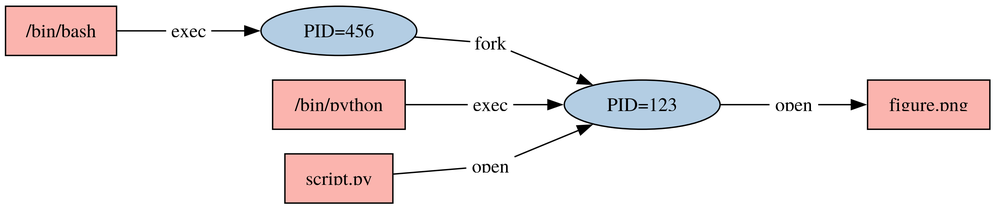
# How to collect computational provenance

Samuel Grayson, Reed Milewicz, Daniel S. Katz, Darko Marinov

## What is provenance?

The inputs (binaries, scripts, data) used to produce specific output

Can be collected *without* modifying programs




## Why provenance?

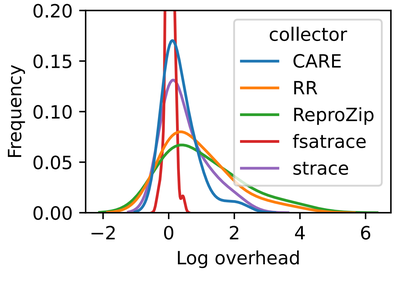
- Reproducibility: what inputs do you need to run this program?
- Caching: when inputs are changed, what outputs are stale
- Comprehension: what version of the data did this output use

## Methods for collecting provenance

| | Safe | Fast | Infallible | Rootless |
|---|---|---|---|---|
| Kern. mod | no | yes | yes | no |
| ptrace | yes | no | yes | yes |
| LD_PRELOAD | yes | yes | no | yes |
| eBPF | yes | yes | yes | no |

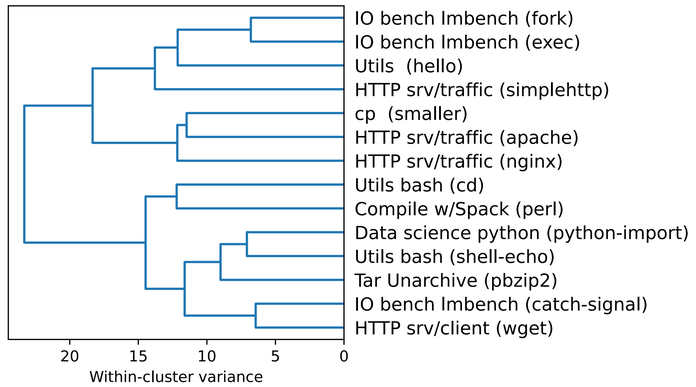
Ptrace is most studied, but LD_PRELOAD and eBPF are most compelling.

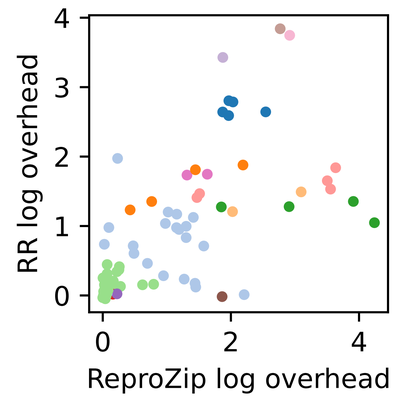
## Is it fast?




Depends on the application!

## How to make it faster?




| Collector | Intercept | IPC syscalls | socket syscalls | chdir syscalls | other syscalls |
|---|---|---|---|---|---|
| **CARE** | 0.6 | 670 | | 2426 | |
| **RR** | 6.5 | | 1144 | | 13568 |
| **ReproZip** | 5.8 | 2187 | 128 | | |
| **fsatrace** | 0.0 | | | | |
| **strace** | 1.7 | 678 | 143 | 1287 | |

## What next?

- Record/replay (get reproducibility "for free")
- Differential debugging
- Make without Makefile
- How to eliminate redundancies?