Fex

A Software Systems Evaluator

Oleksii Oleksenko,
Dmitrii Kuvaiskii, Christof Fetzer

TECHNISCHE UNIVERSITÄT DRESDEN

Pramod Bhatotia

THE UNIVERSITY of EDINBURGH

Code reuse is everywhere…

- Libraries
- Frameworks
- Software components

… but not in evaluation workflow

- **Ad-hoc scripts**
  - Bash / Python / R

… but not in evaluation workflow

- Ad-hoc scripts
  - Bash / Python / R
- **Written from scratch**
  - for each new project

… but not in evaluation workflow

- Ad-hoc scripts
  - Bash / Python / R
- **Written from scratch**
  - for each new project
  - for each new benchmark suite

… but not in evaluation workflow

- Ad-hoc scripts
  - Bash / Python / R
- **Written from scratch**
  - for each new project
  - for each new benchmark suite
  - for each new experiment

Consequences

- Rigid setup
  - hard to **extend**
  - often, leads to **simplistic** evaluation

Consequences

- Rigid setup
  - hard to extend
  - often, leads to simplistic evaluation
- Inconsistent results
  - no guarantee of **reproducibility**

Surprisingly, not many solutions

- **Benchmark suites** [PARSEC, SPEC]
  - narrow view
  - hard to extend

Surprisingly, not many solutions

- Benchmark suites [PARSEC, SPEC]
  - narrow view
  - hard to extend
- **Sound measurement tools** [Stabilizer, Coz]
  - improve experimental environment
  - no automation

Surprisingly, not many solutions

- Benchmark suites [PARSEC, SPEC]
  - narrow view
  - hard to extend
- Sound measurement tools [Stabilizer, Coz]
  - improve experimental environment
  - no automation
- **Build tools** [Automake, CMake, Scons]
  - automatic build configuration
  - only build stage

Design goals

- Extensibility
- Reproducibility
- Practicality

Extensibility

- Goal:
  - easy to create new experiments
- Solution:
  - out-of-the-box experiments
  - customization

Reproducibility

- Goal:
  - guaranteed software stack
- Solution:
  - Docker integration
  - scripts for specific software versions

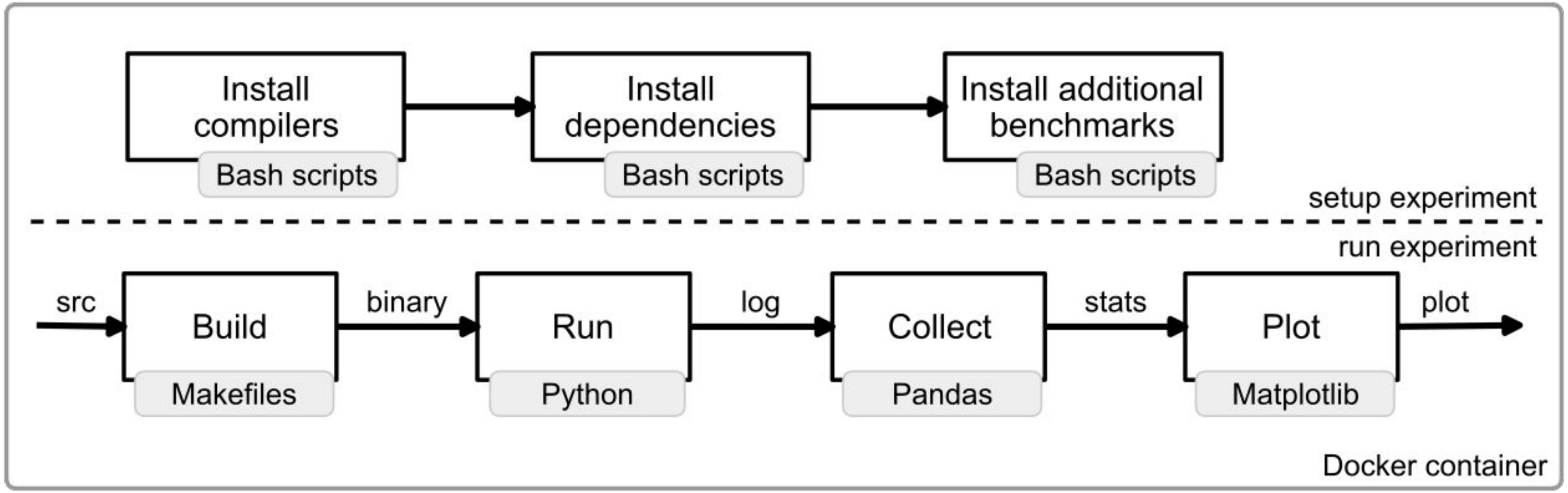
Practicality

- Goal:
  - simple to compose benchmarks
- Solution:
  - loosely coupled build system

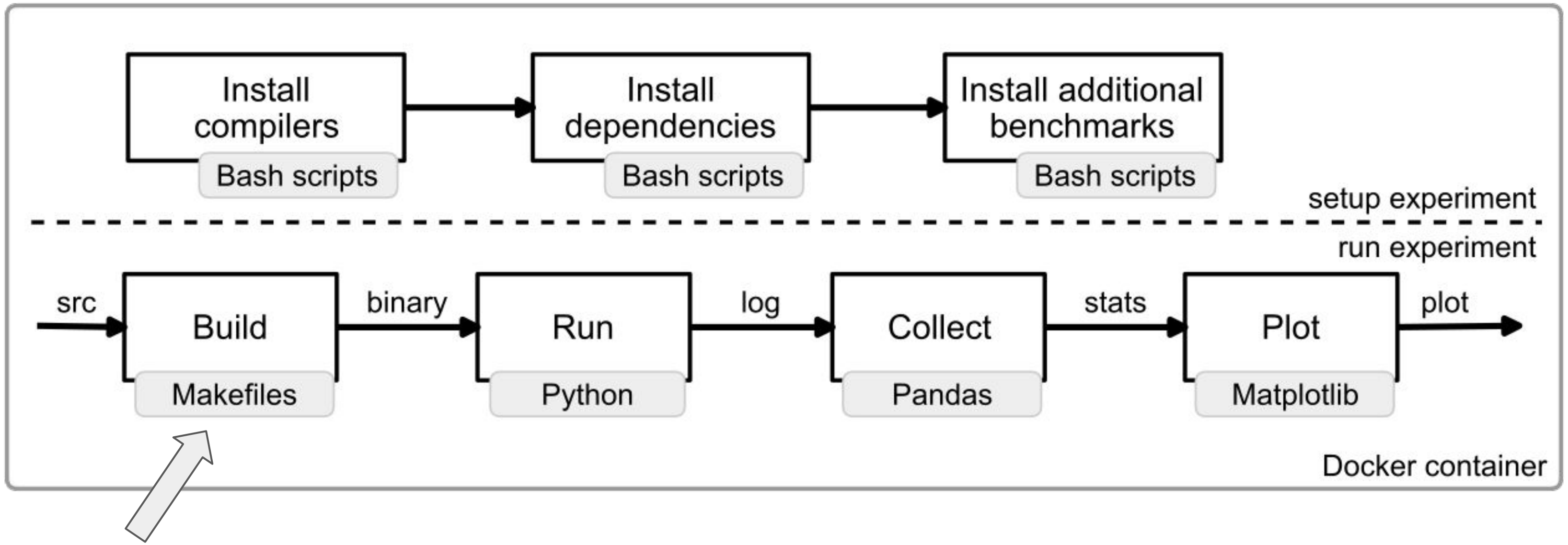
Outline

- ~~Motivation~~
- **Design**
- Demo

Workflow

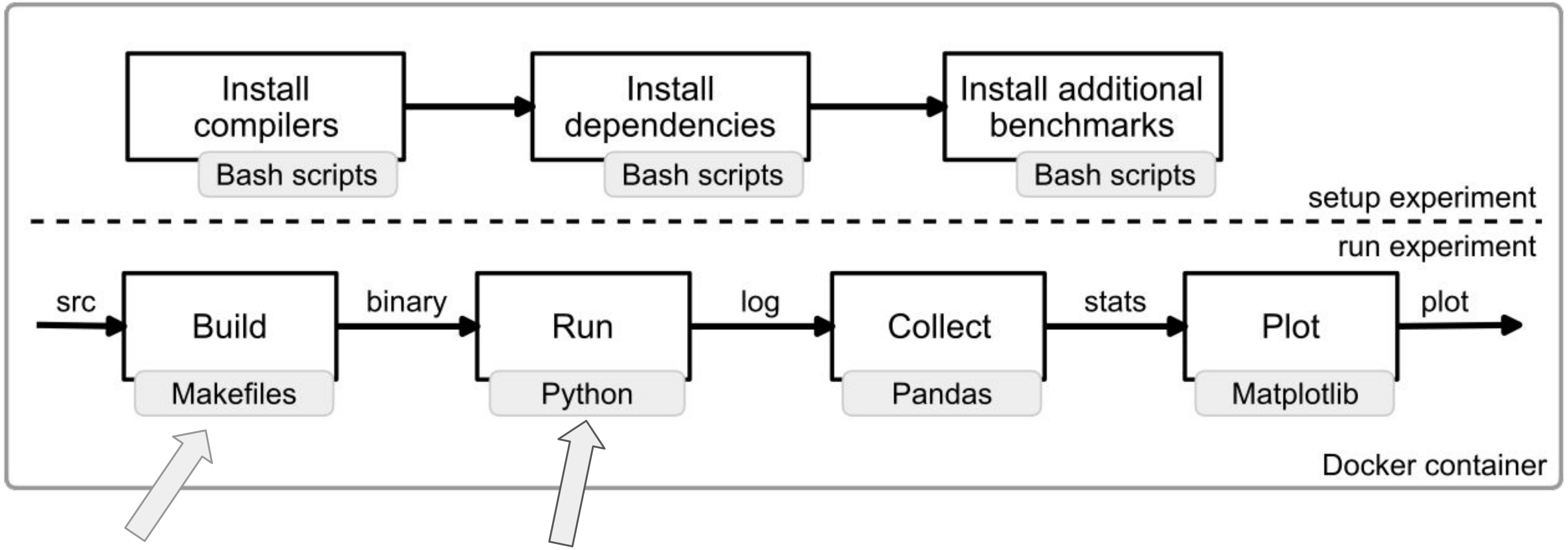

Install compilers
Bash scripts
Install dependencies
Bash scripts
Install additional benchmarks
Bash scripts
setup experiment
run experiment
src
Build
Makefiles
binary
Run
Python
log
Collect
Pandas
stats
Plot
Matplotlib
plot
Docker container

Workflow



- Application-specific
- Type-specific
- Environment variables

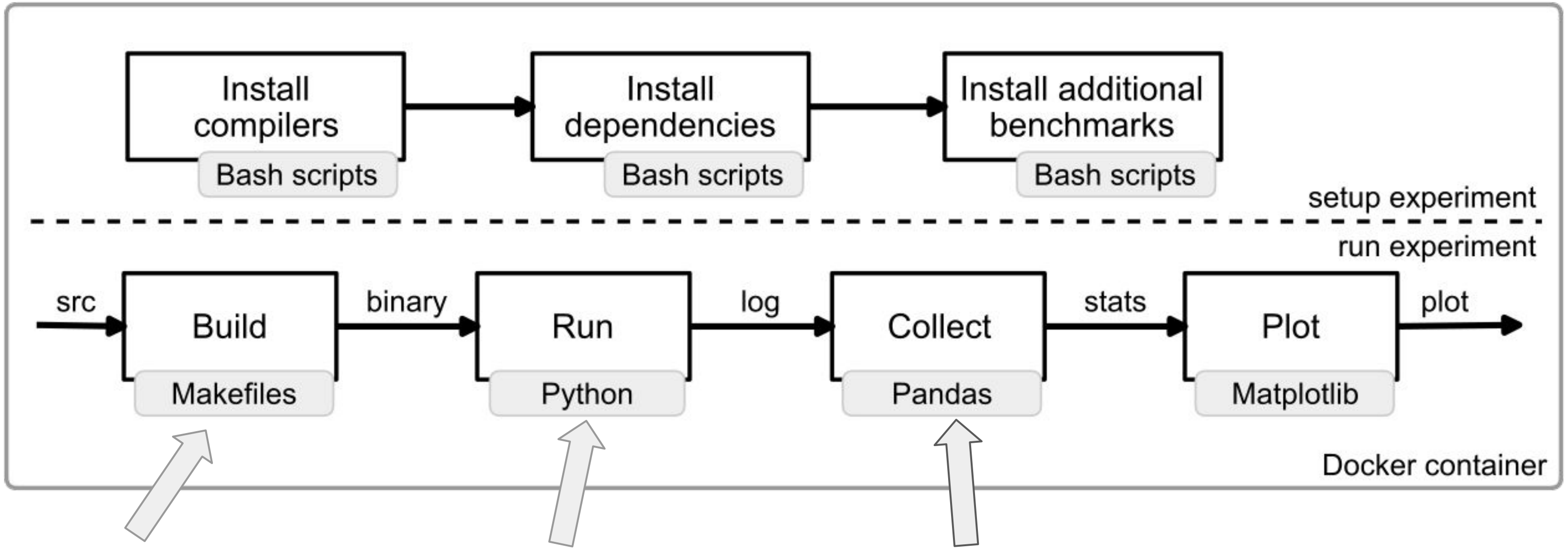
Workflow



- Application-specific
- Type-specific
- Environment variables

- Experiment execution
- Hooks for customization

Workflow

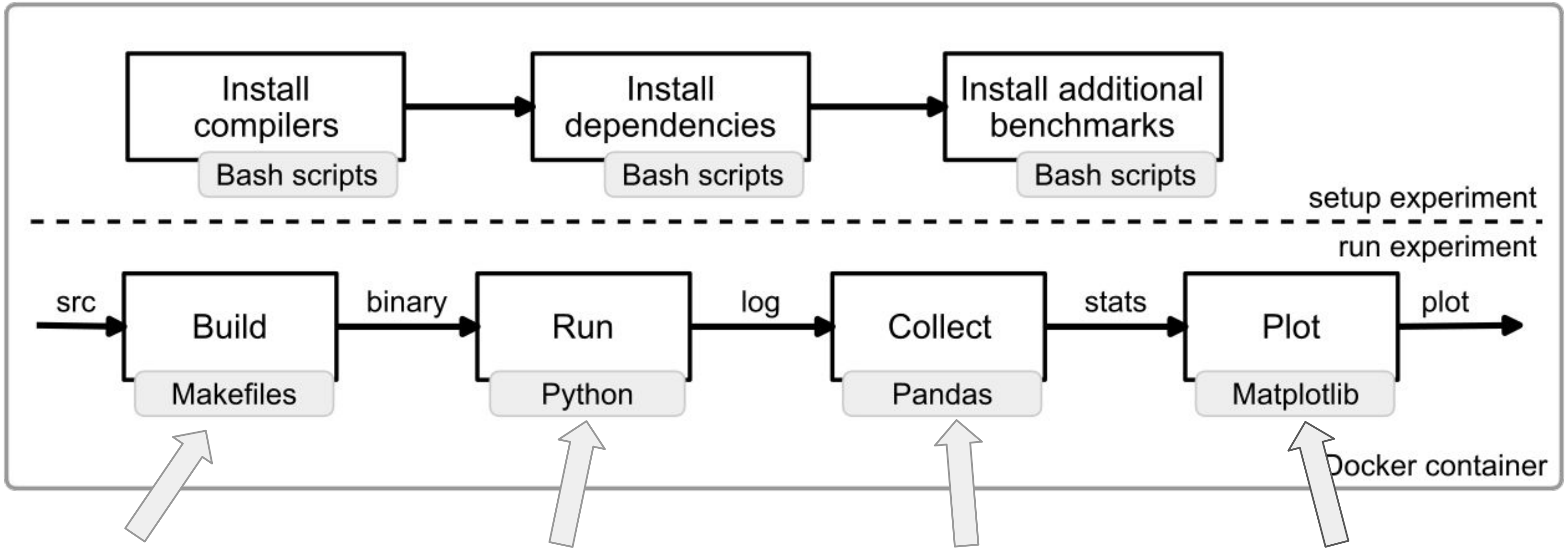


- Application-specific
- Type-specific
- Environment variables

- Experiment execution
- Hooks for customization

- Parse logs
- Aggregate and analyze
- Store results

Workflow



- Application-specific
- Type-specific
- Environment variables

- Experiment execution
- Hooks for customization

- Parse logs
- Aggregate and analyze
- Store results

- Based on matplotlib
- Superclasses for common plots

Outline

- ~~Motivation~~
- ~~Design~~
- **Demo**

A simple experiment

- Evaluate GCC optimizations
  - performance overhead
  - on benchmarks from Phoenix 3.0

Summary

Automate your research to make it:

- efficient
- flexible
- comprehensive
- reproducible

Summary

Automate your research to make it:

- efficient
- flexible
- comprehensive
- reproducible

GitHub

https://github.com/tudinfse/fex

Summary

Automate your research to make it:

- efficient
- flexible
- comprehensive
- reproducible

https://github.com/tudinfse/fex

Thanks!

oleksii.oleksenko@tu-dresden.de

Backup

Outline

- ~~Motivation~~
- ~~Design~~
- ~~Demo~~
- **Example**

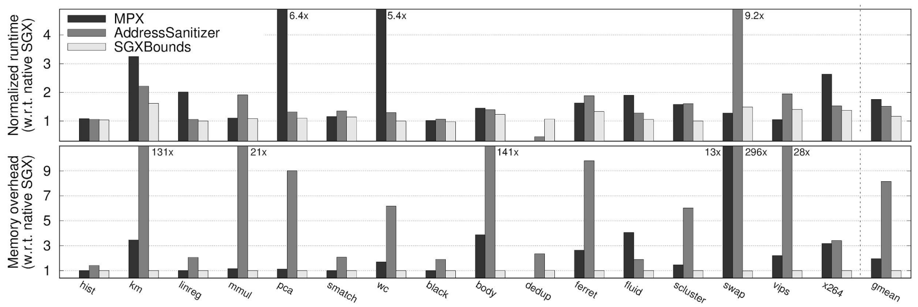
Origin

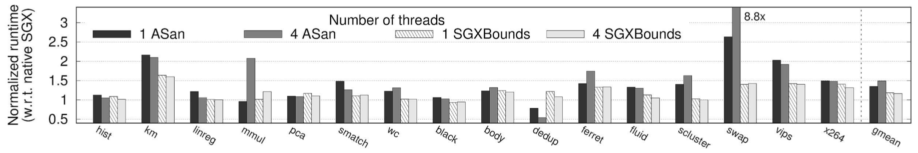
Started as an internal tool:

- Elzar [DSN'16]
- SGXBounds [EuroSys'17]
- MPX Explained

SGXBounds

- 4 experiment types
- 2 environment:
  - in- and outside SGX enclaves
- 2 compilers
- 38 benchmarks
  - 3 benchmark suites
- 3 case-studies
- 1 security benchmark


MPX
AddressSanitizer
SGXBounds
Normalized runtime (w.r.t. native SGX)
Memory overhead (w.r.t. native SGX)
6.4x
5.4x
9.2x
131x
21x
141x
13x
296x
28x
hist
km
linreg
mmul
pca
smatch
wc
black
body
dedup
ferret
fluid
scluster
swap
vips
x264
gmean


Number of threads
1 ASan
4 ASan
1 SGXBounds
4 SGXBounds
Normalized runtime (w.r.t. native SGX)
8.8x
hist
km
linreg
mmul
pca
smatch
wc
black
body
dedup
ferret
fluid
scluster
swap
vips
x264
gmean