Pelican: A building block for exascale cold data storage

Shobana Balarishnan, Richard Black, Austin Donnelly, Paul England, Adam Glass, Dave Harper, Sergey Legtchenko, Aaron Ogus, Eric Peterson, Antony Rowstron

Microsoft Research, Microsoft

Przemysław Przybyszewski

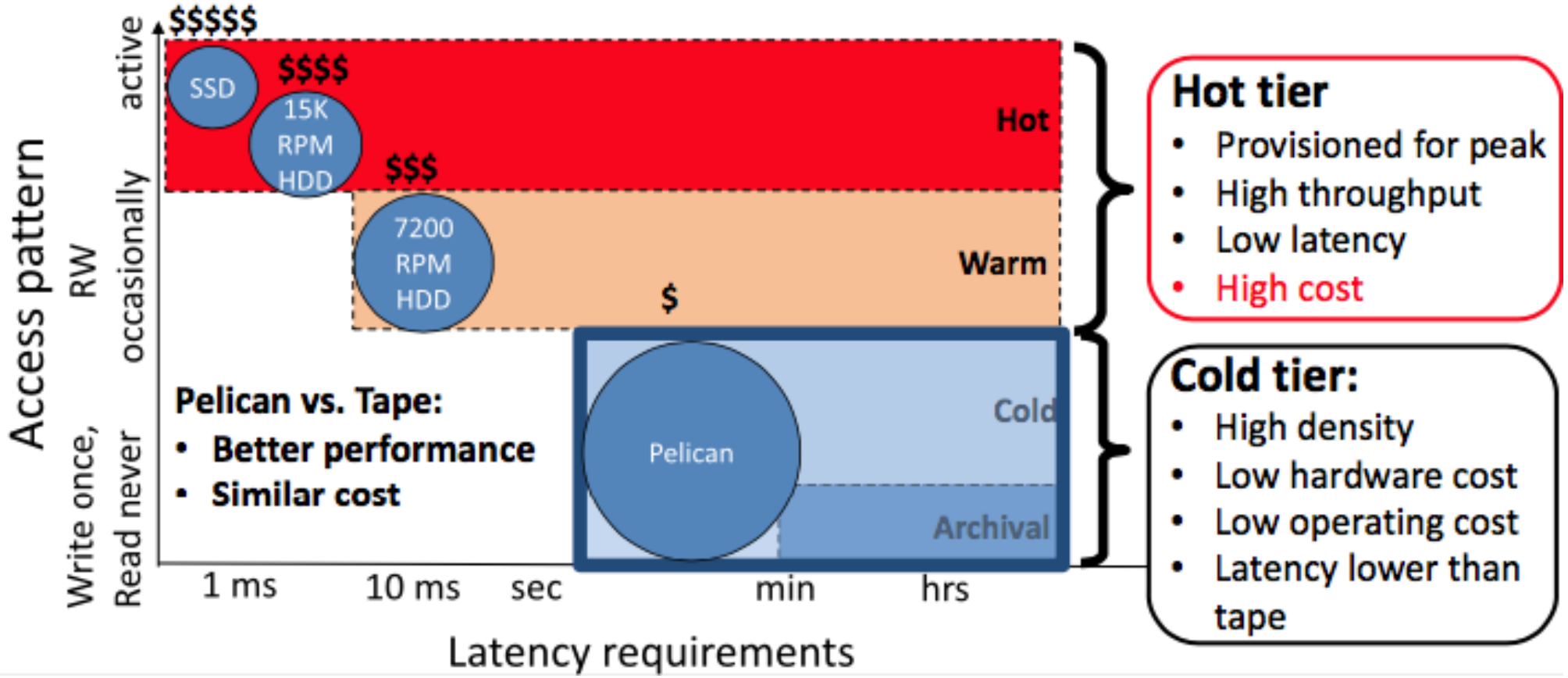
Cold data

Problem

- A significant portion of the data in the cloud is rarely accessed and known as cold data.
- Pelican tries to address the need for cost-effective storage of cold data and serves as a basic building block for exabyte scale storage for cold data.

Cold data in the cloud

- Cold data: **"written once – read rarely"** access pattern
- Large fraction of stored data



Right-provisioning

Provision resources **just** for the cold data workload:

- Disks:
  - Archival and SMR instead of commodity drives

- Power
- Cooling
- Bandwidth

Enough for the required workload

Not to keep all disks spinning

- Servers
  - Enough for data management instead of 1 server/ 40 disks

Pelican: rack-scale appliance for cold data

- Mechanical, hardware and storage software stack co-designed.
- Right-provisioned for cold data workload:
- Only 8% of the disks are spinning concurrently.
- Designed to store blobs which are infrequently accessed.

Pelican hardware

- 52U rack
- 1,152 archival class 3.5’’ SATA disks
- Disks are placed in trays of 16 disks
- Rack consists of six 8U chassis
- Each chassis contains 12 trays
- SATA disks in the tray connected to HBA on the tray
- Tray HBA’s connect to PCI switch on the backplane
- PCI supports virtual switching
- Each chassis connected to both servers
- 4.55 TB for each disk (sum up to 5 PB)
- Goodput of 100 MB/s for each disk (50 active disks required for 40 Gbps)

Resource domains

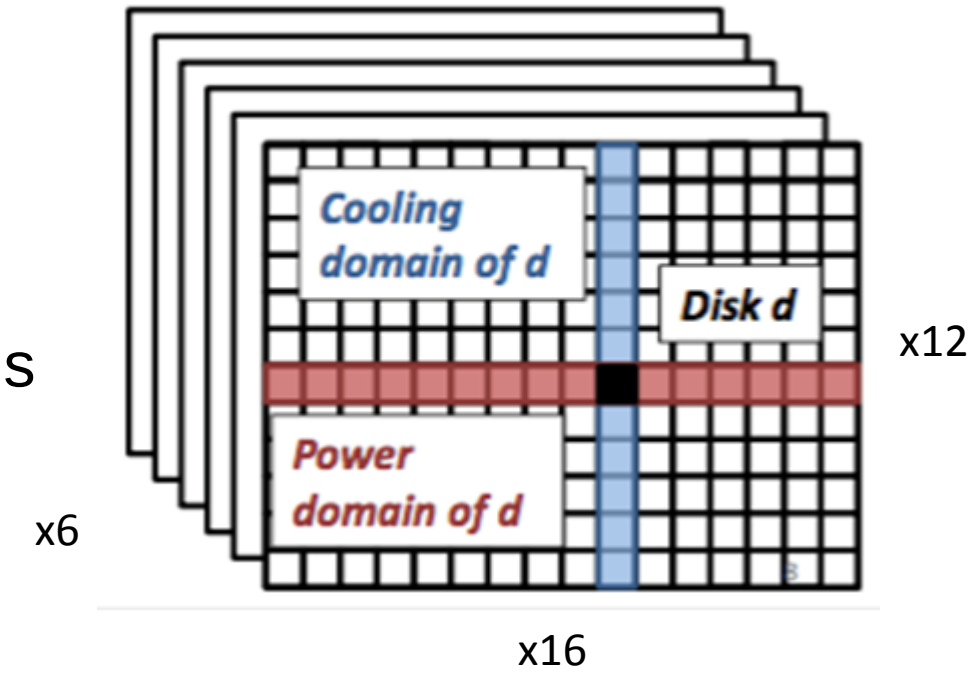
- Each domain is only provisioned to supply its resource to a subset of disks
- Each disk uses resources from a set of resource domains

- ***Domain-conflicting*** - Disks that are in the same resource domain.
- ***Domain-disjoint*** - Disks that share no common resource domains.

- Pelican domains (hard and soft limits)
  - Cooling, Power, Vibration, Bandwidth

Impact of right provisioning on resources

- Systems provisioned for peak performance:
  - Any disk can be active at any time
- Right provisioned system:
  - Disk part of a domain for each resource
  - Domain supplies limited resources
  - Disk active if enough resources in all its domains
- Pelican resource limitations:
  - 2 active out of 16 per power domain
  - 1 active out of 12 per cooling domain
  - 1 active out of 2 per vibration domain



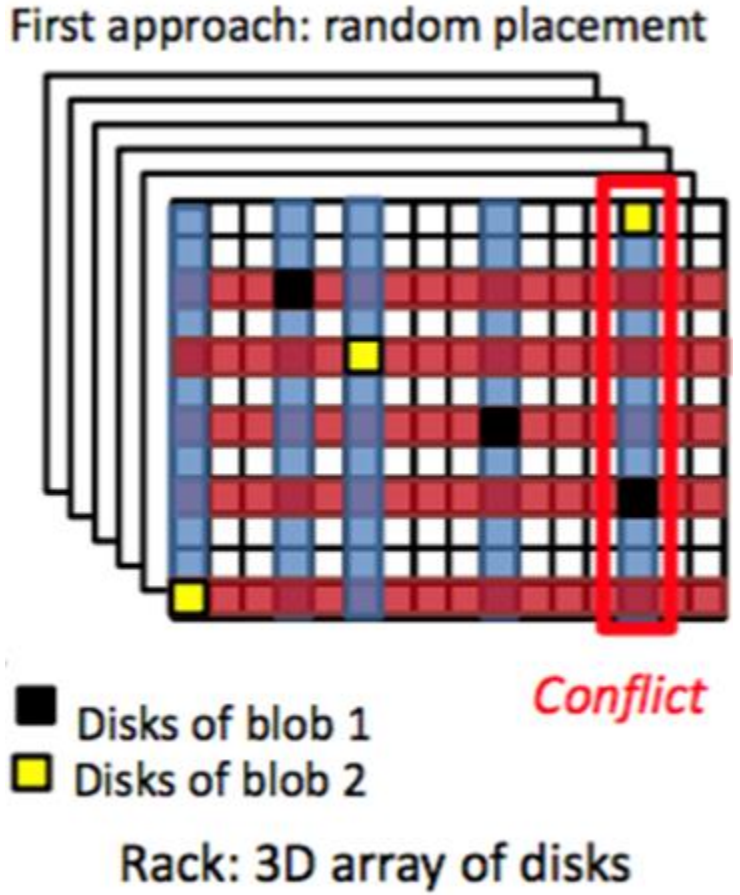
Data placement: maximizing request concurrency

In full-provisioned systems:

- Any two sets of disks can be active
- No impact of placement on concurrency

In right-provisioned systems:

- Sets can conflict in resource requirements
- Conflicting sets cannot be concurrently active
- Need to form sets, that minimize risk of a conflict.

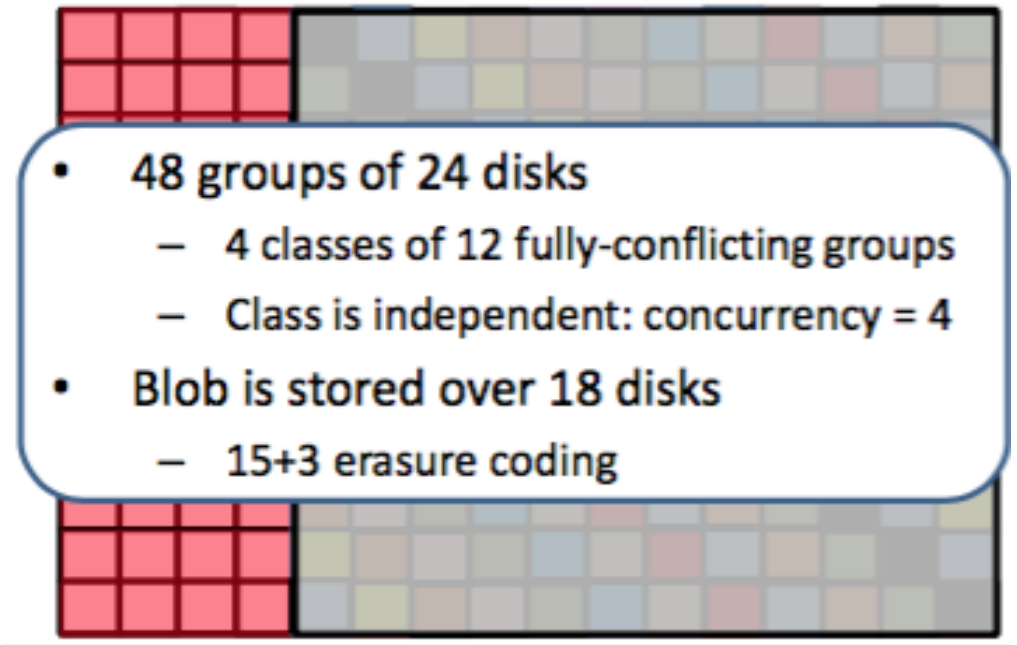


Data placement: maximizing request concurrency

- Blobs are erasure-encoded (Cauchy Reed-Solomon code) on a set of concurrently active disks.
- Blob of sizes from 200 MB to 1 TB as a key-value pair (key is a 20 byte identifier).
- Blob split into 128 kB data fragments. K+R fragments of a blob (stripe) on independent drives.
- Off-Rack metadata called “catalog” holds information about the blob placement
- Pelican: K=15, R=3:
  - Good data durability (three concurrent disk failures without data loss)
  - Saturating 10Gbps network link during single blob read
  - Possible to have 18 domain-disjoint disks

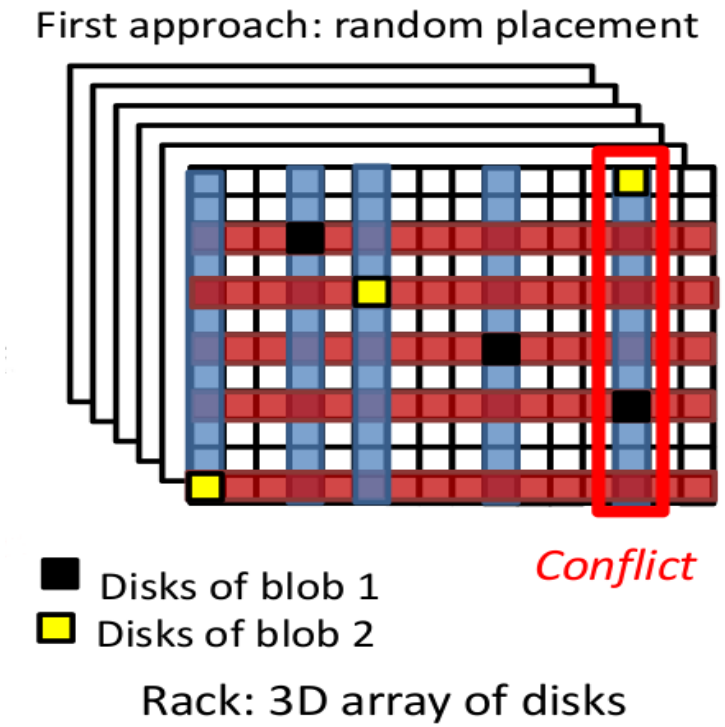
Data placement: maximizing request concurrency

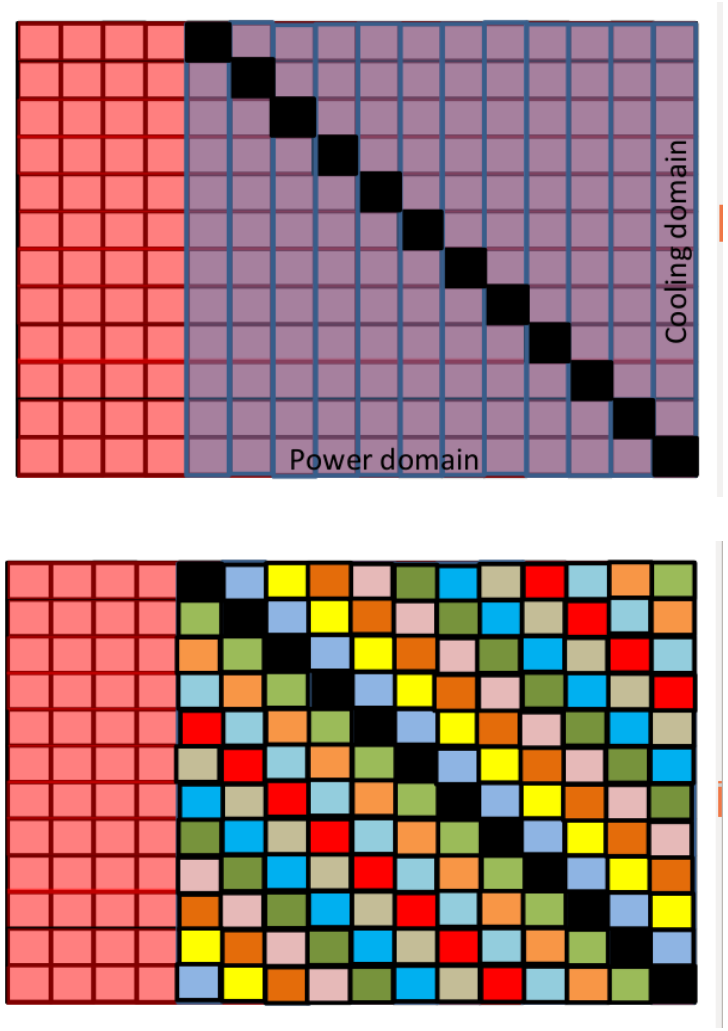
- Idea: concentrate all conflicts over a few set of disks
- Statically partition disk in groups in which disks can be concurrently active
- Property:
  - Either fully conflicting
  - Or fully independent



- Blob is entirely stored in one group
  - Probability of conflict proportional to the size of the group

Data placement: maximizing request concurrency



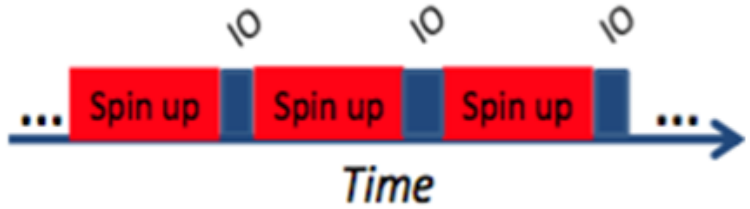


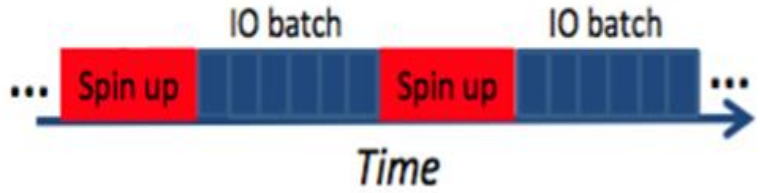
Groups: Pros

- All constraints are encapsulated in a group
- It defines the concurrency, that can be achieved while serving requests
- Span multiple failure domains
- Reduce time required to recover a failed disk

IO scheduling: "spin up is the new seek"

- Four independent schedulers
- Each scheduler: 12 groups, only one can be active
  - Naive scheduler: FIFO
    - Group activation time: 10-14 seconds
    - High probability of spin-ups after each request (92%)
    - Time is spent on doing spin-ups
  - Pelican scheduler: Reordering and rate limiting
    - Limit on maximum reordering
    - Trade-off between throughput and fairness
    - Weighted fair-share between client and rebuild traffic





Implementation

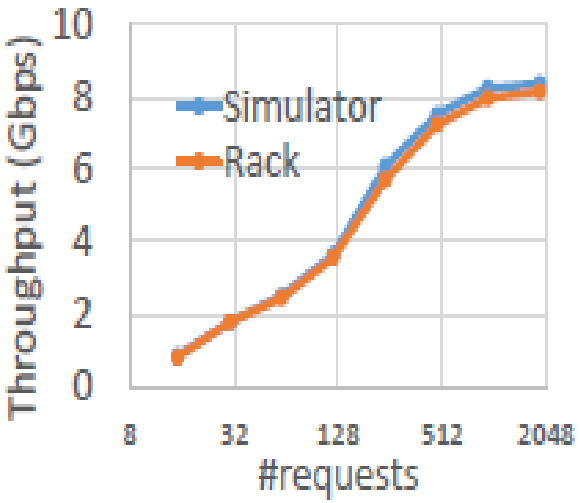
Problems:

1. Ensuring that we do not violate the cooling or power constraints until the OS has finished booting and the Pelican service is managing the disks
2. Unexpected spin ups of disks (Disks would be spun up without a request from Pelican, due to services like the SMART disk failure predictor polling the disk)
3. At boot, once the Pelican storage stack controls the drives, it needs to mount and check every drive

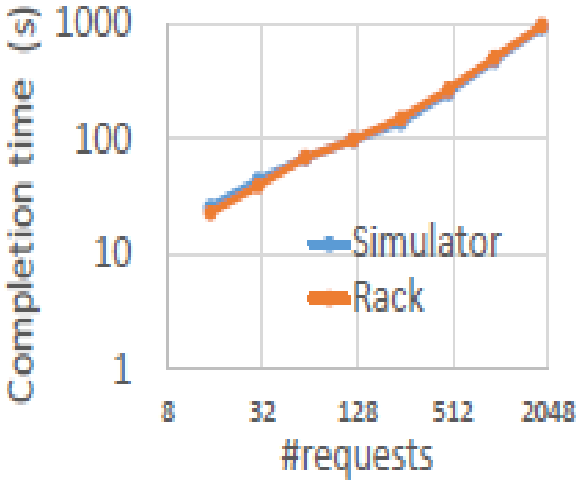
Solutions:

1. Using Power Up In Standby (PUIS) where the drives establish a PHY link on power up, but require an explicit SATA command to spin up
2. Added a special No Access flag to the Windows Server driver stack that causes all IOs issued to a disk marked to return with a “media not ready” error code
3. Generating an initialization request for each group and schedule these as the first operation performed on each group.

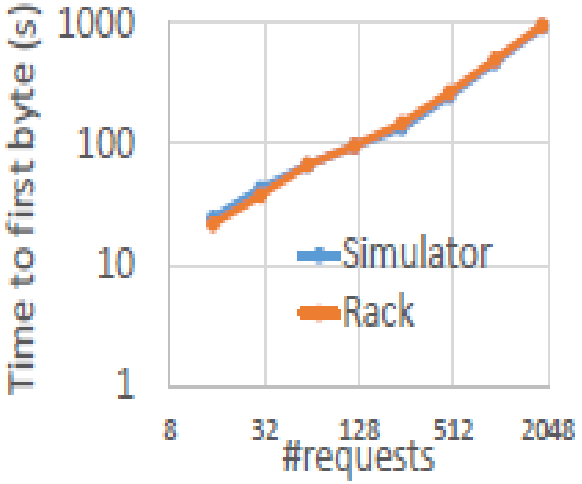
Pelican simulator



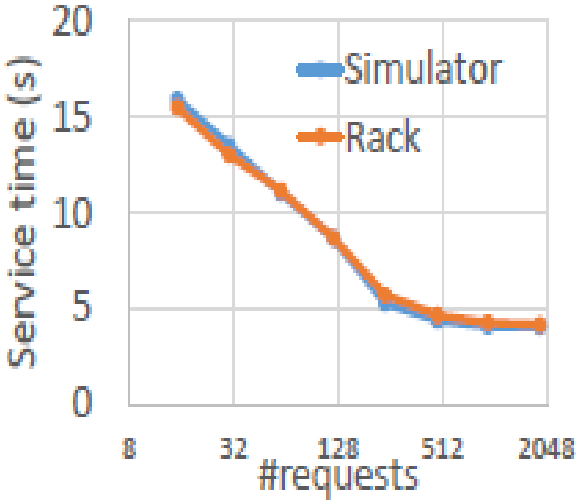
(a) Throughput



(b) Completion time



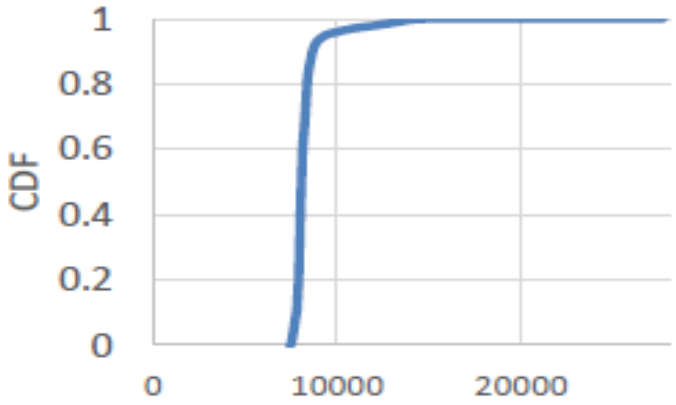
(c) Time to first byte



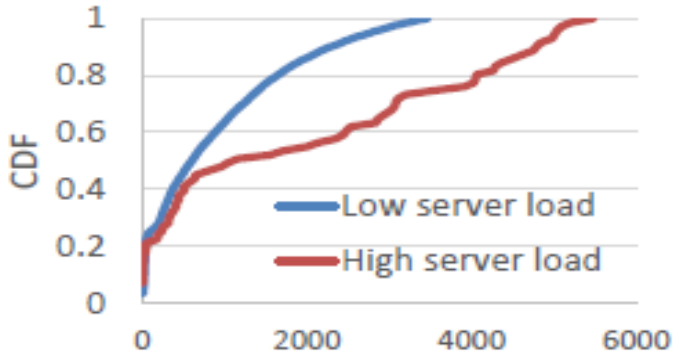
(d) Service time

- 48 groups of 24 disks
- Groups divided into 4 classes
- Blobs stored using 15+3 erasure encoding
- Maximum queue depth is 1000 requests
- Each server runs 2 schedulers and is configured with 20 Gbps

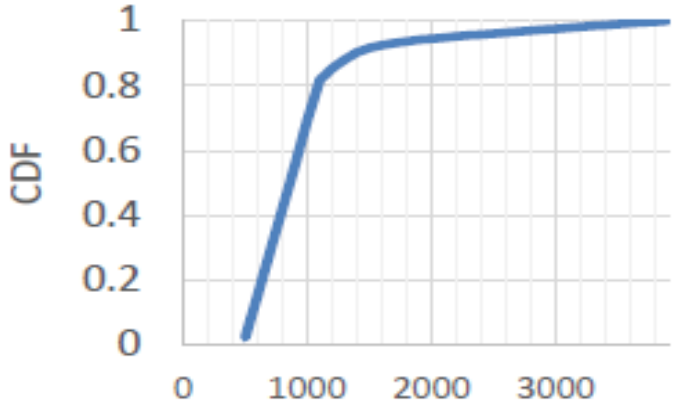
Parameters used in simulation



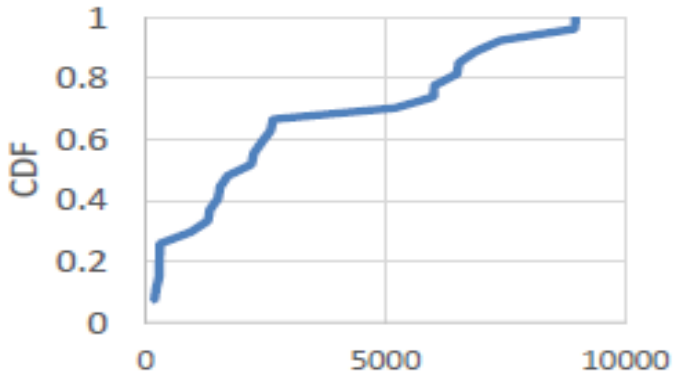
(a) Spin up delay (ms)



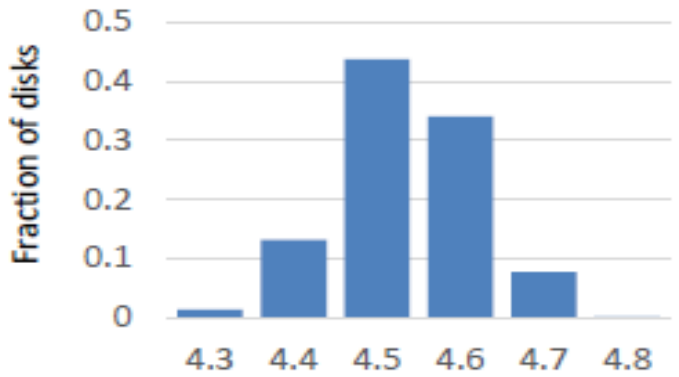
(b) Mount delay (ms)



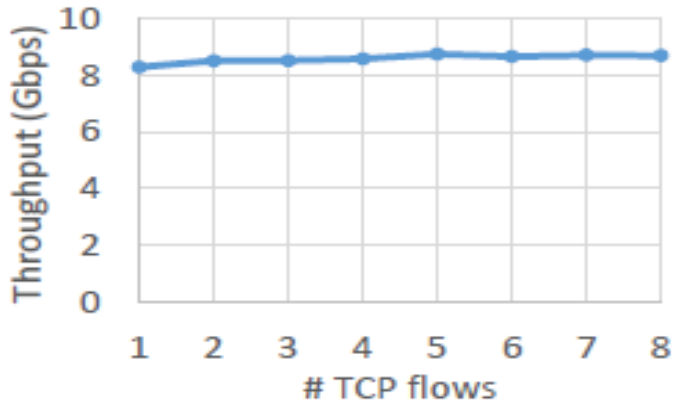
(c) Unmount delay (ms)



(d) Blob size (MB)



(e) Volume capacity (TB)

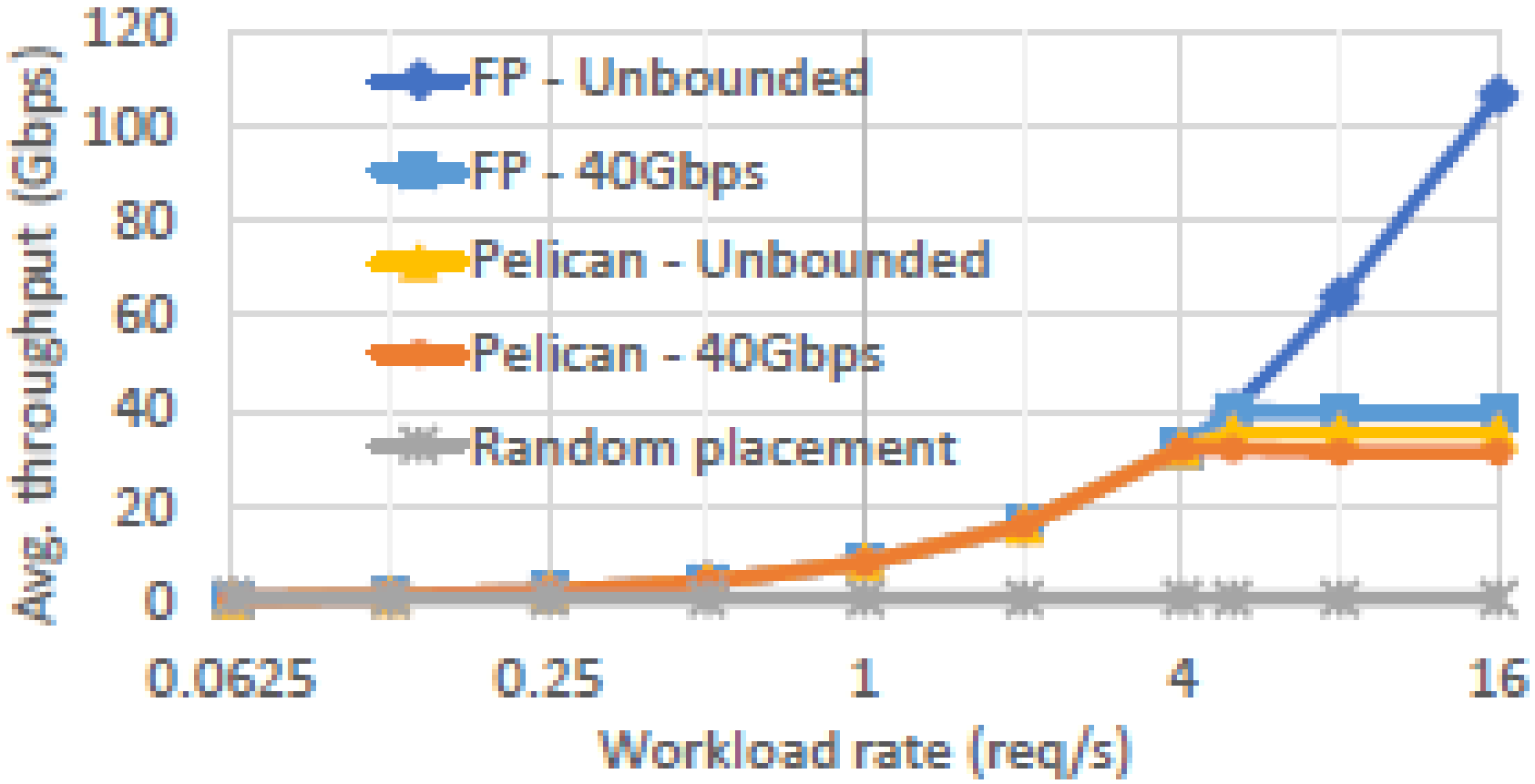


(f) NIC microbenchmark

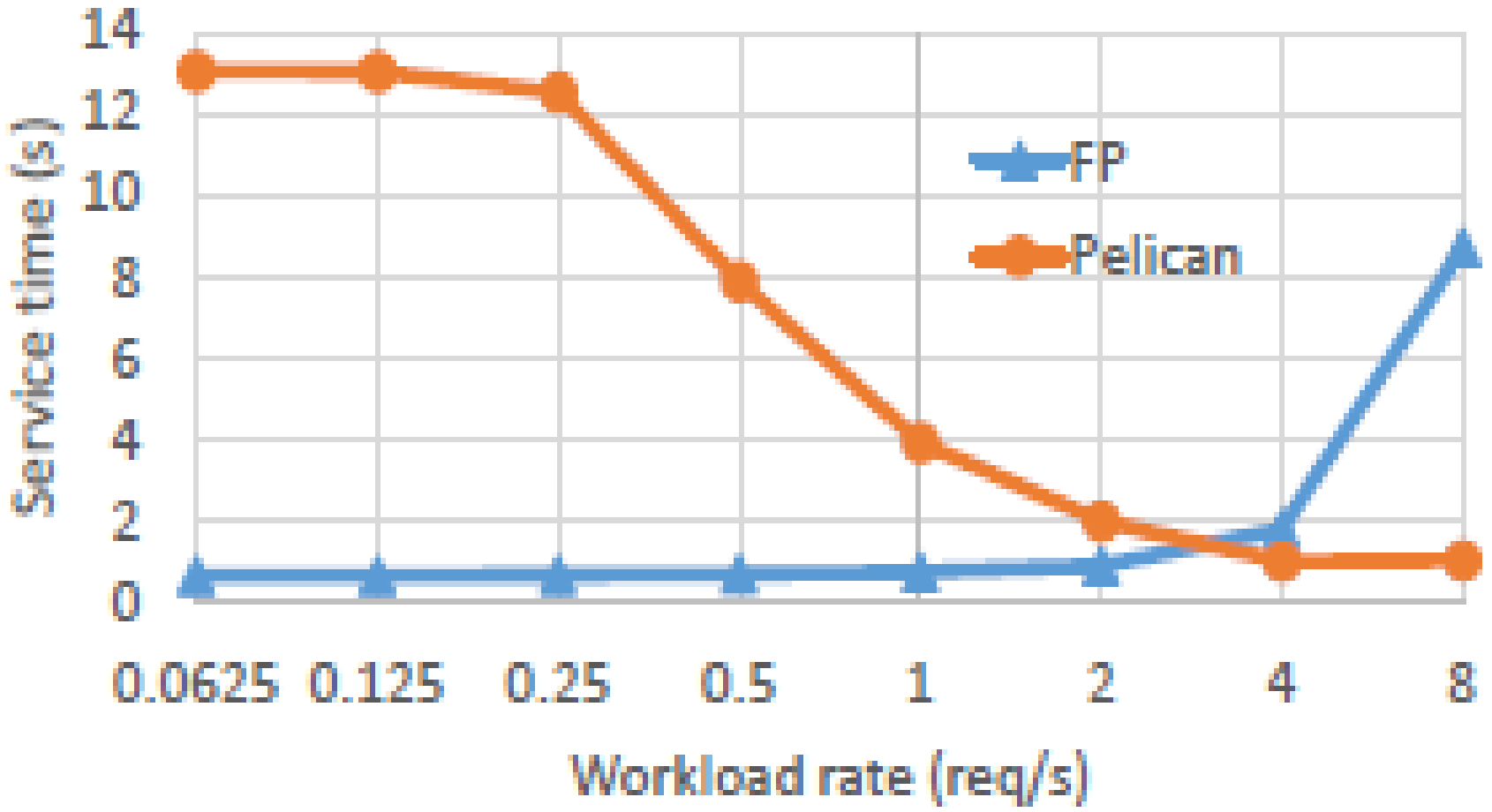
Pelican: Tests

- Pelican vs FP (all disks active)
- Workload:
  - Poisson proces with lambda from 0.125 to 16
- Metrics:
  - Throughput
  - Service time
  - Time to first byte
  - Etc.

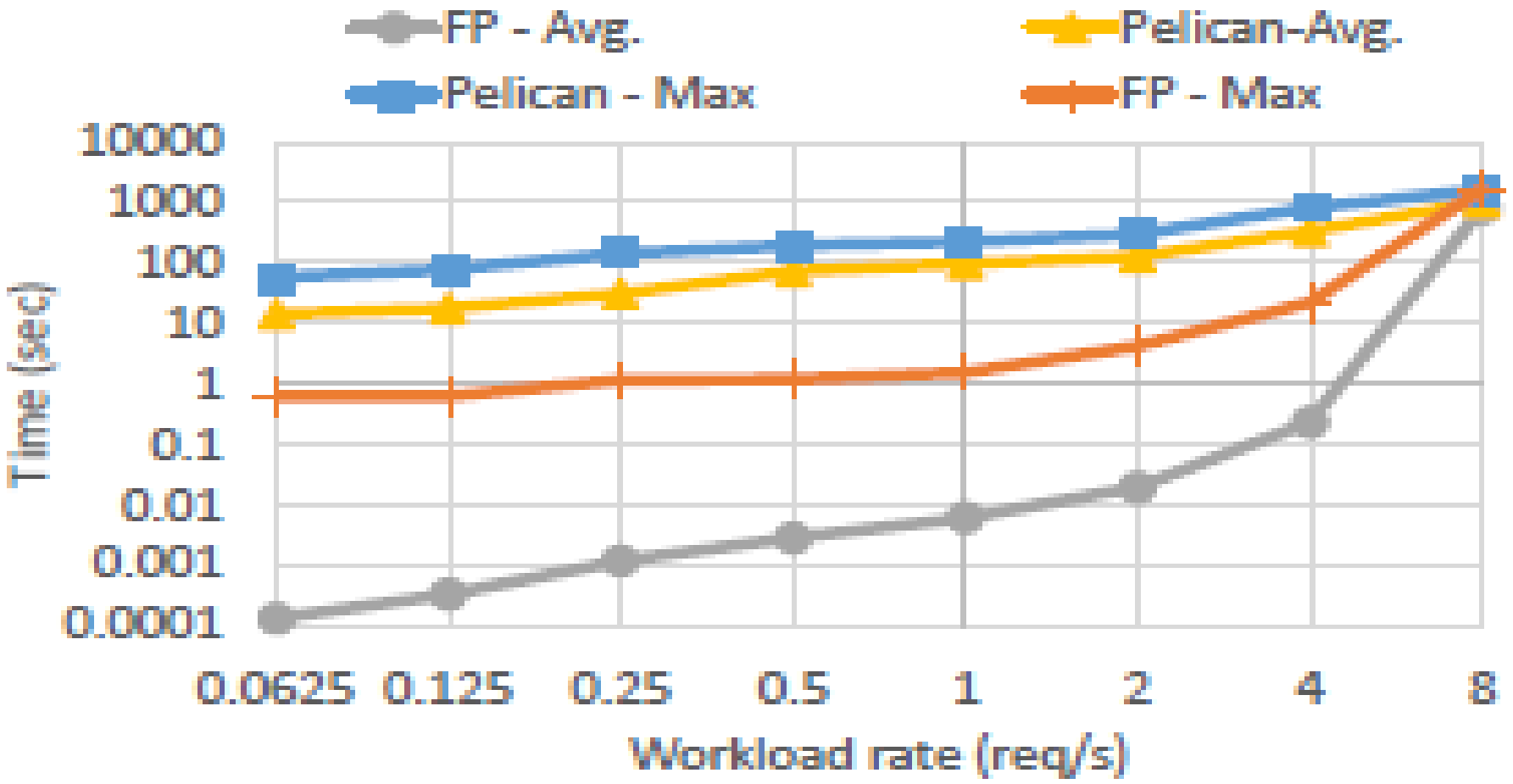
Throughput


Avg. throughput (Gbps)
120
100
80
60
40
20
0
FP - Unbounded
FP - 40Gbps
Pelican - Unbounded
Pelican - 40Gbps
Random placement
0.0625
0.25
1
4
16
Workload rate (req/s)

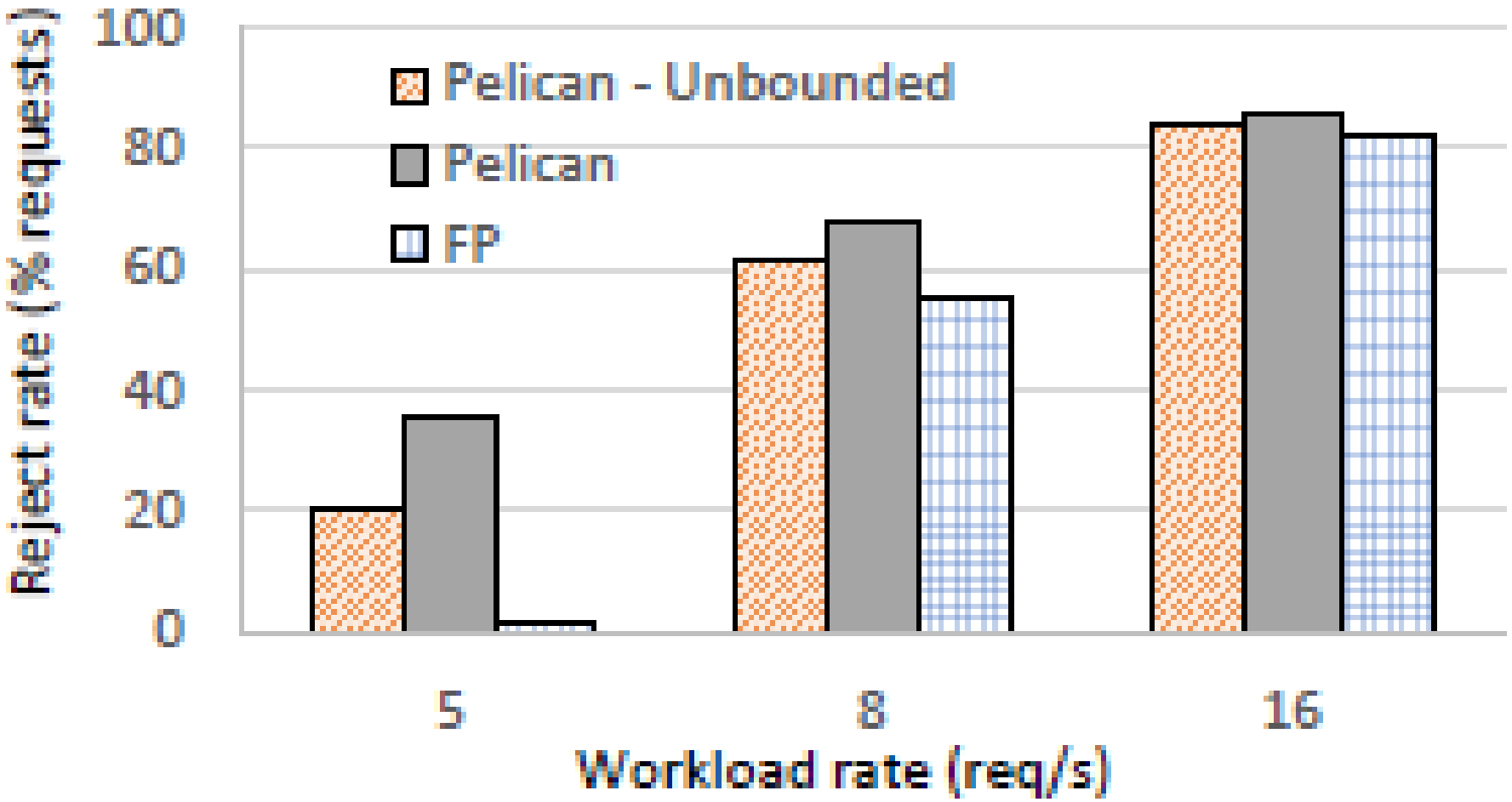
Service Time



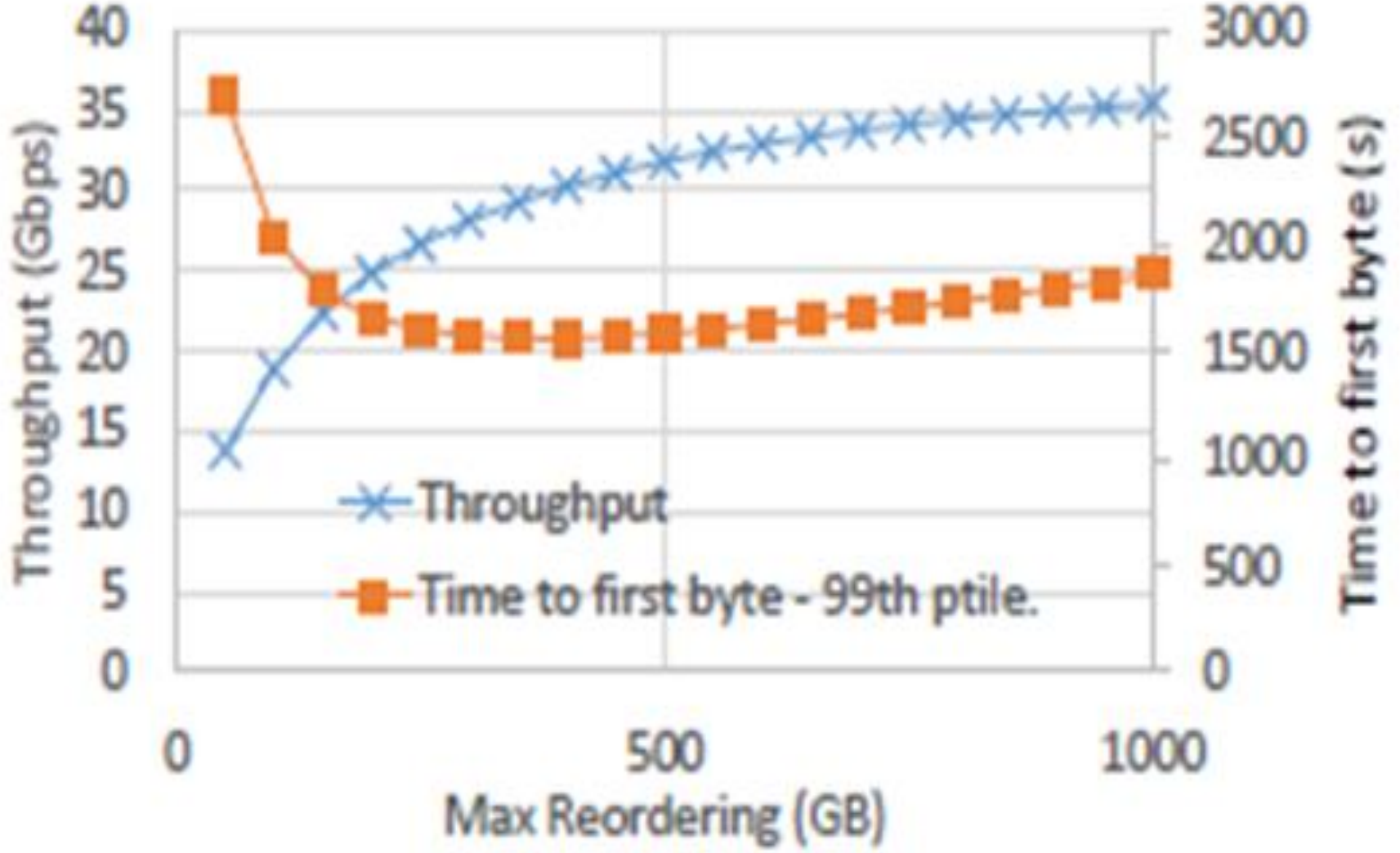
Time to first byte



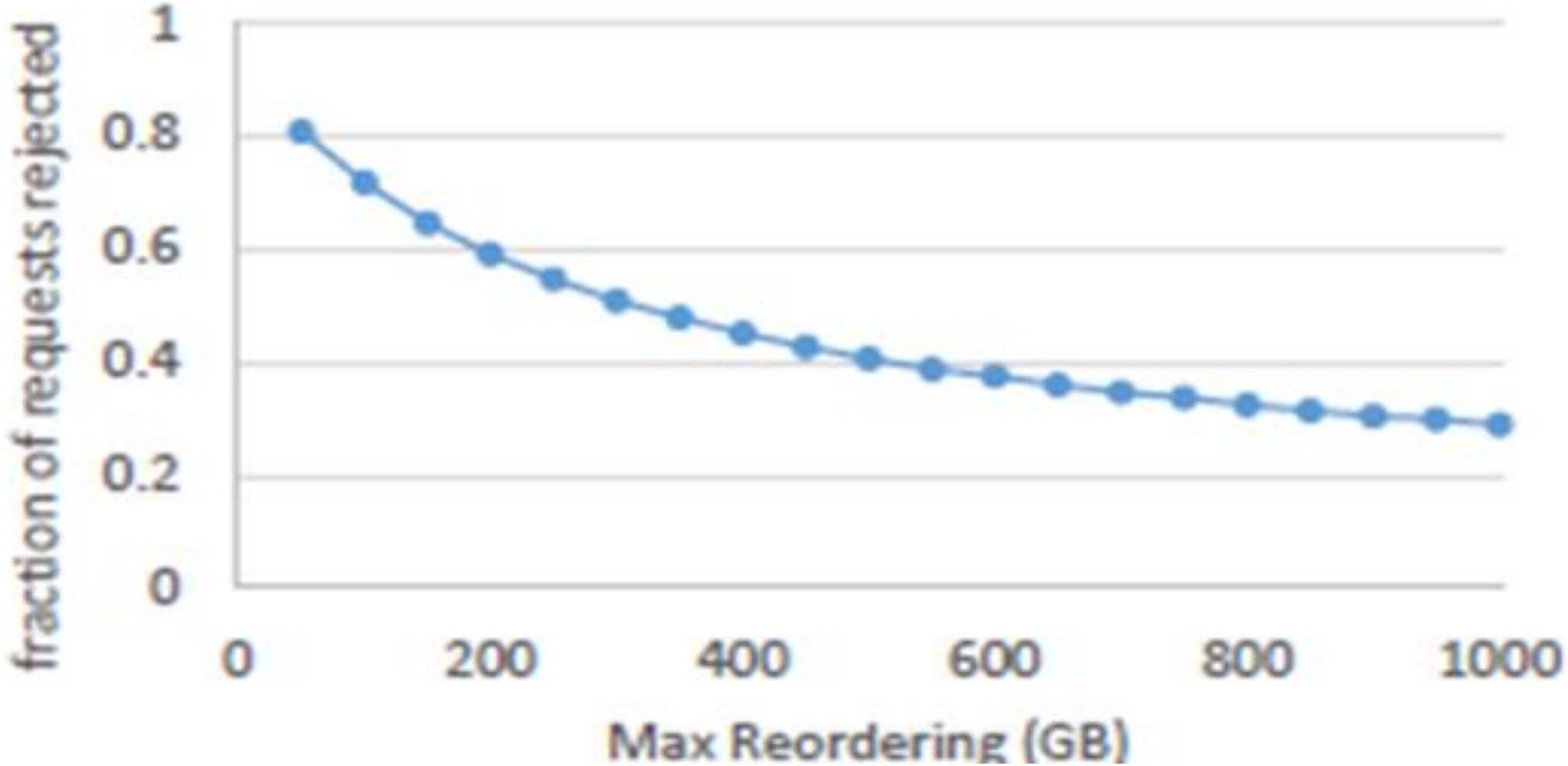
Reject rates



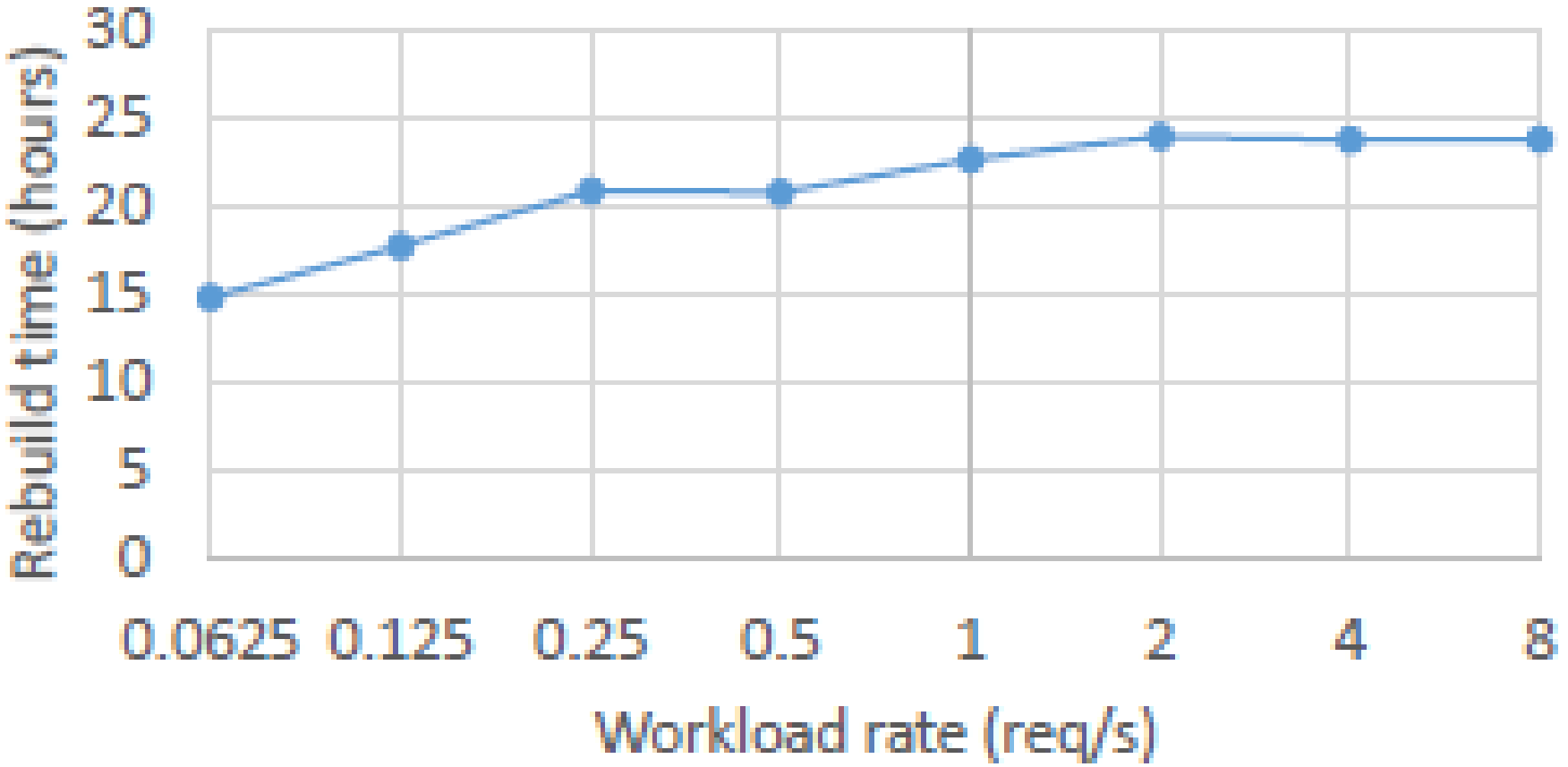
Cost of fairness - throughput



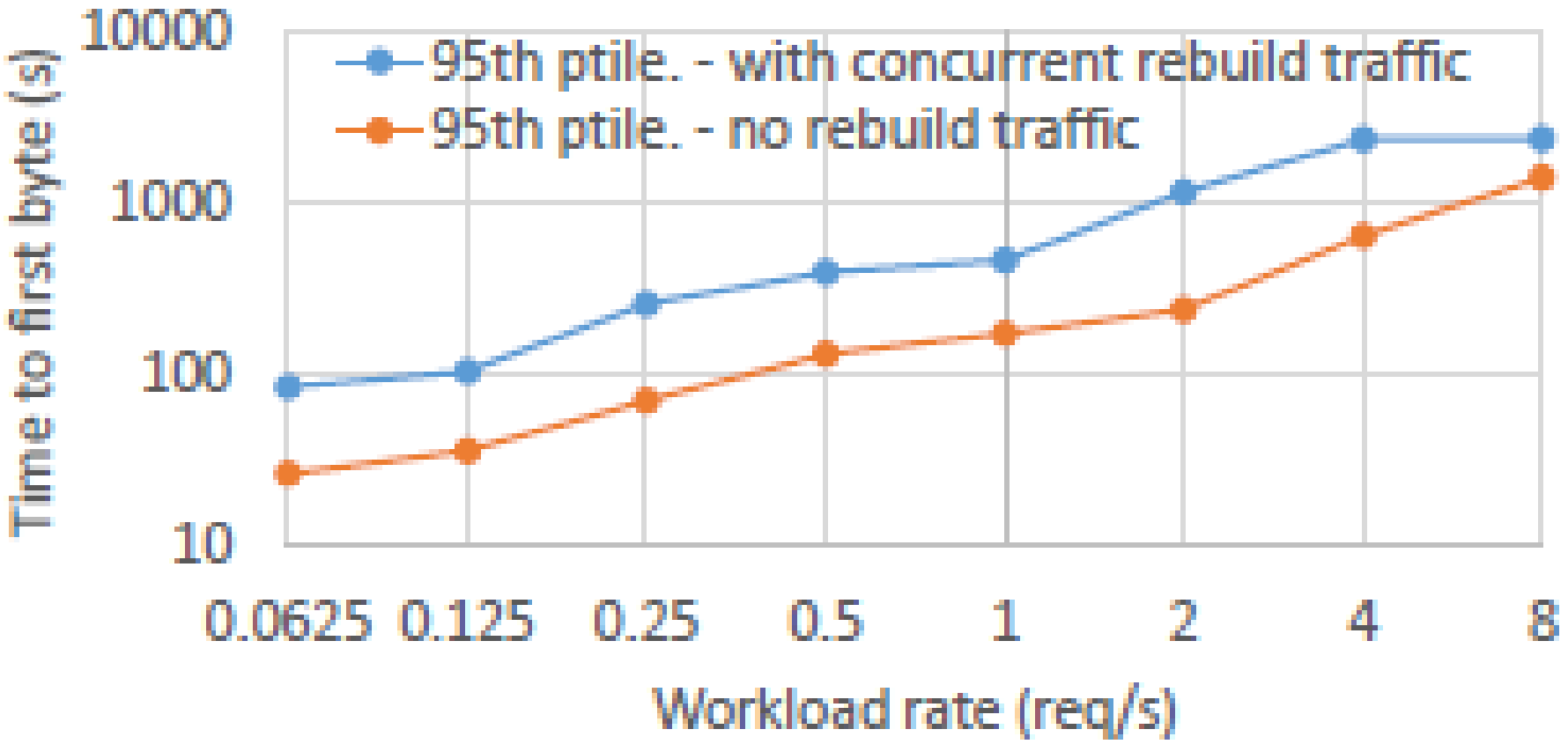
Cost of fairness – rejected requests



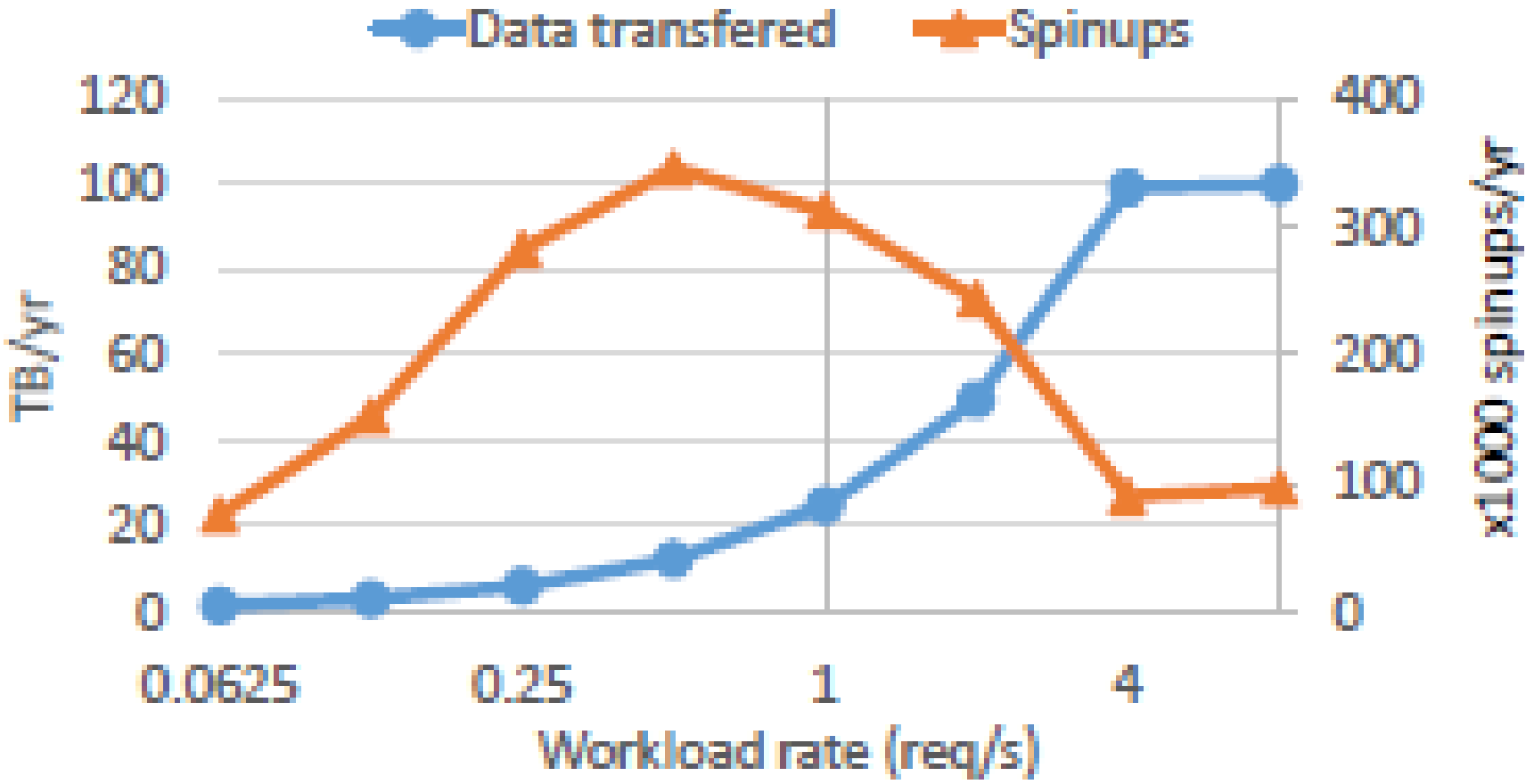
Scheduler performance – Rebuild time



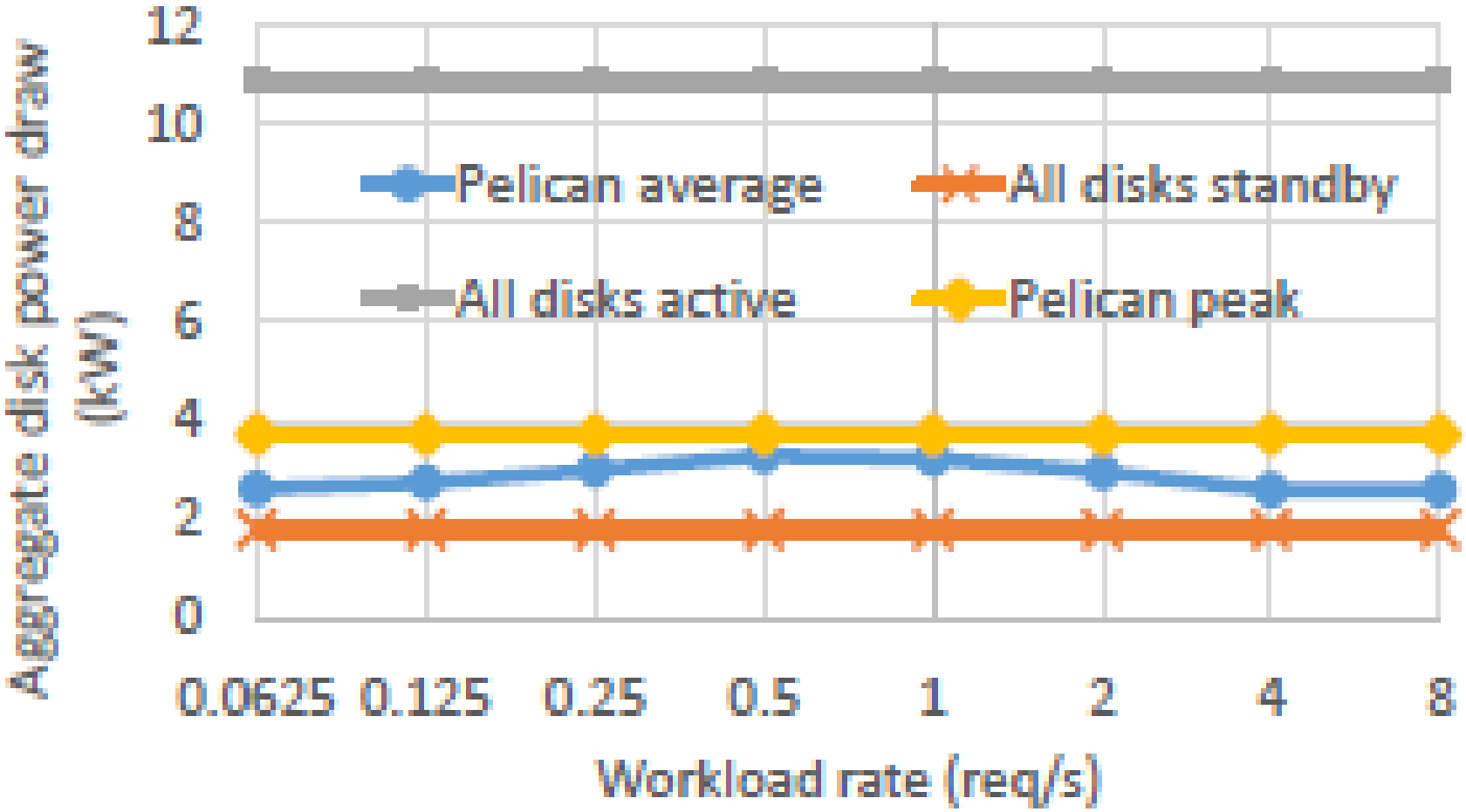
Scheduler performance – impact on client requests



Disk statistics



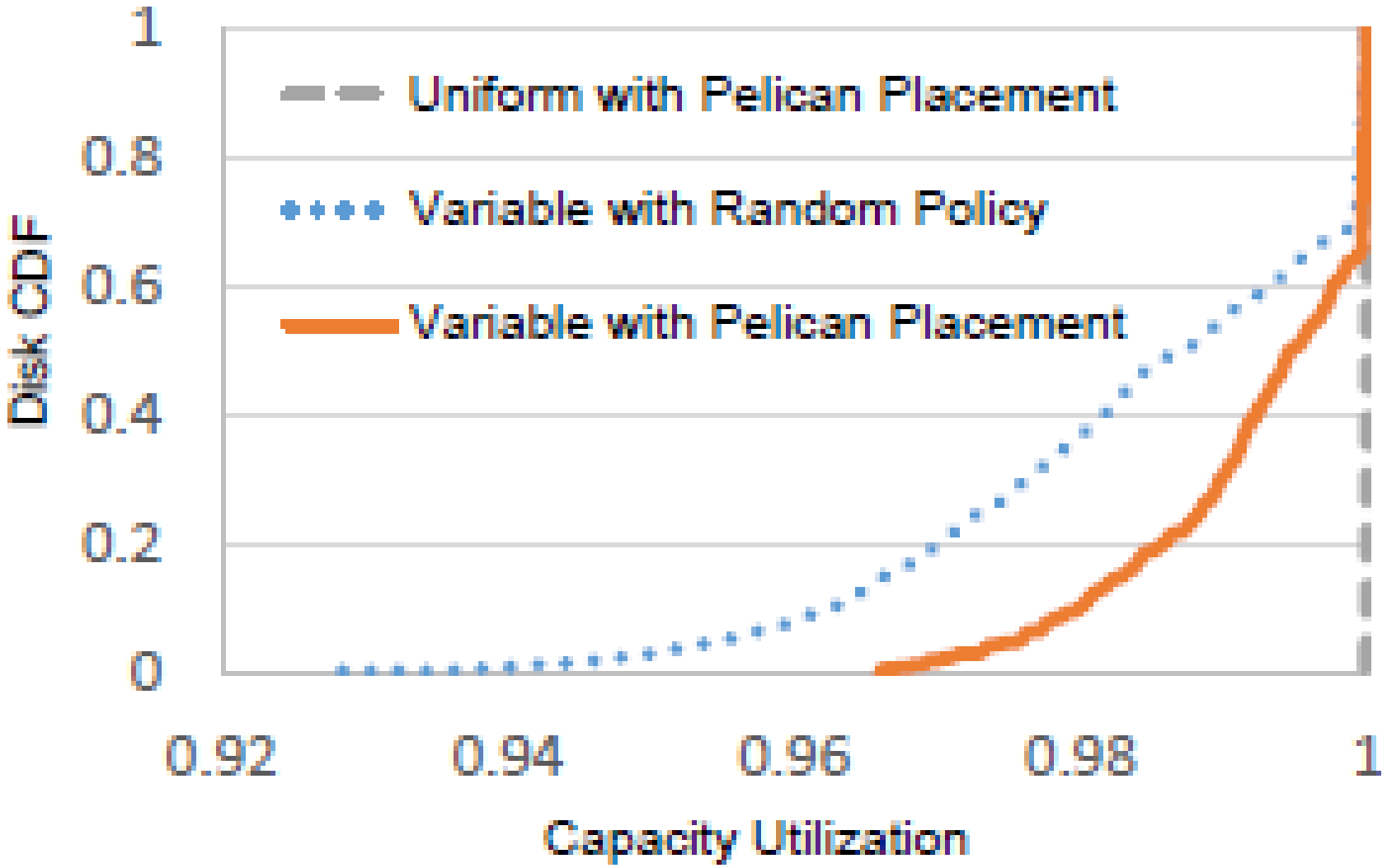
Rack power consumption



Disk states:

- Standby: 1.56 W
- Spinup: 21.53 W
- Active: 9.42 W

Disk utilization



Conclusions

- Rack-scale hardware/software co-design
  - Storage right-provisioned for cold data workload
  - Efficient constraint-aware software storage stack
- Prototype rack storing 5+ PB of raw data in 52U
- Pros and Cons
  - \+ Reduce capital cost and operating cost
  - \+ Meet requirements from cold data workload
  - \+ Erasure codes for fault tolerance
  - \- Sensitive to hardware changes
  - \- Tight constraints – less flexible to changes

Q&A