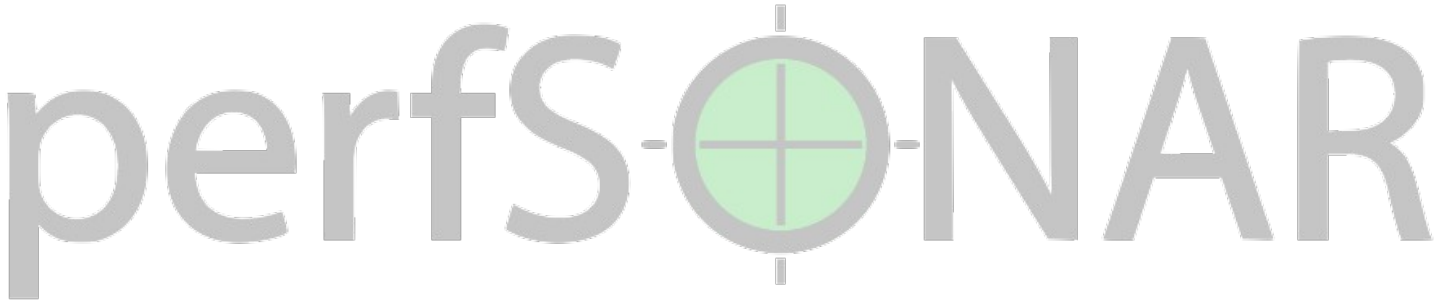


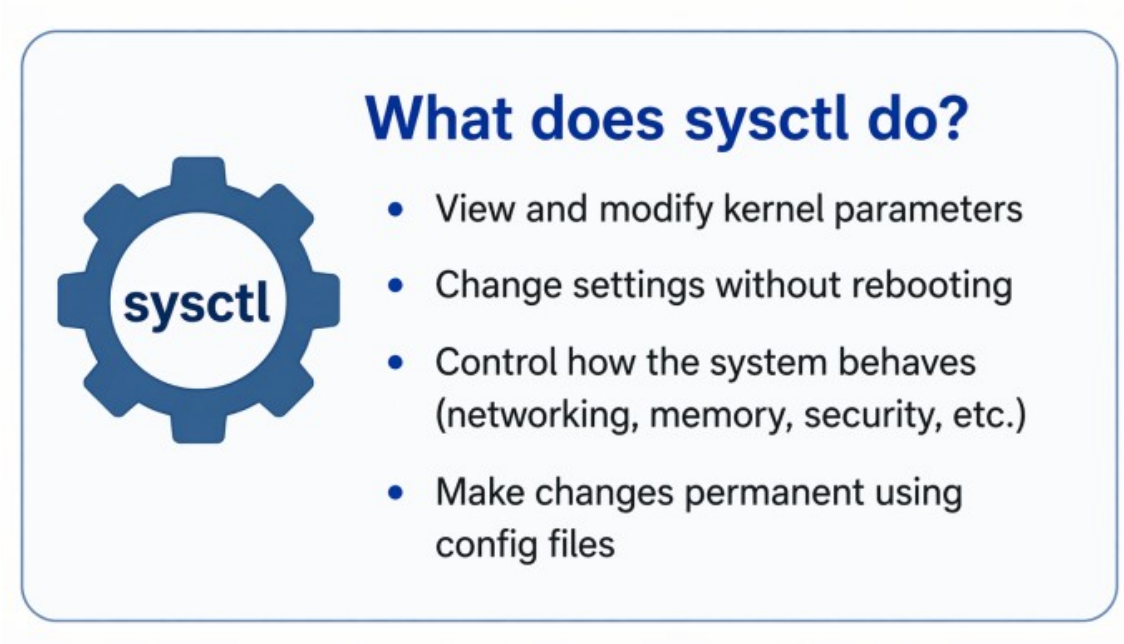
Sysctl Settings

Rezi Tchabashvili, GRENA
PerfSONAR Project Team Member

Paris, France May 2026

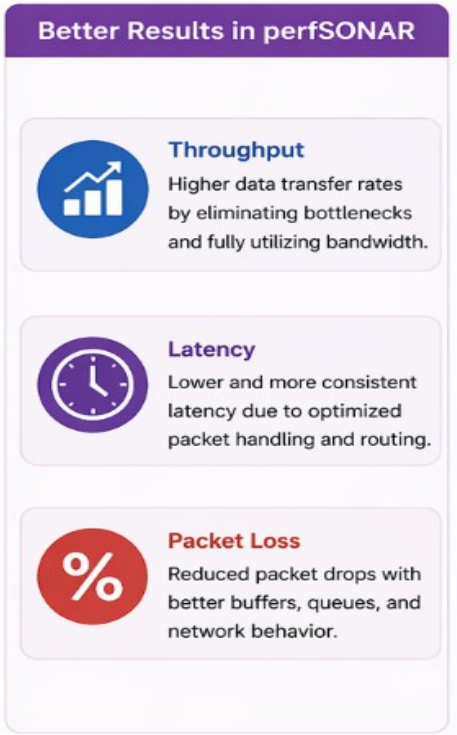
What are Sysctl Settings?

Sysctl settings are tunable kernel parameters that control system behavior such as networking, memory usage, and security, without requiring a reboot or recompiling the kernel.



perfSONAR measurements

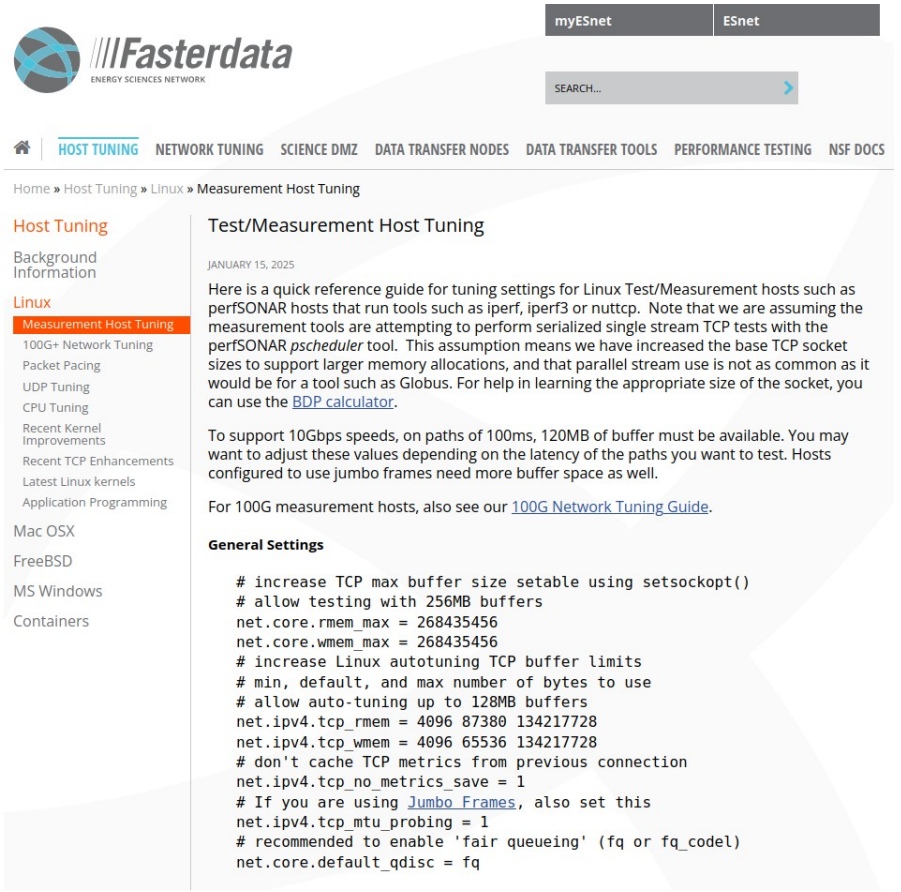
- Sysctl settings can help to optimize these measurements



ESnet's "fasterdata" knowledge base

For optimal performance, the perfSONAR Toolkit applies system tuning settings by default upon installation by using the **perfsonar-toolkit-sysctl** package. These settings are based on the ESnet "fasterdata" knowledge base for high- performance test and measurement hosts and are sufficient for most use cases. However, you can manually verify or adjust the settings in your sysctl settings configuration file.

https://fasterdata.es.net/host-tuning/linux/test-measurement-host-tuning/



Fasterdata
ENERGY SCIENCES NETWORK

myESnet | ESnet

SEARCH...

HOST TUNING | NETWORK TUNING | SCIENCE DMZ | DATA TRANSFER NODES | DATA TRANSFER TOOLS | PERFORMANCE TESTING | NSF DOCS

Home » Host Tuning » Linux » Measurement Host Tuning

Host Tuning
Background Information
Linux
Measurement Host Tuning
100G+ Network Tuning
Packet Pacing
UDP Tuning
CPU Tuning
Recent Kernel Improvements
Recent TCP Enhancements
Latest Linux kernels
Application Programming
Mac OSX
FreeBSD
MS Windows
Containers

Test/Measurement Host Tuning

JANUARY 15, 2025

Here is a quick reference guide for tuning settings for Linux Test/Measurement hosts such as perfSONAR hosts that run tools such as iperf, iperf3 or nuttcp. Note that we are assuming the measurement tools are attempting to perform serialized single stream TCP tests with the perfSONAR *pscheduler* tool. This assumption means we have increased the base TCP socket sizes to support larger memory allocations, and that parallel stream use is not as common as it would be for a tool such as Globus. For help in learning the appropriate size of the socket, you can use the BDP calculator.

To support 10Gbps speeds, on paths of 100ms, 120MB of buffer must be available. You may want to adjust these values depending on the latency of the paths you want to test. Hosts configured to use jumbo frames need more buffer space as well.

For 100G measurement hosts, also see our 100G Network Tuning Guide.

General Settings

```
# increase TCP max buffer size setable using setsockopt()
# allow testing with 256MB buffers
net.core.rmem_max = 268435456
net.core.wmem_max = 268435456
# increase Linux autotuning TCP buffer limits
# min, default, and max number of bytes to use
# allow auto-tuning up to 128MB buffers
net.ipv4.tcp_rmem = 4096 87380 134217728
net.ipv4.tcp_wmem = 4096 65536 134217728
# don't cache TCP metrics from previous connection
net.ipv4.tcp_no_metrics_save = 1
# If you are using Jumbo Frames, also set this
net.ipv4.tcp_mtu_probing = 1
# recommended to enable 'fair queueing' (fq or fq_codel)
net.core.default_qdisc = fq
```

For hosts with 10G NIC optimized for network paths up to 200ms RTT, or a 40G NIC up on paths up to 50ms RTT, use these values instead of the settings above:

```
# increase TCP max buffer size setable using setsockopt() to 512MB
net.core.rmem_max = 536870912
net.core.wmem_max = 536870912
# increase Linux autotuning TCP buffer limit to 256MB
net.ipv4.tcp_rmem = 4096 87380 268435456
net.ipv4.tcp_wmem = 4096 65536 268435456
```

For a host with a 100G NIC sending data across network paths up to 200ms RTT, you'll probably want to set the buffers to use the maximum value of 2GB.

```
# Allow buffers up to 2GB
net.core.rmem_max=2147483647
net.core.wmem_max=2147483647
# increase Linux autotuning TCP buffer limit to 1GB
net.ipv4.tcp_rmem=4096 65536 1073741824
net.ipv4.tcp_wmem=4096 65536 1073741824
```

To apply your changes /etc/sysctl.conf, run:

```
sysctl -p
```

What is configure_sysctl Script

configure_sysctl is a helper script used by perfSONAR to automatically generate and apply optimized sysctl settings.

What it actually does:

1. Checks network interfaces (speed, MTU)
2. Checks our Operating System
3. Applies recommended tuning (from Fasterdata best practices)

Installation: apt install perfsonar-toolkit-sysctl

perfSONAR

1 DEFAULT SYSCTL SETTINGS

```bash
#!/bin/bash

# Only run on new installation
if [[ "$1" != "new" ]]; then
    exit 0
fi

# Ensure directory exists and config file is created
SYSCTL_CONF="/etc/sysctl.d/perfsonar-toolkit-sysctl.conf"
mkdir -p /etc/sysctl.d
> "$SYSCTL_CONF" # create/truncate the file

# Default sysctl settings
declare -A SYSCTL_SETTINGS=(
    [net.core.rmem_max]=67108864
    [net.core.wmem_max]=67108864
    [net.ipv4.tcp_rmem]="4096 87380 33554432"
    [net.ipv4.tcp_wmem]="4096 65586 33554432"
    [net.ipv4.tcp_no_metrics_save]=1
    [net.ipv4.tcp_congestion_control]=htcp
    [net.ipv4.conf.all.arp_ignore]=1
    [net.ipv4.conf.all.arp_announce]=2
    [net.ipv4.conf.default.arp_filter]=1
    [net.ipv4.conf.all.arp_filter]=1
)

MAX_SPEED=0
MAX_MTU=0
```

2 INTERFACE DETECTION, SPEED & MTU

```bash
# Check interfaces
for IFACE in /sys/class/net/*; do
    IFACE_NAME=$(basename "$IFACE")
    [[ "$IFACE_NAME" == "lo" ]] && continue

    # Speed
    if [[ -f "$IFACE/speed" ]]; then
        SPEED=$(cat "$IFACE/speed" 2>/dev/null)
        [[ "$SPEED" =~ ^[0-9]+$ ]] && SPEED=$((SPEED * 1000000))
        [[ "$SPEED" =~ ^[0-9]+$ ]] && (( SPEED > MAX_SPEED )) && MAX_SPEED=$SPEED
    fi

    # MTU
    if [[ -f "$IFACE/mtu" ]]; then
        MTU=$(cat "$IFACE/mtu")
        (( MTU > MAX_MTU )) && MAX_MTU=$MTU
    fi
done
```

3 TUNE BASED ON SPEED & MTU

```bash
# Tune based on speed
if (( MAX_SPEED >= 40000000000 )); then
    SYSCTL_SETTINGS[net.core.rmem_max]=536870912
    SYSCTL_SETTINGS[net.core.wmem_max]=536870912
    SYSCTL_SETTINGS[net.ipv4.tcp_rmem]="4096 87380 268435456"
    SYSCTL_SETTINGS[net.ipv4.tcp_wmem]="4096 65536 268435456"
elif (( MAX_SPEED >= 10000000000 )); then
    SYSCTL_SETTINGS[net.core.rmem_max]=268435456
    SYSCTL_SETTINGS[net.core.wmem_max]=268435456
    SYSCTL_SETTINGS[net.ipv4.tcp_rmem]="4096 87380 134217728"
    SYSCTL_SETTINGS[net.ipv4.tcp_wmem]="4096 65536 134217728"
fi

# MTU tuning
(( MAX_MTU > 8000 )) && SYSCTL_SETTINGS[net.ipv4.tcp_mtu_probing]=1

# Save results
```

4 OS-SPECIFIC SETTINGS & WRITE TO FILE

```bash
# OS-specific settings
if [[ -f /etc/os-release ]]; then
   ./etc/os-release
   case "$ID" in
      centos|rhel)
         [[ "$VERSION_ID" =~ ^7 ]] && SYSCTL_SETTINGS[net.core.default_qdisc]=fq
         [[ "$VERSION_ID" =~ ^6 && MAX_SPEED -ge 10000000000 ]] && N
                        SYSCTL_SETTINGS[net.core.netdev_max_backlog]=250000
         ;;             ;
    debian)
         [[ "$VERSION_ID" =~ ^8 ]] && SYSCTL_SETTINGS[net.core.default_qdisc]=fq
   esac  ;;
fi

# Write sysctl settings to file
{
   echo "########################################"
   echo "# Default perfSONAR sysctl settings"
   echo "########################################"
   for KEY in "${!SYSCTL_SETTINGS[@]}" | sort; do
      echo"$KEY = ${SYSCTL_SETTINGS[$KEY]}"
   done
} > "$SYSCTL_CONF"
```

ESnet ENERGY SCIENCES NETWORK · GÉANT · INDIANA UNIVERSITY · INTERNET2

Default Sysctl Settings

```
#####################################
# Default perfSONAR sysctl settings
#####################################
net.core.rmem_max = 67108864
net.core.wmem_max = 67108864
net.ipv4.conf.all.arp_announce = 2
net.ipv4.conf.all.arp_filter = 1
net.ipv4.conf.all.arp_ignore = 1
net.ipv4.conf.default.arp_filter = 1
net.ipv4.tcp_congestion_control = htcp
net.ipv4.tcp_no_metrics_save = 1
net.ipv4.tcp_rmem = 4096 87380 33554432
net.ipv4.tcp_wmem = 4096 65536 33554432
```

MAX_SPEED < 10 Gbps
MAX_MTU <= 8000

If script sees:

- only loopback (lo)
- Network speed returns unknown (-1)

perfSONAR

Memory buffers

net.core.rmem_max - **Maximum receive socket buffer size in bytes** (maximum amount of incoming data your system is willing to hold in memory before the application reads it)

net.core.wmem_max - **Maximum send socket buffer size in bytes** (Maximum amount of outgoing data your system can hold before sending it over the network.)

TCP buffers

net.ipv4.tcp_rmem - **Defines TCP receive buffer behavior using 3 values** (Controls how TCP automatically adjusts how much incoming data it stores.)

- This is a triple value setting:

  net.ipv4.tcp_rmem = MIN DEFAULT MAX
  net.ipv4.tcp_rmem = 4096 87380 33554432

net.ipv4.tcp_wmem - **Defines TCP send buffer behavior** (controls how TCP decides how much data it can prepare before sending it out)

Traffic control

net.ipv4.tcp_congestion_control - which algorithm TCP uses to control network congestion (Controls speed and behavior when network is busy)

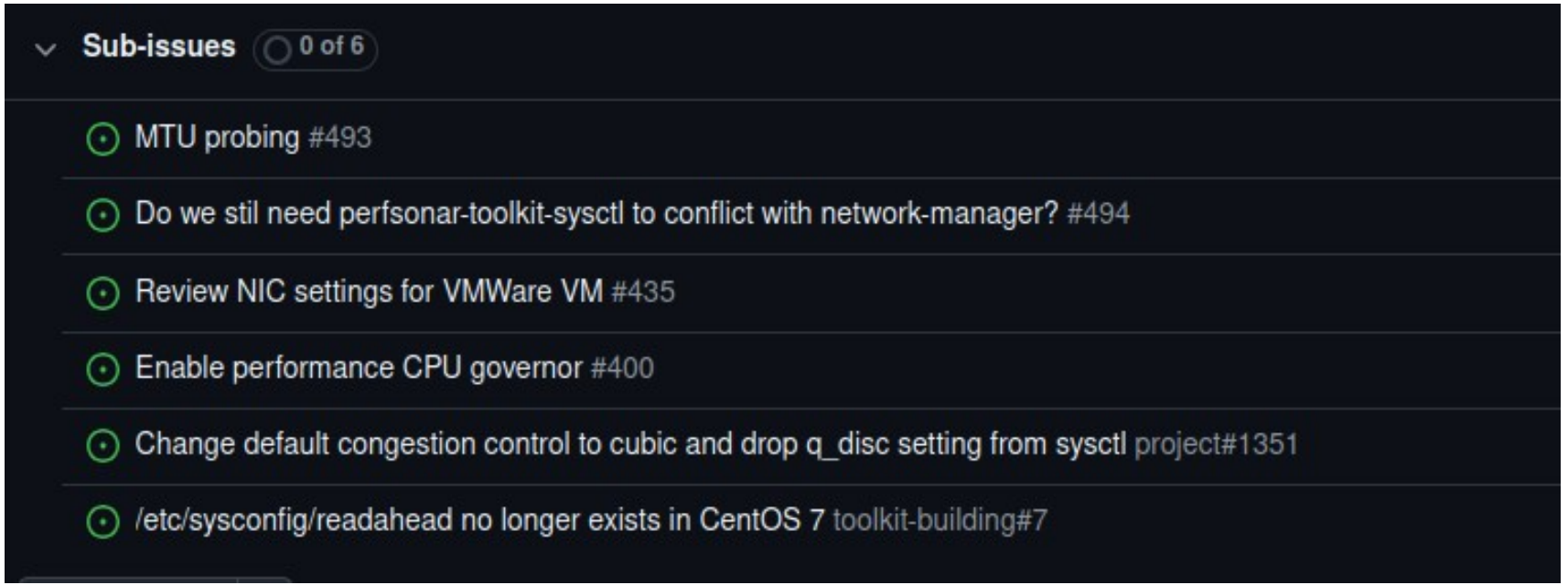
net.core.default_qdisc - It controls how packets are queued and scheduled before being sent out of a network interface

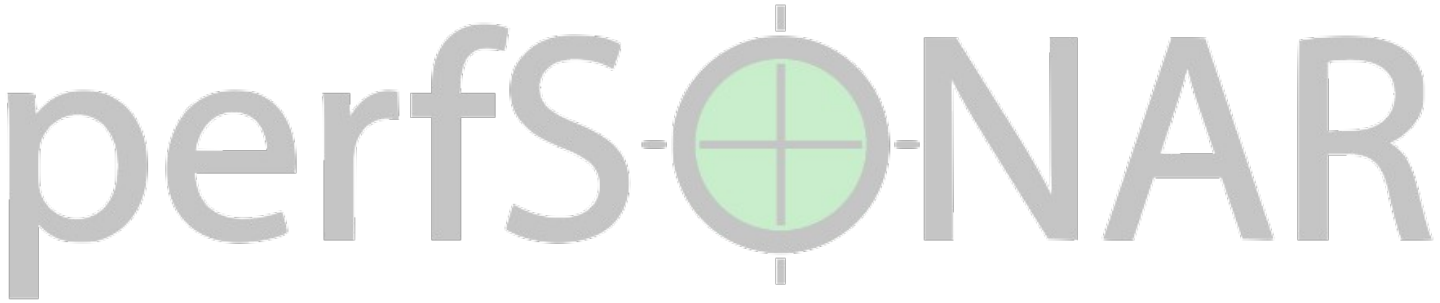
ESnet ENERGY SCIENCES NETWORK GÉANT INDIANA UNIVERSITY INTERNET2

10 Gbps Settings VS 40 Gbps Settings

10 Gbps Settings	40 Gbps Settings
net.core.rmem_max = 268435456	net.core.rmem_max = **536870912**
net.core.wmem_max = 268435456	net.core.wmem_max = **536870912**
net.ipv4.tcp_rmem = 4096 87380 134217728	net.ipv4.tcp_rmem = 4096 87380 **268435456**
net.ipv4.tcp_wmem = 4096 65536 134217728	net.ipv4.tcp_wmem = 4096 65536 **268435456**
net.ipv4.tcp_congestion_control = **htcp**	net.ipv4.tcp_congestion_control = **bbr**
net.ipv4.tcp_no_metrics_save = 1	net.ipv4.tcp_no_metrics_save = 1
net.ipv4.tcp_mtu_probing = 1	net.ipv4.tcp_mtu_probing = 1
net.core.default_qdisc = **fq**	net.core.default_qdisc = **fq**

What's Next?


Sub-issues
0 of 6
MTU probing #493
Do we stil need perfsonar-toolkit-sysctl to conflict with network-manager? #494
Review NIC settings for VMWare VM #435
Enable performance CPU governor #400
Change default congestion control to cubic and drop q_disc setting from sysctl project#1351
/etc/sysconfig/readahead no longer exists in CentOS 7 toolkit-building#7



Thank You

Home Page: https://www.perfsonar.net/
PerfSONAR mailing list: perfsonar@lists.geant.org