Twisted

Présentation & Usecases

Twisted?

- Framework pour applis réseau
- 100% Python, quelques optimisations en C
- Projet opensource stable, bien maintenu
- Programmation asynchrone

Asynchrone?

- Synonymes : “event-driven”, “non-blocking”
- Toutes les fonctions doivent retourner “rapidement”
- 1 seul thread
- Reactor pattern

Sync vs. Async

```
def google_sync(search)
    html = urllib.urlopen("http://google.com/?q="+search").read()
    return [r[1] for r in re.split("<h3>(.*?)</h3>",html)]

print google_sync("pycon fr")
print google_sync("pycon us")
```

```
def google_async(search)
    async_call = twisted.web.client.getPage("http://google.com/?q="+search")
    async_call.addCallback(gotresults)

def gotresults(html)
    print  [r[1] for r in re.split("<h3>(.*?)</h3>",html)]

google_async("pycon fr")
google_async("pycon us")

twisted.internet.reactor.run()
```

“C’est plus compliqué!”

- Oui mais
- 100 requêtes en parallèle ?

“C’est plus compliqué!”

- Oui mais
- 100 requêtes en parallèle ?

```
results = []

class google_thread(Thread):
    def __init__(self,search):
        self.search = search

    def run():
        html = urllib.urlopen(“http://google.com/?q=”+search”).read()
        results.append([r[1] for r in re.split(“<h3>(.*?)</h3>”,html)])

threads = [google_thread(“pycon fr”), google_thread(“pycon us”)]
[thread.start() for thread in threads]
print results
```

Et là ?

```
results = []

class google_thread(Thread):
    def __init__(self,search):
        self.search = search

    def run():
        html = urllib.urlopen("http://google.com/?q="+search").read()
        results.append("Resultats pour '%s' :" % self.search)
        results.append([r[1] for r in re.split("<h3>(.*?)</h3>",html)])

threads = [google_thread("pycon fr"), google_thread("pycon us")]
[thread.start() for thread in threads]
```

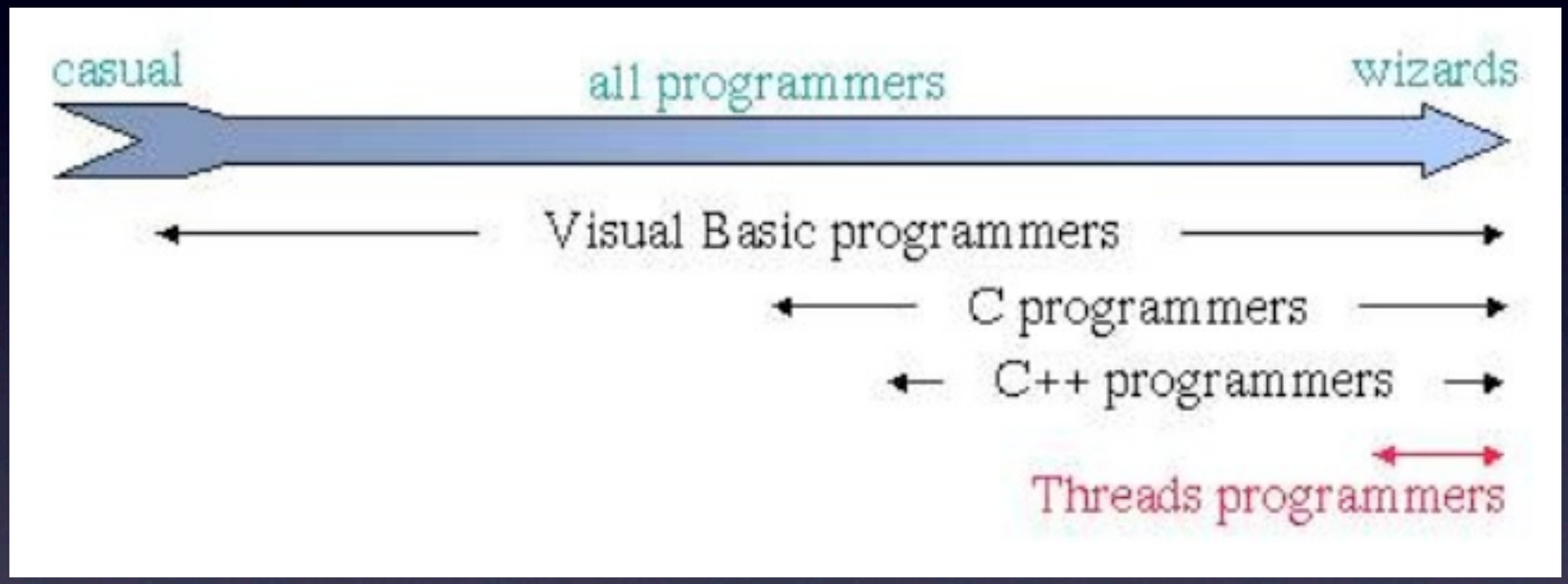
Thread safety

```
results = []

class google_thread(Thread):
    def __init__(self,search):
        self.search = search

    def run():
        html = urllib.urlopen("http://google.com/?q="+search").read()
        acquire_lock(results)
        results.append("Resultats pour '%s' :" % self.search)
        results.append([r[1] for r in re.split("<h3>(.*?)</h3>",html)])
        release_lock(results)

threads = [google_thread("pycon fr"), google_thread("pycon us")]
[thread.start() for thread in threads]
```

Thread safety

- Locks
- Queues
- Semaphores
- ...

Thread safety



Version asynchrone

```
def google_async(search)
    async_call = twisted.web.client.getPage("http://google.com/?q="+search")
    async_call.addCallback(gotresults)

def gotresults(html)
    print [r[1] for r in re.split("<h3>(.*?)</h3>")]

google_async("pycon fr")
google_async("pycon us")

twisted.internet.reactor.run()
```

Version asynchrone

- 1 seul thread!

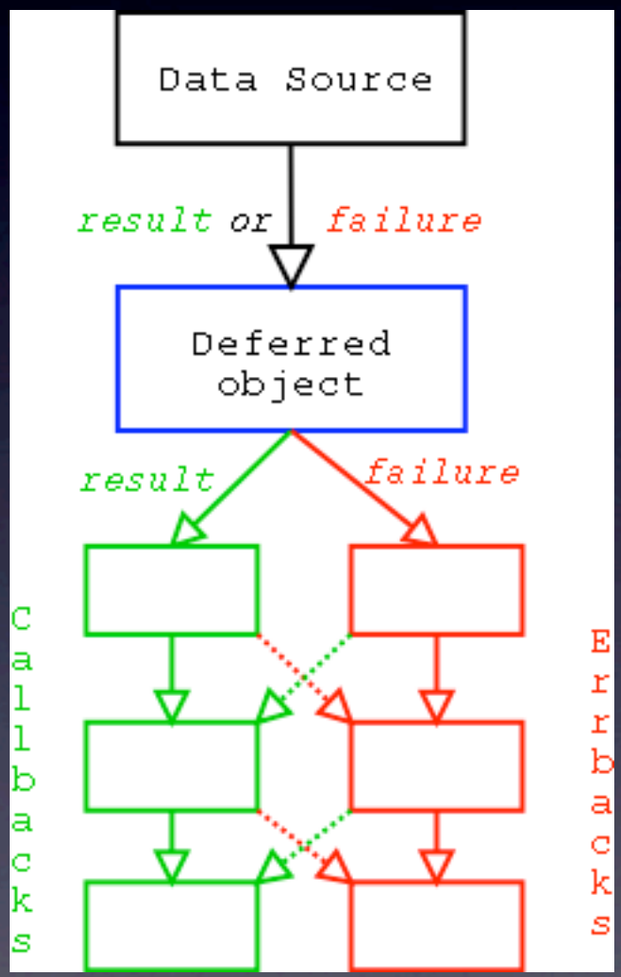
```
results = []

def google_async(search)
    async_call = twisted.web.client.getPage("http://google.com/?q="+search")
    async_call.addCallback(gotresults,search)

def gotresults(html,search)
    results.append("Resultats pour '%s' :" % search)
    results.append([r[1] for r in re.split("<h3>(.*?)</h3>",html)])

google_async("pycon fr")
google_async("pycon us")

twisted.internet.reactor.run()
```

Deferreds

```
def google_async(search)
    async_call = twisted.web.client.getPage("http://google.com/?q="+search")
    async_call.addCallback(gotresults)
```

- Promesse d'un resultat futur
- addCallback
- addErrback
- Chains
- DeferredLists

Deferreds


Data Source
result or failure
Deferred object
result
failure
Callbacks
Errbacks

Reactor

- “Dont call us, we’ll call you”
- Event Loop
- S’occupe d’appeller tous les callbacks
- “Remplace” le GIL, thread switching
- “Pluggable” : select/poll, epoll, GUI, ...

Autres avantages

- Librairie très complète : HTTP, SSH, IRC, DNS, IMAP, Jabber, SMTP, Telnet, ...
- Ne pas réinventer la roue / patterns
- Déploiement rapide d'applis complexes
- Threadpool
- Encapsulation/design
- Moins de surprises

Utilisateurs de Twisted

- Apple
- NASA
- Justin.tv
- Bittorrent / Zope / Freevo / Buildbot / ...
- ???
- Jamendo :)

Twisted chez Jamendo

- Upload servers
- Log servers
- Radio servers
- Widget servers
- Streaming / Download servers

Download servers

- 10k lignes de code
- HTTP “sécurisé”, download queues
- FTP avec virtual filesystem
- Génération de zips à la volée
- Seeds BitTorrent
- Logs UDP / Monitoring
- twisted.manhole

Merci de votre attention!

Sylvain Zimmer
sylvain@jamendo.com
twitter.com/sylvinus