2014/10/14

DataDirect™ NETWORKS
INFORMATION IN MOTION™

Lustre QoS based on NRS policy of Token Bucket Filter

Shuichi Ihara, Li Xi

DataDirect Networks

Why Lustre QoS?

- Lustre is able to provide scalable throughput and IOPS which could be increased linearly with the number of OSTs/MDTs
- Storage systems in HPC centers are usually shared by multiple organizations and various applications
- The ability to guarantee sustained performance of file system is essential
  - User experience, e.g. intolerable delay of ‘ls’
  - Workloads that have certain performance requirements
  - Increasing application areas outside the mainstream HPC, e.g. large parallel application (“file-per-process”) use cases
- Need to provide mechanism to “allocate” or “limit” performance

Quality of Service (QoS)

- Quality of Service (QoS) is a mechanism to ensure a "guaranteed" performance
- QoS was developed mostly in the network world, and especially, on TCP/IP networks, which pose specific QoS challenges
- QoS features are available on many network hardware or network management software products
- QoS is somewhat less common in the storage world, although some (expensive) enterprise storage products claim QoS or QoS-like features

Lustre QoS

- Lustre QoS is hard given the complex software stacks of Lustre
  - Cache affect performance significantly
  - Both server and client side mechanisms are needed for full control
  - Extra communication between server and client might be needed
  - Performance of ordinary operations should not be affected
- We present the first steps towards functional Lustre QoS: NRS TBF policies
  - Classify RPCs according to their NID/JOBID/OPCODE
  - Reschedule RPCs based on their classifications
  - Throttle RPC rate by controlling token rate

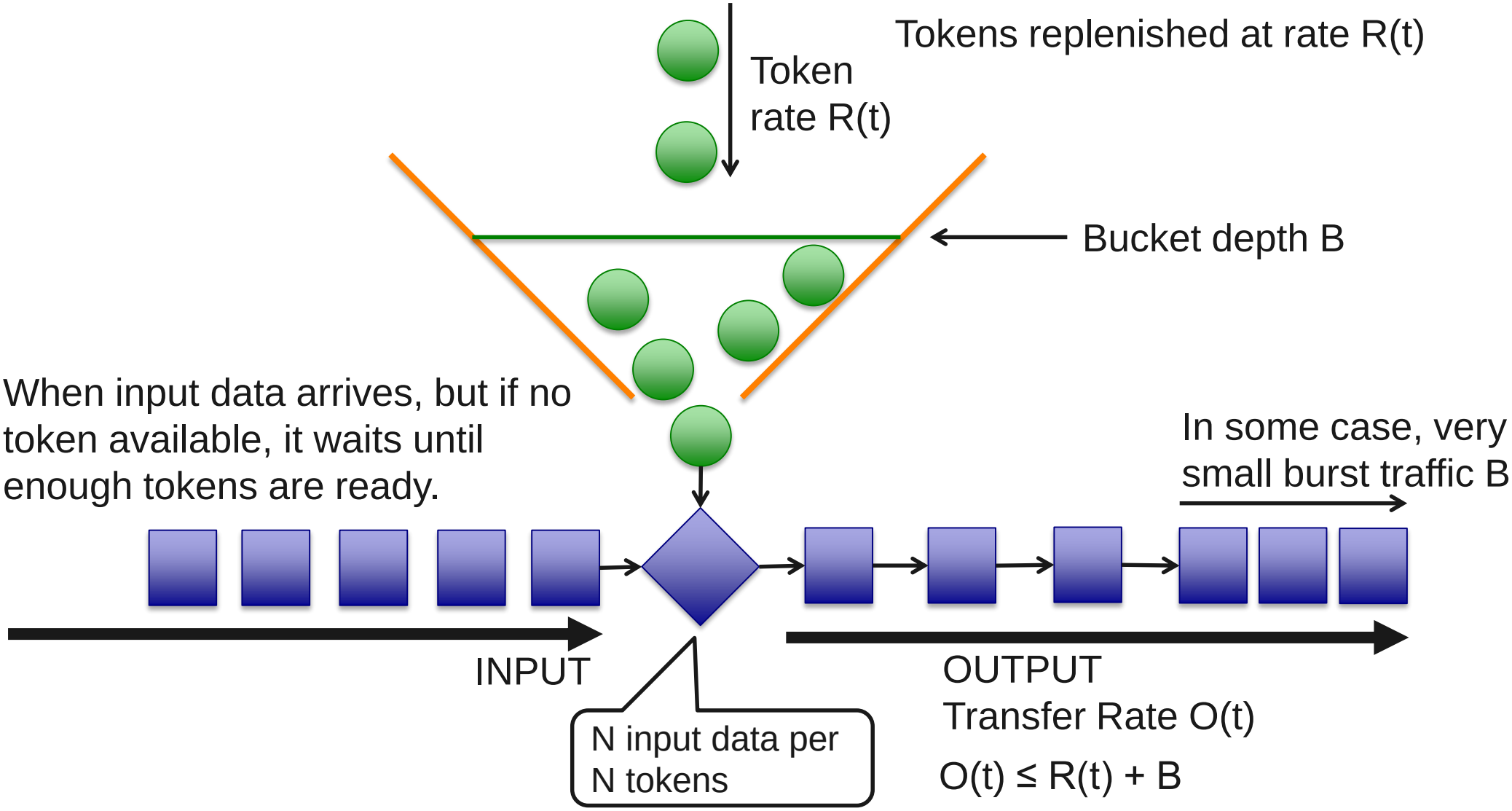
The Network Request Scheduler (NRS)

- A component of the PTLRPC service
- NRS works on the server side and allows handling of incoming RPCs before passing them to the OSS/MDS threads for the backend file system
- This framework, together with a few policy options, was merged into the Lustre mainstream and has been available since Lustre 2.4.0
- The NRS Framework is very flexible and it is fairly easy and straightforward to add new policies
- Policies can reschedule RPCs based on NID/UID/GID/JOBID, etc..

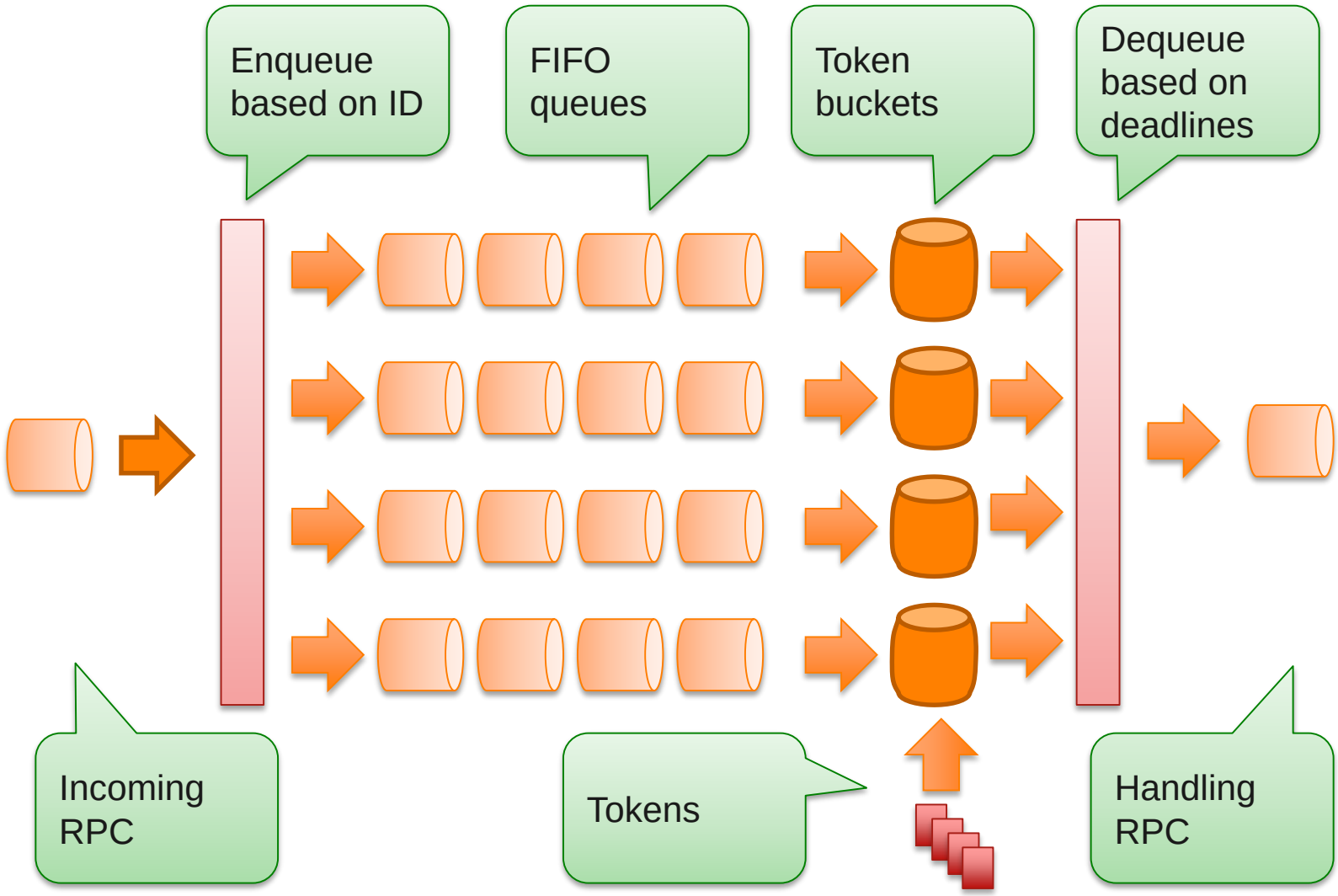
QoS Algorithm: TBF

- Many types of QoS algorithms have been developed over the past few decades
- The Token Bucket Filter (TBF) is a major algorithm used in general network systems
  - It's simple and easy to implement
  - Many Ethernet switches and routers use TBF to enable QoS features
  - TBF can accommodate very small burst traffic, but is also OK for long-term data transmission

The Token Bucket Filter (TBF)



NRS TBF policy of Lustre



TBF patches for Lustre

- LU-3494 libcfs: Add relocation function to libcfs heap
  - Add a function to efficiently change the rank of queue

- LU-3558 ptlrpc: Add the NRS TBF policy
  - Add main framework of TBF policy along with NID/JOBID based policies

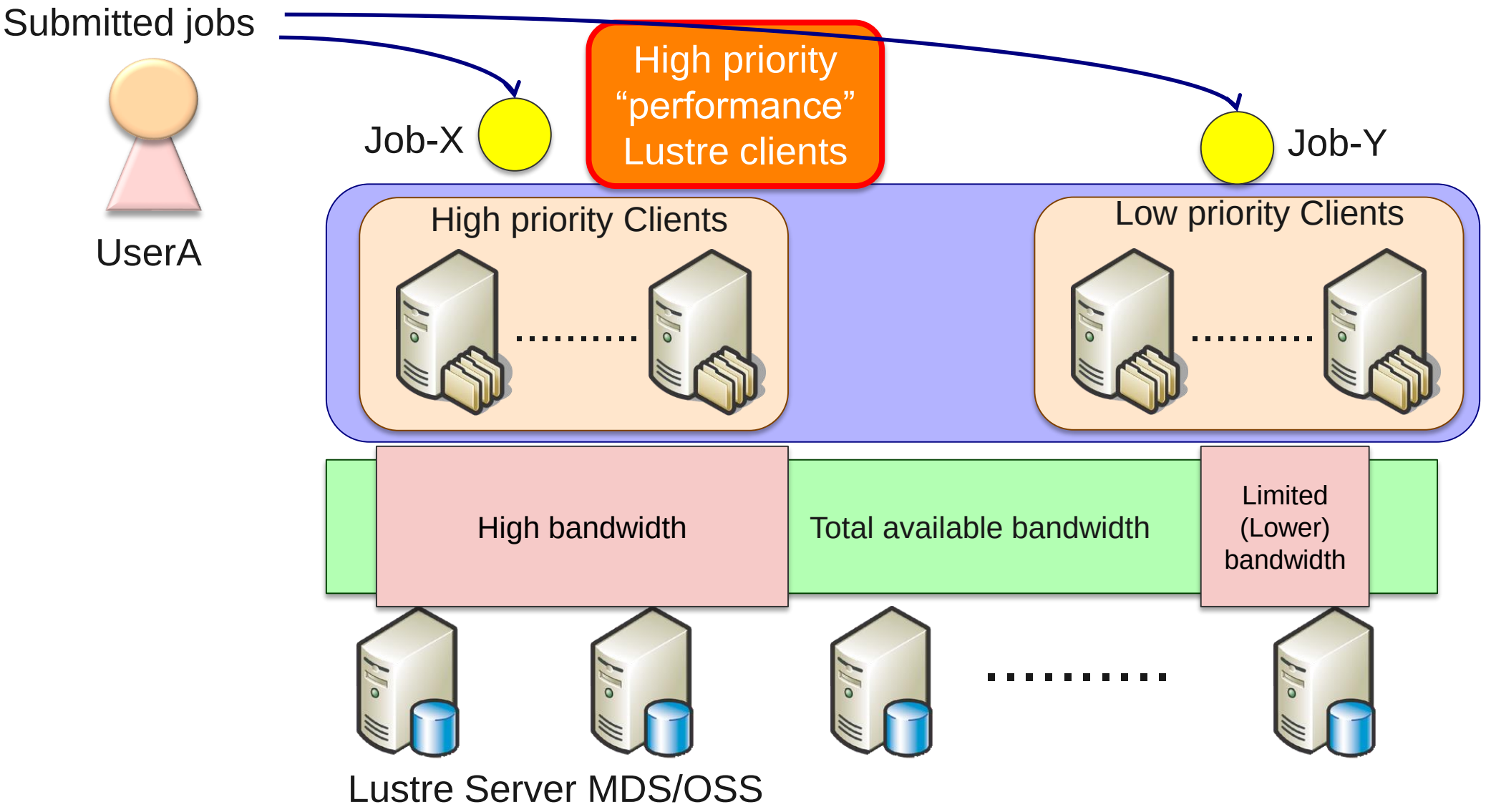
- LU-5580 ptlrpc: policy switch directly in tbf
  - Fix the problem of unable to switch TBF policies directly

- LU-5620 ptlrpc: Add QoS for opcode in NRS-TBF
  - Add TBF policy based on opcode of RPC

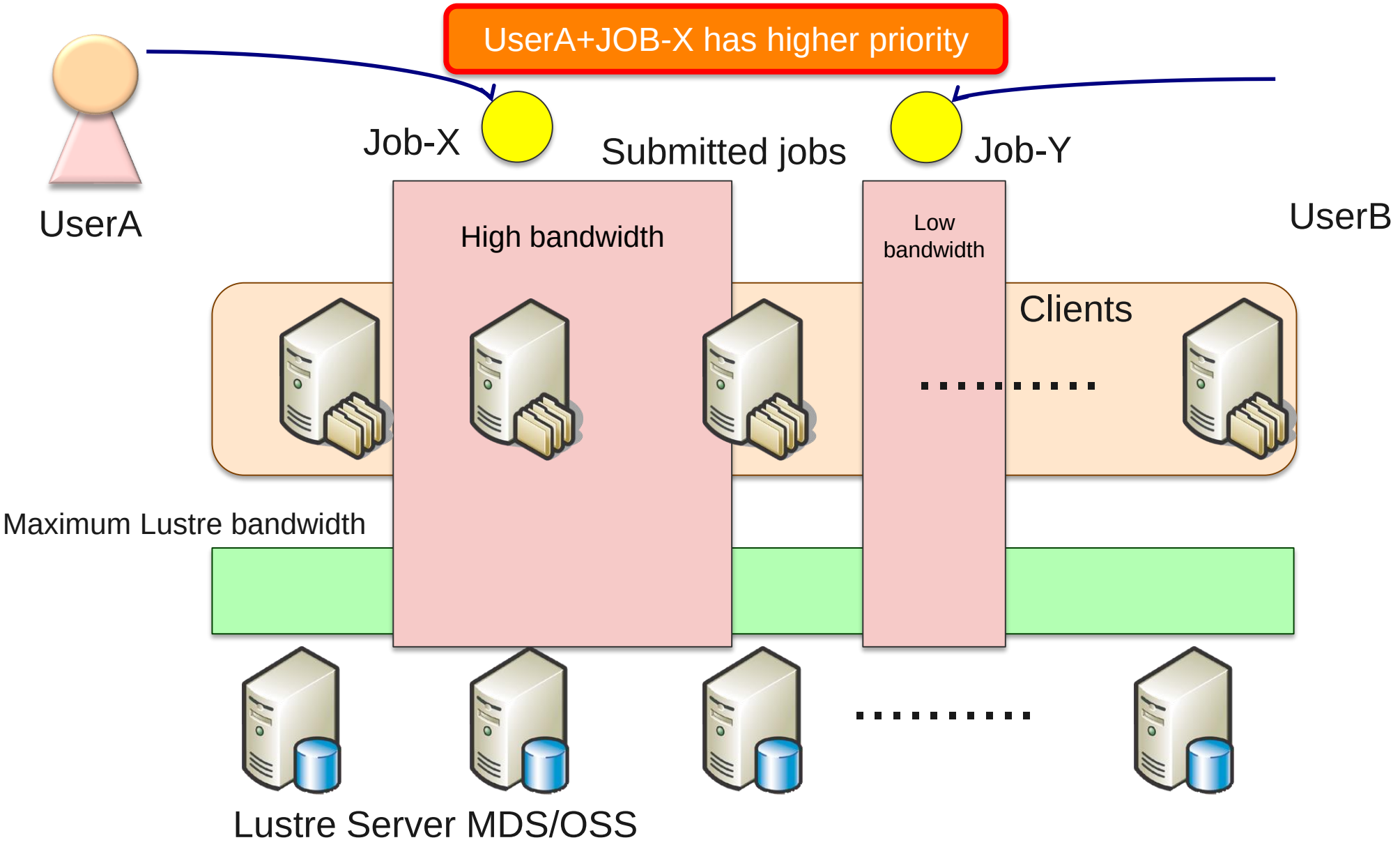
Use cases

- variants of TBF policy have been developed for various use cases
- TBF policy based on NID
  - Manage performance distribution between clients
- TBF policy based on Job ID
  - Manage performance distribution between jobs
  - Could be integrated with job management systems
- TBF policy based on Opcode
  - Manage performance distribution between different kinds of operations

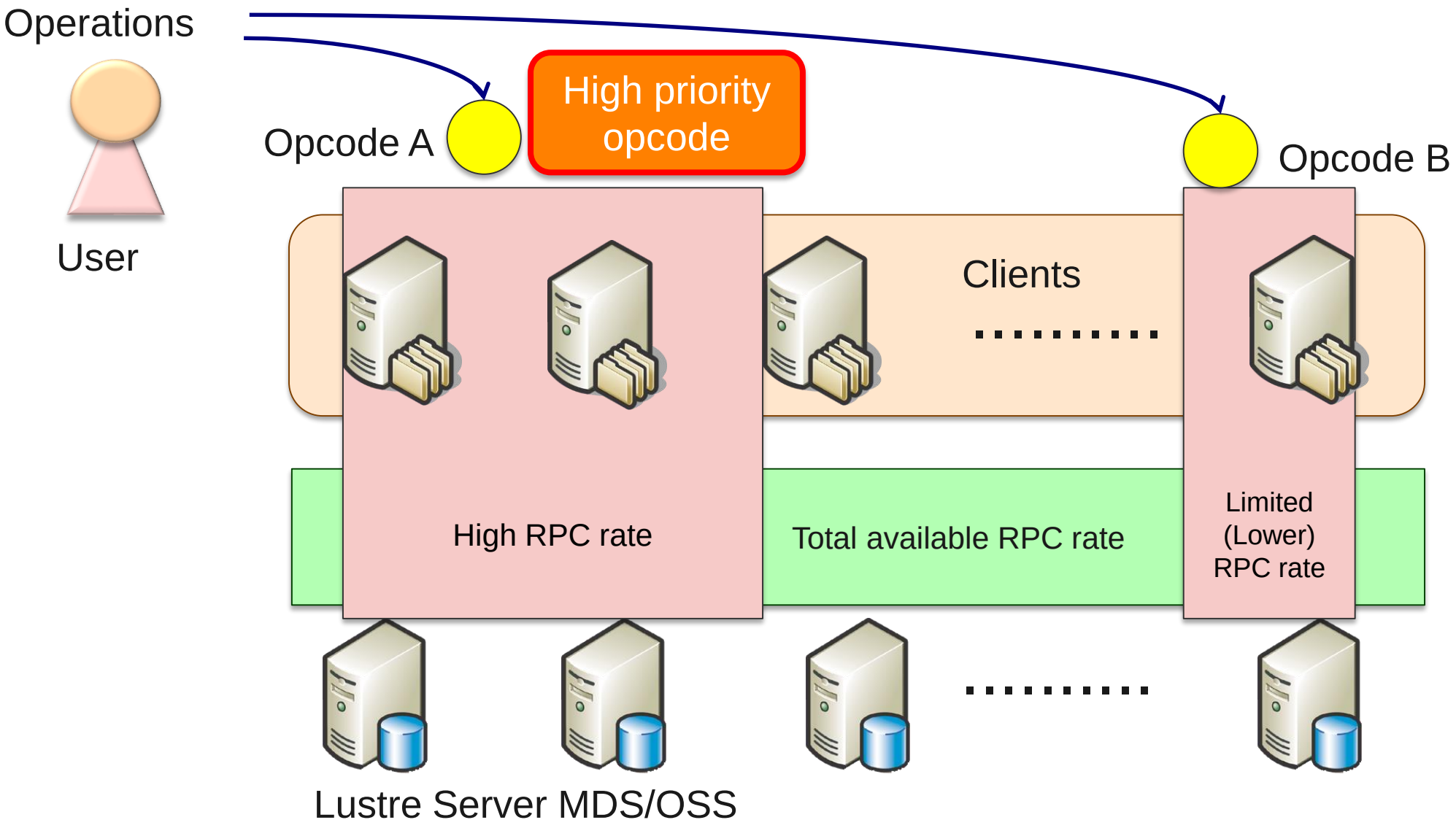
Use Case #1
Lustre QoS based on NIDs (Clients)



Use Case #2
Lustre QoS based on JOBID



Use case #3
Lustre QoS based on Opcode



How to use TBF policies

- ***Change NRS policy to TBF with NID***

 # lctl set_param ost.OSS.ost_io.nrs_policies="<NRS policy> <TBF argument>"

lctl set_param ost.OSS.ost_io.nrs_policies="tbf nid"

- ***Set rule with classification and number of token rate***

 # lctl set_param ost.OSS.ost_io.nrs_tbf_rule="start <TBF's rule name> {NID} <rate>"

lctl set_param ost.OSS.ost_io.nrs_tbf_rule="start rule_client1 {192.168.1.1@o2ib} 1"

lctl set_param ost.OSS.ost_io.nrs_tbf_rule="start rule_clients {192.168.1.[2-16]@o2ib} 10"

- ***Change number of token rate***

 # lctl set_param ost.OSS.ost_io.nrs_tbf_rule="change <TBF's rule name> <new rate>"

lctl set_param ost.OSS.ost_io.nrs_tbf_rule="change rule_client1 100"

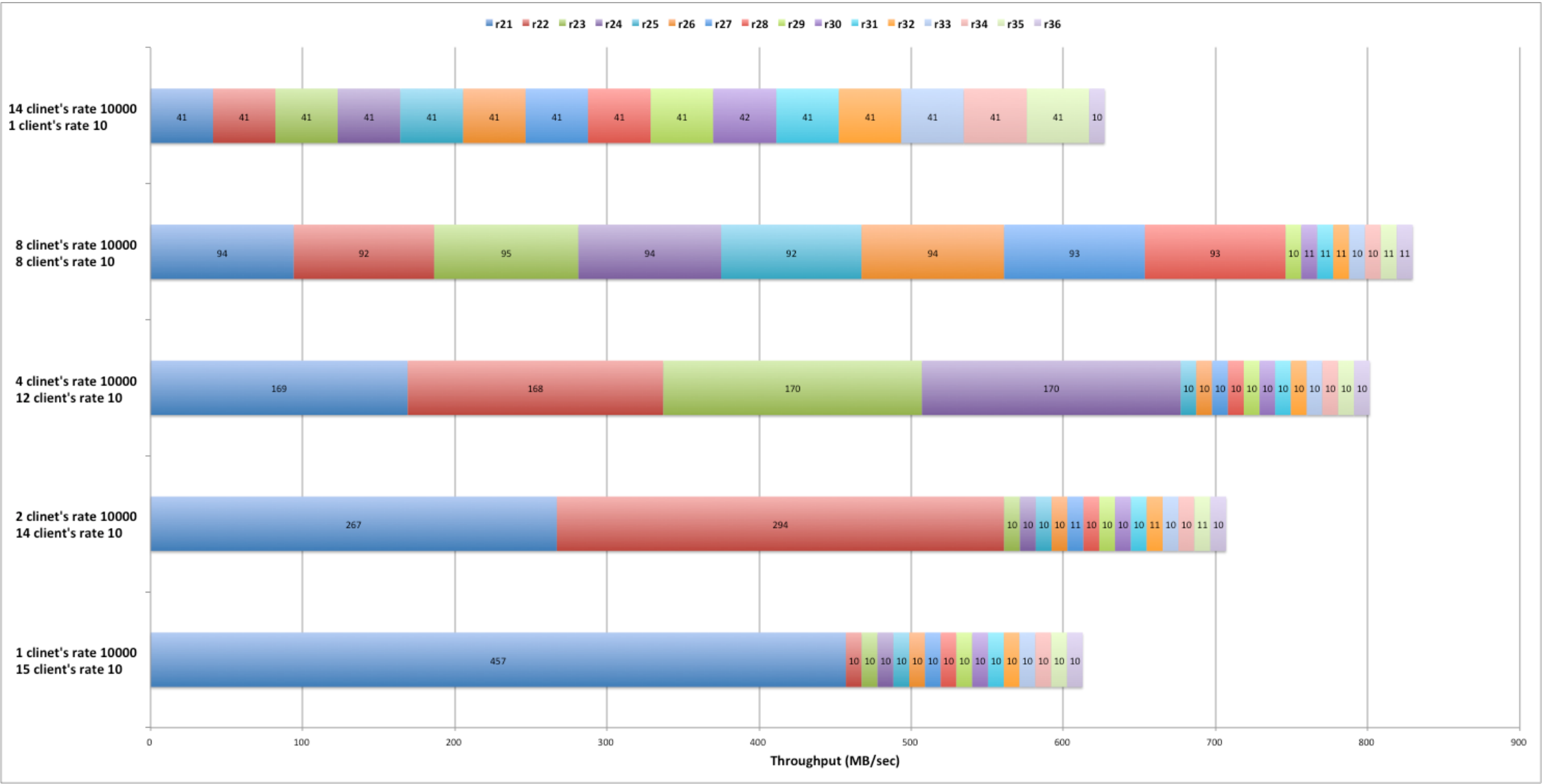
- ***Stop a rule (delete)***

 # lctl set_param ost.OSS.ost_io.nrs_tbf_rule="stop <TBF's rule name>"

lctl set_param ost.OSS.ost_io.nrs_tbf_rule="stop rule_client1"

Test result #1: Lustre QoS based on TBF NID

"dd" command to single OST from multiple clients with various QoS rules (Write, 1MB IO, max_rpc_in_flright=32)



Test result #2: Lustre QoS based on TBF JOBID

Start JOBstats and chang NRS policy to TBF with JOBID

lctl set_param jobid_var=procname_uid
lctl set_param ost.OSS.ost_io.nrs_policies="tbf jobid"

Set rule with classification and number of token

 # lctl set_param ost.OSS.ost_io.nrs_tbf_rule="start <TBF's rule name> {JOBID} <rate>"

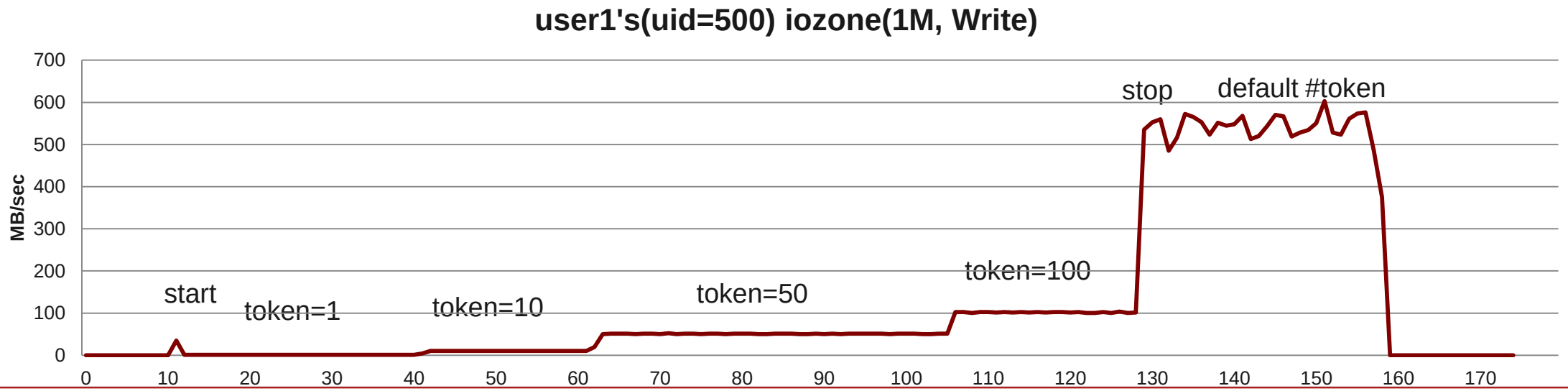
lctl set_param ost.OSS.ost_io.nrs_tbf_rule="start iozone_user1 {iozone.500} 1"

Change number of token

lctl set_param ost.OSS.ost_io.nrs_tbf_rule="change iozone_user1 X" (change X to 10,50 and 100)

Stop a rule (delete)

lctl set_param ost.OSS.ost_io.nrs_tbf_rule="stop iozone_user1"



Test result #3: Lustre QoS based on TBF OPCODE

Chang NRS policy to TBF with OPCODE

lctl set_param jobid_var=procname_uid
lctl set_param ost.OSS.ost_io.nrs_policies="tbf opcode"

Set rule with classification and number of token

 # lctl set_param ost.OSS.ost_io.nrs_tbf_rule="start <TBF's rule name> {OPCODE} <rate>"

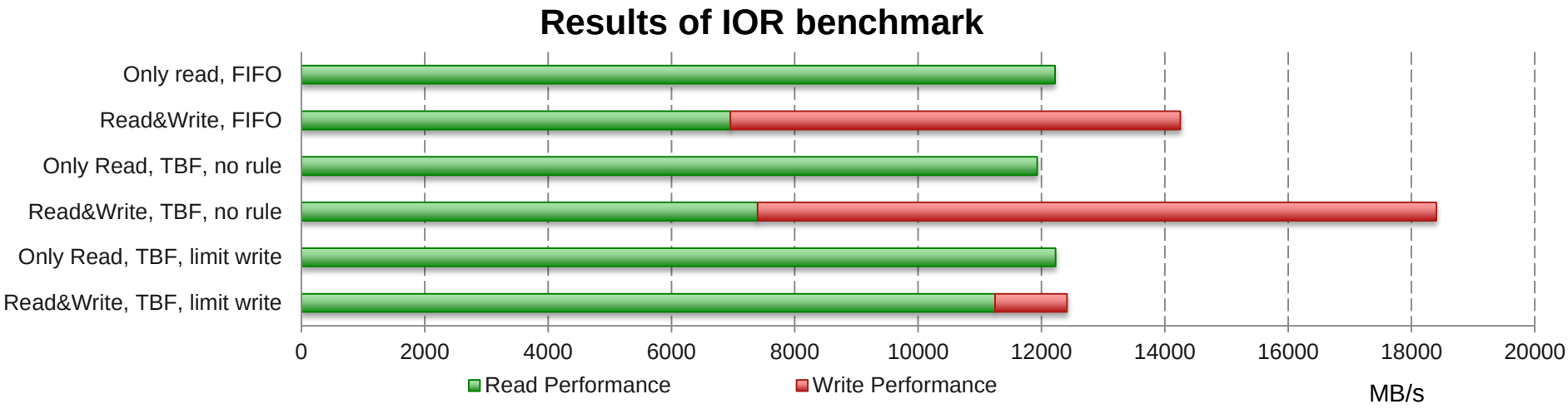
lctl set_param ost.OSS.ost_io.nrs_tbf_rule="start write_limit {ost_write} 50"

Change number of token

lctl set_param ost.OSS.ost_io.nrs_tbf_rule="change write_limit X" (change X to 10,50 and 100)

Stop a rule (delete)

lctl set_param ost.OSS.ost_io.nrs_tbf_rule="stop write_limit"



Summary

- We present an server side QoS mechanism which combined the Lustre Network Request Scheduler (NRS) framework and traditional Token Bucket Filter (TBF) algorithm
- Variants of this policy have been developed for different requirements and purposes and have proved to be useful respectively in different use cases.
- Further work
  - Multi-layered NRS framework
  - Client side QoS

Thank you!