ENDURE: A Robust Tuning Paradigm for LSM Trees Under Workload Uncertainty

Andy Huynh, Harshal A. Chaudhari, Evimaria Terzi, Manos Athanassoulis

Age of Log-Structured Merge-Trees

RocksDB · levelDB · SQLite · cassandra · WT · accumulo · APACHE HBASE · DynamoDB

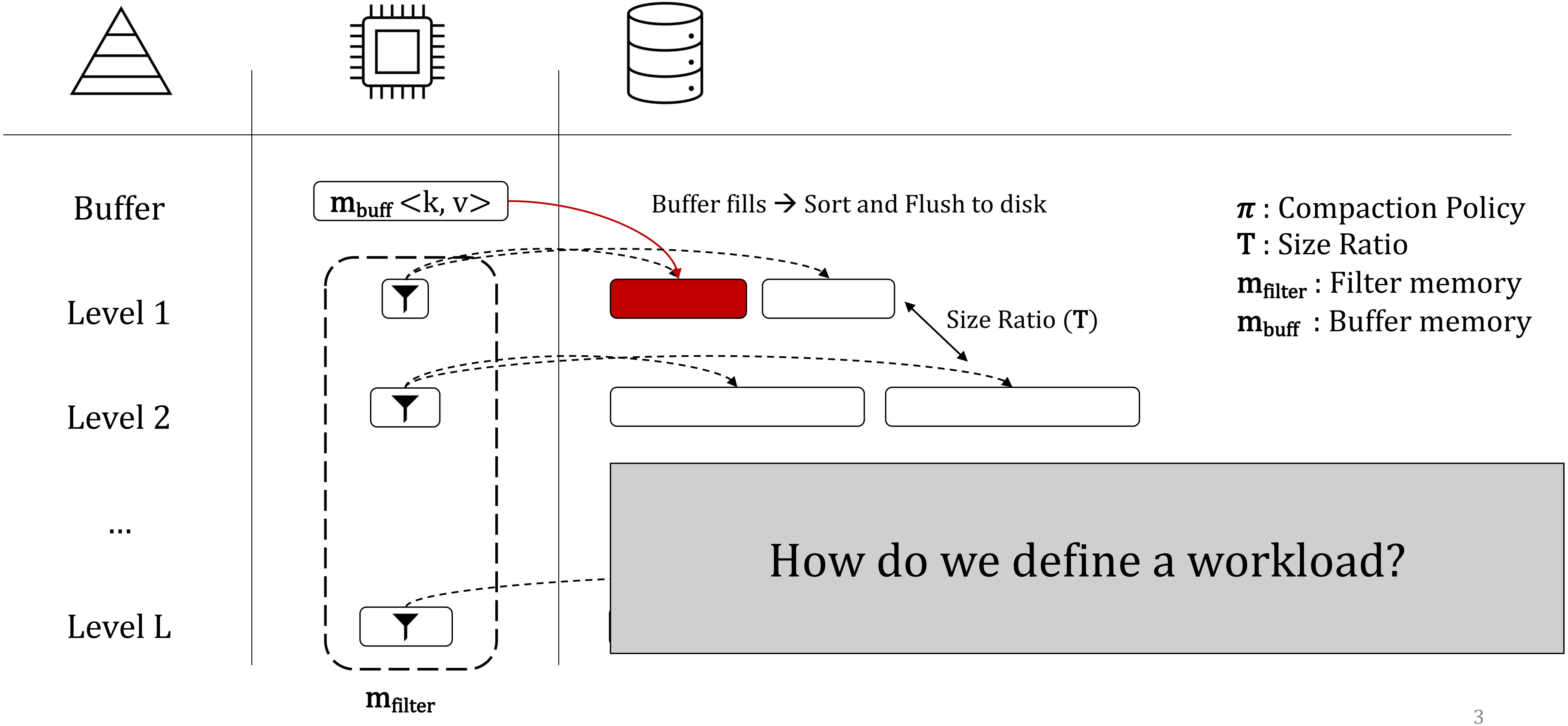
High impact tuning knobs

Compaction · Buffer size · Size ratio

Dictates performance!

How do we go about tuning these knobs?

LSM Trees

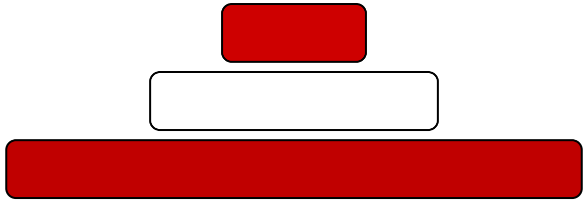


How do we define a workload?

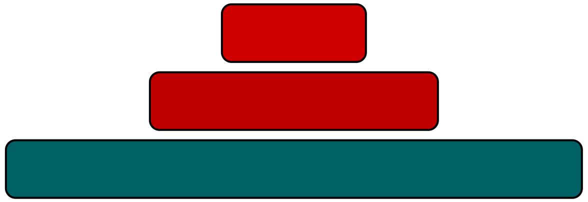
Query Types

Workload : (z_0, z_1, q, w)

Empty Reads : z_0

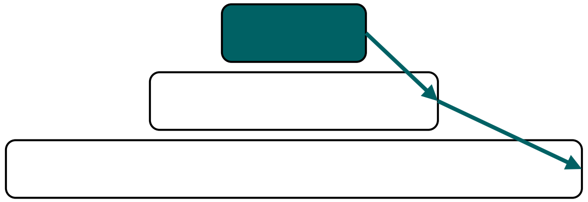


Non-Empty Reads : z_1



Range Reads: q

Writes : w



Cool! How do we go about tuning?

The LSM-Tuning Problem

$\boldsymbol{w}$: Workload   (z_0, z_1, q, w)

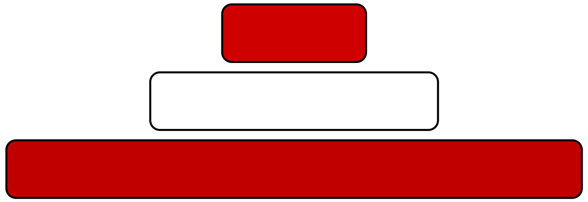
Φ : LSM Tree Design $(m_{buff}, m_{filter}, T, \pi)$

C : Cost

$$\Phi^* = argmin_{\Phi}\, C(\boldsymbol{w}, \Phi)$$

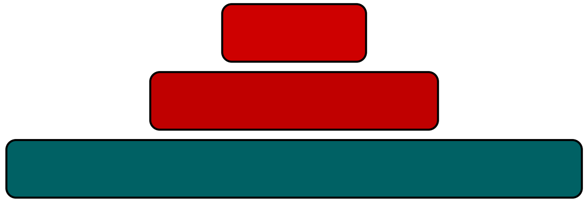
Point Reads

Empty Reads : z_0



$$Z_0(\Phi) = \overbrace{\sum_{i=1}^{L} f_i}^{\text{Sum of false positives}}$$

Non-Empty Reads : z_1

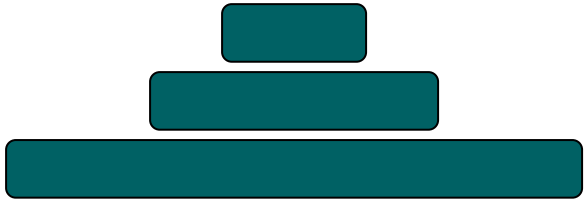


$$Z_1(\Phi) = \overbrace{\sum_{i=1}^{L} \frac{T^{i-1}\cdot(T-1)}{N_f(T)} \cdot \frac{m_{buf}}{E}}^{\text{Probability query is satisfied at level i}} \overbrace{\left(1 + \sum_{j=1}^{i-1} f_j\right)}^{\text{False positives from levels above}}$$

[1] *Niv Dayan, Manos Athanassoulis, and Stratos Idreos. 2017. Monkey: Optimal Navigable Key-Value Store. In Proceedings of the 2017 ACM International Conference on Management of Data (SIGMOD '17).*

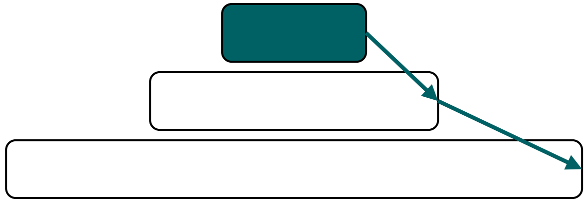
Range-Reads and Writes

Range Reads: q



$$Q(\Phi) = \overbrace{S_{RQ} \cdot \frac{N}{B}}^{\text{Sequential read based on selectivity}} + \underbrace{L}_{\text{1 I/O per Seek per level}}$$

Writes : w



$$W(\Phi) = \overbrace{\underbrace{\frac{L}{B}}_{\text{Writes only flush once buffer is full}} \cdot \frac{T-1}{2} \cdot (1 + A_{rw})}^{\text{Average number of merges a write will participate in}}$$

[1] *Niv Dayan, Manos Athanassoulis, and Stratos Idreos. 2017. Monkey: Optimal Navigable Key-Value Store. In Proceedings of the 2017 ACM International Conference on Management of Data (SIGMOD '17).*

The LSM-Tuning Problem

$\boldsymbol{w}$: Workload (z_0, z_1, q, w)

Φ : LSM Tree Design $(m_{buff}, m_{filter}, T, \pi)$

C : Cost (I/O)

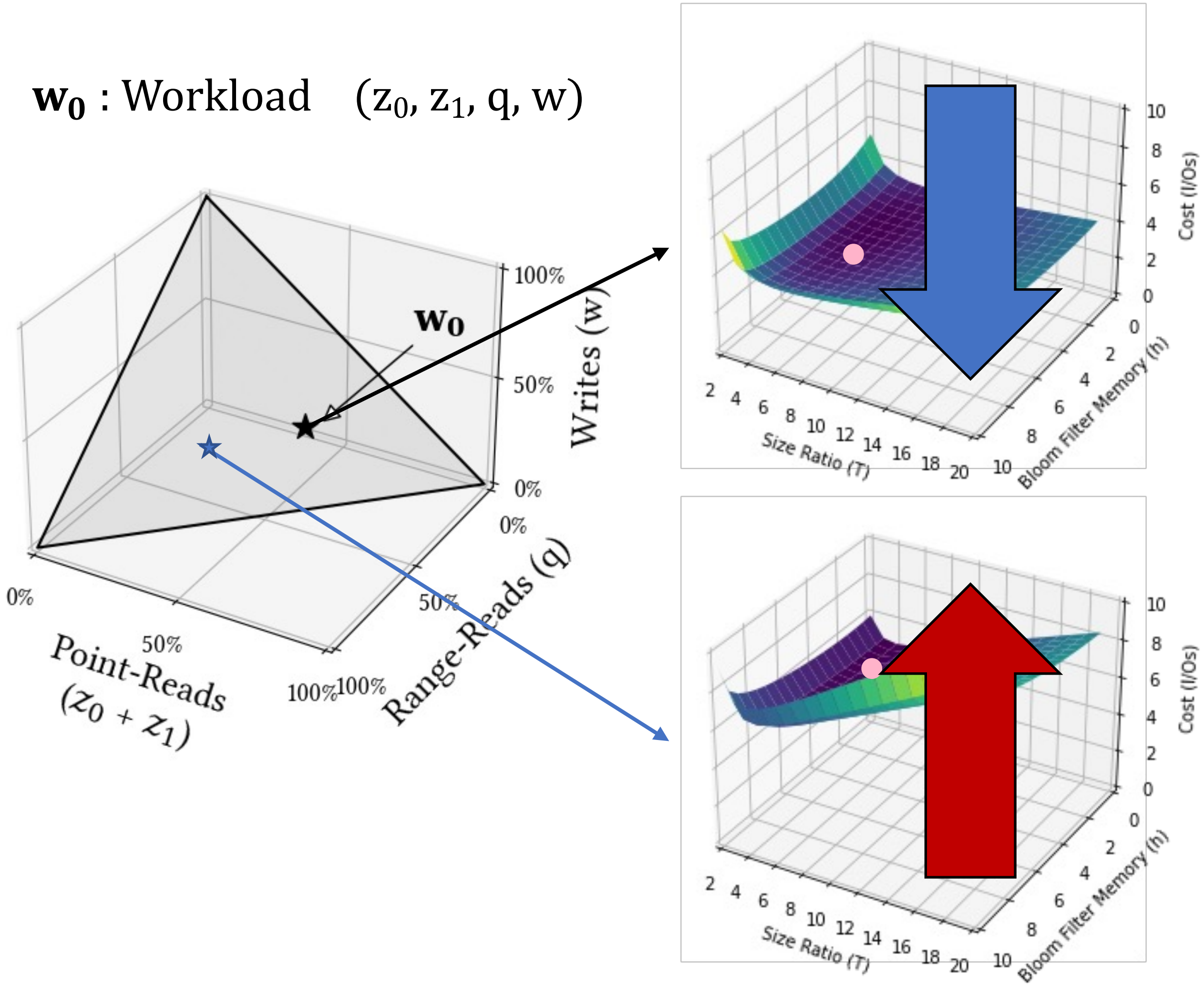
$$\Phi^* = argmin_{\Phi}\, C(\boldsymbol{w}, \Phi)$$

Define our cost function

$$C(\hat{\mathbf{w}}, \Phi) = \hat{\mathbf{w}}^\intercal \mathbf{c}(\Phi) = z_0 \cdot Z_0(\Phi) + z_1 \cdot Z_1(\Phi) + q \cdot Q(\Phi) + w \cdot W(\Phi)$$

Tuning Problems

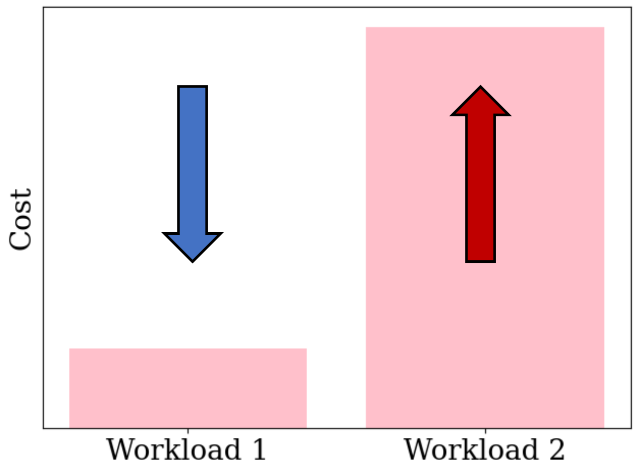
$\mathbf{w_0}$: Workload (z$_0$, z$_1$, q, w)



Optimal configuration for the workload

Optimal tuning depends on workload

Workload uncertainty leads to **sub-optimal** tuning



Outline

Introduction

LSM Trees Notation

Nominally Tuning LSM Trees

ENDURE: Robustly Tuning LSM Trees

The ENDURE Pipeline

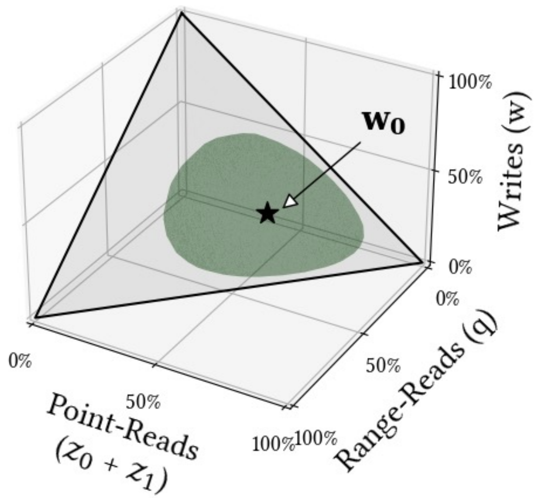
ENDURE Evaluation

The LSM-Tuning Problem

w : Workload (z_0, z_1, q, w)

Φ : LSM Tree Design $(m_{buff}, m_{filter}, T, \pi)$

C : Cost (I/O)



$$\Phi^* = argmin_\Phi\, C(\boldsymbol{w}, \Phi)$$

Nominal

U_w^ρ: Uncertainty Neighborhood of Workloads

ρ : Size of this neighborhood

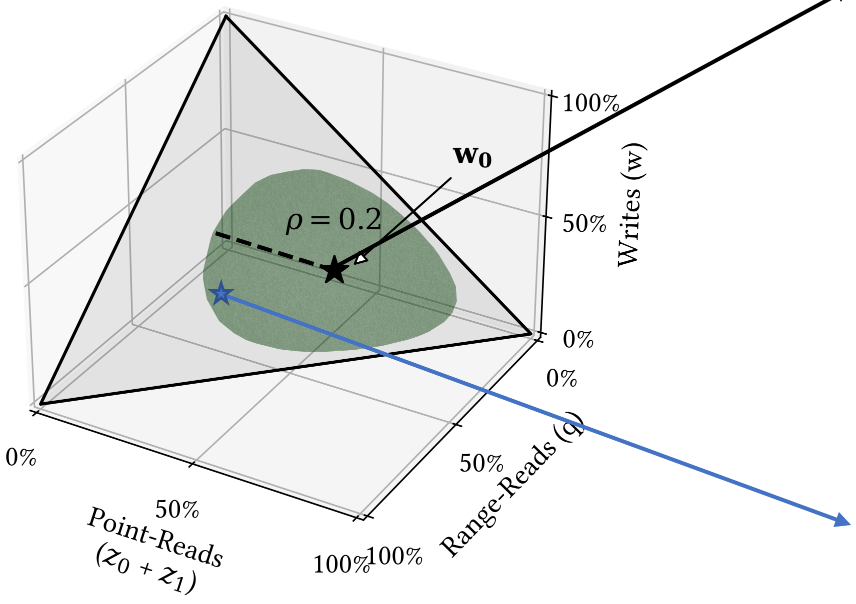
Robust

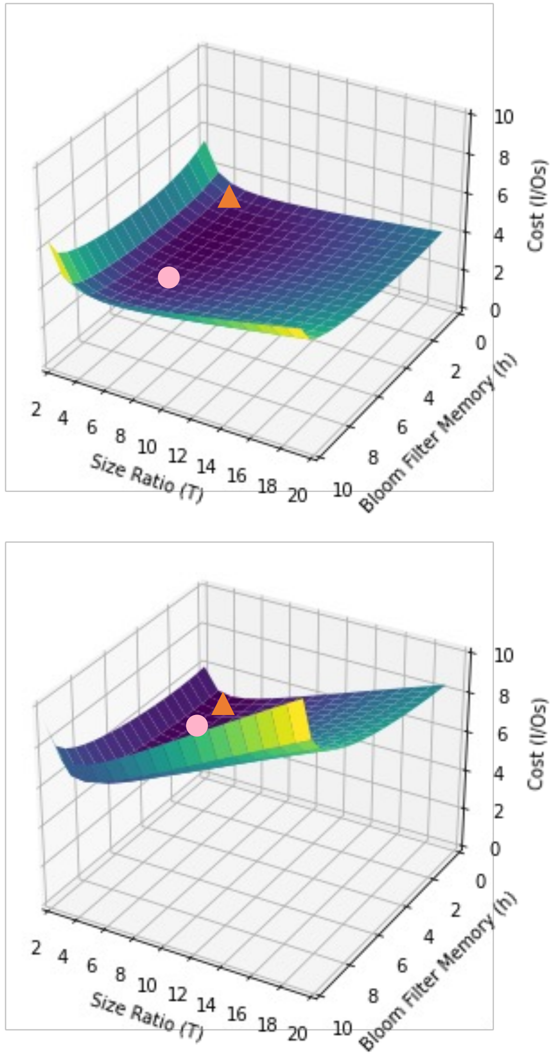
$$\Phi^* = \mathrm{argmin}_\Phi\, C(\hat{\boldsymbol{w}}, \Phi)$$

$$s.t., \quad \hat{\boldsymbol{w}} \in U_w^\rho$$

Robust Tuning

$\mathbf{w_0}$: Workload (z_0, z_1, q, w)



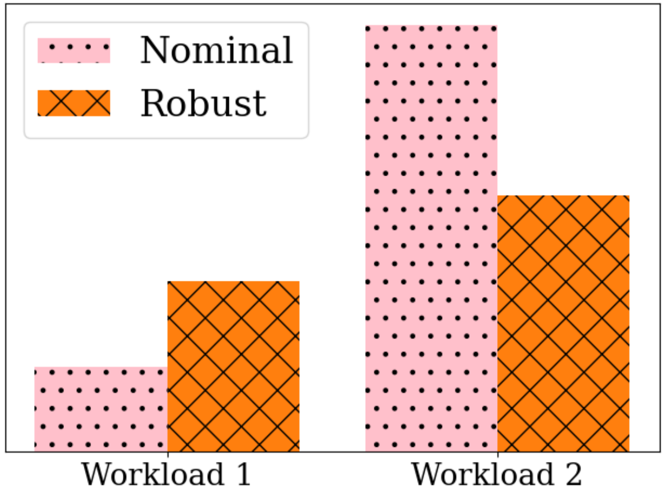


$$\Phi^* = \text{argmin}_\Phi \, C(\hat{\boldsymbol{w}}, \Phi)$$

$$s.t., \quad \hat{\boldsymbol{w}} \in U_w^\rho$$

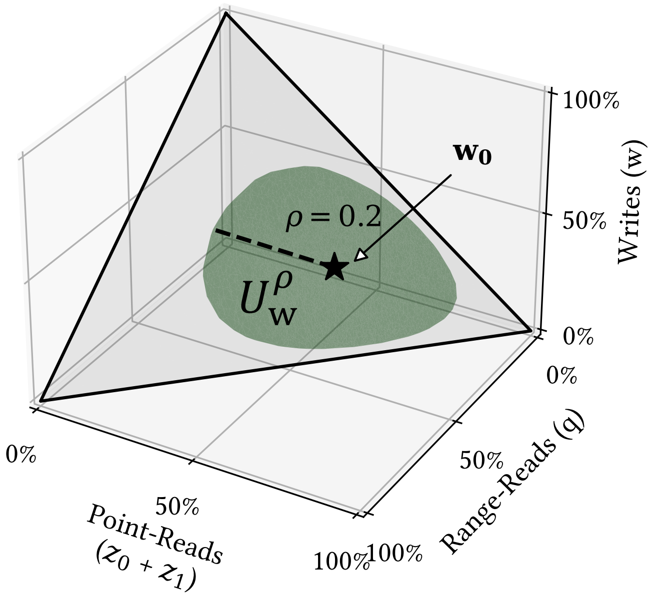
Optimal configuration for the workload

Robust configuration for the workload neighborhood



Uncertainty Neighborhood

Workload Characteristic

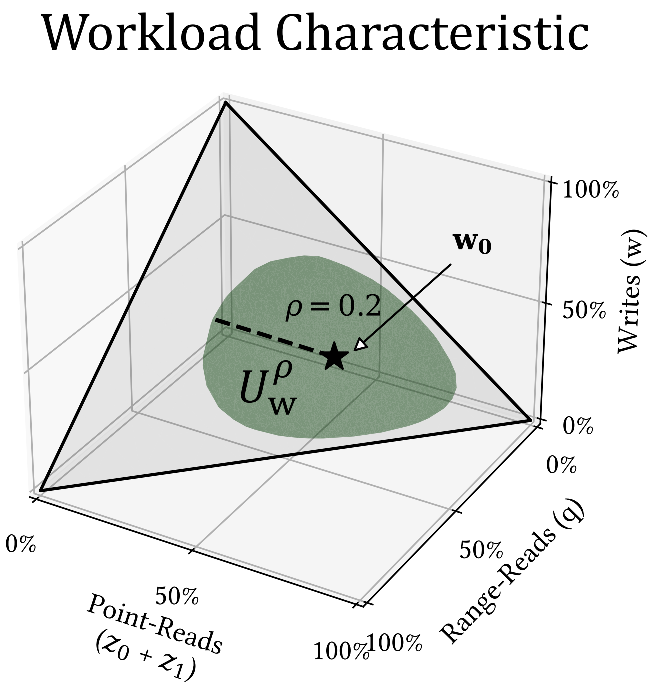


Neighborhood of workloads (ρ) via the KL-divergence

$$I_{KL}(\widehat{w}, w) = \sum_{i=1}^{m} \widehat{w}_i \cdot \log(\frac{\widehat{w}_i}{w_i})$$

U_{w}^{ρ}: Uncertainty Neighborhood of Workloads
ρ : Size of this neighborhood

Calculating Neighborhood Size



Historical workloads
maximum/average uncertainty among workload pairings

User provided workload uncertainty

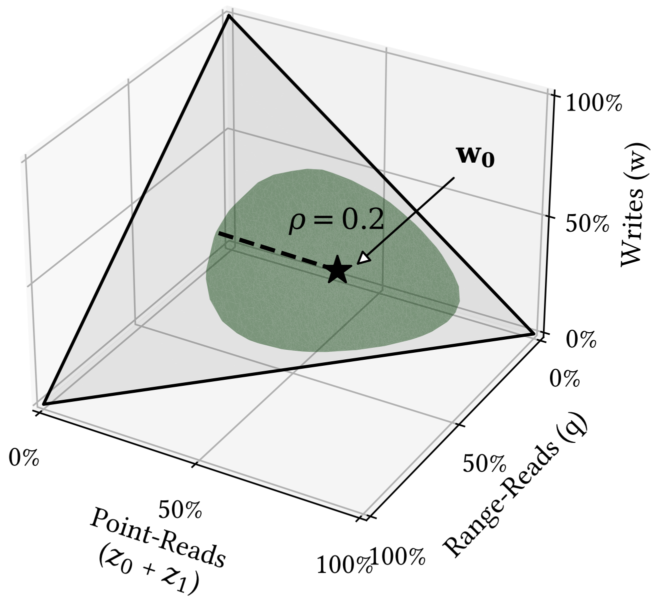
U_{w}^{ρ}: Uncertainty Neighborhood of Workloads
ρ : Size of this neighborhood

Solving Robust Problem

Iterating over every possible workload is expensive

$$\Phi_R = \arg\min_{\Phi} \ \hat{\mathbf{w}}^\intercal \mathbf{c}(\Phi)$$

$$\text{s.t.} \quad \hat{\mathbf{w}} \in \mathcal{U}^{\rho}_{\mathbf{w}}$$



Solving Robust Problem

Iterating over every possible workload is expensive

$$\Phi_R = \arg\min_{\Phi} \ \hat{\mathbf{w}}^\intercal \mathbf{c}(\Phi)$$
$$\text{s.t.} \quad \hat{\mathbf{w}} \in \mathcal{U}^{\rho}_{\mathbf{w}}$$

Rewrite as a min-max

$$\min_{\Phi} \max_{\hat{\mathbf{w}} \in \mathcal{U}^{\rho}_{\mathbf{w}}} \hat{\mathbf{w}}^\intercal \mathbf{c}(\Phi)$$

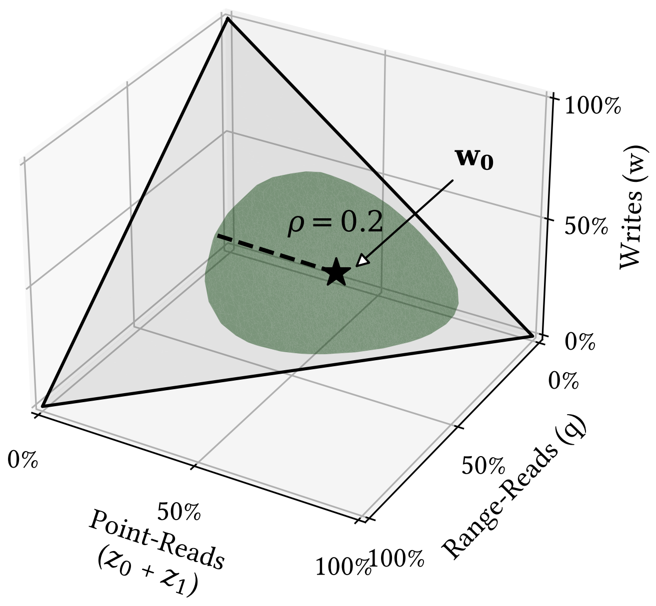
Find the dual of the maximization problem to reduce to a feasible problem [2]

$$\min_{\Phi, \lambda \geq 0, \eta} \left\{ \eta + \rho\lambda + \lambda \sum_{i=1}^{m} w_i \phi^*_{KL} \left(\frac{c_i(\Phi) - \eta}{\lambda} \right) \right\}$$

[2] Aharon Ben-Tal, Dick den Hertog, Anja De Waegenaere, Bertrand Melenberg, and Gijs Rennen. 2013. Robust Solutions of Optimization Problems Affected by Uncertain Probabilities.

ENDURE Pipeline

Workload Characteristic



System Information

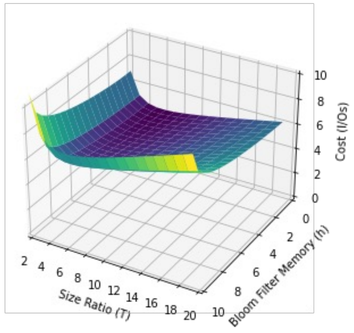
Page Size

Memory Budget

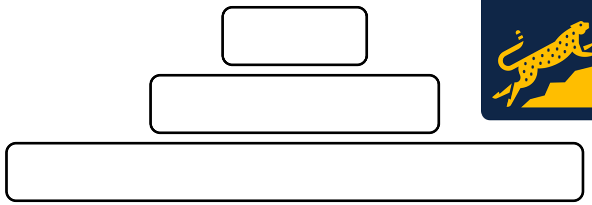
ENDURE

Solves the Robust Problem

Expected performance



RocksDB Configuration

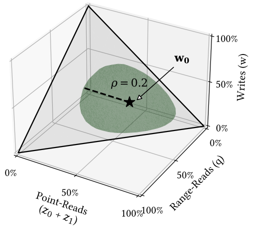


Testing Suite

ENDURE in Python, implemented in tandem with RocksDB

Uncertainty benchmark

- 15 expected workloads
- 10K randomly sampled workloads as a test-set



Normalized delta throughput

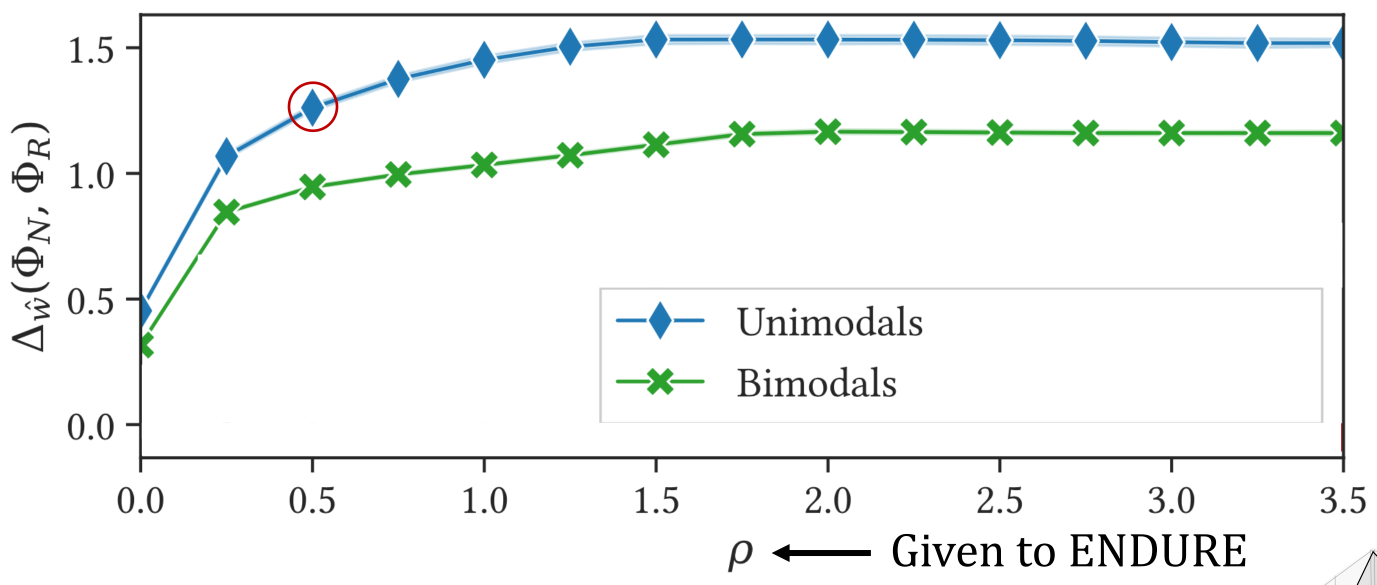
$$\Delta_{\mathbf{w}}(\Phi_1, \Phi_2) = \frac{1/C(\mathbf{w},\Phi_2) - 1/C(\mathbf{w},\Phi_1)}{1/C(\mathbf{w},\Phi_1)}$$

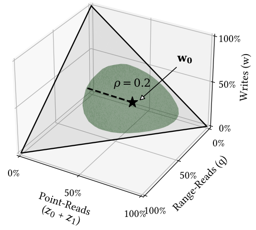
Nominal vs Robust: > 0 is better

1 means 2x speedup

Index	(z_0, z_1, q, w)				Type
0	25%	25%	25%	25%	**Uniform**
1	97%	1%	1%	1%	**Unimodal**
2	1%	97%	1%	1%	
3	1%	1%	97%	1%	
4	1%	1%	1%	97%	
5	49%	49%	1%	1%	**Bimodal**
6	49%	1%	49%	1%	
7	49%	1%	1%	49%	
8	1%	49%	49%	1%	
9	1%	49%	1%	49%	
10	1%	1%	49%	49%	
11	33%	33%	33%	1%	**Trimodal**
12	33%	33%	1%	33%	
13	33%	1%	33%	33%	
14	1%	33%	33%	33%	

Impact of Workload Type

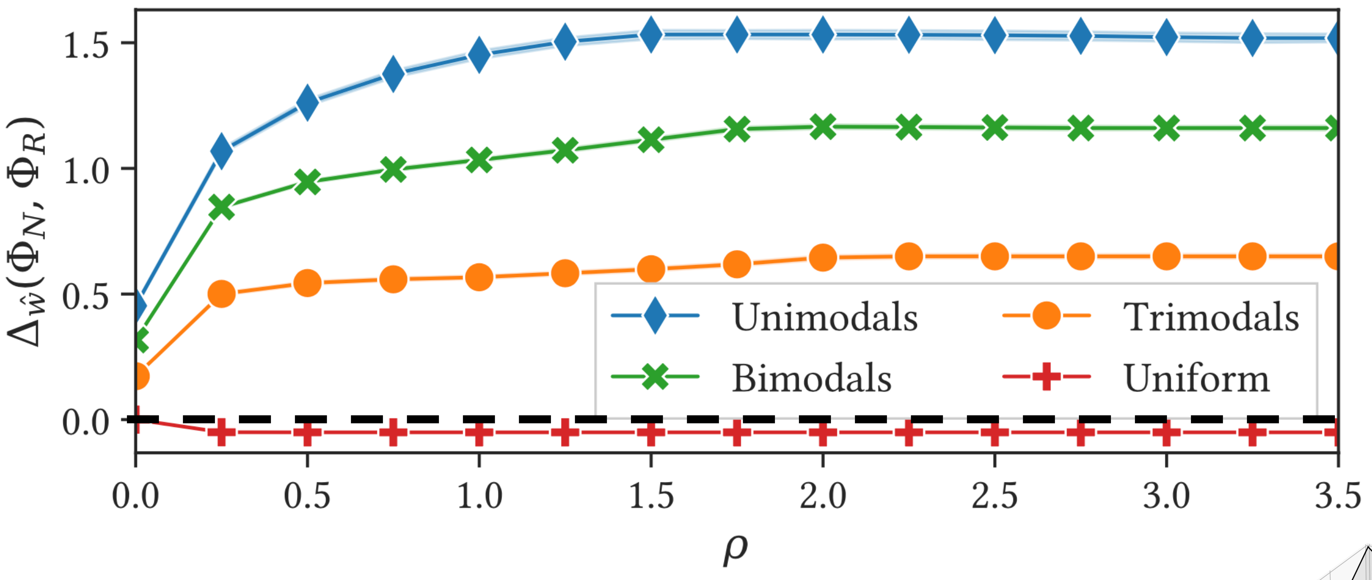




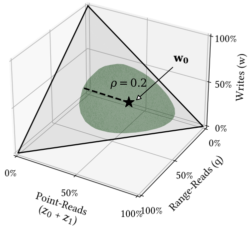
Index	$(z_0,$	$z_1,$	$q,$	$w)$	Type
0	25%	25%	25%	25%	**Uniform**
1	97%	1%	1%	1%	**Unimodal**
2	1%	97%	1%	1%	
3	1%	1%	97%	1%	
4	1%	1%	1%	97%	
5	49%	49%	1%	1%	**Bimodal**
6	49%	1%	49%	1%	
7	49%	1%	1%	49%	
8	1%	49%	49%	1%	
9	1%	49%	1%	49%	
10	1%	1%	49%	49%	
11	33%	33%	33%	1%	**Trimodal**
12	33%	33%	1%	33%	
13	33%	1%	33%	33%	
14	1%	33%	33%	33%	

Unbalanced workloads result in overfitted nominal tunings

Impact of Workload Type



Index	(z_0, z_1, q, w)				Type
0	25%	25%	25%	25%	**Uniform**
1	97%	1%	1%	1%	**Unimodal**
2	1%	97%	1%	1%	
3	1%	1%	97%	1%	
4	1%	1%	1%	97%	
5	49%	49%	1%	1%	**Bimodal**
6	49%	1%	49%	1%	
7	49%	1%	1%	49%	
8	1%	49%	49%	1%	
9	1%	49%	1%	49%	
10	1%	1%	49%	49%	
11	33%	33%	33%	1%	**Trimodal**
12	33%	33%	1%	33%	
13	33%	1%	33%	33%	
14	1%	33%	33%	33%	

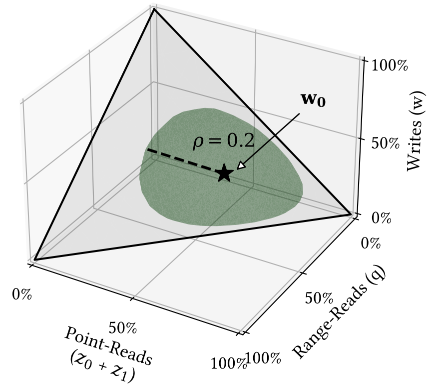


<u>Unbalanced</u> workloads result in overfitted nominal tunings

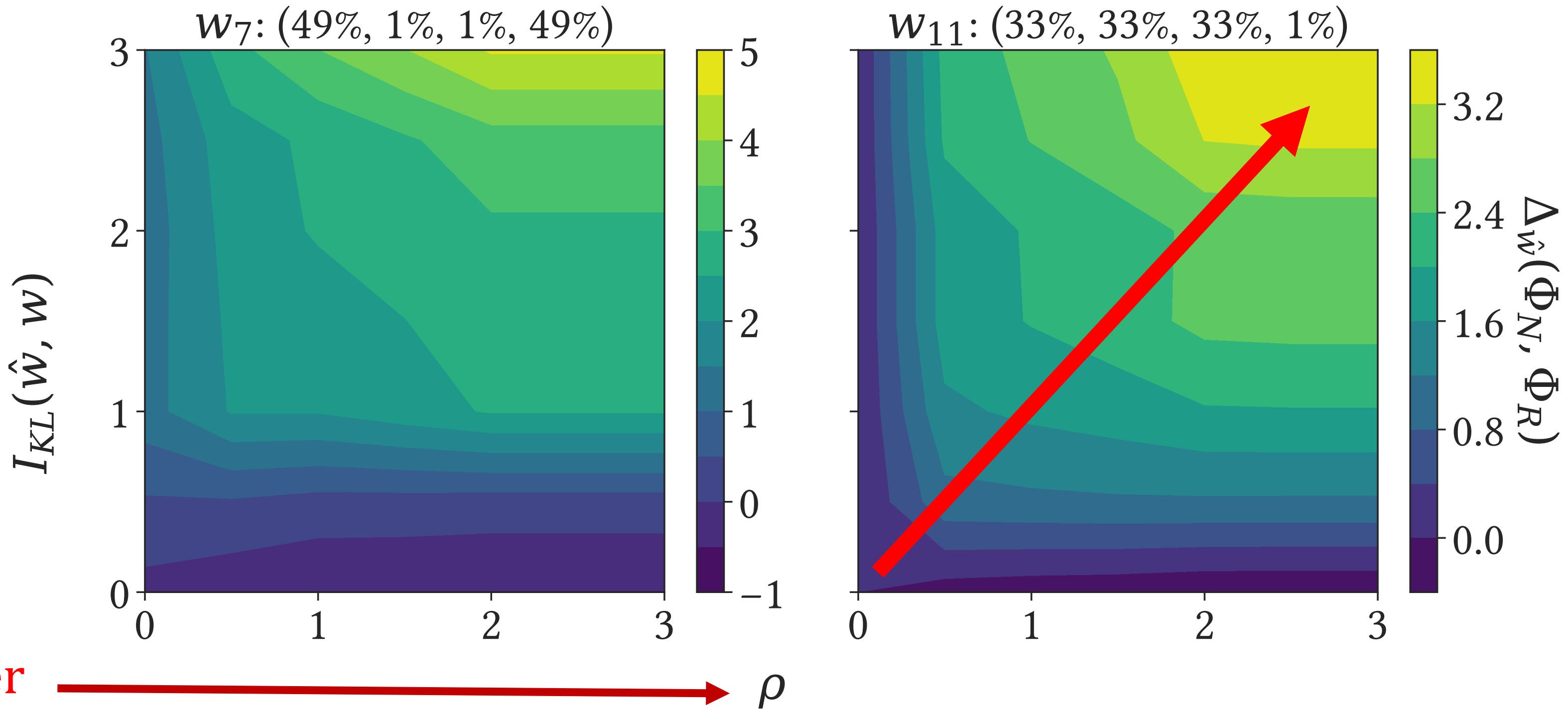
Tuning with uncertainty ($\rho > 0.5$) provides benefits

Relationship of Expected and Observed ρ

Observed ρ: distance from executed workload to expected workload



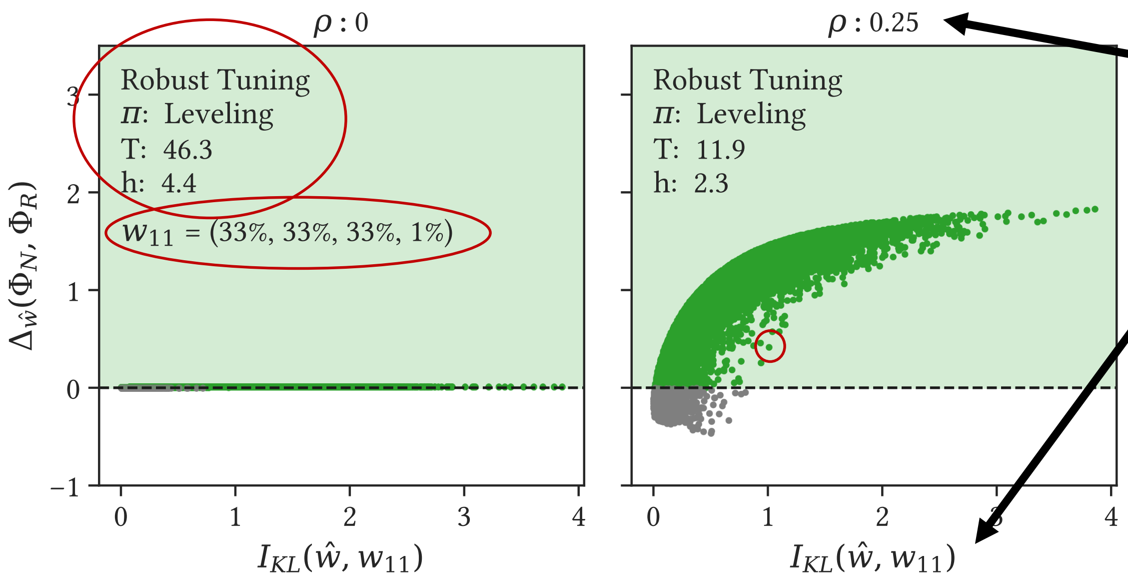
Expected ρ: workload given to tuner



Highest throughput when observed and expected ρ **match**

Lowest throughput when ρ is **mismatched**

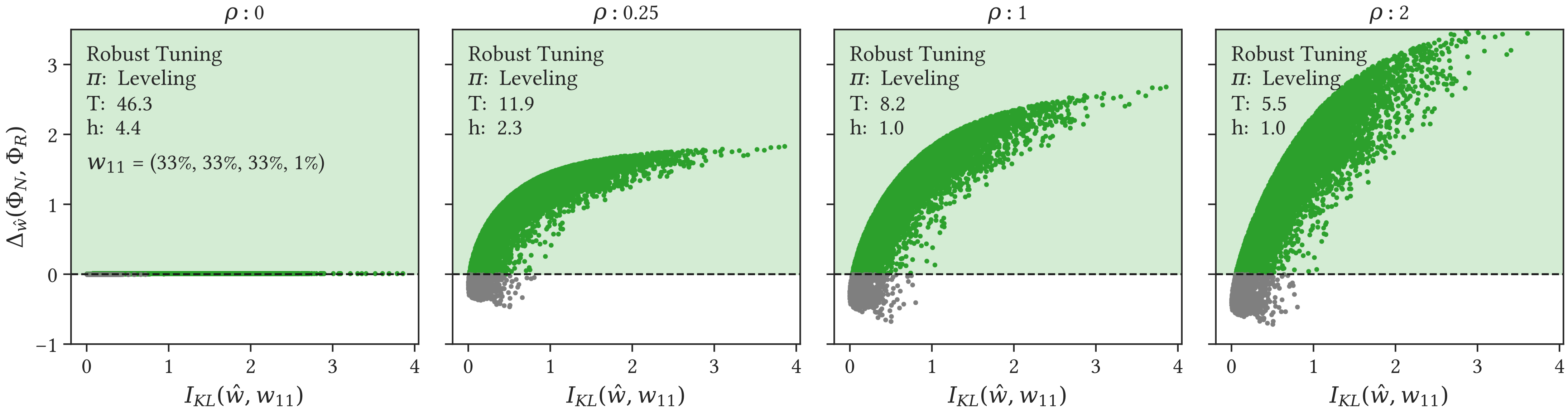
Impact of Observed vs Expected ρ



Expected ρ: given to tuner

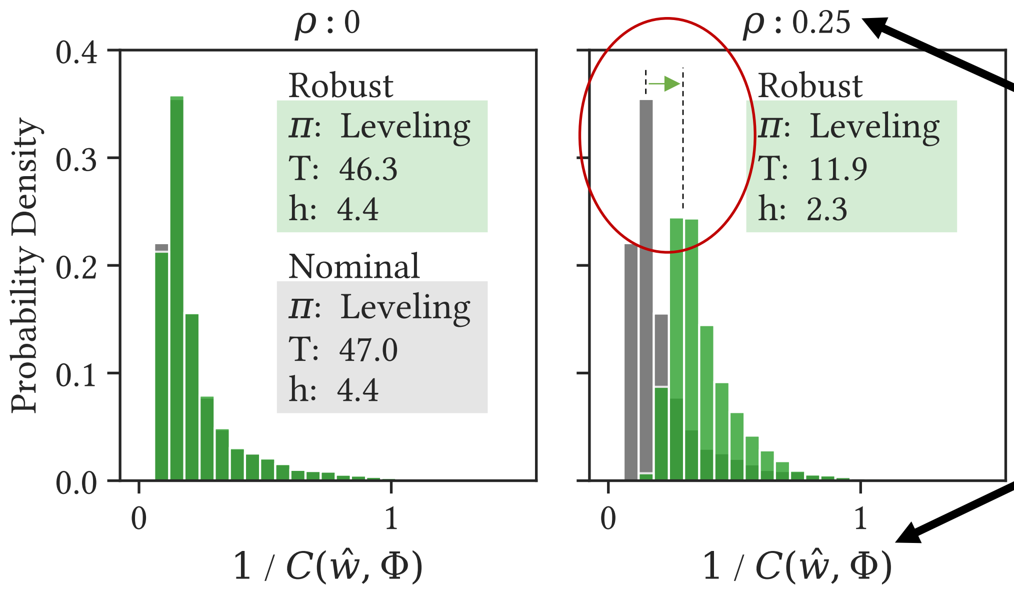
Observed ρ: distance from executed workload to expected workload

Impact of Observed vs Expected ρ



- Higher expected ρ accounts for more uncertainty,
- Potential speed up of 4x
- Higher expected ρ → anticipates writes → shallow tree

ρ and Performance Gain Distribution

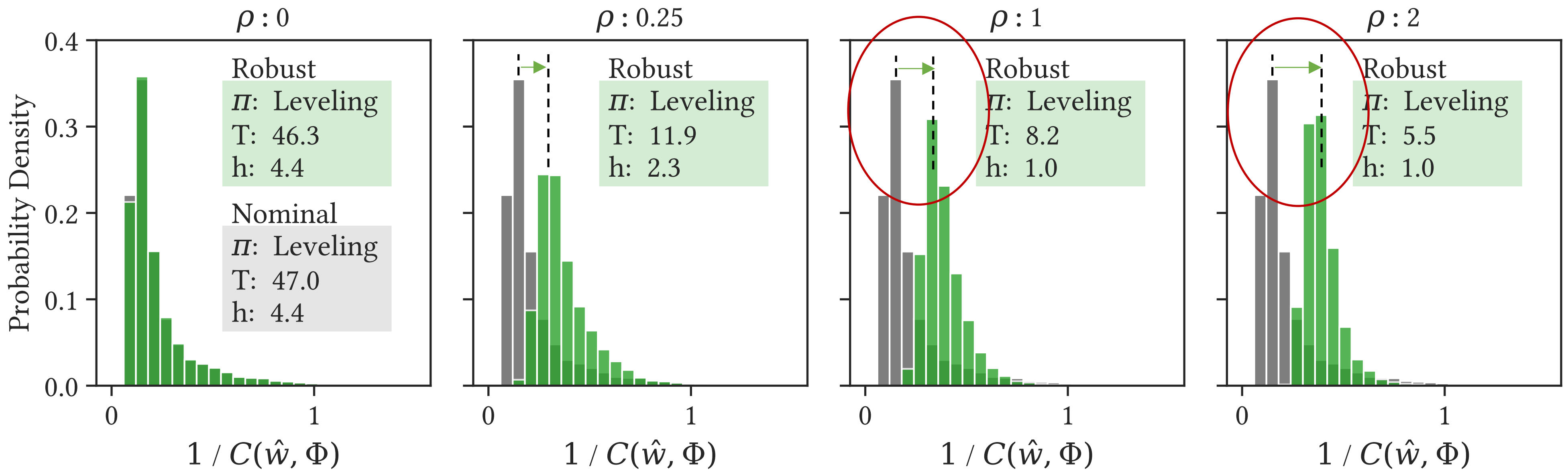


Expected ρ: given to tuner

Throughput

$w_{11} = (33\%\,, 33\%, 33\%, 1\%)$

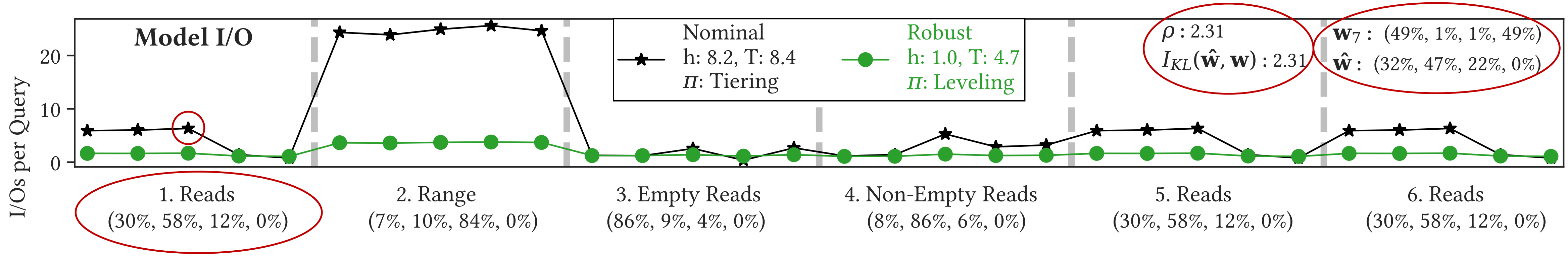
ρ and Performance Gain Distribution



$$w_{11} = (33\%\,, 33\%, 33\%, 1\%)$$

Peak of the distribution moves towards higher throughput as we consider higher uncertainty

Workload Sequence on RocksDB

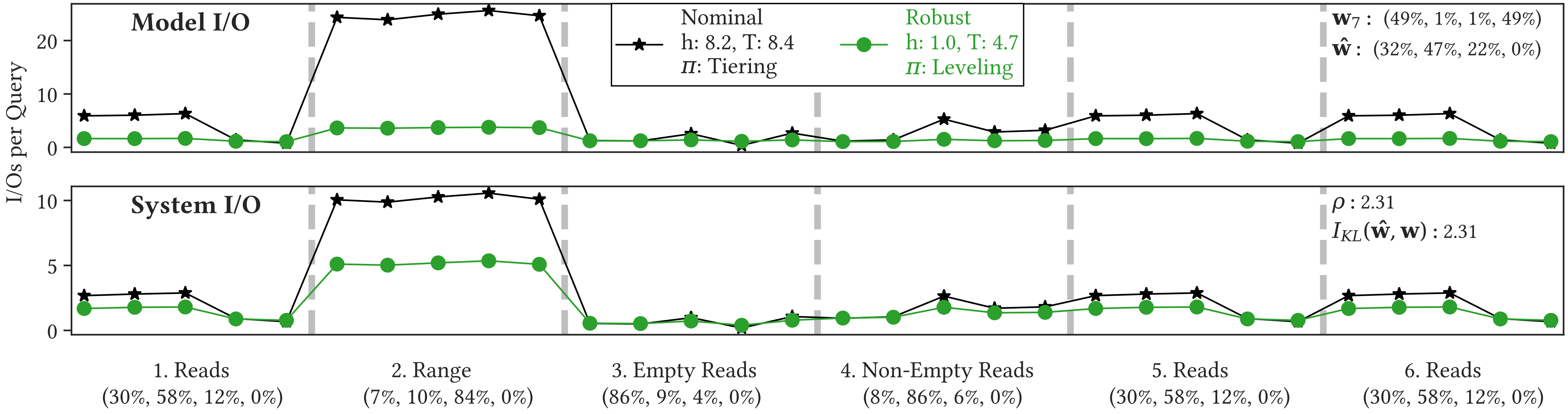


RocksDB instance setup with 10 million unique key-value pairs of size 1KB

Each observation period is 200K queries, with 5 observations per session 6 million queries to the DB

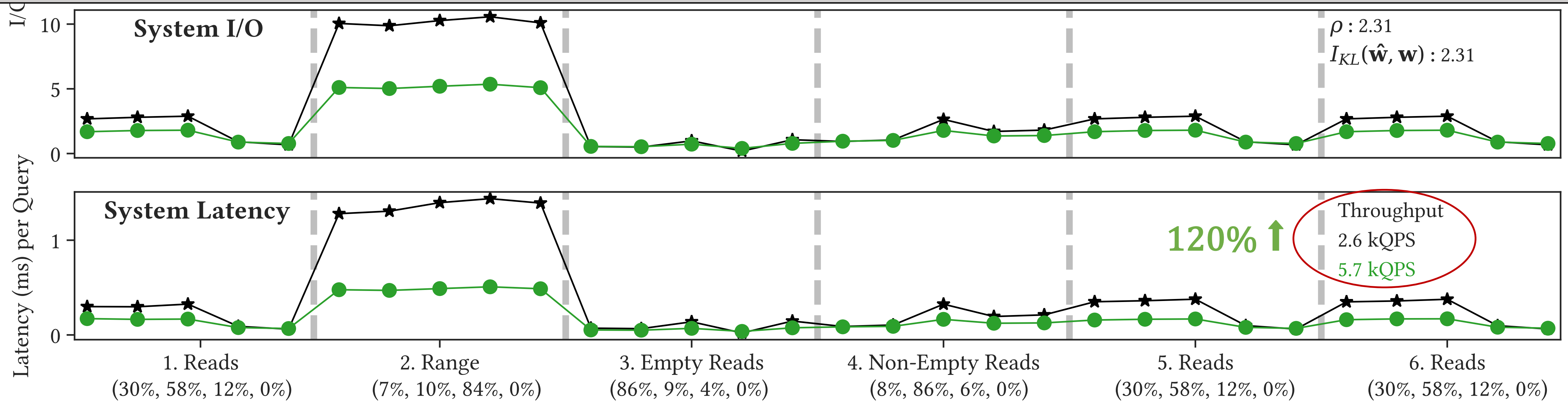
Writes are unique, range queries average 1-2 pages per level

Workload Sequence



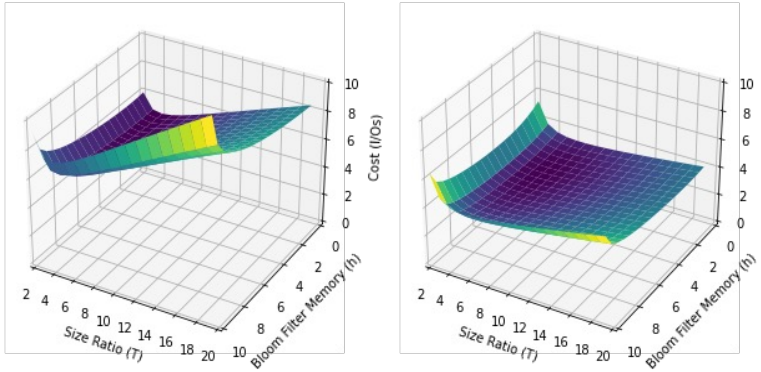
Workload Sequence

Small subset of results! Take a look at the paper for a more detailed analysis

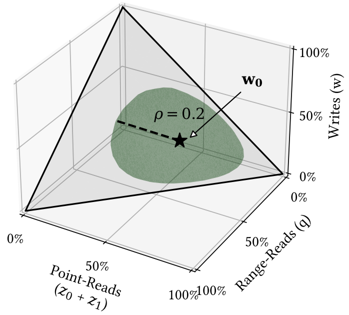


Thanks!

Workload uncertainty creates suboptimal tunings



ENDURE: robust tuning using neighborhood of workloads



Deployed ENDURE on RocksDB

disc.bu.edu/

www.ndhuynh.com/

@nd_huynh