Considering the Development Workflow to Achieve Reproducibility with Variation

Michael Mercier[1,2] michael.mercier@inria.fr
Adrien Faure[1,2] adrien.faure@inria.fr
Olivier Richard[2] olivier.richard@inria.fr

Reproducibility with Variation

"If I have seen further it is by standing on the shoulders of Giants."

Feitelson taxonomy about reproducibility levels:

1. **Repetition**: Reproduce the exact same experiment with the same inputs
2. **Replication**: Reproduce the same experiment with different inputs
3. **Variation**: Reproduce the experiment with variation

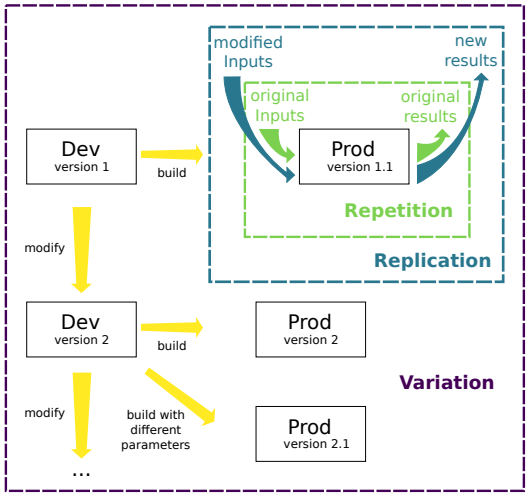
Why we should ease reproducibility with **Variation**?

- Enable others (even yourself!) to continue your work
- Enable collaboration
- Test new variations, add more parameters, and develop new features

HPC experimentation is based on software
=> Variation implies to modify the experiment software environment

Experiment development workflow

Development and Production environments



Software environment definition:

- set of executables and scripts
- with their dependencies (libraries, interpreters)
- and their configuration (environment variables, configuration file, access keys, ...)

Repetition and **Replication**

=> Only 1 **Static** environment

Variation

- *Dynamic* development environment
- *Dynamic* production environment

Development workflow that enables Variation

Requirements

1. Reconstructible environments
2. Exhaustive content definition of Development and Production environments
3. Enable environment reconstruction with modification
4. Enable environment sharing

Documentation of the environments is not enough:

- Not capturing hidden dependencies
- Not runnable: may drift from reality

Containers does not fit requirements:

- Container build process is not deterministic
- No complete definition of what's inside

The functional package managers approach

a.k.a. FPM

General purpose package managers

A package definition is a function
=> dependencies in input
=> package in output

The packages set is defined as a libraries

- Nix
  - since: 2005
  - language: Nix expression
  - packages: 40 000+
- Guix
  - since: 2012
  - language: GNU Guile (Scheme dialect)
  - packages: 7800+

Properties

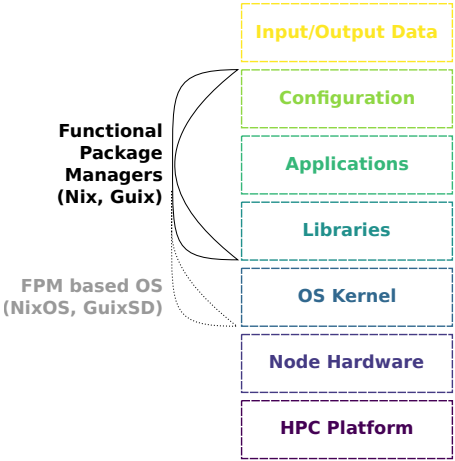
- Possibility to define software environment
- Environment are installable and runnable
- Unprivileged user build and installation
- Transparent source/binary installation (allows custom packages)
- Fine grain sharing of software pieces
- Not side effect effects install (rollbacks)

How FPM fits requirements

Requirements for Variation

1. Reconstructible environments
   => every dependencies are fixed
   => build happen in a sandbox (no hidden dependencies, every artefact is cached)
2. Exhaustive content definition of Development and Production environments
   => everything is described in Nix expression language
   => package definition defines build environment (for free!)
3. Enable environment reconstruction with modification
   => environment as code: easy to modify and versioned
   => binary cache avoid to reconstruct what's not necessary
4. Enable environment sharing
   => share package definition + binary closure
   => closure import/export with Nix

FPM scope and limitations



Approaches for the other requirements of Variation

- Experiment input/output management
  - capturing: Sumatra
  - storing: GitLFS, Git Annexe, ...
- No Kernel level management
  - NixOS, GuixSD, Kameleon
- Specific Hardware requirement
  - Grid5000, Chameleon

Conclusion and Discussion

Summary

1. HPC community should aim at reproducibility with variation
2. Variation requires to take into account development workflow
3. Proposed approach: use functional package manager (FPM)
4. FPM are hard to use, but it is worthwhile!

Steep learning curve

- Functional language not common in HPC community
- FPM paradigm are very different from classic package managers

Nix good properties

- Environment portability
- Fine grain dependencies management
- Built-in caching of the artefact enforce recontructiblility

We need environment persistence

- source and binary (closure) backup
- => Internet Archive, Software Heritage

Considering the Development Workflow to Achieve Reproducibility with Variation

Michael Mercier[1,2] michael.mercier@inria.fr
Adrien Faure[1,2] adrien.faure@inria.fr
Olivier Richard[2] olivier.richard@inria.fr