Accelerating Video on Demand (VOD) Networks with Solarflare OpenOnload

SolarflareTechnologyBrief

Executive Summary

To respond to customer requests to increase Nginx performance in video on demand (VOD) networks, Solarflare has recently enhanced its OpenOnload application acceleration stack to benefit the networking and application requirements for VOD deployments. This paper shows that OpenOnload with the Nginx application can improve the efficiency and performance of the video on demand networks.

The most significant finding is the efficiency benefit of the Solarflare adapter and OpenOnload software stack versus the Intel adapter and its standard kernel network stack. At 2x10GbE, the Solarflare adapter requires only two cores to achieve the maximum Nginx performance limit of about 18,000 simultaneous connections. In contrast, the Intel adapter needs three cores to fill up the bandwidth of 2x10GbE links. The Solarflare adapter with OpenOnload is therefore 1.5 times more efficient or achieves up to 34% greater performance than Intel. At 40GbE Solarflare shows itself to be 1.67 times more efficient supporting 35,000 simultaneous connections with just 3 cores and up to 77% higher performance than the Intel adapter. Bottom line: Solarflare delivers to network architects and IT planners the ability to significantly reduce CPU utilization.

Introduction: Scope and Purpose

Rapid growth in the VOD market stimulates competition and innovation among content providers, broadcasters, and traditional telcos as well as in the vendor ecosystems that sell into this industry.[1] Many VOD service providers and the related content delivery network (CDN) operators strive to compete and differentiate themselves by offering higher definition video, better service level agreements (SLAs), as well as different and new programing. Many providers are looking to increase the performance components of their VOD platforms by optimizing each layer of the stack, including compute, storage, network, and application.

In particular, VOD and content delivery network (CDN) operators are investing in the improvement of their Nginx application performance. Nginx[2] (or its commercial version Nginx Plus) is deployed in many Web and content delivery networks (CDNs) for servers to support long-lived HTTP connections. This paper shows the superior Nginx application scalability and throughput of servers equipped with Solarflare 10GbE and 40GbE network adapters when combined with the OpenOnload high-performance network stack. More specifically, this paper investigates the use case where a web server handles many concurrent long-lived requests modelling a website delivering video on demand.

[1] Video on demand continues to revolutionize the TV and film industries, not to mention change the landscape of the government regulatory regimes for over the top (OTT) services in the USA and elsewhere in the world. A recent Nielsen report indicated that over 40 percent of USA households have access to at least one subscription-based video on demand service. (http://ir.nielsen.com/files/doc_presentations/2015/total-audience-report-q4-2014.pdf) John Oliver's 13 minute monologue last year on net neutrality prompted over 45 thousand comments on the FCC web site (https://www.youtube.com/watch?v=fpbOEoRrHyU) and http://www.bloomberg.com/politics/articles/2015-02-26/how-john-oliver-transformed-the-net-neutrality-debate-once-and-for-all

[2] http://wiki.nginx.org/Main

sales@solarflare.com

US 1.949.581.6830 x2930

UK +44 (0)1223 477171

HK +852 2624-8868

www.solarflare.com

OpenOnload Defined

Solarflare OpenOnload is a Linux-based, high-performance application accelerator that delivers high throughput and application scalability. Because OpenOnload presents a standard BSD sockets API to Nginx, the application requires no modification. OpenOnload performs network processing at user-level and is binary-compatible with existing applications that use TCP/UDP with BSD sockets. **(See Figure 1)** It comprises a user level shared library that implements the protocol stack, and a supporting kernel module.

Solarflare has been developing the OpenOnload TCP/UDP stack since 2002. Since 2006, the product has been deployed at over 1000 sites specifically for hybrid user-space/kernel operation. OpenOnload supports the latest Linux performance features and ensures robust and full RFC compliance.

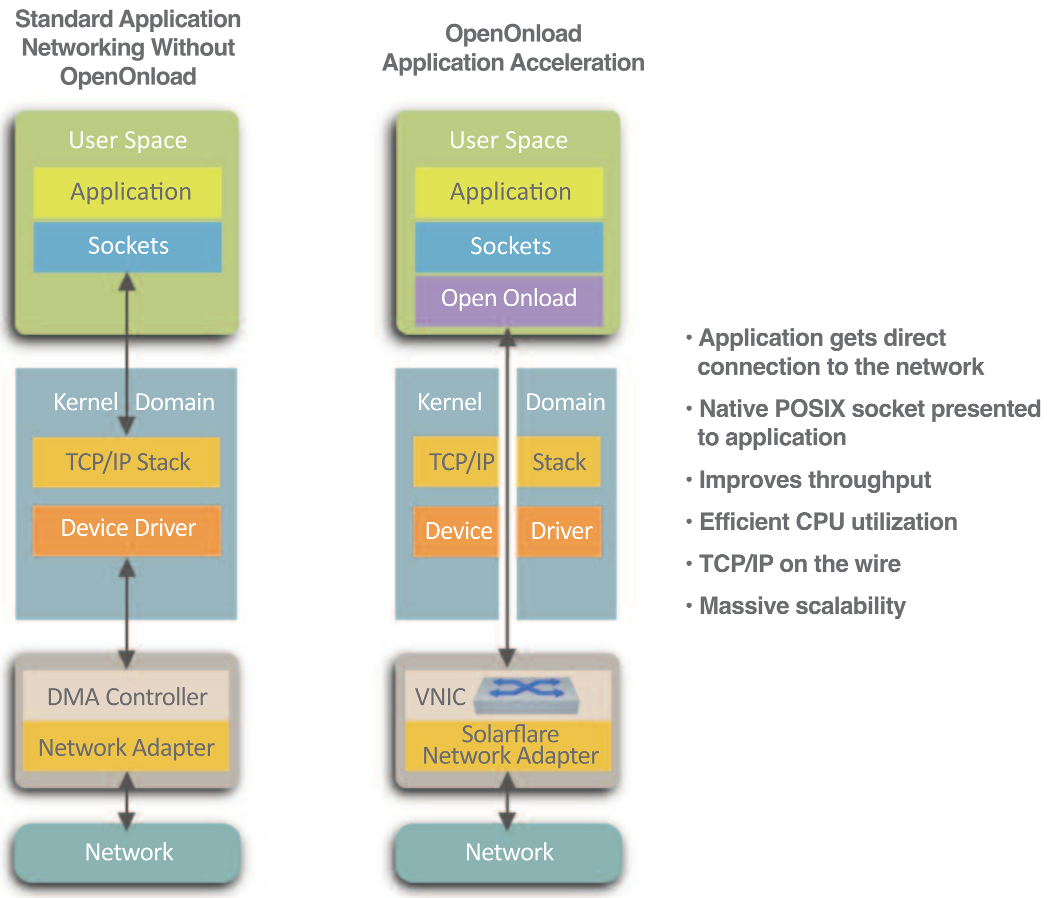
Standard Application Networking Without OpenOnload
OpenOnload Application Acceleration
User Space
Application
Sockets
Open Onload
Kernel Domain
TCP/IP Stack
Device Driver
DMA Controller
Network Adapter
VNIC
Solarflare Network Adapter
Network
• Application gets direct connection to the network
• Native POSIX socket presented to application
• Improves throughput
• Efficient CPU utilization
• TCP/IP on the wire
• Massive scalability

Figure 1. How OpenOnload works.

OpenOnload and Nginx Acceleration

Three key features of OpenOnload contribute to increasing Nginx application performance. First, OpenOnload allows network processing at user-level, bypassing the OS kernel. Because OpenOnload includes a TCP stack that runs completely in user space, it avoids the overhead of system calls that consume CPU resources. In addition, application and network performance is improved without sacrificing the security and multiplexing functions that the OS kernel normally provides.

sales@solarflare.com
US 1.949.581.6830 x2930
UK +44 (0)1223 477171
HK +852 2624-8868
www.solarflare.com

Second, OpenOnload now supports the SO_REUSEPORT socket option that allows multiple TCP listening sockets to bind to the same IP address and port and to distribute incoming connection requests between these sockets. This feature can be exploited by applications to increase their scalability. OpenOnload's implementation of SO_REUSEPORT is superior to that of the Linux kernel because OpenOnload has the ability to instantiate independent, per process, per thread network stacks each of which has direct access to hardware. This eliminates lock contention between processes when sending to and receiving from the network stack.

Third, OpenOnload supports socket caching, which avoids the overhead of additional system calls. This improves performance when connections are short-lived.

Test Design and Methodology

The System Under Test (SUT)

This benchmark used multiple servers to create the workload directed at the system under test (SUT). The specifications of the system under test can be found in **Table 1**. More detail is available in the **Appendix**.

SUT Server Specification	
Server Model	DELL R630 PowerEdge
RAM	64 GB
Chipset	Intel C610
CPU Specifications	
Number of NUMA nodes	2
Cores per node	6
Threads per core	1
Model	Intel® Xeon® CPU E5-2620 v3 @ 2.40GHz
Architecture	Haswell
Clock speed	2400 MHz
L1 Cache	32 kB (instr.) +32 kB (data)
L2 Cache	256 kB
L3 Cache	15360 kB
NUMA node 0 CPU	0,2,4,6,8,10
NUMA node 1 CPU	1,3,5,7,9,11

Table 1. SUT hardware specifications.

sales@solarflare.com

US 1.949.581.6830 x2930

UK +44 (0)1223 477171

HK +852 2624-8868

www.solarflare.com

Server Adapter Cards

The server adapters used for this test are listed in **Table 2**. Both adapters are housed in the SUT on separate NUMA nodes. The Solarflare adapter is housed in a PCIe slot associated with NUMA node 1, while the Intel adapter is associated with NUMA node 0.

For the benchmarking the Solarflare adapter was used with the OpenOnload stack. The Intel adapter was driven using the i40 driver. See the **Appendix** for the summary of drivers and versions.

Company	Model
Solarflare	Flareon Ultra SFN7142Q
Intel	Converged Network Adapter XL710
Solarflare	Flareon SFN7002F
Intel	Converged Network Adapter X710

Table 2. Server adapters.

Methodology and Workloads

In the HTTP community multiple tools are available for load testing including ApacheBench and WRK. For this benchmark exercise WRK was chosen over ApacheBench (ab) because WRK needs a smaller number of load generators to create the required load and the desired VOD test workload was more easily configured using WRK. The WRK test workload was modified to model a buffered video on demand traffic stream of a large CDN/VOD service provider.

The workload was designed to emulate a typical video on demand workload in three respects. First, the workload used long-lived connections between the client and server. Second, the test modeled a 1 MB buffer on the video client (i.e. in WRK) that drained at a target client side VOD consumption rate of 1 Mbps, where the client makes new requests for further data when the client buffer drops below 50% full. Third, the modified WRK used client buffer underrun to indicate that the server was not meeting the required quality of service and to consequently indicate a test failure.

The benchmark used one instance of WRK per core on each load generator. Each instance ran multiple connections that periodically downloaded some payload from the SUT. The aim of the benchmark was to determine the number of streaming connections that the SUT could sustain without any of them underflowing their local buffer. **Table 3** shows the connection settings.

Connection Property	Value
Total payload requested	10 MB
Drain speed of the buffer	1Mb/s
Buffer size	1 MB
Watermark level	50%
Number of distinct IP addresses used	20,000

Table 3. SUT settings for the Nginx server.

SOLARFLARE®

sales@solarflare.com

US 1.949.581.6830 x2930

UK +44 (0)1223 477171

HK +852 2624-8868

www.solarflare.com

At both 10GbE and 40GbE, the benchmarking compared an Intel adapter and the Solarflare adapter with OpenOnload.

OpenOnload

In the OpenOnload case Nginx processes were accelerated with OpenOnload configured in spinning mode. This means that application threads using OpenOnload could busy-wait when making network stack calls for a configurable duration, before sleeping and then being re-scheduled as the result of an interrupt when work arrived. In practice this means that the application context performed most tasks that would have otherwise been done in the interrupt context. As a consequence, OpenOnload generated very little interrupt load, eliminating the need to dedicate (logical) CPU cores to interrupt handling. Therefore for benchmarking, logical cores were assigned on the basis of using NIC local package logical cores first, and then using the NIC remote package cores. Interrupt handlers were assigned to each of the NIC local cores.

Results

Figure 2 represents the summary of the comparative Nginx performance tests for the server adapters with two 10GbE ports bonded using Link Aggregation. The plot shows the maximum number of sustained connections against the number of threads (or cores). The maximum limit of the system under test is of course the bandwidth of the bonded 2x10GbE links. The plots of data series show the flattening of the respective performance curves at 18,000 to 19,000 simultaneous connections.

The most significant finding is the improved efficiency of the Solarflare SFN7002F adapter with OpenOnload software versus the Intel adapter with its standard network stack. The Solarflare adapters need to use only two cores to achieve the maximum limit of about 18,000 simultaneous connections. The Intel adapter in contrast needs three cores to fill up the bonded bandwidth of the 2x10GbE links. The Solarflare server adapter with OpenOnload is shown therefore to be 1.5 times more efficient than the Intel adapter. Commensurately, the Solarflare adapter achieves up to 34% greater performance measured in maximum simultaneous connections than the Intel adapter.

Nginx Performance 2x10GbE

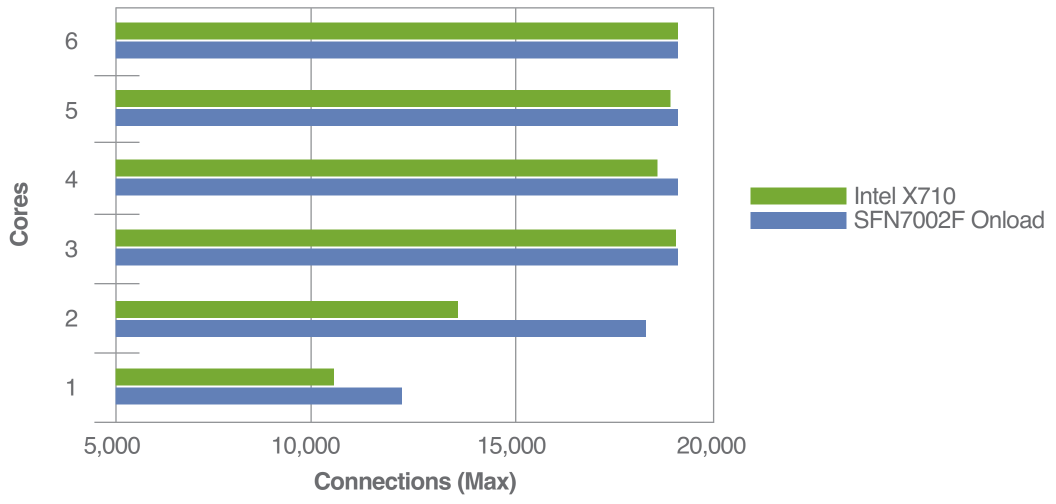

Figure 2. Maximum simultaneous connections vs. number of cores at 2x10GbE.

SOLARFLARE®

sales@solarflare.com

US 1.949.581.6830 x2930

UK +44 (0)1223 477171

HK +852 2624-8868

www.solarflare.com

Figure 3 represents the summary of the comparative tests for the 40GbE server adapters. The plot shows the maximum number of sustained connections against the number of threads (or cores). Once again, the Nginx performance of the system under test is ultimately limited by the bandwidth of the 40GbE link, about 38,000 to 39,000 connections. Similar to the 2x10GbE configuration, the Solarflare SFN7142Q 40GbE server adapter with OpenOnload is found to be much more efficient in its use of server resources than the Intel adapter. The Solarflare solution is able to fill up the link and achieve near maximum performance of 35,000 simultaneous connections with the use of just three cores. In contrast, the Intel adapter takes four to five cores to achieve maximum Nginx application performance. Solarflare shows itself to be up to 1.66 times more efficient than the Intel adapter. Similarly, Solarflare delivers significant performance increases that range up to 77% higher than the Intel XL710.

Nginx Performance 40GbE

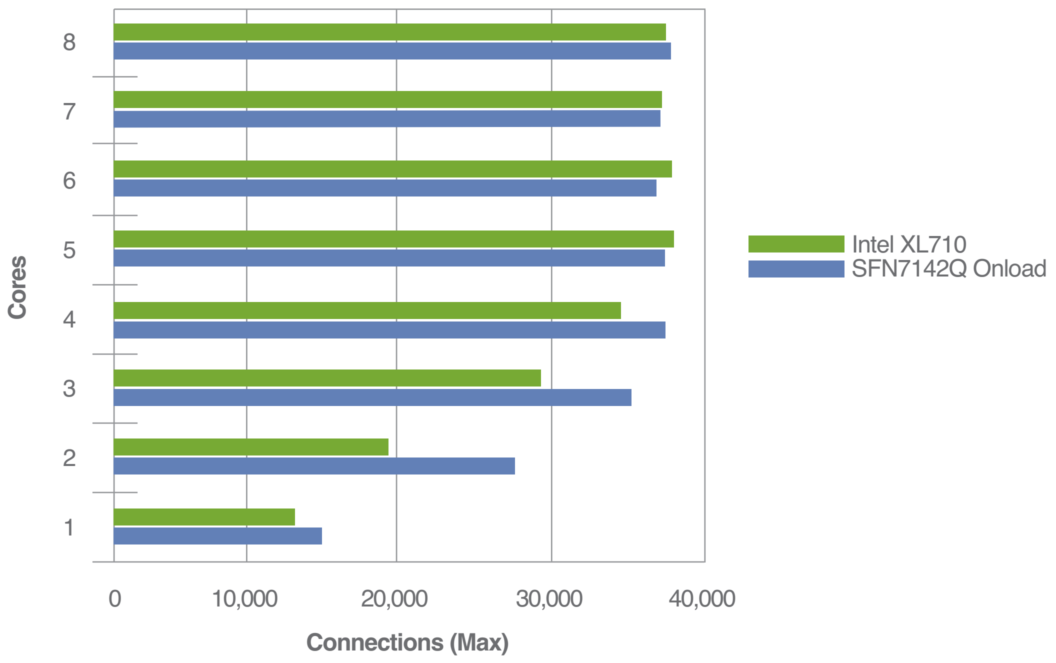

Figure 3. Maximum simultaneous connections vs. number of cores at 40GbE.

Conclusions

This Nginx application test was expressly designed to discover the differences between the respective server I/O systems, particularly the Solarflare OpenOnload software with SFN7142Q server adapter versus the Intel software with Intel XL710 server adapter. The benchmarking has shown that when examining Nginx application performance, Solarflare server adapters with OpenOnload are more efficient in their use of server resources and can achieve higher performance than Intel adapters at both 2X10GbE and 40GbE.

OpenOnload makes a significant contribution to Solarflare's greater efficiency and performance. As discussed above, three key features of OpenOnload contribute to its ability to increase Nginx application performance. First, OpenOnload allows network processing at user-level, bypassing the OS kernel and thereby avoiding the overhead of system calls that consume CPU resources.

sales@solarflare.com

US 1.949.581.6830 x2930

UK +44 (0)1223 477171

HK +852 2624-8868

www.solarflare.com

Second, OpenOnload supports the SO_REUSEPORT socket that increases Nginx scalability. Finally, OpenOnload supports socket caching, which avoids the overhead of additional system calls and improves performance when connections are short-lived.

With one possible exception, not much information appears to be in the public domain about average or typical CPU utilization in large scale-out compute environments. That being said, in general the business goal at VOD service providers is to make the most efficient use possible of all server resources including CPU utilization in order to achieve capex if not also opex savings.[3] To support this end, Solarflare delivers the ability to make more efficient use of CPU resources.

Bottom Line: For servers running Nginx with long-lived connections simulating video on demand traffic, Solarflare outperforms Intel both in terms of more efficient use of CPU resources and higher maximum Nginx performance.

Appendix

Benchmark Configuration and Nginx Workload

System Configuration

The benchmark was performed on Linux servers, running Red Hat Enterprise Linux 7. The specifications of the load generators can be found in **Table 4**. The drivers for the server adapters are listed in **Table 5** (next page).

Load Generator Server Specification	
Server Model	DELL R620 PowerEdge
RAM	32 GB
Chipset	Intel C602
CPU Specifications	
Number of NUMA nodes	2
Cores per node	6
Threads per core	1
Model	Intel® Xeon® CPU E5-2643 v2 @ 3.50GHz
Architecture	Ivybridge
Clock speed	3600 MHz
L1 Cache	32 kB (instr.) + 32 kB (data)
L2 Cache	256 kB
L3 Cache	25600 kB
NUMA node 0 CPU	0,2,4,6,8,10
NUMA node 1 CPU	1,3,5,7,9,11

Table 4: Load generator specifications.

[3] The exception being Barroso, Clidaras and Hölzle, *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*, (Synthesis Lectures on Computer Architecture), 2nd edition July 2013. www.morganclaypool.com

SOLARFLARE®

sales@solarflare.com

US 1.949.581.6830 x2930

UK +44 (0)1223 477171

HK +852 2624-8868

www.solarflare.com

Driver	Version
Solarflare Kernel Driver	v4_5_1_1010
Intel Driver	i40e-1.2.38
Open Onload	openonload201502-u1

Table 5: Server adapter drivers.

Network settings were chosen that represent as closely as possible conditions in VOD networks. The list of shared system specifications can be seen in **Table 6**.

System Configurations	
OS	Red Hat Enterprise Linux 7
Kernel version	3.10.0-123.el7.x86_64
MTU	1500 b
IP version	4
Hyperthreading	Disabled
Kernel cmdline	rd_NO_LUKS rd_NO_LVM rd_NO_MD rd_NO_DM LANG=en_GB.UTF-8 vconsole.font=latarcyrheb-sun16 vconsole.keymap=uk nomodeset biosdevname=0

Table 6: Common system configurations.

For the purposes of this benchmark, the following Linux services were disabled:

- NetworkManager
- ebtables
- iptables
- irqbalance

The Linux network settings that were changed for this benchmark can be found in **Table 7** and **Table 8** (next page). These alterations remove a series of bottlenecks in the kernel to enable more accurate relative comparisons of the performance of the various adapters and drivers.

Overridden Linux sysctl Settings

Linux sysctl option	Override
net.ipv4.tcp_max_syn_backlog	65535
net.core.netdev_max_backlog	8192
net.core.somaxconn	65535
net.ipv4.tcp_fin_timeout	0
net.ipv4.ip_local_port_range	10000 61000
net.ipv4.tcp_tw_reuse	1
net.ipv4.tcp_tw_recycle	1
net.ipv4.tcp_rfc1337	1
net.ipv4.tcp_max_tw_buckets	5880000
net.ipv4.neigh.default.gc_thresh1	65000
net.ipv4.neigh.default.gc_thresh2	65500
net.ipv4.neigh.default.gc_thresh3	65536
net.ipv4.xfrm4_gc_thresh	65536
fs.file-max	4000000
fs.nr_open	4000000
kernel.shmmni	16000

Table 7: List of overridden Linux sysctl settings.

Overridden Linux Kernel Settings

Linux kernel option	Override
nmi_watchdog	0
perf_event_paranoid	-1
kptr_restrict	0

Table 8: List of overridden Linux kernel settings.

10 and 40GbE Network Setup

Network connectivity was accomplished with a Layer 2 10/40GbE high density top of rack switch. Except for bridging appropriate ports, no additional setup was performed. The IPv4 subnet used for this benchmark was 172.17.0.0/28 with the SUT holding the first address and each load generator having a unique IP in this subset. The test set up implements multiple IP addresses. Specifically, the test set up enabled enough IP addresses so for any one run each connection could have its own unique address. In practice, however, some IP addresses at any one time might initiate more than one connection due to the random selection of IP addresses.

Nginx and Client (WRK) Workloads

The benchmark uses two principal software components. The load generators are driven using the WRK benchmark which is described below. The system under the test is driven using the Nginx application.

WRK is a benchmark that was created to test HTTP transaction rates. WRK was chosen over ApacheBench (ab) because WRK needs a smaller number of load generators to create the required load and the desired VOD test workload was more easily configured using WRK[4].

The code of WRK was modified to add several features that were necessary for the methodology of this benchmark. Firstly, a video buffer was integrated into WRK, which controls when the client requests more payload. The operation of the buffer is as follows:

- After initial connection, repeated requests are made for data using the common target URL. Upon receipt of each fragment from the Socket, the fill level is updated accordingly. The data is not used.
- Once the buffer is filled it stops filling and changes its state to full.
- When instructed by its parent thread, it will then start a timer of 500ms, which upon firing will decrement the fill level by the specified drain rate.
- After the fill level is decremented, the fill level is examined:
  - If it has dropped below zero, the buffer is taken out of service and the parent thread has a counter incremented for dropped connections.
  - If the buffer has not underflowed and is above the watermark then another timer is set and no further action taken.
  - If the buffer has not underflowed and the level is below the watermark then a request is made for more of the payload.
- Upon receipt of payload data the buffer level is incremented by the number of bytes received.
- New connections added via the control API also go through the fill stage before being checked for underflow.

The modifications to the WRK code also include the addition of a control socket that allows instructing the running WRK processes to open new connections and queries those processes for the number of opened, streaming, and dropped connections. This control socket communicates over a separate network and does not interfere with any bandwidth measurements. The settings used for the WRK benchmark are listed in **Table 9** (next page).

sales@solarflare.com

US 1.949.581.6830 x2930

UK +44 (0)1223 477171

HK +852 2624-8868

www.solarflare.com

WRK Clients Settings

Version	3.1.1 [epoll]
Setting	Value
Payload size	10MB
Initial connection count	64
Step size (connection count increase)	256
Interval (connection count increase)	1 secs.
Expected Link Speed	40Gb/s
Thread count	All available cores
IRQ/CPU mask	All available cores
Drain rate	1Mb/s
Rate limit ramp up	True
IP Range	1
Test buffer size	5MB
Watermark level	50%
Data Blocks	128
Approach Step size (connection count increase)	32
Approach Interval (connection count increase)	30 secs.
Approach Start Limit (connection count increase)	95%

Table 9. Settings used for the WRK clients.

Definitions of terms used in **Table 9**:

Step size indicates the size of the increments by which we ramp up the number of streaming connections. Smaller step_size means smoother ramp up and more precise measurement, but also a longer test.
Interval indicates how often in seconds we check for the number of streaming connections and if sufficient proportions of open connections are streaming, we open more connections (the number we open is step_size).
Data blocks identify the number of individual 10MB files that we can stream so that we don't stream always the same 10MB payload. The content of these blocks is randomized before each test.
Approach step size indicates the step_size as above, but when we are in approach phase. We enter the approach phase when we are close to the link speed (by default 95% of link speed. We slow down the ramp up and take smaller steps to make the measurement more precise. Notice at 32, its value is much smaller that the step size of 256.
Approach interval is similar to interval above, approach interval indicates how often in seconds we check for the number of streaming connections but when in approach phase. Notice that it is much longer than the interval.
Approach start limit is the percentile of link speed when we enter approach phase. Default is when we hit 95% of expected link speed.

SOLARFLARE®

sales@solarflare.com

US 1.949.581.6830 x2930

UK +44 (0)1223 477171

HK +852 2624-8868

www.solarflare.com

SolarflareTechnologyBrief

Nginx

The settings used for the Nginx server are listed in **Table 10** below.

Nginx Server Settings	
Version	Nginx/1.7.7
Setting	Value
SO_REUSEPORT	True
Thread Count	Variable, see Section 5
Total Concurrency	20 k
Sendfile	False
Rate Limit	5 M
Payload Size	10 MB
CPU/IRQ Mask	Variable, see Section 2.4

Table 10: Load generator specifications.

The SO_REUSEPORT socket option for more efficient port allocation is now supported natively in the latest version of Nginx 1.9.9. The payload content for the Nginx server is random data that is re-generated every time the benchmark is run. The data is then co-located on the file system and mounted as tmpfs in order to ensure that the content that Nginx is serving is resident in memory on the same system, rather than on the server's disk subsystem.

SOLARFLARE®

sales@solarflare.com

US 1.949.581.6830 x2930

UK +44 (0)1223 477171

HK +852 2624-8868

www.solarflare.com