Whamcloud

LIME: A Framework for Lustre Global QoS Management

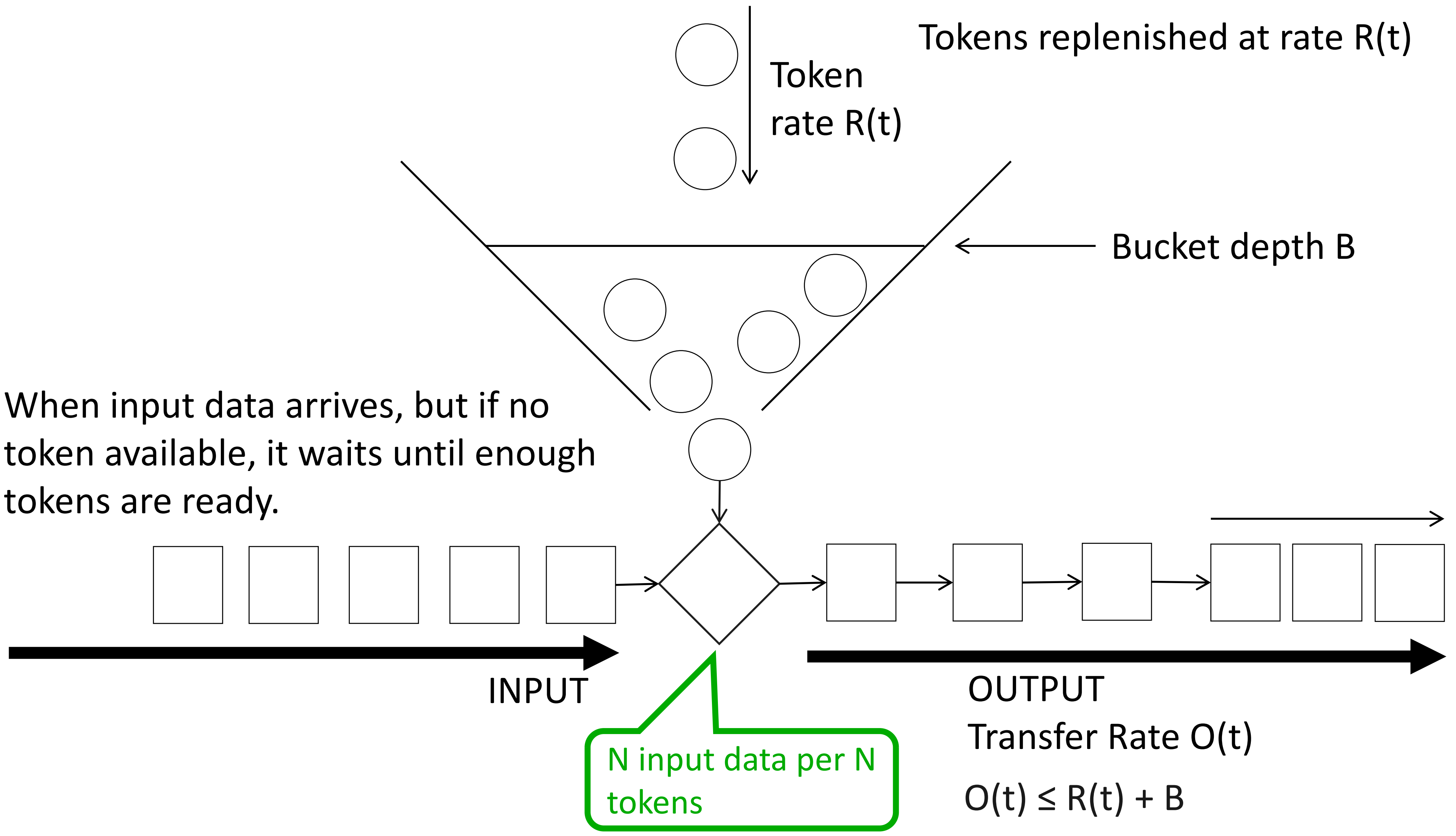
Li Xi – DDN/Whamcloud
Zeng Lingfang - JGU

DDN® STORAGE

Why QoS of Lustre?

- Quality of Service (QoS) is a mechanism to ensure a "guaranteed” performance
  - “ls” latency needs to be short for good experience
  - Some applications have fixed I/O timeout
  - Data stream keeps on flowing into storage continuously with a constant rate
- TBF has been improved continuously for this purpose
  - Different TBF types: UID/GID/NID/JobID/Opcode/General
  - Newly implemented features: Hard Token Compensation strategy, change rule order
  - A large group of OSTs/MDTs to manage
- A paper has been published to summarize the work
  - A Configurable Rule based Classful Token Bucket Filter Network Request Scheduler for the Lustre File System,
- Users are starting to use it
  - Congestion of Lustre happens less than before, but still happens
  - AI users starts to use TBF to prevent congestion of MDT

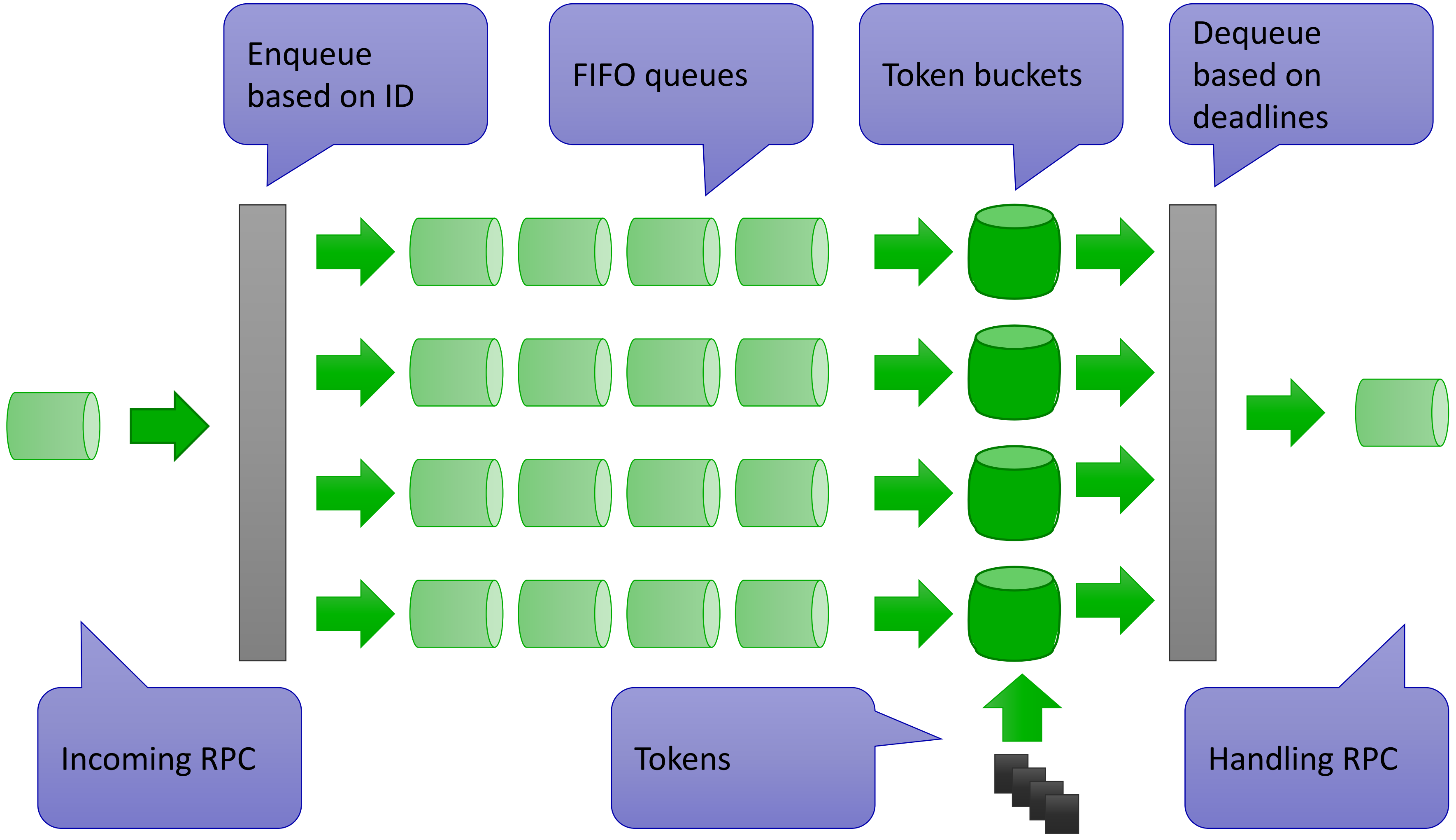
The Token Bucket Filter (TBF)



Tokens replenished at rate R(t)
Token rate R(t)
Bucket depth B
When input data arrives, but if no token available, it waits until enough tokens are ready.
INPUT
N input data per N tokens
OUTPUT
Transfer Rate O(t)

$O(t) \leq R(t) + B$

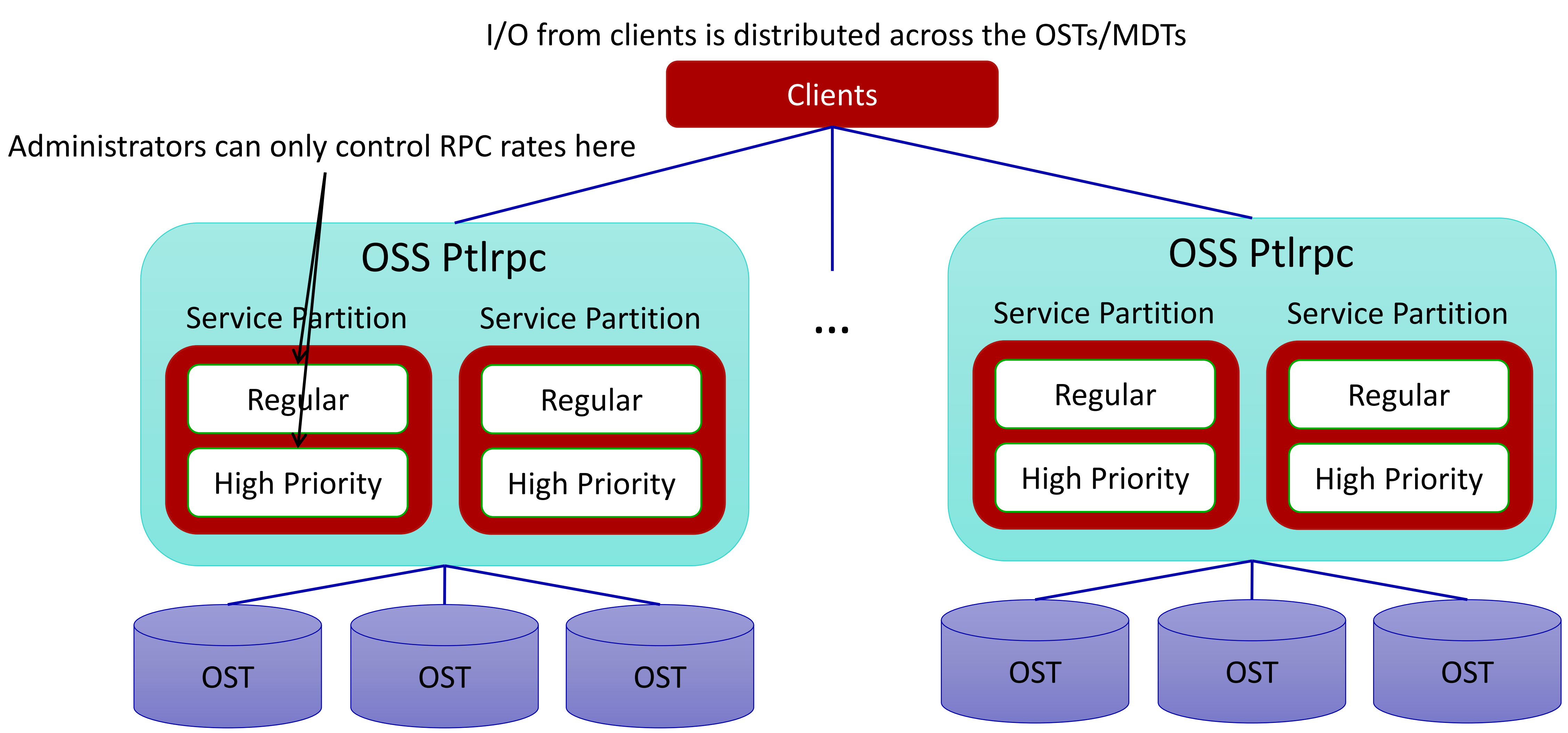
The TBF Implementation for Lustre



Limitations of Current TBF Policy

- TBF is able to control individual OST/MDT, but no global management
  - I/O of applications are distributed across all OSTs/MDTs
- TBF can throttle I/O performance, but can not guarantee performance
  - Some applications need guarantee of I/O performance
- TBF can only limit RPC rate, not bandwidth or IOPS itself
  - Multiple MDTs/OSTs can be attached to a single MDS/OSS
  - Each MDS/OSS has several PTLPRC service partitions
  - Each service partition has high-priority NRS head and regular NRS head
  - TBF can limit the RPC rate of a certain classification on each NRS head
  - The mapping of RPC rate to bandwidth/IOPS depends on all of these factors
- Administrators need a global QoS mechanism
  - Simplified interface
  - Automatic management

Why Global QoS is not Easy?



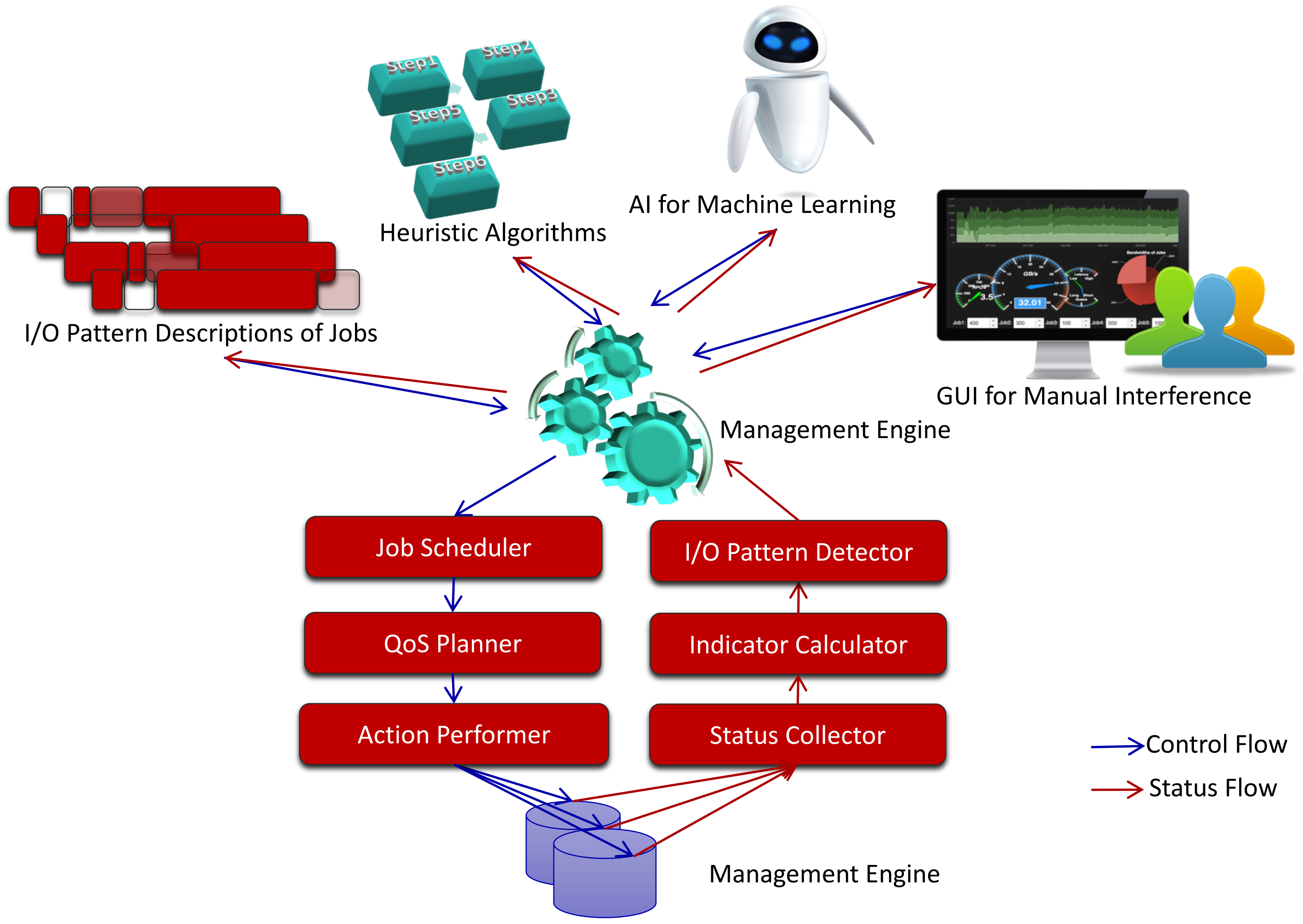
What is Needed for a Better QoS of Lustre?

- Basic mechanisms inside Lustre
  - Implemented: NRS TBF policy
  - Implementing: LU-9809 QoS policies for object allocation that can be controlled by external tools
  - Implementing: LU-7982 Client side QoS based on jobid
- A global performance monitoring system
  - Analysis of I/O patterns
  - Summarize statistics
- A centralized management framework
  - Configure global TBF rules on all OSS/MDS
  - Make decisions according to statistics
  - Enforce consistent policies across the whole file system
- Collaboration from users of the file system
  - Users should have enough motivation to optimize their application
  - Penalty will be enforced for bad behaviors
  - High-priority users/application have higher I/O rates

LIME: Lustre Intelligent Management Engine

- https://github.com/DDNStorage/Lime
- Lustre statistics collector based on Collectd
  - Supports different Lustre versions: 1.8/2.5/2.7/2.10/2.12/…
  - Collects all kind of statistics from Lustre /proc or /sys entires
- Time-series database based on Influxdb
  - Several other choices for time-series databases: Opentsdb
  - LIME can query the database for statistics during a time period
- Monitoring GUI based on Grafana
  - Grafana is more powerful and flexible than most of the other analytics and monitoring GUIs
- System management framework
  - The control center can SSH to a cluster of nodes and execute commands
- Different QoS Policies for different purposes
  - “Decay” Policy to enforce a throughput/IOPS quota

The framework of LIME



→ Control Flow

→ Status Flow

QoS Warning Message on Client

- When a process's I/O is being throttled by a TBF rule on server side, warning messages will be printed to its TTY
- Warning messages can be enabled or disabled when defining TBF rule
- The printing rate of warning messages can be tuned (1 message per 10 seconds)
- Message examples:
  - QoS watermark limit of uid "0" has been reached. Reducing RPC rate of process with pid "2417" according to the rule ”uid_0".
  - QoS watermark limit of group id "0" has been reached. Reducing RPC rate of process with pid "4378" according to the rule ”gid_0".
  - QoS watermark limit of nid "0@<0:0>" has been reached. Reducing RPC rate of process with pid "27384" according to the rule ”nid_local".
  - QoS watermark limit of jobid "dd.0" has been reached. Reducing RPC rate of process with pid "28601" according to the rule ”jobid_dd_0".
  - QoS watermark limit of opcode "ost_write" has been reached. Reducing RPC rate of process with pid "27378" according to the rule ”ost_write".
- Patch: LU-11192 ptlrpc: console warning for TBF on client

Decay policy of LIME

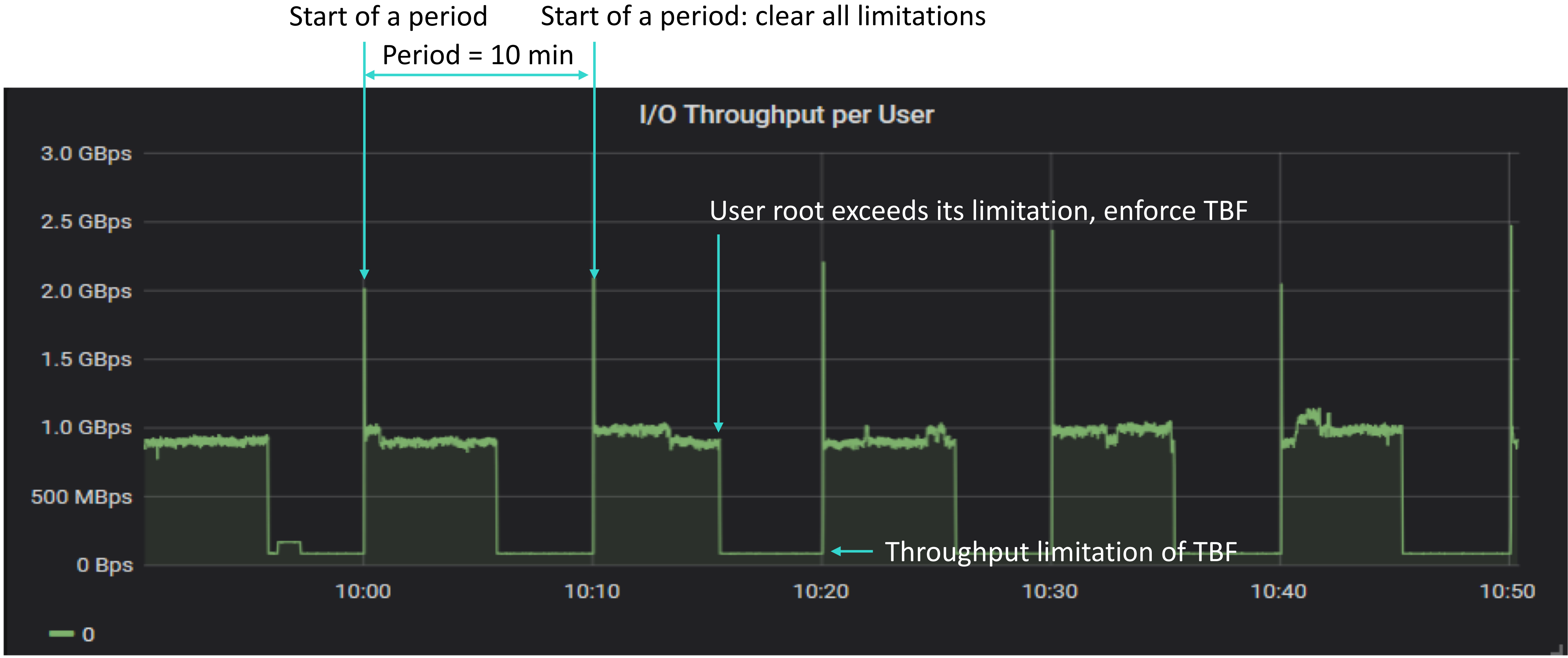
- Time period of 24 hours
  - Time period could be changed to one hour, one week or one month
- Throughput/IOPS will be recorded for all users
  - Performance monitoring system is used to collect the usage of throughput/IOPS
  - Influxdb commands are queried to get the throughput/IOPS of each user during this time period
- Upper limitation of throughput/IOPS for each user
  - Different users can have different upper limitations
  - If a user reaches the limitation, TBF rules will be enforced for that user on all OSTs/MDTs
- At the beginning of each time period, all the TBF limitation will be removed
- Use QoS warning to notify users when throttling their I/O rate
- Extension: hard limitation and soft limitation
  - Soft limitation <= hard limitation
  - TBF rules of hard limitation are stricter than TBF rules of soft limitation

Configuration Example of Decay Policy

```
LPMon_server_hostname: server17    # Hostname of Lustre Performance Monitoring server
LPMon_collect_interval: 1          # Collect interval of Lustre Performance Monitoring in seconds
enabled: true                      # Whether QoS management is enabled
interval: 600                      # QoS interval in seconds
mbps_threshold: 70                 # mbps_threshold * interval is the throughput limit of MB
throttled_oss_rpc_rate: 10         # Default RPC per second on each OSS partition
iops_threshold: 100                # iops_threshold * interval is the metadata operation limit
throttled_mds_rpc_rate: 10         # Default RPC per second on each MDS partition
users:
  - uid: 0
    mbps_threshold: 500            # Overwrites global mbps_threshold for this user
    iops_threshold: 5000           # Overwrites global iops_threshold for this user
    throttled_oss_rpc_rate: 20     # Overwrites global throttled_oss_rpc_rate for this user
    throttled_mds_rpc_rate: 20     # Overwrites global throttled_mds_rpc_rate for this user
```

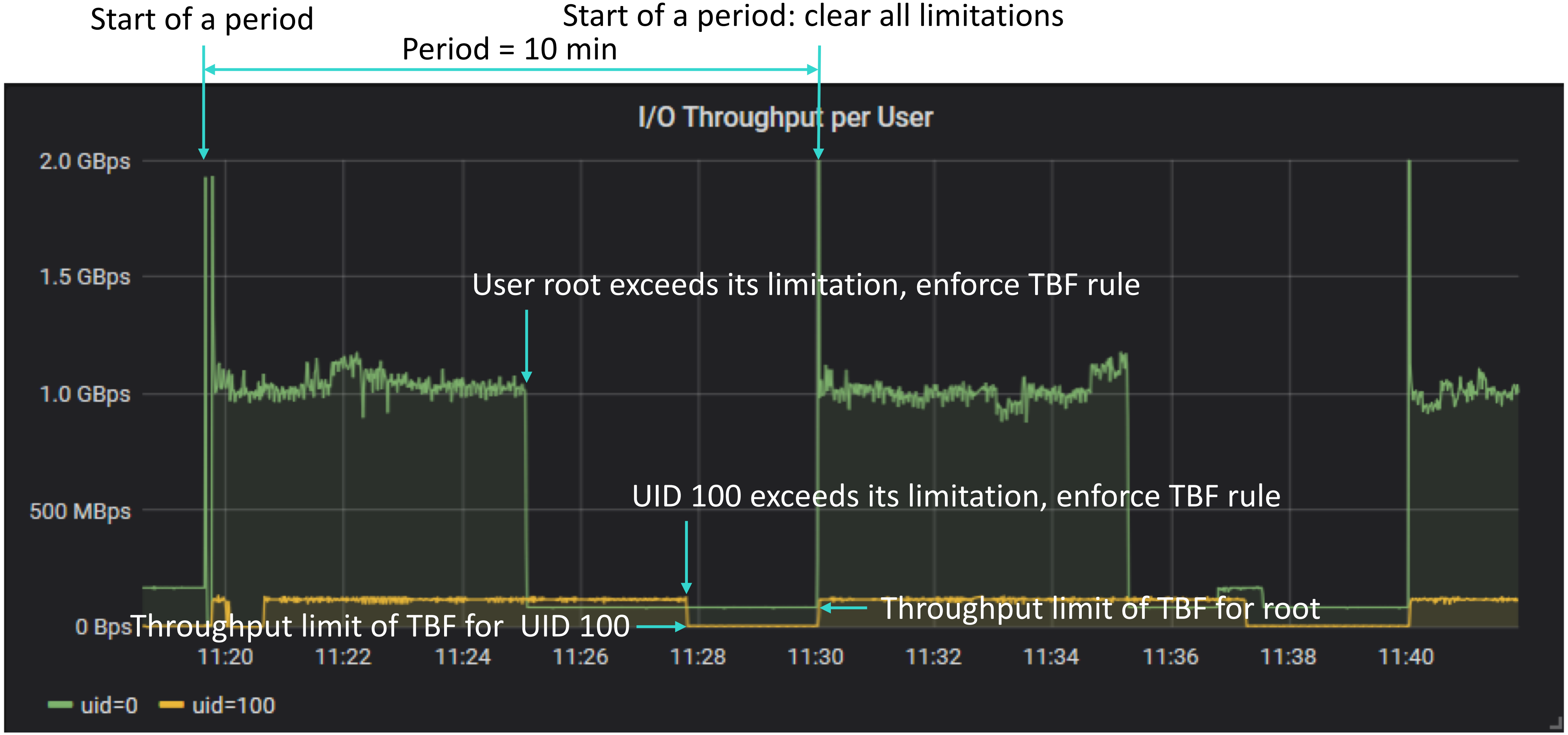
Test Result of Decay Policy – I/O throughput(1)

I/O pattern: dd if=/dev/zero of=/lustre/file bs=1048576



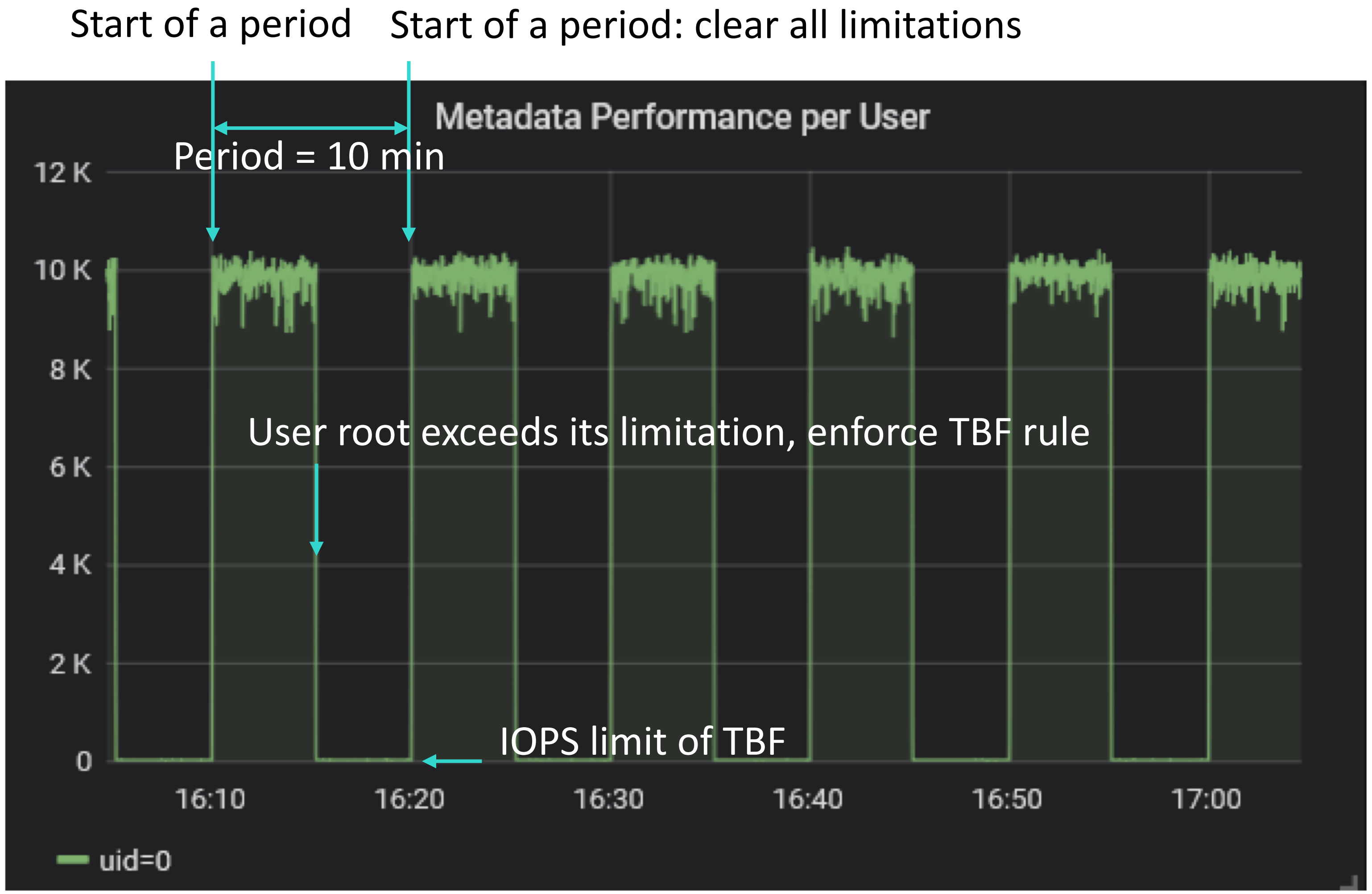
Test Result of Decay Policy – I/O throughput(2)

I/O pattern: dd if=/dev/zero of=/lustre/file bs=1048576



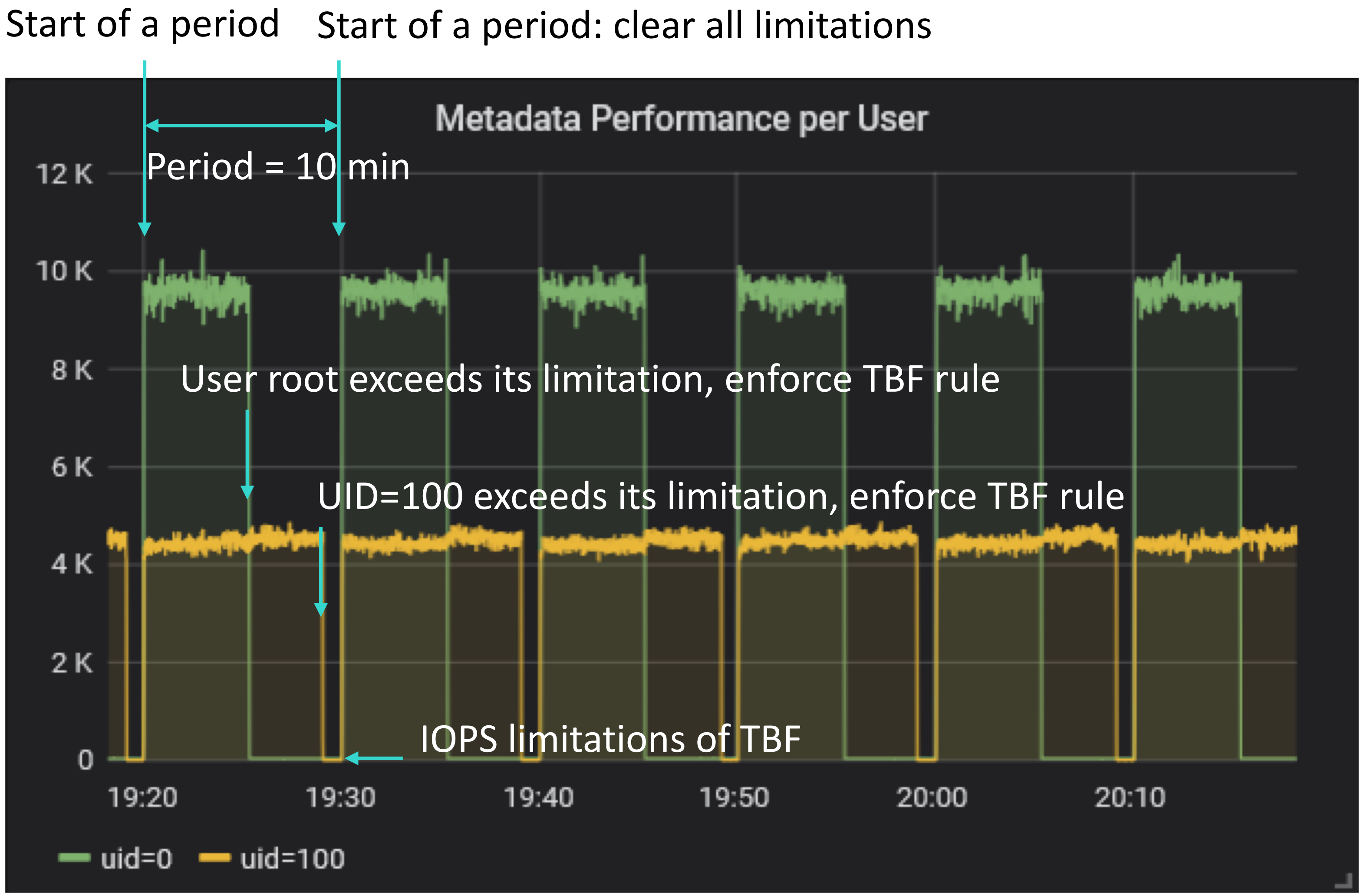
Test Result of Decay Policy – Metadata Performance(1)

I/O pattern: repeatedly create and remove files



Test Result of Decay Policy – Metadata Performance(2)

I/O pattern: repeatedly create and remove files



Why Decay Policy Looks Promising?

- Simple
  - Easy to tune the parameters to proper values
- Comprehensible
  - Similar to the semantics of capacity/inode quota
- Clear consequence
  - Constant I/O rate limitation is the penalty of exceeding the limitation
- Little dependency
  - Works well on any file system with any performance
- No limitation of I/O patterns
  - Applications can choose how to use the credit
- Easy for users to react
  - Any optimization to reduce I/O would help to avoid exceeding the quota
- No negative impact for innocent users
  - Users who have little I/O are not impacted by throttling mechanism

Other policies

- Strategy
  - Only try to achieve a single target at one time, do not mix problems together
  - Define policies that can be used together at the same time
- Penalty policy for burst I/O
  - Motivation: some ugly applications cause congestion of the whole system
  - Throttle the I/O of the user/job for a short time period and then relieve it
- Congestion-control policy for a MDT
  - Use latency of ‘ls -l’ as an indicator of congestion
  - In order to eliminate interference from OSTs, create files with no OST object (mknod)
  - When large latency detected, throttling all operations except RPC with opcode of “ldlm_enqueue”
- Congestion-control policy for an OST
  - Latency of “ls -l” is an good indicator of congestion too
  - In order to eliminate interference from MDTs, need to add a new RPC on OSC
    - Get the sizes of a list of objects on OSTs and calculate the latency of it
  - When large latency detected, throttling all operations except the RPC of a specific opcode

Future work

- Testing and tuning of LIME policies
  - Policies should work on all conditions
- Client side QoS: LU-7982
  - Fix the problem that different jobs/users running on the same client affect each other
  - Balance usage of page cache and RPC slot
- Lustre object allocation policy on MDT: LU-9809
  - RTDS(Real-Time Dynamic Striping): A policy based striping framework
  - Use LIME to control the striping policy so as to achieve QoS goals
- Choose Lustre pool to place file's objects by user-defined policies: LU-11234
  - Useful for storage tiering
  - SSD pools for cache
  - Data placement for Lustre on demand
  - Use LIME to control the policy so as to achieve QoS goals

Whamcloud

DDN® STORAGE