비 휘발성 메모리의 특성을 고려한 쓰레기 수집 기법

*이도근, 원유집
컴퓨터 소프트웨어 학과
한양대학교
e-mail : matrureelf@hanyang.ac.kr, youjip.won@gmil.com

Garbage Collection Technique for Non-volatile Memory Characteristics

*Do-Keun Lee, You-Jip Won
Dept. of Computer Software
Hanyang University

Abstract

기존에 개발된 쓰레기 수집 기법은 DRAM 기반의 시스템에 맞추어 설계되었기 때문에, 비 휘발성 메모리의 특성으로 인한 쓰레기들은 고려하지 않았다. 본 논문에서는 비 휘발성 메모리 상에서 발생할 수 있는 쓰레기들을 추적/관리할 수 있는 쓰레기 수집 기법을 개발하였고, 비 휘발성 메모리 할당 플랫폼인 HEAPO[1]상에서 이를 구현하였다.

I. 서론

일반적으로 쓰레기 수집 기법은 더 이상 사용되지는 않지만 프로그램이 점유하여 반납되지 않은 상태의 메모리 공간들을 모아 시스템으로 반납하는 기법이다[2][3]. 기존에 개발된 쓰레기 수집 기법들은 전원이 꺼지면 데이터가 소실되는 DRAM의 특성을 고려하여 프로그램의 runtime 동안에만 초점을 맞추어 개발되었다. 이러한 이유로 대부분의 쓰레기 수집 기법은 프로그래밍 언어 레벨에서만 지원되고 시스템 레벨에서는 고려하지 않는다. 비 휘발성 메모리는 전원 유무와 관계없이 영속적으로 데이터를 보관할 수 있는 메모리로서 코드 실행시킬 수 있는 메모리의 특성과 데이터를 반 영구적으로 보존할 수 있는 스토리지의 특성을 전부 가지고 있다. 비 휘발성 메모리에 상주하는 객체들은 어떠한 이유로든 해당 객체를 찾아갈 수 있는 메타데이터 혹은 포인터가 사라지게 되면 쓰레기화 된다. DRAM 기반의 시스템과 달리 비 휘발성 메모리 기반 시스템에서는 쓰레기들이 점유하는 영역을 명시적으로 초기화 하지 않으면 영속적으로 시스템 메모리를 점유하기 때문에 이를 해결할 수 있는 쓰레기 수집 기법이 필요하다. 본 논문에서는 Allocation Tree라는 메모리 할당 추적용 메타데이터 구조를 만들어 비 휘발성 메모리의 할당을 관리하고, 이를 이용하여 비 휘발성 메모리 영역의 쓰레기를 추출하여 처리하는 기법을 개발하였다. 본 기법은 리눅스 기반 시스템에 HEAPO 플랫폼을 이용하여 구현되었고, 실제 머신에서 실험하여 이 쓰레기 수집 기법이 정상적으로 동작함을 증명하였다.

II. 본론

2.1 Allocation Tree

본 논문에서는 비 휘발성 메모리 할당을 관리 추적하기 위하여 Allocation Tree를 설계하였다. 이 자료구조는 사용자의 비 휘발성 메모리 영역에 대한 할당 요청을 기록하는 메타 데이터의 집합으로서 트리 형태를 이루고 있다. Allocation Tree는 영속 힙 기반의 비 휘

발성 메모리 할당 플랫폼인 HEAPO와 연동되어 동작하며 그 과정은 그림 1과 같다. 사용자 프로세스가 해당 객체 저장소에 할당 요청을 하면 Allocation Tree에 할당할 메모리 영역의 시작 주소와 2.3에서 설명할 저장소 ID가 입력된다.

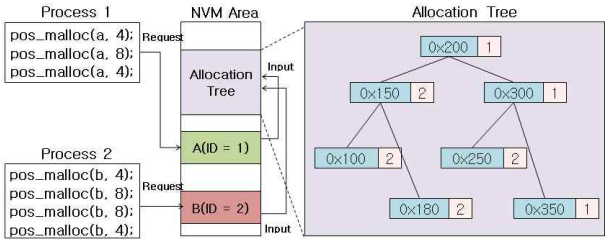

그림 1. Allocation Tree의 동작 과정

2.2 쓰레기 수집

본 논문에서 쓰레기는 '사용자가 더 이상 접근할 수 없지만, 시스템 상에서는 할당된 상태'의 메모리 영역을 의미한다. 프로그래머의 실수 혹은 시스템 오동작으로 인하여 할당만 되고 사용자가 더 이상 사용할 수 없는 영역들은 비 휘발성 메모리에서는 영구적으로 메모리에 남아 있을 수 있다. 이 기법의 목적은 이들을 효율적으로 수집하고 처리하여 시스템 메모리의 공간 활용도를 높이는 것이다. 할당 된 상태의 메모리들의 정보는 Allocation Tree에서 찾을 수 있고, 접근이 가능한 데이터들은 사용자 데이터에서 찾을 수 있으므로 두 정보를 비교하여 쓰레기를 추출할 수 있다. 쓰레기 판정 과정은 그림 2와 같다. 사용자 데이터를 순회하며 각 노드마다 각 시작 주소가 Allocation Tree에 존재하는 지 확인한다. 모든 노드의 순회가 끝나면, Allocation Tree를 순회하여 이전 과정에서 찾지 못한 노드들을 쓰레기로 판정하고 수집한다. 이렇게 수집된 노드들은 비 휘발성 메모리 영역이 부족해지면 일괄 해제되어 시스템의 메모리 공간 활용도를 높인다.

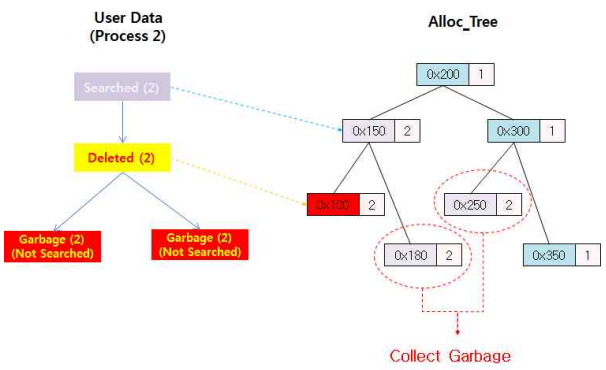

그림 2. 쓰레기 수집 과정

2.3 저장소 ID

할당된 메모리의 시작 주소만으로는 효율적인 쓰레기 수집을 하기에 정보가 부족하다. Allocation Tree는 모든 프로세스의 할당 정보를 가지고 있는 전역적 메타데이터이기 때문에 다른 프로세스가 사용하고 있는 메모리 영역을 쓰레기로 판정할 위험이 있다. 따라서 저장소 ID를 Allocation Tree에 같이 입력하여 현재 쓰레기 수집을 시도하는 객체 저장소에서만 쓰레기 수집이 되도록 하였다. 저장소 ID는 영속 객체 저장소가 생성될 때 부여되는 정수이며, key-value 형태로 시작 주소와 저장소 ID가 Allocation Tree에 입력된다.

Ⅲ. 구현

구현에 사용된 시스템은 Intel i7-3770, 16GB DRAM, Linux 2.6.32로 구성되어 있다. 비 휘발성 메모리영역에 대한 할당은 HEAPO 플랫폼을 사용하였다. HEAPO는 가상 주소 공간의 일부를 Non-volatile Memory(MVM) 영역으로 예약하고, 이 영역에 대한 할당/해제를 담당하는 라이브러리 및 시스템 콜로 구성되어 있다. HEAPO 플랫폼에 Allocation Tree를 B Tree 형태로 구현하여 포팅하였고, 저장소 ID는 HEAPO의 커널 동작 부분 중 저장소 생성 부분에 ID 생성 코드를 넣어 구현하였다. 저장소 ID는 4바이트 카운터를 이용하여 부여되고, 여러개의 저장소가 동시에 생성될 경우 같은 ID를 할당받는 상황을 막기 위하여 스핀락을 통해 원자적인 동작을 하게 하였다. 저장소 ID는 라이브러리 레벨에서 확인할 수 있어야 쓰레기 수집이 가능하기 때문에 커널모드에서 생성된 ID를 사용자 모드로 전달하기 위한 시스템 콜이 필요하다. 이 시스템 콜 또한 구현하여 HEAPO의 시스템 콜 목록에 추가하였다.

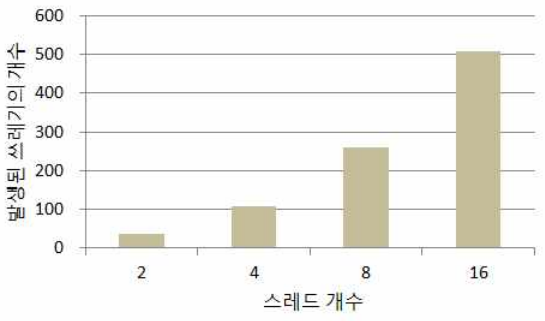

그림 3. 스레드의 개수에 따른 발생 쓰레기의 수

성능 실험에 사용된 워크로드는 여러개의 스레드가

트리에 노드를 동시에 삽입하는 동작을 반복하는 프로그램이며, 이 워크로드는 여러개의 쓰레기를 생성함을 확인하였다. 그림 3은 스레드의 개수에 따른 쓰레기의 발생량을 나타낸 그래프이다. 똑같이 16000개의 노드를 트리에 입력했을 때, 스레드의 개수가 증가함에 비례하여 발생하는 쓰레기의 개수도 증가함을 볼 수 있다.

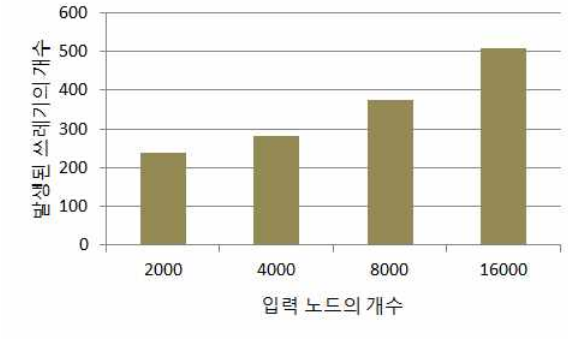

그림 4. 입력된 노드의 개수에 따른 발생 쓰레기 수

그림 4는 입력되는 노드의 개수에 따른 쓰레기의 발생량을 나타낸 그래프이다. 16개로 스레드를 고정시켜놓고, 삽입되는 노드의 수를 달리하여 생성되는 가비지의 수를 조사하였다. 노드의 입력이 많아지면 쓰레기의 발생 횟수가 많아지기는 하지만 정 비례로 증가하는 것은 아님을 볼 수 있다. 이로서 발생하는 쓰레기의 개수는 쓰레드 수의 영향을 더 많이 받음을 알 수 있다.

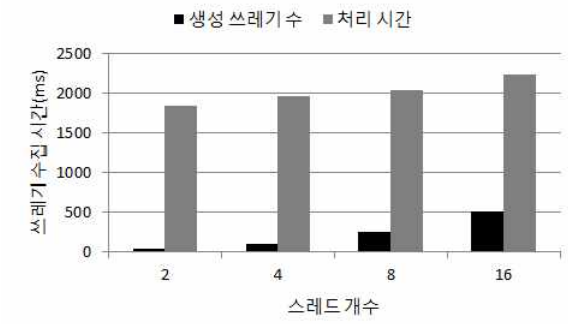

그림 5. 스레드의 개수에 따른 수집 시간

그림 5는 트리에 입력되는 노드의 개수를 16000개로 고정시키고 스레드의 수만 달리하여 쓰레기를 발생시킨 후 이를 처리하는데 걸린 시간을 조사한 그래프이다. 생성되는 쓰레기의 개수가 2배 가까이 증가하여도 전체 쓰레기 수집 시간은 약간만 증가함을 볼 수 있다. 그림 6은 스레드의 수를 고정시킨 후 입력 노드의 수에 따른 쓰레기 수집 시간을 조사한 그래프이다. 2배 정도 많은 쓰레기를 처리하는데 약 100배 정도의 시간이 더 걸리는 것을 볼 수 있다.

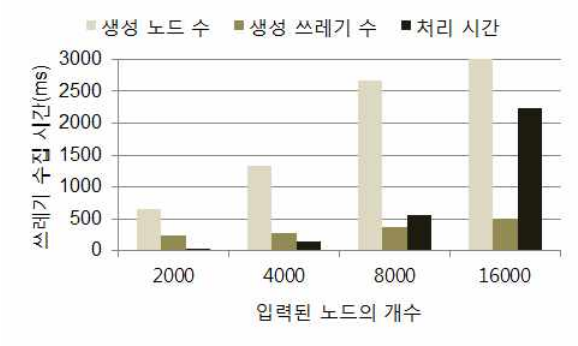

그림 6. 입력된 노드의 개수에 따른 수집 시간

그림 5의 상황과 비교해 보면, 전체 쓰레기 수집 시간은 생성 쓰레기의 수 보다는 다른 요인에 더 영향을 받음을 유추할 수 있다. 그림에서 볼 수 있듯이, 쓰레기 처리 시간은 생성된 노드의 수에 비례함을 볼 수 있는데, 이는 쓰레기를 해제하여 메모리에 반납하는 시간 보다는 Allocation Tree를 탐색하여 쓰레기를 판정하는 시간이 전체 쓰레기 수집/처리 시간에 대부분의 영향을 준다는 결론을 낼 수 있다.

Ⅳ. 결론 및 향후 연구 방향

본 논문에서는 비 휘발성 메모리를 위한 쓰레기 수집 기법을 개발하였고, 이를 영속 힙 기반의 HEAPO 플랫폼 위에 구현하여 실험하였다. 쓰레기 생성 워크로드를 만들어 수행 시간을 측정하였고, 영속 객체의 할당 및 Allocation Tree 동작이 정상적으로 동작함을 확인할 수 있었다. 향후 계획으로는 이 기법의 수집 시간과 타 기법과의 수집 오버헤드를 비교 분석 하는 것과, Allocation Tree 탐색 시간을 줄여 쓰레기 수집 시간을 줄이는 것이 앞으로의 연구 과제이다.

참고문헌

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

[2] D. F. Bacon and V. T. Rajan. Concurrent cycle collection in reference counted systems. In ECOOP, pages 207 - 235, 2001.

[3] D. F. Bacon, C. R. Attanasio, H. B. Lee, V. T. Rajan, and S. Smith. Java without the coffee breaks: a nonintrusive multiprocessor garbage collector. SIGPLAN Not., 36(5):92 - 103, 2001.