Science and Technology Facilities Council

Enhancing XRootD Load Balancing for High-Throughput transfers

Jyothish Thomas, James Walder, Thomas Byrne, Guilherme Amadio

jyothish.thomas@stfc.ac.uk, james.walder@stfc.ac.uk, tom.byrne@stfc.ac.uk

Science and Technology Facilities Council

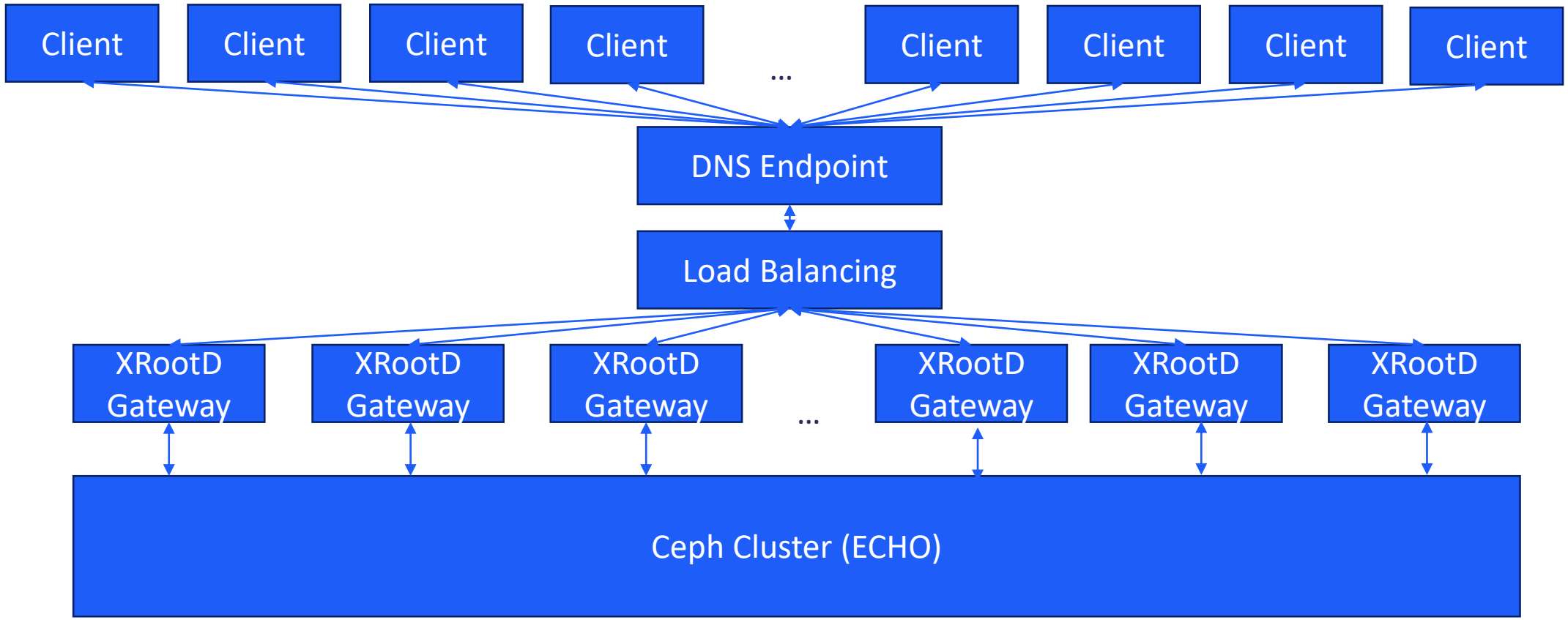
The RAL Tier-1 Setup

The RAL Tier 1 disk storage setup

- RAL has a large Ceph object store disk pool (Echo)
- Echo is accessed through a set of XRootD server gateways
- Clients contact a single DNS endpoint, behind which is a pair of XRootD redirectors that balance the load between the servers
- Servers are equivalent in functionality, i.e, all servers access the same storage backend, hence the load balancing only needs to consider the load on the server
- Servers have a constant stream of traffic, averaging 5GB/s between Aug-Sep 2024, with some periods of higher traffic.

Science and Technology Facilities Council

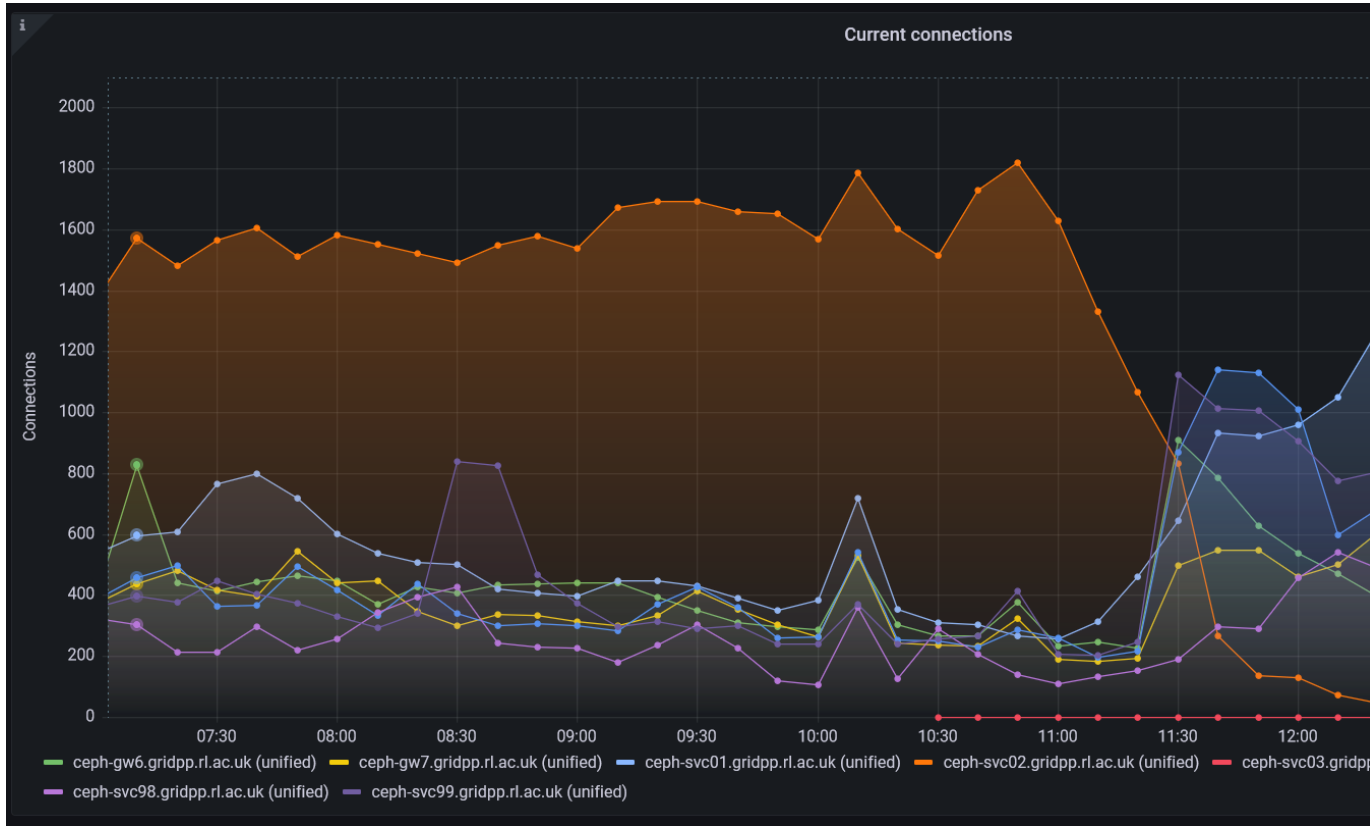
The RAL tier 1 setup



Science and Technology Facilities Council

DNS Round Robin limitations

- Relying on clients (and DNS servers) to choose randomly (and often) for an even distribution
- Not actually loadbalancing
  - Slower gateways will end up with more active transfers
  - No backoff mechanism to allow overloaded gateway to recover
- (RAL specific) No direct control over our DNS, requests via central IT helpdesk made hardware interventions very time consuming



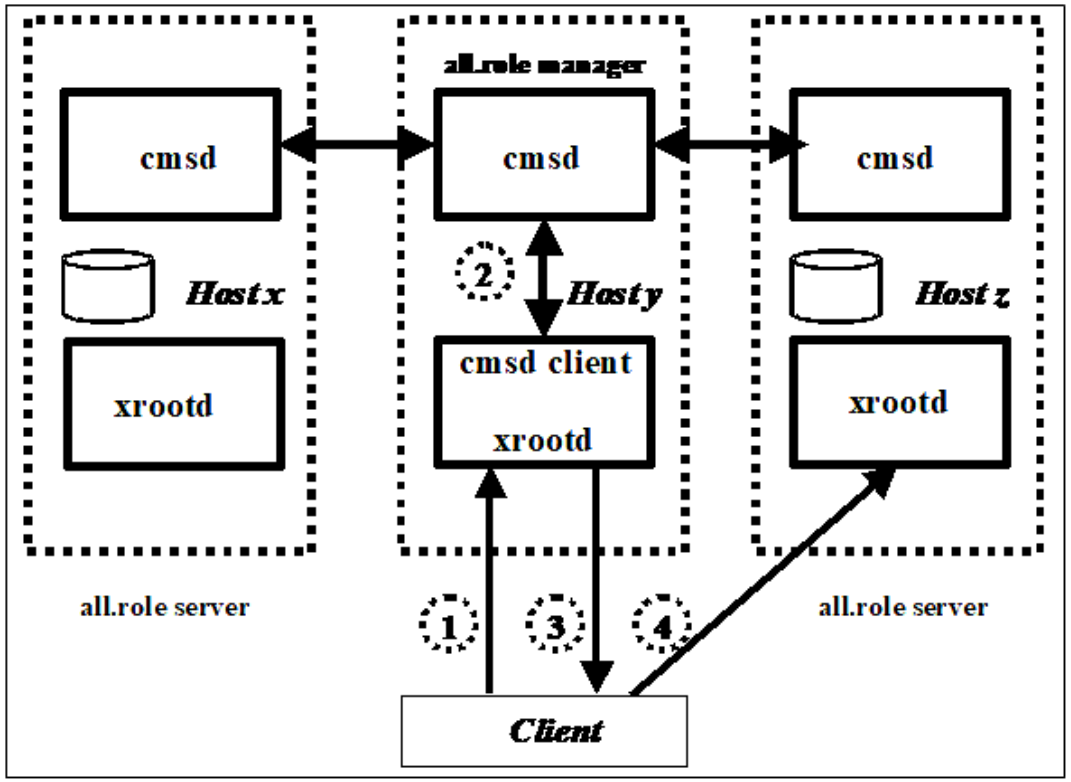
Occurrences of gateway hotspotting tracked down to DNS server configuration at another site

Science and Technology Facilities Council

XRootD Cluster Management

XRootD Cluster Management

- XRootD has a native Cluster Management Service Daemon (CMSD)
- This adds an XRootD manager that manages a cluster of XRootD servers
- The manager has a load balancing component that can distribute load according to a user defined weighted sum of server reported metrics (io,sysload, memory load,etc..)

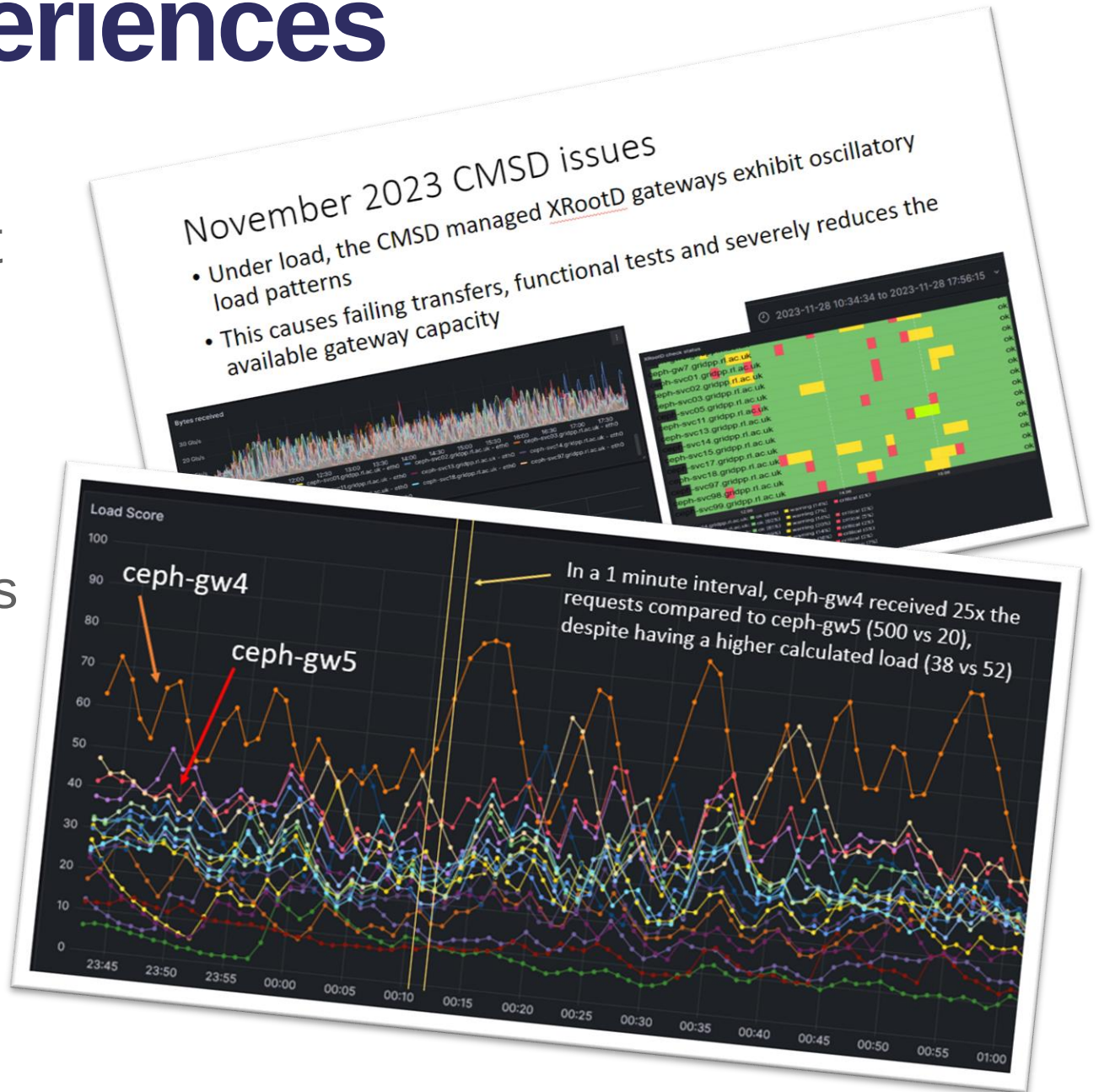


XRootD Cluster Management Service Configuration Reference (5.7)

Science and Technology Facilities Council

XRootD CMSD experiences

- Although a major improvement compared to DNS RR, we observed some odd behaviours at high load
  - Load and connection oscillation
  - Unexpected placement decisions
- Investigations lead to tuning which helped, but only to a point.

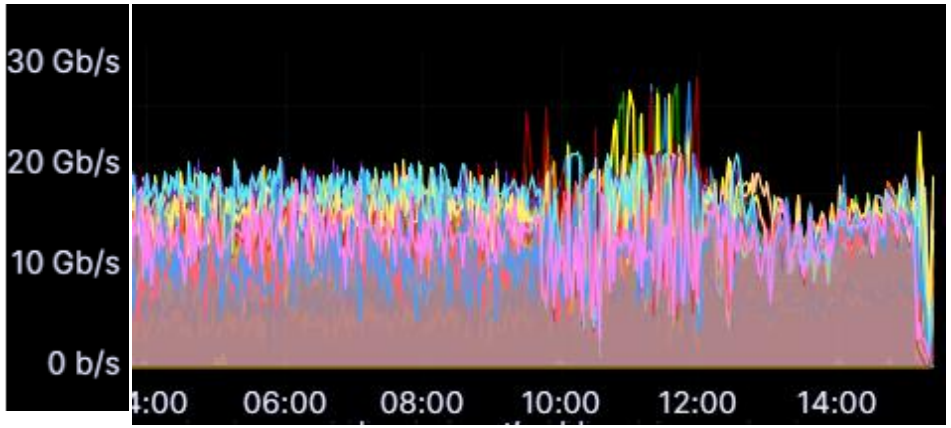


Science and Technology Facilities Council

Observations under high load

- The CMSD load balancing algorithm assigns the incoming transfers to the least loaded server and any servers with a load within the configured fuzz value
- This works well under normal conditions but when all servers are at close to full capacity, it results in server instability as the load keeps moving between subsets of servers

XRootD load balancing to RR (bytes received)

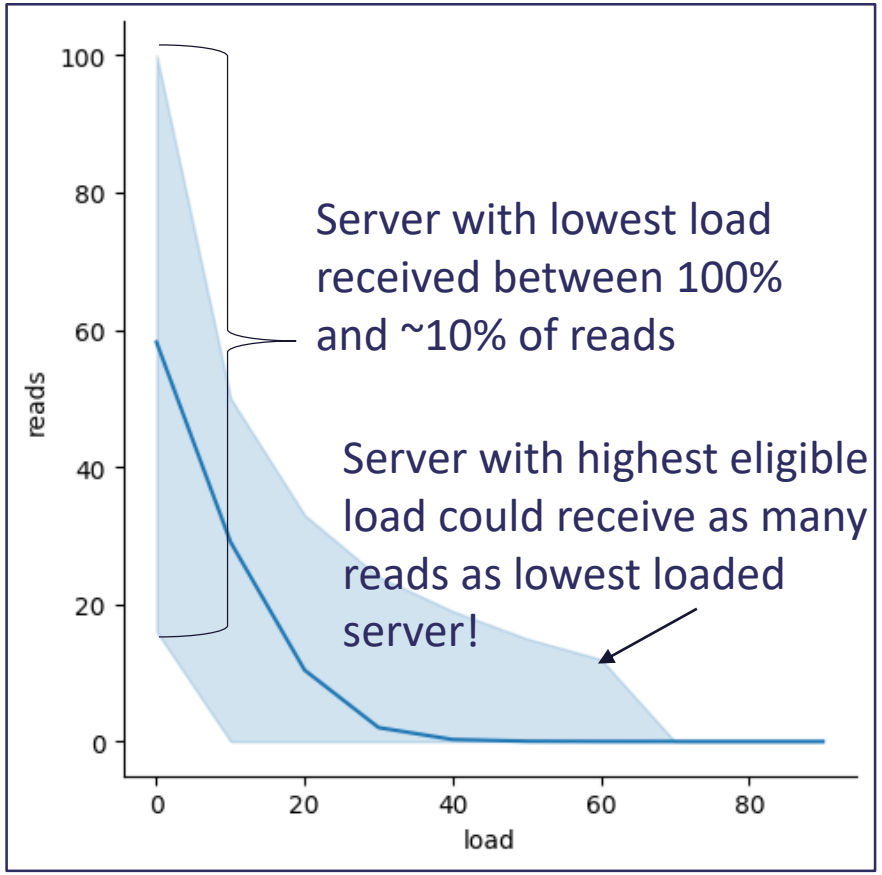


Round Robin was performing better when all servers were at near maximum capacity

Science and Technology Facilities Council

Studying the CMSD load balancing algorithm

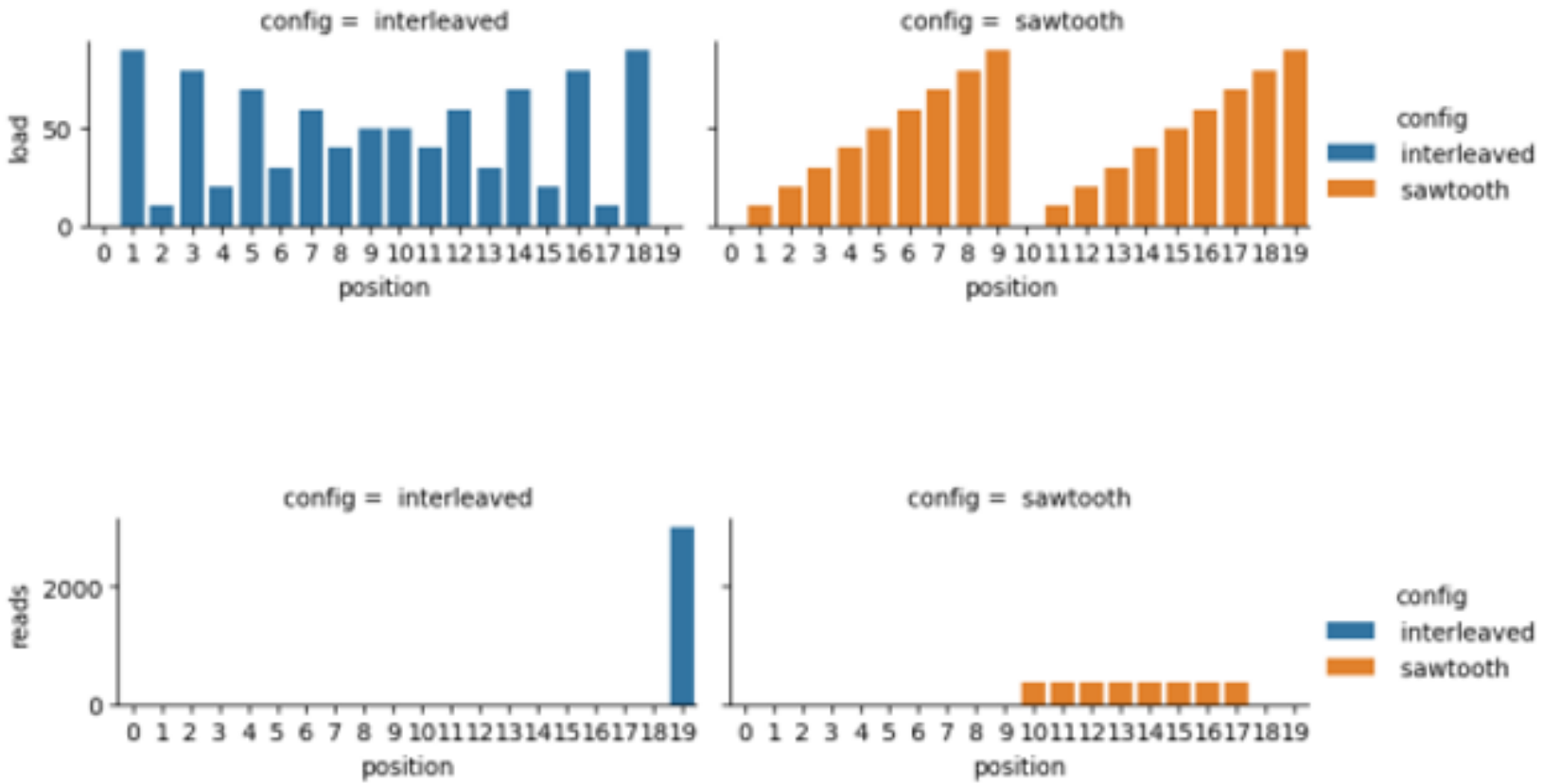
- Investigation into the CMSD *SelByLoad* algorithm revealed selection had a very wide standard deviation on assignment distributions when the server order is randomized (seen right).
- The algorithms efficient single-pass nature made it hard to select servers in a way that wasn't biased by order (as shown on the next slide)



Distribution of 100 requests over 10 servers with load scores 0,10,…,90. The test was repeated 10,000 times with the internal server order shuffled each time.

Science and Technology Facilities Council

Problematic patterns



Science and Technology Facilities Council

XRootD CMSD SelByLoad Analysis, Thomas Byrne 2024

New load balancing algorithm

Aims

Aim	Explanation
Use all available server capacity	Load reporting is asynchronous to the load balancing, so the reported loads have some latency. Prevent load hopping
The decision algorithm should be fast	RAL sees an average request frequency of 50Hz , with peaks of 220Hz and the load balancing decision should not impact the other components of the CMSD
Load distribution should be sensible, not perfect	As the assignment decision is timebound, the main concern is to distribute load evenly within a timeframe, not necessarily optimizing every individual transfer
The algorithm should be resilient	Overloaded servers should be able to join back with no manual intervention
The algorithm should be ordering agnostic	The order the servers login should not affect the load balancing decision

Weighted Random Selection algorithm

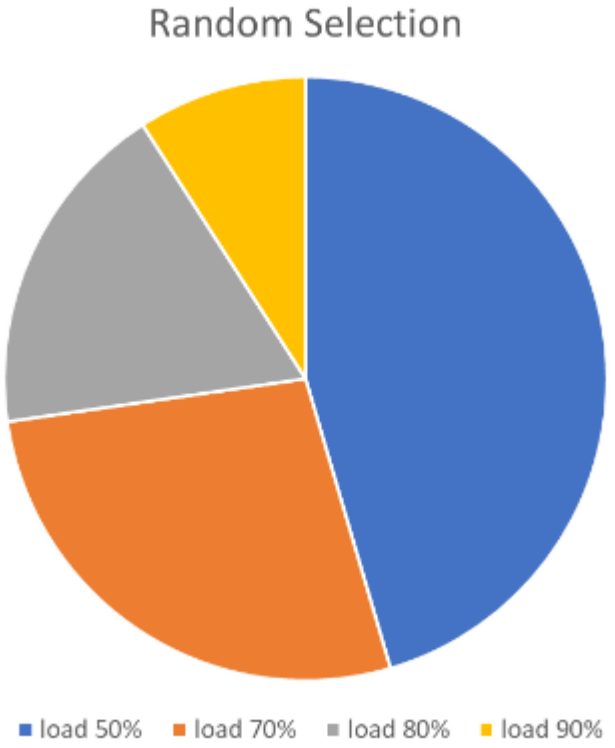
- Given n nodes,
- Each node is assigned an inverse load:
  - iload = Fuzz + (100 – weighted_sum(loads))
- For each node i, the inverse load is added to the total sum. The current sum after each node is kept track of in a separate tracking array
  - Weights[i] = $\sum_{k=0}^{i} iload(k)$
- A random number is generated between 0 and the final total sum
  - SelN = random(0,Weights[n])
- Starting from the first index, the first node that had a tracked sum value greater than the random number is assigned the transfer
  - for i = 0..n ; if Weights[i]>SelN : return i

Science and Technology Facilities Council

How the new algorithm works

The basic principle of this algorithm is a random weighted selection. An easy way to picture it is as follows:

- For a set of 4 servers with loads of:
  - 50%
  - 70%
  - 80%
  - 90%

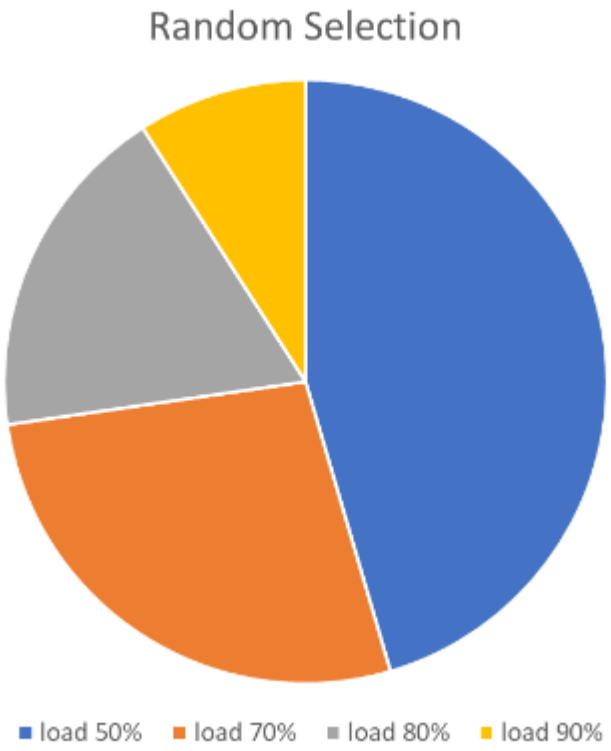


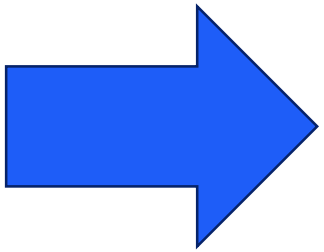
Science and Technology Facilities Council

How the new algorithm works

- If a node is unreachable or cannot be used by the cluster manager, it has an effective slice size of 0, meaning it will never be selected.
- A fuzz value allows an even distribution if all nodes are overloaded and provides a baseline for each slice size, allowing some tuning adjustments for a more even distribution.
  For example, a fuzz of 20 on the same values above will result in this wheel:

How the new algorithm works

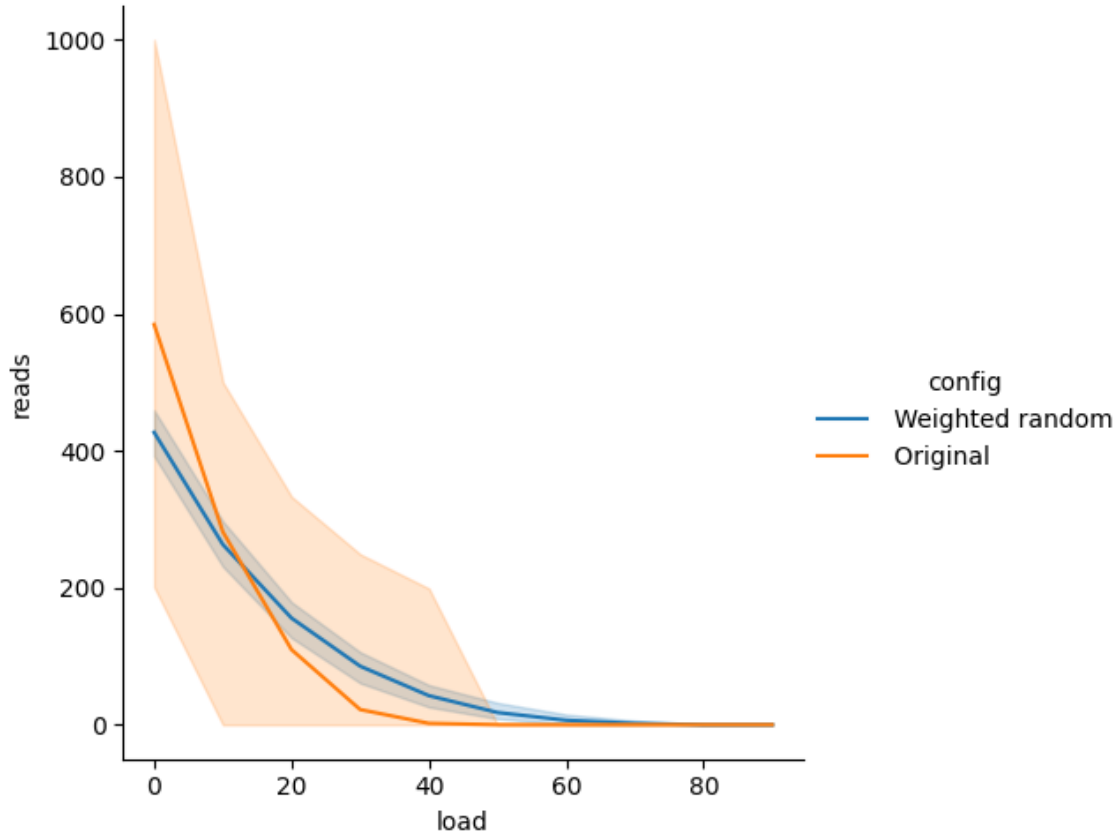




Science and Technology Facilities Council

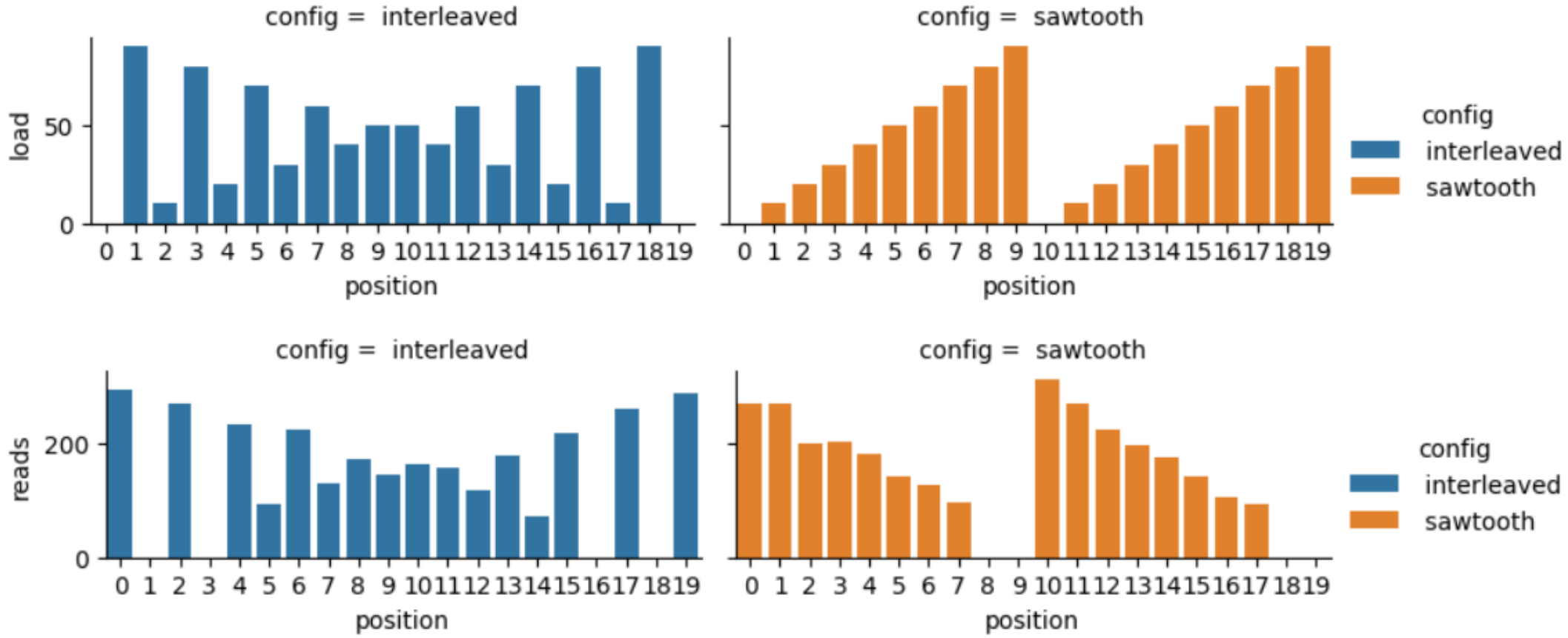
Results

- The new algorithm has a smaller deviation from the mean selection when the order of loads is shuffled (right plot)
- The aims set out by the algorithm were accomplished



Science and Technology Facilities Council

Problematic patterns – new algorithm



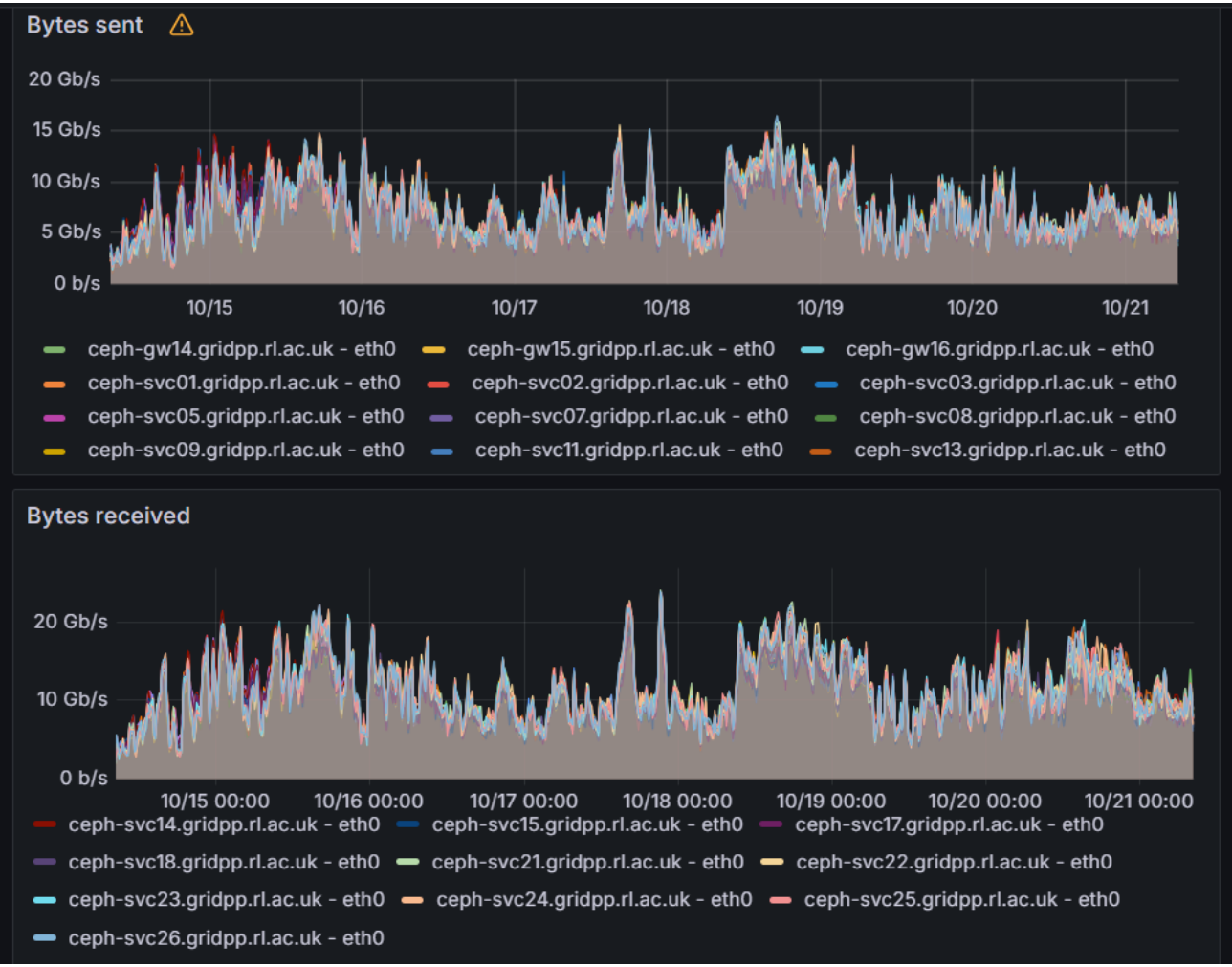
Facilities Council

XRootD CMSD SelByLoad Analysis, Thomas Byrne 2024

Results

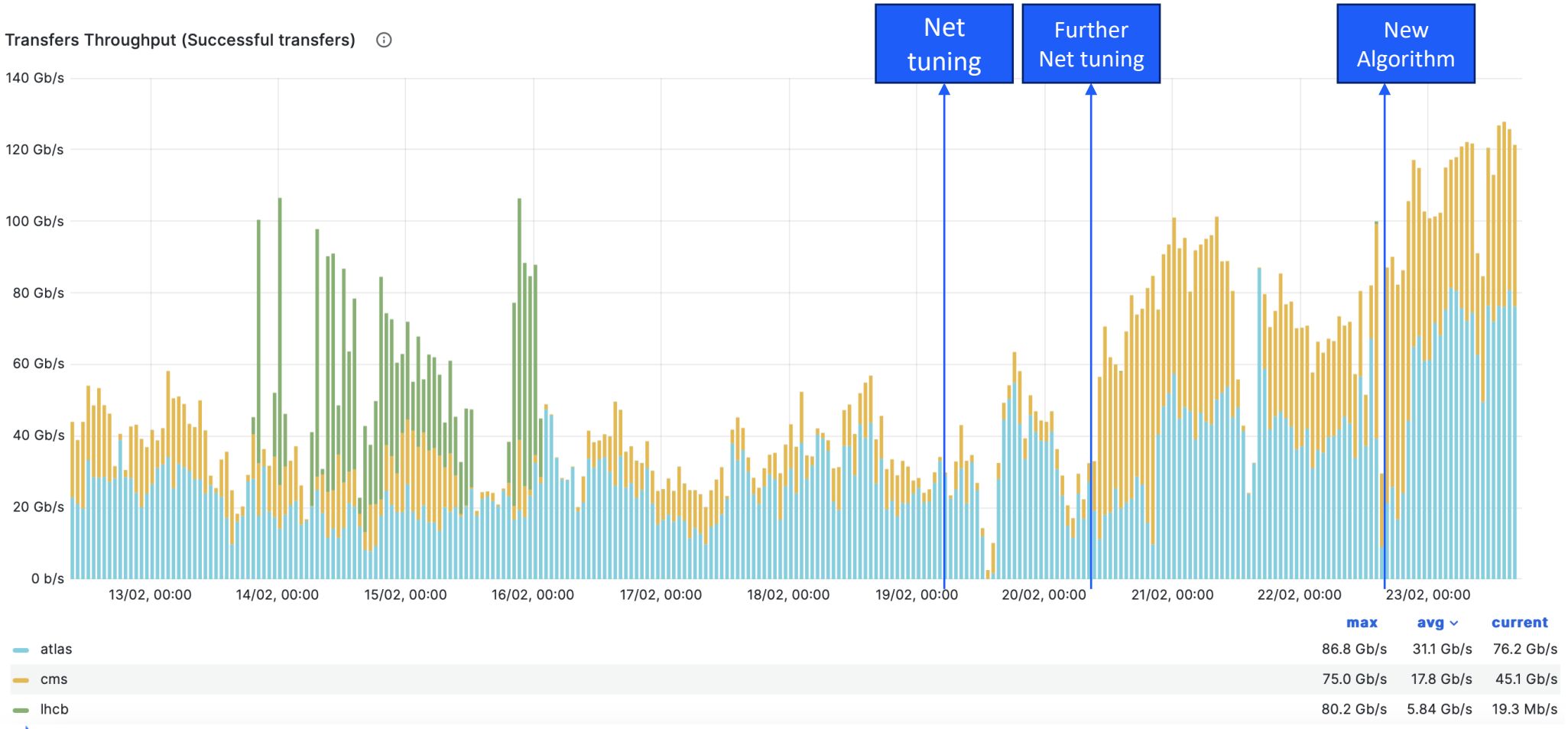
Aim	Explanation
Use all available server capacity	All available servers will have a chance to be selected
The decision algorithm should be fast	The algorithm performed well even under very high loads and full network saturation of all our servers. It has O(n) time and space complexity
Load distribution should be sensible, not perfect	The load distribution was kept even throughout the available servers. Simulations of 10 million decisions resulted in 10 maximum consecutive selection of the same server.
The algorithm should be resilient	As servers get more transfers and get more loaded, the chance of getting selected decreases. As the transfers free up and the load decrease, the reverse happens
The algorithm should be ordering agnostic	The geometry of this algorithm makes it order agnostic by nature

In Production



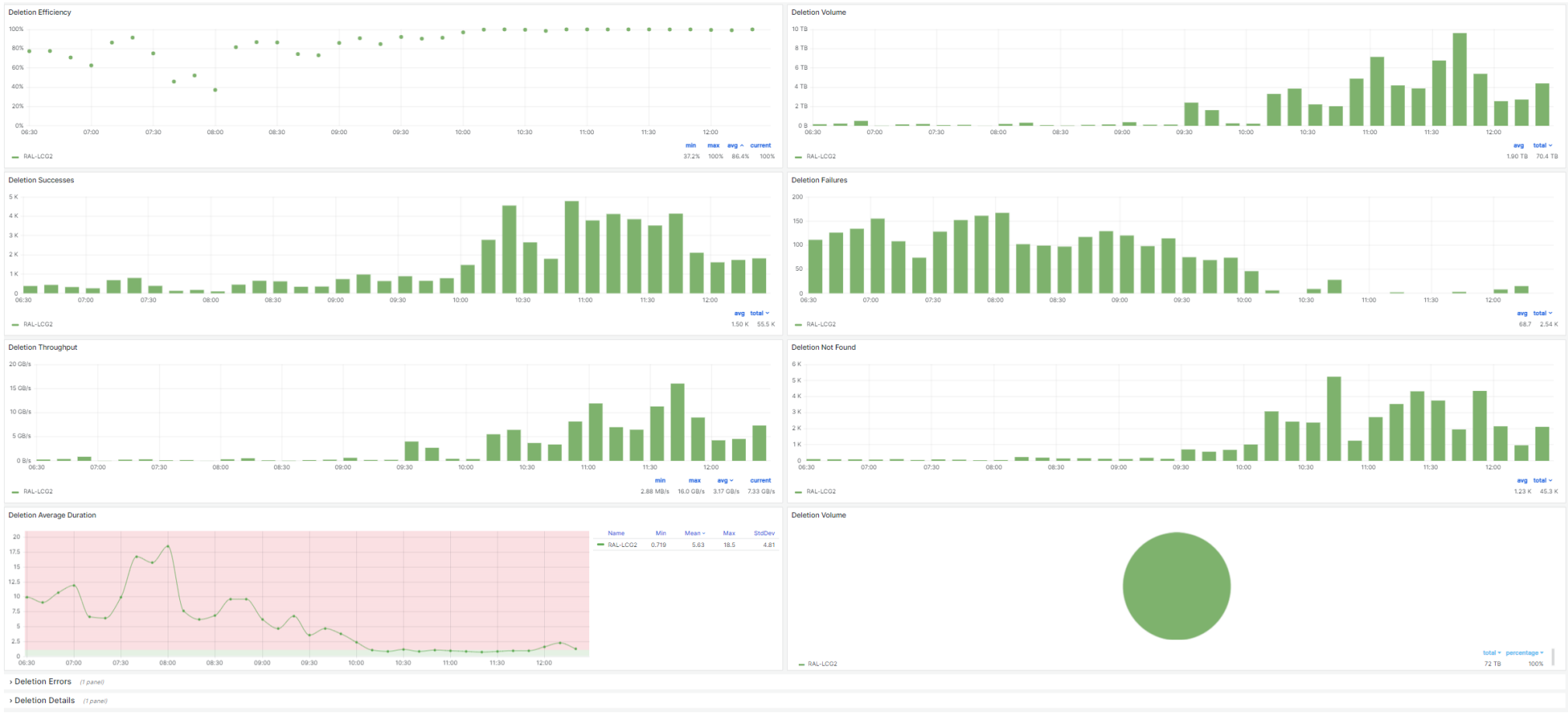
Science and Technology Facilities Council

In Production



Science and Technology Facilities Council

Additional Benefits(RAL): deletion efficiency



Science and Technology Facilities Council

Thank You