REPRODUCIBLE CONTAINERS FOR SCIENTIFIC COMPUTING

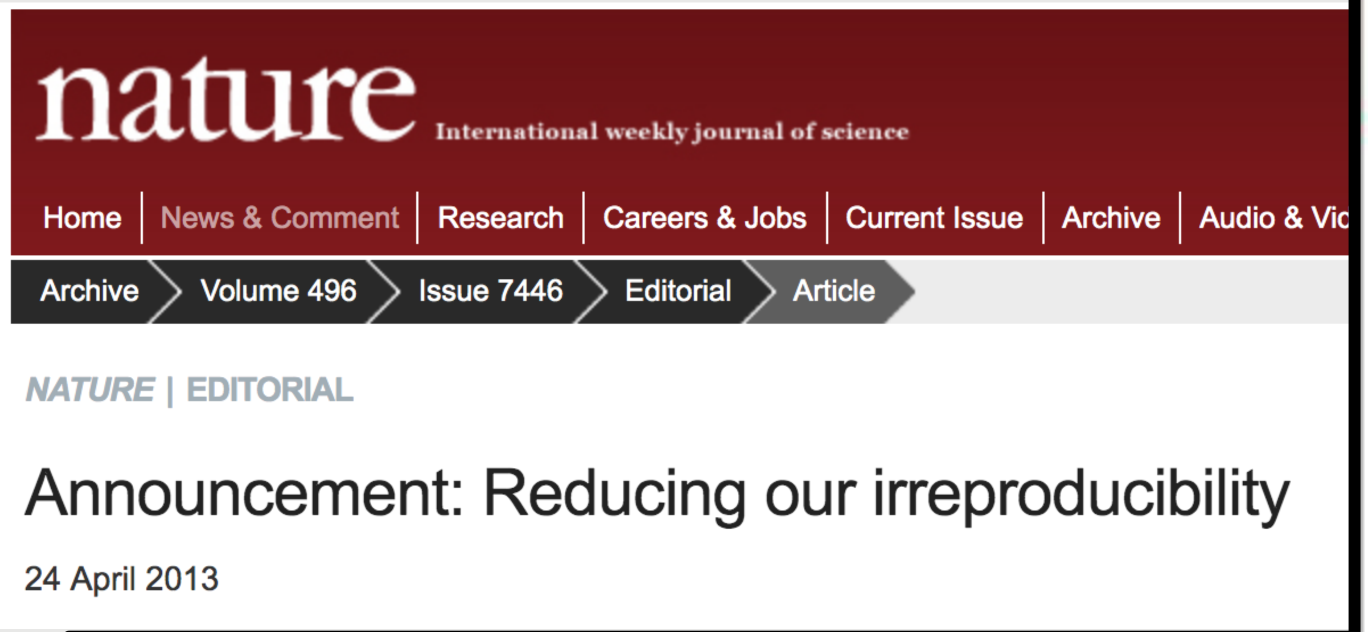
Tanu Malik

The DICE Laboratory

School of Computing

DePaul University, Chicago IL

Reproducibility and Replicability in Sciences: A growing concern



nature International weekly journal of science

Home | News & Comment | Research | Careers & Jobs | Current Issue | Archive | Audio & Vi

Archive > Volume 496 > Issue 7446 > Editorial > Article

NATURE | EDITORIAL

Announcement: Reducing our irreproducibility

24 April 2013

The Economist

Washington's lawyer surplus

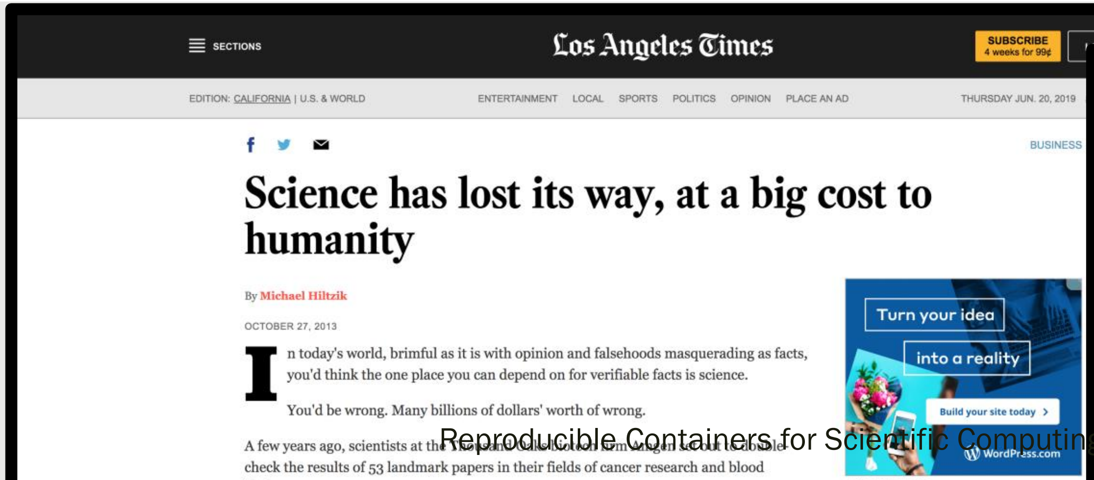
How to do a nuclear deal with Iran

Investment tips from Nobel economists

Junk bonds are back

The meaning of Sachin Tendulkar

HOW SCIENCE GOES WRONG



SECTIONS Los Angeles Times SUBSCRIBE 4 weeks for 99¢

EDITION: CALIFORNIA | U.S. & WORLD ENTERTAINMENT LOCAL SPORTS POLITICS OPINION PLACE AN AD THURSDAY JUN. 20, 2019

BUSINESS

Science has lost its way, at a big cost to humanity

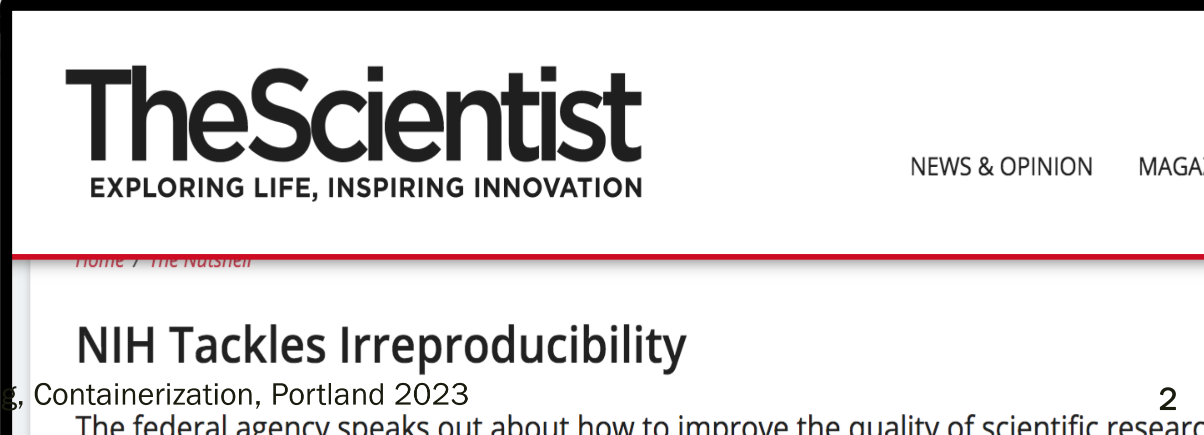
By Michael Hiltzik

OCTOBER 27, 2013

In today's world, brimful as it is with opinion and falsehoods masquerading as facts, you'd think the one place you can depend on for verifiable facts is science.

You'd be wrong. Many billions of dollars' worth of wrong.

A few years ago, scientists at the Thousand Oaks biotech firm Amgen set out to double-check the results of 53 landmark papers in their fields of cancer research and blood



TheScientist EXPLORING LIFE, INSPIRING INNOVATION

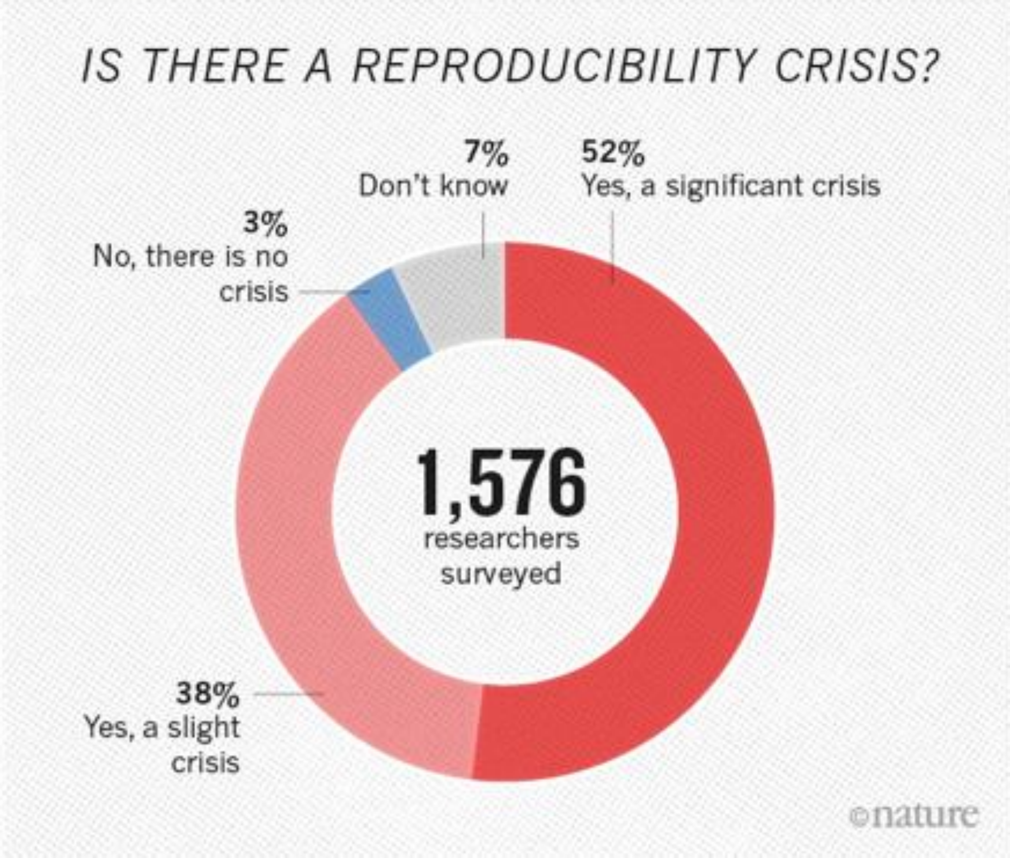
NEWS & OPINION MAGA

Home / The Nutshell

NIH Tackles Irreproducibility

The federal agency speaks out about how to improve the quality of scientific researc

Concerns across sciences



Source: 1500 scientists lift the lid on reproducibility. *Nature Survey. Accessed 25th May, 2016*

Concerns in Computer Sciences

- Definition: Given an experiment described at *t*, Can it be downloaded at *t*′, and its source code be built within a reasonable amount of time.

N = 402 experiments

Response	Percentage	# of experiments
No response to code requests	36.1%	145
Declined to provide code	10.2%	41

N = 219 experiments

Response	Percentage	# of experiments
Failed to build	5.02%	11
Built <= 30 minutes	32.4%	71
Built > 30 minutes	15.9%	35
Reasonable effort	46.6%	102

T. Proebsting, A. M. Warren, and C. Collberg. 2015.
Repeatability and benefaction in computer systems research. University of Arizona TR 14. Vol. 4. 1–68.

Challenges in Reproducibility and Replicability

- **Reproducibility:** Obtaining consistent results when using the same or similar input data, computational steps, conditions of analysis, etc.

- **Replicability:** Obtaining consistent results when using different input data, computational steps, conditions of analysis, etc.

Challenges

- ➢ Need for guarantees-based or statistical-based methods for conduct of reproducible research
- ➢ Need for infrastructure that supports reproducible research
- ➢ Need for policies that incentivize and enforce reproducible research

National Academies of Sciences, Engineering, and Medicine 2019. Reproducibility and Replicability in Science. Washington, DC: The National Academies Press. https://doi.org/10.17226/25303.

Foundations of Reproducible Scientific Computing

Guarantees

Lineage-based methods

Replay: VLDB’22

Debugging: TaPP’20, HiPC’22

Guarantees: TaPP’13, ICDE’15, MDPI’18

Infrastructure

Container-based Tools

w/ Lineage: eScience’17, eScience ’19, eScience’22

Size Reduction: HiPC’20, Access’23,

Vs Docker: ICCS’15

Documenting: PARCO’20

Policies

Artifact Evaluation

Surveys: PRECS’22, IEEE CiSE’21

Foundations of Reproducible Scientific Computing

Guarantees

Lineage-based methods

Replay: VLDB’22

Debugging: TaPP’20, HiPC’22

Guarantees: TaPP’13, ICDE’15, MDPI’18

Infrastructure

Container-based Tools

w/ Lineage: eScience’17, eScience ’19, eScience’22

Size Reduction: PARCO’20, HiPC’20, Access’23,

Vs Docker: ICCS’15
Documentation: PARCO’20

Policies

Artifact Evaluation

Surveys: PRECS’22, IEEE CiSE’21

Share and Reproduce an Application

Alice wants to share her input data files and program source code with Bob

Bob wants to reproduce Alice’s application to validate her outputs.

Share and ~~Reproduce~~ Repeat an Application

Alice wants to share her input data files and program source code with Bob

Bob wants to repeat Alice’s application to validate her inputs and outputs.

Alice’s sharing options

1. Email a tar/gzip
2. Build a website with model code, parameters, and data
3. Create a virtual machine or container

Not sufficient for reproducible research

1. ~~Email a tar/gzip~~
2. ~~Build a website with model code, parameters, and data~~
3. ~~Create a virtual machine or container~~

Not sufficient for reproducible research

1. ~~Email a tar/gzip~~
2. ~~Build a website with model code, parameters, and data~~

    Missing environment files: No isolation guarantee

3. ~~Create a virtual machine or container~~

    Failing rebuilds: No repeatable guarantee

Docker for repeating an application?

```
# syntax=docker/dockerfile:1
FROM golang:1.16-alpine AS build

# Install tools required for project
# Run `docker build --no-cache .` to update dependencies
RUN apk add --no-cache git
RUN go get github.com/golang/dep/cmd/dep

# List project dependencies with Gopkg.toml and Gopkg.lock
# These layers are only re-built when Gopkg files are updated
COPY Gopkg.lock Gopkg.toml /go/src/project/
WORKDIR /go/src/project/
 # Install library dependencies
RUN dep ensure -vendor-only

# Copy the entire project and build it
COPY . /go/src/project/
RUN go build -o /bin/project

# This results in a single layer image
FROM scratch
COPY --from=build /bin/project /bin/project
ENTRYPOINT ["/bin/project"]
CMD ["--help"]
```

Because this line installs the most recent version upon rebuilding

This line no longer rebuilds

Can we compose Containerization and Lineage?

http://sciunit.run

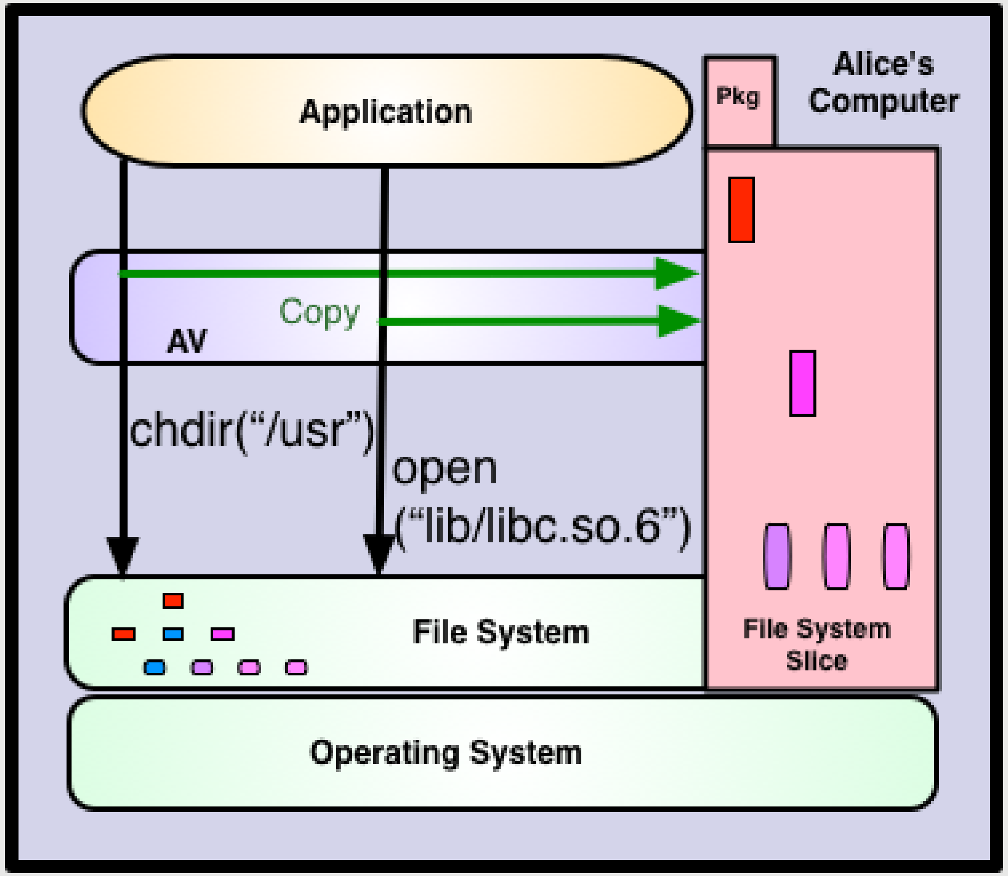
Sciunit: Compose Containerization and Lineage

http://sciunit.run

Key Idea: Identify and isolate data dependencies during program execution and infer lineage between dependencies

D.H. Ton That, G. Fils, Z. Yuan, T. Malik. Sciunits: Reusable Research Objects. In IEEE eScience Conference (eScience), 374-383, 2017

Host: Use *ptrace* to observe executions

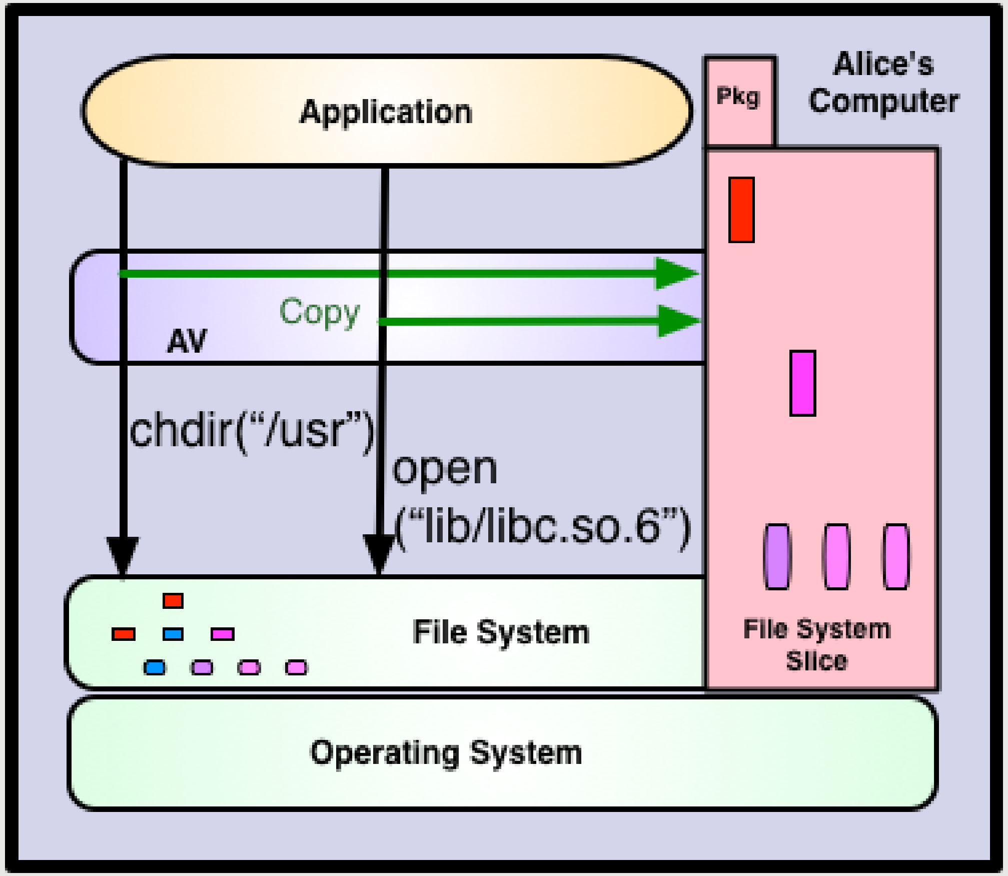


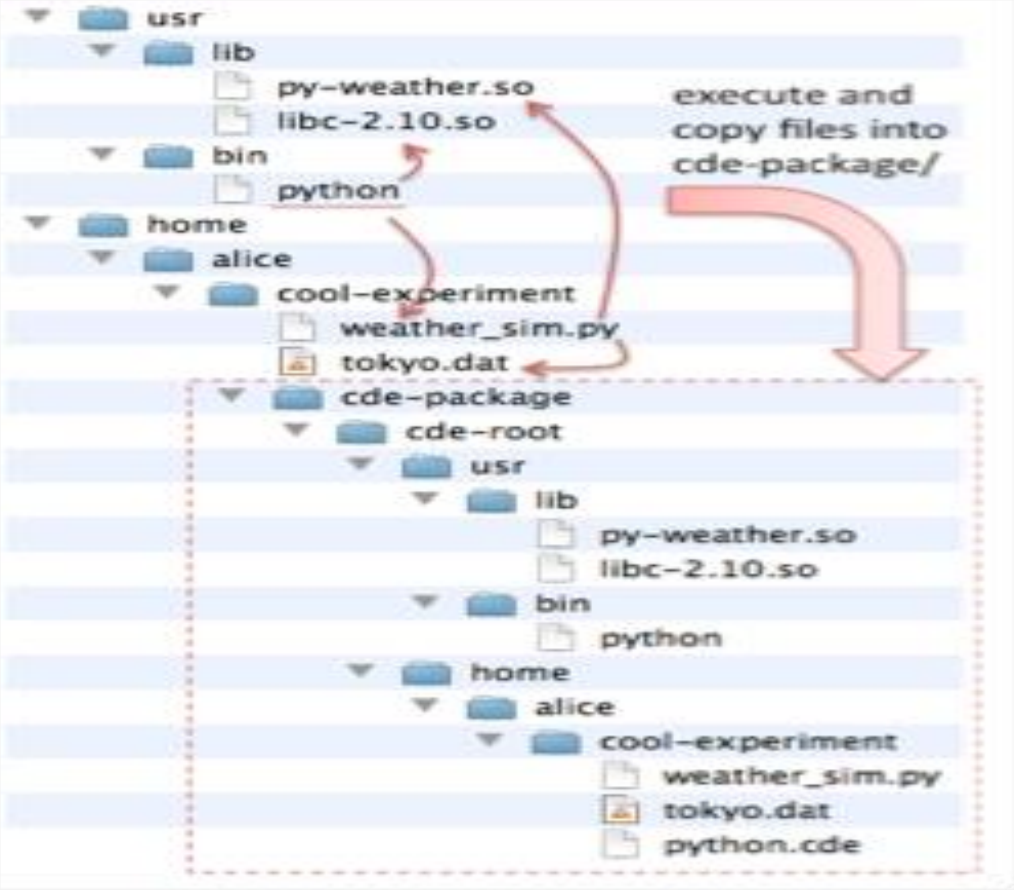
Audit Phase

- Audit ~50 system calls related to process control, file I/O, and network
  - If file is /dev/random capture return bytes as well
- At the time of interception:
  - Generate an execution trace of system call events in real-time
  - Copy files mentioned as part of system calls into a container

Create a *chroot*-based container

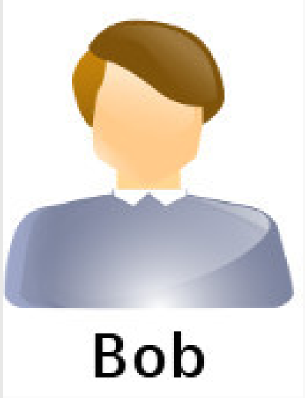
- Audit provenance during container creation time



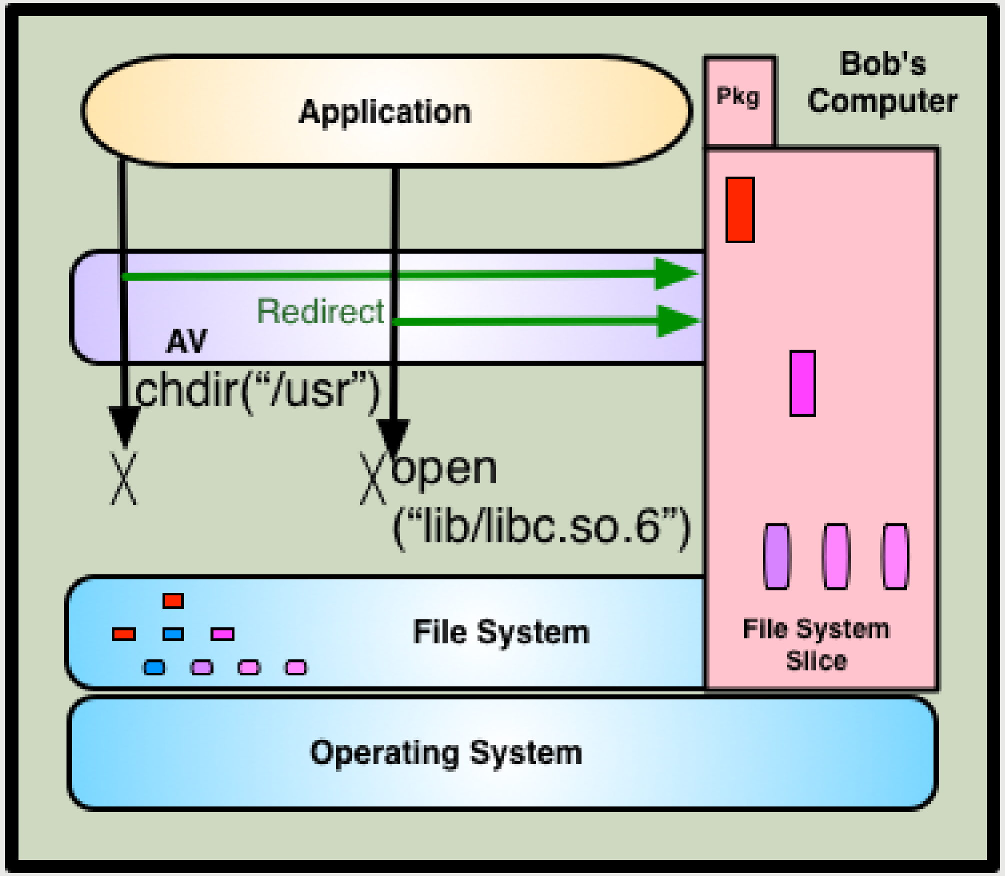

```
▼ usr
  ▼ lib
      py-weather.so
      libc-2.10.so
  ▼ bin
      python
▼ home
  ▼ alice
    ▼ cool-experiment
        weather_sim.py
        tokyo.dat
      ▼ cde-package
        ▼ cde-root
          ▼ usr
            ▼ lib
                py-weather.so
                libc-2.10.so
            ▼ bin
                python
          ▼ home
            ▼ alice
              ▼ cool-experiment
                  weather_sim.py
                  tokyo.dat
                  python.cde
```

Alices shares *sciunits* and Bob repeats them

sciunits



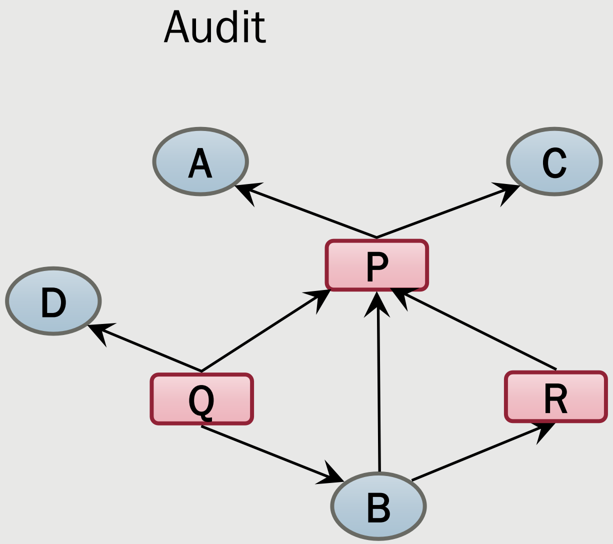
Target: Use *ptrace* to redirect executions into the container

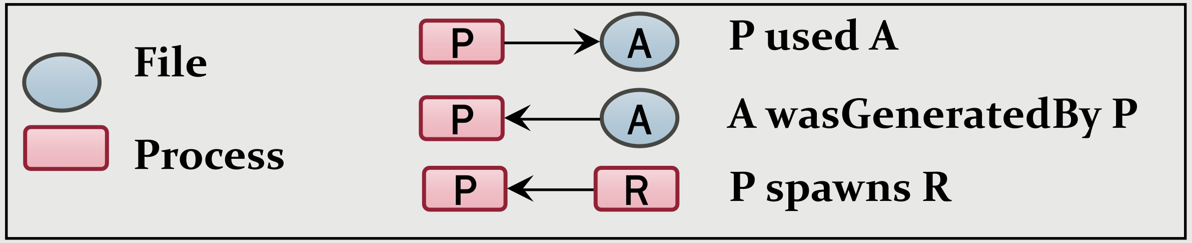


Repeat Phase

- Redirection during repetition is only for file- and network-related system events.
- Repeat execution happens within a process and file namespace.

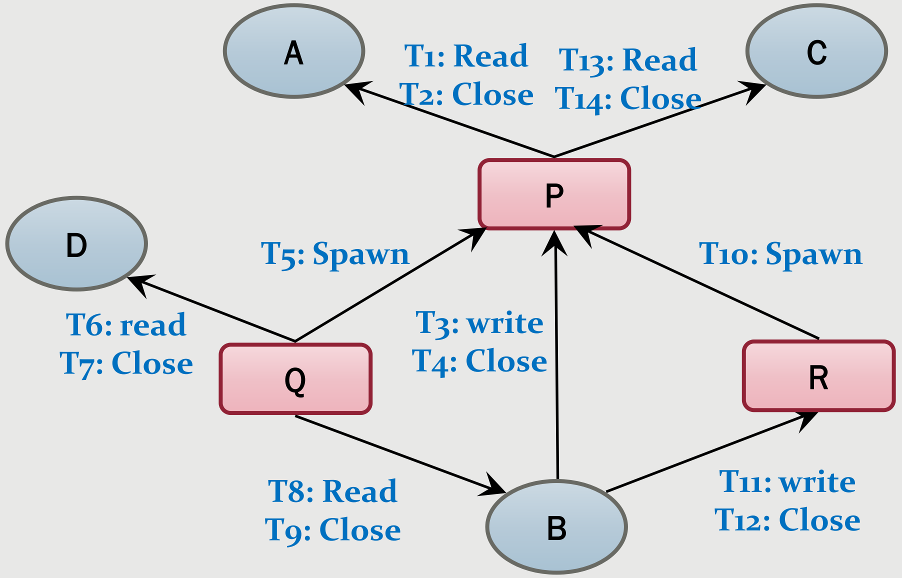
Mapping system events to a provenance graph



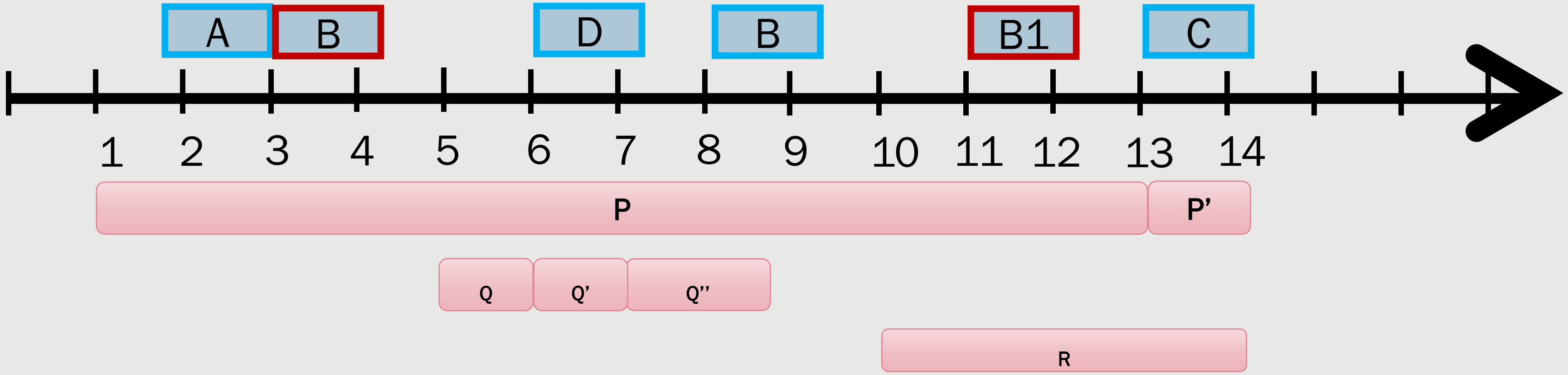


Z. Yuan, D.H. Ton That, S. Kothari, G. Fils, T. Malik. Utilizing Provenance in Reusable Research Objects, In *Special Issue on Using Computational Provenance*, MDPI Informatics, Vol 5(1), 2018.

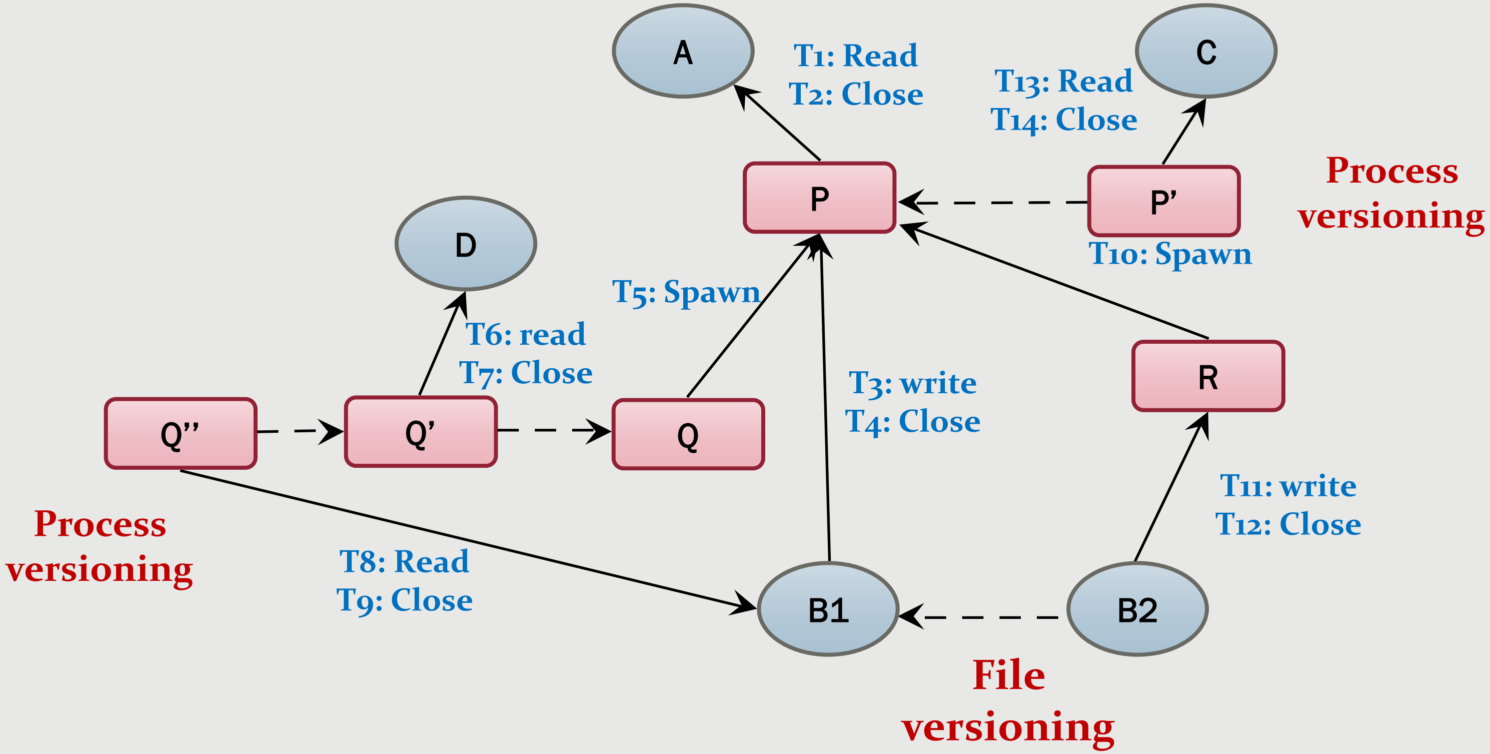
Versioning of process and file nodes



- ➢ Each file is versioned if it is written over
  - ➢ Write on a file changes file content
- ➢ Each process if versioned it it reads a new file
  - ➢ Read in a process changes process state



Inferencing causality of nodes



- If a file is read later, it is not causal on files written before.
  - B1 or B2 is not causal on C
- Changes to files can determine the causal graph that is impacted.
  - Changing A will impact all nodes, but the effect of C and D

Execution Trace

- Let P be a program
- A execution trace L for P is a 2-tuple < G, R >
- Provenance G = (V, E, T) with nodes V and edges $E \subseteq V \times V$.
  - Each node $v \in V$ and edge $e \in E$ has annotations
  - $T : E \longrightarrow T \times T$ is a function mapping edges to intervals from a discrete time domain T
- Package (R, <) in which elements $R \subseteq V$ are organized as a tree s.t $r_i \in R$ maps to a content in $v \in V$ and v has an outgoing edge

L is a repeatable iff

- L is deterministic, i.e., it will lead to the same result, $R = \{R_1, \ldots, R_n\}$, every time t’ > t
- L(G) includes necessary activities, entities and edges,
  - *it does not leave out activities, entities and edges in G(V,E,T) that may have caused R*
- L(R) is sufficient
  - *does not include superficial elements that do not cause R*

Q. Pham, T. Malik, B. Glavic, I.Foster. Light-weight Database Virtualization. In *IEEE International Conference on Data Engineering*, ICDE, 2015.

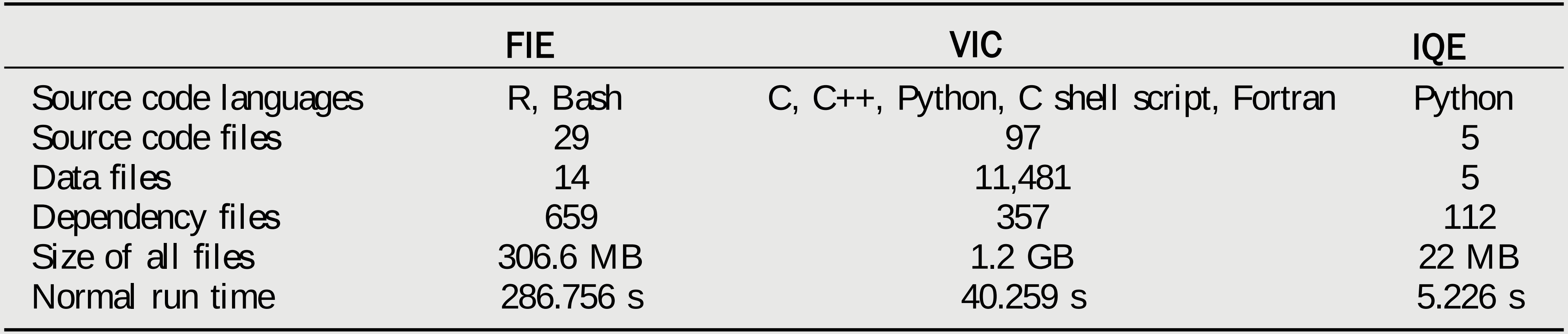
Evaluation

- Use cases:
  - FIE: Chicago Food Inspections Evaluation (~ 307 MB)
    - *A ML prediction model of food inspections*
  - VIC: Variable Infiltration Capacity (~ 1.2 GB)
    - *VIC: A Hydrology application*
  - IQE: Incremental Query Execution (~ 22 MB)
    - *A DB application with incremental query processing*
- Base Lines:
  - Docker
  - IncPy

Real Applications--Description

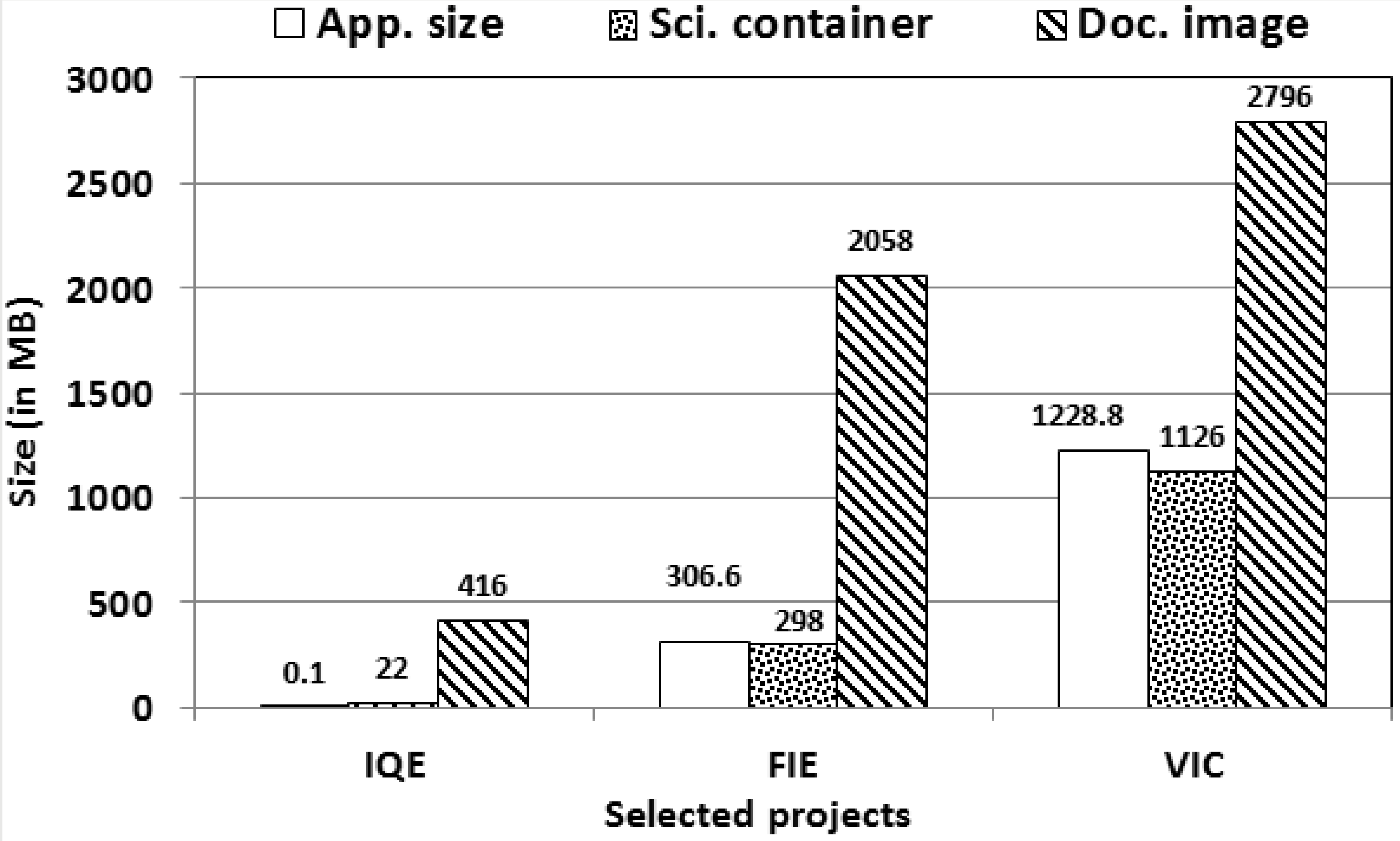
- FIE: A ML prediction model of food inspections
- VIC: A Hydrology application
- IQE: A DB application with incremental query processing

TABLE I: Usecases descriptions.



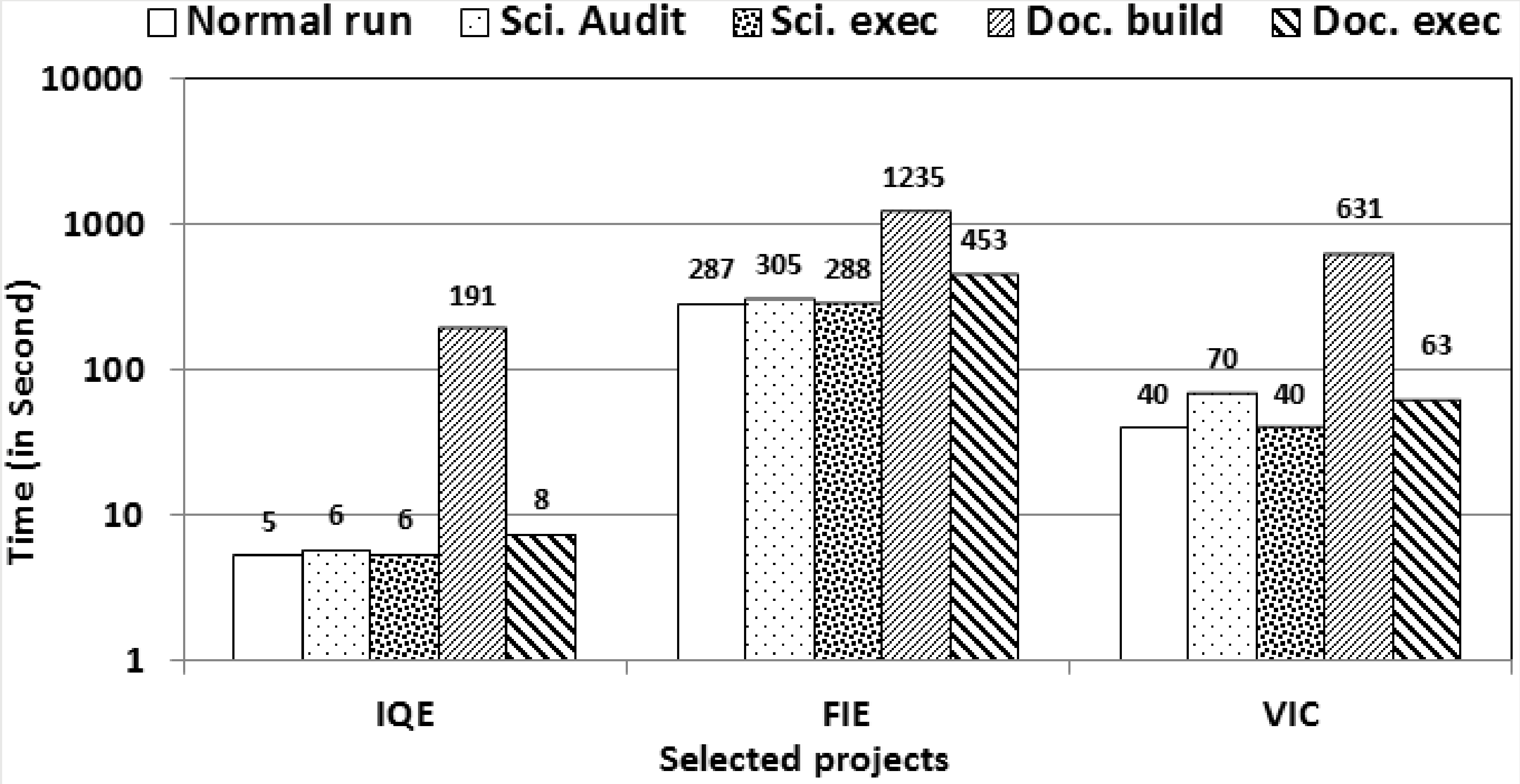
	FIE	VIC	IQE
Source code languages	R, Bash	C, C++, Python, C shell script, Fortran	Python
Source code files	29	97	5
Data files	14	11,481	5
Dependency files	659	357	112
Size of all files	306.6 MB	1.2 GB	22 MB
Normal run time	286.756 s	40.259 s	5.226 s

Container size



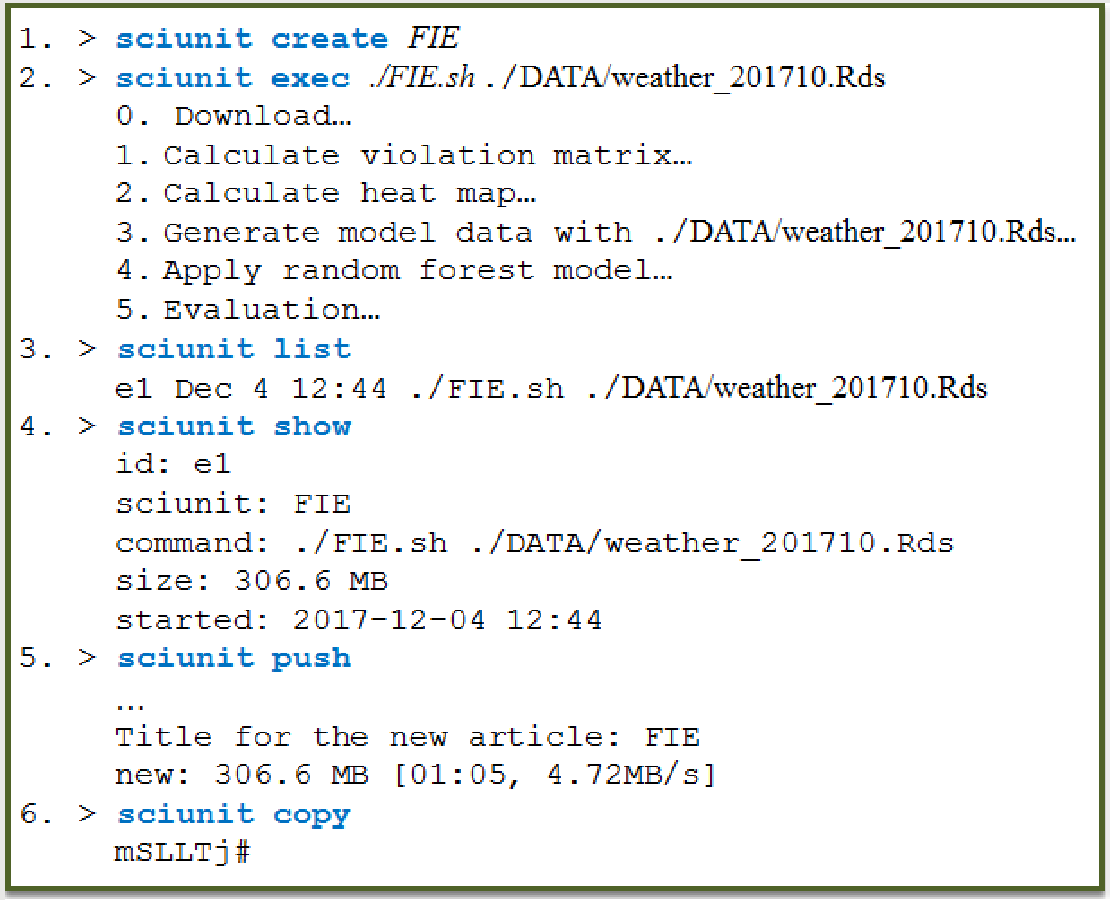
1. Docker containers are 19X, 7X and 2.5X larger than those of Sciunit.
2. Sciunit containers are even slightly smaller than the original application package size.

Auditing and re-execution time



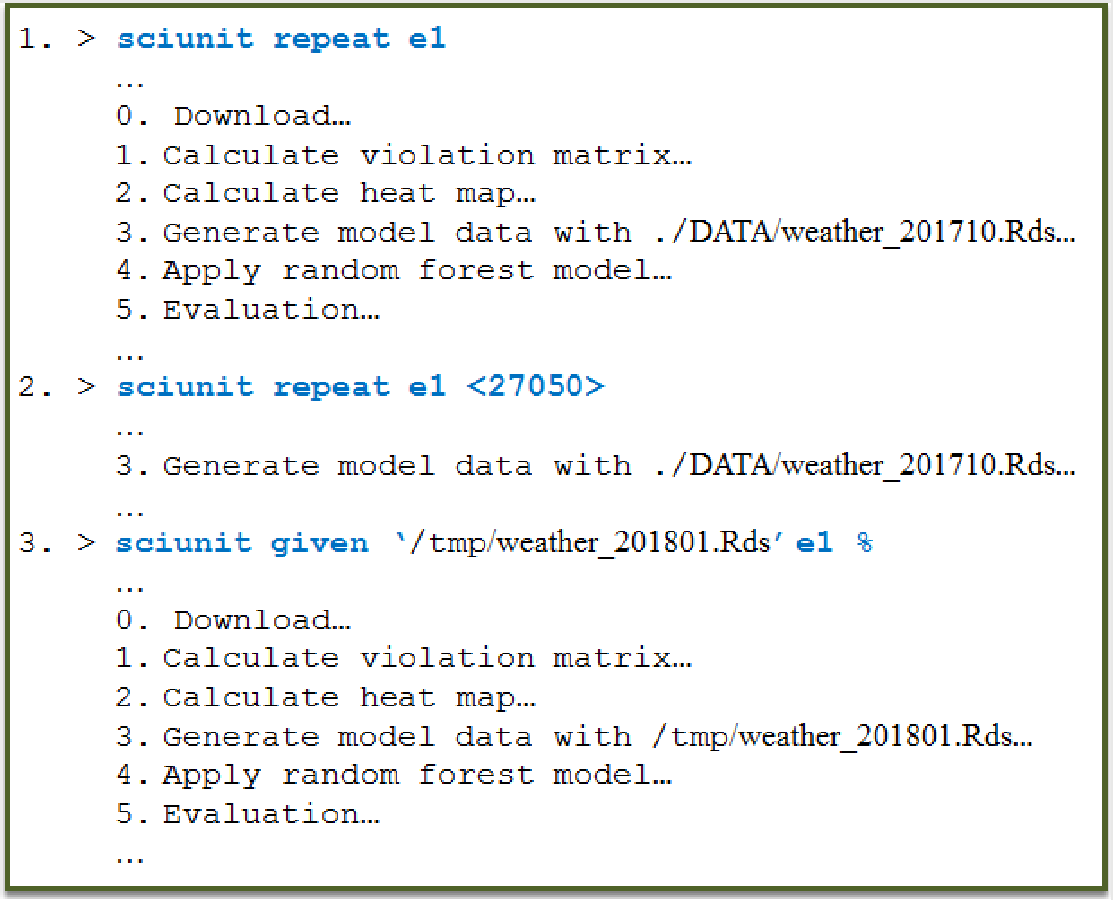
1. Docker spends 4X and 9X longer time than Sciunit to build FIE and VIC container, 7X and 2.5X larger.
2. Sciunit only spends slightly higher time than the original execution time to build containers.

Sample Interaction



```
1. > sciunit create FIE
2. > sciunit exec ./FIE.sh ./DATA/weather_201710.Rds
     0. Download…
     1. Calculate violation matrix…
     2. Calculate heat map…
     3. Generate model data with ./DATA/weather_201710.Rds…
     4. Apply random forest model…
     5. Evaluation…
3. > sciunit list
     e1 Dec 4 12:44 ./FIE.sh ./DATA/weather_201710.Rds
4. > sciunit show
     id: e1
     sciunit: FIE
     command: ./FIE.sh ./DATA/weather_201710.Rds
     size: 306.6 MB
     started: 2017-12-04 12:44
5. > sciunit push
     …
     Title for the new article: FIE
     new: 306.6 MB [01:05, 4.72MB/s]
6. > sciunit copy
     mSLLTj#
```

Alice’s Computer



```
1. > sciunit repeat e1
     …
     0. Download…
     1. Calculate violation matrix…
     2. Calculate heat map…
     3. Generate model data with ./DATA/weather_201710.Rds…
     4. Apply random forest model…
     5. Evaluation…
     …
2. > sciunit repeat e1 <27050>
     …
     3. Generate model data with ./DATA/weather_201710.Rds…
     …
3. > sciunit given ‘/tmp/weather_201801.Rds’ e1 %
     …
     0. Download…
     1. Calculate violation matrix…
     2. Calculate heat map…
     3. Generate model data with /tmp/weather_201801.Rds…
     4. Apply random forest model…
     5. Evaluation…
     …
```

Bob’s Computer

Summary

- FLINC: Notebook-based containerization
- CHEX: Notebook-based multiversion replay
- Kondo: Reducing the size of data-intensive containers.
- ProvScope: Container-based differencing
- Sched: Scheduling of application-virtualized containers.

- Please check out our work at:
  - *https://dice.cs.depaul.edu/*
  - *https://github.com/depaul-dice*

Thank You

- Please check out our work at:
  - *https://dice.cs.depaul.edu/*
  - *https://github.com/depaul-dice*

- Questions/Contact: tanu.malik@depaul.edu