SOLARFLARE®

Technology Brief
Application Note – OpenOnload® 201502

Solarflare Technology Brief

Executive Summary

The 201502 release of OpenOnload adds a number of major features to expand the platforms that can benefit from OpenOnload acceleration, and deliver further improvements in performance for an even wider set of applications.

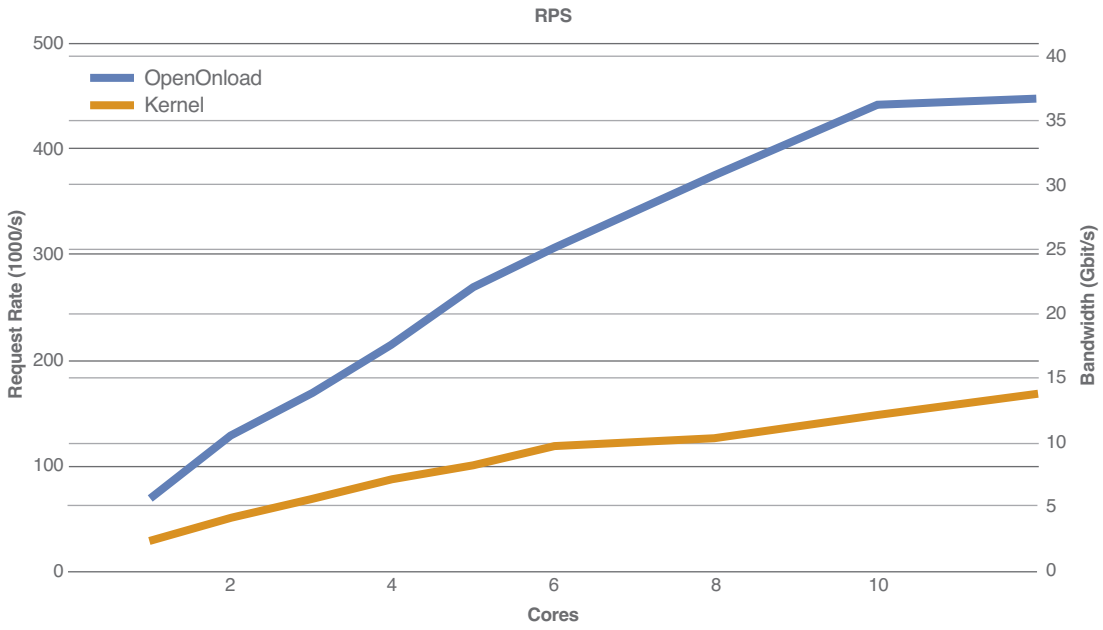
- **Higher performance** through a new scalable epoll implementation, and socket caching to double the achievable connection rate
- **Lower latency** for TCP transmissions through a new "delegated sends" API
- **New platforms** that now benefit from OpenOnload acceleration, including Ubuntu, Debian, KVM and Docker containers
- **Enhanced management** through a new remote monitoring tool/API, black and white listing of network interfaces, and improved ef_vi error counters

Higher Performance

The OpenOnload-201502-u2 release includes two major new features that vastly improve OpenOnload's scalability and extend the set of applications that can realize the performance benefits of OpenOnload.

The first of these is socket caching. This feature allows OpenOnload to cache file descriptors and filters to system calls into the OS for each new socket created. For applications with a high rate of incoming connections (e.g. web servers and other similar use cases) this feature translates into large increases in achievable connection rate and moves OpenOnload well beyond the performance of the native kernel network stack for applications with high connection rates.

Figure 1. Nginx CPU Scaling



sales@solarflare.com

US 1.949.581.6830 x2930

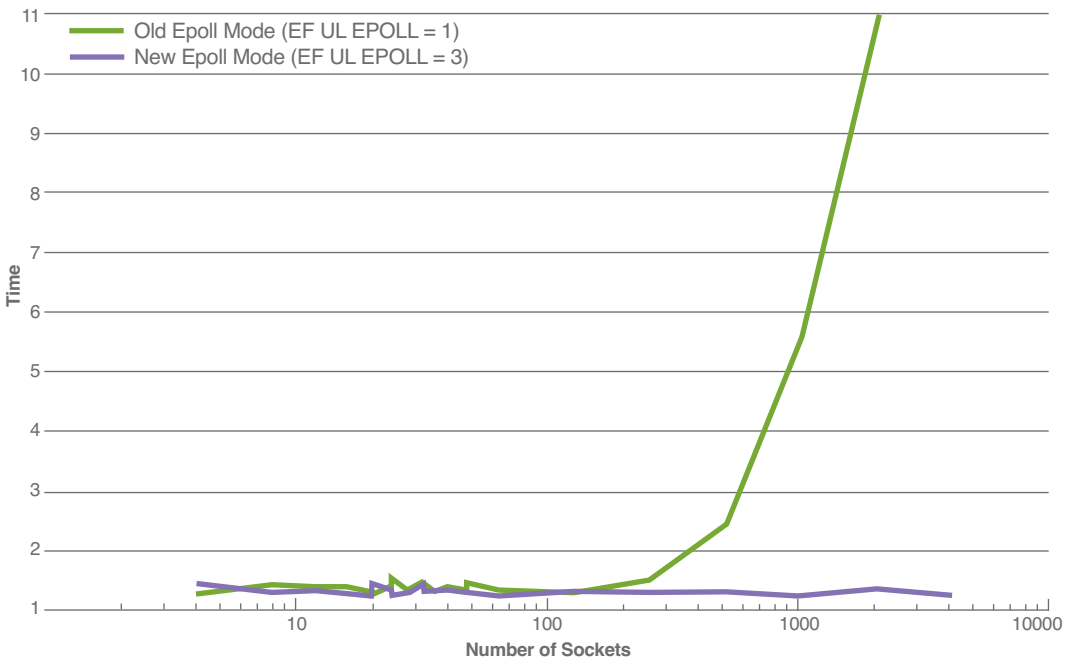
UK +44 (0)1223 477171

HK +852 2624-8868

www.solarflare.com

The second is a new scalable epoll mode[1], which now scales linearly rather than in proportion to the size of the epoll set. For large epoll sets this translates to a big improvement in performance and scalability, as shown in **Figure 2** below.

Figure 2. Epoll Mode Scaling



Lower Latency

The OpenOnload-201502 release includes a new API which significantly reduces latency on the critical path for TCP sends.

Prior to this release, users have had the choice of using OpenOnload with its standard and fully featured POSIX sockets API, user-space TCP and UDP protocol stack, and transparent interception library; or ef_vi which has a non-standard API and fewer features but absolute best latency.

The ef_vi API is essentially a Layer-2 interface. Ethernet frames are delivered to user-space with flexible (high-layer) flow steering. Application code using the API is required to implement any protocol processing required. This requirement is acceptable for datagram-oriented applications – for example financial market data consumption over UDP/IP multicast – but is not useful for complex reliable stream oriented protocols such as TCP. Instead, TCP applications have been recommended to use POSIX sockets with OpenOnload for best performance.

However, OpenOnload-201502 introduces a new OpenOnload delegated sends API, which allows the application to use OpenOnload's standard sockets API and TCP/IP stack for socket creation protocol management, demultiplexing and all usual data send receive operations, but also benefit from sending directly by invoking ef_vi and achieve the absolute best latency on the critical path.

[1] OpenOnload-201502-u2 or later should be used to get the benefits of scalable epoll mode.

SOLARFLARE®

sales@solarflare.com

US 1.949.581.6830 x2930

UK +44 (0)1223 477171

HK +852 2624-8868

www.solarflare.com

When using the delegated send API, sockets are created and managed through the POSIX API as normal. When an application knows it will need to make a critical path send, it requests through the delegated sends API that OpenOnload allow it to take over. OpenOnload provides the application with the current TCP/IP headers and details of how much data is authorized to be sent. The application is then free to make its send at the appropriate time using the ef_vi API or any other low-latency path. Having sent the data the application returns control, together with the data sent, to OpenOnload. This final step allows OpenOnload to correctly continue with protocol processing for the socket. For example, OpenOnload may update TCP state and perform whatever acknowledgements and retransmissions are later needed.

Diagrams illustrating the different call paths when using the delegated sends API are included in **Appendix 1**.

New Platforms

OpenOnload-201502 extends the set of platforms that are supported by OpenOnload. As well as Red Hat Enterprise Linux and SUSE Linux Enterprise Server, there is now support for Debian and Ubuntu Linux. With all supported Linux distributions Solarflare aims to support at least the latest two major release versions.

In addition, OpenOnload is now able to be used within a KVM guest, allowing the same performance benefits to be realized with direct and safe access to the hardware from the guest. This is achieved by passing through either a VF or PF to the guest for network access in the standard way.

The 201502 release also includes support for OpenOnload running in a Docker container. This will be extended in a future release to include support for separate network namespaces.

Finally, OpenOnload can be configured to run using multiple VFs or PFs in a non-virtualized environment for some specialized network configurations.

Full details of how to configure OpenOnload for use on these new platforms are included in the OpenOnload User Guide.

Enhanced Management

To increase the ease with which OpenOnload can be deployed across a large set of servers, and any issues in network performance quickly spotted, the OpenOnload 201502 release includes three new features.

Black and white listing of network interfaces allows the user to configure, on a system-wide basis, the network interfaces on which OpenOnload will be allowed to accelerate traffic. Users can specify either a white list (only those listed will be used) or a black list (OpenOnload will ignore those interfaces listed). This allows users to restrict which interfaces are used to, for example, run SolarCapture® on one interface and OpenOnload on another, or avoid attempts to use OpenOnload on older NICs or those without OpenOnload licenses in a mixed-NIC system.

SOLARFLARE®

sales@solarflare.com

US 1.949.581.6830 x2930

UK +44 (0)1223 477171

HK +852 2624-8868

www.solarflare.com

Remote monitoring of OpenOnload is now possible through a preview of a new API to provide details similar to those previously only available via running onload_stackdump on the server. An example client is included that will connect to the server and return full details of the OpenOnload stacks present in a structured JSON format. This output can then be parsed further to provide alerts or notification of problems well before they become critical. Since remote monitoring is a preview feature, Solarflare is very interested in receiving customer feedback.

Finally, ef_vi users can now access error counters for drops and their causes for each RX queue in use. These additional counters enable fine-grained monitoring of the network interface and facilitate improved diagnostics for any issues experienced. Details of all the features described here, including the ef_vi delegated send API with worked examples, are included in the release notes, User Guide, and the OpenOnload distribution.

Appendix 1

The following diagrams illustrate how the call paths vary when using delegated sends.

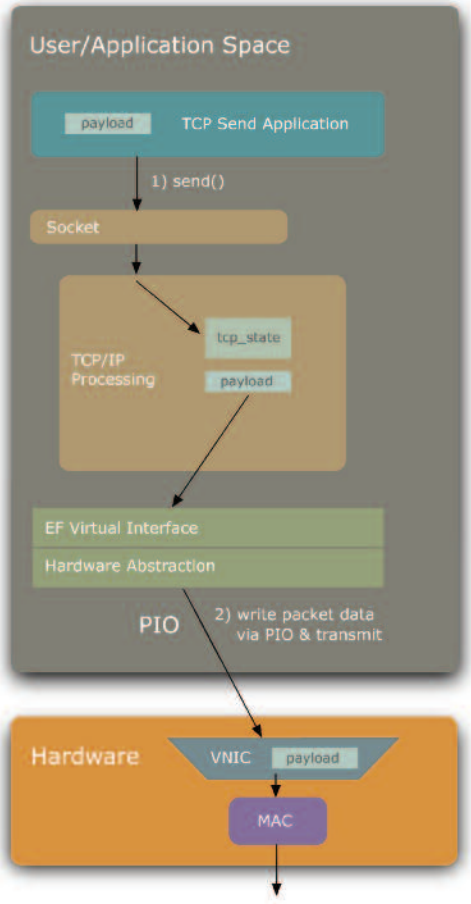


Figure 3. For a standard send() call, the entire operation forms the critical path.

SolarflareTechnologyBrief

SOLARFLARE®

sales@solarflare.com

US 1.949.581.6830 x2930

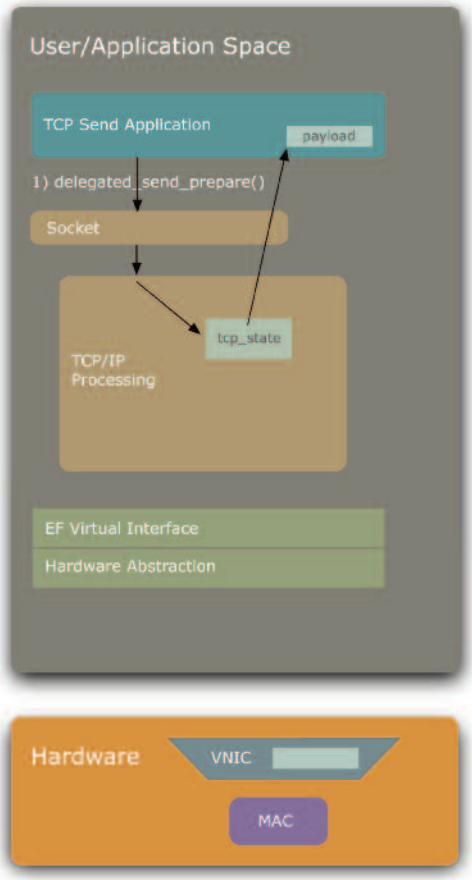
UK +44 (0)1223 477171

HK +852 2624-8868

www.solarflare.com

Appendix 1 – continued

Get TCP State



Copy TCP Packet to VNIC PIO

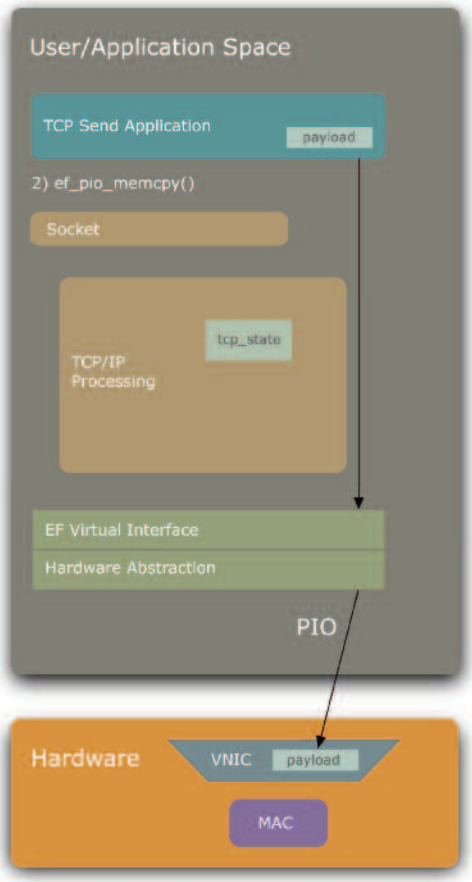


Figure 4. When the delegated send API is used, the caller can copy the data in advance using ef_pio_memcpy(), and use ef_vi_transmit_pio() to send it (next page).

sales@solarflare.com

US 1.949.581.6830 x2930

UK +44 (0)1223 477171

HK +852 2624-8868

www.solarflare.com

Appendix 1 – continued

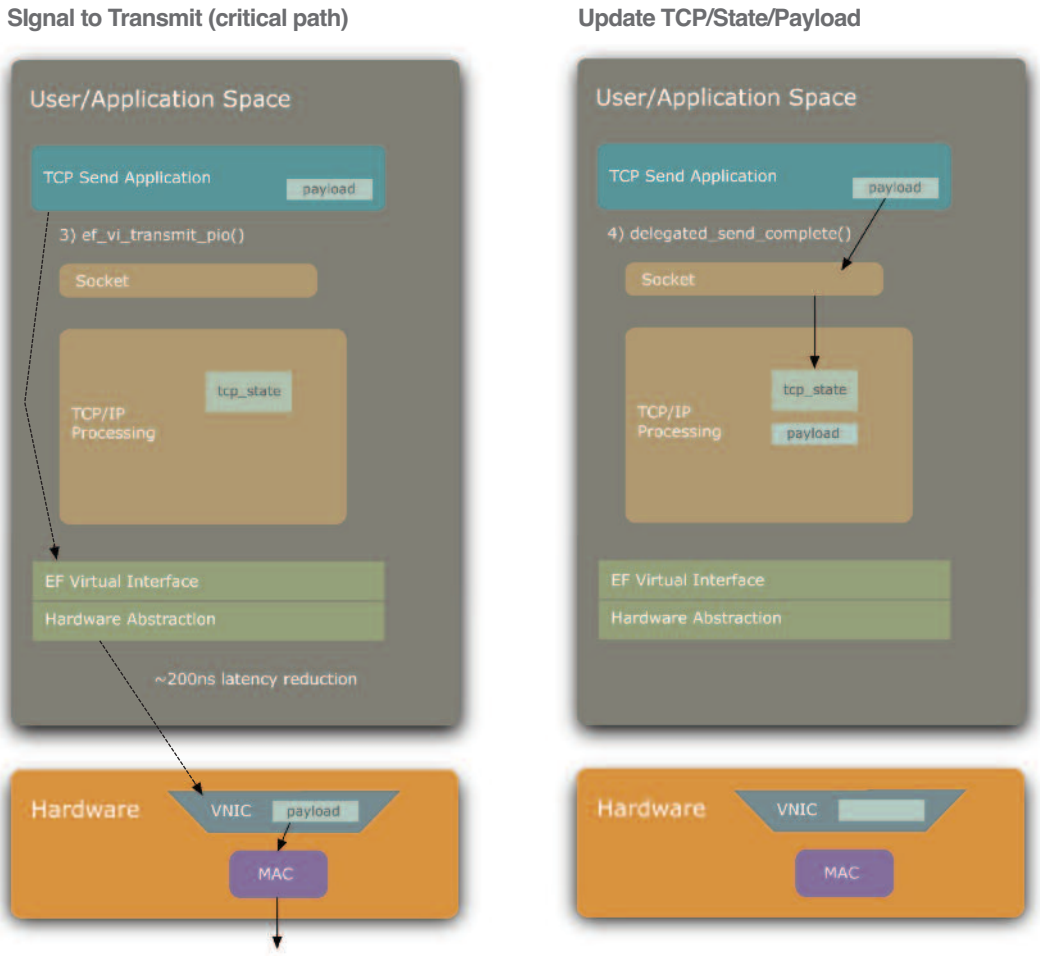


Figure 5. When the delegated send API is used, the caller can copy the data in advance using ef_pio_memcpy() (previous page), and use ef_vi_transmit_pio() to send it.

sales@solarflare.com

US 1.949.581.6830 x2930

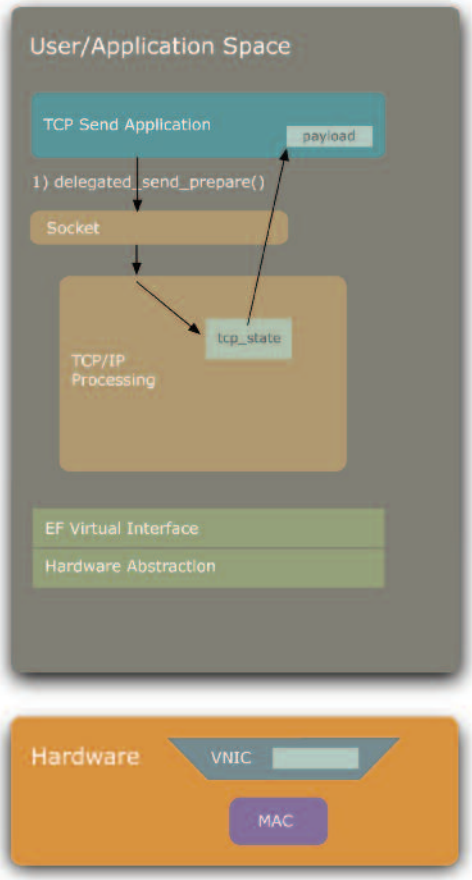
UK +44 (0)1223 477171

HK +852 2624-8868

www.solarflare.com

Appendix 1 – continued

Get TCP State



Copy & Signal to Transmit (critical path)

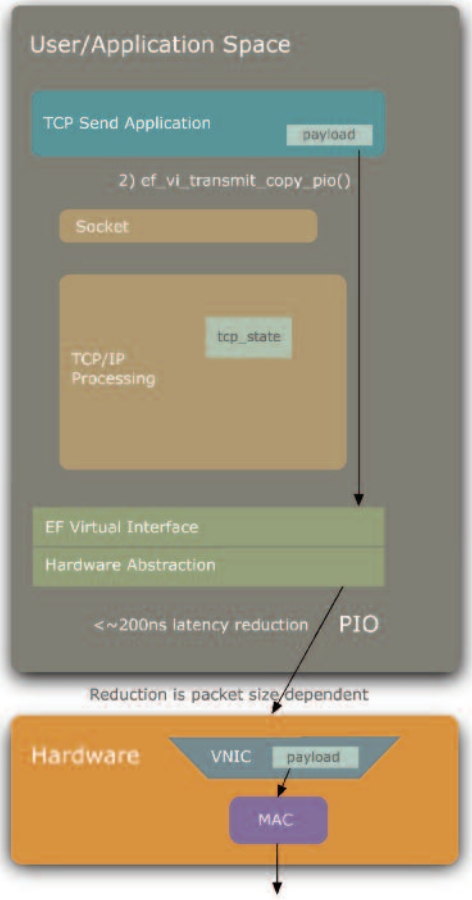


Figure 6. Using the delegated send API when the caller doesn't know the payload in advance, so ef_vi_transmit_copy_pio() is used (continued on next page).

SOLARFLARE®

sales@solarflare.com

US 1.949.581.6830 x2930

UK +44 (0)1223 477171

HK +852 2624-8868

www.solarflare.com

Appendix 1 – continued

Update TCP State/Payload

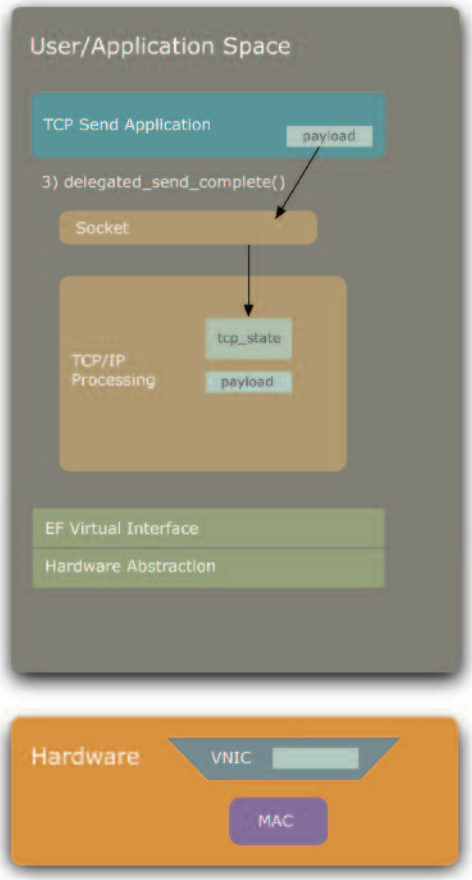


Figure 6 (cont'd). Using the delegated send API when the caller doesn't know the payload in advance, so ef_vi_transmit_copy_pio() is used.

sales@solarflare.com

US 1.949.581.6830 x2930

UK +44 (0)1223 477171

HK +852 2624-8868

www.solarflare.com

SF-114367-TC Issue 2
OOL201502_TB_072415