Instructions to Reproduce the Results in the Paper "High-Performance Agent-Based Simulation"

Contents

Overview

This repository contains the necessary files to reproduce the results in the paper: **High-Performance Agent-Based Simulation**.

It consists of code repositories (`SF2-code.tar.gz`) and a self-contained Docker image (`SF3-bdm-publication-image.tar.gz`). We executed all simulations and benchmarks in the paper inside this Docker container.

The table below shows an overview of our benchmark hardware and software configuration.

System	Type	Description
System A	CPU	4x Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz
	Memory	504 GB
	Host operating system	CentOS Linux release 7.9.2009 (Core)
	Kernel	3.10.0-1160.53.1
	Docker version	19.03.13
System B	CPU	4x Intel(R) Xeon(R) CPU E7-8890 v3 @ 2.50GHz
	Memory	1008 GB
	Host operating system	CentOS Linux release 7.9.2009 (Core)
	Kernel	3.10.0-1160.59.1
	Docker version	19.03.5
System C	CPU	2x Intel(R) Xeon(R) E5-2683 v3 CPUs @ 2.00GHz
	Memory	62 GB
	Host operating system	CentOS Stream 8
	Kernel	4.18.0-365
	Docker version	20.10.7

Software	Description
Intel oneAPI	2022.0.0 (build 621730)
BioDynaMo	v1.01-73-g5b6101a3
C++ compiler	g++ (Ubuntu 9.3.0-17ubuntu1~20.04) 9.3.0
Python	3.9.1
CMake	3.19.5
Java	openjdk version “11.0.10” 2021-01-19
Maven	Apache Maven 3.6.3
Cortex3D	0.03 with modifications We used maven as build system, added support for headless execution, and removed calls to `Thread.sleep`.
NetLogo	6.2.0

Please follow the instructions in this file to reproduce all results.

Extract code repositories

Start by downloading the file `SF2-code.tar.gz`. Extract the archive into an empty directory and change into this directory using the following commands. Make sure that the absolute path to this directory contains no spaces.

```
mkdir reproduce
tar -xzf <path-to>/SF2-code.tar.gz -C reproduce
cd reproduce
```

Now, `reproduce` contains all necessary code to produce all results shown in the paper. The code consists of the `biodynamo`, and `bdm-paper-examples` repositories. BioDynaMo is the high-performance simulation engine shown in the paper, and the directory `bdm-paper-examples` contains the simulations, benchmarks, plots, and utilities to generate the results.

License information is available in the following files:

- `biodynamo/LICENSE`
- `bdm-paper-examples/LICENSE`

Directory `metadata` contains more detailed information about the hardware and software configuration of the used systems.

- System A: `metadata/system-a`
- System B: `metadata/system-b`

The `bdm-paper-examples` repository

The following list explains the directory structure.

- `bdm`: This directory contains the five simulations shown in Table 1 in the paper. These simulations are detailed in https://doi.org/10.1093/bioinformatics/btab649.
- `benchmark`: This directory contains the scripts that define the compile and runtime parameters used for a specific study.
- `docker`: This directory contains scripts to build, run, load, save, and connect to a Docker container. It also includes the definition of the Docker image.
- `netlogo`: This directory contains the epidemiology implementation for NetLogo and scripts to execute it.
- `other-tools/cx3d`: This directory includes Cortex3D and its simulation implementations.
- `plot`: This directory contains all matplotlib scripts to combine the raw results into figures for the paper.
- `util`: This directory contains various scripts needed to execute the benchmarks.
- `visualization`: Contains ParaView scripts to visualize the simulations. We did not use these scripts for this publication.

- `run-*.sh`: These scripts execute one or more benchmarks for each simulation and gather metadata about the software and hardware configuration of the underlying machine.

Mapping between simulation names in the paper and `bdm-paper-examples`

Name in the simulation	Name in `bdm-paper-examples`
Cell proliferation	`cell-grow-divide`
Cell clustering	`soma-clustering`
Epidemiology use case	`epidemiology`
Neuroscience use case	`pyramidal-cell`
Oncology use case	`tumor-spheroid`

Install host machine prerequisites

Docker

Please install a recent Docker version (>= 19.0.3) and follow the instructions to use docker as a non-root user.

Intel Vtune Sampling Driver

To collect performance results using Intel Vtune, install the Intel Vtune sampling driver (version 2022.2.0) on the host machine by following this guide.

Linux `screen` command

We use the Linux `screen` command to execute long-running commands over SSH connections. With `screen`, processes are not terminated if we close the SSH session. If this command is not available on the system, please install it:

```
# on Ubuntu
sudo apt install -y screen
# on CentOS
sudo yum install -y screen
```

Here are the most important `screen` commands:

```
# Start a new screen
screen

# Disconnect from the screen by pressing CTRL + a, d
# That means: press 'CTRL' and 'a' at the same time,
# release both, and then type 'd'
```

```
# Reconnect
screen -r
```

Have a look at this tutorial for more details on using `screen`.

Load Docker image

Download the publication's docker image (`SF3-bdm-publication-image.tar.gz`) if you haven't done so already. After finishing this step, load the docker image using the following commands. This step takes around 5 min to complete. The extracted image requires ~30GB of disk space.

```
cd bdm-paper-examples
docker/load.sh <path-to>/SF3-bdm-publication-image.tar.gz
```

If this step completes successfully, you should be able to see an entry for `bdm-publication-image-v7` with ID `9e13327e93a5` in the output of the command `docker images`. A sample output is shown below.

```
REPOSITORY                TAG       IMAGE ID       ...  SIZE
...
bdm-publication-image-v7  latest    9e13327e93a5   ...  28.5GB
...
```

The Dockerfile that underlies the image can be found at `bdm-paper-examples/docker/Dockerfile`

Reproduce results

General information

- Results are always generated in the directory `bdm-paper-examples/result`. Make sure this directory is empty before starting a script.
- The resulting plots can be found in the directory `bdm-paper-examples/result/evaluation`.
- Please ensure that the system you are using has enough memory and disk space to run the simulations.
- The information regarding required memory and disk space given below only includes the execution of the script.
- Execution time on your system might differ significantly, depending on your hardware.
- Use a local filesystem to avoid authentication issues with network file systems.
- Do not run the scripts as `root` user, and do not use `sudo`.
- Only run one script at the same time.
- If you are facing issues that you cannot resolve, don't hesitate to get in touch with lukas.breitwieser@cern.ch or contact@biodynamo.org for help.

Result directory structure

As mentioned above, all results will be generated in the directory `bdm-paper-examples/result`. Please find a description of its directory structure below.

- `evaluation`: This directory contains plots generated from the individual simulation results. The plots are accompanied by a `csv` file containing the plot's raw numbers.
- `reproduce`: Contains information on how to reproduce the generated results.
  - Exact source code used: `bdm-paper-examples` and `biodynamo` repositories
  - Additional details in `metadata/run-*` about used memory, disk space, docker container id, and more.
- `metadata`:
  - `log`: Complete log of the whole execution.
  - `system-info`: Metadata about the hardware and software configuration used.
  - `benchmark`: Subdirectories contain log files of individual benchmark runs (e.g., load balancing study for the epidemiology use case)
- `tmp`: Contains large temporary files (e.g., the raw results from Intel Vtune and advisor).
- Simulation result directories (`cell-grow-divide`, `epidemiology`, `soma-clustering`, `pyramidal-cell`, `tumor-spheroid`) contain the measurements for each benchmark for each simulation. For example, the `epidemiology/operation-breakdown` directory contains the runtime measurement for each operation.
- `run-*.tar.gz`: Archive of the whole result directory excluding the `tmp` folder.
- `run-*.tar.gz.sha256`: Sha256 hash code of the result archive to detect modification.

The following instructions assume that the current working directory is `reproduce/bdm-paper-examples`.

Functional Evaluation

This script is a quick check to see if the steps so far were successful. The script compiles and executes all simulations shown in Table 1 in the paper with ~1000 agents.

- Executed on System A in 15 min
- Required memory: 12 GB
- Required disk space: 1 GB

```
# Create a new Docker container and run the script
docker/run.sh ./run-functional-evaluation.sh
```

If the script returns exit code 0, it means that the checks were successful and that we can start running the scripts to reproduce all results in the paper.

Main results (Figure 4a and 9–13 in the paper)

This script executes the majority of benchmarks described in the evaluation section of the paper. We separated some benchmarks to reduce the memory consumption. The subsequent sections show how to execute these separated benchmarks.

- Executed on System A in 12d
- Required memory: 13 GB
- Required disk space: 10 GB

```
# Move the previous result
mv result ../result-<choose-name>
```

```
# Create a new screen
screen
# Create a new Docker container and run the script
docker/run.sh ./run-main.sh
# Disconnect from the screen by pressing CTRL + a, d
```

Mapping between the output of the script and the paper:

Figure in the paper	Result file (inside `result/evaluation`)
Figure 4a	`operation-breakdown.svg`
Figure 9 (left)	`optimization-overview/all.svg`
Figure 9 (right)	`optimization-overview/all-memory.svg`
Figure 10a	`full-sim-scalability/all.svg`
Figure 10b	`optimization-overview-scalability/cell-grow-divide-runtime.svg`
Figure 10c	`optimization-overview-scalability/soma-clustering-runtime.svg`
Figure 10d	`optimization-overview-scalability/epidemiology-runtime.svg`
Figure 10e	`optimization-overview-scalability/pyramidal-cell-runtime.svg`
Figure 10f	`optimization-overview-scalability/tumor-spheroid-runtime.svg`
Figure 11a (left)	`environment/all-144.svg` [1]
Figure 11a (right)	`environment/all-18.svg` [2]
Figure 11b (left)	`environment/all-144-update-environment.svg`
Figure 11b (right)	`environment/all-18-update-environment.svg`
Figure 11c (left)	`environment/all-144-agent-ops.svg`
Figure 11c (right)	`environment/all-18-agent-ops.svg`
Figure 11d (left)	`environment/all-144-memory.svg`
Figure 11d (right)	`environment/all-18-memory.svg`
Figure 12 (left)	`load-balancing/all-144-threads.svg`
Figure 12 (right)	`load-balancing/all-18-threads.svg`
Figure 13 (left)	`mem-alloc-comp/all-total-runtime.svg`
Figure 13 (right)	`mem-alloc-comp/all-memory.svg`
Figure 14 (left)	`alternative-exec-modes/all.svg`
Figure 14 (right)	`alternative-exec-modes/all-memory.svg`

Comparison with Cortex3D and NetLogo (Figure 8 in the paper)

This script executes the comparison of BioDynaMo with Cortex3D and NetLogo.

- Executed on System A in 3h
- Required memory: 83 GB
- Required disk space: 10 GB

```
# Move the previous result
mv result ../result-<choose-name>
```

[1] 144 corresponds to the number of logical CPU core on the system.
[2] 18 corresponds to the number of physical CPU cores on the first NUMA domain.

```
# Create a new screen
screen
# Create a new Docker container and run the script
docker/run.sh ./run-comparison-with-others.sh
# Disconnect from the screen by pressing CTRL + a, d
```

Mapping between the output of the script and the paper

Figure in the paper	Result file (inside `result/evaluation`)
Figure 8	`comparison-with-others/all.svg`

Runtime and space complexity (Figure 6 in the paper)

This script analyses the runtime and memory consumption of BioDynaMo, with the number of agents increasing from 10^3 to 10^9.

- Executed on System B in 2d 9h
- Required memory: 610 GB
- Required disk space: 16 GB

```
# Move the previous result
mv result ../result-<choose-name>
# Create a new screen
screen
# Create a new Docker container and run the script
docker/run.sh ./run-runtime-complexity.sh
# Disconnect from the screen by pressing CTRL + a, d
```

Mapping between the output of the script and the paper

Figure in the paper	Result file (inside `result/evaluation`)
Figure 6 (left)	`runtime-complexity/all.svg`
Figure 6 (right)	`runtime-complexity/all-memory.svg`

Workload profiling (Figure 4b and Figure 5 in the paper)

This script performs the microarchitecture and roofline analysis for all simulations in Table 1 in the paper.

- Executed on System A in 20h
- Required memory: 47 GB
- Required disk space: 32 GB

```
# Move the previous result
mv result ../result-<choose-name>
# Create a new screen
```

```
screen
# Create a new Docker container and run the script
docker/run.sh ./run-profiling.sh
# Disconnect from the screen by pressing CTRL + a, d
```

The file `result/evaluation/uarch-analysis.svg` corresponds to Figure 4b in the paper.

Roofline plot

You have to extract the information from the individual roofline plots in `result/evaluation/roofline/*.html` and enter it in the file: `other-tools/nersc-roofline/Plotting/data.txt` to generate the aggregate plot. Here are the required steps:

1. Update the values for `memroofs`. The numbers correspond to the order of `mem_roof_names`.
2. Repeat the step for `comproofs`.
3. For each simulation, update the arithmetic intensity (AI) and GigaOPS (FLOPS) of the whole simulation indicated by the right crosshair (DRAM based AI) in `result/evaluation/roofline/*.html`.

Now, you can generate the plot based on the updated data file.

```
# Execute plot/roofline.sh in the Docker container
docker/exec.sh plot/roofline.sh
```

The generated plot can be found in `result/evaluation/roofline/roofline.svg` and corresponds to Figure 5 in the paper.

Biocellion comparison small (Figure 7 in the paper)

This script executes the small-scale comparison with Biocellion, and the optimization analysis in Figure 7b (right).

- Executed on System C in 7h
- Required memory: 11 GB
- Required disk space: 1 GB

```
# Move the previous result
mv result ../result-<choose-name>
# Create a new screen
screen
# Create a new Docker container and run the script
docker/run.sh ./run-biocellion-cmprsn-single-node.sh
# Disconnect from the screen by pressing CTRL + a, d
```

Figure in the paper	Result file (inside `result/evaluation`)
Figure 7b (right)	`optimization-overview-16-cpus/biocellion-cell-clustering.svg`

Open the spreadsheet at `bdm-paper-examples/bdm/biocellion-cell-clustering/comparison.ods`

and follow the instructions below to calculate the speedups.

- Copy the content of `biocellion-cell-clustering/single-node/runtime` into the cells `B32:B34`

Biocellion comparison large (Figure 7 in the paper)

This script executes the large-scale comparison with Biocellion, the optimization analysis in Figure 7b (left), and also renders the visualization in Figure 7a.

- Executed on System B in 16.5h
- Required memory: 500 GB
- Required disk space: 1 GB

```
# Move the previous result
mv result ../result-<choose-name>
# Create a new screen
screen
# Create a new Docker container and run the script
docker/run.sh ./run-biocellion-cmprsn-cluster.sh
# Disconnect from the screen by pressing CTRL + a, d
```

Figure in the paper	Result file (inside `result/evaluation`)
Figure 7b (left)	`optimization-overview/biocellion-cell-clustering.svg`
Figure 7a	`biocellion-cell-clustering-final-state-highres.png`

Open the spreadsheet at `bdm-paper-examples/bdm/biocellion-cell-clustering/comparison.ods` and follow the instructions below to calculate the speedups.

- Copy the content of `biocellion-cell-clustering/cluster/runtime` into the cells `B7:B9`
- Copy the content of `biocellion-cell-clustering/cluster-281m/runtime` into the cells `E7:E9`
- Copy the content of `biocellion-cell-clustering/single-node/runtime` into the cells `E32:E34`
- Enter the number of physical CPU cores of the used benchmark hardware into `B18` and `E18`