Coordinated Caching in Planet-Scale CDNs: Analysis of Feasibility and Benefits

Kianoosh Mokhtarian
University of Toronto, ON, Canada

Hans-Arno Jacobsen
University of Toronto, ON, Canada

Abstract—Video Content Distribution Networks (CDNs) serve a significant fraction of the Internet traffic through a global network of cache servers. In a planet-scale CDN with millions of videos, cache servers only consider their own request patterns for managing their content. We analyze how, in the absence of cooperative caching, the knowledge of requests in remote serving locations can lead to better caching decisions overall and can reduce serving costs for the CDN. We call this practice cache coordination. Our analyses in this paper are based on actual video workload data from a global CDN. We analyze the spatial correlation of video popularities worldwide, the effectiveness and feasibility of cache coordination, and its scalability; from a city to across countries.

I. INTRODUCTION

Over half of the entire Internet traffic consists of video streams. YouTube alone is estimated to serve up to 30% of the Internet traffic across different continents [1]. Such substantial volumes are usually delivered to users through a network of geographically distributed cache servers called a Content Distribution Network (CDN). Figure 1 illustrates the layout of a generic CDN. Such a network consists of on-net servers located in the CDN network, e.g., in its datacenters or peering sites, as well as a large number of off-net servers located inside ISPs [2], [3]. For example, Akamai reports their presence in a growing number of 1,300 ISPs around the world with 170,000 servers as of 2015 [4].

The primary concern for a global CDN delivering voluminous video traffic is to avoid high traffic handling costs and overload at bottlenecks; in terms of latency, it is often just enough to keep the user-to-server RTT within a reasonable limit. For example, to serve the user traffic of an ISP, usually only a few of the server locations are preferred such as one inside that ISP or one behind a peering link to the ISP (cf. Figure 1). These restrictions are essential to voluminous content delivery in today’s Internet, and are the basis of our considered CDN model. Servers at different locations handle the traffic of different user networks (IP prefixes). They cannot freely cooperate in either serving each other’s users or redirecting users to each other based on, e.g., what individual video ID is requested [5], [6], [7], [8]—even assuming one manages to track the highly dynamic availability of millions of files on thousands of servers. Rather, in our considered CDN model, the traffic of user networks from different ISPs is assigned to server locations primarily based on traffic criteria: costs and constraints as the above, and RTT upperbounds. Then, the servers in each location manage their own cache contents based on the request traffic they each receive. This scalable model is known as the non-cooperative pull-based model in the literature [9], [10].

Therefore, each server maintains a dynamic collection of videos that are most popular for the user networks it serves. Instead of bringing in and caching all requested videos, a server may simply redirect a small fraction of requests (e.g., to an upstream, larger server location) for video pieces that are too unpopular to live on the disk (§ V). Given the several-Tbps volume of traffic, enhancing the efficiency of caching can save substantial traffic\(^\text{1}\). The paramount goal of the CDN is to serve as much traffic as possible locally by the preferred servers for each user network to make the service economically feasible. The success in achieving this goal depends largely on how effective the servers can manage their cached contents and let as little traffic as possible get past them: either cache-fill traffic into the server or traffic redirected and served by a less preferred location. This is the problem of our interest in this paper.

We explore the idea that a group of video CDN servers in different locations exchange request logs, e.g., what videos were recently requested and when. The knowledge of video requests in remote locations can lead to stronger popularity information and better caching decisions locally, e.g., by updating the local LRU state. This practice, which we refer to as cache coordination, does not interfere with the CDN’s traffic mapping criteria and allows the servers to still serve the same users.

\(^\text{1}\)While one might presume the cheap cost of disks as a perfect solution to this problem, the power-law distribution of video accesses [11] shows that just a few percent of higher caching efficiency requires an exponential increase in disk space, even if ignoring the constant growth of the data and traffic.
mapped to them based on these criteria. This is in contrast to *cooperative caching* between servers for collectively serving a wider user population [8], [12], [13], e.g., sending users in one ISP to off-net servers in another (competing) ISP, sending users to on-net servers with no peering path to the ISP, or servers proxying traffic from those in other ISPs or behind no peering link.

We use actual workload data from a large-scale, global video CDN to drive our analytical studies and trace-based experiments. We first investigate the key factor driving the success or failure of coordination: the correlation of requests across servers worldwide. Then, through extensive experimentation, we analyze the (dis)advantages of cache coordination, its performance tradeoffs, its relationship with workload correlations, and the forming of coordination groups across the CDN. We derive intuitive and non-intuitive findings throughout this paper, all backed by numerical validation, which can guide the design of different real-world content delivery solutions. Our contributions are as follows.

1) We conduct a detailed analysis of the spatial correlation of video popularities around the world. This is the first such study, to the best of our knowledge. Apart from our coordinated caching goal, this analysis is of value on its own and can benefit the design of different content delivery solutions such as server placement/provisioning and global traffic mapping.

2) We identify the practical considerations and challenges for real-world cache coordination, we build the proper mechanism and explore the following questions (§ VI).
   - Can a small group of neighboring, non-cooperative cache servers improve their efficiency by simply exchanging video request information? What are the overheads and their tradeoff with the achievable gains?
   - What is the role of the correlation of server workloads in the effectiveness of cache coordination?
   - Is cache coordination between arbitrary servers simply ineffective or can it be harmful (compared to just leaving each server in isolation)?

3) We investigate the possibility of expanding coordination to beyond neighboring servers, to country and across countries, in order to answer the following questions (§ VII).
   - Can servers that have coordinated locally and already strengthened their video popularity data gain further benefit by coordinating across wider scopes?
   - When does the increasing overhead of expanding cache coordination negate its diminishing gains? How can the right coordination clusters be formed across the CDN?

Both our popularity correlation and cache coordination analyses show that interests are strongly correlated within a country *even with linguistic diversity*. Correlation across countries can be of a broad range of values even between nearby countries, depending primarily on language. Moreover, we find that cache coordination in its smallest form, only between a few off-net CDN locations in the same *metro*\(^2\), can reduce the costly traffic getting past the servers by 3–11%: up to hundreds of Gbps of costly traffic—petabytes per day. On the other hand, coordination with unrelated servers even across a small group would fail: caching efficiency worsens compared to just leaving the servers alone. Extended coordination across an entire country is still effective even in those lingually divided—Canada and India are examined in that regard. Country-wide coordination can potentially increase the traffic saving to over 20% with no scalability issue. However, further extending coordination to across countries, even of the same language, yields a marginal saving (~1%) comparable to its overhead. Cache coordination across other countries such as neighboring but lingually unrelated is harmful by up to 11% loss of efficiency.

This paper is organized as follows. Section II reviews the related work. Section III introduces the datasets. Spatial popularity correlations are analyzed in Section IV. Section V reviews non-coordinated caching in a video CDN. In Section VI, we build our cache coordination mechanism and examine coordination between neighboring server locations. We analyze the scalability of cache coordination in Section VII. Section VIII presents concluding remarks and related open problems.

II. RELATED WORK

Caching content on a network of servers has been considered in various forms. Cooperative Web caching techniques for hierarchical or distributed caches are surveyed and analyzed in [12]. Performance gains of Web cache cooperation are analyzed in [13], [8]. For the specific case of CDNs, different methods have been proposed [9], such as having servers exchange digests of the content they can serve [14], report their contents to a centralized directory [15], or be selected to serve users (partially) based on content hashes [5], [6], [7].

These techniques can optimize metrics such as the first-byte latency or the collective hit rate but are not suitable for our considered CDN scale and traffic assignment criteria.

This is because prior approaches including the above have always considered cooperation of caches where a group of servers collectively serve a wider user population by either serving/proxying content on behalf of each other or redirecting users to each other based on the requested object ID—likely a more profitable approach than just cache coordination (only exchanging popularity data), if it was applicable. However, in our case a server has to serve only the user traffic that is assigned to itself based on traffic assignment policies and constraints, and the servers cannot arbitrarily proxy traffic through each other. Thus, to optimize the CDN’s caching performance beyond per-individual-server algorithms, we opt to have the servers coordinate their popularity data to improve the serving of *their own request traffic*. Note that coordination is between servers at different locations. Coordination among co-located servers is irrelevant given content sharding—distributing content IDs over the servers (§ VI).

\(^2\)We refer to a metropolitan area as a *metro*, e.g., the metro of Los Angeles includes the city of Los Angeles, Long Beach and Anaheim among others.
Content placement across a CDN has been studied in [16], [17], [18], [19], [20], among others. These approaches depend on global knowledge of per-location content popularity and availability and are therefore more suitable for CDNs in which maintaining such data and (logically) centralized content management is feasible. We target CDNs with highly dynamic request patterns, thousands of server location and millions to billions of videos. Unlike the complex and non-scalable methods for arranging video pieces on the servers—all to enforce that requests get the files always at their first point of landing—in our model we offload the task of content management to the servers and allow a small fraction of requests to be redirected. This way, each server eventually hosts what is most popular among users mapped to it. This model enables a CDN that is horizontally scalable with rapidly growing demand and allows full compliance with traffic assignment criteria.

Techniques for mapping users to CDN locations are investigated in [21], [22]. They assume the requested files are always available (or are cache-filled) at the selected server and better suit applications dealing with small pieces of data. A general content-independent mapping scheme is presented in [23] which is orthogonal to our work and can additionally be employed in the CDN model we consider.

Cache management has been analyzed extensively in the literature [24]. LRU is known as the most widely used scheme in Web caches, for its simplicity and effectiveness given the temporal locality of access patterns [25], [26]. We also use an adapted form of LRU. Variants of LRU such as Greedy Dual Size (GDS) [25] and GDS-Popularity [27] make it sensitive to factors such as variable object sizes and try to maximize the request hit rate. We deal with fixed-size chunks (§ V) and we care about the byte hit rate (not request hit rate), hence the size of cache-filled/evicted data is not of concern. Other LRU variants try to incorporate access frequency information such as LRU-K [28] and LNC-W3 [29]. Our workload demonstrates a long, heavy tail in the access frequency distribution. Aside from hot content that will stay in the caches anyway, the files on the borderline of caching, which comprise the vast majority of fetches and evictions, lie on this tail and are usually accessed very few times during their lifetime in the cache—too few to provide meaningful frequency information for distinguishing popularities. The Adaptive Replacement Cache (ARC) [30] distinguishes items accessed once and more than once and tries to adaptively partition the cache between the two. S4LRU [31] partitions the cache to 4 equal-sized segments represented by 4 equal sub-ranges of the full LRU queue; items are first inserted to the head of the last LRU quarter and get promoted to higher ones if accesses again. The above techniques and the like only address the classic problem of cache replacement. In an earlier work [32], we developed caching algorithms that meet the specific requirements of video CDN cache servers: a more involved version of the xLRU scheme used here (§ V) and a more complex algorithm called Cafe Cache. These algorithms are designed for efficient caching on individual servers such as managing the tradeoff between the ingress and redirected traffic, which is an orthogonal problem to the one studied in this paper.

Previous works also study video CDN workloads including the analysis of video popularities. A long-tail Zipfian distribution for access counts is reported in [11], [33], though only for a local workload at an edge network. We have observed the same distribution at both global and local scales which we do not detail in this paper to save space. The authors of [34], [35] examine global popularities but report a different tail pattern: Zipfian head but no long tail. This is because the global data used in these approaches is based on a crawl (e.g., through “related videos”), which gives a smaller chance to unpopular videos to appear in the data. Our data is uniform across all videos at both local and global scales. Regarding popularity correlations, the authors of [11], [33] confirm that video popularities are local, in that the global popularity of videos and the local popularity in a certain region exhibit no strong correlation; a description of the reasons underlying local interests can be found in [36]. Our analysis also confirms this result, which we do not repeat. However, an important class of information for content delivery is the similarities and differences of video access patterns from region to region across the world. Our detailed analysis of spatial popularity correlations is, to the best of our knowledge, the first of its kind in the literature, and its applications are not limited to the cache coordination goal of the present paper (§ IV-C).

III. Datasets

We base our analyses in this paper on two sampled datasets from a large-scale, global CDN serving user-generated video content. The datasets comprise anonymized request logs for a one-month period in 2013. Each entry includes the following: timestamp, the requested video ID (hashed), byte range, playback ID, and the request origin broken down by country, province, and city. Dataset I includes traces of the entire CDN worldwide, down-sampled with a large factor for the feasibility of the analyses and the global evaluations. This is done through random sampling by video ID to maintain the actual distribution of the workload over server locations and over user networks. The sampling ratio (N-to-1) is properly taken care of and made transparent in the evaluations, e.g., a 1 MB video piece is set to use N MB disk space and bandwidth. The results reported in the paper are at original scale. The N-to-1 sampled data includes over 2 billions of request entries for about 6 million distinct videos\(^3\). While a representative dataset at scale, we observed that Dataset I can be too noisy—of too coarse resolution—to serve for analyses at small scales such as individual off-net serving locations in local ISPs. We therefore do not use Dataset I for per-server evaluations; the consequent limitations of our analyses are discussed at the end of Section VII. On the other hand, to conduct a number of such detailed experiments with minimum noise, i.e., the samples more accurately representing the actual scale, we obtained

\(^3\)The precise video corpus size and serving capacity of the actual-scale CDN are not relevant to our analyses in this paper.
Dataset II for smaller-scale experiment cases. This dataset includes sampled traces of a seven server locations as described in Section VI-B and is collected with a much smaller sampling factor—the sampled traces of a server rack in this dataset are approximately equivalent to the actual-scale traces of one server.

IV. SPATIAL CORRELATION OF VIDEO POPULARITIES

In this section, we aim to identify the spatial correlation of workloads across server locations. An analysis of workloads along other dimensions such as temporal evolution of video popularities over their lifetime are outside the scope of our analysis in this paper and can be found elsewhere in the literature [37], [36]. Here, we investigate between which locations, and to what extent, the popularity of a video in one location may inform about its popularity in another.

A. Metrics and Popularity Classes

We measure the correlation of popularities between two workloads \(X\) and \(Y\). The popularity of a video in this analysis is measured as the number of its playbacks over the one month period of our traces. We plot each video as a data point on the \(X-Y\) plane based on its view count in workloads \(X\) and \(Y\). An example is plotted in Figure 2a in log scale for two nearby server locations which are two ISPs in a metro in North America. Figure 2c shows this for the workloads originated in two adjacent Spanish speaking countries. Note that given the long-tailed, Zipfian distribution of view counts, these charts in linear scale would have almost all data points concentrated near the origin (figure not shown). We also plot in Figures 2b and 2d the videos according to their ranks in the respective workload (zoomed). Unlike view count, rank is a metric uniformly distributed along the axes.

Fig. 2. Correlation of popularities between the workload of two servers in the same metro (Figures 2a and 2b) and that of two neighboring Spanish speaking countries (2c and 2d). Axes are normalized to \([1, 100]\).

We compute the Spearman correlation of popularities \([38]\), i.e., the linear correlation coefficient of video ranks. Unlike raw view counts, ranks have a uniform (unbiased) distribution suitable for correlation analysis. They are not sensitive to the two workloads having different distributions—different Zipfian skewness. The correlation between workloads \(X\) and \(Y\) is therefore calculated as follows, where \(r_X(v)\) is video \(v\)'s rank in \(X\) and \(R_X = \text{average}_v(r_X(v))\).

\[
 r = \frac{\sum (r_X(v) - R_X)(r_Y(v) - R_Y)}{\sqrt{\sum (r_X(v) - R_X)^2 \sum (r_Y(v) - R_Y)^2}}. \tag{1}
\]

Correlations for different popularity classes. In Figure 2 and the like for other cases, we observe a much stronger correlation for top videos while there is barely any correlation for the unpopular ones. This means, the interests become more local as we go deeper in the popularity list. Moreover, for videos on the long tail of the popularity curve, a small number of views over a month is too noisy signal to meaningfully distinguish the videos in terms of popularity, hence weaker correlations. Therefore, rather than a single correlation coefficient for two workloads, we quantify the correlation separately for different subsets of videos which we refer to as hot, popular, occasional and all videos. These groups correspond to the top 0.1% videos, top 1%, top 10% and the complete (100%) set. The correlation values are \([0.94, 0.90, 0.69, 0.10]\) and \([0.61, 0.61, 0.56, 0.03]\) for Figures 2b and Figures 2d, respectively.

B. Correlations

We would like to identify which groups of workloads are correlated and analyze our two hypotheses of geographic proximity and cultural similarities, assumed represented by language in our analysis. We first examine the level of correlation between server locations in the same metro, which are expected to exhibit the strongest correlation compared to other scales. We then widen our focus and examine correlations across provinces and countries. We refer to both provinces and states simply as provinces.

Intra-metro correlations. A correlation value close to 0 indicates no similarity and a value of 1 indicates identical workloads. However, given the inherent dynamics and noise in video view counts, we can only expect up to a certain level of correlation value (not exactly 1), even between closely similar workloads. As a basis for understanding these levels, we analyze intra-metro correlations: that of workloads of different
subsets of users coming from the same metro. These workloads correspond to the servers located in and serving the same metro but in different ISPs (where applicable). For each given metro in which more than one server location exists, we measure the pairwise correlation coefficient between all locations. We run this analysis for a representative set of 10 metros selected from all 6 continents of the world. Figure 3a plots the results broken down by video popularity class. Intra-metro values in Figure 3a are the strongest correlations to expect between any group of workloads across wider regions.

**Intra-country correlations between provinces.** In this and the following analyses, we partition the requests based on their origin location, not based on the server locations they landed on. The former demonstrates the natural correlation of local interests, whereas the latter is CDN specific—the CDN’s architecture, resources and constraints.

Interests within a country are often strongly correlated. Figure 3b plots the average pairwise correlation between U.S. states. The same analysis for a number of other countries across all continents demonstrates similar correlations: values between 0.71–0.90 and between 0.57–0.73 for the Popular and Occasional classes, respectively. For countries with major lingual diversity, however, one may expect more independent local interests. Our analysis shows this conjecture is true only to a small extent. Taking Canada as the first example, we measure the pairwise correlation between the top 5 provinces by population, four of which speak mainly English and the other one French (Quebec). Table I lists the obtained correlations. The values show that the expected drop from Set 1 to Set 2 can be expected between intra-metro and intra-country values examined here.

We do not also examine intra-province correlations across cities and metros. A statistically meaningful analysis of correlations requires a large set of data points (videos) and a significant number of views for each examined metro. Given our sampled data, this leaves in most cases only one or very few suitable candidates (metros) in a province. Nevertheless, intra-province correlation can be expected to our current analysis. Related works studying the source of popularities can be found in [39], [36].

**Worldwide correlations across countries.** We investigate which countries exhibit correlation and examine two intuitive hypotheses of geographic proximity and lingual relation. We compute the correlation coefficients between every pair of countries among a set of 80 selected countries around the world with largest workloads. Figure 4 depicts these correlations by the countries’ distance for the Popular class. The same plot for Hot and Occasional classes (figure not shown) shows the same pattern as Figure 4 with Y-axis values spanning wider and narrower, respectively. The distance on the X-axis is measured as the capitals’ great-circle distance, i.e., the distance on the Earth’s surface.

**Observation 1:** Popularity in nearby countries can be of any correlation (0.2–0.7) and correlated countries can be of any distance (0–20,000 km). Figure 4 helps us examine the effect of geographic proximity. While there is a slight decreasing trend between distance and correlation, we find that many of the strongly correlated, nearby countries are in fact correlated due to lingual closeness rather than geographic proximity (analyzed next). A few examples showing this differentiation are highlighted in Figure 4. Moreover, lingual closeness can give rise to strong popularity correlations regardless of geographic distance.

**Observation 2:** Language dominates geography. To better separate the effect of language and geographic proximity, we analyze correlations between lingually related but non-neighboring countries on the one hand, and between nearby but lingually different countries on the other hand. We make this comparison for two languages, English and Spanish, and for a lingually diverse neighborhood in West Europe. English and Spanish are the most suitable candidates as two languages spoken in several, distant countries, while also having sufficient video workload in our down-sampled traces. West Europe is the most suitable candidate as one neighborhood hosting several different languages and having enough workload. This comparison is shown in Table II.
Popularities across countries are lower and more diverse, where having different local languages, are still strongly correlated. Otherwise, the middle range, the part that observes the most diverse range of videos to exchange access information for, because: (i) servers cannot properly handle videos on the mid range of the popularity curve. The head which is hot content will automatically live on most servers using any algorithm and the far tail on none. In the meantime, the middle range, the popular and occasional classes, is the part that observes the most diverse range of correlation values. This correlation is strong within a metro and diminishes for wider scales. Moreover, video popularities across the provinces of a country, even in Canada and India having different local languages, are still strongly correlated. Popularities across countries are lower and more diverse, where the dominant factor is language irrespective of geography. Geographic proximity, on the other hand, can sometimes be neutral in shaping correlations. This information is particularly of interest for coordinated caching as it can forecast the effectiveness and guide the formation of coordination groups, as we analyze in the next sections.

In addition to cache coordination, the correlations analyzed here provide insights for the design and operation of other content delivery components. Notable examples include the assignment of worldwide user networks to server locations and the provisioning of them with servers/disks. For large serving sites assigned traffic from different regions, the caching performance is largely a factor of the diversity of the incoming request profile. The same server configuration and caching algorithm can yield entirely different hit rates depending on where the servers are deployed and the user networks assigned to them; whether in North America between the U.S. and Canada or in Europe; whether handling traffic from one or multiple countries. If there is no strong correlation between the assigned user networks, a serving site may need to be equipped with up to several times more servers/disks to achieve the same level of caching efficiency given the skewed workload distribution. Moreover, correlations can be taken into account as a secondary factor in mapping user networks to server locations; at the very least, where two or more equal-cost traffic assignment options exist. This avoids unrelated request profiles landing on the same servers and increases the overall caching efficiency.

The role of language and geographic proximity in popularity correlations. The numbers in each cell show the average, minimum and maximum correlation value across country pairs in each group.

<table>
<thead>
<tr>
<th></th>
<th>Hot</th>
<th>Popular</th>
<th>Occasional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spanish/English</td>
<td>0.57</td>
<td>0.54</td>
<td>0.60</td>
</tr>
<tr>
<td>(0.36–0.79)</td>
<td>(0.40–0.73)</td>
<td>(0.42–0.58)</td>
<td></td>
</tr>
<tr>
<td>Mixed neighborhood</td>
<td>0.43</td>
<td>0.42</td>
<td>0.40</td>
</tr>
<tr>
<td>(0.33–0.55)</td>
<td>(0.33–0.55)</td>
<td>(0.35–0.46)</td>
<td></td>
</tr>
<tr>
<td>Worldwide</td>
<td>0.30</td>
<td>0.35</td>
<td>0.35</td>
</tr>
<tr>
<td>(0.0–0.79)</td>
<td>(0.11–0.73)</td>
<td>(0.14–0.65)</td>
<td></td>
</tr>
</tbody>
</table>

This section reviews the caching model in our target CDNs (non-coordinated); see Section I for the general network model. We review the caching requirements, the xLRU scheme for addressing them, and our target performance metrics—all the necessary pieces to study cache coordination in the following sections.

The need for avoiding unpopular content. Each CDN server needs to host the most popular content for the corresponding request traffic. This normally requires redirecting requests for content too unpopular to live on the disk (HTTP 302), as detailed shortly—the server can sometimes act as a proxy too, without caching the data, but only if its egress is under utilized and its ingress not constrained. In addition to polling the disk, caching unpopular content can overload the server’s disk drives with excessive writes, given the large size of video objects and their heavily skewed distribution: a large fraction of videos observe only one or very few accesses during their lifetime in the cache before being replaced with other (possibly unpopular) content. The consequent writes overflow the disks and harm not only the write-incurring requests but also the regular read operations for other requests that are simply cache hits. We have observed that for every extra write operation, we lose 1.2–1.3 reads.

Partial-request model. Client players often download a video by issuing a series of byte-range requests. This allows fine-grained control over the player buffer. At the server, the first segments of a video receive the highest number of hits compared to the rest [11]. It is thus not efficient to cache-fill or evict video files in their entirety. To simplify the support for partial caching, a common practice that we also adopt is to divide the disk and the files into small chunks of fixed size $K$ (e.g., 2 MB). This is to eliminate the complexities and inefficiencies of continually allocating and de-allocating disk space to segments of arbitrary sizes. Both our non-coordinated and coordinated caching schemes operate at chunk level and the servers store only the popular pieces of large videos. Note that even with chunking, it is still a non-trivial decision for the server...
whether to cache-fill or redirect a byte-range request where the request has some chunks present in the cache and some missing. That is, although the server may store videos partially, it needs to either fully serve or fully redirect a requested byte range: clients can request different byte ranges at their own choice from different servers, but they do not also partially download a single byte range from multiple servers. This requirement is supported by the xLRU scheme.

The xLRU scheme. We base our analyses on the xLRU algorithm for non-coordinated caching on each server, although our coordination mechanism is orthogonal to the underlying algorithm and is similarly applicable to other algorithms. In [40], we quantitatively analyze the rationale behind the choice of xLRU over the alternative algorithm Cafe [32] and also over other caching methods from previous works, even imagining they were applicable.

The xLRU scheme operates based on two LRU queues. First, an LRU disk cache stores partial video files as fixed-size chunks with an LRU replacement policy. To minimize the cache-fill traffic and avoid too many redirects at the same time, it is best for the server to keep only the most popular videos on disk (from the server’s perspective). Thus, the second LRU queue, a video popularity tracker on top of the disk cache, tracks the popularity of each video file as its last access time—how recently a chunk of the file was requested. The popularity tracking algorithm shares similarities with the LRU-2 algorithm [28]: if there is no previous request for the video, it fails the popularity test and is redirected. It is also redirected if the age of the last request for the video is older than the age of the oldest chunk on disk, also known as the cache age. Otherwise, the request is sent to the disk cache for serving, including the cache-fill of any missing chunks. Out of a series of byte-range requests in a playback, which are not necessarily aligned with chunk boundaries, we do the popularity test and popularity update only on the first request of a playback and the decision is remembered for all other requests carrying the same playback ID. This is to avoid that possibly multiple partial hits of a chunk in the same playback mislead the popularity tracking algorithm. The complexity of all operations of the disk cache and the popularity tracker is $O(1)$. This is by implementing both components using a linked list maintaining access times in sorted order and a hash table that maps keys to list entries.

Redirections. If deciding to redirect a request, the destination server is selected by the CDN similarly to the way the initial server for each request is selected: according to a mapping of worldwide user networks to server locations based on traffic constraints and costs. For instance, one can employ a secondary map which defines the destination of redirected requests from each user network. The detailed making of these decisions is outside the scope of the current work, nevertheless, the redirection choice usually includes a higher level, larger serving site in a cache hierarchy that captures redirections (and cache-fill requests) of its downstream servers, besides possibly serving some user networks of its own. The redirection choice may also include a sibling location which also peers with the user network(s) that the initial location serves. Note that the mapping decision in selecting the server location to redirect to, similar to selecting the initial server for the request, is based on traffic factors rather than the individual video ID requested (§I), e.g., no redirection to off-net servers in other ISPs.

The metric of interest: cost saving. Throughout the experiments in this paper, we monitor the amount of undesired traffic which is what gets past each server, i.e., the cost for the CDN. Suppose a server is receiving requests for $D$ Mbps of demand traffic. Some of the requested videos already exist in the cache, some are missing and should be cache-filled, and some are missing but are too unpopular to be brought in and replace an existing item, thus redirected. The server is serving $E$ Mbps of traffic, $I$ of which is through cache-fill. That is, a volume of $E - I$ is served directly from the cache. A portion of the requests gets redirected from the server to be served by another location as described above, which makes up a volume of $R$. Notice that $D = E + R$. Also, $I + R$ is the traffic that gets past the cache. One can define the cache efficiency metric as $(E - I)/(E + R)$, i.e., the standard cache hit ratio extended to incorporate redirections. However, in our case, we are primarily interested in the inefficiency of the servers, which directly reflects a cost for the CDN. Note that traffic served to users directly from the server’s cache $(E - I)$ is almost free since the cost for space and power is small compared to that of bulky traffic; this cost can also be normalized out since it is the same whether serving from disk, serving by cache-fill or serving from a different location of the CDN. However, requests served by cache-fill or redirection $(I + R)$ generate undesired traffic. In the case of off-net CDN servers, this incurs undesired traffic on the ISP’s uplink. In case of on-net locations, this incurs either traffic over the CDN’s backbone network, which requires provisioning, or traffic between a user network and a non-preferred server location, possibly traversing constrained or expensive Internet paths.

We measure $I + R$ as the traffic that directly concerns the CDN and we monitor its relative variations in our experiments. We refer to this metric simply as costly traffic; although, note that the cost referred to is not always directly monetary, as explained above. To illustrate, suppose CDN servers each receiving request for 1 Gbps of traffic with $(E, I, R) = (800, 200, 200)$ Mbps, i.e., 400 Mbps traffic gets past each server. Suppose an alternative cache management scheme, such as coordination, reduces $I$ and $R$, and it improves the servers’ operating point to $(E, I, R) = (810, 170, 190)$, i.e., 360 Mbps cost. In this case, the cost-incurred traffic for the CDN observes a 10% reduction.

VI. LOCAL CACHE COORDINATION

This section presents and analyzes a cache coordination mechanism on top of the xLRU scheme. The idea underlying cache coordination is to (ideally) have each server of a coordinating group see requests arriving at all others: what videos and chunks have been requested. This practice provides stronger popularity information for each server. It allows to admit and cache those videos that are more likely to be popular and evict those more likely unpopular, i.e.,
cache admission and replacement decisions when handling requests; the breakdown is analyzed at the end of this section.

We analyze the different technical considerations and implementation challenges for real-world cache coordination, we explore the design space, and we build the proper coordination mechanism in Section VI-A. We analyze the gains and tradeoffs of coordinating neighboring server locations in a metro in Section VI-B.

A. The Coordination Mechanism

We refer to each group of coordinating servers as a coordination cluster, which can be as small as a single metro or as expansive as across countries. We analyze the right choice of coordination clusters in Section VII. Coordination clusters are non-overlapping, i.e., there is a transitive relation between coordinators. The sophistication of non-transitive coordination across partly overlapping clusters is unnecessary (§ VII).

Syncing the state data. Given the (intentional) simplicity of xLRU, coordination of a group of xLRU caches is a rather straightforward task to implement. Coordination is done by servers exchanging their cache state data periodically—specifically, the part of the state that changed since last time. Thus, the extra functionality added to the servers includes maintaining a list of videos accessed since the last data exchange and a list of their accessed chunks. The former is to be shared with the popularity tracker of other servers and the latter with their disk caches (§ V). We analyze the size of this data and its tradeoff with the exchange period shortly.

The popularity data exchanged between servers includes \(<videoID, age>\) and \(<chunkID, age>\) entries. Age refers to the time since the video/chunk was last touched and is unaffected by the servers’ clocks being out of sync, unlike absolute timestamps. Upon receiving this data, a server reorders the entries in its two LRU queues if some videos/chunks appear fresher in the received data compared to their organic access time at the server. For either of the popularity tracker and the disk cache, denote the number of received entries by \(M\) and the items in the LRU queue by \(N\) (typically \(M < N\)). Assuming an LRU implementation with a hash table and a linked list sorted in recency order (§ V), this operation takes a total of \(O(M)\) time and is independent of \(N\). This is because all the new entries from the received (sorted) data are inserted near the head, less than \(M\) iterations into the local list. This means that the critical data structures do not need to be thread-locked for long for importing the data.

Data aggregation. The popularity data arriving at a location from the others includes plenty of redundant data: popular videos appear in almost every report received, as the coordinating locations have correlated interest profiles. Among the many appearances, only the one with the smallest age is taken and the others are discarded. To avoid the redundancy, coordination in a cluster is managed by a coordination master, a role handled by one of the servers of the cluster. Note that alternatively pushing this functionality to one dedicated entity on behalf of all coordination clusters, while a cleaner design and simpler to maintain w.r.t. availability, negates the purpose of keeping traffic local (for local coordination). Coordination in the cluster is carried out based on a pull model, where the master periodically queries all coordinating locations for their recent popularity data. This eliminates the asynchrony that might happen if each coordinating server was to initiate a data push instead, and it also simplifies configuring the coordination system. Once pulled everyone’s data, the master aggregates the data by taking the minimum-age value for each video and chunk and pushes the aggregate back to the locations.

A global coordination configuration defines the coordination clusters and the master for each cluster; there is no need for a master election procedure between the coordinating locations. The formation of coordination clusters worldwide is discussed at the end of Section VII. The coordination master is specified at rack/location level rather than an individual server and is ensured to be always up as described shortly.

The (dynamic) sharding problem. A critical part in the cache coordination system is to isolate the data sent to each server to only the content ID space relevant to it. The content ID space of a server is a function of the number of racks \((N_r)\) and the number of servers per rack \((N_s)\) in that server location. Specifically, to avoid content duplicates in the same location and increase the depth of the caches, a common practice is to shard content IDs over the racks through a hash function, e.g., simply \(hash(id) \mod N_s\) or a more complex scheme such as consistent hashing [41], and then similarly over the \(N_r\) servers of a rack.

The number of racks per location and servers per rack may vary from location to location. That is, there is no one-to-one mapping between the servers of different locations in terms of the covered content ID space. An example is illustrated in Figure 5. Moreover, the sharding is not static: upon failure of a server, one or all other servers of the rack take over its role, i.e., its IP addresses and the corresponding content ID buckets,

![Fig. 5. Coordination between 3 locations. The numbers show the sharding of the content ID space over the servers in each location. This space is illustrated as 12 buckets in the figure (i.e., \(hash(id) \mod 12\)). The bottom node in Location A Rack 2 is the coordination master.](image-url)
until the server comes back up. This is normally handled within the rack, through a mechanism such as electing in a failure-safe manner one of the servers as the rack’s health master that monitors and assigns shards to other servers and itself. This way, the rest of the CDN system only tracks whole racks, not individual servers. The failure handling information is not necessarily exposed to outside the rack. This means that the sharding of content IDs over the servers is not always known to other coordinating locations or the coordination master.

Flexible-shard subscriptions. One may address the sharding issue by multiplexing and demultiplexing the popularity data over sibling servers in a rack and the co-located racks in a location, through a local master in each location that is aware of the local sharding configuration. However, we opt for a simpler mechanism where the servers exchange data with the coordination master directly, as shown by the dashed lines in Figure 5. To eliminate redundant data, each server advertises the content IDs of its interest to the coordination master—piggybacked to its popularity data pulled by the master. The ID space is defined through the hash functions (buckets) that result in a content ID landing on that server. For example, in Figure 5, \( \{2/3, \{2, 4\}/4 \} \) represent the advertisement of the bottom server of Rack 2 in Location B: content IDs that fall in the 2nd bucket (out of 3) in the inter-rack sharding hash function and fall in either the 2nd or 4th bucket (out of 4) in the inter-server function.

Coordination master availability. Server failures are common events and occur in both short and long terms. While a transient failure/restart of the master may have no noticeable effect on cache coordination, a longer lasting failure is equivalent to the servers reverting back to non-coordinated mode: a considerable increase in costly traffic as evaluated shortly. We ensure the availability of the master of each cluster by taking advantage of the health monitoring procedure already running in each rack. That is, we pick one of the available racks in each coordination cluster at random and specify one of the IP addresses handled by the selected rack (e.g., always the highest) as the coordination master; note that rack availability is information known CDN-wide whereas server availability is not necessarily\(^6\). This ensures that one and only one server picks up this role in the cluster even if the original server behind that IP address goes down. That is, there is no need to monitor and actively switch the master upon failures. Also, the configuration data does not need to be re-synchronized over the cluster upon every server failure and change of the master. Rather, the same code that is running on all the servers, periodically invokes the coordination component, e.g., every 10 minutes. This component proceeds with issuing a pull request to all IP addresses of the participant racks/locations only if it finds that the master IP of the cluster currently belongs to itself.

B. Experimental Results

We use the traces of Dataset II to analyze the gains of coordinating neighboring servers; recall that Dataset I does not allow server-level experiments. Dataset II includes the traces of servers in five metros that have multiple server locations in different ISPs. These metros are located in Asia, Europe, North and South America, and Australia to provide a diverse and representative set. The dataset also includes two small sets of servers in unrelated locations which are used for one of the analyses in this section.

We feed the traces to the xLRU servers simulated for each cluster. We run these experiments in both non-coordinated and coordinated mode. The servers operate with their default configuration. Our goal is to find out whether cache coordination yields noticeable benefit in the first place, what range its overhead varies in, and to what extent its success depends on the correlation of popularities. The metric of our interest is the saving in the amount of traffic that gets past the servers, a.k.a., costly traffic (§V). The different cache servers take from hours to a few days to warm up. We measure their input and output

\(^6\)One may alternatively conduct location-wide health monitoring over the servers of multiple racks, presenting a whole location as a large server to the rest of the CDN, i.e., individual rack and server availabilities are not known globally. Coordination configuration, however, is unaffected by such difference of granularity, since in the latter case the health system ensures the availability of some server behind every valid IP location-wide, allowing the system to specify the master at location level rather than rack level.
traffic in the second half of the evaluated period where they all have reached their steady state.

Figure 6 summarizes the gains and overheads of intra-metro coordination. Coordination only within a metro yields 3–11% saving in cost-incurred traffic: a significant volume CDN-wide of up to hundreds of Gbps on unwanted/costly paths. The cost saving depends on the workload correlations and the number of coordinating server locations (3 to 10). In this experiment, the metro corresponding to Cluster 2 is serving a more localized set of videos, hence higher correlations (quantified shortly), and has the highest number of server locations. Figure 6 also shows the impact of the data exchange period. On the one hand, by increasing this period by \(K\) times, we increase the size of each popularity message, though at a ratio smaller than \(K\). Because the number of distinct videos and chunks watched in a 10 minute window is not as large as 10 times the same number in a 1 minute window. We plot in Figure 6a the size of the messages as a function of the exchange period for our five clusters, which confirms a slight sub-linear behavior. Each message behind this figure contains the popularity data aggregated and compressed by the coordination master (covering the whole content ID space) to be pushed back to all coordinating locations, hence called one-way overhead in the figure; the data uploaded by each location is also upper bounded by the numbers in Figure 6a. Also, increasing the exchange period yields less frequent data transfers, hence a smaller amortized data rate as plotted in Figure 6b.

On the other hand, increasing the period also means the servers catch up with each other’s popularity data with a larger delay—10, 60, 600, 1800 and 3600 sec in our experiments. Yet, as shown in Figure 6c, such delays have only a minor impact on the effectiveness of cache coordination. For example, increasing the exchange period from 10 seconds to 10 minutes and 1 hour, results on average in a 0.1% and 0.4% reduction in traffic saving, respectively.

The results in Figure 6 suggest that frequent exchanges of popularity data is unnecessary. To also avoid the small loss (up to 0.6%) of effectiveness due to large exchange periods such as 1 hour—note that this number still translates into an amount of traffic on the order of 1 to 100 Mbps in the limited experiments above—an exchange period of 1 to 10 minutes appears as a proper balance. While the data exchange rate shown in Figure 6b is negligibly small in all cases (< 120 kbps), the messages may be sent as a whole rather than slowly over the period. A period of 1 to 10 min keeps the size of these messages small as well: one message of average size ~0.5 MB every minute or 2–4 MB every 10 minutes. We have also analyzed the instantaneous gains and overheads during peak and off-peak hours of the day (figure not shown). The results show that the peak-hour message size is up to 1.8 times the daily average, while the traffic saving in peak hours is also between 1.4 to 1.7 times the daily average. This keeps the above numbers still within the same range, e.g., <1 MB one-way overhead every minute. The costly traffic saved by this coordination is 1.6 Gbps.

We also would like to examine how important the correlation of popularities is to the effectiveness of coordinated caching. This analysis is not as straightforward, since the effectiveness is shaped by a compound of factors such as the number of coordinating locations and the workload volume of each, besides the popularity correlations. Also, the correlation is defined between the workload of only two locations. Nevertheless, in this experiment we try to isolate the effect of only the popularity correlation value by hand-picking 3 to 4 locations in each cluster (metro) such that they have comparable loads and nearly equal pairwise correlations. We conduct this experiment on four metro-based coordination clusters as well as two sample clusters (intentionally) comprising servers across different countries: one in Europe from countries speaking different languages and one in South America from Spanish speaking countries. All considered server locations are off-net and serve local traffic.

Figure 7 shows the result for each cluster as a data point on the plot. The results based on Popular and Occasional-class correlations are shown separately, and the exchange period is 10 minutes. The four highest savings in each figure belong to the four metros. The case with negative saving belongs to the European cluster. It shows that coordination fails across unrelated servers: it is harmful compared to just leaving the servers non-coordinated. The case with 1.2% saving belongs to the South American (Spanish) cluster. Figure 7 shows that coordination gain is a direct factor of popularity correlation. Note that we do not derive a fixed, one-size-fits-all linear formula between the two, given the contending factors in the real world as discussed. Though, in general our different experiments (at different scales) show that coordination between three or more locations with >0.5 workload correlation in the Occasional class yields non-negligible gains and those with <0.4 is ineffective or even counter-productive.

Cache coordination enables both better cache admission and better cache eviction. We analyze the breakdown of the gain between these two components to get a better understanding of the underpinnings of coordination gain—also to examine whether we can obtain most of the gain with only one component. In two separate experiments, we limit coordination to only the popularity manager and only the disk cache. In the metro with the highest saving (11%) in the previous experiment, we find that the obtained savings by only popularity manager and only disk cache coordination are respectively 7.5% and 5.3% (plot not shown). Breakdowns in a similar range have...
been observed in other cases. This suggests that coordination in both popularity manager and disk cache are necessary to realize the highest saving.

VII. EXTENDING THE COORDINATION DOMAIN

We observed that neighboring locations in a metro can coordinate their state and reduce their costly traffic with minimal communication overhead. In this section, we examine whether such coordination is also beneficial at wider scales, such as across a country or between countries. Such extensions have a number of consequences on the expected gain. First, popularity profiles are not as strongly correlated across larger areas (§ IV-B). Second, the extension increases the communication overhead in both the size of the aggregated data and the number of its recipients. Third, coordination in smaller domains already strengthens the popularity state maintained at the servers, casting doubt on how further helpful another layer of (wider) coordination with its consequent overheads could be.

We analyze scaling up the coordination domain in two levels: between provinces of a country and across countries. We conduct this analysis using Dataset I and hypothetical server locations designated for the respective coordination units. For example, to study inter-province coordination within a country, we imagine one custom-provisioned server location for handling the request traffic of each province of that country. This is to isolate the effect of only inter-province coordination, i.e., assuming intra-province coordination is already done at its best. In other words, each per-province server location can be thought of as the servers in that province at full coordination. Moreover, this practice assigns requests to servers based on the requests’ origin location rather than the server they happened to land on in the CDN. This eliminates the (non-organic) dependence of a server location’s workload on the particular CDN properties such as its existing resources and traffic mapping policies.

Similarly, in the analysis of coordination between countries, we designate one large enough server location for each country. Each server location is provisioned with enough egress capacity for the incoming load and a range of different disk sizes. We run a server location, which in the real world consists of a series of servers handling different (non-overlapping) shards of the content ID space, as one large server with the aggregate disk and egress capacity of its individual servers.

A. Efficiency Gain

Figure 8a plots the saving percentage when the per-province servers placed in a country coordinate every 10 minutes, i.e., the extra saving added if we expand intra-province to inter-province (intra-country) coordination. The figure includes the results for six countries selected across different continents of the world, including two with lingual diversity. This is to span a representative range of outcomes. Different disk sizes are considered in these experiments to examine the saving when the servers are operating at different levels of strength: the larger the disk size, the smaller the costly traffic. This traffic ranges from tens to hundreds of Gbps per country in the above experiments with different disk sizes. The incremental saving in costly traffic in this expansion step ranges from 4% to 8% in most cases.

It is interesting to note that in the two countries with lingual diversity among their provinces, Canada and India, the smallest saving is achieved, yet still positive: about 3–4%. We ensure that it is not only the overall saving that is positive (e.g., there is positive gain even in coordinating the servers in Quebec with the other Canadian provinces) through another experiment which isolates pairs of provinces and confirms the gain. For instance, the saving between the provinces of British Columbia (English) and Quebec (predominantly French) is 1.5% whereas this number is 2.2% between British Columbia and Ontario (both English); these small magnitudes are due to having only two coordinators in the isolated experiment. It is noteworthy that in the case with 10X disk size, a slight anomaly is observed. In this case, a number of servers in the experiment take a long time to fill up and reach their steady state, even longer than a month which is the duration of our datasets. This has affected the measured saving amount, which is the average value over the second half of the month, and caused a drop most significantly for India and to a smaller extent for France and Brazil. The otherwise increasing trend in cost saving with disk size indicates that by growing the depth of the caches, we deal with unpopular videos and chunks watched less and less frequently. The increasingly noisy information on a video’s popularity could better benefit from the additional information coming from other caches.

We analyze inter-country coordination in two classes: between countries located in the same continent and between countries speaking the same language irrespective of their continent. We conduct the continent-wide experiment in Europe, South and North America: from significant language diversity to no diversity: for this reason Mexico is considered South America. We conduct the language-based experiment between 6 English and 19 Spanish speaking countries. The results show that examining other continents and languages is not necessary. Figure 8b illustrates the saving in each of the cases. Coordination among countries of different languages worsens the servers’ performance, hence a negative saving. This harm can be as significant as −11% for the case of Europe and −4% for South America, where most countries speak Spanish combined with Portuguese and smaller populations of French and Dutch. North American servers (U.S. and Canada) can achieve a minor saving of up to 0.7% by extending their coordination from intra-country to between the two countries. If including another 4 English speaking countries, the saving grows to [1%, 1.4%]. This saving is smaller among Spanish speaking countries and is in [−0.5%, 1.1%], meaning a more local interest in those countries. Specifically, the saving is positive for large disks. This is because such cache spaces can
Figure 8 plots the overhead of exchanging data for the above experiments. The values represent the average rate and size of the data sent by the coordination master to each location (after compression). The size of the data in the reverse direction, i.e., pulled from each location, is therefore upper-bounded by the plotted value too. Note that the servers’ disk size is irrelevant to the message sizes. Figure 8c shows how the messages expand by extending the coordination domain. This is due to the growing number of distinct videos and chunks watched in the past 10 minutes when extending from a metro to a country and continent. The size of the messages distributed country-wide every 10 minutes is in most cases up to 60 MB which translates into <1 Mbps of amortized overhead if transmitted over the 10 minute period; the resultant delay in sending the last piece has no noticeable impact on the achieved saving as analyzed in Section VI-B. For inter-state coordination in the U.S., this overhead is 3.5 Mbps, i.e., a more diverse range of videos are watched in the U.S. Similarly, a broader range of videos are watched in English speaking countries than Spanish, although, more strongly correlated across the former (cf. Figure 8b).

Note that these numbers represent the overhead traffic from the coordination master to each server location. This overhead is typically on the order of 0.1% of the costly traffic of each designated server location in our experiments—one per province or country. However, when going to the actual individual server locations of a CDN such as all off-net locations in small ISPs, this overhead can be over 1% of the location’s costly traffic. This cancels out the saving achieved by inter-country coordination. For coordination within a country, the total overhead and consequently the exact remaining profit of coordination are CDN-specific and depend on the exact number of locations where the CDN has deployed servers. Nevertheless, our results suggest that country-wide coordination is generally an advisable practice. For example, given the 3.5 Mbps overhead and assuming a 100 Gbps saving in the costly traffic by inter-state coordination in the U.S., the total overhead can approach the saving (and render the coordination ineffective), if the number of server locations just in the U.S. reaches 30,000 which is unlikely.

C. Summary and Limitations

We found that country-wide coordination is profitable in our examined countries, it has small overhead (<4 Mbps per server location), and it brings an added cost saving of 4–8% on top of any saving by province-wide and smaller scale coordinations. This includes even the multi-lingual countries Canada and India. These results match the findings in Section IV-B on workload correlations within countries. The results suggest that country-wide coordination is generally a profitable practice. However, further coordinating servers across countries is not advised, as even the highest gain it can add to previous gains is marginal and comparable to the introduced overhead. Therefore, up to country-wide coordination is the right choice of clusters, and overlapping coordination clusters would not become necessary.

We also found that being in the same nation is a stronger factor in having similar workloads than speaking the same language across different countries: compare the coordination gain of Canadian provinces analyzed earlier and that of the 6 English speaking countries. This is a similar result to the analyses in Section IV-B on the correlation levels within and between countries (cf. Tables I and II).

We analyzed the incremental traffic saving that can be achieved by extending the coordination domain separately in each step. The percentage values provide an understanding of each extension step as well as a rough estimate of the compound saving from non-coordinating caching to country-wide coordination. However, we cannot quantify an exact value for the compound saving of the actual CDN server locations coordinating, rather than the ones designated for each province/country in our experiments. This is because our Dataset I is down-sampled with a small factor for practicality reasons. It currently contains over 2 billions of requests for millions of distinct videos. While a representative sample on a country-wide basis (and province-wide for the large countries in our experiments in this section), the small data that lands on each individual server location is not an accurate sample for realistically evaluating many of the locations with no more than a handful of servers—the case
with a considerable fraction of off-net locations. We also have not analyzed province-wide coordination between cities/metros. Besides having too small data on a per-city basis (§ IV-B), our traces also lack city-breakdown information for some parts. The corresponding gain can be assumed having values between metro-wide savings (4–11%) and country-wide (3–9%). At the end, the compound gain can be estimated as the aggregate of the gain of metro-wide coordination (where applicable; 4–11%), the gain of extending metro-wide to province-wide coordination (up to ~10%), and the gain of extending province-wide to country-wide coordination (3–9%); conservatively, from 10% to over 20% saving in costly traffic. Though, the exact quantification of the compound saving through modeling and/or modeling-assisted evaluation is suggested as future work (§ VIII); note that given the substantial volume of traffic, even the smallest end (<10%) yields up to hundreds of Gbps traffic (petabytes per day) cut on unwanted and costly paths.

VIII. CONCLUSIONS

We analyzed the benefits and the scalability of distributed cache coordination across a global video CDN. Based on real CDN workloads, we found that video popularities are strongly correlated within a metro or even a country, even considering lingual diversity (>0.5 correlation). This translates into a positive gain from coordination, as confirmed by our experiments. We found that the workloads of nearby countries may not be correlated, and correlated countries may not be nearby since language is the main driving factor for inter-country correlations. Though, even within the realm of one language, interests may be more local (Spanish) or less (English). We have built a cache coordination mechanism for real-world CDNs and showed that exchanging cache state data as infrequently as every 10 minutes can achieve nearly all the traffic saving that a 10-second, frequent exchange obtains, while it reduces the overhead to be negligible. This saving can be up to hundreds of Gbps of traffic on unwanted and costly Internet paths. Moreover, our results showed a close relationship between the correlation of workloads and the profitability of coordination. The results also showed that arbitrary coordination can actually worsen the caching performance compared to simply leaving the caches non-coordinated. We analyzed the extent to which cache coordination can scale, which guides the formation of coordination groups throughout a CDN. We found that up to country-wide server coordination is beneficial even in countries with lingual diversity, while coordination between countries is generally ineffective or harmful given the associated overheads.

Our analyses provides guidelines for static configuration of cache coordination clusters throughout the CDN. As a next step, we would like to develop an automated method for configuring cache coordination dynamically by, e.g., sampling the workloads landing on each serving site and analyzing their correlations. This is a useful technique for the possible discrepancy between the user-to-server mapping criteria of a CDN and the users’ geography, which is not uncommon for large serving sites; that is, we cannot define a bounded geographic region for the users of these sites. Furthermore, in this work, we quantified the savings of cache coordination at different scaling steps, though not the exact compound saving CDN-wide. As discussed, this requires concurrent evaluation of a remarkably large number of server locations with per-server traces of reasonable size (not significantly sampled)—a practice beyond our current means. We would like to design methods and/or models to quantify the compound savings. Finally, the analyses in this paper provide the basis for a fine-tuned, production-level coordinated caching system. Further optimizations that can be done as next steps include more involved exchange of popularity data (e.g., using bloom filters) and more complex state syncing (e.g., differentiating videos reported by most or just a few neighbors)—whether they can achieve a gain worth their complexity.

REFERENCES
