CollabFuzz: A Framework for Collaborative Fuzzing

Sebastian Österlund Elia Geretto Andrea Jemmett Emre Güler

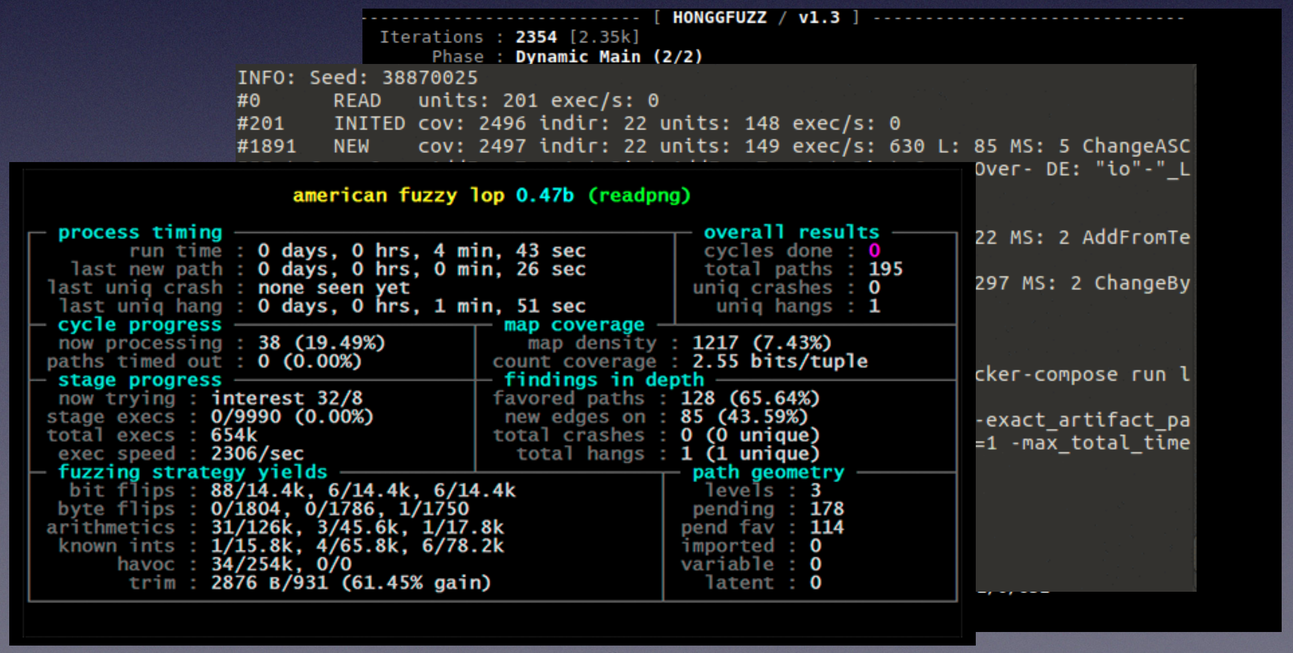
Philipp Görz Thorsten Holz Cristiano Giuffrida Herbert Bos

Vrije Universiteit Amsterdam *Ruhr-Universität Bochum*

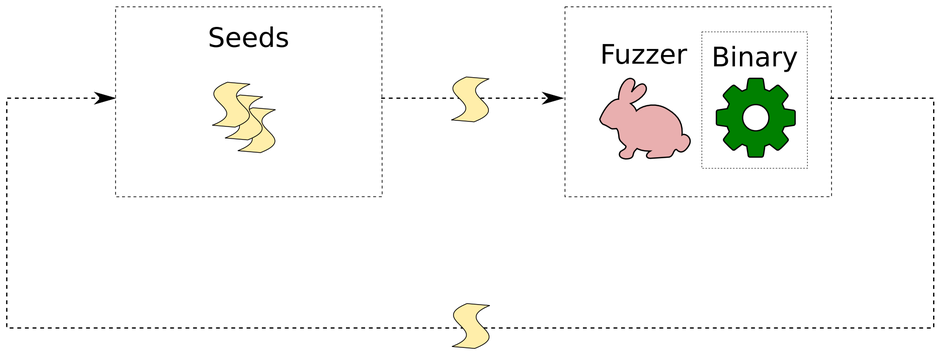
What's all the fuzz about?

Answer: run them *together*, orchestrated by CollabFuzz

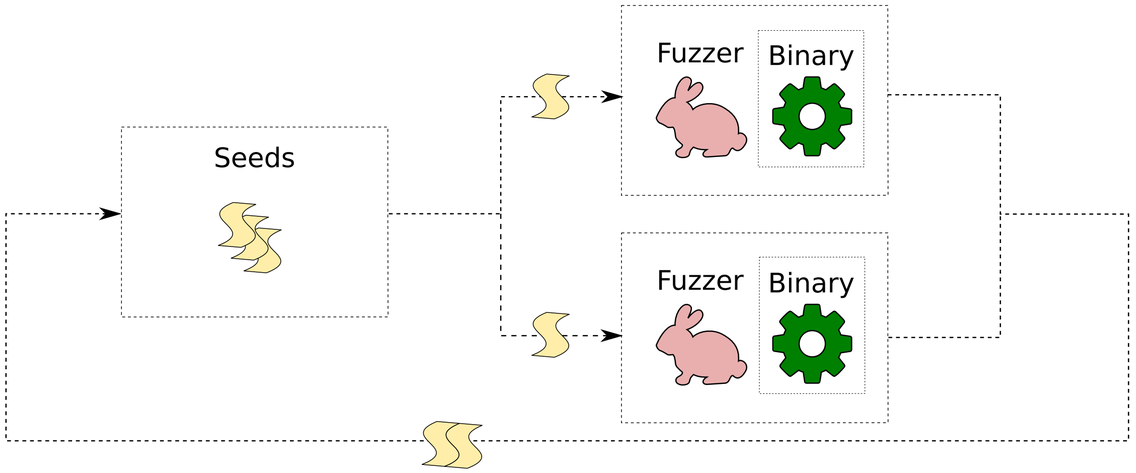
- **Problem**: Many **different** fuzzers to choose from! Many scripts to write!



Collaborative fuzzing



Single

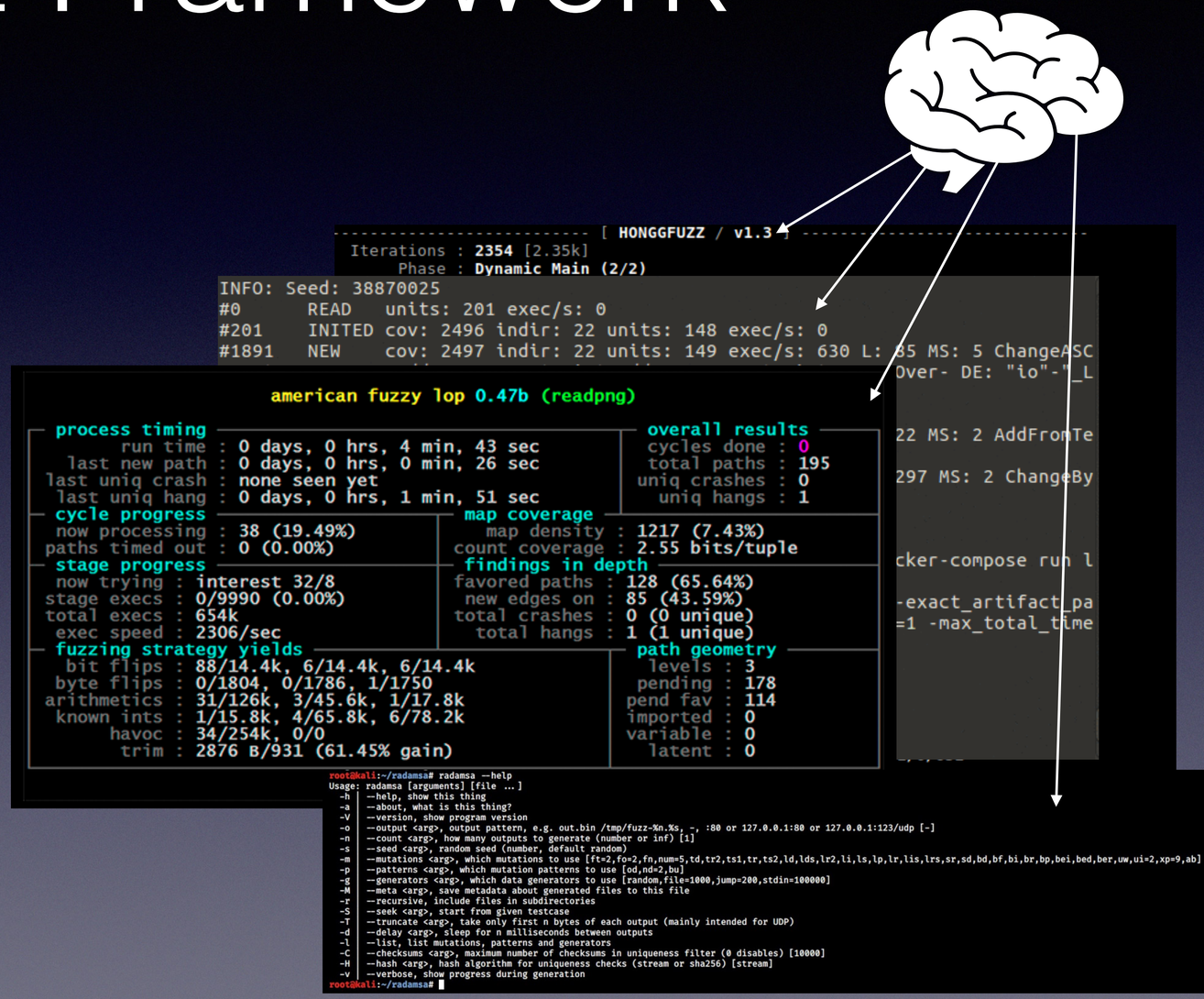


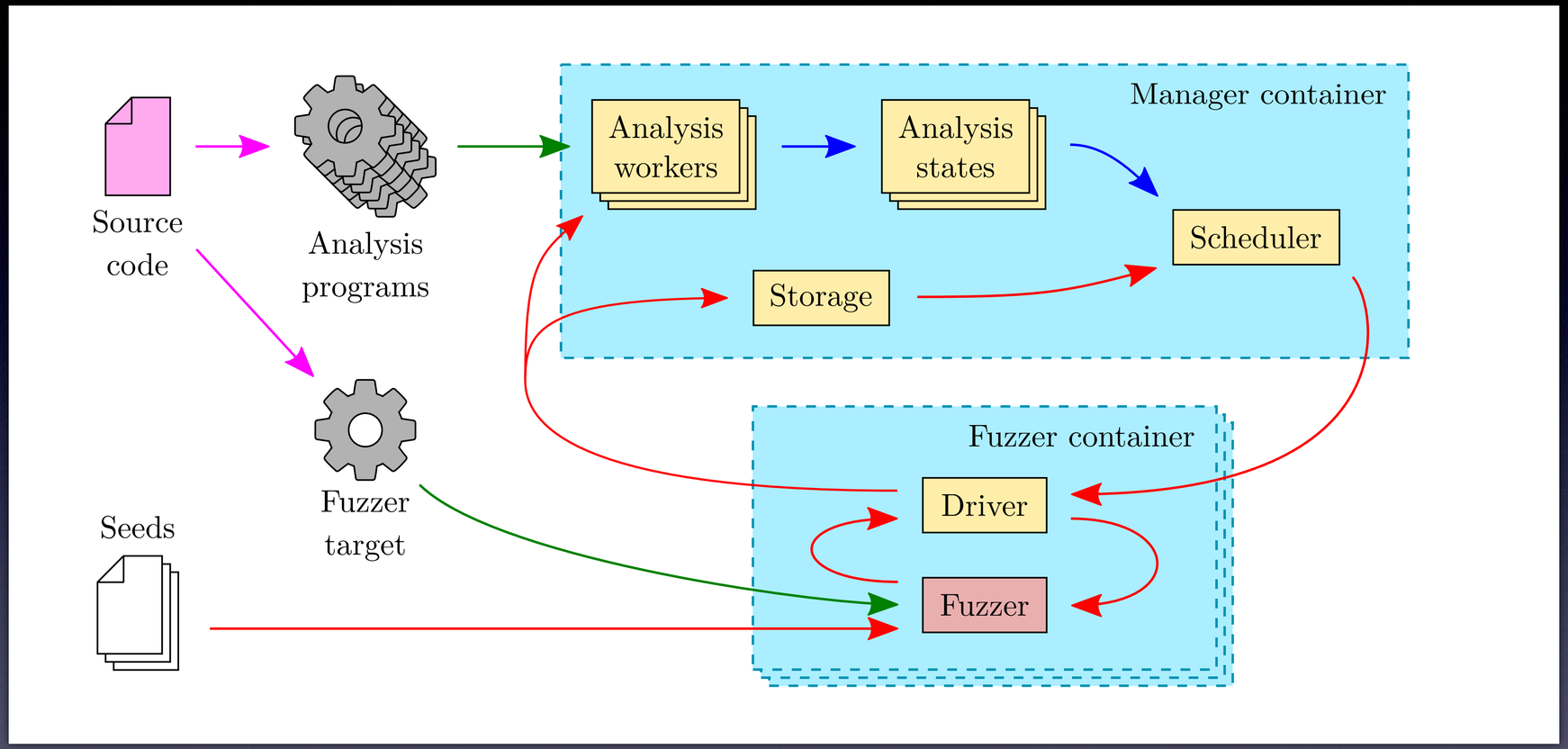
Collaborative

EnFuzz: Ensemble Fuzzing with Seed Synchronization among Diverse Fuzzers (USENIX Sec ‘19)

CollabFuzz Framework

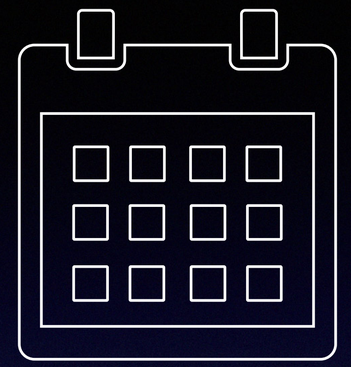
- Allows for a **central manager** to orchestrate many fuzzers
  - The managed fuzzers can be **different** fuzzers
  - Supports: AFL, AFLFast, AFL++, FairFuzz, Honggfuzz, LibFuzzer, QSYM, Radamsa
- A framework to collect results and perform **analysis** during a collaborative fuzzing campaign





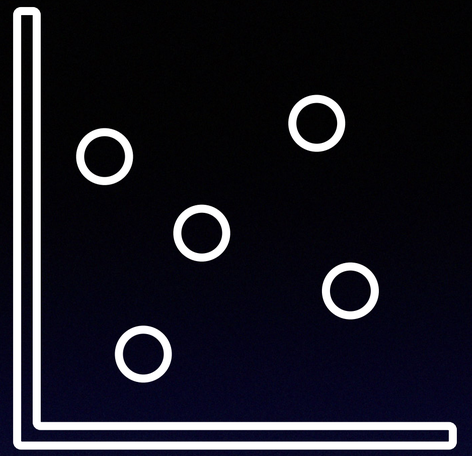
CollabFuzz Framework

Scheduling policies



- **Broadcast/ EnFuzz**: send all test cases to all fuzzers
- **(Cost-) Benefit**: use a “benefit” heuristics to select what seeds to send out
- In essence two possible “planes” to schedule on
  - *Temporally*: ***when*** to send out test case
  - *Spatial*: ***which*** fuzzers should get the test case

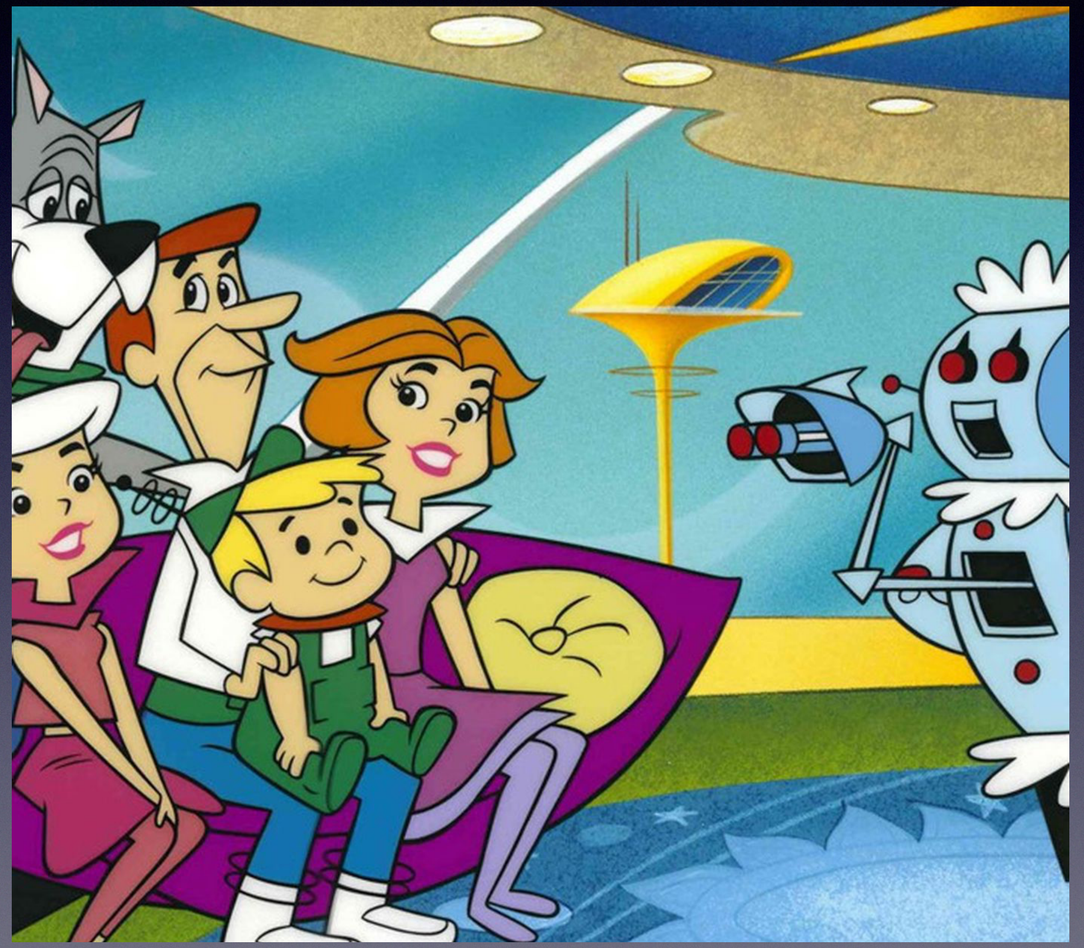
Scheduling results



- We could not observe any **statistically significant improvements**
- Reasons:
  - No actual control over what the fuzzers work on (**interface**)
  - We might not use the right **features**

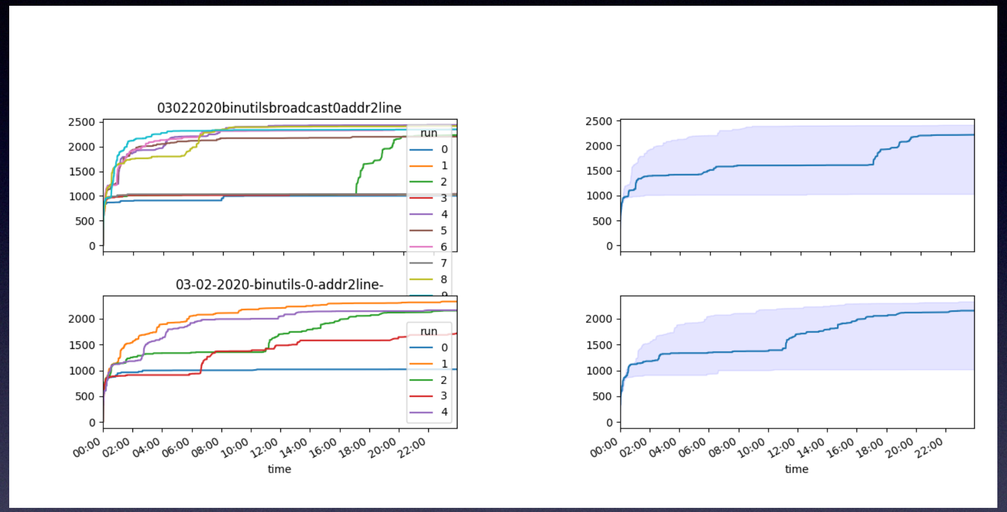
Future ideas in scheduling

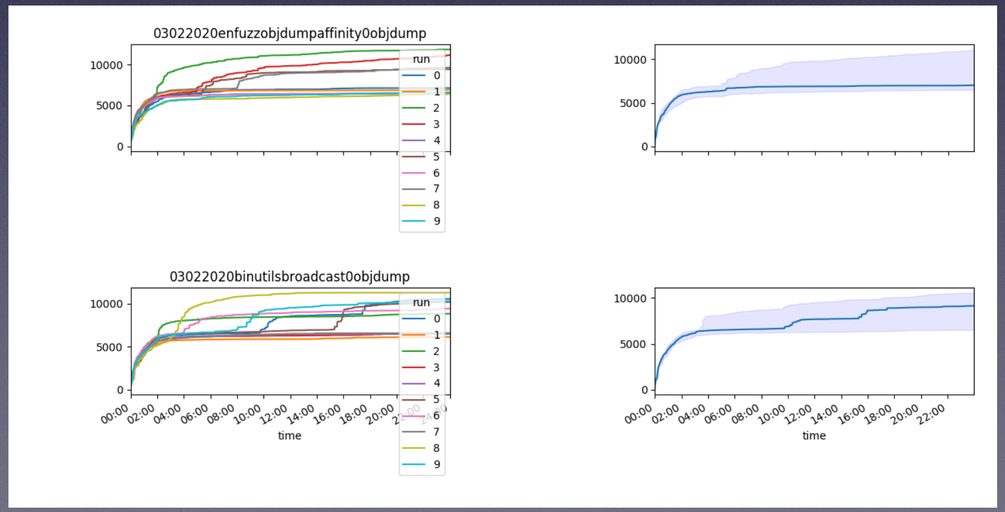
- Scheduling at a different level
  - Test case vs. resource scheduling
- More fine-grained control
  - Branch-level scheduling



Using CollabFuzz for Evaluations

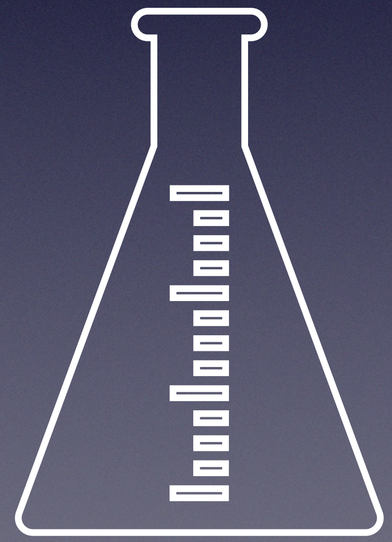
- **Repeatability** of experiments
- Large-scale experiments on a cluster
- Real-time analysis of campaign





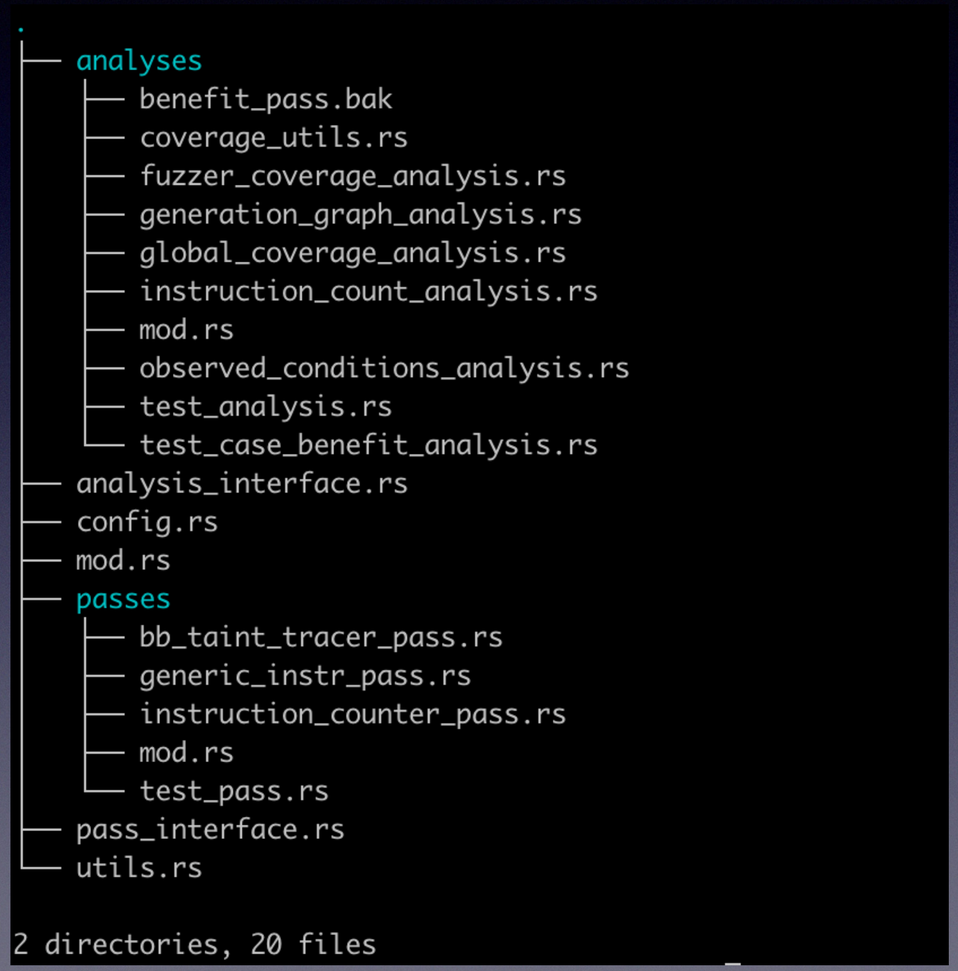
Analysis

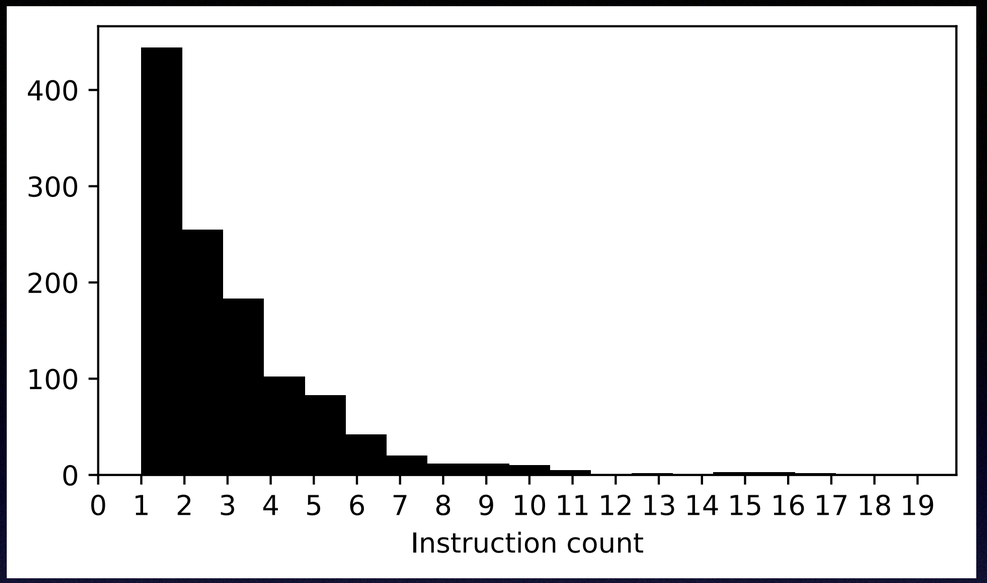
- **Analysis passes** can run when a new test case is discovered by a fuzzer
- E.g., get the coverage of the test case
- Allows pipelining (i.e., do taint track analysis if there is new coverage)
- Quite flexible

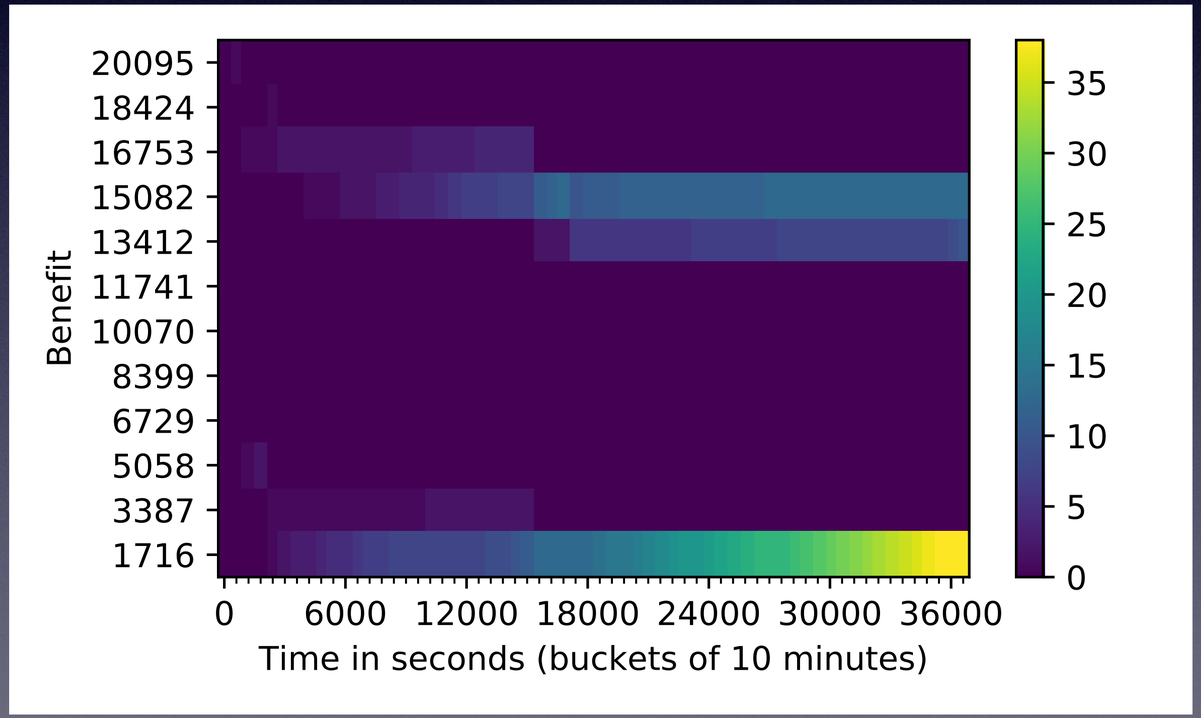


Analysis Passes

- Coverage
- Taint information
- Tainted instruction count pass
- Branch analysis
- Flexible: global aggregate data, per-fuzzer data, per test-case data




400
300
200
100
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
Instruction count


Benefit
20095
18424
16753
15082
13412
11741
10070
8399
6729
5058
3387
1716
0 6000 12000 18000 24000 30000 36000
Time in seconds (buckets of 10 minutes)
35
30
25
20
15
10
5
0

Technical Info

- Each fuzzer needs a **“driver”**
  - Program to monitor the fuzzer (we have a generic one for AFL-like fuzzers)
- Drivers communicate with the framework over ZeroMQ
  - Allows large-scale **distributed** fuzzing
- Framework: **Rust**, driver: Python

Configuration

- Sets up experiments using YAML files
- Easy to queue long-running experiments
- All fuzzer-targets are contained within Docker containers
- It's even possible to run experiments on a cluster using Docker Swarm/ Kubernetes

```
---
targets:
  -
    binary: guetzli
    name: guetzli
    input: ./inputs/guetzli
  -
    binary: json
    name: json
    input: ./inputs/json
experiment:
  name: "my-cool-experiment"
  timeout: 36000
  repeat: 10
  use_collab: true
  enable_afl_affinity: true
  scheduler: "test_case_benefit"
fuzzers:
  -
    name: qsym
    type: qsym
    parallel: 1
  -
    name: libfuzzer
    type: libfuzzer
    parallel: 1
  -
    name: honggfuzz
    type: honggfuzz
    parallel: 2
```

Targets

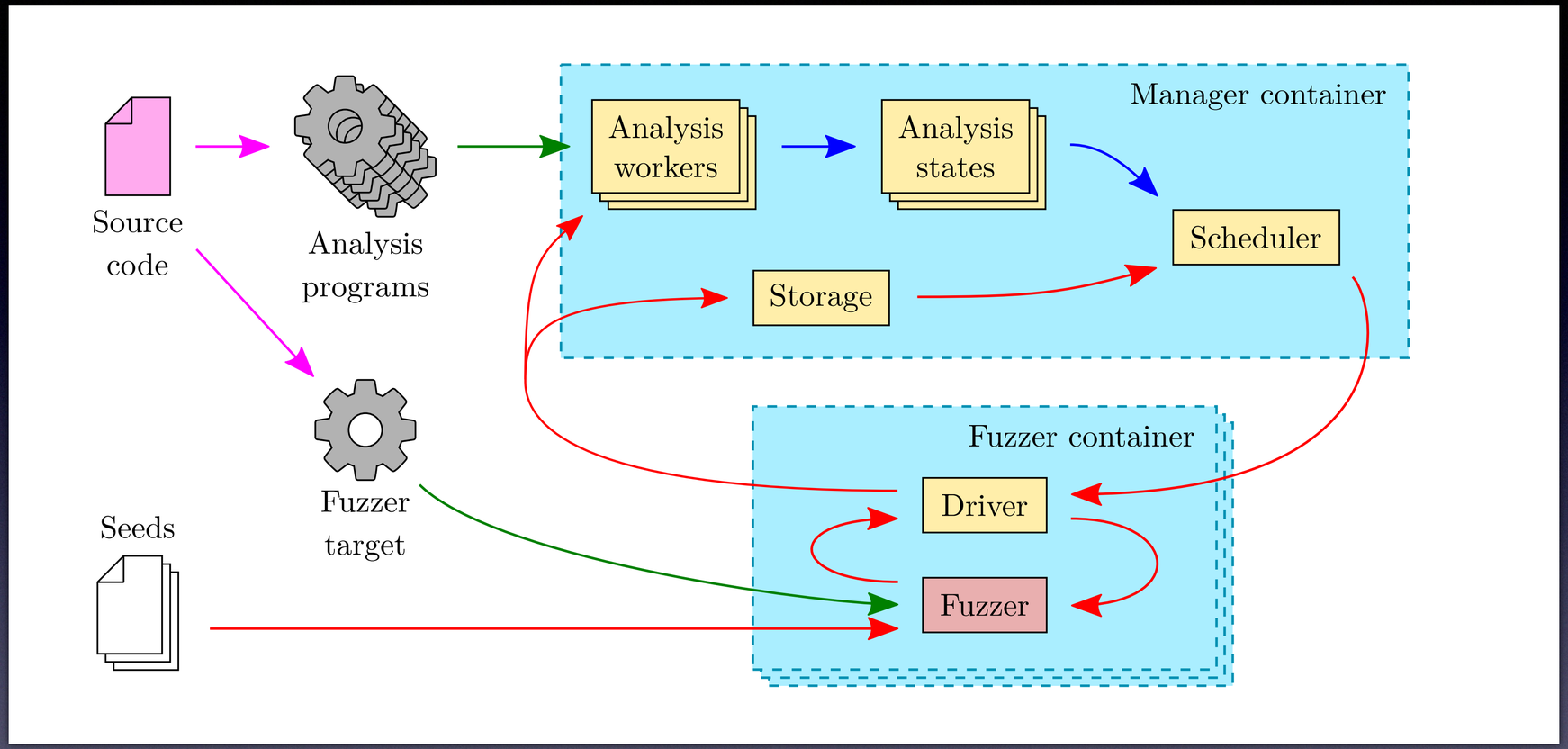
Parameters

Fuzzer config

Logging

- The framework dumps all data in a sqlite database
- E.g., all the analysis pass results, scheduling decisions, fuzzer events

```
sqlite> .schema
CREATE TABLE fuzzer_types (
             id            INTEGER PRIMARY KEY,
             description   TEXT
           );
CREATE TABLE test_case_types (
             id            INTEGER PRIMARY KEY,
             description   TEXT
           );
CREATE TABLE fuzzer_event_types (
             id            INTEGER PRIMARY KEY,
             description   TEXT
           );
CREATE TABLE analysis_types (
             id            INTEGER PRIMARY KEY,
             description   TEXT
           );
CREATE TABLE fuzzers (
             fuzzer_id         INTEGER PRIMARY KEY,
             fuzzer_type_id    INTEGER REFERENCES fuzzer_types
           );
CREATE TABLE test_cases (
             hash              TEXT PRIMARY KEY,
             test_case_type_id INTEGER REFERENCES test_case_types,
             discovery_fuzzer  INTEGER REFERENCES fuzzers,
             discovery_time    INTEGER
           );
CREATE TABLE dispatch (
             test_case_hash    TEXT REFERENCES test_cases,
             fuzzer_id         INTEGER REFERENCES fuzzers,
             dispatch_time     INTEGER
           );
CREATE TABLE fuzzer_events (
             fuzzer_id         INTEGER REFERENCES fuzzers,
             event_type_id     INTEGER REFERENCES fuzzer_event_types,
             event_time        INTEGER
           );
CREATE TABLE analysis_states (
             test_case_hash    INTEGER REFERENCES test_cases,
             analysis_id       INTEGER REFERENCES analysis_types,
             analysis_dump     BLOB,
             PRIMARY KEY(test_case_hash, analysis_id)
           );
```



CollabFuzz Framework

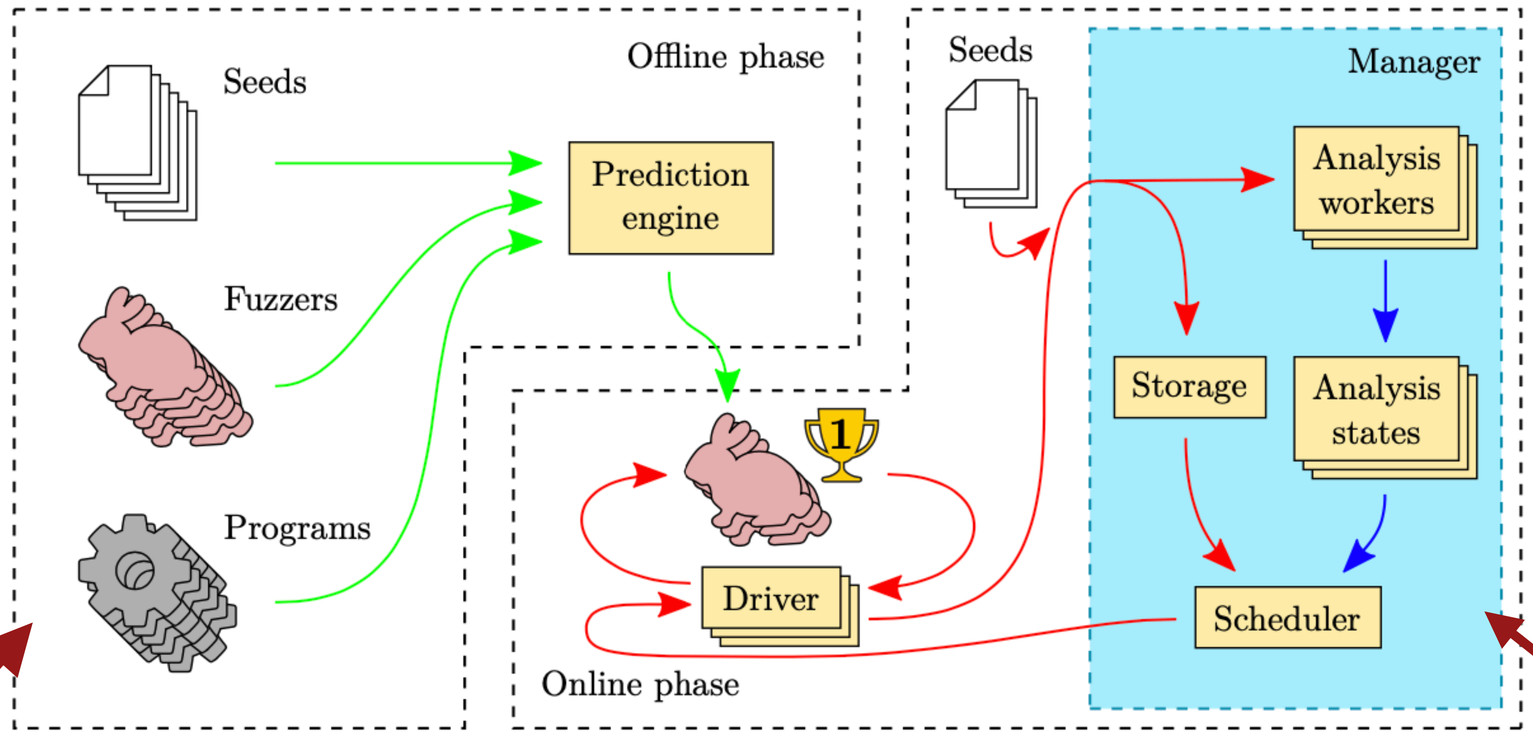


Fig. 2: An overview of CollabFuzz and its different components.

CUPID (ACSAC '20)

CollabFuzz (EuroSec '21)

DEMO

@vu5ec
@sirmc

Conclusion

- CollabFuzz: orchestrate collaborative fuzzing campaigns, also in a distributed setting
- Allows expression of scheduling policies
- Enables real-time analysis of large-scale campaigns
- Available at: github.com/vusec/collabfuzz

CollabFuzz: A Framework for Collaborative Fuzzing

Sebastian Österlund Elia Geretto Andrea Jemmett Emre Güler

Philipp Görz Thorsten Holz Cristiano Giuffrida Herbert Bos

Vrije Universiteit Amsterdam *Ruhr-Universität Bochum*