University of Basel

HPC Class - Chap 7 - Reproducilibity

Quentin Guilloteau[1]

[1]University of Basel, Switzerland

Quick poll (1/2)

Practices

- How are you plotting? Python/Gnuplot/R/...
- How are you running your experiments?
- How are you sharing code/data?
- How are you managing your software environments?
- Do you use notebooks? Jupyter/Rmarkdown/Orgmode/...

Quick poll (2/2)

Never have I ever...

- forgot which parameters I used for an experiments?
- forgot which version of the code I used?
- lost some scripts?
- forgot why I did something several months agos?
- failed to re-run something I did several months ago?

Introduction

The Reproducibility Crisis

- Baker [1], [2] raised awarness about the issue
- since then all the fields of science are getting concerned/interested in repro

What is "Reproducibility"?

- It depends who you ask .... No consensus between sciences
- In computer science: ACM gave some definitions [3]
  - *Repeatability* (Same team, same experimental setup)
  - *Reproducibility* (Different team, same experimental setup)
  - *Replicability* (Different team, different experimental setup)
- some sciences use also the term "Robustness"

What is the goal of Reproducibility?

- should not all research work be reproducible?
- is it a proof of the correctness of the work?
- restore trust in science
- or an effort so that future researchers can use the work?
  - *"Standing on the shoulders of giants"*
  - the objective: ✨ precise introduction of Variation ✨

Open Science and FAIR [4]

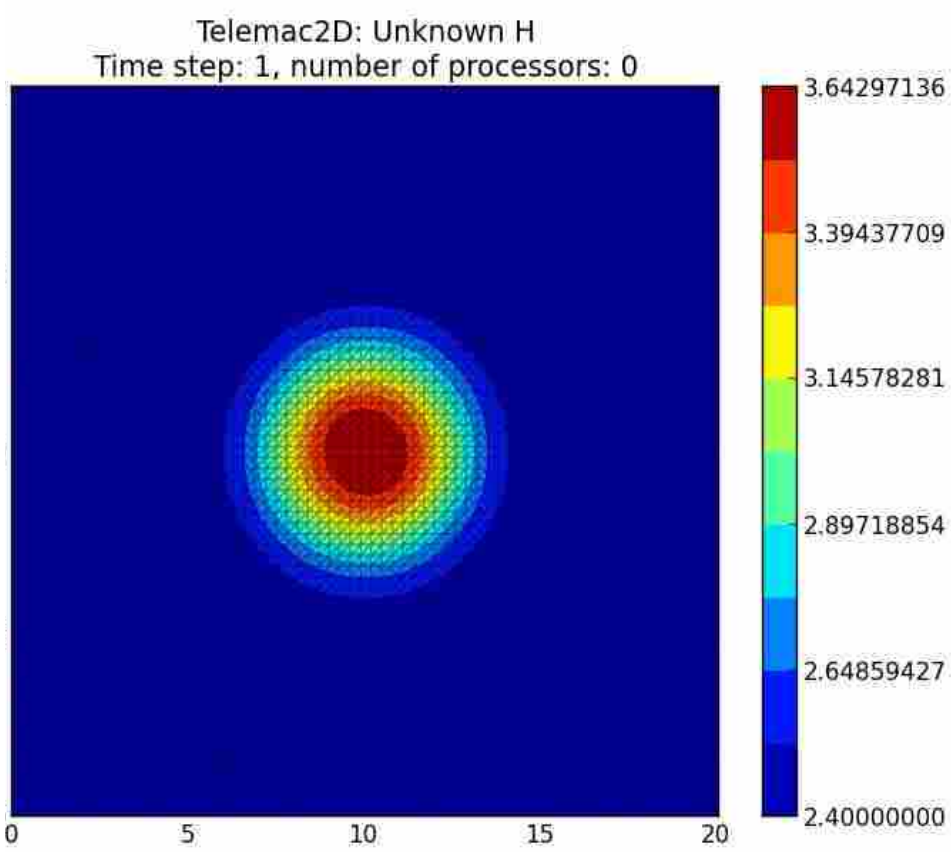
- **F** indable, **A** ccessible, **I** nteroperable, **R** eusable
- Open Science: one key concept is "**Transparency**"

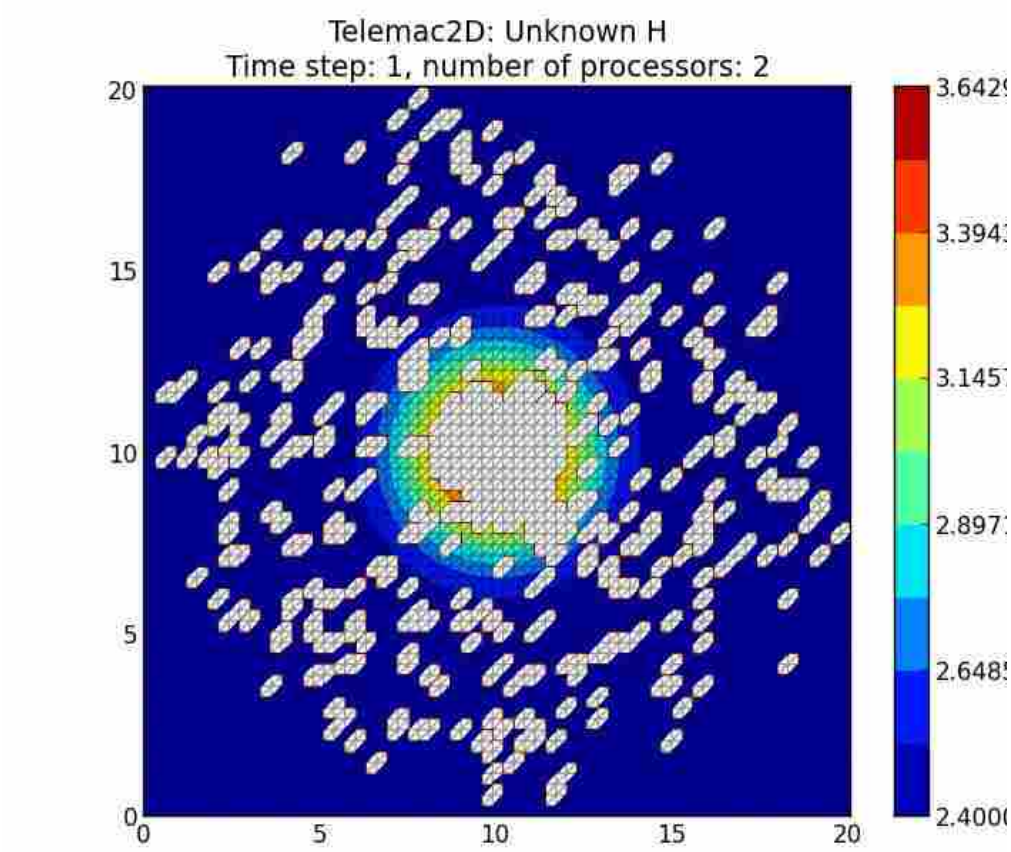
What about in Computer Science?

"Computers are deterministic, hence no reproducibility issues" – a fool

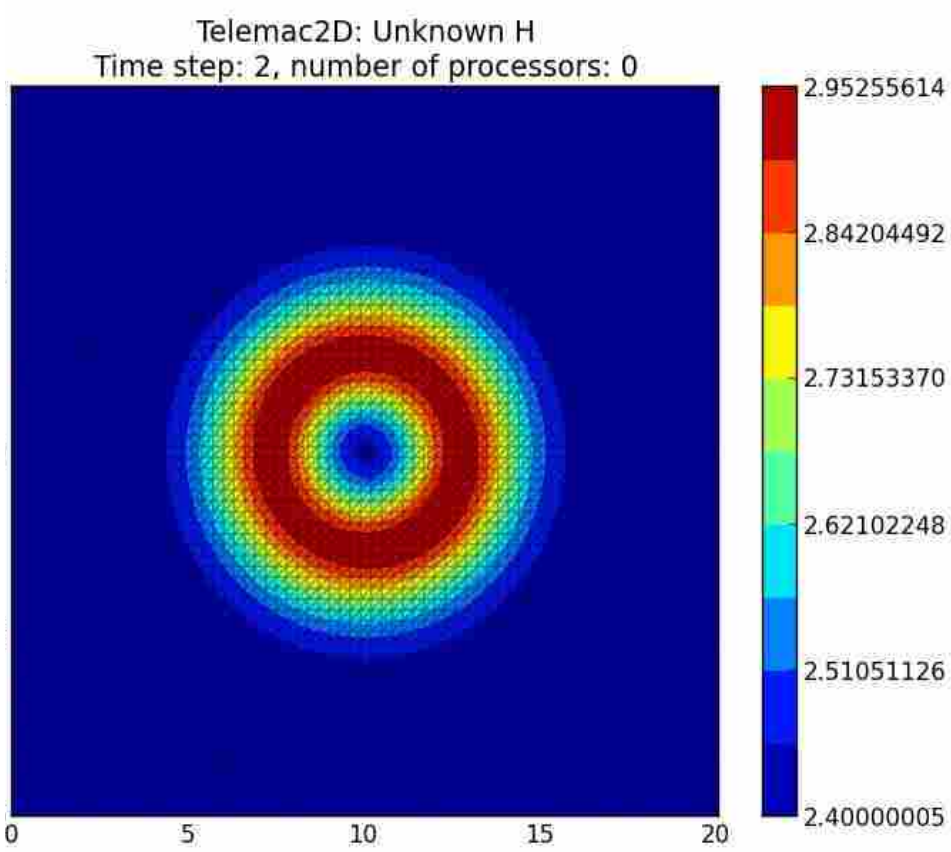
- floating point associativity ($\lfloor a + \lfloor b + c \rceil \rceil \neq \lfloor \lfloor a + b \rceil + c \rceil$)
- order of compilation flags has an impact on the performance
- the size of the environment variable has an impact on performance [5]
- computers are physical machines: heat, noise?! 📽️
- Experiments are *"quick"* and *"free"* ⟶ leads to some quick and dirty work
- Programming errors, mistakes (Excel 🤡 [6])
- How to compare 2 solutions?
- many more...

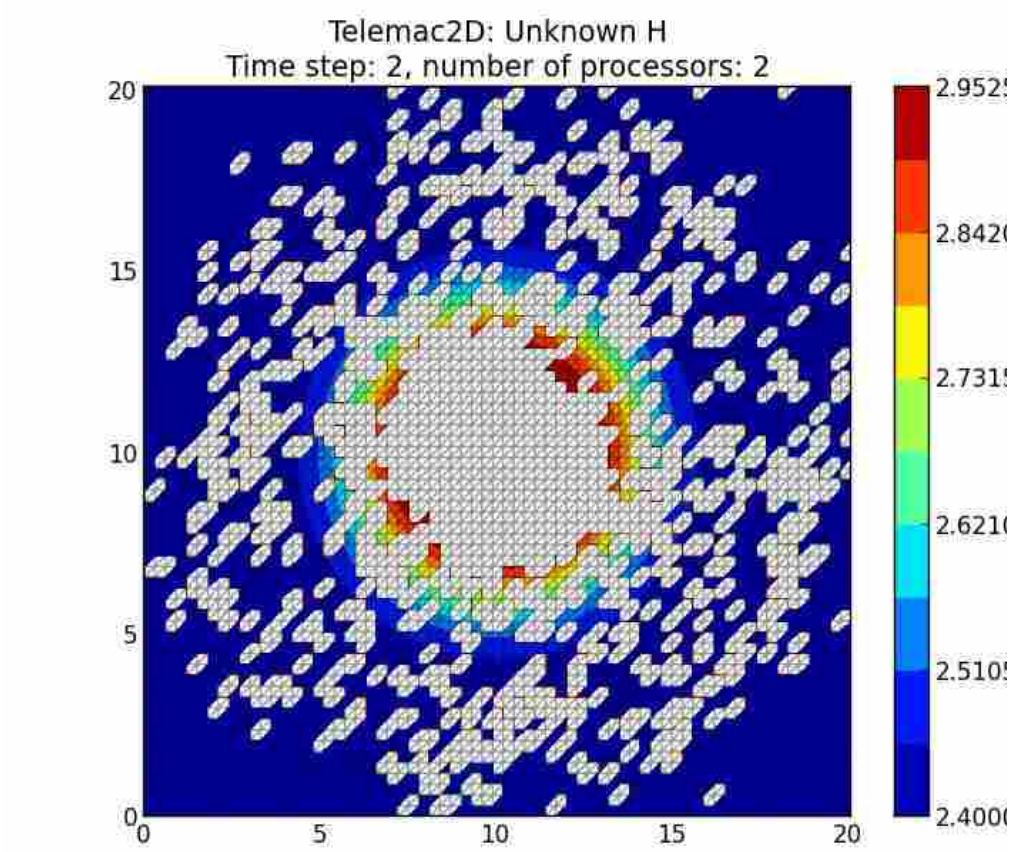
Simulation of a drop of water [7]



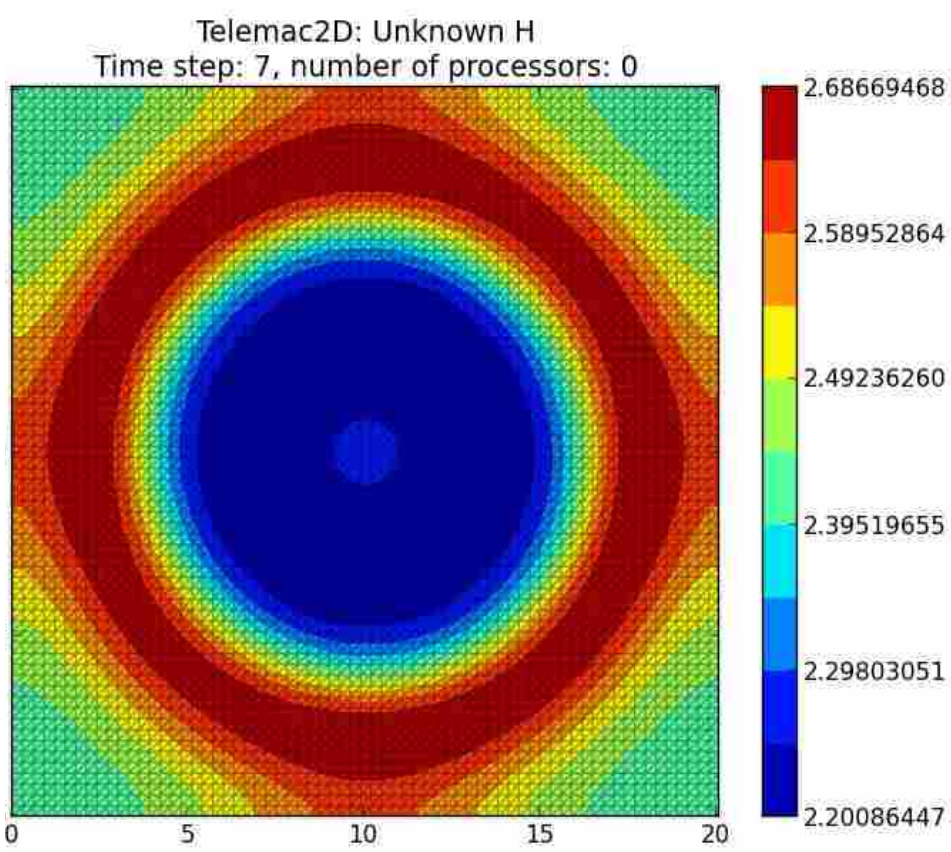


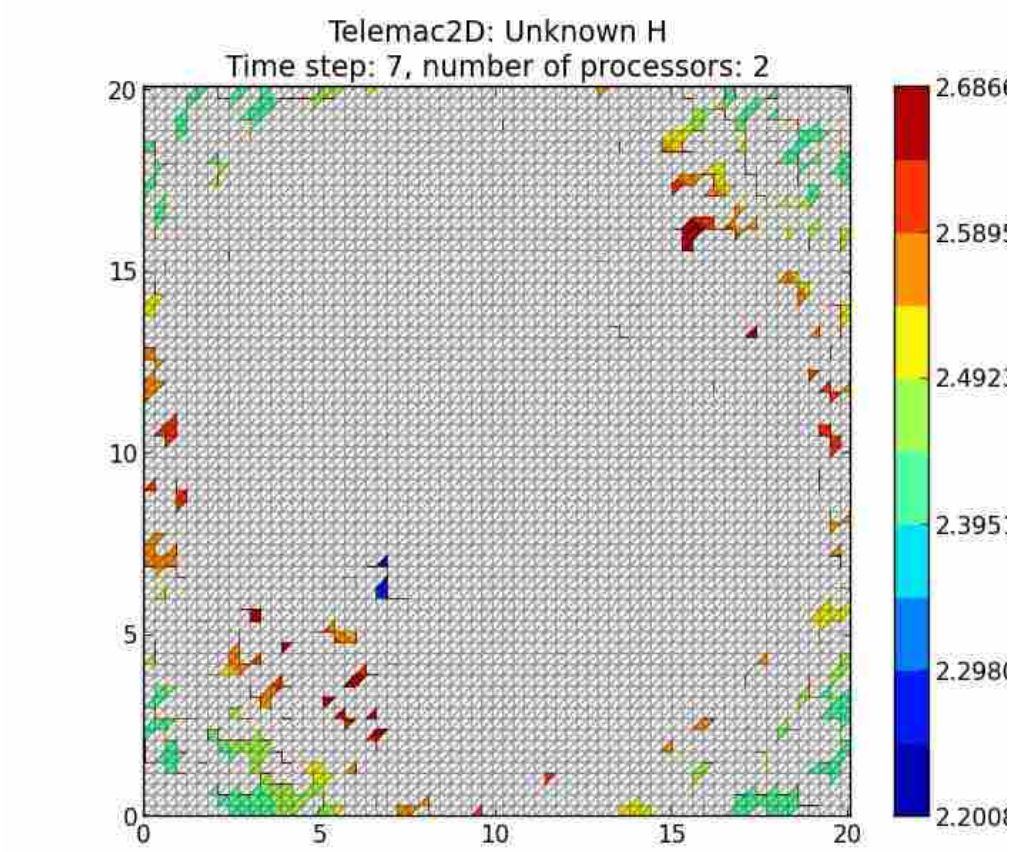
Simulation of a drop of water [7]



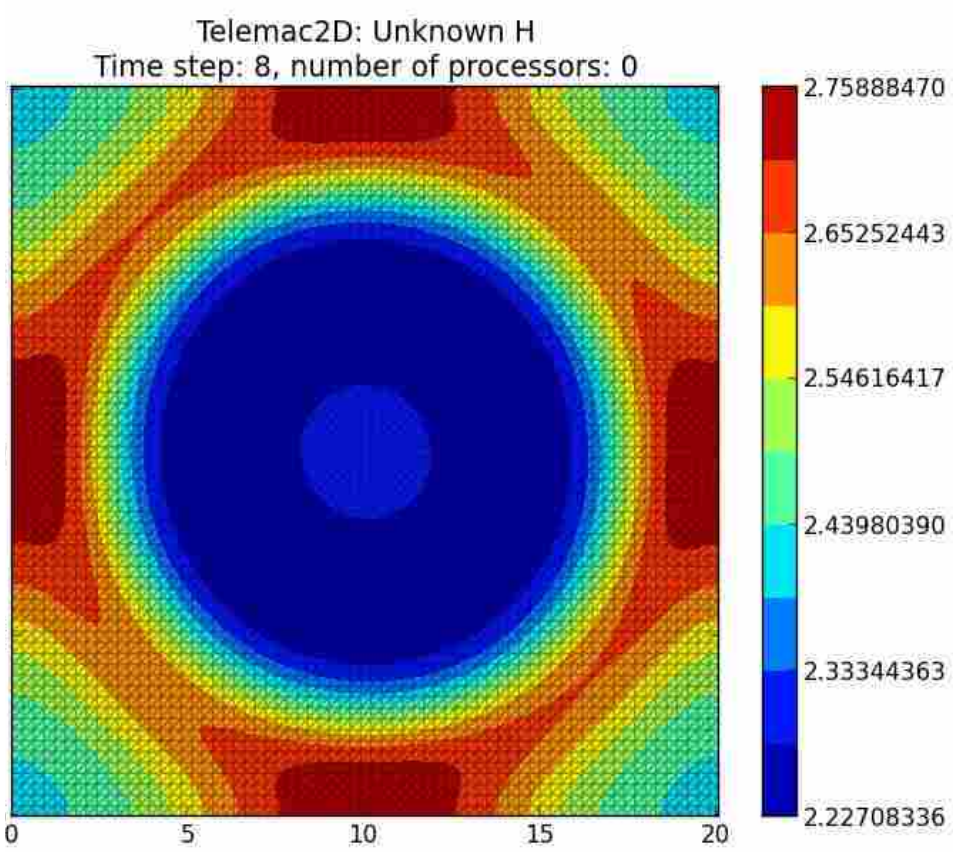


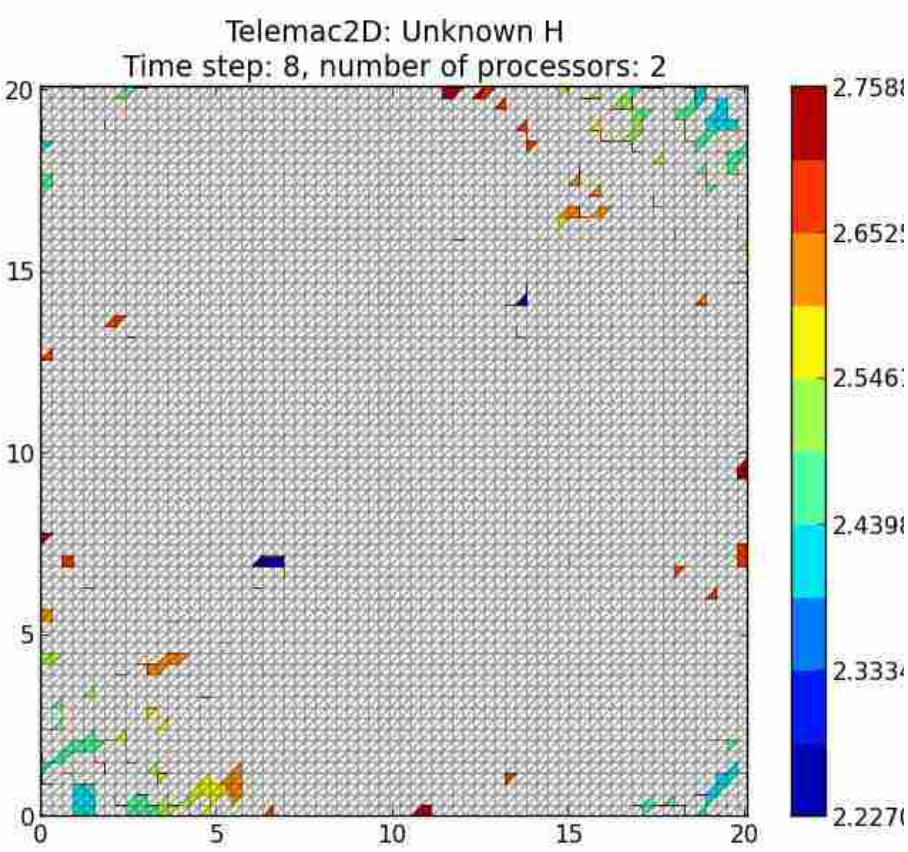
Simulation of a drop of water [7]





Simulation of a drop of water [7]



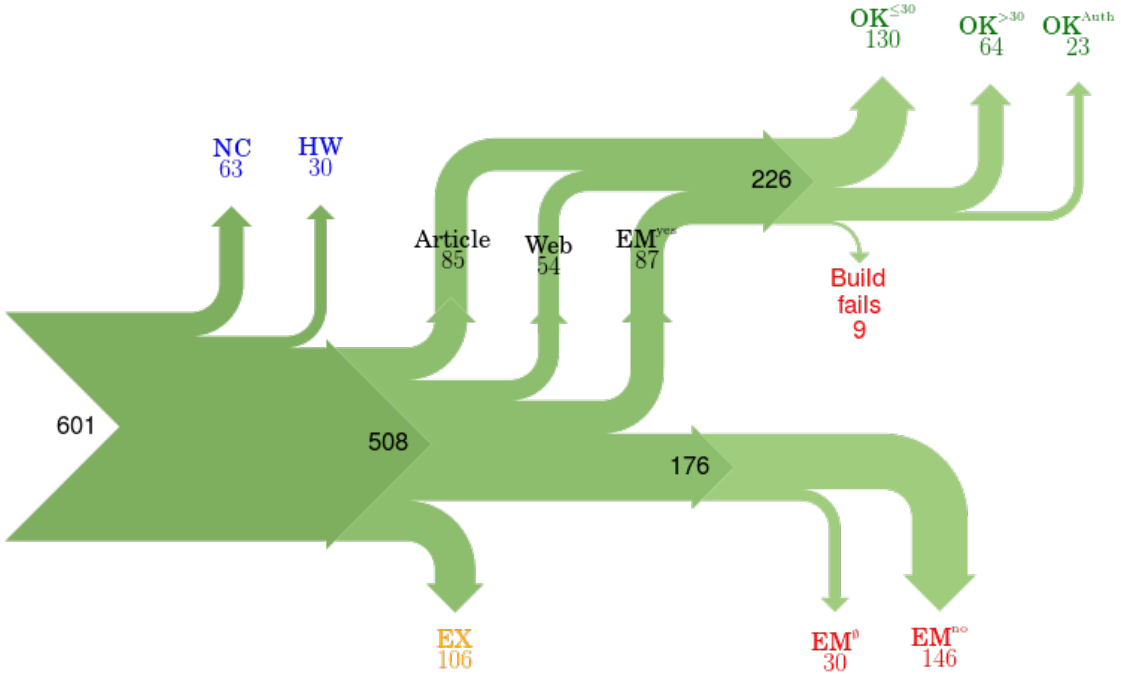


What does the community think?

- Hunold [8] : Surveyed the HPC community in 2015
  - 94% think the state of reproducibility needs to be improved
  - 58% think that 10% of the papers in HPC are reproducible
  - why not sharing code/data/scripts?
    - 84% to retain a “competitive” advantage
    - 87% it is not rewarding
    - 90% irrelevant because evolution is too fast
  - 42% think that their own work are mostly not reproducible

"My dog ate my code"

- Collberg et al. [9] (2015)
- looked at 600 papers and try to get the code and reproduce the results
- and contacted the authors
- funny responses from the authors



Some answers

"I am just hoping that it is a stable working version of the code, and matches the implementation we finally used for the paper. Unfortunately, ***I have lost some data when my laptop was stolen last year****. The bad news is that* ***the code is not commented and/or clean****. So, I cannot really guarantee that you will enjoy playing with it."*

"⟨STUDENT⟩was a graduate student in our program but ***he left*** *a while back so I am responding instead. For the paper we used a prototype that included many moving pieces that* ***only ⟨STUDENT⟩ knew how to operate*** *and we did not have the time to integrate them in a ready-to-share implementation before he left"*

"Most importantly, ***I do not have the bandwidth*** *to help anyone come up to speed on this stuff."*

Some recent findings

Artifact Evaluation

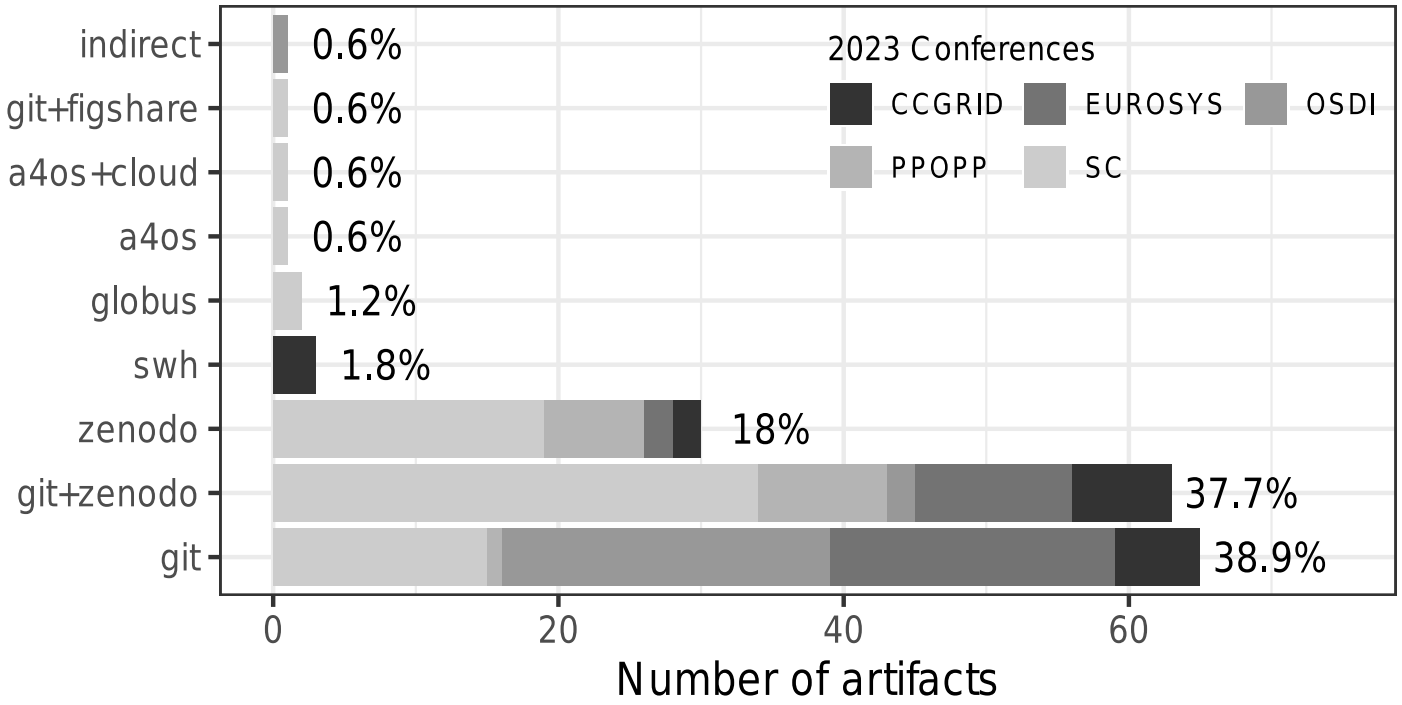
- Conferences and journals are now having an "Artifact Evaluation" process to make sure that the accepted papers are "reproducible"
- Authors send their code, scripts, data and a description of how to use it
- then reviewers will try to rerun experiments

- the goals:
  - validate results
  - restore trust
  - promote artifact sharing

How are authors sharing their artifacts?

- Guilloteau, Ciorba et al.
- Looked at 5 *top conferences* of 2023 in HPC/distributed systems
- 296 papers in total
- 157 had an artifact description
- 101 had the top badge
- 168 URLs to code in the papers
- 154 URLs valid at the time of the study

Sharing the code/data



- mostly just a Git(Hub|Lab) URL and/or a Zenodo archive

The worst case scenario

What if GitHub closes? What if the authors delete the repository?

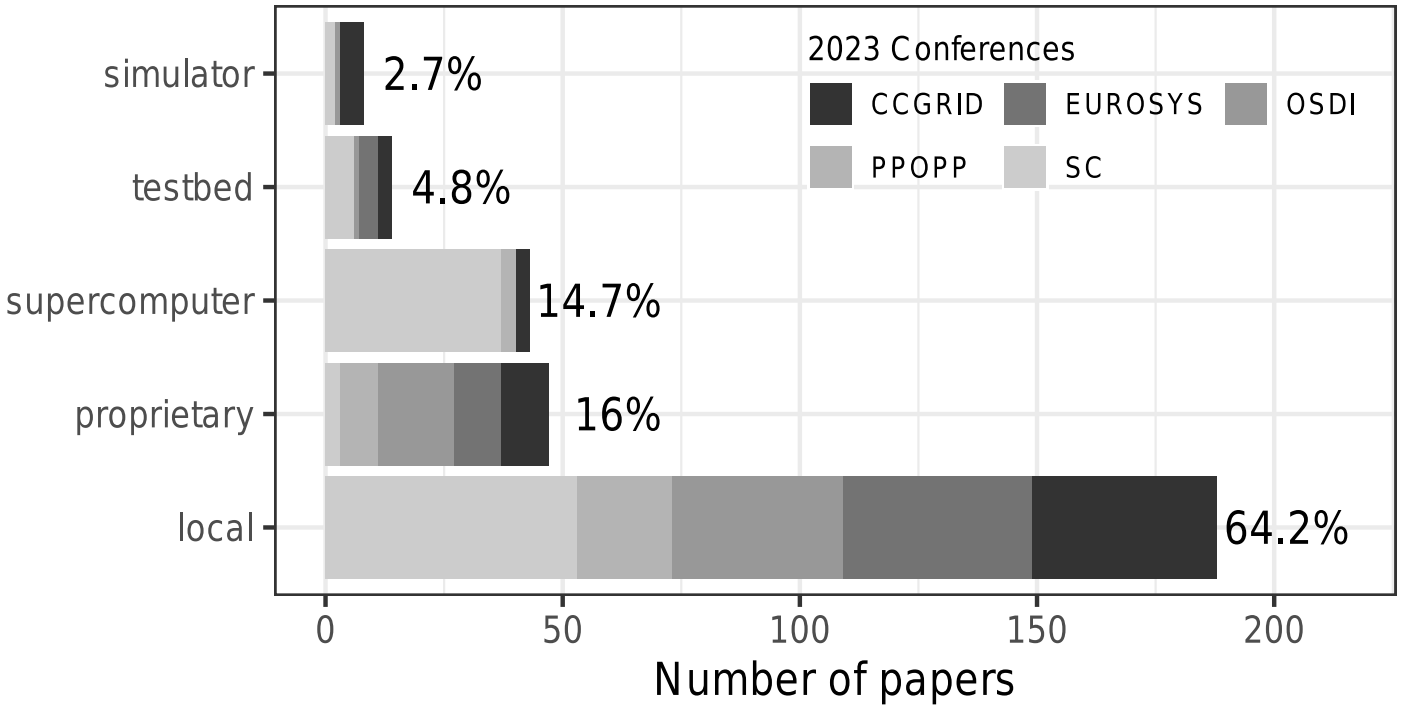
"Bah! GitHub will never close!" (😬 Google Code, Forge Inria)

Can you imagine if the proof of a math paper disappears? (Fermat 🙃)

Recommendation for long term archiving:

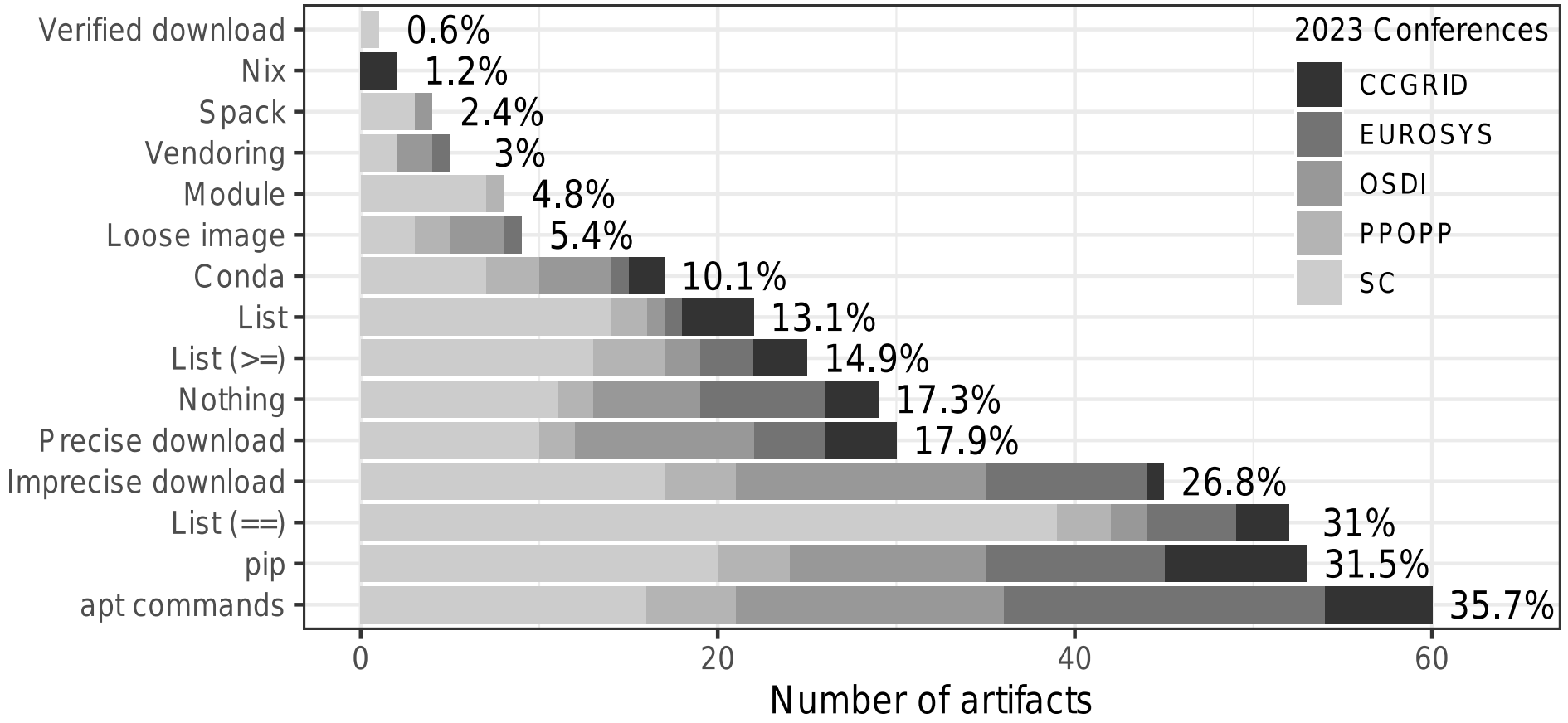
- For source code: Software Heritage (UNESCO) 
- For data: Zenodo (CERN)

Where are the experiments done?



- How to get access to **those** machines? → Azure/AWS/Google Cloud .... 💸

How is the software environment captured/described?



Some software environments problems

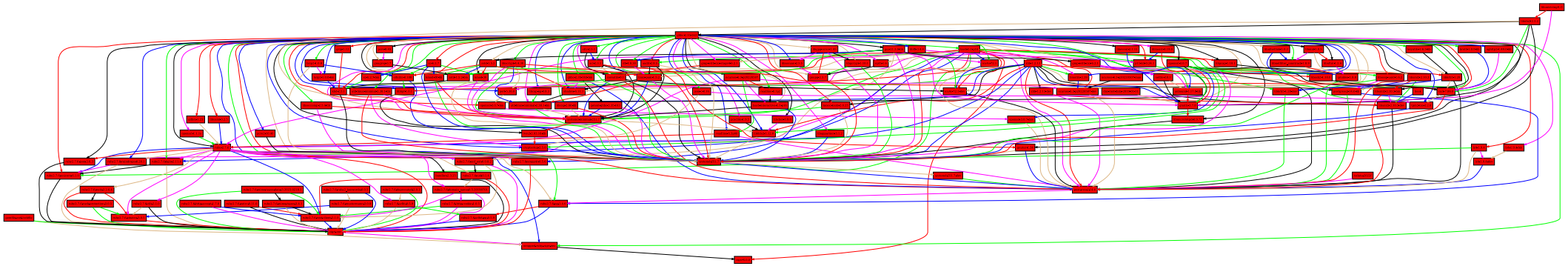
- **requirements.txt** → what about the **other** dependencies?
- List of packages (**apt install**) → what if forgotten packages? install **specific** version?
- Module → what is their lifetime? how to share? how to modify?
- Virtual Machines and Containers
  - Binary format → no transparency! → better: the recipe (e.g., Dockerfile)
  - Container → what about the OS/Kernel/Drivers ?
- Spack ? → better but not (totally) reproducible
- Perennity of the shared artifact?
- Non-free software?

What is wrong with this Dockerfile?

```
FROM ubuntu
RUN apt-get update -y && \
    apt-get install -y \
     build-essential\
     ...\
     simgrid
RUN git clone https://github.com/JohnDoe/chord.git
WORKDIR chord
RUN curl -L https://tinyurl.com/patchchord -o increase-timeout.patch
RUN git apply increase-timeout.patch
RUN cmake .
RUN make
ENTRYPOINT ["./chord"]
```

What a "complex" software environment looks like

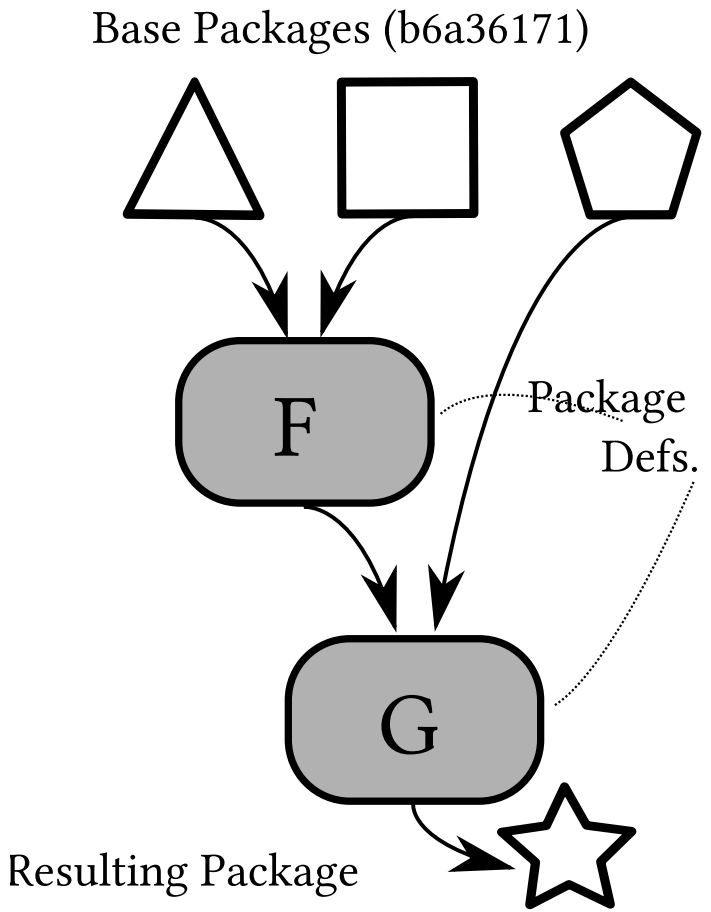
CiGri: a single component of the system during my PhD



How to manage such software environments in a *reproducible* way?

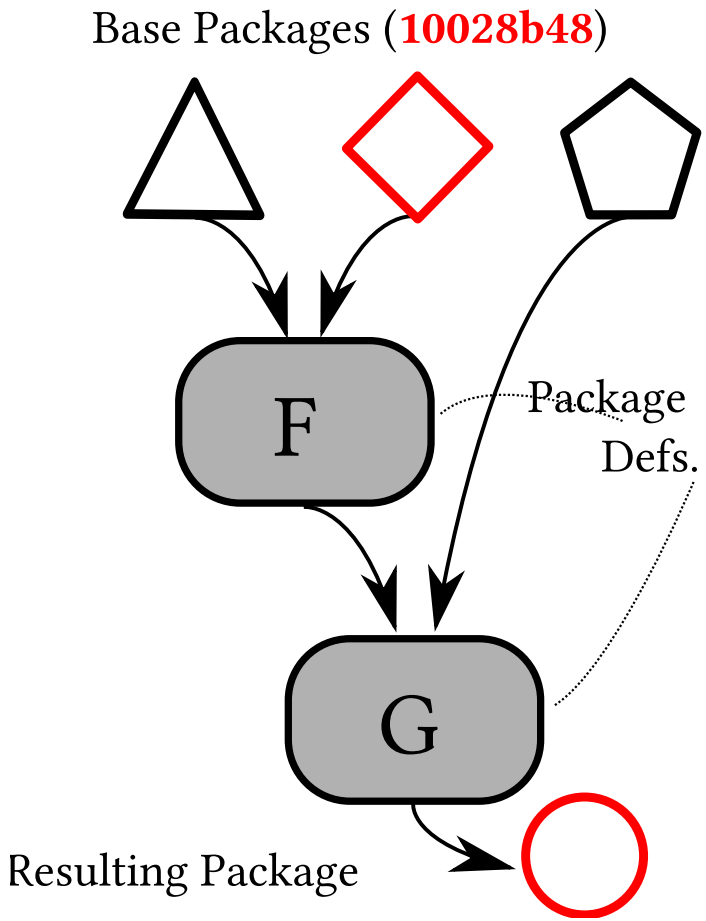
Functional Package Managers

- Nix [10], Guix [11]: reproducible by design!
- packages = functions
  - inputs = dependencies
  - body = commands to build the package
- sandbox, no side effect
- **/nix/store/hash(inputs)-my-pkg**
- immutable, read-only
- precise definition of $PATH
- can build: container, VM, etc.



Functional Package Managers

- Nix [10], Guix [11]: reproducible by design!
- packages = functions
  - inputs = dependencies
  - body = commands to build the package
- sandbox, no side effect
- **/nix/store/hash(inputs)-my-pkg**
- immutable, read-only
- precise definition of $PATH
- can build: container, VM, etc.



Workflow

Doing experiments

Never do experiments by hand

Doing experiments

NEVER do experiments by hand

Doing experiments

NEVER do experiments by hand

Doing experiments

NEVER do experiments by hand

State of the Practice

The state of the practice:

copy-paste commands from a readme or long bash scripts 😬

- error-prone
- not robust
- not extensible
- not scalable

Then what to use?

✨ Workflow Managers ✨

Workflow Managers

- For example: Snakemake [12], Nextflow [13]
- Like Make, but better
- express the workflow as **rule**s that take **input**s and give **output**s
- manages the dependencies
  - if an input is updated, will re-run everything that depended on this input, and so on
- great expressiveness
- scalable, integration with SLURM, K8S, etc.
- shareable (CWL)
- extendable

An example

Scenario

- parallel matrix multiplication code
- several input matrices
- several number of threads
- several scheduling strategies (LB4OMP)

```
rule expe:
  input:
    bin="my_mat_mult",
  output:
    "data/{matrix}/{sched}/{nb_threads}/{iter}.csv"
  shell:
    "OMP_SCHEDULE={wildcards.sched}\
     OMP_NUM_THREADS={wildcards.nb_threads}\
     {input.bin} {wildcards.matrix} > {output}"
```

→ “I want **data/webbase01/VISS/20/1.csv**”

Conclusion

Checklist for your future

- Use Git
- Do not be shy to share everything (code, data, metadata)
- Automatize as much as possible
- Pay attention to your plots 
- Pay attention to your statistical analysis
- Benchmarking crimes [14]
- Statistical dances 

Conclusion

- Reproducibility is gaining in importance in the scientific community
- Computer Science and HPC are also victims
- The state of the practice is not good enough!
- It is also **your** role to educate your peers on these problematics
- Machine Learning? 🎥

MOOCS:

- MOOC RR1 📚
- MOOC RR2 📚 (starting in a few days)

Conclusion

- Reproducibility is gaining in importance in the scientific community
- Computer Science and HPC are also victims
- The state of the practice is not good enough!
- It is also **your** role to educate your peers on these problematics
- Machine Learning? 🎥

MOOCS:

- MOOC RR1 📚
- MOOC RR2 📚 (starting in a few days)

What about the environmental cost of Reproducibility?

Next week (2nd of May)

Ch.08 Performance engineering by Jonas Müller Korndörfer

Bibliography

[1] M. Baker, "Reproducibility crisis," *Nature*, vol. 533, no. 26, pp. 353–366, 2016.

[2] M. Baker, "1,500 scientists lift the lid on reproducibility," *Nature*, vol. 533, no. 7604, pp. 452–454, May 2016, doi: 10.1038/533452a.

[3] ACM, "Artefact review badging."

[4] UNESCO, "Understanding open science." doi: 10.54677/UTCD9302.

[5] T. Mytkowicz, A. Diwan, M. Hauswirth, and P. F. Sweeney, "Producing wrong data without doing anything obviously wrong!," *ACM Sigplan Notices*, vol. 44, no. 3, pp. 265–276, 2009.

[6] D. Lewis, "Autocorrect errors in Excel still creating genomics headache.," *Nature*, 2021.

[7] R. Nheili, P. Langlois, and C. Denis, "First improvements toward a reproducible Telemac-2D," in *XXIIIrd TELEMAC-MASCARET User Conference*, 2016.

[8] S. Hunold, "A survey on reproducibility in parallel computing," *arXiv preprint arXiv:1511.04217*, 2015.

[9] C. Collberg, T. Proebsting, and A. M. Warren, "Repeatability and Benefaction in Computer Systems Research - A Study and a Modest Proposal," p. 68–69, 2015.

[10] E. Dolstra, M. de Jonge, and E. Visser, "Nix: A Safe and Policy-Free System for Software Deployment," p. 14–15, 2004.

[11] L. Courtès, "Functional Package Management with Guix," *arXiv:1305.4584 [cs]*, May 2013, Accessed: Jun. 13, 2020. [Online]. Available: http://arxiv.org/abs/1305.4584

[12] J. Köster and S. Rahmann, "Snakemake—a scalable bioinformatics workflow engine," *Bioinformatics*, vol. 28, no. 19, pp. 2520–2522, 2012.

[13] P. Di Tommaso, M. Chatzou, E. W. Floden, P. P. Barja, E. Palumbo, and C. Notredame, "Nextflow enables reproducible computational workflows," *Nature biotechnology*, vol. 35, no. 4, pp. 316–319, 2017.

[14] E. van der Kouwe, D. Andriesse, H. Bos, C. Giuffrida, and G. Heiser, "Benchmarking crimes: an emerging threat in systems security," *arXiv preprint arXiv:1801.02381*, 2018.