CAVE: Caching 360° Videos at the Edge

Citation for the original published paper (version of record):
Ali-Eldin Hassan, A., Goel, C., Jha, M. et al (2022). CAVE: Caching 360° Videos at the Edge. NOSSDAV 2022 - Proceedings of the 2022 Workshop on Network and Operating System Support for Digital Audio and Video, Part of MMSys 2022: 50-56.
http://dx.doi.org/10.1145/3534088.3534350

CAVE: Caching 360° Videos at the Edge

Ahmed Ali-Eldin †, Chirag Goel ‡, Mayank Jha‡, Bo Chen*, Klara Nahrstedt*, Prashant Shenoy‡
† Chalmers University of Tech., ‡ University of Massachusetts Amherst, * University of Illinois Urbana-Champaign
† ahmed.hassan@chalmers.se, ‡ {cgoel, mkjha}@cs.umass.edu, * {boc2,klara}@illinois.edu, ‡ shenoy@cs.umass.edu

ABSTRACT

While 360° videos are gaining popularity due to the emergence of VR technologies, storing and streaming such videos can incur up to 20X higher overheads than traditional HD content. Edge caching, which involves caching and serving 360° videos from edge servers, is one possible approach for addressing these overheads. Prior work on 360° video caching has been based on using past history to cache tiles that are likely to be in a viewer's field of view and has not considered methods to intelligently share a limited edge cache across a set of videos that exhibit large variations in their popularity, size, content, and user abandonment patterns. Towards this end, we present CAVE, an adaptive edge caching framework that intelligently optimizes cache allocation across a set of videos taking into account video content, size, and popularity. Our experiments using realistic video workloads shows CAVE improves cache hit-rates, and thus network saving, by up to 50% over state-of-the-art approaches, while also scaling to up to two thousand videos per edge cache. In addition, in terms of scalability, our developed algorithm is embarrassingly parallel, allowing CAVE to scale beyond state-of-the-art solutions that typically do not support parallelization.

CCS CONCEPTS

• **Networks → Location based services**; **Network management**.

ACM Reference Format:
Ahmed Ali-Eldin †, Chirag Goel ‡, Mayank Jha‡, Bo Chen*, Klara Nahrstedt*, Prashant Shenoy‡. 2022. CAVE: Caching 360° Videos at the Edge. In *The 32nd edition of the Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'22), June 17, 2022, Athlone, Ireland*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3534088.3534350

1 INTRODUCTION

While 360° videos are becoming popular since they provide a more immersive experience to users compared to traditional HD videos. However, streaming a high resolution 360° video can require more than 500 Mb/s in comparison to 25 Mb/s for a 4K high definition 2D video, i.e., a 20x increase in bandwidth. Streaming at such high-bandwidth over the Internet can be problematic, therefore the use of edge caching is appealing. Since the size of 360° videos is substantially larger than traditional videos, more intelligent caching schemes that make efficient use of storage space at the edge without compromising network bandwidth saving are desirable. As a result, several research efforts have recently proposed intelligent caching schemes for 360° videos [5, 19, 23].

One such promising approach to reduce storage and bandwidth requirements at the edge, is to employ Field-of-View (FoV) aware caching [19, 23]. Since the human eye is only capable of focusing on a limited area, spanning around 60° horizontally) and around 55° vertically [26], the key premise of FoV-aware caching methods is that multiple users watching the same video have similarities in their FoVs in terms of what portions and objects within each 360° frame are viewed. Thus, caching popular FoVs in the video at the highest bit-rate, while caching less popular FoVs at lower bit-rates can significantly reduce the storage requirements at the edge. Typically, a 360° video streams at 30 to 120 frames per second [16], with each frame divided into tiles. A FoV is thus composed of a subset of spatially contiguous tiles (see Figure **??**). FoV-aware approaches cache all tiles of a frame at low resolution and the tiles of common FoV at high resolution. In doing so, they save substantial storage space at the edge cache while ensuring the most likely tiles to be viewed are delivered from the edge at high resolution, yielding network bandwidth and storage savings.

While FoV-aware caching shows promise for more efficient storage space usage on edge proxy servers, current approaches in the literature have some limitations. First, current 360° edge caching methods do not perform any active cache space management—by virtue of treating each video independently and using a passive LRU/LFU cache. However, intelligent cache management across multiple videos and even within a video can improve hit rates while maximizing network bandwidth savings. Today, to the best of our knowledge, there are no existing active caching methods for 360° videos that optimize cache space across videos. Video popularity distributions typically follow a Zipf distribution. Hence, allocating higher storage space to cache more tiles of popular videos can yield disproportionately higher network saving. Similarly, variance between FoVs of users for a single video, varies considerably as seen in Figure 2 with some tiles very popular while others seldom viewed. Hence, rather than caching a fixed number of tiles per frame, adapting the number of tiles based on FoV variance, or lack thereof, coupled with video and frame/FoV popularities can yield better savings compared to existing caching solutions.

Motivated by the above observations, we present, CAVE, an adaptive approach to *CA*che *V*ideos at the *E*dge. Our paper makes the following contributions:

- Our approach uses observed popularity of videos along with user viewing variance to optimally partition the cache space across multiple videos to maximize network savings.
- Our approach also performs fine-grain caching adaptation within a video. It adaptively varies the numbers of tiles cached per frame based on the FoV viewing patterns, while taking into account video abandonment [25, 28] .

NOSSDAV'22, June 17, 2022, Athlone, Ireland

ACM ISBN 978-1-4503-9383-6/22/06…$15.00
https://doi.org/10.1145/3534088.3534350

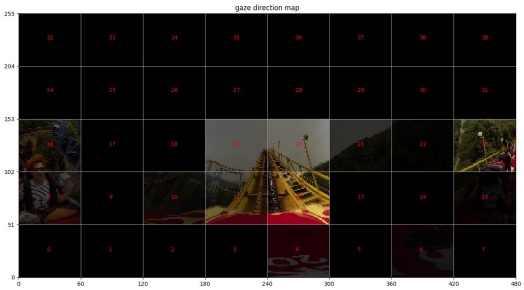

Figure 1: An equi-rectangular representation of a 360° video divided into 40 tiles. The brighter a tile, the more popular it is among the users in the dataset. (based on the roller-coaster video from [9])

- We show using real and synthetic video traces that CAVE improves cache hit-rates under realistic user-abandonment, and popularity dynamics by up to 50% over state-of-the-art caching methods. In addition, our system is parallelizable, and is capable of handling thousands of videos in parallel.

2 BACKGROUND

There has been many recent advances in the field of 360°videos, 360°streaming, and on edge-caching. We hence start by providing a background on these areas.

2.1 360° Videos

A 360° video encompasses a 360° view of a scene from the perspective of the camera with each frame providing an omnidirectional spherical view of the captured scene [16]. The spherical view is mapped onto a 2D plane for purposes of encoding To view the video, users typically use a head mounted display (HMD) that provides an immersive view of the captured scene. A 360° video sphere is divided into a number of tiles (see Figure 1). A tile is a motion-constrained rectangle, typically of fixed size, that is encoded using DASH [29]. A head mounted display can display only a portion of the 360° video frame at any instant. This subset is referred to as the user Field of View (FoV). HMDs have sensors that track the user's head movement as the user turns their head, and the FoV corresponding to the the head's orientation is displayed from the 360° image, allowing for an immersive experience. Typically, an FoV covers a 120°×55° portion of the 360° scene. Since only a small subset of the tiles in each frame are actually viewed by the user (as shown in Figure 1 where the darkest tiles are never viewed), it is quite wasteful from a network standpoint to transmit all the tiles in each frame when only a subset of them will be displayed.

2.2 Adaptive 360° Streaming

To address the above issue, researchers have developed adaptive 360° streaming techniques based on DASH. These adaptive streaming techniques predict head movement. The predictions are used by the client to estimate a FoV and prefetch tiles of the predicted FoV at higher resolution [4, 27]. If tiles of the video are encoded at multiple resolutions, the tile outside the predicted FoV can be prefetched at lower resolution in case the head prediction is incorrect. Therefore, client-side prefetching still allows for the tiles in the actual FoV to be displayed but at a lower quality. Such adaptive techniques yield

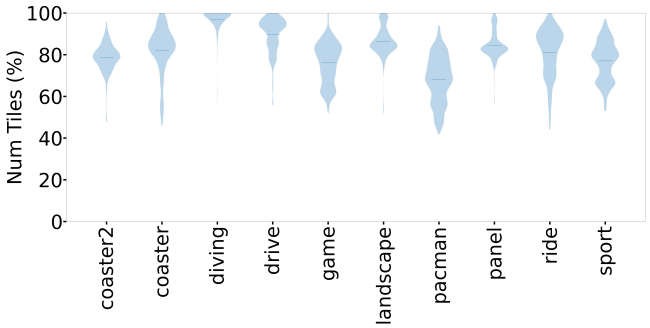

Figure 2: A Violin plot of the distribution of the number of tiles viewed in a frame from the dataset in [17] with 50 users viewing each video. The percentage of tiles viewed per frame varies widely within a video and across different videos.

significant bandwidth savings over a naive approach of fetching all tiles of each frame at high resolution from the server[12, 22, 31].

2.3 360° Edge Caching

While head-movement predictions and adaptive FoV prefectching by the end-client yield network savings, streaming 36-° videos over the Internet still consume considerable bandwidth. Consequently, researchers have studied approaches for caching 360° videos on edge proxies, serving DASH streaming requests from the edge to further reduce the network bandwidth usage [19, 23]. However, the large sizes of 360° videos imply that storing the entire videos at high resolution on edge servers is not feasible [23]. Consequently, similar to FoV prediction-based adaptive 360° streaming approaches, edge caching approaches can estimate a possible FoV for each frame and cache a subset of the tiles in the estimated FoV at full resolution and other tiles at lower resolution [23].

There are two main approaches to 360° caching, namely, tile-based global caching [23] and history-based FoV caching [19]. Tile-based caching methods, e.g., in [23], uses an optimization based on multi-choice knapsack to determine which subset of tiles to cache to minimize error. The system solves a global optimization problem that considers every tile in every video as part of the optimization of the entire system. The main shortcoming with this approach is that it does not scale beyond a limited number of videos. The second approach, FoV aware caching, uses actual histories of viewing patterns to determine a common FoV based on correlations of FoVs across users [19]. This approach does resolve many of the issues with the tile-based method. Nevertheless, the approach on looks at videos individually, not considering how videos vary in popularity, and hence does not provide adaptive methods for sharing the cache space across *multiple* 360° videos. Optimally partitioning the cache space across multiple competing videos is challenging and prior 2D caching methods [28] do not directly apply due to new challenges raised by 360° videos.

Within each video, correlation between FoV across users may vary from scene to scene. To show an example, Figure 2 shows the distributions of the number of tiles viewed in each frame across all users as a violin plot in a video dataset with 50 users. The distributions vary considerably with some video frames having all their tiles viewed, while some other frames will see that less than 50% of their tiles viewed. This Figure suggests that splitting

the cache equally between all videos will have diminishing returns since some videos have a much smaller variance when it comes to which tiles are viewed by a user.

In addition, caching more tiles per frame for a very popular video and fewer tiles per frame for a less popular one will yield higher network savings over a static approach that caches a fixed number of tiles per frame for all videos. Similarly, caching fewer tiles for a frame with highly correlated FoVs and more for a frame with overlapping, but less correlated FoVs, may increase hit rate over caching a constant number of tiles per frame. Such adaptive caching algorithms across and within omnidirectional videos have not been explored previously.

3 CAVE OVERVIEW

3.1 Problem Statement

We assume an edge caching service where the content owner would like to cache a mixture of videos that are predicted to be popular, e.g., a newly released episode in a popular series, or a movie that is anticipated to be popular, along with videos that are already highly popular. We assume that the caching storage size at the edge is limited, and that the edge proxy can only cache a fraction of the video catalogue. The goal of our paper is to develop an adaptive 360° edge caching system with the following characteristics:

- The edge cache should perform adaptive cache management that uses video popularities to adapt the cache space used for each video so as to optimize hit rate and network savings; the allocations should adapt as popularities change over time.
- The edge cache should employ adaptation within a video using FoV correlation data to adapt how many tiles of each frame should be cached.

Our adaptive caching approach is guided by a key observation: different videos should be allocated a proportion of the cache that is dependent on the video popularity, the video size, the frame order, and the per tile popularity. As video popularities change over time, so should the cache allocations. When new videos are pushed to the cache (e.g., by a content provider that is releasing a new show or movie), a content-based method should be used to intelligently seed the cache with tiles that are likely to see the most hits.

3.2 Formulating Edge Optimization

Consider an edge cache with size S. The cache hosts a set $V_{cache} = \{v_{c1}, v_{c2}, ..., v_{cn}\}$ of n 360° videos chosen out of a large video catalogue $V_{cat} = \{v_1, v_2, ..., v_c\}$, i.e., $V_{cache} \subset V_{cat}$. Each video v_i is composed of a set of frames $F = \{f_{i1}, f_{i2}, ..., f_{ij}\}$, and each frame f_{ij} of video v_i is composed of a set of $T = \{t_{ij1}, t_{ij2}, ..., t_{ijk}\}$ tiles, each of which can be cached at either low resolution, or a high resolution. Each tile t_{ijk} has a popularity based on the number of users u_{ijk} who viewed the tile in their FoV over the past τ hours. Each video v_i in the catalogue V_{cat} has popularity u_i representing the number of users who have streamed any part of v_i over the past τ hours. Our goal is to select the subset of the videos forming V_{cache}, from the catalogue, to be cached at the edge. The selected videos should be a combination of videos with minimal or no history (based on anticipated popularity by the content owner), and popular videos.

Once a video is selected, the entire parts of the video that have been viewed, i.e., in case of abandonment, is cached in some resolution. Cached tiles can either be stored at low resolution or high resolution. Our goal is to maximize the number of high resolution tiles served to users from the edge caching layer across all videos, i.e., a perfect cache would only host the tiles from the FoVs that will be viewed by future users in the highest possible resolution—as any other tiles waste cache space. To achieve this goal, the optimization problem can be formulated as a knapsack problem where we are trying to maximize the success rate of selecting high resolution tiles streamed from the cache, which only happens if the most popular tiles are cached. We first define a per tile score for a tile t_{ijk} based on both the popularity of the tile, and the frame to which the tile belongs. We define the per tile score to be, h_{ijk}, where u_{ijk} is a history based score. The per tile score can be viewed as the utility of a tile. The knapsack problem is then aiming to maximize the utility of all cached tiles. This can be formulated as:

$$\text{Maximize} \sum^{V_{cat}} \sum^{F} \sum^{T} u_{ijk}\chi_{ijk}. \tag{1}$$

where χ_i is a decision variable whether a tile t_{ijk} is cached. Assuming that each tile has a size of B MB, then the above optimization is subject to the following constraints,

$$\chi_{ijk} \in \{0, 1\}, \tag{2}$$

$$u_{ijk} \geq P_{tile}, \tag{3}$$

$$\sum^{V_{cat}} \sum^{F} \sum^{T} B\chi_{ijk} \leq S_{highRes}, \tag{4}$$

where P_{tile} is the minimum number of viewers that need to have the tile in their FoV for that tile to be cached, and $S_{highRes}$ is the cache space available for high-resolution tiles. In this formulation, the first constraint is the decision constraint on which tiles to cache. The second constraint is a constraint on the per tile popularity which discards tiles that are not-popular is users' FoVs. The third constraint states that the sum of all the cached tiles should fit in the given cache size.

4 OPTIMIZING CACHE ALLOCATIONS

Knapsack problems are NP-hard making the previously formulated problem intractable for any realistic 360° video caching service with long videos and/or a large catalog of videos. We thus use a heuristic approach to solving the problem; dividing the optimization into two sub-problems, the first of which is to find which videos to cache and how much to allocate for each video. The second one solves the problem of which tiles within a selected video will be cached in the allocated cache space.

To motivate our heuristic based approach, we make the following observation; the popularity u_{ijk} of any tile t_{ijk} is less than or equal to the number of views u_{ij} of the encompassing frame. Since each frame contains anywhere between tens to a few hundred tiles, using a heuristic solution that considers frame popularity as an indirect measure to the tile popularity, we significantly simplify the optimization problem and reduce the solution space. However, since some frames will have users looking everywhere, only relying on frame popularity might result in caching tiles with lower popularity—as tiles in less popular frames with all viewers having

overlapping FoVs have a higher u_{ijk}. In order to mitigate this problem, we couple the frame popularity with the statistical variance in the popularity of the number of views per tile within a frame. *Higher variance between tile popularities of a frame indicate that a few tiles in the frame have large u_{ijk} values, and thus are popular tiles, while the rest of the frame tiles have low or no views. Low variance would indicate that user views tend to be equally distributed across all tiles of the frame, and therefore viewers are looking everywhere in that frame.* These two observations for the basis of our cache size optimization algorithm.

4.1 Cross-video cache allocation

Streaming service videos have popularity dynamics that typically follows a Zipf like distribution [2][33]. Hence, only popular videos or videos that are expected to have high popularity in the near future should be cached. Videos with no view history are cached based on their content only. In addition, the subset of videos that have a popularity higher than a threshold $P_{threshold}$ are also cached, but based on the history viewing patterns. Algorithm 1 describes our popularity based scoring approach to videos. We first filter all

Data: Size of Cache S, Video Catalogue V_{cat}, video popularity u_i, Tile popularity u_{ijk}, video popularity threshold $P_{threshold}$
Result: Score for every v_i and f_{ij}

1 $V_{cache} \leftarrow [\,]$;
2 Find set of videos V_{cache} with $u_i > P_{threshold}$;
3 **for** *each video* v_i *in* V_{cache} **do**
4 $\quad \phi_{ij} \leftarrow$ Calculate frame-wise score using eq 5;
5 $\quad \overline{\phi_i} \leftarrow$ mean of top scores h_{ij} that fit S/n cache size;
6 $\quad h_i \leftarrow u_i/\overline{\phi_i}$;
7 Calculate allocation for each video $A_i \leftarrow \frac{h_i S}{\sum_n h_i}$;

Algorithm 1: Finding a score for each video to cache

videos in the catalogue choosing the ones with popularity greater than $P_{threshold}$. This leaves us with a set of n videos to be cached based on history. We note that some of these videos will have high user abandonment, and thus only a fraction of these videos will be streamed. Caching abandoned content that will not be streamed is a waste of the edge resources.

As noted, to decide which tiles to cache, video frames with less variability in FoVs between users would require relatively less cache space as compared to videos with higher variations. FoV variations in video frames can be calculated using the variance in the tile popularity for each frame. Hence, a frame with high popularity and with views concentrated on a subset of tiles, i.e., having higher statistical variance, is the most desirable to cache while a frame with low popularity and equal distribution of views across all tiles, hence low statistical variance, is less desirable. We calculate a per frame score, ϕ_{ij}, for each frame f_{ij} of a video v_i based on the tile views variance σ_{ij} and the number of users u_{ij} who viewed that frame as follows,

$$\phi_{ij} = u_{ij} * \sigma_{ij}^2. \tag{5}$$

In order to find the per video cache allocation size, we first assume that the cache is split equally among the chosen videos, i.e., each video will be allocated S/n of the cache size. This is a good approximation to start the allocation with when no history is available. In addition, this assumption allows us to calculate a weight h_i for the entire video v_i that can be used for comparing the utility of caching each of the videos. To do so, the frames across each video are ranked based on their scores to find the top subset of the frames that fit in S/n size of the cache. We note that this weighting is not directly used to choose which frames to cache, but are rather a first step towards finding the per video cache allocation. The frame scores of each video are averaged to get ($\overline{\phi_i}$) for each video v_i. As The overall video popularity u_i is then normalized by ϕ_i, such that, $h_i = u_i/\overline{\phi_i}$. In effect, this normalization allows us to normalize the overall video popularity with the average popularity of all frames as the average popularity of all frames has a maximum equal to u_i, but can be lower based on user abandonment. Since ϕ_i also includes how concentrated are the FoVs of users are, this means that h_i as a score would decrease the overall allocation to videos with concentrated FoVs as they do not need a lot of space. Once all videos are scored, the next step in CAVE's cache optimization is choosing the per video cache size A_i using the following heuristic,

$$A_i = \frac{h_i S}{\sum_n h_i}. \tag{6}$$

4.2 Frame and Tile Level allocation

Once the per video allocation is set, CAVE then optimizes the per frame and per tile allocation (as described in Algorithm 2). Each frame is assigned a weight inversely proportional to its variance but directly proportional to its popularity (Line 4 and 5). Since in almost all cases, abandonment happens after viewing the first few minutes of a video, earlier frames tend to have much higher u_{ij} compared to a later frame. This observation means that for DASH streaming, all frames required for an earlier segment will be available.

The frames are then sorted based on their weights and their position in the video (Line 6). For each frame, CAVE computes a normalized weight ω_{ij} (Line 9), that is then used to get an initial size allocation $A_{initial}$ proportional to that weight (Line 10). However, the actual size allocated to a frame is the minimum between three values; $A_{initial}$, the remaining available cache size for the video A_i, or the size T_i of all tiles that have tile user views $u_{ijk} > P_{tile}$ (Line 11). CAVE then recalculates the remaining cache space that can be used for the rest of the frames until no remaining cache space is available (Line 12).

Tile Scoring Each tile in a frame of a video is assigned a score. After finalizing the video and frame allocation optimization, the highest scoring tiles that fit every frame allocation are cached at the highest resolution. The remaining tiles in the frame are cached at the lowest resolution

5 EVALUATION

We implemented CAVE in Python 3.7 (around 1000 LoC). We integrate our implementation with an emulator for evaluating the performance at scale. As there are limited datasets available for 360° videos, the emulator either streams the 360° videos from the real (alas limited) dataset or uses synthetic workloads as described next.

Data: V_{cache}, h_i, u_{ijk}, S
Result: Cache allocation size for every v_i and f_{ij}

$F_{cache} \leftarrow$ Cached frames for a video;
for *each video* v_i *in* V_{cache} **do**
  for *each frame* f_{ij} *in* v_i **do**
    $h_{ij} = u_{ij}/\sigma^2_{ij}$;
  $F_i \leftarrow$ sort frames based on h_{ij} and for frames with equal score, sort earlier frames first;
  for *each frame* f_{ij} *in* F_i **do**
    while $A_i > 0$ *and* $h_{ij} > 0$ **do**
      $\overline{\omega}_{ij} \leftarrow \frac{h_{ij}}{\sum_{v_i} h_{ij}}$;
      $A_{initial} \leftarrow \omega_{ij} A_i$;
      $A_{ij} \leftarrow min(T_i, A_{initial}, A_i)$;
      $A_i \leftarrow A_i - A_{ij}$;

Algorithm 2: Frame and Tile Level optimization

5.1 Evaluation Setup

Datasets We use a real dataset [17] which includes traces from 50 users across 10 one-minute videos. The dataset is diverse, covering slow and fast paced videos. and includes computerized and natural images. Every frame is divided into 200 tiles, and on average an FoV contains approximately 34 tiles. While the dataset provides some key insights to how users view 360° videos, it is limited in the number of videos, the number of users, and the length of videos. In addition, the dataset does not capture video popularity dynamics or user abandonment as one would expect in a streaming service.Video popularity in streaming services tend to follow a gamma distribution, whereas user abandonment follows a Zipf-Mandelbrot distribution [3, 6–8, 18].

We therefore synthesize two larger traces based on the real dataset. We generate video popularities using the gamma distribution parameterized based on video view-counts on Youtube. The generated dataset construct synthetic videos ten times longer than the real dataset. To capture user abandonment behaviour, length of streaming session is varied using the Zipf-Mandelbrot distribution. For each frame, the probability of every FoV is calculated using the data from the dataset with an added Gaussian noise. We generate two datasets, "Workload A" which has a total of 955 user traces, and "Workload B", which has a total of 1910 user traces.

Performance Comparison. To evaluate the performance of our approach, we run experiments comparing CAVE to the FoV based caching described in [19]. However, since the work in [19] focuses on passive caching, and our focus is on active caching, we adapt the approach for active caching operation. We refer to [19] as *history-based* in the rest of this Section. In addition, we compare a minimalist version of CAVE with no cross-video optimization, but only using tile scores from historical data and content based analysis, to the full CAVE with all added optimization levels. This minimal version of CAVE is akin to adding content-based caching plus frame-based optimization to approaches such as the one in [19].

5.2 Evaluating CAVE

Evaluating the need for Cross Video Optimization. To show why inter-video cache adaptation is desirable for maximal cache utility, we plot the cache hit-rates for the individual videos in the dataset when only using frame-level adaptations as shown in Figure 3. We make multiple observations; first, there are diminishing returns for increase cache sizes for a video beyond a certain point as less popular FoVs do not really add to the cache performance even if stored. For example, if the 25% top scoring tiles from the Pacman video are cached, the cache hit-rate is around 92%. Even doubling the allocated cache size to this video will result in no material benefits. The second observation we make is that the hit-rate varies widely between videos, suggesting that instead of using the extra caching capacity available for videos with hit-rates already above 90%, the capacity should be reallocated to videos that can see an improvement by utilizing this extra cache capacity, thus increasing the overall systems performance and cache hit-rates.

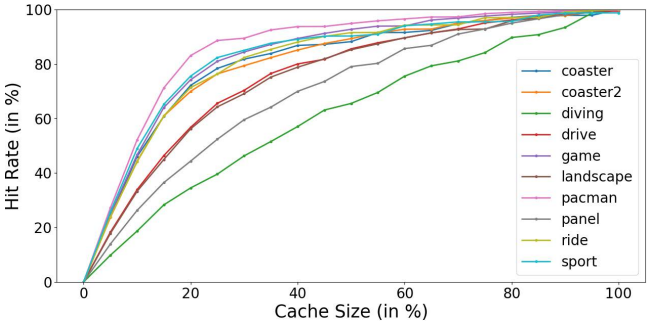

Figure 3: Since some videos have varying slopes, we can exploit their higher returns on lower cache sizes.

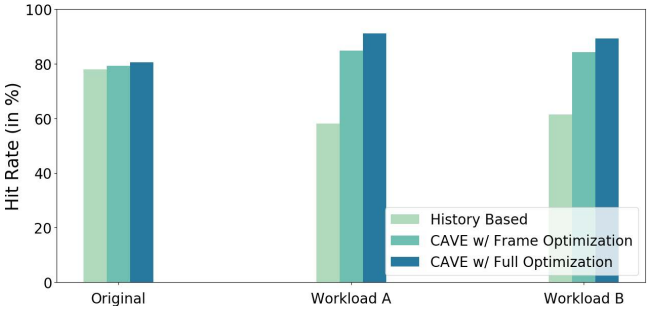

Figure 4: Comparison with State of the Art: Hit rates of history-based approach and CAVE.

5.3 Comparison With History-based Methods

We now show how CAVE improves the caching performance with realistic popularity dynamics and also compare it with the state of the art history-based approach. In this experiment, we use the two synthetic datasets—described previously—in the emulator to emulate the user viewing patterns when the cache size amounts to 35% (2 times average FoV size) of the total video sizes. We also evaluate the full CAVE on the limited real dataset that we use without modifying anything in that dataset. Figure 4 shows the hit-rate when popularity dynamics are present in the workload. We compare CAVE with inter and intra video allocation optimizations along with object, tile, and frame scoring mechanisms in addition to

Table 1: Comparison of QoE (Workload B)

	History-Based	CAVE
Macular Vision	71.8%	90.05%
Near Peripheral	69.8%	89.88%
Mild Peripheral	61.2%	89.06%

history-based approaches. *Our results show that CAVE outperforms history-based approaches, increasing the hit-rate by over 50%.* Even with no optimizations, our approach outperforms history-based approaches by over 25% increase in cache hit-rate. These gains, not only translate to better caching performance, but also to network savings. Every cache miss results in the user device requiring to (expensively) fetch data from the remote content owner servers. This translates to much worse delays at the user devices, and a much higher bandwidth requirements for the content provider. We also measured the cache sizes CAVE allocates to different videos in all experiments. In the interest of space, we report only that the diving video in the real dataset was 29% of the entire cache, and the minimum cache allocation was for the pacman video which was allocated 5% of the entire cache size. These are the videos with the most and least FoV variations in the dataset (see Figure 2).

5.4 Quality of Experience

360° videos are very interactive and appealing but also bring challenges in quality of experience. For example, having a mix of high resolution and low resolution tiles within a FoV can cause motion sickness, headache, dizziness, etc. This effect gets amplified when it's within the user's macular vision as compared to the user's peripheral vision (see Figure **??**). Hence, it is important for caching and streaming algorithms to take this as a metric for evaluation as a high hit rate alone might not be sufficient to build an effective system. In Table 1 we compare CAVE with history-based approaches w.r.t. the percentage of tiles from different areas of vision that get served from the cache. We see that CAVE, in efforts to increase hit rate, did not imply a drop in QoE for macular vision which remains to be the best performing as compared to the peripheral vision.

5.5 System Scalability

Using frame level (and not on tile level) optimization in CAVE results in increased optimization scalability. The optimization is highly parallelizable as the frame and tile level allocations are independent across videos, while the cross video optimization being mostly parallelizable also. Since CAVE is an active cache, cache reallocation is run periodically as defined by the provider, but possibly every couple hours. To show the scalability of CAVE, we ran three experiments assuming that the cache holds 500, 1000, and 2000 videos respectively, with an average video length of 20 minutes, and an average video size of 2.5 GB. We ran the experiments for edge cache sizes 0.5 TB, 1 TB and 2 TB respectively. When using one core on a laptop to run the optimizations, the average computation times for each of the above scenarios is, 10 minutes, 25.67 minutes, and 44 minutes respectively However, the computations are embarrassingly parallel, so for an 8-core edge server, the average computation times are 1.25 minutes, 3.5 minutes, and 8 minutes respectively. Given that the optimization run every few hours, we believe that the time taken by CAVE is adequate even when the dataset size is much larger since using a 32 core processor would significantly cut the time. In addition, these optimizations do not need to run on the edge nodes themselves, but can be rather run on a more powerful server in the cloud. In comparison, prior approaches are less scalable and not amenable to parallelization.

6 RELATED WORK

With newer open-source and proprietary 360° datasets [9, 11, 13, 15, 17, 21, 32], there has been a surge in research on 360° streaming.
Workload characterization of 360° videos has been extensively studied. This includes characterizing user head movement behavior and analyzing the aggressiveness of tile prefetching versus required storage [4], characterizing the bitrates of thousands of YouTube 360° videos [1], studying cross-user similarities between viewers of the same video and utilizing them for caching [5], and saliency analysis of videos and their correlation with user viewing behavior [21].
Streaming 360°videos is another well studied problem with many proposed approaches [10, 34]. The Navigation Graph [24] approach models both the temporal and spatial viewing behaviors for performing view predictions. Guan et al. [14] utilize the moving speed of user's viewpoint (degs/sec), change in luminance and difference in depth-of-fields to as quality-determining factors to model user-perceived quality. The authors then introduce Pano, a streaming system that leverages the above factors providing variable-sized tiling schemes to balance encoding efficiency and QoE. Finally, CAVE can be easily integrated with many of the proposed streaming techniques in the literature. One particularly interesting approach is the ones developed in BAS-360° [30] and in ClusTile [35]. BAS-360°introduces the concept of macro-streaming units consisting of a set of tiles representing a meaningful visual part while rendering. ClusTile develops , a method to compute tiling of 360-degree videos to reduce streaming bandwidth. CAVE can use these units as the basis for caching, reducing the computational cost of CAVE while allowing for much better cache optimizations..
Caching 360° videos has also received some attention. Caching videos at small base stations has been proposed. [20] for content caching and delivery that relies on the coding of 360° videos into multiple tiles and layers to determine where a video should be placed in the network. A caching scheme the decides the per tile resolution based on the previous viewing statistics of the videos has also been proposed [23].

7 CONCLUSION

This paper presented CAVE, an active edge caching framework for 360° videos taking that leverages video popularity, content, length, and abandonment rate. We proposed content-based FoV estimation based on saliency maps and objects in each frame that indicate viewer interest, and adaptively combined it with viewer FoV histories. We presented intelligent methods to adaptively partion the edge cache across videos and across frames and tiles within each video. Our results showed that CAVE can provide up to 50% improvement in cache hit-rates, and thus network bandwidth reduction over the current state-of-the-art. Our future work will involve use of segment-based caching and use of other vision-based methods for further inter and intra-video cache optimization.

Acknowledgment This work was funded by NSF grants 1836752 and 2105494, Army Research Lab contract W911NF-17-2-0196, and NGI-Atlantic grant #04-336.

REFERENCES

[1] S. Afzal, J. Chen, and K. Ramakrishnan. Characterization of 360-degree videos. In *Proceedings of the Workshop on Virtual Reality and Augmented Reality Network*, pages 1–6, 2017.

[2] A. Ali-Eldin, M. Kihl, J. Tordsson, and E. Elmroth. Analysis and characterization of a video-on-demand service workload. In *Proceedings of the 6th ACM Multimedia Systems Conference*, pages 189–200, 2015.

[3] A. Ali-Eldin, M. Kihl, J. Tordsson, and E. Elmroth. Analysis and characterization of a video-on-demand service workload. In *Proceedings of the 6th ACM Multimedia Systems Conference*, MMSys '15, page 189–200, New York, NY, USA, 2015. Association for Computing Machinery.

[4] M. Almquist, V. Almquist, V. Krishnamoorthi, N. Carlsson, and D. Eager. The prefetch aggressiveness tradeoff in 360 video streaming. In *Proceedings of the 9th ACM Multimedia Systems Conference*, pages 258–269, 2018.

[5] N. Carlsson and D. Eager. Had you looked where i'm looking: Cross-user similarities in viewing behavior for 360° video and caching implications. In *Proc. ACM/SPEC International Conference on Performance Engineering (ACM/SPEC ICPE),*, 2020.

[6] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon. I tube, you tube, everybody tubes: Analyzing the world's largest user generated content video system. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, IMC '07, page 1–14, New York, NY, USA, 2007. Association for Computing Machinery.

[7] X. Cheng, C. Dale, and J. Liu. Statistics and social network of youtube videos. In *2008 16th Interntional Workshop on Quality of Service*, pages 229–238, 2008.

[8] X. Cheng, J. Liu, and C. Dale. Understanding the characteristics of internet short video sharing: A youtube-based measurement study. *IEEE Transactions on Multimedia*, 15(5):1184–1194, 2013.

[9] X. Corbillon, F. De Simone, and G. Simon. 360-degree video head movement dataset. In *Proceedings of the 8th ACM on Multimedia Systems Conference*, pages 199–204, 2017.

[10] M. Dasari, A. Bhattacharya, S. Vargas, P. Sahu, A. Balasubramanian, and S. R. Das. Streaming 360-degree videos using super-resolution. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pages 1977–1986. IEEE, 2020.

[11] E. J. David, J. Gutiérrez, A. Coutrot, M. P. Da Silva, and P. L. Callet. A dataset of head and eye movements for 360 videos. In *Proceedings of the 9th ACM Multimedia Systems Conference*, pages 432–437, 2018.

[12] C.-L. Fan, J. Lee, W.-C. Lo, C.-Y. Huang, K.-T. Chen, and C.-H. Hsu. Fixation prediction for 360 video streaming in head-mounted virtual reality. In *Proceedings of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 67–72, 2017.

[13] S. Fremerey, A. Singla, K. Meseberg, and A. Raake. Avtrack360: an open dataset and software recording people's head rotations watching 360° videos on an hmd. In *Proceedings of the 9th ACM Multimedia Systems Conference*, pages 403–408, 2018.

[14] Y. Guan, C. Zheng, X. Zhang, Z. Guo, and J. Jiang. Pano: Optimizing 360 video streaming with a better understanding of quality perception. In *Proceedings of the ACM Special Interest Group on Data Communication*, pages 394–407. 2019.

[15] H.-N. Hu, Y.-C. Lin, M.-Y. Liu, H.-T. Cheng, Y.-J. Chang, and M. Sun. Deep 360 pilot: Learning a deep agent for piloting through 360 sports videos. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1396–1405. IEEE, 2017.

[16] J. Huang, Z. Chen, D. Ceylan, and H. Jin. 6-dof vr videos with a single 360-camera. In *2017 IEEE Virtual Reality (VR)*, pages 37–44. IEEE, 2017.

[17] W.-C. Lo, C.-L. Fan, J. Lee, C.-Y. Huang, K.-T. Chen, and C.-H. Hsu. 360 video viewing dataset in head-mounted virtual reality. In *Proceedings of the 8th ACM on Multimedia Systems Conference*, pages 211–216, 2017.

[18] L. Maggi, L. Gkatzikis, G. Paschos, and J. Leguay. Adapting caching to audience retention rate. *Computer Communications*, 116, 12 2017.

[19] A. Mahzari, A. Taghavi Nasrabadi, A. Samiei, and R. Prakash. Fov-aware edge caching for adaptive 360 video streaming. In *Proceedings of the 26th ACM international conference on Multimedia*, pages 173–181, 2018.

[20] P. Maniotis, E. Bourtsoulatze, and N. Thomos. Tile-based joint caching and delivery of 360° videos in heterogeneous networks. *IEEE Transactions on Multimedia*, 2019.

[21] A. Nguyen and Z. Yan. A saliency dataset for 360-degree videos. In *Proceedings of the 10th ACM Multimedia Systems Conference*, pages 279–284, 2019.

[22] A. Nguyen, Z. Yan, and K. Nahrstedt. Your attention is unique: Detecting 360-degree video saliency in head-mounted display for head movement prediction. In *Proceedings of the 26th ACM international conference on Multimedia*, pages 1190–1198, 2018.

[23] G. Papaioannou and I. Koutsopoulos. Tile-based caching optimization for 360 videos. In *Proceedings of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 171–180, 2019.

[24] J. Park and K. Nahrstedt. Navigation graph for tiled media streaming. In *Proceedings of the 27th ACM International Conference on Multimedia*, MM '19, page 447–455, New York, NY, USA, 2019. Association for Computing Machinery.

[25] S. Sen, J. Rexford, and D. Towsley. Proxy prefix caching for multimedia streams. In *IEEE INFOCOM*, volume 3, pages 1310–1319 vol.3, 1999.

[26] R. H. Spector. *Visual Fields–Clinical Methods: The History, Physical, and Laboratory Examinations.* Butterworths, 1990.

[27] L. Sun, F. Duanmu, Y. Liu, Y. Wang, Y. Ye, H. Shi, and D. Dai. Multi-path multi-tier 360-degree video streaming in 5g networks. In *Proceedings of the 9th ACM Multimedia Systems Conference*, pages 162–173, 2018.

[28] K.-L. Wu, P. S. Yu, and J. L. Wolf. Segment-based proxy caching of multimedia streams. In *Proceedings of the 10th international conference on World Wide Web*, pages 36–44, 2001.

[29] M. Xiao, C. Zhou, Y. Liu, and S. Chen. Optile: Toward optimal tiling in 360-degree video streaming. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 708–716, 2017.

[30] M. Xiao, C. Zhou, V. Swaminathan, Y. Liu, and S. Chen. Bas-360: Exploring spatial and temporal adaptability in 360-degree videos over http/2. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pages 953–961. IEEE, 2018.

[31] L. Xie, Z. Xu, Y. Ban, X. Zhang, and Z. Guo. 360probdash: Improving qoe of 360 video streaming using tile-based http adaptive streaming. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 315–323, 2017.

[32] Y. Xu, Y. Dong, J. Wu, Z. Sun, Z. Shi, J. Yu, and S. Gao. Gaze prediction in dynamic 360 immersive videos. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5333–5342, 2018.

[33] H. Yu, D. Zheng, B. Y. Zhao, and W. Zheng. Understanding user behavior in large-scale video-on-demand systems. *ACM SIGOPS Operating Systems Review*, 40(4):333–344, 2006.

[34] Y. Zhang, P. Zhao, K. Bian, Y. Liu, L. Song, and X. Li. Drl360: 360-degree video streaming with deep reinforcement learning. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 1252–1260. IEEE, 2019.

[35] C. Zhou, M. Xiao, and Y. Liu. Clustile: Toward minimizing bandwidth in 360-degree video streaming. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pages 962–970. IEEE, 2018.