Microproject - Python Performance

Advisor: Kaiser Pister
Rachael Lang
Department of Computer Sciences, University of Wisconsin - Madison
Spring 2023

Objective

The objective of this microproject is to confirm the claim that Python 3.11 is 10-60% faster than Python 3.10. On average, the Python development team measured a 1.25x speedup on the standard benchmark suite (What's New In Python 3.11). This is primarily a replication study on performance improvement, specifically the "faster runtime" claim.

Implementation

The environment I used for this study was the Command Prompt on Windows 10, and I changed versions by specifying the file path in the environment variables. I read through the "What's New In Python 3.11" release documentation and implemented benchmark tests based on the reports. In particular, I focused on PEP 659: Specializing Adaptive Interpreter. The following code snippets are the code I ran to test various operations, as outlined in the documentation. Below each line of code is the output given in versions 3.10 and 3.11, respectively.

Binary Operations

```
# input
>>> python -m timeit 246810+135790

# output
3.10: 20000000 loops, best of 5: 11.9 nsec per loop
3.11: 20000000 loops, best of 5: 10.8 nsec per loop
```

```
>>> python -m timeit 96000-400000
3.10: 20000000 loops, best of 5: 12.0 nsec per loop
3.11: 20000000 loops, best of 5: 11.4 nsec per loop
```

```
>>> python -m timeit 246810*135790
3.10: 20000000 loops, best of 5: 11.5 nsec per loop
3.11: 20000000 loops, best of 5: 11.2 nsec per loop
```

Subscript

```
>>> python -m timeit new_list=[0, 1, 2, 3, 4, 5, 6] new_list[5]
3.10: 2000000 loops, best of 5: 101.0 nsec per loop
3.11: 2000000 loops, best of 5: 92.8 nsec per loop
```

Store Subscript

```
>>> python -m timeit new_list=[0, 1, 2, 3, 4, 5, 6] new_list[5]=10
3.10: 2000000 loops, best of 5: 98.8 nsec per loop
3.11: 2000000 loops, best of 5: 94.7 nsec per loop
```

Load Global Variable

```
>>> python -m timeit print(5000)
3.10: 50 loops, best of 5: 4.46 msec per loop
3.11: 50 loops, best of 5: 4.10 msec per loop
```

Analysis

How much faster did version 3.11 run compared to version 3.10?

Binary Operations

- Addition: 20.17% faster
- Subtraction: 15.00%
- Multiplication: 13.91%

Subscript

- Get an element from a list: 16.53%

Store Subscript

- Assign a value to an element in a list: 11.73%

Load Global Variable

- Call the built-in print() function: 15.92%

How do these statistics compare to the published figures and to each other?

Compared to published figures

- Python: 10-25%
- Test: 11.73-20.17%

Variation among different operations

- Running different operations resulted in a wide range of runtime improvement
- Running the same test multiple times consistently increased the runtime

Observations

Some key observations I had from running the benchmark tests is that program performance is difficult to test, especially since many factors can affect the results. I also had the opportunity to read an assortment of technical documentation and release reports, especially the "What's New" report, which was particularly interesting to me. Additionally, in reading PEP 659, I got to see some of the organizational structure within the Python Software Foundation, which I found intriguing.

Learning Outcomes

From this microproject, I learned how to set up environment variables and how to run benchmark tests. I also looked at open-source code on Github and used CPython, which I learned is the default Python interpreter that uses C. I also encountered a lot of technical terms I didn't fully understand, which seemed like a great opportunity to fall down Wikipedia rabbit holes.

Further Exploration

Potential topics for further research include testing the faster startup tests as mentioned in the Python documentation, running similar performance tests for other programming languages, and exploring how the physical and virtual environments affect performance. Another question that arose from the study was why repeatedly running the same lines of code consistently increased the runtime.

Sources

Python 3.10
Python 3.11
What's New In Python 3.11
PEP 659: Specializing Adaptive Interpreter