Burstable Cloud Block Storage with Data Processing Units

Junyi Shu, Kun Qian, Ennan Zhai, Xuanzhe Liu, Xin Jin

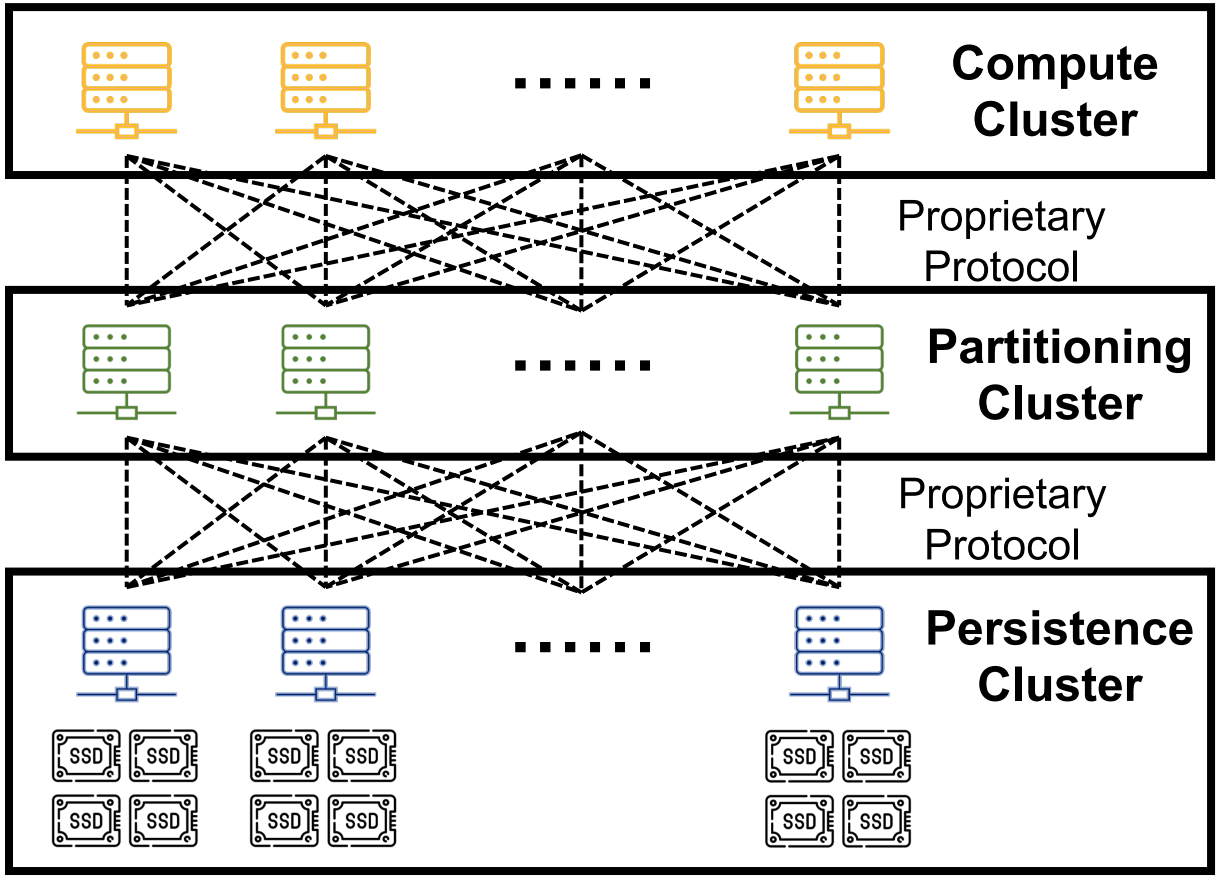
北京大學 PEKING UNIVERSITY

Alibaba Cloud

Cloud block storage

Three-layer disaggregated architecture

- Adopted by Alibaba (SIGCOMM '22), Azure (NSDI '23) and others
- Enhanced availability & elasticity

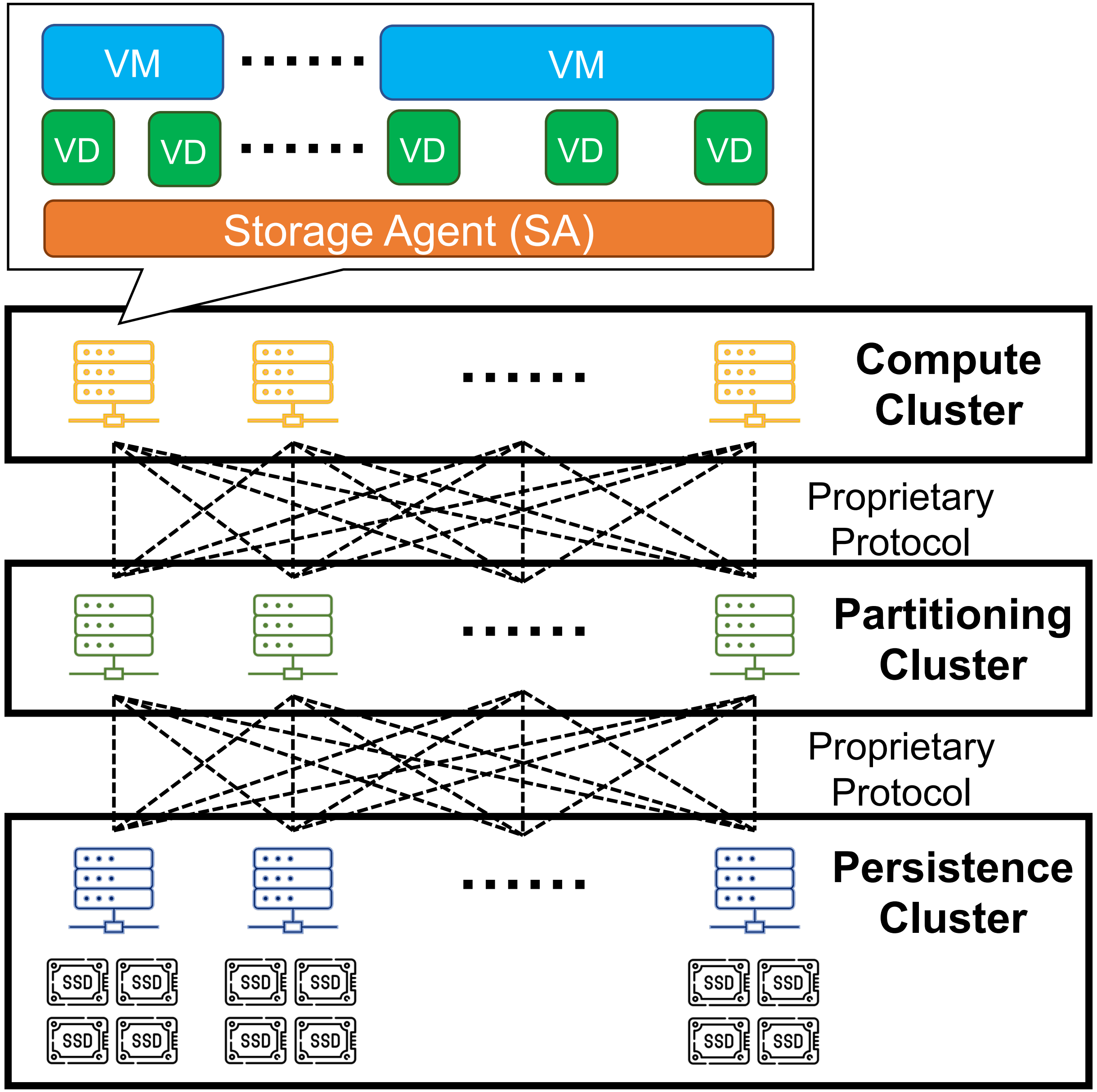


Cloud block storage

Three-layer disaggregated architecture

- Adopted by Alibaba (SIGCOMM ’22), Azure (NSDI ’23) and others
- Enhanced availability & elasticity

A compute node serves user VMs and virtual disks (VDs)



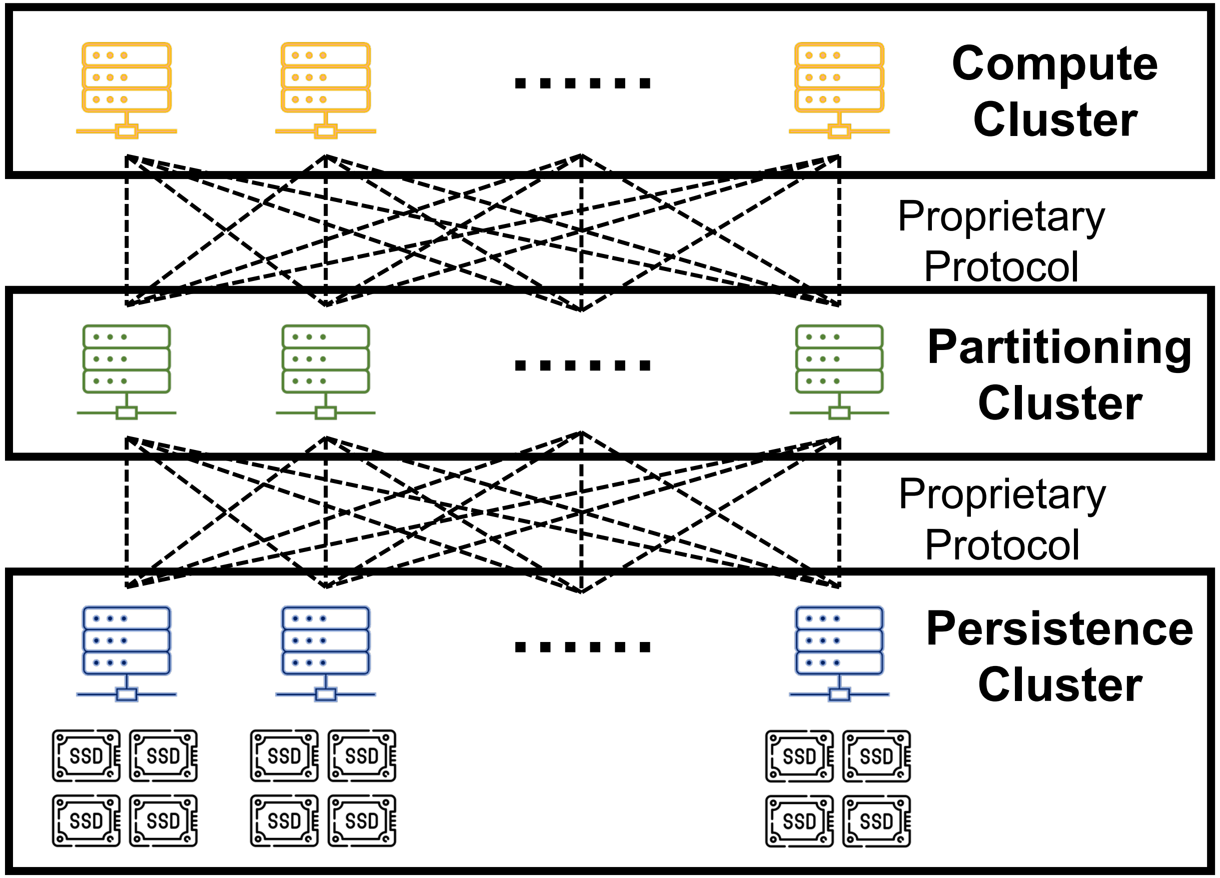
Cloud block storage

Three-layer disaggregated architecture

- Adopted by Alibaba (SIGCOMM '22), Azure (NSDI '23) and others
- Enhanced availability & elasticity

A compute node serves user VMs and virtual disks (VDs)

The persistence cluster is a DFS that stores data for many services



Cloud block storage

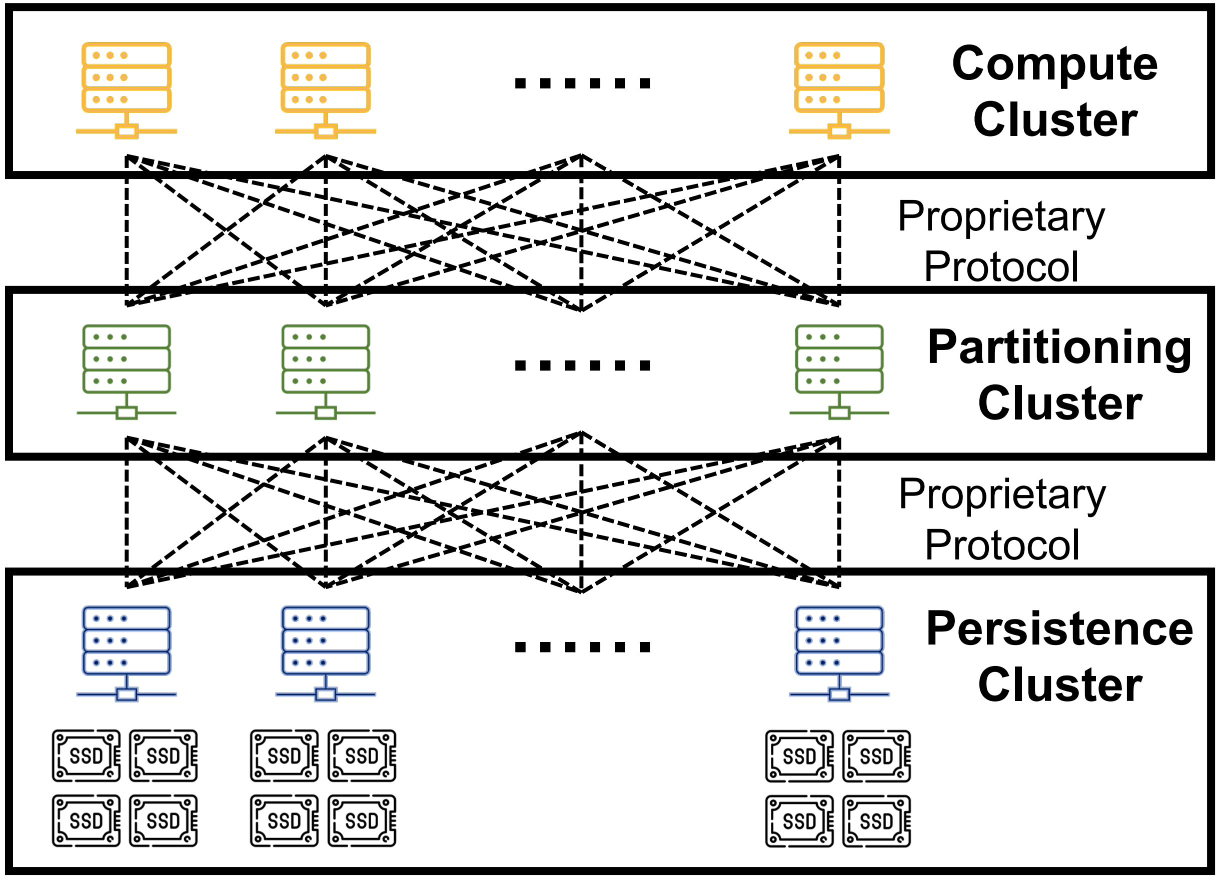
Three-layer disaggregated architecture

- Adopted by Alibaba (SIGCOMM ’22), Azure (NSDI ’23) and others
- Enhanced availability & elasticity

A compute node serves user VMs and virtual disks (VDs)

The partitioning cluster handles CBS-specific logic

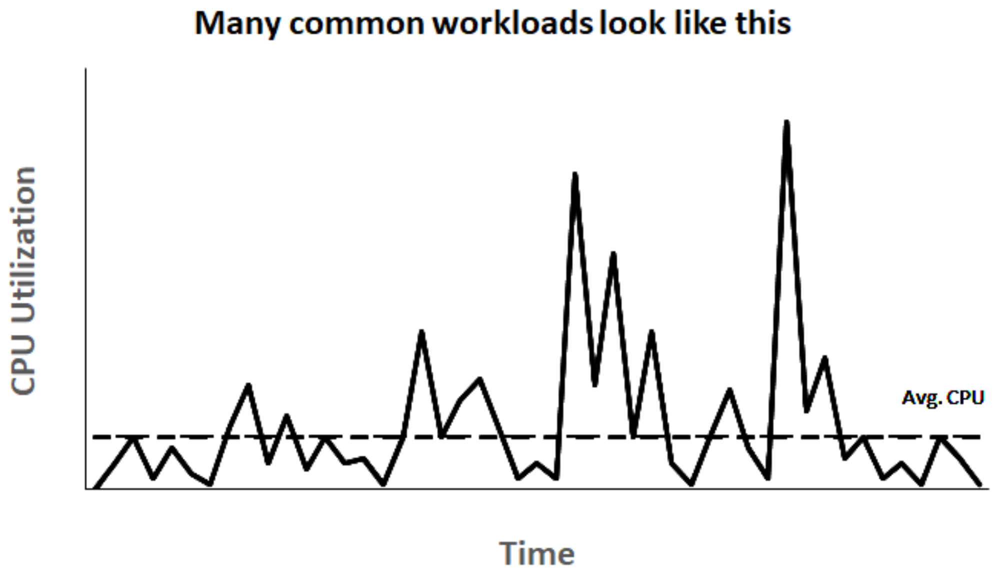
The persistence cluster is a DFS that stores data for many services



Beyond elasticity: burst at anytime

Burstable VM instance

- Provide a base-level CPU performance
- Able to burst above it anytime
- Suitable for fluctuated workload



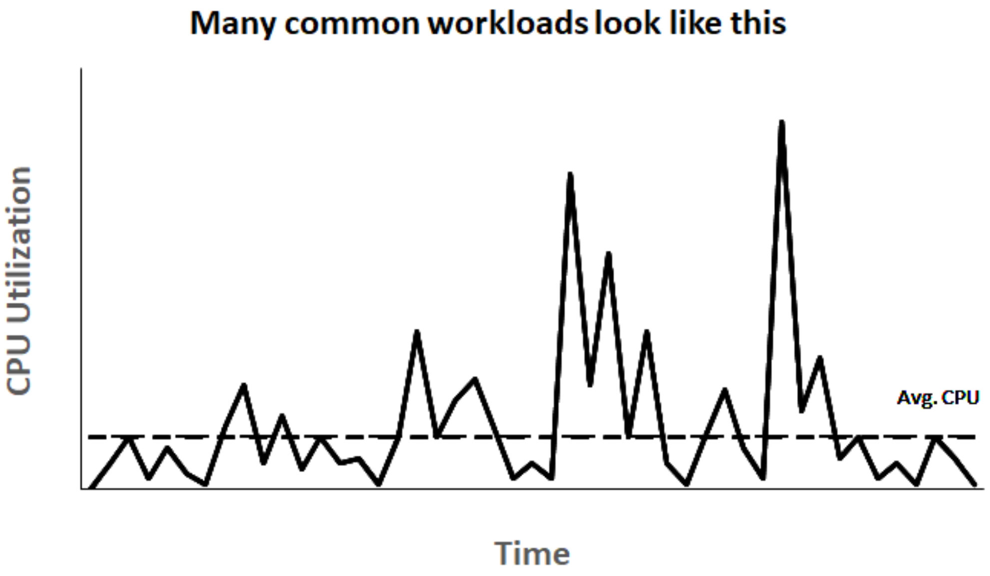
Beyond elasticity: burst at anytime

Burstable VM instance

- Provide a base-level CPU performance
- Able to burst above it anytime
- Suitable for fluctuated workload

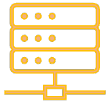
Burstable virtual disk

- Provide a base-level IOPS/bandwidth
- Able to saturate more I/Os when bursting
- I/O utilization is also fluctuated

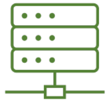


	CloudA	CloudB	Alibaba Cloud
Burst support	✓	✓	✓
Credit-based burst	✓	✓	✓
Paid burst	✗	✓	✓
Max burst IOPS	3k	30k	1000k
Max burst BPS (MB/s)	N/A	1000	4096

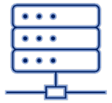
How is provisionable IOPS/bandwidth determined?



Compute Node

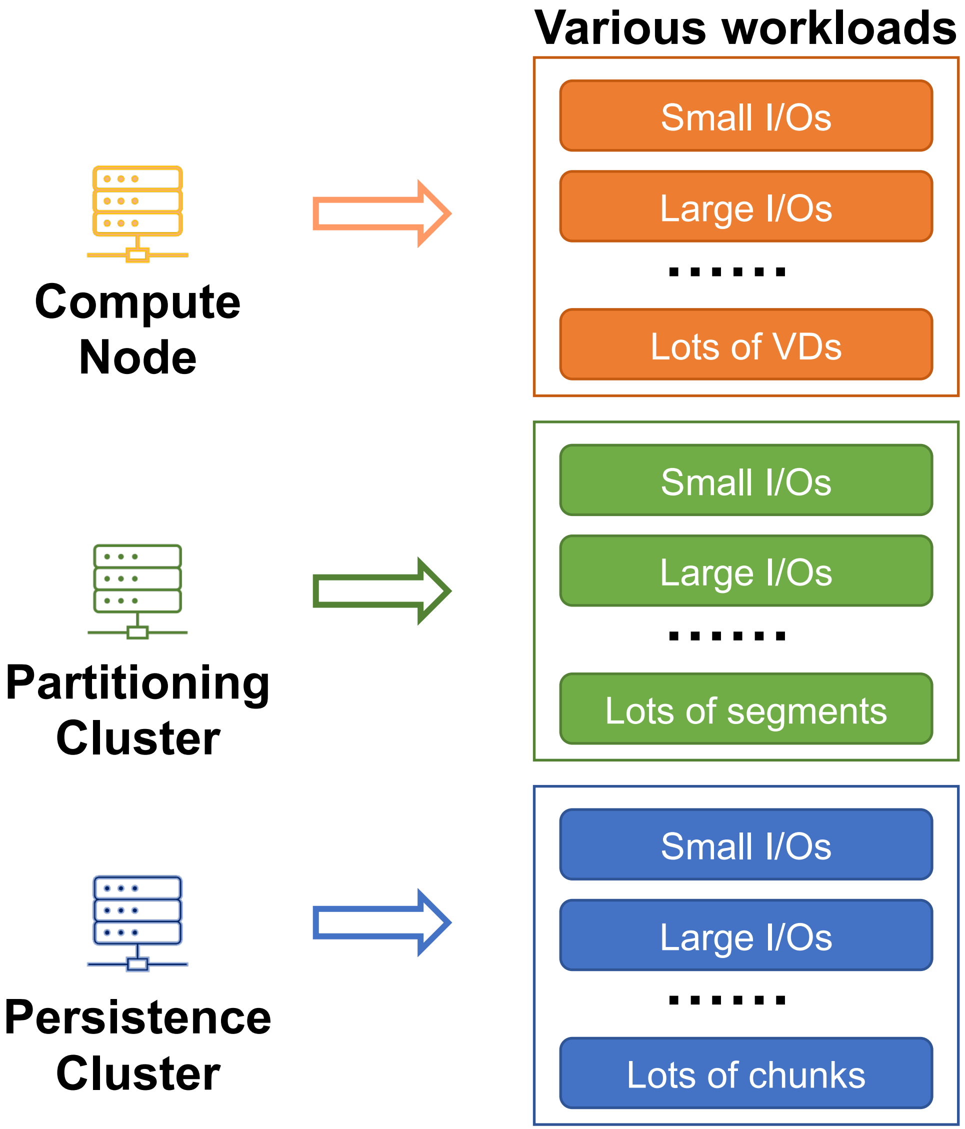


Partitioning Cluster

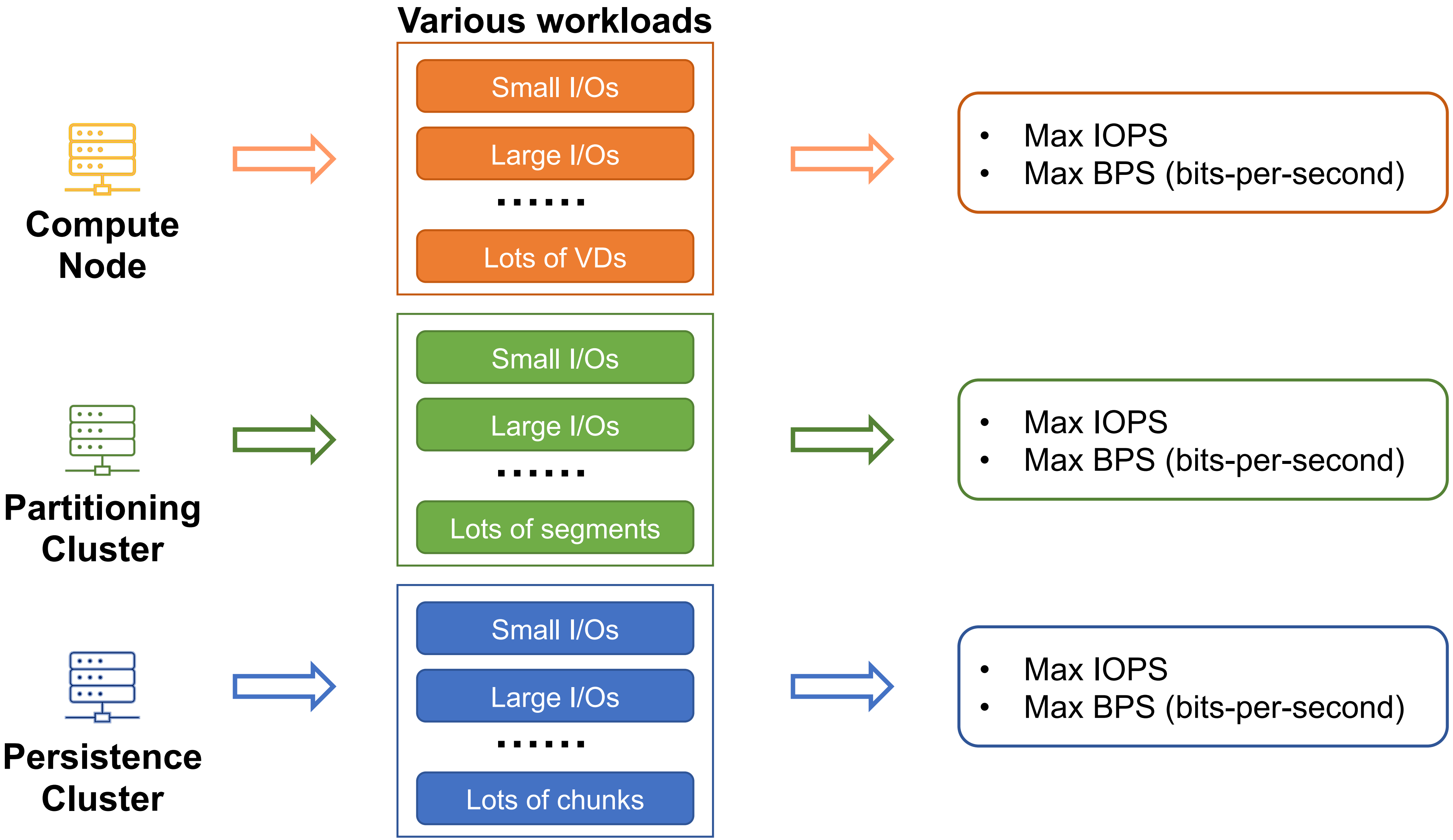


Persistence Cluster

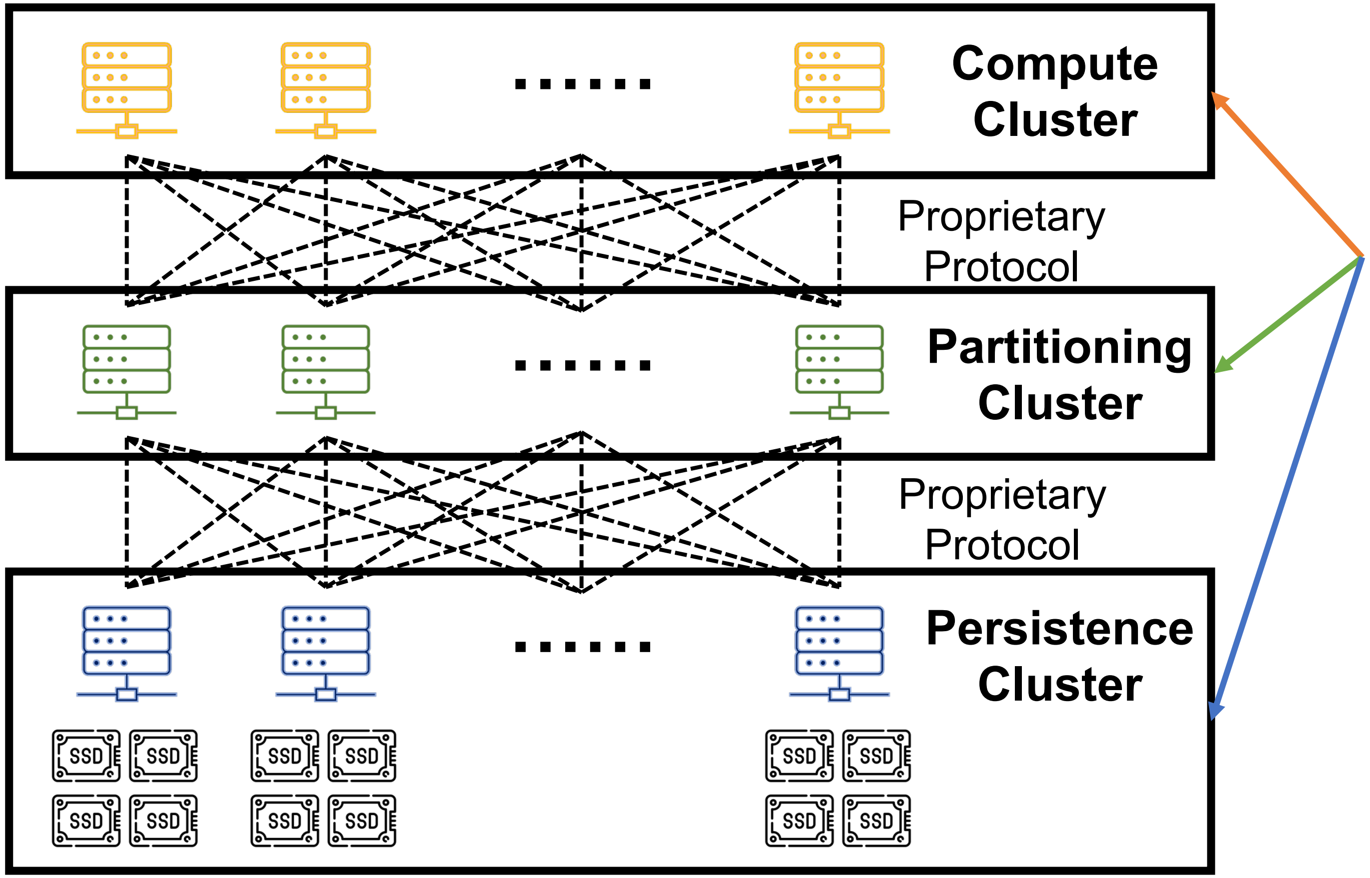
How is provisionable IOPS/bandwidth determined?



How is provisionable IOPS/bandwidth determined?



Do we have the IOPS/bandwidth to support burst?



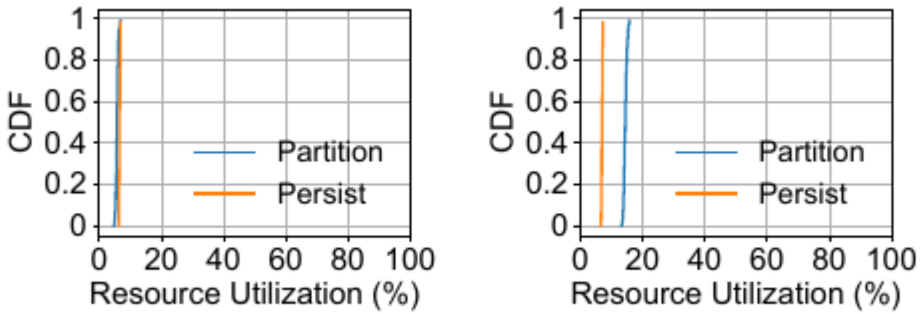
Supporting burst requires spare IOPS/bandwidth at each layer

Do we need to provision additional resources just for supporting burst?

Insight 1: low utilization of backend clusters

IOPS/BPS usage is low

- **~78%** disk capacity is used
- **<20%** IOPS/BPS is used
- Vertical CDF -> **well balanced load**

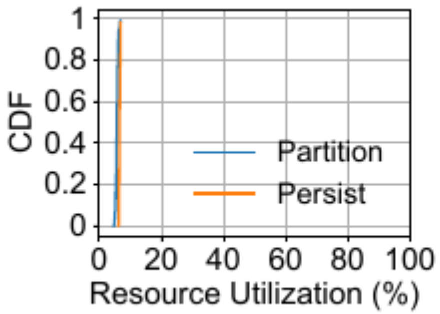


(a) IOPS (b) BPS

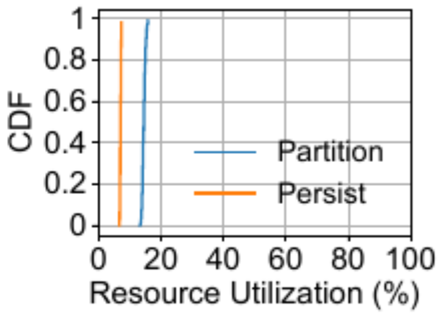
Insight 1: low utilization of backend clusters

IOPS/BPS usage is low

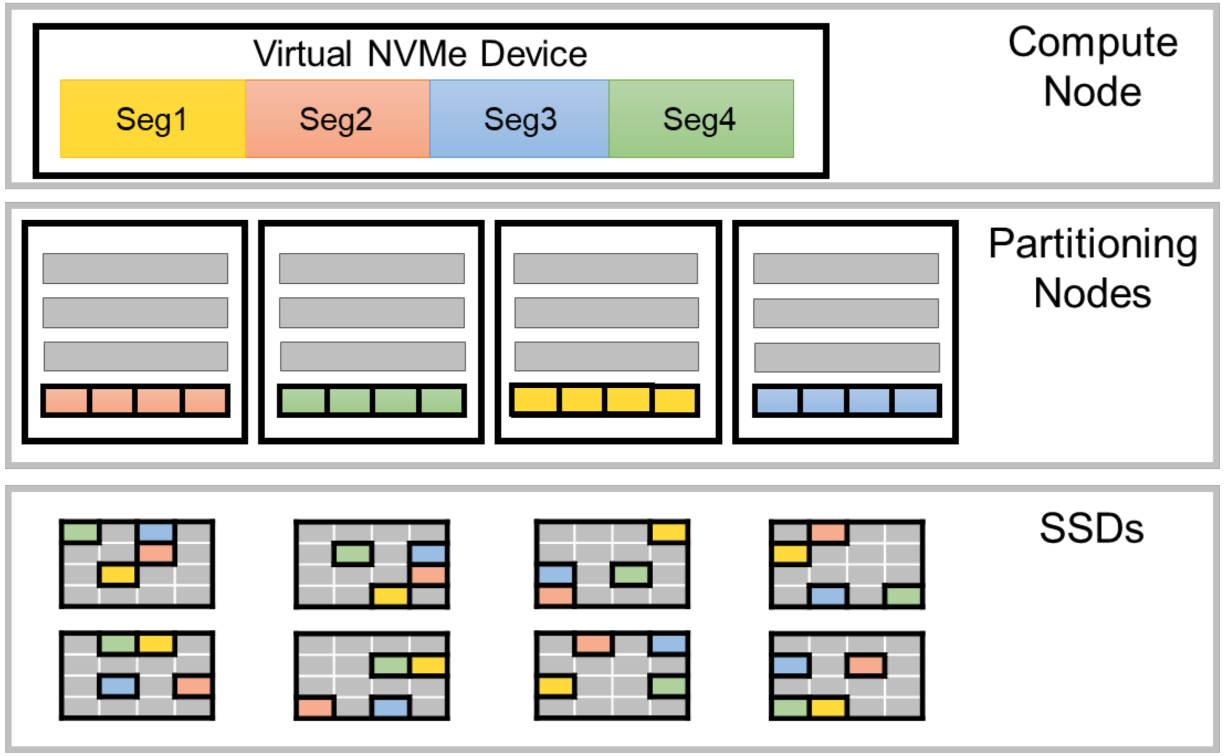
- **~78%** disk capacity is used
- **<20%** IOPS/BPS is used
- Vertical CDF -> **well balanced load**



(a) IOPS

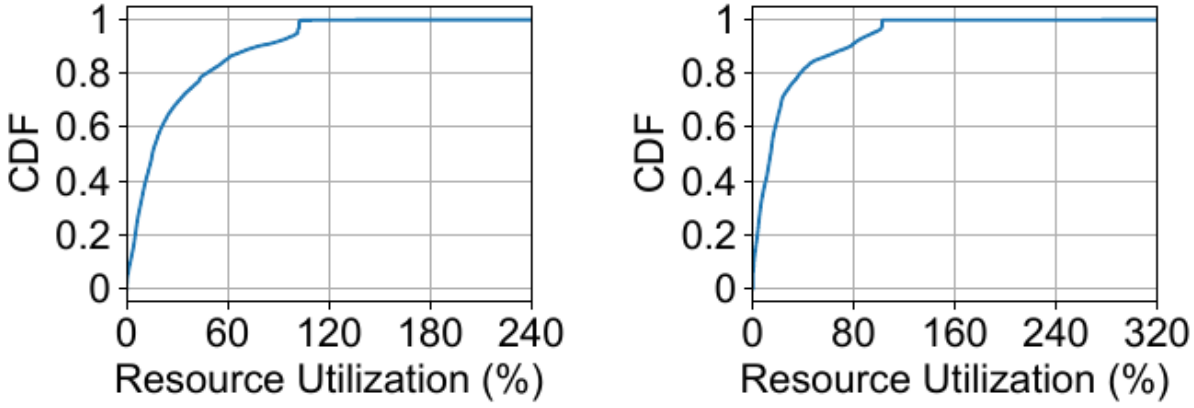


(b) BPS



Insight 2: diverse utilization of compute clusters

IOPS/BPS distribution of all tenants in a compute cluster

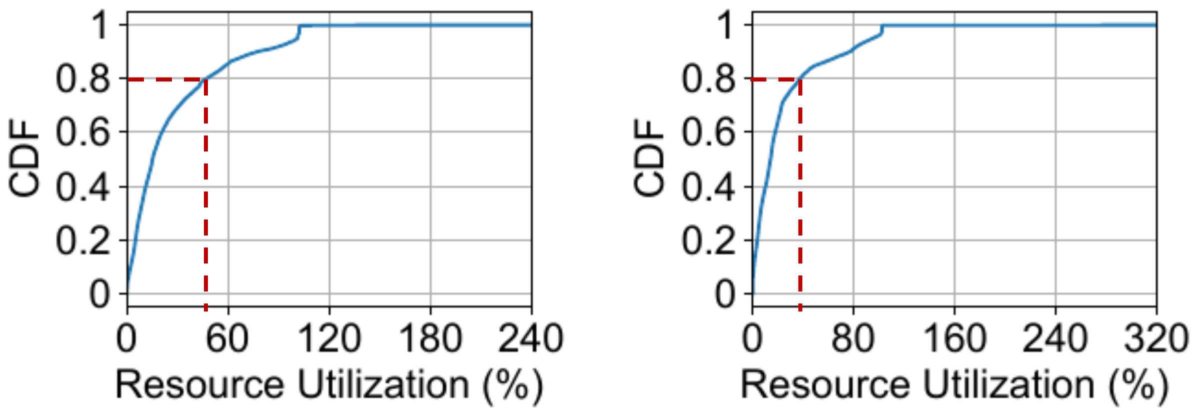


(a) IOPS (b) BPS

Insight 2: diverse utilization of compute clusters

IOPS/BPS distribution of all tenants in a compute cluster

- Over **80%** of the tenants use only **<50%** of their provisioned IOPS/BPS

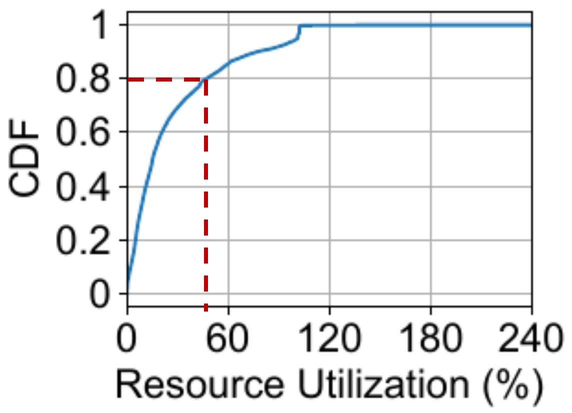


(a) IOPS (b) BPS

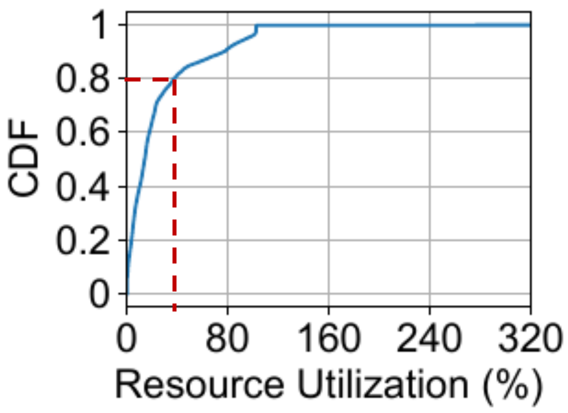
Insight 2: diverse utilization of compute clusters

IOPS/BPS distribution of all tenants in a compute cluster

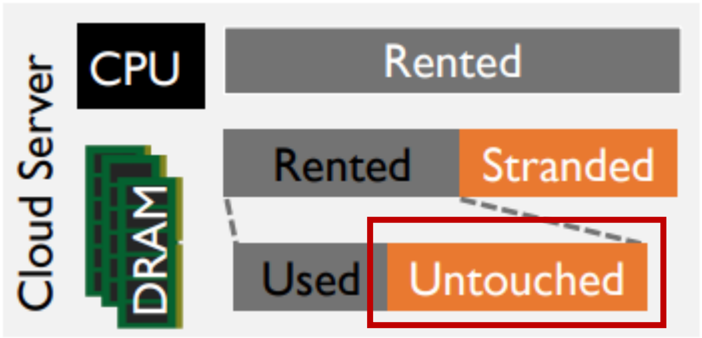
- Over **80%** of the tenants use only **<50%** of their provisioned IOPS/BPS
- Overprovisioning tendency is common in public clouds



(a) IOPS



(b) BPS

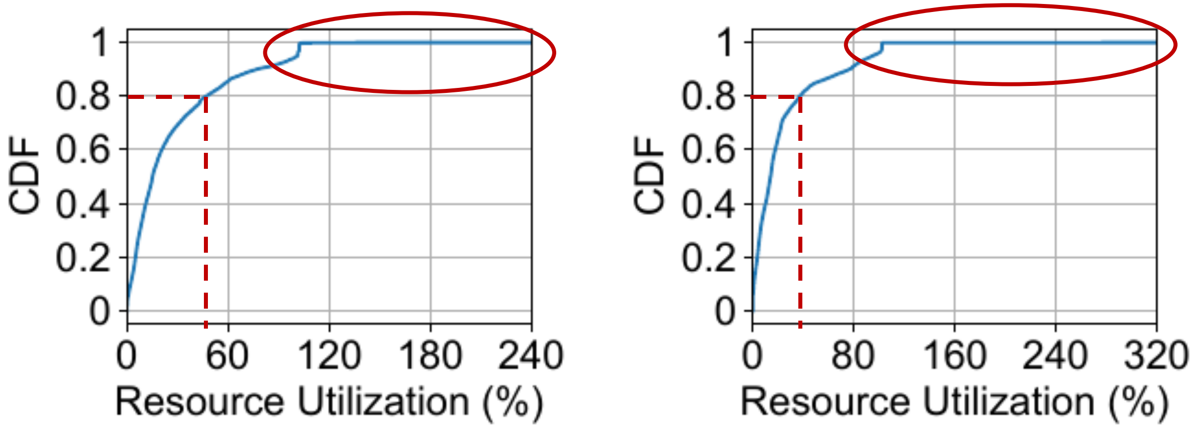


Pond [ASPLOS '23]

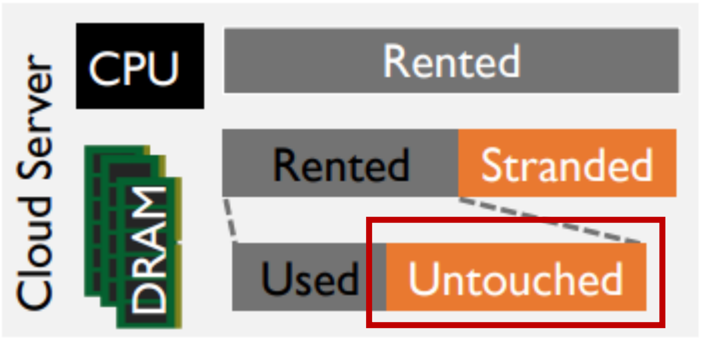
Insight 2: diverse utilization of compute clusters

IOPS/BPS distribution of all tenants in a compute cluster

- Over **80%** of the tenants use only **<50%** of their provisioned IOPS/BPS
- Overprovisioning tendency is common in public clouds
- Long tail on the need for bursting



(a) IOPS (b) BPS

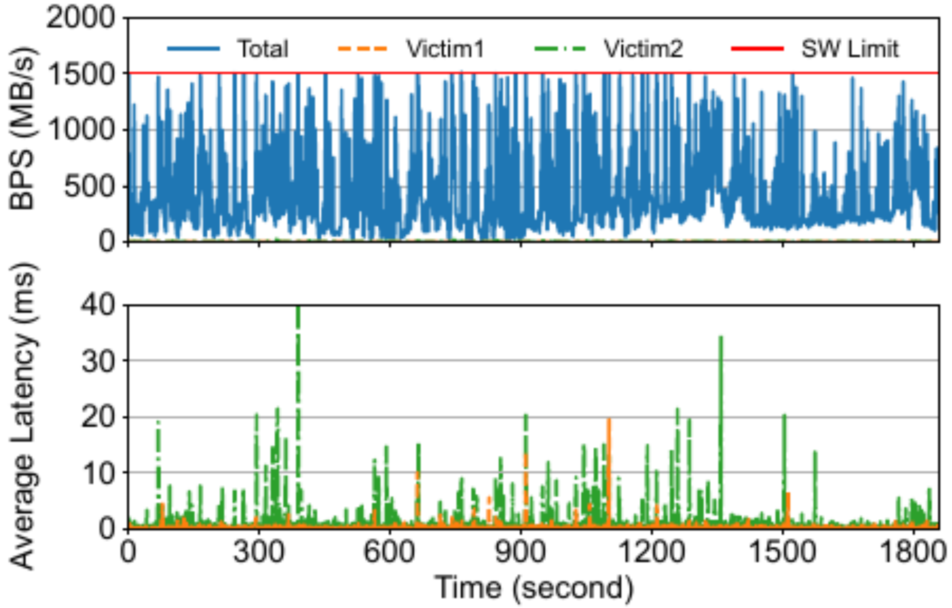


Pond [ASPLOS '23]

What if we just allow tenants to burst in the wild?

The base-level tenants suffer high latency

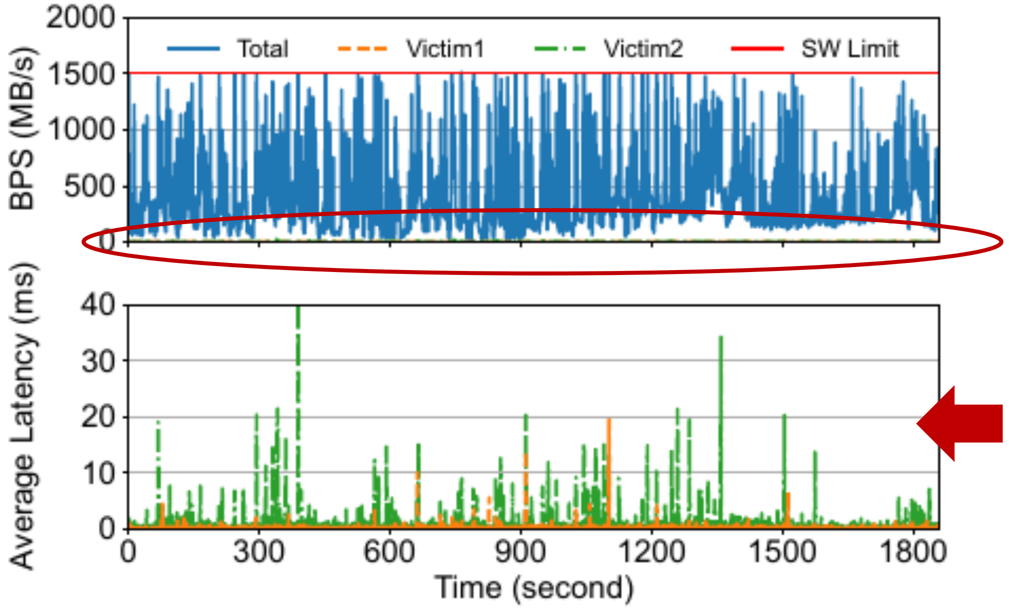
- The traffic created by Victim1 and Victim2 is negligible
- Overall BPS utilization reaches 100% frequently
- Victims experience **>10ms** average latency (norm is sub-millisecond)



What if we just allow tenants to burst in the wild?

The base-level tenants suffer high latency

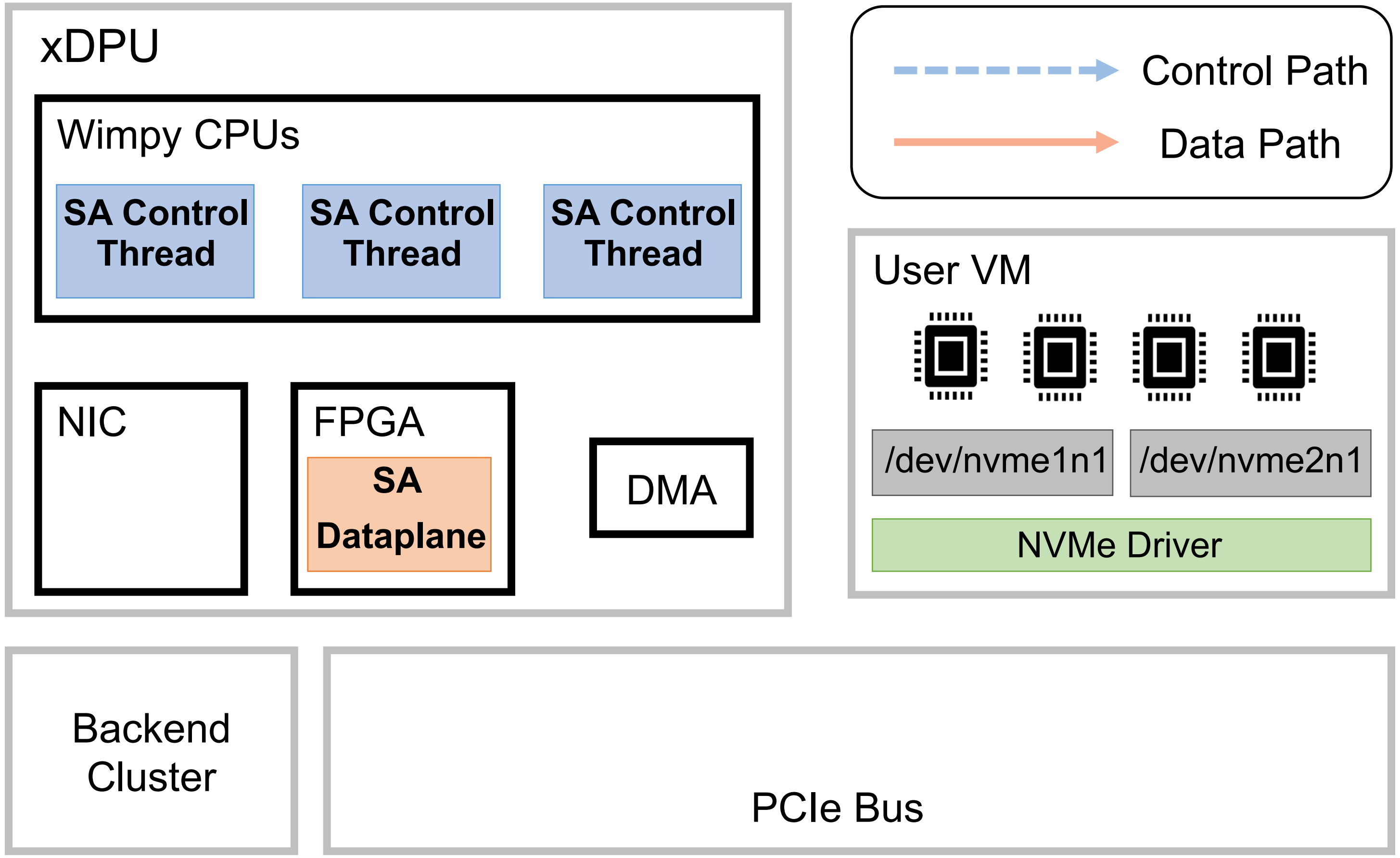
- The traffic created by Victim1 and Victim2 is negligible
- Overall BPS utilization reaches 100% frequently
- Victims experience **>10ms** average latency (norm is sub-millisecond)



Why?

Key component of a compute node: xDPU

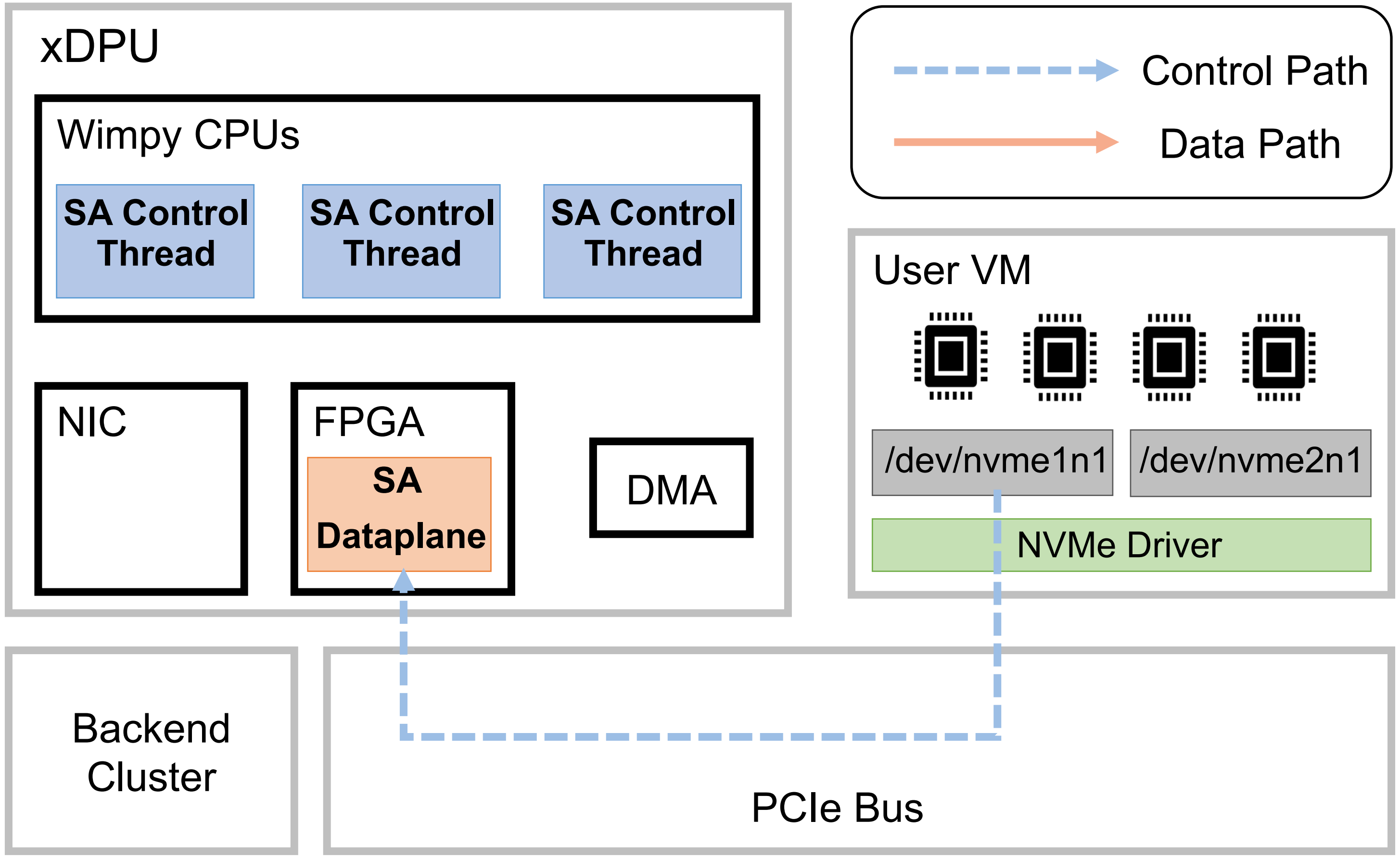
I/O workflow



Key component of a compute node: xDPU

I/O workflow

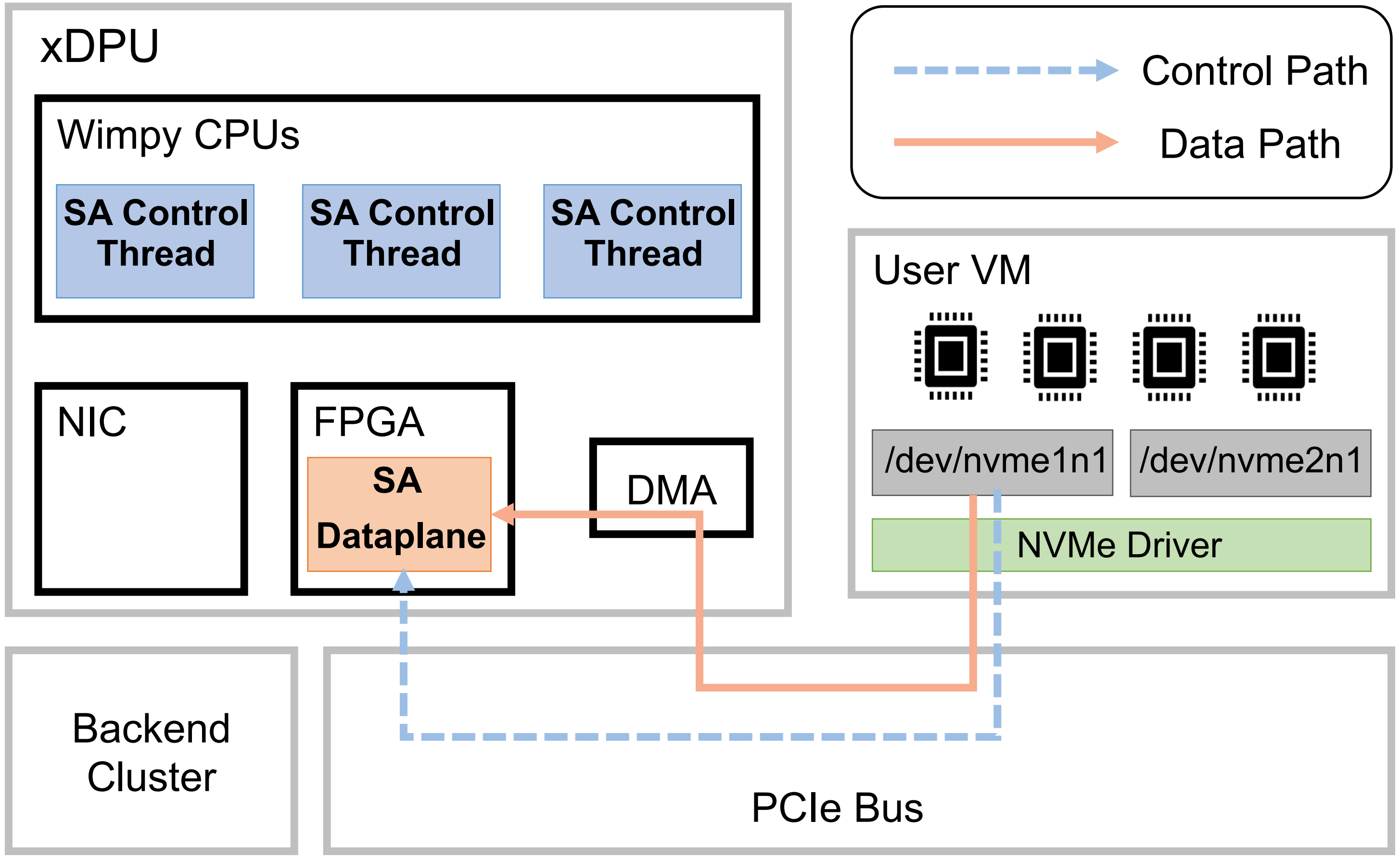
- T0: NVMe control command arrives



Key component of a compute node: xDPU

I/O workflow

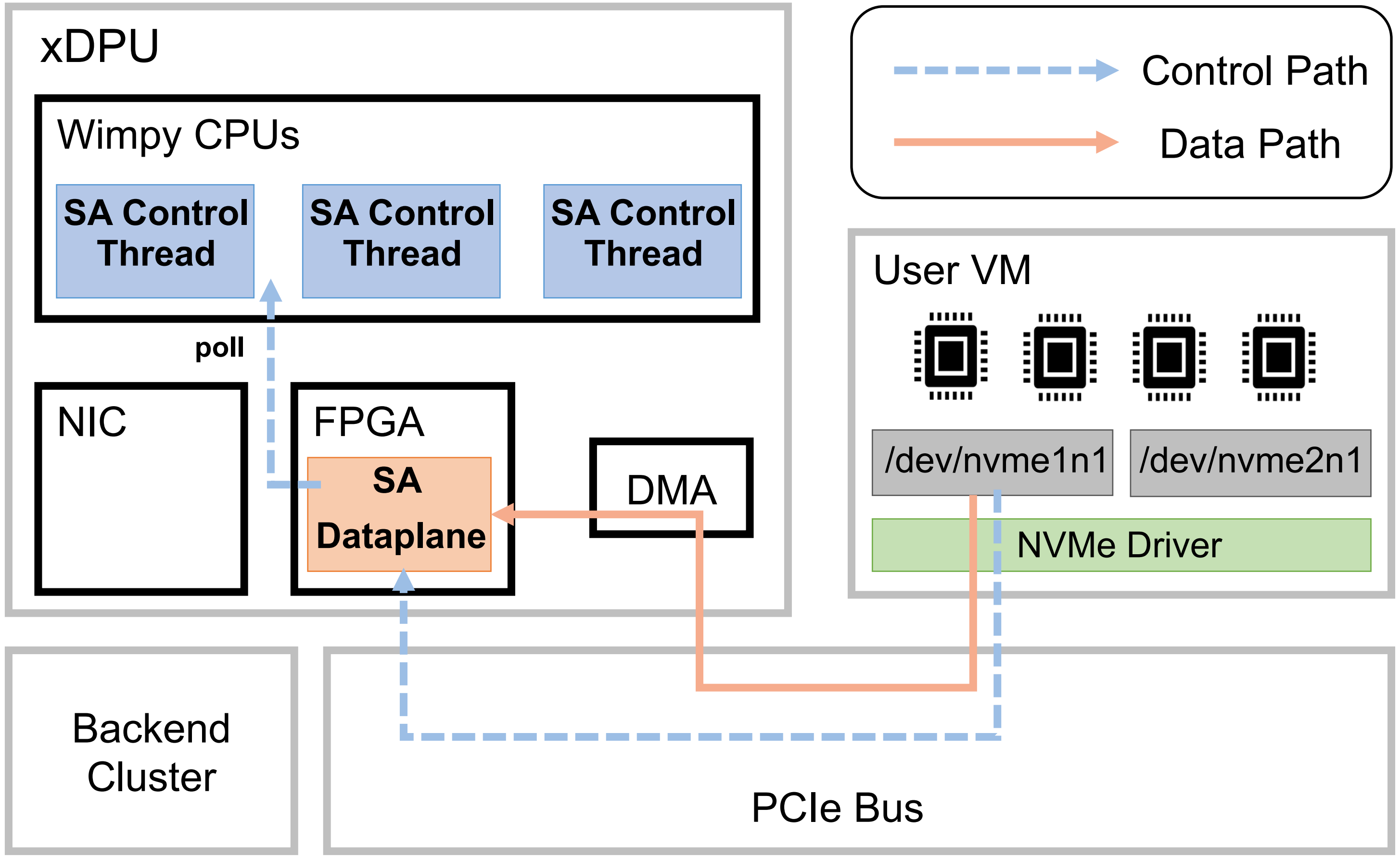
- T0: NVMe control command arrives
- T1: data moves to DPU



Key component of a compute node: xDPU

I/O workflow

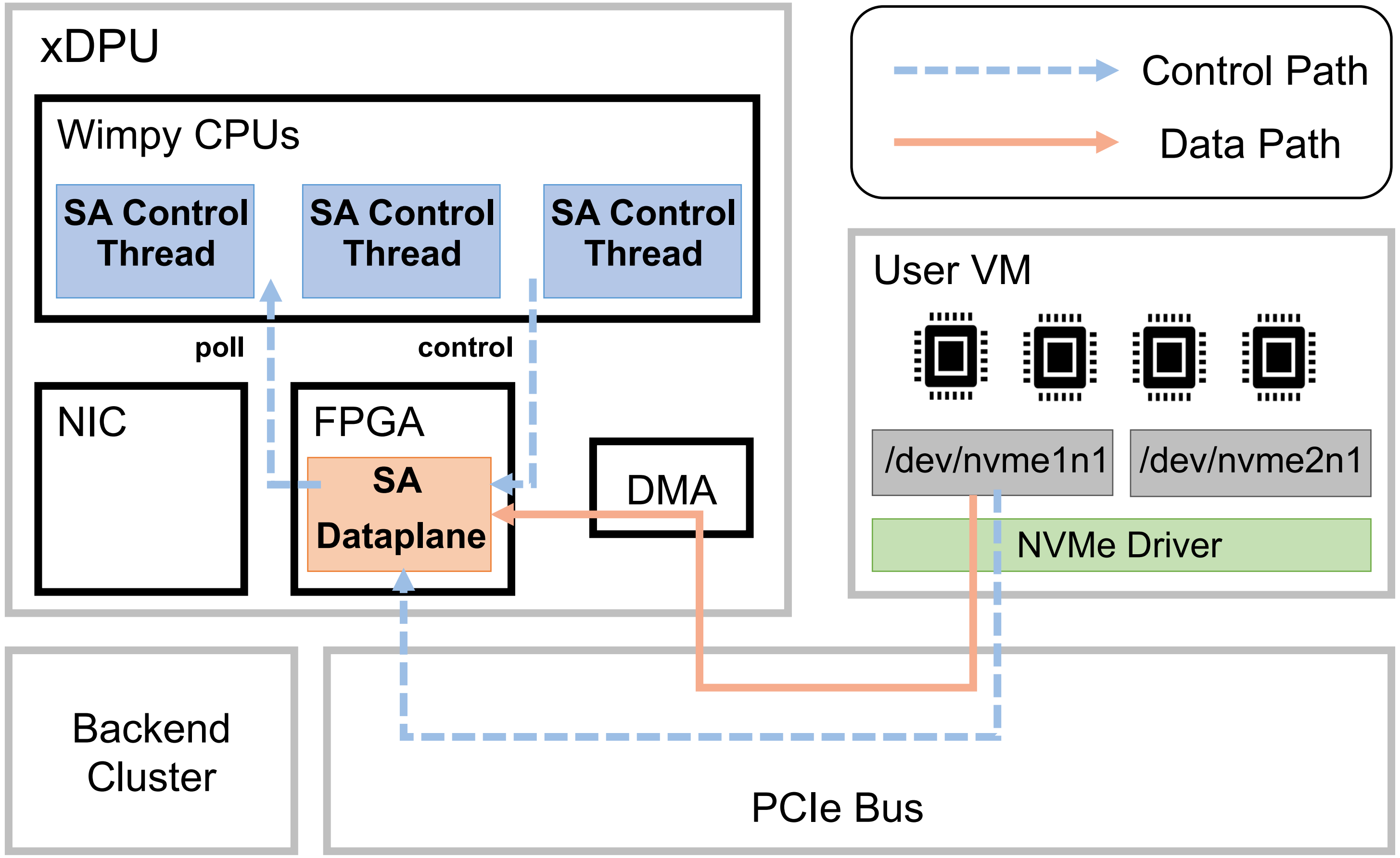
- T0: NVMe control command arrives
- T1: data moves to DPU
- T1: control thread polls I/O metadata



Key component of a compute node: xDPU

I/O workflow

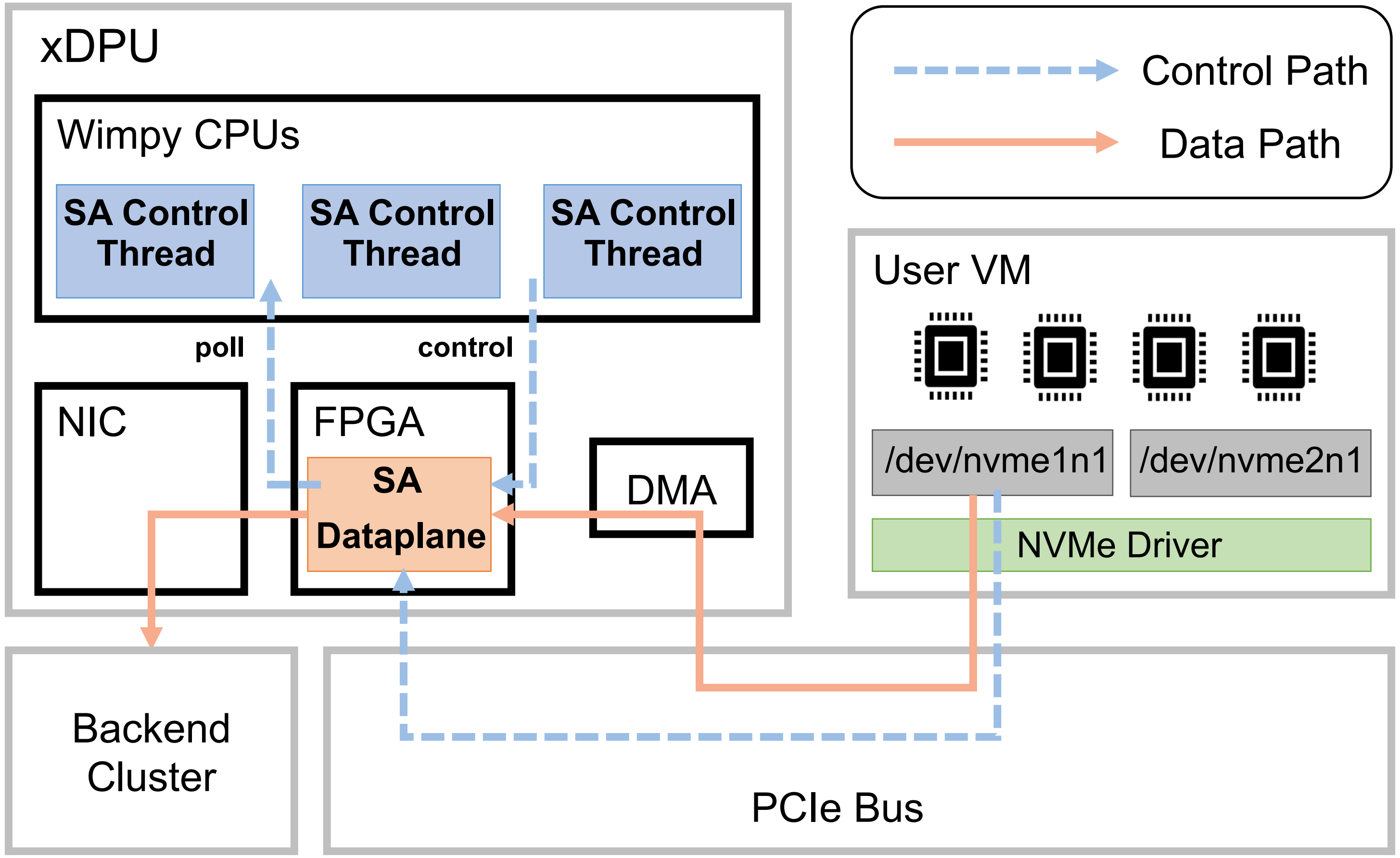
- T0: NVMe control command arrives
- T1: data moves to DPU
- T1: control thread polls I/O metadata
- T2: control thread finish header encapsulation and send signal



Key component of a compute node: xDPU

I/O workflow

- T0: NVMe control command arrives
- T1: data moves to DPU
- T1: control thread polls I/O metadata
- T2: control thread finish header encapsulation and send signal
- T3: send I/O when both header and data are ready



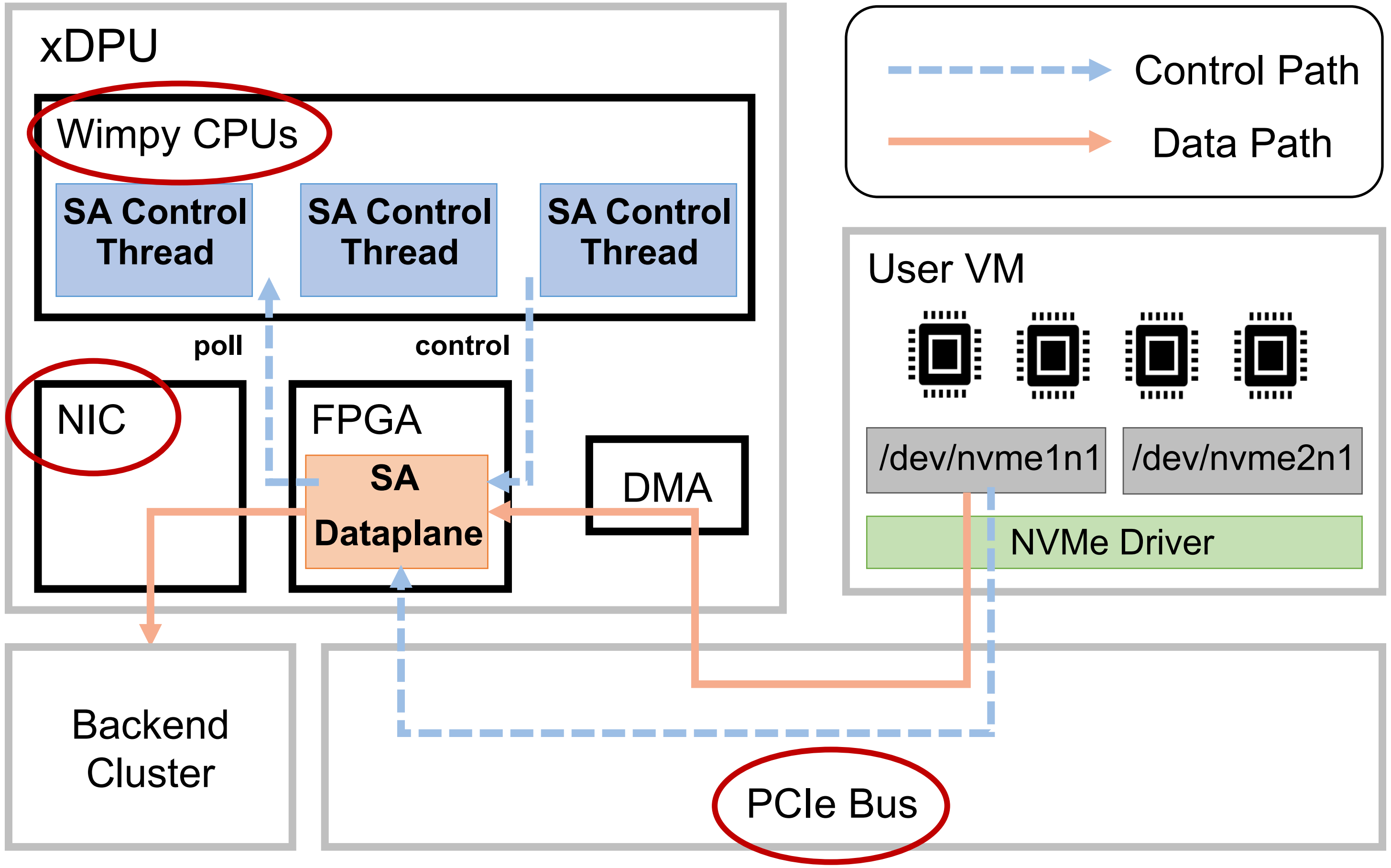
Key component of a compute node: xDPU

I/O workflow

- T0: NVMe control command arrives
- T1: data moves to DPU
- T1: control thread polls I/O metadata
- T2: control thread finish header encapsulation and send signal
- T3: send I/O when both header and data are ready

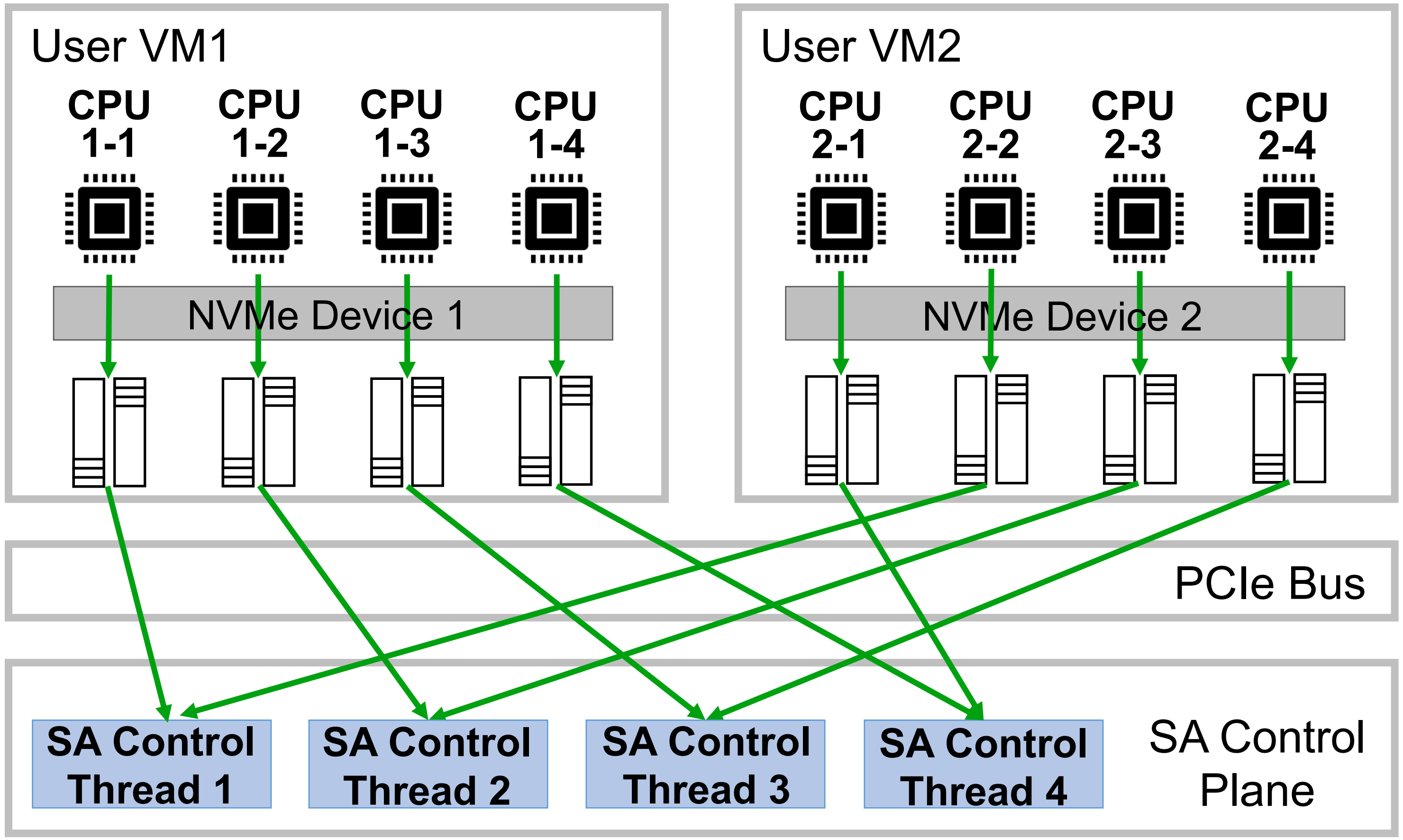
Bottlenecks

- Wimpy CPUs
- Interconnects: PCIe + NIC



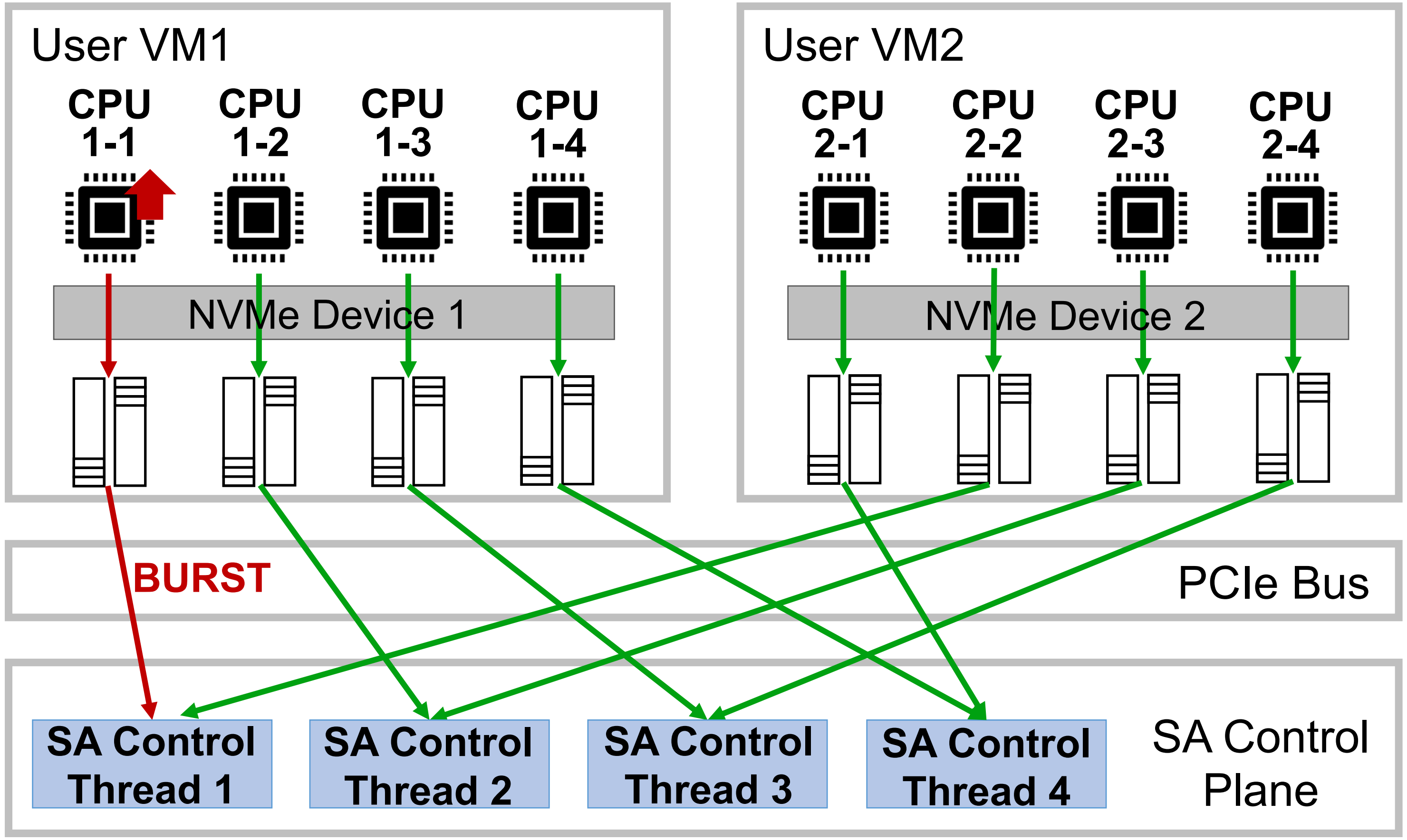
Insight 3: load imbalance on xDPU

An illustrative example of load imbalance



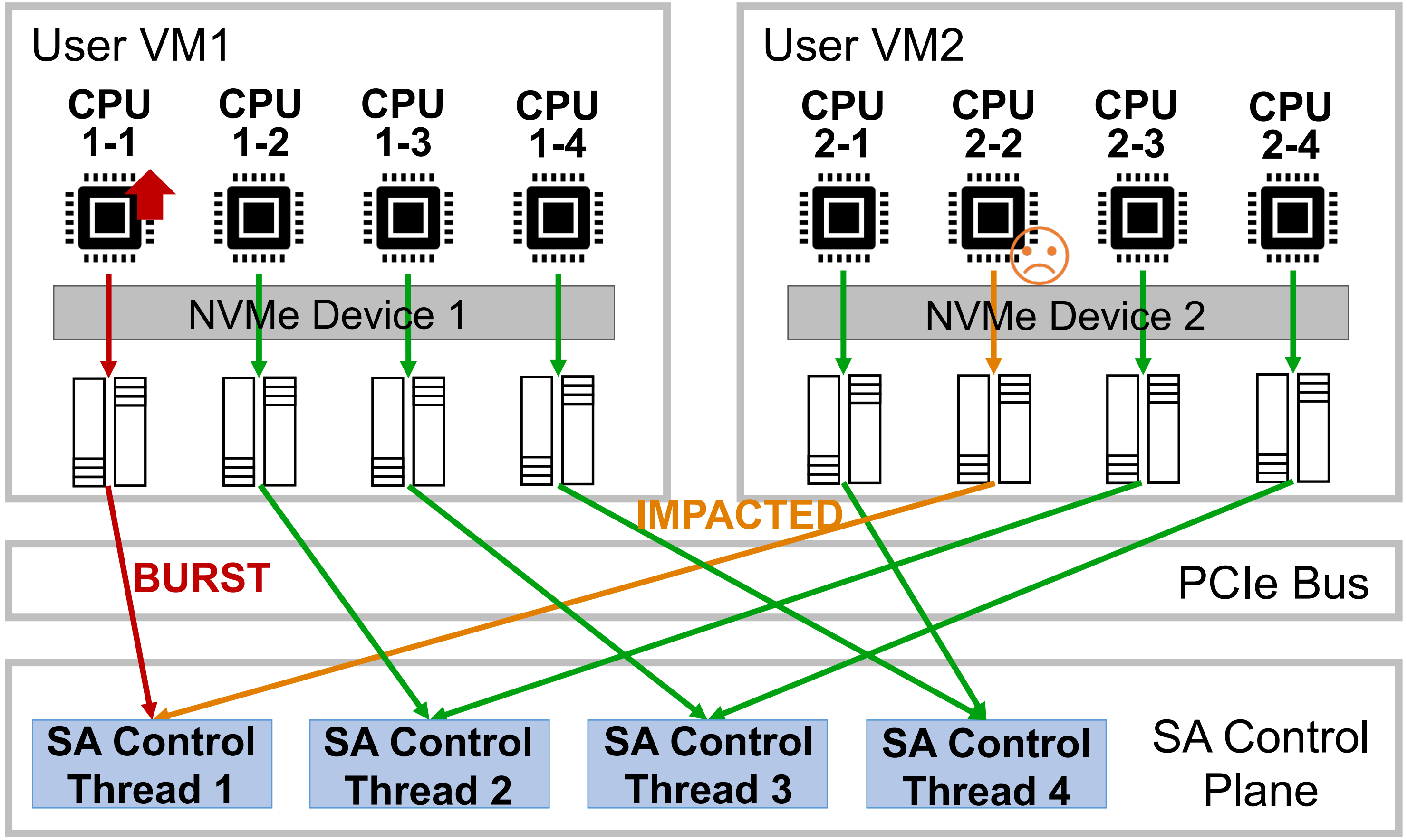
Insight 3: load imbalance on xDPU

An illustrative example of load imbalance



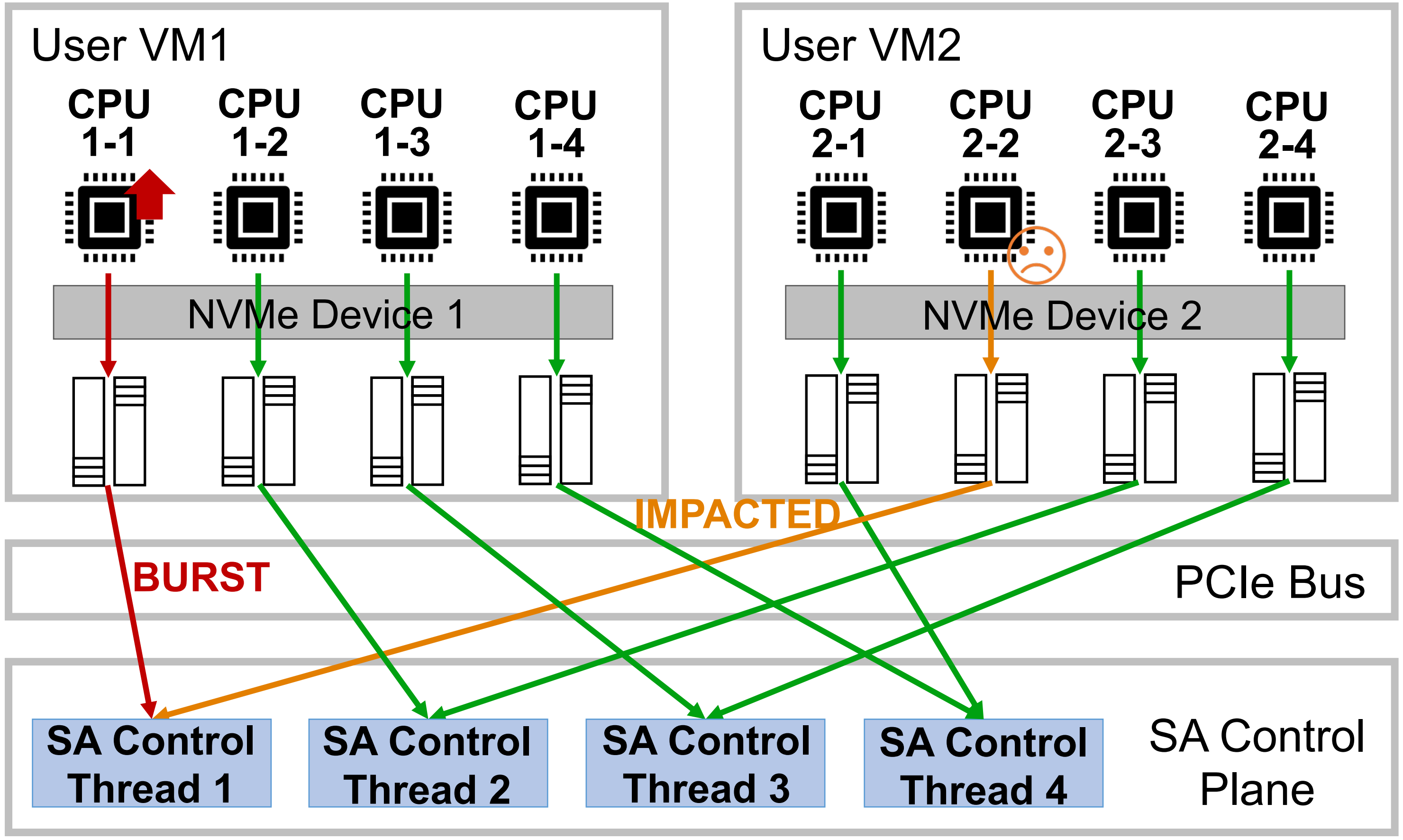
Insight 3: load imbalance on xDPU

An illustrative example of load imbalance



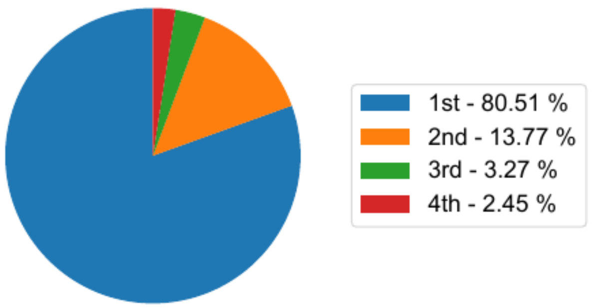
Insight 3: load imbalance on xDPU

An illustrative example of load imbalance



vCPUs are not evenly used

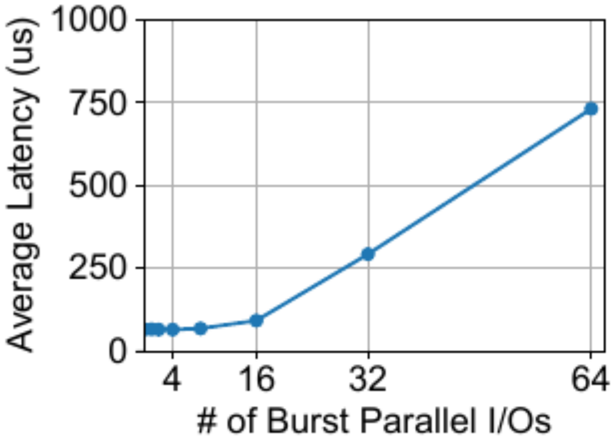
- **>80%** of the I/Os are processed by the busiest core on 4-core VMs.



Insight 4: resource competition on xDPU

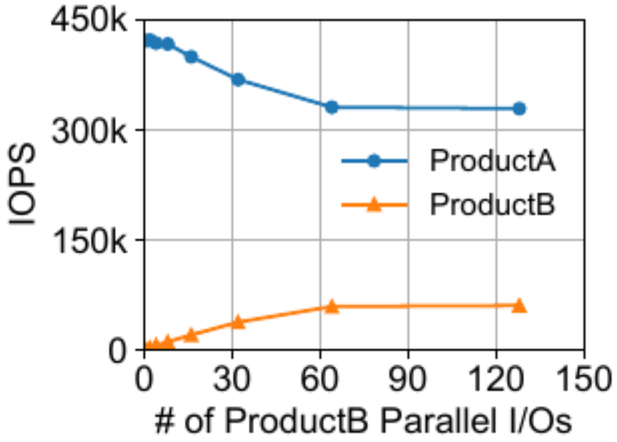
Case 1: victim latency at burst

- Within a thread, higher burst results in latency increase of victims



Case 2: different products

- Different products vary in their ability to compete for resources



Opportunity, challenges, and goal

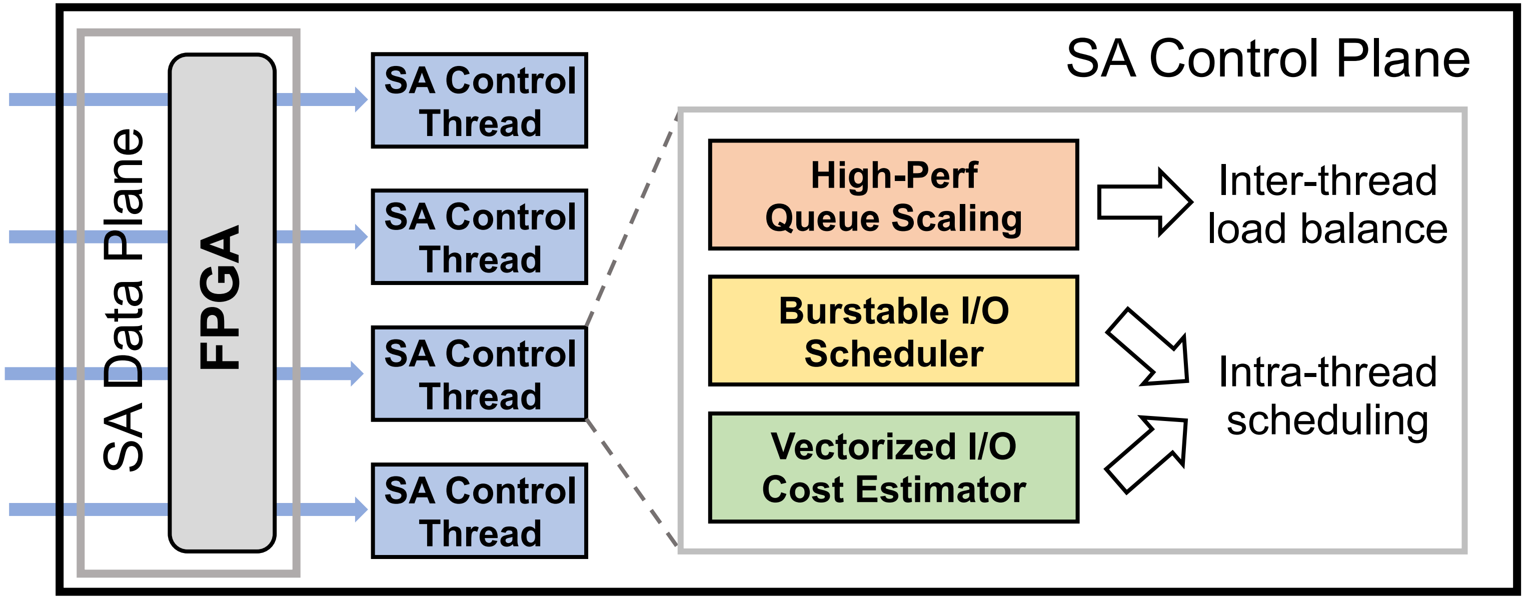
- Non-bottleneck: low backend resource utilization
- Opportunity: diverse utilization on compute nodes
- Challenge 1: load imbalance among threads
- Challenge 2: undesired resource allocation within a thread

Opportunity, challenges, and goal

- Non-bottleneck: low backend resource utilization
- Opportunity: diverse utilization on compute nodes
- Challenge 1: load imbalance among threads
- Challenge 2: undesired resource allocation within a thread

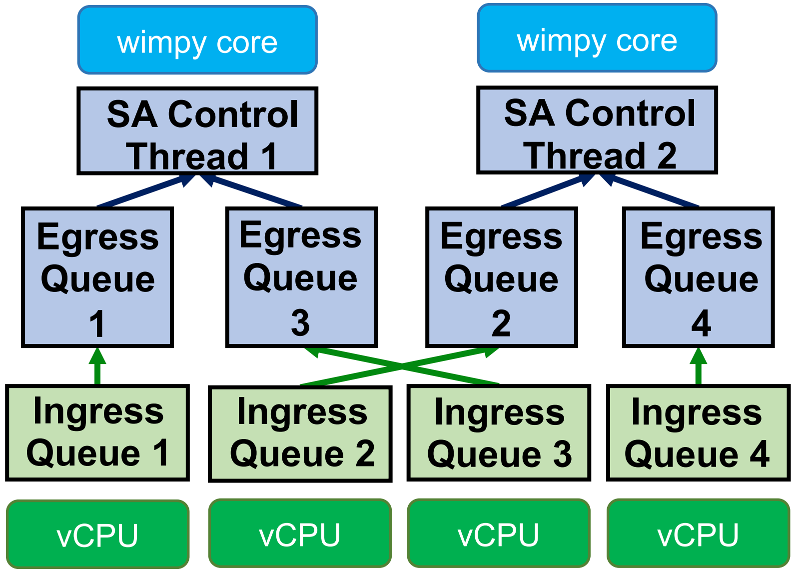
We need to design an I/O scheduling system on xDPU that:
1) keeps the load balanced among threads to avoid congestion;
2) and allocates resources with a thread to support burst and limit tenant interference.

BurstCBS Overview



FPGA-assisted inter-thread load-balancing

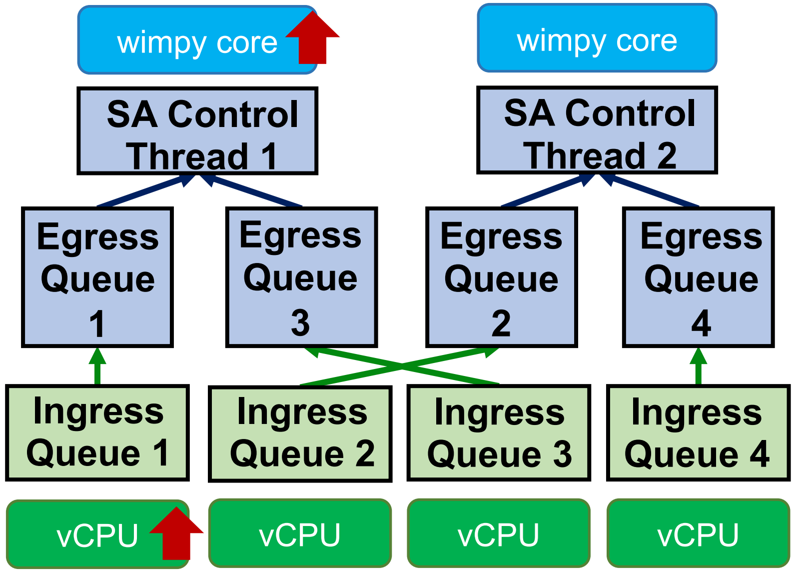
1-to-1 mapping



FPGA-assisted inter-thread load-balancing

1-to-1 mapping

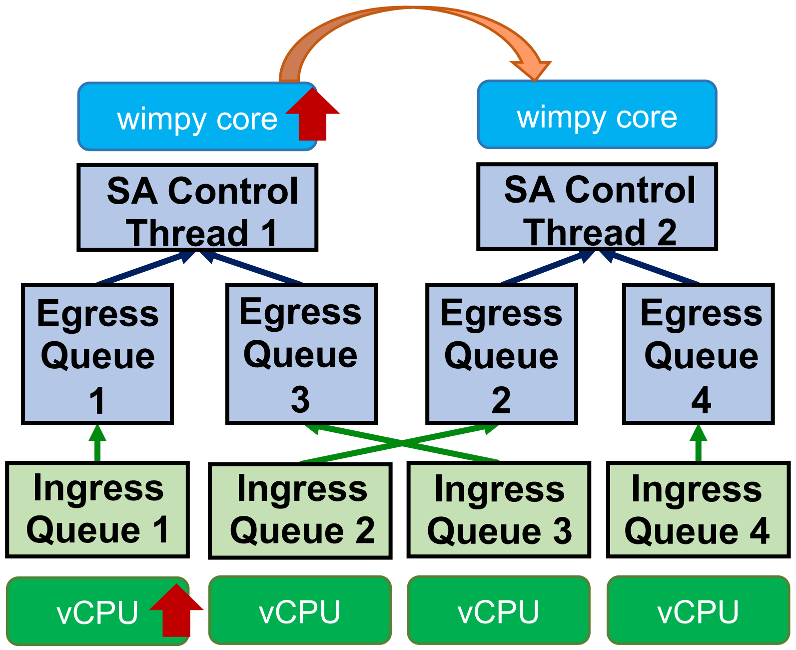
- Causes load imbalance when a vCPU starts bursting



FPGA-assisted inter-thread load-balancing

1-to-1 mapping

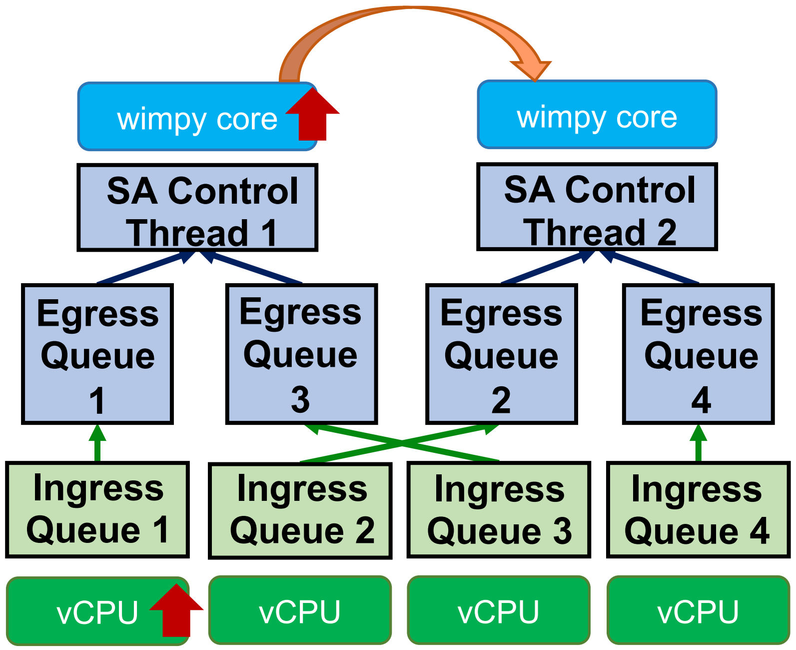
- Causes load imbalance when a vCPU starts bursting
- **Work stealing** can alleviate load imbalance

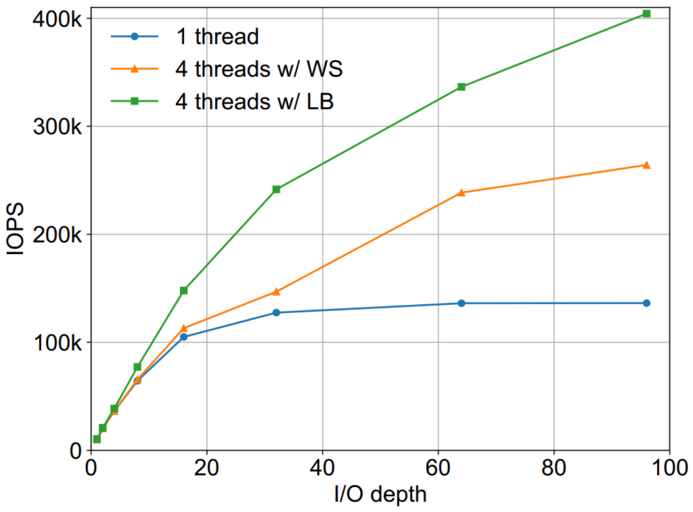


FPGA-assisted inter-thread load-balancing

1-to-1 mapping

- Causes load imbalance when a vCPU starts bursting
- **Work stealing** can alleviate load imbalance
- **~35%** throughput loss





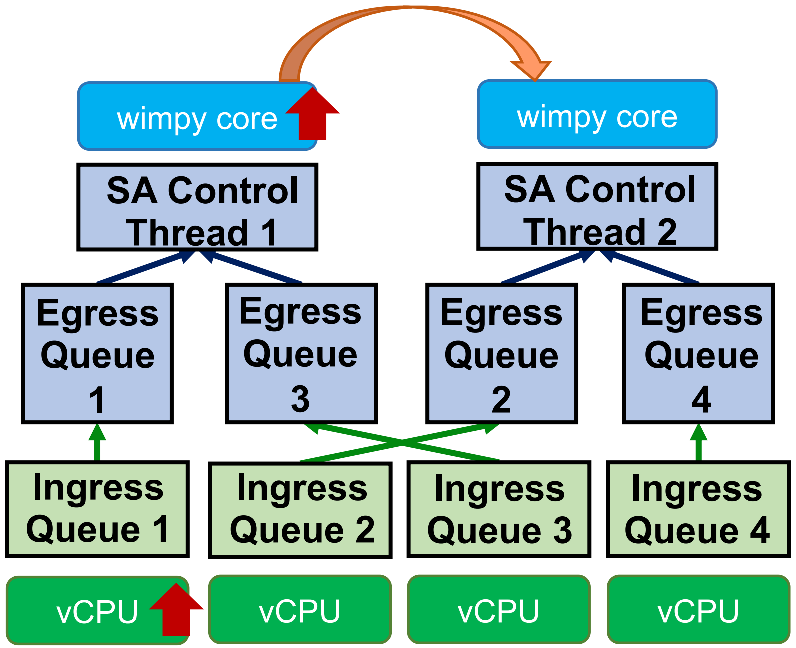
FPGA-assisted inter-thread load-balancing

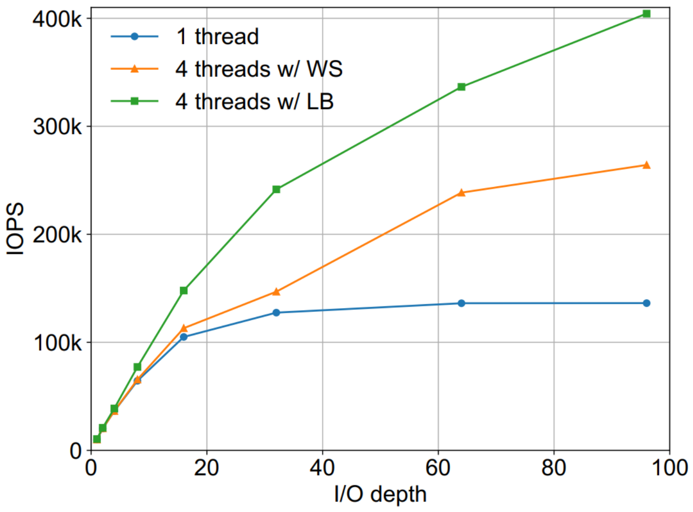
1-to-1 mapping

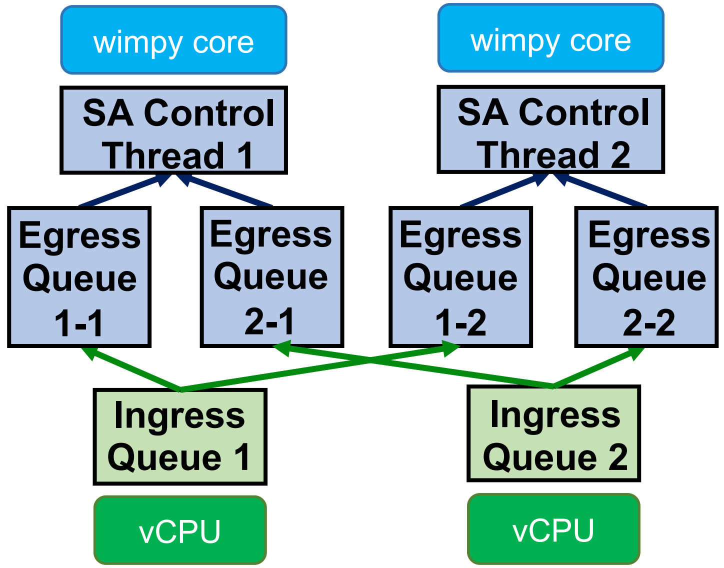
- Causes load imbalance when a vCPU starts bursting
- **Work stealing** can alleviate load imbalance
- **~35%** throughput loss

1-to-N mapping

- Always load balanced
- At line rate with no throughput loss
- Consumes more FPGA resources



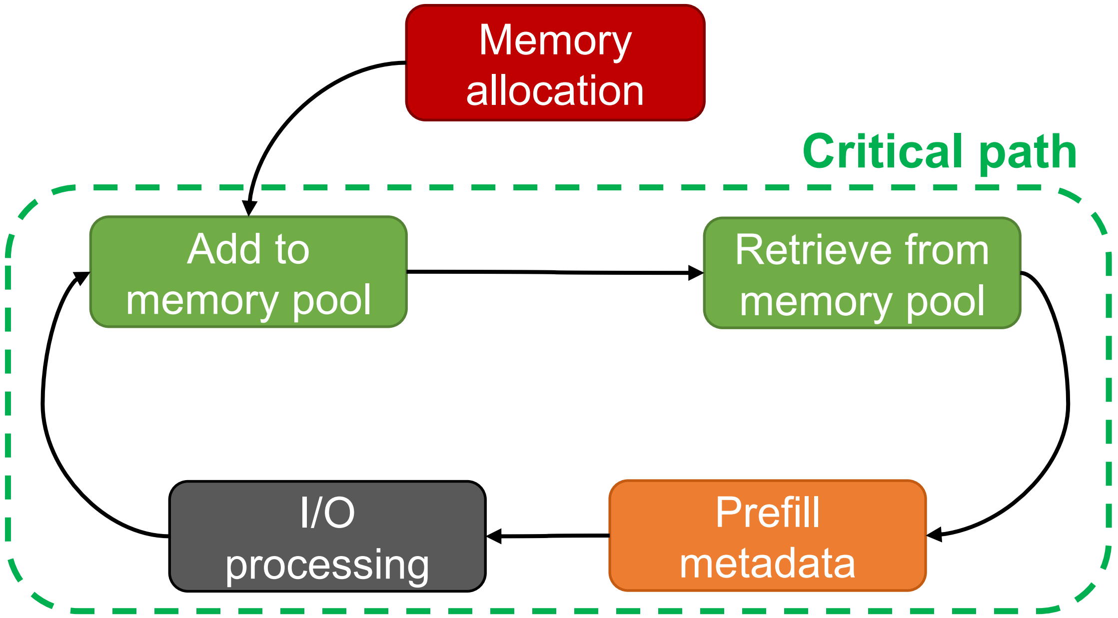




Efficient tiered memory pooling

I/O memory buffer lifecycle

- Main goal is to avoid slow & heavy operations on the critical path

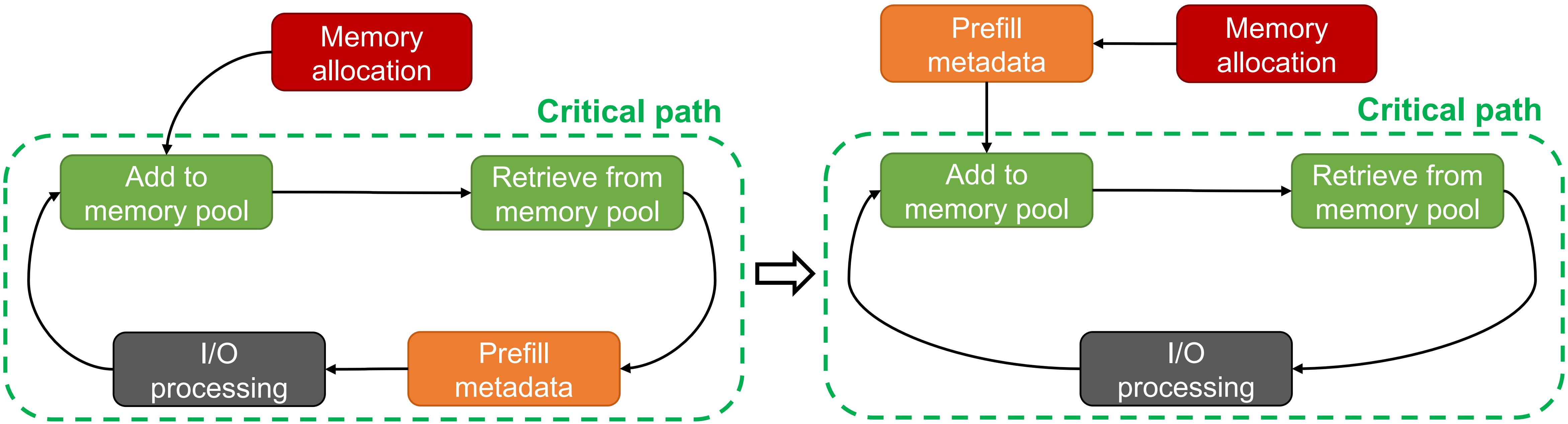


Global memory pool

Efficient tiered memory pooling

I/O memory buffer lifecycle

- Main goal is to avoid slow & heavy operations on the critical path
- VM/VD/IOQ metadata prefilling becomes non-trivial for supporting 1m+ IOPS



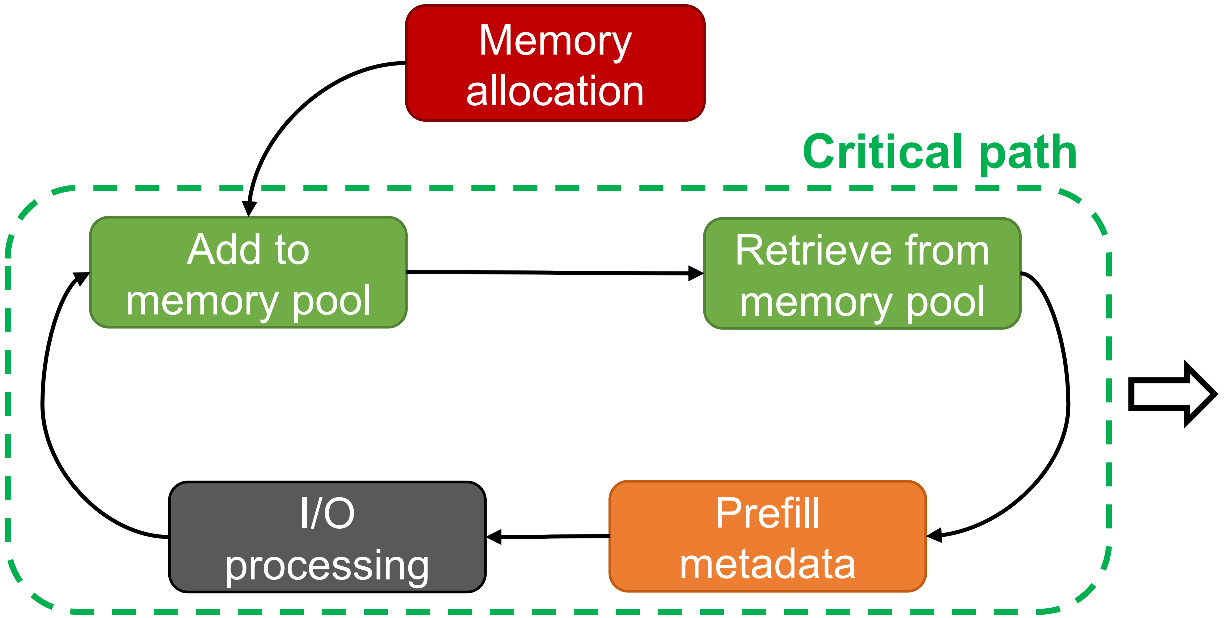
Global memory pool

I/O Queue dedicated memory pool

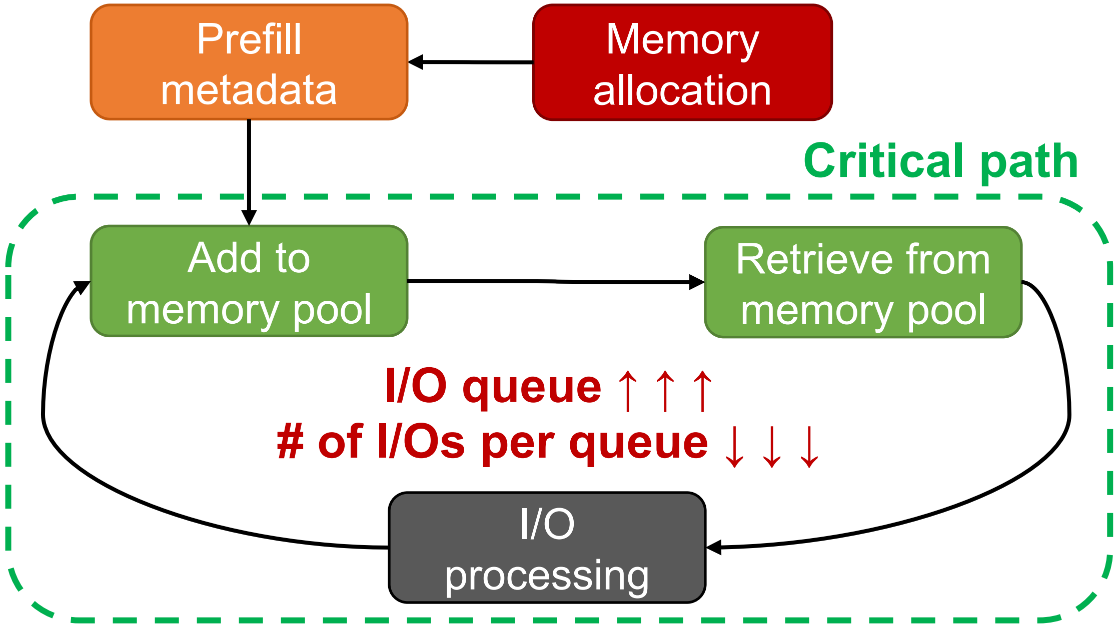
Efficient tiered memory pooling

I/O memory buffer lifecycle

- Main goal is to avoid slow & heavy operations on the critical path
- VM/VD/IOQ metadata prefilling becomes non-trivial for supporting 1m+ IOPS



Global memory pool

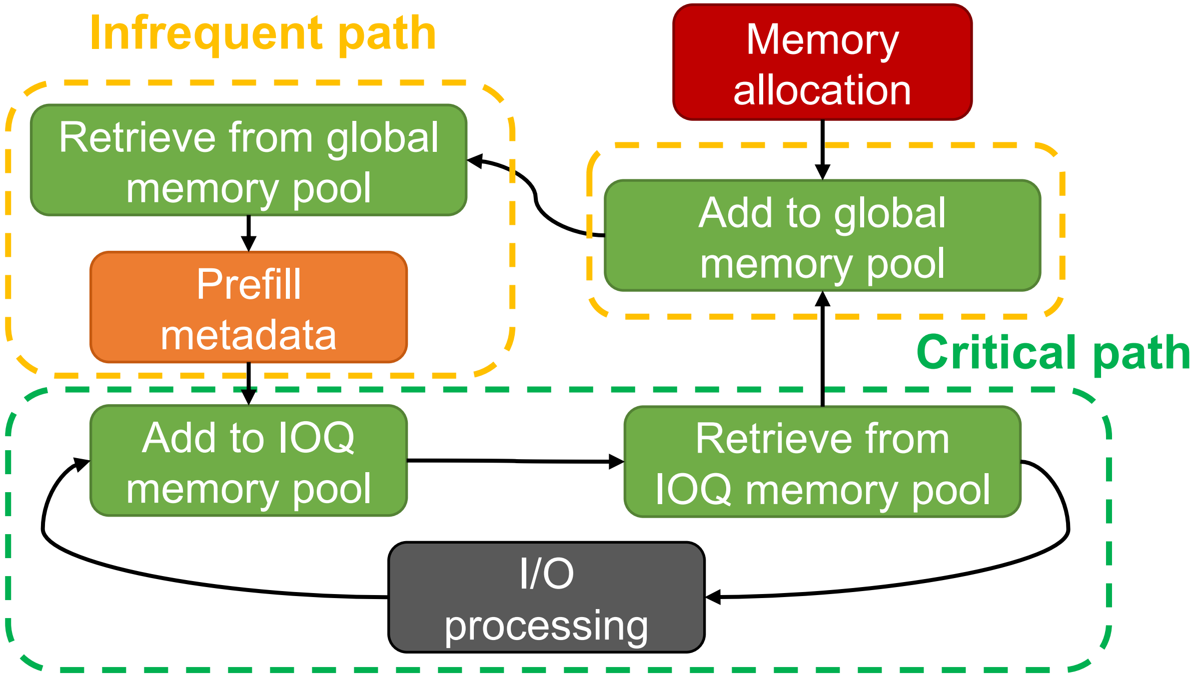


I/O Queue dedicated memory pool

Efficient tiered memory pooling

I/O memory buffer lifecycle

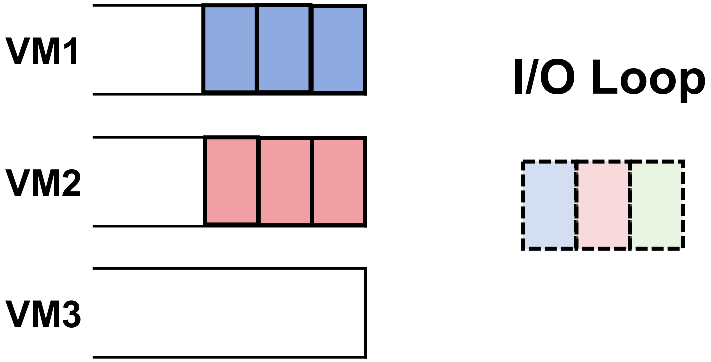
- Main goal is to avoid slow & heavy operations on the critical path
- VM/VD/IOQ metadata prefilling becomes non-trivial for supporting 1m+ IOPS
- Two-tier memory pool to avoid prefilling while supporting more I/Os



Existing systems for intra-thread scheduling

BaseCBS

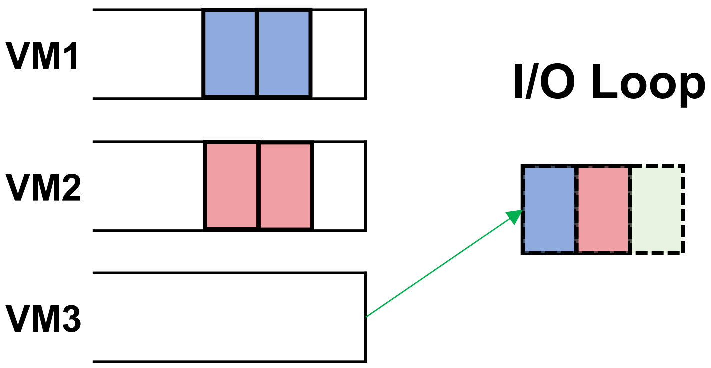
- Isolated performance
- Does not support burst



Existing systems for intra-thread scheduling

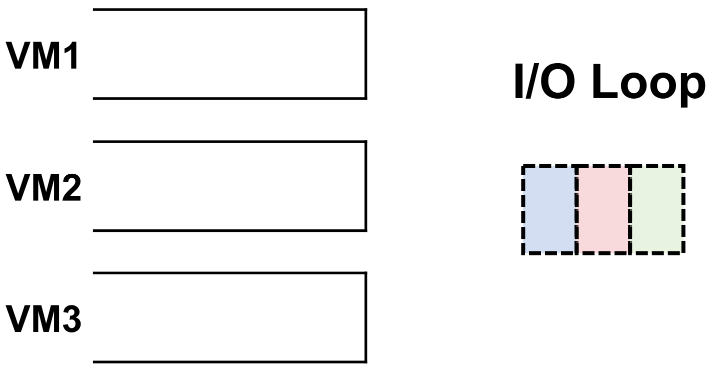
BaseCBS

- Isolated performance
- Does not support burst

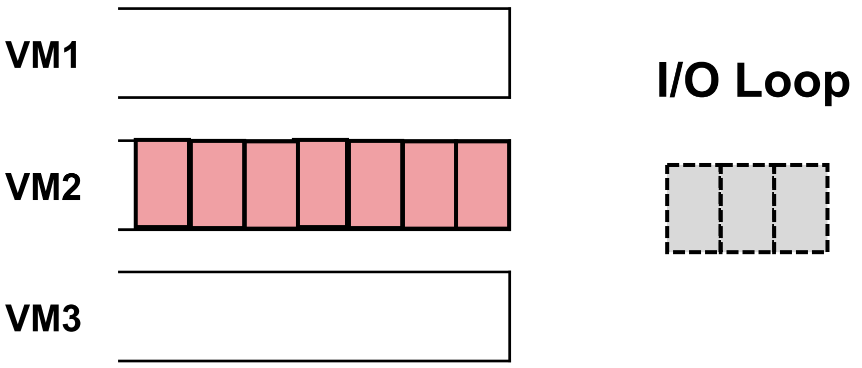


Existing systems for intra-thread scheduling

BaseCBS
- Isolated performance
- Does not support burst



WildCBS
- Support maximum burst
- Latency increase on base-level tenants



Assume I/O queue of VM2 is always full

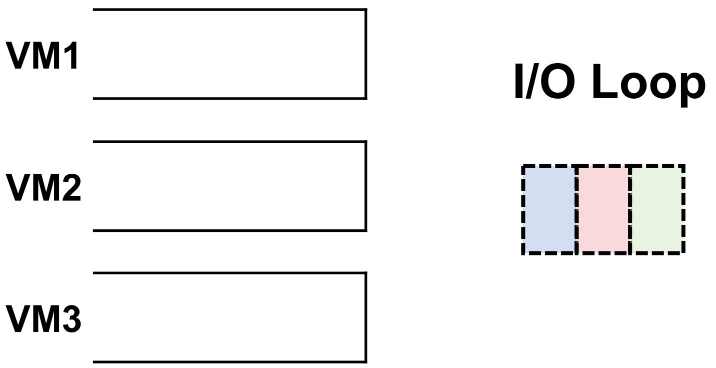
Existing systems for intra-thread scheduling

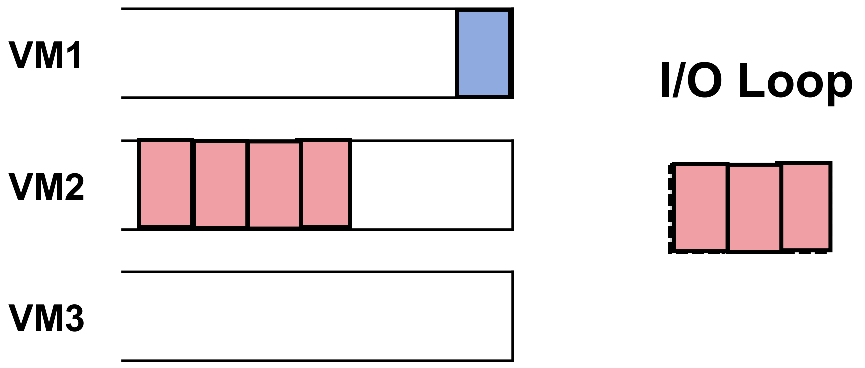
BaseCBS

- Isolated performance
- Does not support burst

WildCBS

- Support maximum burst
- Latency increase on base-level tenants



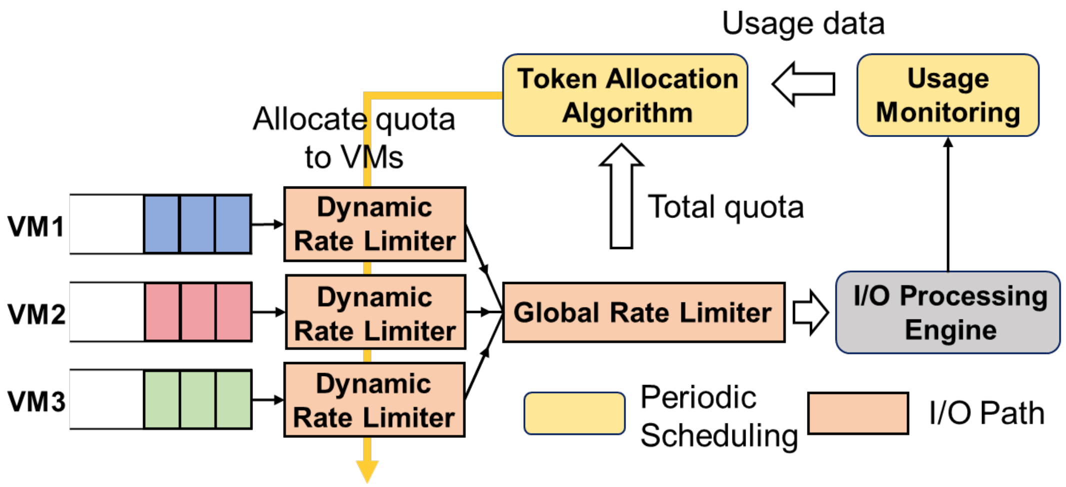


Assume I/O queue of VM2 is always full

Burstable I/O scheduler

Supports burst while protecting base-level tenants

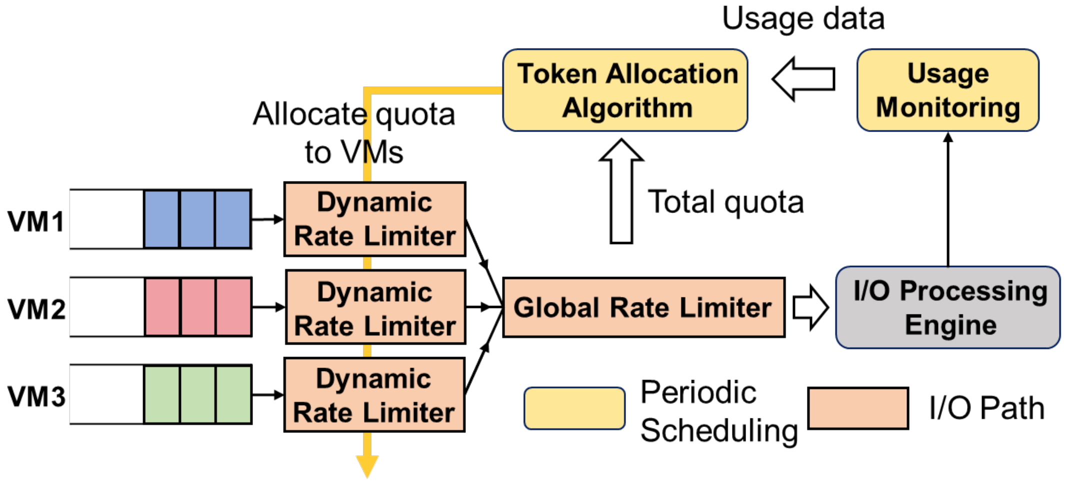
- Predicts usage of next window based on statistics of last N windows
- Leverages unused provisioned resources for bursting
- Provide fallback mechanisms for base-level tenants protection

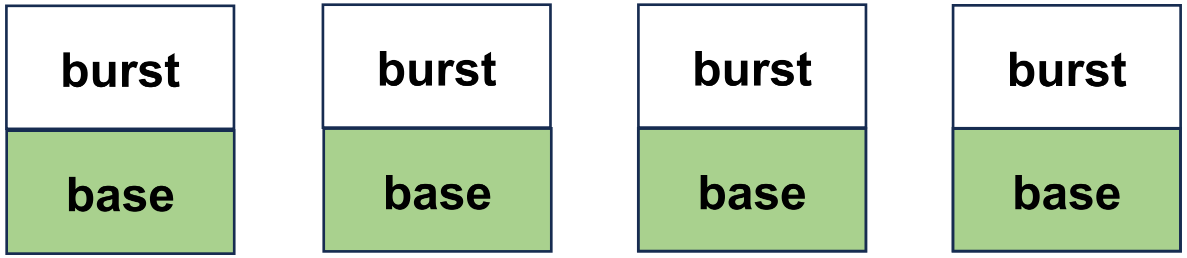


Burstable I/O scheduler

Supports burst while protecting base-level tenants

- Predicts usage of next window based on statistics of last N windows
- Leverages unused provisioned resources for bursting
- Provide fallback mechanisms for base-level tenants protection

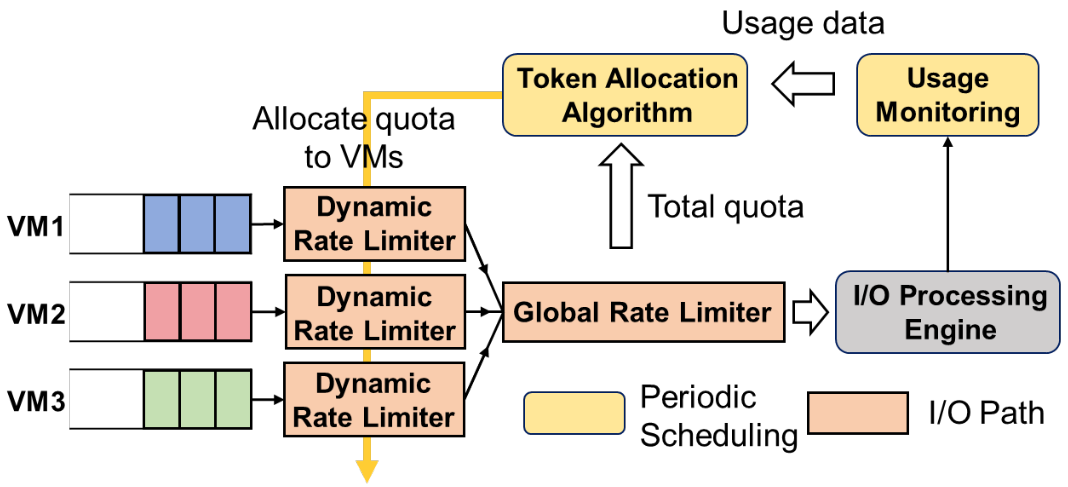


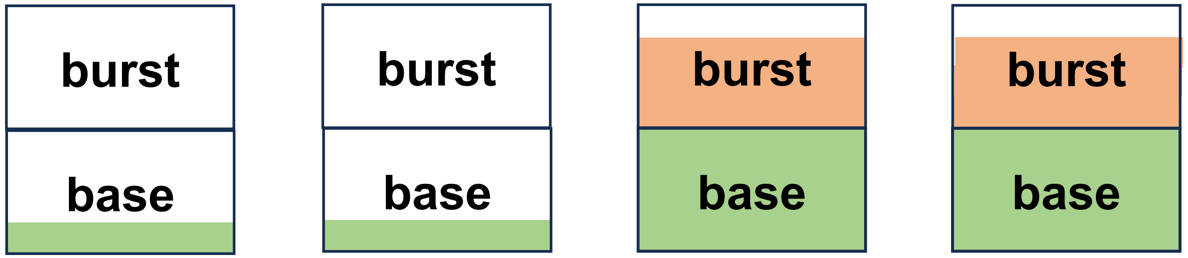


Burstable I/O scheduler

Supports burst while protecting base-level tenants

- Predicts usage of next window based on statistics of last N windows
- Leverages unused provisioned resources for bursting
- Provide fallback mechanisms for base-level tenants protection

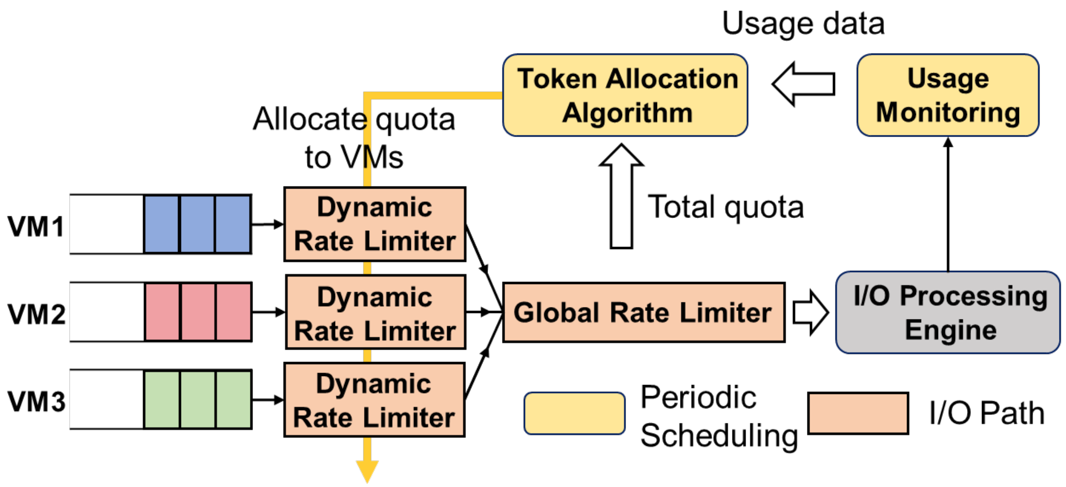


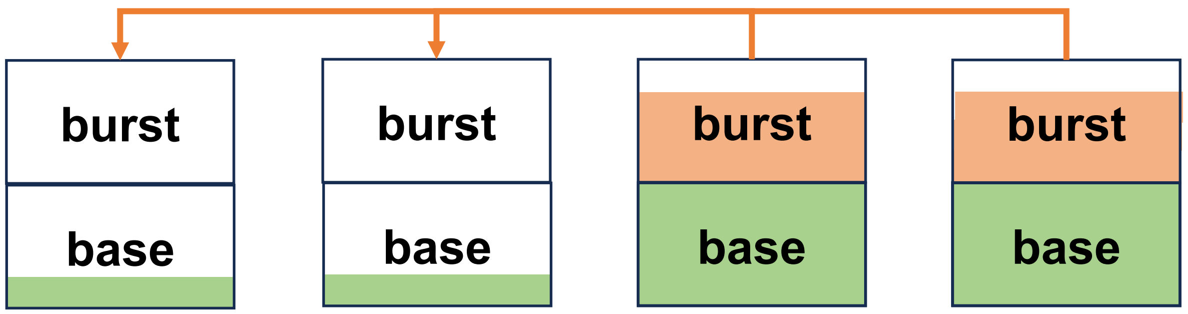


Burstable I/O scheduler

Supports burst while protecting base-level tenants

- Predicts usage of next window based on statistics of last N windows
- Leverages unused provisioned resources for bursting
- Provide fallback mechanisms for base-level tenants protection

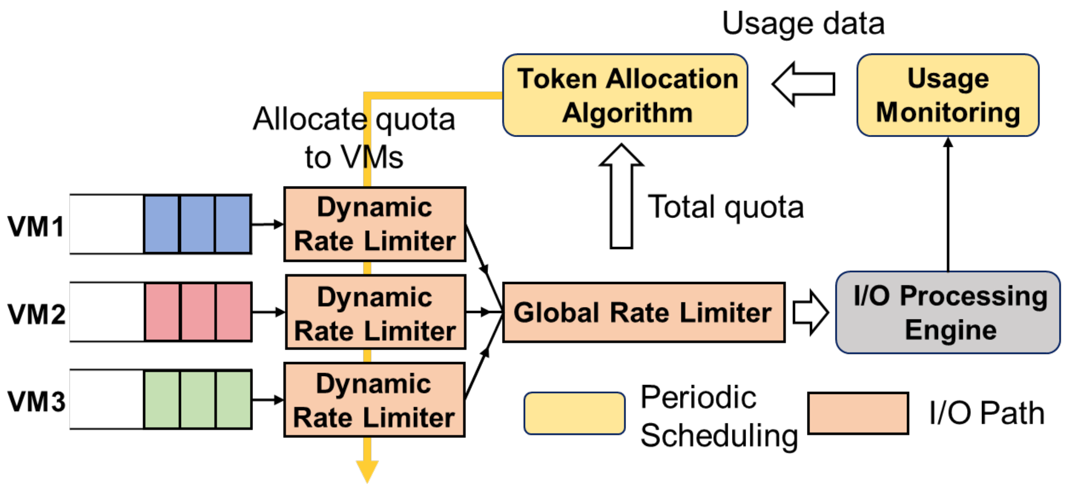


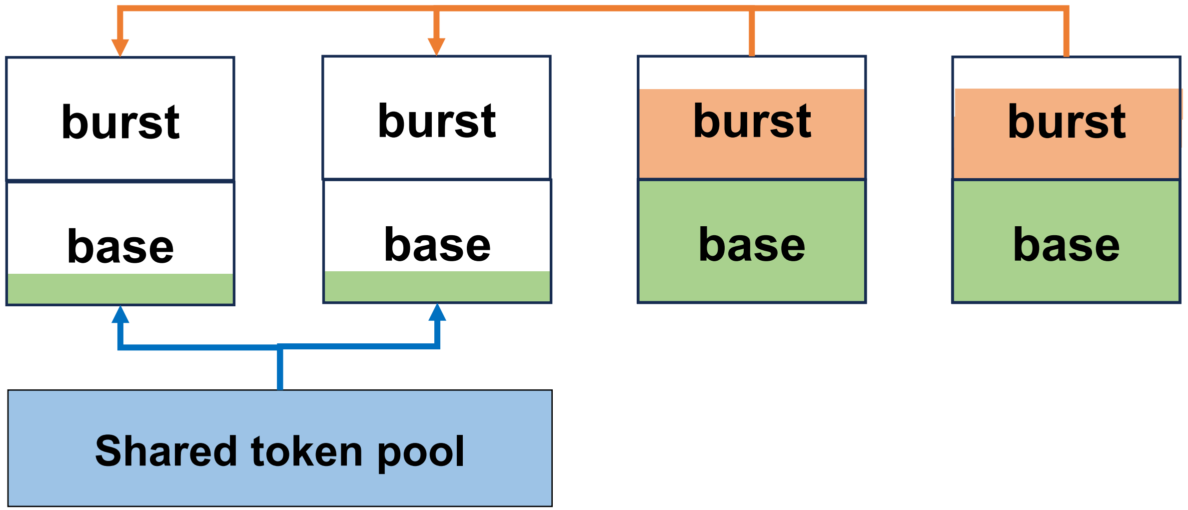


Burstable I/O scheduler

Supports burst while protecting base-level tenants

- Predicts usage of next window based on statistics of last N windows
- Leverages unused provisioned resources for bursting
- Provide fallback mechanisms for base-level tenants protection





Vectorized cost estimator

SSD cost estimation

- SSD as a black-box
- Scalar cost with linear estimation: ReFlex [ASPLOS ’17], IOCost [ASPLOS ’22]

Heterogeneity in consumed resources on xDPU

- Small I/Os are bottlenecked on CPU time
- Large I/Os are bottlenecked on NIC bandwidth

I/O type	CPU time	Data egress	Admittable # of I/Os per ms (100Gb NIC)	Admittable # of I/Os per ms (CPU)
4KB write	1.16us	4KB	3276	**862**
128KB write	6.18us	128KB	**102**	161

Vectorized cost estimator

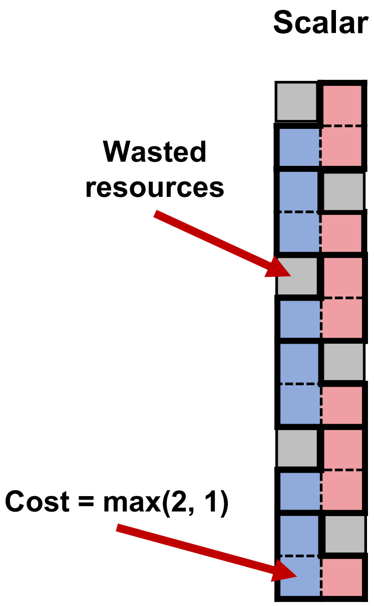
SSD cost estimation

- SSD as a black-box
- Scalar cost with linear estimation: ReFlex [ASPLOS ’17], IOCost [ASPLOS ’22]

Heterogeneity in consumed resources on xDPU

- Small I/Os are bottlenecked on CPU time
- Large I/Os are bottlenecked on NIC bandwidth

I/O type	CPU time	Data egress	Admittable # of I/Os per ms (100Gb NIC)	Admittable # of I/Os per ms (CPU)
4KB write	1.16us	4KB	3276	**862**
128KB write	6.18us	128KB	**102**	161



Vectorized cost estimator

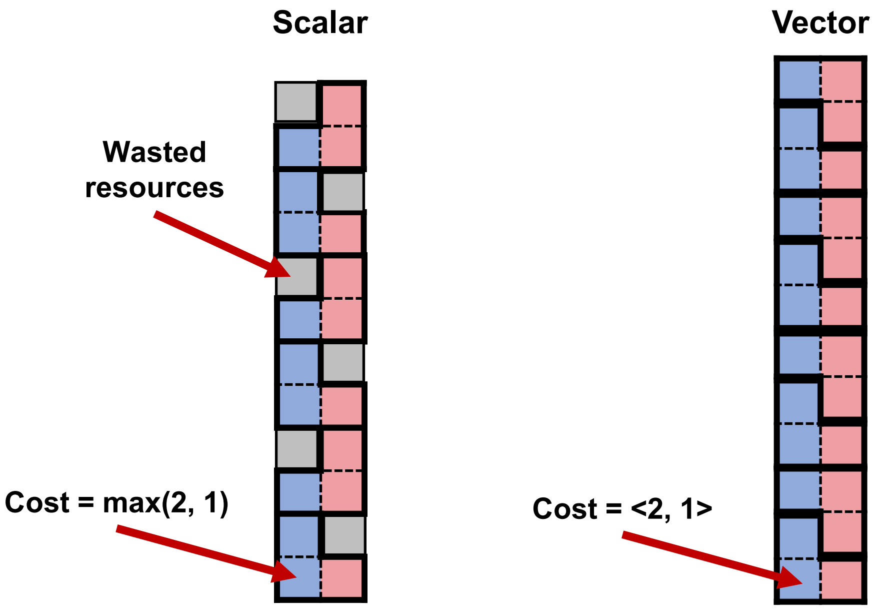
SSD cost estimation

- SSD as a black-box
- Scalar cost with linear estimation: ReFlex [ASPLOS ’17], IOCost [ASPLOS ’22]

Heterogeneity in consumed resources on xDPU

- Small I/Os are bottlenecked on CPU time
- Large I/Os are bottlenecked on NIC bandwidth

I/O type	CPU time	Data egress	Admittable # of I/Os per ms (100Gb NIC)	Admittable # of I/Os per ms (CPU)
4KB write	1.16us	4KB	3276	**862**
128KB write	6.18us	128KB	**102**	161



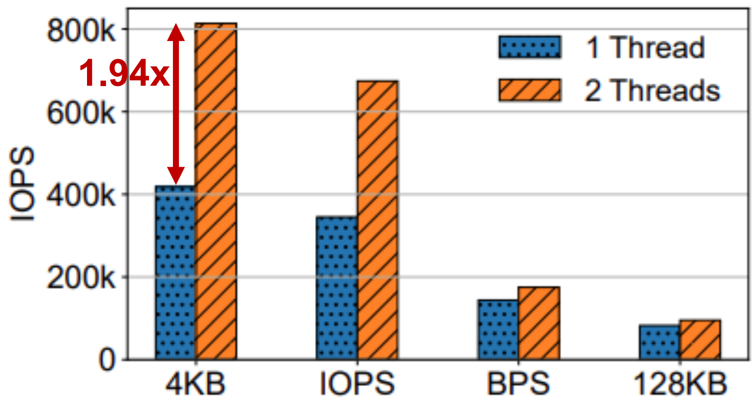
Evaluation

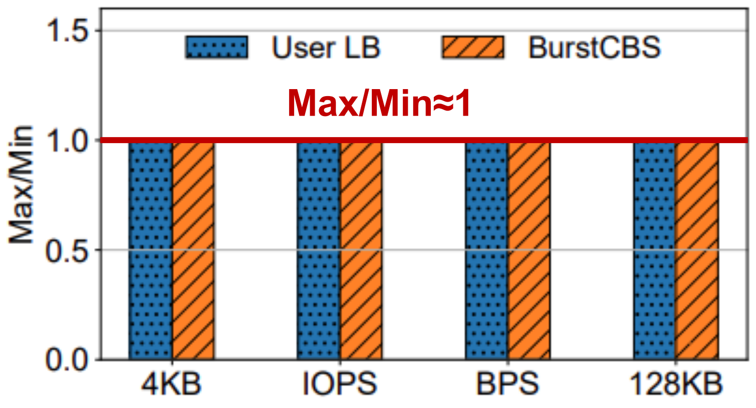
- Setup
  - One compute node equipped with the newest version of xDPU
- Baselines
  - BaseCBS & WildCBS
- Workload
  - BPS-intensive: 4-128KB, which are the most common I/O sizes
  - IOPS-intensive: 4-16KB, which resembles many transactional DBs

Is load balanced among multiple threads?

Linear scaling & near-perfect load balancing

- Linear throughput scaling when adding more control threads
- Load is balanced among all the threads

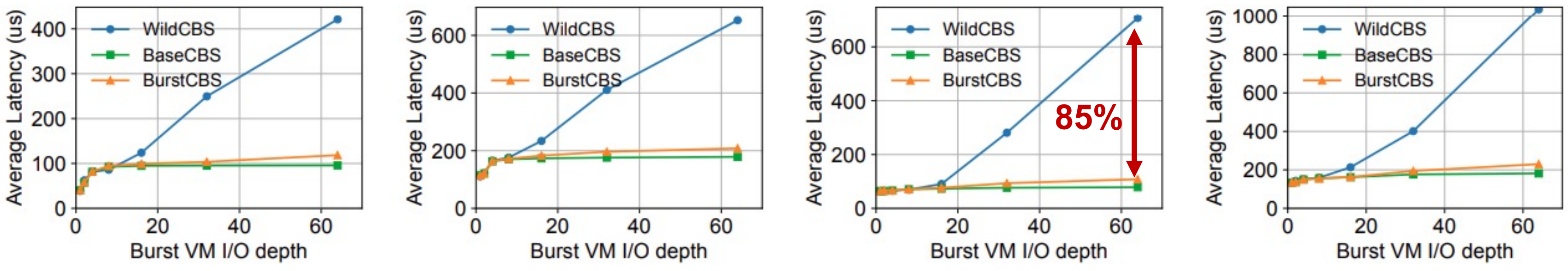




How does a burst tenant impact its neighbors?

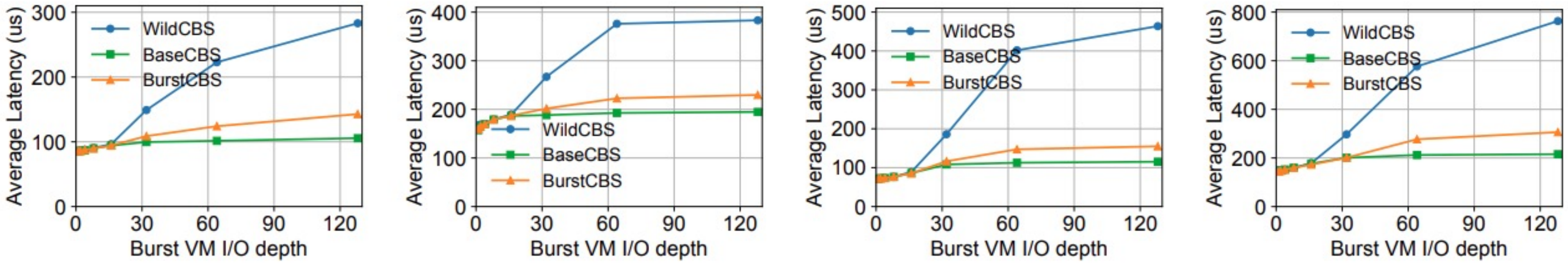
Achieves near-ideal performance isolation

- Up to **85%** latency reduction



(a) ProductA write (b) ProductA read (c) ProductB write (d) ProductB read

IOPS-intensive workload



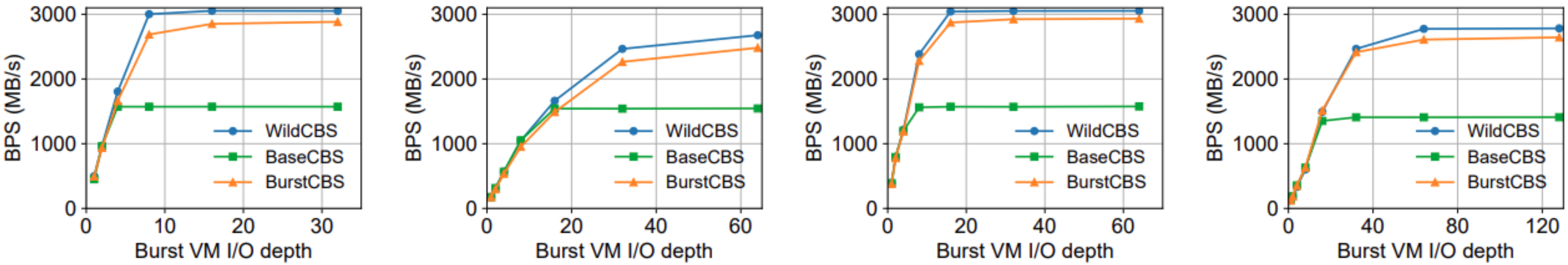
(a) ProductA write (b) ProductA read (c) ProductB write (d) ProductB read

BPS-intensive workload

What is the maximum burst capability?

Supports similar level of burst to WildCBS

- Only **5%-8%** throughput loss due to global shared resource pool

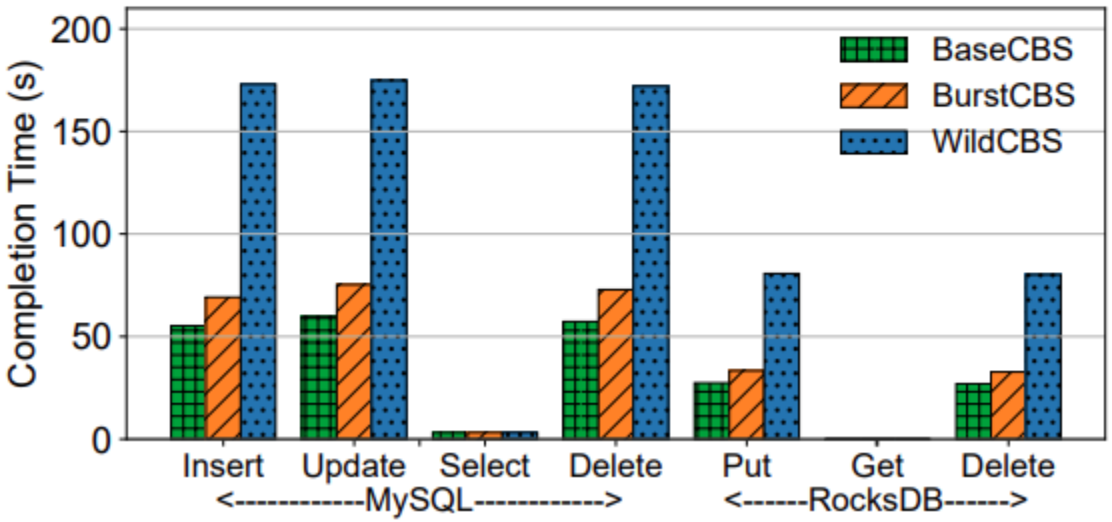


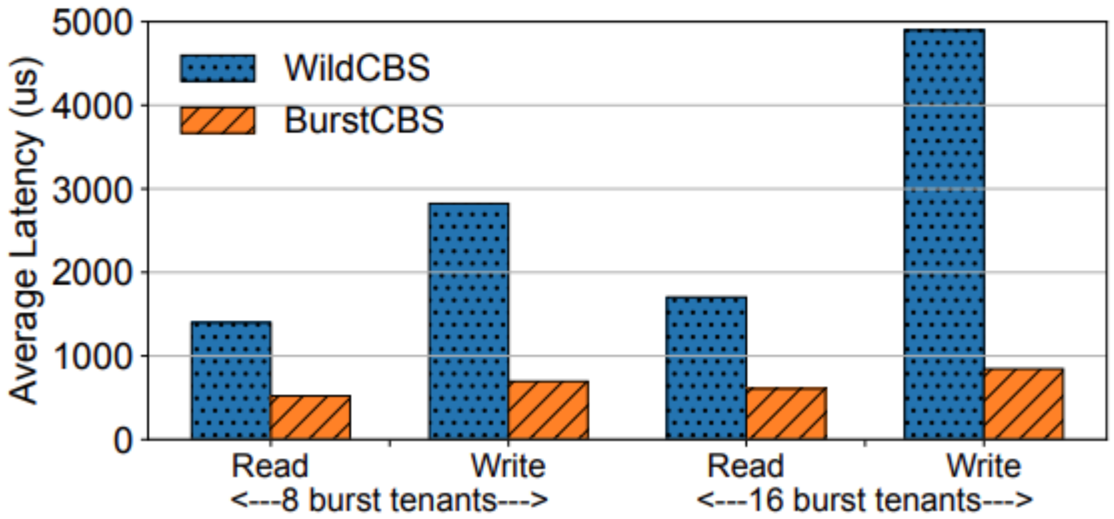
(a) BPS-intensive, ProductA (b) IOPS-intensive, ProductA (c) BPS-intensive, ProductA+B (d) IOPS-intensive, ProductA+B

Application performance improvement

Effectively reduces latency of transactional databases

- **~60%** latency reduction on MySQL and RocksDB write operations
- Up to **83%** latency reduction on our internal relational database service

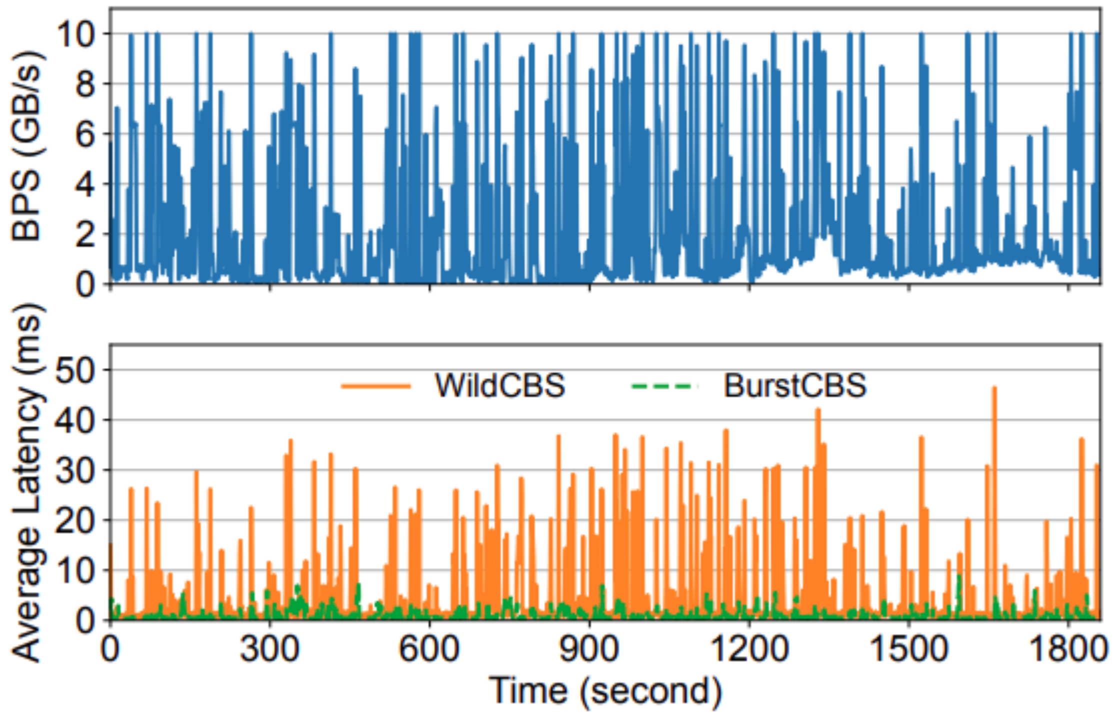




Application performance improvement (cont.)

Benefits for our internal relational database service

- Average query latency: up to 47ms -> **less than 10ms**



More details in our paper

- Handling of I/O cost mis-estimation
- Scheduler scalability
- Responsiveness to sudden tenant activation
- …

Conclusion

- BurstCBS: an I/O scheduling system that supports burst and keeps performance interference limited
  - High performance queue scaling for efficient load balancing among threads
  - Burstable I/O scheduler and vectorized I/O cost estimator for intra-thread scheduling
- BurstCBS provides up to 85% average latency reduction during bursts

北京大學 PEKING UNIVERSITY

Alibaba Cloud

shujunyi@pku.edu.cn

Thanks!