Rejuvenator : A Static Wear Leveling Algorithm for NAND Flash Memory with Minimized Overhead

Muthukumar Murugan and David H.C. Du

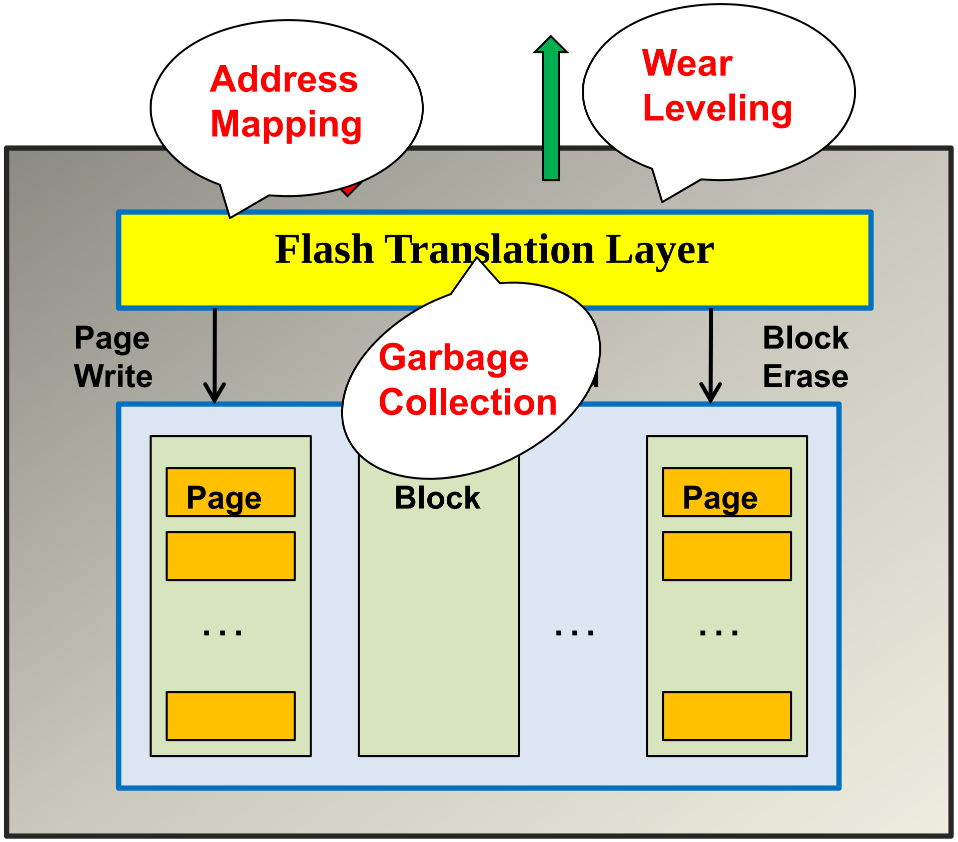
University of Minnesota

Agenda

- NAND flash memory – Background
- Wear leveling – Background
- Motivation
- Rejuvenator – Design
- Adaptability in Rejuvenator
- Evaluation
- Conclusion

Background: NAND Flash Memory

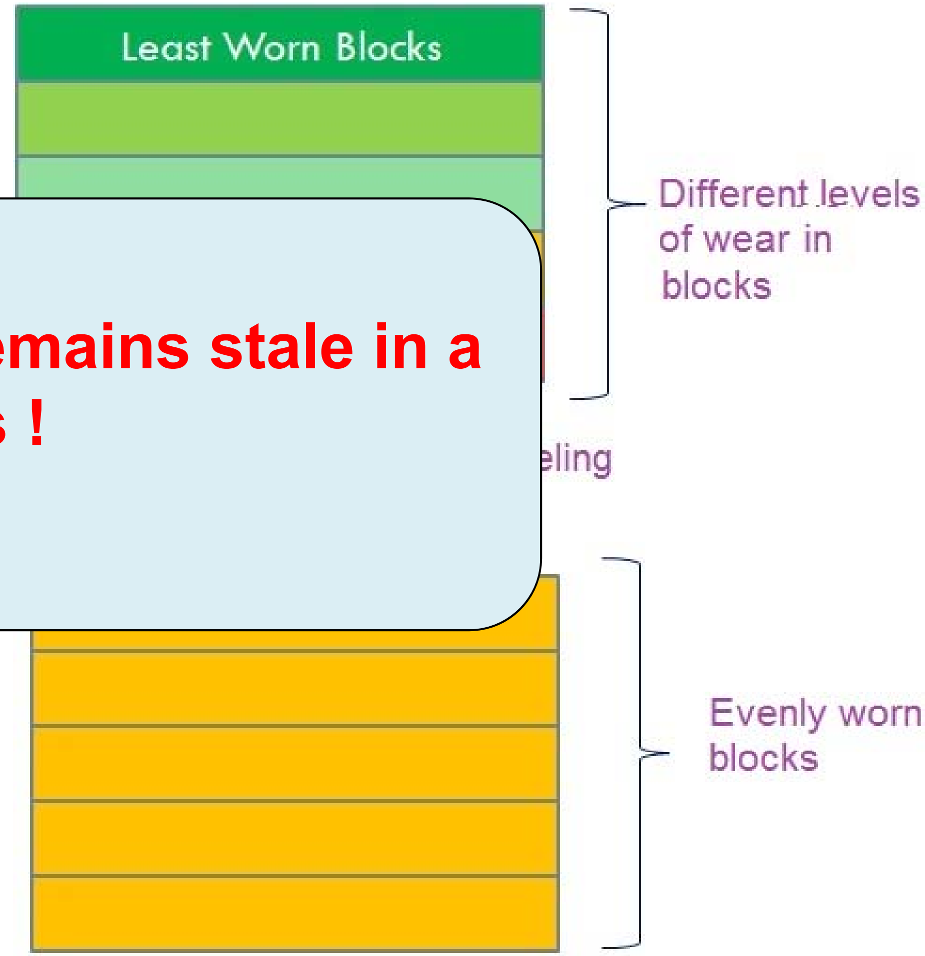
- An array of flash blocks
- Read/write in **page** units
- Typical **block** = **128K**;
  page = **2K** or **4K**
- Must erase **block** before write
- **Read** = **25** microseconds
- **Write** = **200** microseconds
- **Erase** = **1500** microseconds
- Limited number of erases per block
  - **100K** for **SLC**
  - **10K** for **MLC**



Background: Wear Leveling

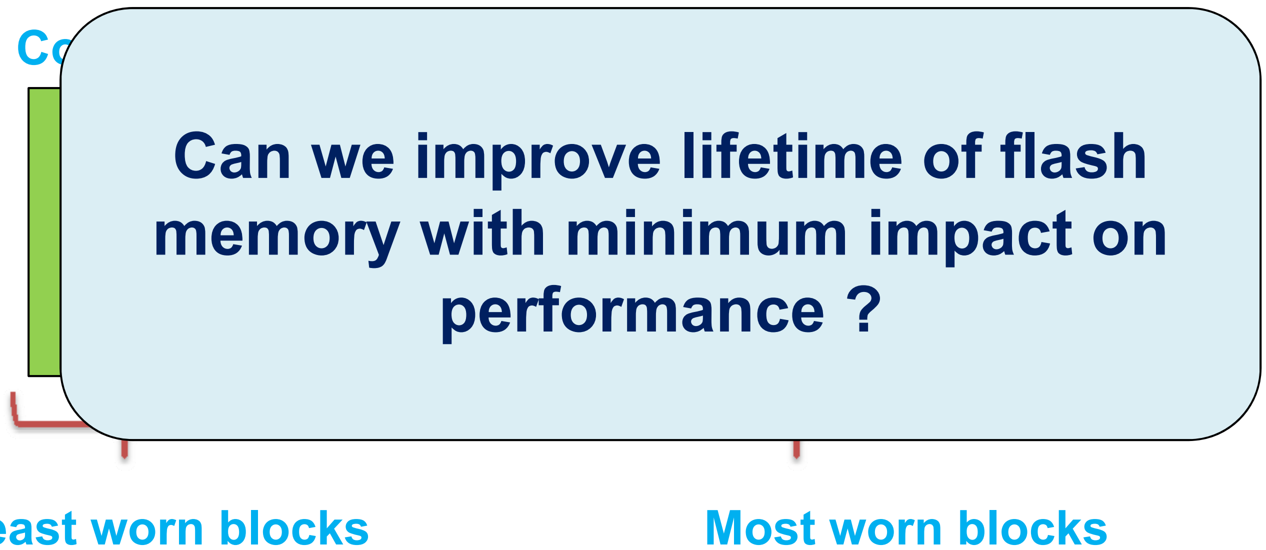
- Frequently written blocks wear out faster
- Need t
  block
- Dyna
  - Writ
  that
  - Writ
  that have higher erase counts

Issue : Cold data remains stale in a few blocks !



Background: Static Wear Leveling

- Static wear leveling
  - Moves stale cold data around periodically



- Rejuvenator
  - Static wear leveling algorithm
  - Comprehensive design WL, GC and FTL components

UNIVERSITY OF MINNESOTA

Motivation

- General wear leveling goals :
  - Improve **lifetime** of flash memory
  - Reduce **variance** in erase counts of blocks
- Rejuvenator goals :
  - Prevent a **single block** from reaching its lifetime faster than other blocks
  - Reduce **write amplification** due to static cold data migration
  - Do static cold data migration judiciously !
  - **Adapt** to changing workloads and rate of increase in erase counts

Existing wear leveling algorithms

- TrueFFS:
  - Virtu ... in of physical eras
  - ***Folding*** changes mapping to one physical erase unit
- Static wear leveling done periodically
- Valid data in the chain copied to one physical erase unit

Observation : High variance in erase counts

Existing wear leveling algorithms

- Dual Pool :
  - Two … d
  - Hotr … count

Observation : More than necessary data migrations due to constant threshold

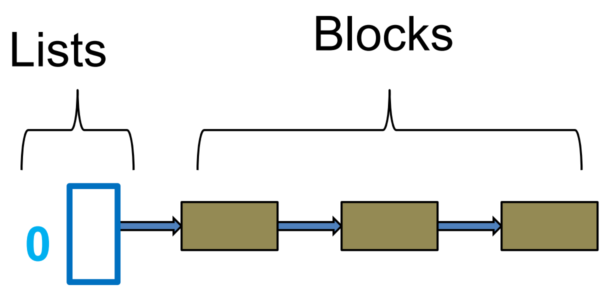
- Coarse grained control of blocks
- Threshold based static wear leveling
  - Swap data between oldest and youngest blocks
- No explicit hot data identification

Can we control variance in erase counts with just enough cold data migrations ?

Rejuvenator – Design

Rejuvenator : Overview

- Maintain lists of blocks based on erase count
- Initially all blocks associated with list *0*

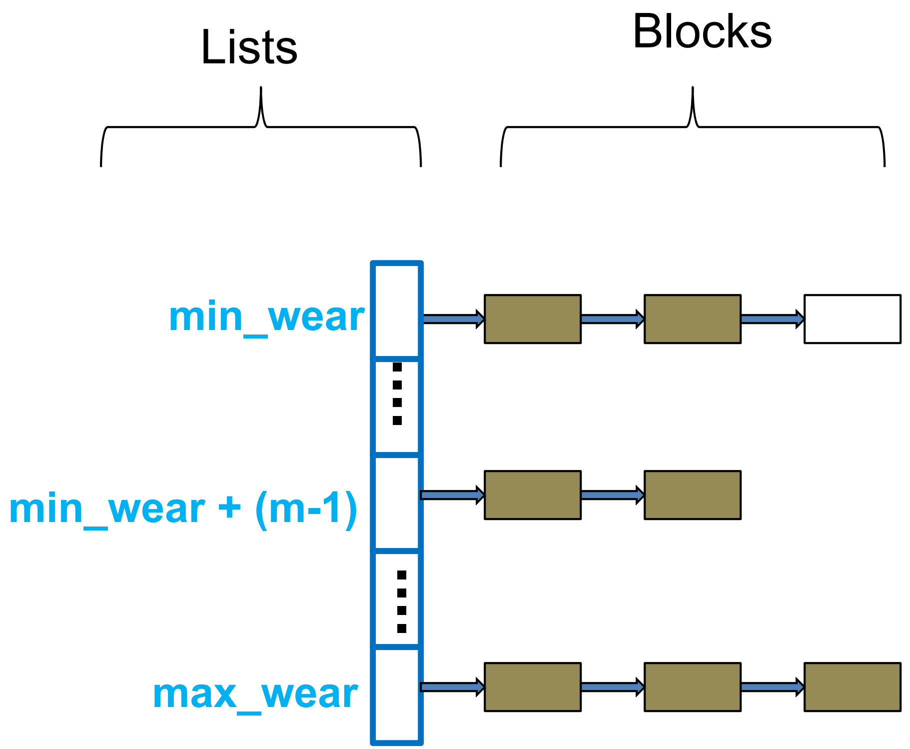


Rejuvenator : Overview

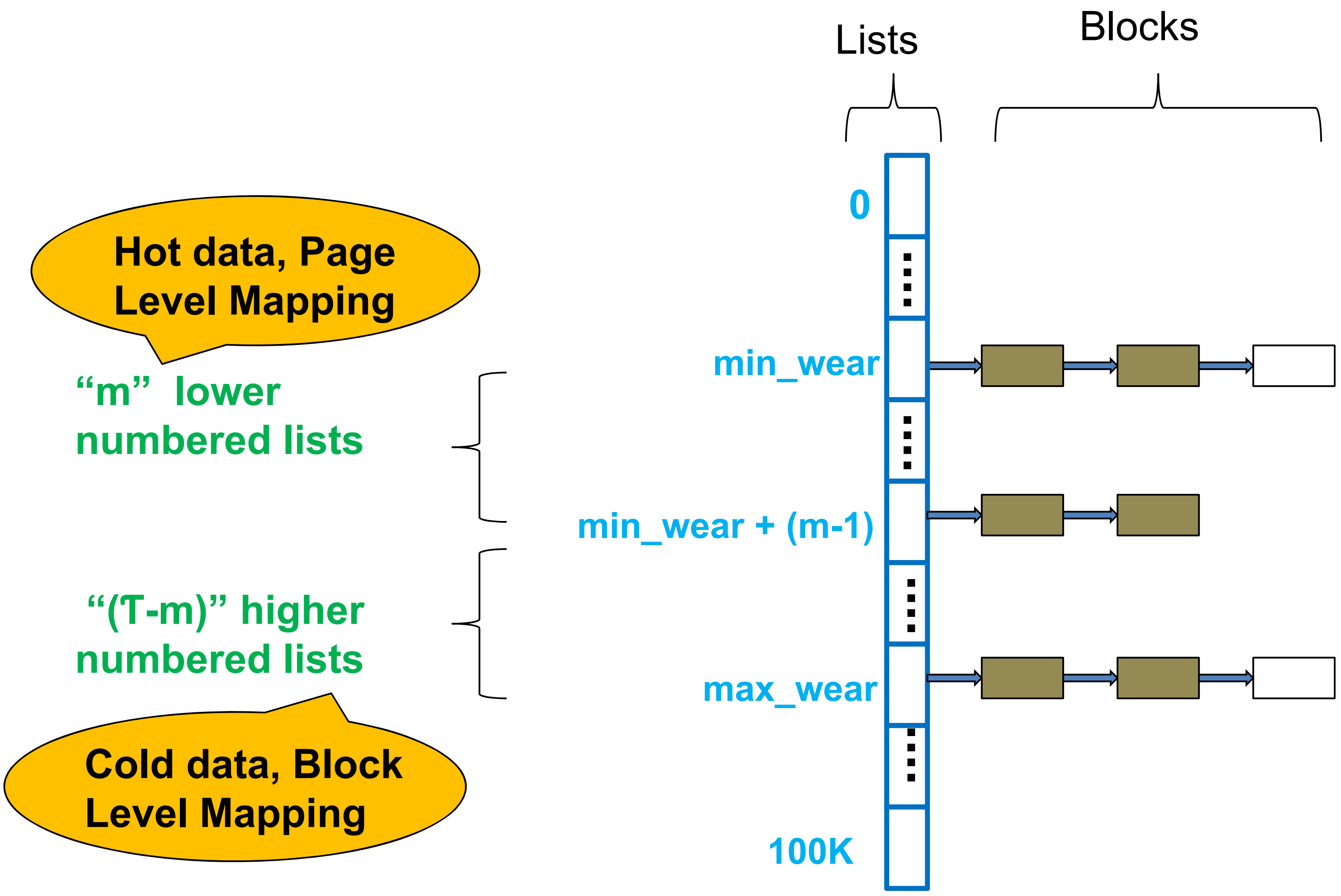
- Maintain lists of blocks based on erase count
- Initially all blocks associated with list *0*
- As blocks are erased they get associated to higher lists
- Difference between minimum and maximum erase count is

diff = max_wear – min_wear

diff $\leq$ *T-1*



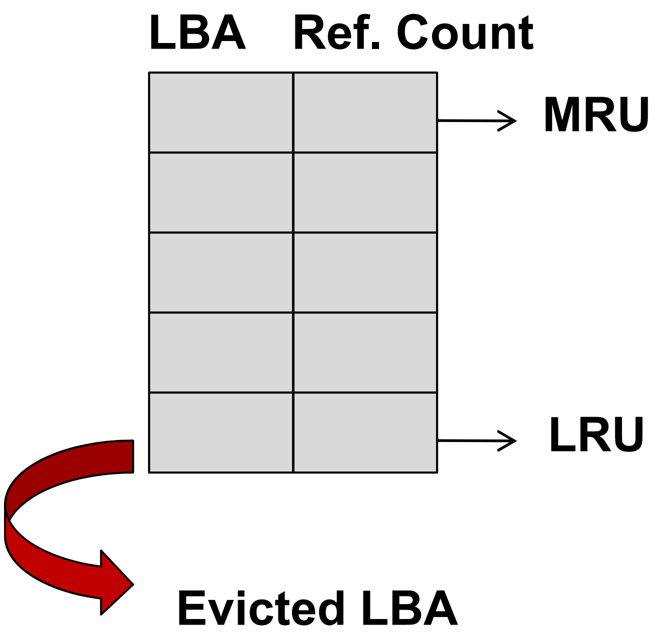
Rejuvenator : Mapping



University of Minnesota

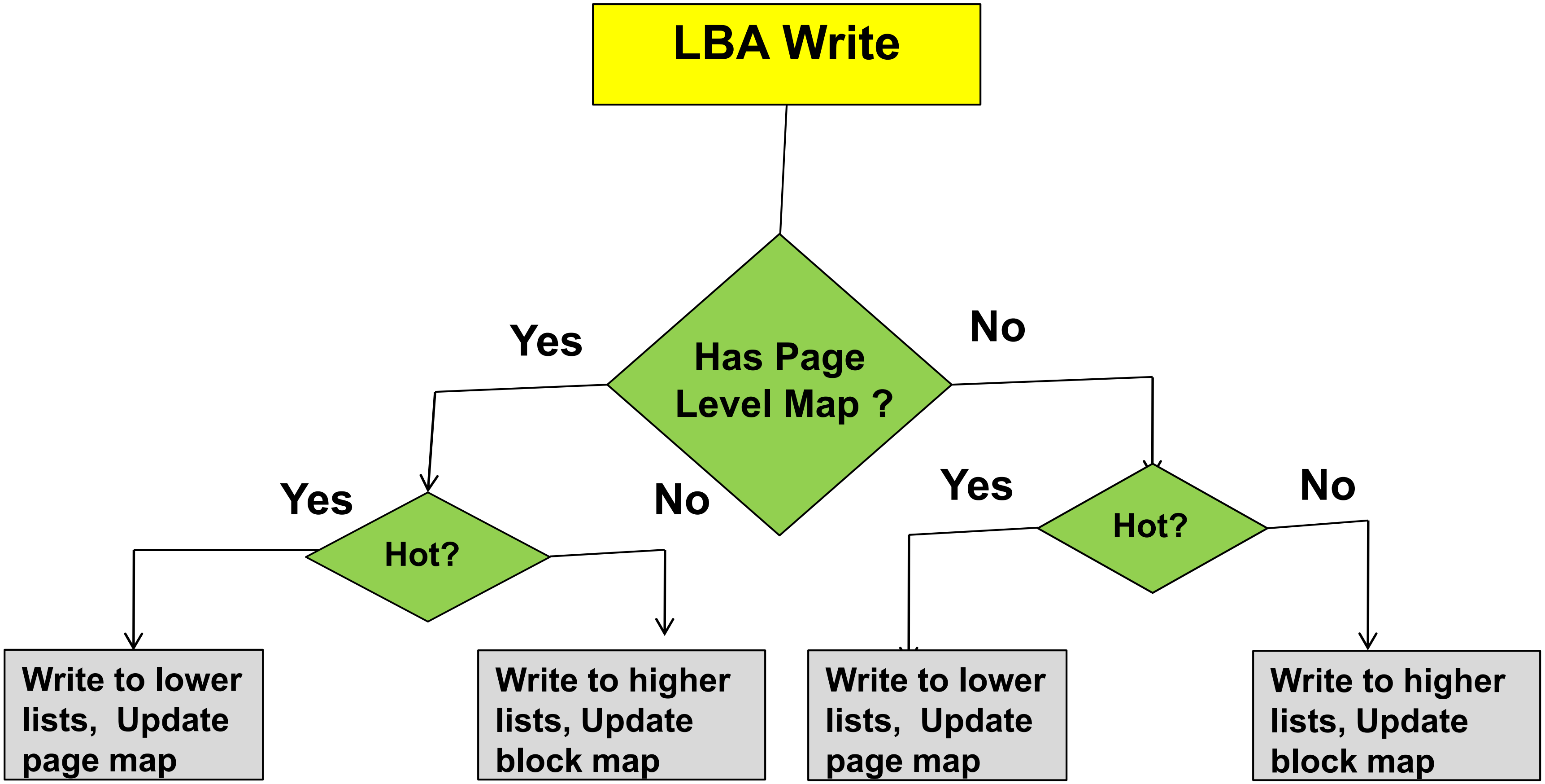
Rejuvenator : Hot data Identification

- Account for recency and frequency
- LRU list with reference counts
- Window size : 1024
- Hot : Most frequently written LBAs
- Any LBA having ref. count
  - > Average reference count is hot



UNIVERSITY OF MINNESOTA

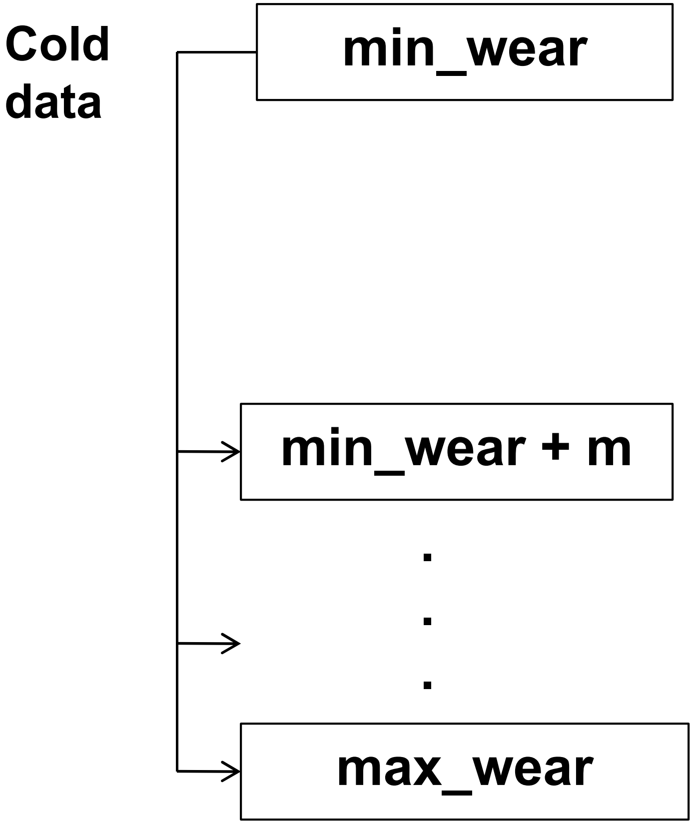
Rejuvenator : Handling Writes



Rejuvenator : Data Migrations

- Sliding window size ≤ T
- Window movement restricted at
  - Top : Cold data accumulation in lower lists

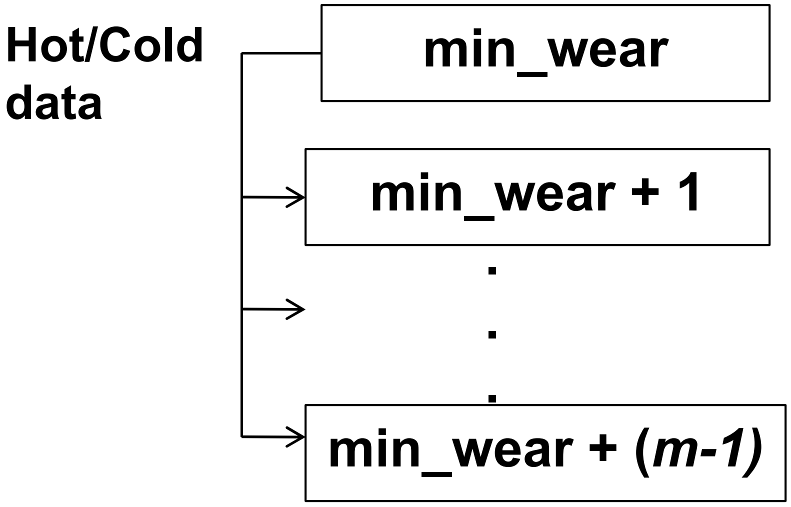
diff = T - 1



Rejuvenator : Data Migrations

diff = T - 1

- Sliding window size ≤ T
- Window movement restricted at
  - Bottom: Invalid blocks accumulate in list *min_wear +(T-1)*



Very Rare !

Rejuvenator : Garbage Collection

- Garbage collection
  - Copy valid pages of blocks elsewhere
  - Erase current block

No. of clean and invalid pages

Cleaning Efficiency =

of blo

Enable efficient GC via intelligent wear leveling

- Garbage collection starts in lower numbered lists
- Intuitively :
  - Lower numbered lists have lesser erase counts
  - Contain more invalid pages and hence better cleaning efficiency

Adaptability

Impact of the value of τ

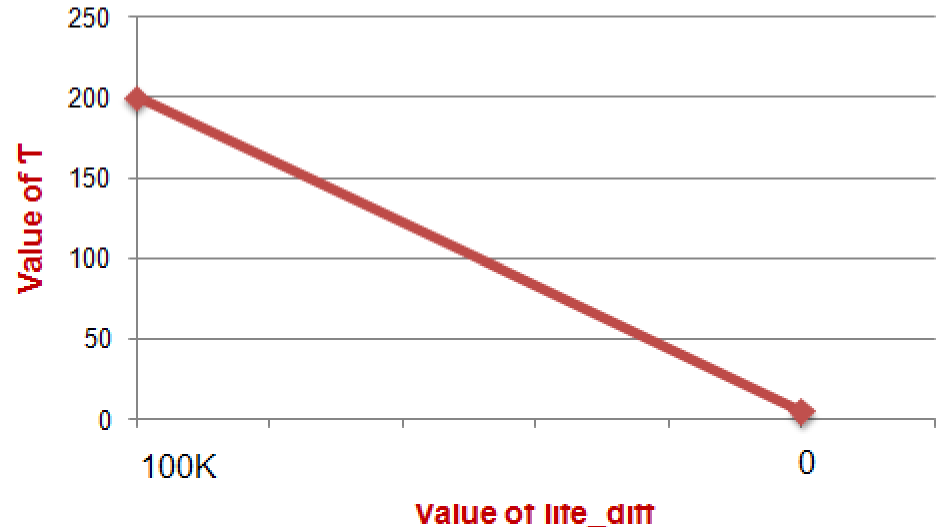
- Larger value of τ
  - Large variance in erase counts
- Smaller value of τ
  - Static cold data migration is done more often
- **Goal :** Strike a balance between the two
- Adapt the value of τ depending on lifetime of flash memory
- Tighten the constraints on variance of erase counts gradually

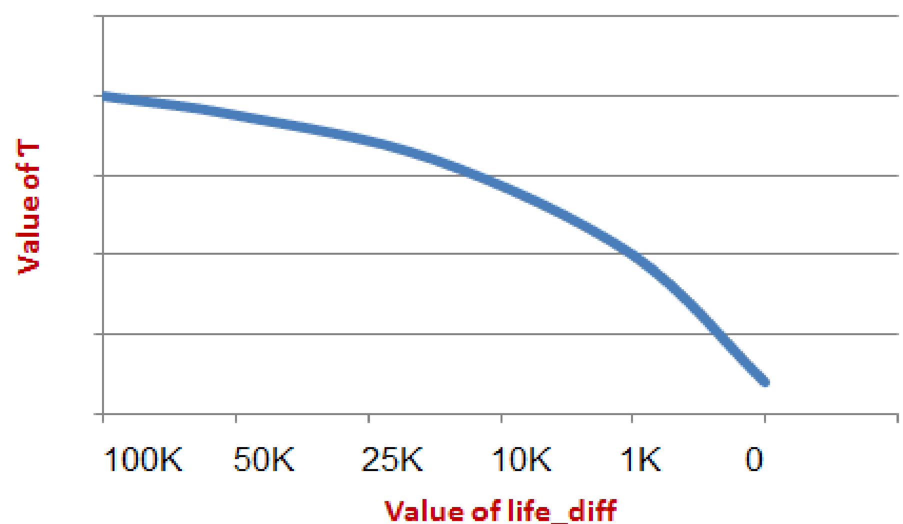
Adapting the value of τ

- The value of τ is very large in the beginning
- As the blocks get older the value of τ is reduced gradually
- Decrease in τ α life_diff

life_diff = 100K – max_wear

- Decreasing τ
  - Linear
  - Non-linear





Adapting the value of m

- Value of m controls proportion of blocks storing hot data

- Adapting to workload pattern changes
  - Increase m when hot data flow is more
  - Decrease m when cold data flow is more

Evaluation

Overheads in Rejuvenator

- Memory for the lists :
  - 4 bytes per block address
  - 1 TB flash (2KB page, 128 KB block) requires ~ 32MB of memory
- Memory for mapping tables :
  - < 10 % hot data (page level mapping)
  - at most 250 MB of memory for 1 TB of flash
- O (1) time for list association of blocks
- No block copy for hot writes

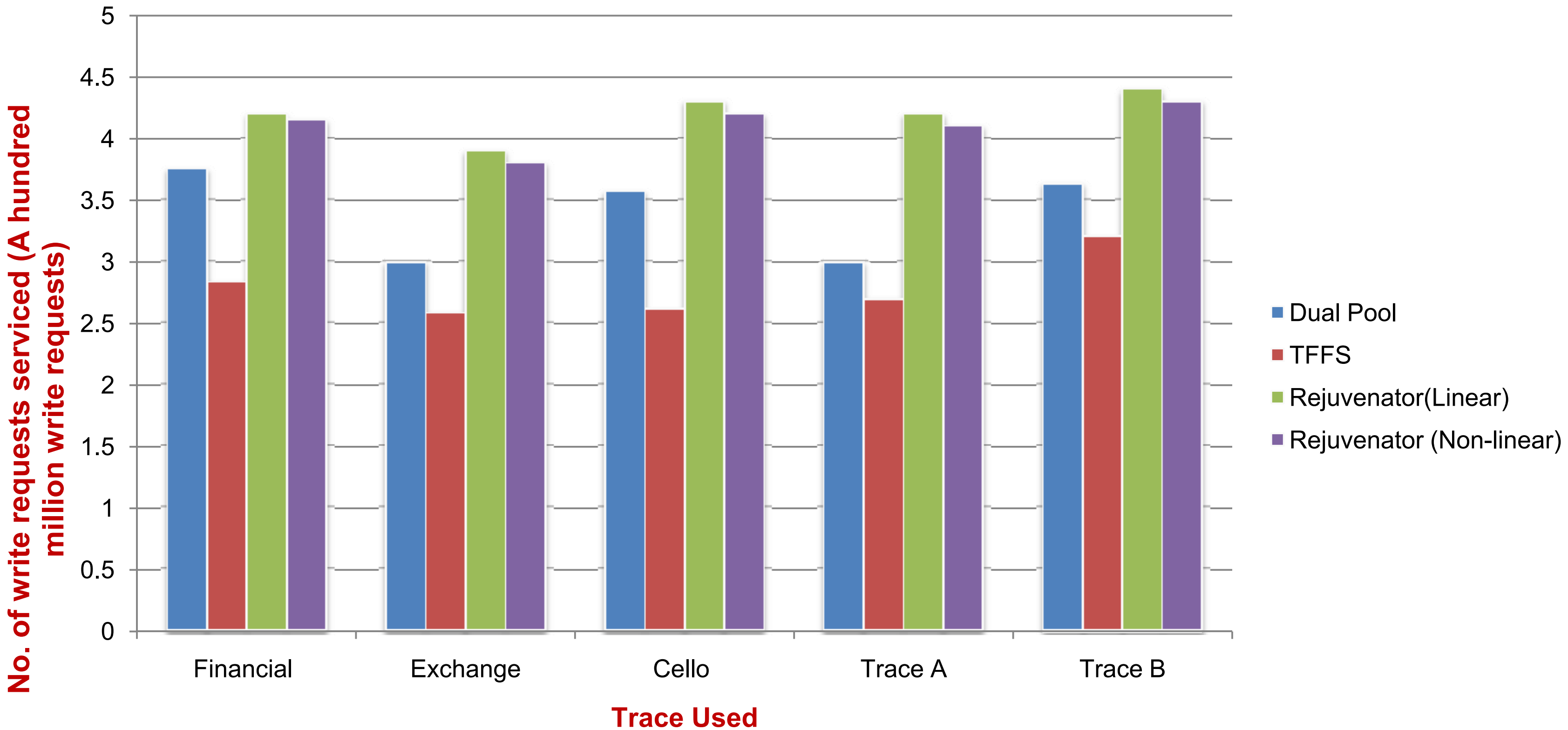
Simulation Environment

- Simulator written in C++
- Takes LBAs from trace as input
- Consider small portion of SSD
- Maximum erase count of blocks : *2K*
- Traces used : Financial, Exchange, Cello
- Synthetic traces:

    - A : Random writes

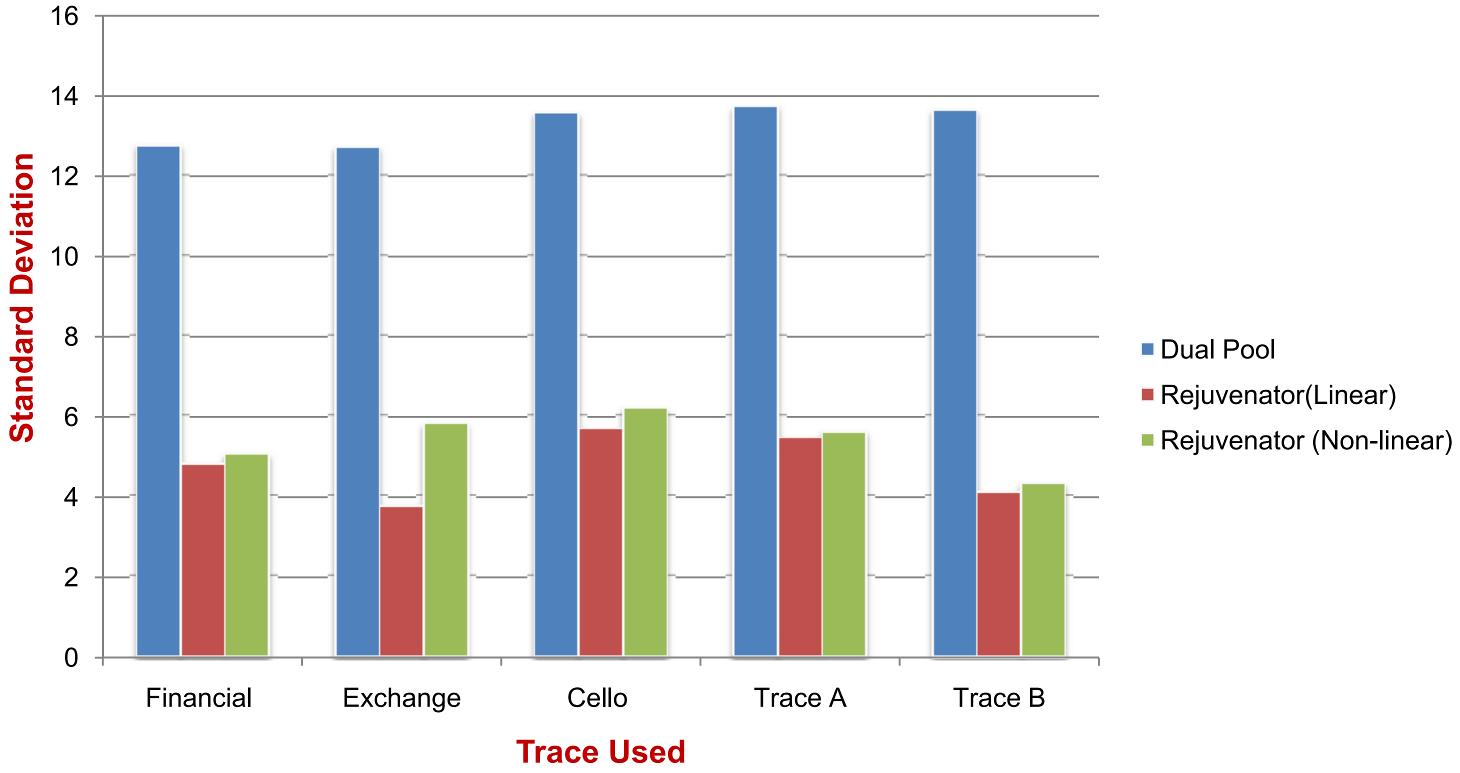
    - B : 50% sequential

Lifetime



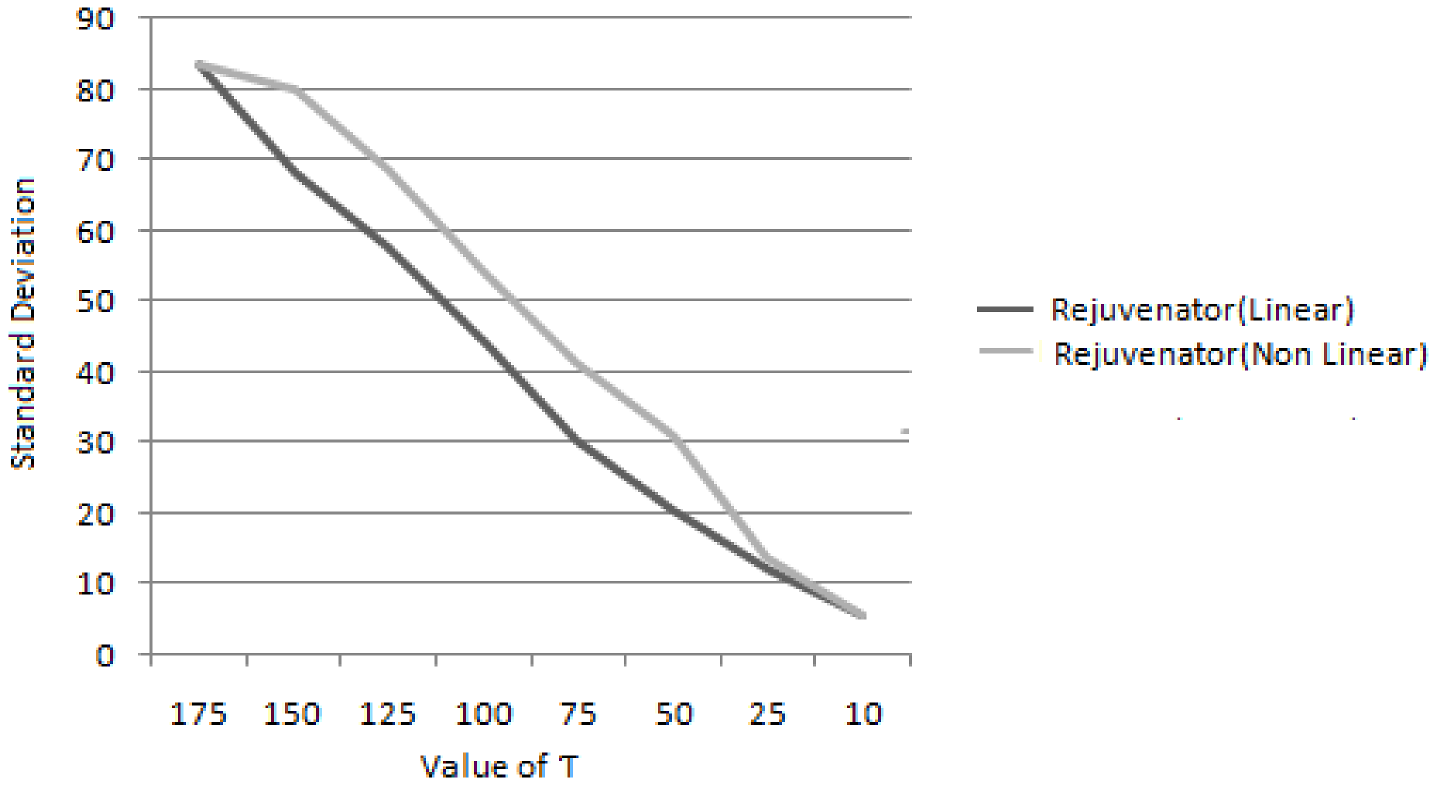
25% Improvement on the average

Standard Deviation in Erase counts



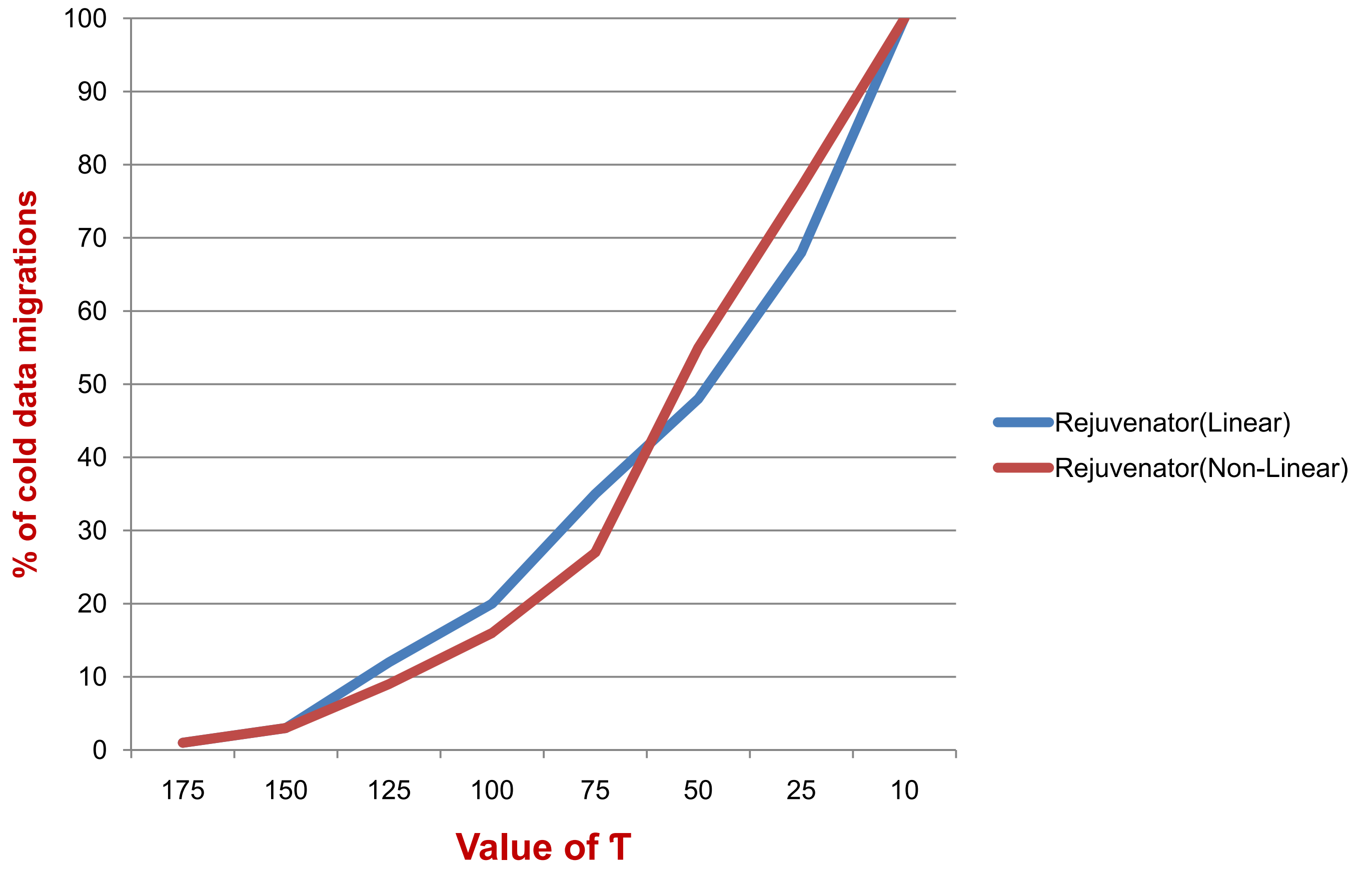
Better control of variance in erase counts

Standard Deviation in Erase counts - Trend



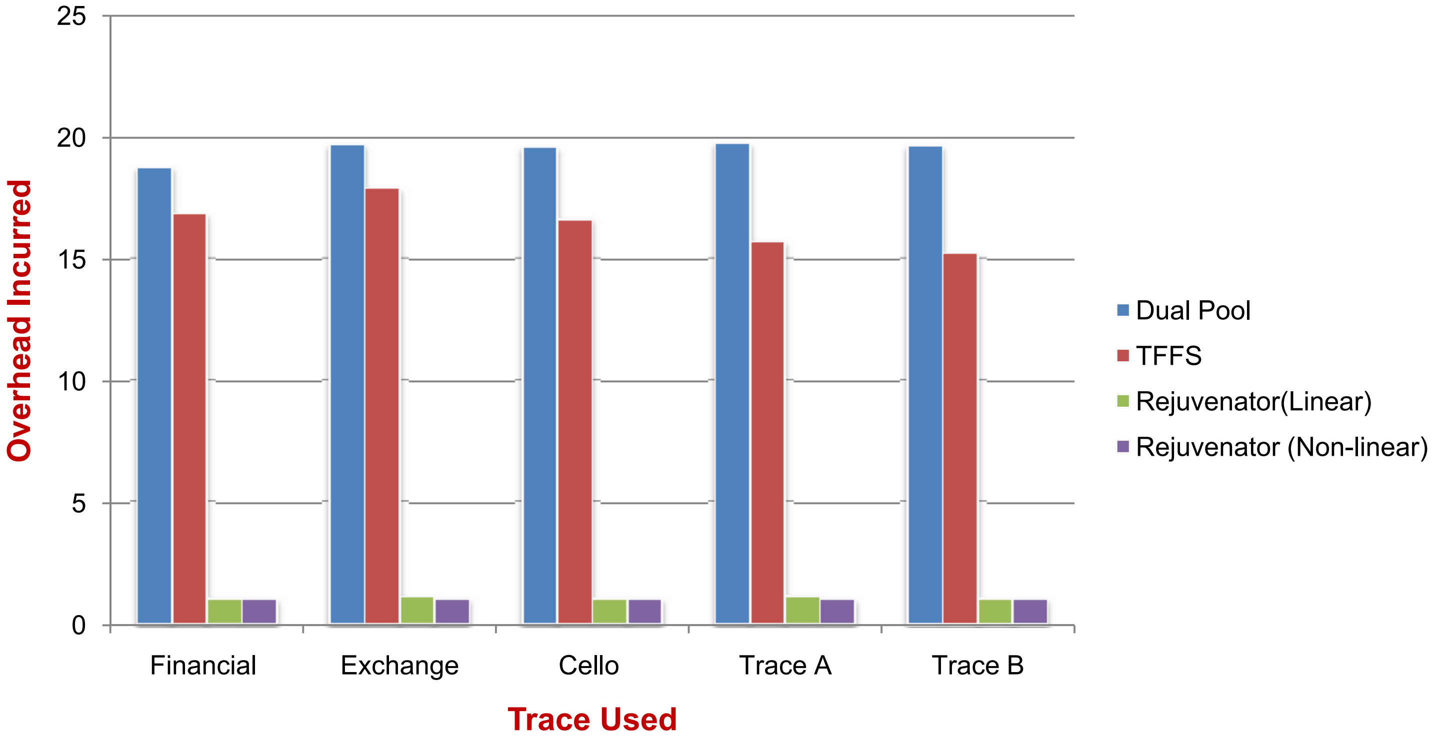
Tighter constraints only in the end

Static Cold Data Migrations- Trend



Tighter constraints only in the end

Wear Leveling Block Erase Overhead



Reduced by 15-18 times

University of Minnesota

Conclusion

- Rejuvenator
  - manages variance in erase counts with just enough static cold data migrations
  - improves lifetime of flash memory
  - manages data according to degree of hotness
  - deals with performance – lifetime tradeoff
- Rejuvenator adapts to changes in workload patterns
- A case for integrated wear leveling and GC operations