International Journal of Pattern Recognition
and Artificial Intelligence
Vol. 30, No. 9 (2016) 1660001 (16 pages)

DOI: 10.1142/S0218001416600016

World Scientific
www.worldscientific.com

Queueing Analysis of Continuous Queries for Uncertain Data Streams Over Sliding Windows

Guoqing Xiao[*,§], Kenli Li[*,†,¶], Xu Zhou[*,‖] and Keqin Li[*,†,‡,**]

[*]*College of Information Science and Engineering, Hunan University*
Changsha, Hunan 410082, P. R. China

[†]*National Supercomputing Center in Changsha*
Changsha, Hunan 410082, P. R. China

[‡]*Department of Computer Science, State University of New York*
New Paltz, New York 12561, USA
[§]*xiaoguoqing@hnu.edu.cn*
[¶]*lkl@hnu.edu.cn*
[‖]*zhouxu@126.com*
[**]*lik@newpaltz.edu*

Received 19 November 2015
Accepted 1 March 2016
Published 30 May 2016

With the rapid development of data collection methods and their practical applications, the management of uncertain data streams has drawn wide attention in both academia and industry. System capacity planning and Quality of service (QoS) metrics are two very important problems for data stream management systems (DSMSs) to process streams efficiently due to unpredictable input characteristics and limited memory resource in the system. Motivated by this, in this paper, we explore an effective approach to estimate the memory requirement, data loss ratio, and tuple latency of continuous queries for uncertain data streams over sliding windows in a DSMS. More specifically, we propose a queueing model to address these problems in this paper. We study the average number of tuples, average tuple latency in the queue, and the distribution of the number of tuples and tuple latency in the queue under the Poisson arrival of input data streams in our queueing model. Furthermore, we also determine the maximum capacity of the queueing system based on the data loss ratio. The solutions for the above problems are very important to help researchers design, manage, and optimize a DSMS, including allocating buffer needed for a queue and admitting a continuous uncertain query to the system without violation of the pre-specified QoS requirements.

Keywords: Data management; data streams; QoS; queueing theory; sliding windows; uncertain databases.

1. Introduction

In recent years, uncertain data management has received wide attention with the emergence of a large number of practical applications, such as sensor network,

[¶]Corresponding author.

market surveillance, location-based service (LBS), mobile object tracking and other various applications. Uncertain data is inherent in these applications due to various factors, such as data noise, data leakage, transmission delay, privacy preservation, and inaccuracy or incompletion in measurement, etc. Similarly, surveys and imputation techniques create data which is uncertain in nature. This has created a need for uncertain data management algorithms and applications in both academia and industry,[2] among which pivotal techniques in this respect include query processing over uncertain databases, such as top-k query,[18,33,35] skyline query,[15,24,38] and nearest neighbour query,[9,25] etc.

What's more, data in many scenarios aforementioned are often generated dynamically and continuously with uncertainty which can be modeled as an uncertain (or a probabilistic) stream database.[11,13,20,30,22,23] For instance, locations of objects are usually updated periodically in LBS applications, while many sensors connected to the base stations collect uncertain data continuously in sensor networks. As another example, an online comparative shopping application gets up-to-date goods from multiple websites dynamically. Each good is associated with a *trustability* value which is derived from customers' feedback on product quality, service quality, and surrounding environment, etc. This "trustability" value can also be regarded as existence probability of the product since it represents the probability that the good occurs exactly as described in the website in terms of quality and surrounding environment.[5,37] Thus, *trustability* is an important factor which defines whether customers purchase the product. Generally, this kind of databases can be modeled as an uncertain (or a probabilistic) stream database. For all these applications, the semantics of continuous query processing need to be extensively studied. The queries should allow customers to perform real-time tracking on streaming data, and their results are continuously updated to reflect the change of streams.

Continuous query processing over uncertain data streams, however, is challenging for several reasons.[6] First, the probabilistic streams generated are highly correlated both temporally and spatially. These correlations must be taken into account during query evaluation, since they can dramatically alter the final query results. Second, high-rate data streams should be handled in real-time and efficiently. Third, query semantics become ambiguous since we are dealing with sequences of tuples and not sets of tuples, thereby we have to redefine the query semantics for uncertain streams. It is therefore imperative to design both space- and time-efficient query processing techniques and develop efficient data stream management systems (DSMSs) that can efficiently perform query processing over such streams.

The processing requirements for uncertain streaming databases are very different from traditional database applications, since research on traditional database management systems (DBMSs) mainly concentrates on the data that has been collected and stored. However, the streaming data management applications are required to process data in a single pass and they have to perform queries fast enough in order to keep up with the high-rate data streams. A number of applications even

demand a pre-specified quality of service (QoS), e.g. memory requirement, tuple latency, and data loss ratio, etc., to be be satisfied for answering queries in a timely way.

It is clear that a traditional DBMS, which has little consideration about QoS requirements, is not designed to process uncertain streaming data since it assumes that stored data can be accessed as many times as required. Those new processing requirements are forcing a re-examination of the techniques and methods used in a DBMS. Recently, a number of designated DSMSs have been proposed to support those new requirements for traditional operators,[4,8,10,27] such as *Select*, *Project*, and *Join*, etc. A common feature of these system architectures is that they associate a queue with each operator to support continuous queries over data streams. There is consensus that a queue structure is necessary for effectively dealing with the unpredictable features of a continuous data stream. However, none of them has analyzed how a queue in such a system affects its performance, and how it can be used to satisfy pre-specified QoS requirements.

Jiang *et al.*[21] studied the problems of system capacity planning and QoS verification for traditional unbounded data streams. However, due to the intrinsic differences between uncertain stream databases and traditional stream databases, these methods and techniques aforementioned cannot be applied to process uncertain data streams directly. Additionally, queries on uncertain data streams are very challenging because of the strict space and time requirements of processing both arriving and expiring tuples in high-rate streams, combined with the difficulty of dealing with the exponential blowup in the number of possible worlds induced by the uncertain data model. On the other hand, efficient stream processing techniques are difficult to design because of the unbounded characteristic of data streams. Motivated by this, the focus of this paper is on the analysis of queues used for continuous uncertain streaming data processing as a first step in this direction, including system capacity planing, tuple latency, and data loss ratio, etc.

Our contributions. System capacity planning and QoS metrics are two very important problems for DSMSs to process uncertain data streams efficiently due to unpredictable input characteristics and limited memory resource in the system. This motivated us to explore an effective strategy to estimate the memory requirement, data loss ratio, and tuple latency of continuous queries for uncertain data streams over sliding windows in a DSMS. More specifically, we propose a queueing model to address these problems in this paper. We study the average number of tuples, average tuple latency in the queue, and the distribution of the number of tuples and tuple latency in the queue under the Poisson arrival input data streams in our queueing model. Furthermore, we also determine the relationship between the maximum capacity of the queueing system and the data loss ratio.

The solutions for above problems are very important to help researchers design, manage, and optimize a DSMS, including allocating buffer needed for a queue and admitting a continuous uncertain query to the system without violation of the pre-specified QoS requirements.

The rest of this paper is organized as follows. Section 2 briefly reviews continuous query processing over uncertain data streams and related works about efficient design of data stream management systems. Section 3 introduces the preliminaries about uncertain data stream, sliding-window model, possible world model, and QoS metrics definitions. Section 4 gives the queueing analysis under steady state. Finally, Sec. 5 concludes this paper.

2. Related Work

As continuous query processing for uncertain data streams over sliding-windows in a DSMS is inherently related to continuous uncertain queries (Sec. 2.1) and efficient design of DSMSs (Sec. 2.2), some representative works are summarized in this section.

2.1. *Continuous uncertain queries*

In recent years, with the rapid development of data collection methods and the practical applications, many query processing techniques and algorithms have been proposed to deal with the problem of continuous queries for uncertain data streams in database community.

Following the possible world semantics, researchers studied in succession the aggregate operators, top-k queries, and skyline queries for uncertain data streams based on sliding windows. Aggregates, such as averages, histograms, heavy hitters, etc., over uncertain data streams have been studied recently.[12,19,20] Cormode and Garofalakis[12] presented for the first time space- and time-efficient algorithms for approximating complex aggregate queries, including the number of distinct values and join/self-join sizes, over uncertain data streams. Jayram *et al.*[19,20] proposed algorithms for computing commonly used aggregates over an uncertain data stream. The authors presented the first one pass streaming algorithms for estimating the expected mean of an uncertain stream, and proposed a general method to obtain unbiased estimators for frequency moments for uncertain databases. Aggarwal and Yu[1] provided a general framework for clustering uncertain data stream. Zhang *et al.*[36] designed sampling-based algorithms for probabilistic streaming data to identify all likely frequent items with theoretically guaranteed high probability and accuracy, and the approaches proposed only cost sublinear memory and showed excellent scalability frequent items retrieval in uncertain data streams. Jin *et al.*[22,23] designed a unified framework for processing sliding window top-k queries on uncertain streams. Zhang *et al.*[37] studied the problem of efficient processing of continuous skyline queries over sliding windows on uncertain streaming data for a user-defined probability threshold. Ding *et al.*[16] presented a novel sliding window skyline model where an uncertain tuple may take the probability to be in the skyline at a certain timestamp t. Feng *et al.*[17] presented probabilistic top-k dominating query over sliding windows.

We can see from these query semantics that almost all existing works study query processing on uncertain data streams on the basis of possible world semantics. However, these works focus only on query semantics without taking limited memory, response time, and data loss ratio into account. In this paper, for the first time, we study the QoS required by a continuous query for uncertain data streams over sliding windows based on possible world semantics.

2.2. *Efficient design of DSMSs*

As mentioned before, since the stored data can be accessed as many times as needed, a traditional DBMS cannot be used to process streaming data directly. Those new processing requirements are forcing a re-examination of the techniques and approaches used in a DBMS. From the perspective of a system, a number of different DSMSs have been proposed and investigated to support those new requirements for traditional operators,[4,8,10,27,21,29] such as *Select*, *Project*, and *Join*, etc.

Babu and Widom[4] proposed the *STEAM* architecture which tried to build a general data processing system that could support the functionalities of traditional databases and stream databases. Carney *et al.*[8] presented the *Aurora* architecture which processed data streams with some pre-specified QoS requirements by decoupling continuous queries into several operators. Chen *et al.*[27] proposed the *Fjord* system which supported both continuous data streams and traditional static databases by connecting the push-based operators with the pull-based operators by means of queues. The three works provided a processing framework of data streams from the perspective of macro; and all of them used a queue data structure to perform query processing in real-time or with pre-specified QoS requirements. None of them has taken the problem of how the behavior of a queue affects the performance of the system into account, and how to predict the response time of the system from the perspective of micro. Arasu *et al.*[3] characterized the memory requirements of a continuous query for all possible instances of streams, and their results provided a way to estimate a continuous query within limited memory. But they have not presented an approach to evaluate the tuple latency (i.e. the response time) requirement of a continuous query given the computing resource and memory size of a system. Jiang and Chakravarthy[21] presented the analysis of relational operators used for traditional unbounded data streams using queueing theory and studied the behaviors of streaming data in a continuous query processing system.

However, due to the intrinsic differences between uncertain stream databases and traditional stream databases, these methods and techniques aforementioned cannot be applied to process uncertain data streams directly. In addition, queries on uncertain data streams are very challenging because of the strict space and time requirements of processing both arriving and expiring tuples in high-rate streams, combined with the difficulty of dealing with the exponential blowup in the number of possible worlds induced by the uncertain data model. What's more, efficient stream processing techniques are difficult to design because of the unbounded and

unpredictable characteristic of data streams. Motivated by this, in this paper, we provide analysis of queues for continuous uncertain streaming data query processing for the first time.

3. Preliminaries

In this section, we firstly introduce the uncertain data stream model. Then we present two types of sliding-window model, namely, count-based window and time-based window, in Sec. 3.1. In Sec. 3.2, we provide the basics with respect to an uncertain data model and its corresponding possible world semantics. After that, we proceed to give the formal definitions of the QoS metrics in Sec. 3.3.

Let DS be an uncertain data stream containing a sequence of tuples in a d-dimensional numeric space. Each uncertain tuple t can be represented by a quality vector $(t.1, t.2, \ldots, t.d)$, where $t.i$ $(1 \leq i \leq d)$ denotes the evaluation value of t on the ith quality attribute. $Pr(t)$ $(0 < Pr(t) \leq 1))$ denotes the appearance probability that tuple t exists in DS. Without loss of generality, we assume that the values of $t.i$ is numerical non-negative scores and is normalized in [0,1]. Besides, smaller score values are preferable on each dimension. Let $f(\cdot)$ be a ranking function, for arbitrary two uncertain tuples t and o, we denote $t \prec_f o$ if $f(t) < f(o)$, and we say t's rank is higher than that of o; in a similar fashion, $t \succ_f o$ means t's rank is lower than o's. Without loss of generality, we assume that the ranks of all uncertain tuples are unique. In many applications, a data stream is append-only,[23,26,34] that is, there is no deletion of data element involved before its expiration. In this paper, we study the memory allocation and QoS metrics restricted to the append-only uncertain data stream.

3.1. *Sliding-window model*

Due to the unbounded characteristic of data streams, efficient stream processing techniques are difficult to design. Streaming data is also highly time-sensitive: each (uncertain) tuple arrives with a timestamp, and people are usually more interested in the recent tuples than those in the far past.

There are two types of models for dealing with the time-dependent data streams. One of them is the so-called time-decaying model, which assigns a weight to each tuple that is exponentially decreasing over time. This model usually works together with statistical aggregates, such as mean, median, count, etc., but may not be well defined for some other query processing techniques, such as top-k queries, skyline queries, etc. The other one is the sliding-window model, where people are interested in evaluating queries on tuples that have arrived in, say, the last 24 hours or the most recent N tuples, whereas expired or older tuples are not taken into account since they are of less importance, as illustrated in Fig. 1. In addition, sliding-window queries are generally required to be continuous, that is the users should be warned no matter when the query results change, so that they always have the up-to-date query results for the current sliding window.

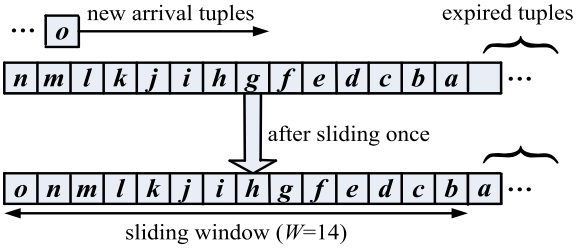

Fig. 1. An example of the sliding-window model.

A sliding window takes a range w (i.e. the window size) and an optional slide s (i.e. the granularity of window sliding) as parameters. It disintegrates an input uncertain data stream into sequences of tuples by sliding the window. In general, there exist two types of basic sliding window, i.e. count-based sliding window and time-based sliding window. In a count-based sliding window, the number of active tuples remains unchanged, thus r new tuples arrive then the r oldest tuples expire. In a time-based sliding window, the number of active tuples may be non-constant, and the set of active tuples is made up of all tuples arrived in the last T run-time instances.

Let us suppose that the tuples from the uncertain data stream are processed in a first in, first out (FIFO) way, that is, the tuple arrived first expires first, too. For simplicity, we assume that N is the sliding window size, and the most recent N items in DS are denoted by DS_N. In the streaming databases, time progresses in every update, and the arrival and expiration time instances of a tuple t satisfy $t.exp = t.arr + N$. Furthermore, we label tuples by integers in terms of the arriving orders of streaming data. Particularly, the order of any tuple t in DS is labeled by an integer $\tau(t)$, which means that t arrives $\tau(t)$th in DS.

3.2. *Possible world model*

Given an uncertain stream database DS composed of a sequences of tuples. Many uncertain data processing pay attention to mainly two types of uncertainty, namely, tuple-level uncertainty and attribute-level uncertainty. For tuple-level uncertainty, every tuple is uncertain while its attribute value is deterministic. For attribute-level uncertainty, every tuple is certain while its attribute value is uncertain, and every attribute value corresponds to a probability. In an uncertain database, there may exist some generation rules between tuples, such as exclusion or coexistence, but in most cases tuples are independent of each other. In this paper, we only consider tuple-level uncertainty with all tuples mutually independent. In practice, almost all tuples are mutually independent in the context of tuple-level uncertainty.

There exist many works on modeling uncertain data. One of the most popular is the model based on possible world semantics,[14] where an uncertain database is regarded as a set of possible world instances associated with their probabilities.

Each possible world $\mathcal{W}$ is a subset of uncertain data objects, and the set of all worlds is denoted by the possible world space Ω. The probability of each world is computed as the joint probability of the existence of the world's objects and the absence of all other data objects. Since all objects are mutually independent, we can obtain:

$$Pr(\mathcal{W}) = \prod_{t \in \mathcal{W}} Pr(t) \cdot \prod_{t \notin \mathcal{W}} (1 - Pr(t)), \tag{1}$$

where $\sum_{\mathcal{W} \in \Omega} Pr(\mathcal{W}) = 1$

3.3. *QoS metrics definition*

Although uncertain data streams tend to be strongly correlated in space and time, the correlations are generally quite structured, with the same set of dependence and independence repeated across time. What's more, most real-life uncertain data streams are Markovian in essence, with the state at time instance "t+1" being independent of the states at previous time instances given the state at time "t". In some situations, the state at time "$t + 1$" may depend on an invariable number of states in the recent past. In general, this is a result of the underlying physical process itself being Markovian essentially.[6,28,31,32] This motivates us to apply a queueing model to study the system capacity planning and QoS requirements in a DSMS.

Tuple latency (i.e. response time), memory requirement, and data loss ratio are fundamental QoS metrics used by most stream-based applications. We propose a queueing model to address these problems on the basis of the Markovian characteristics of uncertain data streams aforementioned. Specifically, we study the average number of uncertain tuples, average tuple latency, and the distribution of the number of tuples and tuple latency in our queueing model, respectively. Moreover, we also study the relationship between the maximum capacity of the queueing system and the data loss ratio. In the following, we give the formal definition of both tuple latency and memory requirement in a DSMS.

***Problem Definition* 1 (*Memory Requirement*).** Given a continuous uncertain query $\mathcal{Q}$ over an uncertain data stream DS, the total memory requirement $\mathcal{C}(t)$ needed by this query $\mathcal{Q}$ depends on the number of the uncertain tuples at time instance t,

$$\mathcal{C}(t) = \hbar \cdot \mathcal{N}(t), \tag{2}$$

where $\hbar$ is the size of an uncertain tuple, and $\mathcal{N}(t)$ is the number of uncertain tuples in the queue.

***Problem Definition* 2 (*Tuple Latency*).** Given a continuous uncertain query $\mathcal{Q}$ over an uncertain data stream DS, the tuple latency (i.e. the response time) $\mathcal{L}(t)$ at time instance t is defined as the summation of the waiting time $\mathcal{W}(t)$ and the service time $\mathcal{S}(t)$ in the queue, i.e.

$$\mathcal{L}(t) = \mathcal{W}(t) + \mathcal{S}(t). \tag{3}$$

However, the waiting time $\mathcal{W}(t)$, the service time $\mathcal{S}(t)$, and the number of uncertain tuples $\mathcal{N}(t)$ in the above definitions are random variables with a continuous parameter, namely, time parameter t. It is very hard to solve the probability distribution function (pdf) for the number of uncertain tuples $\mathcal{N}(t)$ and cumulative distribution function (cdf) for tuple latency $\mathcal{L}(t)$ in the queue. On the other hand, even if we can obtain the solutions on them, the overhead to continuously evaluate their values is extremely large.

Consequently, in this paper, we try to find their approximate mean values, in which they can provide sufficient information to design, optimize, and manage the QoS requirements of DSMSs. As a result, the above problem definitions can be simplified as the following problems that determine their mean values.

***Problem Definition* 3 (*Mean Memory Requirement*).** Given a continuous uncertain query $\mathcal{Q}$ over an uncertain data stream DS, the total mean memory requirement $\overline{\mathcal{C}(t)}$ needed by this query $\mathcal{Q}$ depends on the mean number of the uncertain

Table 1. Frequently used symbols and their descriptions.

Symbol	Description
DS	an uncertain stream database
o	an uncertain tuple in DS
$Pr(o)$	object o existing probability in DS
d	the size of dimensionality
$\mathcal{W}$	a possible world
$Pr(\mathcal{W})$	the probability of $\mathcal{W}$
Ω	possible world space
$f(\cdot)$	the ranking function
$\mathcal{W}_N$	the size of the sliding window
η	the acceptable maximum data loss ratio
$\hbar$	the size of each uncertain tuple
$\mathcal{B}$	the maximum capacity of the queueing system
λ	the arrival ratio of streaming tuples
μ	the service ratio of the queueing system
$\mathcal{C}(t)$	the total memory requirement at time instance t
$\mathcal{N}(t)$	the number of uncertain tuples in the queue at time instance t
$\mathcal{L}(t)$	the tuple latency at time instance t
$\mathcal{W}(t)$	the waiting time at time instance t
$\mathcal{S}(t)$	the service time at time instance t
$\overline{\mathcal{C}(t)}$	the total mean memory requirement at time instance t
$\overline{\mathcal{N}(t)}$	the mean number of tuples in the queue at time instance t
$\overline{\mathcal{L}(t)}$	the mean tuple latency at time instance t
$\overline{\mathcal{W}(t)}$	the mean waiting time at time instance t
$\overline{\mathcal{S}(t)}$	the mean service time at time instance t
$\overline{\mathcal{N}}$	the mean queue length in the steady state
$\overline{\mathcal{N}_q}$	the mean length of waiting queue in the steady state
$\overline{\mathcal{W}}$	the mean sojourn time of each tuple in the steady state
$\overline{\mathcal{W}_q}$	the mean waiting time of each tuple in the steady state
$\overline{\mathcal{F}_q(t)}$	the distribution function of mean waiting time in the steady state

tuples at time instance t,

$$\overline{\mathcal{C}(t)} = E[\mathcal{C}(t)] = \hbar \cdot E[\mathcal{N}(t)] = \hbar \cdot \overline{\mathcal{N}(t)}, \tag{4}$$

where $\hbar$ is the size of an uncertain tuple, and $\overline{\mathcal{N}(t)}$ is the mean number of uncertain tuples in the queue.

***Problem Definition* 4 (*Mean Tuple Latency*).** Given a continuous uncertain query $\mathcal{Q}$ over an uncertain data stream DS, the mean tuple latency (i.e. the mean response time) $\overline{\mathcal{L}(t)}$ at time instance t is defined as the summation of the mean waiting time $\overline{\mathcal{W}(t)}$ and the mean service time $\overline{\mathcal{S}(t)}$ in the queue, i.e.

$$\overline{\mathcal{L}(t)} = E[\mathcal{L}(t)] = E[\mathcal{W}(t)] + E[\mathcal{S}(t)] = \overline{\mathcal{W}(t)} + \overline{\mathcal{S}(t)}. \tag{5}$$

The frequently used symbols and their descriptions in this paper are summarized in Table 1.

4. Steady State Analysis

Generally, we only study the situation of steady state of the queue system. The queue system comes to a steady state if the arrival rate of uncertain tuples is equal to their leaving rate. In this section, we study the queueing model where the arriving process of uncertain data streams conforms to a Possion distribution.

At first, we study the situation of the arriving process of uncertain streams is a Possion stream, i.e. the time interval sequences $\{J_k; k \geq 1\}$ of arrival of uncertain tuples are independent and obey the negative exponential distribution $F(t) = 1 - e^{-\lambda t}$ $(t \geq 0)$ with parameter λ $(\lambda > 0)$, where λ is the average arriving rate of uncertain tuples per unit time. On the other hand, the service time sequences $\{B_k; k \geq 1\}$ of uncertain streaming tuples are also independent and conform to the negative exponential distribution $G(t) = 1 - e^{-\mu t}$ $(t \geq 0)$ with parameter μ $(\mu \geq 0)$, where μ is the average leaving rate of uncertain tuples per unit time, namely the average service rate of the queueing system. The assumption is reasonable since we need to process the exponential blowup in a number of possible worlds induced by the uncertain data model. Furthermore, the arriving process $\{J_k; k \geq 1\}$ and service process $\{B_k; k \geq 1\}$ of uncertain tuples are mutually statistically independent of each other. For a single processor, its buffer size is the maximum capacity of the queueing system, denoted by $\mathcal{B}$. Due to the uncertain data stream being unbounded and infinite, the problems of system capacity planning and QoS metrics can be modeled as an optimization problem of in the $M/M/1/\mathcal{B}/\infty$ queueing system.

Suppose that $\rho = \lambda/\mu$, P_n denotes the probability which has n uncertain tuples in the queueing system. On the basis of the conclusion of the $M/M/1/\mathcal{B}/\infty$ queueing system,[7] the mean waiting time of each uncertain tuple $\overline{\mathcal{W}_q}$ in the steady state is the

following:

$$\overline{\mathcal{W}_q} = \begin{cases} \dfrac{1}{\mu}\left[\dfrac{\rho}{1-\rho} - \dfrac{\mathcal{B}\rho^{\mathcal{B}}}{1-\rho^{\mathcal{B}}}\right], & \rho \neq 1, \\ \dfrac{1}{2\mu}(\mathcal{B}-1), & \rho = 1, \end{cases} \tag{6}$$

the distribution function of mean waiting time $\overline{\mathcal{F}_q(t)}$ in the steady state is:

$$\overline{\mathcal{F}_q(t)} = \begin{cases} 1 - \dfrac{1-\rho}{1-\rho^{\mathcal{B}}} \displaystyle\sum_{n=1}^{\mathcal{B}-1} \rho^n \sum_{r=0}^{n-1} e^{-\mu t} \frac{(\mu t)^r}{r!}, & \rho \neq 1, \\ 1 - \dfrac{1}{\mathcal{B}} \displaystyle\sum_{n=1}^{\mathcal{B}-1} \sum_{r=0}^{n-1} e^{-\mu t} \frac{(\mu t)^r}{r!}, & \rho = 1, \end{cases} \tag{7}$$

the mean sojourn time of each tuple $\overline{\mathcal{W}}$ in the steady state is:

$$\overline{\mathcal{W}} = \overline{\mathcal{W}_q} + \frac{1}{\mu} = \begin{cases} \dfrac{1}{\mu}\left[\dfrac{1}{1-\rho} - \dfrac{\mathcal{B}\rho^{\mathcal{B}}}{1-\rho^{\mathcal{B}}}\right], & \rho \neq 1, \\ \dfrac{1}{2\mu}(\mathcal{B}+1), & \rho = 1, \end{cases} \tag{8}$$

the average queue length $\overline{\mathcal{N}}$ in the system under steady state is:

$$\overline{\mathcal{N}} = \lambda_e \cdot \overline{\mathcal{W}} = \begin{cases} \dfrac{\rho(1-\rho^{\mathcal{B}})}{(1-\rho)(1-\rho^{\mathcal{B}+1})} - \dfrac{\mathcal{B}\rho^{\mathcal{B}+1}}{1-\rho^{\mathcal{B}+1}}, & \rho \neq 1, \\ \dfrac{\mathcal{B}}{2}, & \rho = 1, \end{cases} \tag{9}$$

and the average length of the waiting queue $\overline{\mathcal{N}_q}$ in the queueing system is:

$$\overline{\mathcal{N}_q} = \overline{\mathcal{N}} - (1 - P_0) = \lambda_e \cdot \mathcal{W}_q = \begin{cases} \dfrac{\rho}{1-\rho} - \dfrac{\rho(1+\mathcal{B}\rho^{\mathcal{B}})}{1-\rho^{\mathcal{B}+1}}, & \rho \neq 1, \\ \dfrac{\mathcal{B}(\mathcal{B}-1)}{2(\mathcal{B}+1)}, & \rho = 1, \end{cases} \tag{10}$$

where

$$P_0 = \begin{cases} \dfrac{1-\rho}{1-\rho^{\mathcal{B}+1}}, & \rho \neq 1, \\ \dfrac{1}{\mathcal{B}+1}, & \rho = 1, \end{cases} \tag{11}$$

$$P_n = \begin{cases} \dfrac{(1-\rho)\rho^n}{1-\rho^{\mathcal{B}+1}}, & \rho \neq 1, \\ \dfrac{1}{\mathcal{B}+1}, & \rho = 1, \end{cases} \tag{12}$$

$\lambda_e = \lambda(1 - P_{\mathcal{B}}) = \mu(1 - P_0)$ denotes the effective arriving rate of uncertain streaming tuples, and

$$1 - P_{\mathcal{B}} = \begin{cases} \dfrac{1 - \rho^{\mathcal{B}}}{1 - \rho^{\mathcal{B}+1}}, & \rho \neq 1, \\ \dfrac{\mathcal{B}}{\mathcal{B}+1}, & \rho = 1. \end{cases} \tag{13}$$

Theorem 1. *In the steady state, the mean memory requirement tuple latency* $\overline{\mathcal{C}(t)}$ *in terms of the mean queue length* $\overline{\mathcal{N}}$:

$$\overline{\mathcal{C}(t)} = \hbar \cdot \overline{\mathcal{N}} = \begin{cases} \hbar \cdot \dfrac{\rho(1 - \rho^{\mathcal{B}})}{(1 - \rho)(1 - \rho^{\mathcal{B}+1})} - \dfrac{\mathcal{B}\rho^{\mathcal{B}+1}}{1 - \rho^{\mathcal{B}+1}}, & \rho \neq 1, \\ \hbar \cdot \dfrac{\mathcal{B}}{2}, & \rho = 1, \end{cases} \tag{14}$$

where $\hbar$ is the size of each uncertain tuple.

Theorem 2. *In the steady state, the mean tuple latency* $\overline{\mathcal{L}(t)}$, *namely, the mean sojourn time of an uncertain tuple, is the summation of the mean waiting time and the mean service time*:

$$\overline{\mathcal{L}(t)} = \overline{\mathcal{W}} = \overline{\mathcal{W}_q} + \frac{1}{\mu} = \begin{cases} \dfrac{1}{\mu}\left[\dfrac{2 - \rho}{1 - \rho} - \dfrac{\mathcal{B}\rho^{\mathcal{B}}}{1 - \rho^{\mathcal{B}}}\right], & \rho \neq 1 \\ \dfrac{1}{2\mu}(\mathcal{B} + 3), & \rho = 1 \end{cases}, \tag{15}$$

where $1/\mu$ is the mean service time of each uncertain tuple.

Theorem 3. *In the steady state, the maximum capacity of the queueing system* $\mathcal{B}$ *satisfies*:

$$\begin{cases} \dfrac{(1 - \rho)\rho^{\mathcal{B}}}{1 - \rho^{\mathcal{B}+1}} \leq \eta, & \rho \neq 1 \\ \mathcal{B} \geq \dfrac{1}{\eta} - 1, & \rho = 1 \end{cases}, \tag{16}$$

where η is the acceptable maximum data loss ratio.

Proof. In the steady state, the arrival rate should be equal to the rate of departure. In this case, the data ratio is $P_{\mathcal{B}}$, which denotes the probability that the queueing system is full.

Thus, if η is the acceptable maximum data loss ratio, then

$$P_{\mathcal{B}} \leq \eta$$

should be satisfied.

According to Eq. (12), we have

$$\begin{cases} \dfrac{(1-\rho)\rho^{\mathcal{B}}}{1-\rho^{\mathcal{B}+1}} \leq \eta, & \rho \neq 1 \\ \mathcal{B} \geq \dfrac{1}{\eta} - 1, & \rho = 1 \end{cases}.$$

□

Theorem 4. *Given the computation resource, in the steady state, the maximum input rate the queueing system can handle should satisfy*:

$$\lambda \cdot \frac{1-\rho}{1-\rho^{\mathcal{B}+1}} \cdot \overline{\mathcal{W}_q} + \mu \geq \mathcal{W}_N, \tag{17}$$

where $\mathcal{W}_N$ is the size of the sliding window.

Proof. Assume the average queueing length that maintains the sliding window be $\mathcal{L}_q$, the size of the sliding window be $\mathcal{W}_N$, the service rate is $1/\mu$, then in the steady state, we have

$$\mathcal{L}_q = \mathcal{W}_N - \mu \leq \overline{\mathcal{N}_q},$$

that is,

$$\mathcal{W}_N \leq \overline{\mathcal{N}_q} + \mu = \lambda \cdot \frac{1-\rho}{1-\rho^{\mathcal{B}+1}} \cdot \overline{\mathcal{W}_q} + \mu.$$

□

5. Conclusions

In this paper, we propose a queueing model to analyze theoretically system capacity planning and QoS requirements. We study the average number of tuples, average tuple latency in the queue, and the distribution of the number of tuples and tuple latency in the queue under the Poisson arrival of input data streams in our queueing model. Furthermore, we also determine the maximum capacity of the queueing system based on the data loss ratio. In terms of the queueing model, we can determine whether a DSMS can process a continuous query with QoS requirements over given an uncertain data streams. The estimation of memory requirement provides useful guideline for the buffer allocation. The tuple latency estimation provides an effective guideline to manage and optimize DSMSs.

Acknowledgments

The research was partially funded by the Key Program of National Natural Science Foundation of China (Grant Nos. 61133005, 61432005), the National Natural Science Foundation of China (Grant Nos. 61370095, 61472124), and the International Science & Technology Cooperation Program of China (Grant No. 2015DFA11240), and the National High-tech R&D Program of China (Grant No. 2015AA015303).

References

1. C. C. Aggarwal and P. S. Yu, A framework for clustering uncertain data streams, *Proc. of the 24th Int. Conf. on Data Eng. (ICDE)* (2008), 150–159.
2. C. C. Aggarwal and P. S. Yu, A survey of uncertain data algorithms and applications, *IEEE Trans. Knowl. Data Eng.* **21**(5) (2009) 609–623.
3. A. Arasu, B. Babcock, S. Babu, J. McAlister and J. Widom, Characterizing memory requirements for queries over continuous data streams, *ACM Trans. Database Syst.* **29**(1) (2004) 162–194.
4. S. Babu and J. Widom, Continuous queries over data streams, *ACM Sigmod Record* **30**(3) (2001) 109–120.
5. D. Barbará, H. Garcia-Molina and D. Porter, The management of probabilistic data, *IEEE Trans. Knowl. Data Eng.* **4**(5) (1992) 487–502.
6. K. Bhargav and A. Deshpande, Efficient query evaluation over temporally correlated probabilistic streams, *Proc. of the 25th Int. Conf. Data Eng.* (*ICDE*), 1315–1318, 2009, IEEE.
7. U. N. Bhat, *An Introduction to Queueing Theory-Modeling and Analysis in Applications* (Springer, 2008).
8. D. Carney *et al.*, Monitoring streams: A new class of data management applications, *Proc. of the 28th Int. Conf. Very Large Data Bases (VLDB)*, 215–226, 2002, VLDB Endowment.
9. M. A. Cheema, X. M. Lin, W. Wang, W. J. Zhang and J. Pei, Probabilistic reverse nearest neighbor queries on uncertain data, *IEEE Trans. Knowl. Data Eng.* **22**(4) (2010) 550–564.
10. J. Chen, D. J. DeWitt, F. Tian and Y. Wang, Niagaracq: A scalable continuous query system for internet databases, *ACM SIGMOD Record* **29**(2) (2000) 379–390.
11. R. Cheng, D. V. Kalashnikov and S. Prabhakar, Querying imprecise data in moving object, *IEEE Trans. Knowl. Data Eng.* **16**(9) (2003) 1112–1127.
12. G. Cormode and M. Garofalakis, Sketching probabilistic data streams, *Proc. of the Int. Conf. Manage. Data (SIGMOD)*, 281–292, 2007, ACM.
13. N. Dalvi and D. Suciu, Efficient query evaluation on probabilistic databases, *Proc. of the 30th Int. Conf. Very Large Data Bases (VLDB)*, 864–875, 2004, VLDB Endowment.
14. A. Das Sarma, O. Benjelloun, A. Halevy and J. Widom, Working models for uncertain data, *Proc. of the 22nd Int. Conf. Data Eng. (ICDE)*, 1–12, 2006, IEEE.
15. X. F. Ding and H. Jin, Efficient and progressive algorithms for distributed skyline queries over uncertain data, *IEEE Trans. on Knowl. Data Eng.* **24**(8) (2012) 1448–1462.
16. X. F. Ding, X. Lian, L. Chen and H. Jin, Continuous monitoring of skylines over uncertain data streams, *Inform. Sci.* **184**(1) (2012) 196C–214.
17. X. Feng, X. Zhao, Y. J. Gao and Y. Zhang, Probabilistic top-k dominating query over sliding windows, in *Web Technologies and Applications* (Springer, 2013), pp. 782–793.
18. M. Hua, J. Pei, W. J. Zhang and X. M. Lin, Ranking queries on uncertain data: A probabilistic threshold approach, *Proc. of the ACM Int. Conf. Manage. Data (SIGMOD)*, 673–686, 2008, ACM.
19. T. S. Jayram, S. Kale and E. Vee, Efficient aggregation algorithms for probabilistic data, *Proc. of the 18th SIAM Symposium on Discrete Algorithms (SODA)*, 346–355, 2007, ACM.
20. T. S. Jayram, A. Mcgregor, S. Muthukrishnan and E. Vee, Estimating statistical aggregates on probabilistic data streams, *Acm Trans. Database Systems* **33**(4) (2008) 26.
21. Q. C. Jiang and S. Chakravarthy, Queueing analysis of relational operators for continuous data streams, *Proc. of the 15th Int. Conf. Inf. Knowl. Manage. (CIKM)*, 271–278, 2003, ACM.

22. C. Q. Jin, K. Yi, L. Chen, J. X. Yu and X. M. Lin, Sliding-window top- k queries on uncertain streams, *Proc. of the 34th Int. Conf. Very Large Data Bases (VLDB)*, 301–312, 2008, VLDB Endowment.
23. C. Q. Jin, K. Yi, L. Chen, J. X. Yu and X. M. Lin, Sliding-window top-k queries on uncertain streams, *The VLDB J.* **19**(3) (2010) 411–435.
24. X. Lian and L. Chen, Monochromatic and bichromatic reverse skyline search over uncertain databases, *Proc, of the ACM Int. Conf. Manage Data (SIGMOD)*, 213–226, 2008, ACM.
25. X. Lian and L. Chen, Probabilistic group nearest neighbor queries in uncertain databases, *IEEE Trans. Knowl. Data Eng.* **20**(6) (2008) 809–824.
26. X. M. Lin, Y. D. Yuan, W. Wang and H. J. Lu, Stabbing the sky: Efficient skyline computation over sliding windows, *Proc. of the 21st Int. Conf. Data Eng. (ICDE)*, 502–513, 2005, IEEE.
27. S. Madden and M. J. Franklin, Fjording the stream: An architecture for queries over streaming sensor data, *Proc. of the 18th Int. Conf. Data Eng. (ICDE)*, 555–566, 2002, IEEE.
28. J. Mei, K. L. Li, A. J. Ouyang and K. Q. Li, A profit maximization scheme with guaranteed quality of service in cloud computing, *IEEE Trans. Comput.* **64**(11) (2015) 3064–3078.
29. A. J. Ouyang, K. L. Li, X. W. Fei, X. Zhou and M. X. D, A novel hybrid multi-objective population migration algorithm, *Int. J. Pattern Recogn.* **29**(01) (2015) 1559001.
30. C. Ré, N. Dalvi and D. Suciu, Efficient top-k query evaluation on probabilistic data, *Proc. of the 23rd Int. Conf. Data Eng. (ICDE)*, 886–895, 2007.
31. P. Sen and A. Deshpande, Representing and querying correlated tuples in probabilistic databases, *Proc. of the 23rd Int. Conf. Data Engineering (ICDE)*, 596–605, 2007, IEEE.
32. P. Sen, A. Deshpande and L. Getoor, Exploiting shared correlations in probabilistic databases, *Proc. of the 34th Int. Conf. Very Large Data Bases (VLDB)*, 809–820, 2008, VLDB Endowment.
33. M. A. Soliman, I. F. Ilyas and K. C. C. Chang, Top-k query processing in uncertain databases, *Proc. of the 23rd Int. Conf. Data Eng. (ICDE)*, 896–905, 2007, IEEE.
34. Y. F. Tao and D. Papadias, Maintaining sliding window skylines on data streams, *IEEE Trans. Knowl. Data Eng.* **18**(3) (2006) 377–391.
35. G. Q. Xiao, K. L. Li, K. Q. Li and X. Zhou, Efficient top-(k, l) range query processing for uncertain data based on multicore architectures, *Distrib. Parallel Dat.* **33**(3) (2015) 381–413.
36. Q. Zhang, F. F. Li and K. Yi, Finding frequent items in probabilistic data, *Proc. of the Int. Conf. Management of Data (SIGMOD)*, 819–832, 2008, ACM.
37. W. J. Zhang, X. M. Lin, Y. Zhang, W. Wang and J. X. Yu, Probabilistic skyline operator over sliding windows, *Proc. of the 25th Int. Conf. Data Eng. (ICDE)*, 1060–1071, 2009, IEEE.
38. X. Zhou, K. L. Li, Y. T. Zhou and K. Q. Li, Adaptive processing for distributed skyline queries over uncertain data, *IEEE Trans. Knowl. Data Eng.* **28**(2) (2016) 371–384.

Guoqing Xiao is currently pursuing his Ph.D. in the Department of Information Science and Engineering, Hunan University, Changsha, China. His research interests include uncertain data management, parallel and distributed computing.

Kenli Li received his Ph.D. in Computer Science from Huazhong University of Science and Technology, China, in 2003. He is currently a Professor of Computer Science and Technology at Hunan University and the Deputy Director of National Supercomputing Center in Changsha. His major research contains data management, parallel computing, grid and cloud computing, and DNA computer. He currently serves on the editorial boards of IEEE Trans. on Computers and International Journal of Pattern Recognition and Artificial Intelligence.

Xu Zhou received her master's degree from the Department of Information Science and Engineering, Hunan University, in 2009. She is currently pursuing her Ph.D. in the Department of Information Science and Engineering, Hunan University, Changsha, China. Her research interests include parallel computing and data management.

Keqin Li is a SUNY Distinguished Professor of Computer Science. His current research interests include data management, parallel computing and distributed computing. He has published over 350 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He is currently or has served on the editorial boards of IEEE Trans. on Parallel and Distributed Systems, IEEE Trans. on Computers and IEEE Trans. on Cloud Computing. He is an IEEE Fellow.