A Artifact Appendix

A.1 Abstract

Our artifact can be downloaded through Github or ACM DL, and it provides the source code of our QIThread library, as well as scripts to download all the benchmarks used in our paper, automatically build them, and generate time data from these benchmarks using our library. Our installation instructions are tested on a Ubuntu Linux 16.04 machine with an x86_64 architecture, and our experiments are conducted on a machine having two Xeon E5-2670 processors with a total of 32 logical processors and 128GB memory. While our artifact should be runnable on any multi-core machine, some benchmarks are configured to create 24 threads in our experiments. All benchmarks and dataset needed for evaluation can be downloaded using our scripts or are already included in our artifact.

A.2 Artifact Check-List (Meta-Information)

- **Algorithm:** Semantic-aware scheduling algorithms for a synchronization-determinism runtime.
- **Program:** SPLASH-2x, NPB 3.3.1, PARSEC 2.0, Phoenix 2, pbzip2, aget, pfscan, bdb_bench3n, openldap, mplayer men-coder, redis, ImageMagick, and STL libstdc++-v3. Scripts are included to automatically download and build them.
- **Compilation:** GCC 5.4 and GCC 4.7 installed from Ubuntu repository. Instructions are provided to install them.
- **Data set:** Included or downloaded with scripts provided.
- **Run-time environment:** Tested on Ubuntu Linux 16.04 x86_64. Instructions are provided to install all the necessary libraries and tools from a fresh Ubuntu installation.
- **Hardware:** Some benchmarks are configured to create 24 threads in our experiments, so we recommend a configuration with at least 24 logical processors.
- **Metrics:** Overheads against nondeterministic executions are reported for all the benchmarks with all scheduling policies turned on.
- **Output:** The metrics are written to csv files in a folder created for each evaluation script execution.
- **Experiments:** A script to evaluate everything is provided. Configuration files and evaluation scripts for evaluating each individual benchmark suite are also included.
- **How much disk space required (approximately)?:** About 40 GB.
- **How much time is needed to complete experiments (approximately)?:** For our machine, it takes about 70 hours to run all the benchmarks with the default configuration.
- **Publicly available?:** Yes, it's available on both GitHub and ACM DL.

A.3 Description

A.3.1 How Delivered

Our artifact is available on Github:
<https://github.com/chyiz/QiThread>
and ACM DL:
<https://doi.org/10.1145/3300171>

A.3.2 Hardware Dependencies

We conducted our experiments on a machine with two Xeon E5-2670 processors and 128GB memory. While our code should be runnable on any multi-core machines, we recommend a configuration with more than 24 logical processors, since benchmarks are sometimes configured to create 24 threads in our experiments.

A.3.3 Software Dependencies

Our build scripts assume a Ubuntu Linux 16.04 installation on an x86_64 platform. For other Linux distributions, you may use the provided Dockerfile to create a Docker image. You need to have Docker installed to use the Dockerfile.

Software packages required to build the library and benchmarks will be installed automatically by the build scripts. No other software is needed to build or run the experiments.

A.4 Installation

Clone our repository from Github:

```
git clone https://github.com/chyiz/QiThread.git
```

The easiest way to use our artifact is to create a Docker image from the Dockerfile provided in the repository. In the project root folder, run:

```
sudo docker build -t qithread .
```

After a while, a Docker image will be available with tag qithread. You may run it with:

```
sudo docker run -it qithread
```

You will then be able to run all the experiments inside the Docker image.

Alternatively, you can compile our library and all the benchmarks directly. However, our build scripts assume an x86_64 Ubuntu 16.04 environment. For other Linux distributions, you may need to make some necessary changes or just use the Dockerfile.

Please see more detailed installation instructions in the README file provided in the artifact.

A.5 Experiment Workflow

To run the experiments, go to the eval folder and execute:

```
./eval_policy.py all-default-config.cfg
```

The evaluation results will be saved to a folder named all-default-config<Execution_Date_and_Time>.

A symbolic link current is also modified to always point to the latest evaluation results.

To extract all results into one file, run:

```
./get_all_results.sh current/ > results.csv
```

The results are then stored into a comma separated values (csv) file that can be imported to a spreadsheet.

At last, to generate the graphs used in our paper, open generate-figure.ipynb in the eval folder with Jupyter Notebook and run all cells. It will read the results.csv file in the same folder by default.

A.6 Evaluation and Expected Result

The results contain all 108 benchmarks evaluated, and each of them reports the average overhead and multiple execution times when running with different configurations. Each benchmark will have four or five rows in the final results: one non-det row representing nondeterministic execution as the baseline, one no-hint row for round-robin scheduling with no performance hints, one hinted row for round-robin scheduling with all performance hints, one all-policies row representing the Q_ITHREAD results with all policies on, and some benchmarks have one no-pcs-hint row for round-robin scheduling with only soft barrier hints but no performance critical section hints.

Given the hardware differences, it may be difficult to reproduce the exact same numbers as in our paper, but we expect the trend to be the same.

A.7 Experiment Customization

The experiment parameters can be adjusted by modifying the evaluation configuration files (.cfg in the eval folder). Several sample configuration files are provided in the artifact. For example, all-default-config.cfg runs all the benchmarks under (1) round-robin scheduling with or without performance hints and (2) Q_ITHREAD with all policies enabled, and all-compare-policies.cfg additionally tries other different combinations of policies. There are also configuration files, like phoenix-compare-policies.cfg, that only execute one benchmark suite.

You may start by copying a segment from one of the sample configuration files and modify some of the parameters. Some interesting ones are listed below:

- REPEATS: number of tests to run for a benchmark. Execution times will be averaged to get an overhead.
- INPUTS: command line parameters for a benchmark.
- EXPORT: environment variables for a benchmark.
- INIT_ENV_CMD: commands to initialize the environment before running a benchmark.
- TARBALL or GZIP: data files that need to be extracted before running a benchmark.
- REQUIRED_FILES: files that need to be copied to the evaluation folder before running a benchmark. This can be configuration files and data files that do not require decompression.
- RUN_CONFIGS: a list of scheduling policies to try.

A.8 Methodology

Submission, reviewing, and badging methodology:

- <http://cTuning.org/ae/submission-20180713.html>
- <http://cTuning.org/ae/reviewing-20180713.html>
- <https://www.acm.org/publications/policies/artifact-review-badging>