2

스토리지 클래스 메모리 도입에 따른 컴퓨팅 패러다임과 시스템 소프트웨어의 변화

홍익대학교 ▌도인환 · 노삼혁*

1. 서 론

비휘발성 속성과 램의 속성을 동시에 제공하는 스토리지 클래스 메모리(Storage Class Memory; 이후 SCM으로 참조)의 등장은 전통적으로 휘발성 메인 메모리에서의 프로그램 실행을 전제로 하는 기존 컴퓨팅 패러다임에 변화를 가져올 수 있다. 이에 본 논문에서는 SCM의 도입에 따른 컴퓨팅 시스템의 메모리 계층구조의 발전 방향과 해당 시스템 구조에서 SCM을 효과적으로 활용하는 시스템 소프트웨어의 역할에 대해서 살펴보고자 한다. 이를 통해서 SCM의 등장이 기존의 컴퓨팅 패러다임에 어떠한 영향을 초래할 것인지에 대해서 논의하고자 한다.

최근 등장한 SCM 기술은 비휘발성 속성을 제공하면서 동시에 고속의 바이트 단위 랜덤 접근을 지원하는 메모리를 일컫는다[1]. 대표적인 SCM으로는 PCM (또는 PRAM), FeRAM, 그리고 MRAM이 주목받고 있으며, 현재 주요 메모리 제조사에 의해서 주도적으로 연구 개발, 그리고 상용화되고 있다. 특히, 현재의 기술 수준을 감안할 때 2012년부터 SCM이 본격적으로 컴퓨팅 시스템에 보급되기 시작하여 2020년에는 서버 급 시스템 환경에서까지 SCM이 널리 보급될 것이라는 전망이 제기된 바 있다[2,3]. 이처럼 SCM은 머지않아 컴퓨팅 시스템의 대중적인 컴포넌트로써 그 자리매김을 확고히 할 것으로 예상된다.

전통적으로 컴퓨팅 시스템의 동작은 고속의 바이트 단위의 랜덤 접근을 지원하는 휘발성 메인 메모리와 저속의 블록 단위의 접근을 지원하는 비휘발성 저장매체를 포함하는 시스템의 메모리 계층구조를 그 전제로 한다. 이에, 프로그램의 구동을 위해서는 먼저 비휘발성 저장매체에 저장되어 있는 프로그램의 실행 이미지를 메인 메모리로 적재하는 과정이 선행된다. 또한 메인 메모리에서 유지 및 관리되는 프로세스의 정보나 사용자 데이터는 휘발성이므로 필요에 따라 해당 프로세스가 종료되기 전에 비휘발성 저장매체에 기록되는 것이 요구된다. 이러한 전통적인 컴퓨팅 패러다임에서는 다양한 한계점들이 존재한다. 예를 들어, 시스템 또는 프로그램은 구동을 위해서 매번 부팅이나 로딩 과정을 필요로 한다는 제약 사항을 가진다. 뿐만 아니라, 저장장치의 느린 속도와 메인 메모리의 휘발성 속성은 시스템의 성능과 안정성을 저해하는 불가피한 요인으로 작용한다.

고속의 저장매체로써 활용될 수 있는 SCM의 등장은 전통적인 시스템의 메모리 계층구조에서의 시스템 성능 및 안정성과 관련된 한계점들을 단번에 해소할 수 있는 가능성을 열어준다. 먼저, 기존의 메모리 계층구조에서 SCM을 비휘발성 쓰기 캐시나 고속의 저장장치로 활용하는 것이 시스템의 입출력 성능을 향상시킬 수 있음은 다소 직관적으로 이해될 수 있다. 개선된 입출력 성능을 바탕으로, 메인 메모리에서 안전하게 유지되어야 하는 데이터들을 더 자주 저장장치에 기록함으로써 더 높은 수준의 데이터 안정성을 제공하는 것이 가능하다. 이와 같은 SCM의 활용을 통해서 전통적인 시스템 환경에서는 트레이드오프 관계로 인식되어온 시스템의 입출력 성능 향상과 안정성 향상이라는 두 가지 요구사항을 동시에 만족시킬 수 있게 된다[4].

비휘발성 메인 메모리로의 SCM 활용은 시스템의 안정성과 성능 향상의 이슈를 넘어서 시스템의 부팅과 종료에 관한 개념을 변화시킬 수 있다. SCM의 도입은 기존의 파일로 표현되는 데이터의 비휘발성에서 벗어나 프로그램 코드를 포함하는 프로세스 혹은 메모리 주소 공간 전체에 대한 비휘발성을 가능하게 한

* 종신회원

† 이 논문은 2007년도 정부(과학기술부)의 재원으로 한국과학재단의 국가지정연구실사업으로 수행된 연구임(No. R0A-2007-000-20071-0).

다. 따라서 전원의 인가 유무와 상관없이 메인 메모리에 관리되는 전체 시스템의 상태정보는 지속적으로 유지될 수 있다. 이로써 기존의 시스템에서 부팅과 종료라는 개념이 제거될 수 있고, 전원의 인가 유무와 상관없이 영속적으로 수행되는 시스템을 실현하는 것이 가능해진다.

궁극적으로 저장장치와 메인 메모리를 모두 대체하는 용도로 SCM이 도입되는 환경에서는 저장장치와 메인 메모리 간의 경계가 사라지게 되어, 전통적인 컴퓨팅 시스템의 메모리 계층구조와 프로그램의 실행 메커니즘에 혁신적인 변화를 꾀할 수 있다. 이러한 환경에서는 저장매체에서 유지되던 파일의 개념과 메모리에서 유지되던 프로세스의 개념에 대한 구분이 모호해진다. 따라서 프로그램의 실행 시 저장매체로부터 데이터를 읽어서 메모리에 적재하는 과정이나 프로그램의 종료 시 갱신된 데이터를 저장매체에 반영하는 등의 기존의 입출력 과정 자체가 필요 없게 된다. 이러한 환경에서는 더 이상 기존의 컴퓨팅 패러다임을 고수할 필요가 없게 된다.

본 논문에서는 SCM 도입에 따른 시스템 구조와 컴퓨팅 메커니즘의 급격한 변화가 전체 컴퓨팅 패러다임에 변화를 가져오지 않을까 조심스럽게 내다본다. 이에 컴퓨팅 패러다임의 변화 가능성을 살펴보고자 하며, 이후의 논문 구성은 다음과 같다. 먼저 SCM의 특성과 개발 동향에 대해서 살펴본다. 그리고 전통적인 컴퓨팅 패러다임과 해당 컴퓨팅 환경에서 부각되는 한계점들을 살펴본다. 이어서 SCM의 단계적인 도입에 따라서 시스템의 메모리 계층구조가 변화되는 방향을 전망하고, 각각의 컴퓨팅 시스템 구조에서 SCM의 효과적인 활용을 위한 시스템 소프트웨어의 변화와 발전 방향에 대해서 살펴보고자 한다.

2. 스토리지 클래스 메모리

스토리지 클래스 메모리(SCM)는 플래시 메모리처럼 비휘발성 속성을 제공하면서, 동시에 전형적인 램인 DRAM이나 SRAM처럼 고속의 바이트 단위 랜덤 접근을 지원하는 메모리 기술들을 총칭한다[1]. 현재 주요 반도체 제조사들은 SCM에 관련된 연구 개발을 추진하고 있으며, PCM(또는 PRAM), FeRAM, 그리고 MRAM이 대표적인 SCM 기술로써 주목받고 있다. 기존의 전형적인 비휘발성 메모리로는 낸드, 노어 플래시 메모리를 들 수 있지만 이들 플래시 메모리 기술이 바이트 단위의 랜덤 읽기/쓰기를 지원하지 못한다는 특성을 감안한다면 SCM으로 분류되기에는 한계가 있다.

표 1은 플래시 메모리, 그리고 SDRAM과 비교하여 PCM, FeRAM, 그리고 MRAM의 특성을 보여준다. 표 1에서 PCM을 제외한 나머지 메모리 특성들은 모두 특정 제품의 명세서(Data Sheet)를 기반으로 추출된 정보들이다. SCM은 SDRAM에 비해서 다소 느리거나 버금가는 접근 속도를 보이면서도 비휘발성 속성을 제공한다는 점에서 당연 주목할 만하다.

SCM은 플래시 메모리에 비해서 여러 가지 장점이 있다. 첫째, SCM은 SDRAM처럼 바이트 단위의 읽기와 쓰기를 제공한다. 이는 SCM 상에서 프로그램이 직접 수행될 수 있음을 의미한다. 둘째, 접근 속도 측면에서 SCM은 낸드 플래시 메모리를 크게 앞선다. 쓰기 접근 속도 측면에서 SCM 기술은 노어 플래시 메모리와 전혀 다른 특성을 나타냄을 알 수 있다. 셋째, SCM은 플래시 메모리와는 달리 쓰기 성능을 저해하는 소거 연산을 필요로 하지 않는다. 마지막으로, 내구성 측면에서 플래시 메모리는 각 셀에 대해서 반복적인 쓰기 연산에 대하여 현실적인 한계점을 가지는 반면, SCM은 SDRAM 수준의 내구성을 확보함으로써 현실적

표 1 플래시 메모리, SDRAM과 대비되는 SCM의 특성

		플래시 메모리		RAM	Storage Class Memory		
		NAND	NOR	SDRAM	PCM	FeRAM	MRAM
휘발성 여부		비휘발성	비휘발성	휘발성	비휘발성	비휘발성	비휘발성
메모리 접근 단위	읽기/쓰기	페이지	바이트	바이트	바이트	바이트	바이트
	소거	블록	블록	–	–	–	–
메모리 접근 속도	읽기	12us	110ns	50~75ns	DRAM-comparable	110ns	35ns
	쓰기	200us	80us	50~75ns	Not match DRAM	110ns	35ns
	소거	2ms	0.6s	–	–	–	–
내구성 (쓰기 횟수)		10^5	10^5	10^{15}	$< 10^{12}$	10^{12}	10^{15}
집적도 (칩당 용량)		4Mb	512Mb	256Mb	128Mb	4Mb	4Mb

으로 내구성에 대해서 염려할 필요가 없다.

다양한 형태의 메모리에 대한 접근 속도와 집적도를 명세서에 나타나 있는 수치에 근거해서 서로 평가하는 것에 큰 의미를 부여하는 데는 한계가 있다. 이들 메모리의 접근 속도와 집적도 관련 정보는 각 제조사마다 그리고 적용된 기술마다 상이할 수 있기 때문이다. 또한 새롭게 등장한 SCM 기술은 초기 개발 단계인 반면 기존 메모리들은 이미 기술적으로 성숙 단계에 있기 때문이다. 그럼에도 불구하고 표 1에서 고려하는 플래시 메모리와 SDRAM은 최신의 메모리 특성과는 다소 차이가 있음을 밝혀둔다. 표 1에서는 특정 임베디드 시스템 환경에 탑재되어 있는 메모리 기술을 고려하기 때문이다.

현재의 SCM 기술 수준에서 접근 속도는 최신의 SDRAM의 접근 속도와 비교해서 다소 느리며, 낸드 플래시 메모리에 비해서 낮은 집적도를 보인다. 최신의 DDR3 SDRAM 접근 속도는 수 나노 초 범위로 알려져 있으며, 최신의 노어/낸드 플래시 메모리와 SDRAM의 집적도는 각각 512Mb, 128Gb, 8Gb 정도로 알려져 있다. 그렇지만 SCM 기술들의 발전 가능성을 고려해 본다면 SCM은 잠재적으로 SDRAM에 버금가는 성능과 낸드 플래시 메모리를 능가하는 집적도를 보일 것으로 기대된다. 참고로, PCM의 집적도에 대한 확장 가능성은 다른 어떠한 메모리 기술보다 더 좋은 것으로 알려져 있다[3].

현재 PCM, FeRAM, 그리고 MRAM을 비롯한 다양한 SCM 기술은 다수의 주요 반도체 제조사들에 의해서 경쟁적으로 연구 개발되고 있다. PCM은 인텔이나 삼성전자를 비롯한 세계 굴지의 반도체 제조사들에 의해서 개발되고 있으며, 집적도와 확장성 측면에서 가장 유망한 SCM 기술로 주목받고 있다[2]. 삼성전자는 2007년 512Mb 용량의 PCM 프로토타입 개발에 성공한 바 있으며, 휴대전화의 노어 플래시 메모리를 대체하는 용도로의 PCM 상용화 가능성을 시사한 바 있다[5]. 최신 보도는 인텔과 STMicroelectronics사의 합작회사인 Numonyx사가 2009년 현재 128Mb PCM 칩의 양산에 성공하여 상용화를 준비하고 있다고 전한다[6].

FeRAM과 MRAM은 PCM의 상용화에 앞서 이미 최대 4Mb 용량의 칩이 Ramtron사와 Freescale사에 의해서 각각 출시되어 임베디드 시스템 환경에서 주로 활용되고 있다[7,8]. 비교적 낮은 집적도를 보이고 있는 이들 두 SCM에 대해서도 그 접근 속도와 집적도를 향상시키고자 하는 노력들이 지속적으로 이루어지고 있으며, 최근에 FeRAM의 획기적인 집적도 향상 기술의 개발에 관한 보도는 이 같은 사실을 뒷받침해 준다[9].

최근의 몇몇 연구는 2012년부터 SCM이 널리 보급되기 시작하여 2020년에는 높은 집적도와 낮은 가격을 바탕으로 하드디스크를 완전히 대체할 것으로 전망하고 있으며, 또한 SCM의 메모리로써의 활용 가능성 또한 배제하지 않고 있다[2,3]. 이에, 현재의 SCM의 기술 수준과 개발 동향을 고려할 때 SCM은 머지않아 임베디드 시스템 환경은 물론 서버 시스템 환경으로 도입되어 그 활용 영역이 현저히 확대될 수 있을 것으로 전망된다.

3. 전통적인 컴퓨팅 패러다임

전형적인 컴퓨팅 시스템은 CPU, 메인 메모리, 저장장치, 그리고 기타 주변 장치들로 구성되며, 프로그램의 실행을 위해서 이들 구성 요소들이 시스템 소프트웨어에 의해서 유기적으로 관리된다. 그림 1은 프로그램을 수행하는 관점에서 시스템의 메모리 계층구조와 프로그램의 수행 방식을 도식화한다.

프로그램의 실행 이미지는 실행 파일의 형태로 하드디스크와 같은 비휘발성 저장장치에 존재한다. 일반적으로 저장장치들은 바이트 단위 접근을 지원하지 않기 때문에, 비휘발성 저장장치에 존재하는 프로그램 명령어들은 CPU에 의해서 직접 수행될 수 없다. 그러므로 프로그램을 실행하기 위해서는 그림 1(a)처럼 저장장치에서 파일의 형태로 존재하는 프로그램의 실행 이미지를 메인 메모리의 프로세스 주소공간으로 적재하는 과정이 필수적이다.

메인 메모리에서 관리되는 모든 정보는 전원 차단 시 소실되므로 프로그램의 운용에 필수적인 정보들을

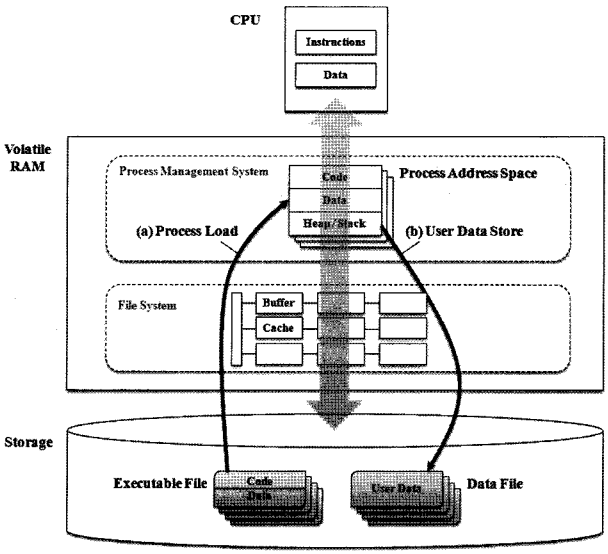


그림 1 전형적인 시스템의 메모리 계층구조와 프로그램 수행 메커니즘

지속적으로 그림 1(b)처럼 비휘발성 저장장치에 기록하는 과정이 요구된다. 이와 같은 메모리 계층구조에서는 메인 메모리와 저장장치 간의 입출력 성능 향상을 목적으로 휘발성 메인 메모리에 파일에 대한 읽기와 쓰기용 버퍼 캐시를 두는 것이 일반적이다.

프로그램을 실행하는 동안 메인 메모리와 저장장치에 중복된 데이터를 유지하는 기존의 시스템은 휘발성 메인 메모리에서 변경된 정보를 비휘발성 저장장치에 반영하는 과정을 수반하기 때문에 시스템의 안정성과 수행성능 측면에서 한계점이 존재한다. 기존의 시스템에서는 프로그램을 수행하는 동안 프로세스 주소공간에 유지되는 다양한 정보 중에서 사용자 데이터와 같이 소실되면 재생할 수 없는 정보들을 저장매체에 기록해야 한다.

저장매체와 메모리 간의 처리량 차이를 고려할 때, 메인 메모리와 저장매체에 중복해서 존재하는 데이터를 매번 동기화하는 것은 시스템 수행 성능에 심각한 저하를 초래한다. 이를 해결하기 위한 수단으로써 고속의 휘발성 메모리에서 파일에 대한 읽기와 쓰기를 캐시하는 기법이 널리 사용된다. 이는 시스템의 수행성능을 높이기 위해서 데이터에 대한 안정성을 희생하는 방법이라 할 수 있다. 이와 같이 기존의 컴퓨팅 패러다임에서는 시스템의 수행 성능과 안정성 사이에 트레이드오프 관계가 형성된다.

기존의 시스템에서는 시스템의 부팅 시에 매번 저장매체에 저장되어 있는 정보를 바탕으로 메인 메모리의 프로세스 주소공간을 다시 구성해야 한다. 시스템이 부팅되는 과정은 다음과 같다. 먼저 하드웨어를 점검하고 운영체제를 메모리에 적재한 후에 운영체제 코드를 실행한다. 이어서 메인 메모리와 프로세스, 그리고 파일을 관리하는 하위 시스템들을 구동한다. 최종적으로 다양한 시작 프로그램들을 실행함으로써 완료한다. 메인 메모리가 휘발성인 시스템의 구동을 위해서 필수적인 이 같은 부팅 과정은 필연적으로 오랜 시간 지연을 초래한다는 문제점을 가진다.

기존의 시스템은 부팅 직후에 시스템이 종료되기 직전의 실행 상태를 되돌려주지 않고 항상 특정 초기화된 실행 상태를 제공한다. 즉, 종료 전의 시스템과 재시작 이후의 시스템은 서로 다른 시스템 상태를 가지므로 영속적으로 동작하지 않는다. 시스템의 긴 부팅 과정과 더불어 이러한 비영속적인 컴퓨팅 환경은 사용자로 하여금 컴퓨팅이 불필요한 시점에서 조차도 시스템이나 프로그램을 종료하는 것을 주저하게 만들며, 이러한 한계 상황은 휘발성 메인 메모리를 사용하는 기존 컴퓨팅 패러다임에서는 해결하기 힘든 한계점으로 존재한다.

4. SCM 도입에 따른 시스템 소프트웨어의 발전 방향

현재 SCM 기술은 FeRAM과 MRAM의 형태로 임베디드 시스템 환경에서 제한적으로 도입되고 있는 실정이지만, 접근 속도와 확장성 측면에서 SCM 기술이 점진적으로 성장함에 따라서 컴퓨팅 시스템의 중요한 구성요소로써 자리매김할 것으로 생각된다. 이에, 본 논문에서는 그림 2와 같이 SCM이 단계적으로 도입될 것으로 전망한다.

먼저, 메인 메모리로써 DRAM과 저장매체로써 하드디스크 혹은 플래시 메모리가 널리 사용되고 있는 현재의 메모리 계층구조에 SCM이 빠른 저장매체로써 추가되는 환경을 고려한다. 이는 기존 시스템 구조와의 호환성과 높은 비용과 낮은 집적도를 보이는 현재의 SCM 기술 수준을 고려할 때 매우 현실적이다. 두 번

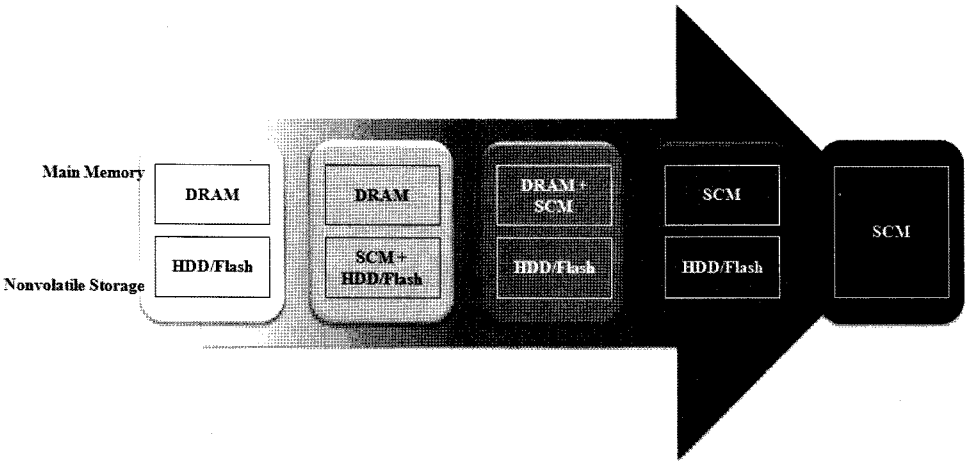


그림 2 SCM의 단계적 도입에 따른 컴퓨팅 시스템의 메모리 계층구조 변화 전망

째, SCM이 메인 메모리의 일부로써 추가될 수 있다. 이러한 도입은 SCM이 보유한 램의 속성과 비휘발성 속성을 모두 활용한다는 측면에서 바람직하다. 세 번째, SCM이 메인 메모리로써 DRAM을 모두 대체하는 시스템 환경을 고려한다. 이는 SCM의 기술 수준이 접근 속도와 처리량, 그리고 집적도 측면에서 DRAM과 견줄 수 있게 될 때 가능할 것이다. 마지막으로, SCM의 집적도가 저장매체 수준으로 충분히 향상된다면 SCM이 메인 메모리와 저장매체를 대체한 시스템이 등장할 수 있을 것으로 기대된다.

본 절에서는 SCM이 단계적으로 도입된 각각의 시스템 환경에서 시스템 소프트웨어 측면에서의 SCM 활용 방안과 SCM 활용을 통해서 3절에서 언급한 기존 시스템 환경에서의 한계점들이 어떻게 극복될 수 있는지에 대해서 살펴본다.

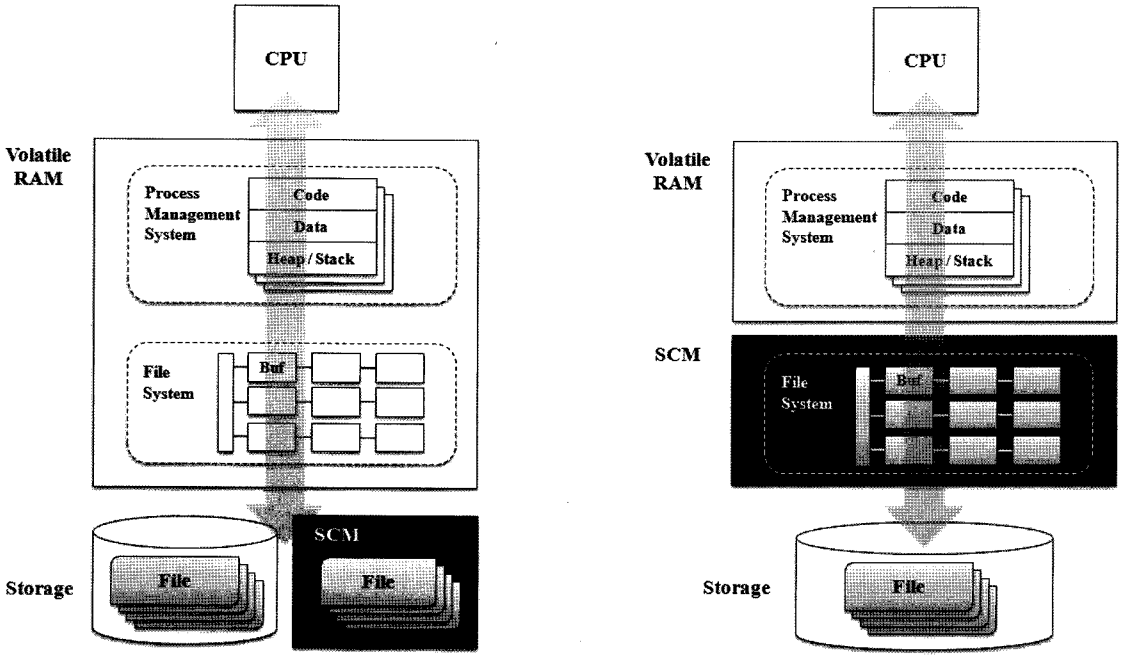
4.1 빠른 저장매체로의 SCM 활용

비휘발성 속성을 제공하면서도 메모리 수준의 빠른 접근 속도를 지원하는 SCM의 등장은 시스템의 수행 성능과 안정성 향상에 크게 기여할 수 있다. 전통적으로 저장매체의 느린 읽기/쓰기 접근 속도는 컴퓨팅 시스템의 수행 성능을 결정짓는 병목으로 간주된다. 그림 3은 이러한 병목현상을 해결하고자 기존의 메모리 입출력 계층구조에서 SCM을 활용하는 방안을 보여준다. 그림 3(a)는 기존 저장매체의 확장된 공간으로써 빠른 저장매체인 SCM을 활용하는 모습을, 그림 3(b)는 파일시스템의 쓰기 버퍼 캐시의 용도로 SCM을 활용하는 모습을 보여준다. 현재 고비용과 낮은 집적도를 보이는 SCM 기술 수준을 고려할 때, 소량의 SCM을 이와 같이 제한적인 용도로 활용하는 것은 현실성 있는 접근이다.

제한적인 용량의 SCM을 빠른 저장매체로써 효과적으로 활용하려는 다양한 시스템 소프트웨어 기법들이 기존한다. 먼저 그림 3(a)처럼 추가적인 저장 공간으로 SCM을 활용하는 기존 연구들에 대해서 살펴본다. 이어서 그림 3(b)에 나타난 바와 같이 SCM을 쓰기 버퍼 캐시로 고려한 기존 연구들에 대해서 논의한다.

파일시스템의 관점에서 보면 높은 빈도로 수정되면서 동시에 작은 공간을 차지하며, 다른 데이터들보다 안정성 측면에서 더 중요한 데이터를 빠른 저장매체에 유지하는 것이 성능 향상에 도움이 된다. 따라서 기존의 연구들은 파일시스템의 메타데이터를 SCM에 유지함으로써 적은 양의 SCM으로도 시스템의 입출력 성능과 파일시스템의 안정성을 동시에 향상시키는 것을 가능하게 한다.

구체적으로, Miller 등은 메타데이터를 MRAM에 유지하고 파일 데이터를 저장매체에 보관하는 HeRMES 파일시스템을 제안한 바 있다[10]. 그리고 MRAMFS는 MRAM 사용을 절약하기 위해서 파일시스템 메타데이터를 압축하여 MRAM에 유지하는 방식을 취한다[11]. 이 두 파일시스템은 전통적인 디스크 기반의 파일시스템으로써 범용 시스템 환경을 대상으로 한다.



(a) 기존 저장매체의 확장 공간 (b) 쓰기 버퍼 캐시 공간

그림 3 빠른 저장매체로써 SCM이 도입된 시스템의 메모리 계층구조

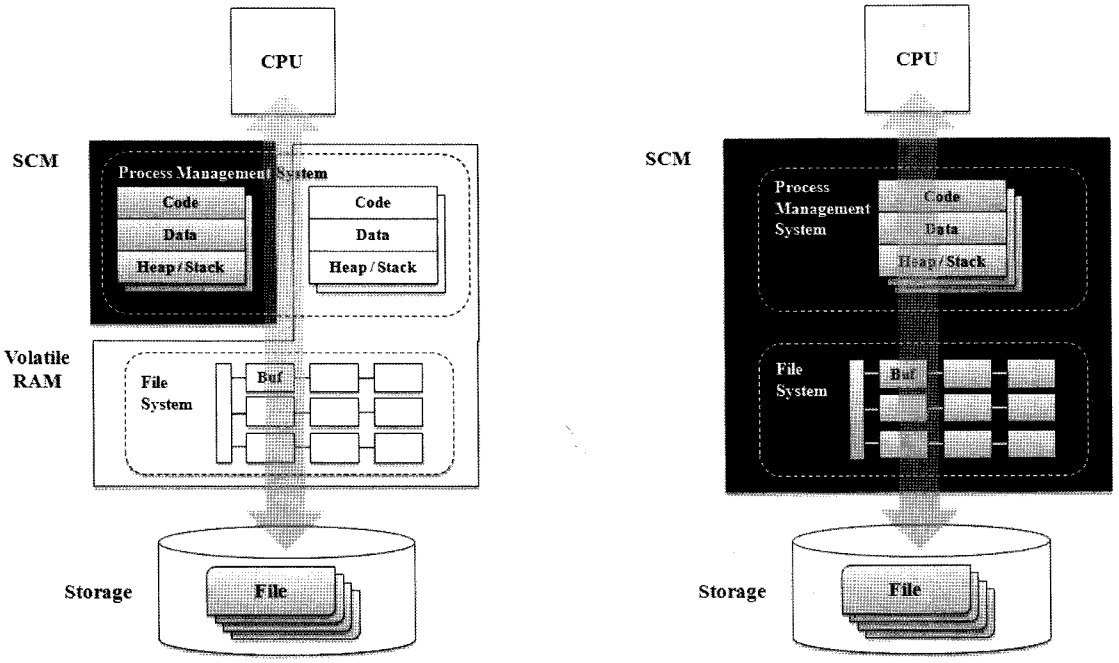
최근에는 플래시 메모리를 주 저장매체로 하는 임베디드 시스템 환경에서 SCM을 활용함으로써 플래시 메모리에 대한 입출력 속도와 안정성을 향상시키고자 하는 연구들이 다수 등장한다. 먼저, 파일시스템의 모든 메타데이터를 SCM에서 항상 마운트 된 형태로 관리하며 파일 데이터만 낸드 플래시 메모리에 저장하는 MiNV 파일시스템이 제안된 바 있다[4]. 이와 유사한 접근방식을 취하는 PFFS 파일시스템은 PCM의 활용에 초점을 두며, PCM의 내구성 향상을 지원하는 활용 방안을 제안한다[12].

플래시 메모리 기반 SSD (Solid State Disk)에서 플래시 변환 계층(FTL)의 메타데이터를 효율적으로 관리하려는 목적으로 FeRAM을 활용하는 고성능 SSD 구조를 제안하는 연구가 존재하며, 해당 연구는 소량의 FeRAM의 활용이 SSD의 성능 향상에 기여함을 보여준다[13]. 앞서 언급한 연구들과 비슷하게, 임베디드 시스템 환경에서 파일시스템의 메타데이터와 플래시 변환 계층(FTL)에서 관리되는 메타데이터들을 PCM에 저장함으로써 안정성에 대한 우려 없이 성능을 향상시키고자 하는 기법이 제안된 바 있다[14].

기존의 시스템에서는 휘발성 메모리를 쓰기 캐시로 사용하기 때문에 예상치 못한 시스템 붕괴 시 쓰기 캐시된 데이터에 대한 무결성을 보장하지 못한다는 안정성 측면에서 약점을 가진다. 이러한 약점을 보완하기 위해서 캐시된 데이터를 높은 빈도로 저장매체에 동기화하는 것은 시스템 성능의 저하를 초래한다. 이와 같이 트레이드오프 관계에 있는 입출력 성능과 안정성을 동시에 향상시키고자 SCM을 파일 캐시로 도입한 연구들이 몇몇 존재한다.

먼저, SCM을 쓰기 캐시로 도입할 경우에 캐시된 데이터의 안정성이 보장되어 저장매체로의 동기화 시점이 자유로워짐에 따라서 기존의 최적 캐시 교체 정책으로 알려진 MIN 알고리즘을 능가하는 캐시 교체 정책의 개발이 가능함을 밝힌 연구가 기존한다[15]. 임베디드 시스템 환경에서 FAT 파일시스템의 메타데이터를 SCM에 캐시함으로써 입출력 성능 및 안정성 향상은 물론 저장매체로 사용된 플래시 메모리의 내구성과 에너지 효율의 향상이 가능함을 제시한 연구가 존재한다[16]. 마지막으로 SCM을 쓰기 캐시로 도입하고 트랜잭션 단위의 쓰기를 지원함으로써 이 SCM 캐시를 활용하는 파일시스템은 기존의 EXT3와 같은 저널링 파일시스템의 안정성을 가짐과 동시에 그 수행 성능 또한 향상될 수 있음을 실험적으로 확인한 연구도 존재한다[17].

기존 연구들을 통해서, 시스템 입출력 성능과 안정성과 관련된 전통적인 컴퓨팅 패러다임에서의 한계점이 SCM을 저장장치의 일부 혹은 쓰기 캐시로 활용함으로써 극복될 수 있음을 확인하였다. 그럼에도 불구하고, 단순히 빠른 저장매체로의 SCM 활용은 SCM의 램 속성을 충분히 활용하지 못한다는 비효율적인 측면을 포함한다. 따라서 이후 본 논문에서는 SCM이 제공하는 램 속성과 비휘발성 속성의 활용을 극대화하



(a) 메인 메모리의 일부분으로 활용 (b) 전체 메인 메모리를 대체

그림 4 메인 메모리로써 SCM이 도입된 시스템 메모리 계층구조

는 시스템 소프트웨어 기법들에 대해서 중점적으로 논의하고자 한다.

4.2 비휘발성 메인 메모리로의 SCM 활용

전통적으로 시스템의 메모리 계층구조와 프로그램의 실행 및 시스템 구동은 메인 메모리가 휘발성이라는 것을 전제로 하고 있다. 이러한 전제가 바뀐다면 기존의 컴퓨팅 환경에 두드러진 변화를 초래할 수 있을 것이다. SCM이 메인 메모리의 일부 혹은 전체로 도입되어 메인 메모리의 내용이 전원이 차단된 이후에도 유지된다면, 기존에는 불가능했던 어떤 새로운 개념 혹은 특성이 제공될 수 있지는 않을까? 본 절에서는 이 물음에 관해서 우리가 수행하고 있는 시스템 소프트웨어적인 연구들에 초점을 맞추어 논의를 진행하고자 한다.

그림 4는 SCM이 메인 메모리의 일부 혹은 전체를 대체할 경우에 가능한 메모리 계층구조의 모습을 보여준다. 그림 4(a)는 SCM이 메인 메모리의 일부분으로 도입되며, 특정 프로세스의 전체 주소 공간을 유지하는 용도로 SCM을 활용하는 모습을 보여준다. 이 경우에, SCM에서 실행되는 프로세스는 모든 프로세스 주소공간이 비휘발성이므로 전원 차단 이후에도 지속적으로 해당 프로세스 주소공간을 유지할 수 있다. 따라서 해당 프로세스는 전원 차단 시에도 종료될 필요가 없다. 시스템이 종료된 상황이라고 하더라도 해당 프로그램은 종료된 것이 아니라 중단되어 있다고 말할 수 있다. 이처럼 프로그램을 SCM 상에서 실행함으로써 영원히 지속적으로(영속적으로) 해당 프로그램을 수행하는 것이 가능해진다.

SCM을 활용하여 프로세스에 영속성을 부여하기 위해서는 SCM 메모리와 SCM에서 수행되는 프로세스를 영속적으로 관리하는 기능이 운영체제에 새롭게 포함되어야 한다. SCM 메모리 관리자는 리눅스의 버디/슬랩 메모리 관리자처럼 SCM의 메모리 공간을 동적으로 할당하고 해제하는 기능을 제공하며, 시스템의 재부팅 이후에도 종전의 메모리 할당 정보를 유지할 수 있도록 동작한다. 또한 기존의 DRAM 메모리 관리자와 호환 가능한 인터페이스를 제공해야 한다. 이를 통해서 프로세스 주소공간이 SCM에 할당되든지 혹은 DRAM에 할당되든지에 상관없이 기존 운영체제의 프로세스 실행 메커니즘에 의해서 해당 프로세스가 수행될 수 있다.

SCM 영속적 프로세스 관리자는 재부팅 과정에서 DRAM 초기화로 인하여 일관성이 어긋난 SCM과 DRAM의 프로세스 실행 정보를 동기화함으로써 SCM에 상주하는 프로세스가 영속적으로 수행될 수 있도록 한다. 시스템의 종료와 재부팅 과정은 DRAM에 존재하는 모든 프로세스 관련 정보를 초기화한다. 반면, SCM에서 관리되는 프로세스 관련 정보는 종료 직전의 상태로 유지된다. 다시 말해서 DRAM과 SCM에 각각 존재하는 프로세스 관련 정보들 사이에 일관성이 어긋나게 된다. 그러므로 SCM에서 실행되던 프로세스는 바로 수행될 수 없다. SCM 영속적 프로세스 관리자는 프로세스 관리를 위한 커널의 자료구조를 SCM에 상주하는 프로세스를 인식하도록 조정하는 작업을 수행함으로써 SCM과 DRAM에서 관리되는 프로세스 정보들을 동기화할 수 있다. 이를 통해서 SCM에 상주하는 프로세스가 시스템의 재시작 여부와 상관없이 영속적으로 동작할 수 있게 된다.

SCM 활용을 통한 프로세스의 영속성 보장은 특정 프로세스의 안정성을 극대화하는 용도로 활용될 수 있다. 데이터베이스 시스템에서 사용되는 트랜잭션의 개념을 도입하여 SCM에 유지되는 프로세스 주소공간의 일련의 정보들을 원자적으로 갱신될 수 있다면, SCM에서 실행되는 프로세스의 영속성은 갑작스런 시스템 붕괴 상황에서도 보장될 수 있다. 따라서 SCM에서 수행되는 프로세스는 그 영속성으로 인해서 시스템 붕괴 시에도 극도의 안정성을 보장받을 수 있게 된다.

본 연구팀에서는 프로세스 단위의 영속성을 보장하는 기법의 실현 가능성을 실제 구현을 통해서 살펴보았다. 이를 위해 SCM 메모리 관리자와 SCM 영속적 프로세스 관리자의 기능을 제공하도록 리눅스 운영체제를 수정하였다. 그리고 FeRAM이 탑재된 실제 임베디드 시스템 환경에서 리눅스 운영체제와 지정된 일부 사용자 프로세스들을 FeRAM에서 실행하고 나머지 프로세스들은 램에서 수행하였다. 기본적인 테스트 결과, SCM에서 수행되는 프로세스들에 대해서 영속성을 보장하는 것이 실현 가능함을 확인한 바 있다. 시스템 붕괴 시의 프로세스 영속성 보장에 대한 실현 가능성 확인은 차후 연구 과제로 남겨 둔다.

전체 메인 메모리를 SCM으로 대체한 경우의 시스템 메모리 계층구조는 그림 4(b)와 같이 표현된다. 이 경우에, 쓰기 버퍼 캐시가 비휘발성으로 전환되므로 4.1절에서 언급한 비휘발성 쓰기 캐시의 도입에 따른 안정성과 수행 성능 향상이라는 이득을 자연스럽게 얻을 수 있다. 더불어 메인 메모리에서 실행되는 모든 프로세스의 주소공간이 시스템의 종료 혹은 전원의 인가 여부와 상관없이 항상 유지될 수 있으므로, 프로세스에 대한 영속성을 넘어서 시스템 차원에서의 영속성을 제공할 수 있다. 프로세스가 항상 실행 상태

로 비휘발성 SCM 메인 메모리에 존재하므로 전원의 중단과 재공급 시에 시스템의 종료와 부팅 과정을 생략하고 시스템에 전원이 인가되면 즉각적으로 전원 중단 직전의 실행 상태로 복귀해서 영속적으로 시스템을 유지하는 것이 가능해진다.

즉각적인 부팅과 영속적 컴퓨팅은 SCM의 도입만으로 가능한 것이 아니라 당연히 시스템 소프트웨어의 지원을 필요로 한다. 영속적 컴퓨팅과 즉각적인 부팅을 위해서는 전원 차단 직전의 시스템 상태 정보를 모두 비휘발성으로 전환하는 것이 요구된다. SCM이 메인 메모리로 사용됨으로써 메인 메모리의 모든 정보는 비휘발성으로 전환된다. 이제 CPU의 레지스터 정보와 주변 장치들의 버퍼 정보만 비휘발성으로 전환된다면 시스템의 모든 상태 정보는 전원의 공급 여부와 상관없이 항상 유지된다. 이를 위해서 여전히 휘발성인 시스템 정보들을 전원 차단 직전에 SCM에 기록하고, 전원이 재공급되면 해당 정보들을 SCM으로부터 읽어서 CPU 레지스터와 주변장치의 버퍼 내용을 복원한다. 결국, SCM 메인 메모리와 시스템 소프트웨어의 지원으로 즉각적인 부팅 및 영속적 컴퓨팅이 가능해 진다.

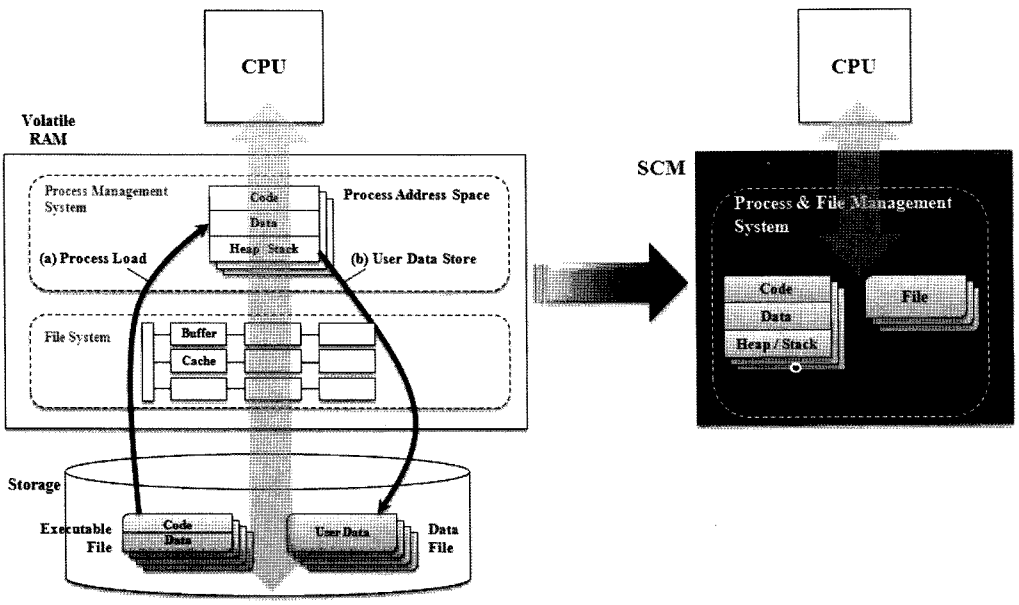
본 연구팀에서는 FeRAM이 장착된 실제 임베디드 시스템 환경에서 이와 같은 영속적 컴퓨팅 및 즉각적인 부팅을 제공하는 프로토타입 시스템을 개발함으로써 실현 가능성을 확인한 바 있다[18]. 프로토타입 시스템 구현을 위해서 시스템의 부트로더와 리눅스 2.6.21 운영체제를 수정하였다. 성능 평가 결과, 프로토타입 시스템은 전원 인가 시 2.5ms 이내에 종전의 영속적인 시스템 상태로 즉시 복귀할 수 있음을 확인하였다.

이와 같이, 메인 메모리로의 SCM 도입과 시스템 소프트웨어에서의 효과적인 활용은 영속적 컴퓨팅과 즉각적인 시스템 구동이라는 새롭고 매력적인 컴퓨팅 개념을 제공할 수 있다. 뿐만 아니라 기존의 컴퓨팅 패러다임에서 문제되어온 시스템의 안정성 향상에 대한 요구를 영속적 컴퓨팅을 통해서 완화시킬 수 있다.

4.3 저장매체와 메인 메모리를 통합하는 용도로의 SCM 활용

기존의 시스템을 포함해서 지금까지 논의된 SCM을 활용하는 시스템에서는 소량의 휘발성 혹은 비휘발성 메인 메모리와 대용량의 비휘발성 저장장치를 활용하여 마치 대용량의 비휘발성 메인 메모리가 존재하는 것과 같은 환상을 제공하고자 한다. 이를 위해서 기존의 시스템 소프트웨어는 프로세스를 관리하는 시스템과 파일을 관리하는 시스템을 필요로 하며 이들 두 시스템 간의 복잡하고 유기적인 상호작용을 요구한다. 기존의 메인 메모리와 저장장치가 모두 SCM으로 대체되어 기존의 시스템에서 제공하고자 하는 대용량의 비휘발성 메인 메모리가 SCM의 기술 발전으로 현실화 된다면, 기존의 복잡한 컴퓨팅 패러다임에 현격한 변화를 예상할 수 있다.

그림 5(b)는 SCM이 대용량의 비휘발성 메인 메모리로써 기존의 메인 메모리와 저장장치를 모두 대체하는 경우에 시스템 메모리 계층구조의 변화를 보여준다. 그림 5(a)에서 보여주는 기존의 메모리 계층구조와는



(a) 기존의 메모리 계층구조(그림 1과 동일) (b) 새로운 메모리 계층구조

그림 5 메인 메모리와 저장장치가 SCM으로 통합됨에 따른 메모리 계층구조의 단순화

달리 더 이상 휘발성 메인 메모리와 비휘발성 저장장치라는 이원적인 메모리 계층구조는 존재하지 않는다. 이와 같은 혁신적인 메모리 계층구조의 변화는 자연스럽게 기존 시스템 소프트웨어에 대한 변혁을 요구할 것으로 생각된다. 과연 시스템 소프트웨어는 어떻게 변화되는 것이 바람직한가? 이 고민에 대한 결론은 컴퓨팅 패러다임의 변화로 귀결된다.

메인 메모리와 저장장치가 모두 SCM으로 대체된 시스템에서는 파일과 프로세스의 주소공간이라는 개념의 구분이 필요치 않다. 실행 파일이 곧 프로세스의 실행 이미지로써 CPU에 의해서 바로 실행될 수 있으며, 프로세스의 주소공간에 저장되어 있는 사용자 데이터는 그 자체로 데이터 파일이 될 수 있다. 따라서, 그림 5(b)에는 그림 5(a)에 나타나 있는 프로세스 적재(Process Load) 과정과 사용자 데이터 저장(User Data Store) 과정을 찾아볼 수 없다. 또한 저장장치에 대해서 데이터를 읽거나 쓰는 연산이 제거되므로, 기존의 캐시는 존재의 이유를 잃게 된다.

SCM이 메인 메모리와 저장장치를 통합한다는 것은 다시 말해서 시스템 소프트웨어가 SCM을 활용하여 프로세스와 파일의 통합적 관리를 지원한다는 것을 의미한다. 이에, 기존 운영체제에 대한 수정이 아니라 완전히 다른 방식으로 동작하는 운영체제가 필요해진다. 본 논문에서는 파일과 프로세스를 통합적으로 관리하는 새로운 운영체제에 대해서 다음과 같은 요구사항을 설정한다. 첫째, 새로운 운영체제는 프로세스의 주소공간에 유지되는 사용자 데이터를 파일의 형식으로 보여줄 수 있는 메커니즘을 제공한다. 둘째, 실행 파일을 프로세스의 실행 이미지로 인식하여 실행 파일에서 바로 실행할 수 있는 메커니즘을 제공한다. 이와 같은 파일과 프로세스의 통합적 관리는 기존 시스템과 대비해서 저장장치와 메모리 간의 복사 요청을 제거하고 메모리와 메모리 사이의 복사 요청을 최소화함으로써, 프로세스의 구동과 파일의 생성 시에 시스템의 성능을 극단적으로 향상시킬 수 있다.

본 연구팀에서는 파일과 프로세스를 통합적으로 관리하는 새로운 운영체제 개발을 진행하고 있다. 초기의 결과로써 위의 첫 번째 요구사항을 만족하는 운영체제를 개발하여 FeRAM이 탑재된 임베디드 시스템 환경에서 운용 가능성을 확인하였다[19]. 기초적인 성능 평가 결과, 기존의 대표적인 실시간 uC/OS-II 운영체제와 비교해서 파일에 대한 생성/읽기/쓰기와 같은 파일시스템 관련 연산의 수행 속도가 수배에서 최대 수백 배 향상됨을 확인한 바 있다.

지금까지 살펴본 바와 같이, SCM의 점진적인 도입으로 인하여 변화되는 시스템 메모리 계층구조와 해당 시스템 구조에서 SCM을 효과적으로 활용하기 위한 시스템 소프트웨어의 변화와 발전은 획기적인 수준의 시스템 수행 속도 향상과 안정성 향상을 가능하게 한다. 그 뿐만 아니라 기존의 시스템 환경에서는 지원되지 않는 영속적 컴퓨팅과 즉각적인 시스템 구동이라는 새로운 형태의 컴퓨팅 개념을 제공할 수 있다. 더 나아가, 메인 메모리와 저장장치를 통합하는 형태로의 SCM 도입은 기존의 파일과 프로세스 개념 사이의 경계를 무너뜨림으로써 새로운 형태의 컴퓨팅 패러다임에 대한 등장을 예견한다.

5. 결 론

스토리지 클래스 메모리(SCM)는 비휘발성 속성을 제공하는 동시에 전형적인 램처럼 고속의 바이트 단위 랜덤 접근을 지원하는 메모리이다. 현재 PCM, FeRAM, 그리고 MRAM으로 대표되는 SCM의 기술 수준과 개발 동향을 고려할 때 SCM은 머지않아 일상적인 시스템 구성 요소로써 활용될 것으로 예상된다. 이러한 SCM의 등장은 시스템의 메모리 계층구조와 시스템 소프트웨어의 변화를 촉진할 것으로 판단된다. 이에 본 연구에서는 SCM의 단계적인 도입에 따른 시스템 구조의 변화 모습과 SCM을 효과적으로 활용하기 위한 시스템 소프트웨어의 발전 방향에 대해서 살펴보았다. 더불어 SCM 기술의 급격한 성장과 시스템 메모리 계층구조로의 혁신적인 도입에 따른 컴퓨팅 패러다임의 변화 가능성에 대해서 논의하였다.

본 논문에서 논의된 내용을 정리하면 다음과 같다. 첫째, 기존에 SCM을 빠른 저장매체로 활용한 연구들을 통해서 소량의 SCM을 저장장치의 일부 혹은 쓰기 캐시로 활용함으로써 시스템의 입출력 성능과 안정성 향상에 크게 기여할 수 있음을 확인하였다. 둘째, SCM이 메인 메모리의 일부 혹은 전체로써 활용되어 메인 메모리의 내용들이 전원이 차단된 이후에도 유지된다면, SCM에서 수행되는 프로세스는 시스템의 종료 여부와 상관없이 영속적으로 수행되어 극도의 안정성을 제공할 수 있다. 더 나아가, 모든 프로세스의 주소공간이 시스템의 종료 혹은 전원의 인가 여부와 상관없이 항상 유지될 수 있다면, 프로세스에 대한 영속성을 넘어서 시스템 차원에서의 영속성을 제공할 수 있다. 마지막으로, 기존의 메인 메모리와 저장장치가 모두 SCM으로 대체된다면, 저장장치와 메인 메모리의 간의 경계가 사라지게 됨으로써 새로운 형태의 프

로세스와 파일 관리 메커니즘이 요구된다. 이와 같이, 시스템 메모리 계층구조로의 SCM 도입과 이를 적극적으로 활용하려는 시스템 소프트웨어의 변화는 컴퓨팅 패러다임에 변화를 이끌어 낼 것으로 예상된다.

참고 문헌

[1] G. W. Burr, B. N. Kurdi, J. C. Scott, C. H. Lam, K. Gopalakrishnan, and R. S. Shenoy, "Overview of candidate device technologies for storage-class memory," IBM Journal of Research and Development, vol. 52, no. 4/5, 2008.

[2] R. F. Freitas and W. W. Wilcke, "Storage-class memory: The next storage system technology," IBM Journal of Research and Development, vol. 52, no. 4/5, 2008.

[3] R. Freitas, W. Wilcke, B. Kurdi, G. Burr, "Storage Class Memories," Tutorial at the 7th USENIX Conference on File and Storage Technologies (FAST'09), 2009.

[4] I. H. Doh, J. Choi, D. Lee, and S. H. Noh, "Exploiting Non-Volatile RAM to Enhance Flash File System Performance," In Proceedings of the 7th ACM & IEEE International Conference on Embedded Software (EMSOFT'07), 2007.

[5] Technology Review, http://www.technologyreview.com/Infotech/20148.

[6] Numonyx Memory Solutions, http://www.numonyx.com.

[7] Ramtron International, http://www.ramtron.com.

[8] Freescale Semiconductor, http://www.freescale.com.

[9] Highest Density FeRAM, http://my.opera.com/kaycee/blog/show.dml/2988141.

[10] E. L. Miller, S. A. Brandt, and D. D. E. Long, "HeRMES: High-Performance Reliable MRAM-Enabled Storage," In Proceedings of the 8th Workshop on Hot Topics in Operating Systems (HOTOS-VIII), 2001.

[11] N. K. Edel, D. Tuteja, E. L. Miller, and S. A. Brandt, "MRAMFS: A Compressing File System for Non-Volatile RAM," In Proceedings of the 12th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'04), 2004.

[12] Y. Park, S.-H. Lim, C. Lee, and K. H. Park, "PFFS: A Scalable Flash Memory File System for the Hybrid Architecture of Phasechange RAM and NAND Flash," In Proceedings of the 23th ACM Symposium on Applied Computing (SAC'08), 2008.

[13] J. H. Yoon, E. H. Nam, Y. J. Seong, H. Kim, B. Kim, S. L. Min, and Y. Cho, "Chameleon: A High Performance Flash/FRAM Hybrid Solid State Disk Architecture," IEEE Computer Architecture Letters, vol. 7, no. 1, 2008.

[14] J. K. Kim, H. G. Lee, S. Choi, and K. I. Bahng, "A PRAM and NAND Flash Hybrid Architecture for High-Performance Embedded Storage Subsystems," In Proceedings of the 8th ACM International Conference on Embedded Software (EMSOFT'08), 2008.

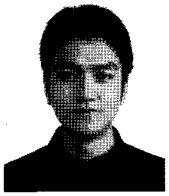
[15] K. H. Lee, I. H. Doh, J. Choi, D. Lee, and S. H. Noh, "Write-Aware Buffer Cache Management Scheme for Nonvolatile RAM," In Proceedings of the 3rd Conference on IASTED International Conference (ACST'07), 2007.

[16] I. H. Doh, H. J. Lee, Y. J. Moon, E. Kim, J. Choi, D. Lee, and S. H. Noh, "Impact of NVRAM Write Cache for File System Metadata on I/O Performance in Embedded Systems," In Proceedings of the 24th ACM Symposium on Applied Computing (SAC'09), 2009.

[17] Y. J. Kim, I. H. Doh, E. Kim, D. Lee, and S. H. Noh, "Impact of Exploiting Storage Class Memory as a Write Buffer Cache on the Performance and Reliability of File Systems," In Proceedings of the International Workshop on Operating System Support for Next Generation Large Scale NVRAM (NVRAMOS'09 Spring), 2009.

[18] I. H. Doh, Y. J. Moon, J. S. Park, E. Kim, J. Choi, D. Lee, and S. H. Noh, "In Search of Alternative Uses of Byte-Addressable Non-Volatile RAM: A Case Study of a Green Web Server Cluster," In Proceedings of the 1st Workshop on Integrating Solid-state Memory into the Storage Hierarchy (WISH'09), co-located with ASPLOS'09, 2009.

[19] S. Baek, K. Sun, J. Choi, E. Kim, D. Lee, and S. H. Noh, "Implementation Study of an Unified SCRAM Manager for Storage Class Random Access Memory," In Proceedings of the 3rd International Workshop on Software Support for Portable Storage (IWSSPS'08), co-located with EMSOFT'08, 2008.



도 인 환

2003 홍익대학교 컴퓨터공학 학사
2006 홍익대학교 컴퓨터공학과 석사
2006~현재 홍익대학교 컴퓨터공학과 박사과정
관심분야 : 운영체제, 내장형시스템, 차세대 비휘발성 메모리, 그린컴퓨팅
E-mail : ihdoh@necsst.ce.hongik.ac.kr

노 삼 혁

1986 서울대학교 컴퓨터공학과 학사
1993 메릴랜드대학교 컴퓨터과학과 박사
1993~1994 조지워싱턴대학교 객원 조교수
1994~현재 홍익대학교 정보컴퓨터공학부 교수
관심분야 : 운영체제, 플래시메모리소프트웨어, 내장형시스템, 차세대저장장치
E-mail : samhnoh@hongik.ac.kr