HAProxy Running with Onload® Sees a 215% Performance Gain

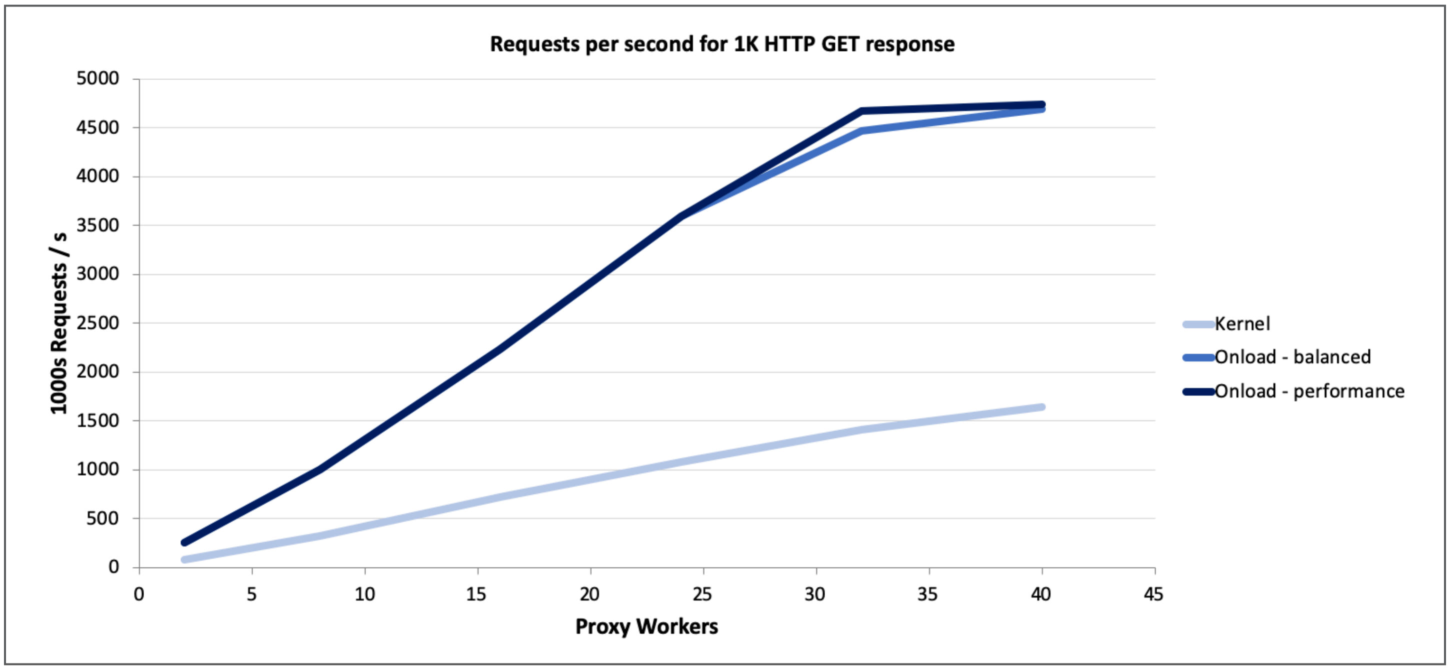
SOLARFLARE® A XILINX COMPANY

HAPROXY

What is HAProxy?

HAProxy is a free, very fast and reliable solution offering high availability, load balancing, and proxying for TCP and HTTP-based applications. It is particularly suited for very high traffic web sites and powers quite a number of the world's most visited ones. It is now shipped with most mainstream Linux distributions, and is often deployed in cloud platforms.

HAProxy is heavily network dependent by design, so its performance can be significantly improved through enhancements to the underlying networking layer.



Key Observations from Performance Testing

- Solarflare's Onload delivers an average performance gain of 360% for HAProxy when measuring connections per second (CPS) across the range of two to 40 workers.
- When processing 1KB GET requests using 40 or fewer HAProxy instances Onload on 100GbE delivers an average performance gain of 215%.
- When processing 10KB Get requests using 40 or fewer HAProxy instances Onload on 100GbE delivers an average performance gain of 95%.
- When looking at the 99th percentile tail latency Onload averaged out at 15% better than the kernel at 2.0ms with a peak value of 3.1ms through 3 million requests per second (RPS). It should be noted that at 1.5 million RPS the kernel latency shot through the roof at 8.8 seconds and kept climbing.

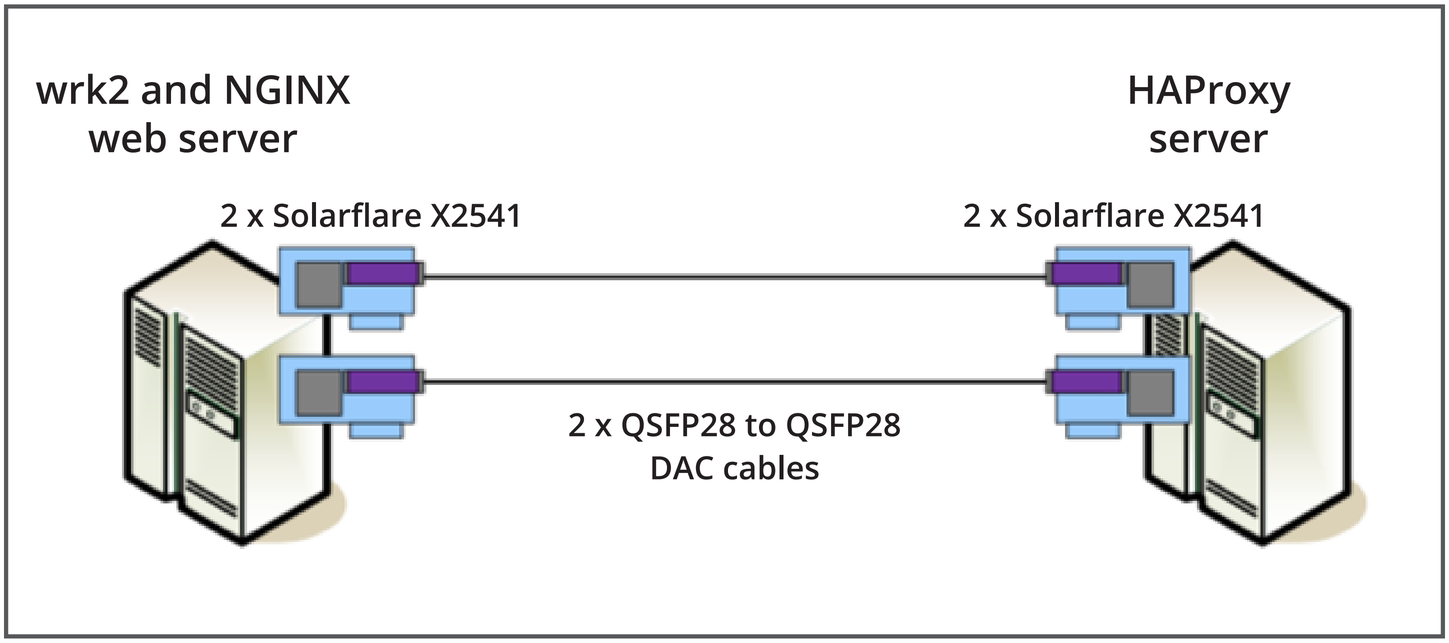
Why HAProxy Benefits from Kernel Bypass

Since HAProxy is network intensive, every request includes network processing overhead. Whenever an application like HAProxy touches hardware, other than the CPU or memory, and in this case the network, it must make at least one, and sometimes several calls to the operating system kernel. Each request is additional overhead that requires both CPU cycles and processing time. Solarflare's Onload moves the network processing required by HAProxy from the kernel into HAProxy's own application space in memory. This single modification improves HAProxy performance by 215% on average as can be seen in the above graph.

Description of Test Platforms

For this testing, we used two Dell EMC PowerEdge R640 dual socket Intel Xeon systems, each with a pair of 100GbE Solarflare X2541 cards. These cards were then connected back to back for this testing.

Both systems had two Intel Gold 6148 CPUs clocked at 2.40GHz with 20 cores per processor, 192 GB of memory.



Tuning Configuration

Below are the changes we made to the standard install beyond simply leveraging Onload:

- Enable clustering (multiple EF_CLUSTER_* options) so that the adapter does the work of spreading the load to different workers.
- Enable socket caching (EF_SOCKET_CACHE_MAX) to improve performance of establishing both accepted and outgoing active connections.
- Use TCP shared local ports feature (multiple EF_TCP_SHARED_LOCAL_PORTS_* options) for outgoing connections further improving performance.
- Use Epoll mode 3 (EF_UL_EPOLL=3) which gives best epoll performance when a large number of sockets are in the epoll set.
- Enable event polling without interrupts within epoll_wait calls to reduce latency and avoid context switches (EF_POLL_USEC and multiple EF_*_SPIN options)
- Use scalable filter mode (EF_SCALABLE_FILTERS and EF_SCALABLE_FILTERS_ENABLE options) to avoid resource constraints when using a very large number of connections.

Observations

HAProxy relies on the operating system's communications stack to process network I/O requests, but in high core count environments, this stack has become the new bottleneck. Here are some additional points to consider:

- The connections per second shows great improvement with Onload, peaking at an improvement of 496% over the kernel stack. With large numbers of proxy workers (32 to 40) the Onload performance begins to level out.
- The requests per second also shows great improvement with Onload, peaking at an improvement of 232% over the kernel stack. With 40 worker processes, results continue to improve, indicating that further performance is available from Onload.
- The throughput shows significant improvement with Onload, peaking at an improvement of 122% over the kernel stack.
- The latency figures output by wrk2, show the time from when the packet should have been sent (according to the configured packet rate), until when the packet was actually received. The 99th percentile figure is reported. When the kernel stack packet rate is raised above 1.5 million requests per second, it can no longer maintain this rate. Jitter increases, the number of outliers exceeds 1%, and so the reported latency suddenly and dramatically increases. Any further small increase in load would make the server appear completely unresponsive to an end user. In contrast, Onload continues to deliver low latency with 3 million requests per second and is actually trending towards even lower latency. The stable and low value for the 99th percentile of latency indicates low jitter and predictable performance.

For More Testing Details

Check out Solarflare's **Onload HAProxy Cookbook** for the exact installation and testing process.

The above benefit statements are the result of benchmarking designed to focus on the value of optimizing networking through Onload kernel bypass. Real world use cases are not the same as benchmarks and as such the role that networking plays may vary, so your overall measurable benefits may be different.

For more information please visit:

solarflare.com

SOLARFLARE®
A **XILINX** COMPANY

Contact Us:

US +1 949 581 6830
UK +44 (0) 1223 477171
HK +852 2624 8868
Email: sales@solarflare.com

SF-122408-CD Issue 3
HAProxy Benchmark Results 112219