Improved Adaptive Generational Garbage Collection Algorithm for Java Card

Yang Fubiao[1]
[1] Department of Computer
Guangdong University of Technology
Guangzhou 510006, China
youngfb@qq.com

Li Daiping[2]
[2] Department of Computer
Guangdong University of Technology
Guangzhou 510006, China

Abstract—The memory size of Java is very limited, while the Java Card specification does not specify the garbage collection mechanism, resulting in the utilization of system memory resources is not efficiently. In order to solve this problem, we adopted cross-layer design ideas, took Flash physical storage characteristics and the features of Java Card object allocation into consideration, designed the generational adaptive mark-sweep-copy garbage collection algorithm for Java Card. In this paper we use small part of RAM resource, effectively reduced the number of write operation of Flash, and provide a good balance of wear and then prolong the life of the Flash. The Algorithm of the garbage collection significantly improved the system performance.

Keywords-component; mark-sweep-copy;Flash; Object allocation; wear balance; Java Card

.

I. Introduction

Java language allows programmer to explicitly allocate memory resource without consideration the recycle and release work of memory, this task is automatically completed by The Java Virtual Machine Garbage collector. Java Card, by contrast, has not explicitly stipulated and presented the memory recycle and release in the technical specification, so need designer to design project with the actual situation. In this paper, according to the Java Virtual Machine requirement of realistic pictures, by adopting the idea of cross-layer design, integrated flash physical medium and the features of distribution of Java Card Objects, designed Java Card garbage collection generational adaptive tags-cleaning-replication algorithm. Algorithm that takes up a little bit RAM Space and fully takes the characteristic of the Flash into account, can effectively detect and recycle the garbage objects in system, implement memory fragments remove, increase the Java Card virtual Machine execution efficiency, and by reducing the operation to Flash to provide good balance of wear and prolong the service lift of the flash.

.

II. Memory Allocation and Object Management

A. Memory Allocation

The Java Card Storage structure consists of ROM, RAM and FLASH. In Java Card system the code section of JCRE (JCVM, API class and other software) resident in ROM, Applet programs also storage in ROM. RAM as temporary data storage, the stack and heap allocated from the RAM in system running time besides the middle results, local variables and some native methods such as encrypt algorithm generated final and middle results. Permanent data and download Applet Class resident in non-volatile flash memory. Java Card system memory structure shown in Figure 1:

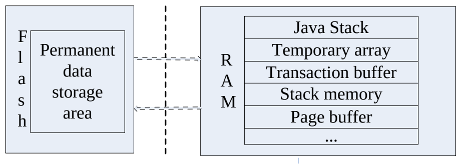

Figure 1： Java Card storage allocation structure

In Java Card System, memory is very limited, so Java Card creates application instance and related data by new operation code that storage in flash memory. In the Java Card technologies at these days, RAM usually is use to storage temporary and process methods invoking, and on the other hand, still need to keep permanent data into flash memory when power loss. Java Card has special command to allocation memory in flash for Java instance objects, and Java virtual machine automatically recycle memory space is not same, if no reference directed to the data that resident in flash, there has not any command to release the allocated space, then Java card virtual machine would be allocated some garbage data in flash memory. So the Java Card, in deed, needs garbage collection algorithm to deal with the memory space recycle.

B. Java Card Objects Management Architecture

Thanks to the need of balance of Flash wear, Java Card Objects need flexible methods to move, and avoid writes frequently on a flash area lead to the flash block rapidly loss of work. So in the Java Card object management, does not directly address pointer is adopted to improve the positioning of an object, but the table structure, by the object management list, all the object's address and references.

The object management list is a pointer array, save in the fee volatile area. The contents of the array is the physical address of each object, and the array index is the reference object. Object reference zero in Java Card system used to represent a Null reference (Null), and the table of the object management list number starting from one. Get an object reference, the address of the object is obtained by a lookup table command can be directly. To be able to access the object.

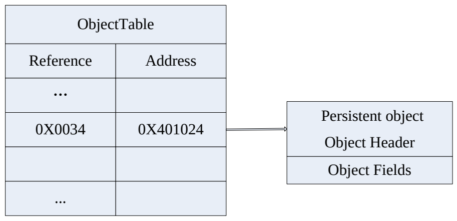

Figure 2: The Java Card object management system

Java Card object's physical address is just at the time of access objects used by the virtual machine, when show exclusive relationship all use object references. In Java Card specification use 16 bits to define a application, therefore Java Card can hold at most 2 ^ 16 – 1 objects that presented space is large enough for Java Card. In each object head, and holds a pointer to the object of the class reference, can generate the class of the object, through the reference saved the size of the object instances in the class, as well as the reference domain starting position and number, the garbage collection algorithms in system can be determined through object domain traversal of the object reference. Therefore, in the garbage collection algorithm design can through for the object's viability detection that distinguished from active object to garbage objects.

III. Design of Garbage Collection Algorithm

Garbage collection algorithm is divided into object detection and garbage collection, the garbage detecting algorithm testing objects viability to distinguish whether an object should be recycled. The traditional garbage detection algorithm is reference counting and reference tracking. But the reference count is difficult to solve the problem of reference each other, and therefore not applicable Java Card system; And reference tracking algorithm of recursive operation for Java Card system of RAM resources limited is a test, so this article design a garbage detection mechanism based on bitmap.

A. Garbage Detection Based on BitMap

RAM with fast ability to read and write in Java Card memory space, so this paper in the garbage collection phase allocated one unused area in virtual machine stack is used to create tag bitmap. Bitmap need space and system of the largest number of objects, because the object reference indexes corresponding with the object management list, and the biggest object reference not more than 1000, so can link the object with the serial number of the bitmap. We, at first, through a BitMap detect stack space, the virtual machine by the stack pointer to the top of the stack refers to the space between Stack_END is divided into three regions, followed by the virtual machine implementation of the bytecode space, the space that save the BitMap and deposit delay processing table space of Java Card. That shown in Figure 3.

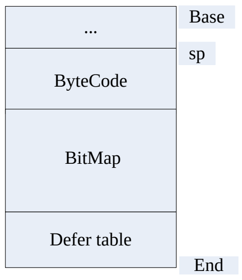

Figure 3: Java Card garbage detection algorithm mark bitmap

Because of the limitation of chip physical space can't in the limited system pace for deep JCVM garbage detection recursive algorithm. So we use the specified depth of recursion marking method, when you arrive at a certain depth, algorithm will be treated as the current object reference in a delayed treatment in the table, in the depth of the recursion meet again when recursive tag on this object. In BitMap, the length of the delay treatment table by the biggest free space in the stack and jointly determine the size of the BITMAP, in this paper, the system delay processing of the set table for eight, in order to improve the efficiency of garbage collection. Due to the limitation of system resources, we recursion depth not more than 3, if more than 3, then we will defer the current object into the Defer table. Garbage detection algorithm flow chart shown in Figure 4.

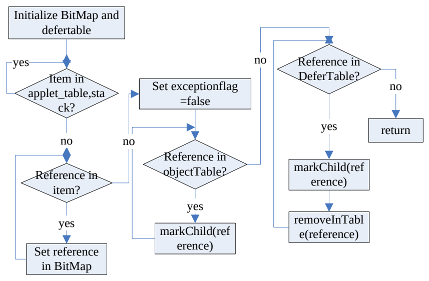

Figure 4: mark algorithm structure chart

So when it marks that directly mark the root object then traverse the entire object table, if an object is the active object, then the recursive tag sub-objects, when the object reference number in the current mark in the front of the object, then for child objects are recursive tag; If the child object references in the back of the current object reference, so we just marked, when then traverse to the child object to in-depth recursion; If child objects have been marked, then skip this child objects. Through this process can mark all the active objects in heap space. System will to recursion all object that tags, when running to that object, to mark its child objects, so that we can in a cycle effectively complete recursive calls of all child objects. After the current object recursive tag 3 layers, if delay object table is not empty, you can recursively tag delay object in the table, when after the completion of the delay object table tag, tag algorithm is restarted. Because before has marked the part of the tag does not need to begin from the root object, thus the recursion depth has a lot to reduce than last time, thus the object which last so can smoothly in the tag. End tag after as well as the completion of the garbage detection algorithm, the system can use the tag bitmap to separate active objects and garbage objects. From the bitmap marking can get object reference number. Reference number can be used to get the object's physical address, which can perform garbage collection strategies of the system.

B. *Garbage Collection Policy*

According to the characteristics of the Java Card and the features of Flash, Flash objects can be divided into two generations, the current distribution of objects in the frame for the new generation, and other objects in the frame of the old generation. Under the new generation of frame, if launched the garbage collection algorithms, the algorithm only marked-clear. Page frame in the new generation is full, the system automatically start garbage collection, performs the mark-replication algorithm, the current page frame survived several garbage collection active objects is copied to the old generation, the current page will be marked failure, and be erased by the sector. Within the page frame, page distribution carried out in accordance with the First principle of (First-fit), starting from the current page frame, if found the first to meet the requirements of distribution of continuous space cease to search. The use of a new generation of page frame must keep the frame number of the page in order, that is to say, the old page frame in the next will be used as a new generation, which the old generation of objects in the frame can be tight replication, eliminate the old generation of fragments, but also provides the old generation takes up the Flash page frame wear balance.

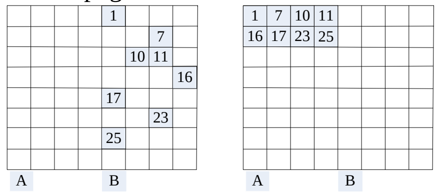

Figure 5. Garbage replication algorithm execution before and after

In Figure 5, initial frame A to the current old generation page frame, B as A new generation of page frame. The distribution of the object in B frame in accordance with the principle of adaptation in the first time. If B frame under the top and start the garbage collection algorithm, then performs the mark-cleaning the algorithm, the active object tags in memory, and later, at the end of the tag that garbage object will be removed from the system. If B frame can't meet a new object allocation request, so that is to say B frame is full, the system starts the garbage collection algorithms to perform the tag-replication algorithm. Page frame B cleaning experience several garbage collection marks, form the active object to the left as shown in figure 5. Current need to assign an eight page length for an object, and the new generation can not meet the requirements, thus system startup garbage collection, use the tag-replication strategy, first determine the activities in the tag in the bitmap memory objects, the current active object for the next system start replication strategy, you need to copy active objects in current page frame to the old page frame, the current old frame is A, in the old generation distribution also uses the first adaptation principle, the new generation of each object is copied to the old generation page frame in order to save. Through to the success of object replication, page frame after the copy of the situation of the active objects as shown in figure five on the right, thus marking-sweep algorithm can efficiently collect the garbage objects in the page frame, but will form debris inside the page frame. Tag-object replication algorithm can effectively use generational features, will be through a lot of garbage collection long-lived objects to reproduce, eliminate the long-lived objects between pieces, efficient use of Flash resources. For wear and balanced, the tag-copying algorithm, the whole frame of object is copied to the new page frame, when the page frame A saved the oldest object, and may form some pieces in multiple garbage collection. Tag-replication algorithm starts, sets the page frame H to the old generation page frame, and A active objects are copied to the H, finally release the frame of A new generation, A free and become the current old generation page frame.

C. Algorithm Analysis

Because the system always has a complete spare frame, it ensures the tag-replication algorithm can run smoothly, but after using this algorithm Flash utilization rate in the project limit is 87.5%. Would actually be more small, in order to avoid system at high load time memory allocation disequilibrium, adaptive adjustment algorithm according to the load, when the Flash utilization rate more than 80%, degradation of tags-cleaning algorithm, in addition to the current the latest old generation page frame, use the first adaptation in the global method to allocate resources; When Flash utilization rate as low as 75%, system will again start mark-replication algorithm.

Because the Flash of utilization rate is low, a new generation of stored in the original object is less, so can allocate more objects, so the new generation space was soon exhausted, which leads to frequent tag replication process. In order to avoid excessive duplicate action, we set the 80% mark-replication algorithm stop threshold, Flash utilization rate than the threshold, the degradation of tags-cleaning the garbage collection algorithm. At the same time set up the 75% mark-recovery of replication algorithm threshold, when the Flash utilization rate is lower than the threshold, garbage collection algorithms to restore tag-copy.

The time characteristic of the Flash system is Write sensitive, the system starting in the early stage, Flash objects in less, write less time-consuming and less garbage collection algorithm execution times. In subsequent applications, the system objects are created, use and remove, at the same time. about 5% system objects will perform frequent write operation, the utilization rate of the system was between 30% ~ 50%.Through for the write operation time of the Flash system, we find that the write operation takes uniform and smooth, the write operation time was good capitation to every sector of the Flash, thus make Flash system has good wear balance ability.

IV. Conclusions

Classic garbage collection algorithm has been introduced in this paper, analyzes the JCVM storage structure and object distribution, was proposed based on adaptive generational type tag-cleaning-copy of the Java Card garbage collection algorithms, the algorithm designed combined with the flash medium physical characteristics and the requirement features of Java Card, make whole Java Card to achieve optimal performance. Through example analysis, the algorithm implementation good Java Card garbage collection of an object, good for Flash improves the wear is balanced, will write the number of times and time sharing in Flash each page frame, page also raised its balanced ability and better efficiency.

References

[1] IM Y. JUNG, SUNG I. JUN, KYO I. CHUNG. A Persistent Memory Management in Java Card . WSEAS Transactions on Systems. Vol. 2, no. 1, pp. 160-165. Jan. 2003.

[2] A.J.R. Aendenroomer and S. Huang. Dynamic Flash-memory Allocation for Smartcards: how to cope with limited space (in a short life). IEEE Conference Publications.2007, Page(s): 835 – 840.

[3] Cap, C.H.; Maibaum, N.; Heyden, Lars. Extending the Data Storage Capablities of a Java-based Smartcard. IEEE Conference Publications. 2001 , Page(s): 680 – 685.

[4] Sun Microsystems, Inc., Virtual Machine Specification, Java Card™ Platform, Version 3.0.1, Classic Edition. May, 2009.

[5] Sun Microsystems, Inc., Java CardTM 2.2 Runtime Environment (JCRE) Specification, June, 2002.

[6] Ding Yu-Xin, Cheng Hu. Implementation of the precise Garbage Collection For Java Virtual Machine. CHINESE J. COMPUTERS[J]. 1999,22(11) :1228-1232.

[7] Zhou Zhiming, Understanding the JVM Advanced Features and Best Practices. China Machine Press. 2011-06.

[8] Guang Hu, Zhilei Chai, Shiliang Tu. Memory access mechanism in embedded real-time Java processor. Computer and Automation Engineering (ICCAE), 2010 The 2nd International Conference on[C]. 2010 , Page(s): 786 – 790.

[9] ZHANG Shi-jun, GAO Shu, Improved Garbage Collection Algorithm for Incremental JVM, Computer Engineering[J]. 2012-01.