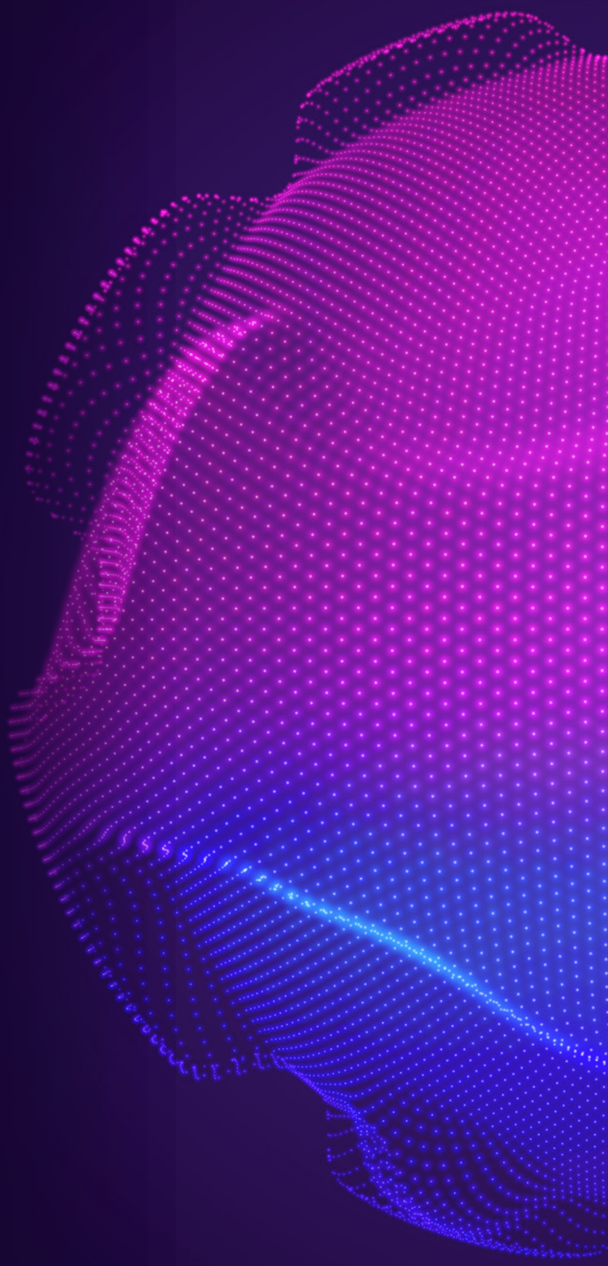
CDN Tuning for OTT - ''Why Doesn't It Already Do That?"

When you initially onboarded your OTT traffic to a CDN, you probably went with default settings. And to be honest, why wouldn't you? A standard media configuration is designed for the short http-based segment delivery at scale. It removes the bottleneck of your origin connectivity, taking you from a couple of hundred Mb/s to several Gb/s (and Tb/s) of OTT traffic to your customers as your service grows. So, do you understand what this standard configuration is doing, and how to fine tune it for your current use case?



What are you getting, out of the box?

A media configuration is designed specifically for chunk-based delivery over http or https.

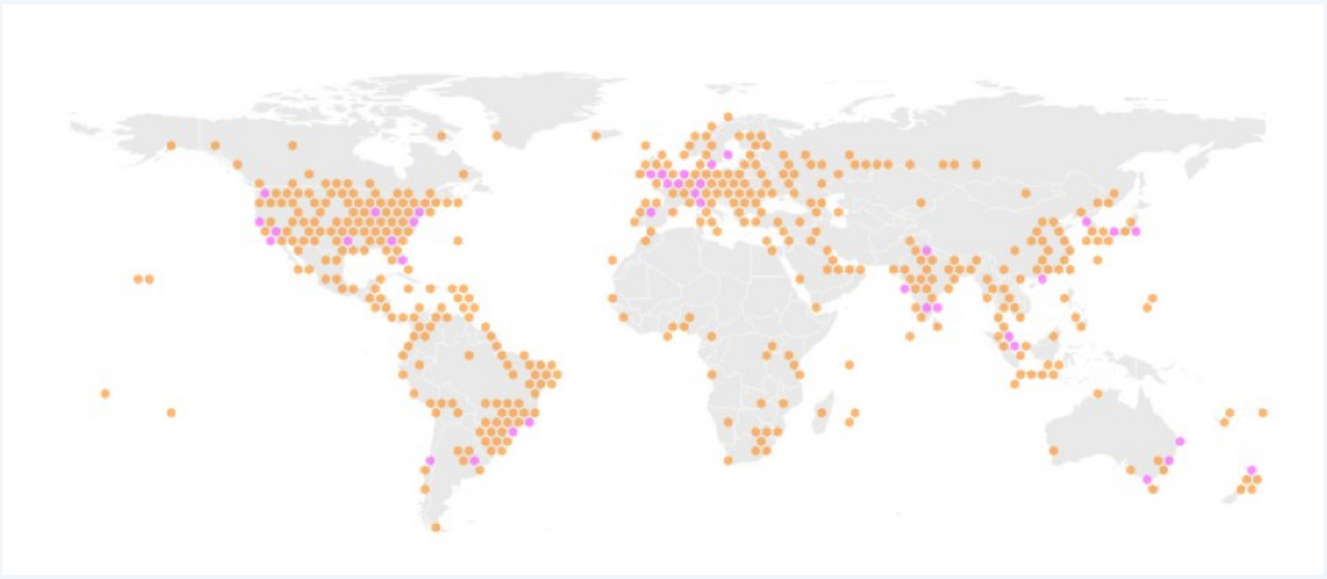
- **Custom TCP settings –** TCP traffic directly between a customer origin and client is usually quite sensitive to congestion — it's what TCP was designed for: fault tolerance (over an ever-congested public Internet). The number of TCP packets in flight for a given request slowly ramps up after the initial handshake. Whenever packets time out due to mid-route congestion, the number of packets per request drops by half immediately and slowly ramps up again. This improves reliability of data, but at the expense of throughput. The standard media configuration uses custom TCP settings specifically for adaptive streaming over http — persistent connections to reduce TCP connect times and make faster ramp-up and faster/more suitable TCP timeouts.

- **Improved routing back to origin –** when getting a route for a request from client to origin, Internet traffic paths generally default to the lowest number of hops (subject to network peering agreements). However, this isn't always the fastest route, as hop congestion isn't taken into account. Akamai media configurations will look at alternate routes to the next cache layer or origin and route the requests according to load and traffic, bypassing slow hops usually caused by congested peering between networks over the Internet. Sometimes the quickest way from A to B is via C!

- **Cache settings specific to streaming format –** the CDN media configuration isn't just a scaling extension to your origin. It's designed to default to best practice caching settings for the streaming format you are using, in order to populate edge caches efficiently and reduce traffic back to your origin, even if you don't configure these settings or content types on your own origin.

- **Tiered distribution –** all media configs come with tiered distribution. Regional edge caches with no cached or expired objects check with a common parent tier or edge peer before going back to origin, reducing offload. This tiered distribution will be regionally specific to your customer footprint.

- **Media map –** edge cache machines and parent tiers can be grouped into "maps". These maps determine how traffic is routed using traffic behaviour patterns. Media maps can be specific to geography — such as local (in-country) traffic distribution or worldwide traffic — or content type, such as short-tail or long-tail VoD, live streams, or even high-demand "event" traffic. Traffic over your media configuration will be delivered specifically using a media map rather than a generic http(s) web traffic map, which improves throughput and reliability tuned specifically for short chunk-based streaming. The machines allocated to some maps may favour memory read/write access while others may favour disk read speed, for example. Other maps may offer a balance between the two, or favour improved edge peer connectivity over improved parent connectivity. Ensuring you are on the right map for your content is always worth revisiting if your needs have changed since you launched the service.



So, all good so far. Why revisit it?

IIf you haven't revisited your media configuration for some time, you may see some changes to the self-service options. These are mainly about fine-tuning to be more specific to your requirements. By understanding where your origin is and your audience distribution is, even if you've moved to smaller fragments or higher resolution since launch, underlying settings can improve your performance. As an example, if you've attempted to reduce your latency by reducing your fragment length, then the TCP connect timeouts, retry settings and even peer request timeouts would work better if they were aligned with your fragment length. If you've tweaked your bitrate ladder, adding higher bitrates and larger resolutions may mean that your TCP settings need to be adjusted to provide the right performance. If you originally set your audience location to unknown or global, and are actually serving a regional audience, configuring this correctly will improve cacheability and performance. Even adjusting the popularity characteristics can improve performance by moving long-tail content to a lower cache turnover footprint so your content is less likely to be evicted from cache by other short-lived Internet traffic.

Now that you've fine tuned your configuration for a more appropriate audience footprint, selected the correct origin location, understood your content popularity and even adjusted to match your latest encoding ladder and fragment size, what more needs to be done? For most OTT customers, this out-of-the-box configuration with use-case-appropriate self-tuning will serve them well, even when they are serving hundreds of Gb/s of linear simulcast traffic. The CDN will handle scaling, improve throughput to origin in contrast to direct connections, mitigate DDOS attacks, and even provide additional features such as authentication or edge-based logic rules. The usual CDN functionality such as logic for origin health detection, origin failover and retry options, and dynamic construction of response objects such as manifests can be added in addition to the defaults through self service.

Now we can start to look at KPIs you've established as your service has matured and tune your delivery configuration around those KPIs. You don't need to be serving Tb/s of traffic to know that a percentage of traffic will still go back to your origin. Is your origin particularly sensitive to load spikes? Do you want to reduce your origin offload even further? Do you have a KPI around client buffering instances? One of the questions we usually get when we propose fine-tuning a configuration for a specific solution is "why doesn't it already do that?" Well, the answer can often be complex, but it boils down to balancing customer-specific business objectives that may not always be mutually compatible. Out of the box, a media configuration offers a balance of reliability, performance, and offload. First and foremost, it is designed to scale. When you start to tweak in favour of one of those objectives, it is often at the expense of one of the others, and it is ultimately understanding the business priorities of a service that allows us to provide the right solution.

One of the tuning tweaks we can introduce, for example, is mid-tier retry. The configuration constantly measures the response throughput for a forward object request to a parent. If it falls below certain throughput thresholds, it opens a simultaneous concurrent connection. The object will be delivered from whichever connection returns it first. It can even construct a large object response from partial byte-range requests via separate routes. This is good for mitigating mid-tier slowness, especially involving network providers peering over the public Internet, and improves KPIs based around buffering, but at the expense of additional traffic requests going to the origin.

If offload is your key objective (for example, you can't dynamically scale your origin farm, or you are sensitive to egress costs or demand spikes), we can look to see if the traffic distribution would benefit from the introduction of an additional cache tier or additional peer requests. Additional parent or grandparent tiers can provide more offload, especially where the audience is more widely distributed. However, additional tiers introduce additional hops — something you want to be avoiding if zero-buffer performance or ultra-low latency is your main KPI. The CDN can also smooth spikes at the edge by queuing requests whilst a forward object request is "in flight."

Popular items can push other objects out of edge cache storage during a demand lull. Sometimes cache eviction is unavoidable, even when an object is within TTL, due to the finite edge capacity. As CDNs continue to add edge capacity, it is nonetheless shared among other streaming traffic. Large customers with high traffic needs can even request custom maps, specifically targeting their audience footprint and origin location. You can even scale up to a managed or licensed CDN deployment that offers dedicated machines for your traffic. Even without a dedicated managed CDN (MCDN), understanding your content's popularity (long tail, etc.) can assist in selecting a delivery map where you have more chance of a cache hit and objects aren't competing with short-lived content for cache footprint.

Other adjustments — checking with non-obvious edge peers for objects before going forward to a cache parent, tweaking retry counts before taking any fail action, queuing requests for the same object or response code on the edge, manipulating cache keys, consolidating or spreading cache footprint, failing over to alternate maps, etc. — all have some degree of compromise. They will either favour performance, reliability, or offload, usually at the expense of the others. However, if your business objectives need weighting in one specific area, we can help give the config a nudge in the right direction, by engaging with Professional Services Consultants and ensuring we understand your KPIs.

Similar under-the-hood adjustments can be made if you are pushing the boundaries with either ultra-low latency or extreme VR/UHD bitrates and need specialist tuning such as object pre-fetching to maximise performance or to give you that buffer zone on reliability. When your OTT traffic levels start to hit Tb/s, even the smallest tweak — such as adding or reducing a second in TCP retry or timeout settings — can have a noticeable impact on overall performance and crucial operational KPIs that you report back to the business.

About Author

Del Fowler is an Enterprise Architect with 20+ years of Media Streaming experience, working with broadcast and blue-chip companies. His specialties are transcode engine design and OTT workflow optimizations. Del is a certified ScrumMaster and a streaming engineer by heart. During his 3-year tenure at Akamai he has participated in a number of notable projects such as major sporting events and customer consultancy on VR and low latency requirements.