Artifact Evaluation Instructions

Welcome to the "Collaborative zkSNARKs" artifact. The artifact is a distributed protocol; it requires many machines to evaluate. Our experimental infrastructure uses GCP, so we're going to give you access to a "coordinator" machine that is logged in with GCP.

Please follow the directions below. The coordinator is a real machine (not a VM), is used by us regularly, and is logged into our GCP account.

At the end of this document we provide instructions for using a local VM to replicate the experiments that can be run locally.

Connect to the coordinator

1. Give us your public key using HotCRP.
2. Wait for us to confirm that we have granted that key access.
3. `ssh aeval@128.12.176.8`

Build the collaborative proofs

1. `cd ~/multiprover-snark/mpc-snarks`
2. `git clean -fd`
3. Check that `git rev-parse HEAD` outputs 98cc63c7b885ade04989a5505050504ae7f2aac0.
4. `cargo build --release`

- You can `cargo clean` first to force a clean build.

5. Optional: run the test suite `./test.zsh`

- If it exits with a zero return code, it was successful.

Collect the data

1. Run all experiments with `time ./analysis/collect/artifact_eval.zsh`

- This should take approximately 24 minutes.
- Alternatively: you can run the experiments one-by-one:
  1. `time ./analysis/collect/bad_net.zsh | tee ./analysis/data/bad_net.csv`
  – This runs locally and should take approximately 6 minutes
  2. `time ./analysis/collect/weak_machines.zsh`
  – This runs on GCP and should take approximately 10 minutes
  3. `time ./analysis/collect/Npc.zsh`
  – This runs on GCP should take approximately 8 minutes

Make & inspect the plots

1. Generate all plots: `./analysis/plotting/artifact_eval.zsh`

2. Copy plots to your machine: `scp 'aeval@128.12.176.8:multiprover-snark/mpc-snarks/analysis/pl .`
3. Analyze:
4. Varying constraint counts: mpc.pdf should be comparable to Figure 8
5. Varying prover count: Npc.pdf should be comparable to Figure 9
6. Varying link capacity: bad_net.pdf should be comparable to Figure 10

Optional: reproduce the local experiments using a VM

Ubuntu machine setup

(You can skip this, the VM is already set up. We include it so you know how that machine was set up)

1. New machine, at least 8GB RAM, 10GB disk

- Ubuntu 20.04 server

2. Do Ubuntu installation

- username: user password: user
- updating took a while

3. `apt install zsh libgmp-dev neovim autoconf pkg-config libtool apache2-dev apache2 dnsmasq-base protobuf-compiler libprotobuf-dev libssl-dev libxcb-present-dev libcairo2-dev libpango1.0-dev tmux units r-base virutalbox-guest-utils`
4. `curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh`

- nightly

5. `cargo install ripgrep`
6. Install the mahimahi shell network emulator

- clone it
- apply patches
  - empty PICKY_CXXFLAGS in configure.ac (compiler is pickier now)
  - add mm-rate-to-events to install list in scripts/Makefile.am (need this)
- `./autogen.sh && ./configure && make -j 8`
- `sudo sysctl -w net.ipv4.ip_forward=1`

7. Install R libraries: ggplot2, dplyr, readr, scales
8. Set up folder sharing `sudo adduser user vboxsf && sudo systemctl enable virutalbox-guest-utils.service`

Build the collaborative proofs

Download the VM here: https://doi.org/10.5281/zenodo.5889564.

1. `cd ~`
2. `git clone -b artiface-eval https://github.com/alex-ozdemir/multiprover-snark`
3. `cd multiprover-snark/mpc-snarks`
4. `cargo build --release`
5. Optional: run the test suite `./test.zsh`

- If it exits with a zero return code, it was successful.

Collect the data

1. `time ./analysis/collect/bad_net.zsh | tee ./analysis/data/bad_net.csv`

- This should take approximately 6 minutes

Make & inspect the plots

1. Varying numbers of provers

- Run: `Rscript ./analysis/plotting/bad_net.R`
- Output plot: ./analysis/plots/bad_net.pdf
- It should be comparable to Figure 10