appgate

Appgate SDP® - Performance and Scaling

Type: Technical guide

Date: June 2021 (v3)

Applies to: v5.3

TABLE OF CONTENTS

Executive Summary 2
Background 2
Performance and Scale 3
    Appgate SDP – A Different Approach 4
Benchmarking 5
Controller 6
    Test Setup vs. Real Life 6
    Controller Performance Metrics 7
    Physical & Virtual Appliance Performance 9
    Use of Multi-Controller Groups 15
    Controller Summary 15
Gateways 16
    Gateway Performance Metrics 17
    Physical Ax Appliance Performance 19
    Virtual Appliances Performance 25
    URL Access 30
    Use of HA Gateways 31
    Gateway Summary 31
Client 32
    Benchmark Testing 32
Connector 34
    Connector Performance Metrics 34
    Physical Ax-M performance 35
    Use of HA Connectors 38
    Connector Summary 38
Conclusion 39
About Appgate 39

EXECUTIVE SUMMARY

For enterprises to adopt the Software-Defined Perimeter as their chosen philosophy and not just treat it as a Cloud security side-show there is one critical litmus test that must be passed - the performance test. Any system which will become the heart of an enterprise's connectivity model has to be able to scale massively. It must handle the countless sessions that users and IoT devices create from both inside and outside traditional networks. Any such system also has to be able to handle many 10s of gigabits per second of throughput whilst utilizing the highest levels of encryption. Lastly the resources the system demands to perform at these levels should allow the use of cost-effective off-the-shelf platforms – both physical and virtual.

This paper explores many different aspects of the attainable performance envelope of Appgate SDP and details the likely resource requirements to achieve the performance required to deploy successfully within your own enterprise.

BACKGROUND

Appgate SDP is a Software-Defined Perimeter – a network security model that dynamically creates one-to-one network connections between user or devices and the resources they need to access. It delivers a scalable, distributed, high-availability architecture for access to micro-segmented network resources underpinned with technology optimized for hybrid and cloud environments.

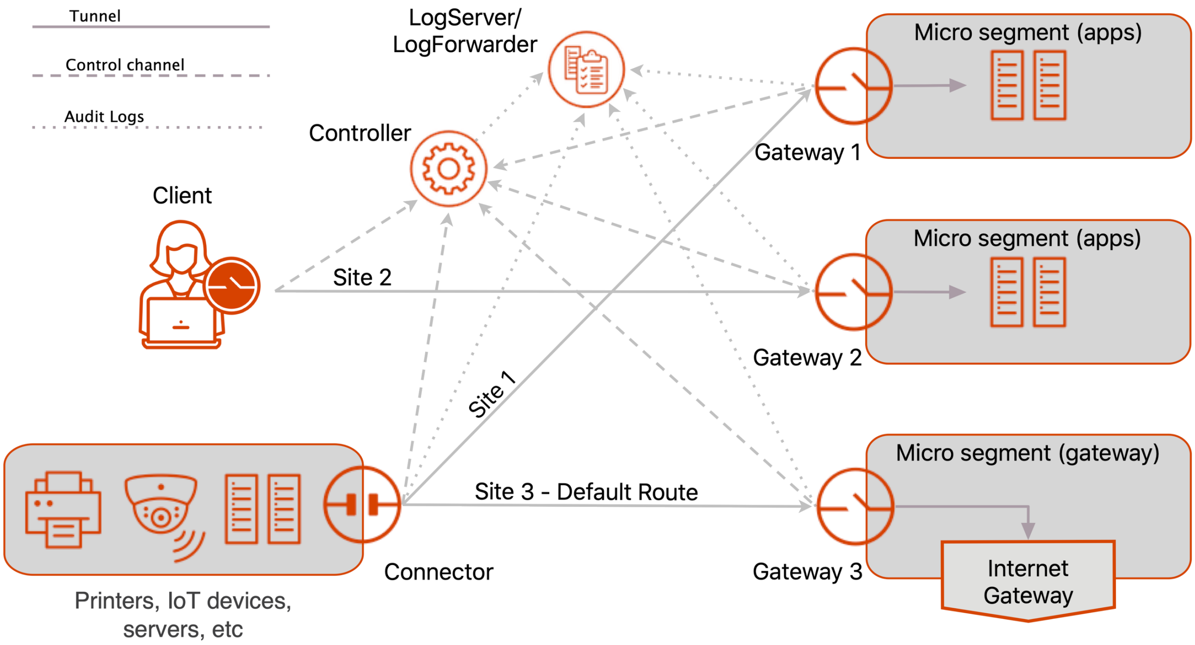
Appgate SDP comprises a number of key elements: Controller (policy decision point), Gateway (policy enforcement point), Client and Connector. It makes use of internal PKI based algorithms to create mutual trust relationships between these different elements.

Client devices authenticate to the Controller, which evaluates credentials, and applies access policies (based on the person, environment and infrastructure). The Controller returns a cryptographically signed token back to the Client, which contains the authorized set of network resources. The network driver on the Client establishes secure tunnels and forwards the token to the appropriate Gateway(s). It also establishes the required routing rules to direct the traffic through the appropriate tunnels.

When the user attempts to access a resource – for example by opening a web page on a protected server, The Gateway applies additional conditions in real time which check for example, the user's network location, the device's attributes, or the time of day.

Connectors are effectively multi-Client Appliances designed to handle traffic to/from local resources and forward them to Gateways in the same way as user originated traffic. In a sense they could be thought of as an extension to a Site.

Gateways may permit access, deny access, create an alert or require an additional action from the user, such as prompting for a one-time password. Once granted, access to the resource is via the secure tunnel from the Client, through the Gateway an onto the server. Access is logged by the LogServer, ensuring that there's a permanent, auditable record of user access. Appgate SDP can also feed the LogForwarder which sends logs into a SIEM or IDS for analysis and response. Appgate SDP supports all major desktop, server and mobile operating systems, and all major cloud and virtualization platforms.



This paper covers the performance of all the main elements in the Appgate system. For Controller and Gateway, we include some simple rules of thumb to allow sizing of these two components to meet any given performance criteria. Client performance does matter as well, but detailed analysis is not easy as the individual Client tunnel performance figures depend on the user's hardware! We have included figures for when it is running on what is effectively a device with unlimited resources. The figures for the Linux Client are relevant when going on to measure the performance of Connector which is included at the end of this paper.

PERFORMANCE AND SCALE

Historically, VPN systems have been found wanting either from a performance point of view (throughput) or from a scalability point of view (concurrency).

- Performance was often limited by the ability of the processors to perform the required cryptographic operations; many products featured hardware acceleration to try to mitigate this.
- Scale was restricted because most designs were stateful – so required state synchronization when scaling up; a process which cannot scale beyond a handful of nodes.

These are the two main reasons why most organizations have not widely deployed SSL / TLS across their networks despite the significant security advantages this approach offers.

Appgate SDP – A Different Approach

Appgate SDP was developed using a different approach. The distributed architecture has allowed different functional blocks to be implemented as stand-alone appliances and optimized for their specific needs:

- The token passing model allows the tokens to hold all the session state information meaning the appliances are stateless, eliminating the requirement for these functional blocks to communicate in real-time. Unlike the traditional single entry active-passive approach, this enables linear scaling for each protected Site.
- Critically, when clustering (to achieve scale), the use of tokens eliminates the bulk synchronization of session states between Gateways.
- Each session has its own micro private firewall, limiting the size of any given firewall ruleset the system needs to navigate when forwarding a network packet, this results in a very small firewall state table associated with each session.

The tokens themselves are secured because they reside on the Client which is considered an insecure location. Another requirement of a distributed architecture is the need to use TLS for all connections within the system. Even though TLS is commonplace these days the cryptographic processes involved in making TLS connections and securing tokens remains very CPU intensive, requiring both cryptographic signing/verification and encryption/decryption operations.

Intel have been progressively introducing improvements including AES New Instructions (Intel® AES-NI) and the PCLMULQDQ instructions which are designed specifically to increase the performance of AES-GCM operations. ECDSA, RSA and DSA signing and verifying operations have been speeded up by introducing new ADOX and ADCX instructions. It is only with some of these recent advances in CPU capabilities that it now possible to deliver this type of heavy cryptographic workload without compromising performance and scalability. Appgate SDP's design is optimized to take full advantage of these new multi-core processors and their associated capabilities.

Unfortunately, there have also been some performance sapping vulnerabilities unearthed since 2018 relating to hyperthreading. At the time these required some dramatic mitigations that affected the performance of Appgate SDP Gateways quite adversely with a reduction of about 20% for single user maximum throughput. The nature of these vulnerabilities means there is no way to exploit these on a dedicated system such as an Appgate SDP physical appliance, so these mitigations are not a must-have. Clearly for Cloud or for shared virtual hosts then they must be in place.

As time has passed newer processors and upgraded operating systems have gone some way to compensate for these mitigations and today with the Controller there is no measurable performance difference with or without these mitigations. The same cannot be said for the Gateways, where there remains a big benefit in terms of reduced vCPU requirements for a given throughput.

Appgate SDP appliances come in Cloud, virtual and physical form. For some applications this choice does not matter very much from a technical perspective. However, when it comes to the networking environment there can be some very significant differences. Gateways may need to be treated as a special case and if network performance (throughput, DoS resilience, latency, etc) matters then simply deploying a Gateway appliance with some default settings to a virtual host or in the Cloud will not necessarily deliver the best outcome.

The physical appliances that Appgate supplies mainly use the latest Xeon chipsets from Intel. All the performance and scale figures included in this paper all assume AES-NI support is in use. It is not recommended to run Appgate SDP on systems that do not support AES-NI. If you are running in the Cloud, then please ensure you choose instance types that support AES-NI.

BENCHMARKING

In addition to virtual and Cloud instances of Appgate SDP, Appgate supplies pre-installed physical appliances when required. The current models are:

- The Ax-H3 (optimized for Controller and Gateway)
  - 1 x Intel® Xeon® Gold 6208U 2.9G, 16core/32thread
  - 2,933 MHz bus speed
  - 64GB RAM
  - Dual PSU
  - Dual cached RAID SSD disk
- The Ax-G3 (optimized for Gateway)
  - One Intel® Xeon® E-2244G processor, 3.8 GHz, 4core/8thread
  - 2666 bus speed
  - 16GB RAM
  - Single PSU
  - Single SSD disk
- The Ax-M (optimized for Connector)
  - One Intel® Pentium® N4200 processor, 1.1 GHz, 4core
  - 2400 bus speed
  - 4GB RAM
  - Single PSU
  - Single SSD disk

The theoretical level of performance that can be achieved by using a distributed, stateless, token-based architecture optimized around the new cryptographic operations available in these new CPUs exceed what any normal enterprise would require as we will now explain.

This new approach allows the performance and scalability to be considered separately for each of the elements; Controller, Gateway, Client and the Connector (which is effectively a collection of Clients running on an appliance).

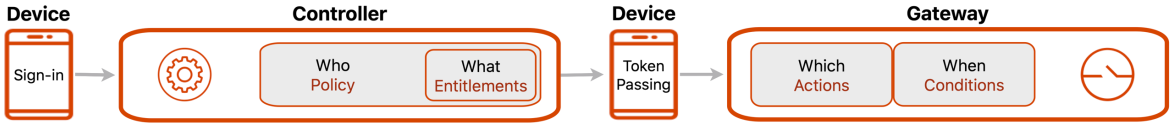
The Controller is the first port of call for setting up a new session, therefore in this paper we examine it first. The workload is then handed over to the Gateways to handle the TLS tunnels and application traffic, which is covered in the second part of this paper.

The third part covers the Client. In normal usage the performance of the Client will be governed by the hardware on which it is running. And given that the performance of an individual Appgate SDP Client can exceed the 1Gb/s network limit of most users' PCs - then the absolute performance will be constrained by the local Client network (or reaching the network throughput limit of an individual Gateway).

The final part explores the Connector which in one sense can be considered as a special use case of the Client. But Connectors can serve as a Site extension, effectively extending the reach of Gateways.

CONTROLLER

The Controller is a stateless appliance that mainly performs on-demand i.e. Client-requested transactions using REST calls. These on-demand requests usually relate to setting up a new session or refreshing an existing one. As such, short periods of unavailability of a Controller may go unnoticed by users unless their tokens happen to expire during that time.



To understand how performant the Controller is in real life situations, it is important to understand what is meant by session. This is NOT the same as a user signing-in to a typical VPN. The Controller provides different services depending on the user's situation:

- For a new session – VPN token, Claims token, Entitlement token and IP address
- For a signed-out user - Claims token, Entitlement token and IP address
- For an existing user – Claims token, Entitlement token

Remember that Appgate SDP tries to stay connected (even after a device has slept), so the most common scenario is the last one.

There are several parameters which should be considered when specifying a Controller:

1. I/O – unlikely to be an issue as traffic volumes are very small
2. Disk – SSD required for database and to ensure logs can be written fast enough
3. RAM – requires only a fixed amount of ram
4. Clock speed - will affect i/o performance before packets even reach the Controller software
5. CPU - choosing the optimal number of CPU for the required performance
6. Policy – the more Policies configured the longer the sign-in times
7. Multi-Controller group – use of more than one Controller to improve performance
8. Token renewals – frequent renewals will add to the Controller loading

Test Setup vs. Real Life

Testing has been performed on hardware, virtual and Cloud based Controllers.

For hardware testing, the majority of the testing was performed on members the Ax-H family of physical appliances which Appgate supplies. They were configured as the Controller and LogServer, although the LogServer was not accessed during the test.

For both hardware and virtual platform testing the performance metrics were measured in an isolated environment using a 40Gb network. The tester clients run on very performant servers which also have 40Gb network interfaces. This therefore represents the very best that can be achieved. In real life there are many other elements that will have an effect on the overall sign-in times (measured at the Appgate Client):

- the performance of the Client device used
- the client device's network connection (to the Controller)
- The use of an external LDAP will also have a big influence as an external (to the Appgate SDP) system query has to be performed.

- the use of Policy assignment scripts which might make external system queries.
- and once the Client has signed into the Controller there is still the final step of signing into the Gateway(s) to add.

The cloud testing performed might be considered more representative as the test Clients were run on more normal hardware, connected to a high bandwidth internet link in Sweden and connecting to a Cloud based Controller in the UK.

The test system used can run between 5 and 100 test clients in parallel. These perform signs-ins one after another as fast as possible. At the top end of this range, the available test sign-in rates exceed what any normal enterprise would ever require. The full sign-in process is performed and completes when the VPN tunnel is up (using a separate Gateway appliance). The test completes when the specified number of users are all signed in.

For the majority of tests, a simple configuration of one Policy with a few Entitlements was used and the users were all in the local database (used for Client authentication). Later we examine the effect of more complex Policy configurations.

Controller Performance Metrics

There is a difference between the way physical and virtual appliance behave when it comes to Controllers. Things such as disk and RAM behave much the same whether in the Cloud, on a virtual host or on physical hardware. However, this is not the case for CPUs where more are required to achieve the same sign-in rate when using virtual hosts - but the optimal (maximum) number should not be exceeded!

I/O

The Controller only sends and receives relatively small tokens. It also receives REST API calls and has to send logs to the LogServer or LogForwarder. None of this amounts to very much in terms of traffic volume so a 1Gbe NIC will normally suffice.

In real life situations the Controller is likely to make external calls to an LDAP server and maybe elsewhere if there is an external call such as from a *user claim script* or from a Policy *assignment criteria script*. All these external calls will add to the sign-in times presented in this paper. Network performance (latency) is yet another which can add to the sign-in times. Imagine 2 international round trips from the Controller talking to slow servers and 1 or more seconds can easily be added to the sign-in times.

Disk

A standalone Controller requires only 5GB of disk space but to allow for upgrades and storage of local logs (max 4GB), we suggest 20GB minimum.

During sign in the Controller makes frequent access to the disk where the system's PostgreSQL database is stored. Combine this with the ability of the Controller to utilize as many vCPU as are available, and the disk I/O becomes very critical. Analysis has shown that a fast disk is required (must be SSD); a good RAID controller with caching enabled should be used and the fastest system bus speed should be specified.

Audit Logs are handled within each appliance using logd. Logd has the option to write audit logs in one of 3 ways; Guaranteed, Default or Performance. For details of Audit Log Persistence mode please refer to the Audit logs section in the manual. Guaranteed mode waits for the write to disk – so disk speed might have a very significant performance impact.

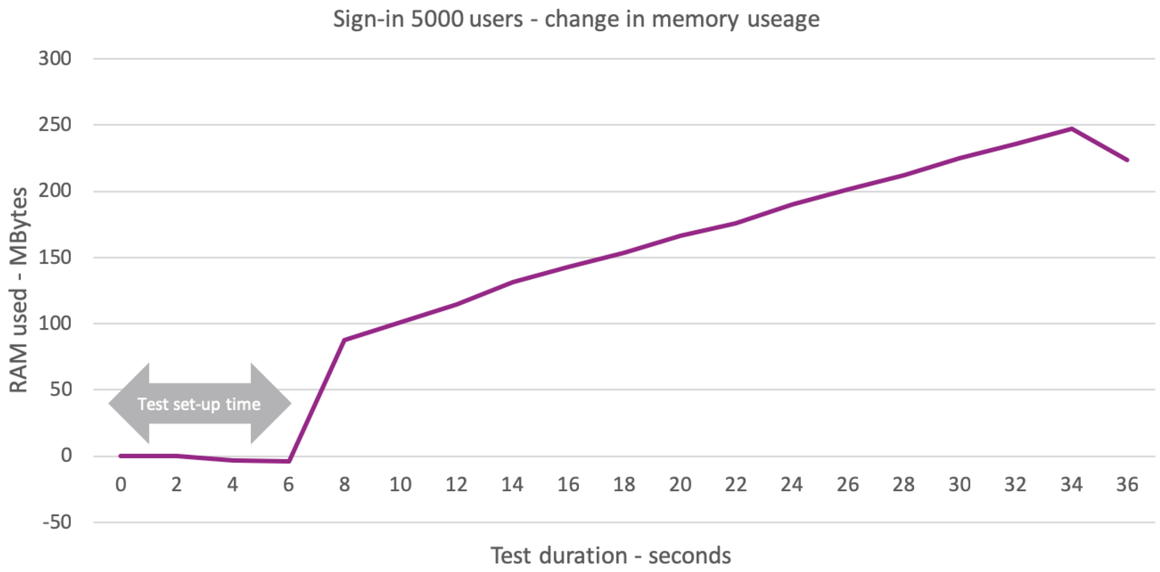
For most enterprises, the audit logs are usually moved to either an Appgate SDP LogServer or some other external SIEM so disk size is not so important.

The Ax-H$_3$ appliances are specified with faster SSD disks paired with caching RAID controllers.

RAM

Testing for RAM usage during sign-in.

Test configuration		
5000 users	20 test clients	Ax-H with 20vCPU



From the graph it can be seen that any additional RAM utilization even at a high sign-in rate (200users/sec) is negligible. A production server with 4GByte of RAM (2GB base requirement + 2GB for Controller) will be sufficient in most cases.

Clock Speed

The Controller is a CPU intensive function, so the general rule of thumb is the faster the clock speed the better. Fewer faster CPU will reduce the amount of thread switching which will aid overall performance. But exact CPU requirements are harder to make generalisations about because of the database access speed constraints that come into play eventually. There are two metrics that need to be established to quantify Controller performance.

1. Average sign-in time (for a user)
2. Sign-in rate (users per minute)

These metrics (and the CPU requirements) will vary depending on whether hardware, virtual and Cloud based Controllers are being used. The best-case scenario is when running on hardware and worst-case scenario is when the Controller is Cloud based. We will go through all three in order.

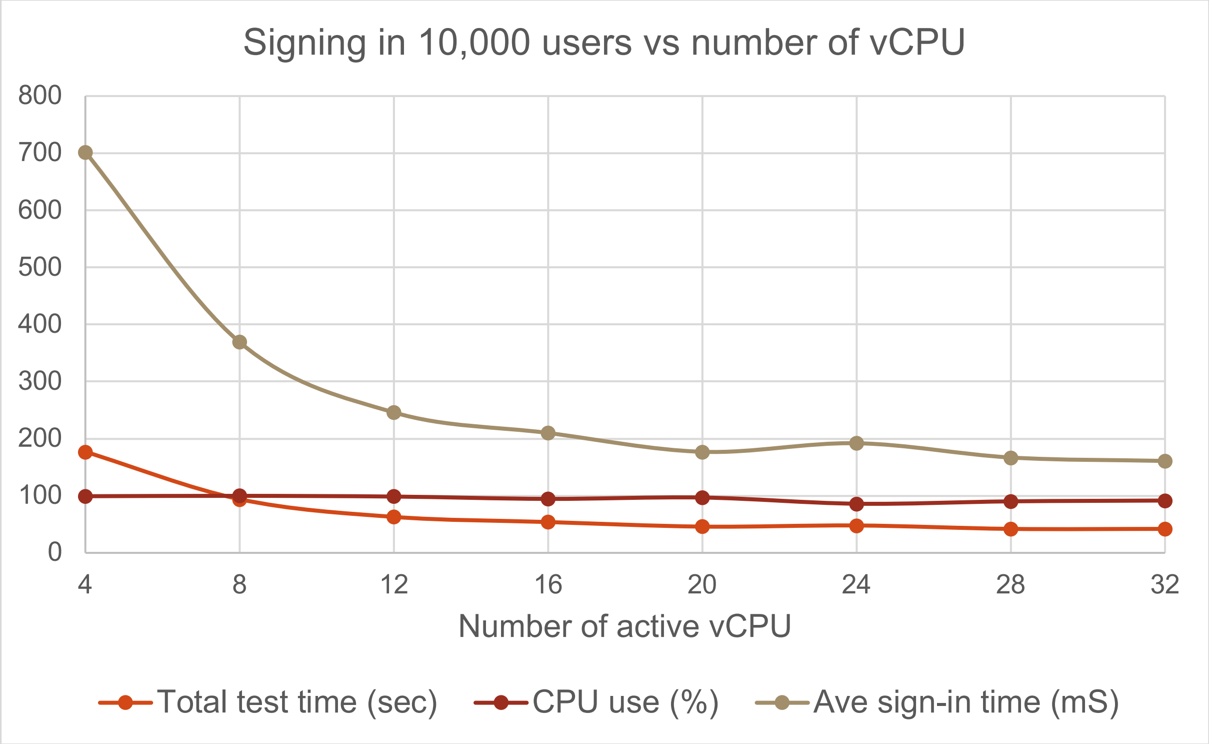
Physical & Virtual Appliance Performance

Testing Ax-H$_3$ for maximum sign-in performance rate.

Test configuration		
10000 users	40 test clients	Ax-H$_3$ with 4-32vCPU

The Ax-H$_3$ is an appliance that has been optimized for use as a Controller.

Using only 12 of its 32vCPU the test was completed in about 60 seconds – which works out at a rate of almost **14users/vCPU/second** (which we like to use as our preferred metric for defining Controller performance).

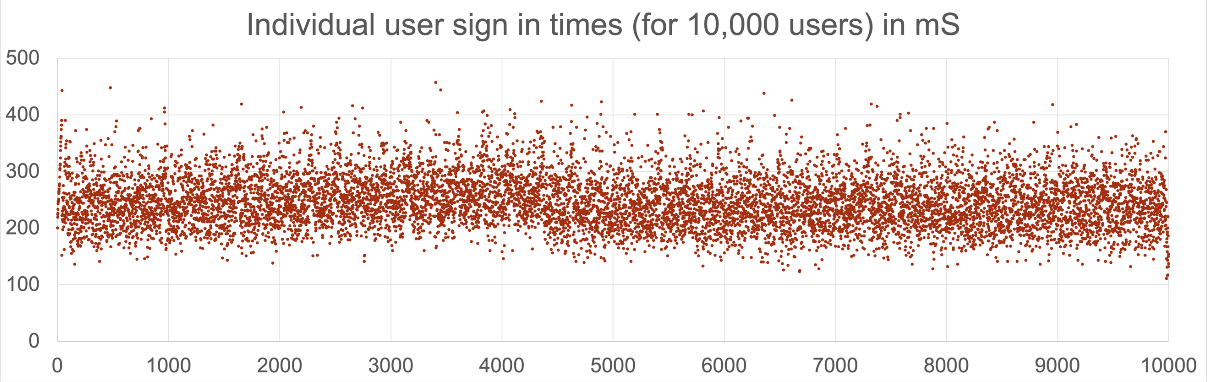


The CPU always remains close to the 100% mark irrespective of the number of vCPU being used. Initially the vCPUs are busy doing what they should – signing in users. With more than 16vCPU the internal bus becomes so congested that the OS spends most of its CPU cycles swapping processes that are blocked in order to try to optimise individual user sign in times.

Below 12vCPU, the number of vCPU will affect the perceived performance of the Controllers. Beyond 16vCPU the performance has flat-lined so there is no real point in specifying a Controller with more than 16 vCPU unless the congested bus is remedied first.

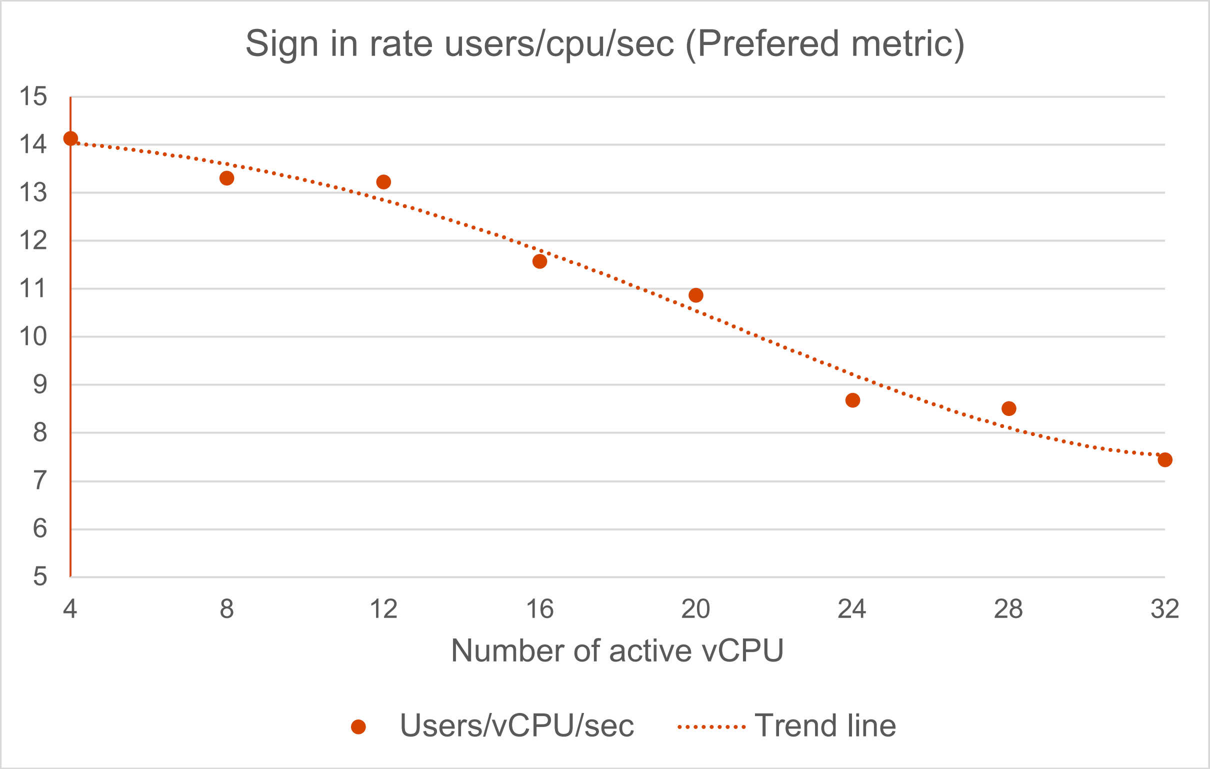
One key aspect of interest for the Controller is the sign-in time for an average user. This follows a similar (but more pronounced) profile as the overall sign-in time for all 10,000 users. At 12vCPU the average user sign-in time is down to 250mS which is very acceptable. By using more vCPU it is possible (but costly) to get down to about200mS but this is not recommended.

The graph above shows average sign in times, so another key aspect is the variance from that average those users will actually experience. The graph below shows the actual sign in time for each of the 10,000 users. This is taken from the 12vCPU test run as it represents a system that is neither starved not over provisioned in terms of vCPU.



The result is quite consistent, mostly staying in the range 150 to 350mS.

Users/vCPU/second is a very useful metric for estimating the size of a Controller. However from the analysis above you can see that this only hold true up to about 16vCPU.



As shown above, below 16vCPU a Controller such as an Ax-H_3 can pretty much guarantee a rate in excess of **10users/vCPU/second**. This makes a useful way to estimate requirements quickly: So, 6000 users in a minute would require 10vCPU.

Above 16vCPU a Controller's bus becomes congested and **users/vCPU/second** takes a pronounced downward shift. By the time you get to 32vCPU the rate has dropped to almost exactly half of the 4vCPU rate!

These results show that the number of vCPU is important. However, you should not simply throw more vCPU at the problem as this is very cost inefficient. Two 10vCPU Controllers will sign-in about 270 users/second; whereas a single 20vCPU Controller will sign-in about 220 users/second.

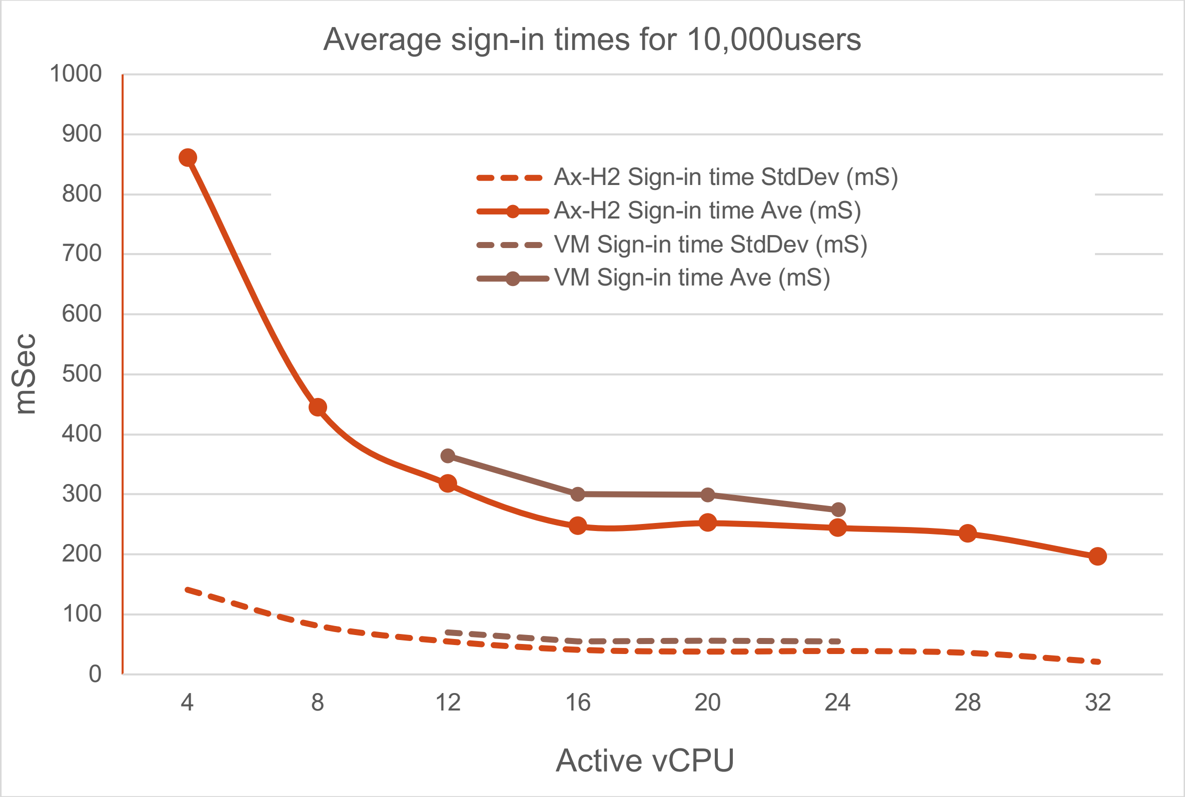
Testing relative Controller sign-in performance rate when deployed on a VM.

The AxH_2 was now reconfigured to run KVM. This will therefore give an exact like for like comparison between running on Hardware and on a VM. The range 12-24vCPU was chosen as this was where we were seeing the Controller achieving it maximum performance on hardware.

Test configuration		
10000 users	40 test clients	AxH_2 KVM with 16vCPU

Firstly, a straight comparison was done to the hardware sign in performance.

The measured average sign-in time was **300mS;** this is 17% slower than on hardware when averaged over the 4 test points; the biggest difference being 20% at 16vCPU.



The overall time to sign-in time the 10,000 users takes similarly 17% longer; the biggest difference being 22% at 16vCPU.

The VM achieves a sign-in rate of **9users/vCPU/second** when using 12vcpu. However, the efficiency of using more vCPUs deteriorates after this point; at 16vCPU the rate has fallen to **8users/vCPU/second**.

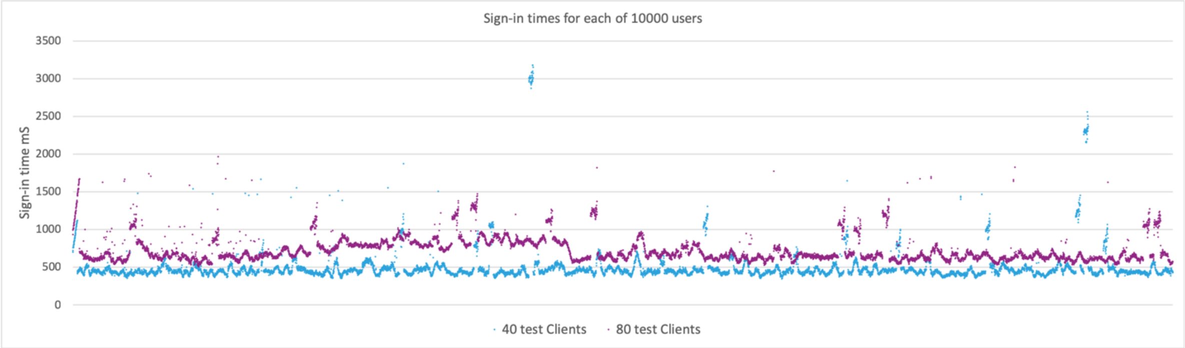
If you were specifying Controllers; then in the example where you have 3000 stores, each having 10 devices that need to sign-in in the 10 minute period prior to opening.

The sign-in rate must therefore be 3000/minute or 50/second. This can easily be achieved using a single v5.2 hardware Controller. If you were specifying a VM then you need say 8vCPU. This would be able to at least 80uses/sec. Having a few vCPU extra is a good idea as the Controller has other tasks to perform, VM may have some other limitations and sign-ins will not be spread uniformly.

Testing for relative Controller sign-in performance rate when deployed on a typical Cloud Controller.

As we have shown a Controller can utilize many vCPUs (up to about 16), so, an Azure instance with 16vCPU was selected – a Standard_F16s. With the Controller in a different region from the user, this would represent close to the best you can expect from a Cloud based Controller because there will be considerable latency which is likely to spread any short-term loading reducing the need for many vCPUs.

Test configuration		
10000 users	40 & 80 test clients	Azure F16s with 16vCPU



The graph gives some idea of the spread with the worst sign-in times around 3 seconds. Because the Controller in a different country, the vagaries of the internet come into play in these results.

The average sign-in times for the 40 tester Client (blue) was similar to a virtualised Controller; extending by almost 50% when 80 tester Clients (mauve).

The sign in rates were similar for 40 and 80 tester clients. But were noticeably slower than the virtualized Controller mainly due to latency issues. **At 16vCPU the rate achieved was about 60%of what was achieved on a VM (in test lab conditions).**

To meet the same sign-in rate as previously (50 users/sec) with one Controller we need a minimum of about 12vCPU.12vCPU in a VM would provide 12*9 users/sec. If we factor this back by 60% then we can still achieve 60 user/sec.

Policy count vs sign-in rate - Physical vs Virtual Appliances

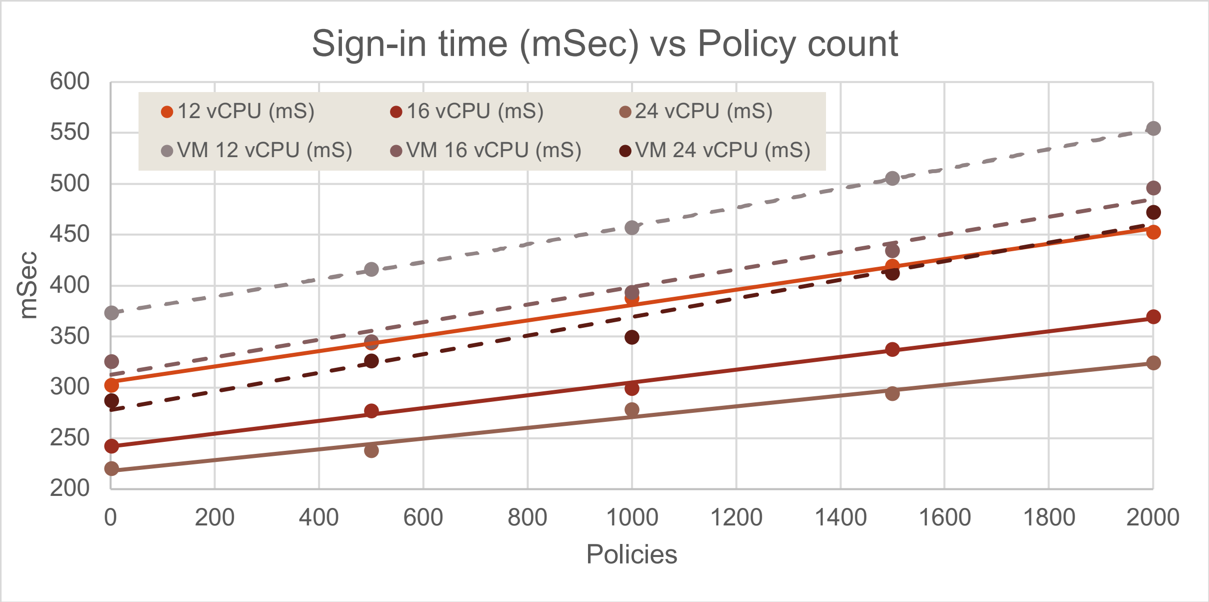
There are three main tasks performed by the Controller which have a significant effect on sign-in times.

1. The user has to be verified by the Identity Provider. This usually means making a call to an external system so falls outside the scope of this analysis.
2. The required cryptographic operations. This is a relatively predictable process which will remain constant irrespective of the configuration of the Appgate SDP system.
3. Data base queries to establish the rights of the specific user. This process will vary according to the configuration of the Appgate SDP system and relates to the number of Policies, as the Controller has to iterate over each one to see if it applies!

We have already established that the Controller’s sign-in performance varies with the number of vCPU, so for this test we only used reasonably generous numbers of vCPU since that would be the best way to configure systems with very many Policies. 12,16, 20 and 24 vCPU were tested. The results for 12, 16 and 24 are shown below. As expected from the earlier findings; the results for 24 were slightly better than 16 but the improvement is marginal compared to the change between 12 and 16.

Testing the sensitivity of the sign-in times to varying Policy counts from 1 to 5000.

Test configuration		
10000 users	40 test clients	Ax-H_2 & Ax-H running KVM



From the graph above you can clearly see that sign-in times increase linearly with increasing Policy counts. Tests up to 10k Policies were performed to confirm that the linear relationship holds true beyond 2000.

The starting points and the slope are worse for the VM than for the Ax-H_2. Both cases can be easily be calculated by just adding a Policy count factor to the nominal time achieved with 1 Policy configured.

- For Ax-H_2 the additional factor is approximately $\frac{PolicyCount}{16}$ mS. So, for a system with 1600 Policies configured then you will be adding about 100mS to your nominal sign-in times.
- For a VM the additional factor is approximately $\frac{PolicyCount}{12}$ mS. So, for a system with 1600 Policies configured then you will be adding about 135 mS to your nominal sign-in times.

The reason for the VM performance is because of the nature of the operations being performed. There is very little processing in the VM itself, most of the time is taken accessing the database which requires a call via the VM to the disk.

In the nominal case with only one Policy there is no assignment decision to be taken in the Controller. However, if the resulting Entitlements contain very many conditional Actions, then these would have to be parsed by the Gateways – thus extending the overall sign-in times. It is therefore very important to consider at the design stage the balance between the Controller - making Policy assignment decisions and the Gateway - dealing with the resultant Conditional Actions.

In real life a mix of Policies (evaluated on Controller) and Actions/Conditions (evaluated on Gateway) should be used. The number of Policies being limited, but their payload increased with more Entitlements included in each one. Remember, one Controller may support many more than one Gateway; so Gateways are much more able to absorb the load imposed by having to make many conditional decisions. The Gateway load is also spread over time (when a user actually accesses a specific resource) whereas a Controller's workload is more likely to be concentrated at specific sign-in times such as a shift change.

Token Renewals – the Effect on CPU Loading

It should also be remembered that token renewal will add load on the Controller. This frequency can be set in the admin UI and the default setting is unlikely to present a significant extra load. However, if it was to be set to a timeframe shorter than the expected login window, then the sizing of the Controller should take this into account. Renewals have an additional overhead required to validate the old tokens before issuing new ones. This is typically about a 20% increase over a new sign-in.

By example: If we take our 3000 store organisation and it buys another organisation of 3000 stores in a time zone 1 hour different – then without token renewals the same specification would work fine. If however, the token life was set (badly) to 60 minutes, then the renewals would happen at exactly the same time as the 2nd time zone signs-in. This renewal load being added to the sign-in load will impose an additional 120% loading on the Controllers. So, we would need to add say anther 8vCPUs taking the total number to 16. Whilst this scenario is not entirely realistic, token renewal timings should be thoughtfully specified and capacity budgeted when specifying your own virtual machines.

Use of Multi-Controller Groups

It is possible to use more than one Controller; grouped together in what is called a multi-Controller group. There is a theoretical limit of 48 Controllers, however a realistic limit will probably be in the 2-3 Controllers range (the current limit from Appgate SDP version 5.x is set to 6). Because the group uses multi-master each Controller uses its own database. This means that from a scaling point of view each Controller works independently and will simply perform at the levels already discussed above.

A second Controller can be added to create the group, and after a short period of database replication will become active in multi-master mode. Remember, Appgate SDP stores no status information in the database; almost everything in the system is handled in real-time or by tokens. The only information in the database is the Policy configuration data and relatively static information such as user's tunnel IP address. The token-issuing activities performed at sign-in do not therefore create any significant database replication requirement.

The most likely benefit of using a multi-Controller group is in providing an HA sign-in service (better uptime) or to provide a closer geographic location for users (lower latency) when geo-based DNS load balancing is used.

By example: To support an organization of 60,000 users, all of whom sign-in over a +/-5 minute window, would imply a rate of 6000 users/minute or 100 users/ second. If we assume a total of 200 Policies, then we can assume we will only achieve 90% of the nominal sign-in rate.

We are using virtual Controllers with 12vCPU on a managed network which will be doing about 8 users/vCPU/second. So, a single Controller is sufficient but we want two Controllers so it would be possible to reduce each to say 8vCPU. Controllers do perform other services such as handling the admin UI and REST calls so some additional vCPU capacity should always be allowed for. It might therefore be better to stay with two 12vCPU Controllers to give full failover capability in this case.

Controller Summary

The Controller can deliver sign-in performance appropriate for the most demanding of enterprises. However, it has to be specified and configured appropriately for the expected workload.

If Controller sign-in times of under 500mS are required then the critical factor is to reduce added latency, use sufficient vCPU and have a modest number of Policies configured.

The required sign-in rates should be translated into the preferred metric of users/vCPU/sec. A maximum of 16vCPU should be specified per Controller and then estimate the number of Controllers. Where very high sign-in rates and/or Policy counts are required then a physical appliance should be used.

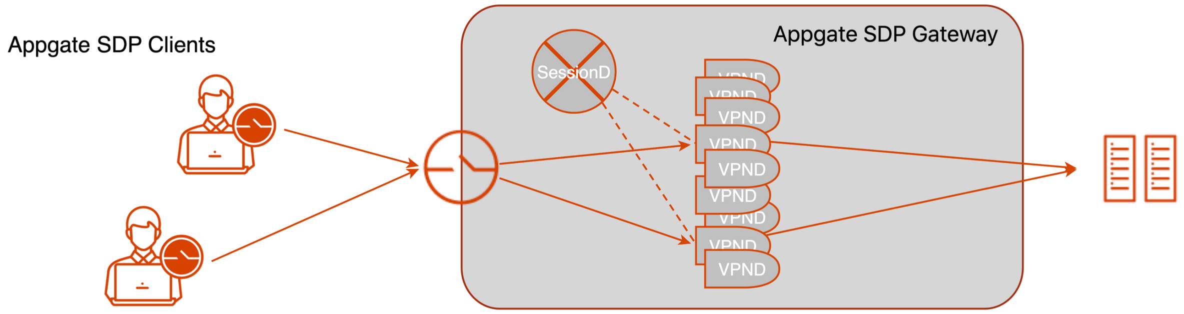
Rule of thumb: For sub 500mS sign-in times specify CPU based on 5-10 users/vCPU/second.

See Instance Sizing in the manual for the latest recommendations.

GATEWAYS

The Gateway is a stateless appliance that handles tokens and performs the (D)TLS tunnel termination and firewalling tasks on the application traffic from users and devices.

To understand Gateway performance/scale it is important to understand a little about how it works.



SessionD is the brain of the Gateway and processes all the tokens to decide what traffic will be allowed. The initial part of the process can be considered as signing-in and whilst this is not as heavy as the initial Controller sign-in, it is still something that needs to be considered. The more Actions a user has the longer it will take to perform this part of the process. (It is not recommended to have more than 8000 Actions per user session.)

Thereafter sessiond either sends the appropriate rulesets to the user's vpnd instances or to the NGiNX instances that handle URL access (the HTTP up Action types).

Consideration should also be given to the system load imposed by running any Entitlement scripts, access criteria scripts and use of name resolvers. Any changes to device script claims which are checked every 5 minutes (as well as other time-based conditions) can trigger re-evaluations which can consume considerable resources in the Gateway.

External calls made by scripts/resolvers may also take additional time making the system less responsive.

- Sessiond's CPU usage will be somewhat proportional to the number of sign-ins / re-evaluations it performs. A Gateway configured to re-evaluate all users every 5 minutes will be 12x busier than one set to perform re-evaluations every hour! RAM usage does increase with users but because of the way Java allocates and re-uses memory it does so in a slightly irregular manner.
- NGiNX's RAM usage can be considered a (small) constant.
- Each vpnd instance handles the connection/firewall thread for a session. Each one is completely independent, taking full advantage of all the CPUs available and sharing resources such as I/O. vpnd is not Java based, so the resources it consumes reflect the number of concurrent sessions.

There are several parameters which should be considered when specifying a Gateway:

1. Disk - ensuring logs can be written fast enough
2. RAM - having sufficient RAM for the users
3. Clock speed - affects I/O performance before packets even reach the Gateway software
4. CPU - choosing the optimal number of CPU for the required performance
5. I/O – the limiting factor when it comes to the data throughput limit
6. HA Gateways (virtualized) – having enough (but not too many) on one virtual host

Gateway Performance Metrics

There is a noticeable difference between the way physical and virtual appliances scale. Things such as disk and RAM behave much the same, but this is not the case for CPUs, where on virtual hosts there is an optimal number which should not be exceeded.

Disk

A stand-alone Gateway theoretically requires only 4GB of disk space but to allow for upgrades and the storage of local logs (max 4GB), we suggest 20GB minimum. Logs are ‘rotated’ every 30 minutes so disk usage will generally follow a sawtooth pattern. For most enterprises, the audit logs are usually moved to either the Appgate SDP LogServer or some other external SIEM via the LogForwarder.

A busy Gateway creates log records at a considerable rate which will (normally) be written to disk. Because of this disk write speed can be very critical and should be minimised on busy Gateways. Appgate SDP systems generate about 1 log record per user per second. Most of these will originate from the Gateway but the per appliance rate will be well below the 1 log/sec as the generation will be spread across multiple Gateways.

Audit Logs are handled within each appliance using logd which has three Audit Log Persistence modes: Default, Performance or Guaranteed. The latter can have a very significant performance impact. For details of these 3 modes please refer to the Audit logs section in the manual. Appgate’s physical appliances are specified with faster SSD disks to ensure audit log writing does not impact the overall system performance. The same model should be adopted for virtualized appliances.

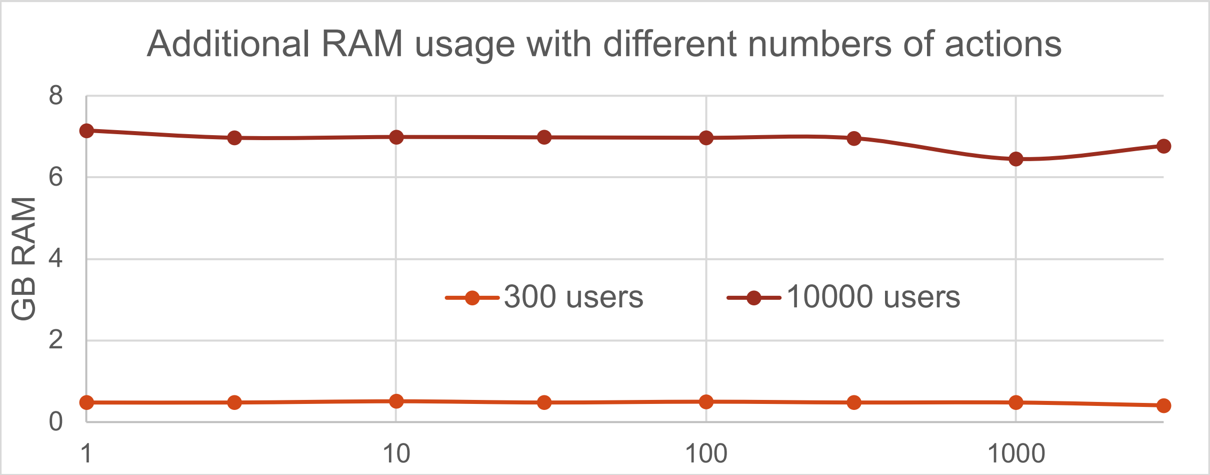
RAM

Let’s cover RAM usage first as this is the same irrespective of where the appliance is run.

Sessiond and each VPNd instance are the main users of memory. Since the rules and separate VPNd instances are created per user, the overall memory usage is proportional to the number of users on a Gateway. Always allow for enough memory for the maximum number of users planned. There is also a base memory requirement for the OS, Java and the other daemons. This base memory is somewhat determined by the amount available in the machine and can be less than 1GB. A VM with 4GB RAM might have 1GB is assigned.

Testing 300 and 10000 concurrent (non URL access) sessions vs ruleset complexity.

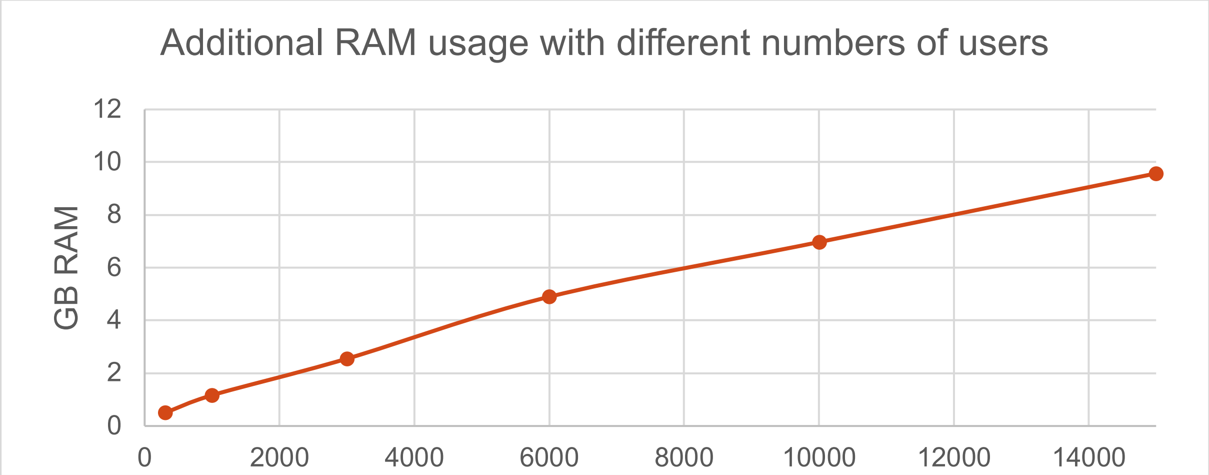
Test configuration		
300-15000 users	1-3000 Actions	Ax-G



The first thing to establish was the sensitivity of memory usage vs. the complexity of the rulesets. Usage was checked across a range of user counts and numbers of Actions; two of the user counts are shown in the graph above. The conclusion is that memory usage is unaffected by the number of Actions assigned to the user. This was tested beyond what is shows (to 6000 Actions) and the behaviour remains linear.

The graph above clearly shows that higher numbers of users require more RAM, so this was tested next.

RAM is the only major factor that needs to be pre-matched to the number of sessions expected. The graph below validates the linear scalability model that has resulted from the design approach adopted for Gateways. The testing extended to 15000 users and the linear relationship makes it easy to estimate RAM requirements.



Since it is better to have too much than too little then you should allow about an additional 1GB RAM per 1000 concurrent users. This would mean that a total of 16GB would be plenty for a Gateway with 10,000 users.

NOTE: The Appgate SDP appliance does NOT use disk caching for RAM. So, when the appliance runs out of RAM it does just that. The OS will shut down daemons it considers to be using too much memory so user connections will be interrupted. It is therefore important to ensure enough RAM is specified when sizing appliances.

I/O

A Gateway's I/O capabilities are covered in some detail later as this is tightly bound to the number of vCPU. Appgate SDP can support 1Gbe, 10Gbe 25Gbe (and 40Gbe) NICs. Different makes and specs of NIC cards will clearly have an effect on the throughput performance in terms of the headline Gb/s figures. But NIC cards also vary in terms of their ability to handle a given number of packets. If you have very small MTUs then it is possible to hit this packet limit long before the stated bits/s limit is reached.

Clock Speed

The Gateway function is very CPU intensive due to the cryptographic tasks and the packet shuffling requirements. Unsurprisingly, analysis has confirmed that the faster the CPU clock speed the better. The way processors handle I/O is not related to the number of CPUs - the I/O system has to feed packets to the CPUs. I/O is in fact dependent on the underlying system clock speed, so if very high packets/second rates are required, then the highest possible clock speeds should be used (>3GHz).

DoS performance is also related to I/O handling (packets/second). Again, analysis has shown that the faster the CPU the better the Gateway will handle DoS attacks. There is a separate paper covering this topic.

Physical Ax Appliance Performance

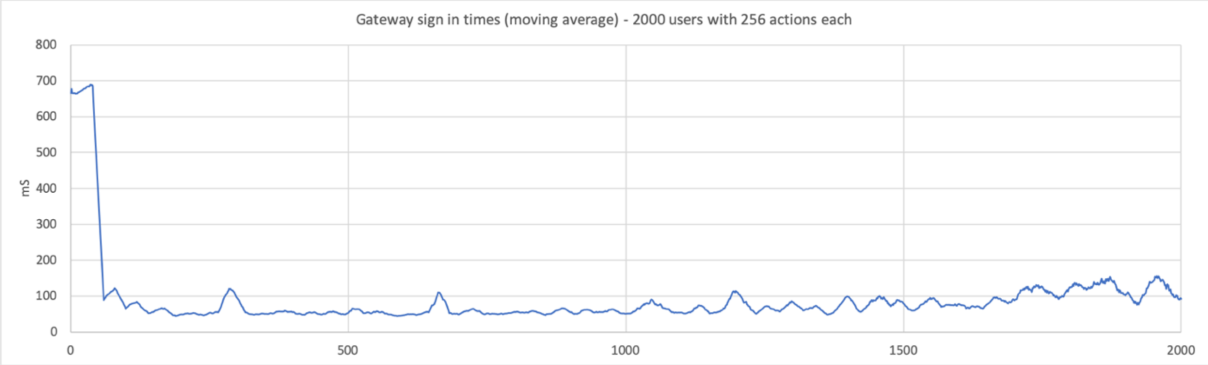
Signing-in

Because Appgate SDP is a token-based system then the Gateways know nothing about the users until a connection request arrives from the Client. At this time the Gateway has to set up the TLS tunnel, verify the token and set up all the related access rules. Think of this a signing in to the Gateway!

Even though the process is less cumbersome than it was with the Controller there are still some complex cryptographic operations to perform (which can be considered a constant). In the case of the Gateway, there is also a variable element, which is the number of Actions and Conditions the user has been assigned.

Testing Gateway sign-in performance to identify the shortest sign-in times.

Test configuration		
200 & 2000 users	64-262k Actions	Ax-H_2 with 48vCPU



Like the Controller, this process is handled in Java which has a warm-up period - seen towards the left side of the graph. After this the sign-in time settles for the rest of the test.

The test took about 14 seconds to sign-in the 2000 users – a rate of about 150 users/second with an average 55% CPU loading. This shows that there were more vCPU available than required. This is similar to the Controller and equates to about 8.5 events/vCPU/sec.

The scheduler will optimise individual sign-in times so the number of vCPU can be reduced. With only 16vCPU active the sign-in times took about 30% longer – 7 events/vCPU/sec.

In most situations sign-in takes under 100mS however with many actions this will take considerably longer. The sign-in time (in mS) can be estimated using:

$$80 + \left(\frac{ActionCount}{60}\right)$$

With about 25,000 actions (which is not recommended) sign-in would take about ½ second.

Controllers are unlikely to sign-in users all at once, however a Gateway might experience a much higher sign-in / token renewal rates in the event of failover or when a widely used Condition changes. This is mitigated to some extent by the design of Appgate SDP - reconnections are made based on user activity, so the load will be spread over time. Even so you should ensure there are an adequate number of vCPU available for this eventuality.

We suggest you allow 1vCPU per 1000users. This would allow each user to perform some kind of sign-in/re-evaluation/token-renewal event once every 2 minutes. But this figure is very dependent on the system configuration. A very static set-up may only require a couple of events per day. A very dynamic one with auto-scaling might require a re-evaluation event every minute!

Passing traffic

Once the sessions ready to accept traffic then the next requirement is to establish the number of vCPU that are actually useful for Gateway to pass traffic as effectively and efficiently as possible.

NOTE: The URL access (HTTP up Action type) is handled by NGiNX so the vCPU usage model explored below does not apply for this use case. (See the specific section later for URL access.)

For throughput testing up to 32 simulated Clients were used. At up to 2Gb/s each means the actual test is limited by the Client test machine's 40Gbe NIC. The number of tester Clients was usually set to a multiple of the vCPU count to ensure an even spread of users across the available vCPUs.

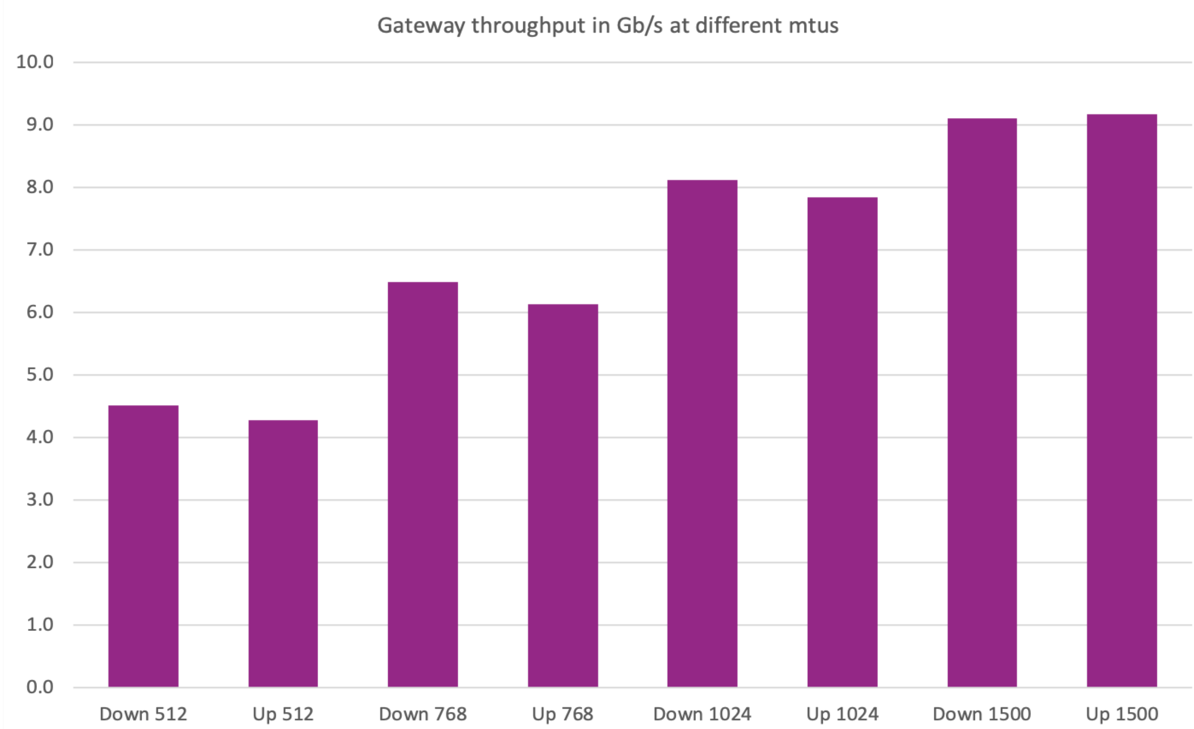
The Appgate Ax-G family of appliances are designed to meet the needs of a Gateway appliance. They come with a single 8vCPU processor clocked at 3.6 or 3.8Ghz (good for large amounts of inbound packets). They have a minimum of 16GB of RAM to allow for many thousands of simultaneous users. They come with a dual 1Gb/s and 10Gb/s NICs.

With a transmission efficiency of approximately 94% (due to IP, TCP &TLS framing overheads, headers, etc.), the best we can expect measured at the Ethernet level is a theoretical throughput 9.4Gb/s. In reality the actual throughput will be slightly less this because the Appgate SDP system itself also uses some bandwidth for sending tokens, state information, health-checks, logs, etc. between the different parts of the system. We should allow say 2% for this; so 9.2Gb/s is the realistic maximum that could be expected for Gateways with a 10Gb/s NIC.

Testing Ax-G Gateway for maximum throughput at various mtu

Test configuration		
32 users (test clients)	512-1500 mtu	Ax-G with 8vCPU

The 32 test Clients transfer packets of several different sizes (mtu). Each run was set to do the uploads/downloads as fast as possible which has the effect of saturating the NIC (similar to DoS) which in itself limits the packet throughput capability of the appliance. From the graph it can be seen overall throughput rates are lower with smaller payloads but with the default mtu of 1500 the Ax-G achieved 9.2Gb/s.



Real life might include a mix of packet sizes all at the same time and these would be rate limited, so another test was performed using 32 test Clients with the following mix:

Number	*Rate Mb/s*	*mtu*	*Equivalent Gb/s*	*Actual down Gb/s*	*Actual up Gb/s*
6	175	512	5.6	5.6	5.5
10	275	1024	8.8	8.8	8.8
16	325	1500	10.4	10.3	10.4
	Weighted Ave Gb/s >>		**9.0**	**9.0**	**9.0**

With this mix, the theoretical maximum possible throughput would be 9.0Gb/s. The figures for 512 and 1024 were better than shown in the graph above because the DoS effect was removed (rate limiting). The slight shortfall in 512mtu upload number made up for by the 1500mtu number. Overall the Ax-G achieved the expected 9.0Gb/s.

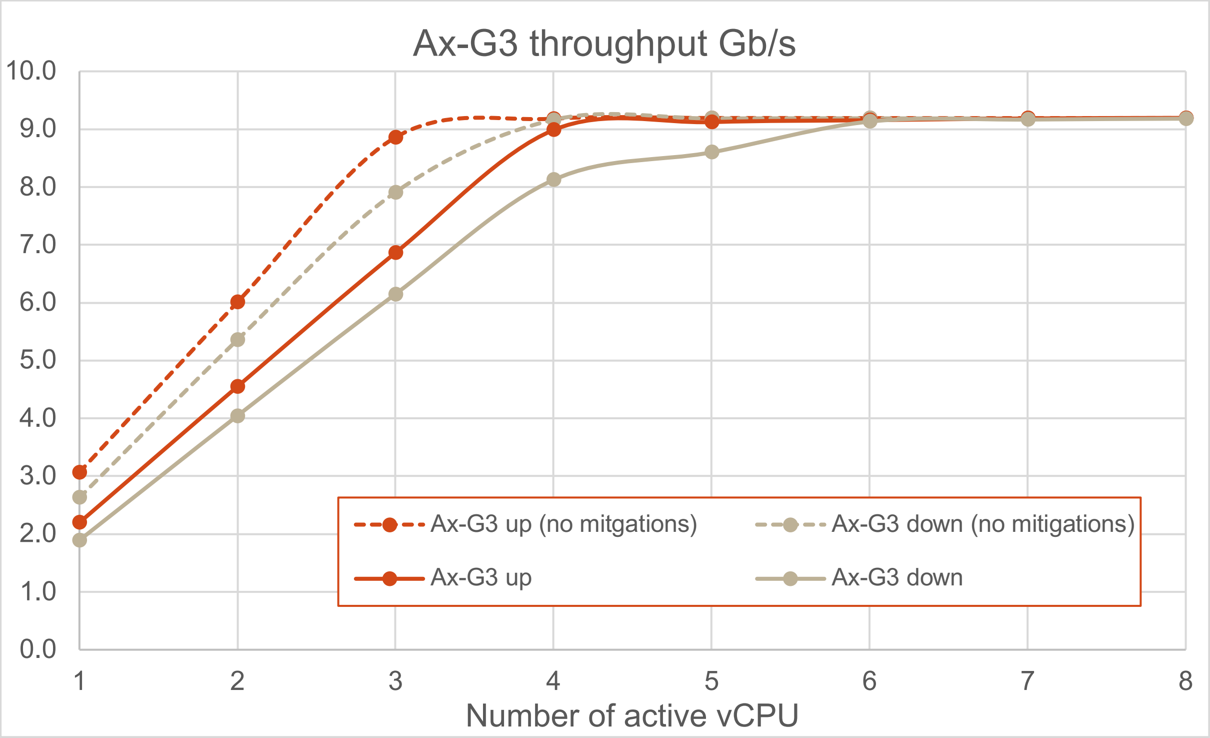
So, for an appliance with a nominal 10Gb/s throughput - the specification of the Ax-G will deliver symmetrical performance at or close to network speeds with its 8vCPU.

Testing Ax-G_3 Gateway – Identifying the optimum number of vCPU

The Ax-G_3 appliance is also the ideal hardware to identify the optimum number of vCPU a Gateway needs (for systems with 10Gb/s NICs) as we know it will have an excess of vCPU.

Test configuration		
24 users (test clients)	default mtu	Ax-G with 1-8vCPU

The 24 test clients share the 10Gb/s – so each should get a theoretical 52MB/s.



As can be seen in the graph, below 4 vCPUs the throughput had a linear relationship to the number of vCPU available. Beyond this point throughput was largely flat. So, specifying enough vCPU is important, however adding more vCPU than you need (for passing traffic) is pointless. This can be seen clearly as the Gateway continued to operate at its maximum throughput of 9.2Gb/s (48MB/s per user) irrespective of the number of additional vCPUs.

In the case of the Ax-G_3 appliance the 'spare' vCPUs are specified to handle the activities the Gateway has to perform such as running scripts, performing condition evaluations and resolving hostnames. Remember to allow for this when specifying a virtual host.

The mitigations relating to hyperthreading vulnerabilities (Spectre, Meltdown, etc) have also been benchmarked. These mitigations can be safely removed as there is no way to exploit these on a dedicated system such as an Appgate SDP physical appliance. As can be seen in the graph, the results for 'no mitigations' saved almost one vCPU in the linear phase (below 3vCPU) while achieving the same throughput. That extends to the maximum throughput which can be sustained using one less vCPU. For larger enterprises with more complex environments, it is strongly recommended to run Gateway appliances with mitigations *off*. Instructions for doing this using cz-config are included in the manual.

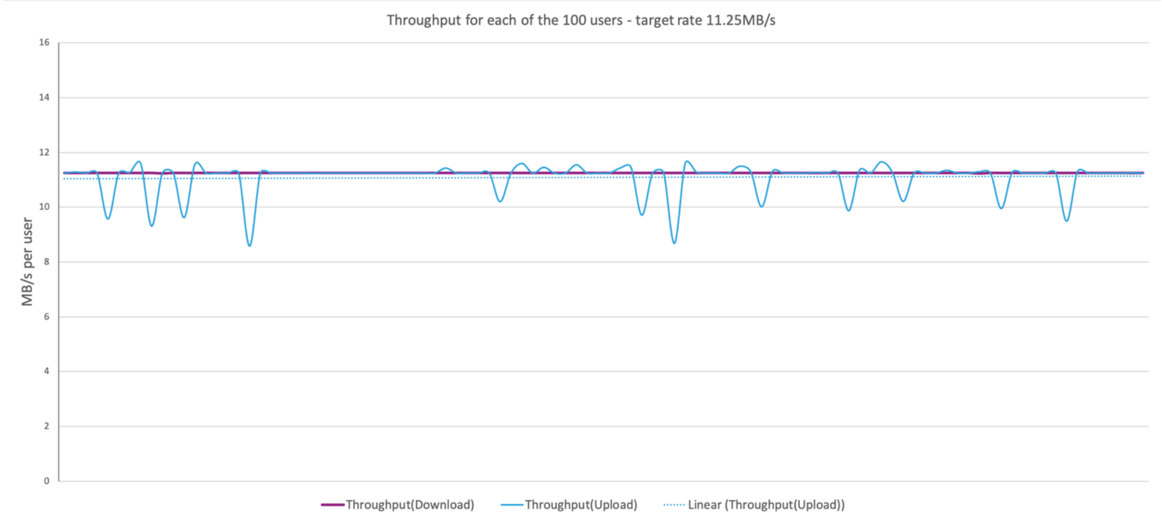
Workload Distribution

Using one 10Gbe interface in and one 10Gbe interface out, the appliance should be able to run at a nominal 10Gb/s or 125MB/s throughput. The linear scalability means that this throughput will simply be shared equally across however many sessions are active. This has been validated for a range of different session counts by using simulated Clients. We could run between 1 and 1000 of these Clients in parallel with varying complexity of firewall rulesets.

With a transmission efficiency of approximately 94% (due to framing overheads, headers, etc.), the best we can expect is a maximum theoretical throughput per user of 10/8/100*0.94*1000 = 11.75MB/s. In reality the actual throughput will be slightly less than the theoretical maximum because the Appgate SDP system itself is using I/O for sending tokens, state information, health-checks, logs, etc. between the different parts of the system. So, for 100 users at a rate of 11.25MB/s/user each, this should be just achievable at 9Gb/s throughput on the Gateway.

Testing Gateway workload distribution on a machine with a 10Gb/s nominal throughput capability.

Test configuration		
100 users	default mtu	GW with 32vCPU



The result is as expected with the upload showing some slight variance due to having 100 processes competing for the 32 vCPU threads available on this specific test Gateway. The average is the most important metric here – it shows that the Gateway distributes throughput across the users in a fairly even way even when running at its theoretical limit.

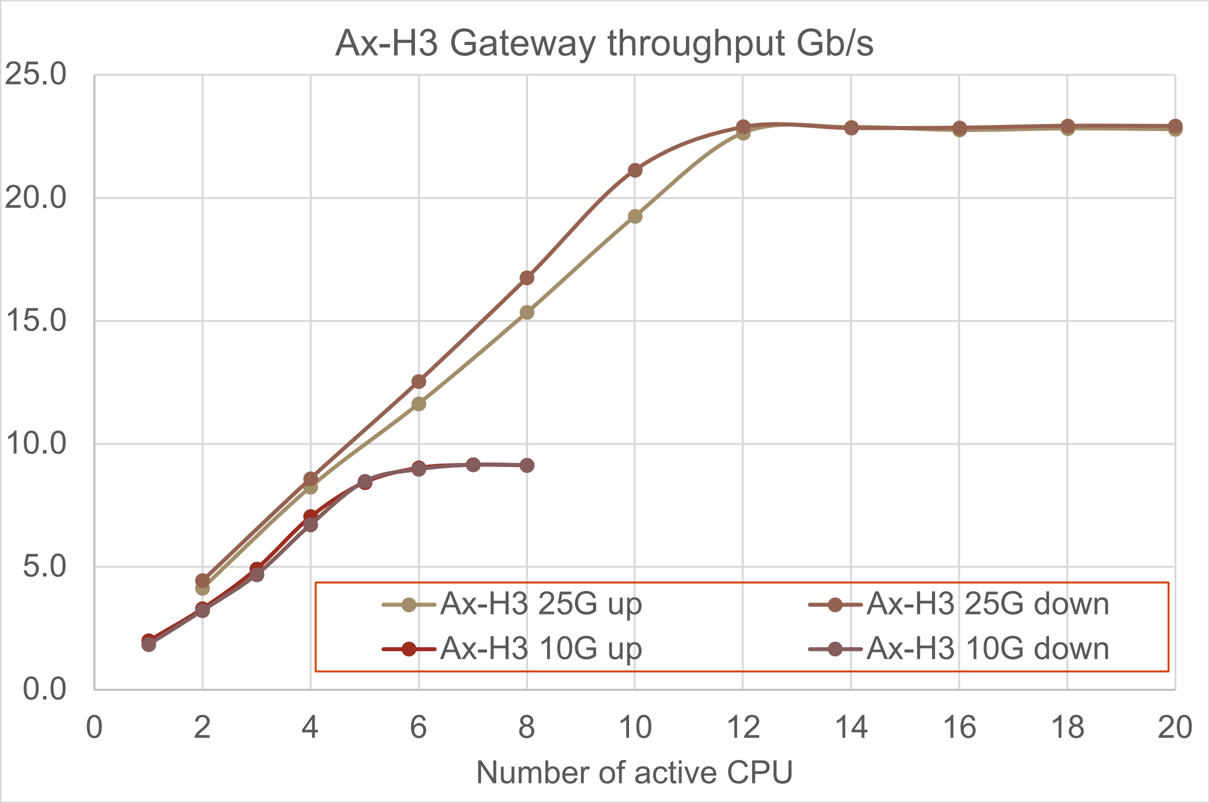
Testing Ax-H$_3$ as a Gateway to identify the maximum throughput

The Appgate Ax-H$_3$ appliance has been designed to meet the needs of a fast Gateway appliance (and a Controller). It has a single processor with 32vCPU. It has a clock speed of 2.9Ghz. It has 64GB of RAM to allow for other functions even with thousands of active sessions.

Test configuration		
24 users	default mtu	AxH$_3$ with 32vCPU

The AxH$_3$ comes with a dual 25Gb/s NIC. A test was run with 24 test Clients to establish the maximum throughput vs number of vCPU active that could be achieved on a v5.3 Gateway. The result was 22.9Gb/s up and down. As before with a 94% efficiency and a 2% allowance for the Appgate SDP system; the theoretical maximum of 23Gb/s has been achieved.

The AxH$_3$ also comes with a dual 10Gb/s NIC. The same test was run again with 24 test Clients to establish the maximum throughput of that NIC. The result was 9.1Gb/s up and down – so almost as good as the highly optimized Ax-G$_3$.



In the linear phase (below 5vCPU), the Ax-H$_3$ requires one more vCPU than the Ax-G$_3$ to achieve the same throughput – but it has a slower CPU - so that is expected.

Below 10Gb/s, the 25Gb/s NIC is less efficient, requiring slightly more vCPU than the 10Gb/s NIC. Again, the throughput had a linear relationship to the number vCPU available up to 12 vCPU. Beyond this point throughput was largely flat.

With only half of the vCPU in use, there is plenty of processing power remaining (and bandwidth using the other NICs) for the Ax-H$_3$ to perform the other Gateway tasks required while also being able to act as a Controller.

Virtual Appliances Performance

Whereas on hardware, having more vCPUs has no noticeable effect on throughput once the network interface has been fully utilized; on virtualized hosts the behaviour is markedly different. In this case, going beyond the optimal number by adding more vCPU, actually reduces performance - and the more you add the worse the performance gets!

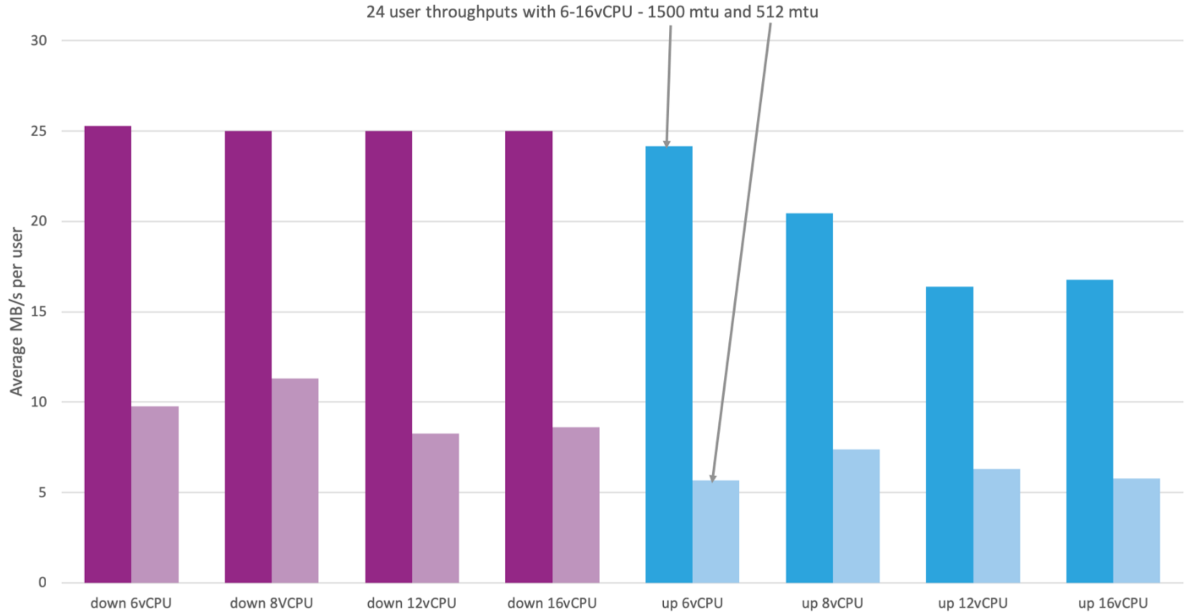
Because there are rather more parts in the puzzle when running on a virtual host, the performance test procedure had to be a bit more elaborate. This performance evaluation was performed on a v4 appliance so v5 appliances can be expected to perform slightly better. However, the point here is not the absolute performance metrics – but the approach and principles used in order to get the best performance.

The Gateway has been designed to be massively scalable with very many simple firewall processes in play at any one time. If you mix the Linux scheduler with the virtual host's hypervisor, and then give it lots of individual firewall processes to juggle; as the number of vCPU goes up, the amount of thread switching also goes up. Linux and the hypervisor try their best maximize the utilization of the available vCPUs, but this is done at the expense of leaving the actual workloads alone to get on with their job. Thread switching is a very costly process and the effect of this is to actually reduce the overall performance of the Gateway as a whole.

To help determine the ideal configuration for a typical virtual host - a KVM instance was used as the host for virtual Appgate SDP Gateway appliance testing. Knowing that we need 6 vCPU (on physical appliances) for 10Gb/s made a good starting point.

Testing vCPU on KVM based Gateway for maximum throughput.

Test configuration		
24 users	512 & 1500 mtu	GW on KVM with 6/8/12/16vCPU



This test also included smaller 512 mtu packet size to apply some extra stress on the Gateway in addition to the default (1500 mtu). In some of these situations (up traffic) 6vCPU appeared to be the better option than 8vCPU - however when you look at consistency across the users and the CPU loading 8vCPU was clearly the best compromise. This result might be expected as you are likely to need slightly more vCPU than when running on a physical appliance. The choice of 8vCPU was reaffirmed several times during testing when more exhaustive checks were also performed at both 7 and 9vCPU.

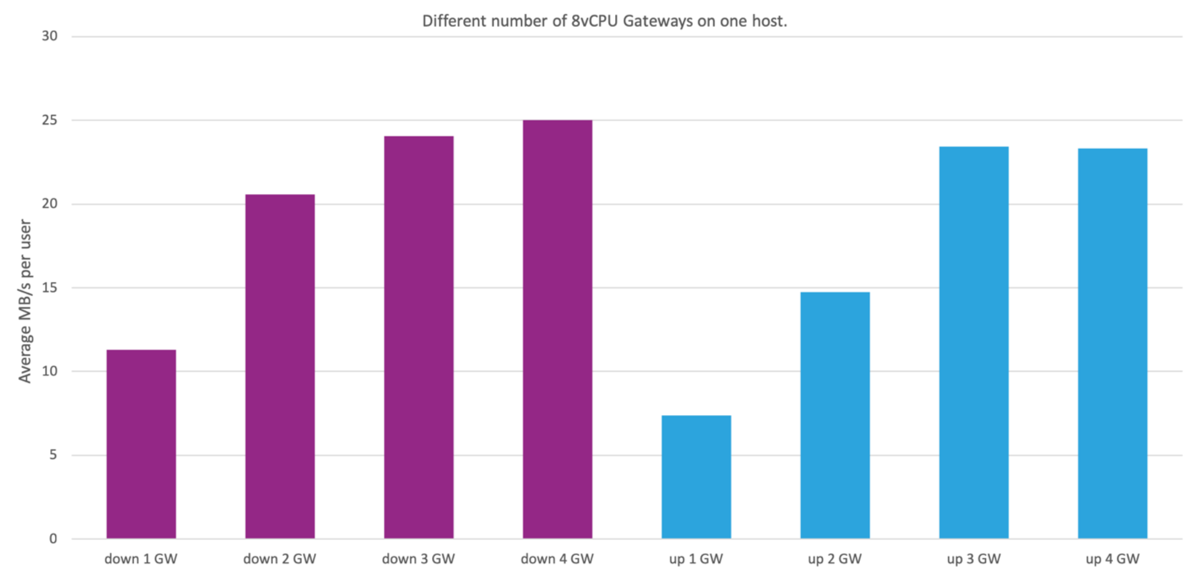
But with throughputs ranging from 5-25 MB/s/user (1-5Gb/s) for a virtual host to deliver 10Gb/s (or even more extreme, 40Gb/s), this out of the box configuration for virtualized Gateways should not be adopted. At best the performance was about half of the equivalent physical appliance.

Since these results show we can’t add vCPU to increase the throughput then the only other option is to add more Gateways to the host.

Testing multiple KVM based Gateways for maximum throughput.

Test configuration		
24 users	default mtu	1-4 GW on KVM with 8vCPU each

In the graph below 1, 2 ,3 and 4 Gateways were tried on one KVM host.



Four Gateways gives a slight advantage over three; but when you take into consideration that computer systems are usually specified in multiples of two (NICs, RAM, vCPU etc), then four might be the better option if maximum throughput is your aim. If not, then there appears to be little to be gained from going over 4 Gateways.

Performing the test again using the default mtu of 1500 and without any rate limit set, both the 3 and 4 Gateway combinations (of 8vCPU each) just hit the 47MB/s/user, which at slightly over 9Gb/s is all you can expect from a 10Gbe NIC.

However, if your host has more than one 10Gbe NIC or a 25/40 Gbe NIC then some additional steps are clearly required to further improve throughput performance. The most important of these is to take advantage of a standard called SR-IOV.

In the world of virtualization, single root input/output virtualization (SR-IOV) allows individual PCIe resources to be managed for say performance reasons. So, a specific PCI Express resource could be shared with a specific guest within a virtual environment on a physical server machine. Using SR-IOV eliminates the vagaries of the host’s virtual network adapter performance and ties the Gateways to the NICs in much the same way as is the case on a physical appliance.

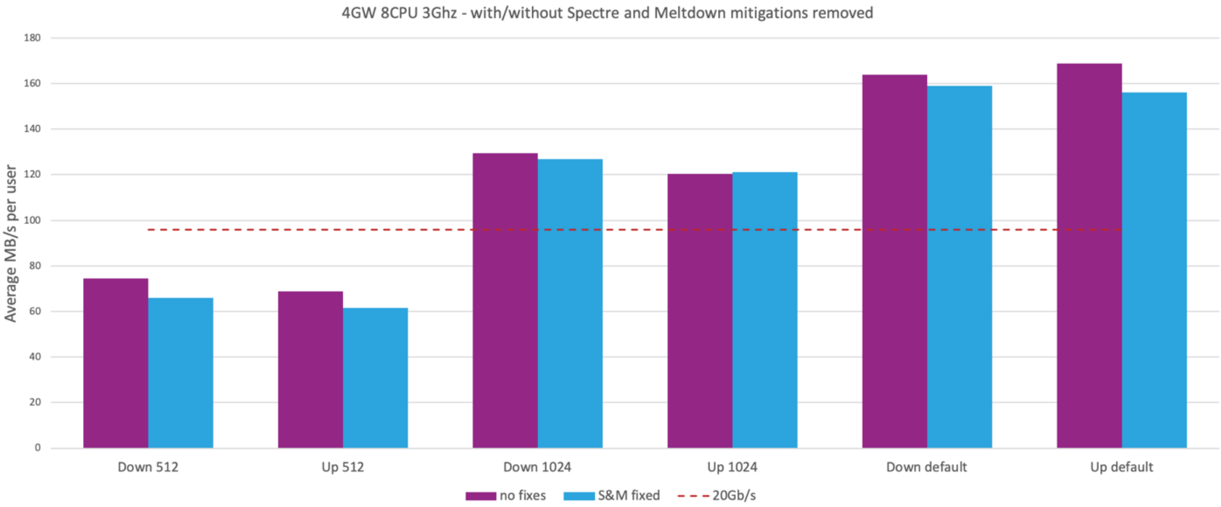
Having found a way to improve the virtualized networking, the next area that needs to be optimized is the actual hardware of the virtual host. There are several things that can be optimized here:

- The type of NIC being used
- The clock speed of the system
- The removal of the Spectre and Meltdown mitigations (on dedicated hosts)

For the tests so far, the host had been using a 2.2Ghz processor with 2133 bus speed. For the next step the server was upgraded to 3.0Ghz processor with 2666 bus speed. The two 10Gbe NICs replaced with one Mellanox 40Gbe NIC. The preferred set up of 4 Gateways with 8vCPU each was adopted. The 4 Gateways streamed using SR-IOV to the 40Gbe NIC.

Testing SR-IOV with 4 KVM based Gateways for maximum throughput.

Test configuration		
24 users	512, 1024 & default mtu	4 GW on KVM with 8vCPU each



The graph shows what all these optimizations have achieved with performance well above 20Gb/s using an average mtu of 1024. At the default mtu of 1500 the advantage to removing the Spectre and Meltdown mitigations can be clearly seen with about 32Gb/s average through puts. But remember this is only suitable for dedicated hosts!

Self-inflicted DoS!

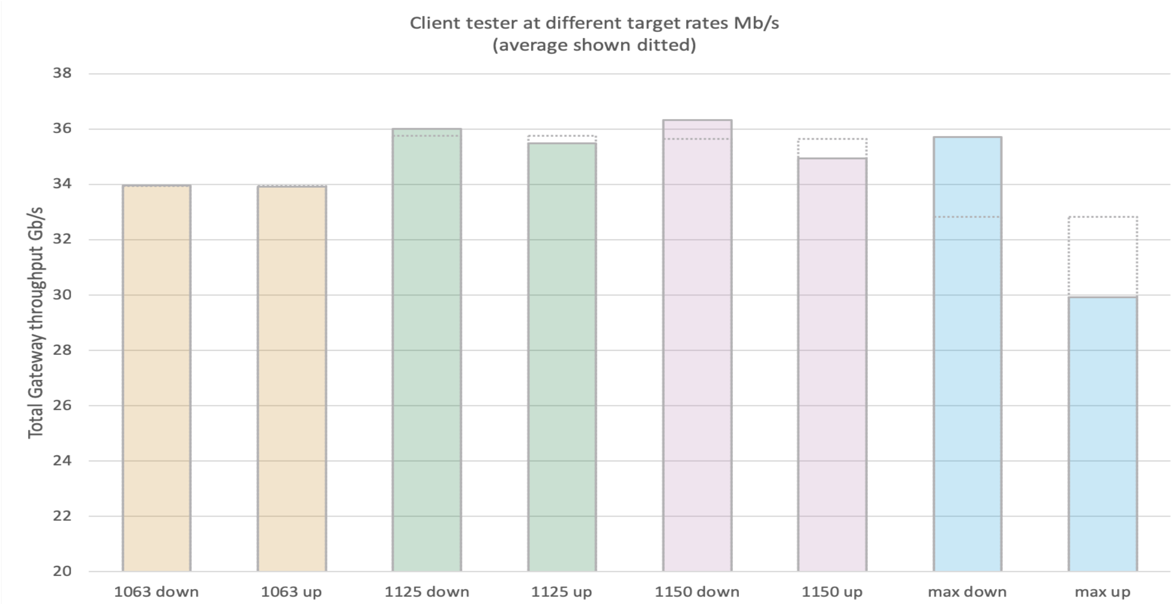
The remaining factor affecting performance is actually the profile of the traffic that the Gateway is expected to handle. Clearly, if the traffic from the Client is less than the capacity of the Gateway then the Gateway throughput will be commensurately lower. Interestingly, if the Client traffic is greater than the Gateway's capacity (packets per second) then it begins to experience what amounts to a self-inflicted Denial of Service attack. In such a situation, trying to deal with this onslaught of traffic means the overall throughput performance actually falls. There is therefore an optimal traffic flow at which the best performance can be achieved.

In this case we have a machine with a 40Gb/s nominal throughput capability. With a transmission efficiency of approximately 94% (due to framing overheads, headers, etc.), the best we can expect is a maximum theoretical throughput per user of 40*0.94 = 37.6Gb/s. Take off something for the Appgate SDP overhead and we get maybe 37Gb/s.

The test clients were configured to send traffic at different rates to evaluate the best rate for throughput whilst avoiding the effect of performing self-inflicted Denial of Service attack!

Testing SR-IOV with 4 KVM based Gateways with rate-limited inbound traffic.

Test configuration		
24 users	default mtu	4 GW on KVM with 8vCPU each



At 1125 Mb/s/Client, the Gateway host is now achieving average up/down rates of 35.75Gb/s. The theoretical limit is 37Gb/s, so this is all that you can realistically expect from a 40Gbe NIC.

KVM vs ESXi

Having arrived at an optimized model, we also performed comparative testing on VMW ESXi. The performance is not as good a KVM. In the above test at 1125 Mb/s down is similar at 36.1Gb/s, but up is limited to just 32.1Gb/s. In testing ESXi, it appears to require more CPU than KVM, this has 2 consequences; firstly, removing the Spectre and Meltdown fixes has a more profound effect of almost 40% better using mixed TCP packet sizes; and secondly ESXi can benefit from more vCPUs (10 instead of 8) particularly the case if the Spectre and Meltdown fixes have not been removed.

Allow for sign-in events

Don’t forget to add some vCPU to allow for token events. The metric we derived earlier will still apply but should be lowered because of the VM host so assume 5 events/vCPU/sec. Try not to add too many or it will negatively impact the throughput performance. If you have a 3000 user system with 5 minute token events then you should allow 2vCPU for this.

Typical single Gateway virtual deployment

Whilst these throughput numbers are interesting for the very largest enterprises, most organizations have more modest requirements, so we also benchmarked a simpler deployment model, more likely to be encountered on a typical virtual host or in the Cloud.

We configured a more typical virtual host specification on KVM which looks like: 1 Gateway; 8vCPU; host clocked at 2.2Ghz; SR-IOV configured to a 2 port 10Gbe NIC (1 port in 1 port out); Spectre & Meltdown fixes still in place. **With this configuration you can expect a maximum throughputs of 6Gb/s up and 8Gb/s down.** This assumes minimal token events/user.

URL Access

Before summarising Gateway performance, we should finally consider the performance of the new *HTTP up* Action type that was introduced in v5.0 (with a single NGiNX instance. In v5.2 NGiNX can now use up to 4 cores (8vCPU), which is the maximum possible. This means that the performance is easy to benchmark using an Ax-G$_3$.

Some simple metrics have been generated to form a rough comparison with the *TCP up* Action type handling the same type of traffic on an Ax-G$_3$ based Gateway.

Comparing the Gateway URL access using NGiNX vs TCP access for a single user.

Test configuration		
1 user	Multiple http and https requests	GW with 8vCPU

These figures are mainly of interest for HTTPS connections and provide some guidance as to performance however we include HTTP ones for comparison. Remember by adding more Gateways to a Site you multiply the capability the *HTTP up* Action type just as you do for *TCP up.*

The figures when using *TCP up* were limited by the source and destination test configurations which were reaching their limits. This test limitation means the *TCP up* Action types are likely to be better than shown and able to handle many times more connections per second.

Firstly, we looked at establishing (new) connections:

New requests	*HTTP*			*HTTPS*		
Action type	**1KB**	**4KB**	**16KB**	**1KB**	**4KB**	**16KB**
TCP up (ms)	0.7	1.1	5.8	5.2	5.5	8.7
TCP up (req/s)	131k	103k	46k	28k	26k	20k
HTTP up (ms)	34.3	29.1	42.7	12.2	12.5	14.1
HTTP up (req/s)	24k	18k	30k	8.1k	7.9k	7.4k

For *TCP up* the size of the payload is the only real consideration. HTTPS is bigger than HTTP and 16k is bigger than 1k! For *HTTP up* the HTTP traffic is representative of using the alternative path through the Gateway (via NGiNX) and is still able to manage about 20% of the requests/s. With HTTPS the types of operation changes markedly with more cryptographic operations slowing the rate of requests; the compensation being reduced concurrency allowing faster average time per request.

Then, we looked at existing connections:

Existing connections	*HTTP*			*HTTPS*		
Action type	**1KB**	**4KB**	**16KB**	**1KB**	**4KB**	**16KB**
TCP up (ms)	0.8	1.7	5.9	1.0	2.3	6.1
TCP up (req/s)	485k	228k	68k	397k	169k	65k
HTTP up (ms)	60.7	53.2	48.1	38.6	40.5	44.1
HTTP up (req/s)	46k	41k	31k	10k	9.9k	9.0k

For *TCP up* there is little for the Gateway to do apart from delivering the payload hence the high rate/second. For *HTTP up* there are less cryptographic operations which increase the rate at which the system can handle requests; the downside being increased concurrency which makes average time per request longer especially for small payloads where there are still the fixed overheads such as having to write audit log records.

Use of HA Gateways

A highly-specified appliance such as the Ax-H3 (CPU and RAM) can fully support a 25Gbe NIC. This is the practical limit within a single Appgate SDP Gateway appliance today. However, the system has been designed to support multiple Gateways on each Site. There is no established limit to the number that can be deployed; and because there is no inter-Gateway communication, Gateways will scale linearly. If one appliance can handle the throughput needs for say 5000 users, then six on the same Site should be able to handle 30,000 users.

The other tangible benefit of using multiple Gateways is for hardware failover. This can be configured to happen using either state-less or stateful failover.

Remember for virtualized appliances you should be using two different hosts for hardware failover (they must use SI-IOV and have no more than 4 Gateways per host). If using Cloud Gateways, then specify them in different availability sets or groups to avoid risk of co-habitation.

Gateway Summary

The CPU analysis showed that it is possible to specify systems such that they can reach the point where the I/O on any given appliance is running at its theoretical data throughput limit. This is mainly restricted by the amount of CPU available. On physical appliances, 4-5vCPUs are enough to take full advantage of a 10Gbe NIC and 16-18vCPU are needed for a 25Gbe NIC. Any spare vCPU will be used by sessiond for handling token events. Consider adding more vCPU if this will present a significant load (i.e frequent events).

From the earlier analysis it has been shown that with smaller packet sizes, the number of packets rise and the maximum throughput in bytes/s actually falls. The figures are roughly: 80% with mtu of 1024 and 40% with mtu of 512. It is therefore important to understand the packet size mix that will be handled by the system. With this information you can re-rate the specification of the Gateways to suit.

To summarize:

- The Ax-G3 and Ax-H3 will both operate up to their built in I/O capabilities. Where higher throughputs are required then multiple Gateways should be used.
- For CPU - decide on the throughput required for the Gateway and then specify the number of vCPU required. On virtual appliances you should assume about twice the number of vCPU to achieve the same throughputs as on physical. Remember – only go above 8vCPU on a virtual host if sign-in events matter more than throughput.
- For RAM - there is a base requirement for the OS – allow 1GB, and there is the Appgate SDP system requirement – allow 1GB. On top of that there needs to be an allowance for the number of concurrent users. Remember - never run out of RAM as the system does not use disk caching.

Rule of thumb: Each vCPU should be able to deliver at least 1Gb/s of throughput. Allow a similar number of vCPU to the Controller for peak sign-in loads based on 5-10 events/vCPU/second. Remember users can't pass traffic until they have signed-in!

Rule of thumb: *Specify RAM according to your use case. Allow 2GB* + 800KB/user

See Instance Sizing in the manual for the latest recommendations.

CLIENT

The Appgate SDP Client comes in many flavours, the majority of which use a multi-tunnel network adapter. This is used to establish multiple TLS tunnels to the Gateways that handle the application traffic from the users. The Lite Client uses a different approach based on a local web server, PAC script and an HTTP proxy that provides multiple TLS connections with the Controllers and Gateways. Since both use TLS there are no differences as far as the Controller and Gateway are concerned.

The Client performance has been improved over the various versions however, in real life there are many factors that might limit a user's throughput performance. For any given Client these are likely to be a result of one or more of the following:

- Hardware limitations in the user's device restricting the performance of the tunnel(s)
- Local network limitations such as having access to only 100Mb/s or 1Gb/s networks
- Other applications running on the user's device consuming resources
- Gateway NIC limitations when it is running near its maximum throughput
- Gateway CPU where the user's single vCPU process is the limiting factor

Performance testing of Clients is therefore a somewhat academic exercise. However, it is interesting to establish some sort of performance envelope for the Connector and for the headless Clients which might be used on powerful servers to provide (reverse) access to a specific resource for very many users.

Benchmark Testing

To establish the practical limit for Client performance a server was used which in theory, offers premium performance including a 10Gb/s network interface. The appropriate Appgate SDP Client was then installed. This is a very valid test for server-based Clients. For desktops, because the Client is not multi-threaded it will not be taking advantage of the available CPUs and is actually limited by the clock speed (which can often be somewhat higher in many user's devices). Download and upload speeds were tested in each case.

Linux

The Linux Client was installed on Ubuntu 18.04. Iperf3 and Curl were used to check the performance. For curl, both http and https performance were measured.

MB/s	*Iperf3*	*Curl http*	*Curl https*
Up	345	305	270
Down	340	333	329

Performance in the region of 2.5Gb/s is therefore achievable from a Linux server.

Windows

The Windows Client was installed on Windows 10 and on Windows Server 2016. Again Iperf3 and Curl were used to check the performance. For curl, both http and https performance were measured.

MB/s	*Iperf3*	*Curl http*	*Curl https*
Lite W10 up	-	29	29
Lite W10 down	-	76	72
Full W10 up	269	89	89
Full W10 down	194	226	220
Full WS2016 up	298	256	257
Full WS2016 down	262	211	221

Performance is not as good as Linux but 2Gb/s is achievable from a Windows server.

macOS

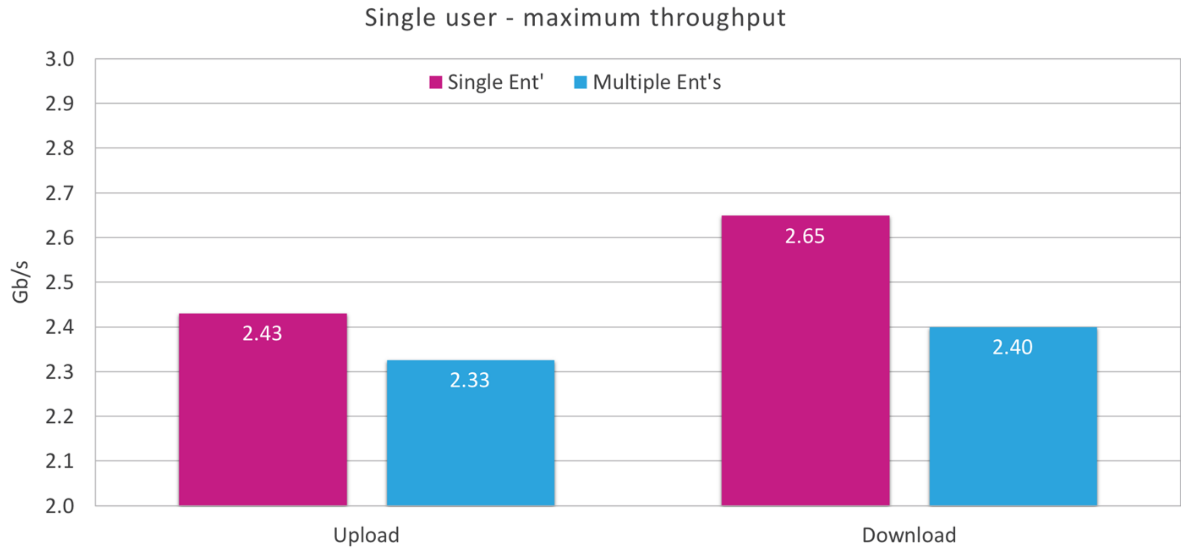
macOS is only supported for desktop users so is not as critical as Linux and Windows servers. V5.x has been tested on a macbook using a thunderbolt 1G wired ethernet adapter. Curl was used to check both http and https performance to a test server. The results for both were the same at 0.86Gb/s down and 0.8 Gb/s up which is close to the 1Gb/s hardware limit.

Multiple Entitlements

With measured performance levels now consistently exceeding what most users' PCs can deliver; another test has created to better measure real life performance. This test is performed twice; once with a single Entitlement and again with multiple Entitlements (more similar to a real user).

As the graph below shows the results for a specific test set up using v4.1 of the Client and performance BIOS setting:

Single Entitlement performance was upload 2.43Gb/s and download 2.65Gb/s. With multiple Entitlements this was degraded: upload 96% and download 91%. The degradation in performance was expected, even so users with multiple entitlements are only likely to see a reduction of 10% in their theoretical maximum performance.



CONNECTOR

Connector Performance Metrics

The Connector is the same core appliance build but optimised for running multiple instances of the Linux Client. Each of these instances runs in its own Linux namespace and can be assigned its own Policy.

Unlike Controllers and Gateways, the Connector does not have a large Java application running - so its memory footprint is noticeably smaller.

There are several parameters which should be considered when specifying a Connector:

1. Disk – is non-critical
2. RAM – only minimal requirements
3. Clock speed - will affect i/o performance because each Client is *not* a multi-threaded process.
4. CPU – A minimum of 2 is recommended for basic operation. More should be specified to achieve higher total throughputs or when specifying significant numbers of Policies/Clients.
5. I/O – unlikely to represent a significant issue as throughputs of several Gb/s are possible.

Disk

A stand-alone Connector requires only 4GB of disk space but to allow for upgrades and the storage of local logs (max 4GB), we suggest 20GB minimum.

The only audit logs generated relate to appliance events which are very rare and for most enterprises these are usually moved to either an Appgate SDP LogServer or some other external SIEM. For the Connector, debug logs include the Client logs which will only amount to a few MB per Policy/Client. This means that disk speed (and size) is much less critical than with other appliance types.

Appgate’s physical appliances are specified with faster SSD disks to ensure audit log writing does not impact the overall system performance. These are more than adequate for the Connector.

RAM

Without any Java processes, the RAM usage is less than the other appliance types. But, you should still budget 0.5GB for the OS/appliance and the minimum one Policy/Client even though they use rather less than this.

Each Policy/Client has the same memory requirement, so it is very easy to estimate the incremental requirements. Allow 250MB for the base appliance + 25MB per Client instance. 30 Policies/Clients would therefore require an appliance with 1GB of RAM. You should always over-specify the amount of RAM required - if you specify 2GB of RAM then this should cover all eventualities.

NOTE: The Appgate SDP appliance does NOT use disk caching for RAM. So, when the appliance runs out of RAM it does just that. The OS will shut down daemons it considers to be using too much memory so user connections will be interrupted. It is therefore important to ensure enough RAM is specified when sizing appliances.

Clock Speed

The Connector uses an embedded version of the Appgate SDP headless Client. This does not support multi-threading so more vCPUs will not help individual Client performance;

however clock speed will help with the handling of I/O so the faster the CPU the better in this respect.

CPU

As with Gateways, throughput is the metric that matters, and CPU is the key parameter which affects the overall performance of the Connector. Testing has been performed on two different physical appliances which also provides us with enough information to be able to specify any virtual appliances that might be required.

The Connector itself could be considered a sort of virtualisation host and the Clients the guests. The Connector therefore bares some similarities to the virtualised Gateway model we discussed earlier. Policies/Clients are all the same as each other so adding many to the one host again breeds a kind of mono-culture. This will typically starve the host of some specific types of resources which will show up as a non-linear scaling.

CPU is consumed both by the appliance (which handles all the traffic routing and I/O flows) and by the many Policies/Clients that may have been configured. Test results include the minimum recommended configuration of 2 vCPU.

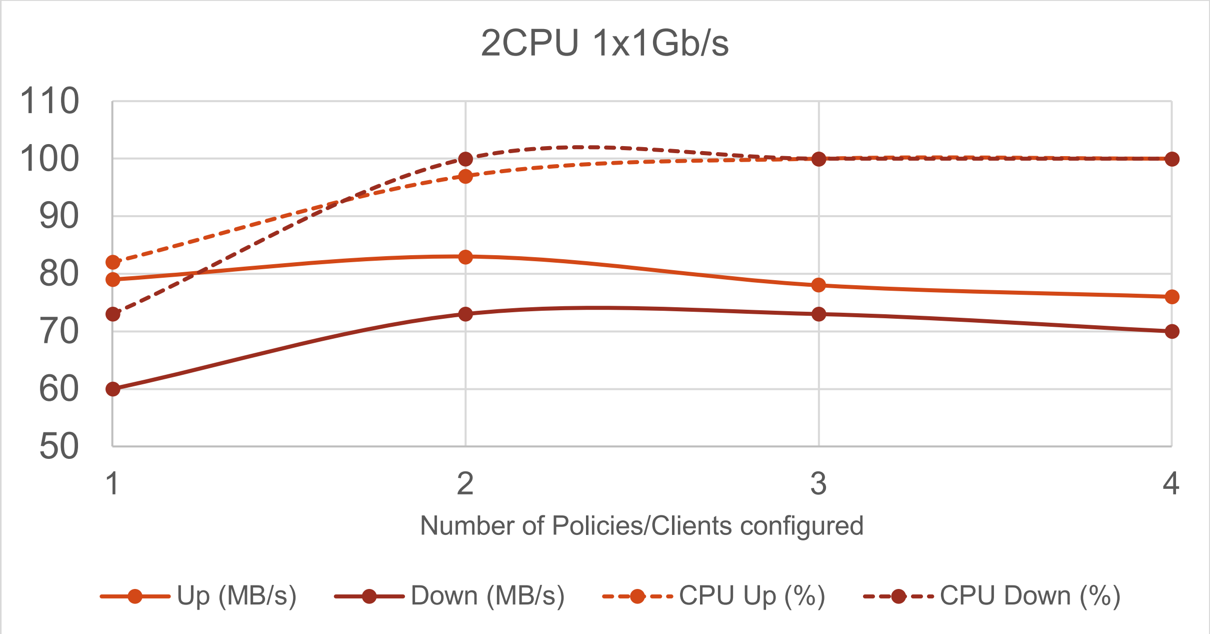
Physical Ax-M performance

The Appgate Ax-M appliance has been designed to meet the needs of a Connector. This is a small footprint fanless device. It has a single processor with 4CPU with a clock speed of 1.1Ghz (2.5Ghz boost). It has 4GB of RAM - which is more than enough. There is a 2 port 1Gb/s NIC.

All tests were performed with the mitigations for CPU based vulnerabilities such as Spectre and Meltdown turned off - as these do not apply in the case of a stand-alone appliance.

Testing Connector Ax-M appliance maximum throughput using only 1 NIC.

Test configuration		
1-4 Policies/Clients	1*1Gb/s	IoT Cn. with 2vCPU

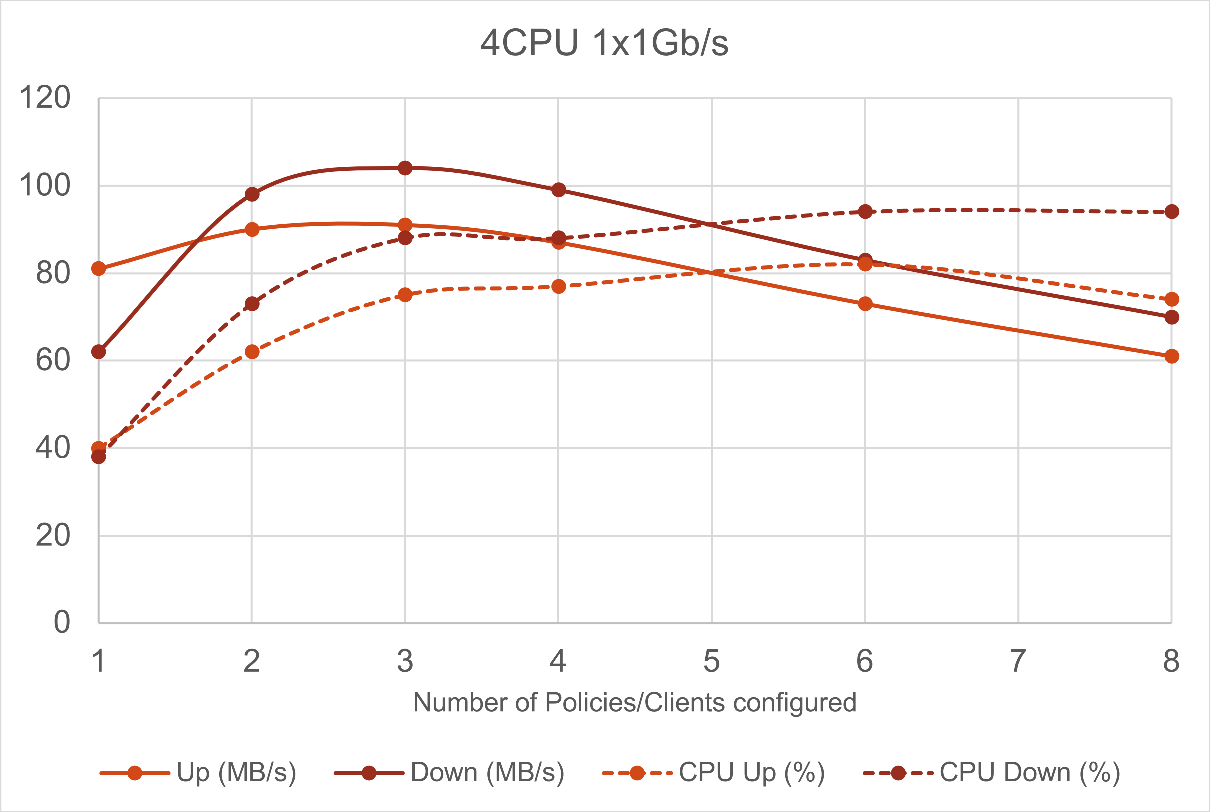


This arrangement can be used when only 1 (or 2) Policy/Client(s) are required and where throughputs of up to 0.5Gb/s suffice. This might for instance be a good way to support a remote printer which needs to be accessible within the Appgate SDP Collective.

Even though this arrangement can support more Policies/Clients (up to 4) we do not recommend running the CPU at 100% so throughputs would need to be limited to a few 100Mb/s.

Testing Connector Ax-M appliance maximum throughput using only 1 NIC.

Test configuration		
1-8 Policies/Clients	1*1Gb/s	IoT Cn. with 4vCPU

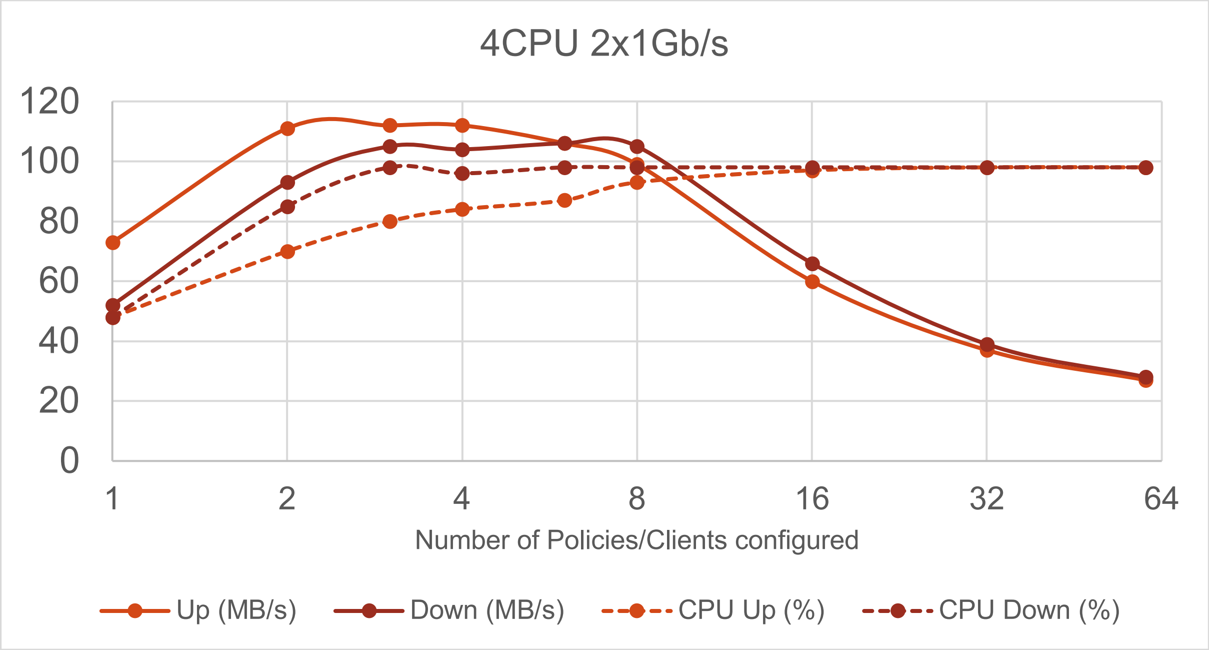


This arrangement can be used with more Policies/Clients (4-6) and where throughputs of up to 0.7Gb/s suffice. This might for instance be a good way to support a small remote office where minimal disruption to the existing network is required but where users, printers and IoT devices should be handled separately.

This arrangement becomes less efficient with more than 4 Policies/Clients. By the time 8 is reached the CPU usage is actually falling because of I/O congestion blocking traffic.

Testing Connector Ax-M appliance maximum throughput using 2 NIC (one in one out)

Test configuration		
1-60 Policies/Clients	2*1Gb/s	IoT Cn. with 4vCPU



This arrangement can be used with even more Policies/Clients (8-12) and with throughputs of up to 0.9Gb/s which is the maximum expected from a 1Gbe NIC. This might for instance be a good way to support a site that has many local devices of several different types (each using their own Policy/Client), which all require isolation from the internet.

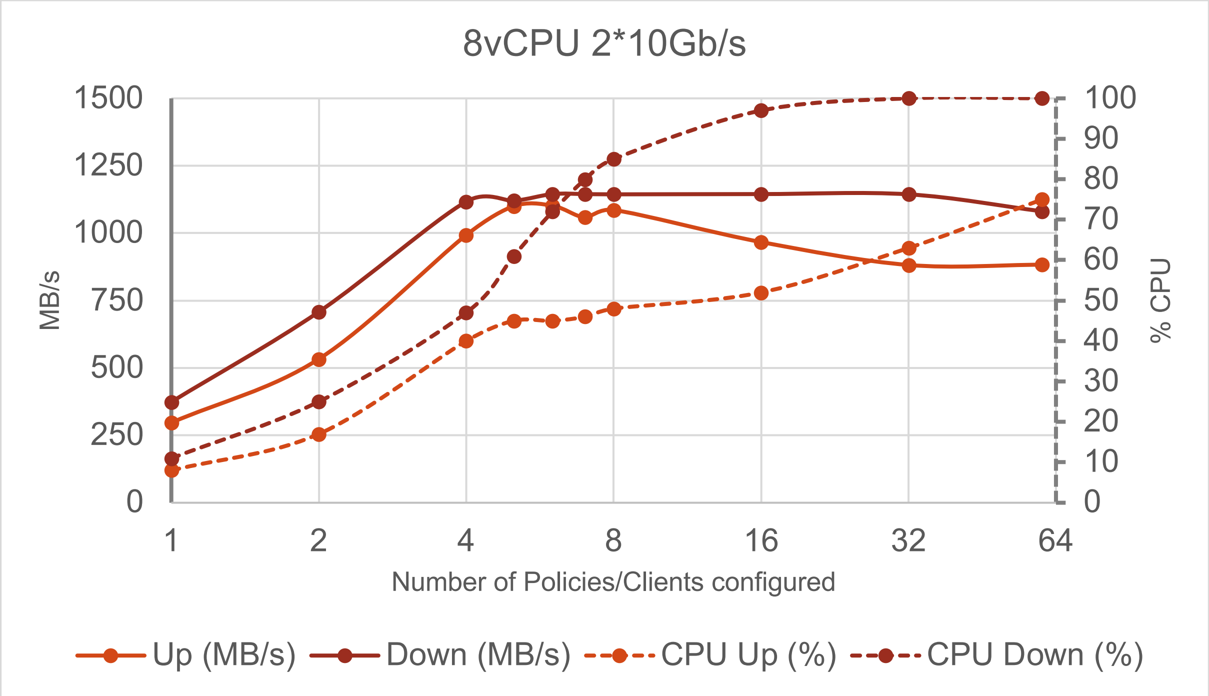
Beyond 12-16 Policies/Clients this arrangement becomes much less performant as thread swapping starts to take its toll. 16 Policies/Clients should be considered the realistic limit.

Ax-G

There are situations where 1Gb/s NICs are not adequate for the throughput required. To support faster NICs more CPU will be required, so to establish the maximum performance envelope of the Connector the Ax-G appliance (8vCPU) was used. This appliance type was originally specified for Gateway use and whilst over-specified in terms of RAM, it has much more capable CPUs which happen to provide the near perfect CPU to NIC balance when it comes to delivering the maximum throughput that can be achieved by the Connector.

Testing Connector Ax-G appliance maximum throughput using 2 NIC (one in one out).

Test configuration		
1-60 Policies/Clients	2*10Gb/s	IoT Cn. with 8vCPU



The single Policy/Client has performance which is in line with the Linux Client numbers discussed earlier. With 2 Policies/Clients configured throughputs of about 5Gb/s are possible - demonstrating the benefits of a very fast CPU (given that each Policy/Client is limited by the single-thread process it uses).

Between 4 and 8 Policies/Clients the performance (9.2Gb/s) becomes constrained by the 10Gb/s NIC itself. Once the number of Policies/Clients exceeds the number of vCPUs then swapping and resource bottlenecks caused by the monoculture in the Connector start to limit the up traffic. The down traffic becomes limited once the CPU is 100% busy. Even so the Ax-G could sustain 8Gb/s up and down with 16 Policies/Clients configured.

Use of HA Connectors

The Connector can be specified for HA operation by having them share a virtual IP. However they operate in active/passive mode so there is no performance benefit to be gained.

Connector Summary

The Connector is not a very resource hungry appliance.

2-4GB RAM is the most that will ever be required. 2vCPU are recommended. 4-8vCPU should be specified as the number of Policies/Clients and throughputs increase. Bi-directional throughputs that fully utilise 10Gb/s NICs are possible on appropriate hardware.

Rule of thumb: Always use a minimum of 2vCPUs and…

- add more vCPU as the number of Policies/Clients increase (matching numbers is good)
- specify faster CPUs if you want to achieve Connector throughputs >1Gb/s

See Instance Sizing in the manual for the latest recommendations.

CONCLUSION

The scale/performance limit of the Appgate SDP system is defined by the resources available within the Collective as opposed to any specific design limitation.

- Controllers are CPU sized according to sign-in/cpu/second.
- Gateways are CPU sized according to required throughput with 8vCPU being a preferred number. RAM is sized according to user/rule memory limit.
- Both Controllers and Gateways can be sized-up in terms of I/O, CPU or RAM to meet a given requirement. Beyond that they can both be teamed-up to increase scale/performance of the entire collective in a linear way.
- Clients (on the right hardware) can operate way beyond their normal network speed (1 Gb/s).
- Connectors are light weight appliances that just require enough vCPUs (maximum of 8vCPU) to get the job done.

The Appgate SDP system easily meets the requirements most enterprises demand in terms of complexity, scale and performance. At one extreme, individual users can enjoy speeds of 2Gb/s, and at the other 100Gb/s of total throughput can be easily achieved on a Site. And all this within a single Collective supporting many tens of thousands of simultaneous users connected and working.

ABOUT APPGATE

Appgate is the secure access company that provides cybersecurity solutions for people, devices, and systems based on the principles of Zero Trust security. Appgate updates IT systems to combat the cyber threats of today and tomorrow. Through a set of differentiated cloud and hybrid security products, Appgate enables enterprises to easily and effectively shield against cyber threats. Appgate protects more than 650 organizations across government and business. Learn more at www.appgate.com