Memory Management in Android

Mr. Jignesh Jain
Student, Bachelor of Engineering
K. J. Somaiya College of Engineering
Mumbai, India.

Mr. Meet Satra
Student, Bachelor of Engineering
K. J. Somaiya College of Engineering
Mumbai, India.

Mr. Pratik Kumar Jain
Student, Bachelor of Engineering
K. J. Somaiya College of Engineering
Mumbai, India.

Prof. Era Johri
Assistant Professor, Department of IT
K. J. Somaiya College of Engineering Mumbai, India.

***Abstract*— In mobiles, primary memory is a paramount resource and thus utilization of this resource should be optimum. As RAM (Random Access Memory) is a "costly" affair, it should be utilized at optimum to hold the data that is directly accessible by the processor. Researchers are working in the computer fraternity to maximize the performance of this resource. In this paper, we discuss the current challenges which the programmers faced while programming the RAM in mobile phones and propose some optimum solutions in order to solve the issues. We discussed various factors like Object Creation, Use of Primitive types, Static Constants, Hidden Cost of Abstractions, data, containers, caching policies and services to optimize the RAM performance. A comparison regarding the performance of the two different operating system mainly Android and iOS is also stated.**

***Keywords*--Objects Creation, Use of Primitive types, Static Constants, Hidden Cost of Abstractions, data-containers, caching policies, services, Android and iOS.**

I. INTRODUCTION

Primary storage that holds data in the mobile device is known as RAM (Random Access Memory).RAM acts as a container that keeps things ready for the CPU. It is used differently by different operating systems and is also infinitely (almost) rewritable and very fast. When we start a particular application on the phone, the application gets loaded from the device memory into the RAM. This same process is observed in computers and the main reason behind doing this is because it increases the access speed and reduces the access time. It was designed to be used fully by Android, Mac OS and Ubuntu. The widgets, system , different tools, graphics are first loaded in the ram and then the available ram is used for loading different applications. Load the paramount applications into the ram and keep it there until the RAM gets completely filled up. If new applications needs to be occupied by the ram and there is no space available then flush the existing RAM and now there will be space left for a new application to occupy the RAM. More the system is being used the performance of the RAM gets better and better.

If you have 50 applications installed on your mobile devices, but there are few applications which are being used again and again. The Continuous opening and closing of specific applications will be stored in the RAM because these applications are frequently used. It takes a long time to load an application from the secondary storage which in return consumes more battery and it affects the execution time. Operating system utilizes maximum RAM and hence that is the reason why you can't free it up 100%. Everything is cleared in the RAM as you restart your mobile device. Performance is determined by how the operating system utilizes the RAM. When the OS starts up, all the icons and widgets (and other graphical items) are rendered and this is the thing the phone is doing before it's responsive. Even if you are viewing them or not, those pictures, files of icons/widgets/etc. will be persistently in the RAM as long as the OS has booted; and 'cleaning' the RAM should not cause the icons to reload.

The RAM is generally rapidly occupied and freed. Occupancy is never fixed even if you get a fixed RAM size (i. e. When you are performing multiple operations, the RAM gets filled and when no application is running it automatically gets cleared). In android, some or the other application is always running and hence you will never find RAM utilization as null.

II. THEORY

When an application runs on the phone, it has been loaded from the device memory into the RAM. It does this in the same way, the computer does, but its access speed is much faster greater in RAM. Clearing apps in the RAM depends on the Smartphone, but most background apps will automatically be loaded back into the RAM leading to decrease in free space straight away. Hence Memory optimization should be done for smooth running of the device.

Here in this paper we provide some solutions in order to optimize memory:

A. Unnecessary Object's Creation should be avoided:

Since Garbage collection is not free, creating more objects in the application leads to frequent forced running of the garbage collector. This utilizes resources which are required to boost user experience and responsiveness. Creating temporary objects will hurt. A large number of small allocations can lead to heap fragmentation.[1]

B. Use of Primitive types over Object:

Since a boxed object of type Integer occupies four times memory as primitive, int-primitive types must be used wherever possible. Similarly, a boxed object of type Boolean occupies much more memory than the primitive Boolean object. Hence, as a boxed object occupies much more memory than the primitive type, primitive object should be used over a boxed object.[1]

C. Advantages of Static Constants over Enums:

The use of enums should be avoided and rather static-final variables should be used for constants. This is because enums require more than twice as much as memory as static constants.[1]

D. Creation of Unnecessary/Inner Classes should be avoided:

In Java, all classes, including anonymous inner classes which create an object and write access or methods internally, uses about 500 bytes of code.

E. Hidden Cost of Abstractions:

A code with multiple layers of abstraction is considered to be good practice for object oriented languages. An increase in the code will lead to more execution time which will directly increase the memory it is going to take. So multiple layers are avoided.Abstraction should be used only at places where it provides significant benefit.[1]

F. Services should be used efficiently:

In order for applications to run in background, services are useful but they are expensive. A service should be stopped from running unless it is absolutely required. The life-cycle of a service can be managed efficiently by using IntentService, which finishes itself once its work is done. For rest of the services, stopService() or stopSelf() methods should be called after the work is done which is the responsibility of the application developer.[1]

G. Releasing Memory when User Interface Becomes Hidden:

While navigating through activities, when onPause() and onStop() callbacks are made, resources associated with that activity should be released. The resources here are either a network or database connection or a broadcast receiver, etc.[1]

When a user navigates between applications and if the User Interface components of the previous applications are in background then the applications receives an onTrimMemory() callback in the activities in background, where the Android System needs to kill any cached process to reclaim memory for running some other process. The TRIM_MEMORY_UI_HIDDEN LEVEL SHOULD BE LISTENED and the User Interfaces should be released. For Example: Textviews, ImageViews, etc.[1]

H. Using Optimized data containers:

Optimized Data containers and support libraries like Sparse Array, Sparse Boolean Array and Array Map are provided by Android .These containers can act as a substitute for Hash Map where the keys are of type primitive like int, boolean. As Hash Map requires an Integer object to store integer values, it occupies more memory than is actually required in case where more number of entries are needed in the map. When compared to Hash Map, ArrayMap consumes less memory but it is slower in access. Hence it must be used only when the numbers of elements are less than 100.
Some Examples of Optimized data containers which can be used in place of their HashMap equivalents:

Table I

Hashmap Types

Hash Map	Array class
<Integer,Object>	Sparse Array
<Integer, Booleant>	SparseBooleanArray
<Integer,Integer>	SparseIntArray
<Integer, longt>	SparseLongArray
<Long,Object>	LongSparseArray

Also use of raw arrays like int[] should be done in performance-critical sections of code or where working with hundreds or thousands of element at a time if possible.[1]

I. *Using a cache:*

A cache allows objects to be reused which are expensive to create. After loading the object into memory, it is treated as cache for the object. Android operating system has a LruCache class which is a part of API 12. This class provides cache implementation based on least recently used cache (LRU cache). LRU cache is used to keep track of the members. It consists of a predefined size which once exceeded, eliminates the items which was last accessed. This behavior is showed in the diagram below.[5]

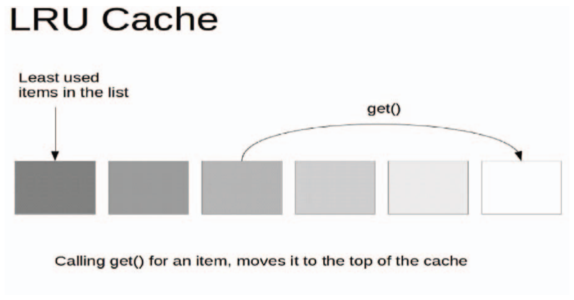
LRU Cache
Least used items in the list
get()
Calling get() for an item, moves it to the top of the cache

Fig.1. LRU Cache Implementation [5]

J. Cleaning of cache:

According to API 14 of Android, onTrimMemory() method gets overridden. When an Android system needs resources for processes in the foreground, it calls this method to clean up the memory so that the memory can be utilized. [5]

III. COMPARISON

Following table compares the performance of the two operating systems:

Table II

Comparison between android and iOS[6]

	Android	**iOS**
Uses Java	Yes	No
Garbage Collection	Present	Absent
RAM Requirement	Twice then iOS	Normal
Battery Requirement	More	Normal

IV. CONCLUSION

In this paper, we discussed various approaches such as object creation avoidance to avoid garbage collection, use of primitive types over objects as it requires less memory, using of static constants over enums as it requires half the memory, freeing of inner classes, the hidden cost of abstractions in terms of multiple layers, efficient use of services, use of optimized data containers in case of a large number of entries, caching policies to be used, services and some programming styles modified in order to reduce the consumption memory by processing maintaining high responsive time to users.

V. REFERENCES

[1]https://hsc.com/Blog/Best-Practices-For-Memory - Optimization-on-Android-1.

[2]http://www.androidcentral.com/ram-what-it-how-its-used- and-why-you-shouldn't-care

[3]http://forums.androidcentral.com/samsung-galaxy-s4/304704-why-does-my-phone-use-so-much-ram.html

[4]http://forum.le.com/in/index.php?threads/ram-what-it-is-how-its-used-and-why-you-shouldnt-care.926/

[5]http://android.stackexchange.com/questions/80867/understanding-ram-sd-card-and-device-memory-of-an-smartphone

[6]http://www.vogella.com/tutorials/AndroidApplicationOptimization/article.html

[7]http://www.cultofmac.com/303223/ios-twice-memory-efficient-android-heres/

[8]http://indiatoday.intoday.in/technology/story/ios-is-vastly-more-efficient-than-android/1/401334.html

[9]
http://stackoverflow.com/questions/14680953/how-to-finish-a-destroyed-activity

[10]http://www.sitepoint.com/understanding-the-life- cycle-events-of-activity-in-android/

[11]http://stackoverflow.com/questions/16185069/activity-
instance-still-exists-even-after-ondestroy-is-called