Guix@GRICAD

Software packaging and deployment at GriCAD

Exa-DI Annuel Meeting - 24/02/2026

P.-A. Bouttier

TOC

1. A bit of history
2. Guix for users
3. Guix for operators

TOC

1 A bit of history

2 Guix for users

3 Guix for operators

GRICAD supercomputing context

- 4 compute clusters in-house (over 25 000 CPU cores, approximately 100 GPUs)
- …connected to each other and to lab clusters via a local computing grid
- Uses:
  - HPC, HTC
  - Data processing
  - Visualization, training & development
- **High heterogeneity** in use cases, communities, user skill levels, and consequently, software stacks
- **The challenge**: finding a software environment management tool that meets very heterogeneous constraints

Open Science context

- Open Science : How to ensure reproducibility of digital processing?
    - Good software development practices
    - And the software environment?
- Practice changes
    - Increasing use of non-HPC platforms: JupyterHub, BinderHub, cloud computing...
    - ...within the same processing chain
    - Development of distributed compute/data platforms

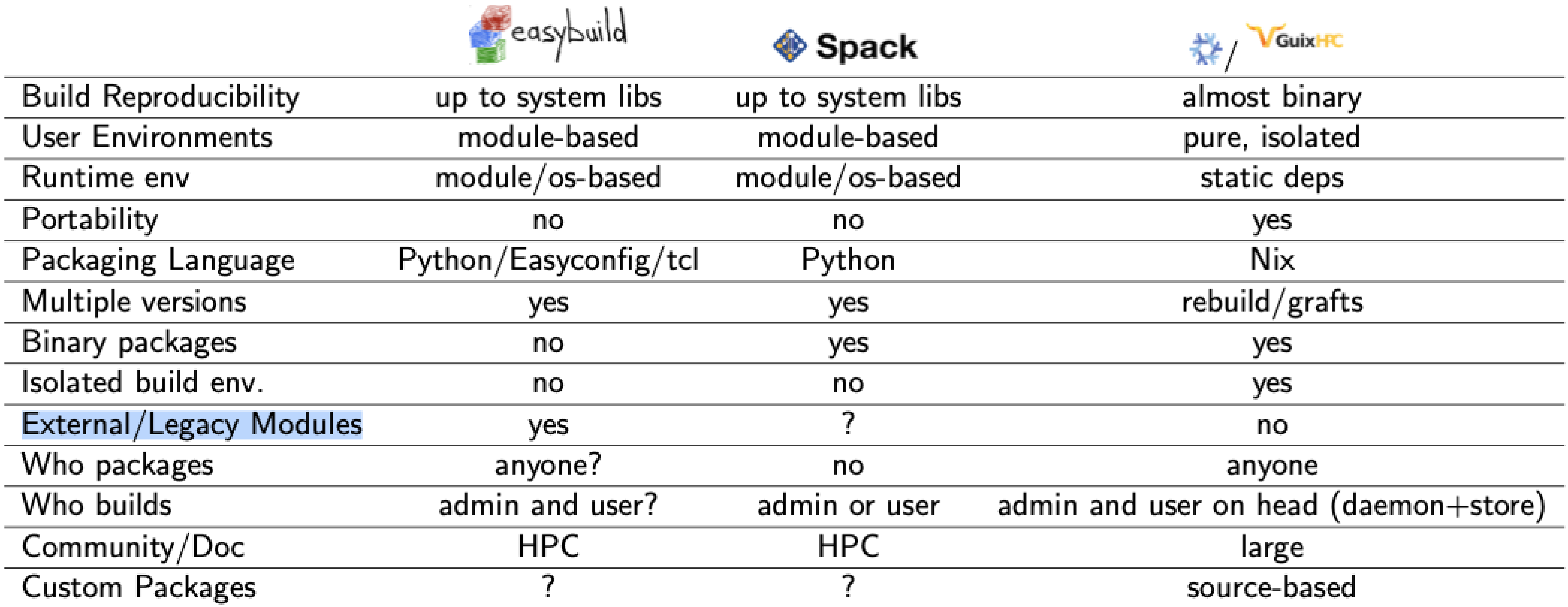
The Good Ol' days

- Until 2015, use of the module system
  - Classic, well-known
  - Used at tier-1 and tier-2
  - Usages and communities (many) were more homogeneous
- But issues started to surface:
  - **Very average portability** (essential for our small grid!)
  - **Duplication of effort** proportional to the number of clusters
  - **Marginal community aspect** (e.g. no direct sharing of packages, binaries)
  - ...

The utopia of software environments?

- **Isolation from the host system**, at execution AND during construction
- **Maintenance** (well-managed dependency tree), **reproducibility** (unique package version - src, compile, desc, defs,... - same output everywhere), **portability**
- **Completely functional in user space**
- **Automated workflow**: custom packages, CI/CD and automatic rebuilds, from personal PC to clusters
- **OS-independent**

Welcome to the Jungle



	easybuild	Spack	/ GuixHPC
Build Reproducibility	up to system libs	up to system libs	almost binary
User Environments	module-based	module-based	pure, isolated
Runtime env	module/os-based	module/os-based	static deps
Portability	no	no	yes
Packaging Language	Python/Easyconfig/tcl	Python	Nix
Multiple versions	yes	yes	rebuild/grafts
Binary packages	no	yes	yes
Isolated build env.	no	no	yes
External/Legacy Modules	yes	?	no
Who packages	anyone?	no	anyone
Who builds	admin and user?	admin or user	admin and user on head (daemon+store)
Community/Doc	HPC	HPC	large
Custom Packages	?	?	source-based

Our final choices

- Installation and deployment of Nix in 2015, Guix in 2018
- Today, Nix and Guix are available, with some modules for certain users

TOC

1. A bit of history
2. **Guix for users**
3. Guix for operators

Guix on the user side, Pros and Cons (1/2)

Cons

- A new tool: requires support (documentation, assistance)
- Support for changes in practices (e.g. detaching from version numbers)
- Some important software stacks still need to be packaged: Intel OneApi, AI frameworks => Nix to the rescue

Guix on the user side, Pros and Cons (2/2)

Pros

- Reproducible and therefore truly portable (unlike Spack or Conda)
- Very practical for development environments (like `venv`, but for everything)
- Easy-to-use Guix commands
- Large package base (between 20,000 and 50,000 if you add specific channels)
- User autonomous (if package exists)

TOC

1. A bit of history
2. Guix for users
3. Guix for operators

Guix on the support team side, Pros and Cons (1/2)

Cons

- Steep learning curve for packaging
- Some software, especially proprietary software, is instrinsicly difficult to package

Guix on the support team side, Pros and Cons (2/2)

Pros

- Greater user autonomy
- A much more portable deployment solution
- Community aspect:
    - sharing packages, recipes, problem-solving
    - Shared documentation
    - Support for support (mailing lists, dedicated Mattermost, Café Guix, Guix-HPC days, etc.)
    - Much more exciting!

Thank you all for your attention!