APB – A Linux based Application Performance Benchmarking Tool

White Paper

Bensingh Beno
Software Architect
Luxoft GmbH
November 2020

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH OPEN SOURCE PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN GNU GPLV2 TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, I ASSUME NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF OPEN SOURCE PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

What is APB tool & why is it needed? :

Application performance benchmarking is a quint-essential requirement to measure the performance quality of a software application.

There are numerous tools / packages / frameworks available to measure performance (ANtutu, GpuPrime, nvidia-smi etc) . Although, very limited amount of them work for embedded automotive systems and also do not behave the same for different architectures (ARM,X86). APB provides an exquisite amount of features that makes itself stand apart from other conventional benchmarking tools.

Features and Capabilities of APB :

Measure & Analyse core performance attributes (CPU, Memory,I/O)
Measure & Analyse GPU resource consumption
Measure & Analyse Systemcalls and bottlenecks from application to kernel

Plot a readable chart of performance metric for visual comparision and analysis.
Analyse Boot behavior and startup times.

Sample Use Case for Analysis used in this whitepaper:

In this whitepaper we use a single application developed on 3 different frameworks (Java, Qt & Flutter) running on 2 different Operating Systems - Linux & Android. In the end we conclude which framework/OS performs better using APB tool's benchmarks & plots. Since same hardware is used for all cases, the analysis is also called as **FPA** – Fair Peformance Analysis

Environment:

The below elements form the common foundation of the analysis of both candidates.

Element	Linux	Android
Hardware - SOC	Raspberry Pi 3B	Raspberry Pi 3B
OS Name	Raspbian Buster	Android Lineage OS 16
Linux Kernel	4.19.58-v7l+	4.19.102-v7
Total CPU	Quad Core 1.2GHz Broadcom BCM2837 64bit CPU	< - - Same
Total RAM	1GB RAM	< - - Same
Display	1 HDMI - HD 1280 x 720	< - - Same
UC1: Use Case Boot	**Boot Time from Power on to Full Desktop Access**	< - - Same
UC2: After Boot Statistics	**CPU, Memory load after full boot**	< - - Same
UC3 : Use Case Application	**Launch and app that can load a compressed image from disk and render it to the display (fullscreen). The app is written in Java (Android), Flutter (Android) and Qt (Linux)**	< - - Same
UC4: Time taken for UC3	**Launch UC1 and measure time elapsed from launch to full execution.**	< - - Same
UC5: App Execution Statistics	**CPU, Memory load , SystemCall Context Switches during UC3**	< - - Same

FPA Results

UC1: Use Case Boot :

	Linux	Android
Boot Time	~ 20 Seconds	~ 70 Seconds

UC2: After Boot Statistics

Conditions:

Wifi Driver - OFF

Default Apps - NONE

Network - ETH ONLY

	Linux	Android
CPU Usage Avg after Boot	2 %	3 %
RAM Usage Avg after Boot	128 MiB	128 MiB
Default Process Count	130	168

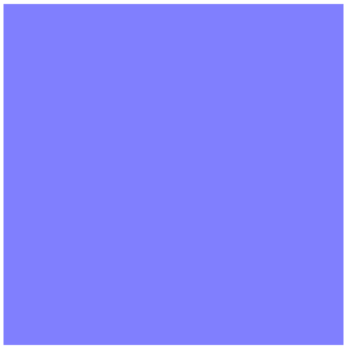
UC3 : Use Case Application

Linux Qt App : https://github.com/bensinghbeno/design-engine/tree/master/projects/qt/imagedisplay

Android Java App :https://github.com/bensinghbeno/design-engine/tree/master/projects/android/native_ImageDisplay

Android Flutter App : https://github.com/bensinghbeno/design-engine/tree/master/projects/android/flutter_ImageDisplay

Image :



UC4: Time taken for UC3

	Linux - Qt App	Android - Java App	Android - Flutter App
Time for Full launch	2 Seconds	4 Seconds	5 Seconds

UC5: App Execution Statistics

CPU - Plot :

Java & Flutter App on Android , Qt App on Linux

The app is launched 3 times successively and hence the 3 cpu peaks / Dips.

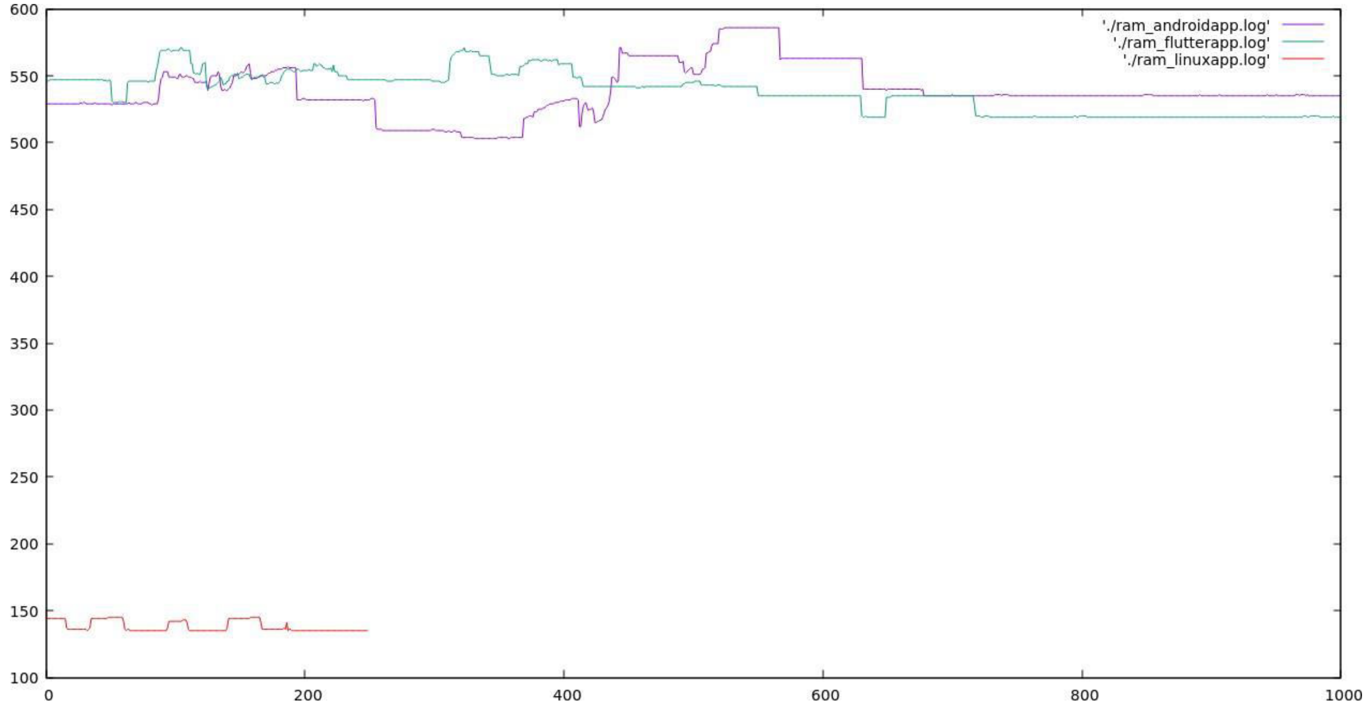
Cpu usage is captured using **iostat** tool.

Ram Plot : (MebiBytes)

Java & Flutter App on Android , Qt App on Linux

The app is launched 3 times successively and hence the 3 ram peaks / Dips.

Ram values are captured using **free** tool.



System Calls :

System calls are captured using **strace** tool.

	Linux - Qt App	**Android - Java App**	**Android - Flutter App**
System Call Count	5945	24461	52890

Conclusion

	Linux - Qt App	**Android - Java App**	**Android - Flutter App**
CPU Usage	LOW ~ 15 %	HIGHEST ~ 65 %	HIGH ~ 50 %
RAM Usage	LOW ~ 10 MiB	HIGHEST ~ 70 MiB	MEDIUM ~ 10 MiB
System Call Count	LOW	MEDIUM	HIGH

The Qt Application running on Linux OS is the best performer!!