내장형 자바 환경에서 Generational GC동작 특성

이종호°, 이인환
한양대학교
jhlee°@csl.hanyang.ac.kr, ihlee@hanyang.ac.kr

Characteristics of Generational GC in an Embedded Java Environment

Jongho Lee°, Inhwan Lee
Division of Electrical and Computer Engineering, Hanyang University

요 약

현재 무선단말기를 비롯한 많은 내장형 기기에서 자바 플랫폼을 채택하고 있다. 내장형 기기는 제공되는 메모리 용량의 제약으로 인하여 J2ME 플랫폼을 주로 사용하고 있다. J2ME 플랫폼은 CDC와 CLDC 컨피큐레이션으로 나뉘며, CDC는 JAVA 어플리케이션의 수행에 있어서 JVM보다 적은 동적 수행 메모리를 요구하는 CVM을 채택하고 있다. CDC는 CLDC에 비하여 메모리의 제약이 상대적으로 적은 셋탑박스 등과 같은 내장형 기기에서 주로 사용된다. 본 논문에서는 J2ME 플랫폼의 VM중에 하나인 CVM상에서 어플리케이션 수행에 따른 Generational GC의 특성에 대하여 알아본다. 특히, 오브젝트의 life time, 가비지 컬렉션 주기, pause time 및 young generation의 크기에 따른 동작 특성을 중점적으로 고찰한다.

1. 서 론

J2ME(Java2 Platform, Micro Edition) 플랫폼이 제공하는 두 가지 컨피규레이션은 CDC(Connected Device Configuration)와 CLDC(Connected Limited Device Configuration)이다. 메모리 관점에서 구분하면, CLDC가 CDC에 비해 동적 및 정적 메모리를 적게 요구한다. 내장형 기기의 특성상 한정된 메모리를 효율적으로 사용하는 것이 수행 성능 못지않게 중요한 요소이다. 따라서 CDC의 VM인 CVM과 CLDC의 KVM도 효율적인 메모리 사용을 염두에 두고 설계되었다. CVM과 KVM도 JVM과 마찬가지로 메모리 할당과 회수를 위하여 가비지 컬렉터를 사용한다. 기존의 연구에서 KVM과 JVM의 가비지 컬렉션에 대한 연구는 많이 진행되어 왔으나 [1, 2], CVM에 대한 가비지 컬렉션에 대한 연구는 상대적으로 부족하였다. 따라서 본 논문에서는 CVM상의 가비지 컬렉션의 동작 특성에 대하여 알아본다.

2. CDC, CVM과 가비지 콜렉터

2.1. CDC

CDC는 일반적으로 2M bytes이상의 메모리를 제공하는 내장형 기기를 위한 자바 플랫폼으로 J2SE와의 호환성을 염두에 두고 개발되었다. CDC는 J2SE API의 부분집합을 지원하며, 적은 메모리환경에서 최적화된 클래스 라이브러리를 제공한다. 또한 다양한 내장형 기기 상에서 호환성 있는 자바 어플리케이션 수행 환경을 제공하기 위하여 다음과 같은3가지의 프로파일을 제공한다.

- Foundation Profile

네트워크와 I/O등을 지원하기 위한 클래스들을 추가적으로 제공한다.

- Personal Basic Profile

Foundation Profile의 기능에 추가적으로 라이트웨이트 컴포넌트(Lightweight component) 툴킷을 구축하기 위한 구조를 제공하며, Xlet 어플리케이션 프로그래밍 모델을 지원한다.

- Personal Profile

Personal Basic Profile의 기능에 추가적으로 완전한 AWT(Abstract Window Toolkit), applet, 제한된 자바 빈을 지원한다.

2.2. CVM

CVM은 Java2 플랫폼 버전 1.3 사양(Specification)과 라이브러리와 완전한 호환성을 제공한다. CVM은 다양한 내장형 기기에 적용 가능하도록 이식성을 높였으며, 효율적인 메모리 사용 및 ROM에서 클래스 파일실행 등의 기능을 제공한다. 특히 PersonalJava에서 메모리 누수를 일으키는 원인인 conservative 가비지 컬렉션을 개선하여, exact 가비지 컬렉션을 지원한다.

2.3. 가비지 컬렉터

CDC RI(Reference Implementation)에는 3종류의 가비지 컬렉터가 구현되어 있다. 그리고 메모리 시스템을 VM과 분리하여 CVM에서 정의한 인터페이스만 구현하면 사용자가 구현한 가비지 컬렉터를 손쉽게 적용할 수 있는 구조로 되어있다. CDC RI에서 제공되는 가비지 컬렉터는 다음과 같다.

- Generational 가비지 컬렉터
- Mark-compact 가비지 컬렉터
- Semispace copying 가비지 컬렉터

CDC에서 기본적으로 generational 가비지 컬렉터가 사용되며, 가비지 컬렉션 pause time과 수행 시간에 있어서 좋은 성능을 나타낸다 [3]. Generational 가비지 컬렉터는 young과 old 2개의 generation으로 구성된다. Young generation에는 copy 컬렉션, old generation에는 mark-compact 컬렉션을 사용한다.

3. 측정 환경

측정은 RedHat 7.1 Linux, Intel Pentium CPU상에서 CDC Personal Profile RI 1.0.1버전으로 수행하였다. 가비지 컬렉터는 generational 가비지 컬렉터를 사용하였다. 힙은 8M bytes의 고정된 크기를 사용하였고, Young generation의 크기는 별도로 명시하지 않으면 기본 크기인 1M bytes를 사용하였다.

3.1. 어플리케이션

Generational 가비지 컬렉터의 동작 특성을 측정하기 위하여 벤치마크 프로그램인 Java-Olden[4]을 사용하였다. 표1은 Java-Olden에 포함된 어플리케이션의 크기, 최대 메모리 사용량과 측정 시 사용된 입력 파라미터를 정리한 것이다.

3.2. JVMPI

JVMPI(Java Virtual Machine Profiling Interface)[5]는 CVM과 프로파일 에이전트간의 인터페이스를 제공한다. 측정을 위하여 프로파일 에이전트를 작성하여, CVM에서 JVMPI를 통하여 제공되는 오브젝트 생성, 소멸, 이동 및 GC 시작, 종료 이벤트들을 통해 전달되는 데이터를 수집하였다.

4. 측정 결과

4.1. 오브젝트 Life Time

Generational 가비지 컬렉션은 대부분의 오브젝트가 짧은 life time을 가진다는 측정결과에 바탕을 두고 있다. Young generation의 크기를 너무 작게 하거나, 가비지 컬렉션을 너무 빈번하게 수행하면 너무 많은 오브젝트들이 old generation으로 옮겨지고 이는 가비지 컬렉션의 비용을 증가 시키는 요인이 된다.

Generation 가비지 컬렉터는 오브젝트의 life time이 가비지 컬렉션의 성능을 결정짓는 중요한 파라미터가 된다. 표2는 각 프로그램의 오브젝트 life time을 측정한 결과를 나타낸 것이다. 측정은 오브젝트가 참조되지 않게 되는 시점과 실제로 가비지 컬렉션되는 시점과의 차이를 최소화하기 위하여 메소드 호출이 리턴되는 시점에서 매번 가비지 컬렉션을 수행하도록 하였다.

4.2. 가비지 컬렉션 Pause Time

가비지 컬렉션에 의한 pause time은 리얼타임이나 사용자 인터페이스를 통하여 수행되는 어플리케이션의 경우 중요한 파라미터이다. CVM상의 가비지 컬렉터는 요청된 오브젝트를 할당할 수 없을 경우 가비지 컬렉션을 수행하도록 구현 되어있으나, 가비지 컬렉션 주기에 따른 영향을 측정하기 위하여 정해진 크기(100K, 400K, 800K bytes)의 메모리 할당이 일어났을 때 마다 강제적으로 가비지 컬렉션을 수행하도록 하였다. 측정 결과를 표3에 나타내었다. 표3에서 pause time은 평균값을 나타낸 것이다. Free bytes, Promote bytes 및 Forward bytes는 각각 가비지 컬렉션 수행 중에 해제된 크기, young에서 old generation으로 옮겨진 오브젝트의 크기 및 young generation의 semispace간에 옮겨진 오브젝트의 크기를 나타내며 어플리케이션 수행동안에 발생된 크기의 합산 값이다.

Bh, Health, Power와 같이 life time이 짧은 오브젝트가 많은 프로그램의 경우에는 가비지 컬렉션의 횟수가 많아 질 수록 그에 따라 pause time도 작아지나, Em3d, Perimeter, Treeadd와 같이 life time이 긴 오브젝트가 많은 프로그램의 경우에는 가비지 컬렉션의 횟수가 많아져도 pause time의 변화는 크지 않음을 알 수 있다.

4.3. Young Generation 크기

Young generation의 크기에 따른 영향을 측정하기 위하여 generation의 크기를 각각 2M, 3M 및 4M bytes로 설정하고 어플리케이션을 수행하였다. 측정 결과를 표4에 나타내었다. Young generation의 크기를 증가시키면 Bisort, Mst, Voronoi 프로그램은 가비지 컬렉션이 한번도 발생되지 않았다. 반면에 old generation으로 옮겨지는 오브젝트의 크기가 큰 프로그램인 Em3d, Treeadd의 경우에는 메모리 할당 예외가 발생되어 프로그램 수행이 중단되었다. Young generation이 커질수록 가비지 컬렉션의 발생 횟수는 줄어들고 pause time은 길어지나, 전체적인 가비지 컬렉션 수행시간은 줄어듦을 알 수 있다. 사용자와의 상호작용이 없고, 실시간성을 요구하지 않는 어플리케이션의 경우에는 young generation의 크기를 키움으로써 가비지 컬렉션에 의한 전체 pause time을 줄임으로써 수행 효율을 높일 수 있음을 알 수 있다.

5. 결론

CVM은 셋탑박스와 같이 사용자와의 인터액션이 큰 내장형기기부터 네트워크 프린터와 같이 사용자의 인터액션이 필요치 않은 내장형기기까지 다양한 환경과 어플리케이션을 고려하여 설계되었다. 측정 결과를 통하여 다양한 환경과 어플리케이션을 한정된 메모리상에서 수행하기 위해서는 가비지 컬렉션을 응용분야에 맞게 설정을 하는 것이 중요함을 알 수 있다.

6. 참고 문헌

[1] G. Chen, M. Kandemir, N. Vijaykrishnan and M. J. Irwin. PennBench : A Benchmark Suite for Embedded Java. 5th Workshop on Workload Characterization (WWC5). 2002.

[2] Martin Hirzel, Johannes Henkel, Amer Diwan, Michael Hind. Understanding the Connectivity of Heap Objects. International Symposium on Memory Management (ISMM). 2002.

[3] Sun Microsystems. CDC Foundation Profile Porting Guide. 2002.

[4] B. Cahoon Java-Olden benchmarks. http://www-ali.cs.umass.edu/~cahoon/olden.

[5] Sun Microsystems. Java Virtual Machine Profiling Interface. http://java.sun.com/j2se/1.3/docs/guide/jvmpi/jvmpi.html.

표 2. 오브젝트 Life Time

프로그램	0.1	0.2	0.3	0.4	>0.4
Bh	92.0	0.3	0.0	0.0	7.7
Bisort	12.6	11.1	11.1	11.1	54.1
Em3d	10.8	5.3	5.3	5.3	73.3
Health	84.7	1.8	0.3	0.2	13.0
Mst	12.8	11.5	11.5	11.5	52.7
Perimeter	11.3	10.2	10.2	10.2	58.1
Power	96.5	0.0	0.0	0.0	3.5
Treeadd	10.8	10.3	10.3	10.3	58.3
Tsp	66.9	1.6	1.6	1.6	28.3
Voronoi	45.7	3.4	3.3	2.6	45.0

표 1. Java-Olden 프로그램

프로그램	크기 (K bytes)	최대 사용량 (bytes)	프로그램 설명	입력 파라미터
Bh	17.1	1315724 (16%)	Solves the N-body problem using hierarchical methods.	-b 500 -s 10
Bisort	4.7	1568572 (96%)	Sorts by creating and merging bitonic sequences.	-s 100000
Em3d	7.3	6979036 (99%)	Simulates electromagnetic waves propagation in 3D object.	-n 2000 -d 100
Health	9.9	1892416 (27%)	Simulates Columbian health care system.	-l 5 -t 500 -s 0
Mst	7.5	1585404 (96%)	Computes minimum spanning tree of a graph.	-v 50
Perimeter	9.8	3830280 (98%)	Computes perimeter of quad-tree encoded raster images.	-l 16
Power	11.2	1969376 (9%)	Solves the power system optimization problem.	N/A
Treeadd	3.2	5500596 (99%)	Adds the values in a tree.	-l 20
Tsp	6.0	3516216 (51%)	Computes estimate for traveling salesman problem.	-c 60000
Voronoi	14.1	1252980 (69%)	Computes Voronoi diagram of a set of points.	-n 2048

표 3. 가비지 컬렉션 주기에 따른 영향 (100K, 400K, 800K bytes)

프로그램	GC 횟수	Pause time	Free bytes	Promote bytes	Forward bytes	프로그램	GC 횟수	Pause time	Free bytes	Promote bytes	Forward bytes
Bh	81	39	7925552	648240	4045564	Perimeter	38	258	65220	3503688	3729880
	21	93	7872132	566768	1319564		10	294	64912	3042288	3625476
	11	146	7872132	164504	693608		7	262	64912	2681840	3723108
Bisort	16	131	62132	1253968	1483240	Power	217	130	21266084	751156	899800
	4	139	61824	587772	1174044		55	177	21202608	751100	932320
	2	188	61772	587704	587704		28	238	21302060	751092	1339024
Em3d	68	139	62892	6607408	6837540	Treeadd	54	500	62160	5145152	5374408
	18	154	62584	6329576	6915952		14	552	61852	4683736	5269972
	14	115	62584	5963960	6913336		11	454	61848	4781988	5396716
Health	68	132	5857072	1102852	22194480	Tsp	67	173	3697552	2898292	3048228
	18	199	5901108	929148	6310716		17	213	3553604	2650208	2956368
	9	276	5538280	811844	3526184		12	208	3553600	2470004	3136324
Mst	16	152	62960	1253300	1481904	Voronoi	18	100	699312	871468	1683824
	5	219	62652	996540	1581980		5	129	594452	667180	1073096
	2	222	62600	586940	586940		3	181	609208	447104	1330004

표 4. Young generation 크기에 따른 영향 (2M, 3M, 4M bytes)

프로그램	GC 횟수	Pause time	Free bytes	Promote bytes	Forward bytes	프로그램	GC 횟수	Pause time	Free bytes	Promote bytes	Forward bytes
Bh	3	366	5942140	97392	251896	Perimeter	2	356	64912	2032216	2097132
	2	557	5995148	97392	198872		2	531	64912	3080792	3145708
	1	739	4045040	0	149248		0	0	0	0	0
Bisort	0	0	0	0	0	Power	10	312	19468736	751044	751656
	0	0	0	0	0		7	464	20517392	751044	751572
	0	0	0	0	0		5	618	19469184	751044	751208
Em3d	6	81	62584	5242820	7275272	Treeadd	4	508	61852	4132440	4194292
	-	-	-	-	-		2	755	61852	3083860	3145712
	-	-	-	-	-		-	-	-	-	-
Health	3	496	4907164	543448	840820	Tsp	4	293	3471592	2150972	2765968
	2	760	5031216	429484	830748		2	444	2714640	1405524	2171264
	1	1017	3595040	0	599252		2	590	3630688	1048576	3709304
Mst	0	0	0	0	0	Voronoi	0	0	0	0	0
	0	0	0	0	0		0	0	0	0	0
	0	0	0	0	0		0	0	0	0	0