Improving I/O Resource Sharing of Linux Cgroup for NVMe SSDs on Multi-core Systems

USENIX HotStorage 2016

Sungyong Ahn*, Kwanghyun La*, Jihong Kim**
*Memory Business, Samsung Electronics Co., Ltd.
**Seoul National University

Outline

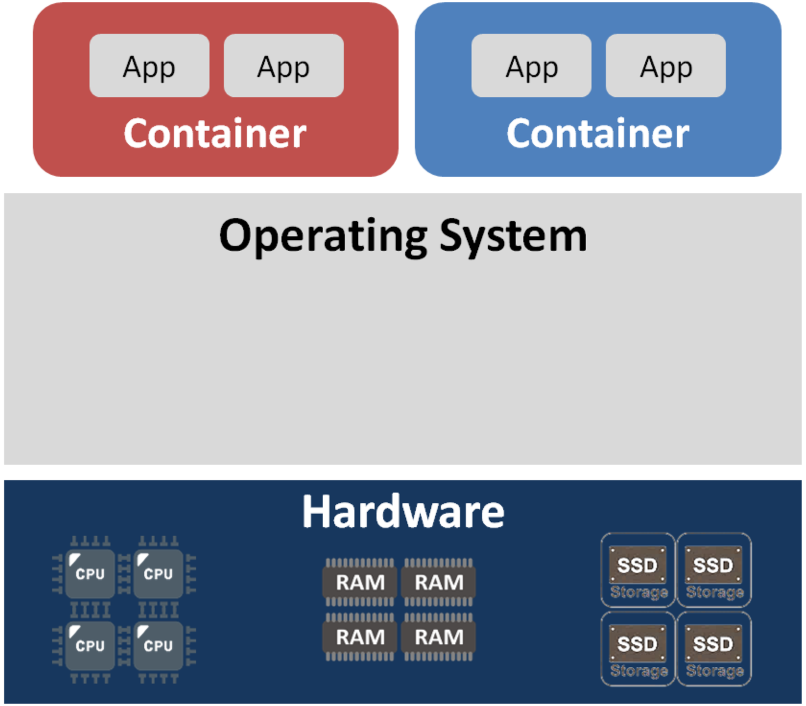
- Introduction
- Motivation
- Contributions
- Weight-based dynamic throttling (WDT) scheme
- Experimental Results
- Conclusion

Outline

- **Introduction**
- Motivation
- Contributions
- Weight-based dynamic throttling (WDT) scheme
- Experimental Results
- Conclusion

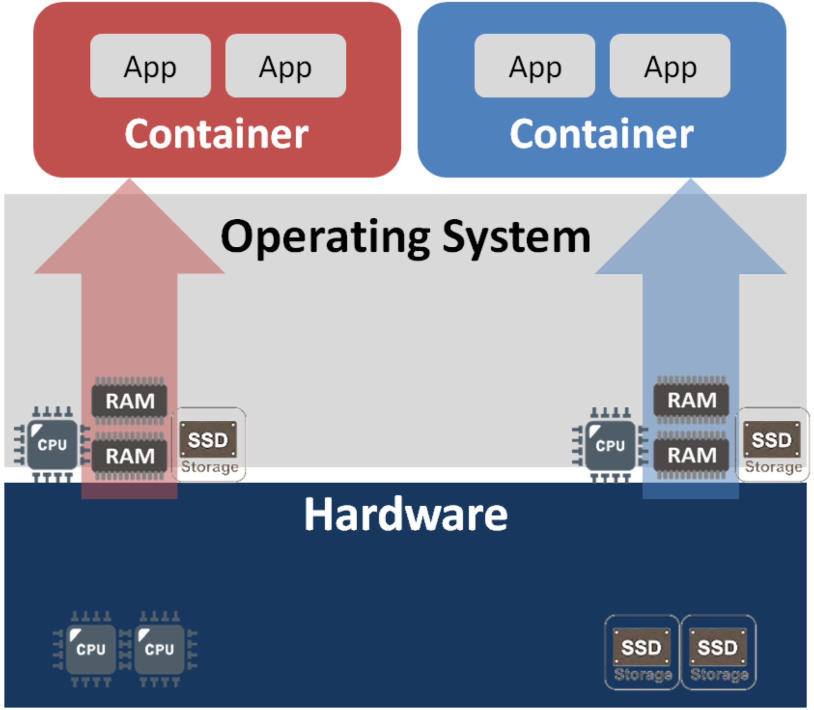
OS-level Virtualization

- **Multiple isolated instances (containers) running on a single host.**



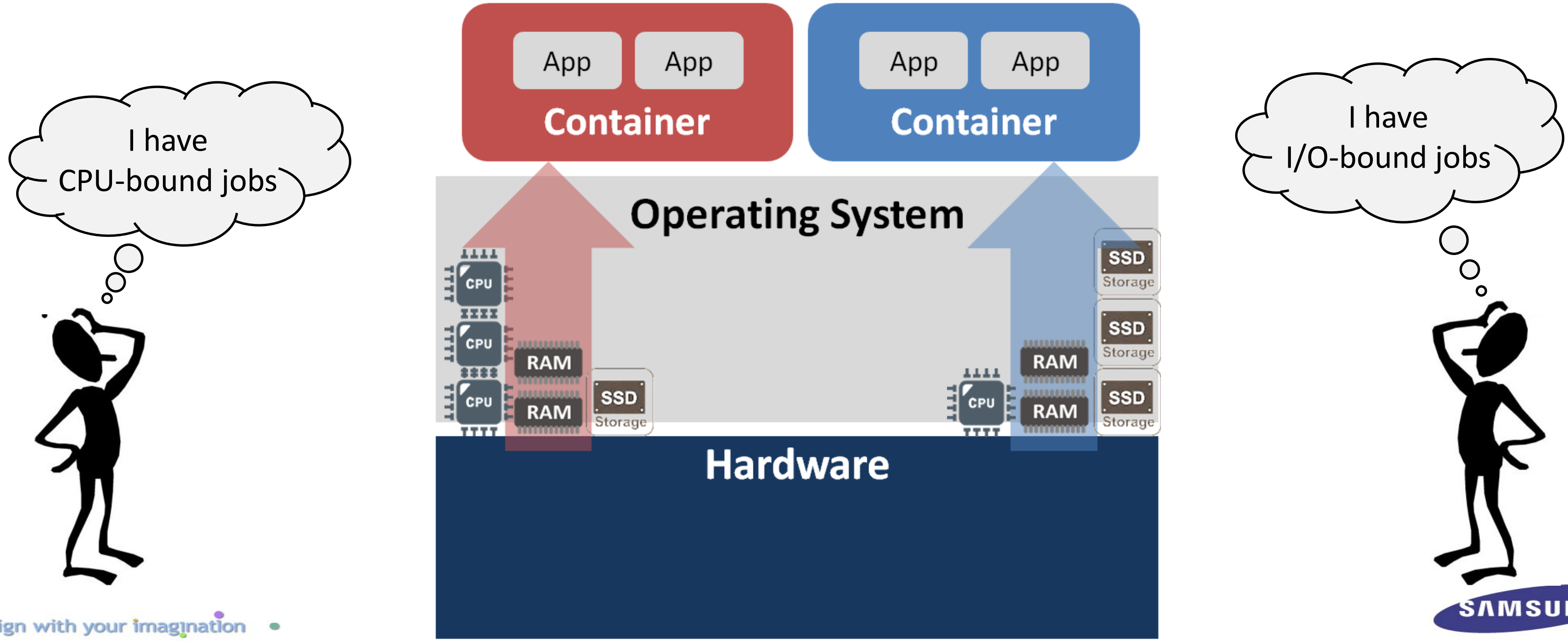
OS-level Virtualization

- **Multiple isolated instances (containers) running on a single host.**
  - Hardware resources should be isolated and allocated to containers



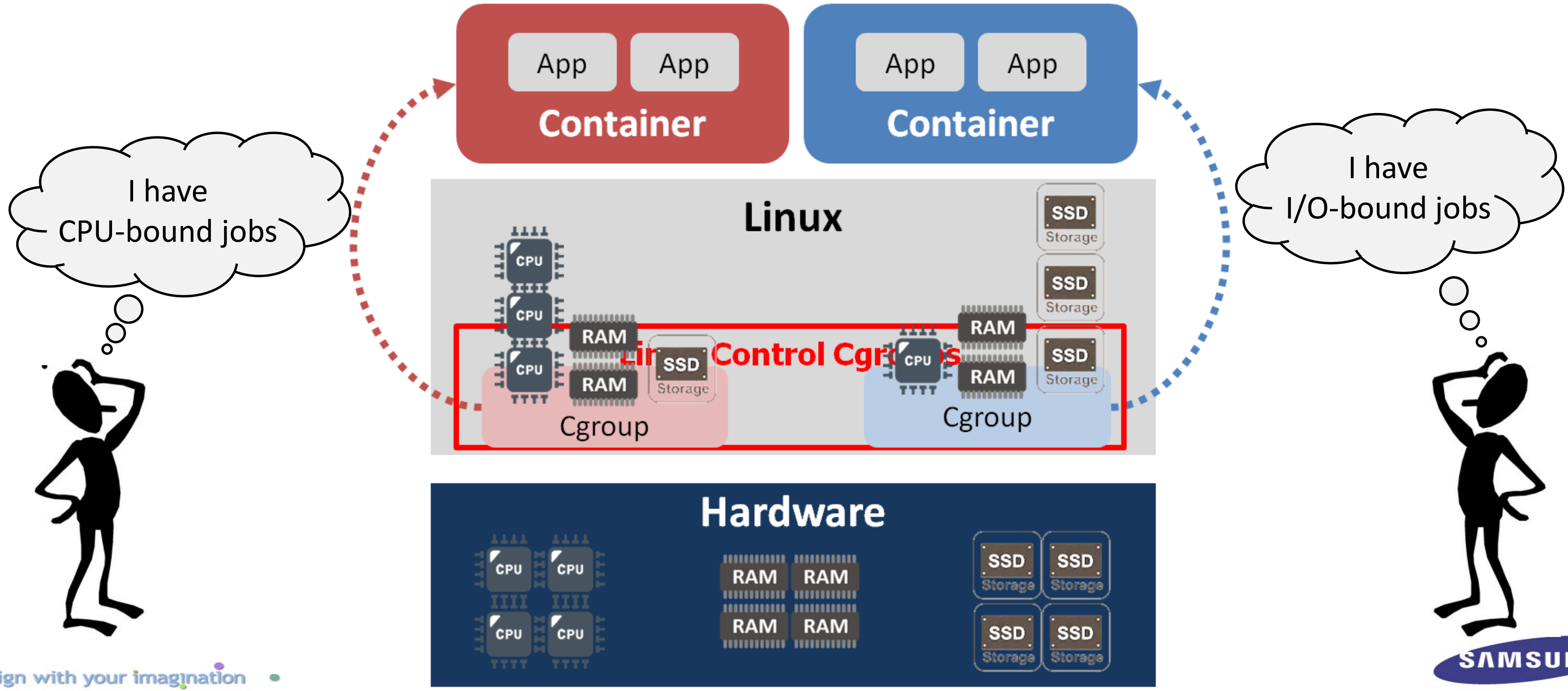
OS-level Virtualization

- **Multiple isolated instances (containers) running on a single host.**
  - **Hardware resources should be isolated and allocated to containers**
  - **Different resource requirements should be satisfied**



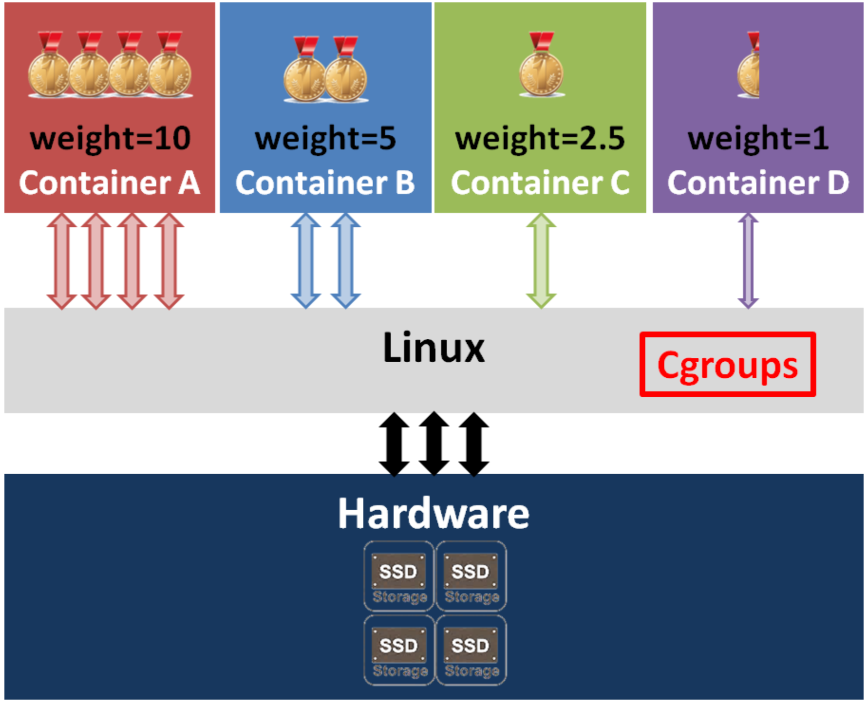
Linux Control Groups (Cgroups)

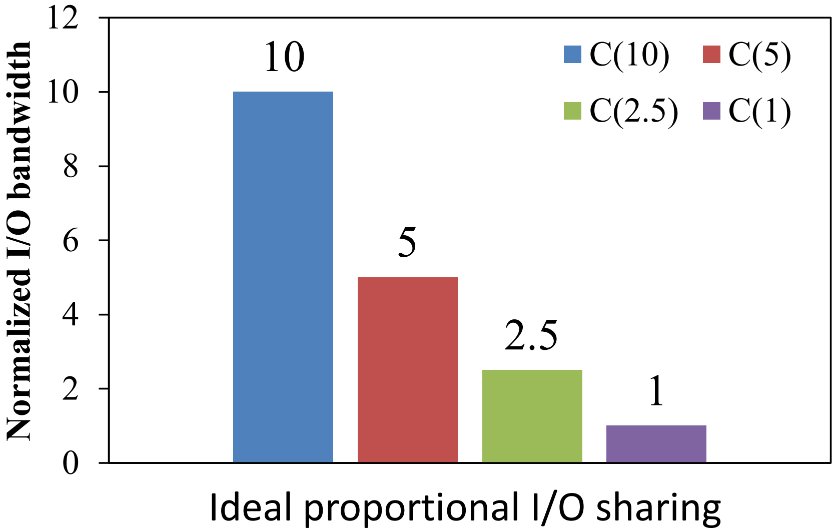
- **Kernel-level resource manager of Linux**



Proportional I/O scheme in Linux Cgroups

- **I/O bandwidth is shared according to I/O weights**





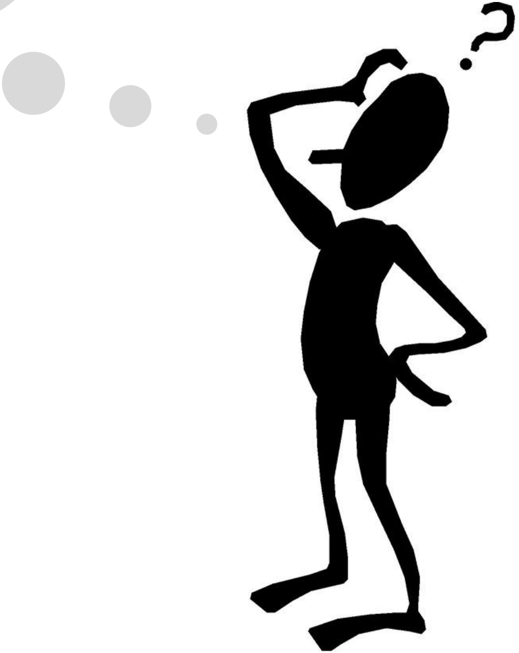
Outline

- Introduction
- **Motivation**
- Contributions
- Weight-based dynamic throttling (WDT) scheme
- Experimental Results
- Conclusion

Questions

When Linux Cgroups work with NVMe SSD,

1. **Can they share the I/O resource in proportion to I/O weights?**
2. **Is it scalable?**



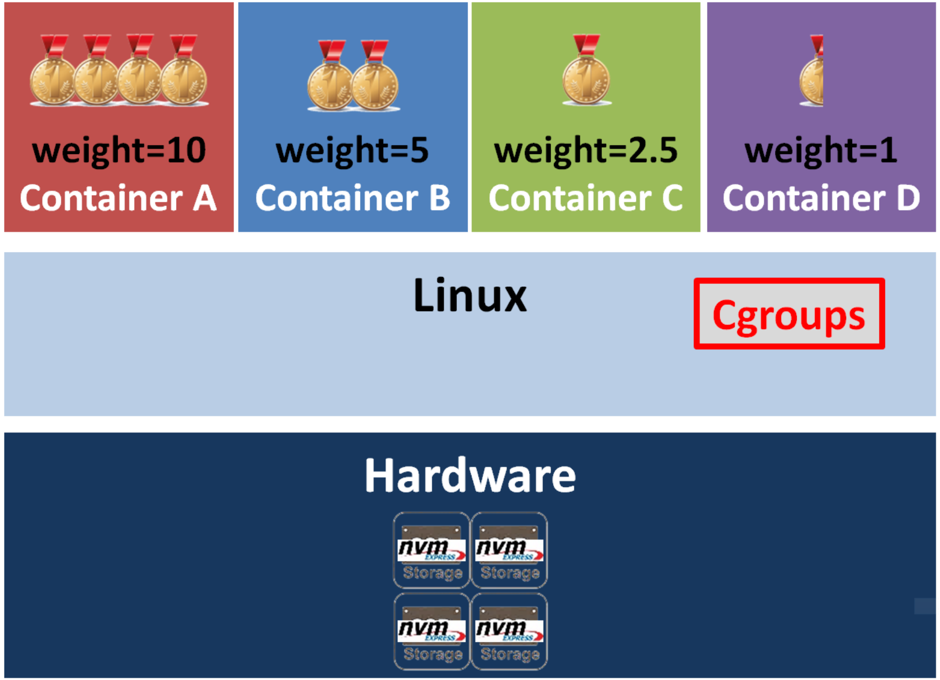
Questions

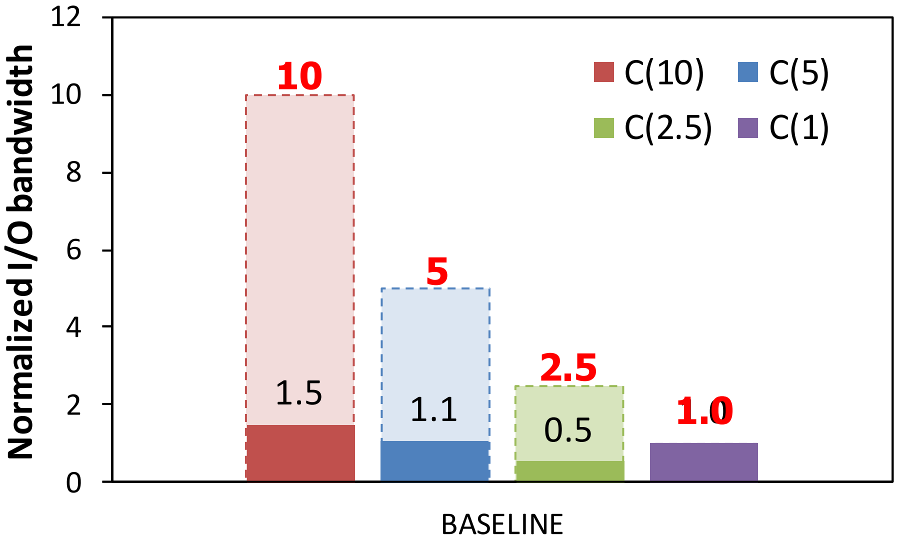
When Linux Cgroups work with NVMe SSD,

1. **Can they share the I/O resource in proportion to I/O weights?**
2. **Is it scalable?**

Proportional I/O with NVMe SSDs

- **Existing Cgroups cannot support the proportional I/O to NVMe SSDs**

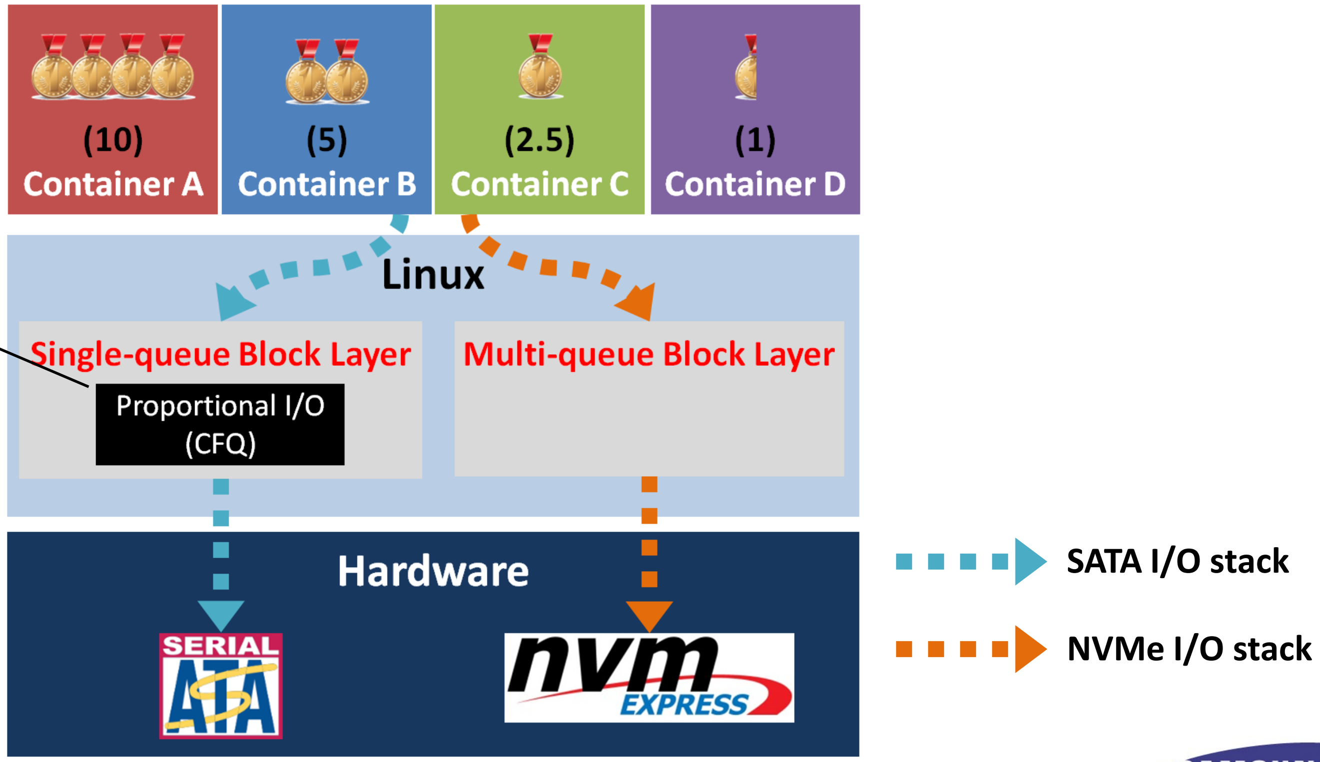




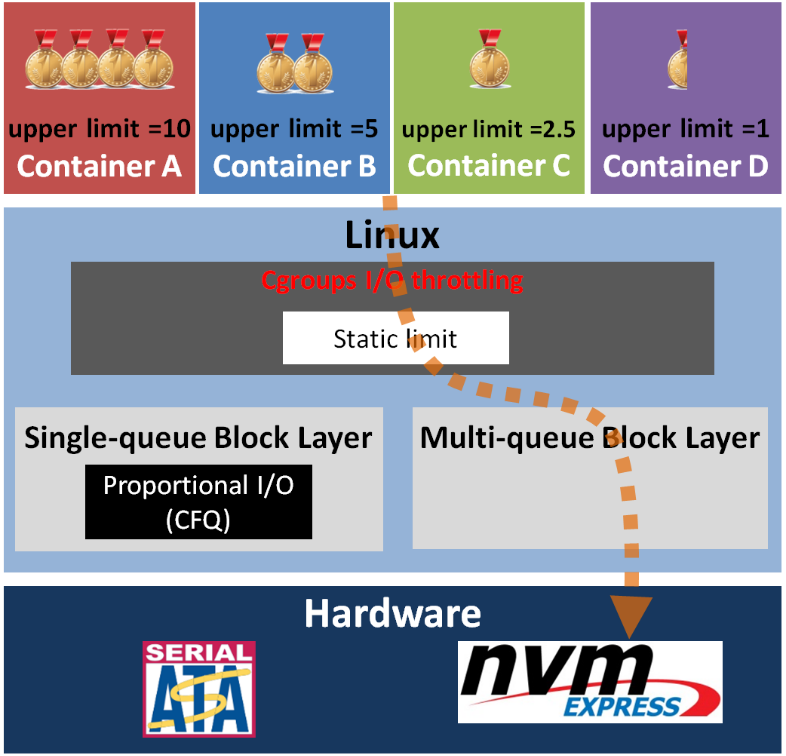
Because…

- **NVMe SSDs have different I/O stack from SATA storage**

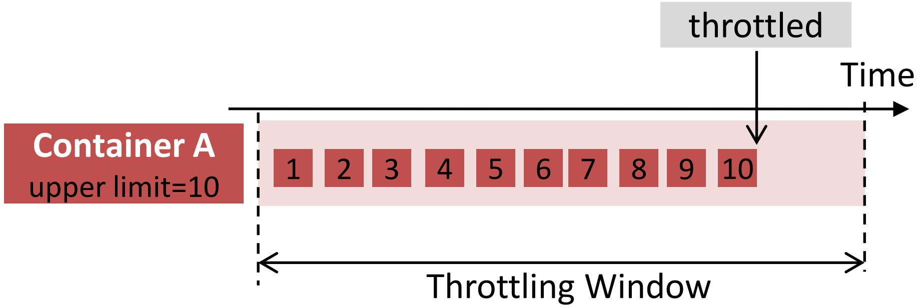
Existing proportional I/O scheme is implemented in single queue block layer



First Attempt: Using the Existing Static Throttling

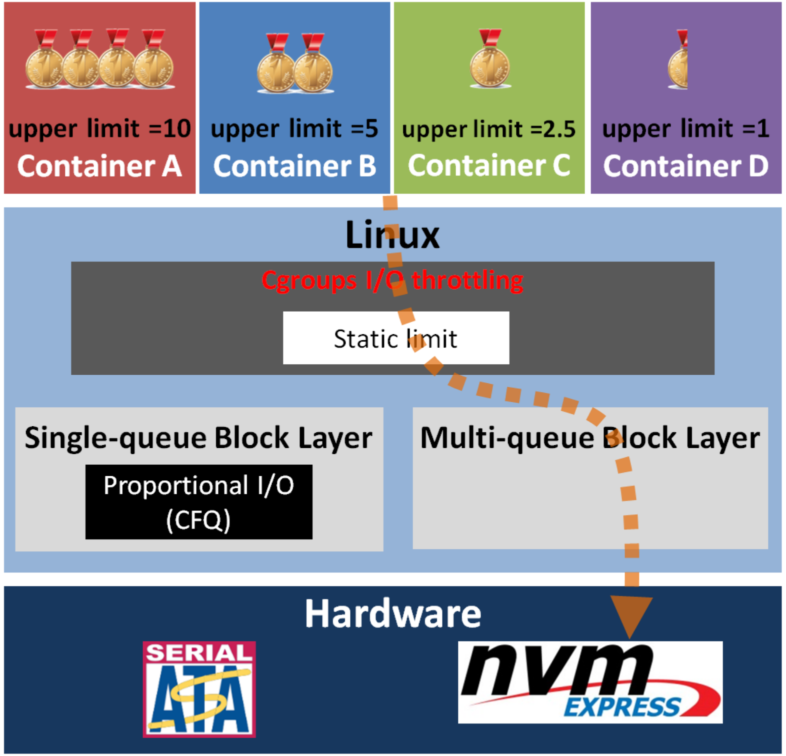


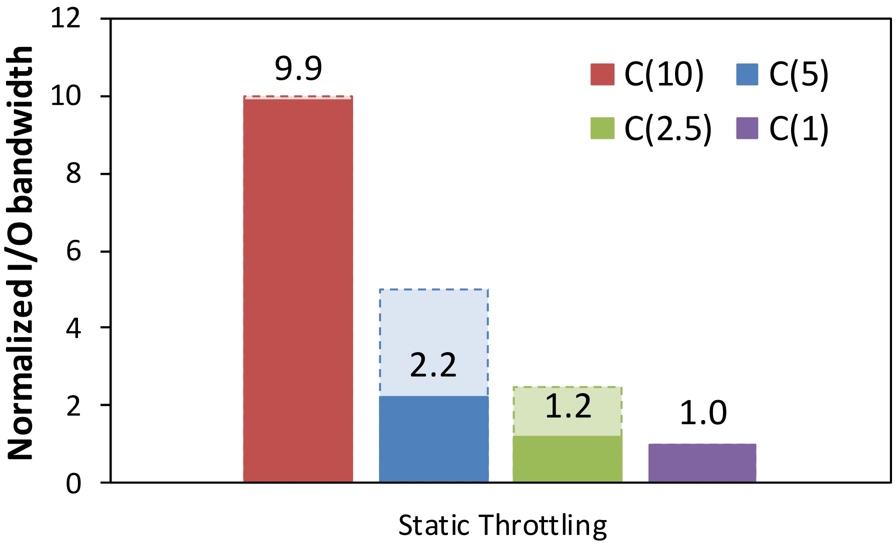
- **Upper limit of I/O bandwidth**
  - Limit the maximum number of bytes or I/O requests for particular time interval (throttling window)



First Attempt: Using the Existing Static Throttling

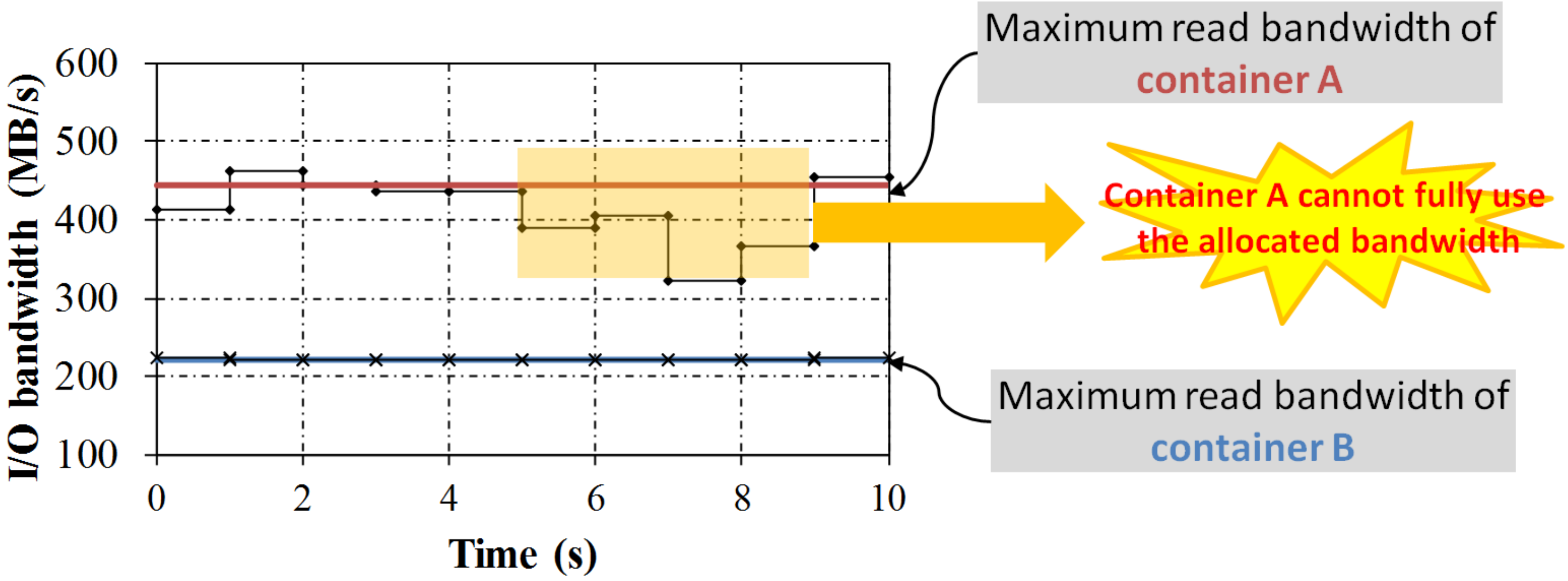
- **Static throttling is not enough to support the proportional I/O**





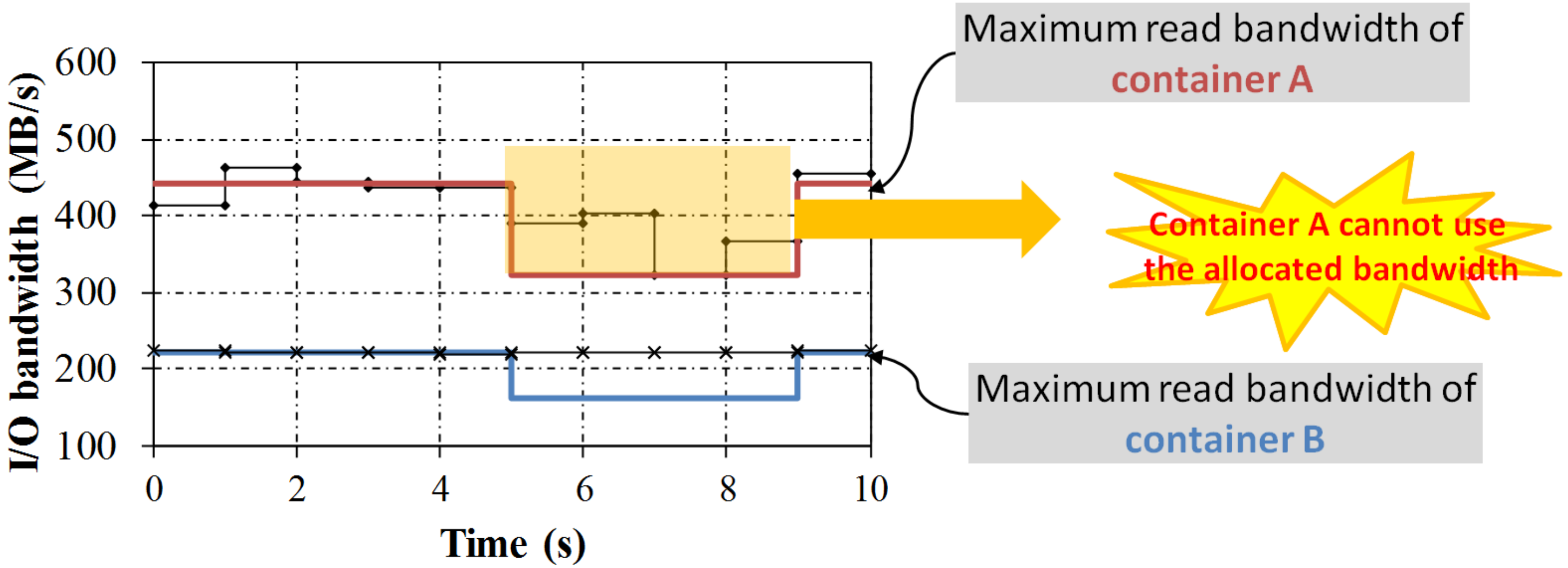
Because...

- **I/O workloads fluctuate with time**



Because...

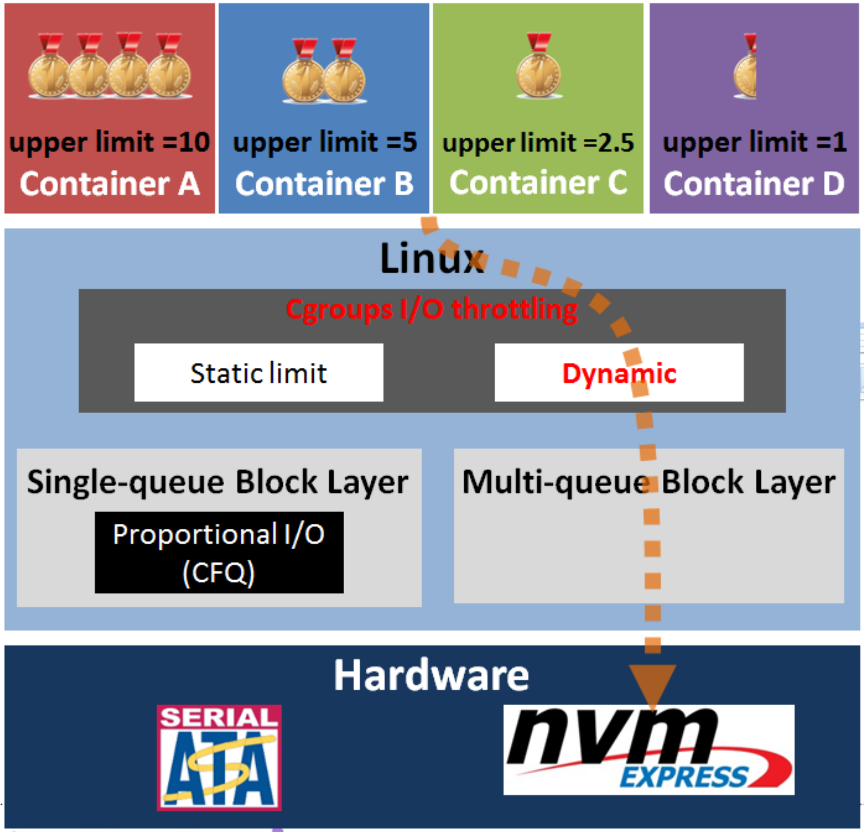
- **I/O workloads fluctuate with time**



Outline

- Introduction
- Motivation
- **Contributions**
- Weight-based Dynamic throttling (WDT) scheme
- Experimental Results
- Conclusion

Contributions

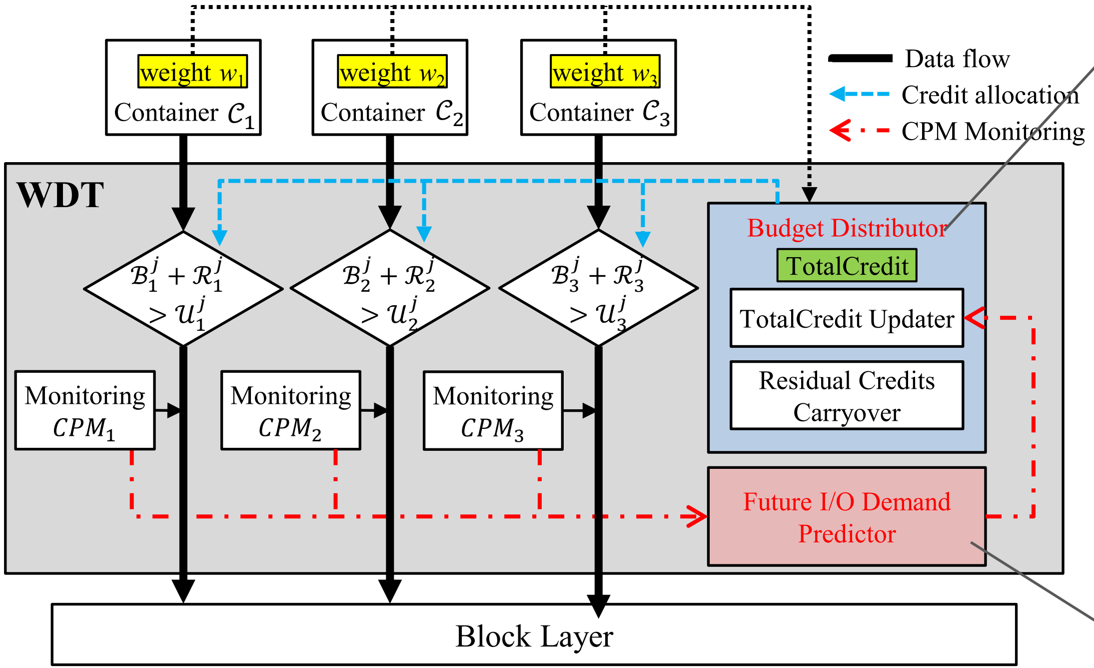


- ✓ We achieved the proportional I/O for NVMe SSDs.
- ✓ We achieved the scalable performance of Linux Cgroups.

Outline

- Introduction
- Motivation
- Contributions
- **Weight-based Dynamic throttling (WDT) scheme**
- Experimental Results
- Conclusion

Overview of WDT Scheme

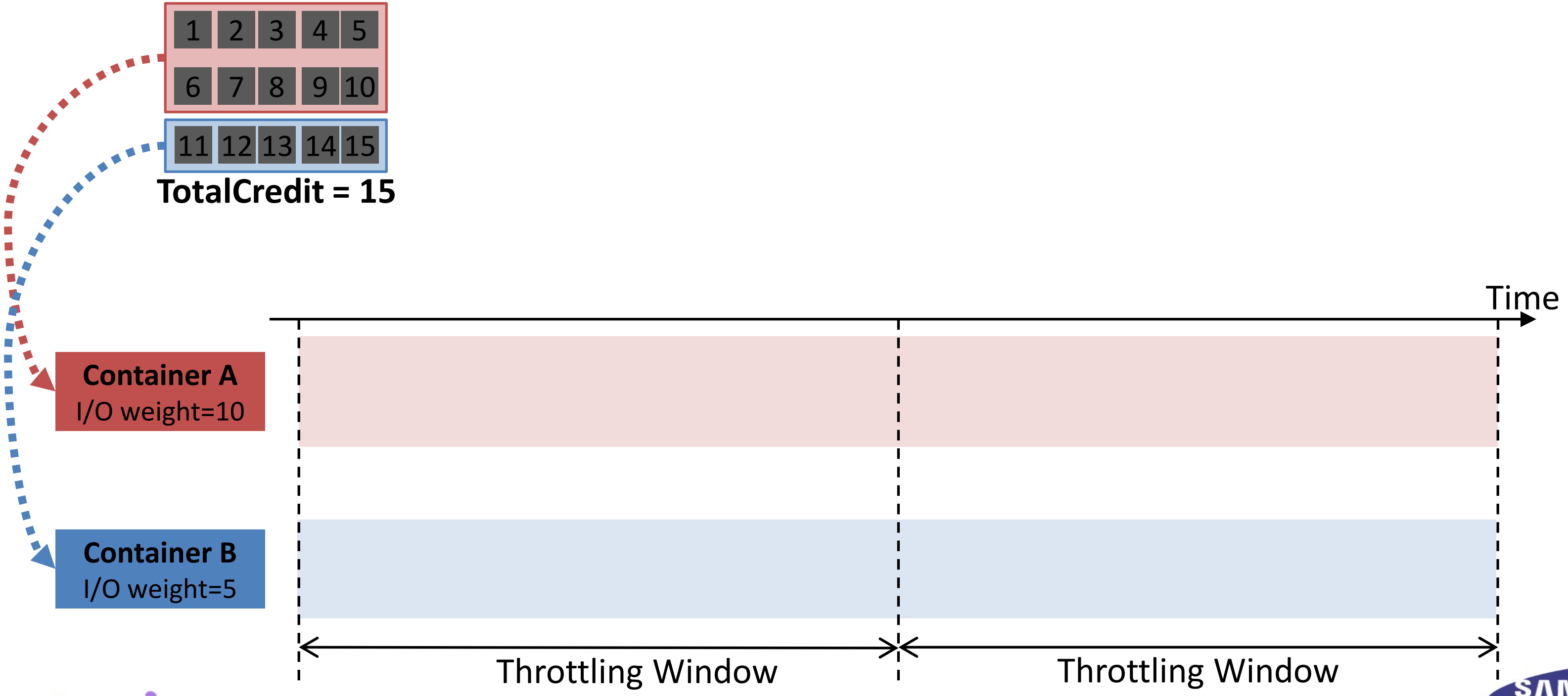


Distributing the credits to containers according to I/O weights

To update TotalCredit, future I/O demand is predicted

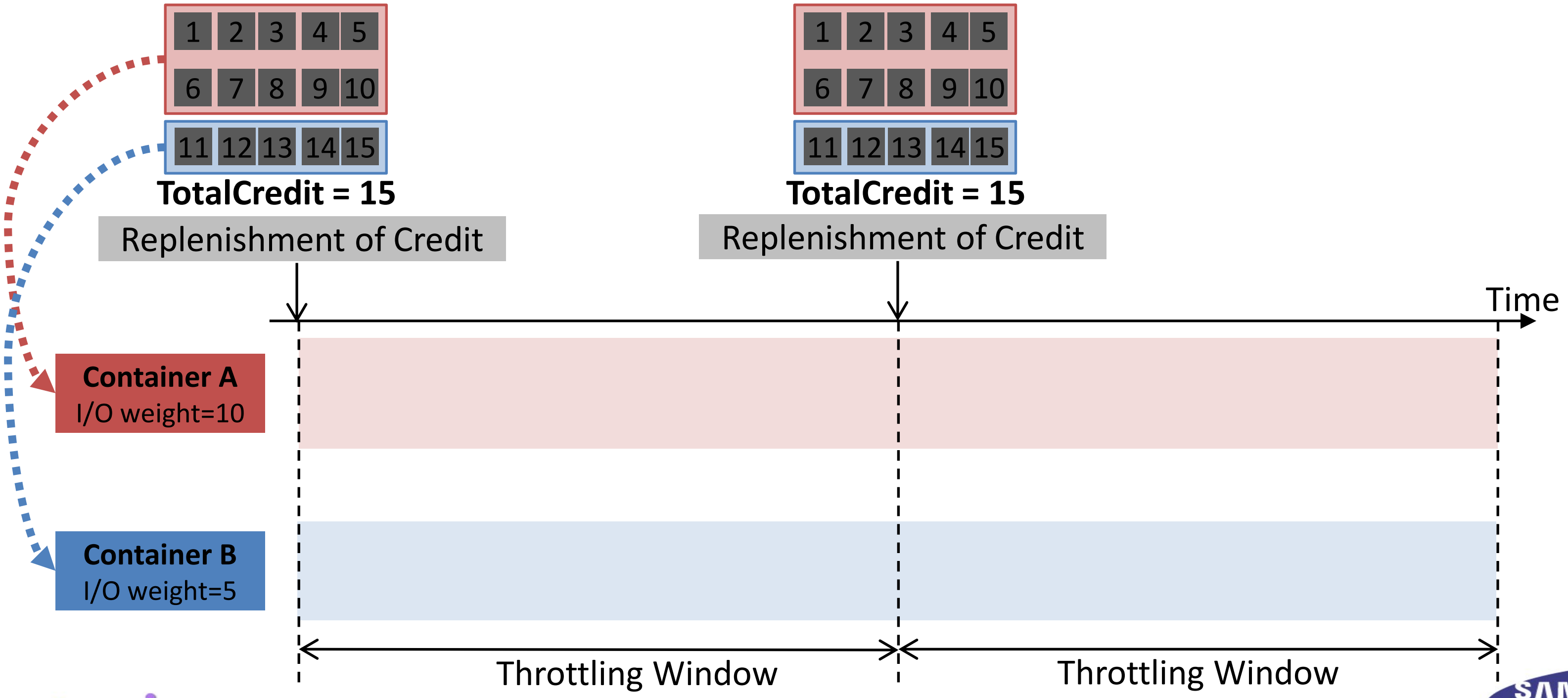
Budget Distributor

- **All containers are allocated credits in proportion to their I/O weight.**



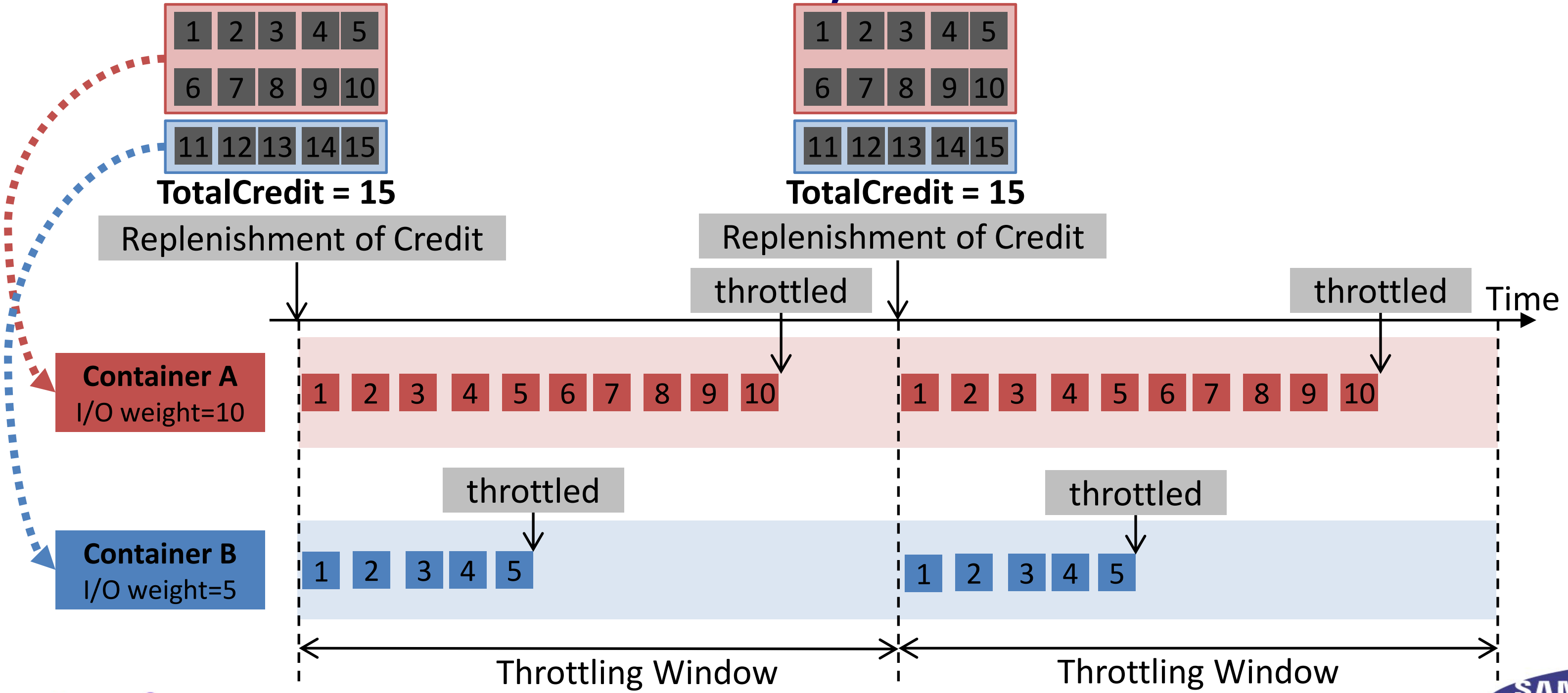
Budget Distributor

- All containers are allocated credits in proportion to their I/O weight.
- Credits are replenished periodically.



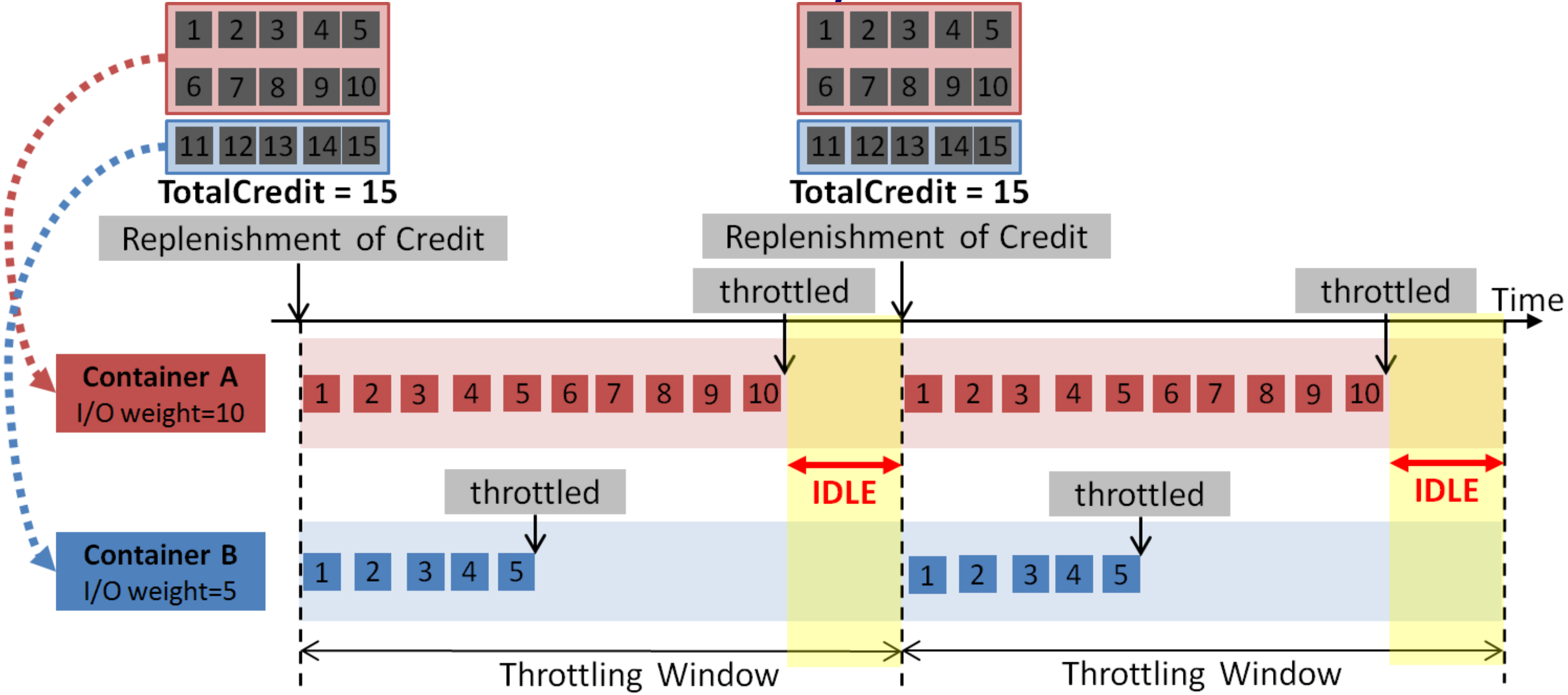
Budget Distributor

- All containers are allocated credits in proportion to their I/O weight.
- Credits are replenished periodically.
- If a container has no available credit, it is throttled.



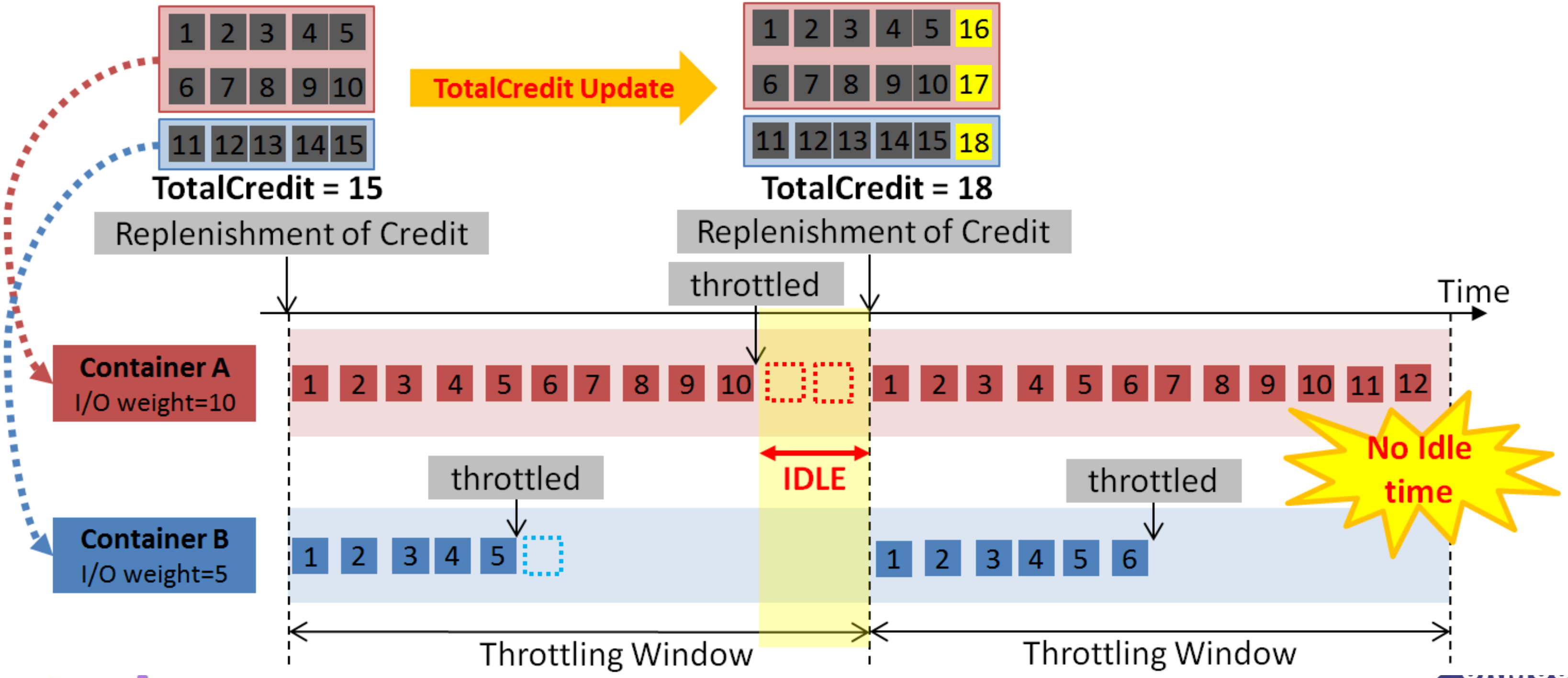
Budget Distributor

- All containers are allocated credits in proportion to their I/O weight.
- Credits are replenished periodically.
- If a container has no available credit, it is throttled.



TotalCredit Updater

- **In order to remove storage idle time, TotalCredit is adjusted.**



Technical Challenge

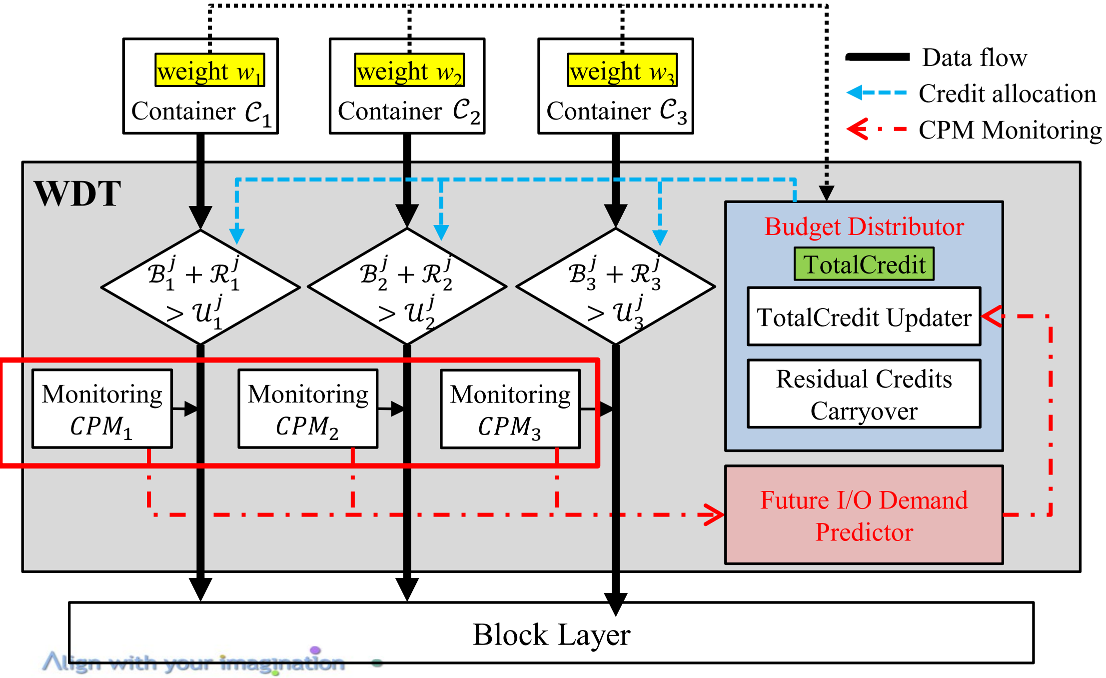
✓ **How to predict TotalCredit required for the next interval?**

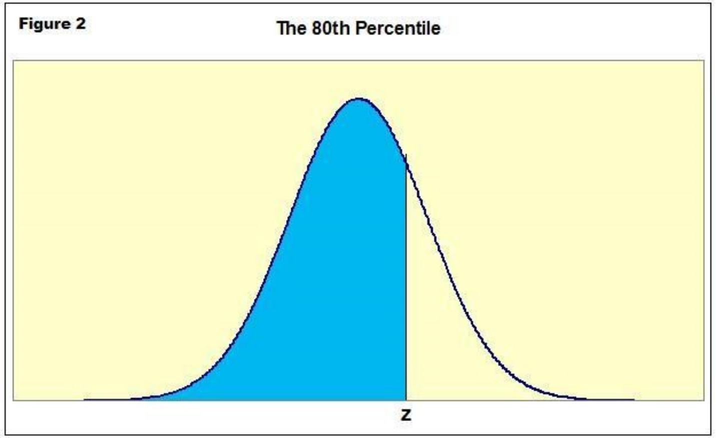
Technical Challenge

✓ How to predict **future I/O demand** for the next interval?

Future I/O Demand Predictor

- **Monitoring I/O demand of each container for every interval**
  - **Prediction of the future I/O demand from cumulative distribution function**
    - 80th percentile of a cumulative distribution of I/O demand (assuming normal distribution)





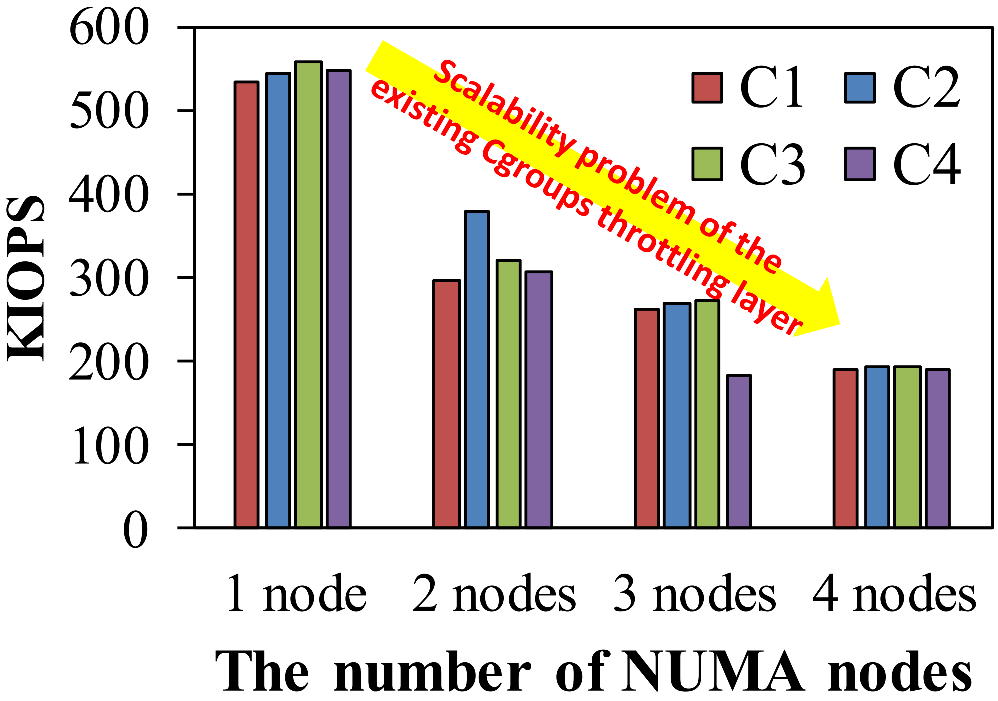
Questions

When Linux Cgroups work with NVMe SSD,

1. Can they share the I/O resource in proportion to I/O weights?
2. **Is it scalable?**

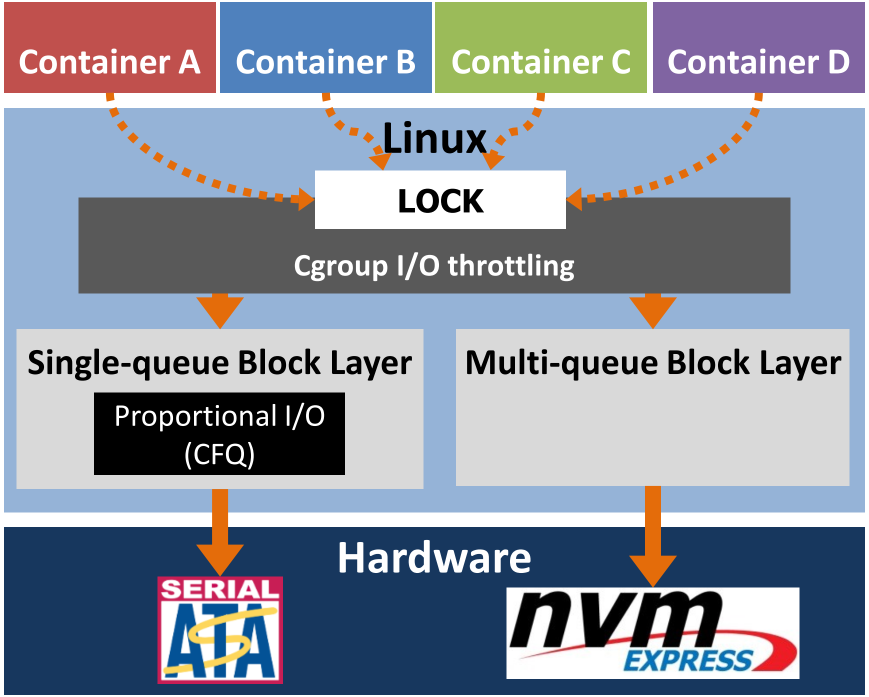
Scalability of the Existing Cgroups on NUMA

- **Scalability problem of the existing Cgroups throttling layer**

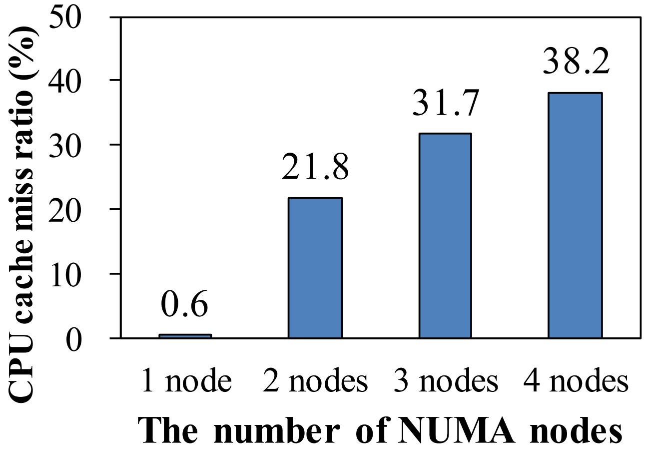


Because…

- **All containers share a single request_queue lock across NUMA nodes**

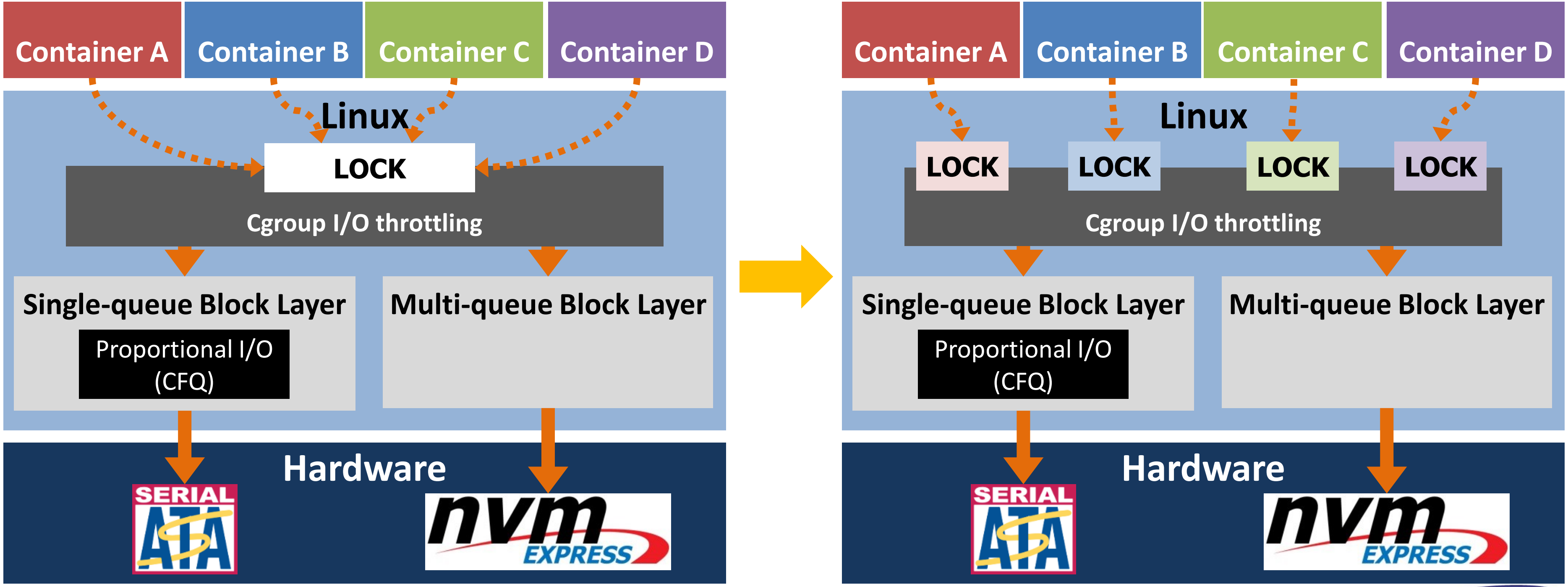


- ✓ **Lock contention**
- ✓ **Remote memory accesses to the lock state**
- ✓ **Cacheline invalidations caused by cache coherence protocol**



Per-container Locks

- We adopt fine-grained per-container locks
- The cache miss ratio decreases to 12.8% from 38.2%



Outline

- Introduction
- Motivation
- Contributions
- Weight-based dynamic throttling (WDT) scheme
- **Experimental Results**
- Conclusion

Experiment Setup

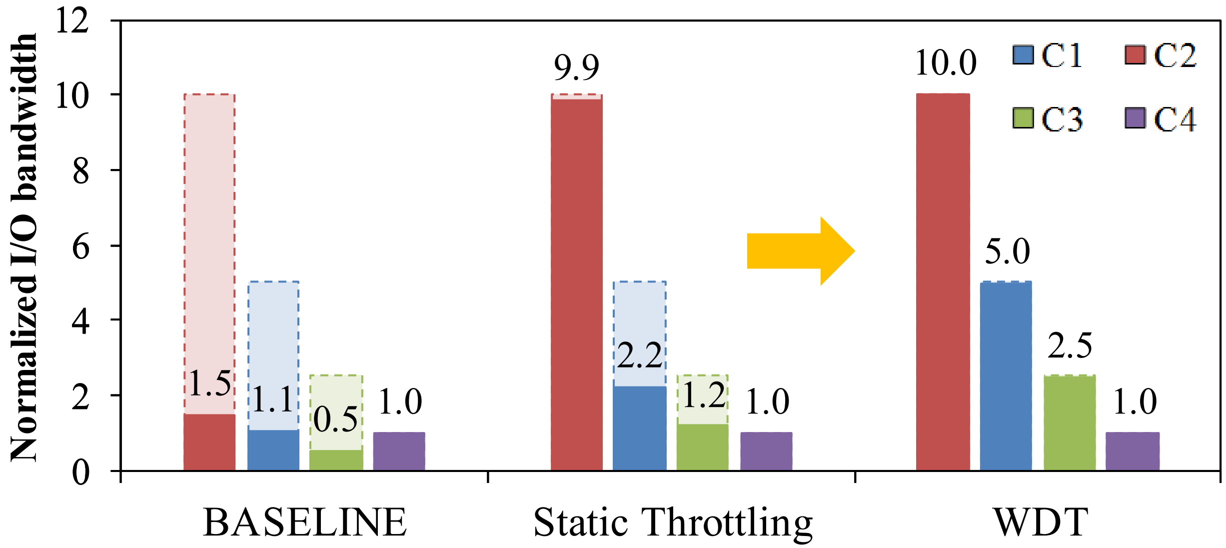
- **Linux kernel 4.0.4 (modified)**
- **Dell 4-nodes NUMA machine**
- **Samsung NVMe SSDs**
- **Block I/O traces replayer**
  - UMass trace repository
  - SNIA IOTTA repository

Dell R920 Server
(4-nodes NUMA machine)

Samsung XS1715 NVMe SSDs

Result 1: Proportional I/O Support

- **WDT scheme satisfies the proportional sharing requirements**



Result 2: Performance Scalability

- WDT- : Using single spin lock
- WDT : Using per-container locks

Outline

- Introduction
- Motivation
- Contributions
- Weight-based dynamic throttling (WDT) scheme
- Experimental Results
- **Conclusion**

Conclusion

- **Proposed the weight-based dynamic throttling scheme to support proportional I/O sharing for NVMe SSDs.**
- **Proposed the per-container locks for scalable performance.**

Align with your imagination

Thank you

SAMSUNG