# StatStream: a Monitor for Time Series Data Stream Statistics

August 27, 2001

**Abstract**

In many applications, new data is being generated every day, every hour and even every second. For example, sensors generate petabyte streams daily; the ATT long distance data stream consists of approximately 300M records per day from 100M customers. [6]; for each stock listed in New York Stock Exchange (NYSE), every second there are several ticks, trading and prices changing. These applications require a new data management model, Data Streams, which is attracting growing attention in academic institution and industry application.

## 1 Introduction

In many applications, new data is being generated every day, every hour and even every second. 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[17]. The ATT long distance data stream consists of approximately 300M records per day from 100M customers. [6]; for each stock listed in New York Stock Exchange (NYSE), every second there are several ticks, trading and prices changing. These applications require a new data management model, Data Streams, which is attracting growing attention in academic institution and industry application. Data streams provide information that is ordered by time. Unfortunately it is difficult to process these data in set-oriented data management systems. Instead we simply preserve such data streams in often-unusable archives. Turning those data into valuable information and knowledge will have profound effects on telecommunications, network monitoring, environmental monitoring and financial analysis. Due to the special features of data streams, a Data Stream Management System (DSMS) should ideally have the following character as contrast to traditional database management systems (DBMS).

- The data is changing constantly, exclusively through insertions of new elements, instead of some preserved data

- Queries on the data streams are continuously evaluated and the stream may never end. Instead of trying to store full streams on disk and use them later, a standing query or monitor on the data streams are more appropriate.
- The recent data in the stream has far more importance than older data. Most of the time we would employ a sliding window model that only process information available more recently, rather that a unrestricted window.
- It is either unnecessary or impractical to operate on large portions of the data multiple times, ideally, a practical algorithms on data streams should require only one pass on the data.
- A Data Stream Management System should be able to operate in bounded memory space. To trade accuracy for efficiency, approximations can be used. Approximations may be based on sampling, histograms, Fourier or Wavelet transform.

In this paper, we will discuss a system that monitor the statistics for high-speed time series Data Stream. The analysis of times series is based on the computation of the statistics. If the time series come in the form of high speed data streams, it is crucial to compute the statistics in constant time and bounded memory. For example in financial trading, all the real time trading strategy is based on the online computation of the statistics of streaming market data acquired in real time. It is well know that due to the Efficient Market Hypnosis, it is impossible to mine long term trend in stock market. But short time trend still exists, and a successful trader should exploit this before the short time trend disappears or become a long term trend that is well known to everybody. This requires a fast algorithm to update the statistics of time series data stream. We propose a framework to compute the statistics in constant time. The statistics we considered in our framework include most of the statistics that a trader might be interested. It should be noted that although our algorithms are greatly motivated by the applications in finance, they also apply to many other application, such as sensor data processing, medical application.

# 2 Problem Definition

In this section, we introduce the problem of monitoring the statistics of time series data stream. We describe the model of time series data streams in 2.1. In 2.2, we enumerate the three categories of windows based on which the statistics of time series are computed. The statistics we will monitor is listed in 2.3

## 2.1 time series data streams

Consider time series of real number in a data streams. Each record in the streams is a tuple (streamID, timepoint, value). Many time series are in the same data stream and different time series are distinguished by different streamed. The substreams come in time order, which we abstract as timepoints. The timepoints start from one and increase by one each time a new value of the substream is available. The timepoints of different substreams are non-decreasing too. Let $s_i$ or $s[i]$ denotes the value of substream $s$ with timepoint $i$. $s[i:j]$ denotes the subsequence of time series $s$ in timepoint $i$ through $j$. $s^i$ denote a substream with streamID $i$. Also we use $t$ to denote the latest timepoint, i.e., now. The statistics we will monitor can be denote as $stat(s_j^{i_1}, s_j^{i_2}, ..., s_j^{i_k}, j \in windows)$. We will discuss the meaning of window in the next section.

Note that we do not assume that the time series in the stream have value in all the timepoints. It is possible that some substream misses some value in certain timepoints. This is more a more realistic model. Our algorithm will have some special treatment for missing data. The only assumption we have is that the timepoints are non-decreasing, which is implied by the definition of "stream".

## 2.2 temporal span

Generally speaking, the statistics of time series can be based three categories of temporal span.

1. Landmark windows In this temporal span, statistics are compute based on the value with timepoint no less that than a landmark timepoint. Stat(s, landmark(k)) will be computed in the subsequence of time series $s[i], i \geq k$. A special case, unrestricted window, is when $k = 1$, the statistics are based on all the available data.

2. Sliding windows In most applications, a sliding windows model is more appropriate for data stream. Given the length of the sliding window l and the current timepoint now, $stat(s, sliding(l))$ will be computed in the subsequence $s[t - l + 1; t]$.

3. Damping windows The temporal span of sliding window use only the more recent data and discard the old data. In some application, although the old data are not as important as the new data, they still have some impact on our decision-making and one would like the old data to be expired more smoothly. For example, in the computation of average, a sliding window model will compute the average as

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

While in a damping window model the weights of data decrease exponentially as the timepoint traced back to history. An average in damping window can be computed as follow

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

Other statistics in a damping window can be defined similarly. Notice that another advantage of using damping window over sliding window is that we don't need to expire time series moving out of sliding window expressively. Deleting a value from the sliding window need to access the expiring data and perform computation to undo its effect in the statistics. For example, In average, we need to minus the old data from the average.

## 2.3 statistics

Let the time series in consideration be $s_i, i = 1, ..., k$. The statistics we will monitor are

1. Average

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

2. Standard deviation

$$std(s) = \frac{k \sum_{i=1}^{k} x_i^2 - (\sum_{i=1}^{k} x_i)^2}{k-1}$$

3. Maximum and Minimum

$$max(s) = \max_{i=1,\ldots,k}(s_i), min(s) = \min_{i=1,\ldots,k}(s_i)$$

4. Best Fit Slope: the slope of the line that best fit the time series.

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

$$\bar{s} = avg(s_i)$$

5. Correlation coefficients

$$corr(r, s) = \frac{\frac{1}{k}\sum_{i=1}^{k} r_i s_i - \overline{rs}}{std(r) * std(s)}$$

6. Autocorrelation: the correlation of the series with itself when there is lag between.

7. Beta: the sensitivity of a substream's values to the values of another substream $r$.

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

# 3 statistics over sling window

To compute the statistics over a sliding window, we will maintain the synopsis data structure for time series in the sliding window that is sufficient to compute the statistics. Our framework further subdivides the sliding windows to shorter windows, basic windows to facilitate efficient expiring of old data and incorporating of new data.

Let the data within sliding window be $s[t-l+1:t]$. Suppose $l = kb$, where $b$ is the length of the sliding window and $k$ is the number basic windows within the sliding window. Let $S[0], S[1], ..., S[k-1]$ denote the basic windows, where $S[i] = s[t-l+i*b+1 : t-l+(i+1)*b]$.

## 3.1 simple statistics

We will start with the computation of moving average and standard deviation. For the definitions of average and standard deviation, clearly it is enough to maintain the sum of the time series and the sum of the square of the time series in order to compute the average and standard deviation. We will use this case to illustrate our framework for incrementally maintaining digests. In the case of average, the digest for time series over the sliding window is the $\sum(s[t-l+1:t])$. For each basic window $S[i]$, we also maintain $\sum(S[i])$. After $b$ new data from the stream are available, we compute the sum over the new basic window $S[n]$. The sum over the basic window is updated as follows.

$$\sum_{new}(s) = \sum_{old}(s) + \sum S[n] - \sum S[n+1]$$

. As we use rotating basic windows within the sliding window, $S[n]$ is the new basic window and $S[i+1]$ is therefore the expiring basic window. The computation of standard deviation, minimum and maximum are similar.

For the computation of best fit slope, notice that it is sufficient to compute $bfs0(s) = \sum_{i=t-l+1}^{t} s_i * i$. We maintain digest for basic windows as follows,

$$bfs1(S[i]) = \sum_{j=1}^{b} s[t-l+i*b+j]i$$

. The incrementally maintain of $bfs0(s)$ can be achieved as the following equation holds.

$$bfs0_{new}(s) = bfs0_{old}(s) + l * \sum S[n] + b \sum s + bfs1(S[n]) - bfs1(S[n+1])$$

## 3.2 correlation

The efficient computing of correlation is more complicated than the statistics we discussed so far. But the same framework still applies to the computation of correlation. It should be noted that to maintain the correlation of pair of stream, it is enough to maintain the cross product of the pair. But as the number of substreams could be very huge and the number of pairs will be the square of the number of substreams. And for autocorrelation, we might need to compute for different lags. To maintain the cross product, for each pair we need to compute it for each timepoint. This can be very time consuming. Fortunately, in almost all the application, people are interested only in pairs that are highly correlated. If we have a way to compute the correlation coefficients approximately, all the pairs can be filtered into a much less number of pairs. We can then examine these highly correlated pairs closely. To compute the correlation coefficients approximately, we need to employ some data reduction techniques. Different from the simple statistics we consider in the last section, the computation of correlation coefficients require information of the shape of the time series, instead of just some accumulate information. Our general approach is to use curve fitting in each basic window to reduce the size of data. First we will consider compute the cross product when the two series are aligned in basic windows. Then we will extend our approach to a general case when the basic windows are not aligned.

## 3.3 cross product with windows aligned

We will compute the cross product of two series $S^x$ and $S^y$. The basic windows of the two series are aligned. That is

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

. It is enough to compute the cross product in a basic window, $x_1, x_2, ...x_b$ and $y_1, y_2, ...y_b$.

Let $f_1(x), f_2(x), ...$ be a family of continuous function. We approximate time series in the basic window $S[i] = s[t-l+i*b+1 : t-l+(i+1)*b] = x_1, x_2, ...x_b$ with the family of functions.

$$x_j = \sum_{m=1}^{n} c_m^x f_m(j) \qquad j = 1, ..., b$$

$c_m^x, m = 1, ..., n$ are $m$ coefficients to approximate the time series with the function family $f$. The cross product is therefore

$$\sum_{j=1}^{b} x_j y_j = \sum_{i=1}^{b} (\sum_{m=1}^{n} c_m^x f_m(j) \sum_{p=1}^{n} c_p^y f_p(j))$$

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

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

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

$$W(m,p) = 0 \qquad m \neq p$$

$$W(m,m) = V(m) \neq 0$$

We have

$$\sum_{j=1}^{b} x_j y_j = \sum_{m=1}^{n} c_m^x c_m^y V(m)$$

Using this curve fitting technique, we reduce the space requirement from $b$ to $n$ and the time requirement for computing cross product from $b$ to $n$ too. Now the questions become how can we choose the function family and how efficient can we compute the coefficients for curve fitting. One of the function families we can choose is triangular function family. By performing a Discrete Fourier Transform in the times series in the basic window, we can compute the coefficients in constant time.

$$\omega = \frac{2\pi}{b}$$

$$x_j \approx c_0 + \sum_{p=1}^{n} (c_p \cos(pj\omega) + c_{n+p} \sin(pj\omega)) \qquad j = 1, ..., b$$

$$\begin{cases} c_0 = \overline{x} & \\ c_p = \frac{2}{b} \sum x_j \cos(pj\omega) & p = 1, ..., n \\ c_{n+p} = \frac{2}{b} \sum x_j \sin(pj\omega) & \end{cases}$$

Noticed that $\cos(pj\omega), \sin(pj\omega)$ in the formula above can be computed and stored in memory before the data streams come in.

### 3.4 cross product without windows aligned

It is not always the case that we can have two series with basic windows aligned. Some correlations are computed between two series with lags, and the autocorrelation by definitions is on series with lags. Suppose we have two series $s^x$ and $s^y$ and the basic windows of the two series are not aligned. Given $s^x = s_1^x, ..., s_l^x = S^x[1], ..., S^[k]$ and $s^y = s_u^y, ..., s_{u+l-1}^y = s_u^y, ..., s_v^y, S^y[1], ..., S^y[k-1]$,

$$crossproduct(s^x, s^y) = \sum_{i=1}^{l} (s_i^x s_{u+i-1}^y)$$

. It is sufficient to compute the cross product of a partial basic window. That is, the cross product of the last $a$ values of one basic window $S^1 = x_1, ..., x_b$ with the first $a$ values of another basic window $S^2 = y_1, ..., y_b$.

$$crossproduct(S^1, S^2, a) = \sum_{j=1}^{a} x_j y_{b-a+j}$$

This can be computed using only the DFT coefficients of basic windows $S^1$ and $S^2$.

$$\begin{aligned}\sum_{j=1}^{a} x_j y_{b-a+j} &= \sum_{i=1}^{a}(\sum_{m=1}^{n} c_m^x f_m(j) \sum_{p=1}^{n} c_p^y f_p(j)) \\ &= \sum_{m=1}^{n}\sum_{p=1}^{n} c_m^x c_p^y (\sum_{j=1}^{a} f_m(j) f_p(j)) \\ &= \sum_{m=1}^{n}\sum_{p=1}^{n} c_m^x c_p^y W(m,p,a)\end{aligned}$$

If we precompute the table of $W(m,p,a); m,p = 1...,n; a = 1,...,b/2$, the cross product can be computed with the DFT coefficients only in time $O(n^2)$ without requiring the aligning of basic windows.

# 4 StatStream System

StatStream use our innovative algorithm for data streams to achieve high performance. StatStream maintains these digests incrementally and thus guarantees constant time computation of statistics for data streams.

## 4.1 parameters for StatStream

- Frequency to Update Statistics f This parameter tells the program to compute and output the statistic every f timepoints. For example, if f=5, then the program will output the result every 5 timepoints of a substream. For efficiency's sake, f should divide l.
- Correlation Threshold Only substream pairs whose absolute value of correlation coefficients larger than a specified threshold will be reported. The default value is 0.9.
- Duration over Threshold Some users might be interested in only those pairs with correlation coefficients above the threshold for a pre-defined period. This parameter provides such users with an option to specify a minimum duration. The default value is 1.

## 4.2 Categories Comparing Method

Some users might not be interested in the comparison of all substream pairs. They can then divide these substreams into different categories and the computation of the correlation will be based upon these categories. There are two categories options: inter-categories and intra-categories. For the inter-categories options, the two streams in a pair will be compared only if they are from different categories. For the intra-categories options, the two streams in a pair will be compared only if they are from the same category. For example if we have 5 substreams from 2 categories. C1:(s1,s2,s3),C2:(s4,s5)

- With the inter-categories option, the following pairs will be compared: (s1,s4),(s1,s5),(s2,s4),(s2,s5),(s3,s 4),(s3,s5)
- With the intra-Categories option, the following pairs will be compared: (s1,s2),(s1,s3),(s2,s3),(s4,s5)

Without the category table, we will compute the correlation of all pairs with different streamIDs.

# 5 Empirical study

## 5.1 Synthetic data

## 5.2 NYSE TAQ data

The New York Stock Exchange (NYSE) Trade and Quote (TAQ) database provides intraday trades and quotes data for all stocks listed on NYSE and AMEX, and NASDAQ, and SmallCap issues. The database grows in the speed of 10GB per month. The historical data back to 1993 is 500GB.

# 6 Related work

There is increasing interest in data streams. In theoretical foundations, Henzinger et al.[14] studied the space complexity of algorithms that make only one pass over the data streams. Algorithms and complexity for computing approximate frequency moments [2], L1-difference[8],spot-checking[9] for data streams were also examined. Babu et al.[3] proposed a frame work for evaluating continuous queries over data streams. Known systems for continuous query include Tapestry[18],NiagaraCQ[5]. Correlated aggregates over data streams was studied by Gehrke et. al [16].

Recently, many researches have been focus on data streams in the data mining community. Hancock[6] was designed as a domain-specific language to extract signatures from massive transaction streams. The algorithms for constructing decision trees[7], association rule mining[15] and clustering [13] for data streams were proposed. Recent work of Manku et al.[1, 10], Greenwald et al[12] were focused on the problem of approximate quantiles computation for data streams. Ganti et al. introduced data span dimension, unrestricted window and most recent window, to the mining and monitoring for evolving data. The work by Yi et al[19] in the online data mining for co-evolving time sequences is very relevant to our work. Their method, MULSCLES, use multi-variate linear regression to discover correlations and to forecast missing or delayed values.

Also, the data reduction techniques, for example, the New Jersey data reduction [4] are very relevant to our works. Gibbons et al. [11] considered the use of synopsis data structures to provide fast answer to queries with little space. Samples [],histograms [] and wavelet [] were used as synopsis data structures.

# 7 Future work

# 8 Conclusion

# References


[1] In *SIGMOD Conference*, 2000.

[2] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. In *ACM Symposium on Theory of Computing*, pages 20–29, 1996.

[3] S. Babu and J. Widom. Continuous queries over data streams, 2001.

[4] D. Barbara, W. DuMouchel, C. Faloutsos, P. J. Haas, J. M. Hellerstein, Y. E. Ioannidis, H. V. Jagadish, T. Johnson, R. T. Ng, V. Poosala, K. A. Ross, and K. C. Sevcik. The new jersey data reduction report. *Data Engineering Bulletin*, 20(4):3–45, 1997.

[5] J. Chen, D. J. DeWitt, F. Tian, and Y. Wang. NiagaraCQ: a scalable continuous query system for Internet databases. In *SIGMOD Conference*, pages 379–390, 2000.

[6] D. P. Corinna Cortes, Kathleen Fisher 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.

[7] P. Domingos and G. Hulten. Mining high-speed data streams. In *SIGMOD Conference*, pages 71 – 80, 2000.

[8] J. Feigenbaum, S. Kannan, M. Strauss, and M. Viswanathan. An approximate l 1 -difference algorithm for massive data streams. In *IEEE Symposium on Foundations of Computer Science*, pages 501–511, 1999.

[9] J. Feigenbaum, S. Kannan, M. Strauss, and M. Viswanathan. Testing and spot-checking of data streams (extended abstract). In *Symposium on Discrete Algorithms*, pages 165–174, 2000.

[10] S. R. G. S. Manku and B. G. Lindsay. Random sampling techniques for space efficientonline computation of order statistics of large datasets. In *SIGMOD Conference*, pages 251–262, 1999.

[11] P. B. Gibbons and Y. Matias. Synopsis data structures for massive data sets. In *DI- MACS Series in Discrete Mathematics and Theoretical Computer Science*, volume 50, 1999.

[12] M. Greenwald and S. Khanna. Space-efficient online computation of quantile summaries. In *SIGMOD Conference*, 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] M. Henzinger, P. Raghavan, S. Rajagopalan, o data, and s series. special issue on computing on very large datasets. *Technical Note*, 011, 1998.

[15] C. Hidber. Online association rule mining. In *SIGMOD Conference*, pages 145–156, 1999.

[16] F. K. Johannes Gehrke and D. Srivastava. On computing correlated aggregates over continual data streams. In *SIGMOD Conference*.

[17] P. Keogh, E. 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.

[18] D. Terry, D. Goldberg, D. Nichols, and B. Oki. Continuous queries over append-only databases. In *SIGMOD Conference*, pages 321–330, 1992.

[19] B.-K. Yi, N. Sidiropoulos, T. Johnson, H. V. Jagadish, C. Faloutsos, and A. Biliris. Online data mining for co-evolving time sequences. In *ICDE*, pages 13–22, 2000.