FOUNDATIONS OF COMPUTING AND DECISION SCIENCES
Vol. 37 (2012) No. 2

DOI: 10.2478/v10209-011-0007-8

CONTENT-BASED LOAD SHEDDING IN MULTIMEDIA DATA STREAM MANAGEMENT SYSTEM

Rafal MAISON*, Maciej ZAKRZEWICZ†

Abstract. Overload management has become very important in public safety systems that analyse high performance multimedia data streams, especially in the case of detection of terrorist and criminal dangers. Efficient overload management improves the accuracy of automatic identification of persons suspected of terrorist or criminal activity without requiring interaction with them. We argue that in order to improve the quality of multimedia data stream processing in the public safety arena, the innovative concept of a Multimedia Data Stream Management System (MMDSMS) using load-shedding techniques should be introduced into the infrastructure to monitor and optimize the execution of multimedia data stream queries. In this paper, we present a novel content-centered load shedding framework, based on searching and matching algorithms, for analysing video tuples arriving within multimedia data streams. The framework tracks and registers all symptoms of overload, and either prevents overload before it occurs, or minimizes its effects. We have extended our Continuous Query Language (CQL) syntax to enable this load shedding technique. The effectiveness of the framework has been verified using both artificial and real data video streams collected from monitoring devices.

Keywords: data streams, load shedding, data stream management system, overload management.

1. Introduction

Aftermath of the September 11 attacks, efforts to spot potential terrorist or criminal activities have widely intensified. This led to increased interest in systems able to identify suspects in public places such as airports, railway stations, stadiums and subways. Automatic face recognition of these people is the most natural way to determine identity. In practice, visual recognition is more effective than using biometric sensors, because identification can be performed without the suspicious person being aware of the

* Poznan University of Technology, Institute of Computing Science, ul. Piotrowo 2, 60- 965 Poznan, Poland, e-mail: rafal.maison@cs.put.poznan.pl

† Poznan University of Technology, Institute of Computing Science, ul. Piotrowo 2, 60- 965 Poznan, Poland, e-mail: maciej.zakrzewicz@cs.put.poznan.pl

verification. Biometric feature such as a retinal pattern, fingerprint, or DNA sample needs physical interaction with the monitored target to gather.

Facial recognition included the following phases: detection, tracking and classification [5][24][31]. In addition to recognizing individuals suspected of nefarious activities, systems implementing these steps seek to decrease the number of false positives for persons who are not real risks.

We conclude that a very important capability of these data systems is the continuous processing of information from multiple data streams, such as video feeds, rather than from scalar data stored in relational databases. We argue that by running monitoring queries over multimedia data streams grabbed from devices installed in selected public places, security agencies can continuously uncover individuals suspected of nefarious activities in more efficient and productive ways [20]. We claim that in order to increase the efficiency of multimedia data stream processing in the public safety sector, the novel concept of a Multimedia Data Stream Management System (MMDSMS) using load-shedding techniques should be introduced into the monitoring infrastructure to improve the execution of multimedia data stream queries. Figure 1 presents such a monitoring architecture at an airport.

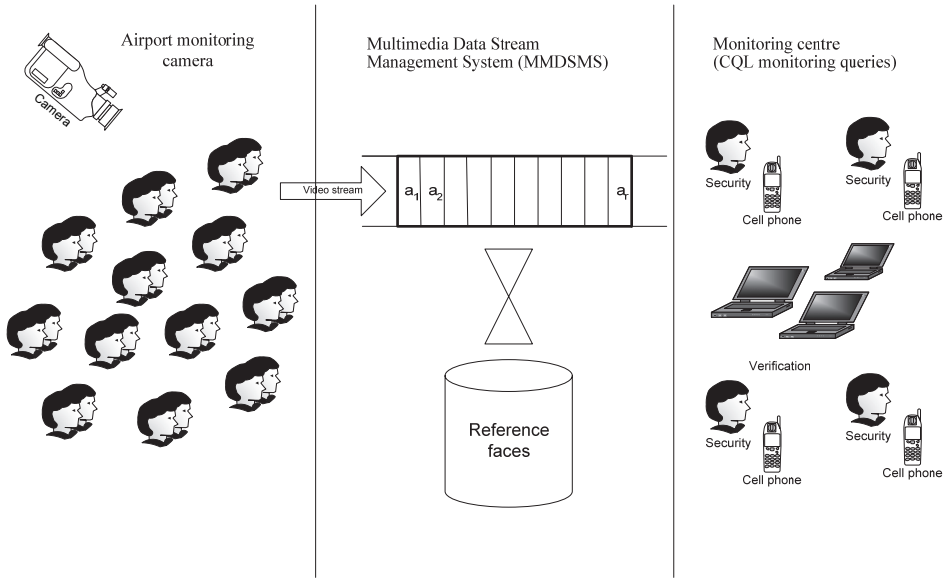

Figure 1. Public safety infrastructure continuously searching for individuals suspected of nefarious activities

We propose the following data model for representing video tuples streamed from multimedia devices:

- STREAM_ID – unique stream identifier
- FEED_NAME – a label for the processed stream
- FRAME – a video frame captured from the high resolution camera
- DATA_TYPE – the multimedia data type

- TIME – the time of video frame arrival

A set of reference faces is stored as multimedia picture instances in Multimedia Databases. A typical video tuple consists of several fields including:

- PICTURE_ID – an unique image identifier
- PICTURE_DESCRIPTION – a label for the image
- PICTURE_DATA – a face picture
- DATA_TYPE – the multimedia data type

Given these elements, the schema of a video stream grabbed at an airport can be formally expressed in Multimedia Continuous Query Language (MCQL) as:

CREATE STREAM airport_cam (
integer STREAM_ID,
string FEED_NAME,
picture FRAME,
string DATA_TYPE,
timestamp TIME); (2)

and the schema of the face reference relation as:

CREATE TABLE faces (
string PICTURE_ID,
string PICTURE_DESCRIPTION,
image PICTURE_DATA,
string DATA_TYPE); (3)

Furthermore, the following MCQL query involving the similarity-based operator *SIMILAR WITH* [20] can be employed to track particular persons appearing in the video stream and identify those who are potentially suspected of nefarious activities by comparing their images against a reference faces stored in a multimedia database:

SELECT f. picture_description
FROM airport_cam ac, faces f
WHERE f.picture_data
SIMILAR ac.frame WITH 0.6; (4)

The *SIMILAR WITH* operator is expressed as:

picture1 SIMILAR picture2 WITH probability

where

picture1 – first picture used for similarity comparison
picture2 – second picture used for similarity comparison
probability – the probability of similarity between the two compared pictures

The monitoring query above is executed on a stream of transient multimedia tuples. When several multimedia streams are processed, the time needed for face recognition increases and the video frame analysis rate is decreased. The average number of identified faces reduced as resource contention grew, because the query engine evaluated join operations for each streamed frame even though the arrival rate was higher than the ability to process would allow. Consequently, when overload occurs, certain tuples can not be processed in time, leading to inaccurate query results. In order to manage accuracy under such circumstances, queries should consciously ignore some tuples so that monitoring query results are either not impacted or minimally impacted. Algorithms to discern those "less important" tuples are called load-shedding algorithms [28][29][30].

In order to increase accuracy of automatic human recognition in our query language, a new clause has been introduced to employ our content-based load-shedder in times of overload; namely, *on overload content_analysis*. The video stream can be defined in MCQL as:

```
CREATE STREAM airport_cam (
      integer STREAM_ID,
      string FEED_NAME,
      image FRAME,
      string DATA_TYPE,
      timestamp TIME)
ON OVERLOAD CONTENT_ANALYSIS;   (5)
```

In this paper, we proposed a novel load shedding algorithm, based on multimedia tuple content analysis and based on the prediction-based approach [21]. Our approach significantly improves the accuracy of multimedia data stream query results by means of a searching and matching algorithm to extract key frames from the video stream during result evaluation. The method tracks and registers all symptoms of overload, and either prevents overload before it occurs or reduces its effects.

2. Related Work

In order to generalize solutions to data stream query processing problems, the concept of a data stream management system was introduced [1][6][11] and a number of DSMSs have been developed. The Aurora model [1] deals with overload by using two load-shedding techniques to drop tuples. The first technique mainly relies on the creation and analysis of quality of service (QoS) graphs based on tuple processing delay and drop percentage to determine appropriate shedding rates for those portions of data streams that are most tolerant to drops. The second technique, called semantic load shedding, analyzes value-based QoS graphs which assign importance to data tuples based on administrator-defined criteria. Wherever possible, less relevant tuples (those believed not to influence the query accuracy) are dropped.

STREAM [6][23] introduced the concept of query plans based on load shedding operators and relationships between tuples that vary over time. The query answer degradation minimization can be transformed into an optimization problem [8].

TelegraphCQ [62] has data triage [25][26] to deal with load shedding in an adaptive manner. The main idea is to evaluate precise query answers for tuples that can be handled within a specified time window and to summarize the characteristics of any excess, less-important tuples for which the query engine lacks processing resources. The summarization technique provides compression functions, which condense arriving tuples into compact representational sets of data. It also provides computational operators and rendering methods that analyze these summaries and approximate the sets of results that would have been generated by precise query processing [26]. When overload occurs, the data triage framework considers specified constraints on tuple delivery time delay to determine how many tuples to send to the summarizer, as well as which tuples to send.

Other techniques are focused on controlling data delay QoS [65], or on using a feedback control loop to continuously monitor data stream management system state [15]. In the latter, information about the DSMS processing rate is transmitted to a controller and compared against reference values. The results of the comparison are passed to a load shedder so that data drop rules may be altered as necessary to bring the two in closer alignment. By dropping excess data, load-shedding techniques can significantly influence the accuracy of queries.

A DSMS that does not handle overload situations may become unstable and cause uncontrolled delays during result evaluation. Our solution addresses a number of major problems, including detecting the point in time at which overload occurs, deciding how many tuples should be dropped, and predicting which tuples are unimportant to the result. Histograms [16][17], wavelets [10][22][33], and samples [2][3][7][14] may be employed to store dropped tuples.

In [21] the prediction-based approach has been proposed, which significantly improves the accuracy of data stream query results by using several components: a Statistics Manager, a Prediction Module, and a Load Controller. For tuples arriving at the query engine, the Statistics Manager computes statistics such as selectivity, variance, mean for all tuples and mean for groups of tuples. Later, the Prediction Module predicts which tuples do (and do not) influence the query results. It monitors queries that are being evaluated by the query engine and generates tuple-dropping rules using syntax analysis (in the case of simple filtering queries) or statistical analysis (for aggregates). The dropping rules help minimize result degradation and ensure system stability. The Load Controller continuously monitors the state of the query engine and its input queues and adjusts the tuple-dropping rate based on rules generated by the Prediction module.

In order to effectively process dynamic multimedia data streams and continuous queries, Multimedia Data Stream Management Systems (MMDSMS) have been recently introduced in a computation-oriented multimedia data stream model [12]. The recent research does not propose any load shedding algorithm.

The rest of the paper is organized as follows. First, we describe the essential components of our framework. Next, we present analysis of our new content-based load shedding algorithm, key frame extraction, and multimedia continuous query execution using the similarity based join [20]. Finally, we provide the results of our experiments.

3. Architecture of the Content-based Load Shedder

We designed our module to run on top of the prediction-based load shedding framework [21], and added new components to its architecture which support multimedia feeds [20]. These new components enhance the existing load shedding technique and incorporate a new multimedia data extractor into the query engine; however, the existing Prediction Module and Load Controller remain unchanged. The functional description of existing components has been presented in previous work [21]. Our content-based load shedder consists of the following components, as illustrated in Figure 2:

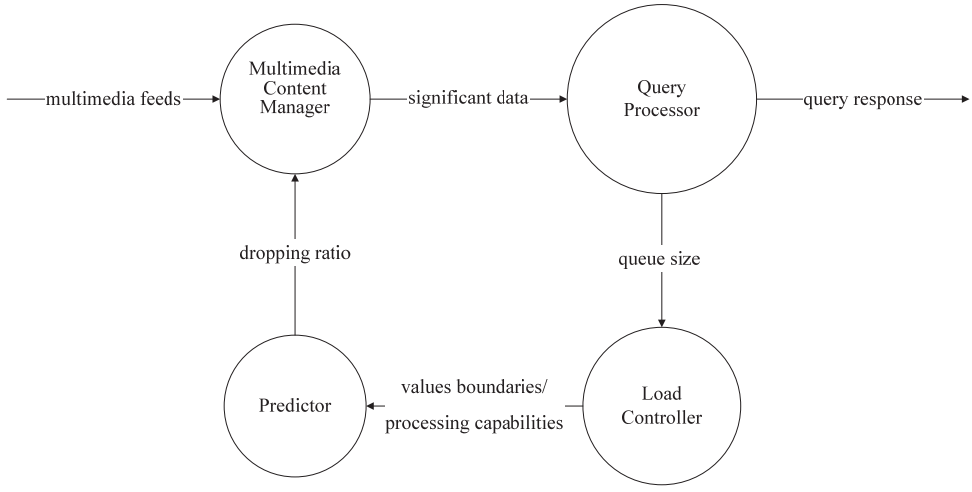

Figure 2. Content-based load shedding architecture

- Multimedia Content Manager - Computes similarity between frames to detect whether there is a change in scenery or not. Afterwards it delivers extracted key frames to the query engine for processing (e.g. face recognition). When the similarity between two compared multimedia tuples exceeds the tuple-dropping threshold level ρ, the new tuple is discarded. The latest value of ρ is evaluated by the Prediction Module.

- Prediction Module - Predicts which tuples do (and do not) influence the query results. The Prediction Module monitors queries that are being evaluated by the query engine and generates the tuple-dropping threshold level ρ mentioned above based on the analysis of incoming video queries.

- Load Controller - Continuously monitors the state of the query engine and its input queues. The number of tuples buffered in the input queues is passed to the Load Controller to be compared against a reference level. If they are different, the

MMDSMS will adjust the tuple-dropping rate based on threshold level computed by the Prediction Module.

3.1. Multimedia Content Manager

The multimedia content manager computes similarity between frames to detect whether there is a change in scenery or not. The less-similar frames are called key-frames and provide representative content for a given sequence within a video feed. The problem of key frame extraction has been widely discussed and plenty of algorithms have been proposed [13][19][4][9][35]. In processing transient multimedia data streams in the Multimedia Data Stream Management System (MMDSMS), the fundamental goals for the selection process are effectiveness and low computational expense. In order to accelerate key frame extraction, the visual content-based approach has been employed; however, discussion of the exact extraction algorithm used is not within the scope of this paper.

The similarity between two video tuples is defined as the difference of their visual content. Examples of visual content include color, shape and texture. For given frames of i width by j height, the correspondence of their visual content is denoted as the difference of their color histograms in the HSV space. For a multimedia data stream defined as $s = \{f_1, f_2, f_3, ..., f_n\}$, similarity can be expressed as a cost comparison of two consecutive images [37]:

$$C_e = \sum_{i=0}^{width} \sum_{j=0}^{height} \min(Hist_a(i,j), Hist_b(i,j)) \quad (1)$$

The similarity assessment is made on the basis of a threshold ρ, allowing control of the density of the comparison. Increasing the value of the ρ parameter means that compared frames can be more similar and still meet the qualification of being key frames. By analyzing the application of this method in the MMDSMS, it can be observed that the greater the value of ρ, the higher the number of multimedia tuples that must be processed by the query engine. An example of this processing might be face or shape recognition. Table 1 shows the differences in key frame extraction using various values of ρ against a 5-minute video feed with a 30 frame/second rate. The overall number of frames in this feed is 9000.

Table 1. The number of key frames out of 9000 in the 5-minute video feed

Density	Number of K-frames
ρ = 0.60	673
ρ = 0.85	823
ρ = 0.90	1284
ρ = 0.95	2033

In the case of overload, the ρ threshold can be decreased so that the query engine processes a smaller number of tuples while still maintaining an acceptable level of accuracy. This parameter ρ is more commonly called the tuple-dropping threshold level. Its value will be calculated by the Prediction Module based on information provided by the Load Controller. Since the process of face recognition is greater than an order of magnitude more computationally complex than the extraction of key frames, the MMDSMS can achieve much greater performance during overload by processing only key frames instead of all tuples. We call the algorithm for performing this optimization content-based load shedding.

3.2. Cost of Processing a Multimedia Query

In contrast to traditional DSMSs where simple operators can be used to compare data, our MMDSMS employs video search and pattern matching algorithms to find matches. Figure 1 depicts the scenario of joining multimedia streams together. In order to calculate the monitoring query (4) result, which searches for individuals who potentially pose a threat, a full join is performed. The join operation between two video streams, A and B, can be realized as two separate tasks: a single join operation of A frames to B frames and a single join operation of B frames to A frames. In the first step, the join operator takes each A stream frame $\{a_1, a_2, \ldots, a_r\}$ as it comes into the system and searches the B stream sliding window tuples for matching pictures. Let us denote this cost as C_{SB}. Meanwhile this search is being executed, B stream frames are also arriving the system and the B stream sliding window must be updated to include the new frames and to delete any tuples that have fallen outside the window. Let us denote the cost of updating the window for each incoming B frame as C_{UB}. This means that for all A stream frames r_A and B stream frames r_B arriving in a given period of time, the A to B single join cost C_A at this time can be expressed as:

$$C_A = r_A \times C_{SB} + r_B \times C_{UB}$$

Similarly, the B to A single join cost in the same period of time can be defined as:

$$C_B = r_B \times C_{SA} + r_A \times C_{UA}$$

Putting it all together, the cost for full join during this time period is denoted as:

$$C_j = (r_A \times C_{SB} + r_B \times C_{UB}) + (r_B \times C_{SA} + r_A \times C_{UA})$$

Within proposed MMDSMS, we have implemented two join techniques that can be utilized when combining multimedia data streams: nested loop join (NLJ) and hash join (HJ). The NLJ approach for single A-to-B joins, involves comparing each arriving A stream frame with each frame in stream B's sliding window. This C_{SB} cost can be denoted as the number of frames in the sliding window (henceforth defined as N_B) multiplied by cost of comparing two frames (henceforth expressed C_f), or:

$$C_{SB} = N_B \times C_f$$

The cost of comparing two frames C_f can be represented as:

$$C_f = \begin{cases} C_a + C_e + C_m & key-frame \\ C_a + C_e & otherwise \end{cases}$$

where C_a is the cost of a sliding window access, C_e is the cost of comparison of two subsequent images to obtain similarity and C_m is the cost of search and pattern matching algorithms utilized to find matches between two frames. Additionally, since there is nothing special utilized in updating B's sliding window other than adding frames to and deleting frames from a queue, the C_{UB} cost can be expressed as 2 times C_a . Putting it all together, the cost of a single NLJ join from stream A to stream B is defined as:

$$C_A = r_A \times N_B \times C_f + r_B \times 2 \times C_a$$

3.3. Density of the parameter ρ

Monitoring the multimedia tuples entering the MMDSMS allows us to track the resources used to process a multimedia data stream. For a given sliding window, the time remaining in the window as $t_r = t_e - t_c$, where t_e is the sliding window end time and t_c is the current time. The MMDSMS uses all its resources when the total number of multimedia tuples entering the system in the remaining time t_r is equal to or greater than the remaining processing capability N_r . Multimedia tuples added to the input queue also may not be processed by the query engine in time; therefore, N_r is additionally decreased by the number of multimedia tuples enqueued in the input queues. Based on [21], N_r can be expressed as follows:

$$N_r = \frac{t_r}{C_f} - N_q \ ,$$

where N_q defines the number of multimedia tuples in the input queue, t_r defines time remaining in the window and the cost of comparing two frames C_f can be denoted as:

$$C_f = \begin{cases} C_a + C_e + C_m & key-frame \\ C_a + C_e & otherwise \end{cases}$$

where C_a is the cost of a sliding window access, C_e is the cost of comparison of two consecutive images to obtain similarity and C_m is the cost of search and pattern matching algorithms used for image processing such as face or shape recognition.

In order to detect overload, the Load Controller continuously monitors multimedia tuple arrival over short periods of time. Let t_b define the time to receive N_b tuples. The assumed rate of tuples for the remaining part of the window is denoted as $R_b = N_b / t_b$. Correspondingly, the expected number of multimedia tuples entering at the MMDSMS by the end of the current window is expressed as $N_e = t_r * R_b$ .Thus, the Load Controller should shed load when $N_r < N_e$. Let N_ρ express the number of multimedia tuples exceeding the current window processing capability: $N_\rho = N_e - N_r$. The threshold ρ for a given query plan can be defined as $\rho = N_\rho / N_\rho$.

In cases where the MMDSMS is not capable to process tuples for most of the sliding window time, the excess multimedia tuples are enqueued and must be discarded when the window ends in order to fulfill result computation criteria. Notice that, from the point of view of the final answer, the shedder drops both useless and useful tuples. The goal of our content-based load shedding is to discard those multimedia tuples that will affect the final query answer the least.

3.4. Determining the dropping ratio

The Load Controller continuously tracks the threshold ρ using a feedback mechanism and checks whether the number of enqueued tuples N_q is rising. If it is, the Load Controller automatically adjusts the threshold ρ by reducing the capacity of the remaining window N_r . The global policy for the Load Controller is expressed as follows:

$$m(N_q) = \min\{t_r * r_B * \rho\}$$

Since a full join operation between two video streams, A and B, can be performed as two separate tasks – a single join operation of A frames to B frames and a single join operation of B frames to A frames – the global policy is defined as follows:

$$m(N_q) = \min\{(r_A \times N_B \times C_f + r_B \times 2 \times C_a) \times \rho\}$$

At any point of time during the current sliding window, the Load Controller may adjust the threshold ρ to limit the number of unprocessed tuples. Whenever the tuple rate is changing and $N_r < N_e$ has been violated, some of the arriving data will be summarized to fulfill $m(N_q)$.

When the determination is performed to start shedding load, the Prediction Module has to decide whether to drop each multimedia tuple or to forward it to the query processor for further evaluation. From the beginning of the sliding window *W*, the volume of tuples entering the system continues to grow. The Load Controller tracks the query processor and input queues and computes the value of ρ if overload occurs. When a new multimedia tuple enters the system, its visual content similarity to the previous tuple is evaluated based on (1) and compared with ρ. If the value is greater than the threshold, then it falls into the same category as the previous tuple [37] and can be discarded.

4. Experimental Results

The proposed approach has been verified against the real-world problem in public safety. Our MMDSMS has been deployed on hosts configured with two 2.2 GHz Intel Core*2 Duo central processing units (CPUs) and 3.0 GB of random access memory (RAM). Our tests have been performed against video from a high-density camera commonly used in the public safety sector. We have populated our reference face database using a set of sample faces from the AT&T Laboratories Cambridge Database of Faces. In order to activate the proposed content-based algorithm and improve accuracy of automatic human identification, the new *on overload content_analysis* has been implemented.

Our MMDSMS first extracts facial information from the video frames and renders it into a normalized format before comparing it against a reference set of faces. This automated face recognition process consists of several phases such as: video tuple frame segmentation, which leads to the separation of "face-like" regions within a grabbed image; facial feature extraction from these regions; and face identification on the basis of extracted features. Since a "face" can change depending on angle of view, scene lighting, and a person's mood (e.g., gloomy, cheerful), therefore facial feature extraction is a challenging problem. In order to generalize issues related to face detection/recognition, numerous algorithms have been proposed [34][36][27][18]. Our MMDSMS uses the principal component analysis (PCA) with eigenfaces algorithm to provide a video tuple matching similarity-based operator for join operations. It also employs the Viola-Jones method based

on Haar cascade face detection and appropriate face cascade classifiers for facial feature detection [32].

In order to measure multimedia data stream query processing accuracy, partial results of the query in progress are analyzed. The query accuracy denotes the effectiveness of the proposed algorithm and is measured by root mean square error (RMSE), expressed as follows:

$$RMSE = \sqrt{\frac{\sum_{i=o}^{j} \left(\left| N[i]_{exact} - N[i]_{meas} \right| \right)^2}{j}},$$

where N_{meas} denotes the number of faces recognized by the tested algorithm and N_{exact} denotes the exact number of faces carried by multimedia tuples.

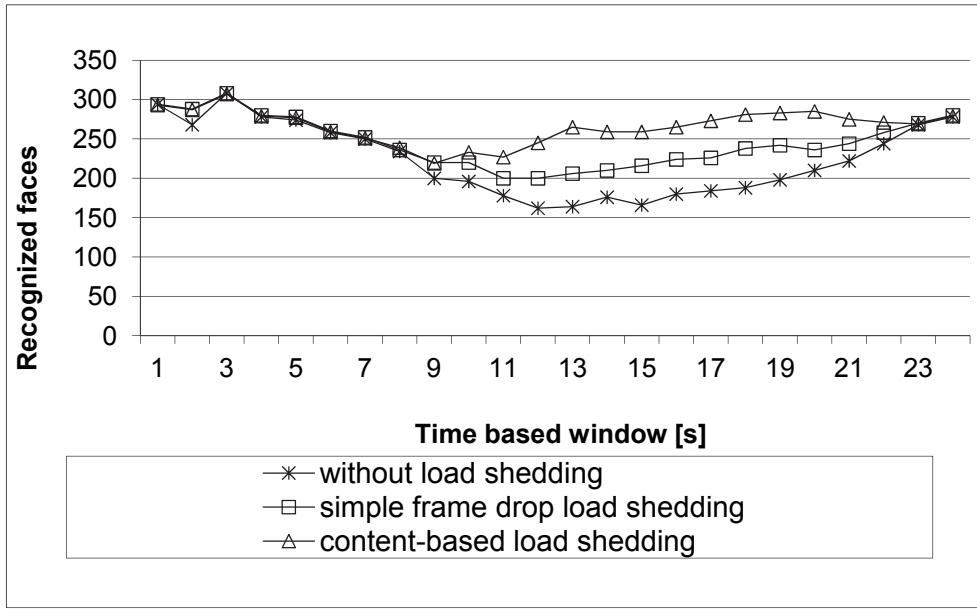

Figure 3. Average number of recognized faces for the registered joins

The error metric above has been evaluated for MMDSMS systems without load shedding, with simple frame drop load shedding, and with content-based load shedding. We first analyzed query time response under large overload. Next, we evaluated the query results for our monitoring scenario (4). The experiments showed that our content-based approach generated fewer errors for the monitoring query.

When multiple multimedia streams are processed, the time required for face identification increases and the video frame analysis rate is reduced. In our experiments, test runs of twelve nested loop joins between multimedia data stream frames and static reference pictures have been processed by the query engine simultaneously. Each test run has been executed three times, once with simple frame drop load shedding, once with content-based load shedding, and once without load shedding. Figure 3 above shows comparative graphs of these tests.

In the non-load-shedding case, overload started after the 8th second and ended around the 23rd second. The average number of recognized faces decreased as resource contention grew, because the query engine performed join operations for each arriving frame even though the arrival rate was higher than processing capabilities would allow.

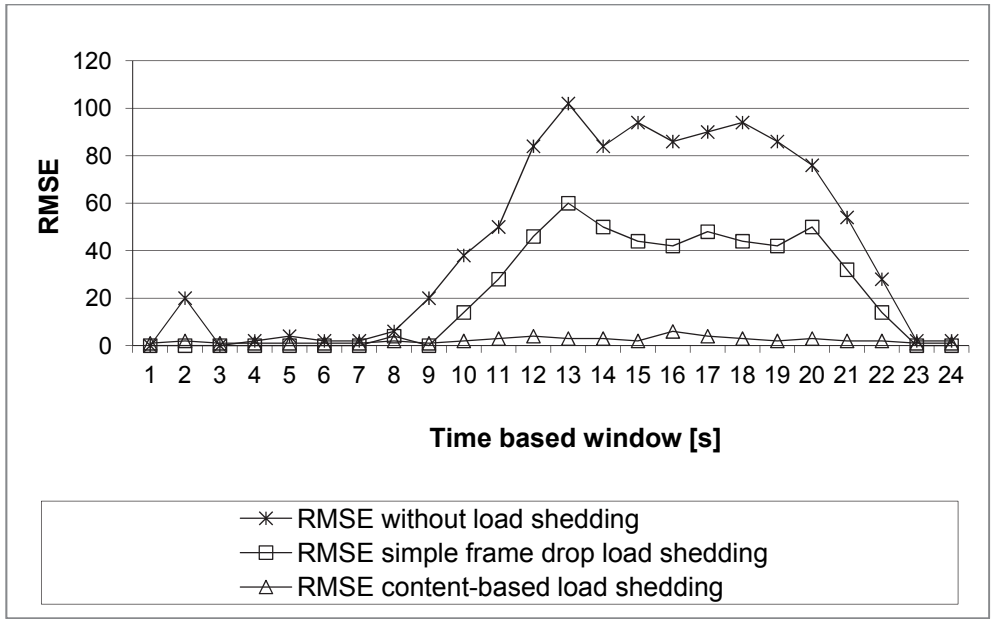

Figure 4. Query result inaccuracy evaluated for the public safety monitoring query

As shown in Figure 4, there is significant difference between the RMSE metrics for the techniques. In general, the higher the number of faces in the multimedia data stream, the greater the error introduced. In the simple frame drop load-shedding case, when overload starts after the 8th second, the input queue exceeds the remaining window capacity and tuples arriving after this limit has been reached are removed. This prevents resources from running out; however, important tuples may be (and in fact are) discarded. This approach reduces query accuracy.

On the other hand, in content-based load shedding, the Load Controller monitors the query processor and input queues. Based on feedback information, the Load Controller checks whether the number of enqueued tuples is rising and automatically adjusts the value of ρ accordingly. The higher the number of enqueued tuples, the lower the value of the

threshold, as shown in Table 1. Thus, the most valuable tuples are those which have been identified as key-frames. Based on our previous assumption, such tuples should not be discarded since they maximize monitoring query (4) result accuracy. Hence, our content-based load shedding algorithm performs considerably better when visual content analysis is employed, and the average number of recognized faces is higher.

5. Conclusion

In this paper, we have described our novel content-based load shedding framework, based on searching and matching algorithms, for analysing video tuples arriving within multimedia data streams. The framework tracks and registers all symptoms of overload, and either prevents overload before it occurs, or minimizes its effects. We have shown that at any point in the current sliding window, our content-based load shedding module minimizes the number of unprocessed tuples by appropriately adjusting the drop threshold. As a result, our technique minimizes query result inaccuracy. Our Load Controller can continuously monitor changes in the video stream and can accurately decide whether to summarize incoming tuples or forward them to the query processor. We have extended our Multimedia Continuous Query Language (MCQL) syntax to enable this load shedding technique. The effectiveness of the framework has been verified using both artificial and real data video streams collected from monitoring devices commonly used in the public safety sector. We are currently adapting the proposed technique to support correlated aggregates over multimedia data streams; however, more extensive study is required to provide a comprehensive analysis.

References

[1] D. J. Abadi, D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S. Zdonik, “Aurora: A New Model and Architecture for Data Stream Management,” Internat. J. Very Large Data Bases (VLDB J.), 12:2 (2003), 120–139.

[2] S. Acharya, P. B. Gibbons, and V. Poosala, “Congressional Samples for Approximate Answering of Group-By Queries,” Proc. ACM SIGMOD Internat. Conf. on Management of Data (SIGMOD ’00) (Dallas, TX, 2000), pp. 487–498.

[3] S. Acharya, P. B. Gibbons, V. Poosala, and S. Ramaswamy, “Join Synopses for Approximate Query Answering,” Proc. ACM SIGMOD Internat. Conf. on Management of Data (SIGMOD ’99) (Philadelphia, PA, 1999), pp. 275–286.

[4] Amiri, A., Fathy, M., Naseri, A., "Key-frame extraction and video summarization using QR-Decomposition", Digital Content, Multimedia Technology and its Applications (IDC), 2010 6th International Conference on, On page(s): 134 - 139, Volume: Issue: , 16-18 Aug. 2010

[5] N. Apostoloff and A. Zisserman, "Who Are You? – Real-Time Person Identification," Proc. British Machine Vision Conf. (BMVC '07) (Coventry, UK, 2007), pp. 509–518.

[6] A. Arasu, B. Babcock, S. Babu, J. Cieslewicz, M. Datar, K. Ito, R. Motwani, U. Srivastava, and J. Widom, "STREAM: The Stanford Data Stream Management System," Stanford Univ. InfoLab, 2004, <http://dbpubs.stanford.edu:8090/pub/2004-20>.

[7] B. Babcock, S. Chaudhuri, and G. Das, "Dynamic Sample Selection for Approximate Query Processing," Proc. ACM SIGMOD Internat. Conf. on Management of Data (SIGMOD '03) (San Diego, CA, 2003), pp. 539–550.

[8] B. Babcock, M. Datar, and R. Motwani, "Load Shedding for Aggregation Queries over Data Streams," Proc. 20th Internat. Conf. on Data Eng. (ICDE '04) (Boston, MA, 2004), pp. 350–361.

[9] Camara-Chavez, G., Precioso, F., Cord, M., Phillip-Foliguet, S., de A. Araujo, A., "An interactive video content-based retrieval system", Systems, Signals and Image Processing, 2008. IWSSIP 2008. 15th International Conference on, On page(s): 133 - 136, Volume: Issue: , 25-28 June 2008

[10] K. Chakrabarti, M. N. Garofalakis, R. Rastogi, and K. Shim, "Approximate Query Processing Using Wavelets," Proc. 26th Internat. Conf. on Very Large Data Bases (VLDB '00) (Cairo, Egy., 2000), pp. 111–122.

[11] S. Chandrasekaran, O. Cooper, A. Deshpande, M. J. Franklin, J. M. Hellerstein, W. Hong, S. Krishnamurthy, S. Madden, V. Raman, F. Reiss, and M. Shah, "TelegraphCQ: Continuous Dataflow Processing for an Uncertain World," Proc. 1st Biennial Conf. on Innovative Data Syst. Res. (CIDR '03) (Asilomar, CA, 2003).

[12] S. Chang, L. Zhao, S. Guirguis and R. Kulkarni, "A Computation-Oriented Multimedia Data Streams Model for Content-Based Information Retrieval." In Multimedia Tools and Applications, Volume 46, Nos. 2-3, pages 399-423, Jan 2010.

[13] Chasanis, V.T., Likas, A.C., Galatsanos, N.P., "Scene Detection in Videos Using Shot Clustering and Sequence Alignment", Multimedia, IEEE Transactions on, On page(s): 89 - 100, Volume: 11 Issue: 1, Jan. 2009

[14] S. Chaudhuri, R. Motwani, and V. Narasayya, "On Random Sampling over Joins," Proc. ACM SIGMOD Internat. Conf. on Management of Data (SIGMOD '99) (Philadelphia, PA, 1999), pp. 263–274.

[15] G. F. Franklin, J. D. Powell, and A. Emami-Naeini, Feedback Control of Dynamic Systems, Prentice Hall, Upper Saddle River, NJ, 2002.

[16] Y. E. Ioannidis and V. Poosala, "Histogram-Based Approximation of Set-Valued Query-Answers," Proc. 25th Internat. Conf. on Very Large Data Bases (VLDB '99) (Edinburgh, Scot., 1999), pp. 174–185.

[17] L. Lim, M. Wang, and J. S. Vitter, "SASH: A Self-Adaptive Histogram Set for Dynamically Changing Workloads," Proc. 29th Internat. Conf. on Very Large Data Bases (VLDB '03) (Berlin, Ger., 2003), pp. 369–380.

[18] S.-H. Lin, S.-Y. Kung, and L.-J. Lin, “Face Recognition/Detection by Probabilistic Decision-Based Neural Network,” IEEE Trans. Neural Networks, 8:1 (1997), 114–132.

[19] Jiebo Luo, Papin, C., Costello, K., "Towards Extracting Semantically Meaningful Key Frames From Personal Video Clips: From Humans to Computers", Circuits and Systems for Video Technology, IEEE Transactions on, On page(s): 289 - 301, Volume: 19 Issue: 2, Feb. 2009

[20] R. Maison, D.J. Steen, M. Zakrzewicz and Z. Biniek, “Monitoring High Performance Data Streams in Vertical Markets: Theory and Applications in Public Safety and Healthcare,” Bell Labs Technical Journal, 16(3) (2011), 163–180.

[21] R. Maison and M. Zakrzewicz, “Prediction-Based Load Shedding for Burst Data Streams,” Bell Labs Technical Journal, 16(1) (2011), 121–132.

[22] Y. Matias, J. S. Vitter, and M. Wang, “Dynamic Maintenance of Wavelet-Based Histograms,” Proc. 26th Internat. Conf. on Very Large Data Bases (VLDB ’00) (Cairo, Egy., 2000), pp. 101–110.

[23] R. Motwani, J. Widom, A. Arasu, B. Babcock, S. Babu, M. Datar, G. Manku, C. Olston, J. Rosenstein, and R. Varma, “Query Processing, Resource Management, and Approximation in a Data Stream Management System,” Proc. 1st Biennial Conf. on Innovative Data Syst. Res. (CIDR ’03) (Asilomar, CA, 2003).

[24] A. Pentland, B. Moghaddam, and T. Starner, “View-Based and Modular Eigenspaces for Face Recognition,” Proc. IEEE Conf. on Comput. Vision and Pattern Recognition (CVPR ’94) (Seattle, WA, 1994), pp. 84–91.

[25] F. Reiss and J. M. Hellerstein, “Data Triage: An Adaptive Architecture for Load Shedding in TelegraphCQ,” Proc. 21st Internat. Conf. on Data Eng. (ICDE ’05) (Tokyo, Jpn., 2005), pp. 155–156.

[26] F. Reiss and J. M. Hellerstein, “Declarative Network Monitoring with an Underprovisioned Query Processor,” Proc. 22nd Internat. Conf. on Data Eng. (ICDE ’06) (Atlanta, GA, 2006), p. 56.

[27] F. Sameria and S. Young, “HMM-Based Architecture for Face Identification,” Image and Vision Comput., 12:8 (1994), 537–543.

[28] N. Tatbul, U. Çetintemel, S. Zdonik, M. Cherniack, and M. Stonebraker, “Load Shedding in a Data Stream Manager,” Proc. 29th Internat. Conf. on Very Large Data Bases (VLDB ’03) (Berlin, Ger., 2003), pp. 309–320.

[29] N. Tatbul and S. Zdonik, “Window-Aware Load Shedding for Aggregation Queries over Data Streams,” Proc. 32nd Internat. Conf. on Very Large Data Bases (VLDB ’06) (Seoul, Kor., 2006), pp. 799–810.

[30] Y.-C. Tu, S. Liu, S. Prabhakar, and B. Yao, “Load Shedding in Stream Databases: A Control-Based Approach,” Proc. 32nd Internat. Conf. on Very Large Data Bases (VLDB ’06) (Seoul, Kor., 2006), pp. 787–798.

[31] M. A. Turk and A. P. Pentland, “Face Recognition Using Eigenfaces,” Proc. IEEE Conf. on Comput. Vision and Pattern Recognition (CVPR ’91) (Maui, HI, 1991), pp. 586–591.

[32] P. Viola and M. Jones, “Robust Real-Time Object Detection,” Proc. 2nd Internat. Workshop on Statistical and Computational Theories of Vision (SCTV ’01) (Vancouver, BC, Can., 2001).

[33] J. S. Vitter and M. Wang, “Approximate Computation of Multidimensional Aggregates of Sparse Data Using Wavelets,” Proc. ACM SIGMOD Internat. Conf. on Management of Data (SIGMOD ’99) (Philadelphia, PA, 1999), pp. 193–204.

[34] L. Wiskott, J.-M. Fellous, N. Krüger, and C. von der Malsburg, “Face Recognition by Elastic Bunch Graph Matching,” IEEE Trans. Pattern Analysis and Machine Intelligence, 19:7 (1997), 775–779.

[35] Qing Xu; Pengcheng Wang; Bin Long; Sbert, M.; Feixas, M.; Scopigno, R.; , "Selection and 3D visualization of video key frames," Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on , vol., no., pp.52-59, 10-13 Oct. 2010 doi: 10.1109/ICSMC.2010.5642204

[36] W. Zhao, A. Krishnaswamy, R. Chellappa, D. L. Swets, and J. Weng, “Discriminant Analysis of Principal Components for Face Recognition,” Face Recognition: From Theory to Applications (H. Wechsler, P. J. Phillips, V. Bruce, F. Fogelman Soulié, and T. S. Huang, eds.), Springer-Verlag, Berlin, Heidelberg, New York, 1998, pp. 73–85.

[37] Y. Zhuang, Y. Rui, T.S. Huang and S. Mehrotra, "Adaptive key frame extraction using unsupervised clustering," Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference on , vol.1, no., pp.866-870 vol.1, 4-7 Oct 1998.

Received September, 2011