Python Shedding

Ivan Dyedov - Senior Engineer
Rob Harrigan - Director of Engineering

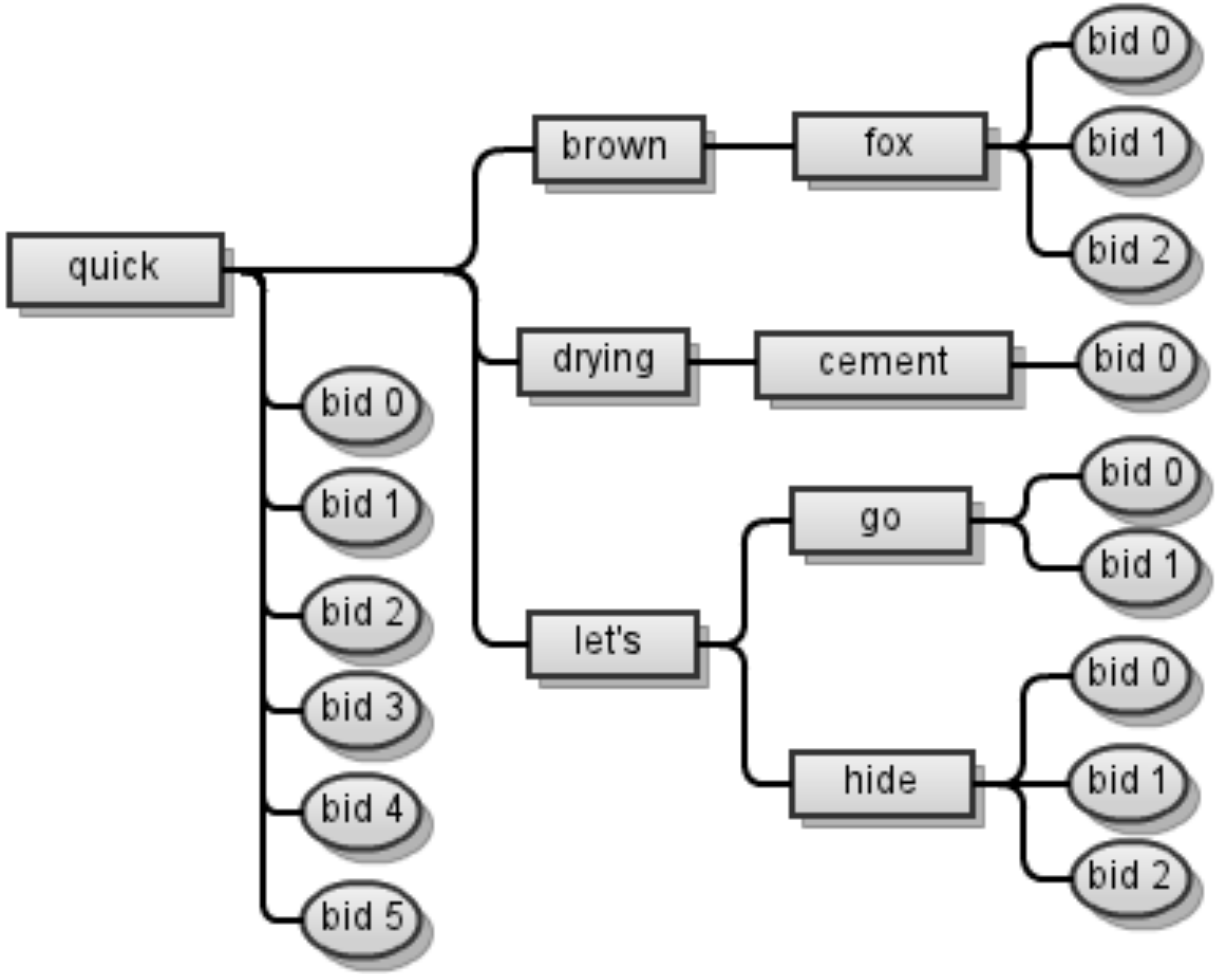
A look at the evolution of an ad server

In the Beginning

There was PHP and it was good (enough)

- Primary Data Structure is a deeply nested tree (Similar to trie, but word based).
  - Fast lookups O(log n)
  - Compressed when in memory because of word repetitions
- Highly tuned lookup algorithm yielded about 60 requests/sec
  - Tree gets reloaded every request
  - As the inventory grew in size, most of the request time was spent unserializing tree

Example Data Structure



PHP needs to go ... Daemon-off

As our inventory size grew it becomes apparent that we need to keep tree in memory for subsequent requests. We need a long running daemon process.

3 contenders:

1. Java
2. Python
3. PersistentPerl (SpeedyCGI)

And the winner is

Java wins in the performance category. But no one wanted to maintain a Java app.

Python is a close second, very similar in performance and much easier to maintain

Perl failed miserably and has been banned from future competitions for bad behavior (no one wanted to maintain Perl)

Embrace the Python

Strengths:

- Nested tree structure and lookup algorithm ported easily
- Code is a wonder to behold
- Tree matching/updating in < 50 lines of code instead of 100+ in PHP
- Serves about 300 requests/sec
- Inventory is loaded into memory and reloaded periodically

Global Interpreter Lock (GIL)

- CPython
- Allows only the thread that's holding the GIL to run, others wait until they can acquire GIL
- Pretty much means that only one thread can execute at one time
- Also present in Ruby

Effects of the GIL - Code

```
NUM_TASKS = 100000000 # 100 mill

def thread_func(num_tasks):
    while num_tasks > 0:
        num_tasks -= 1

num_threads = int(sys.argv[1])
# subdivide task into num_threads tasks
threads = [
        Thread(target=thread_func, args=(NUM_TASKS // num_threads,))
        for i in xrange(num_threads)
] # create threads
[t.start() for t in threads] # start all threads
[t.join() for t in threads]  # wait for all threads to complete
```

Effects of the GIL - Results

- on Macbook Pro (Core i7, 4 cores)
  - no threads: 7.126 sec
  - 1 thread: 7.544 sec
  - 2 threads: 9.735 sec
  - 4 threads: 10.209 sec
  - 8 threads: 11.253 sec

What the GIL meant for us

- Most WSGI servers in python utilize processes rather than threads to get concurrency benefits because of the GIL
  - ex: gunicorn, uwsgi
  - otherwise you're limited to just 1 CPU
- Inventory is duplicated for each worker process
  - 2 GB x 8 workers = 16 GB
  - As inventory size grew we quickly approached the maximum memory capacity of the instances.
- Restart workers each time we reload inventory

Shared memory?

- Ideally we would like one copy of inventory that all workers can access.

- Possible option: share memory between processes
  - Not meant for large objects or complex data structures
  - Communication between processes is much harder than between threads

First layer of skin

Create an Inventory service in Java that we can interact with from Python using sockets

Strengths:

- We can have just one copy of the inventory, leaving a lot more room for inventory to grow
- We can use all the CPUs from one process

Weakness:

- Socket inter-process communication became a bottleneck

Somebody killed my Python

It became clear that without shared memory Python was no longer the best solution for our problem

Solution: port Python code to Java to combine with Inventory service

Server is now entirely written in Java, code is now more verbose and difficult to maintain, but we can process about 600 requests/sec

Still love for Python?

Absolutely, this is just one project that did not work out. *It's not Python it's us.*

We have another ad server without complex data structures that is using Python with great success.

- Runs under gunicorn w/ gevent for async outbound requests - no callback soup
- More maintainable / easy to extend
- Still very performant

In Conclusion

- We love Python
- We use it for dashboards (flask)
- We use it for servers
- We use it for log processing and jobs
- We only don't use when it's not the best tool for the job
- We wish it did not have the GIL

Questions / Suggestions?

Thank you!