LOGZILLA DOCUMENTATION

Performance Tuning: Networking

Tune UDP syslog ingest performance on the LogZilla host, including netcat and loggen benchmarking and kernel socket buffer adjustments

Administration · Generated June 12, 2026 · logzilla.ai/docs/administration/network-tuning

Networking performance (UDP)

Steps to test UDP ingest performance and adjust kernel parameters. Commands are examples; ports and values must be verified for the environment.

Testing UDP performance

- Use a non-conflicting high port for testing (for example, 5514) to avoid disrupting the running syslog receiver and to avoid privileged-port requirements.
- Start a UDP listener using netcat:

  ```
  netcat -u -p 5514 -l > /tmp/logs
  ```

  Alternatively (on systems where `nc` is used):

  ```
  nc -u -l 5514 > /tmp/logs
  ```

- In another terminal, generate messages using loggen (part of syslog-ng):

  ```
  ./loggen -r 10000 -D -I 10 127.0.0.1 5514
  ```

- Example output from loggen:

  ```
  average rate = 10877.62 msg/sec, count=108783, time=10.006, msg size=256,
  bandwidth=2719.40 kB/sec
  ```

- Verify line count matches (or closely matches) the loggen reported count:

  ```
  wc -l /tmp/logs
  ```

  Example:

  ```
  108783 /tmp/logs
  ```

- Check UDP statistics:

```
netstat -su
```

Modern equivalent using `ss`:

```
ss -u -s    # UDP summary
ss -u -a    # all UDP sockets
```

Key fields:

- `packets received`: total packets since last boot.
- `packets to unknown port`: packets sent to a closed port (for example, service stopped while senders keep sending).
- `packet receive errors`: errors while receiving/processing; a single packet can generate multiple errors.

Tuning UDP buffers

If `netstat -su` shows errors, increase UDP buffer sizes incrementally and re-test after each change.

- Increase receive buffer max (example: 32 MB):

```
sysctl -w net.core.rmem_max=33554432
```

The default on many Linux distributions is about 122 KB (`net.core.rmem_default = 124928`).

- Additional tuning options (examples):

```
# UDP memory (in pages), min rmem
# NOTE: values are environment-dependent; increase gradually and test
echo 'net.ipv4.udp_mem = 192576 256768 385152' >> /etc/sysctl.conf
echo 'net.ipv4.udp_rmem_min = 4096' >> /etc/sysctl.conf

# Alternative runtime example for udp_mem
sysctl -w net.ipv4.udp_mem='262144 327680 393216'
```

Optional transmit buffer tuning (device-dependent):

```
sysctl -w net.core.rmem_default=212992
sysctl -w net.core.wmem_default=212992
sysctl -w net.core.wmem_max=33554432
```

net.ipv4.udp_mem values are page-based. With `PAGE_SIZE=4096`, the maximum above equals `385152 * 4096 = 1,577,582,592` bytes.

- Increase the incoming packet queue size:

```
sysctl -w net.core.netdev_max_backlog=2000
```

Persisting settings

Settings changed with `sysctl -w` are not persistent. To make the changes permanent, append to `/etc/sysctl.conf` and reload:

```
# Example: persist rmem_max at 32 MB
echo 'net.core.rmem_max=33554432' >> /etc/sysctl.conf

# Apply changes
sysctl -p
```

Re-run tests after each change to validate impact and adjust values as needed.

Alternatively, create a drop-in file and reload all settings:

```
cat >/etc/sysctl.d/99-logzilla-tuning.conf <<'EOF'
net.core.rmem_max=33554432
EOF

sysctl --system
```

Advanced NIC tuning (optional)

These options can improve performance on multi-queue NICs in high-throughput environments. Hardware and kernel support vary; test changes incrementally in staging before applying to production.

Multi-queue visibility

```
ls -d /sys/class/net/eth0/queues/rx-* 2>/dev/null || true
ls -d /sys/class/net/eth0/queues/tx-* 2>/dev/null || true
```

Receive Packet Steering (RPS) and Receive Flow Steering (RFS)

View and set global flow entries (example values):

```
cat /proc/sys/net/core/rps_sock_flow_entries
echo 32768 > /proc/sys/net/core/rps_sock_flow_entries
```

Distribute per-queue flow counts and CPUs (example; adjust mask to available CPUs, for example `f` = CPUs 0-3):

```
for q in /sys/class/net/eth0/queues/rx-*/rps_flow_cnt; do echo 4096 > "$q"; done
for q in /sys/class/net/eth0/queues/rx-*/rps_cpus; do echo f > "$q"; done
```

Transmit Packet Steering (XPS)

Pin transmit queues to CPUs (example; adjust CPU mask):

```
for q in /sys/class/net/eth0/queues/tx-*/xps_cpus; do echo f > "$q"; done
```

Note: RPS/RFS/XPS settings reset on reboot. Persist via udev rules or a systemd unit executed at boot. Not all systems support `xps_cpus`, in which case it will indicate `write error: No such file or directory`.

NIC ring buffers

Inspect and (if supported) increase ring sizes cautiously:

```
ethtool -g eth0          # query
ethtool -G eth0 rx 4096 tx 4096   # set (hardware/driver dependent)
```

NIC offload features

View current offload settings:

```
ethtool -k eth0
```

Only disable specific offloads during targeted troubleshooting. Defaults are generally optimal.

Interrupt mapping visibility

```
grep -E 'eth0' /proc/interrupts
```

Most distributions run `irqbalance` to spread IRQ load automatically. Manual pinning is advanced and environment-specific.