Accelerating SSD Steady-State Analysis using a WAF-based Performance Interface

Roshan Nair
DSLAB, EPFL

George Candea
DSLAB, EPFL

Takeaway: SSD steady-state analysis can be done using WAF as a proxy for performance and lifetime.

Reasoning with SSD performance is challenging!

Cause
- Complex internal policies
- Transparent operation
- Internal data relocation

➡

Effect
- Huge write amplification
- Unpredictable performance
- Reduced SSD lifetime

WAF is a proxy for perf.

$$\text{WAF} = \frac{\text{NAND Writes}}{\text{Host Writes}}$$

WAF ↑ = Perf. ↓

WAF ↑ = Lifetime ↓

Fresh SSD Performance != Real Performance

Fresh SSD
- No internal ops triggered
- Not representative (WAF = 1)

➡

Steady-State SSD
- Internal ops stabilize
- Representative (WAF >=1)

Time to steady-state

- Writes of 2x to 10x SSD size needed.
- Can takes hours, days, or even weeks…

Challenge: SSD steady-state analysis takes *hours to weeks*.

Status Quo = Repeated Benchmarking = Extremely SLOW!

Reset SSD ➡ Run Precondition ➡ Run Experiment ➡ ⌛ Wait for steady-state perf. (hours, days, or weeks…)

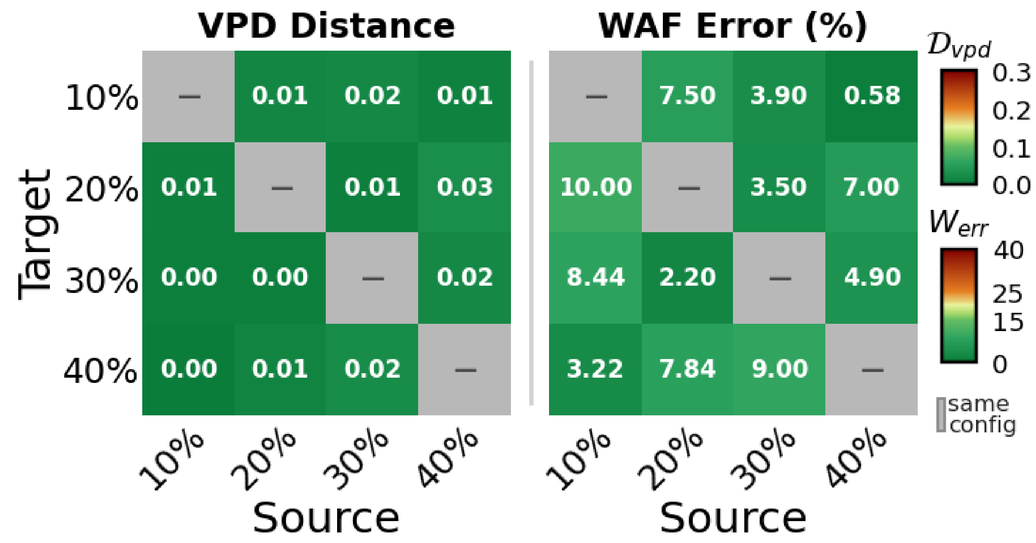
↻ Repeat slow and inefficient workflow for ***every*** change of interest.

Solution: Mathematically Transform SSD state in *seconds*!

DITTO: A WAF-estimation based toolchain for steady-state analysis!

1. Source <SSD Config, Workload> ➡ Run to steady-state (only ***once***) ➡ Capture SSD state using WAF and VPD (Valid Page Distribution)

2. Target Configs ✚ *Fixed* Source <Configs, VPD>
   ⬇
   Mathematical Transform (DITTO Transform) = Predicted VPD, WAF

↻ Repeat for different target configs.
Get target steady-state WAF in ***seconds***!

100% rand. write, varying OP