소프트웨어 분산 공유메모리 시스템 상에서 효율적인 일관성 모델

○
김 태 규, 홍 영 식
동국대학교 컴퓨터공학과

An Efficient Consistency Model for Software Distributed Shared Memory Systems

Tae Gyu Kim, Young Sik Hong
Dept. of Computer Engineering, Dongguk Univ.

요 약

분산 공유메모리 시스템(DSM)의 성능 향상을 위해 일관성 모델의 측면에서 많은 연구가 진행되었다. 분산 공유메모리 시스템의 성능을 저하시키는 가장 큰 요인은 거짓공유 문제와 별도의 통신비용 문제를 들 수 있는데, 동기화 연산에 의한 일관성 유지방법, 홈-기반 접근방법 등의 보다 완화된 메모리 모델로서 이러한 문제점을 해결하려는 연구가 진행되어 왔고, 어느 정도 타당한 결과를 보였다. 본 논문에서는 동기화 연산에 의한 일관성 모델을 기초로 동적 홈-기반 접근방법을 제안하며, 이것은 홈에서의 이점 및 부하를 여러 프로세서에게 분산시켜 시스템 전반의 성능향상을 가져온다.

1. 서론

분산 시스템에서는 공유메모리를 구성하는 것이 어렵기 때문에 일반적으로 메시지 전송을 통해 프로세서간 통신이 이루어지고 있다. 그러나 메시지 전송에 의한 통신은 기존 병렬 알고리즘의 수정이 필요하며, 프로그래밍의 복잡도를 증가시키는 등 여러 단점이 있기 때문에 분산시스템 상에서 가상의 공유 메모리를 구성하는 분산 공유메모리 시스템이 제시되었다 [3].

분산 공유메모리 시스템을 소프트웨어적으로 구성하는 경우, 프로토콜 상의 오버헤드와 프로그램 수행과는 관계없는 별도의 통신이 시스템 성능을 저하시키고 있는데, 특히 각각의 프로세서가 소유한 메모리 복사본들의 일관성 유지를 위해서는 프로세서간의 많은 통신이 필요하다. 따라서 이러한 일관성 유지비용을 줄이기 위해, 메모리 참조의 모든 상태에서 일관성을 유지하기보다는 필요한 부분만의 일관성을 고려하는 완화된 메모리 모델들이 제안되었다. 그 대표적인 예로 동기화 연산에 기초한 약성 일관성(weak consistency), 연성 일관성(release consistency), 지연 연성 일관성(lazy release consistency, LRC) 모델과, 홈-기반 접근방법을 사용하는 자동-갱신 연성 일관성(automatic update release consistency, AURC), 홈-기반 지연 연성 일관성(home-based lazy release consistency, HLRC) 모델들이 있다.

본 논문에서는 동기화 연산에 의한 일관성 모델에 기초한 동적 홈-기반 접근방법을 제안하며, 이것은 분산 공유메모리 상에서 페이지 부재 회수를 줄임으로써 성능향상을 가져올 수 있다. 2장에서는 기존에 연구되었던 소프트웨어 분산공유메모리 시스템의 일관성 모델에 대해 살펴보고, 3장에서는 동기화 연산에 의한 일관성 모델로써 본 논문에서 제안된 동적 홈-기반 정책을 설명한다. 4장에서는 실험을 통해 기존의 일관성 모델과 비교하여 성능 평가하고, 5장에서 결론을 맺는다.

2. 기존연구

순차적 일관성(sequential consistency)과 프로세서 일관성(processor consistency) 같은 엄격한 일관성 모델의 단점으로 프로세서간 많은 통신과 프로토콜 상의 오버헤드를 들 수 있는데, 이런 단점을 해결하기 위하여 완화된 일관성 모델들이 제안되어 왔다. 연성 일관성 모델[1]에서는 프로그램 수행 중 충돌을 일으키는 자원에 대해서만 동기화 연산을 통해 일관성을 유지하는 정책을 사용하고 있다. 이러한 일관성 모델에서는 자원과 관련된 동기화 연산에 의해 해당 자원에 접근하는 구간을 형성할 수 있으며, 이러한 구간 사이에서만 일관성을 유지한다. 동기화 연산에 의한 일관성 유지방법은 기존의 엄격한 모델과 동일한 프로그램 수행 결과를 나타내지만 부가적인 메시지 전송 및 프로토콜 상의 오버헤드를 크게 줄일 수 있다

TreadMarks 분산 공유메모리 시스템[2]에서 제안된 지연 연성 일관성 모델은 기존의 동기화 연산에 의한 일관성 유지 방법에 덧붙여 공유메모리의 변경된 내용 전송이 다음에 수행되는 동기획득(acquire) 연산이 일어날 때까지 지연되는 모델이다. 이와 같은 방법은 일관성을 유지하기 위한 자료 전송이 프로세서가 필요할 때까지 지연되기 때문에 메시지 수를 현격히 줄일 수 있으며 프로토콜에 대한 오버헤드도 상대적으로 적다. 또한 자료 전송 시 페이지 내용 전체를 전송하지 않고 갱신이 일어난 부분을 취합하여 변경부분(diff)을 생성 후 이것을 전송하는 방법으로 메시지 전송시간을 줄이고 있다.

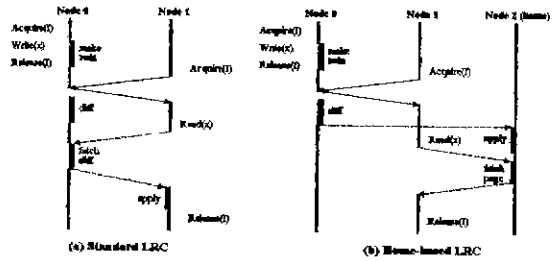
LRC 모델에서는 지연된 자료의 전송 이외에 자원의 변경 부분(diff)을 이용한 다중 기록자 프로토콜을 사용함으로써 거짓 공유(false sharing) 문제를 어느 정도 해결하고 있다. 즉 서로 다른 프로세서가 같은 공유자원을 접근하고 있지만 실제 공유하는 부분이 다른 경우 이 두 프로세서에 대해 동시에 자원의 변경을 허용함으로써 실제 공유하지 않는 부분의

불필요한 전송을 줄이는 방법이다.

이러한 다중 기록자 프로토콜은 많은 이점을 갖고 있지만 변경부분을 생성하는 것에 오버헤드가 있고, 다중 기록을 허용하기 때문에 동기획득을 하려는 프로세서는 여러 프로세서로부터 각각의 변경부분을 전송 받아야 하는 문제가 있다. 따라서 많은 수의 프로세서가 분산 공유메모리 시스템에 참여하는 경우 이러한 오버헤드 때문에 그 성능향상이 미비한 것으로 연구되었다 [6].

Shrimp 시스템[6]에서는 자원의 변경부분을 생성하기 위한 연산을 줄이고 다중 기록자 프로토콜을 사용하지만 하나의 프로세서(home)에서는 항상 모든 변경부분이 적용되어 유효한 복사본을 유지하는 홈-기반 접근방법이 연구되었다. 자동-갱신 연성일관성(AURC) 과 홈-기반 지연 연성 일관성(HLRC) 모델이 그것이다.

[그림 1.] 에서와 같이 HLRC 모델의 경우 홈(home)은 자



[그림 1.] Home-Based Lazy Release Consistency

원의 변경부분을 모두 전송 받아서 항상 유효한 복사본을 유지하고, 동기획득을 원하는 프로세서는 홈으로부터 공유페이지 전체를 전송 받는 방법을 사용하고 있다. 이러한 방법은 변경부분 연산 오버헤드가 많이 줄어들고, 홈 프로세서에서는 페이지 부재가 일어나지 않으며, 동기획득을 원하는 프로세서는 한번의 메시지 전송으로 유효한 복사본을 전송 받을 수 있는 이점을 갖는다. 따라서 응용 프로그램에 따라 홈을 적절히 지정하면 홈의 이점에 의한 성능향상을 가져올 수 있는데, 예를 들어 읽기 연산 빈도가 높은 다수의 프로세서와 쓰기 연산 빈도가 높은 하나의 프로세서가 있는 경우 쓰기 연산 빈도가 높은 프로세서를 홈으로 지정하면 홈에 의한 성능향상을 기대할 수 있다[6].

홈 프로세서의 이점을 살펴보면 홈이 아닌 프로세서에서 페이지 부재가 일어난 경우 페이지를 전송 받는 비용은 아래와 같고

$$C_F = T_f + T_m + T_t + T_{tr} + T_m$$

T_f : page fault time T_m : message latency

T_t : receive trap time T_{tr} : page transfer time

홈 프로세서인 경우,

$$C_F = 0$$

이지만 유효한 복사본을 유지하기 위해 쓰기 연산이 일어난 프로세서로부터 변경부분을 전송 받아야 하는 다음의 오버헤드가 있다.

$$C_U = W \times P_w \times (T_t + T_a)$$

P_w : 다른 프로세서에서 write가 일어날 확률

T_a : diff application time

W : 프로그램 수행중 전체 write 회수

프로세서가 프로그램 수행 중 오랜 시간 홈의 역할을 하는 경우 홈의 이점을 가질 수 있지만 적절하게 홈이 지정되지 못한 경우는 위 오버헤드가 페이지 부재 시 얻는 이점보다 클 수 있으며 이것은 시스템 성능을 저하시킬 수 있다.

3. 동적 홈-기반 지연 연성 일관성 모델 (dynamic home-based LRC, DHLRC)

본 논문에서 제안하는 동적 홈-기반 지연 연성 일관성 모델(DHLRC)은 기본적으로 HLRC가 갖는 모든 조건을 만족한다. 기존 모델과 다른 점은 하나의 프로세서에 홈을 고정시키는 것이 아니라 프로그램 수행에 따라 홈 권한을 동적으로 지정한다는 점이다.

DHLRC의 기본 정책은 별도의 메시지 없이, 즉 기존의 동기 획득연산과 페이지 요구 연산 메시지를 이용하여 홈을 페이지 부재가 자주 일어나는 프로세서로 이전하는 것이다. 이를 위해 각 프로세서는 자신의 페이지 부재 회수를 기록하고 일정 임계값을 넘으면 기존의 홈 프로세서에게 이전을 요청한다. 후에 다른 프로세서에서 페이지 부재가 자주 일어나면 다시 홈이 해당 프로세서로 이전될 수 있다.

이때 얻을 수 있는 이점으로는 홈에서 페이지 부재가 일어나지 않는 점을 모든 프로세서가 공유할 수 있다는 것인데, 이것은 페이지 부재가 자주 일어나는 프로세서에게 페이지의 홈 권한을 이전하여 페이지 부재를 줄일 수 있게 한다. 또한 이러한 홈의 이전은 만약 프로그램 수행 초기에 지정되었던 홈이 더 이상 홈과 관련된 페이지에 접근을 하지 않는 경우, 홈의 권한을 페이지 부재가 자주 일어나는 프로세서로 이전함으로써 계속해서 변경부분을 전송 받아야 하는 오버헤드를 줄일 수 있다.

DHLRC는 다음과 같이 동기화 연산 및 페이지 요구 메시지에 홈 이전 메시지를 포함하는 방법을 사용하여 추가적인 메시지 없이 홈을 이전한다는 특징을 갖는다. 홈이 아닌 프로세서에서 홈 이전을 요청하는 경우,

page request message + home migration message

를 하나의 메시지로 보낸다. 이때 별도의 메시지를 발생시키지 않도록 하기 위해 읽기 및 쓰기 실패에 따라 다른 이전 방법을 사용한다. 읽기 실패인 경우 바로 홈을 이전 받는 것이 아니라 일단 홈 이전을 예약(reserve) 하고 후에 기존의 페이지 소유자에게 동기 획득연산으로 홈의 변경을 알리고 홈을 이전한다. 즉 아래와 같은 메시지를 사용한다.

홈의 예약:

page request message + home reserve message

홈의 알림:

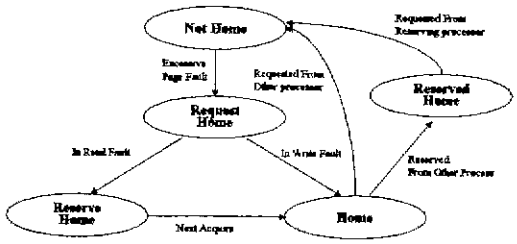
lock acquire message + home notify message

홈의 이전:

page request message + home migration message

쓰기 실패로 인한 홈 이전인 경우 자신이 페이지의 소유자가 될 것이기 때문에 페이지 요구 메시지를 이용하여 바로 홈의 이전만을 행한다.

각각의 프로세서는 홈과 관련하여 다음과 같은 상태를 갖게 된다.



[그림 2] State of Processor

홈이아닌 프로세서는 과도한 페이지 부재로 인해 홈 프로세서에게 홈-이전요청 상태로 전이되는데 이때 읽기에 의한 실패인 경우 예약상태(reserve)로 전이되고, 쓰기에 의한 실패인

경우 바로 홈을 이전 받아 바로 홈의 권한을 가질 수 있다. 일단 홈의 권한을 갖으면 홈 상태를 유지하다가 다른 프로세서로부터 홈 이전 예약을 받아 예약된(reserved) 상태로 전이되며 예약된 프로세서로부터 이전 요청을 받으면 홈 권한을 넘겨주어 홈이 아닌 상태로 전이된다.

4. 실험 결과 및 분석

본 논문에서 제안한 일관성 모델의 성능평가를 위해 Mint 시뮬레이터를 사용하여 시뮬레이터의 후반부 부분에 일관성 프로토콜을 구현하여 실험하였다[5]. 실험에 대한 기본적 연산의 인자는 [표 1.]과 같다. 성능을 측정하기 위해서는 SPLASH-2 벤치마크 프로그램을 사용하였고 응용 프로그램으로는 Water Nsquared, Barnes-Hut을 사용하였으며 문제 크기는 [표 2.] 와 같다

[표 1.] Basic Operation Cost

Operation	microsecond
Message Latency	50
Page Transfer	92
Receive Interrupt	430
Twin Copy	120
Diff Creation	560
Diff Application	215
Page Fault	290
Page Invalidation	200

[표 2.] Problem Size

Application	ProblemSize
Water-Nsquared	343 mole
Barnes-Hut	tk14.0
MP3D	20000 mols

[표 3.] Simulation Result on Processor 8

App.	Alg.	Exec.Time	SpeedUp	PageFault	Message
Water Nsquared	LRC	3059372	4.11	583	9938
	HLRC	2921701	4.18	512	9848
	DHLRC	2808100	4.32	479	9364
Barens Hut	LRC	26701260	3.16	44335	79744
	HLRC	25351503	3.33	40302	95353
	DHLRC	22600328	3.73	38513	87334
MP3D	LRC	1594843	4.1	1872	4230
	HLRC	1655223	3.95	1623	4347
	DHLRC	1408692	4.64	1331	4010

[표 3.]은 각 일관성 모델에 대해 두 가지 벤치마크 프로그램에 대한 실험 결과이다. 8개의 프로세서를 사용하였을 때 결과이며, 프로세서 수에 따른 성능향상(Speed Up), 페이지 부재 수, 수행 중 발생한 메시지 수로 비교하였다.

Water Nsquared 의 경우 DHLRC 모델은 LRC, HLRC 모델에 대해 각각 8.3%, 3.8%의 수행시간 감소와, 17.9%, 6.9%의 페이지 부재 수 감소, 그리고 5.8%, 4.9%의 메시지 수 감소를 보이고 있다. 홈을 고정하고 수행하는 HLRC 모델에 비해 적은 페이지 부재 수에 의해 전체 메시지 수가 줄어들고 이것은 전체적인 성능향상으로 나타나고 있다. Barnes-hut 의 경우에도 LRC, HLRC 모델에 비해 15.3%, 10.8%정도의 성능향상을 보이고 있다. 이것은 Water 의 경우보다 각 노드에서 발생하는 페이지 부재 수가 비슷하므로 홈의 이점 및 부하를 더 잘 공유하고 있다고 할 수 있다. 또한 MP3D의 경우 LRC, HLRC 모델에 비해 약 11.6%, 14.8%의 수행시간 감소를 보이고 있다. DHRLC모델에서 얻을 수 있는 성능 향상은 홈인 프로세서의 페이지 부재시 비용이 들지 않는 이점을 최대한 이용하는 것이다. 예를 들어, [표 4.]의 HLRC모델에서 P2의 경우 페이지 부재 수는 가장 적지만 통신시간은 매우 길게 나타났다. 하지만 DHRLC의 경우는 페이지 부재 수와 통신시간을 모든 프로세서가 비슷한 수준으로 유지함으로써 성능향상을 얻을 수 있다.

[표 4.] 프로세서당 페이지 부재 수 및 통신비용
(Barnes-Hut, 프로세서 개수 = 8)

	LRC		HLRC		DHLRC	
	Page Fault	Comm. Time	Page Fault	Comm. Time	Page Fault	Comm. Time
P1	5217	9402262	5463	10946743	4792	11289480
P2	5933	14990506	2670	14789133	4820	11256335
P3	5117	12194666	5029	9928125	4549	11120163
P4	5853	9481182	5878	11602614	4874	12037958
P5	5545	9936925	5104	10857898	4590	11213743
P6	4896	7851830	5526	10841748	5086	11558198
P7	5283	9996653	4165	88833543	4797	11323881
P8	6491	1638890	5467	107446360	4648	11997260

5. 결론 및 향후 과제

본 논문에서는 HLRC 모델에서 홈을 동적으로 유지하는 동적 홈-기반 지연 연성 일관성 모델을 제안하였다. HLRC 모델에서 홈은 페이지 부재를 갖지 않는 장점이 있지만 이러한 상태를 유지하기 위해 공유메모리의 변경이 일어난 모든 프로세서에게 변경부분을 전송 받아야 하는 부하가 있다. 반면, 본 논문에서 제안한 동적 홈-기반 정책에서는 홈이 갖는 페이지 부재가 일어나지 않는 이점을 분산 공유메모리 시스템에 참여하는 모든 프로세서들이 공유하고, 홈이 고정되어 발생하는 불필요한 오버헤드를 어느 정도 보완하여 시스템의 성능을 향상시킨다. 또한 홈 이전을 위해 별도의 메시지가 필요하지 않는 점도 장점이 될 수 있다.

향후 과제로는 페이지 부재만을 갖고 홈을 적절히 지정하는 것에는 제한점이 있기 때문에 홈의 이전 기준을 페이지 부재 이외의 요소로 실험해 보아야 하며, 홈이 이전된 후 바로 다시 이전되지 않는 효율적인 알고리즘을 고안해야 할 것이다.

참고문헌

[1] J. B. Carter, J. K. Bennett, and W. Zwaenepoel, Implementation and Performance of Munin. Proceedings of the 13th ACM Symposium on Operating Systems Principles, Oct. 1991.

[2] Pete Keleher, Alan L. Cox, Sandhya Dwarkadas, and Willy Zwaenepoel, TreadMarks: Shared Memory Computing on Networks of Workstations. IEEE Computer, Vol 29, No. 2, pp. 18-28, 1996

[3] Kai Li and Paul Hudak, Memory Coherence in Shared Virtual Memory Systems. ACM Transection on Computer Systems, Vol 7, No.4, November 1989.

[4] Jack E. Veenstra and Robert J. Fowler, MINT Tutorial and User Manual. Technical Report 452. The University of Rochester Computer Science Department, September 1993.

[5] Steven Cameron Woo, Moriyoshi Ohara, Evan Torrie, Jaswinder Pal Singh, and Anoop Gupta, The SPLASH-2 Programs: Characterization and Methodological Considerations. Proceedings of the 22nd Annual International Symposium on Computer Architecture, pages 24-36, June 1995.

[6] Yuanyuan Zhou, Liviu Iftode and Kai Li, Performance Evaluation of Two Home-Based Lazy Release Consisteeny Protocols for Shared Virtual Memory Systems. Proceedings of the 2nd Symposium on Operating Systems Design and Implementation, October 1996.