한국컴퓨터정보학회 하계학술대회 논문집 제26권 제2호 (2018. 7)

DMA를 사용한 페이지 Zeroing을 통한 Linux 기반 시스템의 사용자 응답성 향상 기법

양석우[O], 김정호[*]
[O]서울대학교 전기정보공학부
[*]서울대학교 융합과학기술대학원 융합과학부
e-mail: {swyang, jhkim}@redwood.snu.ac.kr[O*]

User Interactivity Improving Mechanism in Linux-based Systems by Using Page Zeroing with DMA

Seok-Woo Yang[O], Jung-Ho Kim[*]
[O]Dept. of Electrical and Computer Engineering, Seoul National University
[*]Department of Transdisciplinary Studies, GSCST, Seoul National University

● 요 약 ●

데스크탑과 모바일 기기가 고성능화됨에 따라 다양한 분야에서 고사양 응용들이 출시되고 있다. 이러한 응용들의 응답성은 사용자 경험을 결정하는 중요한 요소들 중 하나이다. 측정에 따르면 고사양 응용들 중 하나인 웹 브라우저의 응답시간에서 페이지 zeroing에 소요되는 시간이 적지 않은 비중을 차지한다. 또한 페이지 zeroing이 발생시키는 캐시 오염에 의해 추가적인 성능 저하가 발생한다. 본 논문은 페이지 zeroing에 소요되는 시간을 단축하고, zeroing에 의한 캐시 오염으로 인해 발생하는 시스템의 성능저하를 방지하기 위한 기법을 제안한다. 제안된 기법은 사용자 응답시간이 아닌 구간에서 페이지들을 DMA를 사용하여 캐시를 거치지 않고 zeroing하여 보관해 두었다가 페이지 할당 요청시 선 zeroing 된 페이지들을 응용에게 제공한다. 이를 Linux 커널 4.17이 탑재된 데스크탑 환경에서 구현하였고 실험을 통해 확인한 결과 응답시간이 평균 20% 단축됨을 확인하였다.

키워드: 리눅스(Linux), 사용자 응답시간(user response time), 페이지 zeroing(page zeroing)

I. Introduction

데스크탑과 모바일 기기가 고성능화 됨에 따라 증강 현실, 게임, 멀티미디어, 웹 서비스 등 다양한 분야에서 고사양 응용들이 출시되고 있다. 이러한 고사양 응용들은 대개 사용자의 입력에 따라 해상도가 높고 레이아웃 구조가 복잡한 프레임의 렌더링을 수행한 후 화면에 출력한다. 이 과정에서 많은 양의 물리 메모리가 요구되는데, 물리메모리는 Linux의 물리 메모리 관리 메커니즘에 따라 응용에게 제공된다. 따라서 Linux의 물리 메모리 관리 메커니즘은 고사양 응용들의 사용자 응답성을 결정하는 중요한 요소들 중 하나이다.

본 연구진은 물리 메모리 관리 메커니즘이 사용자 응답시간에 미치는 영향을 측정하기 위하여 Linux 기반 데스크탑 환경에서 사용자 응답시간동안 수행되는 일련의 과정을 분석하였다. 이 때 사용자 응답시간은 마우스/키보드로부터 사용자의 입력이 수신된 순간부터 입력에 대한 결과 프레임이 화면에 출력되기까지의 시간이다.

기존 연구에 따르면 고사양 응용들 중 하나인 웹 브라우저의 총 응답 시간에서 물리 메모리 관리 메커니즘의 페이지 zeroing 작업에 소요되는 시간이 약 3%로 적지 않은 비중을 차지한다[4]. 뿐만 아니라 웹 브라우저의 사용자 응답시간 중 발생한 LLC(last level cache) 미스 중 페이지 zeroing에 의한 LLC 미스가 약 21%를 차지하였다.

이렇듯 사용자 응답시간에 페이지 zeroing이 큰 영향을 끼치는 이유는 다음과 같다. 사용자 응답시간의 대부분을 차지하는 프레임 렌더링은 많은 양의 동적 메모리 공간을 할당받는다. 이렇게 할당받은 동적 메모리 공간에 접근하는 과정에서 많은 수의 마이너 폴트들이 발생하게 된다. 마이너 폴트는 페이지 폴트의 한 분류이며, 할당받은 가상 메모리 공간에 태스크가 최초로 접근할 때 발생하는 페이지 폴트이다. 이러한 마이너 폴트가 발생하면 마이너 폴트 핸들러가 커널에서 물리 메모리 공간을 제공받아 응용에게 할당한다. 이 때 커널은 보안을 위해 필수적으로 페이지에 0을 채우는 zeroing을 수행한다. 페이지 zeroing을 통해 다른 응용이나 커널의 데이터가 유출되는 것을 방지한다.

측정에 따르면 zeroing 과정은 전체 마이너 폴트 처리 시간의 약 20%를 차지한다. 뿐만 아니라 마이너 폴트 처리 중 발생하는

LLC 미스의 약 47%가 페이지 zeroing에 의해서 발생한다. 이러한 LLC 미스에 의하여 캐시에 존재하는 데이터들이 방출되는 캐시 오염(cache pollution) 현상이 발생한다. 캐시 오염 현상은 사용자 응답성이 중요한 응용뿐만 아니라 백그라운드에서 수행되는 다른 응용들의 캐시 적중 확률을 떨어뜨리고, 따라서 추가적인 메모리 접근을 발생시킨다. 캐시 접근에 비하여 긴 지연을 가지는 메모리 접근이 많이 발생하면 응용의 성능이 저하된다. 이렇듯 페이지 zeroing은 다양한 면에서 시스템의 성능을 저하시키며, 따라서 페이지 zeroing에 의한 성능 저하를 줄이는 것은 중요하다.

페이지 zeroing으로 인한 성능 저하를 줄이는 기존 연구는 크게 페이지 zeroing의 발생 횟수를 줄이는 연구와 페이지를 미리 zeroing 해두는 연구로 분류된다. 우선 페이지 zeroing의 발생 횟수를 줄이는 기법으로는 커널 레벨 페이지 재사용 기법이 있다[1]. 이 기법은 커널 수준에서 각 태스크별로 메모리 공간을 관리하며 각 태스크가 해제한 페이지를 태스크별 메모리 공간에 저장해 두었다가, 태스크가 페이지를 요청하면 재할당 함으로써 페이지 zeroing 없이도 데이터의 유출을 막는다. 하지만 이 기법은 기존에 존재하는 응용들과 호환되지 않는다는 단점이 있다. 기존의 응용들은 페이지가 zeroing된 것을 전제로 구현되어 있는 경우가 있기 때문이다.

페이지 zeroing을 미리 수행하는 기법으로는 Windows 커널의 zero page thread[2]와 Linux 기반 시스템의 적응형 페이지 선 zeroing 기법[3]이 있다. Zero page thread는 별도의 시스템 thread가 미리 자유 페이지들에 대하여 zeroing 과정을 수행하고, 응용에게 미리 zeroing된 페이지를 제공하는 기법이다. 이를 통하여 페이지 할당 요청시 페이지 zeroing에 의한 지연을 제거한다. 하지만 이 기법 현재 수행중인 프로세스가 없을 때만 페이지 zeroing을 수행하기 때문에 굉장히 보수적으로 페이지 zeroing을 하여 성능 이득이 적다는 단점이 있다. 적응형 페이지 선 zeroing 기법은 별도의 시스템 thread가 미리 zeroing을 수행한다는 점은 zero page thread와 동일하지만, 현재 수행중인 태스크가 있더라도 사용자 응답시간 중이 아니라면 페이지 zeroing을 적극적으로 수행한다. 하지만 이 기법 역시 [2]와 마찬가지로 페이지 zeroing을 수행할 때 캐시 오염이 발생한다는 단점이 있다.

본 논문은 DMA를 사용한 사용자 응답시간 인식 페이지 선 zeroing 기법을 제안한다. 이 기법은 사용자 응답시간이 아닌 시간 구간에 DMA를 사용하여 페이지 zeroing을 수행한 후, 커널에서 zeroing된 페이지를 별도로 관리하다가 zeroing된 페이지의 할당 요청이 발생했을 때 제공한다. 제안된 기법은 기존 응용 프로그램이나 라이브러리의 수정 없이, 커널만을 수정하여 구현되었으므로 앞서 언급된 [1]의 단점을 극복하였다. 또한 제안된 기법은 사용자 응답시간 중이 아니라면, DMA를 사용하여 캐시 오염 없이 zeroing을 적극적으로 수행하여 앞서 언급된 [2]와 [3]의 단점을 해결하였다.

제안된 기법을 Linux 기반 시스템과 X Window GUI 프레임워크 환경에서 구현하고 사용자 응답시간을 측정하였다. 실험 결과 기존의 커널에 비하여 제안된 기법이 적용된 커널에서 응답 시간이 평균 약 20% 단축되었다.

II. Linux의 물리 메모리 할당 메커니즘

본 장에서는 제안된 기법의 이해를 돕기 위해 Linux의 물리 메모리 할당 메커니즘을 설명한다. Linux의 물리 메모리 할당 메커니즘은 물리 메모리를 관리하는 두 계층의 자료구조와, 이 자료구조에 기반하여 메모리 공간의 할당과 해제를 수행하는 할당자로 이루어져 있다. 본 장에서는 Linux의 물리 메모리 관리 자료구조에 대해서 먼저 설명하고, 물리 메모리 할당자의 동작을 설명한다.

Linux는 보통 4KB 크기의 페이지 단위로 물리 메모리를 관리하며, 자유 페이지들의 전역적인 리스트를 가지고 있다. 그리고 CPU의 각 코어 별로 전역 자유 페이지 리스트로부터 일정 수의 페이지를 가져와 코어 당 자유 페이지 리스트를 만들어 관리한다. 코어 별로 자유 페이지 리스트를 구성함으로써 여러 태스크들이 동시에 전역 자유 페이지 리스트에 접근하면서 발생하는 경쟁 상태(race condition)를 줄일 수 있다.

Linux의 물리 메모리 할당자 동작은 할당과 해제 동작으로 구성된다. 먼저 할당을 설명하고 해제에 대하여 설명하겠다. 물리 메모리 할당은 요청되는 페이지의 개수에 따라서 전역 또는 코어 당 자유 페이지 리스트에서 페이지를 제공받는다. 요청된 페이지의 개수가 두 개 이상이면, 커널은 전역 자유 페이지 리스트로부터 페이지를 제공받는다. 요청되는 페이지의 개수가 한 개이면 코어 당 자유 페이지 리스트에서 페이지를 제공받는다. 만약 코어 당 자유 페이지 리스트가 비어 있다면, 커널은 전역 자유 페이지 리스트에서 일정 개수의 페이지를 가져와 코어 당 페이지 리스트를 채운 다음, 재할당을 시도한다.

물리 메모리 해제시에는 해제된 페이지들의 연속성과 개수에 따라 전역 자유 페이지 리스트 또는 코어 당 자유 페이지 리스트로 반환된다. 여러 개의 연속적인 페이지들이 해제될 때는 전역 자유 페이지 리스트로 반환되고, 그 외의 경우에는 코어 당 자유 페이지 리스트로 반환된다. 이 때 코어 당 페이지 리스트에 지나치게 많은 페이지가 쌓이는 것을 막기 위해 페이지 해제 과정에서 코어 당 자유 페이지 리스트의 크기가 일정 이상으로 커지면 미리 설정된 개수만큼 전역 자유 페이지 리스트로 페이지들을 반환한다.

III. DMA를 사용한 페이지 zeroing 기법

본 논문에서 제안하는 기법의 목표는 zeroing에 의한 사용자 응답시간 지연을 단축하는 것이다. 이를 위해 DMA를 사용하여 페이지를 선 zeroing하고, 페이지 할당 요청시 선 zeroing된 페이지를 제공함으로써 페이지 할당 과정에서 zeroing에 소요되는 시간과, zeroing 때문에 발생하는 캐시 오염을 줄인다. 그리고 사용자 응답시간을 줄이기 위해 사용자 응답시간이 아닌 시간 구간을 인식하고, 해당 시간 구간 동안만 zeroing을 수행한다.

다음은 제안된 기법을 위해 기존 Linux의 물리 메모리 할당 메커니즘에서 수정 또는 추가된 요소들이다. 첫째, zeroing된 페이지의 할당 요청이 발생했을 때 선 zeroing된 페이지를 제공하는 zeroed 페이지 할당자. 둘째, 자유 페이지의 선 zeroing을 수행하는 페이지 zeroing 관리자. 셋째, 페이지 zeroing 관리자가 동작할 시점을 알려주는 사용자 응답시간 인식기. 본 장의 나머지 부분에서는 각 요소에

Table 1. 실험 환경

HW	CPU	Intel i7-6700k
	Memory	8GB DRAM
SW	Framework	X Window11 R7.7
	Kernel	Linux Kernel 4.17.3

대해서 구체적으로 설명한다.

3.1. Zeroed 페이지 할당자

Zeroed 페이지 할당자는 zeroing된 페이지들을 관리하며, 페이지 할당 요청이 발생했을 때 zeroing된 페이지를 제공한다. 이를 위하여 zereod 페이지 할당자는 제안된 기법에서 추가된 자료구조인 전역 zeroed 페이지 리스트와 코어 당 zeroed 페이지 리스트를 가진다. 그리고 Linux의 페이지 할당/해제 핸들러를 수정하여 zeroing된 페이지의 할당 요청이 발생했을 때, 코어 당 zeroed 페이지 리스트로부터 페이지를 제공받도록 한다. 코어 당 zeroed 페이지 리스트가 비어있다면, 수정된 페이지 할당/해제 핸들러는 전역 zeroed 페이지 리스트에서 코어 당 zeroed 페이지 리스트로 일정 수만큼 페이지를 가져온 후, 할당을 시도한다.

3.2. 페이지 zeroing 관리자

페이지 zeroing 관리자는 DMA를 사용하여 사용자 응답시간이 아닌 시점에 페이지의 zeroing을 수행한다. 이를 위해 페이지 zeroing 관리자는 사용자 응답시간 인식기로부터 사용자 응답시간이 시작됨을 전달받으면 동작이 중지되고, 사용자 응답시간이 종료됨을 전달받으면 다시 동작 가능한 상태가 된다.

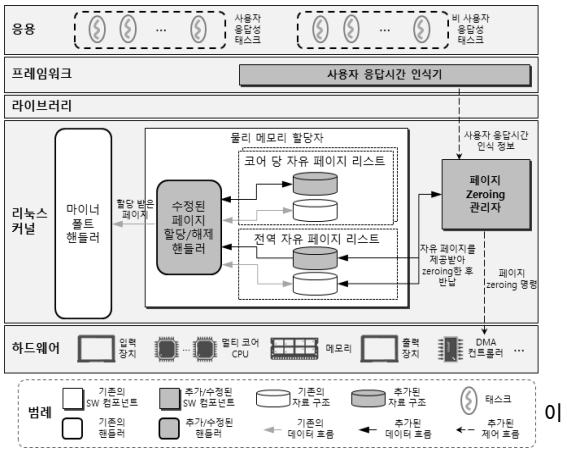

Fig. 1. 제안된 기법 개관

페이지 zeroing 관리자는 동작 가능한 상태가 되면, 전역 자유 페이지 리스트로부터 페이지를 제공받아 zeroing한 다음 전역 zeroed 페이지 리스트에 저장하는 동작을 전역 zeroed 페이지 리스트의 크기가 일정 수준에 도달할 때 까지 수행한다. 이 zeroing 동작은 DMA를 사용하여 디스크로부터 0으로 채워진 영역을 전역 자유 페이지 리스트로부터 제공받은 페이지로 복사함으로써 이루어진다. 이 zeroing 과정은 CPU를 사용하여 페이지를 0으로 채우는 기존 zeroing과 달리 CPU의 캐시 공간을 오염시키지 않는다.

3.3. 사용자 응답시간 인식기

사용자 응답시간 인식기는 사용자에 의하여 입력이 발생한 시점부터, 입력의 결과가 화면상에 출력되는 시점까지의 시간 구간을 인식한다. 이를 위하여 사용자가 발생시킨 마우스/키보드에 의한 입력이 X Window 프레임워크의 사용자 입력을 처리하는 서버인 X Server에 도달 하는 시점을 사용자 응답시간의 시작지점으로 지정한다. 그리고 사용자 입력에 대한 결과로 렌더링된 프레임이 다시 X Server에 전달되는 시점을 사용자 응답시간의 종료지점으로 지정한다. 사용자 응답시간 인식기는 인식한 사용자 응답시간의 시작과 종료를 시스템콜을 통해 페이지 zeroing 관리자에게 알려준다.

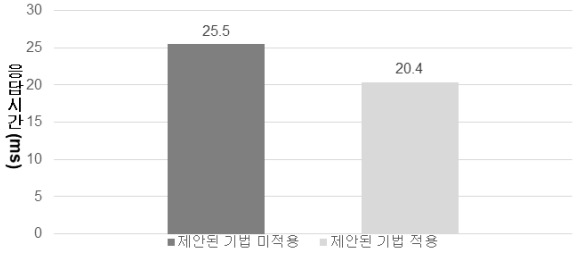

Fig. 2. 응답시간 측정 결과

IV. 실험을 통한 검증

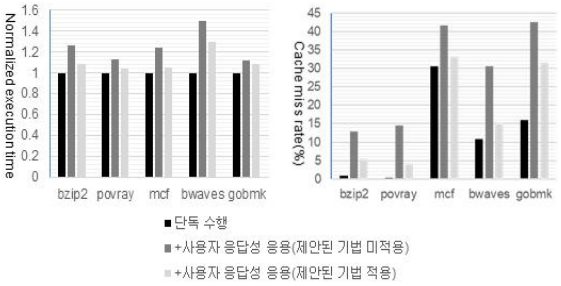

Fig. 3. 정규화된 SPEC2006 실행 결과와 캐시 미스 비율

본 연구진은 제안된 기법을 Table. 1 과 같은 실험 환경에 구현하고 실험들을 통해 검증하였다. 본 장에서는 두 개의 실험 설계를 설명하고 실험 결과를 기술한다. 실험 설계로는 검증하고자 하는 가설, 대조군, 실험군, 측정값, 워크로드, 실험 방법이 있다.

4.1. 제안된 기법이 사용자 응답성 태스크에 미치는 영향

첫 번째 실험의 검증하고자 하는 가설은 제안된 기법을 통해 응답시간이 단축되는지 여부이다. 대조군은 제안된 기법이 적용되지 않은 시스템이며 실험군은 제안된 기법이 적용된 시스템이다. 측정값은 한 응용의 수행 시간을 응답 시간으로 모델링하여 측정하였다. 응용이 수행하는 워크로드는 동적 메모리의 할당, 접근, 간단한 연산, 해제 작업을 반복하여 페이지 할당 요청을 빈번히 발생시킨다. 실험 방법은 모델링된 응용을 10회 수행하여 평균값을 측정하는 것이다. Fig.

2.는 실험 결과를 나타낸다. 제안된 기법을 적용한 시스템에서의 평균 응답 시간이 그렇지 않을 때에 비하여 약 20% 개선되었다.

4.2. 제안된 기법이 백그라운드 응용에 미치는 영향

두 번째 실험의 검증하고자 하는 가설은 제안된 기법을 통해 사용자 응답성 응용와 백그라운드 응용이 함께 수행되는 환경에서 백그라운드 응용의 캐시 미스 비율을 낮추고, 성능 저하를 완화할 수 있는지 여부이다. 대조군과 실험군은 앞선 실험과 같으며 측정값은 백그라운드 응용의 캐시 미스 비율과 수행시간이다. 워크로드는 백그라운드 응용의 경우 SPEC2006 벤치마크, 사용자 응답성 응용은 앞선 실험의 모델링된 응용이다. 실험 방법은 사용자 응답성 응용이 수행중인 상황에서 백그라운드 응용을 수행하여 평균값을 측정하는 것이다. Fig. 3.은 실험 결과를 나타낸다. 제안된 기법을 미적용한 시스템에서는 백그라운드 응용의 수행시간은 백그라운드 응용 혼자 수행될 때에 비하여 최소 1.12배에서 1.5배까지 증가하였으며, 캐시 미스 비율은 최소 1.3배에서 66배까지 증가하였다. 제안된 기법을 적용한 시스템에서는 백그라운드 응용의 수행시간이 혼자 수행될 때에 비하여 최소 1.03배에서 1.29배까지 증가하였으며, 캐시 미스 비율은 최소 1.13배에서 16배까지 증가하였다. 이를 통해 제안된 기법을 적용하였을 때, 사용자 응답성 응용에 의한 백그라운드 응용의 성능 저하를 완화할 수 있음을 보였다.

제안된 기법으로 인한 런타임 오버헤드는 시스템콜을 통하여 사용자 응답시간이 아닌 시간 구간을 페이지 zeroing 관리자에게 알려줄 때 발생하는데, 이 런타임 오버헤드는 매우 미미하여 사용자 응답시간에 거의 영향을 끼치지 않는다.

V. Conclusions

본 논문은 Linux 기반 시스템의 사용자 응답성 향상을 위해 DMA를 사용한 페이지 선 zeroing 기법을 제안하였다. 제안된 기법은 사용자 응답시간이 아닌 시점에 DMA를 통해 캐시 오염 없이 페이지를 zeroing하고, 페이지 할당 요청시 선 zeroing된 페이지를 제공한다. 실험 결과 응답시간이 평균 20%만큼 단축되었다.

ACKNOWLEDGEMENT

"본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 대학ICT연구센터육성지원사업의 연구결과로 수행되었음" (IITP-2018-2013-1-00717)

REFERENCES

[1] Valat, Sébastien, Marc Pérache, and William Jalby. "Introducing kernel-level page reuse for high performance computing." Proceedings of the ACM SIGPLAN Workshop on Memory Systems Performance and Correctness. ACM, 2013.

[2] M. Russinovich and D. A. Solomon. Windows Internals: Including Windows Server 2008 and Windows Vista, Fifth Edition. 2009.

[3] Seokwoo Yang, Jungho Kim. "Adaptive Page Pre-Zeroing Mechanism for Improving User Interactivity in Linux-Based System." KIPS, 2018.

[4] Jeongho Kim, Jonghun Yoo, Sungju Huh and Seongsoo Hong, "Lazy Unmapping of Anonymous Pages for Reducing Page Faults in Linux-based Smartphones," KCC, 2013.