Hybrid.media: High Velocity Video Ingestion in an In-Memory Scalable Analytical Polystore

Mark Simmons, Daniel Armstrong, Dylan Soderman, Michael Gubanov
Department of Computer Science
University of Texas at San Antonio

***Abstract*—Recent advances in image recognition algorithms are making it possible to query databases for static images which look like a desired target image. This technology is expanding to image recognition within video files as image recognition algorithms advance. Blending the ideas of static image query with video image recognition leads to the subject of our present work. Hybrid.media is our work in progress to ingest video files into an Hybrid In-Memory Polystore Database [Gubanov, 2017], [Podkorytov et al., 2017] and add query support for finding those video files. This work will provide the basis for future work to query videos containing specific items based on content in a video.**

***Keywords*-Video-search; Image recognition; Large-scale Data Management; Data Fusion.**

I. Problem Statement

Implementing a database which is capable of storing and managing various formats of data ranging from audio, text, unstructured or unstructured data, videos, and other assortments has been an ongoing task in the field of Database Management Systems [Gubanov, 2017], [Gubanov et al., 2009], [Gubanov et al., 2011], [Gubanov et al., 2014], [Gubanov and Stonebraker, 2014], [Gubanov et al., 2009], [Gubanov and Pyayt, 2013]. For video files, they have traditionally been difficult to store in databases because they are large and have variable file-sizes, making them not the best fit for standard relational database engines and query languages. In addition, video storage and retrieval is relatively slow because of disk Input/Output (I/O) when not stored in memory. Traditional databases also lack automated content analysis functionality.

A conventional workaround for this is by not actually storing video files but instead storing a file system path to a video file in the database. This practice however is insufficient for video indexing or for inclusion in database backups. Traditional implementations also do not allow querying video files based on image recognition algorithms to find desired content. Absent tagging or other meta-data fields are required in order to provide information about video content.

The current standard practice for storing video in databases does not involve putting in the binary files. Instead, video files are stored on the file-system and a path to that file gets stored in the database [Gilboa-Solomon and Yehudai, 2009], []. The up-side of doing this is that databases do not become bloated with huge files, but the downside is that the content of the video is non-searchable, unless additional meta data is also stored along with each file path. It is also possible to store the binary video file as a binary large object (blob) [Miller et al., 2015] in most modern relational databases. A few reasons to take this approach are to take advantage of transactional commits and rollbacks, preventing orphaned video files, and having backups include video files. However, this is generally viewed as an inefficient use of relational database functionality. In addition, video files differ in size which makes determining the column width difficult to manage. Binary files are more complicated to serve through a web page, and cloud storage also is not an option.

In-memory databases however, have the benefit of having much faster data access than disk-based databases. As RAM prices continue to drop, it is becoming increasingly economical for video and other large data objects to be stored in memory rather than on disk. However, there is a risk of data loss in the event power is interrupted since Random Access Memory (RAM) is typically volatile. Regardless, in-memory databases are growing in popularity with 59 different Hybrid.media applications listed on Wikipedia's list of in Memory Database page, 2017[Wikipedia,].

II. Solution

Because of modern advances in memory technology [Perry,], we can have the contents stored in an in-memory database. The particular in-memory database we use within this project is an hybrid-in-memory polystore database - Hybrid.media, a part of [Gubanov, 2017] which is currently configured with 1 TB of RAM and supports SQL-like queries to access data in a fast, highly-parallelized and scalable fashion. It allows partitioning across multiple machines, which in turn enables multiple nodes to work on data in a fast and efficient manner. This opens the door to future works which will be detailed under the Future Plans section.

We have added an additional feature to Hybrid.media to enable storage of large sets of video data. With the file readily available in memory at all times, any analysis or processing on the video file would skip the file read

Input/Output delay. A scalable, in-memory database engine allows rapid ingestion and analysis of numerous videos simultaneously for enterprise or security purposes. It also has the ability to ingest large quantities of data in the `Load` command. Here we describe our extensions to this load command to work with all types of video file formats. To properly store large amounts of data in memory, we have developed a `ByteArray64` Java object which abstracts the task of storing bytes that would exceed the upper bound limitation of integer array types in Java.

This object abstracts a 64 bit indexed array through use of a two-dimensional byte array where the first index indicates the starting gigabyte and the secondary index indicates which byte within that gigabyte. More formally, for an array A, primary index i and secondary index j the reference A_{ij} refers to byte $(i * 2^{30}) + j$.
This object can be constructed using a `FileInputStream` object and will fill itself with the data from the given file. It can also write itself to a file in the same way. Perhaps the most powerful use of the `ByteArray64` is it's ability to seamlessly return a byte array containing any range of its data that can fit into a single byte array. This allows it to be read in chunks to increase performance.

III. Architecture

1) Video Ingestion: Using a previously implemented `load` command which is used to store csv files in memory, we adjust it with the following syntax in our database shell:

```
"LOAD /<region name>
'<file path>' 'delimiter' 'Mode'"
```

Where we have added an `mode` parameter which specifies a video file, the program checks for the file and reads it. This ingestion will take the binary data of the video file and store it as a single Java object onto a specified region, which acts as a relational data-table.

2) Query Support for Video Files: Retrieval of stored video files from a region is handled in the same way all other select queries are done in Hybrid.media [Gubanov, 2017]. For example, from extracting metadata and storing them in their appropriate attributes, the following query can be done:

```
"SELECT name FROM /region"
```

Where "name" is the title of all videos, taken from the region.

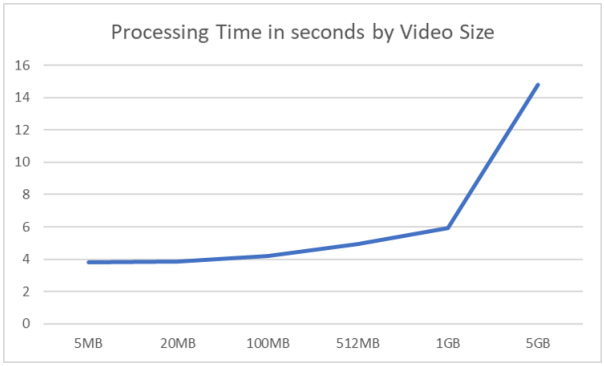
3) Metadata Extraction During Ingestion: In order to use a video's metadata in the database, the database must extract it from the video file. Our project may include logic to extract metadata after reading the video file into memory, before storing it. Video metadata is stored in many formats, differing per file format such as ".mp4", ".avi" and ".mpg." Supporting any file format could be considered a secondary task, though adding more supported formats in the future would be useful.

4) Store metadata along with video: Once metadata about the video has been extracted, the `VidObj` object can be modified to store this additional metadata along with it is current fields, and it can easily be recalled at a later point.

5) Query video based on metadata: With the metadata extracted and stored, our project must then add functionality to account for queries based on pieces of video metadata, not simply on the file metadata created initially to satisfy our primary goals.

IV. Evaluation

Small videos do not vary much in processing velocity, though with larger videos over multiple Gigabytes there is noticeable difference in ingestion speed. This ingestion velocity does not scale linearly with video size, as shown in the averaged trials below.



A. Video Processing Comparison

To test the efficiency of our `VidObj` object for processing videos, we compared it against reading videos directly from a file. We ran several trials reading a sample video of $\sim$ 2.4 Gigabytes, running a simple hashing algorithm on 100 Megabyte chunks of the file and then averaging the milliseconds for several of these trials. All trials were completed on a standard hard disk and DDR3 RAM and the hard drive read cache was flushed between every trial.

As expected, reading the entire file into the `VidObj` object initially and then processing the data took slightly longer than just processing the file directly, though taking less than 7% longer on average. The trials using a previously loaded `VidObj` was processed *significantly faster* - in under 5% of the time it took to process the same video from a file.

V. Future Plans

To further our currently existing implementation of video ingestion, we plan on extracting additional metadata. Users could specify query conditions based on metadata about the

video. This metadata may include resolution, framerate or video format.

Ingestion can work with various forms of data video-formatted information. With future development beyond the scope of this project, the program could apply machine learning algorithms during the ingestion of a video file and store derived data about the video in the database as well.

With video ingestion enabled in Hybrid.media, there are certain processes that are capable of being implemented, especially in context with Hybrid.media at our disposal. Such implementations include image processing, identification, and retrieval through the use of deep-learning algorithms [Wang and Sng, 2015]. The practical use of this is vast, where CPU-intensive jobs that are usually associated with training and predicative information can drastically decreased.

Querying can also involve the use of video-frames [Ide, 2009], or segments of video that resemble other frames contained in an alternate video. With in Hybrid.media and its ability to handle complex analytical queries, it is even feasible to carry out frame-by-frame video analysis. Another computationally-intensive technique that can be done on real-time can be Dynamic Time Warping (DTW) which utilizes "comparison of a query video frame to multiple (usually a fixed number of frames after the frame matched in the previous iteration) reference video frames". Such an application can help for copyright infringement, advertisers, and general video-duplication retrieval.

Video repair [Shao et al., 2011] for incomplete entries on a video-dataset is also a viable path. As data and information becomes more unstructured and varied in format, so too can the potential loss of information contained in said data. Neighboring videos that yield similarity (such as being a part of a series) could also be used to repair associated episodes or sequels to the standard, higher quality exhibited in other videos [Romano et al., 2016]. Training data like this can help resolve the "Single Image Super-Resolution (SISR) problem as described in the aforementioned paper.

Future improvements to video ingestion will also involve additional details derived from the video binary such as file format (e.g. MP4) or other content metadata for use in analysis.

VI. Related Work

Google has a recent test product that performs image search by looking for an image within a video using an image to start with. At the Cloud Next event in San Francisco March 2017, Google unveiled its Cloud Video Intelligence API. The tool, which is currently available to developers in a closed beta, analyzes videos to make their contents searchable. "With the tool, you can search one or more videos using keywords and get back a list of results showing you where in the video you can find the objects relevant to your search terms."[Karissa Bell,].

References

[Gilboa-Solomon and Yehudai, 2009] Gilboa-Solomon, F. and Yehudai, Z. (2009). Virtual video clipping and ranking based on spatio-temporal metadata. US Patent App. 11/946,067.

[Gubanov, 2017] Gubanov, M. (2017). Polyfuse: A large-scale hybrid data fusion system. In *ICDE*.

[Gubanov et al., 2009] Gubanov, M., Popa, L., Ho, H., Pirahesh, H., Chang, J.-Y., and Chen, S.-C. (2009). Ibm ufo repository: Object-oriented data integration. In *VLDB*.

[Gubanov and Pyayt, 2013] Gubanov, M. and Pyayt, A. (2013). Readfast: High-relevance search-engine for big text. In *ACM CIKM*.

[Gubanov et al., 2011] Gubanov, M., Pyayt, A., and Shapiro, L. (2011). Readfast: Browsing large documents through unified famous objects (ufo). In *IRI*.

[Gubanov and Stonebraker, 2014] Gubanov, M. and Stonebraker, M. (2014). Large-scale semantic profile extraction. In *EDBT*.

[Gubanov et al., 2014] Gubanov, M., Stonebraker, M., and Bruckner, D. (2014). Text and structured data fusion in data tamer at scale. In *ICDE*.

[Ide, 2009] Ide, I. (2009). *Video Querying*, pages 3292–3296. Springer US, Boston, MA.

[Karissa Bell,] Karissa Bell, Mashable March 8, . A new google tool actually lets you search videos for specific objects. http://mashable.com/2017/03/08/google-video-intelligence-api/#he42dnXvM8q1/. Accessed November 03, 2017.

[Miller et al., 2015] Miller, M., Boedigheimer, C., Whiteford, D., and Chandrasekaran, A. (2015). Manipulating binary large objects. US Patent 9,112,935.

[Perry,] Perry, M. J. Chart of the day: The falling price of memory. http://www.aei.org/publication/chart-of-the-day-the-falling-price-of-memory/. Accessed November 05, 2017.

[Podkorytov et al., 2017] Podkorytov, M., Soderman, D., and Gubanov, M. (2017). Hybrid.poly: An interactive large-scale in-memory analytical polystore. In *ICDM DSBDA*.

[Romano et al., 2016] Romano, Y., Isidoro, J., and Milanfar, P. (2016). RAISR: rapid and accurate image super resolution. *CoRR*, abs/1606.01299.

[Shao et al., 2011] Shao, L., Zhang, H., Wang, L., and Wang, L. (2011). Repairing imperfect video enhancement algorithms using classification-based trained filters. *Signal, Image and Video Processing*, 5(3):307–313.

[Wang and Sng, 2015] Wang, L. and Sng, D. (2015). Deep learning algorithms with applications to video analytics for a smart city: A survey. *arXiv preprint arXiv:1512.03131*.

[Wikipedia,] Wikipedia. List of in-memory databases. https://en.wikipedia.org/wiki/List_of_in-memory_databases/. Accessed November 04, 2017.