Data Sketches

Fast, Approximate Analysis of Big Data[1]

by Lee Rhodes, Yahoo, Inc.

Abstract

In the analysis of *big data*[2] there are often problem queries that don't scale because they require huge compute resources to generate exact results, or don't parallelize well. Examples include *count distinct*[3], quantiles, most frequent items, joins, matrix computations, and graph analysis. Algorithms that can produce "good enough" approximate answers for these problem queries are a required toolkit for modern analysis systems that need to process massive amounts of data quickly. For interactive queries there may not be other viable alternatives, and in the case of real-time streams, these specialized algorithms, appropriately called *streaming algorithms,* or *sketches*, are the only known solution. This methodology has helped Yahoo successfully reduce event processing times from days to hours or minutes on a number of its internal platforms. This article provides a short introduction to sketching and to DataSketches[4], an open source library of a core set of these algorithms designed for large analysis systems.

The Distinct Count Computational Challenges

Removing Duplicates. Suppose we have a small stream of visits to our new bookstore: *{Alice, Ben, Dorothy, Alice, Ben, Dorothy, Alice, Ben}.* The total count of visits is 8 and the distinct count of visitors is 3. In order to compute the distinct count exactly, the system must retain a reference to each distinct identifier it has seen so far in order to ignore the duplicates. This means that the system must reserve *O(n)* space, where *n* is the anticipated maximum number of distinct identifiers. This is straightforward if we know that the number of distinct identifiers is small.

Now extend the scale of this distinct counting challenge to streams that contain billions of identifiers with many duplicates. This is not an unrealistic scenario: Yahoo sees over a billion distinct users in a month[5]. It can be even larger as our input streams often include multiple identifiers that we want to count, such as cookies, login-IDs, device-IDs, session-IDs, etc. The space required is now $O(n_1 + n_2, \ldots)$, where n_i is the number of distinct identifiers of type *i.*

---

[1] Originally published: https://yahooeng.tumblr.com/post/135390948446/data-sketches, 17 Dec 2015

[2] The term "big data" is a popular term for truly massive data, and is somewhat ambiguous. For our usage here, it implies data (either in streams or stored) that is so massive that traditional analysis methods do not scale.

[3] *count distinct* is the formal term, borrowed from SQL that has an operator by that name, for the counting of just the distinct (or unique) items of a set ignoring all duplicates. For our usage here, it is reads more smoothly to just refer to distinct count or unique count.

[4] http://DataSketches.github.io

[5] Yahoo Q3 2015 earnings

Partitioning and Non-Additivity. The challenge becomes exacerbated when we have to partition the data by anything other than the identifier itself. Partitionings that are often dimensions of interest to the business include time, product, location or other parameters.

For example, suppose we had decided to partition the visits to our bookstore by product area and day:

Product Area	Monday	Tuesday
Fiction	Alice, Dorothy	Ben, Dorothy
Nonfiction	Alice, Ben	Alice, Ben

We have 4 partitions each with a distinct count of 2, but simple addition of these count values across any combination of more than one of these partitions will result in the wrong distinct count due to set overlap. The distinct count values are *non-additive*. This is a more sinister form of duplication because not only has the total storage requirement increased, but the duplicates across partitions cannot be removed! It is generally not possible to partition the data by business dimensions and guarantee that the identifier sets do not overlap.

This *non-additivity* property eliminates the possibility of answering queries across multiple partitions by only referring to the distinct count values for each partition. Any query across multiple partitions requires an entirely new distinct count operation that would have to read from the raw data or a copy of it. This *non-additive* property also has an impact on the system architecture in that we cannot create the nice aggregate hypercubes of a data mart and then query only the rows that qualify some predicate and sum up the results.

In other words, exact distinct count operations do not scale well. This non-additivity property of distinct counts is generally well understood by systems engineers. However, what is less well known is that now there are advanced algorithms that help us address the scalability challenge of distinct counting.

Sketching Algorithms

The name "sketch", with its allusion to an artist's sketch, has become the popular term to describe algorithms that return "good enough" approximate answers to queries, as these algorithms typically create small summary data structures that approximately resemble the much larger stream that it processed.

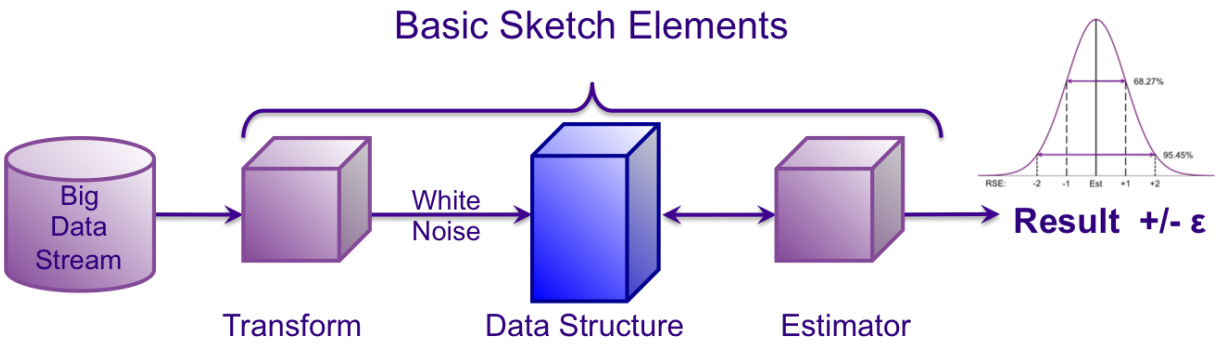
Sketching is a relatively recent development[6] and has evolved from a synergistic blend of theoretical mathematics, statistics and computer science. Sketching refers to a broad range of algorithms and has experienced a great deal of interest and growth since the mid-1990's coinciding with the growth of the Internet and the need to process and analyze massive data.

[6] Graham Cormode. Sketch Techniques for Approximate Query Processing, NOW publishers, 2011

There are several common characteristics of sketches:

- **Streamable**. Sketches are especially suitable for environments where huge amounts of data flow by as a stream of individual items. Each item of the stream, examined only once, must quickly update a small sketch summary data structure.
- **Approximate with predictable error**. Sketches achieve their amazing speed and predictable error by taking advantage of a fundamental assumption: *If we can accept a certain margin of error, it is often possible to develop algorithms that can compute the result substantially faster with fewer resources.* An important subset of sketches, called "stochastic streaming algorithms" intentionally introduce random variables into the algorithm.
- **Sublinear in size**. The required storage resources grow more slowly than the input data size (or don't grow at all) and can be orders-of-magnitude smaller (typically kilobytes to megabytes). Sketches allow the user to configure the size of the sketch as a trade-off with accuracy.
- **Mergeable, thus "additive".** To be useful in large data processing systems the sketch summary data structures should be "mergeable". That is, the merge of two sketches should produce the same result (within the specified error bounds) as if the two streams that produced the two sketches had been combined prior to being submitted to a single sketch. This enables arbitrary partitioning of the input data and fast "summing" of the intermediate sketches for fast query results.
- **Highly parallelizable**. Since summary data structures are mergeable, the computations using these summary data structures are highly parallelizable and suitable for use in large-scale compute environments such as Hadoop or Druid.

Distinct Count Sketch, High Level View



The first stage of a distinct count sketching process is a transformation that gives the input data stream the property of white noise, or equivalently, a uniform distribution of values. This is commonly achieved by hashing of the input distinct keys and then normalizing the result to be a uniform random value between zero and one.

The second stage of the sketch is a data structure that follows a set of rules[7] for retaining a small bounded set of the hash values it receives from the transform stage. This fixed upper bound on sketch size enables straightforward memory management.

The final element of the sketch process is a set of estimator algorithms that, upon a request, examines the sketch data structure and returns a result value. This result value will be approximate but will have well established and mathematically proven error distribution bounds.

As an example of accuracy, a sketch configured to retain 16,000 values will have a relative error of less than 1.6% with a confidence of 95%. The error distribution is approximately Gaussian or bell shaped (as shown in the figure), and is independent of the size of the input stream, which can be in the many billions of distinct items.

The DataSketch Library

DataSketches is a Java software library of streaming algorithms specifically designed for the analysis of massive data. The library includes multiple high performing sketching algorithms and numerous other supporting algorithms targeted to the practical application of these advanced algorithms in real systems.

- **Sketch Adaptors** are provided for Hadoop Pig, Hadoop Hive, and Druid. In both Hive and Druid, the adaptors are being integrated as built-in functions by the respective teams.
- **Maven deployable**. The library is designed to be deployed using Maven and the required jars are available from Maven Central. These jars can be integrated into any system that is JDK 7 or JDK 8 compatible. The sketches-core repository has no run-time dependencies, which makes it especially easy to integrate into virtually any Java-based system.
- **Robust, High Quality Implementations**. Extensive unit test code coverage, comprehensive javadocs and code documentation, thorough accuracy and performance characterization, and time-tested usage in major back-end analysis systems inside Yahoo combine to make this library very robust and suitable for production applications.
- **Benchmarking.** Code used for characterizing components of the library for speed, accuracy and performance is included in the test package hierarchy. This provides transparency as to how the many data plots on the web site were constructed.

[7] See KMV Tutorial for a walk-through of example rules

- **Counting Distinct Identifiers**. Currently the library includes two different update sketch families as well as union, intersection and difference operations for set expressions. These sketches are all part of the *Theta Sketch Framework*, described in our ICDT 2016 paper mentioned below. Most of these sketches can also be configured to operate either on the java heap or off-heap. In addition, there are two versions of the famous Hyper-Log Log (HLL) sketch, which is ideal for environments where only distinct counting and merging are required and space is extremely tight.

- **Quantiles.** Given a large stream of numeric values, such as web-page load times or interactive query response times, we often desire to characterize the distribution of these values to understand the impact on users, for example, the median, 5th, and 95th percentile values. These are called quantiles. The quantile sketch provides approximate answers to these queries with a well understood error bound that is independent of the distribution of the input values. This will become available soon.

- **Frequent Items.** Given a large stream of identifiers with many duplicates, we would like to know which identifiers occurred most frequently. This will become available soon.

- **MurmurHash3**. In addition to the above sketches is a fast, extended version of Austin Appleby's MurmurHash3[8] hash algorithm that can also accessed with a Pig UDF.

- **Memory Package**. A flexible, general purpose Memory Package for managing native memory data structures from within Java.

- **Science**. The core science behind this library is documented in *A Framework for Estimating Stream Expression Cardinalities* by Anirban Dasgupta, Kevin Lang, Lee Rhodes and Justin Thaler. This paper has been accepted for presentation and publication at ICDT 2016. A pre-publication version of the paper is available at the Cornell University Archive.

Our Experience at Yahoo

Our experience at Yahoo in using this library has had a profound impact on a number of our internal platforms that must deal with massive data. Processing times for distinct count operations have been reduced by orders-of-magnitude. The mergeability and additivity property of sketches has enabled the simplification of complex reporting systems that had many thousands of process steps down to a few dozen. And recently, Yahoo made available real-time user count metrics for Flurry that enabled mobile app developers to view the number of distinct users visiting their application within 15 seconds of real-time. All of this has been made possible with the DataSketches Library.

[8] MurmurHash3_x64_128 r150