MiDAS:
Minimizing Write Amplification in Log-Structured Systems through Adaptive Group Number and Size Configuration

★**Seonggyun Oh**, ★Jeeyun Kim, Soyoung Han,
Jaeho Kim‡, Sungjin Lee, Sam H. Noh†

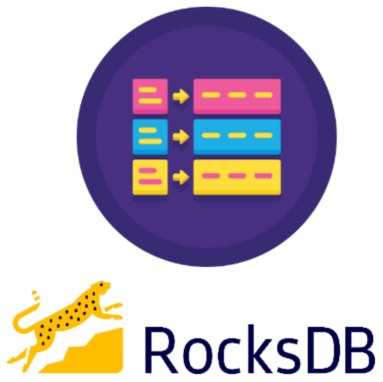
DGIST †Virginia Tech ‡Gyeongsang National University

★ These authors equally contributed to this work

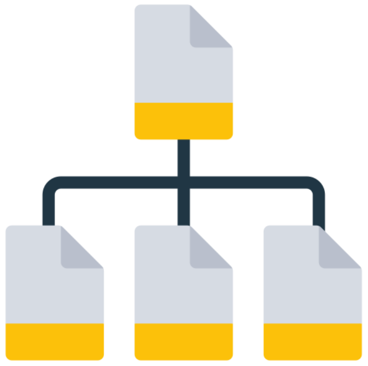
22nd USENIX Conference on File and Storage Technologies

Log-Structured Systems

- Widely used in various applications
  - key-value stores, file systems, and storage firmware
- Suitable for emerging storage media that only supports append-only writes



LSM-Tree based
KV stores



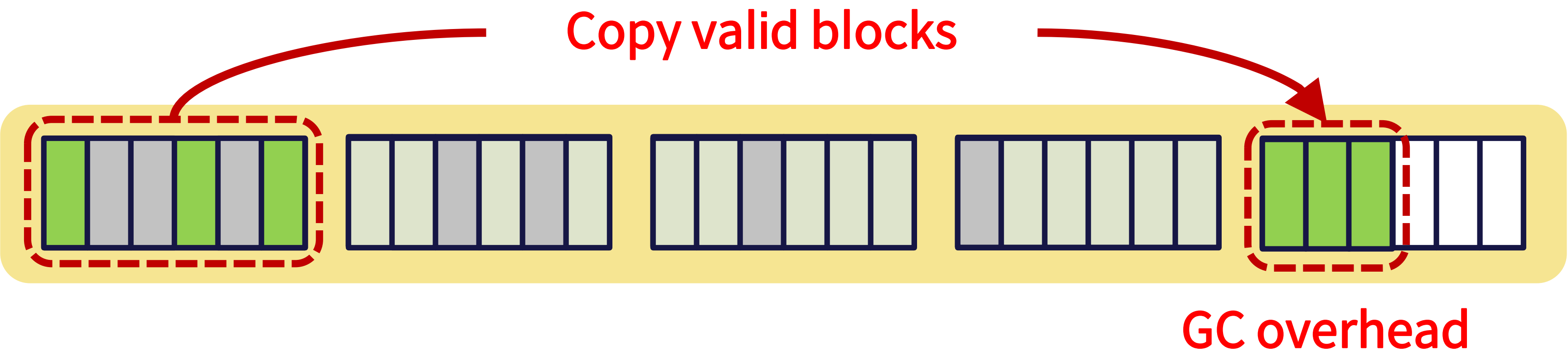
File systems
(F2FS, BtrFS)

Storage firmware
(Flash-based SSDs, ZNS SSDs)

Problem: GC Overhead in Log-structured Systems

- For GC, systems first select victim segment and rewrite valid blocks into free space
  - Incurs additional writes by copying valid blocks: GC writes

- Write amplification factor (WAF): The factor that shows additional writes by GC
  - $$WAF = \frac{\text{User writes} + \text{GC writes}}{\text{User writes}}$$

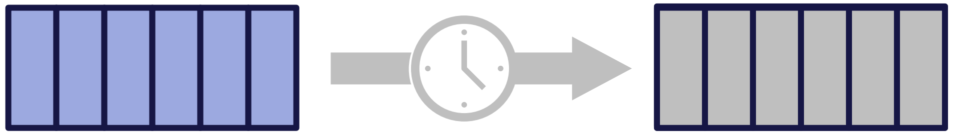


How to reduce WAF?

Approach for Reducing WAF: Data Placement

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

If blocks with similar invalidation times are grouped together

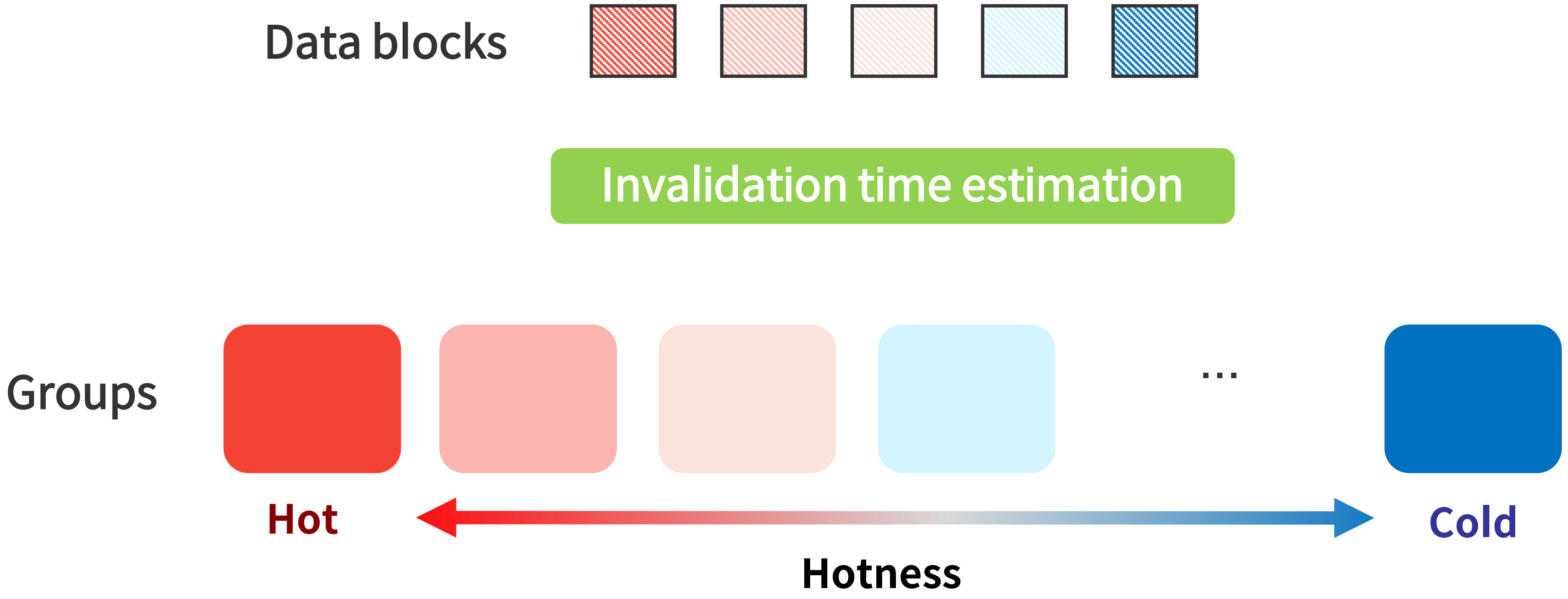


No valid data!
(WAF = 1)

Group data blocks with similar invalidation time

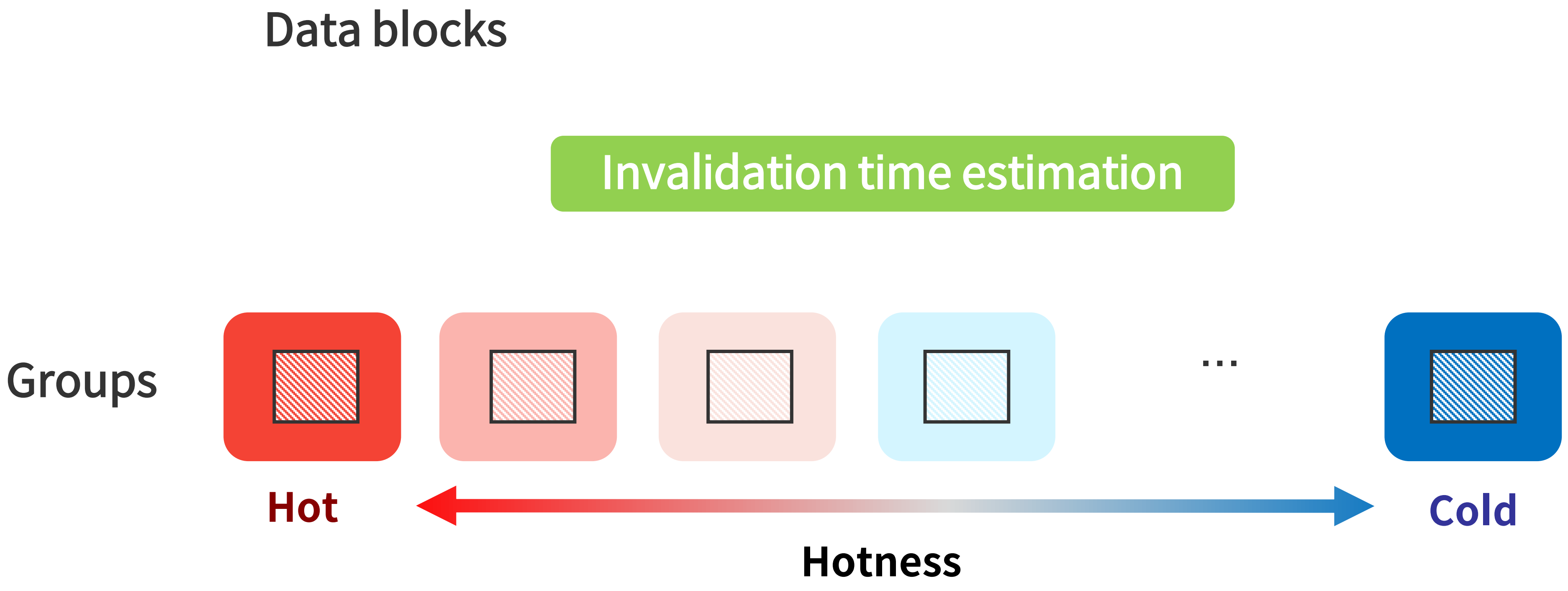
Approach for Reducing WAF: Data Placement (cont’)

- Data blocks are placed in groups according to their hotness
  - blocks with short invalidation time: Assigned into hot groups



Approach for Reducing WAF: Data Placement (cont’)

- Data blocks are placed in groups according to their hotness
  - Blocks with short invalidation time: Assigned into hot groups

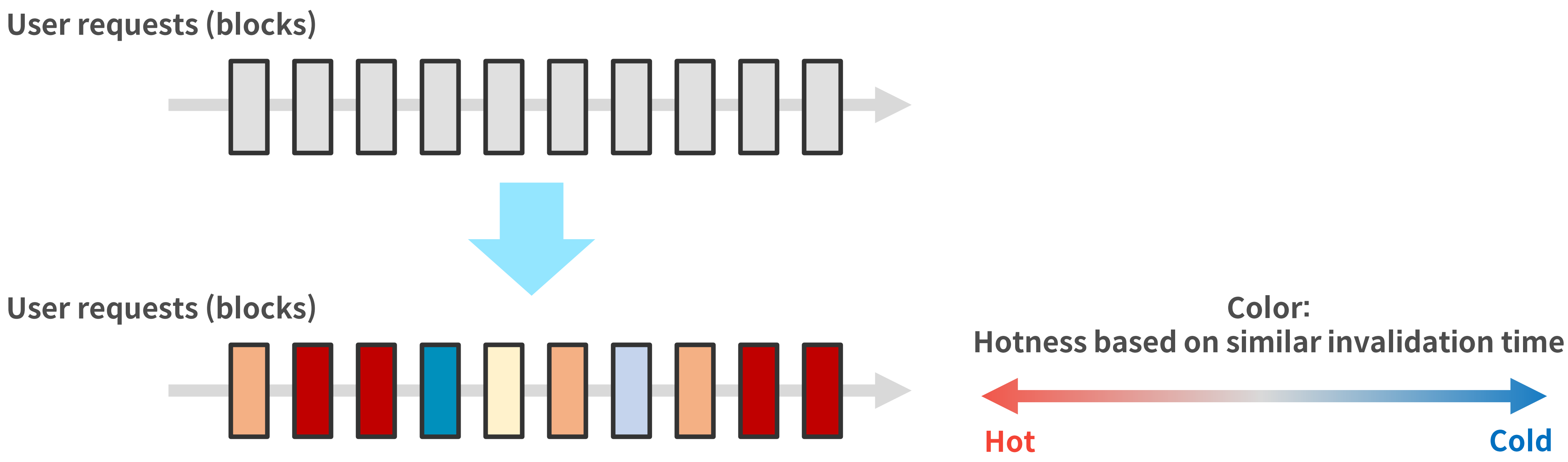


Then, what is optimal data placement?

Index

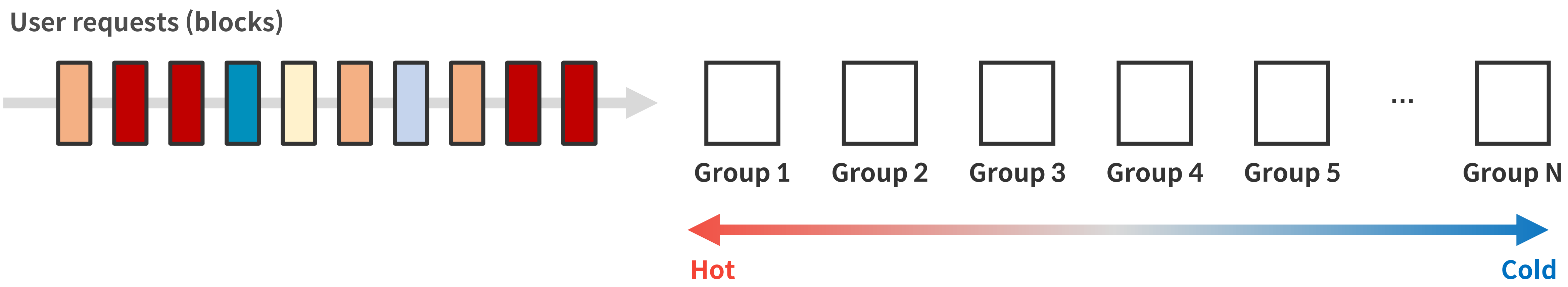
Optimal Data Placement

- **Three conditions for optimal data placement**
  - Predict invalidation time of all blocks exactly



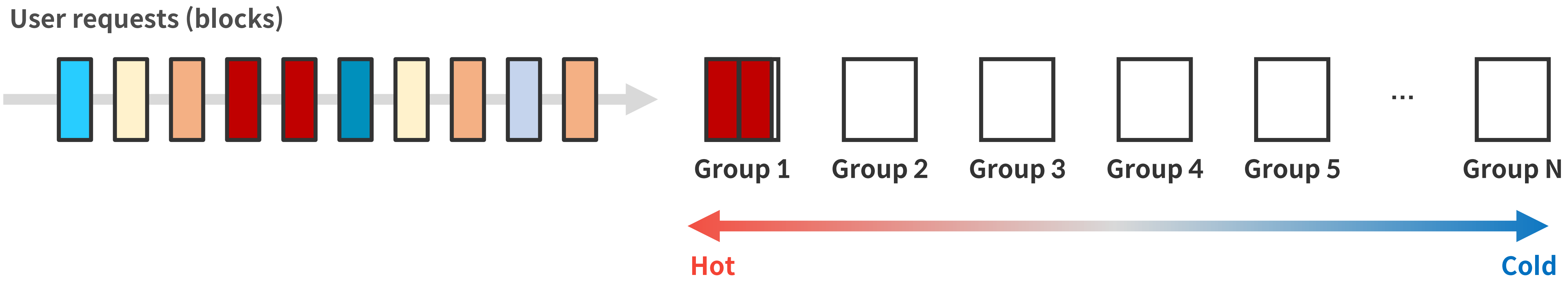
Optimal Data Placement (cont')

- **Three conditions for optimal data placement**
  - Predict invalidation time of all blocks exactly
  - Set number of groups based on number of colors



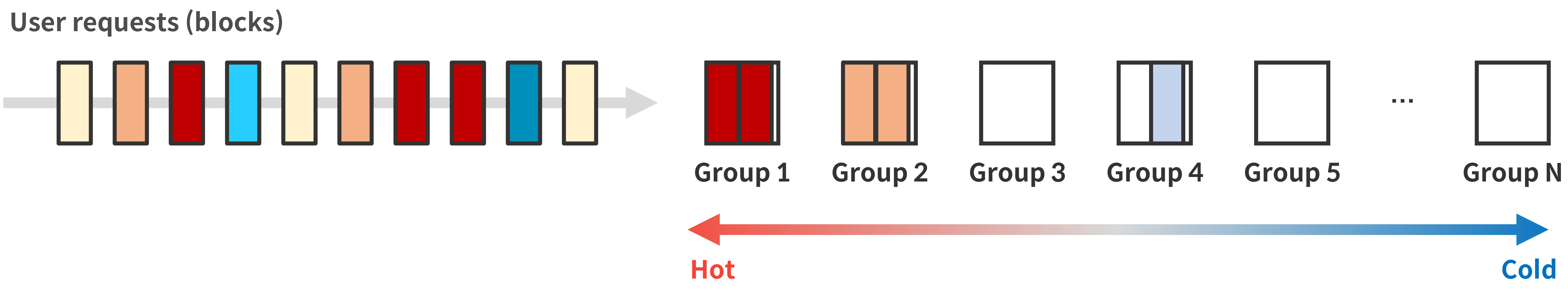
Optimal Data Placement (cont')

- **Three conditions for optimal data placement**
  - Predict invalidation time of all blocks exactly
  - Set number of groups based on number of colors



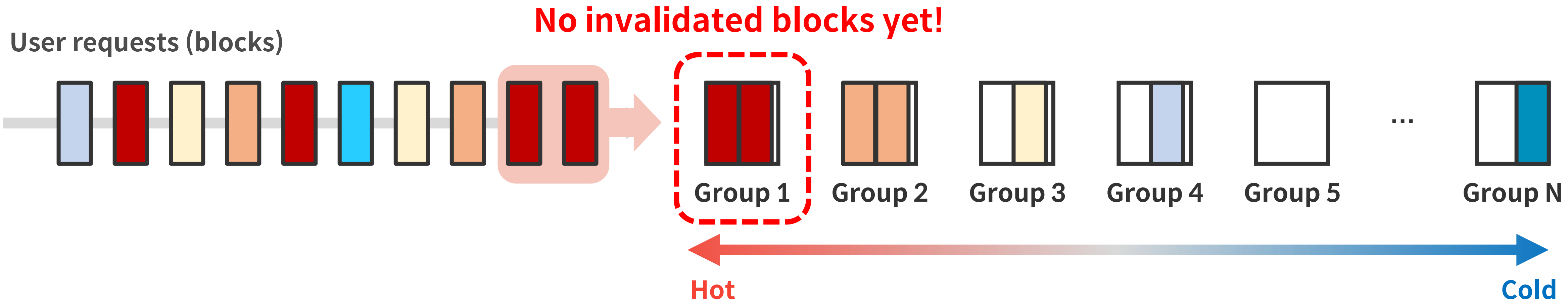
Optimal Data Placement (cont')

- **Three conditions for optimal data placement**
  - Predict invalidation time of all blocks exactly
  - Set number of groups based on number of colors



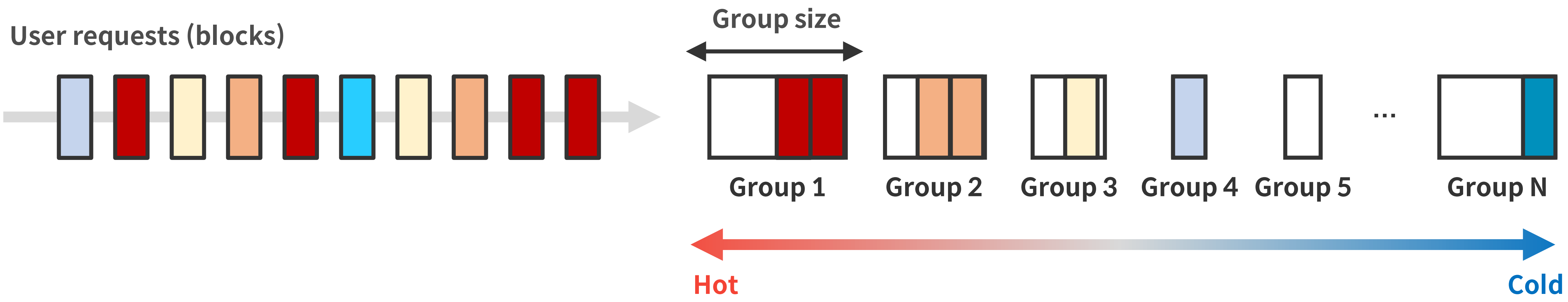
Optimal Data Placement (cont')

- **Three conditions for optimal data placement**
  - Predict invalidation time of all blocks exactly
  - Set number of groups based on number of colors



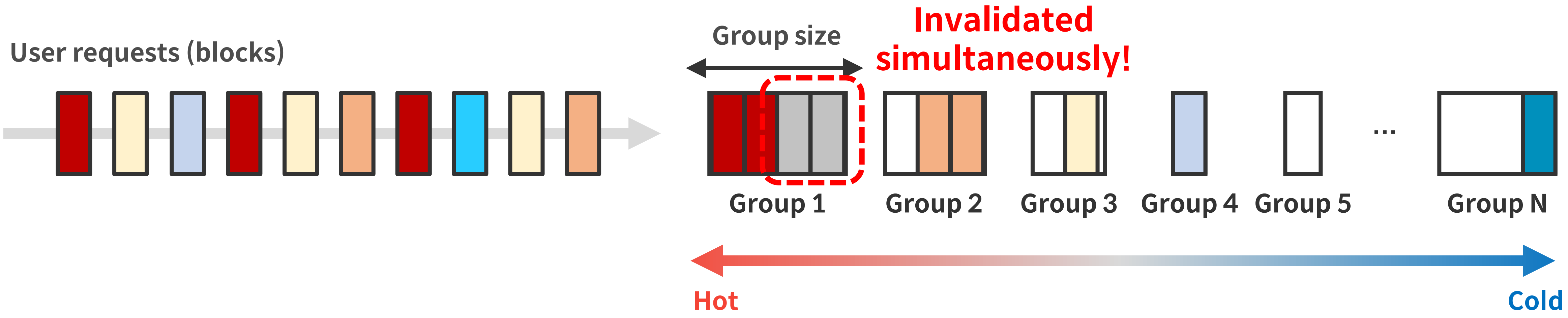
Optimal Data Placement (cont')

- **Three conditions for optimal data placement**
  - Predict invalidation time of all blocks exactly
  - Set number of groups based on number of colors
  - Set appropriate group size



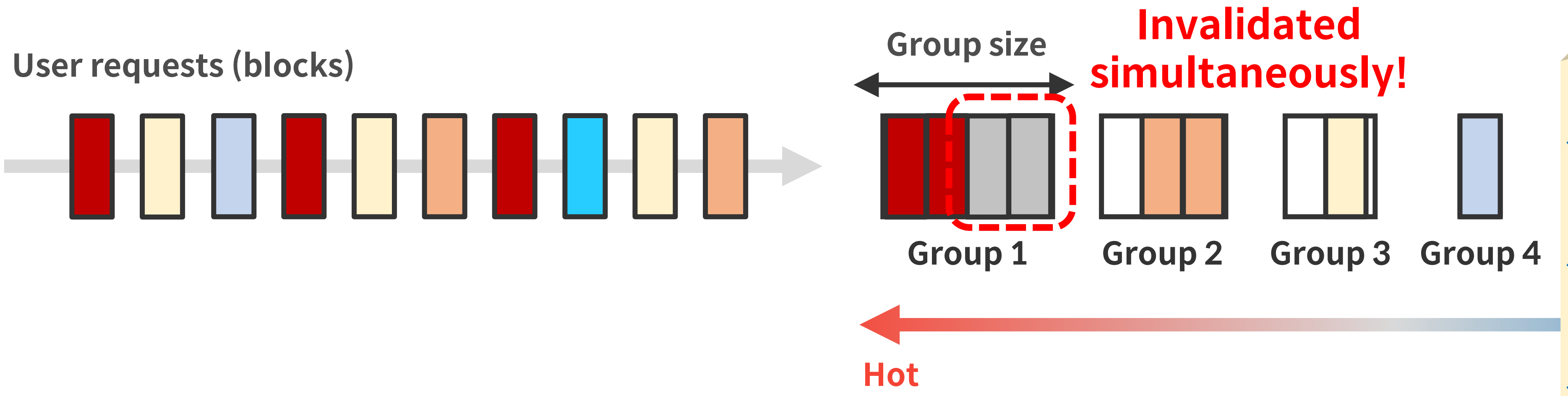
Optimal Data Placement (cont')

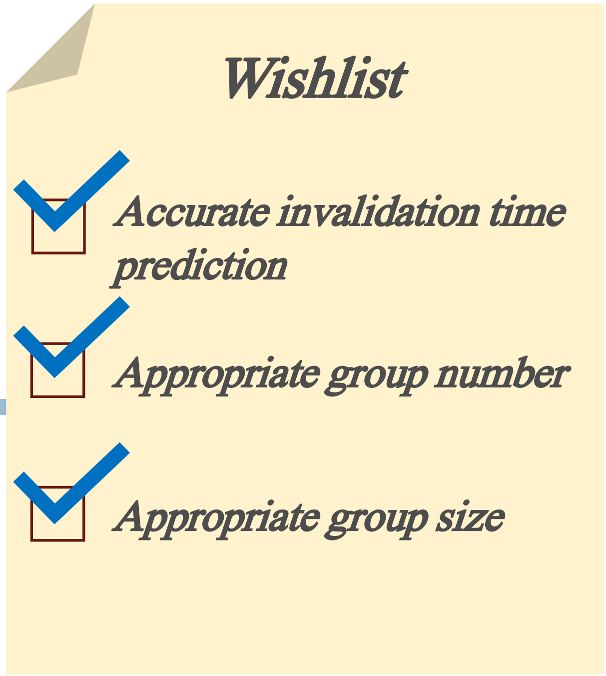
- **Three conditions for optimal data placement**
  - Predict invalidation time of all blocks exactly
  - Set number of groups based on number of colors
  - Set appropriate group size



Optimal Data Placement (cont')

- **Three conditions for optimal data placement**
  - Predict invalidation time of all blocks exactly
  - Set number of groups based on number of colors
  - Set appropriate group size





Index

- Log-Structured System and Its Problem
- **Motivation**
  - What is Optimal Solution?
  - **Why Not Optimal?**
- MiDAS Design
- Evaluation
- Conclusion

Wishlist

- [ ] *Accurate invalidation time prediction*
- [ ] *Appropriate group number*
- [ ] *Appropriate group size*

How about existing techniques?

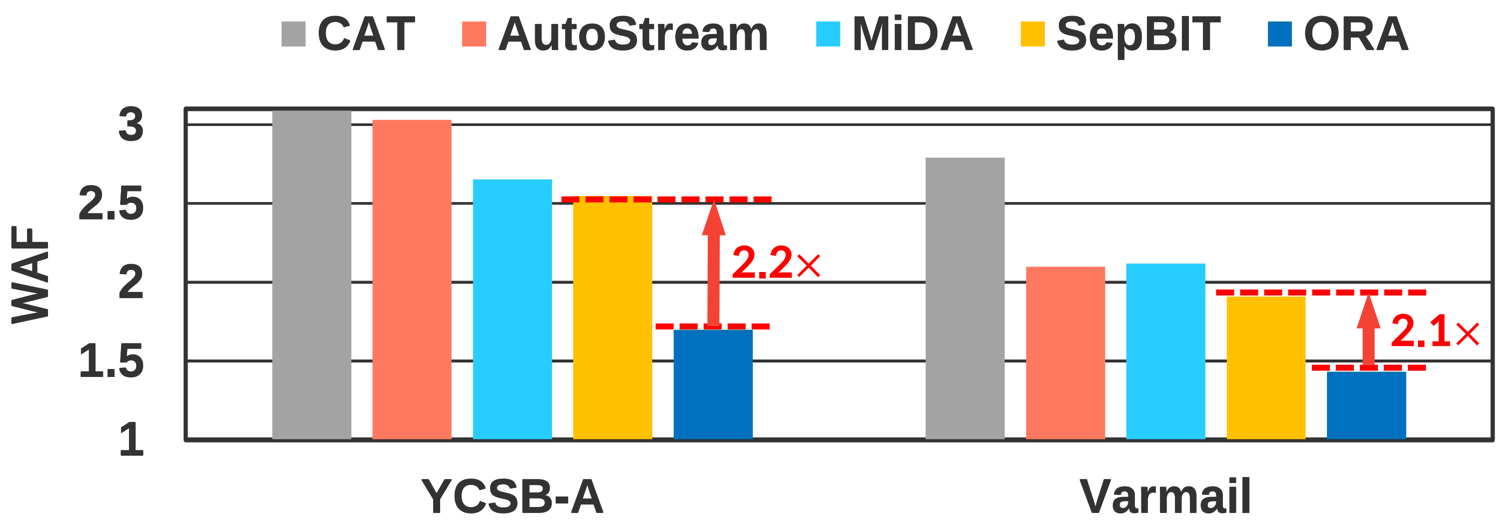
Oracle Algorithm (ORA)

- Implementing Optimal algorithm is NP-Complete

- We implement Oracle algorithm (ORA) to imitate Optimal algorithm
  - Perform trace analysis to find when data blocks are invalidated (updated)
  - Utilize K-mean clustering to decide the number of groups
  - Experientially find appropriate group sizes

ORA vs SOTA Techniques

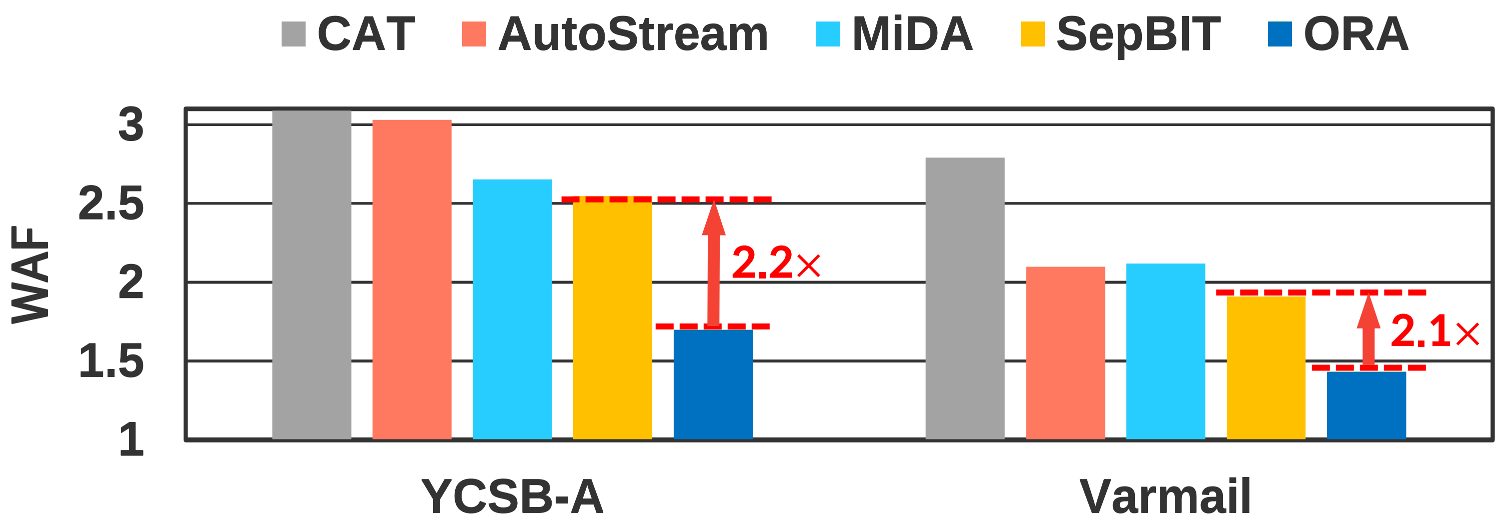
- Large gap between ORA and existing SOTA techniques
  - SOTA: MiDA [APSYS'21], SepBIT [FAST'22]



- **SepBIT** [Separating Data via Block Invalidation Time Inference for Write Amplification Reduction in Log-Structured Storage (FAST'22)]
- **MiDA** [Lightweight Data Lifetime Classification Using Migration Counts to Improve Performance and Lifetime of Flash-Based SSDs (APSys'21)]
- **AutoStream** [AutoStream: Automatic stream management for multi-streamed SSDs (SYSTOR'17)]
- **CAT** [Cleaning Policies in Mobile Computers Using Flash Memory (Journal of Systems and Software 1999)]

ORA vs SOTA Techniques

- Large gap between ORA and existing SOTA techniques
  - SOTA: MiDA [APSYS'21], SepBIT [FAST'22]

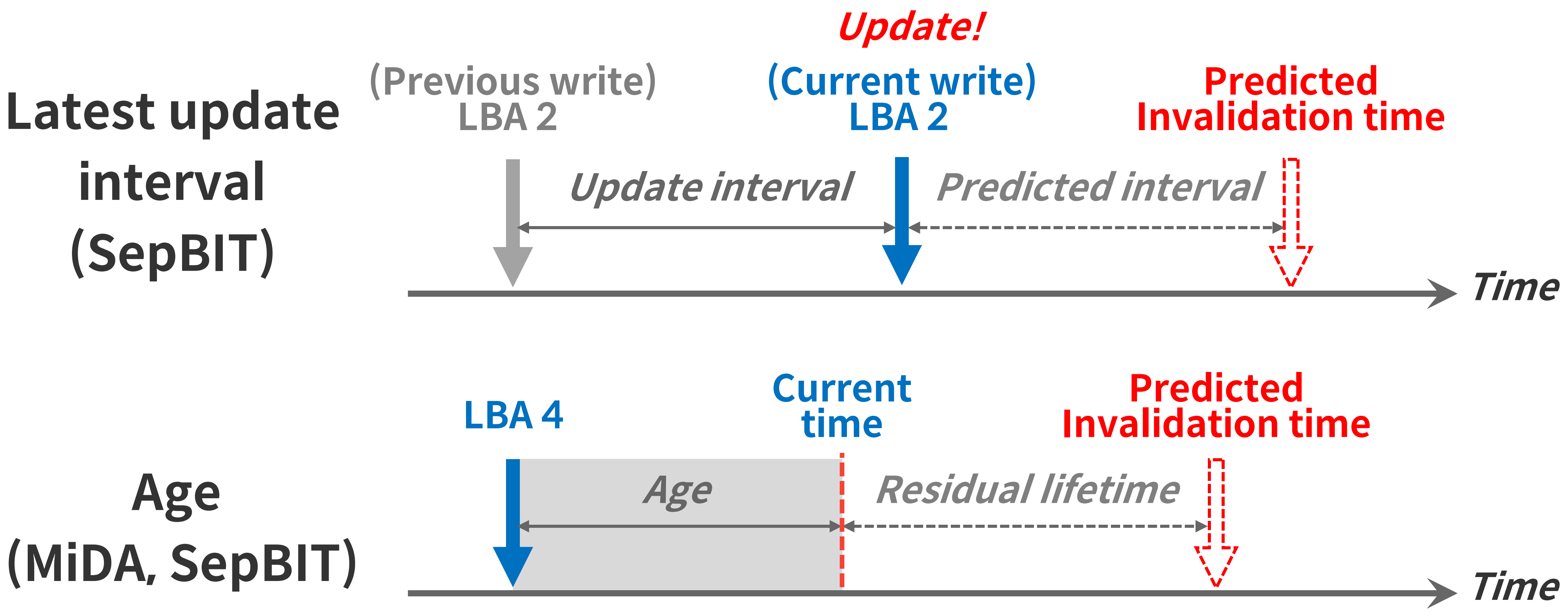


- SepBIT [Separating Data via Block Invalidation Time Inference for Write Amplification Reduction in Log-Structured Storage (FAST'22)]
- MiDA [Lightweight Data Lifetime Classification Using Migration Counts to Improve Performance and Lifetime of Flash-Based SSDs (APSys'21)]
- AutoStream [AutoStream: Automatic stream management for multi-streamed SSDs (SYSTOR'17)]

Why the gap between ORA and SOTA?

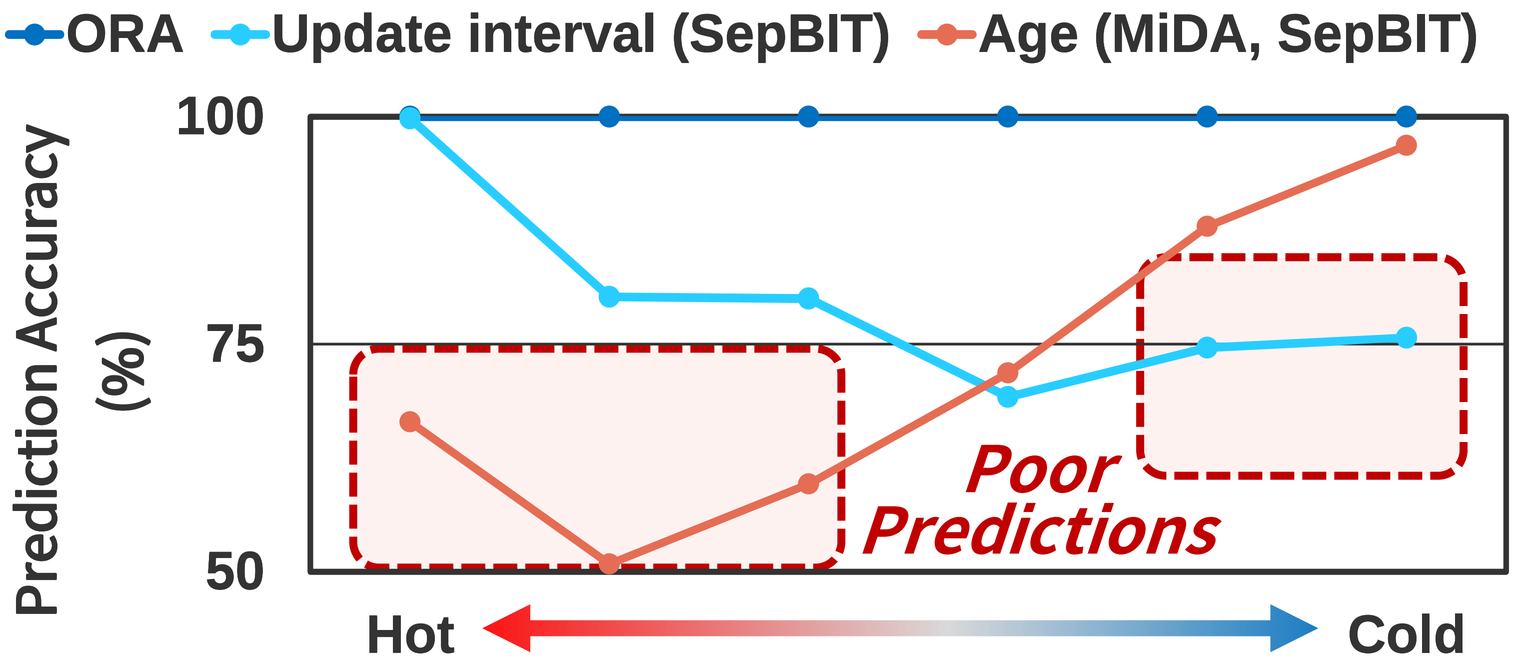
1. Inaccurate Estimation

- SOTA techniques estimate block invalidation time inaccurately



1. Inaccurate Estimation (cont')

- SOTA techniques estimate block invalidation time inaccurately



Wishlist

- [X] *Accurate invalidation time prediction*
- [ ] *Appropriate group number*
- [ ] *Appropriate group size*

2. Lack Consideration of Group Number (cont')

- Fail to set appropriate number of groups

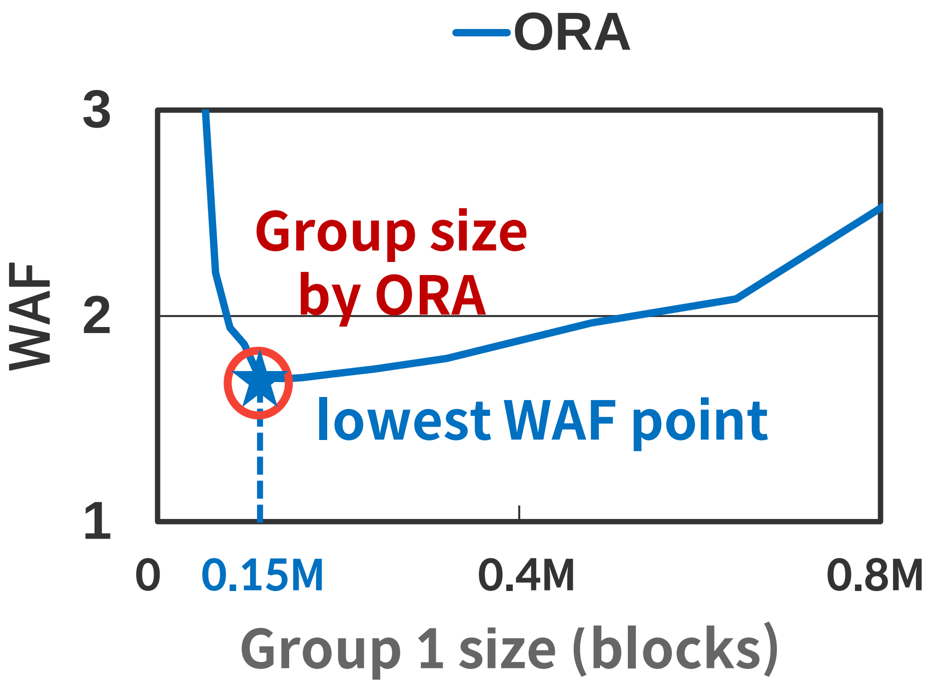
Existing Approaches	Optimal Approach
Fixed number of groups • SepBIT: 6 groups • MiDA: 8 groups	appropriate number of groups
Do not adjust to workload patterns	Dynamic adjusted to workload patterns

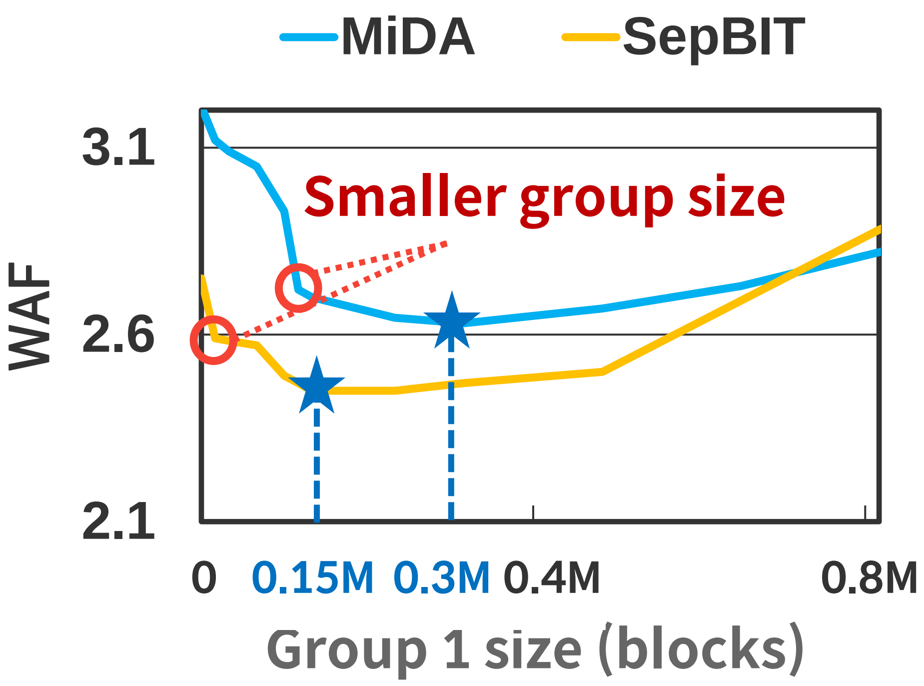
Wishlist

- [X] *Accurate invalidation time prediction*
- [X] *Appropriate group number*
- [ ] *Appropriate group size*

3. Lack Consideration of Group Size

- Fail to set appropriate group sizes





Wishlist

- X *Accurate invalidation time prediction*
- X *Appropriate group number*
- X *Appropriate group size*

Motivation Summary: Why Not Optimal?

1. Inaccurate prediction of block invalidation time
   - blocks may be assigned to wrong group

How to predict invalidation time accurately?

2. Lack consideration of group number
3. Lack consideration of group sizes

Lack consideration of group configuration

   - Inappropriate group configuration incur unnecessary data copies

How to find best group configuration?

Wishlist

- X *Accurate invalidation time prediction*
- X *Appropriate group number*
- X *Appropriate group size*

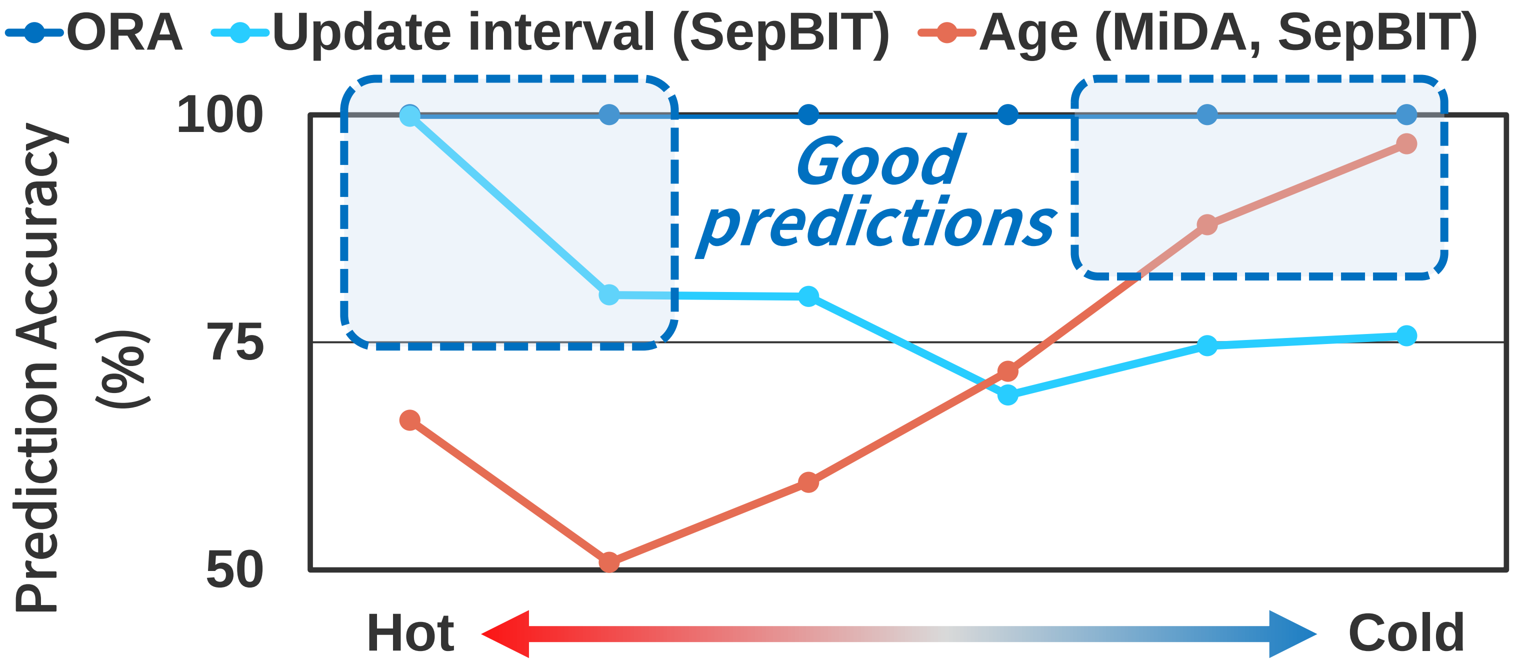
Index

Wishlist

- [ ] *Accurate invalidation time prediction*
- [ ] *Appropriate group number*
- [ ] *Appropriate group size*

Design Goal #1: Accurate Invalidation Time Prediction

- How to accurately estimate invalidation?
  - Fortunately, we have hints from previous work
- Each method shows different strength



Wishlist

- [x] *Accurate invalidation time prediction*
  Different estimation methods by block's hotness!
- [ ] *Appropriate group number*
- [ ] *Appropriate group size*

Design Goal #2: Finding Best Configuration

- How to find best number of groups and their sizes?

What is "best"? → Lowest WAF!

- How to estimate WAF for given workload and system?

→ **Mathematical modeling of system and workload pattern analysis**

Wishlist

- *Accurate invalidation time prediction*

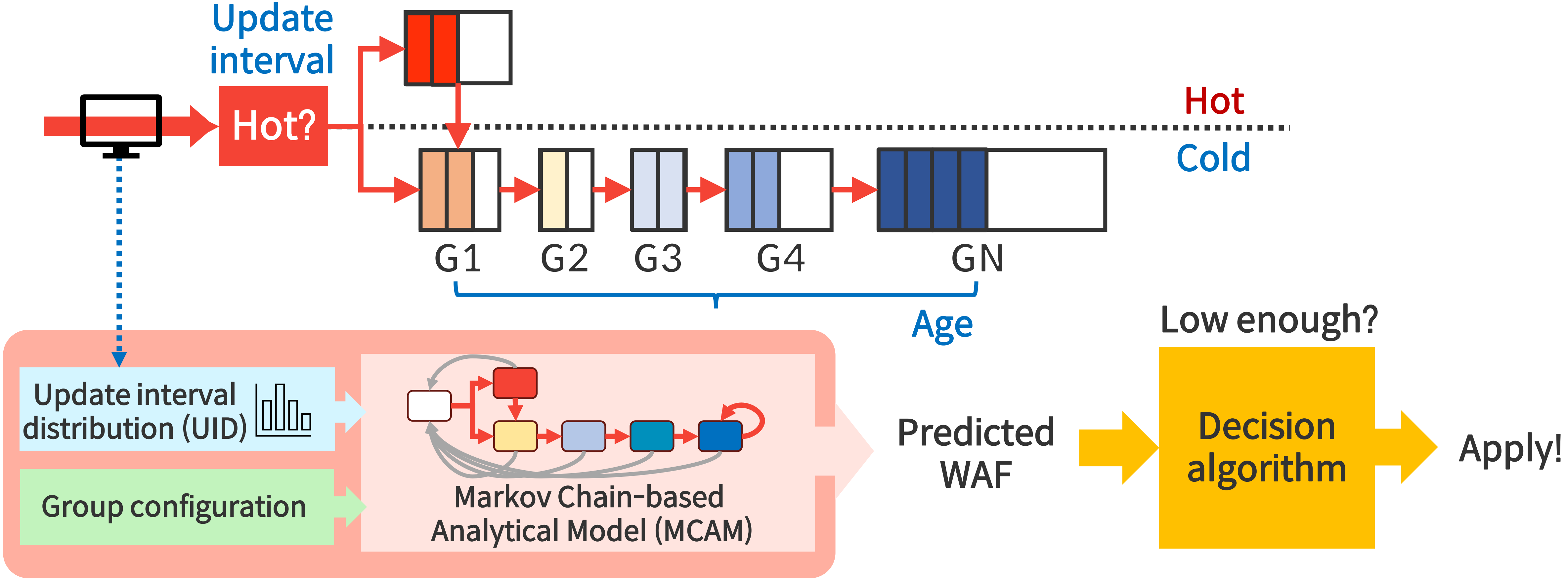
  Different estimation methods by block's hotness!

- *Appropriate group number*
- *Appropriate group size*

  Finding best configuration by modeling system and workload pattern analysis!

Design Overview of MiDAS

1 *Accurate Invalidation time prediction using update interval and age methods*

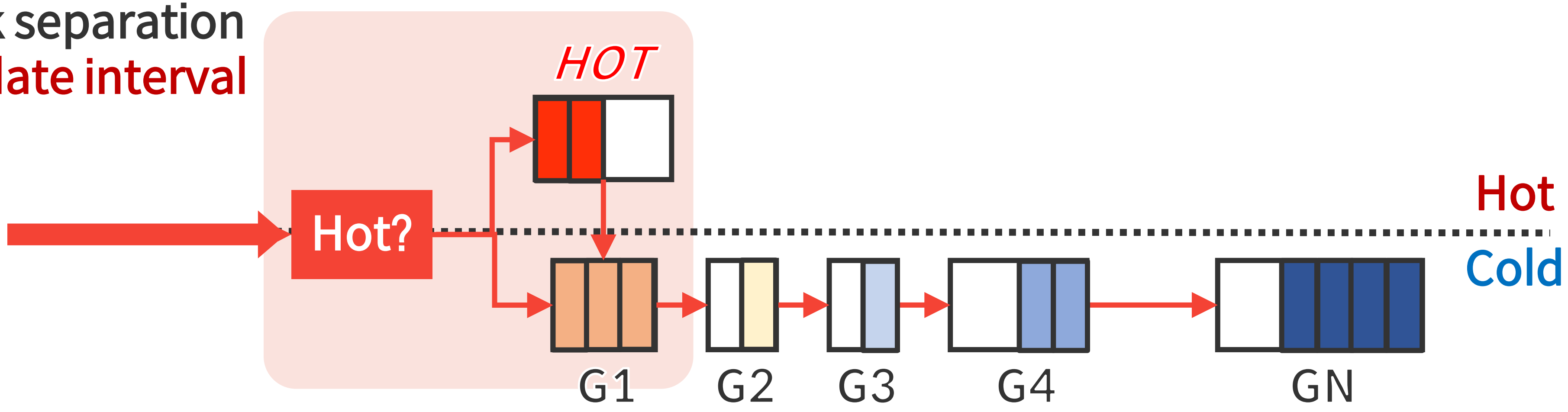


2 *Finding best configuration by WAF prediction module (MCAM and UID) for given group configuration*

Design #1: Accurate Invalidation Time Prediction

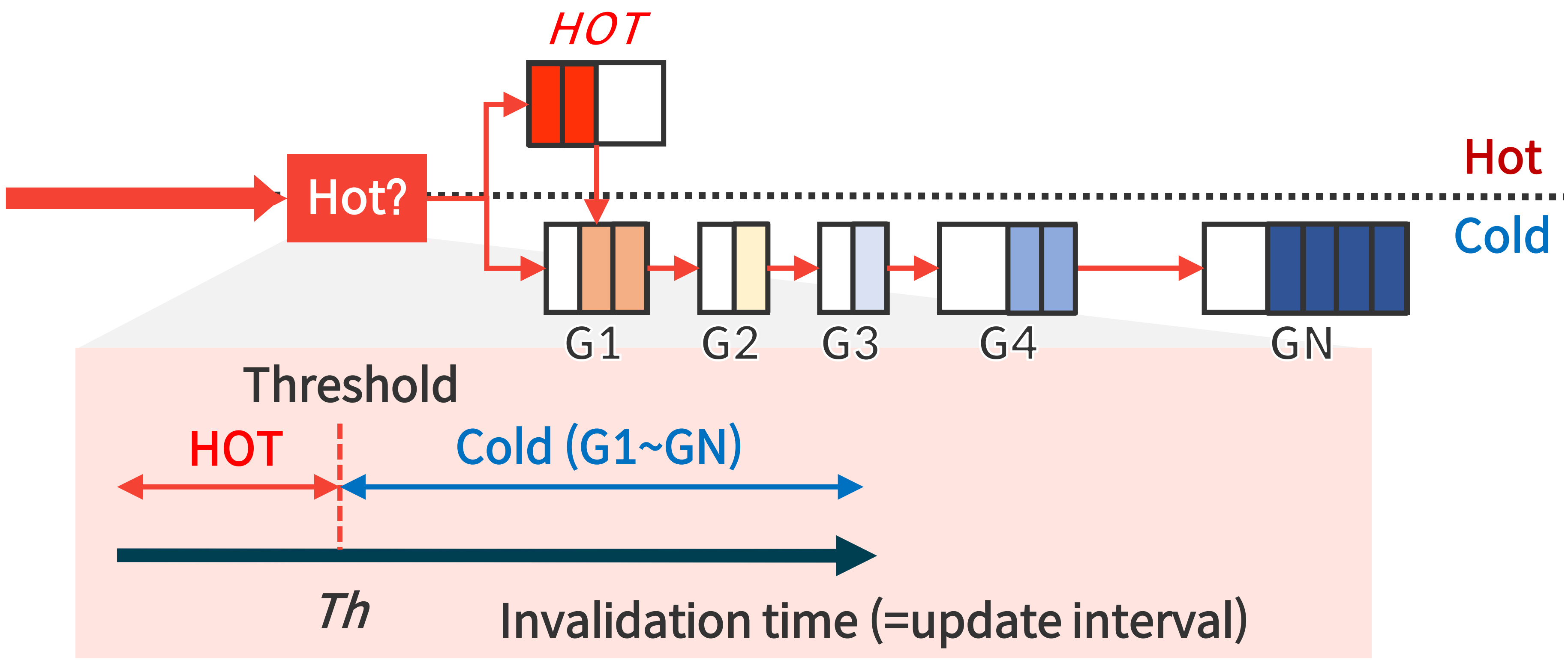
- Hot/cold block separation using accurate block invalidation time estimation methods
  - Hot block separation: Use SepBIT method!
    - **Separating hot blocks using latest update interval**

Hot block separation using update interval



Design #1: Hot-Cold Threshold

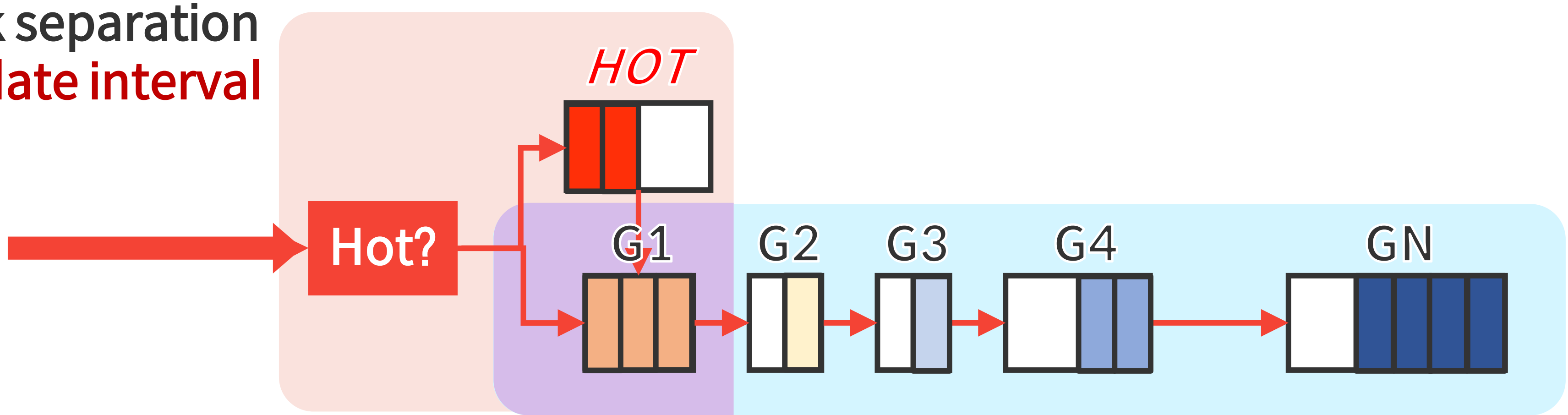
- Invalidation time threshold separates hot and cold blocks
  - Invalidation time $\leq Th$: HOT
    $\geq Th$: Cold
- Question: How to set hot-cold threshold?



Design #1: Accurate Invalidation Time Prediction

- Hot/cold block separation using accurate block invalidation time estimation methods
  - Hot block separation: Use SepBIT method!
    - **Separating hot blocks using latest update interval**
  - Cold block separation: Use MiDA method!
    - **Separating cold blocks using migration count**
      (age ∝ migration count)

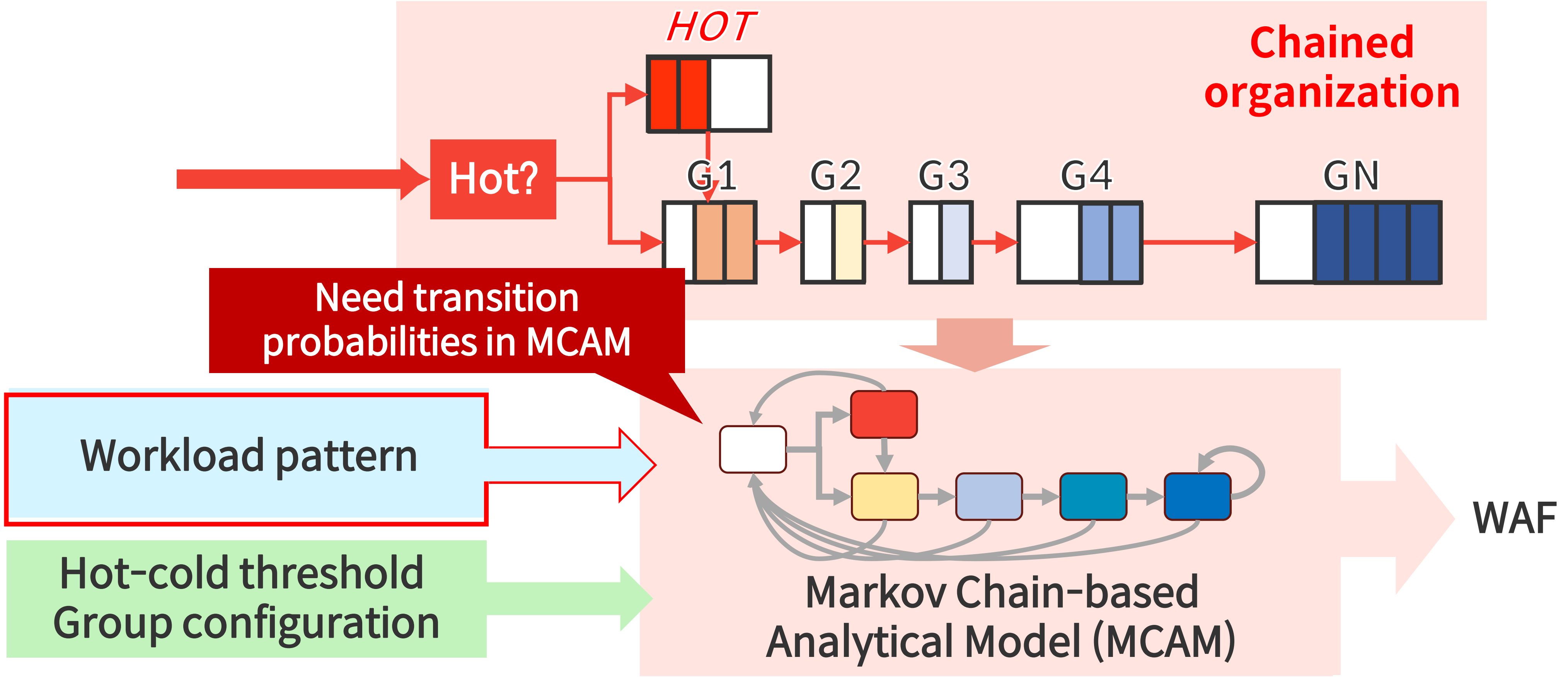
Hot block separation using update interval



Cold block separation using migration count (age)

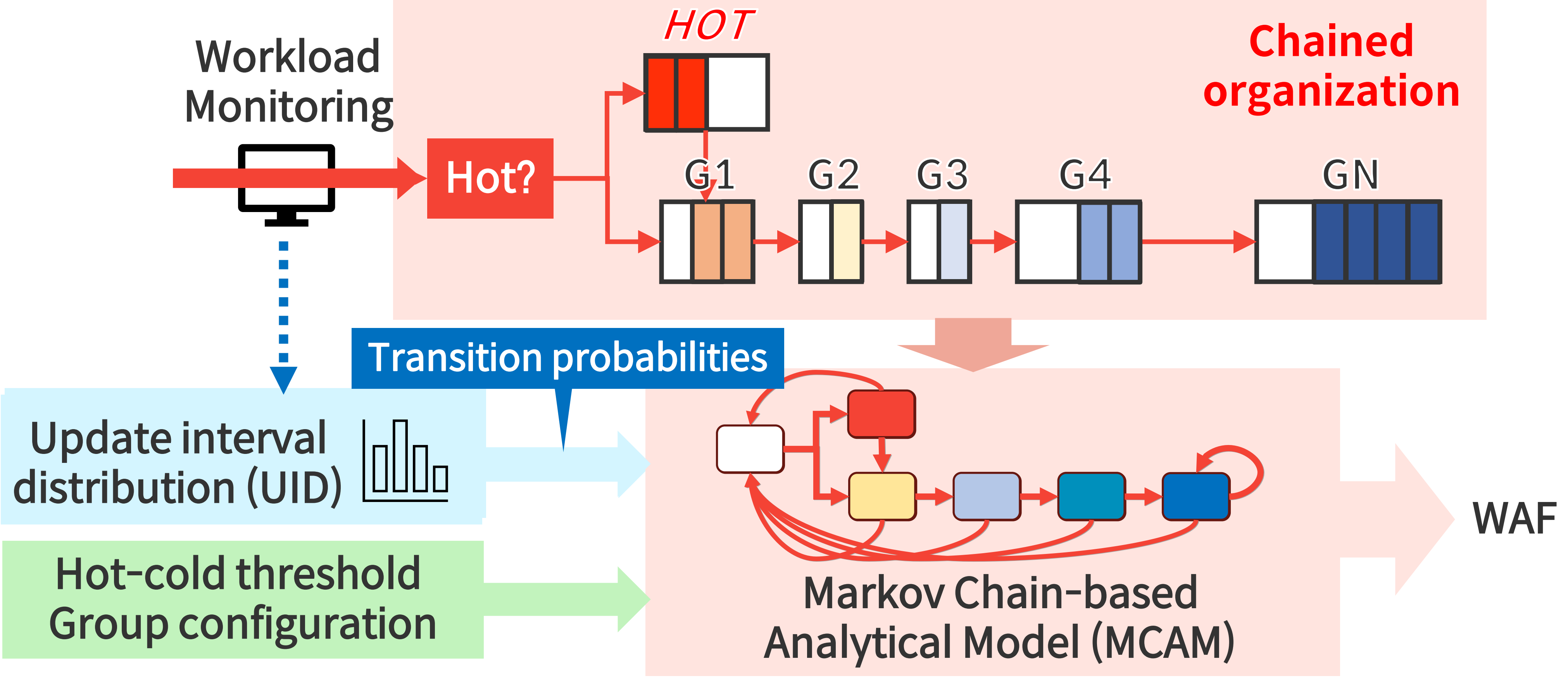
Design #2: Finding Best Configuration

- Two elements for WAF prediction to find best group configuration and hot-cold threshold
  - Mathematical system modeling → Markov chain-based analytical model (MCAM)



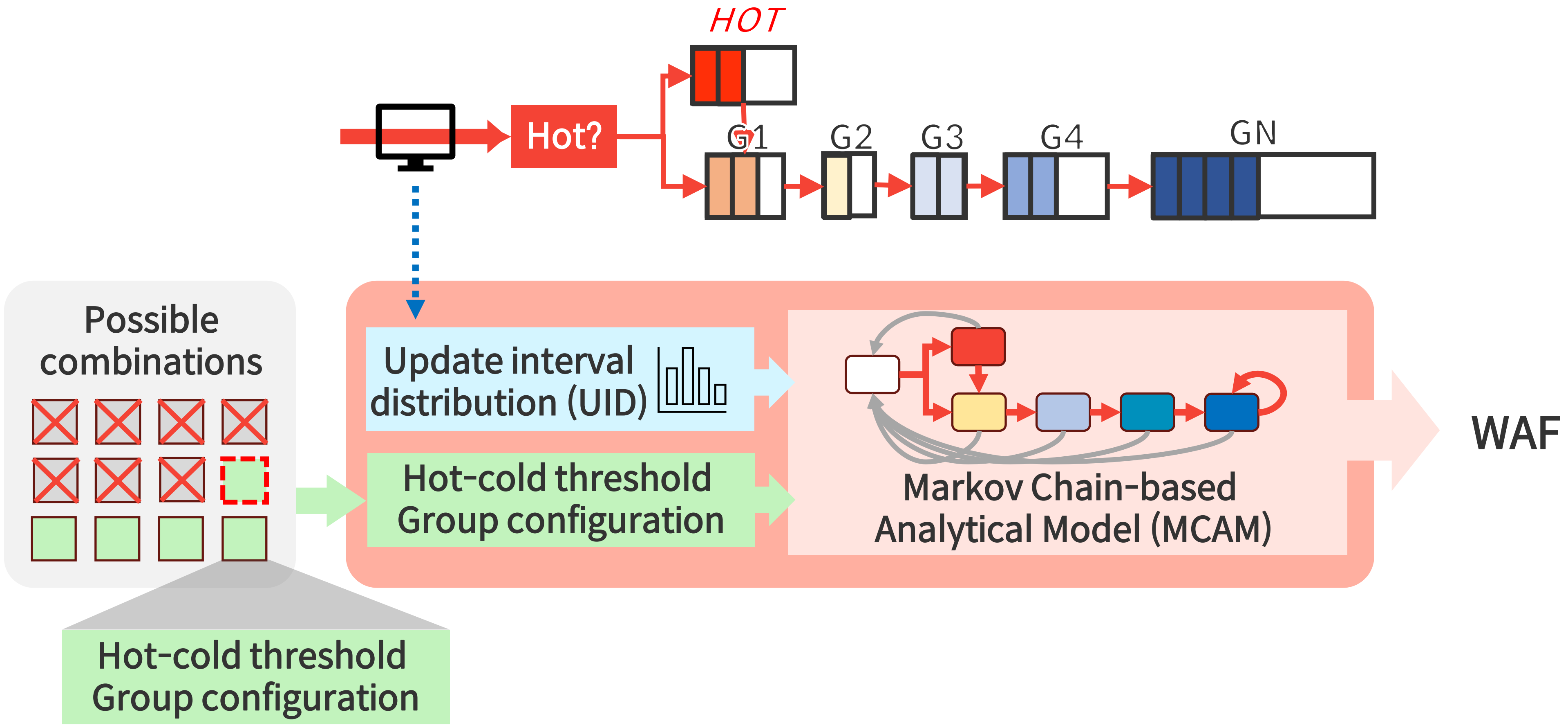
Design #2: Finding Best Configuration (cont')

- Two elements for WAF prediction to find best group configuration and hot-cold threshold
  - Mathematical system modeling → Markov chain-based analytic model (MCAM)
  - Workload pattern analysis → Update interval distribution (UID) for incoming blocks



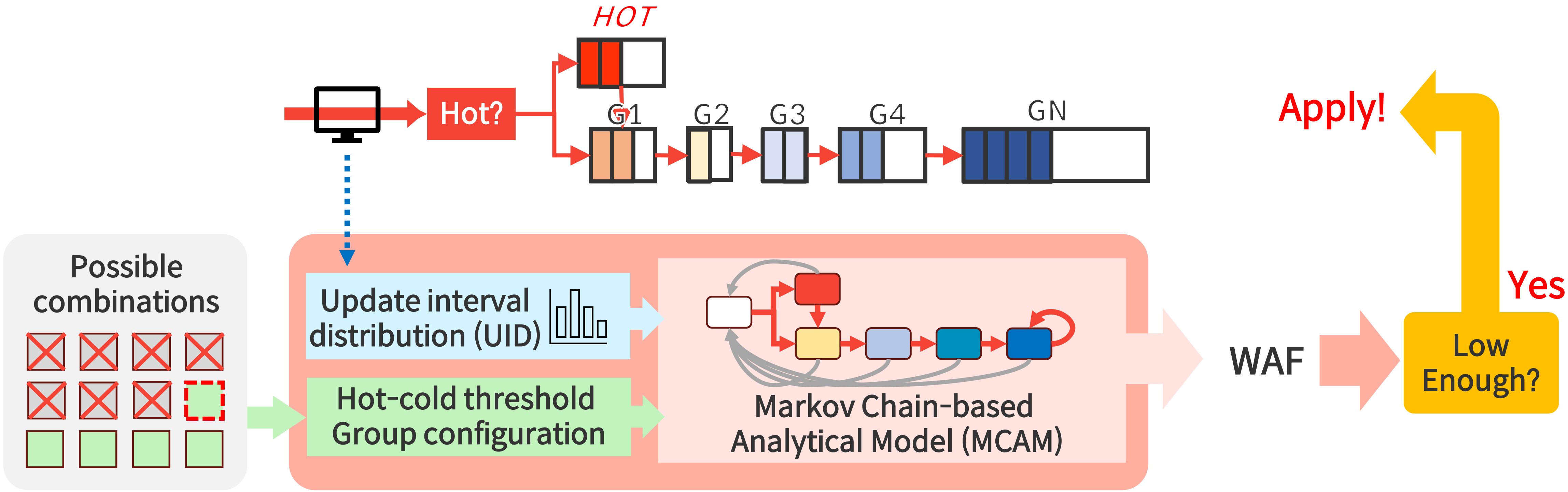
Design #2: Finding Best Configuration (cont')

- Finding best one by evaluating WAF for various combinations of hot-cold threshold and group configuration
  - Combination: hot-cold threshold and group configuration



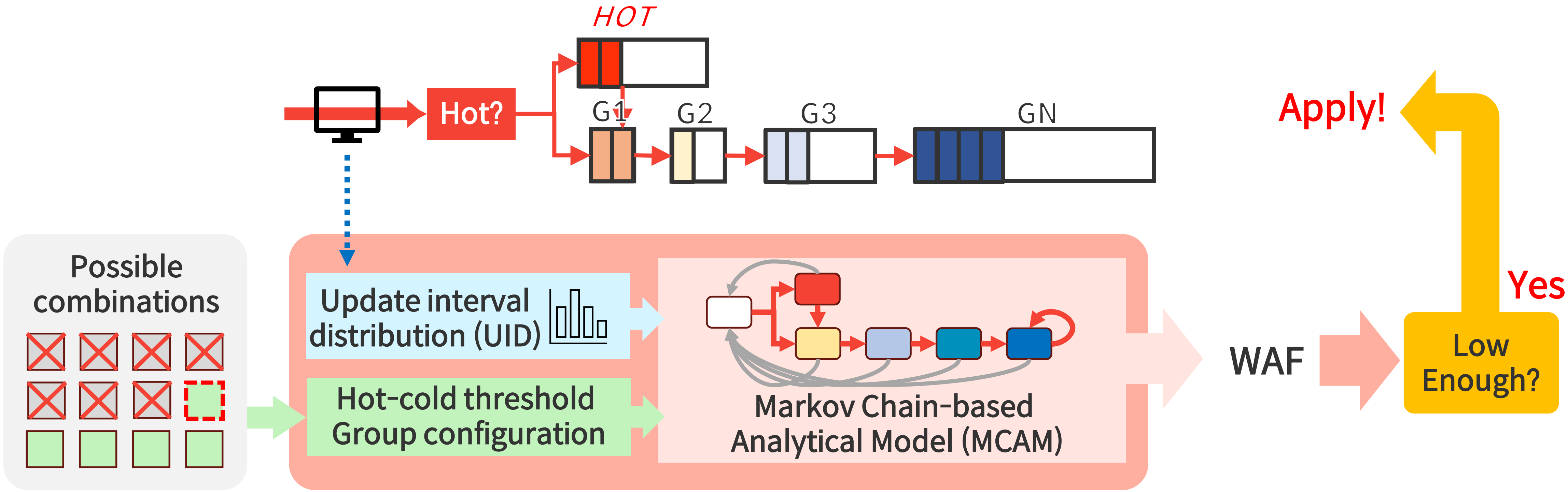
Design #2: Finding Best Configuration (cont')

- Finding best one by evaluating WAF for various combinations of hot-cold threshold and group configuration
  - Combination: hot-cold threshold and group configuration



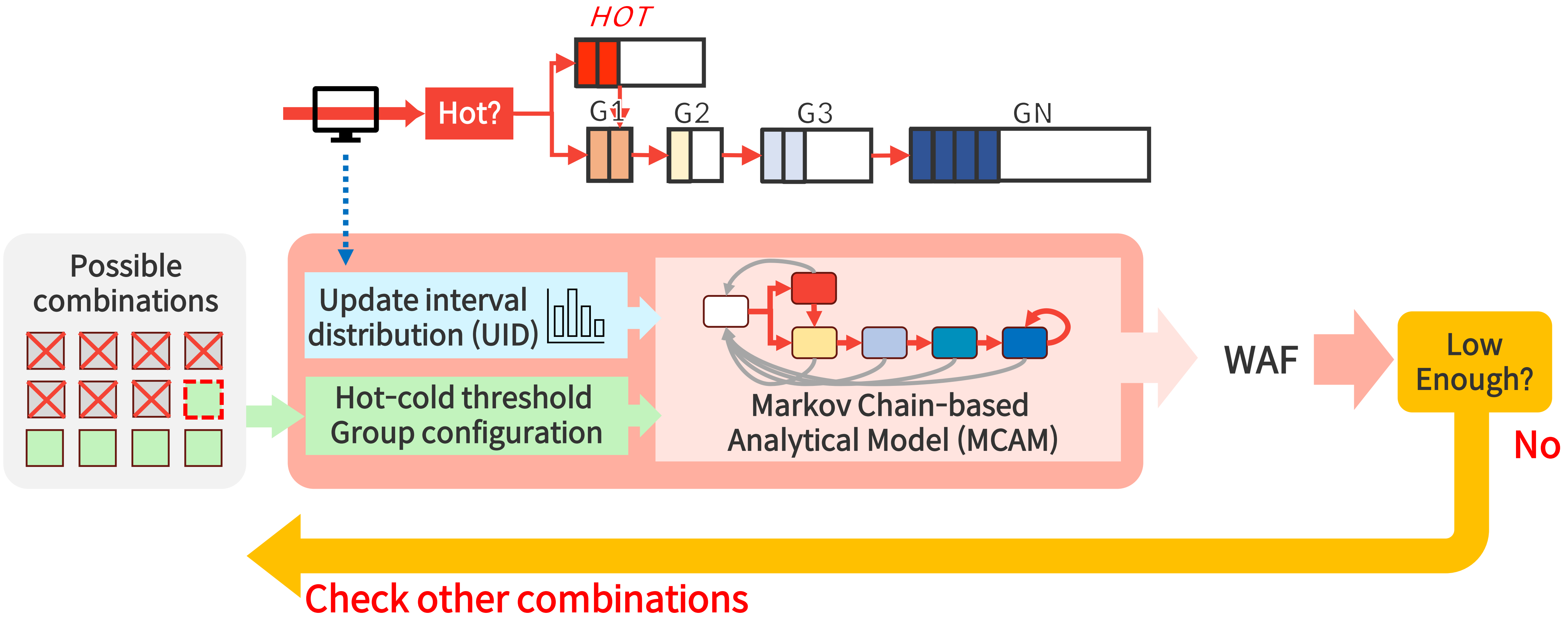
Design #2: Finding Best Configuration (cont')

- Finding best one by evaluating WAF for various combinations of hot-cold threshold and group configuration
  - Combination: hot-cold threshold and group configuration



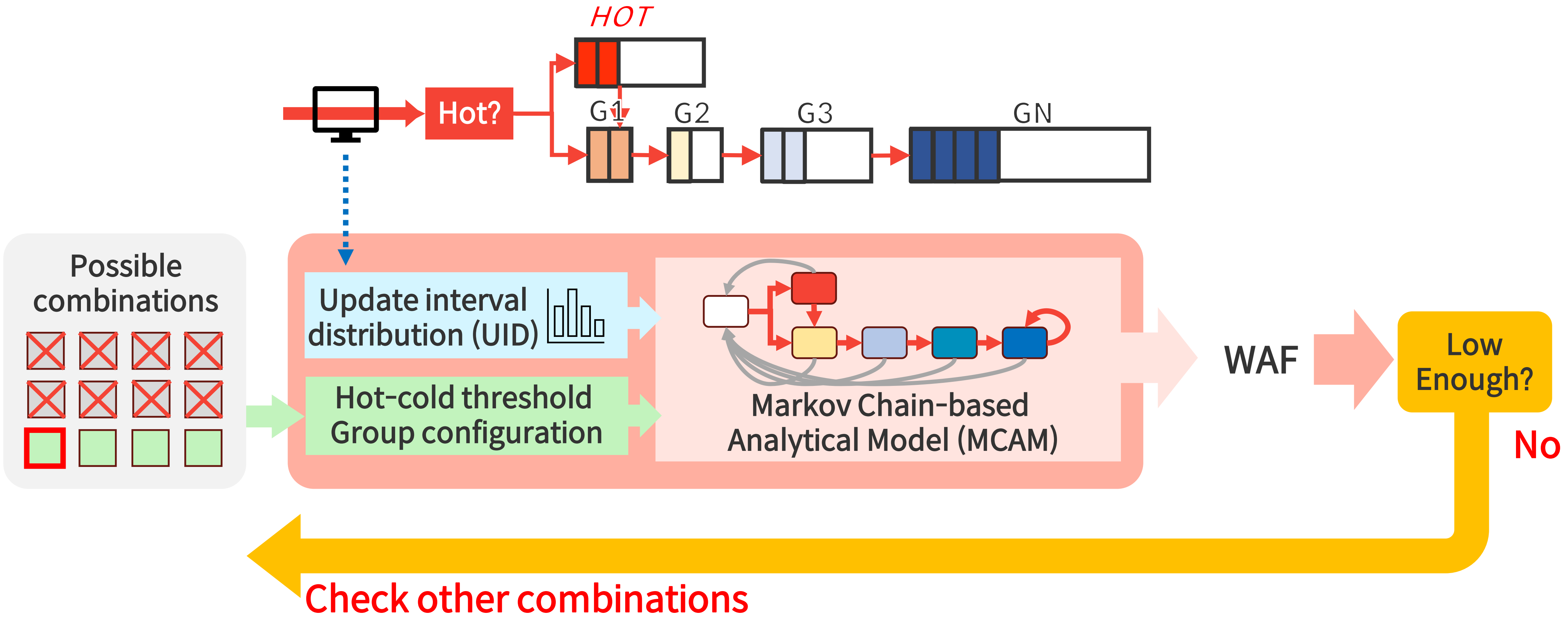
Design #2: Finding Best Configuration (cont')

- Finding best one by evaluating WAF for various combinations of hot-cold threshold and group configuration
  - Combination: hot-cold threshold and group configuration



Design #2: Finding Best Configuration (cont’)

- Finding best one by evaluating WAF for various combinations of hot-cold threshold and group configuration
  - Combination: hot-cold threshold and group configuration



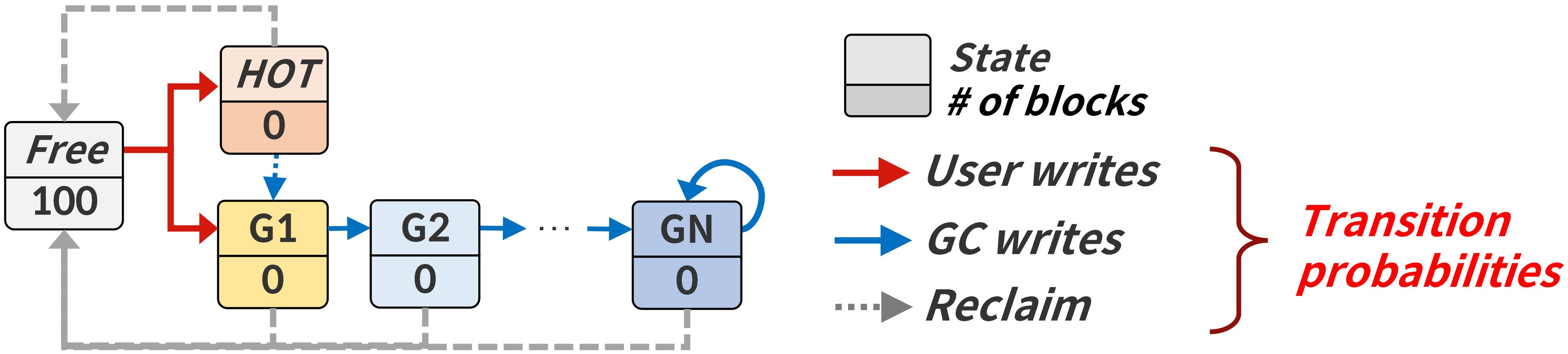
Index

- Log-Structured System and its problem
- Motivation
- **MiDAS Design**
  - Design Goal
  - Design Overview
    - Accurate invalidation time prediction
    - **Finding best group configuration**
      - **MCAM: Markov-Chain based Analytical Model**
      - **UID: Update Interval Distribution**
- Evaluation
- Conclusion

Markov Chain-based Analytical Model (MCAM)

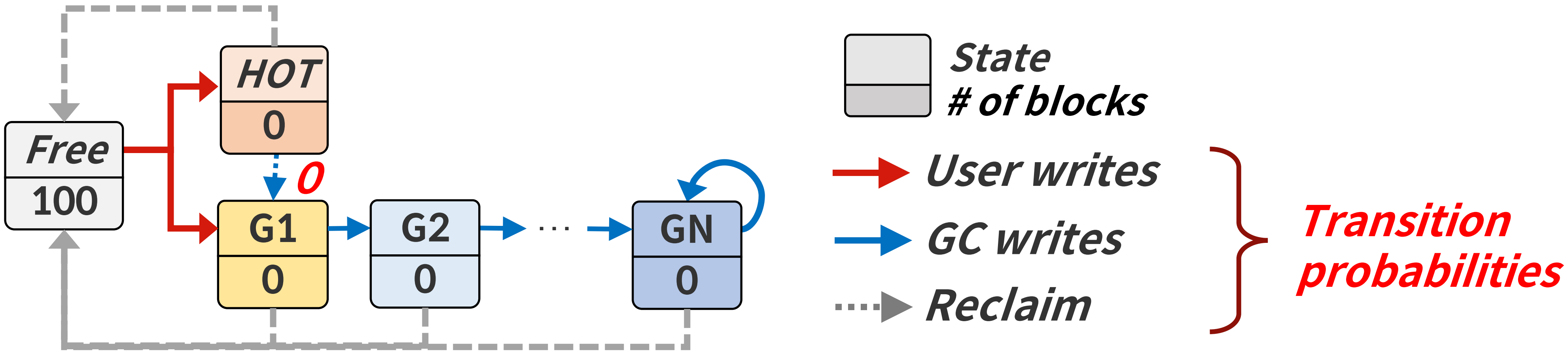
- Two main components in MCAM
  - States: Data blocks in MiDAS have one of two states: Free or HOT~GN
    - Free: Block is invalidated and reclaimed for future write
  - Transition probabilities: Movement ratios between those states

Can be obtained by using UID



WAF Prediction using MCAM

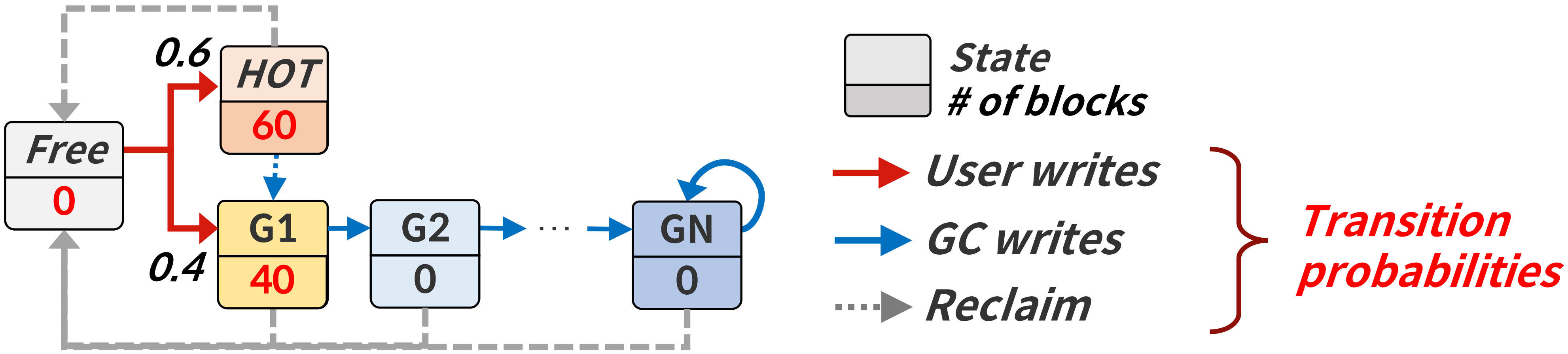
- Initial state
  - All blocks are *Free*
- Assumption
  - Blocks in HOT state are invalidated within *HOT* group
    - **Transition probability from HOT to G1 is 0**



WAF Prediction using MCAM (cont')

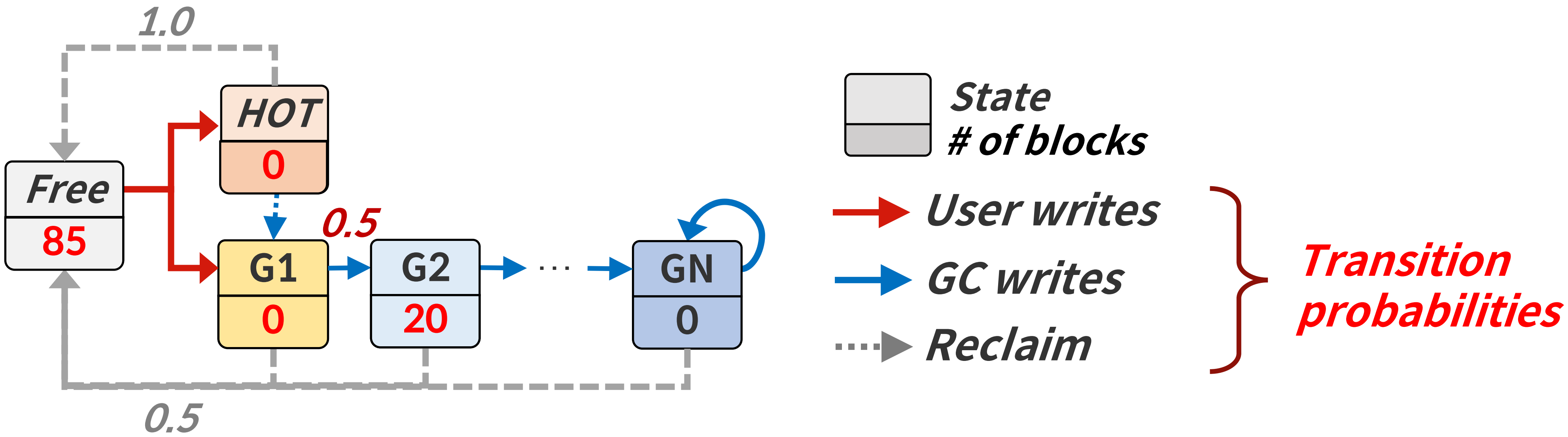
- Step update
  - Blocks are moved according to transition probabilities

Free blocks are moved according to transition probabilities from *Free to HOT* and from *Free* to G1



WAF Prediction using MCAM (cont')

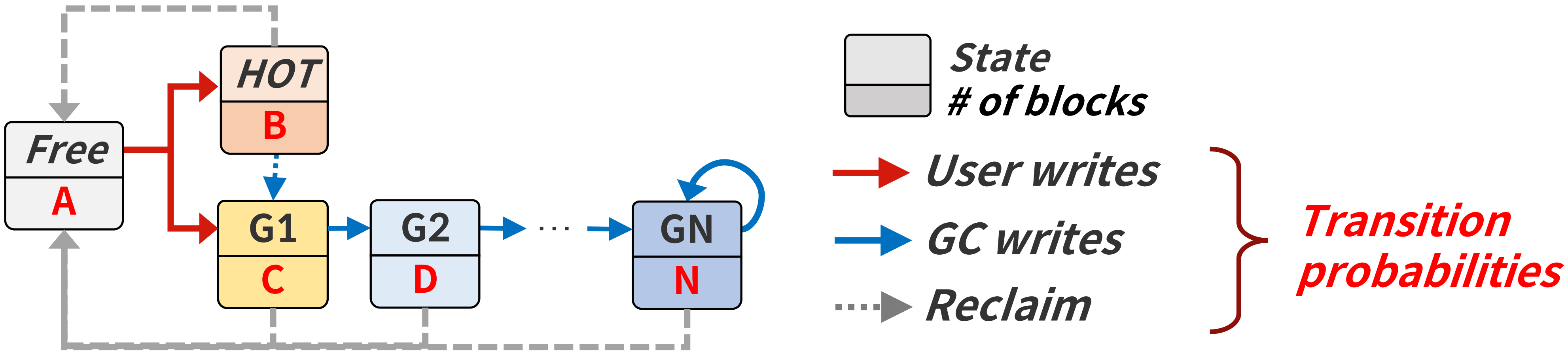
- Step update
  - Blocks are moved according to transition probabilities
  - Step update is repeated until number of blocks in state converges
    - Values (number of blocks) are guaranteed to be converged *



* [Markov chains: Gibbs fields, Monte Carlo simulation, and queues (Springer Science & Business Media, 2001)]

WAF Prediction using MCAM (cont’)

- Step update
  - Blocks are moved according to transition probabilities
  - Step update is repeated until number of blocks in state converges
    - Values (number of blocks) are guaranteed to be converged *



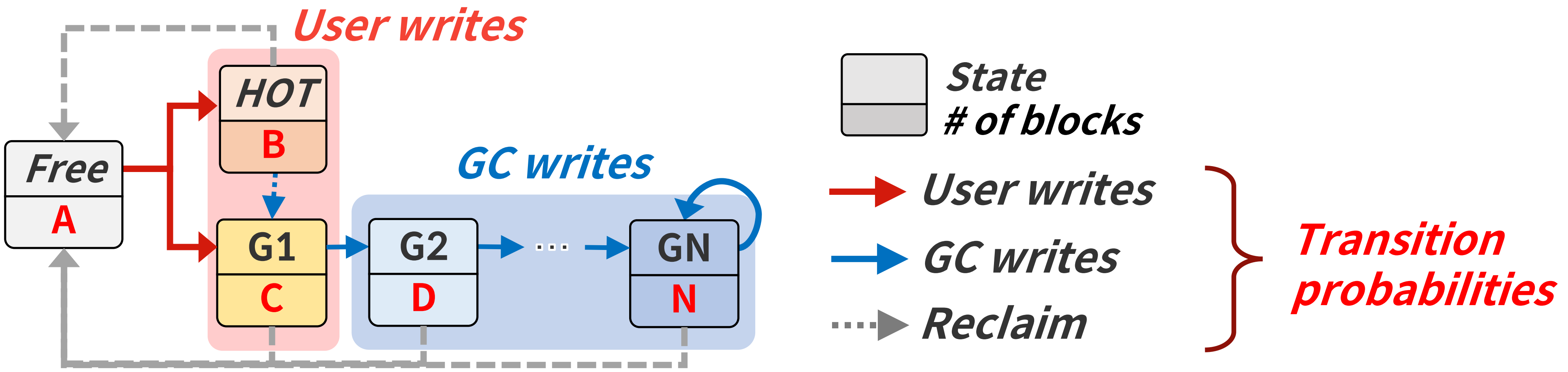
* [Markov chains: Gibbs fields, Monte Carlo simulation, and queues (Springer Science & Business Media, 2001)]

WAF Prediction using MCAM (cont')

- WAF prediction
  - Utilizing number of blocks for *HOT*~GN in final step

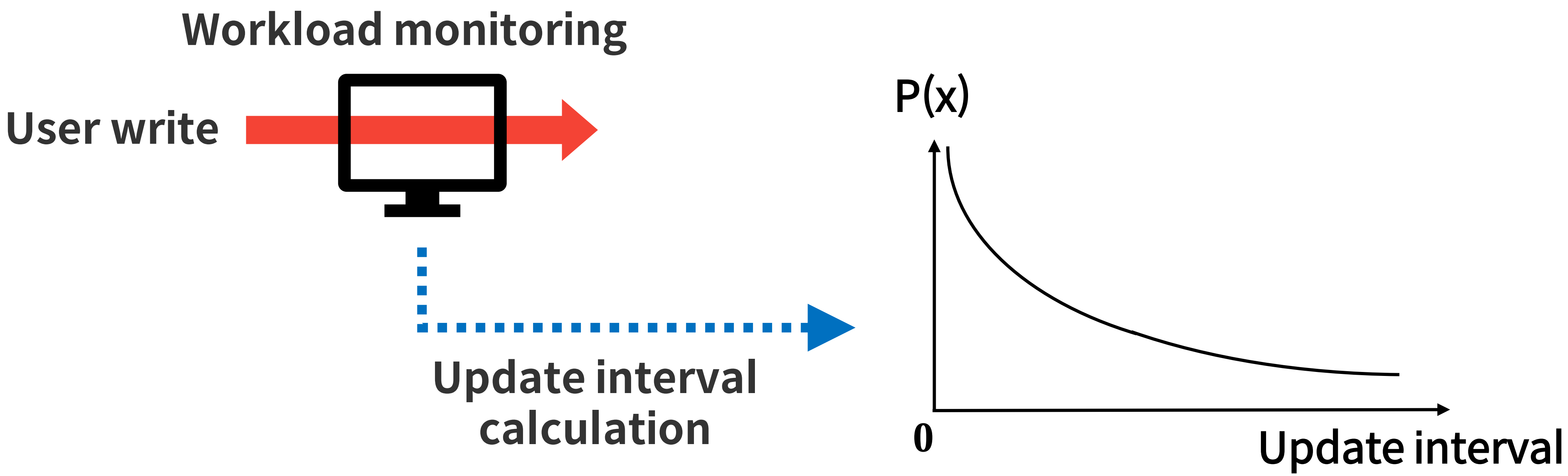
Remaining Issue: Obtaining transition probabilities using UID!

- Recall definition of WAF
  - $\mathbf{WAF} = \dfrac{\textit{User writes}+\textit{GC writes}}{\textit{User writes}} = \dfrac{(B+C)+(D+\cdots+N)}{B+C}$



Update Interval Distribution (UID)

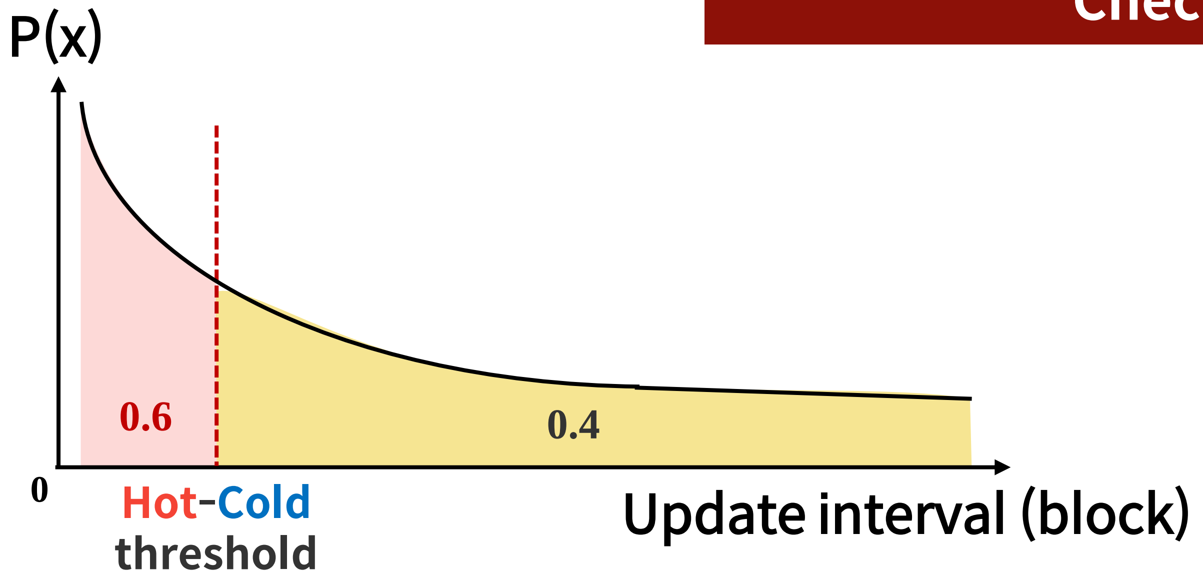
- UID is update interval distribution for incoming blocks into system
  - UID can be created by workload monitoring
- UID provides probabilistic statistics on what update interval blocks will have



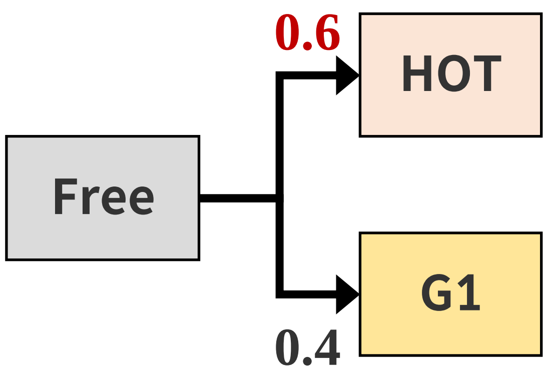
Predicting Transition Probabilities using UID

- For given group configuration and hot-cold threshold,
- Estimation of transition probabilities from Free to HOT and from Free to G1
  - Ratio of blocks with update interval shorter than hot-cold threshold is equal to transition probability from Free to HOT

Check our paper for details!



UID

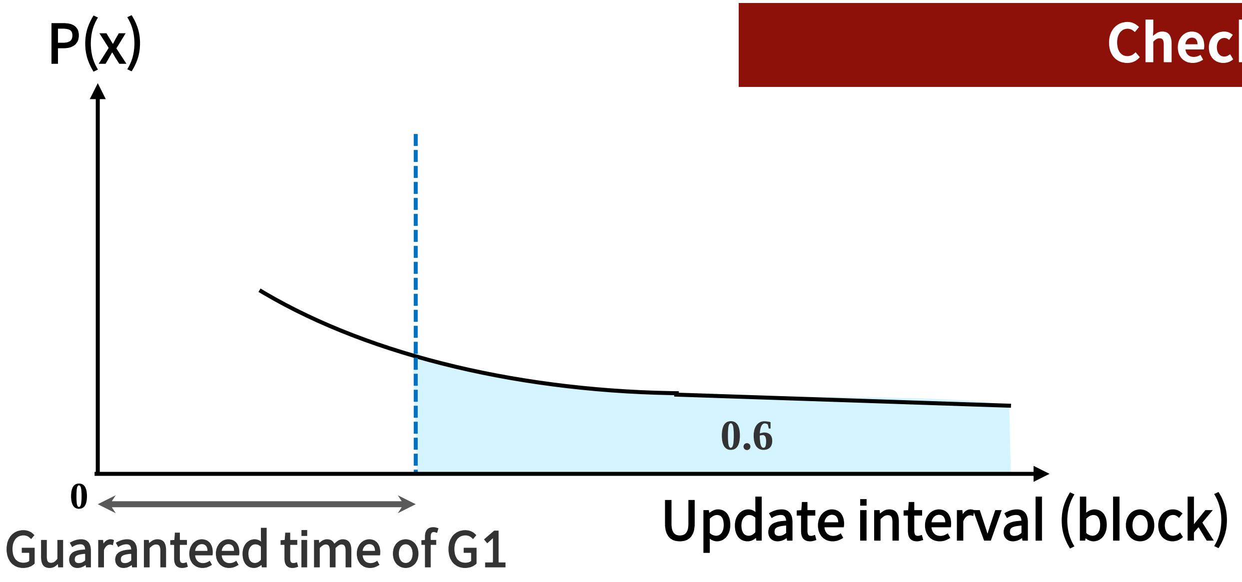


MCAM

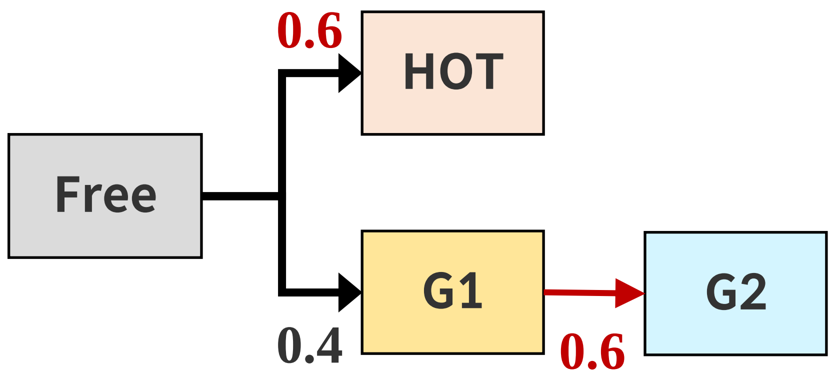
Predicting Transition Probabilities using UID (cont')

- For given group configuration and hot-cold threshold,
- Estimation of transition probability from G1 to G2
  - Ratio of blocks with update interval longer than guaranteed invalidation time of Gi is equal to transition probability from G1 to G2

Check our paper for details!



UID of G1



MCAM

Other Issues in MiDAS

- Finding best configuration
  - Evaluating all possible configurations is time-consuming
  - MiDAS adopts time-efficient algorithm to find configuration that provides sufficiently low WAF

- Updating group configuration *for changed workloads*
  - MiDAS may not work well when workload I/O patterns change rapidly
  - MiDAS performs periodic workload monitoring to deal with workload pattern change and irregular I/O patterns

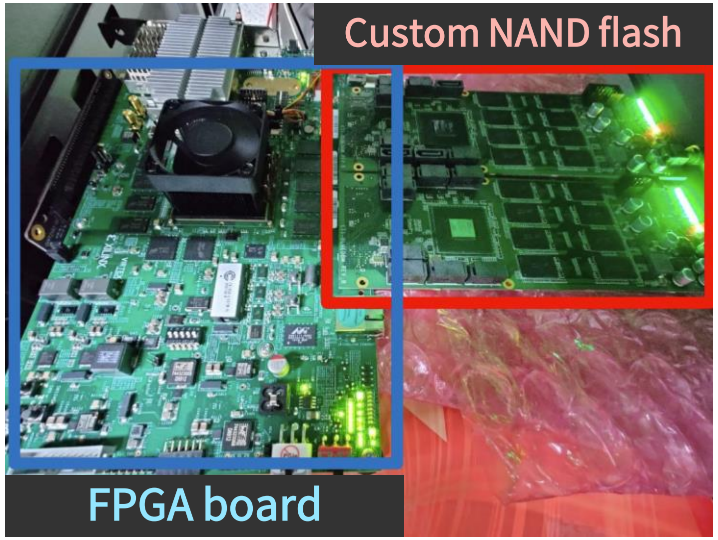
Check our paper for details!

Index

- Log-Structured System and its problem
- Motivation
- MiDAS Design
- **Evaluation**
  - **Experimental Setup**
  - **Experimental Results**
- Conclusion

Experimental Setup

- Implemented on FPGA-based SSD prototype
  - 128GB capacity with 4KiB blocks and 64MiB segments
- Benchmark
  - Varmail of Filebench
  - YCSB-A and –F on MySQL
  - TPC-C on MySQL
  - Alibaba Cloud trace and Exchange of Microsoft Enterprise trace
- SOTA GC techniques
  - CAT, AutoStream, MiDA and SepBIT

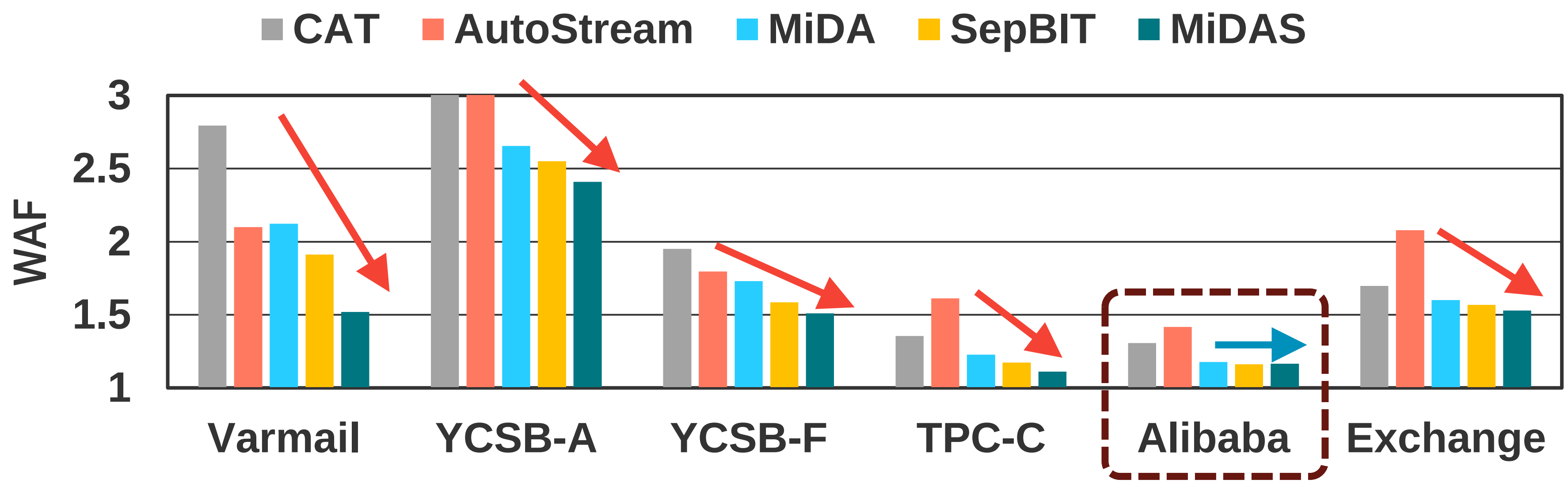


Objectives

- Does MiDAS effectively reduce WAF compared to other SOTA techniques?
- How does MiDAS reduce WAF?
- Does MiDAS work well with low-overhead?

WAF Analysis

- **25% reduced WAF** compared to other SOTA techniques
  - 16.5% reduced WAF compared to SepBIT
- No WAF reduction in Alibaba workload **For throughput, MiDAS is better**
  - Not enough time to collect workload information due to short trace file
  - Irregular I/O pattern in Alibaba workload

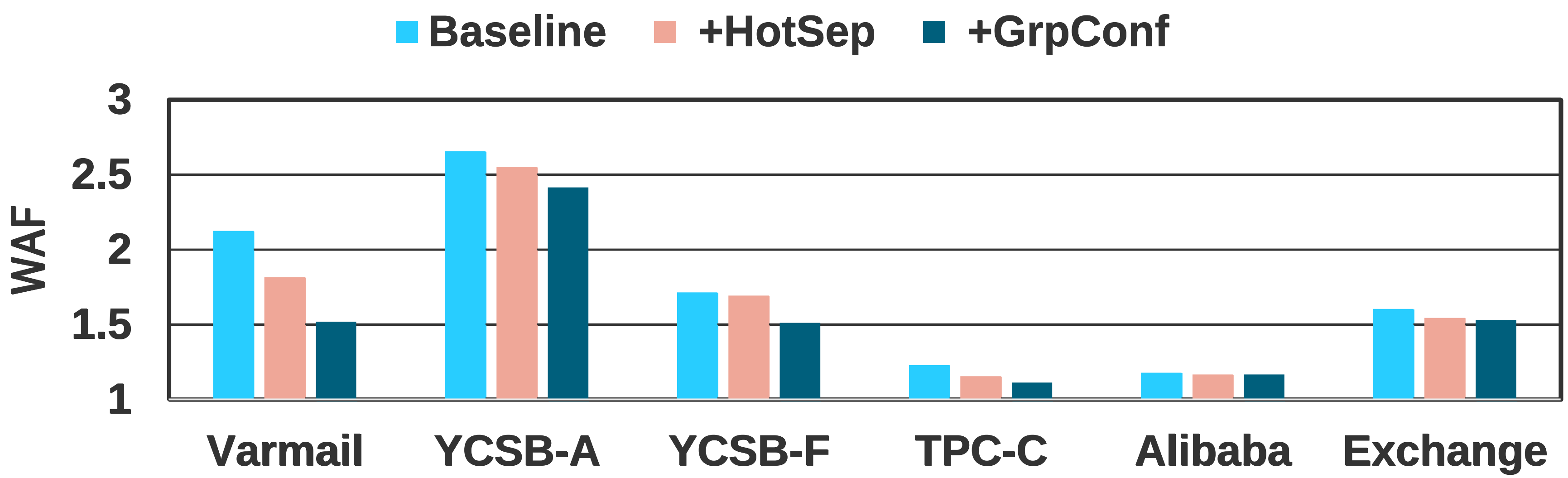


Overall WAF of each techniques

Impact of Each Design of MiDAS

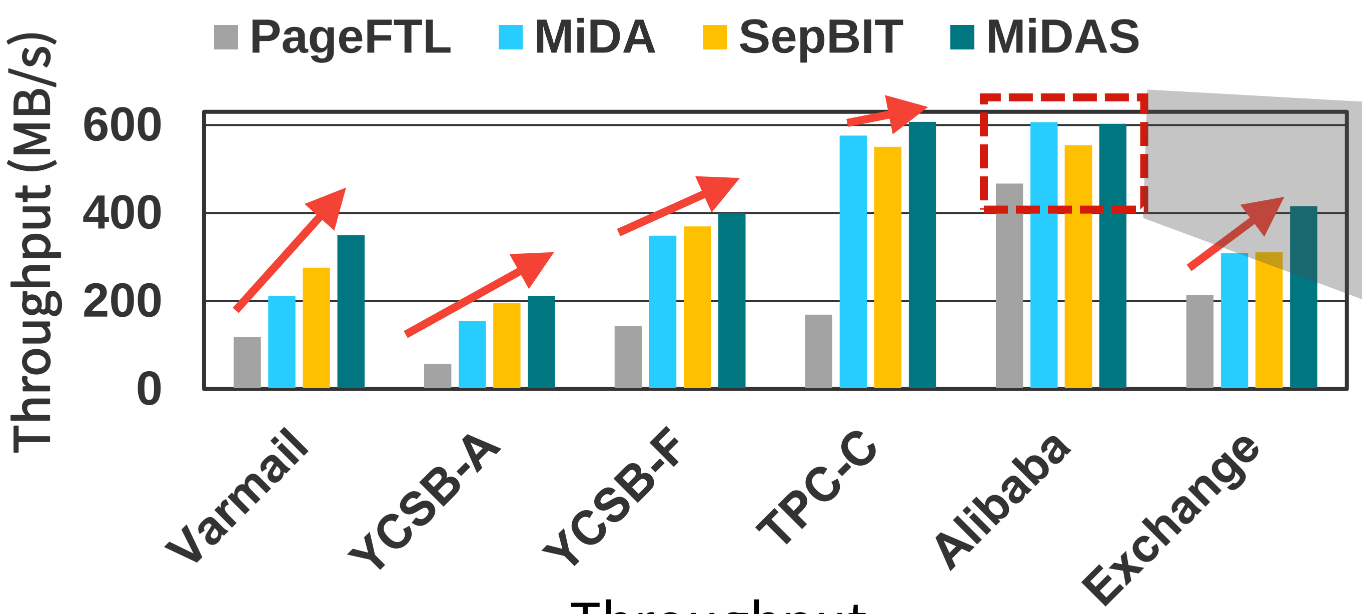
- Baseline: MiDA – data placement based on migration count (age)
  - +HotSep: Hot block separation using update interval for accurate prediction
  - +GrpConf: Applying best hot-cold threshold and group configuration

- **12% and 31% reduced WAF** for +HotSep and +GrpConf compared to baseline

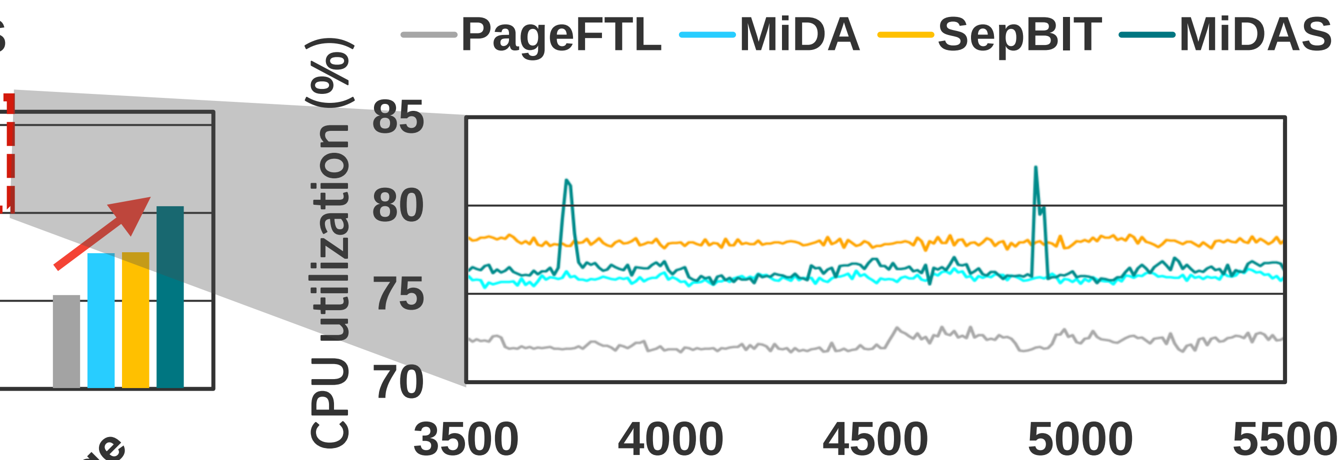


Throughput and Overhead Analysis

- 2.55x, 1.24x, and 1.15x higher throughput than PageFTL, MiDA, and SepBIT
  - Low GC overhead
  - Low computation overhead
    - 9% higher throughput than SepBIT in Alibaba workload
- Low CPU overhead to run MiDAS compared to SepBIT



Throughput



CPU utilization

Conclusion

- Limitation of existing techniques:
  - Inaccurate invalidation time prediction
  - Lack of consideration for number of groups and their sizes
- Solution: MiDAS mitigates GC overhead for log-structured systems by overcoming limitations of existing techniques
  - Employments of analytical models: UID and MCAM, to find best hot-cold threshold and group configuration
- Results
  - 25% reduced WAF
  - 54% improved throughput
  - Low overhead

Thank you

Seonggyun Oh (sungkyun123@dgist.ac.kr)