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

Seonggyun Oh, Jeeyun Kim, and Soyoung Han, DGIST; Jaeho Kim, Gyeongsang National University; Sungjin Lee, DGIST; Sam H. Noh, Virginia Tech

2024. 07. 24

Presentation by Suji Park, Seonju Koo

sujipark@dankook.ac.kr

pigeon99@dankook.ac.kr

DANKOOK UNIVERSITY

Dankook University System Software Laboratory

Contents

1. Background

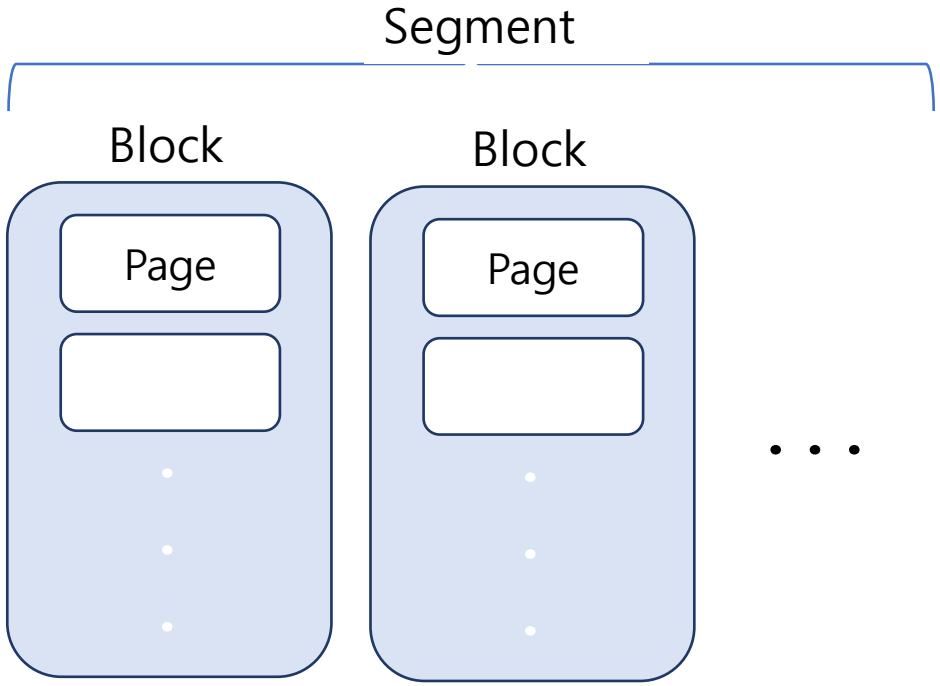
1.1 SSD

Solid State Drive (SSD)

- "Erase before write"

Log-structured System

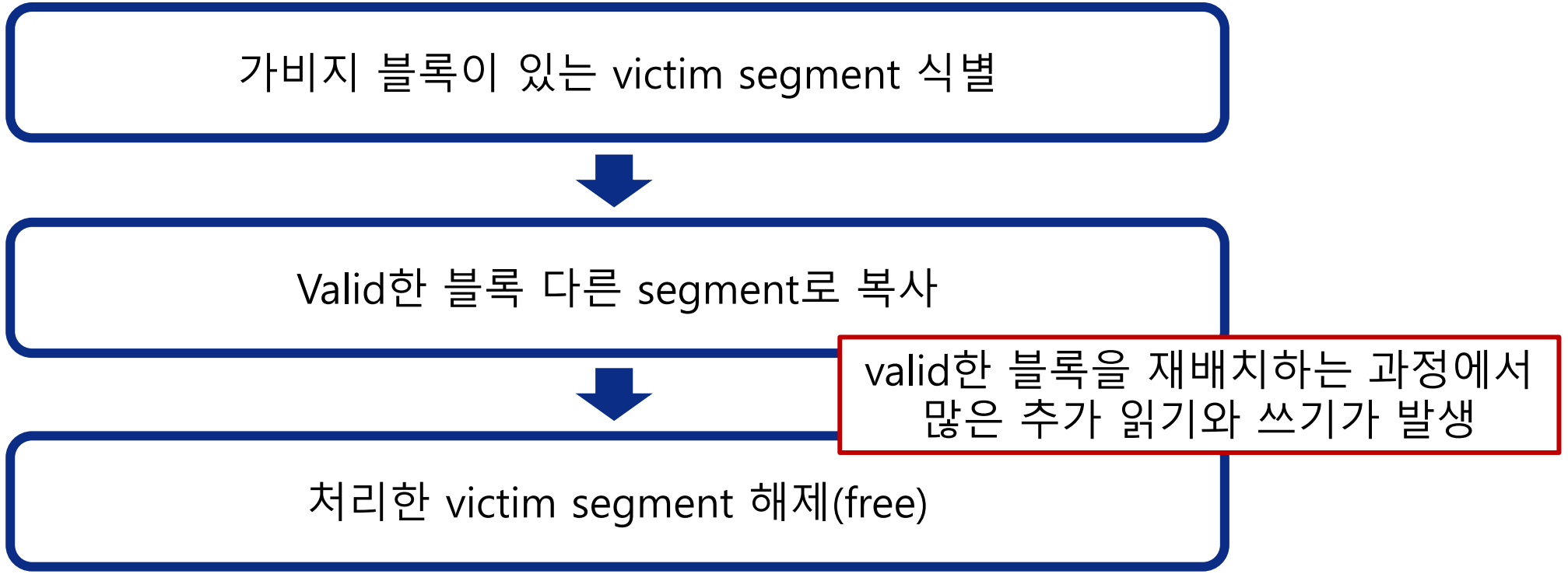
: Erase 작업 감소 -> 수명 연장



- Page : Read/Write 단위 (4KB)
- Block : Erase 단위 (MB)

1.2 Garbage Collection

Log-structured system의 Garbage Collection (GC)



1.3 WAF

- WAF(쓰기 증폭 인자) = 실제 storage에 쓴 block 수 / 사용자가 쓴 block 수

- 기존 연구 : victim selection과 data replacement 활용
  - 한계

    1. Hot, cold 구분
       - lifespan 예측 어려움
    2. 저장공간의 비효율적인 분할
       - 그룹 수 & 크기 지정

1.4 Victim Selection Policies

- **u** : valid 블록 복사 수
- age : 세그먼트 생성으로부터 얼마나 오래 살아있는지

- Goal : GC동안 live 블록 복사수를 최소화

1. **FIFO**
   - Oldest 세그먼트 선택

2. **Greedy**
   - Lowest utilization(u)의 세그먼트 선택

3. **Cost-Benefit (CB)**
   - utilization & age 이용
   - 각 세그먼트 점수 계산 점수 = $\frac{u}{age \times (1-u)}$
   - WAF, GC cost 감소
   - 구현 복잡

1.5 Data Replacement Policies

CAT	-**업데이트 빈도**에 따라 데이터 블록을 hot과 cold로 분류 -live block을 여러 그룹에 걸쳐 이동하여 그룹 크기를 **동적**으로 변경
AutoStream	-**업데이트 빈도**에 따라 분류 -> 결정된 group id가 포함된 데이터 블록을 SSD로 보냄
MiDA	-**segment group chain**으로 그룹끼리 연결됨 -GC의 희생자로 선택될 때까지 valid하면 **나이**가 +1인 다음그룹으로 이동하는 방식
SepBIT	-**최신 업데이트 간격 & 나이** 고려 -임계값 이용 -> 데이터 블록 할당되는 그룹 결정 -라이브 블록 재배치 할 때 블록 수명을 측정하여 적절한 GC 그룹으로 보냄

2. Motivation

2.1 Experimental Setup

YCSB
- Yahoo! Cloud Serving Benchmark

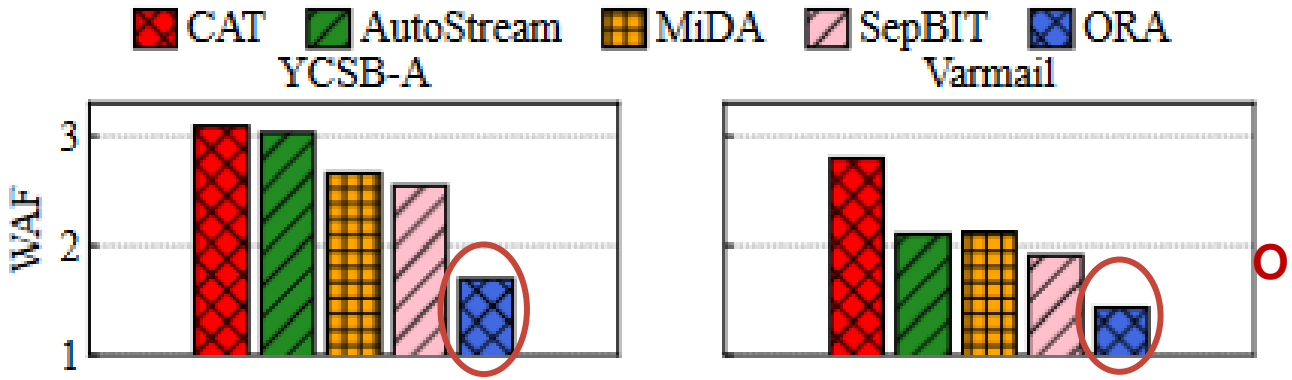
- Flash-based SSD
- YCSB-A runs on MySQL (44억 = 16.4TiB)
- Varmail of Filebench benchmark (40억 = 14.9TiB)
- 트레이스 기반 시뮬레이터

  - 64MiB 의 세그먼트가 있는 128GiB 스토리지 공간 모델링

- Victim Selection : CB

2.2 Analysis based on ORA

- Invalidation time : 블록이 기록된 시점과 무효화되는 시점 사이 기록된 사용자 작성 블록 수
- Invalidation time이 짧은 블록은 더 hot함



Oracle Algorithm (ORA)

Fig. 1. WAF of ORA and existing SOTA techniques

2.2 Analysis based on ORA

- (a) 최신 업데이트 간격 (SepBIT)
- (b) 업데이트 빈도 (CAT, AutoStream)
- (c) age 기반 (SepBIT, MiDA)

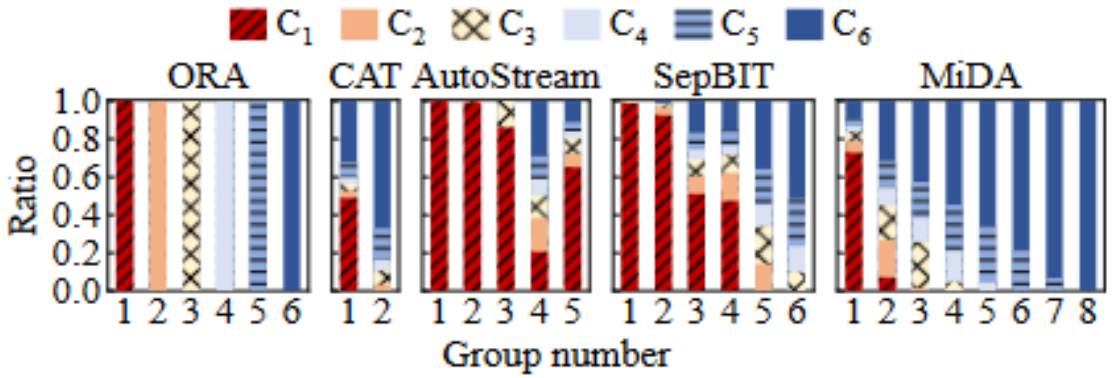


Fig. 4. Distribution of blocks over groups for YCSB-A

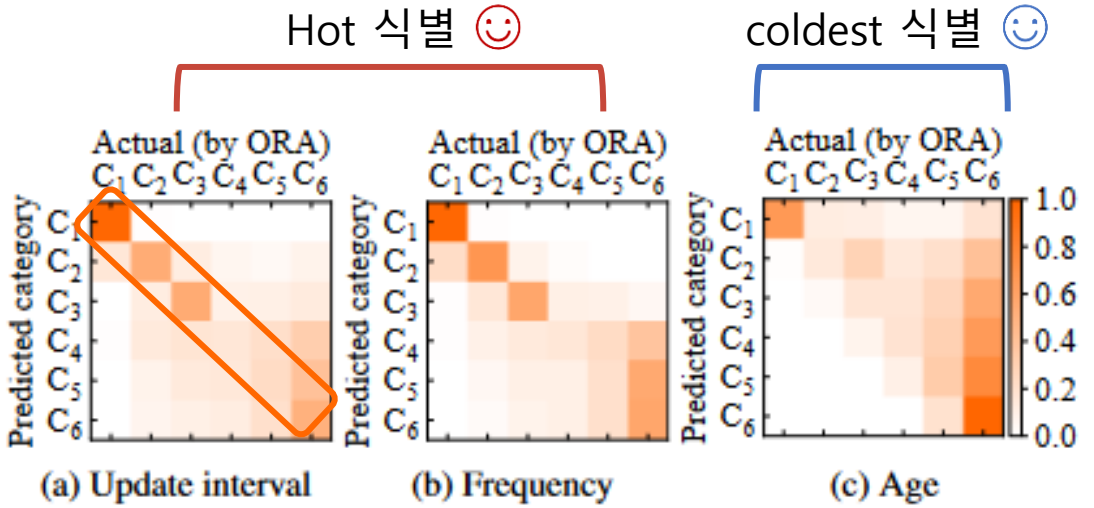


Fig. 2. Accuracy of block invalidation times for YCSB-A according to prediction techniques

Table 1: Ranges of invalidation times for C_1–C_6 in ORA

	C_1	C_2	C_3	C_4	C_5	C_6
Range	<250K	250K–5M	5M–14M	14M–28M	28M–62M	>62M

2.2 Analysis based on ORA

- Hotness를 정확하게 평가하여 적절한 그룹 사이즈를 선택하여 WAF의 최소를 선택

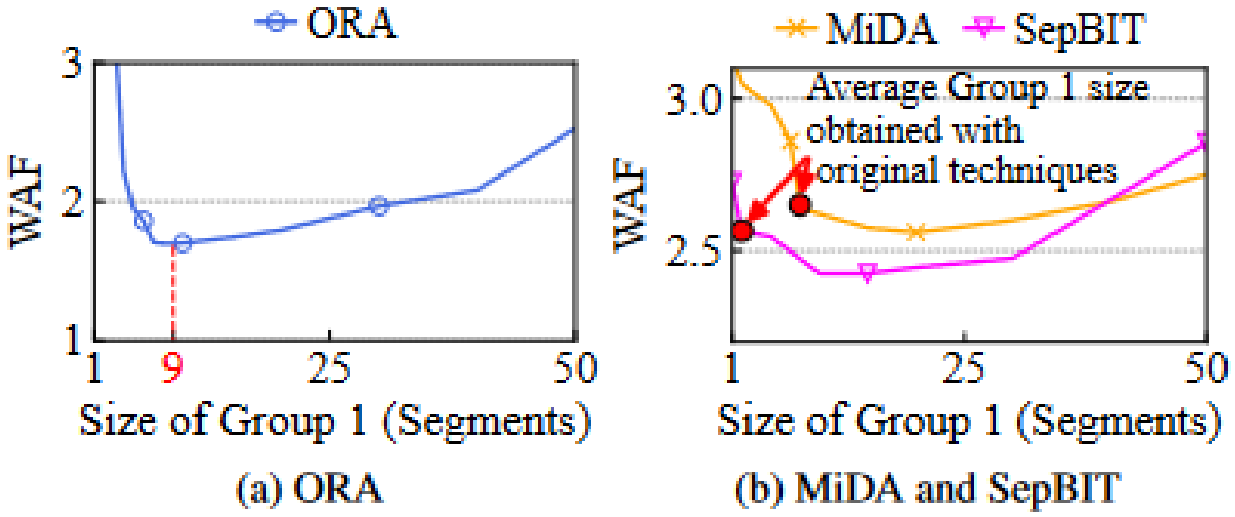


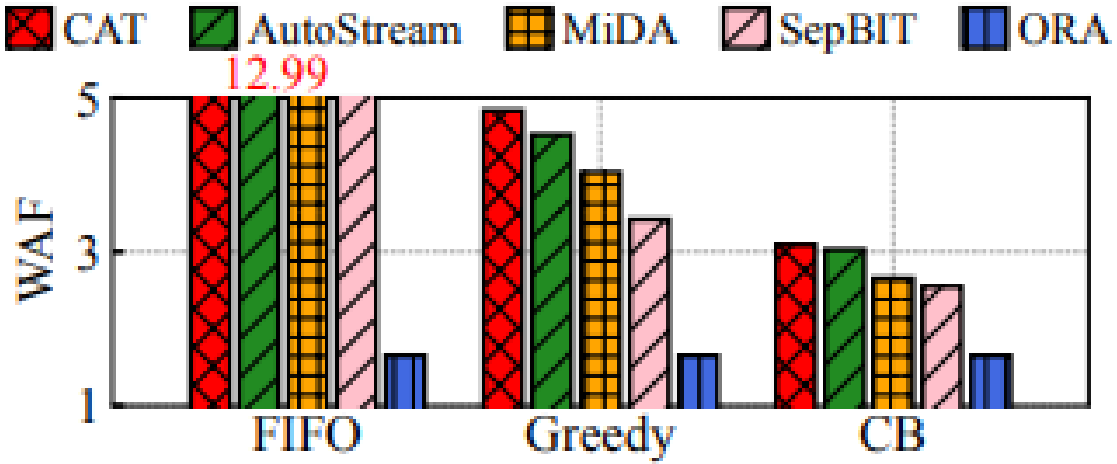
Fig. 3. Impact of group size on WAF for YCSB-A

Table 2: Size of each group per technique for YCBS-A (unit: segment): fixed values for ORA, while the rest are averages through workload processing

Group number	1	2	3	4	5	6	7	8
ORA	9	17	35	45	133	1,809	-	-
CAT	184	1,864	-	-	-	-	-	-
AutoStream	2	3	2	1,336	705	-	-	-
SepBIT	2	1	17	2	85	1,941	-	-
MiDA	7	79	95	126	128	117	98	1,398

2.3 Impact of victim selection

- ORA는 모든 정책에서 비슷한 WAF를 보여줌
  - 적절한 데이터 배치 정책을 사용하면 FIFO, Greedy, CB 모두 비슷함



2.4 Summary

- 현재 SOTA 기술은 데이터의 hotness 예측에 부정확
  - 그러나 최신 업데이트 간격 및 업데이트 빈도 기반 접근 방식은 hot 데이터를 비교적 잘 예측함, 연령 기반 예측 접근 방식은 cold 데이터를 비교적 잘 예측 -> 모든 methods를 적절히 이용하여 예측 정확도를 높여야 함

- 세그먼트 그룹 수와 크기는 GC 효율에서 중요한 영향 미침
  - WAF를 최소화하는 그룹 수와 크기 값을 적절히 찾아야 함

- FIFO는 그룹 크기가 적절하게 설정된 경우 정교한 정책과 마찬가지로 효율적
  - 적절한 데이터 배치 프레임워크를 사용하면 FIFO가 victim selection 정책으로 충분

3. Design

3.1 Overview of MiDAS

Age : 한 그룹에서 다른 그룹으로의 이동 횟수

- MiDAS: 주어진 워크로드에 따라 WAF를 최소화하기 위해 segment 그룹 수 와 크기를 자동으로 결정하는 기술
- SepBIT의 G1~3이 공간이 작아 발생했던 overflow 방지 위해 MiDAS는 HOT을 동적으로 조정

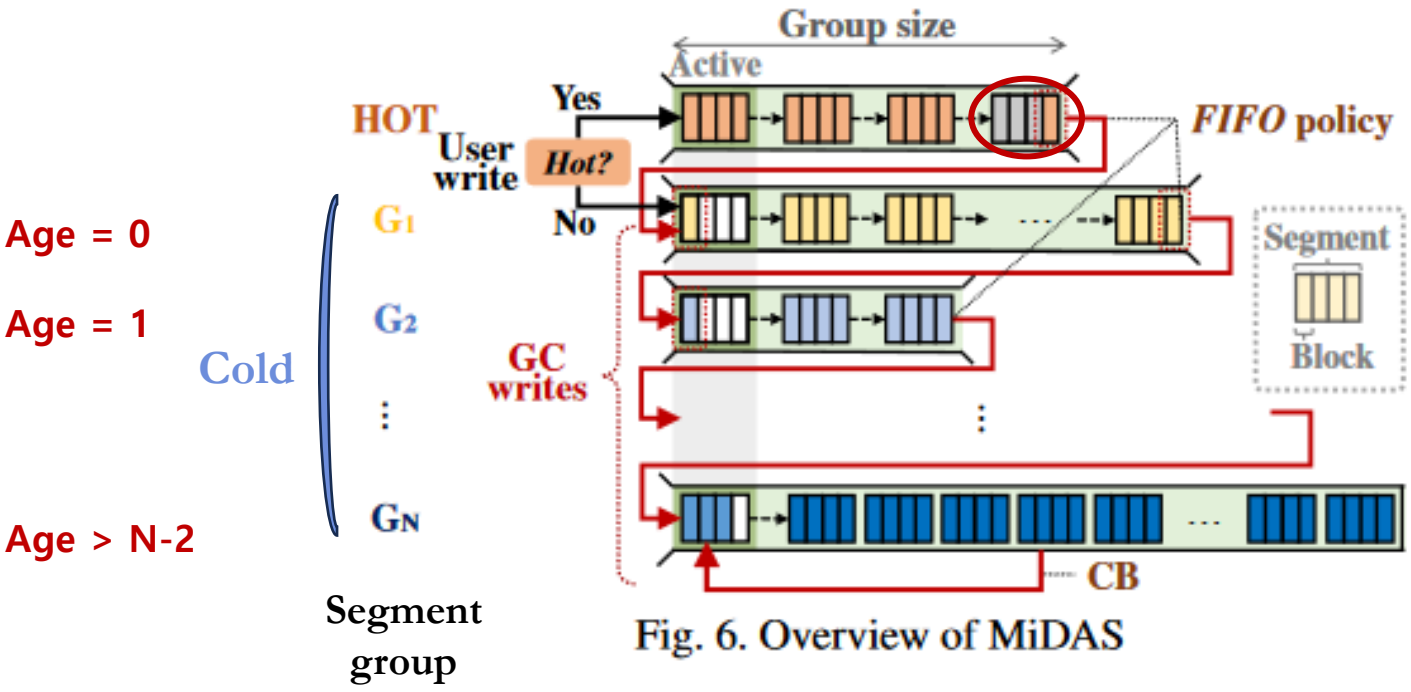


Fig. 6. Overview of MiDAS

3.1 Overview of MiDAS

Update Interval Distribution(UID)

한 그룹 안에서 몇개의 block이 valid하여 다음 그룹으로 이동되는지를 알려줌

+

Markov-Chain-Based-Analytical-Model(MCAM)

주어진 특정 그룹 구조에 따라 정확한 WAF 값 예측

전체 WAF 최소화 하는 그룹 수와 크기의 가장 효과적인 조합 결정

seg그룹이 제거 전 block이 무효화 되도록 충분히 큰 공간 가짐

⇨ **FIFO 사용**

3.2 Hot Block Separation

- CAT & AutoStream
  - update 빈도(횟수) 기반으로 hot-cold 경계 정의

- SepBIT
  - 내부 큐에 user 작성 블록 번호 넣고, 다시 참조된 블록은 HOT group으로 전송
  - (HOT group)평균 체류 시간에 비례하여 큐의 길이 설정
  - Cons
    ① 분류 잘되면 : CB 알고리즘으로 빠르게 제거 -> 평균 체류시간 감소 -> 큐 길이 짧아짐
    ② cold가 섞이면 : cold block이 평균 체류시간 증가시킴 -> 큐 길이 증가 -> HOT group에 더 많은 cold 생김

3.2 Hot Block Separation

MiDAS

- 1. Tight admission control
  - SepBIT에서의 평균 체류 시간을 참고하여 임계 시간을 설정하고, 이 시간을 기준으로 임계시간 내에 3번 이상 update 발생시 HOT group으로 이동

- 2. HOT group 크기 동적 조절
  - UID와 MCAM 이용하여 예상되는 WAF 예측 가능
  - 전체 WAF 최소화 되도록 HOT의 크기를 나머지 그룹과 함께 결정

3.3 Prediction of WAF using MCAM

- 주어진 그룹 구성, 상태 사이의 전이 확률등으로 WAF 값 예측
- 각 그룹의 블록 수가 변하지 않을 때까지 전이된 결과, Sn에서의 값을 이용함
- WAF = (User-write+GC write로 쓴 블록 수) / (User-write로 쓴 블록 수)

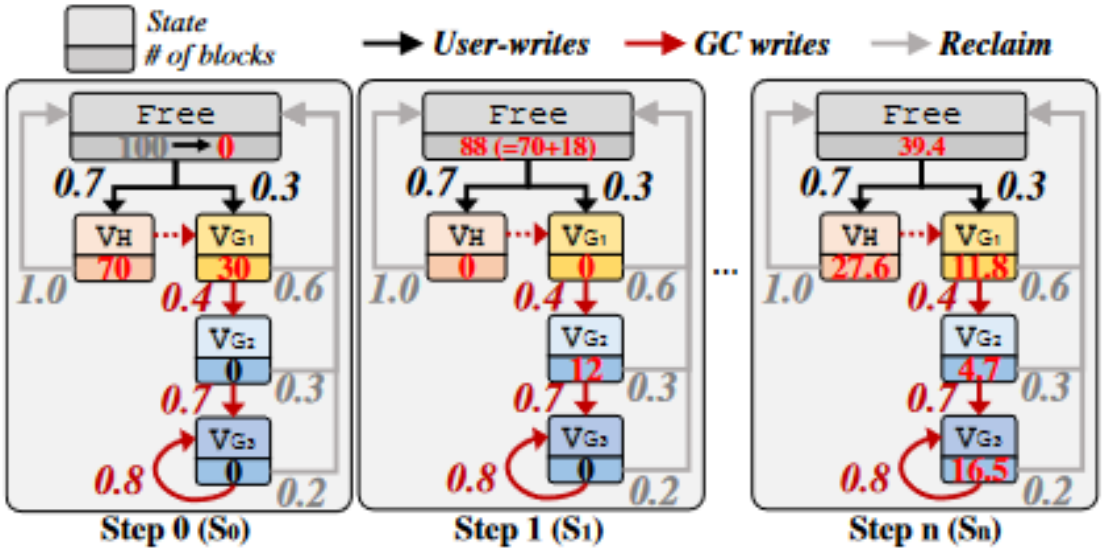


Fig. 7. Diagram of MCAM and WAF prediction process using MCAM as states transition

MCAM으로 WAF 예측

: 그룹 간 전이 확률 필요

-> UID 사용

3.4 Estimating Transition Probabilities

- UID: 사용자가 작성한 블록의 업데이트 간격에 대한 PMF(확률질량함수)
- UID를 사용하여 그룹 전이확률 추정
- 특정 간격 안의 P(x) 값을 모두 더하면 특정 그룹에서 invalid되는 블록 확률을 구할 수 있음
- 전이 확률 = G_{N+1} 의 invalid 확률 / (G_N 의 invalid 확률 + G_{N+1}의 invalid 확률)

$$T_{V_{G_i} \rightarrow V_{G_{i+1}}} = \frac{\sum_{j=W+W_{G_i}}^{max} p_j}{\sum_{j=W}^{max} p_j},$$

$$\text{where } W = \begin{cases} \sum_{k=1}^{i-1} W_{G_K} & \text{if } i > 1 \\ 0 & \text{otherwise.} \end{cases}$$

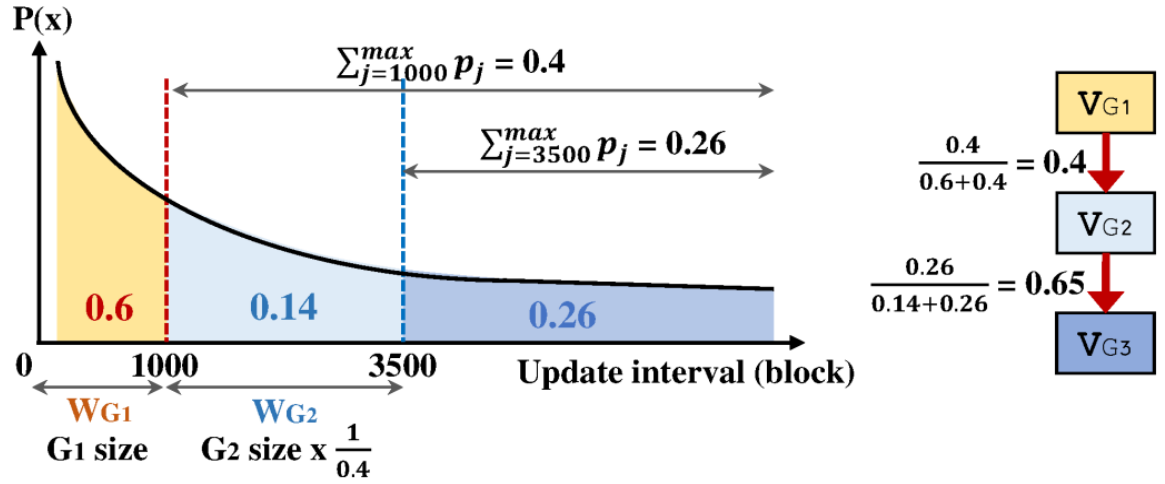


Fig. 8. Estimating transition probabilities using UID

3.4 Estimating Transition Probabilities

- HOT을 고려한 전이 확률 계산
- 임계시간보다 짧은 업데이트 간격 부분을 HOT UID로 분리
- 3번 업데이트를 고려하지 않아도 5% 이내 오차 -> 횟수를 고려하지 않고 계산한 근삿값 사용

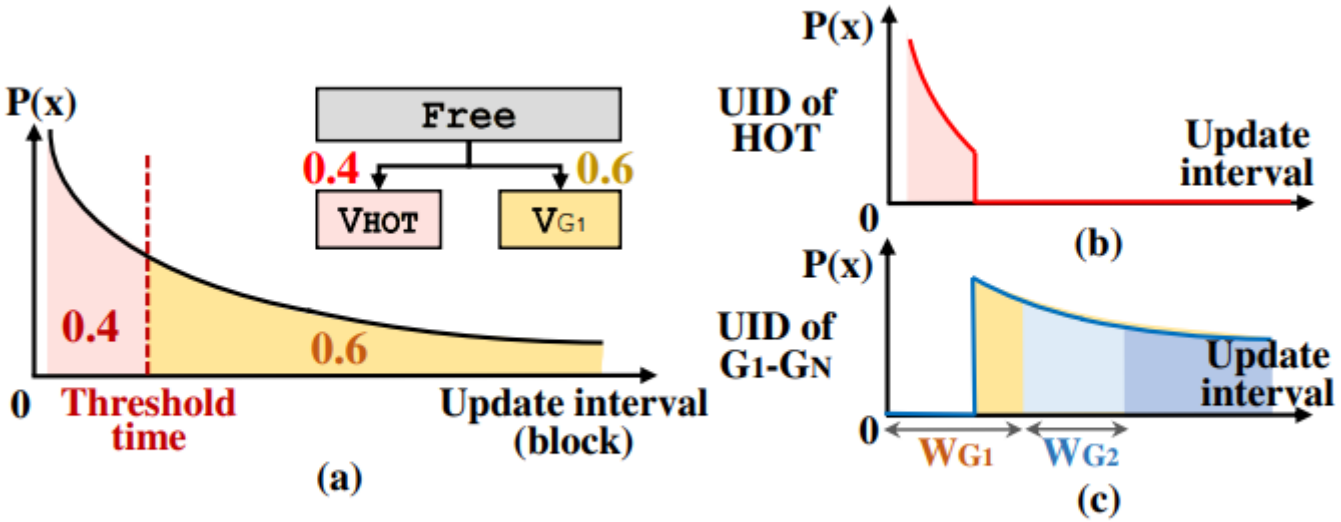


Fig. 9. Estimating transition probabilities, including *HOT*

G_N 전이 확률

: 블록 age 다양성으로 인해 Desnoyers 모델로 계산함

3.5 Configuring group with MCAM and UID

- 그룹 구성 선택(GCS) 알고리즘 사용
  1) **그룹 수, 크기** 결정
  2) 그룹 크기 세밀 **조정**
- 5회 연속 분할에서 WAF 감소가 0.5%를 넘지 않을 때까지

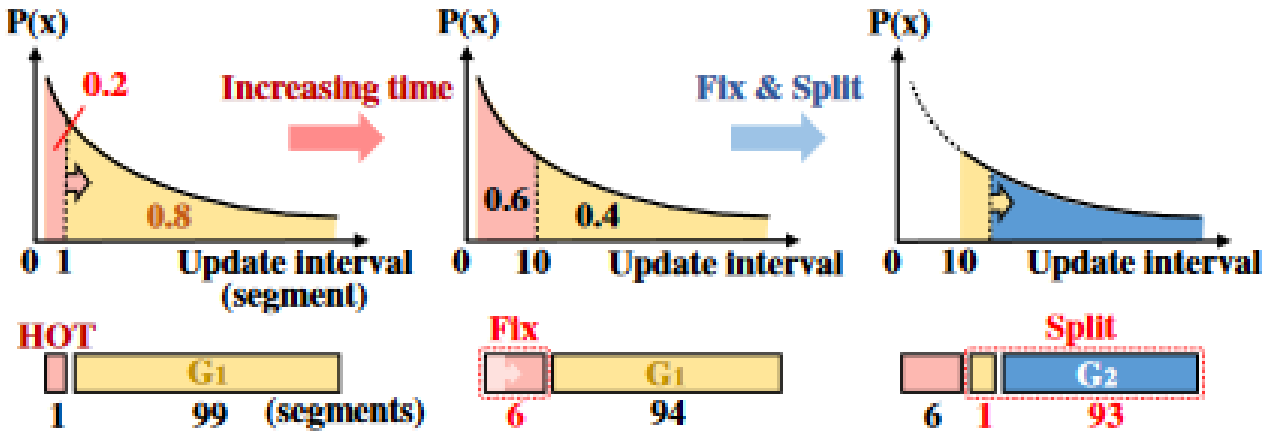


Fig. 10. Finding best group configuration with GCS

3.5 Configuring group with MCAM and UID

이전 그룹을 우선시하여 크기를 조정하면 마지막 그룹 G_N에서 높은 WAF 발생

해결

1. HOT~G_{N-1}에서 각각 G_N에 세그먼트 하나씩을 재할당하며 WAF 측정
2. WAF가 이전보다 크면 이동한 세그먼트는 제자리로 돌아감
3. WAF 감소 없을 때까지 반복

3.5 Configuring group with MCAM and UID

- 워크로드 변화에 따라 UID도 계속 변화
- WAF 악화를 피하기위해 MiDAS는 UID를 업데이트
- 동일한 길이의 epoch로 시간을 나누고, 해당 간격마다 UID 측정
- 이전 UID와 비교하여 차이가 5%를 초과하면 UID를 교체
- 새로운 UID 기반으로 그룹 재구성

3.5 Configuring group with MCAM and UID

불규칙한 I/O 패턴 처리

- 전이 확률의 예측 값과 실제 값을 비교하여 오차 계산
- 높은 오차율(10%초과)이 발견된 그룹부터 그 이후의 그룹을 하나로 병합
- 새 UID가 생성되기 전까지 MiDA 정책 사용

4. Implement

4. Implement

- 메모리 오버헤드 최적화

MiDAS 메인 구조

- FIFO 큐로 구성된 Cold 그룹이 Linked-List로 관리됨
- 세그먼트 이동 시 데이터가 직접 이동하지 않고 포인터 위치만 바꿔 줌

MCAM

- 전이 확률, 블록 수 같은 몇 가지 매개변수만 유지하면 됨

적은 메모리 소비

HOT, UID

- 다양한 데이터 구조를 유지해야 함 → 최적화 필요

4. Implement

- 메모리 오버헤드 최적화

HOT Filter	• 데이터 블록의 HOT 승격여부 관리하기 위한 필터 • 블록 당 2비트 크기 • (1TiB 당 60MiB 정도 메모리 차지)

  - UID 추정에 쓰임

Time Stamp Table	Interval Count Table
모든 블록의 갱신 간격 저장	특정 갱신 간격의 블록 수 저장 (단위: 블록)

4. Implement

- 메모리 오버헤드 최적화

HOT Filter	• 데이터 블록의 HOT 승격여부 관리하기 위한 필터 • 블록 당 2비트 크기 • (1TiB 당 60MiB 정도 메모리 차지)

  - UID 추정에 쓰임

Time Stamp Table	Interval Count Table
블록 100개 중에 1개 정보만 기록하는 방식으로 샘플링하여 최적화	갱신 간격의 범위를 넓혀 1-16K 범위면 동일 간격 가진 것으로 간주

4. Implement

- 실험 환경

Trace 기반 시뮬레이션, SSD 프로토타입 사용

(Free-space가 총 SSD 용량의 0.1%미만으로 떨어지면 GC 발생)

- SSD 프로토타입 스펙

: Quad Core ARM Cortex-A53, 256MiB DRAM, 256GiB Flash Array Card

4. Implement

- 워크로드 특징

- FIO-H: 데이터 접근 패턴이 특정구역 집중적
- FIO-M: 데이터 블록의 업데이트 간격이 유사함
- Varmail: 쓰기 집중적
- YCSB-A, F: 쓰기 집중적
- TPC-C: 많은 Hot, Warm 블록 포함함
- Alibaba: 짧고 집중적인 쓰기 패턴
- Exchange: 작은 워크로드로 구성, I/O 패턴이 불규칙적

Table 3: Characteristics of each workload

Workloads	FIO-H & -M	Varmail	YCSB-A & -F	TPC-C	Alibaba	Exchange
Notation	F-H & F-M	V	Y-A & Y-F	T-C	Ali	Ex
Write size (TiB)	15.1	14.9	16.4 & 18.0	16.1	Up to 2.8	Up to 2.9
Device size (GiB)	128				40-200	40
Request distribution	Zipfian (1.0 & 0.8)	Zipfian			-	-

5. Experiment

5.1 Comparison of GC Efficiency

- SOTA와 WAF 비교

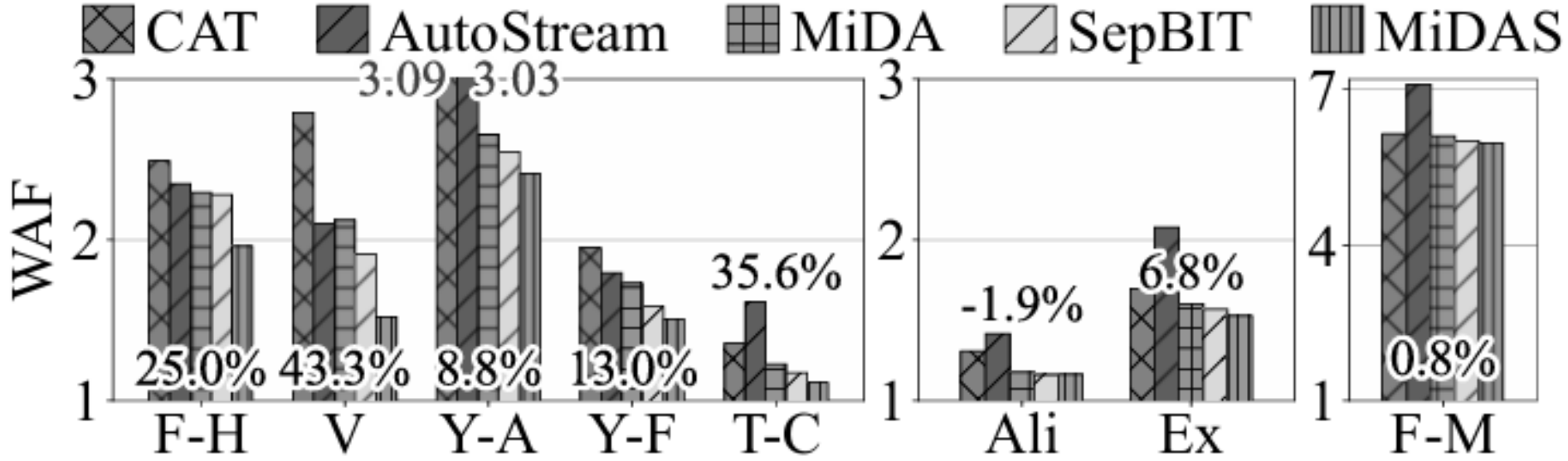


Fig. 11. Overall WAF of each technique (the numbers represent the improvement on WAF by MiDAS relative to SepBIT)

5.1 Comparison of GC Efficiency

- SOTA와 WAF 비교

SepBIT에 비해 개선된 WAF 양

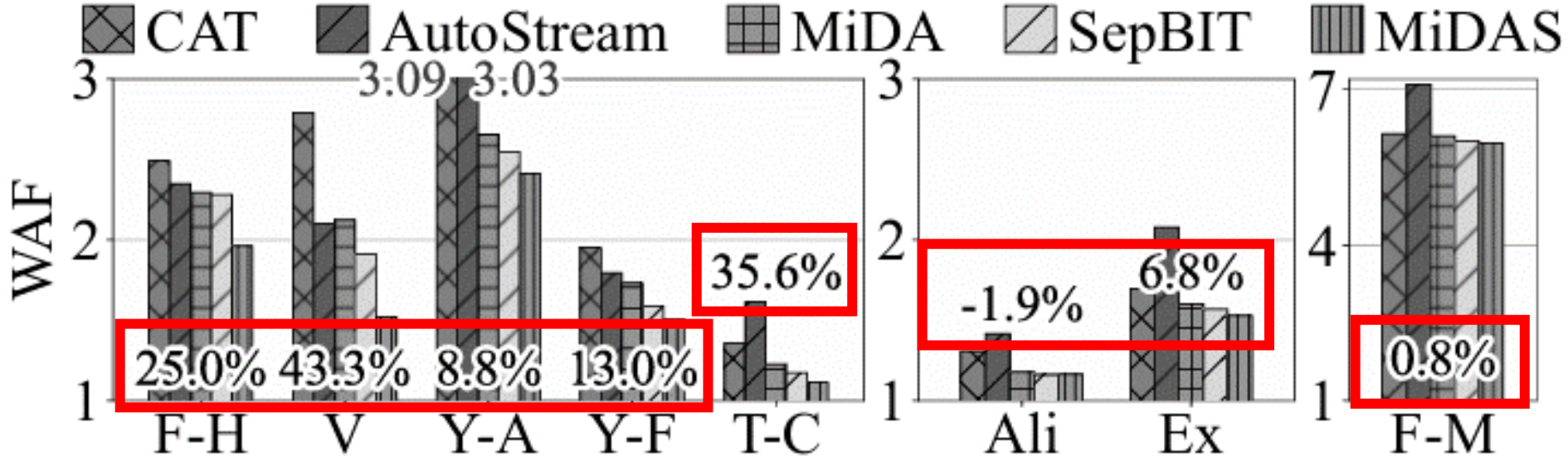


Fig. 11. Overall WAF of each technique (the numbers represent the improvement on WAF by MiDAS relative to SepBIT)

5.1 Comparison of GC Efficiency

- SOTA와 WAF 비교

SepBIT에 비해 개선된 WAF 양

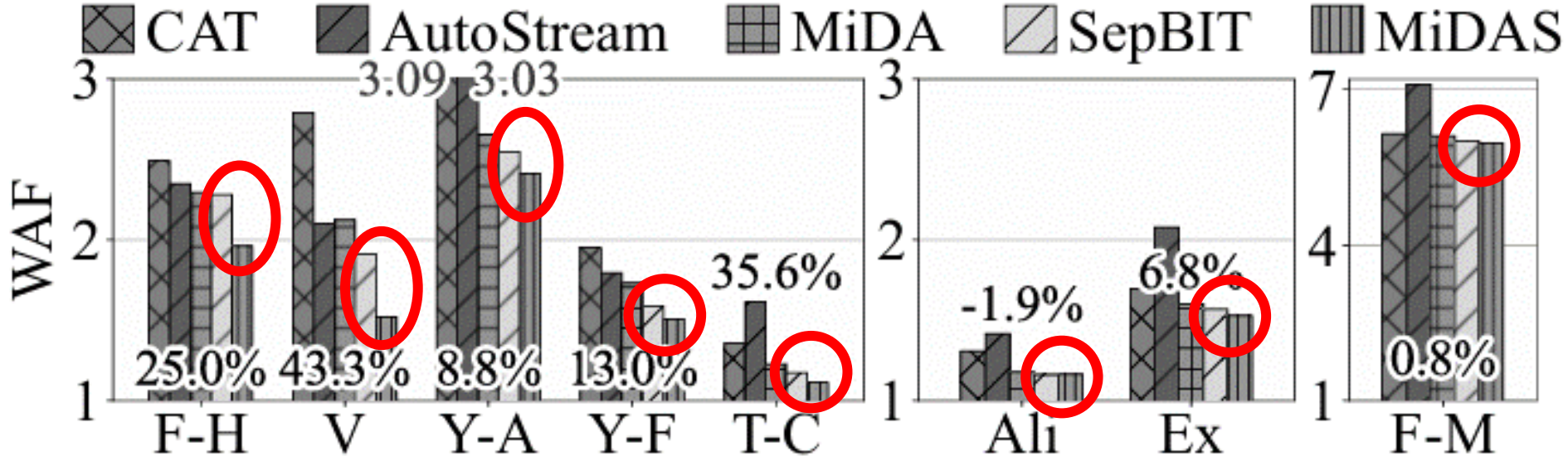
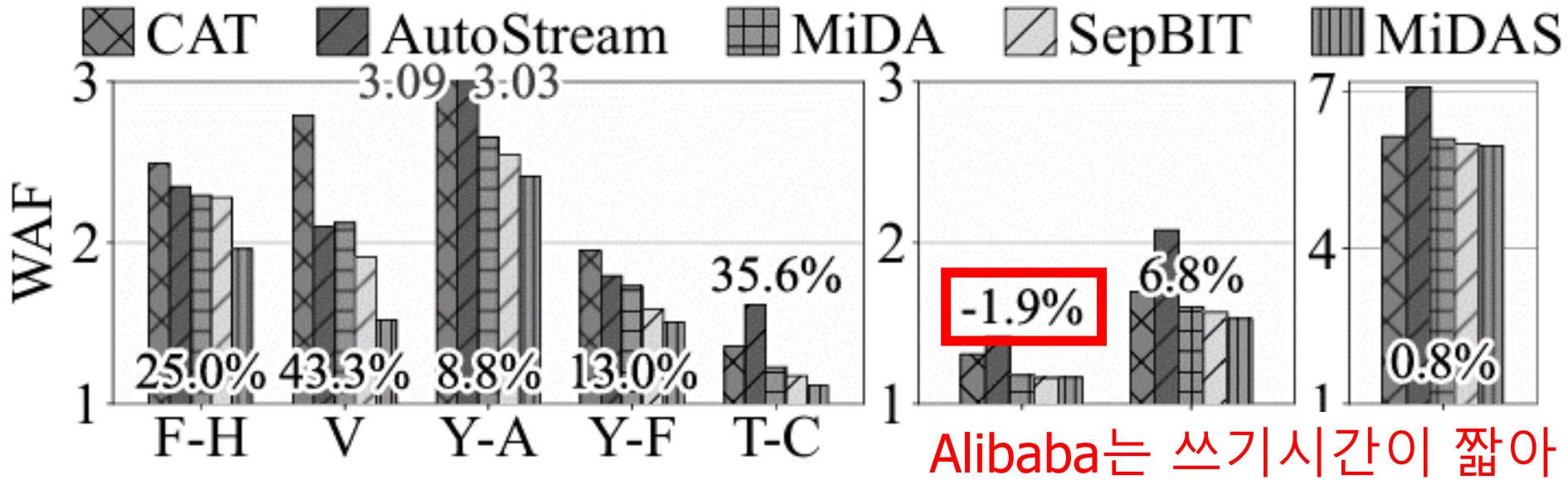


Fig. 11. Overall WAF of each technique (the numbers represent the improvement on WAF by MiDAS relative to SepBIT)

5.1 Comparison of GC Efficiency

- SOTA와 WAF 비교



Alibaba는 쓰기시간이 짧아 UID 혜택X

Fig. 11. Overall WAF of each technique (the numbers represent the improvement on WAF by MiDAS relative to SepBIT)

5.2 Impact of Each Component of MiDAS

- 원 기술 MiDA에 MiDAS만의 기술 HotSep, GrpConf, IrrHand를 차례로 적용하여 결과를 비교함

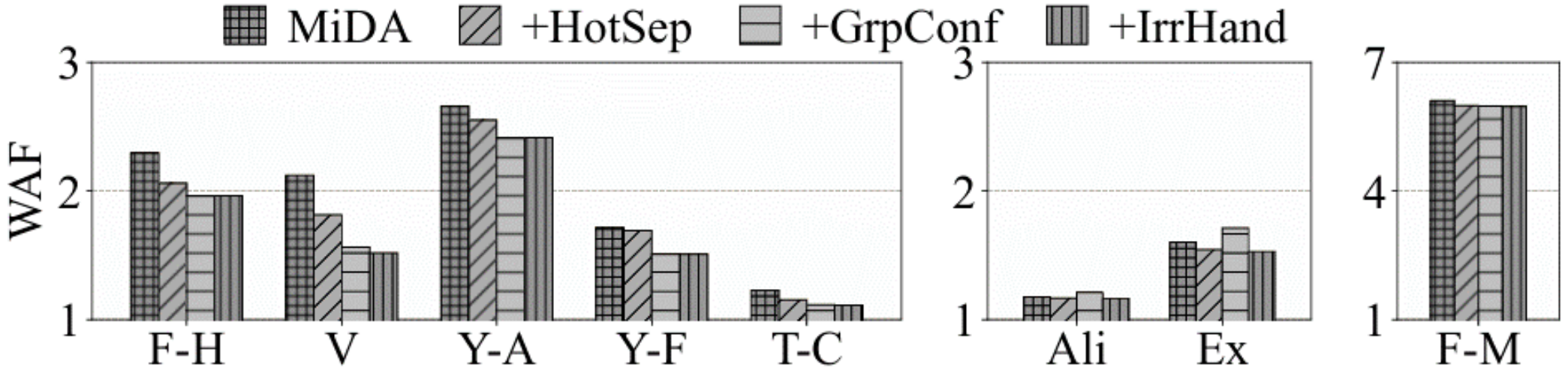


Fig. 12. Impact of individual components of MiDAS

5.2 Impact of Each Component of MiDAS

- MiDA에 HotSep, GrpConf, IrrHand 차례로 적용한 결과

변동성 거의 없는 워크로드

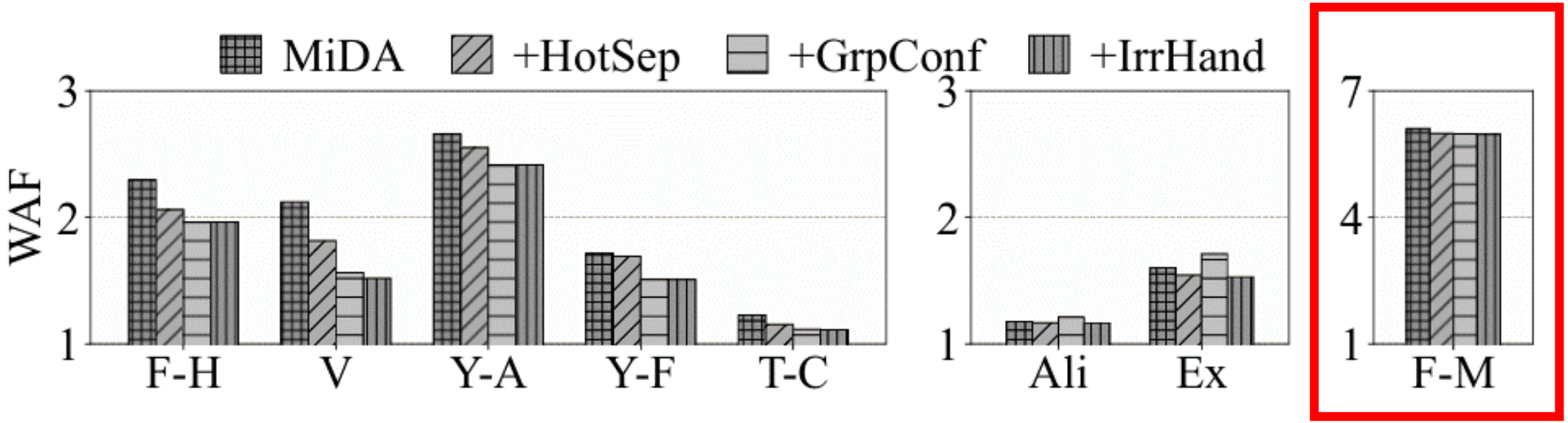


Fig. 12. Impact of individual components of MiDAS

5.2 Impact of Each Component of MiDAS

- MiDA에 HotSep, GrpConf, IrrHand 차례로 적용한 결과

일반적인 워크로드

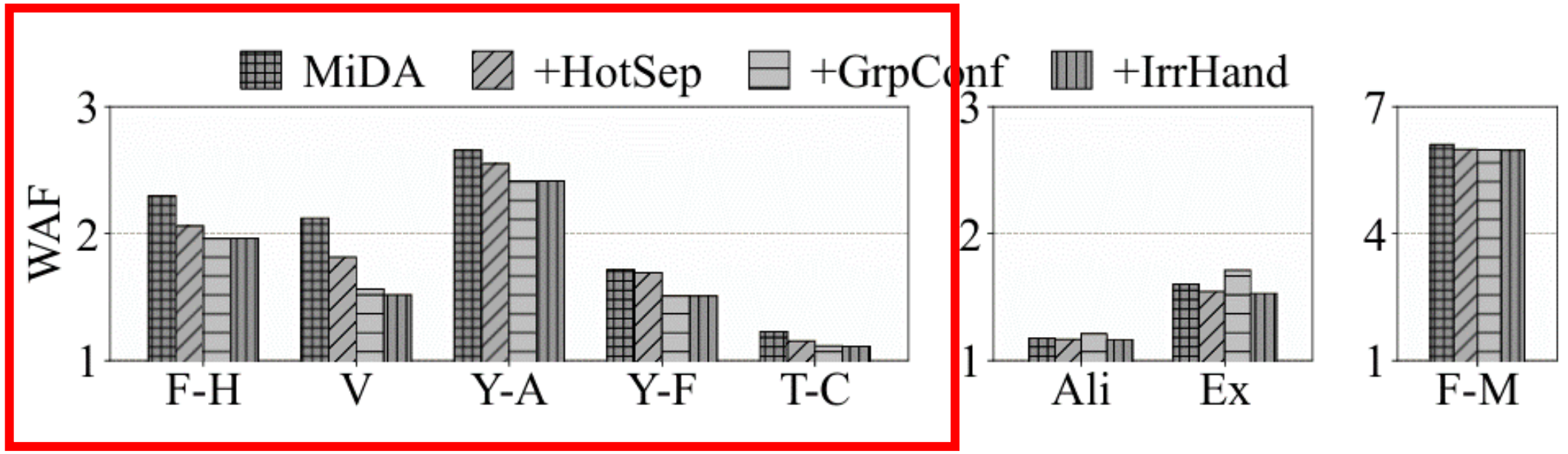
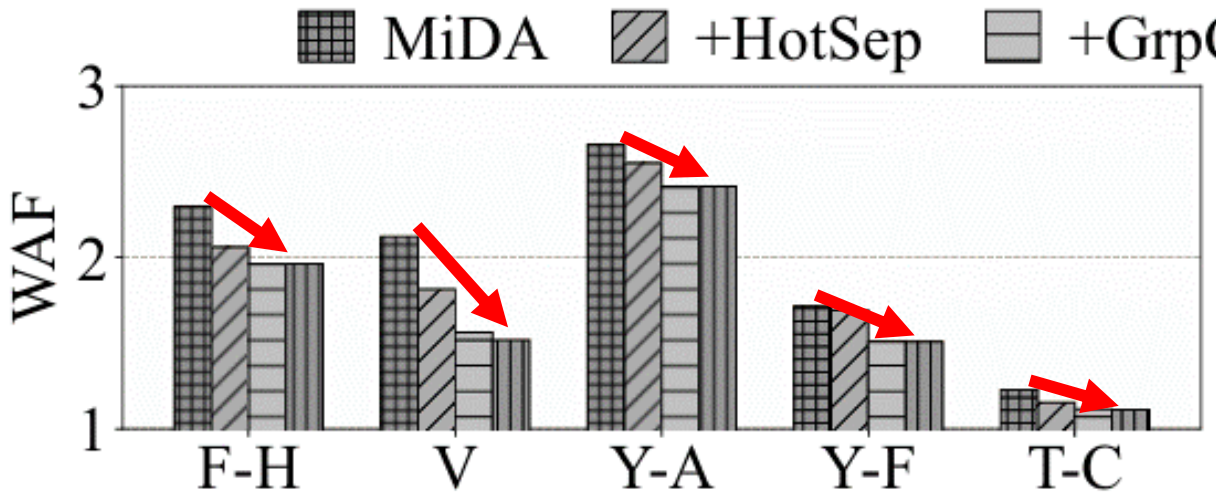


Fig. 12. Impact of individual components of MiDAS

5.2 Impact of Each Component of MiDAS

- MiDA에 HotSep, GrpConf, IrrHand 차례로 적용한 결과

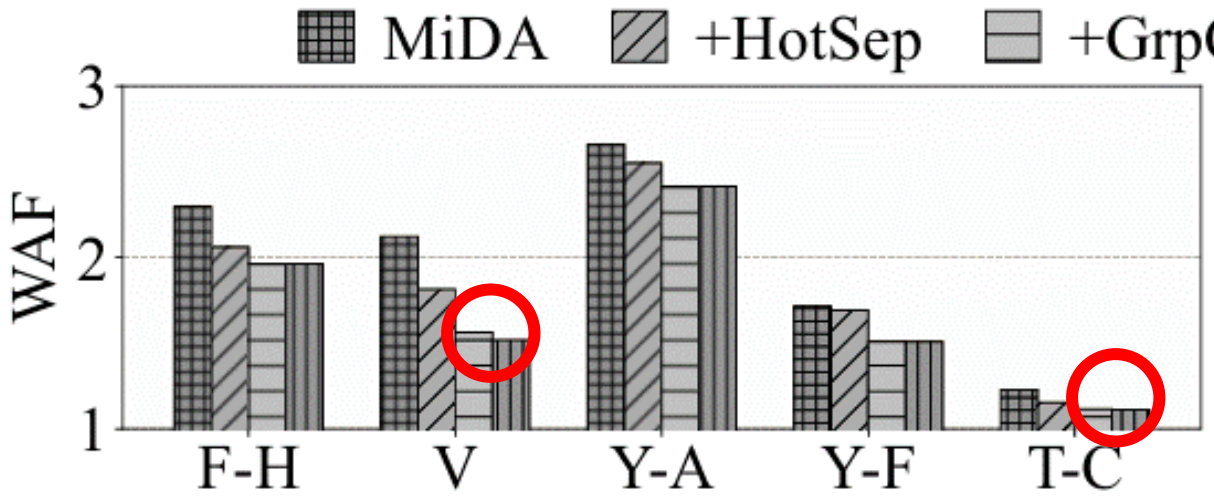


HotSep, GrpConf 모두 워크로드에서 WAF 감소 효과

Fig. 12. Impact of individual components of MiDAS

5.2 Impact of Each Component of MiDAS

- MiDA에 HotSep, GrpConf, IrrHand 차례로 적용한 결과



IrrHand는 약간의 불규칙성 포함한 Varmail, TPC-C에서만 작은 효과

Fig. 12. Impact of individual components of MiDAS

5.2 Impact of Each Component of MiDAS

- MiDA에 HotSep, GrpConf, IrrHand 차례로 적용한 결과

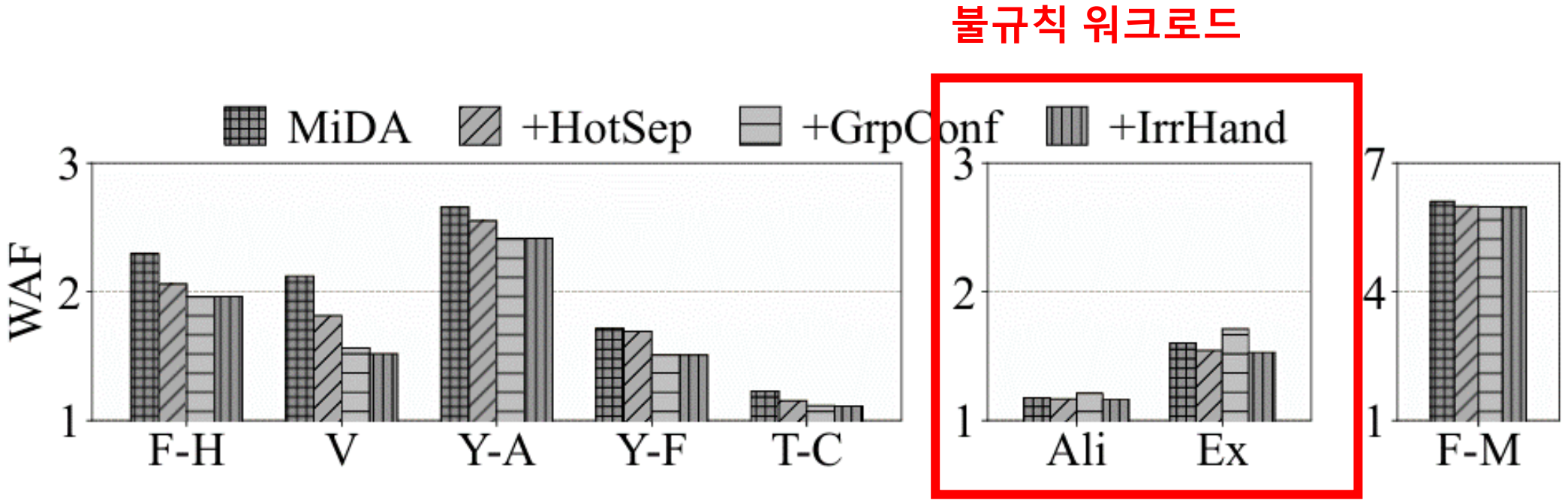


Fig. 12. Impact of individual components of MiDAS

5.2 Impact of Each Component of MiDAS

- MiDA에 HotSep, GrpConf, IrrHand 차례로 적용한 결과

HotSep: 효과 미미

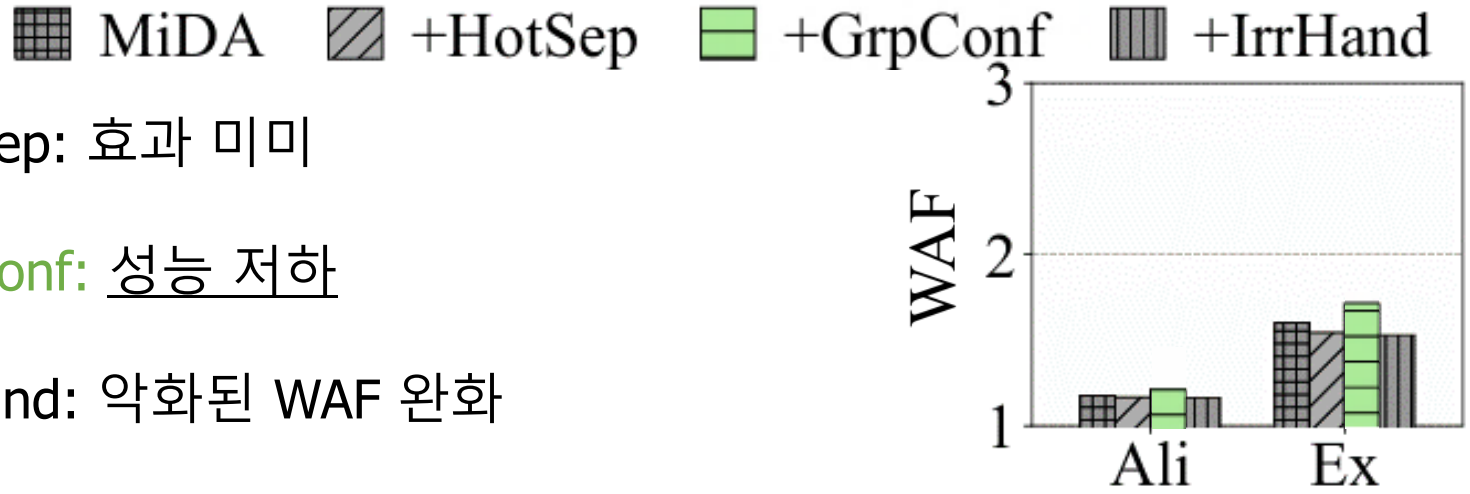
GrpConf: 성능 저하

IrrHand: 악화된 WAF 완화

Fig. 12. Impact of individual components of MiDAS

5.2 Impact of Each Component of MiDAS

- MiDA에 HotSep, GrpConf, IrrHand 차례로 적용한 결과



HotSep: 효과 미미

GrpConf: 성능 저하

IrrHand: 악화된 WAF 완화

Fig. 12. Impact of individual components of MiDAS

5.2 Impact of Each Component of MiDAS

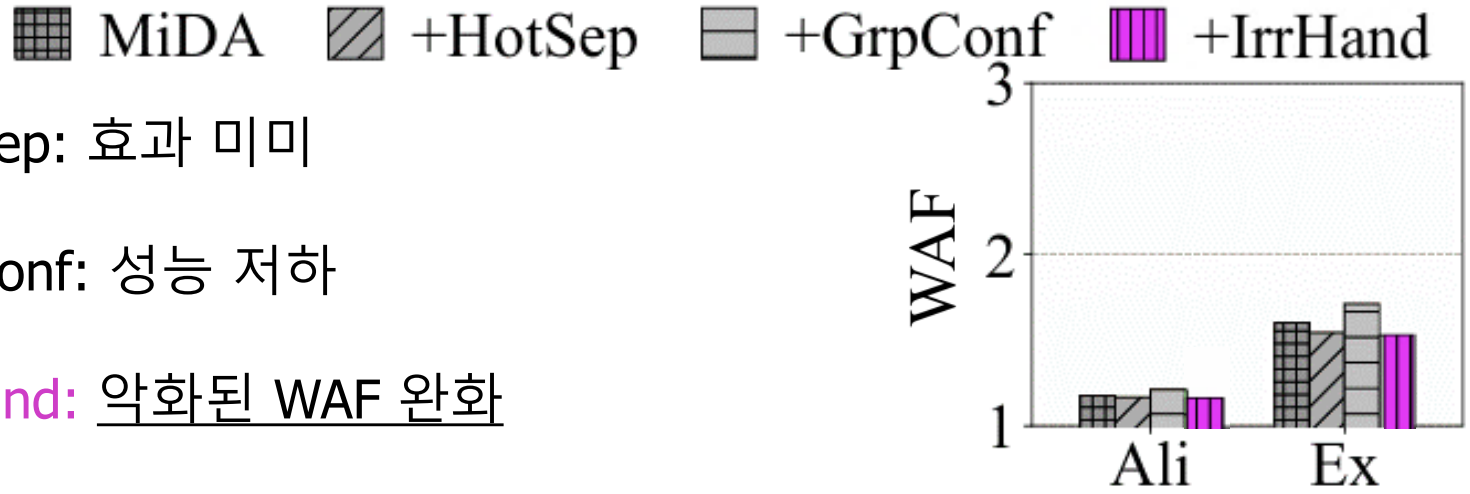
- UID 추정에서 Alibaba, Exchange는 큰 불규칙성 때문에 높은 오차율을 보임
  → 잘못된 그룹 형성으로 이어짐

Table 4: Accuracy of UID in predicting transition probabilities

Workloads	F-H	V	Y-A	Y-F	T-C	Ali	Ex	F-M
Avg. error (%)	1.82	6.90	2.33	1.78	4.81	**11.44**	**8.16**	1.33
Avg. error (%) w/ +IrrHand	1.82	1.97	2.33	1.78	2.38	4.67	3.36	1.33

5.2 Impact of Each Component of MiDAS

- MiDA에 HotSep, GrpConf, IrrHand 차례로 적용한 결과



HotSep: 효과 미미

GrpConf: 성능 저하

IrrHand: 악화된 WAF 완화

Fig. 12. Impact of individual components of MiDAS

5.3 Impact of epoch length

- UID와 Epoch 길이 관계

  : Epoch 길이에 따라 UID 측정 빈도, 결과 달라짐

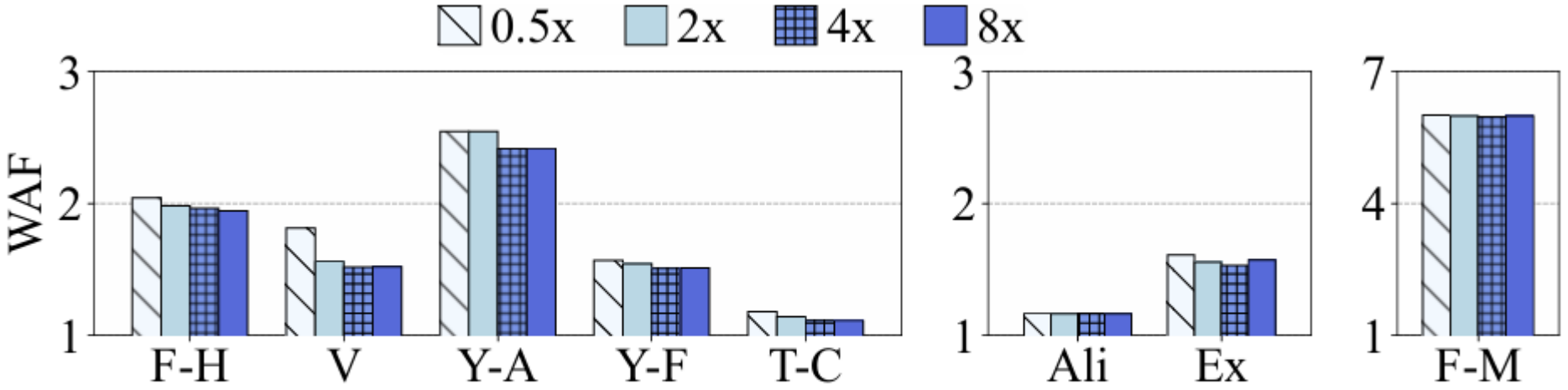


Fig. 13. Impact of length of epoch on generating UID

5.3 Impact of epoch length

- UID와 Epoch 길이 관계

  : Epoch 길이에 따라 UID 측정 빈도, 결과 달라짐

Epoch 줄이면 WAF 증가

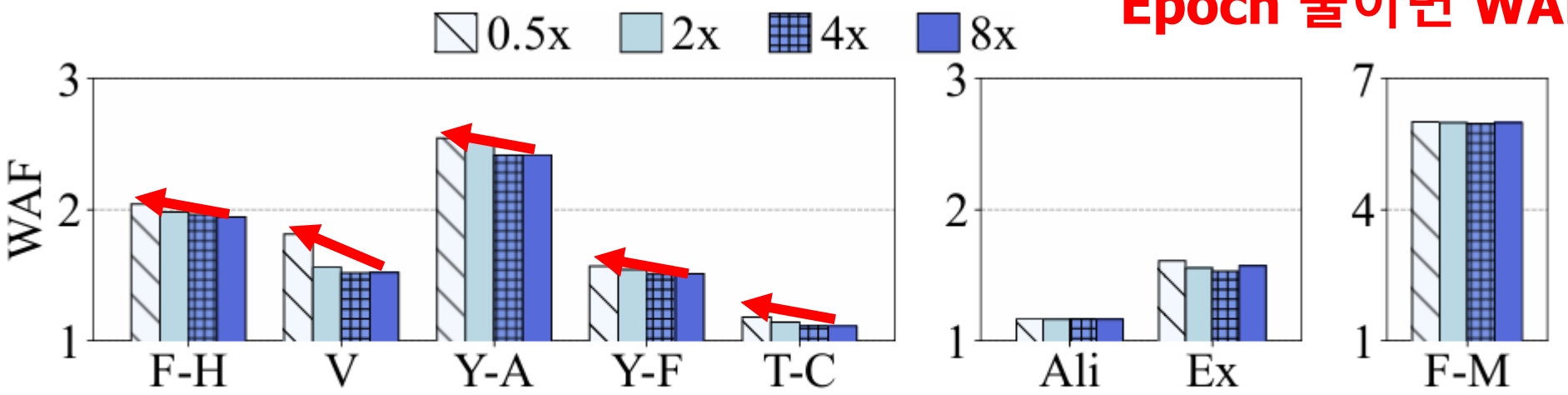
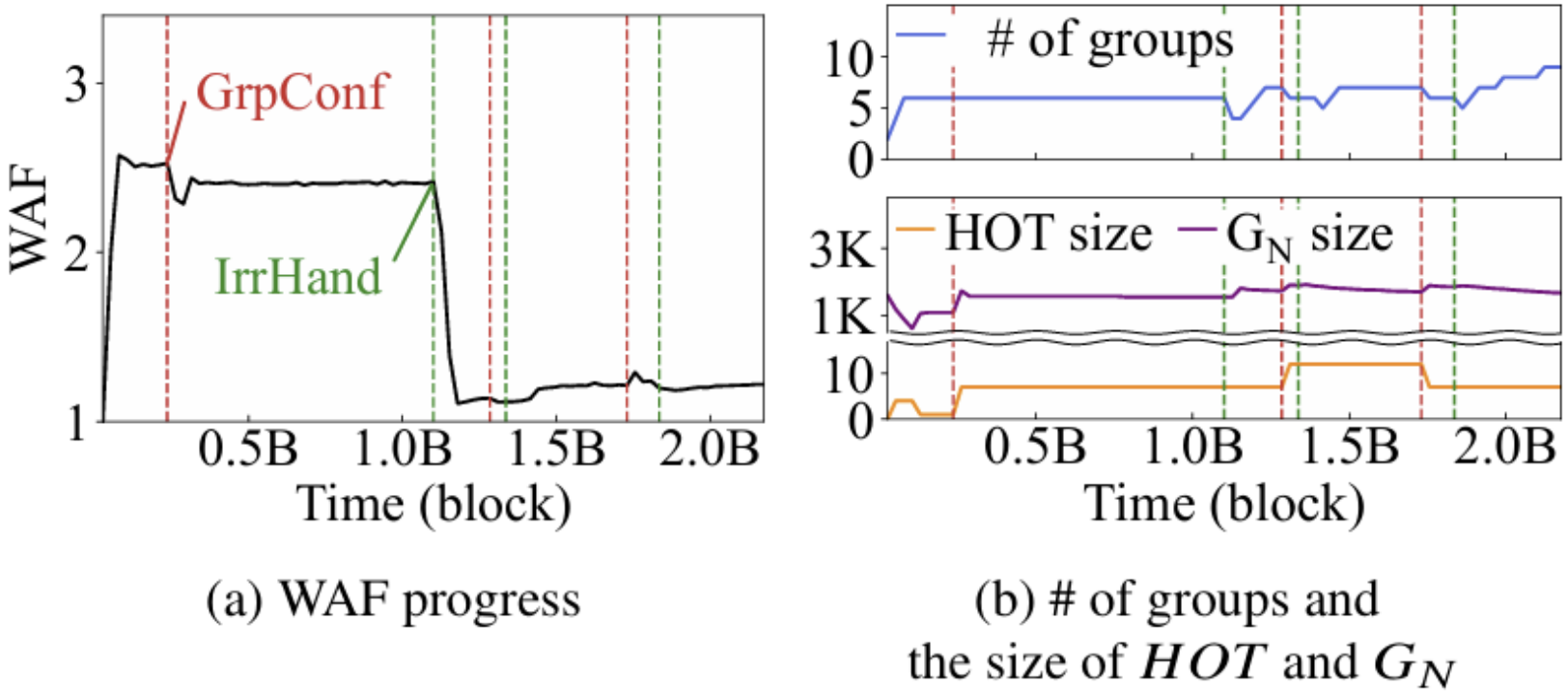


Fig. 13. Impact of length of epoch on generating UID

5.4 Adapting capability

- 워크로드 패턴에 따른 MiDAS 반응



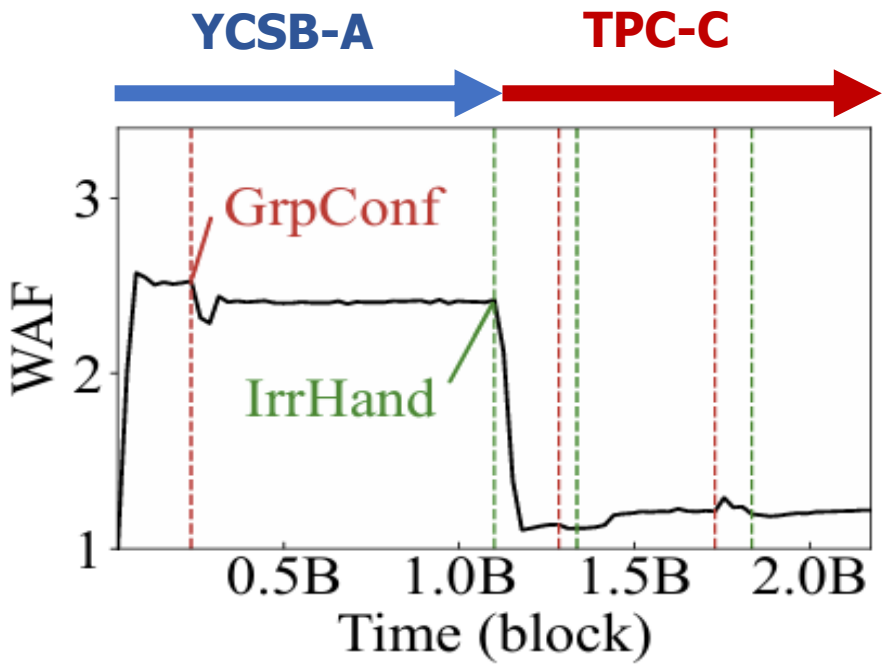
(a) WAF progress

(b) # of groups and the size of HOT and G_N

Fig. 14. MiDAS adapting to workload change

5.4 Adapting capability

- 워크로드 패턴에 따른 MiDAS 반응



(a) WAF progress

실험 환경

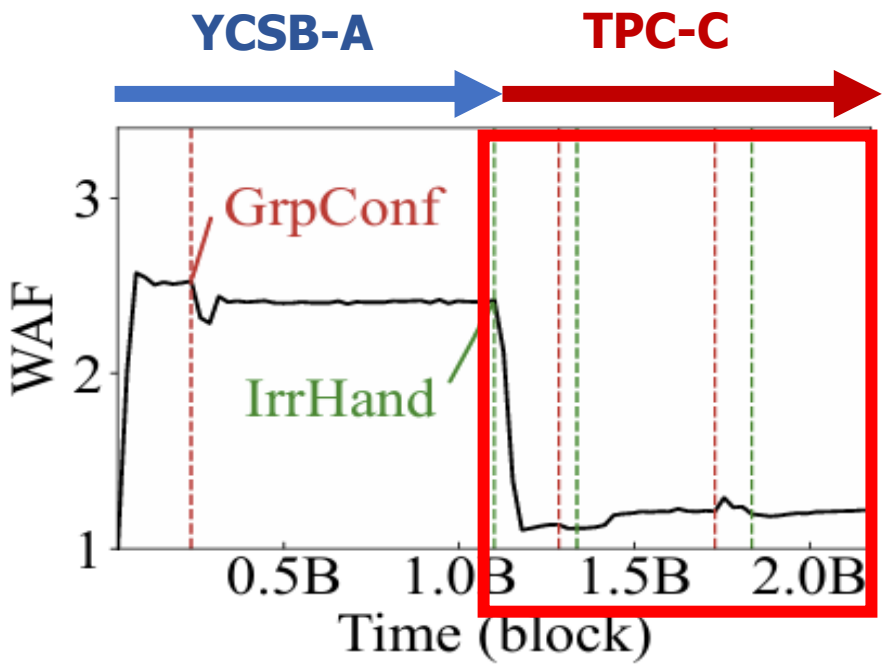
0 ~ 1.1billion(B): YCSB-A

1.1billion(B) ~ : TPC-C

Fig. 14. MiDAS adapting to workload change

5.4 Adapting capability

- 워크로드 패턴에 따른 MiDAS 반응



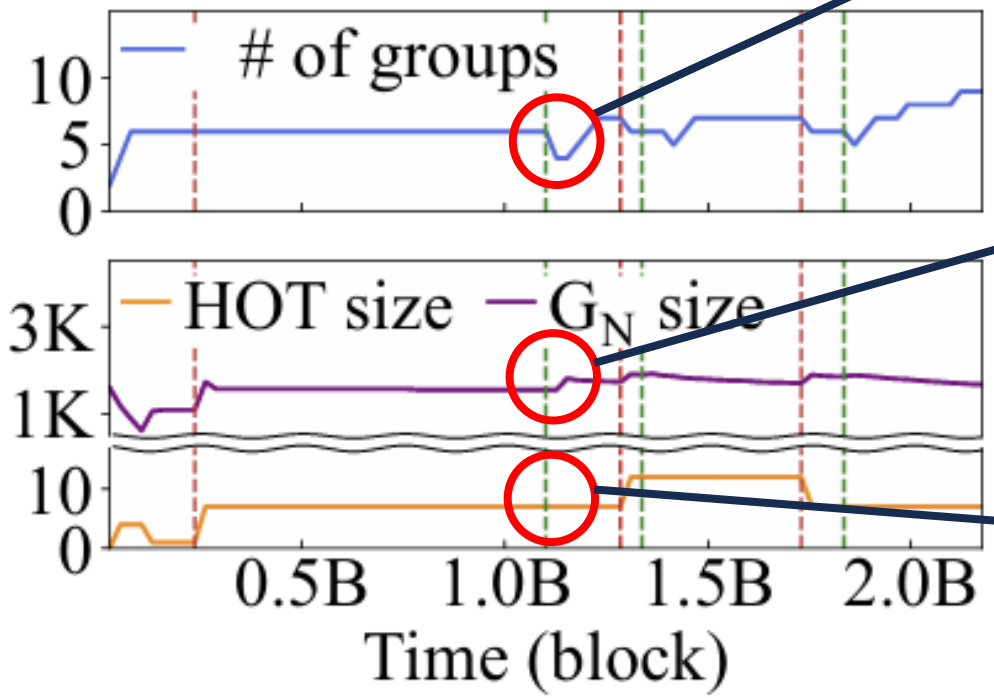
(a) WAF progress

워크로드가 TPC-C로 변경된 시점부터 극적인 패턴 변화로 IrrHand, GrpConf 호출 빈도 증가

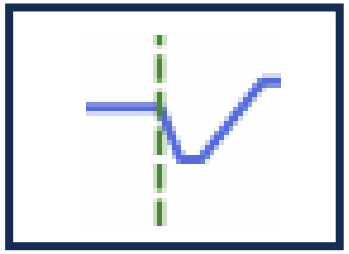
Fig. 14. MiDAS adapting to workload change

5.4 Adapting capability

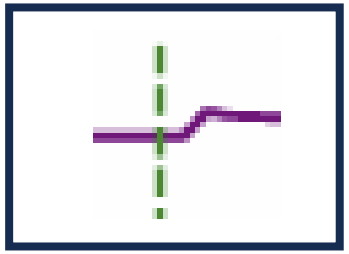
- 워크로드 변화 시점에서 그룹 수, 크기 상태



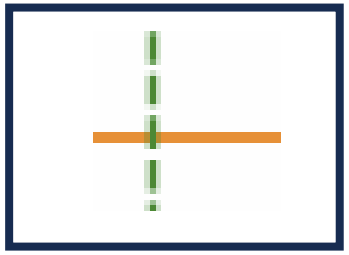
(b) # of groups and the size of HOT and G_N



높은 오류율 가진 그룹 병합으로 인해 그룹 수 감소



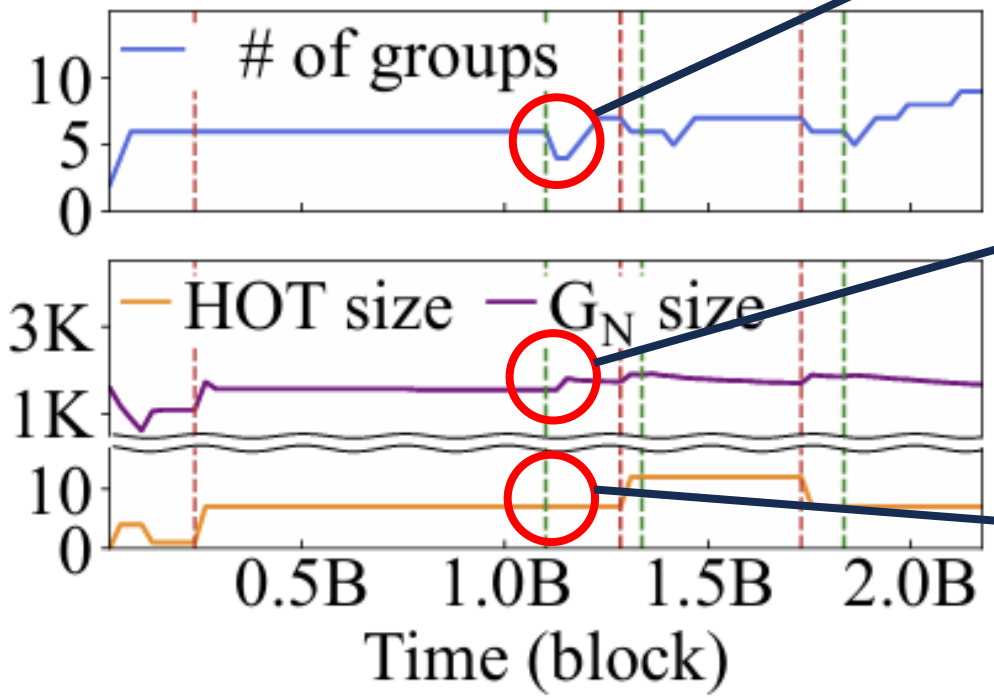
높은 오류율 가진 그룹 병합으로 인해 그룹 크기 변동



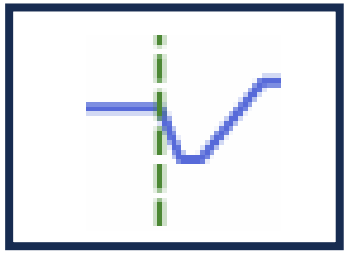
HOT 크기는 오류율이 낮아 변하지 않음

5.4 Adapting capability

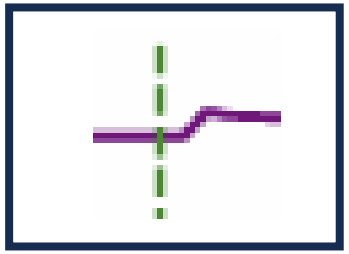
- 워크로드 변화 시점에서 그룹 수, 크기 상태



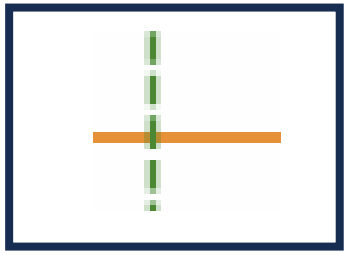
(b) # of groups and the size of HOT and G_N



높은 오류율 가진 그룹 병합으로 인해 그룹 수 감소



높은 오류율 가진 그룹 병합으로 인해 그룹 크기 변동

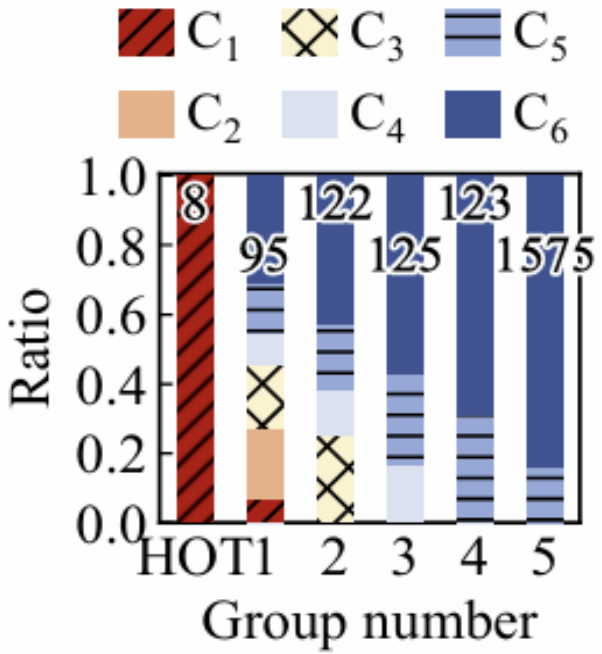


HOT 크기는 오류율이 낮아 변하지 않음

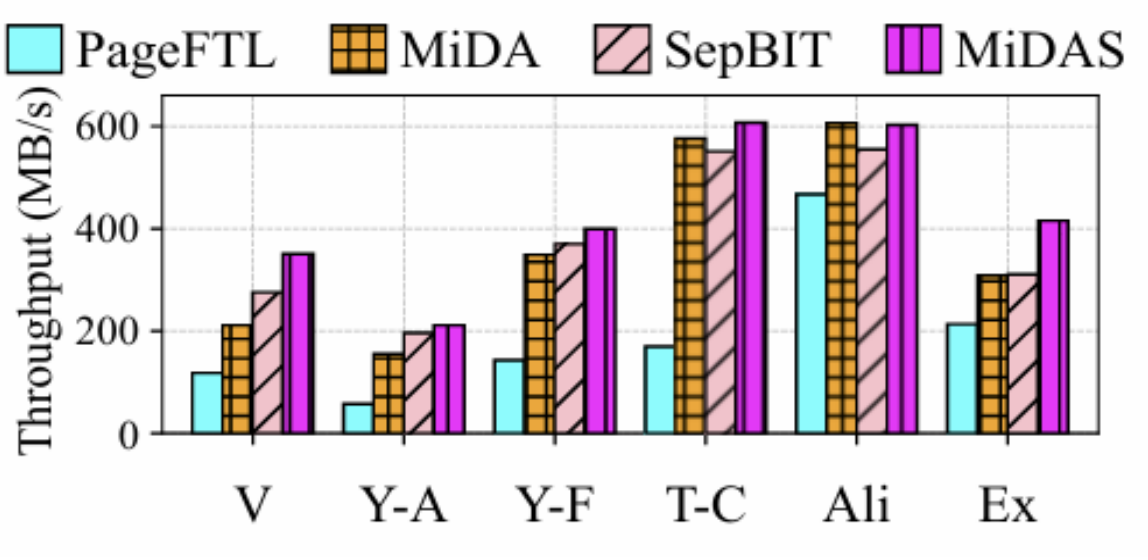
MiDAS는 워크로드의 변화에 따라 필요한 그룹 수, 크기를 효과적으로 조정하고 있음

5.5 Comparison with ORA

- MiDAS 적용 결과(YCSB-A)



(a)

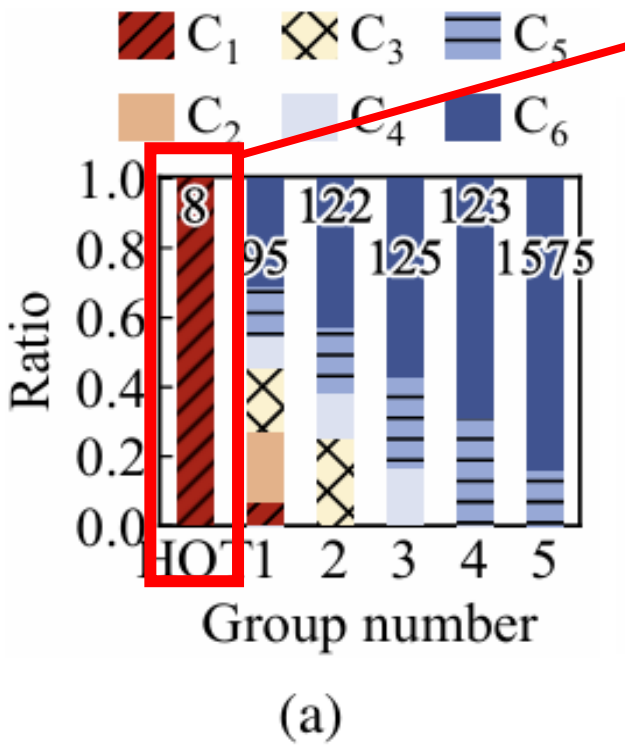


(b)

Fig. 15. (a) MiDAS block distribution, (b) Throughput result

5.5 Comparison with ORA

- MiDAS 적용 결과(YCSB-A)



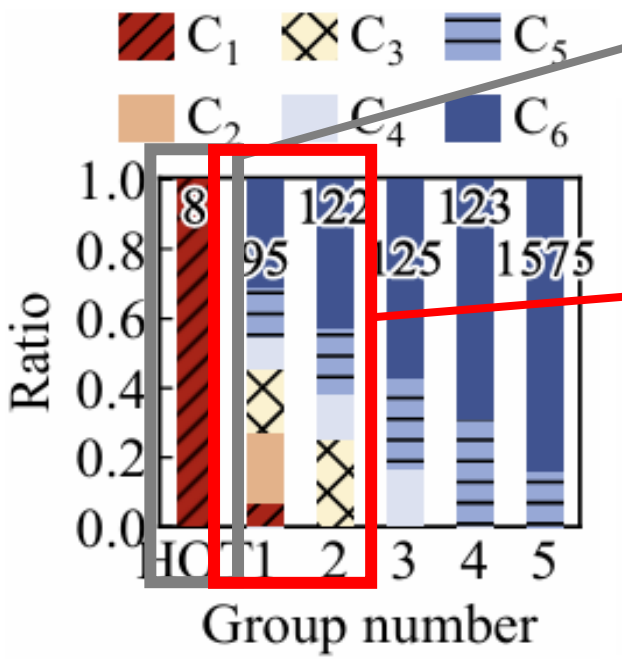
(a)

대부분의 HOT 블록 제대로 배치

Fig. 15. (a) MiDAS block distribution, (b) Throughput result

5.5 Comparison with ORA

- MiDAS 적용 결과(YCSB-A)



(a)

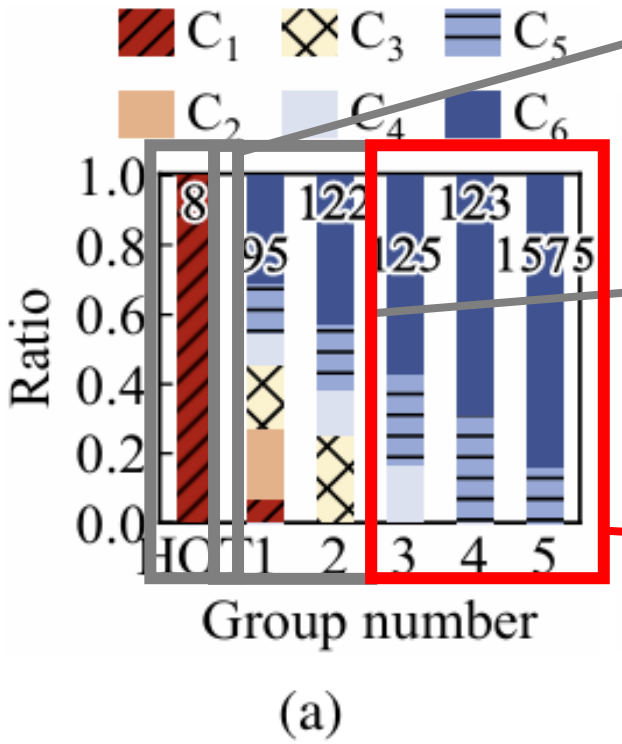
대부분의 HOT 블록 제대로 배치

그룹 이동 수로 분류하는 나이기반 이동 정책으로 인한 Cold 분류 오류

Fig. 15. (a) MiDAS block distribution, (b) Throughput result

5.5 Comparison with ORA

- MiDAS 적용 결과(YCSB-A)



대부분의 HOT 블록 제대로 배치

그룹 이동 수로 분류하는 나이기반 이동 정책으로 인한 Cold 분류 오류

후속 그룹에서 효율적으로 Cold 분류하여 전체 WAF 줄임

Fig. 15. (a) MiDAS block distribution, (b) Throughput result

5.5 Comparison with ORA

- MiDAS 적용 결과(YCSB-A)

결과적으로, 다른 기술에 비해 높은 처리량을 보임

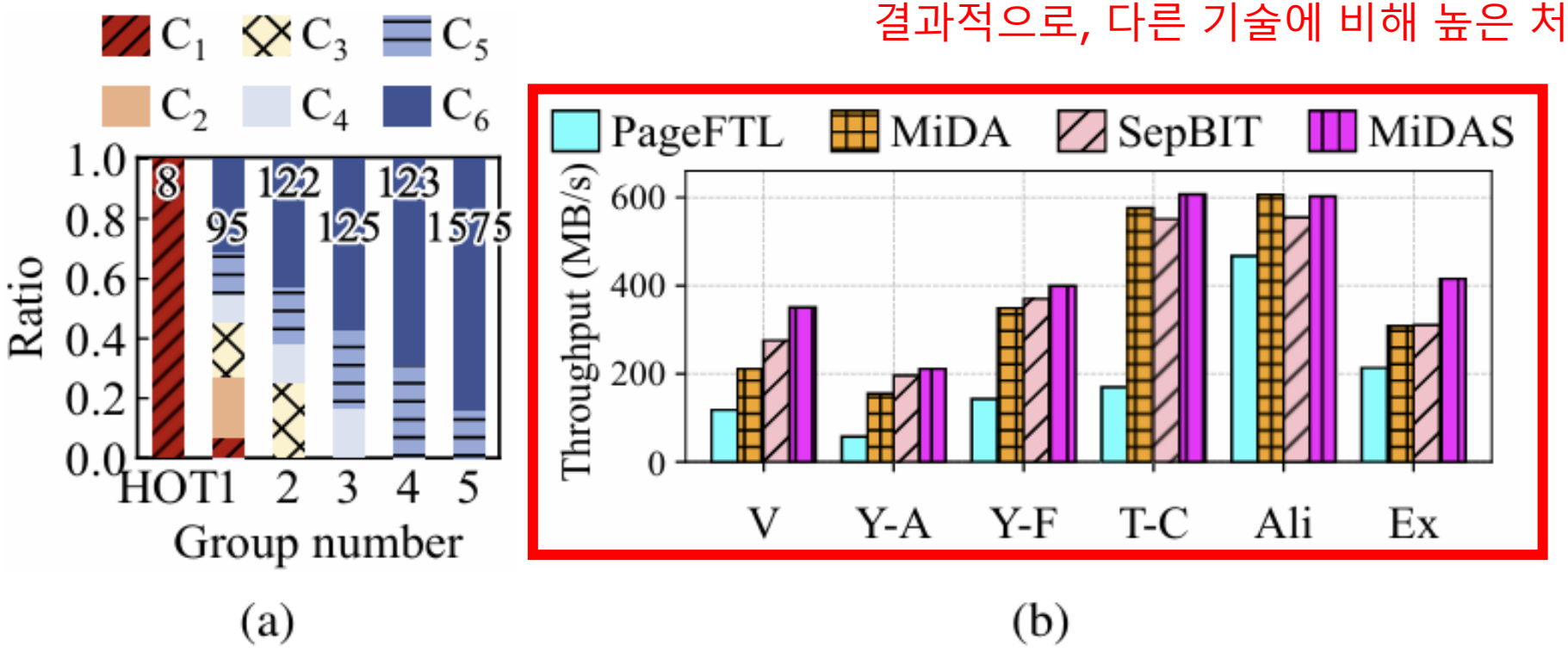
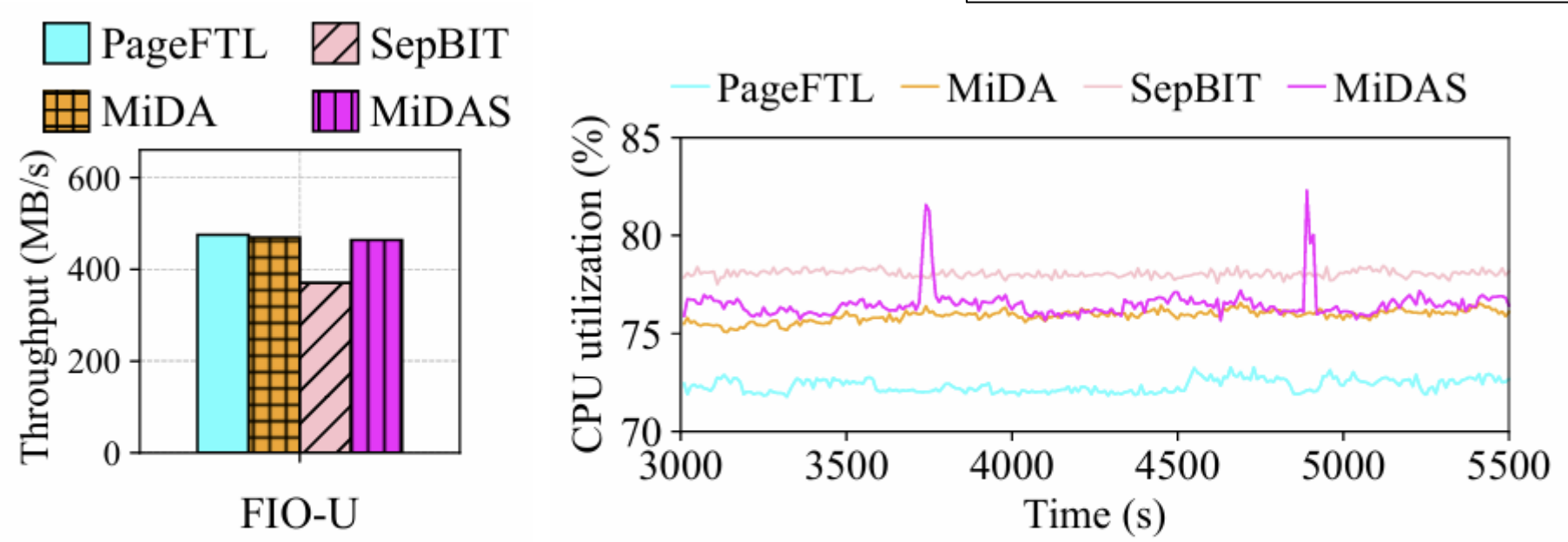


Fig. 15. (a) MiDAS block distribution, (b) Throughput result

5.6 Experiments on SSD Prototype

- MiDAS 적용 결과(FIO-U)

실험환경: GC의 영향 배제하기 위해 FIO-U 사용하여 모든 WAF 거의 동일하도록 구성

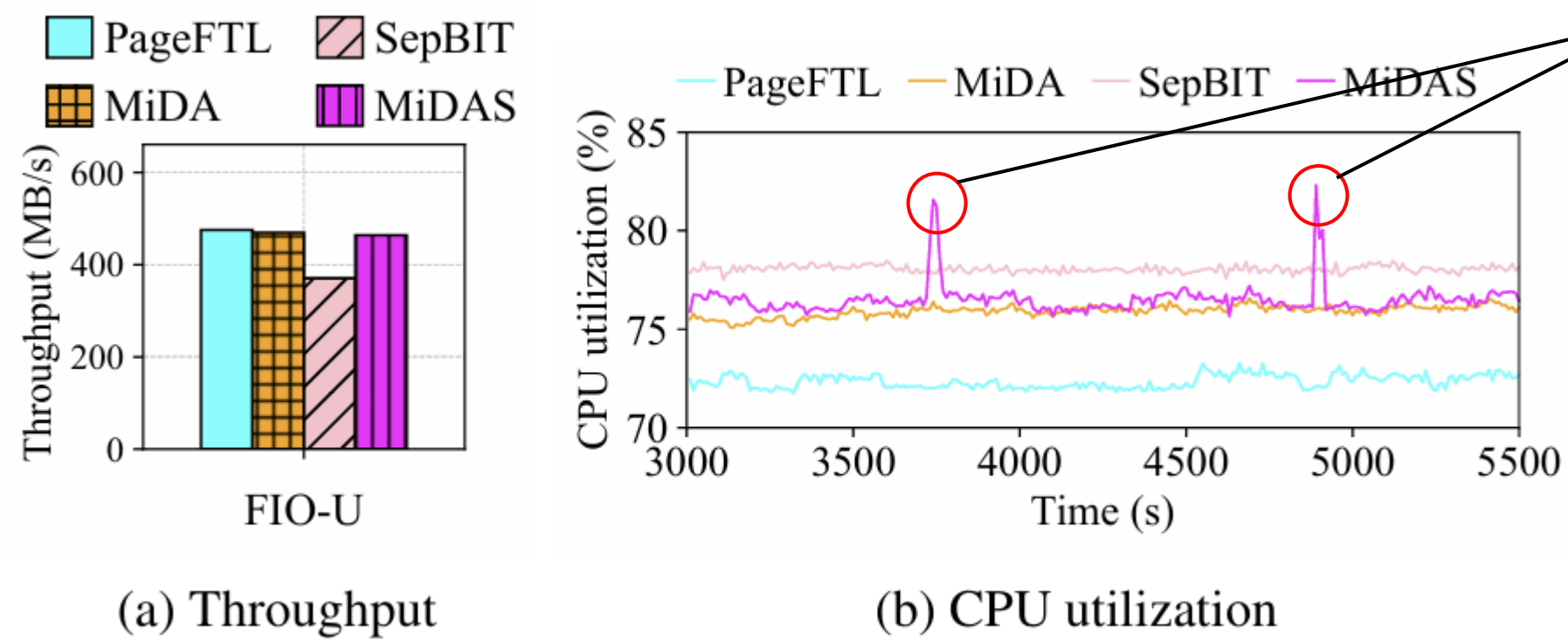


(a) Throughput (b) CPU utilization

Fig. 16. Results from our SSD prototype for FIO-U

5.6 Experiments on SSD Prototype

- MiDAS 적용 결과(FIO-U)



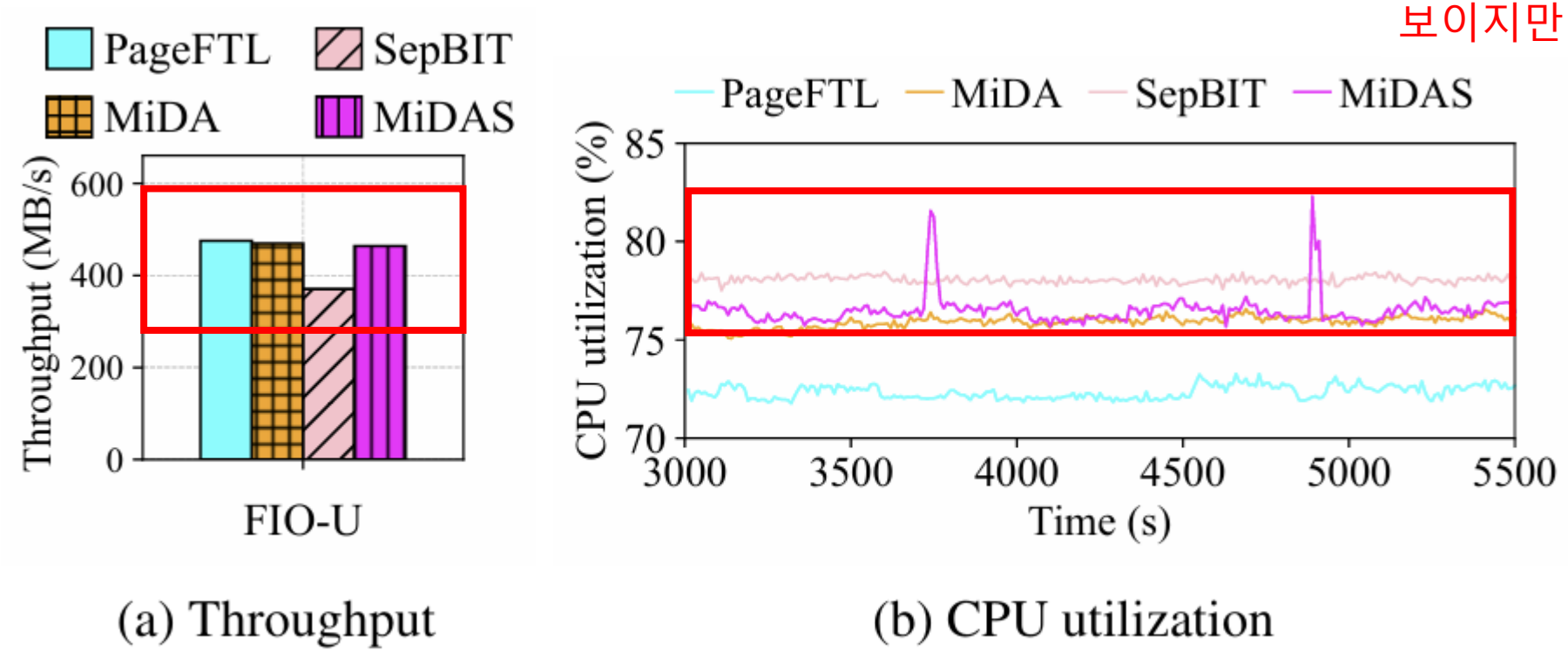
(a) Throughput (b) CPU utilization

Fig. 16. Results from our SSD prototype for FIO-U

5.6 Experiments on SSD Prototype

- MiDAS 적용 결과(FIO-U)

PageFTL, MiDA보다 높은 CPU 이용률 보이지만 처리량은 유사



(a) Throughput (b) CPU utilization

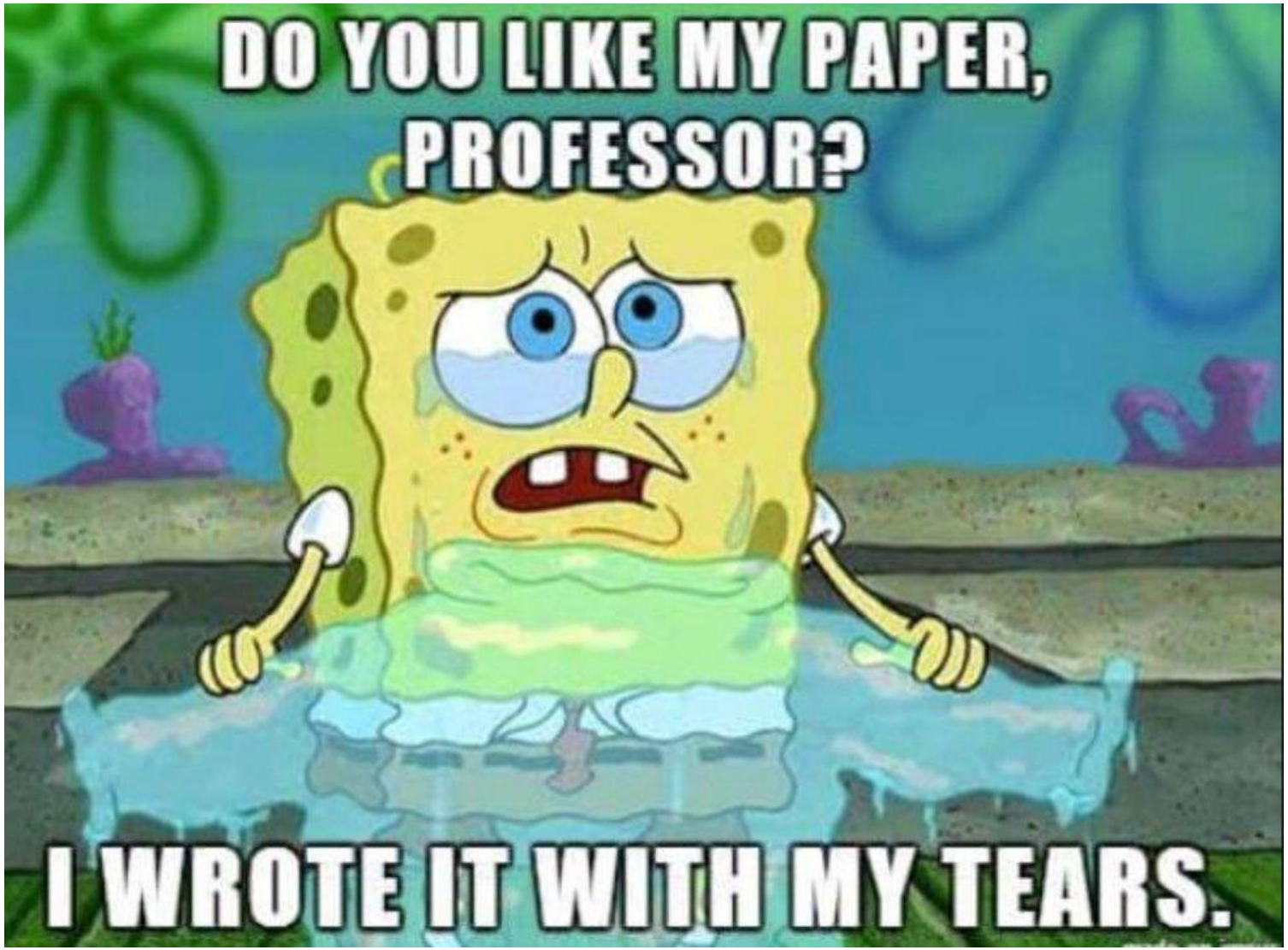
Fig. 16. Results from our SSD prototype for FIO-U

6. Conclusion

6. Conclusion

- Log Structured System에서 GC 오버헤드를 줄이기 위해 고안됨
- UID와 MCAM을 사용하여 그룹 간 데이터 이동을 최소화 한 체인 구조 사용
- Hot-Cold 데이터 분리하고, 그룹 크기 동적으로 조정하여 전체 GC 비용 최소화
- MiDAS는 기존 기술에 비해 WAF는 25% 감소 시키고, 처리량은 54% 증가시킴

6. Conclusion



Thank you

Q&A