1. Artifact README

1.1 Abstract

The artifact contains all the JARs of version 0.12 of the DataSketches library, before it moved into Apache (Incubating), as well as configurations and shell scripts to run our tests. It can support the results found in the evaluated section of our PPoPP'2020 paper Fast Concurrent Data Sketches. To validate the results, run the test scripts and check the results piped in the according text output files.

1.2 Artifact check-list (meta-information)

- **Algorithm: Θ Sketch**
- **Program: Java code**
- **Compilation: JDK 8, and each package is compiled using maven**
- **Binary: Java executables**
- **Run-time environment: Java**
- **Hardware: Ubuntu on 12 core server and 32 core server with hyperthreading disabled**
- **Metrics: Throughput and accuracy**
- **Output: Runtime throughputs, and runtime accuracy**
- **How much time is needed to prepare workflow (approximately)?: Using precomipled packages, none.**
- **How much time is needed to complete experiments (approximately)?: Many hours**
- **Publicly available?: Yes**
- **Code licenses (if publicly available)?: Apache License 2.0**

1.3 Description

1.3.1 How delivered

We have provided all the JAR files we used for running our tests, along with scripts. Meanwhile, the project has migrated to the Apache DataSketches (Incubating) library, which is an open source project under Apache License 2.0, and is hosted with code, API specifications, build instructions, and design documentations on Github.

1.3.2 Hardware dependencies

Our tests require a 12-core Intel Xeon E5-2620 machine, and four Intel Xeon E5-4650 processors, each with 8 cores. Hyper-threading is disabled on both machines..

1.3.3 Software dependencies

Building and running the JAR files requires JDK 8; the files don't compile otherwise. To use the automated scripts, we require python3 and git to be installed. The Apache DataSketches (Incubating) library has been tested on Ubuntu 12.04/14.04, and is expected to run correctly under other Linux distributions.

1.4 Installation

First, clone the repository:

```
$ git clone https://github.com/ArikRinberg/FastConcurrentDataSketchesArtifact
```

We have provided the necessary JAR files for recreating our experiment in the repository. We have also provided the repository as a zip file to this artifact.

1.5 Experiment workflow

For convenience, we provide the JAR files required and the configurations used to run our tests.

1. After cloning the repository:

```
$ cd FastConcurrentDataSketchesArtifact
```

   In the current working directory, there should be the following JAR files:

   - memory-0.12.1.jar
   - sketches-core-0.12.1-SNAPSHOT.jar
   - characterization-0.1.0-SNAPSHOT.jar

2. Next, run the tests:

```
$ python3 run_test.py TEST
```

   Where TEST is one of the following: figure_1, figure_6_a, figure_6_b, figure_7, figure_8, figure_9, or table_2.

3. The results of each test will be in txt files in the current working directory, either SpeedProfile or AccuracyProfile:

   SpeedProfile: The txt file contains three columns: **InU** – the number of unique items (the x axis of most graphs), **Trials** – the number of trials for this run, **nS/u** – nano seconds per update. The y axis of the throughput graphs is given as updates per second, therefore a conversion is needed.

   AccuracyProfile: The txt file contains the columns corresponding to the figure legend, where **InU** is the number of unique items. And, for example, $Q(.5)$ corresponds to the 50^{th} precentile.

1.6 Evaluation and expected result

For Figures 1, 7, 8 and 9, the expected results are runtime throughput in nanoseconds per update. For Figure 6, the expected results are accuracy.

Figure 5 will be removed from the camera ready version, and therefore is not supported.

1.7 Experiment customization

Each curve in each experiment is customised in the corresponding configure file. The main customisations for the conf files are:

- **Trials_lgMinU / Trials_lgMaxU:** Range of number of unique numbers over which to run the test.
- **LgK:** Log size of the global sketch.
- **CONCURRENT_THETA_localLgK:** Log size of the local sketch.

- **CONCURRENT_THETA_maxConcurrencyError:** Maximum error due to concurrency. For non-eager tests, set to 1.
- **CONCURRENT_THETA_numWriters:** Number of writer threads.
- **CONCURRENT_THETA_numReaders:** Number of background reader threads. For our mixed workload, we used 10 reader threads.
- **CONCURRENT_THETA_ThreadSafe:** Is true if the test should use the concurrent implementation, false if the test should use a lock-based implementation.

1.8 Working with source files

Alternatively, follow the build instructions on Apache DataSketches (Incubating) apache page (`https://datasketches.apache.org/`), in order to building the above mentioned JAR files, now called:

- incubator-datasketches-java (`https://github.com/apache/incubator-datasketches-java`)
- incubator-datasketches-memory (`https://github.com/apache/incubator-datasketches-memory`)
- incubator-datasketches-characterization (`https://github.com/apache/incubator-datasketches-characterization`)

The version number of incubator-datasketches-java and incubator-datasketches-memory must comply with the version numbers required by incubator-datasketches-characterization. The characterization JAR file is an unsupported open-source code base, and does not pretend to have the same level of quality as the primary repositories. These characterization tests are often long running (some can run for days) and very resource intensive, which makes them unsuitable for including in unit tests. The code in this repository are some of the test suites we use to create some of the plots on our website and provide evidence for our speed and accuracy claims. Alternatively, the datasketches-memory and datasketches-java releases are provided from Maven Central using the Nexus Repository Manager. Go to `repository.apache.org` and search for "datasketches".

For convenience we have included these repositories as modules in our main repository along with specific branches and commit id's that are known to compile. To compile the jar files:

```
$ git clone https://github.com/ArikRinberg/FastConcurrentDataSketchesArtifact
$ cd FastConcurrentDataSketchesArtifact
$ source customCompile.sh
```

The shell script takes care of initialising the submodules, building the source files, and copying the correct JAR files to the current directory.

Workflow for custom JAR files.

1. After cloning the repository:

   ```
   $ cd FastConcurrentDataSketchesArtifact
   ```

   In the current working directory, there should be the following JAR files:

   - datasketches-memory-1.1.0-incubating.jar
   - datasketches-java-1.1.0-incubating.jar
   - datasketches-characterization-1.0.0-incubating-SNAPSHOT.jar

2. For each .conf file in the conf_files folder, the following line must be altered:

   From: JobProfile=
   com.yahoo.sketches.characterization.uniquecount.TEST

   To: JobProfile=
   org.apache.datasketches.characterization.theta.concurrent.TEST

   Where TEST is either ConcurrentThetaAccuracyProfile or ConcurrentThetaMultithreadedSpeedProfile.

3. Finally, the following line must be altered in run_test.py:

   From: CMD=
   'java -cp "./*" com.yahoo.sketches.characterization.Job '

   To: CMD='java -cp "./*" org.apache.datasketches.Job '

4. The tests can now be run as explained in Item 3.