Spec-Guide Proxy Capacity Analysis: Charter Internal Note. Draft Version 1.0.0

Brad Schoenrock
Video Operations Engineering
Charter Communications
Greenwood Village, CO

Oct. 2019

Contents

1 Introduction **3**

2 Networking Usage **3**

3 CPU Utalization **4**

4 Memory Consumption **4**

5 Disk Usage **4**

6 Application Performance **5**
6.1 Apache TrafficServer 5
6.2 Advanced Intrusion Detection Environment(AIDE) 7

7 Conclusion **7**

1 Introduction

Spectrum guide proxy servers (also known as cache servers) serve as a reverse proxy and a parent cache to the stitchers. When the SGUI application on the stitcher requires an asset to be displayed for the user it must retrieve that asset from its source. If the stitcher has retrieved this asset before then it may be located in the local Apache Trafficserver (ATS) instance which caches that asset, and a call to the source may not be nessicary, improving response time of the application. If the asset is not in the local ATS instance then the stitcher reaches out to the proxy, and if the asset has been retrieved by any stitcher in the RDC then proxy may find it in the parent ATS cache on the proxy. If the asset is not located in the parent cache then the proxy makes the call to the source to retrieve the asset. The proxy may store the asset in its parent cache, and pass it to the stitcher to be stored in its local cache as it is being rendered for the requesting user. Since the stitcher doesn't reach out directly, the IP of the stitcher is not exposed, and so the proxy provides a layer of security as well as improving the response times for asset retrieval.

Since the stitchers run their own local cache the calls to the proxies are limited. Each stitcher will reach out to the proxy once in the application defined TTL. Since the local cache limits the calls to the proxy in this way the growth of the customer base does not directly increase the load on the proxy. The load on the proxy instead scales with the number of stitchers on the market, and so indirectly scales with customer count. The number of stitchers increases with customer count, but when stitchers are replaced for end of life concerns they are replaced with higher memory densitiy blades which will lower the overall count. This will naturally happen as memory becomes cheaper and denser over time in accordance with Moore's law. A larger factor to consider with proxies is the utalization by features of the SGUI application, and the configurations associated with asset retrieval.

The proxies should be assessed for networking usage, CPU use, memory consumption, disk space utilization, and the performance of ATS. Note that there are daily spikes in proxy resource use occour at midnight local time for CPU and memory use. This is due to

2 Networking Usage

Each proxy is peaking at less than 100MB of bandwidth use. Networking resources allocated to the proxy are allocated by VMWare as a distributed port group, which is managed by VMWare. Each hypervisor has multiple 10GB NICs bundled together to serve all of the VMs on the hypervisor. Bandwidth is dynamically allocated to VMs based on need at the time, and a thourogh understanding of VM bandwidth usage will be left to the hypervisor capacity analysis.

3 CPU Utalization

CPU use of the proxies is quite low. With 4 cores per node usage remains at approx. 3-8% used. Every market has 4 CPUs except RENONV and TWCHI which have 2 cores per node. The AIDE utility does push that usage to 100% but see section 6.2 for details on that process. With the execption of AIDE configuration there are no concerns over CPU use in any markets.

4 Memory Consumption

Proxies have 4GB of memory per node. The usage of RAM is dominated by ATS, which allocates a RAM cache for the most accessed objects for faster retrieval. The default setting for this RAM cache is approx. 10MB per 1GB of DB allocated. For a 15GB DB 150MB of ram is expected to be used solely for the in RAM cache ATS allocates. Raising the size of the DB means that more memory would be used. Currently ATS is using approx. 1GB of ram in total, so raising the size of the DB should increase ATS memory usage. Since ATS is the only thing using significant amounts of memory on the proxy, it can be increased, up to a point, without concern. If the DB were to be expanded beyond approx. 48GB then fixing the size of the RAM cache instead of letting it float based on DB size would be reccomended in order to avoid running out of memory on the Proxy.

5 Disk Usage

It is reccomended that ATS DB on proxies be 32GB and stored on its own 50GB partition within the filesystem /var/cache/trafficserver. Currently only renonv matches that standard, while all other live proxies have 15GB DB and are not on their own partition.

Stitchers already have ATS DB on a seperate 21GB partition, with a DB size of 15GB. Since the stitcher is a local cache and is expected to be served from the parent cache the smaller size is accepted since retreival from the parent cache on the proxy is quite fast. The local cache on the stitcher is also expected to be lower since not all assets will be retrieved for all stitchers, but the proxy serves the whole market and so holds all accessed assets.

ATS logs its own processes in binary format under a shared logging mount in /var/log. The binary format can be decoded locally on the proxy with the traffic_logcat command, but are not able to be uploaded into a querieable application such as Kibana or Splunk. Formatting in binary is a configurable setting, and it would be worthwhile for those logs to be saved in ascii and uploaded in order to be visible to other groups in the organization for debugging their own processes which might be using ATS on both stitchers and proxies.

6 Application Performance

6.1 Apache TrafficServer

ATS on the proxy serves as a parent cache to the stitchers. The performance of this cache is important to the latency and responsivness of the Spec-Guide experience. The performance of the stitcher cache is less important since if an asset is not found in the stitcher cache the asset can be found in the parent cache. Pulling the asset from the parent cache is an operation with very low latency because the parent cache is located in the same datacenter as the local stitcher cache. The performance of the local stitcher cache helps to reduce load on the parent cache on the proxy, and so if load on the parent cache becomes high an optimization of the local stitcher cache can be performed.

Much has been written about optimizing the performace of caching systems. Optimizing the CPU usage and memory usage can lower the load on the underlying hardware, but since ATS is performing within acceptable bounds of the underlying hardware these optimizations are not nessicary. What we are concerned with here is how well ATS is serving the SGUI application, and that is heavily dependant on configurations of the application. When an asset is being retrieved for use it is important that it exist in the cache if it is possible that it be located there. All assets should have an application configured Time To Live (TTL). The TTL ensures that assets which change over time get refreshed at reasonable intervals (like sports scores or weather updates) and that assets which are no longer needed are removed so they are not stored (like movie posters when Spec-Guide stops hosting that piece of content). The degree to which assets are available can be measured by the number of hits and misses in the trafficserverDB.

Log files for the trafficserver can be analyzed with the traffic_logcat command. The results from 10 min of logging for one node in SLDCMO are shown in table 1. The first row recognizes all attempts to the cache. TCP_HIT and TCP_MISS are how many times they were or weren't found in the DB. TCP_REFRESH_MISS and TCP_REFRESH_HIT are how many times an expired token in the DB had to be verified as stale and had to be refreshed, or was verified as not changed with an if-modified-since request. TCP_IMS_HIT and TCP_IMS_MISS are when an if-modified-since request is explicitly made, and the cache responds accordingly. TCP_MEM_HIT is when the object is found in ATS memory cache, and the DB isn't accessed. There is no TCP_MEM_MISS because it is expected that a majority of requests are not found in the relativly small memory cache.

Since our goal is to reduce calls to the source endpoints the maximization of the ratio of TCP_HIT to TCP_ messages should be optimized. The ratio of TCP_HIT to TCP_MISS messages works as well since the majority of messages (approx. 80%) are TCP_MISS. This shows that this ratio is TCP_HIT/TCP_MISS = 4.6%. One reason for a low hit miss ratio could be that ATS is too small for our needs, and would require additional resources to address the expected utalization of the SGUI application. Another common reason for a low hit miss

message	count
“TCP_”	51468
“TCP_HIT”	1929
“TCP_MISS”	41589
“TCP_REFRESH_MISS”	771
“TCP_REFRESH_HIT”	289
“TCP_IMS_HIT”	3078
“TCP_IMS_MISS”	3781
“TCP_MEM_HIT”	31

Table 1: Hit Miss results from trafficserver logging. TCP_HIT/TCP_MISS = 4.6%

ratio is that we operate with very large sets of cacheable data which sees sparse access, and so assets are only accessed once or twice before expiring. To address this case a review of how we configure ATS headers on the application would need to be performed and assets either need their TTL increased to match expected usage or have a no-cache and no-store header added so these assets are not being stored in ATS if their usage is too sparse to merit caching. Lastly if assets are being given a no-cache header but are not being given a no-store header entry then this means that the asset will not be accepted from the cache, but once retrieved from the endpoint it will be stored in the cache regardless. All calls in those cases would be registered as TCP_MISS regardless of the apparent intention of bypassing the cache.

By performing this hit/miss analysis by endpoint we can identify which endpoints are routinely missing, and with what frequency. The front end team responsible for SGUI provided a list of endpoints with no-cache headers configured, and those endpoints are registering TCP_MISS messages. It was discovered that an ATS configuration to ignore no-cache headers was enabled, and so when a no-cache header is being provided by the application, it gets ignored and cached anyway registering a TCP_MISS. The most notable endpoints which operate in this way are vinona data collection and the weather edge application. Once the endpoints which have no-cache headers configured are removed from the analysis, the hit/miss ratio comes up to 53%.

The expected hit/miss ratio for high access items that are supposed to be cached should be (n-1)/n for the proxy cache where n is the number of stitchers. This is because the stitchers have a local cache, and only the first stitcher should miss the cache while the rest hit since that call from the proxy has already been made. Any subsequent calls on that stitcher are then found in the local stitcher cache and don’t make the call to the proxy. In sldcmo for instance we have 136 stitchers and should expect a best case hit% of 99.2%. That represents an idealized (and frankly unachievable) scenario but with proper configurations the 70-90 percent range should be achievable.

A set of configuration changes is reccomended to help optimize ATS, and to improve the robustness of the system. First, a change to logging is reccomended

so that log files are stored in ascii format instead of binary. That will allow ATS logs to be uploaded into Kibana and/or other log aggregators to be viewable to other teams outside VO who might need to know about how their endpoints are being handled through the caching process. Second, enabling negative response caching on both stitchers and proxies will prevent excess load from being put on the proxies in the case of a missing endpoint. By enabling negative response caching a failed endpoint won't be hit repeatedly and will be stopped by the stitcher before getting to the proxy. Third, imposing a max TTL on ATS will ensure that all data held in ATS will expire at some point. This provides a measure of protection against tokens existing in ATS for prolonged periods of time past their application defined TTL. Fourth, disabling the setting which ignores no-cache headers (which will allow us to respect a no-cache header provided) will allow us to not cache things which we don't want to cache, avoiding unnessicary load on the proxies. Lastly increasing the DB size is advisable. The machines already have hard drive space for an increased DB (DB is 15 GB on a 71GB partition), so using that resource would be advisable. It would also be advisable to give the trafficserverDB its own partiion on proxies similar to how they are set up on stitchers.

6.2 Advanced Intrusion Detection Environment(AIDE)

AIDE is a host-based intrusion detection system which takes a "snapshot" of the state of the system, register hashes, modification times, and other data regarding the files defined by the administrator. With this snapshot malicious changes to the filesystem can be detected and reported. Since the proxy serves as an intermidary between the stitchers and the microservices endpoints it is important that it be a focus of security. AIDE scans the system once per day, scheduled for 06:00UTC. When that scan occurs there is a large spike in resource utalization, taking up all available CPU resources for the duration of the scan. The extent of this load could be mitigated by reducing the number of hashes required from 5 (sha1, rmd160, sha256, sha512, and tiger) down to two (or for extra paranoia three). Another option which would be benificial would be running AIDE with the nice utility built into linux. The nice utility lowers the priority of the AIDE process allowing core functionality of the proxy to continue by de-prioritizing the AIDE process resource consupmtion. The AIDE process would still be running, and if resources are available the entire CPU would be used, but other processes (like ATS or system level functions) would be given priority to CPU resources.

7 Conclusion

Reconfiguration of the applications on the proxies is highly reccomended. Reconfiguration of AIDE will yield benifits of preventing full VM resource consumption on a daily basis. Reconfiguration of ATS will allow logs to be aggregated for increased visibility and will yield large gains for ATS optimization,

but full optimization is dependedent on developers being conscientious of their use of the resource.

This effort was made possible by inter-team cooperation. The front end team informed VO of the Real Time Sports (RTS) feature they were developing, and it was because of that inter-team communication that the extra load that feature represents could be assessed and capacity for the near term could planned for. Without visibility into that upcoming feature this type of analysis would have been wildly innacurate. This is because RTS alone is currently projected to make 30x the number of calls going to the proxies by everything else we are doing today. In the future capacity planning for all components needs to be handled in this way so that load can be addressed with a changing infrastructure and environment in mind.