Myricom: Extreme Performance Ethernet

Search...

- Hardware
  - Myri-10G Network Adapters
  - Legacy Myri-10G HPC Switches
  - Legacy Myrinet-2000 Network Adapters
  - Legacy Myrinet-2000 Switches
- Software
  - Myri10GE
  - DBL
  - Sniffer10G
  - MVA
  - Myri-10G Network Adapter Tool Kit
  - Legacy VideoPump
  - Legacy MX
  - Legacy MPI Implementations with MX
  - Legacy Fabric Management System (FMS)
- General Information
  - Software Licensing
  - RoHS Directive
  - Return Merchandise Authorization (RMA)

How do I troubleshoot slow Myri10GE performance?

Table of Contents

- Introduction
- Checklist
- Hints for tuning Linux LAN performance
- Hints for tuning Windows LAN performance
- Hints for tuning Solaris LAN performance
- Hints for tuning FreeBSD LAN performance
- Hints for tuning Mac OS X LAN performance

Introduction

Which PCI-Express motherboard and PCI chipset are you using? and is it a server or desktop machine?

Is your Myri-10G network adapter performance comparable to the following?

- Expected Myri10GE Performance Measurements
- Expected MX-10G Performance Measurements

If your performance is slower than expected, we recommend that you first verify all of the information in the general **Checklist** and then proceed to the OS-specific LAN performance **Tuning** recommendations included below.

We realize that some of the tuning steps are impractical to deploy for real-world applications; however, we recommend that all customers review the tuning steps because we want to ensure that there are not some non-obvious bottlenecks limiting performance (e.g., misconfigured system memory, SELinux stealing CPU cycles, etc). If your machines are capable of achieving line rate with jumbo frames (or even standard frames), it is useful to see that rate with benchmarks to make sure that you have not overlooked some tuning option.

For suggestions in tuning WAN performance, refer to:

- For a 10G WAN network, what Linux tuning do you recommend?
- For a 10G WAN network, what FreeBSD tuning do you recommend?.

For suggestions in tuning NFS performance on Linux 10GbE networks, refer to:

- Do you have recommendations for tuning NFS on a 10GbE network?.

For an abbreviated list of suggestions for Linux, please read:

- What are some of the essential settings to check to improve performance on Linux?

Checklist

We recommend that you first check that we have an x8 PCIe link, and then check that there isn't something wrong with the machine that prevents even a software loopback test, not involving our hardware or driver, from going fast. And lastly, make sure that CPU Frequency Scaling is disabled for the processor.

1. **Check that we have an x8 PCIe 1.1 or 2.0 link**:

   Do you have a Myri-10G "Gen1" (2.5 GT/s) or "Gen2" (5.0 GT/s) PCI-Express Network Adapter?

"Gen2 (PCIe 2.0)" x8 PCIe adapters are two-port adapters with Myricom Product codes **10G-PCIE2-***.

Verifying the x8 "Gen2" (5.0 GT/s) PCIe link speed on two-port Myri-10G adapters

How do I determine if my Gen2 Myri-10G Network Adapter (e.g., 10G-PCIE2-8B2-2C) is installed into a Gen2 or Gen1 PCIe slot on my motherboard?

Verifying the x8 "Gen1" (2.5 GT/s) PCIe link speed on single-port Myri-10G adapters

On Linux:

```
# lspci -d 14c1:8 -vvv | grep Width
                Link: Supported Speed 2.5Gb/s, Width x8, ASPM L0s, Port 0
                Link: Speed 2.5Gb/s, Width x8
```

Important Note: We have observed a bug in **lspci** where the "negotiated width" (the second width in the output) does not reflect the actual one, and is just a copy of the first, "supported width". Type

```
        lspci -xxx -vvv -d 14c1:8
```

and examine the hex dump for the following two fields:

```
60: 05 00 00 00 10 58 00 00 81 f4 03 00 00 00 81 00

                            ^                 ^
                            |                 |
                        supported         negotiated
                           link              link
                           width             width
```

On FreeBSD:

```
% sysctl dev.mxge | grep pcie_link_width
```

On Mac OS X:

```
% sysctl net.myri10ge.en2.lanes
```

On Windows:

On Windows operating systems, you can check the PCIe link speed in the output of **myri10ge_counters.exe**.

Additional Notes:

- Do you have a Myri-10G "Gen1" (2.5 GT/s) or "Gen2" (5.0 GT/s) PCI-Express Network Adapter? For optimal performance, is the "Gen2" Adapter installed into a x8 "Gen2" PCIe slot? For diagnostic details, refer to How do I determine if my Gen2 Myri-10G Network Adapter (e.g., 10G-PCIE2-8B2-2C) is installed into a Gen2 or Gen1 PCIe slot on my motherboard?.

  Myri-10G "Gen1" PCI Express Network Adapters are compatible with "Gen2" slots in hosts; the adapter auto-negotiates operation in the widest available mode (x8, x4, x2, or x1) supported by the slot it is plugged into, and at the 2.5 GT/s data rate.

  Similarly, Myri-10G "Gen2" PCI Express Network Adapters auto-negotiate operation in the widest available mode (x8, x4, x2, or x1) supported by the slot it is plugged into, and at the highest data rate (5 or 2.5 GT/s), but these "Gen2" PCI Express Adapters cannot achieve full performance in "Gen1" PCI Express slots in a host.

  Myri-10G "Gen1" and "Gen2" network adapters are PCIe3-compatible.

- Some PCI-e chipsets that provide a x16 link may **not** support a x8 PCI-Express link width; a link is only required to support its maximum width and x1. Carefully read the hardware specifications of the motherboard to see if its PCI-e x16 slot can "downshift" to x8.
- On some motherboards, the PCI-e slots are all the physical size required by a x16 slot, but electrically of different widths. Refer to the motherboard documentation for the supported width of each of the PCI-e slots.

2. **If you're using Myri10GE, check that the slow-performing test runs quickly in loopback mode.**
   For netperf:

```
% netperf
TCP STREAM TEST to localhost
Recv   Send    Send
Socket Socket  Message  Elapsed
Size   Size    Size     Time     Throughput
bytes  bytes   bytes    secs.    10^6bits/sec

 87380  65536   65536    10.00    8334.21
```

   For iperf:

```
% iperf -c localhost
<...>
[  4]  0.0-10.0 sec  9.23 GBytes  7.92 Gbits/sec
```

3. **Disable CPU Frequency Scaling**. Check the configuration for the motherboard's processor. If CPU frequency scaling (frequency ramping) is enabled, this feature can severely impact the 10G performance. It should be disabled. **On Linux**: To check if CPU Frequency Scaling is enabled on your machine, do the following. When the machine is idle, grep MHz in **/proc/cpuinfo** and see if it is less than the expected CPU frequency.

If these tests yield as-expected performance, please provide us Myricom Technical Support (help@myricom.com) with the following information.

- If you're using Myri10GE, please send us the output of the **myri10ge_bugreport.sh** script. For further details, refer to What is myri10ge_bugreport.sh?.
- Are you testing the Myri-10G Network Adapters point-to-point (switchless) or through a switch? Do you see the same performance in either configuration?

different 10GBase-SR or 10GBase-LR transceiver module and do you see the same performance?
- If you're using Myri10GE, with which MTU are you testing, and which benchmark are you running (**netperf**, **iperf**, etc)? Please ... output and command line arguments to the benchmark that you are using.
- Please send us the output of **dmidecode**. This will provide us with more information about the system, including the BIOS revision, how many CPUs are installed, how many memory DIMMs are installed, etc.

Hints for tuning Linux LAN performance

For an abbreviated list of suggestions for Linux, please read: What are some of the essential settings to check to improve performance on Linux?

For ESX tuning hints, see Do you have performance tuning suggestions for the Myri10GE ESX 4 or 5 driver?.

In addition to the Linux performance tuning suggestions in the Myri10GE Linux README, we also recommend the following checklist for best performance:

1. **Network Buffer Sizes**
   For best performance with Myri10GE or Ethernet Emulation with MX-10G, we recommend that several network buffer sizes be increased from their defaults. Please add the following lines to **/etc/sysctl.conf**

   ```
   net.core.rmem_max = 16777216
   net.core.wmem_max = 16777216
   net.ipv4.tcp_rmem = 4096 87380 16777216
   net.ipv4.tcp_wmem = 4096 65536 16777216
   net.core.netdev_max_backlog = 250000
   ```

   and execute the command

   ```
   sysctl -p /etc/sysctl.conf
   ```

   Note: The first four settings serve to tune things at the socket level. They are only used when the host is an endpoint of a socket connection, and do nothing on a bridge/router. The fourth setting is a NAPI related parameter, and controls how many packets get handled via net_rx_action before they are handed off to a low priority thread. This will affect routing/bridging. The larger this value, the more efficiently the network stack can handle an incoming flood of packets, but the less responsive the machine will be during the flood. See http://www.geocities.com/asimshankar/notes/linux-networking-napi.html, section 2.2.1.

2. **Module compilation**
   You may wish to rebuild with

   ```
       make clean
       make MYRI10GE_ALLOC_ORDER=$ORDER
   ```

   Where $ORDER ranges in value from 1..3.

   Our driver receives into PAGE_SIZE buffers. Increasing the order tells the driver to use larger buffers (where the buffer size is PAGE_SIZE << $ORDER). This decreases the frequency of allocations at the price of doing larger memory allocations which are more prone to fail under heavy memory pressure. We default to 0 order allocations, so as to avoid the risk of allocation failures under memory pressure.

   A good value to choose is make MYRI10GE_ALLOC_ORDER=2, as it results in 16KB allocations. This is the same size allocation as a driver which does not use PAGE_SIZE buffers, and simply allocates 9KB jumbo frames. You may want to experiment with making MYRI10GE_ALLOC_ORDER=3, but they are a bit more likely to fail under heavy memory pressure.

3. **Modprobe arguments**
   You may wish to load the myri10ge module with the **myri10ge_wcfifo=0**argument.

   ```
   # modprobe myri10ge myri10ge_wcfifo=0
   ```

   This results in PIO writes taking a faster path through the NIC at the cost of slightly higher host CPU utilization. This parameter was deprecated in version 1.4.4 and is no longer used.

4. **Interrupt Coalescing**
   Myri10GE
   The Myri10GE driver is ethtool compliant, and you may adjust the interrupt coalescing parameter via

   ```
   ethtool -C $DEVNAME rx-usecs $VALUE
   ```

   The default setting is 25us for Linux kernels prior to 2.6.16, and is 75us for Linux kernels 2.6.16 and later. The default setting is a compromise between latency and CPU overhead. You may wish to reduce **rx-usecs** if latency is more important and you are using a low-latency switch or a point-to-point connection. Similarly, you may wish to increase **rx-usecs** if you are interested in reducing CPU overhead for large transfers. Note that **rx-usecs** controls both transmit and receive coalescing.

   You should try adjusting the interrupt coalescing settings via the command above, where good choices for $VALUE are 50, 75, and 100. The default $VALUE results in slightly lower latency at the cost of slightly higher CPU overhead. If you find a value which works substantially better for you, you can make it permanent by using the myri10ge_intr_coal_delay=$VALUE modprobe argument.

   If lowest possible latency is the most important, refer to the hints in the Myri10GE README, as well as the FAQ entry: How do I achieve the lowest possible latency with Myri10GE?.

   Ethernet Emulation with MX-10G

   If you will be using Ethernet Emulation with MX-10G and you would like to maximize bandwidth (at the cost of higher latency), we recommend increasing the default interrupt coalescing delay from 10us to 80us, with the following command

   ```
   # <install_path>/bin/mx_intr_coal -s 80
   ```

5. **Adaptive Interrupt Coalescing**
   Adaptive interrupt coalescing can provide a good balance between latency and bandwidth optimization. On a server with a mixed workload with latency critical components, consider enabling adaptive interrupt coalescing. To enable adaptive interrupt coalescing, use ethtool:

   ```
       ethtool -C eth2 adaptive-rx on
   ```

overhead. You may wish to experiment with CPU binding, as mentioned below.

7. **Iperf arguments**
   It is generally best to not explicitly specify a window size with **-w**, as this defeats the magic window size auto-tuning done by the Linux kernel. You may want to use smaller length **-l**, such as 64k or 128k.
8. **Write Combining**
   When you load the Myri10GE or MX-10G driver, does it say that Write Combining (WC) is enabled? Enabling Write Combining (WC) on the device's memory range will improve performance. Beginning with version 1.3.0 of the Myri10GE driver, to enable write combining, the driver uses the PAT feature of modern CPUs, rather than using MTRRs. If you are using the latest release of the Myri10GE driver, and WC is disabled, refer to For optimal performance with Myri10GE and MX-10G, how do I enable write-combining in the PCI chipset?.
9. **MSI versus Legacy Interrupts**
   When you load the Myri10GE or MX-10G driver, does it say that MSI interrupts are enabled? If dmesg shows a message like the following, it means that the Linux kernel or BIOS did not allow our device to use MSI interrupts:

   ```
       myri10ge: Error setting up MSI on device 0000:05:00.0, falling back to xPIC
   ```

   or

   ```
       mx WARN: Error -22 setting up MSIs, falling back to legacy interrupts
   ```

   Refer to Why can't I enable MSI on my motherboard? for further explanation and suggestions.

   If it's not possible to enable MSI interrupts with the specific Linux release that you're using, you can make xPIC interrupts less expensive by loading the driver with:

   ```
   # modprobe myri10ge myri10ge_deassert_wait=0
   ```

   or set it at runtime via

   ```
   'echo 0 > /sys/module/myri10ge/myri10ge_deassert_wait'
   ```

   Not using MSI or **myri10ge_deassert_wait=0** costs about 500Mb/s in our performance measurements for a single stream.

10. **MSI-X interrupts and Multiple Receive Queues**
    If your kernel, motherboard and adapter are MSI-X capable, you can take advantage of hardware steering of incoming IPv4 traffic into multiple sets of receive queues. Refer to the **myri10ge_max_slices** load-time option as described in the Myri10GE README. **Note**: For best performance with MSI-X interrupts, we recommend Linux 2.6.20 and later.
11. **Intel Direct Cache Access (DCA)**
    If you have a recent Intel server or workstation chipset, you may be able to take advantage of DCA. DCA causes DMA writes posted by the NIC to be prefetched into a CPU's cache, thereby reducing cache misses and increasing CPU efficiency for received network traffic. For further details, please read the Myri10GE README and Does the Linux Myri10GE driver support Intel Direct Cache Access (DCA)?. For troubleshooting suggestions, please read What does the kernel message "dca_add_requester() failed" indicate?.
12. **Intel Hyperthreading (HTT)**
    If **low latency** is important, we recommend that you disable hyperthreading as it can have an unpredictable affect on latency (consider 2 application threads scheduled on the same core, while others are idle). We recommend that you disable HTT for benchmarking either Myri10GE or MX. **Why?** When HyperThreading (HTT) is enabled, the kernel scheduler may sometimes inadvertently place two CPU intensive execution threads on different HTT cores which correspond to the same physical CPU core. Eg, 2 threads are forced to share the same physical CPU. When running netperf TCP/IP, benchmarks, there is much more overhead than when running a kernel-bypass MX ping-pong benchmark. Hence it is more likely that 2 CPU intensive threads may collide like this. This sharing can generally be avoided with careful use of the netperf "-T $local, $remote" CPU binding to bind the application, and /proc/irq/${irq}/smp_affinity to bind the NIC's interrupt handler. However, it is far easier to benchmark without worrying about these factors, which is why we suggest disabling HTT for benchmarking.
13. **CPU binding/affinity**
    Myri10GE and netperf
    single-core machines
    For the Linux 2.6.9 thru 2.6.15 kernels with which we have tested, the most consistently good netperf results are obtained when the client process (**netperf** sender) is bound to a different CPU than its interrupt handler and the server process (**netserver** receiver) is bound to the same CPU as its interrupt handler. There is very little variance in results between runs or between kernels with this bonding, but it is not the default. Thus, we recommend the use of the **-T0,1** flag to **netperf** version 2.4.2 to ensure this bonding. With the default CPU affinity in **netperf**, and using Linux kernels 2.6.11.9 through 2.6.15.7, we have seen a loss of about 1 Gb/s performance on the sender on dual Opterons.
    dual-core machines
    We obtain better results on a single-socket dual core system when we bind **netserver** to a different core than the interrupt handler. If it were a multi-socket system, then you would want to bind to the same socket, but a different core than the interrupt handler. We suggest looking in **/proc/interrupts**, finding which CPU the irq was bound to, and setting the **netserver** affinity to a different core on the same socket. You may also want to kill irqbalance if it is running, as it may move the smp affinity of the irq without your knowledge.
    Further Details
    On NUMA systems, we generally see the best performance if we bind the netperf/netserver processes to the same *socket* as the interrupt handler is bound to. The interrupt handler and application should both be bound to the CPU closest to the PCI bus our device is plugged into. Generally, you can find this by looking at the /sys/devices/pci$BUS/$BUS:$SLOT/local_cpus cpu mask. To keep things simple, let's a assume a dual CPU, dual core system where the PCI bus is closest to socket0, and our device is eth5. Do the following:

    ```
      grep eth5 /proc/interrupts to find the irq
      echo 01 > /proc/$IRQ/smp_affinity

    run netperf with the -T0,0 argument to bind the processes to the

     same CPU core as the interrupt handler
          OR
     run netperf with the -T1,1 argument to bind the processes to the
     a different CPU core on the same socket as the core handling
     interrupts.
    ```

    MX-10G testing

    Similarly for MX-10G, if you have a multi-core multi-socket motherboard, have you tried binding the MX application to a specific CPU to see if that makes a performance difference?

```
tl --cpubind X --membind X my_app my_app_arg0 my_app_arg1 ...
```

If you're using MPICH-MX, have you tried specifying **MX_CPUS** as discussed in What Run-Time Tuning Parameters for MPICH-MX are available?

If you're using Open MPI, we recommend using the run-time parameter **--mca mpi_paffinity_alone 1**.

14. Which **Linux release (uname -a)** are you using?
    We have encountered slower performance with Linux kernels 2.6.12 through 2.6.16, but this seems to have been rectified with 2.6.17. For good performance on dual-core motherboards, we recommend 2.6.19 or later. For best performance on Intel Core2 motherboards, you need to use 2.6.19 or later. Earlier versions of the Linux kernel use the wrong copy routine for the Core2 CPU, which makes copies into and out of the kernel much more expensive.
15. **TCP timestamps**
    On some Opteron systems, we have found that we have gotten much better throughput by disabling TCP timestamps than one might expect from simply having 12 more bytes of payload in every frame. Try disabling TCP timestamps via

    ```
    sysctl -w net.ipv4.tcp_timestamps=0
    ```

16. Is **TSO enabled**?
    TSO is enabled by default. Refer to the README for details.
17. **Is C1E ("Enhanced Idle") enabled?**
    For best performance on Intel Xeons, we recommend that C1E ("Enhanced Idle") is disabled. Any machine which uses a recent Intel Xeon CPU has C1E. On some machines, it is possible to disable C1E in the BIOS. On other machines, there is no setting. In general, you should disable all power saving in the BIOS if you care about latency. Features like "C1E", speedstep (power saving), and other "enhanced idle" features are great for high-density datacenters with power and cooling constraints and servers which are idle most of the time. However, the introduce a substantial latency when doing interrupts, especially the sort of interprocessor interrupt that is used to wake a process sleeping on another CPU. On recent Nehalem (core i7) based Xeons, I halved round trip time (38us to 19us RTT) by disabling these features in the BIOS:

    ```
    C1E: Disable
    C6 state: Disable
    Speedstep: Disable
    ```

18. Looking at the Linux kernel config file, do you see a large number of **debug options set/enabled**?
    If yes, for decent performance, we strongly suggest that you use a Linux kernel without these debugging options enabled. For example, **CONFIG_DEBUG_SLAB** will definitely negatively impact performance.
19. Is **kernel auditing enabled** in the Linux kernel?
    If yes, for best performance, we strongly suggest that you disable kernel auditing at linux boot time (**audit_enable=0 audit=0**). After booting with this option, please check in the output of **/proc/cmdline** that the option has been passed correctly.
20. Is **SELinux support enabled** in the Linux kernel?
    If yes, for best performance, we strongly suggest that you disable SELinux support by adding **selinux=0** to your kernel command line. After booting with this option, please check in the output of **/proc/cmdline** that the option has been passed correctly.
21. Make sure (using **vmstat**, **mpstat**, etc) that the system is CPU bound and is using all of a single CPU. If it is not, then the insufficient TCP buffer space would be one explanation. Make sure to increase the network buffer sizes as outlined in the README.
22. **GRO versus LRO?**
    On recent kernels (2.6.29 for MYRI10GE_RX_SKBS, or 2.6.31 for a normal build), the driver will default to GRO (so that it can do lro-like things with IPv6). You may see better performance from LRO, rather than GRO. To disable GRO, use ethtool (where supported) or load the module with **myri10ge_gro=0**. **Note:** RHEL5 / CentOS 5 has GRO support in their 2.6.18-164* and newer kernels (should be RHEL/Centos 5.4, but some Centos 5.3 boxes have had that kernel as well).

Hints for tuning Windows LAN performance

As of March 31, 2014, we no longer support Windows XP, 2003, or Vista.

- Which parameters should I consider for tuning assessments on Windows?
- I am seeing poor network performance or high network latency on Windows 2008 and 2012. How do I improve the performance?
- **Performance Tuning Guidelines for Windows 7**: Windows 7 should be autotuning.
- **Performance Tuning Guidelines for Windows Server 2008 R2**: [http://www.microsoft.com/whdc/system/sysperf/Perf_tun_srv-R2.mspx]
- **Performance Tuning Guidelines for Windows Server 2008**: [http://www.microsoft.com/whdc/system/sysperf/Perf_tun_srv.mspx]
- **Performance Tuning Guidelines for Windows Server 2003**: [http://download.microsoft.com/download/2/8/0/2800a518-7ac6-4aac-bd85-74d2c52e1ec6/tuning.doc]
- **Is the Windows Firewall service running?**
  If the Windows Firewall service is running at all (even if it's turned Off in the Firewall settings), performance (frame size and CPU utilization) will be negatively impacted. This service must be stopped/disabled for best performance.

  Note:

  - You can stop the Base Filtering by calling

    ```
    cmd\> net stop bfe
    ```

    from the command prompt.
  - Please also read How to enable ping response on Windows 7?
- **Is RSS (Receive Side Scaling enabled?**
  For Windows 7, Windows 2008 and later: To improve the receive performance, please verify that RSS (Receive Side Scaling), adjustable via the the adapters network properties, is enabled.
- **An explanation of driver settings for NDIS6 drivers (Windows Vista, Server 2008, Windows 7)** and recommended settings can be found here: What are the driver settings available for Windows NDIS6?.
- **We recommend the following TCP registry entries in Windows 2000, XP, and 2003.**

  ```
  HKLM\System\CurrentControlSet\Services\Tcpip\Parameters:
  ```

  -
    - Tcp1323Opts, type REG_DWORD, value set to 1.
    - TcpWindowSize, type REG_DWORD, value set to 512K. For further details, here's a website that discusses TCP registry entries in Windows.

  [http://rdweb.cns.vt.edu/public/notes/win2k-tcpip.htm]

- DefaultReceiveWindow, type REG_DWORD, value set to 65536
- DefaultSendWindow, type REG_DWORD, value set to 65536

Be aware that this sets the default buffer sizes for ALL sockets. This means that on machines that open many sockets you may run into resource exhaustion. Do not use these values without trying the TCP/IP changes first.

- **Slow Vista Network Speed?**
There have been a number of internet posts related to slow network performance when using Windows Vista. For example, here are some suggestions:
  - [http://www.petri.co.il/improve_windows_vista_network_performance.htm]

  Also, please check the value of **NetworkThrottlingIndex**in the Registry. If you see the following value in the registry:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Multimedia\SystemProfile\
Name: NetworkThrottlingIndex
Werttyp: DWORD
```

  set the value to **ffffffff**and you will get smooth playback and capture performance.

  Win7 has the same setting to a limited value but is seems to ignore it. For more details, refer to [http://support.microsoft.com/kb/948066].

- **Slow Windows 7 performance?**

  Please read Why doesn't my Myri10GE performance on Windows 7 match the performance you report?.

- **Benchmarking Network Performance on Windows?**
The performance of socket applications under the Windows operating system is **very** sensitive with respect to the underlying socket API. A key aspect for getting good performance is that the **Winsock2 API** has been used. **Winsock2** introduces overlapping of communication and allows multiple outstanding send or recv requests at a time. Sockets need to be created using **WSASocket**with the overlap flag. The network benchmarking program
  - **ntttcp**

  is a good example of a benchmark program which uses the **Winsock2 API**. **NTTTCP**is a closed-source benchmark available from Microsoft at
  - [http://www.microsoft.com/whdc/device/network/TCP_tool.mspx]

  and is based on the original **ttcp** benchmark. Performance results can vary and are dependent on CPU type and the Windows operating system version. Refer to Myri-10G 10-Gigabit Ethernet Performance Measurements web page for further details. In contrast, some **UNIX**network benchmarking tools like
  - iperf,
  - netperf,
  - nuttcp,
  - ttcp,
  - NetIO

  and others do **not** use the **Winsock2 API** and **may not** perform well (and sometimes not even function correctly) on Windows. Even worse can be the performance when an additional intermediate library such as **cygwin.dll** is required to run the application. You can check your source code and look for **WSASend** and **WSARecv**which will indicate the use of the Winsock2 API. </ol>

Hints for tuning Solaris LAN performance

As of March 31, 2014, we no longer support Solaris 10, Solaris 11, or OpenSolaris.

We recommend the Solaris performance tuning suggestions in the README, as well as the tuning hints for 10GbE on Solaris at [http://www.solarisinternals.com/wiki/index.php/Networks].

In short, there is not a lot of tuning that can be done on Solaris.

Example Solaris myri10ge.conf file.

1. For best performance, please use our GLDv3 Myri10GE driver.
   For details, please read Is the Myri10GE GLDv3 Solaris driver available in Solaris 11 and OpenSolaris?.
2. We recommend the following settings in **/etc/system**:

```
set ddi_msix_alloc_limit=8
set pcplusmp:apic_multi_msi_max=8
set pcplusmp:apic_msix_max=8
set pcplusmp:apic_intr_policy=1
```

   (Note that the pcplusmp may be obsolete is recent S10 And S11.)

3. Here are a few recommends for **/kernel/drv/myri10ge.conf**:
   - Depending on whether or not you are planning to use a 1500b MTU, you can save space by limiting the driver to allocate based on a 1500b MTU by setting:

```
myri10ge_mtu_override=1500;
```

   - If you plan to have lots of connections, you may want to increase the rx buffer counts:

```
myri10ge_bigbufs_initial=4096;
myri10ge_bigbufs_max=32768;
```

   - To enable TCP LRO:

```
myri10ge_lro=1;
```

     To change the max number of packets aggregated together in LRO:

```
myri10ge_lro_max_aggr=2;
```

fixed in recent kernels. If increasing this above 2 results in poor performance, your kernel does not have the fix.

Most of these can be tuned via ndd on /dev/myri10ge0

4. **Lowest Latency?**
   If lowest possible latency is the most important, refer to the hints in the Myri10GE README, as well as the FAQ entry: How do I achieve the lowest possible latency with Myri10GE?.
5. **Highest Bandwidth?**
   If highest possible bandwidth is the most important, refer to the hints in the Myri10GE README, as well as the FAQ entry: How do I achieve the highest possible bandwidth with Solaris Myri10GE?.
6. **For best network performance, which version of OpenSolaris do you recommend?**
   OpenSolaris post snv 106 adds substantially and non-trivial overhead, including extra context switches for the transmit and reception of every packet due to extra threads running in the layer above the device driver. For the best Solaris performance, we suggest Solaris 10, or an older OpenSolaris (from before "crossbow" integrated in build 105). That is the reason why our performance benchmarks for Myri10GE are taken when running 2008.11 (snv_101b_rc2).
7. **Do you recommend iperf to benchmark network performance on Solaris?**
   No, we instead recommend that you use **netperf** to benchmark network performance on Solaris. We do **not** recommend **iperf** to benchmark network performance on non-Linux platforms because iperf is written in such a way that works well on Linux, and which does not impact performance at low packet rates, but which really hurts performance on 10GbE on non-Linux platforms. To elaborate, most network benchmarks (netperf, uperf, *ttcp) take 2 gettimeofday() timestamps across the entire benchmark, one at the start, and one at the end. Iperf calls gettimeofday() around every socket read or write. It then queues up those timestamps to be processed by another thread. On some platforms, gettimeoday() actually results in a system call which triggers a read of a hardware clock across a slow (ISA-like) bus. At the very least, the queuing of timestamps for handling by another thread generates quite a bit of overhead. To mitigate this, you can use a large socket read/write size (**-l 64K** or **-l 128k**) rather than the default 8K. That is why we recommend a benchmark like **netperf**, which focuses less on the threads and timekeeping subsystems and more on networking. We also recommend Sun's uperf, http://www.uperf.org/ for benchmarking Solaris.
8. **Do I need to use a large TCP window size for benchmarking?**
   Yes. Keep in mind that Linux auto-tunes TCP window sizes, and Solaris **does not**. This means that to get a window large enough for decent 10GbE single stream performance, you will need to specify at least a 512KB window size (**-w 512KB** with iperf, or **-s 512K -S 512K** with netperf).

Hints for tuning FreeBSD LAN performance

As of March 31, 2014, we no longer support FreeBSD 6.3, 6.4, 7.x, or 8.x.

We recommend enabling LRO

```
% ifconfig mxge0 up; ifconfig mxge0 lro
```

and to let FreeBSD auto-tune the socket buffer sizes (e.g., leave the network settings at their defaults).

Note: Auto-tuning is only present in FreeBSD 7 and 8, not in FreeBSD 6.

You can verify the PCIe link width in the output of

```
% sysctl dev.mxge | grep pcie_link_width
```

Hints for tuning Mac OS X LAN performance

As of March 31, 2014, we no longer support Mac OS X 10.4, 10.5, 10.6, or 10.7.

Please read the Mac OS X performance tuning suggestions in the Myri10GE Mac OS X README. We also recommend the following checklist for best performance:

1. **Is LRO enabled?** Macs will have poor performance when receiving standard frames (1500-byte MTU) unless LRO is enabled as outlined in the Performance Tuning section of the MacOSX README.
2. **Are you running the Mac OS X 10.5 or 10.6 driver, and with which MTU?** Macs will have poor performance when transmitting standard frames (1500-byte MTU) unless you are running MacOSX 10.6 and are using the 10.6 specific version of our Myri10GE driver. This is because MacOSX 10.6 added support for TSO, which should greatly improve transmit performance of standard frames. (The older MacOSX network driver API (prior to 10.6) does not support TCP Segmentation Offload, and a machine without TSO usually cannot saturate a 10G link when the connection has a MSS of 1448 bytes. You are using the older driver API if you are using the 10.4/10.5 driver on 10.6, or if you are using the 10.5 or older driver.)
3. **Have you increased the socket buffer size?**To increase the socket buffer size used by the filesystem for its connections, the simplest way is to increase the system-wide default socket buffer size. For example, to 768K

   ```
   sudo sysctl -w net.inet.tcp.sendspace=786432
   sudo sysctl -w net.inet.tcp.recvspace=786432
   ```

   Depending on the kernel's default value of **kern.ipc.maxsockbuf**, that may also need to be increased. It is large enough in 10.6, but in 10.5 you will want to also increase it:

   ```
   sudo sysctl -w kern.ipc.maxsockbuf=2097152
   ```

   Also read: How can I restore the socket buffer sizes in MacOSX 10.6?. These suggestions also apply to 10.7.

4. **Iperf arguments** Low iperf performance? Did you increase the socket buffer size on the MacOSX side? Remember that Linux will auto-tune socket buffers, but MacOSX will not. When sending from MacOSX, you will want to explicitly specify add the arguments **-w 768k -l 256k** and see if that improves the performance. The **-w 768k** will increase the socket buffer size, and the **-l 256k** will increase the socket write size from iperf's 8KB default to 256k. We recommend the use of **netperf** or **iperf** to benchmark network performance on Mac OS X.

Contact our sales channels > Request more information > Follow us >

Privacy Policy | Contact