Poor person's parallel processing

/bin/id

- Senior Red Team Analyst at $BIGCORP
- Former Red Teamer, ~~Pentester, Forensic Investigator~~ at $OTHERBIGCORP
- Used to administer $(cat /dev/random)

- Been using Python (almost) daily since 2005
- Wrote network sniffer, password crackers, distributed HeartBleed scanner, ...

Every presentation needs ... a Sun Tzu quote

“The spot where we intend to fight must not be made known; for then the enemy will have to prepare against a possible attack at several different points; and his forces being thus distributed in many directions, the numbers we shall have to face at any given point will be proportionately few.“

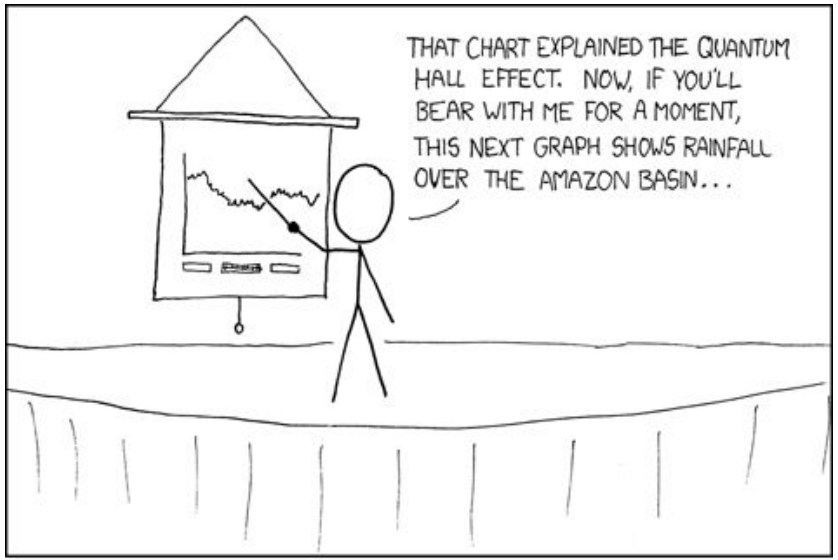
Sun Tzu, Art of War

Every presentation needs ... an image of a cat

```
$ xxd /bin/cat | head

00000000: 7f45 4c46 0101 0100 0000 0000 0000 0000  .ELF............

00000010: 0200 0300 0100 0000 689e 0408 3400 0000  ........h...4...

00000020: d4c2 0000 0000 0000 3400 2000 0900 2800  ........4. ...(.

00000030: 1d00 1c00 0600 0000 3400 0000 3480 0408  ........4...4...

00000040: 3480 0408 2001 0000 2001 0000 0500 0000  4... ... .......

00000050: 0400 0000 0300 0000 5401 0000 5481 0408  ........T...T...

00000060: 5481 0408 1300 0000 1300 0000 0400 0000  T...............

00000070: 0100 0000 0100 0000 0000 0000 0080 0408  ................

00000080: 0080 0408 0cb8 0000 0cb8 0000 0500 0000  ................
```

@JirkaV, PyCon CZ 2017

Every presentation needs ... ~~a meme~~ XKCD slide



https://xkcd.com/365/

IF YOU KEEP SAYING "BEAR WITH ME FOR A MOMENT", PEOPLE TAKE A WHILE TO FIGURE OUT THAT YOU'RE JUST SHOWING THEM RANDOM SLIDES.

@JirkaV, PyCon CZ 2017

I need to....

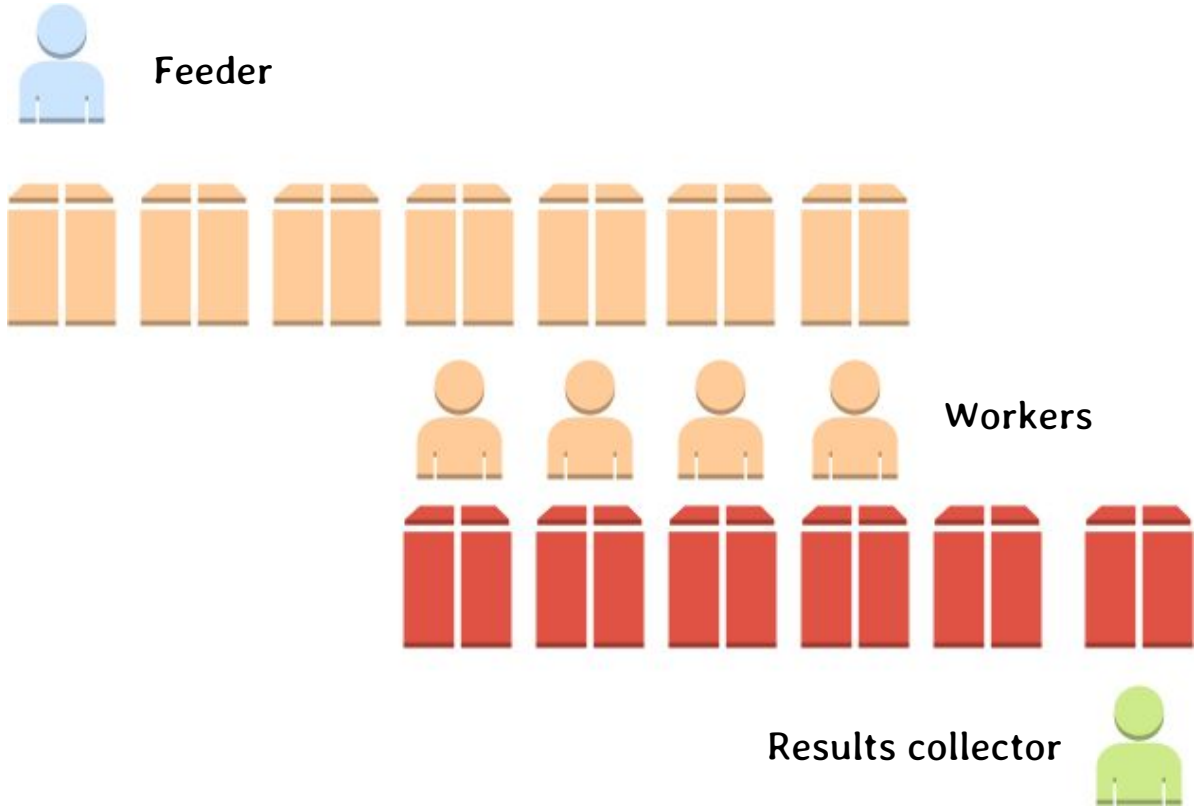
- Resolve 90K DNS names (yes, I know about ADNS & Co.)
- Crack password stored using custom algorithm (can't use JtR, Hashcat, ...)
- Scan a large number of hosts for open ports in a quick, but controlled way
- Do other resource intensive task that can be split into a lot of small chunks

... often where "pip install" or "~~apt-get~~ ~~aptitude~~ apt install" is not an option

import multiprocessing

- External package, became part of Python standard library in 2.6
- Mimics “threading” module, uses independent processes
- Does a lot of complex work in the background, you get a simple interface
- Has a lot of functionality for distributed work (hint: use queues)

~~Demo~~ Example - simple check for prime numbers



Example - feeder

```
def feeder(queue, wanted, stop_marker):
    fed = 0
    while fed < wanted:
        data = randint(1, 10**10)
        queue.put(data)
        fed += 1
    queue.put(stop_marker)
```

Example - worker

```
from prime import is_prime

def worker(input_queue, results_queue, stop_marker):

        while True:

                n = input_queue.get()

                if n == stop_marker:

                        input_queue.put(stop_marker)  # for the other workers

                        results_queue.put(stop_marker)  # for the collector

                        return  # quit the worker

                result = is_prime(n)

                results_queue.put( (n, result) )  # (42, False) .. (73, True)
```

Example - collector

```
def collector(results_queue, stop_marker, workers_count):
    workers_finished = 0
    while workers_finished < workers_count:
        result = results_queue.get()
        if result == stop_marker:
            workers_finished += 1
            continue
        if result[1]:
            print('{} is {}a prime'.format(result[0], '' if result[1] else 'not '))
```

Example - bundling it together

```
import multiprocessing

from worker import worker ; from feeder import feeder ; from collector import collector

WANTED = 100 ; WORKERS_CNT = 15 ; MAX_QUEUE_SIZE = 200 ; STOP_MARKER = '::QUIT::'

input_queue = multiprocessing.Queue(MAX_QUEUE_SIZE) ;  results_queue = multiprocessing.Queue(MAX_QUEUE_SIZE)

for x in range(WORKERS_CNT):

        p = multiprocessing.Process(target=worker, args=(input_queue, results_queue, STOP_MARKER))

        p.start()

f = multiprocessing.Process(target=feeder, args=(input_queue, WANTED, STOP_MARKER))

f.start()

collector(results_queue, STOP_MARKER, WORKERS_CNT)
```

Example run

```
$ python prime_checker.py

[d] Worker 6036 starting...

[d] Worker 4700 starting...

[d] Collector starting...

[d] Feeder starting...

[d] Feeder quittting...

591008101 is a prime

7537556069 is a prime

2461013419 is a prime

[d] Worker stopping...

[d] Collector quitting...
```

I can do that with “X”. How about multiple hosts?

How do we connect processes across network? That must be difficult!

Solution?

I can do that with “X”. How about multiple hosts?

How do we connect processes across network? That must be difficult!

Solution -> add a Manager!

from multiprocessing.managers import BaseManager

multiprocessing - server

```
from multiprocessing.managers import BaseManager ; import queue


class QueueManager(BaseManager):
        pass
work_queue = queue.Queue(QUEUE_SIZE) ; results_queue = queue.Queue(QUEUE_SIZE)
QueueManager.register('work_queue', lambda:work_queue) ; QueueManager.register('results_queue', lambda:results_queue)
m = QueueManager(address=(LISTEN_ADDRESS, PORT), authkey=AUTHKEY)
server = m.get_server()
s.serve_forever()
```

multiprocessing - feeder example

```
from multiprocessing.managers import BaseManager

class QueueManager(BaseManager):

        pass

QueueManager.register('work_queue')

manager = QueueManager(address=(SERVER_ADDRESS, PORT), authkey=AUTHKEY)

manager.connect()

work_q = manager.targets_queue()

while True:

        work_q.put( somework )
```

multiprocessing - worker example

```
...

QueueManager.register('work_queue') ; QueueManager.register('results_queue')

manager = QueueManager(address=(SERVER_ADDRESS, PORT), authkey=AUTHKEY)

manager.connect()

work_q = manager.targets_queue() ; results_q = manager.results_queue()

while True:

        queue_elem = input_q.get_nowait()

        if queue_elem == STOP_MARKER:

                input_q.put(STOP_MARKER)  # for other workers, may take a while

        results_q.put( do_some_work(queue_elem) )
```

I have a question ...

I have a question ...

... or I know how to do it better!

'!tgjivnxiynredtgsuivly frqohfr zuoofyr hkfndaehbT'[::-2]