# Utilitarian Performance Isolation in Shared SSDs

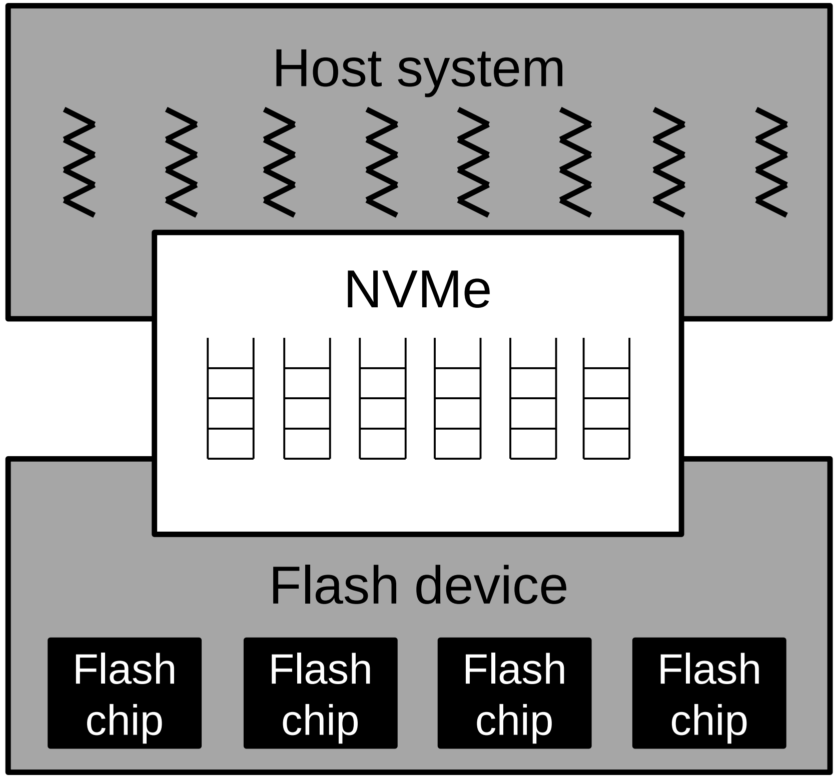
Bryan S. Kim
(Seoul National University, Korea)

# Flash storage landscape

Exploit parallelism!

Expose parallelism!

Increase parallelism!




- MQ blk IO (SYSTOR13)
- NVMeDirect (HotStorage16)
- SPDK (CloudCom17)

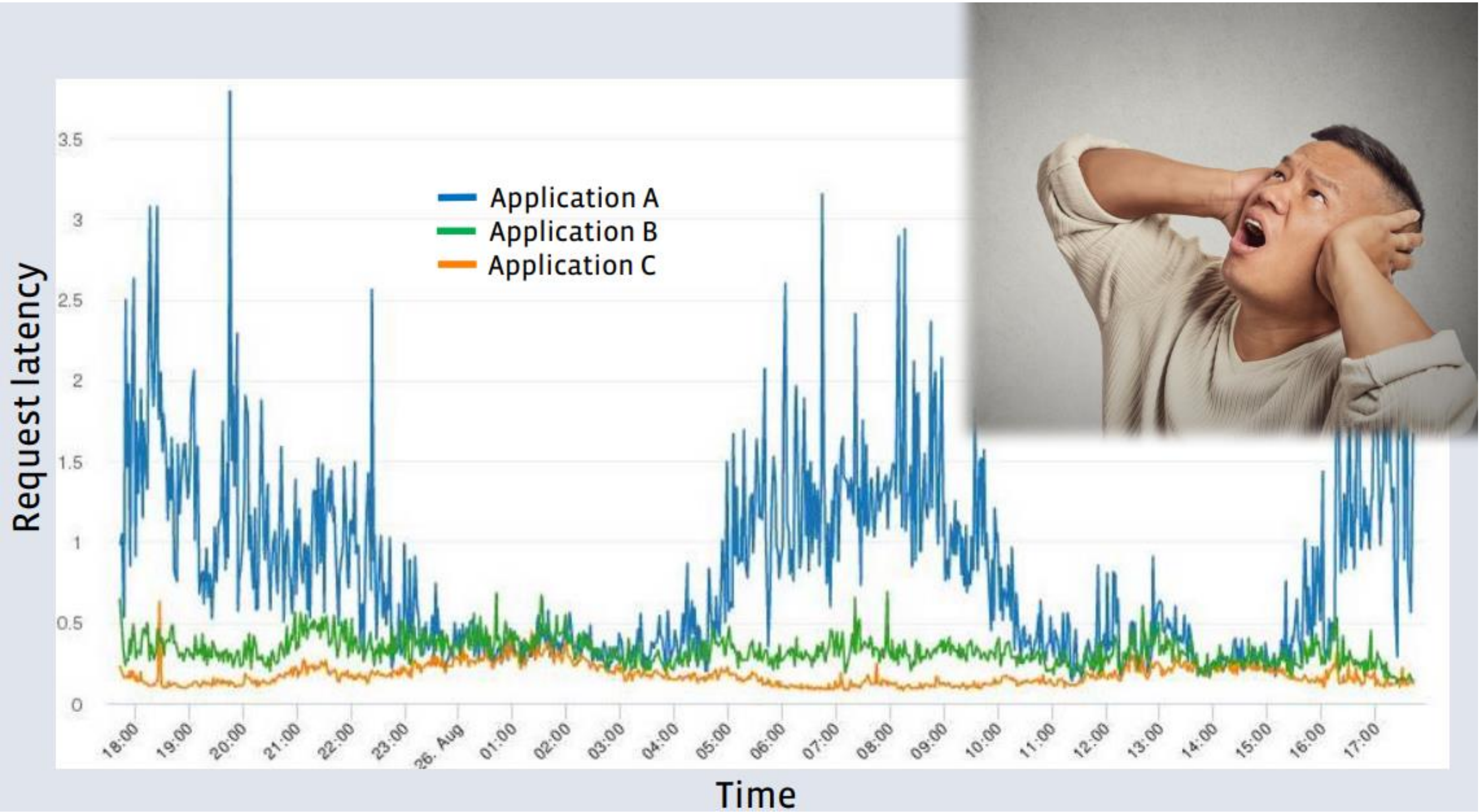
… and more

nvm EXPRESS (2013)

- Ozone (TC11)
- SSDsim (ICS11)
- Sprinkler (HPCA14)

… and more

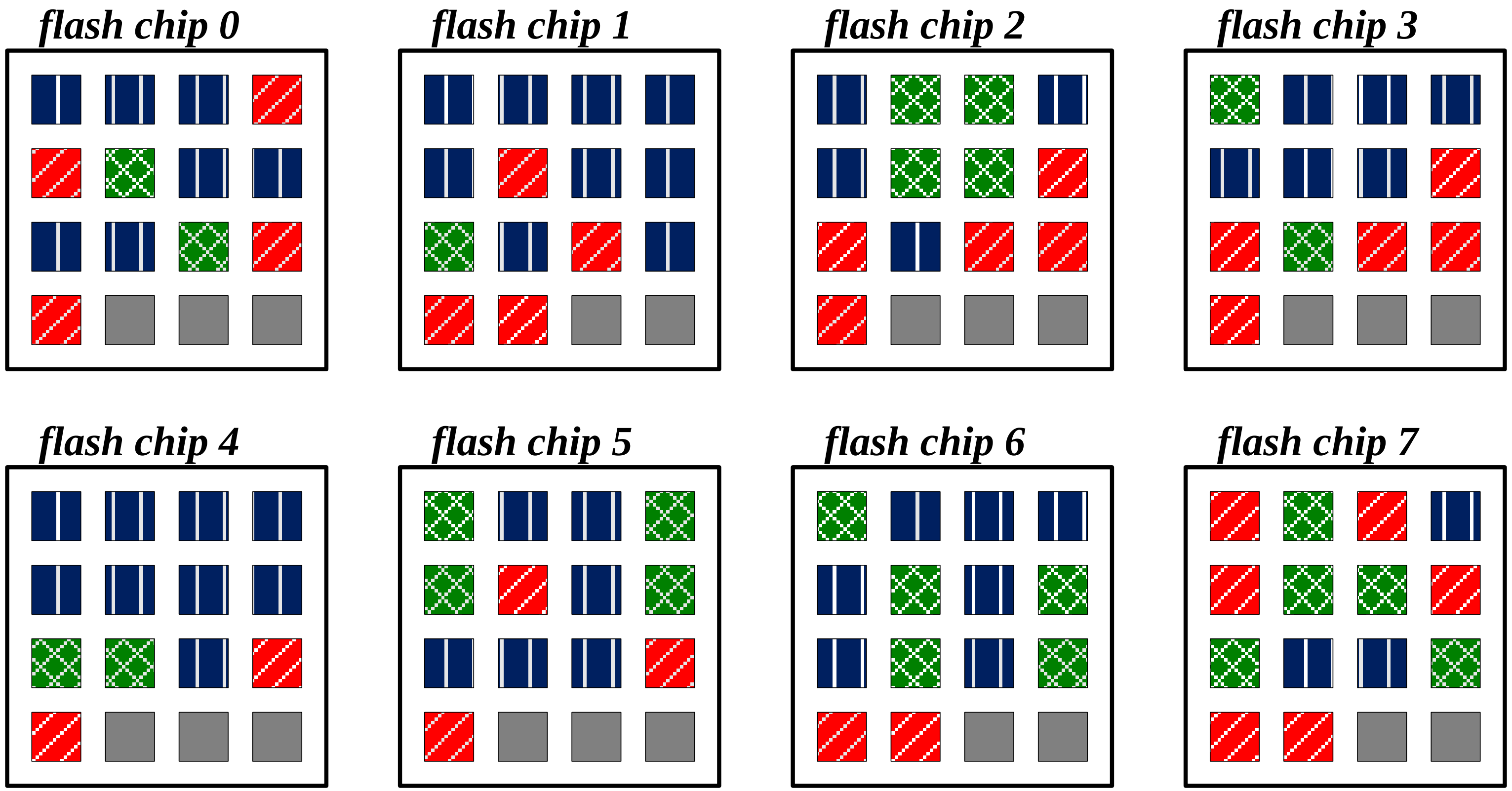
# Noisy neighbors




C. Petersen and A. Huffman, “Solving Latency Challenges with NVM Express SSDs at Scale”, Flash memory summit 2017,

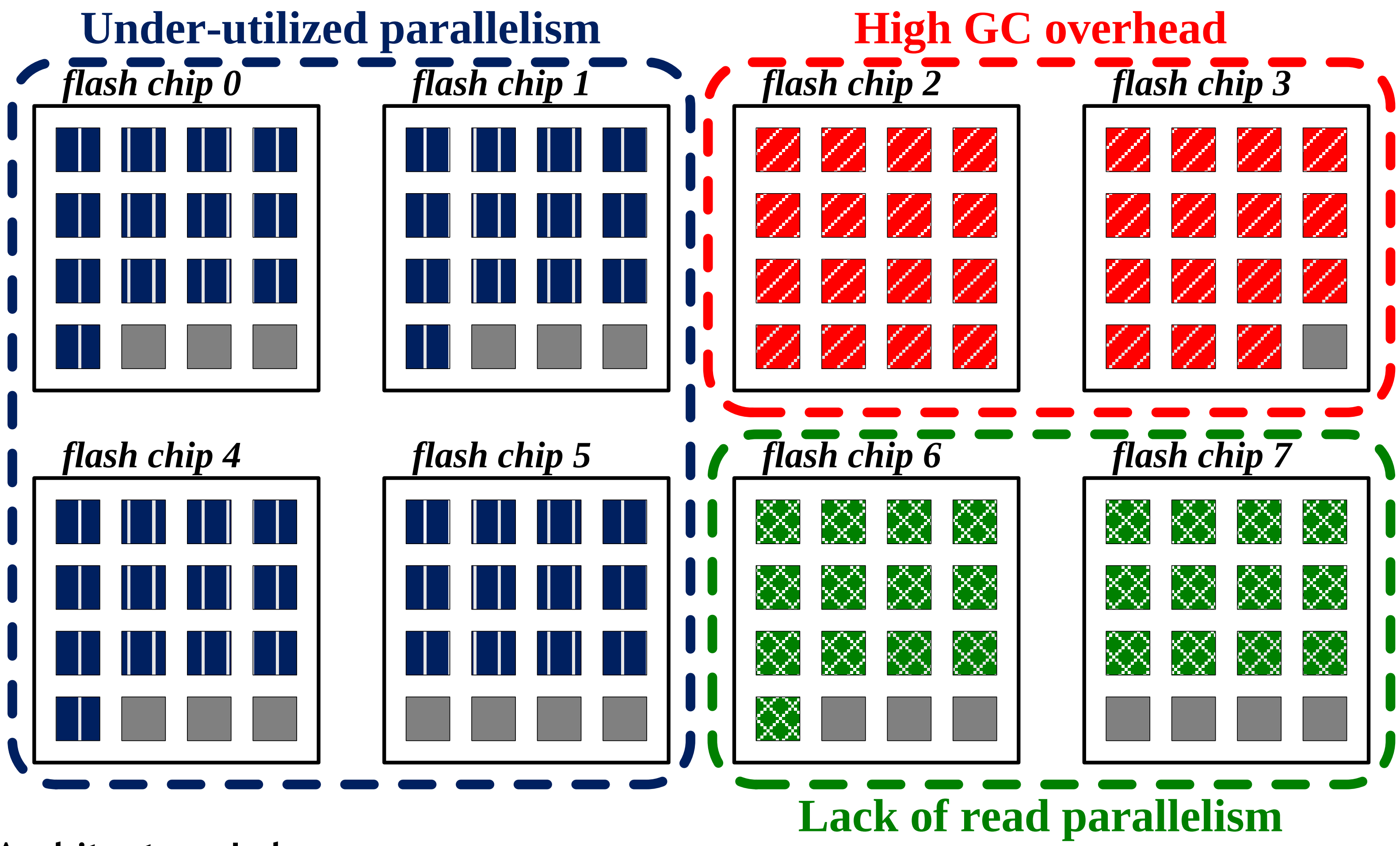
# Unified sharing of resources (free-for-all)

```
Blue tenant  : large capacity; infrequent access
Red tenant   : Write-intensive
Green tenant : QoS-sensitive
```

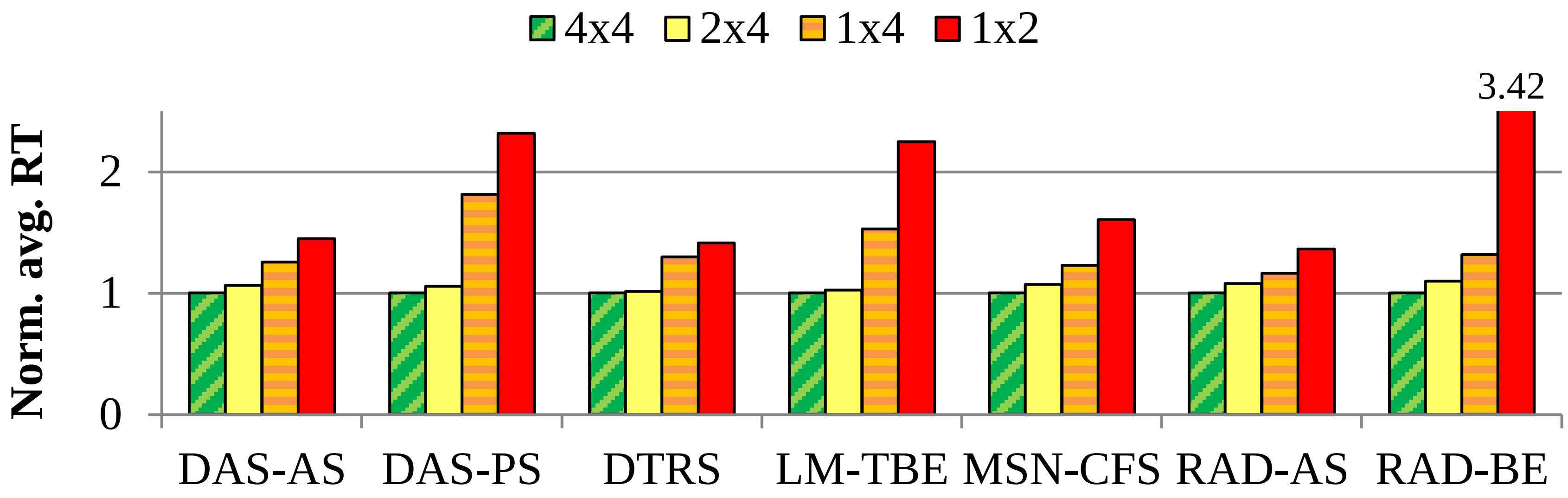


# Partitioning of resources (egalitarian)

Blue tenant : large capacity; infrequent access
Red tenant : Write-intensive
Green tenant : QoS-sensitive



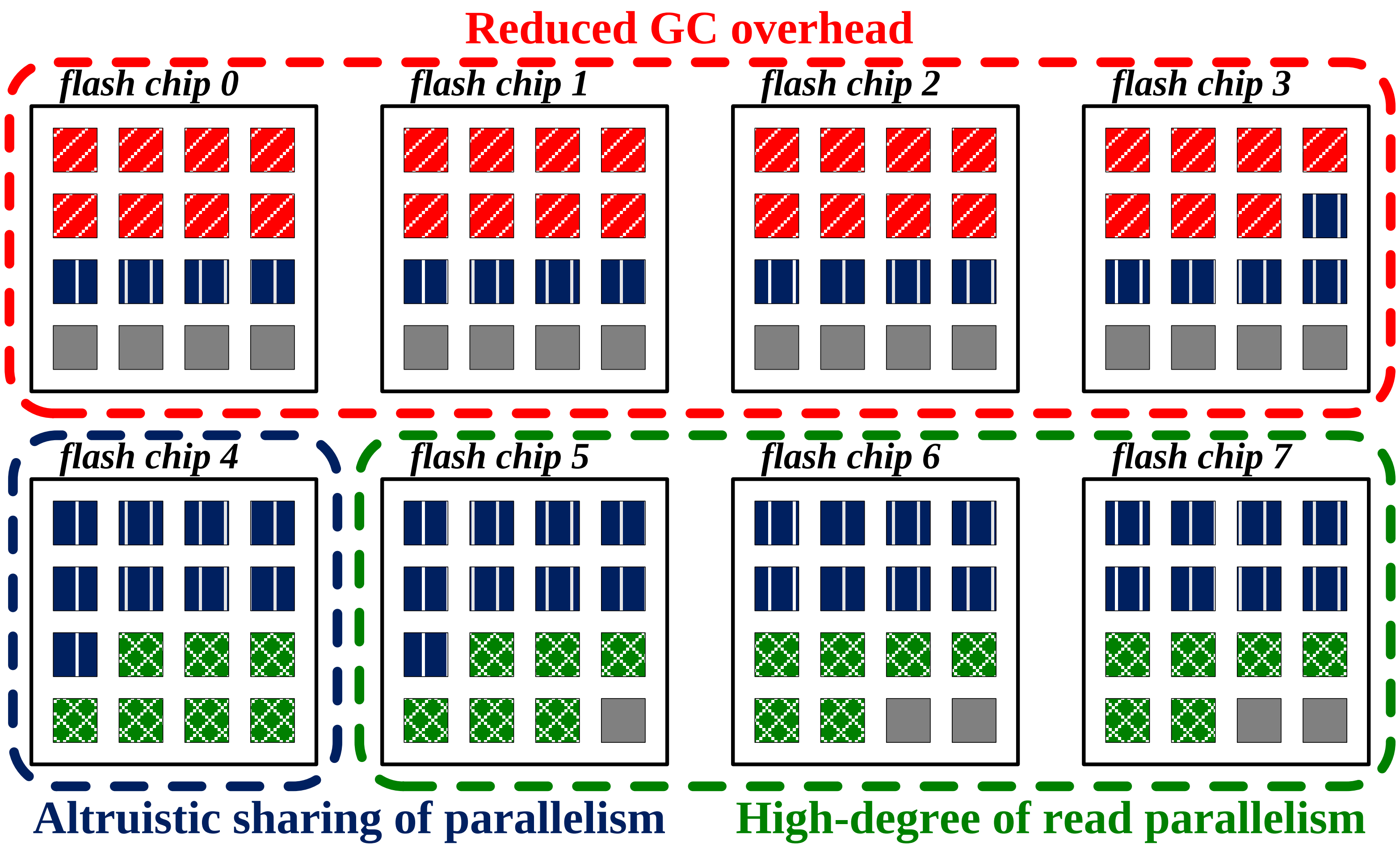
# Slashing parallelism for isolation




Performance suffers from reduced parallelism!

# Dynamic allocation of resources (utilitarian)

Blue tenant : large capacity; infrequent access
Red tenant : Write-intensive
Green tenant : QoS-sensitive

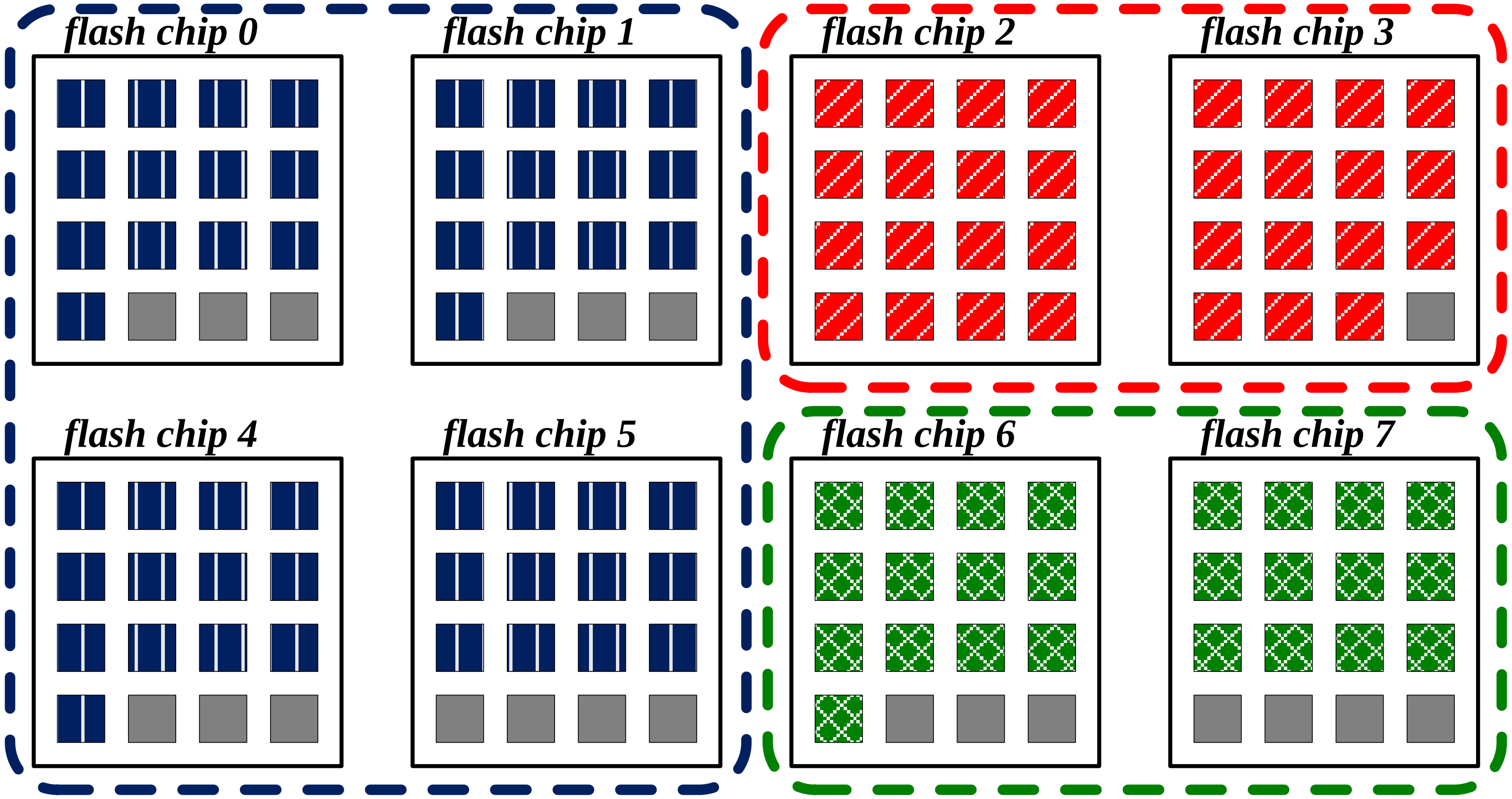


# Utilitarian performance isolation

- **Lessons from storage arrays**
  - Monitor each tenant's fair share of I/O
  - Determine optimal data placement
    - To balance the load across multiple storage devices…
    - … while considering data relocation overheads
- **Key insight**
  - Flash memory's challenges → Flash memory's opportunities
    - Need to maintain mapping → Easy to balance load
    - Need to garbage collect → Easy to relocate data
- **The utilitarian approach**
  - Compute tenant's utility (measure of received service)
  - Determine the allocation set (a set of chips for writing data) for each tenant
    - Allocation sets are mutually exclusive and collectively exhaustive
  - Allow data relocation among sets if needed

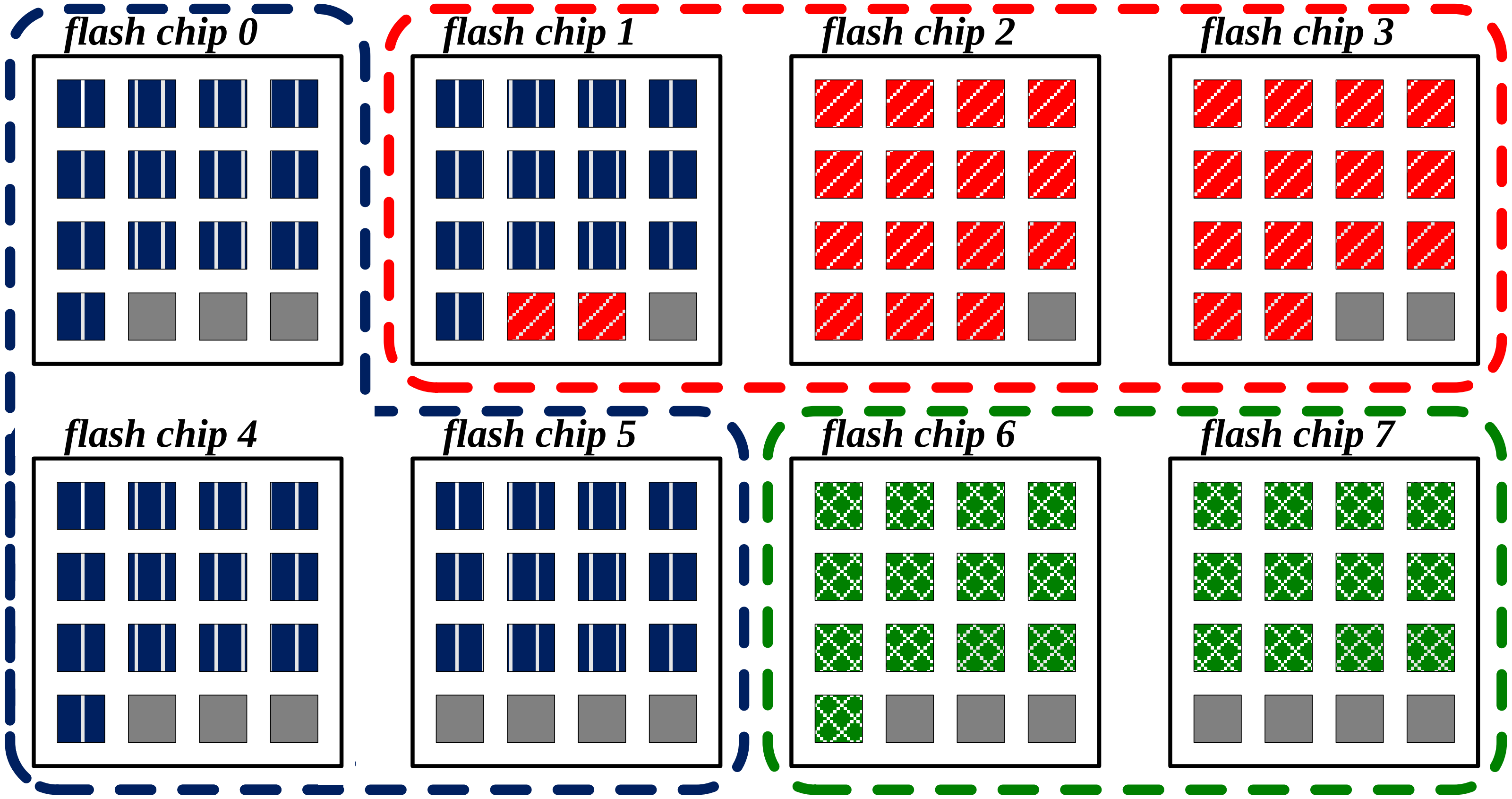
# Utility of tenants

Blue tenant's utility : 0.9
Red tenant's utility : 0.7
Green tenant's utility : 0.8



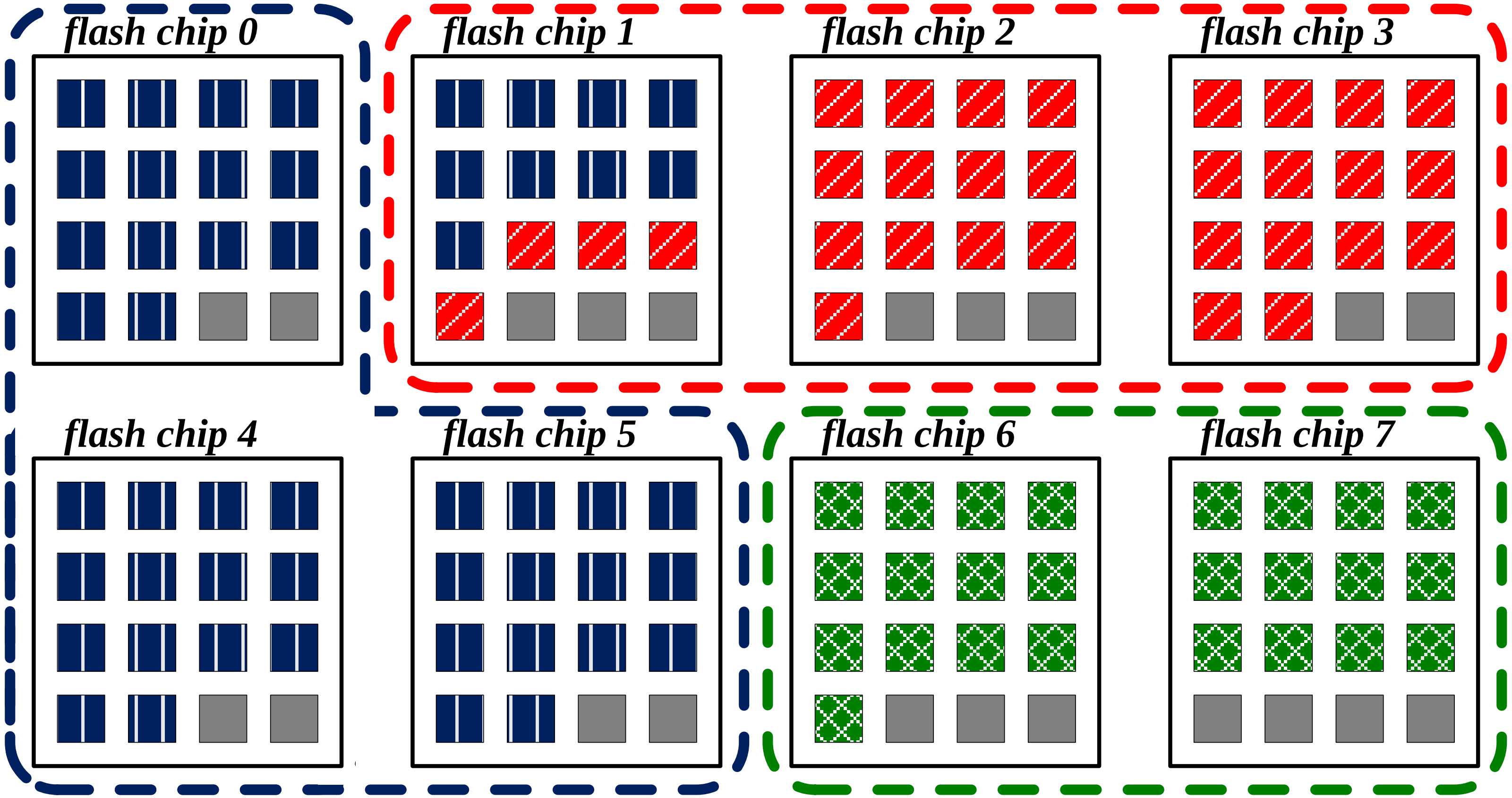
# Load balancing

Red tenant's writes are striped across a larger set of flash memory chips.
Blue's performance loss is minor.



# Data relocation

Garbage collection in chip 1 isolates some of Blue tenant's data by relocating them to its set.
Red tenant's GC efficiency improves.



# Utility function

- **Utility**
  - Utility of a tenant is high when its reads experience less traffic

$$Util(t,S) = \frac{\sum_{c}^{chips} N_r(t,c)}{\sum_{c}^{chips} \left(\frac{N_r(t,c)}{1 - Traffic(c,S)}\right)}$$

- **Traffic**
  - Traffic of a chip indicates the overall busyness of the chip

$$Traffic(c,S) = \frac{\sum_{t}^{tenants} \left(N_r(t,c)\cdot\tau_r + N_p(t,c,S)\cdot\tau_p\right)}{Time_{window}}$$

# Set allocation & data relocation

- **Set allocation**
  - Objective
    - Find allocation set that minimizes max-min ratio of utility across all tenants
  - Approximation
    - Transfer one chip from max utility tenant to min utility tenant
    - Avoid thrashing by transferring only if it balances the utility
    - Select a chip that experienced least number of reads

- **Data relocation**
  - Considering the number of reads of a “foreign” block during garbage collection
    - “Foreign” blocks that high number of reads are incentivized to relocate to its own set
    - Infrequently accessed cold data may remain in another set

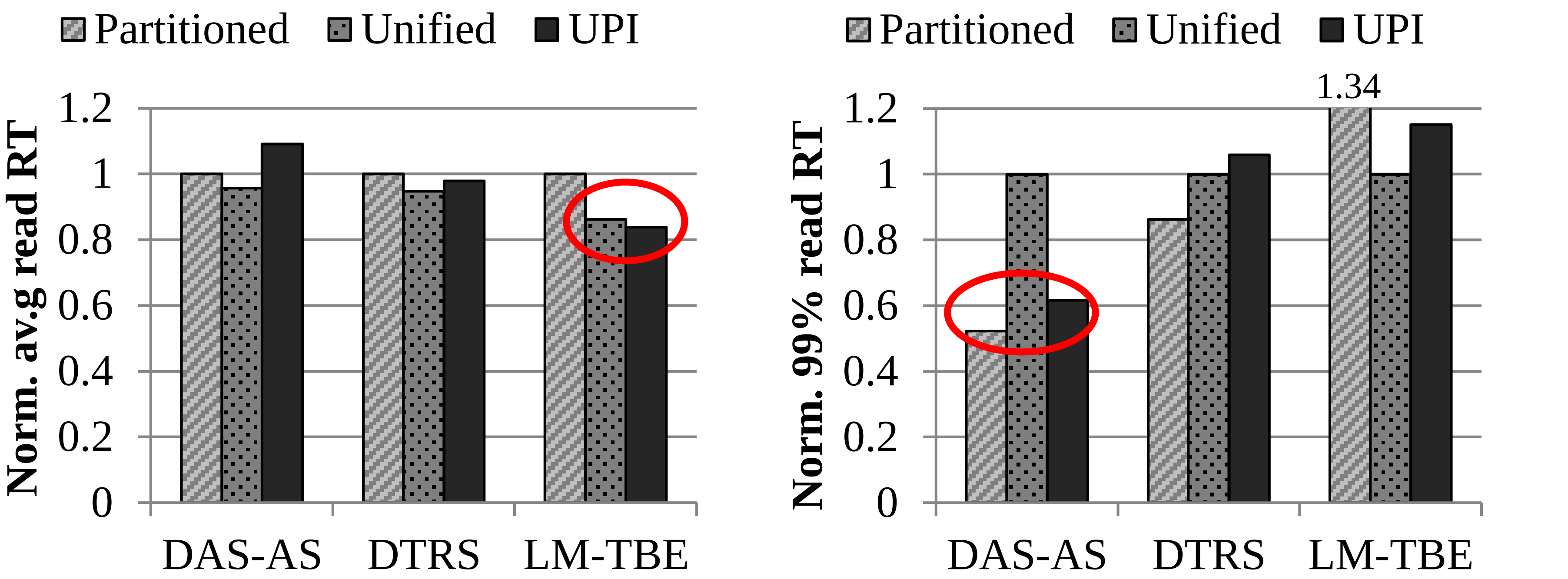
# Evaluation environment & methodology

- **Storage system configuration**
  - 150GB storage with 28% over-provisioning
    - 3 channels x 4 chips/chan
  - Garbage collection: reclaims space for writes + considers “foreign” block reads

- **Workload configuration**
  - 3 real-world I/O traces collected from MS production servers
    - DAS-AS: lowest throughput, highest read-to-write ratio
    - DTRS: relatively random workload with bursts of writes
    - LM-TBE: large sequential reads and writes

# Average performance

Partitioned : dedicates channel to each tenant
Unified : shares all resources among tenants
UPI : dynamically allocates based on utility



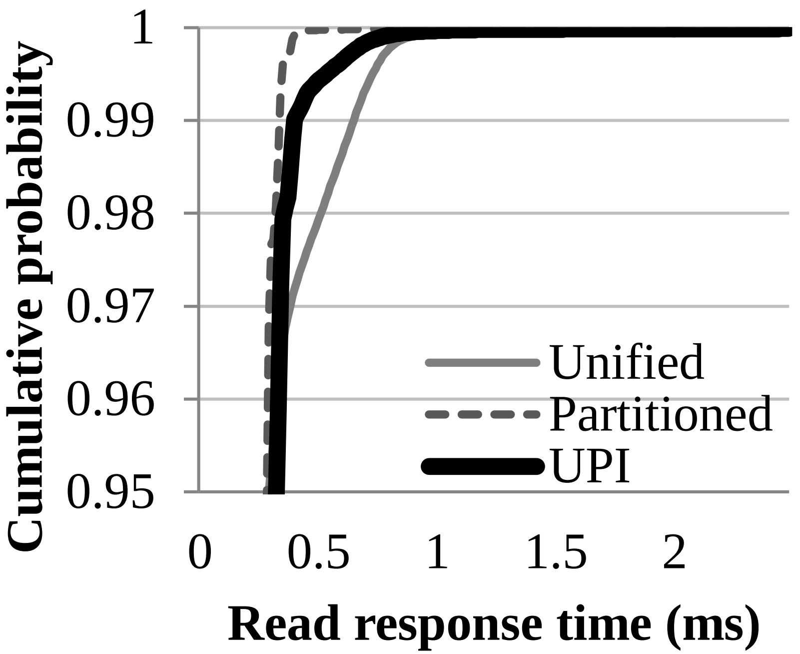

16.1% reduction for throughput-oriented application

38.5% reduction for latency-critical application

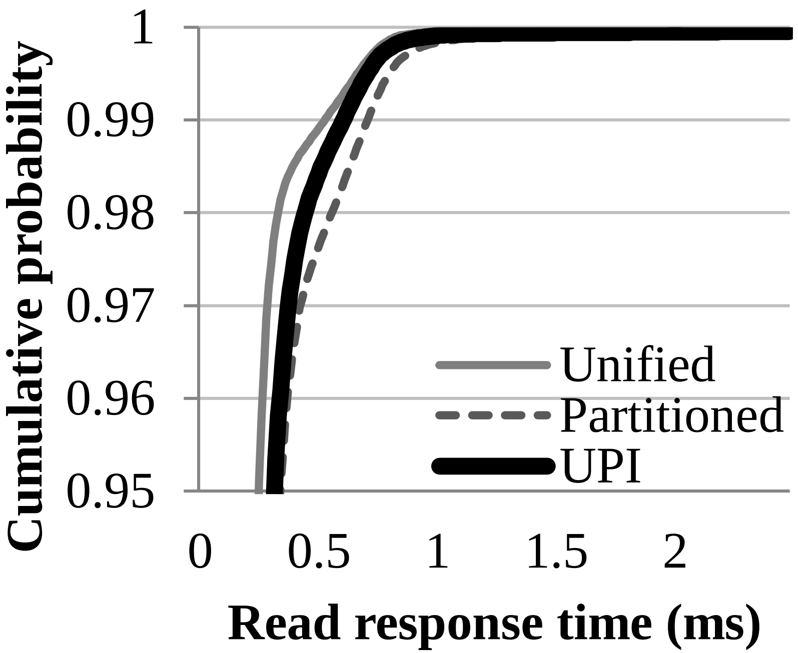
# QoS performance

Partitioned : dedicates channel to each tenant
Unified : shares all resources among tenants
UPI : dynamically allocates based on utility

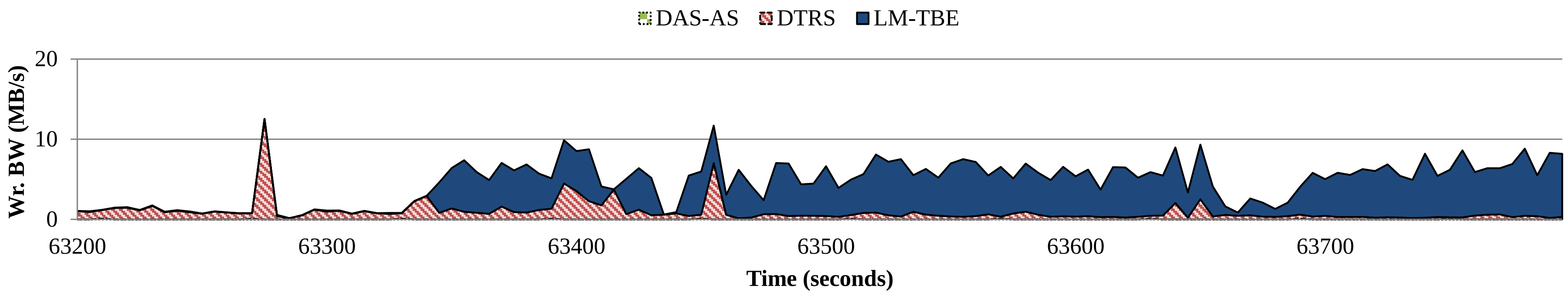
DAS-AS

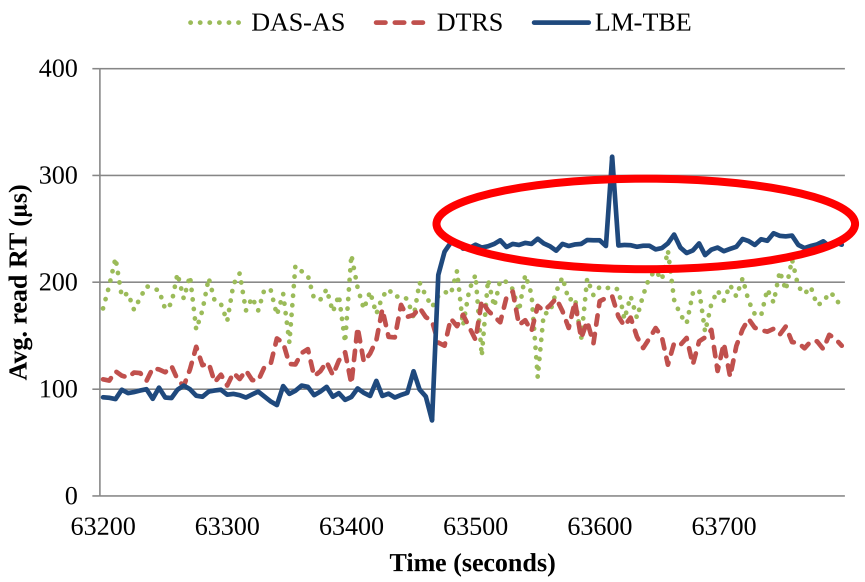



LM-TBE

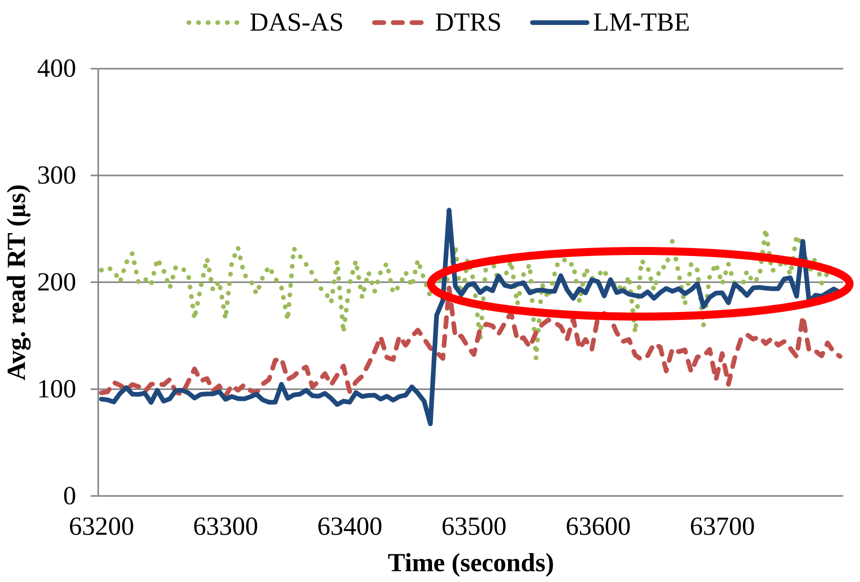


# Microscopic view




**Partitioned**




**UPI**

# Conclusion

- **Dynamic allocation of resources based on utility**
  - Decouple parallelism, isolation, and capacity
  - Balancing the load by distributing write traffic
  - Relocate data through existing SSD management mechanisms

**Utilitarian Performance Isolation**

reduces average response time
by 16.1% for high-throughput workload
reduces 99% QoS
by 38.5% for latency-critical workload