

Python Twisted

Mahendra M
@mahendra



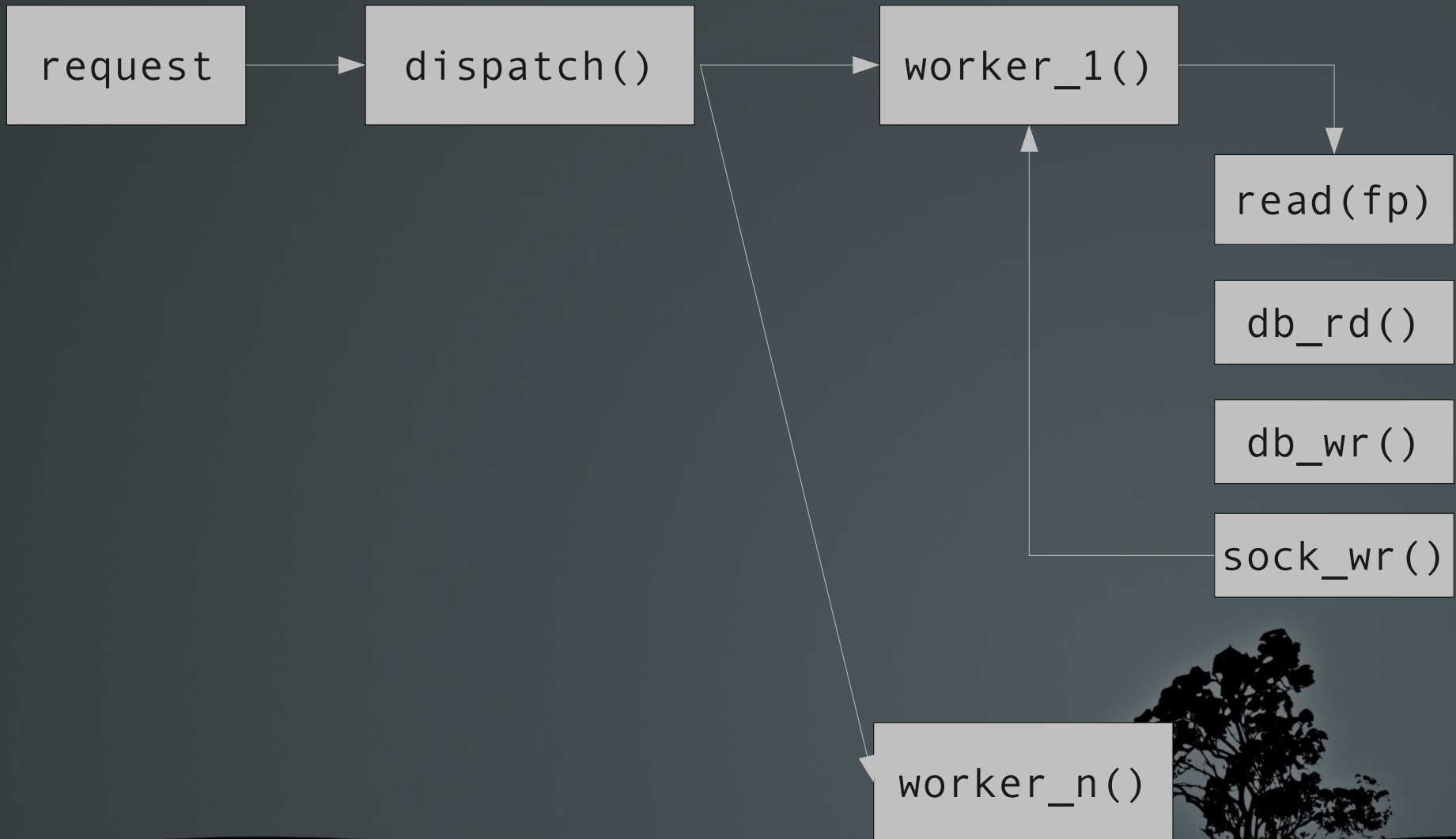
Methods of concurrency

- Workers
 - Threads and processes
- Event driven

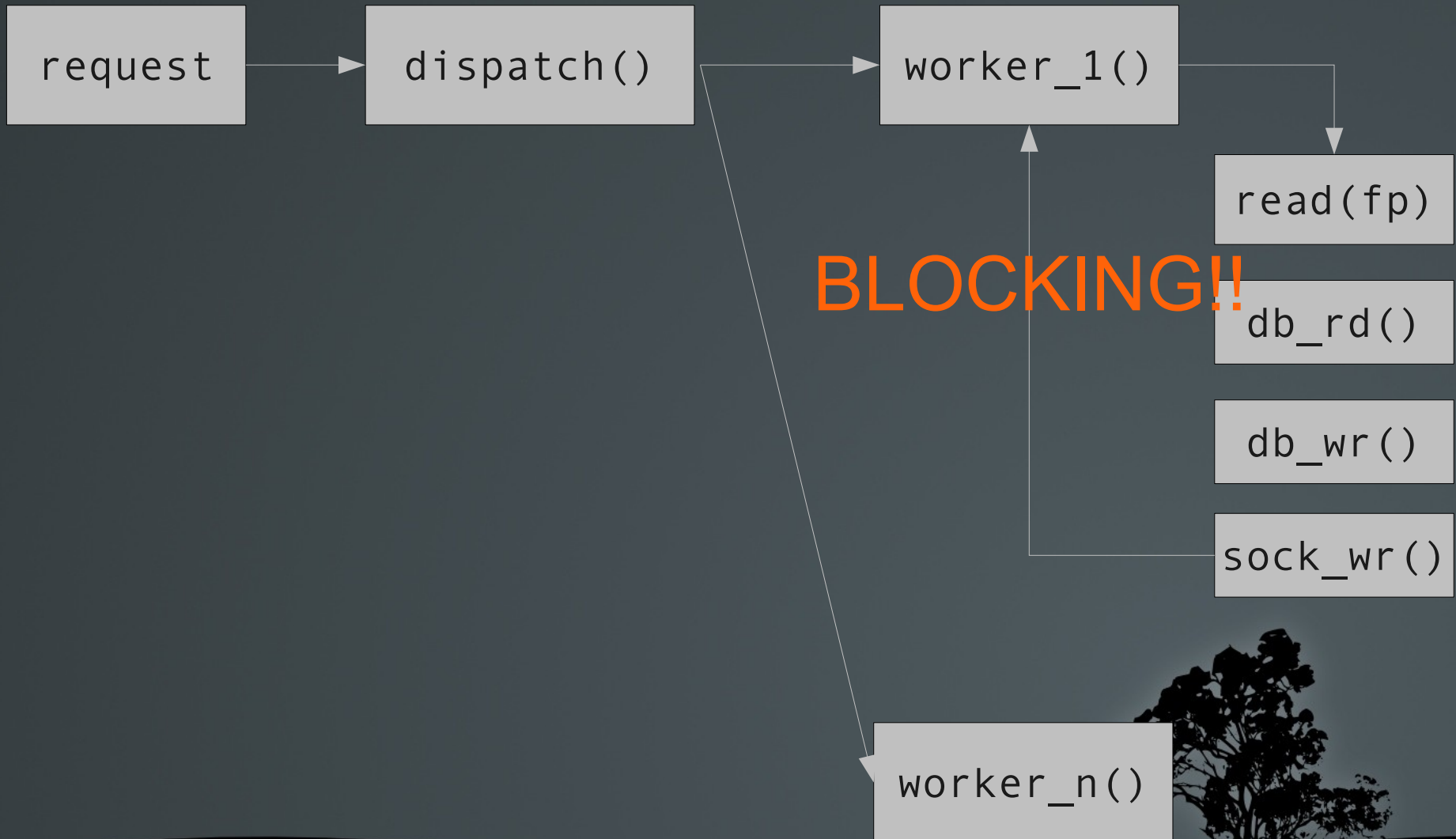
Let us examine this with the case of a web server



Worker model



Worker model



How does it scale ?

- A worker gets CPU when it is not blocking
- When it makes a blocking call, it sleeps till the sys-call is requested
- At this time another worker gets CPU
- Worker might block before it completes it's allocated timeslice.
- This model has worked great and still works great
- Eg: Apache, Squid



Overheads

- Worker management
 - Thread creation
 - Process creation and management
 - Synchronization
 - Scheduling (though this is left to OS)
- More system calls for blocking operations



Event Driven Code

- Non blocking code blocks
- Code execution on events
 - Data on sockets, timer, new connection
- Execution triggered from an event loop
- Full use of CPU timeslice



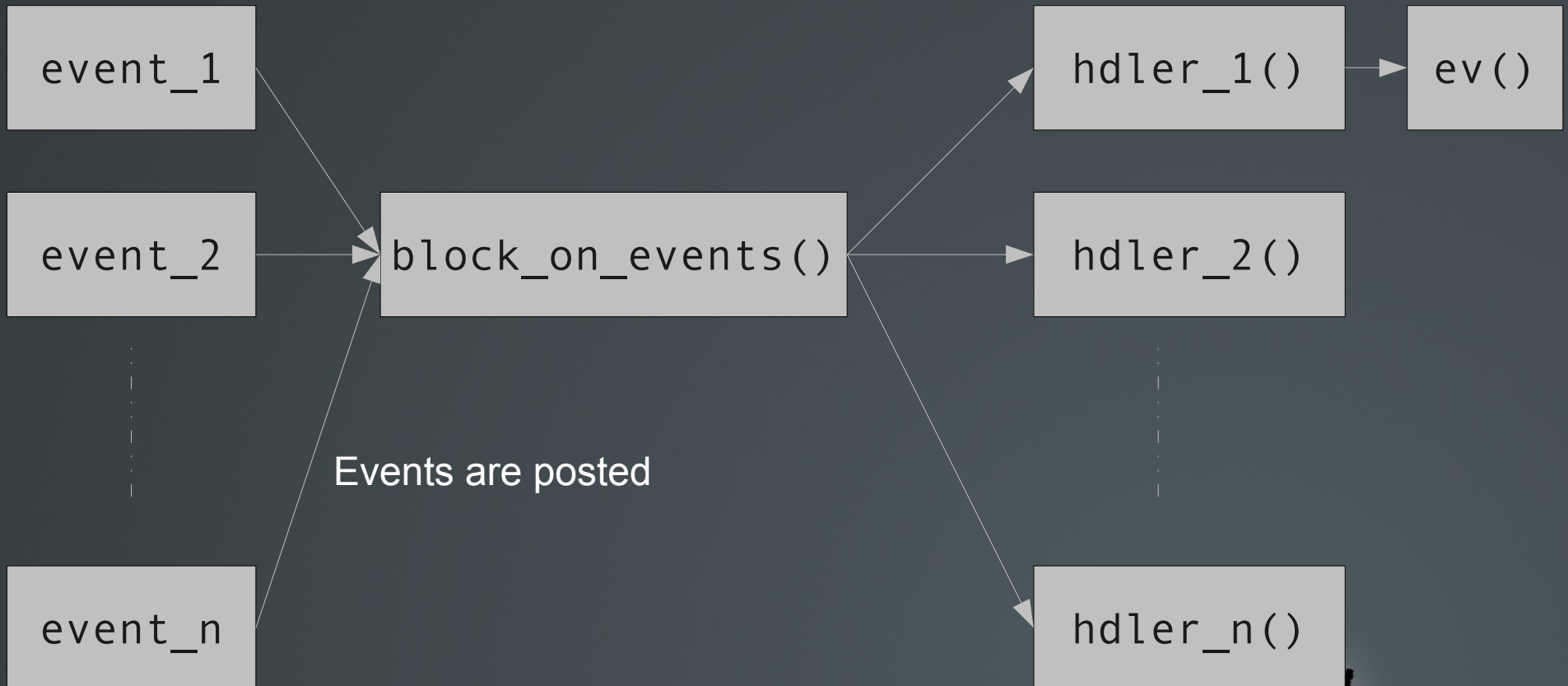
Visually ...

Non blocking functions



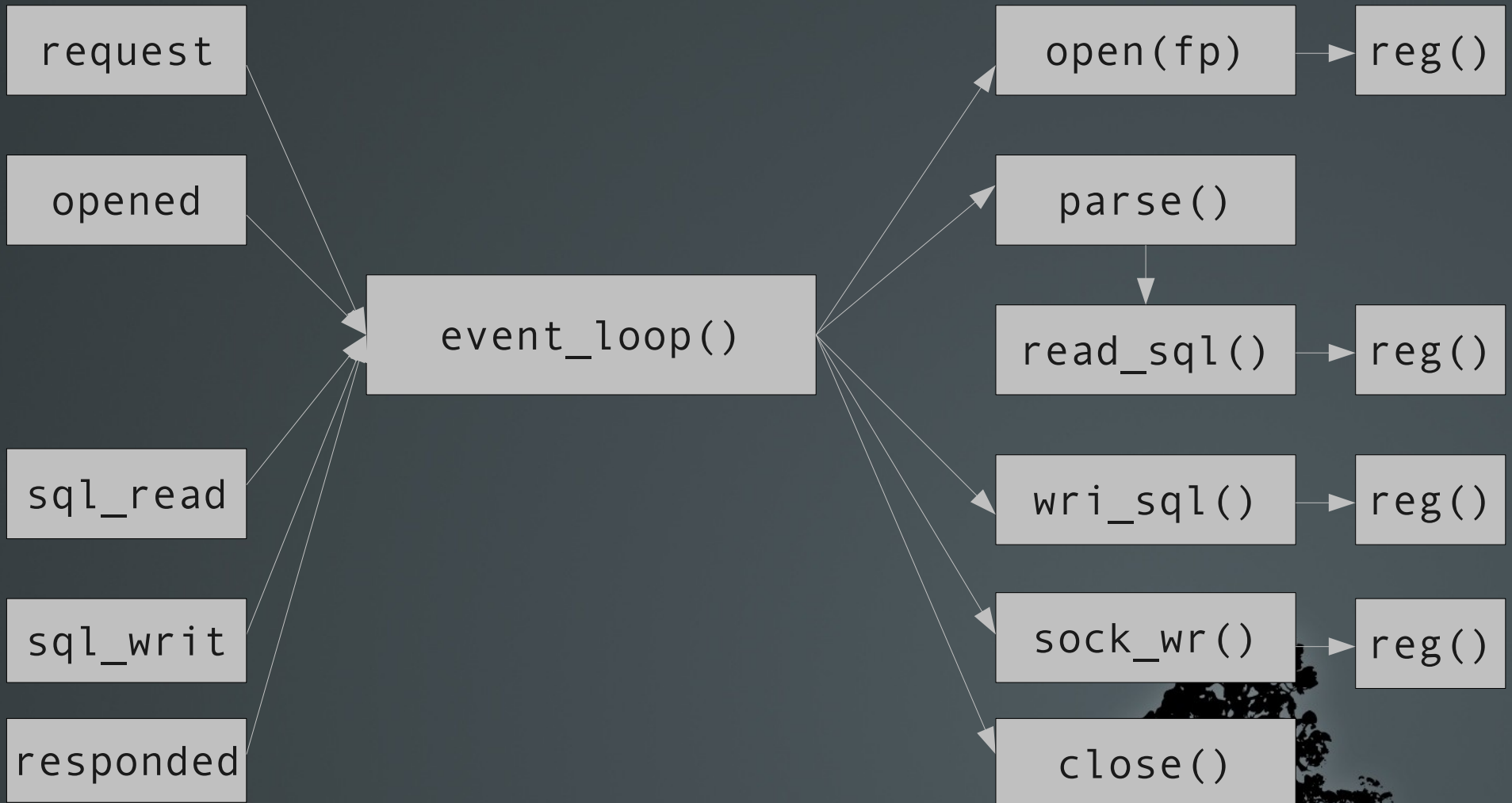
Visually ...

Non blocking functions



Web Server

Non blocking functions



Event Driven Designs

- Nginx, Tornado
- Varnish
- Memcached
- OS support
 - epoll – Linux
 - kqueue – BSDs



Event Driven Libraries

- Libevent
- Python-twisted
- Java NIO
 - Apache MINA, Tomcat (not default)
 - Jetty
- QT



Drawbacks

- Tougher to code, design and maintain
- Workers required to make use of multiple CPUs
- All callbacks must be non-blocking
 - Tough to get non-blocking libraries for all modules
- No isolation
 - A block in any event loop can freeze everything



Python Twisted

- Event driven programming framework
- MIT licensed
- 8 years old and stable
- Support for large number of protocols
 - Client and server support
 - HTTP – SOAP, REST, CouchDB, XMLRPC,
 - Sockets, TCP/IP, Multicast, TLS, SSH, IMAP ...
 - SMTP, NNTP, FTP, Memcached, AMQP, XMPP, ...



Deferred

- The central concept of twisted
- A callback returns a deferred to indicate that the job is not done yet.
- The caller can add callbacks to a deferred.
- The callbacks are invoked then the job is done

Eh ?



Deferred example

```
from twisted.internet import reactor


# Define a success callback
def cb( arg1, arg2 ):
    print "Timeout after %d %s" % ( arg1, arg2 )

# Define an error callback
def eb( error ):
    Print "error %s" % error

# Invoke a non blocking task
deferred = reactor.callLater( 4 )

# Register the callbacks on the returned deferred
deferred.addCallback( cb, 4, 'twisted is great' )
deferred.addErrback( eb )

# Run the event loop
reactor.run()
```




Twisted Server

```
from twisted.internet.protocol import Protocol, Factory
from twisted.internet import reactor

class QOTD(Protocol):
    def connectionMade(self):
        self.transport.write("Welcome\r\n")
        self.transportloseConnection()

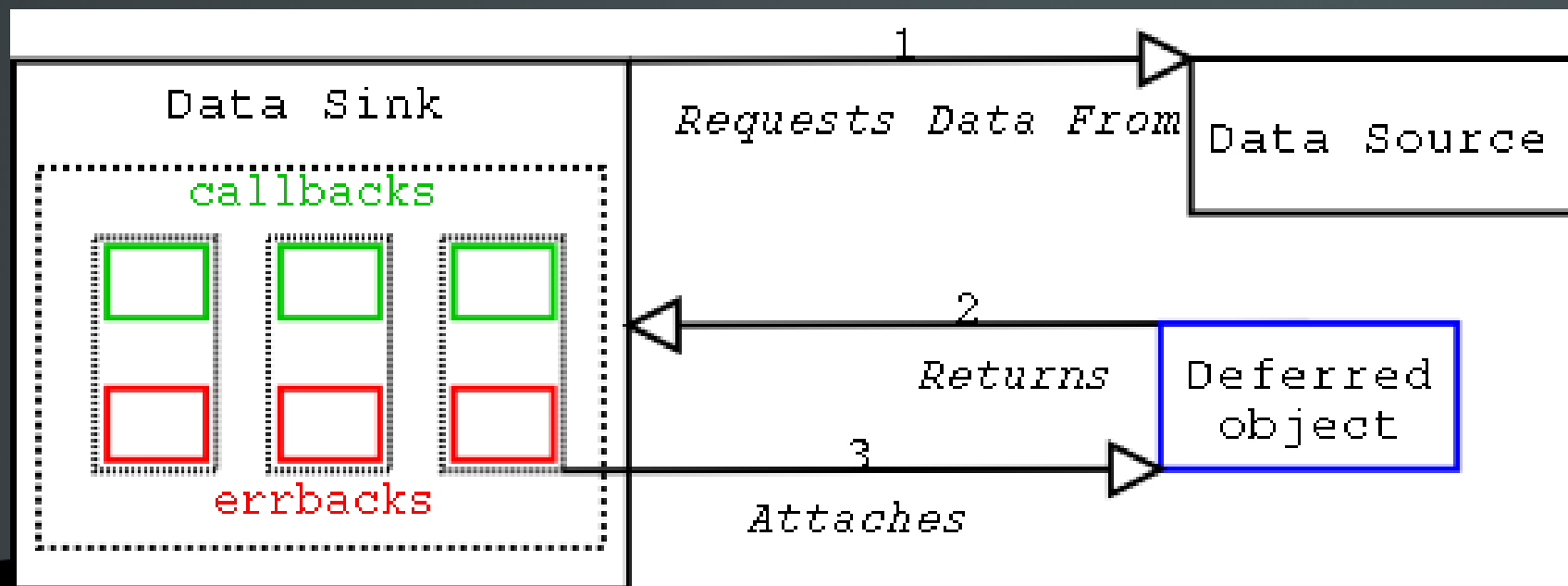
# Next lines are magic:
factory = Factory()
factory.protocol = QOTD

# 8007 is the port you want to run under.
reactor.listenTCP(8007, factory)
reactor.run()
```

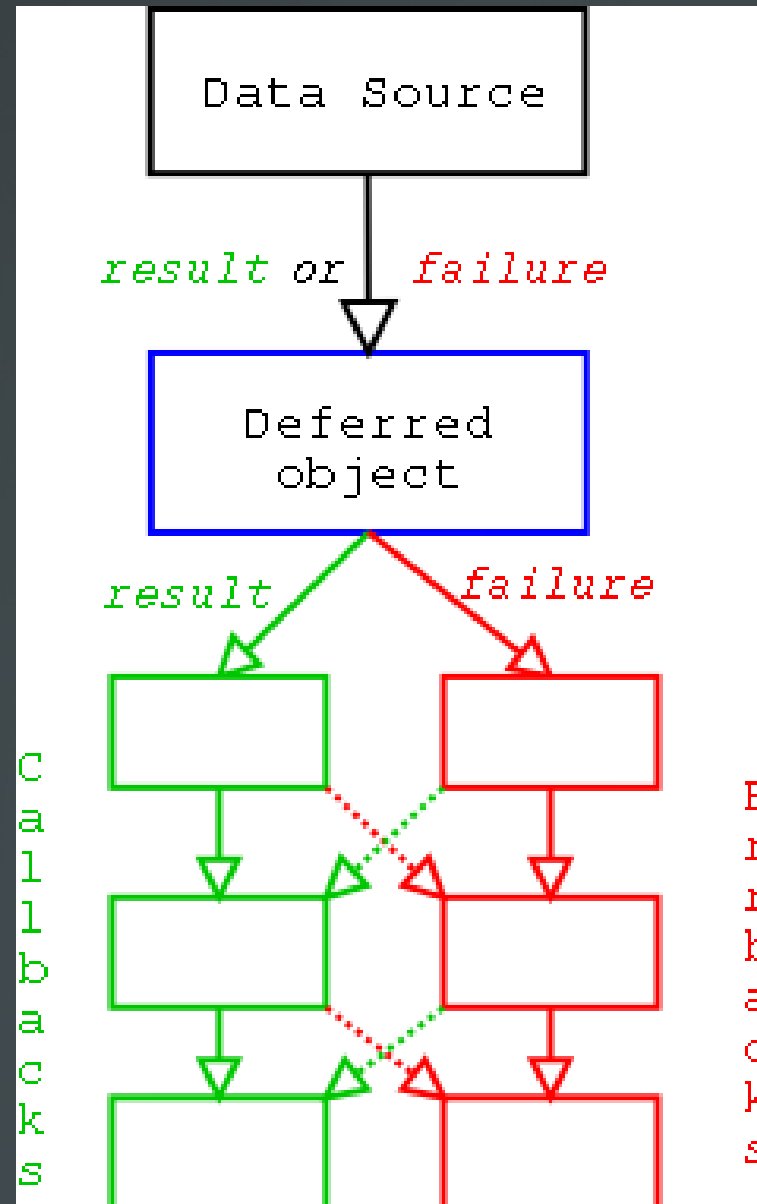


Deferred chaining

- A callback can register and return another deferred
- This callback can return another deferred
- In short we have a deferred chain ...
- Web server example:



Deferred Chaining



Advanced twisted

- Twisted application support
 - Mixing and matching twisted applications
- Command line 'twistd' runner
 - Pre-defined twisted apps
 - Web, telnet, xmpp, dns, conch (ssh), mail, ...
- Plugin architecture
- ADBAPI – for RDBMS



Advanced twisted

- Perspective Broker
 - RPC and object sharing
 - Spreading out servers
- Cred – Authentication framework
- Deferring to threads
- External loops (GTK, QT)
- Streaming support
- MVC framework (Complex. Very complex)



Drawbacks

- Single threaded by design
- Makes use of only one core/CPU
- Need external modules for using multiple CPU
 - Run multiple instances of twisted on a box
 - `num_instances = num_cpus`
 - Use nginx (HTTP), HAProxy for load balancing



Demos



Links

- <http://twistedmatrix.com/trac/>

