Unleashing the performance of multi-actuator drives

Paolo Valente
Linaro

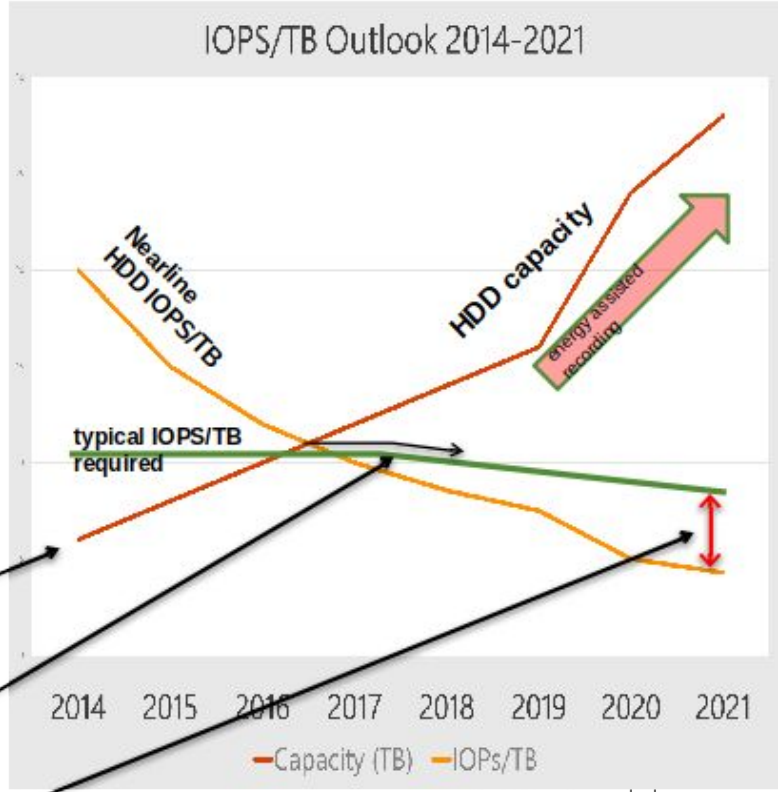
Tim Walker
Seagate Technology

Linaro Connect
VIRTUAL • FALL 2021

HDDs for Cloud Duty: Stranded Unusable Capacity

IOPS / TiB erosion limits usable capacity

- New recording technology is driving HDD capacity to 60TB+ per spindle
- Servo-mechanical capability does not scale with areal density → reducing IOPS/TB
- Latency driven workloads cannot utilize the capacity gains as IOPs/TB drops below minimum workload QOS
- To meet read latency QOS, customers can:
  - Replace HDD with flash
  - Short-stroke the HDD, leaving unused capacity stranded
  - Deploy lower capacity drives
    - ***All detrimental to storage TCO***



HDD capacity increasing

Data management advances trending minimum IOPS/TB requirements down

HDD IOPS/TB Performance Gap

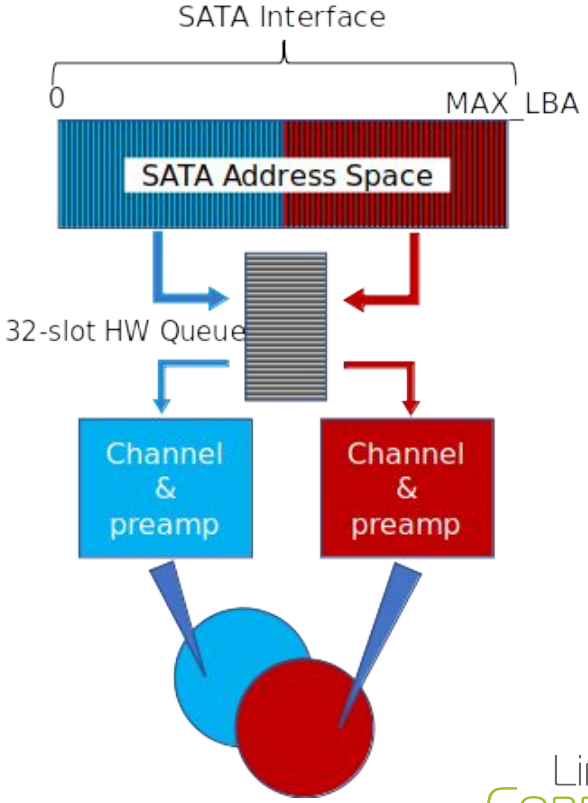
Single-namespace Split Actuator HDD

The split-actuator approach divides the disks into groups, each addressed by an independent actuator. A given sector is reachable by only one actuator - none of the address space is shared.

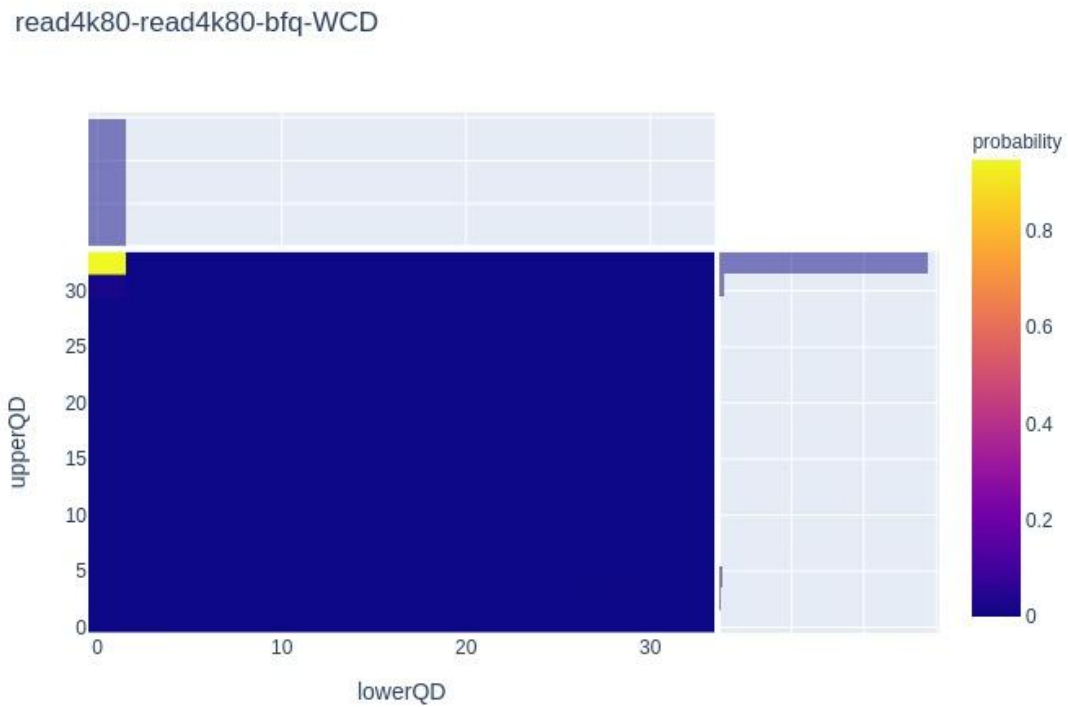
Seagate's SATA version contains two actuators and maps the lower half of the SATA LBA space to the lower actuator and the upper half to the upper. There are no changes to the IO protocol, except for a log page to report the LBA:Actuator mapping.

SATA allows 32 commands to be queued in the NCQ hardware queue. The 32-slot HW queue is a shared resource that services both actuators.

LIke any shared resource, we can either manage it, or it will manage us...



Yet, if fed without per-actuator control ...



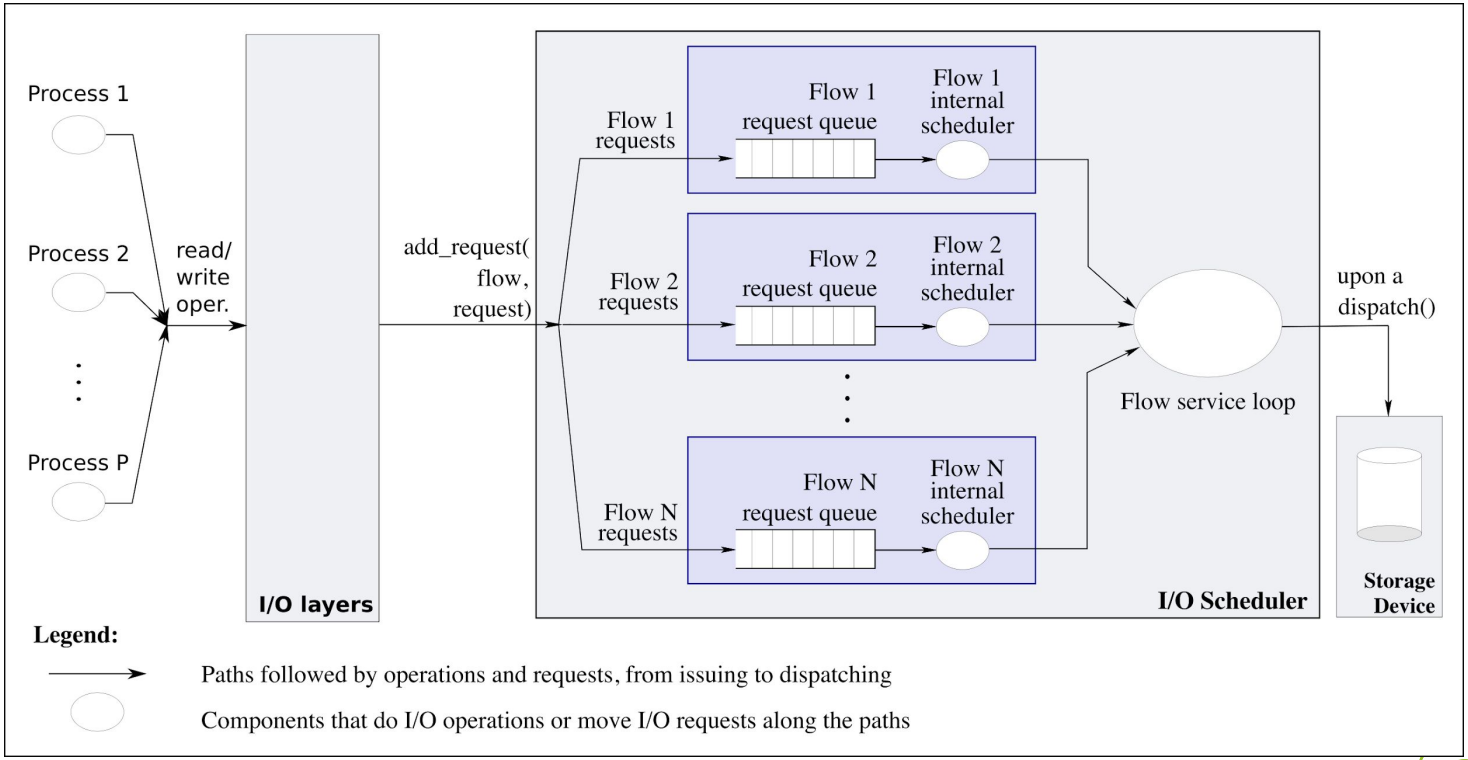
- Some actuator may remain underutilized or even idle
- This may limit maximum throughput drastically

- Need to control actuator load
- The I/O stack already contains a component for controlling I/O
  - I/O scheduler

I/O schedulers

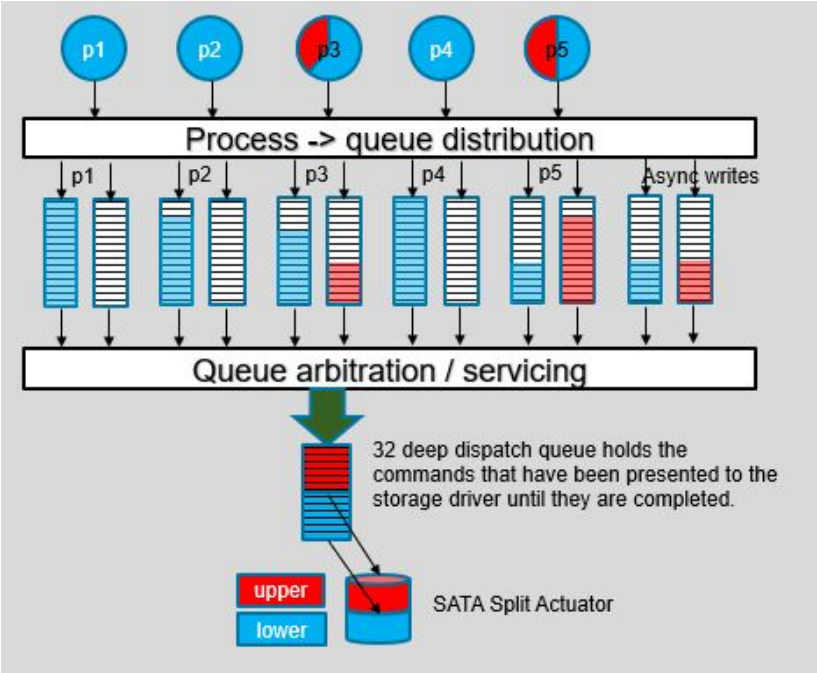
- Decide the order in which I/O requests are to be served
  - Order in which competing processes access storage
- So as to guarantee:
  - High I/O throughput
  - Low latency
  - High responsiveness - Low lag
  - Fairness
  - Other goals ...
- Important for our problem:
  - The scheduler should provide a good ground for implementing flexible and accurate control on per-actuator load
- Available schedulers: *none*, *mq-deadline*, *kyber*, *BFQ*
- The BFQ I/O scheduler does have a rich infrastructure

BFQ infrastructure

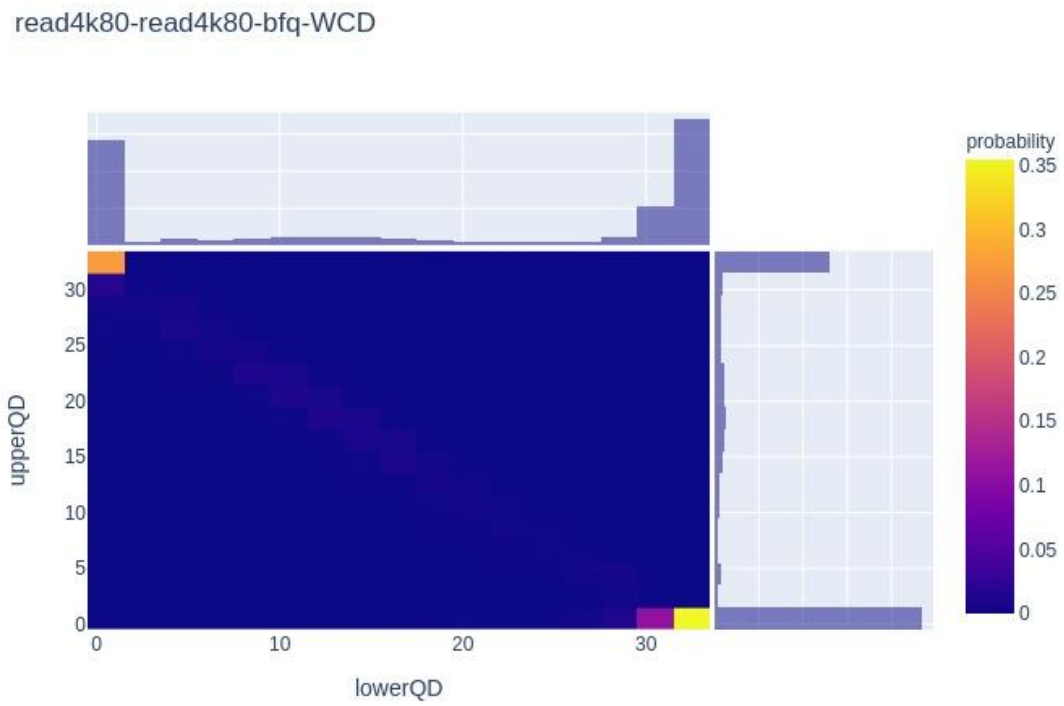


Base idea: split queues

- BFQ considers the I/O of each process as a separate I/O flow
  - And enqueues it in a separate queue
- Idea: just split each queue into one queue for each actuator
- So, each process will be associated with N separate queues, one for each actuator

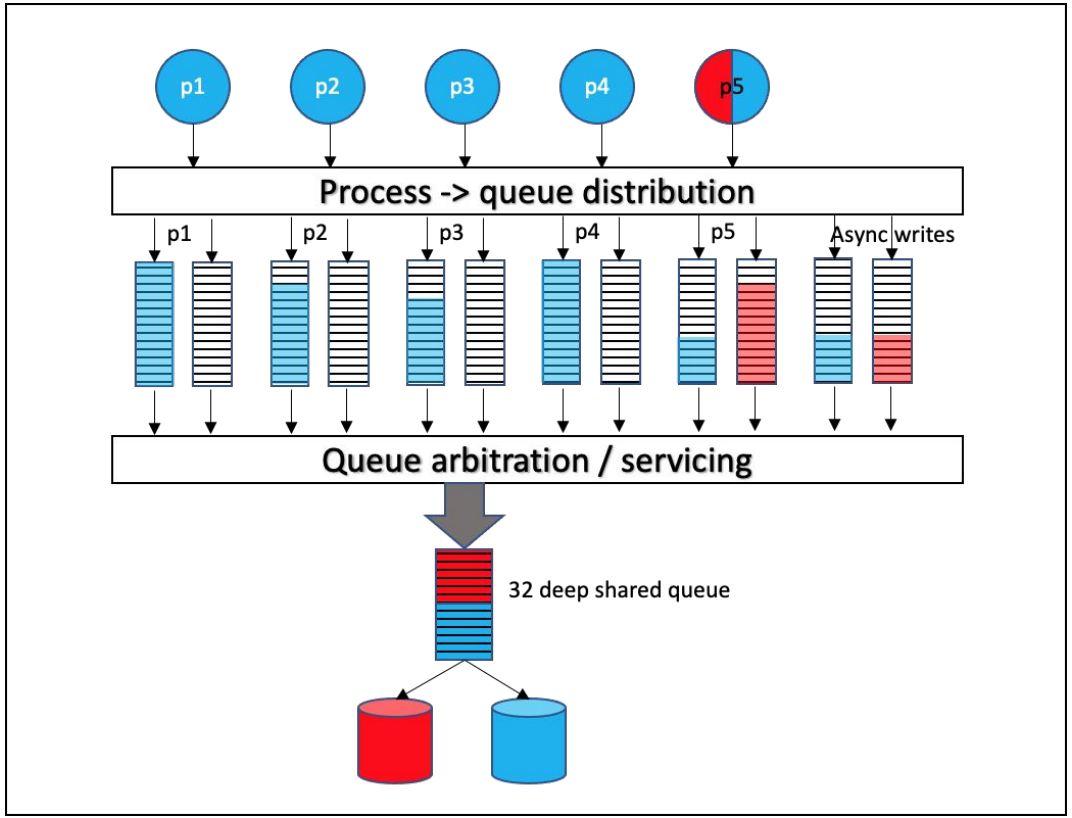


First improvement



- Now two opposite states are equiprobable
- Still, either only the upper or only the lower actuator is busy
- Because BFQ serves queues containing sequential I/O one at a time, and for a long time each
  - This service scheme maximises throughput with a single actuator
  - But not with two actuators ...

General problem



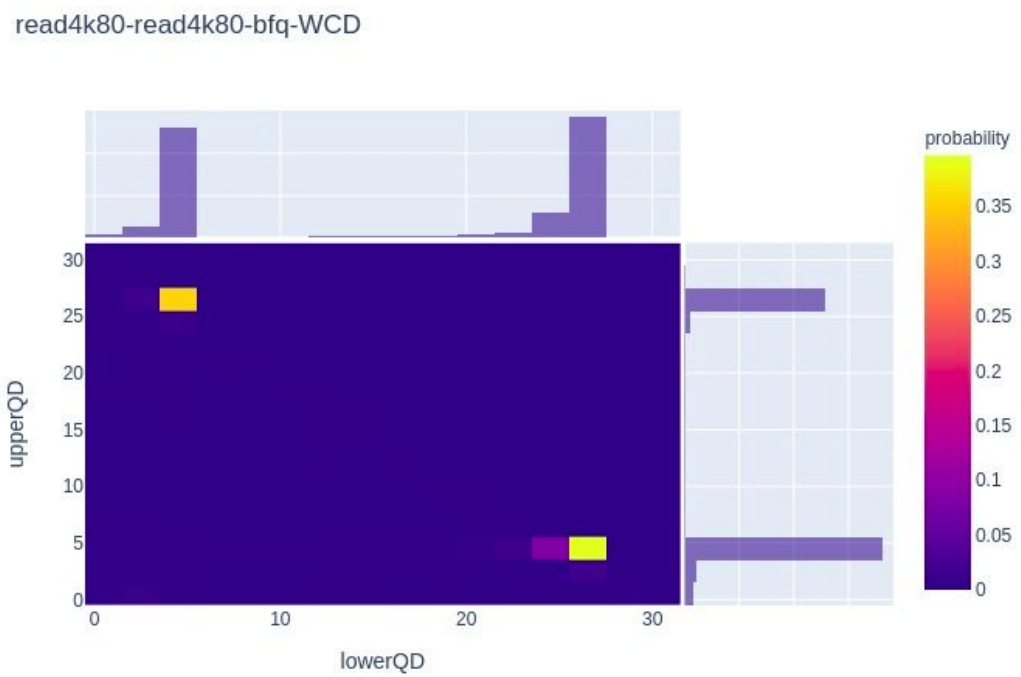
- There may be cases where some actuator is even more severely underutilized:
  - Many queues contain I/O for a given actuator, while few queues contain I/O for other actuators
  - One queue, containing I/O for a given actuator, has a much higher weight than queues that contain I/O for other actuators

General solution: inject I/O

1. Find a per-actuator load threshold (in number of I/Os) such that, if the load of an actuator is close to that threshold, but not higher than that that threshold
   a. the actuator is busy enough to be fully or almost fully utilized
   b. so few slots are *stolen* to the other actuators, in the drive's internal queue, that there is no or almost no utilization loss for the other actuators
2. When the I/O stack asks for a new I/O to dispatch, check whether there is some actuator with a load below the threshold. If there, then check whether the queue currently in service contains I/O for that actuator
   a. If it does, then just dispatch the head I/O of the in-service queue
   b. If it does not, then check whether there is some other queue containing I/O for that actuator. If there is, then dispatch the head I/O of that queue, instead of the head I/O of that in-service queue. If there is not, then just dispatch the head I/O of the in-service queue

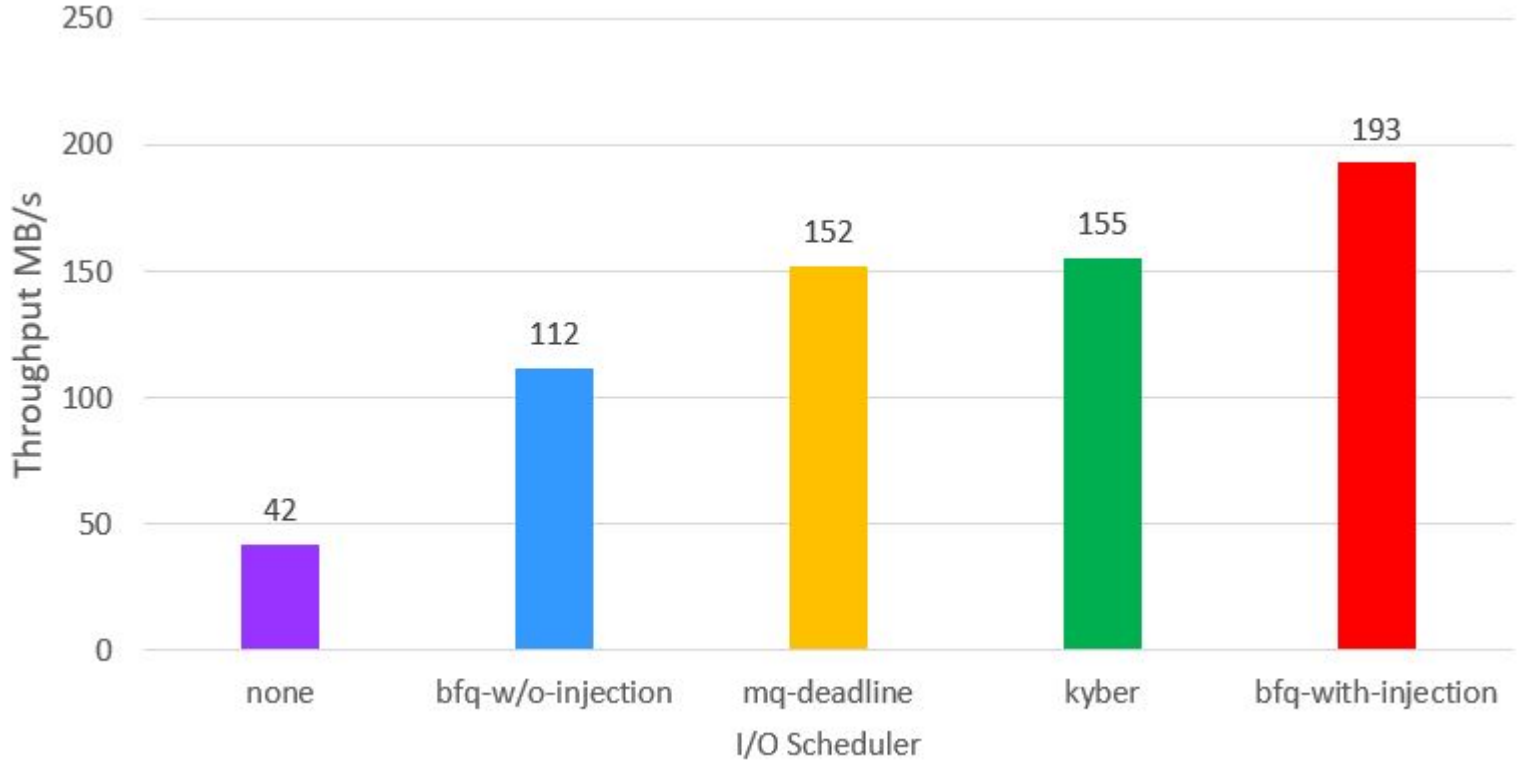
- We are currently using 4 as a threshold

Results: minimum per-actuator utilization



- While an actuator is highly utilized, the load of the other is still around the threshold
- And viceversa

Results: throughput



Discussion

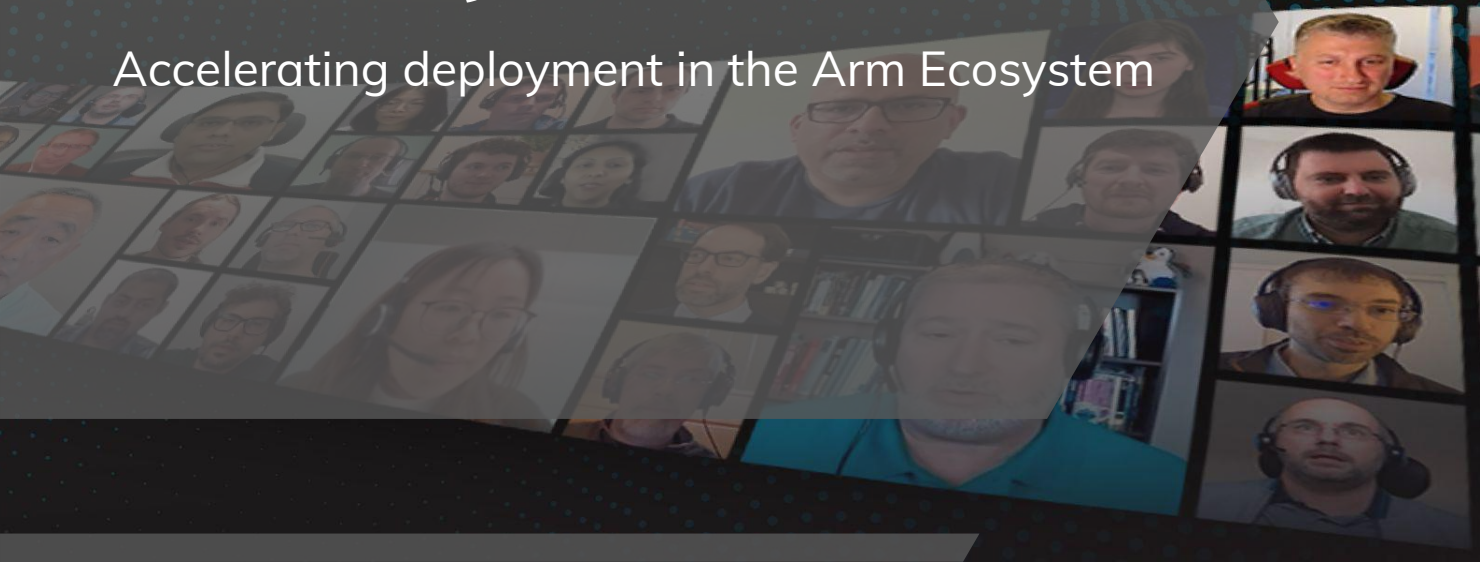
- This is still a preliminary contribution
- Results shown in this presentation cover just one workload
- No production code available yet

Open issues:

- This injection mechanism does not take bandwidth distribution into account
  - A queue may receive an amount of extra bandwidth not related to its weight
- What is the best value for the injection threshold?
  - Most certainly, it depends on the workload
- Since the best threshold value depends on the workload,
  - Maybe the threshold should be updated dynamically as a function of the workload
  - Or maybe a good, static value could be found that guarantees good performance with all workloads
    - This option would reduce complexity

Thank you

Accelerating deployment in the Arm Ecosystem



Linaro Connect
VIRTUAL • FALL 2021