소규모 내장형 자바가상기계에서 메모리 공간 재사용을 위한 고정 크기 메모리 할당†

김성수°, 양희재
경성대학교 컴퓨터공학과
sskim@conet.ks.ac.kr, hjyang@star.ks.ac.kr

Fixed-Size Memory Allocation for Memory Space Reuse in Small Embedded Java Virtual Machine

Sungsu Kim°, Heejae Yang
Dept. of Computer Engineering, Kyungsung University

요 약

자바가상기계는 힙 영역과 자바 스택 영역에 객체와 스택 프레임을 할당할 공간이 없을 때 가비지 콜렉션과 함께 이미 해제된 힙과 자바 스택 영역을 재사용 가능 하도록 메모리 공간을 재구성하게 된다. 한편 메모리 단편화로 인해 객체 또는 스택 프레임을 더 이상 할당하지 못하는 경우 자바가상기계는 컴팩션을 수행하여 메모리 단편화를 제거하면서 메모리를 재구성 한다. 하지만 자바가상기계에서 메모리 재구성은 가비지 콜렉션및 컴팩션과 함께 길고 예측할 수 없는 지연시간에 의해 내장형 자바가상기계의 성능을 저하 시키는 단점을 가진다. 본 논문은 소규모 내장형 자바가상기계의 성능을 개선하기 위한 방안으로, 가변 크기를 가지는 객체와 스택 프레임을 고정 크기로 변환하여 메모리를 할당하는 고정 크기 메모리 할당에 대해 기술 하고 있다. 고정 크기 메모리 할당은 메모리 전체 사용율은 떨어지지만 외부 단편화가 발생하지 않기 때문에 회수된 메모리 공간을 재구성 하지 않고도 힙 영역과 자바 스택 영역에 객체와 스택 프레임을 할당 가능 하다. 본 논문에서 기술한 고정 크기 메모리 할당 방식으로 객체와 스택 프레임을 할당하게 되면 가변 크기 메모리 할당 보다 약 10% ~ 30% 효율향상을 보였다.

1. 서 론

자바가상기계에서 메모리 데이타 영역은 개념적으로 클래스 영역, 힙 영역, 자바 스택 영역, 네이티브 메소드 스택 영역으로 분류하며, 객체를 저장하는 힙 영역과 메소드 호출시 생성 되는 스택 프레임을 저장하는 자바 스택 영역은 바이트 코드 실행 중 빈번하게 할당되어 진다. 할당될 객체들의 크기는 객체가 포함하는 필드 수에 의해 가변 크기를 가지며, 스택 프레임의 크기는 스택 프레임이 포함하고 있는 오퍼랜드 스택의 크기와 지역변수 배열의 크기에 의해 가변 크기를 가진다[1].

자바가상기계는 힙 영역에 객체를 저장할 공간이 부족할 경우 가비지 콜렉션을 수행하여 사용하지 않는 객체를 해제한 후, 힙 영역의 메모리 공간을 재구성 하여 할당 공간을 확보 하게 된다. 그리고 자바 스택 영역에 스택 프레임을 저장할 공간이 부족할 경우 메소드 호출이 종료되어 이미 해제 되어진 스택 프레임의 메모리 공간을 재구성 하여, 자바 스택 영역의 공간을 확보 하게 된다. 이 때 메모리 재구성은 할당 가능한 연속된 메모리 공간을 가지는 블록들을 생성하게 되고 그 결과 힙과 자바 스택 영역에 대한 전체 메모리 접근으로 인해 오버헤드가 발생된다. 그리고 객체와 스택 프레임의 가변적 크기에 의해 발생한 외부 단편화는 연속된 메모리 공간 부족으로 가비지 콜렉션의 발생 횟수를 증가 시키거나 최악의 경우 컴팩션을 발생시킨다[2].

본 논문은 가비지 콜렉션 이후 메모리 재구성을 수행 하지 않아도 객체와 스택 프레임이 할당 가능한 소규모 내장형 자바가상기계의 고정 크기 메모리 할당에 대해 기술 하고 있다. 고정 크기 메모리 할당은 가변 크기를 가지는 객체와 스택 프레임을 고정 크기로 변환하여 각 메모리 데이타 영역에 할당하기 때문에 메모리 전체 사용율은 떨어지지만 외부 단편화가 발생하지 않아 가비지 콜렉션 수행 후 회수된 메모리 공간을 재구성 하지 않고도 객체와 스택 프레임을 할당 가능하다. 그리고 가변 크기 할당보다 상대적으로 가비지 콜렉션의 발생 횟수를 감소시키는 장점을 가진다.

본 논문의 구성은 다음과 같다. 2절은 본 논문의 관련 분야에 대해 소개를 하고 3절은 가변 크기 메모리 할당에 대해 간략히 기술했으며, 4절은 자바가상기계에서 고정 크기 메모리 할당에 대해 기술 했다. 5절에서는 고정 크기 메모리 할당의 실험 및 성능 평가를 했고 마지막 6절에 결론을 맺는다.

2. 관련 연구

고정 크기 메모리 할당을 적용한 자바가상기계로 simpleRTJ[3, 4]가 있다. simpleRTJ는 제한된 자원을 사용하는 내장형 시스템에서 자바가상기계의 메모리 절감과 클래스 내부 정보를 효율적으로 접근하기 위해 기존의 클래스 파일을 클래스 이미지 파일로 변환하여 사용한다. 클래스 이미지 파일은 기본적으로 정적 클래스 적재의 특성을 가지고 있기 때문에 자바가상기계가 사용하는 클래스 정보들을 모두 포함하고 있으며, 자바가상기계의 실제 동작에 필요 없는 정보들은 제외되어 있어 클래스 파일보다 상대적으로 소량의 크기를 가진다. 또한 클래스 이미지 파일 내부에는 자바가상기계에서 생성하게 되는 가장 큰 객체 및 스택 프레임의 크기를 명시하고 있어 자바가상기계에서 고정 크기 메모리 할당이 가능하다. 즉 simpleRTJ는 클래스 이미지 파일에 명시된 객체와 스택 프레임의 고정 크기로 메모리를 할당하여 가비지 콜렉션 수행 후 회수된 메모리 공간을 재구성 하지 않고도 힙 영역과 자바 스택 영역에 객체와 스택 프레임을 할당할 수 있다.

3. 가변 크기 메모리 할당

가변 크기 메모리 할당을 사용하는 내장형 자바가상기계로는 KVM[5, 6]이 있다. KVM에서 객체와 스택 프레임은 동일한 메모리 영역에 할당하여 사용되며, 가변 크기 메모리 할당을 위해 블럭 단위로 영역을 표현한다. 블럭은 할당 가능한 연속 메모리들의 구분으로 그림 1-a와 같이 블럭의 하위에서 상위로 객체와 스

† 이 논문은 2003년도 정보통신부 지원 정보통신기초기술연구지원사업에 의해 연구되었음.

택 프레임을 할당 한다. 만약 더 이상 할당할 공간이 없다면 가비지 콜렉션이 발생하여 사용하지 않는 객체 영역을 해제한다.

그림 1-b는 자바가상기계에서 사용하지 않는 객체와 스택 프레임의 가비지 영역을 블럭 단위로 메모리 공간 재구성을 수행하는 것을 나타내며 상위 블럭은 하위 블럭을 연결하고 마지막 블럭은 NULL을 가리켜 블럭 리스트의 끝을 나타낸다. 만약 블럭 리스트의 끝을 만났는데도 메모리를 할당하지 못할 때는 가비지 콜렉션이 발생한다

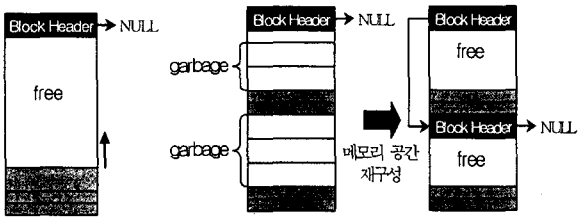

그림 1-a
메모리 할당

그림 1-b
메모리 공간 재구성

그림 1 가변 크기 메모리 할당과 메모리 재구성

그림 1과 같이 가변 크기 메모리 할당에 의한 메모리 관리는 생성하려는 객체와 스택 프레임이 가변 크기를 가지기 때문에 가비지 콜렉션이 발생하면, 항상 메모리 공간 재구성을 수행해야 한다. 또한 각 블럭들은 외부 단편화가 발생할 가능성이 높아서, 할당 가능한 연속 메모리의 부족으로 인한 가비지 콜렉션이 발생할 가능성이 높다. 최악의 경우 가비지 콜렉션을 수행하더라도 메모리를 할당하지 못하며, 이 때는 컴팩션을 수행하여 외부 단편화를 제거해야 한다.

4. 고정 크기 메모리 할당

고정 크기 메모리 할당을 위해 클래스의 인스턴스 크기와 스택 프레임의 가장 큰 크기를 이미지 파일 변환기[7]를 통해 클래스 이미지 파일 내부에 명시 한다.

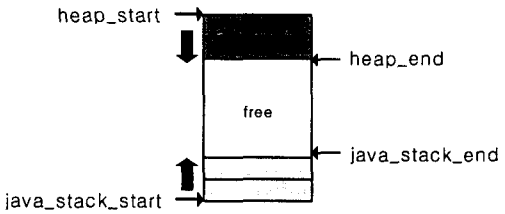

그림 2 고정 크기 할당을 위한 메모리 구조

고정 크기 메모리 할당은 그림 2에서 보는 것 과 같이 객체는 메모리의 상위부터 하위로 할당하고 스택 프레임은 메모리의 하위에서 상위로 할당하게 된다. 즉 메모리 상위는 힙 영역으로, 하위는 자바 스택 영역으로 사용된다. 각 영역은 heap_end와 java_stack_end로 서로의 경계를 나타내고 free 영역에 객체와 스택 프레임이 할당되면 마지막 할당된 위치로 그 경계가 이동하게 된다. 만약 heap_end와 java_stack_end가 서로 교차되어 그 경계를 침범하면 가비지 콜렉션과 함께 heap_end 와 java_stack_end의 위치 값을 이동·시켜 free 영역을 확보하게 된다.

고정 크기 메모리 할당에서 메모리 공간의 재구성은 컴팩션에 의해 수행하게 된다. 컴팩션은 힙 영역의 내부 단편화로 더 이상 객체를 생성하지 못할 때 수행하게 된다.

4.1 객체의 할당

자바가상기계에서 생성하는 객체 중 String 과 배열은 바이트 코드 실행 중에 객체 크기가 결정된다. 즉 고정 크기 메모리 할당에서 String과 배열은 이미 산출된 객체의 고정 크기와 비교하여 작거나 같으면 고정 크기로 할당하고, 반대로 크다면 필요한 실제 크기로 객체를 할당하게 된다.

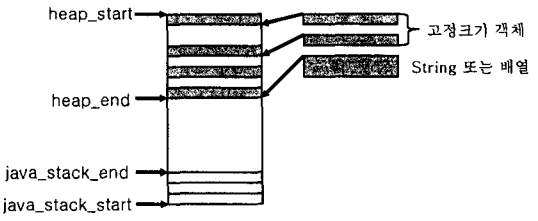

그림 3 객체의 고정 크기 메모리 할당

고정 크기 메모리 할당에서 객체는 그림 3과 같이 가비지 콜렉션 수행 후 회수된 영역에 우선 할당한다. 이 영역은 메모리 공간의 재구성을 수행하지 않고 객체를 할당 가능하도록 회수되기 직전의 객체크기를 가진다. 그 결과 할당 하려는 객체의 크기가 비어있는 공간보다 작거나 같다면 이 영역에 객체를 할당하고 객체의 크기가 비어 있는 공간보다 크다면 힙의 마지막 영역에 할당하여 heap_end를 변경된 힙의 마지막 위치로 이동 시킨다.

고정 크기 메모리 할당에서 객체의 고정 크기보다 큰 String과 배열 객체의 할당과 해제는 힙 영역의 내부 단편화를 발생 시킨다. 특히 고정 크기 메모리 할당은 가변 크기 메모리 할당 보다 메모리 사용 율이 낮기 때문에 가변 크기 메모리 할당보다 컴팩션이 발생할 가능성이 높다.

4.2 스택 프레임의 할당

메소드가 호출될 때마다 스택 프레임은 자바 스택 영역에 할당된다. 결국 메소드에서 또 다른 메소드를 호출하기 때문에 하나의 스택 프레임 위에 또 다른 스택 프레임이 놓이는 스택 프레임 리스트로 서로 연결 되어 생성된다. 스택 프레임 리스트에서 최상 단에 놓인 스택 프레임은 쓰레드에 의해 프로그램이 실행되고 있기 때문에 활성 스택 프레임(active stack frame) 이라고 부른다.

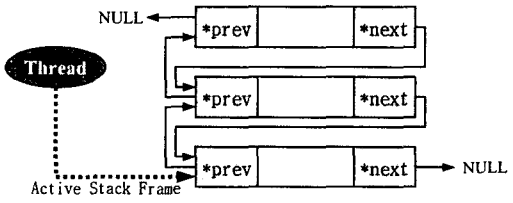

그림 4 스택 프레임의 구조

그림 4는 스택 프레임의 구조를 나타낸 것으로 활성 스택 프레임은 메소드의 실행 종료에 의해 메모리가 해제되며, 그 즉시 쓰레드는 하위 스택 프레임을 활성화 하게 된다. 즉 자바가상기계는 바이트 코드 실행 시간에 자바 스택 영역에 할당된 스택 프레임의 해제가 가능하다. 고정 크기 메모리 할당에서 자바 스택에 생성되는 스택 프레임은 항상 고정 크기를 가지고 있어 바이트 코드에 의해 해제된 스택 프레임 영역을 새로운 스택 프레임으로 할당 가능하다. 또한 외부 단편화가 없기 때문에 자바 스택 영역은 컴팩션을 수행하지 않는다.

표 1 시뮬레이션 결과

	고정 크기 메모리 할당				가변 크기 메모리 할당			
쓰레드 개수	가비지 콜렉션 발생 횟수	메모리 평균 할당 개수	컴팩션 발생 횟수	메모리 평균 접근 개수	가비지 콜렉션 발생 횟수	메모리 평균 할당 개수	컴팩션 발생 횟수	메모리 평균 접근 개수
1	94	1,171	0	150	131	840	0	227
2	195	1,129	0	159	265	831	0	221
4	421	1,046	0	173	543	811	0	209
8	985	894	0	199	1,149	766	0	191
16	2,716	648	1	256	2,537	694	0	180
20	3,321	663	18	301	3,352	657	0	170
25	5,158	533	20	339	4,703	585	0	179
30	8,086	408	21	379	6,027	548	0	187

5. 실험 및 성능 평가

본 절에서는 시뮬레이션을 통해 고정 크기 메모리 할당과 가변 크기 메모리 할당의 성능을 측정해 보았다. 본 실험을 위한 시뮬레이터는 64KB 힙 메모리를 가지는 KVM에서와 같이 힙과 자바 스택을 위해 64KB의 메모리 공간을 가진다,

메모리에 할당되는 객체와 스택 프레임의 크기는 CLDC[8]와 simpleRTJ[9]의 클래스 파일 형식 분석에 의해 가변 크기 메모리 할당에서 객체는 4Byte~60Byte, 스택 프레임은 12Byte~80Byte의 가변 크기를 가지고 고정 크기 메모리 할당에서 객체는 60Byte, 스택 프레임은 80Byte의 고정크기를 가진다. 객체와 스택 프레임의 생성 비율은 객체가 30%, 스택 프레임이 70% 비율로 생성한다. 고정 크기 메모리 할당에도 메모리 단편화가 발생 하도록 객체의 2%는 배열 형태로 객체의 2배 크기를 할당 하도록 했다.

표 1의 시뮬레이션 결과는 소규모 내장형 자바가상기계에서 쓰레드의 개수를 최대 30개로 한정 한다고 가정 후 스레드의 개수가 증가함에 따라 고정 크기 메모리 할당과 가변 크기 메모리 할당에서 가비지 콜렉션의 발생 횟수, 가비지 콜렉션 발생하기 직전 메모리에 할당한 객체와 스택 프레임의 평균 개수, 컴팩션 발생 횟수, 객체와 스택 프레임을 메모리에 할당하기 위해 평균적으로 메모리를 접근한 개수를 나타낸다.

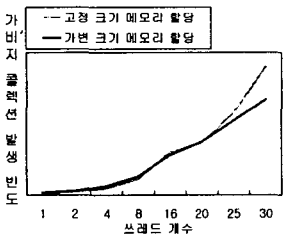

그림 5 가비지 콜렉션 빈도

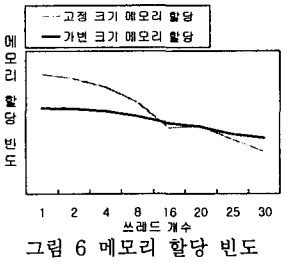

그림 6 메모리 할당 빈도

그림 5의 가비지 콜렉션 빈도에서는 고정 크기 메모리 할당이 가변 크기 메모리 할당보다 약 10%~30% 효율을 보이면서 비슷한 증가 추세를 가진다. 하지만 쓰레드 개수가 8개 이상부터 고정 크기 메모리 할당은 메모리 내부 단편화에 따른 컴팩션의 수행과 함께 가비지 콜렉션 발생 빈도가 급속하게 증가하게 되어 가변 크기 메모리 할당보다 성능이 떨어지게 된다.

그림 6의 메모리 할당 빈도에서는 고정 크기 메모리 할당의 경우 쓰레드의 개수가 8개 이하는 가변 크기 메모리 할당 보다 메모리 할당 개수에 있어서 약 10%~30% 효율을 가진다. 하지만 메모리 할당 빈도가 급속히 떨어져 스레드 개수가 8개 이상 부터는 가변 크기 메모리 할당보다 메모리 할당 개수의 효율이 감소된다.

표 1의 가변 크기 메모리 할당에서 메모리 평균 접근 개수는 쓰레드 개수가 20개인 경우 최적의 상태를 나타내었다. 반면 고정 크기 메모리 할당에서는 쓰레드 개수에 증가함에 따라 메모리 평균 접근 개수 또한 증가함을 보였다.

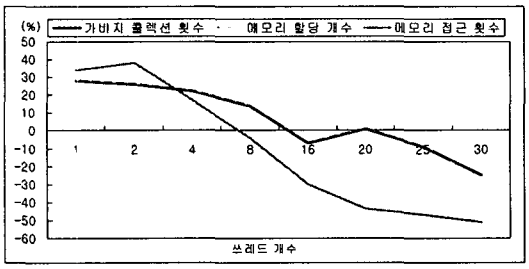

그림 7 고정 크기 메모리 할당의 효율 분석

그림 7은 고정 크기 메모리 할당의 효율 분석을 그래프로 나타낸 것으로 쓰레드의 개수가 8개 이하인 경우 고정 크기 메모리 할당은 가변 크기 메모리 할당보다 효율이 높다는 것을 보여준다.

6. 결 론

소규모 내장형 자바가상기계에서 고정 크기 메모리 할당은 쓰레드의 개수가 8개 이하 일 때 가변 크기 메모리 할당 보다 높은 효율을 가진다. 하지만 쓰레드 개수가 증가할 수록 고정크기 메모리 할당은 급격한 성능 저하를 보인다.

쓰레드의 증가는 객체와 스택 프레임의 빈번한 할당과 해제를 가지고 있다 즉 고정 크기 메모리 할당은 전체 메모리 효율이 가변 크기 메모리 할당보다 낮기 때문에 쓰레드 개수가 적고 객체와 스택 프레임의 할당 빈도가 낮은 경우에서 자바가상기계의 성능을 향상 시킬 수 있다.

참 고 문 헌

[1] J. Engel, *Programming for the Java Virtual Machine*, Addison-Wesley, 1999.

[2] Richard Jones, and Rafael Lins, *Garbage Collection: Algorithms for Automatic Dynamic Memory Management*, John Wiley & Sons, England, 1996

[3] 양희재, "simpleRTJ 자바가상기계에서 클래스 파일의 메모리상 배치", *한국정보과학회* 추계학술대회, 2002. 10.

[4] RTJ Computing, *simpleRTJ: a small footfrint Java VM for embedded and consumer device*, http://www.rtjcom.com

[5] Sun microsystem, Connected, *Limited Device Configuration specification Version 1.0a*, 2000

[6] Sun microsystem, Java 2 Platform Micro Edition(J2ME) Technology for Creating Mobile Devices White Paper, 2001

[7] 지정훈, 양희재, "내장형 자바가상기계를 위한 클래스 파일 변환기의 설계 및 구현", *한국정보과학회* 춘계학술대회, 2003. 4.

[8] 양희재, "CLDC 클래스 파일 형식에 대한 분석", 한국*정보처리학회* 추계학술대회, 2003. 10

[9] 양희재, "simpleRTJ 클래스 파일의 형식 분석", *한국해양정보통신학회 학술대회*, 6권, 2호, pp.373-377, 2002. 11.