매니코어 시스템을 위한 리눅스 가상 메모리 관리의 락 경합 분석

An Analysis of Lock Contention in Linux Memory Management for Many-core System

저자
(Authors)

경주현, 임성수
Joohyun Kyong, Sung-Soo Lim

출처
(Source)

한국정보과학회 학술발표논문집 , 2015.06, 1571-1573 (3 pages)

발행처
(Publisher)

한국정보과학회
KOREA INFORMATION SCIENCE SOCIETY

URL

http://www.dbpia.co.kr/Article/NODE06394479

APA Style

경주현, 임성수 (2015). 매니코어 시스템을 위한 리눅스 가상 메모리 관리의 락 경합 분석. 한국정보과학회 학술발표논문집, 1571-1573.

매니코어 시스템을 위한 리눅스 가상 메모리 관리의 락 경합 분석

경주현 ○ 임성수

국민대학교 컴퓨터공학부

Joohyun0115@gmail.com, sslim@kookmin.ac.kr

An Analysis of Lock Contention in Linux Memory Management for Many-core System

Joohyun Kyong○ Sung-Soo Lim

School of Computer Science Kookmin University

요 약

본 논문은 매니코어 환경에서 리눅스 가상 메모리 관리의 확장성에 대한 문제점을 제시 하였고, 이를 해결하기 위해 메모리 병목지점에 대한 락 경합에 대해서 분석하였다. 이를 위해 120코어를 가진 매니코어 시스템에서 리눅스 커널을 대상으로 실험 및 분석하였다. 분석 방법은 멀티 프로세스 기반의 벤치마크와 멀티 쓰레드 기반의 벤치마크를 락 프로파일링 하여 문제점을 도출 하였다. 본 논문의 결과는 향후 매니코어 환경에서 리눅스 메모리 관련 확장성을 향상시키기 위해 유용하게 사용될 수 있다.

1. 서 론

최근 공정이 미세해짐에 따라 단일 CPU에 점점 많은 코어를 넣고 있다. 이와 동시에 상대적으로 낮은 가격을 가지는 서버의 코어 수가 증가되고 있다. 따라서 서버에서 사용되는 CPU들이 멀티코어 시스템에서 매니코어 환경으로 변화되고 있다. 이와 동시에 특정 목적을 위해 만들어진 슈퍼컴퓨터와 달리, 서버에서 많이 사용되는 범용 운영체제 중 하나인 리눅스의 확장성에 대한 연구 필요성이 증가되고 있다. 또한 코어 수가 증과 됨과 동시에 메모리의 크기 역시 상대적으로 증가되고 있다[1]. 이처럼 증가되는 메모리 사이즈에 대해서 효율적인 관리가 반드시 필요한 상황이다.

하지만 매니코어 환경에서 리눅스 운영체제에 대해서는 확장성에 대해 개선해야 할 문제점이 존재한다[1][2]. 특히 리눅스 커널의 메모리 관리에 대해서 해결해야 할 이슈들이 있다. 이러한 문제를 해결하기 위해서는 매니코어 시스템 위에서 동작하는 리눅스 커널의 메모리 관리에 대한 확장성에 대한 정확한 분석이 필요한 상황이다. 특히 확장성에 가장 영향을 미치는 락에 대한 분석이 필요하다. 하지만 아직 리눅스 커널을 대상으로 100코어 이상의 매니코어 시스템에서 락에 의해 확장성을 저해 시키는 부분에 대한 분석이 아직 부족한 것이 문제이다.

* 이 논문은 2015년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No.B0101-15-0644, 매니코어 기반 초고성능 스케이러블 OS 기초연구)

본 논문은 이러한 문제점을 해결하기 위하여, 120코어 시스템을 대상으로 리눅스 커널의 메모리 관리에 대한 확장성을 테스트 해보았고, 그 중 락에 대한 문제점 분석하였다. 본 논문은 메모리 관리에 대한 문제점을 분석하기 위해 다중 멀티 프로세스를 사용하는 벤치마크와 다중 쓰레드로 동작하는 벤치마크를 사용하여, 두 벤치마크를 대상으로 어느 부분에 대해 락 경합이 많이 발생하는지를 분석하였다. 이를 통해 향후 개선해야 할 방향을 제시하였다.

본 논문의 구조는 2장에서는 연구 동향에 대해서 설명하며, 3장에서는 벤치마크를 사용하여 매니코어 환경에서 리눅스의 확장성 문제를 설명한다. 4장에서는 앞장에서 제시한 문제점을 락 프로파일링 하여 프로파일링 결과와 문제점에 대해서 설명하며. 마지막으로 5장에서는 결론 및 향후 연구에 대해서 설명한다.

2. 연구 동향

매니코어를 위한 락 기법에 대한 연구는 확장성 있는 락에 대한 연구[3]와 이를 매니코어 기반의 리눅스 커널에 적용을 하는 연구[4]가 있다. 예를 들어 확장성이 뛰어난 큐 기반의 락인 MCS를 리눅스 커널에 적용하여 확장성을 향상시키는 방법[4]이 있다. 다음으로 읽기 쓰기 락을 지연 처리가 가능하도록 하여, 뛰어난 확장성을 제공하는 RCU에 대한 연구[5]와 이를 활용한 알고리즘에 대한 연구[6][7]들이 개발되었다. 예

를 들어 리눅스의 단일 주소 공간 때문에 발생하는 쓰레드 간의 mmap과 페이지 폴트의 락 경합을 해결한 Bonsai[6]와 Bonsai의 성능 문제점을 해결하여 리눅스 디폴트 메모리 관리에 사용하는 레드블랙 트리를 RCU로 개선한 연구[7]가 있다.

3. 리눅스 메모리 관리에 대한 확장성 분석

리눅스의 락에 대해 분석하기 이전에 리눅스 메모리 관리에 대해 확장성을 분석하였다. 이를 위해 크게 두 가지의 종류의 벤치마크를 이용하였다. 하나는 멀티 프로세스 기반으로 수행하는 방법과 다른 하나는 멀티 쓰레드 기반으로 병렬화를 수행하는 방법에 대해서 확장성을 분석하였다. 분석을 위한 환경은 표 1과 같다.

표 1 실험 환경

CPU	Intel Xeon E7-4800/8800 v2
코어 수	120코어
메모리	755GB
NUMA 노드 수	8 소켓(소켓당 15코어)
운영체제	리눅스 커널 3.19.0-rc4

3.1 멀티 프로세스 기반 벤치마크 확장성 분석

멀티 프로세스 기반 벤치마크에 대해 확장성을 분석하기 위해, 매니코어 연구[8]에서 사용한 AIM7 벤치마크를 사용하였다. 모든 실험은 물리적인 코어만 실험하기 위해 인텔 하이퍼스레딩 기능을 끄고 측정하였으며, 리눅스 커널의 주파수를 최대로 설정하여 실험을 수행하였다. 파일 시스템은 메모리 병목지점만 측정하기 위해, 메모리 파일 시스템인 tmpfs를 사용하여 측정 하였다. 측정 결과는 그림 1과 같이 앞서 매니코어 연구에서 제기한 문제와 똑 같은 상황이 재현되었다.

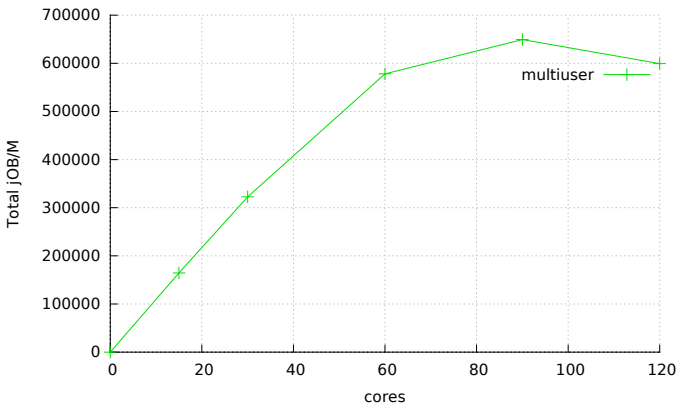

그림 1 AIM7 벤치마크(tmpfs + multiuser)

60코어 까지는 성능에 대한 확장성이 있으나 60코어 이후에는 성능 증가 폭이 줄어들었으며, 120코어 에서는 오히려 성능이 감소하는 문제점이 있다.

3.2. 멀티 쓰레드 기반 벤치마크 확장성 실험

멀티 쓰레드 기반을 측정하기 위한 본 논문에서는 SPECJbb2013[9] 벤치마크를 사용하였다. 단일 프로세스 위에서 동작시키기 위해 싱글 JVM위에 동작하도록 설정하였다. JVM 메모리 사이즈는 SPECJbb2013의 디폴트인 24GB로 설정하여 실험하였다.

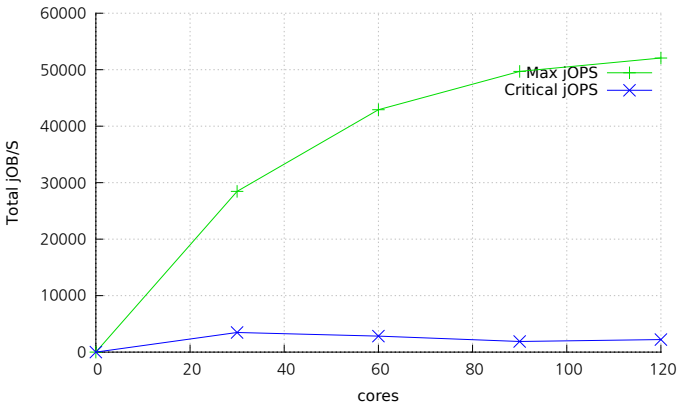

그림 2 SPECjbb2013 벤치마크

그림 2의 그래프 결과 MAX jOPS는 AIM7과 비슷하게 코어 수가 증감함에 따라 확장성이 떨어짐을 볼 수 있고, Critical jOPS는 30코어 이후에는 오히려 감소하는 것을 볼 수 있다.

이처럼 멀티 프로세스 기반의 벤치마크와 멀티 쓰레드 기반의 벤치마크 둘 다 확장성에 대해서 문제를 가진다. 본 논문에서는 확정성 병목 지점을 분석하기 위해 락 경합을 분석하였다. 분석 내용은 4장에서 설명한다.

4. 락 경합 분석

본 논문은 매니코어 환경에서 2가지 유형에 대해서 리눅스 커널의 메모리 관리에 대한 락 경합을 분석하였다. 모든 분석은 리눅스의 lockstat[10]를 사용하여 벤치마크 수행 시 발생하는 락 경합을 분석하였다.

4.1 멀티 프로세스 기반 벤치마크 락 경합 분석

먼저 멀티 프로세스 기반의 벤치마크인 AIM7을 동작시키고 동시에 120코어 대해서 락 경합을 분석하면 그림 3과 같은 결과를 가진다.

AIM7 벤치마크의 경우 상당히 많은 부분이 anon_vma에서 쓰기 락 경합이 발생한다. 이는 리눅스 역 매핑(reverse mapping)을 효율적으로 수행하기 위한 자료구인 anon_vma를 수많은 fork에 의해 프로세스를 생성하면서 발생하는 락 경합 문제이다. 리눅스 역 매핑 문제와 관련된 자세한 커널의 내부 구조는 지면상 생략한다. 다음으로 anon_vma의 읽기 락이 발생하였고, 다음으로 파일 시스템 접근 때문에 발생하는 락인 f_pos_lock이 많이 발생하였다.

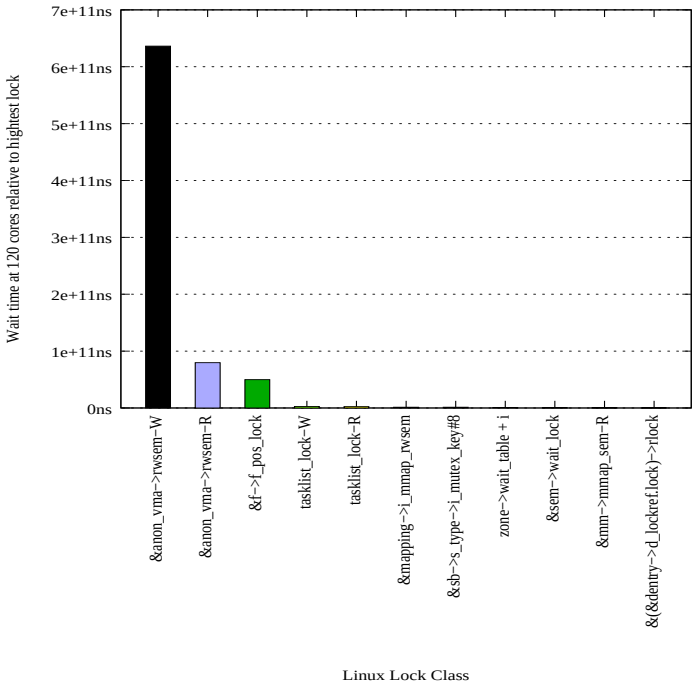

그림 3 AIM7 락 프로파일링(120코어)

4.2 멀티 쓰레드 기반 벤치마크 락 경합 분석

멀티 쓰레드 기반의 벤치마크에 대해서 락 경합을 분석하면 그림 4와 같다. 상당히 많은 부분이 리눅스 메모리 관리 구조체인 mm_struct의 내부 변수인 mmap 세마포어와 페이지 테이블 락에 의해서 발생하였다. mmap 세마포어에 대한 문제점은 Bonsai[6]에 의해 발견된 이슈인 페이지 폴트와 mmap 간의 락 경합으로 발생한 현상이다. 추가로 코어 수의 증가로 과도한 로드밸런싱에 의한 런큐 락 문제가 발견되었다.

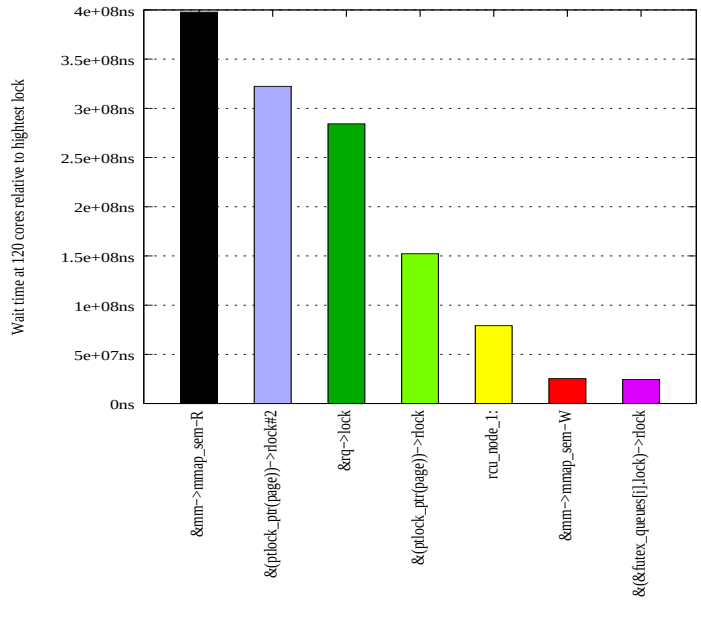

그림 4 SPECjbb2013 락 프로파일링(120코어)

5. 결론 및 향후 연구

본 논문은 120코어 시스템을 대상으로 매니코어를 위한 리눅스의 확장성에 대한 문제점을 제시하였다. 또한 다중 프로세스와 다중 쓰레드를 대상으로 락에 대한 문제점을 분석하였다. 향후 연구로 발견된 리눅스 역 맵핑에 의해 발생하는 확정성에 대한 문제점을 해결할 예정이다.

6. 참고문헌

[1] Kleen, Andi. "Linux multi-core scalability." Proceedings of Linux Kongress. 2009.

[2] Boyd-Wickizer, Silas, et al. "An Analysis of Linux Scalability to Many Cores." OSDI. Vol. 10. No. 13. 2010.

[3] Mellor-Crummey, John M., and Michael L. Scott. "Synchronization without contention." ACM SIGARCH Computer Architecture News. Vol. 19. No. 2. ACM, 1991.

[4] Boyd-Wickizer, Silas, et al. "Non-scalable locks are dangerous." Proceedings of the Linux Symposium. 2012.

[5] McKenney, Paul E. "Is parallel programming hard, and, if so, what can you do about it?." Linux Technology Center, IBM Beaverton (2011).

[6] Clements, Austin T., M. Frans Kaashoek, and Nickolai Zeldovich. "Scalable address spaces using RCU balanced trees." ACM SIGARCH Computer Architecture News. Vol. 40. No. 1. ACM, 2012.

[7] Arbel, Maya, and Hagit Attiya. "Concurrent updates with rcu: Search tree as an example." Proceedings of the 2014 ACM symposium on Principles of distributed computing. ACM, 2014.

[8] 정진환, 김강호, 김진미, 정성인. “Manycore 운영체제 동향.” 전자통신 동향 분석 29 권 5 호, 2014.

[9] SPECjbb2013, https://www.spec.org/jbb2013

[10] Pepper, Tim. "Linux kernel lock profiling with lockstat." (2007).