Lustre QoS solutions
based on NRS TBF and client side performance balancing

Li Xi

DataDirect Networks

Why need solutions of QoS?

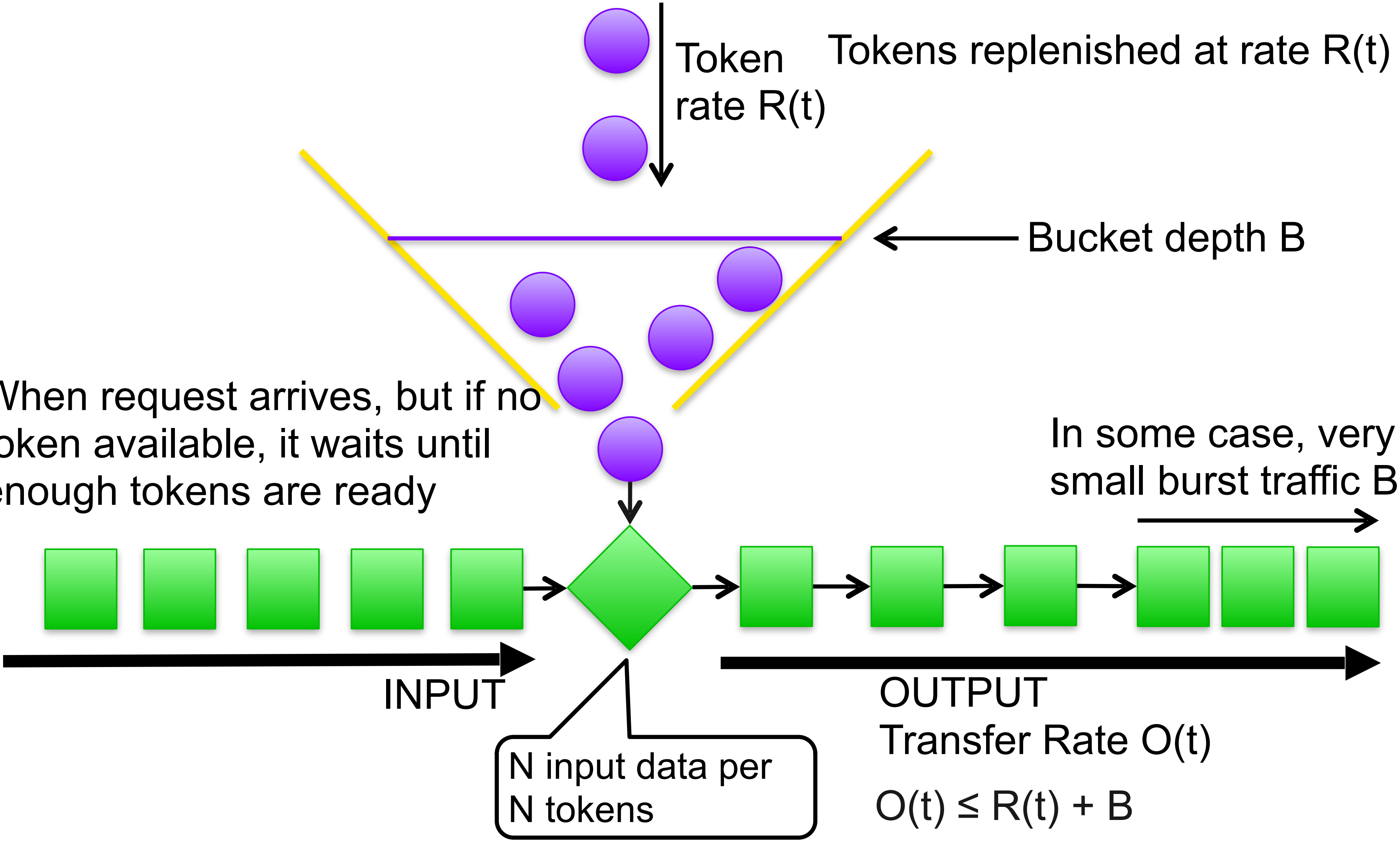
- **Lustre is able to provide scalable throughput and IOPS**
- **Storage systems in HPC centers are usually shared by multiple organizations and various applications**
  - Applications occupies as much bandwidth as possible from the shared storage without regard for the interests of others
- **Various QoS solutions are necessary to satisfy different requirements of performance guarantee**
  - Prevent crazy applications that congest the storage
  - Improve user experience, e.g. intolerable delay of 'ls'
  - Assure workloads of reliable bandwidth
  - Enable use cases outside the mainstream HPC, e.g cloud
- **QoS is still a fresh topics for most file systems**
- **But Lustre already has a series of solutions for QoS!**

Original Solution: TBF policy of NRS

- **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 throttle RPC rates**
  - RPCs are classified according to NID/Job ID
  - The rates of RPC classifications can be throttled to a rate limitation
  - Rules can be configured by administrator to adjust RPC limitations in run time
  - Examples:

```
# lctl set_param ost.OSS.ost_io.nrs_policies="tbf nid“
# lctl set_param ost.OSS.ost_io.nrs_tbf_rule="start rule_clients
  {192.168.1.[2-16]@o2ib} 10"
```

Algorithm of TBF



Token
rate R(t)

Tokens replenished at rate R(t)

Bucket depth B

When request arrives, but if no token available, it waits until enough tokens are ready

In some case, very small burst traffic B

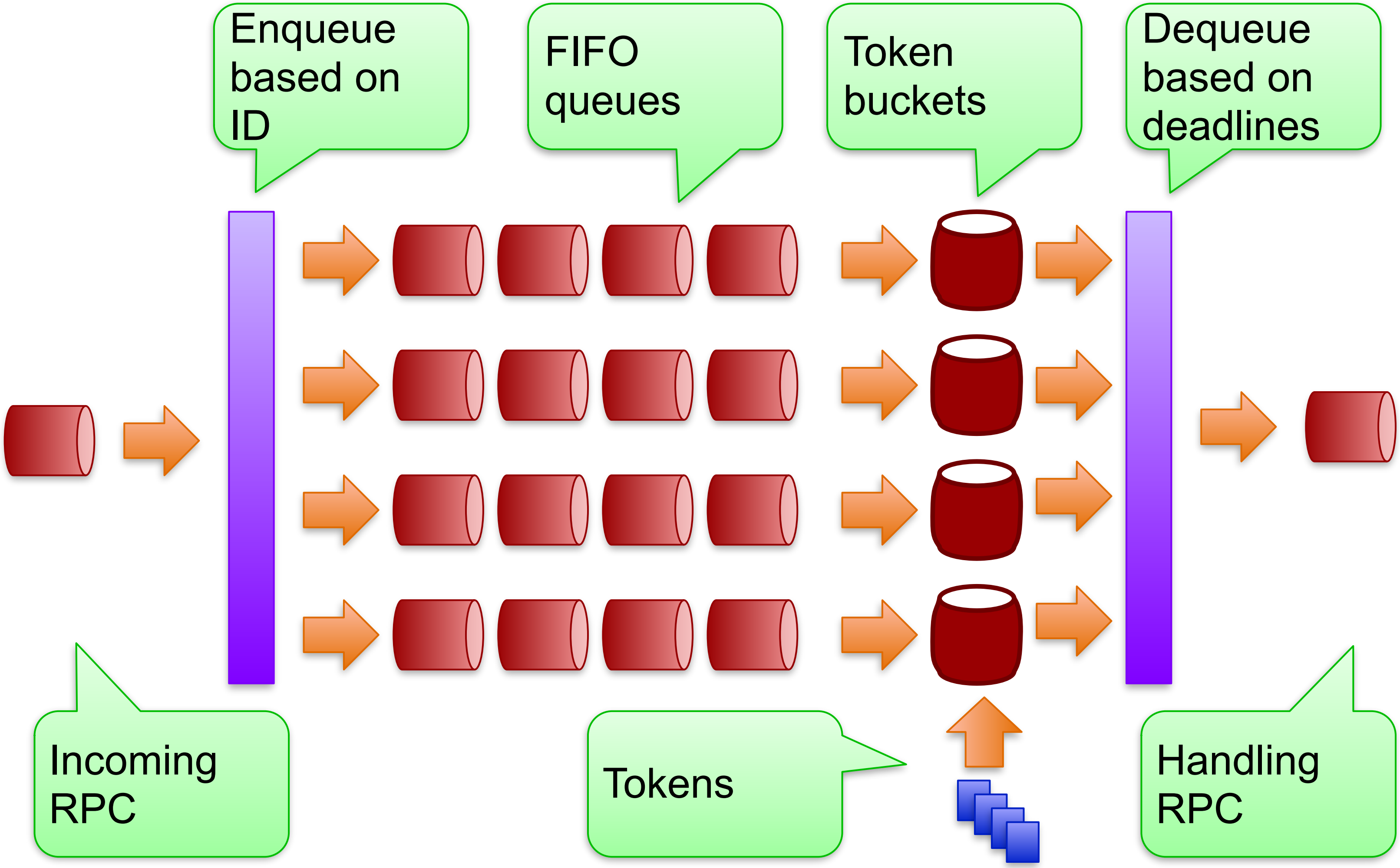
INPUT

N input data per
N tokens

OUTPUT
Transfer Rate O(t)

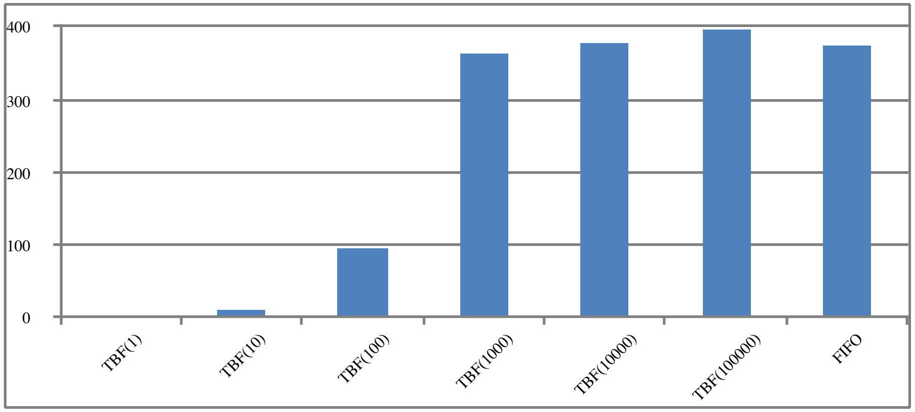
$O(t) \leq R(t) + B$

TBF policy of NRS

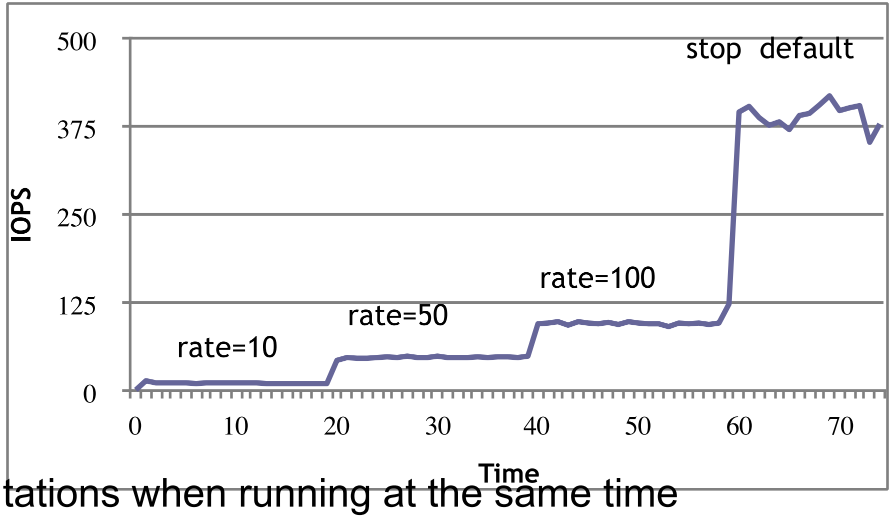


Test results of TBF policy

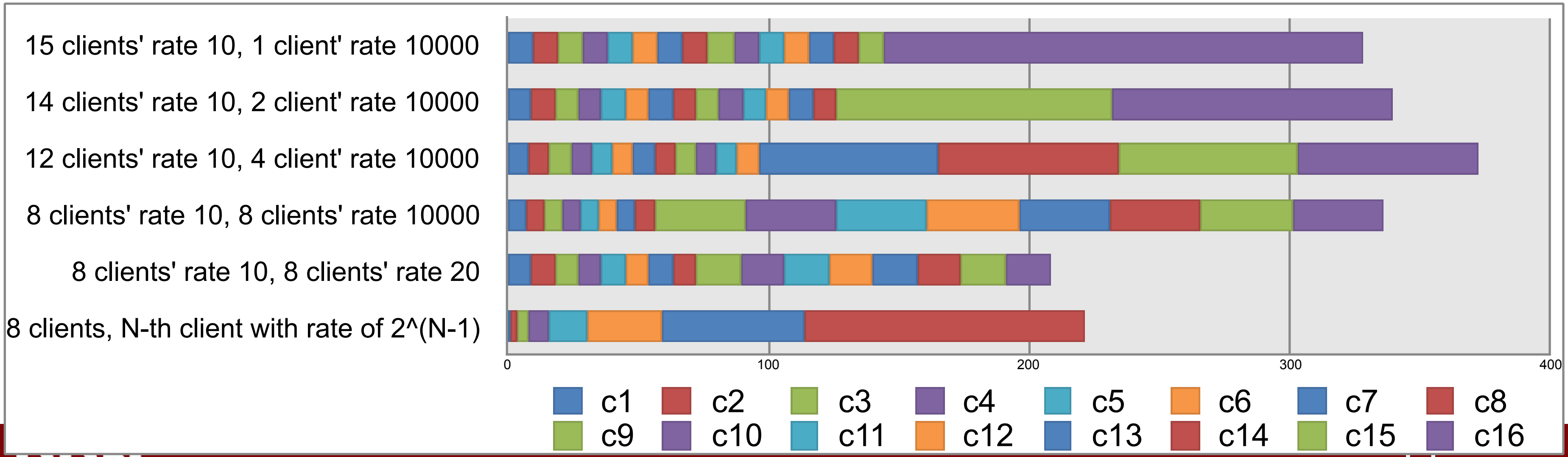
I/O performance with different RPC rate limitations



Real time change of IOPS when changing RPC rate



IOPS of clients with different RPC rate limitations when running at the same time



7

Extended Solution: Dependency rule of TBF

Limitation of original TBF:

- Not able to adapt the RPC rates dynamically according to the load of service
- RPCs with low rate limitation can’t use utilize the available bandwidth even the load of server is light

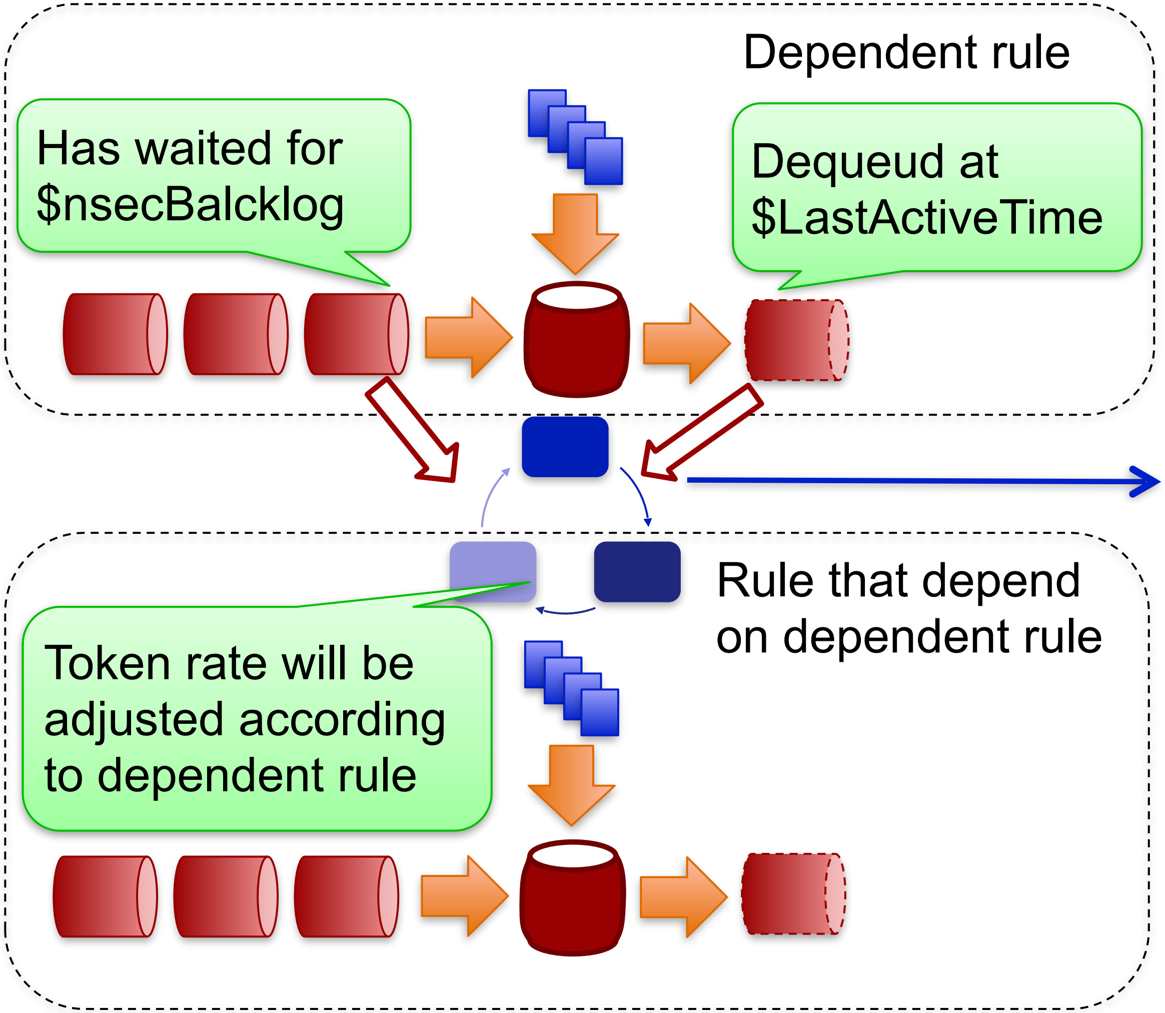
Solution: LU-8433

- A the RPC rate rule can be depend on the real time rate of another rule
- Example:

```
start ruleB <matchCondition> deprule=ruleA lowerrate=$r1 upperrate=$r2
```

- If any classes of rule A is not able to reach its RPC rate limitation which means the service is under too heavy load, classes of rule B will decrease its rate until the lower limitation
- Otherwise, the rate limitation of rule B will be increase until the upper limitation

Dependency rule of TBF



Dependent rule

Has waited for $nsecBalcklog

Dequeud at $LastActiveTime

Rule that depend on dependent rule

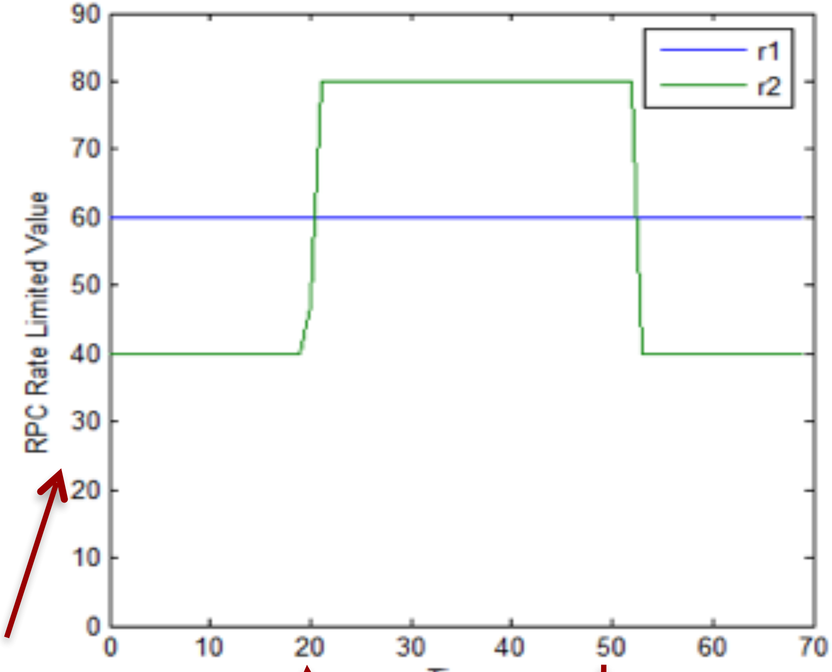
Token rate will be adjusted according to dependent rule

```
Procedure UpdateRuleRate(rule, deprule)
    passed = now – deprule.lastActiveTime;
    if passed > λ* deprule.nsecs &
        deprule.nsecsBacklog < β * deprule.nsecs then
            # increase the RPC rate
            rule.speedup <<= 1
    else
            rule.speedup >>= 1
    end if
    rule.rate = rule.speedup + rule.lowerrate
    cap rule.rate at rule.upperrate
end procedure
```

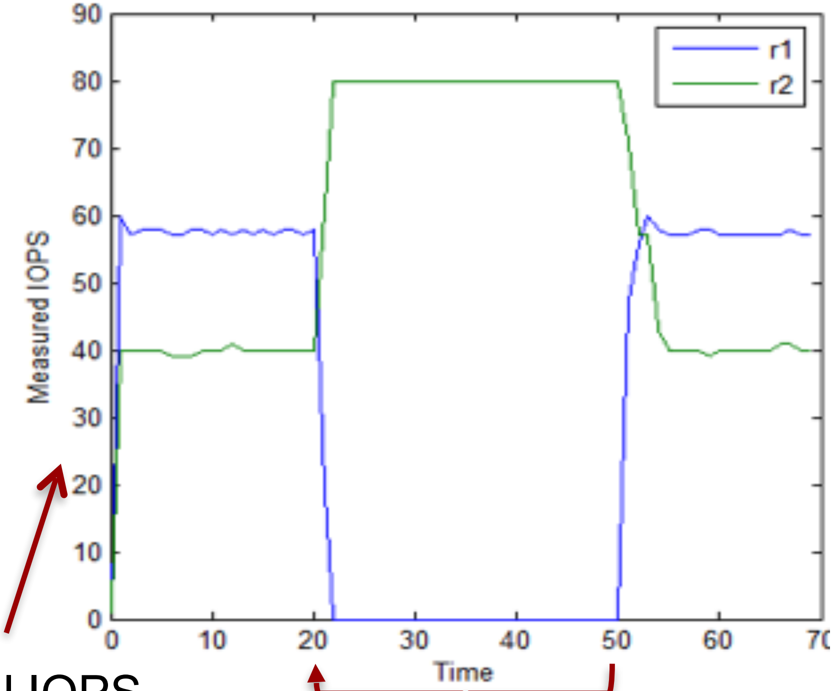
Test result of dependency rule of TBF (1)

Use the NRS evaluation system to test the behavior

- r1: upper limitation of 60 RPC/s, stops I/O during [20s, 50s]
- r2: deprule=r1 lowerrate=40 upperrate=80, consistent I/O



Rate limitation



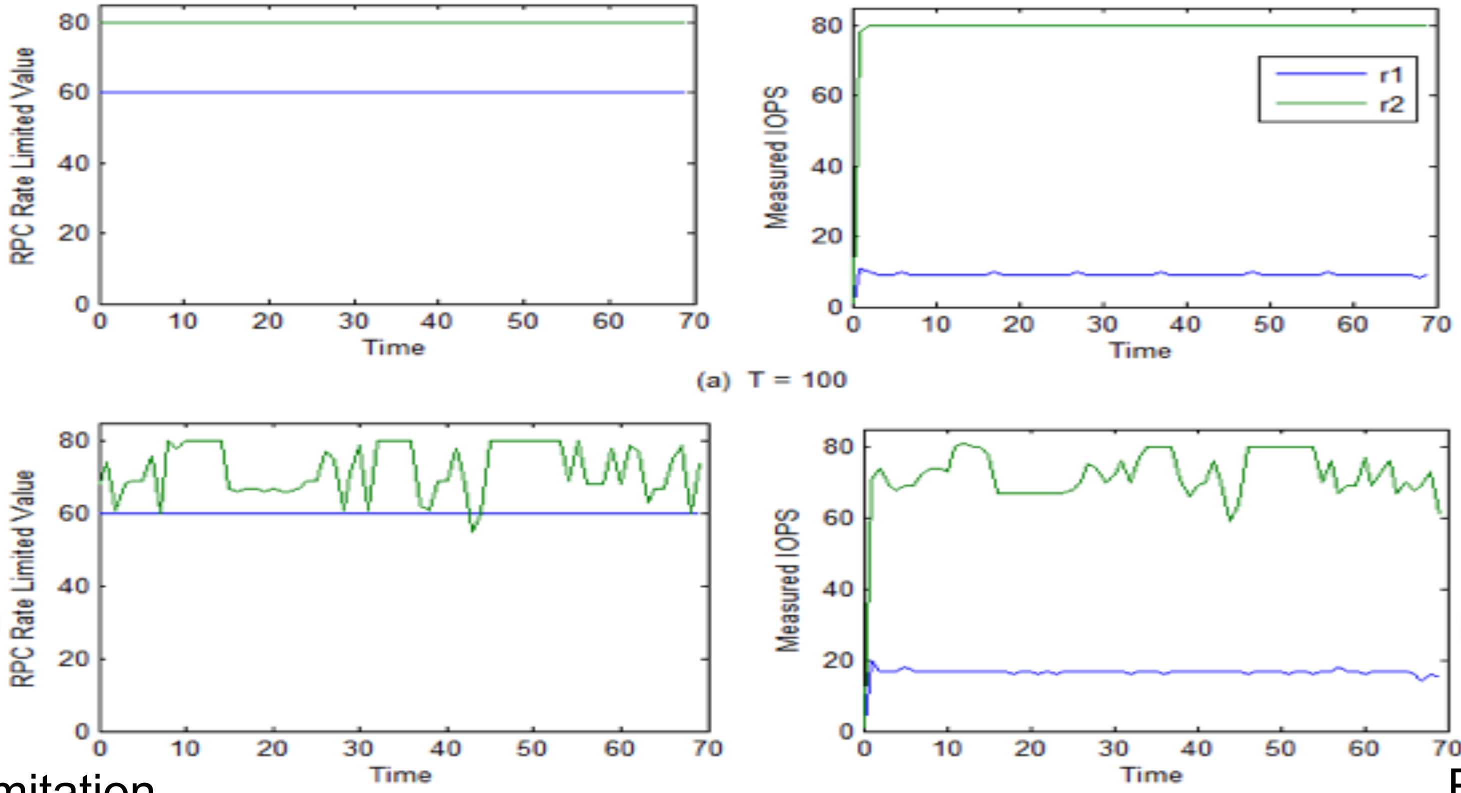
Real IOPS

r1 is not doing any I/O, so the rate limitation of r2 is increase

Test result of dependency rule of TBF (2)

Use the NRS evaluation system to test the behavior

- r1: upper limitation of 60 RPC/s, stops I/O during [20s, 50s]
- r2: deprule=r1 lowerrate=40 upperrate=80, consistent I/O
- The total bandwidth is lower than 100 RPC/s



(a) T = 100

(b) T = 50

Rate limitation

Real IOPS

DDN STORAGE

Extended Solution: Rule Expression of NID + Job ID in TBF

Limitation of original TBF policy:

- The classification can only based on either NID (Network ID) or JobID
- The administrator might want to limit the RPC rate of a specific job on a specific client

Solution: LU-7470

- Rules with expression of NID and Job ID can be configured
- Logical AND and logical OR is allowed to be used in the expression
- Example:

```
JOBID={dd.5}&NID={192.168.1.1@tcp},JOBID={dd.0}&NID={192.168.1.*@tcp} 1000
```

Extended Solution: Advanced Expression of IDs in TBF

Limitation of expression of NID and Job ID

- Only NID and Job ID can be used in the expression
- More IDs (uer, group, operation code, cgroup) might be required for different use cases
- Original expressions are limited to the form of

(A && B) || (C && D) || (E && F) || …

Solution: LU-8674

- A lightweight internal policy engine is implemented for HSM, OST pool migration, file heat, etc. in LU-8674
- The general expression might be able to used for the expression of TBF rule
- Any logical expression can be used, such as:

(((A && B) || C) && D && E) || F …

Client Side QoS Solution: Performance Balancing & Allocation

Limitation of TBF based solutions

- TBF based solutions can only control performance on the server side
- Administrators might want to prevent one application/user exhausts all of the bandwidth of the client, especially on login node
- The problem of JobID based NRS TBF
  - Job ID with high RPC rate limit will get unexpected RPC rate if another job ID has low RPC rate limit
  - The reason is jobs with low RPC rates on server side occupy most of resources on client side
  - This problem can't be fixed without modification of client side codes

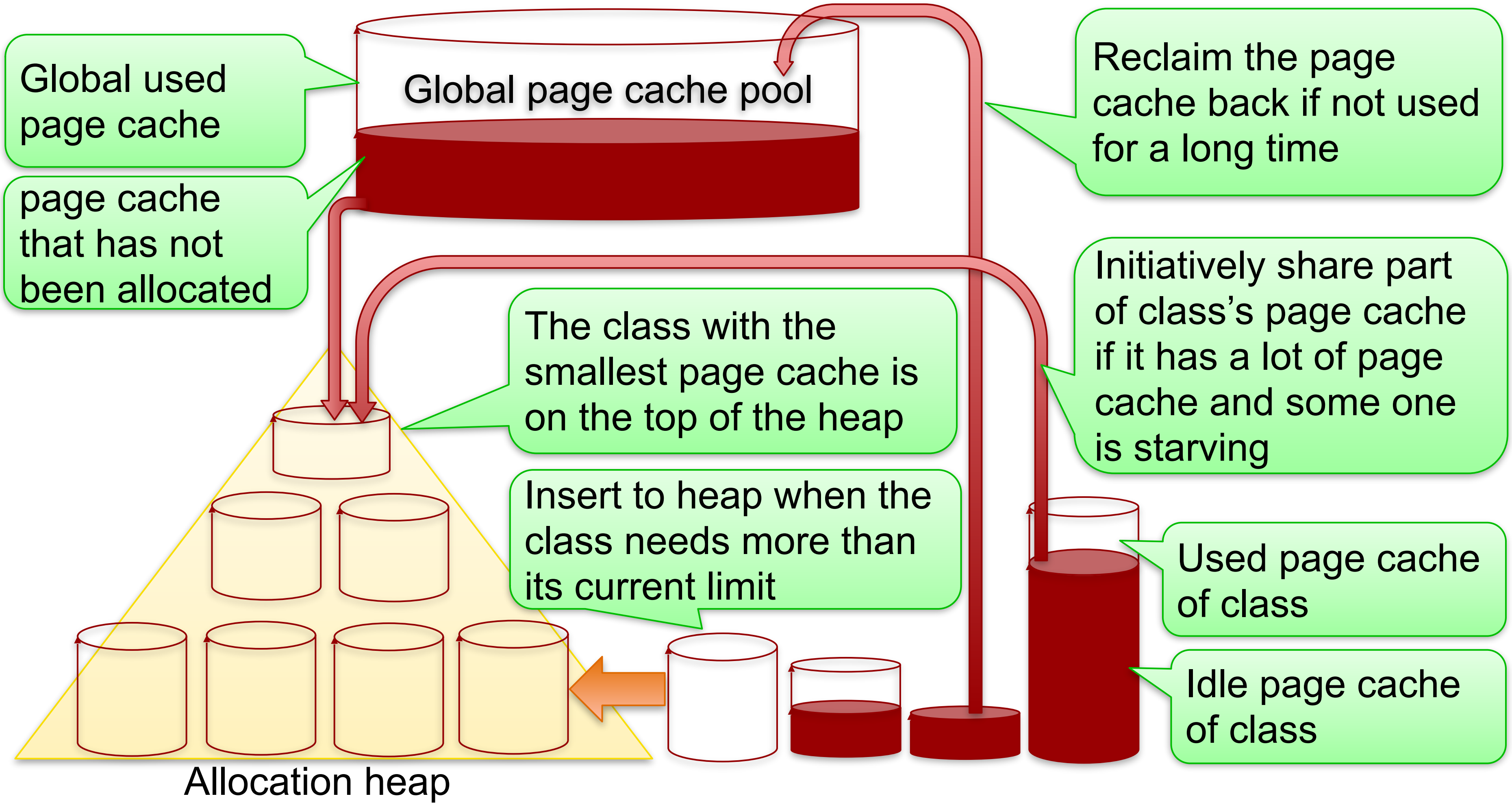
Solution: LU-7982

- Client side QoS framework which can balance/allocate resource slots on a client are between jobs/users
- LU-7982 osc: qos support for page cache usage
- LU-7982 osc: qos support for in flight RPC slot usage

Client Side QoS Solution

- **Resource slots on a client are not always balanced between jobs/users without QoS**
- **Client side QoS is able to balance/control the usage of these resources**
  - I/O requests are classified based on job ID or other IDs
  - Slots are allocated and managed based on classifications
  - **Max In-flight-RPC slots**: needed by all RPCs, i.e. cached read/write and also direct I/O
    - All classifications will be sorted on a heap according to the in flight RPCs
    - The sending RPCs will be generated for the classification on the top of the heap
    - RPC slots are balanced between jobs
  - **Max Readahead size**: needed by cached read
    - Readahead algorithm is completely replaced new framework: Parallel-readahead (LAD2016)
  - **Max Page cache**: needed by cached write
    - Page cache usages are balanced between jobs

Page Cache Usage Balancing



Global used page cache

page cache that has not been allocated

Reclaim the page cache back if not used for a long time

The class with the smallest page cache is on the top of the heap

Insert to heap when the class needs more than its current limit

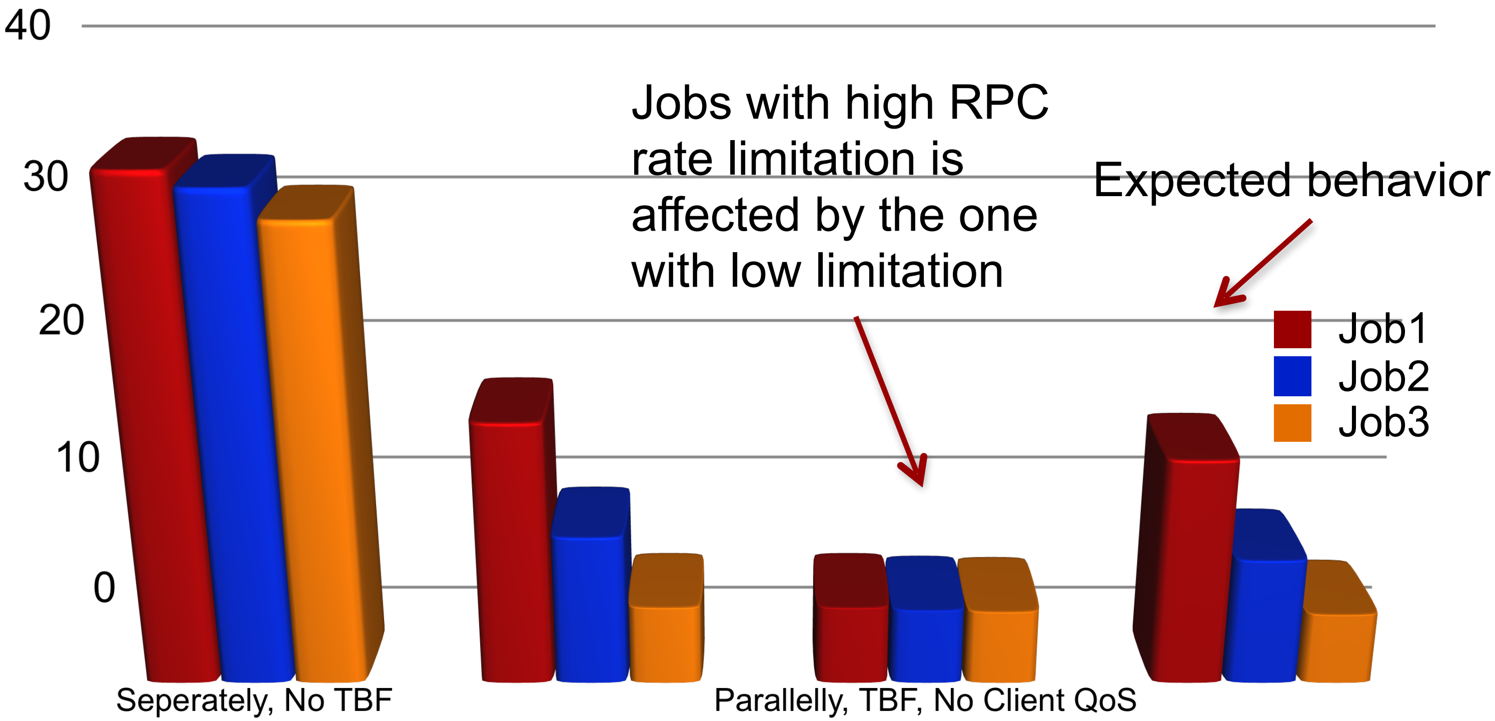
Initiatively share part of class's page cache if it has a lot of page cache and some one is starving

Used page cache of class

Idle page cache of class

Results of Client Side QoS

- **Client side QoS is able to fix the JobID TBF problem on single client**
  - TBF rules: Job1 < 20 RPC/s, Job2 < 10 RPC/s, Job3 < 5 RPC/s



Future Advanced Solutions: Cluster Wide QoS

- **Limitations of current QoS solutions:**
  - The total maximum bandwidth/IOPS that a classification can get is sum of the RPC rate limitations on all OSTs/MDTs
  - Not able to provide a centralized cluster wide mechanism to control the entire performance of a classification
- **Solution: A central QoS management system (Future)**
  - Should be able to tune the current QoS components on all OSTs/MDTs/clients in real time
  - Should be able to collect all necessary real-time information from all components to make QoS decisions
  - Should be able to smartly make QoS decisions based on both the status of the system and also the demand of administration

Conclusion

- **A series of QoS solutions are already provided**
  - Original TBF policy
  - Dependency rule of TBF
  - Rule Expression of NID + Job ID in TBF
  - Advanced expression of IDs in TBF
  - Client side performance balancing
- **Each solution itself have limitations which require enhancement**
- **But the various solutions together can enable a lot of QoS use cases**
- **A central QoS management system might be able to provide even more capability in the future**

Thank you!