StatStream: Statistical Monitoring of Thousands of Data Streams in Real Time

Yunyue Zhu Dennis Shasha
yunyue@cs.nyu.edu shasha@cs.nyu.edu
Courant Institute of Mathematical Sciences
Department of Computer Science
New York University

November 4, 2001

Abstract

Consider the problem of monitoring tens of thousands of time series data streams in an online fashion and making decisions on them. In addition to single stream statistics such as average and standard deviation, we also want to track correlations between all pairs of streams. A stock market trader might use such a tool to spot arbitrage opportunities. This paper proposes efficient methods for solving this problem based on Discrete Fourier Transforms and a three level time interval hierarchy. Extensive experiments on synthetic data and real world financial trading data show that our algorithm beats the direct computation approach by several orders of magnitude. It also improves on previous Fourier Transform approaches by allowing the efficient computation of time-delayed correlation over any size sliding window and any time delay across streams or within a single stream. Correlation also lends itself to an efficient grid-based data structure. The result is a fully on-line, incremental, fixed response time, approximate yet accurate system for monitoring the pairwise correlations of 10,000 streams on a single PC. The algorithm is embarrassingly parallelizable.

1 Introduction

Many applications consist of multiple data streams. For example,

- In mission operations for NASA's Space Shuttle, approximately 20,000 sensors are telemetered once per second to Mission Control at Johnson Space Center, Houston[15].
- There are about 50,000 securities trading in the United States, and every second up to 100,000 quotes and trades (ticks) are generated.

Unfortunately it is difficult to process such data in set-oriented data management systems, though object-relational time series extensions have begun to fill the gap in a logical sense at least[20]. For the performance to be sufficiently good however, "Data Stream Management Systems" (DSMSs) [3], whatever their logical model, should exploit the following characteristics of the application:

- Updates are through insertions of new elements (with relatively rare corrections of older data).

- Queries (moving averages, standard deviations, and correlation) treat the data as sequences not sets.
- Since a full stream is never materialized, queries treat the data as a never-ending data stream.
- One pass algorithms are desirable because the data is vast.
- Interpretation is mostly qualitative, so sacrificing accuracy for speed is acceptable.

This paper presents the algorithms and architecture of StatStream, a data stream management system. The system computes a variety of single and multiple stream statistics in one pass with constant time (per input) and bounded memory. The statistics we considered in our framework include most of the statistics that a securities trader might be interested in, just to show its use for one practical application. The algorithms, however, are applicable to other disciplines, such as sensor data processing and medicine. We divide our contributions into functional and algorithmic. Our functional contributions are:

1. We compute multi-stream statistics such as synchronous as well as time-delayed correlation and vector inner-product in a continuous online fashion. This means that if a statistic holds at time t, that statistic will be reported at time $t + v$, where v is a constant independent of the size and duration of the stream.
2. For any pair of streams, each pair-wise statistic is computed in incremental fashion and requires constant time per update. This is done using a Discrete Fourier Transform approximation.
3. The approximation has a small error under natural assumptions.
4. Even when we monitor the data streams over sliding windows, no revisiting of the expiring data streams is needed.
5. The net result is that on a Pentium 4 PC, we can handle 10,000 streams with a delay window v of only 5 minute.

Our algorithmic contributions mainly have to do with correlation statistics. First, we distinguish three time periods:

- timepoints – the smallest unit of time over which the system collects data, e.g. second.
- basic window – a consecutive subsequence of time points over which the system maintains a digest incrementally, e.g., a few minutes.
- sliding window – a user-defined consecutive subsequence of basic windows over which the user wants statistics, e.g. an hour. The user might ask, "which stocks were correlated with a value of over 0.9 for the last hour?"

The intermediate time interval that we call basic windows yields three advantages:

1. Results of user queries need not be delayed more than the basic window time. In our example, the user will be told about correlations between 2 PM and 3 PM by 3:05 PM.
2. The basic window allows the computation of correlations over windows of arbitrary size as well as time-delayed correlations with high accuracy.

3. The size of the basic window can be adjusted to trade speed for accuracy. In our experiments we construct the size to be the largest that ensures that correlations are calculated with an error of 0.01.

A second algorithmic contribution is the grid structure each of whose cells store the hash function value of a stream. The structure itself is unoriginal but the high efficiency we obtain from it is due to the facts that we are measuring correlation and have done the time decomposition mentioned above.

The remainder of this paper will be organized as follows. The data we consider and statistics we produce are presented in Section 2. Section 3 presents our algorithms for monitoring high speed time series data streams. Section 4 discusses the system StatStream. Section 5 presents our experimental results. Section 6 puts our work in the context of related work.

2 Data And Queries

2.1 Time Series Data Streams

We consider data entering as a time ordered series of triples (streamID, timepoint, value). Each stream consists of all those triples having the same streamID. (In finance, a streamID may be a stock, for example.) The streams are synchronized.

Each stream has new value available at every periodic time interval, e.g. every second. We call the interval value the **timepoint**. For example, if the periodic time interval is a second and the current timepoint for all the streams is i, after one second, all the streams will have a new value with timepoint $i+1$. (Note that if a stream has no value at a timepoint, a value will be assigned to that timepoint based on interpolation. If there are several values during a timepoint, then a summary value will be assigned to that timepoint.)

Let s_i or $s[i]$ denote the value of stream s at timepoint i. $s[i:j]$ denotes the subsequence of stream s from timepoints i through j inclusive. s^i denotes a stream with streamID i. Also we use t to denote the latest timepoint, i.e., now. The statistics we will monitor will be denoted $stat(s_j^{i_1}, s_j^{i_2}, ..., s_j^{i_k}, j \in windows)$. We will discuss the meaning of windows in the next section.

2.2 Temporal Spans

In the spirit of the work in [9, 10], we generalize the three kinds of temporal spans for which the statistics of time series are calculated.

1. **Landmark windows:** In this temporal span, statistics are computed based on the values between a specific timepoint called landmark and now. $stat(s, landmark(k))$ will be computed on the subsequence of time series $s[i], i \geq k$. An unrestricted window is a special case when $k = 1$. For an unrestricted window the statistics are based on all the available data.

2. **Sliding windows:** In financial applications, at least, a sliding window model is more appropriate for data streams. Given the length of the sliding window w and the current timepoint t, $stat(s, sliding(w))$ will be computed in the subsequence $s[t-w+1:t]$.

3. **Damped window model:** In this model, recent sliding windows are more important than previous ones. For example, in the computation of a moving average, a sliding window model will compute the average as

$$avg = \frac{\sum_{i=t-w+1}^{t} s_i}{w}$$

In a damped window model, by contrast, the weights of data decrease exponentially into the past. For example, a moving average in a damped window model can be computed as follows:

$$avg_{new} = avg_{old} * p + s_t * (1-p), 0 < p < 1$$

Other statistics in a damped window model can be defined similarly.

In this paper, we will focus on the sliding window model, because it is the one used most often. Also, the algorithms for sliding windows are the most general, so it can be specialized to the other two temporal spans.

2.3 Statistics To Monitor

Consider the stream $s_i, i = 1, ..., w$. The statistics we will monitor are

1. Average

$$avg(s) = \frac{\sum_{i=1}^{w} s_i}{w}$$

2. Standard deviation

$$std(s) = \sqrt{\frac{\sum_{i=1}^{w} s_i^2 - w avg(s)^2}{w-1}}$$

3. Maximum and Minimum
4. Best Fit Slope: the slope of the line that best fit the time series.

$$bfs(s) = \frac{\frac{1}{w}\sum_{i=1}^{w} s_i i - avg(i)avg(s)}{\frac{1}{w}\sum_{i=1}^{w} i^2 - avg(i)^2}$$

5. Correlation coefficients

$$corr(s,r) = \frac{\frac{1}{w}\sum_{i=1}^{w} s_i r_i - avg(s)avg(r)}{std(s)std(r)}$$

6. Autocorrelation: the correlation of the series with itself at an earlier time.
7. Beta: the sensitivity of the value of a stream s to the values of another stream r (or weighted collection of streams).

$$beta(s,r) = \frac{\frac{1}{w}\sum_{i=1}^{w} s_i r_i - avg(s)avg(r)}{std(r)^2}$$

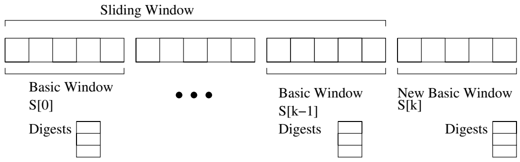

Figure 1: Sliding windows and basic windows

3 Statistics Over Sliding Windows

To compute the statistics over a sliding window, we will maintain a synopsis data structure for the stream to compute the statistics rapidly. To start, our framework subdivides the sliding windows equally into shorter windows, which we call **basic windows**, to facilitate the efficient elimination of old data and the incorporation of new data. We keep digests for both basic windows and sliding windows. For example, the running sum of the time series values within a basic window and the running sum within an entire sliding window belong to the two kinds of digests respectively. Figure 1 shows the relation between sliding windows and basic windows.

Let the data within a sliding window be $s[t-w+1:t]$. Suppose $w = kb$, where b is the length of a basic window and k is the number of basic windows within a sliding window. Let $S[0], S[1], ..., S[k-1]$ denote a sequence of basic windows, where $S[i] = s[(t-w)+ib+1:(t-w)+(i+1)b]$. $S[k]$ will be the new basic window and $S[0]$ is the expiring basic window. The j-th value in the basic window $S[i]$ is $S[i;j]$.

The size of the basic window is important because it must be possible to report all statistics for basic window i to the user before basic window $i+1$ completes (when it will be necessary to report the statistics for window $i+1$).

3.1 Simple Statistics

As a concrete example, consider the computation of the moving average. Since the moving average always involves w points, the information to be maintained is $\sum(s[t-w+1:t])$. For each basic window $S[i]$, we maintain the digest $\sum(S[i])$. After b new data points from the stream become available, we compute the sum over the new basic window $S[k]$.

The sum over the sliding window is updated as follows.

$$\sum_{new}(s) = \sum_{old}(s) + \sum S[k] - \sum S[0]$$

The computations of minimum, maximum and standard deviation are similar.

For the computation of the best fit slope, it is sufficient to compute the following digest based on the sliding window:

$$\phi(s) = \sum_{i=1}^{w} i s_{t-w+i}$$

We maintain the following digests on the basic windows:

$$\tau(S[i]) = \sum_{j=1}^{b} S[i;j]j$$

Theorem 1 *The incremental maintenance of $\phi(s)$ can be achieved by use of the following equation:*

$$\phi^{new}(s) = \phi^{old}(s) + w\sum S[k] - b\sum_{new}(s) + \tau(S[k]) - \tau(S[0])$$

Proof We have

$$\phi^{old}(s) = \sum_{i=0}^{k-1}\left(\tau(S[i]) + ib\sum(S[i])\right)$$

$$\phi^{new}(s) = \sum_{i=0}^{k-1}\left(\tau(S[i+1]) + ib\sum(S[i+1])\right)$$

$$\phi^{new}(s) - \phi^{old}(s)$$

$$= \tau(S[k]) - \tau(S[0]) + (k-1)b\sum(S[k]) - b\sum_{i=1}^{k-1}\sum(S[i])$$

$$= \tau(S[k]) - \tau(S[0]) + kb\sum(S[k]) - b\sum_{i=1}^{k}\sum(S[i])$$

∎

3.2 Correlation Statistics

The efficient computation of correlation and beta potentially requires the examination of all pairs of streams. To compute the correlation coefficients efficiently, we need to employ some data reduction techniques. We will use the Discrete Fourier Transform.

In contrast to the simple statistics we considered in the last section, the computation of correlation requires information on the shape of the time series, instead of just some cumulative information. Our general approach is to use curve fitting in each basic window to reduce the data size. In almost all applications, end users are interested only in pairs that are highly correlated. We will also show how report these pairs without examining all pairs of streams using DFT and hash techniques.

We start by a quick review of DFT. We will explain how to compute the vector inner-product when the two series are aligned in basic windows. This approach is extended to the general case when the basic windows are not aligned. Then we will show our approach for reporting the highly correlated stream pairs in an online fashion.

3.3 The Discrete Fourier Transform

In this section, we review the Discrete Fourier Transform and some of its properties. We will follow the convention in [2]. The Discrete Fourier Transform of a time sequence $x = x_0, x_1, ..., x_{w-1}$ is a sequence $X = X_0, X_1, ..., X_{w-1} = DFT(x)$ of complex numbers given by

$$X_F = \frac{1}{\sqrt{w}}\sum_{i=0}^{w-1} x_i e^{-j2\pi Fi/w} \qquad F = 0, 1, ..., w-1$$

where $j = \sqrt{-1}$. The inverse Fourier transform of X is given by

$$x_i = \frac{1}{\sqrt{w}}\sum_{F=0}^{w-1} x_F e^{j2\pi Fi/w} \qquad i = 0, 1, ..., w-1$$

The following properties of DFT can be found in any textbook on DFT.

- The DFT preserves the Euclidean distance between two sequence x and y.

$$d(x, y) = d(X, Y)$$

- (Symmetry Property) If x is real sequence, then

$$X(i) = X^*(w - i), \qquad i = 1, 2, ..., w - 1$$

  X^* is the complex conjugate of X.

Since for most real time series the first few coefficients contain most of the energy, we would then expect those coefficients to capture the raw shape of the time series[2, 8]. For example, the energy spectrum for the *random walk* series, which models stock movements, declines with a power of 2 with increasing coefficients. And for *black noise*, which successfully models series like the water level of a river as it varies over time, the energy spectrum declines even faster with increasing number of coefficients.

3.4 Inner-product With Aligned Windows

The vector inner-product for two series $x = (x_1, ..., x_w), y = (y_1, ..., y_w)$, denoted as $\psi(x, y)$, is just the sum of the products of their corresponding elements:

$$\psi(x, y) = \sum_{i=1}^{w} x_i y_i$$

Given two series s^x and s^y, when the two series are aligned,

$$\psi(s^x, s^y) = \sum_{i=1}^{k} \psi(S^x[i], S^y[i])$$

So, we must explain how to compute the inner-product of two basic windows: $x_1, x_2, ...x_b$ and $y_1, y_2, ...y_b$.

Let $f : f_1(x), f_2(x), ...$ be a family of continuous functions. We approximate the time series in each basic window, $S^x[i] = x_1, x_2, ...x_b$ and $S^y[i] = y_1, y_2, ...y_b$, with a function family f. (We will give specific examples later.)

$$x_i \approx \sum_{m=0}^{n-1} c_m^x f_m(i), y_i \approx \sum_{m=0}^{n-1} c_m^y f_m(i)$$

$$i = 1, ..., b$$

$c_m^x, c_m^y, m = 0, ..., n - 1$ are n coefficients to approximate the time series with the function family f.

The inner-product of the two basic windows is therefore

$$\sum_{i=1}^{b} x_i y_i \approx \sum_{i=1}^{b} \Big(\sum_{m=0}^{n-1} c_m^x f_m(i) \sum_{p=0}^{n-1} c_p^y f_p(i) \Big)$$

$$= \sum_{m=0}^{n-1} \sum_{p=0}^{n-1} c_m^x c_p^y \Big(\sum_{i=1}^{b} f_m(i) f_p(i) \Big)$$

$$= \sum_{m=0}^{n-1} \sum_{p=0}^{n-1} c_m^x c_p^y W(m,p)$$

where $W(m,p) = \sum_{i=1}^{b} f_m(i) f_p(i)$ can be precomputed. If the function family f is orthogonal, we have

$$W(m,p) = \begin{cases} 0 & m \neq p \\ V(m) \neq 0 & m = p \end{cases}$$

Thus,

$$\sum_{i=1}^{b} x_i y_i \approx \sum_{m=0}^{n-1} c_m^x c_m^y V(m)$$

With this curve fitting technique, we reduce the space and time required to compute inner-products from b to n. It should also be noted that besides data compression, the curve fitting approach can be used to fill in missing data. This works naturally for computing correlations among streams with missing data at some timepoints.

Triangular function families have the right properties. We perform the Discrete Fourier Transforms for the time series over the basic windows, enabling a constant time computation of coefficients for each basic window. Following the observations in [2], we can obtain a good approximation for the series with only the first n DFT coefficients,

$$x_i \approx \frac{1}{\sqrt{b}} \sum_{F=0}^{n-1} X_F e^{j2\pi F i/b} \qquad i = 1, 2, ..., b$$

3.5 Inner-product With unaligned windows

When we compute the correlation between two streams with time lags, including autocorrelations, the two time series will not necessary be aligned at their basic windows. However, the digests we keep are enough to compute such correlations.

Without loss of generality, we will show the computation of *the n-approximate lagged inner-product* of two streams with time lags less than the size of a basic window. Given two such series, $s^x = s_1^x, ..., s_w^x = S^x[1], ..., S^x[k]$ and $s^y = s_{a+1}^y, ..., s_{a+w}^y = S^y[0], S^y[1], ..., S^y[k-1], S^y[k]$, where for the basic windows $S^y[0]$ and $S^y[k]$, only the last a values in $S^y[0]$ and the first a values in $S^y[k]$ are included.($a < b$) We have

$$\psi(s^x, s^y) = \sum_{i=1}^{w} (s_i^x s_{a+i}^y)$$

$$= \sum_{i=1}^{k} \big(\rho(S^x[i], S^y[i-1], a) + \rho(S^y[i], S^x[i], a)\big)$$

where

$$\rho(S^1, S^2, a) = \sum_{i=1}^{a} S^1[i] S^2[b-a+i]$$

For $S^1 = x_1, ..., x_b$ and $S^2 = y_1, ..., y_b$, $\rho(S^1, S^2, a)$ is the inner-product of the first a values of S^1 with the last a values of S^2. This can be computed using only their first n DFT coefficients.

$$\sum_{i=1}^{a} x_i y_{b-a+i} \approx \sum_{i=1}^{a} \Big(\sum_{m=0}^{n-1} c_m^x f_m(i) \sum_{p=0}^{n-1} c_p^y f_p(b-a+i) \Big)$$

$$= \sum_{m=0}^{n-1} \sum_{p=0}^{n-1} c_m^x c_p^y \big(\sum_{i=1}^{a} f_m(i) f_p(b-a+i) \big)$$

$$= \sum_{m=0}^{n-1} \sum_{p=0}^{n-1} c_m^x c_p^y W(m,p,a)$$

This implies that if we precompute the table of

$$W(m,p,a) = \sum_{i=1}^{a} f_m(i) f_p(b-a+i)$$

$$m,p = 0...,n-1; a = 1,...,\lfloor b/2 \rfloor$$

we can compute the inner-product using only the DFT coefficients in time $O(n^2)$ without requiring the alignment of basic windows.

Theorem 2 *The n-approximate lagged inner-product of two time series using their first n DFT coefficients can be computed in time $O(kn)$ if the two series have aligned basic windows, otherwise it takes time $O(kn^2)$, k is the number of basic windows.*

It is not hard to show that this approach can be extended to compute the inner-product of two time series over sliding window of any size.

Corollary 1 *The inner-product of two time series over sliding windows of any size with any time delay can be computed using only the basic windows digests of the data streams.*

3.6 IO Performance

It might be desirable to store the summary data for future analysis. Since the summary data we keep are sufficient to compute all the statistics we are interested in, there is no need to store the raw data stream. The summary data will be stored in disk sequentially by the order of basic windows.

Let N_s be the number of streams, N_b be the number of basic windows, S_b be the size of basic windows and N_c be the number of DFT coefficients we use($2N_c$ real number), the I/O cost to access the data for all streams within a specific period will be

$$\frac{N_s N_b Sizeof(float)(2+2N_c)}{PageSize}$$

while the I/O cost for the exact computation is

$$\frac{N_s N_b Sizeof(float) * S_b}{PageSize}$$

The improvement is a ratio of

$$\frac{S_b}{2+2N_c}$$

The first 2 in the above equations corresponds to two non-DFT elements of summary data: the sum and the sum of the squares of the time series in each basic window. Also the I/O costs above assume sequential disk access, a reasonable assumption given the time-ordered nature of data streams.

3.7 Monitoring Correlations Between Data Streams

The above curve fitting technique can be use for computing inner-products and correlations over sliding windows of any size, as well as with any time delay across streams. To do online monitoring of synchronized streams over a fixed size of sliding window with correlation above a specific threshold, we use an approach based on DFT and a hash technique that will report such stream pairs quickly.

First, we introduce the normalization of a series over sliding windows of size w, $x_1, x_2, ..., x_w$, as follows.

$$\hat{x}_i = \frac{x_i - \overline{x}}{\sigma_x}, \qquad i = 1, 2, ..., w$$

where

$$\sigma_x = \sqrt{\sum_{i=1}^{w} (x_i - \overline{x})^2}$$

As the following theorem suggests, the correlation coefficient of two time series can be reduced to the Euclidean distance between their normalized series.

Theorem 3 *The correlation coefficient of two time series $x_1, ..., x_w$ and $y_1, ..., y_w$ is*

$$corr(x, y) = 1 - \frac{1}{2} d^2(\hat{x}, \hat{y})$$

where $d(\hat{x}, \hat{y})$ is the Euclidean distance between $\hat{x}$ and $\hat{y}$.

Proof First, we notice that

$$\sum_{i=1}^{w} \hat{x}_i^2 = \sum_{i=1}^{w} \hat{y}_i^2 = 1$$

$$corr(x, y) = \frac{\sum_{i=1}^{w} (x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{w} (x_i - \overline{x})^2} \sqrt{\sum_{i=1}^{w} (y_i - \overline{y})^2}} = \sum_{i=1}^{w} \hat{x}_i \hat{y}_i$$

$$\begin{aligned} d^2(\hat{x}, \hat{y}) &= \sum_{i=1}^{w} (\hat{x}_i - \hat{y}_i)^2 = \sum_{i=1}^{w} \hat{x}_i^2 + \sum_{i=1}^{w} \hat{y}_i^2 - 2 \sum_{i=1}^{w} \hat{x}_i \hat{y}_i \\ &= 2 - 2corr(x, y) \end{aligned}$$

∎

By reducing the correlation coefficient to Euclidean Distance, we can apply the techniques in [2, 22] to report sequences with correlation coefficients higher than a specific threshold.

Theorem 4 *Let the DFT of the normalized form of two time series x and y be $\hat{X}$ and $\hat{Y}$ respectively,*

$$corr(x, y) \geq 1 - \epsilon^2 \Rightarrow d_n(\hat{X}, \hat{Y}) \leq \epsilon$$

where $d_n(\hat{X}, \hat{Y})$ is the Euclidean distance between series $\hat{X}_1, \hat{X}_2, ..., \hat{X}_n$ and $\hat{Y}_1, \hat{Y}_2, ..., \hat{Y}_n$.

Proof As DFT preserves Euclidean distance, we have

$$d(\hat{X},\hat{Y}) = d(\hat{x},\hat{y})$$

Using only the first n and last $n, (n << w)$ DFT coefficients[2, 22], from the symmetry property of DFT, we have

$$\sum_{i=1}^{n}(\hat{X}_i - \hat{Y}_i)^2 + \sum_{i=1}^{n}(\hat{X}_{w-n} - \hat{Y}_{w-n})^2$$

$$= 2\sum_{i=1}^{n}(\hat{X}_i - \hat{Y}_i)^2 = 2d_n^2(\hat{X},\hat{Y}) \leq d^2(\hat{X},\hat{Y})$$

$$corr(x,y) \geq 1-\epsilon^2 \Rightarrow d^2(\hat{x},\hat{y}) \leq 2\epsilon^2$$

$$\Rightarrow d^2(\hat{X},\hat{Y}) \leq 2\epsilon^2 \Rightarrow d_n(\hat{X},\hat{Y}) \leq \epsilon$$

▮

From the above theorem, we can examine the correlations of only those stream pairs for which $d_n(\hat{X},\hat{Y}) \leq \epsilon$ holds. We will get a superset of highly correlated pairs and there will be no false negatives.

We can extend the techniques above to report stream pairs of high negative correlation.

Corollary 2 *Let the DFT of the normalized form of two time series x and y be $\hat{X}$ and $\hat{Y}$.*

$$corr(x,y) \leq -1+\epsilon^2 \Rightarrow d_n(-\hat{X},\hat{Y}) \leq \epsilon$$

Proof We have

$$DFT(-x) = -DFT(x)$$

$$corr(x,y) \leq -1+\epsilon^2 \Rightarrow \sum_{i=1}^{w}\hat{x}_i\hat{y}_i \leq -1+\epsilon^2$$

$$\Rightarrow \sum_{i=1}^{w}(-\hat{x}_i)\hat{y}_i \geq 1-\epsilon^2 \Rightarrow d_n(-\hat{X},\hat{Y}) \leq \epsilon$$

▮

Now we discuss how to compute the DFT coefficients $\hat{X}$ incrementally. The DFT coefficients $\hat{X}$ of the normalized sequence can be computed from the DFT coefficients X of the original sequence.

Lemma 1 *Let $\hat{X} = DFT(\hat{x}), X = DFT(x)$, we have*

$$\begin{cases} \hat{X}_0 = 0 \\ \hat{X}_i = \frac{X_i}{\sigma_x} \quad i \neq 0 \end{cases}$$

We can maintain DFT coefficients over sliding windows incrementally[11].

Theorem 5 *Let X_m^{old} be the m-th DFT coefficient of the series in sliding window $x_0, x_1, ..., x_{w-1}$ and X_m^{new} be that coefficient of the series $x_1, x_2, ..., x_w$,*

$$X_m^{new} = e^{\frac{j2\pi m}{w}}(X_m^{old} + \frac{x_w - x_0}{\sqrt{w}})$$

$$m = 1, ..., n$$

This can be extended to a batch update based on the basic windows.

Corollary 3 *Let X_m^{old} be the m-th DFT coefficient of the series in sliding window $x_0, x_2, ..., x_{w-1}$ and X_m^{new} be that coefficient of the series $x_b, x_{b+1}, ..., x_w, x_{w+1}, ..., x_{w+b-1}$,*

$$X_m^{new} = e^{\frac{j2\pi mb}{w}} X_m^{old} + \frac{1}{\sqrt{w}}\left(\sum_{i=0}^{b-1} e^{\frac{j2\pi m(b-i)}{w}} x_{w+i} - \sum_{i=0}^{b-1} e^{\frac{j2\pi m(b-i)}{w}} x_i\right)$$

$$m = 1, ..., n$$

The corollary suggests that to update the DFT coefficients incrementally, we should keep the following n digests for the basic windows.

$$\sum_{i=0}^{b-1} e^{\frac{j2\pi m(b-i)}{w}} x_i, \qquad m = 1, ..., n$$

By using the DFT on normalized sequences, we also map the original sequences into a bounded feature space.

Theorem 6 *Let $\hat{X}_0, \hat{X}_1, ..., \hat{X}_{w-1}$ be the DFT of a normalized sequence $x_1, x_2, ..., x_w$, we have*

$$|\hat{X}_i| \leq \frac{\sqrt{2}}{2}, \qquad i = 1, ..., n, n < w/2$$

Proof

$$\sum_{i=1}^{w-1} (\hat{X}_i)^2 = \sum_{i=1}^{w} (\hat{x}_i)^2 = 1$$

$$\Rightarrow 2\sum_{i=1}^{n} \hat{X}_i^2 = \sum_{i=1}^{n} (\hat{X}_i^2 + \hat{X}_{w-i}^2) \leq 1$$

$$\Rightarrow |\hat{X}_i| \leq \frac{\sqrt{2}}{2}, \qquad i = 1, ..., n$$

∎

From the above theorem, the DFT feature space R^{2n} is a cube of diameter $\sqrt{2}$. Thus we can use a grid structure to report near neighbors efficiently.

We will use the first $\hat{n}$, $\hat{n} \leq 2n$, dimensions of the DFT feature space for indexing. We superimpose a $\hat{n}$-dimensional orthogonal regular grid on the DFT feature space and partition the cube of diameter $\sqrt{2}$ into cells with the same size and shape. There are $(2\lceil \frac{\sqrt{2}}{2\epsilon} \rceil)^{\hat{n}}$ cells of cubes of diameter ϵ. Each stream is mapped to a cell based on its first $\hat{n}$ normalized DFT coefficients. Suppose a stream x is hashed to cell $(c_1, c_2, ..., c_{\hat{n}})$. To report the streams whose correlation coefficients with x is above the threshold $1 - \epsilon^2$, only streams hashed to cells adjacent to cell $(c_1, c_2, ..., c_{\hat{n}})$ need to be examined. Similarly, streams whose correlation coefficients with x is less than the threshold $-1 + \epsilon^2$, must be hashed to cells adjacent to cell $(-c_1, -c_2, ..., -c_{\hat{n}})$. After hashing the streams to cells, the number of stream pairs to be examined is greatly reduced. We can then compute their Euclidean distance, as well as correlation, based on the first n DFT coefficients.

3.8 Parallel Implementation

Our framework facilitates a parallel implementation by using a straightforward decomposition. Consider a network of k servers to monitor the N_s streams. We assume these servers have similar computing resources.

The work to monitor the streams has two stages.

1. Compute the digests and single stream statistics for the data streams. The N_s streams are equally divided into k groups. The server $i(i = 1, ..., k)$ will read those streams in the i-th group and compute their digests, single stream statistics and hash values.

2. Report highly correlated stream pairs based on the grid structure. The grid structure is also partitioned evenly into k parts. Each server will read in its part plus those cells adjacent to that part. Note that only the first n normalize DFT coefficients need to be communicated between servers, the overhead for communication is greatly reduced. Each server will report those stream pairs that are highly correlated in its part.

4 StatStream System

StatStream runs in a high performance interpreted environment called K[1]. The system follows the algorithmic ideas above and makes use of the following parameters:

- **Correlation Threshold** Only stream pairs whose absolute value of correlation coefficients larger than a specified threshold will be reported, because the others are not of interest. The higher this threshold, the fewer streams whose exact correlations must be computed.

- **Sliding Window Size** This is the time interval over which statistics are reported to the user. If the sliding window size is 1 hour, then the reported correlations are those over the past hour.

- **Duration over Threshold** Some users might be interested in only those pairs with correlation coefficients above the threshold for a pre-defined period. For example, a trader might ask "Has the one hour correlation between two stocks been over 0.95 during the last 2 minutes?" This parameter provides such users with an option to specify a minimum duration. A longer duration period of highly correlated streams indicates a stronger relationship between the streams while a shorter one might indicate an accidental correlation. For example, a longer duration might give a stock market trader more confidence when taking advantage of such potential opportunities. A longer duration also gives better performance, because we can update the correlations less frequently.

5 Empirical Study

Our empirical studies attempt to answer the following questions.

- How great are the time savings when using the DFT approximate algorithms as opposed to exact algorithms? How many streams they can handle in real time?

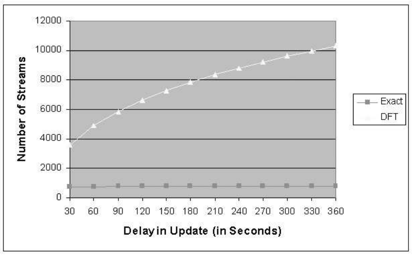

Figure 2: Comparison of the number of streams that the DFT and Exact method can handle

- What's the approximation error when using DFT within each basic window to estimate correlation? How does it change according to the basic and sliding window sizes?
- When we use the approximate algorithm to report high correlated pairs on real data set, what is its precision?

5.1 Synthetic Data Experiments

The time series streams are generated using the random walk model. For stream s,

$$s_i = 100 + \sum_{j=1}^{i}(u_j - 0.5), \qquad i = 1, 2, ...$$

where u_j is a set of uniform random numbers. Our experiments were performed on a 1.5GHz Pentium 4 PC with 128 MB of main memory.

Suppose that the synchronized streams have new data every second. The user of a time series stream system asks himself the following questions:

1. How many streams can I track at once in an online fashion? (Online means that even if the data come in forever, I can compute the statistics of the data with a fixed delay from their occurrence.)
2. How long is the delay between a change in correlation and the time when I see the change?

Our system will compute correlations of all pairs at the end of each basic window. As noted above, the computation for basic window i must finish by the end of basic window $i+1$ in order for our system to be considered on-line. Otherwise, it would lag farther and farther behind over time. Therefore, some of the correlations may be computed towards the end of the basic window $i+1$. So, the user perceives the size of the basic window as the maximum delay between the time that a change in correlation takes place and the time it is computed.

The net result is that the answers to questions (1) and (2) are related. We can increase the number of streams at the cost of increasing the delay in reporting correlation.

Figure 2 shows the number of streams vs. the minimum size of the basic window for a uniprocessor and for different versions of the algorithm. In the DFT method, we choose the number of coefficients to be 6.

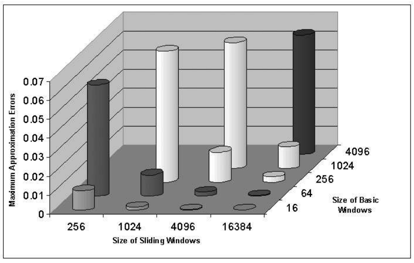

Figure 3: Approximation error with different basic/sliding window sizes

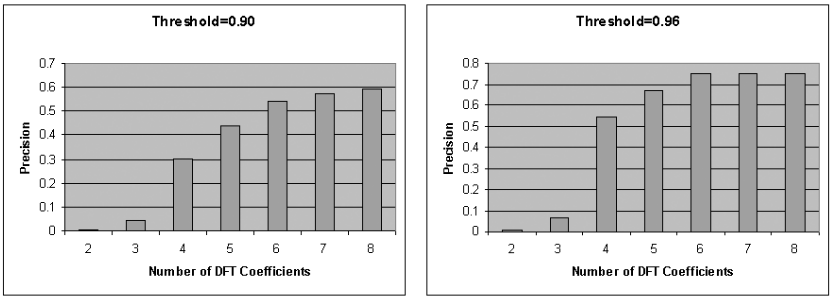

Figure 4: The precision using different numbers of coefficients

The number of streams handled by our system increases with the size of the basic window, while there is no perceivable change for the exact algorithm.

Precision Measurement

When compute the correlations of all pairs based on DFT curve fitting in each basic window, the size of the basic window can be adjusted to trade speed for accuracy. Our experiment shows that the error of correlation coefficients using this approach is very small, especially for sliding windows of large size.

Figure 3 shows the maximum approximate errors for correlation coefficients of pairs out of 100 streams when we use the first 2 DFT coefficients in each basic windows. It compares the precision using different sizes of sliding and basic windows. Smaller basic windows are more precise as we would expect. The experiment also indicates that for a larger sliding window, a larger basic window can still give very good approximation.

5.2 Stock Exchange Data Experiments

The New York Stock Exchange (NYSE) Trade and Quote (TAQ) database provides intraday trade and quote data for all stocks listed on NYSE, AMEX, NASDAQ, and SmallCap issues. The database grows at the rate of 10GB per month. The historical data since 1993 have accumulated to 500GB.

The data we use in our experiment are the tick data of the stocks indexed by SP500 during the first two weeks of 2001. During this period there are 452 streams that have tick data each work day. Every day each stream holds about 400 timepoints, in this case minutes, corresponding to a six and a half hour trading day. We choose a sliding window of two weeks and a basic window of a day.

Table 1: Comparison of StatStream with the Most Similar Methods

Properties	Agrawal et al.	Datar et al.	Gehrke et al.	StatStream
Online Algorithm	No	Yes	Yes	Yes
Guaranteed response time	Not relevant	Amortized	Yes	Yes
Search data structure	Yes	Not relevant	Not relevant	Yes
Auto-correlation	No	No	No	Yes

Figure 4 shows the precision using different numbers of coefficients. The correlation coefficient thresholds in the figure are 0.9 and 0.96 respectively. The precision is the ratio of the number of pairs whose correlations are above the threshold and reported by StatStream, to the number of pairs that are reported by StatStream(including some false positives). The experiment shows that with 6 DFT coefficients for the series in sliding windows, we can achieve high precision. Also the system guarantees no false negatives. The experiment shows that our choice of the number of Fourier coefficients is consistent with the observation made in [2].

6 Related Work

There is increasing interest in data streams. In the theoretical literature, Datar et. al [6] study the problem of maintaining data stream statistics over sliding windows. They focus on single stream statistics and propose an online data structure, exponential histogram, that can be adapted to report statistics over siding windows at every timepoint. They achieve this with a limited memory and a tradeoff of accuracy. Our online basic window synopsis structure can report the precise single stream statistics with a delay of the size of basic window, but our multi-stream statistics also trade accuracy against memory and time.

Gehrke et al. [10] also study the problem of monitoring statistics over multiple data streams. The statistics they are interested in are correlated aggregates when the number of streams to be monitored is small. A typical query in phone call record streams is the percentage of international phone calls that are longer than the average duration of a domestic phone call. They use histograms as summary data structures for the approximate computing of correlated aggregates. For some applications, such as finance, correlation coefficients are used more often as multiple streams aggregates, because they are robust to shifts and scaling. Still, in spirit if not in technique, our work is similar to theirs.

Recently, the data mining community has turned its attention to data streams. A domain-specific language, Hancock[5], has been designed at AT&T to extract signatures from massive transaction streams. Algorithms for constructing decision trees[7], association rule mining[14] and clustering [13] for data streams have been proposed. Recent work of Manku et al.[18, 19], Greenwald et al.[12] have focused on the problem of approximate quantile computation for individual data streams. Ganti et al. [9]introduce data span dimensions, including landmark windows and sliding windows, to clustering and association rule mining. Our work is complementary to the data mining research because our fast techniques for finding correlation coefficients can be used to speed up the clustering algorithms.

The work by Yi et al.[25] for the online data mining of co-evolving time sequences is very relevant to our work. Their method, MUSCLES, uses multivariate linear regression to interpolate missing or delayed values. They also extend their method to correlation detection. [1] Both MUSCLES and our method are incremental algorithms. Our approximate algorithm is faster than MUSCLES' exact algorithm, resulting in a shorter delay. Also, our method is able to compute auto-correlation efficiently.

Time series problems in the database community have been focused on discovering the similarity between an online sequence and an indexed database of previously obtained sequence information. Traditionally, the Euclidean similarity measure is used. The original work by Agrawal et al. [2] utilizes the DFT to transform data from the time domain into frequency domain and uses multidimensional index structure to index the first few DFT coefficients. In their work, the focus is on the whole sequence matching. This was generalized to allow subsequence matching [8]. Rafiei and Mendelzon[21] improve this technique by allowing transformations, including shifting, scaling and moving average, on the time series before similarity queries. The distances between sequences are measured by the Euclidean distance plus the costs associated with these transformations. By using the correlation measure in our work, we can discover counter-trends, which correspond to negative correlations. In the work described in this paragraph, the time series are all finite data sets. The maintenance of digests for basic windows also permits the computation of other statistics, such as autocorrelation. But the main difference between our work and the previous work is that our comparisons are online streams vs. online streams as opposed to queried sequences vs. database. Table 1 compares our work with some of the previous work.

Other techniques such as Discrete Wavelet Transform (DWT)[4, 23], Singular Value Decomposition (SVD)[17] and Piecewise Constant Approximation (PCA)[24, 16] are also proposed for similarity search. Keogh et al. [16] compares these techniques for time series similarity queries. With the exception of SVD, these techniques are based on curve fitting. Hence they are alternative ways of computing digests and could be used in our sliding window/basic window framework.

7 Conclusion

Maintaining multi-stream and time-delayed statistics in a continuous online fashion is a significant challenge in data management. Our paper solves this problem in a scalable and efficient way that gives a guaranteed response time with high accuracy.

The Discrete Fourier Transform technique reduces the enormous raw data streams into a manageable synoptic data structure and gives good I/O performance. For any pair of streams, the pair-wise statistic is computed in an incremental fashion and requires constant time per update using a DFT approximation. Experiments conducted using synthetic and real data show this technique to be very efficient and precise for long time series. On a PC costing less than $1,000, we can detect high pairwise correlations among 10,000 streams that deliver data every second with a delay of less than five minutes. We can do this in a continuous way, because our algorithm uses a fixed amount of memory.

References

[1] http://www.kx.com.

[1] Our method for computing correlation coefficients can be applied to the interpolation of values missing from one data stream. We exploit the fact that high-valued correlation coefficients imply high-valued regression coefficients. We can choose streams that are highly correlated to the delayed stream and perform a linear regression-based interpolation of the missing values.

[2] R. Agrawal, C. Faloutsos, and A. N. Swami. Efficient Similarity Search In Sequence Databases. In D. Lomet, editor, *Proceedings of the 4th International Conference of Foundations of Data Organization and Algorithms (FODO)*, pages 69–84, Chicago, Illinois, 1993. Springer Verlag.

[3] S. Babu and J. Widom. Continuous queries over data streams. *SIGMOD Record*, 30(3):109–120, 2001.

[4] K.-P. Chan and A. W.-C. Fu. Efficient time series matching by wavelets. In *Proceedings of the 15th International Conference on Data Engineering, Sydney, Australia*, pages 126–133, 1999.

[5] C. Cortes, K. Fisher, D. Pregibon, and A. Rogers. Hancock: a language for extracting signatures from data streams. In *ACM SIGKDD Intl. Conf. on Knowledge Discoveryand Data Mining*, pages 9–17, 2000.

[6] M. Datar, A. Gionis, P. Indyk, , and R. Motwani. Maintaining stream statistics over sliding windows. In *SODA*, page to appear, 2002.

[7] P. Domingos and G. Hulten. Mining high-speed data streams. In *Proc. ACM SIGMOD International Conf. on Management of Data*, pages 71 – 80, 2000.

[8] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In *Proc. ACM SIGMOD International Conf. on Management of Data*, pages 419–429, 1994.

[9] V. Ganti, J. Gehrke, and R. Ramakrishnan. Demon: Data evolution and monitoring. In *Proceedings of the 16th International Conference on Data Engineering, San Diego, California*, 2000.

[10] J. Gehrke, F. Korn, and D. Srivastava. On computing correlated aggregates over continual data streams. In *Proc. ACM SIGMOD International Conf. on Management of Data*, 2001.

[11] D. Q. Goldin and P. C. Kanellakis. On similarity queries for time-series data: Constraint specification and implementation. In *Proceedings of the 1st International Conference on Principles and Practice of Constraint Programming (CP'95)*. Springer Verlag, 1995.

[12] M. Greenwald and S. Khanna. Space-efficient online computation of quantile summaries. In *Proc. ACM SIGMOD International Conf. on Management of Data*, pages 58–66, 2001.

[13] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams. In *the Annual Symposium on Foundations of Computer Science,IEEE*, 2000.

[14] C. Hidber. Online association rule mining. In *Proc. ACM SIGMOD International Conf. on Management of Data*, pages 145–156, 1999.

[15] E. Keogh and P. Smyth. A probabilistic approach to fast pattern matching in time series databases. In *the third conference on Knowledge Discovery in Databases and Data Mining*, 1997.

[16] E. J. Keogh, K. Chakrabarti, S. Mehrotra, and M. J. Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. In *Proc. ACM SIGMOD International Conf. on Management of Data*, 2001.

[17] F. Korn, H. V. Jagadish, and C. Faloutsos. Efficiently supporting ad hoc queries in large datasets of time sequences. In J. Peckham, editor, *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA*, pages 289–300. ACM Press, 1997.

[18] G. Manku, S. Rajagopalan, and B. Lindsley. Approximate medians and other quantiles in one pass and with limited memory. In *Proc. ACM SIGMOD International Conf. on Management of Data*, pages 426–435, 1998.

[19] G. S. Manku, S. Rajagopalan, , and B. G. Lindsay. Random sampling techniques for space efficientonline computation of order statistics of large datasets. In *Proc. ACM SIGMOD International Conf. on Management of Data*, pages 251–262, 1999.

[20] L. Molesky and M. Caruso. Managing financial time series data: Object-relational and object database systems. In *VLDB'98, Proceedings of 24rd International Conference on Very Large Data Bases, August 24-27, 1998, New York City, New York, USA*. Morgan Kaufmann, 1998.

[21] D. Rafiei and A. Mendelzon. Similarity-based queries for time series data. In *Proc. ACM SIGMOD International Conf. on Management of Data*, pages 13–25, 1997.

[22] D. Rafiei and A. Mendelzon. Efficient retrieval of similar time sequences using dft. In *Proc. FODO Conference, Kobe, Japan*, 1998.

[23] Y.-L. Wu, D. Agrawal, and A. E. Abbadi. A comparison of dft and dwt based similarity search in time-series databases. In *Proceedings of the 9 th International Conference on Information and Knowledge Management*, 2000.

[24] B.-K. Yi and C. Faloutsos. Fast time sequence indexing for arbitrary lp norms. In *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt*, pages 385–394. Morgan Kaufmann, 2000.

[25] B.-K. Yi, N. Sidiropoulos, T. Johnson, H. V. Jagadish, C. Faloutsos, and A. Biliris. Online data mining for co-evolving time sequences. In *Proceedings of the 16th International Conference on Data Engineering, San Diego, California*, pages 13–22, 2000.