

Lesson10 performance tuning activity

PyPy vs cPython

According to the PyPy documentation, PyPy has several advantages over cPython including:

- Increased speed
- Less memory usage
- cPython compatibility
- Stackless mode support for micro-threads

Due to the stated advantages of using PyPy over cPython for heavy computational work, I will explore PyPy vs. cPython on some of our earlier code and document the performance differences. In particular, I'll be comparing the two interpreters while running a double recursive function to compute a value in the Fibonacci series.

The code is as follows:

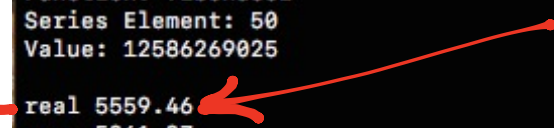
```
def fibonacci(n):  
    '''  
    This function will return the 'n'th value in the fibonacci series.  
    The starting elements in the series are 0, 1.  
    '''  
    if n == 0:  
        return 0  
    if n == 1:  
        return 1  
    else:  
        return fibonacci(n-1) + fibonacci(n-2)
```

The time will be measured using the GNU Time module from the command line. To make things interesting, I'll be computing the series to the 50th value. Tests were ran on a 2017 MacBook Pro. While the function was running, a single processor thread was observed consistently between 99-100% confirming that this function is making heavy use of the processor to compute the value(s).

PyPy did indeed show a marked improvement on this heavy computational program. Here are the results:

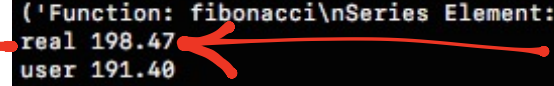
cPython:

```
Ryans-MacBook-Pro:lesson10 rdrovdahl$ time -p python3 series.py  
Function: fibonacci  
Series Element: 50  
Value: 12586269025  
real 5559.46  
user 5361.37  
sys 171.61  
Ryans-MacBook-Pro:lesson10 rdrovdahl$
```



PyPy:

```
Ryans-MacBook-Pro:lesson10 rdrovdahl$ time -p pypy series.py  
(Function: fibonacci\nSeries Element:', 50, '\nValue:', 12586269025, '\n')  
real 198.47  
user 191.40  
sys 6.28  
Ryans-MacBook-Pro:lesson10 rdrovdahl$
```



As you can see, there is a dramatic difference between the two. PyPy was 28 times faster than cPython for this particular function.

There are some drawbacks to using PyPy having to do with Python extension support but they claim to work with most libraries. Based on the results of this testing, it would be worth giving PyPy a look anytime heavy computational work is occurring.