Android Application Usage Profiling

- Profiling an Android Application
 - Gathering Memory Usage Statistics
 - Log Messages
 - Performance Monitors
 - Data Analysis
 - System Information

Profiling an Android Application

To gather application running statistics on the system, you can profile the system using Android Studio Performance Profiling tools https://develop er.android.com/studio/profile/index.html.

The IDE provides a set of different tools which can effectively be used to profile any application running on a device or an emulator. To profile the memory usage, CPU/GPU usage or Network bandwidth usage, we can use the Android Monitor https://developer.android.com/studio/profile/andr oid-monitor.html.

The Android Monitor helps us in these tasks:

- Log messages, which can be either system or hardware defined.
- Monitor memory, CPU and GPU usage by the application.
- Network bandwidth usage statistics.

To enable Android Monitor you need to take care of some prerequisites and dependencies as specified.

- 1. The device should remain connected to the system via USB cable and the system should be able to detect the device.
- Enable ADB integration by selecting Tools > Android > Enable ADB Integration. Enable ADB Integration should have a check mark next to it in the menu to indicate it's enabled.
- 3. Make sure Android Device Monitor is not running currently.
- 4. In your app, set the "debuggable" property to "true" in the manifest or "build.gradle" file (it's initially set by default).

Now we just need to display the Android Monitor and when we run the application on the device, we can see its statistics be noted on the corresponding screen for "log messages (logcat)" or "performance monitors (monitors)". To view the Android Monitor, you can select it from View > Tool Windows > Android Monitor. Or you can also display Android Monitor by clicking on the Android Monitor button, which is on the bottom of the main window by default or using the shortcut Alt+6. A screenshot of an app being profiled is shown below.



Gathering Memory Usage Statistics

Android Monitor provides us with various tools to check the usage statistics of the android application. Here we will list down some tools and try to understand how these help in understanding the memory usage of the application.

Log Messages

The "logcat" monitor gives us the log messages while the application is deployed and running in real time. It helps us in understanding what is

happening in the system which is very useful for debugging. For example if an application crashes the best way to find out the cause of the crash is to check the last few commands of the logcat. The screenshot below shows the logcat monitor while an app is running.

And	Android Monitor & .																		
	🔲 Sa	msu	ng SM-G935	5F Android 7.0, API	24 🔻	face	facebook.f8demo (31408) [DEAD] 🔻												
Ō	∎¥ lo	gcat	Monitors +"							Ve	rbose	• •				🗹 Regex	Show o	only selected app	olication 🔻
 ■ ■ ■ ? 		30 31 34 31 34 31 34 31 34 31 35 31 12 31 13 31 13 31 13 31 13 31 13 31 33 31 33 31 33 31	408-31408/1408-3143/1408-3143/1408-3143/1408-31430/1408-31408/1408-31408/1408-31408/1408-31408/1408-31408/1408-31408/1408-31491/1408-31491/1408-31491/1408-31491/1408-31491/1408-3149/1408-314300400000000000000000000000000000000	acebook. †8demo D/V facebook. f8demo E/F facebook. f8demo E/F facebook. f8demo E/F facebook. f8demo D/V facebook. f8demo V/I facebook. f8demo I/I facebook. f8demo I/I facebook. f8demo I/I facebook. f8demo D/V facebook. f8demo D/V facebook. f8demo D/V facebook. f8demo D/V facebook. f8demo D/V facebook. f8demo A/N facebook. f8demo A/N facebook. f8demo A/N	iewRoot: 8DEMO: (8DEMO: (8DEMO:) iewRoot: iewRoot: inputMetl nputMetl nputTran penGLRe; penGLRe; ibGLESv: aali win: ameraMan ative: ; ibc: Fa; (del	Impl@d Couldn done. Instan Impl@d timpl@d thodMan ansport anderer enderer v1: STS nsys: E anagerG [F giv termin atal si 06-19 abugger	d7433d6[Clas n't parse ne ntiating pre d7433d6[Clas d7433d6[Clas d7433d6[Clas d7433d6]Clas clas d7433d6[Clas d7433d6]Clas clas tri Initializ r: Swap beha S GLApi : DT EGLint new L Glabal: Com ven tensor_f nating. ignal 6 (SIC 16:38:50.38 rd: handling	ssifyCamera edictor ssifyCamera ssifyCamera ssifyCamera ssifyCamera ssifyCamera ssifyCamera annel cons read EGL, v avior 1 TS is not vavior 1 TS is not vavior 1 TS is not solve support solve supp	ra]: setV: ata. ra]: MSG_M ra]: MSG_M t: tba=anc structed: version 1 allowed 1 frface(egl_ o camera s :27] Check ode -6 in 3097 W/ : pid=3146	RESIZED WINDOW F droid.vi - mServi fd=88 .4 for Pack winsys Service k failed tid 314 08 uid=1	REPORT: DCUS_CHA ew.input ce.start display* : output 37 (Asyr] 8247 gid	ci=Rect((NNGED 1 Emethod.Ect EInputOrWi accebook.ff *, void*, t->size() ncTask #1]	ClassityCame 0, 0 - 0, 0) ditorInfo@92 indowGainedFi Bdemo EGLSurface, values) id=31437	raj touchMo vi=Rect(0, 01544 nm : ocus EGLConfig, size() outp	de=true , 0 - 0, 0 facebook , egl_win: put size:	0) or=1 .f8demo ic= sys_surface 37748736 g	null **, egl_cc iven size:	olor_buffer_form : 11087562	at*, EGLBoole

Performance Monitors

There are four types of performance monitors which helps us monitor the performance of the app in real time. We can get the statistics for Memory Usage, CPU Usage, Network Usage and GPU Usage with the help of this monitor. They give us tools to record specific additional data in files which can be analysed by using specific analysis tools. The screenshot below shows the performance monitor with each of the four resources being monitored in real time.



Data Analysis

System Information

In the Android Monitor main window, we can find the System Information icon on the side as marked in the screenshot below.



On clicking the icon you will get a list off system information that you can access. These are the various outputs of the "dumpsys" command.

- · Activity Manager State dumpsys activity
- Package Information dumpsys package
- Memory Usage dumpsys meminfo

• A screenshot of the memory usage statistics is shown below. It gives us most of the information we need to know about the amount of memory being used by the application.

CMakeLists.txt ×		xt ×	📀 andro	id × 🖹	org.tens	orflow.der	no_2017.	06.19_11.	.19.txt ×	🧕 AndroidManifest.xr	ml ×	🖟 local.properties ×	🖟 gradle-wrapper.properties × 🧿		
1	Applicat	ions M	emorv Usa	ae (in Ki	lobvtes):										
2	Uptime: 2	236507	865 Realt	ime: 9785	96080										
3															
4	** MEMIN	FO in	pid 15067	[org.ter	isorflow.c	lemo] **									
5			Pss	Private	Private	SwapPss	Heap	Неар	Heap						
6			Total	Dirty	Clean	Dirty	Size	Alloc	Free						
7															
8	Native	Неар	105435	105384	28	94	117760	100077	17682						
9	Dalvik	Неар	65201	65168	0	82	1//6/	10661	/106						
10	Dalvik	utner	81/	816	0	U									
11		Stack	424	424	U	0									
12	Otho	snillelli r. dov	4	4	0	0									
1.0	othe	mman	10250	300	769/	44									
15	ank	mman	345	000	/004	44									
16		mmap	54	0	24	õ									
17	.dex	mmap	564	500	64	õ									
18	.oat	mmap	2928	0	804	ō									
19	.art	mmap	1358	888	76	9									
20	0ther	mmap	18	4	0	0									
21	EGL m	track	38304	38304	0	0									
22	Uni	known	379	376	0	18									
23		TOTAL	226333	212256	8684	247	135527	110738	24788						
24															
25	App Sum	mary													
20				FSS(ND)											
27		lav	a Hean	66132											
29		Nativ	e Heap:	105384											
30			Code:	9464											
31			Stack:	424											
32		Gr	aphics:	38304											
33	P	rivate	Other:	1232											
34			System:	5393											
35							_								
36			TOTAL:	226333	TOTA	AL SWAP PSS	: 2	47							
3/	Objects														
39	objects		Views	٥	Vi	ewRootTmpl		1							
40			ntexts:	3		Activities	:	ī							
41			Assets:	3	Ass	setManagers	:	2							
42	L	ocal B	inders:	11	Pro	xy Binders	:	20							
43	Pa	arcel	memory:	5	Pa	arcel count	:	21							
44	Deat	h Reci	pients:	2	OpenS	SSL Sockets	:	Θ							
45															
46	SQL	uruce	V. UCED												
47	DACECO	MEMOR	Y_USED:	0				•							
40	PAGECA	CHE_UV	ENFLOW:	U	P	ALLUC_SIZE	•	0							
49 50															
50															

- Memory Usage over Time dumpsys procstats
- Graphics State dumpsys gfxinfo

Some of the other Data Analysis tools which can be viewed using Android Monitor are as listed.

1. HPROF Viewer and Analyzer: This tools dumps the Java Heap to an HPROF file using the Memory Monitor. It's icon can be found, as one of the three icons, on the memory monitor just beside the play/pause button. The output of the dump can be viewed in the HPROF

- Viewer which gives a nice class hierarchical view of the heap dump.
 Allocation Tracker: This tool can also be found on the Memory Monitor which is used to track down all the memory allocation data in the app. It displays each method responsible for the allocation along with the size and the number of instances.
 Method Trace
- 4. GPU Debugger