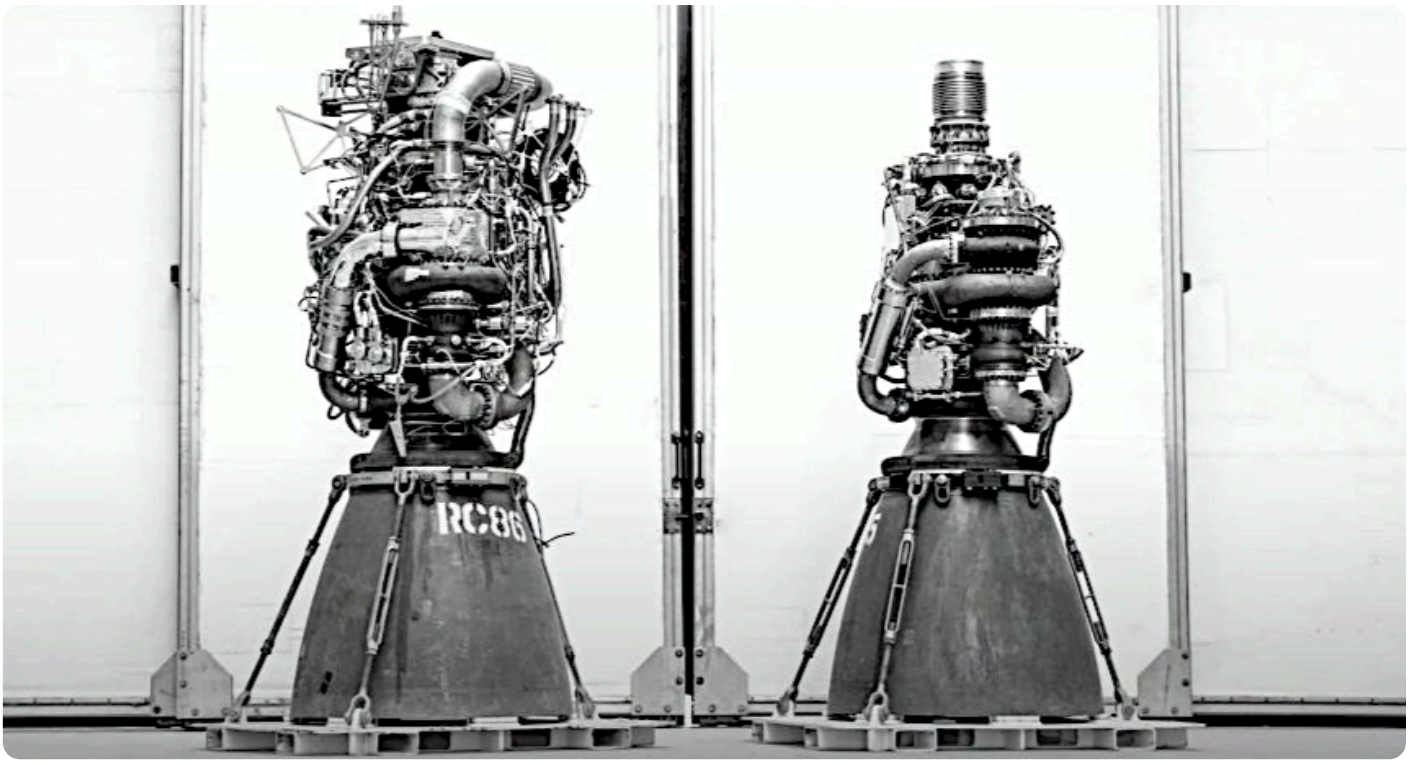


Linux, Android & RedFish OS

Roberto A. Foglietta
GNU/Linux Expert and Innovation Supporter
Published Oct 13, 2024

*This article is based on a post written in Italian and **published** on LinkedIn the same day. However, the content of the article differs from a mere translation in English.*

Premise

11th October 2024 - Google is preparing to let you run Linux apps on Android, just like Chrome OS. Google is developing the framework to let you run Debian in a virtual machine on your Android device. - **lnkd.in/dPG79nyf**

Engineers at Google started work on a new Terminal app for Android a couple of weeks ago. This Terminal app is part of the Android Virtualization Framework (AVF) and contains a WebView that connects to a Linux virtual machine via a local IP address, allowing you to run Linux commands from the Android host. Initially, you had to manually enable this Terminal app using a shell command and then configure the Linux VM yourself. However, in

recent days, Google began work on integrating the Terminal app into Android as well as turning it into an all-in-one app for running a Linux distro in a VM.

In the first place, it must be said that Google is not redefining the architecture of Android but only providing a user-space tool that allows Debian to run on their mobile platform and then for any other derived Linux distribution, also.

Google AVF architecture

Curious because considering that Android's kernel is actually Linux, it would be useful to do the opposite: run Linux and virtualise Android. In fact, it already does, Android runs by a hypervisor called pKVM. - **lnkd.in/dVu8vyQp**

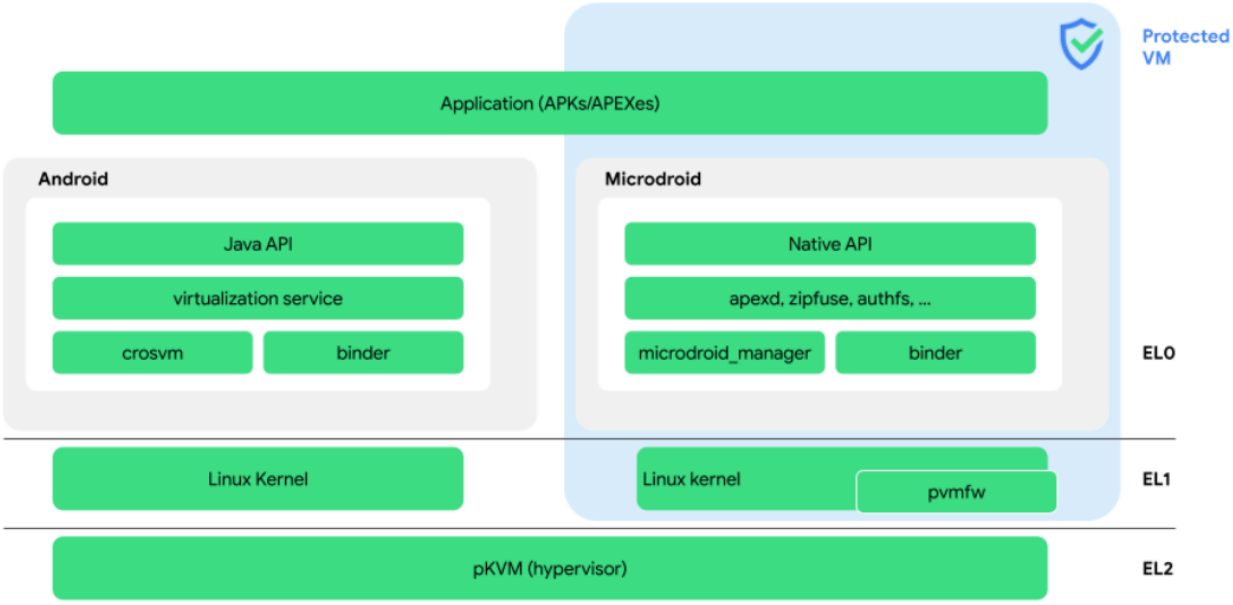


Figure 1. *AVF architecture*

Android Virtualization Framework (AVF) overview

However, not with a virtual machine, which would significantly degrade the system's performance (usually by 1/2), but through segregation, which is absolutely possible, since unlike a generic VM, execution on the processor is supervised but transparent.

Moreover, ARM CPUs already have specific instructions to achieve this type of segregation at the code execution level (e.g. restricting access to RAM to certain segments, only those allocated).

RedFish OS approach

According to the Android documentation, everything is just in place: Linux kernel, hypervisor, separation of layers, but at the same time there is nothing like RedFish OS.

It seems that there is everything and instead there is nothing (cit.) simply because such architecture is ‘normal’ -made at the state of art - but in its development never took into account devices hardware fragmentation and the optimization of costs of scale for a mobile devices market that barely did not even exist before Android or not at that scale.

What are the advantages of the RedFish OS approach?

The advantages of an Android container are that all the hardware management part would go out of the picture, so Android would no longer be an operating system but an application run-time framework and a development environment for apps. Just as it is today, but in a container. As it is today, precisely.

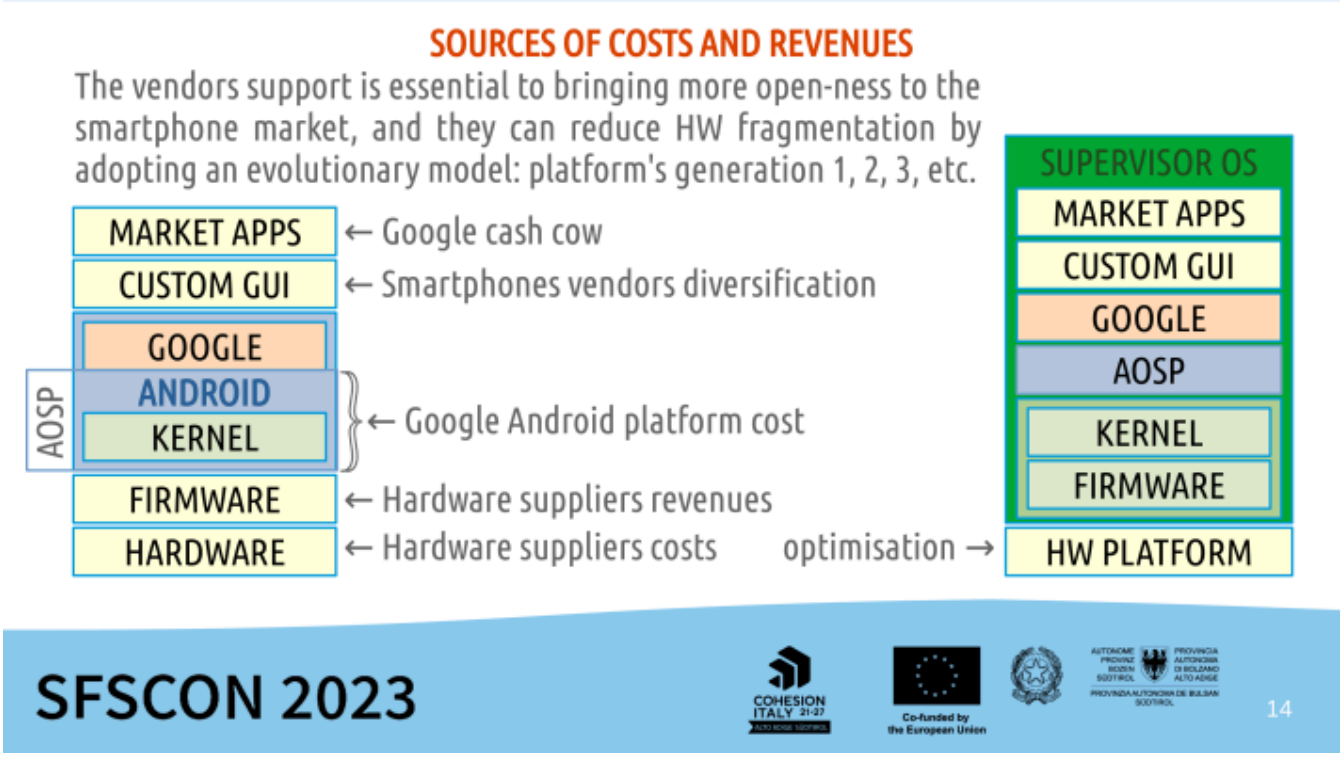


Image source:

This, in essence, is the concept behind RedFish OS. It is not everything, but it is the fundamental part. Hence, its nature of a dual-mode bootable image for

rescue/maintenance and superved execution of the Android system.

As the rest of its value-added functionalities would rely on these two markets separation without which it could be just an app with root privileges. Like a VM for running Debian application within the Android platform like ChromeOS is able to do.

Consultancy

This case shown the importance of the consultancy.

Because consultancy gives its best when a company need a vertical specialist in a specific topi for a limited amount of time and that verticalization is hard to find around or it would take too much time developing internally.

On the other side of spectrum, consultancy gives its best when a project is going to face a challenge never faced before. In this case, a multiply-mildly-verticalised specialist is essential.

Someone that is not strongly tied with a technology or a view, in particular but s/he developed enough experience and a method good enough to drive STEM people further the current state of the art.

The verb 'to drive' has been purposely used here because the mystic and vague idea of the leadership did not work in facing the unknown costs effort and when the quest is quite challenging.

People will not follow in that conditions and not even wish to be pushed into it. Hence, driving is the last resort or accepting a no-go as alternative option.

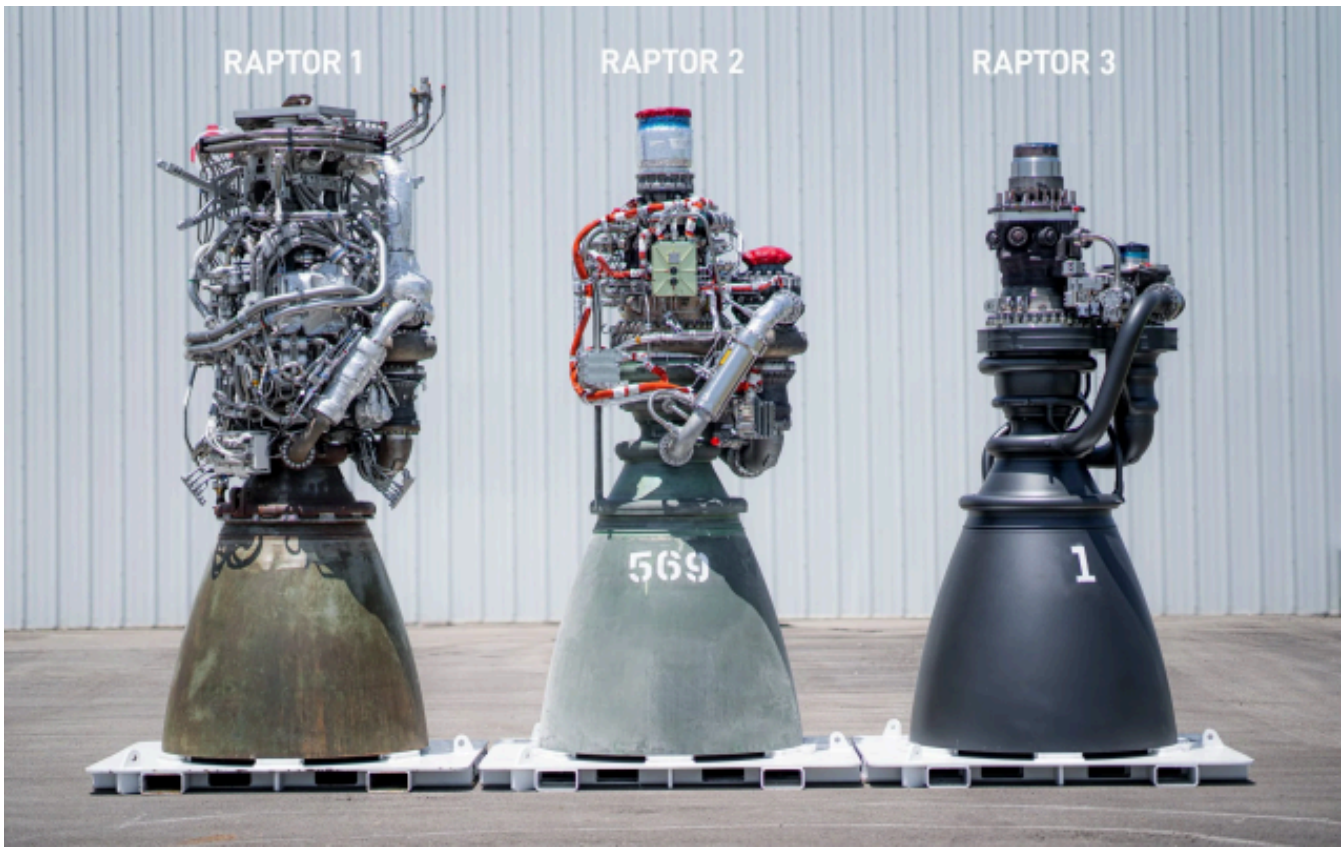
However, a driver is doomed to not remain in the position after the convoy reach its destination. Because, people do not like be driven but they prefer to follow someone around a Candy Land where each their step give them a little satisfaction.

Plus, there is always another convoy need a driver to cross a desert where there is not a candy for each step but just effort.

Simplicity

The simplicity is the ultimate form of the sophistication. The STEM people are able to create impressing clockworks in which the complications are managed precisely and beautifully. Swiss luxury watch manufacturing is a nice example of this.

Also the engines of the Space Shuttle belong within this example. However, with the Starship's launcher project by Elon Musk's industries we saw something going further: the Raptor engine in its 1st generation was a beautiful clockwork of complications.



Raptor engine evolution among three generation

Was it necessary achieve such sophistication? The number of propulsion engines in the Space Shuttle was 3 while the Starship's launcher has more than 30. An order of magnitude greater.

Such a new scale required to drop any kind of complication in order to cut every single costs included those related to maintenance and because each piece is at risk of failing during a launch. Hence, as less pieces are involved as lower risk of encountering a failure.

One of the basic way to reduce complications is disengagement or separation but layers are a kind of separation. The disengagement principle also implies to create some degrees of independence (freedom) among some macro-layers.

In fact, a compact/rigid system has several good proprieties but flexibility and resilience are not among them, thus also security without a complete redundancy.

Conclusion

The fortune of IBM was being forced by the US anti-trust to release the specification for free-as-free-beer their IBM AT PC standards.

Comparing the AT architecture with the ARM one, it was mediocre at those times as well. Despite its shortcomings, being a universal and free-of-charge standard made it the most successful HW platform until the advent of the ARM based mobile devices.

Instead, their proprietary and hard-to-duplicate DR-DOS did not had the same fortune compared to the mediocre MS-DOS and much easier to duplicate. That's made the begin of the Microsoft fortune.

The virality of being duplicated almost without restrictions and then spreading around even without any marketing pressure to push it.

Hence, it does not matter how good Android is. It spread around earning a 2x size market in number of devices in circulation (or in use) compared the completely closed and proprietary alternative by Apple. However, not so much in net revenues and in particular per unit.

The key to change this scenario for the better of all the players relies in introducing a degree of freedom among the macro-layers of the Android architecture.

This will bring more flexibility and those, will adopt the better-than-the-previous generations device models like Apple did and does, will achieve a great advantage.

This despite flexibility will help also those who prefer rely on the hardware fragmentation for increasing their revenues but gross revenues are not necessarily net incoming.

People, who can put some more extra money on buying a device, will prefer to buy the Nth generation that a device that gave brought them good value/service in the past rather than be tempted by a cheaper and unknown alternatives.

Still, cheaper and soon-unsupported alternatives will always in the wild because as much the confusion as the quest for perfection are both two fundamental traits of the human nature whatever the market segmentation is or will be.

Related papers

- **RedFish OS business presentation with an executive introduction**
- **GitHub RedFish OS' collection of SFSCON 2023 paper collection**
- **P²C² management style and my SCRUM in a nutshell**

Share alike