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

Marcos Schwarz (RNP) and Brian Tierney (ESnet)

SC24 INDIS workshop

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

BIG TCP

GSO and GRO offload techniques allows Linux to use internally a super-sized TCP packet of 64KB, to reduce CPU cycles and interrupts

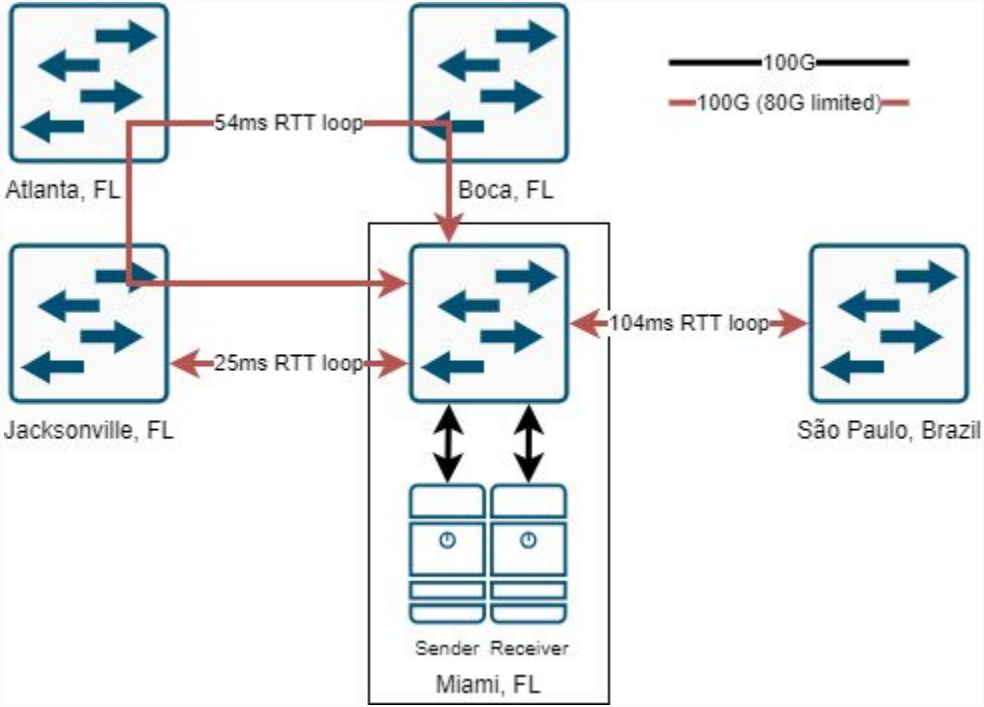
- This TCP packet will be fragmented to the MTU size by the NIC driver (i.e. 1500B or 9000B)

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

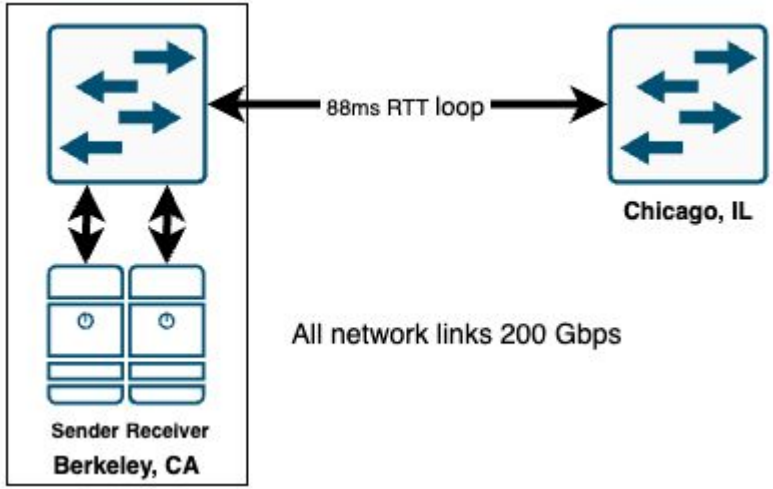
- Initial implementation for IPv6 in Linux 5.19 and for IPv4 since 6.3
- MSG_ZEROCOPY and BIG TCP cannot be used simultaneously without a custom built kernel

Test Environments

AmLight



ESnet



Note: AmLight hosts are Intel, ESnet hosts are AMD

Why 2 Testbeds?

Testbed have different characteristics:

ESnet:

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

AmLight:

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

Host Tuning: /etc/sysctl.conf and Other Tuning

```
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
```

MTU = 9K

disable irqbalance

iommu=pt

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

Best to separate application cores from IRQ cores

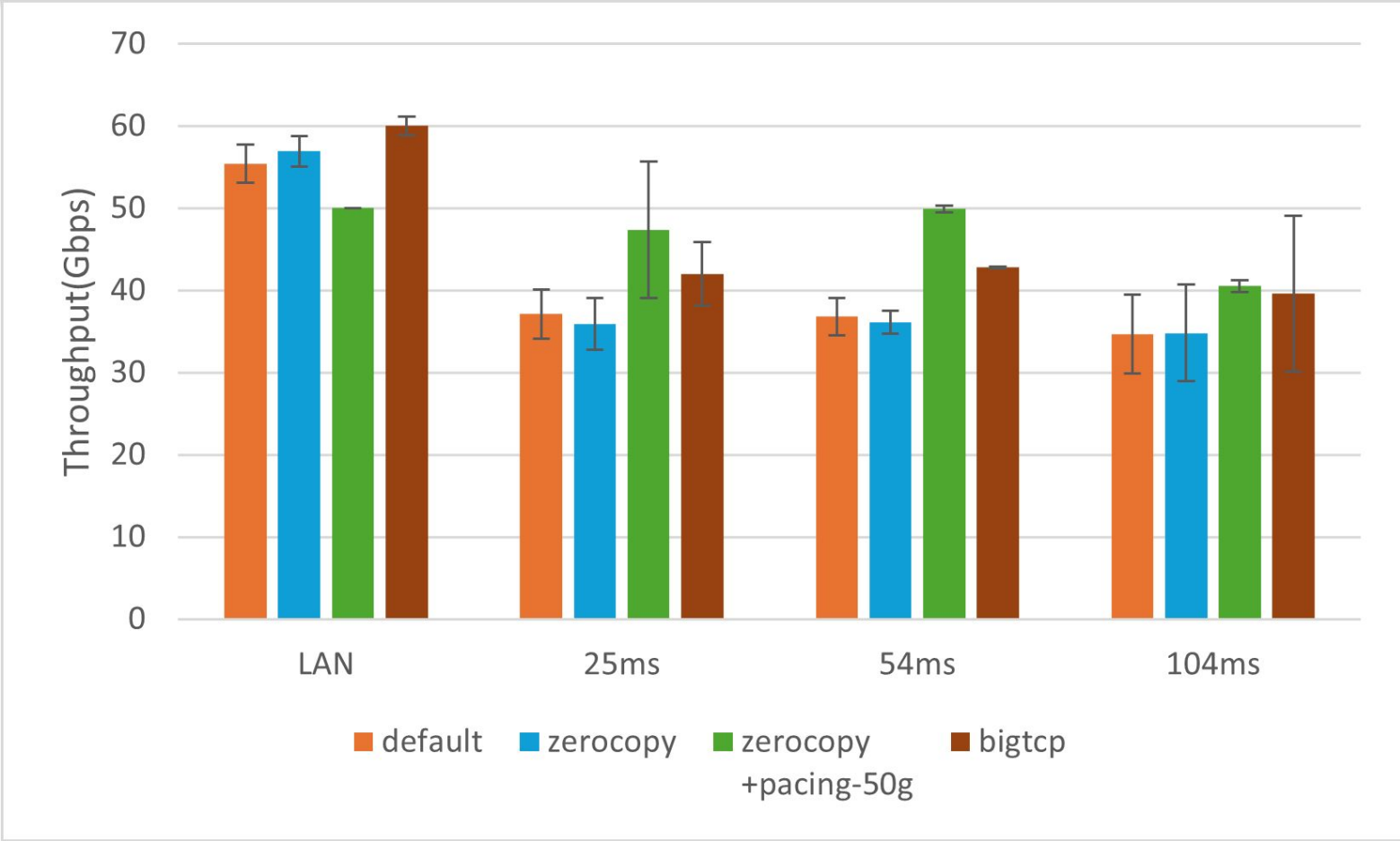
- set_irq_affinity_cpulist.sh 0-7 ethN
- numactl -C 8-15 iperf3 (client and server)

set_irq_affinity_cpulist.sh is a script provided by Nvidia/Mellanox

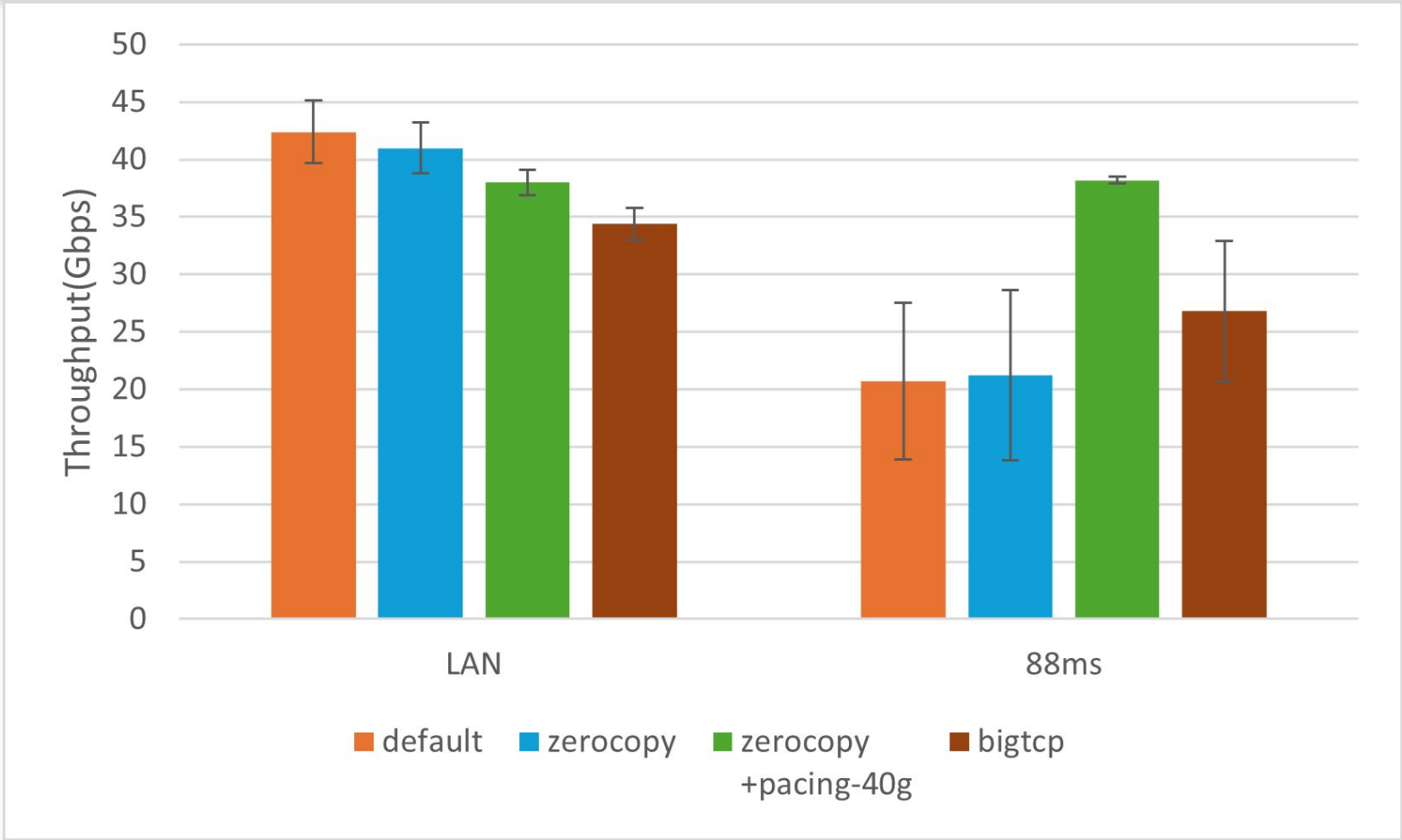
Data Collection

- 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, Linux 6.8



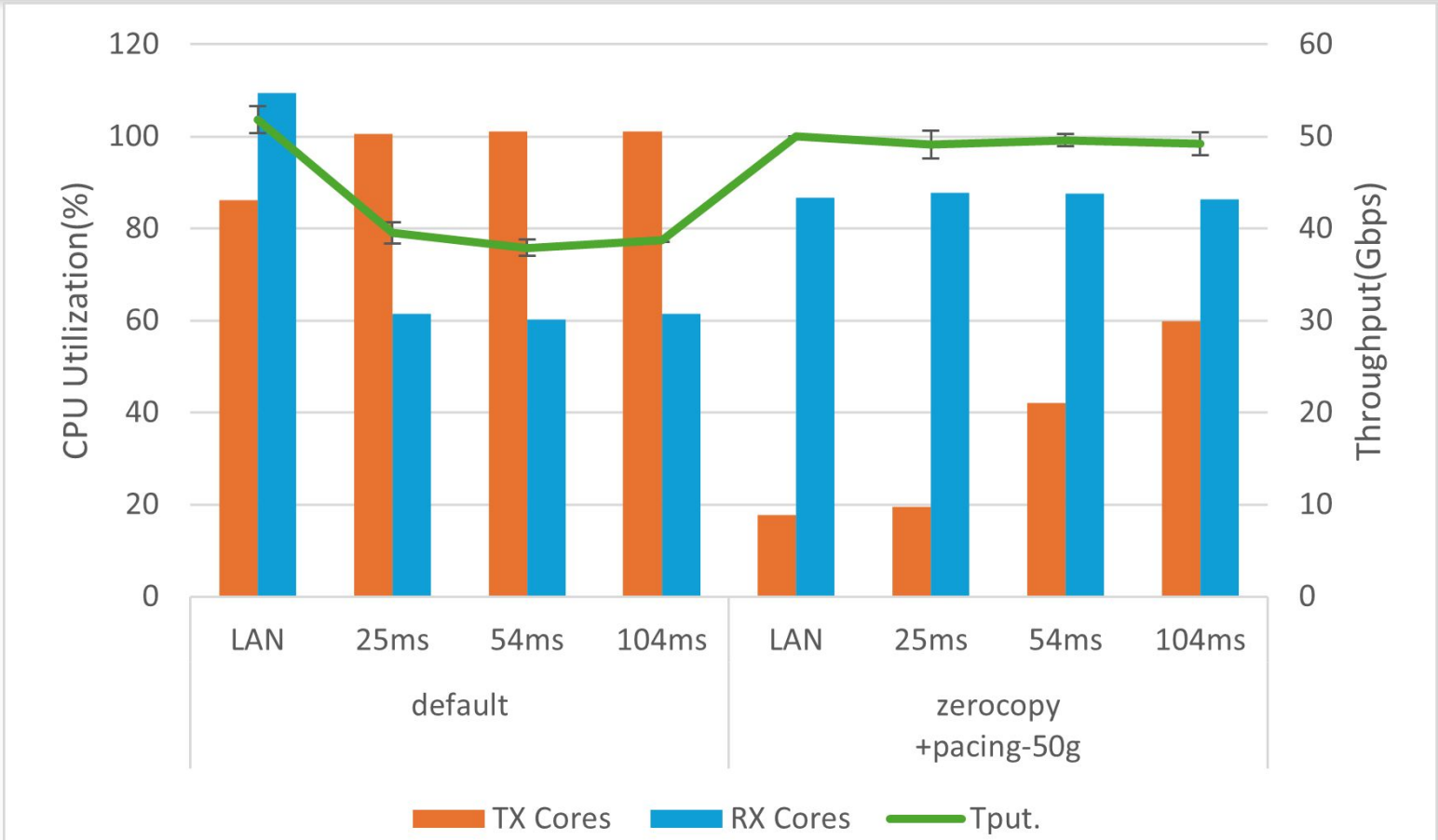
Results: Single Stream, MSG_ZEROCOPY, ESnet, Linux 6.8



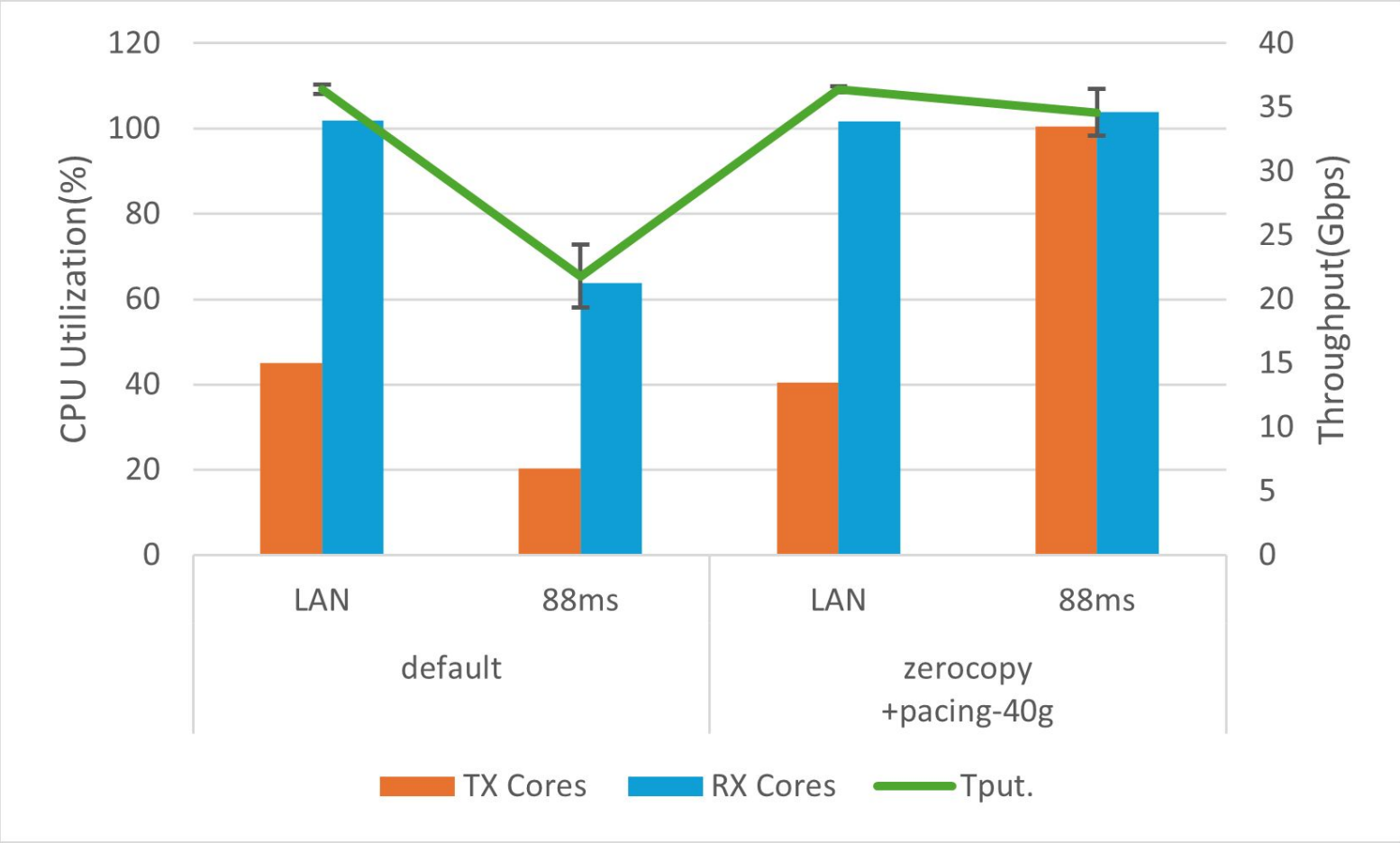
The importance of pacing (**--fq-rate**)

- 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 only when used with pacing provide a consistent throughput increase
- Provides a stable flow when used with MSG_ZEROCOPY
- We submitted a patch to **iperf3** to support pacing above 32 Gbps

Results: CPU Utilization, Intel Host, Linux 6.5



Results: CPU Utilization, AMD Host, Linux 6.8



Summary of CPU Results

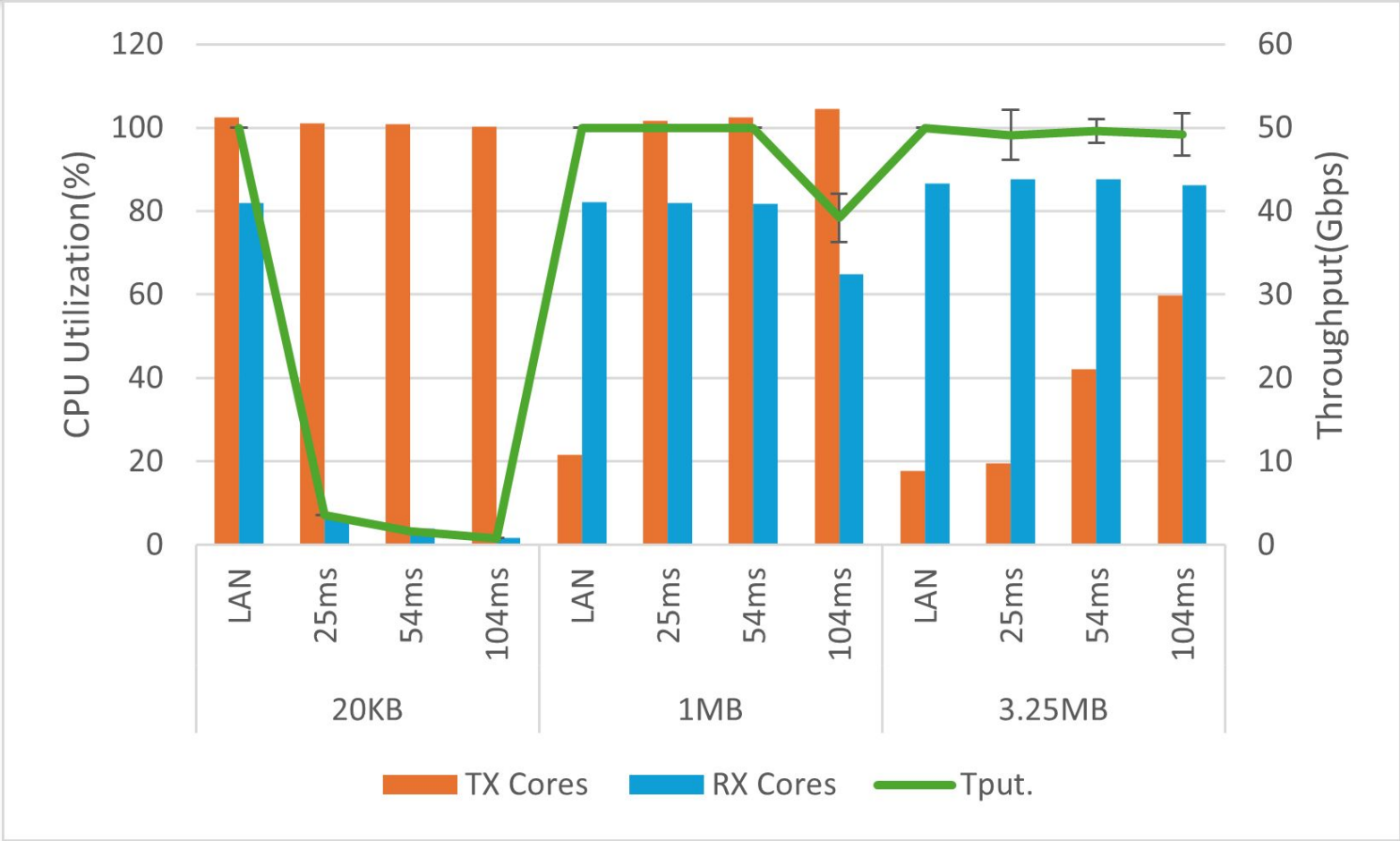
Default scenario

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

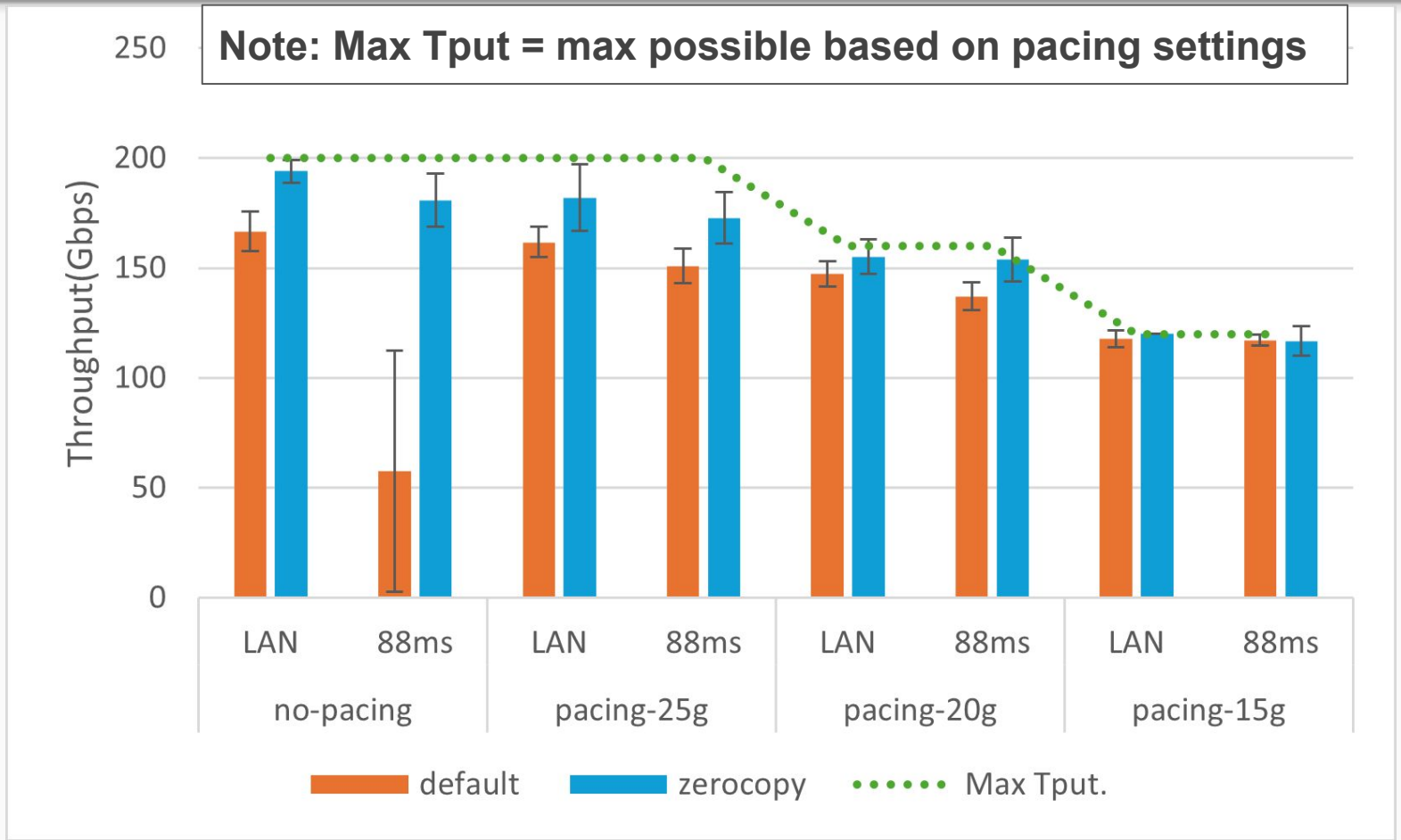
MSG_ZEROCOPY with Pacing at 50G scenario

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

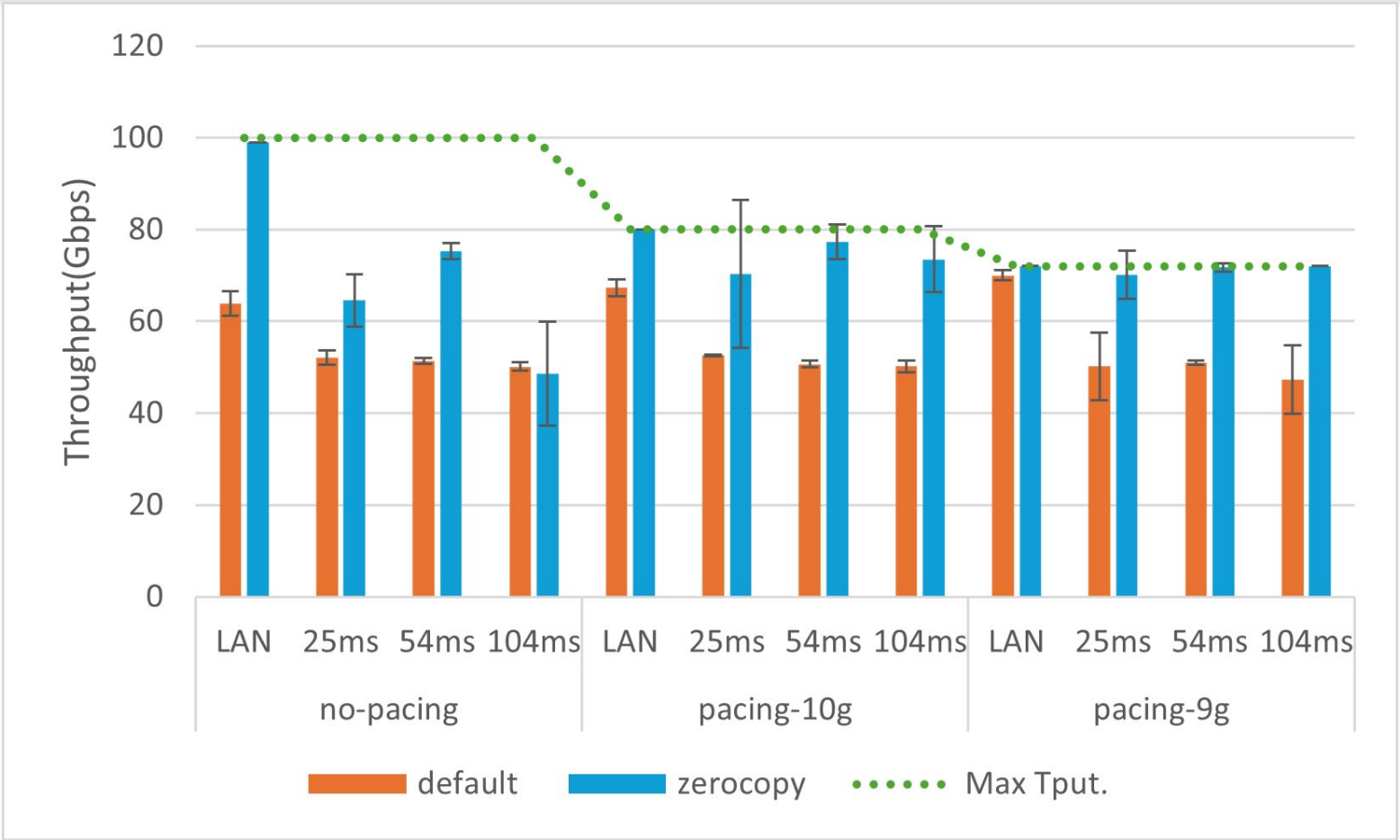
Results: CPU Utilization vs optmem_max, Intel host, Linux 6.5



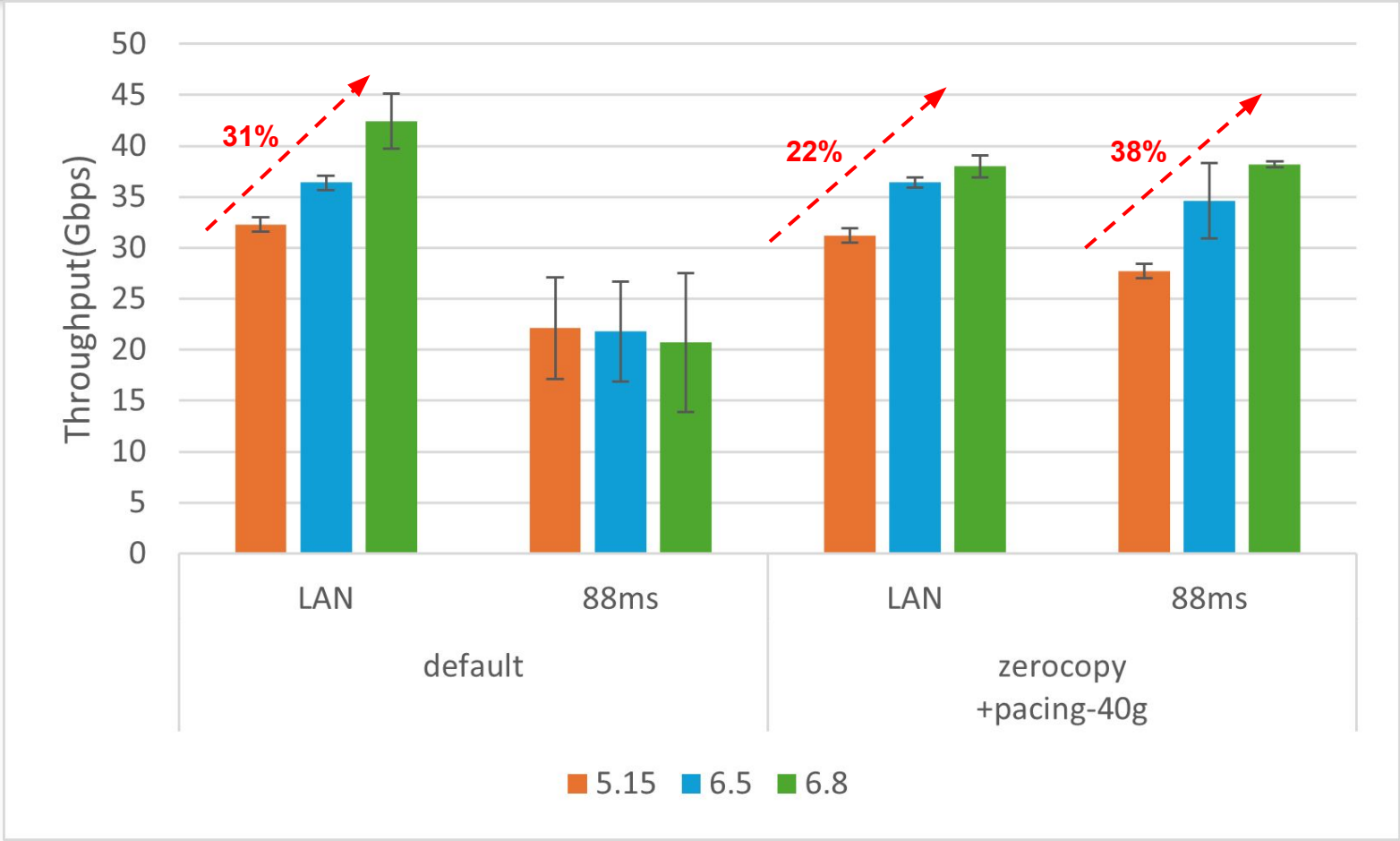
Results: Multi-Stream, ESnet



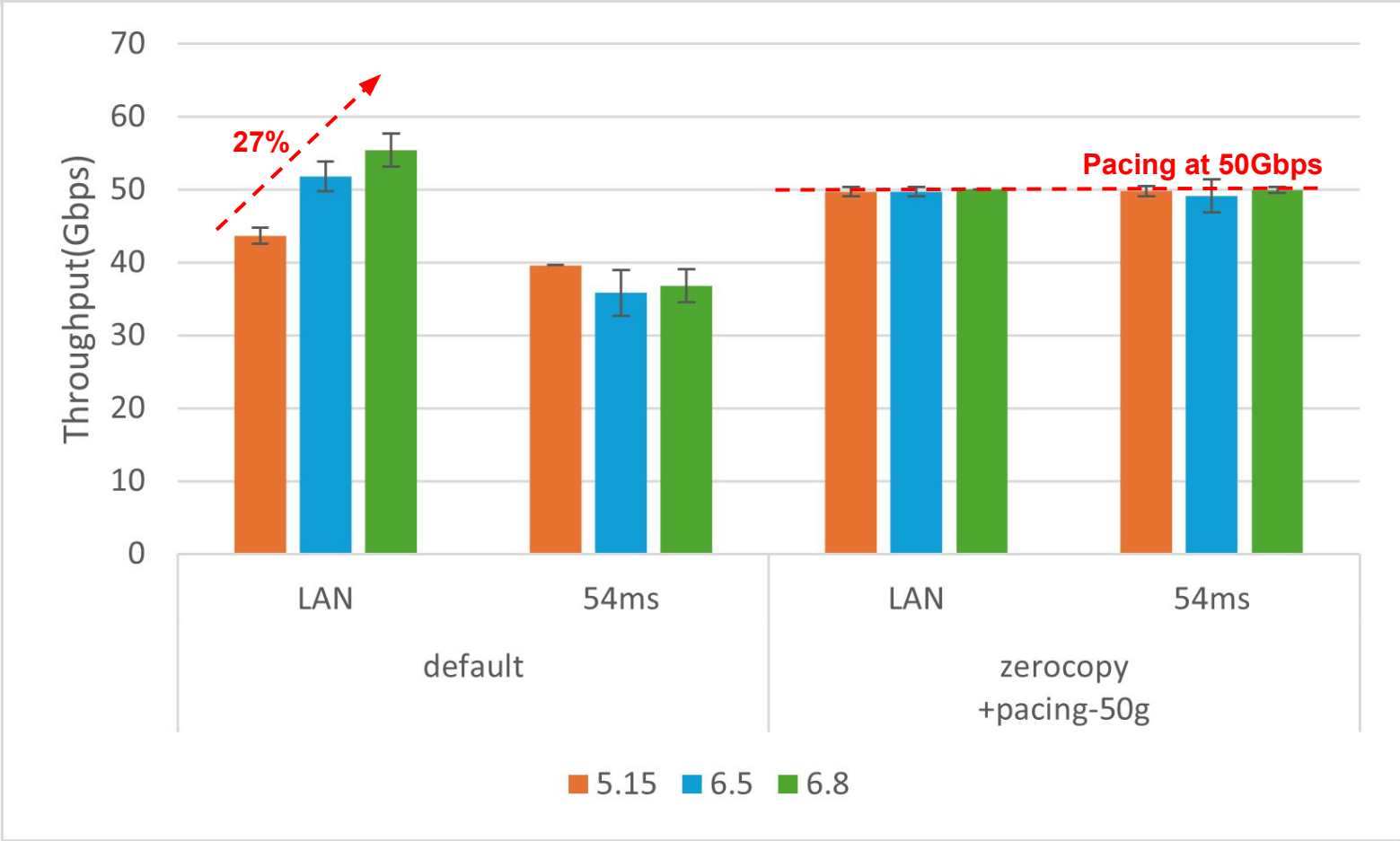
Results: Multi-Stream, AmLight



Results: kernel version comparison, ESnet



Results: kernel version comparison, AmLight



Summary of Kernel Version Results

Using the newest Linux kernel (v6.8), we see up to 38% improved performance on the WAN, and 31% better on a LAN, compared with the 5.15 kernel.

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

To install 6.8 on Ubuntu 22

- apt install linux-image-generic-hwe-22.04
- 6.11 should be available soon (for Ubuntu 24)

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

- https://elrepo.org/

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 network devices that support IEEE 802.3x flow control when possible. If not, be sure to use some level of packet pacing.

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.

Future Work

- Introduce loss and congestion using controlled background traffic
  - Revisit CUBIC vs BBR (v1 and v3)
- Reproduce tests in a 400G environment
  - Expected to reach 400G with 20x20G or 10x40G
- To achieve full BIG TCP potential, a custom Linux kernel must be built with option MAX_SKB_FRAGS=45
  - SKB45 breaks Nvidia ConnectX device driver (mlx5)
  - SKB45 + BIG TCP + MSG_ZEROCOPY: 65% improvement (preliminary result)
- HW-GRO, header/data split and RX Zerocopy (more on next slide)

Preliminary Results: Receiver HW GRO

New receiver side optimizations are available for Nvidia ConnectX-7 network cards on Linux 6.11, which include receiver side hardware accelerated GRO and header-data split.

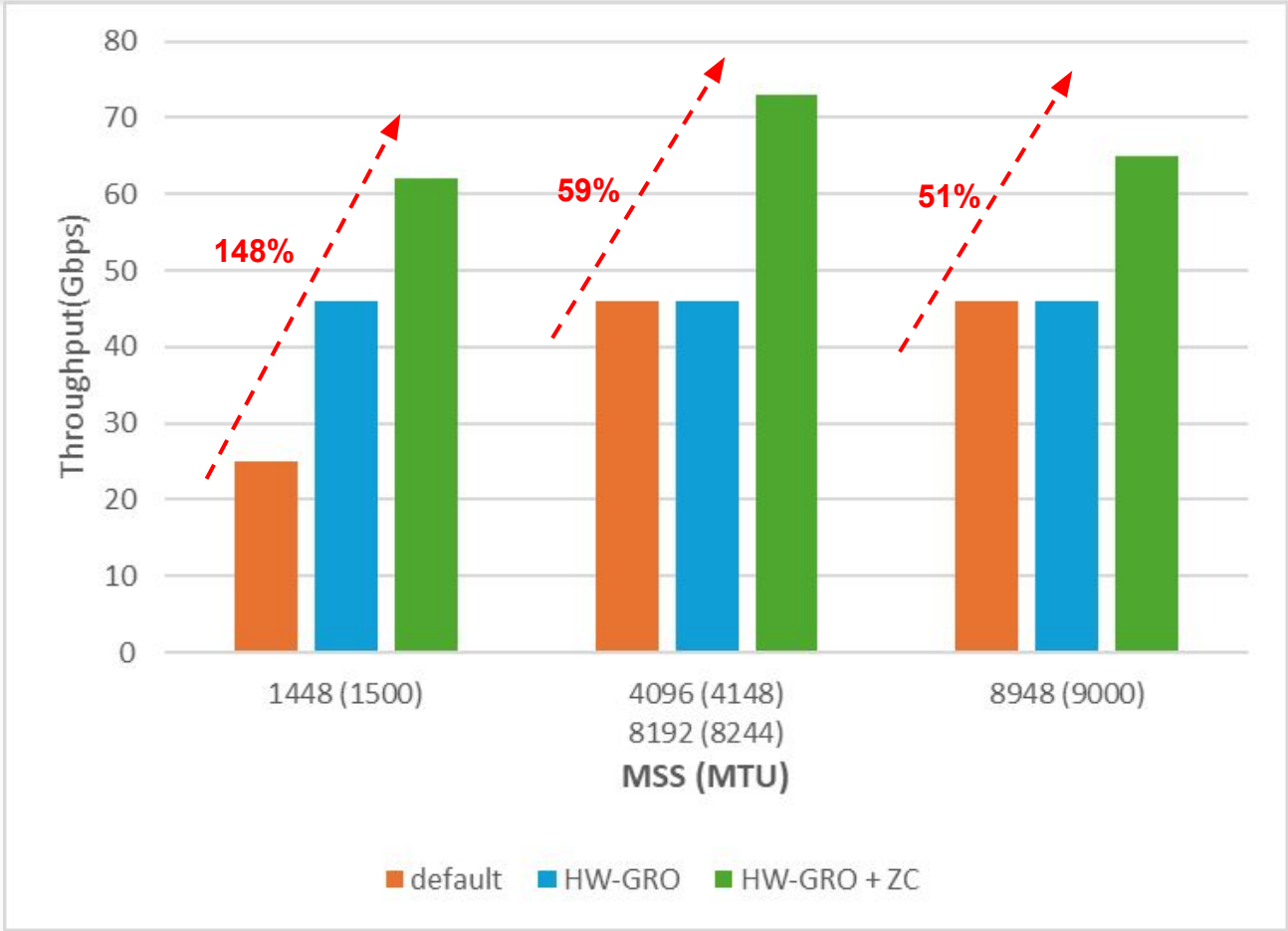
Preliminary results from the developer suggests up to 60% throughput improvement for single stream tests on 1500B MTU.

Our initial results show a 33% improvement on AMD hosts (40 Gbps vs 53 Gbps), and a 5% improvement (62 Gbps vs 65 Gbps) on Intel hosts after enabling hardware GRO on the receiver for single stream tests with a 9K MTU.

For tests with a 1500B MTU on Intel hosts we saw an impressive 160% improvement in throughput (24 Gbps vs 62 Gbps).

Note that this is not supported on older ConnectX NICs

Preliminary Results: HW GRO (LAN, Intel host at UCSD)



For More Information

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

- general 100G tuning advice

https://fasterdata.es.net/host-tuning/linux/100g-tuning/6x-kernels/

- includes link to the INDIS paper (INDIS 2024 web page is not yet up)

All raw test data is available on github:

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

Questions?

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