ESnet
ENERGY SCIENCES NETWORK

Recent Linux TCP Updates, and how to tune your 100G host

Nate Hanford, Brian Tierney, ESnet
bltierney@es.net
http://fasterdata.es.net

Internet2 Technology Exchange,

Sept 27, 2016, Miami, FL

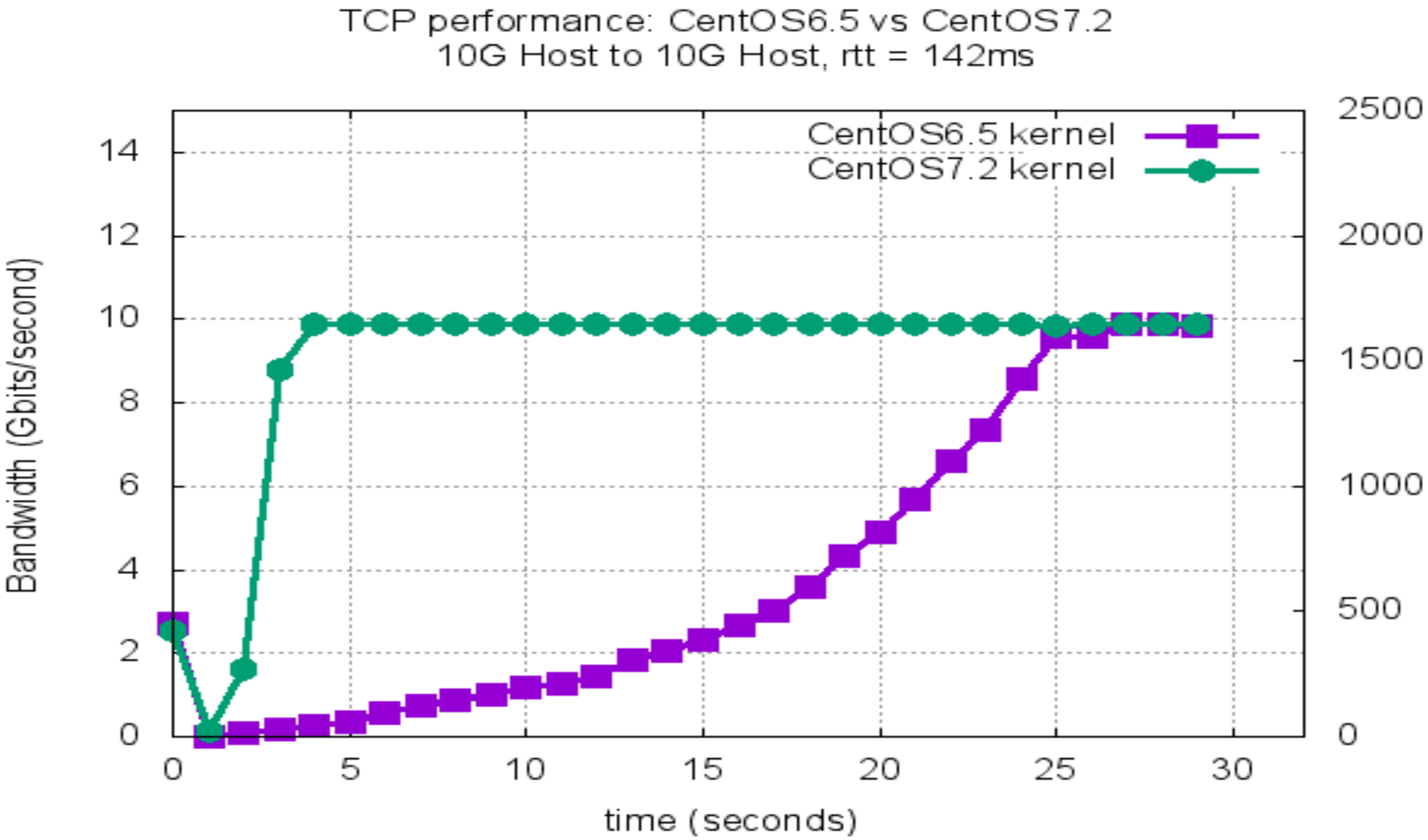
U.S. DEPARTMENT OF ENERGY
Office of Science

BERKELEY LAB

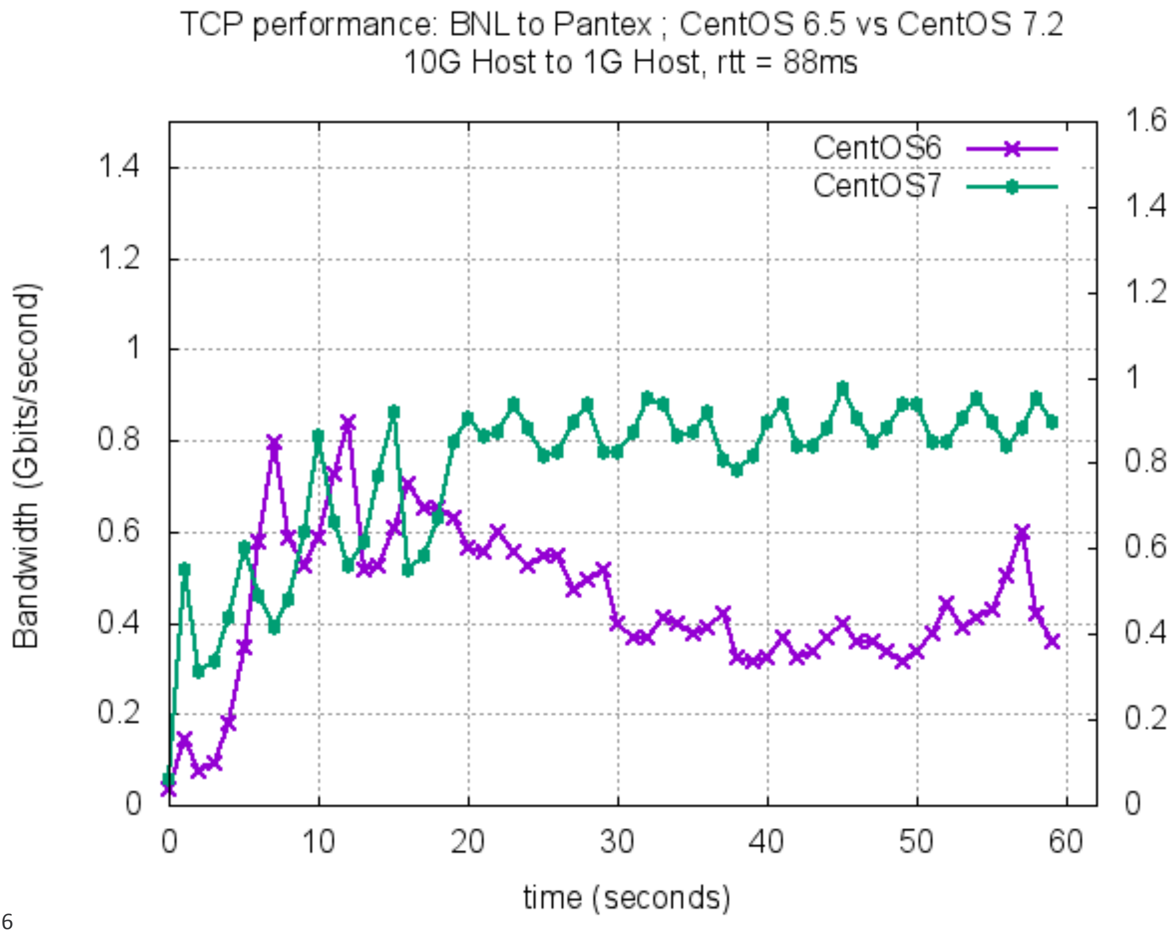
Observation #1

- TCP is more stable in CentOS7 vs CentOS6
  - Throughput ramps up much quicker
    - More aggressive slow start
  - Less variability over life of the flow

Berkeley to Amsterdam



New York to Texas



Observation #2

- Turning on FQ helps throughput even more
  - TCP is even more stable
  - Works better with small buffer devices

- Pacing to match bottleneck link works better yet

TCP option: Fair Queuing Scheduler (FQ)

Available in Linux kernel 3.11 (released late 2013) or higher

- available in Fedora 20, Debian 8, and Ubuntu 13.10
- Backported to 3.10.0-327 kernel in v7.2 CentOS/RHEL (Dec 2015)

To enable Fair Queuing (which is off by default), do:

- tc qdisc add dev $ETH root fq

  Or add this to /etc/sysctl.conf:

  net.core.default_qdisc = fq

To both pace and shape the traffic:

- tc qdisc add dev $ETH root fq maxrate Ngbit
  - Can reliably pace up to a maxrate of 32Gbps on a fast processors

Can also do application pacing using a ‘setsockopt(SO_MAX_PACING_RATE)’ system call

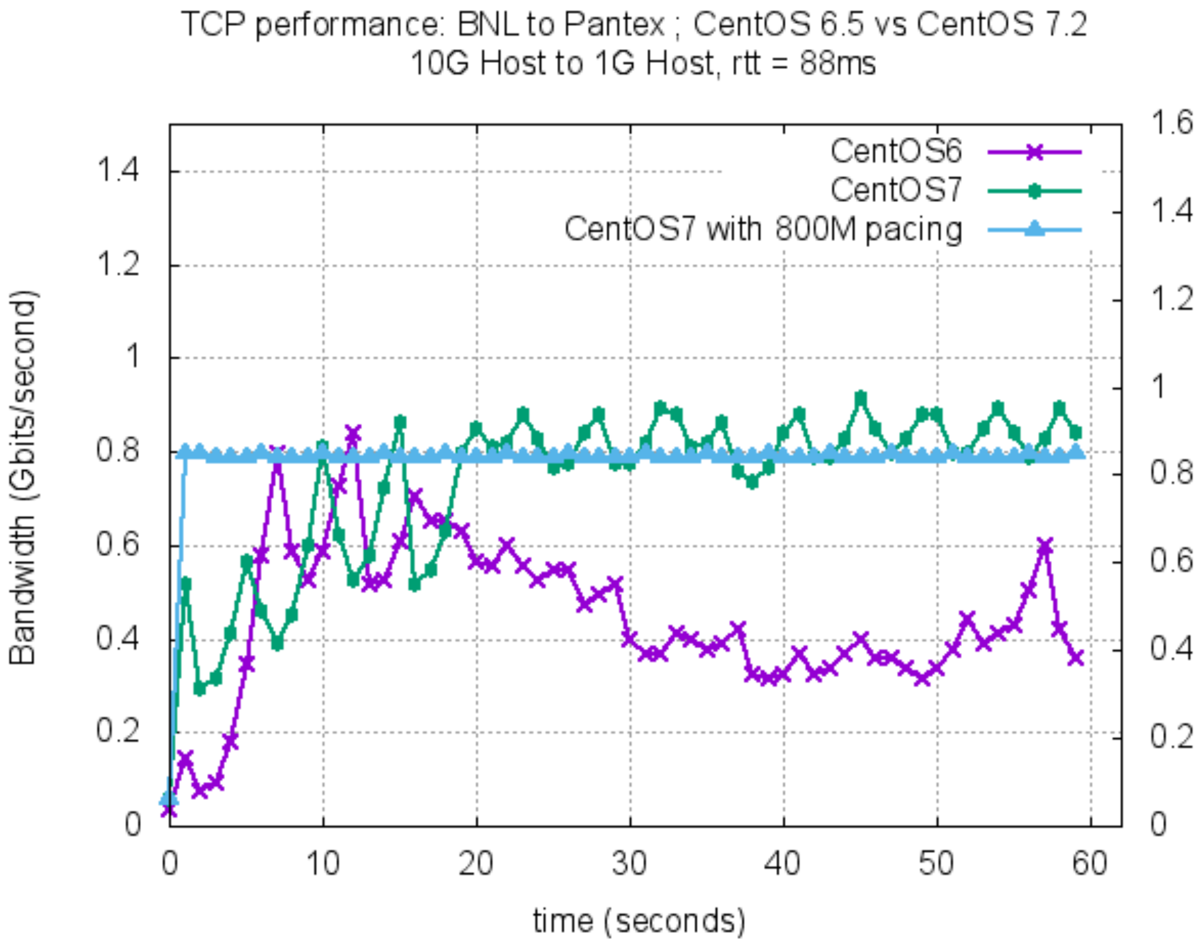
- iperf3 supports this via the “—bandwidth’ flag

FQ Background

- Lots of discussion around 'buffer bloat' starting in 2011
  - https://www.bufferbloat.net/
- Google wanted to be able to get higher utilization on their network
  - Paper: "B4: Experience with a Globally-Deployed Software Defined WAN, SIGCOMM 2013
- Google hired some very smart TCP people
    - Van Jacobson, Matt Mathis, Eric Dumazet, and others

- Result: Lots of improvements to the TCP stack in 2013-14, including most notably the 'fair queuing' pacer

New York to Texas: With Pacing

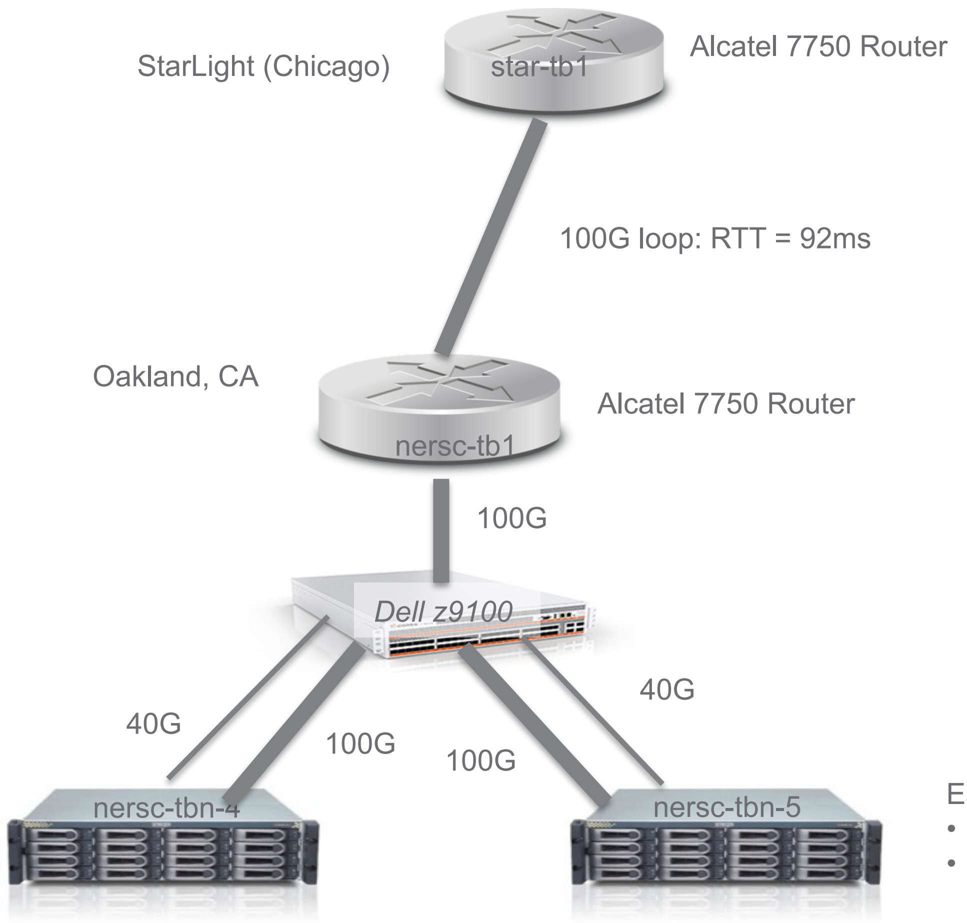


100G Host Tuning

Test Environment

- Hosts:
  - Supermicro X10DRi DTNs
  - Intel Xeon E5-2643v3, 2 sockets, 6 cores each
  - CentOS 7.2 running Kernel 3.10.0-327.el7.x86_64
  - Mellanox ConnectX-4 EN/VPI 100G NICs with ports in EN mode
  - Mellanox **OFED Driver 3.3-1.0.4 (03 Jul 2016), Firmware 12.16.1020**
- Topology
  - Both systems connected to Dell Z9100 100Gbps ON Top-of-Rack Switch
  - Uplink to nersc-tb1 ALU SR7750 Router running 100G loop to Starlight and back
    - 92ms RTT
  - Using Tagged 802.1q to switch between Loop and Local VLANs
  - LAN had 54usec RTT
- Configuration:
  - MTU was 9000B
  - **irqbalance, tuned, and numad were off**
  - core affinity was set to cores 7 and 8 (on the NUMA node closest to the NIC)
  - All tests are IPV4 unless otherwise stated

Testbed Topology



Each host has:

- Mellanox ConnectX-4 (100G)
- Mellanox ConnectX-3 (40G)

ESnet

Our Current Best Single Flow Results

- TCP
  - LAN: 79Gbps
  - WAN (RTT = 92ms): 36.5 Gbps, 49 Gbps using ‘sendfile’ API (‘zero-copy’)
  - Test commands:
    - LAN: nuttcp -i1 -xc 7/7 –w1m -T30 hostname
    - WAN: nuttcp -i1 -xc 7/7 –w900M -T30 hostname
- UDP:
  - LAN and WAN: 33 Gbps
  - Test command: nuttcp -l8972 -T30 -u -w4m -Ru -i1 -xc7/7 hostname

Others have reported up to 85 Gbps LAN performance with similar hardware

CPU governor

Linux CPU governor (P-States) setting makes a **big** difference:

```
RHEL: cpupower frequency-set -g performance
Debian: cpufreq-set -r -g performance
```

57Gbps default settings (powersave) vs. **79Gbps** 'performance' mode on the LAN

To watch the CPU governor in action:

```
watch -n 1 grep MHz /proc/cpuinfo
  cpu MHz       : 1281.109
  cpu MHz       : 1199.960
  cpu MHz       : 1299.968
  cpu MHz       : 1199.960
  cpu MHz       : 1291.601
  cpu MHz       : 3700.000
  cpu MHz       : 2295.796
  cpu MHz       : 1381.250
  cpu MHz       : 1778.492
```

CPU frequency

- Driver: Kernel module or code that makes CPU frequency calls to hardware
- Governor: Driver setting that determines how the frequency will be set
- Performance Governor: Bias towards higher frequencies
- Userspace Governor: Allow user to specify exact core and package frequencies
- Only the Intel P-States Driver can make use of Turbo Boost
- Check current settings: `cpupower frequency-info`

	P-States Performance	ACPI-CPUfreq Performance	ACPI-CPUfreq Userspace
LAN	79G	72G	67G
WAN	36G	36G	27G

TCP Buffers

```
# add to /etc/sysctl.conf
# allow testing with 2GB buffers
net.core.rmem_max = 2147483647
net.core.wmem_max = 2147483647
# allow auto-tuning up to 2GB buffers
net.ipv4.tcp_rmem = 4096 87380 2147483647
net.ipv4.tcp_wmem = 4096 65536 2147483647
```

2GB is the max allowable under Linux

WAN BDP = 12.5GB/s*92ms = 1150MB (autotuning set this to 1136MB)

LAN BDP = 12.5GB/s*54us = 675KB (autotuning set this to 2-9MB)

Manual buffer tuning made a big difference on the LAN:

- 50-60 Gbps vs 79 Gbps

zerocopy (sendfile) results

- iperf3 –Z option
- No significant difference on the LAN
- Significant improvement on the WAN
  - 36.5 Gbps vs 49 Gbps

IPv4 vs IPv6 results

- IPV6 is slightly faster on the WAN, slightly slower on the LAN
- LAN:
  - IPV4: 79 Gbps
  - IPV6: 77.2 Gbps
- WAN
  - IPV4: 36.5 Gbps
  - IPV6: 37.3 Gbps

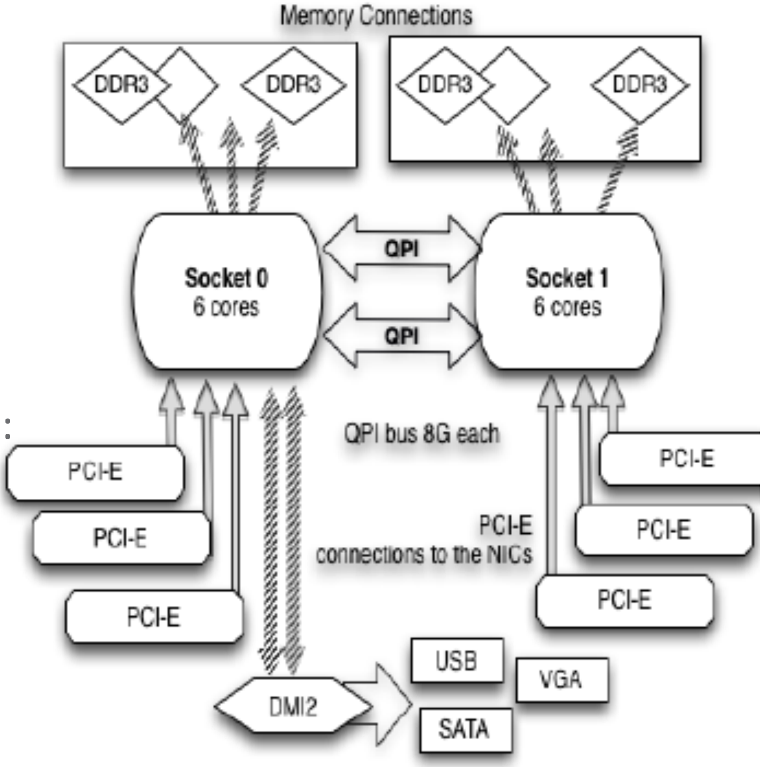
Don’t Forget about NUMA Issues

- Up to 2x performance difference if you use the wrong core.
- If you have a 2 CPU socket NUMA host, be sure to:
  - Turn off irqbalance
  - Figure out what socket your NIC is connected to:

    ```
    cat /sys/class/net/ethN/device/numa_node
    ```
  - Run Mellanox IRQ script:

    ```
    /usr/sbin/set_irq_affinity_bynode.sh 1 ethN
    ```
  - Bind your program to the same CPU socket as the NIC:

    ```
    numactl -N 1 program_name
    ```
- Which cores belong to a NUMA socket?
  - cat /sys/devices/system/node/node0/cpulist
  - (note: on some Dell servers, that might be: 0,2,4,6,...)



Settings to leave alone in CentOS7

Recommend leaving these at the default settings, and none of these seem to impact performance much

- Interrupt Coalescence
- Ring Buffer size
- LRO (off) and GRO (on)
- net.core.netdev_max_backlog
- txqueuelen
- tcp_timestamps

Tool Selection

- Both nuttcp and iperf3 have different strengths.
- nuttcp is about 10% faster on LAN tests
- iperf3 JSON output option is great for producing plots
- Use both! Both are part of the 'perfsonar-tools' package
  - Installation instructions at: http://fasterdata.es.net/performance-testing/network-troubleshooting-tools/

OS Comparisons

- CentOS7 (3.10 kernel) vs. Ubuntu 14.04 (4.2 kernel) vs. Ubuntu 16.04 (4.4 kernel)
  - Note: 4.2 kernel are about 5-10% slower (sender and receiver)
- Sample Results:
  - CentOS7 to CentOS7: 79 Gbps
  - CentOS7 to Ubuntu 14.04 (4.2.0 kernel): **69 Gbps**
  - Ubuntu 14.04 (4.2) to CentOS7**: 71 Gbps**
  - CentOS7 to Ubuntu 16.04 (4.4 kernel) : 73 Gbps
  - Ubuntu 16.04 (4.4 kernel) to CentOS7: 75 Gbps
  - CentOS7 to Debian 8.4 with 4.4.6 kernel: 73.6G
  - Debian 8.4 with 4.4.6 Kernel to CentOS7: 76G

BIOS Setting

- DCA/IOAT/DDIO: ON
  - Allows the NIC to directly address the cache in DMA transfers
- PCIe Max Read Request: Turn it up to 4096, but our results suggest it doesn't seem to hurt or help
- Turboboost: ON
- Hyperthreading: OFF
  - Added excessive variability in LAN performance (51G to 77G)
- node/memory interleaving: ??

PCI Bus Commands

Make sure you're installing the NIC in the right slot. Useful commands include:

Find your PCI slot:

```
    lspci | grep Ethernet
       81:00.0 Ethernet controller: Mellanox Technologies MT27700 Family
  [ConnectX-4]
```

Confirm that this slot is PCIeGen3 x16:

```
    lspci -s 81:00.0 -vvv | grep PCIeGen
       [V0] Vendor specific: PCIeGen3 x16
```

Confirm that PCI MaxReadReq is 4096B

```
  lspci -s 81:00.0 -vvv  | grep MaxReadReq
      MaxPayload 256 bytes, MaxReadReq 4096 bytes
```

If not, you can increase it using 'setpci'

- For more details, see: https://community.mellanox.com/docs/DOC-2496

Benchmarking vs. Production Host Settings

There are some settings that will give you more consistent results for benchmarking, but you may not want to run on a production DTN

Benchmarking:

- Use a specific core for IRQs:

```
/usr/sbin/set_irq_affinity_cpulist.sh 8 ethN
```

- Use a fixed clock speed (set to the max for your processor):

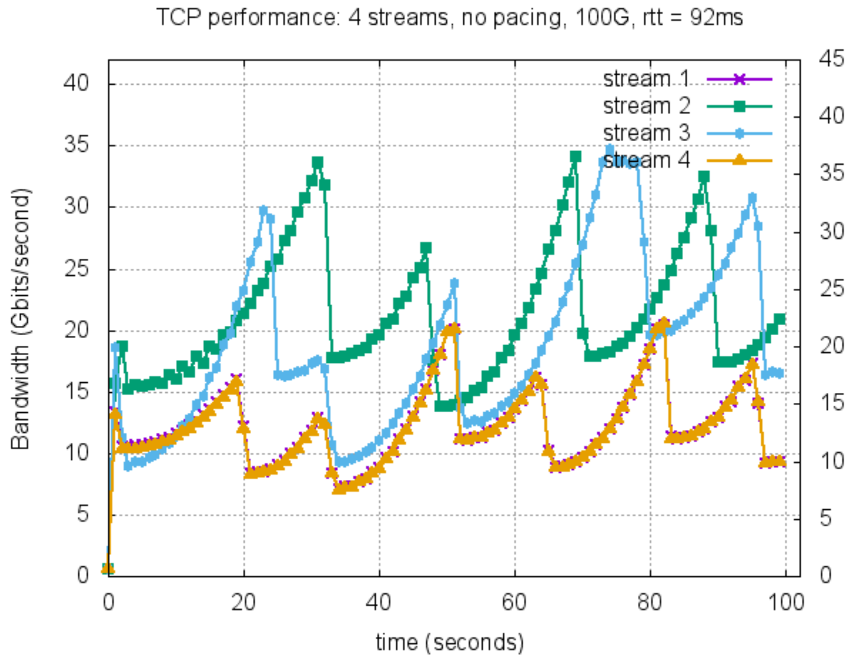
```
– /bin/cpupower -c all frequency-set -f 3.4GHz
```

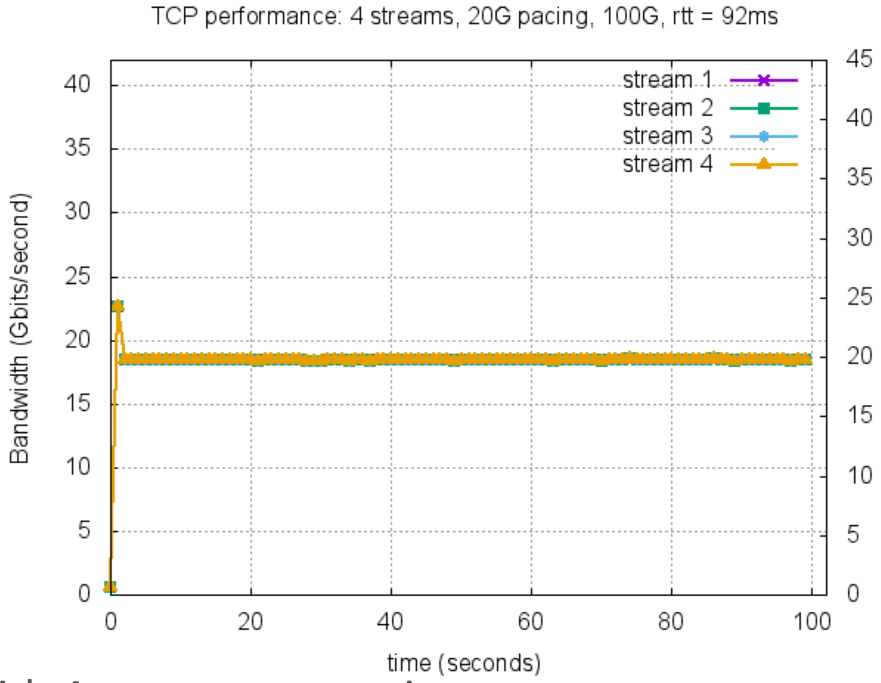
Production DTN:

```
/usr/sbin/set_irq_affinity_bynode.sh 1 ethN
/bin/cpupower frequency-set -g performance
```

FQ on 100G Hosts

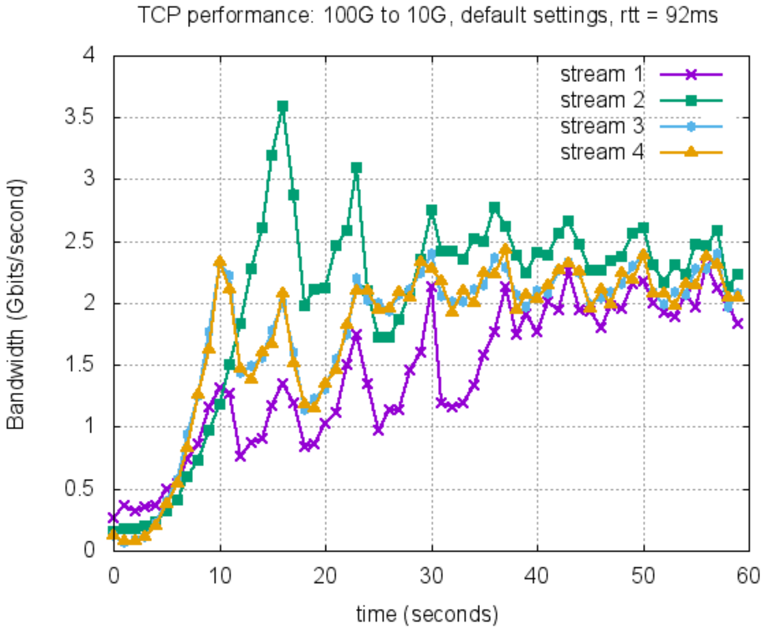
100G Host, Parallel Streams: no pacing vs 20G pacing

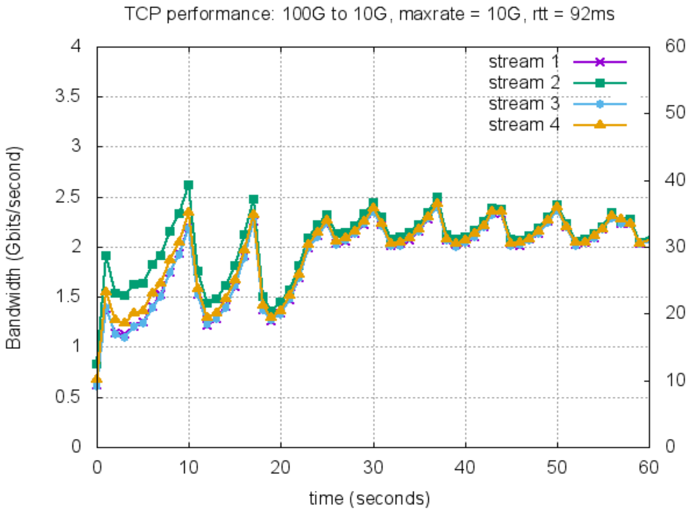


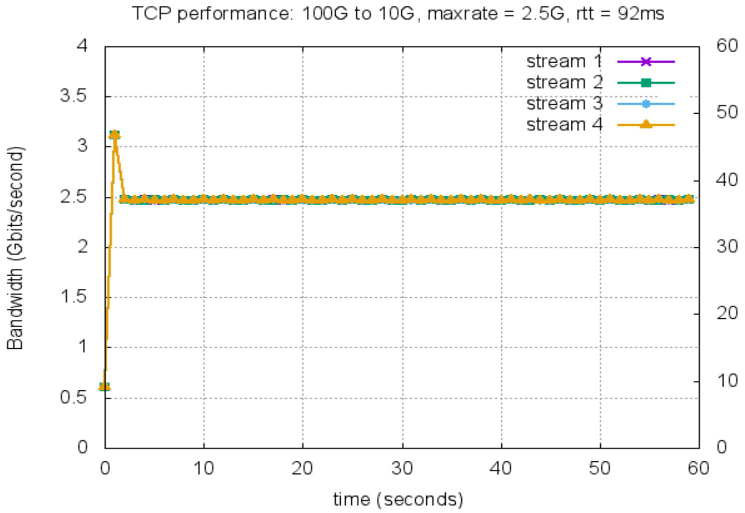


We also see consistent loss on the LAN with 4 streams, no pacing
Packet loss due to small buffers in Dell Z9100 switch?

100G Host to 10G Host


TCP performance: 100G to 10G, default settings, rtt = 92ms

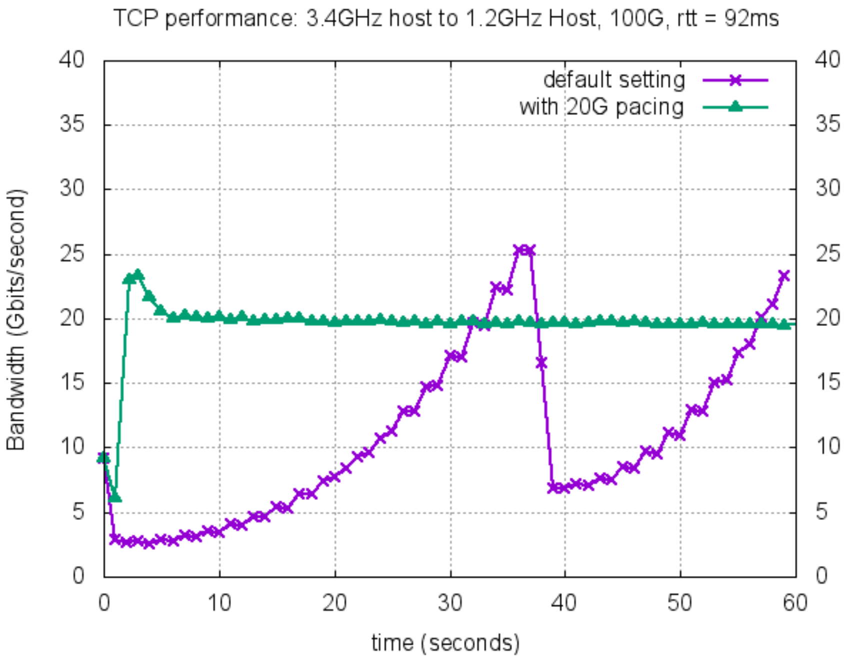

TCP performance: 100G to 10G, maxrate = 10G, rtt = 92ms


TCP performance: 100G to 10G, maxrate = 2.5G, rtt = 92ms

Fast Host to Slow host

Throttled the receive host using ‘cpupower’ command:

```
/bin/cpupower -c all frequency-set -f 1.2GHz
```



Summary of our 100G results

- New Enhancements to Linux Kernel make tuning easier in general.
- A few of the standard 10G tuning knobs no longer apply
- TCP buffer autotuning does not work well 100G LAN
- Use the 'performance' CPU governor
- Use FQ Pacing to match receive host speed if possible
- Important to be using the Latest driver from Mellanox
  - version: 3.3-1.0.4 (03 Jul 2016), firmware-version: 12.16.1020

What’s next in the TCP world?

- TCP BBR (Bottleneck Bandwidth and RTT) from Google
  - https://patchwork.ozlabs.org/patch/671069/
  - Google Group: https://groups.google.com/forum/#!topic/bbr-dev
- A detailed description of BBR will be published in ACM Queue, Vol. 14 No. 5, September-October 2016:
  - "BBR: Congestion-Based Congestion Control".

- Google reports 2-4 **orders of magnitude** performance improvement on a path with 1% loss and 100ms RTT.
  - Sample result: cubic: 3.3Mbps, BBR: 9150Mbps!!
  - Early testing on ESnet less conclusive, but seems to help on some paths

Initial BBR TCP results (bwctl, 3 streams, 40 sec test)

Remote Host	Throughput	Retransmits
perfsonar.nssl.noaa.gov	htcp: 183 bbr: 803	htcp: 1070 bbr: 240340
kstar-ps.nfri.re.kr	htcp: 4301 bbr: 4430	htcp:1641 bbr: 98329
ps1.jpl.net	htcp: 940 bbr: 935	htcp: 1247 bbr: 399110
uhmanoa-tp.ps.uhnet.net	htcp: 5051 bbr: 3095	htcp: 5364 bbr: 412348

Varies between 4x better and 30% worse, all with WAY more retransmits.

More Information

- http://fasterdata.es.net/host-tuning/packet-pacing/
- http://fasterdata.es.net/host-tuning/100g-tuning/
- Talk on Switch Buffer size experiments:
  - http://meetings.internet2.edu/2015-technology-exchange/detail/10003941/
- Mellanox Tuning Guide:
  - https://community.mellanox.com/docs/DOC-1523

- Email: BLTierney@es.net

Extra Slides

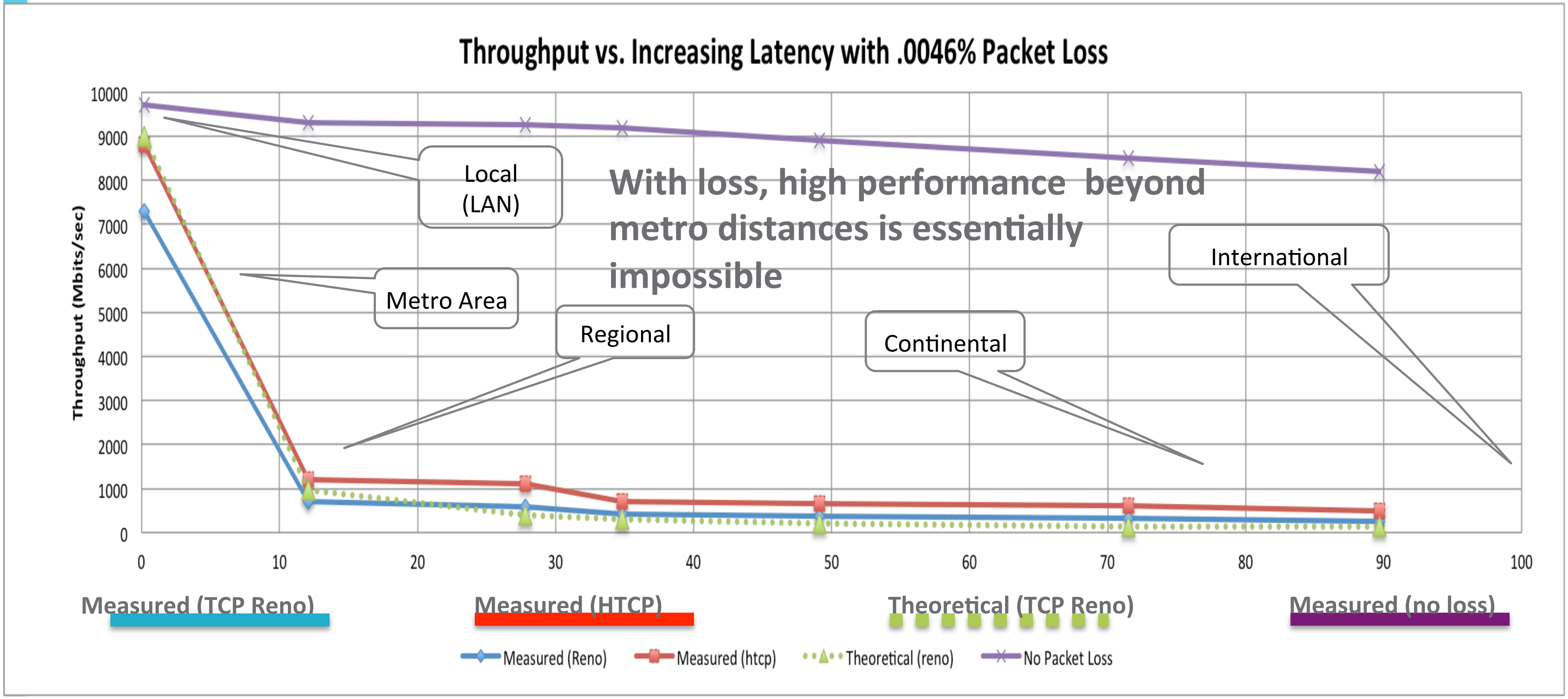
mlnx_tune command

- See: https://community.mellanox.com/docs/DOC-1523

Coalescing Parameters

- Varies by manufacturer
- usecs: Wait this amount of microseconds after 1st packet is received/ transmitted
- frames: Interrupt after this many frames are received or transmitted
- tx-usecs and tx-frames aren't as important as rx-usecs
- Due to the higher line rate, lower is better, until interrupts get in the way (at 100G, we are sending almost 14 frames/usec
- Default settings seem best for most cases

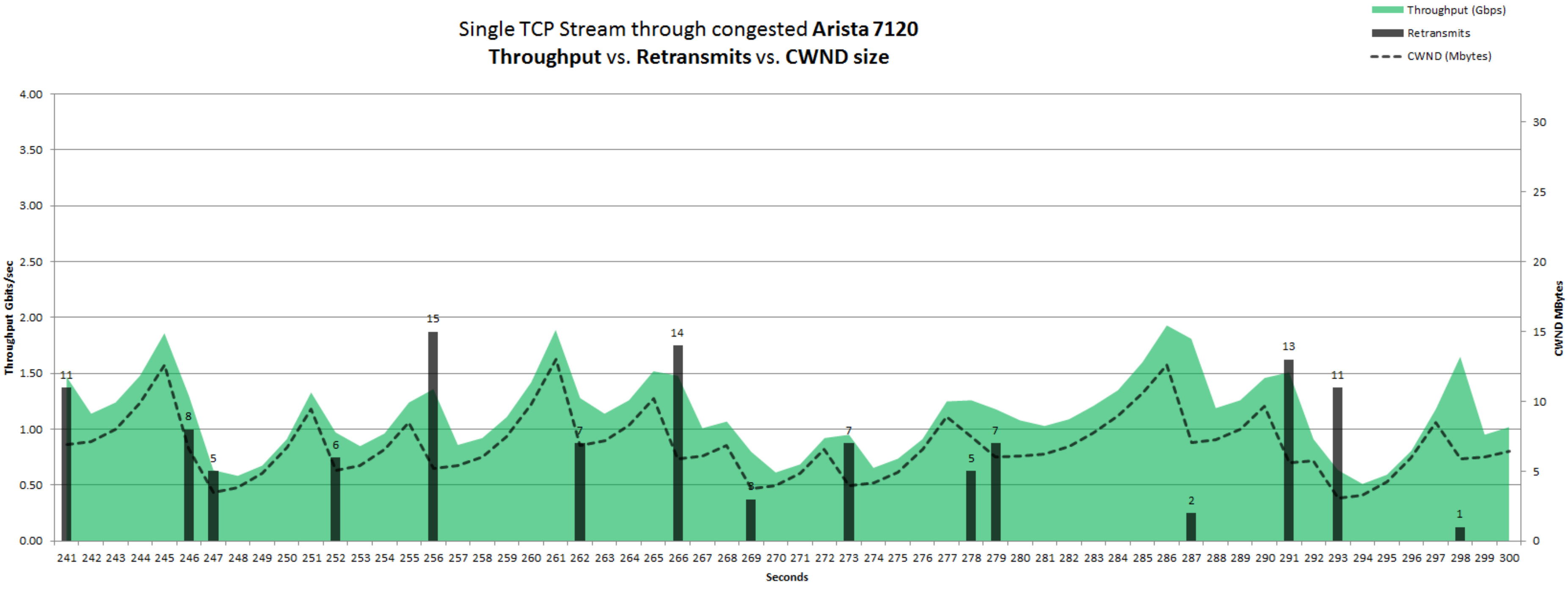
A small amount of packet loss makes a huge difference in TCP performance



Throughput vs. Increasing Latency with .0046% Packet Loss

With loss, high performance beyond metro distances is essentially impossible

TCP’s Congestion Control

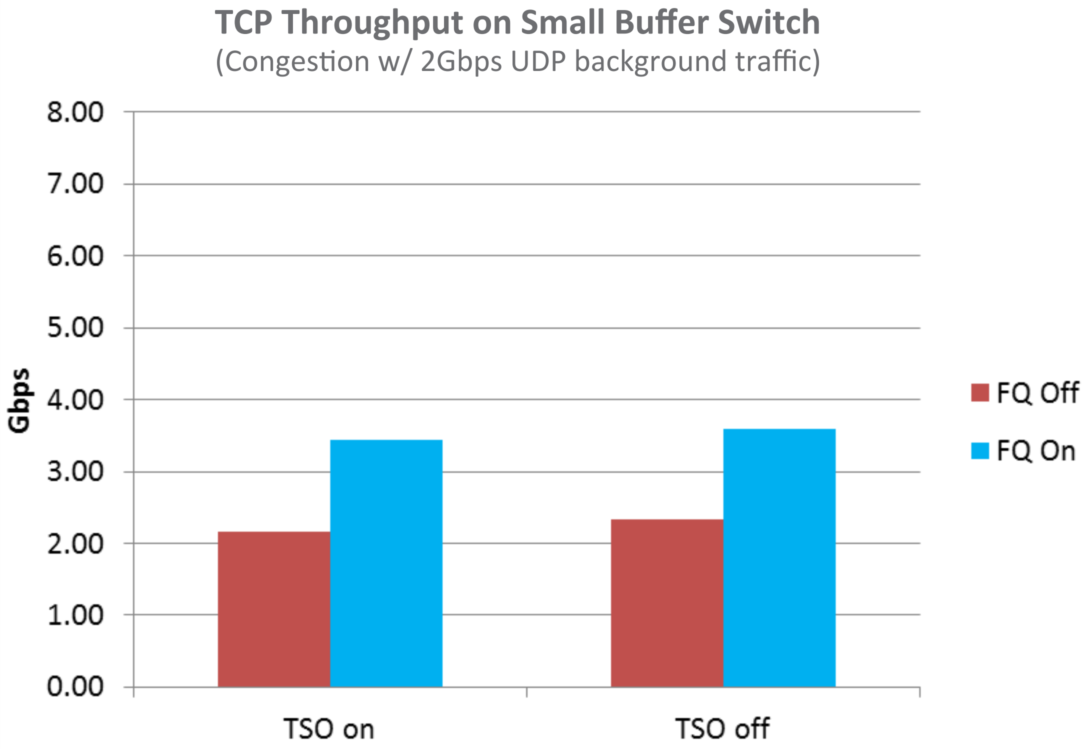


Single TCP Stream through congested **Arista 7120**
Throughput vs. **Retransmits** vs. **CWND size**

50ms simulated RTT
Congestion w/ 2Gbps UDP traffic
HTCP / Linux 2.6.32

Slide from Michael Smitasin, LBLnet

Fair Queuing and and Small Switch Buffers



Requires CentOS 7.2 or higher

```
tc qdisc add dev EthN root fq
```

Enable Fair Queuing

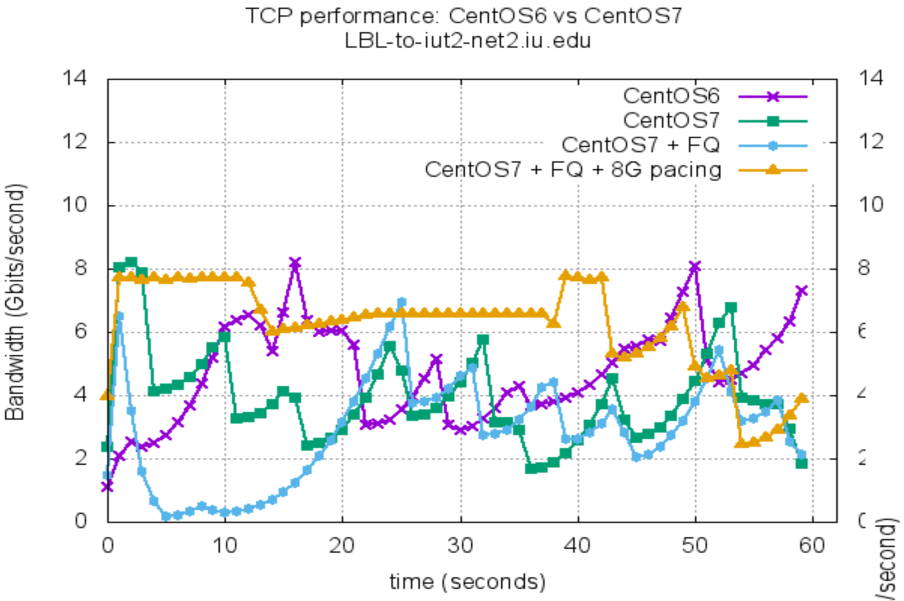
Pacing side effect of Fair Queuing yields ~1.25Gbps increase in throughput @ 10Gbps on our hosts

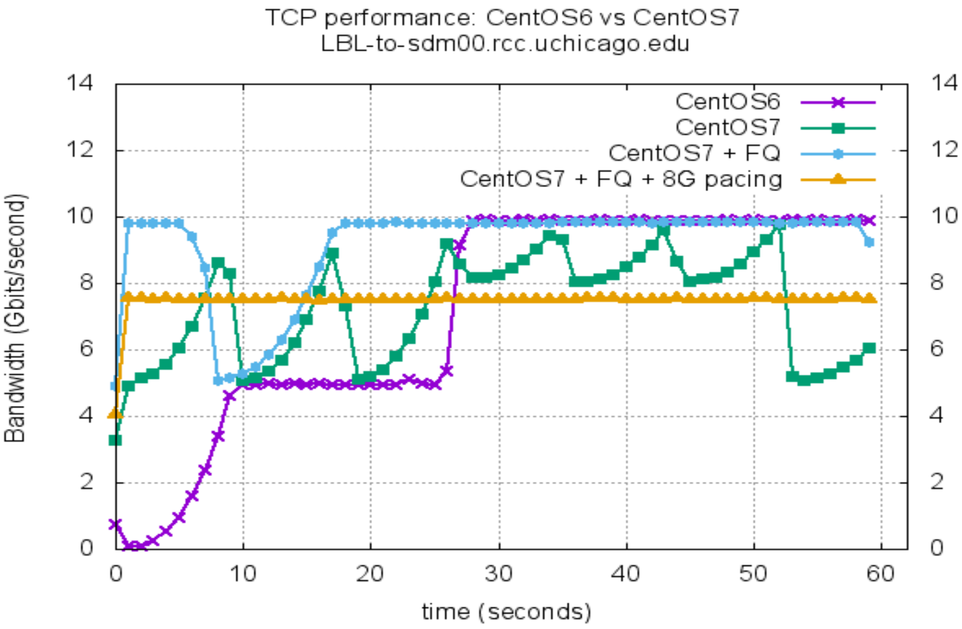
TSO differences still negligible on our hosts w/ Intel X520

Slide from Michael Smitasin, LBL

ESnet

More examples of pacing helping





Parallel Stream Test 1

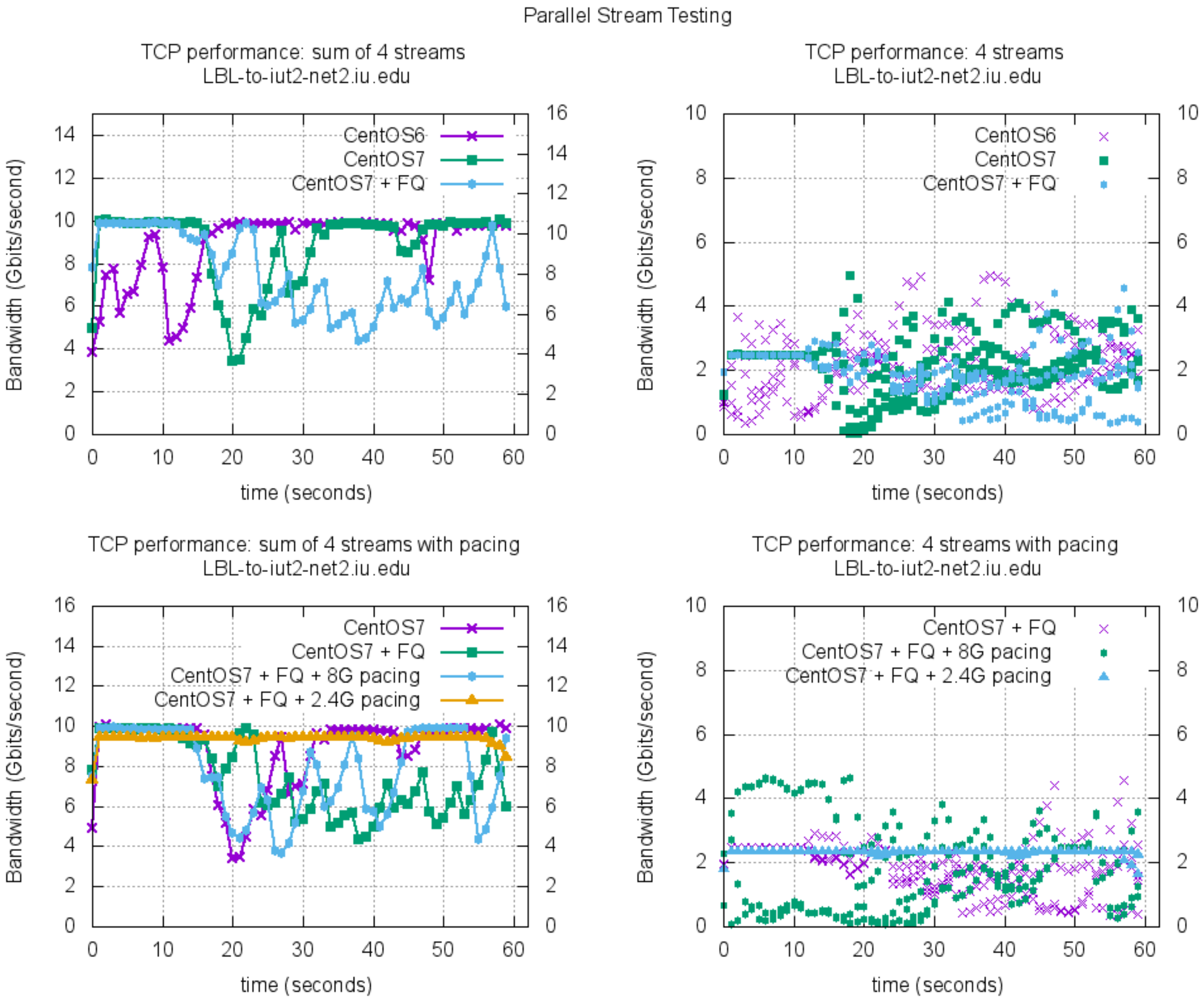
Left side:

sum of 4 streams

Right side:

tput of each stream

Streams appear to be much better balanced with FQ, pacing to 2.4 performed best

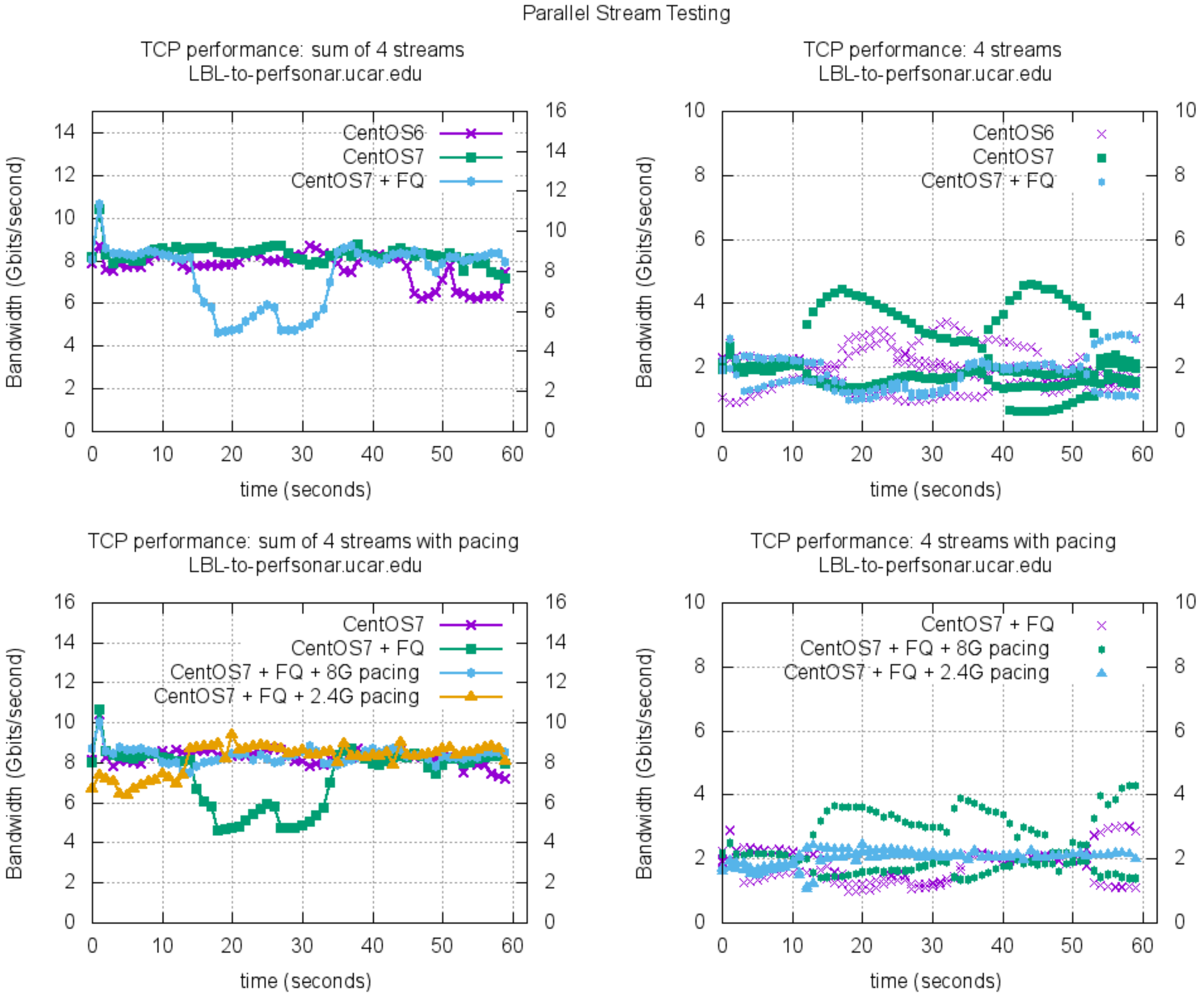


Parallel Stream Test 2

Left side:
 sum of 4 streams

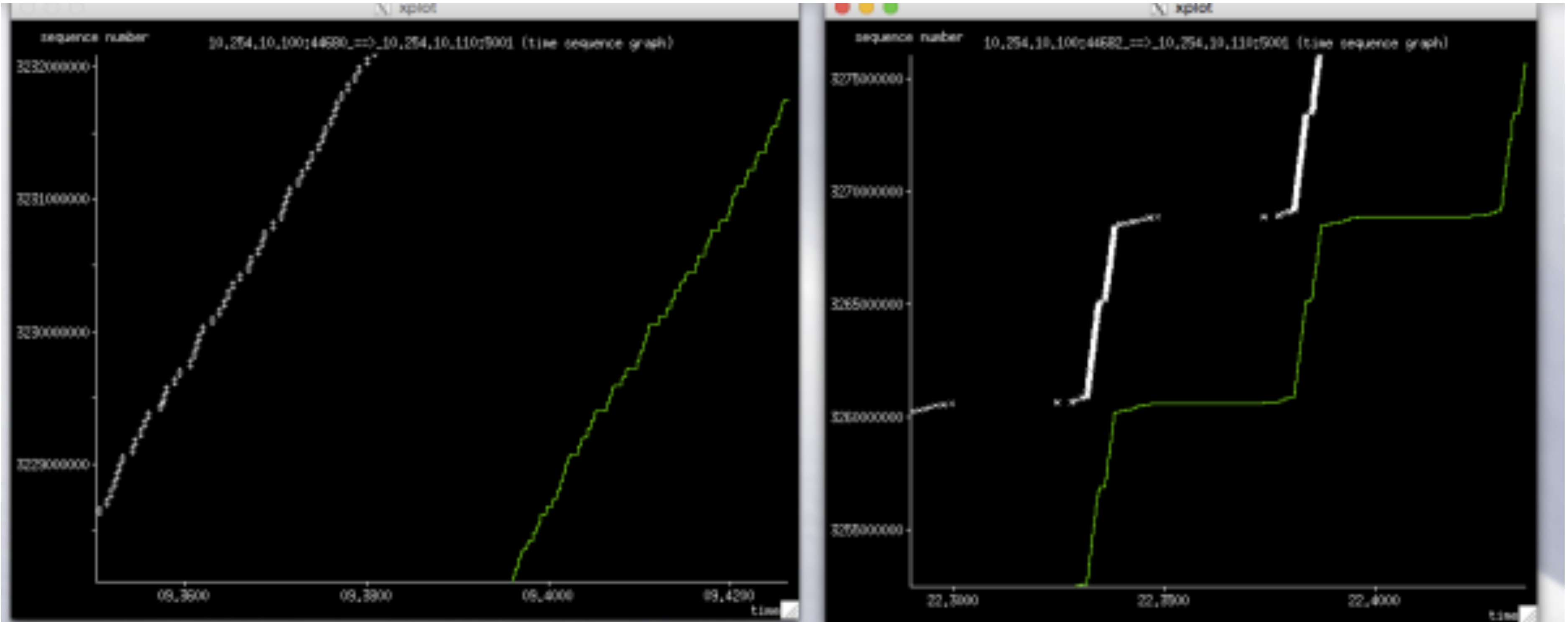
Right side:
 tput of each stream

Streams appear to be much better balanced with FQ



FQ Packets are much more evenly spaced

tcptrace/xplot output: FQ on left, Standard TCP on right



Run your own tests

- Find a remote perfSONAR host on a path of interest
  - Most of the 2000+ worldwide perfSONAR hosts will accept tests
    - See: http://stats.es.net/ServicesDirectory/
- Run some tests
  - bwctl -c hostname -t60 --parsable > results.json
- Convert JSON to gnuplot format:
  - https://github.com/esnet/iperf/tree/master/contrib

ESnet