NUMA 인지 PM 파일시스템의 성능 확장성 평가 및 분석

변홍수, 김준형, 김영재, 박성용
서강대학교 컴퓨터공학과
{ byhongsu, junehyung, youkim, parksy}@sogang.ac.kr

Scalability Evaluation and Analysis for NUMA-aware PM File System

Hongsu Byun, June-Hyung Kim, Youngjae Kim, Sungyong Park
Department of Computer Science and Engineering, Sogang University, Seoul, Republic of Korea

요 약

PM(Persistent Memory) 파일시스템들의 I/O 성능은 NUMA(Non-Uniform Memory Access) 구조의 특징인 비대칭적인 메모리 접근 성능에 큰 영향을 받는다. 대부분의 기존 PM 파일시스템들은 NUMA를 인지하지 않고 설계되어 잦은 원격 메모리 접근이 발생하고 이는 파일시스템 확장성의 병목지점이 된다. 최근의 연구는 이러한 문제를 해결하기 위해 파일시스템의 중요 자료구조를 재설계하거나 NUMA 인지적인 메모리 할당 정책을 이식하였다. 하지만, 그들의 연구는 실제 PM 모듈이 장착된 2개의 NUMA 노드만을 가지는 서버 위에서 실험되었기 때문에 미래에 더 많은 NUMA 노드의 환경에서 발생하는 문제를 예측하지 못한다. 따라서, 본 연구에서 우리는 더 많은 NUMA 노드를 갖는 서버에서 NUMA 인지적인 PM 파일시스템의 성능 확장성 평가를 수행하며 결과를 분석한다. 이 NUMA 인지적인 PM 파일시스템을 8개의 NUMA 노드를 갖는 환경에서 실험한 결과, 동시적 파일 쓰기 워크로드에 대해서 I/O 처리량 측면에서 노드가 2개 일때보다 더 선형적인 확장성을 보였다. 반면 읽기 비중이 높은 읽기 쓰기 혼합 워크로드에서는 더 많은 노드의 NUMA 환경에서 더 잦은 원격 메모리 접근이 발생하여 I/O 처리량이 하락하였다.

1 서 론

영구적 메모리(PM, Persistent Memory)라는 차세대 메모리 기술이 등장함에 따라 이들을 저장매체로 활용하기 위한 파일시스템들이 등장하였다. 대표적으로 NOVA 파일시스템 [1]은 기존 블락 기반의 파일시스템들보다 훨씬 높은 I/O 처리량과 로그 구조를 통한 강한 파일 데이터 일관성을 제공한다.

최근 매니코어 서버에 적합한 PM 파일시스템 디자인을 위한 연구들이 존재한다 [2, 3, 4]. 이들은 파일시스템이 더 많은 I/O 요청들을 동시에 처리하여 PM의 높은 대역폭을 최대한 활용하는 것을 목표로 한다. 특히 [3]는 대부분의 매니코어 서버들이 NUMA(Non-Uniform Memory Access) 구조를 기반으로 하기 때문에 NUMA를 고려하지 않고 디자인된 PM 파일시스템들은 많은 확장성 병목지점을 가지고 있다고 주장하였다. 예를 들어 기존 NOVA는 모든 파일 데이터와 메타 데이터를 하나의 NUMA 노드에 배치하기 때문에 다른 노드에서 수행되는 I/O 쓰레드는 원격 메모리 접근을 통해 높은 지연 시간으로 파일에 접근할 수 밖에 없다. 더 많은 쓰레드가 동시에 파일에 접근할수록 원격 메모리 접근 횟수가 빈번해지기 때문에 큰 오버헤드가 발생한다. 따라서 그들은 쓰레드마다 파일 데이터와 메타데이터를 지역 노드에 장착된 PM에 배치할 수 있는 메모리 할당 기법을 제안하고, 기존 NOVA의 성능 확장성 문제를 개선하였다.

선행 연구는 개선된 NOVA를 검증하기 위하여 실제 PM인 Intel Optane DC PM를 장착한 서버에서 실험되었다. 하지만 이 첫 번째 PM 상품은 하드웨어적인 병목지점을 많이 가지고 있을뿐만 아니라 해당 서버는 두 개의 NUMA 노드만을 가지기 때문에 선행 연구의 실험들로 PM 파일시스템의 확장성을 분석하기에는 한계가 있다. 따라서 우리는 미래에 PM이 발전되어 높은 대역폭을 보이고, 더 많은 NUMA 노드에 장착될 수 있는 상황을 가정한다. 이를 위해 본 연구는 8개의 NUMA 노드를 갖는 매니코어 서버에서 DRAM을 기반으로 에뮬레이션 된 PM 위에서 개선된 NOVA를 실험 평가 및 분석한다. 대표적인 실험 결과로, 동시적 파일 쓰기 워크로드 실험에서는 NUMA 노드 수가 많을수록 개선된 NOVA의 I/O 처리량은 쓰레드 수에 따라 선형적으로 증가하였다. 반면 쓰기와 읽기가 혼합된 워크로드에서는 읽기의 비중이 높을수록 개선된 NOVA의 성능이 감소하였다. 이 이유는 많은 노드가 사용 될수록 읽기 쓰레드들이 원격 메모리에 배치된 파일을 읽게 될 확률이 증가하고 잦은 원격 메모리 접근으로 인한 성능 문제가 나타났기 때문이다.

"본 연구는 과학기술정보통신부 및 정보통신기획평가원의 SW중심대학지원사업의 연구결과로 수행되었음"(2015-0-00910).

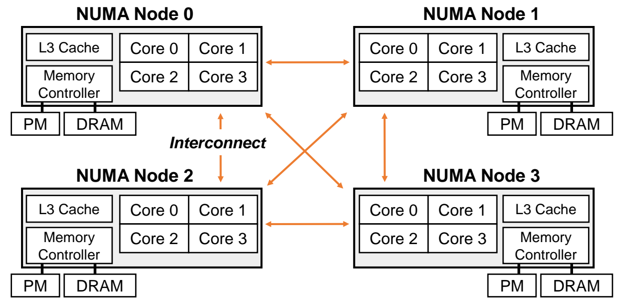
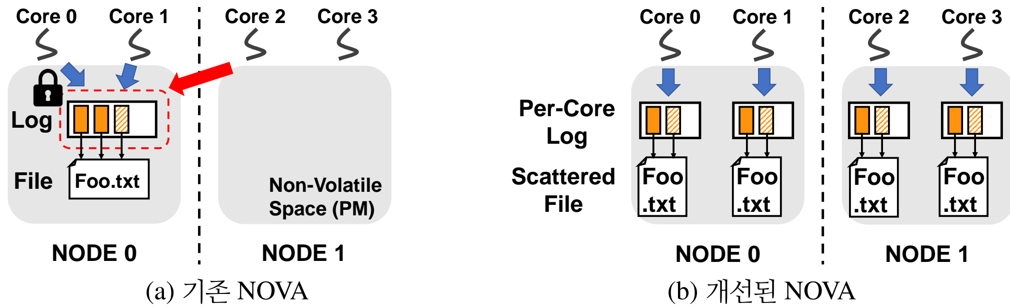


그림 1: NUMA 기반 매니코어 서버 아키텍처.

2 배경 지식 및 연구 동기

2.1 NUMA 시스템

오늘날 많은 매니코어 서버들은 NUMA 구조로 이루어져 있다. 그림 1은 4개의 노드를 갖는 NUMA 기반의 시스템 구조의 예시이다. 각 노드는 CPU 소켓과 메모리 컨트롤러를 포함한다. 예시에서 한 CPU 소켓은 4개의 코어를 갖고 코어들은 CPU의 L3 캐시를 공유한다. 노드의 메모리 컨트롤러에는 PM 혹은 DRAM 모듈들이 부착될 수 있다. 각 노드들은 QPI와 같은 IC(Interconnect) 링크에 의해 서로 연결된다.



(a) 기존 NOVA (b) 개선된 NOVA

그림 2: NOVA와 개선된 NOVA [3]의 파일 로깅 방식 비교

IC를 통하여 원격 노드의 메모리를 접근하는 경우 지역 노드의 메모리를 접근할 때보다 높은 지연시간을 보인다. NUMA 시스템에서 응용 프로그램의 성능을 최적화하기 위한 일부 알고리즘들은 쓰레드들의 원격 메모리 접근을 최소화시키는 것에 집중한다. 예를 들어 리눅스의 메모리 할당 정책 중 하나인 *first-touch*는 쓰레드가 현재 수행 중인 노드의 메모리를 우선적으로 할당받도록 한다; 이 쓰레드는 지역 메모리 접근을 통해 데이터를 접근할 수 있다. OS에 의해 관리되는 DRAM 공간과 다르게 PM 공간은 파일시스템이나 PMDK 같은 PM 전용 라이브러리에 의해 관리된다. 따라서 PM 파일시스템의 내부 메모리 할당자 역시도 NUMA를 고려하여 디자인 되어야 한다. 우리는 다음 섹션에서 NUMA 인지적인 PM 파일시스템에 대해서 설명한다.

2.2 NUMA 인지적 PM 파일시스템

기존 파일시스템 연구들은 매니코어 시스템에서 동시적인 파일 I/O가 수행될 때 I/O 처리량 측면에서 확장성을 제공할 수 있도록 노력해왔다. 그중 동시적 공유 파일 접근에 대한 I/O 성능을 개선하기 위한 파일시스템 최적화 연구들도 있었다. 많은 PM 파일시스템들에도 이러한 최적화 기술들이 적용되어왔지만 동시적 공유 파일 I/O 성능을 완전히 개선하지는 못했다. 매니코어 서버는 대부분 NUMA 구조이기 때문이다. 블락 기반의 파일시스템과 다르게 메모리 파일시스템들의 I/O 성능은 NUMA 구조의 비대칭적인 메모리 접근 속도에 큰 영향을 받았다.

예시로 대표적인 PM 파일시스템인 NOVA는 NUMA 기반의 매니코어 환경에서 그림 2(a)에 설명되어 있는 두 가지 이유로 낮은 동시성에 의한 확장성 문제를 가지고 있다. 그림 2(a)는 NOVA에서 I/O 쓰레드가 동시에 파일을 업데이트하려는 상황이다. 첫번째, NUMA를 인지하지 못한 NOVA는 모든 파일의 데이터와 메타데이터를 단일 노드의 PM 공간에 배치한다. 따라서 다른 노드에서 수행되는 쓰레드들은 원격 메모리 접근을 통해 파일에 접근해야 하므로 I/O 성능이 크게 저하된다. 두번째로 NOVA는 파일시스템의 일관성을 보호하기 위해 로그 구조의 디자인을 사용하며 PM의 아이노드 단위의 로그에 파일의 업데이트 내용을 기록한다. 로그 구조는 락으로 보호되기 때문에 동시적으로 공유 파일을 쓰기하는 쓰레드들은 이 락에 의해 직렬화된다. 뿐만 아니라 서로 다른 노드의 쓰레드들이 파일의 로그를 공유하는 경우 일부 쓰레드들에게 원격 메모리 접근을 통한 로깅이 강제된다.

이러한 문제를 해결하기 위해 최신의 연구들은 PM 파일시스템을 위한 NUMA 친화적인 기술들을 디자인하였다 [3, 4] . 특히 [3]에서 확장 가능한 PM 파일시스템을 디자인하기 위해 제안한 두 기술은 다음과 같다: 첫 번째, 파일시스템은 여러 NUMA 노드에 위치한 PM 모듈들을 가상화하여 단일 주소 공간으로 사용한다. 이때 파일시스템의 메모리 할당자에 리눅스의 *first-touch* 정책을 적용해 파일 데이터와 메타 데이터를 지역 PM 공간에 우선적으로 배치하여 원격 메모리 접근 문제를 줄였다. 두 번째, 락 프리한 CPU 코어 단위 파일 데이터 구조를 디자인하고 파일시스템에 적용하였다. 예를 들어 그림 2(b)는 이 디자인이 적용된 NOVA에서 동시적 공유 파일 쓰기가 일어나는 상황이다. NOVA 파일시스템의 아이노드 단위의 로깅은 락 프리 코어 단위 로깅으로 확장될 수 있다. 이 경우 쓰레드들은 지역 노드의 PM에 각자의 로그와 파일 데이터를 배치할 수 있다. 로그는 더 이상 공유되지 않으므로 쓰레드들은 동시에 파일을 업데이트할 수 있으며 원격 메모리 접근 문제 역시 완화 할 수 있다. 하지만 이는 초기 데이터(파일 혹은 로그) 배치만을 고려한 쓰기 최적화된 디자인이다. 읽기 쓰레드들은 여전히 원격 메모리 접근을 통해 파일에 접근할 여지가 있다. 원격 메모리 읽기의 오버헤드는 쓰기의 경우에 비해 적지만 더 큰 NUMA 노드 환경에서는 치명적일 수 있다. 우리는 이 가설을 확인하기 위해 매니코어 시스템에서 개선된 NOVA의 확장성 실험을 진행한다.

3 실험 결과

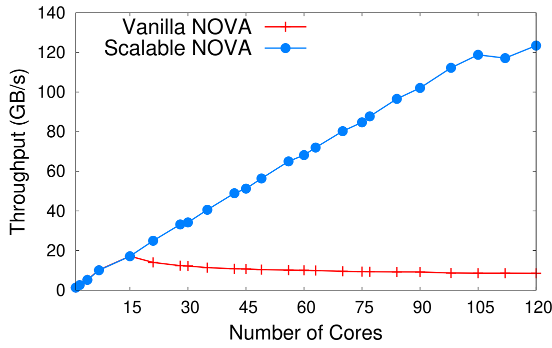
3.1 실험 환경 설정

우리는 8개의 NUMA 노드를 갖는 120코어 서버에서 실험을 진행하였다. 해당 서버는 실제 PM이 장착되어 있지 않기 때문에 DRAM을 기반으로 PM을 에뮬레이션하였다. PM 에뮬레이션을 위해서 리눅스 커널이 지원하는 DRAM 기반의 비휘발성 메모리 에뮬레이션 도구 [5]를 사용하였다. 자세한 서버 사양은 테이블 1에 나와 있다.

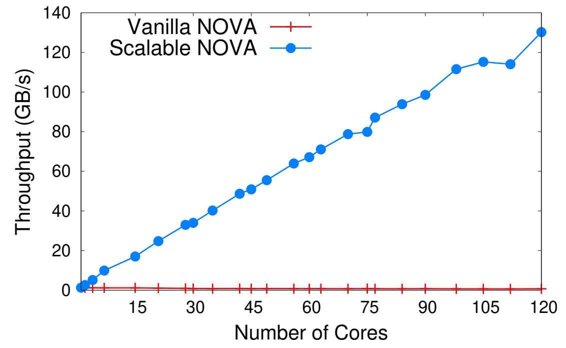
파일시스템의 확장성 측정을 위하여 다양한 워크로드 패턴을 생성할 수 있는 FxMark 벤치마크 [2]를 사용한다. 특히 우리는 다음과 같은 세 가지 병렬 I/O 워크로드를 사용한다. 개인 파일 쓰기 워크로드는 다중 쓰레드가 각자 파일에 동시적 쓰기를 수행한다. 공유 파일 쓰기 워크로드는 다중 쓰레드가 하나의 큰 파일에 동시적 쓰기를 수행한다. 공유 파일 혼합 워크로드는 쓰기 혹은 읽기를 수행하는 쓰레드 수의 비율을 변경하며 하나의 큰 파일에 대한 I/O를 수행한다. 개인 파일 혼합 워크로드 실험 결과는 공유 파일 혼합 워크로드의 결과와 유사하여 생략되었다. 우리는 쓰레드와 데이터 간의 원격 메모리 접근 문제를 명확히 확인하기 위해 모든 실험에서 I/O 쓰레드들을 CPU 코어에 1:1 바인딩하였다. 따라서 실험의 CPU 코어 수는 쓰레드 수와 같다.

표 1: 테스트베드

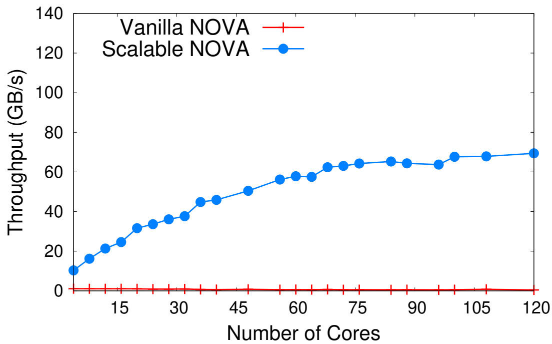
CPU	Intel(R) Xeon(R) CPU E7-8870 v2 2.30GHz CPU 소켓 (노드) 수 (#): 8 소켓 (노드) 당 코어 수 (#): 15 총 코어 수 (#): 120
메모리	DDR3, 96 GB * 8개 (=968GB)
비휘발성 메모리	32 GB (DRAM 기반 에뮬레이션)



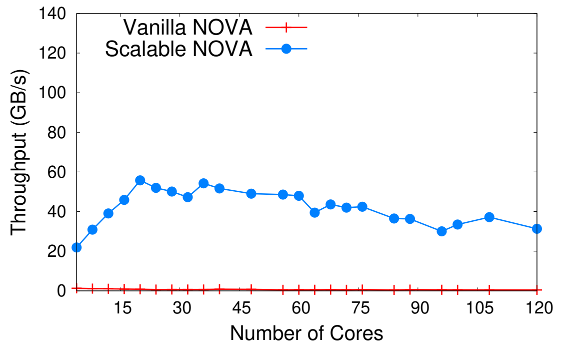
(a) 쓰기 전용 (개인 파일)



(b) 쓰기 전용 (공유 파일)



(c) 혼합 워크로드 (공유 파일 75% 쓰기, 25% 읽기)



(d) 혼합 워크로드 (공유 파일 25% 쓰기, 75% 읽기)

그림 3: 매니코어 서버에서 다양한 워크로드 패턴에 따른 NOVA 파일시스템의 확장성 평가.

3.2 실험 결과 및 분석

실험은 두 가지 NOVA 버전, 기존 NOVA(*Vanilla NOVA*)와 개선된 NOVA(*Scalable NOVA*)의 I/O 처리량을 비교하였다. 그림 3(a)와 그림 3(b)는 NOVA와 개선된 NOVA를 CPU 코어 수를 증가시켜가며 쓰기 전용 워크로드로 실험한 결과이다. 그림 3(a)의 개인 파일 쓰기의 경우 기존 NOVA의 I/O 처리량은 15코어에서 최대 17.15 GB/s까지 증가하였다가 이후로 감소하기 시작한다. 이는 기존 NOVA가 모든 파일 데이터가 단일 노드에 배치하여 서버의 NUMA 바운더리인 15코어 이후로 쓰레드들이 원격 메모리 접근을 통해 쓰기를 수행하기 때문이다. 특히 기존 NOVA는 그림 3(b)의 공유 파일 쓰기 시 파일시스템 락에 의해 전혀 성능 확장성을 보이지 못한다. 기존 NOVA는 1코어에서 1.29 GB/s의 I/O 처리량을 제공하며 코어가 증가 할수록 성능이 약간 하락한다. 반면 그림 2(b)에서 설명된 이유에 의해 개선된 NOVA의 I/O 처리량은 두 워크로드에서 모두 코어 수가 증가함에 따라 선형적으로 증가하였다.

그림 3(c)와 그림 3(d)는 두 NOVA를 CPU 코어 수를 증가시켜가며 공유 파일 혼합 워크로드로 실험한 결과이다. 그림 3(c)는 쓰기 쓰레드의 비율이 75%, 읽기 쓰레드의 비율이 25%일때의 실험 결과이다. 기존 NOVA의 파일시스템 락은 I/O 패턴의 읽기 비중이 높아져도 성능 병목지점이 되었다. 혼합 워크로드에서 기존 NOVA의 I/O 처리량은 공유 파일 쓰기 워크로드의 성능에 바운드 되었다. 개선된 NOVA의 성능은 여전히 CPU 코어 수 증가에 따라 선형적으로 증가하였지만, 쓰기 전용 워크로드들에 비해 증가폭이 높지 않았다. 이는 쓰기 쓰레드들이 지역 노드에 우선적으로 배치한 파일 데이터나 로그에 대해서 다른 노드의 읽기 쓰레드들이 원격 메모리 접근을 통해 접근할 가능성이 존재하기 때문이다. 그림 3(d)를 보면 읽기의 비율이 높아지고 노드가 많아질수록 읽기 쓰레드들의 원격 메모리 접근 문제는 심각해졌다. 개선된 NOVA의 I/O 처리량은 36코어에서 최대 54.24 GB/s를 제공했지만, 이후로는 성능이 오히려 하락하였다.

4 결론

우리는 최근 설계된 NUMA 인지적인 PM 파일시스템의 확장성 실험을 진행하고 분석하였다. 그 결과, 기존의 설계는 쓰기 최적화된 데이터 배치 방식을 채택하여 선형적인 성능 확장성을 보였다. 하지만 I/O 패턴 중 읽기 비율이 높아질수록 I/O 처리량이 낮아지는 문제가 있었다. 원격 메모리 읽기 오버헤드는 쓰기에 비해 비교적 적어 간과되었지만, 더 많은 NUMA 노드와 더 빠른 PM 환경에서는 시스템 전체 성능에 큰 영향을 주었다.

참고 문헌

[1] J. Xu and S. Swanson, “NOVA: A Log-structured File System for Hybrid Volatile/Non-volatile Main Memories,” in *Proceedings of the 14th USENIX Conference on File and Storage Technologies (FAST)*, 2016.

[2] C. Min, S. Kashyap, S. Maass, and T. Kim, “Understanding Many-core Scalability of File Systems,” in *Proceedings of the 2016 USENIX Conference on Usenix Annual Technical Conference (ATC)*, pp. 71–85, 2016.

[3] J.-H. Kim, Y. Kim, S. Jamil, and S. Park, “A NUMA-aware NVM File System Design for Manycore Server Applications,” in *Proceedings of the IEEE Intel Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, 2020.

[4] J. Wang, D. Jiang, and J. Xiong, “NUMA-Aware Thread Migration for High Performance NVMM File Systems,” in *Proceedings of the 36th International Conference on Massive Storage Systems and Technology (MSST)*, 2020.

[5] M. Maciejewski, “How to emulate Persistent Memory.” https://pmem.io/2016/02/22/pm-emulation.html, 2016.