Quality Management in Database Systems - A Thesis Proposal

Yi-Cheng Tu
Advisor: Prof. Sunil Prabhakar
Committee: Prof. Ahmed Elmagarmid, Prof. Dongyan Xu
Department of Computer Sciences, Purdue University

Abstract

In this proposal, we argue that *quality* parameters such as delay and uncertainty are more important performance metrics than processing time, which is the only concern in query optimization in traditional DBMS. We explore specific problems on how to maintain quality in multimedia and stream databases, with a focus on the latter. Data stream processing have attracted a great deal of attention from the database community. The continuous feature of both data and queries in data stream management systems (DSMSs) place great demand on system resources. However, queries can be processed with different levels of quality such as timeliness, reliability, and uncertainty. In this proposal, we discuss the problem of how to maintain quality in query processing in DSMSs. We focus on two types of quality metrics: general Quality-of-Service parameters that apply to all types of queries and *uncertainty* parameter for probabilistic queries. A widely-used approach to maintain QoS (especially tuple delays) in DSMS query processing is load shedding, i.e., dropping data. Current load shedding solutions utilize simple, intuitive ideas in determining the time and amount of load to be discarded and do not work well in the presence of system/environmental disturbances. We propose a solution based on feedback control theory with significantly improved long-term performance. For probabilistic queries, an important optimization is to have the least uncertainty in query results under resource contraints. We propose to approach this problem by exploiting the temporal dependence in stream data.

I. Introduction

A majority of DBMS research has focused on efficient processing of queries. In these systems, the only performance metric of interest is query procesing time. However, with the flourishment of new database applications such as multimedia and data streams, we realize that a number of new performance metrics are of more importance than processing time. In this proposal, we call these metrics the *quality* in query processing. Leaving quality as an application-level issue is not a choice because the optimizations needed to accomplish quality maintenance for multiple queries can only be performed in the database engine. On the other hand, query optimization functionalities in current DBMSs are ill-fitted for the maintenance of quality. Therefore, additional modules need to be developed in the DBMS core to handle quality-related concerns.

Our work on quality management in databases was first motivated by the QoS requirements on query result delivery in VDBMS - a multimedia DBMS developed in Purdue. In this system, users specify not only the query, but the quality (e.g., resolution, frame rate) of the media objects (as query results) to be delivered. While quality in this problem is actually the well-established quality-of-service (QoS) requirement in multimedia systems, replying exclusviely on a QoS-provisioning OS or middleware is not enough. We proposed a quality-aware multimedia DBMS that directly takes quality into account during query optimization. In this proposal, however, we focus more on quality issues in data stream management.

Applications related to processing of data streams have attracted a great deal of attention from the database community. With great social/economical interests, these applications flourish in a number of fields such as environment monitoring, system diagnosis, financial analysis, and mobile services. Unlike traditional data that are mostly static, stream data are produced continuously (e.g. from a sensor network) and are generally not kept in storage after being processed. Furthermore, most queries against stream data are persistent queries that output results whenever they are available. Thus, data stream processing brings great challenges to DBMS design: it imposes a *data-active, query-passive* DBMS model instead of the *data-passive, query-active* model for traditional DBMSs [1]. In recent years, a number of Data Stream Management Systems (DSMSs) have been developed [1]–[4]. In traditional DBMS where static data and snapshot queries are the mainstream, we focus

on efficient processing of queries and the query results are supposed to be the same no matter what query plan is used. In stream data management, however, (continuous) queries can be processed with different levels of timeliness, reliability, and uncertainty. We name these parameters the *quality* of query processing in DSMSs. In this proposal, we focus on two types of quality metrics.

Similar to those in other real-time applications [5], the first class of quality parameters in DSMSs can be applied to all types of queries. Important QoS parameters in DSMSs include: processing delay, data loss ratio, sampling rate, etc. A salient feature of data stream management is the real-time constraints associated with query processing. In many applications of DSMS, query results are required to be delivered before either a firm (e.g. tracking of stock prices) or soft (e.g. network monitoring for intrusion detection) deadline. Therefore, processing delay is the most critical quality in these applications.

The other class of quality parameters only applies to probabilistic queries. Most streaming data are inherently uncertain or incomplete due to errors in data collection (e.g., sensor networks) and resource limitations (e.g., battery power in sensors). Therefore, it makes more sense to reason and query stream data in a probabilistic manner. For example, a probabilistic range query returns a set of obejects as well as the probability that each object falls into the specified range. The uncertainty involved in the query results becomes a quality metric as users always prefer answers with high certainty. Previous study [6] has shown that the definition of uncertainty depends on the type of probabilistic queries.

Our interests are on the maintainence of the above two classes of quality in data stream environments. The difficulty of quality maintainence comes from two aspects: physical resource limitations and the time-variant behaviour of data sources. In practice, a DSMS could accommodate hundreds or even thousands of streams and quality requirements may easily be violated due to overloading. Even with careful query optimization and admission control, the runtime fluctuations of application resource usage (e.g. bursty arrivals) may still cause temporary congestion that interferes with real-time data processing. The target of our work is to answer the question of *"how to keep the best quality under limited resource"*. Tradeoffs have to be performed between critical and non-critical quality parameters and we are to find the best such tradeoffs. For example, to improve processing delays, we can increase data loss rate by *load shedding* [7] or reduce the window size for windowed operations [8]. For many cases, the problem can be formulated as an optimization whose solutions are not easy to find efficiently. What further complicates the issue is that the inputs to the optimization change over time.

II. Related Work

Research on QoS control was first motivated by the real-time requirements of multimedia applications. Most of these efforts emphasize system and network level resource management, which is provided as a service of the operating system [9] or a middleware [10]. The system maps QoS requirements of applications to resource use (system QoS) and QoS control is accomplished by regulating resource allocation to individual applications.

Current efforts on DSMSs have addressed system architecture [3], [11], query processing [12], [13], query optimization [14], and stream monitoring [15]. Relatively less attention has been paid to a unified framework to support quality maintenance. An important issue related to quality control in DSMSs is the development of scheduling policies of query operators. Two relevant efforts present scheduling algorithms that minimizes tuple delays [16] and runtime memory consumption [17].

Load shedding has been extensively utilized to deal with overloading in DSMSs [7], [18], [19]. [18] discusses load shedding strategies that minimize the loss of accuracy of aggregation queries. To increase accuracy of arbitrary queries, a *data triage* approach that exploits synopses of the discarded data is proposed in [19]. Earlier work on QoS-driven load shedding in the context of the Aurora [7] DSMS (now becoming a part of the Borealis project [20]) is closely related to our study. In [7], three critical questions about load shedding are raised: *when*, *where*, and *how much* to shed. With the goal of controlling system load on a desirable level, the Aurora load shedder works well in an environment with stable or slow-changing data arrival rates. To some extent, our work aims to provide better answers to the questions of *when* and *how much* to shed load under a highly dynamic environment.

The application of control theory, which will be used in our study, is inspired by [21] and [22] that deal with the problem of managing deadline misses in real-time systems. However, the system and metrics used in [21] and [22] are totally different from ours. While our goal is to work directly on a real DSMS, both [21]

and [22] evaluate their designs by simulations. Feedback control has also been used to assist QoS adaptation of multimedia applications [23].

Probabilistic query processing in DSMSs is first mentioned in [6]. In this paper, type definitions and evaluation algorithms are thoroughly discussed. Deshpande *et al.* [24], [25] proposes the usage of probabilistic models to generate data acquisition plans for sensor networks with the goal of saving sensor battery power. In their work, simple models (time-inviriant Guassian) are used and only snapshot queries are considered. In our study, we plan to extend both aspects within the context of our problem. The value of forecasting models in times series anlysis are being realized by the database community: in [26], predictions generated by Kalman filters are used to process deterministic queries; In another paper [27], Brownian models are utilized for similar purposes. Indexing techniques have also been benifited by forecasts obtained from the ARIMA model [28]. Works in close spirit to our quality control scheme include [29], [30] where the problem of approximate data replication was considered. In their solution, adaptive filters are set to data streams and a parametric algorithm to control the tradeoffs between data consistency and system performance is developed. This strategy, however, is shown [25] to be less efficient than simple methods that use statistical models.

III. Current Research

A. QuaSAQ: Enabling QoS in Multimedia Databases

Our work on quality issues in DBMS was originally motivated by the problem of QoS support in multimedia databases in the context of the VDBMS project.[1] In this project [31], we envision users accessing the multimedia databases via a simple user interface. In addition to specifying the multimedia items of interest, the user specifies a set of desired quality parameter bounds. To accomplish such quality-aware media retrieval, it is not enough to deploy a multimedia DBMS on top of a QoS-provisioning OS since the latter does not handle QoS specification/mapping. Thus, we argue that QoS support has to be integrated into the DBMS.

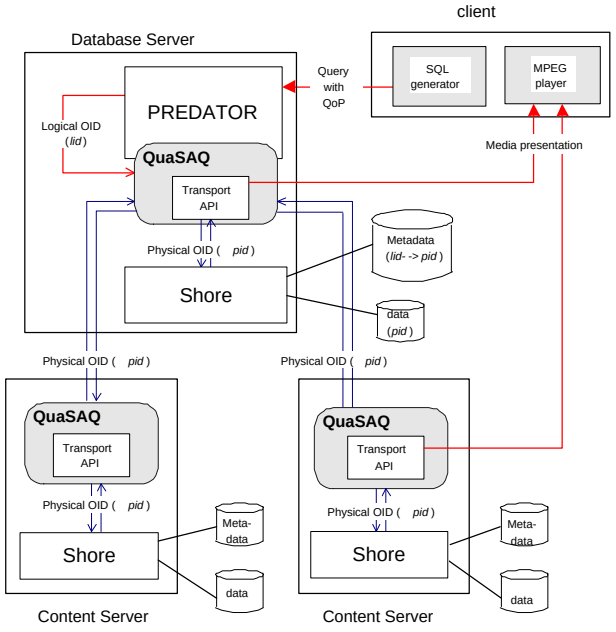

Fig. 1. Architecture of QuaSAQ prototype.

The key idea of our approach is to augment the query evaluation and optimization modules of a distributed database system (D-DBMS) to directly take QoS into account. To incorporate QoS control into the database, user-level QoS parameters are translated into application QoS and become an augmented component of the query. For each raw media object, a number of copies with different application QoS parameters are generated offline by transcoding and these copies are replicated on the distributed servers. Based on the information of data replication and possible QoS adaptation options (e.g. frame dropping during playback), the query processor generates various plans for each query and evaluates them according to a predefined *cost model*. The query evaluation/optimization module also takes care of resource reservation to satisfy low-level QoS. For this part, we propose the design of a unified QoS-provisioning API and implementation modules, that enable negotiation and control of the underlying system and network QoS APIs, thereby providing a single entry-point to a multitude of QoS layers (system and network).

QuaSAQ is prototyped and evaluated as a part of the VDBMS system (Fig. 1). Experiments run on the QuaSAQ prototype show significantly improved QoS and throughput in media query processing.

[1] http://www.cs.purdue.edu/vdbms

B. Quality-Aware Data Replication

This study was derived from the topic mentioned in Section III-A. In contrast to other database applications, multimedia data can have a wide range of quality parameters such as spatial and temporal resolution, and compression format. In a quality-aware multimedia database, users can request data with a specific quality requirement due to the needs of their application, or the limitations of their resources. The database can support multiple qualities by converting data from the original (high) quality to another (lower) quality to support a user's query, or pre-compute and store multiple quality replicas of data items. On-the-fly conversion of multimedia data (such as video transcoding) is CPU intensive and can limit the level of concurrent access supported by the database. Storing all possible replicas, on the other hand, requires unacceptable increases in storage requirements. We show that the relative storage usage required to store all qualities is on the order of the number of such qualities. With an overall storage constraints, the selection of multiple-quality replica becomes a problem. Solutions to this problem can also be found useful in a number of other (database) problems, namely, materialized view selection, cube computation, and selection of multi-quality chemical/biological data.

We study the problem under two different system models: Hard Quality, and Soft-Quality, and establish that the problem is NP-hard in both cases. Under the soft quality model, users are willing to negotiate their quality needs, as opposed to the hard quality system wherein users will only accept the exact quality requested. In the hard-quality system, we focus on minimizing request reject rate due to unavailability of the requested quality replica, or in other words, maximizing throughput. Via rigorous probabilistic analysis, we managed to reduce the problem to a 0-1 Knapsack problem and we propose an efficient solution. According to this solution, we only need to consider the access rate and size of each individual replica in selecting them while other features such as playback time and transcoding costs can be safely ignored. For the soft quality system [32], an important optimization goal is to minimize utility loss. The problem is found to be a variation of the famous k-median problem. We propose two powerful greedy algorithms to solve this problem. Extensive simulations show that our algorithm performs significantly better than other heuristics and finds near-optimal selection plans.

A unique feature of our study is that we also consider the dynamic replica selection problem where query patterns are assumed to be non-static. By exploiting one interesting feature of our original solution, we develop an elegant algorithm to handle changes of query patterns. While our static algorithm runs for cubic time, the dynamic version runs on logarithmic time and acheives the same level of optimality in terms of the quality of replicas selected.

C. Control-Based Load Shedding in DSMSs

1) Introduction: Load shedding is an important method to guarantee quality (especially tuple delays) in data stream processing under overloading situations. In performing load shedding, we always have to answer the following three questions: 1) *When* to shed load? 2) *How much* load to shed? and 3) *Where* (in the query network) to shed load? Current load shedding algorithms, which can be annotated in Fig 2, uses rule of thumb in making some of the critical decisions such as how much load to shed.

```
1  for every T time units
2      if measured load L is greater than
         system capacity L_0
3        do shedding load with amount L − L_0
4      else allow L_0 − L more load to come
```

Fig. 2. Generic load shedding algorithm

This simple algorithm works well when input load changes infrequently. In other words, system input stays in steady state for most of the time. However, this is not the case in practice. Streaming data are intrinsically dynamic with respect to the arrival patterns [33]. As an example, Fig. 3 shows the traces of real TCP traffic recorded from a cluster of web servers. Note the number of packet arrivals per unit time fluctuates within the range of $[120, 450]$ and no obvious period can be observed. In addition to bursty arrivals, some characteristics of DSMS such as the per-tuple processing (CPU) cost also changes over time. As a result, the algorithm shown in Fig. 2, which depends heavily on static estimates of system status, could easily fail. This opens great opportunities for optimization towards an auto-configuring DSMS that smartly adjusts quality levels of streams in response to fluctuations of system status and input rates.

Our solution to the problem adapts techniques derived from the field of control theory. Specifically, we transform the problem into a feedback control loop and design a feedback controller based on a dynamic model of the raw DSMS system derived from experiments. Feedback (closed-loop) control is known to be an excellent tool in maintaining runtime stability of systems under internal/environmental uncertainties such as modeling error, input, and output disburtances. On the contrary, the algorithm shown in Fig. 2 is an example of *open-loop* control and cannot handle the aforementioned uncertainties. The main difference between closed-loop and open-loop control methods is that the control decisions are made taking current system output (feedback) into consideration.

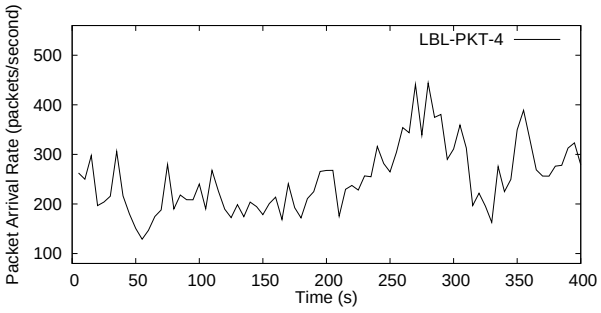

Fig. 3. Fluctuations in the arrival rate of real TCP traffic.

2) The feedback control loop: Figure 4 shows the architecture of the proposed control-based load shedding framework. The system to be controlled (i.e, *plant* in control terms) is the DSMS. Plant status is monitored periodically and output signals are sent to the controller. We use the Deadline Miss Ratio (M), which is the fraction of tuples that miss their processing deadlines, as the output signal. The fluctuations of data arrival rate and the inaccuracy in CPU cost estimation of operators are modeled as disturbances to the system. A load shedder (*actuator*) drops tuples from the streams to reduce the total load on the CPU. The percentage of tuples to drop is determined by the controller, while choosing the victim tuples to drop depends on the load shedding algorithm we choose. In this study, we use the simplest load shedding strategy: choosing victim tuples randomly.

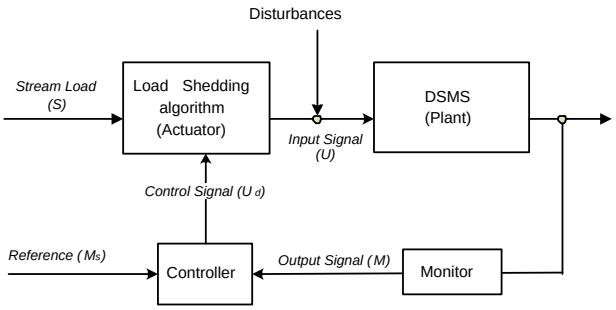

Fig. 4. The control-based load shedding framework.

The feedback control loop works as follows: a desired deadline miss ratio M_s is specified by the system administrator. The monitor measures the output signal M at the end of every sampling period. The controller compares M to M_s and generates the control signal: the amount of load that needs to be shed. For convenience, we define the control signal (U_d) to be the percentage of load that needs to be kept. In other words, $1 - U_d$ is the shedding factor. The controller then sends the load change U_d to the actuator. Clearly, the generation of input signal U_d is the key point in this loop. We use a classical Partial-Integral-Derivative (PID) controller in this work. Please refer to [34] for more details on system model and controller design.

3) Results: In this work [34], we evaluate the idea by using a DSMS simulator, the core of which is an operator scheduler adapting the Earliest Deadline First (EDF) policy. The EDF-based system has very simple input/output model that eases our controller design.

We test the simulator with synthetic and real stream data and compare the performance of our feedback-control-based method with that of an open-loop strategy ("static") similar to the approach in Fig. 2. According to Fig. 5, the control-based strategy performs better in reducing overshoots when compared with static shedding. This is further supported by the deadline miss ratio measured: deadline miss events are abundant in static shedding while very few are observed in control-based shedding. Control-based shedding shows less undershoot than static shedding in both Fig 5a and Fig 5b.

4) Summary: This study shows the validity of using feedback control to guide load shedding in DSMSs for the purpose of maintaining tuple delays. A common practice for DSMSs to overcome excessive incoming requests, load shedding is a difficult problem due to the bursty data input and time-dependent tuple processing cost. We use a feedback control loop to dynamically adjust load under such uncertainties. Compared to previous work, the control-based approach leads to significantly better quality with less waste of resources. We evaluate our idea by simulations. However, real challenges of sysem modeling and model-based controller design cannot

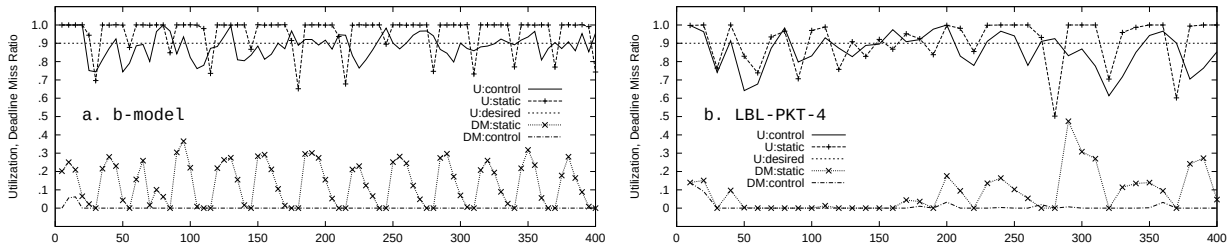

Fig. 5. Performance of different load shedding strategies under different bursty traffic patterns. a. Synthetic traffic; b. real TCP traffic.

be properly addressed unless we work on a real-world DSMS.

D. Control-Based Load Shedding in Borealis

In Section III-C we presented the idea of using feedback control in making critical load shedding decisions and evaluated the approach by simulations. However, such research will be convincing only when we work directly on a real DSMS. Controlling real systems brings extra challenges to the design of feedback controller. In this section, we enumerate these challenges and discuss our solutions. In this work, we define average tuple delay as the control signal and use the Borealis stream manager [20] as our experimental system.

1) Modeling Borealis: The effectiveness of feedback control depends heavily on the acuracy of the system model of the DSMS. Specifically, we need a dynamic model that describes the response (in the form of output signal) of the system to various input signals. While such models are given beforehand in simulations, modeling a real-world system is non-trivial. Fortunately, *system identification* techniques have been established by the control engineering community. The basic idea is to model the plant as a difference equation with unknown parameters. Then we can determine the order and parameters of the difference equation experimentally.

As to the Borealis system, which uses a *Round-robin* operator shceduler, we can start the modeling process by a simple analysis. Let $y(k)$ be the average delay of all tuples (i.e., output signal) that arrived during the k-th controlling period. The Borealis system model is the following discrete function:

$$y(k) = q(k-1) \cdot c(k) = c(k) \sum_{i<k} \left[f_{in}(i) - f_{out}(i) \right] \tag{1}$$

where $c(k)$ is the expectation of per-tuple CPU cost, $q(k-1)$ is the number of tuples in the query network, and $f_{in}(k)$ (or $f_{out}(k)$) is number of packets that entered (or left) the DSMS during the k-th period. For now, we set the cost factor $c(k)$ to a constant c. The intuition behind the above model is: as the round-robin policy assigns no priorities to jobs and the waiting queue of individual operators are FIFO, the tuples coming after period k will not be processed until all outstanding tuples are processed. We experimentally verified the above model by feeding the raw Borealis system with synthetic inputs and the results strongly supported the above model.

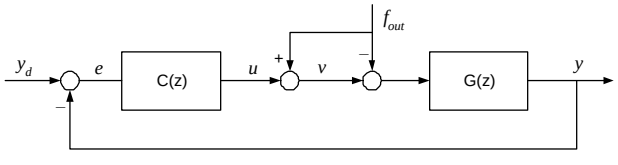

Fig. 6. The control-based load shedding framework.

2) Controller design: With the Borealis system model, we can use standard control theoretical techniques to design our feedback controller. With the first-order model shown in Eq. (1), we derive a basic control loop as illustrated in Fig. 6, where y_d is the preset reference value for delay time, $e = y_d - y$ is the error signal, and u represents the controller output (with the same unit as f_{in}). The meaning of u is: the *increase* of the number of outstanding tuples allowed in the next control period. Therefore, we denote $v = u + f_{out}$ as the desired data flow rate to the database as f_{out} tuples will leave the queue. $C(z)$ is the controller transfer function and $G(z)$ is the plant transfer function. According to control theory (details skipped here), we have $G(z) = \frac{cT}{z-1}$ and $C(z) = \frac{b_0 z + b_1}{c(z+a)}$ where T is the control period, b_0, b_1, and a are controller parameters.

From the DSMS point of view, the control goal is to let y (closely) track the target value y_r. Although user satisfaction is not affected when we have $y < y_r$, it implies that more data is lost than what is necessary (under

a overloading situation). Thus, we should set our design goal to fast convergence, meaning that the controlled system is capable of bringing y back to the desired value y_r very quickly when y deviates from y_d in either direction. Another parameter we consider is *system damping*, which measures how smooth system responses are to disturbances. For this purpose, a controller design based on pole placement is desirable to achieve guaranteed performance.[2] We develop the following controller by setting the convergence rate to three control periods and system damping to 1.0:

$$u(k) = \frac{b_0 e(k) + b_1 e(k-1)}{cT} - au(k-1) \tag{2}$$

There are additional challenges in the control of DSMSs that are not addressed by standard control techniques described above. One salient problem is the unavailability of real-time output measurement. Accurate measurements of system output in real-time is essential in control system design. Unfortunately, in our system, the output signal is the delay time, which means the output measurement is not only delayed, but it is delayed by an unknown amount; We propose a solution to the problem based on the system model (Eq. 1): instead of using a measurement of delay y as feedback signal, we use an estimation of y that is derived from $q(k-1)$ as follows:

$$\hat{y}(k) = q(k-1)c(k) \approx q(k-1)c(k-1) \tag{3}$$

It is natural that Eq. (3) adds estimation errors to the closed-loop. We denote the estimation error as $\tilde{y} = y - \hat{y}$. Fortunately, our controller is still found to be robust by the following argument. When estimated output $\hat{y}$ is used as feedback signal, the output of the closed loop system is hence described by:

$$Y(z) = \frac{C(z)G(z)}{1 + C(z)G(z)} Y_d(z) - \frac{C(z)G(z)}{1 + C(z)G(z)} \tilde{Y}(z) \tag{4}$$

The closed-loop system is still stable as long as $\tilde{y}$ is bounded, which is always true. The Y_d term in Eq. (4) shows that the output of the closed-loop system still tracks the target reference signal with designed damping and convergence rate. However, the accuracy is compromised due to the introduction of estimation errors, as represented by the $\tilde{Y}$ term in Eq. (4).

3) Determining control period: Our work described above answers the question of "how much load to shed". Obviously, we need to consider the questions of "when" and "where" within our framework of feedback-control-based load shedding. Within the feedback control loop, the question of "when to shed load" can be answered be determining the control period T. We follow two rules in selecting T for our control system: 1). *Nyquist-Shannon sampling theorem.* As a fundamental rule in information theory, the theorem states that: when sampling a signal, the sampling frequency must be greater than twice the bandwidth of the input signal in order to be able to reconstruct the original signal from the sampled version. In our system, the sampling frequency should thus be at least twice of the changing frequency of the disturbances c and f_{in}. We expect the data input rate to be very bursty therefore high sampling frequency is desirable; 2). *Stochastic feature of output signal.* Although the CPU processing time for individual data tuple may be different, it is the average CPU processing time/delay time within a sampling period that is of interest. Taking the average of the CPU processing time/delay time would statistically eliminate some uncertainties. Lower sampling frequency has more smoothing effect to eliminate the uncertainties. As factors 1) and 2) have opposite preferences on sampling frequency, our choice of T should be a compromise between them.

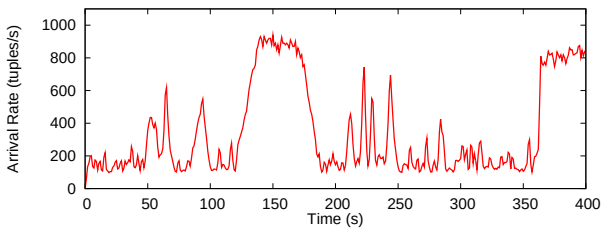

Fig. 7. Data arrival rates in the experimental streams.

4) Experimental Results: We implemented our control-based load shedding framework inside the Borealis system. We feed the system with synthetic bursty data streams as shown in Fig. 7. As compared to the simple algorithm (Fig. 8A), our approach (FB-CTRL) achieves much better control on the processing delay:most of the recorded y values are under the target value. For those rare occasions when $y > y_d$, the errors

[2]System poles are the roots of the denominator polynomial of the closed-loop transfer function. The location of system poles can tell how fast and smoothly the system responds to inputs therefore it is directly related to our design goal of fast convergence.

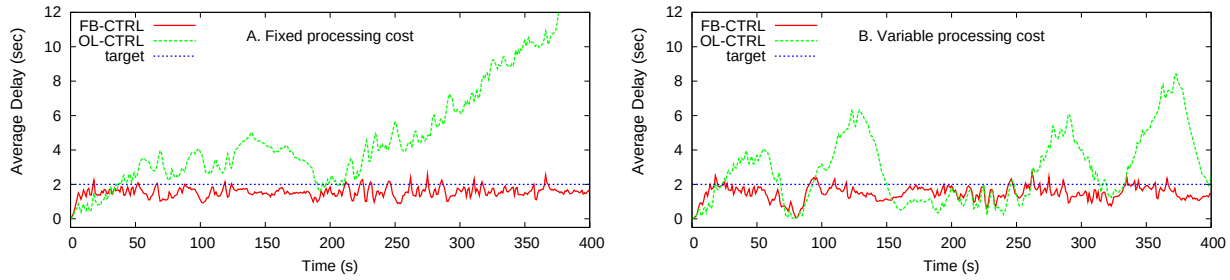

Fig. 8. Performance of different load shedding algorithms.

are small. The simple method (OL-CTRL), on the other hand, generates much more delay violations and renders the system unstable after the 200th second (i.e., y increases unboundedly). One thing to point out is: data loss for both methods are approximately the same (1 to 0.985). Other situations unchanged, we make the average processing cost c to change following a sinusoidal pattern. The results of this experiment are plotted in Fig. 8B. Again, the delays recorded in the FB-CTRL method are under the target value with very few exceptions. For the OL-CTRL method, the average delays seem to fluctuate sinusoidally, following the pattern of the changes of c. Similar to Fig. 8A, data loss are almost the same in both methods.

Furthermore, our approach are found to be robust in that its performance is not affected by the pattern of arrival rate variations, or the change of target value y_d. Details can be found in [35].

IV. Planned Research

A. More control-based research

The studies presented in Sections III-C and III-D can be extended in the following directions.

1) Adaptive control: So far we have worked on a system whose internal mechanisms are fixed. However, system model could evolve over time. For example, we ended up using the average processing cost c as a parameter in our system model. Although experimental results show that the system works well even under variable c (Fig.8B), we cannot prove the stability of the controller. A more convincing way to handle this would be to use a second (outer) loop to capture the dynamics of the system model itself and send it as a feedback to the current (inner) loop. By doing this, we can i) handle more dramatic variations of c; and ii) get better control results in terms of fewer undershoots (i.e., the case of $y < y_d$) which could translate into less data loss.

In addition to the parameters, the structure of system model could also change (e.g., a different operator scheduler is deployed). Adaptive control will generally not work under this situation. The good news is that this will not happen frequently. The solution would be to perform system identification and controller design beforehand for all possible situations. Therefore, we could switch to a new controller at runtime knowing a new scheduler is now in use.

2) Adaptation other than load shedding: We focused on load shedding throughout this paper. Other adaptation strategies, however, are also very popular in dealing with overloading. Generally speaking, the goal of all adaptation methods is to adjust the load that the system needs to process. However, this does not necessarily mean our controller design is not affected if a different adaptation method is utilized. For example, in the case of sampling rate reduction, we generally accomplish this by resetting the width of some adaptive filters on the stream sources [36]. The relationship between the width of the adaptive filters and the resulting data sending rate is generally not linear and hard to quantify. Therefore, this is the same as to add another source of input disturbances. Things could get worse if we adjust load by changing the size of windowed joins. As it happens inside the query network, our system model could be comprised. We need to investigate the behavior of our system and decide if re-modeling is needed.

3) More Sophisticated Quality Model: In this paper, we adapted a very simple model for QoS: we set a target only for the processing delay and data loss is basically regarded as the cost of achieving the main control goal of maintaining delays. The control on data loss is done in a best-effort manner. A more palpable model would involve setting targets for multiple QoS dimensions thus a single-in-multi-out system model has to be utilized. Relating this to discussions in Section IV-A.1, we can also combine different adaptation strategies thus a even

more complicated multi-in-multi-out model is needed. The complexity of such models could make a solution following this path very difficult but it is worth further investigations.

B. Improving quality of probabilistic queries by forecasting models

1) Introduction: In a (naive) push-based DSMS, queries are answered deterministically: we always let all the stream sources send their current values to the database and process the queries based on these updates. This solution has two problems: 1) streams values are just discrete samples of the underlying environment. Inherently, there are random errors involved in the sampling(measuring) of a physical world. Furthermore, as there are inevitably some delays between the time when the value is collected and the time when it is processed by the database system, there is no guarantee that we are dealing with fresh data. However, from the user's point of view, the only interesting thing is to get an idea of what's happening at the time when the query is processed. Therefore, it makes more sense to reason stream data probabilistically. 2) This solution requires updates from all streams at all times. As a result, the system may not have enough resources for doing that. One salient example is for sensor-based data sources: sensors have limited battery power, we cannot afford to let the sensors send updates frequently. Same problem for the bandwidth of the network between the data stream source and the data base. To remedy the above problems, probabilistic versions of the above queries are proposed [6]. For entity-based queries, we now return a set of stream IDs as well as the probability that each ID satisfy the specified condition. For value-based queries, we need to return a probability distribution function (PDF) instead of a single value. As the answers to queries become fuzzy, we may not need updates from all streams to be sent to the DSMS server. Instead, we can selectively acquire updates from individual stream sources. This is called *pull-based* or *data acquisition* method.

Quality of Probabilistic Queries. Nobody likes uncertainties in the query results. They are used simply because of the fuzzy data we got and the limited resources of our computing infrastructure. The quality of query results is thus measured by how far the results are from their deterministic version. For example, if we return a normal distribution as the result of a valued-based query, the smaller the variance of the distribution, the better quality we achieved. Quality of entity-based queries is a little trickier. Let's take the range query as example: if we have a query result that says: stream 5 has probability 0.9 of being in the range of $[a, b]$, we know stream 5 is most likely within the range thus the quality of this result is good. On the other hand, if we have probability 0.1, we know it is most likely out of the range and it is also of high quality. The worst thing is we have probability 0.5, which basically does not tell us much. It seems $|p - 0.5|$ is a good measure of quality here. People have also used *differential entropy* to quantify the uncertainty of a continuous random variable given its pdf.

2) Problem statement: We have n data streams, each can be viewed as a time series that generates a data item (whose value is a real number) every δt time units (without loss of generality, assume δt=1). Let y_i^j be the value of stream j $(j = 1, 2, \ldots, n)$ at time i $(i = 1, 2, \ldots, m)$ with m the most current time. So y_m^j is the current value of stream j. Due to the constraints of the memory, the whole history of the data stream y_i^j may not be stored. Hence, it is assumed that at current time m, only y_i^j for $i \geq i_0$ is retrievable from the memory. The query result q_i is a function of $y_i = [y_i^1, \ldots, y_i^n]$, denoting this by

$$q_i = f(y_i).$$

The problem is an optimization, which can be expressed in the following two flavors:

> PROBLEM 1. Due to system constraints, suppose the system can only update at most c data streams at the time one step forward (i.e., future time $m + 1$). But the system is allowed to selectively choose streams to report their updates. What is the best way to select the c streams? The value c can be either a fixed constant or a variable;
>
> PROBLEM 2. Let R denote the risk (or the discrepancy of the reported query and the actual solution) of a query. We are allowed to update as few data streams as possible for the immediate update of data values as long as R is smaller than a user-specified threshold, say Q. How many data streams to update? Which ones should we update?

3) A sketch of possible solution: The future value of a stream, y_i^j for $i > m$, is determined by the underlying physical system and since it is by definition unknown until it arrives, we can view it as being random. The statistics community has extensively studied time series data and proposed various models to predict future

values of such series. We propose to use prediction generated from these models to guide data acquisition in DSMSs.

If $\hat{y}_i^j$ denote a predictor of y_i^j, let $\hat{y}_i = [\hat{y}_i^1, \ldots, \hat{y}_i^n]$ be the predictor vector of y_i, then one would naturally use $\hat{q}_i = f(\hat{y}_i)$ as the predictor or estimator of q_i, a candidate of R is the mean square error of $\hat{q}_i$:

$$R_i(\hat{q}_i, q_i) = E(\hat{q}_i - q_i)^2 \tag{5}$$

where E is the expected value with respect to the probability distribution of the random variable q_i. Then the above two problems can be formally stated as

PROBLEM 1. $b = \arg\min_{a \in A} R_{m+1}(\hat{q}_{m+1}, q_{m+1})$ where A is the set of all possible selection of c streams among n streams, which has obviously $\begin{pmatrix} n \\ c \end{pmatrix}$ elements.

PROBLEM 2. $b = \arg\min_{a \in A} R_{m+1}(\hat{q}_{m+1}, q_{m+1}),$ such that $R_{m+1} \leq T$ where A is the set containing all possible subset of n streams.

Theoretically, the goal is to solve the above two minimization problems. But $R_{m+1}(\hat{q}_{m+1}, q_{m+1})$ involves unknown quantity q_{m+1}, it is necessary to use a operational surrogate target function which mimics $R_{m+1}(\hat{q}_{m+1}, q_{m+1})$. This necessitates a practical tool which provides a good predictor $\hat{y}_{m+1}^j$, i.e. one-step ahead predictor, based on statistical time series model. The solution to the minimization problem must be obtained within the sampling time interval, so that a decision on how to allocate the resource be made on time. A computationally efficient minimization algorithm is thus necessary. The algorithm must be able to update the solution in a sequential or recursive fashion once the new data enter the system.

There are a number of technical problems we need to address to make the idea work. (i) given the forecasting distributions of all n streams, how to (efficiently) process various types of queries; (ii) how do we judge the relative importance of each forecasting result and choose the ones that will give the most benefit to the quality of query processing; (iii) the computation of probabilistic queries may be CPU intensive. For continuous queries, is it possible to evaluate the queries incrementally? We are developing solutions to all these challenges.

4) Optimization in multi-query DSMS: Many studies in resource management in DSMSs focus on single-query optimization. That is, we assume that there is only one query running in the system and find solutions for different query types one by one. Although it is reasonable to approach the problem by making such simplified assumption, we may find that very little can be learned by studying single-query systems. The reason for this is: as different queries (even queries with the same type) have different data acquisition plan (e.g., solutions to PROBLEM 1 and PROBLEM 2 in Section IV-B.2), we may end up acquiring updates from most, if not all, streams. Under this situation, the pull-based method shows no advantage over the push-based solution [3]. It is important to find solutions for PROBLEM 1 mentioned in Section IV-B.2 under multiple queries. One possible solution is as follows: we change the definition of R given by Eq. (5), which is query-dependent, to something that is query-independent. For example, we can have

$$R_i(\hat{y}, y) = \sum_j E(\hat{y}_i^j - y_i^j).$$

By minimizing R_i with this definition, we are basically minimizing the errors in our estimation of the stream values. The intuition behind this solution is: quality of queries is positively related to the accuracy of data.

V. OTHER WORKS

A. Performance analysis of peer-to-peer media streaming systems

Recent research efforts have demonstrated the great potential of building cost-effective media streaming systems on top of peer-to-peer (P2P) networks. A P2P media streaming architecture can reach a large streaming capacity that is difficult to achieve in conventional server-based streaming services. Hybrid streaming systems that combine the use of dedicated streaming servers and P2P networks were proposed to build on the advantages of both paradigms. However, the dynamics of such systems and the impact of various factors on system behavior are not totally clear. In this study [37], [38], we present an analytical framework to quantitatively study the

[3] Actually, it is even worse due to the overheads for keeping stream states and computing predictions.

features of a hybrid media streaming model. Based on this framework, we derive an equation to describe the capacity growth of a single-file streaming system. We then extend the analysis to multi-file scenarios by solving an optimization problem. We also show how the system achieves optimal allocation of server bandwidth among different media objects. The unpredictable departure/failure of peers is a critical factor that affects the performance of P2P systems. We utilize the concept of *peer lifespan* to model peer failures. The original capacity growth equation is enhanced with coefficients generated from peer lifespans that follow an exponential distribution. We also propose a failure model under arbitrarily distributed peer lifespan. Results from large-scale simulations support our analysis.

B. Change Point Estimation of Bar Code Signals

Bar code is widely used in various businesses as a primary technique for production identification. In practice, bar code detection and reconstruction are prone to errors. Existing methods for bar code signal reconstruction is based on either the local approach or the regularization approach with total variation penalty. We formulate the problem explicitly in terms of change points of the 0-1 step function [39], [40]. The bar code is then reconstructed by solving the nonlinear least squares problem subject to linear inequality constraints, with starting values provided by the local extremes of the derivative of the convolved signal from discrete noisy data. Simulation results show a considerable improvement of the quality of the bar code signal using the proposed hybrid approach over the local approach.

C. Processing entity-based queries with non-value error tolerance in DSMSs

In this work [41], we study the problem of applying adaptive filters for approximate query processing in a distributed stream environment. We propose filter bound assignment protocols with the objective of reducing communication cost. Most previous works focus on value-based queries (e.g., average) with numerical error tolerance. In this study, we cover entity-based queries (e.g., nearest neighbor) with non-value-based error tolerance. We investigate different non-value- based error tolerance definitions and discuss how they are applied to two classes of entity-based queries: non-rank-based and rank-based queries. Extensive experiments show that our protocols achieve significant savings in both communication overhead and server computation.

D. VDBMS - a multimedia DBMS

Real-world video-based applications require database technology that is capable of storing digital video in the form of video databases and providing content-based video search and retrieval. Methods for handling traditional data storage, query, search, retrieval, and presentation cannot be extended to provide this functionality. The VDBMS research initiative [42]–[44] is motivated by the requirements of video-based applications to search and retrieve portions of video data based on content and by the need for testbed facilities to facilitate research in the area of video database management. Our fundamental concept is to provide a full range of functionality for video as a well-defined abstract database data type, with its own description, parameters, and applicable methods. Research problems that are addressed by VDBMS to support the handling of video data include MPEG7 standard multimedia content representation, algorithms for image-based shot detection, image processing techniques for extracting low-level visual features, a high-dimensional indexing technique to access the high-dimensional feature vectors extracted by image preprocessing, multimedia query processing and optimization, new query operators, real-time stream management, a search-based buffer management policy, and an access control model for selective, content-based access to streaming video. VDBMS also provides an environment for testing the correctness and scope of new video processing techniques, measuring the performance of algorithms in a standardized way, and comparing the performance of different implementations of an algorithm or component. The ultimate goal of the VDBMS project is a flexible, extensible framework that can be used by the research community for developing, testing, and benchmarking video database technologies.

VI. PUBLICATIONS

1. **Yi-Cheng Tu**, Jianzhong Sun, Mohamed Hefeeda, and Sunil Prabhakar. An Analytical Study of Peer-to-Peer Media Streaming Systems. Accepted to *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*.

2. Reynold Cheng, Ben Kao, Sunil Prabhakar, Alan Kwan, and **Yi-Cheng Tu**. Adaptive Stream Filters for Entity-Based Queries with Non-Value Tolerance. In *Proceedings of Intl. Conf. on Very Large Databases (VLDB)*, pp.37-48, Trondheim, Norway, August 2005.
3. **Yi-Cheng Tu**, Jingfeng Yan, and Sunil Prabhakar. Quality-Aware Replication of Multimedia Data. In *Proceedings of International Conference of Database and Expert Systems Applications (DEXA)*, pp. 240-249, Copenhagen, Denmark, August 2005.
4. **Yi-Cheng Tu**, Mohamed Hefeeda, Yuni Xia, Sunil Prabhakar, and Song Liu. Control-based Quality Adaptation in Data Stream Management Systems. In *Proceedings of International Conference of Database and Expert Systems Applications (DEXA)*, pp.746-755, Copenhagen, Denmark, August 2005.
5. Leming Qu and **Yi-Cheng Tu**. Change Point Estimation of Bar Code Signals. In *Proceedings of International Conference on Scientific Computing (CSC)*. pp.109-114, Las Vegas, USA, June 2005.
6. **Yi-Cheng Tu**, Sunil Prabhakar, Ahmed Elmagarmid and Radu Sion. QuaSAQ: An Approach to Enabling End-to-End QoS for Multimedia Databases. In *Proceedings of International Conference on Extending Database Technology (EDBT)*, pp.694-711, Herakolin, Greece., March 2004.
7. **Yi-Cheng Tu**, Jianzhong Sun and Sunil Prabhakar. Performance Analysis of A Hybrid Media Streaming System. In *Proceedings of SPIE/ACM Conference on Multimedia Computing and Networking (MMCN)*, pp.69-82, San Jose, CA., January 2004.
8. W. Aref, A. Catlin, A. Elmagarmid, J. Fan, M. Hammad, I. Ilyas, M. Marzouk, S. Prabhakar, **Y. Tu** and X. Zhu. VDBMS: A Testbed Facility for Research in Video Database Benchmarking. *Springer/ACM Multimedia Systems*. 9(6):575-585., June 2004.
9. W. Aref, A. Catlin, A. Elmagarmid, J. Fan, M. Hammad, I. Ilyas, M. Marzouk, S. Prabhakar, **Y. Tu** and X. Zhu. VDBMS: A Testbed Facility for Research in Video Database Benchmarking. In *Proceedings of Intl. Conf. on Distributed Multimedia Systems (DMS)*, pp.160-166, 2003.
10. W. Aref, A. Elmagarmid, J. Fan, J. Guo, M. Hammad, I. Ilyas, M. Marzouk, S. Prabhakar, A. Rezgui, A. Teoh, E. Terzi, **Y. Tu**, A. Vakali, X. Zhu. A Distributed Database Server for Continuous Media (Demo). *In Proceedings of International Conference on Data Engineering (ICDE)*, pp.490-491. San Jose, CA., March 2002.
11. Leming Qu and **Yi-Cheng Tu**. Change Point Estimation of Bi-level Functions. Accepted to Journal of Modern Applied Statistical Methods.

In Submission

12. **Yi-Cheng Tu**, Sunil Pabhakar, Jingfeng Yan, and Gang Shen. Selection of Quality-Specifi c Caches of Multimedia Data. Submitted to journal.
13. **Yi-Cheng Tu**, Song Liu, Sunil Pabhakar, and Bin Yao. Load Shedding in Stream Databases: A Control-Based Approach. Submitted to conference.
14. **Yi-Cheng Tu**. Qaulity-Driven Adaptation in Data Stream Management Systems. Submitted to workshop.

VII. Submission Plan

1. Improving quality of (single) probabilistic queries by forecasting models. VLDB (March 2006);
2. Optimizing quality of probabilistic queries in a multi-query environment. ICDE (July 2006);
3. Control techniques in self-adaptive DBMS. CIDR (August 2006).
4. Journal submission: control-based load shedding in DSMSs. VLDB Journal (early 2006).

References

[1] D. J. Abadi, D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S. B. Zdonik, "Aurora: a new model and architecture for data stream management." *VLDB Journal.*, vol. 12, no. 2, pp. 120–139, 2003.

[2] "Niagara Project, http://www.cs.wisc.edu/niagara/."

[3] T. S. Group, "STREAM: The Stanford Stream Data Manager," *IEEE Data Engineering Bulletin*, vol. 26, no. 1, pp. 19–26, March 2003.

[4] S. Chandrasekaran, A. Deshpande, M. Franklin, J. Hellerstein, W. Hong, S. Krishnamurthy, S. Madden, V. Raman, F. Reiss, and M. Shah, "TelegraphCQ: Continuous Datafbw Processing for an Uncertain World," in *Proceedings of 1st CIDR Conference*, January 2003.

[5] K. Nahrstedt and R. Steinmetz, "Resource Management in Networked Multimedia Systems," *IEEE Computer*, vol. 28, no. 5, pp. 52–63, 1995. [Online]. Available: citeseer.nj.nec.com/nahrstedt95resource.html

[6] R. Cheng, D. Kalashnikov, and S. Prabhakar, "Evaluating Probabilistic Queries over Imprecise Data," in *Proceedings of ACM SIGMOD '03*, June 2003, pp. 551–562.

[7] N. Tatbul, U. Çetintemel, S. Zdonik, M. Cherniack, and M. Stonebraker, "Load Shedding in a Data Stream Manager," in *Proceedings of the 29th VLDB Conference*, August 2003, pp. 309–320.

[8] A. Arasu, B. Babcock, S. Babu, M. Datar, J. Rosenstein, K. Ito, I. Nishizawa, and J. Widom, "Query Processing, Resource Management, and Approximation in a Data Stream Management System," in *Procs. of 1st CIDR Conf.*, January 2003.

[9] D. Yau and S. Lam, "Operating System Techniques for Distributed Multimedia," *International Journal of Intelligent Systems*, vol. 13, no. 12, pp. 1175–1200, December 1998.

[10] K. Nahrstedt, D. Xu, D. Wichadakul, and B. Li, "QoS-Aware Middleware for Ubiquitous and Heterogeneous Environments," *IEEE Communications Magazine*, 2001.

[11] D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, G. Seidman, M. Stonebraker, N. Tatbul, and S. Zdonik, "Monitoring Streams - A New Class of Data Management Applications," in *Procs. of the 28th VLDB Conf.*, August 2002, pp. 84–89.

[12] A. Dobra, M. Garofalakis, J. Gehrke, and R. Rastogi, "Processing complex aggregate queries over data streams," in *Procs. of SIGMOD Conf.*, June 2002, pp. 61–72.

[13] M. Hammad, M. Franklin, W. Aref, and A. Elmagarmid, "Scheduling for Shared Window Joins Over Data Streams," in *Proceedings of 29th VLDB Conf.*, August 2003, pp. 297–308.

[14] S. Viglas and J. Naughton, "Rate-Based Query Optimization for Streaming Information Sources," in *Procs. of SIGMOD Conf.*, June 2002, pp. 37–48.

[15] Y. Zhu and D. Shasha, "StatStream: Statistical Monitoring of Thousands of Data Streams in Real Time," in *Procs. of the 28th VLDB Conf.*, August 2002, pp. 358–369.

[16] D. Carney, U. Çetintemel, A. Rasin, S. Zdonik, M. Cherniack, and M. Stonebraker, "Operator Scheduling in a Data Stream Manager," in *Procs. of the 29th VLDB Conf.*, August 2003, pp. 838–849.

[17] B. Babcock, S. Babu, M. Datar, and R. Motwani, "Chain: Operator Scheduling for Memory Minimization in Data Stream Systems ," in *Proceedings of ACM SIGMOD '03*, June 2003, pp. 253–264.

[18] B. Babcock, M. Datar, and R. Motwani, "Load Shedding for Aggregation Queries over Data Streams," in *Procs. of ICDE Conf.*, 2004.

[19] F. Reiss and J. M. Hellerstein, "Data Triage: An Adaptive Architecture for Load Shedding in TelegraphCQ," in *Proceedings of ICDE*, April 2005, pp. 155–156.

[20] D. J. Abadi, Y. Ahmad, M. Balazinska, U. Çetintemel, M. Cherniack, J.-H. Hwang, W. Lindner, A. Maskey, A. Rasin, E. Ryvkina, N. Tatbul, and S. Zdonik, "The Design of the Borealis Stream Processing Engine," in *Procs. of 2nd CIDR Conference*, January 2005.

[21] C. Lu, J. Stankovic, G. Tao, and S. Han, "Feedback Control Real-Time Scheduling: Framework, Modeling, and Algorithms," *Journal of Real-Time Systems*, vol. 23, no. 1/2, pp. 85–126, September 2002.

[22] K.-D. Kang, S. H. Son, and J. Stankovic, "Managing Deadline Miss Ratio and Sensor Data Freshness in Real-Time Databases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 10, pp. 1200–1216, October 2004.

[23] B. Li and K. Nahrstedt, "A Control-Based Middleware Framework for Quality of Service Adaptations," *IEEE Journal of Selected Areas in Communications, Special Issue on Service Enabling Platforms*, vol. 17, no. 9, pp. 1632–1650, September 1999.

[24] A. Deshpande, C. Guestrin, and S. R. Madden, "Using Probabilistic Models for Data Management in Acquisitional Environments," in *Procs. of CIDR*, January 2005, pp. 317–328.

[25] A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein, and W. Hong, "Model-Drive Data Acquisition in Sensor Networks," in *Procs. of VLDB*, August 2004, pp. 588–599.

[26] A. Jain, E. Y. Chang, and Y.-F. Wang, "Adaptive Stream Resource Management Using Kalman Filters," in *Proceedings of SIGMOD*, June 2004, pp. 11–22.

[27] S. Zhu and C. V. Ravishankar, "Stochastic Consistency, and Scalable Pull-Based Caching for Erratic Data Sources." in *VLDB*, 2004, pp. 192–203.

[28] Y. Xia, S. Prabhakar, J. Sun, and S. Lei, "Indexing and Querying Constantly Evolving Data Using Time Series Analysis." in *DASFAA*, 2005, pp. 637–648.

[29] C. Olston, B. T. Loo, and J. Widom, "Adaptive Precision Setting for Cached Approximate Values," in *Proceedings of SIGMOD*, May 2001, pp. 355–366.

[30] C. Olston, J. Jiang, and J. Widom, "Adaptive filters for Continuous Queries Over Distributed Data Streams," in *Proceedings of SIGMOD*, June 2003, pp. 563–574.

[31] Y. Tu, S. Prabhakar, A. Elmagarmid, and R. Sion, "QuaSAQ: An Approach to Enabling End-to-end QoS for Multimedia Databases," in *Proceedings of 9th Intl. Conf. on Extending Database Technology (EDBT)*, March 2004, pp. 694–711.

[32] Y.-C. Tu, J. Yan, and S. Prabhakar, "Quality-Aware Replication of Multimedia Data," in *Proceedings of Database and Expert Systems Applications (DEXA)*, August 2005, pp. 240–249.

[33] M. Zhang, T. Madhyastha, N. Chan, S. papadimitriou, and C. Faloutsos, "Data Mining Meets Performance Evaluation: Fast Algorithms for Modeling Bursty Traffic," in *Proceedings of the 18th ICDE Conference*, Feburary 2002, pp. 507–516.

[34] Y.-C. Tu, M. Hefeeda, Y. Xia, and S. Prabhakar, "Control-based Quality Adaptation in Data Stream Management Systems. To Appear ," in *Proceedings of DEXA*, August 2005, pp. 746–755.

[35] Y.-C. Tu, S. Liu, S. Prabhakar, and B. Yao, "Load Shedding in Stream Databases: A Control-Based Approach," Purdue University, Tech. Rep., 2005.

[36] C. Olston, J. Jiang, and J. Widom, "Adaptive Filters for Continuous Queries over Distributed Data Streams," in *Proceedings of ACM SIGMOD '03*, June 2003, pp. 563–574.

[37] Y.-C. Tu, J. Sun, and S. Prabhakar, "Performance Analysis of A Hybrid Media Streaming System." in *Proceedings of SPIE/ACM MMCN*, January 2004, pp. 69–82.

[38] Y.-C. Tu, J. Sun, M. Hefeeda, and S. Prabhakar, "An Analytical Study of Peer-to-Peer Media Streaming System." *To Appear in ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*.

[39] L. Qu and Y.-C. Tu, "Change Point Estimation of Bi-Level Functions," *To Appear in Journal of Modern Applied Statistical Methods*.

[40] ——, "Change Point Estimation of Bar Code Signal," in *Proceedings of International Conference on Scientifi c Computing*, June 2005, pp. 109–114.

[41] R. Cheng, B. Kao, S. Prabhakar, A. Kwan, and Y.-C. Tu, "Adaptive Stream Filters for Entity-Based Queries with Non-Value Tolerance." in *VLDB*, 2005, pp. 37–48.

[42] W. Aref, A. Catlin, A. Elmagarmid, J. Fan, M. Hammad, I. Ilyas, M. Marzouk, S. Prabhakar, Y.-C. Tu, and X. Zhu., "VDBMS: Testbed Facility for Research in Video Database Benchmarking." in *Proceedings of Intl. Conf. on Distributed Multimedia Systems (DMS)*, 2003, pp. 160–166.

[43] W. Aref, A. Catlin, A. Elmagarmid, J. Fan, J. Guo, M. Hammad, I. Ilyas, M. Marzouk, S. Prabhakar, A. Rezgui, A. Teoh, E. Terzi, Y.-C. Tu, A. Vakali, and X. Zhu., "A Distributed Database Server for Continuous Media (Demo)," in *ICDE*, March 2002, pp. 490–491.

[44] W. Aref, A. Catlin, A. Elmagarmid, J. Fan, M. Hammad, I. Ilyas, M. Marzouk, S. Prabhakar, Y.-C. Tu, and X. Zhu., "VDBMS: Testbed Facility for Research in Video Database Benchmarking." *Springer/ACM Multimedia Systems*, vol. 9, no. 6, pp. 575–585, June 2004.