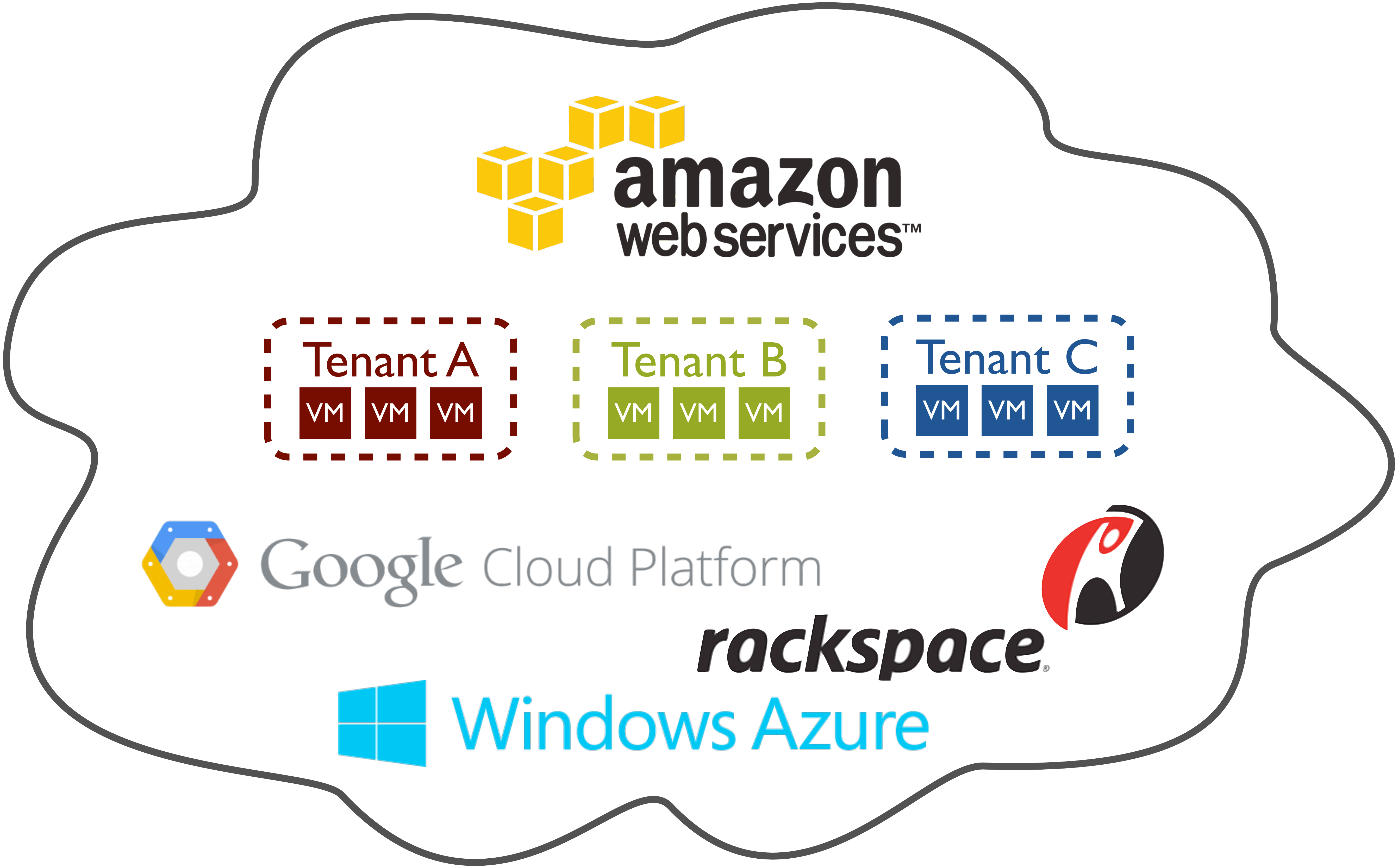
PrincetonUniversity

From application requests to Virtual IOPs: Provisioned key-value storage with Libra

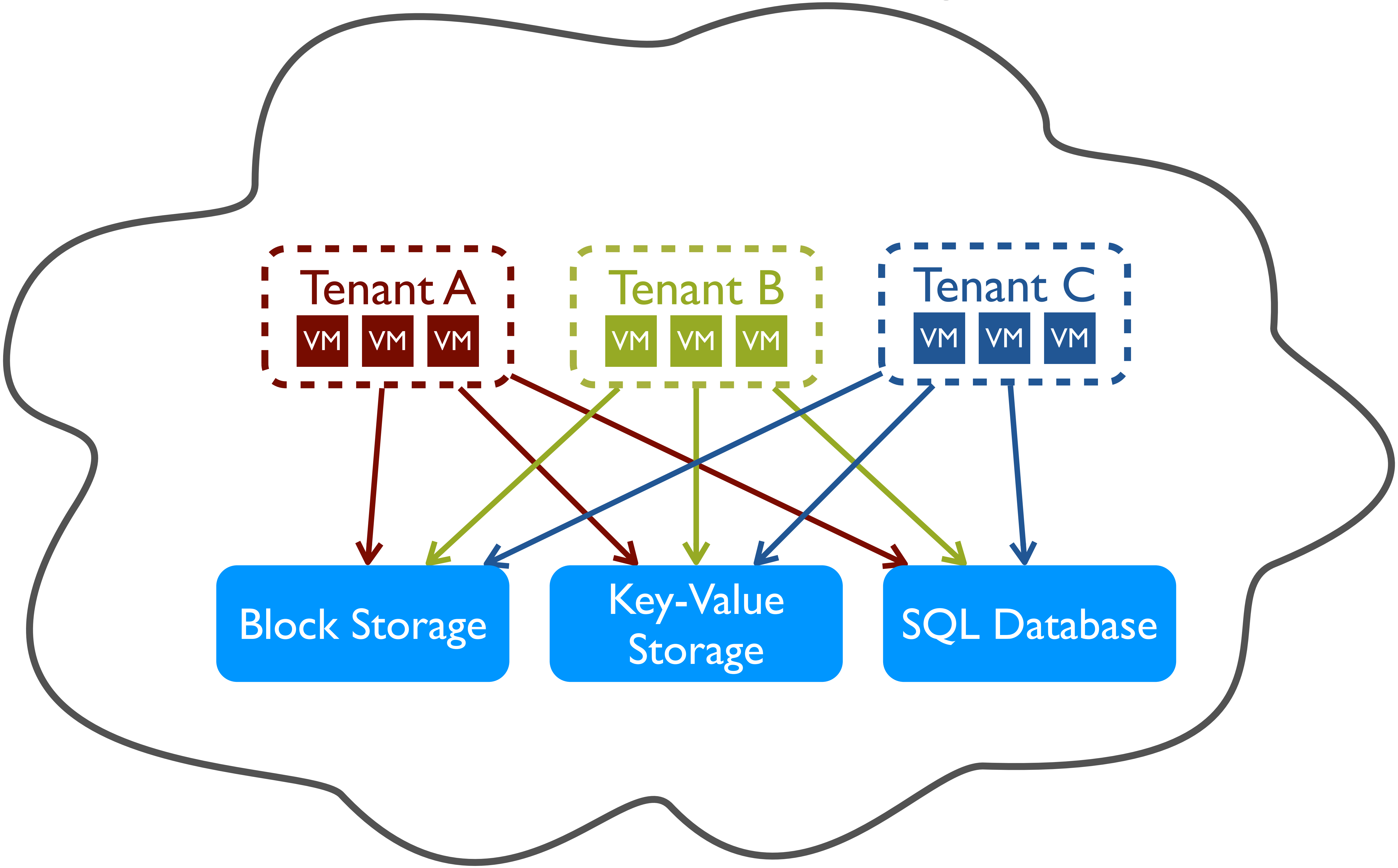
David Shue* and Michael J. Freedman

(*now at Google)

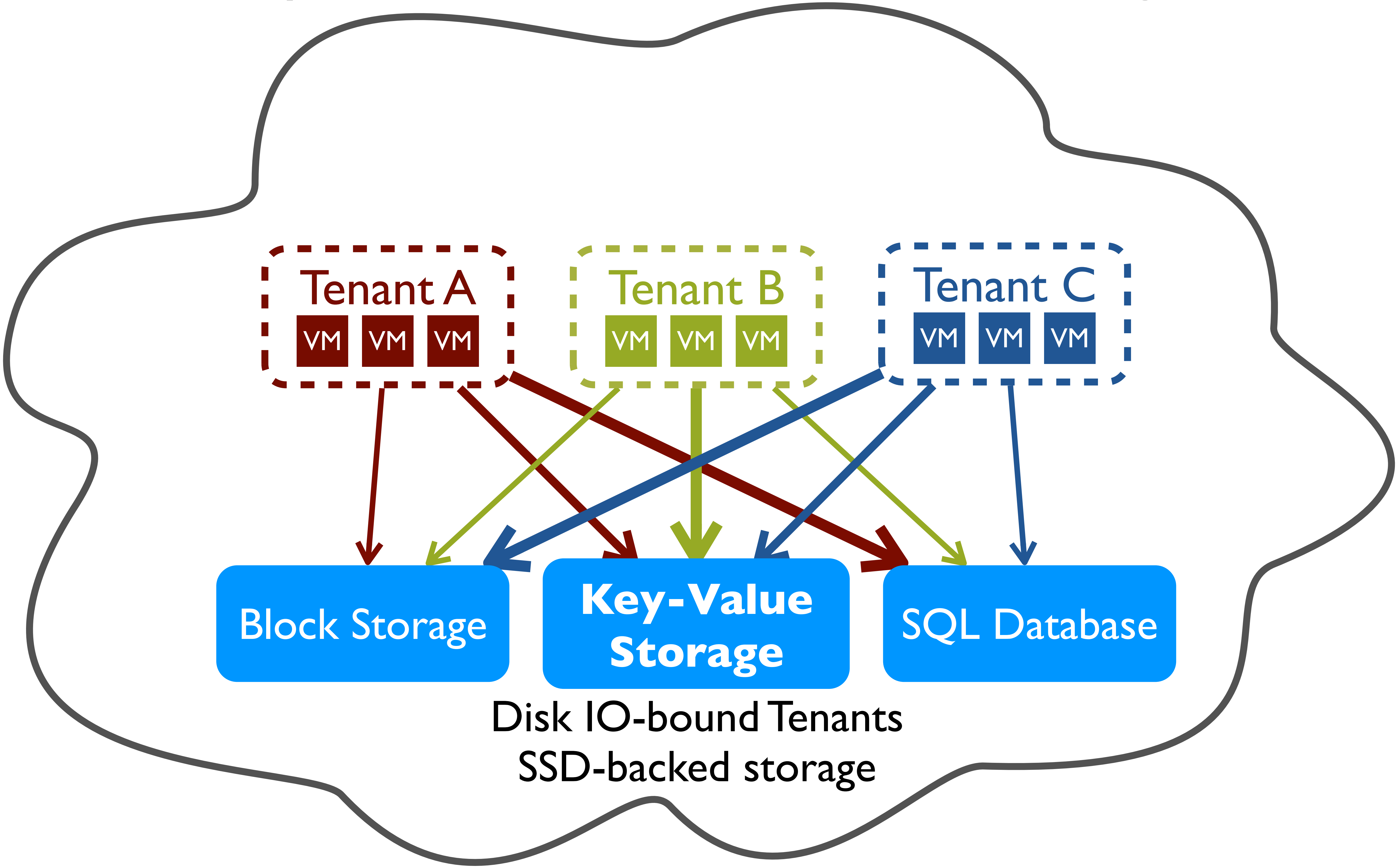
Shared Cloud



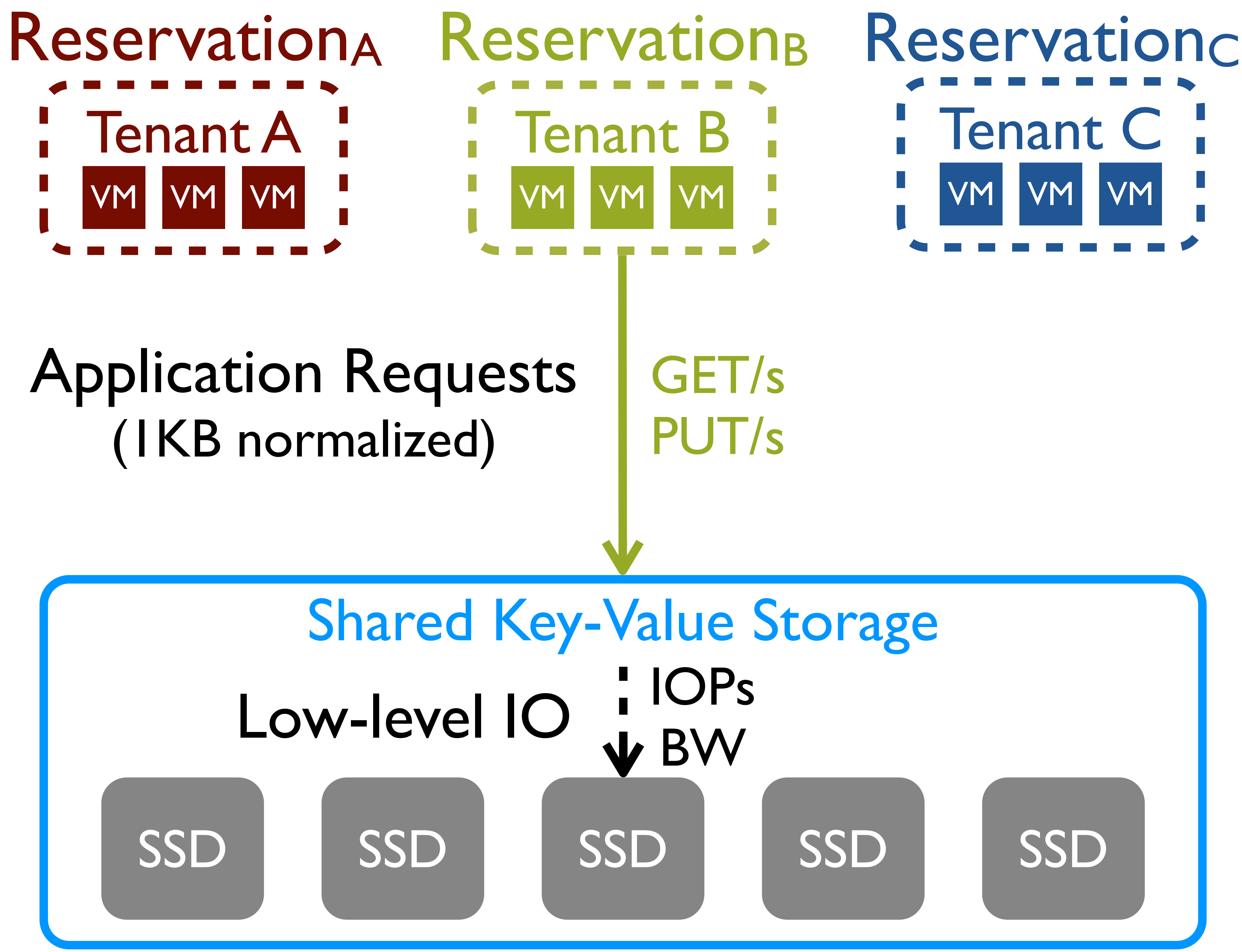
Shared Cloud Storage



Unpredictable Shared Cloud Storage



Provisioned Shared Key-Value Storage



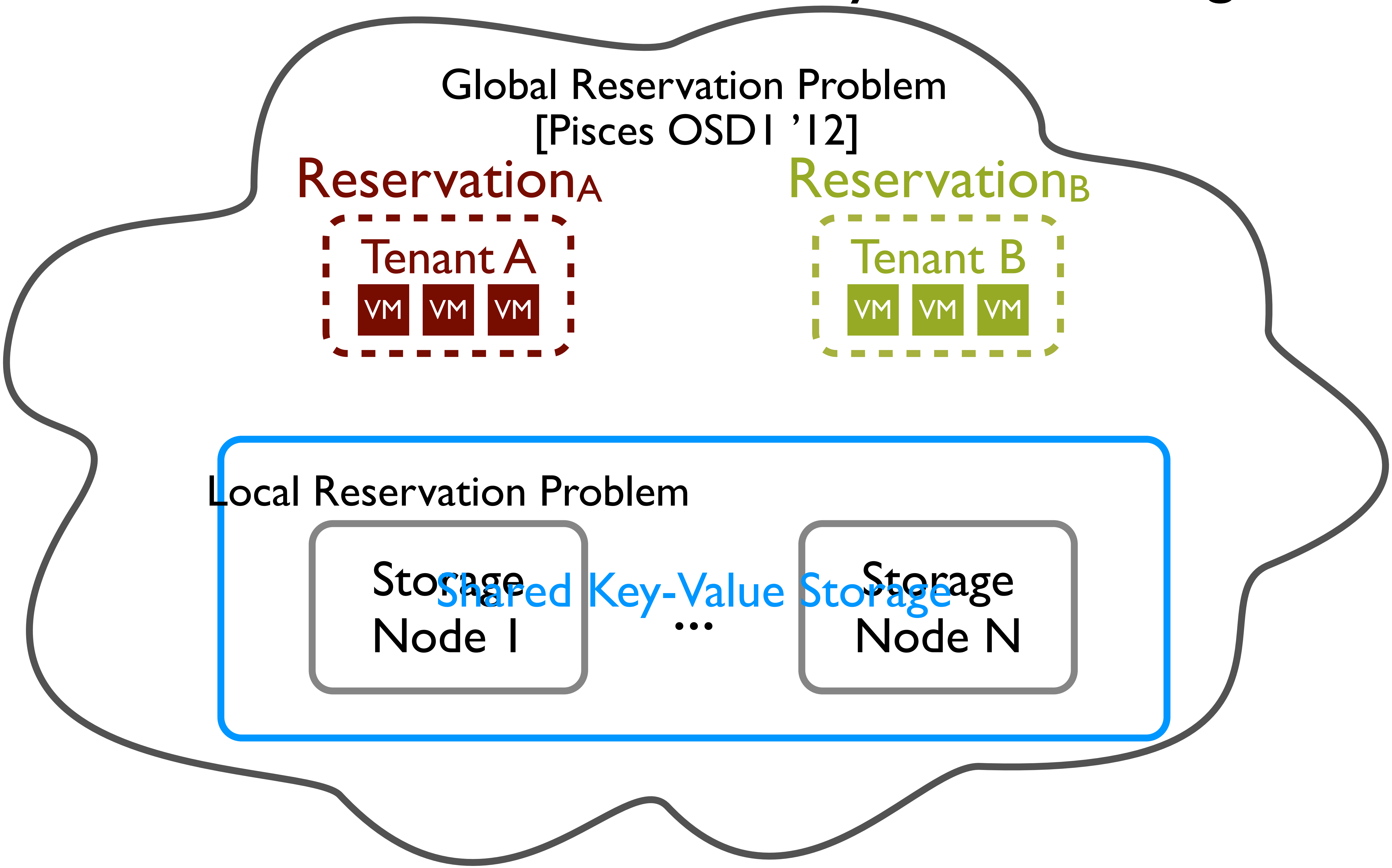
Libra Contributions

- Libra IO Scheduler
  - Provisions *low-level IO allocations* for *app-request reservations* w/ high utilization.
  - Supports arbitrary object distributions and workloads.
- 2 key mechanisms
  - Track per-tenant app-request resource profiles.
  - Model IO resources with Virtual IOPs.

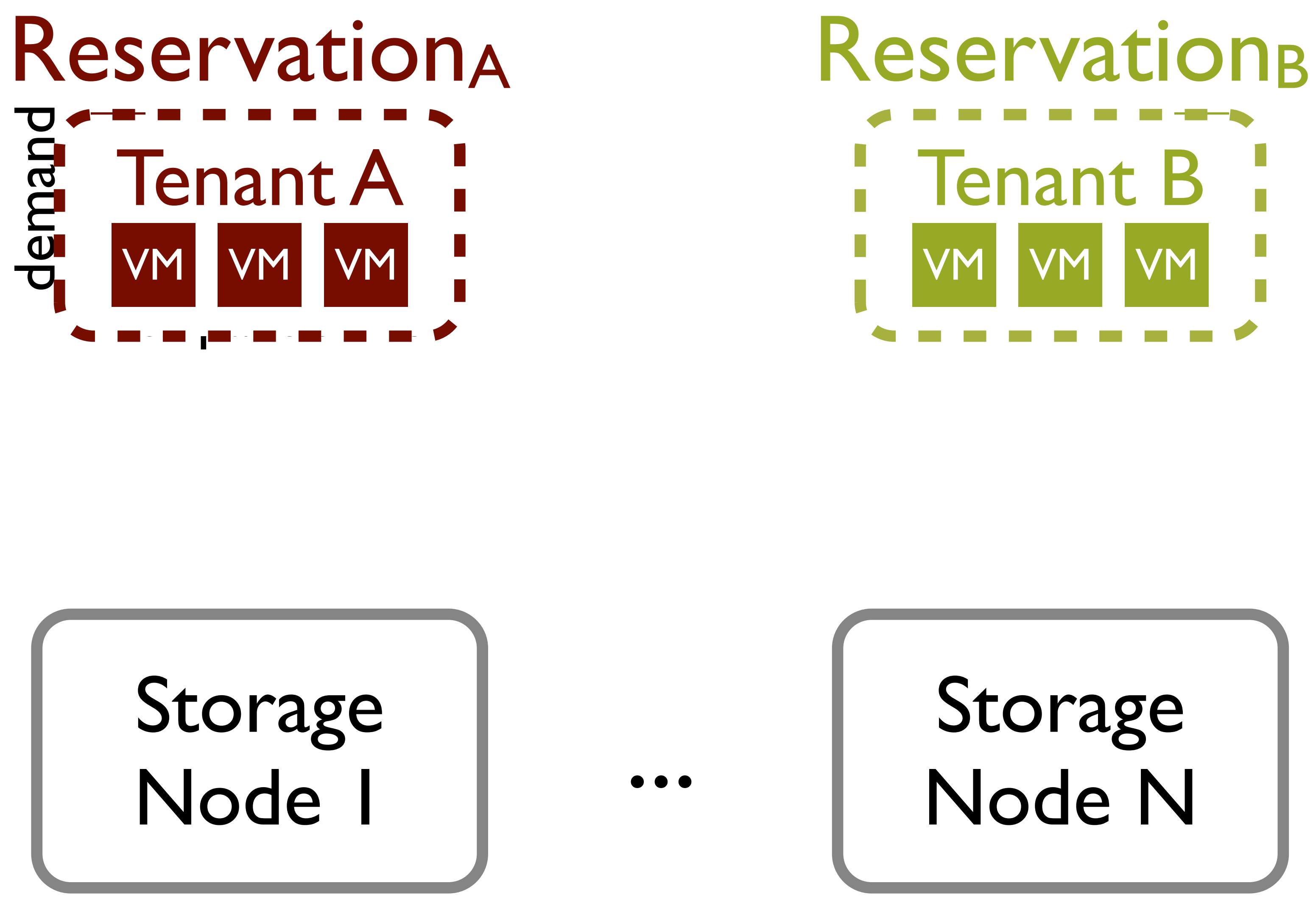
Related Work

	Storage Type	App-requests	Work Conserving	Media
Maestro	Block	N	N	HDD
mClock	Block	N	Y	HDD
FlashFQ	Block	N	Y	SSD
DynamoDB	Key-Value	Y	N	SSD

Provisioned Distributed Key-Value Storage

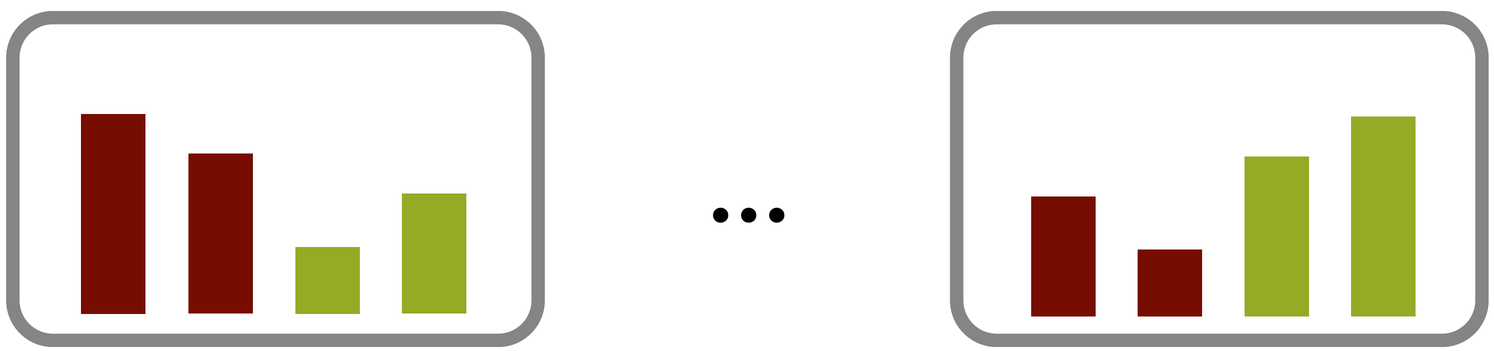


Provisioned Distributed Key-Value Storage

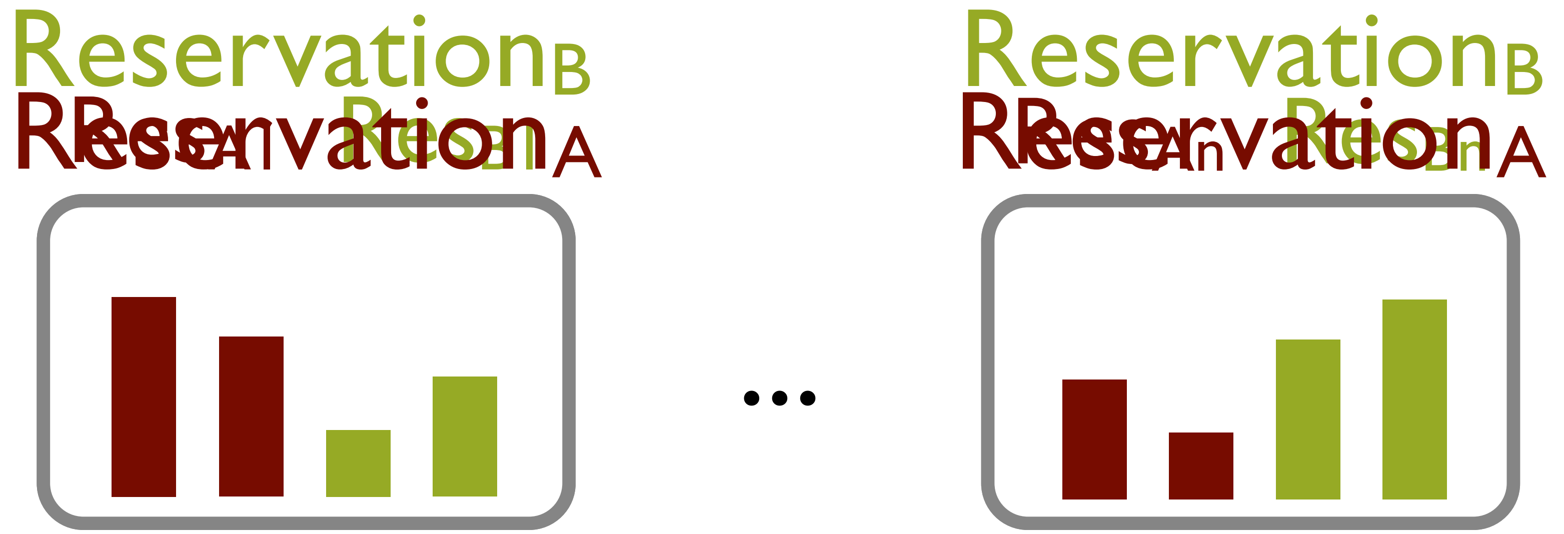


Provisioned Distributed Key-Value Storage

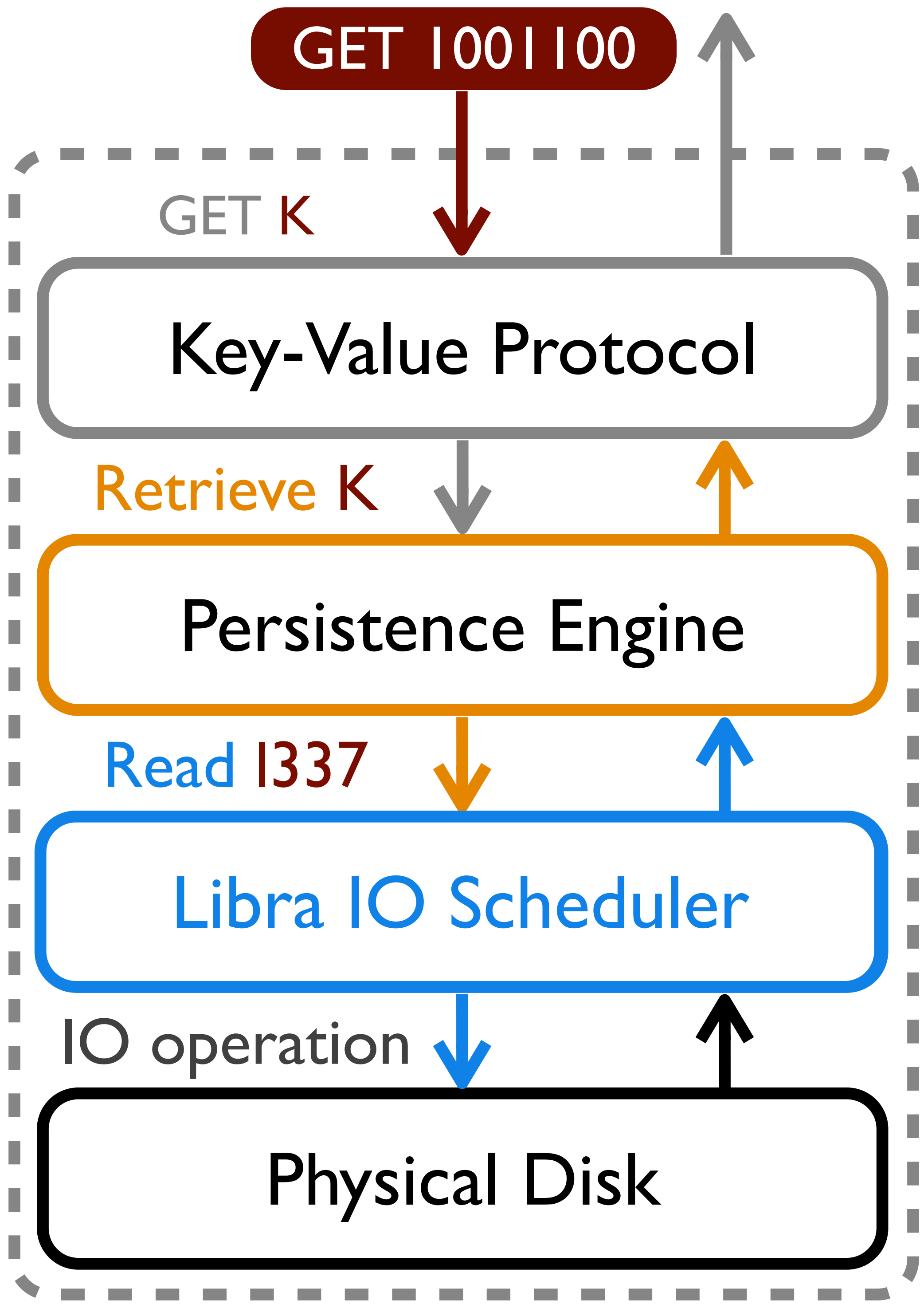
Reservation_A Reservation_B



Provisioned Distributed Key-Value Storage

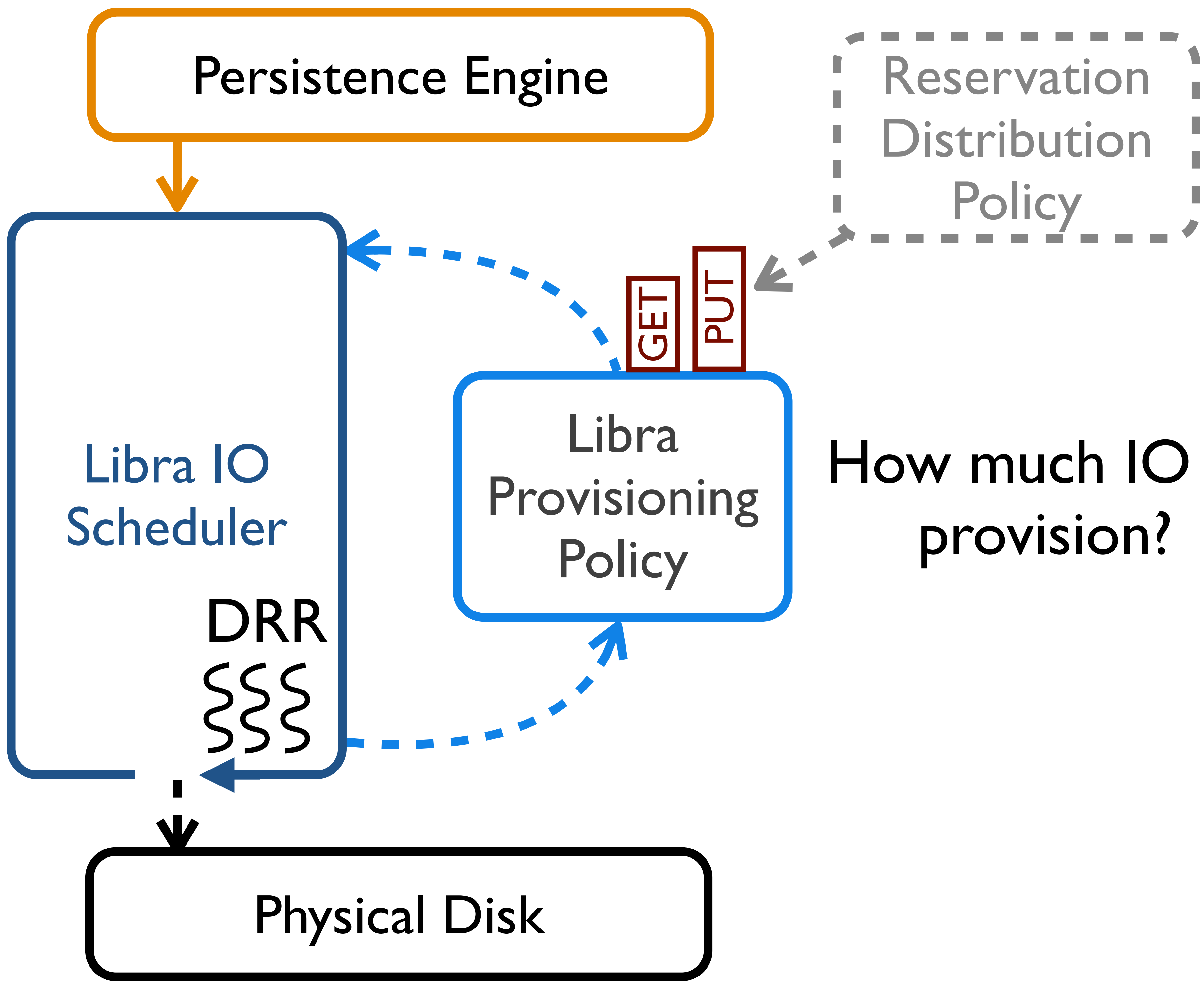


Provisioned Local Key-Value Storage



Libra Design

How much IO to consume?



How much IO to provision?

Provisioning App-request Reservations is Hard

IO Amplification

1 KB PUT ≥ 1 KB Write

Track tenant app-request resource profiles

IO Interference

Variable IO throughput

Underestimate provisionable IO

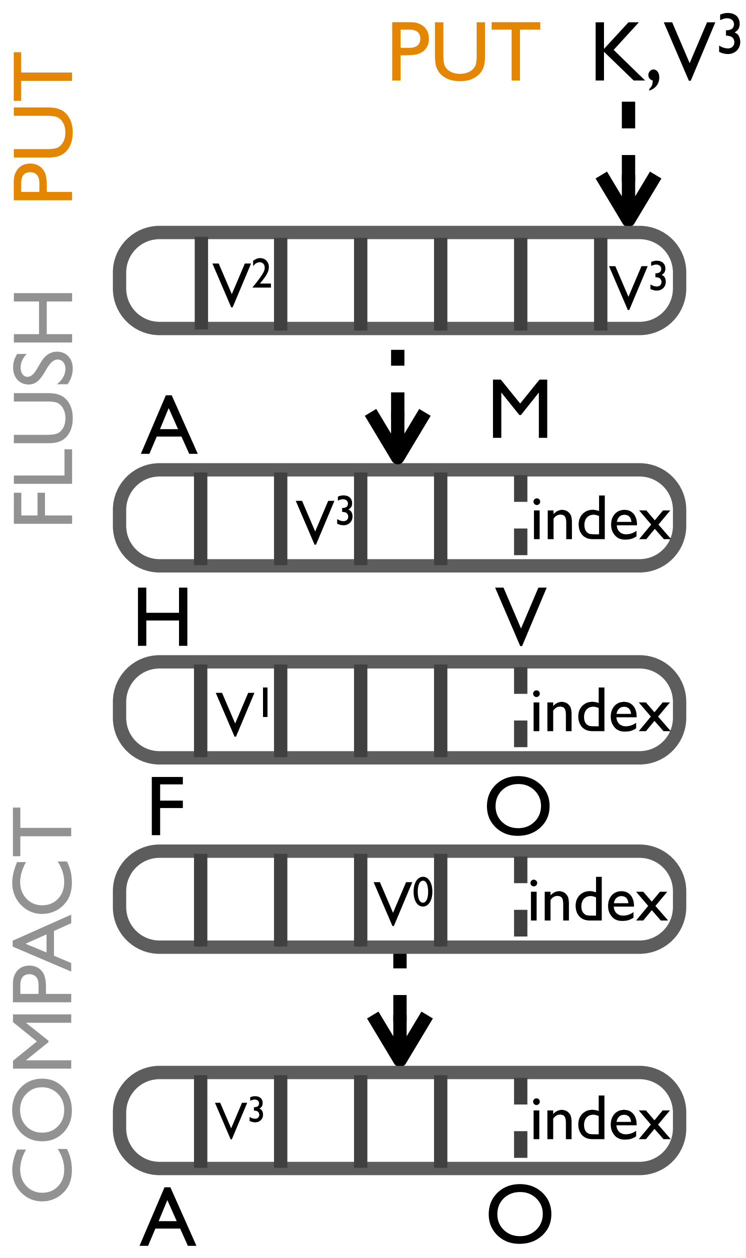
Non-linear IO Performance

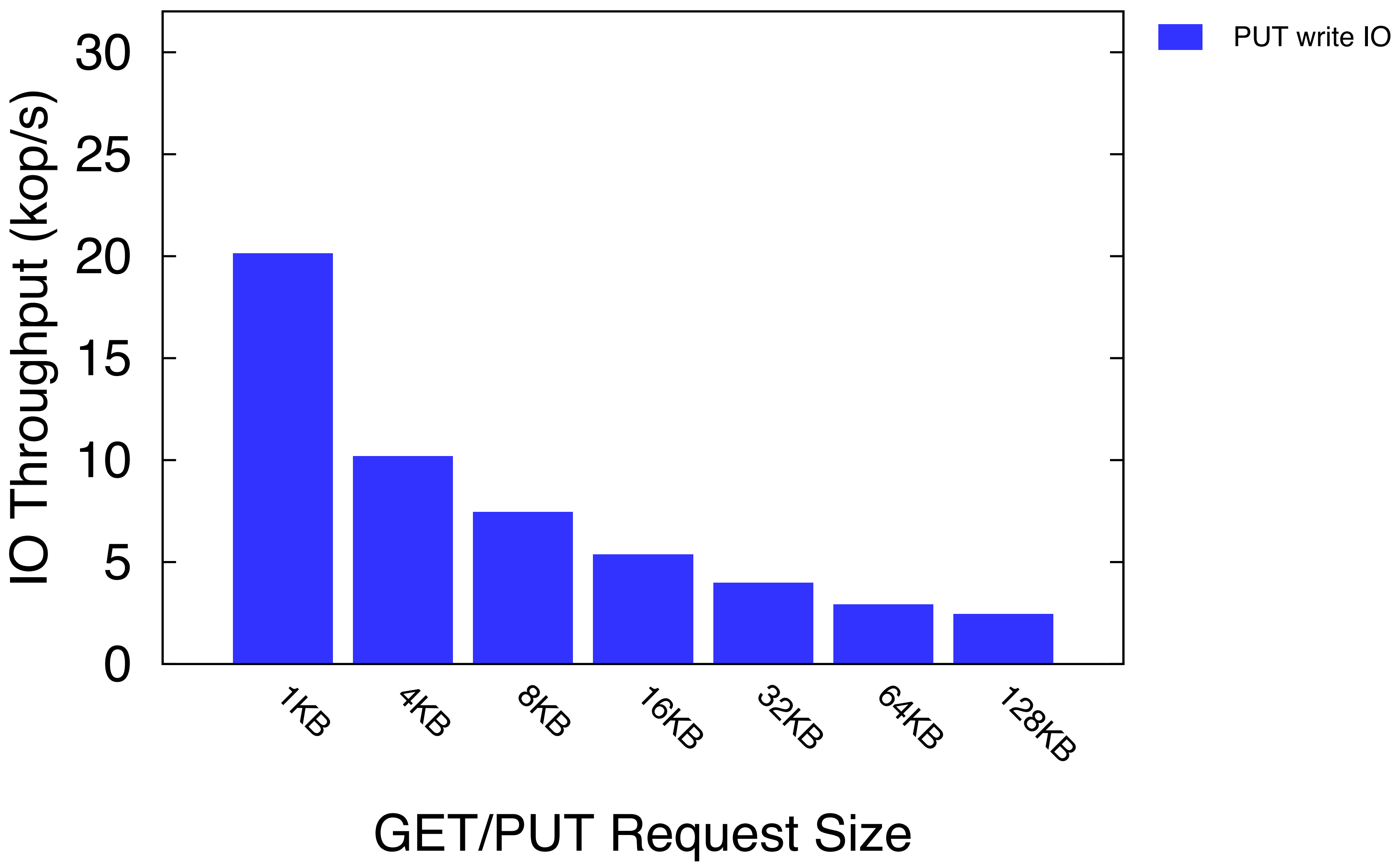
Non-linear IO cost

Model IO with Virtual IOPs

Workload-dependent IO Amplification

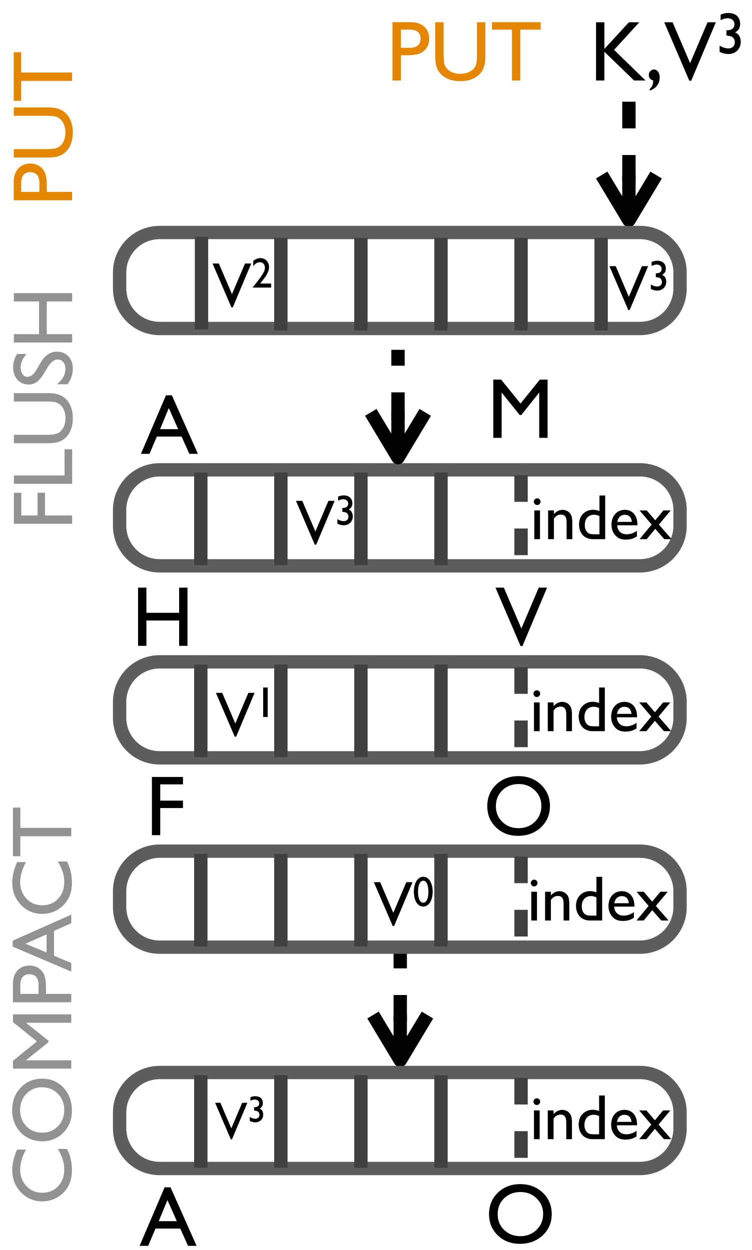
LevelDB (LSM-Tree)

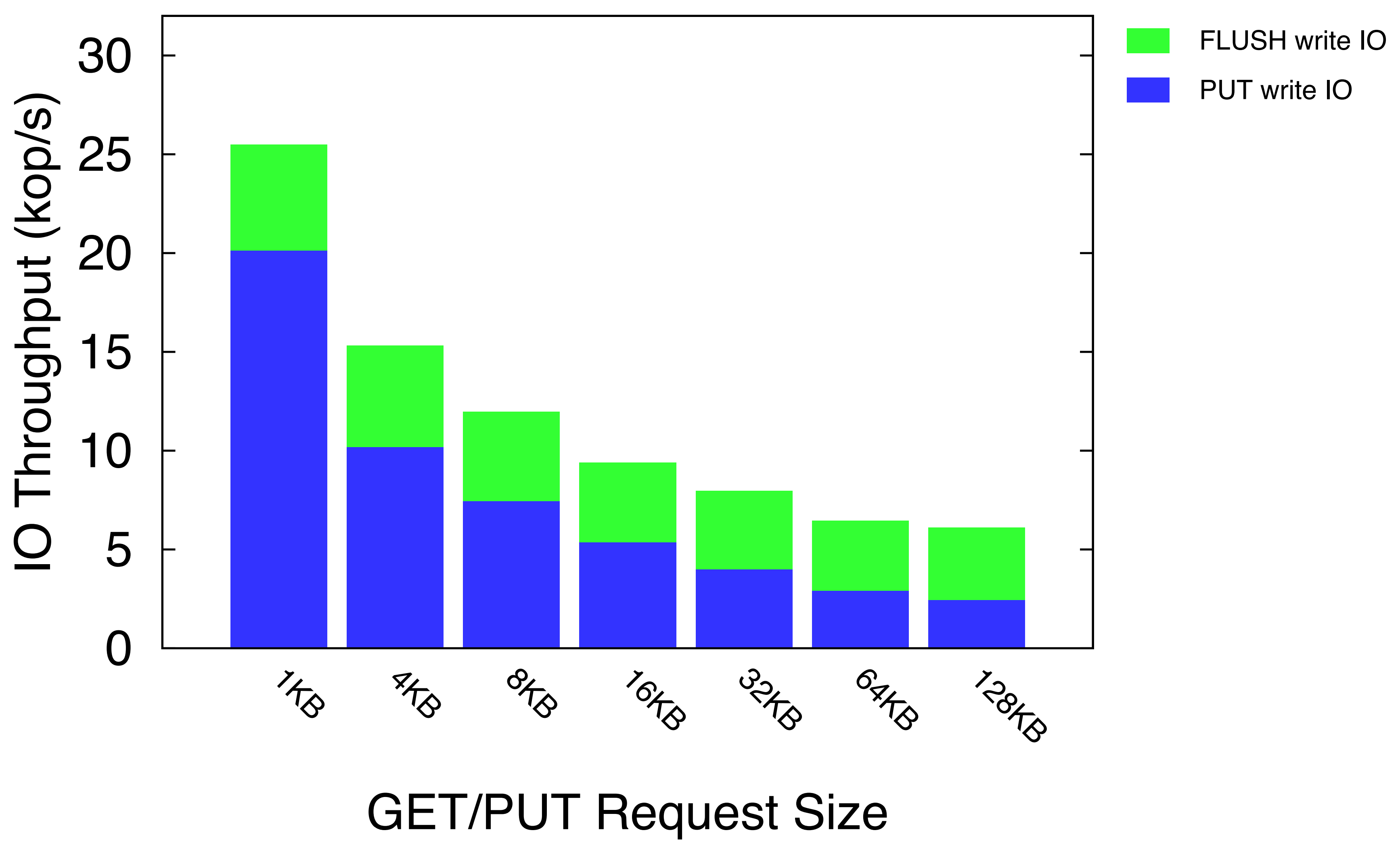




Workload-dependent IO Amplification

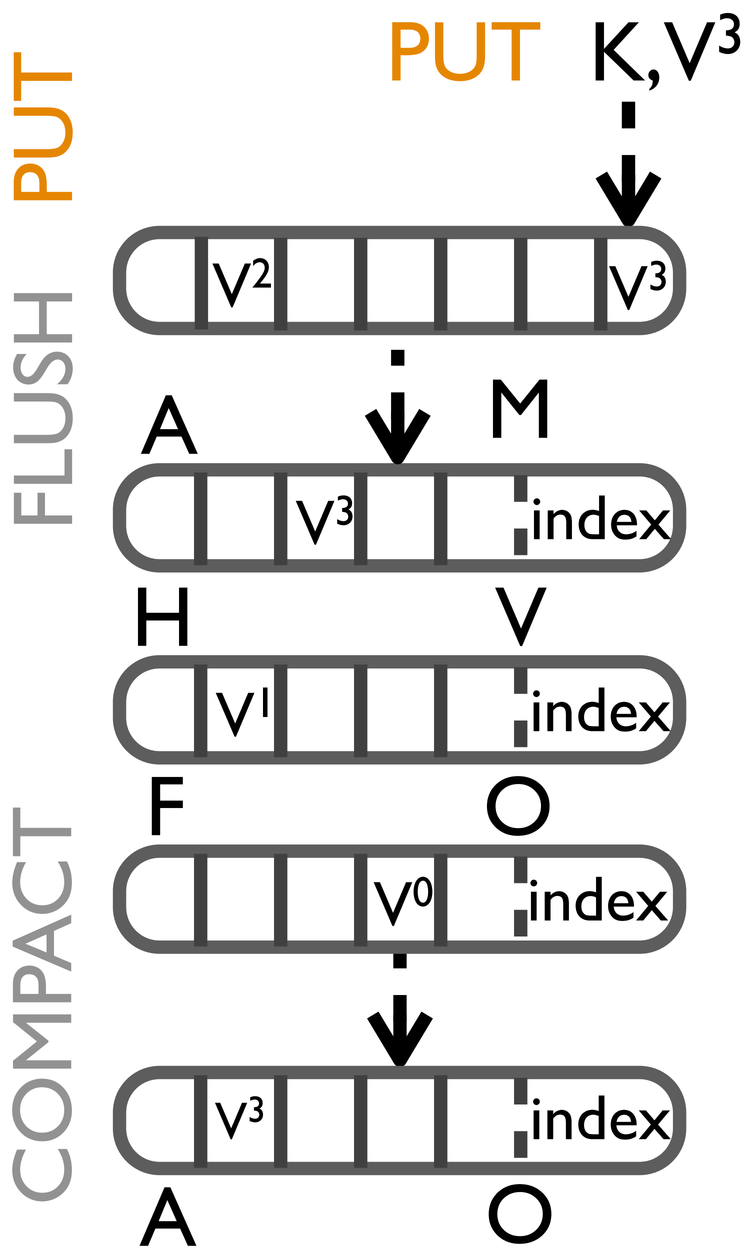
LevelDB (LSM-Tree)

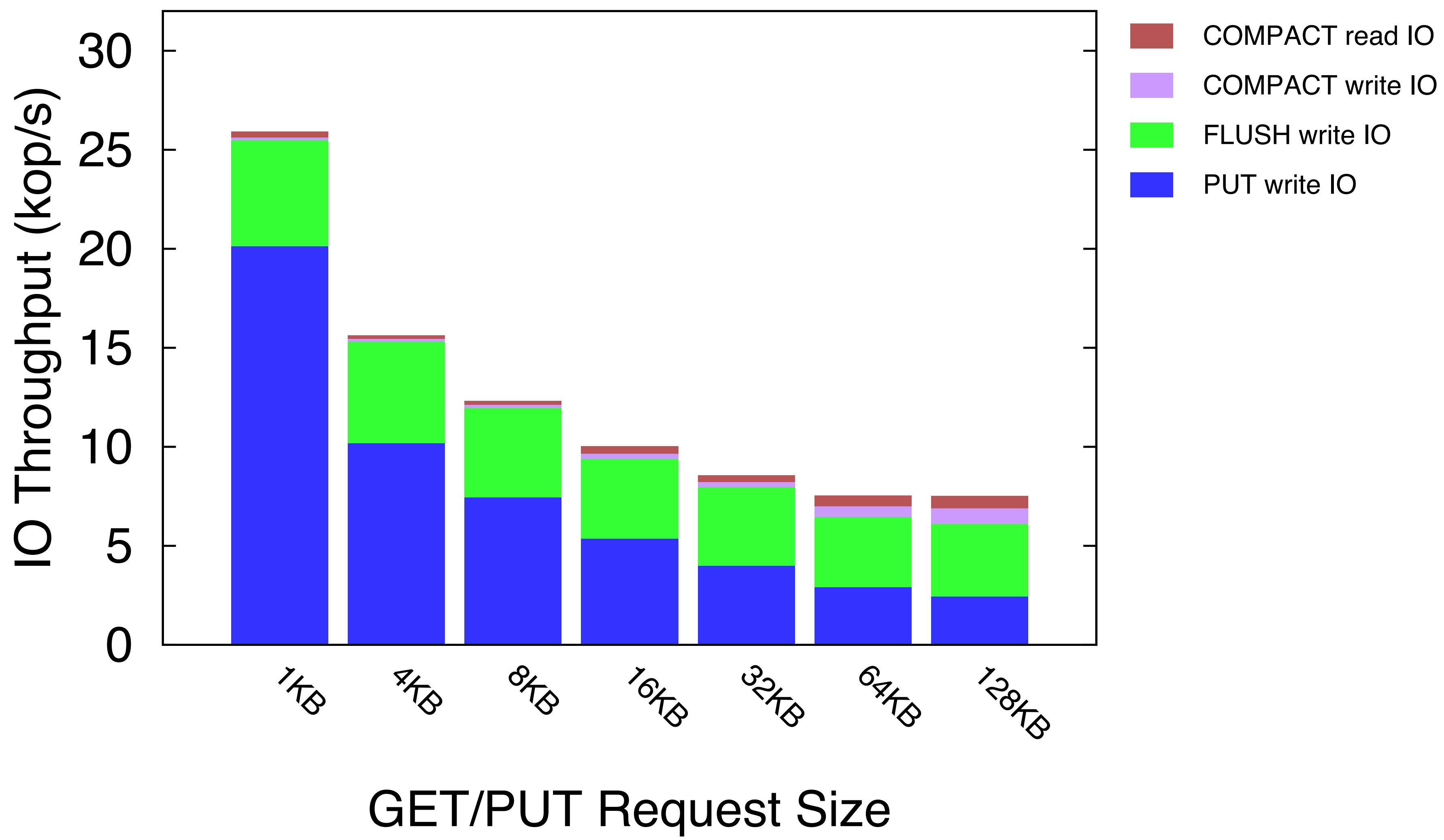




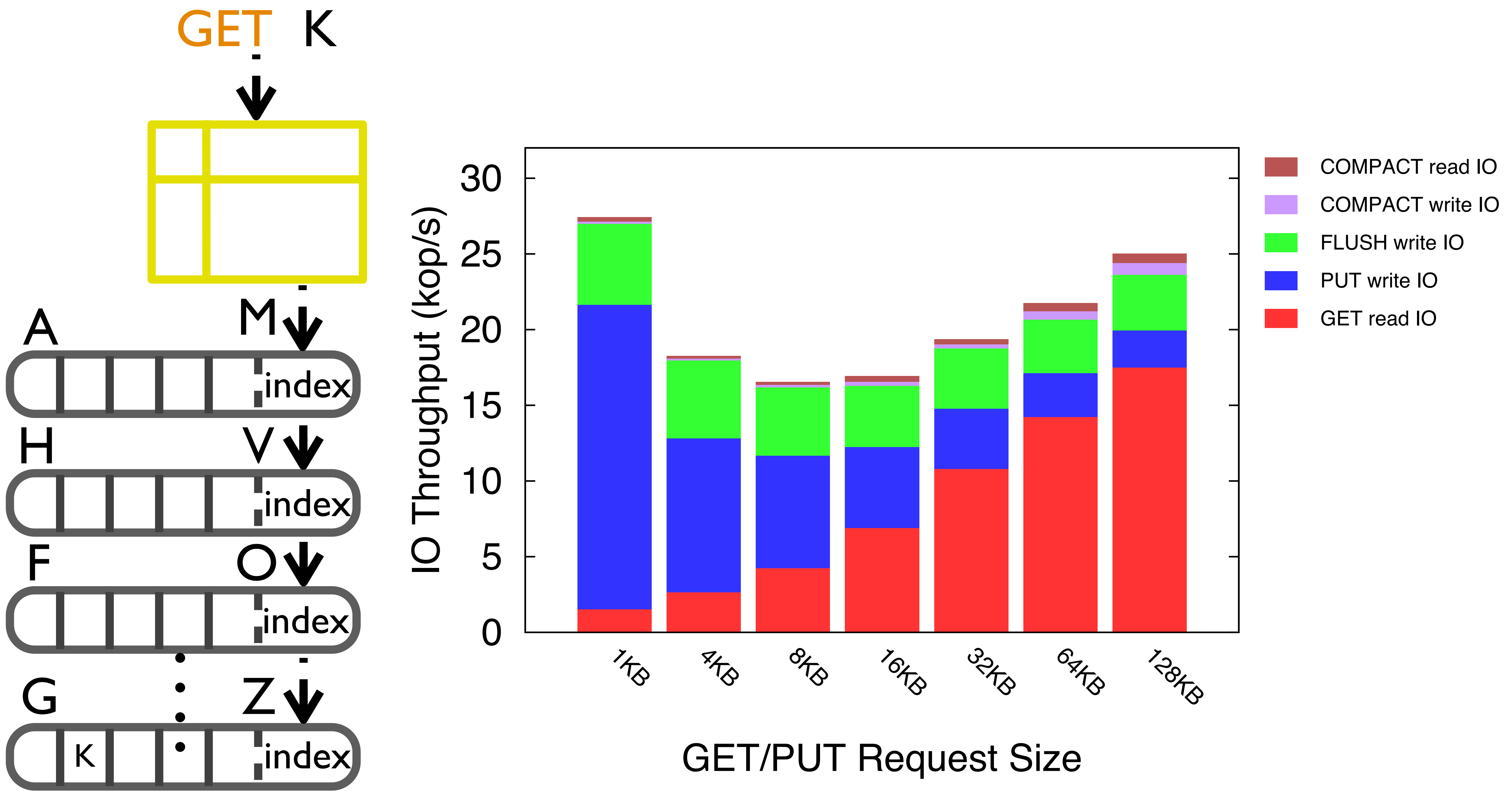
Workload-dependent IO Amplification

LevelDB (LSM-Tree)

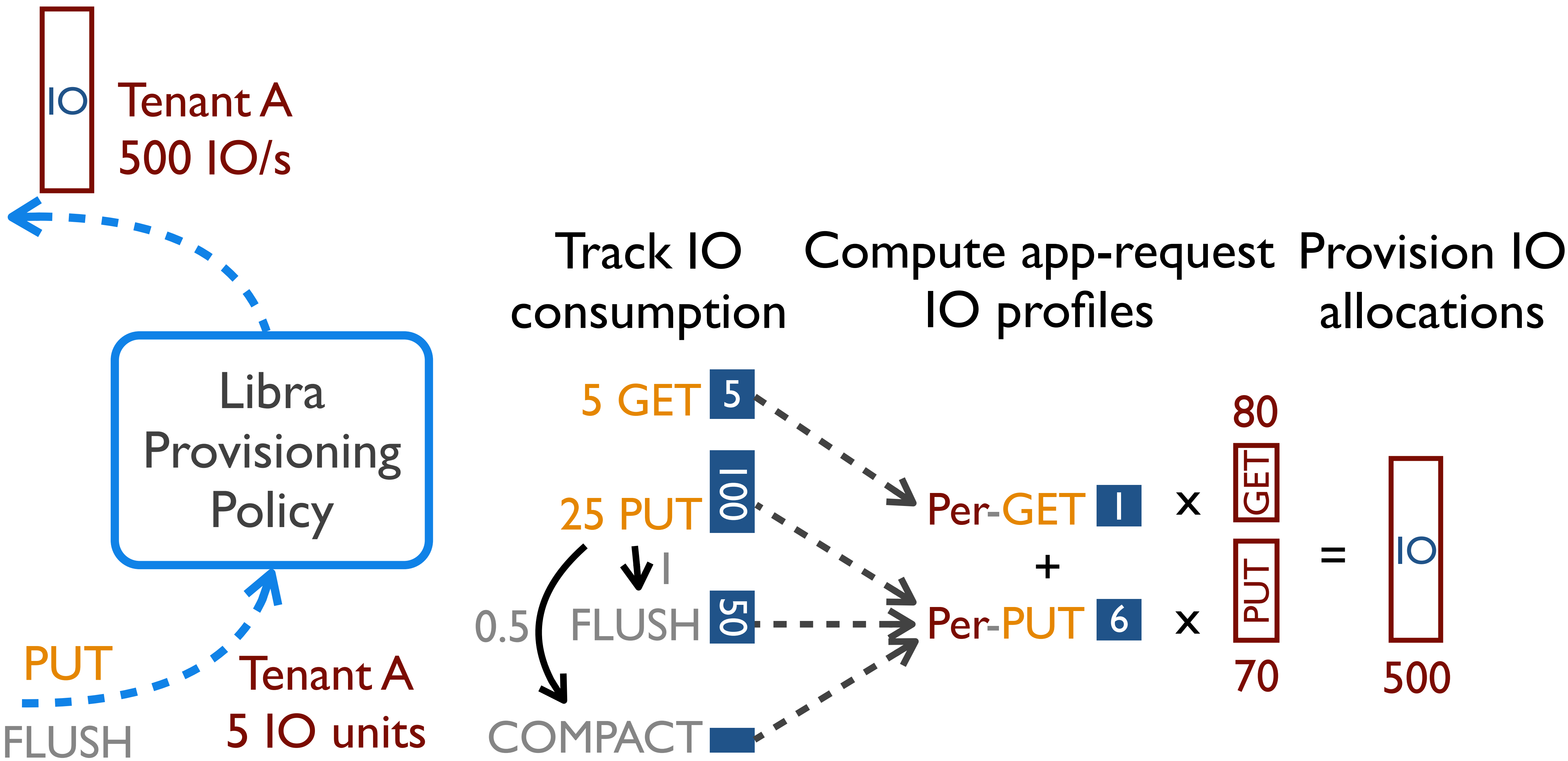




Workload-dependent IO Amplification



Libra Tracks App-request IO Consumption to Determine IO Allocations

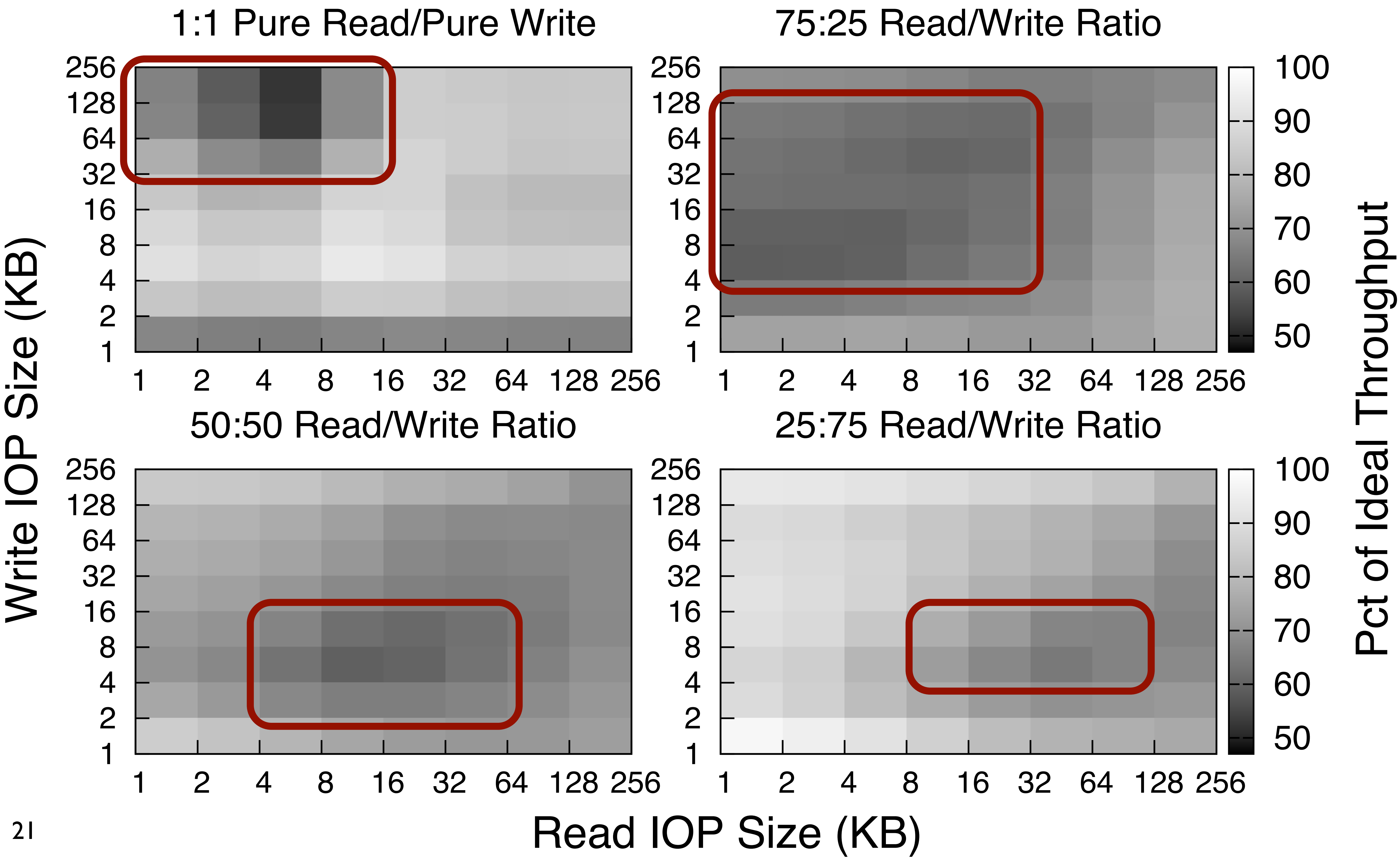


Unpredictable IO Interference

- Die-level parallelism, low latency IOPs
- Shared-controller and bus contention
- Erase-before-write overhead
- FTL and read-modify-write garbage colleciton

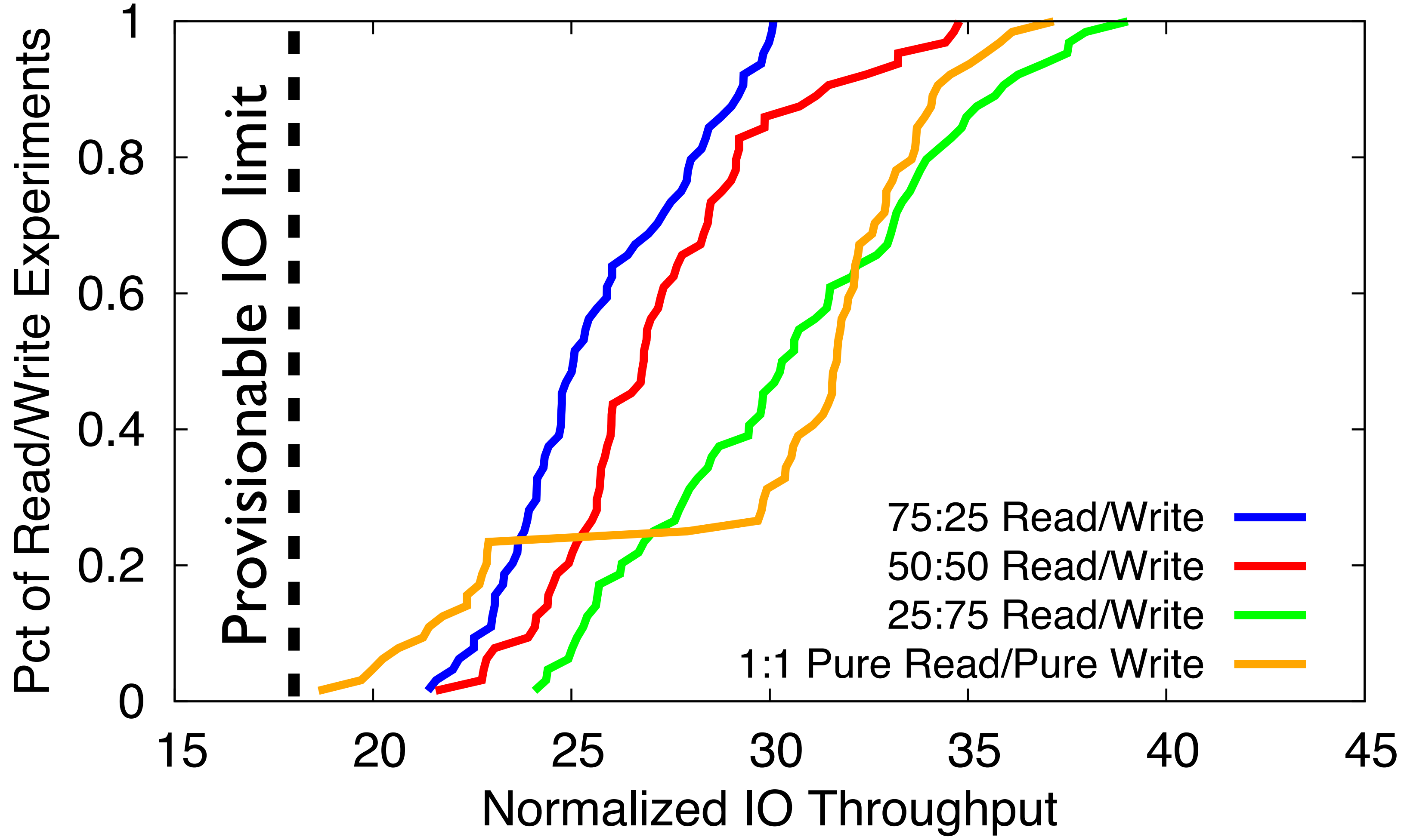
4 read/4 write tenants

Unpredictable IO Interference



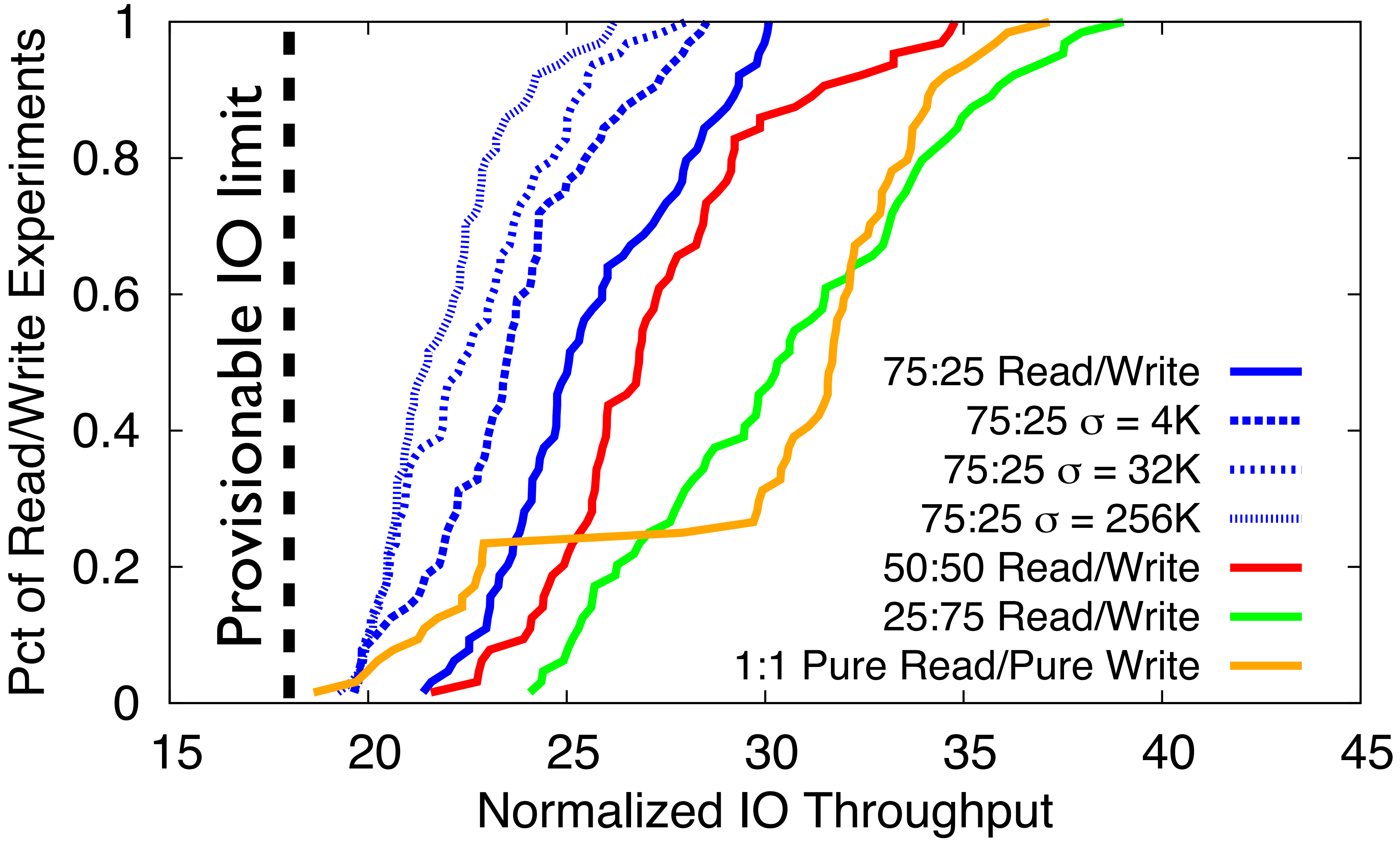
Libra Underestimates IO Capacity to Ensure Provisionable Throughput

Provisionable IO throughput = floor(workloads)
(18 Kop/s)



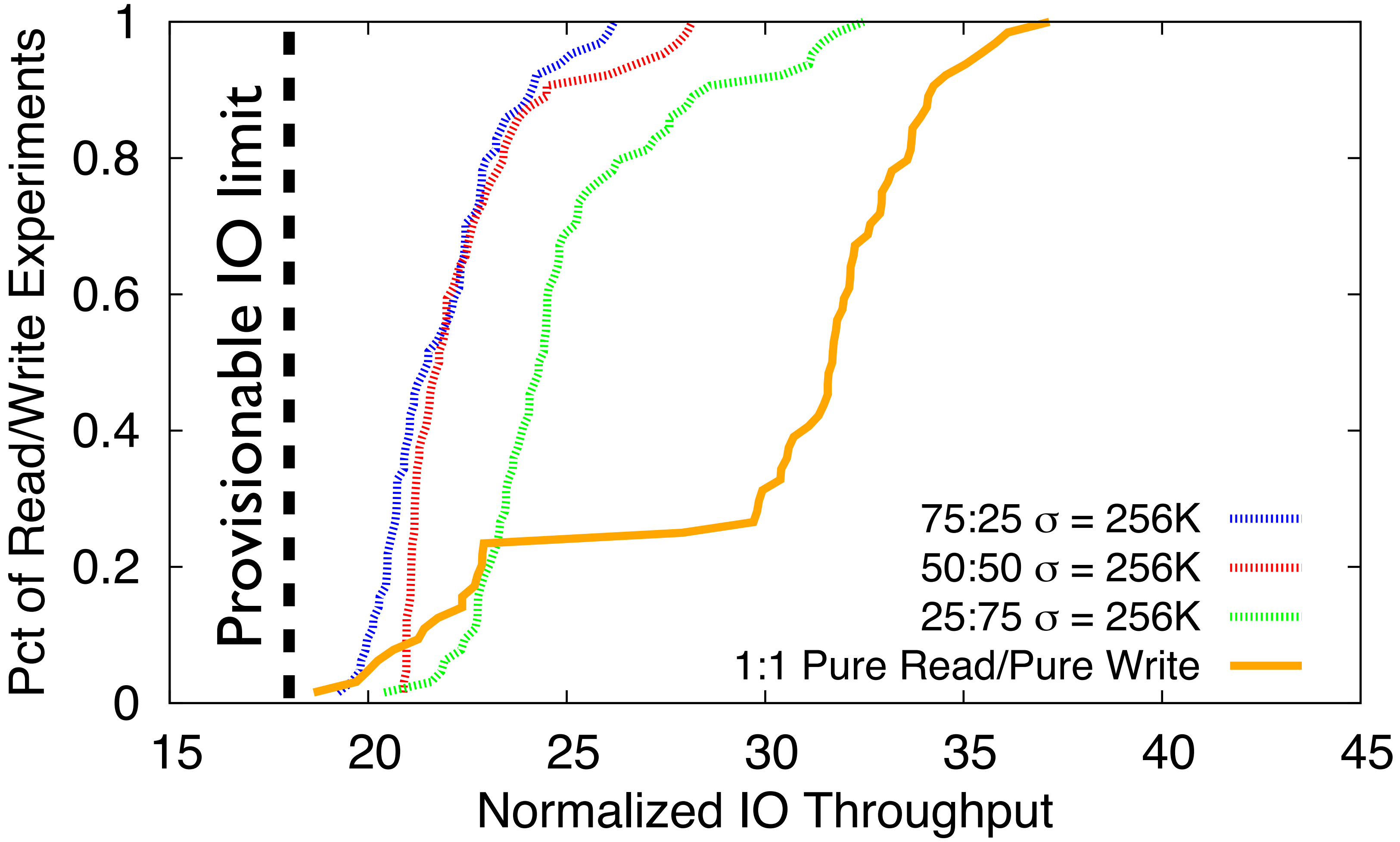
Libra Underestimates IO Capacity to Ensure Provisionable Throughput

Provisionable IO throughput = floor(workloads)
(18 Kop/s)

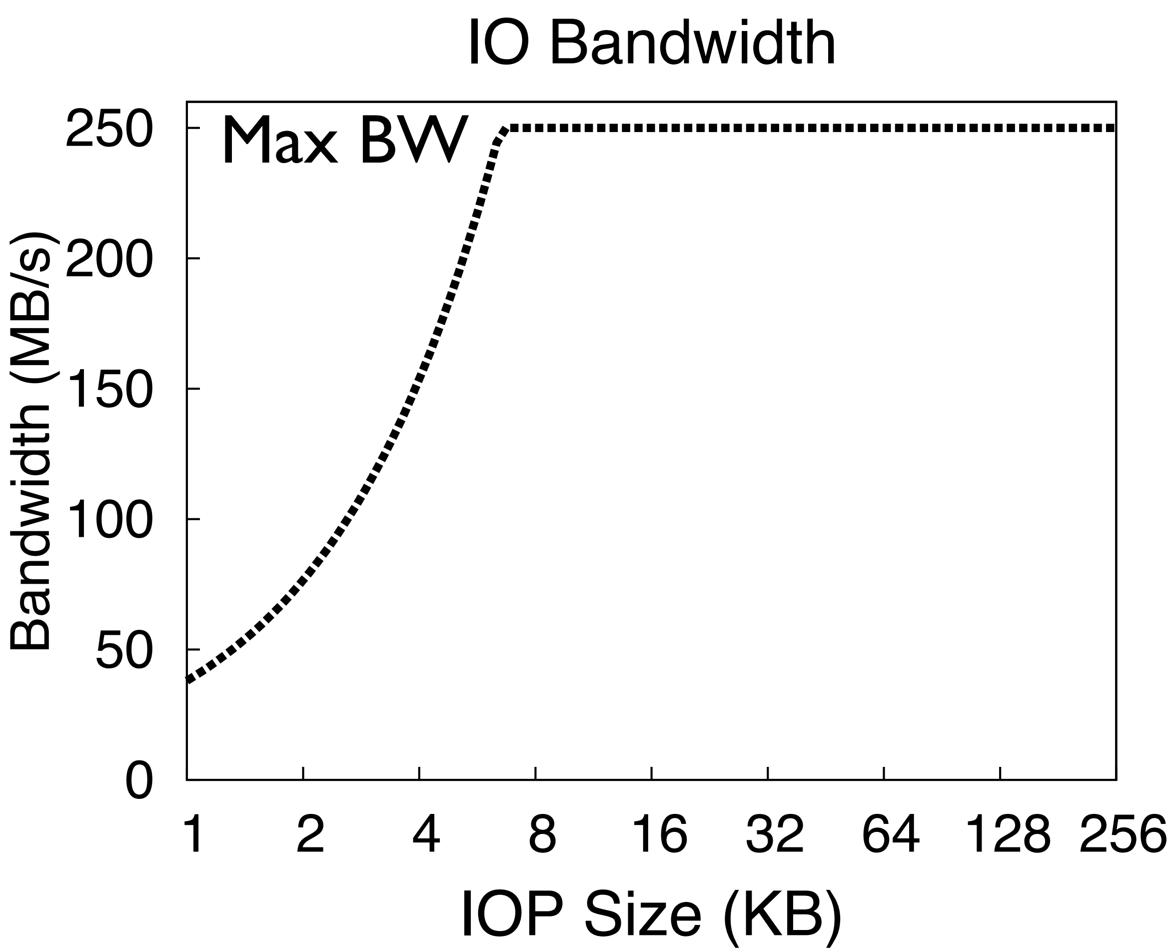


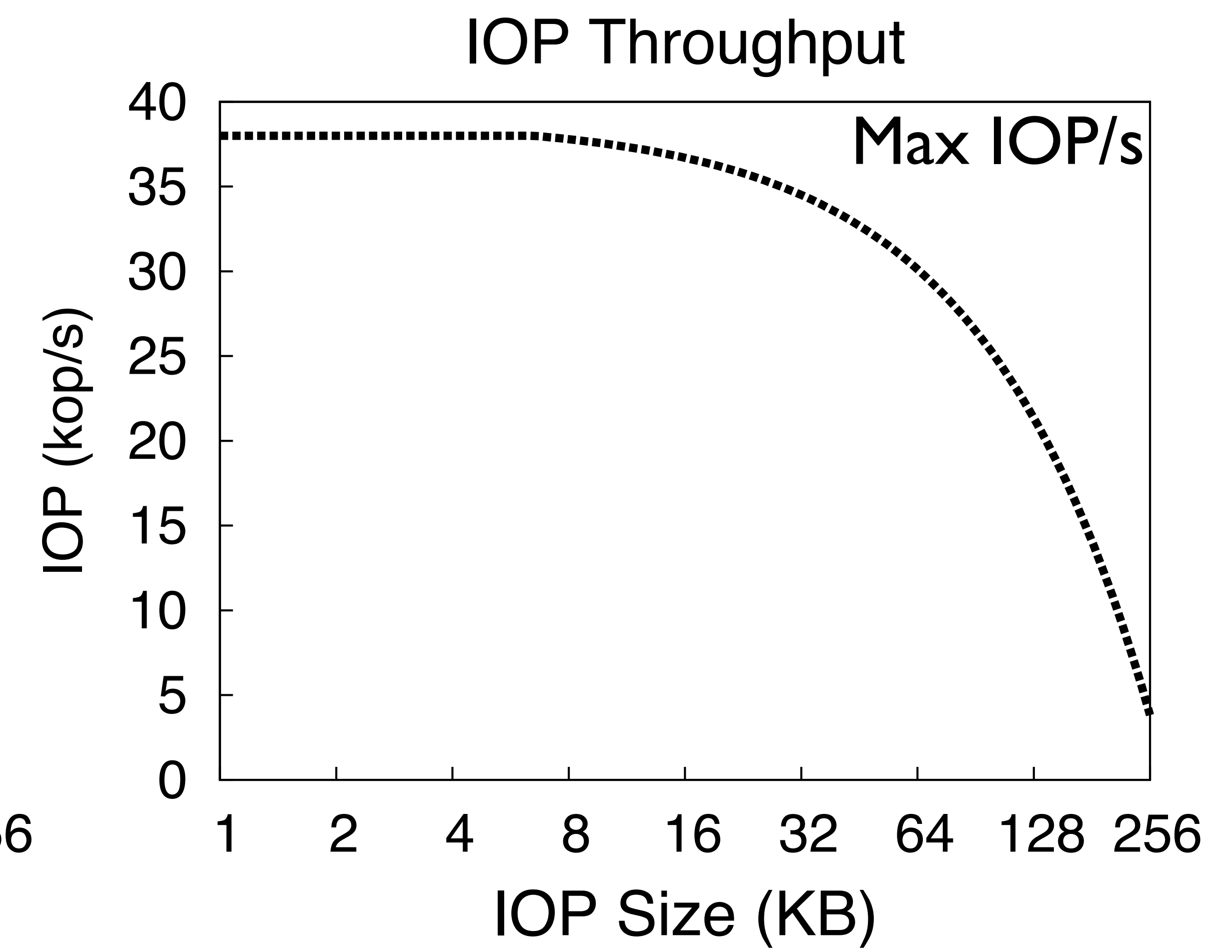
Libra Underestimates IO Capacity to Ensure Provisionable Throughput

Provisionable IO throughput = floor(workloads)
(18 Kop/s)

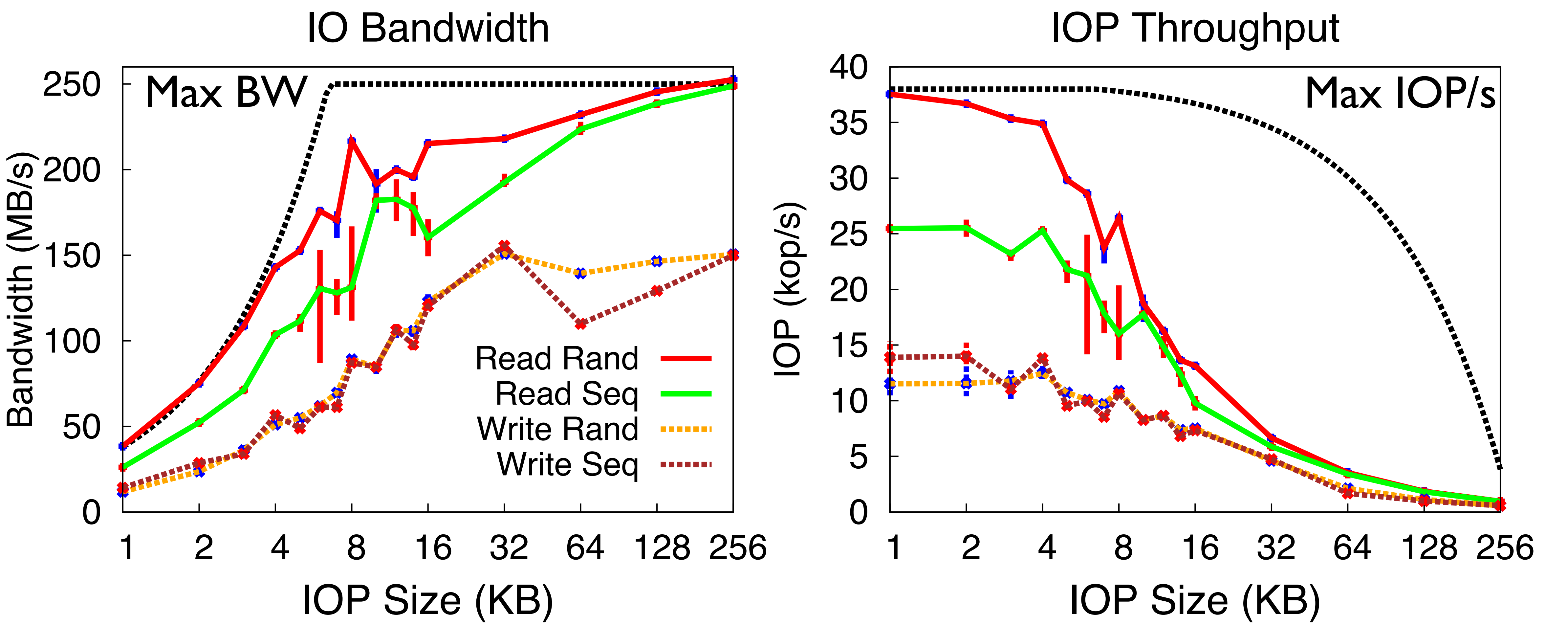


Non-linear IO Performance





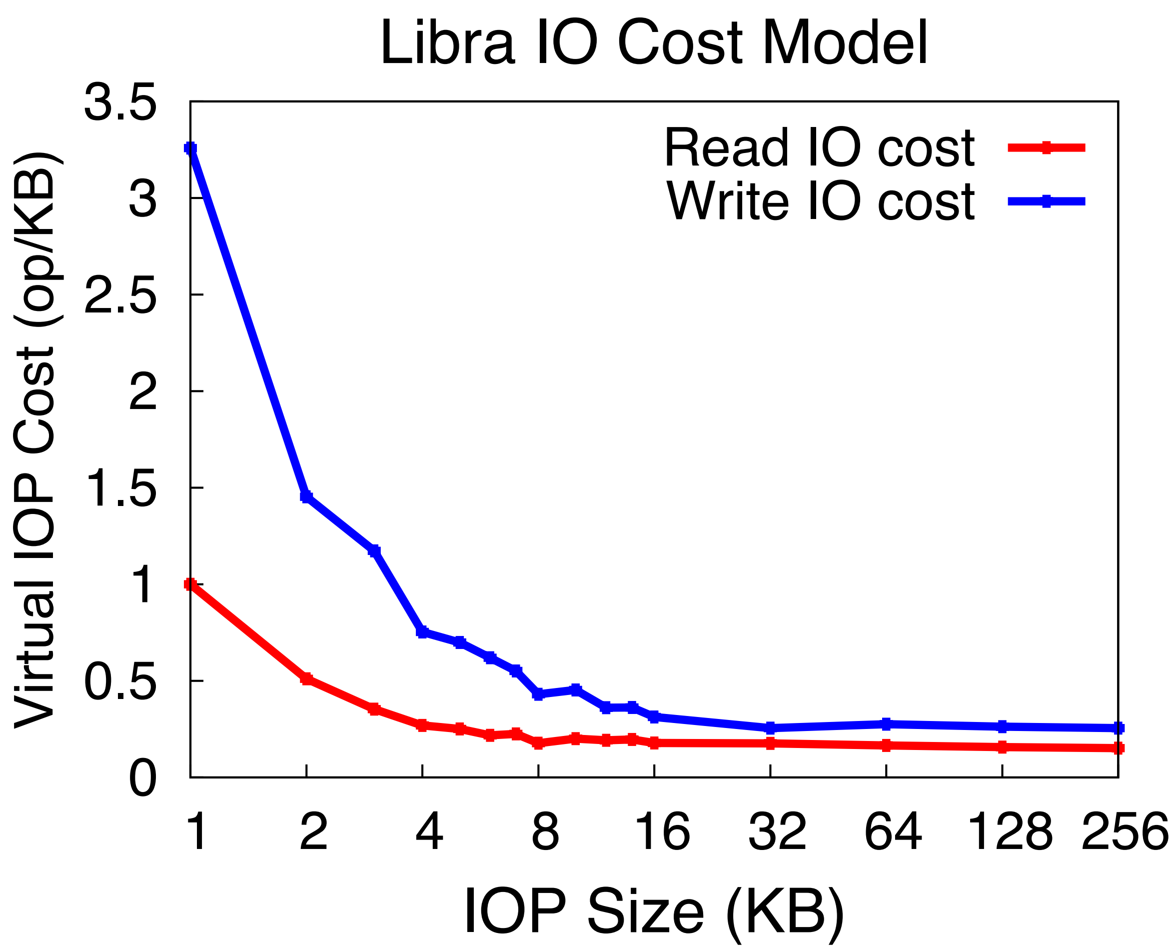
Non-linear IO Performance



Libra Uses Virtual IOPs to Model IO Resources

$$VOP_{CPB}(\text{IOP-size}) = \frac{\text{Max-IOP}}{\text{Achieved-IOP}(\text{IOP-size}) \times \text{IOP-size}}$$

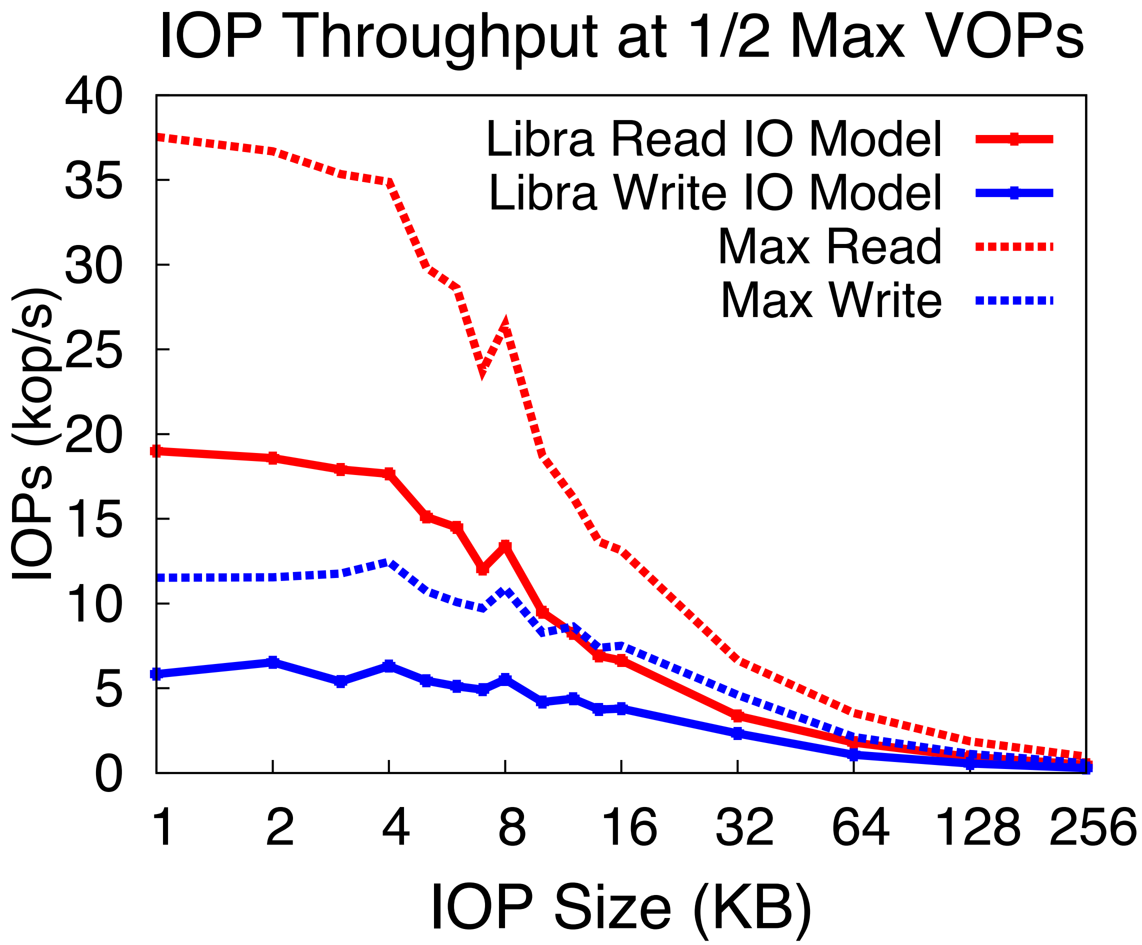
- Unifies IO cost into a single metric
- Captures non-linear IO performance
- Provides IO insulation

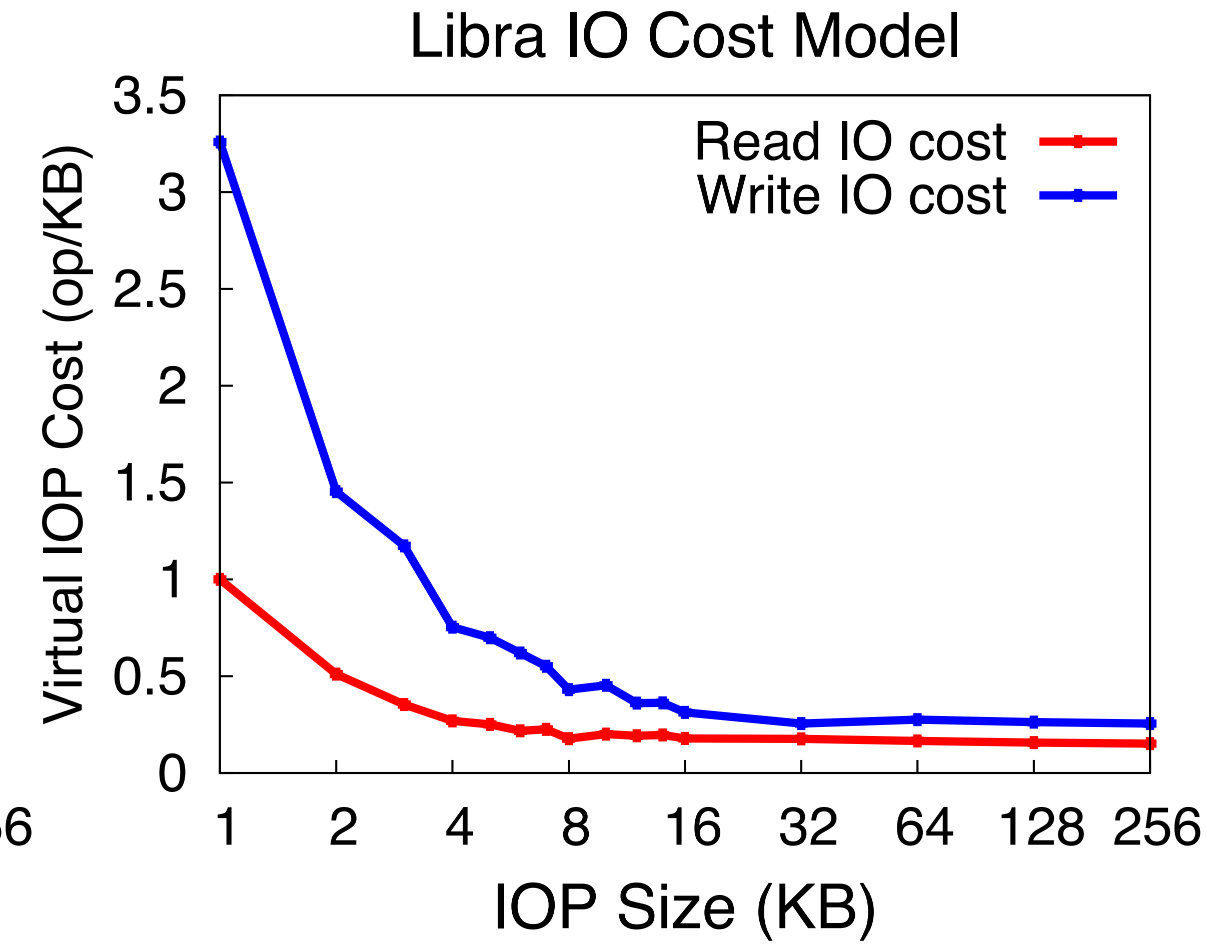


2 equal-allocation tenants
IO Insulation = 1/2 Max Read/Write

Libra Uses Virtual IOPs to Model IO Resources

$$VOP_{CPB}(\text{IOP-size}) = \frac{\text{Max-IOP}}{\text{Achieved-IOP}(\text{IOP-size}) \times \text{IOP-size}}$$

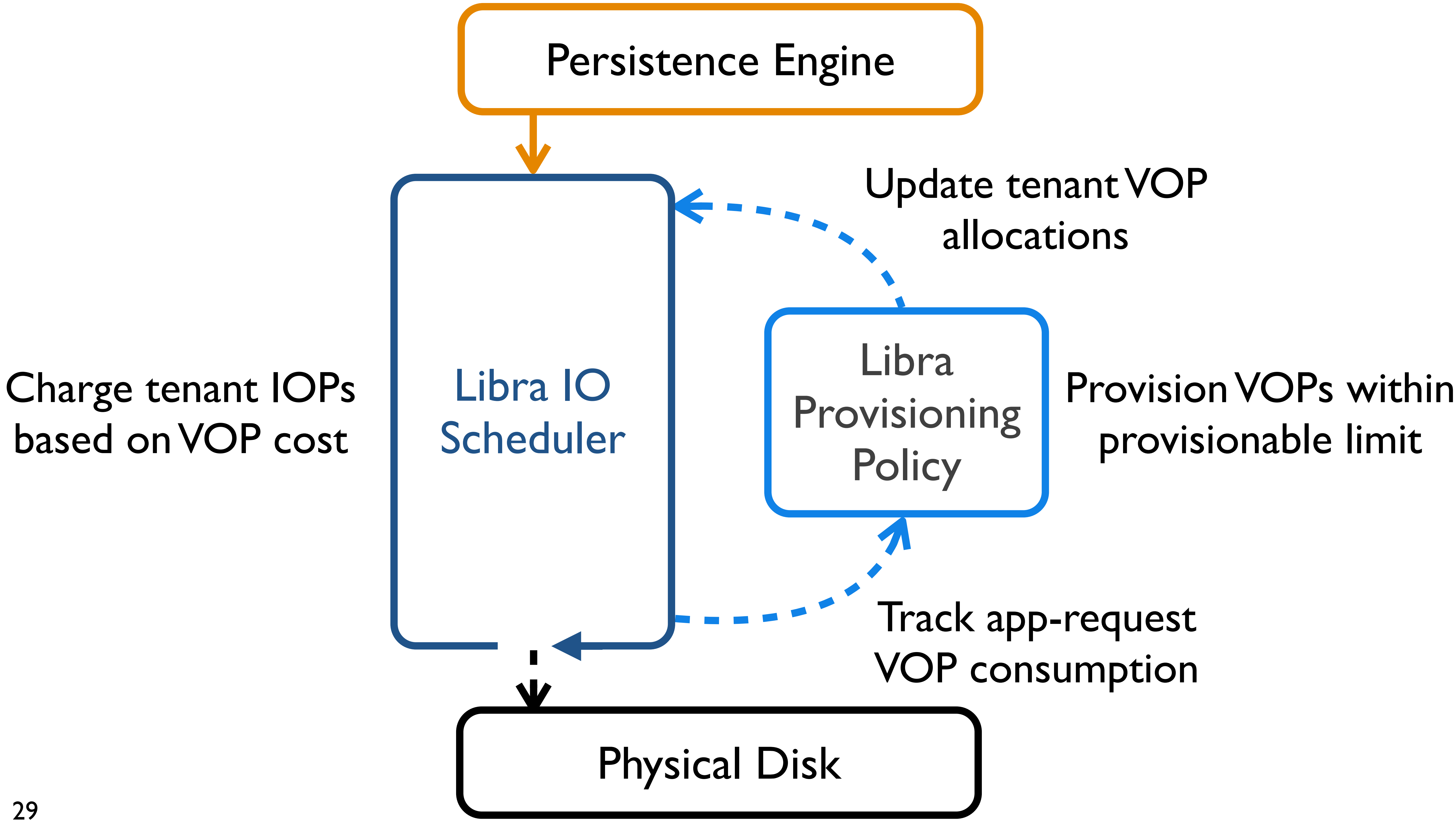




2 equal-allocation tenants

IO Insulation = 1/2 Max Read/Write

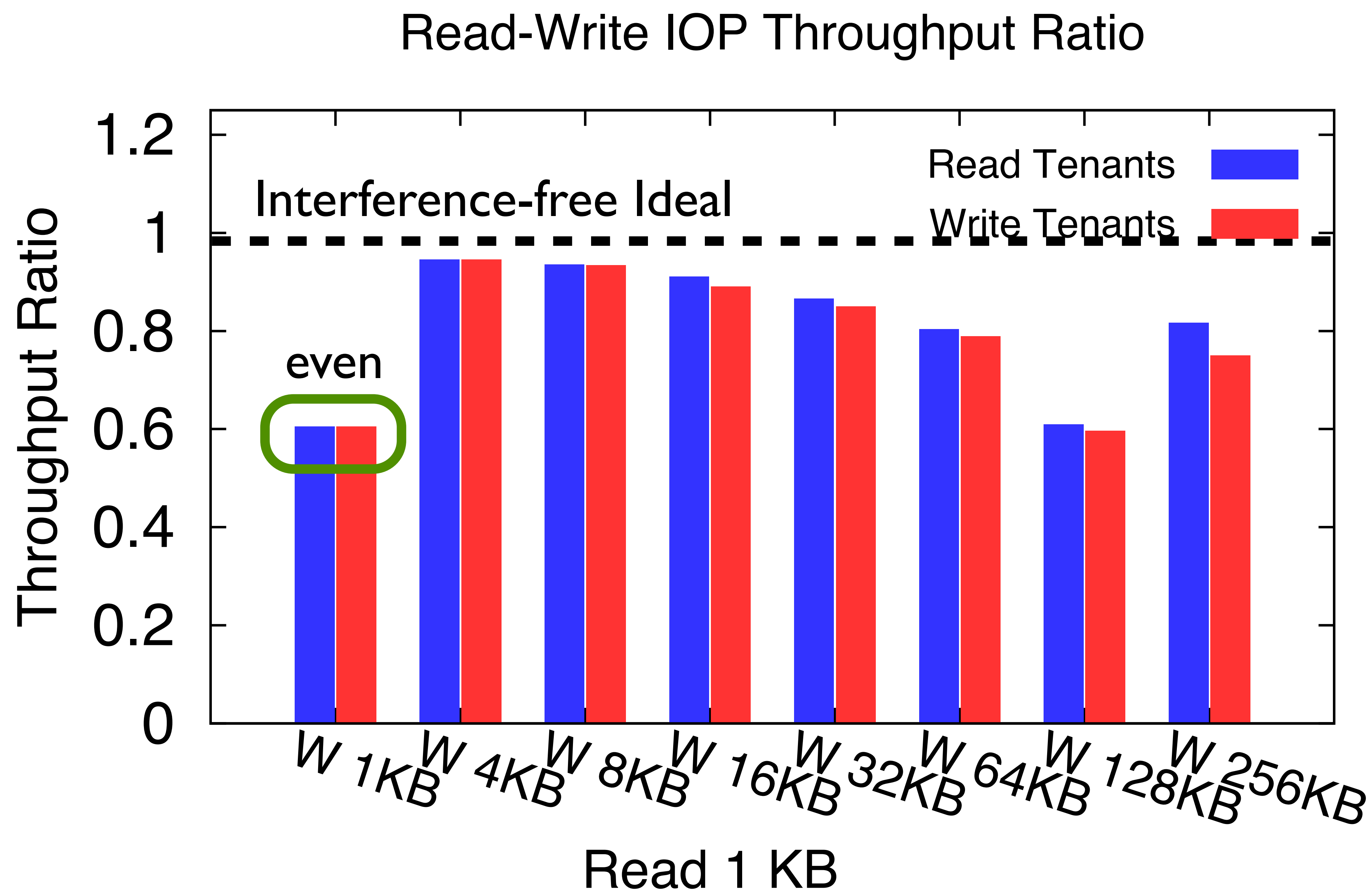
Libra Design



Evaluation

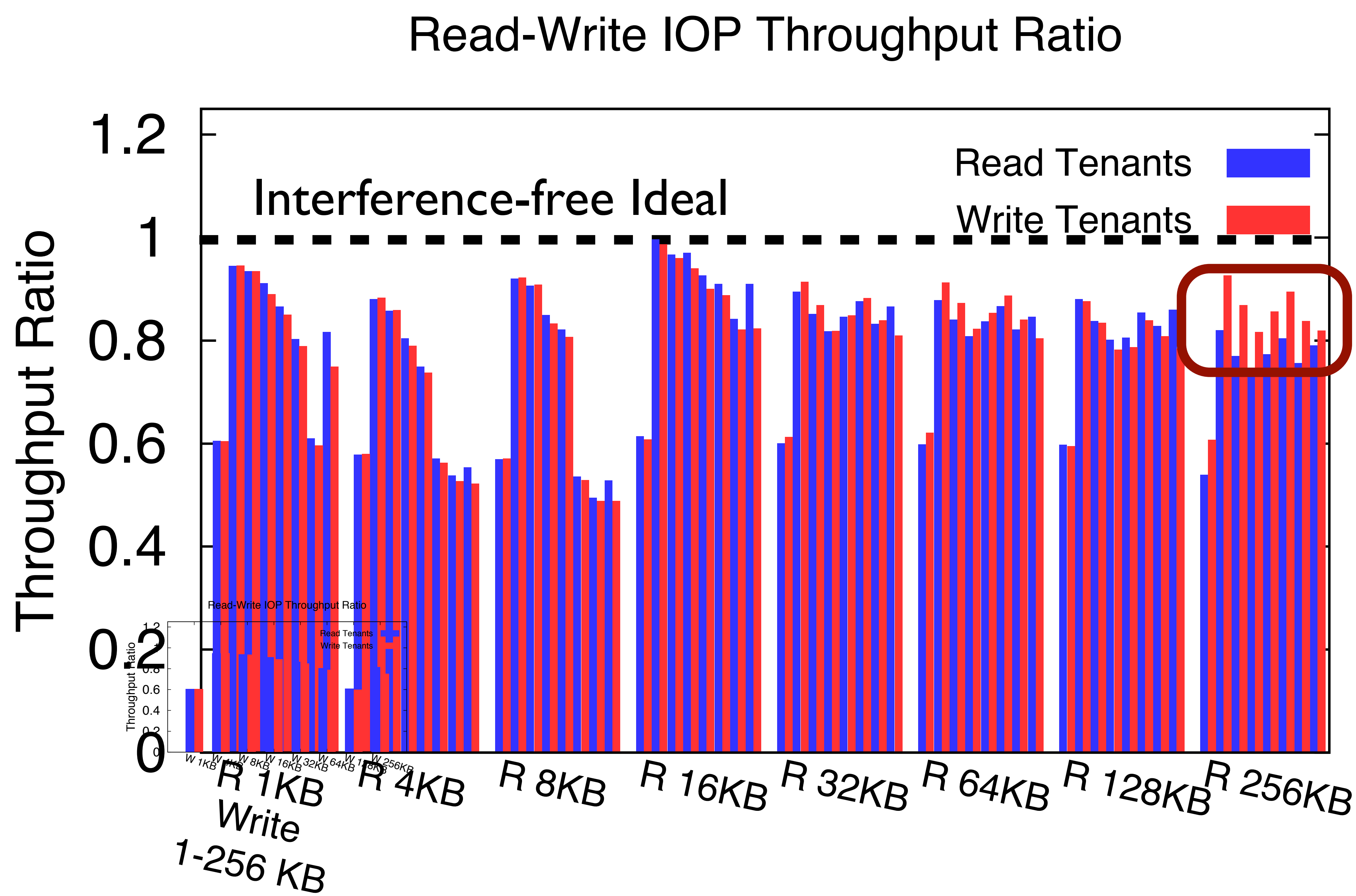
- Does Libra's IO resource model achieve accurate resource allocations?
- Does Libra's IO threshold make an acceptable tradeoff of performance for predictability in a real storage stack?
- Can Libra ensure per-tenant app-request reservations while achieving high utilization?

Libra Achieves Accurate IO Allocations



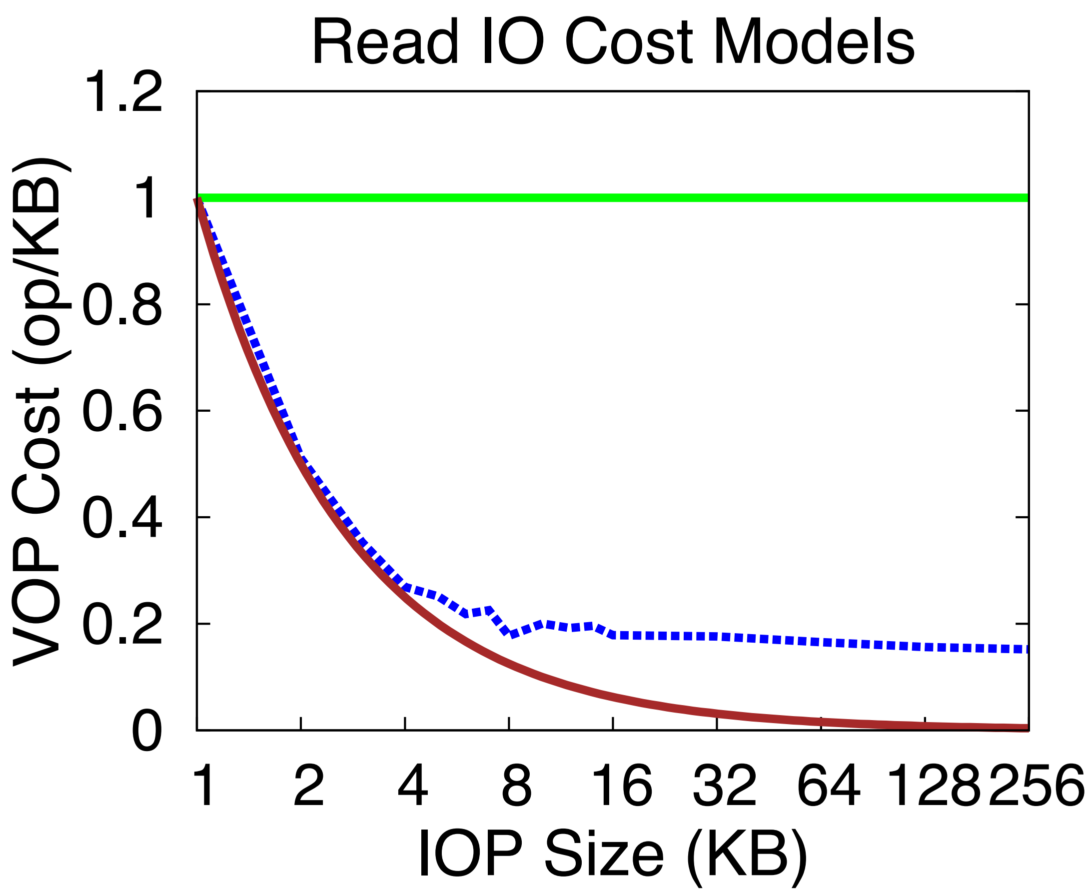
Throughput Ratio = Actual / Expected (IO Insulation)

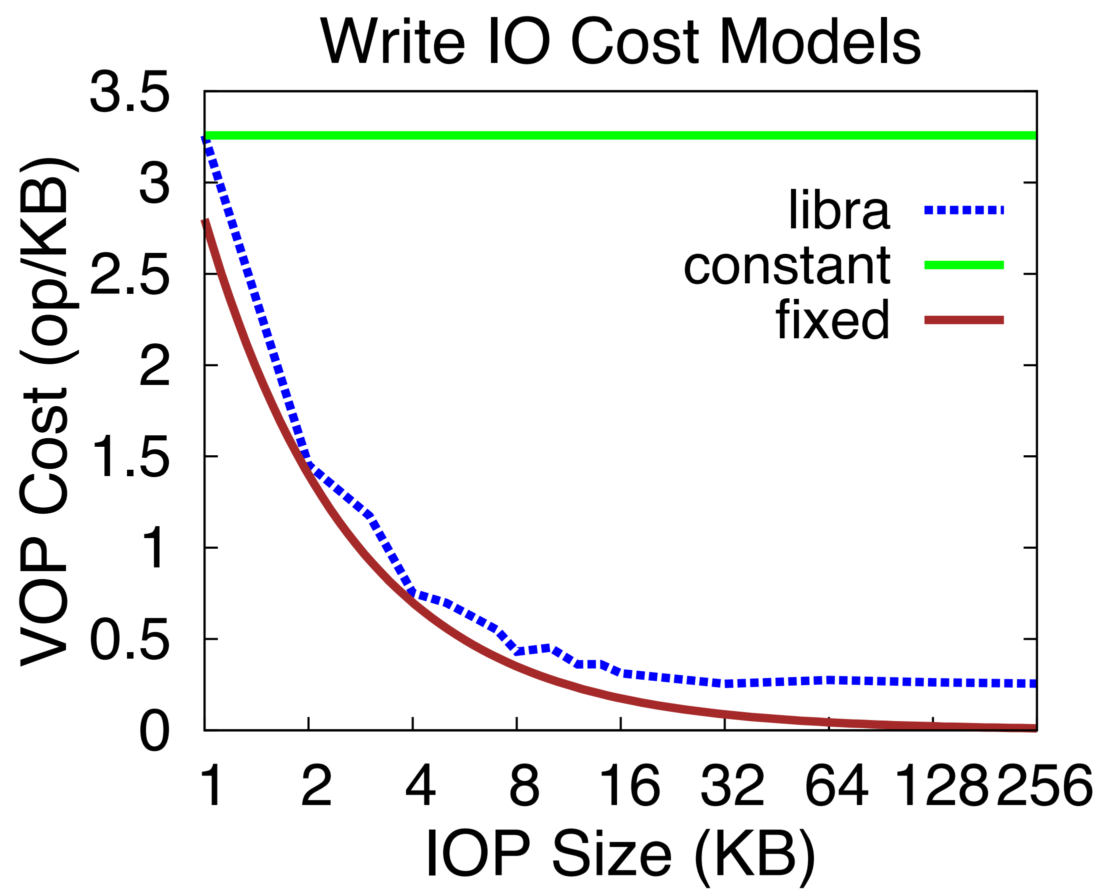
Libra Achieves Accurate IO Allocations



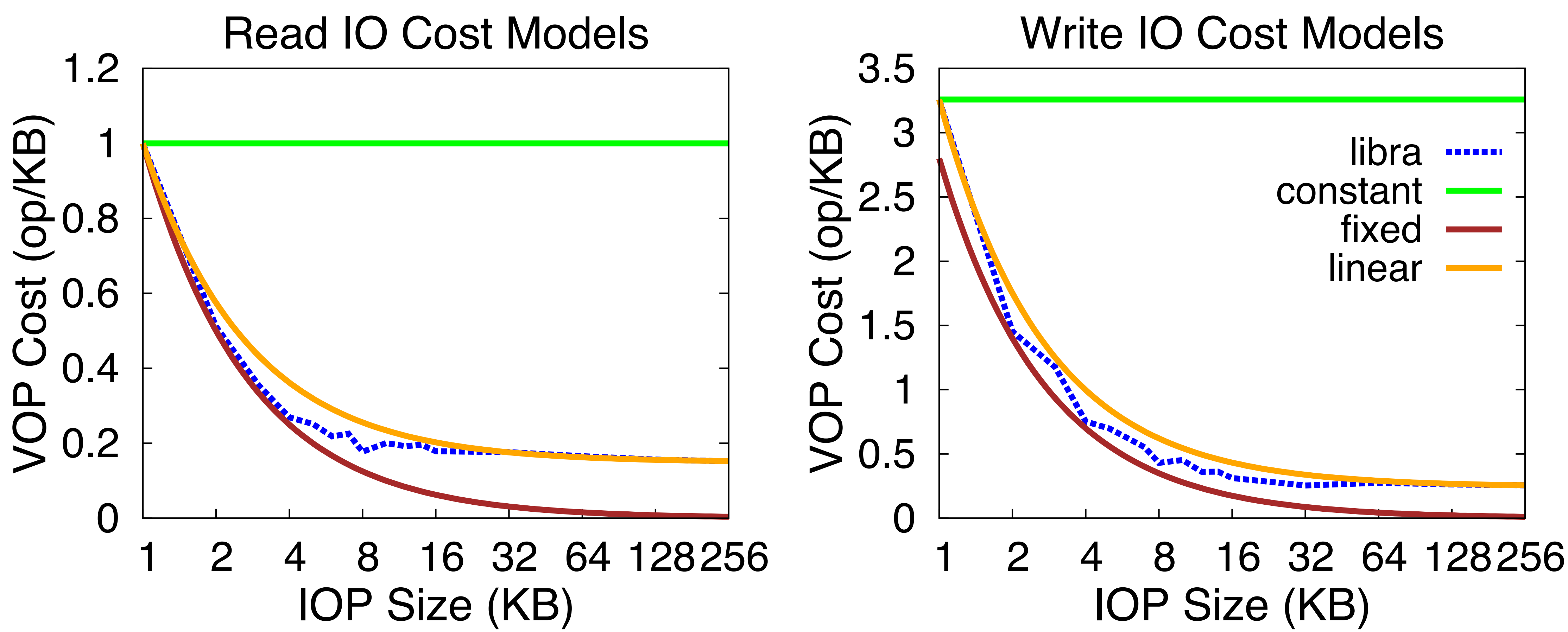
Throughput Ratio = Actual / Expected (IO Insulation)

Libra Achieves Accurate IO Allocations



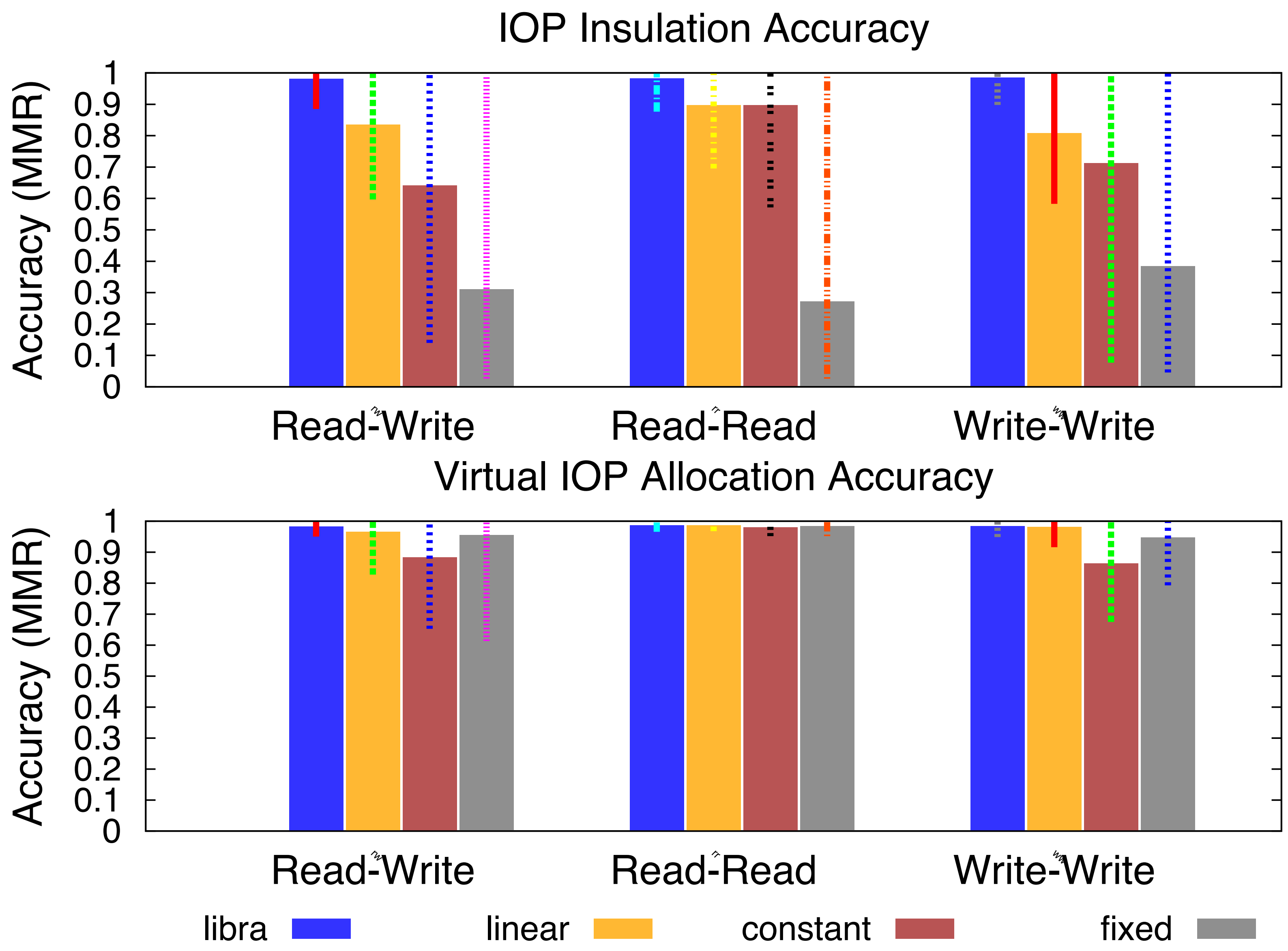


Libra Achieves Accurate IO Allocations

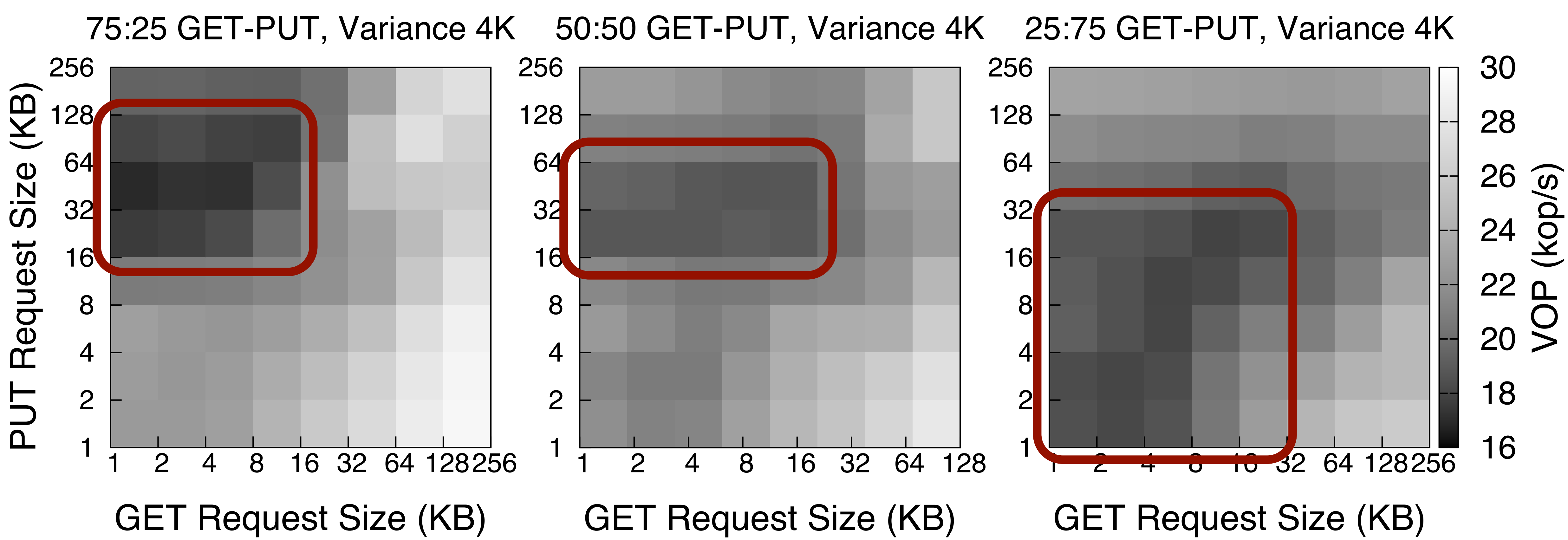


Libra Achieves Accurate IO Allocations

Min-Max Ratio =
Min Throughput Ratio / Max Throughput Ratio



Libra Trades-off Nominal IO Throughput For Predictability



Libra Trades-off Nominal IO Throughput For Predictability

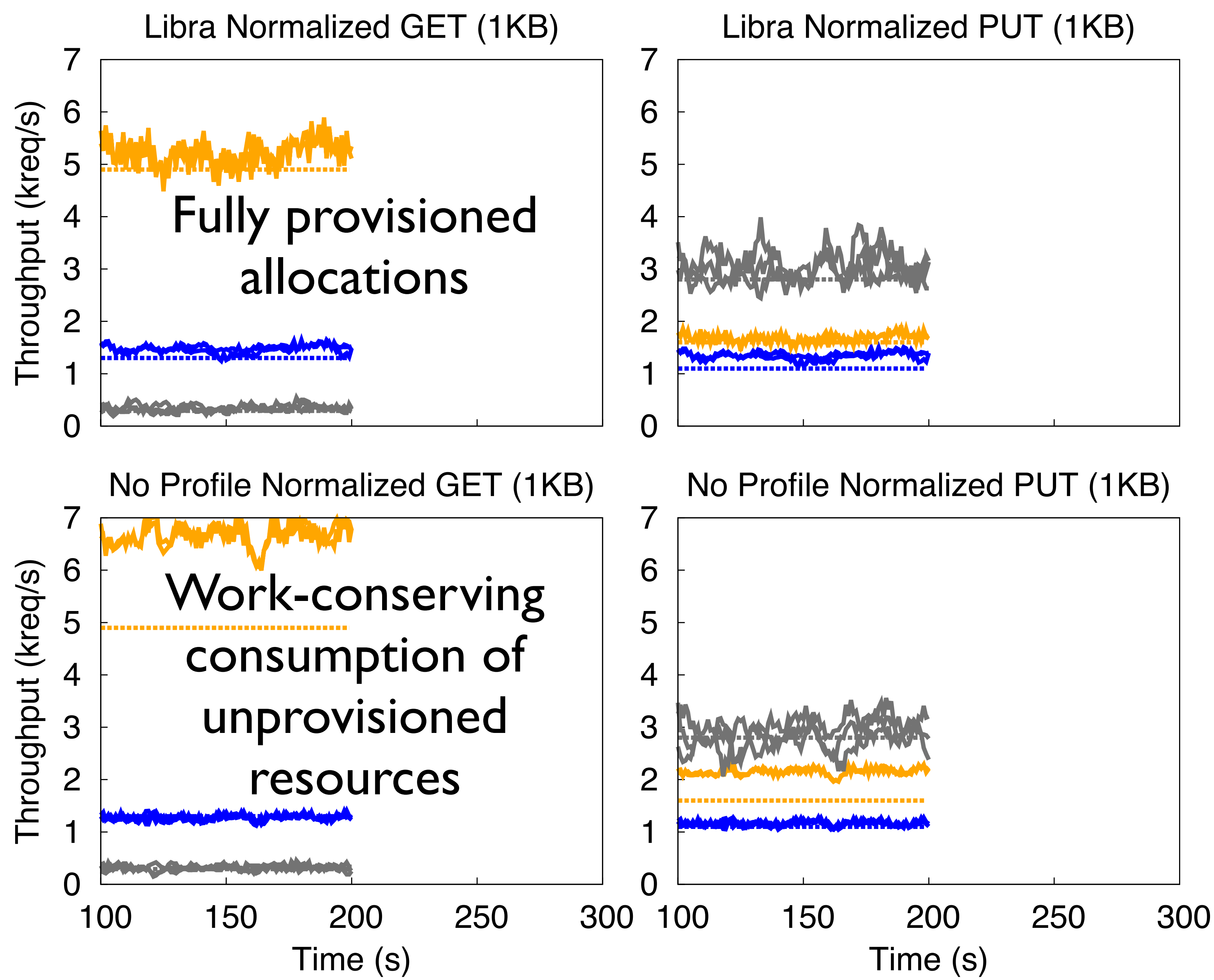
Unprovisionable Throughput As a Percentage of Total Throughput

	Percentile			
Workload	10th	50th	80th	All
99:1	1.6%	30.5%	40.5%	45.8%
25:75	1.4%	14.9%	25.0%	34.7%
1:99	0.7%	12.2%	19.5%	28.1%

< 10th percentile covered by SLA and higher-level policies

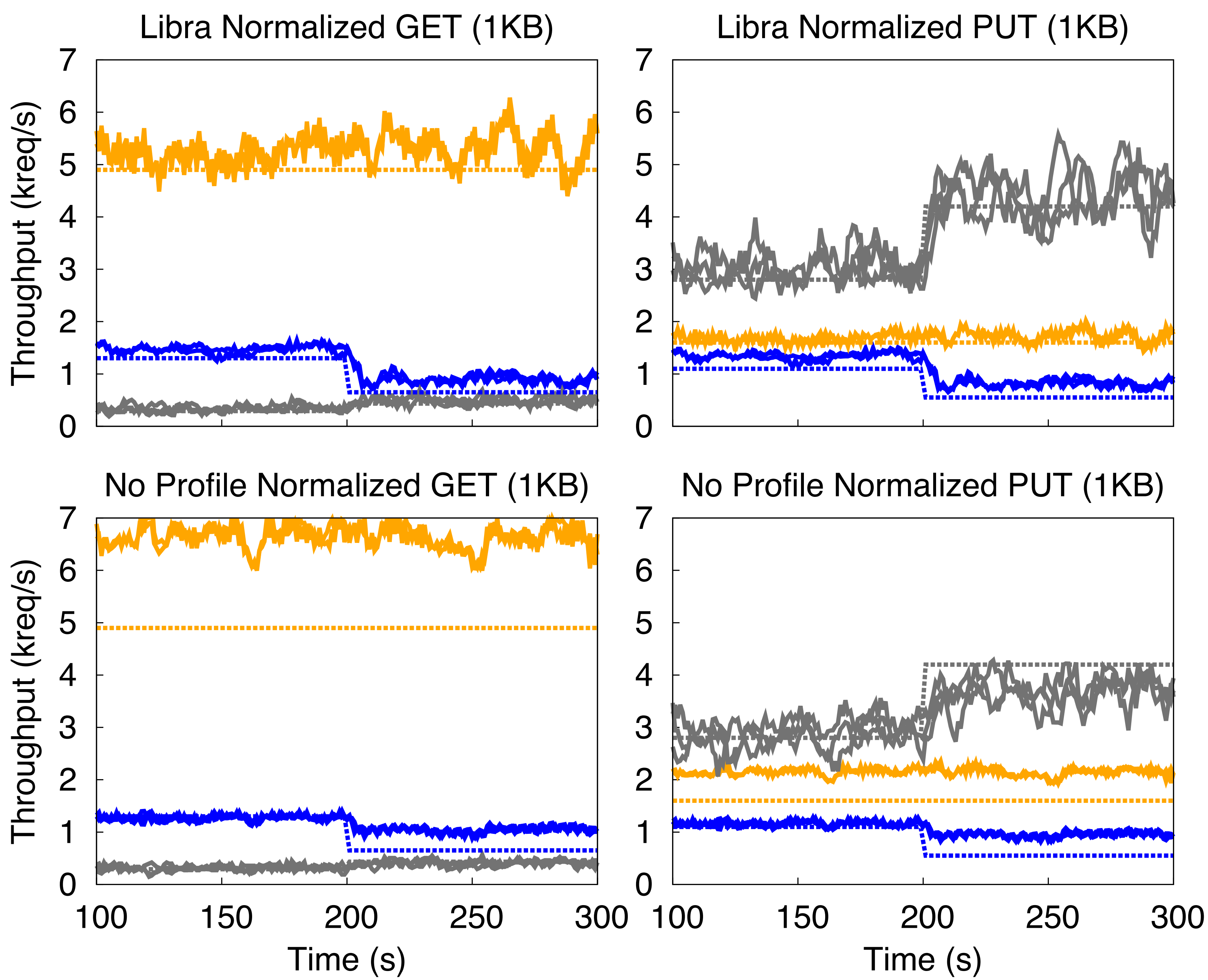
Libra Achieves App-request Reservations

0.5x | | 1.5x
Read Heavy | Mixed | Write Heavy



Libra Achieves App-request Reservations

Read Heavy | Mixed | Write Heavy



Conclusion

- Libra IO Scheduler
  - Provisions *IO allocations* for *app-request reservations* w/ high utilization.
  - Supports arbitrary object distributions and workloads.
- 2 key mechanisms
  - Track per-tenant app-request resource profiles.
  - Model IO resources with Virtual IOPs.
- Evaluation
  - Achieves accurate low-level IO allocations.
  - Provisions the majority of IO resources over a wide range of workloads
  - Satisfies app-request reservations w/ high utilization.