Notions of Fairness in SSD Resource Allocation

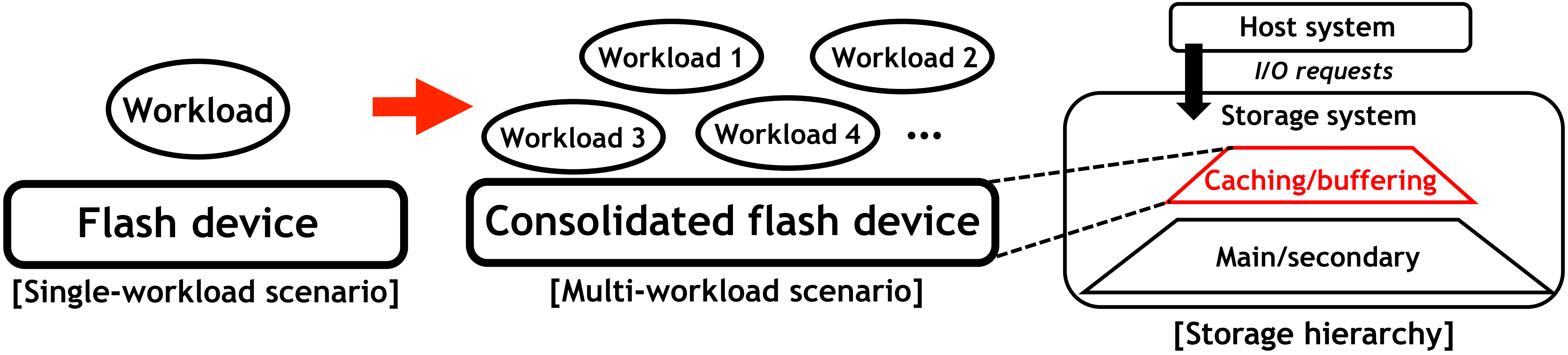
Wonil Choi
Hanyang University

NVRAMOS 2023

Introduction: Success of NAND Flash Memory

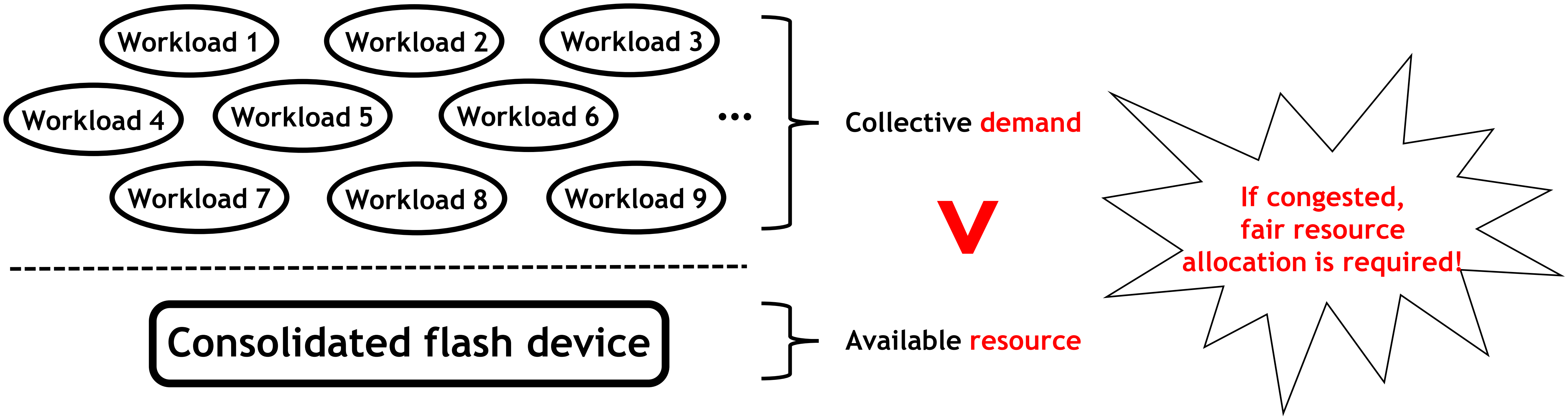
- **Cost aspect: decreased cost-per-bit (or increased memory capacity)**
  - Small feature size (below 10nm)
  - 3D packaging technology (beyond 100 layers)
  - High cell bit density (beyond 4 bit-per-cell)
- **Performance aspect: decreased latencies & increased bandwidths**
  - Advanced NAND flash commands (e.g., full sequence program, suspend/resume)
  - SSD-level enhancement techniques (e.g., caching, parallelism)
  - State-of-the-art host-SSD interfaces (e.g., PCIe, NVMe)
- **Consequently, NAND flash-based SSDs are more widely used**

Introduction: Advent of Consolidated Flash



- Recently, a single flash device is shared by multiple workloads
  - Traditionally, only one workload is executed on a single flash device
  - Thanks to the increased memory capacity and performance, multiple workloads can be executed simultaneously on a single flash device
  - We call such a flash device "consolidated flash"
- Consolidated flash can be found in the storage hierarchy
  - A consolidated flash can be used as main/secondary storage
  - We mainly target a consolidated flash in a caching/buffering layer

Introduction: Fair Resource Allocation for Consolidated Flash in Congestion



- There is a possibility of congestion for flash resources
  - Workloads' collective resource demands > available amounts of resource
  - In such situations, a fair resource allocation is required
- We explore fair resource allocations for consolidated flash

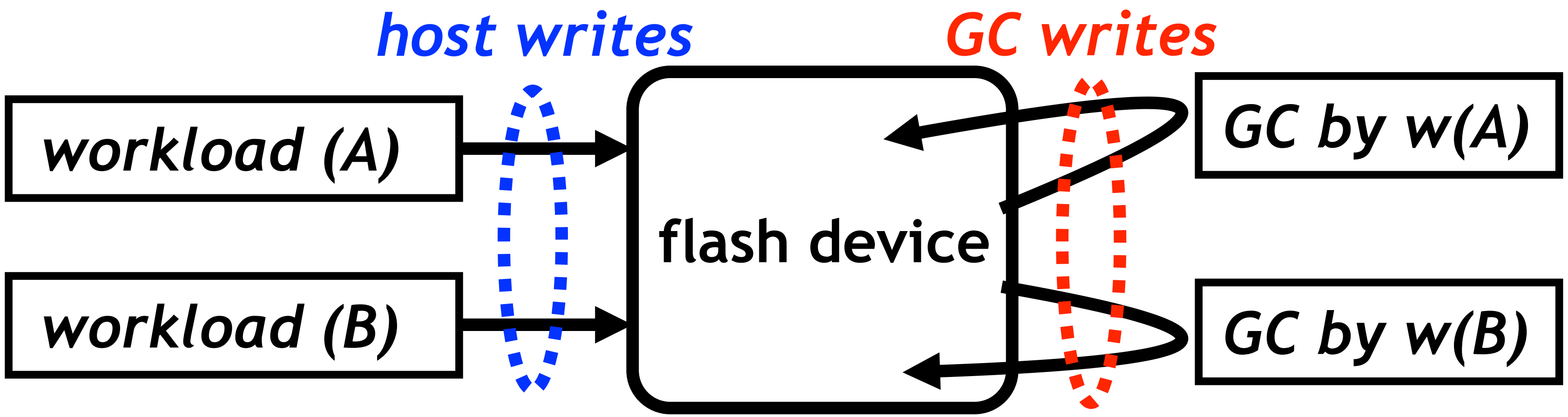
Target SSD Resource Types (Executive Summary)

- A flash device has its own finite lifetime
  - A device can service a fixed number of write operations, after which it becomes unavailable
  - Considered the flash lifetime as a first-class resource to be allocated
- Allocation of flash lifetime?
  - Total # writes the co-running workloads can collectively consume is given (write budget)
  - The write budget is distributed (allocated) to the competing workloads
  - Each workload is not allowed to consume flash lifetime once its allocated writes run out
- (1) Fair allocation of "device lifetime" in isolation
  - Assumed that other resources (bandwidth, capacity) are not bottlenecks
  - Referred to as "write allocation problem"
- (2) Fair allocation of "device lifetime" on par with "capacity" and "bandwidth"
  - Assumed that all or any of the three resource types are bottlenecks
  - Referred to as "multi-resource allocation problem"

(1) Write Allocation Problem and Our Proposal

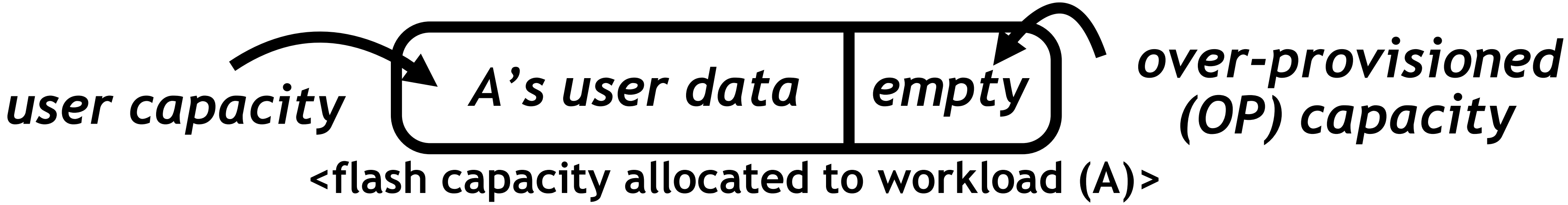
- Three challenges in write allocation problem
- Our corresponding approaches for the challenges
- Our proposed device lifetime management framework
- Compared fair-looking allocation strategies and evaluation of them

Challenge (i): Consideration of Hidden Writes

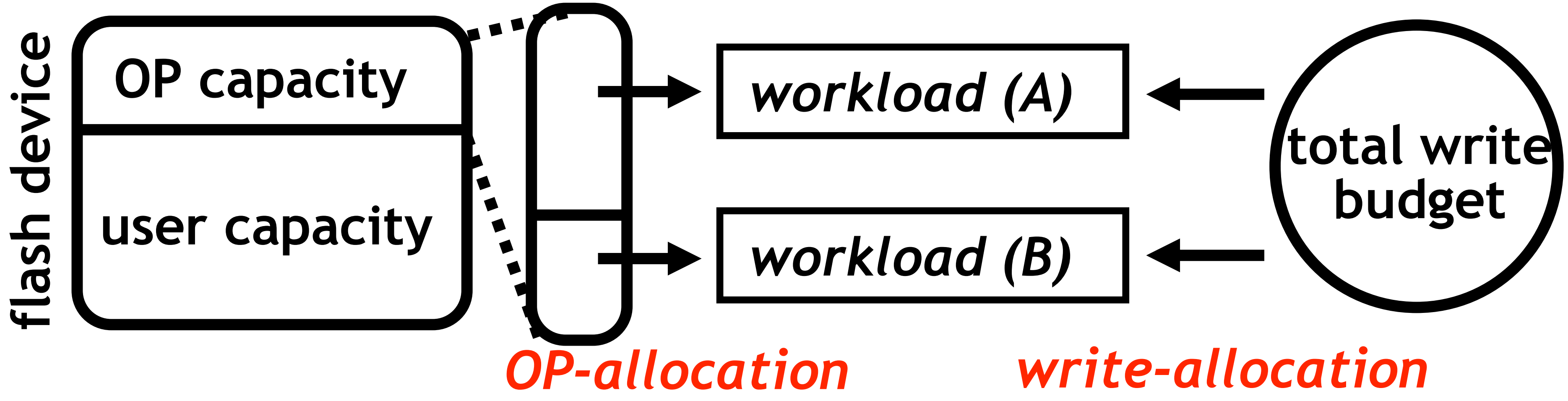


- We need to estimate write demand of each workload for write allocation
  - Write demand = estimated # host writes
  - Predicting # host writes is easy
- Garbage collection (GC) is another major write contributor
  - GC relocates valid data internally, which consumes writes
  - Write demand = estimated # host writes + estimated # GC writes
- Predicting # GC writes is non-trivial

Approach (i): OP Allocation as Part of Write Allocation



- **# GC writes is determined by the allocated OP capacity**
  - The larger OP capacity allocated, the fewer GC writes generated

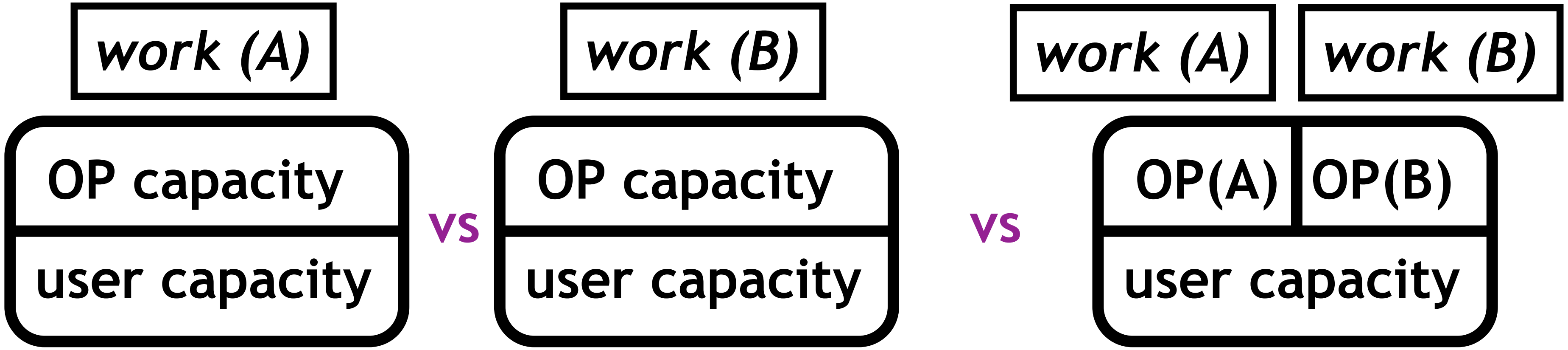


- **Write-allocation must come with corresponding OP-allocation**
  - OP-allocation: total OP capacity is divided to all the workloads
  - We employ an existing model to estimate # GC writes under varying OP sizes

Challenge (ii): Need of Fair Write Attribution

- Fair attribution of resource consumption is key to fair allocation
  - Write allocation = division of total available writes
  - Write attribution = division of already-consumed writes
- However, write attribution is non-trivial

W(·): # total writes   H(·): # host writes   G(·): # GC writes



W(A)=H(A)+G(A)   W(B)=H(B)+G(B)   W(A,B)=H(A)+H(B)+G(A,B)

  - # GC writes increases, when consolidated: G(A,B) > G(A) + G(B)
  - # total writes increases, when consolidated: W(A,B) > W(A) + W(B)
- How can we attribute G(A,B)-G(A)-G(B) to A and B?

Approach (ii): Employing Shapley Value

- We employ Shapley value from cooperative game theory
  - A tool for distributing the total gain (surplus) generated by a group of players participating in a cooperative game
  - The distribution is know to be fair
- The payoff for player i

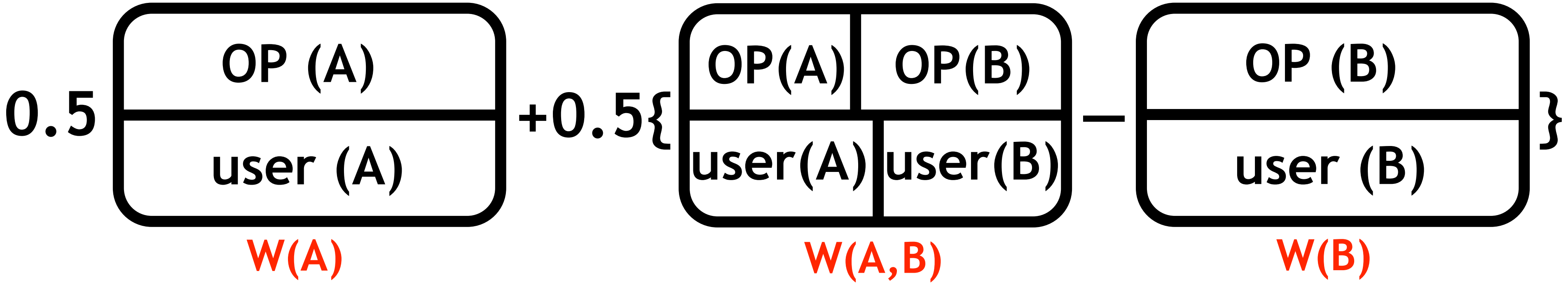
N: cooperation set S: subsets of N excluding i V(S): expected gain S makes

$$\phi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!\,(|N| - |S| - 1)!}{|N|!} (v(S \cup \{i\}) - v(S))$$

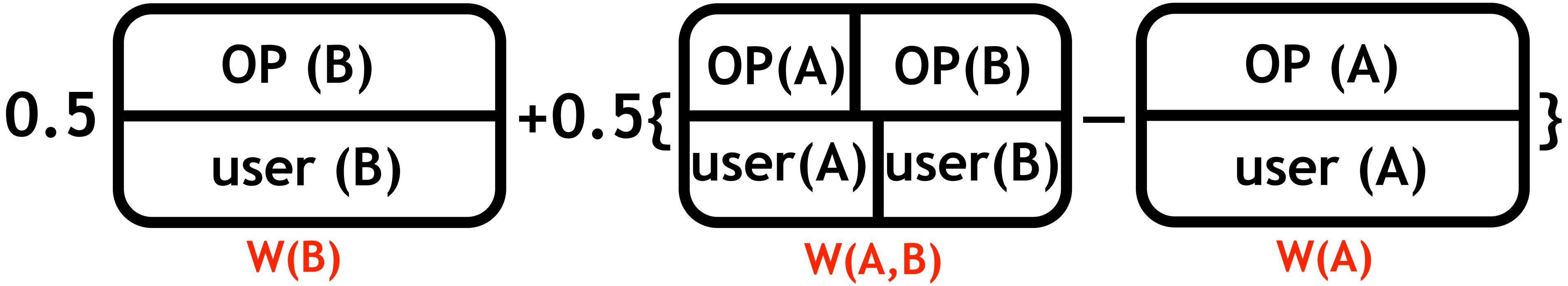
- V(SU{i}) = expected gain S + {i} make $ by S + {i}
- V(S) = expected gain S make $ by S
- V(SU{i}) - V(S) = contribution of i when he/she cooperates with S

Approach (ii): Employing Shapley Value

- Analogy - cooperative players : co-running workloads
  - Total gain : total # writes
  - Payoff for player i : # writes attributed to workload i
- Example: workloads A & B are co-running on a flash device
  - ϕA = 1/2 * W(A) + 1/2 * {W(A,B) - W(B)}

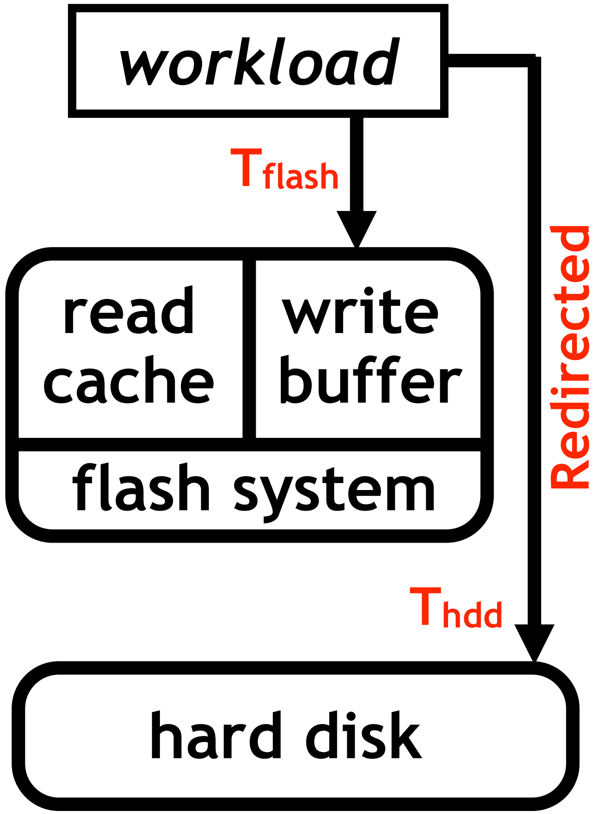


  - ϕB = 1/2 * W(B) + 1/2 * {W(A,B) - W(A)}



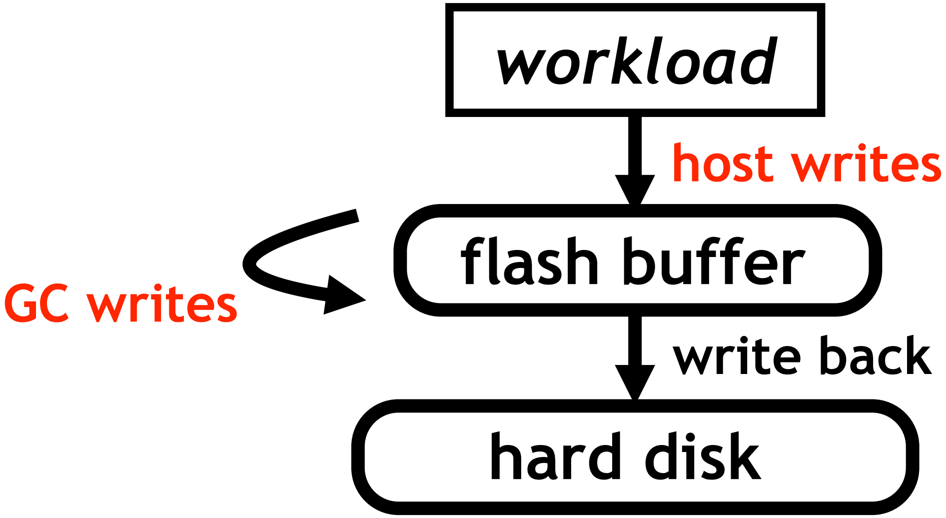
Challenge (iii): Need of Write Control Knob

- We need to enforce a write allocation
  - Each workload should not consume writes beyond its allocation
- When the device is used as main-storage
  - All writes should be serviced
  - Write allocation is a moot concern
- When the device is used as caching+buffering layer
  - Writes can be rejected from flash
  - Such writes are serviced from the next-layer
  - It brings performance (latency) penalty
- We target flash cache+buffer
  - Writes can be serviced from flash with short latencies
  - If necessary, writes can be redirected to hard disk with long latencies

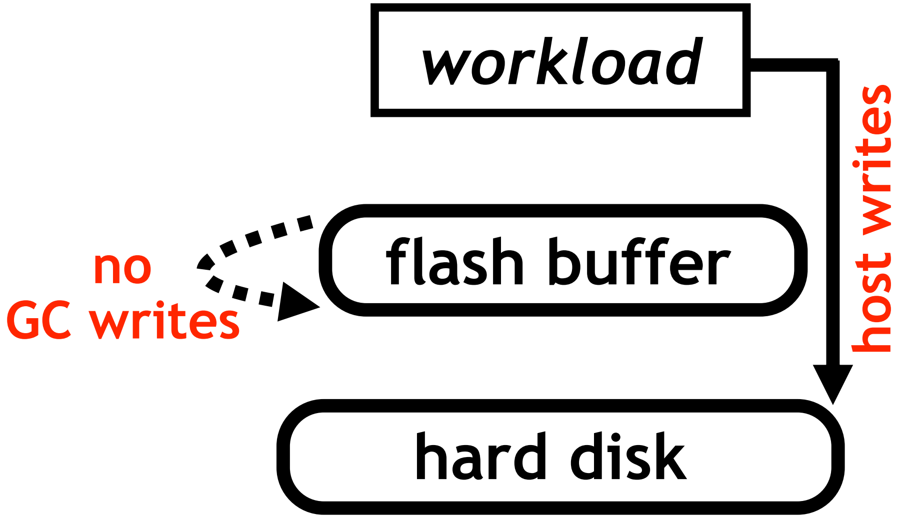


Approach (iii): Redirecting Writes beyond Budget

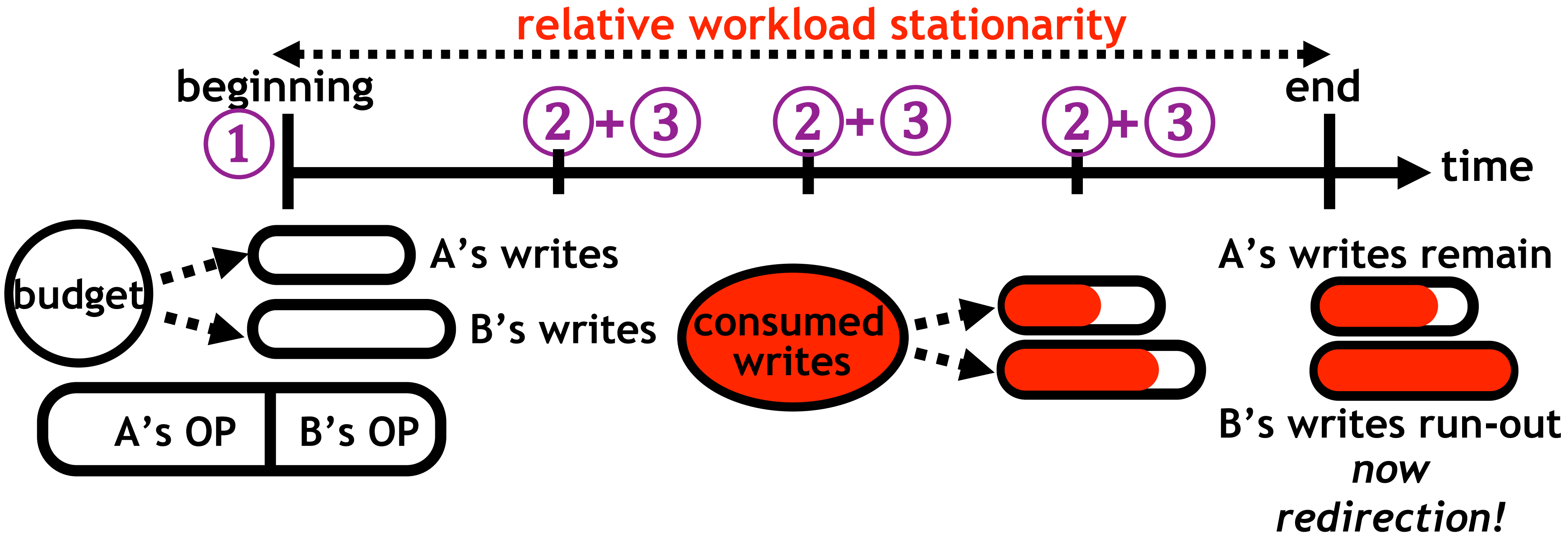
- Write consumption < allocated budget
  - All host writes are serviced from flash
  - Write service times are short
  - GC also consumes writes



- Write consumption >= allocated budget
  - All host writes are redirected to hard disk
  - Write service times are long
  - No GC writes are generated
  - No more writes are consumed by the workload



Proposed Lifetime Management Framework



- ① write-allocation + OP-allocation
  - Total write budget and total OP capacity are divided to co-running workloads
- ② write-attribution periodically
  - Shapley value-based write attribution
- ③ write-redirection if allocated writes run-out
  - Workloads whose allocated writes run-out, their future writes are redirected by end of period

Compared Fair-looking Allocation Strategies

B(·): alloc budget W(·): # total writes H(·): # host writes G(·): # GC writes

- ① **EVEN: partitioning OP evenly**
  - (i) Allocate OP evenly
  - (ii) Estimate demands

A's OP	B's OP

B(A) ← W(A) = H(A)+G(A)   B(B) ← W(B) = H(B)+G(B)

- ② **MMF: max-min fairness division of budget**
  - (i) Allocate OP evenly
  - (ii) Estimate demands
  - (iii) Apply max-min fairness
  - (iv) Re-partition OP based on $\underline{W}$

A's OP	B's OP

W(A) = H(A) + G(A)   W(B) = H(B) + G(B)

B(A) ← $\underline{W}$(A) = W(A)   B(B) ← $\underline{W}$(B) = W(B)

A's OP	B's OP

- ③ **Iso-WAF: equalizing write amplification factor**
  - (i) Find OP partition that achieves iso-WAF
  - (ii) Estimate demands

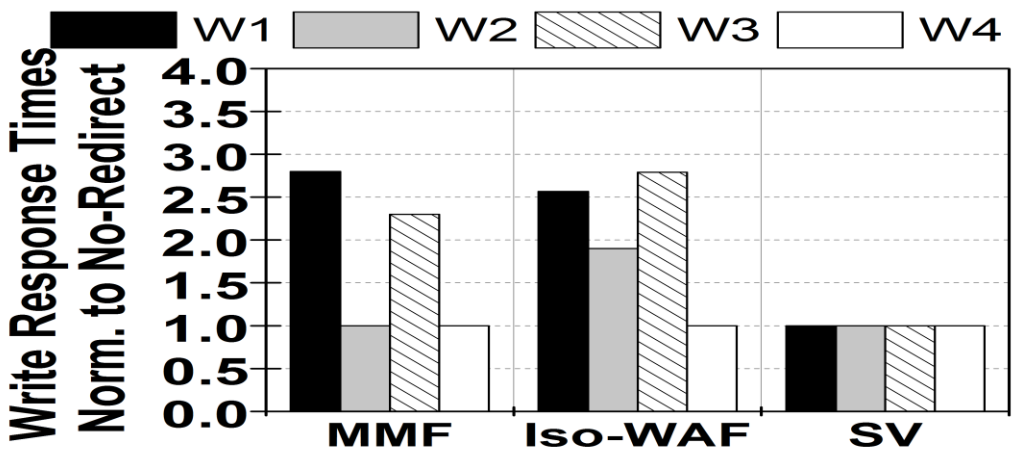
A's OP	B's OP

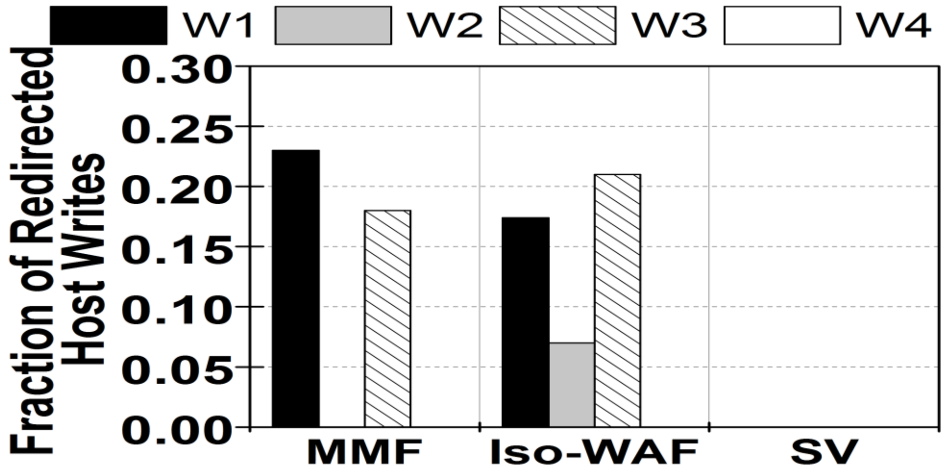
such that $\frac{W(A)}{H(A)} = \frac{W(B)}{H(B)}$

B(A) ← W(A) = H(A)+G(A)   B(B) ← W(B) = H(B)+G(B)

- ④ **SV: Shapley value division of budget**
  - Assuming that total budget is already consumed writes, apply Shapley value

Evaluation



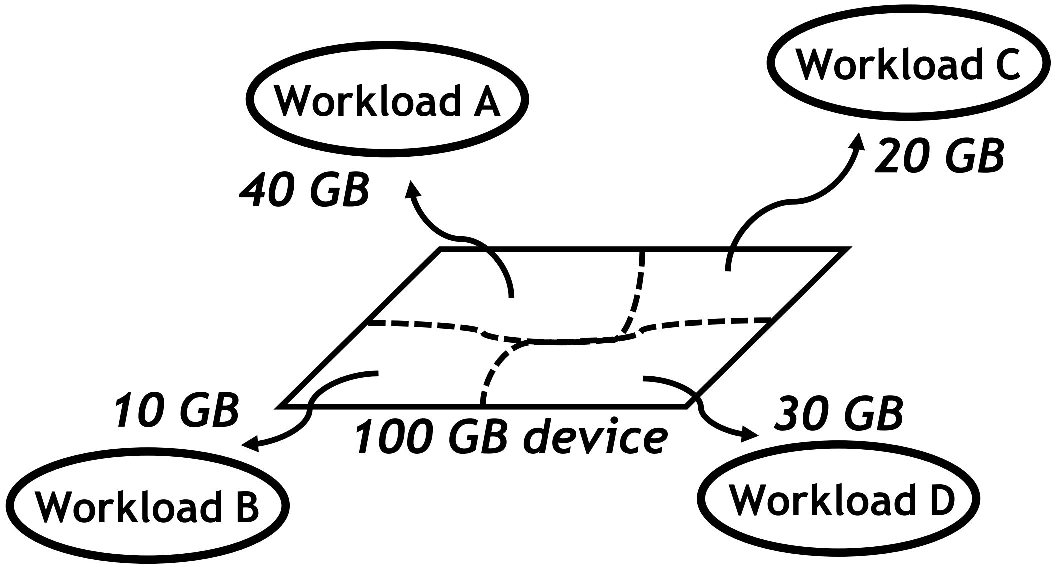


- Individual workloads are combined for multi-workload scenarios
- Evaluation of fairness in write allocation
  - Write response time is determined by # redirected writes
  - We evaluate fairness as the difference in write response times across the workloads
- Results are normalized to scenarios where device lifetime is not considered
- SV allocation is the best in terms of fairness
  - Write response times of the workloads are close to one another
  - SV allocation equalizes fraction of redirected host writes

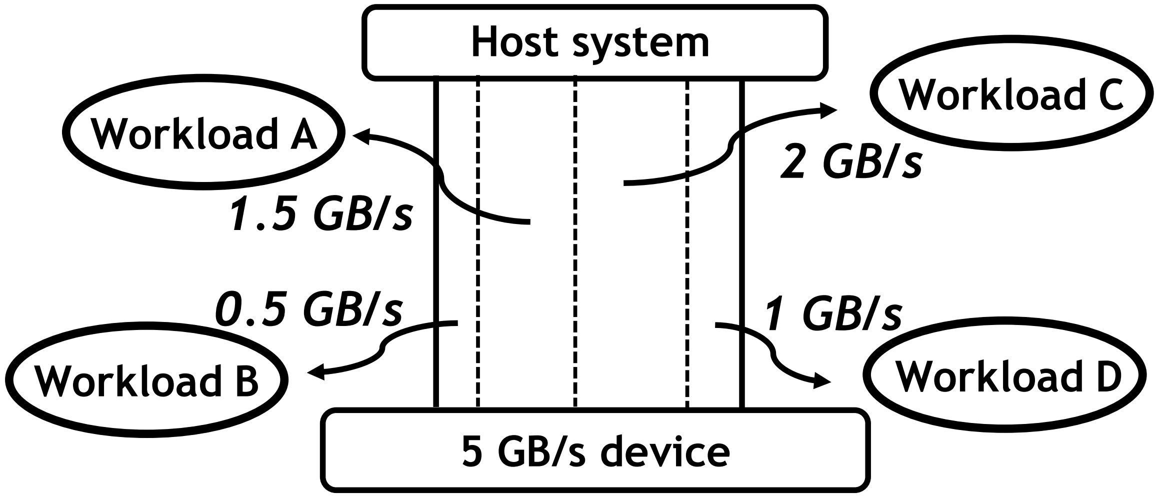
(2) Multi-Resource Allocation Problem and Our Proposal

- Limitations of existing resource allocations
- Dominant resource fairness (DRF) for multi-resource allocation
- Adopting DRF in flash context
- Evaluation DRF-allocations, non-DRF allocations, and their variants

Background: Existing Flash Resource Allocations



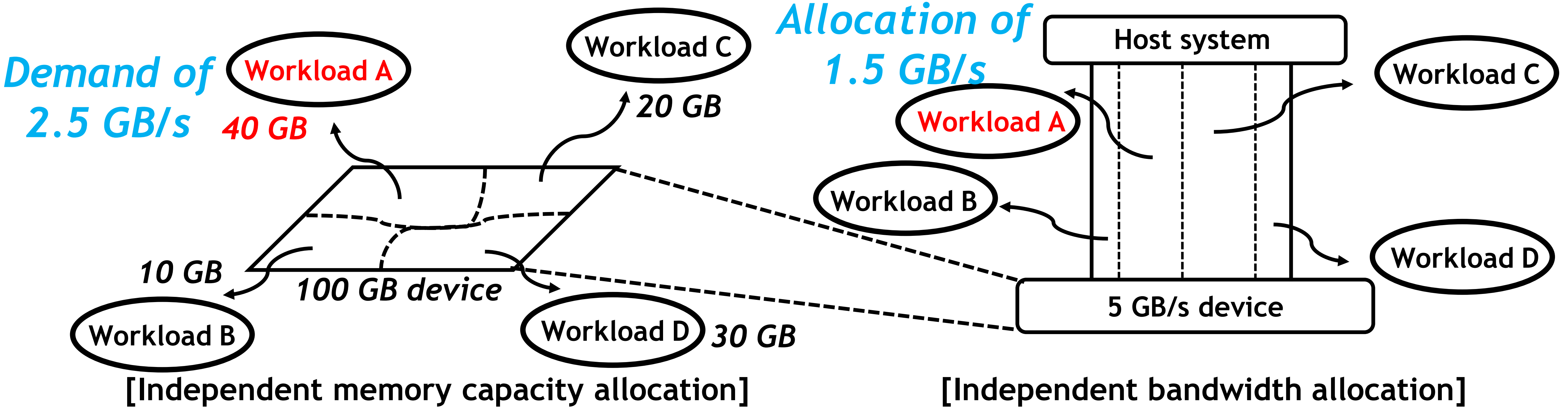
[Memory capacity allocation]



[Bandwidth allocation]

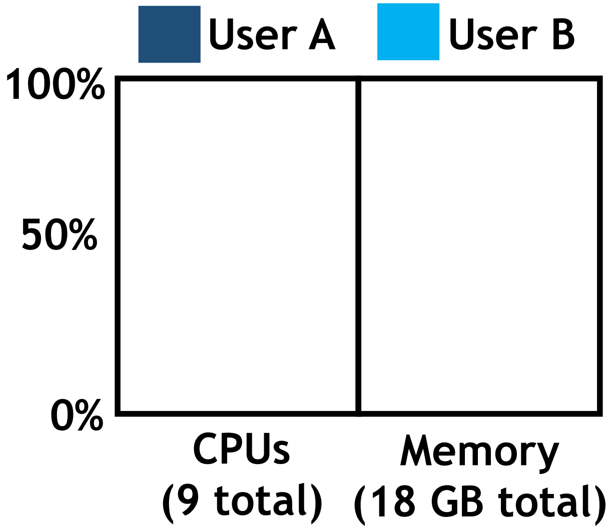
- Existing flash allocation techniques target two primitive resource types, capacity or bandwidth
- Flash capacity allocations
  - In a caching layer, the larger memory capacity for a workload, the higher hit ratio
- Flash bandwidth allocations
  - In a storage system, the more bandwidth for a workload, the higher performance

Motivation: Independent Resource Allocations



- Existing techniques allocate primitive resources **independently**
  - An independent memory capacity allocation does not consider corresponding bandwidth allocation
  - This can decrease the performance of workloads
    - Allocating 40 GB to workload A leads to a bandwidth demand of 2.5 GB/s
    - What if an independent bandwidth allocation gives only 1.5 GB/s to workload A?
- **Our motivation: all relevant resources should be allocated jointly**
  - A workload's demands for different resources are correlated to each other

Related Work: Dominant Resource Fairness (DRF)-1

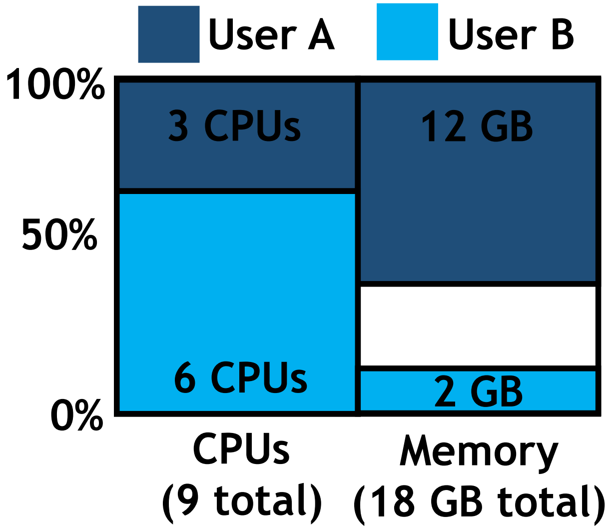


[Given two resources]

User A runs tasks with Demand-vector <1 CPU, 4 GB>

User B runs tasks with Demand-vector <3 CPU, 1 GB>

[Given two users]



Dominant-share of User A

$$\frac{12\text{ GB}}{18\text{ GB}} = \max\left(\frac{3\text{ CPUs}}{9\text{ CPUs}}, \frac{12\text{ GB}}{18\text{ GB}}\right)$$

||

Dominant-share of User B

$$\frac{6\text{ CPUs}}{9\text{ CPUs}} = \max\left(\frac{6\text{ CPUs}}{9\text{ CPUs}}, \frac{2\text{ GB}}{18\text{ GB}}\right)$$

[DRF seeks to equalize dominant-shares]

- DRF [NSDI'11] is a solution to "multi-resource" "fair" allocation problem
  - Given multiple resources: CPUs (9 total) & Memory (18 GB total)
  - Given multiple users with "demand-vector": User A's <1 CPU, 4 GB> & User B's <3 CPU, 1 GB>
    - DRF assumes that individual resources' demands increase in a linear fashion
  - Goal of DRF: users' dominant-shares are equalized
    - A user's dominant-share is the maximum her fractional needs for different resources

Related Work: Dominant Resource Fairness (DRF)-2

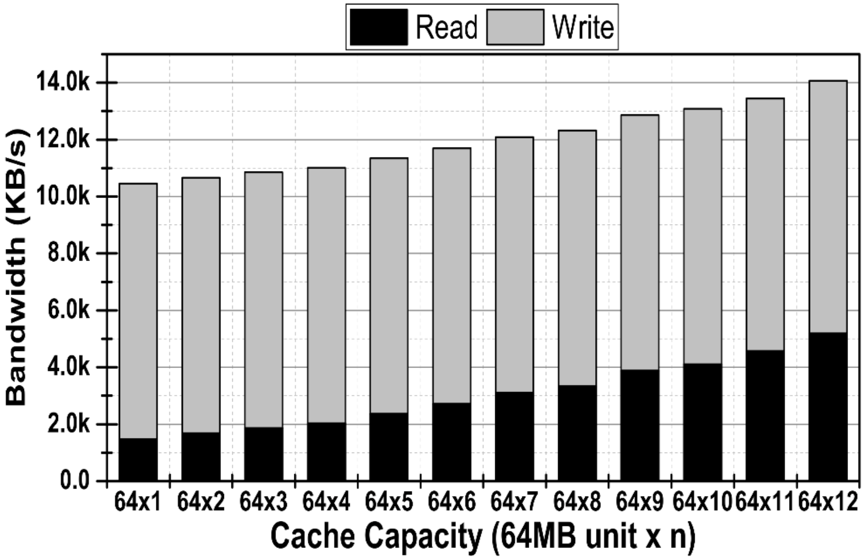
Total Resources: 9 CPUs, 18 GB Memory
User A <1 CPU, 4 GB>, User B <3 CPU, 1 GB>

Schedule	User A		User B		CPU	RAM
	res. shares	dom. share	res. shares	dom. share	total alloc.	total alloc.
User B	$\langle 0, 0 \rangle$	**0**	$\langle 3/9, 1/18 \rangle$	1/3	3/9	1/18
User A	$\langle 1/9, 4/18 \rangle$	**2/9**	$\langle 3/9, 1/18 \rangle$	1/3	4/9	5/18
User A	$\langle 2/9, 8/18 \rangle$	4/9	$\langle 3/9, 1/18 \rangle$	**1/3**	5/9	9/18
User B	$\langle 2/9, 8/18 \rangle$	**4/9**	$\langle 6/9, 2/18 \rangle$	2/3	8/9	10/18
User A	$\langle 3/9, 12/18 \rangle$	**2/3**	$\langle 6/9, 2/18 \rangle$	**2/3**	1	14/18

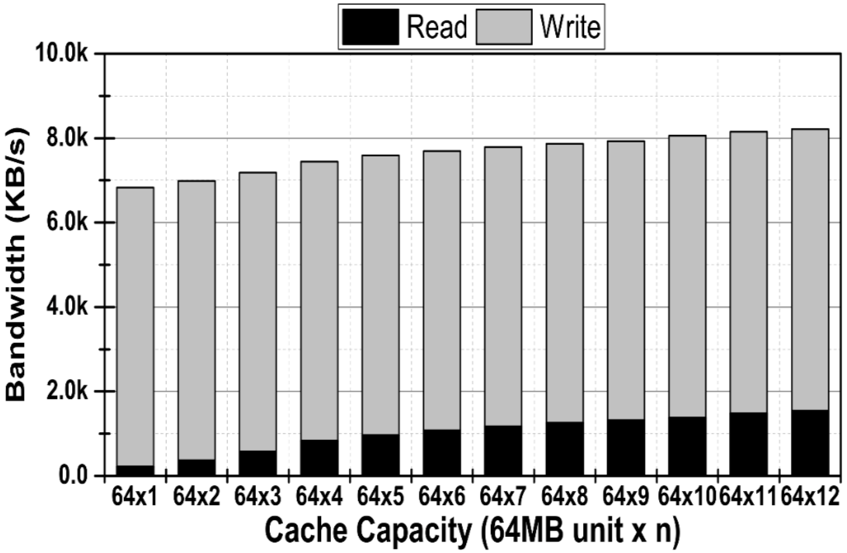
[DRF allocation process based on progressive-filling (PF) algorithm]

- DRF employs "progressive-filling (PF)" algorithm
  - It allocates demand-vector to a workload & calculates/compares dominant-shares
  - It repeats until one resource is fully allocated
- DRF based on demand-vector & PF offers several desirable fairness properties
  - Incentive compatibility, strategy-proofness, envy-freeness, and Pareto efficiency
  - Refer to [NSDI'11] for details

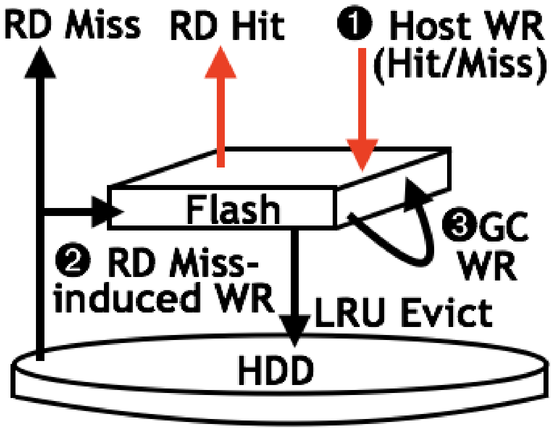
Observation: Flash Mem Cap vs Bandwidth Demands



[Open Storage Trace - *prn*]



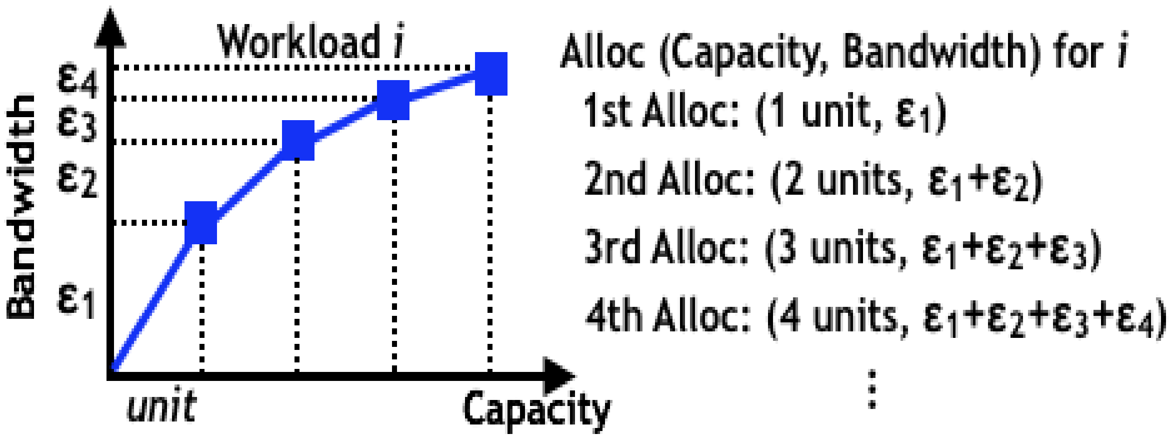
[Open Storage Trace - *hm*]



Bandwidth consumers = RD(read) hit + WR(write)

- **We characterized actual consumption (demands) of memory capacity & bandwidth**
  - Bandwidth consumption under varying memory capacity allocations
  - Bandwidth consumer (1): reads (from the host/workload) that are hit/serviced in flash cache
    - # read hits is determined by its read hit ratio, which depends on its allocated memory capacity
  - Bandwidth consumer (2): all writes (from the host/workload) that are serviced in flash buffer
    - Its bandwidth consumption is independent from allocated memory capacity
- **Demands for the two resources are not linear, thus, vanilla DRF cannot be employed**

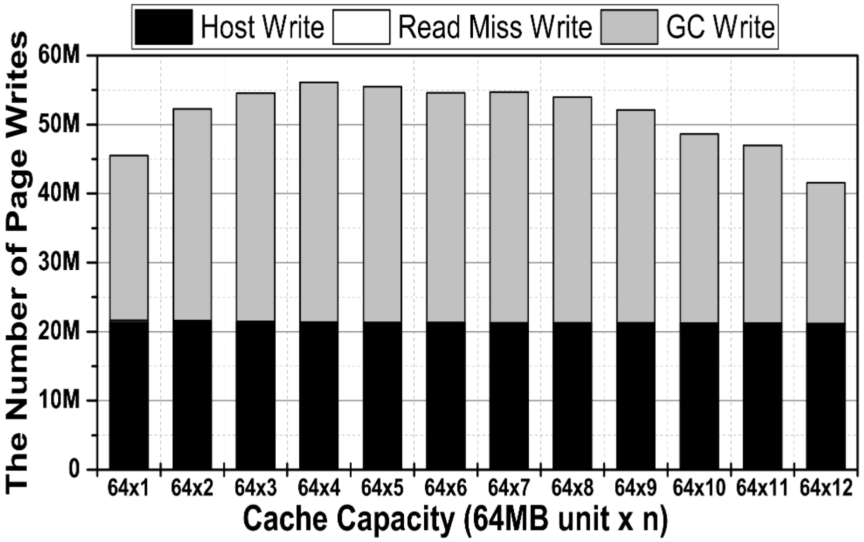
Proposed Allocation: Non-Linearity Aware DRF (nDRF)



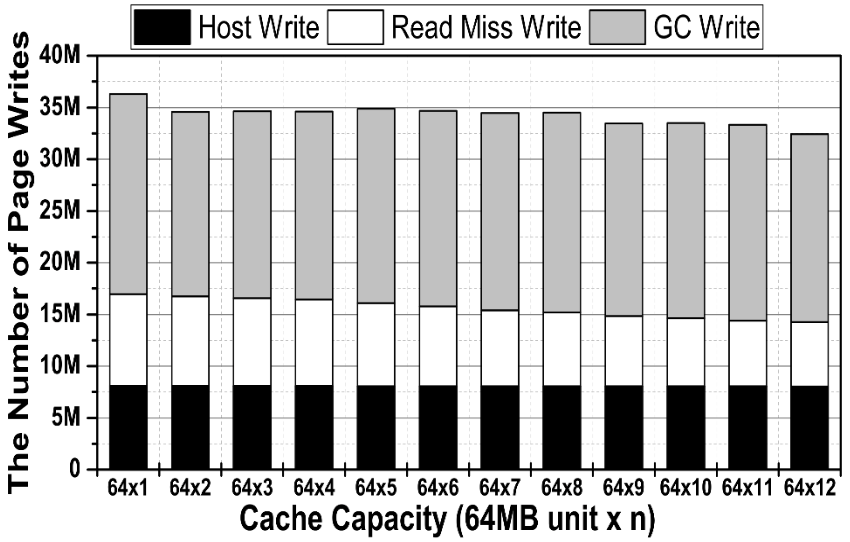
[Our modified PF works with non-linearity in resource demands]

- To embrace non-linear relationship in demands, we modified PF (called nDRF)
- Our modified PF algorithm works as follows:
  - ~~It does not allocate a fixed capacity and bandwidth increments (as in conventional PF)~~
  - It allocates a unit memory capacity increment and its corresponding bandwidth increment
  - This process continues until one of the two resources is fully allocated
- Specifically, nDRF jointly allocates memory capacity & bandwidth

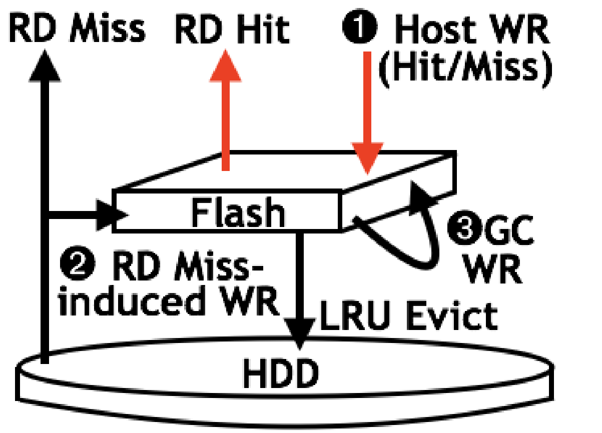
Observation: Flash Mem. Capacity vs Lifetime Demands



[Open Storage Trace - *prxy*]



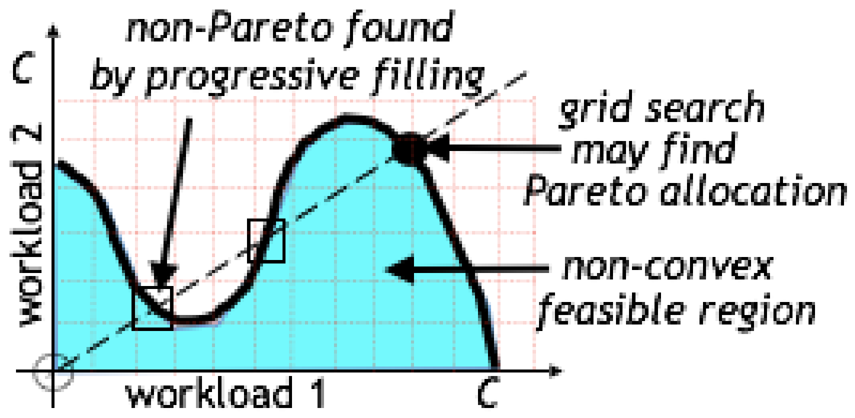
[Open Storage Trace - *prn*]



Lifetime (writes) consumers = ❶ + ❷ + ❸

- We characterized demands of mem. capacity & lifetime (cumulative # writes)
  - Lifetime (writes) consumption under varying memory capacity allocations
  - Lifetime consumer (1): ❶ all writes from the host/workloads
  - Lifetime consumer (2): ❷ writes from HDD to flash cache for read cache misses
  - Lifetime consumer (3): ❸ writes generated during garbage collection process
- Cum. # writes (lifetime) & mem. capacity demands are non-convex/concave

Proposed Allocation: Lifetime-Aware DRF (ℓDRF)



[Our grid search works with non-convex/concave relationship]

- **To embrace non-convex relationship of memory capacity and lifetime demands, we proposed grid search (called ℓDRF)**
  - The modified PF of nDRF may lead to a non-Pareto allocation
  - It evaluates all feasible allocations and finds the one that most equalizes dominant-shares
- **If exploration space is large, one can employ hierarchical approach**
  - (1) Exhaustive grid search with a large memory capacity unit
  - (2) Subsequently, exhaustive grid search with a small memory capacity unit
- **Specifically, ℓDRF jointly allocates capacity, bandwidth, & lifetime**

Example: An nDRF/ℓDRF Allocation

Total amounts of resources: capacity (1,280 MB), bandwidth (81,920 KB/s), # available writes (100,000,000)

Workload	Capacity (MB)	Bandwidth (KB/s)	Dominant-Resource	Dominant-Share
OST - prxy	640	24,180	Capacity	640/1,280 = 0.500
OST - web	576	3,852	Capacity	576/1,280 = 0.450
OST - proj	64	42,278	Bandwidth	42,278/81,920 = 0.516
Total Allocated	1,280	70,310		

[nDRF - memory capacity/bandwidth allocation, assuming writes are not a bottleneck/dominant-resource]

Workload	Cap (MB)	BW (KB/s)	Write (#)	Dominant-Resource	Dominant-Share
OST - prxy	64	23,973	45,538,103	Write	45M/100M = 0.450
OST - web	640	3,853	9,465,327	Capacity	640/1,280 = 0.500
OST - proj	64	42,278	44,525,155	Bandwidth	42,278/81,920 = 0.516
Total Allocated	768	70,105	99,528,585		

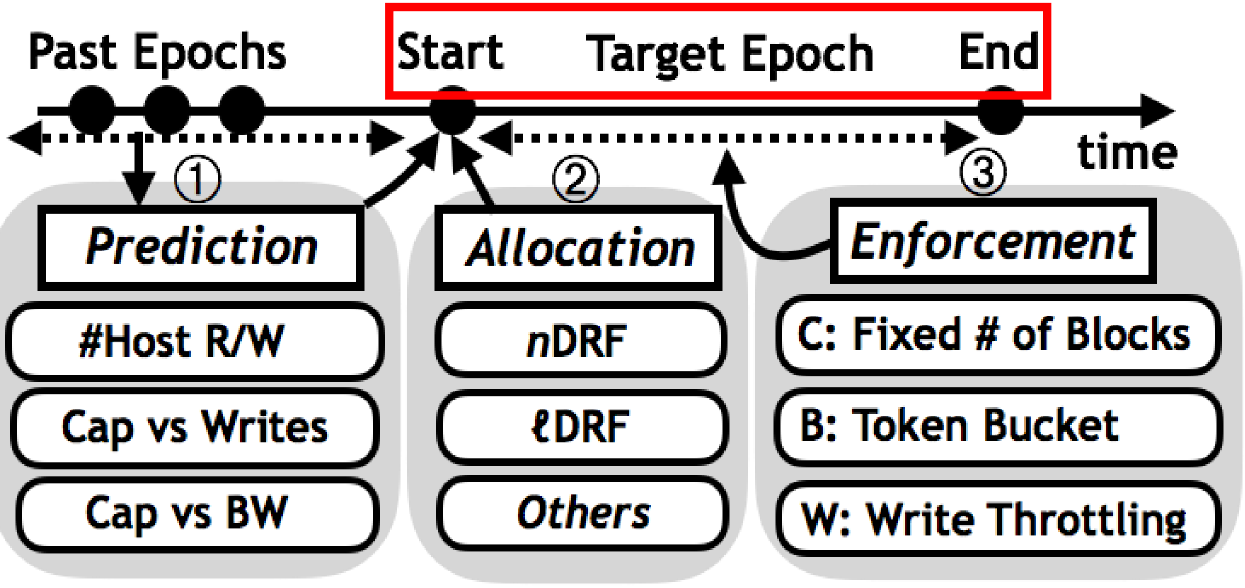
[ℓDRF - memory capacity/bandwidth/lifetime(write) allocation]

Comparison of Different Allocations

	Resource types considered in allocation			Lifetime management method	
Allocation Strategy	Capacity	Bandwidth	Lifetime	Throttling	Automatic
nDRF	O	O			
ℓDRF	O	O	O		O
nDRF + even throttling	O	O		O (even)	
nDRF + MMF throttling	O	O		O (MMF)	
EqualHR	O				
EqualHR + MMF throttling	O			O (MMF)	
MaxCumHR	O				
MaxCumHR + MMF throttling	O			O (MMF)	

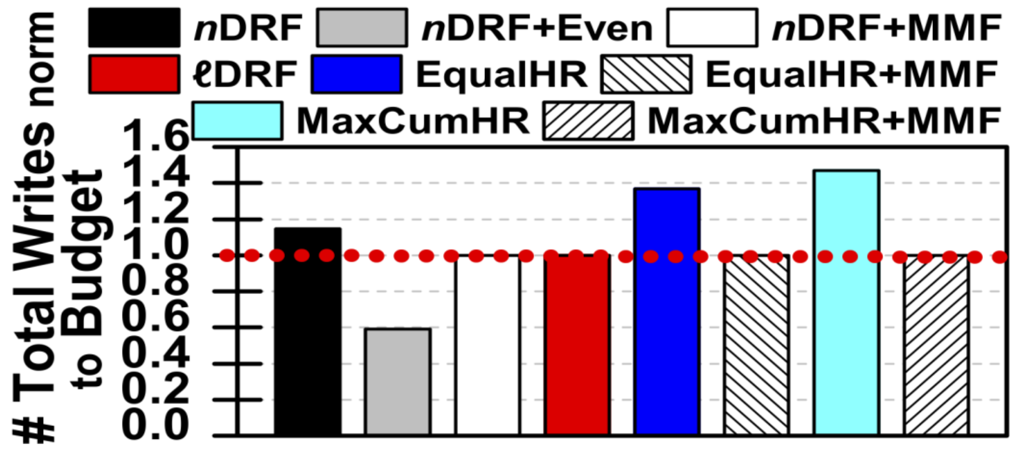
- **For (online) nDRF, lifetime can be managed by additional throttling mechanism**
  - Even throttling: total remaining writes are divided evenly, and each can use only allocated writes
  - MMF throttling: demand max-min fairness is used in dividing total available writes across workloads
- **Also, two often-used online memory capacity allocations: EqualHR (that equalizes hit ratios) & MaxCumHR (that maximizes aggregate hit ratio) across workloads**

Our Proposed Online Adaptive Framework

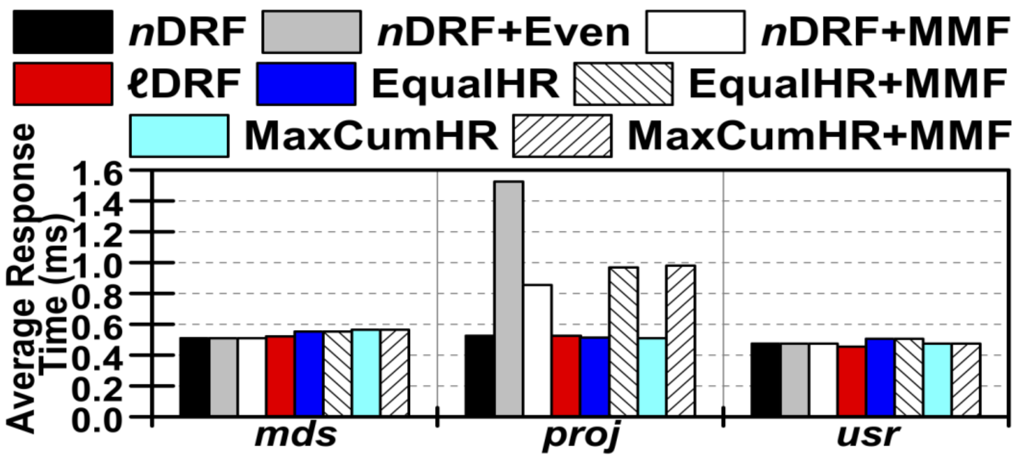


- We proposed an online resource allocation framework
  - A new allocation at every "epoch" – a period of relative workload stationarity
  - ① Demand prediction: predicts workloads' resource demands based on the near-past epochs
  - ② Resource allocation: performs an allocation at the beginning of the target epoch
  - ③ Allocation enforcement: enforces the allocation till the end of the epoch

Evaluation Result: in Offline Settings



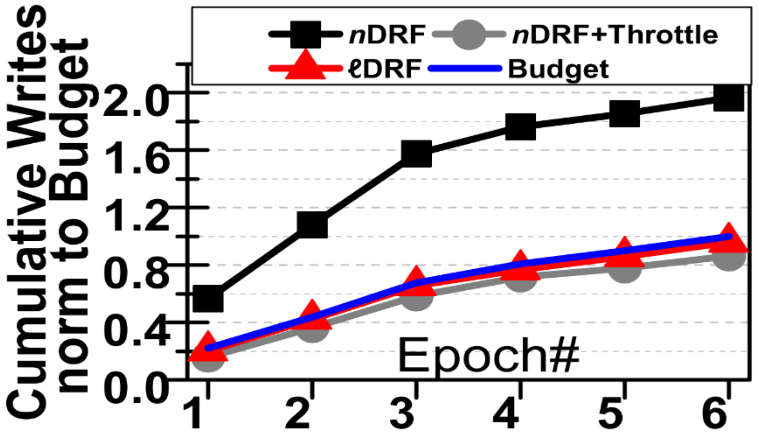
[# consumed writes / total # available writes]



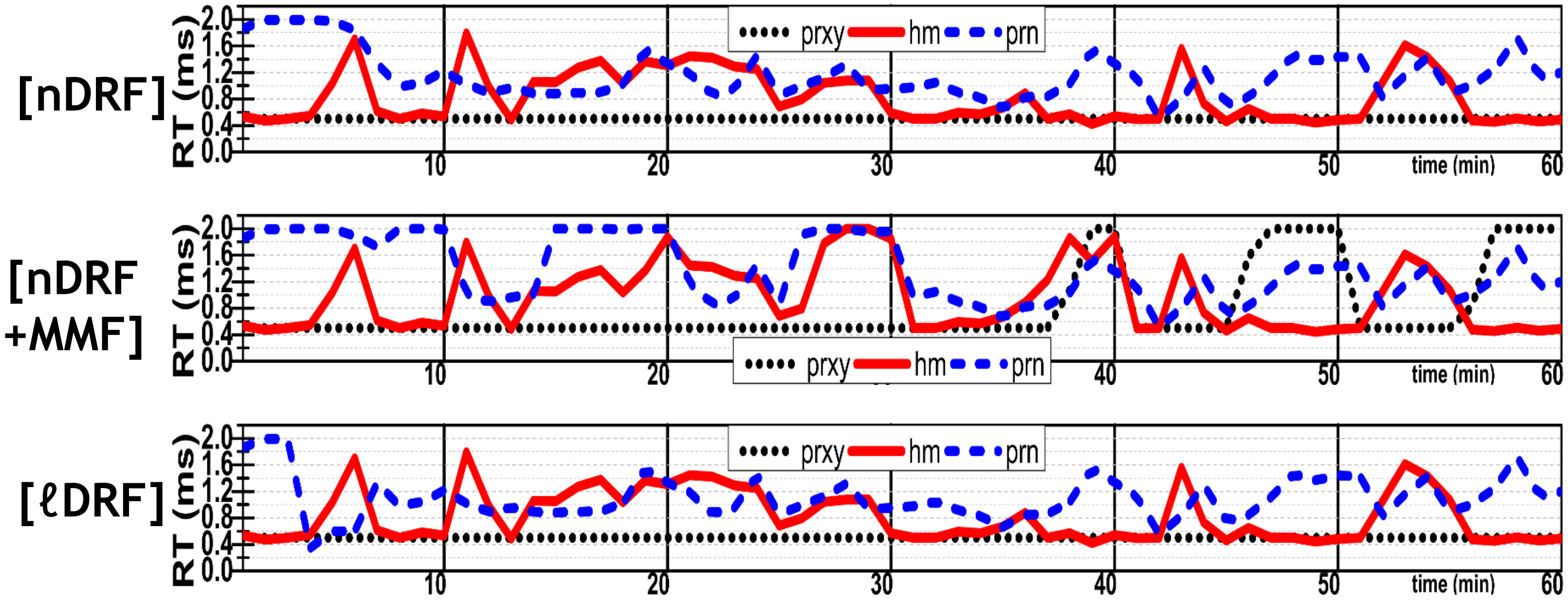
[Average response time]

- 8 different allocation strategies under a consolidation scenario (mds+proj+usr)
- Lifetime consumption (# consumed writes normalized to # available writes)
  - Lifetime-unaware allocations (nDRF, EqualHR, MaxCumHR) consume more writes than available
  - ℓDRF and MMF throttling can make consolidated workloads consume only available writes
- Performance fairness (how equitable response times are)
  - (Lifetime-unaware) nDRF achieves quite equitable response times
  - ℓDRF achieves quite equitable response times while managing the lifetime
  - Throttling-based allocations provide poorer performance fairness

Evaluation Result: in Online Settings



[# consumed writes normalized to total # available writes]



- 3 different allocation strategies under a consolidation scenario (prxy+hm+prn)
  - 6 consecutive 10-min epochs (1 hour execution & re-allocation at every 10 mins)
- ℓDRF/nDRF+MMF consume only available writes, while nDRF does not
- (Lifetime-unaware) nDRF provides quite equitable response times
- When lifetime is managed using throttling (nDRF+MMF), things get worse
- When lifetime is allocated based on ℓDRF, response times are quite equitable

Concluding Remarks

- Consolidating multiple workloads on a single flash device is a common practice
  - There may be a strong need of fair allocation of shared flash resources
- (1) Write allocation problem (device lifetime only)
  - Employed Shapley value for write attribution
  - Compared fair-looking allocation strategies using our framework
- (2) Multi-resource allocation problem (capacity + bandwidth + lifetime)
  - Employed and modified DRF for our context
  - Compared DRF, non-DRF, and others using our framework
- Applying classical concepts from other domains to flash problems
  - Requires accurate demand estimation
  - Assumes stationary workload behaviors