A Concurrent Real-Time White Paper

2881 Gateway Drive
Pompano Beach, FL 33069
(954) 974-1700
real-time.ccur.com

Comparison of the real-time network performance of RedHawk™ Linux® 6.0 and Red Hat® operating systems

By: Joe Korty
Concurrent Senior Consulting Engineer

March 2012

Goal

This paper asks which operating system, RedHawk 6.0, Red Hat RHEL 6 or Red Hat MRG 2.1, is better at real-time networking performance when various network controllers are used.

Setup

Three different network controller pairs were tested on two identical Intel® Core™ i7 systems located in Concurrent's Software Development Lab. The three network controller pairs were each directly connected via a single 14' cat6e Ethernet cable.

- Two Solarflare® SFN5121T 10Gigabit Ethernet PCIe cards supporting OpenOnload®
- Two Intel X520-T2 10Gigabit Ethernet PCIe cards
- Two on-board Realtek® RTL8111/8168B GigE controllers

The Solarflare SFN5121T is a specialized controller optimized for real-time performance. The SFN5121T has the option of being run with a separate userspace TCP/IP stack called OpenOnload. The card was tested both with and without the OpenOnload feature.

Results

In the Detailed Test Results table below, RTT/2 is the transit time and stdev (standard deviation) is a measure of jitter. Real-time performance requires that the stdev be as small as possible.

When OpenOnload is not used, as in the case of a commonly available 1 Gbit or 10 Gbit Ethernet card, RedHawk's shielding implementation provides a substantial improvement in jitter over an equivalent pseudo-shielding setup on RHEL and MRG, especially when under load. (See green highlights below.) In addition, the average transit time (RTT/2) is approximately 20% lower using RedHawk.

OpenOnload greatly improves both transit time and the jitter for all operating systems. No other test case achieves the overall performance of the Solarflare card with OpenOnload.

OpenOnload improves networking performance about equally for all operating systems, however RedHawk Linux with OpenOnload has lower jitter than both RHEL and MRG. When a load is present, the jitter under OpenOnload is more than 50% higher under RHEL and three to four times higher under MRG when compared to RedHawk. (See yellow highlights below.) OpenOnload transit times themselves are about the same for both operating systems.

Conclusions

RedHawk Linux offers a 20% throughput improvement and significant determinism advantages over Red Hat in non-OpenOnload testing. Under OpenOnload, all three operating systems yield similar transit times with RHEL and MRG displaying 50% or more jitter under load. Detailed test results are provided in the table below.

Detailed Test Results

(All values are microseconds.)

NIC	Back-ground Load	CPU Shielding	OS	netperf		pingpong			
				tcp	udp	tcp		udp	
				RTT/2	RTT/2	RTT/2	jitter stdev	RTT/2	jitter stdev
Solarflare SFN5121T 10GBit Ethernet **without** OpenOnload	No	N/A	RHEL	17.0	14.1	17.0	1.794	13.9	.336
			MRG	19.5	15.7	18.6	1.074	15.7	.416
			RedHawk	12.4	10.7	12.5	.283	10.6	.243
	Yes	No	RHEL	28.5	23.7	28.8	34.501	22.9	7.236
			MRG	33.5	26.9	32.1	31.295	26.8	19.900
			RedHawk	22.8	19.4	23.3	1.699	18.5	1.388
	Yes	Yes	RHEL	27.5	22.3	26.1	4.103	21.1	4.430
			MRG	24.1	20.1	24.1	5.510	20.1	2.662
			RedHawk	20.3	16.4	20.8	1.788	16.1	1.103
Solarflare SFN5121T 10GBit Ethernet **with** OpenOnload	No	N/A	RHEL	6.5	6.3	6.4	.204	6.2	.173
			MRG	6.8	6.5	6.8	.415	6.5	.431
			RedHawk	6.5	6.3	6.4	.204	6.2	.174
	Yes	No	RHEL	7.3	6.9	7.1	.502	6.9	.450
			MRG	14.7	12.0	13.6	66.102	10.4	51.631
			RedHawk	7.0	6.5	6.9	.517	6.5	.287
	Yes	Yes	RHEL	7.5	7.0	7.4	.478	6.7	.388
			MRG	7.6	7.1	7.5	1.056	7.1	1.020
			RedHawk	7.4	7.0	7.4	.306	7.0	.253
Intel X520-T2 10GBit Ethernet	No	N/A	RHEL	19.3	17.7	19.2	.787	17.6	.792
			MRG	22.4	20.4	22.3	1.122	24.6	1.033
			RedHawk	17.0	16.9	16.9	.610	16.0	.424
	Yes	No	RHEL	32.6	29.2	32.5	9.791	29.2	9.354
			MRG	38.6	33.7	37.4	26.920	34.1	23.359
			RedHawk	26.7	26.6	26.6	1.981	24.2	1.521
	Yes	Yes	RHEL	30.5	27.0	30.5	2.045	26.4	1.884
			MRG	29.4	21.9	28.8	2.177	25.9	1.896
			RedHawk	25.1	24.8	24.8	1.856	23.1	1.570
Realtek RTL8111/8168B On-board 1Gbit Ethernet	No	N/A	RHEL	26.0	21.1	25.8	1.493	20.8	1.256
			MRG	31.6	29.0	31.5	1.642	28.5	1.745
			RedHawk	20.0	19.3	20.0	.628	19.3	.416
	Yes	No	RHEL	43.4	39.0	43.1	60.350	39.2	82.626
			MRG	48.8	45.4	47.8	25.509	44.8	24.480
			RedHawk	35.9	32.9	36.1	2.649	27.4	3.201
	Yes	Yes	RHEL	42.6	38.1	41.7	47.649	37.8	42.989
			MRG	39.4	34.1	39.2	2.992	33.8	2.336
			RedHawk	31.8	26.0	25.6	1.962	26.3	2.964

Test Definitions

'netperf' is a widely available test for measuring network performance. It does not measure standard deviation. RTT/2 can be calculated from its printed results.

'pingpong' is a network performance test written by Solarflare Corporation. It displays RTT/2 times and standard deviations for a variety of packet sizes. In our tests, we follow the Solarflare convention and report only the results of the 32-byte packet size test.

'RTT/2' is the transit time or the round trip time divided by two. It is an industry standard way of measuring the time it takes to move data from an application on one system to another application on another system.

'shield'. For RedHawk, 'shield -a 0,1' was used. For RHEL6, the internally developed 'cpuguard' script was used to move away all processes and interrupts from cpu 0 and 1 via a variety of techniques. Whether or not shielding is enabled, all tests were run on either cpu 0 or cpu 1.

'stdev' is the standard deviation. This number measures how likely a particular RTT/2 measurement will deviate from the mean. A value of 0 means that all RTT/2 times are the same for a test run (no deviation). Conversely, the larger the standard deviation is, the more likely any particular RTT/2 measurement will deviate significantly from the mean. Real-time requires that this number be as small as possible.

'load'. For the load test, both systems were made to run 'make clear; make -j11' in an infinite loop. This load is slightly non-uniform because there are various places in the build (such as the final link edit step) where the load is significantly less than at other places (such as where every cpu is involved with one gcc(1) invocation or another). To work around this, the load was restarted at various standardized spots in the testing sequence, so that an approximately equivalent load was seen at equivalent spots in testing.

'OpenOnload' is a userspace implementation of the TCP/IP stack. It works only with Solarflare cards.

Test Preparation

Some mix of the following commands was used at various points during the testing process.

1) Make networking run with as few problems as possible:

```
service cpuspeed stop; service irqbalance stop; service iptables stop
ethtool -C eth5 rx-usecs-irq 0 adaptive-rx off
```

2) Commands that shield cpus 0 and 1 from general purpose traffic:

```
shield -a 0,1
/tmp/cpuguard 0,1
```

3) Force the Ethernet interrupts of interest to go to the shielded cpus:

```
for i in $('grep eth[12345] /proc/interrupts | sed s/:.*//); do
        echo 3 >/proc/irq/$i/smp_affinity; done
```

4) Commands used to run various tests, as client or server, on various systems:

```
pkill -f netserver; run -s fifo -P 1 -b 1 netserver
run -s fifo -P 1 -b 1 netperf -t TCP_RR -H 10.134.33.100 -l 10 -- -r 32
run -s fifo -P 1 -b 1 netperf -t UDP_RR -H 10.134.33.100 -l 10 -- -r 32

run -s fifo -P 1 ./sfnt-pingpong
run -s fifo -P 1 ./sfnt-pingpong --maxms=10000 -- tcp 10.134.33.100
run -s fifo -P 1 ./sfnt-pingpong --maxms=10000 -- udp 10.134.33.100
```

(the .33. net is a temporary subnet that consists of just the two Ethernet cards under test).

5) The prefix 'onload --profile=latency' is added to all of the above test invocation commands when that test is to use the OpenOnload userspace TCP/IP library. For example:

```
run -s fifo -P 1 -b 1 onload --profile=latency netserver
```

About Concurrent Real-Time

Concurrent Real-Time is a global leader in innovative solutions serving the aerospace and defense, automotive, and financial industries. As the industries' foremost provider of high-performance real-time computer systems, solutions, and software for commercial and government markets, Concurrent Real-Time focuses on hardware-in-the-loop and man-in-the-loop simulation, data acquisition, and industrial systems. Concurrent's Real-Time product group is located in Pompano Beach, Florida with additional offices in North America, Europe, Asia and Australia. For more information, please visit Concurrent Real-Time at www.real-time.ccur.com.

About Concurrent Computer Corporation

Concurrent (NASDAQ:CCUR) is a global leader in multi-screen video delivery, media data management and real-time computing solutions. Concurrent Real-Time is a global leader in innovative solutions serving the aerospace and defense, automotive, and financial industries. As the industries' foremost provider of high-performance real-time computer systems, solutions, and software for commercial and government markets, Concurrent Real-Time focuses on hardware-in-the-loop and man-in-the-loop simulation, data acquisition, and industrial systems. Concurrent is headquartered in Atlanta with offices in North America, Europe and Asia. Concurrent Real-Time is located in Pompano Beach, Florida. For more information, please visit Concurrent Real-Time at www.real-time.ccur.com.