



Red Hat Performance Briefs

Using Solarflare OpenOnload to Achieve Extreme Low Latency on Red Hat Enterprise Linux 6

Jeremy Eder, Senior Software Engineer

Version 1.0

June 2012





1801 Varsity Drive™
Raleigh NC 27606-2072 USA
Phone: +1 919 754 3700
Phone: 888 733 4281
Fax: +1 919 754 3701
PO Box 13588
Research Triangle Park NC 27709 USA

Linux is a registered trademark of Linus Torvalds. Red Hat, Red Hat Enterprise Linux and the Red Hat "Shadowman" logo are registered trademarks of Red Hat, Inc. in the United States and other countries.

Solarflare and OpenOnload are U.S. registered trademarks of Solarflare Communications, Inc.

UNIX is a registered trademark of The Open Group.

Intel, the Intel logo and Xeon are registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

All other trademarks referenced herein are the property of their respective owners.

© 2012 by Red Hat, Inc. This material may be distributed only subject to the terms and conditions set forth in the Open Publication License, V1.0 or later (the latest version is presently available at <http://www.opencontent.org/openpub/>).

The information contained herein is subject to change without notice. Red Hat, Inc. shall not be liable for technical or editorial errors or omissions contained herein.

Distribution of modified versions of this document is prohibited without the explicit permission of Red Hat Inc.

Distribution of this work or derivative of this work in any standard (paper) book form for commercial purposes is prohibited unless prior permission is obtained from Red Hat Inc.

The GPG fingerprint of the security@redhat.com key is:
CA 20 86 86 2B D6 9D FC 65 F6 EC C4 21 91 80 CD DB 42 A6 0E

Send feedback to refarch-feedback@redhat.com



Table of Contents

1 Executive Summary.....	1
2 Test Configuration.....	2
3 Testing - Phase 1 (Baseline).....	3
3.1 Baseline Setup.....	3
3.2 Baseline Results.....	4
3.3 What does it all mean?.....	5
4 Testing – Phase 2 (Tuned + OpenOnload).....	5
4.1 OpenOnload Setup.....	5
4.2 OpenOnload Results.....	5
4.3 What does it all mean?.....	6
5 Throughput Sanity Check - Phase 3.....	7
Appendix A: Revision History.....	8



1 Executive Summary

This paper examines the performance characteristics of the Solarflare OpenOnload technology on Red Hat Enterprise Linux (RHEL) 6. OpenOnload® is a high performance network stack from Solarflare that dramatically reduces latency and CPU utilization, and increases message rate and bandwidth. OpenOnload runs on Linux and supports TCP/UDP/IP network protocols with the standard BSD sockets API, and requires no modifications to applications to use. It achieves performance improvements in part by performing network processing at user-level, bypassing the OS kernel entirely on the data path. Networking performance is improved without sacrificing the security and multiplexing functions that the OS kernel normally provides.

Red Hat partnered with Solarflare Communications for this effort. Solarflare provided the hardware required for the testing as well as published *Low Latency Quickstart Guide*.

Baselines were established on RHEL6.2, and results compared to those gathered using OpenOnload libraries.

While Red Hat ships a number of kernel-bypass technologies that are similar to OpenOnload such as RoCE and Infiniband/RDMA, Solarflare uses a combination of kernel-bypass, specially designed hardware and optimized drivers to achieve extreme low-latency while maintaining application compatibility and support for TCP/IP protocol.

Solarflare is a member of the Red Hat Partner Program, and has listed the OpenOnload technology in Red Hat's Product Catalog:

<http://redhat.force.com/catalog/PCProductDetail?id=a1k6000000Gzo0AAC>



2 Test Configuration

Commercially available, industry-standard hardware and software was used for the Systems Under Test (SUT).

Hardware	Details
Server	Dell R510 2 Socket – 12 Cores (Hyperthreads Disabled) Intel Xeon X5650 @ 2.67 GHz 48 GB RAM (8GBx3 @ 1333MHz per NUMA node)
Network	2 – Solarflare SFN5122F-R6 Dual-Port SFP+ 10G Ethernet Enterprise Server Adapter Firmware: 3.2.0

Table 1: Hardware Configuration

Software	Details
Operating System	Red Hat Enterprise Linux 6.2 (kernel-2.6.32-220.el6.x86_64)
Solarflare Driver	3.2.0.6044
Solarflare OpenOnload	201109-u2

Table 2: Software Configuration



3 Testing - Phase 1 (Baseline)

The goal of Phase 1 was to establish baseline performance results using Red Hat Enterprise Linux 6.2 and the in-kernel networking stack, on which to make further comparisons. The test used **qpidd-latency-test** from Red Hat's MRG Messaging product. It is provided by the **qpidd-cpp-client-devel** package, version 0.12-6.

qpidd-latency-test is a benchmark utility that sends messages of a specified size at a specified rate, and outputs min/max/avg round-trip-time (RTT) latencies for each message. It is also a multi-threaded benchmark. This test was conducted with # of threads equal to the number of cores in the NUMA node (**- -worker-threads=6**).

3.1 Baseline Setup

The following tuning settings were applied prior to Phase 1 testing:

- Power Management and Systems Management Interrupts (SMIs) were disabled in the server BIOS.
- NIC firmware and drivers were updated to the versions in Table 2.
- Servers were connected through a Cisco 5596 10G switch via fiber.
- The **tuned** latency-performance profile was applied. For more information on the **tuned** package and available profiles, see the *Power Management Guide* on <http://docs.redhat.com>.
- Tuning suggestions made in the Solarflare *Low Latency Quickstart Guide* (not specific to OpenOnload) were applied to the operating system. This document is available at <https://support.solarflare.com>.
 - Lowering coalesce values to 0
 - Disabling interrupt modulation
 - Locking the operating system into cstate 1
 - Binding benchmark applications to a NUMA node dedicated to the benchmark (**numactl -N1 -m1**). Scheduler is allowed to migrate the benchmark, but only within the NUMA node.
 - Setting **qpidd** process to FIFO scheduler: **chrt -f 1**
 - Jumbo frames (9000 byte MTU). Note that OpenOnload limits MSS to 2048 bytes. See *OpenOnload User Guide*, page 35 at <https://support.solarflare.com> for more information.
 - Pin NIC IRQs to NUMA node hosting the benchmark applications. See the following Tech Brief for more information: <https://access.redhat.com/knowledge/techbriefs/optimizing-red-hat-enterprise-linux-performance-tuning-irq-affinity>
 - Disable all unnecessary services and cronjobs



- No changes to operating system **sysctl** values were necessary.
- For more information on performance tuning in RHEL, see the *Performance Tuning Guide* on <http://docs.redhat.com>.

3.2 Baseline Results

Table 3 lists the average round-trip-time (RTT) latency of a message of varying size in milliseconds. The duration of the individual tests was 2 minutes, and the TCP_NODELAY socket option was used in all cases.

Message Rate	8 Bytes	256 Bytes	1024 Bytes
10K	0.1440	0.1466	0.1491
20K	0.1261	0.1307	0.1329
30K	0.1333	0.1350	0.1431
40K	0.1361	0.1428	0.1495
50K	0.1490	0.1633	0.1707
60K	0.1752	0.1901	0.2065
70K	0.2085	0.2274	0.2790
80K	0.2565	0.2727	0.3953
90K	0.3173	0.3616	0.5242
100K	0.4006	0.4648	0.8448

Table 3: Avg RTT Latency - Baseline Results

Latency is low and stable for rates up to at least 90K messages/sec. Larger messages (1024 bytes or higher) do have an impact on latency as more data must be copied to/from memory and ultimately transferred over the wire. For example, at the same 100K/s rate, latency of 1024 byte messages is 1.8x greater than 256 byte messages.



3.3 What does it all mean?

Red Hat Enterprise Linux and Solarflare NICs provide exceptional low latency performance with no exotic tuning efforts. This provides a sound foundation for very low latency network throughput at common message sizes and rates. More CPU time is also required for these heavier workloads.

4 Testing – Phase 2 (Tuned + OpenOnload)

The goal of Phase 2 was to establish performance results when using the same benchmark config/tuning and adding OpenOnload. This isolates OpenOnload libraries as the only environmental variable modified between test runs.

4.1 OpenOnload Setup

During Phase 2 testing, all existing hardware/software tuning was preserved. Other than using the OpenOnload libraries (`onload --profile=latency`), no additional tuning settings were applied prior to Phase 2 testing.

4.2 OpenOnload Results

Identical test procedures and timeframes were used to generate OpenOnload latency data. However, OpenOnload automatically reduces the MSS to < 2048 bytes. See the *Solarflare Onload User Guide Version 201109 u2*, Section 6.11.

Message Rate	8 Bytes	256 Bytes	1024 Bytes
10K	0.0867	0.0825	0.0839
20K	0.0776	0.0752	0.0778
30K	0.0797	0.0825	0.0844
40K	0.0887	0.0909	0.0951
50K	0.0996	0.1003	0.1089
60K	0.1178	0.1227	0.1332
70K	0.1356	0.1410	0.1542
80K	0.1531	0.1603	0.1754
90K	0.1719	0.1811	0.1966
100K	0.1912	0.2041	0.2191

Table 4: Avg RTT Latency - OpenOnload Results



4.3 What does it all mean?

Using the OpenOnload libraries reduces latency in some cases by a factor of 2-4x requiring no code changes traditionally associated with kernel-bypass technologies (such as RDMA). CPU usage was also reduced, allowing the OpenOnload stack to achieve higher message throughput as well (not presented here).

Average TCP Latency, MRG-M qpid-latency-test

Red Hat Enterprise Linux 6.2 and Solarflare OpenOnload 201109-u2

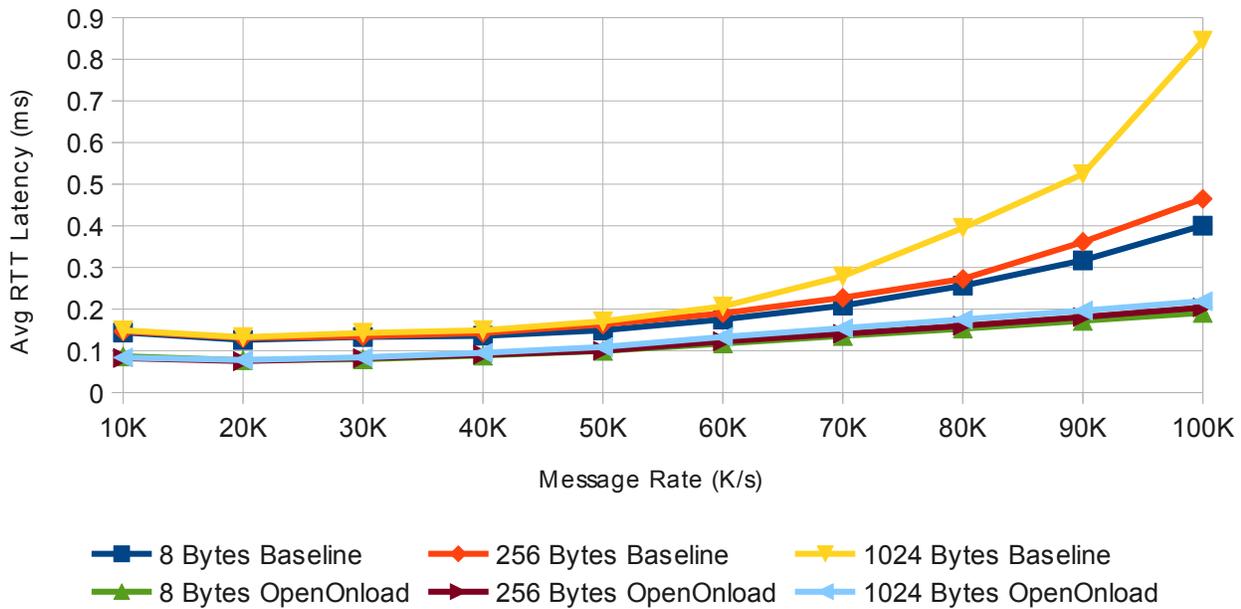


Illustration 1: Average TCP Latency

In this graph, the X-axis represents message rate per second. The Y-axis represents average round-trip-time (RTT). As previously detailed, **qpid-latency-test** reports min/max/avg RTT latency. That is, the time it takes for a message to be sent and reflected back to the sender. Results always use average latency reported by **qpid-latency-test**.



5 Throughput Sanity Check - Phase 3

Typically, low latency comes at the cost of reduced throughput. This is a trade-off system architects must quantify in their environment. The Linux kernel uses queues, buffers and timers in an attempt to improve efficiency. For example, [Nagle's algorithm](#) tends to hurt latency, as it implies waiting for a full packet's worth of data be "ready to send", before the packet is dispatched. For the kernel network stack, the below **netperf** data illustrates that cost, at higher packet rates for messages under 2K in size.

Kernel-bypass techniques such as OpenOnload reduce the CPU cost associated with latency tuning and recapture much of the observed degradation. CPU **sys** time was saved due to reduced context switching. These cycles were reallocated to the benchmark itself, thus improving throughput.

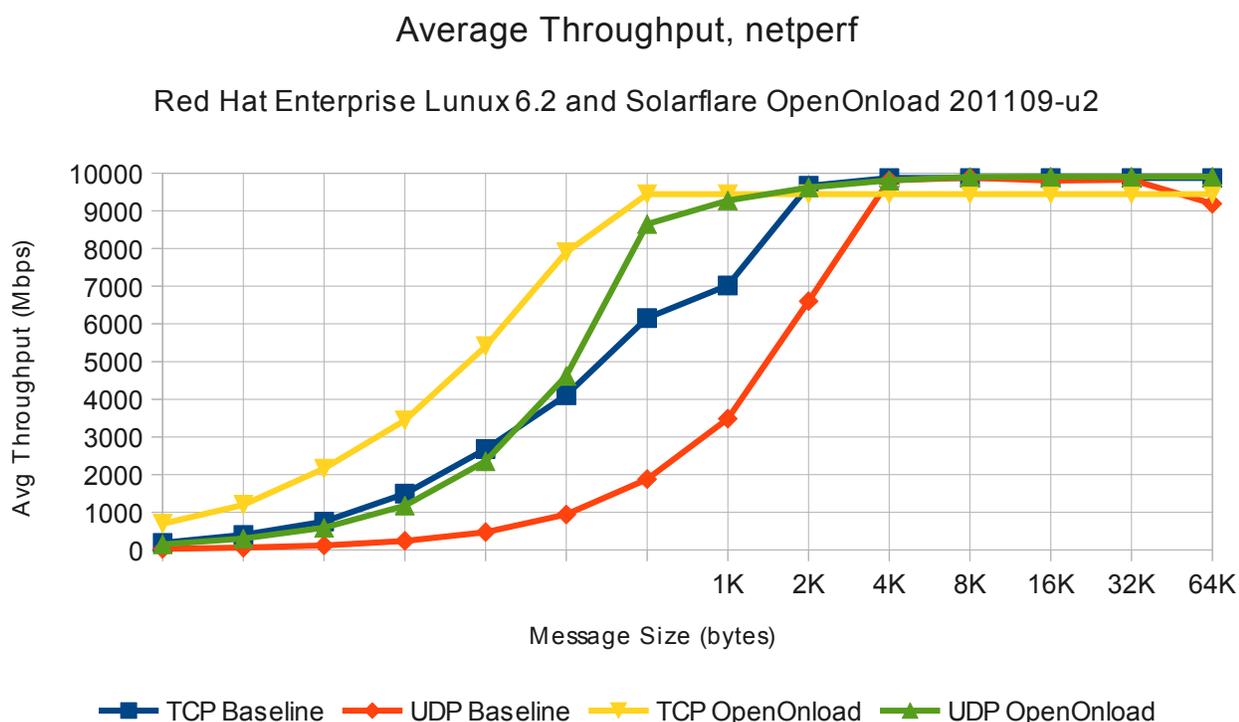


Illustration 2: Average Throughput

The Solarflare OpenOnload kernel-bypass technology affords measurable performance gains for latency-sensitive customers. By removing the traditional context-switching associated with the standard Linux network stack, OpenOnload allows more CPU time available to the user application.



Appendix A: Revision History

Revision 1.0
Initial Release

Tuesday June 12, 2012

Jeremy Eder