트리 자료구조를 이용한 비 휘발성 메모리의 쓰레기 수집 기법

이도근 ○ 원유집

한양대학교 컴퓨터 소프트웨어 학과

matureelf@hanyang.ac.kr, youjip.won@gmail.com

Garbage Collection Technique for Non-volatile Memory Using Tree Data Structure

Dokeun Lee○ Youjip Won

Dept. of Computer Software, Hanyang University

요 약

비 휘발성 메모리는 DRAM과 달리 전원이 차단되어도 그 데이터를 영구 보존 가능하다. 기존 DRAM 기반의 시스템에 맞추어 설계된 쓰레기 수집기들은 비 휘발성 메모리의 소자적 특성으로 인하여 발생한 쓰레기들을 처리할 수 없는 단점이 있었다. 본 논문에서는 비 휘발성 메모리 기반의 시스템에서 사용자가 더 이상 사용이 불가능한 메모리 영역을 모아 해결하는 기법을 제시하고, 이를 영속 힙 기반의 HEAPO에 구현하였다.

1. 서 론

비 휘발성 메모리는 DRAM과 달리 전원이 차단되어도 영속적으로 저장된 내용을 기억할 수 있는 메모리로서, Byte-addressable하여 코드를 실행시킬 수 있는 메모리의 특성과 데이터의 영구보존이 가능한 스토리지의 특성을 둘 다 가지고 있는 차세대 메모리이다. 현재 개발되고 있는 비 휘발성 메모리는 PRAM[1], FRAM[2], STT-MRAM[3], ReRAM[4]등으로, 플래시 메모리에 비하여 쓰기 작업이 자유롭고 속도 또한 DRAM에 근접한다. 각 비 휘발성 메모리의 소자적 특성은 표 1과 같으며, 아직까지는 각 메모리 소자마다 강점을 가지는 부분과 약점을 가지는 부분이 혼재되어 있다. 비 휘발성 메모리에 존재하는 객체들은 소자적인 특성으로 인하여 비 휘발성 성질을 갖는다. 이를 잘 이용하면 데이터를 스토리지로부터 읽어와 메모리에 상주시키는 오버헤드를 없앨 수 있기 때문에 어플리케이션을 더 빠르게 구동할 수 있다. 본 논문에서는 비 휘발성 메모리에 존재하여 시스템의 재 시작과 관계 없이 영속적으로 데이터를 보존할 수 있는 객체를 '영속 객체'라고 명명하였다. 비 휘발성 메모리에 존재하는 영속객체들은 해당 객체를 가리키는 포인터가 존재하지 않을 경우 쓰레기화 된다. DRAM기반의 시스템에서는 시스템 재 시작시에 위와 같은 쓰레기들이 점유하고 있는 메모리 영역이 자동적으로 초기화되기 때문에 이러한 상황이 크게 문제가 되지 않는데 비해, 비 휘발성 메모리 기반의 시스템에서는 쓰레기화 된 메모리 영역들이 자동 초기화 되지 않기 때문에 쓰레기들을 정리할 수 있는 쓰레기 수집 기법이 필요하다. 물론 비 휘발성 메모리 역시 DRAM과 같이 사용할 수도 있으나 영속객체의 장점을 포기해야 하는 단점이 생긴다.

표 1. 비 휘발성 메모리 소자의 특성[]

Device Type	FRAM	MRAM	STT-MRAM	PRAM	ReRAM
Present Density	128Mb/ chip	32Mb/ chip	2Mb/ chip	512Mb/ chip	64kb/ chip
Cell Size(SLC)	6F^2	20F^2	4F^2	5F^2	6F^2
MLC Capability	No	2bits/ cell	4bits/ cell	4bits/ cell	2bits/ Cell
Program Energy/bit	2pJ	120pJ	0.02pJ	100pJ	2pJ
Access Time (W/R)	50/ 75ns	12/ 12ns	10/ 10ns	100/ 20ns	10/ 20ns
Endurance/ Retention	10^15/ 10yr	10^16/ 10yr	10^16/ 10yr	10^5/ 10yr	10^6/ 10yr

일반적인 의미의 쓰레기 수집 기법은 메모리 공간 확보를 위해 사용되지 않는 영역들을 모아서 해제하는 기법이다. 기 개발된 대다수의 쓰레기 수집 기법들은 프로그래밍 언어 레벨에서 지원되고 있으며 자바, 닷넷 등의 언어에서는 포인터 추적 기법, 파이썬 등 에서는 참조 횟수 계산 방식을 이용하여 쓰레기 수집을 한다.

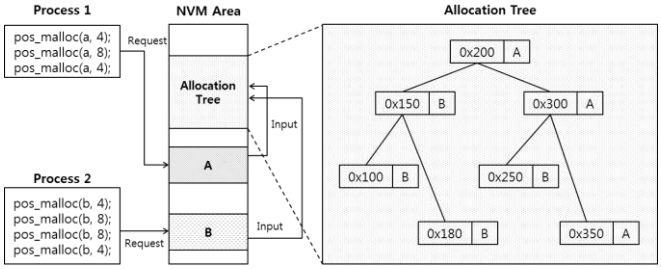

그림 1. Allocation Tree 동작 예

포인터 추적 기법은 한 개 이상의 변수가 접근 가능한 메모리를 앞으로 사용할 수 있는 메모리로 간주하여 이 외의 메모리를 해제하는 기법이고, 참조 횟수 계산 방식은 각 객체의 참조 횟수를 기억하여 참조 횟수가 0이 되면 쓰레기로 간주하고 해당 객체를 해제하는 기법이다. DRAM기반의 시스템에서는 프로그래밍 언어 레벨에서의 쓰레기 수집이 유용할 수 있으나, 비 휘발성 메모리 기반에서는 소자적 특성으로 인한 쓰레기를 처리할 수 없는 단점이 있다. 예를 들어 사용자의 프로그래밍 에러로 발생한 쓰레기는 DRAM기반의 시스템에서 프로그램 종료 시 메모리에 데이터가 남지 않아 문제가 되지 않으나, 비 휘발성 메모리 기반에서는 영구적으로 메모리에 존재하게 되는데 기존에는 이를 처리할 마땅한 방법이 없었다. 본 논문에서는 비 휘발성 메모리 기반의 시스템에서 사용되지는 않고 메모리 공간만 점유하는 쓰레기들을 모아 해결하는 기법을 개발하였고, 영속 힙 기반의 메모리 할당 플랫폼인 “HEAPO”[5] 위에서 이를 구현하였다.

2. 설 계

본 논문에서는 비 휘발성 메모리 기반 시스템에서의 쓰레기 수집을 위해 Allocation Tree를 그림 1과 같이 설계하였다. Allocation Tree는 비 휘발성 메모리 영역의 할당 정보를 관리하기 위한 자료구조로서, 사용자의 할당 요청이 있을 때 마다 해당 영역의 시작 주소를 Allocation Tree에 입력하여 할당된 메모리 영역을 추적/관리한다. 이는 사용자 요청 데이터와는 별개로 관리되는 자료 구조이므로 포인터가 끊겨 사용자가 더 이상 접근이 불가능하게 된 영역의 정보까지 보존할 수 있다. 이처럼, Allocation Tree는 모든 사용자 프로세스의 메모리 할당 정보를 포함하고 있기 때문에 쓰레기 수집기가 Allocation Tree와 사용자 데이터를 비교하여 어떤 영역이 쓰레기인지 판단하는 것이 가능하다. 이 과정에서 Allocation Tree를 순회하여 할당 메모리 영역의 시작 주소를 검색하는 작업이 필요하기 때문에 Allocation Tree의 검색 속도를 빠르게 하는 것 또한 중요한 이슈이다. 본 논문에서는 이를 위해 메모리 할당 정보를 트리 자료구조로 설계하여 최악의 경우에도 log(n)시간 안에 검색이 완료되도록 하였다. 쓰레기로 판단된 영역은 따로 모아 메모리 공간이 부족할 시 쓰레기 수집기에 의해 일괄 해제된다. 그림 2는 본 기법의 쓰레기 수집 과정을 설명한 그림이다. ‘Deleted’ 노드는 해제되어 더 이상 사용할 수 없는 노드이고, 이 때문에 해당 노드의 자식 노드들은 쓰레기가 된다. 모든 메모리의 할당 정보는 오른쪽의 Allocation Tree에서 찾을 수 있으나, 쓰레기가 된 메모리 노드는 사용자 데이터에서 포인터가 끊겨 접근될 수 없다. 쓰레기 수집기는 이 정보를 가지고 사용자의 노드 하나마다 Allocation Tree를 순회하여 해당 노드의 시작 주소가 Allocation Tree에 존재하는지 검사하고 시작 주소가 존재하면 쓰레기가 아닌 것으로 표시한다. 쓰레기 수집기는 쓰레기가 아닌 것으로 표시되지 않은 메모리 영역을 Allocation Tree에서 검색하여 따로 쓰레기 수집 리스트에 모아놓는다. 이렇게 모인 영역은 메모리 공간이 부족할 때 일괄 해제되어 시스템의 메모리 공간 활용도를 높인다.

3. 구 현

본 논문에서는 비 휘발성 메모리에서의 메모리 할당을 기록하고자 HEAPO 플랫폼을 일부 수정하여 Allocation Tree를 구현하였다. HEAPO는 영속 힙을 구현한 공유 라이브러리로서, 프로세스의 비 휘발성 메모리 영역에 대한 할당 요청을 가능하게 한다. Allocation Tree는 비 휘발성 메모리 영역에 전역으로 존재하는 자료 구조이므로 현재 프로세스가 사용하는

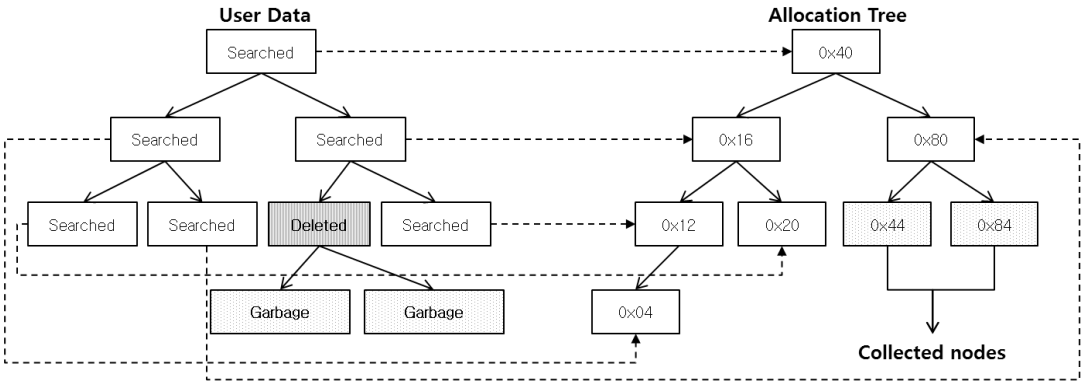

그림 2. Garbage 판정 과정

비 휘발성 메모리 할당 영역에 대해서만 쓰레기 수집을 하기 위해서는 현재 프로세스가 사용하는 영역과 그렇지 않은 영역을 Allocation Tree안에서 구분하는 데이터가 필요하다. 오브젝트 저장소 ID는 현재 프로세스가 사용하는 비 휘발성 오브젝트 저장소 영역을 표시하기 위한 데이터로서 HEAPO의 이름 테이블의 각 오브젝트 저장소들의 이름을 해싱한 값이다. 이 정보와 할당된 메모리 영역의 시작 주소를 동시에 저장하기 위해 Allocation Tree를 Key-value store로 구현하였다. HEAPO가 사용자의 비 휘발성 메모리 영역에 대한 할당을 처리할 때 마다, 쓰레기 수집기는 그림 1과 같이 Allocation Tree에 해당 영역의 시작 주소를 Key로, 요청된 영역의 오브젝트 저장소 ID를 Value로 Allocation Tree에 입력한다.

4. 실험 결과

성능 실험은 Intel i7-3770@3.40Ghz, 16GB DRAM 환경에서 진행되었다. 사용한 OS는 Linux 2.6.32 이고, 트랜잭션 환경에서 B-tree에 여러 스레드가 노드를 동시에 삽입하는 프로그램을 실행하여 쓰레기를 생성하였다. 생성된 가비지들은 노드 삽입이 끝난 후 일괄 해제되며, 이 시간을 측정하여 쓰레기 수집 시간으로 기록하였다. 실험 결과, 발생된 쓰레기 수는 삽입 연산의 횟수보다는 스레드의 개수에 비례함을 알 수 있었으며, 쓰레기 수집시간은 입력된 노드의 개수에 비례함을 알 수 있었다. 이는 생성된 쓰레기를 해제하는 시간보다 쓰레기를 판정하기 위해 Allocation Tree를 탐색하는 시간이 전체 쓰레기 수집 시간의 대부분을 차지하여 발생한 결과이다.

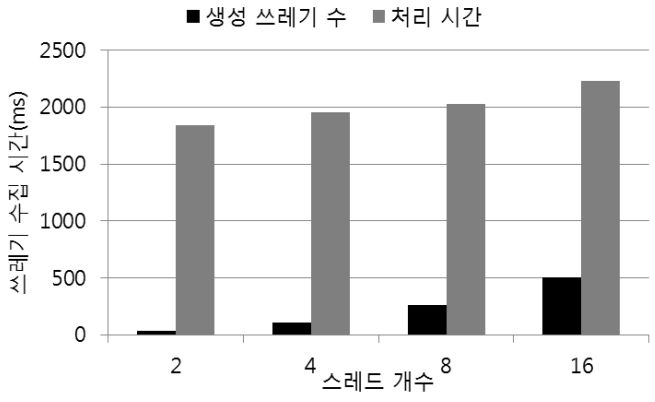

그림 3. 스레드 개수에 따른 쓰레기 수집 시간

5. 결론 및 향후 연구

본 논문에서는 비 휘발성 메모리를 위한 쓰레기 수집 기법을 개발하였고, 이를 영속 힙 기반의 HEAPO 플랫폼 위에서 구현하였다. STM을 이용하여 쓰레기를 생성시켜 본 기법의 수행 시간을 측정하였고 영속 객체의 할당 및 Allocation Tree의 정상 동작을 확인할 수 있었다. 본 기법은 생겨난 쓰레기의 수와 관계 없이 Allocation Tree에 입력된 노드의 수가 많아질수록 증가하는 단점을 가지고 있다. 향후 계획으로는 본 쓰레기 수집기의 수집 오버헤드를 측정하고 다른 쓰레기 수집 기법과의 차이를 분석하는 것이 앞으로의 연구 과제이다.

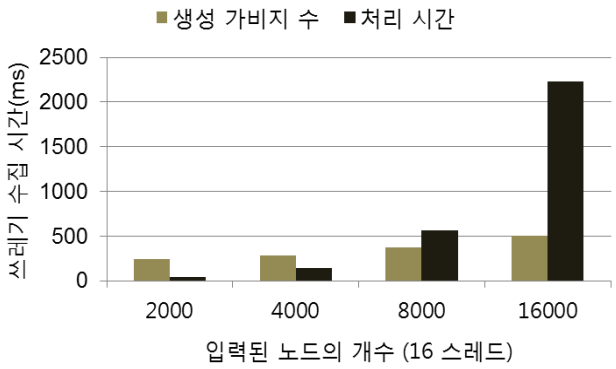

그림 4. 노드의 개수에 따른 쓰레기 수집 시간

6. 사 사

본 연구는 지식경제부 및 한국산업기술평가관리원의 산업원천 기술개발사업(정보통신)의 일환으로 수행하였음, [No.10041608, 차세대 메모리 기반의 스마트 디바이스용 임베디드 시스템 소프트웨어]

참 고 문 헌

[1] Benjamin C. Lee, Engin Ipek (Microsoft Research), Onur Mutlu, Doug Burger (Carnegie Mellon University), "Architecting Phase Change Memory as a Scalable DRAM Alternative", ISCA 09: The 36th International Symposium on Computer Architecture, 2009.

[2] Eaton, S. S., et al. "A ferroelectric nonvolatile memory." Solid-State Circuits Conference, 1988. Digest of Technical Papers. ISSCC. 1988 IEEE International. IEEE, 1988.

[3] Huai, Yiming. "Spin-transfer torque MRAM (STT-MRAM): Challenges and prospects." AAPPS Bulletin 18.6 (2008): 33-40.

[4] “The new microcontrollers with on-chip non-volatile memory ReRAM" (Press release). Panasonic. May 15, 2012. Retrieved May 16, 2012.

[5] Taeho Hwang, Jaemin Jung, and Youjip Won, "HEAPO: Heap-based Persistent Object Store", ACM Transactions on Storage, Vol. 11, Issue 1, Dec. 2014.