DOGI:
Data Placement with Oracle-Guided Insights for Log-Structured Systems

★Jeeyun Kim[1], ★Seonggyun Oh[2], Jungwoo Kim[2],
Jisung Park[1], Jaeho Kim[3], Sungjin Lee[1], Sam H. Noh[4]

[1]POSTECH [2]DGIST [3]Gyeongsang National University [4]Virginia Tech

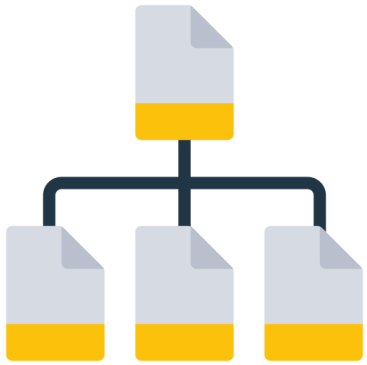
★ These authors equally contributed to this work

24th USENIX Conference on File and Storage Technologies

Log-Structured Systems

- **Widely deployed in various applications**
  - key-value stores, file systems, and storage devices
- **Achieve high write throughput through append-only design**

LSM-Tree based
KV stores

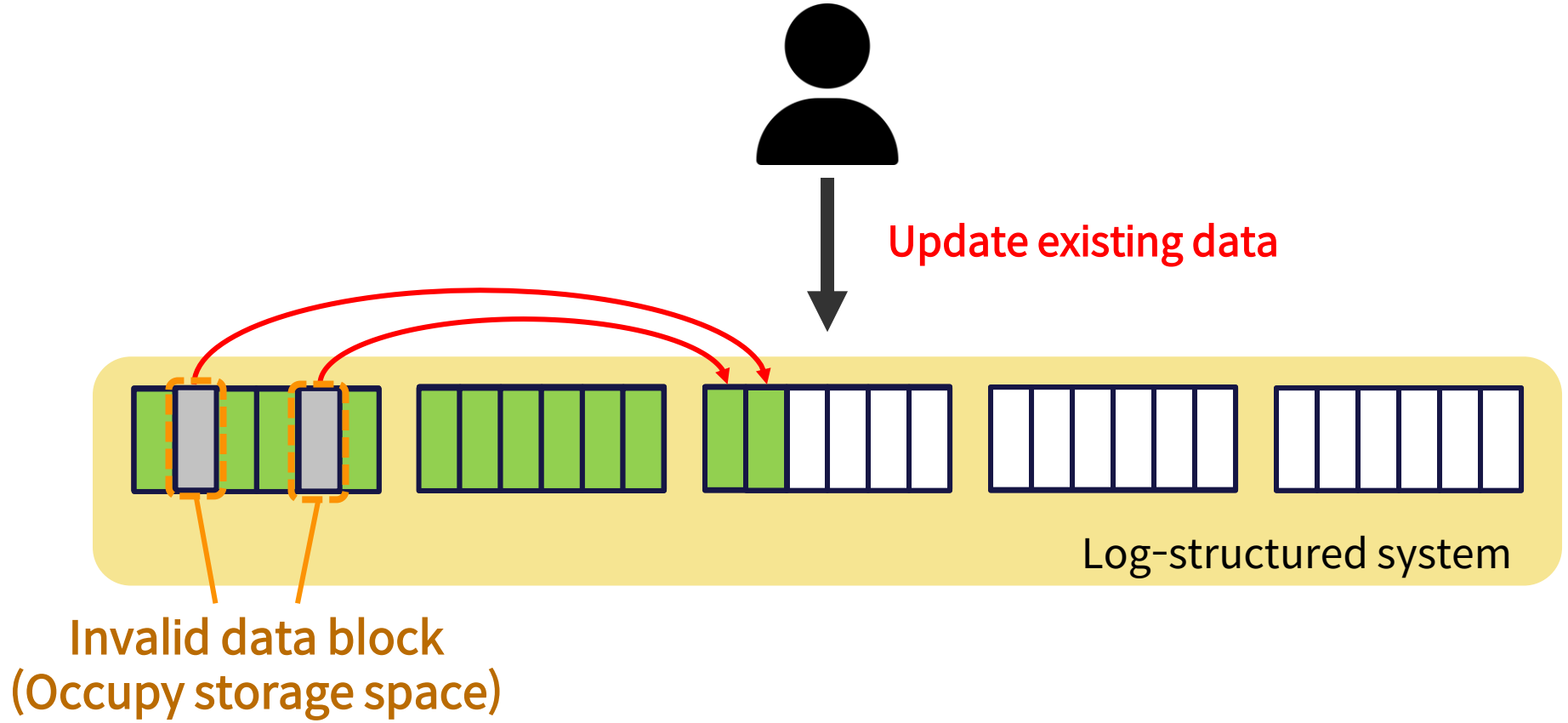


File systems
(F2FS, BtrFS)

Zoned storage
interface

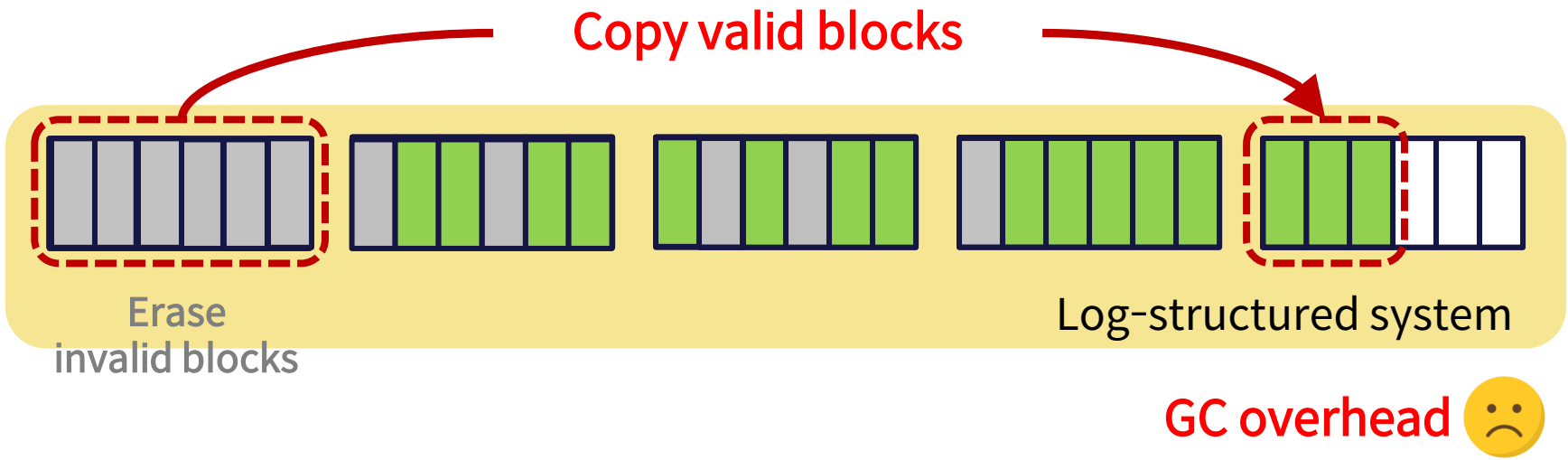
Problem: Invalid Data in Log-structured Systems

- **Append-only write design inevitably produces invalid data**
  - This invalid data unnecessarily occupies storage space



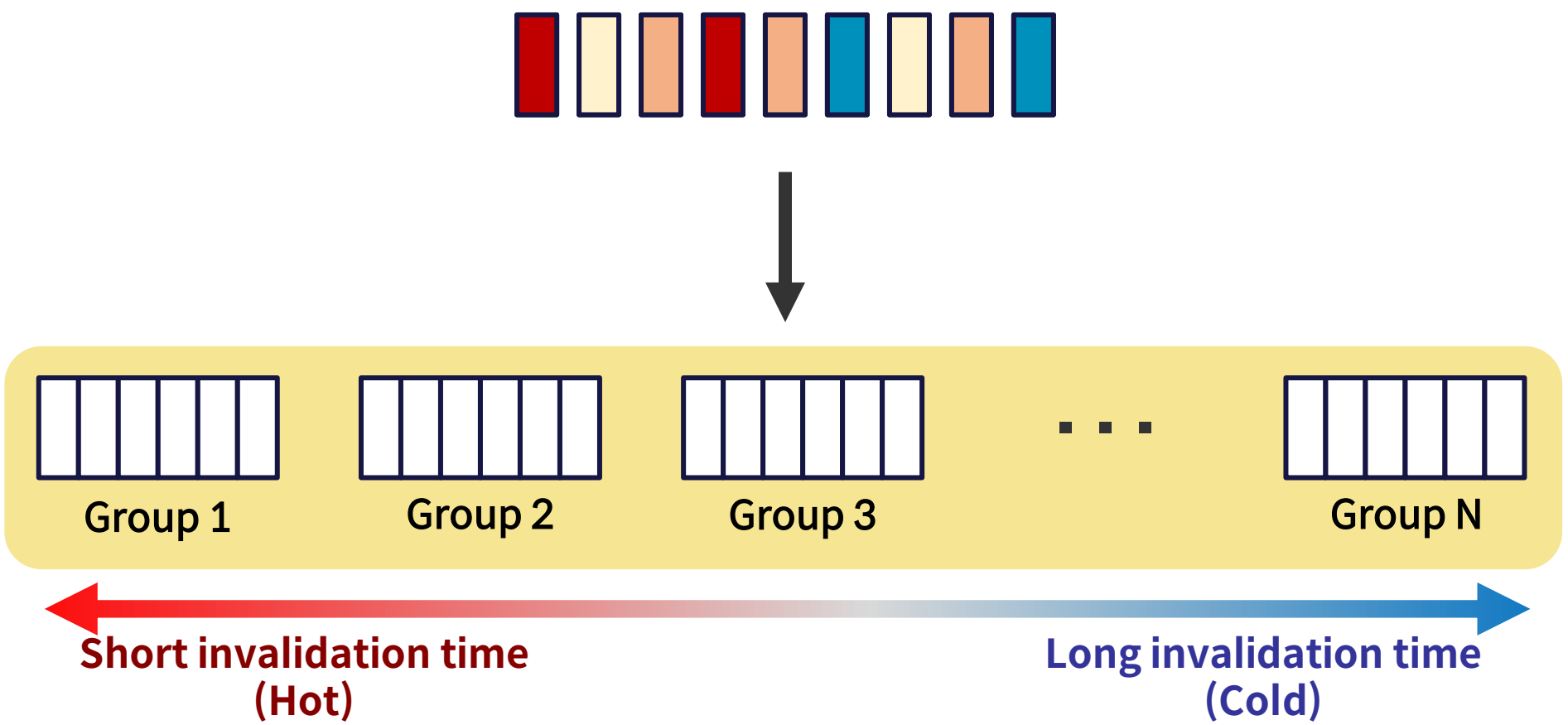
Problem: GC Overhead

- Garbage collection (GC): process of reclaiming invalid data
- **Write amplification factor (WAF)**: The factor that shows additional writes by GC
  - $WAF = \frac{\text{User writes + GC writes}}{\text{User writes}}$



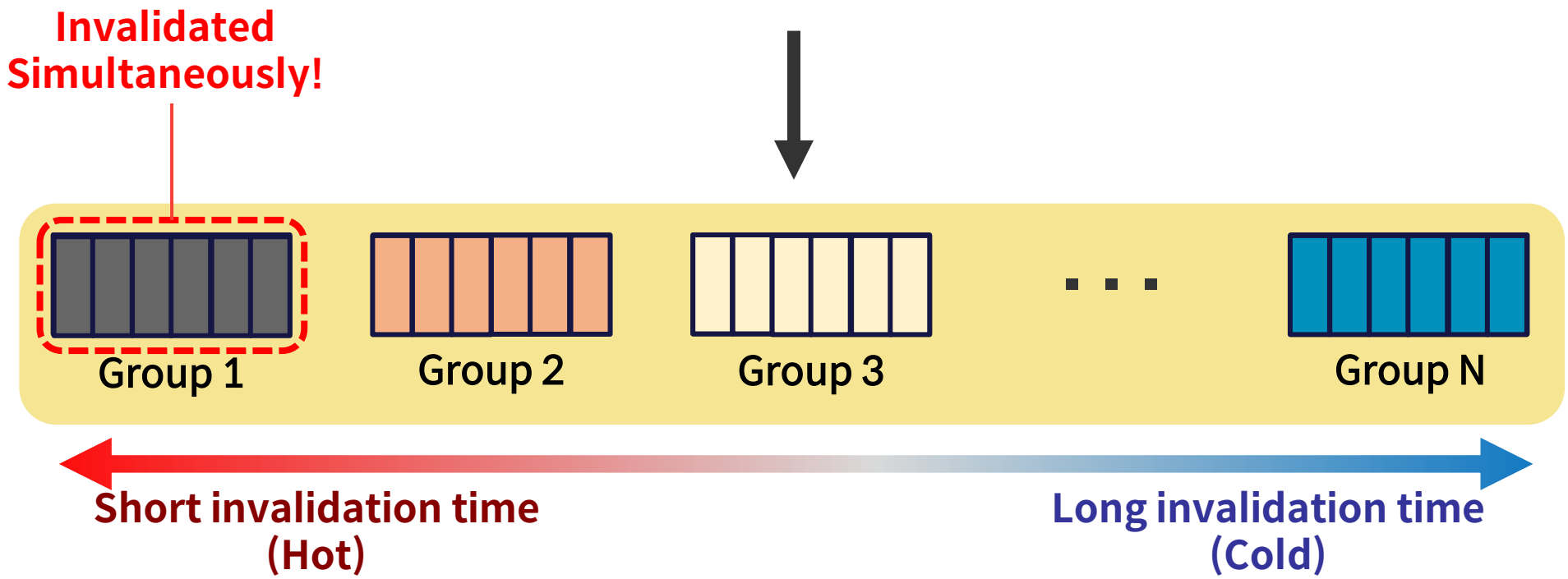
Approach for Reducing WAF: Data Placement

- Goal of data placement is to group data blocks with **similar invalidation time**



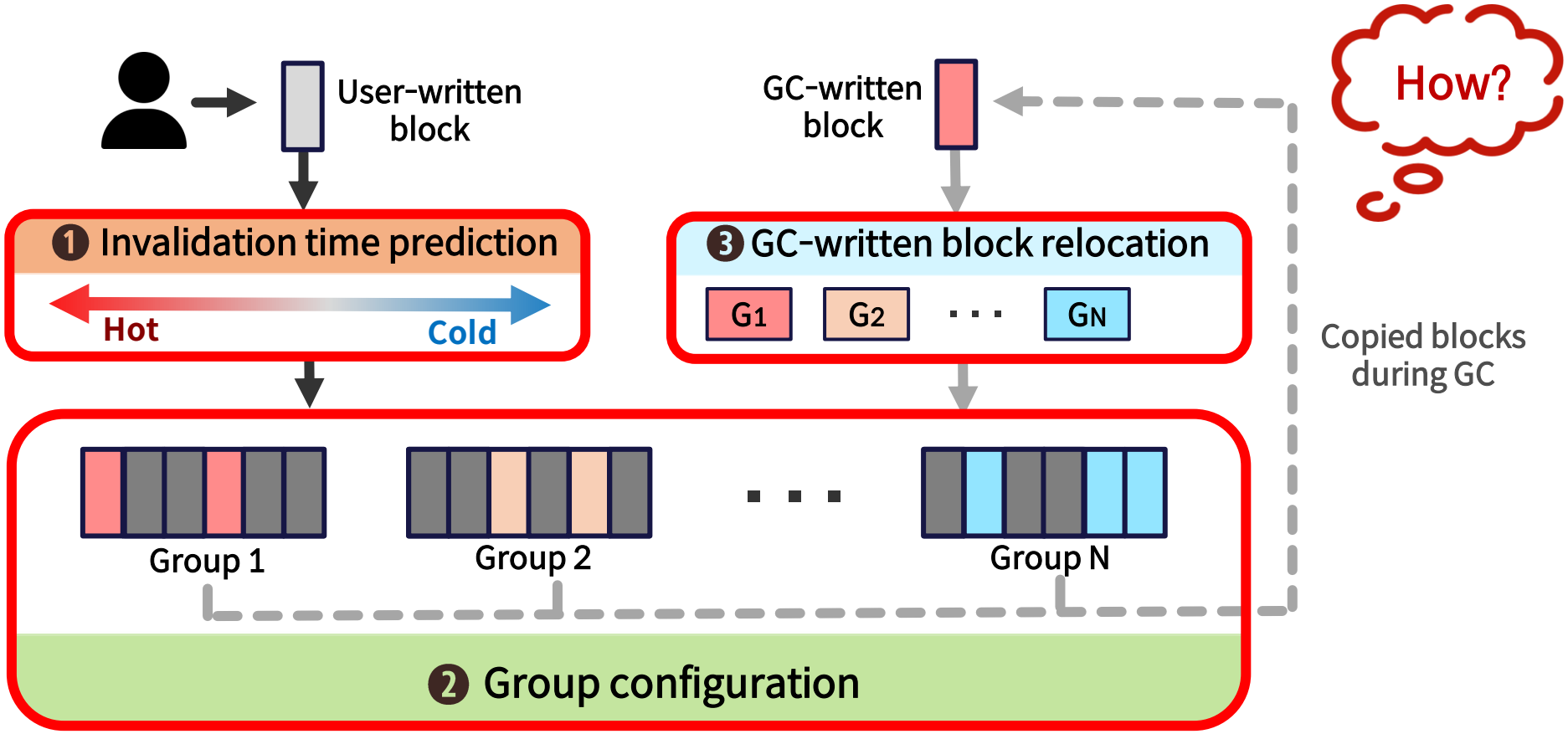
Approach for Reducing WAF: Data Placement (cont')

- Goal of data placement is to group data blocks with **similar invalidation time**



Existing Data Placement Techniques

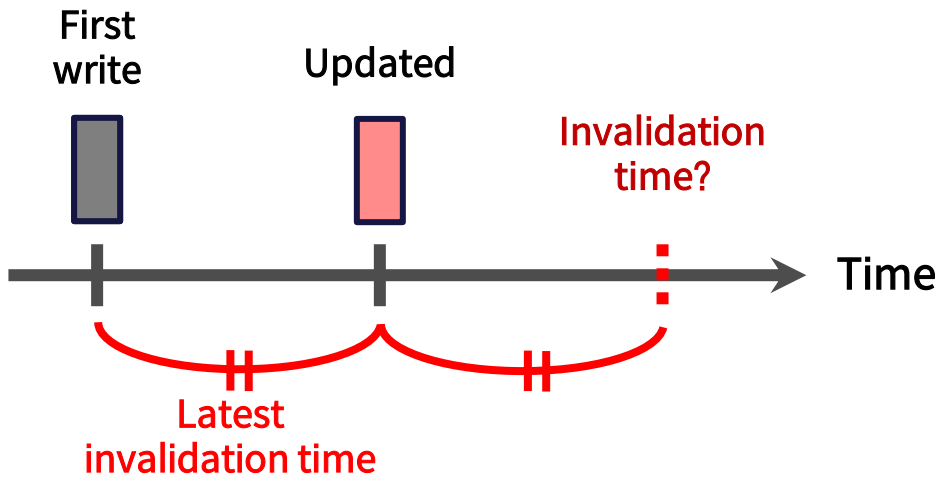
- Various data placement techniques have been proposed to reduce WAF
- We select four existing data placement techniques and analyze how they work
  - MiDAS [FAST'24], SepBIT [FAST'22], PHFTL [DAC'23], ML-DT [SYSTOR'21]



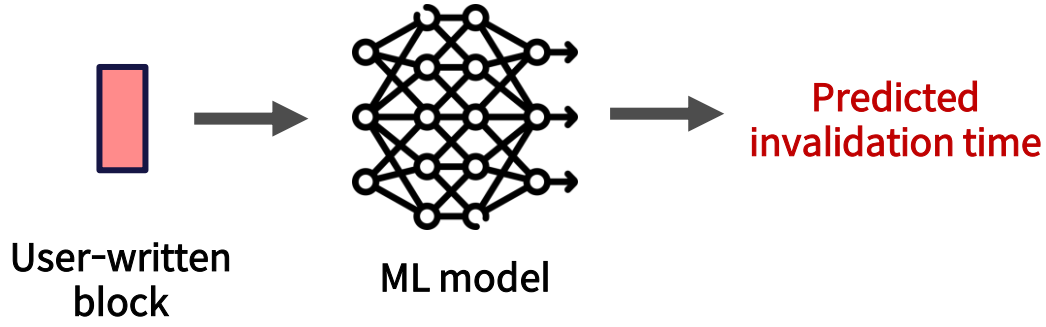
Existing Data Placement Techniques (cont')

Step 1: How to predict the invalidation time of user-written blocks

① Using block's latest invalidation time



② Using ML model

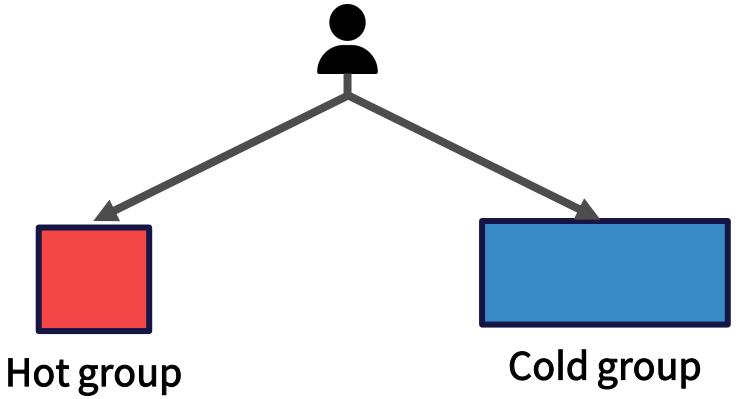


	MiDAS	SepBIT	PHFTL	ML-DT
Invalidation time prediction (user-written block)	Latest invalidation time	Latest invalidation time	ML model	ML model

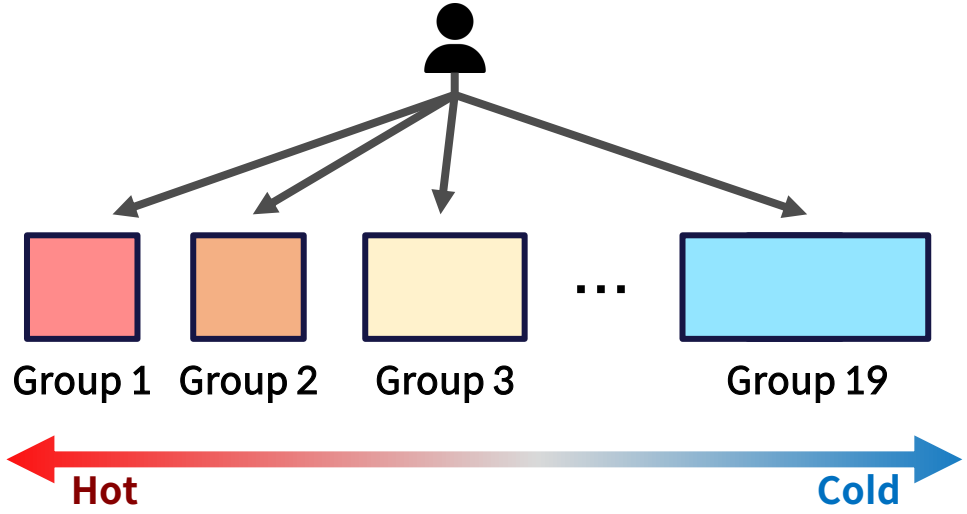
Existing Data Placement Techniques (cont')

Step 2: How to group user-written blocks based on invalidation time

① Separate into hot and cold groups



② Separate into 19 groups

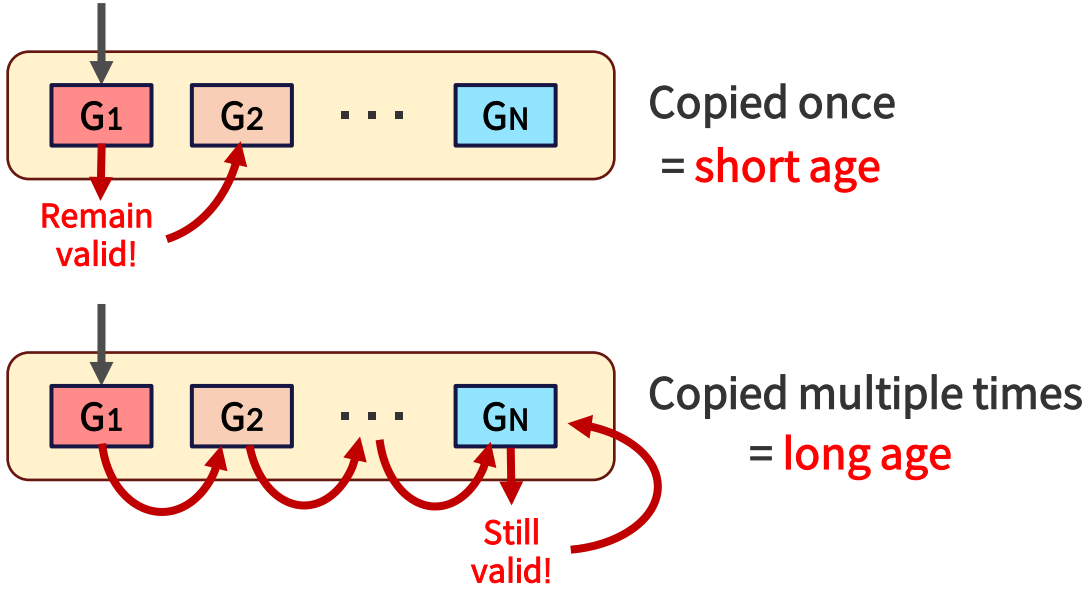


	MiDAS	SepBIT	PHFTL	ML-DT
Group configuration (for user-written block)	2	2	2	19

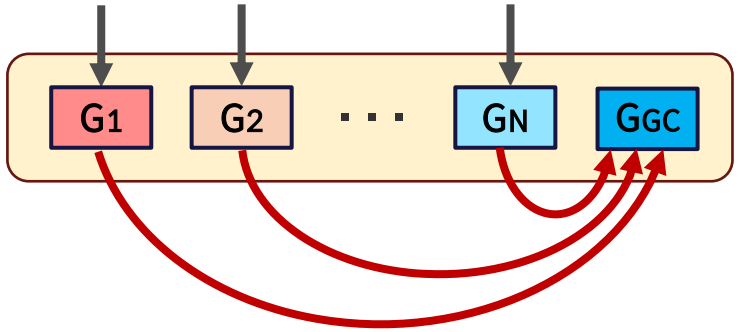
Existing Data Placement Techniques (cont')

Step 3: How to relocate GC-written blocks

① Relocate blocks by their age



② Isolated into a separate group



	MiDAS	SepBIT	PHFTL	ML-DT
Invalidation time prediction (GC-written block)	Age	Age	Age	X

Existing Data Placement Techniques (cont')

- **In summary, techniques differ in their strategies across the three steps**
  - MiDAS [FAST'24], SepBIT [FAST'22], PHFTL [DAC'23], ML-DT [SYSTOR'21]

	MiDAS	SepBIT	PHFTL	ML-DT
User-written block placement	Latest invalidation time	Latest invalidation time	ML model	ML model
Group configuration (for user-written block)	2	2	2	19
GC-written block relocation	Age	Age	Age	X

How close are these techniques to optimal data placement?

Near-optimal Data Placement (NoDaP)

- **We design Near-optimal data placement (NoDaP) as practical reference baseline**
  - Achieve near-optimal performance in all three steps

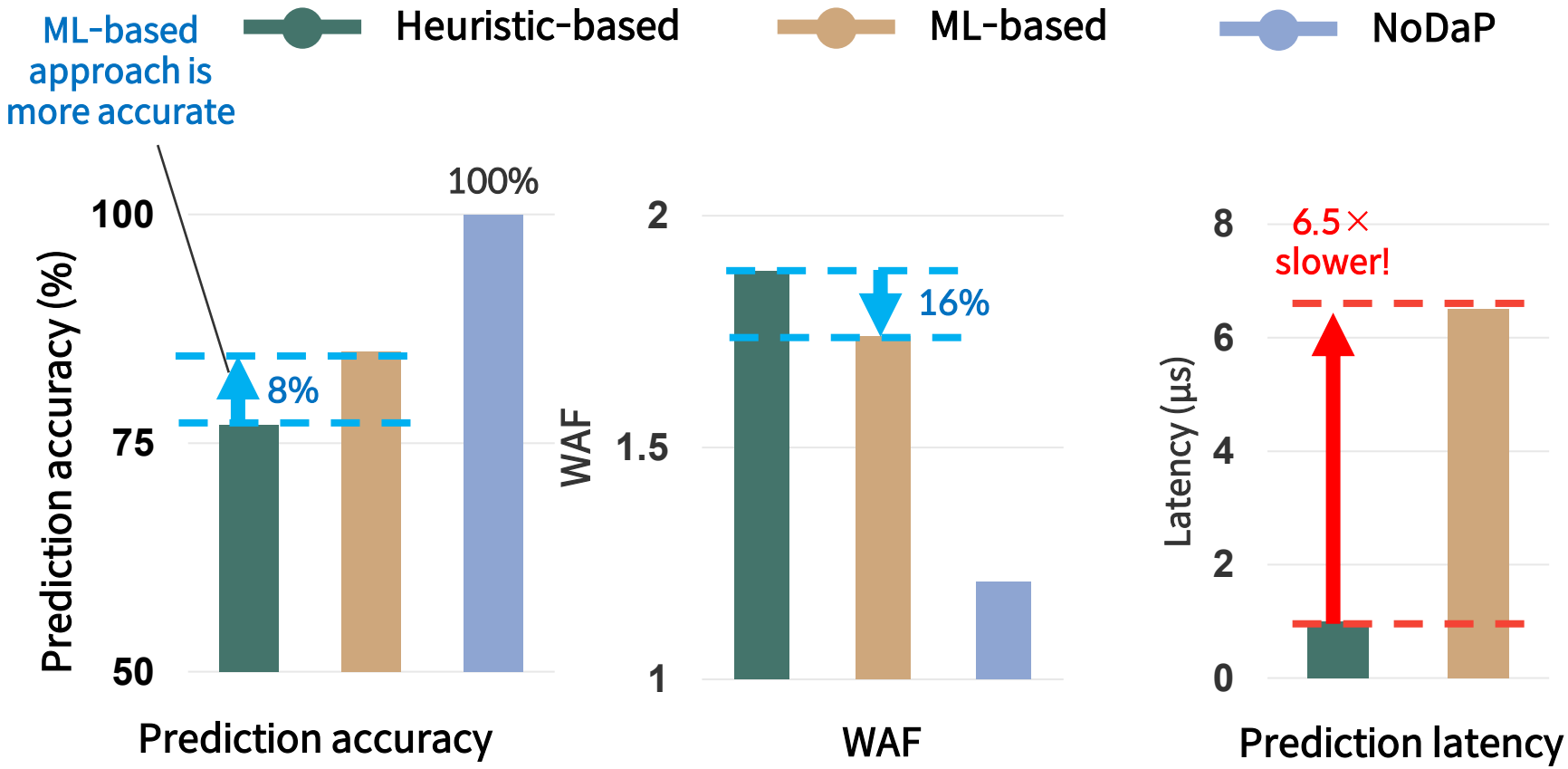
Checklist

- ☐ *Predict user-written block's invalidation time accurately?*
- ☐ *Choose appropriate group configuration?*
- ☐ *Relocate GC-written block appropriately?*

	MiDAS	SepBIT	PHFTL	ML-DT	NoDaP
User-written block placement	Latest invalidation time	Latest invalidation time	ML model	ML model	Exact knowledge
Group configuration (for user-written block)	2	2	2	19	N (varies across workloads)
GC-written block relocation	Age	Age	Age	X	Exact knowledge

Problem 1: Inaccurate Estimation

- **SOTA techniques estimate block invalidation time inaccurately**
  - **Heuristic-based** (i.e., SepBIT and MiDAS) and **ML-based** (i.e., PHFTL and ML-DT)

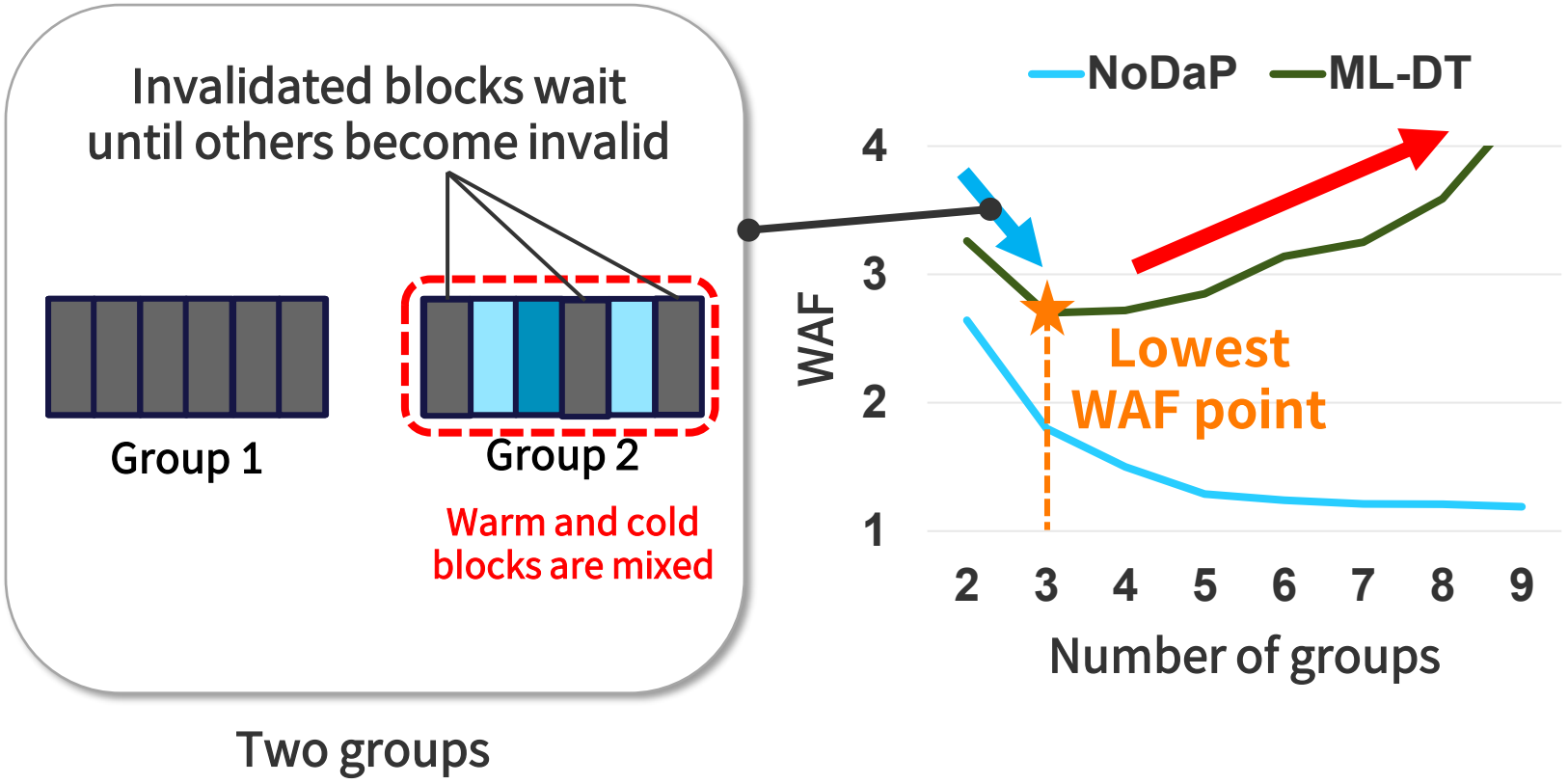


Checklist

- ☒ *Predict user-written block's invalidation time accurately?*
- △ *Predict invalidation time with low latency?*
- ☐ *Choose appropriate group configuration?*
- ☐ *Relocate GC-written block appropriately?*

Problem 2: Lack Consideration of Group Configuration

- **SOTA techniques fail to set appropriate number of groups for user-written block**
  - 2 groups (i.e., MiDAS, SepBIT, and PHFTL), and 19 groups (i.e., ML-DT)

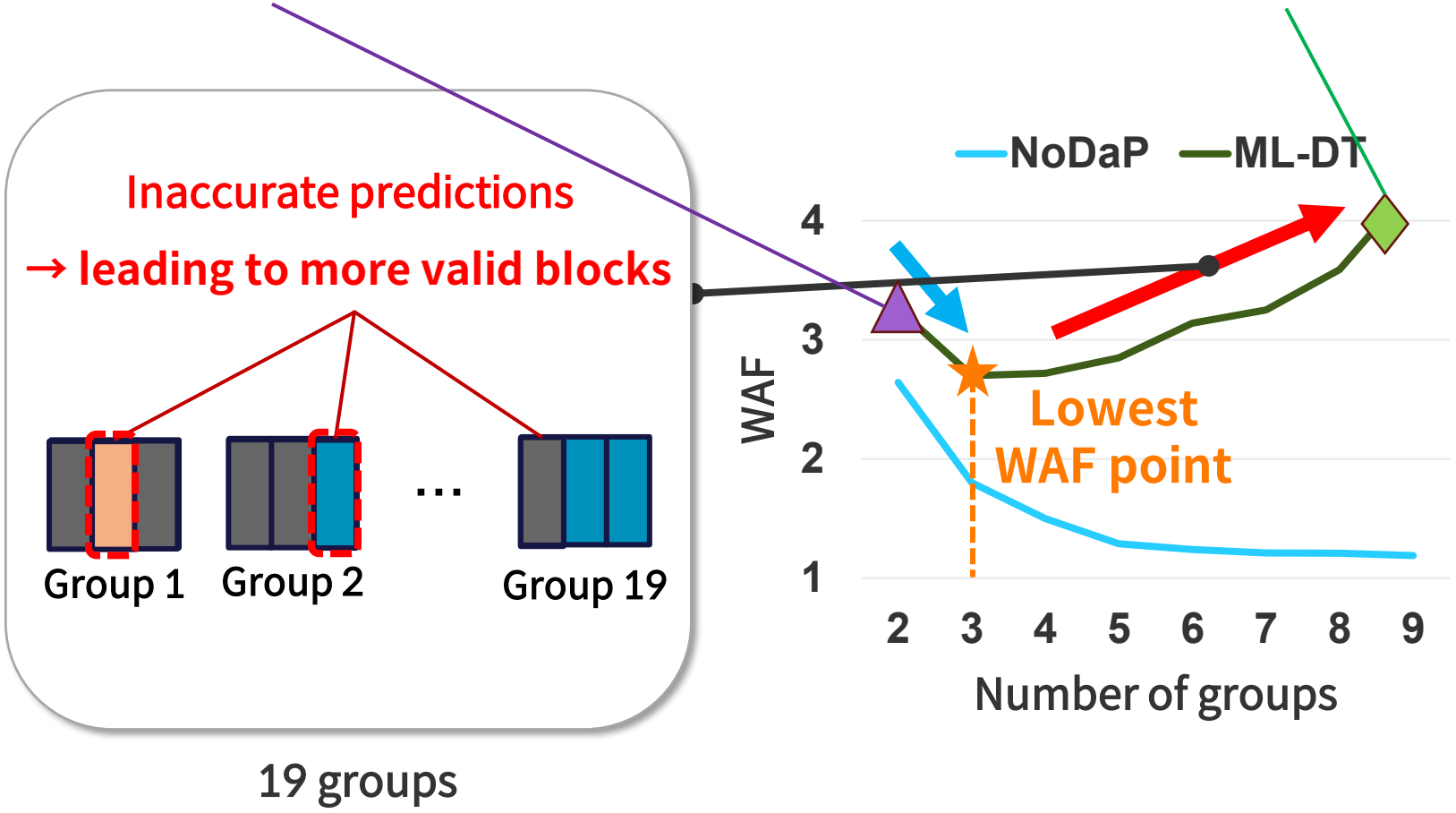


Checklist

- ☒ *Predict user-written block's invalidation time accurately?*
- △ *Predict invalidation time with low latency?*
- ☐ *Choose appropriate group configuration?*
- ☐ *Relocate GC-written block appropriately?*

Problem 2: Lack Consideration of Group Configuration

- **SOTA techniques fail to set appropriate number of groups for user-written block**
  - 2 groups (i.e., MiDAS, SepBIT, and PHFTL), and 19 groups (i.e., ML-DT)

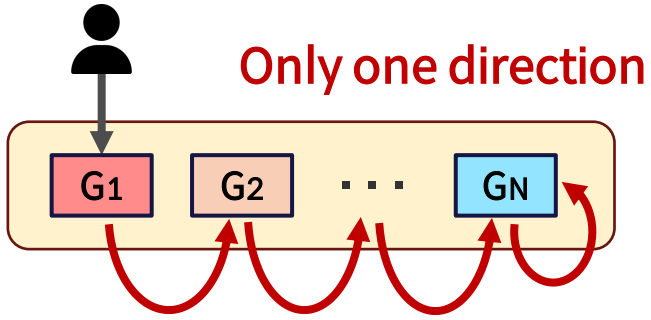


Checklist

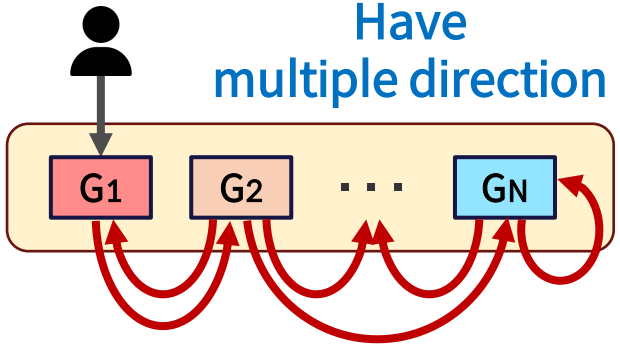
- ☒ *Predict user-written block's invalidation time accurately?*
- △ *Predict invalidation time with low latency?*
- ☒ *Choose appropriate group configuration?*
- ☐ *Relocate GC-written block appropriately?*

Problem 3: Inappropriate GC-written Block Relocation

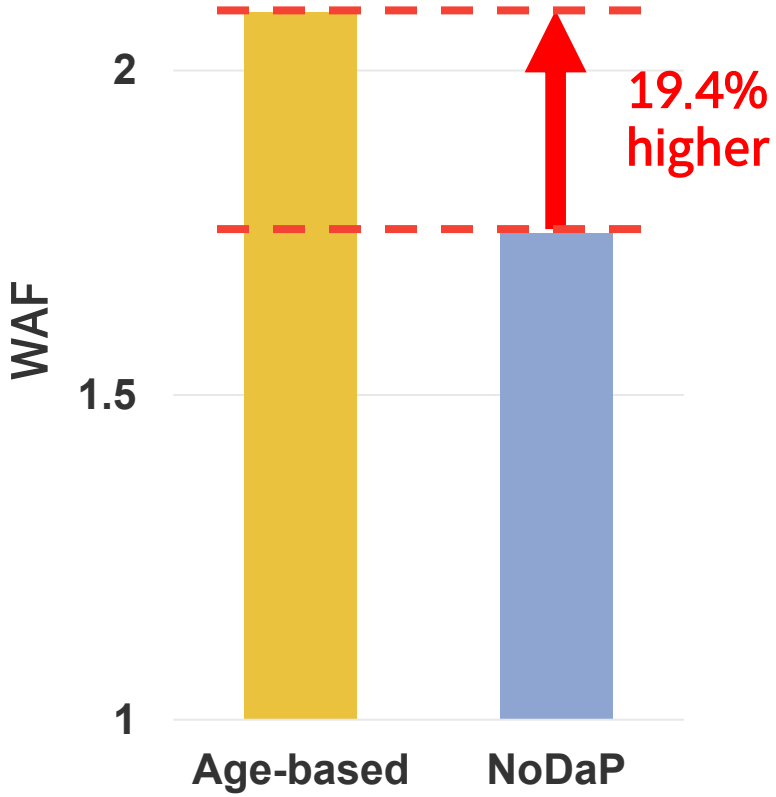
- Age-based relocation of GC-written blocks is insufficient



Age-based relocation



NoDaP's relocation

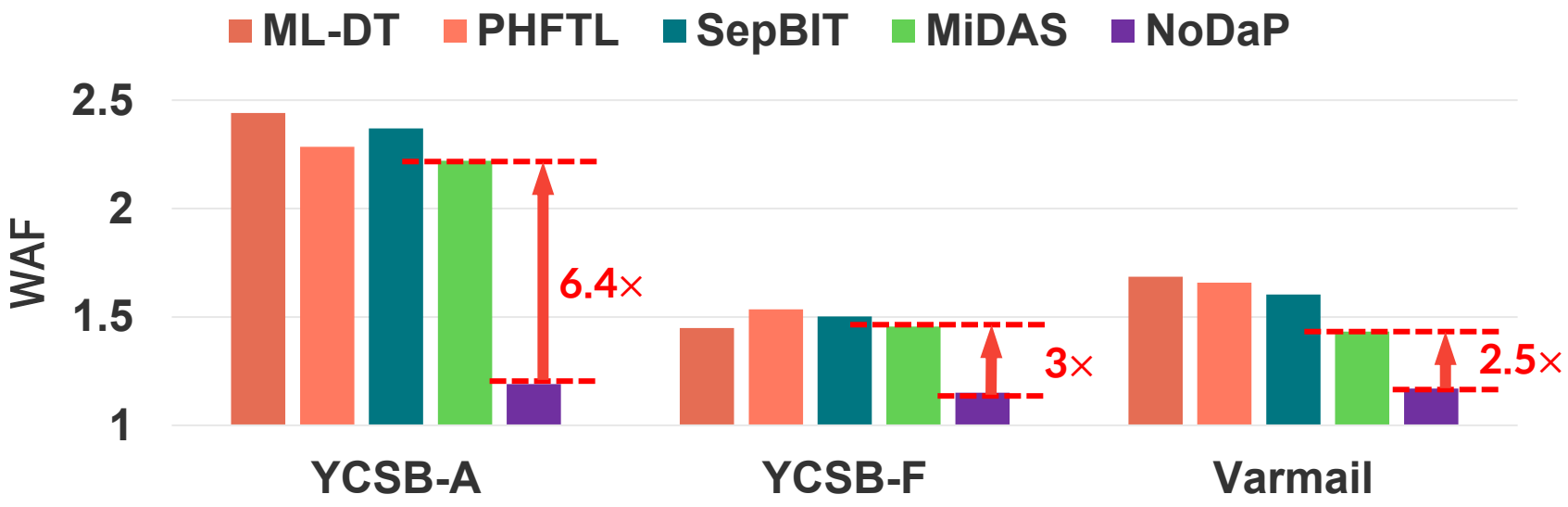


Checklist

- ✗ *Predict user-written block's invalidation time accurately?*
- △ *Predict invalidation time with low latency?*
- ✗ *Choose appropriate group configuration?*
- ✗ *Relocate GC-written block appropriately?*

NoDaP vs SOTA Techniques

- **Large gap** between NoDaP and existing techniques
  - SOTA: MiDAS [FAST'24]



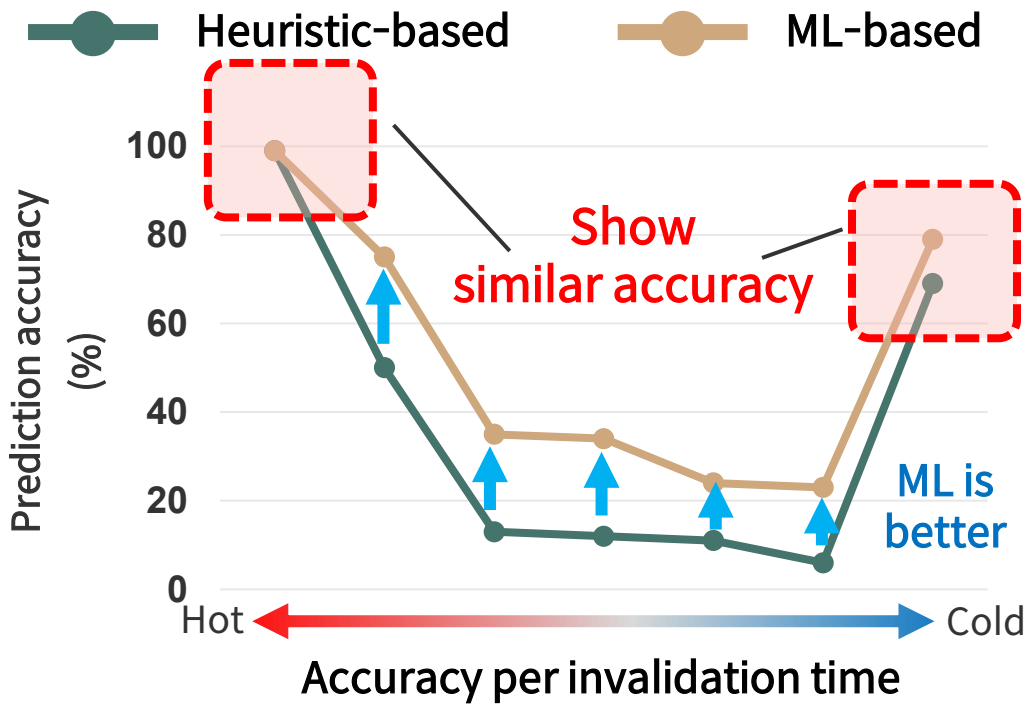
Checklist

- ✗ *Predict user-written block's invalidation time accurately?*
- △ *Predict invalidation time with low latency?*
- ✗ *Choose appropriate group configuration?*
- ✗ *Relocate GC-written block appropriately?*

Design Goal of DOGI

① User-written block placement

- Accurate and low-latency invalidation-time prediction
  - ➡ Use heuristic-based prediction for the hottest and coldest blocks, and use ML model for the remaining blocks



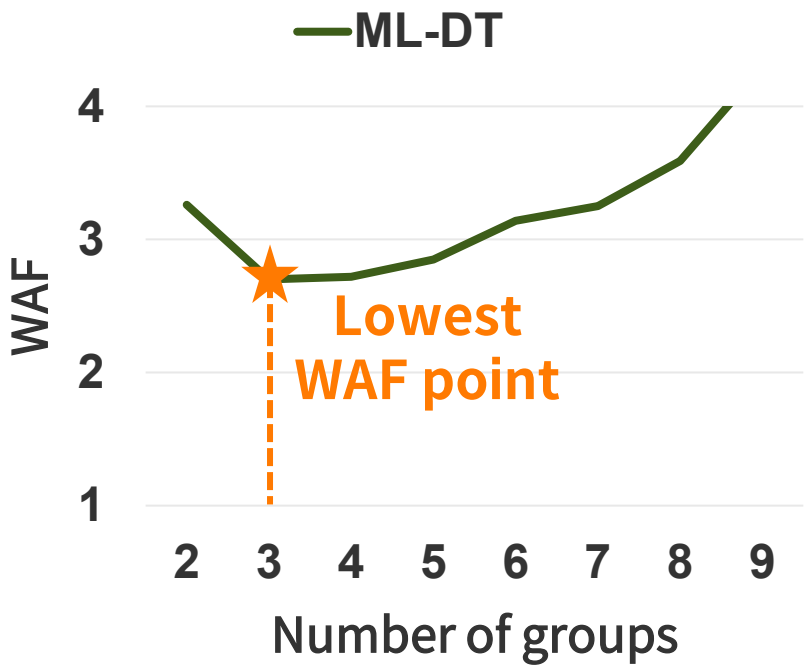
Checklist

- ✓ *Predict user-written block's invalidation time accurately?*
- ✓ *Predict invalidation time with low latency?*
- ☐ *Choose appropriate group configuration?*
- ☐ *Relocate GC-written block appropriately?*

Design Goal of DOGI

② Group configuration selection

- Applying a WAF-minimizing group configuration
  - ➡ Estimate WAF for candidate group configurations
  - ➡ Apply group configuration that minimizes WAF



Checklist

- ✓ *Predict user-written block's invalidation time accurately?*
- ✓ *Predict invalidation time with low latency?*
- ✓ *Choose appropriate group configuration?*
- ☐ *Relocate GC-written block appropriately?*

Design Goal of DOGI

② Group configuration selection

- Applying a WAF-minimizing group configuration
  - ➡ Estimate WAF for candidate group configurations
  - ➡ Apply group configuration that minimizes WAF

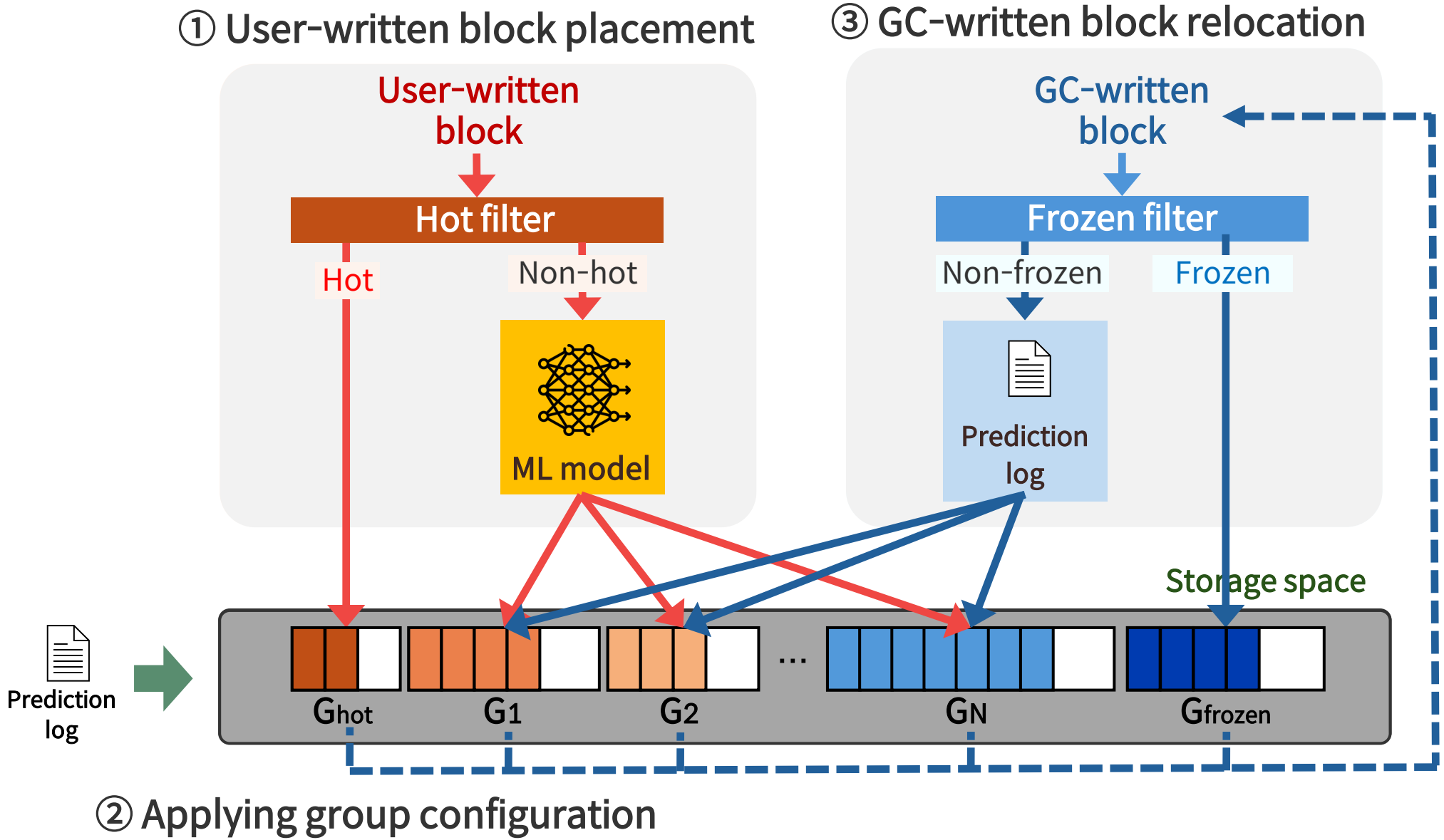
③ GC-written block relocation

- More effective GC-written-block relocation
  - ➡ Relocate GC-written blocks using historical data

Checklist

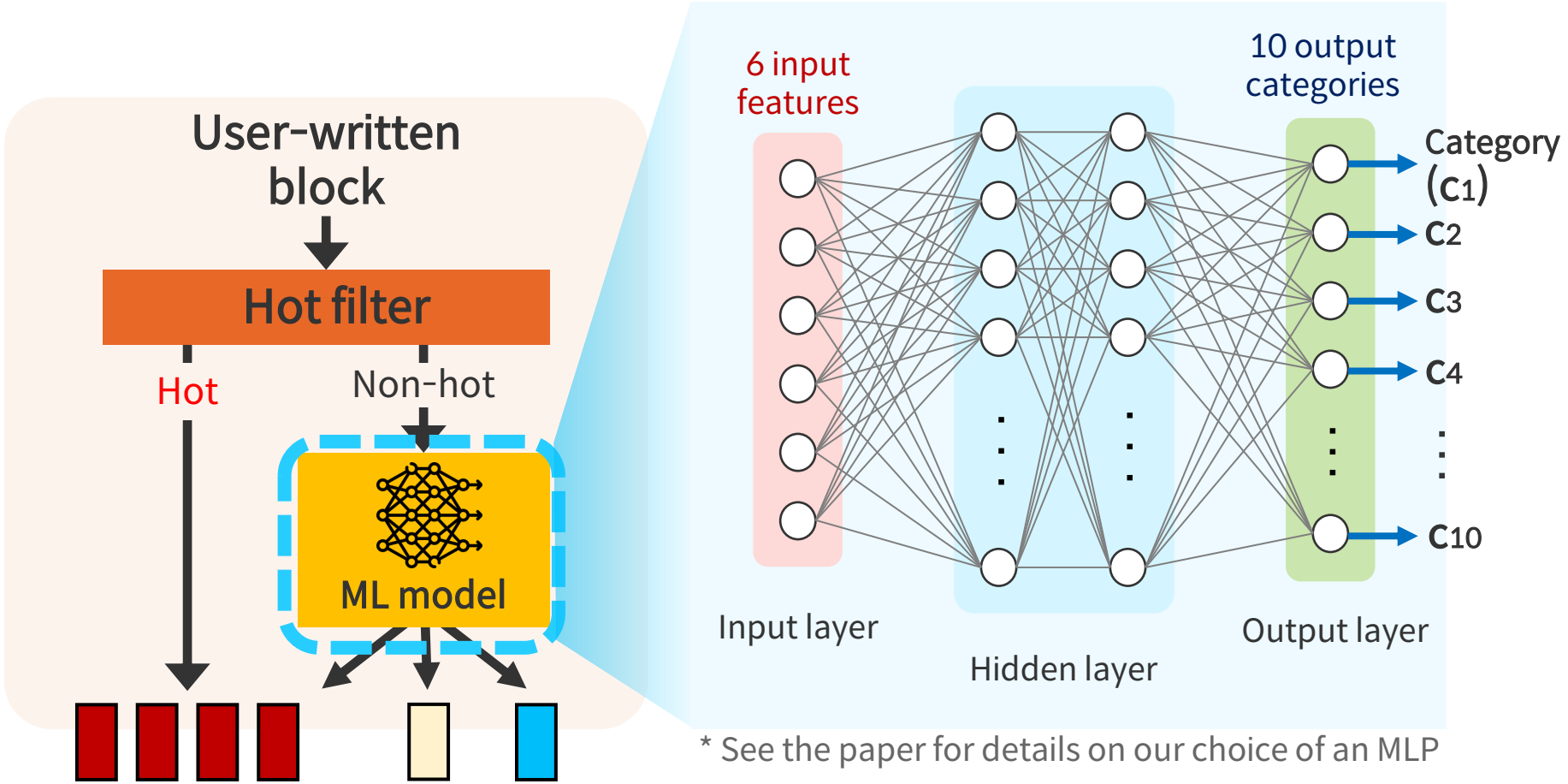
- ✓ *Predict user-written block's invalidation time accurately?*
- ✓ *Predict invalidation time with low latency?*
- ✓ *Choose appropriate group configuration?*
- ✓ *Relocate GC-written block appropriately?*

DOGI Overview



Design 1: User-written Block Placement

- **Accurate and lightweight prediction of user-written block invalidation time**
  - Use **Multi-Layer Perceptron (MLP) model** with six input features and ten output categories



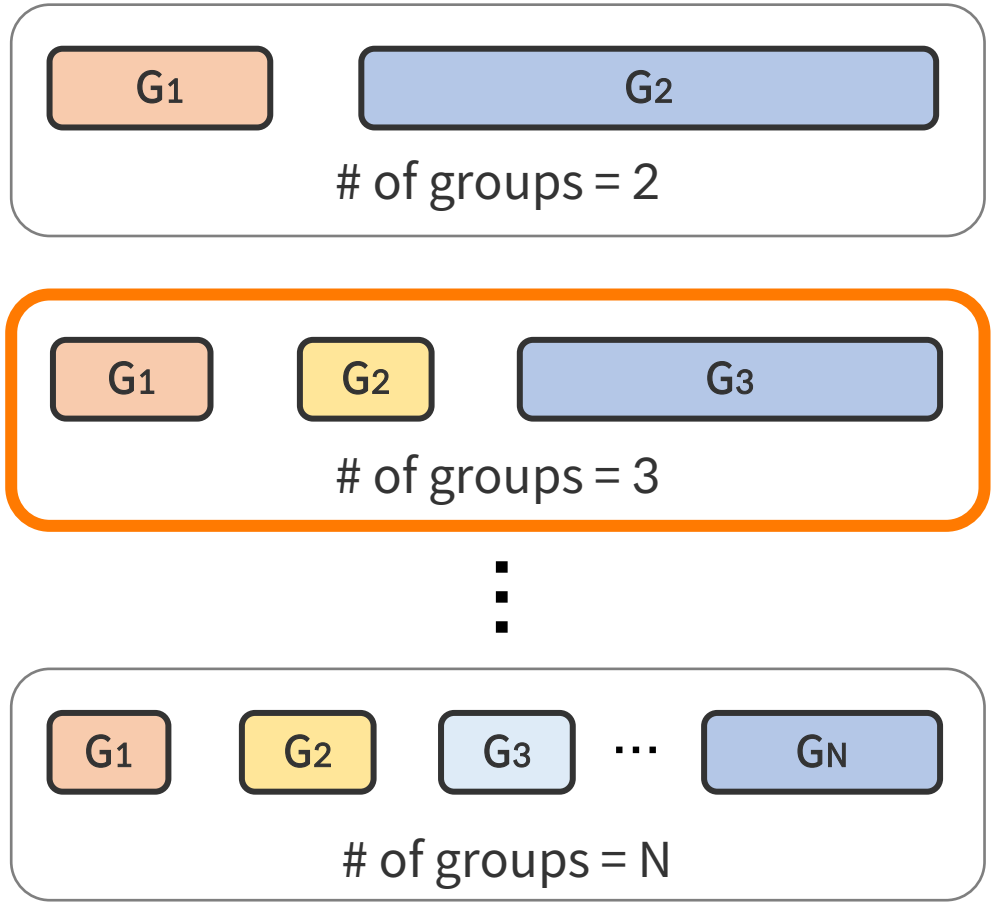
* See the paper for details on our choice of an MLP

Checklist

- ✓ *Predict user-written block's invalidation time accurately?*
- ✓ *Predict invalidation time with low latency?*
- ☐ *Choose appropriate group configuration?*
- ☐ *Relocate GC-written block appropriately?*

Design 2: Applying Group Configuration

- **Assign user-written blocks under WAF-minimizing group configuration**

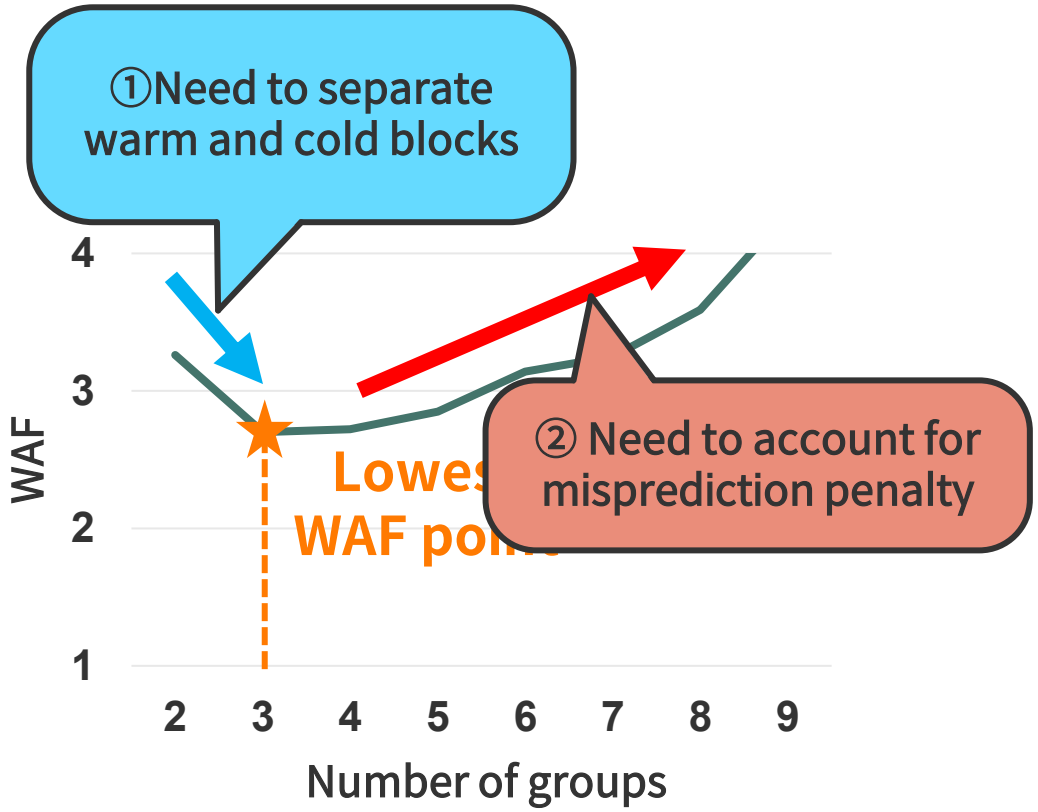


= 3.1

= 2.8

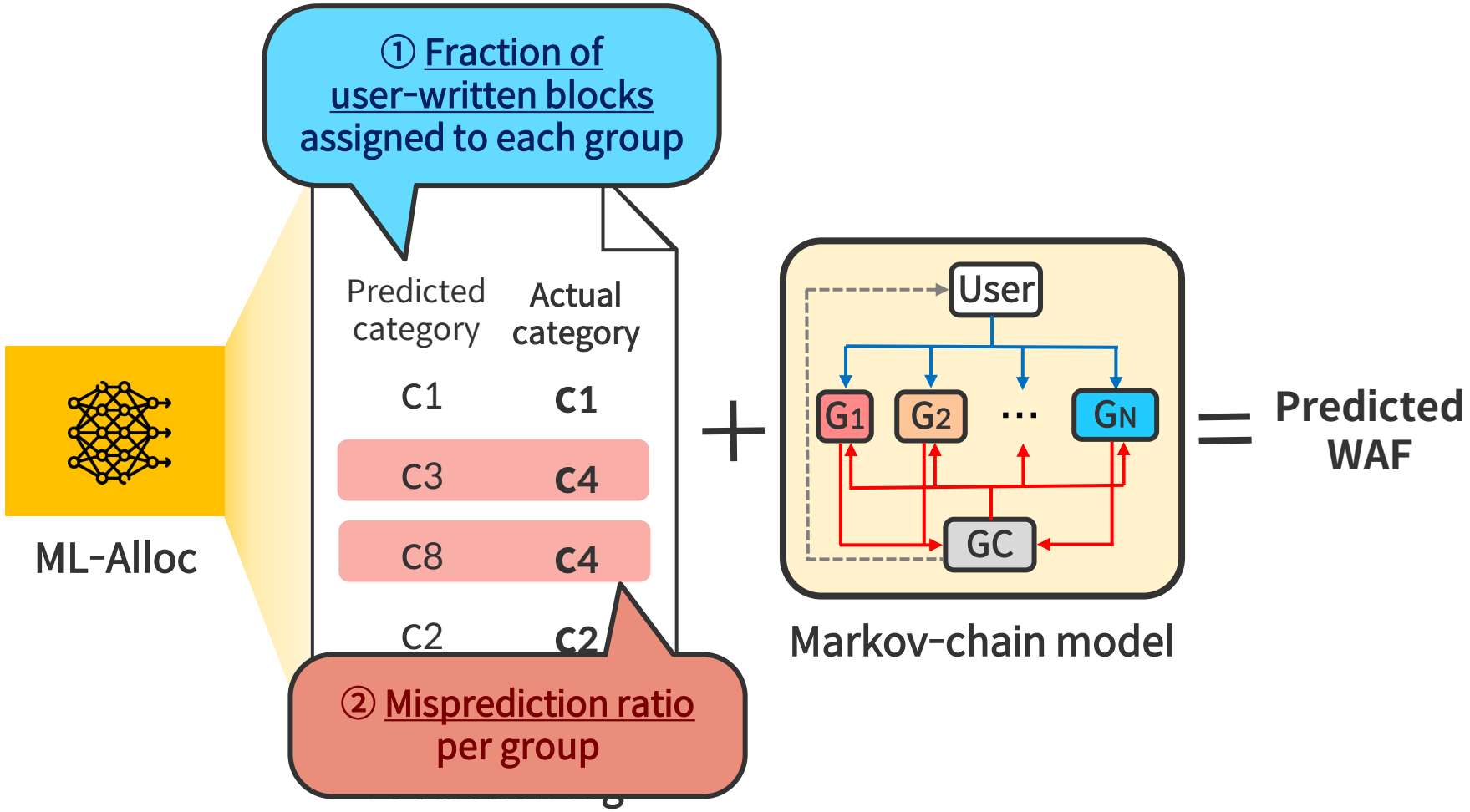
= 3.9

WAF



Design 2: Applying Group Configuration (cont')

- **Predict WAF for candidate group configurations**
  - Using **ML prediction log** and **Markov-chain model**

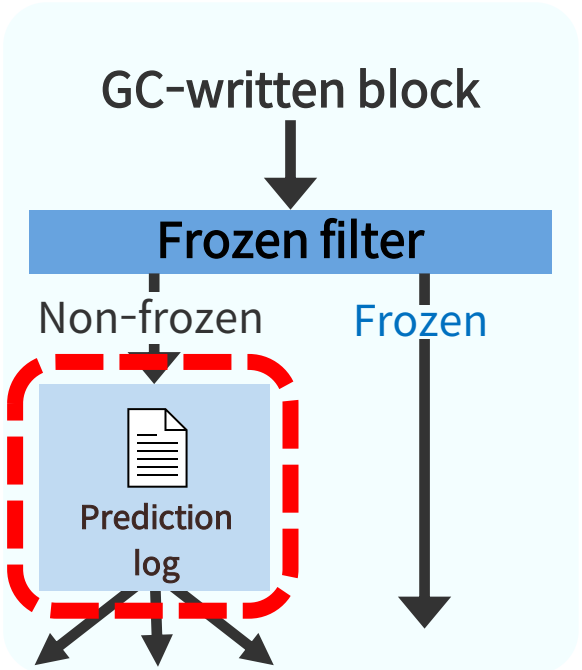


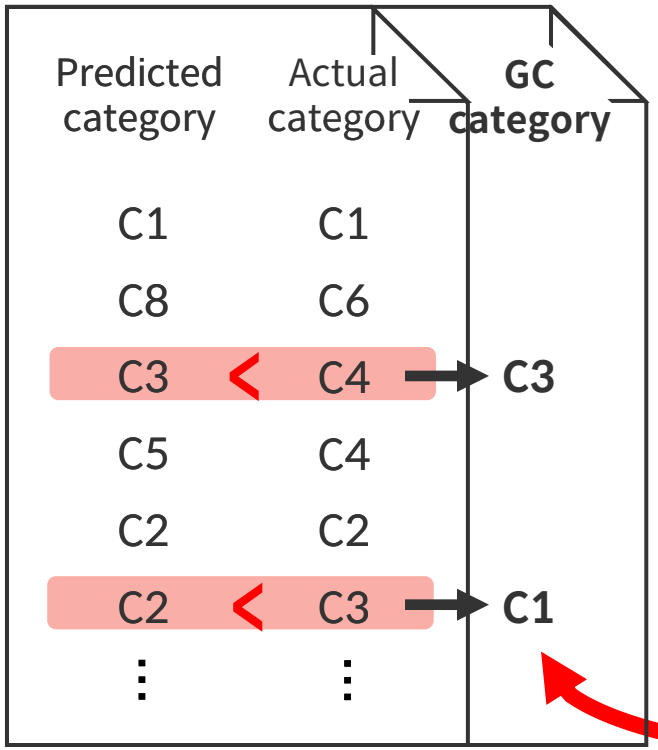
Checklist

- ☑ *Predict user-written block's invalidation time accurately?*
- ☑ *Predict invalidation time with low latency?*
- ☑ *Choose appropriate group configuration?*
- ☐ *Relocate GC-written block appropriately?*

Design 3: GC-written Block Relocation

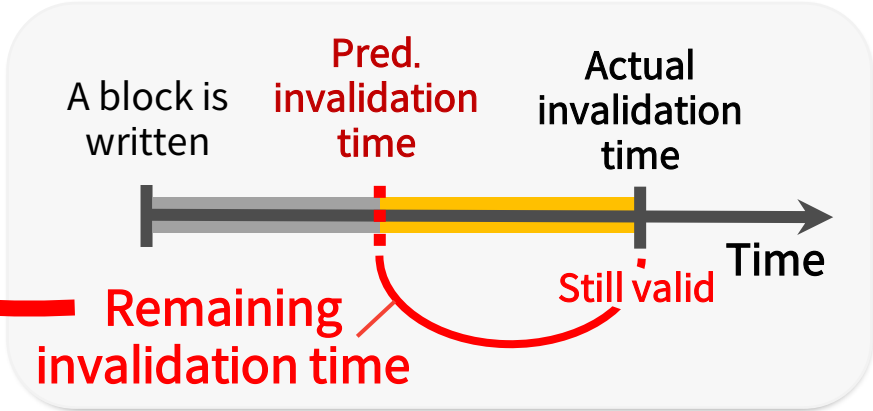
- **Relocate non-frozen blocks using historical data**
  - Using information of **mispredicted blocks in Plog**





Checklist

- ✓ *Predict user-written block's invalidation time accurately?*
- ✓ *Predict invalidation time with low latency?*
- ✓ *Choose appropriate group configuration?*
- ✓ *Relocate GC-written block appropriately?*



Find Out More in Our Paper!

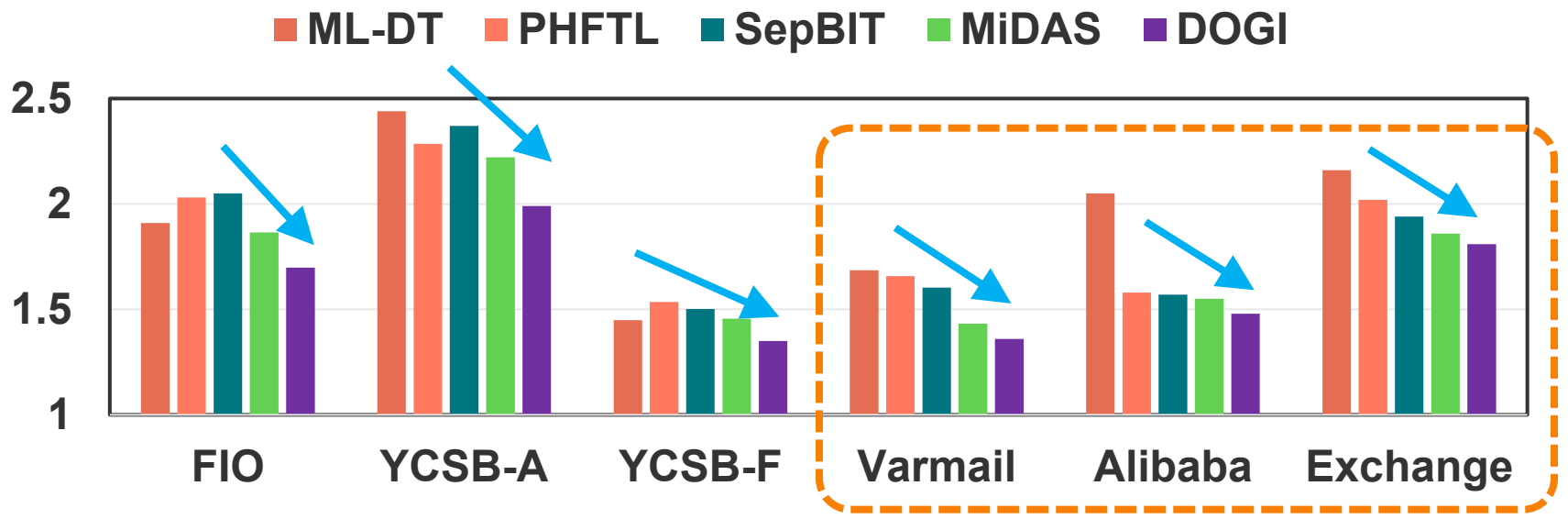
- **DOGI design details**
  - In-depth explanation of the ML model
  - Detailed description of Prediction log (PLog) generation
- **DOGI adaptability**
  - Periodic retraining of the ML model
  - Handling when workload I/O patterns change rapidly
- **Discussion**
  - Memory and computational overheads
  - Scalability considerations

Experimental Setup

- **Implement DOGI prototype on zoned storage**
  - Using 2TB Western Digital ZN540 SSD backend via ZenFS
  - Mapping each segment one-to-one to a ZoneFile
- **Benchmark**
  - **FIO** with Zipf distribution ($\theta = 1.0$)
  - **Varmail** of Filebench
  - **YCSB-A** and **–F** on MySQL
  - **Alibaba Cloud trace** and **Exchange** of Microsoft Enterprise trace
- **SOTA data placement techniques**
  - **MiDAS, SepBIT, PHFTL, and ML-DT**

WAF Analysis

- **15.5% WAF reduction compared to best-performing baseline (i.e., MiDAS)**
  - 25.1% average WAF reduction compared to other SOTA techniques
- **DOGI outperforms SOTA techniques even under dynamic I/O patterns**
  - 4.9 - 10.4% reduced WAF compared to best-performing baseline (i.e., MiDAS)



Overall WAF of each techniques

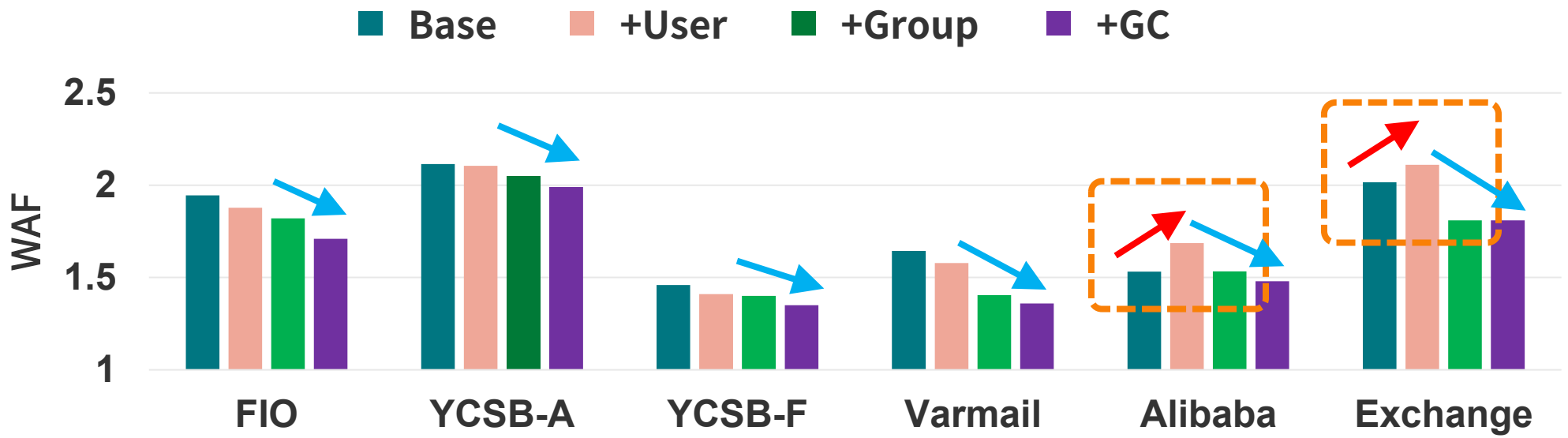
Impact of Each Design of DOGI

- Baseline: SepBIT
  - +User: Separate user-written blocks using hybrid mechanism
  - +Group: Apply group configuration that minimizes WAF
  - +GC: Relocate GC-written blocks using prediction log

Checklist

- *Predict user-written block's invalidation time accurately?*
- *Choose appropriate group configuration?*
- *Relocate GC-written block appropriately?*

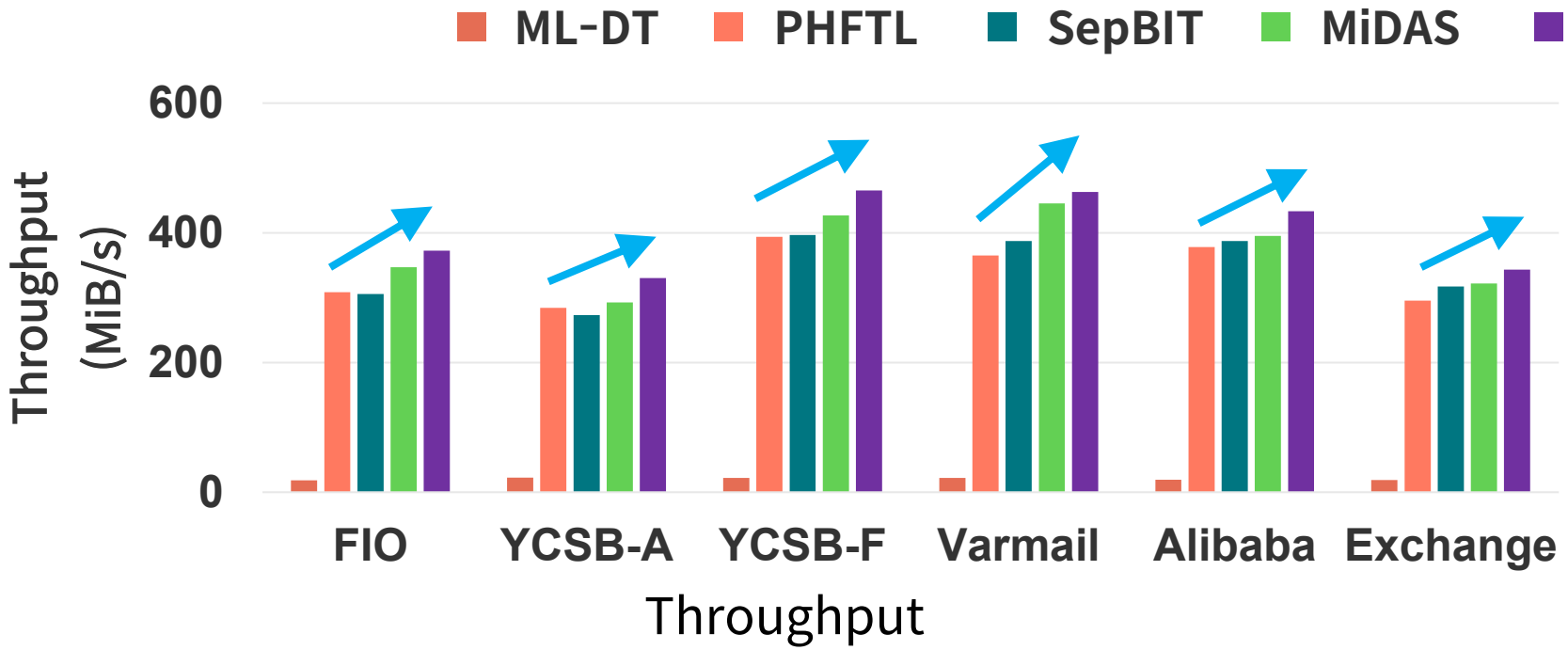
- **-1%, 15% and 22% reduced WAF** for +User, +Group, and +GC compared to baseline

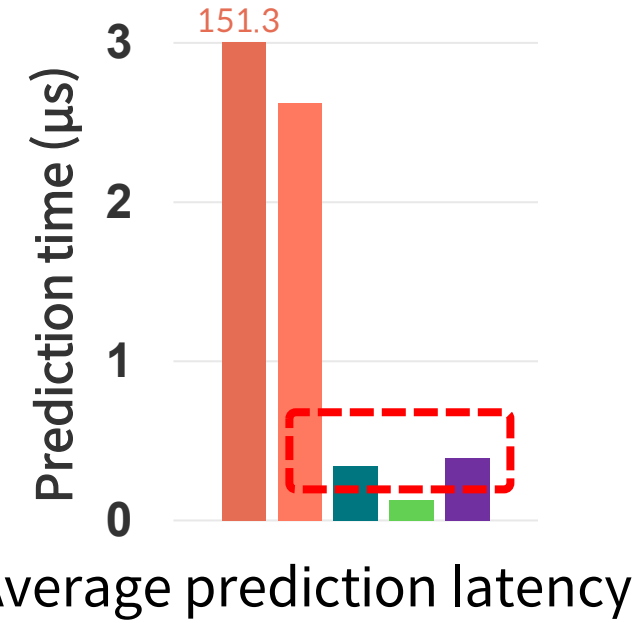


Throughput and Overhead Analysis

- 19.4x, 1.19x, 1.17x, and 1.09x higher throughput than ML-DT, PHFTL, SepBIT, and MiDAS
  - Low GC overhead and prediction overhead

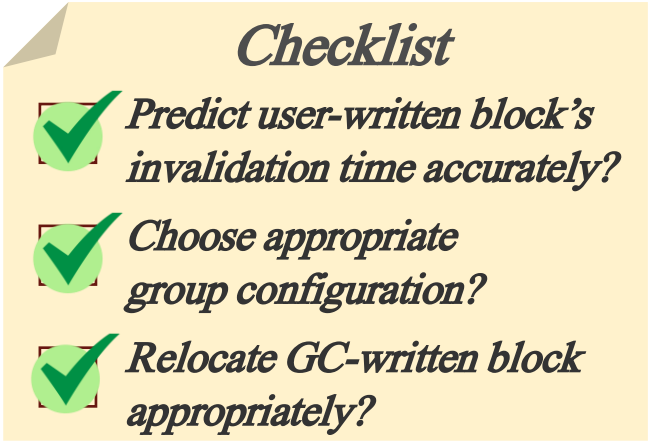
- DOGI enables fast invalidation-time prediction for user-written blocks
  - Faster than ML-DT and PHFTL (ML-based), and similar to SepBIT (heuristic-based)





Conclusion

- Through a comparison with NoDaP, we identify three key problems in existing data placement techniques:
  - Inaccurate prediction for user-written blocks
  - Fail to choose appropriate group configuration
  - Incorrect relocation of GC-written blocks



- DOGI addresses these three problems, reducing WAF by 15.5% and improving write throughput by 9.2% compared to best-performing baseline (i.e., MiDAS)

THANK YOU!