Sandia National Laboratories

Exceptional service in the national interest

SANDIA LABS LUSTRE FILESYSTEM CUSTOMER IO DOS MITIGATION USING NRS TOKEN BUCKET FILTERS

Using Built-In Token Bucket Filter and Scheduling features of Lustre FS

Michael Aguilar, Jason Peters

File Storage Team, HPC Systems

Sandia National Labs, Albuquerque, NM 87114

U.S. DEPARTMENT OF ENERGY NNSA National Nuclear Security Administration

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

SAND2025-01482C

PROBLEM---CURRENT SHARED FILESYSTEM ISSUE

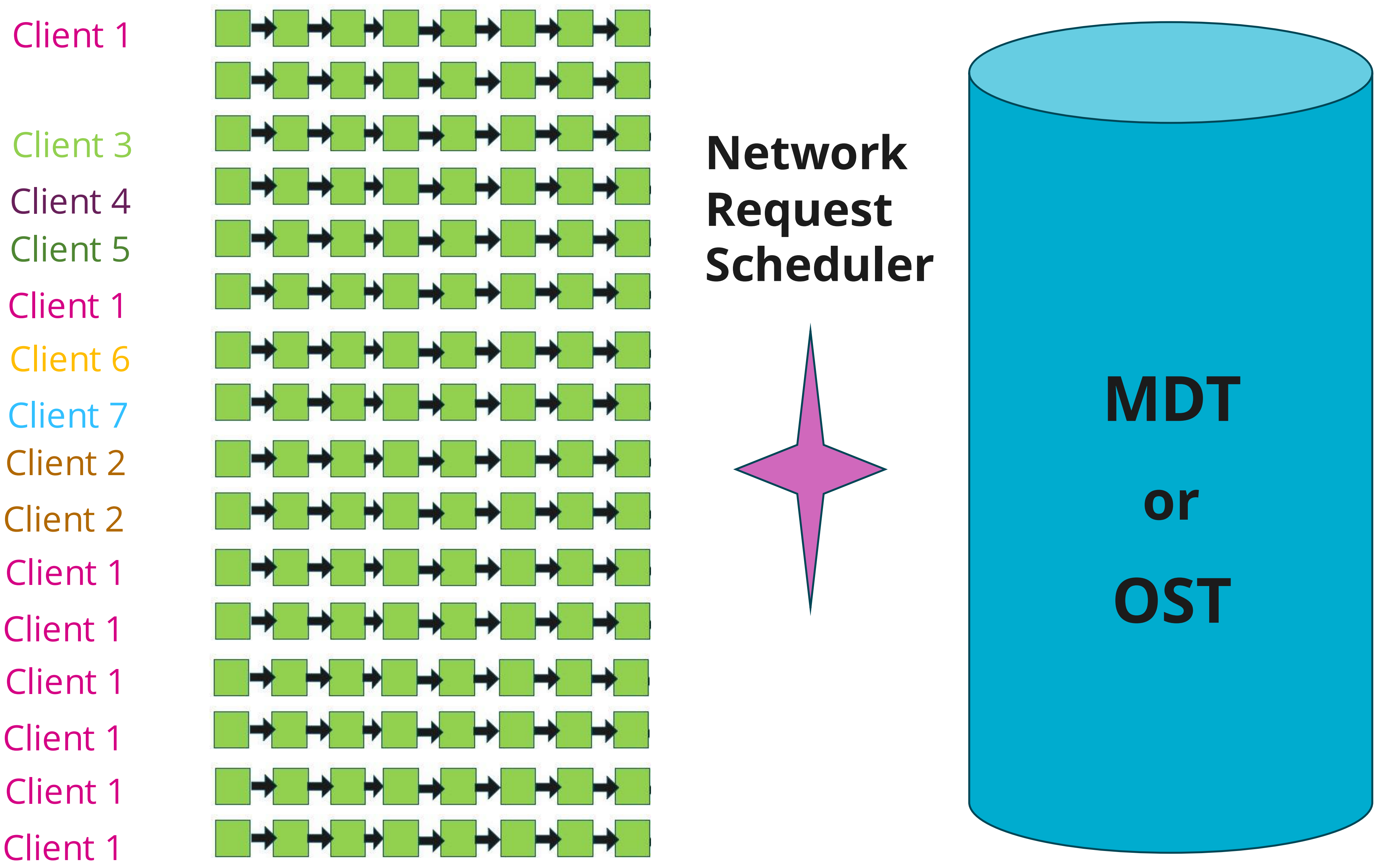
- Scratch filesystems are shared between several users and HPC machines, on Sandia's internal HPC networks
- One 'bad' batch run can create an IO load issue on the Lustre Filesystems and load down the MDTs and OSTs.
- A method for throttling communications to the filesystems, for bad applications, could be advantageous to us, as it could help to reduce the scratch filesystem outages.
  - Our goal is to reduce the impact that any single batch job can have on a shared filesystem
  - We want to be able to make it so the filesystem can respond to all of the users
  - We want to protect the filesystem from 'falling over'.

CURRENT LUSTRE FILESYSTEM INFRASTRUCTURE

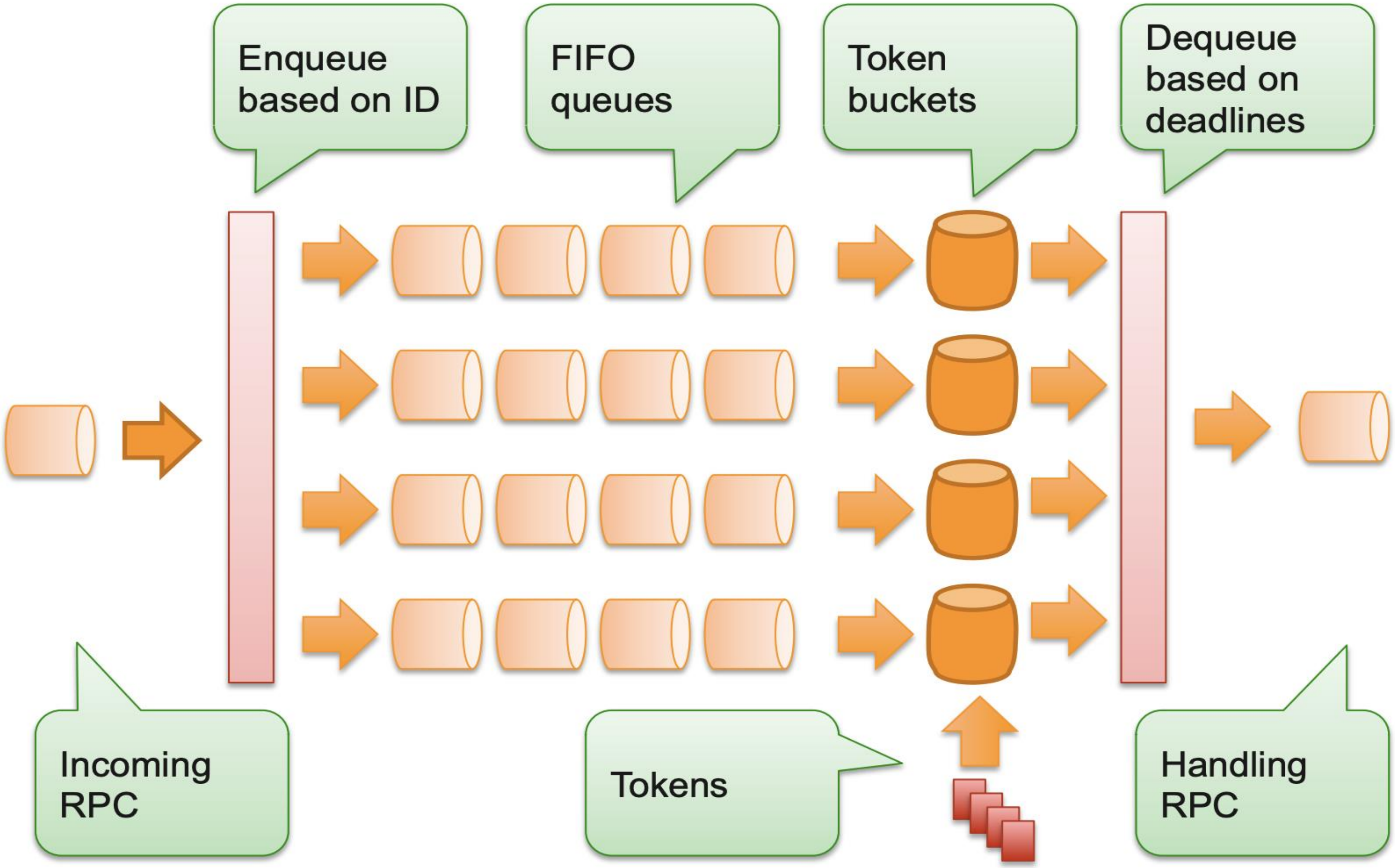
- We can leverage a couple of tools within Lustre to maintain some user control of Lustre Server access.
  - Lustre operates the transactions, with client applications, with Remote Procedure Calls (RPC)
  - In order to maintain filesystem transactional control of the RPCs, Lustre has a ‘Network Request Scheduler’ (NRS) that gets the RPC requests, in parallel, from all of the running batch jobs, on all of the HPC systems
  - The NRS is able to throttle the RPC rates before handing them over to the appropriate Lustre filesystem Metadata and Object Storage Server threads
  - A **Token Bucket Filter (TBF) policy** is the method that Lustre has implemented for throttling control
  - Lustre provides internal information, that **with new modifications**, will be able to provide us with easy UID/GID information on ’bad users’

CURRENT LUSTRE FILESYSTEM INFRASTRUCTURE

RPC IO transaction calls



CURRENT LUSTRE FILESYSTEM INFRASTRUCTURE

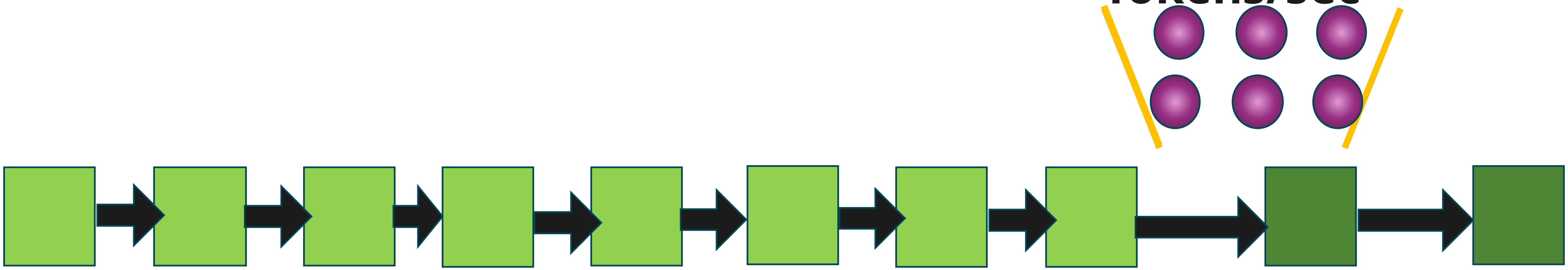


LEVERAGING TOKEN BUCKET FILTER TYPES

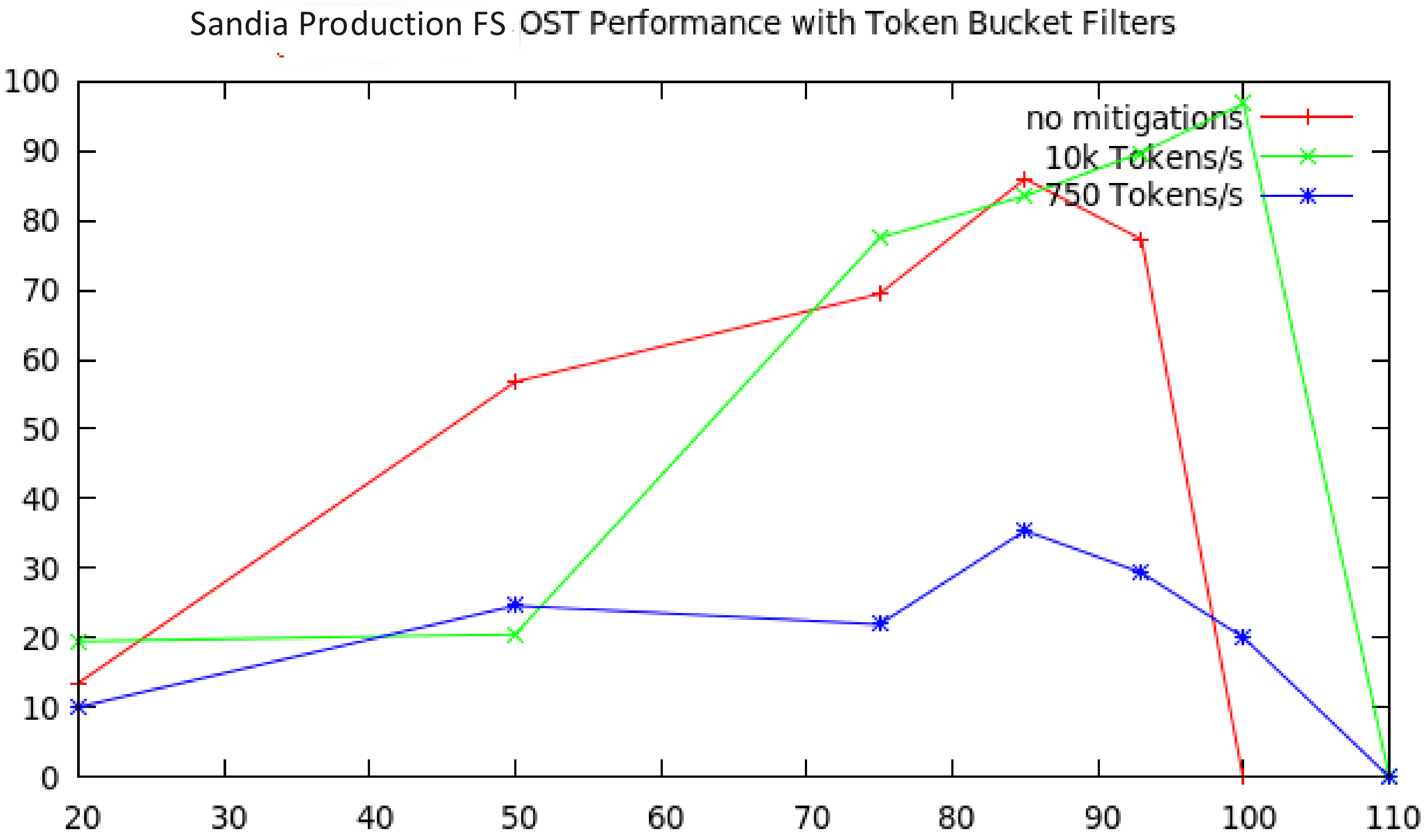
- **NID**
  - We can **throttle** user transactions by **limiting the RPC communication transaction rates available** to all Lustre network connected users, by IP Range. This sets up a **‘base rate’,** an **expectation of a normal TBF** rate for the filesystems.

base_rate=< Preset experimentally gathered value >

Tokens/sec

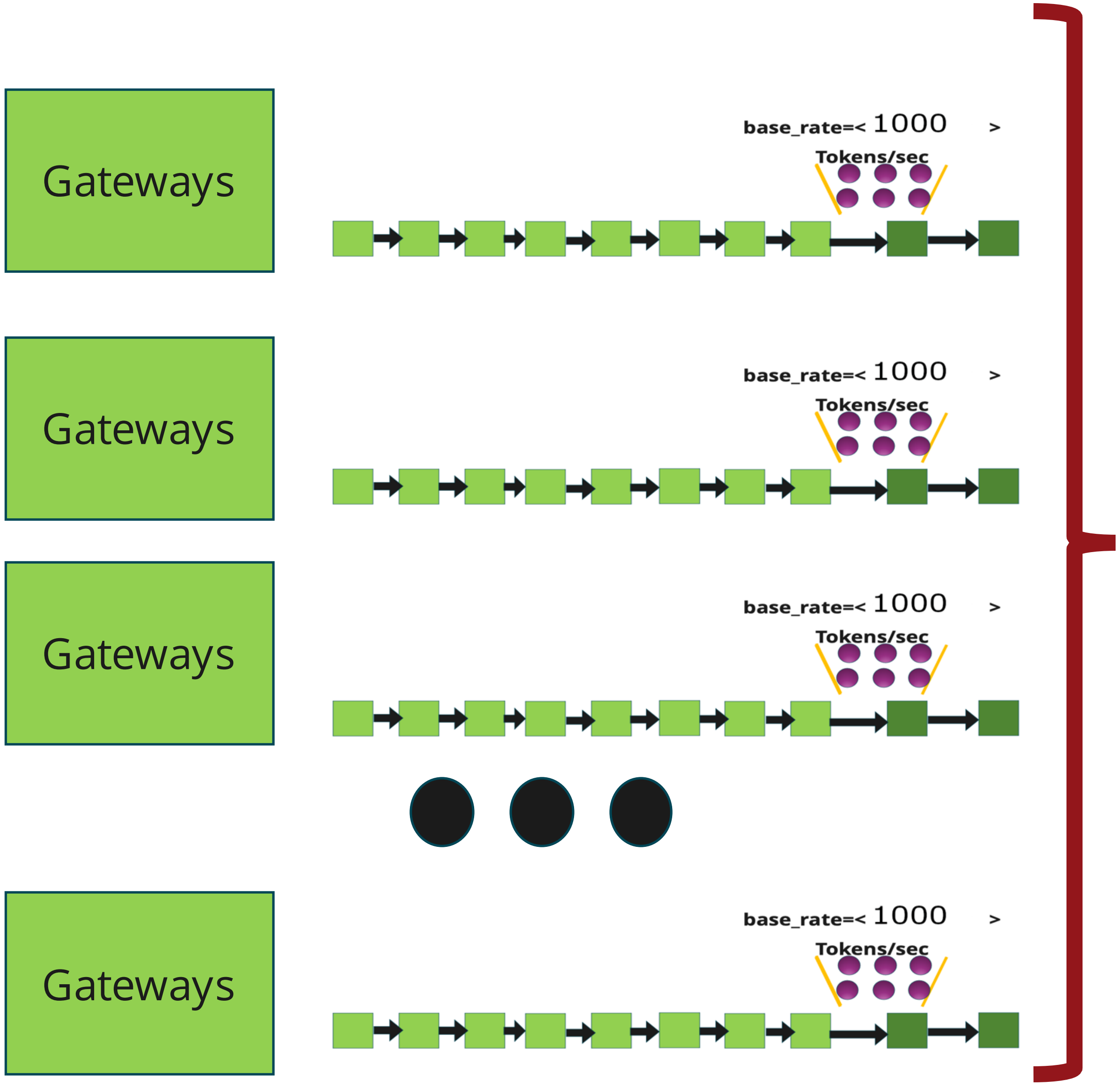


EXPERIMENTAL TOKEN BUCKET FILTER IMPLEMENTATION RESULTS

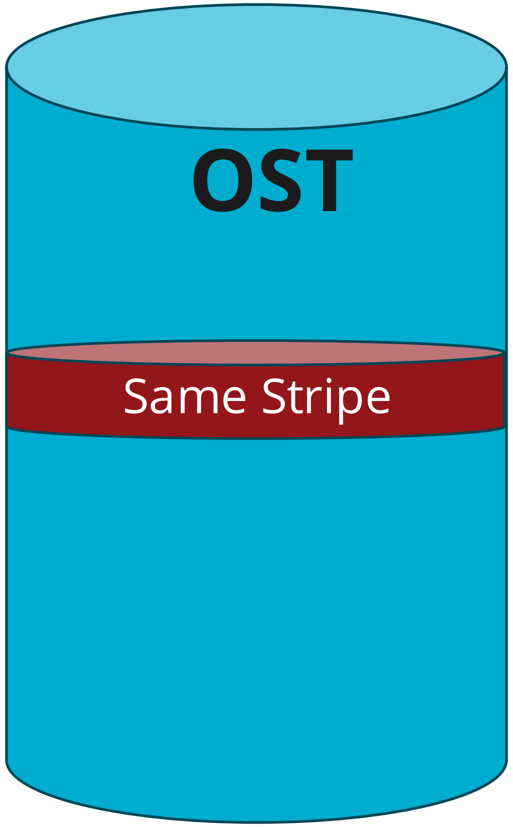


LEVERAGING TOKEN BUCKET FILTER TYPES

Limits to the Base Rate

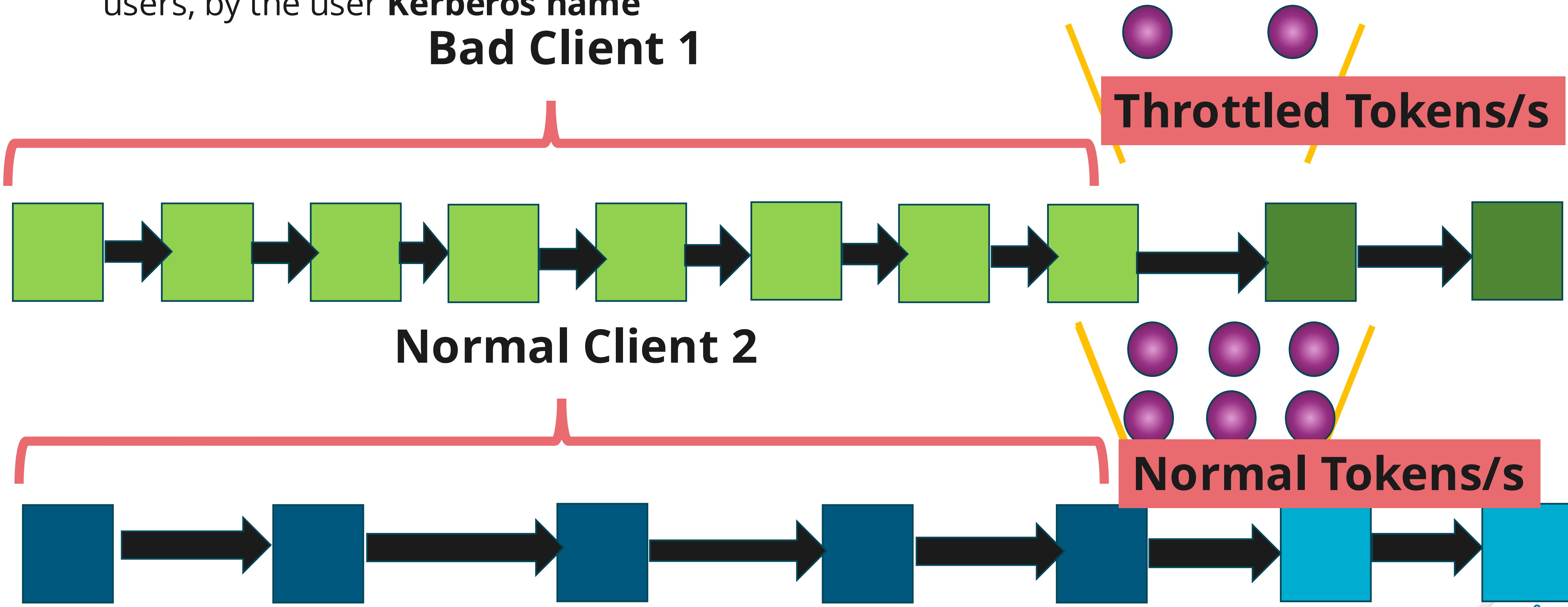


Large number of requests, from a large number of compute nodes, on a single HPC cluster, or a set of HPC Clusters **can still overload** a **single stripe with IO**



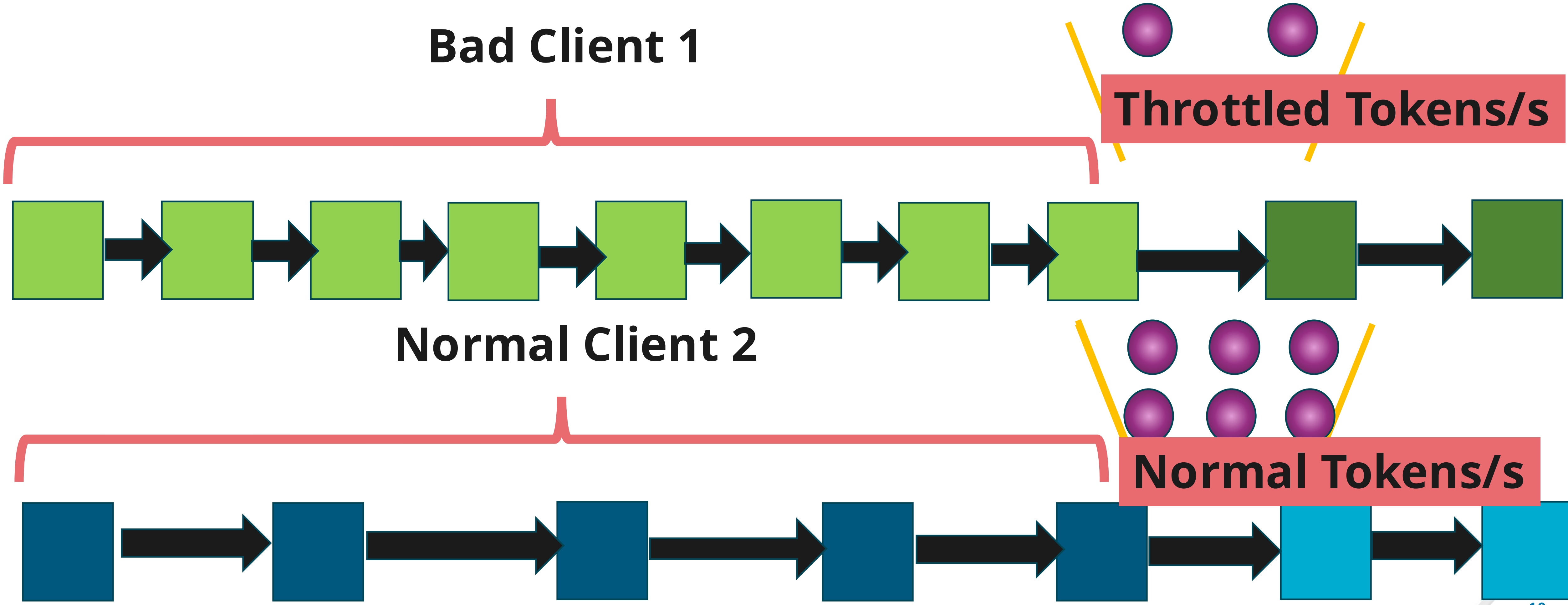
LEVERAGING TOKEN BUCKET FILTER TYPES

- **JOBID/UID**
  - We can **additionally throttle** user transactions by limiting the RPCs available to ‘bad’ users, by the user **Kerberos name**



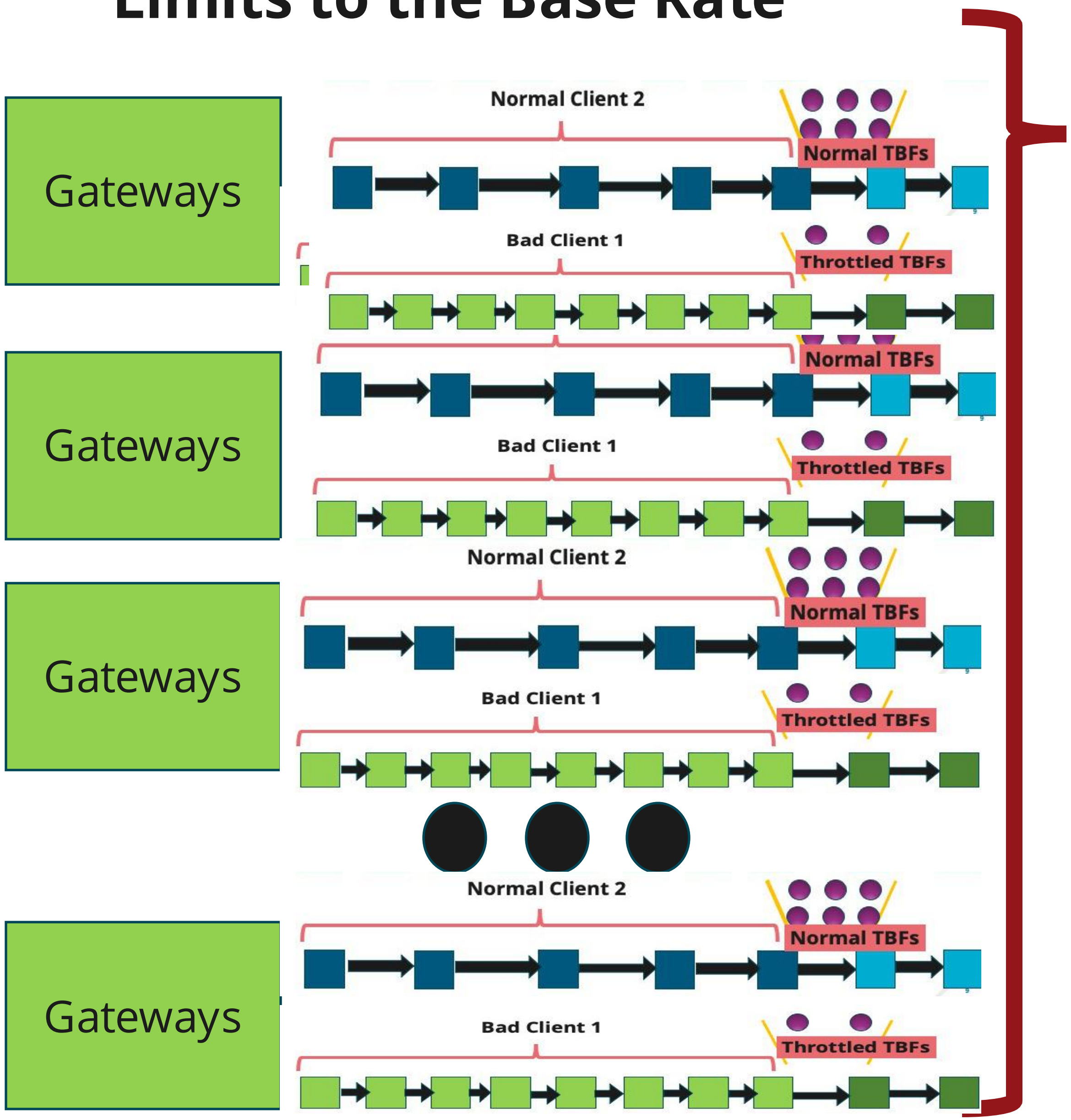
LEVERAGING TOKEN BUCKET FILTER TYPES

- **NID**
  - We can **throttle** user transactions by limiting the RPCs available to ‘bad’ users, by the user **Client Server Address**



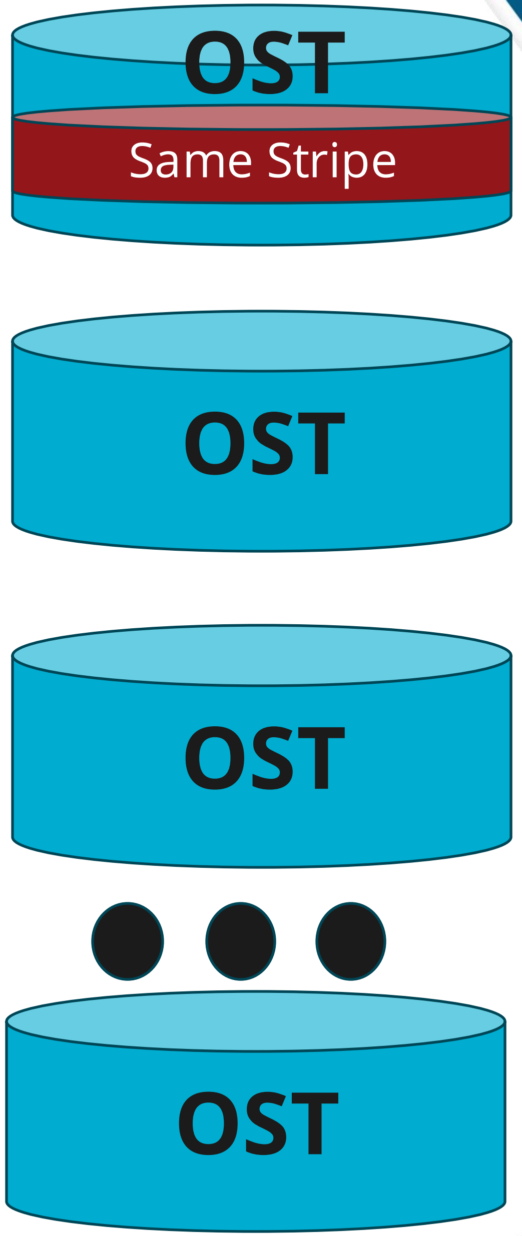
LEVERAGING TOKEN BUCKET FILTER TYPES

Limits to the Base Rate



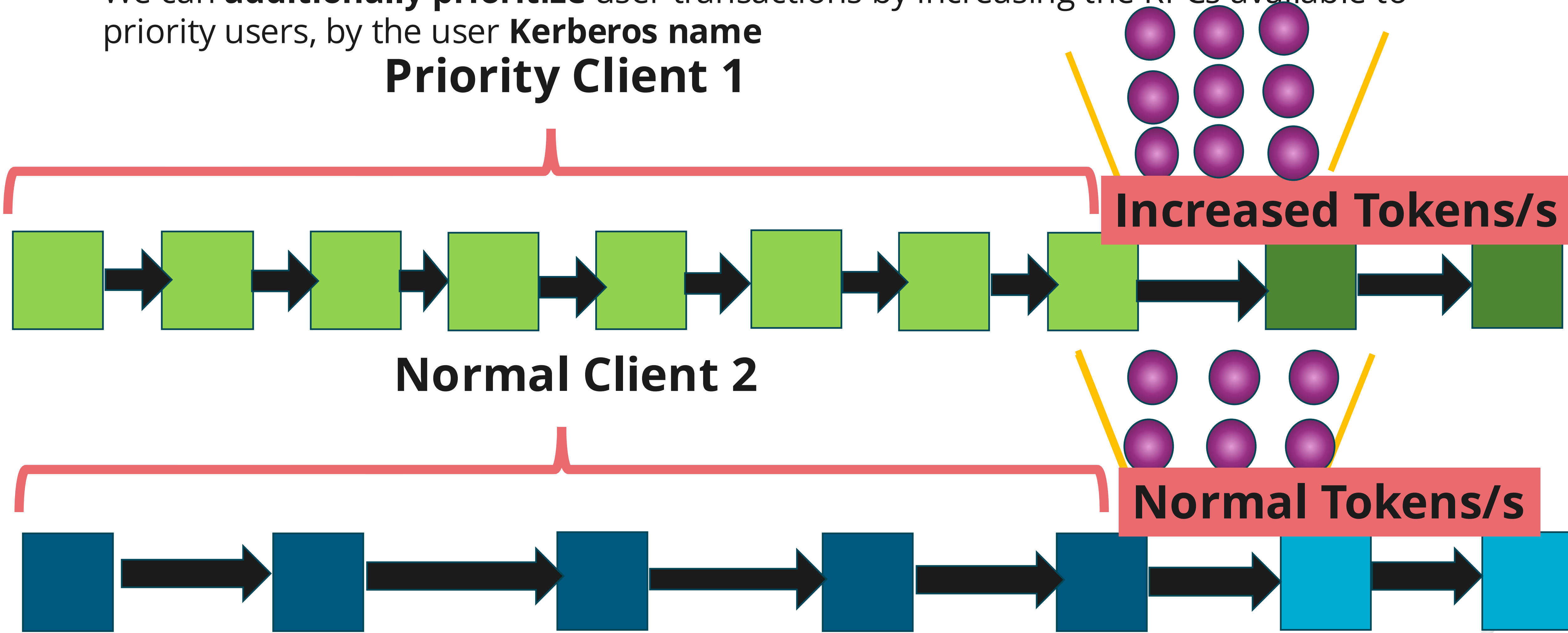
The **Bad Client has a large number of requests**, from a **large number of individual compute nodes**, on a single HPC cluster, or a set of HPC Clusters are now throttled per User. This means that if the **client has jobs running on all connected compute nodes on all of the HPC clusters,** he is still **limited** to the same **quantity of Tokens** and will still be throttled.

Normal Clients still use the **same Base Rate** of Tokens/Sec. Other users still have the usual access to the filesystem



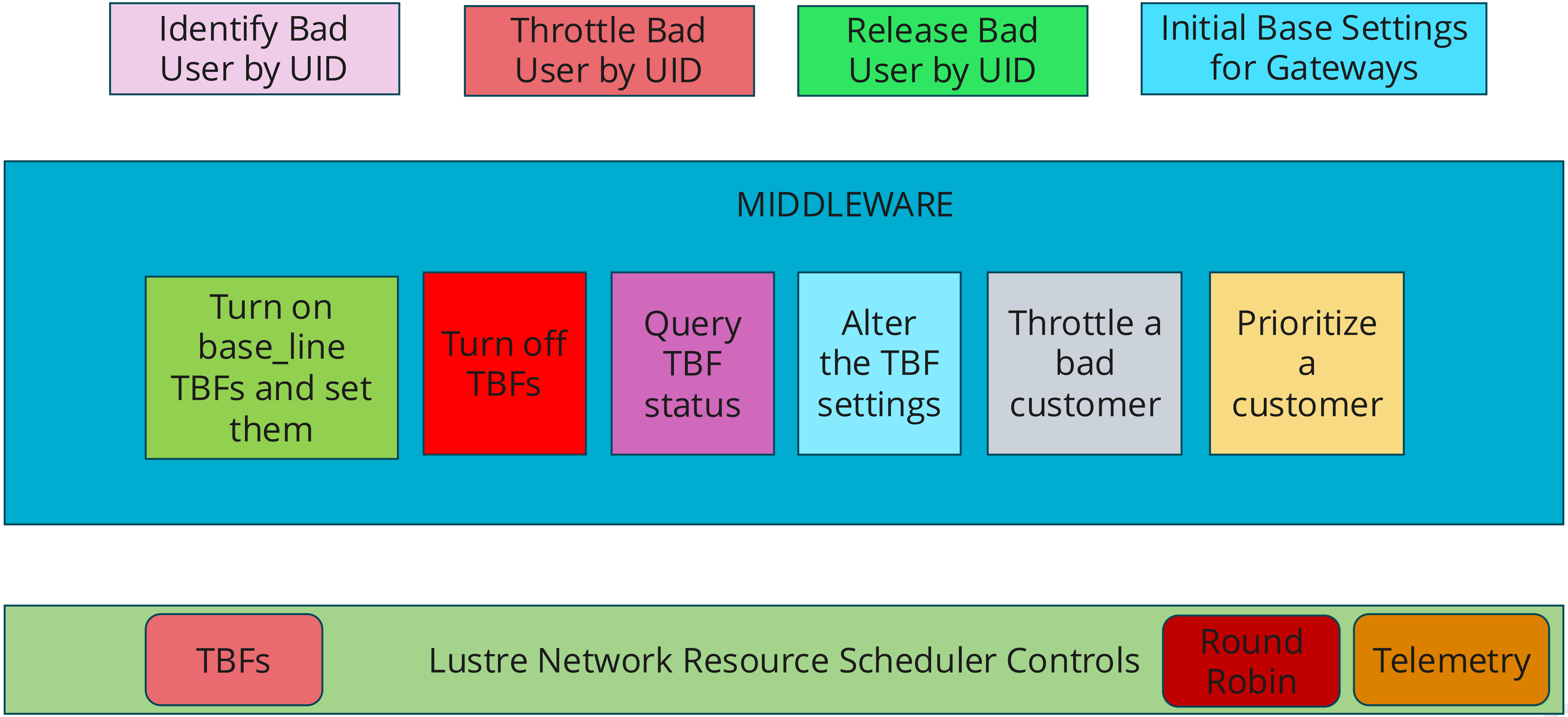
LEVERAGING TOKEN BUCKET FILTER TYPES

- **JOBID/UID**
  - We can **additionally prioritize** user transactions by increasing the RPCs available to priority users, by the user **Kerberos name**



LEVERAGING TOKEN BUCKET FILTER TYPES

Operations Front-End Controls



Identify Bad User by UID

Throttle Bad User by UID

Release Bad User by UID

Initial Base Settings for Gateways

MIDDLEWARE

Turn on base_line TBFs and set them

Turn off TBFs

Query TBF status

Alter the TBF settings

Throttle a bad customer

Prioritize a customer

TBFs

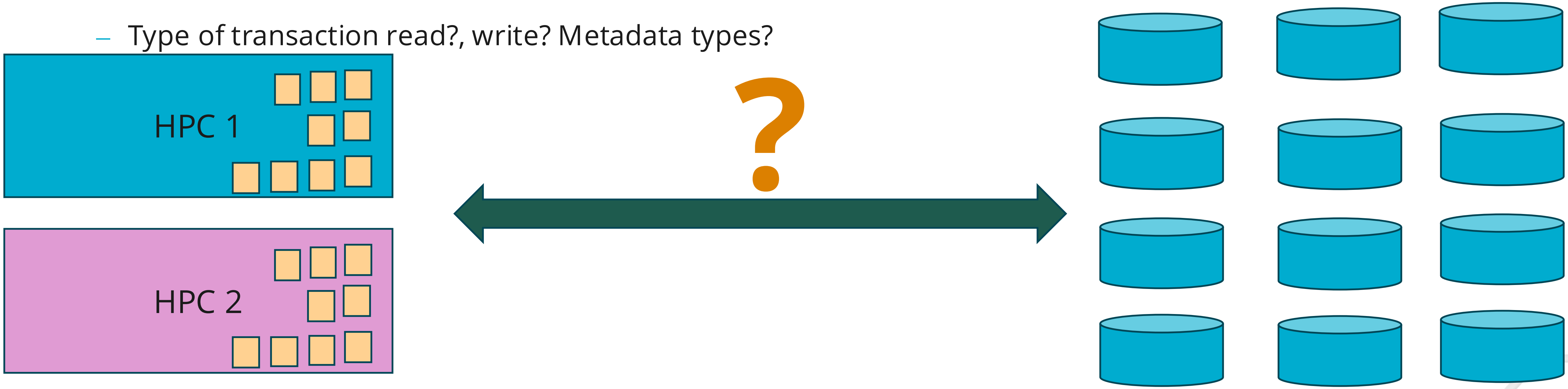
Lustre Network Resource Scheduler Controls

Round Robin

Telemetry

BAD USER IDENTIFICATION TELEMETRY FROM LUSTRE

- The Lustre filesystem is falling over.
  - How do we find the bad user that we want to throttle?
  - Lustre provides us with run-time telemetry that we can use
  - Next: **We are working on identification scripts that allow us to quickly gather the IP address(es) of the bad user(s) and the bad job IDs**
    - **IP address**
    - Running RPC transaction count so that we can sort out the bad batch job(s)
    - Type of transaction read?, write? Metadata types?

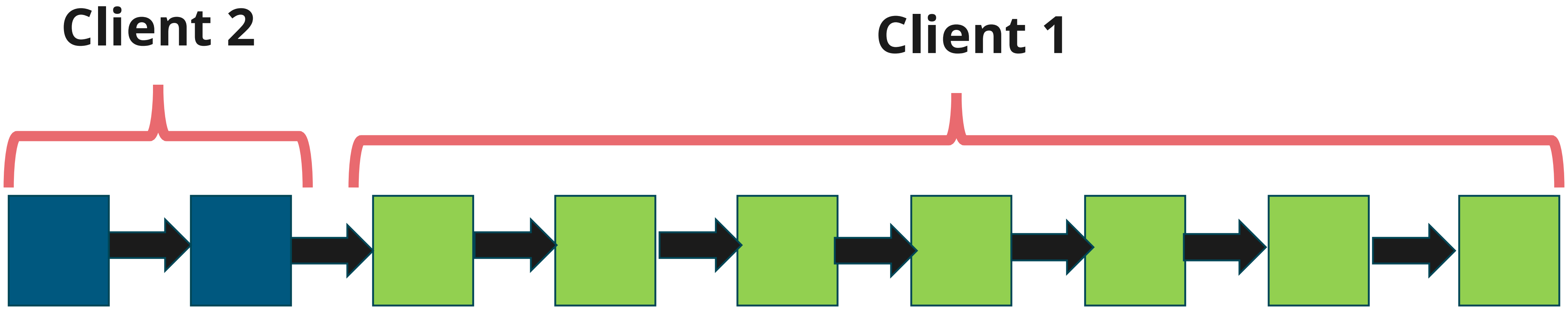


BAD USER IDENTIFICATION TELEMETRY FROM LUSTRE

- **What is missing is the UID/GID of the bad IO customer that we can quickly recover at the MDS and OSS server NRS.**
- New code is being developed for Lustre, to be placed into the Lustre code tree
  - Will provide am streamlined UID/GID association for Bad IO Customers
  - **<UID> <RPC Quantity> <NID address> <Read/Write>**
    - This information is not currently provided in Lustre
  - This information will allow us to work towards **automatically throttling and unthrottling Bad Users**

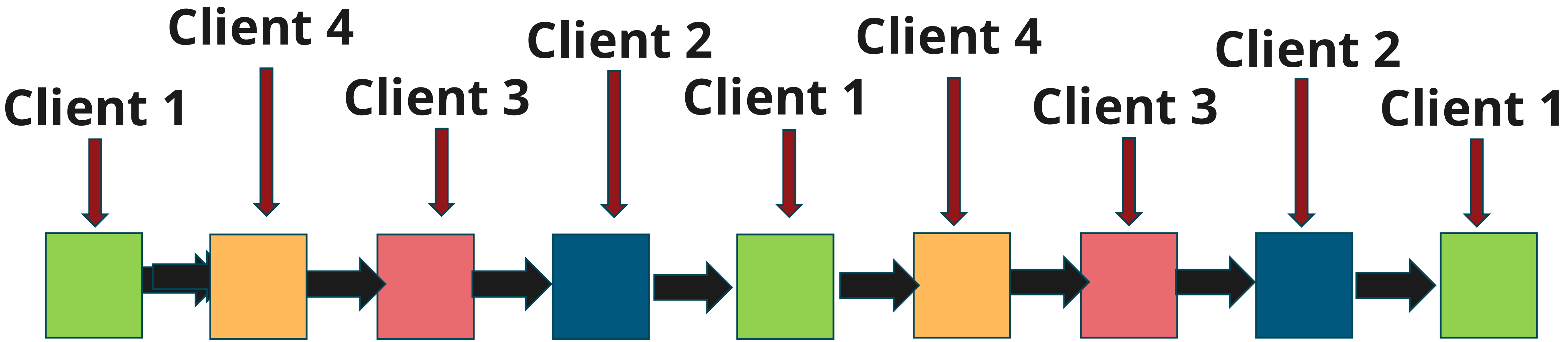
SCHEDULING RPC TRANSACTIONS

- Lustre is set up with a **FIFO RPC scheduling algorithm**. There is Round Robin capabilities that can be implemented to provide better fair-share and provide some initial mitigation to user-based DoS.
  - **Simple**, because you satisfy the Lustre transactions, one-at-a-time
  - **Not optimal** for our **shared filesystems** situation because a single user can more easily ‘bog’ down each server



SCHEDULING RPC TRANSACTIONS

- **Client Round Robin by NID (CRRN)** ★
  - Other clients will be able to get something done with their file IO when a specific user is flooding the Lustre server with IO
  - Each Lustre set of RPC transactions is executed by the servers by NID, as a RPC transaction **'Quantum'** determined by a set quantity of RPC transactions
  - After each Quantum, the next client gets access to the server.
  - **Round-Robin Client IO scheduling and Token Bucket Filters**, at the same time



SPECIAL THANKS AND QUESTIONS

- Chuck Ritter (DDN), Andreas Dilger (Whamcloud), James Simmons (ORNL)
- Questions?