Barcelona, Dec. 2023

ADMIRE - Lustre and Quality of Service

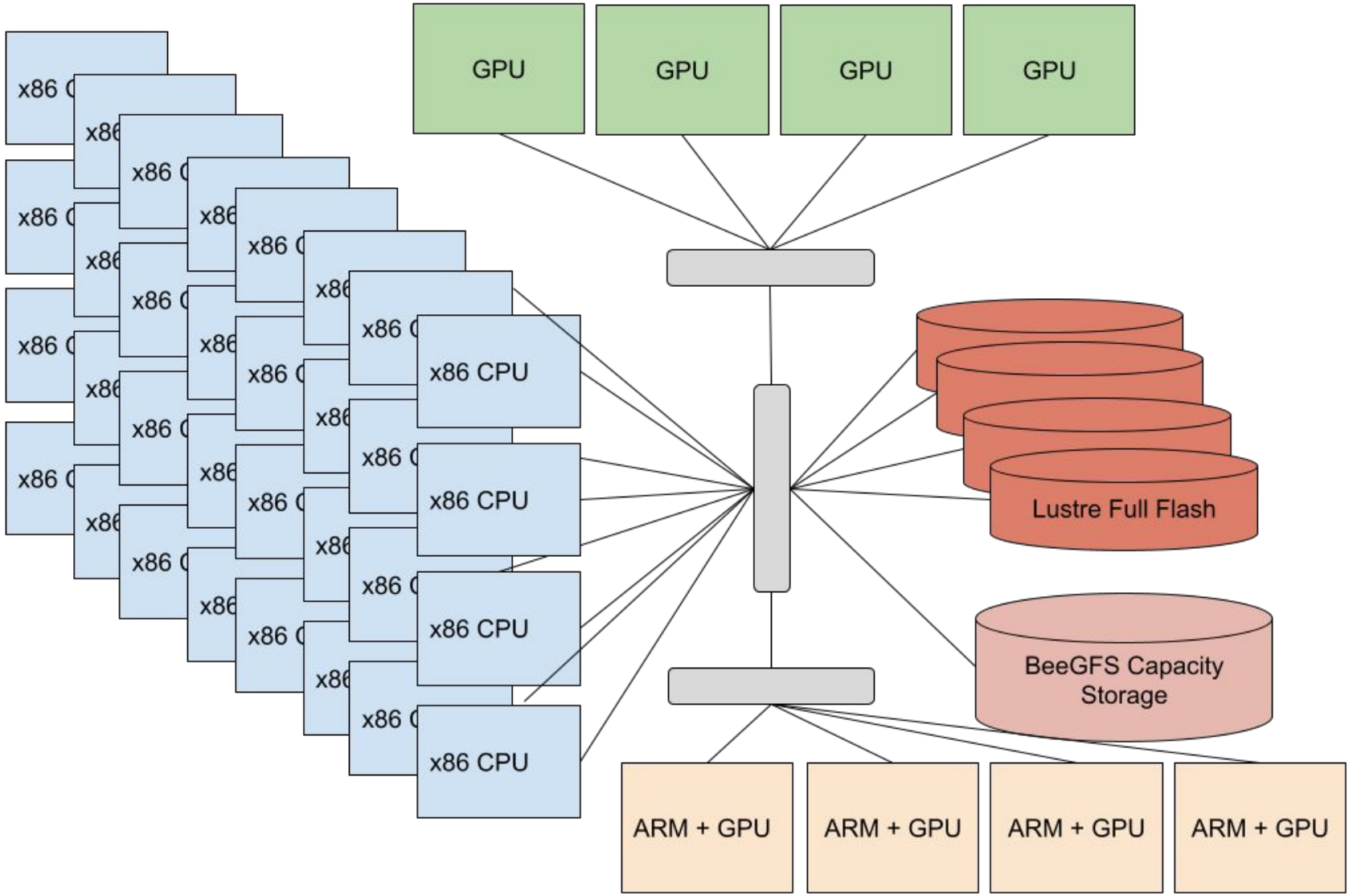
Jean-Thomas Acquaviva – DDN Storage

Workpackage leader

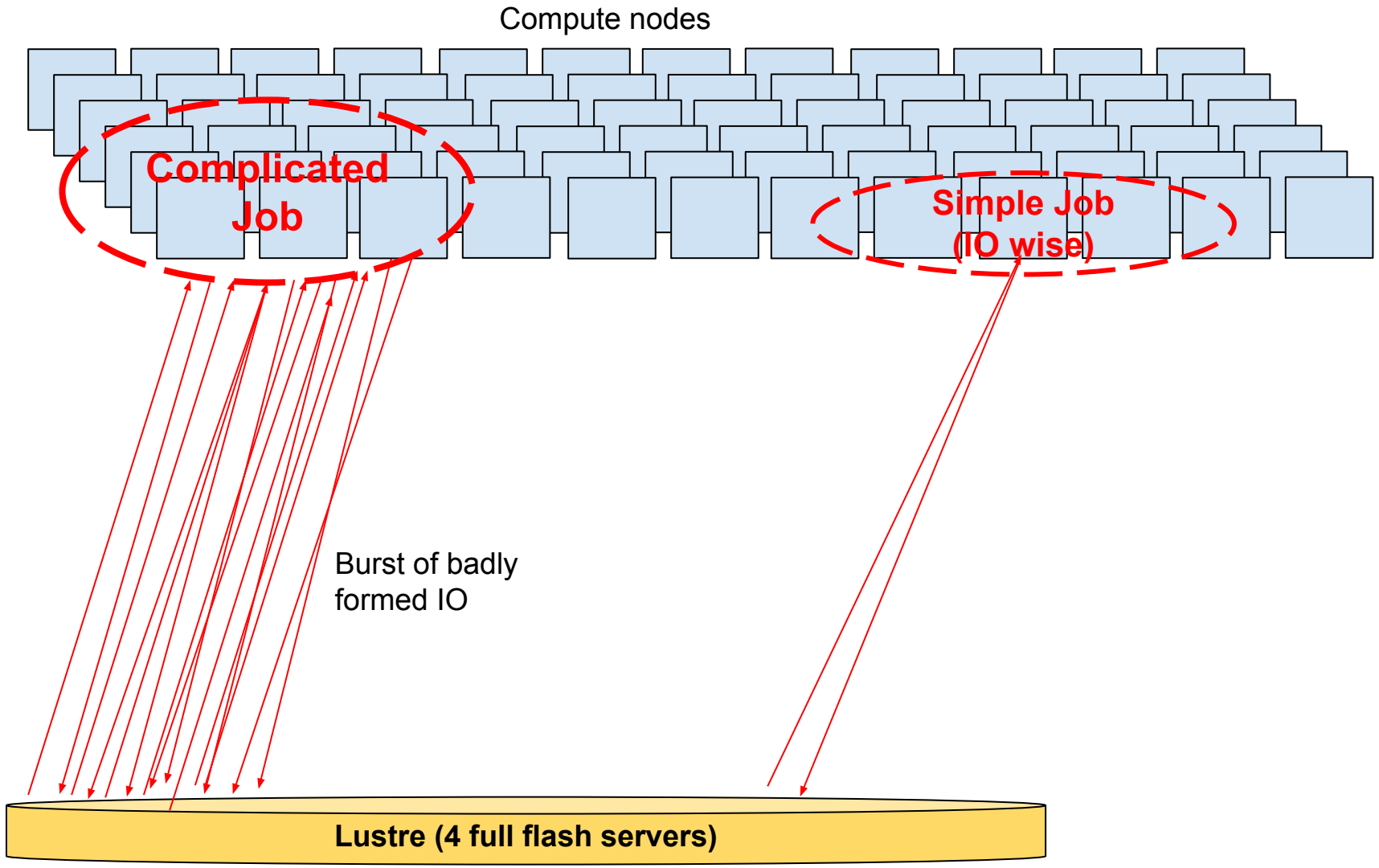
December 11-12 2023.

Grant Agreement number: 956748 — ADMIRE — H2020-JTI-EuroHPC-2019-1

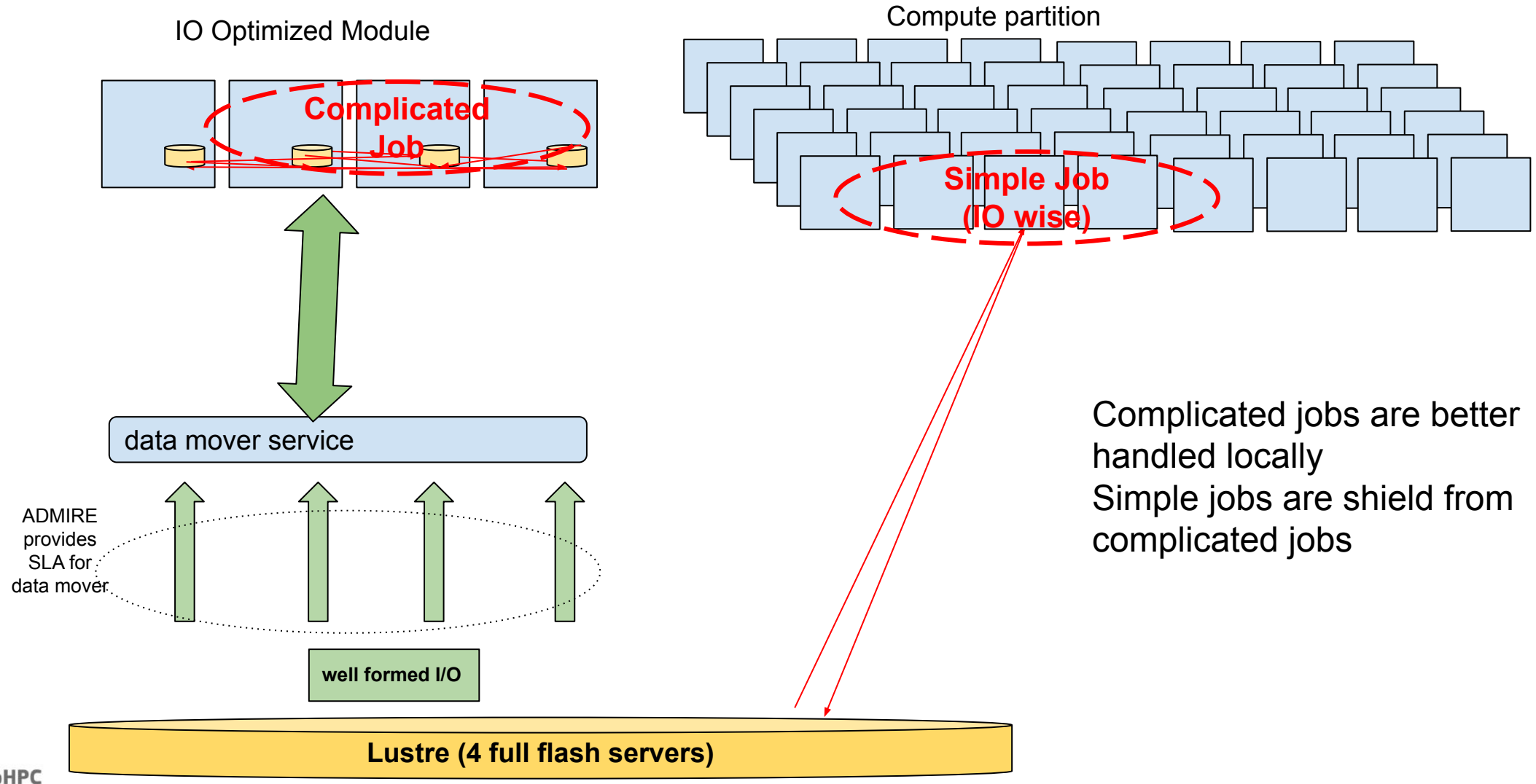
Torino Infrastructure



Handling Demanding and Complex Jobs

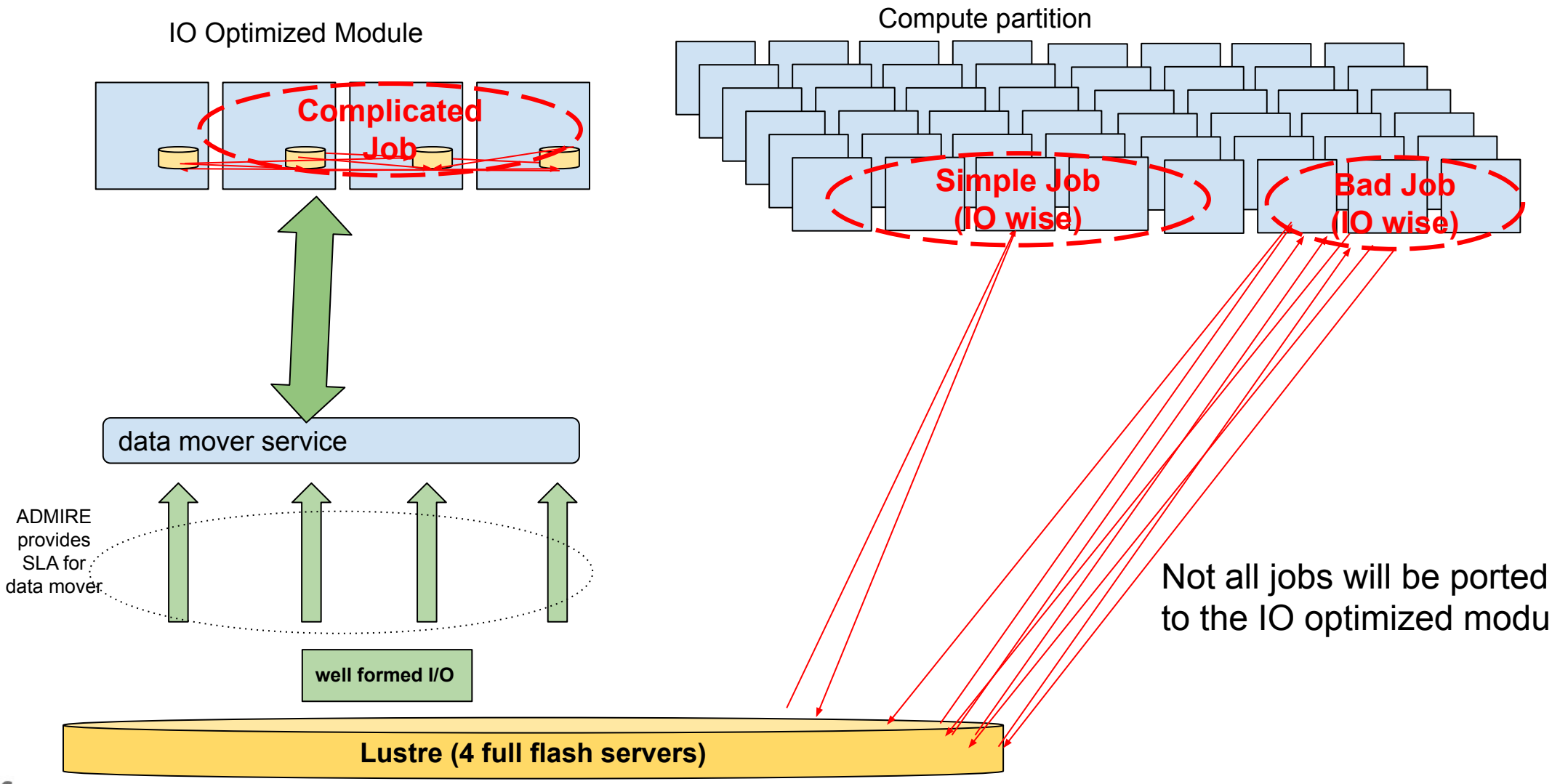


Node Specialization: ADMIRE IO Optimized Module



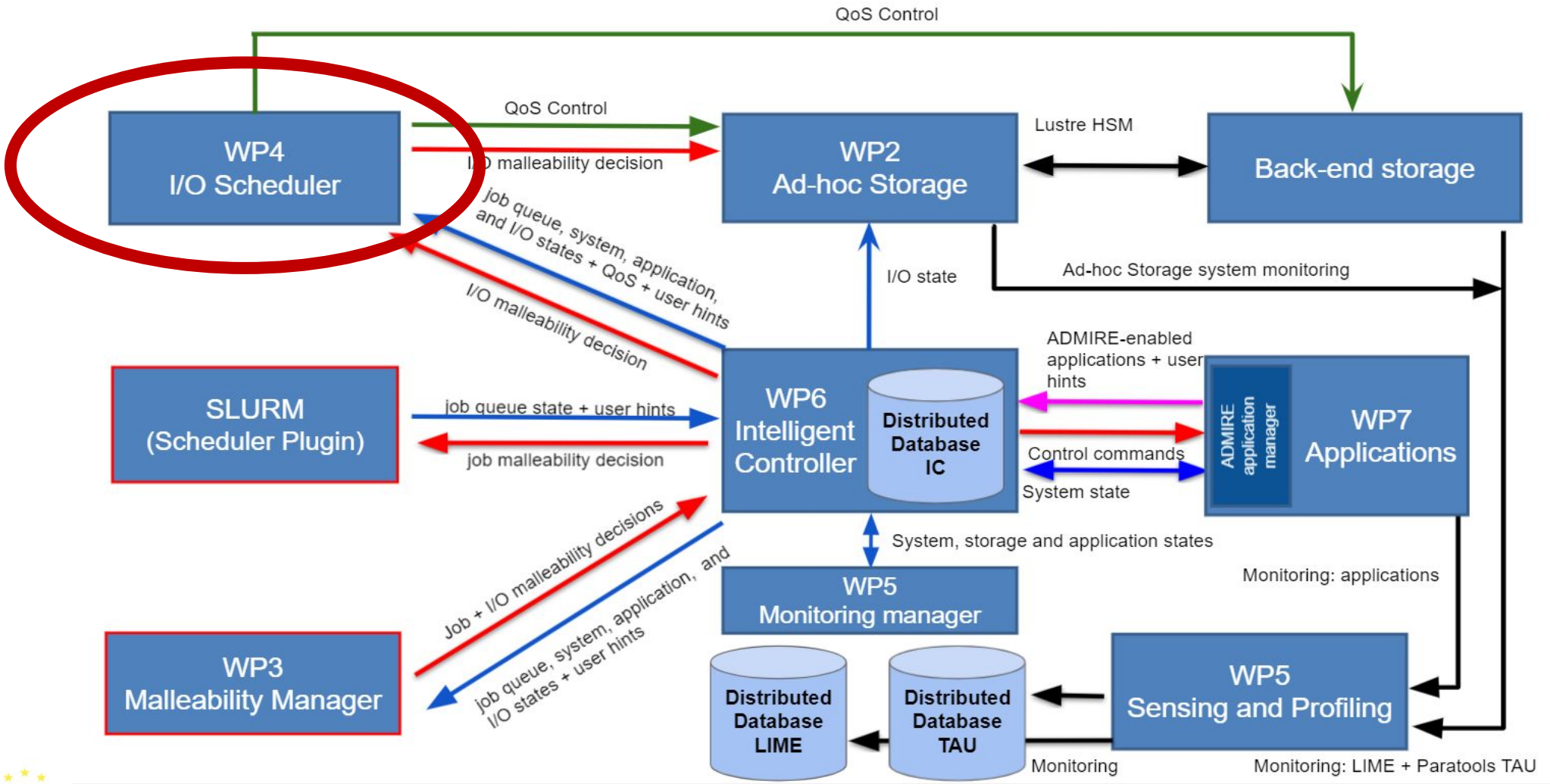
Complicated jobs are better handled locally
Simple jobs are shield from complicated jobs

A single deviant job can degrade system throughput



Not all jobs will be ported to the IO optimized module

Improving Performance Resilience at FS level



Quality of Service Goals

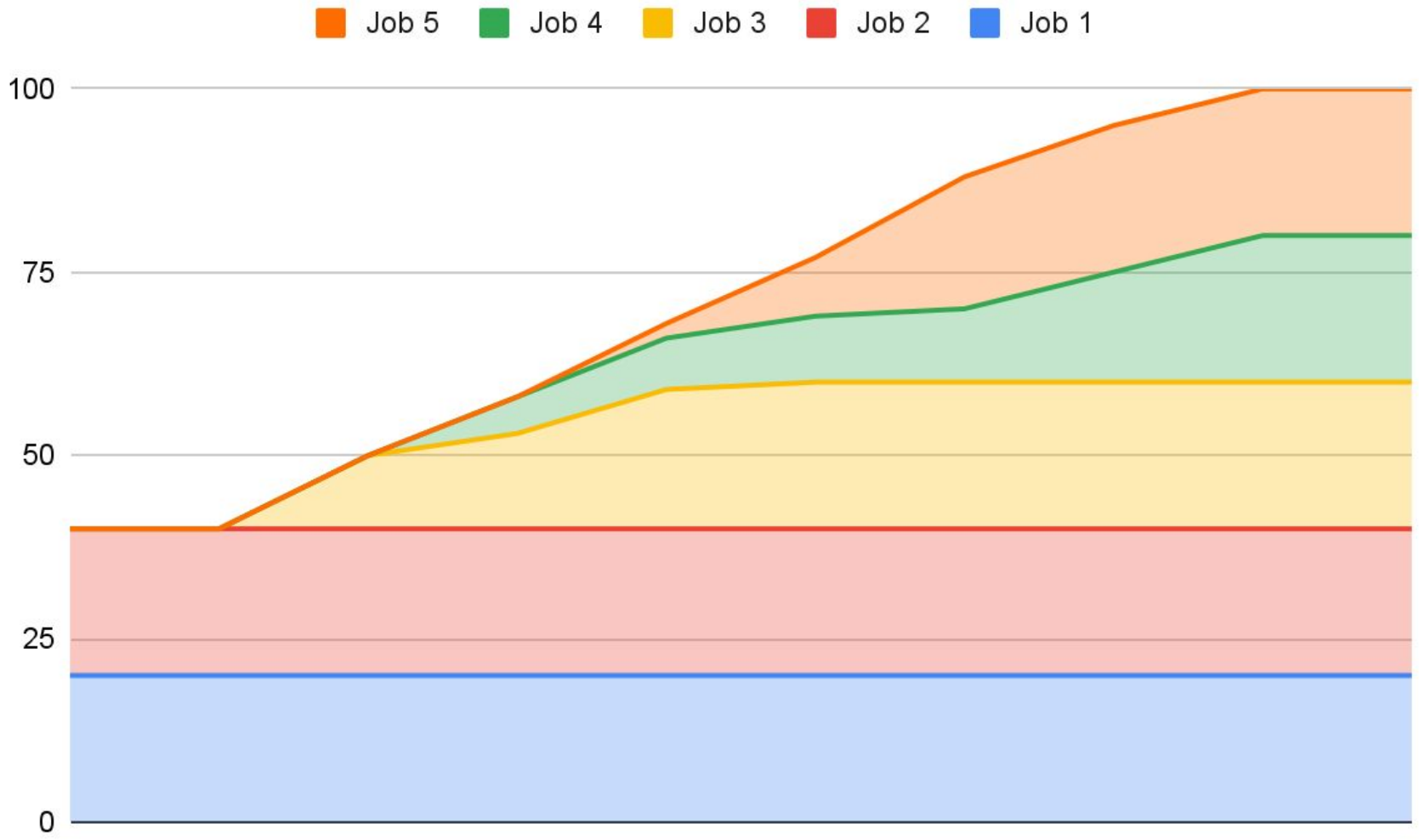
1. Guarantee that all jobs are served fairly
   - Prevent starvation or denial of service
   - Ensure that all jobs are served in accordance with available resources
2. Use all available resources
   - Allows jobs to consume all the available resources
3. Support SLA
   - Distinction between Jobs (production constraints)

Issue #1 Starvation

Job 1 and 2 are squeezed due to apparition of additional jobs on the system

#1 Assign a minimal fraction of resource to each job

Issue #2 Loss of efficiency



Capping the consumption per job leads to suboptimal usage of resources

Type of QoS in Lustre

- Interconnect
  - e.g: Infiniband QoS (with the concept Virtual Lane and Service Lane)
  - Prevent starvation
  - Does not fully optimize resource usage
- LNet
  - Multi-Rail health algorithm: use to depreciate the usage of a local or remote interface if it return a lot of error.
  - Multi-Rail User Defined Selection Policy (UDSP): allow policies for local/remote interface prioritization by NID
- Token bucket filter (TBF)
  - Allow the administrator to define rules to enforce the RPC rate limit on it
  - Initiated in 2014… finalized in ADMIRE!

Token Bucket Filter (TBF)

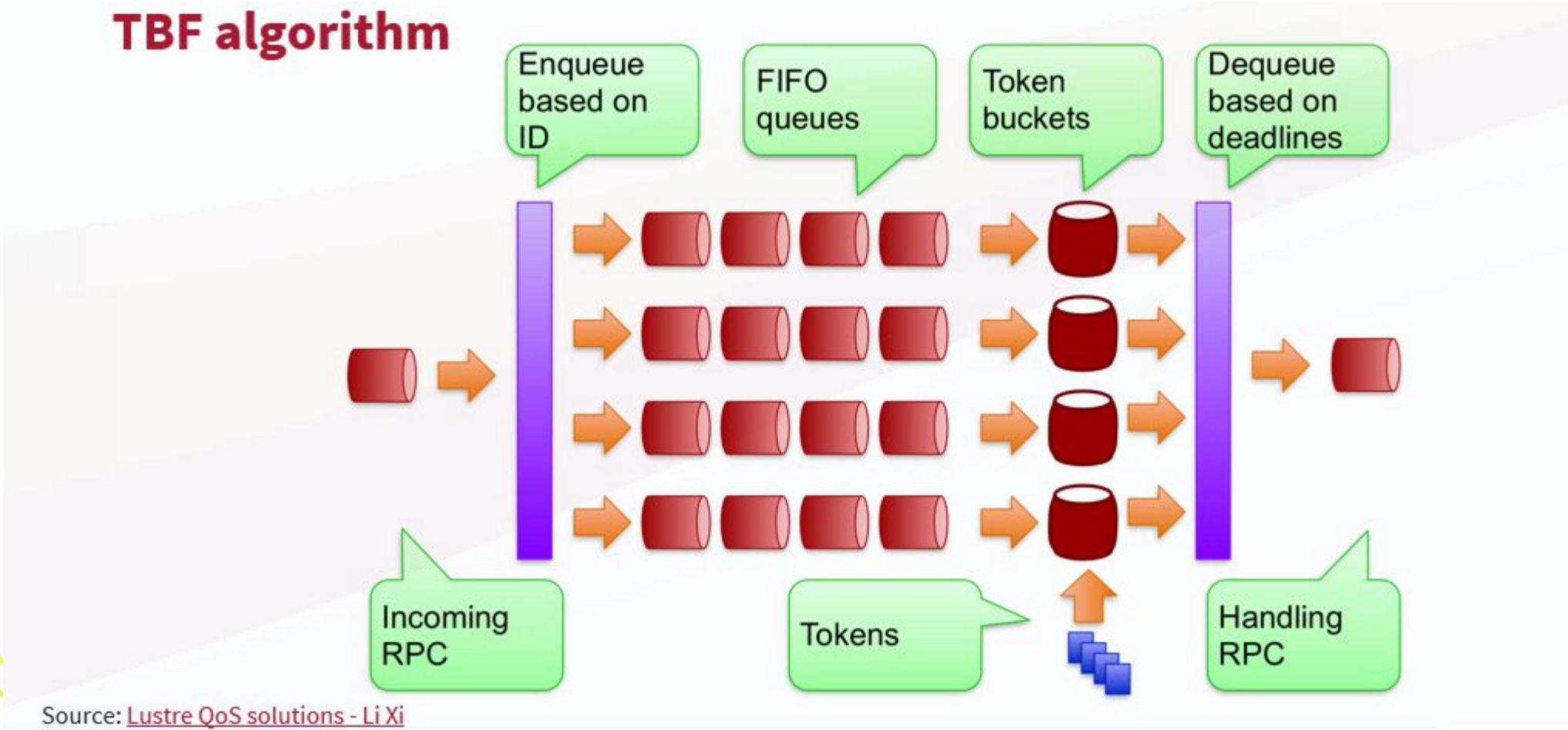
- Allow the administrator to define rules to enforce the RPC rate limit on it.
- NRS (Network Request Scheduler) is able to reschedule/resort/throttle the RPCs before forwarding them to the handling threads on MDS/OSS
- TBF (Token Bucket Filter) is the policy that enables NRS to enforce RPC rates by user defined rules on some users/groups/nodes/jobs
- Taking nominal RPC rate processing of the File system
  - Sysadmin define fraction of the rate using TBF filters

Token Bucket Filter (TBF) - Supported filters

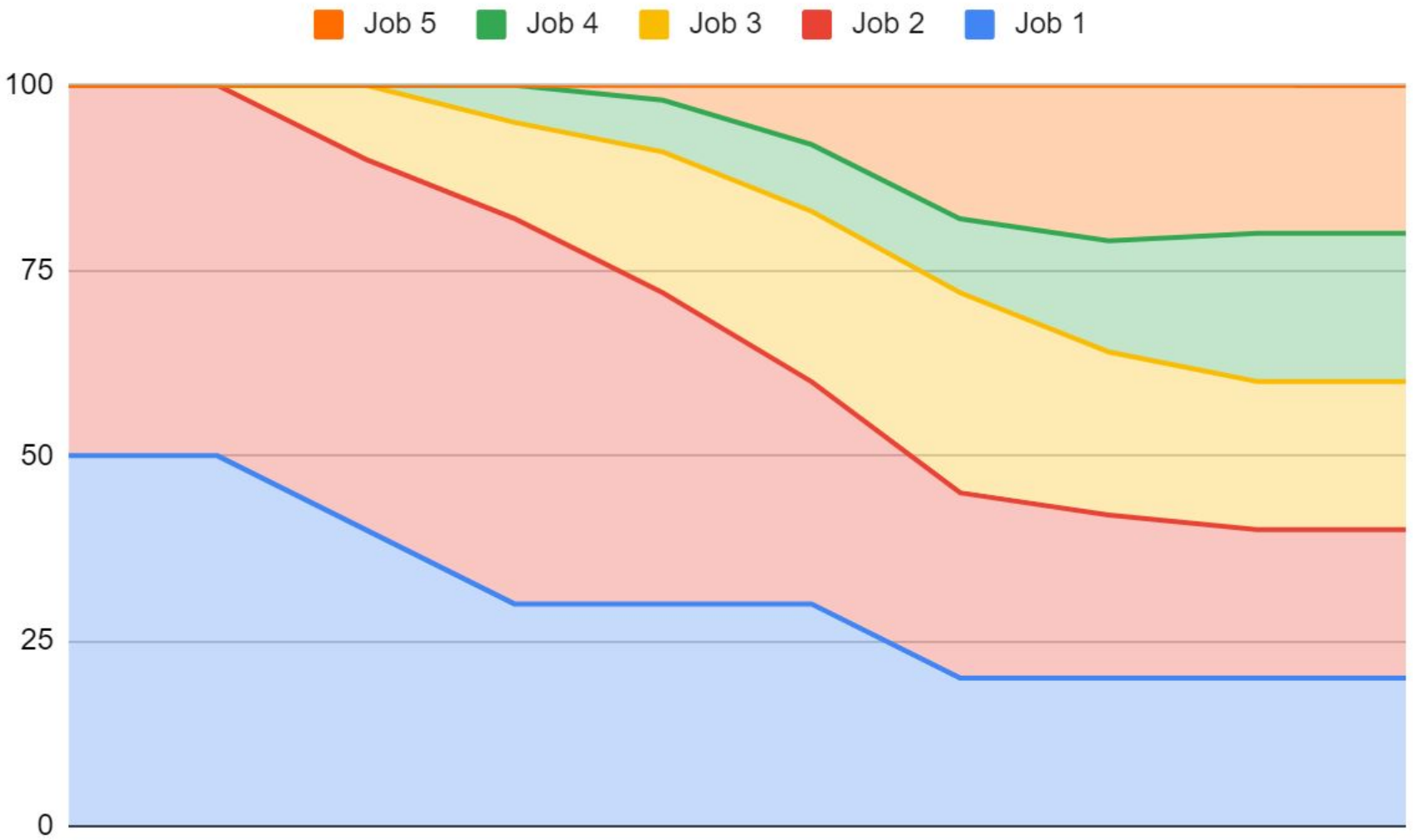
- TBF policies (to activate via parameter nrs_policies of a service):
- "tbf jobid": enforce a rate for each unique jobid.
- "tbf nid": enforce a rate for each unique network node.
- "tbf uid"/"tbf gid": enforce a rate for each unique process UID or GID.
- "tbf opcode": enforce a rate for each unique type of RPC request.
- "tbf": enforce a rate for each unique id constructed with RPC jobid, NID, UID, GID and opcode

TBF Logic: RPC queue management

- **Filter rules applies when queues are congested**
  - Ensure full resource allocation when activity is limited

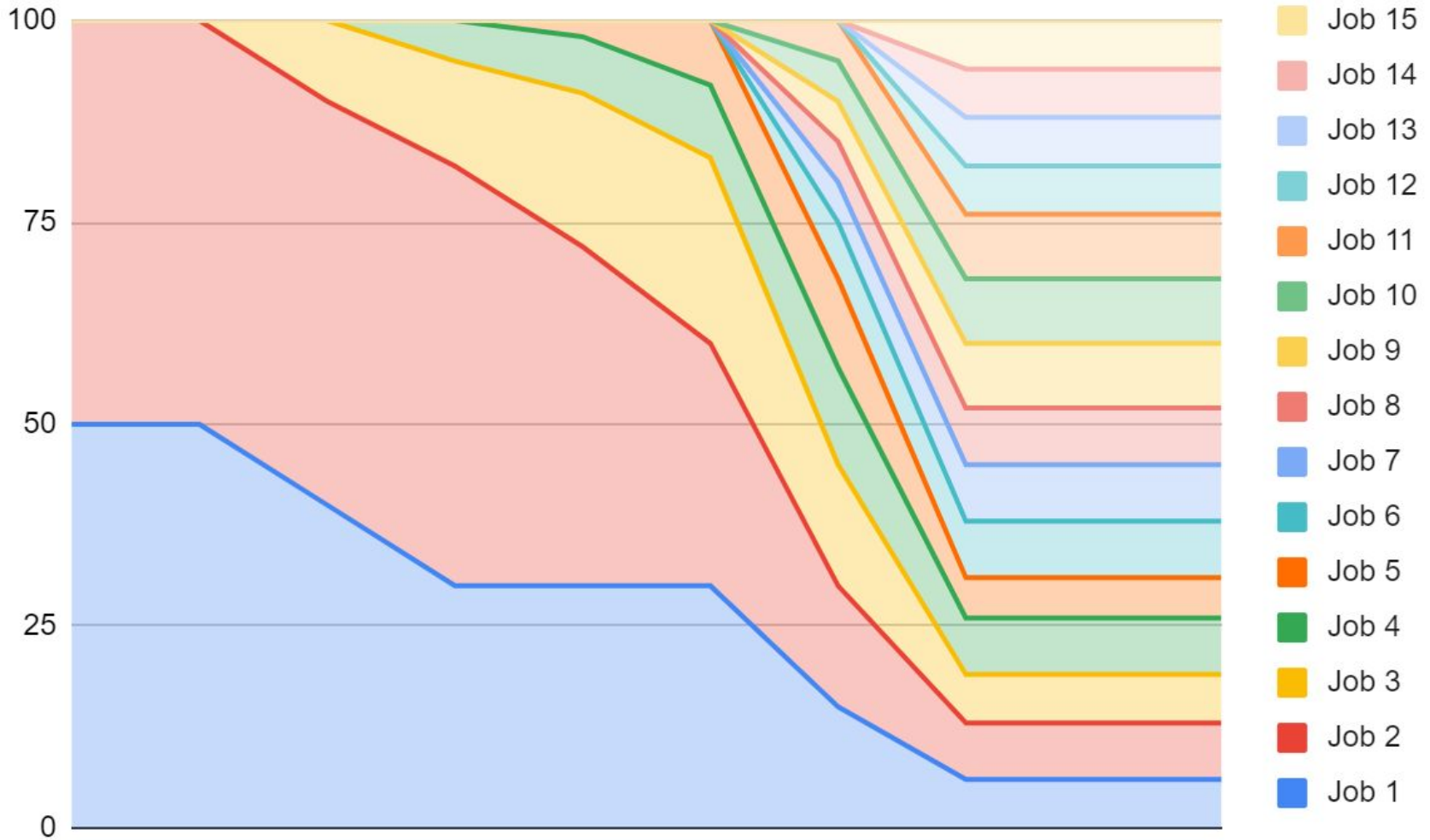


Fair Share: Silenting the Noisy Neighbour



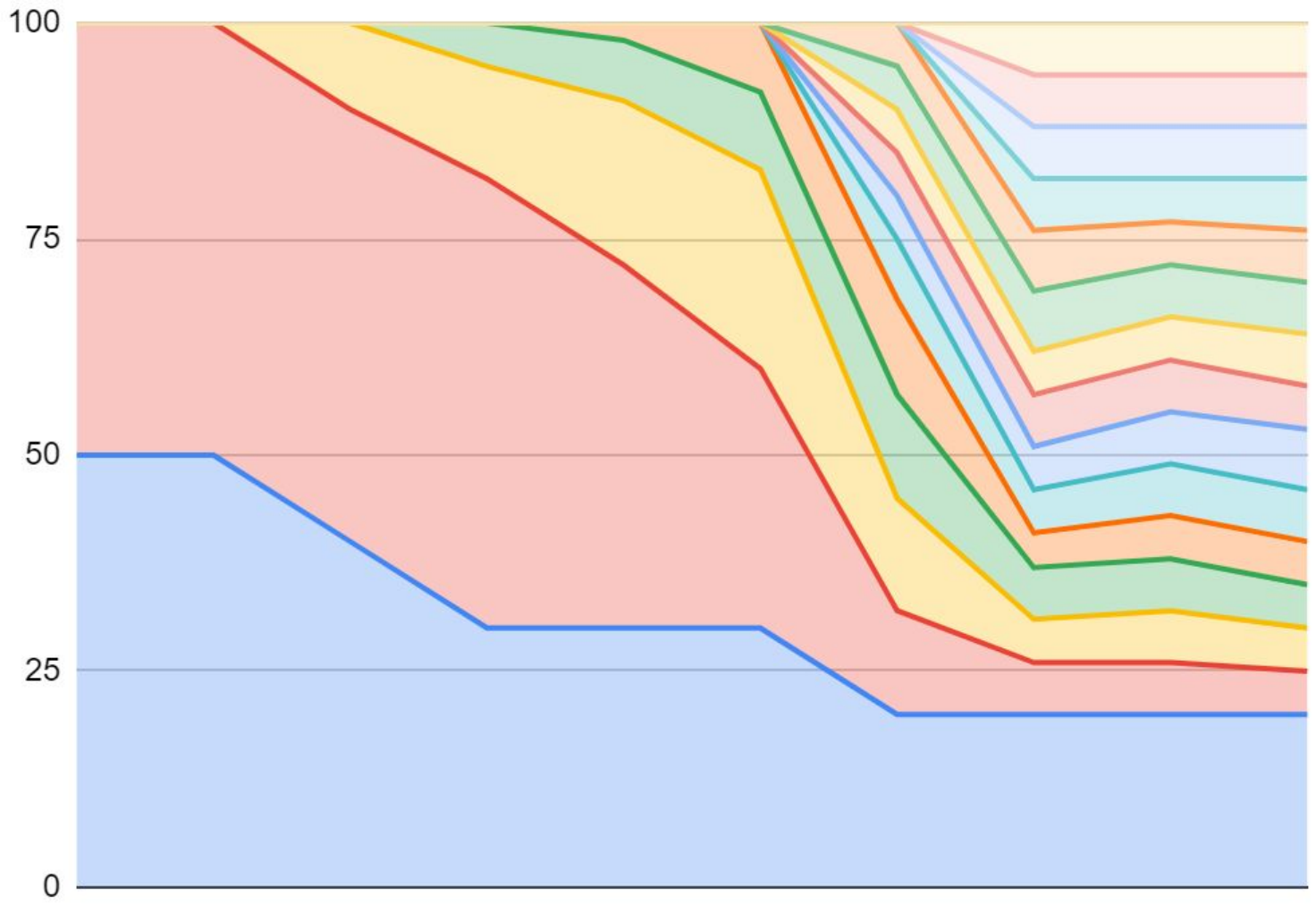
Resources are fully utilized and all jobs obtains 20% of the system

Issue #3 Fair Share: flat hierarchy



Job 1 needs at least 20% to finish on time. It will run late due to too many active jobs in the system

Introducing new kind of job: Real Time



Job 15
Job 14
Job 13
Job 12
Job 11
Job 10
Job 9
Job 8
Job 7
Job 6
Job 5
Job 4
Job 3
Job 2
Job 1

Job 1 is guaranteed to receive needs at least 20% of resources independently of the number of active jobs

Integration: self contained data logistic

TBF Activation on jobid for data service (ost_io)

- `lctl set_param ost.OSS.ost_io.nrs_policies="tbf jobid"`

Limit bandwidth for read and write to 3000 RPC/sec

- if payload is 1MB, bandwidth is capped at 3GB/s, if payload is 16MB bandwidth is capped at 48 GB/S
- `lctl set_param ost.OSS.ost_io.nrs_tbf_rule="change default rate=3000"`

Set bandwidth limit to a specific job

- `lctl set_param ost.OSS.ost_io.nrs_tbf_rule="start rule_fio jobid={fio.*} rate=5000"`

Mark a job as belonging to the Real Time Class

- `lctl set_param ost.OSS.ost_io.nrs_tbf_rule="start rule_fio jobid={ior.*} minrate=1000 rate=5000 realtime=1"`

Lustre QoS: Rate Calibration

```
lctl set_param ost.OSS.ost_io.nrs_tbf_rule="change default rate=3000"
```

- Rate depends on file system capabilities
  - Micro-benchmarking file system peak performance
  - RPC based but not all RPCs have the same wait
    - Depends on the payload
    - Write costlier than Read
- Rate has to be estimated
  - Does not depends only on FS capabilities
    - Network and Compute nodes impact overall RPC rate
- Pragmatic approach based on trial/error (fine tuning)

Who's launching the command?

On every service (data server OSS)

- `#clush -w fast[1-4] lctl set_param ost.OSS.ost_io.nrs_policies="tbf jobid"`

Root access needed

- Only admin can assign resource to jobs
- `lctl set_param ost.OSS.ost_io.nrs_tbf_rule="change default rate=100"`

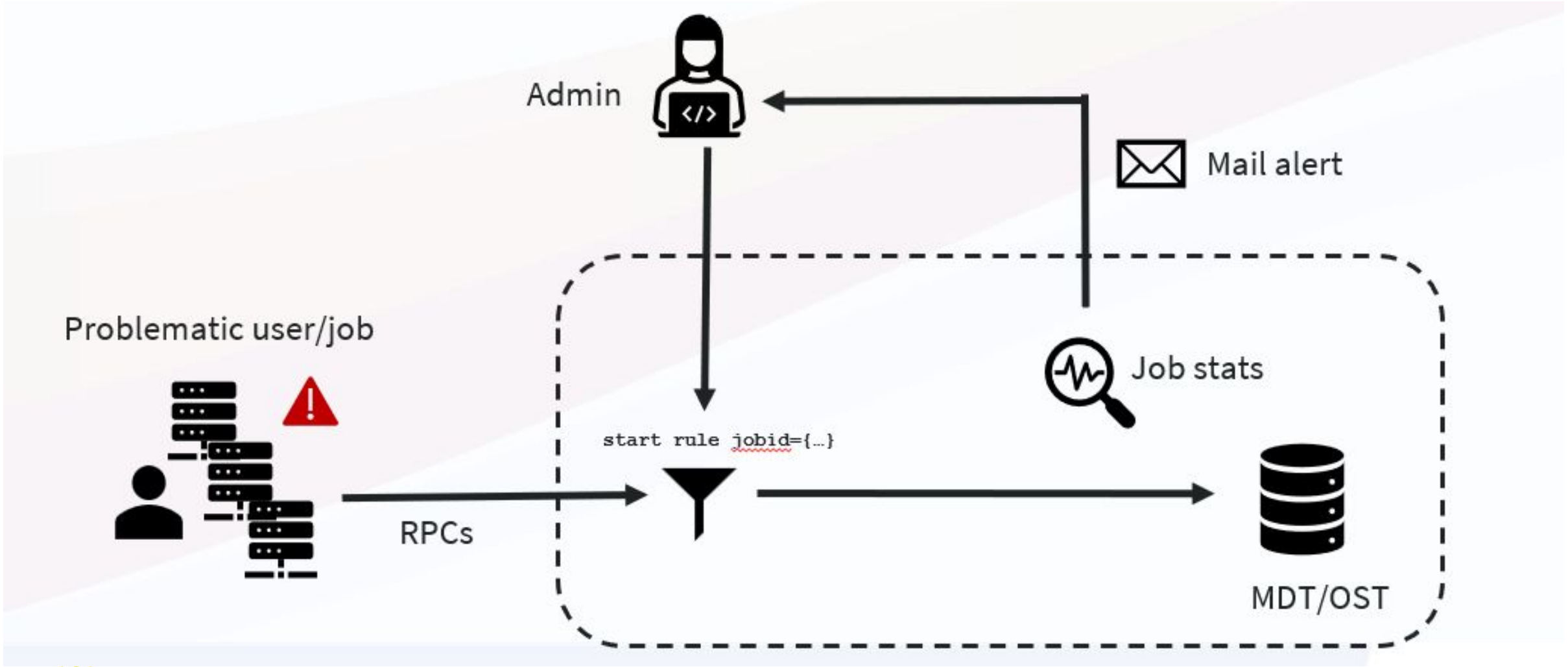
Integration with job scheduler much needed

- Current test on Torino cluster where conducted with manual setting

Writing rules can be complex

- `lctl set_param  ost.OSS.ost_io.nrs_tbf_rule="start comp_rule opcode={ost_write}&jobid={dd.0},nid={192.168.1.[1-128]@tcp 0@lo} rate=100"`

Coupling TBF with notification mechanism



Lustre QoS

- Software Quality of Service
  - Maximise usage of resources
  - Provide Fair Share
  - Does not replace Hardware-based ADMIRE Ephemeral File System
    - EFS improve performance, QoS guarantees performance
- Hierarchy of Service
  - Creation of an extra-category does not close the conceptual issue
    - Production oriented work-around
- Require Policy definition and Scheduler integration
  - Rules have to be dynamically changed in function of the server loads (feedback loop)
- Code made Open-Source

THANKS!

QUESTION?