NUMA 기반의 하이브리드 메모리 시스템에서 I/O 성능 최적화를 위한 성능 평가 및 분석

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

Performance Evaluation and Analysis for I/O Optimization in NUMA based Hybrid Memory Systems

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

요 약

새롭게 등장한 비휘발성 메모리 기술인 영구적 메모리(PM, Persistent Memory)는 메모리 버스 라인에 직접 연결되므로, 쓰레드가 PM의 데이터를 접근하는 성능은 NUMA(Non Uniform Memory Access) 구조에 의해 크게 영향 받을 수 있다. 기존 연구들은 NUMA 시스템에서 수행되는 응용 프로그램의 I/O 성능을 최적화하기 위해 노드간 통신 비용을 줄이고 메모리 접근 지역성을 개선 할 수 있는 데이터 배치 및 부하 분산 전략들을 사용한다. 하지만 DRAM과 매우 다른 PM의 특성 때문에 이러한 전략들은 하이브리드 메모리(PM-DRAM) 시스템 위에서 잘 작동하지 않을 수 있다. 본 연구에서는 NUMA 환경에서 PM의 I/O 성능 프로파일링을 통해 데이터 배치 및 부하 분산 시 고려되어야 하는 지표들을 살펴본다. 대표적인 실험 결과로 PM의 메모리 컨트롤러는 DRAM 기반의 메모리 컨트롤러에 비해 다중 쓰레드에 의한 경쟁에 취약하였다. 28개의 쓰레드가 특정 노드의 PM 메모리 컨트롤러에 대해 동시에 경쟁하는 경우 해당 노드 메모리의 대역폭은 최고 대비 13.7∼16.2× 감소하였다.

1 서 론

NUMA 구조는 최신 매니코어 서버들에 전반적으로 사용되고 있다. NUMA 기반의 서버는 여러 프로세서 노드들로 구성되며 각 노드는 멀티 코어 CPU와 메모리 컨트롤러를 포함한다 (그림 1). 각 노드들은 QPI(QuickPath Interconnect)와 같은 IC(Interconnect) 링크로 연결되며 이를 통해 원격 노드의 메모리에 접근 할 수 있다. NUMA 환경에서 응용 프로그램의 I/O 성능을 최적화하기 위하여 고려해야 하는 다음과 같은 두 중요한 요소들이 있다; 쓰레드들의 원격 메모리 접근은 지역 메모리 접근에 비해 느리다(*지역성*). 그리고 노드 간의 잦은 통신은 IC 링크와 메모리 컨트롤러에 혼잡을 초래한다(*트래픽 혼잡도*). 수 년간 진행되어 온 많은 NUMA 친화적인 시스템을 디자인하는 연구들은 두 요소들을 고려하여 I/O 쓰레드들과 데이터를 정적 혹은 동적으로 배치하기 위한 알고리즘들을 제안하였다 [1, 2].

최근 새로운 비휘발성 메모리 기술인 PM [3, 4]이 등장하였다. 이 메모리는 DRAM보다 더 큰 용량을 제공하면서 데이터의 영구성을 보장 할 수 있다. 또한 PM은 DRAM처럼 메모리 컨트롤러에 부착되기 때문에 NUMA 구조의 영향을 받을 수 있다. 단순히 느린 DRAM으로 가정 되어오던 PM이 실제 시장에 공개됨에 따라 여러 고유한 속성을 갖는 것으로 밝혀졌다 [5](자세한 내용은 섹션 2에서 설명). 하이브리드 메모리(DRAM과 PM) 시스템을 위한 NUMA 인지적인 데이터 및 쓰레드 배치 연구는 복잡한 PM의 특징에 의해 더 어려워졌지만 많이 진행되지 않았다 [6]. 특히, 존재하는 몇 연구들도 실제 PM이 아닌 DRAM에 에뮬레이션 된 PM을 기반으로 진행되었기 때문에 PM의 고유한 특성을 반영하지 못한다. 단순한

이 논문은 2020년도 정부 (과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No.2014-3-00035, 매니코어 기반 초고성능 스케일러블 OS 기초연구 (차세대 OS 기초연구센터)).

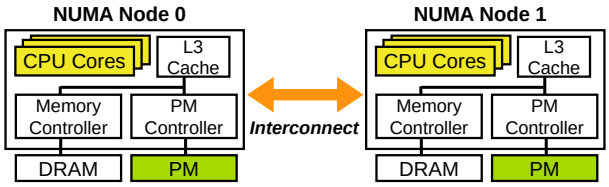


그림 1: NUMA 기반의 하이브리드 메모리 시스템 구조.

DRAM 기반의 기존 알고리즘들의 적용은 완전한 이점을 얻기 힘드며, 알고리즘은 더 많은 결정 요소들을 기반으로 설계되어야한다.

본 연구는 PM의 I/O 성능을 평가하고 분석함으로써 하이브리드 메모리 시스템에서 NUMA 인지적인 데이터 및 쓰레드 배치 시 새롭게 고려되어야 하는 지표를 설명한다. 프로파일링 결과를 요약하면 다음과 같다. 지역 노드 혹은 원격 노드의 PM에 배치 된 데이터에 대한 접근 지연 시간은 데이터 크기나 접근 방식(읽기 혹은 쓰기)에 의해 큰 영향을 받는다. 응용 프로그램의 데이터에 대한 접근 빈도 수만을 이주 결정의 지표로써 사용하는 기존의 알고리즘들은 데이터의 크기와 접근 방식 또한 고려 할 필요가 있다. 또한 쓰레드들의 동시적인 원격 노드의 PM 접근은 DRAM 접근에 비해 더 큰 메모리 컨트롤러의 혼잡을 유발한다. 이때 특정 노드의 PM에 배치된 데이터가 서로 다른 노드의 CPU 코어에 배치된 여러 쓰레드들에 의해 공유될 경우 I/O 성능의 심각한 병목지점이 될 수 있다. 따라서 NUMA 노드 간의 부하 분산 알고리즘들은 쓰레드와 데이터를 이주 시 데이터의 공유 정도 혹은 메모리 컨트롤러의 대역폭 측정 값을 지표로 사용하여야 한다.

2 배경 지식 및 연구 동기

기존 NUMA 시스템에서 응용 프로그램의 성능을 최적화하기 위한 알고리즘들은 동적으로 쓰레드 혹은 데이터를 이주시켜 원격 메모리 접근을 줄이려 노력한다. 하지만 오히려 잦은 이주로 인해 노드 간 트래픽 혼잡

이 야기 될 수 있으므로 알고리즘들은 단순히 지역성뿐만 아니라 데이터의 접근 빈도 수와 같은 지표를 사용해 이주 결정을 한다. 하이브리드 메모리 시스템에서 PM의 이질적인 I/O 성능(지연 시간과 대역폭)은 응용 프로그램의 데이터 접근 패턴만큼이나 중요한 이주 결정 요소가 된다. 예를 들어 실제 PM인 Intel Optane DC Persistent Memory [4]이 장착된 서버에서 지역 노드의 PM에 접근할 때의 지연 시간이 원격 노드의 DRAM을 접근하는 경우보다 높다. 실험 결과에 따르면 원격 노드의 DRAM 접근 평균 지연 시간은 140.5 $nsec$, 지역 노드의 PM 접근 평균 지연 시간은 172.4 $nsec$ 였다. 이 경우 원격 노드의 PM 데이터를 지역 노드로 이주시키는 것보다 원격 노드의 DRAM에 복사본을 놓는 것이 더 효율적이다. 하지만 이러한 접근 방식을 고려하기 위한 선행 연구들이 많이 진행되지 않았다.

우리는 이러한 관점에서 PM의 I/O 성능을 분석하고, 미래의 NUMA 인지적인 시스템 설계의 진행 방향을 제시한다. 우리는 하이브리드 메모리 시스템에서 수행되는 다중 쓰레드 프로그램의 데이터 처리에 대한 두가지 상황을 가정하였다. 첫번째는 쓰레드의 개인 데이터가 원격 PM에 배치되어 있는 경우이다. 쓰레드는 데이터를 낮은 지연 시간으로 접근하길 원하지만, 다수의 NUMA 시스템을 위한 쓰레드/데이터 배치 알고리즘들은 무조건적으로 쓰레드 혹은 데이터를 이주 시키지 않는다 [1, 2]. 대신 이들은 응용 프로그램의 데이터 접근 빈도 수 같은 하드웨어 자원 모니터링 값을 이주 결정에 지표로 사용한다. 하지만 새로운 메모리인 PM은 기존 메모리의 I/O 성능에는 큰 차이가 있다 [5]. 따라서 우리는 지역 노드와 원격 노드의 PM에 할당된 데이터를 접근할 때의 지연 시간 차이를 프로파일링한다 (3.1). 이 연구에서는 응용 프로그램에 따른 데이터 접근 패턴은 다루지 않으며 이는 차후 연구에서 진행 될 것이다.

두번째는 PM 데이터가 서로 다른 노드에 배치된 쓰레드들에 의해 공유되는 경우이다. 특정 노드의 메모리에 배치 된 데이터가 다른 노드들의 쓰레드들로부터 동시적으로 접근 될 때 노드 간의 많은 트래픽으로 인한 IC 링크와 메모리 컨트롤러의 혼잡 문제가 발생한다. 본 실험은 데이터에 대한 동기화 문제는 다루지 않으며 대역폭 측정을 통해 PM의 메모리 컨트롤러가 어느 정도의 부하를 감당할 수 있는지 프로파일링한다 (3.2).

3 PM I/O 성능 프로파일링

우리는 실제 PM의 고유한 특성을 이해하기 위해 Intel Optane DC PM 모듈이 장착 된 매니코어 서버에서 실험을 수행하였다. 서버의 세부 사양은 테이블 1에 설명되어 있다. 실험에 사용 된 하드웨어 설정은 다음과 같다. 단일 노드 내 여러 PM 모듈들의 메모리 공간은 운영체제에 의해 연속된 물리적 주소 공간으로 제공된다. 그러나 서버는 서로 다른 노드의 PM 모듈들을 집계하는 것을 지원하지 않는다.

3.1 데이터 지역성에 따른 지연 시간 측정

이 하위 섹션에서 우리는 PM에 배치된 오브젝트(문자열)를 읽고 쓰는 워크로드의 지연 시간을 측정한다. 우리는 대중적인 PMDK(Persistent Memory Development Kit) 라이브러리를 사용하여 두 노드에 각각 PM

표 1: Intel Optane DC PM 서버.

CPU	Intel(R) Xeon(R) Platinum 8280M v2 2.70GHz CPU 노드 수 (#): 2, 노드 당 코어 수 (#): 28
Memory	노드 당 모듈 수 (#): 6, DDR4, 64 GB * 12 (=768GB)
PM	Intel Optane DC Persistent Memory 노드 당 모듈 수 (#): 6, 128 GB * 12 (=1.5TB)
OS	리눅스 커널 4.13.0

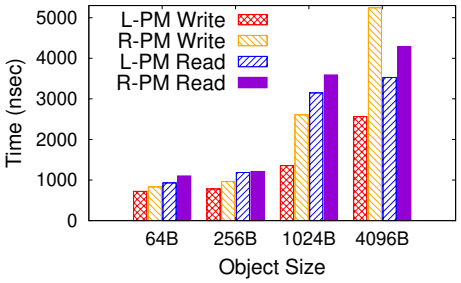


그림 2: 지역성과 크기에 따른 오브젝트 읽기 쓰기 지연 시간 비교.

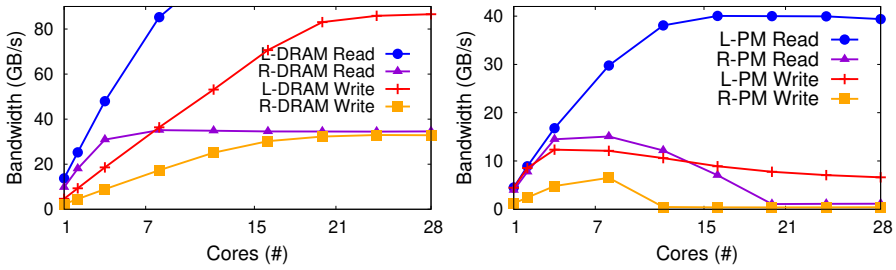
공간 할당자를 생성하였다. 단일 쓰레드는 이 메모리 할당자를 통하여 두 노드에 서로 다른 오브젝트를 각각 생성하고 PMDK의 함수를 이용하여 오브젝트를 읽거나 쓰기를 수행한다. 쓰레드는 한 노드의 CPU 코어 중 한 코어에만 바인딩 되며, 지역 노드의 오브젝트를 읽고 쓰는 경우 L-PM (Read/Write), 원격 노드의 오브젝트를 읽고 쓰는 경우 R-PM (Read/Write) 라고 명칭하였다.

그림 2는 PMDK의 읽기 쓰기 함수를 통해 다양한 크기의 오브젝트에 접근 시 평균 지연 시간을 측정한 결과이다. 그림 2를 보면 읽기의 지연 시간이 쓰기의 지연 시간보다 높은 것을 알 수 있다. 이는 Optane DC PM 서버의 독특한 특성으로 [5], 쓰기의 경우 데이터가 메모리 컨트롤러에 도달하는 순간 영구성을 보장받아 지연 시간이 낮다. 반면 읽기 지연 시간은 실제 PM 미디어로부터 데이터를 가져오는 딜레이가 포함되어 상대적으로 높다. 대표적인 결과로 256 B 오브젝트에 대해 R-PM Read의 지연 시간은 약 1212 $nsec$로 가장 높았으며 L-PM Read보다 약 36 $nsec$ 높았다. 256 B에서 R-PM Write의 지연 시간은 약 962 $nsec$으로 L-PM Write의 지연 시간보다 약 181 $nsec$ 높았다. 지역 노드와 원격 노드 메모리 접근 지연 시간의 차이는 오브젝트 크기가 더 커짐에 따라 더 커졌다. 4096 B의 오브젝트에 대해서 R-PM Read의 지연 시간은 4292 $nsec$로 L-PM Read보다 763 $nsec$ 가량 높았다. 반면 R-PM Write는 5244 $nsec$로 가장 높은 지연 시간을 보였으며, L-PM Write에 비해 2.04× 가량 높았다.

관측 내용: 데이터의 크기에 따라 지역 혹은 원격 PM에 대한 I/O 지연 시간이 상이하다. 읽기와 쓰기 워크로드 모두 오브젝트의 크기가 클수록 원격 메모리 접근에 의한 더 큰 패널티를 받았다. 특히 원격 PM에 큰 오브젝트를 업데이트 하는 경우 성능에 치명적인 영향을 받았다. 결론적으로 NUMA 인지적인 메모리 이주 알고리즘들은 데이터에 대한 접근 빈도 수뿐만 아니라 크기와 접근 방식(읽기 쓰기)를 고려 할 필요가 있다.

3.2 메모리 대역폭 측정

이 하위 섹션에서 우리는 읽기 및 쓰기 전용 워크로드로 CPU 코어 수를 늘려가며 PM 모듈의 대역폭을 평가한다. 기존 메모리 모듈과의 성능 비교를 위하여 DRAM 모듈의 대역폭도 실험하였다. 우리는 실험을 위해 Intel MLC(Memory Latency Checker) 도구를 사용하였다. MLC 도구는 여러 쓰레드가 메모리 모듈에 순차적으로 데이터를 읽거나 쓰는 워크로드의 최대 대역폭을 측정 할 수 있다. 읽기 쓰레드들은 64 바이트 *load* 명령어를 사용하며, 쓰기 쓰레드들은 64 바이트 *store* 명령어 호출 후 실패 원자성을 위해 추가적인 명령어 *CLWB*와 *mfence*를 수행한다. 각 워크로드 쓰레드들은 스케줄링 오버헤드를 최소화 하기 위해 단일 코어에 바인딩 된다. 이 실험에서 우리는 한 노드의 코어들만을 사용하며 네 종류의 메모리 성능을 비교한다: 지역 노드의 DRAM 공간(L-DRAM), 지역 노드의 PM 공간(L-PM), 원격 노드의 DRAM 공간(R-DRAM) 그리고 원격 노드의 PM 공간



(a) DRAM (b) PM

그림 3: 코어 수 증가와 지역성에 따른 메모리 모듈에 대한 읽기 쓰기 대역폭 측정 결과.

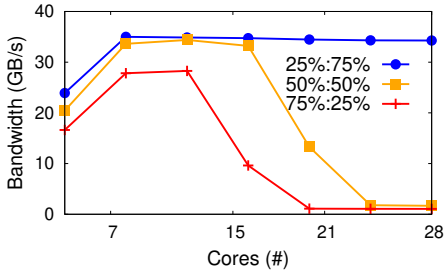


그림 4: 원격 PM과 DRAM에 접근하는 쓰레드의 비율에 따른 읽기 대역폭 측정 결과. 명각 A%:B%는 접근된 PM 대 DRAM의 비율을 의미한다.

(R-PM). 메모리 버퍼는 코어 별로 할당되며 코어 간에 공유되지 않는다.

그림 3(a)는 DRAM 모듈에 대한 읽기 쓰기 대역폭을 측정한 결과를 보여준다. 지역 노드의 메모리를 읽고 쓰는 경우 모두 DRAM의 대역폭은 28개 코어까지 선형적으로 증가한다. 반면 R-DRAM Read의 대역폭은 8코어에서 35.12 GB/s에 도달 한 이후 유지된다. 이는 IC 링크의 최대 대역폭에 도달해 더이상 대역폭이 증가하지 못한 것으로 추측된다. 마찬가지로 IC 대역폭에 의해 R-DRAM Write의 대역폭도 24코어에서 32.4 GB/s로 최대치를 보인다.

그림 3(b)는 PM 모듈에 대한 읽기 쓰기 대역폭을 측정한 결과를 보여준다. L-PM Write의 대역폭은 4코어에서 최대 12.3 GB/s까지 증가하고 4코어 이후 점차 감소한다. 반면 R-PM Write의 최대 대역폭은 8코어에서 약 6.5 GB/s이며 코어 수를 늘릴수록 계속 감소한다. 코어 수가 늘어남에 따라 쓰기 대역폭이 감소하는 이유는 다음과 같이 [5]에 설명된 Optane DC PM 서버의 하드웨어 한계로 추측된다: 데이터는 *CLFLUSH*나 *CLWB* 같은 PM *store* 명령어를 통해 캐시 라인에서 메모리 컨트롤러 내부의 WPQ(Write Pending queue)로 플러시 된다. WPQ에 도달한 데이터는 정전이 발생하여도 Optane DC PM 모듈로 플러시 됨을 보장받기 때문에 영구적이라고 할 수 있다. 하지만 큐에 대한 경합은 쓰기 대역폭에 부정적인 영향을 미칠 수 있다. 흥미롭게도 R-PM Write의 대역폭은 코어 수를 늘림에 따라 L-PM Write보다 더 빠르게 저하된다. 28개 코어에서 L-PM Write의 대역폭은 약 6.6 GB/s이지만 R-PM Write의 대역폭은 0.4 GB/s에 불과하다. 이것은 WPQ 문제와 잦은 원격 메모리 접근으로 인한 메모리 컨트롤러 부하 문제가 맞물려 더 심각한 성능 저하를 유발했다고 추측된다. L-PM Read의 대역폭은 코어 수가 증가함에 따라 28개 코어에서 40.04 GB/s까지 선형적으로 증가한다. R-PM Read의 대역폭도 8코어에서 최대 15 GB/s까지 증가하지만 쓰기의 경우에 설명 된 것과 동일한 이유로 8코어 이후부터 감소하기 시작한다.

관측 내용: 동시적인 원격 PM 접근으로 인한 트래픽 혼잡 문제는 DRAM에 비해 급격한 대역폭 감소를 유발한다. 그림 3(b)를 보면 8코어와 28코어에서의 대역폭을 비교했을 때, 대역폭이 R-PM Write의 경우는 16.2×, R-PM Read의 경우는 13.7×나 감소하였다. 그림 4는 이 문제를 구체적으로 분석하기 위하여 원격 DRAM과 PM의 데이터를 읽기를 수행하는 쓰레드의 비율을 변경해가며 대역폭을 측정한 실험 결과이다. 대표적인 결과로 20개의 코어에서 대조군들의 시스템 전체 대역폭은 각각 34.46 GB/s, 13.28 GB/s 그리고 1.09 GB/s로 원격 PM에 접근하는 쓰레드의 비율이 높아질수록 급하게 하락하였다. 분량 상 생략된 쓰기 실험에 대해서도 비슷한 양상을 보였다.

이처럼 단일 노드의 PM에 대한 경쟁은 시스템 전체의 성능에 큰 영향을 주므로 한정된 쓰레드만으로 메모리에 접근하는 방식이 효율적일 수 있다. 구체적으로 NUMA 친화적인 시스템에 전반적으로 적용되는 *Delegation* 기법이 있다 [7]. 주 아이디어는 전담 쓰레드(서버 쓰레드로 불린다)들이 다른 쓰레드들(클라이언트 쓰레드)의 일을 대신 수행해주는 방식이다. 각 노드마다 PM 데이터에 대한 읽기 쓰기를 수행하는 서버 쓰레드들을 할당할 경우 지역 메모리 접근을 통해 PM에 접근할 뿐만 아니라 PM에 대한 경쟁도 조절 할 수 있다. 서버 쓰레드와 클라이언트 쓰레드 간의 통신은 DRAM의 공유 메모리나 메세지 교환 방식을 통해 수행 할 수 있으며, 이와 관련된 선행 연구는 많이 존재한다.

4 결론

본 논문에서는 PM의 성능 분석을 통해 NUMA 기반의 하이브리드 메모리 시스템에서 I/O 성능 최적화를 위해 고려할 지표를 분석했다. 프로파일링 결과, 데이터의 크기와 접근 패턴에 따라 지역 노드와 원격 노드의 PM에 접근하는 지연 시간의 차이가 상이하였다. 또 여러 쓰레드가 동시에 특정 노드의 PM을 접근하는 경우 메모리 컨트롤러의 큐에 대한 경합으로 인해 시스템 전체의 대역폭이 하락하였다. 특히 원격 메모리 접근이 많을수록 더 큰 성능 하락폭을 보였다.

참고 문헌

[1] M. Dashti, A. Fedorova, J. Funston, F. Gaud, R. Lachaize, B. Lepers, V. Quéma, and M. Roth, "Traffic management: A holistic approach to memory placement on numa systems," vol. 48, pp. 381–394, 04 2013.

[2] S. Blagodurov, A. Fedorova, S. Zhuravlev, and A. Kamali, "A case for numa-aware contention management on multicore systems," in *2010 19th International Conference on Parallel Architectures and Compilation Techniques (PACT)*, pp. 557–558, 2010.

[3] Intel, "Revolutionizing Memory and Storage." https://www.intel.com/content/www/us/en/architecture-and-technology/intel-optane-technology.html, 2017.

[4] B. Beeler, "Intel optane dc persistent memory module (pmm)." https://www.storagereview.com/node/7416, 2019.

[5] J. Yang, J. Kim, M. Hoseinzadeh, J. Izraelevitz, and S. Swanson, "An empirical guide to the behavior and use of scalable persistent memory," in *18th USENIX Conference on File and Storage Technologies (FAST 20)*, (Santa Clara, CA), pp. 169–182, USENIX Association, Feb. 2020.

[6] Z. Duan, H. Liu, X. Liao, H. Jin, W. Jiang, and Y. Zhang, "Hinuma: Numa-aware data placement and migration in hybrid memory systems," in *2019 IEEE 37th International Conference on Computer Design (ICCD)*, pp. 367–375, 2019.

[7] I. Calciu, J. Gottschlich, and M. Herlihy, "Using elimination and delegation to implement a scalable numa-friendly stack," in *5th USENIX Workshop on Hot Topics in Parallelism (HotPar 13)*, (San Jose, CA), USENIX Association, June 2013.