Supermicro Ethernet Switch Performance Test

Straightforward Public Domain Test Demonstrates 40Gbps Wire Speed Performance

SSE-X3348S/SR

Test report- 40G switch port performance on Supermicro SSE-X3348 ethernet switches.

SSE-X3348T/TR

1. Summary

The high-end Supermicro 10-Gbps Ethernet Top-of-Rack (ToR) switches SSE-X3348S/SR and SSE-X3348T/TR have four 40-Gbps Ethernet ports in addition to the forty-eight 10-Gbps Ethernet ports. We performed a basic performance test on those ports to see how close to capacity they can operate. Our test program was the generally available public domain application iperf. We found consistently that the switch ports operated at better than 99% of rated line speed – effectively at "wire speed." We were able to further confirm these results in a benchmark test with a commonly used networking test generator - the Ixia XG12.

2. Objective

Many IT professionals who build and operate high-speed Ethernet networks wish to be assured that their switching equipment performs at the level described in the product specification. Test equipment for evaluating such performance can become quite expensive. However, it is possible to get results without resorting to that expense using a relatively straightforward public domain test application running in one or more high-performance servers such as those from Supermicro.

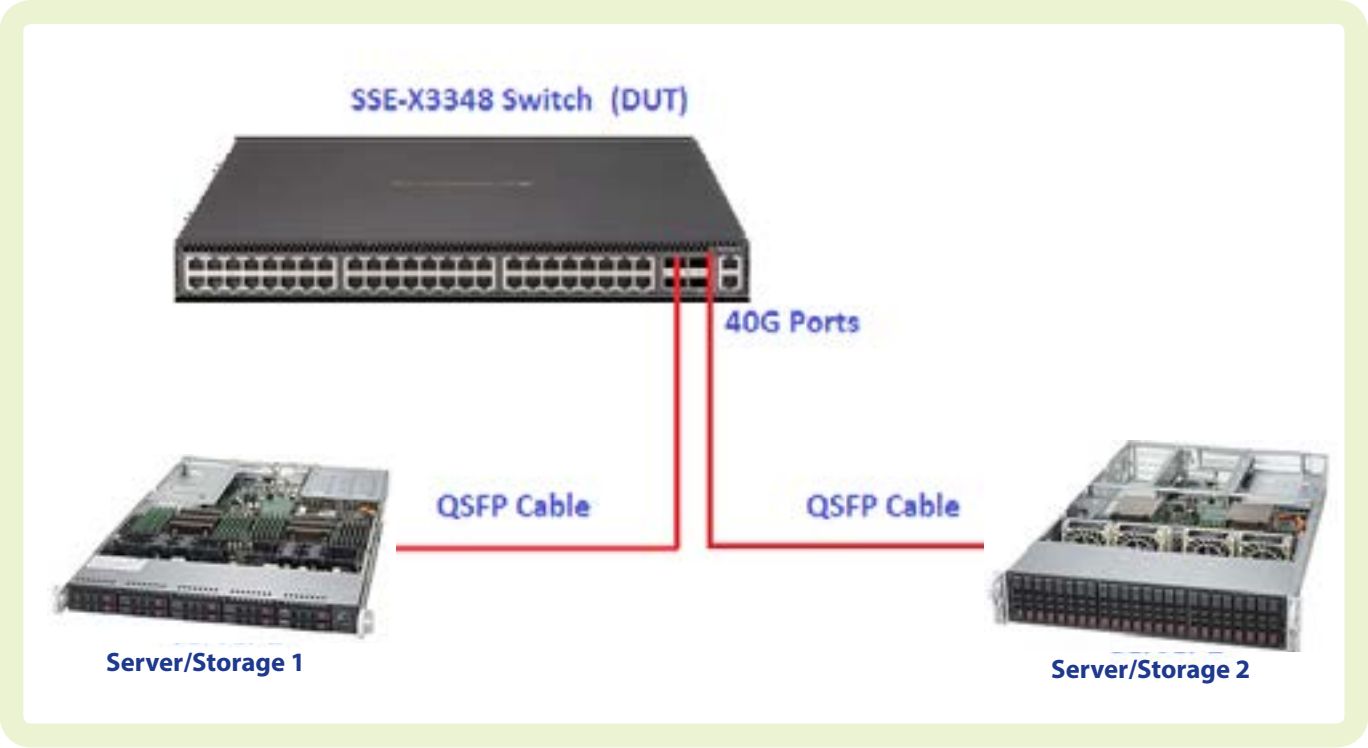
The purpose of this document is to provide details as to how to set up such a test and then how to obtain the best performance from the 40-Gbps ports of Supermicro Switch models SSE-X3348S/SR and SSE-X3348T/TR. This document explains the test methodology, the test bed, and the performance tuning parameters followed in order to obtain the best throughput results. This test determines the maximum throughput the DUT (Device Under Test) can support on the 40-Gbps ports. Users following these steps should easily be able to repeat our "wirespeed" results.

Three cases were tested. The first case was with two servers connected to separate 40-Gbps ports; data was transferred across the switch between them and throughput was measured on the connection. The second case involved four servers connected to separate 40-Gbps ports; data was transferred across the switch between them and throughput was measured on the connections. The final case involved direct connection between the switch and the Ixia XG12 test generator.

3. First Test Case – Two Servers, Two 40-Gbps Ports

3.1 Test Bed:

In this setup we used two servers with Red Hat Linux 6.4 version on them directly connected to the 40-Gbps ports on the DUT as shown below. Both servers were equipped with the Mellanox ConnectX-3 Dual-Port 40 Gigabit Ethernet Adapter Card - Part ID: MCX314A-BCBT and their driver version is "1.5.8.2". The servers were connected to the switch using Supermicro QSFP Cables P/N: CBL-446L 3M length.



3.2 Tuning parameters used in this test:

On the server side

(1) We used the below command to maximize the CPU performance on both the Linux servers for this test.

[root@localhost Desktop]#cpuspeed -C &

[1] 4531

[root@localhost Desktop]#

(2) Ran the "tweak_net.s_" script on both the servers to maximize the performance

(See Appendix A) :

[root@localhost Desktop]# ./tweak_net.s_

[Set sysctls...]

> Set net.core.wmem_max="16777216" [PASS]
> Set net.core.rmem_max="16777216" [PASS]
> Set net.ipv4.tcp_window_scaling="1" [PASS]
> Set net.ipv4.tcp_timestamps="0" [PASS]
> Set net.ipv4.tcp_sack="0" [PASS]
> Default sysctl net.ipv4.tcp_low_latency = 1
> Default sysctl net.ipv4.tcp_adv_win_scale = 2
> Default sysctl net.ipv4.tcp_moderate_rcvbuf = 1
> Default sysctl net.ipv4.tcp_rmem = 4096 87380 16777216
> Default sysctl net.ipv4.tcp_wmem = 4096 65536 16777216
> Default sysctl net.ipv4.tcp_mem = 3077376 4103168 6154752
> Default sysctl net.core.optmem_max = 16777216
> Default sysctl net.core.netdev_max_backlog = 250000
> Default sysctl net.ipv4.tcp_tso_win_divisor = 3

[DO NOT FORGET to tune your kernel paramaters of NIC for max throughput.]
[DO NOT FORGET to enable Jumbo Frame for 10GbE NIC.]
[root@localhost Desktop]#

(3) Set the MTU to 9000 on both the servers:

[root@localhost Desktop]# ifconfig eth30

eth30 Link encap:Ethernet HWaddr 00:02:C9:45:27:30

inet addr:172.33.33.112 Bcast:172.33.33.255 Mask:255.255.255.0

inet6 addr: fe80::202:c9ff:fe45:2730/64 Scope:Link

UP BROADCAST RUNNING MULTICAST MTU:9000 Metric:1

On the Switch side

(4) Set the MTU to 9000 on the connected 40-Gbps interfaces

SMCI# configure terminal

SMCI(config)# interface range qx 0/1-2

SMCI(config-if)# mtu 9000

SMCI(config-if)# end

SMCI#

Switch complete running config attached as Appendix B:

3.3 Test Method/Procedure:

We used the Linux based performance tool iperf to test the 40-Gbps port throughput as shown below:

On the Server side

[root@localhost Desktop]#iperf -s –w512k -l128k

On the Client side

[root@localhost Desktop]# iperf -c 172.33.33.100 -w512k -l128k -i2 –t60 -P4 –d | grep SUM

3.4 Test Results:

Below are the test results. Note that we obtain throughput of 39.6G consistently – 99% of the theoretical maximum, which is effectively "wirespeed".

[root@localhost Desktop]# iperf -c 172.33.33.100 -w512k -l128k -i2 –t60 -P4 –d | grep SUM
[SUM] 0.0- 2.0 sec 9.23 GBytes 39.6 Gbits/sec
[SUM] 2.0- 4.0 sec 9.23 GBytes 39.6 Gbits/sec
[SUM] 4.0- 6.0 sec 9.23 GBytes 39.6 Gbits/sec
[SUM] 6.0- 8.0 sec 9.23 GBytes 39.6 Gbits/sec
[SUM] 8.0-10.0 sec 9.23 GBytes 39.6 Gbits/sec
[SUM] 10.0-12.0 sec 9.23 GBytes 39.6 Gbits/sec
[SUM] 12.0-14.0 sec 9.23 GBytes 39.6 Gbits/sec
[SUM] 14.0-16.0 sec 9.23 GBytes 39.6 Gbits/sec
[SUM] 16.0-18.0 sec 9.23 GBytes 39.6 Gbits/sec
[SUM] 18.0-20.0 sec 9.23 GBytes 39.6 Gbits/sec
[SUM] 20.0-22.0 sec 9.23 GBytes 39.6 Gbits/sec
[SUM] 22.0-24.0 sec 9.23 GBytes 39.6 Gbits/sec
[SUM] 24.0-26.0 sec 9.23 GBytes 39.6 Gbits/sec
[SUM] 26.0-28.0 sec 9.23 GBytes 39.6 Gbits/sec
[SUM] 28.0-30.0 sec 9.23 GBytes 39.6 Gbits/sec
[SUM] 30.0-32.0 sec 9.23 GBytes 39.6 Gbits/sec
[SUM] 32.0-34.0 sec 9.23 GBytes 39.6 Gbits/sec
[SUM] 34.0-36.0 sec 9.23 GBytes 39.6 Gbits/sec
[SUM] 36.0-38.0 sec 9.23 GBytes 39.6 Gbits/sec
[SUM] 38.0-40.0 sec 9.23 GBytes 39.6 Gbits/sec
[SUM] 40.0-42.0 sec 9.23 GBytes 39.6 Gbits/sec
[SUM] 42.0-44.0 sec 9.23 GBytes 39.6 Gbits/sec
[SUM] 44.0-46.0 sec 9.23 GBytes 39.6 Gbits/sec
[SUM] 46.0-48.0 sec 9.23 GBytes 39.6 Gbits/sec
[SUM] 48.0-50.0 sec 9.23 GBytes 39.6 Gbits/sec
[SUM] 50.0-52.0 sec 9.23 GBytes 39.6 Gbits/sec
[SUM] 52.0-54.0 sec 9.23 GBytes 39.6 Gbits/sec
[SUM] 54.0-56.0 sec 9.23 GBytes 39.6 Gbits/sec
[root@localhost Desktop]#

4. Second Case - Performance Test result with all four 40-Gbps ports in use:

In this test we expanded the system to use 4 Servers. We ran the iperf test between Server 1 & Server 2 and at the same time between Server 3 & Server 4.

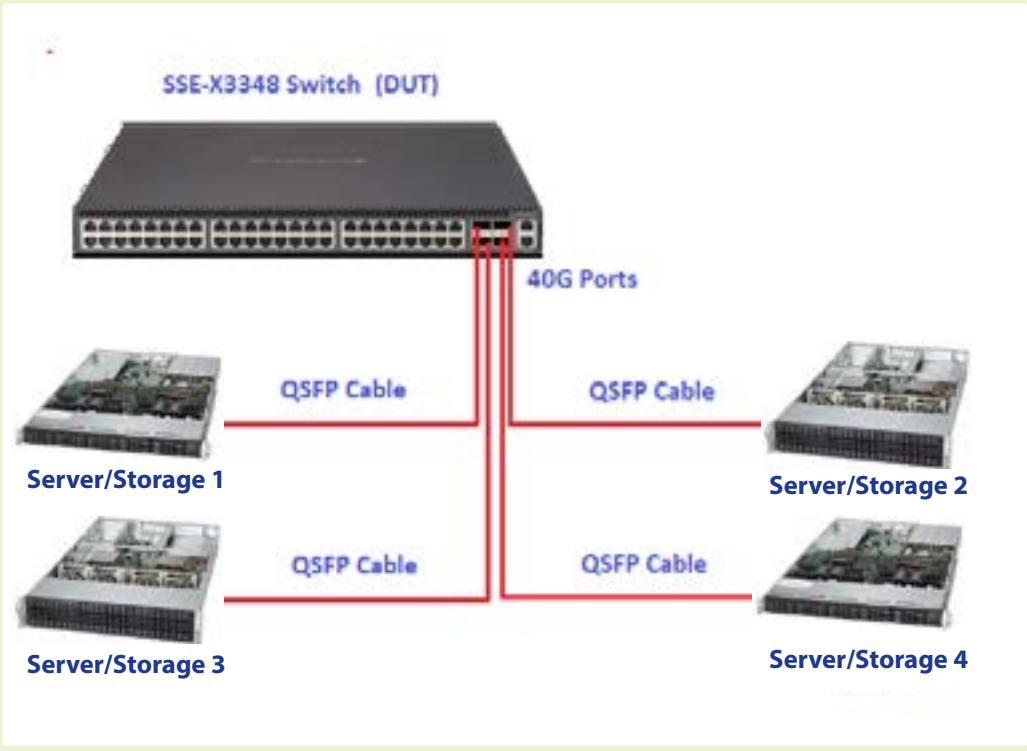
4.1 Test Bed:

Server 1 is an iperf Server and Server 2 is an iperf Client.

Server 3 is an iperf Server and Server 4 is an iperf Client.

4.2 Tuning parameters used in this test:

We used the same parameters for this test as in the previous test as listed in Section 3.2.



4.3 Test Results:

4.3.1 Server 1 and Server 2:

[root@localhost Desktop]# iperf -c 172.33.33.100 -w512k -l128k -i2 –t3600 -P4 –d | grep SUM
[SUM] 88.0-90.0 sec 9.13 GBytes 39.2 Gbits/sec
[SUM] 90.0-92.0 sec 9.11 GBytes 39.1 Gbits/sec
[SUM] 92.0-94.0 sec 9.12 GBytes 39.2 Gbits/sec
[SUM] 94.0-96.0 sec 9.13 GBytes 39.2 Gbits/sec

[SUM] 96.0 - 98.0 sec 9.09 GBytes 39.0 Gbits/sec
[SUM] 98.0 -100.0 sec 9.12 GBytes 39.2 Gbits/sec
[SUM] 100.0-102.0 sec 9.12 GBytes 39.2 Gbits/sec
[SUM] 102.0-104.0 sec 9.11 GBytes 39.1 Gbits/sec
[SUM] 104.0-106.0 sec 9.12 GBytes 39.2 Gbits/sec

4.3.2 Server 3 and Server 4:

[root@localhost Desktop]# iperf -c 172.33.33.150 -w512k -l128k -i2 –t3600 -P4 –d | grep SUM

[SUM] 872.0-874.0 sec 9.17 GBytes 39.4 Gbits/sec
[SUM] 874.0-876.0 sec 9.18 GBytes 39.4 Gbits/sec
[SUM] 874.0-876.0 sec 9.17 GBytes 39.4 Gbits/sec
[SUM] 876.0-878.0 sec 9.22 GBytes 39.6 Gbits/sec
[SUM] 876.0-878.0 sec 9.17 GBytes 39.4 Gbits/sec
[SUM] 878.0-880.0 sec 9.21 GBytes 39.6 Gbits/sec
[SUM] 878.0-880.0 sec 9.17 GBytes 39.4 Gbits/sec
[SUM] 880.0-882.0 sec 9.21 GBytes 39.6 Gbits/sec
[SUM] 880.0-882.0 sec 9.18 GBytes 39.4 Gbits/sec
[SUM] 882.0-884.0 sec 9.21 GBytes 39.6 Gbits/sec
[SUM] 882.0-884.0 sec 9.18 GBytes 39.4 Gbits/sec
[SUM] 884.0-886.0 sec 9.21 GBytes 39.6 Gbits/sec
[SUM] 884.0-886.0 sec 9.18 GBytes 39.4 Gbits/sec
[SUM] 886.0-888.0 sec 9.21 GBytes 39.6 Gbits/sec
[SUM] 886.0-888.0 sec 9.18 GBytes 39.4 Gbits/sec
[SUM] 888.0-890.0 sec 9.21 GBytes 39.6 Gbits/sec
[SUM] 888.0-890.0 sec 9.18 GBytes 39.4 Gbits/sec
[SUM] 890.0-892.0 sec 9.21 GBytes 39.6 Gbits/sec
[SUM] 890.0-892.0 sec 9.17 GBytes 39.4 Gbits/sec
[SUM] 892.0-894.0 sec 9.21 GBytes 39.6 Gbits/sec
[SUM] 892.0-894.0 sec 9.17 GBytes 39.4 Gbits/sec
[SUM] 894.0-896.0 sec 9.21 GBytes 39.6 Gbits/sec
[SUM] 894.0-896.0 sec 9.18 GBytes 39.4 Gbits/sec
[SUM] 896.0-898.0 sec 9.21 GBytes 39.6 Gbits/sec
[SUM] 896.0-898.0 sec 9.18 GBytes 39.4 Gbits/sec
[SUM] 898.0-900.0 sec 9.21 GBytes 39.6 Gbits/sec
[SUM] 898.0-900.0 sec 9.20 GBytes 39.5 Gbits/sec
[SUM] 900.0-902.0 sec 9.21 GBytes 39.6 Gbits/sec
[SUM] 900.0-902.0 sec 9.20 GBytes 39.5 Gbits/sec

[SUM] 902.0-904.0 sec 9.21 GBytes 39.6 Gbits/sec
[SUM] 902.0-904.0 sec 9.20 GBytes 39.5 Gbits/sec
[SUM] 904.0-906.0 sec 9.21 GBytes 39.6 Gbits/sec
[SUM] 904.0-906.0 sec 9.17 GBytes 39.4 Gbits/sec
[SUM] 906.0-908.0 sec 9.21 GBytes 39.6 Gbits/sec
[SUM] 906.0-908.0 sec 9.18 GBytes 39.4 Gbits/sec
[SUM] 908.0-910.0 sec 9.21 GBytes 39.6 Gbits/sec
[SUM] 908.0-910.0 sec 9.19 GBytes 39.5 Gbits/sec
[SUM] 910.0-912.0 sec 9.21 GBytes 39.6 Gbits/sec
[SUM] 910.0-912.0 sec 9.17 GBytes 39.4 Gbits/sec
[SUM] 912.0-914.0 sec 9.21 GBytes 39.6 Gbits/sec
[SUM] 912.0-914.0 sec 9.17 GBytes 39.4 Gbits/sec
[SUM] 914.0-916.0 sec 9.21 GBytes 39.6 Gbits/sec
[SUM] 914.0-916.0 sec 9.17 GBytes 39.4 Gbits/sec
[SUM] 916.0-918.0 sec 9.21 GBytes 39.6 Gbits/sec
[SUM] 916.0-918.0 sec 9.17 GBytes 39.4 Gbits/sec
[SUM] 918.0-920.0 sec 9.21 GBytes 39.6 Gbits/sec
[SUM] 918.0-920.0 sec 9.17 GBytes 39.4 Gbits/sec
[SUM] 920.0-922.0 sec 9.21 GBytes 39.6 Gbits/sec
[SUM] 920.0-922.0 sec 9.20 GBytes 39.5 Gbits/sec
[SUM] 922.0-924.0 sec 9.21 GBytes 39.6 Gbits/sec

Note that we again obtain throughput of 39.6G consistently – 99% of the theoretical maximum, which is effectively "wirespeed".

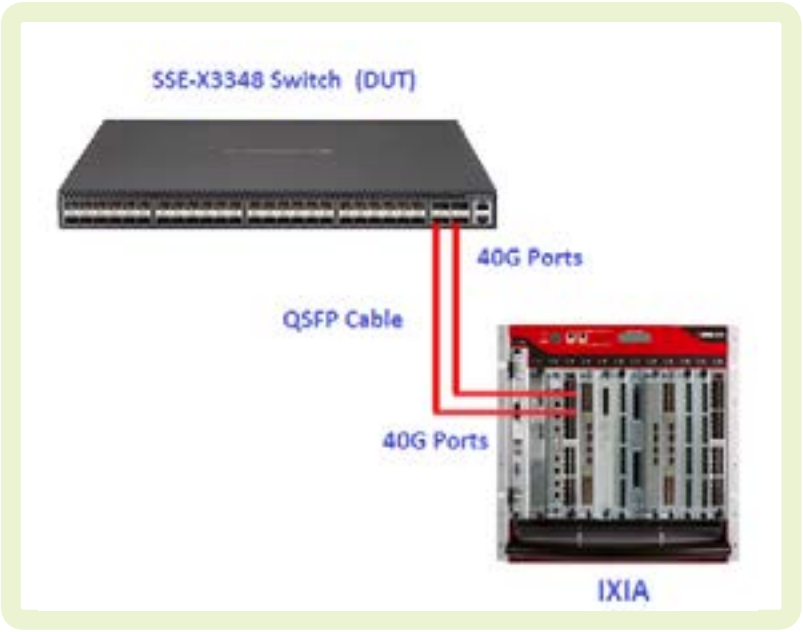
5. FrameLoss test performed on X3348 Switch using Ixia Test Generator:

Finally, as a confirming benchmark, we used a standard test on the Ixia Test Generator to see what results could be obtained through direct measurement without the potential overhead of the iPerf application and the server/NIC combination. The Ixia XG12 was set up to generate 40 Gbps of traffic at a range of frame sizes. The test equipment then compares the received frames with the generated frames and looks for errors, dropped frames, etc.

The Chart in section 5.3 is an output from the Ixia XG12. It shows the results of the FrameLoss test performed on SSE-X3348 Switch with various frame sizes. We tested with Avago's Active Cable (Fiber Optic cable) as well as DAC (Copper cables). In all cases we received 100% of frames sent across the 40-Gbps links without any loss.

5.1 Test Bed - X3348 Switch using Ixia Test Generator:

We used the same parameters for this test as in the previous test as listed in Section 3.2.

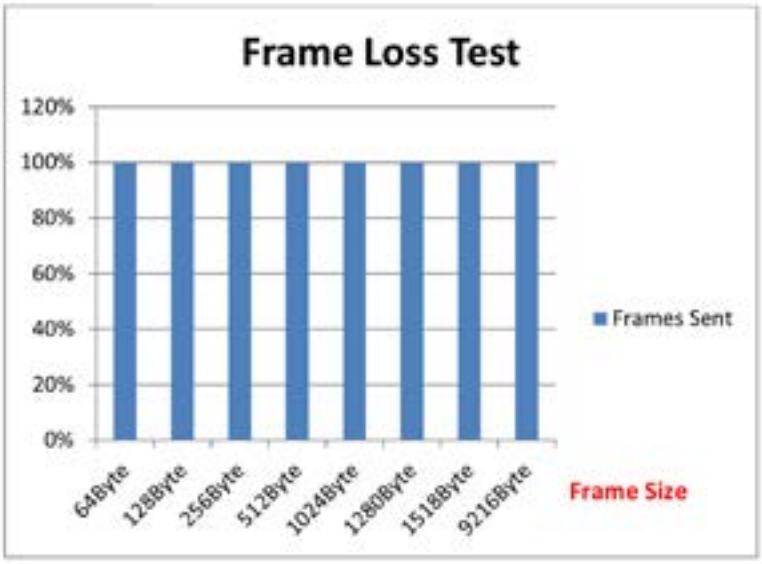


5.2 Test Bed Configuration:

The Configuration and test tools used on this test are as shown below:

Switch firmware	1.0.4-3
Ixia IxNetwork	7.30.917.12 EA
IxOS	6.70.1050.7 EA
Protocol Version	7.30.1030.14
Ixia 40 Module	6 Port Lava AP40/100GE 2P
Ixia Chassis Type	Ixia XG12

5.3 Test Results:



Appendices

Appendix A - tweak net.s Script

```
#!/bin/sh
#
write_sysctls="0"

#moderate_rcvbuf="1"
moderate_rcvbuf=""
tcp_low_latency=""
tcp_adv_win_scale=""

# Increase maximum write socket buffer size.
core_wmem_max="16777216"

# Increase maximum read socket buffer size.
core_rmem_max="16777216"

# Enable TCP window scaling option.
tcp_window_scaling="1"

# Disable timestamps to increase throughput.
tcp_timestamps="0"

# Disable SACK to increase throughput.
tcp_sack="0"

# TCP read buffer (min/default/max), default 4096 87380
 174760.
# overrides net.core.rmem_default.
#ipv4_tcp_rmem="4096 262144 16777216"
ipv4_tcp_rmem=""

# TCP write buffer (min/default/max), default 4096 16384
131072.
# overrides net.core.wmem_default.
#ipv4_tcp_wmem="4096 262144 16777216"
ipv4_tcp_wmem=""
```

```
# TCP memory allocation (min/pressure/max).
# default values are calculated by the kernel at boot time and
depend
# on the amount of physical memory.
ipv4_tcp_mem=""

# max length of iovec or ancilliary data.
# default 20480.
#optmem_max="524288"
optmem_max=""

# log length of network packets. kernel will drop unprocessed
packets
# beyond this. simple algorithm for throughput:
# <backlog> * 100(HZ) * <avg bytes/packet> = throughput
 bytes/second.
# log length of network packets.
#netdev_max_backlog="200000"
netdev_max_backlog=""

# Allows control over what percentage of the congestion
 window can be
# consumed by a single TSO frame. Default is 3 on older
kernels, 8 on new.
tso_win_divisor=""


#
# sysctl config file
sysctl_conf_file="/etc/sysctl.conf"
# sysctl_data array, need to include the sysctl name and the
 data variable.
# Comma delimited.
sysctl_data=(
  "net.core.wmem_max,$core_wmem_max"
  "net.core.rmem_max,$core_rmem_max"
  "net.ipv4.tcp_window_scaling,$tcp_window_scaling"
  "net.ipv4.tcp_timestamps,$tcp_timestamps"
  "net.ipv4.tcp_sack,$tcp_sack"
```

```
"net.ipv4.tcp_low_latency,$tcp_low_latency"
  "net.ipv4.tcp_adv_win_scale,$tcp_adv_win_scale"
  "net.ipv4.tcp_moderate_rcvbuf,$moderate_rcvbuf"
  "net.ipv4.tcp_rmem,$ipv4_tcp_rmem"
  "net.ipv4.tcp_wmem,$ipv4_tcp_wmem"

"net.ipv4.tcp_mem,$ipv4_tcp_mem"
  "net.core.optmem_max,$optmem_max"
  "net.core.netdev_max_backlog,$netdev_max_backlog"
  "net.ipv4.tcp_tso_win_divisor,$tso_win_divisor"
)

# function _set_color(), next print will be in color.
# _set_color pass|fail|warn|norm
_set_color() {
 pass="echo -en \\033[1;32m"
 fail="echo -en \\033[1;31m"
 warn="echo -en \\033[1;33m"
 deft="echo -en \\033[1;35m"
 norm="echo -en \\033[0;39m"

 eval \$$1
 }

 # function _set_column(), column width for print status.
 # Uses globally defined variable 'column'.
 # _set_column <number of columns>
 _set_column() {
  [ -n "$column_width" ] && echo -en "\\033[${column_width}
 G"
 }

 # function not(). negate value for double parentheses testing.
 # (( $(not <value>) ))
 not() {
  if [ -n "$1" ] && (( $1 )); then
    echo 0
  else
    echo 1
  fi
 }
```

```
# function _print_stat(), call to print the status.
# _print_stat "<message>"
_print_stat() {
 _set_color norm
 echo -n " [ "

set_color $1
 echo -n $2
 _set_color norm
 echo " ]"
}

# function Print(), prints a message.
# Print "<message>"
Print() {
 [ -n "$1" ] && prev_print="$1"
 [ -n "$1" ] && echo -en " > $1"
}

# function Info(), prints an info message.
# Info "<message>"
Info() {
 [ -n "$1" ] && echo -e " [ $1 ]"
}

# function DefInfo(), prints an default value info message.
# DeInfo "<message>"
DefInfo() {
 [ -n "$1" ] && { _set_color deft; echo -e " > Default sysctl $*";
_set_color norm; }
}
# function Pass(), prints pass status.
# Pass

Pass() {
 [ -n "$1" ] && prev_print=$1
 [ -n "$1" ] && echo -n " > $1"
 _set_column
 _print_stat pass PASS
}
```

```
# function Fail(), prints fail status.
# Fail
Fail() {
 local out=$1

[ -n "$out" ] && echo -n " * $out"
 _set_column
 _print_stat fail FAIL
 prev_print=$out
}

# function Warn(), prints (optional) message and warn status.
# Warn "<message>"
Warn() {
 local out=$1
 [ -n "$out" ] && echo -n " > $out"
 _set_column
 _print_stat warn WARN
 prev_print=$out
}
Info "Set sysctls..."
if (( $write_sysctls )); then
 Info "Writing sysctl entries to $sysctl_conf_file."

# Create a backup file.
 if [ ! -e "$sysctl_conf_file.perftune.bak" ]; then
  cp -fa $sysctl_conf_file "$sysctl_conf_file.NetTune.bak"
 fi
fi
IFS=$'\n'
for control in ${sysctl_data[@]}; do
 unset IFS
 sysctl_param=${control%%,*}
 sysctl_param=$(echo $sysctl_param | sed 's/^[ \t]*//;s/[
\t]*$//')
 data=${control##*,}

data=$(echo $data | sed 's/^[ \t]*//;s/[ \t]*$//')
 [ -z "$data" ] && { DefInfo $(sysctl $sysctl_param); continue; }
 unset failed_sysctl
```

```
if sysctl $sysctl_param >/dev/null 2>&1; then
   Print "Set $sysctl_param=\"$data\""
   sysctl -w "$sysctl_param=$data" >/dev/null 2>&1 && Pass ||
    { (( failed_sysctl++ )); Fail; }
   if (( $write_sysctls )) && (( $(not $failed_sysctl_file) )); then
    if ! grep $sysctl_param $sysctl_conf_file >/dev/null 2>&1;
 then
      echo "$sysctl_param = $data" >> $sysctl_conf_file ||
       (( failed_sysctl_file++ ))
    else
      # Entry already exists, overwrite it!
      cat $sysctl_conf_file |
       sed "s/$sysctl_param.*/$sysctl_param = $data/" \
        > "$sysctl_conf_file.tmp" ||
        (( failed_sysctl_file++ ))
      if (( $(not $failed_sysctl_file) )); then
       mv -f "$sysctl_conf_file.tmp" $sysctl_conf_file
      fi
    fi
   fi
 else

fi
done
unset IFS
(( $failed_sysctl )) && Warn "Some sysctls failed, system may
 not be tuned!"
(( $failed_sysctl_file )) && Fail "Unable to write to $sysctl_

conf_file."
(( $(not $failed_sysctl) & $(not $failed_sysctl_file) )) && { Info
 "DO NOT FORGET to tune your kernel paramaters of NIC for
 max throughput."; Info "DO NOT FORGET to enable Jumbo
 Frame for 10GbE NIC."; }
```

Appendix B - Switch Configuration

```
SMCI#
SMCI# SH RUN
```

```
BUILDING CONFIGURATION...
SWITCH ID      HARDWARE VERSION            FIRMWARE
 VERSION
0          SSE-X3348TR REV.1  (P5-01)     1.0.4-1

IP ADDRESS DHCP
DEVICE NAME SMCI

VLAN 1
 PORTS GI 0/1-2 UNTAGGED
 PORTS EX 0/1-48 UNTAGGED
 PORTS QX 0/1-4 UNTAGGED
EXIT

SYSTEM MTU 9000

INTERFACE GI 0/1
MTU 9000

INTERFACE GI 0/2
MTU 9000

INTERFACE EX 0/1
MTU 9000

INTERFACE EX 0/2
MTU 9000

INTERFACE EX 0/3
MTU 9000

INTERFACE EX 0/4
MTU 9000

INTERFACE EX 0/6
MTU 9000

INTERFACE EX 0/7
MTU 9000

INTERFACE EX 0/8
MTU 9000
```

Appendix C - just in case, how to download and install iperf:

Step 1. Download the iperf tool from the following site

#wget http://sourceforge.net/projects/iperf/files/iperf-2.0.5.tar.gz/download

Download iperf-2.0.5.tar.gz

Step 2. Extract & Unzip iperf-2.0.5.tar.gz

#tar -xzvf iperf-2.0.5.tar.gz

cd iperf-2.0.5

./configure — configure for your machine

make — compile Iperf

#make install — install Iperf, if desired

40-Gbps NIC Card ： Mellanox ConnectX-3 Dual-Port 40 Gigabit Ethernet Adapter Card - Part ID: MCX314A-BCBT

QSFP+ Cable ： Supermicro QSFP Cable P/N:CBL-446L 3M length

40-Gbps NIC Driver Version:

[root@localhost Desktop]# ethtool -i eth30

driver: mlx4_en (MT_1090110023_CX-3)

version: 1.5.8.2 (May 2012)

firmware-version: 2.30.3000

bus-info: 0000:03:00.0

[root@localhost Desktop]#

For more information on the Supermicro Ethernet Switch Performance Test, please contact your Supermicro Sales Representative or visit:
www.supermicro.com