Gehrig Weber

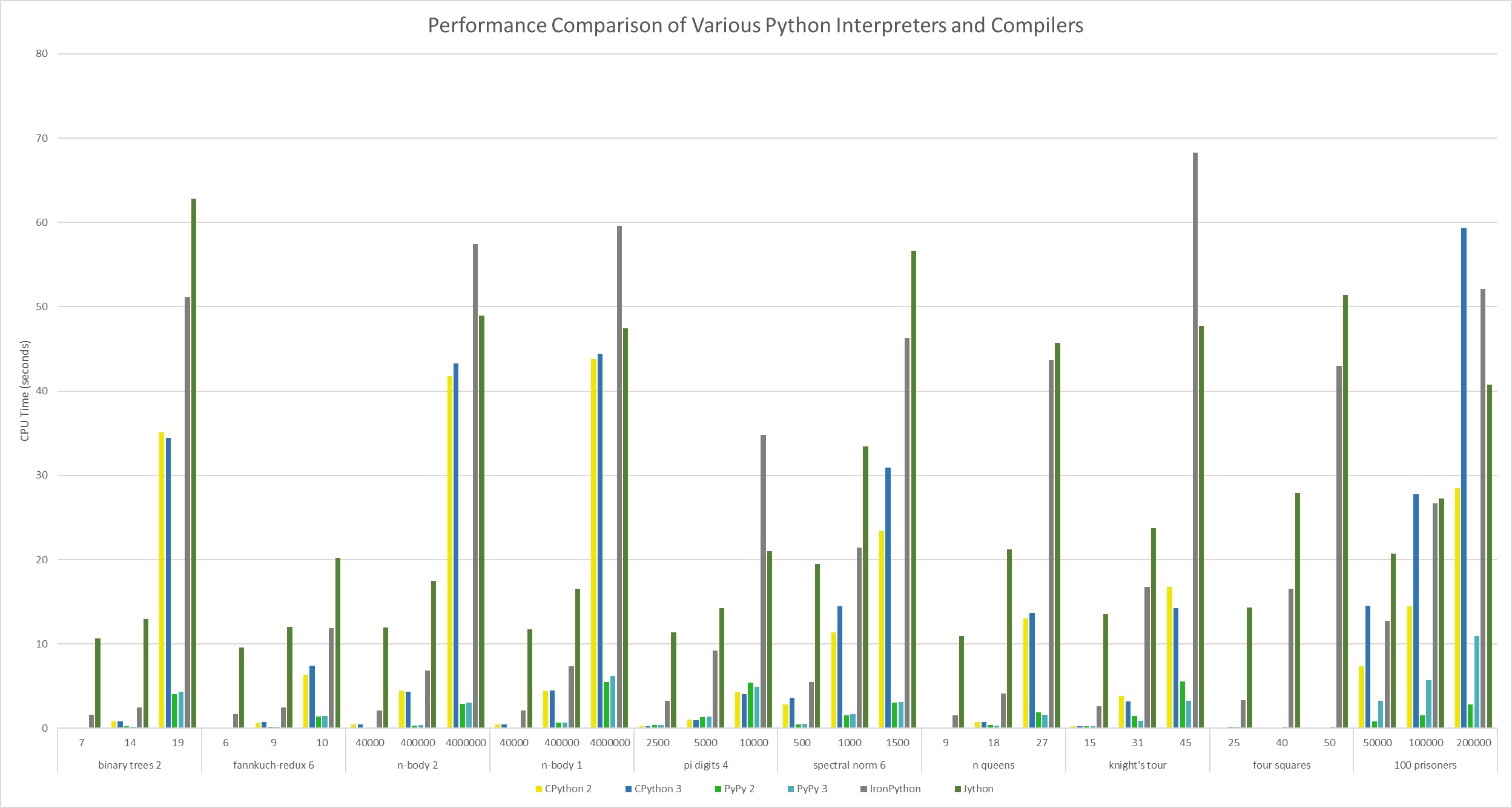
Prof. Krintz

Runtime Systems

June 8, 2020

Comparisons in Performance Between Python Implementations

This project was intended to investigate various Python implementations and evaluate them, based primarily on performance. The implementations are: CPython 2 and 3 (the default Python implementation), PyPy 2 and 3 (just-in-time (JIT) Python compilers), IronPython (an implementation of Python on the .NET runtime, or Common Language Runtime (CLR)), and Jython (an implementation of Python on the Java Virtual Machine (JVM)).



Ten benchmark programs were used to evaluate the implementations’ performance, as seen in the graph above. 3 sets of 5 trials were run with each program. Each set used a different value of N. The results are in CPU time, the total time spent executing the program. They were obtained by adding the user time and system time taken for each of the 5 trials, then averaging those sums. The benchmark programs used will now be detailed. Binary Trees 2 is a program that allocates and deallocates numerous perfect binary trees of increasing depth, from 4 to N. Fannkuch Redux 6 iterates through N! permutations of 1 to N. Let the first number in the sequence be X. For each permutation, the first X numbers are reversed repeatedly, until X = 1. The maximum number of reversals, or flips, is returned at the end. N-Body 2 is a simulation of stellar bodies, which involves performing a constant number of calculations N times. N-Body 1 is identical to N-Body 2, except it uses “\*\* 0.5” instead of “math.sqrt()” to calculate square roots. Pi Digits 4 is a program that prints the first N digits of pi. Spectral Norm 6 calculates the spectral norm of an N-by-N matrix. N Queens uses backtracking to return all possible solutions of the N queens problem (Can N queens be placed on an N-by-N chess board without threatening each other?). Knight’s Tour implements Warnsdorff’s algorithm to return an open tour of an N-by-N chess board with a knight, if one exists. The starting point can be changed, but it was always the top left corner of the board for the trials. Four Squares outputs all combinations of *a, b, c, d, e, f, g* such that *a* + *b* = *b* + *c* + *d* = *d* + *e* + *f* = *f* + *g,* where the values are selected from the range 0 to N without repetition. This involves going through N choose 7 (N!/(N - 7)!) iterations, and doing constant work at each. 100 Prisoners involves going through N iterations of a simulation of the 100 prisoners problem, in which 100 prisoners can each open a maximum of 50 out of 100 drawers in an attempt to find their number. No communication is allowed. All prisoners must find their number in order for any of them to win.

According to the data, the Python implementations, in approximate order from best to worst performing, are PyPy 2, PyPy 3, CPython 2, CPython 3, Jython, and IronPython. The reasons for this will now be examined.

PyPy uses a tracing JIT compiler, which, like most compilers, is faster than CPython’s interpreter. It ends up being faster than IronPython and Jython’s respective JIT compilers, as well. Now, PyPy 2 is faster than PyPy 3 largely because CPython 2 is faster than CPython 3. PyPy also has an efficient garbage collector that results in programs using much less memory than in CPython. Furthermore, PyPy 2 and 3 are extremely compatible with Python 2.7 and Python 3.6.9’s core languages, respectively, establishing PyPy as one of the best Python implementations.

CPython’s use of an interpreter should make it the slowest Python implementation, yet it ends up being faster than IronPython and Jython, even though their runtimes make use of JIT compilers. Between the two versions, CPython 2 is faster than CPython 3 due to the fact that, in CPython 3, an int is what a long was in CPython 2, and longs do not exist. Thus, ints in CPython 3 are twice as big as ints in CPython 2, and operations with ints take much longer in CPython 3.

Jython

However, the aforementioned order does not always hold.

And then lead into exceptions, and then introduce idea that IronPython and Jython may end up being better than CPython in the long run (challenge the conclusions reached in the “usual” paragraphs)

(cut down the program descriptions if running out of space)