ScaleRunner: A Fast MPI-based Random Walk Engine for Multi-CPU Systems

Artifact Overview

Florian Willich and Henning Meyerhenke
Humboldt-Universität zu Berlin, Germany

May 12, 2025

1 Introduction

In this overview document we describe all necessary steps to setup, configure, compile, link, run and evaluate all defined experiments presented in the paper *ScaleRunner: A Fast MPI-based Random Walk Engine for Multi-CPU Systems*. This document should serve everyone interested to reproduce our experiments, however we specifically address the EuroPar'25 artifact evaluation committee to serve as the *Overview Document* of our artifact submission.

Please consider that we conducted our experiments on a cluster with 16 compute nodes, each equipped with 2× 12-Core Intel Xeon X6126 (HT) CPUs, and 192 GB RAM interconnected by a 100 GBit Infiniband Omnipath network. C++ code was compiled using GCC (v12.3) and MPICH (v4.2.0).

Running all experiments with the original graph files can take more than 24 hours. Therefore, we provide a small *example graph data* set which we will use to showcase how to run experiments, as well as plotting the measurements in Sec. 2.

Our experiments are defined using SimexPal [1]. We recommend to read the SimexPal documentation [2] in case you experience any issues with SimexPal we have not covered in this document.

We conver all requirements to run experiments and evaluate results in Sec. 2 with an example data set. In Sec. 3, we provide a step-by-step guide to run all experiments on the data set we used in our paper.

[1] `https://github.com/hu-macsy/simexpal`
[2] `https://simexpal.readthedocs.io/en/latest/`

2 Getting Started

2.1 Software Requirements

The following software was used to conduct and evaluate our experiments:

- SimexPal (commit c848baba0baa8e9794bdcc6b9c5d2a507a840953)
- C++ Compiler GCC (v12.3) with support of C++17
- OpenMP (v4.5)
- MPICH (v4.2.0)
- Python3 (v3.12.3)
- Python Jupyter Notebook [3] with IPython (v8.28.0) and ipykernel (v6.29.5)

To install python packages you may want to activate/install a dedicated python virtual environment. We provide a `requirements.txt` file including all Python3 packages needed to run our evaluation script.

To install the right version of SimexPal you can use pip:

```
pip install git+https://github.com/hu-macsy/simexpal.
    git@c848baba0baa8e9794bdcc6b9c5d2a507a840953
```

Or you install SimexPal by cloning the GitHub repository:

```
git clone https://github.com/hu-macsy/simexpal.git
cd simexpal
git checkout c848baba0baa8e9794bdcc6b9c5d2a507a840953
pip install -e .
```

Please install the same SimexPal version (commit) to later evaluate the results using Jupyter Notebook. If you used a special python kernel to install all required packages, please remember to run the Jupyter Notebook using the same python kernel. Also, consider reading the quick start guide [4] for SimexPal if you have issues with the installation or the usage of the tool.

2.2 How To: Build all Targets

The `experiments.yml` file defines all builds, experiments and their parameters, as well as all graph instances used in our experiments. First, we will need to configure, compile and link all builds defined in the `experiments.yml` file. Please navigate to the root directory of the project where the `experiments.yml` is located, then execute:

```
simex develop
```

[3] `https://jupyter.org/install`
[4] `https://simexpal.readthedocs.io/en/latest/quick_start.html`

With this command, you may encounter two issues:

1. Your C++ compiler does not support certain functionalities needed by our programs. To fix this, you can export the required C++ compiler. For example, for the GNU v12 C++ compiler *G++12* located under `/usr/bin` use:

   `export CXX=/usr/bin/g++-12`.

   In case you have already tried to build using `simex develop`: reconfigure using:

   `simex develop -reconfigure`

   instead.

2. If the `scalerunnerConfig.cmake` file cannot be found since you did not provide the `scalerunner_DIR` to the build `io_benchmark`:

   Please navigate in the `experiments.yml` to the build `io_benchmark`. There you will find the configure steps for the build. The `scalerunner_DIR` is either provided using `/lib/...` or `/lib64/...`. Please comment in (and out) one of the lines depending on the configuration of your system. This may also be the case for build `sr_benchmark`. Now, reconfigure using:

   `simex develop -reconfigure` instead.

2.3 How To: Convert Graph Instance Files

Next, we need to convert the *graph instance files* located under directory `/instances` using the python convert script named `convert.py`. Either call the convert script as described in the `experiments.yml` file under key `instances` or use SimexPal:

```
simex instances install
```

The script reads the `experiments.yml` file to get the graph properties from the instance `extra_args` property as parameters.

Using GDSB[5], the python script converts the graph data, writes the binary data to a given output directory (the provided output path can be absolute or relative) and then creates symbolic links to that binary data in the instance directory.

Please be advised that converting large instances can take a while. SimexPal will return once it's finished converting all instances.

2.4 How To: Launch Experiments

Finally, we can run all experiments! In this case we are using the option:

`--launch-through=fork`

to run one experiment after the other in the same process as the one calling SimexPal:

```
simex e launch --launch-through=fork
```

[5] `https://github.com/hu-macsy/graph-ds-benchmark`

MPI clusters often require different runtimes to execute MPI programs. If you need (or want) to change the runtime that executes the programs using MPI, namely the programs `kk_benchmark`, `sr_benchmark`, and `io_benchmark`, you will have to adapt the `experiments.yml` file under the key `experiments` where you find the arguments passed to call the program. There are also instructions to adapt the arguments to your systems requirements when using `srun`.

After launching all experiments, you can list all experiments and their status using:

```
simex e list
```

Which should now look similar to:

```
Experiment                      started    finished   failures       other
----------                      -------    --------   --------       -----
graphanalysis @ _dev                       4/4
graphfile-input ~ gdsb-mpi-io              4/4
graphfile-input ~ original @ _             4/4
kk-crw ~ 80 @ _dev                         4/4
kk-node2vec ~ 80, homopholy @              4/4
kk-node2vec ~ 80, structure @              4/4
scalerunner-crw ~ 80 @ _dev                4/4
scalerunner-node2vec ~ 80, hom             4/4
scalerunner-node2vec ~ 80, str             4/4
scalerunner-scaling1 ~ 80 @ _d             4/4
scalerunner-scaling16 ~ 80 @ _             4/4
scalerunner-scaling2 ~ 80 @ _d             4/4
scalerunner-scaling4 ~ 80 @ _d             4/4
scalerunner-scaling8 ~ 80 @ _d             4/4
56 experiments in total
```

The output tells us that for all defined experiments we have run 4 experiments and all 4 finished (successfully).

If there have been any failed runs and you would like to re-run them after you fixed the issue: purge all experiments (or purge experiments selectively [6]) using:

```
simex e purge --all -f
```

All experiment output files have been written to the output folder: `/output`.

2.5 How To: Evaluate Results

If you ran the experiments on another machine and you would like to archive and copy the results to your local or any other machine, first archive the data:

```
simex archive
```

This will compress the output folder and the `experiments.yml` file and write the file `data.tar.gz`. Copy this file to the machine you want to evaluate the results on, and extract the contents to the top level such that the directory `output` and the `experiments.yml`

[6] `https://simexpal.readthedocs.io/en/latest/command_line_reference.html#develop`

file is on the same level as the `evaluation.ipynb` file. Doing so will overwrite the already present `experiments.yml` file.

We can now evaluate our experiments using the evaluation jupyter notebook script: `evaluation.ipynb`. Please remember that this requires you to have installed a *Jupyter Notebook Environment* [7].

First, the notebook defines all imports, and therefore all python 3 packages you will need to install. As described above, all required python 3 packages are defined in `requirements.txt`. Now, run all cells of the notebook. You will not need to change anything of the script. If you do not want to write all plots to PDF files (written to `/results/plots`): set the global variable `save_all_plots` to `False`.

Please note that the plotted results are not to be interpreted in any way, our example data set is just enough to provide a running example with very small graph instances.

3 Step-by-Step Instructions

In order to reproduce our experiments, the workflow is *nearly* the same as described in the getting started guide in Sec. 2.

3.1 Download all Instances

First, download all instances defined under the key `instances` in the `experiments.yml` file either directly into the instance directory `/instances` or any other folder.

You can use the following wget commands to download all required instances:

```
wget https://snap.stanford.edu/data/web-NotreDame.txt.gz
wget https://nrvis.com/download/data/rt/rt-retweet-crawl.zip
wget http://nrvis.com/download/data/misc/europe_osm.zip
wget https://snap.stanford.edu/data/amazon0601.txt.gz
wget https://snap.stanford.edu/data/web-Google.txt.gz
wget https://snap.stanford.edu/data/roadNet-CA.txt.gz
wget http://nrvis.com/download/data/misc/human_gene2.zip
wget http://nrvis.com/download/data/misc/cage14.zip
wget https://nrvis.com/download/data/dynamic/rec-amazon-ratings.zip
wget https://nrvis.com/download/data/dynamic/rec-epinions-user-ratings.
    zip
wget https://nrvis.com/download/data/inf/inf-road-usa.zip
wget https://nrvis.com/download/data/bn/bn-human-Jung2015_M87125334.zip
wget https://snap.stanford.edu/data/soc-LiveJournal1.txt.gz
wget https://nrvis.com/download/data/soc/soc-orkut.zip
wget https://networkrepository.com/bn-human-Jung2015-M87126525.php
wget https://networkrepository.com/web-uk-2005-all.php
wget https://nrvis.com/download/data/soc/soc-twitter-mpi-sws.zip
```

Unpack all files, and if located in another directory, do not forget to create symbolic links to the files (without the file ending such as '.edges' or '.mtx') in the directory `/instances`. The files must be named exactly as they are named in the instance definition in the `experiments.yml` file.

[7] `https://jupyter.org/install`

You can also choose to run the experiments using only certain instance sets which then have to be specified under the key `matrix` in the `experiments.yml` file where each experiment is provided a list of instance sets under the key `instsets`. You will find the original `instsets` list there as well which we will now need to comment in, and comment out the `example_graphs` instance set.

Once you downloaded and converted all files defined in 'instsets' as described in Sec. 2.3, the list of instances should mark all instances in green when using:

```
simex e list
```

3.2 Launching Experiments

Before we can start launching all experiments please make sure to have a look at the entries under the key `experiments` in the `experiments.yml` file. Here, we currently execute the programs using `mpirun -n 1` for example. Also the `num_nodes` are set to 1. As described in the `experiments.yml` file you will have to set these parameters to run the experiments using 16 MPI processes (or less). Also, the scaling experiments defined under the key `matrix` are currently commented out. As described, please comment them in again to reproduce our experiments.

As an example on how to run experiments using using the *slurm* on a *queue* (also known as partition) here named *core*:

```
simex e launch --launch-through=slurm --queue=core
```

Otherwise use the forked option as described in Sec. 2.4.

Once all experiments are finished, you can *archive* and copy the results as described in Sec. 2.5.

3.3 Evaluate Your Results

Finally you can evaluate the results using the evaluation jupyter notebook script:
`evaluation.ipynb`
as described in Sec. 2.5.

3.4 Evaluate Our Results

We provide the original experimental results data set in:
`/archives/experimental_results-EUROPAR25.zip`
which was used to plot the data of our paper. In order to plot the results: unzip the file into the root directory of this project. Doing so will write an `output` directory and an `experiments.yml` file. The `experiments.yml` file represents the original `experiments.yml` file used to run and evaluate our experiments. For examples and further documentation we have made small changes to the `experiments.yml` file provided in this artifact, so simply rename the current `experiments.yml` file to `experiments.yml.tmp`. The following commands will do all these steps:

```
rm -rf output
mv experiments.yml experiments.yml.tmp
unzip archives/experimental_resutls-EUROPAR25.zip
mv experimental_resutls-EUROPAR25/* .
```

Now you can run the evaluation script which will plot the same figures as we have presented in our paper.