멀티코어 환경에서의 확장성 향상을 위한 메모리 할당자

조영중, 김인혁, 엄영익
성균관대학교 정보통신대학
{jyj8843, kkojiband, yieom}@skku.edu

Enhanced Memory Allocator for Scalability Improvement On Multicore

Youngjoong Cho, Inhyuk Kim and Young Ik Eom
College of Information and Communication Engineering, Sungkyunkwan University

요 약

멀티프로세서에서 시스템의 병렬성을 향상시키기 위해서 멀티스레드 프로그램을 이용한다. 이러한 멀티스레드 프로그램은 스레드간 역할을 분담하여 작업을 진행하게 된다. 멀티스레드 프로그램에는 생산자-소비자 구조가 있다. 기존 메모리할당자들은 생산자-소비자 구조에 대한 연구가 진행되지 않고, 크리티컬 섹션이 긴 락을 사용하여 성능상에 문제가 있다. 우리는 이러한 문제점을 독특한 메모리 해제 방법을 통해 해결하였고, 실험을 통해 메모리 할당자의 속도가 향상되는 것을 검증하였다.

1. 서론

최근 데스크탑의 프로세서의 수가 증가하고 있다. 데스크탑 사용자들은 프로세서의 수가 증가함에 따라 데이터가 빨리 처리되기를 기대한다. 이러한 데이터를 처리하기 위하여 메모리 할당자를 빈번하게 사용하기 때문에, 메모리 할당자는 데스크탑 성능에 많은 영향을 미친다. 멀티스레드 프로그램에서는 스레드 마다 각자의 역할을 분담하여 시스템의 병렬성을 향상시킨다. 이러한 작업의 예로는 생산자-소비자 구조가 있다. 생산자 스레드는 메모리를 할당받은 후에, 소비자 스레드에게 전달한다. 소비자 스레드는 생산자로부터 전달받은 메모리를 사용후에, 메모리를 해제한다. 기존 데스크탑 메모리 할당자들이 많이 연구가 되어 왔고, 성능상의 발전이 있었다. 하지만 이러한 메모리 할당자들은 생산자-소비자 구조에 대한 많은 고려를 하지 않았고, 이러한 과정을 처리하는 과정에서 락을 사용한다. 락을 소유한 스레드가 크리티컬 섹션에서 공유자료에 대한 접근을 하고 있을 때, 다른 프로세서에서 동작하고 있는 스레드들은 오랜 시간 동안 대기하고 있어야 한다. 이렇게 스레드들이 기다리고 있는 동안, 프로세서는 작업을 하지 못하므로 시스템의 성능이 좋지 않게 된다. 일반적으로 프로세서의 수가 증가하면, 그에 비례하여 시스템의 성능도 선형적으로 증가될 것이라고 생각한다. 하지만 이러한 생산자-소비자 구조에서의 락 사용은 프로세서 수에 비례하여 성능을 향상시키지 못하게 한다. 그렇기 때문에 멀티프로세서 환경의 생산자-소비자 구조에서도 시스템의 성능을 향상시킬 수 있는 메모리 할당자가 필요하다. 우리는 이러한 문제를 해결하기 위해 독특한 해제 기법을 가진 메모리 할당자를 개발했다.

본 논문의 구성은 2장에서 관련연구, 3장에서 디자인을 설명하고, 4장에서 실험결과에 대한 분석을 한다. 마지막은 결론으로 끝을 맺는다.

"본 연구는 지식경제부 및 정보통신산업진흥원의 대학 IT 연구센터 육성지원사업의 연구결과로 수행되었음" (NIPA-2013-(H0301-13-3001))

2. 관련연구

2.1. TCMalloc

생산자-소비자 구조에서 메모리 해제를 위해 메모리가 속한 스레드 캐쉬에 락을 사용해야 한다. 락으로 보호되는 스레드 캐쉬는 메모리 해제 요청을 처리하는 동안 메모리 할당을 처리할 수 없기 때문에, 메모리 할당자의 동시적인 실행을 저하시킨다[1].

2.2. Hoard 메모리 할당자

Hoard 메모리 할당자는 메모리 할당과 해제를 위해 락을 사용한다. 특히, 메모리 해제 과정은 2번의 중첩된 락을 통해 처리된다. 그렇기 때문에 Hoard 메모리 할당자는 생산자-소비자 구조뿐만 아니라 일반적인 경우에서도 성능이 좋지 않다[2].

3. 디자인

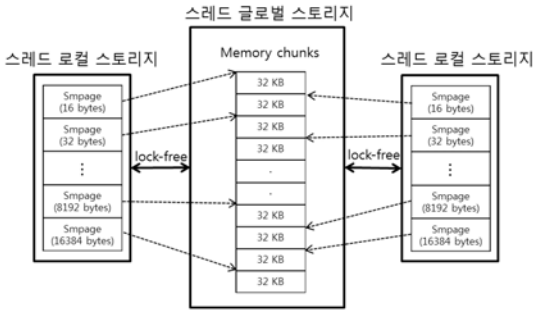

그림 1. 메모리 할당자 구조

우리의 메모리 할당자는 스레드마다 메모리 요청을 처리할 수 있는 스레드 로컬 스토리지를 소유한다. 스레드 로컬 스토리지에서 사용하는 메모리는 모든 스레드가 접근 가능한 스레드 글로벌 스토리지로부터 얻는다[3]. 그리고 스레드 로컬 스토리지는 메모리를 32KB 단위로 관리하며, 각 32KB 메모리에 대한 정보는 Smpage라는 자료구조를 이용하여 관리된다. Smpage에는 32KB 메모리의 주소, 저장될 메모리 객체들의 크기와 메모리 해제시에 사용되는 카운터가 저장된다.

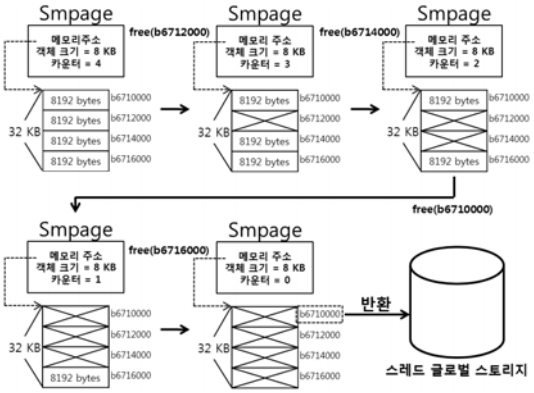

그림 2. 메모리 해제 방법

우리가 제안하는 메모리 해제 방법은 32 KB 메모리 마다 메모리 할당시 설정된 카운터를 이용하여 동작한다. 32 KB 메모리 안에 있는 메모리 객체가 해제될 때마다 카운터를 하나씩 감소시킨다. 카운터를 감소시킬때는 락을 사용하지 않고, 락 보다 크리티컬 섹션이 짧은 어토믹 인스터럭션을 사용한다[4]. 감소연산 결과로 인해 카운터가 0이 된다면, 해당 32 KB 메모리를 어떠한 스레드에서도 사용하지 않는다는 의미이다. 그렇기 때문에 해당 32 KB 메모리를 다른 스레드에서 사용가능하도록 스레드 글로벌 스토리지에 반환해준다. 이러한 구조는 생산자-소비자 구조에서도 카운터값만 변경시켜주기 위한 짧은 크리티컬 섹션을 사용하므로 성능상의 이점이 있다.

4. 실험

4.1. 실험 시나리오

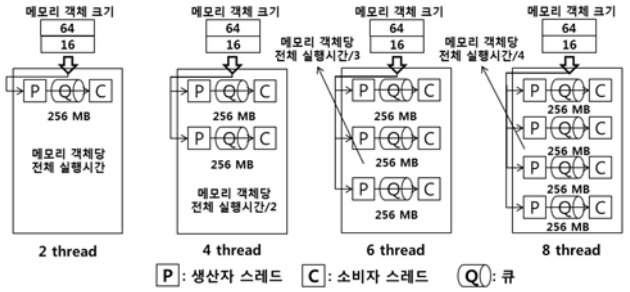

그림 3. 실험 구성

우리는 생산자-소비자 패턴에서 메모리 할당자의 속도를 측정하기 위한 실험을 그림3과 같이 구성하였다. 생산자 스레드는 메모리를 할당받아 큐에 삽입을 해준다[5]. 소비자 스레드는 큐로부터 메모리를 전달받아 사용후 해제한다. 메모리 객체는 16 Bytes와 64 Bytes를 사용하였으며, 각 메모리 객체당 256 MB를 할당을 진행하였다. 시간은 전체 실행시간을 생산자-소비자 개수로 나눠준 평균시간으로 측정하였다.

4.2. 실험 결과

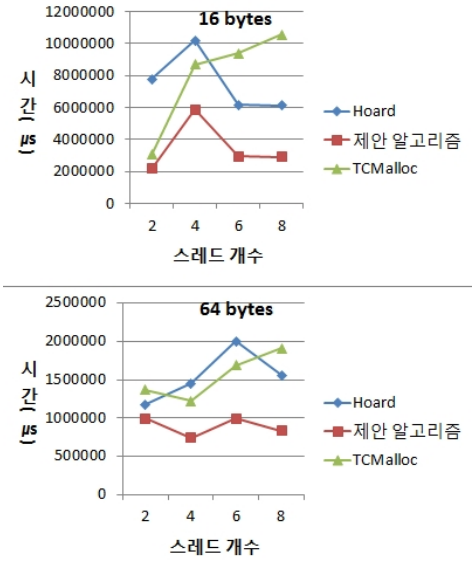

그림 4. 생산자-소비자 구조 실험

우리의 알고리즘이 메모리 객체마다 생산자 소비자 구조에서 좋은 결과를 얻는 것을 확인할 수 있었다.

5. 결론

32 KB 메모리마다 카운터를 두어 스레드 글로벌 영역에 대한 접근을 최소화 하는 우리의 알고리즘이 생산자-소비자 모델에서 기존 메모리 할당자보다 좋은 성능을 내는 것을 실험을 통하여 확인할 수 있었다.

참고문헌

[1] Google, “Tcmalloc: Thread-caching malloc,” http://google-perftools.googlecode.com/svn/trunk/doc/tcmalloc.html.

[2] E. D. Berger, K. S. McKinley, R. D. Blumofe, and P. R. Wilson, “Hoard: a scalable memory allocator for multithreaded applications. In Proc. of the 9th Int. Conf. on Architectural Support for Programming Languages and Operating Systems, 2000, pp. 117-128.

[3] P. P. Wilson, M. S. Jonstone, M. Neely, and D. Boles, “Dynamic stroage allocation: A critical review,” In Proc. Ofthe Int. Workshop on Memory Management, Vol. 986, Springer Verlag, 1995.

[4] D. Dice and A. Garthwaite, “Mostly lock-free malloc,” In Proc. of the 3rd Int. Symp. on Memory management, New York, 2002, pp. 163-174.

[5] P. Tsigas and Y. Zhang, “A simple, fast and scalable nonblocking concurrent fifo queue for shared memory multiprocessor systems,” In Proc. of the 13th ann. ACM symp. On Parallel algorithms and architecutres, New York, 2001,pp. 134-143.