Performance Analysis
April 2007

Linux Networking Scalability on High-Performance Scalable Servers

By Barry Arndt
IBM Systems and Technology Group

Contents

Introduction

Much has already been written about networking performance, optimization, and tuning on a variety of hardware, platforms, and operating systems under various workloads. The proliferation of high-performance scalable servers (such as the IBM® eServer™ xSeries® x460 and the IBM System x™ 3950) has added a new level of complexity to networking and system performance. For instance, scalable servers whose capacity can be increased by adding full chassis (or nodes) add networking scalability across multi-node systems as a significant ingredient in overall system performance. This paper examines and demonstrates Linux® networking scalability on a multi-node, high-performance system as it utilizes system board gigabit Ethernet adapters from 1 to 4 nodes. In addition, the paper describes and demonstrates problematic networking scalability situations and gives tips on how to avoid them.

Systems Configurations

For this paper, the system under test (SUT) is a 4-node IBM eServer xSeries 460 running SUSE Linux Enterprise Server 10 for AMD64 and EM64T (x86-64). Each of the nodes has the configuration shown in the following table.

System	IBM eServer xSeries 460
CPU	4 x 64-bit Intel® Xeon® Processor 7040 3.0 GHz
Memory	32 GB (8 x 1 GB DIMMs distributed on 4 memory cards)
Ethernet Adapters	Broadcom 5704 10/100/1000 dual Ethernet/ on system board/ 64-bit 266 MHz PCI-X 2.0
Network Driver	tg3 c3.49
Network Type	1 Gigabit Ethernet
Threading	Hyper-Threading Technology™

The drivers for all test scenarios are IBM System p5™ 550 systems, each with two Intel Dual-Port Ethernet adapters, running Red Hat Enterprise Linux 4 Update 4. The 4-node bonding test also includes a 2-node IBM eServer xSeries 460 running SUSE Linux Enterprise Server 10 for AMD64 and EM64T (x86-64). The SUT and drivers are networked through a Cisco Catalyst 3750G-24TS switch.

Test Methodology

The netperf benchmark, specifically the unidirectional stream (TCP_STREAM) test, was chosen for the scalability-demonstration workload for a variety of reasons, including its simplicity, measurability, stability on Linux, widespread acceptance, and ability to accurately measure (among others) bulk data transfer performance. It is a basic client-server model benchmark and contains two corresponding executables, `netperf` and `netserver`. The simple TCP_STREAM test times the transfer of data from the netperf system to the netserver system to measure how fast one system can send data and how fast the other can receive it. Upon execution, netperf establishes a control connection (via TCP) to the remote system. That connection is used to pass configuration information and results between systems. A separate connection is used for measurement, during which the control session remains without traffic (other than that required by some TCP options).

In all of the tests outlined in this paper, network throughput and CPU utilization were measured while the IBM eServer xSeries 460 performed either network sends (`netperf`), network receives (`netserver`), or both simultaneously (bidirectional). The throughput between client and server was tracked on the client send side and is reported in this paper as it was recorded by the netperf benchmark. Each full test run for each environment consisted of 3-minute stream tests for each of 15 send message sizes ranging from 64 bytes to 256K bytes. That range includes message sizes of 1460 bytes and 1480 bytes so that their total packet sizes closely bound the default maximum transmit unit (MTU) size of 1500, after which Linux breaks messages into smaller packets to be sent on the network. CPU utilization was measured on the SUT and is reported in this paper as it was recorded by the sar utility (from the sysstat package) as the system average for the duration of the netperf test. All CPU and interrupt behavior information was also derived from the sar data.

Configurations and parameters were modified to affect behavior in the scalability demonstration. Enabling and disabling them in various combinations caused differing results. The *SMP IRQ Affinity* bitmask, /proc/irq/nnn/smp_affinity, can be set to designate which CPUs are permitted to process specific interrupts. Linux sets them to default values at initialization time. A daemon called **`irqbalance`** can be started to dynamically distribute hardware interrupts across processors. If enabled, it iteratively alters the smp_affinity bitmasks to perform the balancing. The **`numactl`** program can be used to bind specific processes to CPUs and/or memory on specific nodes. Linux network *bonding* provides a variety of methods for aggregating multiple network interfaces into a single logical interface, and may be an attractive network administration feature for use on multi-node servers.

Performance and Scalability Results

"Out of the Box" Configuration

The "out of the box" tests were run with no software configuration changes. In this environment, the `irqbalance` daemon is started by default during Linux initialization. SMP IRQ affinity was not changed, and `numactl` and bonding were not used.

The first of the netserver scalability tests utilized a single instance of `netserver` on each of the two system board Ethernet adapters on the first node of the SUT. Each instance of `netserver` listened on a dedicated port and IP address, and each Ethernet adapter's IP address was on a separate subnet to ensure dedicated traffic. The remote drivers ran corresponding instances of `netperf` to provide stream traffic in a one-to-one mapping of remote `netperf` instances to SUT `netserver` instances. The full test run measured network stream throughput and system CPU utilization for 15 different send message sizes for 3 minutes per message size.

The second netserver scalability test utilized all four system board Ethernet adapters on the first two nodes, and the third test utilized all eight system board Ethernet adapters on all four nodes. The number of SUT `netserver` instances and remote `netperf` instances were increased accordingly for each test.

Figure 1 shows the network stream throughput and CPU utilization for the netserver scalability test runs while utilizing the system board Ethernet adapters on 1, 2 and 4 nodes of the SUT. The throughput shown is the sum throughput of all utilized Ethernet adapters for each test run, and CPU utilization shown is the system average for the duration of the each test run.

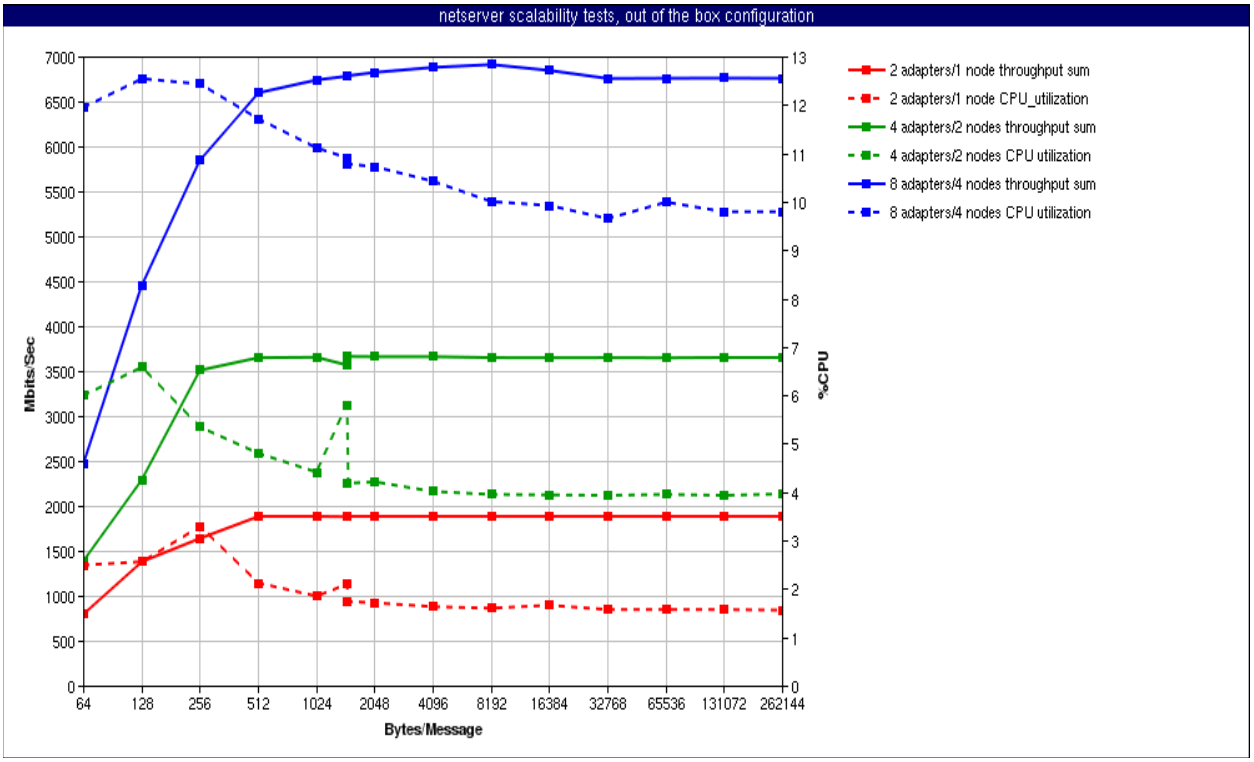

Figure 1. netserver on SUT in "out of the box" configuration

Next, netperf scalability tests were run just like the netserver scalability tests, except that `netperf` was run on the SUT while `netserver` was run on the remote systems.

Figure 2 shows the network stream throughput and CPU utilization for the netperf scalability test runs while utilizing the system board Ethernet adapters on 1, 2 and 4 nodes of the SUT. The throughput shown is the sum throughput of all utilized Ethernet adapters for each test run, and CPU utilization shown is the system average for the duration of the each test run.

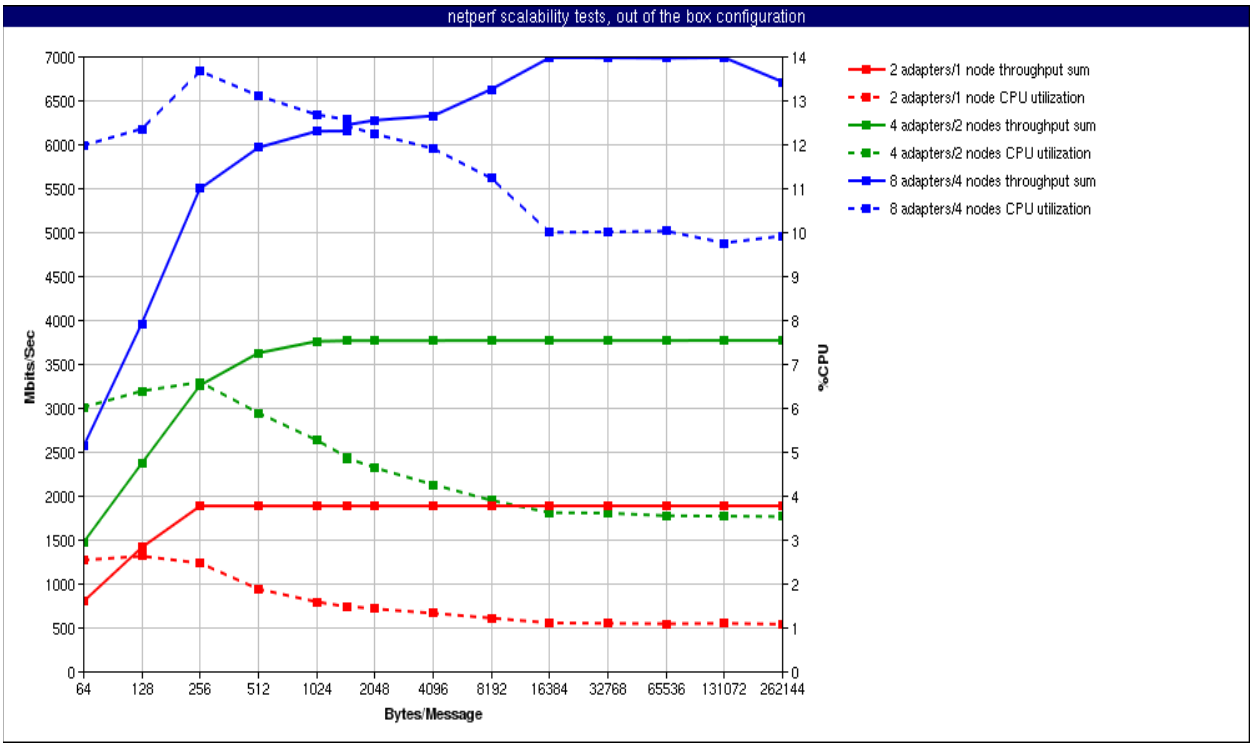

Figure 2. netperf on SUT in "out of the box" configuration

Finally, bidirectional scalability tests were run similar to the previous netserver and netperf tests. However, in this case, only the first system board Ethernet adapter of any node was utilized, and by one instance of `netperf`, along with one instance of `netserver`. Restated, there were two benchmark instances, one `netperf` and one `netserver`, per Ethernet adapter, and only one Ethernet adapter per node was utilized. Each corresponding remote instance of `netperf` or `netserver` ran on its own Ethernet adapter to ensure fullest possible traffic to and from the SUT.

Figure 3 shows the network stream throughput and CPU utilization for the bidirectional scalability test runs while utilizing the system board Ethernet adapters on 1, 2 and 4 nodes of the SUT. The throughput shown is the sum throughput of all utilized Ethernet adapters for each test run, and CPU utilization shown is the system average for the duration of the each test run.

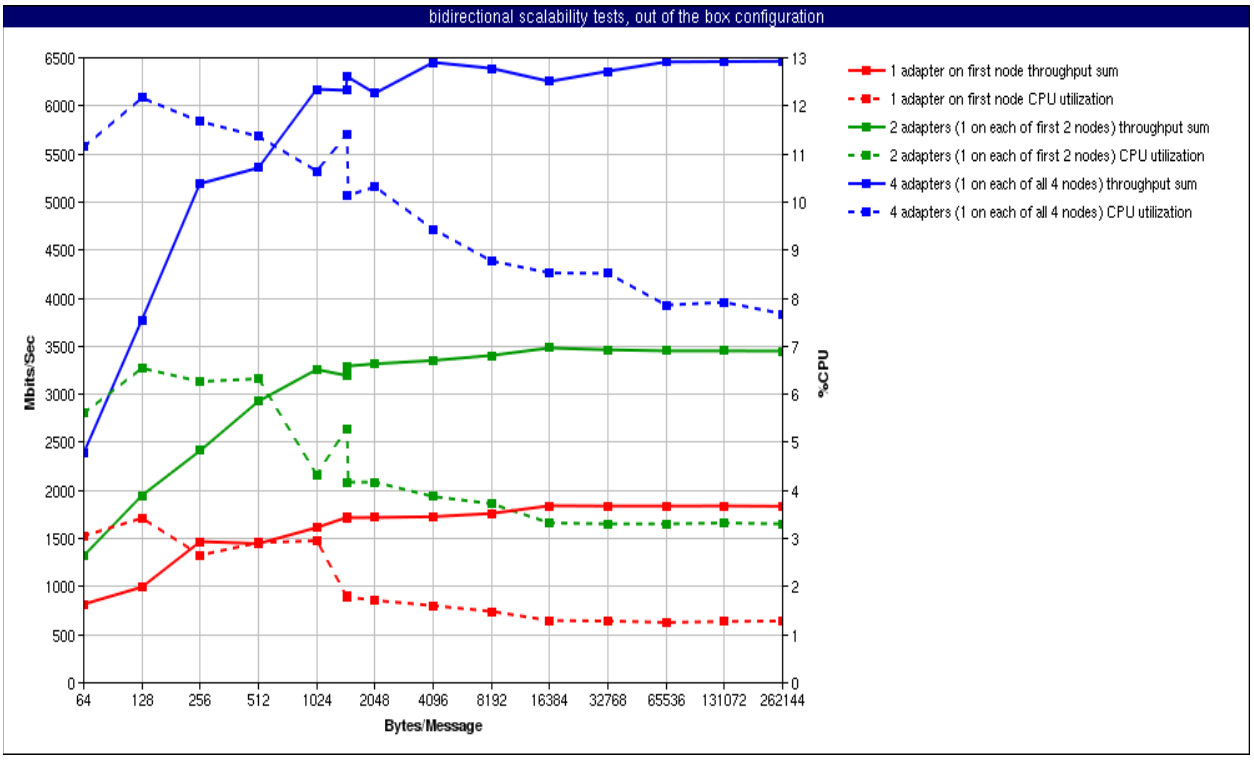

Figure 3. netperf and netserver (bidirectional) on SUT in "out of the box" configuration

Throughput scaling from 2 adapters/1 node to 4 adapters/2 nodes was computed for each send message size. For the netserver scalability tests, those values range from 1.647 for smaller message sizes to 1.944 for larger message sizes. The average for all those values is 1.918. Similarly, CPU utilization scaling from 2 adapters/1 node to 4 adapters/2 nodes was computed for each send message size. For the netserver scalability tests, those values range from 2.770 to 1.623. The average for all of those values in this environment is 2.417.

Throughput and CPU utilization scaling from 4 adapters/2 nodes to 8 adapters/4 nodes was also computed for each message size. These throughput scaling values range from 1.666 to 1.900 with an average of 1.847. The CPU utilization scaling values range from 2.599 to 1.884 with an average of 2.386.

The average throughput scaling from 2 adapters/1 node to 8 adapters/4 nodes over all send message sizes is 3.542. The average CPU utilization scaling from 2 adapters/1 node to 8 adapters/4 nodes over all send message sizes is 5.755.

The scaling computations were made and averaged for the netserver, netperf, and bidirectional tests and are shown in the following table.

SUT Scalability Test	Average Throughput Scaling from 1 to 2 Nodes	Average Throughput Scaling from 2 to 4 Nodes	Average Throughput Scaling from 1 to 4 Nodes	Avg. CPU Utilization Scaling from 1 to 2 Nodes	Avg. CPU Utilization Scaling from 2 to 4 Nodes	Avg. CPU Utilization Scaling from 1 to 4 Nodes
netserver	1.918	1.847	3.542	2.417	2.386	5.755
netperf	1.940	1.732	3.371	3.109	2.537	7.973
bidirectional	1.888	1.892	3.569	2.368	2.274	5.413

Since SMP IRQ Affinitization was not used in this suite of tests, all Ethernet interrupts were processed on CPUs designated by default /proc/irq/nnn/smp_affinity values that were altered by `irqbalance` at initialization. Sar data, which display such things as CPU utilization and interrupts per system CPU, show that all network interrupts were processed on CPUs on the first node regardless of whether or not the Ethernet adapter resided on any other node. This introduced unnecessary node hop latency. Shown below is a subset of sar data from the netserver scalability tests with `netserver` running on all Ethernet adapters on all four nodes. This collection of data is from the 8K message size test and is representative of all tests in this environment. The values are all averages over the course of the 3-minute run.

```
CPU     %user     %nice   %system   %iowait    %steal    %idle
 0      4.50      0.00     70.18      0.00      0.00     25.32
 1      1.89      0.00     88.54      0.00      0.00      9.57
 2      1.68      0.00     70.54      0.00      0.00     27.77
 3      0.66      0.00      4.81      0.00      0.00     94.53
 4      1.90      0.00     80.43      0.00      0.00     17.67
 5      2.44      0.00     82.38      0.00      0.00     15.18
 6      1.79      0.00     70.55      0.00      0.00     27.66
 7      0.55      0.00      5.40      0.00      0.00     94.04
 8      3.91      0.00     67.36      0.00      0.00     28.73
 9      1.66      0.00     82.23      0.00      0.00     16.11
10      0.21      0.00      4.37      0.00      0.00     95.43
11      0.35      0.00      5.53      0.00      0.00     94.12
12      0.13      0.00      1.97      0.00      0.00     97.90
13      0.10      0.00      1.88      0.00      0.00     98.03
14      0.09      0.00      2.14      0.09      0.00     97.68
15      0.07      0.00      1.86      0.89      0.00     97.18
 .
 .      (unlisted values are 0 or near 0)
 .

        eth0     eth1     eth2     eth3     eth4     eth5     eth6     eth7
CPU     i177/s   i185/s   i193/s   i201/s   i209/s   i217/s   i225/s   i233/s

 0   17542.70     0.00     0.00     0.00     0.00     0.00     0.00     0.00
 1       0.00     0.00     0.00     0.00     0.00 19515.86     0.00     0.00
 2       0.00     0.00     0.00 18612.58     0.00     0.00     0.00     0.00
 3       0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00
 4       0.00     0.00     0.00     0.00 18816.80     0.00     0.00     0.00
 5       0.00     0.00 18545.74     0.00     0.00     0.00     0.00     0.00
 6       0.00     0.00     0.00     0.00     0.00     0.00 18609.16     0.00
 7       0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00
 8       0.00 17506.10     0.00     0.00     0.00     0.00     0.00     0.00
 9       0.00     0.00     0.00     0.00     0.00     0.00     0.00 18606.14
 .
 .   (unlisted values are 0 or near 0)
 .
```

Even though throughput scaling was not particularly poor in this environment, CPU scaling drops off considerably as the number of nodes with utilized Ethernet adapters increases. The increasing CPU utilization is attributed to unnecessary node hops for interrupts being taken on Ethernet adapters on non-primary nodes but being processed on CPUs on the primary node. Data collected in other environments will show that total throughput in this environment suffered as well.

"Out of the Box" Configuration with `numactl`

The scalability tests and test methodology used in this environment are the same as in the "out of the box" configuration. The difference in environments is that this one used `numactl` to bind the `netperf` and/or `netserver` applications on the SUT to CPUs and memory on the appropriate nodes. Those bindings ensured that the applications would run on CPUs on the same nodes as the Ethernet adapters they were using. `numactl` is invoked as follows:

```
numactl --cpubind=node --preferred=node netserver
```

where cpubind=*node* specifies the node whose CPU should execute the process, and preferred=*node* specifies the node where memory for the process should be allocated. If the memory cannot be allocated on that node, it will be allocated on another node's memory.

The netserver, netperf, and bidirectional scalability tests were run and data were collected in the same way as in the "out of the box" configuration.

Figure 4 shows the network stream throughput and CPU utilization for the netserver scalability test runs while utilizing the system board Ethernet adapters on 1, 2 and 4 nodes of the SUT. The throughput shown is the sum throughput of all utilized Ethernet adapters for each test run, and CPU utilization shown is the system average for the duration of the each test run.

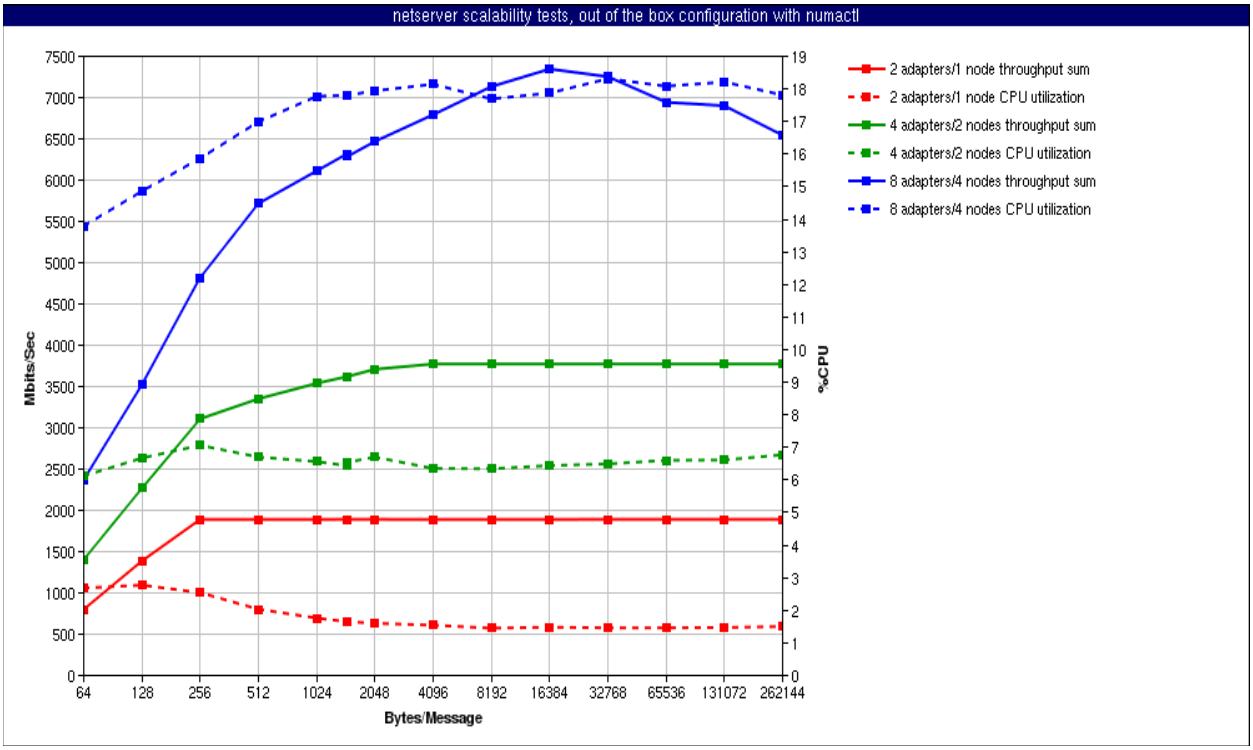

Figure 4. netserver on SUT in "out of the box" configuration with `numactl`

The netperf scalability tests were run just as they were in the previous environment. Figure 5 shows the network stream throughput and CPU utilization for the netperf scalability test runs while utilizing the system board Ethernet adapters on 1, 2 and 4 nodes of the SUT. The throughput shown is the sum throughput of all utilized Ethernet adapters for each test run, and CPU utilization shown is the system average for the duration of the each test run.

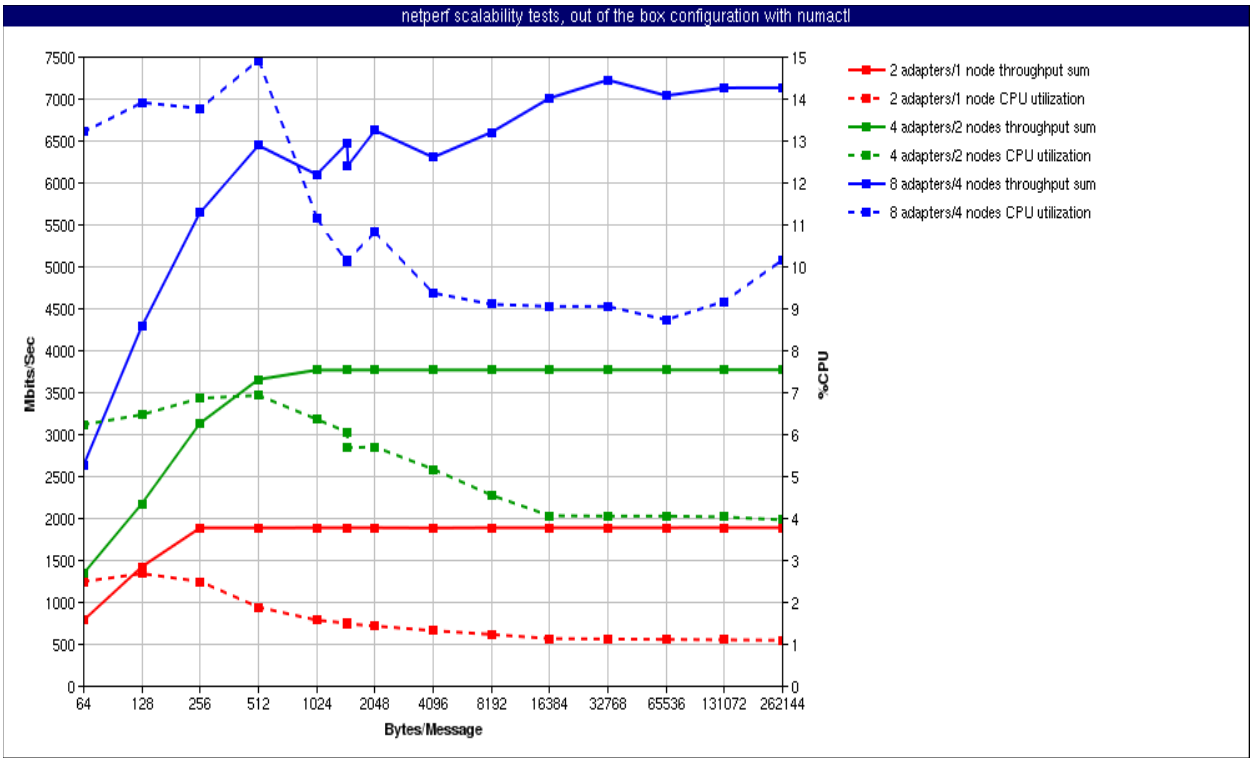

Figure 5. netperf on SUT in "out of the box" configuration with `numactl`

The bidirectional scalability tests were run just as they were in the previous environment. Figure 6 shows the network stream throughput and CPU utilization for the bidirectional scalability test runs while utilizing the system board Ethernet adapters on 1, 2 and 4 nodes of the SUT. The throughput shown is the sum throughput of all utilized Ethernet adapters for each test run, and CPU utilization shown is the system average for the duration of the each test run.

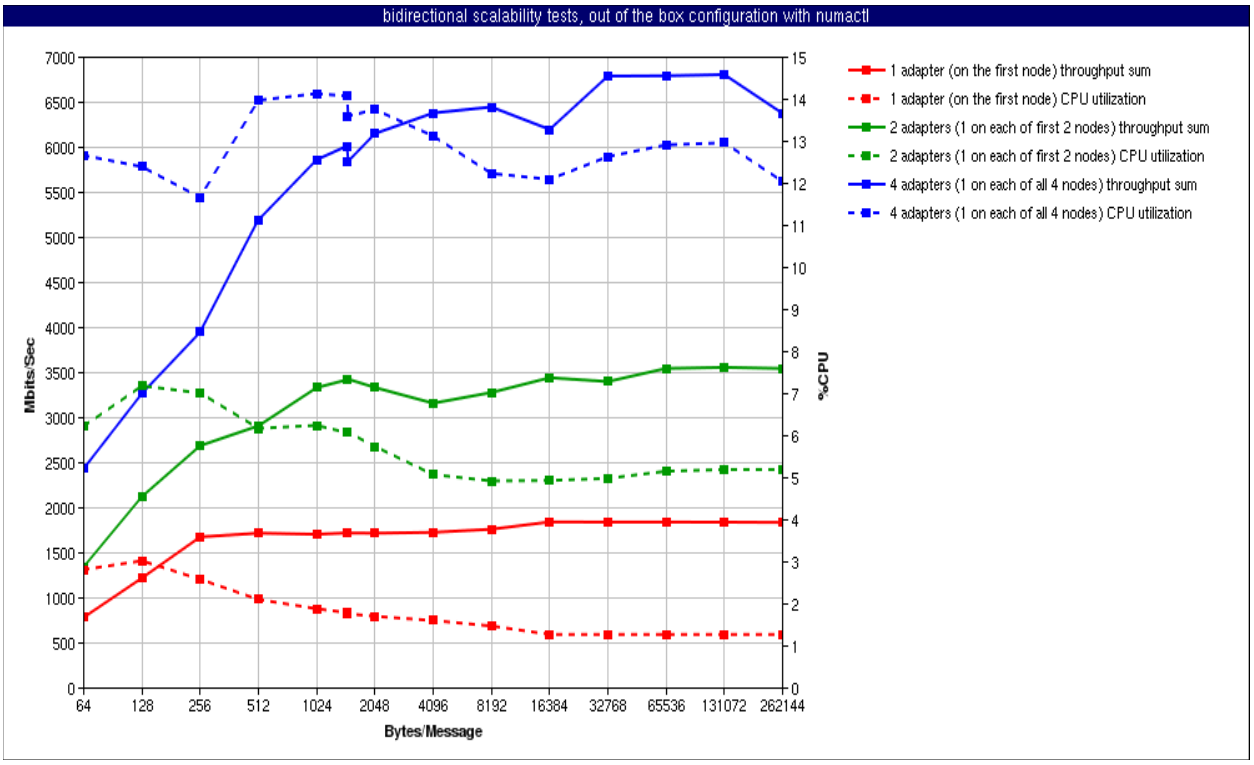

Figure 6. netperf and netserver (bidirectional) on SUT in "out of the box" configuration with `numactl`

Similar to the "out of the box" tests, the scaling computations were made and averaged for the netserver, netperf, and bidirectional tests when scaling from using Ethernet adapters on 1 to 2 nodes, 2 to 4 nodes, and 1 to 4 nodes. The results for this environment are shown in the following table.

SUT Scalability Test	Average Throughput Scaling from 1 to 2 Nodes	Average Throughput Scaling from 2 to 4 Nodes	Average Throughput Scaling from 1 to 4 Nodes	Avg. CPU Utilization Scaling from 1 to 2 Nodes	Avg. CPU Utilization Scaling from 2 to 4 Nodes	Avg. CPU Utilization Scaling from 1 to 4 Nodes
netserver	1.900	1.763	3.362	3.860	2.639	10.332
netperf	1.923	1.807	3.466	3.555	2.056	7.268
bidirectional	1.859	1.807	3.365	3.371	2.288	7.831

As in the previous tests, since SMP IRQ Affinitization was not used in this suite of tests, all Ethernet interrupts were processed on CPUs designated by default /proc/irq/nnn/smp_affinity values that were altered by `irqbalance`. Sar data show that all interrupts were processed on CPUs on the first node regardless of whether or not the Ethernet adapter resided on any other node, even if the application was bound by `numactl` to CPUs and memory on the node of its utilized Ethernet adapter. In fact, binding `netperf` and/or `netserver` to CPUs on the node local to its utilized ethernet adapter while that adapter's interrupts were processed on a different node caused a significant increase in overall CPU utilization.

Shown below is a subset of sar data from the netserver scalability tests with `netserver` running on all Ethernet adapters on all four nodes. This collection of data is from the 8K message size test and is representative of all tests in this environment. The values are all averages over the course of the 3-minute run.

```
CPU     %user     %nice   %system   %iowait    %steal    %idle
 0      3.48      0.00     79.71      0.00      0.00     16.81
 1      0.03      0.00     73.55      0.00      0.00     26.42
 2      0.02      0.00     67.80      0.00      0.00     32.18
 3      0.09      0.00      0.08      0.00      0.00     99.83
 4      0.00      0.00     73.59      0.00      0.00     26.41
 5      0.03      0.00     73.14      0.00      0.00     26.83
 6      0.01      0.00     62.52      0.00      0.00     37.47
 7      0.04      0.00      0.07      0.00      0.00     99.89
 8      3.80      0.00     71.70      0.00      0.00     24.51
 9      0.02      0.00     68.80      0.00      0.00     31.19
 .
17      0.69      0.00     92.92      0.00      0.00      6.39
18      0.01      0.00      0.00      0.00      0.00     99.99
19      0.75      0.00     92.90      0.00      0.00      6.36
 .
32      0.77      0.00     93.60      0.00      0.00      5.63
 .
43      0.80      0.00     93.57      0.00      0.00      5.63
 .
49      0.76      0.00     92.91      0.00      0.00      6.34
 .
63      0.35      0.00     93.31      0.00      0.00      6.35


        eth0     eth1     eth2     eth3     eth4     eth5     eth6     eth7
CPU     i177/s   i185/s   i193/s   i201/s   i209/s   i217/s   i225/s   i233/s

0    16990.40     0.00     0.00     0.00     0.00     0.00     0.00     0.00
1        0.00     0.00 11568.49     0.00     0.00     0.00     0.00     0.00
2        0.00     0.00     0.00 13777.11     0.00     0.00     0.00     0.00
3        0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00
4        0.00     0.00     0.00     0.00 10657.14     0.00     0.00     0.00
5        0.00     0.00     0.00     0.00     0.00 10653.40     0.00     0.00
6        0.00     0.00     0.00     0.00     0.00     0.00 13474.30     0.00
7        0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00
8        0.00 16630.13     0.00     0.00     0.00     0.00     0.00     0.00
9        0.00     0.00     0.00     0.00     0.00     0.00     0.00 12092.25
.
.    (unlisted values are 0 or near 0)
.
```

Processing network interrupts on nodes remote to where the Ethernet adapter resides is certainly not optimal. In that environment, binding network applications with numactl to nodes where the Ethernet adapters reside makes matters worse. Throughput, CPU utilization, and overall scaling suffer as is indicated by the data collected in this and the previous environment.

Ethernet SMP IRQ Affinitization

The scalability tests and test methodology used in this environment are the same as in the “out of the box” configuration. The difference in environments is that this one had the interrupt processing for each Ethernet adapter bound to a CPU on the node in which the adapter resides. Also, `irqbalance` was not used in this configuration.

Binding interrupt processing to a CPU or group of CPUs (affinitizing) is done by manipulating the smp_affinity bitmask for a given interrupt number. This bitmask represents which CPUs should process a given interrupt. When affinitizing interrupts, the `irqbalance` daemon should be terminated first or it will iteratively alter the smp_affinity value. If that happens, affinitization will be nullified, and interrupt processing for an Ethernet adapter may not take place on the intended CPU. In fact, interrupt processing for an Ethernet adapter on one node could switch to a CPU on another node. To terminate `irqbalance`, issue:

```
killproc –TERM /usr/sbin/irqbalance
```

and remove it from boot initialization scripts.

To bind an interrupt to a CPU or group of CPUs, first determine which CPUs should process the interrupt and lay out the bitmask accordingly. The rightmost bit of the mask is set for CPU0, the next for CPU1, and so on. Multiple bits can be set to indicate a group of CPUs. Then set the bitmask value appropriately by invoking the following command:

```
echo bitmask > /proc/irq/IRQ_number/smp_affinity
```

For example, to bind processing of IRQ number 177 to CPUs 4 through 7 (bitmask 11110000), issue:

```
echo f0 > /proc/irq/177/smp_affinity
```

Note that in this study, when setting smp_affinity to a bitmask of more than one CPU, the observed behavior was that the networking interrupts were always processed on the first CPU indicated in the bitmask. If two interrupts’ bitmasks had the same first bit set, then both interrupts were processed on the same CPU indicated by the first set bit. For example, when two Ethernet adapter interrupts both had smp_affinity bitmask of 0000ffff, both were processed on CPU0. Thus, at this point, it may not be wise to overlap smp_affinity bitmasks among Ethernet adapter interrupts unless the intent is to have them processed on the same CPU.

The smp_affinity bitmask values for this test were set as follows:

```
first ethernet adapter on first node:     00000000,000000f0
second ethernet adapter on first node:    00000000,0000f000
first ethernet adapter on second node:    00000000,00f00000
second ethernet adapter on second node:   00000000,f0000000
first ethernet adapter on third node:     000000f0,00000000
second ethernet adapter on third node:    0000f000,00000000
first ethernet adapter on fourth node:    00f00000,00000000
second ethernet adapter on fourth node:   f0000000,00000000
```

Those settings ensured that each Ethernet adapter had its interrupts processed on a CPU on the node on which the adapter resided. CPU 0 was intentionally left free of networking interrupts since it is typically heavily used in many workloads.

The netserver, netperf, and bidirectional scalability tests were run, and data were collected in the same way as in the “out of the box” configuration. Figure 7 shows the network stream throughput and CPU utilization for the netserver scalability test runs while utilizing the system board Ethernet adapters on 1, 2 and 4 nodes of the SUT. The throughput shown is the sum throughput of all utilized Ethernet adapters for each test run, and CPU utilization shown is the system average for the duration of the each test run.

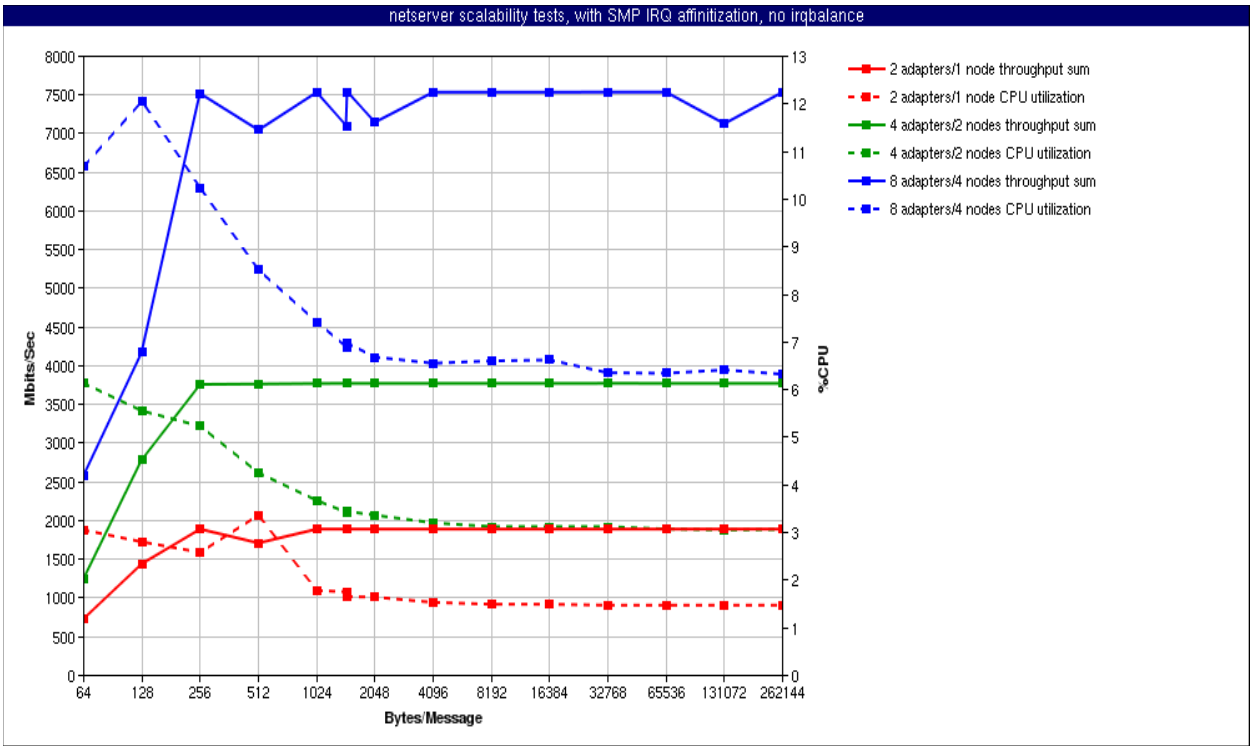

Figure 7. netserver on SUT with Ethernet SMP IRQ affinity, no `irqbalance`

The netperf scalability tests were run just as they were in the previous environment. Figure 8 shows the network stream throughput and CPU utilization for the netperf scalability test runs while utilizing the system board Ethernet adapters on 1, 2 and 4 nodes of the SUT. The throughput shown is the sum throughput of all utilized Ethernet adapters for each test run, and CPU utilization shown is the system average for the duration of the each test run.

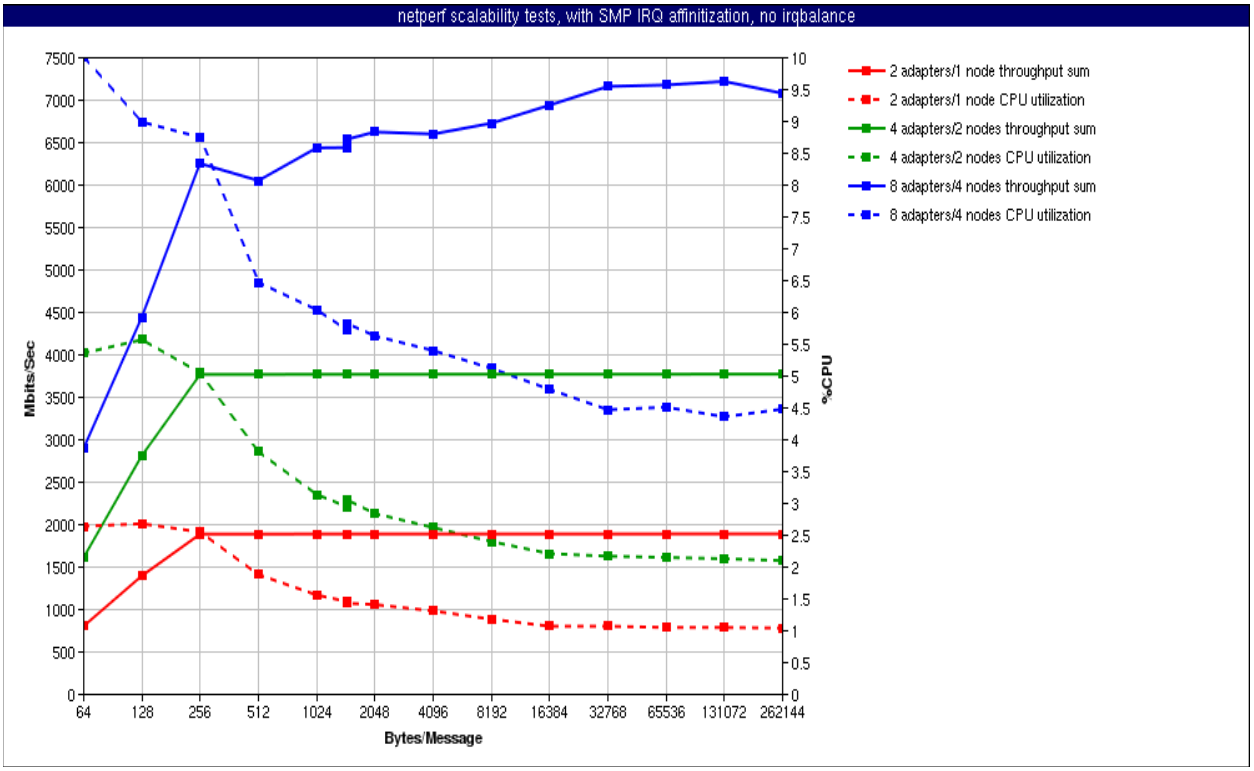

Figure 8. netperf on SUT with Ethernet SMP IRQ affinity, no `irqbalance`

The bidirectional scalability tests were run just as they were in the previous environment. Figure 9 shows the network stream throughput and CPU utilization for the bidirectional netserver scalability test runs while utilizing the system board Ethernet adapters on 1, 2 and 4 nodes of the SUT. The throughput shown is the sum throughput of all utilized Ethernet adapters for each test run, and CPU utilization shown is the system average for the duration of the each test run.

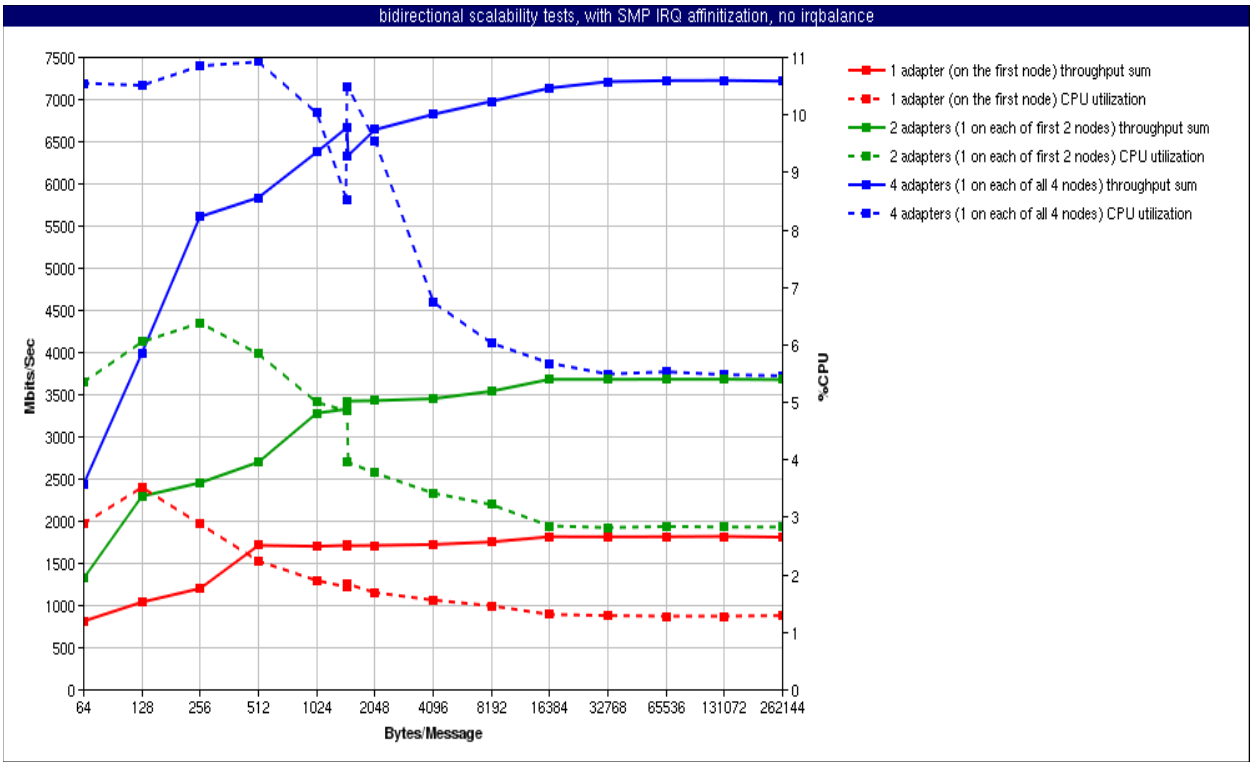

Figure 9. netperf and netserver (bidirectional) on SUT with Ethernet SMP IRQ affinity, no `irqbalance`

Similar to the “out of the box” tests, the scaling computations were made and averaged for the netserver, netperf, and bidirectional tests from 1 to 2 nodes, 2 to 4 nodes, and 1 to 8 nodes. The results for this environment are shown in the following table.

SUT Scalability Test	Average Throughput Scaling from 1 to 2 Nodes	Average Throughput Scaling from 2 to 4 Nodes	Average Throughput Scaling from 1 to 4 Nodes	Avg. CPU Utilization Scaling from 1 to 2 Nodes	Avg. CPU Utilization Scaling from 2 to 4 Nodes	Avg. CPU Utilization Scaling from 1 to 4 Nodes
netserver	1.990	1.941	3.861	2.011	2.035	4.095
netperf	2.002	1.770	3.542	2.042	1.961	4.005
bidirectional	1.971	1.968	3.874	2.241	1.990	4.456

Shown below is a subset of sar data from the netserver scalability tests with `netserver` running on all Ethernet adapters on all four nodes. This collection of data is from the 8K message size test and is representative of all tests in this environment. The values are all averages over the course of the 3-minute run. The data show that all interrupts were processed nicely on the CPUs to which they were bound via SMP IRQ affinitization.

```
CPU     %user     %nice   %system   %iowait    %steal    %idle
 0      0.05      0.00      0.05      0.00      0.00     99.90
 .
 4      2.22      0.00     49.21      0.00      0.00     48.57
 .
12      2.21      0.00     49.27      0.02      0.00     48.51
 .
20      2.25      0.00     51.59      0.00      0.00     46.17
 .
28      2.23      0.00     51.47      0.00      0.00     46.29
 .
36      2.86      0.00     56.06      0.00      0.00     41.08
 .
44      2.29      0.00     51.87      0.00      0.00     45.84
 .
52      2.69      0.00     55.28      0.00      0.00     42.02
 .
60      2.66      0.00     55.35      0.00      0.00     41.98
 .
 .      (unlisted values are 0 or near 0)
 .


        eth0     eth1     eth2     eth3     eth4     eth5     eth6     eth7
CPU     i177/s   i185/s   i193/s   i201/s   i209/s   i217/s   i225/s   i233/s

 4  15969.26     0.00     0.00     0.00     0.00     0.00     0.00     0.00
 .
12      0.00 15959.40     0.00     0.00     0.00     0.00     0.00     0.00
 .
20      0.00     0.00 15721.47     0.00     0.00     0.00     0.00     0.00
 .
28      0.00     0.00     0.00 15693.93     0.00     0.00     0.00     0.00
 .
36      0.00     0.00     0.00     0.00 16000.84     0.00     0.00     0.00
 .
44      0.00     0.00     0.00     0.00     0.00 15981.13     0.00     0.00
 .
52      0.00     0.00     0.00     0.00     0.00     0.00 15855.95     0.00
 .
60      0.00     0.00     0.00     0.00     0.00     0.00     0.00 15700.12
 .
 .   (unlisted values are 0 or near 0)
 .
```

Affinitizing Ethernet adapter interrupt processing to CPUs on their nodes (coupled with terminating irqbalance) greatly reduced CPU utilization increasing throughput and improving both throughput and CPU utilization scalability.

Ethernet SMP IRQ Affinitization with numactl

The scalability tests and test methodology used in this environment are the same as that used in the “out of the box” configuration. This test environment combined the features of the last two tests described in this paper. SMP IRQ affinity was enabled with the same bitmasks as in the last test, `irqbalance` was disabled, and `numactl` was used to bind `netperf` and/or `netserver` on the SUT to CPUs and memory on the appropriate nodes. Those `numactl` bindings ensured that the instances of the applications would run on CPUs on the same nodes as the Ethernet adapters they were using as well as use the memory on those nodes.

The netserver, netperf, and bidirectional scalability tests were run, and data were collected in the same way as in the “out of the box” configuration. Figure 10 shows the network stream throughput and CPU utilization for the netserver scalability test runs while utilizing the system board Ethernet adapters on 1, 2 and 4 nodes of the SUT. The throughput shown is the sum throughput of all utilized Ethernet adapters for each test run, and CPU utilization shown is the system average for the duration of the each test run.

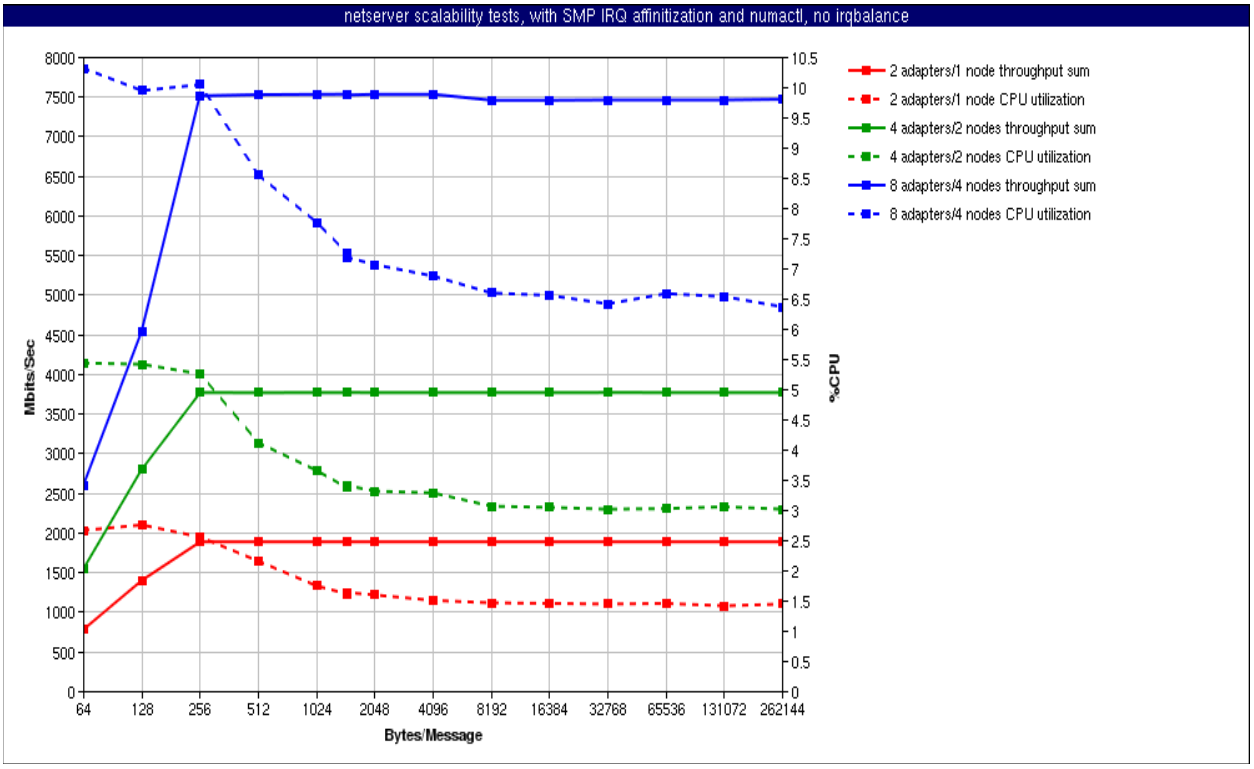

Figure 10. netserver on SUT with Ethernet SMP IRQ affinity and `numactl`, no `irqbalance`

The netperf scalability tests were run just as they were in the previous environment. Figure 11 shows the network stream throughput and CPU utilization for the netperf scalability test runs while utilizing the system board Ethernet adapters on 1, 2 and 4 nodes of the SUT. The throughput shown is the sum throughput of all utilized Ethernet adapters for each test run, and CPU utilization shown is the system average for the duration of the each test run.

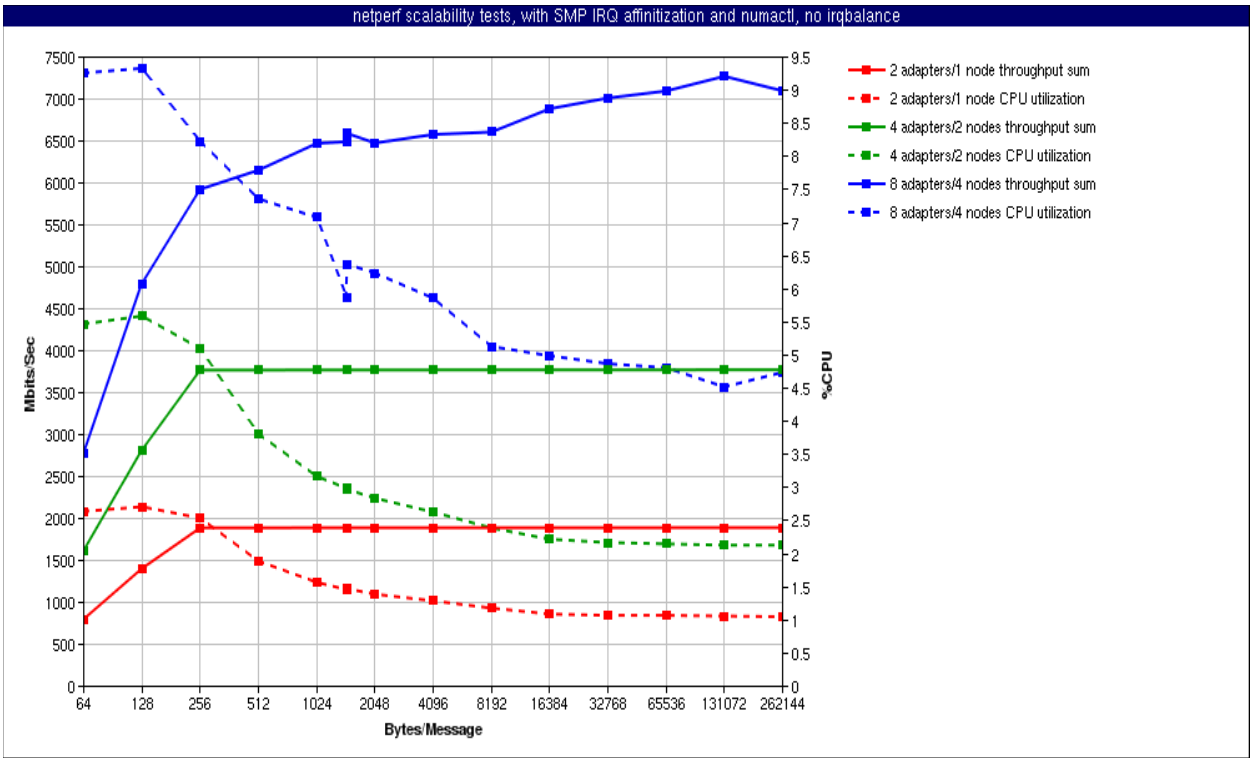

Figure 11. netperf on SUT with Ethernet SMP IRQ affinity and `numactl`, no `irqbalance`

The bidirectional scalability tests were run just as they were in the previous environment. Figure 12 shows the network stream throughput and CPU utilization for the bidirectional scalability test runs while utilizing the system board Ethernet adapters on 1, 2 and 4 nodes of the SUT. The throughput shown is the sum throughput of all utilized Ethernet adapters for each test run, and CPU utilization shown is the system average for the duration of the each test run.

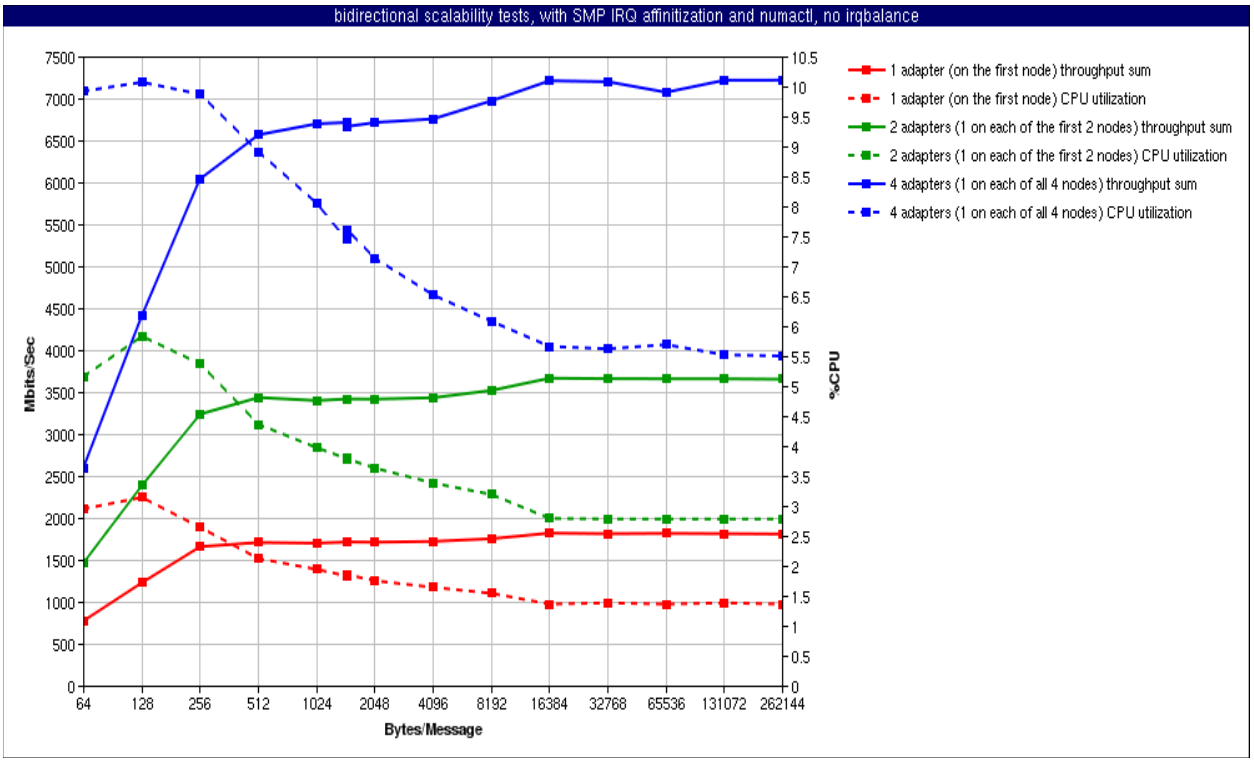

Figure 12. netperf and netserver (bidirectional) on SUT with Ethernet SMP IRQ affinity and `numactl`, no `irqbalance`

Similar to the “out of the box” tests, the scaling computations were made and averaged for the netserver, netperf, and bidirectional tests when scaling from using Ethernet adapters on 1 to 2 nodes, 2 to 4 nodes, and 1 to 4 nodes. The results for this environment are shown in the following table.

SUT Scalability Test	Average Throughput Scaling from 1 to 2 Nodes	Average Throughput Scaling from 2 to 4 Nodes	Average Throughput Scaling from 1 to 4 Nodes	Avg. CPU Utilization Scaling from 1 to 2 Nodes	Avg. CPU Utilization Scaling from 2 to 4 Nodes	Avg. CPU Utilization Scaling from 1 to 4 Nodes
netserver	1.999	1.945	3.888	2.076	2.081	4.324
netperf	2.002	1.762	3.527	2.038	2.064	4.206
bidirectional	1.993	1.935	3.858	2.016	1.960	3.953

Shown below is subset of sar data from the netserver scalability tests with `netserver` running on all Ethernet adapters on all four nodes. This collection of data is from the 8K message size test and is representative of all tests in this environment. The values are all averages over the course of the 3-minute run. The data show that all interrupts were processed on the CPUs to which they were bound via affinitization.

```
CPU     %user     %nice   %system   %iowait    %steal     %idle

 4      2.25      0.00     49.29      0.03      0.00     48.43
 .
12      2.61      0.00     51.82      0.00      0.00     45.58
 .
20      2.25      0.00     51.84      0.00      0.00     45.91
 .
28      2.85      0.00     57.43      0.00      0.00     39.72
 .
36      2.06      0.00     50.21      0.00      0.00     47.72
 .
44      2.04      0.00     53.34      0.00      0.00     44.62
 .
52      2.03      0.00     52.34      0.00      0.00     45.62
 .
60      2.28      0.00     54.28      0.00      0.00     43.44
 .
 .      (unlisted values are 0 or near 0)
 .


        eth0     eth1     eth2     eth3     eth4     eth5     eth6     eth7
CPU     i177/s   i185/s   i193/s   i201/s   i209/s   i217/s   i225/s   i233/s

 4   15962.89     0.00     0.00     0.00     0.00     0.00     0.00     0.00
 .
12       0.00 15660.56     0.00     0.00     0.00     0.00     0.00     0.00
 .
20       0.00     0.00 15690.78     0.00     0.00     0.00     0.00     0.00
 .
28       0.00     0.00     0.00 15696.00     0.00     0.00     0.00     0.00
 .
36       0.00     0.00     0.00     0.00 15726.10     0.00     0.00     0.00
 .
44       0.00     0.00     0.00     0.00     0.00 15574.72     0.00     0.00
 .
52       0.00     0.00     0.00     0.00     0.00     0.00 15682.12     0.00
 .
60       0.00     0.00     0.00     0.00     0.00     0.00     0.00 15700.46
 .
 .   (unlisted values are 0 or near 0)
 .
```

As was shown in the previous tests, affinitizing Ethernet adapter interrupt processing to CPUs on their nodes (coupled with terminating `irqbalance`) greatly reduced CPU utilization while increasing throughput and scalability. Further, using `numactl` to bind the networking application to a CPU and memory on the same node as the Ethernet adapter it is using provides a slight benefit in throughput, CPU utilization, and scalability.

Ethernet Bonding

Having one IP address for some or all of the Ethernet adapters in a large multi-node system can be beneficial for system administrators and network administration. Bonding is a feature included with Linux that, as stated earlier, provides a variety of methods for aggregating multiple network interfaces into a single logical interface. The bonding driver and supporting utility `ifenslave` are included with most Linux distributions. The driver is highly configurable and has six bonding policies to balance traffic across bonded interfaces. These policies include balance-rr (round-robin), adaptive-backup, balance-xor, broadcast, 802.3ad, balance-tlb (transmit load balancing), and balance-alb (adaptive load balancing). The study presented here concentrated on the balance-alb policy, sometimes called “mode 6,” because it is easy to set up, requires no additional hardware or switch configuration, and balances the load of both sends and receives across the bonded interfaces.

To set up bonding on the SUT, the following steps were performed:

1. Load the bonding module with adaptive load balancing (mode 6) and updelay (milliseconds to wait after link recovery before enabling a slave):

```
modprobe bonding mode=6 updelay=200
```

2. Configure the bond interface and bring it up:

```
ifconfig bond0 ip_address netmask netmask broadcast bcast
```

3. Attach all interfaces for bonding to the bond interface:

```
ifenslave bond0 eth0 eth1 eth2 eth3
```

To demonstrate bonding performance and scalability, the netserver scalability tests were run with `irqbalance` disabled and Ethernet SMP IRQ affinity set appropriately as in the last test. The Ethernet adapters being utilized were bonded into one interface with one IP address, and each remote instance of netperf on the drivers sent messages to the IP address of the bonded interface.

The netserver scalability tests were run, and data were collected in the same way as in the “out of the box” configuration except that only one Ethernet adapter per node was utilized. Figure 13 shows the network stream throughput and CPU utilization for the netserver scalability test runs while utilizing the bonded first system board Ethernet adapters on 1, 2 and 4 nodes of the SUT. The throughput shown is the sum throughput of all utilized Ethernet adapters for each test run, and CPU utilization shown is the system average for the duration of the each test run.

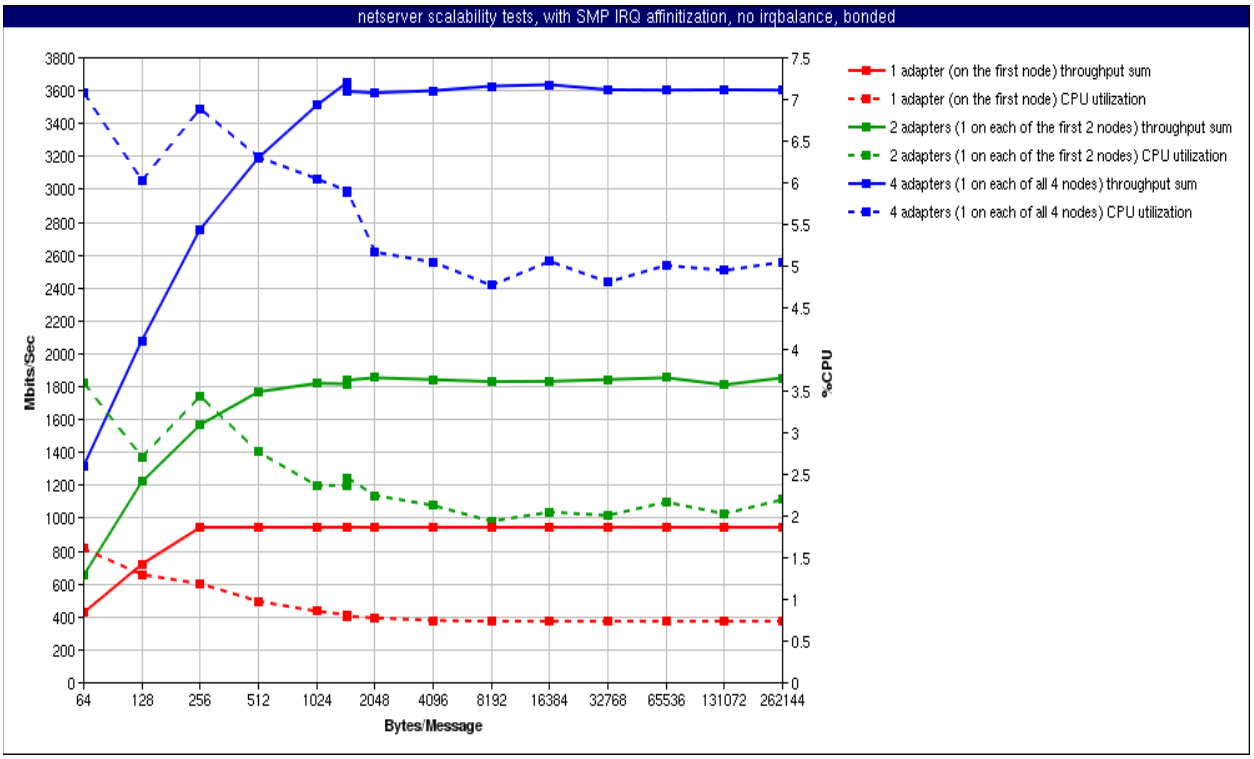

Figure 13. netserver on SUT with Ethernet SMP IRQ affinity, no `irqbalance`, bonded interfaces

Similar to the “out of the box” tests, the scaling computations were made and averaged for the netserver, netperf, and bidirectional tests when scaling from using the first Ethernet adapters on 1 to 2 nodes, 2 to 4 nodes, and 1 to 4 nodes. The results for this environment are shown in the following table.

SUT Scalability Test	Average Throughput Scaling from 1 to 2 Nodes	Average Throughput Scaling from 2 to 4 Nodes	Average Throughput Scaling from 1 to 4 Nodes	Avg. CPU Utilization Scaling from 1 to 2 Nodes	Avg. CPU Utilization Scaling from 2 to 4 Nodes	Avg. CPU Utilization Scaling from 1 to 4 Nodes
netserver	1.879	1.926	3.624	2.771	2.335	6.486

Shown below is a subset of sar data from the netserver scalability tests with `netserver` running on the bonded interface, which is the aggregate of the first system board Ethernet adapter on each of the four nodes. This collection of data is from the 8K message size test and is representative of all tests in this environment. The values are all averages over the course of the 3-minute run. The data show that all interrupts were processed nicely on the CPUs to which they were bound.

```
CPU     %user     %nice   %system   %iowait    %steal    %idle

 4      2.02      0.00     72.32      0.00      0.00     25.67
 .
20      2.37      0.00     77.83      0.00      0.00     19.80
 .
36      1.72      0.00     74.71      0.00      0.00     23.57
 .
52      1.62      0.00     78.48      0.00      0.00     19.89
 .
 .      (unlisted values are 0 or near 0)
 .



        eth0     eth1     eth2     eth3     eth4     eth5     eth6     eth7
CPU     i177/s   i185/s   i193/s   i201/s   i209/s   i217/s   i225/s   i233/s

 4   16353.36     0.00     0.00     0.00     0.00     0.00     0.00     0.00
 .
20       0.00     0.00 15693.97     0.00     0.00     0.00     0.00     0.00
 .
36       0.00     0.00     0.00     0.00 15386.92     0.00     0.00     0.00
 .
52       0.00     0.00     0.00     0.00     0.00     0.00 15570.74     0.00
 .
 .   (unlisted values are 0 or near 0)
 .
```

The following graphs show a comparison of the netserver scalability tests when utilizing 2 adapters (1 on each of the first 2 nodes) and 4 adapters (1 on each of all 4 nodes) with and without bonding. The throughput shown is the sum throughput of all utilized Ethernet adapters for each test run, and CPU utilization shown is the system average for the duration of the each test run.

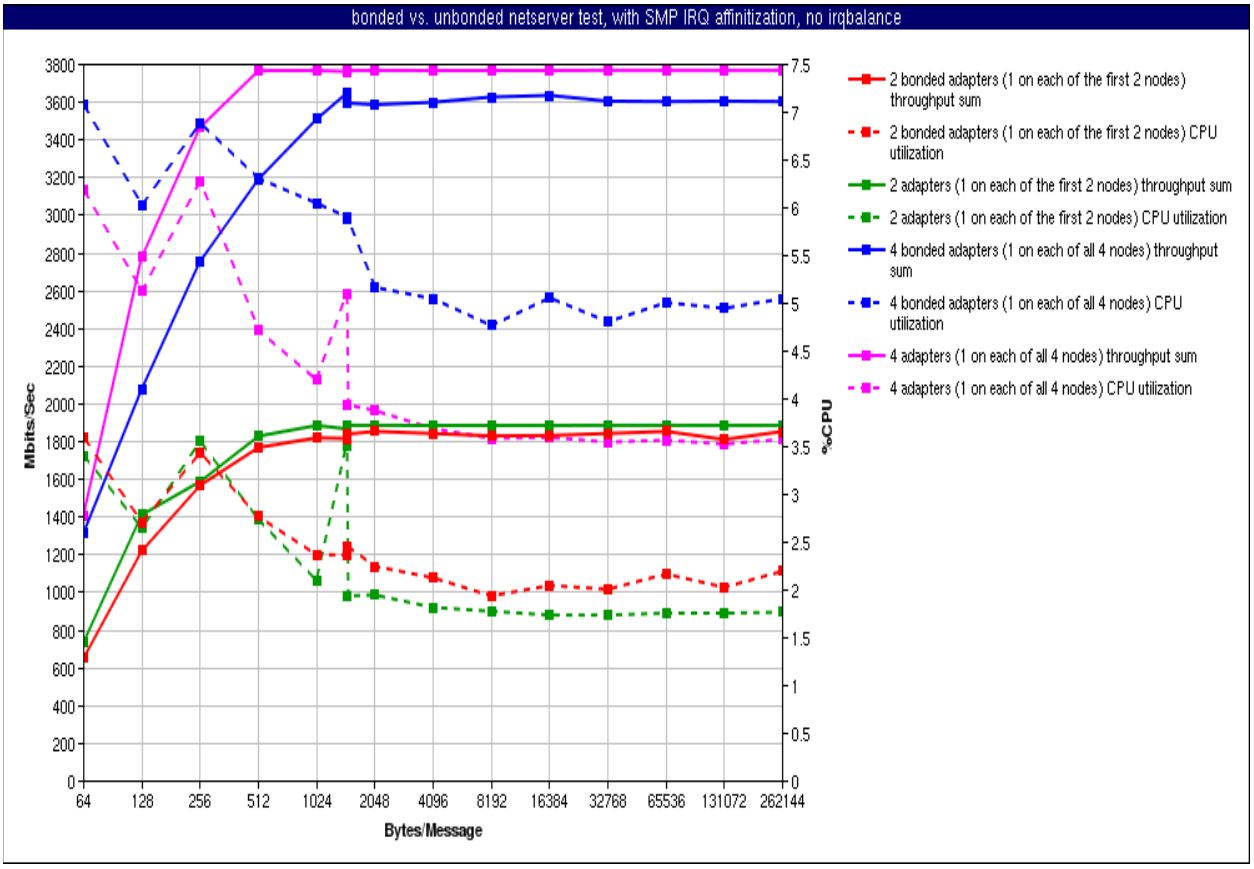

Figure 14. netserver on SUT with Ethernet SMP IRQ affinity, no `irqbalance`, with and without bonding

While there is some overhead associated with Ethernet bonding, the tests described above show that networking over bonded Ethernet interfaces scales well and performs well relative to networking over Ethernet interfaces that are not bonded. The administrative benefits and potential for networking application simplification enabled by bonding may outweigh its performance costs.

Conclusions

When using multiple network adapters across nodes of a high-performance scalable server, the default Linux “out of the box” configuration may not be the best for optimal performance and scalability. In the environment described in this paper, the default Ethernet adapter interrupt processing took place on CPUs on the first node regardless of the node where the adapter actually resided. This behavior degraded networking throughput, CPU utilization, and overall networking performance and scalability. The improper configuration unnecessarily increases CPU utilization, which negatively impacts overall system performance.

For best networking performance and scalability, ensure that Ethernet adapter interrupts are processed on CPUs on the adapters’ local nodes. Binding interrupt processing to appropriate CPUs (affinitizing) can be accomplished by first terminating `irqbalance`, removing it from initialization scripts, properly enabling SMP IRQ affinitization, and placing affinitization configuration in the boot initialization scripts. Affinitizing without first terminating `irqbalance` nullifies affinitizing. When SMP IRQ affinity has been successfully configured, then, if possible, bind the networking applications to the processors that are on the local nodes of the Ethernet adapters being used.

Ethernet bonding is a useful feature in Linux that provides a variety of methods for aggregating multiple network interfaces into a single logical interface. The relatively low overhead cost may far outweigh the administrative benefits for network organization.

The test results presented in this paper show overall Linux networking scalability to be quite good when Ethernet adapters are used across nodes on a properly configured IBM eServer xSeries x460 system. Average throughput scaling over multiple send message sizes is up to 1.999 when moving from utilizing 2 Ethernet adapters on 1 node to 4 adapters on 2 nodes, and is up to 1.945 when moving from utilizing 4 Ethernet adapters on 2 nodes to 8 adapters on 4 nodes. The corresponding average CPU utilization scaling is 2.076 when moving from utilizing 2 Ethernet adapters on 1 node to 4 adapters on 2 nodes, and is 2.081 when moving from utilizing 4 Ethernet adapters on 2 nodes to 8 adapters on 4 nodes.

Additional Information

IBM System x
x86 servers for Windows and Linux
Product details, data sheets, papers, and more.
http://www-03.ibm.com/systems/x/index.html

Tuning IBM System x Servers for Performance
IBM Redbooks®
http://www.redbooks.ibm.com/abstracts/sg245287.html

netperf Homepage
http://www.netperf.org/

SYSSTAT Utilities Homepage
http://perso.orange.fr/sebastien.godard/

IRQBALANCE Homepage
handholding your interrupts for power and performance
http://www.irqbalance.org/

Bonding - LinuxNet
http://linux-net.osdl.org/index.php/Bonding

IBM Systems and Technology Group
3039 Cornwallis Road
Research Triangle Park, NC 27709

Produced in the USA
04-07