WHITE PAPER

AVI Networks®

# Achieving Webscale Elasticity with Modern Software-defined Load Balancers

## INTRODUCTION

The load balancer, also known as application delivery controller (ADC) or reverse proxy, frontends applications and is the invisible face of your application. It accepts connections from end users using browsers and web apps, terminates TLS/SSL (the security protocol providing end-to-end encryption for web transactions), acts as the traffic cop managing connections and requests amongst all the application servers, transforms content, applies policies, and performs a variety of other network functions.

Now imagine any of the following scenarios:

- Your company is an online retailer; Black Friday and Cyber Monday are upon you. You have a successful promo and online traffic to your website suddenly far exceeds your expectations.
- Your business hosts important revenue-generating applications and you are hit by a nasty DDoS attack.
- You are a service provider looking to rollout a new patch or feature to all your customers at a specific time. You expect that the load during this period will be 10 times the normal load.

The load balancer – a critical component of your data center or cloud – has to scale to support your business needs in the aforementioned scenarios. It is the single point of traffic entry to your application, making it the proverbial choke point of your application. As traffic to your application increases above the capacity that you provisioned on the largest load balancing appliance in your infrastructure, what do you do? How do you increase your load balancing capacity even more?

This technical whitepaper outlines new load balancing strategies that will help you handle such abnormal traffic scenarios without adversely affecting end-user experience or breaking the bank. It describes a recent test conducted by Avi Networks that demonstrates elastic scaling from 0 to 1 million SSL transactions per second (TPS), and how you can replicate the test in your own environment.

## ABOUT THIS DOCUMENT

This technical whitepaper provides details on how Avi Networks provides an elastic application services fabric that can scale up or scale down from 0 to 1 million transactions per second with no impact on performance, at a fraction of the cost of a traditional, appliance-based load balancer.

## THE ONSLAUGHT OF ENCRYPTED (SSL/TLS) TRAFFIC AND THE MYTH OF HARDWARE LOAD BALANCERS

Among all tasks that a load balancer performs, SSL/TLS termination is computationally the most expensive. SSL/TLS termination uses asymmetric (public key) encryption for connection establishment and handshake, then switches over to symmetric keys for encrypting the data transfer. Even though the asymmetric encryption during handshake is computationally the most expensive operation, most traffic today has to be encrypted, given that security and privacy concerns are on the rise.

[1] Encrypted internet traffic: a global internet phenomena spotlight, https://www.sandvine.com/trends/encryption.html

Until recently, load balancers required dedicated crypto ASICs and networking hardware to provide the horsepower required for SSL/TLS offload and high volume throughput. However, starting around 2010, due to advances in the x86 processors, memory, and networking hardware, software running on standard x86 servers was able to meet demanding performance requirements. Webscale companies such as Google[2], Facebook[3], and Twitter[4], were able rely on the power of inexpensive x86 hardware to terminate SSL. But the advances in webscale computing remained the forte of the web giants with dedicated teams that were tasked with building the supporting software for their unique needs. Mainstream enterprises still have a hard time building, deploying, and operating scaled out load balancing software on commodity x86 servers. This, of course, meant that hardware load balancer vendors continued to perpetuate the performance myth and charge outrageous prices for network functions that could be performed on standard x86 servers at a fraction of the cost of dedicated hardware appliances.

## THE AVI VANTAGE PLATFORM

So what would the architecture look like for a webscale load balancer that scales?

Let's start by reviewing the different ways to scale load balancers.

### 1. HIERARCHICAL LOAD BALANCER

The simplest way to scale load balancing services is to layer them by chaining services – placing a load balancer (say Tier 1) in front of a group of (Tier 2) load balancers. The Tier-1 load balancer performs a simple Layer 4 (TCP/UDP) flow load balancing to the Tier 2 load balancers. Since Layer 4 load balancers aren't CPU intensive, a single Tier-1 load balancer can handle enough flows for 5 - 10 Tier 2 load balancers. This is usually sufficient performance for most normal sized applications. While this approach is architecturally simple, it is limited by the performance of the Tier-1 load balancer (See Figure 1).

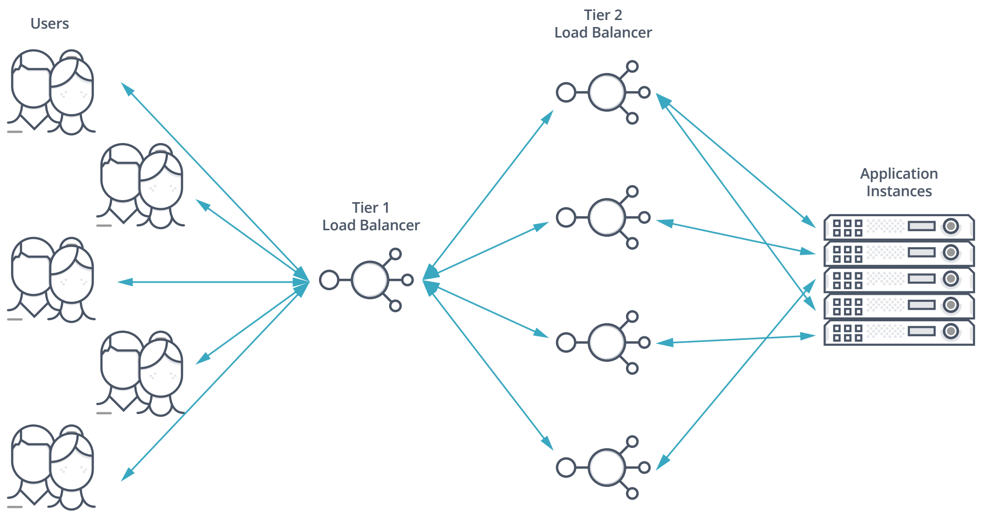


**Figure 1: Hierarchical Load Balancer**

### 2. DNS + PROXY LOAD BALANCER

When users access an application, a DNS lookup of the domain name occurs first - e.g. when a user accesses www.facebook.com, the domain name resolves to an IP address, and the user's browser or app connects to the IP address. Usually, a domain name resolves to a single IP address. However, multiple IP addresses can be associated with a domain name, and the DNS resolver can step through the list of IP addresses and return a different IP address for each DNS query. If four different IP addresses are associated with a single domain name, user 1 will resolve to IP address 1, user 2 to IP address 2, user 3 to IP Address 3, and user 4 to IP address 4. Each IP address can be a load balancer of its own - this just quadruples load balancer capacity. Using the same technique, tens of IP addresses can be associated with a single domain name, with each IP address handled by a separate physical load balancer. While this approach scales well, DNS entries can be cached and have a time-to-live (TTL), during which the same IP addresses are returned to users, even if such IP addresses are stale. (See Figure 2).

[2] Overclocking SSL, 25 Jun 2010 – Nagendra Modagulu et. al. https://www.imperialviolet.org/2010/06/25/overclocking-ssl.html
[3] Expressions of interest in HTTP/2.0, 15 Jul 2012 – Doug Beaver http://lists.w3.org/Archives/Public/ietf-http-wg/2012JulSep/0251.html
[4] Forward Secrecy at Twitter, 22 Nov 2013 – Jacob Hoffman Andrews https://blog.twitter.com/2013/forward-secrecy-at-twitter-0

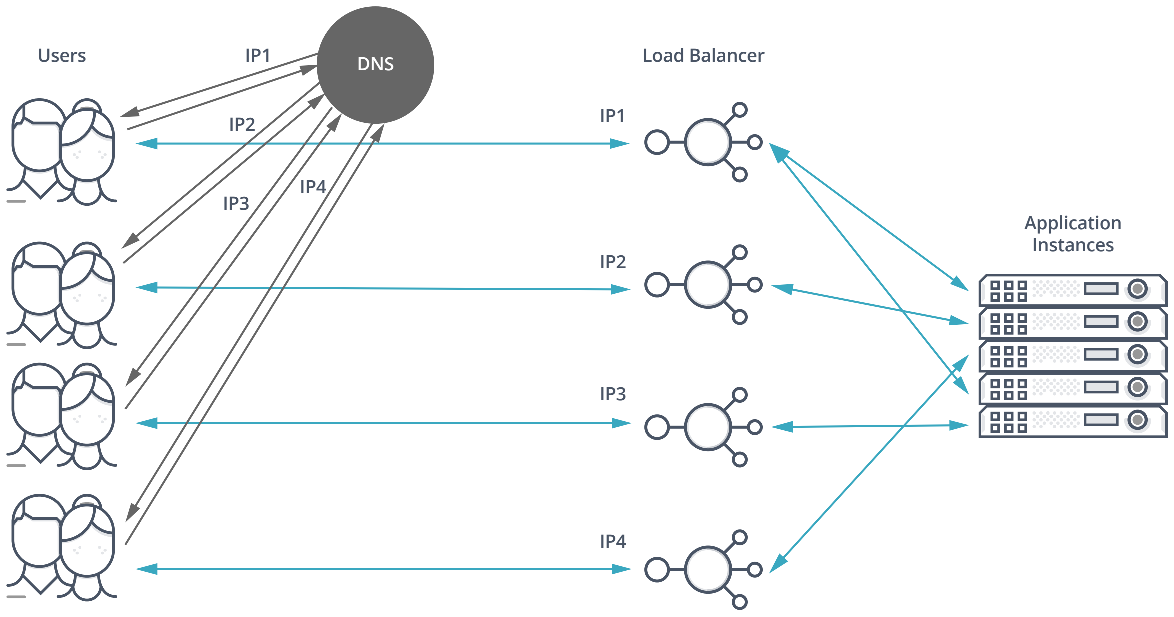


**Figure 2: DNS + Proxy Load Balancer**

## 3. ANYCAST LOAD BALANCER

In this approach, the domain name resolves to a single IP address. The IP address is added to its upstream router with multiple physical load balancers as the next hop. The router performs flow-based equal cost multi pathing (ECMP) and sends every user flow to a different next hop/physical load balancer. Modern routers routinely support at least 64 next hops. This approach scales the best and has none of the limitations of the previous approaches, other than the fact that access to an upstream router is needed (See Figure 3).

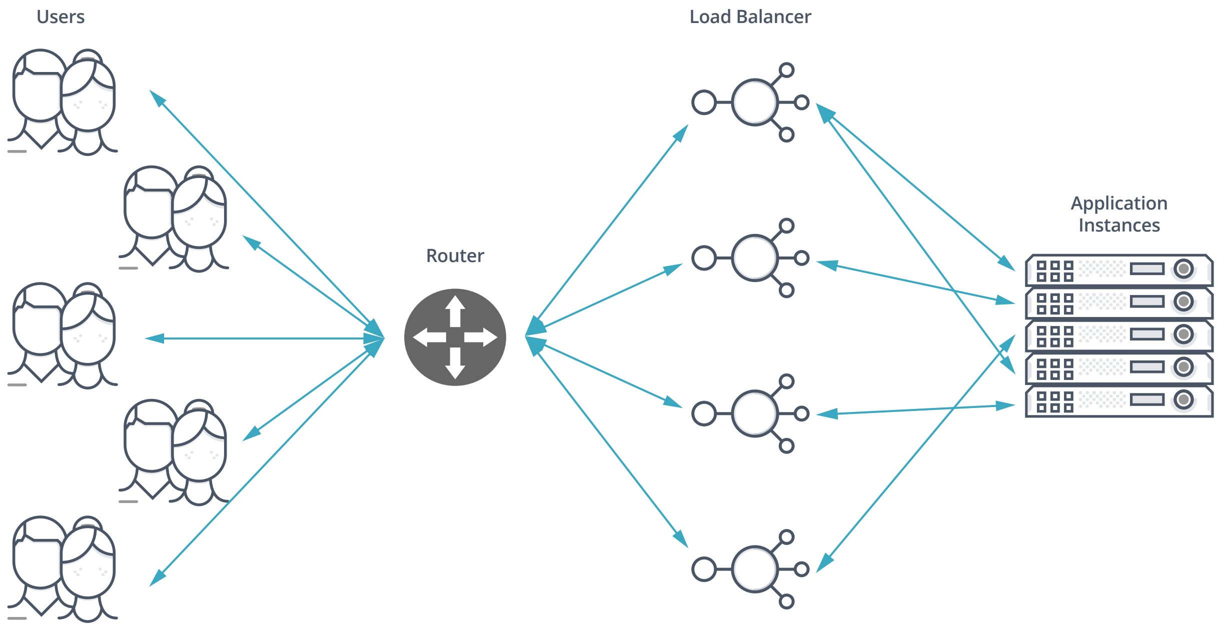


**Figure 3: Anycast Load Balancer**

## SCALING LOAD BALANCERS TO ONE MILLION SSL/TLS TRANSACTIONS PER SECOND

Avi Networks has built an elastic load balancer that scales like those of webscale companies, provides full automation at scale for deployment in your own datacenter or in any public cloud. The Avi Vantage Platform is built on software-defined principles, with separate control and data planes and a central controller that orchestrates a distributed data plane of load balancers (Avi Service Engines). In addition to delivering L4-L7 network services, the Avi Service Engines continuously stream real-time telemetry to the Avi Controller. The platform provides enterprise-class ADC and security features, including world class analytics for everything from aggregate latency and performance for all applications in the environment all the way down to an individual HTTP request for a specific application (See Figure 4).

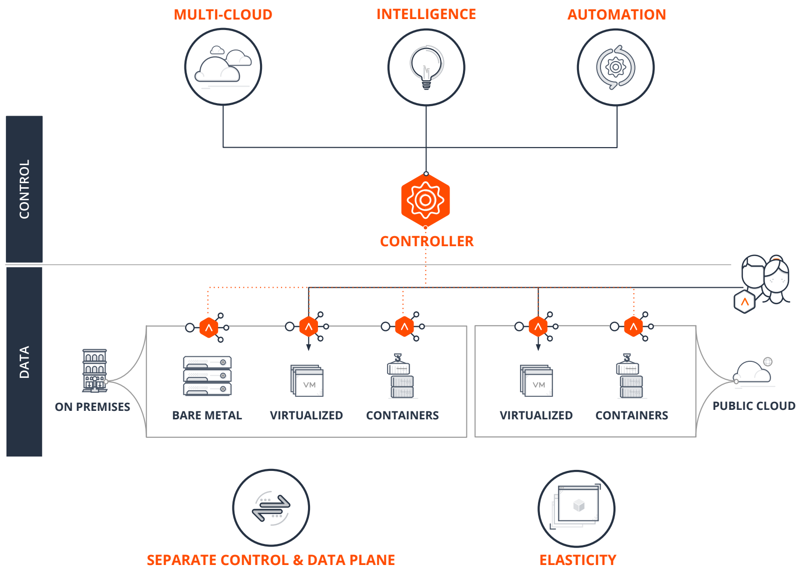


**Figure 4: Scaling Load Balancers**

In response to major retail services provider's request to deploy load balancers for a hybrid cloud deployment model, we set out to see how the Avi Vantage Platform would scale and perform in a public cloud environment using standard virtual machine instances. We can do the same test in any public cloud or in any datacenter, though the test topology and environment will vary depending on the environment. The environment we chose for this initial test was the Google Cloud Platform (GCP).

## SCALABILITY TEST: TOPOLOGY AND CONFIGURATION

GCP supports Anycast and provides APIs to program multiple next hops for a single IP address. This is sufficient to scale the load balancer using the Anycast approach, as described above. In addition, GCP also provides the ability to create up to 32-core virtual machine instances on demand, making compute capacity easily available on demand for the elasticity test. The test used the following topology and configuration (See Figure 5).

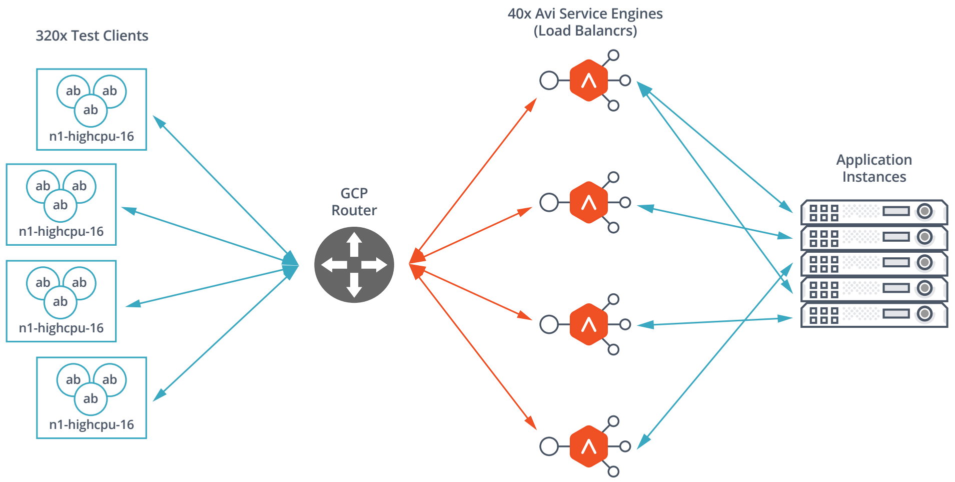


**Figure 5: Topology and Configuration for Scalability Test**

We used GCP virtual machine instances of type ‘n1-highcpu-32’ for Avi Service Engines. These instances provide the highest CPU performance with the highest core count. To generate load using test clients, we used instances of type ‘n1-highcpu-16’. The choice for test clients’ (load generator) instance type was a tradeoff between the time it takes to create such instances (depending on the number of required instances) and keeping them as small as possible.

We also decided to use preemptible instances for test clients; since the cost was a tenth of the cost of non-preemptible instances and there’s no harm done if a few test client instances are terminated; additional test clients can always be created.

## PREPARING THE CUSTOM VM INSTANCES

We needed GCP instances with some pre-installed software packages. We picked CentOS 7 as the base image, installed the docker, psmisc and httpd-tools packages, and created a custom image following the instructions provided in the Google Cloud documentation[5]. Alternatively, we could have installed these packages after creating the VM instances, though that would have simply taken far more time for the instances to become ready, and all the instances would have required publicly routable IP addresses to access software repos. We could have picked Ubuntu as the base image with no material impact on the test or the results.

## INSTANCES, NETWORKS, AND SUBNET

For this test, we created the following VM instances:

- Bootstrap instance - 1 g1-small instance
- Avi Controller - 1 n1-standard-4 instance
- Avi Service Engines (load balancers) - 40 n1-highcpu-32 instances
- Pool server - 1 g1-small instance
- Test clients (load/traffic generators) - 320 n1-highcpu-16 instances

We created a new /23 subnet in a custom network for these instances. You can either pick a subnet that has a sufficient number of IP addresses available or simply create a new subnet.

The virtual IP address is allocated from a subnet that doesn’t exist in GCP; it can be any subnet - public or private (since all traffic is internal), and just a single IP address will be used for a Virtual IP from this subnet.

## AVI CONTROLLER INSTALLATION

We created an n1-standard-4 CentOS 7 image with a public IP and http/https access, then downloaded and installed the Avi Controller on this instance, according to the instructions in the Avi GCP installation guide[6]. Once the Avi Controller was installed, we performed the initial setup.

The Avi Controller provides the orchestration and control plane for a scaled-out load balancer deployment. It performs the following functions.

- Avi Service Engine (load balancer) creation and lifecycle management
- API endpoint and repository of all policy and Virtual Service objects
- Controller instantiates Virtual Services on Avi Service Engines, handles high availability and scale out for Virtual Services
- Cloud Connector task on Avi Controller interfaces with Cloud APIs to program routes and associate virtual IP addresses with interfaces
- Collect, rollup, store, and display metrics from all Avi Service Engines
- Collect, index, and display log analytics

With the Avi Controller providing all the necessary automation, it was easy to deploy and scale webscale load balancing as a software service on commodity x86 servers. The same configuration can be implemented in your own data center or any other public cloud.

The Avi Controller is pre-installed with a 20 core eval license. To obtain an evaluation license for additional cores, please contact us at https://avinetworks.com/contact-us.

[5] Creating, Deleting, and Depracating Custom Images - https://cloud.google.com/compute/docs/images/create-delete-deprecate-private-images
[6] Configuration of Avi Controller and Avi Service Engine instances on GCP - https://kb.avinetworks.com/avi-deployment-guide-for-google-cloud-platform-gcp

## RUNNING THE TEST

We set up a public GitHub repo[7] with the necessary configuration and scripts for anyone to perform the scalability test in GCP. All the configuration necessary is abstracted in config.yaml and the repo describes configurable values and how to customize them.

One of the test choices that we needed to make was the ratio of Avi Service Engines to test clients. Empirically, a ratio of 1:4 was needed to saturate the CPU on the GCP instances running the Avi Service Engines. The Avi Service Engines run a purpose-built software stack that uses high performance DPDK libraries for fast packet processing, bypasses the Linux kernel stack (Linux kernel stack performance limitations[8] are well known) and fully utilizes AES-NI and other crypto special instructions on x86 CPUs to provide the highest possible performance on any commodity server or VM instance.

- **Step 1:** Create the necessary VM instances with a GCP IPAM profile: This step configures the Cloud Connector on the Avi Controller with a GCP IPAM profile and SSH keys to access VM instances on which Avi Service Engines are to be deployed.
- **Step 2:** Create a single pool server: This step starts a pool server.
- **Step 3:** Create Avi Service Engines: This step creates a Avi Service Engine on all VM instances marked for Avi Service Engines. With just a single API call, Avi brings into operation a load balancer that's capable of 1 million SSL/TLS transactions/sec, highlighting the power of automation and orchestration at its best.
- **Step 4:** Create a Virtual Service: This step creates a Virtual Service, instantiates the Virtual Service on all Service Engines; the Cloud Connector invokes GCP APIs to program a route with the Virtual IP as the destination and all Service Engines as next hops for the Virtual IP.
- **Step 5:** Create clients: This step creates VM instances to be used for traffic generation.
- **Step 6:** Start traffic

## VERIFYING THE SCALABILITY TEST WITH REAL-TIME APPLICATION PERFORMANCE ANALYTICS

The Avi Service Engines collect thousands of metrics for every Virtual Service. The Avi Controller collects, aggregates, rolls up, stores, and analyzes these metrics and delivers them either via API or the Avi Controller UI. The following screenshots are from running the test for over an hour (See Figures 6-8).

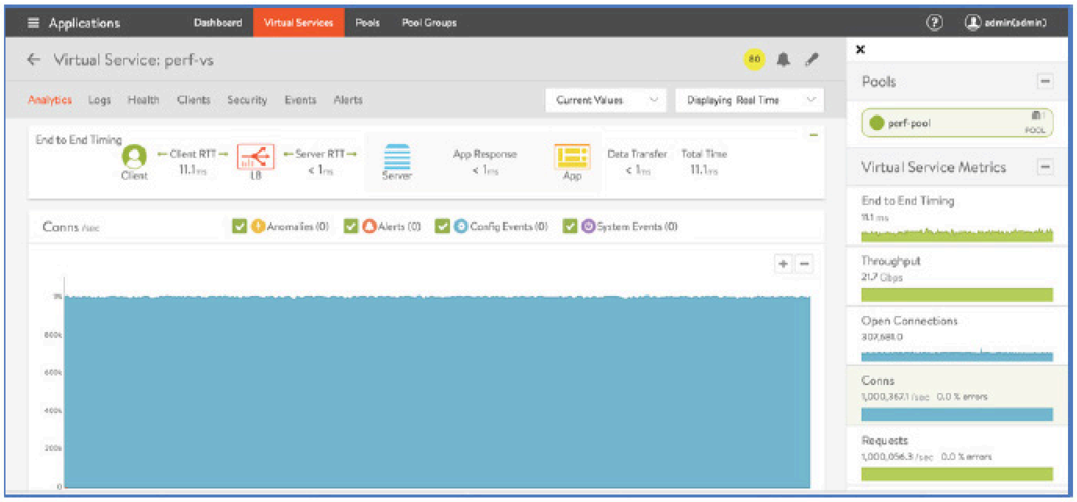


**Figure 6: Virtual Service Dashboard**

[7] https://github.com/avinetworks/avi-test-scripts/tree/master/perf-scripts
[8] Improving Linux networking performance - https://lwn.net/Articles/629155/zz

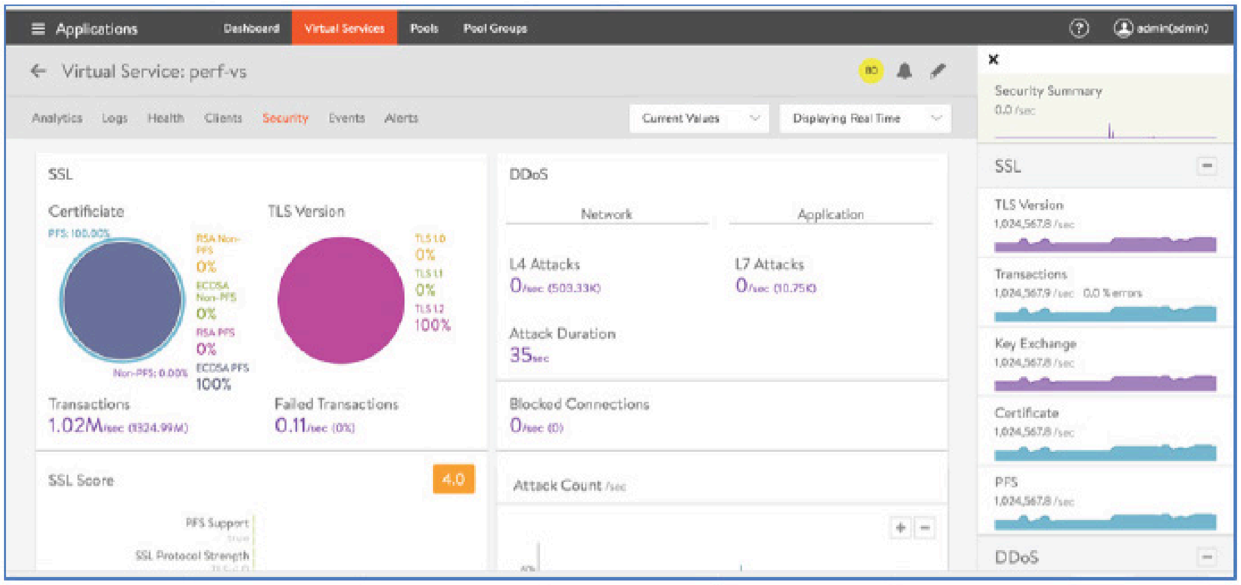
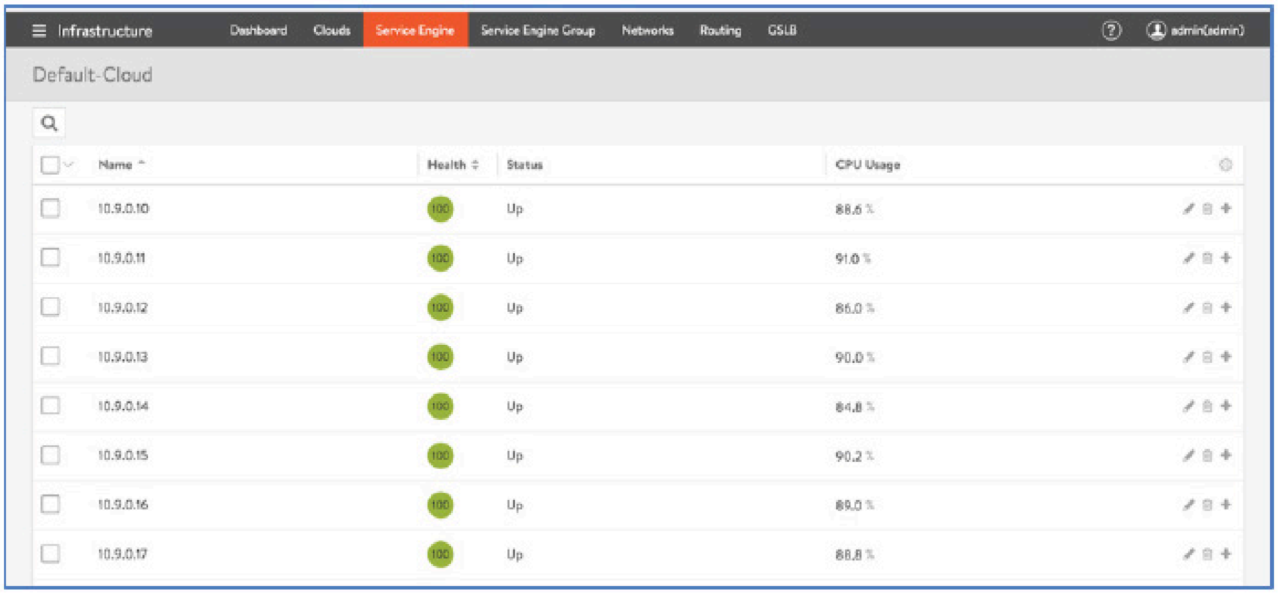


**Figure 7: SSL Analytics Dashboard**



Default-Cloud

| Name | Health | Status | CPU Usage |
|---|---|---|---|
| 10.9.0.10 | 100 | Up | 88.6 % |
| 10.9.0.11 | 100 | Up | 91.0 % |
| 10.9.0.12 | 100 | Up | 86.0 % |
| 10.9.0.13 | 100 | Up | 90.0 % |
| 10.9.0.14 | 100 | Up | 84.8 % |
| 10.9.0.15 | 100 | Up | 90.2 % |
| 10.9.0.16 | 100 | Up | 89.0 % |
| 10.9.0.17 | 100 | Up | 88.8 % |

**Figure 8: Service Engine Analytics Page**

## TIME AND COST COMPARISON

- The total time for the test setup and the steps outlined above: 10 to 15 minutes.
- The time to scale from 0-1 Million TPS after starting test: a few seconds.
- Cloud cost for 40 Avi Service Engine instances: $50 per hour

It was remarkable to be able to create and scale to such webscale capacity within a few minutes. Compare this with the standard hardware purchasing cycle, approvals, lead time, install time, and configuration time for the appliance to provide this class of performance. To make matters worse, once you build up this capacity with hardware, you are stuck with it, depreciating that hardware from that point forward. This one-way scaling is neither cost-effective nor flexible for seasonal changes in traffic.

Actual product cost comparison between Avi and a hardware load balancer that can provide this class of performance shows the hardware load balancer is at least 3.5 to 7 times the cost you would incur with Avi.

## BETTER PERFORMANCE ON BARE-METAL: HYPERTHREADED VS. PHYSICAL CORES

TLS/SSL offload doesn't benefit from HyperThreading. N1-highcpu-32 VM instances consist of 32 HyperThreads or 16 physical cores. This is a single socket on a physical server. Servers typically have at least 2 sockets; hence the compute capacity needed for 1 million SSL TPS is the equivalent of 40 sockets. In a bare-metal-server environment in your own data center, about 15-20 physical servers with 2 sockets each will provide the equivalent compute capacity for a test of this nature.

## KEY TAKEAWAYS

The test provides several insights:

- Proprietary hardware is no longer a requirement for load balancing in the most demanding environments, particularly with the ever-increasing SSL/TLS traffic. In fact, software can scale to better performance than custom hardware can ever provide.
- A highly scalable and elastic load balancing solution with automation, orchestration and analytics similar to that used by webscale companies is now available for deployment by enterprises in any environment.
- An architecture like this can come in handy when you have to handle a DDoS attack – scaling elastically helps handle the attack without adversely affecting service levels while operators use the Avi security dashboard to identify and mitigate the attack.
- As enterprises move towards hybrid clouds with a mix of datacenters and public cloud deployments, new load balancing and traffic management architecture is required. Avi Networks provides the best performance, scale, automation, analytics, scale, and cost-effectiveness for enterprises.