PyWAW #42

Concurrency in Python 4k

Rodolfo Carvalho
Lead Developer @ Base

17 Nov 2014

#42

BASE

Rep Performance Dashboard

May 01, 2014 → Sep 30, 2014 | Filters

Comparing to: Tyler Black's Team

Ryan Thomas

189% quota attainment for period

$94,500 sold / $50,000 quota

	Total Sales	Avg. Deal Size	Deals Sold
	$370,800	$74,160	5
Team average:	$186,520	$37,304	5

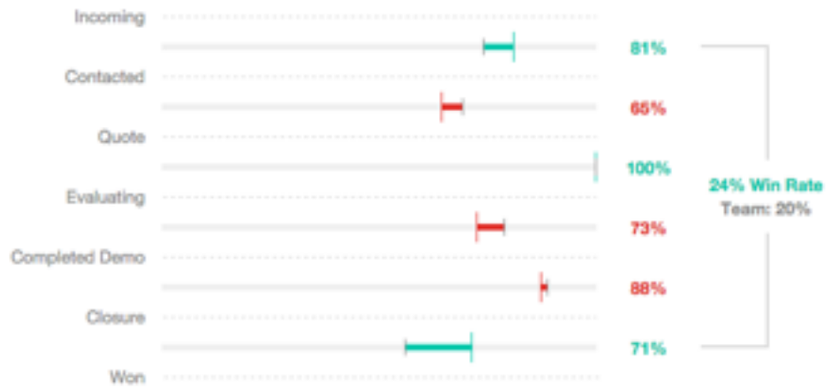
Quota Attainment

September 2014

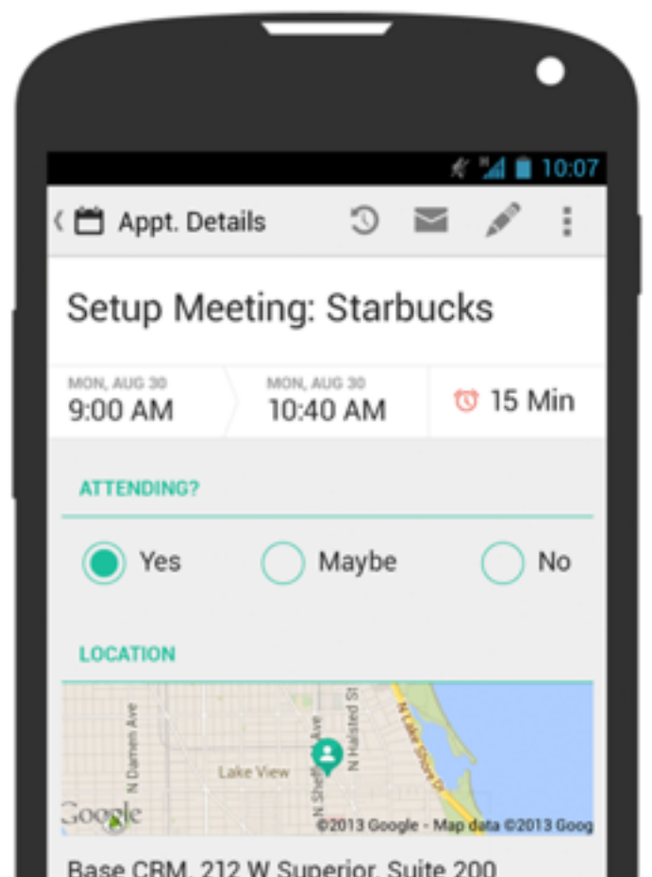
$94,500 / 189%

Quota: $50,000

Conversion Rates



Loss Reasons



complex made easy

focus

Concurrency in Py4k

concurrency


04

old tools

- threading
- multiprocessing
- Coroutines via Enhanced Generators (PEP 342, Python 2.5)

Python 3

- all from Python 2.x
- concurrent.futures
- asyncio (Python 3.4)

18.5. asyncio – Asynchronous I/O, event loop, coroutines and tasks

Note: The asyncio package has been included in the standard library on a *provisional basis*. Backwards incompatible changes (up to and including removal of the module) may occur if deemed necessary by the core developers.

New in version 3.4.

structure

Work done

Time

main

func A

func B

func C

func D

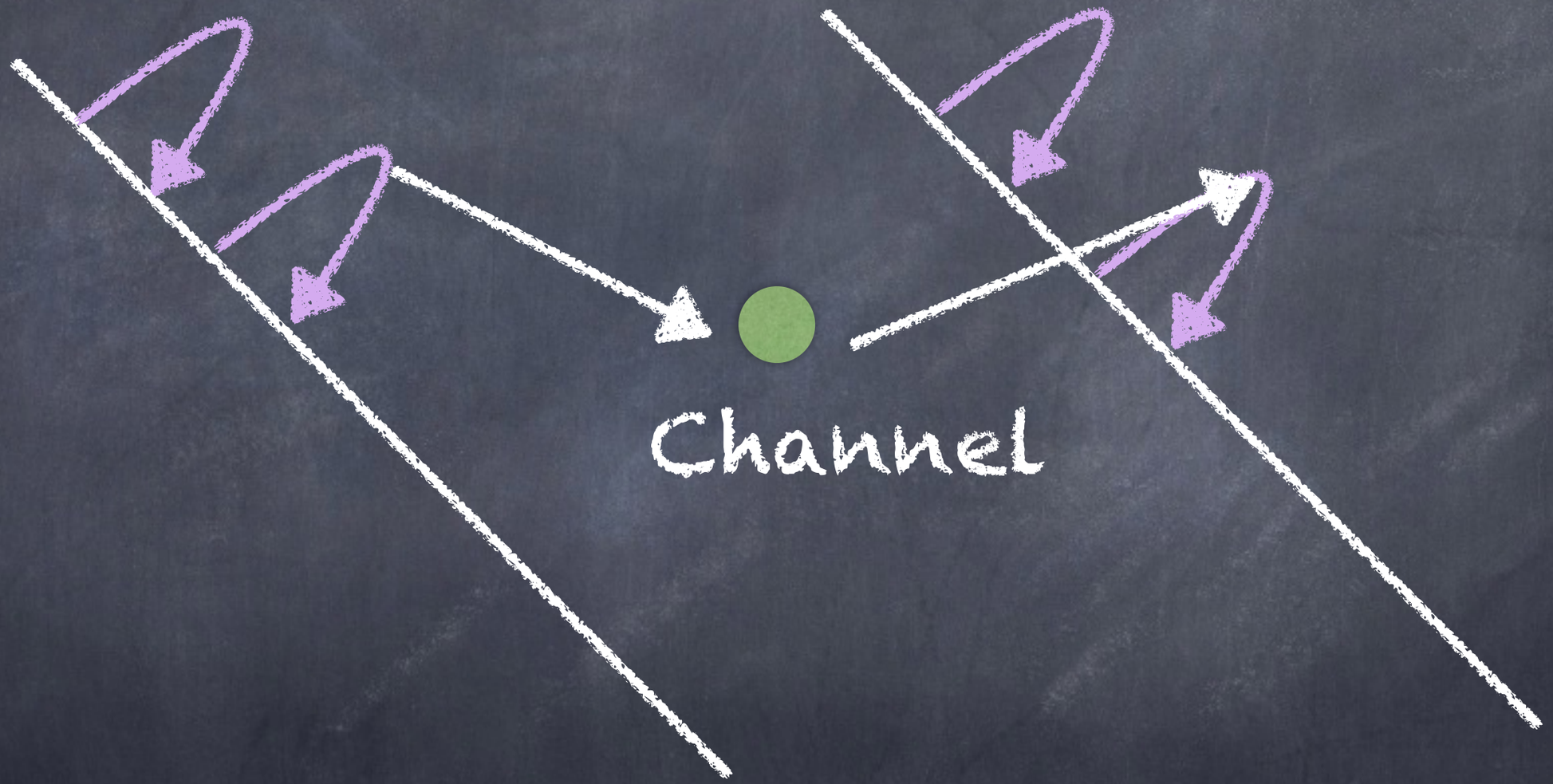
CSP model

Work done

Time

Process A

Process B



Channel

concurrency primitives

- goroutines
- channels
- select

goroutines

```
func MightTakeLong() {
        // ...
}

MightTakeLong()

go MightTakeLong()
```

channels

```
// Declaring and initializing
c := make(chan int)

// Sending on a channel
c <- 1

// Receiving from a channel
// The "arrow" indicates the direction of data flow
value = <-c
```

select

```
func fibonacci(c, quit chan int) {
    x, y := 0, 1
    for {
        select {
        case c <- x:
            x, y = y, x+y
        case <-quit:
            fmt.Println("quit")
            return
        }
    }
}
```

```
func main() {
    c := make(chan int)
    quit := make(chan int)
    go func() {
        for i := 0; i < 10; i++ {
            fmt.Println(<-c)
        }
        quit <- 0
    }()
    fibonacci(c, quit)
}
```

potentially real example

Concurrent search

```
func Google(query string) (results []Result) {
    c := make(chan Result)
    go func() { c <- Web(query) } ()
    go func() { c <- Image(query) } ()
    go func() { c <- Video(query) } ()

    for i := 0; i < 3; i++ {
        result := <-c
        results = append(results, result)
    }
    return
}
```

timeout

```
c := make(chan Result)
go func() { c <- Web(query) } ()
go func() { c <- Image(query) } ()
go func() { c <- Video(query) } ()

timeout := time.After(80 * time.Millisecond)
for i := 0; i < 3; i++ {
    select {
    case result := <-c:
        results = append(results, result)
    case <-timeout:
        fmt.Println("timed out")
        return
    }
}
return
```

Python?

- Python 2.7: gevent
  Greenlets + Queues

- Python 3.4: asyncio
  Coroutines + Queues

- PythonCSP

Other Languages

- occam
- Limbo
- Clojure

1. rodolfocarvalho.net
2. golang.org/s/concurrency-is-not-parallelism
3. golang.org/s/concurrency-patterns
4. www.infoq.com/presentations/core-async-clojure
5. asyncio.org
6. wiki.python.org/moin/Concurrency