Android memory management

1. **Prerequisite**
   - 1.1 Understanding of Android init process & about Zygote process
   - **1.2** Understanding of Dalvik virtual machine
   - **1.3** Memory mapping & pagging concepts
   - **1.4** What is swapping
   - 1.5 Types of device memory
     - **Dynamic RAM**
       Dynamic Memory (also known as RAM) is used by applications that run when the device is turned on. The amount of Dynamic Memory influences how many applications and operating system services can run at the same time. The Android operating system automatically closes applications and services that are not being used.
     - **System RAM**
       System Memory (also known as "System partition" or "/system") is used for the Android OS and for most applications that are pre-loaded from the factory. This type of memory is normally locked, and can only be changed through a firmware upgrade
     - **Internal storage**
       Internal Storage is memory used as" working" memory. It can be compared to the C: drive on a PC or to the startup disk on a Mac.This type of memory is used to store all application downloaded from the Google Play™ Store (and other sources) as well as their settings and data (such as emails, messages and calendar events, for example). All applications have an allocated area which no other applications can access and where
     - **SD card**
       SD Card (known as "/ext_card" from a programmer's point of view, or by other names in other Android products) is the name for the removable SD memory card in all 2013 Sony Mobile products.

2. **Android memory management**
   - Android does not offer swap space for memory,
   - It does use paging and memory-mapping (mmapping) to manage memory. This means that any memory you modify—whether by allocating new objects or touching mmapped pages—remains resident in RAM and cannot be paged out.
   - So the only way to completely release memory from your app is to release object references you may be holding, making the memory available to the garbage collector. That is with one exception: any files mmapped in without modification, such as code, can be paged out of RAM if the system wants to use that memory elsewhere

Sharing Memory

In order to fit everything it needs in RAM, Android tries to share RAM pages across processes. It can do so in the following ways:

- Each app process is forked from an existing process called Zygote. The Zygote process starts when the system boots and loads common framework code and resources (such as activity themes). To start a new app process, the system forks the Zygote process then loads and runs the app's code in the new process. This allows most of the RAM pages allocated for framework code and resources to be shared across all app processes.
- Most static data is mmapped into a process. This not only allows that same data to be shared between processes but also allows it to be paged out when needed. Example static data include: Dalvik code (by placing it in a pre-linked .odex file for direct mmapping), app resources (by designing the resource table to be a structure that can be mmapped and by aligning the zip entries of the APK), and traditional project elements like native code in .so files.
- In many places, Android shares the same dynamic RAM across processes using explicitly allocated shared memory regions (either with ashmem or gralloc). For example, window surfaces use shared memory between the app and screen compositor, and cursor buffers use shared memory between the content provider and client.

Allocating and Reclaiming App Memory

Here are some facts about how Android allocates then reclaims memory from your app:

- The Dalvik heap for each process is constrained to a single virtual memory range. This defines the logical heap size, which can grow as it needs to (but only up to a limit that the system defines for each app).
- The logical size of the heap is not the same as the amount of physical memory used by the heap. When inspecting your app's heap, Android computes a value called the Proportional Set Size (PSS), which accounts for both dirty and clean pages that are shared with other processes—but only in an amount that's proportional to how many apps share that RAM.
- The Dalvik heap does not compact the logical size of the heap, meaning that Android does not defragment the heap to close up space. Android can only shrink the logical heap size when there is unused space at the end of the heap. But this doesn't mean the physical memory used by the heap can't shrink. After garbage collection, Dalvik walks the heap and finds unused pages, then returns those pages to the kernel using madvise. So, paired allocations and deallocations of large chunks should result in reclaiming all (or nearly all) the physical memory used.
- To maintain a functional multi-tasking environment, Android sets a hard limit on the heap size for each app. The exact heap size limit varies between devices based on how much RAM the device has available overall.

Switching Apps

- Instead of using swap space when the user switches between apps, Android keeps processes that are not hosting a foreground ("user visible") app component in a least-recently used (LRU) cache. For example, when the user first launches an app, a process is created for it, but when the user leaves the app, that process does *not* quit. The system keeps the process cached, so if the user later returns to the app, the process is reused for faster app switching.

- If your app has a cached process and it retains memory that it currently does not need, then your app—even while the user is not using it—is constraining the system's overall performance. So, as the system runs low on memory, it may kill processes in the LRU cache beginning with the process least recently used, but also giving some consideration toward which processes are most memory intensive

Reference

[1]Web reference https://developer.android.com/training/articles/memory.html

[2]Webreferencehttps://developer.android.com/guide/components/processes-and-threads.html

[3]Web reference http://talk.sonymobile.com/t5/FAQ/Memory-in-Android-devices/td-p/348064