DOI: 10.2352/ISSN.2470-1173.2016.11.IMAWM-462

Face Search in a Big Data System

Qian Lin*, Carlos Ceja**, Meichun Hsu**, Yongqiang Mou*, Min Xu*

**** HP Labs, Palo Alto, CA, USA***

***** Hewlett-Packard Enterprise, Sunnyvale, CA, USA***

Abstract

Big data applications are growing rapidly as more sensors are connected to the internet and gathering business critical information for processing. Imaging sensors are an important type of sensors for collecting images and video data, and are widely deployed on smartphones, video surveillance networks, and robots. Traditional databases are designed to ingest and search structured information. The analysis of unstructured information such as images and videos is often done separately. In this paper, we describe a big data system with deep integration of face analysis and recognition in images and videos. We show how we can utilize the built-in parallelization in the Vertica database system to accelerate feature computation and search. We also show an example application of the system for face re-identification and face search in images and videos.

1. Introduction

High performance database is a core component in a big data system for large scale ingestion and retrieval of data. Such database often supports user defined functions that can be added to the core database engine for running customized analysis of data. We investigated the possibility of utilizing the user defined function mechanism in Vertica database system for integrating the face analysis and recognition in images and videos into the big data system.

Typically, the face recognition system implements the face enrollment, face verification, and face identification in a software application. To expose the functionality to developers, RESTful API is often used [1]. Facial feature data is internal to the software application and is not exposed to the developers. When analyzing a large number of video files or a large number of video streams from a network of surveillance cameras, the software application can be launched on a computer cluster using a parallel processing platform such as Hadoop. The analysis result is then aggregated into a central database server.

We explore the use of Vertica in face recognition system in this paper. Vertica is a Massively Parallel Processing (MPP) platform that distributes its workload over multiple commodity servers using a shared-nothing architecture [2]. While Vertica supports standard SQL, it is in fact also a parallel and distributed computing platform that allows general computation functions to be executed. Vertica allows users to add User-Defined Extensions (UDxs), which runs inside of Vertica in parallel. Using UDx's, non-SQL computation functions can be launched when data is ingested (loaded) or when a query is performed. This enables the workload to run closer to the data and thus is significantly more efficient for execution. But, more importantly, it allows accessing all these extensions using the familiarity of the SQL language and all existing SQL-based tools, and simplifies the end to end system architecture.

2. Face Recognition Implementation in a Database

The face recognition system typically includes a component for enrolling face features into database, and a component for performing identification. First, the system ingests the image or video data. It then uses a series of processing steps to detect faces in the image or video frame, detect facial landmark points, align faces, and extract facial features. The facial features are then projected into a smaller dimensional space, and enrolled as face features. When the system performs identification, it ingests images and videos, and performs similar operations as the enrollment process to extract face features from images or videos. It also loads the face features, and matches the extracted features against those stored in the database. Figure 1 shows the conventional face recognition system workflow.

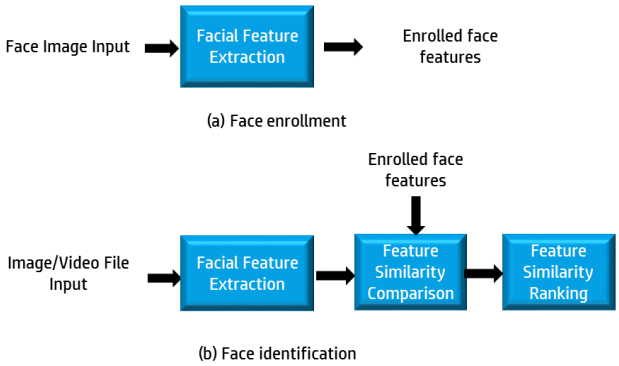

Figure 1: Conventional face recognition system workflow: (a) face enrollment; (b) face identification.

Our objective is to have a simple yet robust implementation of face recognition that can process large number of images in parallel, and store facial features as structured information inside a database. This will provide the flexibility of performing face searches using SQL, and the search can be easily combined with other structured data.

We implement the face recognition function in Vertica database as user defined analytics functions. Rather than performing face search in application, we initiate the workflow from the database, and simplify the application so that it only performs face feature extraction from the image/video input. Based on a query initiated to the database, a user defined function that

©2016 Society for Imaging Science and Technology
DOI: 10.2352/ISSN.2470-1173.2016.11.IMAWM-462

computes the face feature is run. The results, including the locations and times of detected faces, as well as the corresponding face features, are stored in the corresponding fields of the database. When a query is initiated to enroll faces, the face features are calculated and stored in database for enrolled faces. When a query is initiated to identify faces, face features are calculated, and stored in database as unknown faces to be identified. The face features are then compared with those of the enrolled faces using a second user defined function that computes a similarity score between features. The resulting face similarity scores as well as rankings are stored in the database. In practical applications, face features for faces to be identified often have different storage policy as those in the enrollment database. Figure 2 shows the new face recognition system workflow.

Figure 2: New face recognition system workflow.

We describe the processes in more details in the following paragraphs.

(1) Enrollment process:

During the enrollment process, images and videos containing faces to be enrolled are uploaded. While uploading, a user-defined function associated with database is run to detect faces, and to extract face features. Key data fields, including the person name, file name, face bounding box coordinates, and face feature, are stored into the enrollment database.

(2) Verification process:

During the verification process, an image of a person, together with the name of the person, is uploaded. The face feature is extracted, and is compared with the face features with the same name stored in the database. The identity of the person is verified if the similarity score is larger than a threshold.

(3) Identification process:

During the identification process, the images and videos to be evaluated are uploaded. The user-defined function associated with the database is run to detect faces, and to extract face features. The face features along with other analysis results such as face bounding box coordinates are stored in the database. For each incoming face feature, a second user-defined function is run to compare the current face feature with the enrolled features in the database. Top n closest face images can be retrieved and displayed.

(4) Unsupervised identification process:

The system can also be run in unsupervised mode with no separate enrollment/identification stages. This can be used to identify faces that appeared in multiple videos. When the system is running in this mode, faces found in each evaluation video will be enrolled automatically. Each new face in the evaluation video will be enrolled, as well as compared with all previously enrolled faces.

3. Results

We implemented the facial feature extraction and feature comparison user defined functions (UDxs) on Vertica database. We then performed performance analysis to assess the parallelization capabilities using a Vertica cluster. In this section, we first describe the user defined functions that we developed. We then show the performance result. Finally, we show an application of searching faces in video using Vertica.

3.1. User Defined Functions for Face Recognition

The functions of facial feature extraction and feature comparison are as follows:

- Facial feature extraction UDx: it takes image or video as input. Then it performs face detection, face alignment, and facial feature extraction. Finally, it saves analysis result, including facial features, into a Vertica table.

- Facial feature comparison UDx: it takes two features from a Vertica table, and computes a similarity score between the two features.

The images or videos can be placed in Vertica's local file system, or in Vertica's table. Placing them in a Vertica table has the advantage that they will be available to all the nodes in a cluster. In the case that we place images and videos in a local file system, we can use the following SQL command to create tables, and perform feature extraction and feature comparison:

(1) Create media path table, which specifies the media type (image or video) and file location:

```
CREATE TABLE IMG_INFO (mediatype varchar(256),
imgpath varchar(256));
```

(2) Create feature table, which includes the features, as well as face bounding box, facial landmark points, pose (roll, yaw, and pitch angles), and face sharpness:

```
CREATE TABLE FEATURE_EXTRACTION_RET(type
varchar(256), imgpath varchar(256), frame INTEGER,
center_x INTEGER, center_y INTEGER, width INTEGER,
height INTEGER, landmarks VARBINARY(3000),
feature_codes INTEGER,  features VARBINARY(30000),
yaw FLOAT, pitch FLOAT, roll FLOAT, sharpness FLOAT);
```

(3) Insert one media record to table IMG_INFO:

For image type:

```
INSERT INTO IMG_INFO VALUES
('image','/home/dbadmin/workspace/test.JPG');
```

For video type:

```
INSERT INTO IMG_INFO VALUES
('video','/home/dbadmin/workspace/test.avi');
```

DOI: 10.2352/ISSN.2470-1173.2016.11.IMAWM-462

(4) Perform feature extraction:

```
INSERT INTO FEATURE_EXTRACTION_RET(type,
imgpath, frame, center_x, center_y, width, height,
landmarks,feature_codes,features,yaw,pitch,roll,sharpness)
SELECT FacialFeatureExtraction(mediatype, imgpath)
OVER (partition by imgpath) FROM public.IMG_INFO
```

(5) Perform feature comparison:

```
SELECT compareFeature(f1.features, f2.features) FROM
FEA AS f1, FEA AS f2 WHERE  NOT(f1.frame = f2.frame
AND f1.center_x = f2.center_x AND f1.center_y =
f2.center_y);
```

An important consideration for performing data analysis is the location of the data when Vertica database is deployed on a computer cluster. If the image data is located in a local file system, they need to be manually distributed to all the nodes before parallel analysis can happen. We solve this problem by implementing the ingestion of image data into the Vertica tables, so that the image data can be made available for analysis on all Vertica nodes. An additional image loader function is implemented to load the images into Vertica table as follows:

(1) Insert all the images (jpg, jpeg, png, bmp) from specific folder (/tmp/img) into images table:

```
COPY images SOURCE LoadImagesBase64(File='/tmp/img')
DIRECT;
Or Insert one image (1.jpg) image into images table
INSERT INTO images SELECT
LoadImageBase64('/tmp/img/1.jpg') OVER();
```

(2) Insert features by analyzing images in table.

```
INSERT INTO features SELECT
FacialFeatureExtraction(Path, ImageBase64) OVER() FROM
images;
```

where Path is a string with the imgpath (only for reference), and ImageBase64 is the string in base64 (a long varbinary in Vertica).

In the next sub-section, we describe the performance of the face recognition UDxs in a Vertica cluster.

3.2. Parallel Computation Performance

We conducted experiments to test the parallel processing capability of our system. When a large amount of image and video files or streams needs to be processed, feature extraction and feature comparison user defined functions can be launched on multiple CPUs and multiple nodes, and data is stored back in the database. The machines used have Intel(R) Xeon(R) CPU X5680 @ 3.33GHz, and 48GB memory. Three machines have 12 CPUs, and 1 have 24 CPUs. We performed facial feature extraction and face search using face images in the LFW dataset [3]. Table 1 shows our result. We make the following observation.

- Feature extraction time for 13515 faces is 5.808 minutes for 1 node. This translates to about 40 images per second on one node. The time is reduced to 1.535 minutes for 4 nodes. This implies that feature extraction is nearly linearly scalable with the number of computing nodes.

- Feature comparison operation takes much less time. In the "All vs All" case where we ran pair-wise comparisons among all the extracted faces in the dataset, which amounts to over 180 million comparison operations, it takes about 4 hours on one node, which translates into about 12,500 comparisons per second on one node. When we ran the same task on 4 nodes, it takes about one hour. This implies that feature comparison operation is also nearly linearly scalable with the number of computing nodes. Furthermore, feature comparison operations combined with top-20 selection in the "All vs All" case also validates the scalability of the multi-node implementation in accommodating the additional sorting required.

- When the number of comparison operations is small, as illustrated in the "1 vs All" cases, the multi-node implementation is not any faster than a one-node implementation due to the small amount of computation needed in this case. In our next step we will explore Vertica streaming to enable fast feature comparisons with large number of concurrent streaming input. To further optimize the scale of comparisons as the number of images in the database increases, we will explore high-dimensional indexing techniques in Vertica as well.

3.3. Example Application: Person Re-Identification and Face Search in Video

We can apply the face search in database system to the person re-identification problem. Suppose that we want to identify whether a person is observed multiple times by a network of cameras. This can help map the trajectory of the person's movement. We can process the video stream from each camera, and ingest the analysis data including face features into the database. To find the matches, a separate user defined function can be run in parallel to do face comparisons. This makes the computation more efficient.

As an example, we analyzed 18 sample videos in the NICTA ChokePoint dataset [1] with multiple people appearing in the videos. We detected a total of 17649 faces in the video frames and generated the corresponding face features. We can get instantaneous response in query time for top 10 most similar face features. Figure 3 shows our demo user interface for face re-identification using the ChokePoint dataset. Panel 1 shows a list of videos that we processed. Panel 2 shows the video player. Panel 3 shows the face images in a video selected from the video list in Panel 1. Panel 4 shows the most similar faces in other videos when a face image in Panel 3 is selected. When a face in Panel 4 is selected, the corresponding video segment during which the person appears is played in the video player in Panel 2.

For a large camera network observing people over a period of days or in crowded public areas, the number of faces detected in videos can easily be in the order of millions. The new approach scales up to large face numbers very well.

DOI: 10.2352/ISSN.2470-1173.2016.11.IMAWM-462

Our system has the following advantages:

(1) Unified still image face recognition and video face recognition: Our system can handle both face enrollment in images and videos, and face identification in images and videos work consistently.

(2) Flexible search: Searching face features in the enrollment database is run by a user-defined function in the database. Since the search is now supported using SQL, we can provide much more flexibility in search. For example, we can combine face search with other database fields such as location to yield similar faces in a certain location.

(3) Better scalability: By taking advantage of the parallelization in the database cluster, the face enrollment and identification can be parallelized using multiple instances of the database nodes. This can shorten the amount of time needed to process a large number of images and videos.

Name of Operation	Type of Operation	Total # of operations	1 Node Time (minutes)	4 Nodes Time (minutes)
Features and Insertion	PARTITION BY image path	13,515	5.808	1.535
Comparison	1 vs All	13,514	0.022	0.041
	All vs All	182,641,710	243.859	60.755
Top 20	1 vs All	13514	0.042	0.044
	All vs All	270,300	308.012	64.775

Table 1: The processing times of feature computation, comparison, and ranking for 1 Vertica node and 4 Vertica nodes.

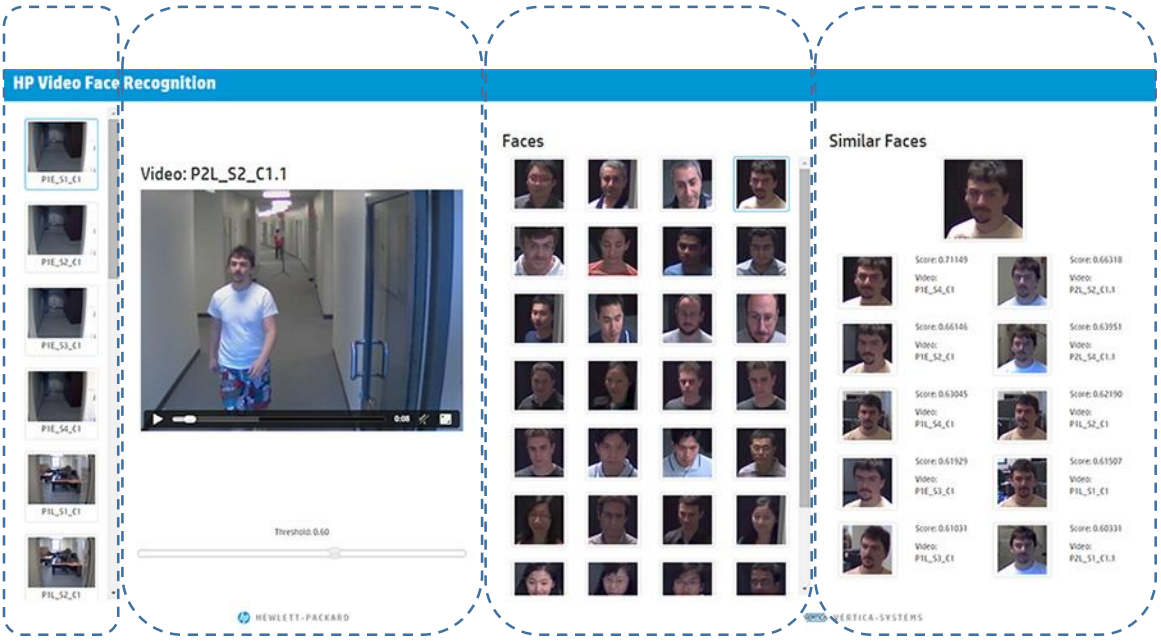

Figure 3: Person re-identification demo user interface using Vertica face search.

DOI: 10.2352/ISSN.2470-1173.2016.11.IMAWM-462

4. Conclusions

We describe a system that implements unified still image and video face recognition in a Massively-Parallel Processing (MPP) database - Vertica. The computation for detecting faces and extract face features is implemented as a user-defined function of the database. The facial feature comparison is also implemented as a user defined function. As a result, a user can issue SQL queries to do face verification and identification, as well as complex search commands by using additional attributes in database. Scalability to process a large number of images and videos is solved using the parallel processing functions in the database.

5. References

[1] Peng Wu, Rares Vernica, Qian Lin, "Cloud based multimedia analytic platform", Proceedings of the 21st ACM international conference on Multimedia, 2013.

[2] A. Lamb, et al, "The Vertica Analytic Database: CStore 7 Years Later", Proc. VLDB, 2012.

[3] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller, "Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments", University of Massachusetts, Amherst, Technical Report 07-49, October, 2007.

[4] Y. Wong, S. Chen, S. Mau, C. Sanderson, B.C. Lovell, "Patch-based Probabilistic Image Quality Assessment for Face Selection and Improved Video-based Face Recognition", IEEE Biometrics Workshop, Computer Vision and Pattern Recognition (CVPR) Workshops, pages 81-88. IEEE, June 2011.

6. Author Biography

Dr. Qian Lin received her BS from Xi'an Jiaotong University in China, her MSEE from Purdue University, and her Ph.D. in Electrical Engineering from Stanford University She is currently a research director working on computer vision and deep learning research at HP Labs. Dr. Lin is inventor/co-inventor for 44 issued patents. She was awarded Fellowship by the Society of Imaging Science and Technology (IS&T) in 2012.

Carlos worked at HP Labs developing prototypes for information visualization and currently collaborates with Advanced R&D for Big Data Platform at Hewlett Packard Enterprise on HPE Vertica. At present is studying his Master Degree in Computer Systems Engineering at ITESO.

Mei is Director of Advanced R&D for Big Data Platform at Hewlett Packard Enterprise, where she is leading development of scalable analytics on HPE Vertica. Prior to her current role, she was Director of Intelligent Information Management Lab at HP Labs where she conducted research in the areas of scalable data management, business intelligence, and information visualization. Before joining industrial R&D, Mei was a professor of computer science at Harvard University.

Yongqiang received his Master Degree in Signal and Information Processing from Xi'an University of Technology at 2012. He is currently working as a R&D engineer at HP Labs. His research interests include computer vision, machine learning, and deep learning, focusing on face analysis, object tracking, large scale image cluster.

Min received his BS and Master from Central South University in China. He is currently a research and development engineer working on computer vision and machine learning at HP Labs.