OOPSLA 2020 Artifact Evaluation

We provide a VM image that has our project ConSysT preinstalled. The VM is for executing the benchmarks that are available in our paper. As the benchmarks spawn 9 processes (each running its own JVM), we suggest that you use at least 12GB of memory to avoid out of memory errors. We tested the VM with 12GB provided memory on a Linux Mint 19 machine using Oracle VirtualBox with VT-x (processor virtualization) enabled.

The user credentials for the virtual machine are:

- Username: consyst
- Password: consyst

Import the image by starting VirtualBox and then File > Import appliance.

Executing the benchmarks

In the paper, there are benchmarks for five different case studies (see Section 6.1). In the following, we explain how to execute them on the provided VM image.

Open a terminal and navigate to the root of the code repository.

```
$ cd ~/consysT-code
```

Each case study as well as the benchmarks for the case study are located in a different folder:

- Counter in `demos/counter`
- TicketShop in `demos/concert-tickets`
- MixT Message Groups in `demos/message-groups`
- E-Commerce in `demos/eshop`
- IP Twitter Clone in `demos/twitter-clone`

Open a terminal and navigate to the directory of the case study that you want to execute. For example, for the Counter case study type:

```
$ cd demos/counter
```

You can execute the performance benchmark from the paper. For that execute the `run-artifact.sh` script located in the benchmark folder.

```
$ ./run-artifact.sh
```

This will execute the performance benchmarks for the `weak`, `strong` and `mixed` configurations. The benchmarks executes a 5 warmup and 5 measure iterations. Depending on the case study and your machine, the benchmark may run several hours. The configuration file of the benchmarks are in `src/main/resources/local`. The benchmarks use the configuration files in `weak`, `mixed` and `strong`, respectively.

The benchmark will generate some raw output in `bench-results/artifact`. However, the script also processes the raw results. After the benchmarks are executed, a browser window opens with two tabs, each showing a graph with the results of the run. One graph (y-axis labelled normalized_mean) shows the percentage of mean runtime over all runs of the `weak` and `mixed` configuration compared to the `strong` configuration (cf. Figure 7 in the paper). The other graph (y-axis labelled mean) shows the absolute mean runtime.

Troubleshooting

In case a benchmark fails to execute, you need to rerun the whole script. Before rerunning, ensure that all processes are stopped. For that, you can kill all Java processes:

```
$ pkill -f java
```