Recent Linux Improvements that Impact TCP Throughput: Insights from R&E Networks

Brian Tierney (ESnet)
and Marcos Schwarz (RNP)

email: bltierney@es.net, marcos.schwarz@rnp.br

Background

Goal of this work is to:

- review recent enhancements to the Linux kernel that impact network throughput
  - and their potential impact on Data Transfer Node (DTN) performance.

In particular, we explore the benefits of:

- MSG_ZEROCOPY
- BIG TCP

Compare performance on three different Linux kernel versions, on Intel vs AMD processors, and over multiple round trip times.

MSG_ZEROCOPY

MSG_ZEROCOPY is a feature in Linux that allows for more efficient data transfers, through copy avoidance between user space and kernel space for network sockets.

- reduces CPU overhead and improves network performance
- available since 2017, but seems that most applications do not yet support it

An alternate form of zerocopy is the Linux sendfile call

- used transfer data from a file descriptor (typically a file) to a socket directly, without moving the data to user space.
- iperf3 has supported sendfile for many years, but the newer MSG_ZEROCOPY method is more general purpose, and easier to program into existing code

We used a patched version of iperf3 that supports MSG_ZEROCOPY

For more info see:

- https://www.kernel.org/doc/html/latest/networking/msg_zerocopy.html
- https://github.com/esnet/iperf/pull/1720

BIG TCP

GSO and GRO (Generic Send/Receive Offload) techniques allows Linux to use internally a super-sized TCP packet of 64KB, to reduce CPU cycles and interrupts

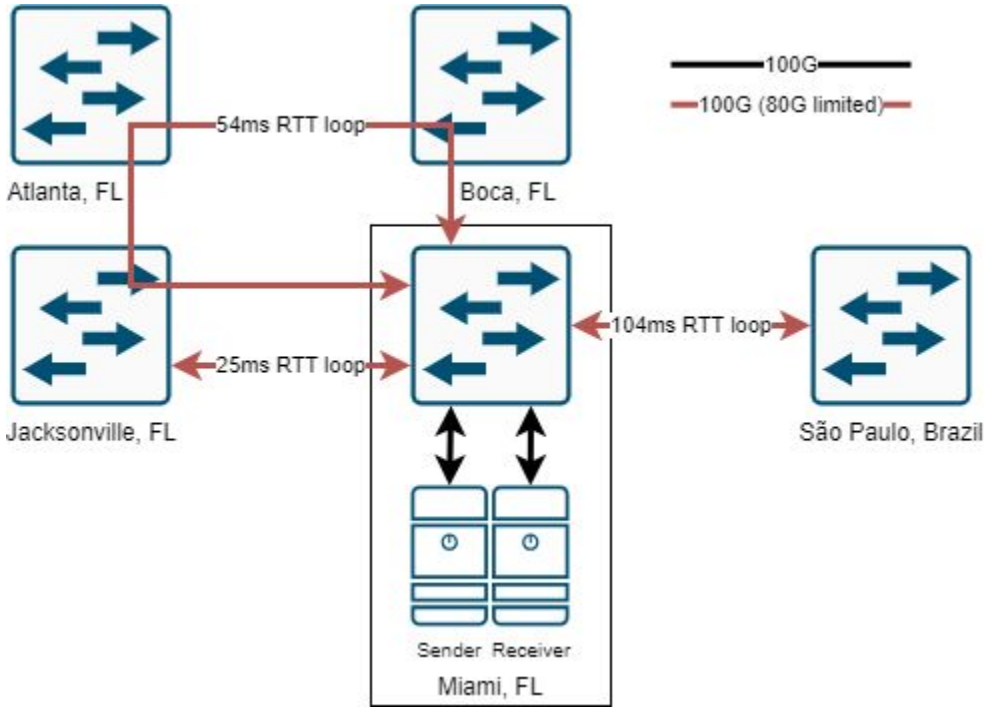
- This TCP packet will be fragmented to the MTU size by the NIC driver (i.e. 1500B or 9000B)

BIG TCP increases GSO/GRO packet sizes up to 512KB

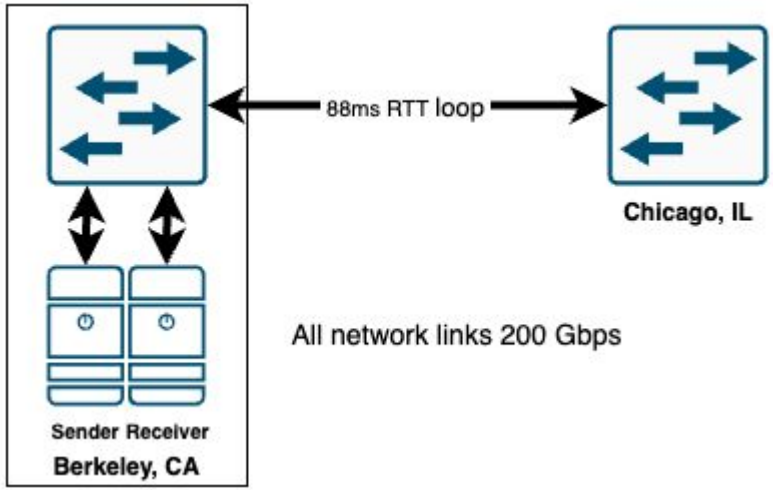
- Initial implementation for IPv6 in Linux 5.19 and for IPv4 since 6.3
- MSG_ZEROCOPY and BIG TCP cannot be used simultaneously without a custom built kernel

Test Environments

AmLight



ESnet



Note: AmLight hosts are Intel, ESnet hosts are AMD

Why 2 Testbeds?

Testbeds have different characteristics:

ESnet:

- AMD Hosts + Nvidia Connectx-7
- 200 Gbps dedicated paths

AmLight:

- more latency options
- 100 Gbps shared paths
  - 80G was available for our testing
- Intel Hosts + Connectx-5

Host Tuning: /etc/sysctl.conf

```
net.core.rmem_max=2147483647
net.core.wmem_max=2147483647
net.ipv4.tcp_rmem=4096 131072 2147483647
net.ipv4.tcp_wmem=4096 16384 2147483647
net.ipv4.tcp_no_metrics_save=1
net.core.default_qdisc=fq
# needed for MSG_ZEROCOPY
net.core.optmem_max = 1048576

# other tuning
# increase ring buffer size (AMD hosts only)
/usr/sbin/ethtool -G eth100 rx 8192 tx 8192
# turn off SMT (aka Hyper Threading)
echo off > /sys/devices/system/cpu/smt/control
# set CPU performance governor
cpupower frequency-set -g performance
```

Other Tuning

MTU = 9K

disable irqbalance

iommu=pt (set in /etc/default/grub)

- this is important!
- increased 8-stream throughput from 80 Gbps to 181 Gbps on the ESnet AMD hosts running the 5.15 kernel.

See: https://fasterdata.es.net/host-tuning/linux/100g-tuning/

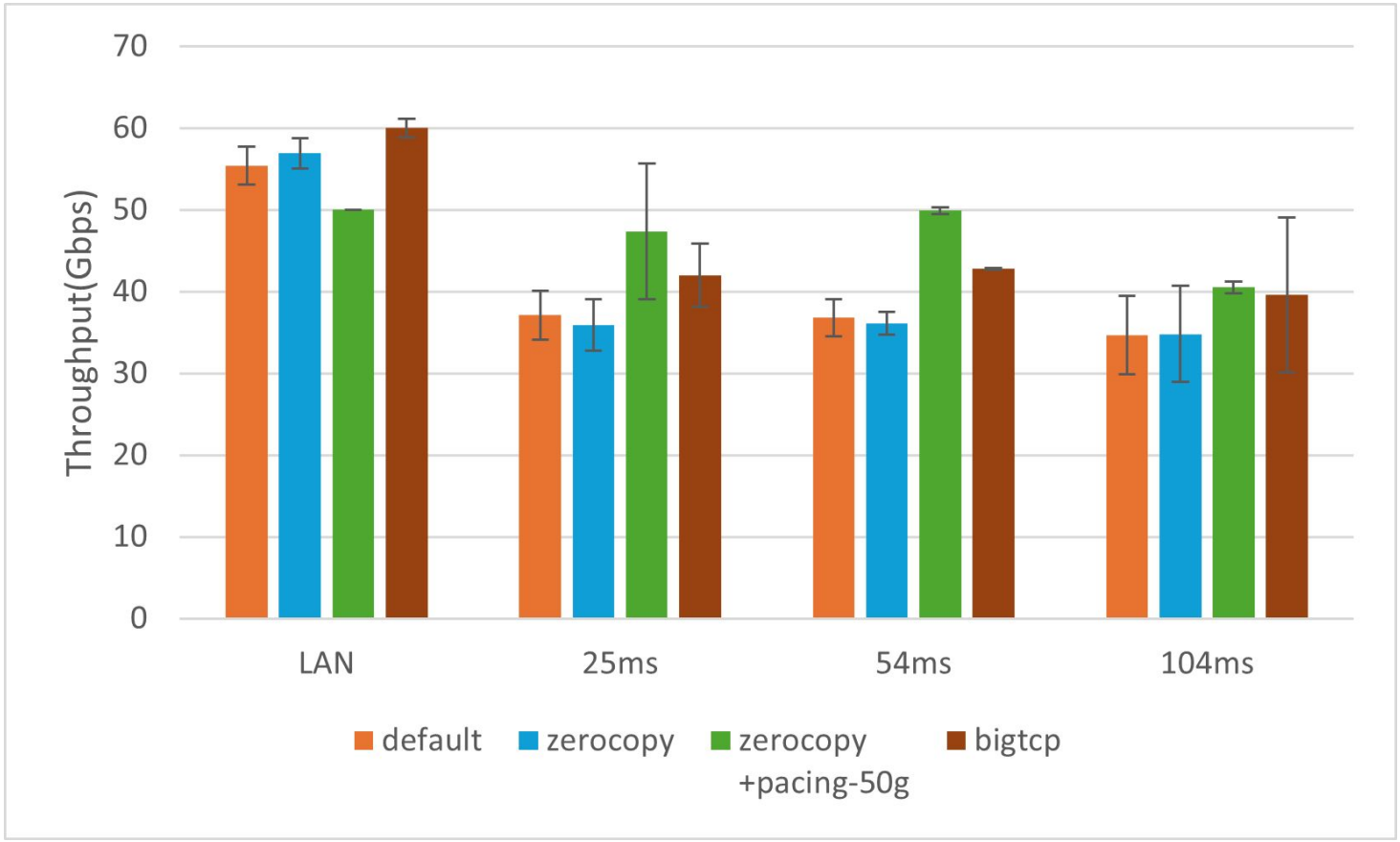
Test Consistency: Proper Core selection

- Initially we had trouble getting consistent results
- Tuning helped, and in addition to the above tuning we found core selection to be important
- Best to separate application cores from IRQ cores
  - set_irq_affinity_cpulist.sh 0-7 ethN
  - numactl -C 8-15 iperf3 (client and server)
- set_irq_affinity_cpulist.sh is a script provided by Nvidia/Mellanox

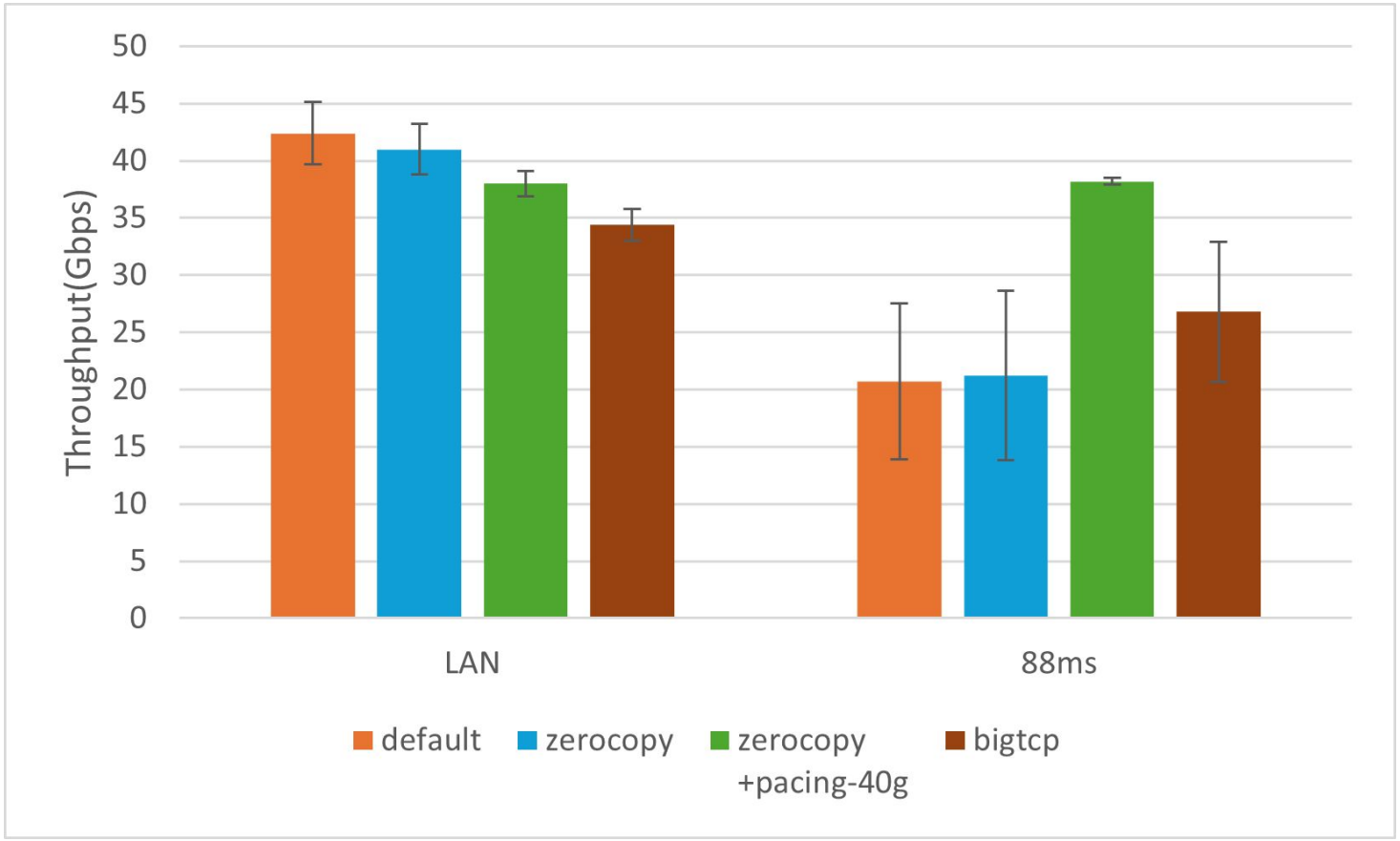
Data Collection

- Results were collected using the ESnet 'Network Test Harness'
  - available on github: https://github.com/esnet/testing-harness
- All tests were run for 60 seconds, and run a minimum of 10 times.
- The test harness includes the ability to run **mpstat** along with **iperf3** to monitor CPU usage, and the ability to enable/disable BIG TCP
  - pacing managed via iperf3 **--fq-rate** option

Results: Single Stream, MSG_ZEROCOPY, AmLight



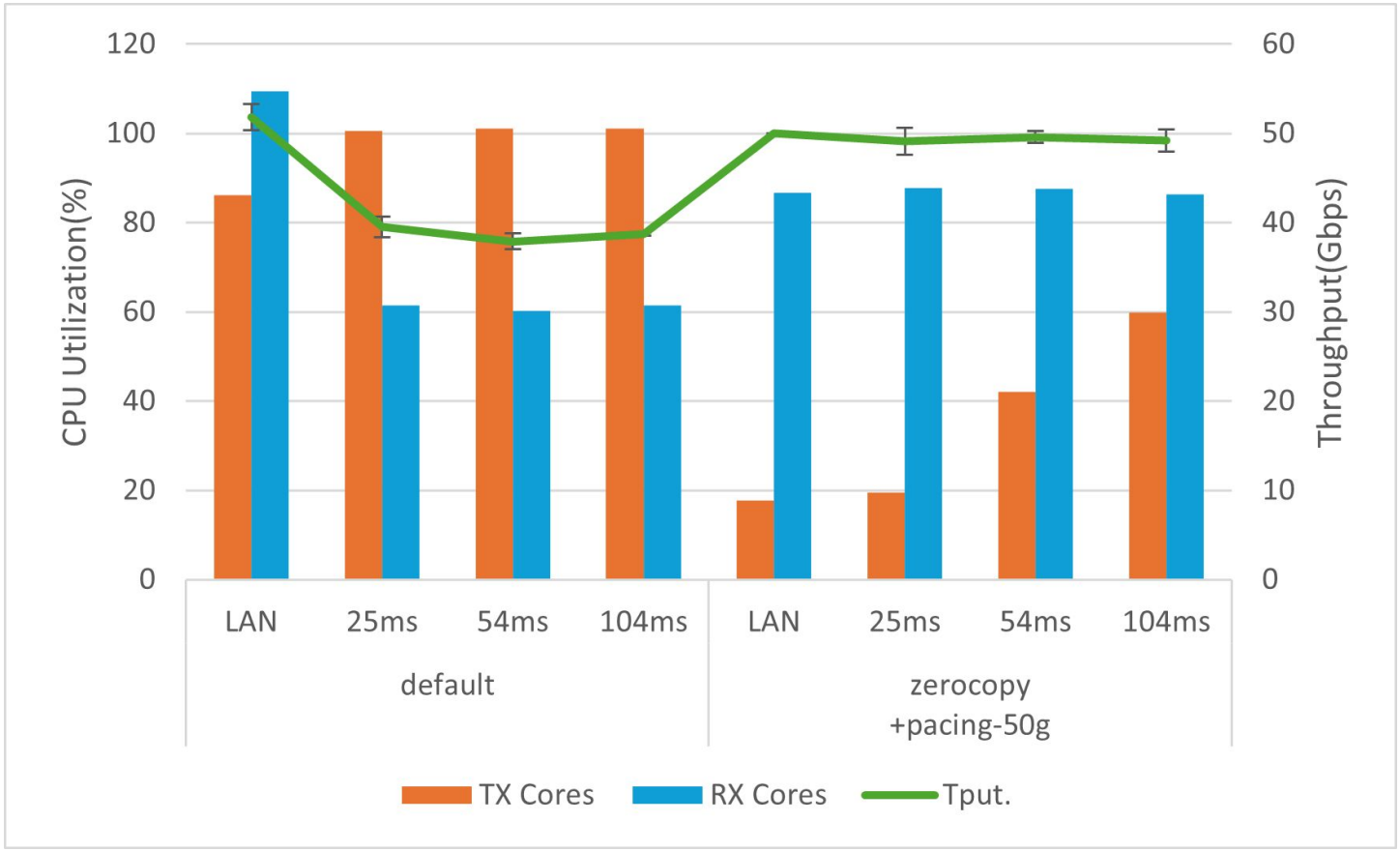
Results: Single Stream, MSG_ZEROCOPY, ESnet



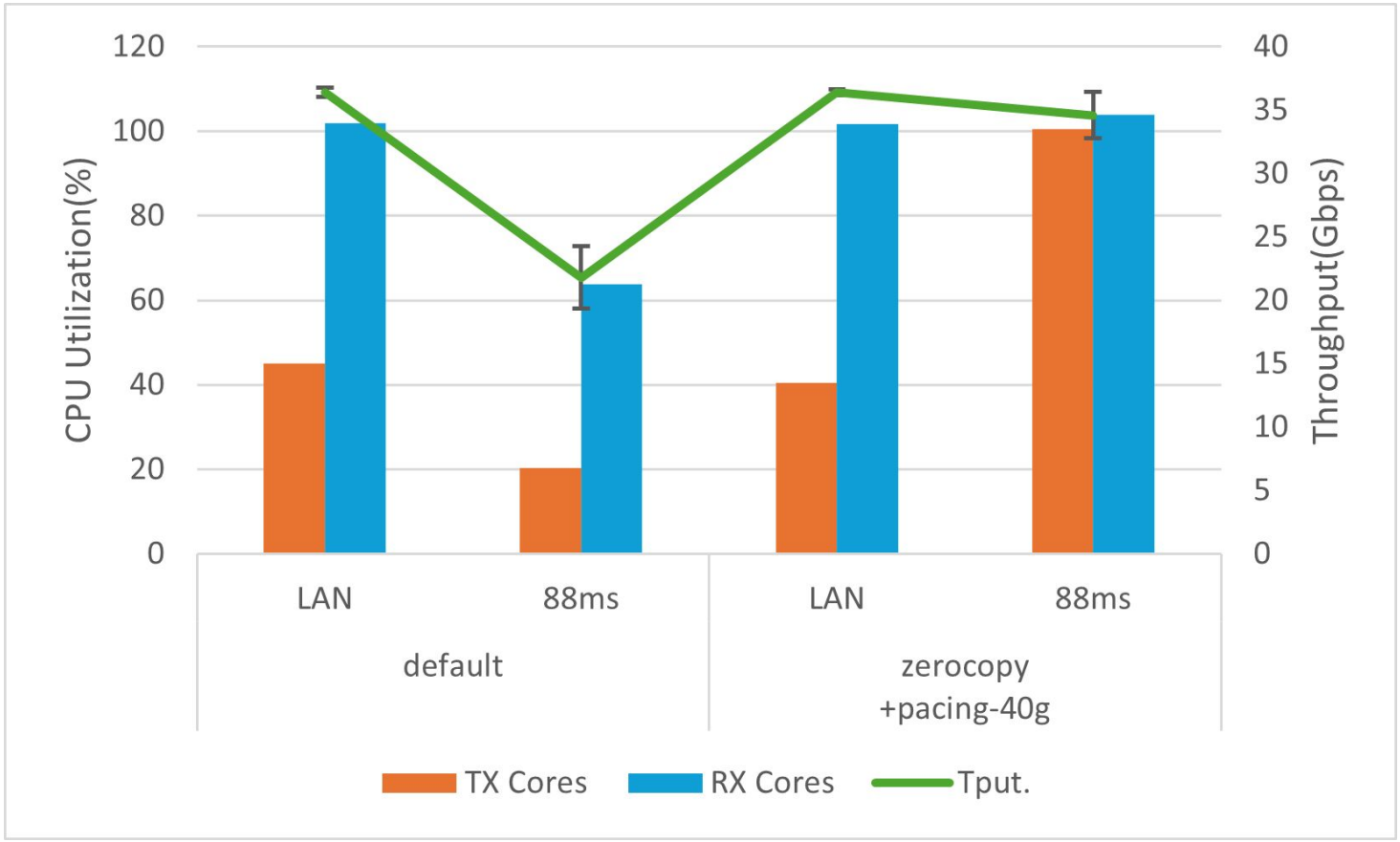
The importance of pacing (**--fq-rate**)

- Spreads out the 'packet train' and reduces CPU utilization on receive side
  - this reduces the likelihood of packet drops by the receive host
- MSG_ZEROCOPY reduces CPU utilization, but will only provide a consistent throughput increase when used with pacing
- We submitted a patch to **iperf3** to support pacing above 32 Gbps

Results: CPU Utilization, Intel Host



Results: CPU Utilization, AMD Host



Summary of CPU Results

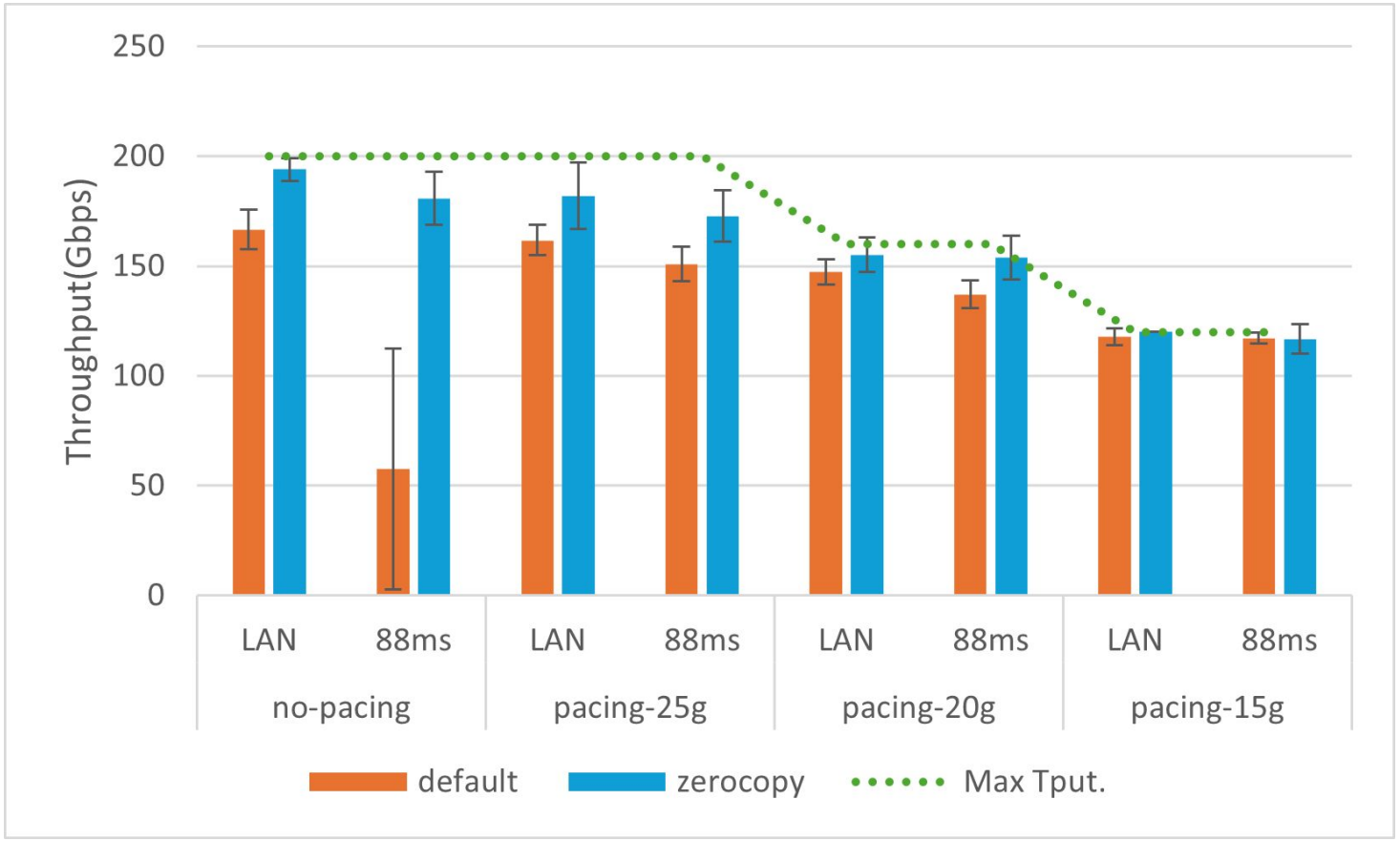
Default settings

- Receiver CPU utilization is the bottleneck on LAN and is directly proportional to throughput on all tests
- Sender CPU is the bottleneck on WAN

MSG_ZEROCOPY with Pacing at 50G

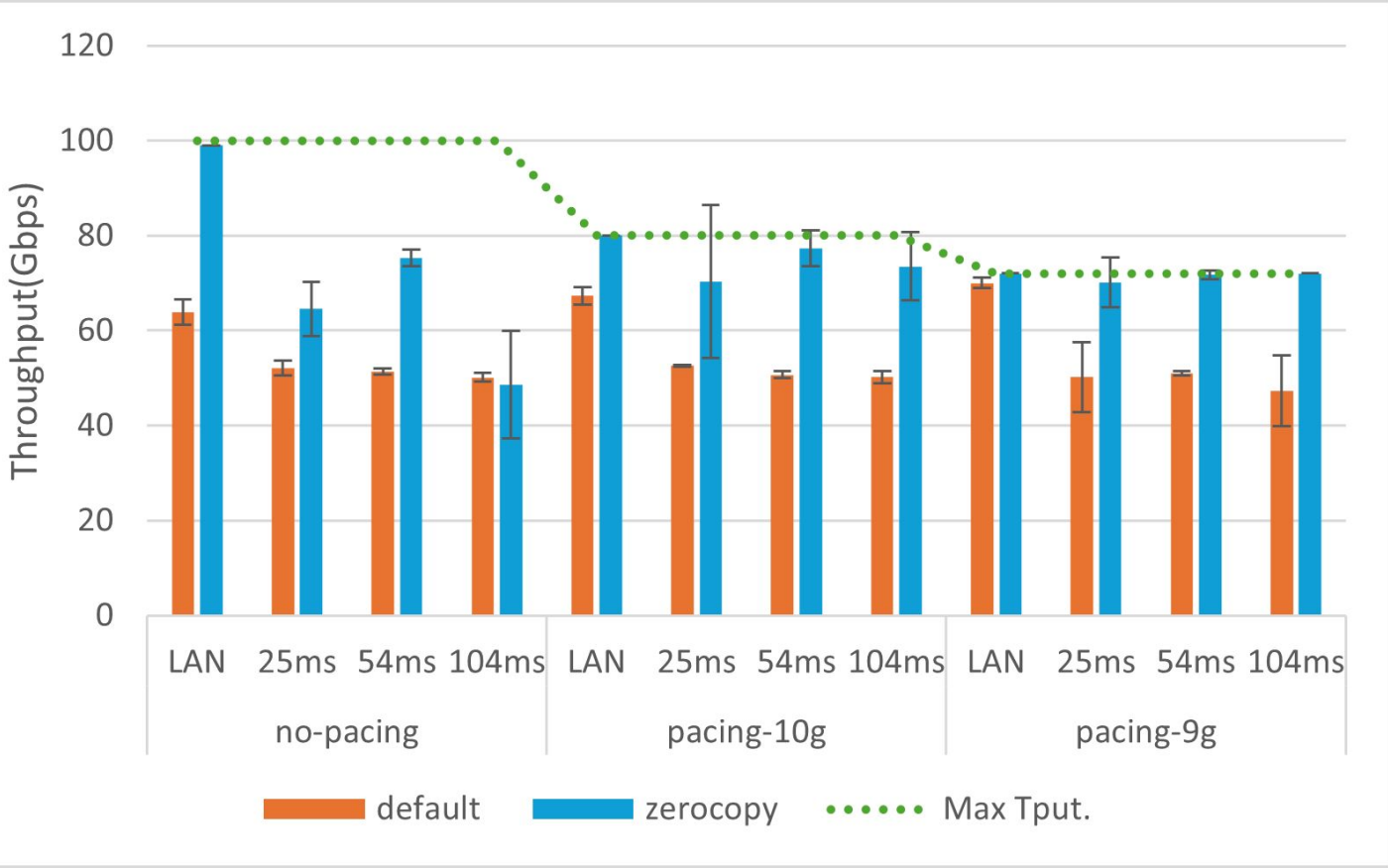
- MSG_ZEROCOPY significantly reduces sender CPU utilization, and receiver CPU is the new bottleneck for WAN
- **Throughput is constant of all paths, regardless of RTT**

Results: Multi-Stream, ESnet



Note:
Max Tput = max possible based on pacing settings

Results: Multi-Stream, AmLight



Pacing Results Summary

LAN Results

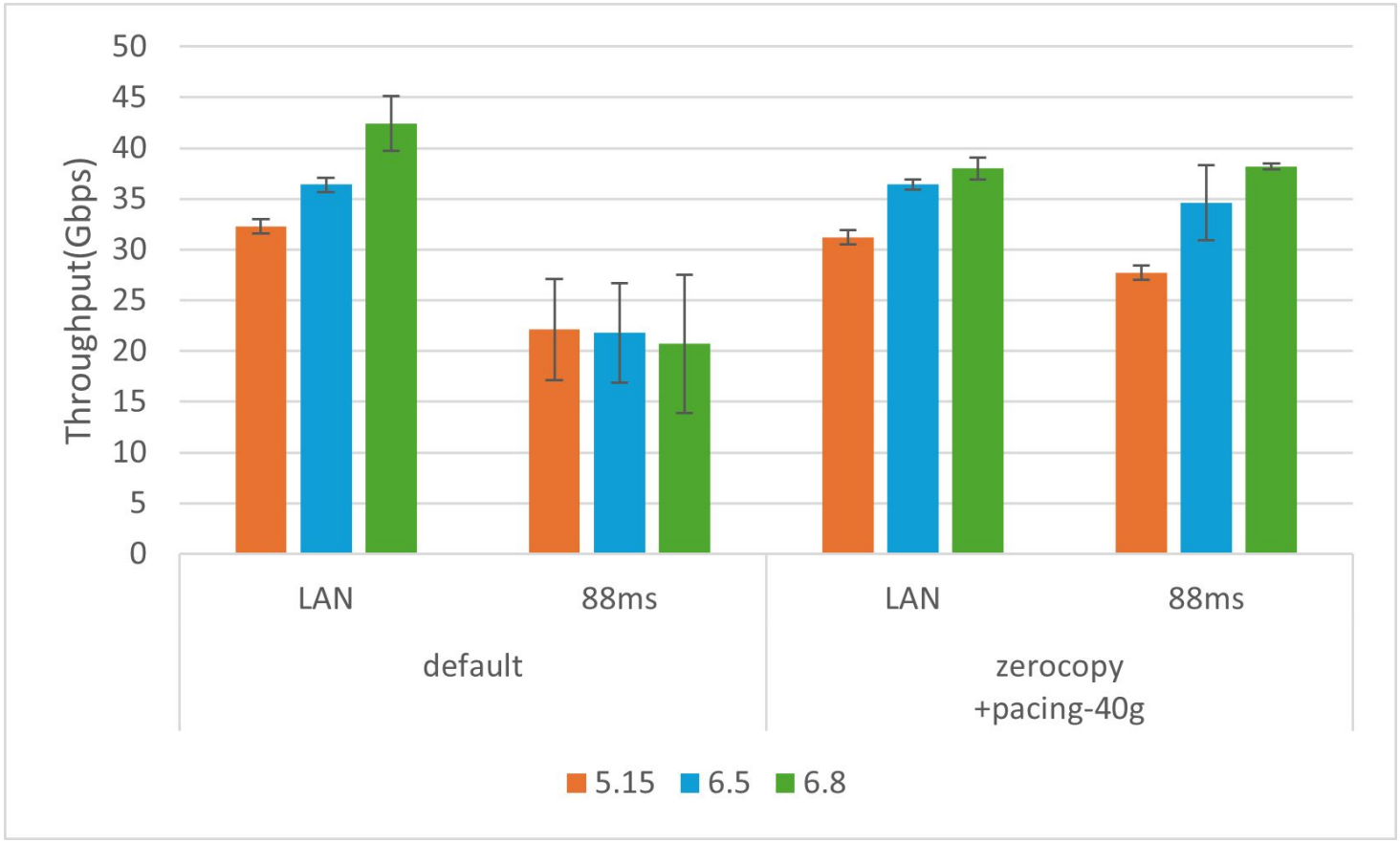
Test Config	Ave Tput	Retr	Min	Max	stdev
unpaced	166 Gbps	242	154	177	8.1
25 Gbps / stream	166 Gbps	70	146	172	9.1
20 Gbps / stream	147 Gbps	83	115	153	12.3
15 Gbps / stream	80 Gbps	118	118	119	0.1

WAN Results

Test Config	Ave Tput	Retr	Min	Max	stdev
unpaced	127 Gbps	73K	119	137	7.2
25 Gbps / stream	136 Gbps	22K	104	157	15.8
20 Gbps / stream	131 Gbps	8K	118	142	8.9
15 Gbps / stream	115 Gbps	4K	108	119	4.7

Note the large stdev until pace all the way down to 15 Gbps, both LAN and WAN

Results: kernel version comparison, ESnet



Summary of Kernel Version Results

Using the newest Linux kernel (v6.8), we see up to 38% improved performance on the WAN, and 30% better on a LAN, compared with the v5.15 kernel.

v6.8 is the default on Ubuntu 24, and with Ubuntu 22, it is quite easy to upgrade to the newest kernel using apt.

To install v6.8 on Ubuntu 22

- apt install linux-image-generic-hwe-22.04

On RHEL-based systems, you can install the newest kernel from the elrepo project

- https://elrepo.org/

BBR Testing

We ran a set of tests comparing Cubic with BBRv1 and BBRv3 in this environment

Since there was no congestion BBR did not have an impact in this environment

We did notice that the throughput of parallel stream BBR tests without pacing were **worse** than CUBIC

- BBR flows were more likely to step on each other, leading to way more retransmits

Conclusions: Host Benchmarking

- Use tuning settings from https://fasterdata.es.net/host-tuning/linux/100g-tuning/
- For maximum performance and flow stability use separate CPU cores for IRQ and your test tool;
- If possible, use a tool that supports MSG_ZEROCOPY, increase optmem max, and use packet pacing. This should provide more stable flows and up to 35% faster throughput;
- Use kernel 6.8 for up to 38% better performance on WAN and 30% better performance on LAN compared with kernel 5.15;
- Use use some form of packet pacing to avoid overrunning receive host.

Conclusions: DTN Tuning

- With enough parallel streams, most of this extra tuning is not needed
- Pacing is recommended so streams do not interfere with each other.
- For a production DTN, determine optimal pacing for your environment
  - If serving data to mainly 10G clients, it might be best to pace to 1 Gbps per flow.
  - If mainly sending data to other 100G hosts, 5-8 Gbps/flow might be fine.
  - Note that the **tc** command can be used to pace all flows on the host (see Fasterdata)
- A heavily used DTN that is running out of CPU cycles would benefit from using tools that support MSG_ZEROCOPY.
  - Software that does user-level checksums, such as Globus, may benefit from the extra CPU cycles.

fasterdata.es.net

- Popular site for information / advice on tuning and techniques for high speed data transfers since the late 1990s.
- Most visited sections include:
  - https://fasterdata.es.net/host-tuning/linux/
  - https://fasterdata.es.net/host-tuning/linux/100g-tuning/
  - https://fasterdata.es.net/host-tuning/linux/packet-pacing/
  - https://fasterdata.es.net/science-dmz/

- We are looking for help updating the network device section:
  - https://fasterdata.es.net/network-tuning/router-tuning/
  - email updates to fasterdata@es.net

For More Information

The results in this talk are from a paper/talk at the IEEE workshop "Innovating the Network for Data-Intensive Science (INDIS)", part of SC24:

- https://scinet.supercomputing.org/community/indis/

Our INDIS paper is available here:

- https://fasterdata.es.net/assets/INDIS_2024_final.pdf
- https://ieeexplore.ieee.org/document/10820770

All raw test data is available on github:

- https://github.com/marcosfsch/INDIS-2024