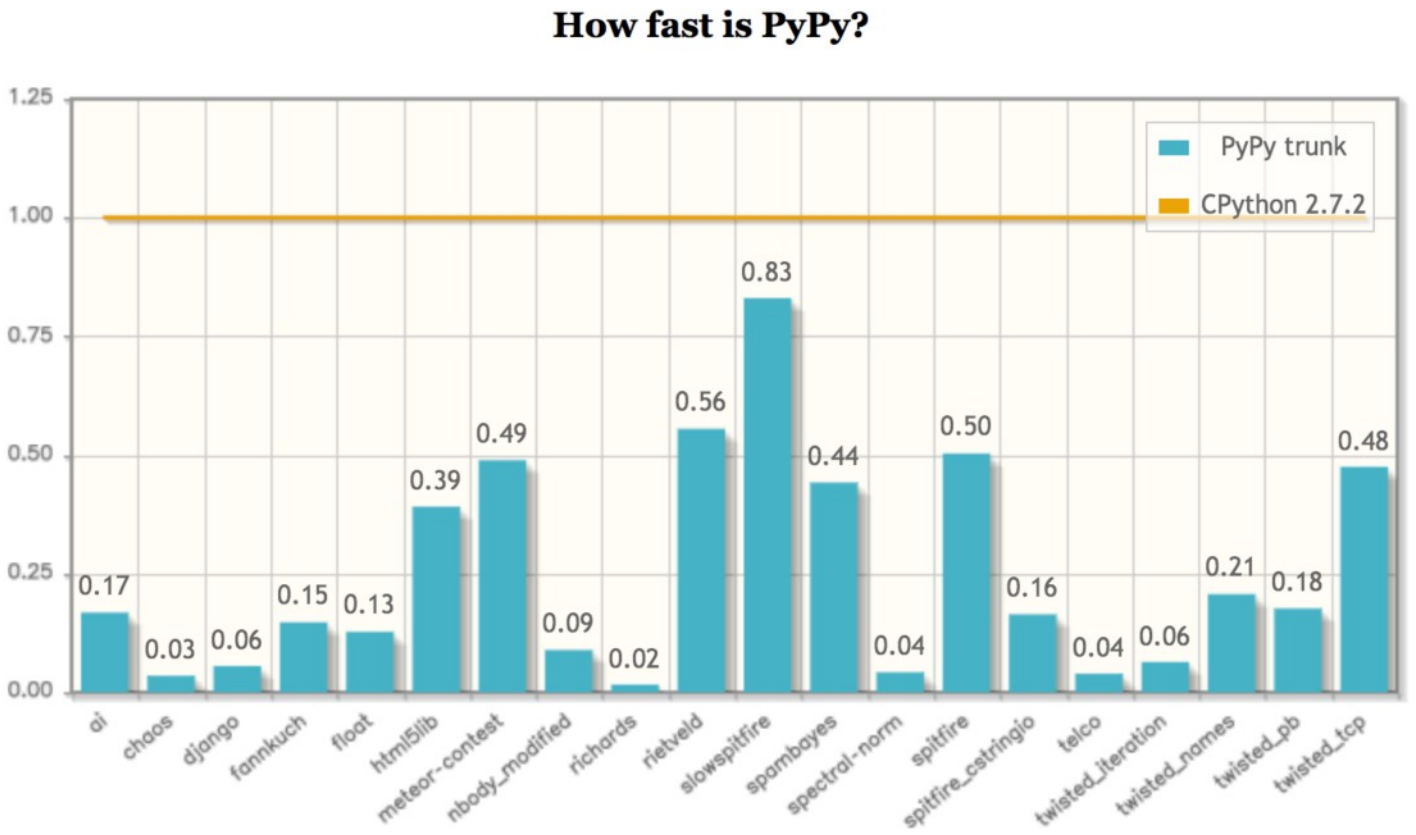
What is PyPy?

PyPy is a compliant alternative implementation of the Python language, written in Python... It's fast!



Plot 1: The above plot represents PyPy trunk (with JIT) benchmark times normalized to CPython. Smaller is better.

What is PyPy?

That's weird! How can an interpreter written in Python be (much) faster than one written in C?

- PyPy is actually written in RPython, a restricted subset of the language amenable to static analysis.
- Conditional code paths are translated to machine code with tracing just-in-time compilation (JIT).
- What is JIT'ed is the interpreter itself, not the user program.

Ref: https://bitbucket.org/pypy/extradoc/src/tip/talk/icooolps2009/bolz-tracing-jit-final.pdf

What is STM?

Concurrency is hard! Models include: shared-memory threads (i.e. with locks); the actor model; shared-nothing or shared-little processes (i.e. with queues) or message passing interface (MPI); and software transactional memory (STM).

STM is a *"concurrency control mechanism analogous to database transactions for controlling access to shared memory in concurrent computing. It is an alternative to lock-based synchronization."*

Ref: https://en.wikipedia.org/wiki/Software_transactional_memory

What is STM?

"STM is very optimistic: a thread completes modifications to shared memory without regard for what other threads might be doing, recording every read and write that it is performing in a log. [...] the reader, [...] after completing an entire transaction, verifies that other threads have not concurrently made changes to memory that it accessed in the past."

Ref: https://en.wikipedia.org/wiki/Software_transactional_memory

What is STM?

You can think of STM as "speculative threading—i.e. perhaps among multiple cores, avoiding the global interpreter lock (GIL).

"Threads execute their code speculatively, and at known points (e.g. between bytecodes) they coordinate with each other to agree on which order their respective actions should be "committed", i.e. become globally visible. Each duration of time between two commit-points is called a transaction."

Ref: http://pypy.readthedocs.org/en/latest/stm.html

What is PyPy-STM?

A *win* in using STM is that code that looks like threaded Python code can actually execute (optimistically) on multiple CPU cores.

"[A]lthough running on multiple cores in parallel, `pypy-stm` gives the illusion that threads are run serially"

Ref: http://pypy.readthedocs.org/en/latest/stm.html#user-guide

What is PyPy-STM?

Let's look at a toy benchmark published with the 2.3r2 release of PyPy-STM. This is running "multithreaded" code:

# of threads	PyPy (head)	PyPy-STM (2.3r2)
N = 1	real **0.92s**	real **1.34s**
N = 2	real **1.77s**	real **1.39s**
N = 3	real **2.57s**	real **1.58s**
N = 4	real **3.38s**	real **1.64s**

Ref: http://morepypy.blogspot.com/2014/07/pypy-stm-first-interesting-release.html

What is PyPy-STM?

```
import thread, sys
def f(n):
    total, lst1 = 0, ["foo"]
    for i in xrange(n):
        lst1.append(i)
        total += lst1.pop()
    sys.stdout.write('%d\n' % total)

T = 4               # number of threads
N = 100000000       # number of iterations in each thread
for i in range(T):
    lock = thread.allocate_lock()
    lock.acquire()
    thread.start_new_thread(f, (N))
    lock.release()
```

Ref: https://bitbucket.org/pypy/benchmarks/raw/default/multithread/minibench1.py

What is PyPy-STM?

Being able to run "threaded code" much faster on multi-core machines is good. What is better is not explicitly using threads at all. i.e. instead of:

```
for key, value in bigdict.items():
    f(key, value)
```

You can use the transaction module and write:

```
import transaction    # Internally uses thread module
for key, value in bigdict.items():
    transaction.add(f, key, value)
transaction.run()
```

Ref: http://pypy.readthedocs.org/en/latest/stm.html#user-guide

Who am I?

A Director of the Python Software Foundation. Chair of PSF's Trademarks, and Outreach and Education, Committees.

I used to be well known as author of the IBM developerWorks column *Charming Python* and the Addison-Wesley book *Text Processing in Python*.

Nowadays, I work at a research lab, D. E. Shaw Research, who have built the world's fastest supercomputer for doing molecular dynamics.