

다양한 환경에서의 매니코어 CPU 시스템의 병렬 쓰기 성능 향상을 위한 리눅스 LRU 관리 최적화 기법의 성능 분석

변은규¹, 구기범¹, 오광진¹, 정기문¹, 방지우²

¹한국과학기술정보연구원

²삼성전자

ekbyun@kisti.re.kr, gibeon.gu@kisti.re.kr, koh@kisti.re.kr, kmjeong@kisti.re.kr,
cilioh14@gmail.com

Performance Analysis of Finer-LRU for Improving Parallel Write Performance in Diverse Many-Core CPU Systems

Eun-kyu Byun¹, Gibeom Gu¹, Kwang-Jin Oh¹, Kimoon Jeong¹, Jiwoo Bang²

¹Korea Institute of Science and Technology Information

²Samsung Electronics

요 약

매니코어 시스템에서 병렬 프로세스가 동시에 I/O를 수행하는 경우 병목 현상으로 인한 성능저하가 발생한다. 이는 하드웨어적인 문제뿐만이 아니라 리눅스에서 사용하는 LRU 알고리즘이 단일 Lock을 사용하는 것임이 확인되었다. 이를 해결하기 위해 Finer-LRU 라는 알고리즘을 이용하여 성능 완화를 시도하였다. 본 연구에서는 이 알고리즘이 ARM, Epyc CPU와 같이 비 Intel 제조 CPU에서도 유효한지를 확인하기 위한 실험을 진행하였고 그 결과를 제공한다. ARM 환경에서는 인텔과 유사한 성능 향상을 보였으나, 동일한 x86 아키텍처의 Epyc CPU에서는 큰 성능 변화가 없음을 확인하였다.

1. 서 론

HPC, 빅데이터, AI 등 대규모 데이터 처리 수요가 늘면서 수십 개의 코어를 가진 매니코어 CPU가 일반화되었다. 이런 환경에서 각 코어에서 동작하는 프로세스들이 동시에 I/O를 수행하는 경우 병목 현상이 발생한다.[1] 본 연구팀은 이전의 연구를 통해 이러한 병목 현상이 하드웨어적인 한계일 뿐만 아니라, 리눅스 커널에서 사용하는 LRU 구조에도 문제가 있음을 발견하였다. 단일 LRU Lock에 대한 과도한 경쟁이 문제임을 발견하고 이를 해결하는 방법으로 Finer-LRU를 제안하고, 개선하여 리눅스 커널 5.18에 구현하고 이를 검증하였다. [2-4]

이러한 검증은 Intel에서 출시한 2종류의 CPU, 즉 Knights Landing[8]과 Xeon에 대해 수행되었었다. 본 연구팀은 추가로 2종의 매니코어 시스템을 확보하여 Finer-LRU가 다른 매니코어 시스템에서도 유효한지를 검증하였고 그 결과를 본 논문에서 제공한다. 이러한 총 4대의 다른 CPU를 가진 매니코어 시스템에서의 성능 측정하였고 그 결과를 비교하였다.

2. 본 론

Finer-LRU를 개발 동기는 매니코어 CPU에서 IOR 벤치마크를 이용하여 MPI 프로세서 여럿이 동시에 쓰기 작업을 수행할 때, 성능저하가 발견되었기 때문이다. 이는 리눅스의 버퍼 캐시를 관리하는 LRU의 Lock 메커니즘이 단 하나의 lock 변수에 의해 처리되기 때문인데 이를 해결하기 위해 Finer-LRU는 FINER_FACTOR 수만큼 lock 변수와 list의 개수를 늘림으로써 리스트 병목을 완화한다. 다음으로 단순히 FINER_FACTOR를 늘린다고 성능이 향상되지 않는다는 점을 확인하였다. 페이지가 LRU 리스트 사이에서 이동하는 경우, 동시에 접근하는 LRU 리스트의 패턴이 존재하는 것을 확인하고 이렇게 동시에 갱신되는 LRU 리스트들은 동일한 lock에서 관리되도록 하여, lock 획득으로 인해 발생하는 성능저하를 완화하였다. 추가로, 이어진 작업에서 동일한 lock을 풀었다가 다시 잡는 상황에서는 lock 관리 최적화 기법을 적용하였다.

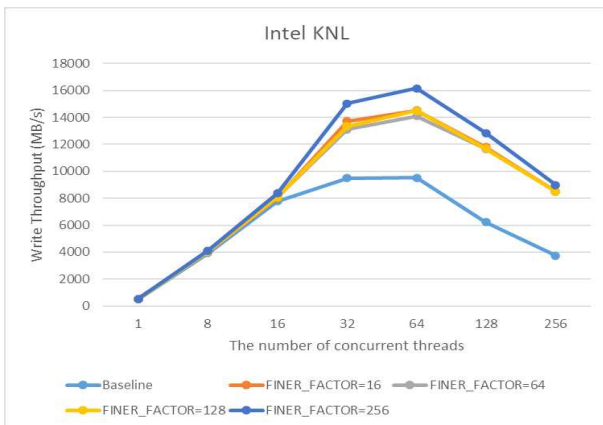
Finer-LRU는 리눅스 5.18 커널에서 구현하였고, 이를 4종류의 매니코어 CPU 환경에서 쓰기 성능을

분석하였다. 성능의 측정에는 IOR[9]과 마이크로 벤치마크를 사용하였으며 MPI 프로세스가 병렬로 쓰기를 수행한다. 아래 표 1은 실험에서 사용한 CPU 목록을 나타낸다. 지난 논문에서 제공했던 Intel사의 2종류의 CPU에 추가로 ARM 아키텍처 기반의 Ampere Altra CPU를 탑재한 서버와 x86 아키텍처 기반의 AMD EPYC2 CPU를 탑재한 서버 2종을 포함하여 총 4종의 결과를 제시한다.

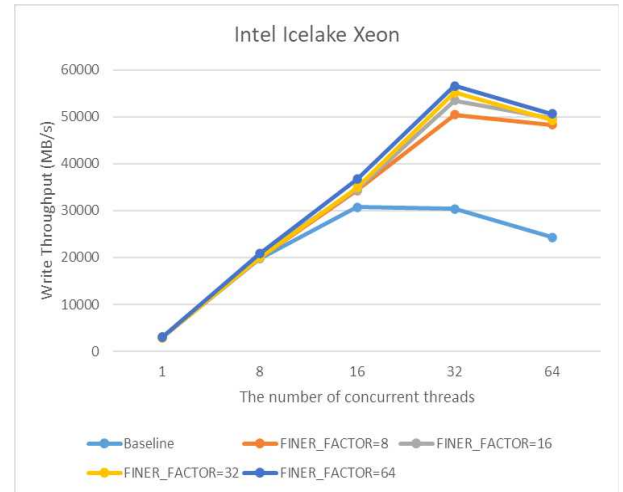
<표 1> Finer-LRU 분석에 사용한 매니코어 CPU 목록

이름	아키텍처	코어/스레드 수
Intel Knights Landing(KNL)	x86/64	64/256
Intel Icelake Xeon	x86/64	32/64
AMD EPYC2	x86/64	64/128
Ampere Altra	ARM	80/80

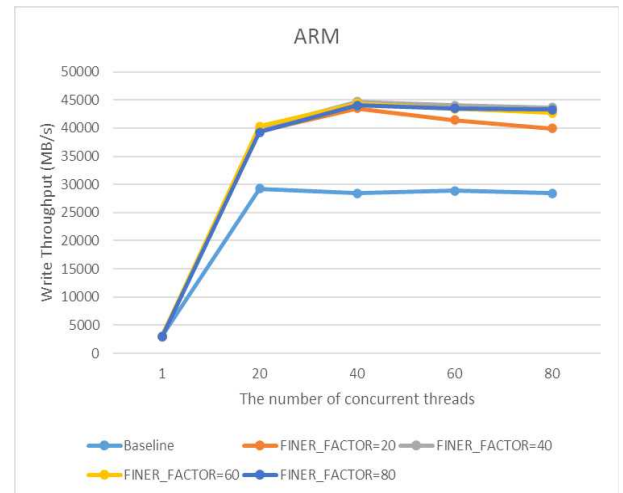
아래 그림1~4는 각 4종의 서버에서 측정한 병렬 순차 쓰기 성능의 총합을 보여준다. 병렬 쓰기에 참여하는 스레드의 개수를 늘려가고, FINER_FACTOR를 바꾸어가며 측정해 보았다. 각 그래프 중 Baseline은 Finer-LRU 미적용 상태를 의미한다. 결과를 보면 KNL, Xeon, ARM CPU의 경우 적용하지 않을 때와 비교하여 성능이 유의미하게 좋아지는 것을 확인할 수 있었다. 다만 인텔 제작 CPU의 경우, 스레드 수가 한계까지 늘어나는 경우 성능 다시 감소한다. ARM CPU도 스레드 수 증가에 따라 성능 향상이 멈추었으나 많은 스레드에서 성능이 줄어드는 현상은 없었다. 반면 EPYC2 CPU의 경우에는 Finer-LRU 적용 여부와 FINER_FACTOR 차이에 따라 성능 차이가 별로 없는 것이 확인되었다. 다만 도달 가능 쓰기 성능 합은 32스레드 정도에서 한계에 도달함이 확인되었다.



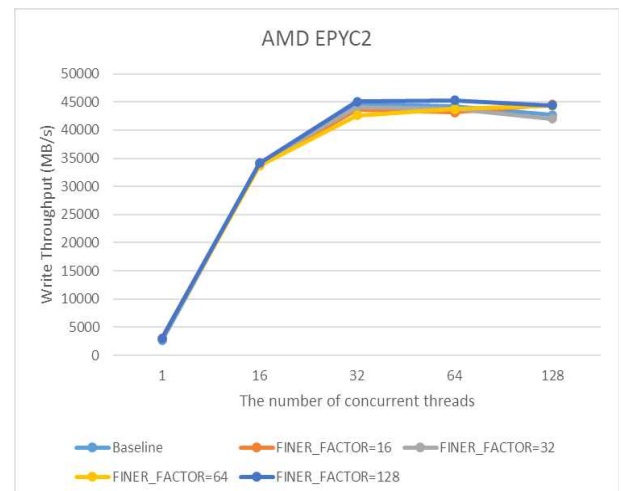
(그림 1) Intel KNL CPU에서 Finer-LRU 적용 후 병렬 순차 쓰기 성능



(그림 2) Intel Xeon CPU에서 Finer-LRU 적용 후 병렬 순차 쓰기 성능



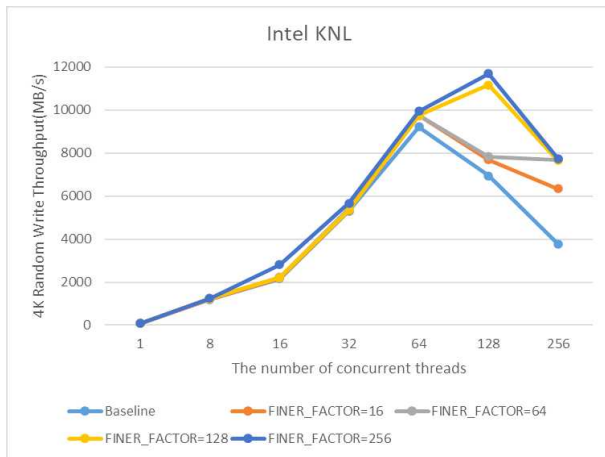
(그림 3) ARM CPU에서 Finer-LRU 적용 후 병렬 순차 쓰기 성능



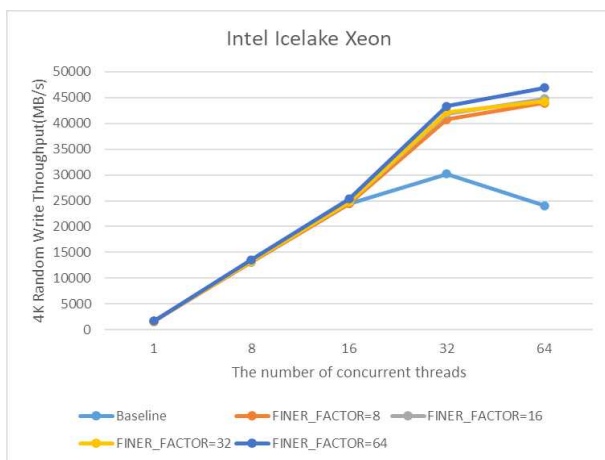
(그림 4) AMD EPYC2 CPU에서 Finer-LRU 적용 후 병렬 순차 쓰기 성능

아래 그림5~8은 각 4종의 서버에서 측정한 병렬 4 Kbyet 무작위 쓰기 성능의 종합 결과를 보여준다. 결과를 보면 순차 쓰기와 유사한 패턴을 확인할 수 있었다. Intel과 ARM CPU 계열에서는 FINER-LRU 적용으로 많은 스레드가 동시에 사용될 때의 쓰기 성능이 향상됨을 확인하였다. 반면 EPYC CPU에서는 FINER-LRU의 효과가 없었다. 또한 Intel 계열 CPU에서는 스레드 개수가 최대치일 때 성능이 오히려 떨어지는 현상이 발견되었다.

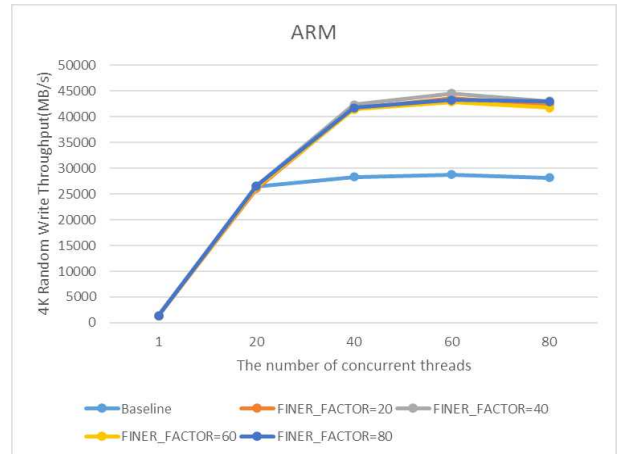
그림으로 제공하지는 않았지만 읽기 성능도 측정해 보았다. 읽기의 경우 LRU 리스트 변경이 없는 경우가 대부분이기 때문에 예상대로 FINER-LRU 적용으로 성능 향상은 없었다. 그러나 성능저하 또한 발견되지 않았는데, 이는 lock이 많아져서 발생하는 오버헤드가 유의미하게 크지 않았음을 의미한다.



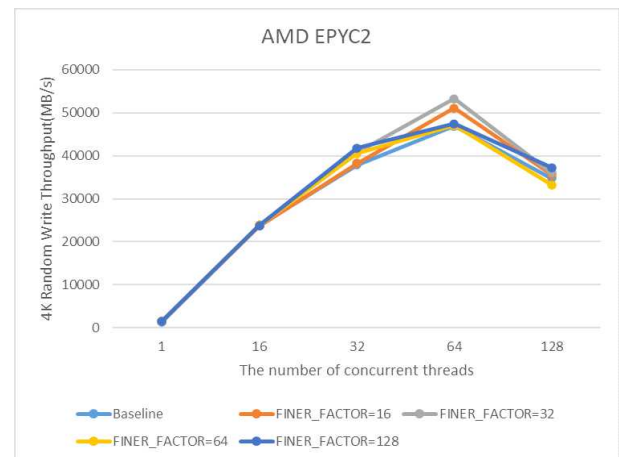
(그림 5) Intel KNL CPU에서 Finer-LRU 적용 후 병렬 4KB 랜덤 쓰기 성능



(그림 6) Intel Xeon CPU에서 Finer-LRU 적용 후 병렬 4KB 랜덤 쓰기 성능

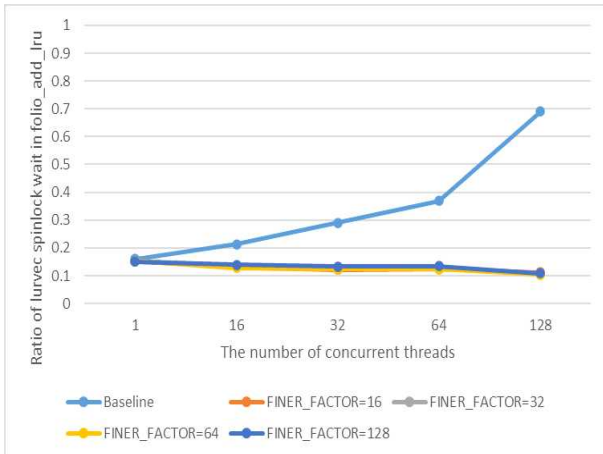


(그림 7) ARM CPU에서 Finer-LRU 적용 후 병렬 4KB 랜덤 쓰기 성능



(그림 8) AMD EPYC2 CPU에서 Finer-LRU 적용 후 병렬 4KB 랜덤 쓰기 성능

EPYC2 CPU가 다른 CPU와 다른 성능 특성을 가지는 것은 ISA같은 문제는 아닌 것으로 보인다. 인텔과 동일한 x86 기반이기 때문이다. 실제로 리눅스 커널 내에서의 LRU 처리에 lock을 잡기 위해 대기하는 시간이 EPYC2 CPU에서 어떻게 변하였는지를 확인한 결과가 그림 9에 주어져 있다. 그래프를 보면 Finer-LRU 미적용시(baseline) 대비 spin lock 소요시간은 크게 줄어들었음을 알 수 있다. lock 대기 시간이 줄어들어도 성능에 영향이 없었다는 것은, EPYC2 환경에서 병렬 메모리 성능에 커널의 영향이 크지 않았다는 것을 의미한다. 가장 의심되는 것은 메모리 컨트롤러의 성능과 대역폭의 차이와 같은 하드웨어적인 특성의 차이일 것으로 예상된다. 이를 완벽히 분석하기 위해서는 하드웨어에 대한 더 깊은 이해가 필요할 것으로 보이고, 이는 향후 연구 목표로 삼을 예정이다.



(그림 9) AMD EPYC2 CPU에서 Finer-LRU 적용
 전후 커널 내 LRU 갱신 시 spinlock 대기 시간

3. 결 론

본 연구팀은 이전의 연구를 통해 매니코어 CPU에서 병렬 쓰기를 수행할 때 발생하는 성능 병목의 원인이 리눅스 커널의 LRU 메커니즘에 있음을 확인하고 이를 해결하기 위한, Finer-LRU를 제안하였다. 본 논문에서는 추가적인 환경에서의 평가를 추가로 실시하였고, 일부 하드웨어에서 기존과 다른 성능 패턴이 나올 수 있음을 확인하였다. 향후 이러한 하드웨어 특성을 고려하여 알고리즘을 설계해야 함을 의미한다. 또한 고병렬 시스템 소프트웨어 설계 시 하드웨어 개발과의 협업이 필수적이라는 교훈을 얻을 수 있었다.

이 논문은 2025년도 한국과학기술정보연구원 (KISTI)의 기본사업의 지원을 받아 수행된 연구입니다. (과제번호: K25L1M2C2)

참고문헌

- [1] S. Mittal, "A survey of techniques for architecting and managing asymmetric multicore processors," ACM Comput. Surv., vol. 48, no. 3, Feb. 2016.
- [2] J. Bang et al. "Finer-LRU: A Scalable Page Management Scheme for HPC Manycore Architectures," 2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 567-576, 2021.
- [3] E. -K. Byun et al, "A Study on Optimizing LRU lock for Improving Parallel I/O Throughput

in Manycore CPU Systems," Proceedings of the Annual Conference of Korea Information Processing Society Conference (KIPS) 2022, Vol.29, No.2, pp.2-4, 2022.

[4] Byun, E.-K., Gu, G., Oh, K.-J., & Bang, J. (2023). Optimizing LRU Lock Management in the Linux Kernel for Improving Parallel Write Throughout in Many-Core CPU Systems. KIPS Transactions on Computer and Communication Systems, 12(7), 209 - 216.

[5] D. Zheng, R. Burns, and A. S. Szalay, "A parallel page cache: Iops and caching for multicore systems," in Proceedings of the 4th USENIX Conference on Hot Topics in Storage and File Systems, ser. HotStorage'12. USA: USENIX Association, 2012, p. 5.

[6] Y. Zhang et al. "FineLock: automatically refactoring coarse-grained locks into fine-grained locks," In Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis, pp. 565 - 568, 2020

[7] K. Ganesh, S. Kalikar, and R. Nasre, "Multi-granularity Locking in Hierarchies with Synergistic Hierarchical and Fine-Grained Locks," in Euro-Par 2018: Parallel Processing, pp.546 - 559, 2018

[8] A. Sodani et al., "Knights Landing: Second-Generation Intel Xeon Phi Product," in IEEE Micro, vol. 36, no. 2, pp. 34-46, 2016.

[9] IOR Benchmark, <https://github.com/hpc/ior>