Fair Resource Allocation in Consolidated Flash Systems

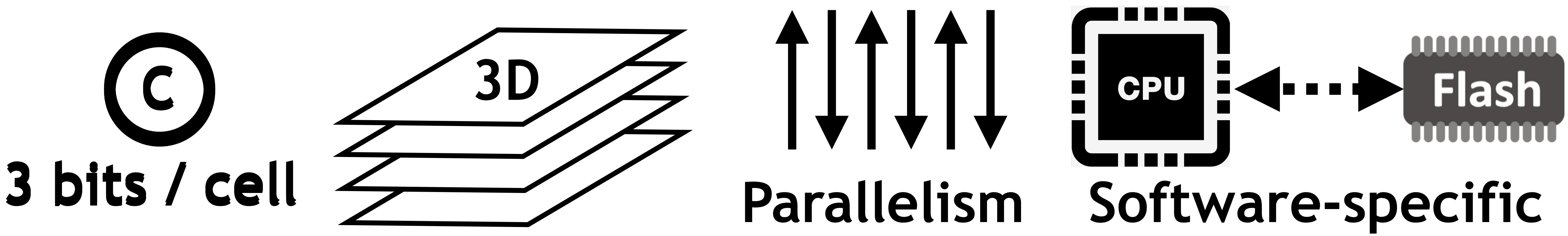
Wonil Choi, Bhuvan Urgaonkar, Mahmut Kandemir, and Myoungsoo Jung*

Pennsylvania State University, *KAIST**

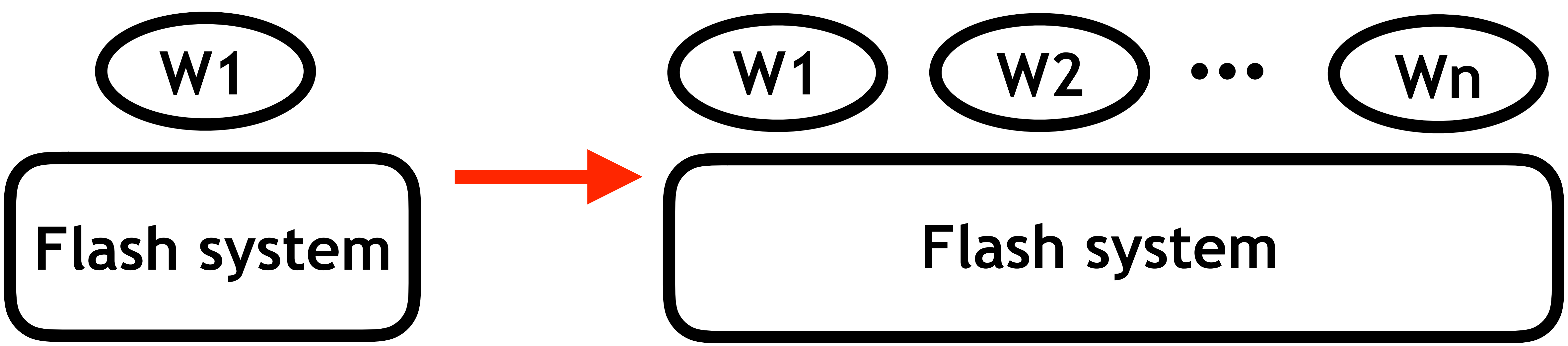
Trend: Consolidated Flash Systems

Flash/SSD technologies have become <u>mature</u>

(1) Massive capacity! **(2) High performance!**



Multiple workloads are <u>consolidated</u> in single system



Motivational Question: How Consolidated Workloads *Fairly* Share Multiple *Resources* in a System?

Consolidated workloads contend to use shared resources

W1 W2 ••• Wn

Different demands
Inter- & intra-

System' Resources

Different types
Limited amount

Our interest is to find a fair allocation of resources

Q1: What resource types do we allocate?

Q2: How can we coordinate different types of resources?

Q3: How can we achieve a fair allocation?

Question 1: What Resources Do We Allocate? - Resource Types of Flash System

Three critical resources are taken into account

(1) Bandwidth

Representative, prime resource in various domains

Literature: Huang [FAST'17], Kim [HotStorage'18]

(2) Capacity

If as main-storage, allocation should accommodate all

If as caching, allocation affects hit-ratio and performance

(3) Lifetime (flash-specific)

We propose to view device lifetime as a critical resource

It is consumable/non-renewable (# page writes is limited)

It is also consumed implicitly (GC writes)

It is coupled with capacity resource (OP-GC relationship)

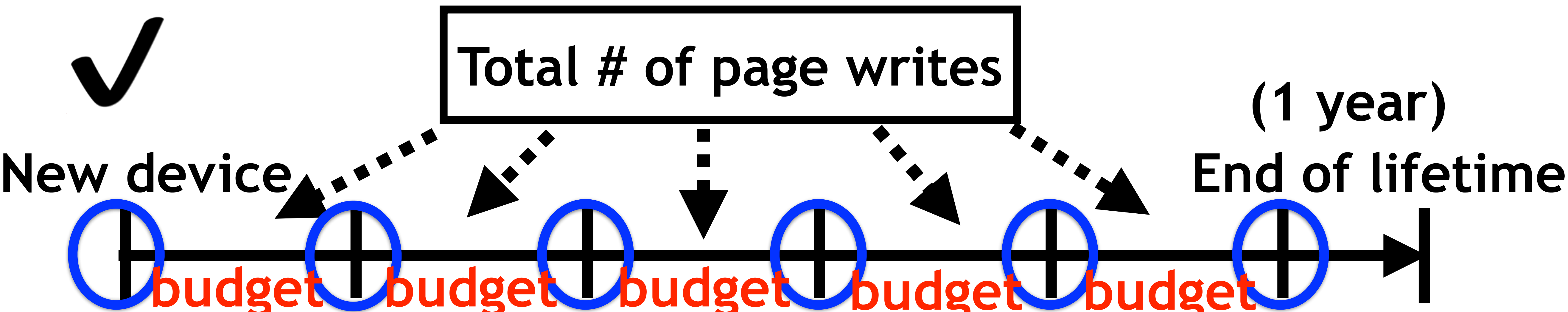
Question 2: How Treating Lifetime on Equal Footing with Bandwidth and Capacity? - Coordination of Three Resources

Bandwidth & capacity can be allocated at small time scale

When workloads change their resource demand patterns

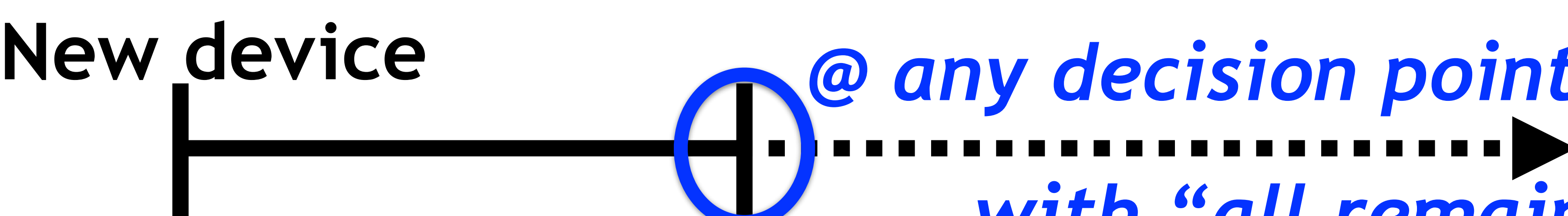
How can we treat lifetime allocation?

Approach (1): goal of ensuring that device lasts for 1 year



new allocation with "given # of writes (budget)"

Approach (2): only interested in fairly dividing the remaining

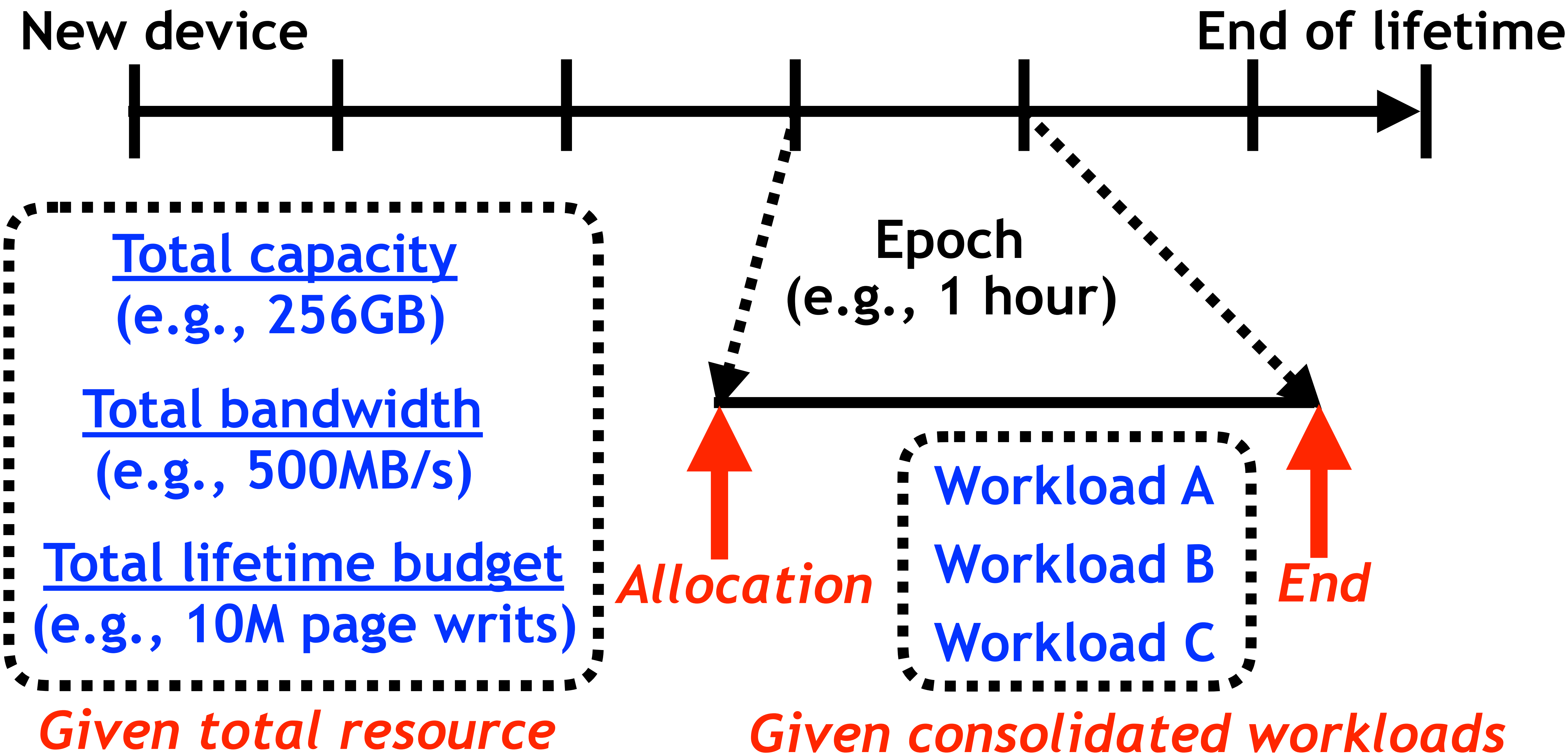


@ any decision point, new allocation with "all remaining writes"

Question 2: How Treating Lifetime on Equal Footing with Bandwidth and Capacity?
- Epoch-based Resource Allocation

Epoch: a period of relative workload stationarity

We propose to allocate resources in each & every epoch



Question 3: How Fair Allocation Can be Achieved? - Dominant Resource Fairness (DRF) [NSDI'11]

Example: two users share two resource types, CPU and Mem

* Total resources: <9 CPU, 18 GB>
* User1's demand: <1 CPU, 4 GB> per task
* User2's demand: <3 CPU, 1 GB> per task

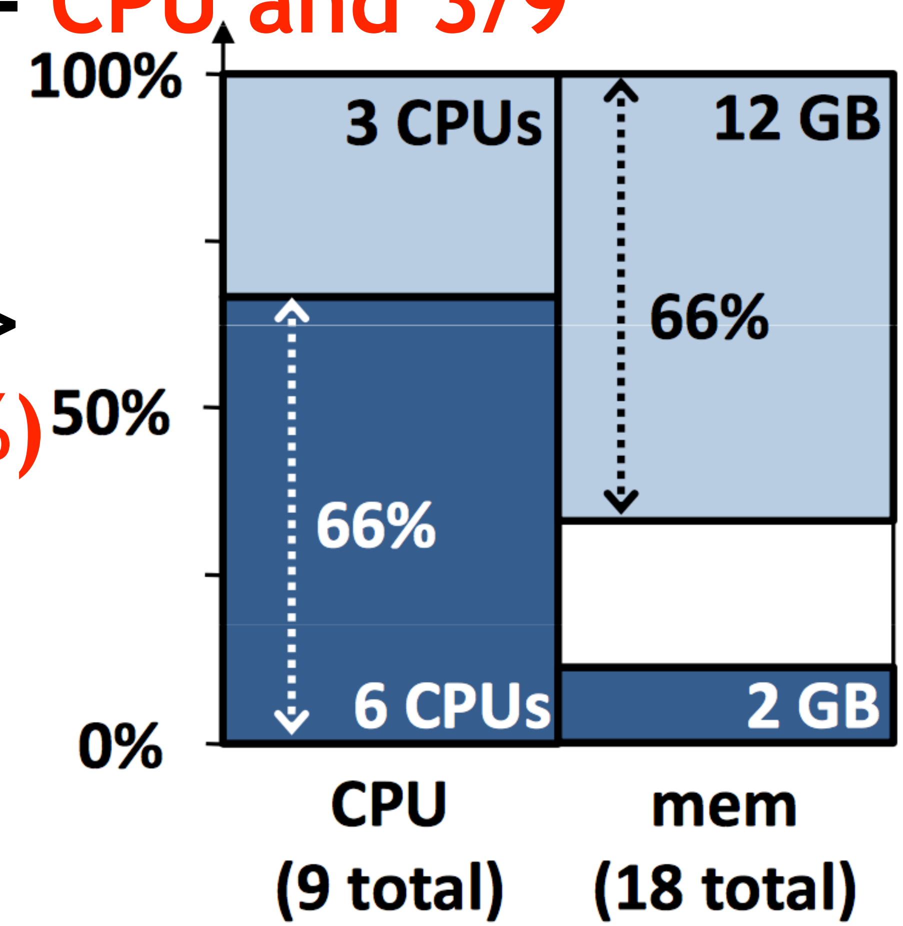
Dominant resource and dominant share (alloc / total amount)

* User1's demand: 1/9 CPU < 4/18 GB - Mem and 4/18
* User2's demand: 3/9 CPU > 1/18 GB - CPU and 3/9

DRF equalizes dominant shares of users

User 1 alloc: 3 tasks <3 CPU, 12 GB>
- dominant share: 12/18 (66%)

User 2 alloc: 2 tasks <6 CPU, 2 GB>
- dominant share: 6/9 (66%)



Proposal: DRF Adaptation in Flash Device Context - Experimental Setup

We propose to adapt DRF to flash resource allocation

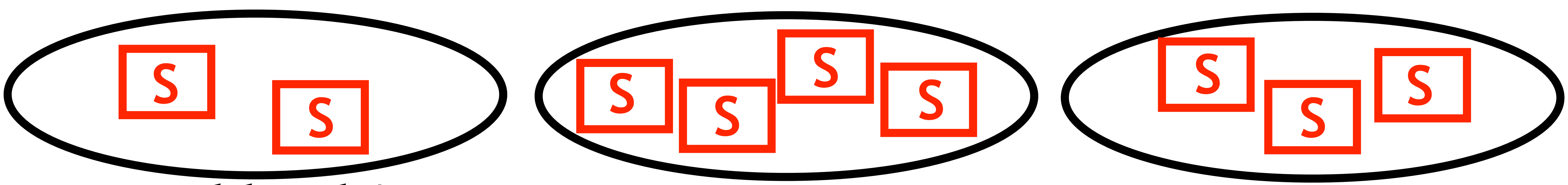
We assume an epoch with given 3 resources

Bandwidth: 512 MB/s
Capacity: 256 GB (flash cache)
Lifetime (write budget): 11M page writes

We assume a set of 3 workloads running in the epoch

DRF works based on workloads' standardized resource demands

Our target workload has its own executable unit (stream)



Workload A	Workload B	Workload C
Req:RD(0.1M)/WR(1M)	RD(1M)/WR(0.1M)	RD(50M)/WR(0.3M)
Share(20GB)/Private(5GB)	S(60GB)/P(20GB)	S(30GB)/P(10GB)
Write Amp(3.0)	Write Amp(1.2)	Write Amp(1.5)
Lifetime-dominant	*Capacity-dominant*	*Bandwidth-dominant*

Proposal: DRF Adaptation in Flash Device Context - Allocation Algorithm and Result

DRF algorithm

(i) pick a random workload & launch its stream

until no more streams can be launched due to lack of resources

(ii) calculate *dominant share* of each workload

(iii) pick the workload whose dominant share is *minimum*

(iv) *launch its stream* towards equal dominant share

DRF allocation result

Workload	# Streams Launched	Bandwidth Alloc	Capacity Alloc	Write Alloc	Dominant Share	Dominant Resource
A	2	7 MB/s	30 GB	**6 M**	6/11.2 M = **0.535**	*Write*
B	4	5 MB/s	**140 GB**	0.48 M	140/256 GB = **0.546**	*Capacity*
C	5	**274 MB/s**	80 GB	2.25 M	274/512 MB/s = **0.535**	*Bandwidth*
Total		512 MB/s	256 GB	11.2M		

Observation 1: Considering Bandwidth & Capacity Only - DRF without Lifetime Management

What if lifetime resource is not explicitly managed?

Workloads are allowed to consume as many writes as they need

Bandwidth & capacity are still considered

DRF allocation result

Workload	# Streams Launched	Bandwidth Alloc	Capacity Alloc	Dominant Share	Dominant Resource	Resultant Writes
A	13	44 MB/s	85 GB	85/256 GB = 0.332	*Capacity*	39M
B	2	3 MB/s	100 GB	100/256 GB = 0.391	*Capacity*	0.24M
C	4	219 MB/s	70 GB	219/512 MB/s = 0.427	*Bandwidth*	1.8M
Total		512 MB/s	256 GB			41.04M

Ignoring lifetime results in

Total write consumption significantly increases (41.04M)

Writes across workloads look quite unfair (39M, 0.24M, 1.8M)

Observation 2: Considering Shared Data among Streams - Non-Linearity in Capacity Demand

Vanilla DRF assumes uniform per-task resource demands

Total demands *linearly* increases, as #task increases

In a storage workload, its streams can share data

Capacity demand *non-linearly* increases, as #streams increases

E.g., shared(20GB)/private(5GB);#streams=1(25GB),2(30GB),…

So far, DRF allocations considered data sharing across streams

What if data sharing across streams is not considered?

DRF allocation result

Workload	# Streams Launched	Bandwidth Alloc	Capacity Alloc	Write Alloc	Dominant Share	Dominant Resource
A	2	8 MB/s	50 GB	**6 M**	6/11.2 M = **0.535**	*Write*
B	1	2 MB/s	**80 GB**	0.12 M	80/256 GB = **0.313**	*Capacity*
C	2	164 MB/s	**120 GB**	1.35 M	120/256 GB = **0.468**	*Capacity*
Total		512 MB/s	256 GB	11.2M		

Dominant resource of C changes from bandwidth to capacity

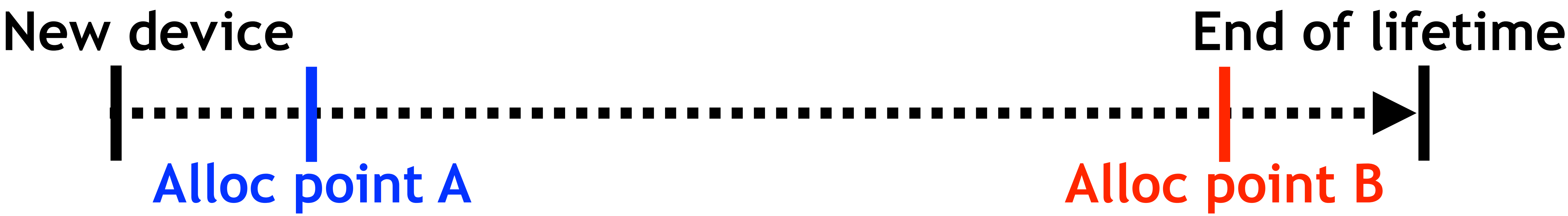
Observation 3: Considering Different Lifetime Policy - Budget = What Lifetime Remains

One can prefer <u>different lifetime management options</u>

Device can last for a longer time? -> tight write budget

Only interested in fairness? -> Budget = what lifetime remains

What if <u>write budget = what lifetime remains?</u>



Alloc point A (earlier part of lifetime)

- Budget is set to a very *large* number
- *No* workloads have lifetime as dominant resource
- This is the same as *DRF without lifetime management*

Alloc point B (latter part of lifetime)

- Budget is set to a very *small* number
- *All (or most)* workloads have lifetime as dominant resource

Conclusions

Consolidating multiple workloads in a single flash system
- Such workloads contend to get shared system resources

Various resource types in a flash system
- Conventional (bandwidth & capacity) & flash-specific (lifetime)

Need of fair resource allocation among consolidated workloads

We propose to employ Dominant Resource Fairness (DRF)
- DRF is said to be fair due to its desirable properties

Our DRF adaptation in flash system empirically reveals,
- Lifetime is a critical resource that needs to be managed
- Non-linearity in resource demand should be considered