제약을 넘어 - Gevent

PYCON KOREA 2014

정민영 @ 비트패킹컴퍼니

발표자

- 정민영
- 미투데이 / 비트패킹컴퍼니
- 07년 Nginx를 만난 후 비동기 덕후
- Python 초보
- kkung@beatpacking.com

오늘의 모든 이야기는

CPython2.X 기준입니다

SHOW TIME

```
def handle_request(s):
  try:
    s.recv(1024)
    s.send('HTTP/1.0 200 OK\r\n')
    s.send('Content-Type: text/plain\r\n')
    s.send('Content-Length: 5\r\n')
    s.send('\r\n')
    s.send('hello')
    s.close()
  except Exception, e:
    logging.exception(e)
```

```
def test():
  s = socket.socket()
  s.setsockopt(socket.SOL_SOCKET,
               socket.SO_REUSEADDR, 1)
  s.bind(('0.0.0.0', 8000))
  s.listen(512)

  while True:
    cli, addr = s.accept()
    logging.info('accept ', addr)
    t = threading.Thread(target=handle_request,
                         args=(cli, ))
    t.daemon = True
    t.start()
```

```
Requests per second:    2099.21
[#/sec] (mean)

Time per request:       487.803
[ms] (mean)
```

```
Requests per second:    4504.64
[#/sec] (mean)

Time per request:       227.321
[ms] (mean)
```

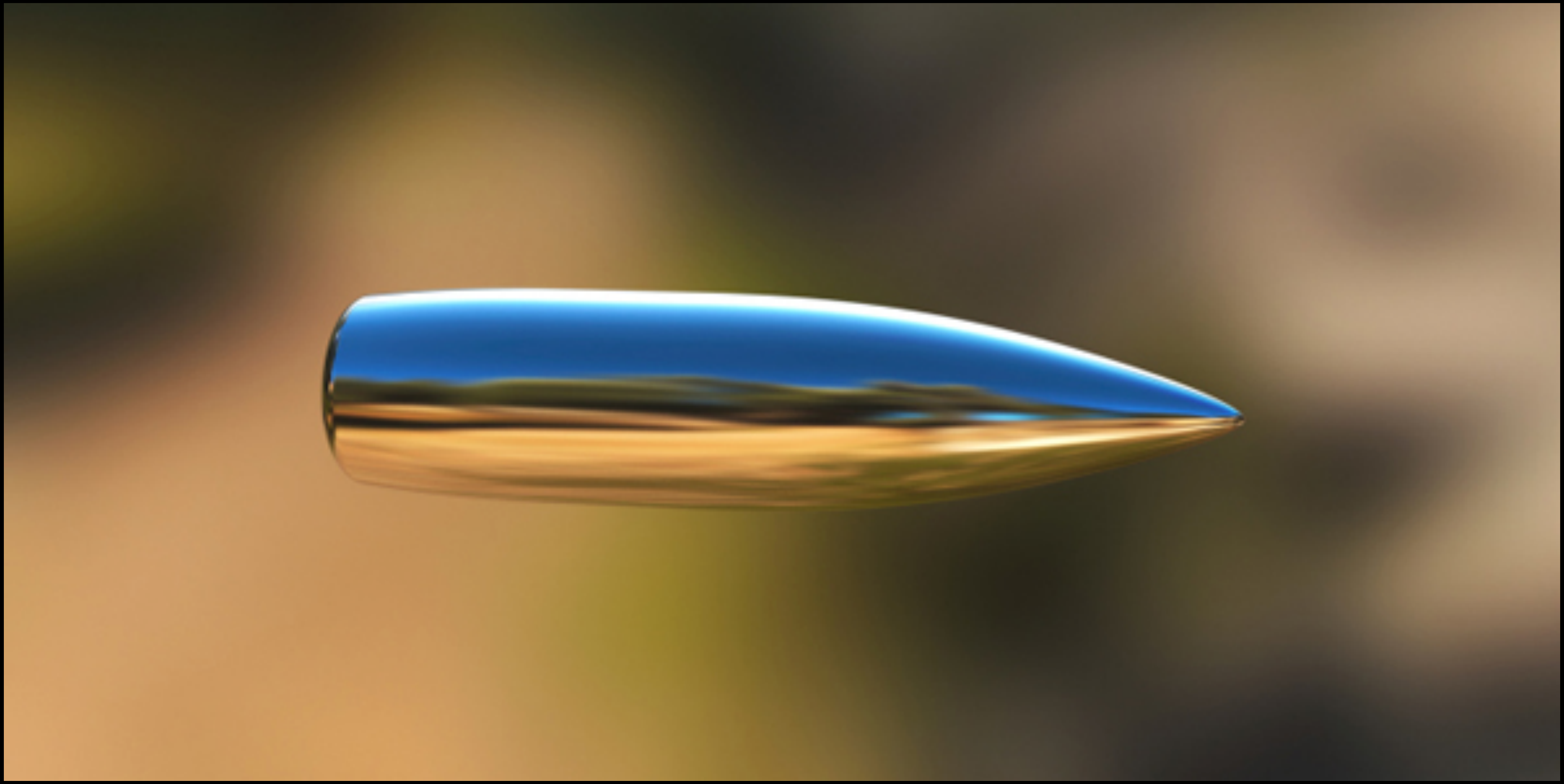
	A	B	
RPS	2099	4504	214%
TPR	487	227	214%

올ㅋ

```
from gevent.monkey
import patch_all
patch_all()
```

단 두 줄로 성능도 두 배

먼저 퇴근하겠습니다!



"제약"

CPU BOUND

수행시간에
CPU가 더 영향이 큰 작업

압축, 정렬, ⋯.

I/O BOUND

수행시간에
I/O가 더 영향이 큰 작업

네트워크, 디스크, ….

대부분의 WEB APP!

Python은 사실상
Single Thread

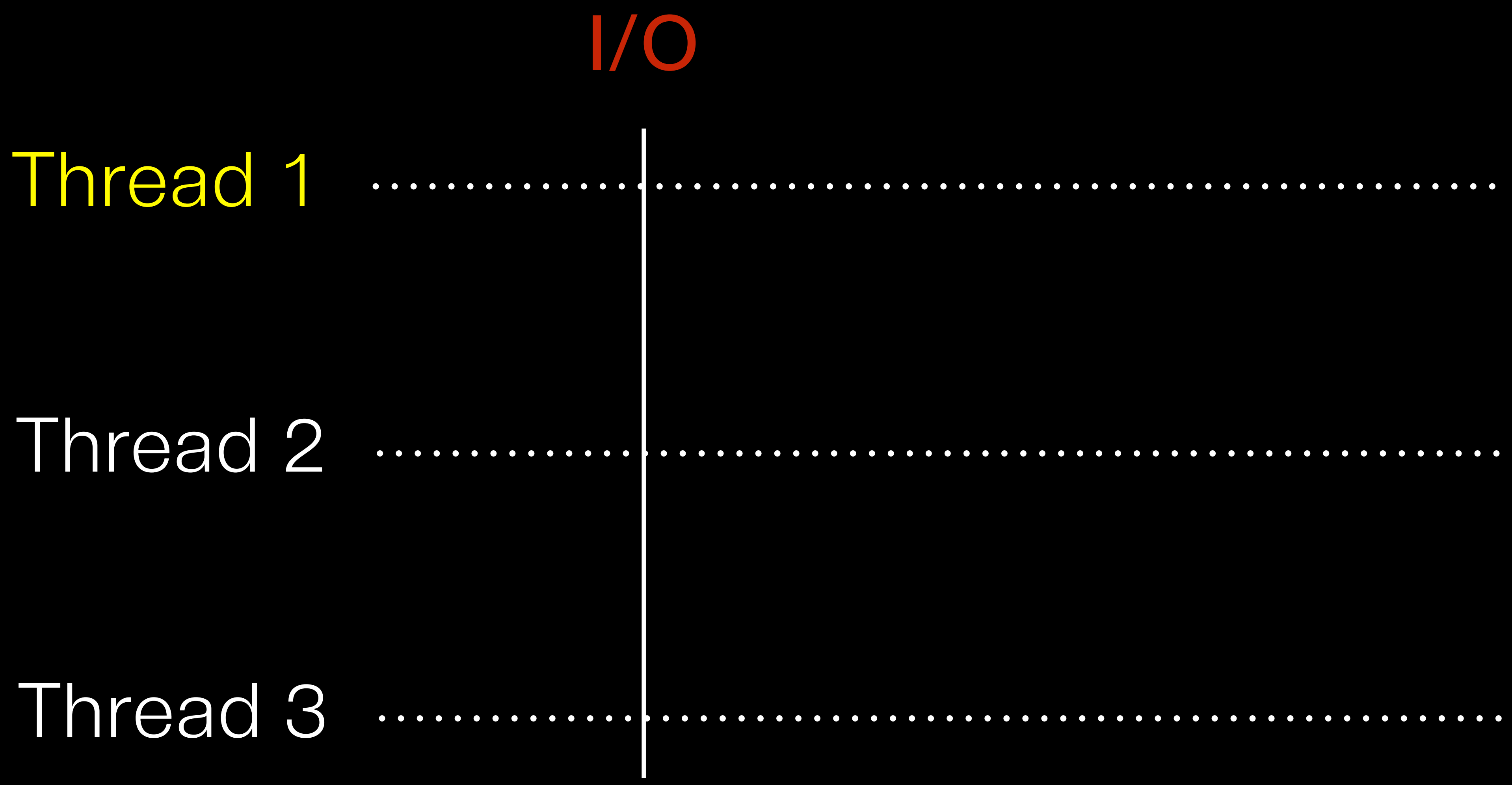
threading
모듈이 있는데요?

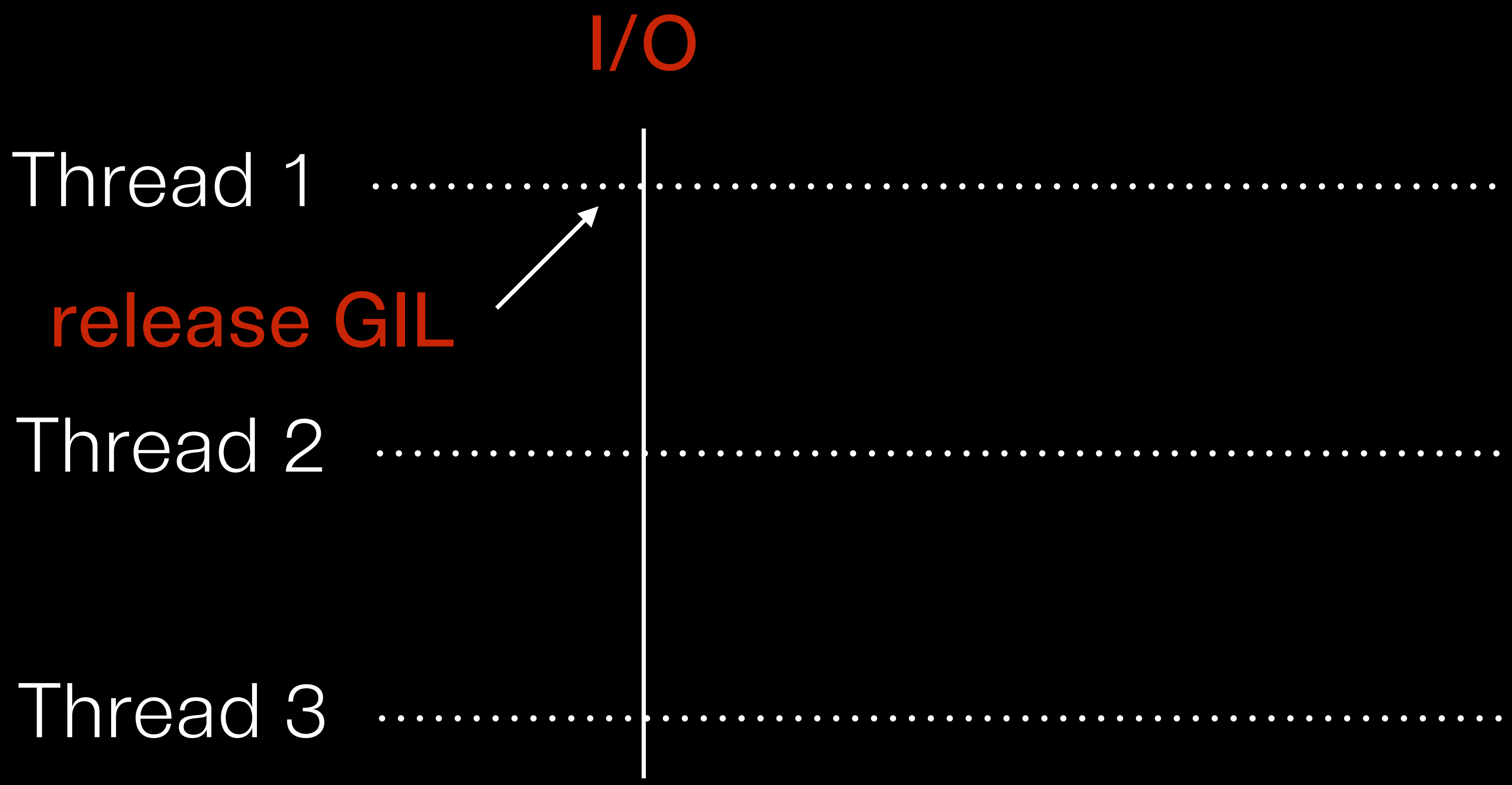
GIL

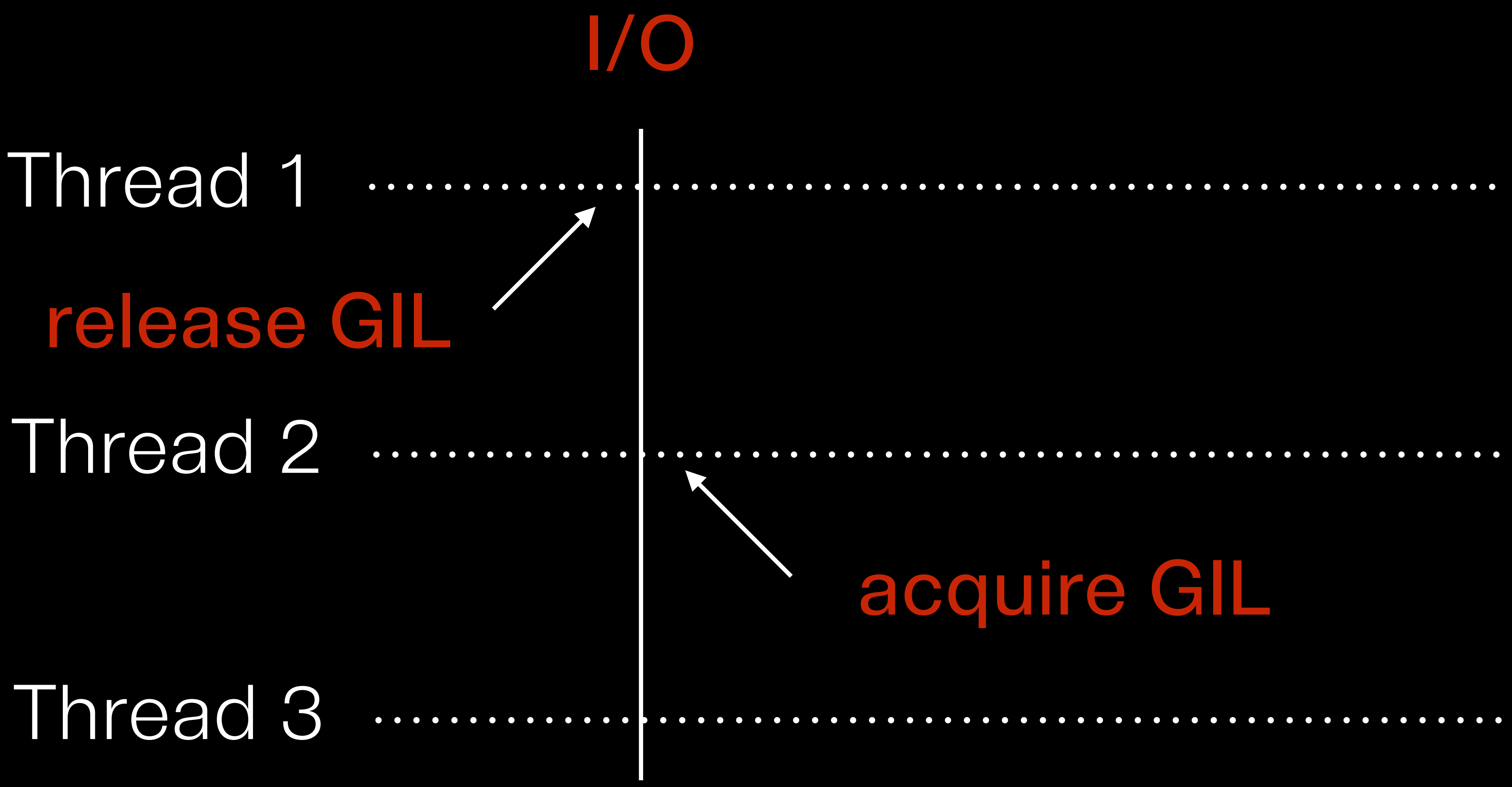
Thread 1

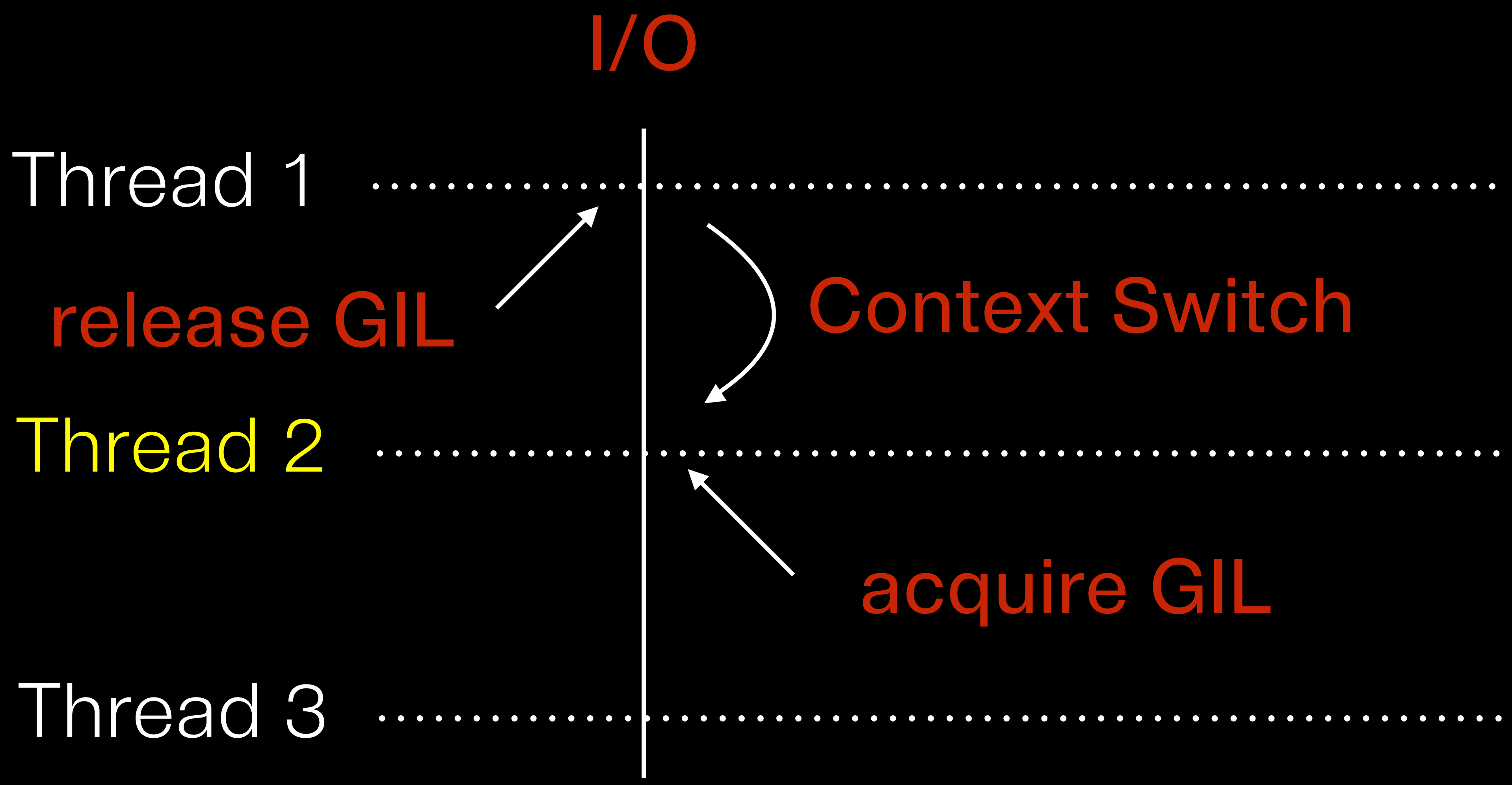
Thread 1
Thread 2

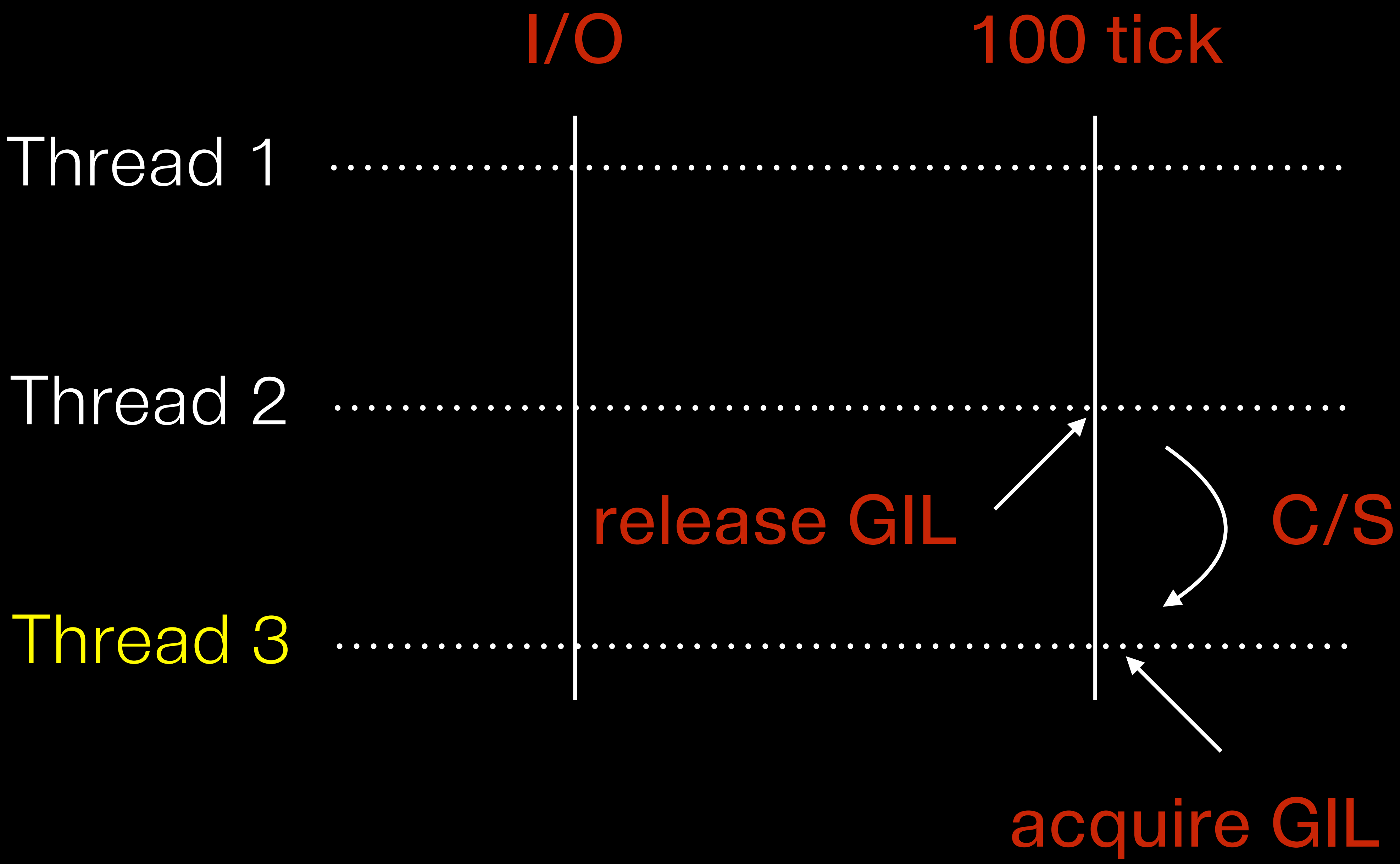
Thread 1
Thread 2
Thread 3


I/O
Thread 1
Thread 2
Thread 3


I/O
Thread 1
release GIL
Thread 2
Thread 3


I/O
Thread 1
release GIL
Thread 2
acquire GIL
Thread 3


I/O
Thread 1
release GIL
Context Switch
Thread 2
acquire GIL
Thread 3


I/O
100 tick
Thread 1
Thread 2
release GIL
C/S
Thread 3
acquire GIL

I/O는 되는거 아닌가요?

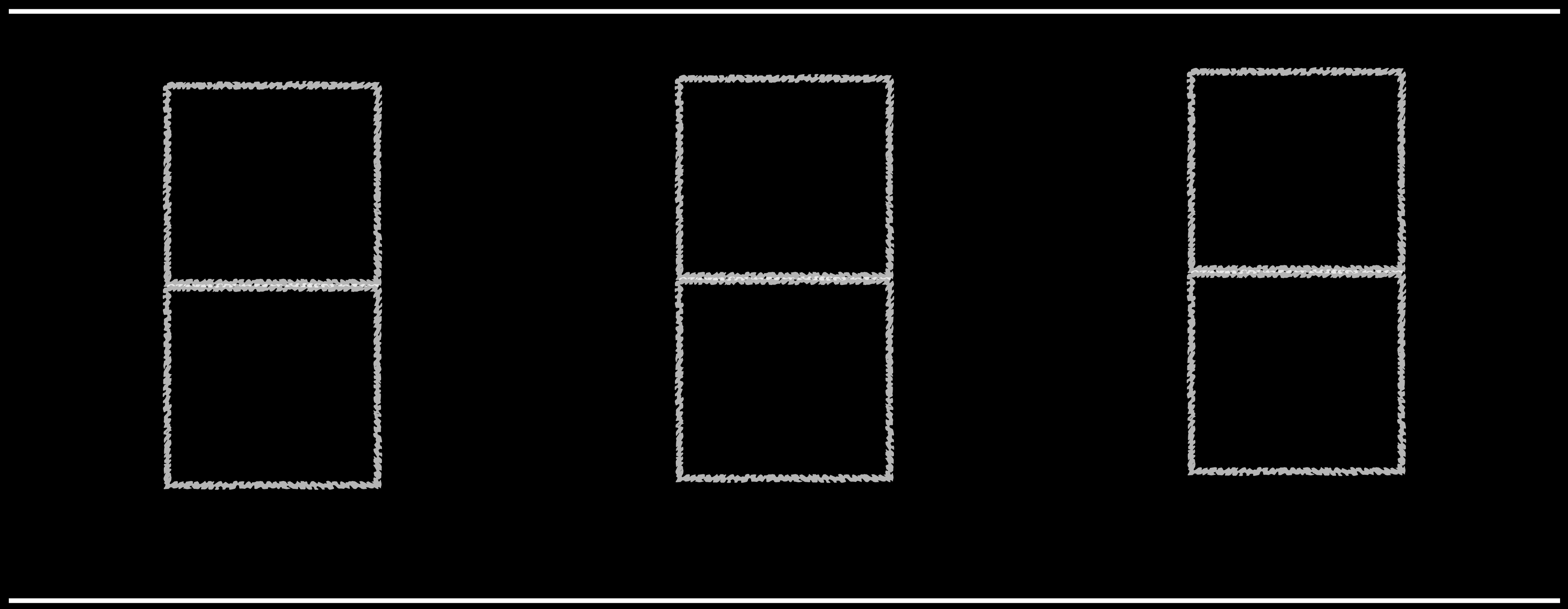
Implicit Scheduling

Thread 실행 순서는
며느리도 모름!

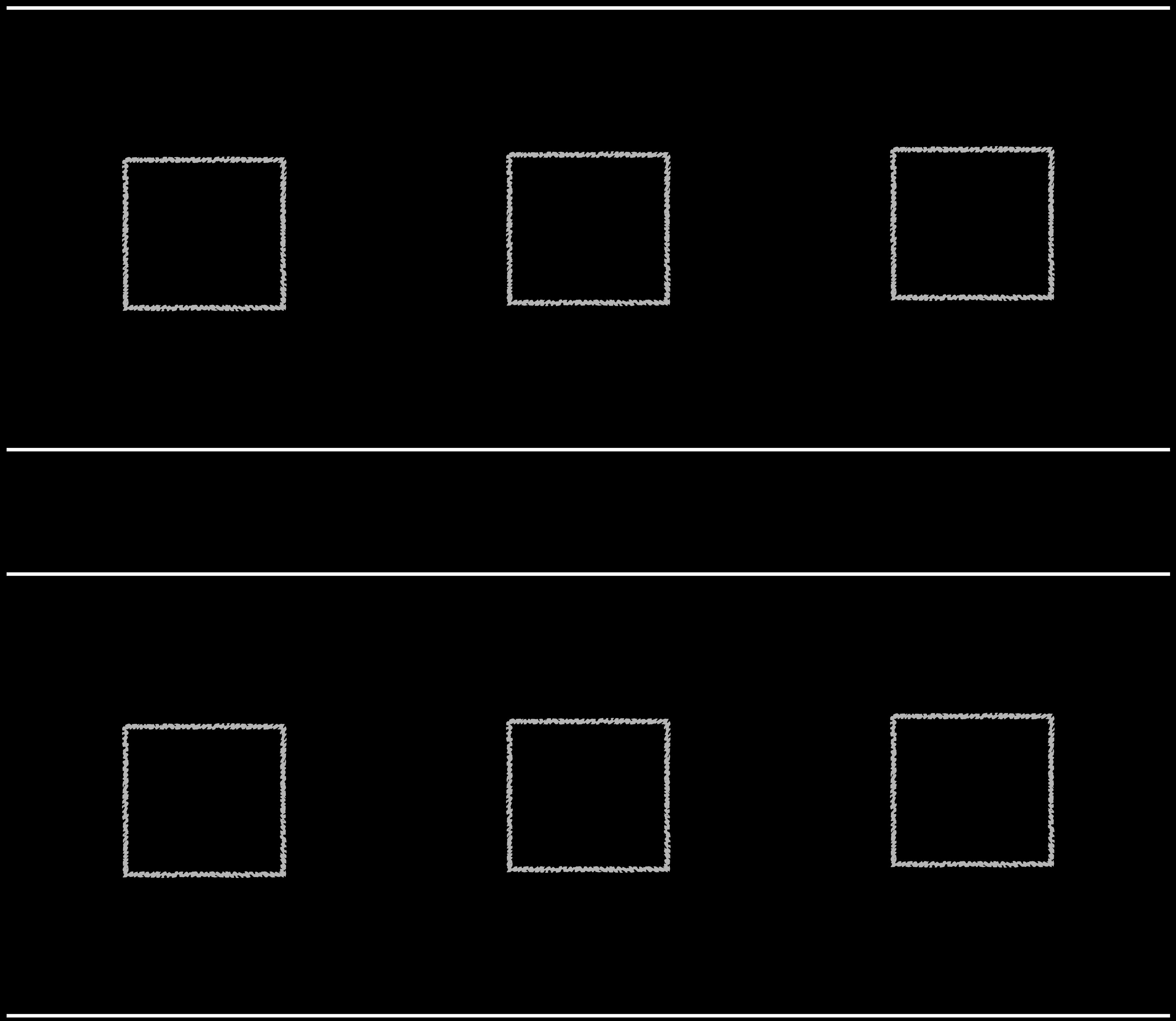
그래서
어쩌라고.

동시성? 병렬성?

동시성 ≠ 병렬성



CONCURRENCY



PARALLELISM

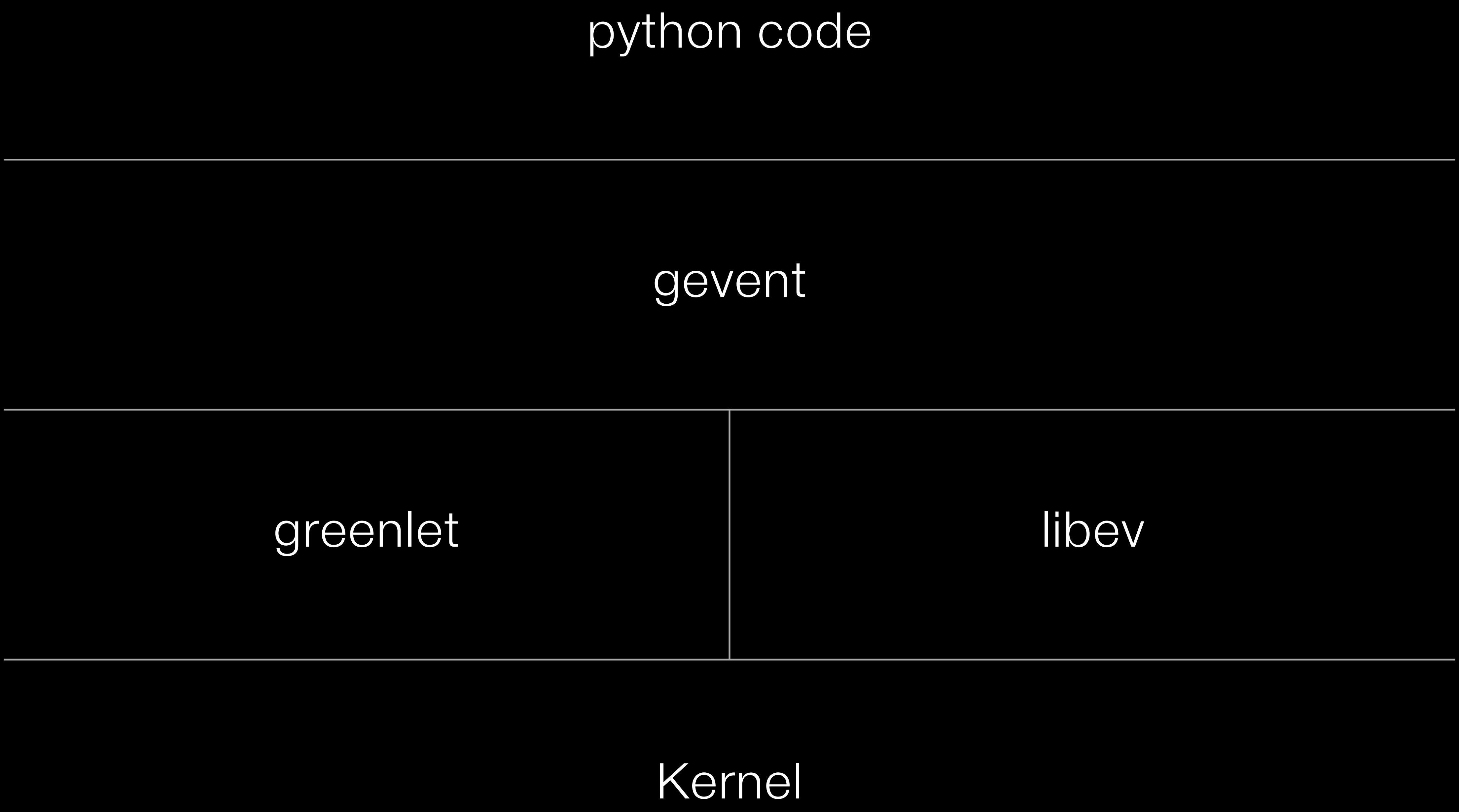
따라서 GIL 때문에

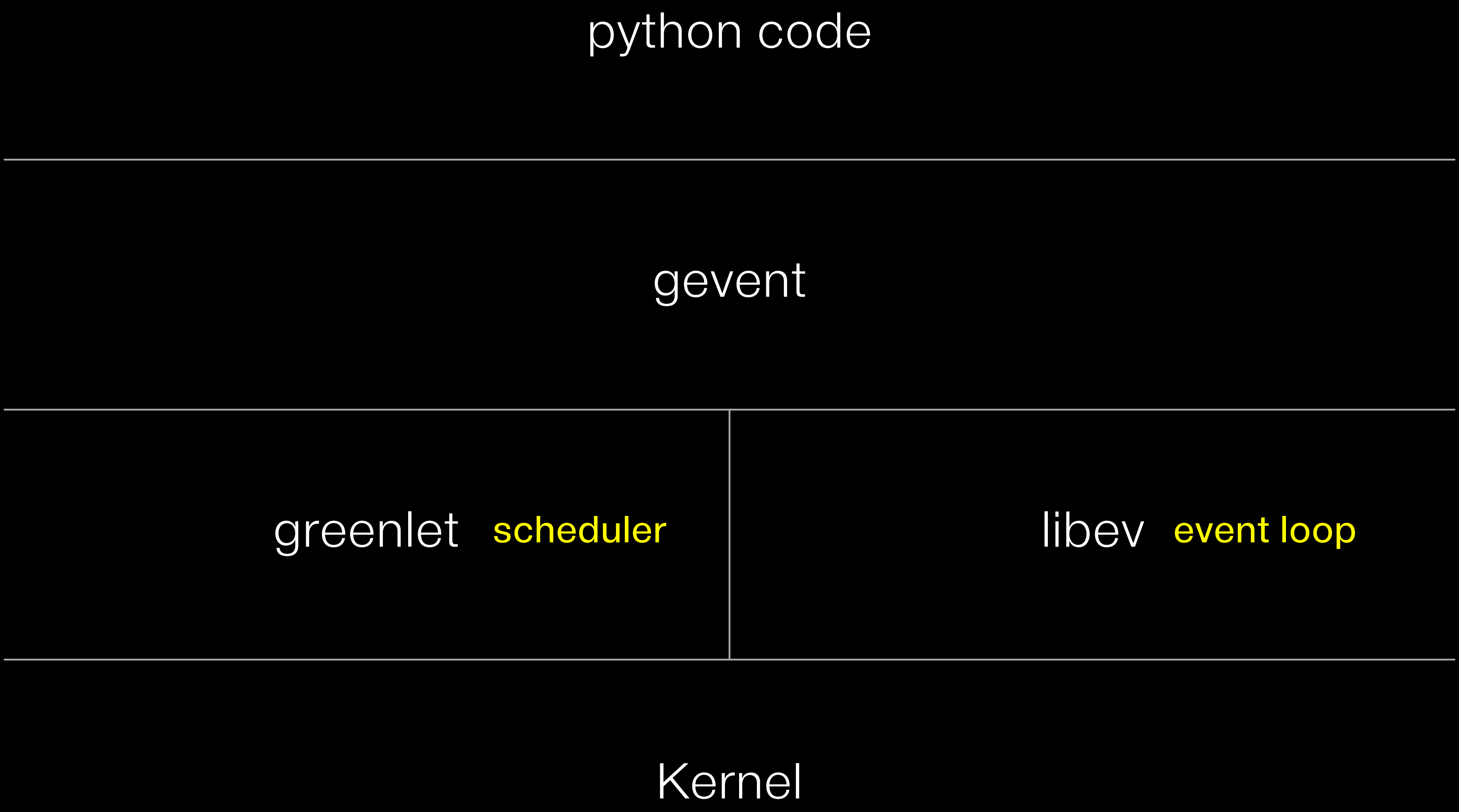
동시성에 집중해야

어떻게?

gevent

scheduler + event loop


python code
gevent
greenlet
libev
Kernel


python code
gevent
greenlet scheduler
libev event loop
Kernel

Greenlet

A “greenlet”, on the other hand, is a still more primitive notion of micro-thread with no implicit scheduling; coroutines, in other words.

coroutine

generator

```python
def hello():
    while True:
        name = (yield)
        print 'Hello %s' % name

c = hello()
c.next()
c.send('kkung')
c.send('PyConKR 2014')
```

```
Hello kkung
Hello PyConKR 2014
```

```python
def hello():
    while True:
        name = (yield)
        print 'Hello %s' % name

c = hello()
c.next()
c.send('kkung')
c.send('PyConKR 2014')
```

```
Hello kkung
Hello PyConKR 2014
```

```
def hello():
    while True:
        name = (yield)
        print 'Hello %s' % name

c = hello()
c.next()
c.send('kkung')
c.send('PyConKR 2014')
```

```
Hello kkung
Hello PyConKR 2014
```

```python
def hello():
    while True:
        name = (yield)
        print 'Hello %s' % name

c = hello()
c.next()
c.send('kkung')
c.send('PyConKR 2014')
```

```
Hello kkung
Hello PyConKR 2014
```

Cooperative Multitasking

명시적인 스케줄링

I/O Bound 최적

Single Thread
스케줄링

너무 많은 일을 하면 X

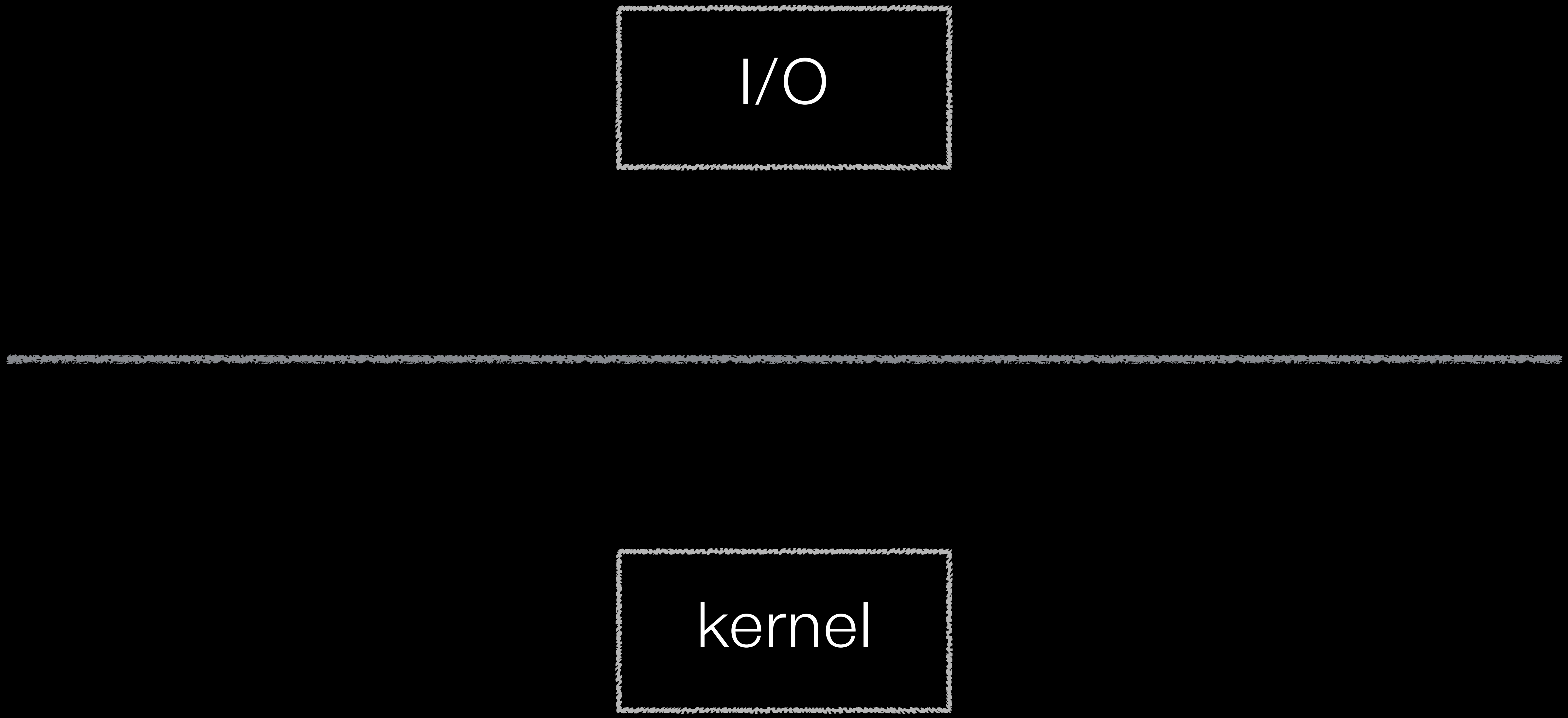
libev

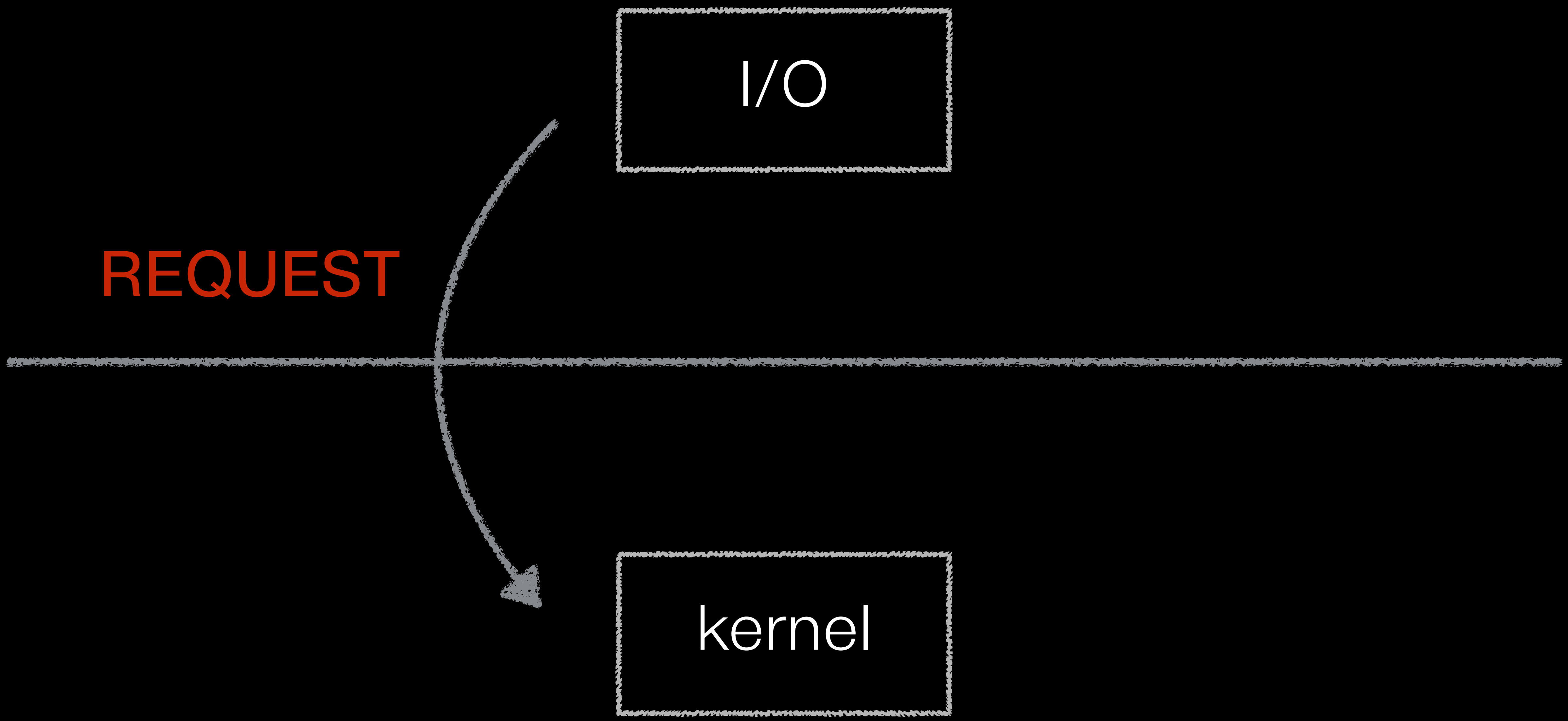
현재 시스템에
가장 적절한 event-loop
시스템 선택

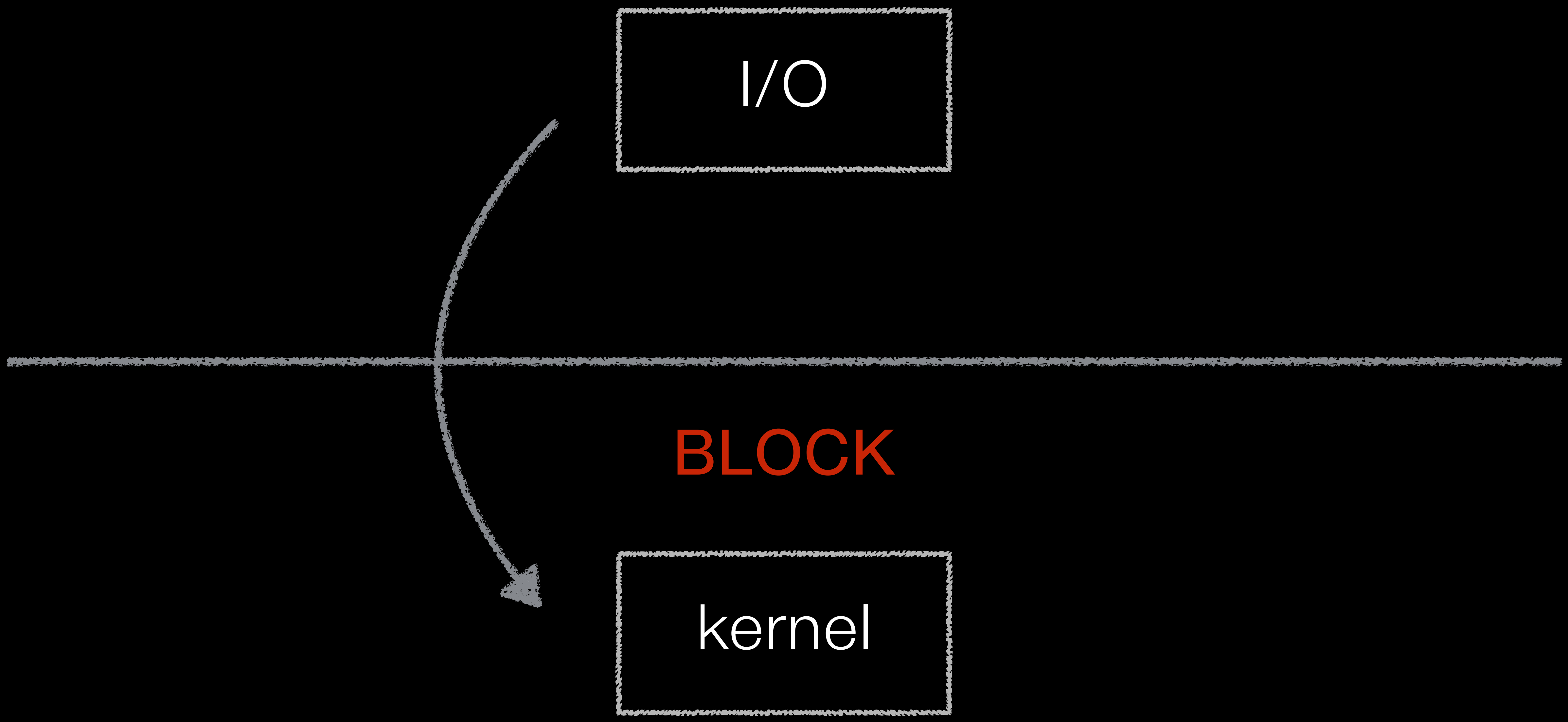
event loop

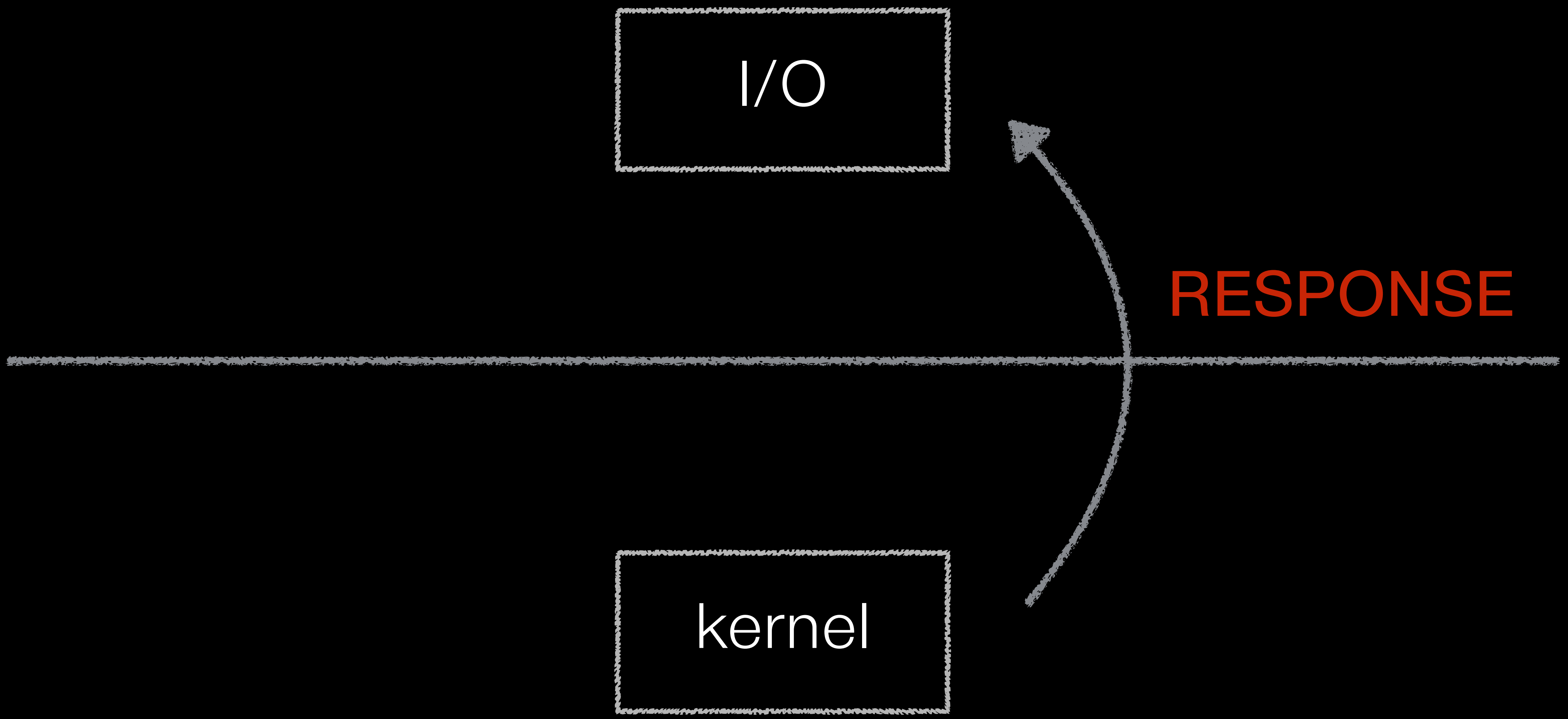
```python
while True:
    events = wait_for_events()
    for event in events:
        handle_event(event)
```

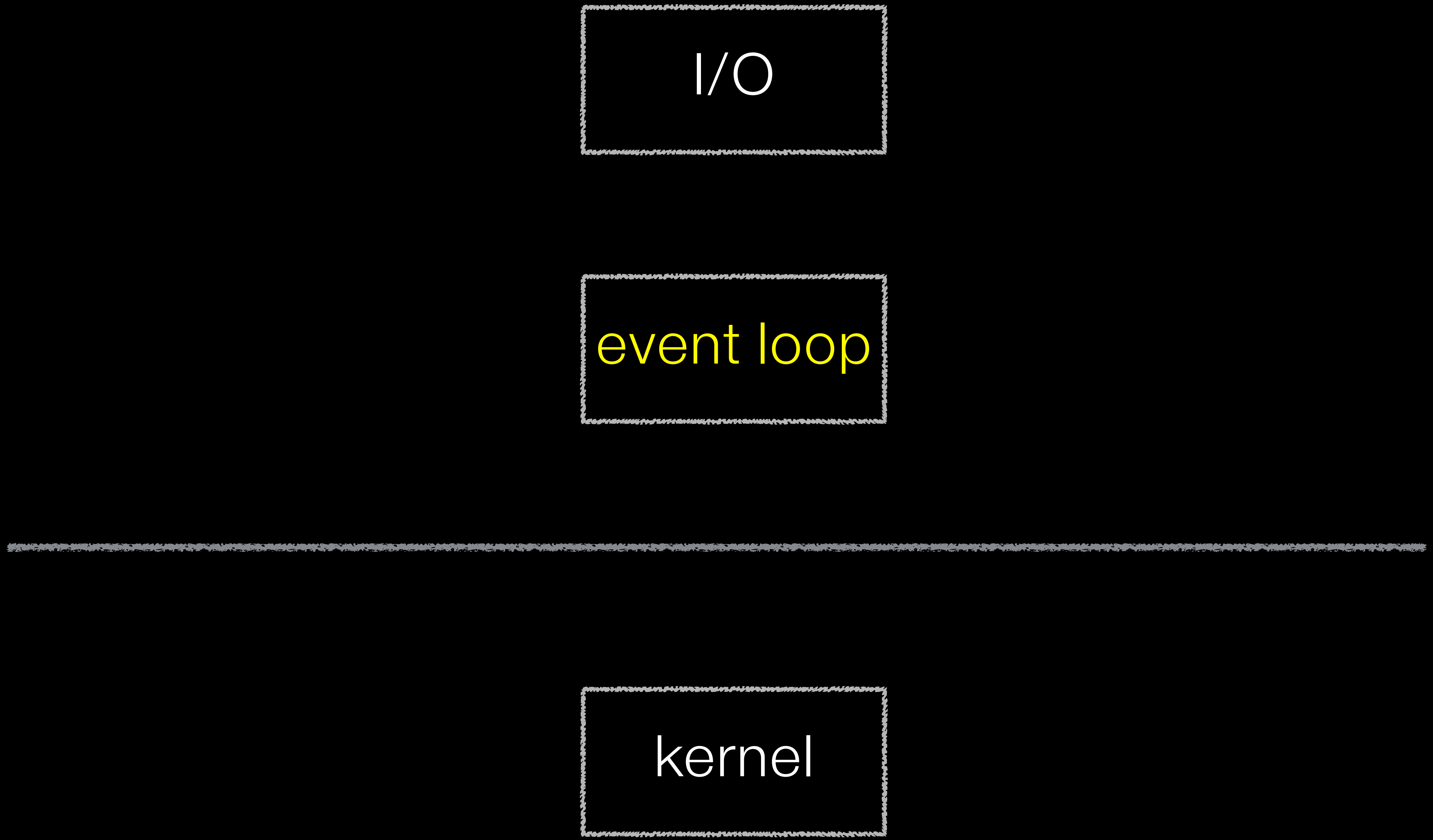
event?

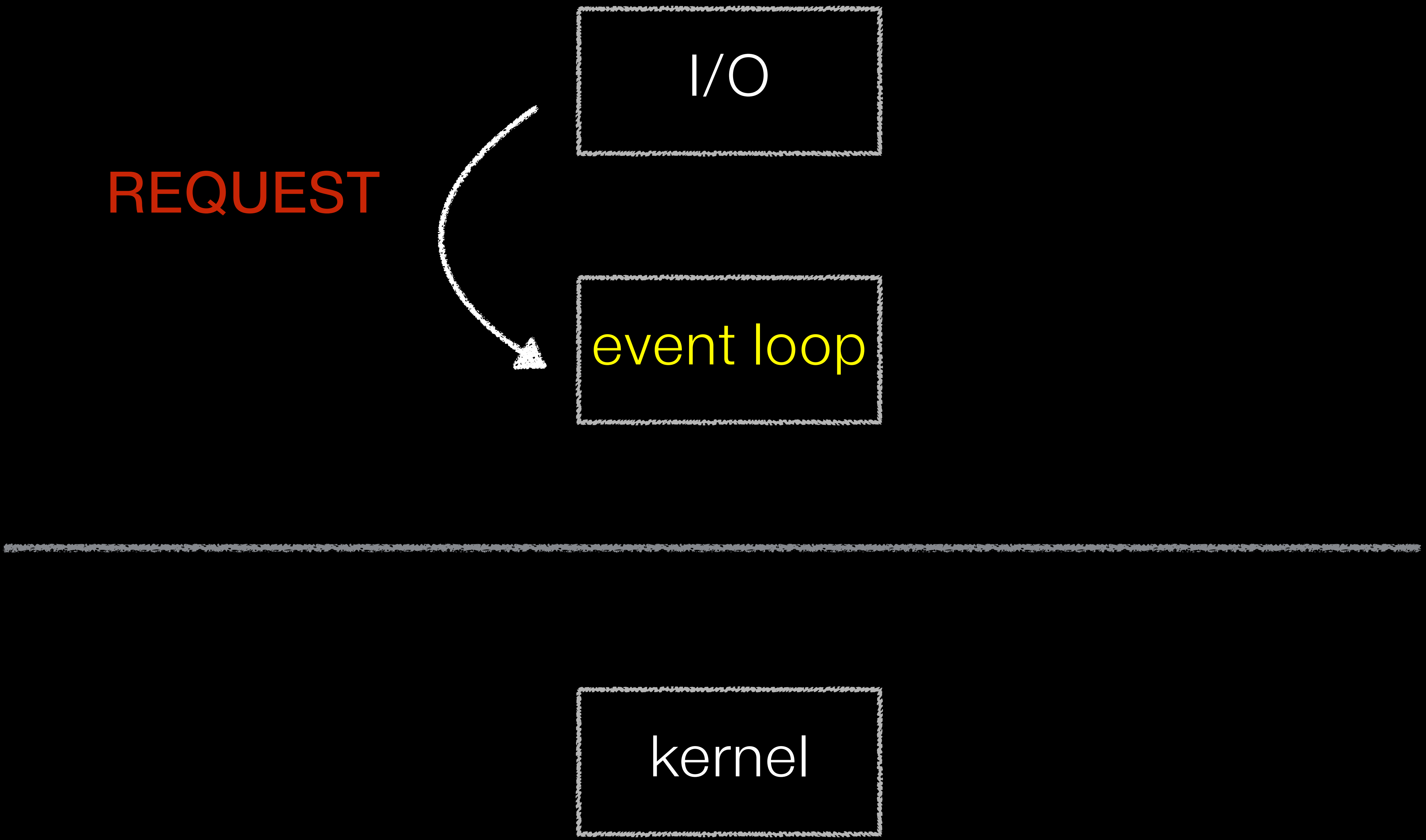

I/O
kernel

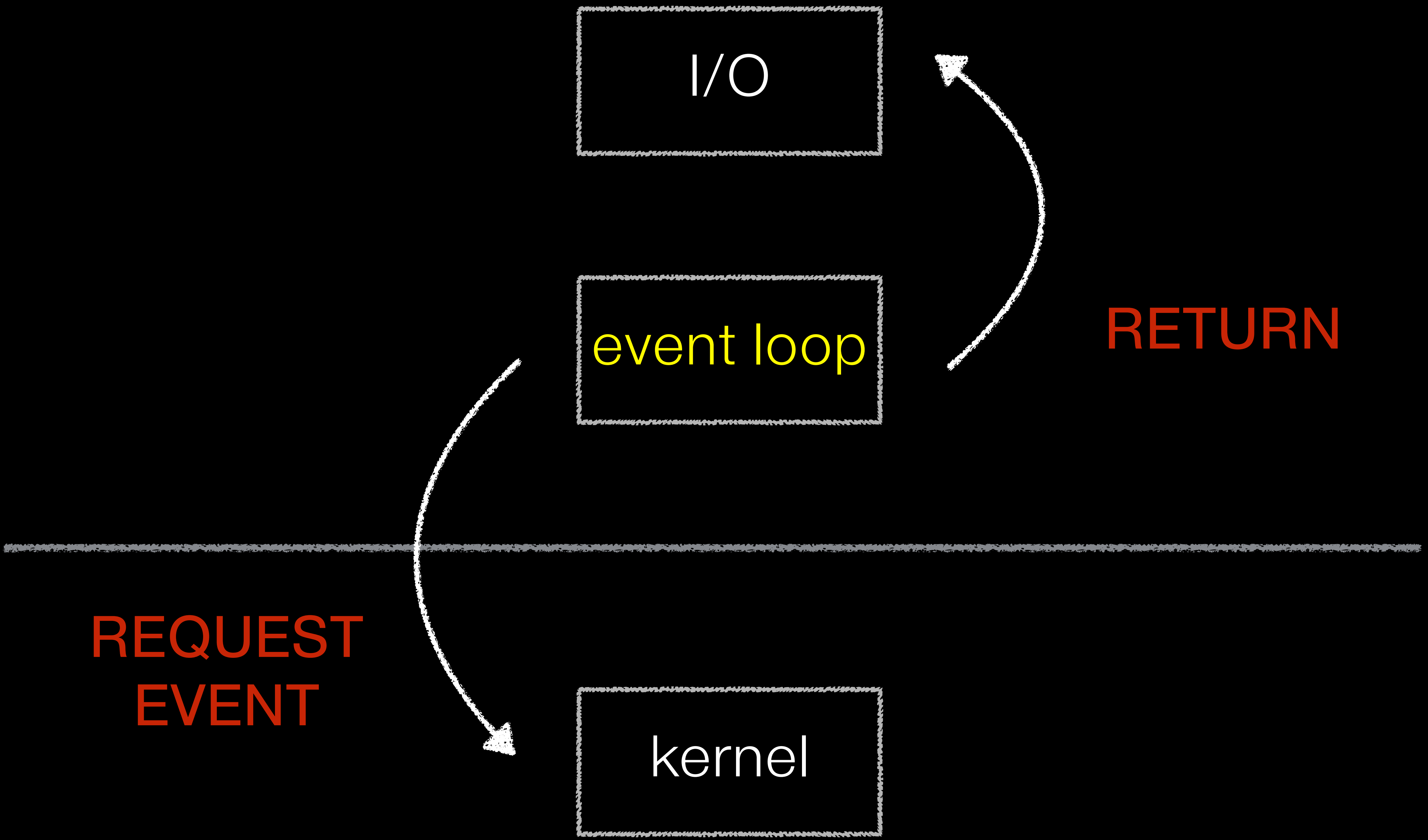

I/O
REQUEST
kernel

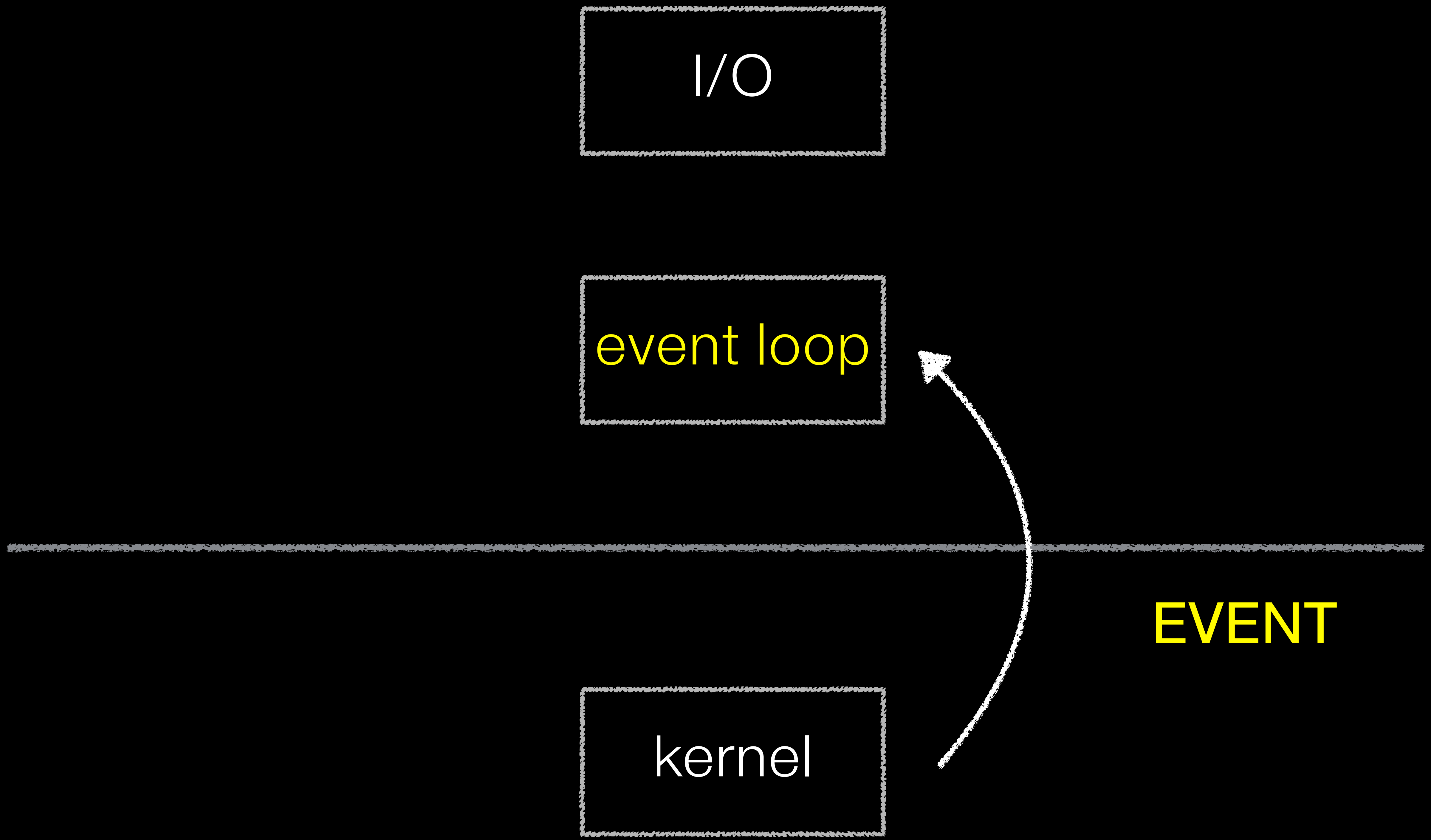

I/O
BLOCK
kernel

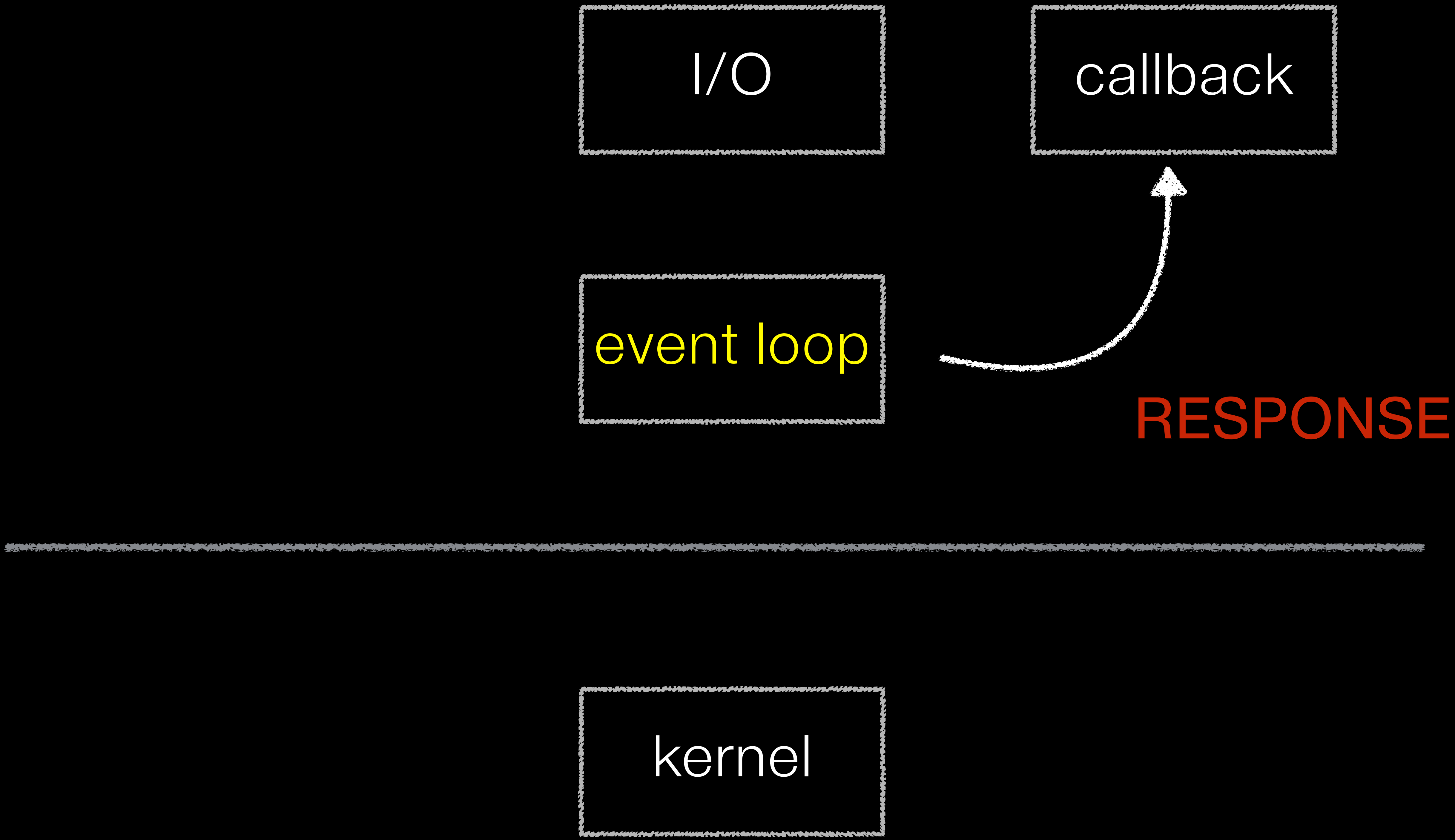

I/O
RESPONSE
kernel


I/O
event loop
kernel


I/O
REQUEST
event loop
kernel


I/O
event loop
RETURN
REQUEST EVENT
kernel


I/O
event loop
kernel
EVENT


I/O
callback
event loop
RESPONSE
kernel

Sync-like API
(NO CALLBACK!)

```
connect('pycon.kr', function(result, socket) {
    socket.read(function(data) {
        socket.write(data, function(result) {
            socket.close(function(result) {
            });
        });
    });
});
```

```
s.connect('pycon.kr')
data = s.read()
s.write(data)
s.close()
```

```
connect('pycon.kr', function(result, socket) {
    socket.read(function(data) {
        socket.write(data, function(result) {
            socket.close(function(result) {
            });
        });
    });
});
```

```
s.connect('pycon.kr')
data = s.read()
s.write(data)
s.close()
```

callbackhell.com

```
def recv(self, *args):
  while True:
    try:
      return sock.recv(*args)
    except error as ex:
      if ex.args[0] != EWOULDBLOCK:
        raise
      self._wait(self._read_event)
```

```
def recv(self, *args):
  while True:
    try:
      return sock.recv(*args)
    except error as ex:
      if ex.args[0] != EWOULDBLOCK:
        raise
      self._wait(self._read_event)
```

```
def recv(self, *args):
  while True:
    try:
      return sock.recv(*args)
    except error as ex:
      if ex.args[0] != EWOULDBLOCK:
        raise
      self._wait(self._read_event)
```

monkey patch

호환성 고려
(C Extension?)

Recap

- CPython은 사실상 Single thread
- I/O가 많다면 Gevent
- Gevent도 여전히 Single thread
- 외부 라이브러리등과의 호환성 문제

감사합니다.

kkung@beatpacking.com

WE
ARE
HIRING