Intelligent Caching In An ABR Multi-Format CDN World

Patrick Wright-Riley, Brian Tarbox
Motorola Mobility, Inc.

Abstract

In their infancy, content libraries contained a few thousand pieces of content and most vendors put a copy of everything everywhere. As the contents grew to tens of thousands of titles, Central Libraries were added and Least Recently Used (LRU), then intelligent caching, was employed. As content libraries have grown by orders of magnitude and now adding Adaptive Bit Rate / multi-format copies to the mix, some suggest intelligent caching is no longer possible. Motorola asserts that intelligent caching is both possible and even more critical today than ever. Intelligent caching still plays a valuable role in the ABR Multi-Format world.

INTRODUCTION

In the last ten years the industry has experienced at least three distinct generations of thinking on the approach to placement and duplication of content. We define the first generation as a time when content libraries were small enough that each Video On Demand (VOD) system maintained its own copy of each piece of content. These libraries were stored on spinning media and were served either directly via disc arrays or DRAM. These libraries tended to contain a few thousand titles of standard definition content. Caching in these systems was something that happened in the disk driver or the VOD server's memory backplane.

Generation two can be characterized by the slow introduction of high definition content and libraries of tens of thousands titles. This increase drove the capital expenditure equation high enough to discourage the placement of all content at every site. Thus, the Central Library approach containing the "Gold Copy" along with smaller edge libraries that maintained copies of the commonly viewed content being watched by subscribers within their domain was introduced. Many VOD systems were constructed with the 80/20 rule where it was assumed that 80% of the subscribers viewed the same 20% of content. Given this assumption, distributed edge libraries used a simple Least Recently Used (LRU) caching algorithm to determine which 20% of the content from the edge library was essential to maintain. As it turned out, this content distribution model did not produce the content storage and reduction in network congestion operators expected. This dilemma led to the development of an alternative approach called intelligent caching. Intelligent caching (IC) incorporated additional information about content viewing behaviors beyond what LRU could provide. From there IC became the norm for caching at the edge. However not so far down the road, the explosion in SD and more HD content storage requirements combined with a growing number of smart devices and tablets, Adaptive Bit Rate/Multi-Rate was destined to become part of the picture.

In the third generation, content libraries jumped again to hundreds of thousands of titles, with HD now dominating the content scene. Today this content is now chunked and replicated into several bit rates and wrapped in several formats. Thus, hundreds of thousands of titles can easily become millions or billions of file chunks linked by manifest files.

Conventional wisdom suggested that reaching these levels of processing and file

management rendered intelligent caching obsolete. It's suggested LRU caching within the content delivery network (CDN) is both the best that can be done and is enough. Given that intelligent caching increased the efficiency of edge content retention such that 98% of the content was properly retained, it seems reasonable to explore if those benefits can be retained in an ABR, multi-bit rate environment.

PROBLEM DEFINITION

What is Caching and What Drives It?

Caching is a predictive activity. When caching, the system uses data about past viewing behaviors to make assumptions about future viewing behavior associated with a given channel. The assumption typically results in the allocation of a scarce resource before it is actually needed. The "hit rate" of the cache is the percent of time that the assumption turns out to be correct. The impact of the hit rate is based on the comparative cost of permanently reserving the resource against the cost of allocating the resource on-the-fly. To achieve this second, more efficient method, caching is most powerful, especially given that in terms of network usage and related congestion, when the cost of real-time allocation is high.

In order to quantify the value of caching we have to look at the differential cost of resource allocation. Disks are substantially slower than memory and networks are substantially slower than disks. Some VOD vendors made a business out of this differential by attempting to build systems where the entire active portion of the content library lived in memory, or DRAM. This was a successful strategy until the growth of the library outpaced the growth in memory chip size. The battle then moved from DRAM vs. Disk to Disk vs. Library where the relative cost of late allocation was even higher.

Basic Caching

Caching algorithms are characterized by the predictive algorithms employed within the CDN. Caches are assumed to be filled with content at all times. Thus, the critical decision is actually which content item to remove from the edge storage. The most basic type of algorithm supporting this capability is the Least Recently Used or LRU. This algorithm maintains a usage timestamp for entries in the cache and when content removal is required will eject the item with the oldest usage time and replace it with new content. Such systems update the usage timestamp whenever there is a "hit" on the item. These algorithms can be compared to the psychological principal of "Win-Stay, Lose-Shift" where a successful outcome will cause a subject to make the same choice again and an unsuccessful outcome will cause the subject to make a different decision.

This leads to the need to understand the content viewing behavior attempting to be predicted. In the case of content viewership there are two approaches: attempting to predict the future behavior of a given viewer or the future behavior of a group of viewers. By tracking content watched by a particular viewer, inferences can be drawn regarding the potential viewing of that content by the set of all other viewers. However, when using an LRU algorithm to do so, the system simplifies the analysis to a single parameter—time last used—and may not be fully representative of the likelihood of future viewing by a group. Thus, although LRU has some value, it is greatly limited when compared to more intelligent, multi-parameter caching algorithms.

Comparison of LRU with Garbage Collection

Java is the world's most popular computer language and its performance is largely dictated by the behavior of its memory

reclamation or garbage collection system. The Java computer language's Garbage Collection (GC) system is one of the world's most studied caching systems. Valuable insights may be gleaned by comparing GC with various other caching algorithms.

One of the primary drawbacks of a simple LRU approach is that it understates or ignores the effect of what GC calls infant mortality of reference. Many objects have a usage model of initial creation followed by limited use, ending with no further activity. In a computer program a variable might be declared, used in a single computation and then discarded. Similarly, in a television viewing experience a user might tune to a channel, watch for a few seconds and then move on. In this scenario the content would have a very high LRU score. In essence, the naïve algorithm employed by LRU would preserve the item in cache in spite of its low actual usage. This confirms the low predictive strength of LRU. This is important when we consider that most CDN "intelligent caching" systems are based on LRU approaches.

To achieve greater predictive power, an algorithm must incorporate a more sophisticated object lifecycle model; an object being a piece of content or a chunk of content carrying specific bite rate and format characteristics. Such a lifecycle model is typically generational. The GC partitions the cache into three generations: 1) Eden, 2) Tenured, and 3) Permanent. When objects are first created they live in Eden. The system periodically scans the memory list looking for items to eject. Items in Eden that are not ejected after two passes, meaning they still have active references to them, are promoted to Tenured. Items living in Tenure that survive more passes are promoted to Permanent and thus remain much longer in cache. There are actually two types of GC passes: full and partial. Partial collections are run quite frequently, have relatively little impact on system throughput and do not examine the Permanent cache. Full collections on the other hand are comparatively rare, can often affect system throughput and do look at the Permanent cache. So, content that exists in the Permanent cache are only occasionally examined for ejection.

Segmented LRU cache uses a similar (though limited) system. There are two LRU lists. Items initially live on the first list and after a second "hit" get promoted to the second list. While this is certainly better than a simple LRU, there is still a world of difference between noticing a second hit and true intelligent caching.

Advantage of Intelligent Caching

Intelligent Caching is a term we reserve for systems incorporating a more sophisticated object usage model. Such a model must acknowledge the realities of content viewing such as channel surfing, free content preview, time of day and day of week viewership patterns and other patterns of apparent viewership that may or may not represent true viewing of content.

The bottom line is that content hits, initial or passive, are not predictive or representative of actual viewership until the aggregate viewing time has exceeded a certain quantum of time. Once the aggregated viewing time of the content has passed a threshold (which may be dynamic and involve multiple analytic parameters) then statistical inferences may be made about the future likelihood of additional views. This is the basis for intelligent caching algorithms and where their value lies above LRU algorithms.

Factors That MAY Diminish Predictability

If in a multi-bit rate, multi-format world where content is delivered over a CDN, one could argue that caching, in its entirety, is unnecessary. There are factors that are

commonly cited as evidence that caching is no longer possible or valuable. These may or may not eliminate the usefulness of all caching algorithms, but they certainly provide a challenge to the usefulness of some caching techniques. It is helpful to remember that the caching algorithm attempts to extrapolate from the past exposure or access of content the future possibility of that content being viewed again, possibly by another viewer. The problem in understanding and valuing cache is that "the same content" may now exist in multiple copies, in different formats and bit rates, with chunks spread across multiple edge streaming servers. (See section Content Affinity for more details about chunk distribution).

Multiple Formats

As we enter into the second half of 2012 the video format battle is raging on. Apple's HLS format appears to be dominant, but the Microsoft and Adobe formats are still contenders. Although it is unclear what position these latter companies are taking with respect to future support, they cannot be discounted. At the same time the DASH specification is evolving and may, over time, acquire significant share. While many hope to support fewer than four formats, that time is not yet here (and may never arrive).

There are at least two ways to address the question of how multiple formats effect cache predictability.

One way is to ask the question, "Does the fact that a piece of content was viewed in a particular format "enough" provide any evidence that it will be viewed again in the future…in that same format and/or in other formats?" Since the multi-format ABR world is so new it's hard to anticipate future usage patterns. To the extent that we can extrapolate from existing usage patterns it seems safe to assert that content reaching a threshold of use is in general more likely to receive future plays than content that has not reached the threshold. It is also reasonable, though untested, that reaching the threshold on a particular piece of content in one format is at least weak evidence for the future popularity of that content under a different format. To state the opposite one would have to assert that popularity in one format provided zero evidence of possible future popularity under another format which is unreasonable.

A second approach to this problem is to think about multistage packaging and common formats. As has been discussed in other papers, there is an ongoing debate of the merits of packaging in various locations. Some lobby the benefits of Central Packaging. Others point out the potential benefits of customization from Edge Packaging. An interesting hybrid approach is to perform an initial round of chunking and manifest creation in the center, followed by a real-time component that transwraps content and performs unique, targeted manifest generation. From a caching point of view this approach defers the combinatorics of multiple formats until well downstream of the CDN. A cache element located "upstream" of this real-time transwrapper might see just a single format, thus diminishing its value.

Multiple Bit Rates

The key to adaptive bit rate streaming is the availability of multiple representations of each piece of content. This can be seen from at least two points of view. On one hand the same content might well be viewed at a different bit rate on a phone, a tablet and a big-screen LCD based simply on the capability of the various devices. In this slice of the world each stream might have a different bit rate, but does not necessarily change its bit rate during the presentation. In the other slice, each client responds to the

ever-changing load on the network by asking for smaller content when the network is slow and larger content when the network is fast. This is the grand assumption behind most ABR streaming.

The problem is that it invokes the dilemma of the commons: when there is a shared and limited resource, the greater good is often different from the individual good. When the network is congested, every viewer will fully support the idea that everyone else should limit their bandwidth such that "I" can continue streaming the highest quality experience. And everyone else feels the same way. This can be controlled if the client software is controlled by the infrastructure providers in that their client software can enforce the self-limiting behavior. On the other hand, does anyone doubt that clients

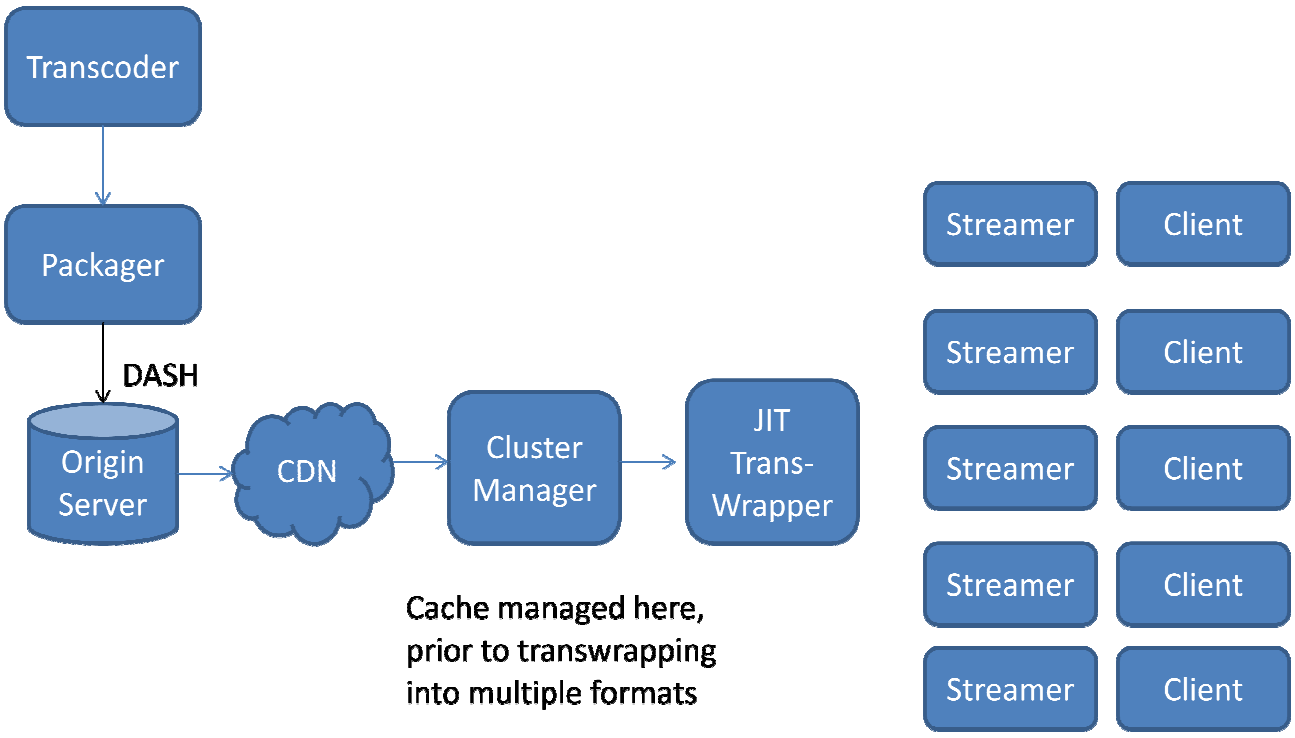

Figure 1: Possible Caching Prior to Transwrapping

will be made available that attempt to game the system to consume more than "their fair share" when the network is congested? We assert that it remains to be seen just how many distinct bit rates are actually active for a given content. So, while at first blush ABR might multiply the number of different copies of each piece of content by a factor of six to ten per format, the actual number may be significantly lower, perhaps three or four per format.

NDVR – Unique Copy

Unique copy basically eliminates the ability to do caching at all. For those unfamiliar with the concept, a legal ruling has

declared that if some number of viewers record the same content, the NDVR system must store a unique and distinct copy of that content for each of those viewers. In the systems, operators are explicitly forbidden to store a single copy and manage viewer access to that copy. So the only opportunity for re-use of segments or manifests would be if an individual user watched a recorded show multiple times—probably not sufficient to take advantage of caching.

While this might be seen as ending any discussion of caching, keep in mind that Unique Copy presents problems for many aspects of the system. It is anticipated that some vendors may push the envelope of mixed common copy / unique copy systems, especially outside North America. In this scenario, caching may have a larger role to play.

Personalization

Personalization is the process of converting a general video stream into one tailored for a particular viewer or group of viewers. Two main categories exist here; ad insertion and blackout (both are discussed in more detail in the paper "*Complexity Considerations for Centralized Packaging vs. Remote Packaging*" being presented at this conference.) In each case a stream that logically could be used to satisfy many stream requests is turned into one that is usable for a subset of those requests. To the degree that this personalization happens upstream of the caching system it will naturally render the caching system useless.

Factors That May Enhance Predictability

While many types of systems suffer from added scale, caching algorithms actually tend to work better in larger environment, if simply because there is more data to use for decision making and there is more content to provide a better opportunity to employ caching to enhance performance. There will undoubtedly be many different sized deployments of video systems, now and in the future. CDN-enabled, multi-format, multi-bit rate systems will be overwhelmingly biased towards the larger of these deployments; the cost of the complexity associated with such CDN systems precludes them from the smaller tier two and tier three deployments.

This then leads to the next important question which is, "Where does the caching engine live in the CDN architecture?" If it lives on the edge server, then it is limited to the total number of streams supported by that server. Many edge servers are relatively small devices supporting only a few thousand streams. The chances of getting meaningful hit rates in such a small environment are correspondingly low. On the other hand, if the caching engine lives near the edge, but in the CDN it might well be able to see dozens or hundreds of the edge servers. This scale changes everything. The chances of getting several play requests for a given content out of several hundred thousand streams is quite reasonable.

The Role of Content Affinity in CDN Caching

Most diagrams of ABR streaming show the client talking directly to an edge Packager or the CDN; the role of any edge server is not discussed. Motorola believes this is a mistake and causes large opportunities for caching via the use of Content Affinity to possibly be overlooked. If the diagrams do show an edge streamer they tend to show only a single one. In almost all cases any reasonably sized deployment will involve dozens or hundreds of edge streamers since each such device typically only supports a few thousand streams at a time.

Technical papers that have included a multiplicity of edge streamers have tended to view them as interchangeable, even on the per stream basis. It has been asserted that each chunk request from a client might be serviced from a different edge streamer, assuming that every edge streamer has the same chunks. This is then described as a resilient stateless design that can trivially survive the loss of one or more edge streamers. Some of that is true, but at a cost. The cost is that by making server selection stateless we remove the possibility of using knowledge from previous states to improve our caching.

Content Affinity is the process whereby all streams for the same content are directed to the same edge streamer. This can result in enormous savings in both disk space and network bandwidth utilization. If all streams for Spiderman, as an example, go to the same streamer, there is a far greater opportunity for fragment re-use than if the streams for Spiderman are distributed randomly to several dozen streamers.

If we accept the gains that can be realized from Content Affinity then we must look to see which deployment models give the best chance of using Affinity to our advantage. Figure 2 shows one such configuration.

The client makes its initial request to a Cluster Manager (CM) which is a control plane application that maintains the knowledge of which edge streamer has which content. The CM selects a streamer and issues an HTTP redirect message to that device. The client re-issues the request to the streamer which either services it directly if possible or defers to the Edge Packager to create the manifest, if necessary.

Note that Content Affinity is a separate concept from caching and the CM contains no storage of manifests or content chunks. The CM simply directs streaming requests in such a way as to increase the likelihood that the target Edge Streamer will already contain the required chunks for a stream.

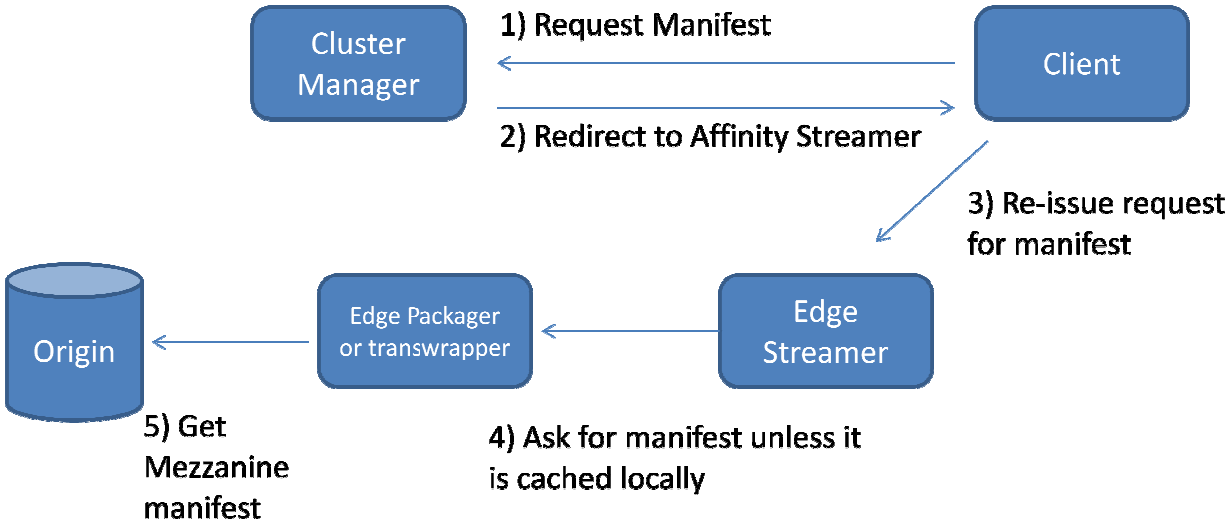

Figure 2: Content Affinity Deployment

Comparing Intelligent Caching with LRU Caching in a CDN

LRU-based caching in a CDN uses no intelligence about the content, its placement, or its usage. The algorithm simply notices which chunks were the least recently used and discards them when it determines that it needs to create space for new chunks. There is a single ordered list of the chunks logically maintained at the edge of the CDN. This single list covers all chunks sent to all edge streamers. It also makes no use of the fact that chunks may actually be related, i.e., being part of larger piece of content. This can be a benefit as well as a drawback.

If a viewer is channel surfing and briefly visits 20 different channels for 5 seconds each, then the system will likely generate the highest time-last-used values for those chunks and so they will remain stored in cache over other content that should be kept instead. A more intelligent system would never have promoted those chunks as they are clearly of transitory usage. On the positive side, since the system views each chunk individually it would not use those minor play times to promote later chunks from the same pieces of content.

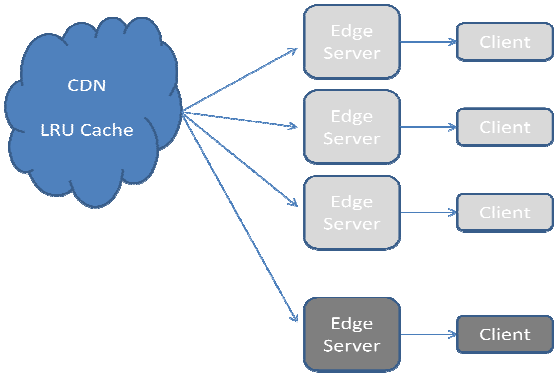

Figure 3: LRU-based Caching of Popular/UnPopular Content

An intelligent caching system would tend to treat such channel surfing as below the threshold for promotion within the cache and would thus not eject other, more popular content.

To put it another way, consider the case where three clients sampled a piece of content but ultimately were watching a different content and a fourth client sampled and then watched the content the others sampled. Putting aside the bit rate and format questions for a moment, we should objectively conclude that the program being watched by three viewers was more popular than the other content and should bias any limited resources such as caching towards the more popular program. The CDN/LRU-based cache cannot do that as it uses a strictly time-last-used algorithm rather than a hit counting-based algorithm. The Content Affinity-based system, on the other hand, allows for the direction of the common content to a common edge streamer and the one-off content to a different edge streamer. This automatically increases the locality of usage of each piece of content to a given pump and thus increases the hit rate of the particular pump's cache.

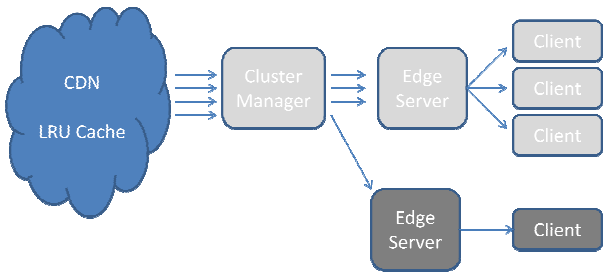

Figure 4: Affinity-based Caching of Popular/UnPopular Content

CONCLUSION

Historically, Intelligent Caching has been shown to provide significant reductions in the need for potentially expensive content storage. This benefit should not be discounted lightly. We have described several of the challenges facing intelligent caching in a multi-format ABR streaming environment. Some of these challenges such as the legal requirement for unique copy NDVR may prove insurmountable. We have, however, shown several opportunities that may allow the use of intelligent caching in other domains to have significant benefits over LRU caching. In particular, we have shown that the affects of Content Affinity can be profoundly and positively affected by efficient, intelligent caching algorithms.