A Artifact Appendix

A.1 Abstract

We consider our paper's artifact to be the set of benchmarks we used in the paper, as well as the results we got by running particular versions of model checking tools (GenMC, HMC, Nidhugg, Herd, rmem, Dartagnan) on the benchmarks set. We do *not* consider the artifact of the paper to be HMC, as it will evolve over time, and the results obtained by running the same benchmarks may differ in the future.

That said, we have made HMC publicly available on GitHub, as part of the GenMC tool (https://github.com/MPI-SWS/genmc). For any bugs, comments, or feedback regarding or HMC, please do not hesitate to contact us.

A.2 Artifact check-list (meta-information)

- **Algorithm:** HMC.
- **Program:** C benchmarks publicly available at GenMC's repository.
- **Run-time environment:** VirtualBox.
- **Output:** Console.
- **Experiments:** Scripts that fully reproduce the paper's results are provided.
- **How much disk space required (approximately)?:** After unpacking the VM image, approximately 10 GB.
- **How much time is needed to prepare workflow (approximately)?:** Everything is already set up.
- **How much time is needed to complete experiments (approximately)?:** <24 hours (see Section A.6).
- **Publicly available?:** Yes.
- **Code licenses (if publicly available)?:** See the pages of the tools for the respective licenses. For all code not explicitly covered by these licenses, GPLv2 applies.
- **Data licenses (if publicly available)?:** GPLv2.
- **Archived (provide DOI)?:** 10.5281/zenodo.3562082

A.3 Description

A.3.1 How delivered

The artifact is available on Zenodo (https://doi.org/10.5281/zenodo.3562082) and consists of a Virtual Machine (VM) containing binaries for all the model checking tools we used, along with all the benchmarks used in the submitted version of our paper, and HMC's source code. These hopefully suffice to validate the claims made in the paper.

A.3.2 Hardware dependencies

None in particular; allocating at least 2GB of RAM for the VM is recommended but not strictly required. Depending on the VirtualBox version there might be restrictions on the CPU (see VirtualBox's manual; https://download.virtualbox.org/virtualbox/6.0.14/UserManual.pdf).

A.3.3 Software dependencies

An operating system in which VirtualBox can be installed (see VirtualBox's manual; https://download.virtualbox.org/virtualbox/6.0.14/UserManual.pdf).

A.4 Installation

1. Download and install VirtualBox (https://www.virtualbox.org/wiki/Downloads), in case it is not already installed. We have tested the VM with VirtualBox 6.0.14 under Debian GNU/Linux.
2. Open VirtualBox, and import our VM by clicking "File" and then "Import Appliance".
3. After starting the VM, you can log in with the username "user" and password "hmc". Once logged in, shortcuts to the terminal and the file manager can be found under "Activities".

A.5 Experiment workflow

The results of the paper are reproduced using some bash scripts which print out some tables corresponding to the ones in our paper.

A.6 Evaluation and expected result

For the following sections we assume that the working directory is `~/asplos20-benchmarks`.

For all tables, we use a clock symbol to denote a timeout, and an X symbol to denote a failure of some sort. As in the paper, the timeout limit is set to 30 minutes. Note that, depending on the RAM the VM is allocated, some entries where a timeout is expected might be shown as failed instead (e.g., for Table 2). This is usually due to a stack overflow, and can happen for either Herd or rmem.

A.6.1 Reproducing Table 1 (~ 5.3 hours)

To reproduce the results of Table 1, please issue the following command:

```
./get-table1.sh
```

A.6.2 Reproducing Table 2 (~ 12.5 hours)

To reproduce the results of Table 2, please issue the following command:

```
./get-table2.sh
```

A.6.3 Reproducing Table 3 (~ 1.5 hours)

Similarly, to reproduce the results from Table 3 issue:

```
./get-table3.sh
```

A.6.4 Reproducing Table 4 (~ 1 hour)

To reproduce the results of Table 4 issue:

```
./get-table4.sh
```

A.6.5 Reproducing Table 5 (~ 30 minutes)

To reproduce the results of Table 5 issue:

```
./get-table5.sh
```

A.6.6 Reproducing the overhead measurements

For this particular subsection, we assume that the working directory is `~/asplos20-benchmarks/plots`.

All the data we used for the scatter diagram and the overhead measurements can be found at the `scatter.dat` file. These are obtained from GenMC's standard test suite results, as well as from the benchmarks used in this paper.

To get a PDF file containing the scatter diagram please issue:

```
latexmk -pdf main.tex && evince main.pdf
```

To get the geometric means of Figure 2, please issue the following commands:

```
./geo-mean.awk scatter.dat 3 7 0.1
./geo-mean.awk scatter.dat 7 5 0.1
./geo-mean.awk scatter.dat 3 5 0.1
```

By invoking `max-overhead.awk` instead of `geo-mean.awk` you can get the maximum overheads instead.

Both these scripts as arguments the name of the data file, the column over which we are normalizing, the overhead column, and the threshold below which we do not consider entries.

A.7 Experiment customization

A.7.1 Running HMC/GenMC

A generic invocation of GenMC looks like the following:

```
genmc [OPTIONS] -- [CFLAGS] <file>
```

Where `CFLAGS` are options that will be passed directly to the C compiler, and `OPTIONS` include several options that can be passed to GenMC. Among these options, the most useful ones are probably the `-unroll=N` switch, which unrolls a loop `N` times, and the `-wb` and `-mo` options, that enable the WB and the MO variant of GenMC, respectively (default is WB). Lastly, `file` should be a C file that uses `pthreads` for concurrency.

To use HMC, please invoke GenMC with the `-imm` option. More information regarding the usage of the tool can be found at the tool's manual (https://github.com/MPI-SWS/genmc/tree/master/doc).

A.7.2 Available benchmarks

The benchmarks we used for the tables of our paper are located in the directory `~/asplos20-benchmarks/benchmarks`. Apart from the benchmarks located in the folder above, many more benchmarks can be found at GenMC's repository (https://github.com/MPI-SWS/genmc/tree/master/tests).

In the above repository and the relevant sub-directories, there is a separate folder for each benchmark, that contains the "core" of the test case, as well as the expected results for the test case, some arguments necessary for the test case to run, etc. In order to actually run a test case, we can run the tool with one of the test case variants, which are located in a folder named 'variants', in turn located within the respective test case's folder.

For example, assuming that GenMC's repository has been cloned at REPO, to run a simple Store Buffering (SB) test case with HMC, please issue:

```
genmc -imm REPO/tests/correct/synthetic/SB/variants/sb0.c
```