Building and installing ABINIT

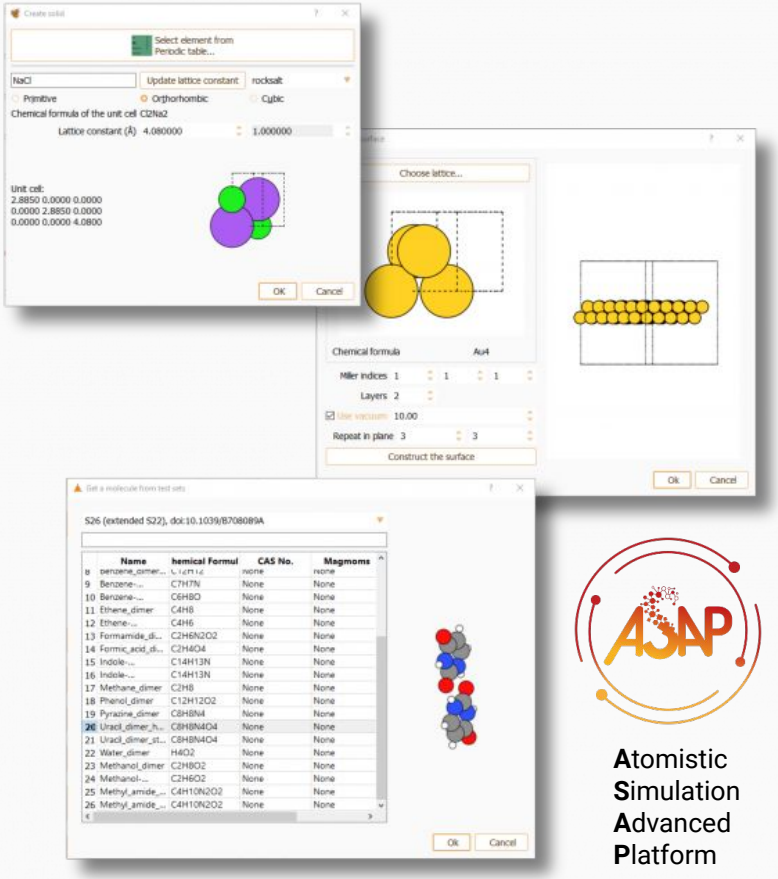
A matter of collaborations

y.pouillon@simuneatomistics.com

SIMUNE ATOMISTICS

Donostia-San Sebastián, Spain
https://www.simuneatomistics.com/
info@simuneatomistics.com

- Make DFT software usable by industry
- Contribute directly to open-source packages
- Train researchers & engineers in atomistic simulations
- Create synergies between universities and companies



Atomistic
Simulation
Advanced
Platform

We hire!

Simune is looking for a developer,
If interested please contact:

careers@simuneatomistics.com

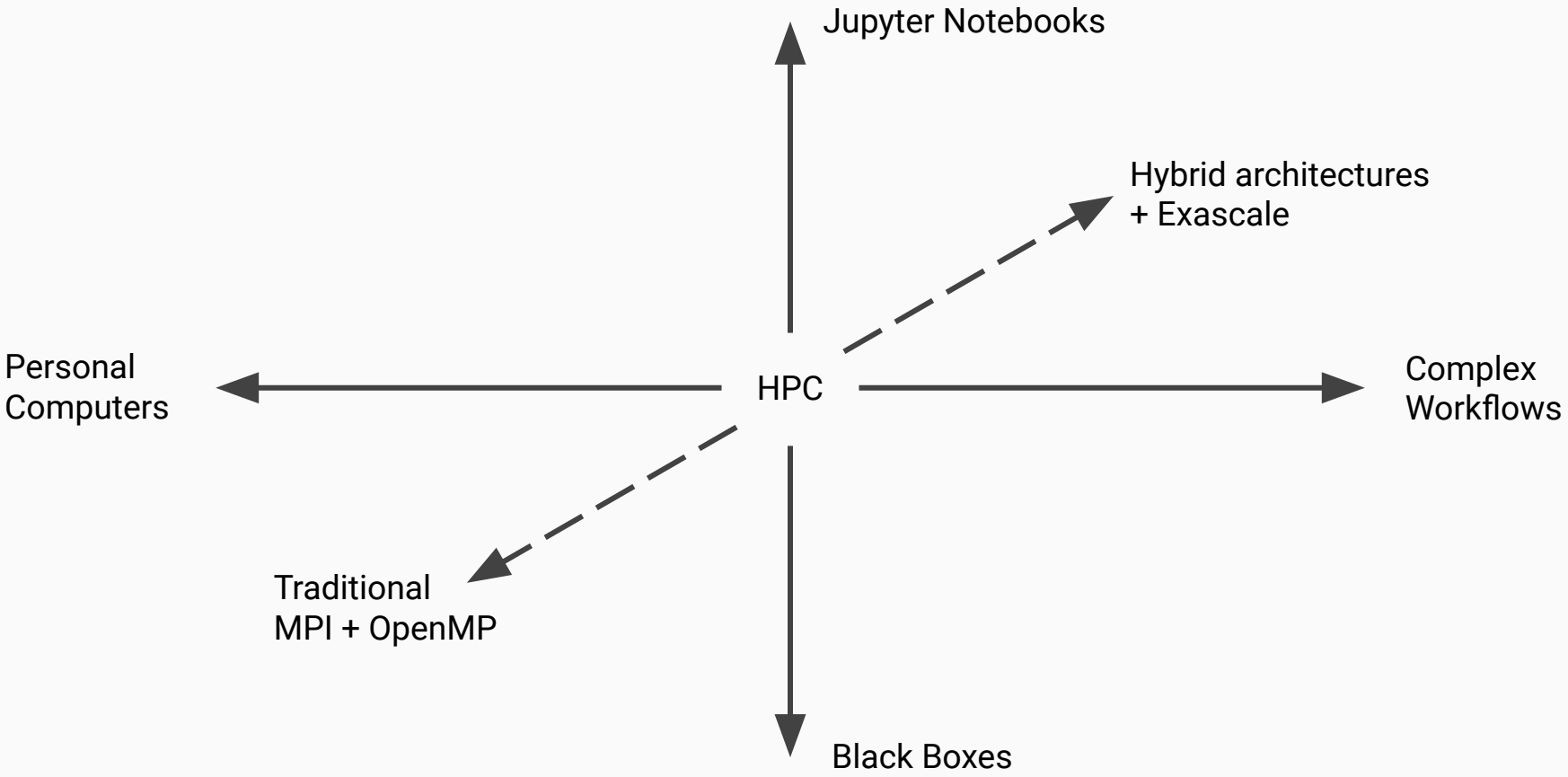
ABINIT over time, seen from the build system

2004-2013: one-dimensional polarisation

Personal Computers ⟷ HPC

Focus on compilers, automation, portability

2013-Now: Multidimensional polarisation



Impact on the build system

Requirements: support at the same time …

- Beginner end-users with laptops and highly-automated frameworks with server farms
- Sliced calculations with high verbosity and wrapped highly-optimised black-box runs
- Traditional parallelism and mixes of OpenMP, MPI, GPUs, FPGAs, …, on new architectures

+ A myriad of intermediate cases
+ Remember the developers!

Addressing the combinatorial explosion

- Minimum number of generic user interfaces for build systems
- Architectural layering: low-level and shareable vs. high-level and specific
- Higher modularity: libraries, source code, data flow
- Comprehensive automation strategies
- Multiple paths to multiple solutions: fallbacks, ESL Bundle, EasyBuild, Spack, …

Common denominator

Must be based on

multilateral

COLLABORATIONS

Let’s build ABINIT!

Who builds ABINIT?

Profiles

- End users
- Developers
- Maintainers & Testers
- Scripts & automated frameworks

Sectors

- Education (Students & Teachers)
- HPC
- Academic Research
- Industrial Research + R&D

Workflow

- Autotools trilogy: *configure, make, make install*
- Optionally: *make check* before *make install*
- Most critical step: configure
  - Adapt the build to the user's computer
  - Help beginners without hindering experts
  - Provide hints when something goes wrong
  - Communication through command-line options
  - Quick-and-dirty help with dependencies: ABINIT Fallbacks

Configure options (details on Friday)

- Conventions
  - *--enable-feature*: internal switches of ABINIT, only *yes* or *no*
  - *--with-package*: external dependencies and their behaviors, can be *yes*, *no*, or a path

- Linear algebra
  - *--with-linalg*: where to look for libraries, or let the build system decide if omitted
  - *--with-linalg-flavor*: what kind of libraries to look for, e.g. Netlib or MKL
  - If not sufficient, can be overridden by e.g. LINALG_LIBS="..."
  - Interacts with *--with-mpi* option (automated): decide whether to look for ScaLAPACK
  - Influences which values *--with-fft-flavor* can take
  - Internally: complex heuristic to cover as many configurations as possible

Collaboration between expertise levels

- Beginner
  - Run configure without command-line option
  - Gather hints for missing dependencies
  - Get help from more expert users (direct, forum, documentation)
  - Use documented configuration template: *~abinit/doc/build/config-template.ac9*
- Intermediate
  - Use command-line options to tune *configure*
  - Fine-tune config file with experts (direct, forum, documentation)
  - Help beginners (direct, forum, documentation)
- Expert
  - Use environment variables to tune *configure*
  - Help intermediate users/developers and beginners (direct, forum, documentation)

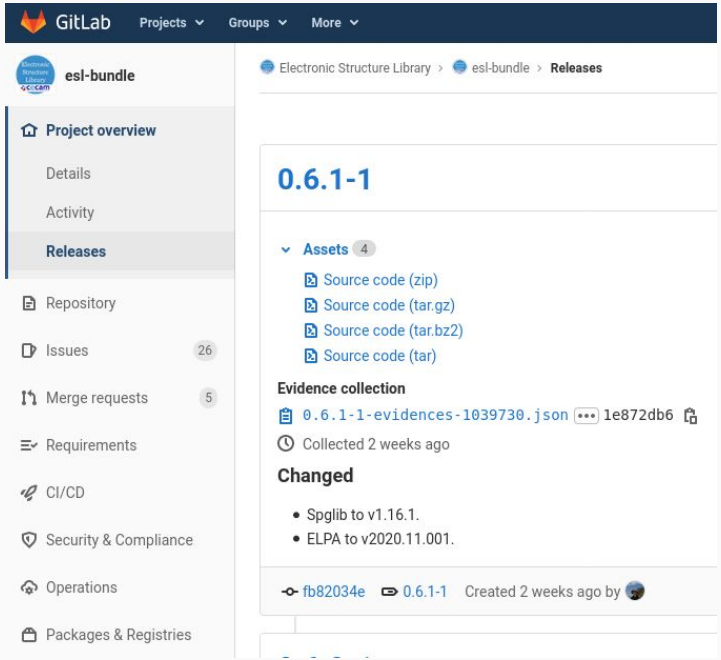
All:
Report problems, at least on Forum!

Beyond the build system

Collaborations within the community

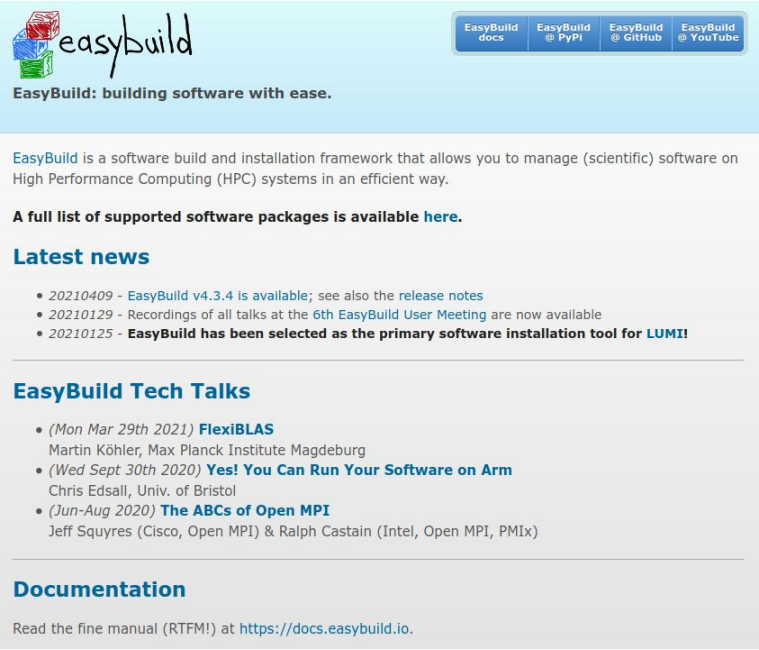
Electronic Structure library: https://esl.cecam.org/

- On Gitlab: https://gitlab.com/ElectronicStructureLibrary
- ESL Bundle: Python-based meta-build system
- ESL Demonstrator: understand code needs
- ESCDF: exchange data between DFT codes



HPC-oriented collaborations

- Build frameworks
  - EasyBuild
  - Spack
- Containerised builds
  - Docker (CI, CD, distribution)
  - Singularity (HPC)
- Collaborations
  - ESL & EasyBuilders
  - ESL & MOLSSI
  - CoEs: E-CAM, MaX, POP, …



easybuild

EasyBuild docs | EasyBuild @ PyPi | EasyBuild @ GitHub | EasyBuild @ YouTube

EasyBuild: building software with ease.

EasyBuild is a software build and installation framework that allows you to manage (scientific) software on High Performance Computing (HPC) systems in an efficient way.

A full list of supported software packages is available here.

Latest news

- *20210409* - EasyBuild v4.3.4 is available; see also the release notes
- *20210129* - Recordings of all talks at the 6th EasyBuild User Meeting are now available
- *20210125* - **EasyBuild has been selected as the primary software installation tool for LUMI!**

EasyBuild Tech Talks

- *(Mon Mar 29th 2021)* **FlexiBLAS**
  Martin Köhler, Max Planck Institute Magdeburg
- *(Wed Sept 30th 2020)* **Yes! You Can Run Your Software on Arm**
  Chris Edsall, Univ. of Bristol
- *(Jun-Aug 2020)* **The ABCs of Open MPI**
  Jeff Squyres (Cisco, Open MPI) & Ralph Castain (Intel, Open MPI, PMIx)

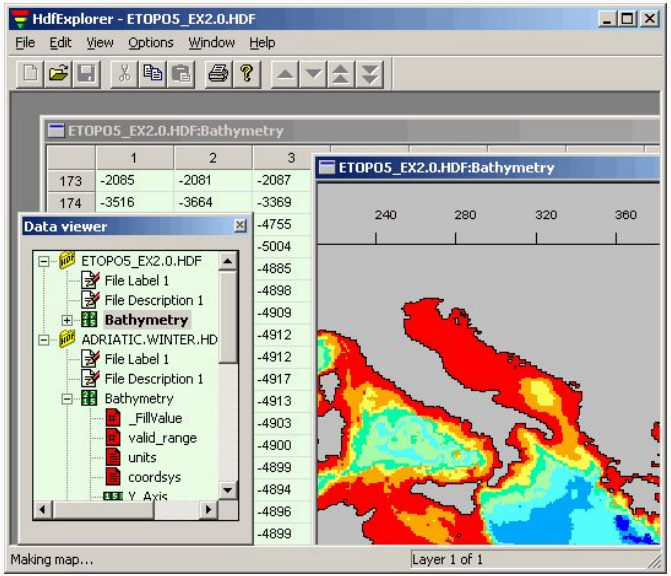
Documentation

Read the fine manual (RTFM!) at https://docs.easybuild.io.

Data-oriented collaborations

- Text-based files
  - Input files: FDF ⇒ LibFDF
  - NC pseudos ⇒ PSML, XMLF90, LibPSML
  - PAW datasets ⇒ PAW-XML, LibPAW
  - JSON, YAML ⇒ Python, BigDFT

- Binary files
  - HDF5, NetCDF
  - ESCDF, LibESCDF (under development)



Current plans

- Improve error messages and hints provided by *configure*
- Improve support for Intel OneAPI + FFTW side effects
- Improve HDF5 and NetCDF detection
- Unify build-system UIs of libraries, utilities & generators (AtomPAW upcoming)
- Develop a testing & templating framework for build systems
- Get rid of the circular dependency between BigDFT & ABINIT?

Thank you for your time!