# 소프트웨어 분산공유메모리의 고장 허용을 위한 원격 로깅 기법*

박소연°, 김영재, 맹승렬
한국과학기술원 전자전산학과 전산학전공
{sypark°, yjkim, maeng}@calab.kaist.ac.kr

# Remote Logging for Fault-Tolerant Software Distributed Shared Memory

Soyeon Park°, Youngjae Kim, Seung Ryoul Maeng
Division of Computer Science, Dept. of Electrical Engineering & Computer Science, KAIST

요 약

소프트웨어 분산공유메모리 시스템의 성능이 높아짐에 따라 최근에는 큰 규모의 클러스터 상에서 사용되는 경우가 많아졌다. 그러나 시스템 규모가 커지면서 고장이 발생하는 가능성도 높아졌다. 시스템의 가용성을 높이기 위하여 고장 허용 기능을 제공하는 분산공유메모리 시스템이 요구되었으며 체크포인팅과 더불어 메시지 로깅에 대한 많은 연구가 이루어져 왔다. 본 논문에서는 고속의 네트웍을 이용하여 원격 노드의 메모리에 로깅하는 방법과 복구 방법을 제안하고 구현을 통하여 성능을 보인다. 원격 로깅은 디스크 접근을 요구하지 않으므로 오버헤드가 적으며 제한적으로 다중 노드의 고장을 허용한다.

## 1. 서 론

소프트웨어 분산공유메모리 (Software Distributed Shared Memory) 시스템은 각 노드에 분산된 메모리를 공유 메모리로 인식할 수 있도록 사용자에게 가상의 이미지를 제공해준다. 최근까지 분산공유메모리 시스템의 성능, 확장성, 프로그램의 용이성을 향상시키기 위해 많은 연구들이 이루어져 왔으며, 그 결과 분산공유메모리 시스템은 큰 규모의 클러스터 상에서 효율적인 병렬처리를 위해 이용될 수 있었다.

한편, 클러스터 시스템의 규모가 점차 커짐에 따라 시스템에서 고장이 발생할 가능성도 높아지게 되었다. 분산공유메모리 시스템에서는 노드들 사이에 데이타가 공유되므로 한 노드에서 고장이 발생해도 전체 노드의 재수행이 요구된다. 따라서 오랜 기간 동작하거나 높은 가용성을 요구하는 응용 프로그램들을 위해 분산공유메모리 시스템에서 고장 허용성을 제공해 주는 것이 중요하다.

고장 허용을 위해 일반적으로 많이 사용되는 방법은 체크포인팅(checkpointing)과 롤백(rollback) 이다. 이는 프로그램 수행 중 주기적으로 시스템 이미지를 안전한 저장 장치에 저장하고, 고장이 발생했을 때 체크포인팅 시점으로 되돌아가서 수행을 계속하는 방법이다. 분산공유메모리 시스템에서는 공유 페이지를 매개로 여러 노드 사이에 의존 관계(dependency)가 형성되므로 한 노드의 롤백이 다른 노드의 롤백을 연속적으로 이끄는 도미노 현상이 발생할 수 있다. 고장 시 다른 노드에 영향을 미치지 않고 복구를 가능하게 하기 위하여, 체크포인팅과 더불어 각 노드 간에 교환되는 메시지를 로깅(logging)하는 방법들이 많이 연구되었다[1,2,3,4]. 그러나 로그 저장을 위해 디스크를 이용하는 방법들은 빈번한 디스크 접근으로 인하여 정상 수행 시 성능을 크게 감소시키는 문제를 가진다[1,2]. 반면, 로그를 메시지 전송 측의 메모리에 저장하는 방법들은 한 노드의 고장만을 가정함으로 적용에 제한적이다[3,4].

본 논문에서는 오버헤드가 적은 원격 로깅 기법과 복구 기법을 제안한다. 원격 로깅은 각 노드 마다 로그 홈이라는 특정한 노드를 할당하여 로그 홈의 메모리에 로깅하는 방법이다. 본 방법은 빈번한 디스크 접근을 요구하지 않으며 로그 홈을 제외한 다른 노드들과 동시에 고장이 발생한 경우에도 복구가 가능하다. 또한, 사용자 계층 통신 프로토콜에서 제공하는 원격 메모리 쓰기 기능을 활용하여 정상 수행 시의 전체 성능 감소를 최소화할 수 있다.

논문의 구성은 다음과 같다. 2절에서는 시스템의 개요를 기술한다. 3절에서는 원격 로깅에 대해 설명하고 4절에서는 복구 방법을 설명한다. 5절에서는 구현된 원격 로깅의 성능을 보이고 6절에서 결론을 맺는다.

## 2. Fault Tolerant KDSM (FT-KDSM)의 개요

FT-KDSM은 기존에 구현된 분산공유메모리 시스템인

* 이 연구는 국가지정연구실 사업의 지원을 받는다.

KDSM-V[5]을 바탕으로 한다. KDSM-V는 VIA 표준에 의거하여 구현된 VI-GM[6]으로 통신하며 메모리 일관성 유지를 위해 Home-based Lazy Release Consistency (HLRC) 모델을 사용한다. FT-KDSM은 고장이 일어나는 경우를 다음과 같이 가정한다. 첫째, 고장-정지(fail-stop) 모델을 가정한다. 즉, 한 노드에서 고장이 발생하면 다른 노드의 상태에 영향을 미치지 않고 정지 한다. 둘째, 노드 사이의 네트웍은 안정적이라고 가정한다. 최근 개발되는 통신 계층들은 그 자체적으로 고장 허용성을 지원하기 때문에 본 논문에서는 네트웍의 고장은 고려하지 않는다. 마지막으로 디스크는 안정적이라고 가정한다.

FT-KDSM에서 각 노드는 독립적으로 체크포인팅을 수행하며 메모리 상태를 변화 시키는 메시지를 다른 노드들로부터 비동기적으로 받았을 때 이를 로깅한다. 각 노드에는 하나의 로그 홈 노드가 지정되는데, 메시지 송신 노드들은 수신 노드의 로그 홈에 같은 메시지를 보내어 로그 홈의 메모리에 로깅한다. 로깅은 VI-GM에서 제공하는 원격 메모리 쓰기 기능을 이용함으로써 로그 홈의 사용자 프로세서에 영향을 주지 않고 적은 비용으로 수행될 수 있다. 고장이 발생한 노드는 마지막 체크포인팅 지점으로 돌아가고 그 이후에 받았던 메시지들을 자신의 로그 홈으로부터 가져와 재적용 시킴으로써 노드의 상태를 고장 발생 이전과 동일하게 변화시켜 간다.

로그 홈에 저장된 로그는 오로지 상대 노드의 복구를 위해서 필요한 정보이기 때문에 상대 노드가 체크포인팅 될 때 메모리에서 삭제되어도 좋다.

## 3. 원격 로깅 (Remote Logging)

다른 노드에 의해 지역 메모리의 내용과 상태가 변화하는 경우는, 지역 페이지를 갱신하는 메시지(Diff)와 원격 메모리 페이지의 복사본을 무효화 시키는 메시지(Write notice)를 받았을 때이다. 이 메시지들은 고장에 대비해 적용 시점에 대한 정보와 더불어 로깅된다.

### 3.1 Diff 로깅

HLRC 모델에서는 페이지 마다 홈 노드를 정하여 가장 최근의 페이지를 유지하도록 한다. 다른 노드들은 페이지 홈으로부터 복사본을 얻으며 interval이 끝날 때 마다 갱신된 내용을 diff 메시지로 만들어 홈 페이지에 적용시킨다.

고장 발생 후 복구 도중에 메모리 읽기가 수행 될 때에는 최근의 메모리 값이 아니라 예전에 읽은 것과 동일한 메모리 값을 읽어야만 고장 발생 이전과 같은 연산 결과를 얻을 수 있다. 원격 로깅에서는 복구 시에 페이지 홈이 예전 시점의 페이지를 다시 만들 수 있도록, 정상 수행 시 diff 메시지를 페이지의 홈과 로그 홈 두 곳에 저장 한다. 그림 1은 diff 로깅의 예를 보여준다.

고장이 발생한 노드는 interval이 시작될 때 자신의 로그 홈으로부터 diff 로그를 가져와 지역 홈 페이지를 갱신 한다. 원격 메모리 접근 시에는 해당 페이지 홈에 요청하여 그의 메모리에 저장된 diff 로그로부터 필요한 예전 페이지를 재생성해서 가져온다. 즉, 페이지의 홈에 diff 메시지를 저장하는 이유는 자신을 제외한 다른 노드의 고장에 대비하기 위해서이다. 따라서 로그 홈에 저장된 로그와는 달리 페이지 홈에 저장된 diff 로그는 체크포인팅 때에 삭제되지 않고 안정적인 저장 장치에 기록되어야 한다. 그리고 기본적인 HLRC 에서와는 다르게 페이지 홈이 자신의 로컬 페이지에 쓰기를 수행할 때에도 diff를 생성하여 자신의 메모리에 로깅한다.

### 3.2 Write notice 로깅

락(Lock)을 다른 노드에 넘겨 주거나 베리어(barrier) 연산을 할 때에는 그 이전 interval에 쓰기를 수행했던 페이지의 목록(Write notice)을 해당 노드에 보내어 복사본을 무효화 시킨다. 고장 허용을 위하여 write notice 메시지는 수신 노드의 로그 홈에 중복하여 보내진다. 이 로그는 오로지 수신 노드의 복구를 위해서 필요한 것이므로 diff와는 다르게 로그 홈에만 로깅된다.

그림 2는 락 수행 시 write notice 로깅 방법을 보여준다. Pj는 write notice와 다른 노드의 interval 정보를 나타내는 타임스탬프 값을 받으면 이를 로그 홈에 보내어 순차적으로 저장한다.

## 4. 고장으로부터의 복구

한 노드에서 고장이 발생하면 다른 노드들은 그 노드와의 통신에서 실패하게 되므로 고장 발생을 알게 되며 그 노드와 새롭게 커넥션을 맺은 후 로그 및 예전 페이지의 요청에 응답한다. 고장이 발생한 노드는 마지막 체크포

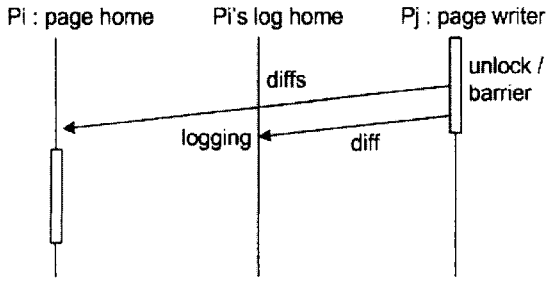


그림 1. Diff 로깅 방법

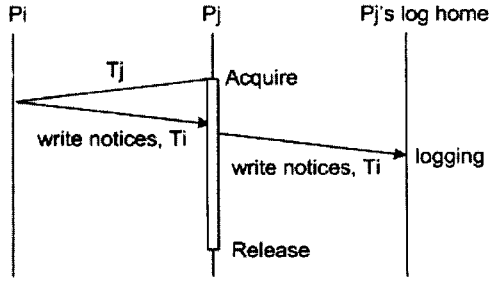


그림 2. Lock write notice 로깅 방법

인트의 메모리 및 프로세스 상태로 롤백하고 그 시점부터 응용 프로그램을 계속 수행하며 복구한다.

복구 도중 새로운 interval이 시작될 때 마다 그 interval 이전까지 받았던 diff 메시지들을 로그 홈으로부터 가져와 지역 홈 페이지를 갱신한다. 원격 페이지 폴트가 발생하면 해당 홈 페이지에 요청 메시지를 보낸다. 페이지 홈은 체크포인트로부터 페이지의 초기 값을 얻고 요구된 시점까지의 diff 로그를 적용시켜 페이지를 재생성 한다. diff 로그에는 그 diff가 생성된 때의 interval과 쓰기를 수행한 노드의 정보가 기록되어 있으므로 요구된 interval과의 비교를 통해 필요한 로그를 찾을 수 있다. 만일 원격 페이지의 홈이 동시에 고장난 경우라도 같은 diff 로그가 페이지 홈의 디스크나 로그 홈에 저장되어 있으므로 복구가 가능하다.

락이나 베리어에 도달하면 해당 interval에 대한 정보를 로그 홈에 보내어 write notice 로그를 얻고, 페이지를 무효화 시킨다.

모든 로그를 다 적용 시키고 복구가 끝나면 이를 다른 노드에게 알려서 정상 수행을 계속한다. 고장 발생으로 인해 통신에 실패했던 노드들은 같은 통신 메시지를 다시 보낸다.

## 5. 성능 평가

본 논문에서는 8대의 Pentium III 850MHz 컴퓨터로 구성된 클러스터 상에서 FT-KDSM 시스템의 성능을 측정하였다. 각 노드는 LANai9.1 프로세서를 장착한 Myrinet 통신망으로 연결된다. 벤치마크로는 SPLASH의 water, barnes, raytrace, FFT와 TSP를 사용하였다. 표 1은 각 응용프로그램에서의 입력 데이터 크기를 보인다.

그림 3은 16노드에서 측정된 FT-KDSM의 실행 시간을 고장 허용 기능이 없는 KDSM-V의 성능에 정규한 것이다. 본 실험은 원격 로깅의 오버헤드를 측정하기 위한 것이므로 체크포인팅은 수행하지 않았다. 실험 결과 원격 로깅으로 인한 전체 성능 감소는 1%~9% 범위 내임을 확인하였다. 이는 로깅을 위한 메시지 수의 증가와 지역 홈 페이지에 대한 추가적인 diff 생성에 의한 성능 감소이다. 시스템 상에 메시지 양이 많아지는 베리어 시점에서는 추가적인 로그 메시지로 인한 네트웍 지연이 성능을 감소시킬 수 있다. 그러나 본 실험 결과는 빠른 네트웍과 원격 메모리 쓰기 기능을 활용하는 경우 원격 로깅이 빈번한 디스크 접근 없이 적은 비용으로 고장 허용을 지원할 수 있음을 나타낸다.

표 1. 벤치마크 프로그램과 입력 데이터 크기

| | |
|---|---|
| water | 1728 mols, 5steps |
| barnes | 64k bodies |
| raytrace | car |
| TSP | 20 cities |
| FFT | 128 x 128 x 64 data points |

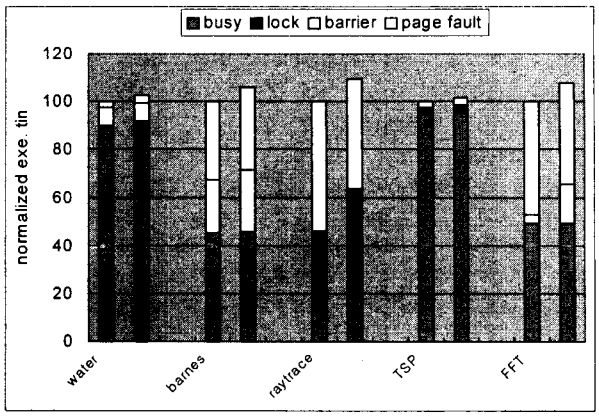


그림 3. FT-KDSM 상에서 원격 로깅으로 인한 성능 감소

## 6. 결 론

본 논문에서는 고장 허용성을 제공하는 소프트웨어 분산 공유메모리 시스템인 FT-KDSM을 구현하였다. 본 시스템은 독립적 체크포인팅과 원격 로깅을 통하여 고장에 대비한다. 원격 로깅 기법은 빈번한 디스크 접근 없이 메시지 로그를 특정 노드의 메모리에 저장함으로써 정상 수행 시의 성능 감소를 최소화 한다. 또한 로그 홈을 제외한 여러 노드에서 동시에 고장이 발생한 경우에도 복구 가능케 하여 시스템의 가용성을 높인다.

## 참고 문헌

[1] G.Suri, B.Janssens, and W.K.Fuchs. Reduced Overhead Logging for Rollback Recovery in Distributed Shared Memory. In Proceedings of the 25th Annual International Symposium on Fault-Tolerant Computing, June 1995.

[2] A.Kongmunvattana and M.F.Tzeng. Coherence Centric Logging and Recovery for Home-based Software Distributed Shared Memory. In Proceedings of the International Conference of Parallel Processing, Sept 1999.

[3] F.Sultan, T.D.Nguyen, and L.Iftode. Scalable Fault-tolerant Distributed Shared Memory. In Proceedings of Supercomputing, 2000.

[4] S.Kanthadai and J.L.Welch. Implementation of Recoverable Distributed Shared Memory by Logging Writes. In Proceedings of the 16th International Conference on Distributed Computing Systems, May 19996.

[5] 박소연, 김영재, 이상권, 맹승렬. " VIA (Virtual Interface Architecture)를 기반으로 하는 소프트웨어 분산공유메모리 시스템의 설계 및 구현" , 제 29회 한국정보과학회 춘계 학술발표논문집(A), 2002.4.

[6]http://www.myri.com/scs/index.html