Dear reviewers:

Really appreciate your time to review our Chipmunk work. Hopefully this process can also bring you some enjoyment when playing around our software.

At a really high level, we want to separate the artifact evaluation into three steps:

1. Run Domino compiler for all benchmarks' mutations (10 for each)
2. Run Chipmunk on all benchmarks' mutations using Banzai ALU
3. Run Chipmunk on all benchmarks themselves using Tofino ALU and test the result by running P4 program in Tofino switch

The whole process may take around several hours to finish and we hope you can get some interesting experience from this artifact evaluation process.

The following is about the details to reproduce our result reported in paper, the same content will be also in the README file from different repos. In order to make life easier, most experiments can be finished within one script, but if you want to know more details, please take a look at my script or email me. Before doing any concrete work, I want to report the machine we used to run our experiments. We use NYU crunchy machines (https://www.courant.nyu.edu/webapps/content/systems/resources/computeservers) with Four AMD Opteron 6272 (2.1 GHz) (64 cores) 256 GB memory and CentOS 7 operating system. But the experiment will also work for the normal linux system **Ubuntu 16.04.6 LTS (GNU/Linux 4.15.0-43-generic x86_64)**. The mac system may cause the result to be inconsistent because of some randomness from program synthesis and mutation generation even if we fix the random seed.

Dependencies : Important

1) **Python 3.6.3**
   https://www.python.org/downloads/release/python-363/
2) **Java**
3) **Antlr 4.7.2**
   https://www.antlr.org/download/antlr-4.7.2-complete.jar
   Installation guide available at Here
4) **Sketch**
   https://people.csail.mit.edu/asolar/sketch-1.7.5.tar.gz
   Installation guide available inside extracted sketch-1.7.5 directory in README. As directed in README, Note: sketch needs to be added to PATH
5) **Clang+llvm**
   https://releases.llvm.org/5.0.2/clang+llvm-5.0.2-x86_64-linux-gnu-ubuntu-16.04.tar.xz
   Extract it, its path will be used later while compiling domino
6) **Banzai**
   https://github.com/packet-transactions/banzai

To install, run: ./autogen.sh && ./configure && sudo make install (Refer README)

7) **Other library dependencies**
   git g++ (>=7.4) build-essential autotools-dev libncurses5-dev autoconf libtool and zlib1g-dev automake (Use a package manager like macports or apt-get to get them.)

8) **Barefoot SDE 8.2.0 (Tested on this version)**

Chipmunk Source Code:

Note1: Please download all the following repositories into one folder

1. Download **chipmunk_experiments-tofino repo** and follow the README file inside to install
   git clone https://github.com/chipmunk-project/chipmunk_experiments-tofino

2. Download **domino-compiler repo** and follow the README file inside to install
   git clone https://github.com/chipmunk-project/domino-compiler

3. Download **domino-examples repo** and follow the README file inside to install
   git clone https://github.com/chipmunk-project/domino-examples

4. Download **chipmunk-tofino repo** and follow the README file inside to install
   git clone https://github.com/chipmunk-project/chipmunk-tofino

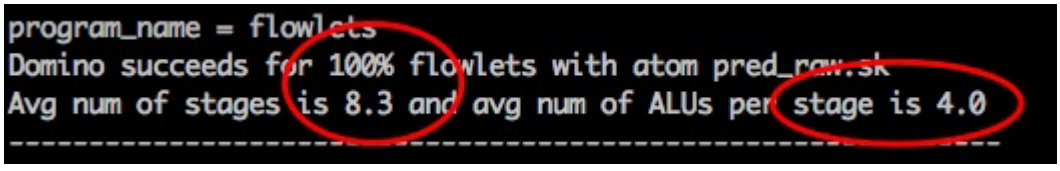
5. Download tofino-boilerplate repo inside the tofino switch
   https://github.com/chipmunk-project/tofino-boilerplate

Part 1: Run Domino compiler for all benchmark's mutations:

1. Go to domino-examples folder
2. Make sure sketch is added to PATH
3. Run the script

```
python3 run_all_domino_examples.py
```

4. Verify the resource usage and successful compilation rate



Part 2: Run Chipmunk on all benchmarks' mutations using Banzai ALU

Because we use 64-cores powerful machines to run all the benchmarks, if you do not have access to those machines, we provide an option to run simpler benchmarks that may finish compilation quickly.

Full list: ['learn_filter.c', 'blue_increase.c', 'blue_decrease.c', 'stateful_fw.c', 'dns_ttl_change.c', 'flowlets.c', 'rcp.c', 'marple_new_flow.c', 'marple_tcp_nmo.c', 'sampling.c', 'stfq.c', 'conga.c', 'snap_heavy_hitter.c', 'spam_detection.c']

Simple list: ['rcp.c', 'marple_new_flow.c', 'marple_tcp_nmo.c', 'sampling.c', 'stfq.c', 'conga.c', 'snap_heavy_hitter.c', 'spam_detection.c']

1. Go to chipmunk_experiments-tofino folder
2. Run the script
   2.1 part of simple example

```
python3 run_expr.py simple_part
```

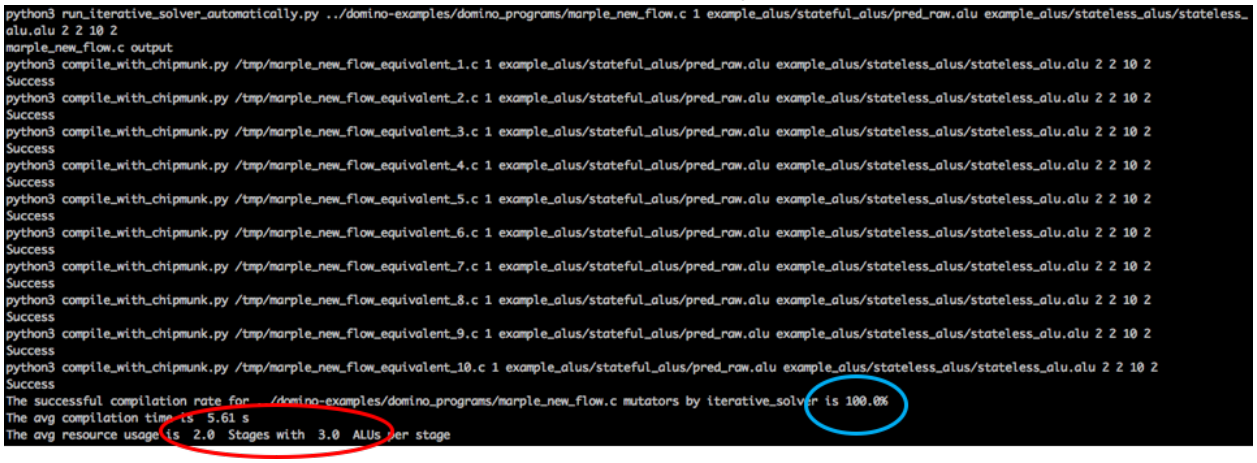
   2.2 all of simple examples (optional)

```
Python3 run_expr.py simple
```

   2.3 all of complex examples (optional) **take a really really long time**

```
python3 run_expr.py complex
```

3. **Verify the resources usage, compilation rate and compilation time.** Because we use parallel mode from SKETCH, so the compilation time may be varied.



Part 3: Run Chipmunk on all benchmarks themselves using Tofino ALU and test the result by running P4 program in Tofino switch

NOTE 1: Because we need to generate the program and verify it one by one, so this may need a little bit manual work
NOTE 2: Based on our opinion, Tofino compiler may have a potential bug because it fails to compile P4 program with code like "((((0))) - x", so we may have to replace it by "- x".

Program list: ['blue_increase.c', 'blue_decrease.c', 'dns_ttl_change.c', 'flowlets.c', 'rcp.c', 'marple_new_flow.c', 'marple_tcp_nmo.c', 'sampling.c', 'conga.c', 'snap_heavy_hitter.c']

1. Go to chipmunk_experiments-tofino folder
2. Run the script for each program individually, and **follow the command line instructions** provided to copy the p4 program to the switch.
   Important 1: You may need to remove ((((0))) from the generated p4 program in case the output prompts.
   Important 2: Refer to step 3 to get the input values needed to run the generated p4 program in tofino.

```
python3 compile_with_tofino.py ../domino-examples/domino_programs/sampling.c 1 3 3 10 2
```

```
python3 compile_with_tofino.py ../domino-examples/domino_programs/marple_tcp_nmo.c 1 3 2 10 2
```

```
python3 compile_with_tofino.py ../domino-examples/domino_programs/rcp.c 1 1 6 10 2
```

```
python3 compile_with_tofino.py ../domino-examples/domino_programs/marple_new_flow.c 1 2 3 10 2
```

```
python3 compile_with_tofino.py ../domino-examples/domino_programs/conga.c 2 1 3 10 2
```

```
python3 compile_with_tofino.py ../domino-examples/domino_programs/snap_heavy_hitter.c 2 1 2 10 2
```

```
python3 compile_with_tofino.py ../domino-examples/domino_programs/blue_increase.c 1 4 2 10 2
```

```
python3 compile_with_tofino.py ../domino-examples/domino_programs/blue_decrease.c 1 4 2 10 2
```

```
python3 compile_with_tofino.py ../domino-examples/domino_programs/dns_ttl_change.c 1 3 6 10 2
```

```
python3 compile_with_tofino.py ../domino-examples/domino_programs/flowlets.c 1 3 5 10 2
```

3. For any benchmark, get the value by directly running C program in **domino-example** folder

```
cd domino_in_c_lan

g++ <.c>

./a.out
```

4. To run the program in tofino, below are the commands inside the tofino switch :

```
cd $SDE
```

```
./p4_build.sh /tmp/autogen.p4
```

```
cd ~/tofino-boilerplate/CP
```

```
./run.sh + feeding the initial value
```

Usage is below :

```
./run.sh field0 field1 field2 field3 field4
[reg_0_register_value_f0=x1 reg_0_register_value_f1=x2
reg_1_register_value_f0=y1 reg_1_register_value_f1=y2
reg_2_register_value_f0=z1 reg_2_register_value_f1=z2]
```

If you have any questions, feel free to ping me by xg673@nyu.edu.
Thanks again for your time.

All members of the Chipmunk research group