Process profiler Documentation

Release 0.1.0

Joel Akeret

July 28, 2016

Contents

A simple process profiler. **propro** can be used in many different ways. Conviniently in can be used on the command line:

```
$ propro --fmt=png <command>
```

for more options call:

```
$ propro --help
```

Another option is to call the profiling programatically

```
import propro
x = propro.profile_cmd("ufig --background-type=chunked_map ufig.config.random")
```

The returned profiling result can than for instance be used for custom plotting.

propro offers the option to profile a single Python function using a decorator

```
import propro
import numpy as np


@propro.profile(sample_rate=0.1, fmt="txt")
def mem_hungry(size):
    a = []
    for i in range(size):
        a.append(np.random.random())

    b = []
    for i in range(size):
        t = []
        for j in range(size):
            t.append(i * a[j])
        b.append(t)

    b = np.array(b)
```

The profiling output is stored in the folder where the Python code was launched.

Finally, **propro** can be embedded in your IPython notebooks. Load the extentsion with

```
import propro

%load_ext propro
```

The profiling can be done on line level

```
%propro -r 0.1 load_pixels(path, PIXEL_COUNT)
```

or on cell level

```
%%propro -r 0.1
X = np.random.normal(size=(200,200,1000))
P, D, Q = np.linalg.svd(X, full_matrices=False)
X_a = np.dot(np.dot(P, np.diag(D)), Q)
print(np.std(X), np.std(X_a), np.std(X - X_a))
```

The output will look something like this if rendered into an image:

CHAPTER 1

Contents:

1.1 Installation

The project is hosted on GitHub. Get a copy by running:

```
$ git clone https://github.com/jakeret/propro.git
```

Install the package like this:

```
$ cd propro
$ pip install -r requirements.txt
$ python setup.py install --user
```

1.2 propro Package

1.2.1 `propro` Package

`propro.__init__.`**`format_date`**(*date*, *fmt='%Y-%m-%d-%H:%M:%S'*)

1.2.2 `cli` Module

Created on Mar 29, 2016

author: jakeret

1.2.3 `magic` Module

1.2.4 `plotting` Module

1.2.5 `propro` Module

class `propro.propro.`**`profile`**(*sample_rate=None*, *timeout=None*, *fmt=u'txt'*, *callname=None*)

Bases: `object`

Decorator to profile a function or method. The profile result is automatically written to disk.

Parameters

- **`sample_rate`** – (optional) Rate at which the process is being queried
- **`timeout`** – (optional) Maximal time the process is being profiled
- **`fmt`** – (optional) The desired output format. Can also be a tuple of formats. Supported: txt and any matplotlib fmt
- **`callname`** – (optional) Name used for plot title and output file name

`propro.propro.`**`profile_pid`**(*pid*, *sample_rate=None*, *timeout=None*)
Profile a specific process id

Parameters

- **`pid`** – The process id to profile
- **`sample_rate`** – (optional) Rate at which the process is being queried
- **`timeout`** – (optional) Maximal time the process is being profiled

Returns ProfileResult A *ProfileResult* namedtuple with the profiling result

`propro.propro.`**`profile_cmd`**(*cmd*, *sample_rate=None*, *timeout=None*)
Profile a specific command

Parameters

- **`cmd`** – The command to profile
- **`sample_rate`** – (optional) Rate at which the process is being queried
- **`timeout`** – (optional) Maximal time the process is being profiled

Returns ProfileResult A *ProfileResult* namedtuple with the profiling result

1.3 Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

1.3.1 Types of Contributions

Report Bugs

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Implement Features

Write Documentation

Process profiler could always use more documentation, whether as part of the official Process profiler docs, in docstrings, or even on the web in blog posts, articles, and such.

Submit Feedback

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

1.3.2 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, and 3.3, and for PyPy. make sure that the tests pass for all supported Python versions.

1.3.3 Tips

To run a subset of tests:

```
$ py.test test/test_propro.py
```

1.4 Credits

1.4.1 Development Lead

- Joel Akeret <jakeret@phys.ethz.ch>

1.4.2 Contributors

None yet. Why not be the first?

1.5 History

1.5.1 0.1.0 (2016-01-01)

- First release on PyPI.

CHAPTER 2

Feedback

If you have any suggestions or questions about **Process profiler** feel free to email me at jakeret@phys.ethz.ch.

If you encounter any errors or problems with **Process profiler**, please let me know!

Index