

AWARE: Workload-aware, Redundancy-exploiting Linear Algebra: Reproducibility Guide

This report provides a guide to reproduce the SIGMOD AWARE paper. There are a few setup requirements to make the implementation run. It is recommended to follow the latest guide at: <https://github.com/damslab/reproducibility>

To produce a full reproduction, a Spark (v 3.2.0) cluster with Hadoop (v 3.3.1) running Java 11 is required, but any single machine should be able to run small-scale experiments covering most of the experiments.

The compute resources used in the paper are 6 Cluster nodes and a main node with 32 virtual cores and 128 GB RAM each. Local storage requirements are at least 100GB and distributed HDFS is 2.0 TB. The machines have similar specifications as m5a.8xlarge instances in AWS. In AWS, it should be simple to start a Spark cluster, but setting such up is not covered in this guide. Note that if one wants to, it should be possible to use newer Hadoop and Spark versions.

Further dependencies are:

- Java 11 and 8 available on main node
- Maven 3.6+
- Git
- rsync (installed per default on Ubuntu)
- ssh (also installed per default on Ubuntu)
- Python 3.6+
- pdflatex - If you want to make the paper.

1 Verification

We verify the setup is correct, and run the following in a terminal:

```
1 java -version
2 mvn -version
3 git --version
4 python3 --version
```

The output should look something like:

```
1 Me:~/github/reproducibility/sigmod2023-AWARE-p5$
   java -version
2 -versionopenjdk version "11.0.16" 2022-07-19
3 OpenJDK Runtime Environment (build 11.0.16+8-post-
   Ubuntu-0ubuntu120.04)
4 OpenJDK 64-Bit Server VM (build 11.0.16+8-post-
   Ubuntu-0ubuntu120.04, mixed mode, sharing)
5 Me:~/github/reproducibility/sigmod2023-AWARE-p5$
   mvn -version
6 Apache Maven 3.8.3 (
   ff8e977a158738155dc465c6a97ffaf31982d739)
7 Maven home: /home/baunsgaard/maven/mvn
8 Java version: 11.0.16, vendor: Ubuntu, runtime: /
   usr/lib/jvm/java-11-openjdk-amd64
9 Default locale: en_US, platform encoding: UTF-8
10 OS name: "linux", version: "5.15.0-46-generic",
   arch: "amd64", family: "unix"
11 Me:~/github/reproducibility/sigmod2023-AWARE-p5$
   git --version
12 git version 2.25.1
13 Me:~/github/reproducibility/sigmod2023-AWARE-p5$
   python3 --version
14 Python 3.8.10
```

For the distributed parts of the experiments further installs are needed, therefore verify the spark and hadoop install with:

```
1 spark-submit --version
2 hdfs version
```

The output should look like:

```
1 $ spark-submit --version
2 WARNING: An illegal reflective access operation has
   occurred
3 WARNING: Illegal reflective access by org.apache.
   spark.unsafe.Platform (file:/home/sbaunsgaard/
   spark-3.2.0-bin-hadoop3.2/jars/spark-unsafe_2
   .12-3.2.0.jar) to constructor java.nio.
   DirectByteBuffer(long,int)
4 WARNING: Please consider reporting this to the
   maintainers of org.apache.spark.unsafe.Platform
5 WARNING: Use --illegal-access=warn to enable
   warnings of further illegal reflective access
   operations
6 WARNING: All illegal access operations will be
   denied in a future release
7 Welcome to
8
9      _--_
10     /  _/  _/  _/  _/  _/  _/  _/  _/  _/  _/
11    /___/  _/  _/  _/  _/  _/  _/  _/  _/  _/  version 3.2.0
12     _/
13
14 Using Scala version 2.12.15, OpenJDK 64-Bit Server
   VM, 11.0.13
15 Branch HEAD
16 Compiled by user ubuntu on 2021-10-06T12:46:30Z
17 Revision 5d45a415f3a29898d92380380cfd82bfc7f579ea
18 Url https://github.com/apache/spark
19 Type --help for more information.
20
21 $ hdfs version
22 Hadoop 3.3.1
23 Source code repository https://github.com/apache/
   hadoop.git -r
   a3b9c37a397ad4188041dd80621bdeefc46885f2
24 Compiled by ubuntu on 2021-06-15T05:13Z
25 Compiled with protoc 3.7.1
26 From source with checksum 88
   a4ddb2299aca054416d6b7f81ca55
27 This command was run using /home/hadoop/hadoop
   -3.3.1/share/hadoop/common/hadoop-common-3.3.1.
   jar
```

If any of the parts are missing or returns errors then please install the missing components. For our setup it is a further advantage if you are able to switch the Spark and Hadoop version to be able to run CLA baselines, since older versions are needed. If it is not possible to switch then all SystemML experiments will not work.

From this point Code is run inside the experiments folder

2 Install

Next we install SystemDS, SystemML, and a Python virtual environment to run baselines, experiments and plotting.

```
1 ./install-all.sh
```

To verify the install we run a few simple scripts.

```
1 ./verify-install.sh
```

The output should be like:

```
1 SYSTEMDS
2 22/09/09 16:51:48 INFO api.DMLScript: BEGIN DML run
   09/09/2022 16:51:47
3 22/09/09 16:51:48 INFO api.DMLScript: Process id:
   725211
4 7.000 4.000 4.000
5 4.000 1.000 8.000
6 7.000 7.000 9.000
7
8 SystemDS Statistics:
9 Total execution time: 0.065 sec.
10
11 22/09/09 16:51:48 INFO api.DMLScript: END DML run
   09/09/2022 16:51:48
12 SYSTEMML
13 22/09/09 16:51:49 INFO api.DMLScript: BEGIN DML run
   09/09/2022 16:51:49
14 22/09/09 16:51:49 INFO api.DMLScript: HADOOP_HOME:
   /home/hadoop/hadoop-2.7.7
15 4.000 7.000 4.000
16 4.000 4.000 1.000
17 8.000 7.000 7.000
18
19 SystemML Statistics:
20 Total execution time: 0.024 sec.
21 Number of executed MR Jobs: 0.
22
23 22/09/09 16:51:50 INFO api.DMLScript: END DML run
   09/09/2022 16:51:50
```

3 Parameters

Before starting the experiments or downloading datasets, we suggest going through the settings to configure the execution of the experiments.

While it is possible to run everything out of the box, in one go, we suggest to go through some setting first in:

experiments/parameters.sh

Note that you have to set the name of the machine you are using and memory settings. It is further possible to change: What version of SystemDS to install, what directory to run the experiments in, settings for remote synchronization of results, change base parameters for the JVM, and much more.

4 Data Preparation

To download and prepare the datasets for local execution use:

```
1 ./setup_local_data.sh
```

Once done, verify the download, by running it again. On the second run it should get done almost instantly and report back:

```
1 Beginning download of Census
2 Census is already downloaded
3 Already constructed metadata for census.csv
4 Already saved training data census.
5 Already saved encoded training data census.
6 Census Download / Setup Done
7
8
9 Downloading Covtype
10 Covtype already downloaded
11 Already setup train_covtype
12 Already setup train_covtype new format ... (
   predicting cov type)
13 Saving covtype training as csv already done.
14 CovType Setup Done
15
16
17 Beginning download of mnist
18 Download part of Mnist is already done p1
19 Download part of Mnist is already done p2
20 Download part of Mnist is already done p3
21 Download part of Mnist is already done p4
22 Unzip of part MNIST already done p1
23 Unzip of part MNIST already done p2
24 Unzip of part MNIST already done p3
25 Unzip of part MNIST already done p4
26 Saving of CSV already done for MNIST
27 Saving of SystemDS binary already done for MNIST
28 Mnist Setup Done
29
30
31 Beginning download of Infinimnist
32 Infinimnist is already downloaded
33 Infinimnist is already unpacked training (2mil)
34 Infinimnist is already unpacked labels (2mil)
35 Infinimnist is already unpacked training (1mil)
36 Infinimnist is already unpacked labels (1mil)
37 Saving 2 mil to csv already done
38 Saving 1 mil to csv already done
39 Saving SystemDS binary of 2 mil already done
40 Saving SystemDS binary of 1 mil already done
41
42
43 Airlines zip already downloaded
44 Airlines already unzipped
45 Airlines already preprocessed to Binary
46 Airlines already preprocessed to CSV
47 Airlines Setup Done
```

5 Run All

If you want to run all experiments (with few exceptions that can be re-enabled) then simply execute the run script. The script can be modified to allow you to select individual experiments to run.

```
1 ./run.sh
```

Note that this will take significant time, and you might want better control of what experiment is run at which time. To select specific experiments comment in and out parts of run.sh, for instance, to make table 5 in the paper comment in:

```
1 # ./code/compression/tab5.sh
```

Some of the experiments are, by default, commented out in the run.sh script. The last experiment is commented out because it takes 35 hours (before it crashes intentionally).

If there are errors or other problems in the execution it is not reported to the user. Instead, all logging is saved into the result files from the run experiments. Sometimes if there are issues with the setup some experiments fail. Errors in experiments are handled in the plotting, and plots do not allow the failed tests to plot correctly. Therefore, if a result is missing in the plots, it is most likely because one of the experiments failed.

6 Plotting

To plot simply use:

```
1 ./plot.sh
```

This generate summary tables and plots of the results from the experiments.

7 Paper

The source code of the paper is located in the report folder. To remake the paper using PDFLatex.

The code inserts the results from the experiments automatically. In a few exceptions, the results can be found in CSV files. In these cases, the results have to manually be copied into the paper.