ZombieNAND: Resurrecting Dead NAND Flash for Improved SSD Longevity

Ellis H. Wilson III [1,2] Myoungsoo Jung [3] Mahmut Kandemir [1]

[1]Department of Computer Science and Engineering,
The Pennsylvania State University

[2]Panasas, Inc.

[3]Department of Electrical Engineering,
The University of Texas at Dallas

September 10th, 2014

Before We Begin: Get the Slides and Paper

Slides and Paper are Available At:

www.ellisv3.com

Contents: I

The Present and Future of Flash

Well-Known Flash Dynamics

- SLC: Fast, Long Life, Small Size
- MLC: Medium, Medium Life, Medium Size
- TLC: Slow, Short Lived, Large Size
- Cells are getting smaller (i.e., slower, shorter-lived)!

Future Flash: As consumers push towards higher-capacity and SSDs slowly replace HDDs, *longevity will return to the forefront of the discussion*

Leveraging the Little-Known

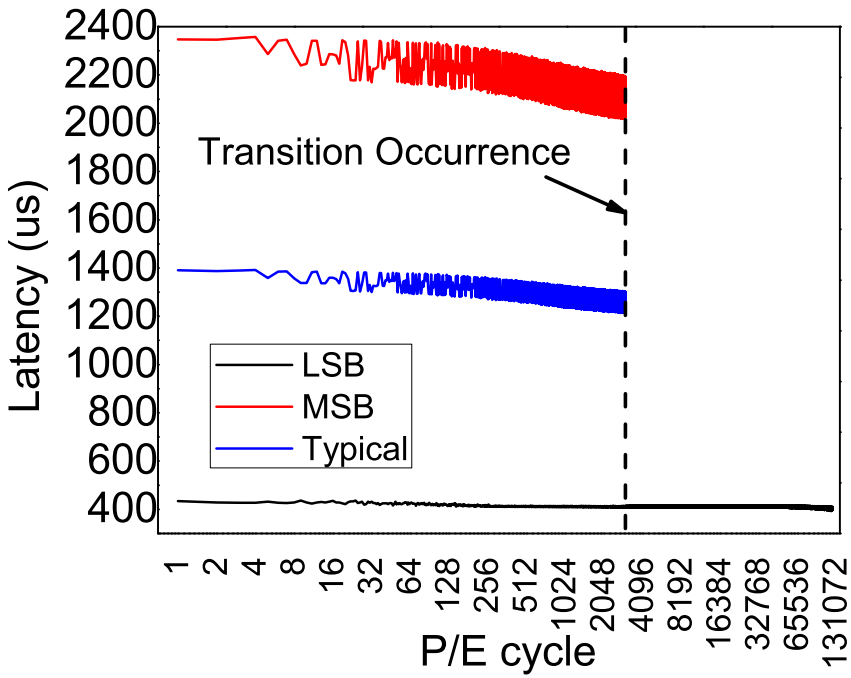
A Little-Known Flash Fact

- SLC, MLC, and TLC are solely logical differentiations
- **Same underlying NAND material!**

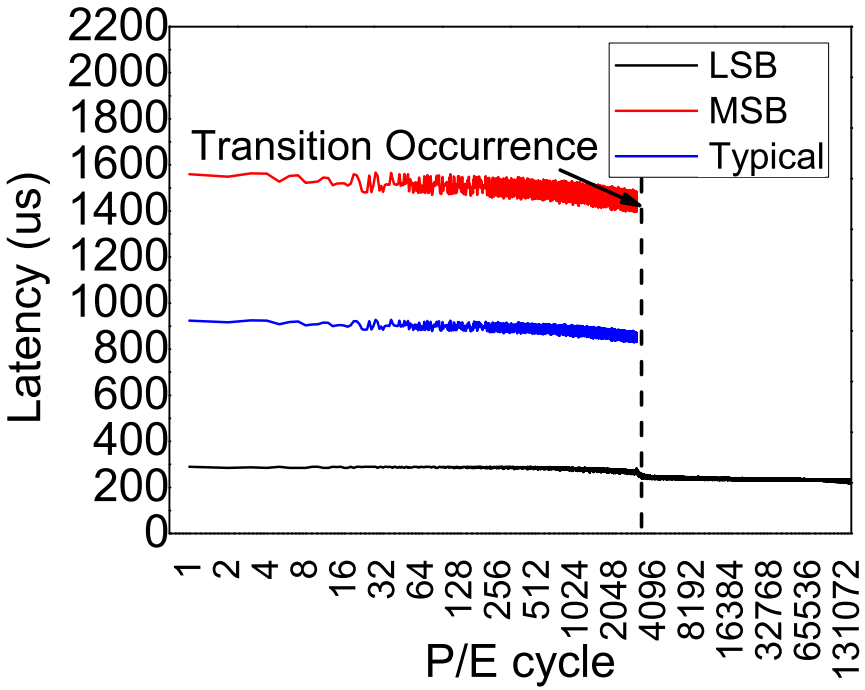
Our Question: Given impending longevity concerns and increasing NAND diversity, can we develop a scheme that will increase longevity *without sacrificing manufacturer longevity guarantees or performance?*

Proof-Of-Concept on Real Hardware

SSD A (MLC):



SSD B (MLC):



Take-away: Potential! But this (write all pages, erase, repeat) is an extremely simplified scenario.

Problem Statement

How Best to Leverage This Trick?

- Sounds Simple: Just transition a block down a bit-level when it approaches death

Open Problems Targetted:

- Upon bit-switch, how long will the new MLC (or SLC) block survive?
- Can we do a double-death?
- How much does ZombieNAND extend lifetime?
- Do current-gen algorithms (e.g., wear-leveling) work with this?
- What is the impact on performance (before and after rebirth)?
- *Don't break any manufacturer guarantees!*

Simulation Framework

Existing Simulators Fall Short

- Existing simulators use simple block counters
- Works when bit-levels remain constant, but when you switch. . .

Extending DiskSim

- Add a physics-accurate stress model
- Add support to existing mechanics (e.g., garbage collection) to handle varying bit-levels blocks

ZombieNAND Oxide Stress Model

1: **procedure** CALC_STRESS($cycle$)
2: $A \leftarrow 0.08$
3: $B \leftarrow 5.0$
4: $Cox \leftarrow 2.15e^{-17}$
5: $q \leftarrow 1.6e^{-19}$
6: $\delta N_{it} \leftarrow A * cycle^{0.62}$
7: $\delta N_{ot} \leftarrow B * cycle^{0.30}$
8: $\delta V_{it} \leftarrow (\delta N_{it} * q)/Cox$
9: $\delta V_{ot} \leftarrow (\delta N_{ot} * q)/Cox$
10: **return** $(\delta V_{it} + \delta V_{ot})$
11: **end procedure**

Conservative estimate: We ignore charge leakage (cell recovery) due to manufacturing variability

Limitations of Existing Wear-Leveling

Existing Wear-Leveling Algorithms Overdo It

- Early experiments with adapted DiskSim demonstrate limited improvements
- Problem: We actually don't want all of the cells to switch simultaneously

Solution: Controlled Wear-Unleveling for Lifetime

$$\text{Early Blocks} \leq \frac{(R - W) \times B}{2^{S-2}} \quad (1)$$

R=reserved percentage, W=high-watermark percentage,
B=number of blocks per element, S=starting bit-level

See the paper for the rest of the wear-leveling and GC algorithms

Experimental Setup: Timings

Fixed access latencies and lifetime by bit-level:

Access Type (unit)	SLC (2KB)	MLC (4KB)	TLC (8KB)
Read (page)	0.025 ms	0.05 ms	0.15 ms
Write (page)	0.2 ms	0.5 ms	1.0 ms
Erase (block)	1.5 ms	1.5 ms	3.0 ms
Lifetime (cycle)	75,000	6,000	1,000

Derived from specification documents from Micron. Fixed access latencies are not reasonable for small studies, but for lifetime studies they work fine.

Experimental Setup: SSD Configuration

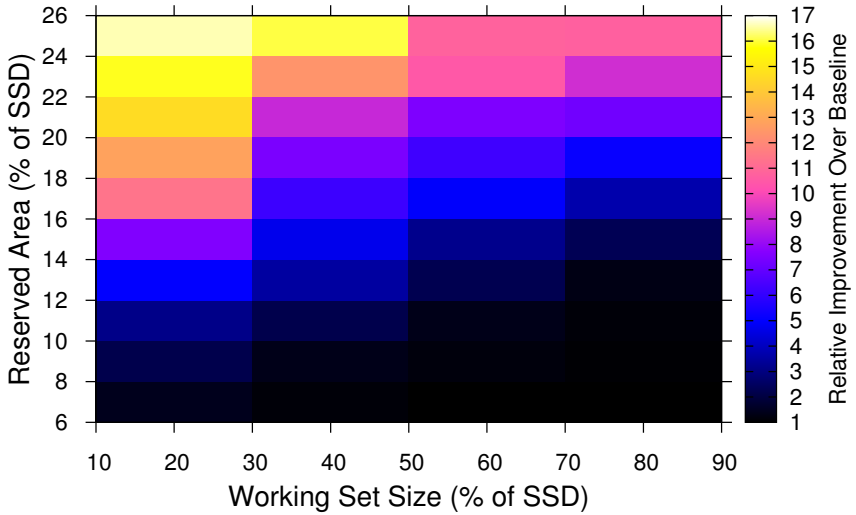
Key experimental SSD configurations.

	Synthetic	Trace-Driven
Flash Chips	1	4
Blocks per Element	128	512
Planes per Element	8	8
Blocks per Plane	16	64
Pages per Block	128	128

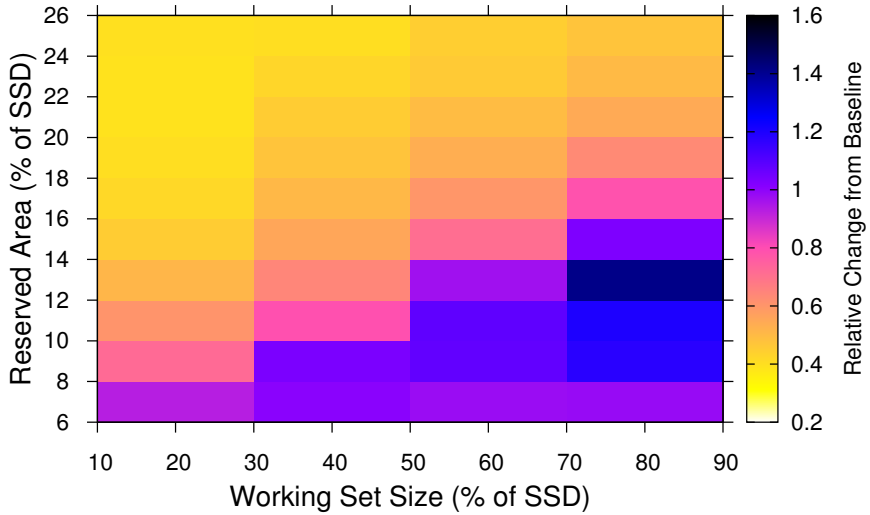
Yes, these are "small" configurations (128MB and 1GB SSD sizes) relative to modern drives (often 128GB to 1TB) due to raw duration of simulation.

Synthetic TLC Results: 50% Read/Write Ratio

Normalized Lifetime

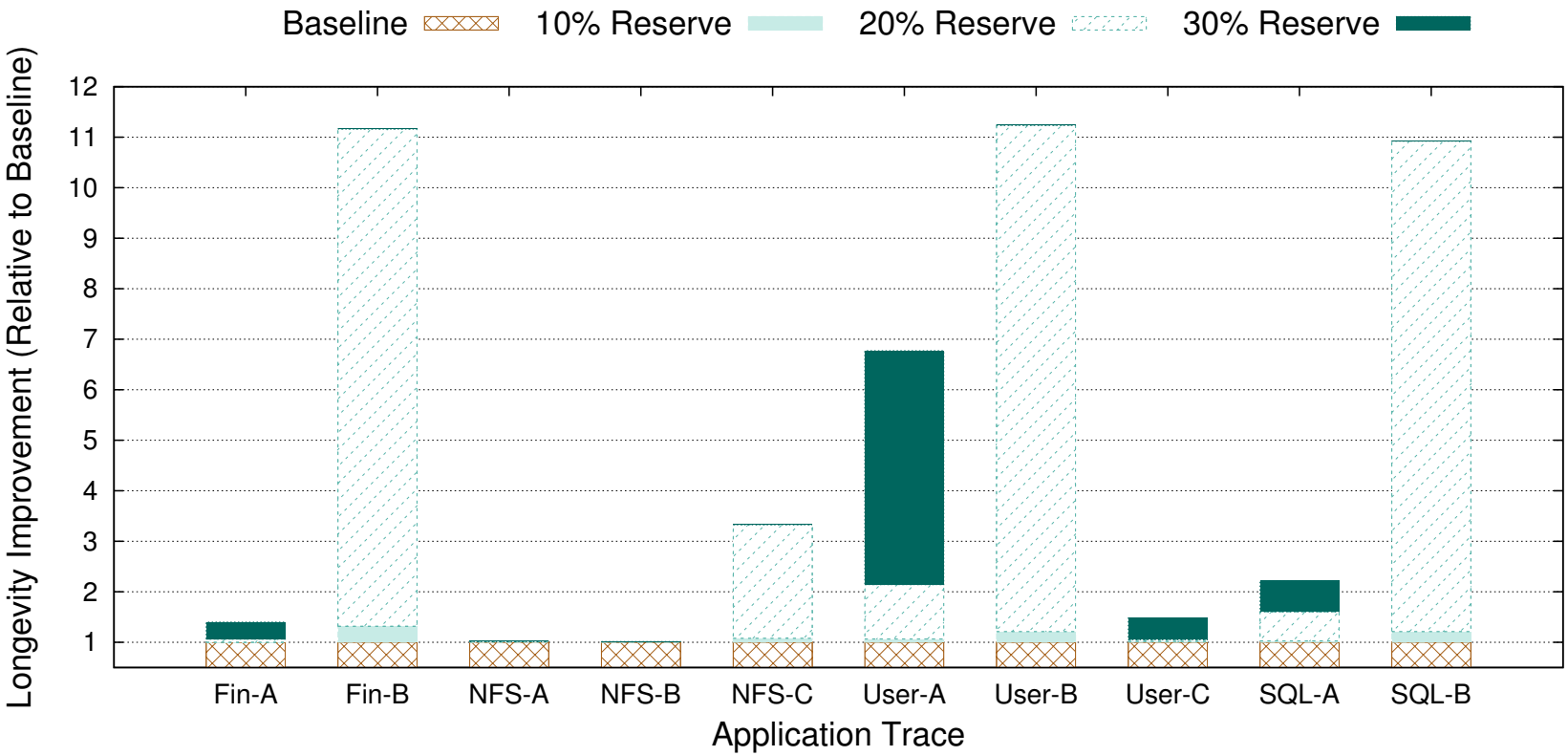


Normalized Latency



Take-aways: 1) All lifetimes are at least as long as the baseline. 2) Latency degradations occur largely after death of baseline. 3) Large lifetime gains (up to 16x) are not unreasonable due to huge differences in TLC/MLC/SLC P/E cycles; latency gains are accordingly less drastic. 4) Other R/W ratios follow same trend (see paper).

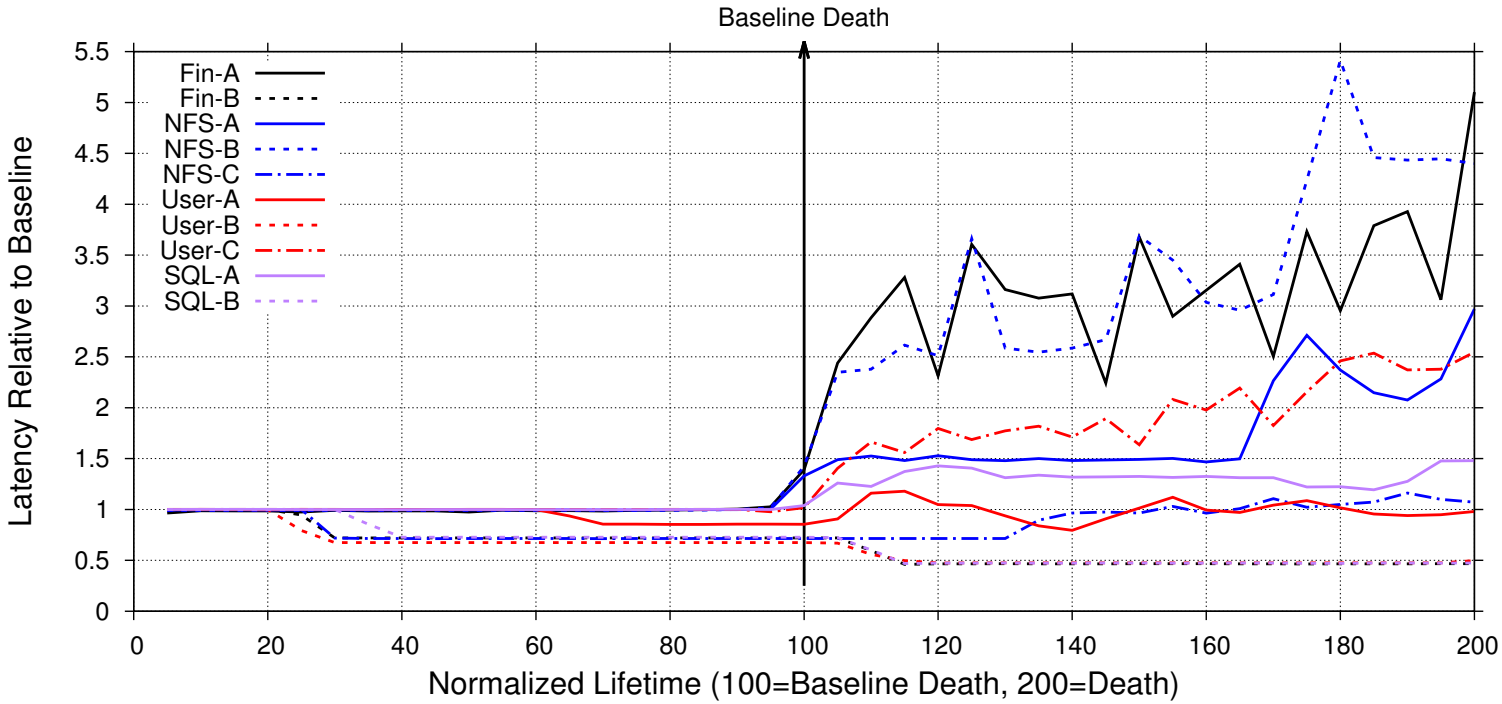
Trace-Driven TLC Simulation Lifetime Results



Take-aways: 1) Still assuring in the worst case we match baseline. 2) Lifetime improvements vary widely across applications. 3) *Efficacy has a strong correlation to address reuse of writes (see paper for details).*

Trace-Driven TLC Simulation Latency Changes Over Lifetime

For TLC, 20% Reserved Scenario (see paper for rest)



Take-aways: 1) Match or exceed performance up until last 5% of baseline life. 2) Some traces get even better after baseline death, some get far worse – desirable compared to complete death. 3) Spill-over accesses drive performance loss in post-baseline area.

Questions?