
vmware®
AVI
Networks®

VMware - Avi - Diligent

Table of Contents

Chapter 1. Document Information

This document includes the Overview of what Avi can offer to Diligent Deployment.

- Overview of Avi Vintage deployment in Vmware vSphere
  - VMware Support in Avi Vantage
  - VMware User Role for Avi Vantage
  - Avi Service Engine Group Options

1.1. This is a Draft Document based on the meeting onsite dated Nov 7th, 2019 between VMware - Avi and Diligent

Chapter 2. Originator

Remo Mattei

Solutions Architect

VMware .Inc

2.1. Owner

VMware and Diligent – Confidential. Restricted Distribution

2.2. Confidentiality

All information supplied to Diligent for the purpose of this project is to be considered Diligent confidential.

2.3. Contact Information

Table 1. VMware - Avi Contact Information

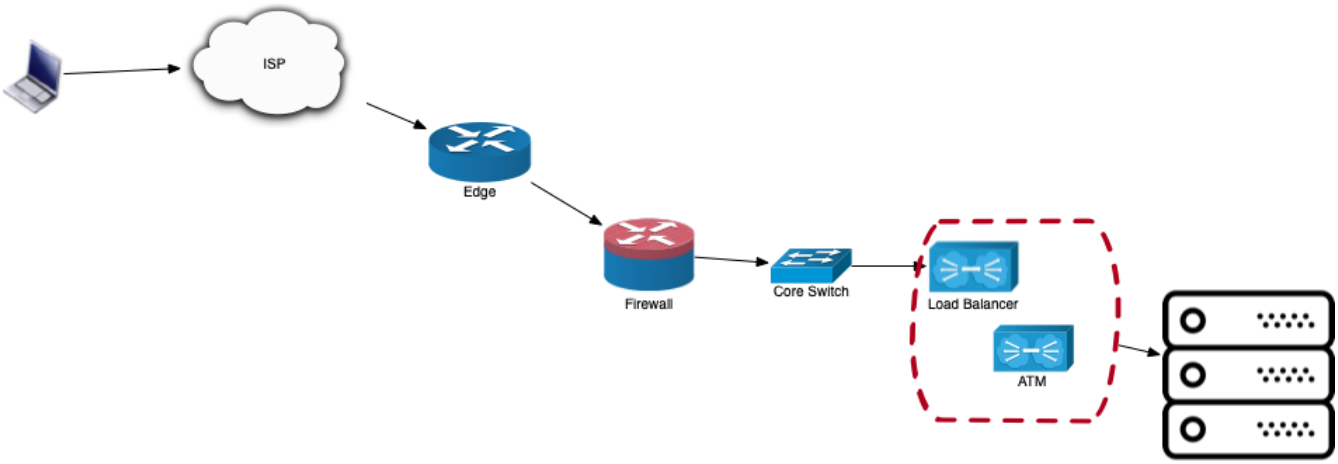
Name	Title	Phone	E-mail
Remo Mattei	Solutions Architect	+1-801-808-8649	rmattei@vmware.com
Justin Osserman	Regional Sales Specialist	+1-347-256-3311	josserman@vmware.com
Cody Arthur	Business Development Representative	+1-707-329-4407	codya@vmware.com

2.4. ATM use at Diligent

Diligent uses ATM from the F5 as following:

- Gateway
- Packet validation - inspections
- Firewall

Avi can act as Gateway, as well as a firewall for specific protocols. We cannot do Packet validation - inspections.



2.5. VMware User Role for Avi Vantage (VMware vCenter 6.5)

2.5.1. Overview

Avi Vantage deployed in a VMware cloud with write access mode requires certain permissions. This article discusses setting up the required user role and permissions for Avi Vantage vCenter cloud.

Vmware User Role - Full Article

2.6. Configuration Used

2.6.1. Overview

Avi Vantage was deployed in Denver, where we enabled SSH. We reserved a block of IPs

- Avi Controller Version: 18.2.6
- Controller IP: 10.74.78.211
- NETWORK: 10.74.78.0/23
  - Reserved to Avi: 10.74.78.211 through 10.74.78.222
- Datacenter: Denver
  - Deployed on HOST: ???

2.7. Cloud Configuration

2.7.1. Overview

Avi Vantage was deployed in Denver, where we enabled SSH. We reserved a block of IPs

- VMware: Read and Write mode
- User: Juan accounts'
- Created an IPAM with the network 10.74.78.212 through 10.74.78.222

- NO DHCP on VMware side
- DHCP for SEs

2.8. Virtual Service Scaling

2.8.1. Overview

This article covers the following virtual service optimization topics:

- Scaling out a virtual service to an additional Avi Service Engine (SE)
- Scaling in a virtual service back to fewer SEs
- Migrating a virtual service from one SE to another SE
- Avi Vantage supports scaling virtual services, which distributes the virtual service workload across multiple SEs to provide increased capacity on demand, thus extending the throughput capacity of the virtual service and increasing the level of high availability.
- Scaling out a virtual service distributes that virtual service to an additional SE. By default, Avi Vantage supports a maximum of four SEs per virtual service when native load balancing of SEs is in play.In BGP environments the maximum can be increased to 32.
- Scaling in a virtual service reduces the number of SEs over which its load is distributed. A virtual service will always require a minimum of one SE.

Advanced Options - Full Article

2.9. Troubleshooting Vmware / Avi

2.9.1. Overview

Installation was simple but we had an issue where the SE would not deploy to vSphere.

Requirements

vSphere will need to have port 22 open in order for Avi Controller to deploy the Service Engine (SE)

Deployed an SE in a no access mode to test and see if vSphere needed SSH access.

2.10. Diligent Testing Process and Results

2.10.1. Overview

Based on the instruction provided by Justin, in the word doc we will record the results below:

Test Methodology
The Proof of Concept (PoC) objectives described in this document are intended to match general use cases for local load balancing against the vendor Device Under Test. This ensures a base level of

compatibility for existing deployed load balancer use cases and for future environments which may be dependent on more flexible deployment architectures. This document may be used as is, or may be modified without limitation as desired by customers of Avi Networks. This document should serve as a framework for establishing the validity of load balancer functionality and performance. The tests included are commonly used across verticals and are intended to prove relevant to a wide audience. Customers are encouraged to extend this document with additional tests that more accurately reflect the unique nature of their requirements. Specific tests may stress functionality of the ADC, while others focus on performance. Performance testing is highly dependent on the capacity of an entire system, including client, server, load balancer, and interconnecting network. In some cases performance may be extrapolated based due to capacity restrictions of the test environment. To achieve the objectives outlined in the following test cases, clients and servers are configured to validate the configuration and completion of the required functionality. Test topology utilizes three networks for client, server, and centralized DUT management. The specific configuration for the Device Under Test (DUT), client, and server are outlined in the Appendix. This document is focused on standard HTTP and HTTPS protocols. These tests may be applied to most any environment, but is primarily oriented to standard north-south client-server applications. For help crafting testing tests for more diverse ecosystems or east-west traffic, reach out to a local Avi field system engineer.

2.10.2. Load Balancing

Validate the successful distribution of traffic across multiple servers using common LB algorithms. For these tests, persistence, connection multiplexing, connection ramp, passive health monitors and similar features that might alter the default behavior of the LB algorithms are disabled. See the knowledge base article on Load Balancing Algorithms for help validating results and further descriptions for how each algorithm works. When a load balancer is utilizing multiple CPU cores, each core independently keeps track of the load balancing it has previously done. The results may seem inaccurate when multiple cores are in use. Though it is possible to filter the results to see each processor, for the purposes of this test a single vCPU core load balancer is recommended. See the knowledge base article Non Uniform Traffic Distribution. The following configuration changes should be performed prior to running the Load Balancing functional tests. These configuration changes are not required in production deployments. Ultimately, a number of features play a role in determining the next best server for load balancing. To isolate and test only the LB algorithm, extraneous features are disabled for the test. A few of these changes are related to different behavior of load generators versus real clients and servers. See the knowledge base article Testing with Load Generators for more explanation of the differences and why these changes are be important.

- Create a Service Engine with 1 vCPU core. Validate the new VS is deployed on the test Service Engine.
- Create a VS
  - VS > Settings: Attach the System-HTTP application profile.
  - VS > Analytics: Enable Real Time Metrics (box is checked, duration is set to 0 (infinite))
  - VS > Analytics: Enable Collect Non-Significant Logs (box is checked, duration set to 0 (infinite))
- Create a pool with two or more servers attached

- Pool > Settings: Disable Passive Health Monitor
- Pool > Settings: Persistence set to none
- Pool > Advanced: Set the Connection Ramp time to 0

Load Balancing: Round Robin

```
ab -c 1 -n 100 http://VIP/
```

Avi

Set the Pool > Settings > Load Balance to the Round Robin algorithm.

Validate

```
Within the VS > Logs page, enable the green Non-Significant logs option.  Select the
time frame of the test on the log histogram ribbon.  Click on Server IP Addresses
under the Log Analytics sidebar menu to see that each server received the same number
of requests.
To validate the ordering of each request, expand logs from the list to see which
server each request was sent to.
```

RESULTS:

We will put the results here

Load Balancing: Ratio

Validate ratio based load distribution across the pool of 2 servers. One server should receive 4 times the number of connections than the other.

- Server-A = 1
- Server-B = 3

Client

```
ab -c 1 -n 500 http://VIP/
```

Avi

Set the Pool > Settings > Load Balance algorithm to Round Robin. Within the Pool > Servers page, set the Ratio of Server-A to 1, and the ratio of Server-B to 4.

Validate

```
Within the VS > logs page, enable the green Non-Significant logs option.  Select a
time frame of the test on the log histogram ribbon for the duration of the test.
Click on "Server IP Addresses" under the Log Analytics sidebar menu.  Server-A should
have received 20% of the requests while Server-B received 80%.
```

RESULTS:

> We will put the results here

Load Balancing: Server Disable

Servers often need to be taken out of rotation for routine maintenance. In order to minimize disruption, a server must be able to be removed from the list of eligible servers while still enabling existing client sessions on the server to remain long enough to complete their transaction. This enables the server to be gracefully disabled over a configurable period of time.

Description

Servers often need to be taken out of rotation for routine maintenance. In order to minimize disruption, a server must be able to be removed from the list of eligible servers while still enabling existing client sessions on the server to remain long enough to complete their transaction. This enables the server to be gracefully disabled over a configurable period of time

Client

```
Via browser:  http://VIP/
Use incognito mode or cleared browser cache to ensure accurate results
```

Avi

```
Set the Pool > Settings > Persistence to IP Address.
Navigate to the Pool > Server tab.  Select the checkbox for the server the browser is
currently connected to.  Select Gracefully Disable, with a 1 minute timer.
```

Validate

```
Access the site, recording the server IP address.
Disable the server the browser is connected to.
Force refresh the page, validating the server IP has not changed.
After 1 minute, force refresh the page again, noting the new server page.  Note that
standard browser behavior is to cache the page and only send 304 HEAD requests, rather
than refresh the page.  Force refresh (Ctl-F5 for Windows) tells the browser to skip
its cache and fetch the page again.
```

RESULTS:

```
We will put the results here
```

2.10.3. Persistence

Persistence is the ability to ensure a user sticks to the same server for a given duration of time. For application sessions, users may not be load balanced to a server that does not have previous session information. Persistence can take many forms depending on the application type. This test focuses on the two most common methods of persistence. * Modify HTTP Profile **Change Templates > Application > System-HTTP profile to disable Connection Multiplex. * Modify Pool** Change Pool > Settings > Load Balance to Round Robin load balancing algorithm.

Persistence: HTTP Cookie

Description

Validate HTTP requests are persisted to a server for the duration of the session. For this test, connection multiplexing is enabled, which enables load balancing of client HTTP requests (in addition to connections) across servers.

Client

```
From a new browser window using incognito mode (or cleared browser cache), open
http://VIP/
```

Avi

```
Change Pool > Settings > Persistence to System-HTTP-Cookie
Change the Pool > Settings > LB Algorithm to Round Robin.
```

Validate

```
1.  Access site via new browser session.  Validate through the logs that requests were
distributed to both servers.  Close the browser.
2.  Change Pool > Settings > Persistence mode to enable HTTP Cookie.
3.  Access site via a new browser session.  Validate through the logs that requests
were persisted to a single server.
4.  Change the Pool > Servers > server ratio to 10 for the server that was not
selected.
5.  Closing the browser and reopening will wipe the cookie and make the next requests
eligible to be sent to a new server.
```

RESULTS:

```
We will put the results here
```

Persistence: Client IP

Description

Validate TCP connection is persisted to a server for 5 minutes. Open a connection to a server. Close the connection. Reconnect again and validate the same server was chosen for the same client.

Client

```
Via browser:  http://VIP/
```

Avi

```
Change Pool > Settings > Persistence to System-Client-IP
```

Validate

```
1.  Access site via new browser session.  Validate through the logs that requests were
distributed to both servers.  Close the browser.
2.  Change Pool > Settings > Persistence mode to enable Client IP.
3.  Access site via a new browser session.  Validate through the logs that requests
were persisted to a single server.
4.  Change the Pool > Servers > server ratio to 10 for the server that was not
selected.
5.  Closing the browser (which terminates the TCP connection) and reopening within 5
minutes will continue to persist.
```

RESULTS:

We will put the results here

2.10.4. Health Monitor

Health monitors are required to validate the successful response from servers prior to sending clients to those servers. Monitors can be passive, which monitor live client traffic, or active, which proactively send a check periodically from the load balancer. See the knowledge base article Health Monitor Overview for more examples

Health Monitor: HTTP

Description

DUT sends a custom HTTP query to the server, validating acceptable response codes and/or content within the response

Avi

```
Create an HTTP virtual service and pool.  Create a new Templates > Health Monitor >
HTTP monitor with the following settings listed below.  Apply these settings and
validate one at a time.
1.  Use GET /health.html HTTP/1.0 as the resource for health checking
2.  Response Code:  2xx, 3xx
3.  Server Response Data:  Welcome
4.  Maintenance Server Response Data:  NotFound
```

Validate

```
Validate each setting by viewing the VS > Pool > Servers page.  When drilling into the
page for an individual server, expand the HTTP health monitor in the monitors table to
see more details about the results.
1.  Validate response codes 2xx marks server up
2.  Validate response code 404 marks server down
3.  Validate response content "Welcome" marks server up
4.  Validate response content "NotFound" marks server down
```

RESULTS:

We will put the results here

Health Monitor: TCP

Description

DUT sends a custom TCP query to the server, validating response data.

Avi

```
Create a new Templates > Health Monitor > TCP monitor, applied to the pool, with the
following settings listed below.  Apply the pool to a virtual service (HTTP or TCP).
Set the Client Request Data field to:  GET /1b.txt HTTP/1.0\r\n\r\n
Set the Server Response Data field to:  200 OK
```

Validate

```
The virtual service should remain green when the monitor is applied.  Optionally
navigate to the Pool > Servers table to drill into an individual server health,
including the response data returned by the server.
```

RESULTS:

We will put the results here

2.10.5. Acceleration

Validate the use of standard HTTP compression via GZIP between client and DUT, with compression disabled between DUT and server. Compression of text based objects generally reduces the object size by about 75%, improving end user experience and reducing network bandwidth.

Acceleration: HTTP Compression

Description

Test the ability to compress HTTP traffic.

Client

```
Via browser:  http://VIP/
```

Avi

```
Edit the Templates > HTTP Application Profile > System-HTTP.  In the Compression tab,
check the Enable Compression box.
Edit the VS > Settings page and ensure the VS is using the System-HTTP for the
Application Profile.
Edit the VS > Analytics page and enable Collect Non-Significant Logs.  Optionally add
a filter with Log All Headers enabled to view the Accept-Encoding headers sent and
received.
```

Validate

```
Navigate to the VS > Logs tab.  Enable Non-Significant Logs to include non-error
traffic.  Add the following filter to the search field:  compression_percentage>0
Expand a log that matches the compression filter.  The middle column will contain a
line that says Compression followed by a number that denotes the percentage of
reduction.
```

RESULTS:

We will put the results here

2.10.6. SSL Termination

SSL traffic flowing through the load balancer can be terminated prior to sending to the server. Terminating SSL has the benefits of consolidating certificate and key management, enforcing policies for cipher and versions, visibility for intermediate devices such as taps, and generally better performance by offloading CPU overhead from application servers. The following SSL Termination tests can use same virtual service configuration. Create a VS with:

- TCP/UDP profile set to the default System-TCP-Proxy
- Application profile set to System-HTTP

Enable SSL with the System-SSL profile and a self-signed SSL certificate of type RSA (the default when the VS is created via Basic mode).

SSL Termination: HTTP to HTTPS Redirect

Description

Redirect HTTP traffic to HTTPS with a 302 redirect.

Client

```
Via browser:  http://VIP/
```

Avi

```
Modify the SSL VS by adding a second port, 80, with SSL checkbox disabled for this
port.  The 443 port remains with the SSL checkbox enabled.
```

Validate

```
Via browser:  http://VIP/   The connection should be HTTP.
Via browser:  https://VIP/  The connection should be HTTPS.
Switch the VS application profile to System-Secure-HTTP, which by default has the HTTP
to HTTPS checkbox enabled.
```

RESULTS:

> We will put the results here

SSL Termination: Specified Cipher

Description

Manually define the allowed SSL ciphers and versions. When a security flaw is exposed for a cipher or version, it is imperative to be able to disable the vulnerable cipher / version. LB must show which clients are connecting with which SSL settings. This is critical to enforce protections from SSL attacks that target vulnerable components of SSL.

- Validate which ciphers are most commonly used by clients.
- Limit the supported ciphers to only include those that support Perfect Forward Secrecy.
- Verify that clients are still able to negotiate the new SSL ciphers.

Client

```
** Via browser:  http://VIP/
** Via client CLI:  ab -Z AES128-GCM-SHA256 -n 10 -c 10 -r https://VIP/
```

Avi

```
To change the ciphers, navigate to Templates > Security > SSL Profile.  Create a new
profile, selecting only ciphers that have the PFS checkbox.  Apply to the virtual
service.
```

Validate

```
   Via browser:  http://VIP/
   Navigate to the DUT VS > Logs page.  Select the SSL tile from the Log Analytics on
the right.  Note the ratio of requests made over PFS and which cipher / encryption
protocols used.
   Switch the VS to the new SSL profile.
   Via browser:  http://VIP/
   Via client CLI:  ab -Z AES128-GCM-SHA256 -n 10 -c 10 -r https://VIP/
   Via client CLI:  ab -Z ECDHE-ECDSA-AES128-SHA256   -n 10 -c 10 -r https://VIP/
(for PFS)
   Refresh the VS > Logs page.  Select the Significance tile from the Log Analytics
on the right.
```

Any errors listed under SSL Error.

RESULTS:

We will put the results here

SSL Termination: SSL to Server Re-encryption

Description

Traffic between the client and the DUT is encrypted HTTP, with the SSL decrypted at the DUT. Traffic is evaluated as unencrypted HTTP, then sent on the wire to servers as encrypted HTTPS.

Client

```
Via browser:  http://VIP/
```

Avi

```
Modify the following options in the pool:
* Default Server Port set to 443
* SSL to Backend Servers set to enabled
* SSL Profile set to System-Standard
```

Validate

```
Clients continue to access the site via SSL port 443.  From the DUT logs, verify that
HTTP specific data is viewable, such as Hostname and URI.  If the traffic was not
decrypted by the DUT, this data would not be viewable.  Validate traffic is sent to
servers via port 443.
Run TCP dump on the client and server NICs to verify traffic is encrypted on the wire.
See Packet Capture for help.
```

RESULTS:

> We will put the results here

2.10.7. Advanced LB

Applications commonly require advanced content switching to provide HTTP redirects or rewriting of content and headers. The DUT should also be able to load balance traffic across multiple pools based on rules. The goal is to ensure a broad set of customization, while constraining the complexity of solution to guarantee manageability by any operator.

Advanced LB: Content Switching

Description

Send all requests for *.jpg to an images pool, all other requests to the default server pool.

Client

```
Via browser:  http://VIP/
```

Avi

```
Create a standard HTTP VS.
Create two pools:

** Pool-A:  First pool contains all but one server.  This is the default pool for the
VS.
** Pool-B:  Second pool contains only the remaining server.

Create an HTTP Request Policy for the VS:
** Match Rule:  Path
*** Criteria:  Ends with

*** Group:  .png

** Action:  Content Switch
*** Pool:  Pool-B
```

Validate

```
From the VS > Logs page, add the following filter (path ends with .png):
uri_path~=".png"
Select the Server IP Address tile from the Log Analytics on the right.  This should
show only the image server.  Alternatively the search criteria could simply be "Pool-
B", then look at the URIs shown for the filtered logs.
Remove the existing filter and replace it with (path does not end with .png)
uri_path!~=".png"
Select the Server IP Address tile from the Log Analytics on the right.  This should
show only servers in Pool-A.
```

RESULTS:

We will put the results here

Advanced LB: HTTP Redirect

Description

Issue a 301 redirect for requests for http://VIP/* to http://VIP:8080/newpath/*

Client

```
Via browser:  http://VIP/
```

Avi

```
Create a standard HTTP VS.  Edit the VS, adding a second port of 81.
The custom path generation is described in the knowledge base article VS Policies.
Create an HTTP Request Policy for the VS:
* Match Rule:  Service Port is 80
* Action:  Redirect
   ** Port:  8080
   ** Status Code:  301
   **  Path:  newpath/path[0:]
```

Validate

```
Access the VS via a browser.  Verify the session has been redirected to port 8080 and
newpath/ is prepended.
```

RESULTS:

We will put the results here

Advanced LB: Proxy Pass

Description

Rewrite all requests to the VIP address to internal.test.com. Rewrite all responses back to the VIP address.

Client

```
Via browser:  http://VIP/
```

Avi

```
Create a standard HTTP VS.
Set the VS > Advanced > Host Name Translate to: internal.test.com
```

Validate

```
Via browser:  http://VIP/
From the DUT VS > Logs page, expand a log and select All Headers.  Expand the Request
and Response headers.  Modified headers should be marked in orange and should include
the Host header on the request and Location header on the response.
```

RESULTS:

> We will put the results here

2.10.8. Security

Security encompasses a wide array of capabilities, such as protocol protection, DDoS mitigation, and best practices for administering a system. After each test in this section, be sure to remove the altered configurations so as to not impede later tests.

Security: Connection Throttling

Description

Limit the number of connections within a time period a client may make. Excess connections should be classified as a DoS attack and be reset. (In a production environment, excess connections are often silently discarded, but many load generators have an easier time dealing with TCP resets.)

Client

```
* ab -n 1000 -c 100 -r http://VIP/
* Via browser:  http://VIP/
```

Avi

```
Create a standard HTTP VS.
Edit the Template > Profile > Application profile used by the VS.  Set the type to
HTTP.  In the DDoS tab, modify the Rate Limit Connections from a Client.
    Threshold:  10 connections
    Time Period:  300 sec
    Action:  Send TCP Reset
```

Validate

```
Launch Apache Bench to generate traffic.  Simultaneously access the site with a
browser.  The browser, which opens 6 connections or less from its IP address, should
not be inhibited.  Apache Bench attempts to open many more connections and will be
denied after the first 10 connections.
Navigate to the VS > Security tab, focusing on the DDoS section.  Click the DDoS
metric tile on the right bar to drill into the DDoS and see the offending clients.
Remove the configuration changes before proceeding to the next tests.
```

RESULTS:

> We will put the results here

Security: Black List

Description

Discard connections from a client that is marked as black listed.

Client

```
Via browser:  http://VIP/
```

Avi

```
Create a standard HTTP VS.
Edit the VS > Policies > Network Security, adding a new rule.
    Rule Name:  blacklist
    Client IP Address:  <IP of client browser>  (This may also be a new IP Group for
larger lists that are reusable)
    Action:  Deny
```

Validate

```
With the rule in place the client browser should fail to connect.  Disable the rule to
validate client browser can now connect.
From within the VS > Logs, add a filter for "blacklist" to see client connection
attempts that were blocked by the rule.
```

RESULTS:

We will put the results here

Security: Client Authentication

Description

Enforce client authentication prior to allowing clients to access secure parts of a website.

Client

```
Via browser:  http://VIP/
Via browser:  http://VIP/secure
```

Server

```
As of ADC Test Plan v9, this test is not yet implemented in the Avi Client/Server
image.
```

Avi

```
Create a standard HTTP VS.
Edit the VS > Advanced, enabling the HTTP Basic Authentication.
* Set the Include URL to /secure
* For the Auth Profile, create a new profile with the following settings:

**  Name:  authprofile
**  LDAP Server:  Use an IP address of one or more of the test servers
**  User ID Attribute:  use
```

Validate

```
Access to the site requires no additional security.
Accessing URL paths that begin with /secure require authentication.  Use root and
password to validate the authentication.
```

RESULTS:

We will put the results here

2.10.9. Analytics Tests

The ability to ascertain the success of a configuration is critical to guaranteeing the uptime of applications. The load balancer must be able to provide actionable data detailing the health of an application, the network, and components in between. It should not require an external ecosystem of products to determine if the DUT is working optimally.

Analytics

The following scenarios encompass common challenges for network and application operators to demonstrate how a load balancer can contribute to issue discovery and resolution. The following tests are configured with basic load balancing to generate a sampling of data as outlined in 2.1.1 Round Robin Load Balancing.

- Create a VS:
  - VS > Settings: Attach the System-HTTP application profile.
    - Templates > Application > System-HTTP profile: Disable Connection Multiplex
    - VS > Analytics: Enable Real Time Metrics (box is checked, duration is set to 0 (infinite))
    - VS > Analytics: Enable Collect Non-Significant Logs (box is checked, duration set to 0 (infinite))
- Create a pool
  - Include two or more server
  - Pool > Settings: Disable Passive Health Monitor
  - Pool > Settings: Persistence set to none
  - Pool > Advanced: Set the Connection Ramp time to 0

Analytics: End User Experience

Description

Validate the success of an end user's interaction with an application. Triage a slow application to determine source of slowness.

Client

```
ab -c 1 -n 100000 http://VIP/128k.txt
```

Server

```
Shortly after the test has begun, add network latency to one of the servers.
If using the Avi client-server image, this can be done using the Linux tc command
```

Avi

```
Create VS as outlined above.
```

Validate

```
Within the VS > Analytics page observe the End to End timing graph.  Move the cursor
over the most recent time in the graph, observing the Client RTT, Server RTT, App
Response, Data Transfer and Totals in the popup box.  Show which is the source of the
application slowness.
```

RESULTS:

We will put the results here

Analytics: Basic VS Metrics

Description

View common metrics for a VS including throughput, open concurrent connections, new connections and new requests per second. Validate granularity of metrics down to a per server level.

Client

```
ab -c 1 -n 100000 http://VIP/
```

Server

```
Add network latency to one of the servers as configured in the previous 3.1.1 test
If using the Avi client-server image, this can be done using the Linux tc command
```

Avi

```
Use the VS configured in 3.1.1
```

Validate

```
1  Within the VS > Analytics page, observe the following metrics by clicking on the
appropriate tile from the metrics bar on the right.
* Throughput
* Open Connections
* New Connections
* Requests

2  Navigate to Pools > poolname and select the Servers tile from the Pool Metrics bar
on the right.  (This is different than the Servers tab in the top menu)  All servers
should be in a tight configuration in the top right corner of the Health / Request
chart.
* Edit the Pool, changing the Load Balance algorithm to Fastest Response.
* Refresh the servers chart, rerunning the client test traffic if necessary.  The
server with induced network latency should be sent less traffic than its peers.

3  Select the Servers tab from the menu near the top of the page.  This table contains
a list of servers and their status.  Clicking the sprocket icon on the top-right of
the table allows additional metrics to be included in the display.  Click into any
server to see additional granularity of data.
```

RESULTS:

We will put the results here

Analytics: TLS Version & Cipher

Description

New vulnerabilities to SSL and TLS are a common occurrence. It is important for administrators to stay current by disabling support for vulnerable configurations. However, many companies do not disable deprecated SSL and TLS versions due to lack of visibility into the client utilization of outdated SSL. In this test, clients connect via various TLS versions, which much be visible to the administrator.

Client

That this test uses curl, rather than Apache Bench used in many other tests. Extend this test using various browsers to see the default settings they will negotiate. The following curl request can be copy and pasted in a single operation.

```
curl https://VIP/ -k --tlsv1.1; curl https://VIP/ -k --tlsv1.1; curl https://VIP/ -k
--tlsv1.2;
```

Avi

Edit the VS created in the previous 3.1.1 test.

```
   Add a new Service Port for 443 SSL.  (This is in addition to the existing 80 port)
   Set the Application Profile to System-Secure-HTTP  (this is similar to System-HTTP
with the HTTP to HTTPS redirect enabled)
   Set the SSL Settings > SSL Profile to System-Default
   Set the SSL Settings > SSL Certificate to System-Default-Cert
```

Validate

```
1  Navigate to VS > Security page.  Set the Display Time to Real Time.  Note the
distribution in the TLS Version pie chart.
2  For SSL / TLS ciphers, navigate to the VS > Logs page.  Select Non-Significant Logs
to display non-errored requests.  From the Log Analytics bar on the right, select SSL.
The cipher types most commonly used are displayed in Encryption Protocol.  For more
granularity, type "ssl_cipher" into the search bar.  Follow the options presented
(such as '=') to create a custom cipher search.  Select Client OS or Browser from the
Log Analytics tab to see the breakdown of real clients (browsers, not load generators)
that are using the filtered cipher.
```

RESULTS:

We will put the results here

2.10.10. Logging

Basic metrics are useful for viewing and understanding the health of a system at a macro level. But many issues require deeper granularity to determine root cause. An ability to trace a session or similarly correlate data in a visual context can save significant time over poring through raw TCPdumps or relying on numerous external points of data. The following scenarios can utilize the same configuration used in section 3.1.3, with a basic VS configured for SSL termination. The following configuration changes should be made.

- Edit the existing VS:
  - Under the Analytics tab, ensure the Collect Non-Significant Logs is enabled with a time set to 0 (infinite). Check the Log All Headers box.
- Edit the existing pool:
  - Set the Load Balance algorithm to Round Robin

Logging: Root Cause Errors

Description

Sift through client requests to quickly root cause common issues. While the use cases are endless, this test focuses on missing object on a web server.

Client

```
Via browser:  http://VIP/
Force refresh this page a number of times.  Be sure to refresh more times than you
have pool servers.
```

Server

```
On one server, remove /logo.png
```

Validate

```
Navigate to VS > Logs tab.  By default, only 'Significant' logs are shown, which are
generally errors.  Enable the Non-Significant Logs on the top right of the histogram.
This includes good (non-error) traffic logs.
    Click the Significance tile on the metrics tile on the right.  This may require
expanding the Log Analytics on the top right.
    Within the Significance dimensions popup, select Request Ended Abnormally
    Most of the filtered logs should show the Response as 404.  Click the URL Path
from the metrics tile on the right Log Analytics bar.
This will show a list of objects that generated 404s or similar errors.  These
requests can be a factor degrading the Performance Score of a VS health score.  Clear
the filter before proceeding to the next test.
```

RESULTS:

We will put the results here

Logging: Browser TLS Cipher

Description

Similar to test 3.1.3, new vulnerabilities to SSL and TLS ciphers are a common occurrence. Client browsers may prioritize different ciphers. This test demonstrates the correlation of browsers to SSL / TLS ciphers which is valuable when determining when to deprecate a cipher.

Client

Note that this test uses curl, rather than ApacheBench used in many other tests. Extend this test using various browsers to see the default settings they will negotiate. The following curl request can be copy and pasted in a single operation.

```
    ab -k Z AES128-GCM-SHA256 -n 1000 -c 10 -r https://<VIP>/128k.txt
    Via browser:  http://VIP/12k.htm
```

Validate

```
Navigate to the VS > Logs page.
 Select Non-Significant Logs checkbox.
 From the Log Analytics bar on the right, select SSL.  The cipher types most commonly
used are displayed in Encryption Protocol.
 For more granularity, type ssl_cipher into the search bar.  Follow the options
presented (such as '=') to create a custom cipher search.  Select Client OS or Browser
from the Log Analytics tab to see the breakdown of real clients (browsers, not load
generators) that are using the filtered cipher.  This enables administrators to make
informed decision of which clients are using a cipher, if it is chosen at all, and
whether it can be disabled.
 Ciphers can be enabled / disabled via the Templates > Security > SSL Profile >
System-Standard
```

RESULTS:

> We will put the results here

2.10.11. Alerting

Alerts provide the backbone of proactive notification for administrators. An inline device such as a load balancer must be able to provide flexible notifications of conditions and translate those notifications into actions. Alert conditions should be flexible enough to cover every range of ecosystem integration and network criteria. Actions should include traditional mechanisms, such as local alerts, syslog, SNMP, and email. More modern actions include an ability to initiate API calls to third party alert monitoring solutions, custom scripts, and ability to scale the capacity of application server infrastructure via integration with virtualization orchestrators. Since these require a significantly more elaborate test bed, they are not included in this test plan.

Alerting: Custom Alert

Description

In this scenario, when a server goes down while other servers in the pool are still up, the load balancer should generate a low priority alert. This test can be extended by adding a second, high priority alert when all servers in the pool are down. It can also be extended by taking proactive action, such as initiating the provisioning of a new server.

Avi

```
* Edit the pool, adding a System-HTTP health monitor
* Navigate to Operations > Alerts > Alert Config
* Create new Alert Config

** Set Object to Virtual Service
** Set Instance to the name of the test POOL
** Set Event Occurs to Server Down
** Set Alert Action to System-Alert-Level-Low (by default, this only generates an
alert and takes no further action)
```

Validate

```
From the CLI of one server, type:  apachectl -k stop
To validate the alert, navigate to the VS > Analytics page.  The alert will be shown
as a red icon on the bottom of the main chart.  This may require switching to Real
Time.  The alert will also show in the VS > Alerts page, and in the bell icon on the
top right of the VS page.
```

RESULTS:

> We will put the results here

2.10.12. Performance Tests

Often performance testing is more of an art than a science. It is very difficult to ascertain when the client, server, or DUT is the bottleneck in the chain. It is always recommended to have significantly greater client and server capacity, ensuring only the DUT is stressed. For more suggestions on ways to optimize performance test beds, see the knowledge base article Testing with Load Generators.

Load Generators

Connections Per Second

Validate the number of new TCP layer 4 connections per second that can be established. This test includes opening the connection, sending a simple request, receiving a response, and a graceful TCP close (FIN) of the connection

Connections Per Second: L4 TCP Fast Path

TCP connections per second are established as layer 4 connections using Fast Path. The DUT is inline with the traffic, but is not a full TCP proxy. Clients send a request and receive a server generated response. Connections are closed gracefully using a TCP FIN.

Client

```
ab -n 1000000 -c 100 -r http://<VIP>/128b.txt
```

Avi

```
Create a VS with the TCP/UDP profile set to System-TCP-Fast-Path.
The Application profile should be set to System-L4-Application.
```

Validate

```
AB client should low to no error rate.
Avi UI:  Select the Virtual Service > Analytics > New Connections metric tile to show
results for the test period.
```

RESULTS:

```
1 Core:
2 Core:
8 Core:
```

Connections Per Second: L4 TCP Proxy

Description

TCP connections per second are established as layer 4 connections using TCP Proxy. The DUT is inline with the traffic, and is terminating client and server TCP connections as a full TCP proxy. Clients send a request and receive a server generated response. Connections are closed gracefully using a TCP FIN.

Client

```
ab -n 1000000 -c 100 -r http://<VIP>/128b.txt
```

Avi

```
Create a VS with the TCP/UDP profile set to the system default TCP Proxy.  The
Application profile should be set to System-L4-Application.
```

Validate

```
AB client should low to no error rate.
Avi UI:  Select the Virtual Service > Analytics > New Connections metric tile to show
results for the test period.
```

RESULTS:

1 Core:
2 Core:
8 Core:

2.10.13. Requests Per Second

Validate the number of HTTP requests per second. This test includes opening a TCP connection, sending 10 requests in serial over the connection prior to closing the connection gracefully and starting again.

Requests Per Second: HTTP

Description

Validate the number of HTTP requests per second. This test includes opening a TCP connection, sending 10 requests in serial over the connection prior to closing the connection gracefully and starting again.

Client

```
ab-mr -k -N 10 -n 1000000 -c 100 -r http://<VIP>/128b.txt
```

Avi

```
Create a VS with the TCP/UDP profile set to the default System-TCP-Proxy.
The Application profile should be set to System-HTTP.
```

RESULTS:

1 Core:
2 Core:
8 Core:

2.10.14. Throughput

Validate the amount network throughput per second through a virtual service and through a load balancer. Generally the traffic through the load balancer will be twice that of the traffic through the virtual service. That is because the load balancer is measuring traffic on the client side of the connection and traffic on the server side of the connection. See the knowledge base article Definition of Throughput for a full definition. The typical bottleneck in throughput tests are the maximum packets per second sustainable by the hypervisor or the max throughput of the network interface cards used by the DUT.

Throughput: TCP

Description

Validate the VS and load balancer throughput for a TCP application. Clients in this test attempt to retrieve large objects.

Client

```
ab-mr -k -N 100 -n 1000000 -c 100 -r http://<VIP>/128kb.txt
```

Avi

```
Create a VS with the TCP/UDP profile set to System-TCP-Proxy.
The Application profile should be set to System-L4-Application.
```

Results

```
1 Core:
2 Core:
8 Core:
VS scaled across 2 load balancers with 2 core each:
```

Throughput: HTTP

Description

Validate the VS and load balancer throughput for a HTTP application. Clients in this test attempt to retrieve large objects.

Client

```
ab-mr -k -N 100 -n 1000000 -c 100 -r http://<VIP>/128kb.txt
```

Avi

```
Create a VS with the TCP/UDP profile set to the default System-TCP-Proxy.
The Application profile should be set to System-HTTP.
```

Results

```
1 Core:
2 Core:
8 Core:
VS scaled across 2 load balancers with 2 core each:
```

Throughput: SSL

Description

Validate the VS and load balancer throughput for an SSL terminated HTTP application. Traffic to backend servers is clear text HTTP. Clients in this test attempt to retrieve large objects. Alternate ciphers may be used. Below are a few examples of ciphers that could be specified in the load

generator: • RSA (No PFS): AES128-GCM-SHA256 • RSA (PFS): ECDHE-RSA-AES128-GCM-SHA256 • Elliptic Curve (PFS): ECDHE-ECDSA-AES128-GCM-SHA256

Client

```
ab-mr -k -N 100 -Z AES128-GCM-SHA256 -n 1000000 -c 100 -r https://<VIP>/128kb.txt
```

Avi

```
Create a VS with the TCP/UDP profile set to the default System-TCP-Proxy.  The
Application profile should be set to System-HTTP.  Enable SSL with the System-SSL
profile and a self signed SSL certificate of type RSA.  If an Elliptic Curve cipher is
used by the load generator, be sure to use an EC certificate rather than an RSA cert
on the load balancer.
```

Results

```
1 Core:
2 Core:
8 Core:
VS scaled across 2 load balancers with 2 core each:
```

Throughput: HTTP POST

Description

Validate the VS and load balancer throughput for a HTTP application. Clients in this test attempt to POST or send large objects to the server.

Client

```
ab-mr -k -N 10 -b 256000 -n 100000 -c 10 -r -p /tmp/ramdisk/128k.hml -T
\'application/x-www-form-urlencoded\' http://<VIP>/uploaded-file
```

Avi

```
Create a VS with the TCP/UDP profile set to the default System-TCP-Proxy.
The Application profile should be set to System-HTTP.
```

Results

```
1 Core:
2 Core:
8 Core:
VS scaled across 2 load balancers with 2 core each:
```

2.10.15. SSL Transactions Per Second

Validate the number of new SSL, or more specifically, TLS transactions per second. This test expects no TLS session reuse. The most common bottleneck in this test is the ability for clients to generate enough SSL connections. For the DUT, the most common bottleneck is CPU, which is saturated around 90 percent. For a broader description of performance testing SSL, see the knowledge base article SSL Performance.

SSL Transactions Per Second: RSA

Description

Validate the number of new RSA TLS transactions per second. Key size is 2k. This test expects no TLS session reuse

Client

```
ab -Z AES128-GCM-SHA256 -n 1000000 -c 100 -r https://VIP/128b.txt
```

Consider twice the number of CPU cores for clients as are allocated for the DUT.

Avi

Create a VS with the TCP/UDP profile set to the default System-TCP-Proxy. The Application profile should be set to System-HTTP. Enable SSL with the System-SSL profile and a self signed SSL certificate of type RSA.

Validate

From the VS analytics pages, navigate to the Security tab > Transactions to validate TPS are new sessions, not reused. Navigate to Key Exchange metric tile to validate all transactions are RSA

Results

1 Core: 2 Core: 8 Core: VS scaled across 2 load balancers with 2 core each:

SSL Transactions Per Second: Elliptic Curve

Description

Validate the number of new Elliptic Curve TLS transactions per second. Key size is 256 bit (SECP256R1). This test expects no TLS session reuse.

Client

```
ab -Z ECDHE-ECDSA-AES128-GCM-SHA256 -n 1000000 -c 100 -r https://VIP/128b.txt
```

Consider twice the number of CPU cores for clients as are allocated for the DUT.

Avi

```
Create a VS with the TCP/UDP profile set to the default System-TCP-Proxy.  The
Application profile should be set to System-HTTP.  Enable SSL with the System-SSL
profile and a self-signed SSL certificate of type EC.
```

Validate

From the VS analytics pages, navigate to the Security tab > Transactions metric tile to validate TPS are new sessions, not reused. Navigate to Key Exchange metric tile to validate all transactions are Elliptic Curve.

Results

```
1 Core:
2 Core:
8 Core:
VS scaled across 2 load balancers with 2 Core each:
```

2.10.16. System Tests

System testing comprises the management and maintenance of the load balancer infrastructure. These tests can be extended based on the proposed deployment architecture, management interaction, or other factors relevant to the production environment.

Management Access

Validate the DUT supports multiple tenants, Roles Based Access Control, and remote authentication for administrators.

Multiple Tenants

Description

System must support an ability to separately partition the load balancer. For example, a single fabric may support production and non-production environments uniquely. There are numerous variations of how tenancy could be deployed. For the purpose of this test, the same load balancers may be used to host both tenants. A user must be configured with minimal rights to the first tenant (such read only), and full admin rights to the second tenant.

Avi

```
From Administrator > Tenants, create a new tenant.
From Administrator > Users, create a new user, populating the name, username, and
password.  From the Tenants and Role section on the right, select the Add Tenant and
choose the admin tenant and choose Application-Operator for the role.  Add a second
tenant, choosing the newly created tenant from this test, with the role set to
Application-Admin.
```

Validate

```
Logout of the DUT and log back in using the newly created local user account.
The user should have access to switching tenants, regardless whether the tenants are
on the same or different load balancers.
Objects created in the new tenant should not be exposed in the initial / default
tenant.
```

2.10.17. Availavility

Validate the DUT provides robust high availability of the system to protect against failures. The DUT should also be able to scale or adjust to traffic loads to accommodate changes in utilization or volumetric attacks.

High Availability

Description

Validate the successful recovery from a catastrophic failure when an active load balancer fails.

Client

```
ab -n 1000000 -c 1 -r http://<VIP>/1k.txt
```

Avi

```
Create a new Infrastructure > SE Group.  If lab resources are constrained, disable any
existing VS and edit the existing SE Group instead.
    Set the High Availability Mode to Active/Active
    From the Advanced tab, set the SE Failure Detection to Aggressive
    Create a basic HTTP VS
    From the Applications > Dashboard page, set the view to VS Tree mode.  Expand the
new VS and validate it is running on two Service Engines.  If a new SE must be spun
up, this may take a few minutes in Write Access mode, or in Read or No Access modes
the administrator must manually create the additional SEs.
```

Validate

```
Initiate the AB test from the client.
    Disable a load balancer, such as deleting a VM or removing the power from the LB
server
    Validate the disruption took place while AB is still running and the test
continues successfully
    Only a small number of test connections should fail
```

Scalability

Description

When the infrastructure reaches a capacity limitation, the load balancer should non-disruptively

increase additional capacity, either automatically or via administrative command.

Client

```
ab -n 1000000 -c 1 -r http://<VIP>/1k.txt
```

Avi

```
Use the same setup as test 5.2.1, with two Service Engines in Active/Active HA and one
VS.  If Avi Vantage is not deployed in Write Access mode in a virtual environment:
    Create a new Service Engine with similar network connectivity to the existing SEs
    Validate the SE is shown as green in the Infrastructure > Service Engine page
Test:
    Initiate AB test traffic
    Navigate to the Applications VS > [VS-name] > Analytics page
    From within the VS Analytics page, mouse over the VS name
    Validate the VS is currently hosted by two SEs
    Click Scale Out.  After the scale out messages have completed, validate the VS is
hosted by three SEs
Validate:
    Edit the VS, navigating to the Analytics page.  Enable the Collect Non-Significant
Logs box and save
    Navigate to the VS > Logs page.  Enable the green Non-Significant Logs button on
the top right
    Select Service Engine from the Log Analytics menu bar on the right
    Validate all Service Engines are handling client traffic
```

Validate

```
Initiate the AB test from the client.
Validate the load balancer shows it has traffic flowing through multiple devices to
increase scale
```

2.10.18. Appendix

The following section defines the test bed, including optimizations necessary to achieve relevant and accurate test results.

Environment

The network, and underlying infrastructure powering the test can change dramatically depending on the environment the test is run within. Avi Vantage can be deployed on bare metal servers, on virtual machines, within containers, and in public clouds. Each of these environments has differing functionality, performance, and network topologies. This test plan was loosely based on a VMware environment, which is the most common and accessible for testing. Client, server, and DUT virtual machines may be deployed with any amount of resources, provided it is adequate to successfully accomplish the desired goals of the test. Since many tests are performed with various size DUTs, it is recommended to create multiple load balancers, or Service Engines. An application can be

configured once, then placed on different sized load balancers. This allows multiple data points to understand the scalability of performance when increasing resources.

See Figure 1

Figure 1. Virtual Machines

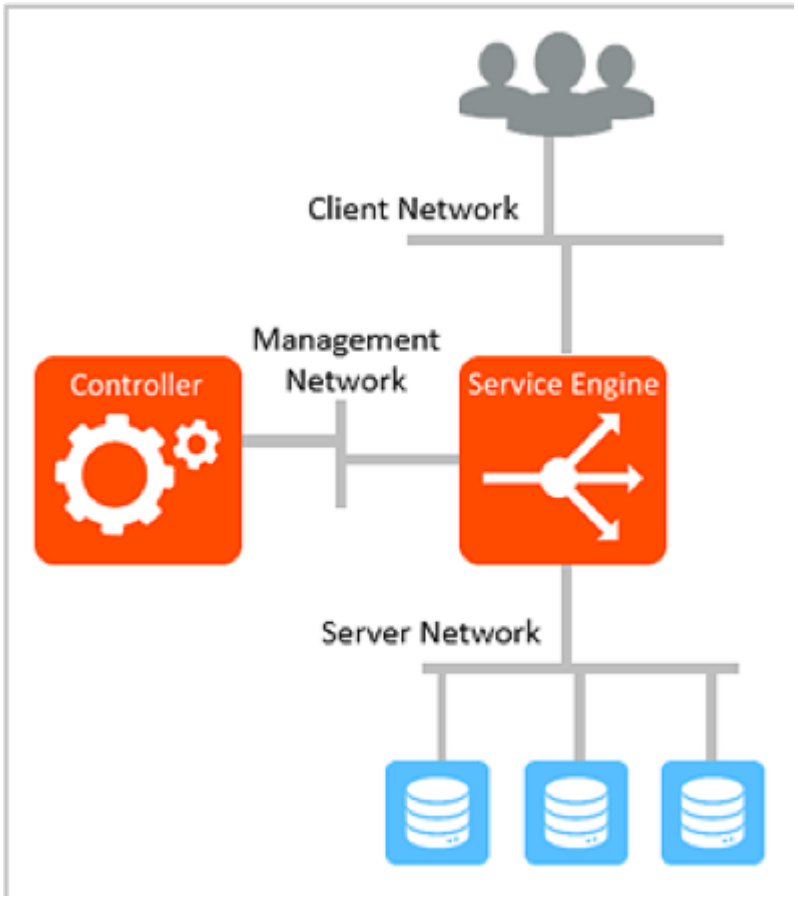


Figure 2. Client Traffic NATed

2.10.19. Client

Any client may be used for generating traffic. This test plan document references Apache Bench, an

open source load generation tool that is freely available for Linux systems. Avi's recommendation is to leverage the Client-Server virtual machine that may be downloaded from the Avi Portal site. This virtual machine includes necessary optimizations to improve the performance of AB, documented below.

Performance

Each AB instance will use a separate thread, which will use a separate CPU core. Regardless how many ABs are run on a single server, they all use the same network interface, which has a limit to the number of source ports free at a time. For this reason, performance may often degrade if more than four ABs are run from a single physical server. Avi generally recommends at least 3x the amount of CPU allocated to the client compared to the DUT. This ensures the client is not the bottleneck, which can often be difficult to prove. So for a 4 CPU core load balancer, Avi recommends 3 client servers, each running 4 AB instances, each with its own CPU core.

Additional Tools

In addition to the stock AB load generator, the Avi client-server image includes two additional tools that can be leveraged for testing. ab-mr: The client-server image provided on the Avi portal includes a standard Apache Bench as well as a modified version of AB, called ab-mr, for "multiple-requests". This modified AB can specify the number of requests per TCP connection, which better emulates client browsers. This capability is not available in the trunk branch of AB, which will only send one HTTP request per connection or an unlimited number of requests independent of the connections. A limitation of this enhanced ab-mr is that it is not able to accurately only send one request per connection. With a few more improvements to this AB variant, Avi expects to publish this code back to the open source community. The second additional tool is a python script which enables the ability to send commands simultaneously to multiple AB instances at a time. Apache Bench is single threaded, so it has a limit to the amount of CPU and the number of ports it is able to use. For higher capacity performance tests, many AB clients are required. Generally this involves multiple clients per server, and multiple client servers. By executing a single AB command via this tool, all AB clients can be instructed to send traffic at the same time.

Optimization

Below are a list of other enhancements made to improve the performance of the client. • sysctl -w net.ipv4.ip_local_port_range=2048 65000 (Adds to the pool of available source ports so client can open more connections) • ulimit -n 90000 (Opens additional file descriptors) • net.ipv4.tcp_tw_recycle = 1 (Tells clients to reclaim ports more aggressively) • net.ipv4.tcp_tw_reuse = 1 (Tells clients to reclaim ports more aggressively) When executing AB commands, follow the same format as described in online AB documentation.

Replace the <VIP> with the IP address or name of the virtual service to be tested. To manually terminate a running test, press CTRL+C Below are a list of common AB flags used for test traffic within this test plan. See the Apache Bench site for an extended list of options for AB. -c Concurrency Number of multiple requests to perform at a time. Default is one request at a time. -n Requests Number of requests to perform for the benchmarking session. The default is to just perform a single request which usually leads to non-representative benchmarking results. Note that AB does not provide an ability to specify the number of requests over a connection. For better emulation of real client traffic, which opens a finite number of requests before closing a

connection, use the modified ab-rm variant. -Z Ciphersuite Specify SSL/TLS cipher suite (See OpenSSL ciphers)

-k Keepalive Enable the HTTP KeepAlive feature, i.e., perform multiple requests within one HTTP session. Default is no keepalive. -r Errors Don't exit on socket receive errors.

Device Under Test

The Device Under Test, or DUT, for this test is the load balancer. This test plan places emphasis on Avi Network's Vantage load balancer, though the test scenarios are intended to be industry standard to the extent that a standard exists. In many cases, the configurations listed for the DUT are system defaults. Too easily performance tests are based on configurations that would never be supported in production environments with standard clients or browsers. This is not the intent of Avi's ADC test plan. The goal is to provide realistic results based on configurations and tuning that would be valid in any production environment. If optimizing the performance to the maximum extent possible is the desired goal, it is recommended to work with an Avi technical engineer to help craft a more custom test configuration. Below are some optional configuration tunings for the load balancer to be tested. • Edit VS > Advanced: Set the Weight field to 8. For load balancers that will be supporting many virtual services, it is important that each VS is given equal access to buffers. When only a single application is being tested, it can help to give it highest priority with full access to resources, particularly send queues in virtualized environments. • Templates > Security > SSL Profile: Disable the Send "Close Notify" Alert. This is a graceful SSL close, which is optional in production environments and ignored in test environments. If packets per second are the bottleneck in a test, this will eliminate an additional, unnecessary packet from each SSL transaction.

Server

Any HTTP server could be used for this test with minimal changes required. The tests are requesting specific objects such as 128k.txt, which is fairly easy to create on any standard server. Avi Networks provides a Client-Server virtual machine on its Avi Portal site which is configured with a simple Nginx web server. The server has the required html objects already, and is ready to go for the testing. The following modifications have been made to improve the server performance and results of the testing. • .html files have been placed onto a RAM disk, which minimizes the disk seek times and latency this may incur. • Logging has been disabled on the server, which insures disk I/O is not a factor causing latency. If the server infrastructure is causing delays or becomes the bottleneck in the performance tests, this will show up as Server RTT in the Avi Vantage end to end timing graphs. If this is the case, try adding additional server resources. If it is unclear if the server is the bottleneck or if Vantage is the bottleneck, consider temporarily replacing the server with a DataScript. The DataScript is a simple script which allows Vantage to respond directly to client requests. If the performance test results improve when using the DataScript, it is likely the servers did not have enough capacity. If the results are the same, it is likely the DUT is at capacity. Place the following DataScript code into a new DataScript's request event field and attach to the virtual service. This example is applicable to a request per second or SSL TPS test. Variations of this rule can be used for throughput or other tests.

```
avi.http.response(200, {content_type="text/html"}, "<html><body>Hello
Avi!</body></html>")
```