



# Memory management in Android

By G3A:-

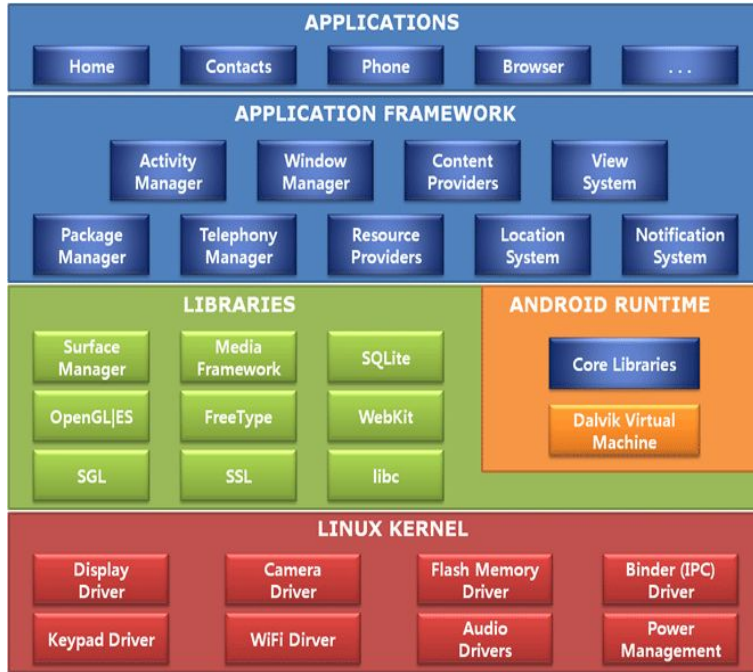
<u>Name</u>	<u>Roll Number</u>	<u>Slides explained</u>
Abhishek Agrahari	190123066	3-5
Manish kumar	190123067	6-9
Vivek Kumar	190101100	10-12

# Table of Contents



- ❖ Overview of memory management
- ❖ Types of memory
- ❖ Memory Pages
- ❖ Garbage Collection
- ❖ Low Memory Management
- ❖ Sharing Memory
- ❖ Calculating Memory Footprint
- ❖ Memory Leaks

# Overview of Memory Management



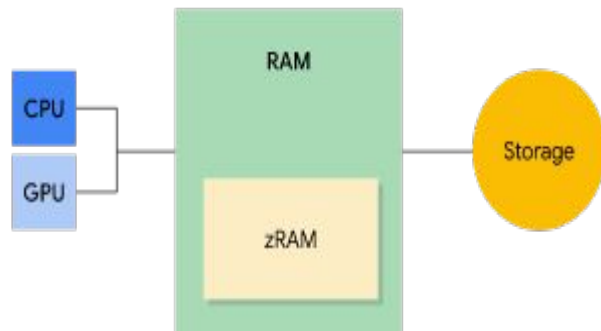
- ❖ Android is a Linux based operating system.
- ❖ Like Java and .NET, Android uses its own run time and virtual machine to manage application memory.
- ❖ Dalvik is a virtual machine that executes all the applications.
- ❖ Each Android application runs in a separate process within its own Dalvik instance.
- ❖ The Android Runtime (ART) and Dalvik virtual machine use paging and memory-mapping (mmapping) to manage memory.



Architecture of Google's Android

# Types of Memory

- Android devices contain three different types of memory: RAM, zRAM, and storage.
- Mobile devices generally use flash memory rather than more spacious hard disk for their persistent storage.
- Only limited number of writes that flash memory can tolerate before it becomes unreliable.
- On Android, storage isn't used for swap space like it is on other Linux implementations. zRAM is a partition of RAM that is used for swap space.
- RAM is the fastest type of memory, but is usually limited in size.
- Everything is compressed when placed into zRAM, and then decompressed when copied out of zRAM.
- Storage contains all of the persistent data such as the file system, libraries etc.



# Memory Pages

RAM is broken up into pages. Typically each page is 4KB in size. Pages are classified as -

1. **Used Pages (dirty pages)** - This Memory is not backed by a file on storage. Therefore these pages cannot be deleted from memory but can be compressed and moved to zRAM for increasing the free memory.
2. **Cached Pages (clean pages)** - They contain an exact copy of a file (or a portion of file) that exists in storage. These pages can be deleted to increase free memory.
3. **Free Pages** - These are unused RAM.

## Used Pages

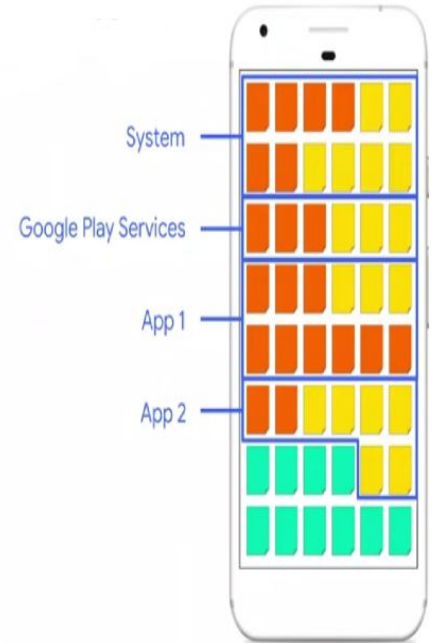
Memory actively being used.

## Cached Pages

Memory backed by device storage.  
Can be reclaimed if needed.

## Free Pages

Memory not currently being used  
for anything

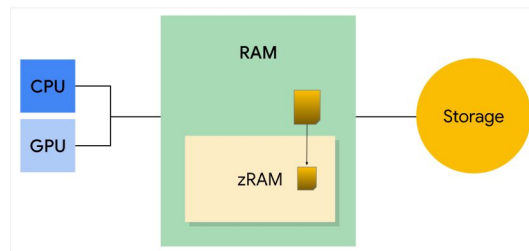


# Garbage Collection



- ❖ The mechanism for reclaiming unused memory within memory environment is known as *garbage collection*.
- ❖ It has two primary goals: find data objects in a program that cannot be accessed in the future; and reclaim the resources used by those objects.
- ❖ Android's memory heap is a generational one, meaning that there are different buckets of allocations that it tracks, based on the expected life and size of an object being allocated. For example, recently allocated objects belong in the *Young generation*. When an object stays active long enough, it can be promoted to an older generation, followed by a permanent generation.

# Low Memory management



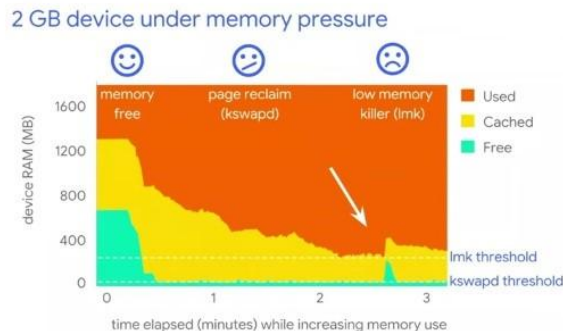
## Kernel swap daemon(kswapd)

- ❖ kswapd is part of the Linux kernel, and converts used memory into free memory.
- ❖ When free memory falls below the low threshold, kswapd starts to reclaim memory. Once the free memory reaches the high threshold, kswapd stops reclaiming memory.
- ❖ Working:
  - kswapd can reclaim clean pages by deleting them
  - If a process tries to address a clean page that has been deleted, the system copies the page from storage to RAM. This is known as *demand paging*.
  - kswapd can move some dirty pages to zRAM, where they are compressed.
  - If a process tries to touch a dirty page in zRAM, the page is uncompressed and moved back into RAM.

# Low Memory management





















## Low-memory killer(LMK)

- ❖ Many times, kswapd cannot free enough memory for the system.
- ❖ If the amount of free memory falls below a certain threshold, the kernel starts killing processes to free up memory. It uses (LMK) to do this.
- ❖ To decide which process to kill, LMK uses an "out of memory" score to prioritize the running processes.



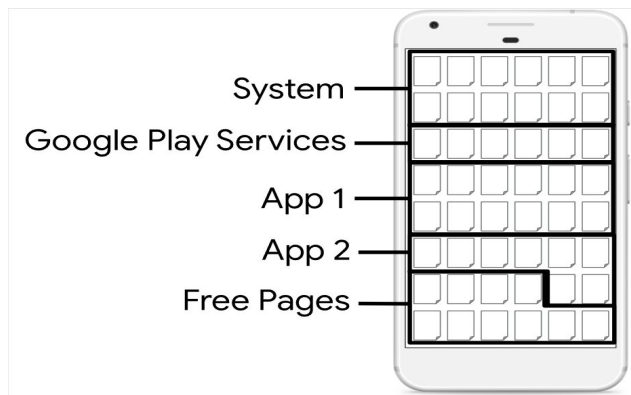


- ❖ Processes with a high score are killed first. The following table lists the LMK scoring categories from high-to-low. Items in the highest-scoring category, in row one, will be killed first:

Background apps	    
Previous app	
Home app	
Services	  
Perceptible apps	  
Foreground app	
Persistent	     
System	<code>system_server</code>
Native	<code>init kswapd netd logd adbd installd</code>

# Calculating memory footprint

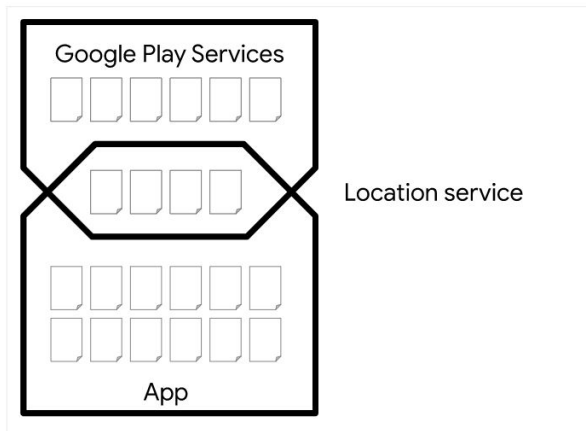
- ❖ Determine how much memory is used by an app.
  - Low memory management
  - Restricting app memory
- ❖ Kernel keeps track of all memory pages in system.
- ❖ Need to account shared pages
  - Apps that access same service or lib will be sharing mem pages.
  - Google play service and a game app may be sharing location service.
- ❖ To determine memory footprint, any of these metrics can be used.
  - **Resident Set Size(RSS)**: The no. of shared and non-shared pages used by the app. Tracks changes in mem allocation.
  - **Proportional Set Size(PSS)**: The no. of non-shared pages and an even distribution of shared pages. Useful when OS wants to know how much memory is used by all processes.
  - **Unique Set Size(USS)**: The no. of non-shared pages



# Sharing memory

In order to fit everything 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.
- ❖ 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 approach allows most of the RAM pages allocated for framework code and resources to be shared across all app processes.



# Memory Leaks



- ❖ Failure of releasing unused objects (leaks) from the memory.
- ❖ One of the main causes for application crash.
- ❖ Causes of memory leaks:
  - Using static views
  - Using code abstraction frequently
  - Unregistered listeners
- ❖ Tools to detect memory leaks:
  - Leak Canary: Memory detection library with ability to decrease down the memory leaks.
    - Functioning: Detecting retained objects, Dumping the head, Analyzing the heap, Categorizing leaks.
  - Android Profiler: a tool that keeps track of memory usage of every application. Detects the performance of the application in the real-time.
    - Parameters: Battery, Memory, CPU Usage, Network Rate



**Thank You!**