CXL 기반의 계층적 메모리 구조 관리를 위한 선택적 Shadow Paging

김정헌 ○, 신동군
성균관대학교 전자전기컴퓨터공학과
kjh990705@skku.edu, dongkun@skku.edu

Selective Shadow Paging for CXL-based Hierarchical Memory Structure Management

Jungheon Kim○, Dongkun Shin
Department of Electrical and Computer Engineering, Sungkyunkwan University

요 약

CXL 메모리의 등장으로 계층화 된 메모리 시스템이 상용화되고 있다. 그러나 CXL 메모리는 기존의 SSD와 같은 저장 장치보다 속도가 빨라 이전에는 드러나지 않았던 소프트웨어 오버헤드 문제가 발생하고 있다. 이러한 오버헤드 문제를 해결하기 위한 메모리 관리 방식들이 연구 되고 있다. 그 중에서도 Non-Exclusive 메모리 계층화 전략을 도입한 NOMAD가 가장 우수한 성능을 보인다. 하지만 NOMAD는 잦은 Shadow Page Fault와 불필요한 메모리 공간 낭비라는 단점을 가지고 있다. 본 연구에서는 Page의 빠른 계층 메모리 History 기반 선택적 Non-Exclusive 메모리 계층화 전략을 제안한다. Promotion 된 모든 Page의 복사본을 유지하는 대신, 유효할 것으로 예상되는 복사본만 유지함으로써 NOMAD의 Quick Demotion 효과를 유지하면서도 불필요한 오버헤드를 줄일 수 있다. 이 전략은 마이크로 벤치마크 평가에서 NOMAD 대비 최대 15%의 성능을 개선 시켰으며 메모리 낭비를 최대 50% 수준으로 감소 시켰다.

1. 서 론

Compute Express Link(CXL) 메모리의 등장으로 현대 컴퓨터 시스템은 용량과 속도가 다른 다양한 메모리 계층을 갖추게 되었다. 운영체제는 메모리 관리를 최적화하기 위해 Page 할당, 회수, 복사 등의 작업을 수행한다. 계층적 메모리 시스템에서는 Page 복사 대신 Page 이동이 주로 이루어진다. Page 이동은 Hot Page를 느린 계층 메모리에서 빠른 계층 메모리로 Promotion하거나, Cold Page를 빠른 계층 메모리에서 느린 계층 메모리로 Demotion하는 방식으로 동작한다. Promotion은 프로그램 실행 중 발생하는 동기식 Page 이동으로 프로그램을 차단시키며, 이는 Page 접근을 막고 Page Fault를 유발하여 높은 비용을 초래한다.

NOMAD[1]는 이러한 비용을 줄이기 위해 Non-Exclusive 메모리 계층화와 Transaction 기반 Page 이동 및 복사를 특징으로 하는 새로운 계층형 메모리 Page 관리 메커니즘을 제안했다. 이 메커니즘은 Page를 Promotion 할 때 Page 이동 대신 복사를 수행하여 비동기식 Promotion을 가능하게 한다. NOMAD는 Page Promotion 시 복사된 Page인 Master Page를 빠른 계층 메모리에 저장하고, 원본 Page인 Shadow Page를 느린 계층 메모리에 유지한다. Page가 Demotion 될 때 Shadow Page가 존재한다면, Page 이동 없이 Master Page에서 Shadow Page로 재매핑(Remap)하여 Quick Demotion을 가능하게 한다.

NOMAD는 Shadow Page 관리로 인해 오버헤드가 발생한다. 첫 번째 오버헤드는 잦은 Shadow Page Fault의 발생이다. NOMAD는 Page 일관성을 유지하기 위해 Master Page에 Write가 수행되는 경우, Shadow Page Fault를 발생시켜 해당 Master Page의 Shadow Page를 삭제한다. **표1**은 Write를 포함하는 Write 워크로드와 합성 워크로드에서 발생하는 Shadow Page Fault 횟수와 Shadow Page Reclaim 횟수를 보여주며, 실험 환경은 **4**와 동일하다. **표1**에서 볼 수 있듯이, NOMAD는 Write, 합성 워크로드에서 불필요한 Shadow Page Fault가 자주 발생하는 것을 확인할 수 있다.

표 1. Shadow Page Fault, Reclaim 횟수

구분	Fault	Reclaim
Write 워크로드	236,183	14,065,758
합성 워크로드	215,714	13,541,511

두 번째 오버헤드는 메모리 공간 낭비이다. Shadow Page는 Quick Demotion을 위해 사용되지만 실제 데이터 접근은 이루어지지 않는다. 표1에서 볼 수 있듯이, 느린 계층 메모리는 Shadow Page로 인한 공간 낭비로 Shadow Page Reclaim이 빈번히 발생한다.

* 이 성과는 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No.NRF2021R1A2C2095125).

불필요한 Shadow Page 생성으로 발생한 공간 낭비는 필요한 Shadow Page 회수를 초래하며, SSD가 탑재된 시스템에서는 Page Swap으로 인한 Disk I/O를 유발할 수 있다. 본 연구는 NOMAD의 잦은 Shadow Page Fault를 줄이고 느린 계층의 공간 낭비를 개선하기 위한 선택적 Shadow Paging (SSP)을 제안한다.

2. 선행 연구

여러 계층의 메모리를 관리하기 위한 다양한 연구들이 제안되었다. **Memtis**[2]는 메모리 hotness 추적을 최적화하였다. 메모리에 할당된 Page들의 접근 분포를 분석하여 Hot Page를 빠른 계층 메모리에 최적으로 배치하는 방법을 제안한다. 그러나 이 방법은 지속적인 메모리 추적이 필요하기 때문에 메모리 압박이 높은 워크로드에서는 적합하지 않다. 또 다른 연구로는 **Transparent Page Placement**(TPP)[3]가 있다. TPP는 비동기식 Page Demotion을 제안한다. 하지만 Page Promotion 시 여전히 동기적으로 Page Migration을 수행하는 문제가 있으며, NOMAD에 비해 최대 8배의 성능 저하를 보였다.

3. 본 론

3.1 NOMAD

NOMAD는 Page에 접근이 발생하면 **PG_referenced** 플래그를 활성화하고, 해당 Page를 Promotion Candidate Queue(PCQ)에 삽입한다. PCQ는 후보 Page들만을 탐색하여 Hot Page를 신속하게 추적한다. 만약 접근한 Page의 **PG_referenced** 플래그가 이미 활성화된 상태라면 **PG_active** 플래그를 활성화한다. PG_active 플래그가 활성화된 Page들은 Hot Page로 간주되어 PCQ에서 Migration Pending Queue로 이동하며, 이후 **kpromote** 스레드에 의해 비동기식으로 Promotion된다. NOMAD는 Promotion된 Page에 대해 Master Page를 생성하여 빠른 계층 메모리에 저장하며, 원본 Page는 Shadow Page 형태로 느린 계층 메모리에 유지한다. Master Page는 읽기 권한만 부여되어 있어 Write 요청 시 Shadow Page Fault가 발생한다. Shadow Page Fault가 발생하면 Master Page에 연결된 Shadow Page를 회수하고, Master Page의 권한을 복구한 뒤 요청된 Write 작업을 수행한다. Page가 Demotion될 때 NOMAD는 **PG_shadowed** 플래그를 사용해 Shadow Page의 존재 여부를 확인한다. Shadow Page가 존재하면 페이지 테이블 엔트리를 업데이트해 Shadow Page를 가리키도록 포인터를 Remap하여 데이터 복사 없이 Quick Demotion을 수행한다. 이 과정에서 기존 Master Page에 대한 참조를 Shadow Page로 간단히 전환함으로써, Demotion 시 불필요한 데이터 이동을 방지하고 메모리 관리 효율성을 크게 높인다.

3.2 Selective Shadow Paging

그림 1은 SSP 메커니즘의 전체 설계를 보여준다. SSP는 Nomad 시스템의 전반적인 흐름을 유지하면서 Page 구조체에 **PG_fastdirty** 플래그를 도입하여 빠른 계층 메모리의 History를 효과적으로 관리한다. 이러한 관리 방식을 통해 계층적 메모리 시스템에서 Promotion 이후 Page의 Write 발생 여부를 추적할 수 있다 SSP는 **PG_fastdirty** 플래그의 상태에 기반하여 선택적으로 Shadow Page 복사 작업을 수행한다. Promotion 과정에서 SSP는 Promotion 되는 각 Page의 **PG_fastdirty** 플래그를 확인한다. 플래그가 0으로 설정되어 있으면 Shadow Page Copy 작업을 수행하고, 1로 설정되어 있으면 Shadow Copy를 유지하지 않고 기존 NOMAD의 Promotion과 동일한 비동기식 Page Migration을 진행한다. Page가 처음 생성될 때 **PG_fastdirty** 플래그는 0으로 설정되어 있어, Page의 최초 Promotion 시 Shadow Copy가 생성되도록 보장한다. Promotion이 발생하면 Shadow Page 복사와 Page Migration 모두 **PG_fastdirty** 플래그를 0으로 초기화한다. 이를 통해 기존의 **PG_dirty** 플래그와 달리 **PG_fastdirty** 플래그는 빠른 계층 메모리에서의 Dirty 상태만을 나타낸다. Promotion된 Page에 Write 요청이 있으면 SSP는 **PG_fastdirty** 플래그를 1로 설정하여 빠른 계층 메모리에서 해당 Page의 Dirty 상태를 표시한다. Demotion 과정에서는 **PG_fastdirty** 플래그 값이 유지된 채로 Page가 이동되며, 이를 통해 이전 빠른 계층 메모리의 상태 정보를 보존한다. 이후의 Promotion 단계에서는 SSP가 **PG_fastdirty** 플래그를 참조하여 빠른 계층 메모리의 히스토리를 확인하고, 이에 따라 Shadow Page Copy를 수행할지 결정한다. 빠른 계층 메모리에서 Clean 상태로 Demotion된 Page는 Promotion 시 Shadow Copy 생성이 허용되지만, Dirty 상태로 Demotion된 Page에 대해서는 Shadow Copy 작업을 제한한다. 이러한 방식을 통해 SSP는 Page의 동적 상태에 기반한 선택적 Shadow Page 작업을 수행하여 메모리 관리의 효율성을 최적화한다.

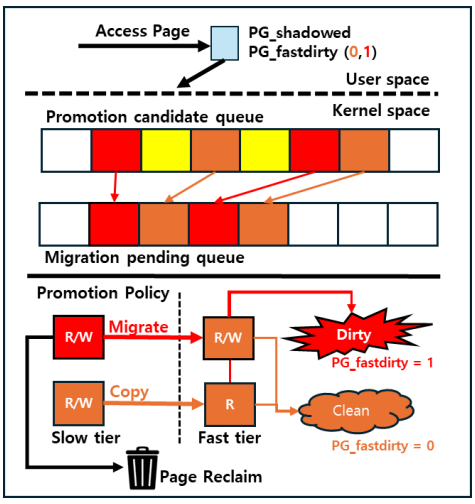

그림 1. Selective Shadow Paging

4. 실 험

4.1 실험 환경

본 연구는 SSP 구현을 위해 Linux 5.13.0 기반

NOMAD 커널을 수정하였으며, CXL 메모리 에뮬레이션에 POND[4]의 아이디어를 활용하였다. 두 개의 소켓을 갖춘 서버에서 두 노드로 구성된 QEMU+KVM 가상 머신을 구축하였다. 첫 번째 가상 노드는 물리 서버의 첫 소켓에 매핑 되고, 두 번째 노드는 두 번째 소켓의 물리 메모리에만 매핑 되었으며 CPU 없이 구성되었다. 각 노드에는 16GB의 메모리가 할당되었다.

4.2 실험 방법

본 연구에서는 NOMAD의 마이크로 벤치마크를 수정하여 사용하였다. 실제 메모리 접근 패턴을 모방한 Zipfian 분포 기반 워크로드로 메모리 접근 테스트를 수행하였다. 실험에는 Read 워크로드, Write 워크로드, 그리고 Read와 Write를 1:1 비율로 수행하는 합성 워크로드(Mixed)를 사용하였다. 모든 워크로드는 13.5GB의 작업 집합 크기와 27GB의 상주 집합 크기로 구성되었다.

4.3 실험 결과

실험은 NOMAD와 SSP에서 워크로드 별로 8회씩 진행하였다. **그림 2**는 NOMAD와 SSP에서 수행한 합성 워크로드 결과를 박스 플롯으로 나타낸 그림이다. 실험 결과, SSP는 NOMAD에 비해 Shadow Page Fault와 Shadow Page Reclaim 횟수를 크게 줄일 수 있었으며, Shadow Page Remap 횟수는 NOMAD와 동일한 수준으로 유지됨을 확인할 수 있었다. **그림 4**는 워크로드 진행에 따른 Shadow page 생성 횟수를 나타낸다. SSP는 NOMAD 대비 약 74%의 Shadow Copy만 수행하였다. 이는 SSP가 Shadow Page Fault를 유발하는 Page에 대해 Shadow Page Copy를 제한하고, Shadow Page Remap이 예상되는 Page에만 Shadow Page Copy를 수행했기 때문이다. **그림 3**은 Write와 Read 단일 워크로드에 따른 결과를 보여준다. Shadow Page Fault와 Reclaim은 Write 워크로드의 결과이며, Shadow Page Remap은 Read 워크로드의 결과이다. Write 워크로드에서 SSP는 Shadow Page Fault와 Reclaim 횟수를 크게 감소시켰다. **그림 4**에서 확인할 수 있듯이 SSP는 NOMAD 대비 약 50%의 Shadow Page만 생성하였다. **그림 5**에 따르면 SSP는 Shadow Page Fault를 크게 줄인 결과로, NOMAD 대비 약 115%의 대역폭을 달성할 수 있었다. Read 워크로드에서 SSP는 NOMAD와 동일한 수준의 Shadow Page Remap과 Shadow Page 생성을 수행했다.

5. 결 론

본 연구에서는 선택적 Shadow Paging 기법을 제안한다. SSP는 Page의 고속 계층 메모리 History를 관리함으로써 Shadow Page를 선택적으로 복사한다. 이를 통해 Shadow Page Fault의 발생 빈도를 감소시키고, Shadow Page로 인한 공간 낭비를 최소화하는 동시에, Shadow Page Remap을 NOMAD와 동일한 수준으로 유지한다. 실험 결과, SSP는 NOMAD에 비해 메모리 사용 효율성이 향상되었으며, 시스템 성능 저하 없이 안정적인 Shadow Page 관리가 가능함을 확인하였다.

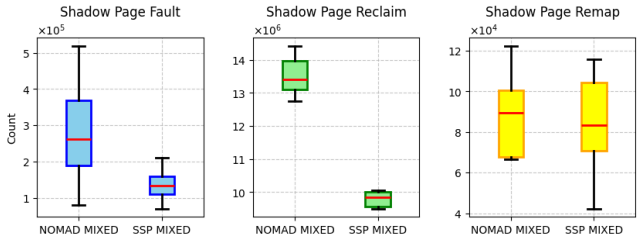

그림 2 Synthetic Workload

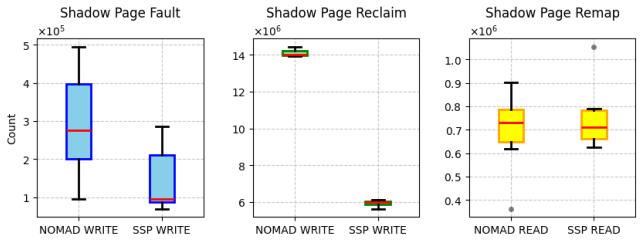

그림 3 Write/Read Workload

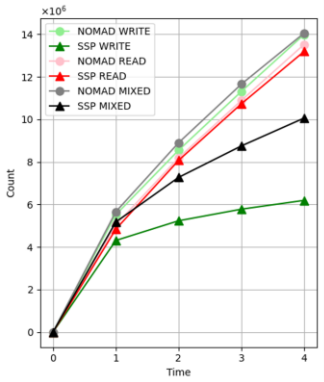

그림 4 Shadow Page 생성 횟수

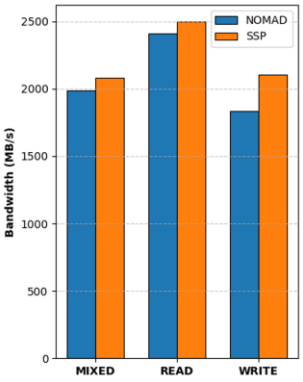

그림 5 Bandwidth

참고 문헌

[1] Xiang, Lingfeng, et al. "Nomad: Non-Exclusive Memory Tiering via Transactional Page Migration." Proceedings of the 18th USENIX Symposium on Operating Systems Design and Implementation (OSDI `24), p. 19–35.

[2] LEE, T., et al. "Memtis: Efficient Memory Tiering with Dynamic Page Classification and Page Size Determination." Proceedings of the 29th Symposium on Operating Systems Principles (SOSP '23), p. 17–34.

[3] MARUF, H. A., et al. "TPP: Transparent Page Placement for CXL-Enabled Tiered-Memory." In Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS `23), p. 742–755.

[4] LI, H., et al. “Pond: CXL-Based Memory Pooling Systems for Cloud Platforms.” In Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS `23), p. 574–587.