VisualCloud Demonstration: A DBMS for Virtual Reality

Brandon Haynes, Artem Minyaylov, Magdalena Balazinska, Luis Ceze, Alvin Cheung
Department of Computer Science & Engineering, University of Washington
Seattle, Washington
{bhaynes, artemvm, magda, luisceze, akcheung}@cs.washington.edu

ABSTRACT

We demonstrate VisualCloud, a database management system designed to efficiently ingest, store, and deliver virtual reality (VR) content at scale. VisualCloud targets both live and prerecorded spherical panoramic (a.k.a. 360°) VR videos. It persists content as a multidimensional array that utilizes both dense (e.g., space and time) and sparse (e.g., bit and frame rate) dimensions. VisualCloud uses orientation prediction to reduce data transfer by degrading out-of-view portions of the video. Content delivered through VisualCloud requires 40% less bandwidth than existing methods and scales to many concurrent connections.

This demonstration will allow attendees to view both live and prerecorded VR video content served through VisualCloud. Viewers will be able to dynamically adjust tuning parameters (e.g., bitrates and path prediction) and observe changes in visual fidelity.

1 INTRODUCTION

Recent advances in computing and network hardware have increased interest in immersive 3D virtual reality (VR) applications. Spherical panoramic VR videos (a.k.a. *360° videos*) are one popular example of these applications; other examples include VR games and augmented reality (AR). 360° videos allow a user, through the use of a VR head-mounted display or mobile device (a *headset*), to observe a scene from a fixed position at any angle. The videos are captured using multiple cameras and produced using software that stitches together parts of each frame to produce an approximate (potentially stereoscopic) spherical representation [14]. Devices that support the ability to record and view VR video have become increasingly popular, and efficiently managing this type of data has thus become increasingly important.

Data volume is a major challenge of VR applications, especially in the presence of mobile viewers, which are subject to bandwidth and battery power constraints. Data sizes involved in streaming and storing 360° videos far exceed those seen with ordinary 2D videos. A single frame of uncompressed 2D ultra high-definition (UHD) video at 4K resolution (3840 × 2160 pixels) requires approximately 24MB to store [1]. In contrast, to render UHD 360° video on a headset with a 120° field of view (FOV), we need a much larger frame (~3× higher resolution) since only a portion of the projection is viewed at any time (see Figure 1). Persisting each such frame requires more than 9× space compared to its 2D counterpart. For stereoscopic videos, this requirement doubles!

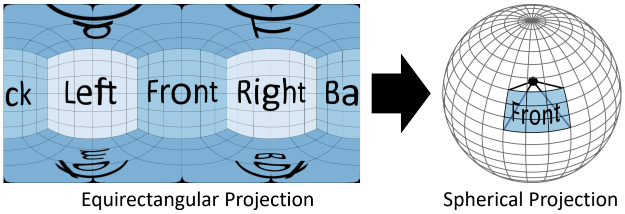


Figure 1: An equirectangular frame encoded within each 360° video (left) and it's projection onto a sphere. Only a small portion of the sphere is within a user's field of view at any time.

Existing streaming VR video platforms (e.g., [12]) treat VR video in the same way as ordinary 2D video, and are thereby poorly-suited for dealing with the massive quantities of data that high-resolution VR video streaming requires. When 360° video is delivered over a network, these approaches reduce bandwidth only in the face of network congestion and do so by sacrificing quality. The approach that they use, called adaptive streaming (e.g., DASH [11]), is illustrated in Figure 2: The server temporally segments an input video and encodes each n-second fragment at various qualities. A client then requests fragments at appropriate quality based on available network capacity. The fragments are concatenated on the client before being rendered. Typical fragment sizes generally fall into the range of one to twenty seconds, and the encoding process may be performed either as a preprocessing step or at time of request (e.g., for live video).

In contrast to the above, our approach is to develop a system that can dramatically cut bandwidth requirements without significant impact on quality. To achieve this goal, we have built a prototype of a new system, called *VisualCloud*. The system's goal is to better support VR applications through existing and novel data management techniques. The current version of the system focuses on 360° videos, but we plan to generalize this in the future. The initial design and prototype further target the effective reduction of bandwidth at the viewer without impacting the immersive experience of 360° videos. Reducing the amount of data streamed to viewers has been shown to reduce both network traffic and also battery consumption on viewers [2].

More specifically, VisualCloud is a new database management system for the storage, retrieval, and streaming of both archived and live VR data. To reduce the amount of data that needs to be transferred to viewers, VisualCloud segments 360° videos both in time and in space. This design is inspired by recent work that demonstrated substantial savings from degrading the out-of-view portions of each frame [13]. Since it is advantageous to deliver out-of-view segments at lower quality, VisualCloud prefetches and prioritizes the spatiotemporal segments that are most likely to be

SIGMOD, 2017
© 2016 ACM. 978-x-xxxx-xxxx-x/YY/MM. . . $15.00
DOI: 10.1145/nnnnnnn.nnnnnnn

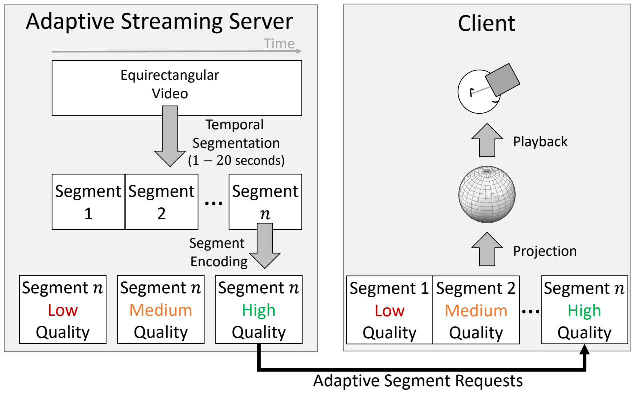


Figure 2: Adaptive streaming in a VR context. On the server, a video is temporally segmented into 1-20 second slices. Each slice is encoded at various qualities. A client then requests segments at a given quality based on network congestion. Segments are concatenated, projected, and rendered on a headset.

viewed. It transfers those segments using the highest resolution and other segments with lower resolutions. Additionally, VisualCloud implements in-memory and near real-time 360° video partitioning and preprocessing to generate multi-resolution data segments and reduce bandwidth utilization even for live streams.

VisualCloud builds on recent work in multidimensional array processing [9] and develops new techniques for VR data storage and retrieval and near real-time in memory processing of VR videos. Our system combines the state of the art in array-oriented systems (e.g., efficient multidimensional array representation, tiling, prefetching) with the ability to apply recently-introduced optimizations by the multimedia (e.g., motion-constrained tile sets) and machine learning communities (e.g., path prediction). VisualCloud reduces bandwidth (and thus also power) consumption on client devices, scales to many concurrent connections, and offers an enhanced viewer experience over congested network connections.

The VisualCloud demonstration will enable attendees to view *both archived and live* 360° video content served through the VisualCloud system. Each participant will begin by specifying VisualCloud tuning parameters (see Figure 5) and then view 360° videos served using those parameters. Participants will be invited to contrast video fidelity for their hand-selected configuration against an expert-tuned configuration and video served using naïve 2D streaming techniques. Attendees will also be able to view the performance advantages afforded by VisualCloud (see Figure 6). Bystanders will be able to view the content delivered to users and bandwidth totals on a separate monitor.

In sum, this demonstration makes the following contributions:

- It presents the architecture of the VisualCloud system.
- It demonstrates that content streamed through VisualCloud is of similar visual fidelity to video transmitted naïvely; in particular, an appropriate set of tuning parameters leads to a viewing experience indistinguishable from naïve delivery.
- It demonstrates that VisualCloud delivers video using approximately 40% less bandwidth when compared to naïve adaptive streaming approaches.
- It demonstrates the above two points on both archived and new streaming data.

2 BACKGROUND

As illustrated in Figure 1, a VR headset accomplishes the illusion of immersion by mapping video data onto a spherical geometry and displaying only the subset of the data that is within the user's current field of view (FOV; typically 90-110°). Video frames are often represented internally in rectangular form; this approach allows for existing 2D video encoding and decoding techniques to be leveraged. To convert spherical video frames into rectangular form, a capture device performs an *equirectangular (ER) projection*; the headset performs the inverse projection prior to rendering.

3 THE VISUALCLOUD SYSTEM

The VisualCloud system contains two major subcomponents – the *VisualCloud Server (VCS)*, which operates on one or more external servers, and the *VisualCloud Client (VCC)*, which runs on a user's headset. The main unit of data exchanged between the VCS and VCC is a video *segment*, which is a spatiotemporal subset of a video encoded at a particular quality level. The VCS is responsible for ingesting live or prerecorded video, spatiotemporally partitioning and encoding it, and serving segments to clients. The VCC, as it receives segments from a server, reconstructs them into a playable video stream and renders it on a headset. These components and their interactions are illustrated in Figure 3.

Both the VCS and VCC contain logic (respectively labeled as the predictor and prefetcher on Figure 3) that serves to identify segments that are likely to be subsequently viewed, based on a user's current position and the behavior of previous users. They also contain a caching layer used to improve performance through the staging of predicted and prefetched segments.

We discuss the VCS and VCC in the following two sections.

3.1 VisualCloud Server Architecture

3.1.1 Storage Manager. There are several challenges associated with building a storage layer for VR video applications. The first challenge lies in choosing the best partitioning of videos into spatiotemporal fragments. The partitioning must ensure that adaptively streaming fragments yields a high-quality experience for end-users while effectively reducing bandwidth. The second challenge lies in laying out the segments on disk taking into account the fact that spatiotemporal segments may be encoded at various qualities to reduce data transfer bandwidth. Finally, for performance reasons it is critical that, where possible, video is stored in a manner that avoids an expensive intermediate decode and encode step.

Addressing these challenges, the VCS storage manager (SM) is responsible for decomposing ingested (potentially live) video and storing it for subsequent delivery to a client. The SM temporally decomposes each video into n-second fragments, and then spatially decomposes each fragment into a grid of video segments. Each three-dimensional segment is then associated with an encoding quality. This decomposition is illustrated in Figure 4.

When persisting each segment, we use the TileDB array-based database system [9], which we extended to support the VP9 [8] codec as an internal compression method. TileDB offers an efficient, low-level API for data represented as a multidimensional array, and allows arrays to be represented using both dense and sparse dimensions. Data tiles are compressed using our integrated video codec and persisted on disk as distinct files.

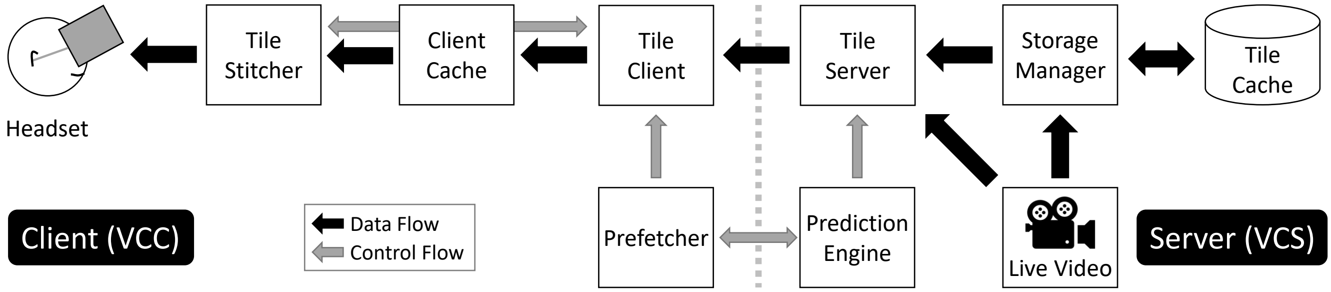


Figure 3: The VisualCloud architecture. The server-based components (VCS) are responsible for ingesting, segmenting, transmitting, and storing video. The VCC requests tiles in prediction-based order, stitches segments together, and displays the result on the headset.

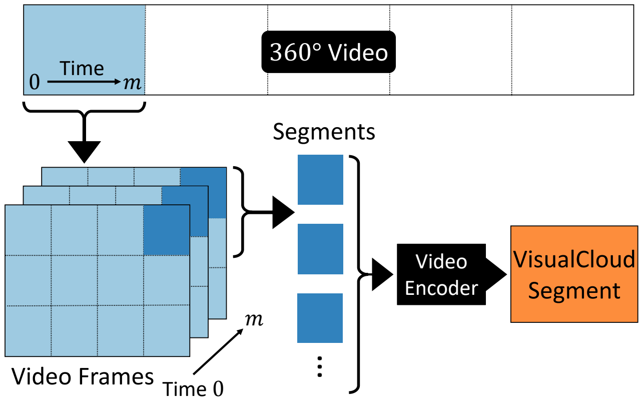


Figure 4: The video spatiotemporal segmentation process, used both in the storage manager and during in-memory stream processing. 360° videos are segmented in time and then spatially decomposed into segments. Each segment is stored as a single encoded fragment.

3.1.2 Prediction. A key challenge for VisualCloud involves predicting which video segments should be downloaded at high quality. For example, downloading all segments at the highest quality wastes bandwidth and risks degraded client performance, while choosing to download important segments at low quality leads to poor user experience. Correctly predicting user behavior to select only those segments a user will actually view is critical for performance.

The VCS prediction component (*predictor*) is used to predict which video segment a user is likely to look at next. VisualCloud uses the model in two ways. First, the predictor applies the model when selecting which segments to degrade in quality (and by how much). Second, the VCC prefectcher (see Section 3.2.1) applies the model to determine the order in which video segments are downloaded, along with what quality to request.

We build on the orientation prediction approach outlined in the Outatime system [7] to make these predictions. Outatime builds a Markov-based model where transition probabilities are applied to estimate a position and orientation at some future time. Outatime makes its predictions based on network round trip time. VisualCloud, on the other hand, predicts farther into the future based on the length of the video segments stored in the SM.

3.1.3 In-Memory Stream Processing. When serving live streams, there is insufficient time to persist video segments on disk through the storage manager. Instead, the VCS performs segmentation in-memory as the data arrives from the capture device. A major challenge here is to create only those segments that will be needed, and deliver those that are not to the SM for subsequent processing. To accomplish this selection task, the VCS leverages the prediction engine and prepares the top-k segments that are most likely to be viewed using the highest quality only and increasingly less likely segments using lower qualities.

Additionally, it is critical that the VCS maintain a sufficiently-large buffer to allow for the various quality levels to be encoded. The VCS ensures this by beginning with a very large initial buffer. As this is exhausted, the VCS drops encoding quality levels until only the lowest quality is prepared. As a last resort, the VCS load-sheds by throttling frames received from the VR capture device.

3.2 VisualCloud Client Architecture

3.2.1 Prefetching. The VCC requests spatiotemporally-segmented video chunks from the VCS at a given quality. There are two challenges associated with this process. First, a client must decide *what order* to use when retrieving the segments – for the best visual experience it is advantageous to prioritize the segments that are likely to be viewed. The client must also decide *what quality* to request when obtaining segments. Segments that are rarely viewed can be safely requested at low quality, while a segment that is likely to be viewed (even if not in the current orientation) should be prioritized. These decisions must be made both for the current time and, when possible, for future periods.

The VCC prefetcher addresses this problem by obtaining the prediction model exposed by the predictor (see Section 3.1.2) and using it to generate a ranking of segments viewed by previous users. It ranks each segment using a method similar to that described in the Forecache system [3], but extends their approach to support the temporal dimension.

3.2.2 Stitching. Once the VCC has obtained a set of segments (at various qualities) for a given unit of time, it must reassemble those segments into an equirectangular video suitable for playback. VisualCloud adopts a technique similar to that described by Zare et al. [13] that allows the segments to be simply and efficiently concatenated. This avoids the need to individually decode each segment and re-encode as a unified video, which is extremely computationally expensive.

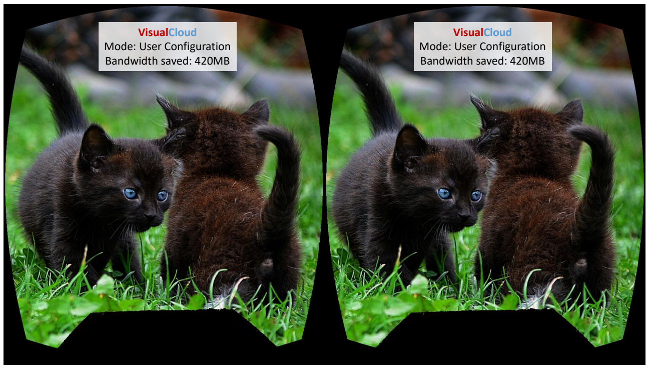
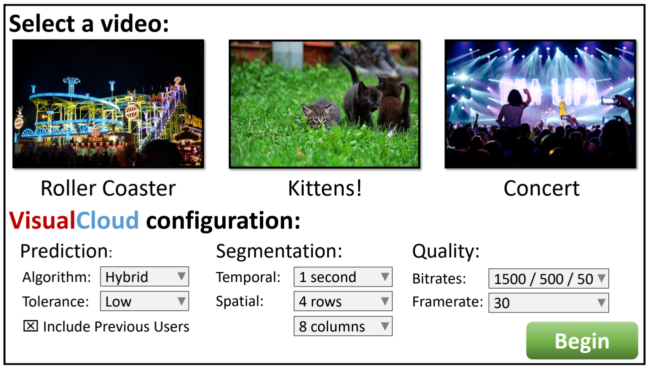


Figure 6: A user's headset view of the 360° video selected and configured in Figure 5. Overlaid is the current bandwidth consumption used to deliver video under the selected configuration.



Select a video:

Roller Coaster

Kittens!

Concert

VisualCloud configuration:

Prediction:
Algorithm: Hybrid
Tolerance: Low
☒ Include Previous Users

Segmentation:
Temporal: 1 second
Spatial: 4 rows
8 columns

Quality:
Bitrates: 1500 / 500 / 50
Framerate: 30

Figure 5: The configuration user interface for the VisualCloud demo. Users select a video to view along with prediction and encoding parameters such as prediction algorithm and segment duration.

4 DEMONSTRATION

Our demonstration will enable attendees to view 360° video content served through VisualCloud at various fidelity levels. Each participant will begin by configuring his or her viewing experience. This involves first selecting what to view; participants may choose either a live feed or from several prerecorded videos. Participants will then adjust VisualCloud tuning parameters, which include prediction algorithm and hyperparameters, spatiotemporal segmentation strategy, and encoding quality. The user interface for the configuration phase is illustrated in Figure 5.

Participants will next don a headset and view 360° video content delivered through VisualCloud. Each video playback session will be segmented into three parts. In the first part, attendees will view the video using their (potentially poorly-selected) configuration parameters. Next, viewers will view video delivered using expert-selected parameters. Finally, viewers will view video delivered using a 2D streaming approach. As illustrated in Figure 6, participants will be able to observe the bandwidth consumed by each method in real-time, and will be invited to contrast the fidelity of each delivery strategy. Bystanders will be able to view the content delivered to users and bandwidth totals on a separate monitor.

VisualCloud will operate the server-side components of the demonstration in the Amazon EC2 cloud; a local server will be available as a backup should on-site bandwidth be too low to adequately support cloud-based delivery. Headsets will be available as part of the demonstration; however, when a user has a mobile device with appropriately advanced hardware, he or she may elect to use that as an alternative by placing it inside a Google cardboard [6] viewer supplied as part of the demonstration.

5 RELATED WORK

Current array database systems [4, 10] are oriented toward data retrieval and analytics and are ill-suited for high-performance real-time streaming. Additionally, array-oriented database systems lack native support for modern video codecs (e.g., [9]) and cannot natively take advantage of the potential optimizations that may be derived through their use.

Some previous systems have explored prediction, prefetching, and speculative execution in a virtual reality context [5, 7]; however, these systems target artificial environments such as those found in games. Other data exploration systems have explored the dynamic prefetching of tiles in a two-dimensional context [3]. Finally, VisualCloud takes advantage of prior work involving bandwidth-reduction techniques for adaptive 360° videos [13].

6 CONCLUSION

In this demonstration, we introduce the VisualCloud system. By representing VR content as a multidimensional array and performing motion prediction, VisualCloud delivers content using less bandwidth, scales to many concurrent connection, and requires less decoding power on client devices. In this demonstration, attendees will observe VisualCloud, tune its parameters, and contrast its visual fidelity with traditional video streaming techniques.

REFERENCES

[1] Parameter values for UHDTV systems for production and international programme exchange. Technical Report BT.2020-2, International Telecommunications Union, October 2015.
[2] M. A. Baker, V. Parameswaran, K. S. Chatha, and B. Li. Power reduction via macroblock prioritization for power aware H.264 video applications. In *CODES+ISSS*, 2008.
[3] L. Battle, R. Chang, and M. Stonebraker. Dynamic prefetching of data tiles for interactive visualization. In *SIGMOD*, 2016.
[4] P. Baumann, A. M. Dumitru, and V. Merticariu. The array database that is not a database: File based array query answering in Rasdaman. In *SSTD*, 2013.
[5] K. Boos, D. Chu, and E. Cuervo. Flashback: Immersive virtual reality on mobile devices via rendering memoization. In *MobiSys*, 2016.
[6] Google cardboard. https://www.google.com/get/cardboard.
[7] K. Lee, D. Chu, E. Cuervo, J. Kopf, Y. Degtyarev, S. Grizan, A. Wolman, and J. Flinn. Outatime: Using speculation to enable low-latency continuous interaction for mobile cloud gaming. In *MobiSys*, 2015.
[8] D. Mukherjee, J. Han, J. Bankoski, R. Bultje, A. Grange, J. Koleszar, P. Wilkins, and Y. Xu. A technical overview of VP9 - the latest open-source video codec. *SMPTE*, 124(1):44–54, 2015.
[9] S. Papadopoulos, K. Datta, S. Madden, and T. G. Mattson. The TileDB array data storage manager. *PVLDB*, 10(4):349–360, 2016.
[10] J. Rogers et al. Overview of SciDB: Large scale array storage, processing and analysis. In *SIGMOD*, 2010.
[11] T. Stockhammer. Dynamic adaptive streaming over HTTP - standards and design principles. In *MMSYS*, 2011.
[12] YouTube. https://www.youtube.com/.
[13] A. Zare, A. Aminlou, M. M. Hannuksela, and M. Gabbouj. HEVC-compliant tile-based streaming of panoramic video for virtual reality applications. In *ACMM*, 2016.
[14] Z. Zhu, G. Xu, E. M. Riseman, and A. R. Hanson. Fast generation of dynamic and multi-resolution 360° panorama from video sequences. In *ICMCS*, 1999.