1998년도 대한전기학회 정기총회 및 추계학술대회 논문집 1998. 11. 28

내장형 장비용 자바 가상 기계에서의 실시간 쓰레기 수집기 알고리즘에 관한 연구

최 원영, 박 재현
인하대학교 자동화 공학과

Real-time Garbage Collection Algorithm for Efficient Memory Utilization in Embedded Device

Wonyoung Choi, Jaehyun Park
Department of Automation Engineering, Inha University

Abstract - Java virtual machine has the garbage collector that automate memory management.

Mark-compact algorithm is one of the garbage collection algorithm that operating in 2 phases, marking and sweeping. One is Marking is marking live objects reachable from root object set. Sweeping is sweeping unmarked object from memory(return to free memory pool). This algorithm is easy to implement but cause a memory fragmentation. So compacting memory, before memory defragmentation become serious. When compacting memory, all other processes are suspended. It is critical for embedded system that must guarantee real-time processing.

This paper introduce enhanced mark-compact garbage collection algorithm. Grouping the objects by their size that minimize memory fragmentation. Then apply smart algorithm to the grouped objects when allocating objects and compacting memory.

1. 서 론

자바(Java)는 Sun Microsystems 에서 1995년5월 소프트웨어 산업의 혁명을 기대하며 만든 프로그래밍 언어이다. 자바의 특징으로는 프로그램의 부품화를 통한 코드 재사용성과 높은 이식성, 그리고 어떠한 시스템에서도 자바 바이트코드(bytecode)를 수행시킬 수 있는 자바 가상 기계(Java Virtual Machine)등이 있다. 이러한 장점들은 개발자들이 프로그램개발 및 시스템에 따른 이식의 부담을 크게 줄여준다. 이에 따라 최근 내장형 장비에서의 자바 가상 기계의 활용을 위한 노력이 크게 늘어났다. 자바 가상 기계의 특징중 쓰레기 수집기(Garbage Collector)는 메모리의 관리를 자동화 해주는 기능을 가지고 있어서 개발자의 메모리 관리 부담을 덜어주게 된다. 하지만 쓰레기 수집기의 수행 방식에 따라 비효율적으로 메모리를 사용하게 될 수 있으며 최악의 경우 메모리 부족이 발생하여 메모리의 크기가 작은 실시간 내장형 장비에 있어서 심각한 문제를 일으키기도 한다. 따라서 실시간성 보장 및 저용량 메모리의 내장형 장비에 자바 가상 기계를 이식 할 경우 이식 대상 장비의 특성을 고려한 효율적인 쓰레기 수집기를 제작해야 한다.

이 논문에서는 기존의 쓰레기 수집 알고리즘중 mark - compact 알고리즘을 소개하고, 저용량의 메모리와 실시간성이 요구되는 내장형 장비의 특성을 고려하여 개선된 알고리즘의 소개와 시뮬레이션을 통한 결과의 분석을 하였다.

2. 본 론

2.1 Mark-Compact 알고리즘

기존의 쓰레기 수집 알고리즘중 mark-compact 알고리즘은 root 객체로 부터 객체의 참조를 추적해 나가면서 살아있는 객체들에 표시(mark)를 하고 난 뒤 표시되지 않은 죽은 객체들을 제거(sweep)하여 자유 메모리로 환원한다. 또한 객체의 다양한 크기에 따른 메모리 단편화 현상을 방지하기 위해 메모리 최적화(compact)작업을 수행한다. Mark-compact 알고리즘은 다른 알고리즘에 비해 메모리로부터 쓰레기의 제거가 용이한 장점이 있으나 메모리 크기에 비례해서 표시와 제거 작업량이 많아지며 메모리 최적화작업 수행 시 다른 모든 작업들은 중단되기 때문에 실시간성이 요구되는 시스템에서 문제가 발생한다.

2.1.2 내장형 장비에서의 쓰레기 수집

내장형 장비는 개인용 컴퓨터 환경에 비해 메모리의 크기가 매우 작고 처리 속도 또한 매우 느리고 실시간성이 보장되어야 하는 경우가 많기 때문에 내장형 장비에서의 자바가상기계의 쓰레기 수집기는 이러한 내장형 장비의 특성을 고려한 알고리즘의 적용이 요구된다.

Mark-compact 알고리즘은 쓰레기 수집에 소요되는 처리시간 및 메모리가 작기 때문에 적은 메모리를 효율적으로 활용해야하고 비교적 느린 처리속도를 가지는 내장형 장비에 적합하나 메모리 최적화 작업이 실시간성 보장에 큰 문제가 된다. 따라서 실시간성의 보장 및 효율적인 메모리 활용이 가능한 메모리 최적화작업이 이루어 질 수 있는 알고리즘의 개발이 요구된다.

2.2 개선한 Mark-Compact 알고리즘

2.2.1 알고리즘 소개.

객체들을 객체의 크기가 메모리 단편화를 최소화할 수 있는 크기로 분류한다. 예로써 객체의 크기가 m, m*2, m*3 인 객체(O_3 객체)들을 메모리의 한쪽 끝에서부터 할당해 나가고 다른 크기의 객체들(O_0 객체)은 반대편 끝에서부터 할당해 나간다. 쓰레기 수집 작업이 수행되고 난 후 O_3 객체가 있는 영역의 제거된 메모리의 크기는 m 의 배수가 된다. 따라서 m 크기의 객체들은 자유롭게 할당 될 수 있으며 m*2, m*3 크기의 객체들은 그 크기에 따라 메모리 할당의 유연성이 달라지게 된다.

또한 메모리 최적화 작업시 모든 객체를 한쪽으로 모으는 대신 빈 메모리영역의 크기를 m*3 배수가 되도록 조절하면 O_3의 모든 객체를 할당 할 수 있기 때문에 최적화 작업의 부하가 줄어들고 메모리 활용의 유연성이 높아진다. m이 충분히 클 경우 m, m*2, m*3보다 조금 작은 크기의 객체들을 O_3 의 각각의 객체들과 동일하게 취급하면 메모리 할당의 유연성은 더욱 높아지게 된다.

Mark-compact 알고리즘과의 차이점은 O_0 객체들에 대해선 mark-compact 알고리즘을 적용하고 O_3 객체에 대해선 동적인 방법으로 메모리를 관리하여 쓰레기 수집기의 수행 시기 및 수행 방식을 보다 효율적으로 하는데 있다.

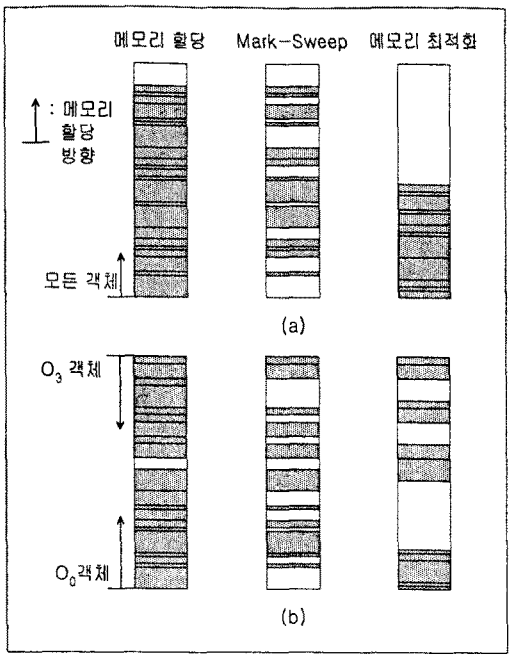

그림 1 (a) Mark-Compact 알고리즘 과
(b) 개선한 Mark-Compact 알고리즘

2.2.2 쓰레기 수집기의 수행시기

쓰레기 수집기의 수행은 실시간성 보장에 가장 큰 영향을 주게된다. 따라서 대상이 되는 시스템의 특성을 고려하여 수행시기를 결정 해야한다. 대부분의 내장형 장비는 시간지연에 민감한 높은 우선 순위를 갖는 프로세

표기	내용
P	높은 우선 순위 프로세스의 실행 주기
T_{hmax}	높은 우선 순위 프로세스의 최대 수행 시간
L	쓰레기 수집기와 낮은 우선 순위 프로세스들이 사용 할 수 있는 시간
L_p	높은 우선 순위 프로세스가 시작 되기까지 남은 시간
T_{gc}	쓰레기 수집기의 최대 수행 시간.
S	메모리 크기.
M_{hmax}	높은 우선 순위 프로세스에 의해 사용되는 최대 메모리 사용량.
M_{free}	자유 메모리의 크기
F_{mf}	O_3 메모리 영역의 메모리 단편화 정도
F_{mft}	메모리 최적화 작업에서 고려하는 F_{mf}의 한계값

표 1. 시간 및 메모리 표기

스가 낮은 우선 순위의 프로세스를 선점하는 선점형 시스템이며 시간지연에 민감한 높은 우선 순위를 갖는 프로세스는 그 수가 적고 주기적이며 사용하는 자원(메모리, 프로세스시간 등)이 매우 적다. 이 논문에서는 시간지연에 민감한 일정한 주기를 가지는 높은 우선 순위 프로세스 1개와 다음 우선 순위를 가지는 쓰레기 수집기와 가장 낮은 우선 순위 가지는 프로세스들로 구성된 시스템에서의 쓰레기 수집기의 수행을 고려하였다. 낮은 우선 순위 프로세스의 수는 제한이 없으며 쓰레기 수집기

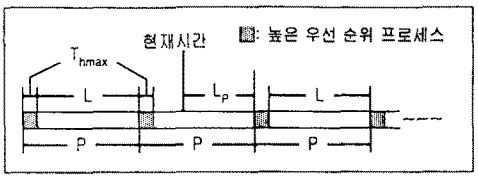

그림 2 시뮬레이션 모델의 시간 표기

와 낮은 우선 순위 프로세스들은 높은 우선 순위 프로세스의 수행주기의 사이에서 실행 되게 된다.

	$Lp > T_{gc}$	$Lp = T_{gc}$
최저 자유 메모리	0	M_{hmax}
쓰레기 수집기 수행시기	새로운 객체할당이 불가능 할 때	$M_{free} < M_{hmax}$ 또는 $F_{mf} < F_{mft}$

표 2. 최저 자유 메모리량 및 메모리 최적화 시기

표 2 는 이 시스템의 실시간성을 보장하기 위한 시간별 최저 자유메모리의 크기와 메모리 최적화 작업 수행시기를 나타낸다. 메모리의 사용은 높은 우선 순위 프로세스의 수행시간까지의 남은 시간이 최대 쓰레기 수집시간보다 많이 남아 있을 경우 메모리를 최대한 사용하며 그 이후의 시간부터는 높은 우선 순위 프로세스의 최대 메모리 사용량을 확보한 뒤 남은 메모리만을 사용하게 된다. 이 모델에서 실시간성을 보장하기 위해서는 주기적으로 실행되는 높은 우선 순위 프로세스가 실행될 때 사용되는 메모리가 충분히 확보되어야 한다. 이는 쓰레기 수집기의 수행후 높은 우선 순위 프로세스에 의해 사용되는 최대 메모리 사용량이상의 메모리를 확보할 수 있다는 가정하에서만 가능하다. 실제 높은 우선 순위 프로세스의 최대 메모리 사용량이 충분히 작을 경우 이 가정은 적합하며 이 논문의 시뮬레이션 모델에서도 이 가정을 적용하였다.

2.3 시뮬레이션

시뮬레이션은 실시간성 처리를 위한 실시간 쓰레기 수집기의 구현을 위해 자바 가상기계의 기능중 프로세스 스케줄 기능과 쓰레기 수집기능만을 C++로 구현한 자바 가상기계에 프로세스 생성기능을 추가해 시뮬레이션 하였다. 사용되는 자원은 프로세스 시간과 메모리만을 고려하였다.

2.3.1 시뮬레이션 모델 및 가정

프로세스 생성기를 통해 낮은 우선 순위 프로세스와 높은 우선 순위 프로세스가 생성된다. 높은 우선 순위 프로세스는 1개이며 일정한 주기를 가지게 되며, 낮은 우선 순위 프로세스의 수는 제한을 두지 않았다.

객체가 할당될 메모리크기는 512k bytes와 1024k bytes 이며 O_0 객체의 크기는 8 bytes에서 128 bytes 사이의 임의의 크기의 객체들이며 O_3 객체의 크기는 32, 64, 96 bytes로 시뮬레이션을 수행하였다. 프로세스의 프로세스 시간은 0 에서 10000 사이의 임의값을 가지며 객체의 크기는 1024 bytes에서 8192 bytes 사이의 임의의 값을 가진다.

2.3.2 시뮬레이션 결과 측정

시간의 측정은 프로세스를 관리하고 처리하는 각 부분마다 가중치를 두어 측정하였으며 이를 바탕으로 총 쓰레기 수집기 수행 시간을 측정하였다. 메모리에 할당되는 객체들의 크기의 합을 측정하여 얼마나 효율적으로

메모리를 사용했는지도 측정하였다. 모든 결과는 O_3 객체의 크기의 비율이 20%, 40%, 60%, 80%, 100% 인 경우에 따른 효과와 병행하여 분석하였다

2.3.3 시뮬레이션 결과 및 분석

기존의 mark-compact 알고리즘은 O_3 객체의 비율을 고려하지 않기 때문에 O_3객체의 변화에 따른 결과는 거의 변화가 없었으며 메모리 크기에 따른 쓰레기 수집기의 수행 시간은 메모리의 크기가 매우 작아 큰 차이를 보이지 않았다. 이는 메모리가 클 경우 쓰레기 수집기의 작업은 많아지지만 메모리가 적을 경우에는 메모리가 작은 만큼 자주 쓰레기 수집기를 수행해야 하기 때문이다.

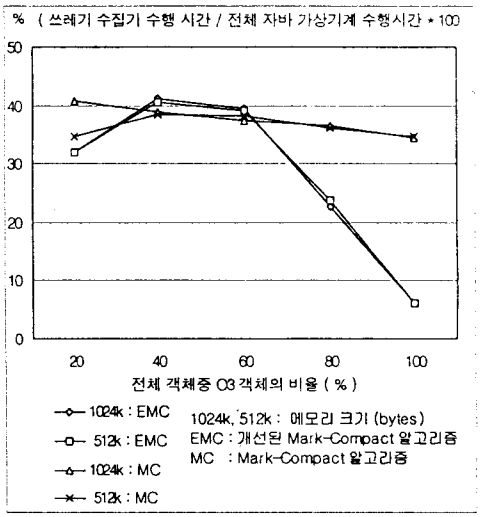

그림 3. O_3 객체의 비율과 메모리 크기에 따른 쓰레기 수집기의 수행시간

개선한 mark-compact 알고리즘의 경우 O_3 객체의 비율이 높은 경우 매우 적은 쓰레기수집의 수행으로도 훨씬 더 많은 객체들을 할당할 수 있었다.

O_3 객체의 비율이 100%인 경우에는 단지 전체 프로세스 시간의 6%만의 시간만으로 메모리에 할당되는 객체의 크기가 mark-compact 알고리즘 보다 3배 가량 많았으며 O_3 객체의 비율이 80%인 경우에는 mark-compact 알고리즘의 60% 의 시간만을 쓰레기 수집에 이용하여 2배나 더 많은 크기의 객체들을 메모리에 할당 할 수 있었다. O3 객체의 비율이 60%일 경우에는 기존의 mark-compact 알고리즘 보다 더 많은 쓰레기 수집기의 수행이 이루어 졌으나 더 많은 객체가 할당되어 쓰레기 수집기를 효과적으로 수행했음을 알 수 있다.

O3 객체가 40%일 경우에는 기존의 mark-compact 알고리즘에 비해 더 낮은 효과를 보여 주지 못했으며 O3 객체가 20% 미만일 경우에는 메모리크기가 512k bytes 일 때는 2.6%, 1024k bytes 일 때는 8.8% 의 쓰레기 수집수행시간이 절약되었다.

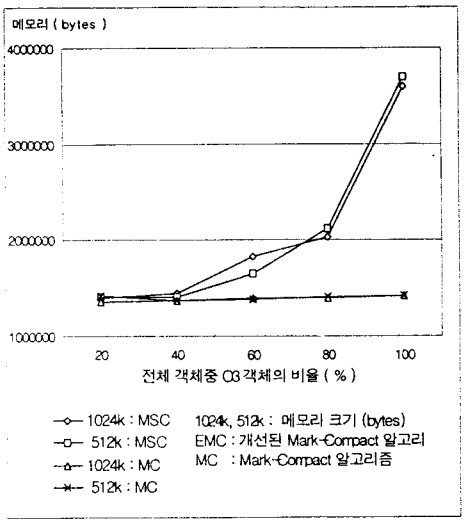

그림 4. O_3객체의 비율과 메모리 크기에 따른 메모리 사용량

3. 결 론

기존의 mark-compact 알고리즘을 메모리 최적화 시기 및 방법을 적용 대상 장비의 특성을 고려하여 개선함으로써 메모리 사용의 효율향상 및 실시간성을 보장하게 되었다.

[참 고 문 헌]

[1] Roger Henriksson, "Scheduling Real Time Ga Collection", proceddings of NWPER'94, Nordic Worksh Programming Environment Research, 1994

[2] Paul R. Wilsen, "Uniprocessor Garbage Col Techniques", Springer-Verlag Lecture Notes in Com Science, N 637, 1992

[3] C.E. McDowell, "Reducing garbage in Java", S Notices (Acm Special Interest Group on Progran Languages), V33 N.9, 1998