임베디드 시스템 구조와 운영체제에 대한 스토리지 클래스 메모리의 효과[1)]

(Impacts of Storage Class Memory on the Embedded Systems Organizations and Operating Systems)

백승재 (Seungjae Baek), 최종무(Jongmoo Choi)

Department of Computer Science & Engineering, Dankook University
{ibanez1383, choijm}@dankook.ac.kr

요 약

본 논문에서는 최근에 활발하게 개발되고 있는 스토리지 클래스 메모리(Storage Class Memory, 이후 SCM)가 컴퓨터 구조와 운영체제에 미치는 영향을 논의한다. SCM은 비휘발성의 특징과 바이트 단위 임의 접근이라는 특징을 동시에 제공하며 따라서 파일을 위한 스토리지 공간 뿐만 아니라 프로그램의 수행을 위한 공간으로도 활용할 수 있다. 이러한 특징은 성능과 비용을 고려한 컴퓨터 시스템 구조의 변화를 가능하게 한다. 본 논문에서는 SCM이 도입될 때 가능한 3가지의 새로운 임베디드 시스템 구조를 분석한다. 그리고 이 구조 중에서 SCM를 가장 적극적으로 메인 메모리와 스토리지로 동시에 사용하는 구조를 위한 새로운 운영체제의 구조를 제안한다. 새로운 운영체제는 SCM을 통합 관리하는 SCM manager와 빠른 부팅을 가능하게 하는 Instant Boot manager라는 새로운 기법을 제공한다. 제안된 운영체제는 임베디드 시스템에서 실제 구현되었으며, 효과적인 통합 관리와 부팅을 제공함을 검증할 수 있었다.

1. 서 론

최근 FeRAM(Ferroelectric RAM), MRAM(Magnetic RAM), PCRAM(Phase-change RAM) 등 다양한 스토리지 클래스 메모리(Storage Class Memory, 이후 SCM이라 칭함. NVRAM (Non-volatile RAM) 또는 SCRAM(Storage Class RAM)이라고도 함)가 활발하게 개발되고 있다 [1]. SCM은 비 휘발성과 함께 바이트 단위 임의 접근이라는 특징을 제공한다. 결국 SCM은 파일 저장 공간인 스토리지로 활용될 수도 있으며 동시에 프로그램 수행 공간인 메인 메모리로 활용될 수도 있다.

SCM은 스토리지로 활용한 대표적인 연구들이 [2, 3, 4]이다. [2]에서는 디스크와 SCM을 함께 사용하는 시스템에서 자주 참조되는 데이터를 SCM에 저장하여 성능 향상을 시도하였으며, [3]에서는 MRAM을 메타데이터의 저장 공간과 쓰기 버퍼로 이용하여 파일 시스템의 성능과 신뢰성을 향상시킨 연구이다. 한편, [4]에서는 플래시 메모리와 FeRAM을 혼용하여 임베디드 시스템에서 저장 장치의 성능 향상을 시도한 논문이다. 반면, SCM을 메모리 공간으로 활용하는 연구도 진행되고 있다 [5, 6]. [5]에서는 비휘발성 메모리를 파일 캐시로 사용하고 운영체제의 결함을 복구하는 방법을 제안하여 시스템의 신뢰성을 높였으며, [6]에서는 PCRAM을 메인 메모리로 활용하여 성능과 전력을 줄이는 방법을 연구하였다. 이러한 연구는 SCM이 실제 다양한 컴퓨터 시스템에 활용될 수 있음을 보여준다.

본 논문에서는 SCM이 도입되었을 때 컴퓨터 시스템의 구조 변화, 특히 임베디드 시스템에서 구조 변화를 연구한다. 우선 본 연구진은 기존의 컴퓨터 시스템이 메인 메모리로 SDRAM을 사용하고 스토리지로 플래시 메모리를 사용하고 있다고 가정한다. 그리고 SCM을 메인 메모리로 사용하는 새로운 구조(SCM as main memory)와 스토리지로 사용하는 구조(SCM as storage), 그리고 두 가지 용도로 동시에 사용하는 구조 (SCM as both)를 기존 시스템과 성능 및 전력 측면에서 비교한다.

그리고 컴퓨터 시스템 구조의 변화 중에서 SCM as both 구조를 위한 운영체제를 설계 및 구현한다. 이 구조는 SCM을 가장 적극적으로 활용하는 구조로 사실상 운영체제의 많은 변화를 요구한다. SCM을 메인 메모리 공간과 스토리지 공간으로 동시에 활용하면 프로그램의 수행 데이터와 파일을 동일 저장 매체에 유지할 수 있어 단일한 객체로 관리할 수 있다. 이는 메모리 객체와 파일 객체 간에 복사를 줄일 수 있어 성능을 향상시킬 수 있으며, 관리 소프트웨어의 복잡도를 크게 줄일 수 있다. 또한 비휘발성 메인 메모리의 특성을 활용하면 인스턴트 부팅을 제공할 수 있으며, 이는 가전제품에 효과적으로 활용될 수 있다.

본 논문의 구성은 다음과 같다. 2절에서는 SCM의 도입으로 가능한 새로운 구조 3 가지를 살펴보고, 각 구조의 장단점을 정성적인 측면에서 논의한다. 3절에서는 3 가지 구조 중에서 SCM만을 사용하는 구조를 위한 운영체제를 설계한다. 4절에서는 설계한 운영체제의 성능을 정량적으로 분석하고, 마지막으로 5절에서 결론을 맺는다.

2. SCM 도입으로 가능한 컴퓨터 구조

그림 1은 SCM 도입으로 가능한 새로운 임베디드 시스템 구조를 보여준다. 그림 1 (a)는 기존의 시스템 구조로 SDRAM을 메인 메모리로 사용하고 플래시 메모리를 스토리지로 사용한다. 플래시 메모리는 NOR 유형과 NAND 유형으로 세분될 수 있으며, NOIR 유형은 주로 부트 로더나 프로그램의 수행 공간으로, NAND 유형은 주로 파일 저장 공간으로 사용된다. 그림 1 (b)는 플래시 메모리 대신 SCM을 스토리지로 활용한 구조이며, 그림 1 (c)는 SDRAM 대신 SCM을 메인 메모리로 활용한 구조이다. 전자는 SCM의 비휘발성 특성을 이용한 것이며, 후자는 SCM의 바이트 단위 접근 가능성을 이용한 것이다. 마지막으로 그림 1 (d)는 SCM을 메인 메모리이면서 동시에 스토리지로 활용한 구조이다. 한편, SCM을 SDRAM과 함께 메인 메

1) 이 논문은 2008년도 정부(교육과학기술부)의 재원으로 한국과학재단의 지원을 받아 수행된 연구임(No. R01-2008-000-12028-0)

모리로 혼용하는 구조나, SCM을 플래시 메모리와 함께 스토리지로 사용하는 혼용 구조(hybrid organization)도 가능하지만 본 논문에서는 고려하지 않는다.

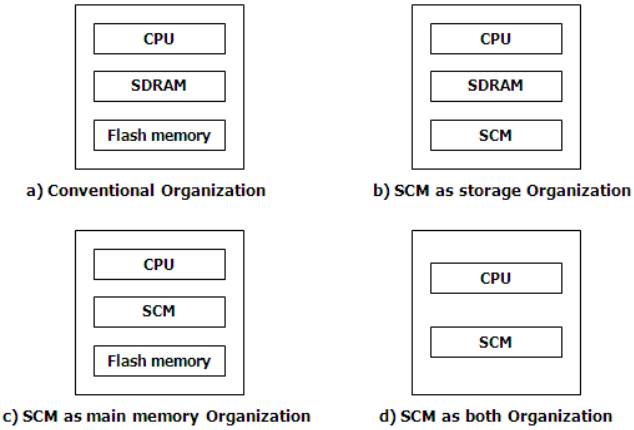

그림 1. SCM을 도입한 임베디드 시스템 구조

각 구조는 다음과 같은 장단점이 있다. 우선 SCM은 플래시 메모리에 비해 접근 속도가 빠르다. [6]의 연구에 따르면 CPU가 4GHz로 동작하는 PC 환경에서 SCM 중에 한 종류인 PCRAM과 플래시 메모리의 접근 속도를 비교하면, PCRAM이 32배 정도 빠른 것으로 분석되었다. 한편 본 연구진의 실험에서는 400MHz로 동작하는 ARM9 CPU를 탑재한 임베디드 환경에서 SCM의 한 종류인 FeRAM과 플래시 메모리의 접근 속도를 비교하였는데, FeRAM이 5~12배 정도 빠른 것으로 측정되었다. 결국 SCM은 플래시 메모리에 비해 빠르며, 따라서 그림 1 (b) 구조는 기존 시스템인 그림 1 (a) 구조에 비해 파일에 대한 빠른 접근 이득을 얻을 수 있다.

반면 SCM은 SDRAM에 비해 느리다. 역시 [6]의 분석에 다르면 PC 환경에서 PCRAM이 DRAM에 비해 4배 정도 느린 것으로 분석되었고, 본 연구진의 실험에서도 사용한 FeRAM이 SDRAM에 비해 2~4배 느린 것으로 측정되었다. 즉, 그림 1 (c)구조는 그림 1 (a) 구조에 비해 프로그램의 수행 속도가 저하될 것이다 (물론 CPU cache의 효과에 의해 2~4배까지 저하되지는 않을 것으로 예상된다).

그림 1 (c)의 구조가 그림 1 (a)의 구조에 비해 얻을 수 있는 장점은 저 전력이다. 컴퓨터 시스템에서 메인 메모리가 소비하는 에너지가 전체의 40%에 달하고 있으며 [6], 이에 따라 메모리의 전력 소비를 줄이려는 노력이 특히 임베디드 시스템에서 활발하게 진행 중이다. [7]의 분석에 따르면 SCM은 DRAM 에 비해 동적 에너지 소비를 평균 53%, 전력 누출에 따른 에너지 소비를 평균 73% 줄일 수 있는 것으로 나타났다. 결국 속도의 저하는 발생하지만 그에 따른 에너지 절약이라는 장점을 얻게 되는 것이다.

그림 1 (d)는 좀 더 적극적인 접근 방법으로 SCM으로 플래시 메모리와 SDRAM을 모두 대체하는 구조이다. 이 구조는 파일 접근에서 속도 이득을 볼 수 있으며, 반면 프로그램 수행에서는 속도 저하를 야기할 것이다. 한편, 저 전력 측면에서는 SCM이 DRAM과 플래시 메모리보다 소비 전력이 적으므로 상당한 이득을 얻을 것이다. 본 연구진이 가장 관심이 있는 구조가 바로 이 구조이며, 이후 이 구조를 위한 운영체제의 구현 및 성능 평가 내용을 다룬다.

3. SCM as both 구조를 위한 운영체제

그림 2는 그림 1 (d) 구조를 위해 본 연구진이 설계한 운영체제의 구조를 보여준다.

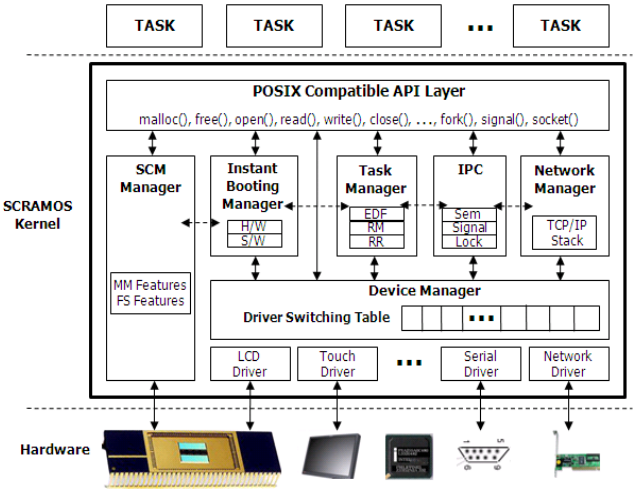

그림 2. SCM as both organization을 위한 운영체제

제안한 운영체제는 uC/OS-II 같은 실시간 운영체제로 POSIX 인터페이스를 따르며 내부적으로 태스크 관리자, IPC관리자, 네트워크 관리자, 장치 관리자, SCM 관리자, Instant Booting 관리자 등으로 구성된다. 이 중에서 SCM을 위해 특별하게 설계된 것이 SCM 관리자와 Instant Booting 관리자이다.

SCM 관리자는 SCM을 직접 관리하며 일부는 파일 시스템 공간으로 사용하고 일부는 프로그램의 수행 공간으로 사용한다. 현재 파일 시스템 공간은 FAT (File Allocation Table) 기법으로 관리하고 있으며, 프로그램의 수행 공간은 buddy 시스템 기법으로 관리하고 있다. 한편, 파일 객체와 메모리 객체가 같은 매체에 유지되어 있음으로, 이 두 객체 간에 복사 없는 전환이 가능하다. 하지만 이 내용은 본 논문의 범위를 벗어나며 관심 있는 독자는 [8]을 참고 하면 된다.

Instant Booting 관리자는 빠른 부팅 및 전원 결함에 대한 복구를 관리하는 모듈이다. 본 연구진이 고려하는 그림 1 (d) 환경에서는 프로그램의 모든 메모리 내용이 비 휘발성 매체에 존재한다. 즉, 전원 off/on 이후에도 그 내용이 그대로 유지된다. 따라서 CPU의 레지스터 내용만 유지할 수 있다면, 전원 on 시점에는 그 전의 off 했던 시점 상태를 그대로 복구하여 수행할 수 있다. 본 연구진은 이것을 Instant booting이라고 부르며, 기존의 부팅(스토리지에서 메인 메모리로 로딩하고 부팅하는)을 Normal booting이라고 한다. Instant booting을 위해서는 전원 off 요청을 받으면, CPU 레지스터 내용을 SCM의 특정 영역에 저장한다. 그 이후 부팅 시에 이 내용이 존재하고 일관성도 맞으면, 그 내용으로 부팅하게 된다 (물론 사용자가 부트 로더에서 Normal booting을 요구하면, 모든 초기화를 진행하는 일반적인 부팅 과정을 진행 한다).

Instant booting을 위해 한 가지 고려해야 할 것은 전원 결함이다. 즉, 갑작스러운 전원 결함이 발생할 경우 CPU 레지스터 내용을 저장할 수 없게 되며 결국 부팅 과정에 일관성이 어긋날 수 있다. 이 문제를 해결하기 위해 본 연구진은 전원 결함을 발견하고, CPU 레지스터 내용을 SCM에 저장할 수 있을 정도의 시간 동안 추가적인 전원을 공급할 수 있는 하드웨어 모듈을 사용하였다. 그림 3은 본 연구진이 직접 제작한 모듈을 보여준다. 이 모듈은 전원 결함을 탐지하는 부분과 잠시 동안

추가적인 전원을 공급할 수 있는 커패시터(capacitor) 등으로 구성된다. 추가적인 전원을 공급하는 시간은 CPU의 레지스터 값만을 저장할 정도의 시간이면 충분하며, ARM에서는 다중 적재/저장을 지원하기 때문에 비교적 적은 시간으로 충분한 것으로 분석되었다 (만일 CPU 캐시 중 dirty 한 것이 있으면 그것도 flush 해야 한다).

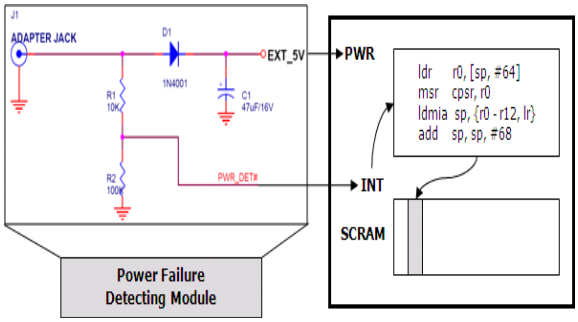

그림 3. 전원 결함 탐지 하드웨어 모듈

Instant booting에서 전원 결함 복구는 하드웨어 모듈을 사용하지 않고 소프트웨어 적인 방법으로도 가능하다. 구체적으로 문맥 교환 시에 태스크들의 CPU 레지스터를 메모리에 저장해 놓기 때문에 현재 running state의 CPU 레지스터 내용만 복구하면 되는데, 현재 running state의 태스크의 가장 마지막으로 문맥 교환 시점에 CPU 레지스터 내용은 이미 저장된 문맥에 존재한다. 따라서 그 상태로 복구는 가능하고, 그 이후 시점부터 전원 결함 시점까지 변경된 메모리 내용만 redo 하면, 가장 마지막 문맥 교환 시점으로 복구하면 된다. 이를 위해서는 COW(Copy on write)를 사용할 수 있는데, 이것은 상당한 부하가 있을 것으로 예상된다. 소프트웨어적인 접근 방법은 계속 연구 중이다.

4. 성능 평가

그림 2에서 설계한 운영체제는 400MHz PXA 255 ARM 9 처리기와 64MB SDRAM, 64MB NAND 유형 플래시 메모리, 32MB FeRAM이 탑재된 임베디드 보드에서 구현되었다. 그림 4는 실험에서 사용한 임베디드 보드를 보여준다.

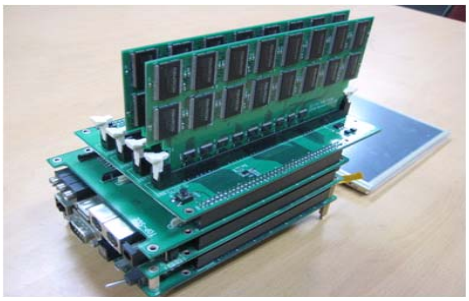

그림 4. 실험에 사용한 임베디드 보드

표 1은 그림 4의 실험 환경에서 SDRAM과 SCM의 접근 시간을 보여준다. 전체적으로 SDRAM의 접근 시간이 SCM의 한 종류인 FeRAM에 비해 2~4o 빠른 것을 알 수 있다. 이는 [6]의 결과와도 일치하며 결국 SCM을 사용하여 프로그램을 수행하면 수행 시간이 증가할 것임을 예상할 수 있다.

표 1. SDRAM과 SCM 접근 시간 (단위: us)

Operation		SDRAM	FeRAM
load	512Byte	10.65	27.11
	1024Byte	21.36	53.41
	4096Byte	82.53	211.47
store	512Byte	7.31	20.16
	1024Byte	14.12	40.71
	4096Byte	56.56	157.58

표 2는 FeRAM와 플래시 메모리의 성능 차이를 보여준다. 예상대로 FeRAM은 4~6배 빠르며 결국 파일 관련 연산의 수행이 빨라질 수 있음을 의미한다.

표 2. SCM과 플래시 메모리 접근 시간 (단위: us)

Operation	FeRAM	Flash memory
read	330	1001
write	370	2303

마지막으로 표 3은 본 연구진이 제안한 Instant booting 기존의 Normal booting 간에 차이를 보여준다. Instant booting이 기존의 방법이 비해 빠르며, 전원 결함이 발생하는 경우에도 적절하게 복구함을 검증하였다. 이 장점은 가전제품이나 휴대폰에 효과적으로 사용될 것으로 기대한다.

표 3. Instant booting의 성능 (단위: second)

Operation	Proposed OS
Normal booting	2.41
Instant booting	0.06

6. 결 론

본 연구에서는 SCM이 가져올 임베디드 시스템의 구조 변화를 논의하고, 가장 적극적으로 SCM을 사용하는 SCM as both 구조를 위한 운영체제를 설계하였다. 구현 실험을 통해 SCM과 SDRAM, 플래시 메모리 간에 성능을 비교하였고, 제안한 Instant booting이 효과적으로 동작함을 검증하였다.

하지만 이 연구는 아직 시작 단계일 뿐 많은 작업이 남아있다. 우선, SCM과 SDRAM, 플래시 메모리의 성능 특성을 좀 더 많은 워크로드에서 비교해야 한다. 둘째, 성능 특성 뿐만 아니라 저 전력 특성도 정량적으로 분석해야 한다. 이를 통해 그림 1에서 살펴본 구조들의 장단점을 좀 더 체계적으로 비교해야 한다. 마지막으로 SCM을 이용한 운영체제의 다양한 응용을 발굴하고 적용해야 한다.

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] A. A. Wang, P. Reiher, G. J. Popek, and G. H. Kuenning, "Conquest: Better Performance Through A Disk/Persistent-RAM Hybrid File System," In the Proceedings of the 2002 USENIX Annual Technical Conference, pp. 15-28, Jun. 2002.

[3] 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, pp. 83-87, May 2001.

[4] In Hwan Doh, Jongmoo choi, Donghee Lee, Sam H. Noh, "Exploiting Non-volatile RAM to Enhance Flash File System Performance", ACM & IEEE International Conference on Embedded Softrware, 2007.

[5] P. M. Chen, W. T. Ng, G. Rajamani, and C. M. Aycock, "The Rio File Cache: Surviving Operating System Crashes," In Proceedings of the 7th International Conference on Architectural Support for Programming Languages and Operating Systems, pp. 74-83, Oct. 1996.

[6] M. K. Qureshi, V. Srinivasan, and J. A. Rivers, "Scalable High Performance Main Memory System Using Phase-Change Memory Technology", In Proceedings of the 36th International Symposium on Computer Architecture, 2009.

[7] P. Zhou, B. Zhao, J. Yan and Y. Zhang, "A Durable and Energy Efficient Main Memory Using Phase Chang Memory Technology", In Proceedings of the 36th International Symposium on Computer Architecture, 2009.

[8] S. Baek, K. Sun, J. Choi, E. Kim, D. Lee and San H. Noh, "Implementation Study of a Unified SCRAM Manager for Storage Class Random Access Memory", IWSSPS, 2008, http://www.iwssps.org/index_2008.php.