BOOTING AUTOMOTIVE ECUS REALLY FAST WITH MODERN SECURITY FEATURES

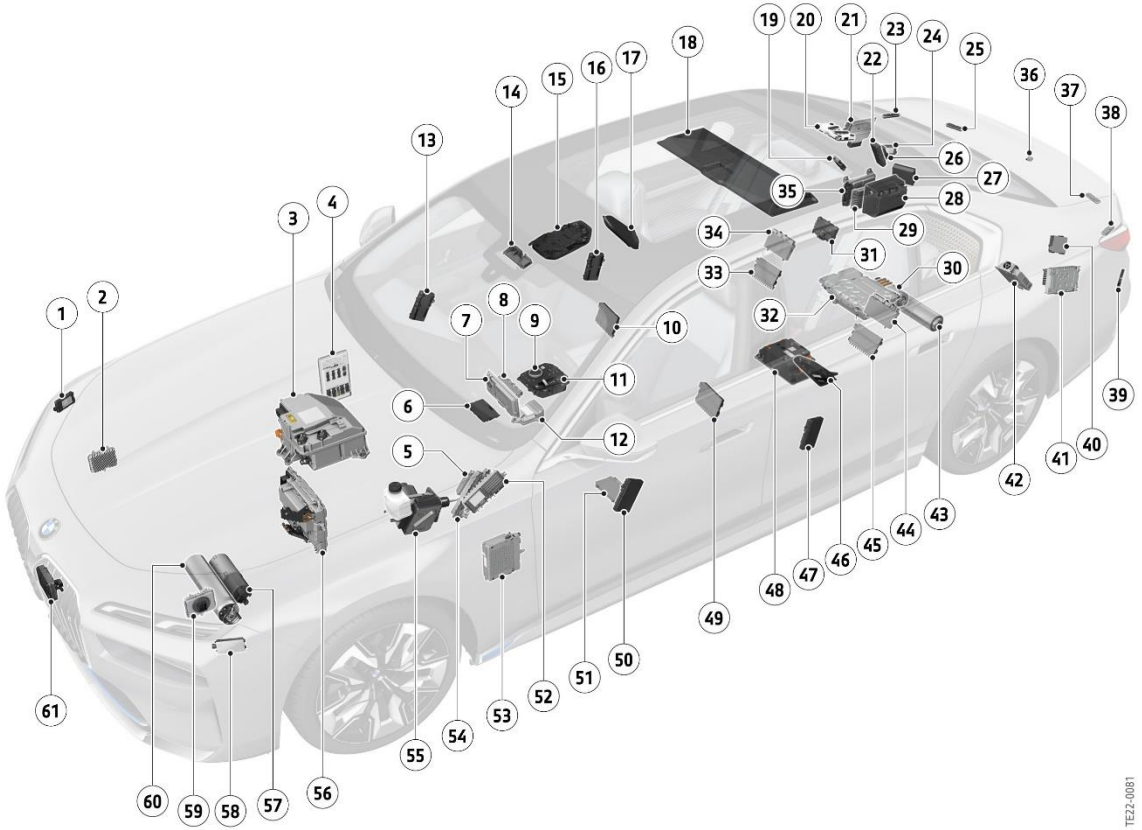
EMBEDDED LINUX CONFERENCE 2022

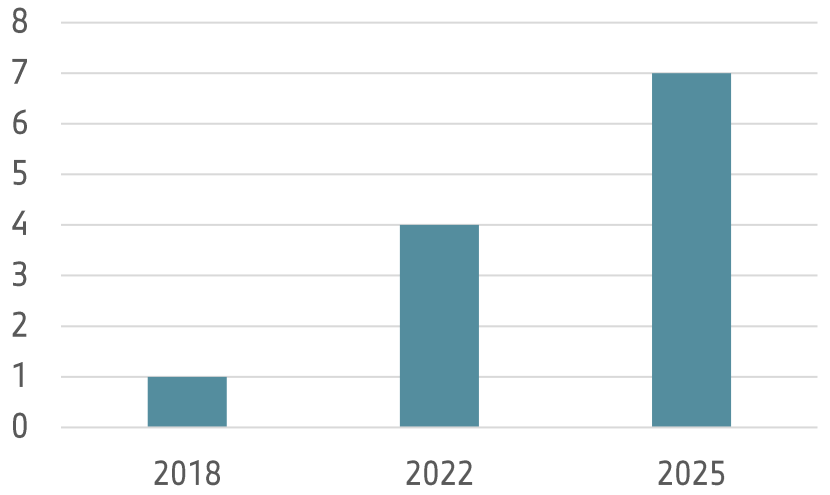
14 September 2022 | Brendan Le Foll | brendan.le-foll@bmw.de

BMW CAR IT

OUR LINUX DISTRIBUTION – NODE0

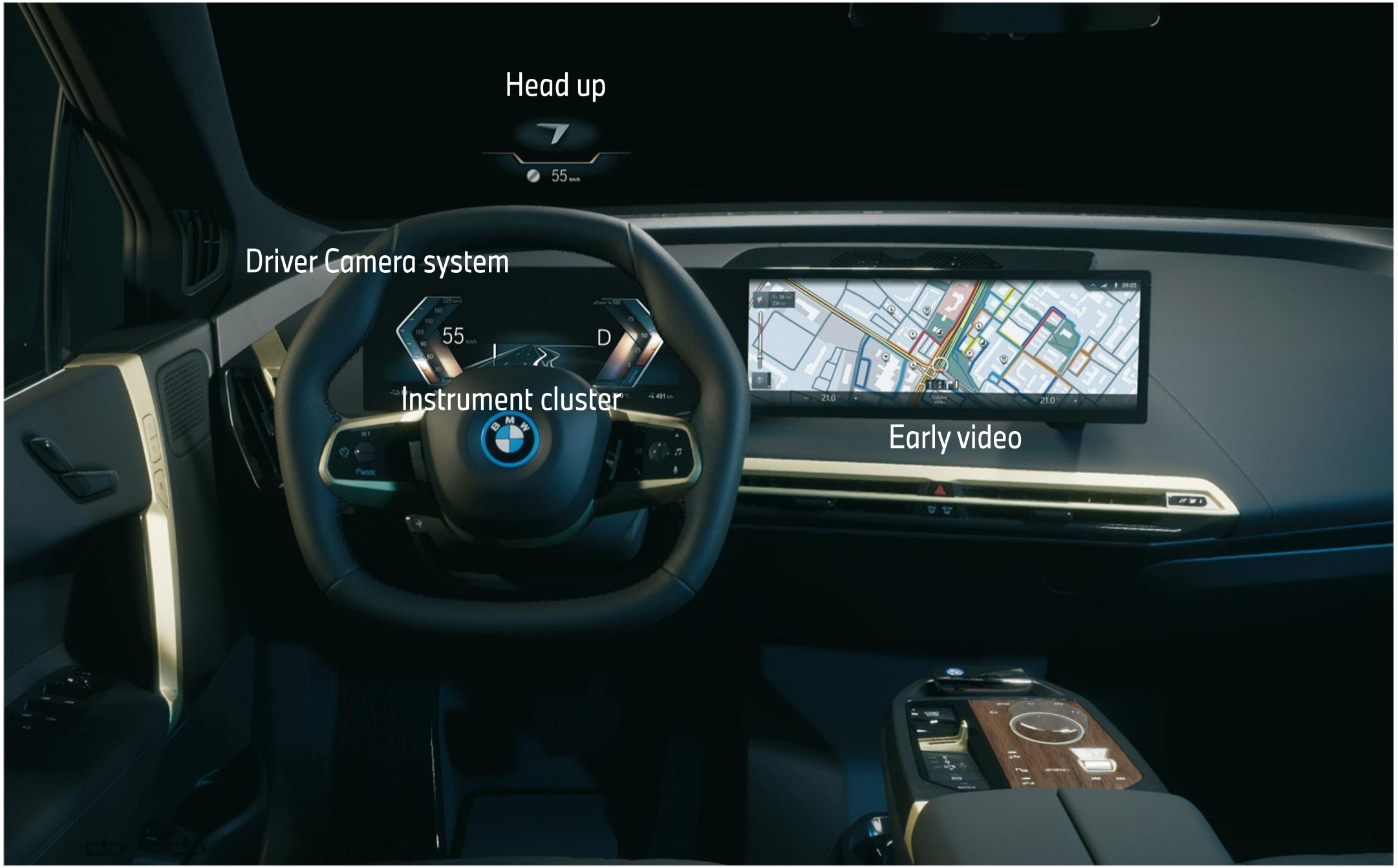
- Over the last few years we have been standardising on our Linux distribution for our in vehicles ECUs – a solution that we call Node0:
  - Yocto Linux based
  - systemd / glibc based linux userspace
  - Integrity protection for all executable artifacts
  - Encryption for userdata
  - Hardware keystore usage
- Avoids delegating early tasks to an RTOS that then handovers the feature to Linux
- It doesn't actually boot that fast – but allows for userspace processes to start in a determined order before alot of the system is ready





- Autosar / Adaptive Autosar free
- First shipped in 4 separate ECUs on the new 7 series

WHY DO WE NEED TO START FAST?


Head up
55
Driver Camera system
55
D
Instrument cluster
Early video

WHAT DO WE MEAN BY 'MODERN SECURITY'?

- Integrity protection for all executables
  - To detect if the system was tampered with
- Secure key storage
  - To protect backend connectivity
  - ECU authentication
  - Prevent physical theft
  - DRM key storage for video playback usecases
- Mandatory access control
- IPC security policy
- Ethernet security/pairing
- Encrypt customer data
- No binaries should run as root, minimal privileges etc...

Source: funroll-loops.info

Let’s get booting

WHAT DO WE DO?

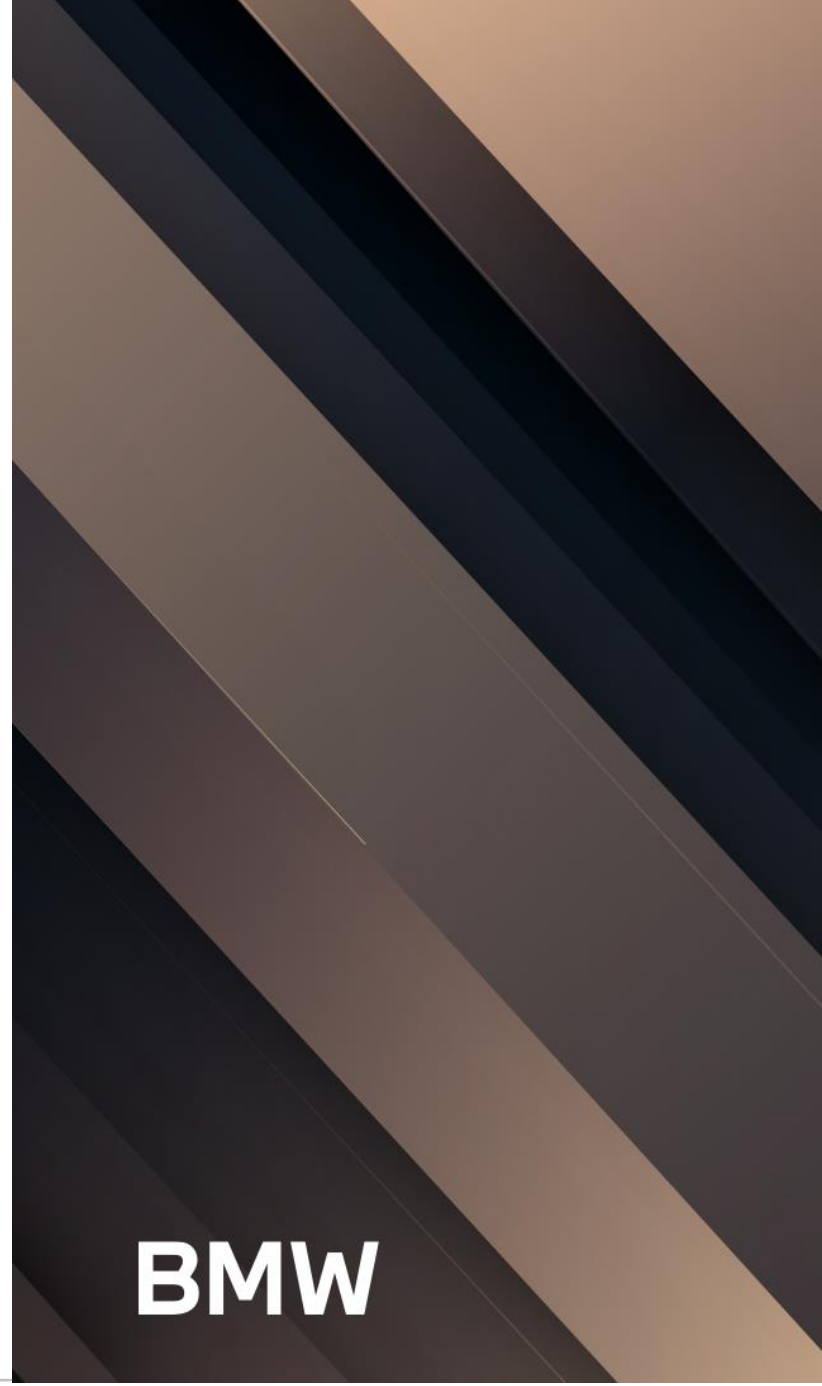
TG22-0928

FIRST – CHEAT!

- It's very difficult to enter a car in < 3s. Try it
- Suspend to RAM is pretty fast
  - Can't boot everything super fast anyways so STR provides a good compromise
  - STR on linux is not amazingly fast, resume optimised is often taking ~1-1.5s
  - i.e. our rear view camera takes ~2.7s in cold boot whilst ~2.2s using suspend to ram
  - Battery protection means STR cannot always be used
  - A lot of ARM SOC vendors don't support STR
- Hibernation is interesting but in my view too damaging on automotive flash where lifetimes of > 15yrs have to be guaranteed and expected from consumers

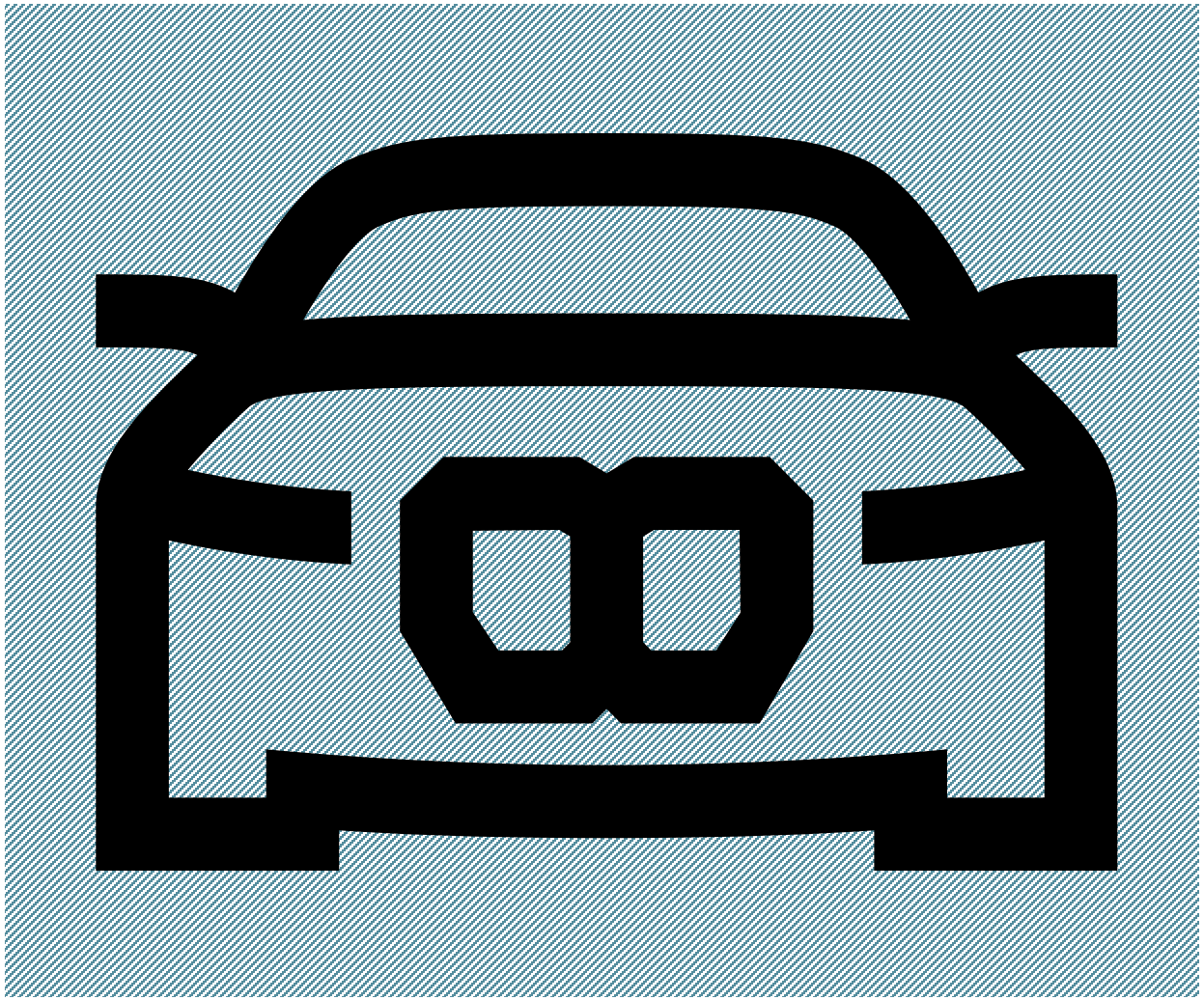
But I need to!

OK - ☺



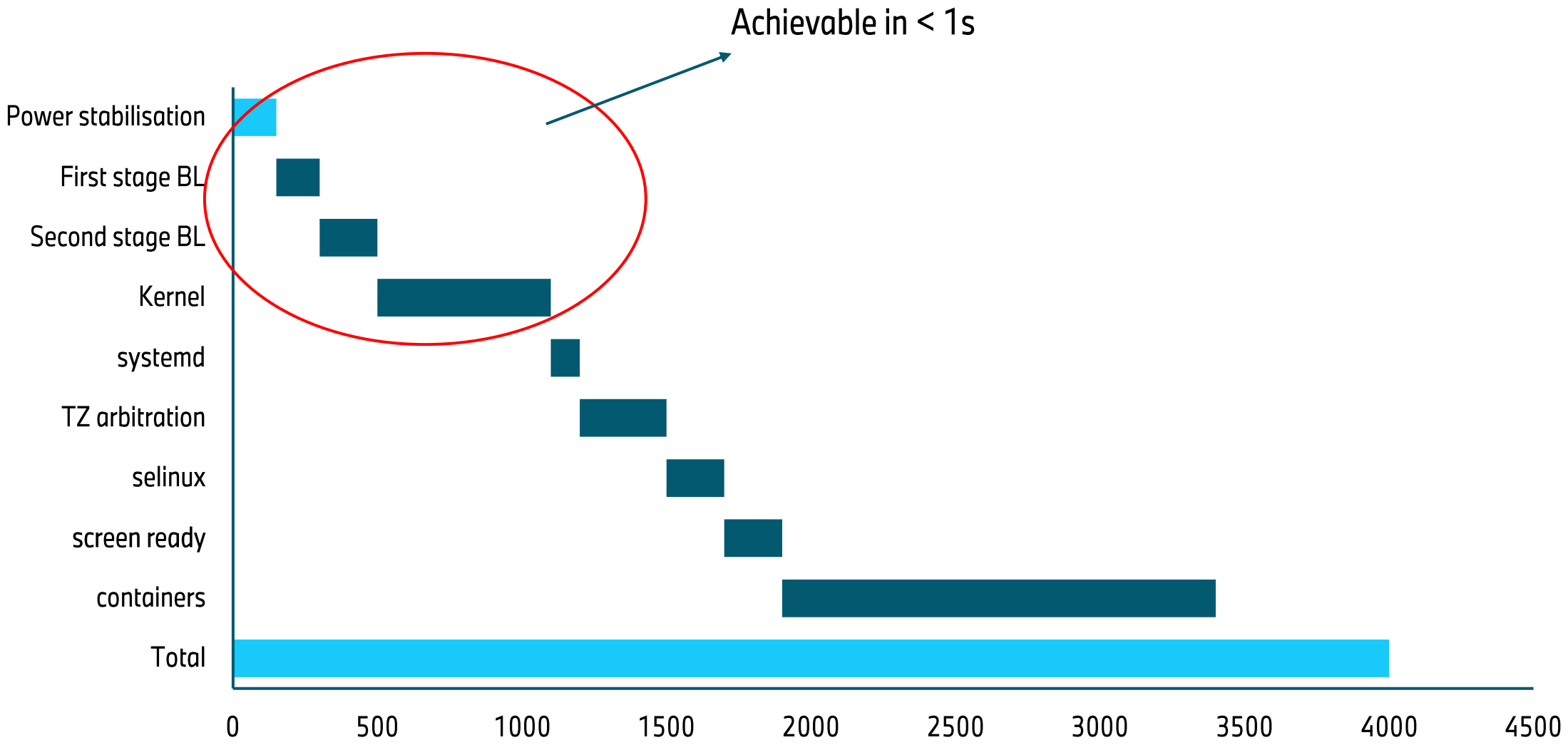
WHAT ARE THE HIGH LEVEL ISSUES?

- A/B update/partitioning
- Integrity checks
  - Secure boot
  - Dm-verity
- Linux kernel load times
  - Kernel modules
- Init system performance
  - Udev
  - Systemd
- Security
  - Polkit
  - Trustzone arbitration

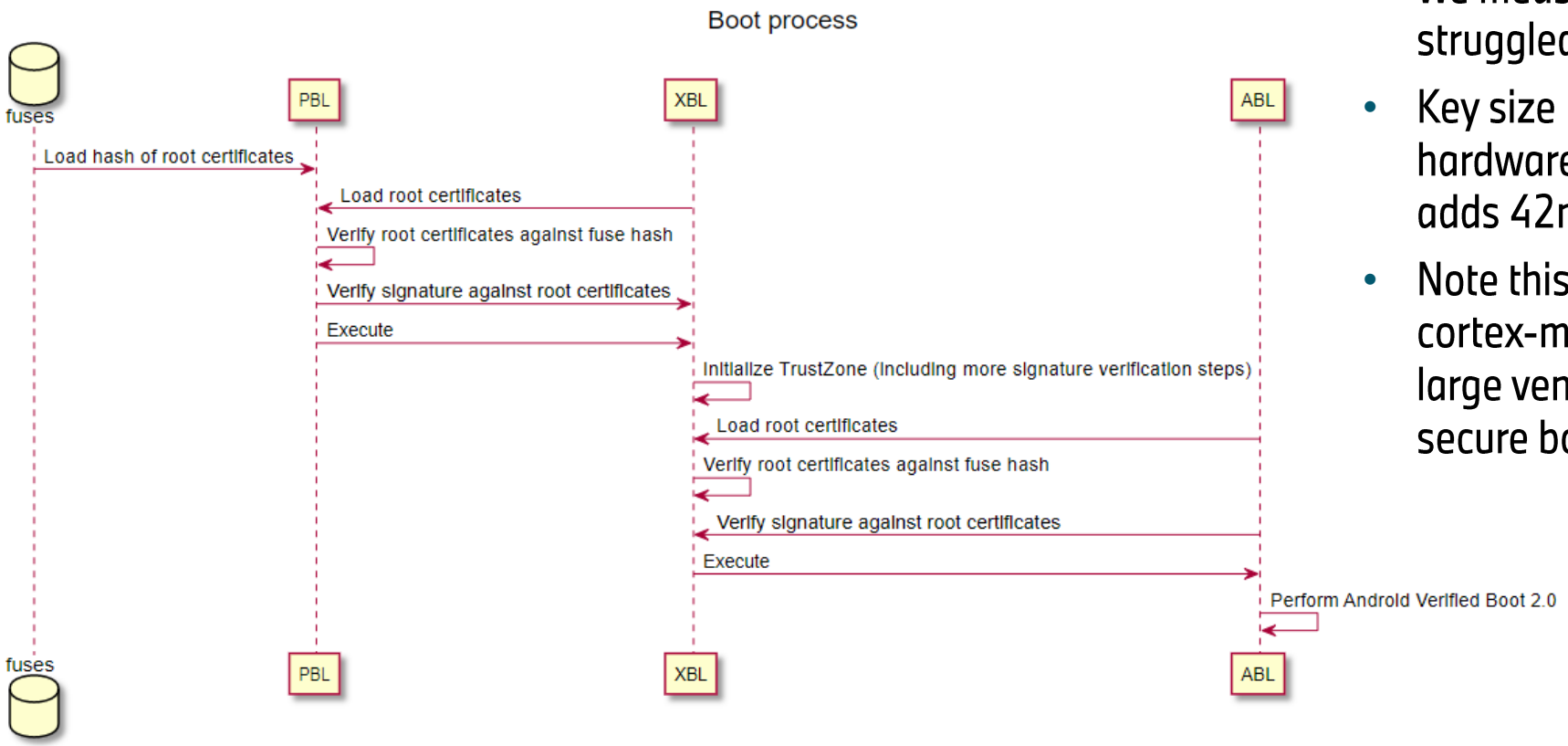


Before Linux

TIMELINE OF BOOT SEQUENCE

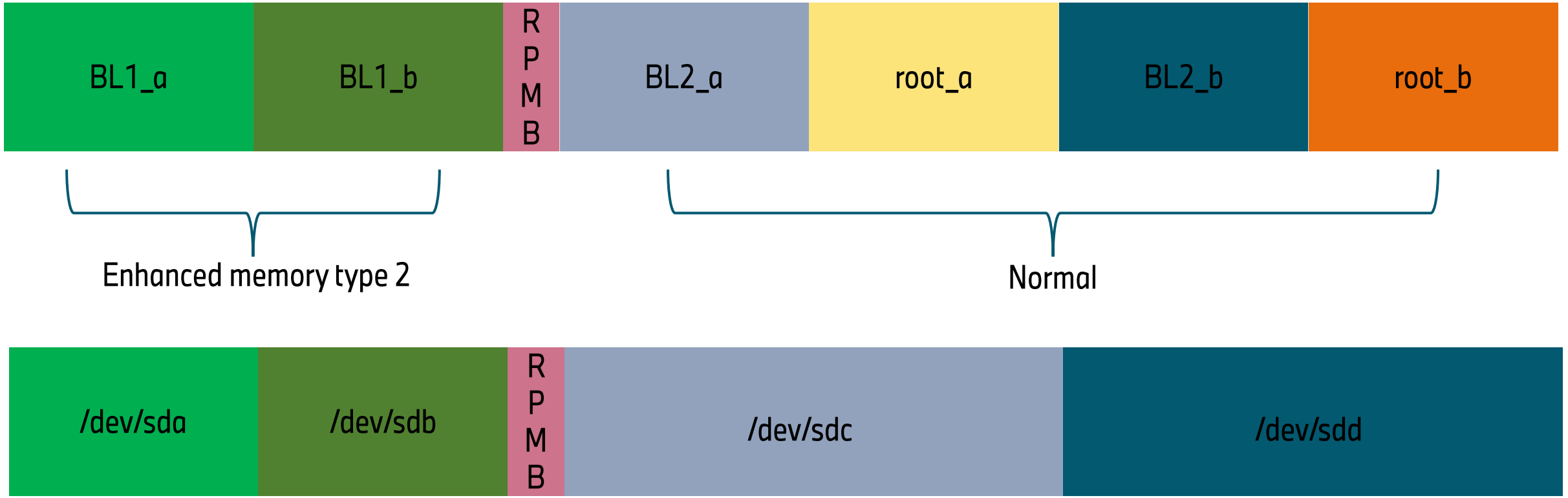


SECURE BOOT ISN'T EVEN SLOW



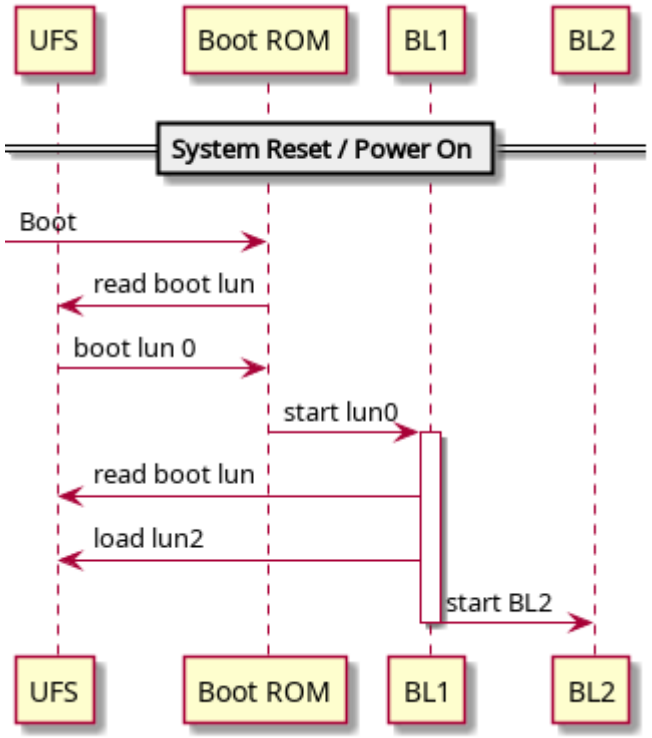
- Such a signing scheme has such minimal impact that two SOC vendors independently told us impact is 'minimal'. We measured on one SOC and also struggled to see a difference
- Key size has minimal impact on modern hardware. Moving from 2k key sizes to 4k adds 42ms (SOC vendor claims 57ms)
- Note this also applies to relatively small cortex-m or –r based microcontrollers from large vendors, they are all able to do secure boot really really fast

MODERN FLASH IS SUPER FAST



- Manual enumeration of block devices avoids randomness
- Udev when you have lots of block devices is slow

A/B PARTITIONING DOESN'T SLOW YOU DOWN!

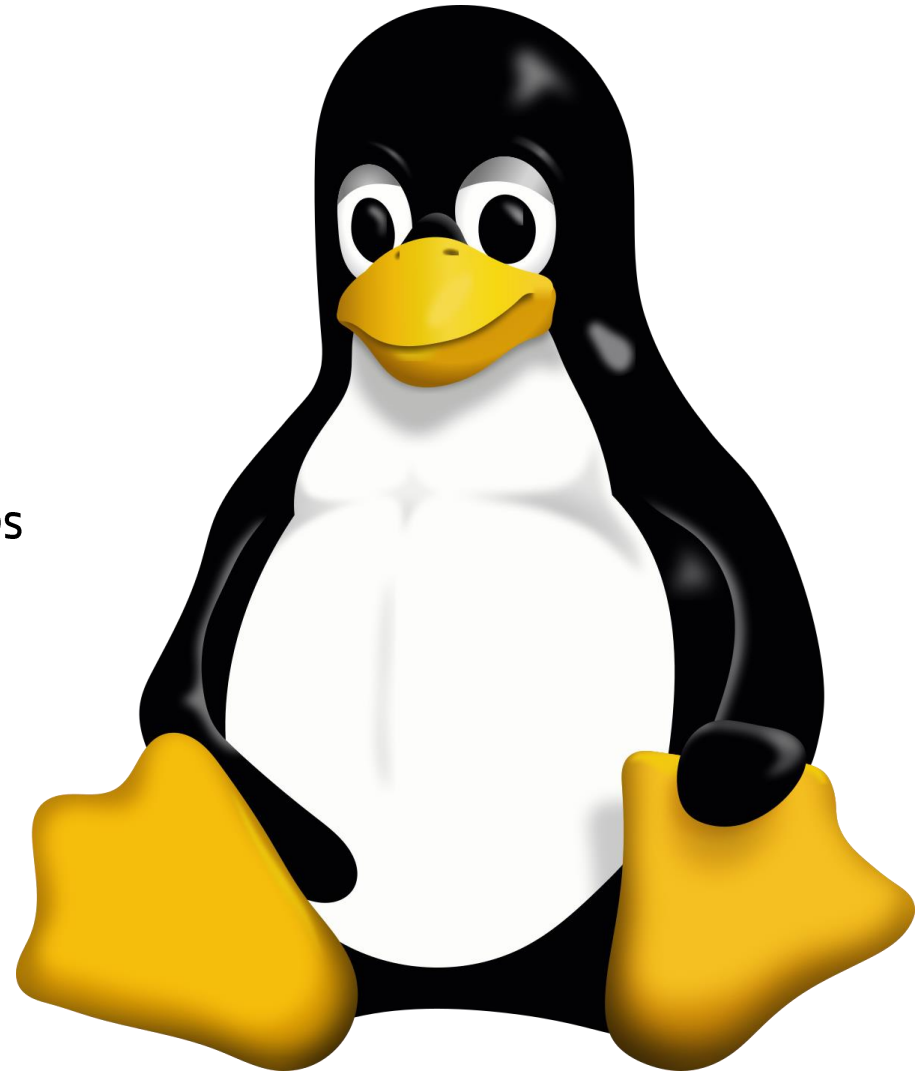


- Dump the NOR SPI
- Works just as well with an eMMC
- Boot ROM is baked in the SOC, can't be bricked in dev!
- Obviously downside is recovery from a failed boot is SOC dependant
- Obviously not secure on it's own

Kernel space

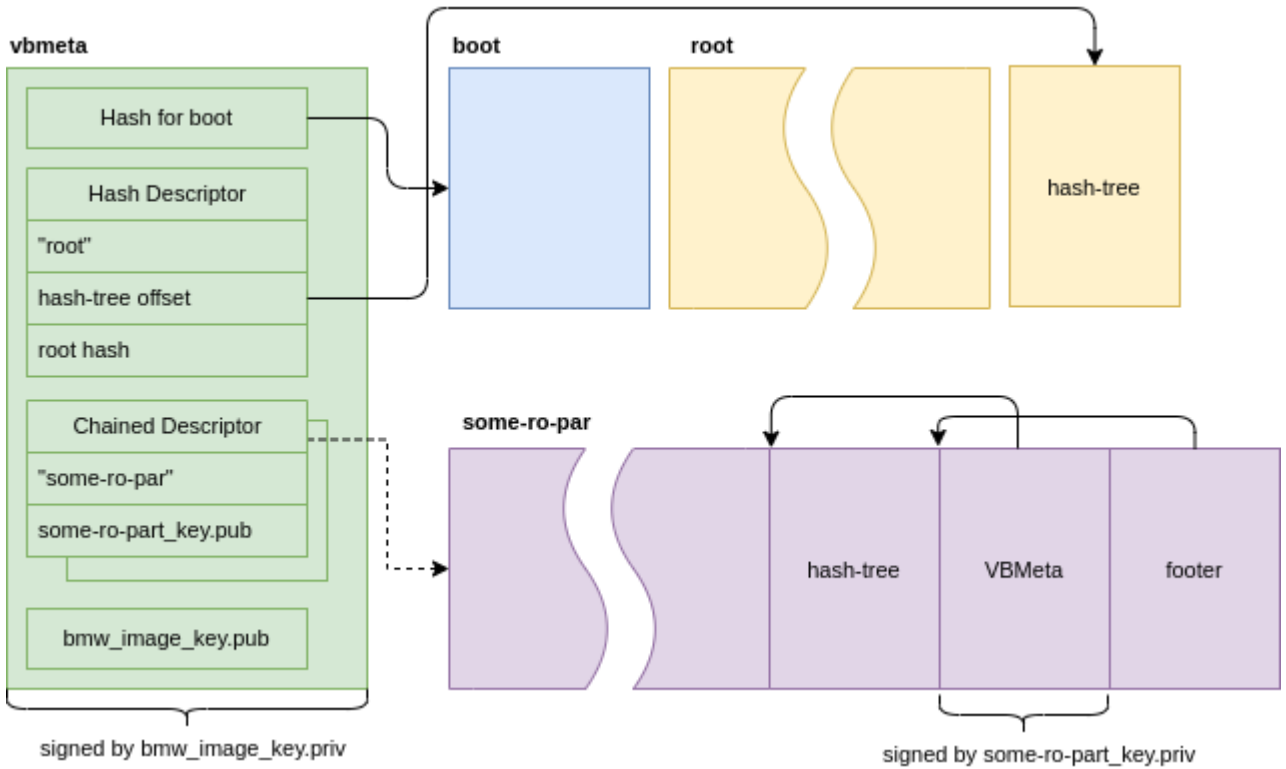
LOADING THE KERNEL

- Essentially we modularise absolutely everything we can
- Hotplug RAM
- Keep CPU frequencies up! -> SCHED_BOOST
- If you have a fat.big.little architecture maximise moving work to the big cores
- Scheduler has a huge impact – WALT vs PELT etc...
  - We chose PELT32 -> reduces core migrations
  - Better latencies for near-RT processes



EXTENDING THE CHAIN OF TRUST TO THE ROOTFS

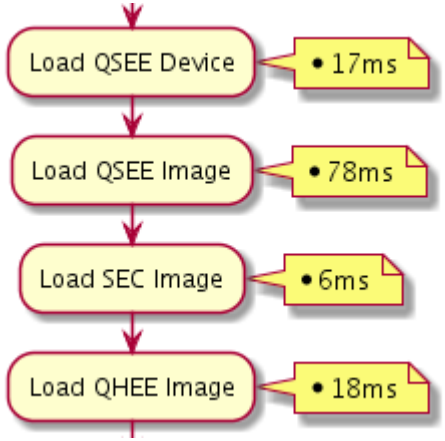
- Dm-verity but skip the initrd -> just like Android
- Using AVB2.0 format from AOSP - https://android.googlesource.com/platform/external/avb/
- Much faster than IMA, doesn't require the use of an initrd
- Does require an AVB enabled bootloader, but increasingly common in modern SOCs in order to support Android
- No need for fscheck!
- FEC from dm-verity allows error correction



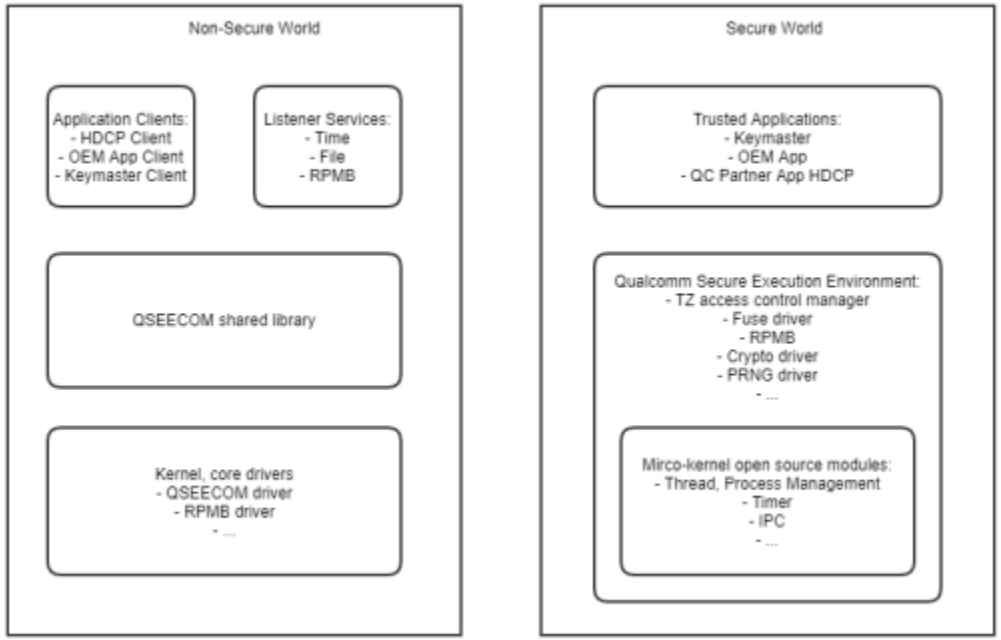
- Combine this with loadpin and you can skip the kernel module signing! Also useful for firmware/PIL

THE ISSUE WITH TRUSTZONE

- Trusted applications run in a different runlevel than the kernel. They can typically be started really early without much need for kernel interaction
- However in order for userspace applications to do anything with trustzone you need the arbitrartion daemon to be loaded



- Hopefully ARM systemready IR (https://www.arm.com/architecture/system-architectures/systemready-certification-program) helps us avoid future problems like this with certain SOC vendors

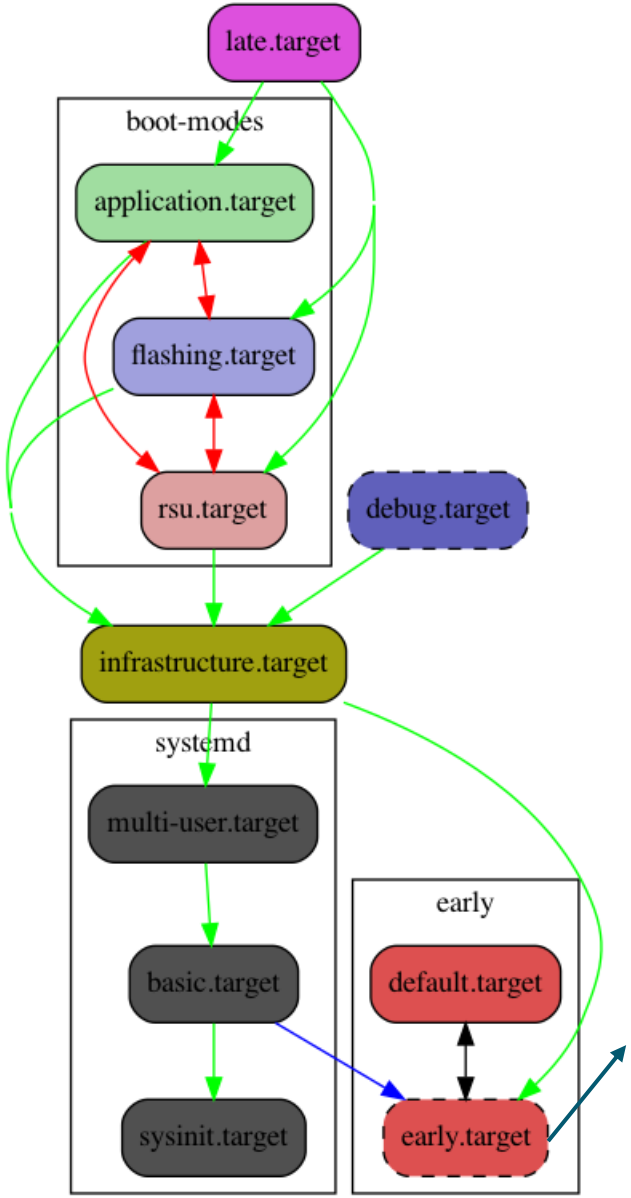


User space

SYSTEMD

- We define a systemd early.target
- The aim is to start certain processes before we have full udev enumeration and sysinit.target is not up
- This allows us to reach boot times that are fast enough to avoid

- Early applications can be started during the early.target when required if they have minimal dependencies




```
dbus.service
systemd-networkd.service
rvp.service
ioc-proxy.service
vsomeipd.service
udpnm.service
nodestatemanager.service
nodestartupcontroller.service
```

•→ X : Wants=X
•→ Y : After=Y
•→ X : Conflicts=X
•→ X : id Y

SYSTEMD – UDEV IS TOO NOISY

- Udev events are replayed in an unpredictable way
- Filtering based on certain subsystems or prioritising is tricky
- Allow early.target to run things before udev has finished
- Avoid having too many events retriggered

- Generally patches have been considered not generic enough and having confusing configuration possibilities e.g.
  https://github.com/systemd/systemd/pull/19637

POLKIT

- If you use sytemd, you use dbus ☺
- Polkit uses javascript for its configuration!

It's very nice, but it's kinda big....

OUR SOLUTION – SMOLKIT!

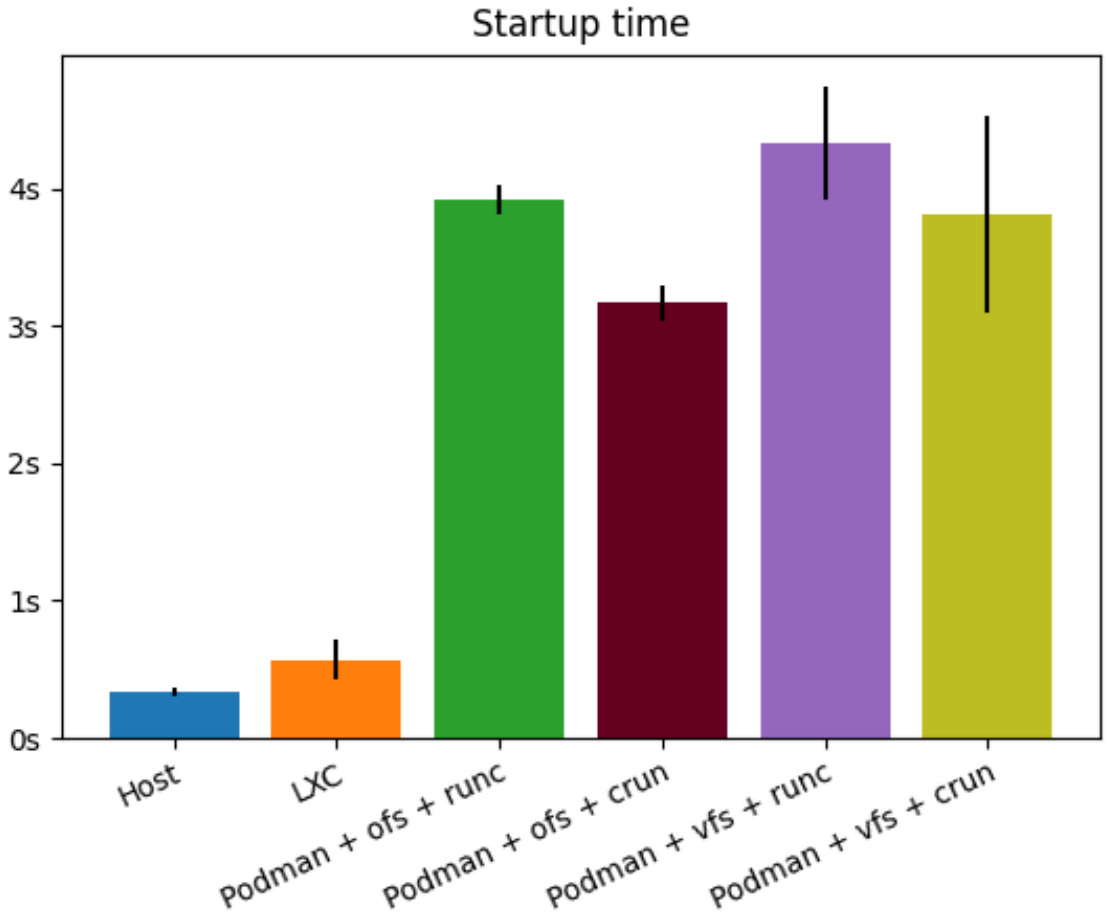
Not quite ready for primetime
but we will get there!

- Inspired by https://github.com/ostroproject/groupcheck

- Recent developments to replace mozjs with ducktape have yield pretty decent size improvements but the parsing remains relatively slow for devices with slower CPUs
- Virtual provider for polkit in yocto
- Will be opensourced later on in the year

CONTAINERS!

- Our containerised system is based on LXC + systemd, note that we chose LXC because it's so much faster to boot than the competition
- Containerised platforms are generally not even considering boot time as a requirement



Conclusion

WHO ELSE HAS THIS PROBLEM?

THANKS FOR LISTENING!