Generating Realistic Wear Distributions for SSDs

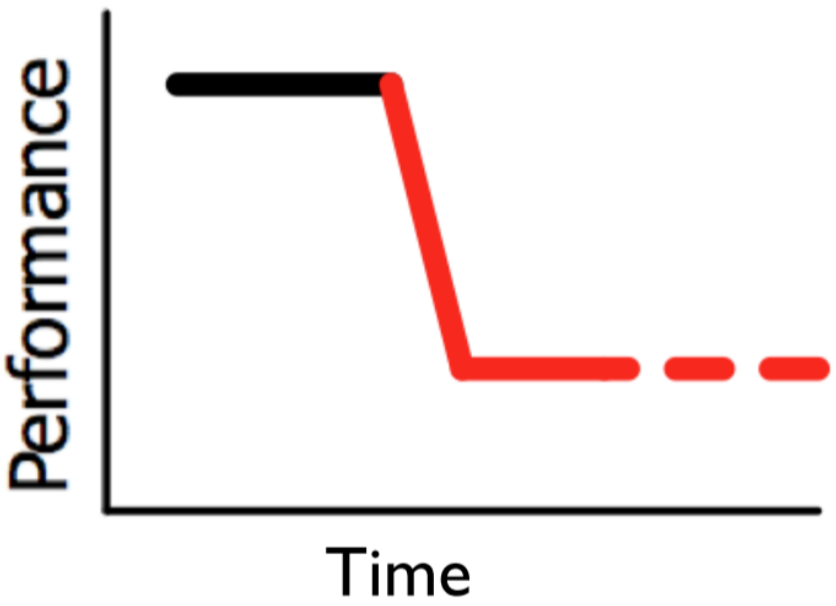
Ziyang Jiao, Bryan S. Kim
Syracuse University

Overview

- The fail-slow symptom
- Challenges in SSD aging
- Related works
- Fast-forwardable SSD
- Evaluation
- Conclusion and future work

The fail-slow symptom of SSDs

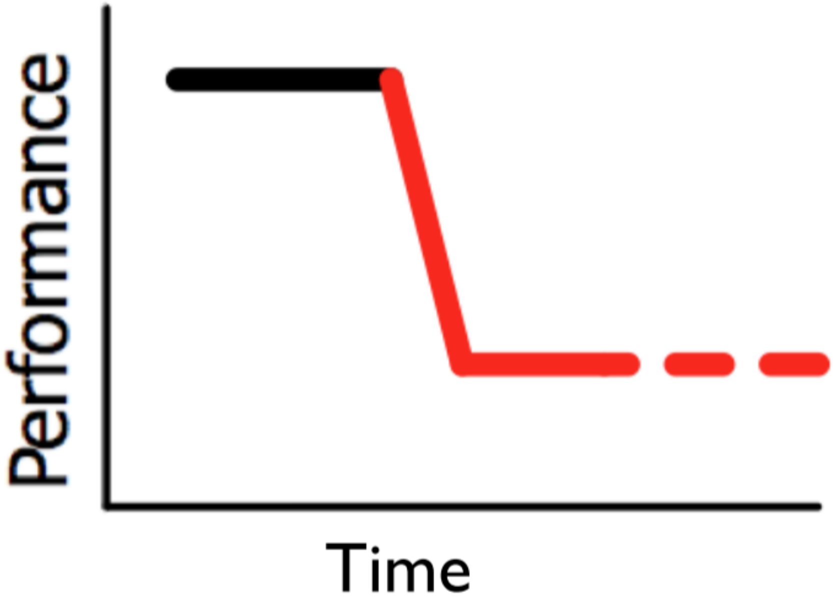
- Performance degradation



- Haryadi S. Gunawi et al, "Fail-Slow at Scale: Evidence of Hardware Performance Faults in Large Production Systems", FAST 2018

The fail-slow symptom of SSDs

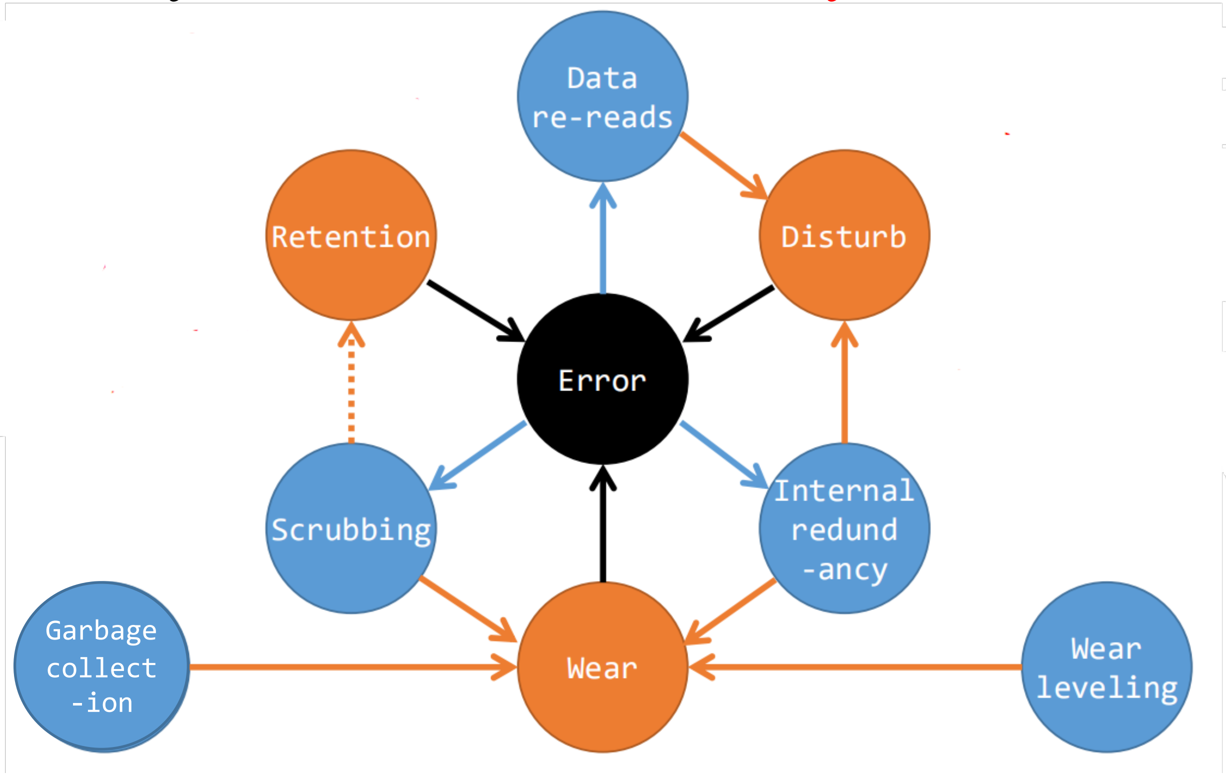
- Performance degradation
- No existing SSD development frameworks consider aging in their design



- Haryadi S. Gunawi et al, “Fail-Slow at Scale: Evidence of Hardware Performance Faults in Large Production Systems”, FAST 2018

Challenges

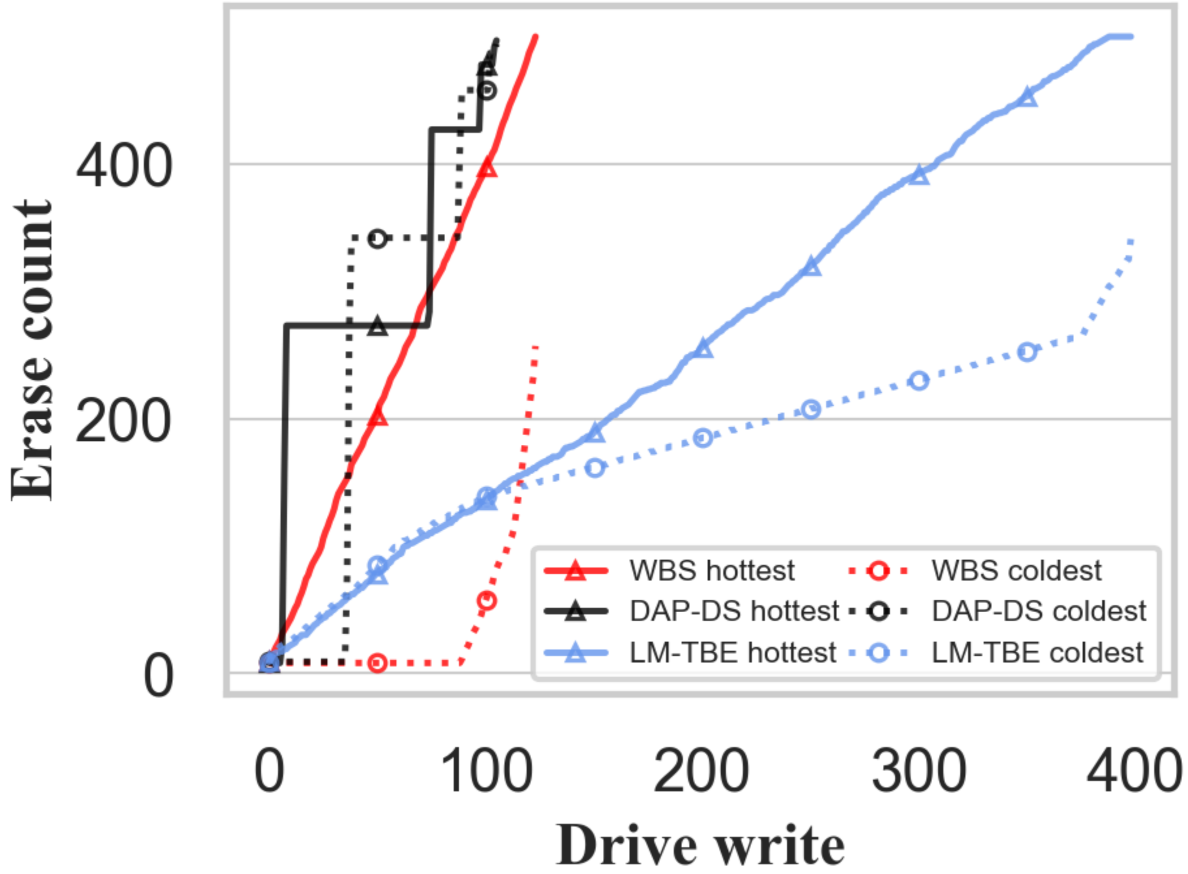
- The overhead of aging process → Efficiency
- The internal intricacy of SSDs → Accuracy



- Bryan S. Kim et al, “CPR for SSDs”, HotOS 2019

Challenges

- The irregularity of block erasure



Current art

Preconditioning:

The process of writing data to the device to prepare it for steady state measurement.

Expensive:

- ✖ **Resources**
- ✖ **Time**

- https://www.snia.org/sites/default/files/technical-work/pts/release/SNIA-SSS-PTS-Enterprise-v1.1.pdf

File system aging

- FS aging is not applicable to SSD aging
  - FS aging: generate a fragmented state of logical block layouts
  - SSD aging: physical aging of blocks

- Preconditioning is more akin to FS aging
  - Populating and invalidating the address space
  - Cannot sufficiently age the device to an end-of-life state.

ML for simulation

- DEVS

  DEVS execution acceleration with machine learning.

  SpringSim 2016: https://dl.acm.org/doi/10.5555/2975389.2975399

  – Consider multiple model candidates

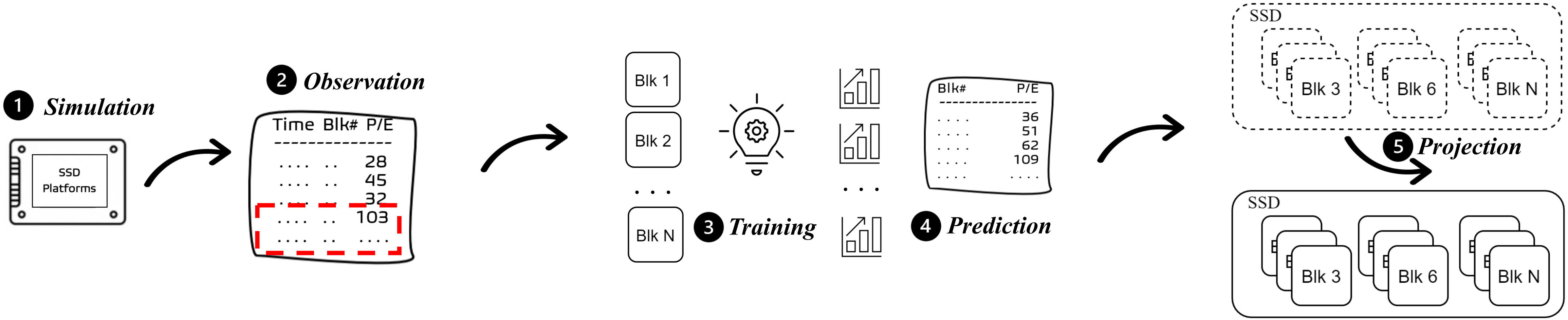
- CML

  Using continuous statistical machine learning to enable high-speed performance prediction in hybrid instruction-/cycle-accurate instruction set simulators.

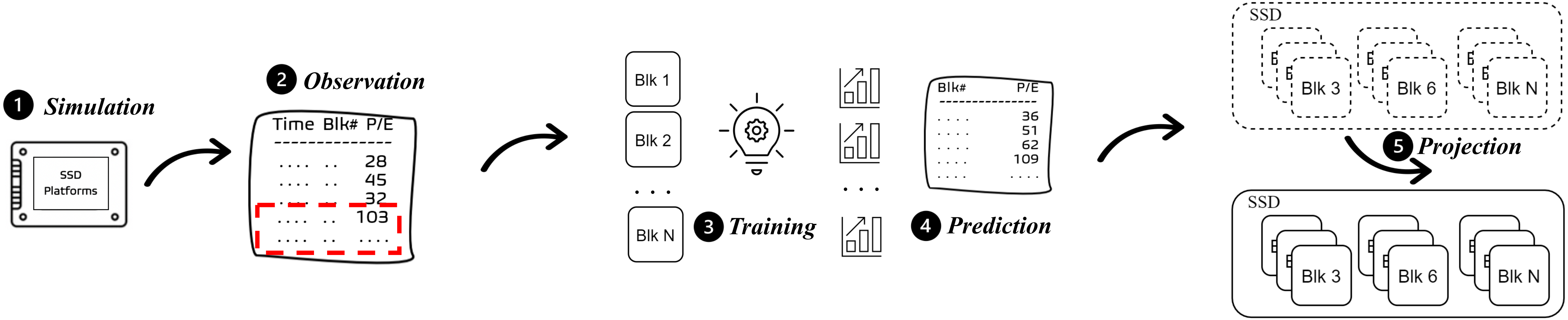
  CODES+ISSS 2009: https://dl.acm.org/doi/10.1145/1629435.1629478

  – Continuously incorporate the latest data to update model

Fast-forwardable SSD



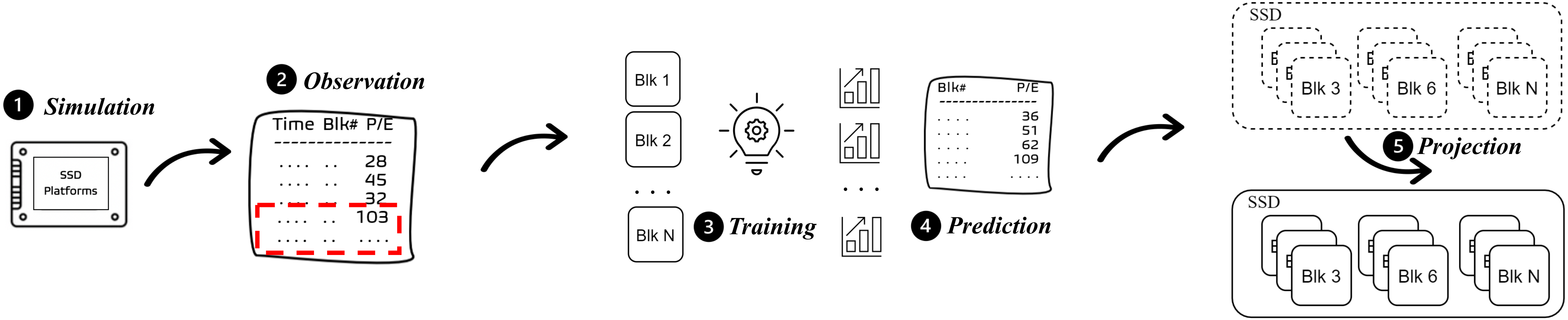
System overview



Simulation & Observation

- Observe SSD’s internal activities
- Output log periodically

System overview

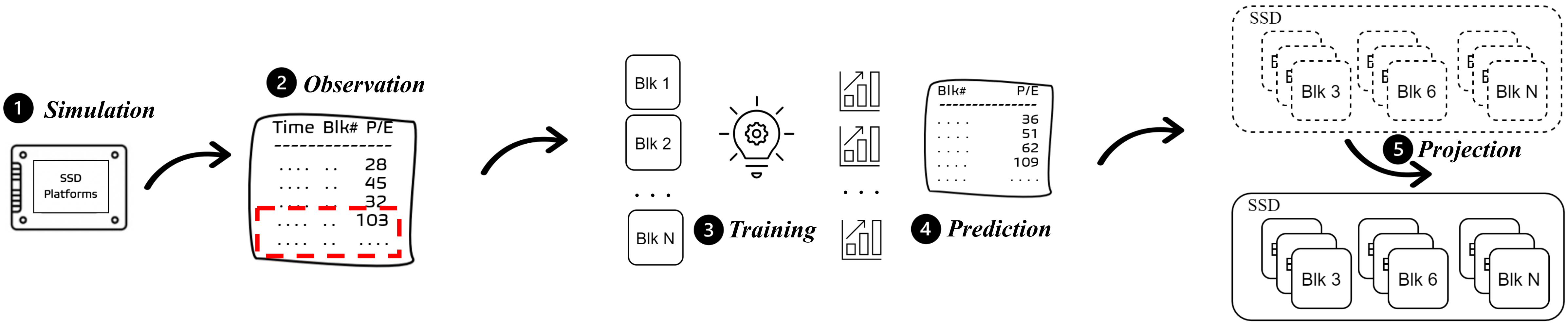


Simulation & Observation → Training & Prediction

- Observe SSD's internal activities
- Output log periodically

- Multiple lightweight regression models (LR)
- Predict future state based on the latest history
- $Acceleration\ Factor\ (AF)$
$$= \frac{simulated\ +\ predicted}{simulated}$$

System overview



Simulation & Observation → **Training & Prediction** → **Projection**

Simulation & Observation

- Observe SSD's internal activities
- Output log periodically

Training & Prediction

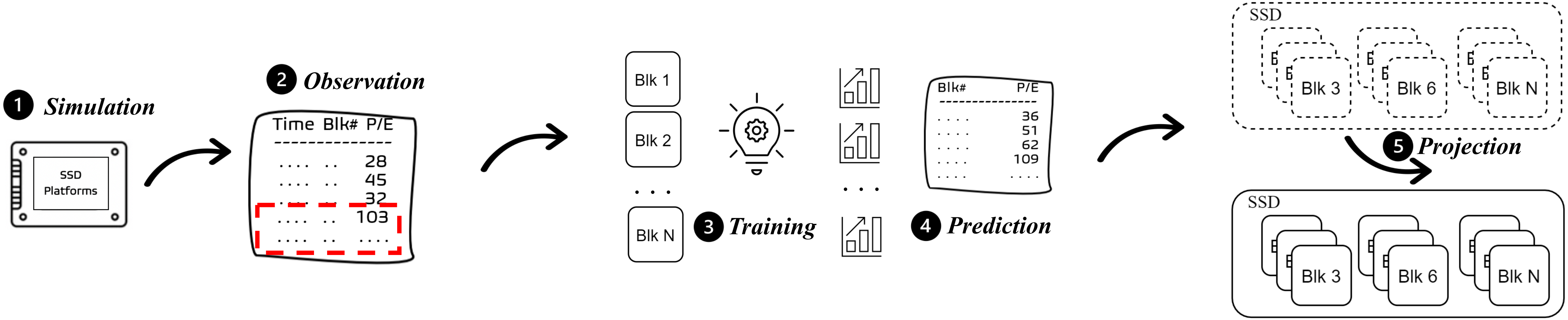
- Multiple lightweight regression models (LR)
- Predict future state based on the latest history
- $Acceleration\ Factor\ (AF)$

$$= \frac{simulated\ +\ predicted}{simulated}$$

Projection

- Project SSD states based on the prediction results

System overview



Simulation & Observation

- Observe SSD's internal activities
- Output log periodically

Training & Prediction

- Multiple lightweight regression models (LR)
- Predict future state based on the latest history
- $Acceleration\ Factor\ (AF)$

$$= \frac{simulated\ +\ predicted}{simulated}$$

Projection

- Project SSD states based on the prediction results

Enhancing efficiency

- Build models for each block:
  - The summed prediction overhead is proportional to the drive capacity (the # of blocks), although the model is lightweight itself.

- Two approaches to further enhancing efficiency:
  - A naïve approach: based on sampling
  - An analytic approach: based on distribution modeling

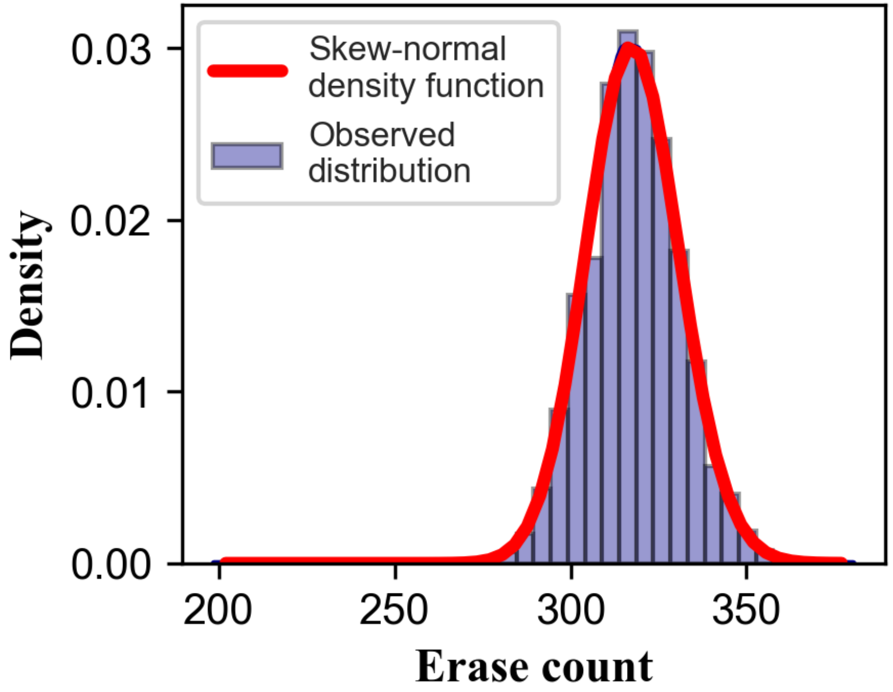
Approximation by distribution modeling

- Challenge: given only information of a subset of blocks, how can we estimate the blocks that behave distinctively than samples?

- Use extrapolation as the estimate method:
  - Assume that the wear distribution of blocks adheres to an underlying measurable distribution $\rho(\cdot)$
  - Estimate the future wear using the prediction result and the density function that models the underlying distribution.

Approximation by distribution modeling

- Approximation by distribution modeling:
  - $\rho(\cdot)$: a skew-normal distribution with skewness α, location μ, and scale parameter σ.
  - $\alpha=0.75$, $\mu=310$, $\sigma=15.1$
  - Fail to reject the null hypothesis on 10^5 samples with $p > 0.1$ using Kolmogorov–Smirnov goodness-of-fit test



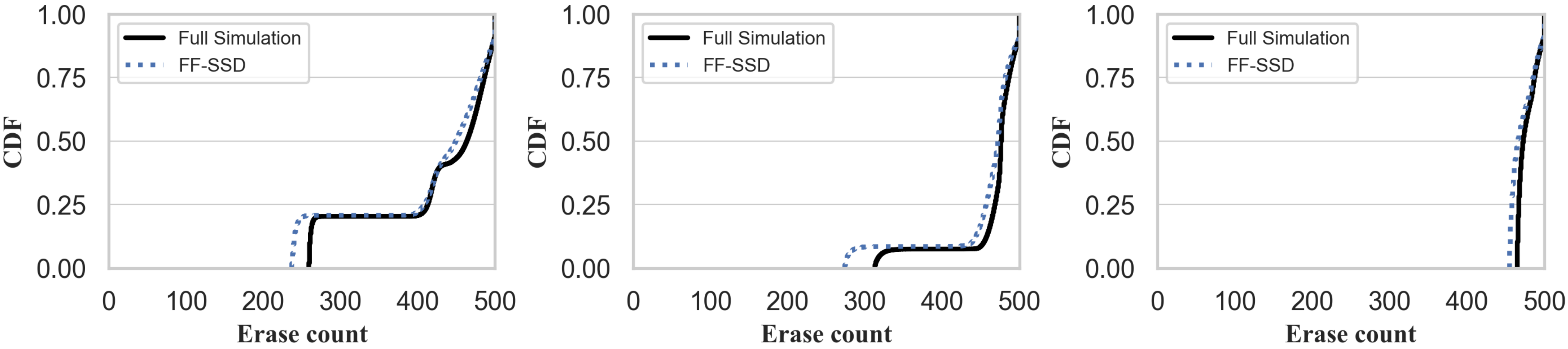
Skew norm fit to the measured distribution

Evaluation

- SSD development platforms:
  - FTLSim - SYSTOR 2012
  - Amber - MICRO 2018
  - FEMU - FAST 2018
- Workloads:
  - YCSB
  - VDI (virtual desktop infrastructure)
  - Microsoft production servers
  - Microsoft enterprise servers

FTLSim			
Pages per block	256	Physical capacity	284 GiB
Page size	4 KiB	Logical capacity	256 GiB
Endurance limit	500	Over-provisioning	0.11
Wear leveling	PWL	Garbage collection	Greedy
Amber			
Channels	8	Page size	4 KiB
Packages per channel	4	Physical capacity	285 GiB
Dies per package	2	Logical capacity	256 GiB
Planes per die	2	Over-provisioning	0.11
Blocks per plane	1136	Garbage collection	Greedy
Pages per block	512	Wear leveling	Var-based
FEMU			
Channels	8	Page size	4 KiB
Luns per channel	8	Physical capacity	16 GiB
Planes per lun	1	Logical capacity	15 GiB
Blocks per plane	256	Over-provisioning	0.07
Pages per block	256	Garbage collection	Greedy

SSD aging until failure on FTLSim

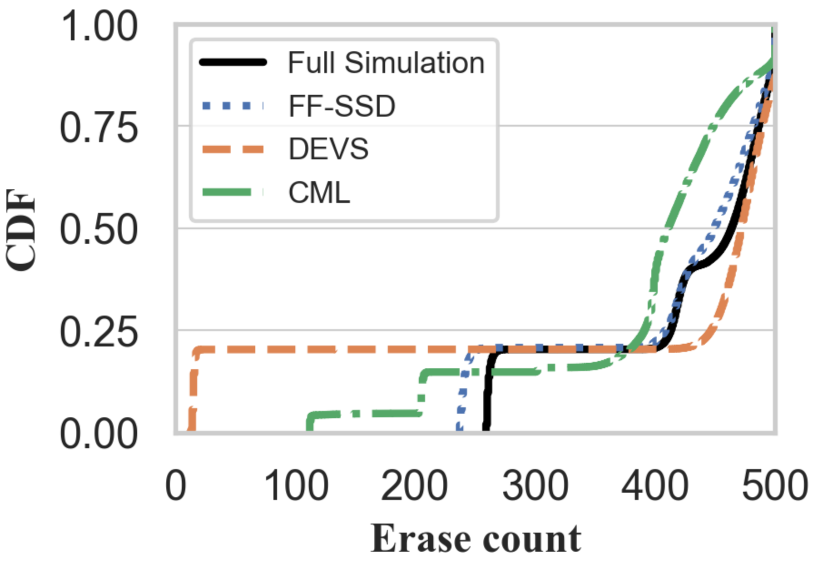


Windows Build Server

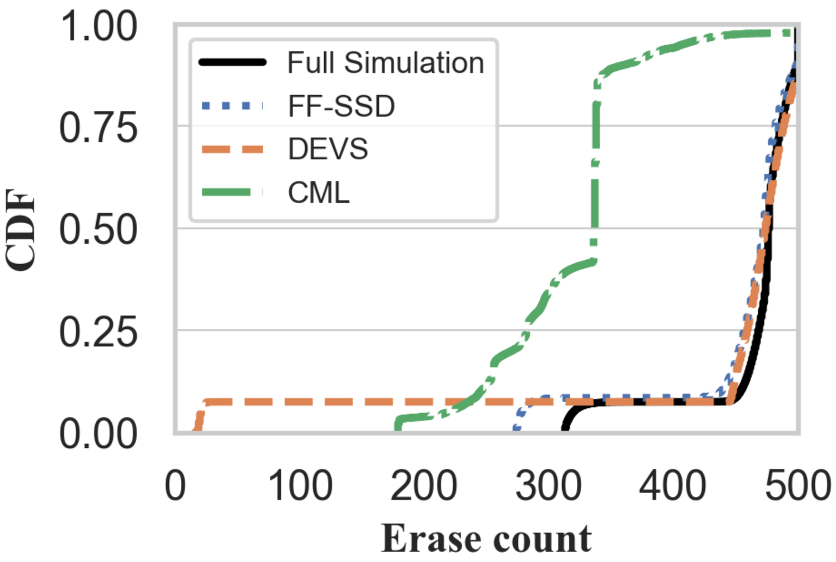
Developer Tools Release

Virtual Desktop Infra

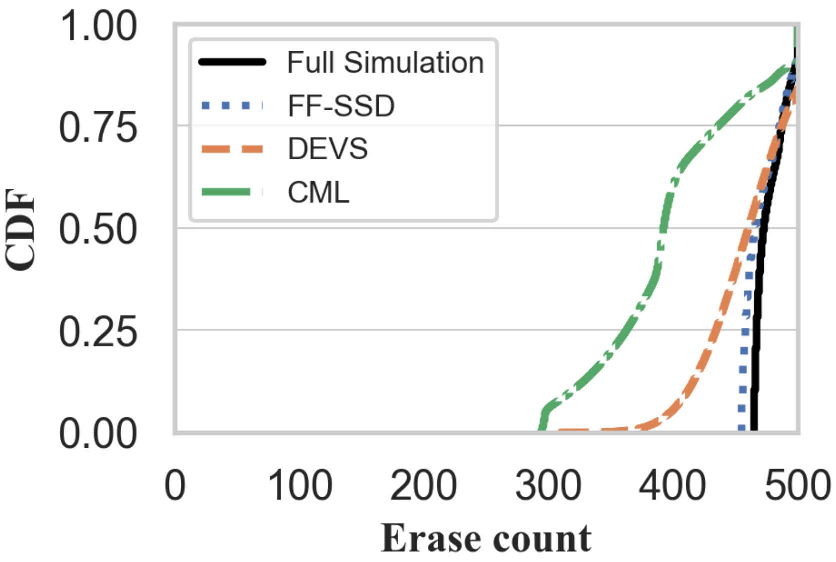
SSD aging until failure on FTLSim



Windows Build Server

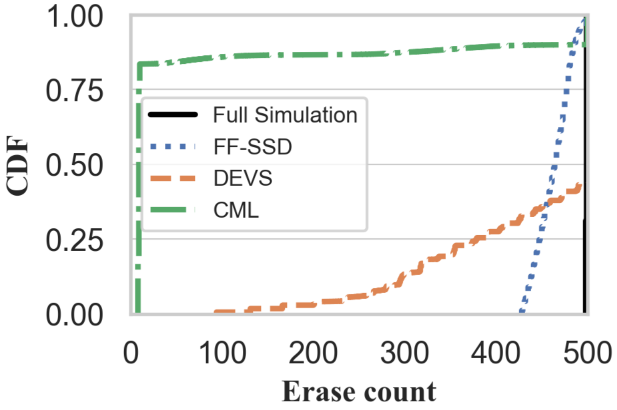


Developer Tools Release

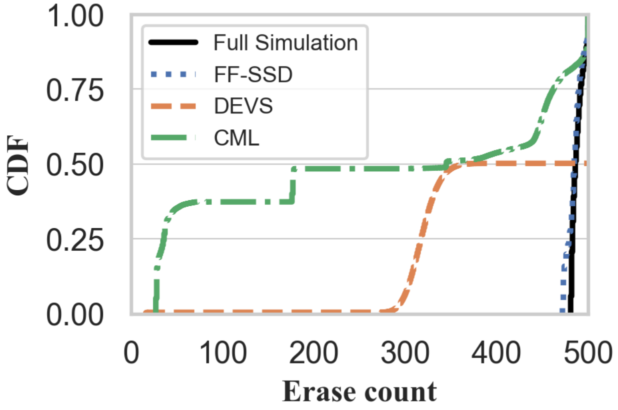


Virtual Desktop Infra

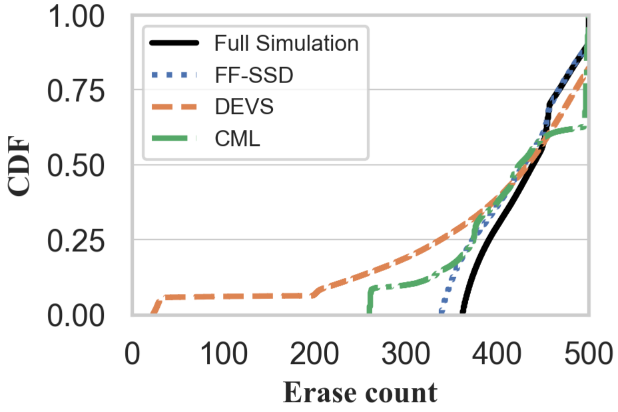
SSD aging until failure on FTLSim



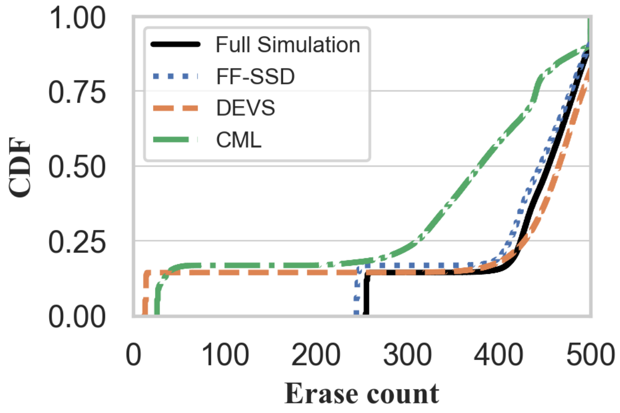
DAP-DS



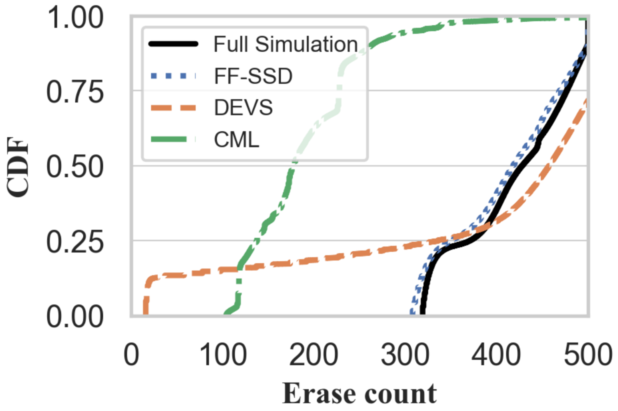
DAP-PS



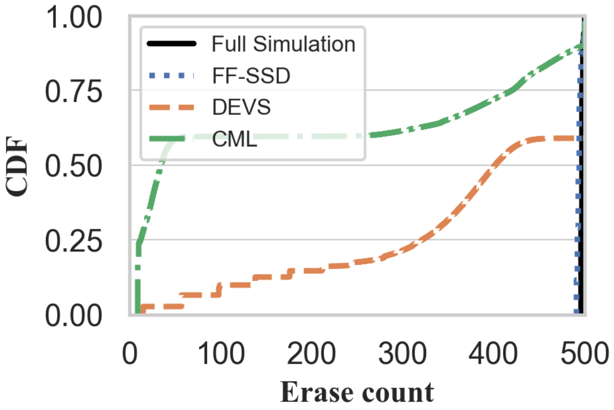
LM-TBE



MSN-BEFS

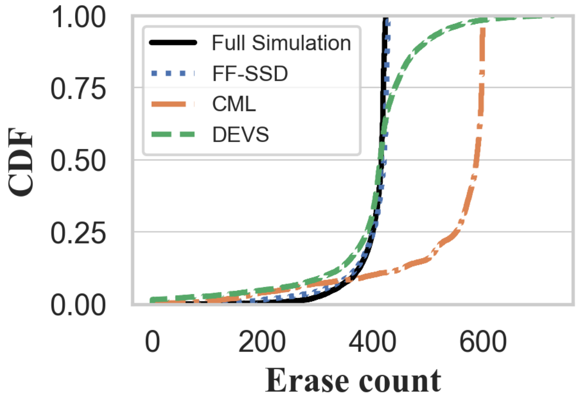


YCSB-A

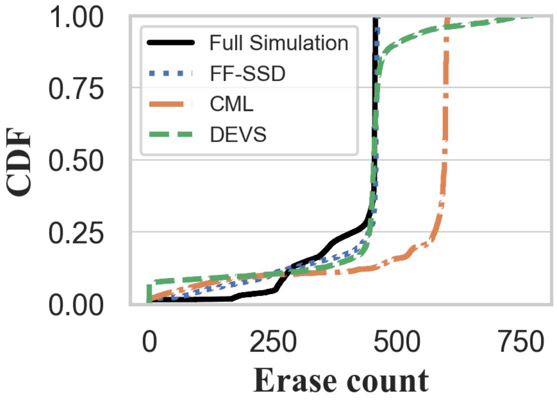


RAD-BE

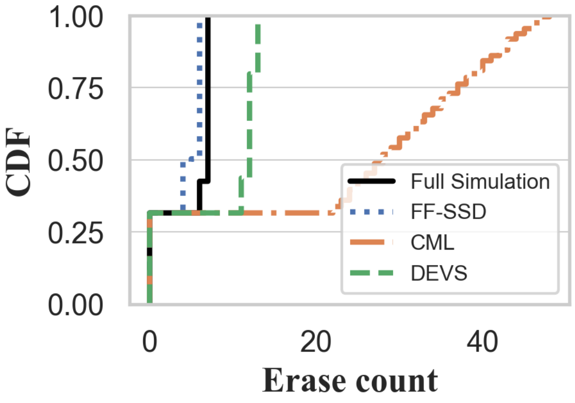
SSD aging on Amber



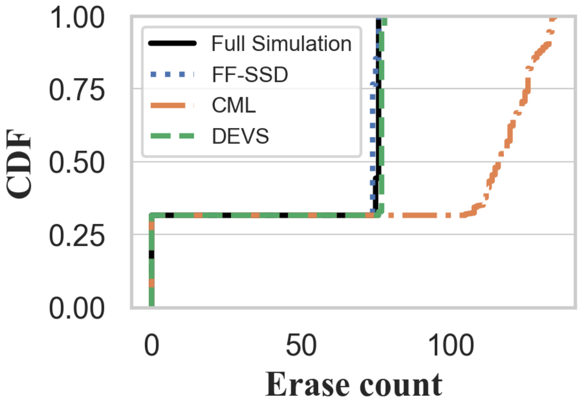
DTRS



WBS



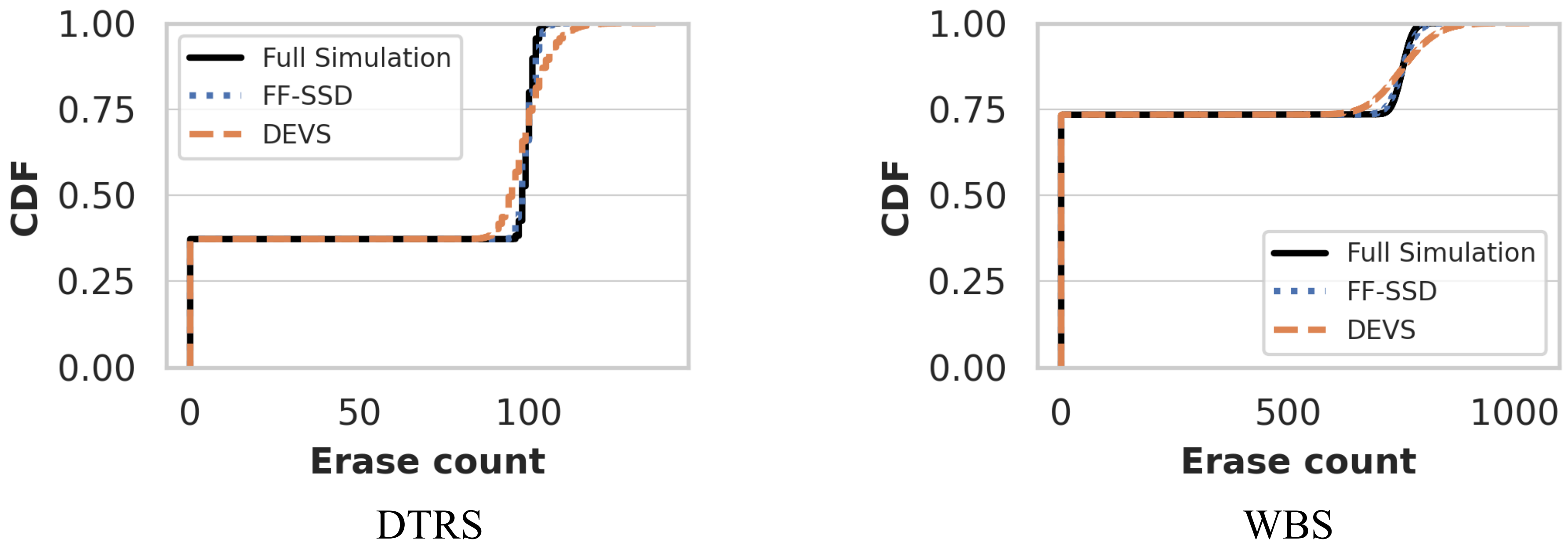
DAP-DS



RAD-RS

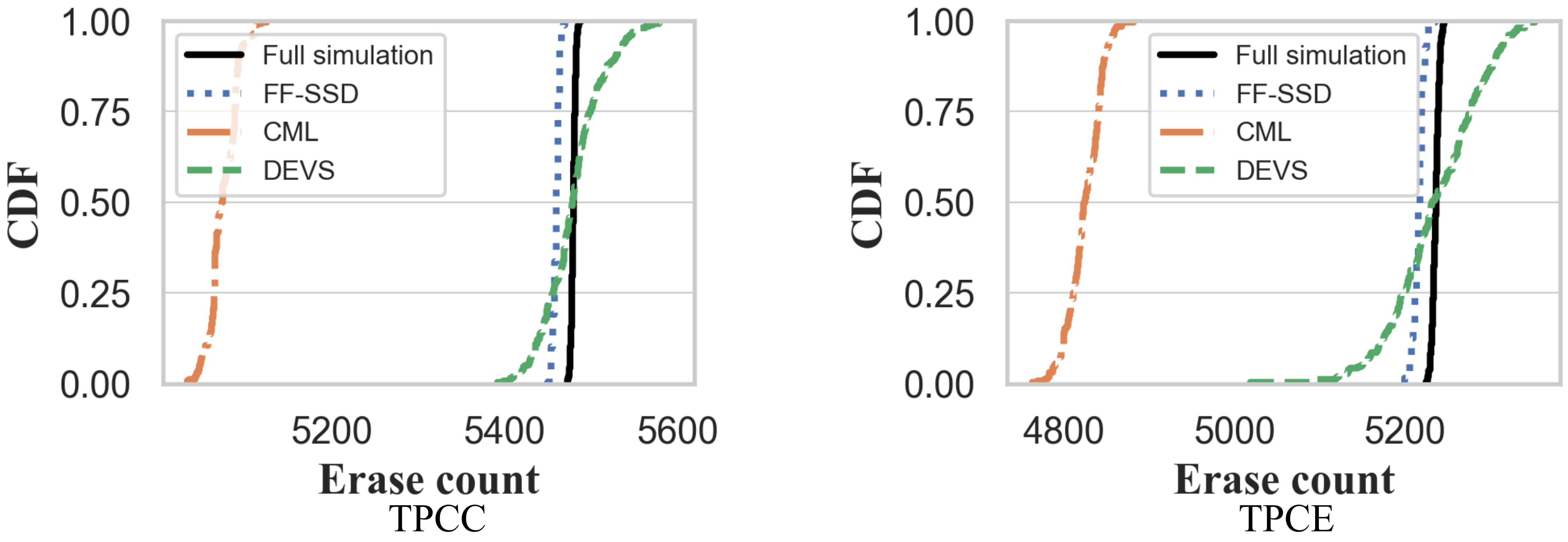
SSD aging with 600 iterations of the workloads on Amber.

Without wear leveling



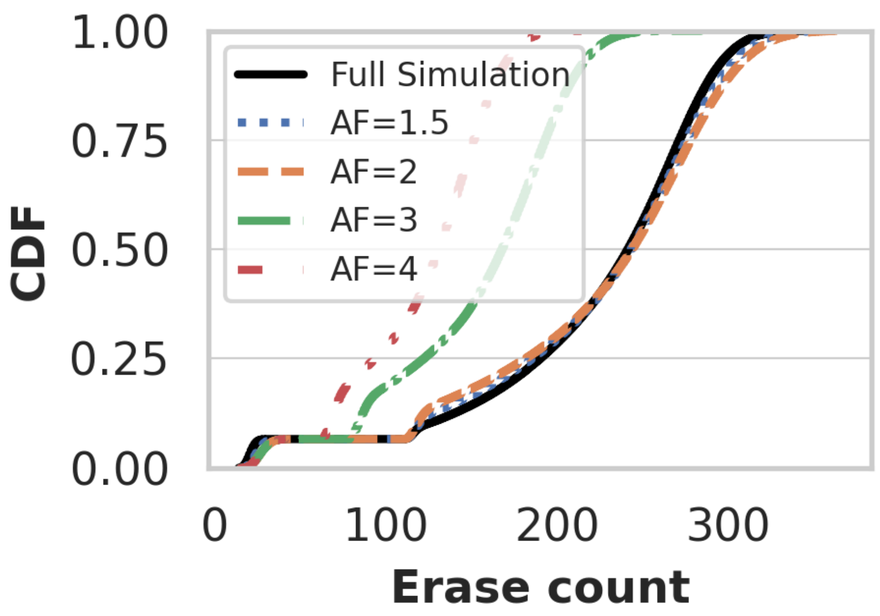
Performance comparison of FF-SSD and DEVS on FTLSim without WL.

SSD aging on FEMU

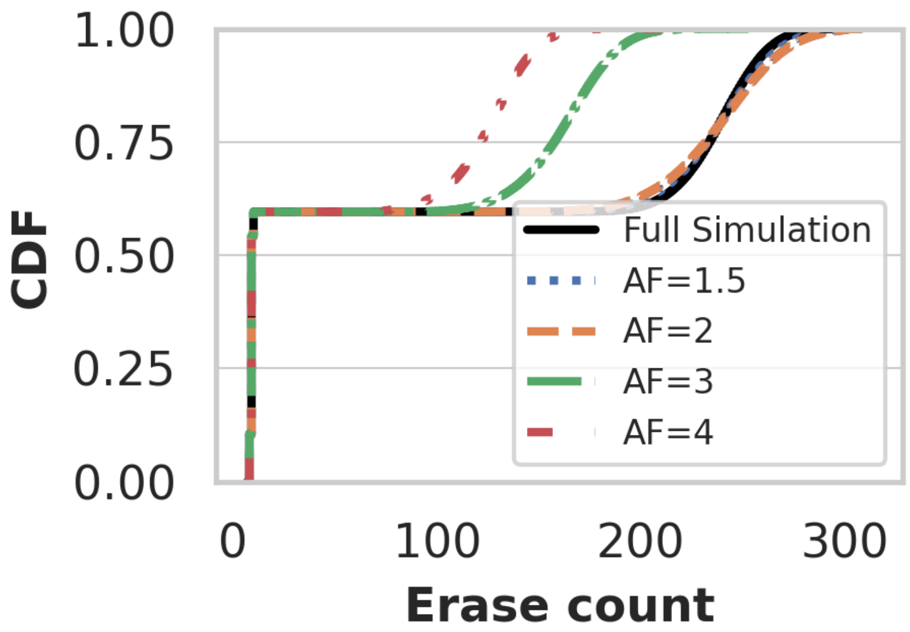


SSD aging with 50 iterations of the workloads on FEMU.

Accuracy and efficiency tradeoff



LM-TBE



RAD-BEFS

The tradeoff between aging accuracy and efficiency.

Conclusion & future work

- We present fast-forwardable SSD, an ML-based SSD aging framework that generates representative future wear-out states.
  - Accurate (up to 99% similarity)
  - Efficient (accelerates simulation time by 2×)
  - Modular (can be integrated with existing simulators and emulators)

- Codebase will be available soon
  - https://github.com/ZiyangJiao/FF-SSD

- Future work
  - Improving accuracy through adaptive acceleration.
  - Predicting on the wear states real SSDs
  - More promising directions…

Thank you!

Ziyang Jiao
zjiao04@syr.edu