BoostFSM Evaluation

We consider the artifact to be the framework BoostFSM and the benchmarks we used in the paper. The results got from this artifact evaluation can prove that the proposed BoostFSM enables scalable FSM parallelization.

A zip file containing the basic version of our implementations of BoostFSM and some real-world FSMs as well as input data has been provided for download and evaluation. We have also provided a VM image which has already built up the necessary software environment. Because we need to use a KNL architecture with 64 cores for performance measurement, i.e., Table 2, part of Figure 4, Figure 16 and Figure 17 in the paper, users are encouraged to contact authors for remote access (to the target machine; and **If you are the conference AE reviewers, please contact the AEC chair**).

In this artifact, we can reproduce the key experimental results shown in the paper (and we want to keep the total evaluation time under 4 hours and our framework evolves over time). This hopefully suffices to validate the claims made in the paper. For any bugs, comments, or feedback, please do not hesitate to contact authors.

Artifact Overview

In general, We provide the following items in our artifact:

- The source codes of BoostFSM;
- The FSM benchmarks evaluated and their corresponding input data;
- A number of scripts for reproducing the major results reported in the paper;

More specifically, the main directory `ASPLOS21_AE` (If you choose to open the VM image on VirtualBox, it is on the desktop; we do provide a VM image, please contact the authors for more details) contains the following:

- `scripts/` is a directory with all the scripts needed to run the evaluations mentioned in the paper;
- `data.zip` is a file holding the corresponding input data;
- `src/`, `include/` and `test/` are directories where the source codes of BoostFSM are;
- `benchmarks` is a directory with the 16 FSMs shown in the paper;
- three one-step evaluation scripts `run.sh`, `run_fast.sh` and `run_perf.sh`.

Getting Started

To verify the state of the artifact, the first step is checking whether the requirements of software environments are met (e.g., centos 7, gcc 4.8.5, enabling bash scripts, cmake 2.8 and boost 1.66.0), but if you are using the VM image or accessing our machine remotely, you don't need to worry about this. We do not provide the way to reproduce Figure 17, because Figure 17 can be easily drawn by repeating the evaluations of Table 2 over input with different sizes (due to the space limitations, we just provide the Medium size input shown in Figure 17).

We provide a script for fast checking. It will generate part of results for Table 1-5 and Figure 16 in the paper, respectively. Though those results may be quite different from what the paper shown, this script in fact can help the users to know whether everything works well in a relatively short time (maybe 30 minutes). **But you can skip this one if you don't think you need a fast check.**

```
$ bash run_fast.sh
```

The major evaluation should be done by using the script `run.sh`, which will generate the time-irrelevant results (i.e., Table 1, 3, 4 and 5) in just one step/command. All results generated from this scripts should be in the reasonable errors ranges, compared with the ones claimed in the paper (more details can be found in the following sections).

```
$ bash run.sh
```

We also provide a script about performance measurement (i.e., time measurement). If both the hardware requirements (a Xeon Phi or similar platforms) and the software requirements can be met, we can use the following script and command to reproduce Table 2, part of Table 4 and Figure 16 in the paper.

```
$ bash run_perf.sh
```

Step-by-Step Instructions for Evaluation

We have provided a script `run.sh` to generate all the time-irrelevant results in one step (as mentioned above), but we also enable very flexible evaluations or manual testing. The total evaluation time of this artifact should be about 4 hours (which depends on the underlying machines) and the total size of this artifact should be 20GB (including the data but excluding the VM image).

Compilation

Under the directory `ASPLOS21_AE`, there is a script `compile.sh`. By executing the following command, we will generate all the executables, and two new folder `bin` and `build` under `ASPLOS21_AE`. And the executables are also copied into `scripts/` for the next steps. Here, we need to ensure software dependencies are met before executing the command.

```
$ bash compile.sh
```

Reproducing Results Separately

After generating the executables, we can break down the executions in `run.sh` into the following parts, and take a deeper look into our artifact. The detailed executions of each part can be checked in the corresponding scripts.

Get Table 1

The results in Table 1 are the collected FSM properties. We just need to use part of the inputs to finish this profiling. By run the following commands, we can get the key results in Table 1.

```
$ cd ./scripts
$ bash GetTable1.sh InputConf_five.in 64
```

Note that, as introduced in the paper, the input data used for profiling should be randomly selected, here we should allow some reasonable differences between the results got and the one shown in the paper. The output results of the above evaluations should be

```
------ Generate the major components in Table 1------
FSM: conv(10^3)   conv(10^6)   SpecAcc
M1   1/2   1/1.99048   0
M2   1/7.93016   1/1   0.0539683
M3   1/1.99683   1/1.70794   0.660317
M4   1/5   1/5   0
M5   1/6.25079   1/1   0.0666667
M6   1/8.40317   1/1   0.031746
M7   1/4.15873   1/1   0.107937
M8   1/2   1/2   1
M9   1/5   1/5   0
M10   1/53.9524   1/20.9841   0.0031746
M11   1/2   1/2   0
M12   1/2   1/2   0
M13   1/2   1/2   0
M14   1/2   1/2   0.31746
M15   1/2   1/2   0
M16   1/1   1/1   1
```

Get Table 3 and 4

The results in Table 3 and 4 are the statistics of Path Fusion. There are two part, for Static Path Fusion and Dynamic Path Fusion, respectively. By run the following commands, we can get the key results in Table 1 (Assume originally you are in `ASPLOS21_AE/`).

```
$ cd ./scripts
$ bash GetTable3a.sh
$ bash GetTable4.sh InputConf.in 64
```

But if the evaluations are executed with using the VM image, please using the following commands (assume the number of available cores is k in your VM setup, which we prefer $k = 64$)

```
$ cd ./scripts
$ bash GetTable3a.sh
$ bash GetTable4.sh InputConf.in k
```

Note that, if the hardware dependencies are not met, the time measured will be quite different from the ones in Table 3 (This execution time will be expected to be 1.5 to 2 hrs). The output results of the above evaluations should be

```
- - - - - - Generate the major components in Table 3a- - - - - -
FSM:  FS  Time
M1  173  0.0568 seconds.
M3  2876  1.22554 seconds.
M4  486  0.219338 seconds.
M8  6655  4.73284 seconds.
M11  19899  36.2001 seconds.
```

and

```
- - - - - - Generate the major components in Table 4- - - - - -
FSM:  VecS  Nuniq  Nfused
M1  1.99444  1132.77  7.42937
M2  1.43016  17128.7  130.918
M3  1.92778  1311.37  11.3365
M4  5 961.351  4.98968
M5  1.2246  10981.3  148.506
M6  1.36349  21121.4  246.257
M7  1.05079  829.622  56.8913
M8  2  896.775  4.7746
M9  5  1339.44  10.3516
M10  11.8349  10464.6  115.902
M11  2  1222.8  13.4976
M12  2  161796  1208.53
M13  2  640.504  3.74921
M14  2  1245.24  7.8881
M15  2  1024.94  6.49444
```

Get Table 5

The results in Table 5 are the speculation accuracy. By running the following commands, we can get the results in Table 5 (Assume originally you are in `ASPLOS21_AE/`, and this execution time will be expected to be 1 to 2 hrs).

```
$ cd ./scripts
$ bash GetTable5.sh InputConf.in 64
```

The output results of the above evaluations should be

```
- - - - - - Generate the major components in Table 5- - - - - -
FSM:   B–Spec   ||   #Iteration   |   Iteration...
M1   0.607143   ||   1.85   |   0.607143   1.00
M2   0.047619   ||   2   |   0.047619   1.00
M3   0.0015873   ||   2   |   0.0015873   1.00
M4   0   ||   2.4   |   0   0.980159   1.00
M5   0.05   ||   2   |   0.05   1.00
M6   0.0460317   ||   2   |   0.0460317   1.00
M7   0.0865079   ||   2   |   0.0865079   1.00
M8   1   ||   1   |   1.00
M9   0   ||   3   |   0   0.015873   1.00
M10   0.616667   ||   2.6   |   0.616667   0.973138   1.00
M11   0   ||   2   |   0   1.00
M12   0.0246032   ||   3   |   0.0246032   0.569841   1.00
M13   0   ||   2   |   0   1.00
M14   0.334127   ||   3   |   0.334127   0.572222   1.00
M15   0   ||   2.05   |   0   0.984127   1.00
M16   1   ||   1   |   1.00
```

Time Measurements (Table 2, part of Table 4 and Figure 16)

If the hardware dependencies are met, users can reproduce the results shown in Table 2, part of Table 4 and Figure 16 in the paper, by executing the following commands. (Again, assume originally you are in `ASPLOS21_AE/`)

```
$ cd ./scripts
$ bash GetTable2.sh InputConf.in 64
$ bash GetTable4_x1.sh InputConf.in 64
$ bash GetTable4_x2.sh InputConf.in 64
$ bash GetFigure16.sh InputConf.in 64
```

The output results of the above evaluations should be

```
Now Try Table 2 and Figure 16 (Please check the Hardware dependencies)
- - - - - - Generate the major components in Table 2- - - - - -
FSM:   Seq(s)   B-Enum,   B-Spec,   S-Fusion,   D-Fusion,   H-Spec
...
M6    9.47893     36.8969     28.3816     /     13.4187     40.0112
...

- - - - - - Generate the major components in Figure 16(a)- - - - - -
B-Enum     D-Fusion     B-Spec     H-Spec     S-Fusion
1    1    1    1    1
0.569912     1.00609     1.8835     1.88386     1.31313
1.09578     1.92198     1.74094     1.49052     2.56572
...
```

The differences between the evaluation results and the ones shown in paper should be less than 5%.

More Flexible Evaluation

This artifact can be also used in other FSM benchmarks and input data. Please follow the format shown in `benchmarks/`, and the commands in the scripts to evaluated other FSMs.