

Artifact Description: A Compiler Framework for Fixed-topology Non-deterministic Finite Automata on SIMD Platforms

ABSTRACT

To allow reproduction of the experiments conducted in this work, we include the NFA traversal source code and the information on launching the experiments on GPU for both SIMD_NFA and the baseline implementation. We first show how to build and run the traversal code and then we cover the kernel configurations and computations used for collecting the reported results, such as the optimal number of input streams and throughput.

1 Description

The source code includes our proposed SIMD_NFA scheme (generated by the compiler) and a variant of the well-known iNFAnt scheme, called iNFAnt+. These sources are publicly available at the following URL:

https://github.com/MarziyehNourian/SIMD_NFA_Traversal

- **Algorithm:** SIMD_NFA and extended iNFAnt
- **Program:** CUDA code
- **Compilation:** NVIDIA nvcc (v8.0.44) and GNU gcc (v4.8.5) with Makefile provided.
- **Binary:** CUDA executable called “regex_gpu”
- **Data set:** Publicly available Fix-topology NFA datasets transformed by the compiler
- **Run-time environment:** REHL 7.5 OS, kernel v3.10 with CUDA 9 toolchain
- **Hardware:** CUDA enabled GPU devices: Tesla K40c, TITAN X and TITAN XP
- **Execution:** scripts and instructions provided
- **Output:** timing data and the information about the states matched during the traversal
- **Experiment workflow:** Download the code from the repository; Run the provided scripts from the main code directory; Check the redirected output files for the match information and the output spreadsheet for the timing results.
- **Publicly available:** Yes

Hardware dependencies- CUDA enabled GPU with compute capability 3.0 or more is required.

Software dependencies- NVCC CUDA compiler, CUDA GPU driver, CUDA Toolchain and libxml2, boost and PCRE libraries are required.

Datasets- NFA datasets: Tested Fixed-topology NFA datasets are represented by memory-layout files and traversal kernels generated by the compiler. Memory-layout files contain transition symbols of 2048 patterns as well as the input stream configuration. Default input stream configuration in the memory-layout files is related to TITAN X device. It is possible to manually define the input stream configuration via the execution scripts.

Input streams: For LD and HD NFAs, we use a trace file containing different gene sequences to build 15KB input streams. For other datasets, we use the 15KB trace files with 50 percent probability of moving deeper into the NFA structure.

Experiment workflow- After cloning the source files from the GitHub repository we recommend to use the scripts in order to compile and execute the code. Compiler directives are used to compile different SIMD_NFA traversal codes relevant to the NFA topology of each dataset and the scripts contain these directives. The following instructions can be used for obtaining and executing the code.

```
$ git clone https://github.com/MarziyehNourian/SIMD_NFA_Traversal.git
$ cd SIMD_NFA_Traversal/Base_Code
$ ./scripts/SIMD_NFA_traversal.sh
$ ./scripts/infant_traversal.sh
```

By calling the program without arguments the usage information is printed. The scripts contain the execution commands for all datasets. The number of thread-blocks to launch is equal to the product of the number of required blocks to cover all patterns by the input stream count. For iNFAnt+, we found 256 to be the optimal number of threads per block and for SIMD_NFA, thread-block size is equal to the warp size.

Input streams related computations- We determine the number of thread-blocks to be run to ensure full occupancy of the target device. The maximum number of thread-blocks that can be resident (i.e., execute in parallel) on a streaming multiprocessor (SM) depends on their shared memory utilization. We allocate two state vectors (i.e., $current_{sv}$, $future_{sv}$) in the shared memory. The shared memory utilization is set to $2 \times SIMD_width \times batch_size \times NFA_size$ bits for SIMD_NFA and $2 \times Max_states_per_block$ bits for iNFAnt+. The maximum number of thread-blocks that can run concurrently on a SM ($max_resident_blocks$) is equal to $\left\lfloor \frac{shared_mem_capacity}{shared_mem_utilization} \right\rfloor$. In order to fully utilize the device, we must process $\left\lceil \frac{max_resident_blocks \times SM_count}{req_simd_units} \right\rceil$ input streams, where SM_count is device dependent and req_simd_units is the number of blocks required to fit all the patterns.

Throughput related computations- We define the traversal throughput as $\frac{\# of input_streams \times input_stream_size}{traversal_time}$. The traversal time includes the time to transfer the information required by the traversal from host to device, the kernel execution time and the time required to copy the results back to the host.

For up-to-date information regarding the source code please check the GitHub repository.