NitPycker Documentation

Release 0.1

Benjamin Schubert

July 13, 2016

Contents

NitPycker is a improved runner for your tests in Python. It allows your tests to run in parallel in a way totally transparent for you.

For a quick jump start, please see *Quickstart*, for more information please read *Why NitPycker ?*.

However, be warned that if you have tests that depend on each others, or tests fixtures that prevent the use of multi-processing, some tests will fail. This is *not* a bug in NitPycker. You can still use nitpycker if you have only a few problems. For this, please see *Avoiding problems*.

CHAPTER 1

Why NitPycker ?

I created NitPycker because running Python unittest in a multithreaded environment was impossible for me. I had a heavy test suite (>90 tests) with integration tests, and running them sequentially would take more than 8 hours. With NitPycker, time got down to 40 minutes.

Other Python testing framework do also provide multithreaded tests, such as py.test, nose and nose2. However, they all have some problems when running in multithreaded environment (code coverage incorrect, path manipulation, deadlocks) meaning that they are not perfect for it. NitPycker is created with multiprocess *in mind*, meaning a better integration and more reliable results. If tests are *independent* and work with Python's unittest, Nitpycker will be fully compatible with them [1].

[1] if it's not, please open an issue, this should get fixed !

CHAPTER 2

Quickstart

Getting started is trivial :

1. Download NitPycker from the NitPycker page on the Python Package Index or by using `pip install nitpycker`.
2. Use `nitpycker` to run your tests

```
$ nitpycker [options] [args] directory
```

CHAPTER 3

Getting help

Bug reports are welcome on the GitHub's project's issue tracker

CHAPTER 4

More information

4.1 NitPycker' command line usage

On installation, NitPycker adds a command line script `nitpycker` so that you can run your tests either by :

```
$ ./nitpycker [options] [plugins] ${start_directory}
```

or :

```
$ python3 -m nitpycker [options] [plugins] ${start_directory}
```

You can also access help through:

```
$ ./nitpycker --help
```

4.2 Test coverage

NitPycker can be used with Coverage.py. You will however need to create a `.coveragerc` file in the directory from which you run your tests.

This file needs to contain all information about how coverage should run. This is needed as processes need to get access to this information and it is for now not possible through the command line.

One thing that is absolutely needed to have meaningful results is to add

`concurrency = multiprocessing` in the `[run]` section as nitpycker uses the multiprocessing package

An example of .coveragerc file is given here

4.3 Solving problems

Sometimes, some tests are required to run sequentially, or it is easier to have them run that way. For this, NitPycker allows you to set a `__no_parallel__ = True` attribute to any module or class, and NitPycker will then run them as if they were run by unittest, in the same order and the same way [1].

If however your tests are isolated and fail to run with NitPycker, please open an issue on the *issue tracker*

[1] In this, NitPycker takes the opposite paradigm compared to nose, which requires you to tell which testes can be run in parallel. Here, you tell which *cannot*

4.3.1 Avoiding problems

These are a few tips that can make you run into trouble is used with parallel tests:

1. sharing variables between tests
2. sharing resources (files)

If you can, you should avoid these, as they also do mean that the tests are not correctly isolated from each others