Count-Min Sketch

An Implementation and Refinement

Algorithmic Techniques for Massive Data

(CS-5390)

|  |  |
| --- | --- |
| **Instructed By**  Dr. Subrahmanyam K | **Authors**  Joseph K J  cs17mtech01001@iith.ac.in  Gautam Muduganti  cs17resch01003@iith.ac.in |

***Abstract***

Count-Min sketch is a sublinear space data structure that can be used to summarize the frequency of items in a data stream. Its uses constant amount of space that does not vary with the size of the stream. It guarantees an -approximation of the actual frequency statistic of any element in the stream with very good confidence bound.

In this project work, we focus on the following

* A Java implementation for the Count-Min Sketch
* Integration with various data streams which include inhouse data streams and cloud based data streams from a data source called PubNub
* Conservative Count-Min which builds on top on the standard Count-Min sketch which limits the space used by the hash functions

Keywords: Count-Min Sketch, Sublinear Algorithm, Conservative Count-Min, PubNub, Java

1. **Introduction**

Count-Min Sketch uses an efficient way to keep track of the number of items that passed by in the stream, that scales well the number of items in the stream. The size of the data structure is solely dependent on accuracy guarantees of the frequency count. The accuracy guarantee is specified by two user specified values, and , meaning that the error in answering the frequency query is within a factor of with a probability .

* 1. **Data Structure**

The Count-Min Sketch is an array of counters of width and depth . The width and depth is computed using the and values provided by the user. Each entry in initialised to zero. The number of rows in the array correspond to the number of hash function that we use and the number of column in the number of buckets to which each item is hashed into. It is also possible that the user might specify the width and depth and the algorithm provides a guarantee on the and values based on that.

* 1. **Update Procedure**

When we see an element in a stream, is hashed with all the hash functions. The counter at each of the hash location is incremented by the count passed into the update procedure. For adding an element, updates are made in the data structure.

* 1. **Update Procedure**

When querying for an element , we hash with each of the hash functions. Then we take the minimum of all the values that are at the hashed location.

1. **Conservative Count-Min**

Count-Min Sketch with conservative update is a modification of the vanilla Count-Min Sketch. It helps to avoid unnecessarily updating counter values and hence reduce the overestimation error. The query procedure for Conservative Count-Min is the same as Count-Min. However, to update an item , with frequency , we first run the query for . This will give the frequency of in the current data structure. The count is updated only if the count the count at the index is greater than .

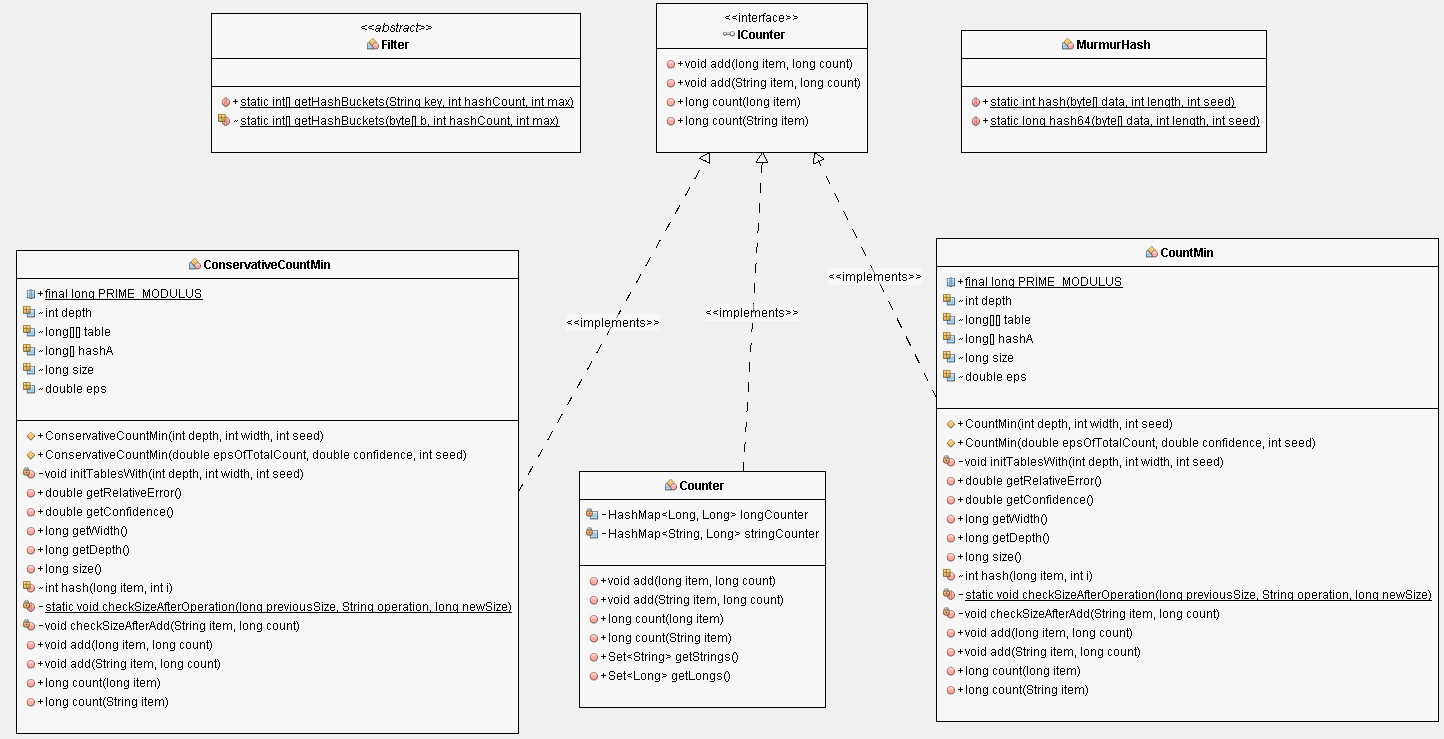
Count is update according to the formula:

The intuition is that since the point query returns the minimum of all the values, we update a counter only if it is necessary. This will reduce overestimation and limit the size of each of the buckets.

1. **Implementation**

We implemented a Count-Min Sketch which can support both numeric and string streams. The implementation is based on the paper *Approximating Data with the Count-Min Data Structure [2]*. The source code is available at - <https://github.com/cs17resch01003/pubnub_countmin>.

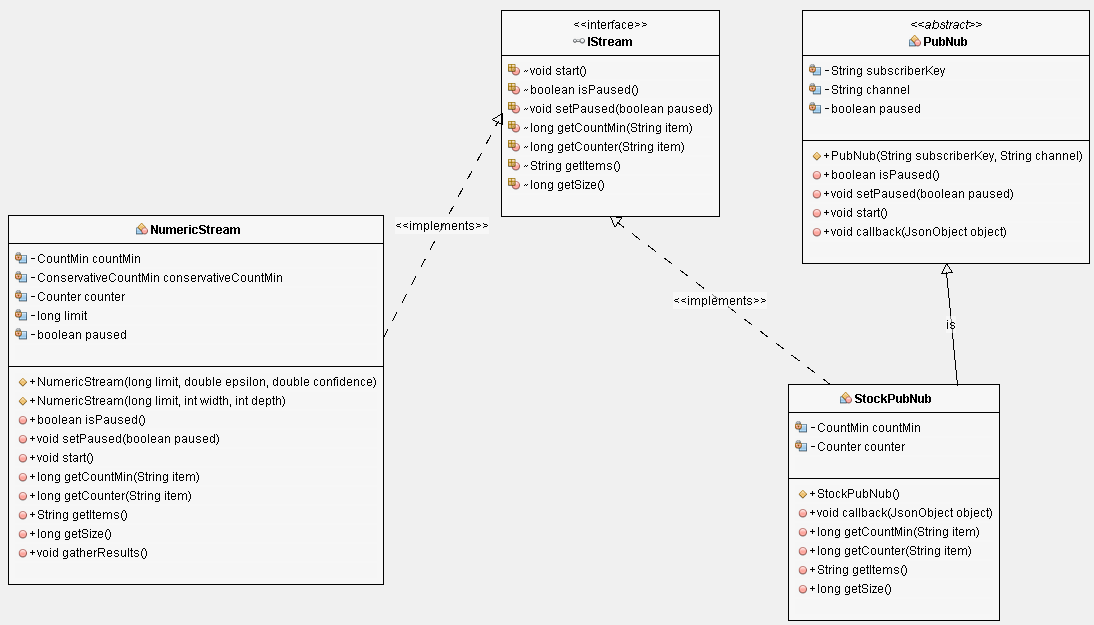
* 1. **Class Diagram**



The counters implement the ICounter interface and we have three different counters -

* Counter - The standard counter which does exact counting
* CountMin - The counter which implements the Count-Min Sketch
* ConservativeCountMin - The counter which implements the Conservative Count-Min Sketch

MurmurHash is a standard algorithm which is used to hash Strings.



All the streams implement the IStream interface and there is a standard abstract class PubNub which all implementations going to PubNub inherit from. We have two streams implemented as a part of this project -

* NumericStream - A stream which generates random numbers with a given limit
* StockPubNub - A publicly available stream which constantly sends the price movements of various commodities

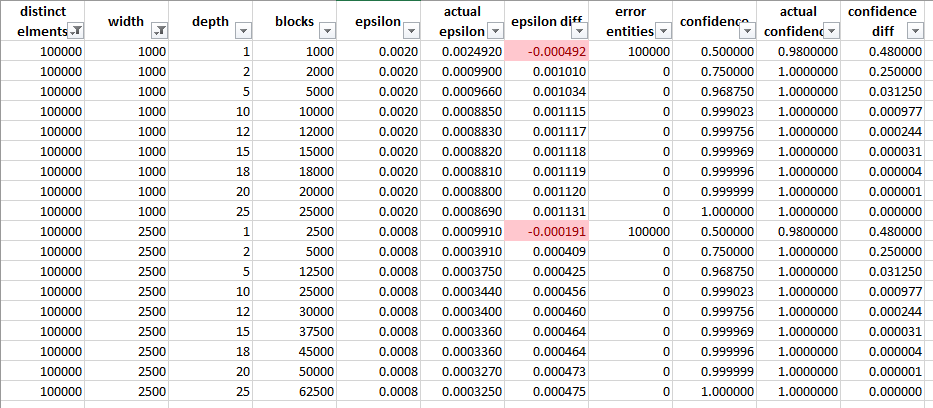
Each of these streams internally uses the three counters discussed above to count the frequency of elements.

1. **Experimental Evaluation**

As a part of the experimental evaluation, we run the algorithm with different widths and depths and calculated the mathematical value of and and the actual values of and . All experiment was done on the numeric stream with at-least 5 million items coming through the stream and the number of distinct elements which the stream was formed was varied through the experiments. Here are the values that were considered for the depth, width and distinct elements -

* Depths: 10, 20, 50, 100, 250, 500, 1000, 2500, 5000, 10000
* Widths: 1, 2, 5, 10, 12, 15, 18, 20, 25
* Distinct Elements: 100, 1000, 10000, 100000

Sample Results:



The complete set of results has 360 different experiments: [Experimental Results.xls](https://github.com/cs17resch01003/pubnub_countmin/blob/master/Experimental%20Results.xlsx)

As it can be seen from the experiment results, the actual values of both and are within the expected values thereby proving that the implementation works.

1. **Summary**

As a part of this project, we implemented the Count-Min and Conservative Count-Min sketches and verified their working through experimental evaluation. We can see that the values of and are significantly impacted with the number of hash functions and the width of each of them. We noticed that increasing the hash functions from 1 to 2 increases the confidence from 0.5 to 0.75 and from 2 to 3 brings it to about 0.968. For all practical purposes at-least 3 hash functions would be required for a proper working of the algorithm.

1. **References**

[1] Sketch Algorithms for Estimating Point Queries in NLP  
Graham Cormode, Amit Goyal, Hal Daume III  
Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, July 2012

[2] Approximating Data with the Count-Min Data Structure  
Graham Cormode and S. Muthukrishnan  
IEEE Software, published by the IEEE Computer Society