PyRosetta-4

What’s new and why should you care!

% cd main/source/src/python/PyRosetta && ./build.py -j8

Sergey Lyskov, GrayLab@JHU, RosettaCon 2016, devel session
PyRosetta-3

- GCCXML → XML
- Boost.Python
- Resulting code: C++98
- Can only be built with GCC and GNU libstdc++
- Mac’s build is limited to use old version of GCC-4.1
- No support for C++11
- SLOOOOO00W
PyRosetta-4

- GCCXML

 Binder: tool for creating Python bindings for arbitrary C++ code.
  - based on Clang libTooling
  - implemented in C++
  - supports parsing of C++11, C++14, C++17, ...
PyRosetta-4

- Boost.Python → PyBind11
- Our-own-custom-build-system → CMake
Architecture

• PyRosetta-3: ‘rosetta’ included both rosetta and PyRosetta code

• PyRosetta-4: two separate entities:
  • rosetta.so (bindings for Rosetta C++ code) and
  • pyrosetta

```python
from __future__ import print_function

import rosetta
import pyrosetta

pyrosetta.init()
print(pyrosetta.version())
```
What’s new in PyRosetta-4?

• Automatic bindings generation for C++ templates

• Classes with virtual function: no run-time overhead for overload classes!

• Function default arguments now bound properly (via C++11 lambda). For example when binding void foo(int a=1, int b=2); we will generate:

```c
void py_foo() { foo(); }
void py_foo(int a) { foo(a); }
void py_foo(int a, int b) { foo(a, b); }
```
What’s new in PyRosetta-4?

- Binding for function accepting pointer to primitive types: int *, double *, bool *, … etc

- Support for bindings of C++11 code and C++14!

- ‘python setup.py install’ → SetPyRosettaEnvironment.sh

- New source location: main/source/src/python/PyRosetta
Build types

- Old PyRosetta-3: namespace/monolith builds
- New PyRosetta-4: **ONLY MONOLITH BUILD**
- Debug/Release/MinSizeRel/RelWithDebInfo
What code is bound?

• All ‘bindable’ regular code (no exceptions!) And this includes:
  
  • ObjexxFCL

  • Option System

• At this moment only skipped code is ether:

  • Template code that could not be instantiated

  • Code that could not be adequately represented in Python
Python versions compatibility

• Thank you for participating in Python version survey!

Which versions of Python should we support for PyRosetta builds?
(13 responses)

- 53.8%: Python-3 only! Let’s embrace the future!
- 38.5%: Python-2 only. I do not care about Python-3 at all and do not plan to use it! Maybe Python-4 later...
- 7.7%: Both Python-2 and Python-3 versions should be supported even if that raise complexity.
Python versions compatibility

- Both generation of bindings for Python-2 and Python-3 now supported
- PyRosetta python code, demos, tests are compatible with both Python-2 and Python-3
What is not yet implemented?

- No Python ‘doc’ strings yet
- print <object>
- PyRosetta ‘apps’ is not yet ported
- PyRosetta GUI tests is not yet ported
- PyRosetta test C001_Carbohydrates_Demo01 fail when ‘installed’
- Python PyMOL Mover implementation is not ported and deprecated
  (please use C++ version instead)
Limitations

• Right now needs to be built with Clang

• `std::pair` and `std::tuple` is *read-only*

• no support for multiple inheritance (and probably will never be due to PyBind11 limitations)
## Build Speed

<table>
<thead>
<tr>
<th></th>
<th>Generation phase</th>
<th>Build phase</th>
<th>Total (Rosetta excluded)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PyRosetta-3</td>
<td>5.9 cpu·h</td>
<td>27.2 cpu·h</td>
<td>33.1 cpu·h</td>
</tr>
<tr>
<td>PyRosetta-4</td>
<td>0.06 cpu·h</td>
<td>5.6 cpu·h</td>
<td>5.66 cpu·h (17%!</td>
</tr>
</tbody>
</table>
## Size and Memory consumption

<table>
<thead>
<tr>
<th>PyRosetta-3</th>
<th>1,368 Mb allocating ~4Gb on scoring!!!</th>
<th>100%</th>
</tr>
</thead>
<tbody>
<tr>
<td>boost.python, release</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PyRosetta-4</th>
<th>346 Mb</th>
<th>25%</th>
</tr>
</thead>
<tbody>
<tr>
<td>PyBind11, release, Linux</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PyRosetta-4</th>
<th>244 Mb</th>
<th>18%</th>
</tr>
</thead>
<tbody>
<tr>
<td>PyBind11, MinSizeRelease, Linux</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Overall PyRosetta-4:

- Binds more code (all templates!)
- Binds all default function arguments
- Builds x5 faster
- Consumes x5 less memory
- Standard Python package
Binder architecture

• No mention of ‘Rosetta’ or Rosetta specific classes in the code!

• Config file to specify bindings options, special binders, … etc: **pluggable architecture**

• I am going to release Binder as separate package under **FreeBSD license** this fall
PyRosetta-4, when? - right now!

http://benchmark.graylab.jhu.edu/submit
How to build?

Install Clang, CMake and Ninja and then:

% cd main/source/src/python/PyRosetta

%.build.py -j8

% python3 build.py -j8
PyRosetta-4 generated source is available at Benchmark test page:

Test: linux.clang.python3.PyRosetta4.unit
Branch: master 「revision: №58790」
Test files: 「file-system-view」 「file-list-view」
Daemon: Hojo-1 Run time: 0:09:13
State: passed

[...] [source] [.0.output.log] [.0.results.json] build-log.txt output.json

[...] [ObjexxFCL] [basic] [core] [cpppdb] [libxml] [numeric] [protocols] [std] [utility]
(CMakeLists.txt ObjexxFCL.cmake all_rosetto_includes.hh basic.cmake
cifparse.cmake core.1.cmake core.2.cmake core.3.cmake
core.4.cmake core.5.cmake cpppdb.cmake libxml2.cmake numeric.cmake
protocols.1.cmake protocols.3.cmake protocols.6.cmake

[...] [carbohydrates] [copydofs] [datacache]
[full_model_info] [metrics] [motif] [ncbb]
[reference_pose] [rna] [signals] [symmetry]
MiniPose.cpp PDBPoseMap.cpp Pose.cpp annotated_sequence.cpp selection.cpp
util.cpp util_1.cpp util_2.cpp utiltmpl.cpp
xyzStripeHashPose.cpp xyzStripeHashPose_fwd.cpp
Thank you!