VIA(Virtual Interface Architecture)를 기반으로 하는 소프트웨어 분산공유메모리 시스템의 설계 및 구현 †

박소연°, 김영재, 이상권, 맹승렬

한국과학기술원 전자전산학과 전산학전공

Design and Implementation of Software Distributed Shared Memory(DSM) over Virtual Interface Architecture(VIA)

Soyeon Park°, Youngjae Kim, Sang-Kwon Lee, Seung Ryoul Maeng

Division of Computer Science, Department of Electrical Engineering & Computer Science, KAIST

{sypark, yjkim, sklee, maeng}@camars.kaist.ac.kr

요 약

최근에는 고성능 네트웍으로 구성된 클러스터 상에서 사용자 수준 통신을 사용하는 소프트웨어 분산 공유메모리 시스템의 연구가 활발히 진행되고 있다. 본 논문에서는 사용자 수준 프로토콜의 표준인 Virtual Interface Architecture(VIA)를 사용하고 확장성있는 Home-based Lazy Release Consistency(HLRC) 모델을 기반으로 하는 소프트웨어 분산공유메모리 시스템을 구현한다. 본 시스템은 VIA의 원격 메모리 쓰기 기능을 최대한 활용하며, 통신 과정에서 통신 버퍼와 사용자 메모리 사이의 복사가 일어나지 않도록 설계되어 높은 성능을 보인다.

1. 서론

소프트웨어 분산공유메모리 (Software Distributed Shared Memory) 시스템은 PC나 웍스테이션으로 구성된 클러스터 상에서 공유 가상 메모리를 제공함으로써 병렬 처리를 용이하게 한다.

일반적으로 소프트웨어 분산공유메모리 시스템은 페이지 단위로 노드 사이에 데이타를 공유하고 일관성을 유지한다. 그러나 페이지 단위는 실제 응용 프로그램 상에서 공유되는 데이타 단위 보다 크기 때문에 거짓 공유 문제를 야기시킨다. 이는 통신량을 증가시키는 원인이 된다. 최근 제안된 *Home-based Lazy Release* Consistency(HLRC) 모델은 완화된 일관성 유지를 통해 통신량을 줄이고 홈 노드의 개념을 도입하여 확장성을 높였다[1].

그러나 소프트웨어 분산공유메모리 시스템의 통신 오버헤드는 여전히 전체 성능에 크게 영향을 미친다. 성능 향상을 위하여, 최근에는 고성능 네트웍으로 구성된 클러스터 상에서 사용자 수준 통신(user level communication)을 사용하는 소프트웨어 분산공유메모리 연구가 활발히 진행 중이다[2, 3]. Virtual Memory-Mapped *Communication*(VMMC)를 바탕으로 GeNIMA[2] 시스템이 구현되었으며 사용자 수준 통신 프로토콜의 표준인 Virtual Interface Architecture(VIA)를 이용하는 HLRC 기반 시스템이 제안되었다[3]. 사용자 수준 통신은 TCP/IP 프로토콜과는 달리, 송수신 과정에서 커널의 관여를 배제하고 메세지 복사의 과정 없이 사용자와 네트워크 장치 사이에서 직접 메세지를 전달하므로 통신 오버헤드가 작다.

본 연구에서는 통신 계층으로 VIA를 사용하고 HLRC 모델을 기반으로 하는 KDSM-V(KAIST Ditributed Shared Memory over VIA) 시스템에 대해 기술한다. 본 시스템은 VIA의 원격 메모리 쓰기(RDMA write) 기능을 최대한 활용하여 통신 오버헤드를 줄인다.

논문의 구성은 다음과 같다. 2절에서는 KDSM-V 시스템의 개요를 기술한다. 3절에서는 VIA를 이용한 성능 향상 방법 및 설계에 대해 설명하고 4절에서 구현된 시스템의 성능 결과를 설명한다. 그리고 5절에서 결론을 맺는다.

2. KDSM-V 시스템 개요

KDSM-V 시스템은 Linux 2.2.15 커널에서 구현되었으며 VIA 표준에 의거하여 구현된 VI-GM[4]을 사용하여 통신한다. 캐쉬 일관성 유지를 위해 페이지 기반 무효화 프로토콜을 사용하고 HLRC 모델을 바탕으로 다중 쓰기(multiple writer) 프로토콜을 지원한다. 본

†이 연구는 국가지정연구실 사업의 지원을 받는다.

시스템은 pthread 라이브러리를 사용하며 응용 쓰레드(application thread)와 통신 쓰레드(communication thread)의 두 쓰레드로 동작한다. 전자는 응용 프로그램을 수행하며 필요한 경우 원격 노드로 데이타 요청 메세지를 보낸다. 후자는 VIA 의 수신 통지(receive notification) 함수를 통해 생성되며 원격 노드로부터 데이타의 요청 및 갱신 메세지를 받았을 때 이를 인지하고 핸들러를 수행한다. 두 쓰레드 사이에서 공유되는 데이터 구조는 pthread의 lock함수를 통해 보호된다.

3. KDSM-V 설계 및 구현

3.1 HLRC의 구현

KDSM-V의 이전 버전인 KDSM[5] 시스템은 TCP/IP의 통신 계층을 이용하여 HLRC를 구현한 시스템이다. KDSM-V는 KDSM의 HLRC 설계를 기반으로 한다.

HLRC는 응용 프로그램의 수행을 동기화 지점(lock release 및 barrier)에 따라 interval로 나눈다. 페이지가 변경될 때 twin을 만들고 interval이 끝날 때 diff를 생성하여 해당 홈 노드에 전송한다. 본 시스템은 토큰을 이용한 분산 lock 프로토콜을 사용하는데 원격 노드로 lock 토큰을 넘겨 줄 때, 이전 interval에서 변경된 페이지들의 정보(write-notice)를 같이 전송하여 캐쉬된 페이지가 무효화 되도록 한다. 무효화된 페이지에 접근하면 페이지 폴트가 발생하며 이 때는 홈 노드에서 최신의 페이지를 가져오게 된다.

3.2 성능 향상 기법

3.2.1 VIA 통신 계층

다음은 KDSM-V가 사용하는 VIA의 송수신 모델이다.

- 기본 송수신(primitive send/recv)
  KDSM-V는 초기화 과정에서, VIA를 통해 송수신 될 메세지의 종류와 대상 노드 별로 여러 통신 버퍼를 생성하고 등록한다. 그리고 각 통신 버퍼의 주소를 기본 송수신을 통해 각 대상 노드에 알린다. 이 후, 대상 노드는 이 주소를 이용하여 원격 메모리 쓰기를 할 수 있다.
- 원격 메모리 쓰기(RDMA write)
  이 송수신 모델은 수신 노드의 특정 메모리를 갱신하되 사용자 프로세스가 메세지의 수신을 인식할 필요가 없는 경우 이용된다. 즉, 수신 노드는 특정 핸들러를 수행하거나 응답 메세지를 보내지 않는다.
- Immediate data 필드를 이용한 원격 메모리 쓰기
  데이타 요청 메세지들은 필요에 따라 비동기적으로 생성되므로 기본 송수신 모델을 사용할 수 없다. 또한 수신 노드는 메세지의 도착을 인식할 수 있어야 한다. KDSM-V는 송신 서술자(descriptor)의 Immediate data 필드를 셋하고 원격 메모리 쓰기를 수행함으로써 수신 노드가 메세지의 도착을 알수 있도록 한다. 수신 노드는 해당 핸들러를 수행하여 메세지를 처리한다.

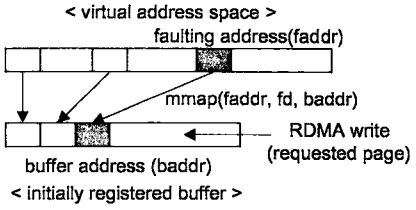

그림 1: 무복사 페이지 전송 프로토콜

3.2.2 페이지 전송

처음으로 페이지에 접근하거나 무효화된 페이지에 접근할 때 페이지 폴트가 발생한다. 처음 페이지에 접근하는 경우에는 페이지 폴트 핸들러가 mmap()을 통해 메모리를 할당 받고 폴트난 주소를 그 메모리로 매핑 시킨다. 두 경우 모두 페이지 폴트 핸들러가 홈 노드에 페이지를 요청하여 최신 페이지를 가져온다. 이 때, 일반적인 분산 공유메모리 시스템에서는 데이타를 수신 노드의 통신 버퍼로 전송 받은 후 할당 받은 메모리 영역으로 복사하게 된다. 복사를 피하기 위해서는 직접 폴트난 가상 주소에 데이타를 전송하도록 하면 된다. 그러나, TCP/IP는 원격 메모리로의 직접 쓰기 기능을 제공되지 않으므로 이는 불가능하다. VIA를 사용하는 경우, 통신에 사용되는 메모리 영역은 반드시 등록되어야 하므로 미리 폴트가 날 수 있는 전체 가상 메모리 영역을 모두 등록하거나 폴트 날 때 마다 등록해야 한다. 전자의 방법은 가상 메모리 사용량이 많은 응용 프로그램에는 적용할 수 없으며 후자는 등록 회수의 제한 및 높은 오버헤드로 인해 효과적이지 못하다. 복사는 수신 노드 뿐 아니라 송신 노드의 가상 메모리와 송신 버퍼 사이에서도 발생한다.

KDSM-V에서는 그림 1과 같은 방법을 통해 무복사 페이지 전송을 구현한다. 초기화 과정에서 미리 큰 크기의 버퍼를 할당하고 이를 등록한다. 본 구현에서는 128M의 버퍼를 사용한다. 그리고 페이지 폴트 발생 시 임의의 메모리가 아닌 등록된 버퍼로 폴트난 가상 주소를 매핑시킨다. 이 기능은 mmap() 함수에서 제공되며 폴트난 가상 주소, 매핑할 버퍼 내 주소, 버퍼 서술자(descriptor) 정보를 이용한다. 폴트난 주소가 등록된 메모리 영역으로 매핑되었으므로 홈 노드는 폴트난 주소로 직접 원격 메모리 쓰기를 수행할 수 있다. 또한, 홈 노드의 페이지 역시 등록된 버퍼내에 존재하므로 데이타 송신 과정에서도 복사를 피할 수 있다. 버퍼의 크기보다 더 많은 양의 페이지에서 폴트가 발생하면 unmap()을 사용하여 버퍼를 재사용 할 수 있으나 본 구현에서는 이를 고려하지 않는다.

3.2.3 동기화 프로토콜

Lock을 해제하는 노드는 lock을 요청한 노드에게 write-notice를 전송한다. 그리고 barrier에 도달했을 때에도 모든 노드가 서로 write-notice를 교환하여 페이지를 무효화 시킨다. 이 때, 송신 노드는 수신 노드의 특정 통신 버퍼로 원격 메모리 쓰기를 하여 write-notice 메세지들을 전송한다. 버퍼가 차게되는 마지막 메세지는 immediate data 필드를 이용하여 전송하고 이를 받은 수신 노드는 버퍼에 수신된 모든 write-notice를 읽

ocean	258 X 258 grid
water	1728 mols, 5steps
barnes	64k bodies
FFT	128 X 128 X 64
SOR	2048 X 2048, 100 iteration

표 1: 벤치마크 프로그램과 입력 데이타 크기

어서 페이지에 적용시킨다. 그리고 원격 메모리 쓰기로 송신 노드의 특정 플레그(flag)를 셋하여 버퍼 재사용이 가능하다는 사실을 알린다.

제안된 방식은 write-notice의 무복사 전송을 가능하게 하며 수신 노드에서 핸들러 호출이 빈번히 발생하는 것을 방지하여 비용을 최소화 한다.

3.2.4 Diff 전송

Diff는 한 interval 동안 페이지가 변경된 내용으로써 동기화 시점에서 홈으로 전송된다. 송신 노드는 immediate data 필드를 이용하여 원격 메모리 쓰기로 diff를 전송한다. 홈은 복사 과정 없이 버퍼에서 바로 diff를 읽어서 페이지에 적용시킨다. 버퍼를 재사용 할 수 있게 되면 송신 노드의 특정 플레그를 셋하여 이를 알린다. 송신 노드는 홈으로부터 응답(ack)을 기다리지 않고 다음 작업을 수행할 수 있지만, 만일 여러 diff 메세지가 보내져야 하는 경우 홈 노드의 버퍼를 재사용 할 수 있을 때까지 기다려야 한다. KDSM-V는 이러한 송신 지연 시간을 줄이기 위해 수신 노드에 두 개의 diff 버퍼를 할당하여 번갈아 사용하도록 구현하였다. 즉, 송신 노드는 홈 노드가 이전 diff의 적용을 마치지 않았을 때에도 다른 버퍼로 다음 diff 메세지를 전송 할 수 있다.

4. 성능평가

본 논문에서는 16대의 Pentium III 850MHz 컴퓨터로 구성된 클러스터 상에서 KDSM-V 시스템의 성능을 측정하였다. 각 노드는 LANai 9.1 프로세서를 장착한 Myrinet 통신망으로 연결 된다. 벤치마크로는 SPLASH2[6]의 ocean, water, barnes, FFT와 Rice 대학의 SOR를 사용하였다. 표 1은 각 응용 프로그램에서의 입력 데이타 크기를 보인다.

그림 2는 16노드에서 측정된 KDSM-V의 실행 시간을 KDSM[5]의 성능에 정규화하여 나타낸다. KDSM[5]은 Myrinet GM[4]을 바탕으로 구현된 TCP/IP 통신 프로토콜을 사용한다. 그림에서 base는 제안된 성능 향상 기법이 적용되기 전의 성능으로써, 페이지의 전송과 동기화 시점에서 모두 복사가 발생할 때의 성능이다. 실험 결과로부터 VIA를 이용한 구현이 TCP/IP를 이용하는 KDSM 비해 성능이 좋음을 알수 있다.

그림 3은 16 노드에서 KDSM-V 성능을 base의 성능에 정규화하여 보인다. 각 응용 프로그램에서 왼쪽 그래프는 성능 향상 기법이 적용되기 전의 결과를 나타낸다. 모든 프로그램에서 KDSM-V가 base에 비해 좋은 성능을 나타낸다. 특히 KDSM-V에서 페이지 전송시 페이지 복사가 일어나지 않게 되므로 페이지를 원격 노드에서 가져오는 시간이 줄었다. ocean의 경우 KDSM-V의 전체 실행 시간이 base에 비해 13.4% 감소

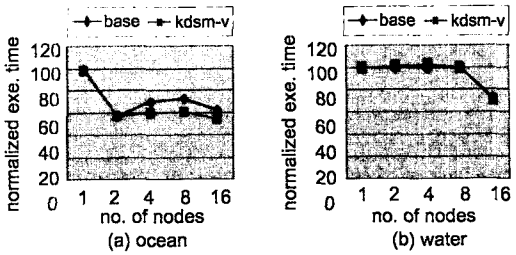

그림 2: TCP/IP를 사용하는 구현과의 비교

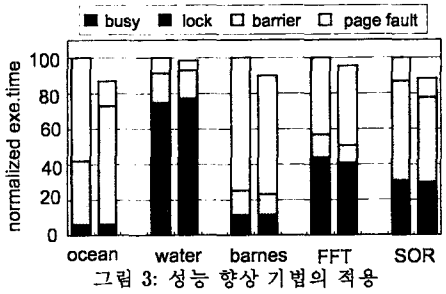

그림 3: 성능 향상 기법의 적용

하였다.

5. 결론

본 논문에서는 VIA 통신 계층 상에서 동작하는 분산공유메모리 시스템인 KDSM-V를 구현하였다. 본 시스템은 고성능 통신망을 바탕으로 VIA에서 제공하는 원격 메모리 쓰기 기능을 최대한 활용하였다. 또한 통신과 관련된 메모리 복사를 피함으로써 통신 오버헤드를 줄였다.

참고 문헌

[1] Y.Zhou, L.Iftode, and K.Li. Performance Evaluation of Two Home-Based Lazy Release Consistency Protocols for Shared Virtual Memory Systems. In *Proceedings of the OSDI Symposium*, October 1996.

[2] A.Bilas, C.Liao, and J.P.Singh. Using Network Interface Support to Avoid Asynchronous Protocol Processing in Shared Virtual Memory Systems. In *Proceedings of the 26th ISCA*, 1999.

[3] M.Rangarajan and L.Iftode. Software Distributed Shared Memory over Virtual Interface Architecture: Implementation and Performance. In *Proceedings of 4th Annual Linux Conference*, October 2000.

[4] http://www.myri.com/scs/.

[5] S.K.Lee, H.C.Yun, and J.W.Lee. Design and Implementation of HLRC Protocol for Software Distributed Shared Memory System. In *Proceedings of KISS Conference*, April 2000.

[6] S.Woo et al. The SPLASH2 Programs: Characterization and Methodological Considerations. In *Proceedings of ISCA*, 1995.