Boosting Python with Rust

The case of Mercurial

FOSDEM 2020

Raphaël Gomès @ OCTOBUS
Mercurial

• Same generation as Git
• Written in Python (200k lines)
• Boosted by C extensions (45k lines)
• Handles huge repos (millions of files and/or revisions)
• Very powerful extension system
Why Rust?
Rust

- Low-level language
- Powerful type system
- No garbage collector
- Compile-time memory safety
- Simple(r) parallelism
Maintainability
Compared to C

• Better signal/noise ratio
• Better compile-time guarantees
• Standardized and modern tooling
• "Safe" by default (unsafe blocks)
Performance

- Comparable to C for sequential code
- Parallel code is much simpler to write and maintain
- Allows for optimizations impossible for C compilers
## Performance

“hg status” experiment by Valentin Gatien-Baron

<table>
<thead>
<tr>
<th>Command</th>
<th>hg</th>
<th>Rust hg</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>2.4s</td>
<td>50ms</td>
</tr>
<tr>
<td>status -u</td>
<td>2.4s</td>
<td>39ms</td>
</tr>
<tr>
<td>status -mard</td>
<td>400ms</td>
<td>14ms</td>
</tr>
</tbody>
</table>
rust-cpython

- A low-level crate for the CPython ABI
- A high-level crate to interact with Python:
  - Expose a Rust module to Python
  - Create Python function and classes
  - Execute Python from Rust
Structure

CPython

Pure Python

C Extensions

hg-cpython

hg-core
A slow start

<Alphare> So, I finished rewriting this function in Rust
<Alphare> The bad news is: it’s twice as slow
Friction with Python

- Complex interface code
- Exchanging data is costly
Friction with Python

- **stat** of 100k files in Rust: 30ms
- Giving the results to Python: 300ms
Possible solutions

• Exchange less data
• Do more in Rust
• Communicate with C directly
Talk to C directly

1 Feb 2020

Boosting Python with Rust
Talk to C directly

CPython

Pure Python

C Extensions

hg-cpython

hg-core
Capsules

- **PyCapsule**: Python object that encapsulates function pointers
- Can be defined in a module, used in another
- Exactly made to share a C API between extensions
Capsules

~ $ python3
Python 3.7.2 (default, Jan 3 2019, 02:55:40)
[GCC 8.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import datetime
>>> datetime.datetime_CAPI
<capsule object "datetime.datetime_CAPI" at 0x7fdfe613b10>
Missing features

- PySet
- Simple support of PyCapsule
- Inheritance for classes written in Rust
- Properties and `setattr`
- Iterators on Rust collections
A Python iterator in Rust

- Should behave exactly as a Python iterator
- Tell the Rust compiler that it really has to let go
- Handle sharing references between the two languages
Upstream work

- **PySet** ✓
- Simple **PyCapsule** support ✓
- Properties ❗
- Iterators on Rust collections ✓
Performance
“hg status” experiment by Valentin Gatien-Baron

<table>
<thead>
<tr>
<th>Command</th>
<th>hg</th>
<th>Rust hg</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>2.4s</td>
<td>50ms</td>
</tr>
<tr>
<td>status -u</td>
<td>2.4s</td>
<td>39ms</td>
</tr>
<tr>
<td>status -mard</td>
<td>400ms</td>
<td>14ms</td>
</tr>
</tbody>
</table>
## Current performance (pathological case, 100k files)

<table>
<thead>
<tr>
<th></th>
<th>Python + C</th>
<th>Python + Rust</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>6.23s</td>
<td>1.59s</td>
</tr>
<tr>
<td>status -mard</td>
<td>1.46s</td>
<td>840ms</td>
</tr>
<tr>
<td>diff</td>
<td>1.5s</td>
<td>880ms</td>
</tr>
</tbody>
</table>
## Current performance
(more realistic case, 260k files)

<table>
<thead>
<tr>
<th></th>
<th>Python + C</th>
<th>Python + Rust</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>2.9s</td>
<td>2.0s</td>
</tr>
<tr>
<td>status -mard</td>
<td>1.7s</td>
<td>1.0s</td>
</tr>
<tr>
<td>diff</td>
<td>1.9s</td>
<td>1.2s</td>
</tr>
</tbody>
</table>
#TODO

- Do more things in parallel
- Better conditional execution
- Rethink the order of execution
- Fewer exchanges between Python and Rust
- Fewer allocations, etc.
#TODO

- Do more things in parallel
- Better conditional execution
- Rethink the order of execution
- Fewer exchanges between Python and Rust
- Fewer allocations, etc.
- ...not start Python ??
A renewed appreciation for Python

• Code is very easy to understand
• You get something that works very quickly
• Allows for experimentation
• It is a lot faster than Rust code you are not done writing
FOSDEM 2020

Thank you!

Raphaël Gomès @ octobus