International Journal of Mathematical Archive-3(9), 2012, 3410-3412
Available online through www.ijma.info ISSN 2229 – 5046

NEW APPROACH TO MANAGE OBJECTS IN ENVIRONMENT JAVA EMBEDDED

Moussaid laila*
TIM laboratory, Faculty of Ben M'sik, Casablanca Morocco

Hanoune Mustafa
TIM laboratory, Faculty of Ben M'sik, Casablanca Morocco

(Received on: 28-08-12; Accepted on: 28-09-12)

ABSTRACT

To automatically manage the heap memory for small device such as mobile phones, PDA .., Java uses the technique garbage collector (GC) which incorporates various algorithms : garbage collector by reference counting, garbage collector by marked and sweeping, garbage collector by copying ...

In this paper we present a new approach for optimizing the heap memory for small device.

This approach inherits the advantages of GC by marked and sweeping and minimizes the limitations of GC by reference. In the first part we present the J2ME environment, in the second part we describe our proposed approach and we conclude our article by prospects.

Keywords: *J2ME, Garbage collector, Embedded System; Memory; Optimization*

I. INTRODUCTION

Today embedded systems are becoming increasingly popular in various professional and personal fields, the thing that makes them a target for researchers to overcome their problems such as the standard optimization of energy and memory...

Sun proposed the platform Java 2 Micro Edition (J2ME) for small devices with low capacity, a small sizes, such as mobile phones, PDA …

The platform Java 2 Micro Edition has two configurations:

-Connected Device Configuration (CDC).

-Connected Limited Device Configuration (CLDC).

Both CDC and CLDC configurations require two virtual machines optimized respectively KVM and CVM. KVM is suitable for devices with 16/32-bit RISC/CISC microprocess-ors/controllers, and with as little as 160 KB of total memory available, 128 KB of which is for the storage of the actual virtual machine and libraries themselves.

Target devices for KVM technology include smart Wireless phones, pagers, mainstream personal digital assistants, and small retail payment terminals. The KVM technology does not support Java Native Interface (JNI). The current implementation is interpreter-based and does not support JIT (Just-in-Time) compilation [1].

To automatically manage the heap memory for small device, Java uses the technique garbage collector (GC) which is an important part of the KVM and is responsible for automatic reclamation of heap allocated storage after its last use by a Java application. Various aspects of the GC and heap subsystems can be configured at KVM runtime.

Which is based on one or more algorithms such as garbage collector by reference counting, garbage collector by mark sweep garbage collector by copying...

Corresponding author: **Moussaid laila***
TIM laboratory, Faculty of Ben M'sik, Casablanca Morocco

II. STATE OF THE ART

A. The GC by copying

The algorithm for GC by copying is based on partitioning the heap into two areas of equal size. Allowances are made by incrementing a pointer in a first zone until the end of it. The algorithm is triggered, browsing the objects in the first zone and copying them into the second. Finally, the role of the two zones is reversed.

A drawback to this copy is the need to use read barriers that are quite costly in execution time.

Indeed, the movement of objects by the GC during the execution of the program involves at least one level of indirection when reading a reference.

B. The GC by mark-sweep

A GC by mark-sweep differentiates the set of objects accessible from the rest of the heap. It is made from an initial state in which no object is marked. These objects are then traversed and marked recursively from root references say, those who have not been visited are considered inaccessible and the memory they occupy is recovered. Unlike reference counting algorithm, it does not require write barrier [3]. It further allows the recovery of cyclic structures. The additional cost in memory size required can be reduced, since only one bit is enough to store the marking information of an object.

C. The GC by reference counting

This technique associates to each object a counter, which represents the number of references on this objet. It increments for the emergence of a new reference when one of these references disappears, it is décrémenté.if the counter reaches zero, the object is referenced and its memory can be recovered. This algorithm is very simple to implement: just add a variable to all objects, and detect the onset, modification, or deletion of a reference (the overhead caused by this mechanism can generally be very reduced [2]). In addition, it allows a predictable time allocation, since the memory is recovered at the death of an object. In addition, the release is being recursively. it has a cost of performance for the worst case proportional to the number of objects in the heap. [4] However, it is irrelevant on cyclical data, It does not prevent the fragmentation of the heap thus an allocation may fail while the size of free memory is sufficient, but it is too fragmented.

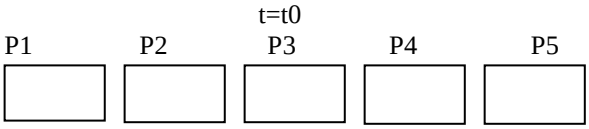
III. PROPOSED APPROACH

To overcome the problem of cyclic structures and to minimize the number of counters for the RM objects by reference counting, we present in this paper a new approach that circumvents this weakness and optimizes memory consumption: We split the memory into small pieces called "pages" from 8 to 16 kb. Our algorithm is based on the idea: the pages used extensively in the recent past have much chance of being used again in the near future. Then, each page is associated with counter. It is initialized to 0 at the beginning, each page using the MSB of the counter R takes the value 1 and in the second use the counter is shifted 1 bit to the right. Periodically at each clock pulse, typically of the order of 20 × 10-3 seconds, all pages are scanned. Example below (figure.1) shows the result after 5 timer interrupts.

After 160 × 10-3 seconds, all pages are scanned using the RM with mark-sweep. More objects used pages 1, 2,4, are defragmented (Figure 2) and the least used pages are not defragmented 3.5

Object	R	counter
1	1	10110000
2	1	10001000
3	0	00010000
4	0	01011000
5	0	00100000

Figure1: situation after five timer interrupts



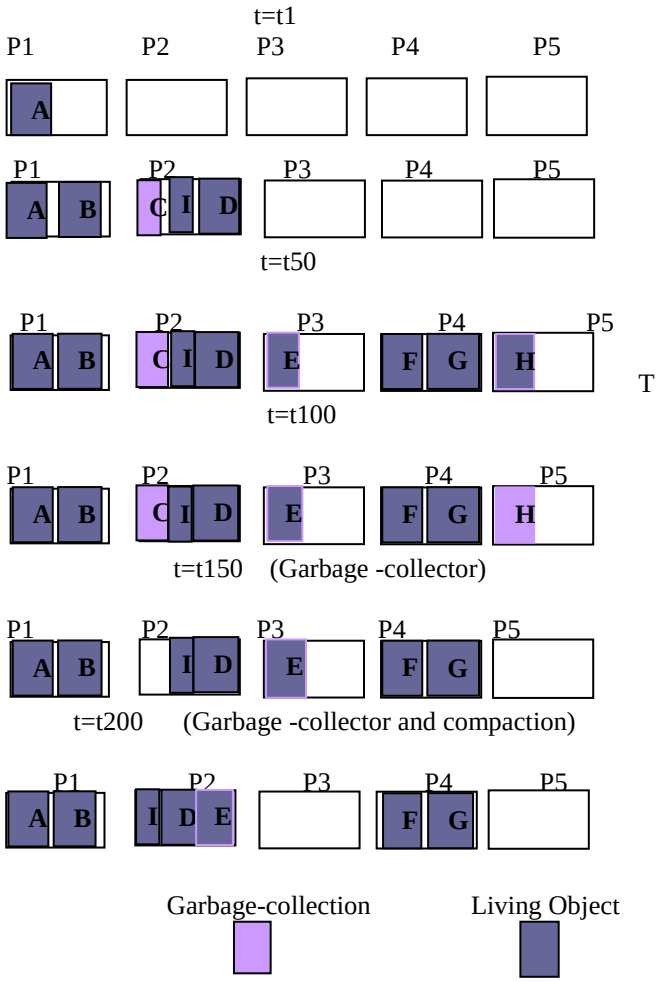


Figure 2: Operation of the GC with Mark-sweep and compaction

IV. CONCLUSION

This algorithm is simple to implement: we assign a counter to each page and not an object as the case of the GC by reference, so it overcomes the problem of cyclic structures recognized by The GC by reference counting. But the time enforcement is closly because it requires at least two courses of all objects.

In our work future, initially we will consider our proposal to improve while avoiding the drawbacks of the algorithm, and generalize our algorithm to the second profile j2me CDC in a second time.

V. RÉFÉRENCES

[1] Richard Jones and Rafael Lins. Garbage collection: algorithms for automatic dynamic memory management. John Wiley & Sons, Inc., New York, NY, USA, 1996.

[2] Antony L. Hosking, J. Eliot B. Moss, and Darko Stefanovic. A comparative performance evaluation of write barrier implementation. In OOPSLA ’92 : conference proceedings on Object-oriented programming systems, languages, and applications, pages 92–109, New York,NY, USA, 1992. ACM Press.

[3] G. Chen, R. Shetty, M.Kandemir, N. Vijaykrishnan, M. J. Irwin, and M. Wolczko.Tuning garbage collection in an embedded java environment. In HPCA '02: Proceedings of the Eighth International Symposium on High-Performance Computer Architecture (HPCA'02), page 92. IEEE Computer Society, 2002.

[4] Richard L. Hudson and J. Eliot B. Moss. Incremental collection of mature objects. In IWMM ’92: Proceedings of the International Workshop on Memory Management, pages 388–403, London, UK, 1992. Springer-Verlag.

Source of support: Nil, Conflict of interest: None Declared

