Paving the way towards Abinit 8 in the post-Moore’s law era

Yann Pouillon¹, Matteo Giantomassi², Jean-Michel Beuken², Thierry Deutsch³, Damien Caliste³, Xavier Gonze²

¹Euskal Herriko Unibertsitatea & ETSF Spain, Donostia-San Sebastián, Spain
²NAPS, Université catholique de Louvain, Louvain-la-Neuve, Belgium
³L_Sim, Commissariat à l’Énergie Atomique, Grenoble, France

Dinard, France — 2013/04/15
Summary of the previous episode

Abinit 6 $\rightarrow$ Abinit 7

“Age is the acceptance of a term of years. But maturity is the glory of years.”
— Martha Graham

- Integrate 3 years of developments & expansion
- Mark a change in paradigms: serial $\rightarrow$ scaling
- Validate setting of test farm
- Validate phase separation of internal routines
- Validate first stage of block splitting (build systems)
- Give a new impulse
A change in context

- 2012: core frequencies down
- 2020: serial code unusable
- Change paradigms now!
- Abinit: in progress (PRACE)
New times, new ways

“*You are never too old to become younger!***”
— Mae West

- End of Moore’s law $\rightarrow$ scale, scale, scale
- Open data & high throughput $\rightarrow$ test, test, test
- Workflow-based methodology $\rightarrow$ split, split, split
- Rapidly evolving context $\rightarrow$ prune, prune, prune
- Nanotechnology $\rightarrow$ collaborate, collaborate, collaborate

Abinit 7 $\rightarrow$ Abinit 8
### The weight of years

#### Growth (Mb/year)

<table>
<thead>
<tr>
<th>Block</th>
<th>Contributors</th>
<th>Growth (Mb/year)</th>
<th>Rhythm (events/month)</th>
<th>Conflicts (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>60</td>
<td>4</td>
<td>500 commits</td>
<td>97</td>
</tr>
<tr>
<td>Tests</td>
<td>10</td>
<td>20</td>
<td>3 new tests</td>
<td>0</td>
</tr>
<tr>
<td>Doc</td>
<td>10</td>
<td>8</td>
<td>3 updates</td>
<td>0</td>
</tr>
<tr>
<td>Other</td>
<td>5</td>
<td>Variable, &lt; 1</td>
<td>Variable</td>
<td>3</td>
</tr>
</tbody>
</table>
Skimming the Fatboy Abinit soup

Ideal workflows

<table>
<thead>
<tr>
<th>Block</th>
<th>Forge</th>
<th>Validation</th>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>S+O = Core</td>
<td>Current structure</td>
<td>Unitary tests</td>
<td>Self-generated</td>
</tr>
<tr>
<td>Tests</td>
<td>One branch</td>
<td>Full runs</td>
<td>Self-generated</td>
</tr>
<tr>
<td>Doc</td>
<td>One branch</td>
<td>Builds</td>
<td>Included</td>
</tr>
</tbody>
</table>

Proposals (to be discussed)

1. Test Farm: split builds from tests
2. Web interface to add & edit full tests
3. Delegate build systems of test suite & documentation
4. Create team to design unitary tests
5. Create team to upgrade documentation
Abinit 8: a rejuvenated project

- Core: $6 \times$ smaller $\iff$ Bazaar $6 \times$ faster
- Tests: no download & easier maintenance if remote management
- Documentation: web-based management possible
- Test Farm: new workflows
- Fully independent build systems
  - Core: Yann Pouillon
  - Tests: Matteo Giantomassi?
  - Doc: Volunteer?

Wait! There’s some fat left there ...

- Split UIs from core: only one more branch
- Most dependencies unrelated to Abinit
- Create unique basis for common needs
- Easier to unify look & feel
- Easier to disentangle Abinit & UIs bugs
Connecting external libraries

Connectors

- DFT
- MATH
- ALGO
- FFT
- TIMER
- TRIO
- LINALG

Multicore

- OpenMP
- MPI
- GPU

Abinit Group (Workshop 2013)
Connectors — Abinit 7

The diagram illustrates the connections between various libraries and tools, indicating their dependencies. Notable components include:

- **Connectors v7**
- **ALGO**
- **DFT**
- **FFT**
- **LINALG**
- **MATH**
- **TIMER**
- **TRIO**
- **Goedecker**
- **DFTI**
- **LibXC**
- **BigDFT**
- **Levmar**
- **AtomPAW**
- **FoX**
- **NetCDF**
- **ETSF_IO**
- **PAPI**
- **Cclock**
- **Libpspio**

The diagram shows a circular dependency, indicating a potential issue that needs to be addressed. Libraries such as **MKL**, **FFTW3**, **Mlib**, **ASL**, **Wannier90**, **src**, and others are interconnected, highlighting the complexity of dependencies in scientific computing environments.

Abinit Group (Workshop 2013)
Balancing complexity

- **Forbid circular dependencies explicitly in abirules**
- Replace fallbacks by build instructions & external project
- Remove unused connector flavors
- Remove dependencies on obsolete libraries
- Remove AtomPAW fallback: no dependency
- Split necessary from optional libraries

- **New configure stage: feature triggers**
- Take decisions earlier
- Manage interactions: Architecture, Platform, Workflow, Abinit
- Refactor complex connectors (e.g. Netlib)
- Prepare externalization of basic components
- Included: LibXC, Libpspio, NetCDF, Netlib
Adapting to complex architectures

- Within a computing node: OpenMP, Threads
- Between computing nodes: MPI and/or GPU
- Adapt to heterogeneous environments
- Provide multi-scale parallelism
- Decision helper for Abinit
- See Matteo Giantomassi’s talk on parallelism
Allowing workflow-based runs

- Provide well-defined paths for data exchange
- Complex systems $\rightarrow$ multiple codes (*Quantum Espresso, Yambo, ...*)
- Complex calculations $\rightarrow$ integration (*ASE, GUIs, Test Farm, ...*)
- Increasing sizes $\rightarrow$ multi-scale modeling (*e.g. DFT + tight-binding*)
- Facilitate external contributions & reviews
- See Ask Hjorth Larsen’s talk on ASE
- See session 7 on GUIs + Test Farm
- See Gabriel Antonius’ talk on Abipy
Explore common libraries providing basic features
DFT/PW: I/O, linear algebra, FFT, maths, XC, pseudopotentials
Use & reuse existing code & standards
Contribute by externalizing Abinit components
See Micael Oliveira’s talk on Libpspio & LibXC
Common part: basic definitions, multicore, timing, toolbox
DFT-related parts of interest to other codes
Very strict naming conventions (e.g. \textit{ab8} prefix)
Very clear API
Very modular structure
Fully informative
\textit{Discussion: complete list?}
Up to now: Python bindings for the Abinit parser

\[ \text{used by } V\_Sim \]

Increasing number of requests for more

Step 1: disentangle BigDFT to free Abinit

Step 2: add dynamic shared objects support to build system

Step 3: release Libabinit

Discussion: solving the BigDFT Ouroboros

Discussion: new bindings
Conclusion: the Abinit 8 todo list

1. Skim the Abinit Fatboy soup
2. Restructure the Forge & the Test Farm
3. Prune & refactor the source code
4. Develop the feature triggers
5. Refactor & optimize the connectors
6. Release Libabinit
7. Design & develop new bindings
8. Checkpoint: next Abinit Workshop
THANK YOU ALL!