Active Flash Management Using Fully Autonomic Control

Conor Ryan
CTO - Software
Conor.Ryan@NVMdurance.com
Take Home Messages

- Routine *Normalized* intrinsic endurance increase of up to 7X in addition to other known approaches
- Multiplies other endurance methods and delivers up to 25X gain over default specifications
- Finding good control parameters is just the start…
  - Flash must be *actively managed* to minimize guard banding (due to variation)
- Active management must be *fast* at enterprise level, especially when doing
  - LDPC
  - Read retry
- Excellent cost/benefit ratio
  - BCH ECC; TLC Flash; Low tail latency; High Endurance
Take Home Messages

- Routine Normalized intrinsic endurance increase of up to 7X in addition to other known approaches
- Multiplies other endurance methods and delivers up to 25X gain over default specifications
- Finding good control parameters is just the start…
  - Flash must be actively managed to minimize guard banding (due to variation)
- Active management must be fast at enterprise level, especially when doing
  - LDPC
  - Read retry
- High Cost/benefit ratio
  - BCH ECC; TLC Flash; Low tail latency; High Endurance

New results
- 7X increase in endurance
- 1Y nm TLC NAND
- No read retry
- High endurance
- Low latency
SSD Desirable Characteristics

Fast ✔

Cheap ✔

Good ✔

It depends on who you ask!
Business always wants cheaper SSDs

Fast ✔

Cheap ✗

Good ✔

Higher density, lower geometries required
Cheaper flash is harder to manage

Low endurance; read retry required; higher variation
More management costs

Cheap ✗

Fast ✗

Extra work/machinery required costs time and money

Good ✗
Cost/benefit trade off

Cost vs. Endurance graph
Cost/benefit trade off

- Cheaper flash
  - Less endurance
  - More effort to recover data

Density vs. Endurance
Cost vs. Endurance

# read retries vs. Endurance
Degradation of Flash

- The two well-known killers of flash
  - Endurance
  - Retention
- Secret killer of SSDs
  - Tail Latency (99th percentile of response time)
  - “Sure, you can get your data back, but it’s going to cost you…”
- 1Y TLC
  - 700 p/e cycles
  - 12 months retention
  - 20+ read retries..
  - What if there is just ONE read retry?
Degradation of Flash

Endurance
400 p/e
Retention
1 week
Degradation of Flash

Endurance
400 p/e
Retention
2 weeks
Degradation of Flash

Endurance
400 p/e
Retention
3 weeks
Degradation of Flash

Endurance
500 p/e
Retention
1 week

© NVMdurance 2015
Degradation of Flash

Endurance
500 p/e

Retention
2 weeks
Degradation of Flash

Endurance
500 p/e

Retention
3 weeks
Degradation of Flash

Endurance
600 p/e
Retention
1 week
Degradation of Flash

Endurance
600 p/e
Retention
2 weeks
Degradation of Flash

Endurance
600 p/e
Retention
3 weeks
Degradation of Flash

Endurance
700 p/e

Retention
1 week
Degradation of Flash

Endurance
700 p/e
Retention
2 weeks
Degradation of Flash

Endurance
700 p/e
Retention
3 weeks
Avoiding data loss

No read retry means more guard-banding

700 p/e cycles at one year retention!
No read-retry, more guard-banding

- Three pronged problem of using lowered geometry and increased density
  - Less endurance
  - Less retention
  - More effort to read data
- LDPC?
  - Powerful, but slow and costly
    - Up to 60% more gates required in client SSDs
Speed of read retry

► Successfully read a page:
  ► Read data into buffer (100µs)
  ► Toggle data out (50µs)
  ► ECC (approx. 50µs)
  ► Total: 200µs

► Each read retry adds:
  ► Change parameters (~nanoseconds)
  ► Repeat the process
Two steps to endurance

- NVMdurance Pathfinder
  - Discovers the endurance gain - the “Potential” of the Flash
  - Suite of Machine Learning algorithms
  - Determine optimal registers for NAND chips before they go into product

- NVMdurance Navigator
  - Exploits the Pathfinder discoveries - delivers on the potential
  - Autonomic system running on controller
    - Manages chip-to-chip variation down to the block level
    - Chooses register values at run-time from those discovered by Pathfinder
Two steps to endurance

- **NVMDurance Pathfinder**
  - Discovers the endurance gain - the “Potential” of the Flash
  - Suite of Machine Learning algorithms
  - Determine optimal registers for NAND chips before they go into product

- **NVMDurance Navigator**
  - Exploits the Pathfinder discoveries - delivers on the potential
  - Autonomic system running on controller
    - Manages chip-to-chip variation down to the block level
    - Chooses register values at run-time from those discovered by Pathfinder
NVMdurance Pathfinder

- Discover optimal parameter sets for each stage of life
  - Gradually increase the “program/erase stress”
  - i.e. get increasingly more aggressive throughout life

- What is least amount of damage that we can cause at the start of life such that the flash is still operational at the end of life?

- What is the best read register set to use for this stage?
  - Doesn’t need to rely on read retry
  - Can use it if available
Pathfinder -- Stages

Early  Middle  Late
Pathfinder

Raw Bit Error Rate vs P/E Cycles

- ECC Limit

Default
- NVMdurance

Lifetime achieved as multiple of intrinsic endurance
Pathfinder

Raw Bit Error Rate vs P/E Cycles

- ECC Limit
- Default
- NVMdurance

Lifetime achieved as multiple of intrinsic endurance
Pathfinder

Stage changes

Window
Pathfinder

Does BER go down?
Only as a consequence of stronger writes
Keep running?
Pathfinder -- Timings

Page Program Time vs P/E Cycles

- **Target**
- **Average**

**Default**

**NVMealurance**

Lifetime achieved as a multiple of the intrinsic endurance
Pathfinder -- Timings

Page Program Time vs P/E Cycles

Average Program Time [us]

Lifetime achieved as a multiple of the intrinsic endurance

Stage changes
NVMdurance Results

- All results are volume tested in hardware
  - All backed up by real data
- Normalized results
  - Baseline calculated as intrinsic endurance at same retention level
  - Same level of ECC available
  - All increases are solely due to Pathfinder-discovered parameters
  - All assume presence of NVMdurance Navigator on the SSD
NVMdurance Results

Vendor 1: 41nm/40bit ECC

Normalized intrinsic endurance using factory parameters

Lifetime achieved as a multiple of the intrinsic endurance
NVMdurance Results

Normalized intrinsic endurance using factory parameters

Lifetime achieved as a multiple of the intrinsic endurance

Vendor 1: 41nm/40bit ECC
Vendor 1: 20nm/40bit ECC

© NVMdurance 2015
NVMdurance Results

Normalized intrinsic endurance using factory parameters

Lifetime achieved as a multiple of the intrinsic endurance

Vendor 1: 41nm/40bit ECC
Vendor 1: 20nm/40bit ECC
Vendor 2: 19nm/40 bit ECC
NVMdurance Results

Normalized intrinsic endurance using factory parameters

Lifetime achieved as a multiple of the intrinsic endurance

Vendor 1: 41nm/40bit ECC
Vendor 1: 20nm/40bit ECC
Vendor 2: 19nm/40bit ECC
Vendor 2: 19nm/120 bit ECC
NVMdurance Results

Normalized intrinsic endurance using factory parameters

Lifetime achieved as a multiple of the intrinsic endurance

Vendor 1: 41nm/40bit ECC  
Vendor 1: 20nm/40bit ECC  
Vendor 2: 19nm/40 bit ECC  
Vendor 2: 19nm/120 bit ECC  
Vendor 1: 16nm/70 bit ECC
NVMdurance Results

Approx. 7X increase in intrinsic endurance consistently achieved across:
- 2 vendors’ devices
- 3 different geometries
- 3 different ECC levels
- Both MLC and TLC

Normalized intrinsic endurance using factory parameters

Lifetime achieved as a multiple of the intrinsic endurance
Tail Latency

- Tail of the distribution of response times

Worst case response time
Tail Latency

- Tail of the distribution of response times

Worst case response time:
- Longer time
- Greater variation
Tail Latency

Tail of the distribution of response times

Worst case response time
- Longer time
- Greater variation

Probability
Response Time
How fast can it be?

- Tail Latency for request queue

- Subsequent reads are delayed
  - Clever write management techniques can mitigate this
    - Caches, weak writes, etc.
    - Not without their own issues: Power failure recovery, Stronger ECC, etc.
Read time variation

- Tail Latency for request queue
  - Gets worse later in life
    - Higher variation
    - Spread of retention

- Choices?
  - Reread until pass?
    - Delays subsequent reads
  - Reinsert read into pipeline
    - Data delayed and/or breaks pipelined instructions
  - Don’t ever fail
    - Not quite the impossible dream it might first appear!
Stages

- Life divided into stages
- Each stage has a set of program/erase registers
  - PLUS a set of read registers specific to that stage
  - “Wear-sensitive read”
Stages

When should we change stages?

- If all flash was the same (no variation) we could do it based on cycles
Stages

- **Guard banding?**
  - A significant amount of the endurance gains will be lost

- **Read retry/LDPC impact?**
  - Overlap would be less, but read time would be impacted at thresholds
Stages with stronger ECC/read retry

- Each stage lasts longer
  - Less overlap
Health, not cycles

Change stage when the “health” of the flash has degraded enough

- “Health Reading”
- BER
  - Absolute and intermediate levels -- “Soft Error Thresholds”
- Operation times
  - Read, write, erase
- When the health of the device dictates; we change stage
- Change based on original Pathfinder training samples

- “Wear-leveling on steroids”
  - We don’t just CYCLE blocks equally, we manage DEGRADATION equally
NVMdurance Navigator

- Wear blocks as much as possible
- Rest outliers as needed
- Change stage when LUN’s health has degraded
- “From each according to his ability, to each according to his needs”
  - Don’t target the WORST block, target ALL blocks
- Extract the full potential of Pathfinder parameter sets by
  - Tracking outliers
  - Tracking degradation
  - Change at last possible moment
Navigator Data

- Track health by setting various thresholds
  - BER
  - Operation timings

- Controller informs Navigator of **Threshold Violations (TVs)**
  - E.g. ECC reports read over “soft threshold” or “critical threshold”

- Navigator monitors
  - Number of TVs for each threshold
  - Levels of each threshold
  - Number of TVs caused by each block
Navigator Actions

- **Block actions**
  - Rest a block that is causing too many TVs
  - Add block to bad block list that is repeatedly causing problems

- **LUN actions**
  - Raise thresholds; gives detailed information on how LUN is degrading

- **Stage actions**
  - Too many TVs at high thresholds; change stage
  - Health close to change point; change stage
  - Cycles close to validated level; change stage
Navigator Internals

Threshold Violations and Health Metrics from STD Flash Controller Firmware

INFERENCE ENGINE

Health Metric Analyzer
Decision Trees

Command Constructor

History DB
Operating Parameter DB

NAVIGATOR

Navigator Commands to STD Flash Controller Firmware
Health Metrics

- Threshold levels
  - BER; tProg; tRead, etc.

- TVlist
  - Size; rate

- Resting blocks
  - Possible outliers

- Block details
  - TV rate

- Historical data
  - Previous stages
Health Metrics

- Threshold levels
  - BER; tProg; tRead, etc.

- TVlist
  - Size; rate

- Resting blocks
  - Possible outliers

- Block details
  - TV rate

- Historical data
  - Previous stages

- Stage Actions
  - Change stage

- LUN Actions
  - Change reporting rate

- Block Actions
  - Suspect block; move data
  - Possible outlier; rest block
Health Metrics

- Threshold levels
  - BER; tProg; tRead, etc.

- TVlist
  - Size; rate

- Resting blocks
  - Possible outliers

- Block details
  - TV rate

- Historical data
  - Previous stages

- Stage Actions
  - Change stage

- LUN Actions
  - Change reporting rate

- Block Actions
  - Suspect block; move data
  - Possible outlier; rest block
  - Probable outlier; bad block
  - Etc. etc. Many actions possible!
Costs

- **Memory footprint**
  - DRAM 300 bytes per block
  - NV storage 75 bytes per block
  - Total per LUN (4096 blocks)
    - $4096 \times 300 = 1200$KB ~ 1MB

- **Minimal configuration**
  - DRAM zero bytes per block
  - NV zero bytes per block

- **Code footprint**
  - 2500 lines of code
  - Less than 10KB compiled

- **CPU usage**
  - Dual core 1.5Ghz Cortex A9
    - Peak: 0.6%
    - Average: 0.5%
  - Single core 700Mhz ARM 11
    - Peak: 1.2%
    - Average: 0.78%
Activity

- Level of Navigator activity varies by time of stage

- Bursty activity

- More conservative
Navigator in operation

- **Threshold Levels**
  - BER level that is considered to be a Threshold Violation (TV)

- **Soft Errors**
  - Housekeeping data; no action required

- **Critical Errors**
  - Some action required
    - Move data
    - Rest block

- **TV rate**
  - Proportion of reads causing TVs
Navigator Snapshots - zero retention

- Soft Error Threshold: 20
- Critical Error Threshold: 55
- Cycles: 500
- TV rate: 54%
- Action rate: 0.03%
- Block TV level: 100%

Threshold Violations
Navigator Snapshots - zero retention

- Soft Error Threshold: 30
- Critical Error Threshold: 55
- Cycles: 500
- TV rate: 0.34%
- Action rate: 0.03%
- Block TV level: 17%
Navigator Snapshots - medium retention

- Soft Error Threshold: 30
- Critical Error Threshold: 55
- Cycles: 500
- TV rate: 0.31%
- Action rate: 0.19%
- Block TV level: 28%
Navigator Snapshots - high retention

- Soft Error Threshold: 30
- Critical Error Threshold: 55
- Cycles: 500

- TV rate: 0.40%
- Action rate: 0.34%
- Block TV level: 38%
Summary

- Level of interaction tunable through thresholds
  - Low thresholds, richer information, more traffic
  - Mid-level threshold, little traffic (<1% of reads)
- Tiny minority require action on part of the controller
- As we approach stage change, rates increase
  - TV rate (constrained by Controller)
  - Critical (constrained by number of restable blocks)
  - Health (too many TVs at top threshold causes stage change)
Abstract Flash Trimming

- Access to test modes required
- Use Abstract access if necessary
  - "Blind" access to registers
- NVMdurance provides interface template
  - Connects high level functionality to flash at abstract level
    - E.g. registers r1...rX
  - Works because Pathfinder learns RELATIONSHIPS between registers
- Flash foundry implements lookup table
- Lookup table encrypted with one-way encryption
Typical NV Mdurance set up

Preproduction

NV Mdurance Pathfinder

Characterization

Flash
SSD with no access to trim settings

Preproduction
NVMdurance Pathfinder

Characterization
Flash

Parameter Sets

Deployment

SSD
Controller
Register Settings

To host

NVMdurance Navigator

© NVMdurance 2015
No access to trim information

Preproduction

NVMdurance Pathfinder

Characterization

AFT

Flash

Parameter Sets

Deployment

SSD

Controller

Register Settings

AFT

Bus

Flash

To host
<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device</td>
<td>1Y nm TLC</td>
</tr>
<tr>
<td>Number of stages</td>
<td>3</td>
</tr>
<tr>
<td># Read retries</td>
<td>1</td>
</tr>
<tr>
<td>Retention</td>
<td>3 weeks</td>
</tr>
<tr>
<td>ECC</td>
<td>70 bits per sector</td>
</tr>
<tr>
<td>Intrinsic Endurance</td>
<td>400 cycles</td>
</tr>
<tr>
<td>Max Cycles (current)</td>
<td>2800</td>
</tr>
</tbody>
</table>
Product shipping soon

Altera Extends Life of NAND Flash Storage with Arria 10 FPGAs

- Altera reference design
  - Proof of concept will have 3X endurance, we are targeting 10X

- Features
  - FPGA, fast and cheap to develop
  - Field upgradeable/reconfigurable to read/write mix
  - Firmware runs in hardware
  - Interchangeable NAND hardware (including mixed)

- Up to
  - 9.6GBps of Bandwidth
  - 1875 KIOPS
  - 24 Flash Channels

- Fast, cheap and very, very good!
Conclusions

► Automatic and Autonomic

► Machine Learning automatically discovers
 ► parameter sets
 ► static parameter sets
 ► the level of endurance that the flash can attain

► Lightweight software running on SSD autonomically
 ► Manages degradation of flash
 ► Minimizes tail latency by removing need for read retry
 ► Actualizes the endurance realized by NVMdurance Pathfinder
Final Remarks

- Industry leading endurance gains
- Demo running at our booth
- Altera board with NVMdurance software at their booth

Stop by and see us at booth #829
Conor.Ryan@NVMdurance.com