Chapter 4 – Results

4.1.1 – Introduction

This chapter analyses the quantitative data gathered through the multitude of performance and qualitative tests carried out on the Video-On-Demand platform using JMeter for consistent and accurate results; the implementation was previously built during the third chapter. The findings listed are also compared to the past theoretical data of previous studies, with the purpose of determining similarities and comparisons between the findings of this research and of other studies which have been mentioned previously in the second chapter.

4.1.2 – Brief Overview

A fully functional streaming setup had been set up with the support of both HLS and dash support, both with the same optimisations implemented. The implementation also included the use of browser-based and desktop-based media players. Both hardware and storage setups were shared between the two streaming servers for the systems

As explained in chapter three, the tests were conducted include the performance comparison of HLS & DASH packaging and delivery. Additionally, the performance analyses of using centralised storage with RAM, compared to traditional hard-disk based storage. The use of non-variable bit rate streams compared to the variable multi bit-rate streams for both HLS and DASH packaging results and analytics were also gathered. Lastly, the performance benefit of the use of cache repackagers and IP based load balancing functionality for region-based cache server deployment were also analysed and tabulated for the end findings.

4.2 – Findings

4.2.1 - HLS vs DASH performance comparison

The information gathered from the tests showed that HLS and DASH packaging both have substantial similarities; however, the performance of HLS does defer from DASH even though both systems are based off a common technology. The first noticeable difference was the network traffic consumed by the stream. It was estimated that HLS took approximately double the theoretical bandwidth compared to DASH. The reason behind this is with regards to the video file chunk size, the recommended and default chunk size of HLS is that of 6 seconds while DASH packaging is based on 2-second chunk video files. Due to this DASH packaging sends smaller files at increased frequency rate while HLS would send fewer files to the client. However, these files would be larger and are sent less frequently, additional to this the longer chunk only provides enough time for the next segment to arrive thus running at a higher throughput than DASH.

Another performance measure was the server performance ratio and client performance ratio. It was calculated that HLS demands 16% to 19% less processing power than the counter-part; this measurement varied according to the efficiency of the player.

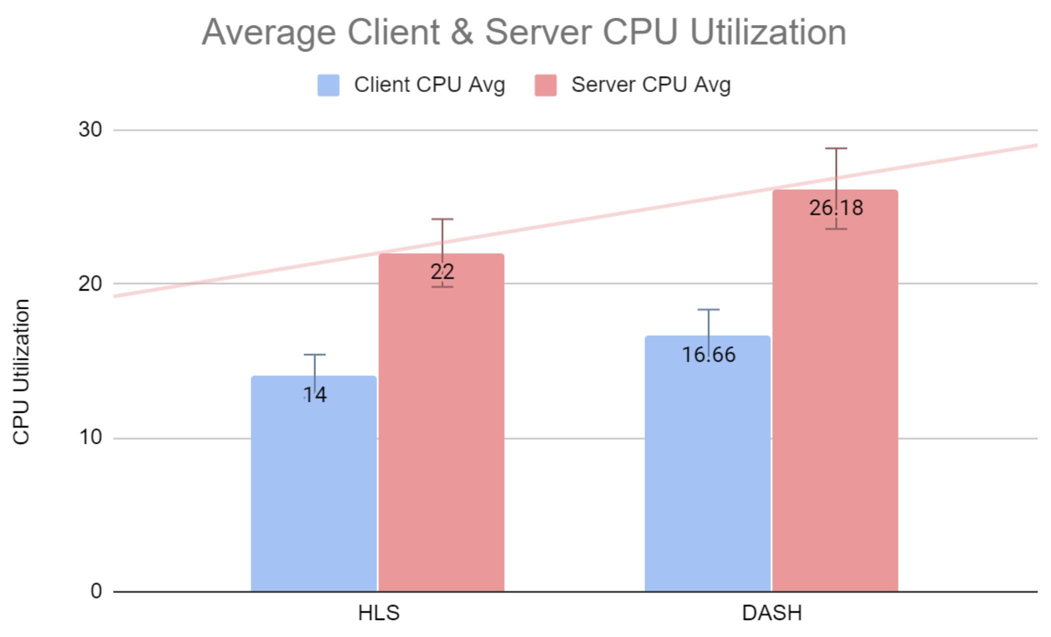

Figure 1 - Graph showing the comparison of HLS and DASH CPU utilisation on Client devices and VOD server. A trendline was asses as utilisation is dependent on concurrent user utilization.

The reasoning behind the processing power difference was observed to be due to HLS having a less complicated, playlist style index page, while DASH is built using a manifest for the delivery and indexing of video segments. The advanced manifest file requires more processing power for the server as well as the player to process and open. While being a minuscule performance hit, the additional processing required by the DASH manifest does scale-up effectively linearly according to the number of users requesting the video stream.

Tune-in times were also observed while gathering test results, this particular test for tune-in times proven difficult to analyse as they are very dependent on media player being used. This was observed that standalone media players experienced a 37% decrease in HLS tune in times compared to DASH. This is believed to be related to the DASH specific separate files which include Master Manifest – Video Segment – Audio Segment to be requested, delivered and parsed into one viewing experience. HLS, on the other hand, uses two files to deliver a stream, Media Playlist & Media Segment, the simplicity of the playlist, as well as the single media segment, allows for a faster tune in. Having said that on browser-based media players, the difference in tune-in speed is not so extreme, being that browsers have caches inbuilt. Thus, master manifests files and playlist files are effectively stored in their cache. Additionally, the few seconds between choosing the media to be played and playing the media allows for the manifest and playlist to be preloaded in the player.

Latency testing for both technologies was conducted, DASH included Master Playlist, Initial Audio Segment, Initial Video segment, Recurring Video Segments, Recurring Audio Segments. HLS latency testing included Media Playlist and Media Segments. A relatable result was concluded from the tests which go hand in hand with the performance results aforementioned earlier as the more complex DASH manifest file approximately tripled the latency traditionally experienced on HLS.

Lastly, quality of the stream was also observed, and the following result was derived; The quality of a non-adaptive stream is derived from how the stream is capable of not dropping frames since both streams are HTTP based and are both very similar technologies, quality is quite similar. However, an estimated 8% increase in dropped frames were observed on HLS. This was traced back to the previously mentioned default six-second chunk size, which requires additional bandwidth compared to DASH. Due to this overhead, service providers tend to lower the chunk size to match DASH for time-critical and high demand viewership like sports channels, however, doing so increases the risk of running out of buffered frames and would give the end-user a worse experience than experiencing dropped frames.

To combat both buffering as well as dropped frames, adaptive bit-rate streams where created so to continue to enhance the viewer experience.

4.2.2 - Adaptive Bit-rate vs Single Bit-rate comparison

Similar tests were run to evaluate and analyse performance counters with regards to variable bit-rate streaming. Once again, it was proven that HLS and DASH delivery are quite similar to each other, but both had a more vital performance outcome in various instances. When working with variable bit-rate, HLS creates an additional XML file called Master Playlist which contains the individual links for the different bit-rate formats of the video file as well as the Media Playlist with the segments for each bit-rate listed separately. Variable bit-rate DASH, on the other hand, uses the same number of files as used in available bit-rate DASH streaming. The single Master manifest file of DASH allows for a reduced transfer rate however has the disadvantage of having to process the more complex manifest as stated earlier. Thus, it was observed that latency had increased substantially due to the additional playlist needed and more complex manifest. The results propose that variable HLS has approximately twice as much latency than non-variable HLS and DASH has three times as much.

Tests for the server-side performance as well as client-side performance, when running both technologies in variable bit-rates proved that the result given was on par with the non-variable result. Additional to this, tune-in speed was affected, mainly when the network infrastructure was under load, as the possibility for the lower bit-rate allowed for a 47% decrease in tune-in speed for HLS and 66% decrease in tune-in speed for DASH respectively as the player preferred to use the lower bandwidth option.

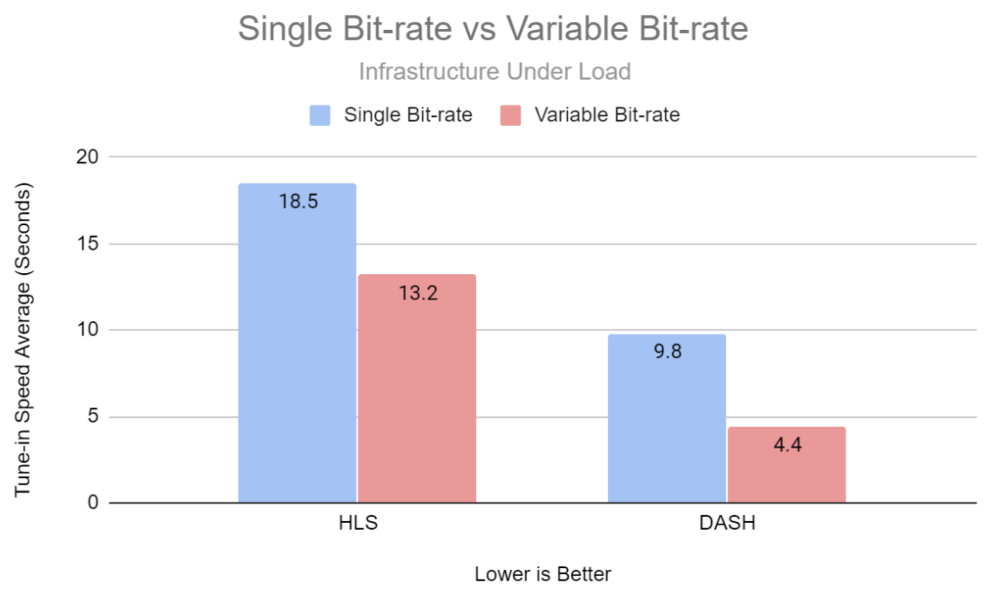

Figure 2 - Graph showing the comparison of HLS and DASH when running a single Bit-rate implementation compared to a Variable Bit-rate implementation.

However, when the network was not under significant load the tune-in times for both technologies took twice as long than having a single bit rate. Additionally, the use of variable bit-rate allowed for balancing the issue of dropped frames, video buffers and viewer satisfaction. The use of lower quality but the consistent stream is critically important when factoring in viewer satisfaction.

Lastly, even though both technologies proposed have similar qualities, they both process variable bit rate differently. This was observed by running a number of tests under the same circumstances. It was found that DASH variable bit-rate can actively swap between different bit-rates more coherently as well as tune-in faster and consistently than HLS. The two main reasons behind this are due to DASH's smaller chunk size, as HLS would have first to finish the six-second video chuck before it can try to stream at lower bit rate compared to the 2-second chunk size. Secondly, the structure of the Master playlist of HLS demands the player to first request the media playlist of the lower-bit rate before attempting to present it, unlike the DASH manifest which contains the URL format of every bit-rate thus can switch quickly and more efficiently. Tune-in time proved a similar experience as the player first tried to present a high bit rate, then requested the media playlist for a lower bit rate, then requested the media segment of the lower bit-rate and finally tried to present it to the user. This proved to increase tune-in compared to the DASH implementation as it can request a lower bit-rate video segment instantly.

4.2.3 - HDD based Storage vs Cache based storage

For VOD streaming, a centralised storage solution is essential for both simple but effective scalable storage as well as for the possibility of having multiple instances pulling directly from one set of data. The tests run compared the use of traditional Hard Disk Drive based storage and the use of Random-Access Memory Disk storage. The use of the RAM-Disk was being tested to be used as a buffer between the slower HDD-based storage and the VOD server. The results exported showed a minimal increase in responsiveness and negligible latency reduction. It was found that even though RAM-based disks are proven to be faster than traditional drives when accessing small amounts of content at a time, i.e. 2-second video segments, the RAM storage is being bottlenecked by the process to find the location of the video file and metadata in the first place. In light of this, cache logging was enabled directly on the VOD server to store both frequently accessed video metadata as well as frequently accessed video segments.

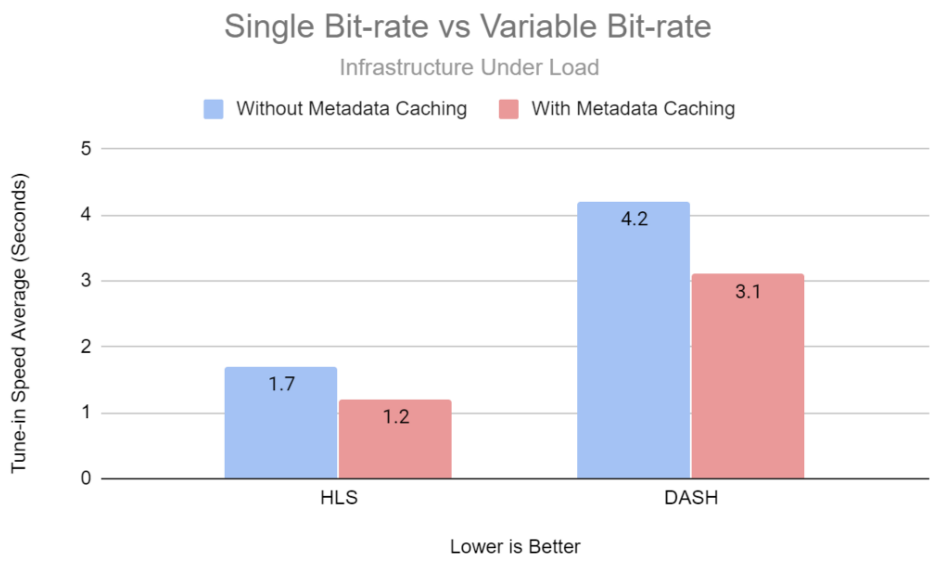

Figure 3 - Graph showing the comparison of HLS and DASH when running a Metadata and Manifest caching implementation compared to running without a Metadata and Manifest implementation.

The results gathered, proved that optimising frequently accessed data and storing for fast access enabled for it to be 26% faster than running the VOD packager without a high speed storage solution as it reduced both latency and tune-in times than storing all the video data in RAM-based storage, apart from the considerable cost overhead of deploying a separate server meant to hold terabytes of data in RAM Disks.

4.2.4 - Single Node vs Dispersed Caching Modules

Results gathered for this set of tests include the testing for the network performance between the main packager and the cache repackager as well as the latency and tune in times of when video data has been already invested in the cache server and during initial ingestion. When initiating an HLS or DASH stream request from a cache server, the cache server effectively checks if the requested segment is stored, if not it requests the stream from the central VOD server.

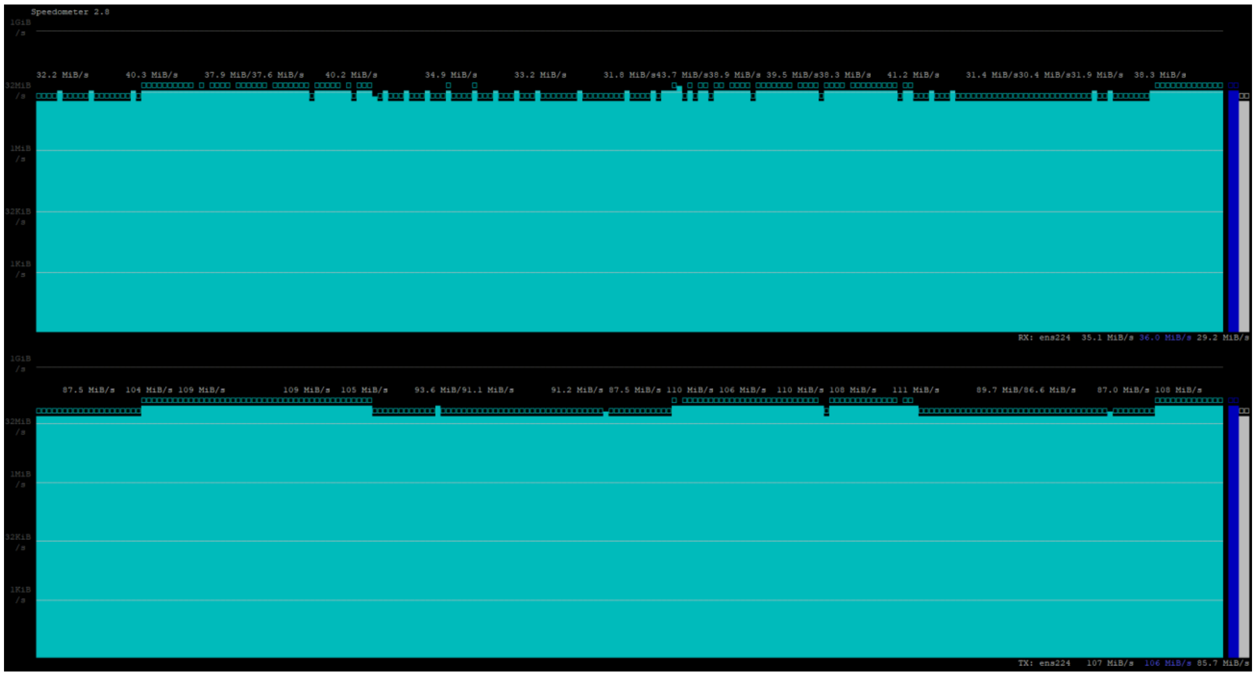

Figure 4 - Cache Server ingesting 36 MiB/s of data from the VOD server, storing it in cache and distributing it between clients at 106 MiB/s.

This allows the cache server to store the new data in its RAM-based cache and can serve multiple identical streams while receiving a single stream from the VOD server as shown in figure 50.

When the data is already stored and indexed in the cache server, it can successfully serve streams to a number of clients without communicating with the central VOD server as shown in the figure below.

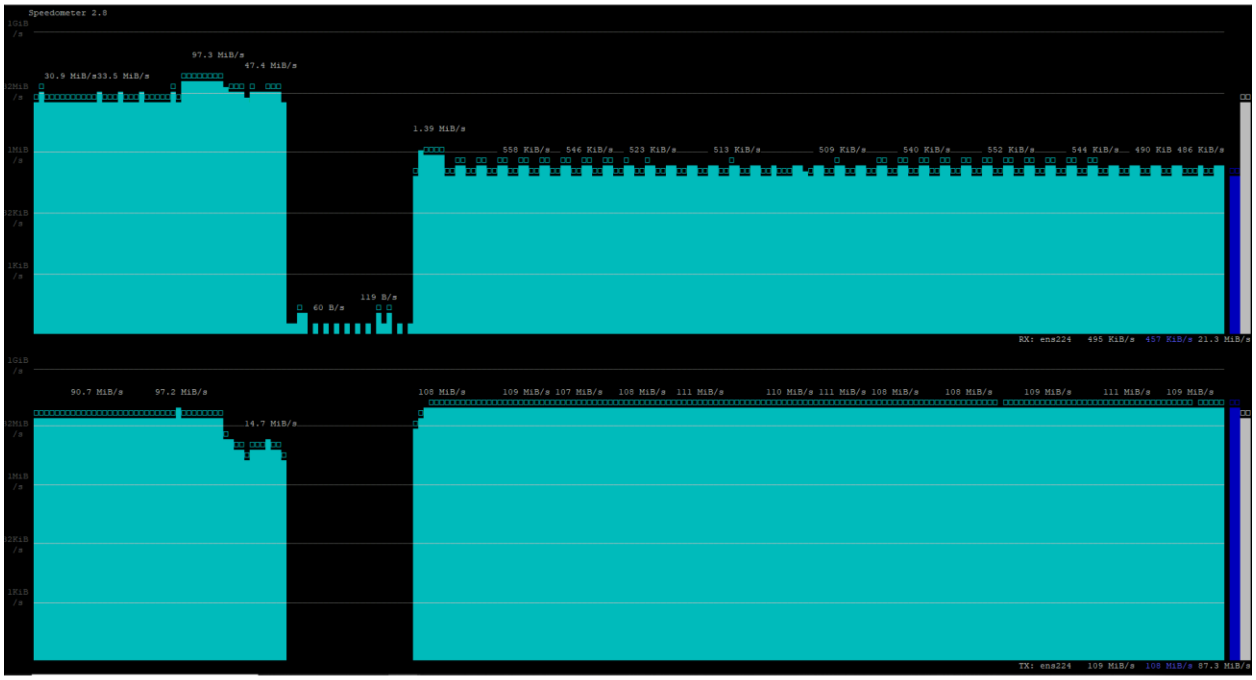

Figure 5 - Stopping VOD ingests and restarting media player of previously ingested file.

A cache server is receiving 457KiB/s of requests and distributing data at 108 MiB/s.

This allows for a number of benefits since cache modules are meant to be installed as close as possible to the end clients this reduces the amount of data which need to be sent across a multi-geographical network thus reducing latency and tune-in speed. Additionally, this allows for an efficient method to build redundancy as well as scale up the client capacity of the infrastructure by adding additional cache modules across towns or countries. The use of RAM-based cache in these modules also benefits from the metadata caching and segment caching, similar to what was found in the previous section.

Lastly, results show that when initiating a new stream with a cache server which is unknown, latency performance was reduced by 18% and tune-in speed by 8% because of the extra overhead processing done by the cache server. On the other hand, when requesting a stream which is already cached by the server, the latency performance was only degraded by 8% and the tune-in speed by 2%. The tests, however, had a significant reduction of dropped frames and buffering time, the reasoning behind this is that the 100 clients where now split between four different cache server and the additional transmission between the VOD server and the cache server had been significantly reduced.

Using cache modules, however, require a system to direct a client to them, this is discussed in the next section.

4.2.5 - Load Balanced Setup vs Point to Point

In this section, a load-balanced implementation was appetited to allow for region-based routing between the clients and the cache server. After successful implementation, test results showed negligible added latency and tune-in speed as well as no effect on dropped frames and buffering as clients where being routed to the allocated cache servers without being restricted by the load balancer. When using a point-to-point system, it proved that it reduced the number of clients it can handle and would add 6% of latency and reduce tune-in speed by 2%. Point-to-point infrastructures also make redundancy and scalability more challenging to implement. Additionally, by using a region-based load balanced setup, the client per region and clients handled went up by 40% with the only limitation for the maximum number of clients which can be handled based on the amount of caching modules deployed.

4.3 Chapter Conclusions

The gathered data brought to light various quantitive aspects which were sought by the researcher. The most important aspects that could be highlighted from the results found are further discussed in the upcoming final chapter. However, the below list shall be mentioned briefly for the purposes of this chapter’s conclusion:

- Results and data gathered were compared and contrasted respectively in order to find correlations amongst the data of already published work in order to confirm the success of the testing methodology.
- A brief overview was given at the start of the chapter so as to introduce the findings to the reader.
- Five themes were extracted from the quantitative data collected, which were discussed in great length and depth in order to satisfy the research aims and objectives.