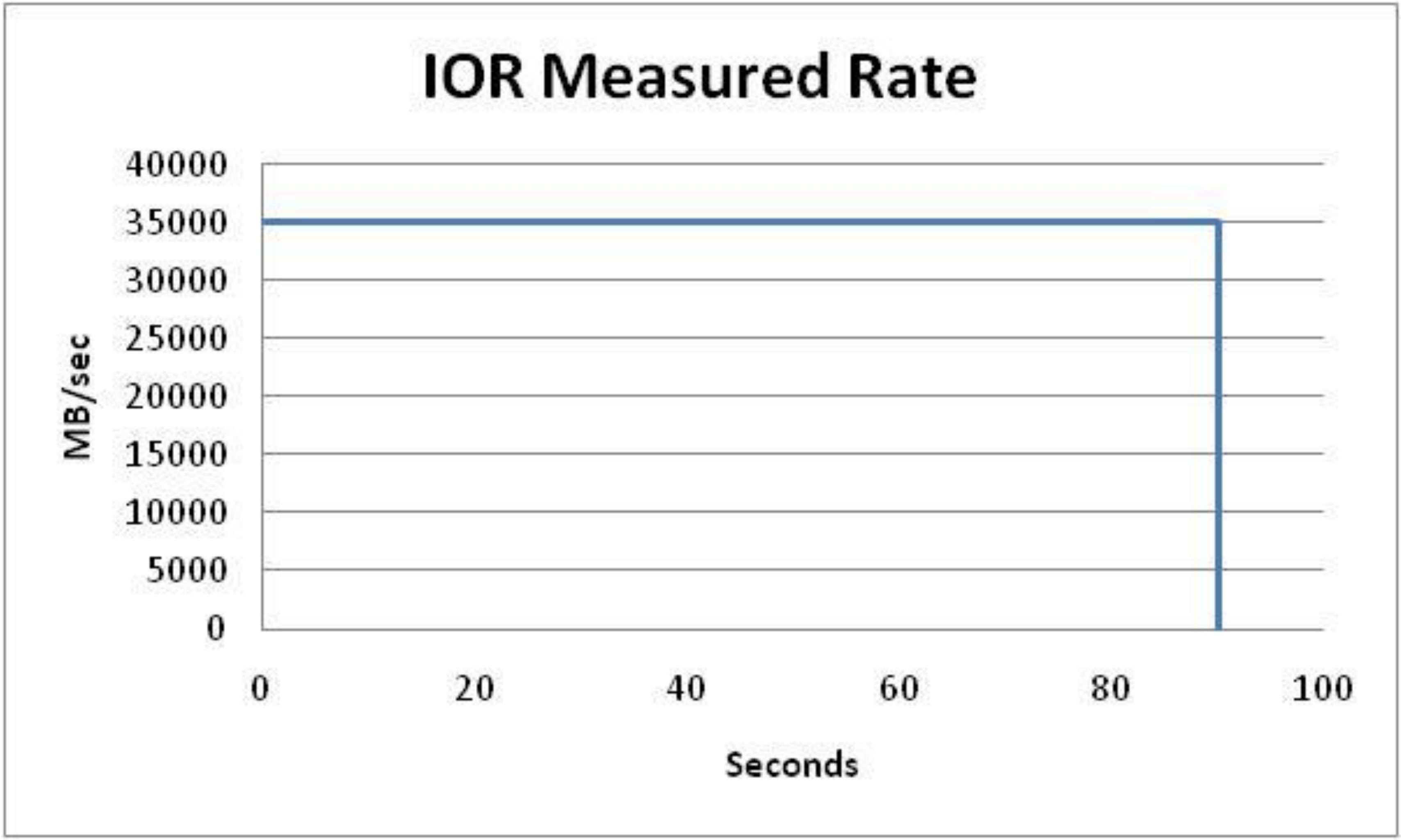
Minimizing Lustre ping effects at scale on Cray systems

Cory Spitz spitzcor@cray.com
Nic Henke nic@cray.com
Chad Zanonie chadz@cray.com
Doug Petesch dpetesch@cray.com

Overview

- IOR and ‘dead time’
- Data collection & analysis
- Root cause: obd_ping
- Problem: routing
- FGR configurations
- Tuning
- OS noise and jitter
- Conclusions & discussion

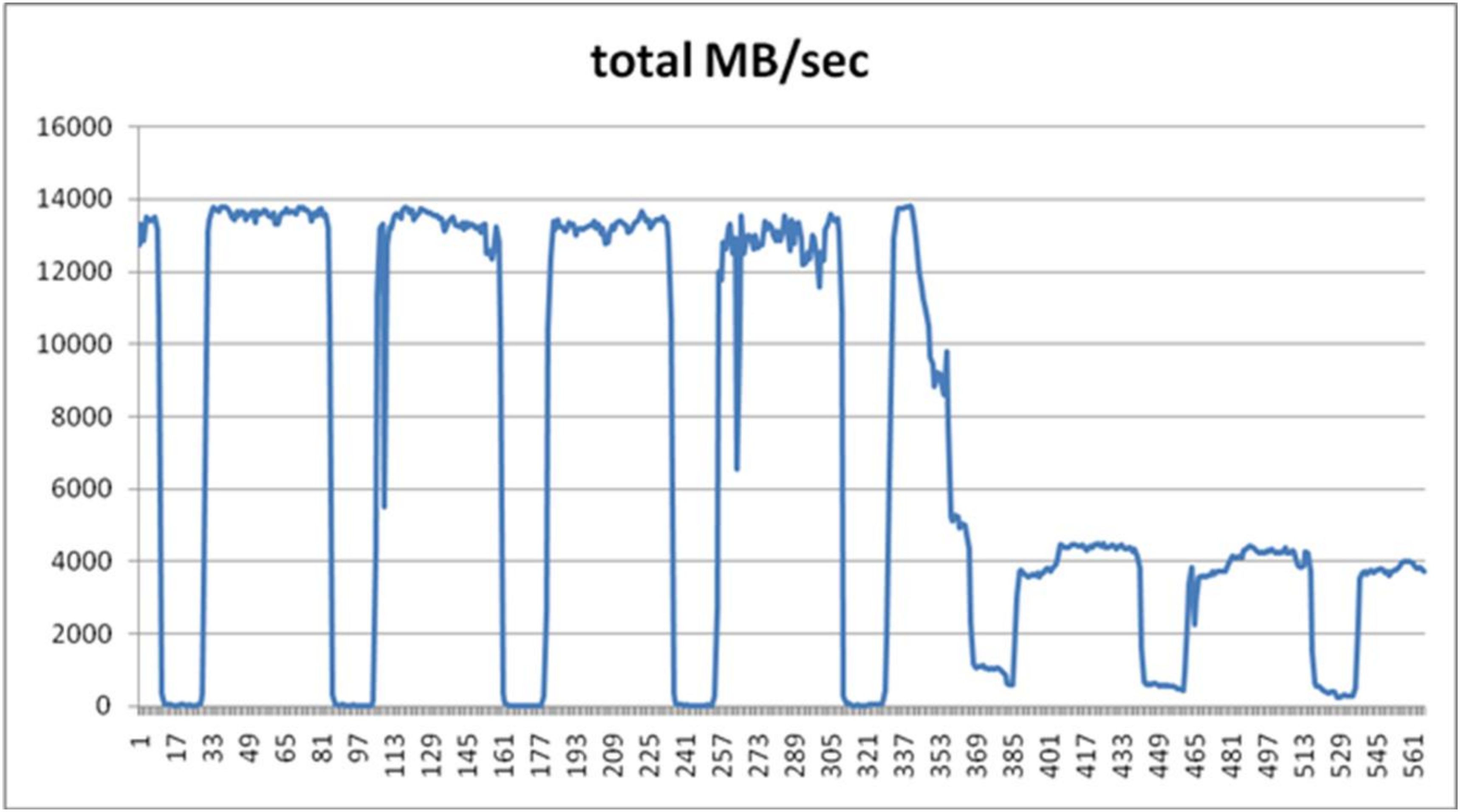
It can't be this simple, right?



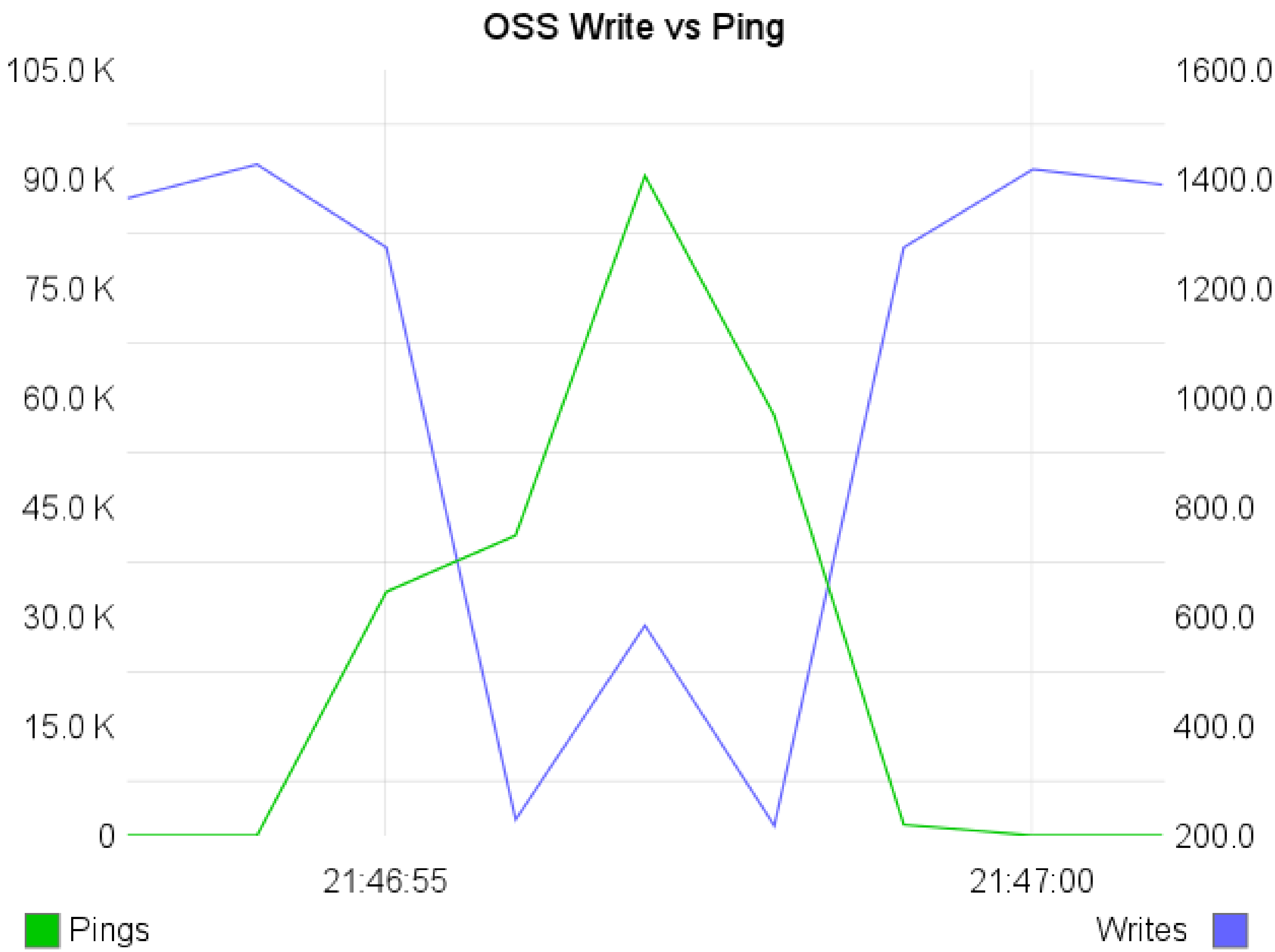
Technique: Data Collection & Visualization

- **Instrumented IOR**
  - IOR only gives us single (avg) number
    - Sustained rate is usually higher than the reported value
  - Only as fast as the slowest reader/writer (unless 'stonewall')
  - Backend monitoring tools like LMT can't associate data with jobs
  - So we've added sub-second sampling with post processing
- **Leverage Collectl**
  - Enhanced to collect LNET, LND, and OSS data
- **Ganglia/Graphite to visualize**
  - Useful for rapid debugging

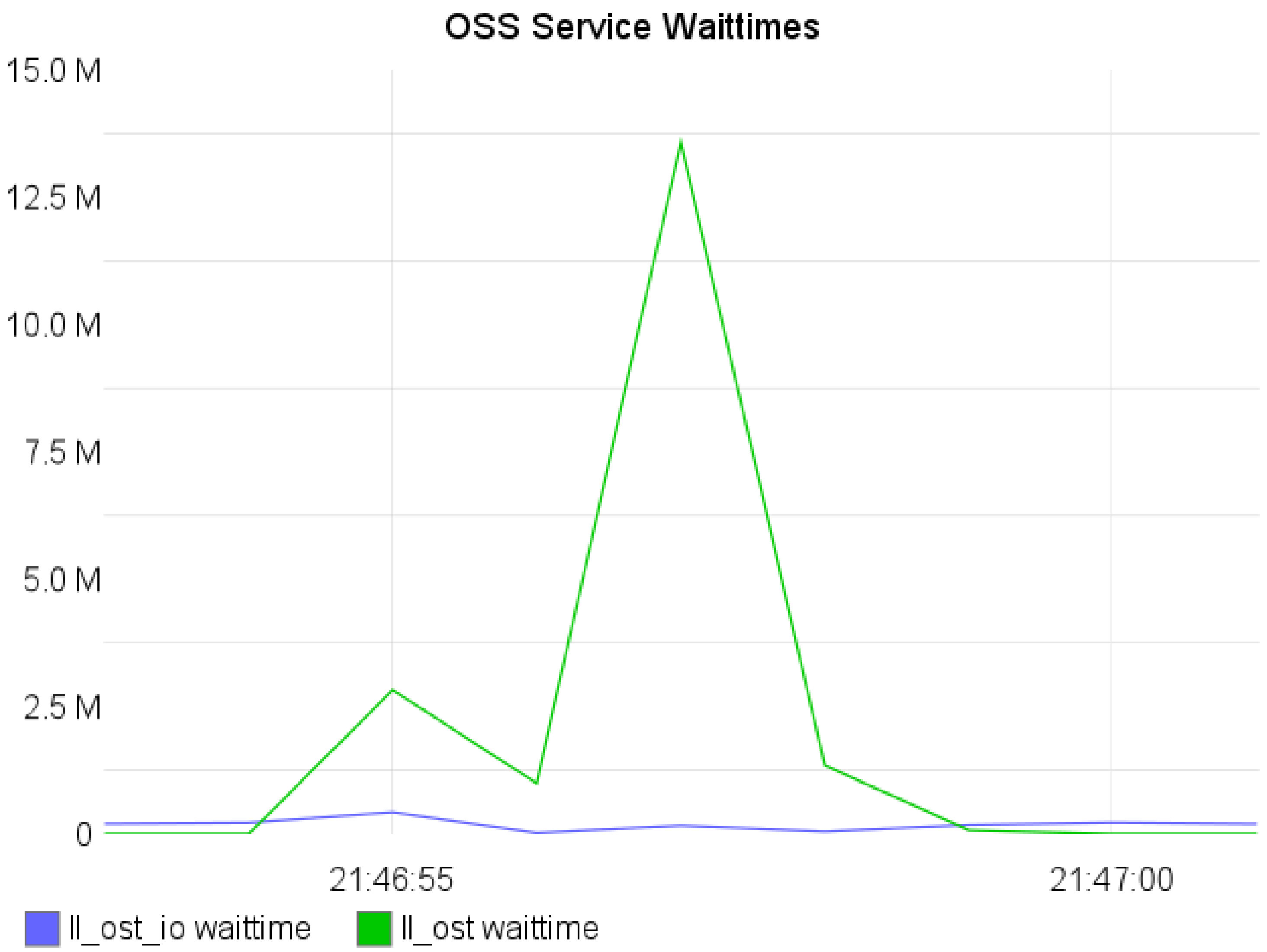
IOR and the 'dead time'



Root cause: obd_ping



I/O service time is nominal during 'deadtime'



The Lustre pinger doesn't scale

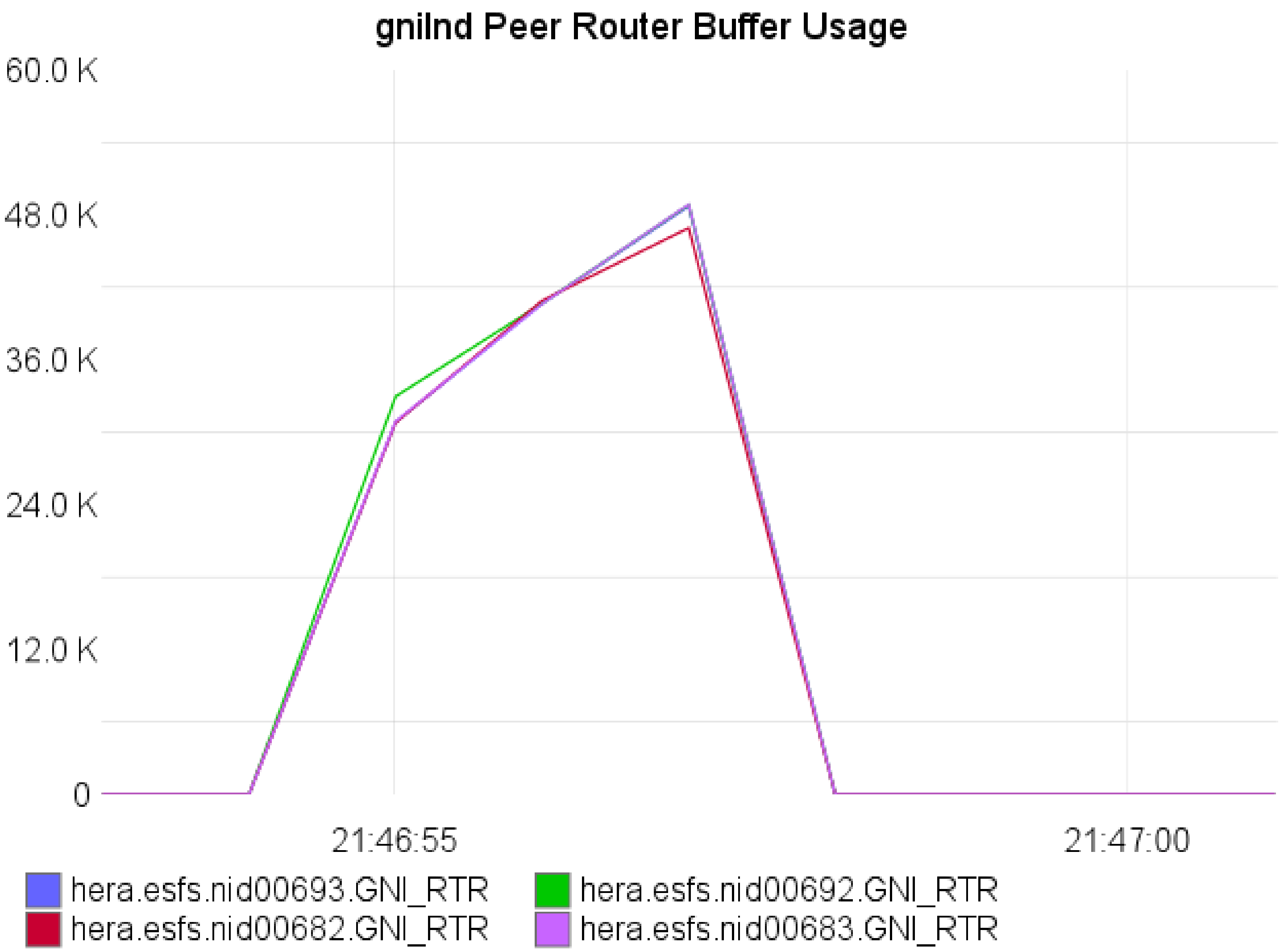
- **Every client pings every target when idle…**
  - …but clients are idle to most targets most of the time
  - unless you use every OST from every client
- **Math for low petascale (typical)**
  - 25000 clients * 4 OSTs per OSS * 360 OSS = 36M pings every 75s
  - This is not conducive to exascale
- **Usually results in multiple second 'deadtime'**
- **Around 4% to 11% reduction in throughput**
- **Instantaneous loading**
  - Synchronized to avoid jitter
- **FGR makes this worse**
  - Fewer IB destinations to send messages from each RTR

Understanding LNET message transmission in a routed environment

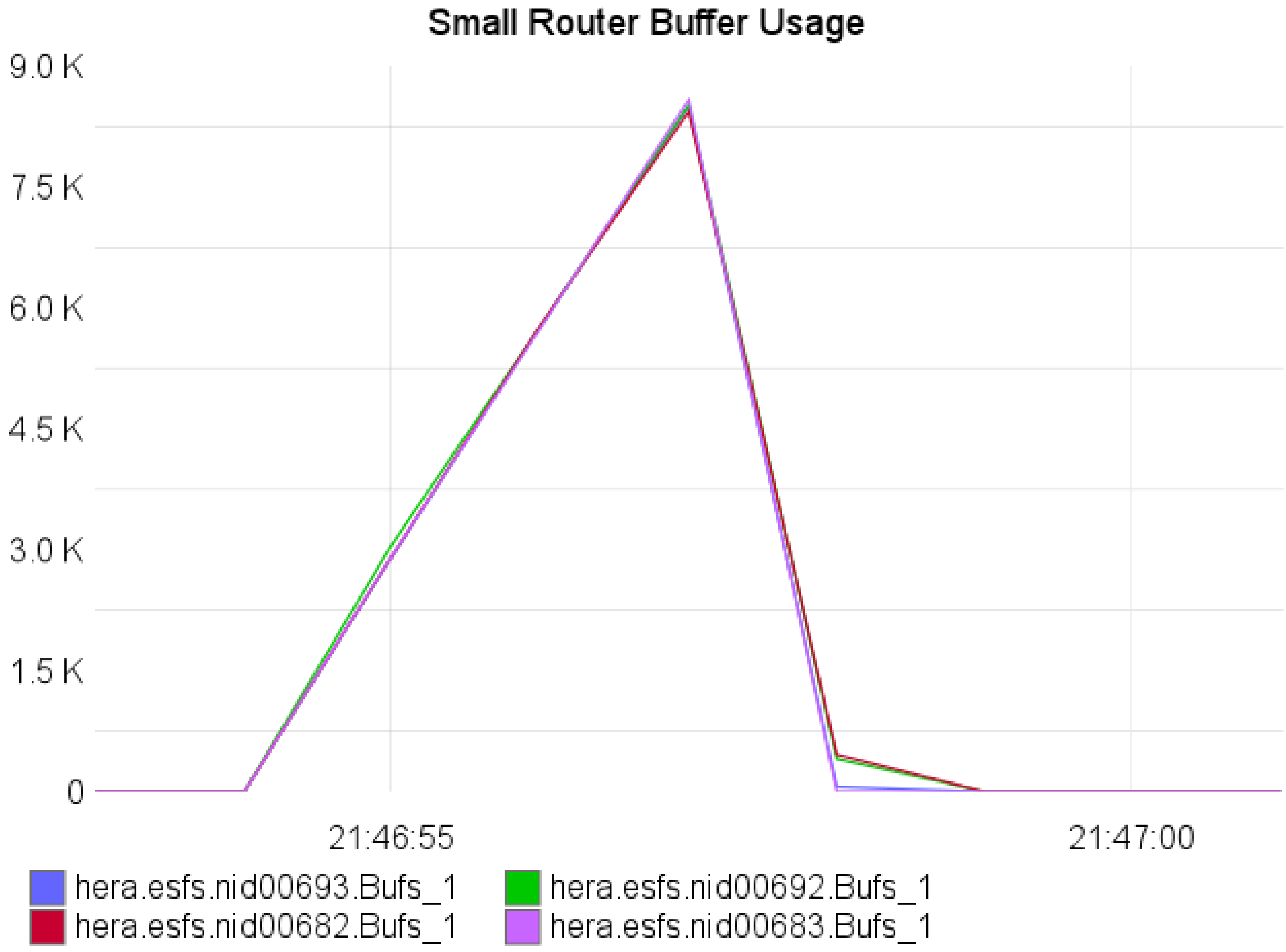
For client to server flows (like pings):

- **Client:**
  - Takes gnilnd peer and ni tx credit, sends message directly to route
  - Once HW tells us it is in remote memory, release credits
- **Router (in a nutshell):**
  - gnilnd tries to acquire peer router buffer credit and router buffer
  - If either fail, it does an ‘eager receive’ and puts into queue to finish later
  - Once it can get all credits/buffers, it copies message into router buffer
  - Then it gets o2iblnd peer tx and ni tx credit and sends over IB
  - If credit gathering fails, goes into queues until credits are available
  - When the TX credits come back, it'll release the buffer and all 4 of the credits.
- **Server:**
  - Tries to find an MD (Memory Descriptor) (with associated backing buffer)
  - Finds one and copies into it
  - Otherwise queue up and wait for buffers to be posted
  - Once message is copied into OSS buffer, o2iblnd tx credits are returned via explicit SW message

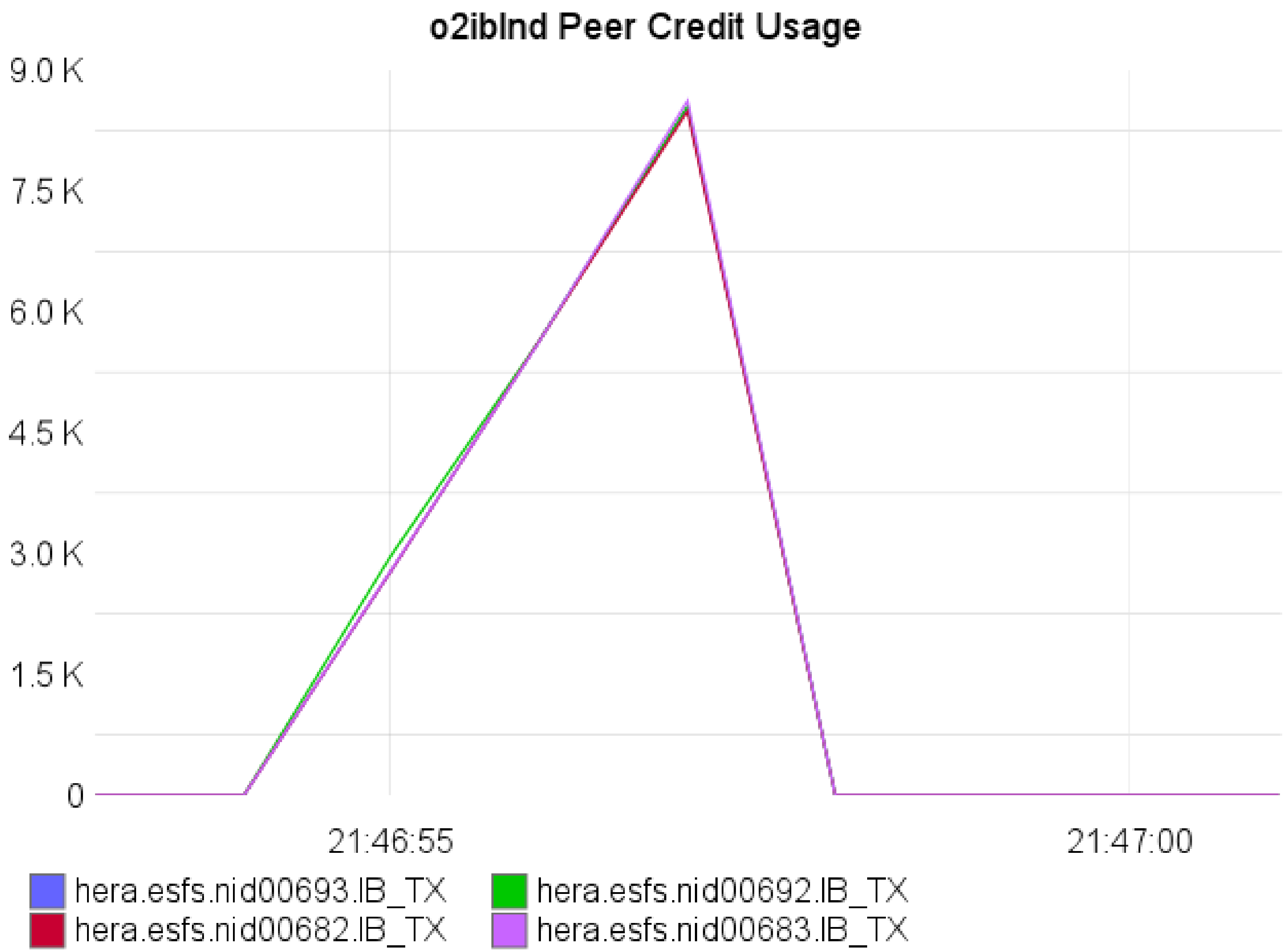
LNET queuing on routers (Gemini rx side)



LNET queuing in router buffers

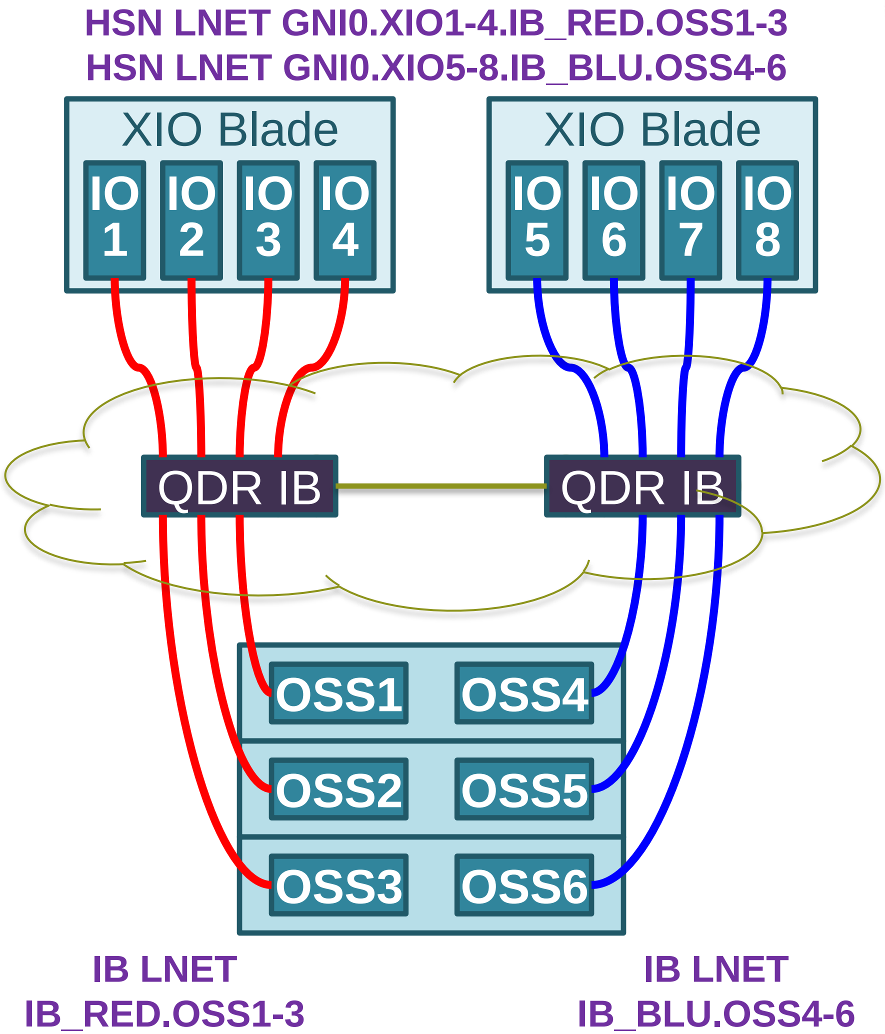


LNET queuing on routers (Infiniband side)

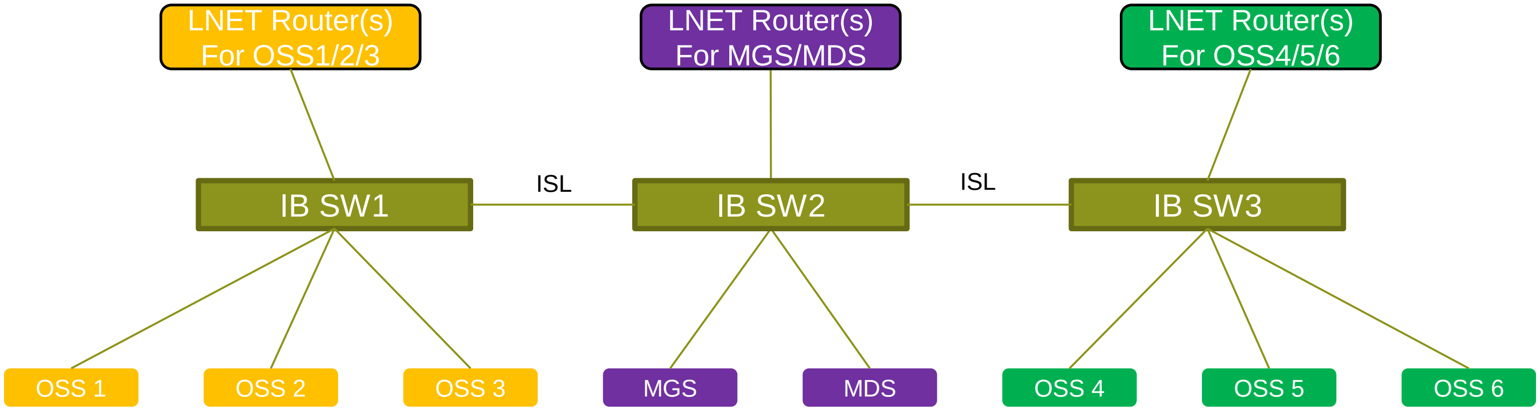


What is LNET Fine Grained Routing?

- Lustre networking can create logical subnets within a fabric
- Define multiple LNETs to isolate I/O to specific physical paths through the fabric
- LNET Fine Grained Routing groups routers and OSSes together
- Eliminates congestion on both fabrics
- Reduces cost of IB fabric
- Easy to configure with clcvt
- Easy loading with LU-1071



FGR Configurations



- **For more details see ORNL: “I/O Congestion Avoidance via Routing and Object Placement” @ CUG 2011**
- **We are using FGR groups**
  - Balance bandwidth, resiliency

Tuning Hints for o2iblnd

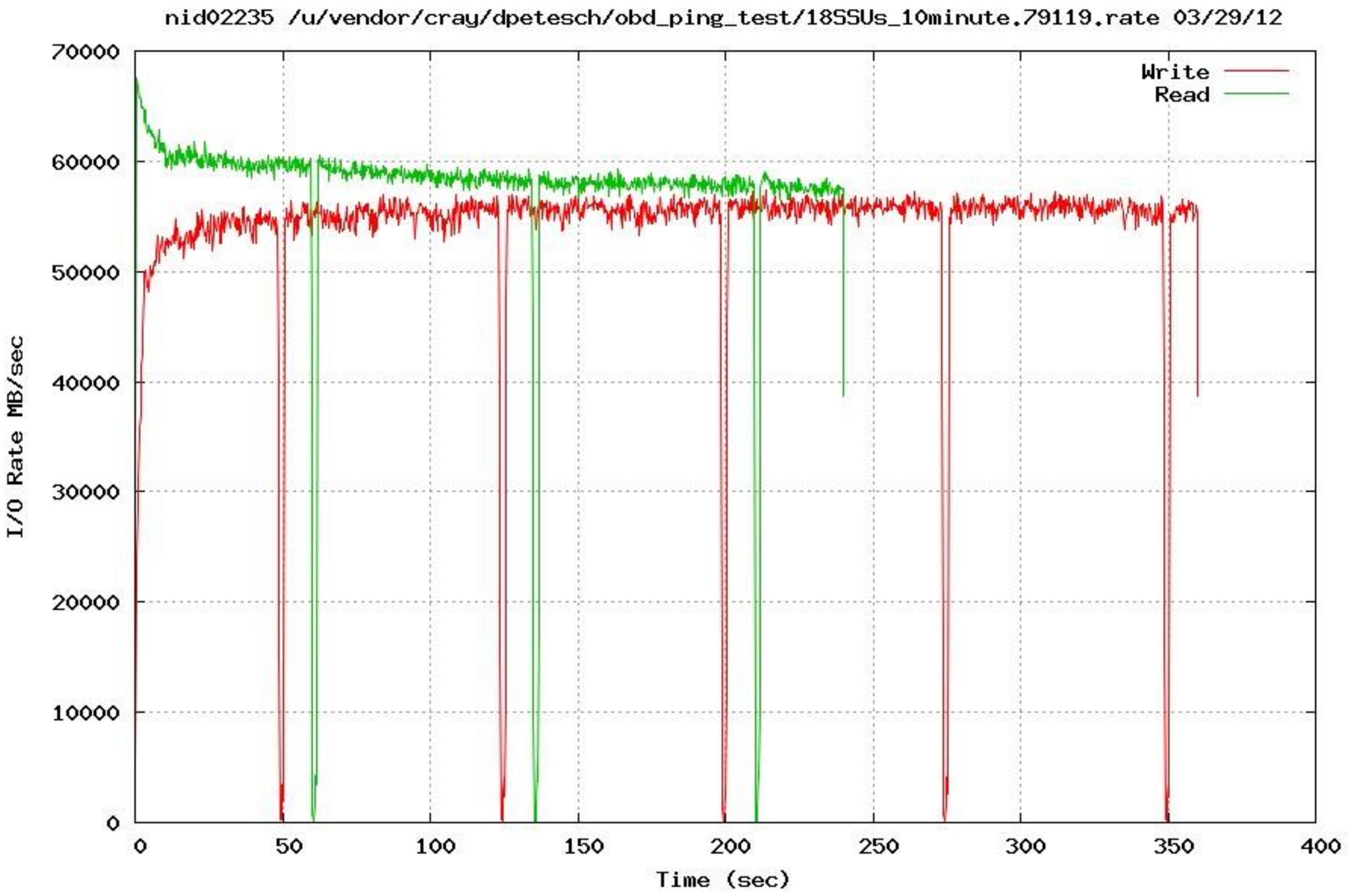
- **Tuning is much more important with FGR**
  - Small number of destinations
  - Need to tune up peer credits
- **IB LND peer_credit rules:**
  - peer_credits limited to 255 in wire structure
  - Rule: peer_credits <= 2x concurrent_sends
  - Need map_on_demand and others for concurrent_sends > 63
    - Has side effect for bulk RDMA
  - peer_credits returned explicitly in o2iblnd
  - Watch out, peer_credits must agree across the fabric
- **options ko2iblnd peer_credits=126 concurrent_sends=63**

LNET Router tuning hints

- **Pings are ~200 bytes yet use 4 KiB buffers**
- **small_router_buffers**
  - Massively undersized by default
  - Client side should dominate estimation
  - gnilnd: default 16 peer router buffer credits
    - Maximum 16 * #clients
  - Still investigating tuning for the general case
  - 16K small_router_buffers seems reasonable and only 64 MiB @ 4KiB
  - Check to see if router buffers are oversubscribed
    - cat /proc/sys/lnet/buffers
    - Is ‘min’ negative?
    - Just consume more memory on routers if you need to go higher

- **options lnet small_router_buffers=16384**
- **options ko2iblnd peer_credits=126 concurrent_sends=63**
- **options ko2iblnd peer_buffer_credits=128**

Tuning can reduce the 'deadtime'



What about OS jitter?

- Pings are a source of noise that leads to OS jitter
- O(n*m) scaling means that we have long(er) interrupts
- Pings are aligned in time to occur at the same time
  - The idea is to minimize system-wide barrier time
- Should we de-synchronize pings?
  - Still send the same amount
  - Would reduce the instantaneous load on routers and o2iblnd
- Jitter can be finicky
  - Tough to say one noise model is good or bad
  - Some apps are sensitive to intra-node jitter
  - Others are sensitive inter-node jitter

Conclusions

- Realtime diagnostic tools are good
- LNET routing not very friendly to small message sends
- o2iblnd can't saturate the wire with small messages
- LNET & LND not easy to understand and tune
- LNET FGR complicates matters
- Should remove jitter & 'deadtime' by removing the pinger

Questions?

Thank You