 대역폭(low bandwidth): 무선 상에서의 대역폭은 중요한 자원이 되기 때문에, 실질적인 정보가 아닌 것에 대해 많은 대역폭을 할당해 줄 수는 없다.

2) 공간 제약성(space limitation): MH 디스크 용량이 제한되어 있기 때문에, 체크포인트에 관한 정보를 MSS 에 이전시켜줘야 한다.

3) 움직임 핸들링(mobility handling): MH 움직임에 관한 정보는 그 움직인 셀에 해당하는 MSS 에 지속적으로 기록된다. 필요한 상황이 되면 다시 역추적해서 움직임에 대한 정보를 얻어 복구한다.

4)끊김 연산(disconnected operation): 네크워크로부터 자주 발생하는 끊김에 대해서, 적극적으로 대처하기 위한 방안들이 고려되어야 한다.



(그림 1) 이동 통신 환경

무선 상에서 발생하는 끊김 연산은 일반적으로 자주 발생하는 문제이기 때문에, 대처 방안뿐만 아니라 이미 끊긴 상태를 어떻게 복원하느냐 하는 문제도 중요하다. 따라서 빠른 회복을 위해서 앞에서 말한 체크포인트뿐만 아니라 메시지 로그를 두어 메시지들의 이동 경로를 기록하여 끊김 상태가 발생하거나 예기치 않은 오류가 발생했을 경우에 효과적인 회복이 가능해야 한다.

본 논문에서는 움직임을 기반으로 한 컴퓨팅 환경에서 체크포인트와 메시지 로그를 사용하는 랜덤 시나리오를 제시하고, 성능을 비교해 본다. 본 논문의 구성은 다음과 같다. 2 장에서는 오랫동안 제시되어 온 체크포인팅 기법과 메시지 로깅 기법을 알아보고, 3 장에서는 기본적인 시스템 환경에 대해 살펴본다. 4 장에서는 이에 대한 랜덤 시나리오를 구성한 뒤 나온 결과에 대해 분석한다. 끝으로 5 장에서 결론을 맺는다.

2. 관련 연구

이동 컴퓨팅 환경은 오랫동안 연구되어 온 분산 처리 환경에서 유래된 것이기 때문에 그 동안 연구되어 온 체크포인팅 기법과 메시지 로깅 기법을 살펴봐야 한다. <표 1>은 체크포인팅 기법과 메시지 로깅 기법을 오류가 나지 않을 경우와 오류가 발생했을 경우 복구 비용에 관해 정리한 표이다.

<표 1> 체크포인팅 기법과 메시지 로깅 기법

		Checkpointing-Only				Message Logging		
		CCP	SCP	CPCP	CICP	PML	CML	OML
Failure-free Operation cost	Message Overhead	High	Mid	High	Low	No	High	Mid
	Storage Access Overhead	Low	Low	High	Low	High	No	Mid
Failure recovery cost	Amount of Recomputation	High	High	High	High	Low	Low	Mid
	Asynchronous Recovery	No	No	No	No	Yes	No	No

2.1 체크포인팅 기법

일정한 시간간격으로 실행 가능한 트랜잭션을 저장하기 위해 로그에 쓰는 것을 체크포인팅 기법이라고 한다. 체크포인트 간격은 체크포인트들 사이의 간격으로 정의된다. 체크포인트를 사용하는 목적은 오류로부터 복구하기 위한 시간을 줄이도록 하기 위함이다.

체크포인팅 기법만 사용하는 경우는 세 가지로 나눠서 생각해 볼 수 있다. CCP (Coordinated Checkpointing Scheme) 기법은 체크 포인트를 활성화하기 위해 관련된 프로세스 전체를 다 필요로 한다[2]. 이 방법은 오버헤드가 많기 때문에 끊김이 자주 발생하여 대역폭이 작은 이동 컴퓨팅 환경에서는 적당하지 않다. CPCP(Communication-pattern Based Checkpointing Scheme) 기법은 각 프로세스 간의 오버헤드를 제거하고 독립적으로 체크포인트하게 된다[3]. 이런 독립적인 체크포인팅이 가능하기 위해서는 일관성 있는 복구가 이뤄져야 하는데, 체크포인팅이 자주 발생하게 되면, 각 프로세스에 대한 통신 패턴 의존도가 높아져 독립적인 속도를 지연시킨다. 따라서, 성능이 현저하게 떨어지게 된다. CICP(Communication-induced Checkpointing Scheme) 기법은 일관성을 보장하기 위해 각 메시지를 인덱스로 만들기 때문에 가장 적은 오버헤드를 갖게 된다[4].

2.2 메시지 로깅 기법

데이터베이스에서 일어나는 대부분의 변화들은 로그에 저장되는데, 메시지 값에 의존적으로 로그에 저장하는 방법을 메시지 로깅이라고 한다. 보통 데이터베이스에서 변화되어 일어나는 행동에 대한 값들을 저장한다.

CML(Causal Message Logging Scheme) 기법은 저장장치로의 잦은 접근을 피하기 위한 방법이다. 따라서 별도의 디스크 공간을 필요로 하지 않을 뿐만 아니라, 메시지를 핸들링하기 위해 정보를 옮기는 것들도 비교적 의존적이다[5, 6]. OML(Optimistic Message Logging Scheme) 기법은 비동기적인 복구를 구현하기 위한 방법으로서 안정적인 로그 빈도를 조절한다. 특이할 만한 점이라면, 벡터 클럭(vector clock)이라는 것을 두어 비정상적인 메시지가 발생시키는 문제들을 해결한다는 것이다[7]. PML(Pessimistic Message Logging Scheme) 기법은 비동기적인 복구를 기반으로 한다. 따라서 프로세스는 시스템의 오류에 독립적으로 반응할 수 있을 뿐만 아니라, 다른 프로세스의 메시지를 기다리지 않고 즉시 복구를 실행할 수 있다[8].

3. 시스템 모델의 구성

3.1 컴퓨팅 환경 구성

모바일 컴퓨팅 환경의 시스템을 구현하기 위해서는 여러 개의 MH 와 고정된 하나의 MSS 가 필요하다. 이 MSS 는 MH 를 지원하기 위해 다양한 서비스를 제공하며, 셀이라고 부르는 일정 지역을 포함하게 된다[9]. MH 와 MSS 는 고성능 무선 통신으로 링크되어 있다. 이 무선 링크는 셀 안에 있는 MH 와 MSS 간에 FIFO(First in First out) 통신을 하게 되며, 작업은 프로세스 단위로 이루어진다. 이 프로세스의 각 상태(state)는 하나의 시퀀스(sequence)로 되어 있고, 하나의 상태에서 다른 상태로 전이될 때를 이벤트(event)라고 한다. 이 이벤트는 다른 프로세스와 상호 교류가 없을 수도 있고(이 상태를 internal state 라고 한다) 서로 메시지를 주고 받을 수도 있다(external state 라고 한다). 이렇게 한 프로세스 안에서 일어나는 이벤트 시퀀스를 컴퓨테이션(computation)이라고 한다.

3.2 오류 처리

프로세스 처리 중 오류가 발생하면 바로 멈추게 되

고(fail-stop), 복구 작업을 시작하게 된다. 오류가 발생하면 MH 나 MSS 에 있는 메모리를 잃어버리기 때문에 안정적인 저장소를 확보해야 한다. 어떤 MH 가 한 셀에서 다른 셀로 이동할 때를 핸드오프(hand-off)라고 하는데, 핸드오프는 각 셀의 경계에서 정상적으로 작동해야 한다.

3.3 저장 장치

MH 의 오류율이 높다면, MSS 에 저장되어 있는 체크포인트와 메시지 로그를 모으기 위한 복구 비용은 높게 측정된다. 뿐만 아니라, 각각의 MH 의 체크포인트와 로그를 옮기는 비용도 안정적인 저장장치를 설계하는데 필요한 기본적인 요소이다. MH 가 작은 범위 안에 있는 셀을 돌 때라면, 체크포인트와 메시지 로그는 MH 근처에 있는 MSS 로만 옮기면 되기 때문에, 약간의 오버헤드만으로도 저장할 수 있다. 오류가 일어나지 않았을 때와 오류가 일어날 때 드는 비용 또한 설계 시 고려해야 할 중요한 요소이다.

3.4 복구 기법

오류가 발생하면 MH 는 체크포인트와 메시지 로그를 고려하여 독립적으로 복구를 수행한다. 여기서 일어나는 복구는 모두 롤백 복구(rollback recovery)이다. MH 가 최근의 체크포인트까지 오류난 것들을 롤백해서 로그된 메시지들을 다시 되돌리는 것을 의미한다

4. 성능 평가

4.1 랜덤 시나리오

실험에 들어 가기 앞서 임의로 MH 와 MSS 간의 메시지 전송, 핸드오프, 오류처리 상황을 고려하여 시나리오를 구성한다.

랜덤 시나리오에서 비교하고자 하는 메시지 로그 기법은 PL(Pessimistic Logging), TL(Trickle Logging), LL(Lazy Logging) 이 세 가지 기법이다. 이 기법들은 OML 기법 중에서 대표적인 기법으로 본 연구 의도와 적합한 특성을 갖기 때문에 랜덤 시나리오에 적용한다.

체크포인트는 주기적으로 (1 회/ 3sec) 발생한다고 가정한다. 메시지는 체크포인트와 동시에 발생하거나 그렇지 않을 수 있다. 핸드오프는 체크포인트와 동시에 발생하거나, 메시지 전송과 동시에 발생하거나, 핸드오프와 체크포인트 둘 다 발생하거나, 핸드오프와 체크포인트 둘 다 발생하지 않을 수 있다. 오류는 체크포인트, 핸드오프, 메시지 전송과 동시에 발생하거나 독립적으로 발생할 수 있다. 본 시나리오에서는 발생 비용을 계산하기 위해 상대 비용을 사용한다. 체크포인트 발생 비용은 기본적인 카운터만을 사용하기 때문에 1 값을 갖도록 하며, 핸드오프 발생 비용 또한 랜덤한 값을 부여하는 카운터만을 요구하기 때문에 1 값을 부여한다. 메시지의 중복 여부를 확인하여 저장해야 하는 메시지 발생 비용은 2 로 한다. 오류 발생 비용은 발생 여부만을 확인하면 되기 때문에 1 로 한다. 기본적인 시스템을 작동하기 위한 상대 비용은 10 이라 가정한다. 이는 MH 와 MSS 간의 정보 교환과 연결 상태 유지를 위한 절대적인 값을 고려한 비용이다.

timer	ckpt#	msg#	HO#	F#	오류가 없을경우 발생정보	네트워크 비용	오류가 생겼을때 발생정	네트워크 비용
1	1	1				10		10
2						10		10
3						10		10
4	2	2	1		HO1,c2,m1,m2	16	HO1,c2,m1,m2,	16
5						16		16
6						16		16
7	3			1		16		16
8		3				16		16
9				2		16		16
10	4		2		HO2,c4,m3	20	HO2,F1,F2,c4,m3	22
11		4	3	3	HO3,c4,m4	24	HO3,c4,m4	26
12						24		26
13	5					24		26
14		5				24		26
15						24		26
16	6		4	4	HO4,c6,m5	28	HO4,F4,c6,m5	31
17						28		31
18						28		31
19	7		5		HO5,c7	30	HO5,c7	33
20		6	6		HO6,c7,m6	34	HO6,c7,m6	37
21			7		HO7,c7	36	HO7,c7	39
22	8					36		39
23		7	8		HO8,c8,m7	40	HO8,F4,c8,m7	44
24						40		44
25	9					40		44
26						40		44
27						40		44
28	10					40		44
29						40		44
30						40		44
31	11					40		44

(그림 2) PL 에 대한 랜덤 시나리오

PL 기법은 핸드오프에 민감하게 반응한다. 오류가 발생하지 않는 경우라면 핸드오프가 증가함에 따라 네크워크 비용이 증가하게 된다. 여기서 말하는 네트워크 비용은 메시지 로그 시에 발생하는 mi(i >= 0)와 체크포인트 발생 비용 ci(i >= 0), 핸드오프 발생 비용 HOi(i >= 0)를 의미하며, 오류가 있을 경우의 네트워크 비용은 오류 발생 비용 Fi(i >= 0)까지를 포함한다. i 는 0 부터 증가되는 값이며, 체크포인트의 경우 주기적으로 증가하지만, 핸드오프나 오류는 랜덤하게 증가한다. 메시지가 전송될 때마다 네트워크 비용이 누적되기 때문에 메시지 전송량이 많을수록 비용이 높아진다. (그림 2)의 시나리오를 살펴보면, 1 초일 때 첫 번째 체크포인트 c1 이 발생하고, 첫 번째 메시지 m1 이 발생하지만, 핸드오프가 일어나지 않았기 때문에 저장되지는 않는다. 발생비용은 기본적인 시스템 운용 비용 10 이다. 핸드오프가 처음 발생하는 시점인 4 초에선 두 번째 체크포인트 c2 가 발생하고 메시지 m2 가 전송되며, 첫 번째 핸드오프 HO1 이 일어난다. 이 때, 메시지와 체크포인트의 내용이 저장된다. 네트워크 비용은 HO1 과 c2, m1, m2 의 절대 비용을 더한 값이다.

LL 기법은 오류가 발생하지 않을 경우 기본적인 시스템 작동 비용인 10 만을 갖게 된다. 반면, 오류가 생길 경우는 오류가 일어나는 시점의 핸드오프 발생 비용과 체크 포인트 발생 비용과 오류 발생 비용을 합하여 네트워크 비용을 산출한다. 오류가 나지 않는다면 어느 시점에서 핸드오프가 발생했는지에 관한 정보조차 확인할 수 없다.

TL 기법은 오류가 발생하지 않을 경우, 핸드오프가 발생한 다음 발생 정보를 저장하여 누적한다. 오류가 발생한다면 첫 번째 오류 발생 이후부터 핸드오프에 따라 비용이 발생하게 된다. 따라서 첫 번째 오류가 발생한 이후인 8 초에 핸드오프 HO1 과 최근의 체크포인트 c3 와 그 전까지 보내졌던 메시지 m1, m2, m3

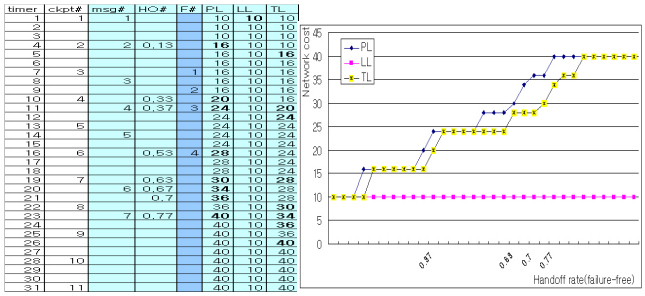
가 동시에 전송된다. 그 이후엔 PL 기법과 동일하게 핸드오프에 민감하게 네트워크 비용이 산출된다.

4.2 성능평가

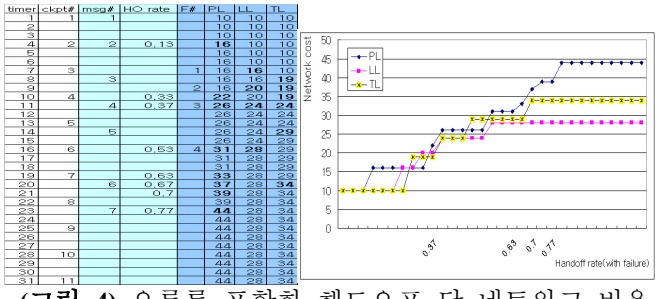
랜덤 시나리오를 바탕으로, 세 가지 메시지 로그 기법에 따른 네트워크 비용을 예측해 보았다. 네트워크 비용은 오류가 발생했을 때와 그렇지 않았을 때를 나눠서 측정할 수 있다. 핸드오프 비율은 단위 시간 당 발생 횟수로 계산하였다.

(그림 3)의 왼쪽 그림은 오류가 발생하지 않을 경우 PL 기법, LL 기법, TL 기법의 핸드오프 비율당 네트워크 비용을 표로 만든 것이다. 핸드오프 비율이 0.13 일 때 PL 기법, LL 기법, TL 기법은 각각 16, 10, 10 의 비용을 갖는다. PL 기법은 핸드오프에 민감하기 때문에 핸드오프 비율에 가장 큰 영향을 받는다. 오른쪽 그래프는 왼쪽 그림에서 계산된 네트워크 비용을 핸드오프 비율에 따라 누적해서 보기 쉽게 나타낸 그림이다.

(그림 4)의 왼쪽 그림은 오류가 발생할 경우의 네트워크 비용을 누적해서 나타낸 것이다. 7 초 일 때, PL 기법은 오류 발생 여부와는 관계없이 핸드오프 발생에 영향을 받기 때문에 일정한 값을 유지하지만, LL 기법은 네트워크 비용이 크게 증가한다. TL 기법은 오류가 발생하는 다음 시점에서 네트워크 비용이 증가한다. (그림 4)의 오른쪽 그래프를 통해서 오류가 발생했을 경우 각각에 대한 네트워크 비용 변화를 확인할 수 있다.



(그림 3) 오류가 없는 핸드오프 당 네트워크 비용



(그림 4) 오류를 포함한 핸드오프 당 네트워크 비용

5. 결론

이동 컴퓨팅 환경에서 MH 는 저장 장치를 작게 하기 위해 MSS 에 끊임없이 자신의 상태 메시지를 보내게 된다. 많은 양의 메시지들이 보내어질 때 불완전한 무선 네트워크 환경에서 끊김이 발생한다. 이것을 방지하기 위해 체크포인팅 기법이나 메시지 로깅 기법 중 하나를 사용해서 복구한다. 본 논문에서는 이동 호스트의 움직임 많은 상황을 고려해서 컴퓨팅 환경을 구성하고 그 환경에서 체크포인팅 기법뿐만 아니라 메시지 로깅 기법을 적용한 시나리오를 구성해서 성능을 비교해 보았다. 그 결과 PL 기법은 메시지의 중요도가 높거나 실시간으로 오류 상황을 보고해야 하는 시스템에 적합하고, LL 기법은 비교적 안정적인 시스템에서 오류 빈도가 낮으며, 복구 작업의 지연을 감수할 수 있는 시스템에 적합하다는 것을 확인했다. 이 두 가지를 접목한 TL 기법은 어떤 요소를 기준으로 설정하느냐에 따라 그 결과가 달라지기 때문에 중립적인 결과값을 갖는다. 앞으로는 어떤 시스템에서든지 시스템 자체 내에서 오류 빈도를 판단하여 그 특성에 맞게 시스템 특성이 조정된다거나 오류 빈도를 조사하여 핸드오프 비율에 따라 시스템이 스위칭될 수 있는 기법을 연구하고자 한다.

참고문헌

[1] D. K. Pradhan, P. Krishan, and N. H. Vaidya, "Recoverable Mobile Environment Design and Trade-off Analysis," Proceedings of the IEEE Fault Tolerant Computing, pp.16-25, 1996.

[2] T. S. Park and N. Y. Woo, "An Efficient Optimistic Message Logging Scheme for Recoverable Mobile Computing Systems," IEEE Transactions on Mobile Computing, pp.265-277, 2002.

[3] M. Chandy and L. Lamport, "Distributed Snapshot: Determining Global States of Distributed Systems," ACM Transactions on Computer Systems, vol. 3, No. 1, pp.63-75, 1985.

[4] B. L. Randell, P. A. Lee, and P. C. Treleaven, "Reliability Issue in Computing System Design," ACM Computing Surveys, pp.123-166, 1978.

[5] T. Park and H. Y Yeom, "Application Controlled Checkpointing Coordination for Fault-Tolerant Distributed Computing Systems," Parallel Computing, pp.467-482, 2000.

[6] L. Alvisi and K. Marzullo, "Message Logging: Pessimistic, Optimistic and Causal," Proceedings of 15th International Conference on Distributed Computing Systems, pp.229-236, 1995.

[7] E. N. Elnozahy and W. Zwaenepoel, "Manetho: Transparent Rollback-Recovery with Low Overhead, Limited Rollback, and Fast Output Commit," IEEE Transactions on Computers, pp.526-531, 1992.

[8] D. B. Johnson, S. W. Smith, and J. D. Tygar, "Completely Asynchronous Optimistic Recovery with Minimal Rollbacks," Proceedings of 25th Symposium, Fault-tolerant Computing Systems, pp.361-370, 1995.

[9] R. Prakash and M. Singhai, "Low-cost Checkpointing and Failure Recovery in Mobile Computing," IEEE Transactions on Parallel and Distributed Computing Systems, pp.1035-1048, 1996.