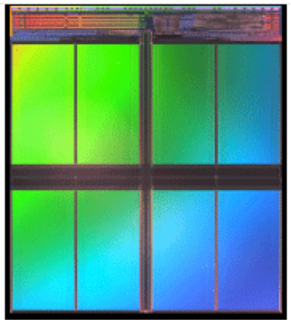


Managing NAND Flash to Optimize Product Longevity

Matt Porter, Chief Software Architect

Embedded Alley Solutions

Introduction / Agenda

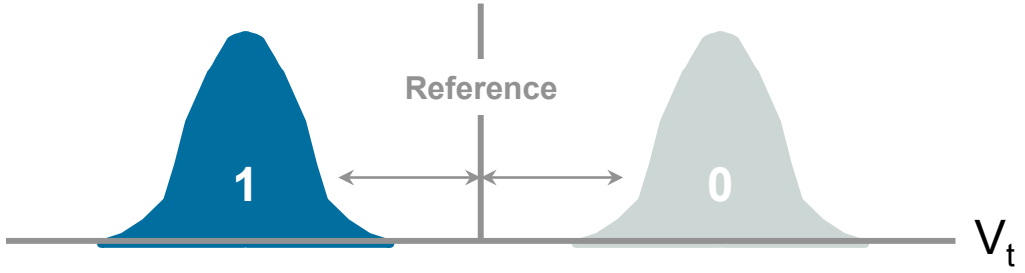
- Flash Design Challenge
  - NAND Flash read/write sensitivity (wear)
  - Performance over End-product lifetime
- NAND flash types
- Linux tools
  - MTD subsystem
  - File Systems
- Application Modelling
  - System Engineering Tool

NAND Flash - Key Choices

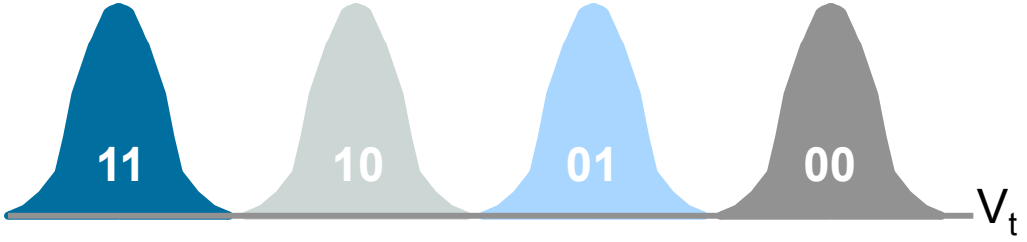
Two main types of NAND Flash

- Single-Layer Cell (SLC)
- Multi-Layer Cell (MLC)

SLC : 2 states, 1 bit

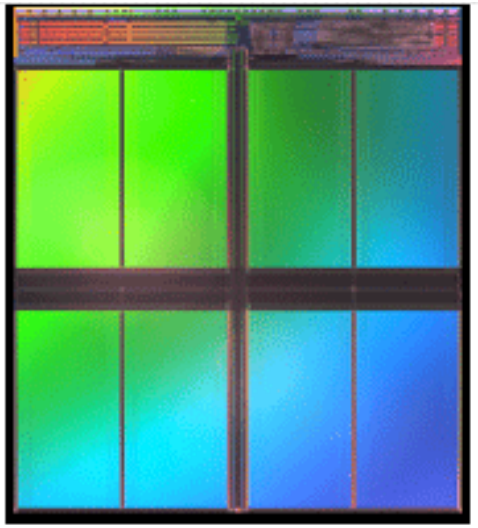


MLC : 4 states, 2 bits



NAND Flash Trade-offs

	SLC	MLC
States / Cell	2	4
Bits / Cell	1	2
Transfer Speed	High	Moderate
Write Cycles	~100K	~10K
Read Impact	No	Yes
ECC Required	Yes	Yes
Cost	Moderate	Low

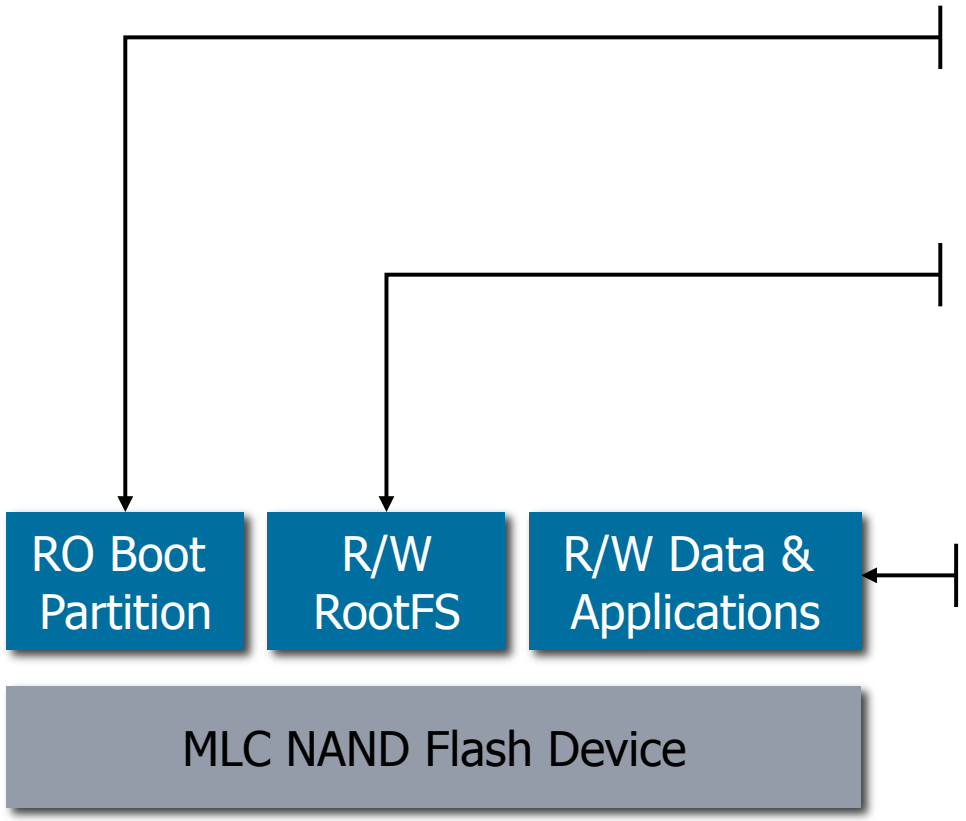


Supporting NAND in Linux

- Starting Point - Linux MTD and UBI
  - MTD : Memory Technology Device subsystem
  - UBI : Unified Block Manager

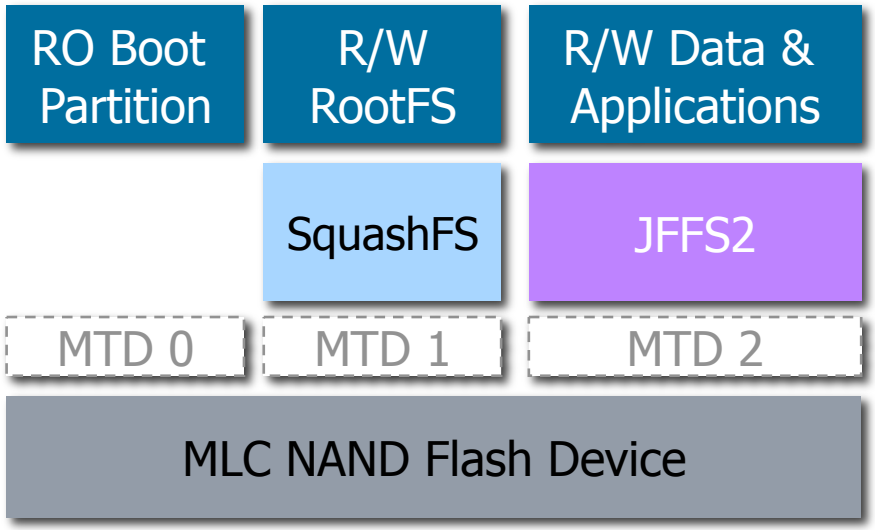
	R/W	Journal	Compressed	Notes
JFFS2	R/W	✓	✓	Supports both NAND and NOR Lengthy Startup / Mount
YAFFS2	R/W	✓		Designed for NAND, variable block
UBIFS	R/W	✓	✓	Fast File System built on UBI, not MTD
CramFS	R		✓	Supports Execute-in-Place
SquashFS	R		✓	Supports Big and Little-endian
UnionFS	R/W			Unifies other File Systems

Basic NAND Flash Platform



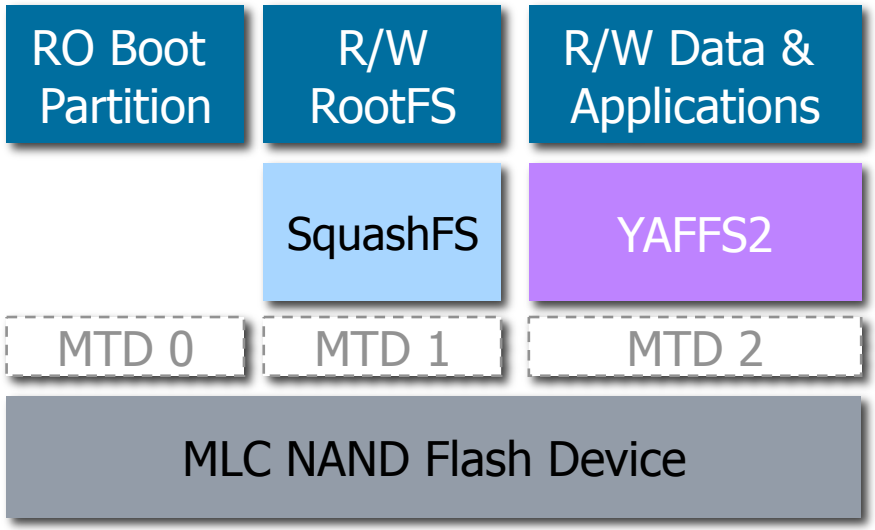
- Boot Partition
  - Read-only
  - Contains boot loader
- Root File System
  - Typically read-only
  - May be R/W for updates
  - SquashFS/CramFS candidate
- Data and Applications
  - Use R/W partition to save data and for updates
- Simple example excludes backup copy partition for update failover

Solution – MTD with JFFS2



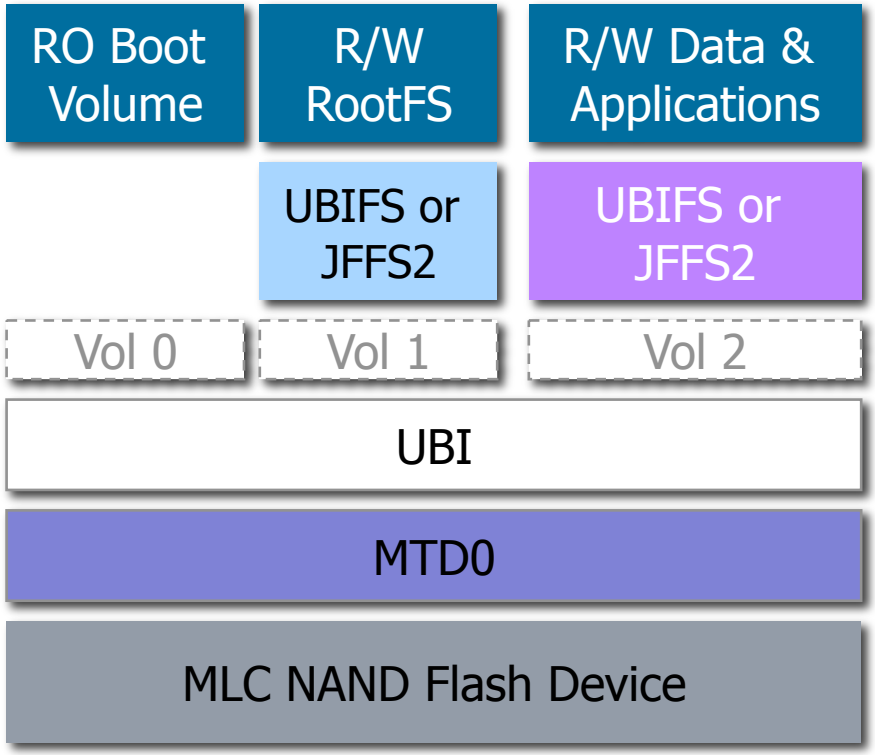
- Bad block management constrained by partition
- Wear-levelling algorithm uses random block replacement heuristic
- No read disturbance mitigation (for MLC)
- Slow mount time due to JFFS2 journal design
- Metadata can conflict with ECC (depending on size)
- Compressed file systems

Solution – MTD with YAFFS2



- Bad block management constrained by partition
- Non-deterministic wear-levelling algorithm
  - Reserves spares for bad block replacement and random block moves
- No read disturbance mitigation (for MLC)
- Faster mount time than JFFS2
- Out of sync meta data can lead to FS inconsistency
- No compression in YAFFS2

Solution – UBI-based File Systems



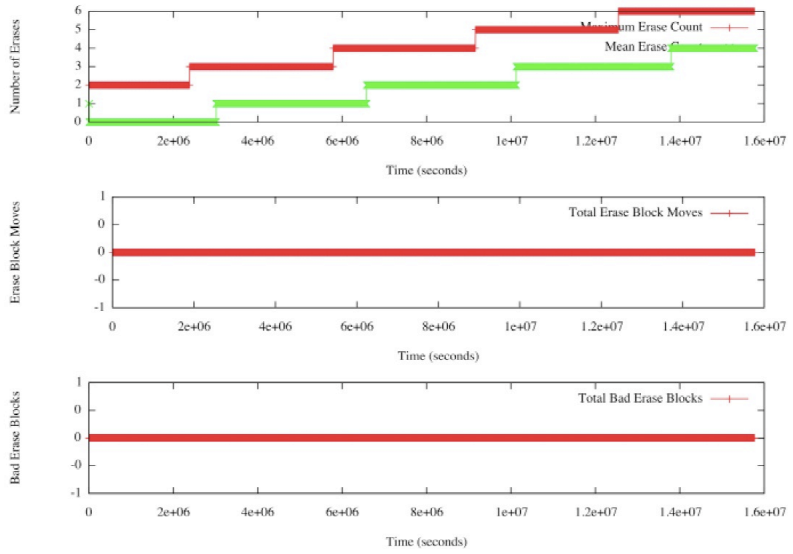
- Separates wear-levelling from file system
  - Bad block management and wear-levelling performed across entire device
- Volumes instead of partitions
- UBI supports both UBIFS and JFFS2
  - A block layer on UBI could handle any file system (FAT and ext3)
- Deterministic wear-levelling
- UBI block management triggered by reads and writes

Characterizing Application Behavior

- Usage Patterns over Product Lifetime
  - Average hours of use per day
  - Read/write/erase sequences in typical use
  - Device lifetime expectancy
- Software Read / Write Behaviors
  - Application operation (NAS, media storage, etc.)
  - Updates
  - Data logging
  - User preferences and data files
  - Performance requirements

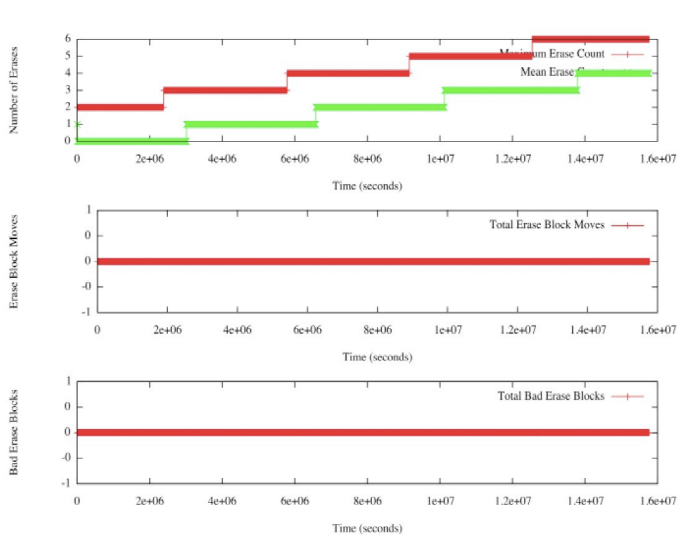
Application Modelling

- Model Application / Device
  - Execute with simulation or actual hardware
  - Collect statistics from logs, instrumented code
- Analyze Results
  - Design meet performance requirements?
  - Provides specified product longevity?



Modelling without Hardware

- NAND simulator (NANDsim)
  - Simulate flash configurations comparable to those deployed in end product
  - Adjust NAND size, page size, sub-page capability

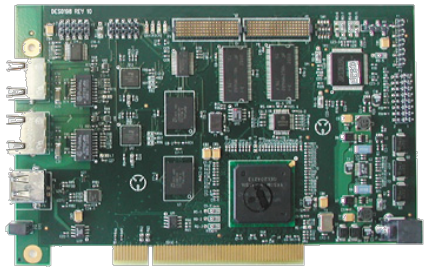


- Test applications
  - Try combinations of file systems and hierarchies
  - Inject ECC errors and write errors
  - Gather fine-grained block wear statistics

Modelling with Actual Hardware

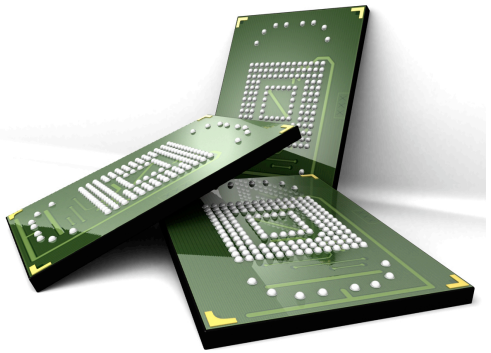
- Production hardware or reference board
  - Target comparable (sub)sets of flash, DRAM, etc.

- Gather NAND statistics using kernel driver interfaces
  - UBI provides several statistics
    - Maximum erase count
    - Mean erase count
      - New, not upstream yet!
    - Erase block moves
    - Bad erase blocks
  - JFFS2 and YAFFS2 require additional instrumentation



Modelling with Actual Hardware
(continued)

- Managed NAND
  - Self-contained flash subsystem accessed via dedicated interconnect / bus
    - Examples include SD and MoviNAND
  - MLC NAND inside
- Modelling Challenge
  - Managed NAND devices are black boxes
    - No statistics attainable through h/w interfaces
    - Could extend MMC subsystem to provide insight into ECC info, but poor ROI
  - Application model run for product lifecycle
    - Success indicated by managed NAND survival



Applying Application Modelling

- Example - Personal Navigation Device (PND)
  - Features
    - GPS-based navigation
    - Retrieves and displays detailed road maps
    - Routes to destinations based on current location
    - Can search among Points of Interest (POIs)
    - Provision for updates to POIs and maps
  - CE Product Lifetime
    - Minimum 3 years
  - Usage pattern
    - Typical 4 hours use per day

Applying Application modelling
(continued)

- PND Hardware Profile
  - LCD
  - Touchscreen
  - Audio output
  - USB gadget
  - Raw NAND only
    - MLC NAND
    - 4GiB total

PND I/O modelling : Writing

- Gather Write Characteristics
  - Application Logs – 100 bytes/second
  - Syslogd – 50 bytes/second
  - Address Book, POIs, search history – 2.3KiB/4 hrs
  - Temporary space – 1KiB/second
  - Uncompressed map data – 6KiB/second
  - Map data – 3 GiB/quarter
  - Update OS/Apps – 32MiB/quarter
  - Update temporary space – 100MiB/quarter

PND I/O modelling : Reading

- Gather Read Characteristics
  - Address book, POIs, Search history –64KiB/4 hrs
  - Temporary space – 1KiB/second
  - Compressed map data – 3KiB/second
  - Uncompressed map data – 6KiB/second
  - Update temporary space – 200MiB/quarter

Applying I/O Characteristics

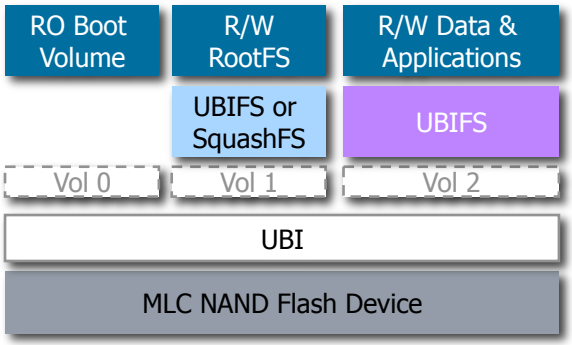
- OEMs, Integrators Use Data to
  - Validate designs, assumptions
  - Provide input to test/QA
  - Develop custom modelling scripts, applications
- Architecture
  - One thread per I/O job
  - Read and write jobs fully exercise NAND
  - Meet I/O timing requirements by managing job execution on a per thread time schedule

Applying I/O Characteristics
(continued)

- Benefits
  - Supports host-based testing
    - Use native host file system to debug models, tools
  - Accelerates complete lifecycle test
    - 100x rate for generating I/O transactions covers typical 3 year lifecycle in days or hours

File System Design Conclusions

- Design Deploys MLC NAND
  - Best choice is a UBI-based hierarchy
  - Read/write file system is UBIFS
  - SquashFS may be used if needed
    - Run on emulated MTD UBI volume
- Use UBIFS/UBI hierarchy as candidate file system layout
  - Rootfs volume
  - Application volume
  - Map data volume



Modelling on NANDsim

- Test candidate UBI file system hierarchy on host
  - Limit the amount of hardware destroyed
  - Determine whether a candidate file system is worth testing on real hardware
  - Tests run faster on a high-end workstation
  - NANDsim provides detailed per block statistics
  - Fast turn-around when testing different file systems
    - Provides identical statistics for each file system type

PND Modelling on Actual Hardware

- NANDsim run is good, so on to actual hardware!
- Script needed to gather UBI statistics
  - Could also come from enhanced JFFS2 or YAFFS2 interfaces
  - Gathers max ECs, mean ECs, EB moves, bad blocks at regular intervals
  - Dumps to delimited text file for post processing
- Modelling tool and statistics script run together
  - Erase MTD device and UBI volumes to zero statistics
  - Execute for equivalent of 3 year, 4 hrs/day product cycle
    - Accelerated, of course!

Analyzing Modelling Data

- Modelling Run Outcome
  - Pass/Fail : Did the hardware survive?
    - How to meet product lifetime requirements?
  - Detailed statistics log
    - Data tracks NAND part approach to failure state
    - Supports planning for additional feature feasibility
    - Provides feedback into system engineering processes
- Iteration
  - Custom modelling can test multiple scenarios
    - How to extend product life-time by cutting write-intensive features and functions?

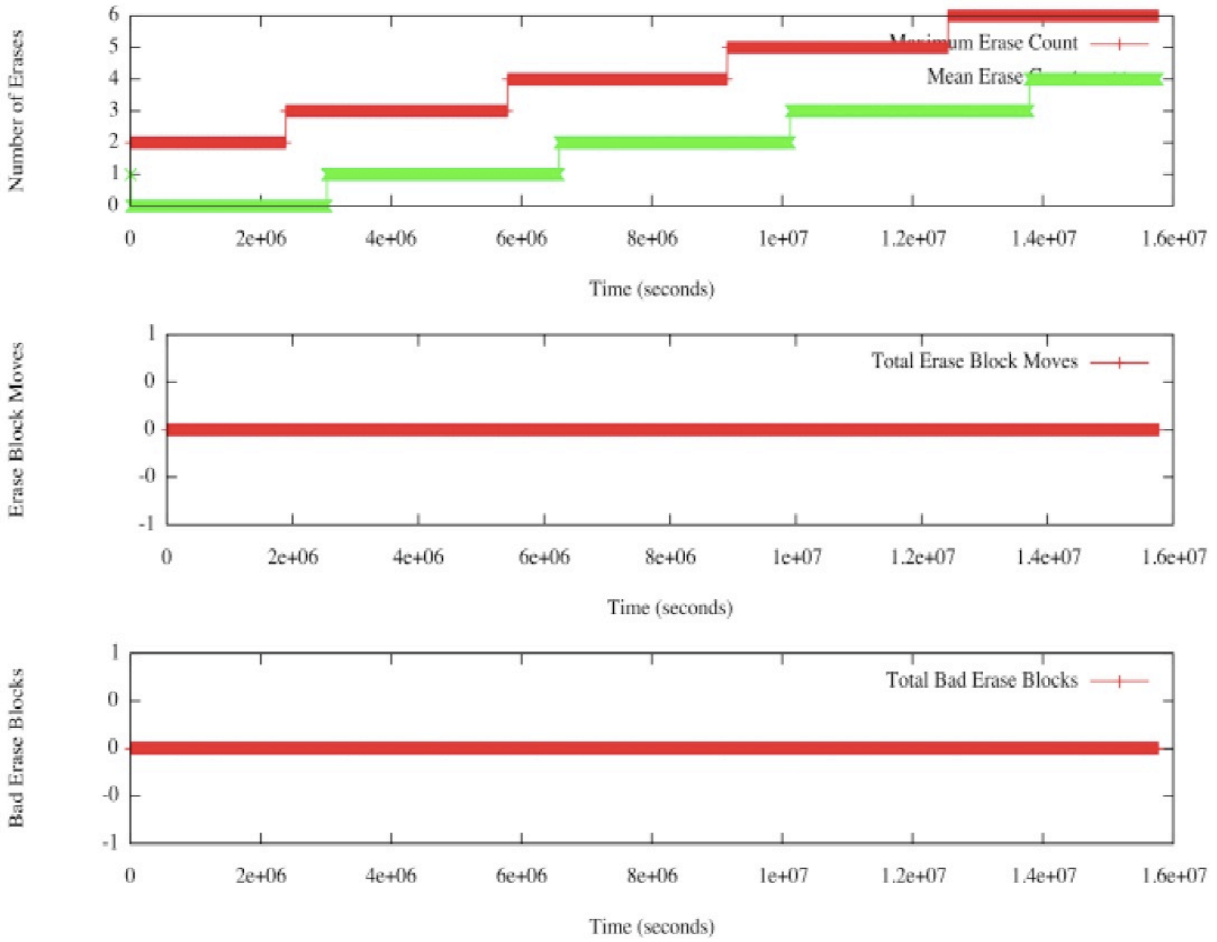
Presenting Modelling Data

- Raw data not easily analyzed
- Data best post-processed for design review
  - Compare modelling runs

Time	Max EC	Mean EC	EB Moves	Bad EBs
3600	2	1	0	0
7200	2	1	0	0
		. . .		
151200	3	1	0	0
154800	3	1	0	0

Presenting modelling Data

Gnuplot or other plotting s/w yields compelling visual representation of modelling data



Conclusion

- For PND use case,
  - UBI+UBIFS is ideal for managing MLC NAND wear
- Application I/O modelling is a useful system tool
  - Helps with NAND selection, assists in application design
  - Allows early decisions on file system choice and hierarchy
  - Valid use cases are key
    - Garbage in – Garbage out
- Future
  - Modelling process also applicable to CPU and RAM
  - Comparable statistics gathering can assist in this process

Q & A