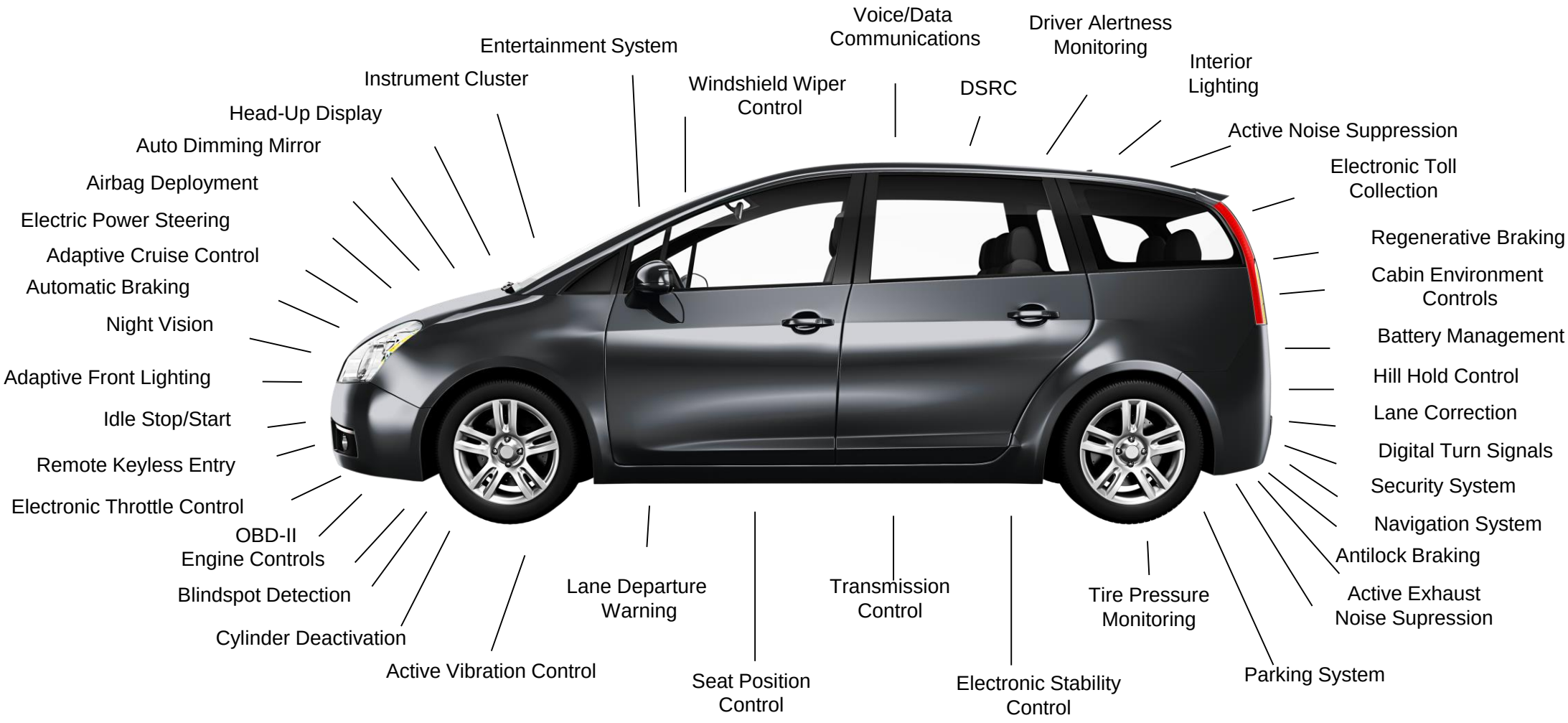
TUXERA

Solving data storage challenges in the automotive projects of tomorrow

Thom Denholm

Technical Product Manager

Automotive ECUs



Agenda

Ultra-quick review of flash memory technology

System trends and associated effects and challenges for Data Integrity

- How flash is affected
- Impacts from file systems

Flash memory revisited

NOR Flash
- Fast read times
- XiP: “eXecute in Place” – direct execution of code from flash
- Slower write and erase times

NAND Flash
- Slower read times
- Faster write and erase times
- Higher density, lower cost per bit

Managed Flash
- eMMC, UFS etc.
- Firmware handles wear leveling, bad block management
- Can contain RAM cache, microcontroller, and more

- Raw flash
  - Requires software to handle wear leveling & bad block management

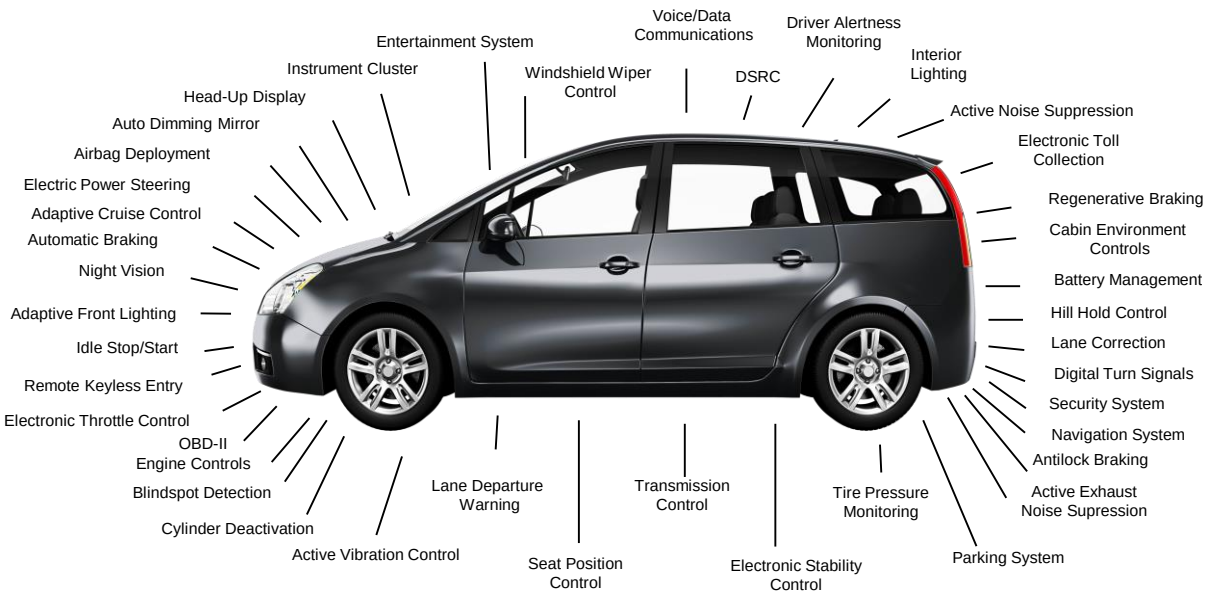
Flash memory revisited

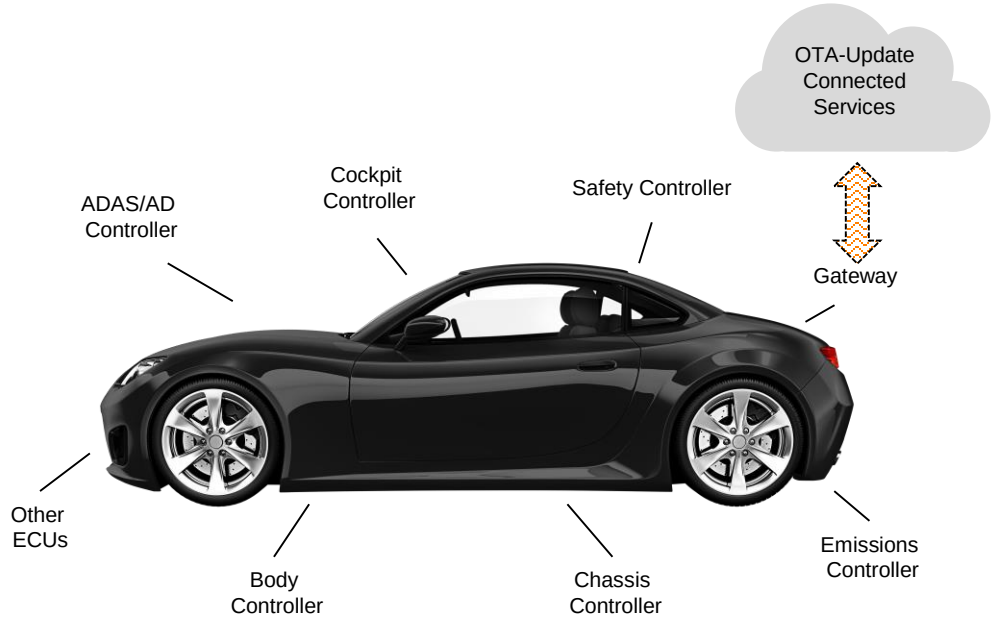
Evolution of NAND Flash

Flash memory revisited

	SLC	MLC	TLC	QLC
Bits per cell	1	2	3	4
Cost	$$$$	$$$	$$	$
Max P/E cycles	100,000	10,000	5,000	300-1,000

Trends in Automotive





30 to 100+ ECUs	6-8 operating systems
Flashed once, minor updates	Considered read-only
Predefined/fixed use case	Flash write is not an issue
Closed system	Power supply, cost
Moderate complexity, high controllability, scope well defined	

6-10 area/domain controllers	Hypervisor + 1 - 4 (RT)OS
Separated partitions by function and Tier 1 / OEM	Android -> new use cases
OEM Apps / new services	Always online, OTA updates
Open system	Flash write is an issue
High complexity, hard to define and enforce design rules	

Challenge #1 – System Integrity: Lifetime

What app will be running 5 years from now?

- How will it interact with the system and flash?

New Apps, OTA updates, DVR/Dashcam and EDR (Event Data Recorder) resulting in greater write load for flash

- Every single app attempts to log data -> performance, write cycles -> how to catch?
- Organizations struggle to define design rules and control them

How to achieve 10-15 years of lifetime for the system

- How to “correctly” dimension the system
- How to keep costs under control (SLC, MLC, TLC...)

Example: Tesla flash memory wear out, November 2019

- 4-year-old Model-S Central units failed due to too many write cycles

https://gizmodo.com/flash-memory-on-some-tesla-cars-is-reportedly- burning-o-1839084282

Do you know how much data gets written?

Write amplification factor

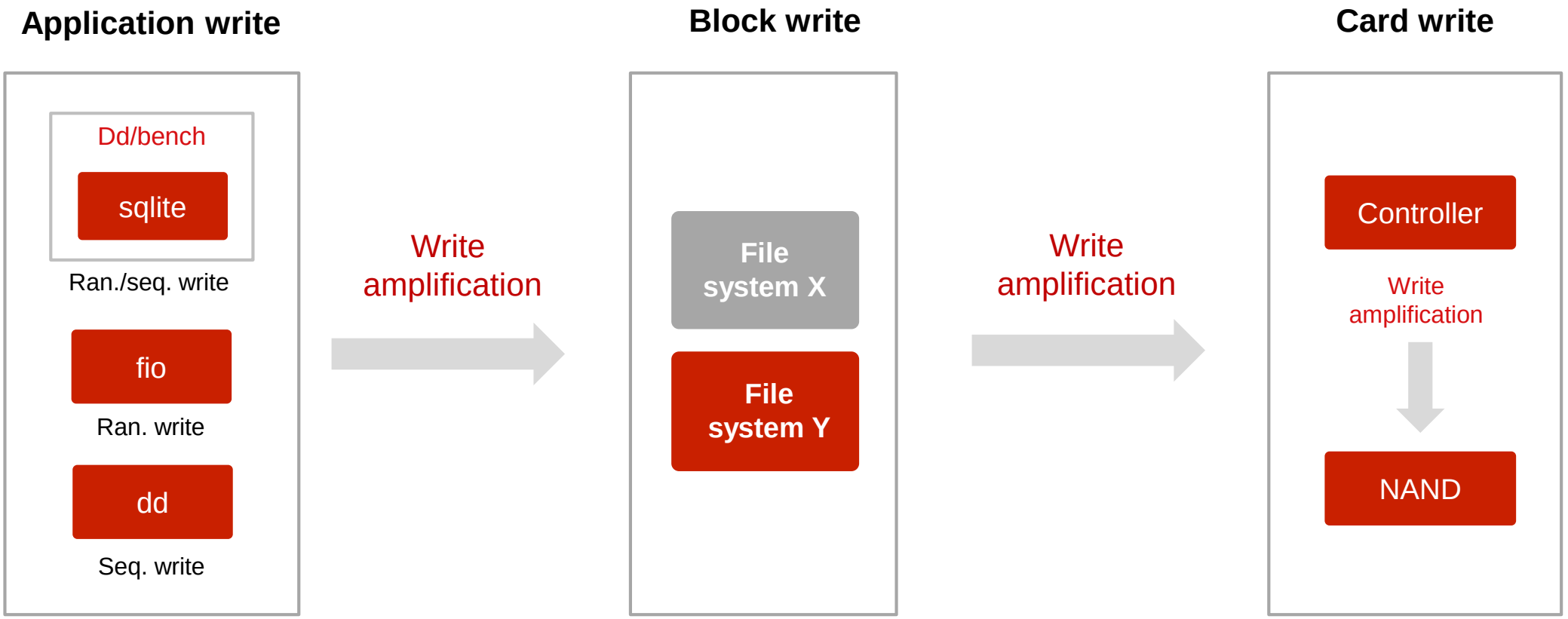
An undesirable phenomenon where the actual amount of physical information written is more than the logical amount intended.

$$\text{Write Amplification Factor} = \frac{\text{Media write}}{\text{Application write}}$$

Concerns

- Leads to flash memory wear-out
- Can cause critical system failure
- Sluggish performance (phones, automotive)

How write amplification multiplies

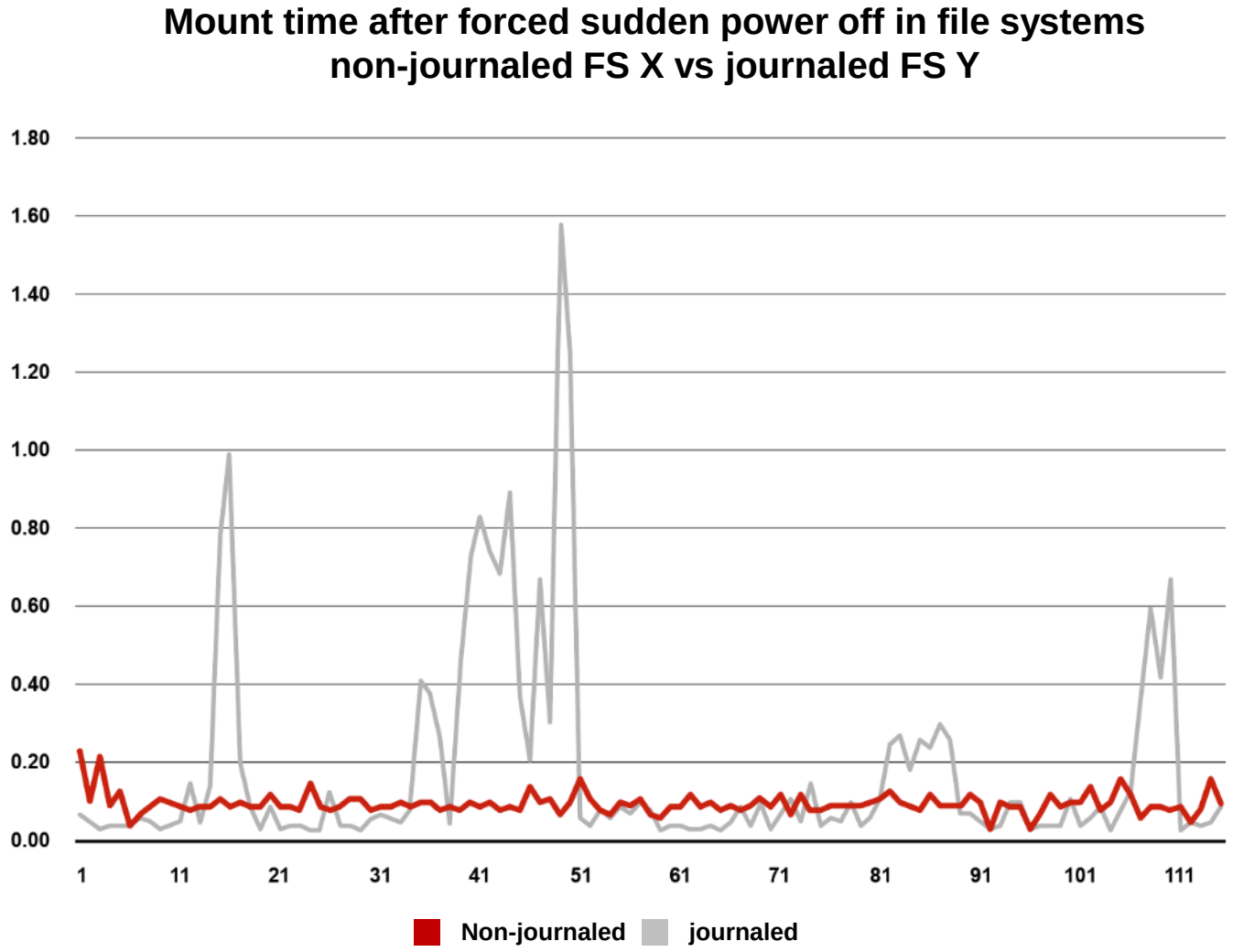


Android: observed WAF from 20 to over 100 - How many write cycles to plan for?

Boot time effects: fast and consistent mount times

2X faster average mount time than journaled FS Y

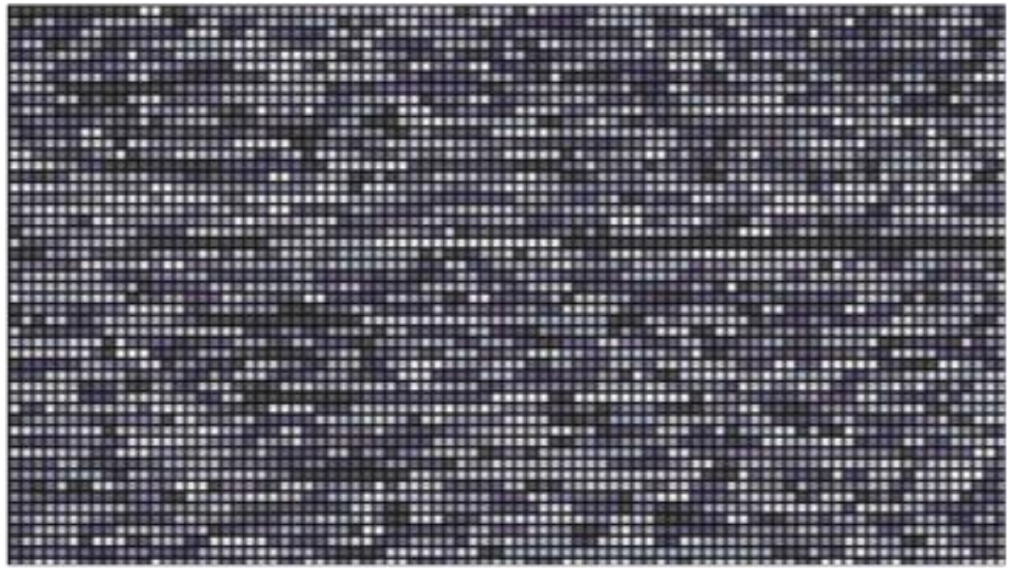
Tests performed on ARMv8-A Cortex-A53 Automotive SoC, 32 GB UFS storage.



Different fragmentation under long-term workloads

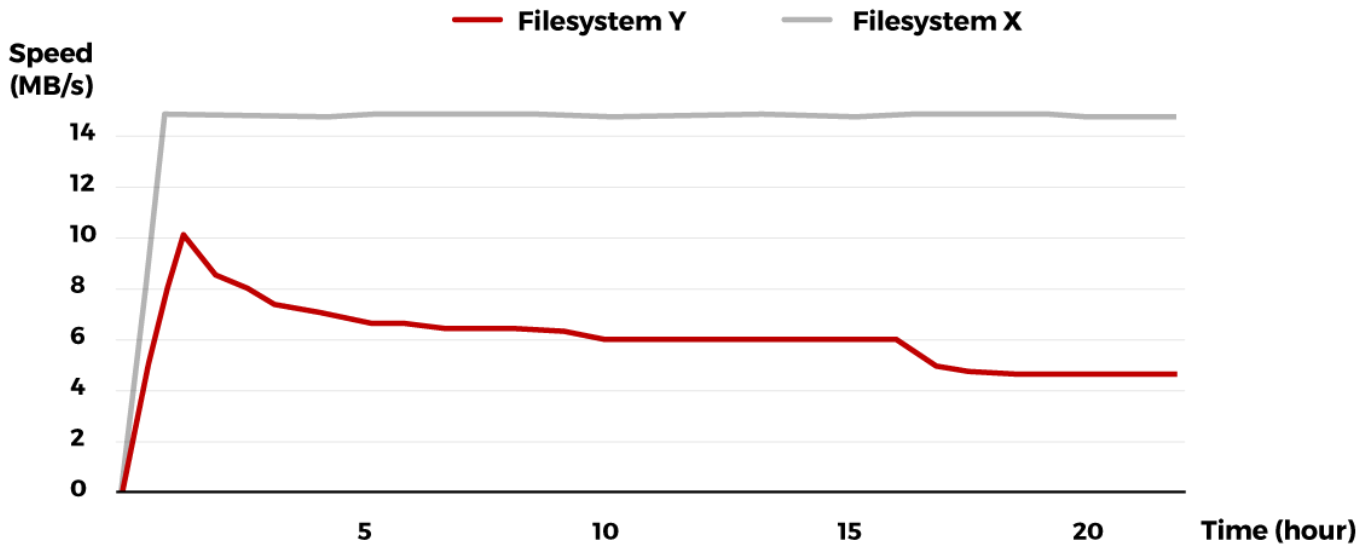
File system X

File system Y

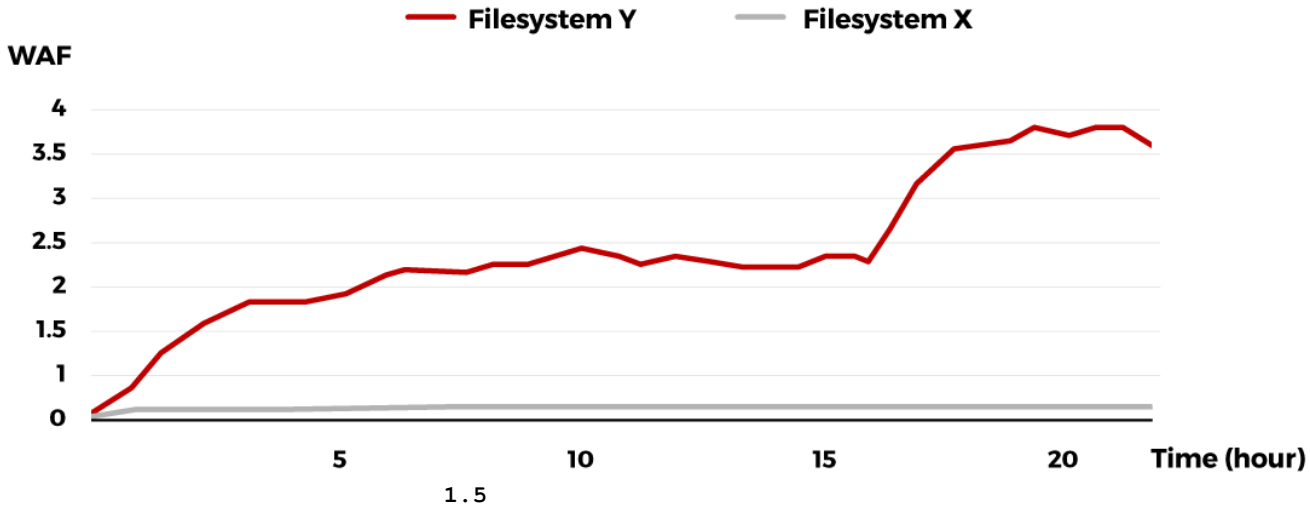


Over the long term

- FS Y performance drops
- erase amplification increases
- FS Y fragmentation gets worse
- Write performance drops
- Read performance drops



- Effects seen on mobile phones
  - 2-5x performance slowdown
  - 1.6-2x longer app launch time
  - Effect on Automotive ?



Challenge #2 – Reliability and performance

- Fragmentation slowing down the system
- WAF slowing down the system
- Enough system performance margin available over lifetime?
  - Perfomance drops of 20% can make systems simply fail
  - Can be a threat for safety critical systems
- Latency
  - Time that an application must wait until a storage operation is completed
  - Can cause frame loss in high quality video recording
    - – experienced in Drones, 4k recorders, relevant for dashcam recorders, DVR etc.

Challenge #3 – Other effects

Fail-safe operation

- Flash controller and file system should handle fails as effectively as possible
- Fail-safe file systems are not equal – there can be significant differences
  - Some file systems result in being corrupted and/or no longer mounting
  - Only upfront stress tests can reveal real behavior
- Example: FCA reboot endless loop for uconnect systems
  - Exact cause is not really known (disclosed)
  - Proper design and a capable file system would allow returning to the last known good state

    https://www.theverge.com/2018/2/15/17017946/fiat-chrysler- rebooting-screen-uconnect-problem

Application bugs

- Example: Spotify desktop bug from November 2016
  - Despite being idle, Spotify wrote 10-700 GB per hour onto the flash for no reason

    https://arstechnica.com/information-technology/2016/11/for-five-months-spotify-has-badly-abused-users-storage-drives/
  - Can this happen with any app?
  - How can we detect anomalies?
  - Bug or a cyber attack?

Conclusion

Flash memory and file systems have a huge impact on:

- Data integrity
- Lifetime of a system
- Performance
- Reliability and fail safety

New use cases with mobile phone-like flexibility create new challenges

- Increased write cycles onto flash cause resulting effects
- Lifetime of automotive systems is 4-5x higher than mobile phones

Thanks

Questions? Contact me:

Thom Denholm – thom.denholm@tuxera.com