RICE

Efficient QoS for Multi-Tiered Storage Systems

Ahmed Elnably

Hui Wang

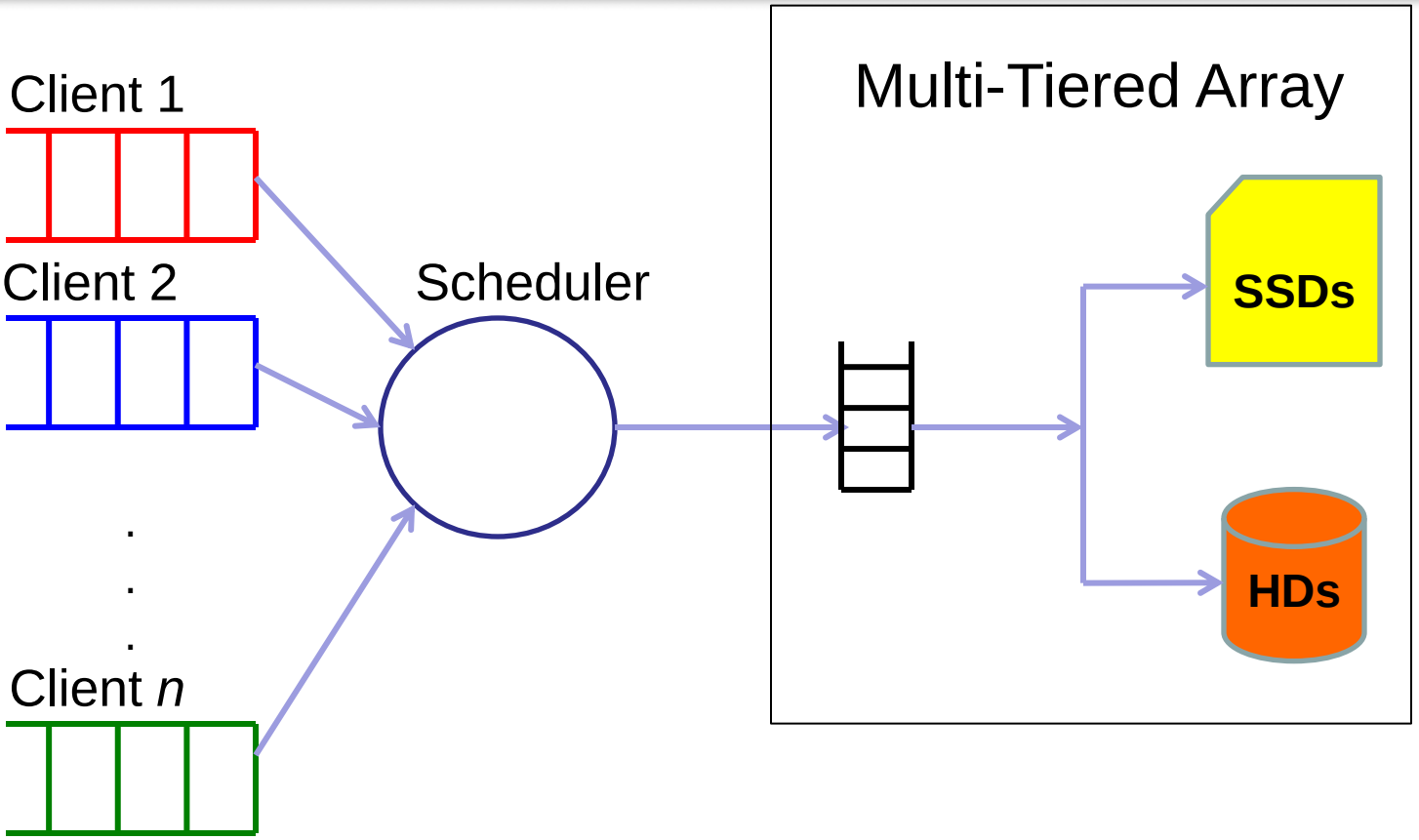
Peter Varman

Rice University

Ajay Gulati

VMware Inc

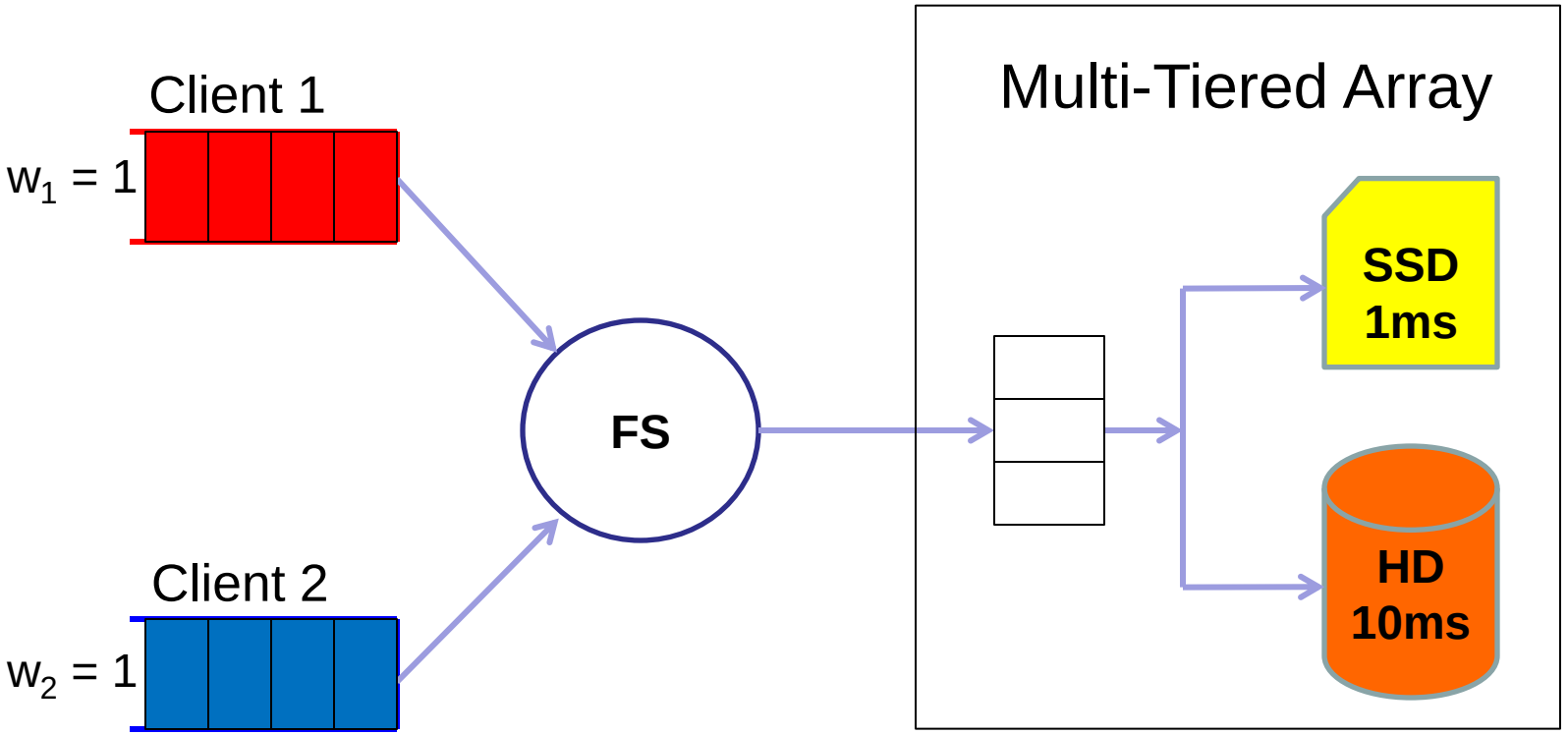
Tiered Storage Architecture



Problem: How to provide QoS in this environment?

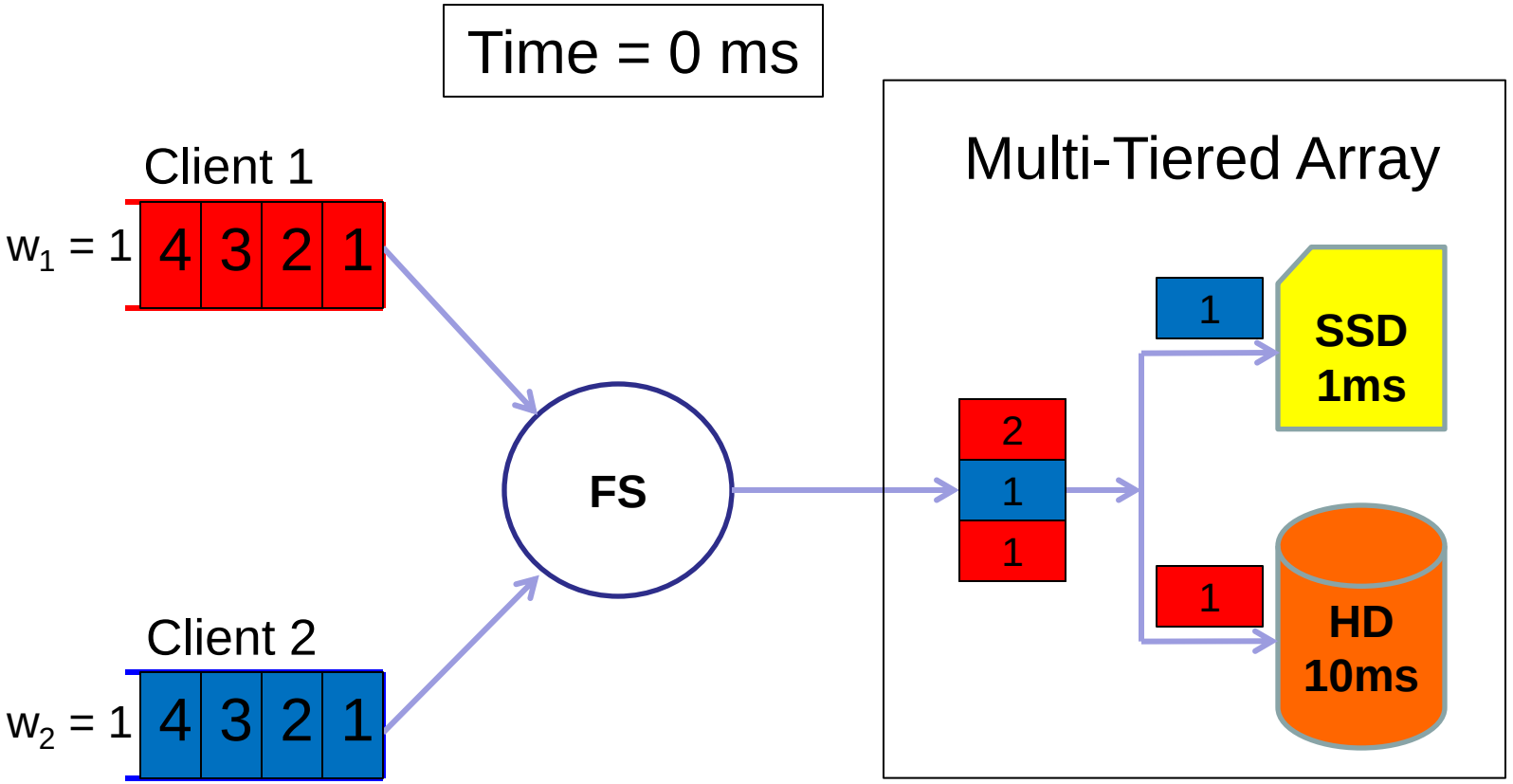
Fair Scheduling Example

Allocate IOs in proportion of client weights

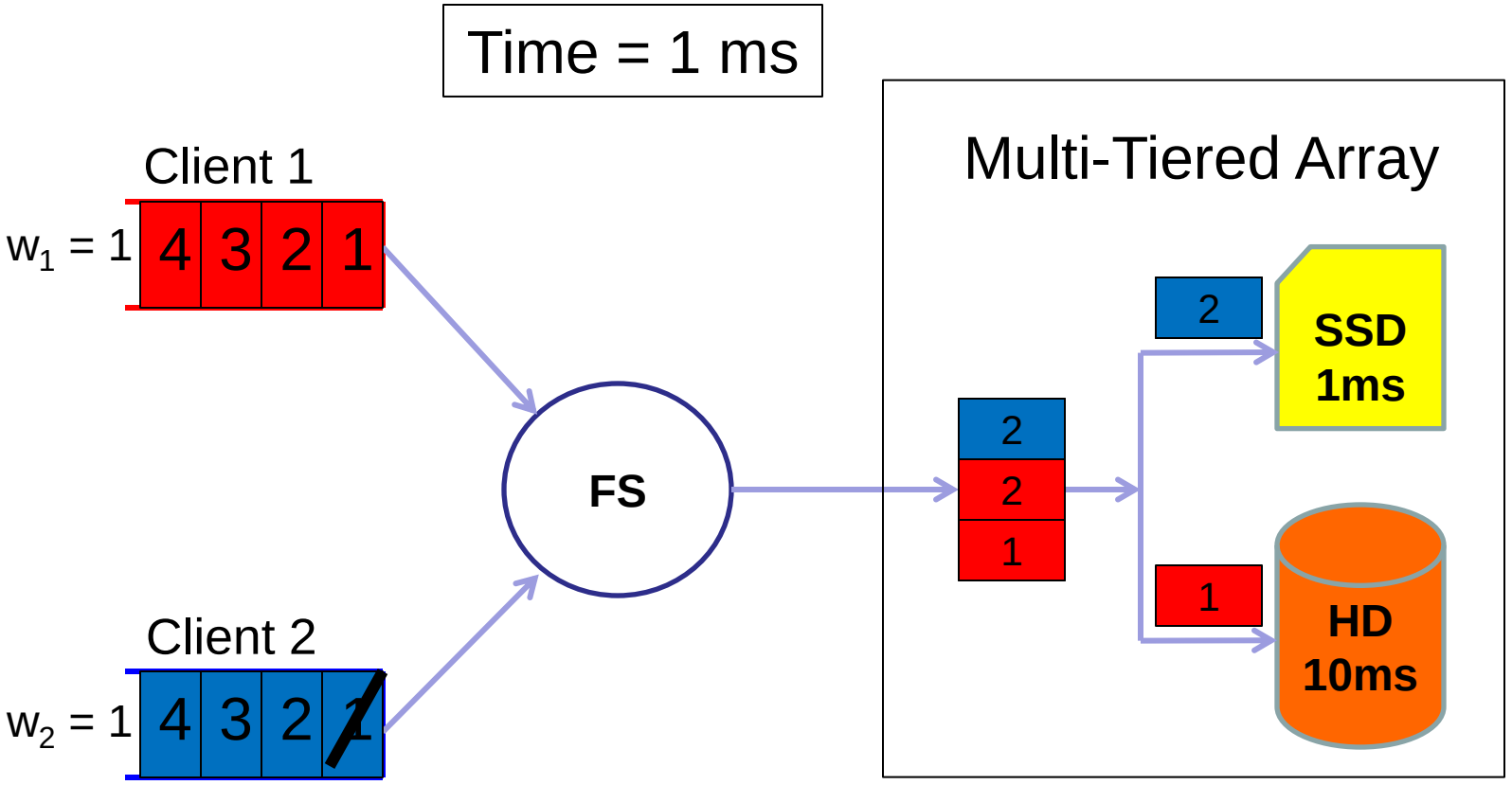


How will Fair Scheduling behave in a multi-tiered storage architecture?

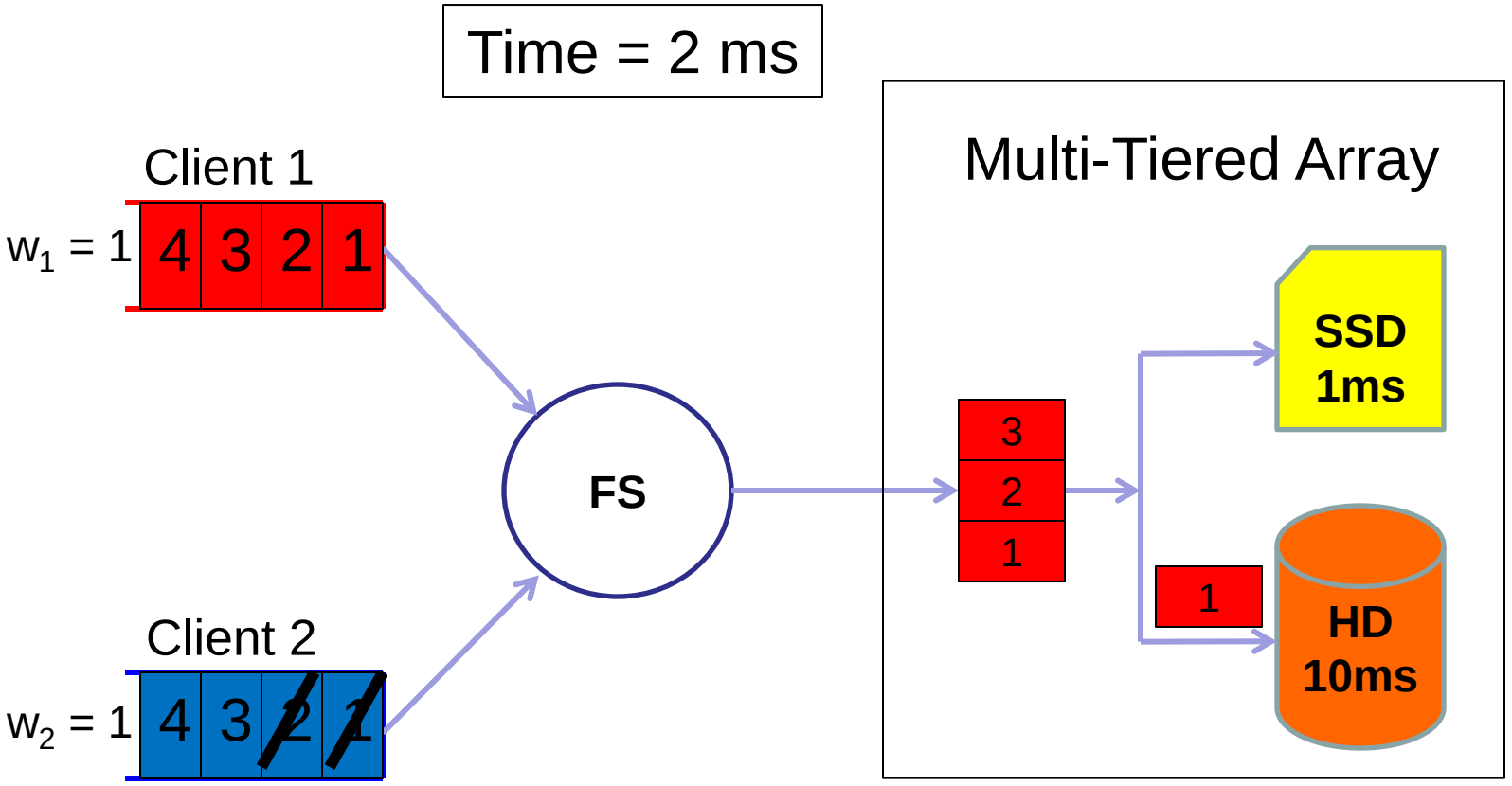
Fair Scheduling Example



Fair Scheduling Example



Fair Scheduling Example



Issue: Low utilization
SSD will idle for 9 out of 10 ms

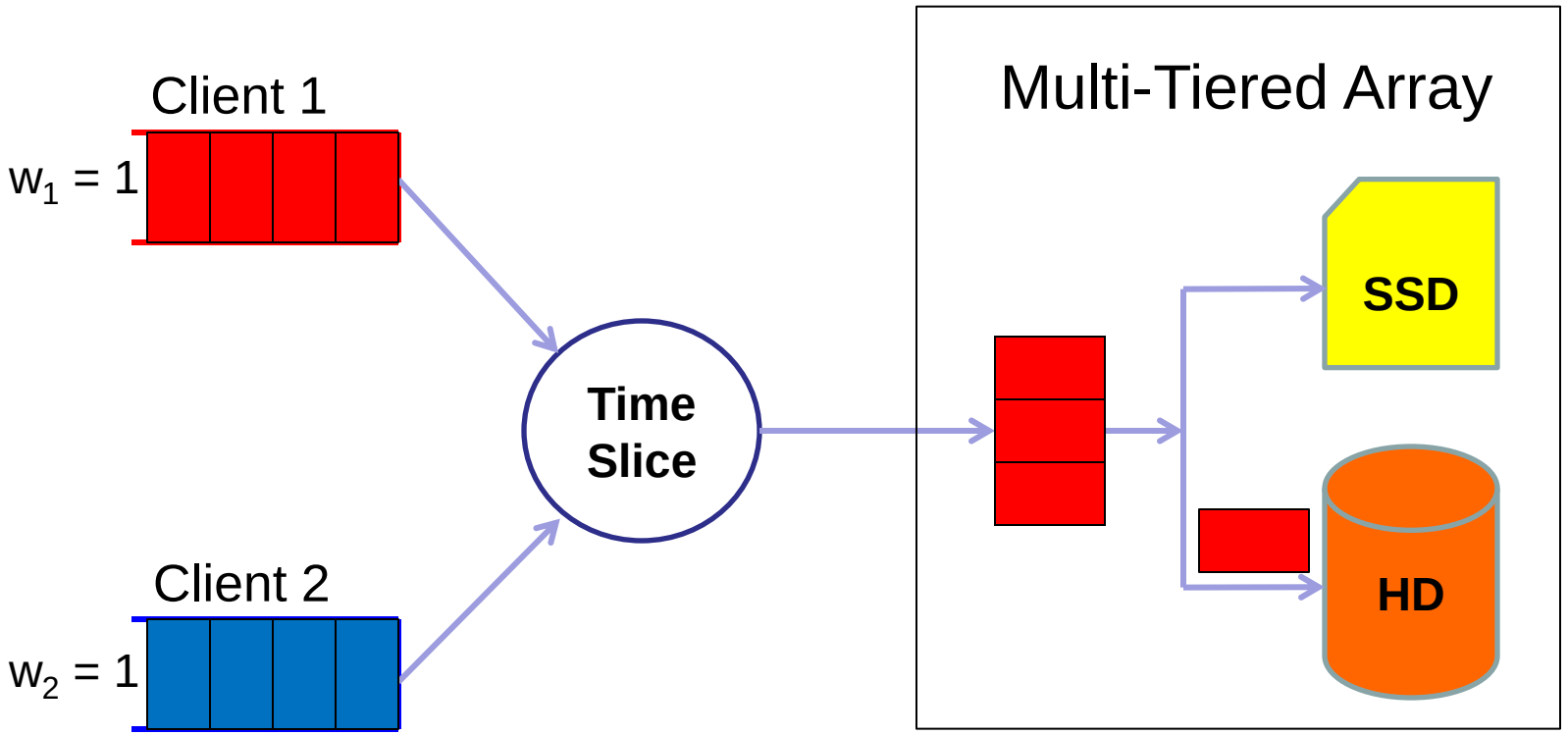
Fair Scheduling in Multi-Tier

Is the behavior of FS reasonable?

The answer is simply … NO!

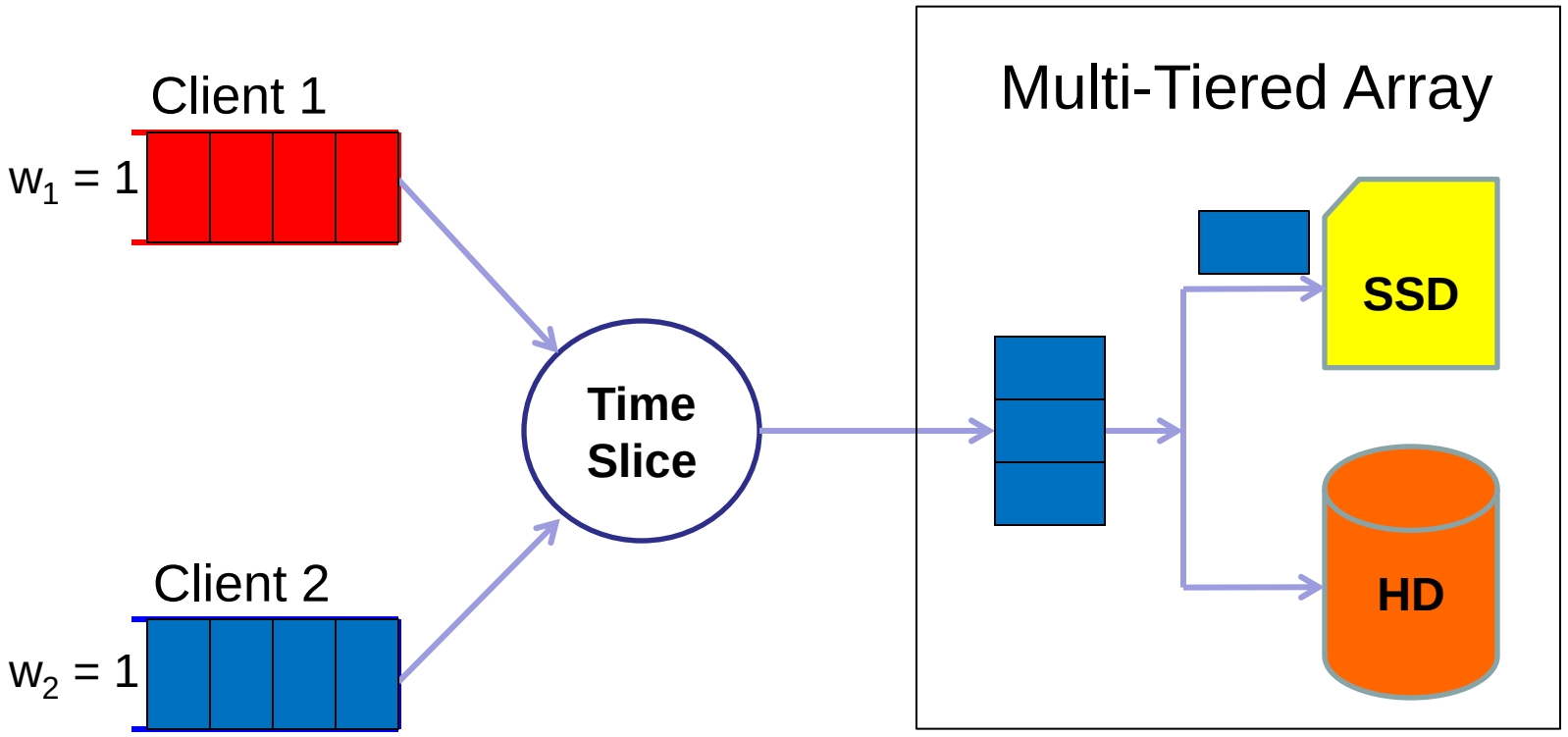
Time Slice Schedulers

Allocate time quantum in proportion of client weights



Time Slice Schedulers

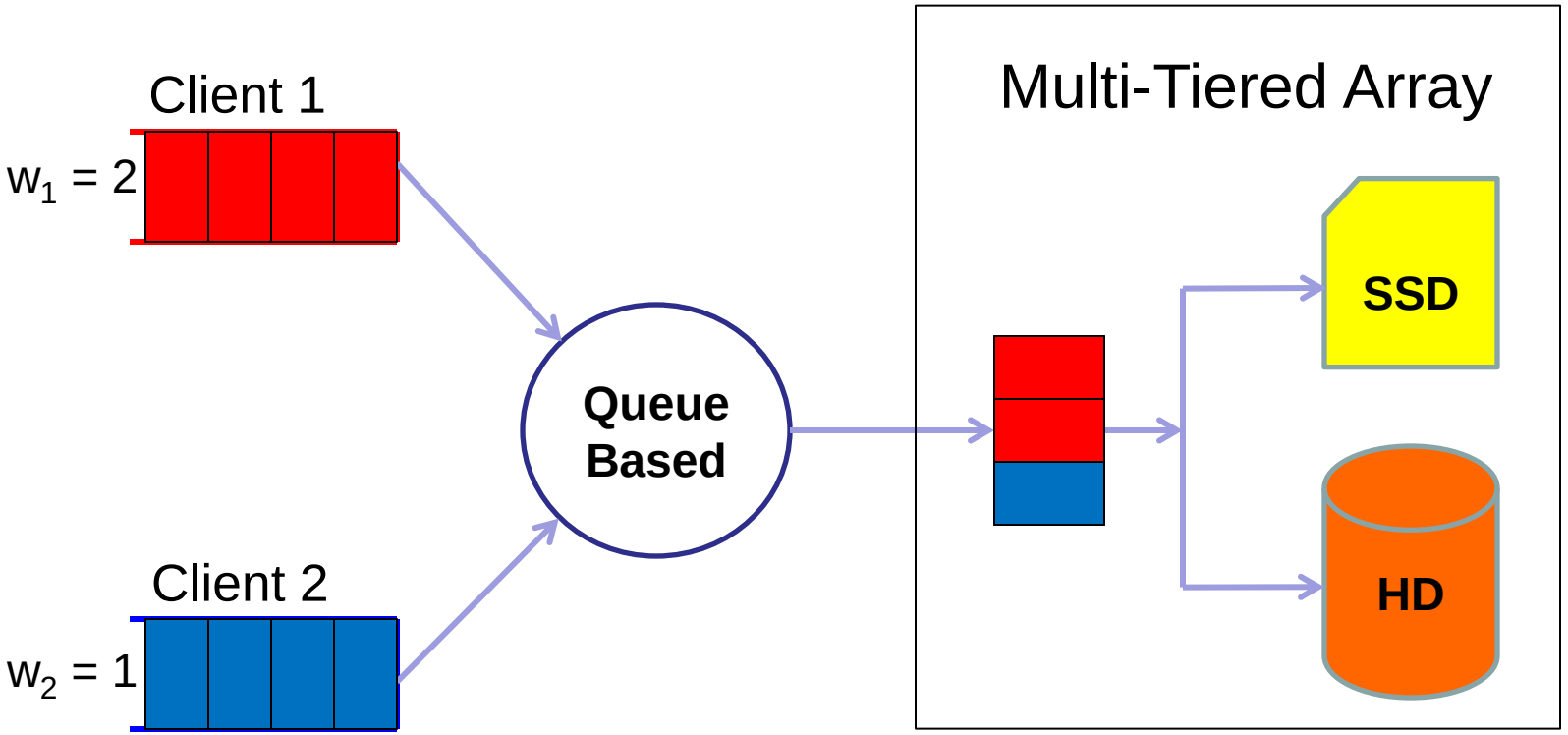
Allocate time quantum in proportion of client weights



Issue: Low utilization
Both devices will idle for large intervals

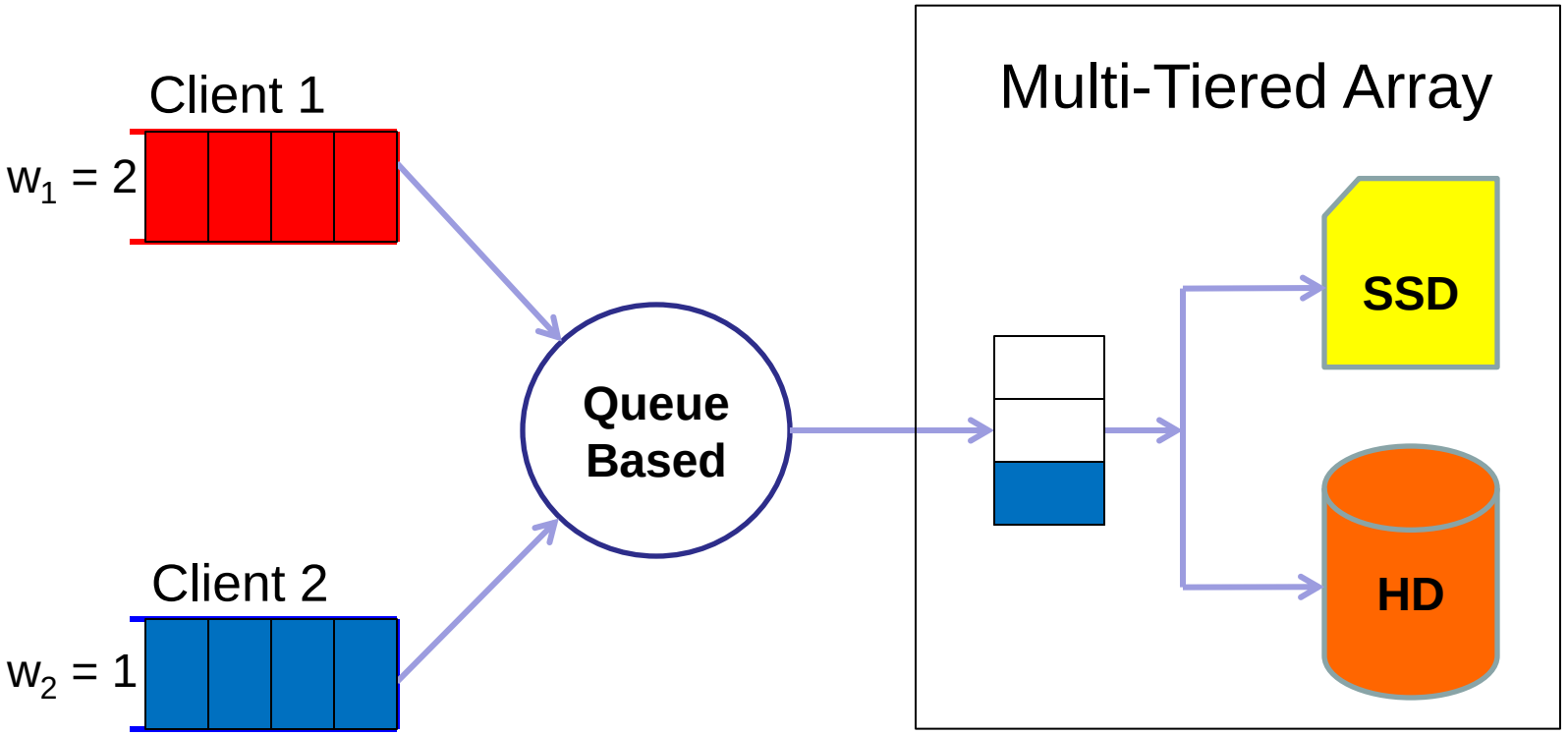
Queue Based Schedulers

Allocate queue slots in proportion of client weights



Queue Based Schedulers

Allocate queue slots in proportion of client weights



Issue: Static queues can cause low utilization
Empty slots are not getting re-used

Agenda

- Problem Motivation
- Reward Scheduling
- Preliminary Evaluation
- Open Issues & Conclusion

Reward Allocation Policy

- Goals
  - Provide QoS (fairness)
  - Higher system throughput / device utilization

Reward Scheduling

- Key Ideas:
  - Reward clients with cheaper IOs
  - Track IO latency of each client separately
  - Try to allocate in the ratio of clients' throughput if running alone (entitlement)

Algorithm Details

- Tag based scheduling
  - Each client queue has a tag
  - Tag represents scheduling priority
  - Scheduler selects queue with smallest tag
- Monitor service time of client requests
- Increment tags using
  - Measured service time of client
  - Weight of the client

Algorithm Details

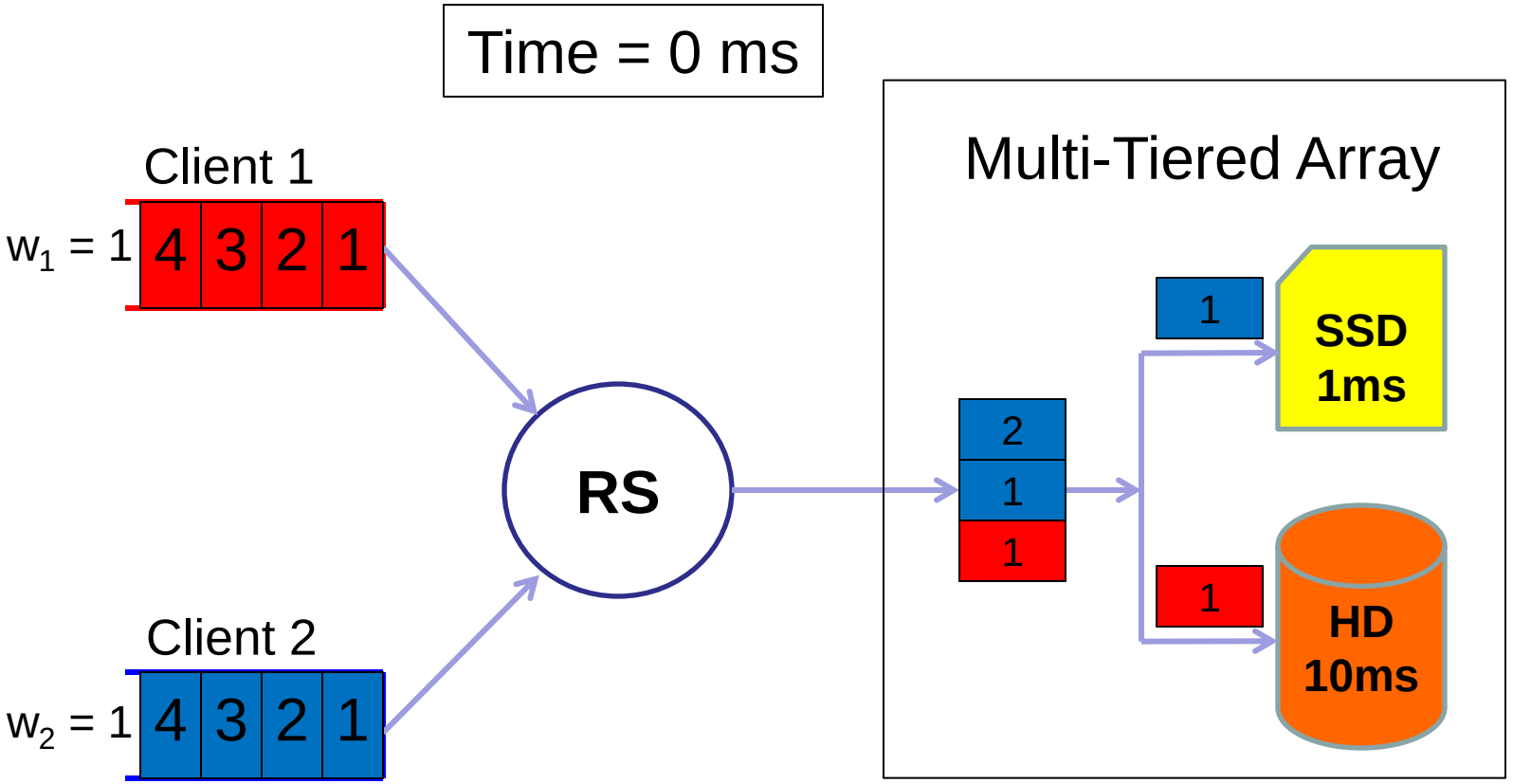
- Φ_j : Measured service time of completing request of client j
- UpdateServiceTime (j, Φ_j)
  - Average of the service time of client j over window

$$\bar{\Phi}_j = \text{UpdateServiceTime}(j, \Phi_j);$$

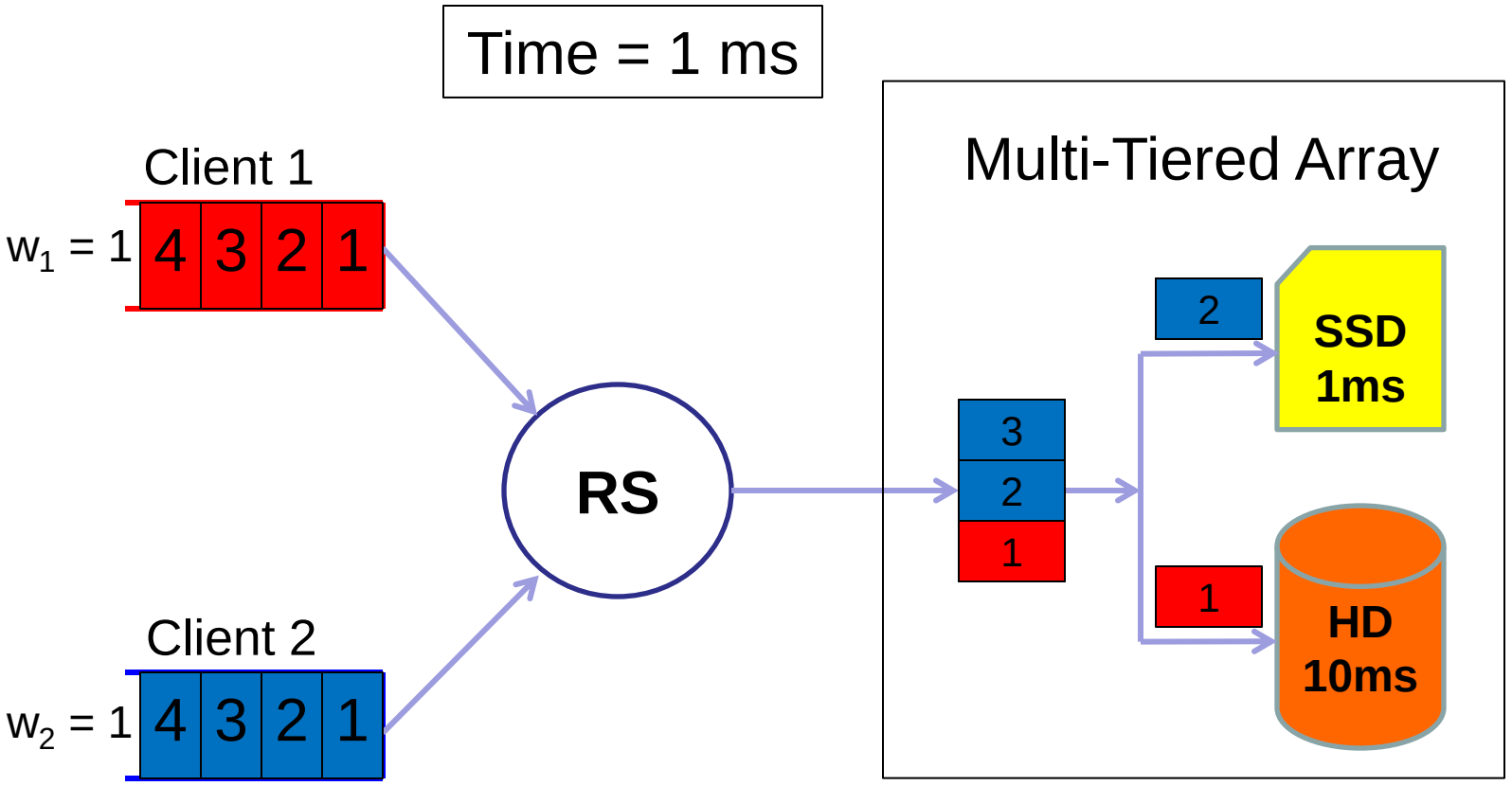
Remove completed request from queue;

$$\text{sTag}_j = \text{sTag}_j + \bar{\Phi}_j / \omega_j;$$

Reward Scheduling



Reward Scheduling



Higher utilization
Client with lower latency gets higher effective weight

Agenda

- Problem Motivation
- Reward Scheduling
- Preliminary Evaluation
- Open Issues & Conclusion

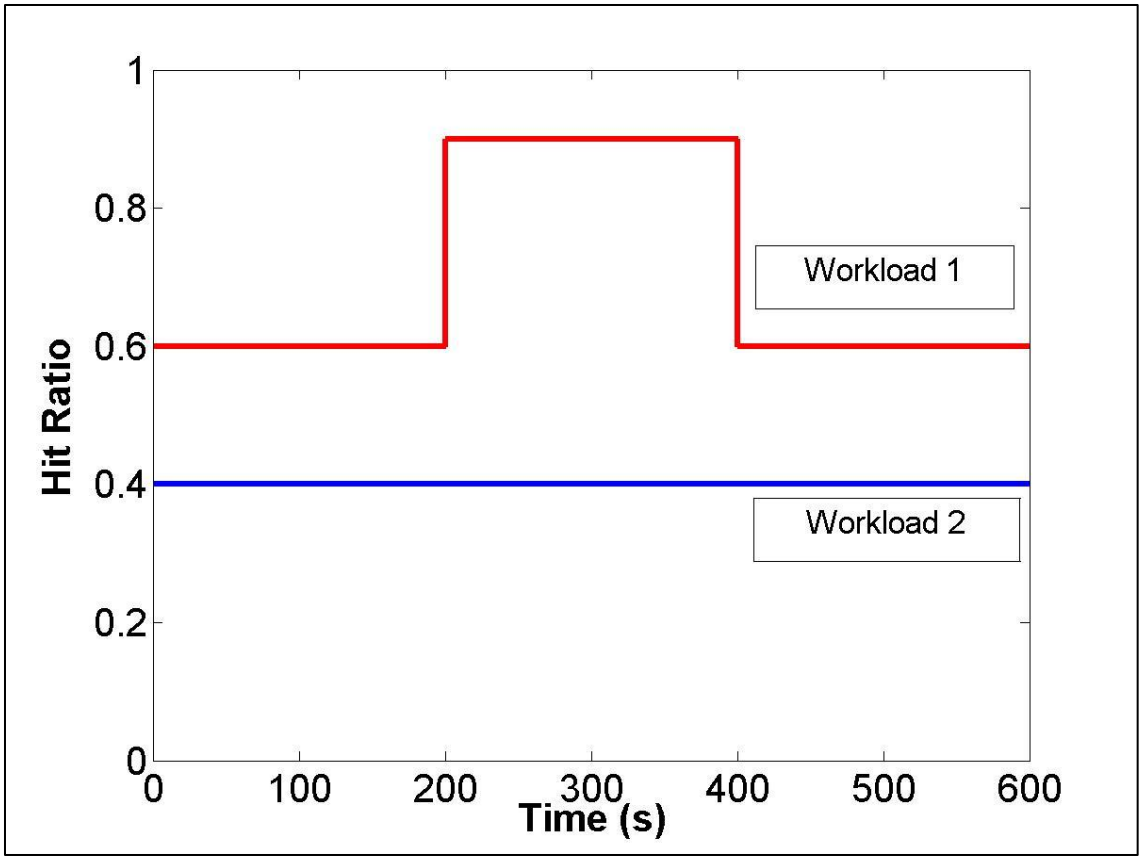
RICE

Experimental Setup

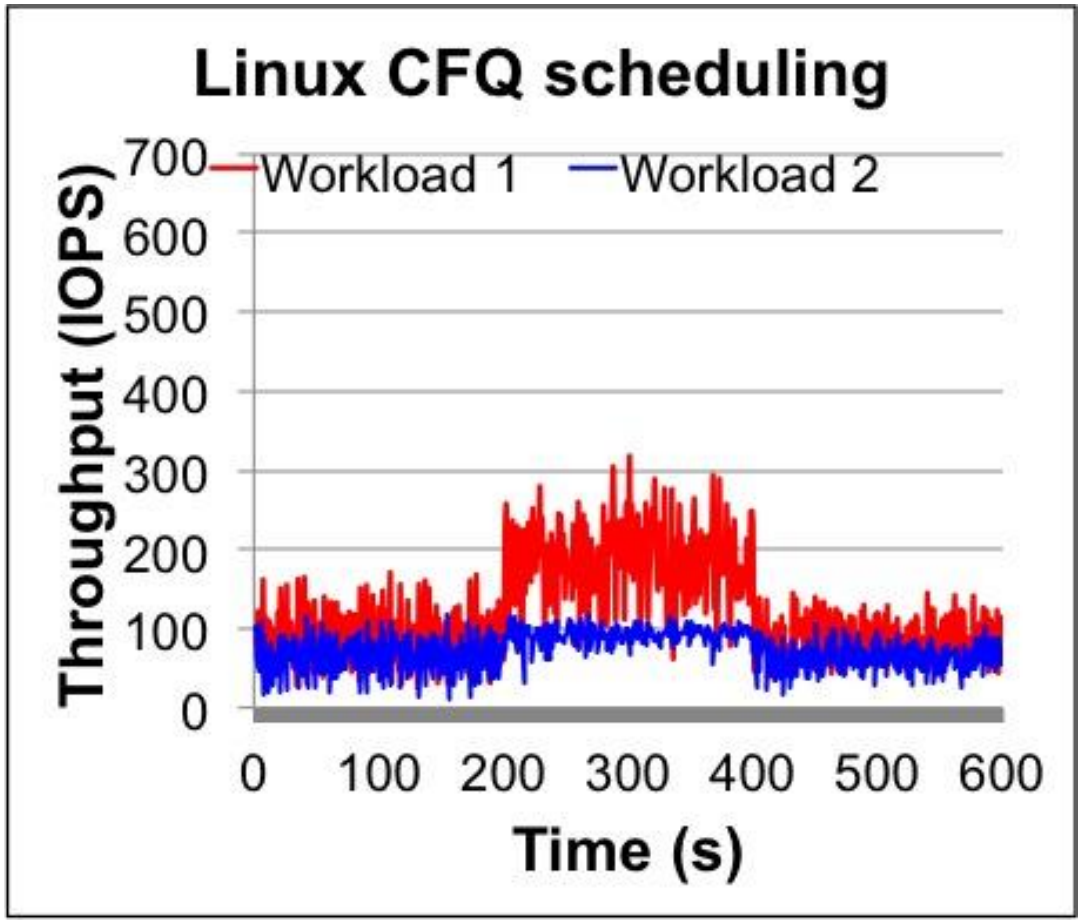
- Two implementations
  - Simulation-based
  - Linux-based evaluation (Interposing a reward scheduler in user space)
- Three different cases
  - Variable hit ratios
  - Variable read-write ratio
  - Variable weights (In paper)

RICE

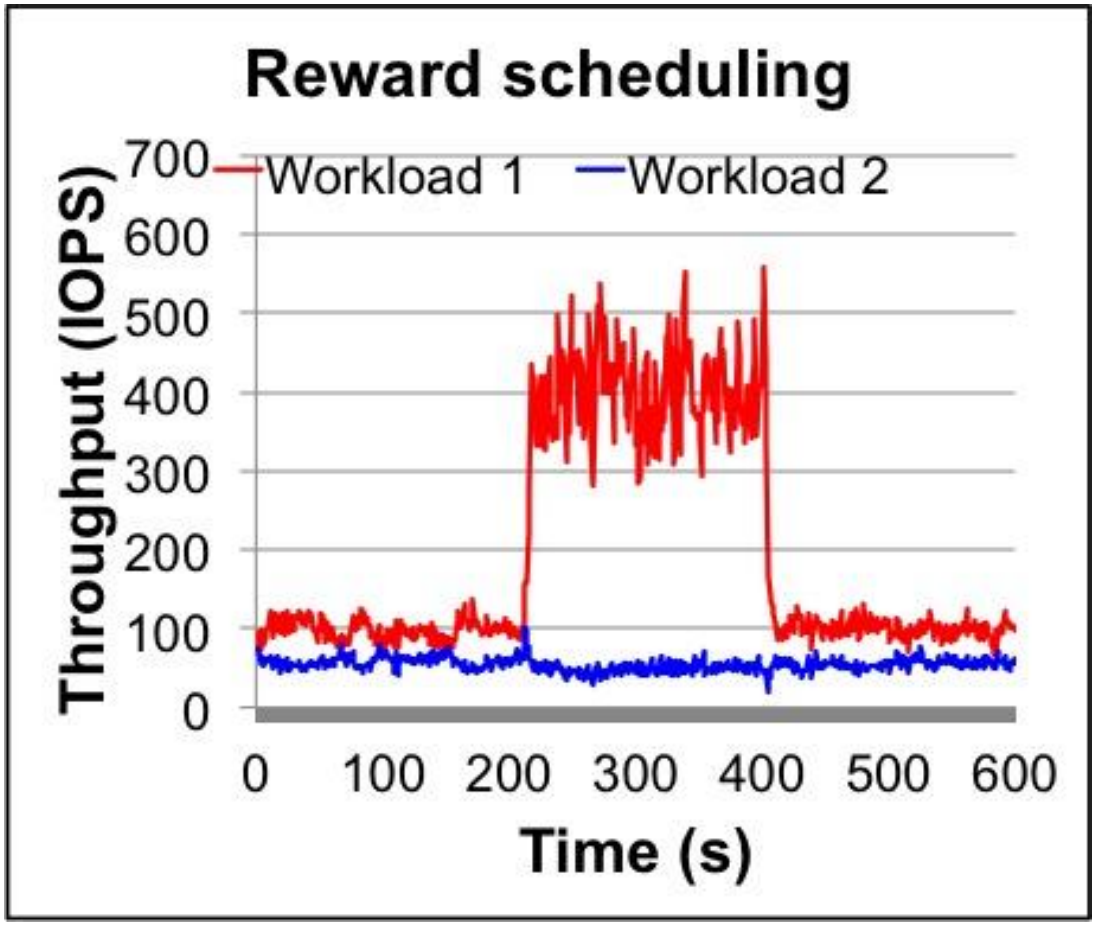
Reward Scheduling vs. CFQ



Reward Scheduling vs. CFQ

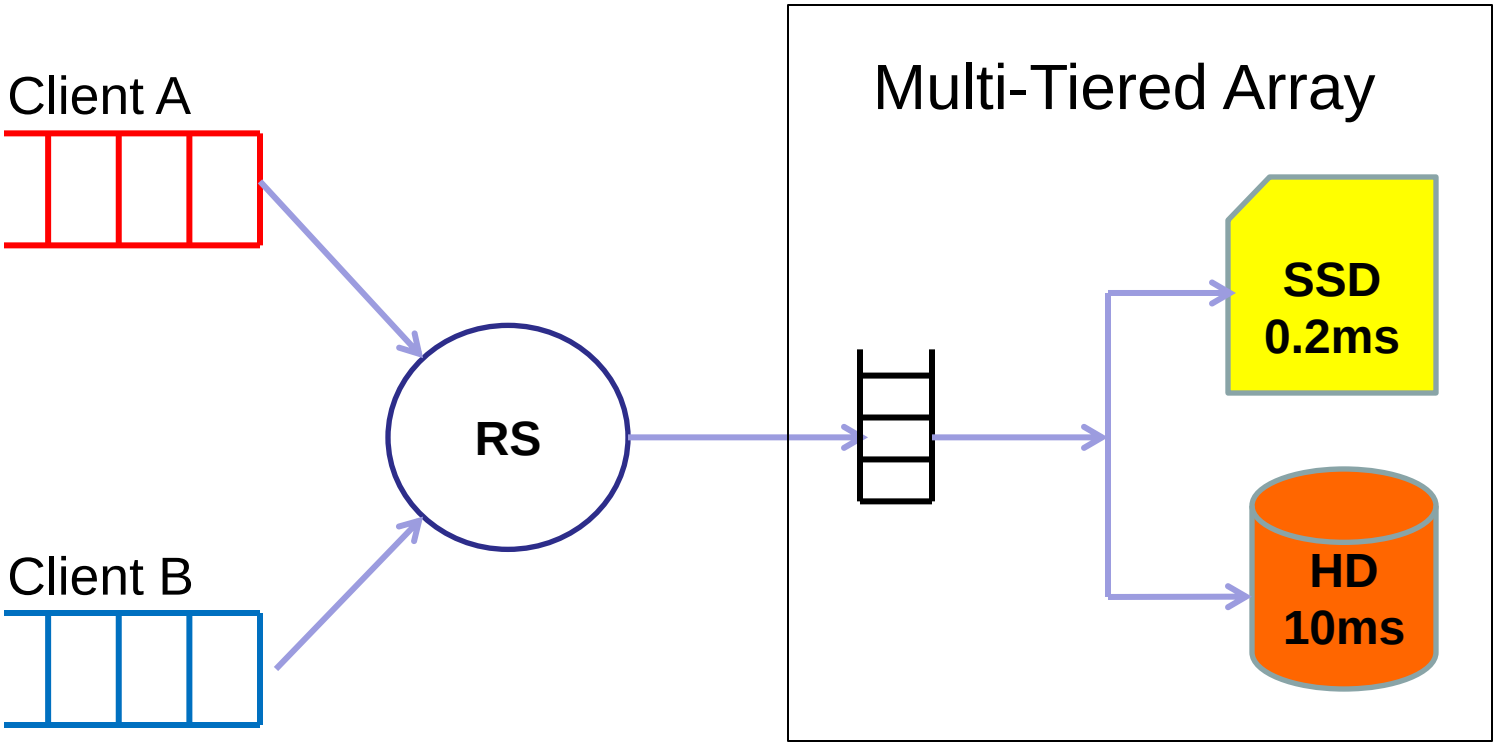


Reward Scheduling vs. CFQ



Isolation between workloads
Higher system throughput

Variable Hit Rate



Variable Hit Rate

- Simulation system

Weights	hit-ratio(A:B)	IOPS (A)	IOPS (B)	IOPS-ratio (A/B)
1:1	1 : 0.2	4923	122	40.4
	1 : 0.5	4841	193	25.1
	1 : 0.75	4612	373	12.4
	1 : 1	2474	2474	1.0

Entitlement of A = 5000 IOPS
Entitlement of B = 5000 IOPS
Ratio A:B = 5000:5000
1:1

Variable Hit Rate

- Simulation system

Weights	hit-ratio(A:B)	IOPS (A)	IOPS (B)	IOPS-ratio (A/B)
1:1	1 : 0.2	4923	122	40.4
	1 : 0.5	4841	193	25.1
	1 : 0.75	4612	373	12.4
	1 : 1	2474	2474	1.0

Entitlement of A = 5000 IOPS
Entitlement of B = 200 IOPS
Ratio A:B = 5000 : 200
25 : 1

Variable Hit Rate

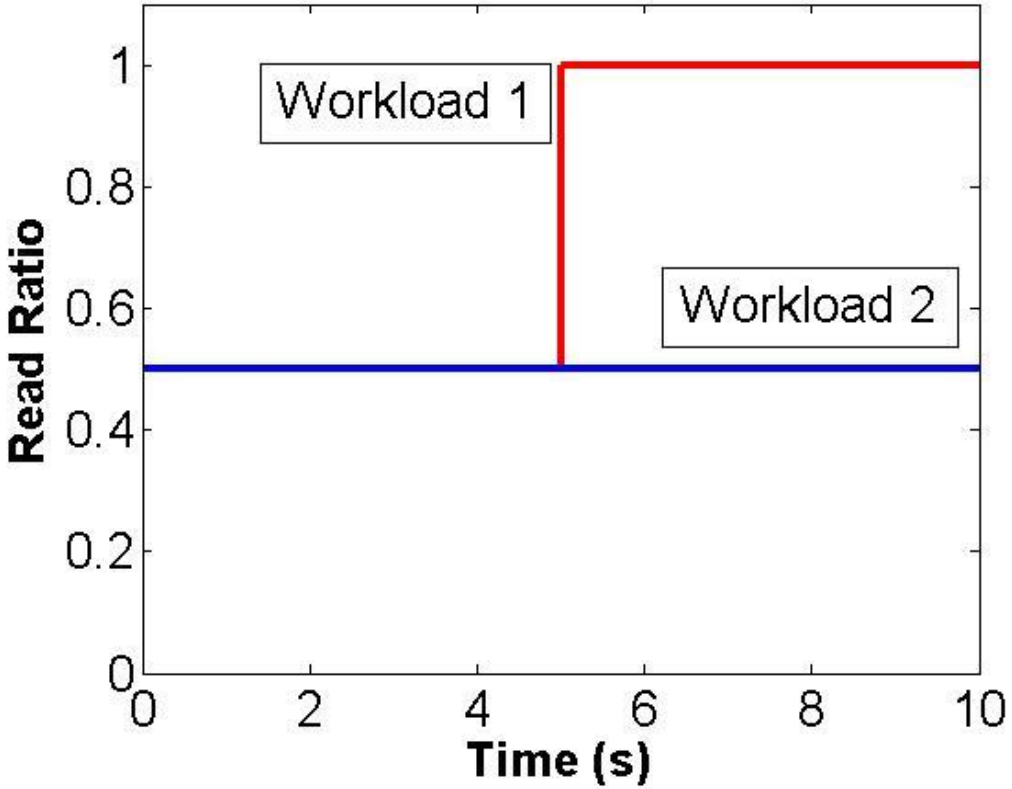
- Linux-based evaluation

Weights	hit-ratio(A:B)	IOPS (A)	IOPS (B)	IOPS-ratio (A/B)
1:1	1 : 0.2	3106	75	41.4
	1 : 0.5	3055	129	23.7
	1 : 0.75	2816	295	9.6
	1 : 1	2000	2033	0.98

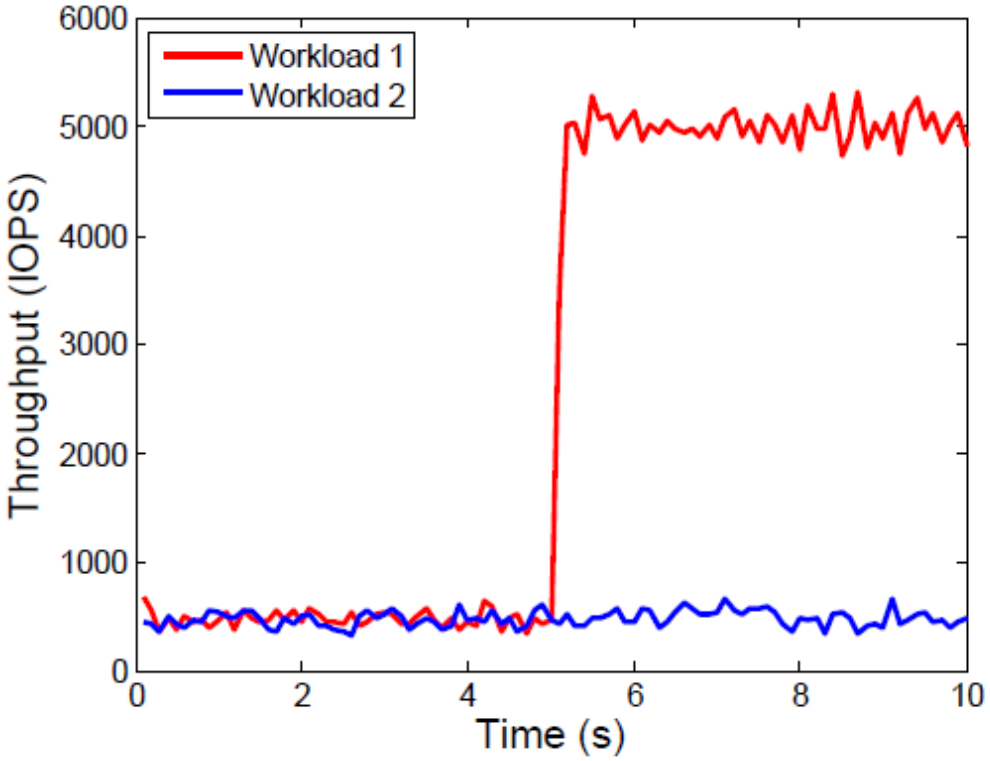
- Simulation system

Weights	hit-ratio(A:B)	IOPS (A)	IOPS (B)	IOPS-ratio (A/B)
1:1	1 : 0.2	4923	122	40.4
	1 : 0.5	4841	193	25.1
	1 : 0.75	4612	373	12.4
	1 : 1	2474	2474	1.0

Variable Read-Write Ratio

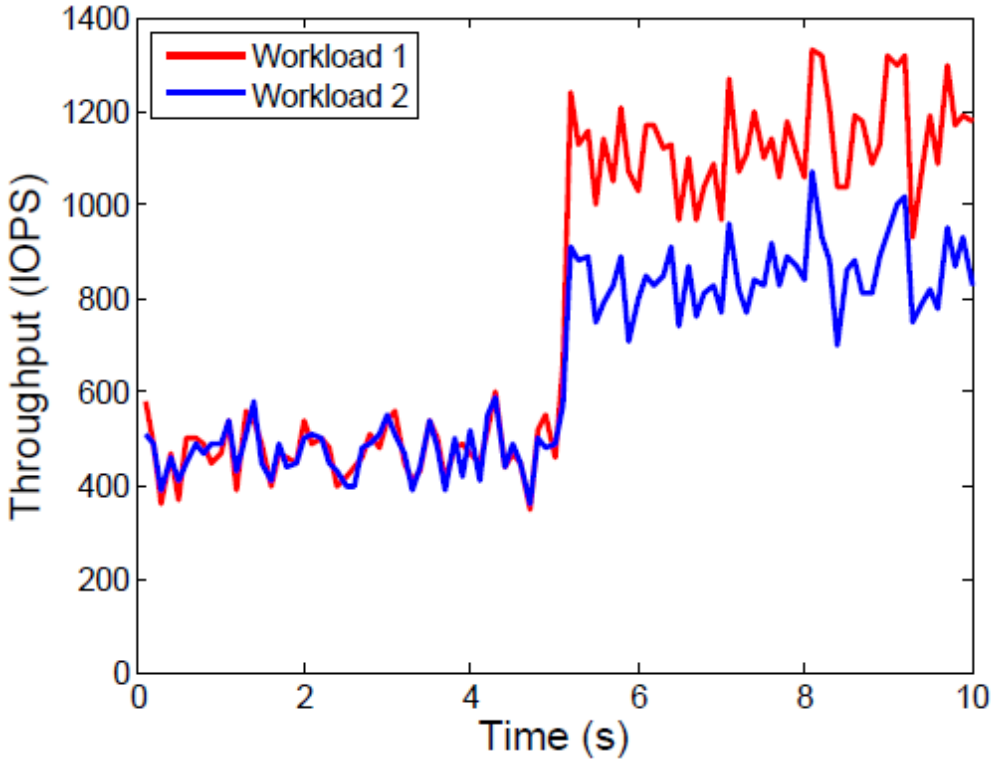


Variable Read-Write Ratio



Queue = 1

Variable Read-Write Ratio



Queue = 8

Differentiation became less due to queuing delay

RICE

Agenda

- Problem Motivation
- Reward Scheduling
- Preliminary Evaluation
- Open Issues & Conclusion

Open Issues

- Better measurement of response time
  - Isolating queuing delay
- Can we make queue-slot based technique work?
- Interaction between cache management and scheduling

Conclusion

- Existing approaches are insufficient
- Reward based allocation policy is an alternative
- Other approaches can potentially work
- We hope to encourage future work in this area

Questions