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

This is the guide to reproduce the paper. To make the execution work there is a few setup requirements. But first it is sincerely recommended to follow the latest guide at: `https://github.com/damslab/reproducibility.git`

To produce a full reproduction of the paper it is expected that one have access to a Spark (v 3.2.0) cluster with Hadoop (v 3.3.1) running Java 11. But any single machine should be able to run small scale experiments covering most of the experiments shown. For baseline experiments

The specifics of the resources used in the paper is: 6 Cluster nodes and a main node with 32 virtual cores and 128 GB RAM each. Local storage requirements is at least 100GB, and distributed HDFS is 2.0 TB.

The machines have similar specifications as m5a.8xlarge in AWS. In there 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 has to be able to switch hadoop and spark versions, to run all baseline experiments.

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

First we verify the setup is correct.

```
java -version
mvn -version
git --version
python3 --version
```

The output should look something like:

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

For the distributed parts of the experiments further installs are needed: Verify the install of :

```
spark-submit --version
hdfs version
```

Output should look like:

```
$ spark-submit --version
WARNING: An illegal reflective access operation has occurred
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)
WARNING: Please consider reporting this to the maintainers of org.apache.spark.unsafe.Platform
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ ‘/ __/  ’_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.2.0
      /_/

Using Scala version 2.12.15, OpenJDK 64-Bit Server VM, 11.0.13
Branch HEAD
Compiled by user ubuntu on 2021-10-06T12:46:30Z
Revision 5d45a415f3a29898d92380380cfd82bfc7f579ea
Url https://github.com/apache/spark
Type --help for more information.

$ hdfs version
Hadoop 3.3.1
Source code repository https://github.com/apache/hadoop.git -r a3b9c37a397ad4188041dd80621bdeefc46885f2
Compiled by ubuntu on 2021-06-15T05:13Z
Compiled with protoc 3.7.1
From source with checksum 88a4ddb2299aca054416d6b7f81ca55
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. If it is not possible to switch then all the 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 python.

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

To verify the install we run a few simple scripts.

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

Output should be like:

```
SYSTEMDS
22/09/09 16:51:48 INFO api.DMLScript: BEGIN DML run
     09/09/2022 16:51:47
22/09/09 16:51:48 INFO api.DMLScript: Process id:
    725211
7.000 4.000 4.000
4.000 1.000 8.000
7.000 7.000 9.000

SystemDS Statistics:
Total execution time:   0.065 sec.

22/09/09 16:51:48 INFO api.DMLScript: END DML run
    09/09/2022 16:51:48
SYSTEMML
22/09/09 16:51:49 INFO api.DMLScript: BEGIN DML run
     09/09/2022 16:51:49
22/09/09 16:51:49 INFO api.DMLScript: HADOOP_HOME:
    /home/hadoop/hadoop-2.7.7
4.000 7.000 4.000
4.000 4.000 1.000
8.000 7.000 7.000

SystemML Statistics:
Total execution time:   0.024 sec.
Number of executed MR Jobs: 0.

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 to go 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

Here it is possible to change what version of systemDS to install. What directory to run the experiments in. Settings for remote synchronization of results. Change parameters for JVM. And much more.

4 Data Preparation

To download and prepare the datasets for local execution use:

```
./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:

```
Beginning download of Census
Census is already downloaded
Already constructed metadata for census.csv
Already saved training data census.
Already saved encoded training data census.
Census Download / Setup Done


Downloading Covtype
Covtype already downloaded
Already setup  train_covtype
Already setup train_covtype new format ... (
    predicting cov type)
Saving covtype training as csv already done.
CovType Setup Done


Beginning download of mnist
Download part of Mnist is already done p1
Download part of Mnist is already done p2
Download part of Mnist is already done p3
Download part of Mnist is already done p4
Unzip of part MNIST already done p1
Unzip of part MNIST already done p2
Unzip of part MNIST already done p3
Unzip of part MNIST already done p4
Saving of CSV already done for MNIST
Saving of SystemDS binary already done for MNIST
Mnist Setup Done


Beginning download of Infinimnist
Infinimnist is already downloaded
Infinimist is already unpacked training (2mil)
Infinimnist is alreadt unpacked labels (2mil)
Infinimnist is alreadt unpacked training (1mil)
Infinimnist is alreadt unpacked labels (1mil)
Saving 2 mil to csv already done
Saving 1 mil to csv already done
Saving SystemDS binary of 2 mil already done
Saving SystemDS binary of 1 mil already done


Airlines zip already downloaded
Airlines already unzipped
Airlines already preprocessed to Binary
Airlines alreay preprocessed to CSV
Airlines Setup Done
```

5 Run All

If you want to run all experiments (with few exceptions) then simply execute the run script:

The script is modified per default to allow you to select individual experiments to run.

```
./run.sh
```

Note that this will take significant time, and you might want to control better what experiment is run at which time.

The following sections specify how to do this.

6 Micro benchmarks

7 local execution

8 distributed execution

9 plotting

10 compilation of paper