Linux® Network Tuning Guide for AMD EPYC™ 7002 Series Processor Based Servers

Application Note

Publication # **56739** Revision: **0.20**
Issue Date: **October 2019**

Advanced Micro Devices

© 2019 Advanced Micro Devices, Inc. All rights reserved.

The information contained herein is for informational purposes only, and is subject to change without notice. While every precaution has been taken in the preparation of this document, it may contain technical inaccuracies, omissions and typographical errors, and AMD is under no obligation to update or otherwise correct this information. Advanced Micro Devices, Inc. makes no representations or warranties with respect to the accuracy or completeness of the contents of this document, and assumes no liability of any kind, including the implied warranties of noninfringement, merchantability or fitness for particular purposes, with respect to the operation or use of AMD hardware, software or other products described herein. No license, including implied or arising by estoppel, to any intellectual property rights is granted by this document. Terms and limitations applicable to the purchase or use of AMD's products are as set forth in a signed agreement between the parties or in AMD's Standard Terms and Conditions of Sale.

Trademarks

AMD, the AMD Arrow logo, AMD EPYC, and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

Linux is a registered trademark of Linus Torvalds.

Contents

List of Tables

Revision History

Date	Revision	Description
October 2019	0.20	Initial release.

Chapter 1 Introduction

There is no single golden rule for tuning a network interface card (NIC) for all conditions. Different adapters have different parameters that can be changed. Operating systems also have settings that can be modified to help with overall network performance. Depending on the exact hardware topology, one may have to make different adjustments to network tuning to optimize for a specific workload. With Ethernet speeds going higher, up to 200 Gb, and the number of ports being installed in servers growing, these tuning guidelines become even more important to achieve the best performance possible.

This guide does not provide exact settings for modifying every scenario. Rather, it recommends steps to check and modify when (or if) they prove to be beneficial for a given scenario. In this guide, the steps are focused on TCP/IP network performance. Appendix A provides tables of recommended tuning parameters as well as results measured in AMD labs.

One general rule of thumb for all performance testing is to ensure your memory subsystem is properly configured. All I/O uses data transfers into or out of memory, so the I/O bandwidth can never exceed the capabilities of the memory subsystem. For the maximum memory bandwidth on modern CPUs, you must populate at least one DIMM in every DDR channel. For AMD EPYC™ 7002 Series Processor-based servers, there are eight DDR4 memory channels on each CPU socket. So, for a single-socket platform, you must populate all eight memory channels. Likewise, on a dual-socket platform, you must populate 16 memory channels.

In addition to this document, AMD recommends consulting any tuning guide available from your NIC vendor. Vendors will sometimes enable specific tuning options for their devices with parameters that can be modified to further improve performance. One example could be the ability to enable or disable interrupt coalescing. Another could allow users to change the number of interrupts the device uses (this variable will be important in section 3.2).

Chapter 2 Ensure Transmit and Receive Ring Sizes are Correct

One of the first places to start tuning the TCP/IP stack for adapters under Linux® is to ensure you are using the maximum number of both transmit (TX) and receive (RX) ring buffers. Some drivers will automatically set the maximum size for the silicon on the NIC, but to be sure this occurs, there are some simple commands that can be executed.

2.1 Use ethtool to Check TX and RX Ring Sizes

The NIC ring values represent the number of buffers that a NIC uses to DMA data into system memory. With more buffers available, more work can be queued to be processed during a single interrupt. Using the **ethtool** utility, you can find both the current ring size and the maximum allowed by the NIC.

Here's an example:

```
root@testsystem:~# ethtool -g enp33s0f0
Ring parameters for enp33s0f0:
Pre-set maximums:
RX:             8192
RX Mini:        0
RX Jumbo:       0
TX:             8192
Current hardware settings:
RX:             512
RX Mini:        0
RX Jumbo:       0
TX:             512
```

You will notice that for this particular adapter, although the silicon and driver allow for a maximum ring size of 8192, it is currently set for 512. This is 1/16th of the maximum allowed. You can use **ethtool** to modify the current setting to have it match the maximum. This is one key component for improving network performance.

```
root@testsystem:~# ethtool -G enp33s0f0 rx 8192 tx 8192
```

After issuing **ethtool** with a "**-G**" (to set the ring size), you should reissue the original "**-g**" command to verify that the change was accepted. These setting changes will not be persistent upon reboot. The reader must understand how their Linux distribution needs to be updated to make these changes persistent.

Chapter 3 Optimize Interrupt Handling for Your NIC

3.1 Change the Number of Interrupt Queues Used by Your NIC

One key item to help network performance is to not have more interrupt queues than you have CPU cores per NUMA node. With multiple queues assigned to a single core, you risk thrashing the interrupt handler swapping between queues. A more efficient manner would be to have a single queue per core to eliminate that thrashing.

There are three types of interrupt queues: receive (RX), transmit (TX), and combined. Combined used a single queue to handle both RX and TX interrupts. Some vendors still have separate RX and TX queues while others implement only combined queues.

We can use the output from lscpu to determine the number of cores per die.

```
root@testsystem:~# lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                256
On-line CPU(s) list:   0-255
Thread(s) per core:    2
Core(s) per socket:    64
Socket(s):             2
NUMA node(s):          32
Vendor ID:             AuthenticAMD
CPU family:            23
Model:                 49
Model name:            AMD EPYC 7742 64-Core Processor
Stepping:              0
CPU MHz:               1497.318
CPU max MHz:           2250.0000
CPU min MHz:           1500.0000
BogoMIPS:              4491.94
Virtualization:        AMD-V
L1d cache:             32K
L1i cache:             32K
L2 cache:              512K
L3 cache:              16384K
NUMA node0 CPU(s):     0-3,128-131
NUMA node1 CPU(s):     4-7,132-135
NUMA node2 CPU(s):     8-11,136-139
NUMA node3 CPU(s):     12-15,140-143
...
Flags:                   ...
```

The output above shows the physical core numbers followed by the logical core numbers. For instance, the highlighted row for NUMA node 2 has physical cores 8–11 and logical cores 136–139. So, in our case we have four physical cores per NUMA node. Therefore, we want eight total interrupt queues. If you have a NIC that combines the RX and TX interrupts, then the following command would be used:

```
root@testsystem:~# ethtool -L enp33s0f0 combined 8
```

Chapter 4 Other Tunings to Optimize for Throughput

4.1 Large Receive Offload (LRO)

Many NIC vendors have tuning guides to help end-users optimize for specific use cases. Those use cases usually involve optimizing for the highest throughput possible or the lowest latency possible, but rarely can achieve both at the same time. One common way to improve network throughput performance is to enable Large Receive Offload (LRO). To enable LRO on an adapter, consult the proper documentation from your NIC provider. Some providers will use standard commands available via **ethtool**, while others might have extra parameters that are needed to fully enable LRO. Please keep in mind that while enabling LRO will help improve your throughput, it could have a negative effect on network latency. So, be sure to tune as appropriate for your workload.

4.2 Transmit Queue Length

The transmit queue length is used by the networking stack to set the number of packets that can be queued before being transmitted. As adapter speeds have increased, sometimes increasing the size of this queue is necessary (the default is 1,000). You can change the transmit queue length using **ifconfig**. Here is an example of increasing the transmit queue length to 20,000:

```
root@testsystem:~# ifconfig enp33s0f0 txqueuelen 20000
```

4.3 x2APIC

With the introduction of the EPYC 7002 Series of processors, AMD has implemented an x2APIC controller. This has two benefits:

- Allows operating systems to work with the 256 CPU threads now available on AMD platforms
- Provides improved performance over the legacy APIC

AMD recommends, but not requires, that you enable the x2APIC mode in BIOS even for lower core count parts. (The AMD BIOS will enable x2APIC automatically when two 64-core processors are installed.)

4.4 Infinity Fabric P-states

Like processor cores, the EPYC 7002 Series processor Infinity Fabric has the ability to go into lower power states (P-states) when being lightly used. This saves power consumption of the overall socket, or allows power to be diverted to other portions of the processor. By default, to enable the best performance per watt, P-states are enabled in the processor. To disable the switching of P-states and force P0 all the time, you must go into the system BIOS and set APBDIS to 1. Higher bandwidth adapters may require the forcing of the P0 state to maintain the highest bandwidth.

4.5 Preferred I/O and PCIe Relaxed Ordering

From PCIe 1.0, there have been both strict and relaxed methods of ordering PCIe packets within a system. The ordering of the packets help maintain data coherency and consistency as data flows between endpoints and main memory. Relaxed ordering is enabled to allow packets to be retired out of order when possible. This maintains data consistency and improves performance in high-bandwidth cases. AMD introduced Preferred I/O as a new feature in the EPYC 7002 Series processors also to help with ordering of PCIe packets. Enabling this for high-bandwidth adapters can result in improved performance in some cases.

4.6 Last Level Cache (LLC) as NUMA

The EPYC line of processors has multiple Last Level Caches (LLCs), or L3 caches. While operating systems can handle the multiple LLCs and schedule jobs accordingly, AMD has created a BIOS option to enable the description of a single NUMA domain per LLC. This can help the operating system schedulers maintain locality to the LLC without causing unnecessary cache-to-cache transactions. For more details on LLC as NUMA.

4.7 Maximum Transmission Unit (MTU)

Maximum Transmission Units, or MTU, define the size of the data packet being transferred over the fabric of a network. The limit for ethernet set by the IEEE 802.3 standard is 1500. Jumbo frames allow for an MTU size of up to 9000. Changing the MTU size to 9000 is easily done using **ifconfig**, but before doing so, you should ensure your full network uses jumbo frames. If the network does not support the payload size that you set with **ifconfig**, then your adapter will be limited to the smaller payload size. You can change the MTU size using the following example:

```
root@testsystem:~# ifconfig enp33s0f0 mtu 9000
```

4.8 IOMMU Settings

The Linux kernel constantly updates. The official mainline releases from The Linux Foundation are found on *http://kernel.org* (the mainline release is maintained by Linus Torvalds and typically contains the latest features). The common enterprise level Linux distributions, however, rarely use a mainline kernel.

AMD has contributed code into the Linux kernel for years. Most recently, the focus has been enabling the Zen and Zen2 architecture contained in AMD EPYC processors. One area of code contribution has been focused on optimizing the input-output memory management unit (IOMMU) code for AMD EPYC processors. These IOMMU patches can have a direct impact on TCP/IP performance, even in a bare metal (non-virtualized) environment. Sometimes, disabling the IOMMU or setting it to pass through mode for the highest bandwidth adapters may be necessary, such as with 200 Gb Ethernet adapters. To set the IOMMU to pass-through mode, the following kernel parameter must be passed in during boot time:

```
iommu=pt
```

Appendix A Networking Tuning Recommendations and Results

Table 1 Network Tuning Recommendations provides the recommended values for each of the options described in the document. Not all adapters require modification from default BIOS, OS, or adapter options.

Table 1 Network Tuning Recommendations

	Dual Port 25 Gb Ethernet	**Single Port 100 Gb Ethernet**	**Single Port EDR Infiniband**	**Dual Port 100 Gb Ethernet**	**Dual Port EDR Infiniband**
BIOS Options					
Local APIC mode	default	x2apic	x2apic	x2apic	x2apic
Determinism mode	default	performance	performance	performance	performance
APBDIS	default	1	1	1	1
Preferred I/O	default	enabled	enabled	enabled	enabled
LLC as NUMA	default	enabled	enabled	enabled	enabled
Adapter Options					
Relaxed Ordering	default	enabled	enabled	enabled	enabled
OS Options					
Ring Buffers	default	maximum	maximum	maximum	maximum
Large Receive Offload (lro)	default	enabled	enabled	enabled	enabled
Transmit Queue Length (txqueuelen)	default	20,000	20,000	20,000	20,000
Interrupts	default	combined 16	combined 16	combined 16	combined 16
MTU Default 1500	default	default	default	default	default

AMD has tested several adapters at multiple speeds using the recommendations from Table 1 Network Tuning Recommendations, and those results are below in Table 2 Network Testing Results.

Table 2 Network Testing Results

Tested Adapter	Fabric Type	Port Speed	Total Ports	Aggregate Bidirectional Bandwidth
Mellanox ConnectX-4	Ethernet	25 Gb	2	95.5 Gbps
Intel X710	Ethernet	25 Gb	2	94.8 Gbps
Mellanox ConnectX-5	Ethernet	100 Gb	1	188.5 Gbps
Broadcom Stratus	Ethernet	100 Gb	1	187.8 Gbps
Mellanox ConnectX-5	Infiniband	EDR	1	195 Gbps
Mellanox ConnectX-5	Ethernet	100 Gb	2	302.2 Gbps
Mellanox ConnectX-5 IB Write	Infiniband	EDR	2	391.7 Gbps
Mellanox ConnectX-5 IB Write	Infiniband	EDR	2	384.95 Gbps