DataSketchesProposal

Apache DataSketches Proposal

Abstract

DataSketches.GitHub.io is an open source, high-performance library of streaming algorithms commonly called "sketches" in the data sciences. Sketches are small, stateful programs that process massive data as a stream and can provide approximate answers, with mathematical guarantees, to computationally difficult queries orders-of-magnitude faster than traditional, exact methods.

This proposal is to move DataSketches.GitHub.io in to the Apache Software Foundation(ASF) incubation process, transferring ownership of its copyright intellectual property to the ASF. Thereafter, it would be officially known as Apache DataSketches and its evolution and governance would come under the rules and guidance of the ASF.

Introduction

The DataSketches library contains carefully crafted implementations of sketch algorithms that meet rigorous standards of quality and performance and provide capabilities required for large-scale production systems that must process and analyze massive data. The DataSketches core repository is written in Java with a parallel core repository written in C++ that includes Python wrappers. The DataSketches library also includes special repositories for extending the core library for Apache Hive and Apache Pig. The sketches developed in the different languages share a common binary storage format so that sketches created and stored in Java, for example, can be fully used in C++, and visa versa. Because the stored sketch "images" are just a "blob" of bytes (similar to picture images), they can be shared across many different systems, languages and platforms.

The DataSketches website includes general tutorials, a comprehensive research section with references to relevant academic papers, extensive examples for using the core library directly as well as examples for accessing the library in Hive, Pig, and Apache Spark.

The DataSketches library also includes a characterization repository for long running test programs that are used for studying accuracy and performance of these sketches over wide ranges of input variables. The data produced by these programs is used for generating the many performance plots contained in the documentation website and for academic publications.

The code repositories used for production are versioned and published to Maven Central on periodic intervals as the library evolves.

The DataSketches library also includes several experimental repositories for use-cases outside the large-scale systems environments, such as sketches for mobile, IoT devices (Android), command-line access of the sketch library, and an experimental repository for vector-based sketches that performs approximate Singular Value Decomposition (SVD) analysis that could potentially be used in Machine Learning (ML) applications.

Background

The DataSketches library was started in 2012 as internal Yahoo [1] project to dramatically reduce time and resources required for distinct (unique) counting. An extensive search on the Internet at the time yielded a number of theoretical papers on stochastic streaming algorithms with pseudocode examples, but we did not find any usable open-source code of the quality we felt we needed for our internal production systems. So we started a small project (one person) to develop our own sketches working directly from published theoretical papers.

The DataSketches library was designed from the start with the objective of making these algorithms, usually only described in theoretical papers, easily accessible to systems developers for use in our internal production systems. By necessity, the code had to be of the highest quality and thoroughly tested. The wide variety of our internal production systems drove the requirement that the sketch implementations had to have an absolute minimum of external, run-time dependencies in order to simplify integration and troubleshooting.

Our internal experiments demonstrated dramatic positive impact on the performance of our systems. As a result, the DataSketches library quickly evolved to include different types of sketches for different types of queries, such as frequent-items (a.k.a, heavy-hitters) algorithms, quantile/histogram algorithms, and weighted and unweighted sampling algorithms.

We quickly discovered that developing these sketch algorithms to be truly robust in production environments is quite difficult and requires deep understanding of the underlying mathematics and statistics as well as extensive experience in developing high quality code for 24/7 production systems. This is a difficult combination of skills for any one organization to collect and maintain over time. It became clear that this technology needed a community larger than Yahoo to evolve. In November, 2015, this factor, along with Yahoo's strong experience and support of open source, led to the decision to open source this technology under an Apache 2.0 license on GitHub. Since that time our community has expanded considerably and the key contributors to this effort includes leading research scientists from a number of universities as well as practitioners and researchers from a number of major corporations. The core of this group is very active as we meet weekly to discuss research directions and engineering priorities.

It is important to note that our internal systems at Yahoo use the current public GitHub open source DataSketches library and not an internal version of the code.

The close collaboration of scientific research and engineering development experience with actual massive-data processing systems has also produced new research publications in the field of stochastic streaming algorithms, for example:

- Daniel Anderson, Pryce Bevan, Kevin J. Lang, Edo Liberty, Lee Rhodes, and Justin Thaler. A high-performance algorithm for identifying frequent items in data streams. In ACM IMC 2017.
- Anirban Dasgupta, Kevin J. Lang, Lee Rhodes, and Justin Thaler. A framework for estimating stream expression cardinalities. In *EDBT/ICDT Proceedings '16, pages 6:1–6:17, 2016.
- Mina Ghashami, Edo Liberty, Jeff M. Phillips. Efficient Frequent Directions Algorithm for Sparse Matrices. In ACM SIGKDD Proceedings '16, pages 845-854, 2016.

- Zohar S. Karnin, Kevin J. Lang, and Edo Liberty. Optimal quantile approximation in streams. In IEEE FOCS Proceedings '16, pages 71–78, 2016.
- Kevin J Lang. Back to the future: an even more nearly optimal cardinality estimation algorithm. arXiv preprint https://arxiv.org/abs/1708.06839, 2017.
- Edo Liberty. Simple and deterministic matrix sketching. In ACM KDD Proceedings '13, pages 581– 588, 2013 (Received Best-Paper Award, KDD is one of the largest data mining focused conferences in the world).
- Edo Liberty, Michael Mitzenmacher, Justin Thaler, and Jonathan Ullman. Space lower bounds for itemset frequency sketches. In ACM PODS Proceedings '16, pages 441–454, 2016.
- Michael Mitzenmacher, Thomas Steinke, and Justin Thaler. Hierarchical heavy hitters with the space saving algorithm. In SIAM ALENEX Proceedings '12, pages 160–174, 2012.
- Edo Liberty and Zohar Karnin. Discrepancy, Coresets, and Sketches in Machine Learning. To be published.
- Arik Rinberg, Alexander Spiegelman, Edward Bortnikov, Eshcar Hillel, Idit Keidar, Hadar Serviansky, Fast Concurrent Data Sketches. https://arxiv.org/abs/1902.10995 - under submission

International Keynotes and Tutorials on Sketching by the Team

- Edo Liberty - Southern Data Conference - Upcoming
- Edo Liberty - Information Theory and Applications (ITA) 2019
- Edo Liberty - TMA Experts Summit 2018
- Edo Liberty - Streaming Quantiles - Shonan Workshop on Processing Big Data Streams 2017
- Edo Liberty - MLConf 2014, NYC - Introducing the Streaming Computation Model
- Edo Liberty and Jelani Nelson - Full tutorial at KDD 2012 (Slides)
- Lee Rhodes - DataSketches, a Required Toolkit for Analysis of Big Data. Hadoop Summit 2015
- Lee Rhodes - DataSketches. Alan Turing Institute, Invited talk. May 2017.

The Rationale for Sketches

In the analysis of big data there are often problem queries that don't scale because they require huge compute resources and time to generate exact results. Examples include count distinct, quantiles, most frequent items, joins, matrix computations, and graph analysis.

If we can loosen the requirement of "exact" results from our queries and be satisfied with approximate results, within some well understood bounds of error, there is an entire branch of mathematics and data science that has evolved around developing algorithms that can produce approximate results with mathematically well-defined error properties.

With the additional requirements that these algorithms must be small (compared to the size of the input data), sublinear (the size of the sketch must grow at a slower rate than the size of the input stream), streaming (they can only touch each data item once), and mergeable (suitable for distributed processing), defines a class of algorithms that can be described as small, stochastic, streaming, sublinear mergeable algorithms, commonly called sketches (they also have other names, but we will use the term sketches from here on).

To qualify as a sketching algorithm we believe it must exhibit the following major properties:

- Streaming. To be truly streaming a sketch can only "touch" each item of the stream once. There is no second chance.
- Amortized per item processing time is constant. Sketch algorithms are designed so that the processing time per item is essentially independent of n, the number of items (or size) of the input stream. There may be specific instances where upon receiving a new item, the sketch needs to resize or rebuild its internal data structures to accommodate more items, but overall, these events are rare, so that when amortized over n, the computational cost is effectively O(1) with a small hidden constant.
- Small Size. Relative to the size of the input stream, sketches retain a small amount of information about the state of the stream observed so far. Sketch sizes are often orders-of-magnitude smaller, typically just kilobytes in size, even for very long input streams. The smaller the retained information in the sketch the faster it can be examined or processed.
- Sublinear space growth. The sublinear property means that as the number of items in the input stream, n, grows, the size of the sketch must not grow proportionately, it must grow much less than that. To be useful, sketch algorithms should grow logarithmically or sub-logarithmically with the size of the input stream, or not at all. Not only must the sketch start small, it must remain small as n increases.
- Mergeability. In order to be useful in large distributed systems, sketches must be mergeable. This means that the internal data structures of two sketches can be combined in a linear or "additive" fashion. Let A and B be streams and + indicate concatenation:
    - *sketch.merge(sketch(A), sketch(B)) sketch(A + B).*
- Some sketches extend mergeability to include operations other than concatenation or union, such as set intersection and set difference. Mergeability enables input streams to be automatically processed in a distributed and parallel manner: split the stream up arbitrarily across many machines, process each piece separately, and merge the resulting sketches.
- Mathematically proven error properties. It is important that results obtained from a sketch have clearly defined and useful error properties. For example, a sketch with a specific configuration might specify that with 95% confidence, the exact answer will be within plus-or-minus 1% of the estimated result produced by the sketch. In order to make such a statement the sketch must essentially be stream independent, which is to say that the sketch must deliver on its claimed error properties independent of the length, the order, the range of values, and the distribution of the items in the input stream. There is always the possibility that with certain stream lengths, orders, or distributions, that the actual sketch error might be considerably better, even zero, but the claimed error properties still hold.
    - In order to make claims like this, the theory behind sketches must include rigorous mathematical proofs that demonstrate that, if implemented correctly, the sketch will produce results with the claimed error properties regardless of the input stream.
    - The sketch is essentially a complex state machine and combined with some arbitrary input stream defines a stochastic process. We then apply probabilistic methods to interpret the states of the stochastic process in order to extract useful information about the input stream itself. The resulting information will be approximate, but we also use additional probabilistic methods to extract an estimate of the likely probability distribution of error.
    - The error of the sketch result for a given input stream is a random variable, and "mathematically proven error properties" means that this random variable has a probability distribution with a mean and a variance that is well-understood. More generally, the estimate returned by a sketch will be within some error of the true answer to any query, with a specified statistical confidence. The definition of this error is determined by the mathematics underlying the specific sketch and can be either additive (i.e., the sketch returns the correct result in the range *result* ± , or multiplicative (i.e., the sketch returns the correct result in the range *result * (1 ±))*.
    - The mathematical proofs should also include the merge process and associated algorithms. It is important to note that there must be no error penalty for merging. I.e., the error properties of the merged sketch must be no worse than the error of a single sketch produced directly from the concatenated streams.

Because sketches are small they can be processed extremely fast, often many orders-of-magnitude faster than traditional exact computations. For interactive queries there may not be other viable alternatives, and in the case of real-time analysis, sketches are the only known solution.

For any system that needs to extract useful information from massive data sketches are essential tools that should be tightly integrated into the system's analysis capabilities. This technology has helped Yahoo successfully reduce data processing times from days to hours or minutes on a number of its internal platforms and has enabled subsecond queries on real-time platforms that would have been infeasible without sketches.

Sketch Algorithms and Science

There is a significant scientific contribution here that is defining the state machine, understanding the resulting stochastic process, developing the probabilistic methods, and proving mathematically, that it all works! This is why the scientific contributors to this project are a critical and strategic component to our success. The development engineers translate the concepts of the proposed state machine and probabilistic methods into production-quality code. Even more important, they work closely with the scientists, feeding back system and user requirements, which leads not only to superior product design, but to new science as well. A number of scientific papers our members have published (see above) is a direct result of this close collaboration.

Sketching (a.k.a. stochastic, streaming, sublinear mergeable algorithms) is really a distinct area of computer science / computational statistics. It is the science behind sketches that distinguishes sketch algorithms from empirical algorithms that tend to be data-sensitive, and cannot provide a-priori (or even a posteriori) bounds on error.

Sketching is a rather distinct field in that the underlying theory of these algorithms is usually only taught in graduate-level elective courses and only at a few universities that have professors that have experience in this field (see Appendix). For example, a freshly-minted Ph.D. in Machine Learning, does not imply that they have any exposure to this area. Even course-work in algorithms does not imply that this area is taught because it is an advanced topic. There are specific academic workshops on streaming sub-linear algorithms, but no dedicated conferences, yet.

The Rationale for Apache DataSketches

Other open source implementations of sketch algorithms can be found on the Internet. However, we have not yet found any open source implementations that are as comprehensive, engineered with the quality required for production systems, and with usable and guaranteed error properties. Large Internet companies, such as Google and Facebook, have published papers on sketching, however, their implementations of their published algorithms are proprietary and not available as open source.

The DataSketches library already provides integrations with a number of major Apache data processing platforms such as Apache Hive, Apache Pig, Apache Spark and Apache Druid, and is also integrated with a number of other open source data processing platforms such as Splice Machine, GCHQ Gaffer and PostgreSQL.

We want to encourage more synergy with other data processing platforms. In addition to the fundamental capabilities of sketching mentioned above, the DataSketches library provides some additional capabilities specifically designed for large data platforms.

- Binary compatibility across language, platform and history. Binary compatibility means that the stored image of a sketch can be fully interpreted and used by the same type sketch in a different language (e.g., C++, Python) or on a different platform. Our guarantee is that sketches that were produced by the earliest versions of our code can still be read and interpreted by the latest versions of our code. This is critically important for systems that might store years worth of sketches, because it is vastly more efficient than attempting to store years worth of raw data. We have found that this property is even vastly more important than backward compatibility of the APIs. Unfortunately, APIs do have to change and evolve, and while we try hard to avoid this, it sometimes is required.
- Accommodations for specific system architecture or language requirements. Through our work with the Druid team we learned the importance of being able to operate sketches off the java heap. As a result, the sketches that we have currently integrated into Druid's aggregation functions have this off-heap (or Direct) capability. By operate we mean that the sketch is able to be updated, queried, and merged without having to be deserialized on to the Java heap first. Our work with PostgreSQL (C++) team has taught us the importance of enabling user specification of malloc() and free() which can be customized to the environment.

We believe that having DataSketches as an Apache project will provide an immediate, worthwhile, and substantial contribution to the open source community, will have a better opportunity to provide a meaningful contribution to both the science and engineering of sketching algorithms, and integrate with other Apache projects. In addition, this is a significant opportunity for Apache to be the "go-to" destination for users that want to leverage this exciting technology.

Apache DataSketches as a Top-Level Project

Because successful development and implementation of high-performance sketches involves knowledge of advanced mathematics and statistics, there might be a tendency to associate the Apache DataSketches project with Apache Commons-Math or Apache Commons-Statistics. This, I believe, would be a mistake for a couple of reasons.

- Language Support. The Apache Commons-Math, Apache Commons-Statistics, and Apache Commons-Lang libraries are exclusively Java libraries by definition. The DataSketches library supports multiple languages (So far: Java, C++, Python).
- Visibility to data processing platform developers. Sketching is a relatively new field in the arsenal of tools available to system developers. Burying this project under the commons math or commons statistics may make it harder to find. We want to encourage synergy with the various platforms to learn to leverage this technology and to provide feedback to us on capabilities in the design of the sketches themselves.
- Sketches solve difficult computational problems that are desirable queries in large data processing systems, such as unique counts, quantiles, CDFs, PMFs, Histograms, Heavy-hitters (TopN), etc. And they solve these problems in a mergeable and streaming way, which makes them suitable for real-time queries.

Initial Goals

We are breaking our initial goals into short-term (2-6 months) and intermediate to longer-term (6 months to 2 years):

Our short-term goals include:

- Understanding and adapting to the Apache development process and structures.
- Start refactoring codebase and move various DataSketches repositories code to Apache Git repository.
- Continue development of new features, functions, and fixes.
- Specific sub-projects (e.g., C++ and Python) will continue to be developed and expanded.

The intermediate to longer term goals include:

- Completing the design and implementation of the C++ sketches to complement what is already available in Java, and the Python wrappers of those C++ sketches.
- Expanding the C++ build framework to include Windows and the popular Linux variants.
- Continued engagement with the scientific research community on the development of new algorithms for computationally difficult problems that heretofore have not had a sketching solution.

Current Status

The DataSketches GitHub project has been quite successful. As of this writing (Feb, 2019) the number of downloads measured by the Nexus Repository Manager at oss.sonatype.org has grown by nearly a factor of 10 over the past year to about 55 thousand per month. The DataSketches/sketches-core repository has about 560 stars and 141 forks, which is pretty good for a highly specialized library.

Development Practices

Source Control

All of our developers have extensive experience with Git version control and follow accepted practices for use of Pull Requests (PRs), code reviews and commits to master, for example.

Testing

Sketches, by their nature are probabilistic programs and don't necessarily behave deterministically. For some of the sketches we intentionally insert random noise into the code as this gives us the mathematical properties that we need to guarantee accuracy. This can make the behavior of these algorithms quite unintuitive and provides significant challenges to the developer who wishes to test these algorithms for correctness. As a result, our testing strategy includes two major components: unit tests, and characterization tests.

Unit Testing

Our unit tests are primarily quick tests to make sure that we exercise all critical paths in the code and that key branches are executed correctly. It is important that they execute relatively fast as they are generally run on every code build. The sketches-core repository alone has about 22 thousand statements, over 1300 unit tests and code coverage of about 98.2% as measured by Atlassian/Clover. It is our goal for all of our code repositories that are used in production that they have code coverage greater than 90%.

Characterization Testing

In order to test the probabilistic methods that are used to interpret the stochastic behaviors of our sketches we have a separate characterization repository that is dedicated to this. To measure accuracy, for example, requires running thousands of trials at each of many different points along the domain axis. Each trial compares its estimated results against a known exact result producing an error for that trial. These error measurements are then fed into our Quantiles sketch to capture the actual distribution of error at that point along the axis. We then select quantile contours across all the distributions at points along the axis. These contours can then be plotted to reveal the shape of the actual error distribution. These distributions are not at all Gaussian, in fact they can be quite complex. Nonetheless, these distributions are then checked against our statistical guarantees inherent to the specific sketch algorithm and its parameters. There are many examples of these characterization error distributions on our website. The runtimes of these tests can be very long and can range from many minutes to hours, and some can run for days. Currently, we have separate characterization repositories for Java and C++ / Python.

It is our goal that we perform this characterization analysis for all of our sketches. By definition, the code that runs these characterization tests is open-source so others can run these tests as well. We do not have formal releases of this code (because it is not production code) and it is not published to Maven Central.

Meritocracy

DataSketches was initially developed based on requirements within Yahoo. As a project on GitHub, DataSketches has received contributions from numerous individual developers from around the world, dedicated research work from senior scientists at Amazon and Visa, and academic researchers from Georgetown University, Princeton, and MIT.

As a project under incubation, we are committed to expanding our effort to build an environment which supports a meritocracy. We are focused on engaging the community and other related projects for support and contributions. Moreover, we are committed to ensure contributors and committers to DataSketches come from a broad mix of organizations through a merit-based decision process during incubation. We believe strongly in the DataSketches premise that fulfills the concept of a well engineered and scientifically rigorous library that implements these powerful algorithms and are committed to growing an inclusive community of DataSketches contributors and users.

Community

Yahoo has a long history and active engagement in the Open Source community. Major projects include: Vespa.ai, Bullet, Moloch, Panoptes, Screwdriver.cd, Athenz, HaloDB, Maha, Mendel, TensorFlowOnSpark, gifshot, fluxible, as well as the creation, contribution and incubation of many Apache projects such as Apache Hadoop, HBase, Pig, Bookkeeper, Oozie, Zookeeper, Omid, Pulsar, Traffic Server, Storm, Druid, and many more.

Every day, DataSketches is actively used by organizations and institutions around the world for batch and stream processing of data. We believe acceptance will allow us to consolidate existing DataSketches related work, grow the DataSketches community, and deepen connections between DataSketches and other open source projects.

Introduction to the Core Developers & Contributors

The core developers and contributors for DataSketches are from diverse backgrounds, but primarily are scientists that love engineering and engineers that love science. A large part of the value we bring comes from this synthesis. These individuals have already contributed substantially to the code, algorithms, and/or mathematical proofs that form the basis of the library.

This core group also form the Initial Committers with write permissions to the repository. Those marked with (*) Meet weekly to plan the research and engineering direction of the project.

Scientists That Love Engineering

- Eshcar Hillel: Senior Research Scientist, Yahoo Labs, Israel. Interests: distributed systems, scalable systems and platforms for big data processing, concurrent algorithms and data structures,
- Kevin Lang: (*) Distinguished Research Scientist, Yahoo Labs, Sunnyvale, California. Interests: algorithms, theoretical and applied mathematics, encoding and compression theory, theoretical and applied performance optimization.
- Edo Liberty: (*) Director of Research, Head of Amazon AI Labs, Palo Alto, California. Manages the algorithms group at Amazon AI. We build scalable machine learning systems and algorithms which are used both internally and externally by customers of SageMaker, AWS's flagship machine learning platform.
- Jon Malkin: (*) Senior Scientist, Yahoo Labs, Sunnyvale. Interests: Computational advertising, machine learning, speech recognition, data-driven analysis, large scale experimentation, big data, stream/complex event processing
- Justin Thaler: (*) Assistant Professor, Department of Computer Science, Georgetown University, Washington D.C. Interests: algorithms and computational complexity, complexity theory, quantum algorithms, private data analysis, and learning theory, developing efficient streaming and sketching algorithms

Engineers That Love Science

- Roman Leventov: Senior Software Engineer, Metamarkets / Snap. Interests: design and implementation of data storing and data processing (distributed) systems, performance optimization, CPU performance, mechanical sympathy, JVM performance, API design, databases, (concurrent) data structures, memory management, garbage collection algorithms, language design and runtimes (their tradeoffs), distributed systems (cloud) efficiency, Linux, code quality, code transformation, pure functional programming models, Haskell.
- Lee Rhodes: (*) Distinguished Architect, lead developer and founder of the DataSketches project, Yahoo, Sunnyvale, California. Interests: streaming algorithms, mathematics, computer science, high quality and high performance code for the analysis of massive data, bridging the divide between theory and practice.
- Alexander Saydakov: (*) Senior Software Engineer, Yahoo, Sunnyvale, California. Interests: applied mathematics, computer science, big data, distributed systems.

Introduction to Additional Interested Contributors

These folks have been intermittently involved and contributed, but are strong supporters of this project.

- Mina Ghashami: [mina.ghashami at gmail dot com] Ph.D. Computer Science, Univ of Utah. Interests: Machine Learning, Data Mining, matrix approximation, streaming algorithms, randomized linear algebra.

- Christopher Musco: [christopher.musco at gmail dot com] Ph.D. Computer Science, Research Instructor, Princeton University. Interests: algorithmic foundations of data science and machine learning, efficient methods for processing and understanding large datasets, often working at the intersection of theoretical computer science, numerical linear algebra, and optimization.

- Graham Cormode: [g.cormode at warwick.ac dot uk] Ph.D. Computer Science, Professor, Warwick University, Warwick, England. Interests: all aspects of the "data lifecycle", from data collection and cleaning, through mining and analytics. (Professor Cormode is one of the world’s leading scientists in sketching algorithms)

Alignment

The DataSketches library already provides integrations and example code for Apache Hive, Apache Pig, Apache Spark and is deeply integrated into Apache Druid.

Known Risks

The following subsections are specific risks that have been identified by the ASF that need to be addressed.

Risk: Orphaned Products

The DataSketches library is presently used by a number of organizations, from small startups to Fortune 100 companies, to construct production pipelines that must process and analyze massive data. Yahoo has a long-term commitment to continue to advance the DataSketches library; moreover, DataSketches is seeing increasing interest, development, and adoption from many diverse organizations from around the world. Due to its growing adoption, we feel it is quite unlikely that this project would become orphaned.

Risk: Inexperience with Open Source

Yahoo believes strongly in open source and the exchange of information to advance new ideas and work. Examples of this commitment are active open source projects such as those mentioned above. With DataSketches, we have been increasingly open and forward-looking; we have published a number of papers about breakthrough developments in the science of streaming algorithms (mentioned above) that also reference the DataSketches library. Our submission to the Apache Software Foundation is a logical extension of our commitment to open source software.

Key committers at Yahoo with strong open source backgrounds include Aaron Gresch, Alan Carroll, Alessandro Bellina, Anastasia Braginsky, Andrews Sahaya Albert, Arun S A G, Atul Mohan, Brad McMillen, Bryan Call, Daryn Sharp, Dav Glass, David Carlin, Derek Dagit, Eric Payne, Eshcar Hillel, Ethan Li, Fei Deng, Francis Christopher Liu, Francisco Perez-Sorrosal, Gil Yehuda. Govind Menon, Hang Yang, Jacob Estelle, Jai Asher, James Penick, Jason Kenny, Jay Pipes, Jim Rollenhagen, Joe Francis, Jon Eagles, Kihwal Lee, Kishorkumar Patil, Koji Noguchi, Kuhu Shukla, Michael Trelinski, Mithun Radhakrishnan, Nathan Roberts, Ohad Shacham, Olga L. Natkovich, Parth Kamlesh Gandhi, Rajan Dhabalia, Rohini Palaniswamy, Ruby Loo, Ryan Bridges, Sanket Chintapalli, Satish Subhashrao Saley, Shu Kit Chan, Sri Harsha Mekala, Susan Hinrichs, Yonatan Gottesman, and many more.

All of our core developers are committed to learn about the Apache process and to give back to the community.

Risk: Homogeneous Developers

The majority of committers in this proposal belong to Yahoo due to the fact that DataSketches has emerged from an internal Yahoo project. This proposal also includes developers and contributors from other companies, and who are actively involved with other Apache projects, such as Druid. We expect our entry into incubation will allow us to expand the number of individuals and organizations participating in DataSketches development.

Risk: Reliance on Salaried Developers

Because the DataSketches library originated within Yahoo, it has been developed primarily by salaried Yahoo developers and we expect that to continue to be the case near term. However, since we placed this library into open-source we have had a number of significant contributions from engineers and scientists from outside of Yahoo. We expect our reliance on Yahoo salaried developers will decrease over time. Nonetheless, Yahoo is committed to continue its strong support of this important project.

Risk: Lack of Relationship to other Apache Products

DataSketches already directly interoperates with or utilizes several existing Apache projects.

- Build
    - Apache Maven
- Integrations and adaptors for the following projects naturally have them as dependencies
    - Apache Hive
    - Apache Pig
    - Apache Druid
    - Apache Spark
- Additional dependencies for the above integrations and adaptors include
    - Apache Hadoop
    - Apache Commons (Math)

There is no other Apache project that we are aware of that duplicates the functionality of the DataSketches library.

Risk: An Excessive Fascination with the Apache Brand

With this proposal we are not seeking attention or publicity. Rather, we firmly believe in the DataSketches library and concept and the ability to make the DataSketches library a powerful, yet simple-to-use toolkit for data processing. While the DataSketches library has been open source, we believe putting code on GitHub can only go so far. We see the Apache community, processes, and mission as critical for ensuring the DataSketches library is truly community-driven, positively impactful, and innovative open source software. While Yahoo has taken a number of steps to advance its various open source projects, we believe the DataSketches library project is a great fit for the Apache Software Foundation due to its focus on data processing and its relationships to existing ASF projects.

Risk: Cryptography

DataSketches does not contain any cryptographic code and is not a cryptographic product.

Documentation

The following documentation is relevant to this proposal. Relevant portions of the documentation will be contributed to the Apache DataSketches project.

- DataSketches website: datasketches.github.io.
- DataSketches website repository: github.com/DataSketches/DataSketches.github.io

We will need an apache website for this documentation similar to

- https://datasketches.apache.org

Initial Source

The initial source for DataSketches which we will submit to the Apache Foundation will include a number of repositories which are currently hosted under the GitHub.com/datasketches organization:

All github.com/datasketches repositories including:

- Java Production Code
  - sketches-core: This repository has the core sketching classes, which are leveraged by some of the other repositories. This repository has no external dependencies outside of the DataSketches/memory repository, Java and TestNG for unit tests. This code is versioned and the latest release can be obtained from Maven Central.
  - memory: Low level, high-performance memory data-structure management primarily for off-heap. This code is versioned and the latest release can be obtained from Maven Central.
  - sketches-android: This is a new repository dedicated to sketches designed to be run in a mobile client, such as a cell phone or IoT device. It should be considered experimental. It is not currently versioned or published to Maven Central.
  - sketches-hive: This repository contains Hive UDFs and UDAFs for use within Hadoop grid environments. This code has dependencies on sketches-core as well as Hadoop and Hive. Users of this code are advised to use Maven to bring in all the required dependencies. This code is versioned and the latest release can be obtained from Maven Central.
  - sketches-pig: This repository contains Pig User Defined Functions (UDF) for use within Hadoop grid environments. This code has dependencies on sketches-core as well as Hadoop and Pig. Users of this code are advised to use Maven to bring in all the required dependencies. This code is versioned and the latest release can be obtained from Maven Central.
  - sketches-vector: This is a new repository dedicated to sketches for vector and matrix operations. It is still somewhat experimental. It is versioned and published to Maven Central.
- Java Non-Production Code
  - characterization: This relatively new repository is for code that we use to characterize the accuracy and speed performance of the sketches in the library and is constantly being updated. Examples of the job command files used for various tests can be found in the src /main/resources directory. Some of these tests can run for hours depending on its configuration. This code is not versioned and not published to Maven Central.
  - experimental: This repository is an experimental staging area for code that may eventually end up in another repository. This code is not versioned and not published to Maven Central.
  - sketches-misc: Demos and other code not related to production deployment. We have no plans to publish this to Maven Central in the future.
- C++ and Python Production Code
  - sketches-core-cpp: This is the C+/Python companion to the Java sketches-core. These implementations are binary compatible with their counterparts in Java. In other words, a sketch created and stored in C+ can be opened and read in Java and visa-versa. This site also has our Python adaptors that basically wrap the C++ implementations, making the high performance C++ implementations available from Python. This code will be versioned.
  - sketches-postgres: This site provides the postgres-specific adaptors that wrap the C++ implementations making them available to the Postgres database users. This code will be versioned.
- C++ and Python Non-Production Code
  - characterization-cpp: This is the C++/Python companion to the Java characterization repository. This code will not be versioned.
  - experimental-cpp: This repository is an experimental staging area for C++ code that will eventually end up in another repository. This code will not be versioned.
- Command-Line Tools - Non Production Code

(These may eventually be replaced by Python scripts.)

- sketches-cmd
- homebrew-sketches
- homebrew-sketches-cmd

These projects have always been Apache 2.0 licensed. We intend to bundle all of these repositories since they are all complementary and should be maintained in one project. Prior to our submission, we will combine all of these projects into a new git repository.

Source and Intellectual Property Submission Plan

Contributors to the DataSketches project have also signed the Yahoo Individual Contributor License Agreement (https://yahoocla.herokuapp.com/) in order to contribute to the project.

With respect to trademark rights, Yahoo does not hold a trademark on the phrase “DataSketches.” Based on feedback and guidance we receive during the incubation process, we are open to renaming the project if necessary for trademark or other concerns, but we would prefer not to have to do that.

External Dependencies

All external run-time dependencies are licensed under an Apache 2.0 or Apache-compatible license. As we grow the DataSketches community we will configure our build process to require and validate all contributions and dependencies are licensed under the Apache 2.0 license or are under an Apache-compatible license.

Viewing all the repositories of the current github.com/datasketches organization, there are a number of types of external dependencies:

- Core Java Runtime Dependencies: these are dependencies that would have to be supplied to run the code in the sketches-core-X.Y.Z.jar (from Maven Central), which contains all of the sketch families. Currently there are only two:
- com.yahoo.datasketches/memory. This package is not technically an external dependency since it must be included as part of the Apache DataSketches repository.
- org.slf4j/slf4j-api. A generic interface-only API that enables different logging tools to be plugged in at runtime. Different systems prefer different logging tools and this API allows interfacing with many popular logging tools. If a user does not supply a logging tool, this API behaves as a no-op.
- Java Test Dependencies:
- org.testng/testng. Used for unit testing for all the java repositories
- Java Build Dependencies: All of the Java repositories use Apache Maven, so there is a long list of Maven dependencies plus a few others that plug into Maven such as:
- org.codehaus.mojo
- org.jacoco
- Java Characterization Dependencies The characterization code is not production run-time code and for the user to inspect and run if they wish. Because this repository contains characterization tests for algorithms from external sources by definition it has dependencies on those external sources.
- Java System Integration Adaptor Dependencies: The code in the sketches-pig, sketches-hive repositories by definition rely on Apache Pig, Apache Hive and Apache Hadoop code.
- C++ and Python Repositories: This is still evolving, but we have tried to limit the dependencies to the C and C++ Standard and Boost libraries.
- C++ System Integration Adaptor Dependencies: So far we only have an adaptor for PostgreSQL which has dependencies on PostgreSQL.
- Ruby / Homebrew Command-Line Tool: This has dependencies on Ruby and Homebrew code (for Mac systems).
- Android-based Sketches: So far, this only has dependencies on Java and no other external dependencies.

Required Resources

Mailing Lists

We currently use a mix of mailing lists. We will migrate our existing mailing lists to the following:

- dev@datasketches.incubator.apache.org
- user@datasketches.incubator.apache.org
- private@datasketches.incubator.apache.org
- commits@datasketches.incubator.apache.org
- issues@datasketches.incubator.apache.org
- builds@datasketches.incubator.apache.org
- notifications@datasketches.incubator.apache.org

Source Control

The DataSketches team currently uses Git and would like to continue to do so. We request a Git repository for DataSketches with mirroring to GitHub enabled similar the following:

- https://gitbox.apache.org/repos/asf/incubator-datasketches.git
- https://github.com/apache/incubator-datasketches.git

Issue Tracking

We request the creation of an Apache-hosted JIRA. The DataSketches project is currently using the public GitHub issue tracker and the public Google Groups forum/sketches-user for issue tracking and discussions. We will migrate and combine from these two sources to the Apache JIRA.

Proposed Jira ID: DATASKETCHES

Initial Committers

The following list of individuals have been extremely active in our community and should have write (commit) permissions to the repository.

- Eshcar Hillel [eshcar at verizonmedia dot com]
- Kevin Lang [langk at verizonmedia dot com]
- Roman Leventov [leventov at apache dot org]
- Edo Liberty [edo dot liberty at gmail dot com]
- Jon Malkin [jmalkin at verizonmedia dot com]

- Lee Rhodes [lrhodes at verizonmedia dot com] & [leerho at gmail dot com]
- Alexander Saydakov [saydakov at verizonmedia dot com]
- Justin Thaler [justin dot thaler at georgetown dot edu]

Affiliations

The initial committers are from four organizations: Yahoo, Amazon, Georgetown University, and Metamarkets/Snap.

Champion

Jean-Baptiste Onofré, [jb at nanthrax dot net]

Nominated Mentors

- Liang Chen, [chenliang613 at apache dot org]
- Kenneth Knowles, [kenn at apache dot org]
- Furkan Kamaci, [furkankamaci at gmail dot com]

Sponsoring Entity

- The Apache Incubator **** This is our 1st choice ****
- Apache Druid. The incubating Apache Druid project might also be a logical sponsor. However, DataSketches has applications in many areas of computing outside of Druid so our preference and recommendation is that DataSketches would ultimately be a top-level Apache project.

Appendix

Academic Classes on Streaming and Sketching Algorithms

We have identified only a few universities in the U.S. that offer classes devoted to streaming and sketching algorithms at the advanced graduate level. These courses exist because these topics are not covered in any standard undergraduate or graduate algorithms classes.

These include:

- Amit Chakrabarti's course at Dartmouth (Data Stream Algorithms, last offered 2015): https://www.cs.dartmouth.edu/~ac/Teach/CS35-Fall15/
- Andrew McGregor's course at UMass Amherst: https://people.cs.umass.edu/~mcgregor/courses/CS711S18/index.html. This is course sometimes called "More Advanced Algorithms", and sometimes called "Data Stream Algorithms.
- Justin Thaler's course at Georgetown, called "Streaming Algorithms": http://people.cs.georgetown.edu/jthaler/COSC548.html.
- Paul Beame's course at University of Washington, called Sublinear (and Streaming) Algorithms: https://courses.cs.washington.edu/courses/cse522/14sp/lectures/index.html.
- Jelani Nelson's course at Harvard, called "Sketching Algorithms for Big Data": https://www.sketchingbigdata.org/fall17/.

All of the courses above are taught by theorists, and with the possible exception of Professor Thaler's course, don't cover many of the algorithms or concepts most central to the Data Sketches library.

For example, Amit Chakrabarti's widely used set of lecture notes https://www.cs.dartmouth.edu/~ac/Teach/data-streams-lecnotes.pdf do not discuss HLL, KMV, Theta sketches, or one-pass algorithms for quantiles. And none of these courses (again, with the exception of Thaler's) identify mergeability as a non-negotiable requirement of any sketch. (They do cover linear sketching algorithms (defined by Graham Cormode at Warwick Univ., U.K.), which are mergeable for stream union operations, but these are slow and space-intensive in practice. Many of these courses also cover insert-only stream algorithms that may not be mergeable).

---

1. In 2017 Verizon acquired Yahoo and merged it with previously acquired AOL. The merged entity was originally called Oath, Inc., but has recently been renamed Verizon Media, Inc., a wholly-owned subsidiary of Verizon, Inc. Since Yahoo is the more recognized name, references in this document to Yahoo, are also a reference to Verizon Media, Inc. ↩