gil那些事儿

峰云就她了

http://xiaorui.cc

相关

- 什么是python解释器
  - CPython vs pypy vs Jython

gil

- 什么是 Global Interpreter Lock
- Python为什么会有gil
- gil的优缺点
- 关于gil的历史

1. get the current length
2. check if there's room for more
3. Append element
4. Incrment the length by 1

Python's 线程

- python线程是系统线程的.
  - POSIX threads (pthreads)
  - Windows threads
  - 受内核来调度并切换上下文

性能对比

- thread only

```
def go_count(n):
    while n > 0:
        n -= 1
COUNT = 100000000 # 100 million
go_count(COUNT)
```

cost 10s

- thread two

```
t1 = Thread(target=go_count,args=(COUNT//2,))
t2 = Thread(target=go_count,args=(COUNT//2,))
t1.start(); t2.start()
t1.join(); t2.join()
```

cost 21s

那么问题来了

- 多线程为什么比单线程还慢？
  - python解释器运行原理
  - 只有一个线程在running
  - 多线程acquire lock的消耗成本

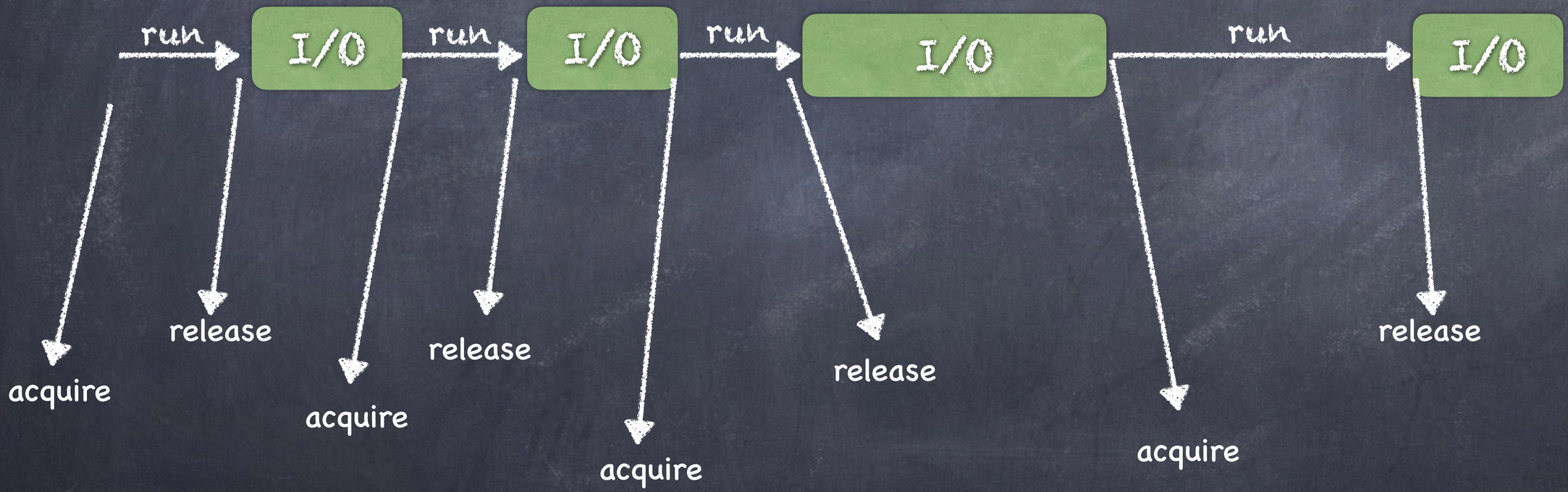
gil

- 简单描述:
  - 拿到gil锁，谁就可以running，当释放gil，send signal
  - 没拿到gil锁，休眠。内核调度到你，你如果没锁，sleep and wait for a signal
- 什么时候释放锁:
  - IO Block
  - every 100 tick force, default
- gil 是mutex + semaphore + condition

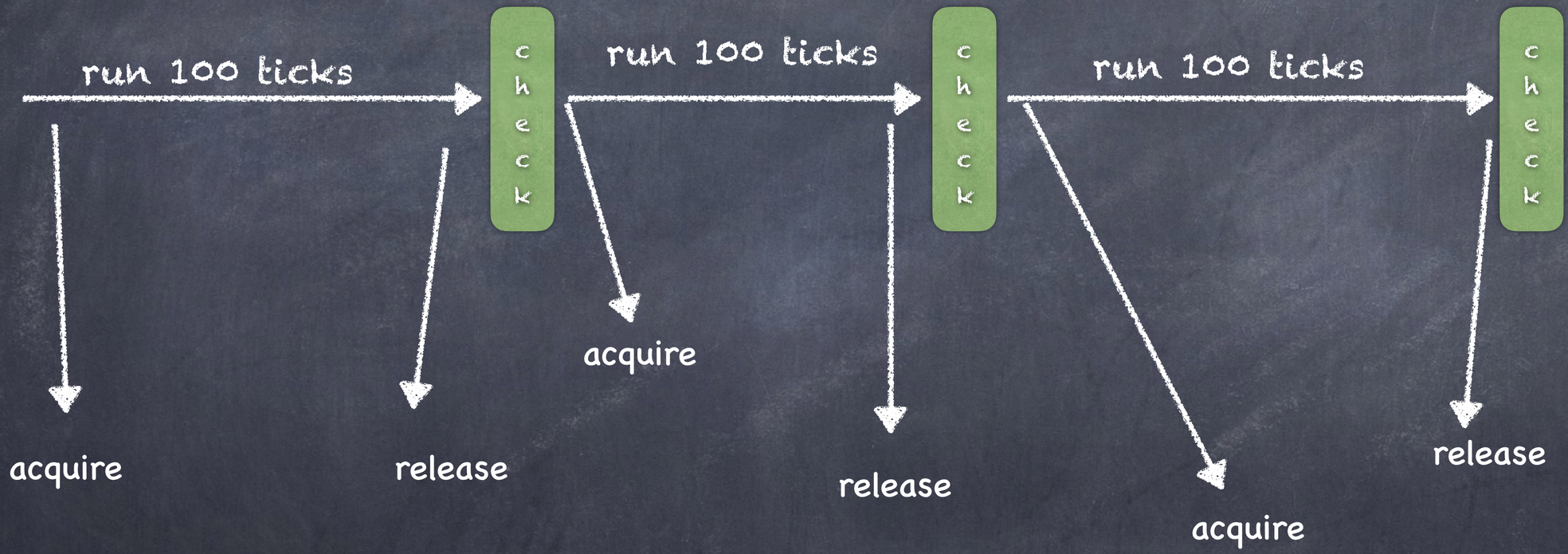
_mysql.c gil process

```
check_connection(self);

Py_BEGIN_ALLOW_THREADS

r = mysql_real_query(&(self->connection), query, len);

Py_END_ALLOW_THREADS
```

scheduler I/O



scheduler counter



Change it using sys.setcheckinterval()

what a check

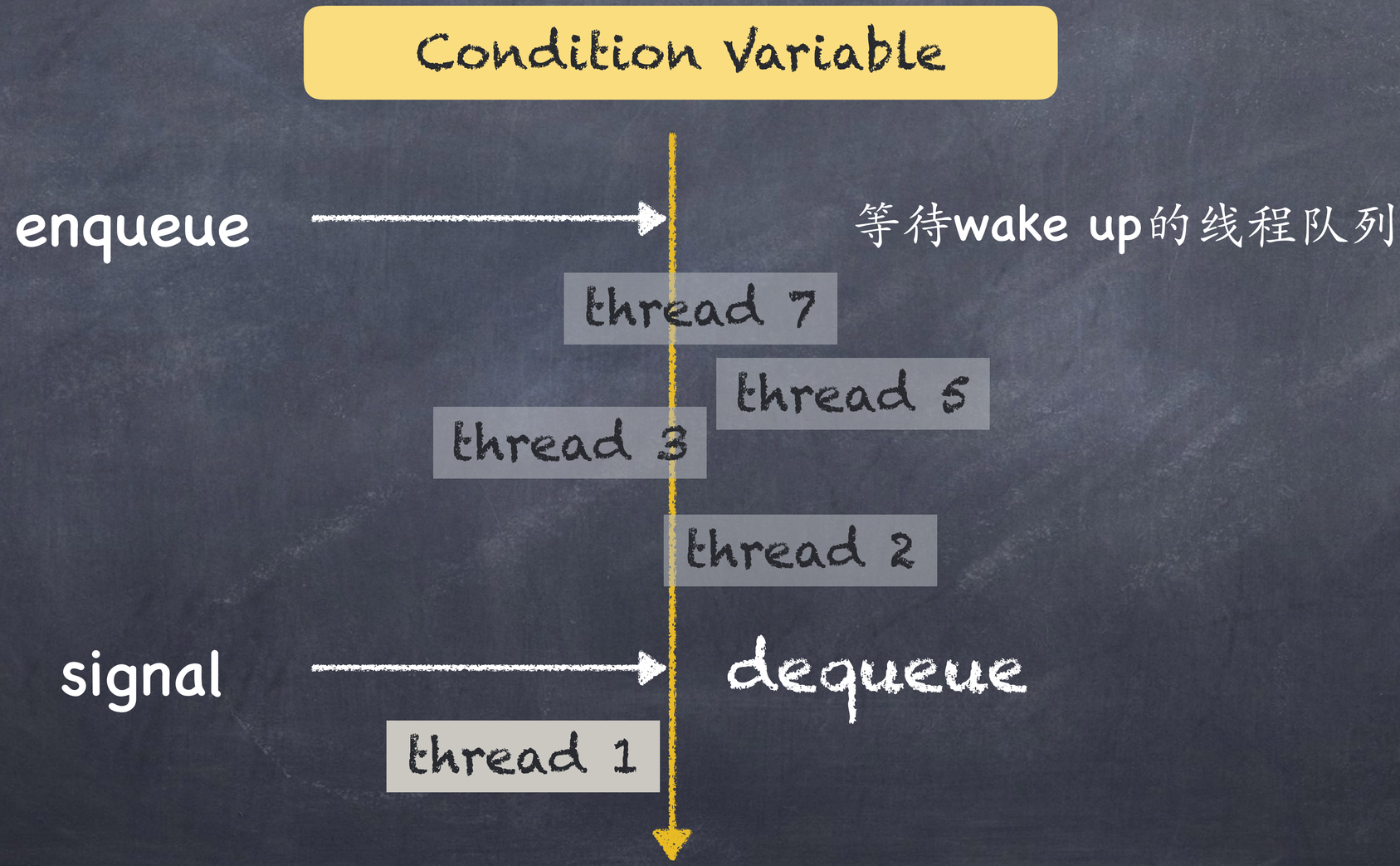
- Periodic "check" is simple
- The currently running thread ...
  - reset the tick counter
  - run signal handler if this is main thread
  - relase the gil
  - Reacquires the gil

A tick ?

```
 6           0 LOAD_GLOBAL              0 (True)
             3 STORE_FAST               0 (a)

 7           6 SETUP_LOOP              25 (to 34)
             9 LOAD_GLOBAL              1 (range)
            12 LOAD_CONST               1 (3)
            15 CALL_FUNCTION            1
            18 GET_ITER
       >>   19 FOR_ITER                11 (to 33)
            22 STORE_FAST               1 (i)

 8          25 LOAD_FAST                1 (i)
            28 PRINT_ITEM
            29 PRINT_NEWLINE
            30 JUMP_ABSOLUTE           19
       >>   33 POP_BLOCK

 9     >>   34 LOAD_FAST                0 (a)
            37 POP_JUMP_IF_FALSE       48

10          40 LOAD_CONST               2 ('xiaorui.cc')
            43 PRINT_ITEM
            44 PRINT_NEWLINE
            45 JUMP_FORWARD             0 (to 48)

11     >>   48 LOAD_CONST               0 (None)
            51 STORE_FAST               2 (b)

12          54 LOAD_CONST               3 (123)
            57 STORE_FAST               3 (c)
            60 LOAD_CONST               0 (None)
            63 RETURN_VALUE
```

```
def go():
    a = True
    for i in range(3):
        print i
    if a:
        print "xiaorui.cc"
    b = None
    c = 123
```

wake up


Condition Variable
enqueue
等待wake up的线程队列
thread 7
thread 5
thread 3
thread 2
signal
dequeue
thread 1

```
t2 100 5351 ENTRY
t2 100 5351 ACQUIRE
t2 100 5352 RELEASE
t2 100 5352 ENTRY
t2 100 5352 ACQUIRE
t2 100 5353 RELEASE
t1 100 5353 ACQUIRE
t2 100 5353 ENTRY
t2 38  5353 BUSY
t1 100 5354 RELEASE
t1 100 5354 ENTRY
t1 100 5354 ACQUIRE
t2 79  5354 RETRY
t1 100 5355 RELEASE
t1 100 5355 ENTRY
t1 100 5355 ACQUIRE
t2 73   5355 RETRY
t1 100 5356 RELEASE
t2 100 5356 ACQUIRE
t1 100 5356 ENTRY
t1 24  5356 BUSY
t2 100 5357 RELEASE
```

进入临界区

获取锁

释放锁

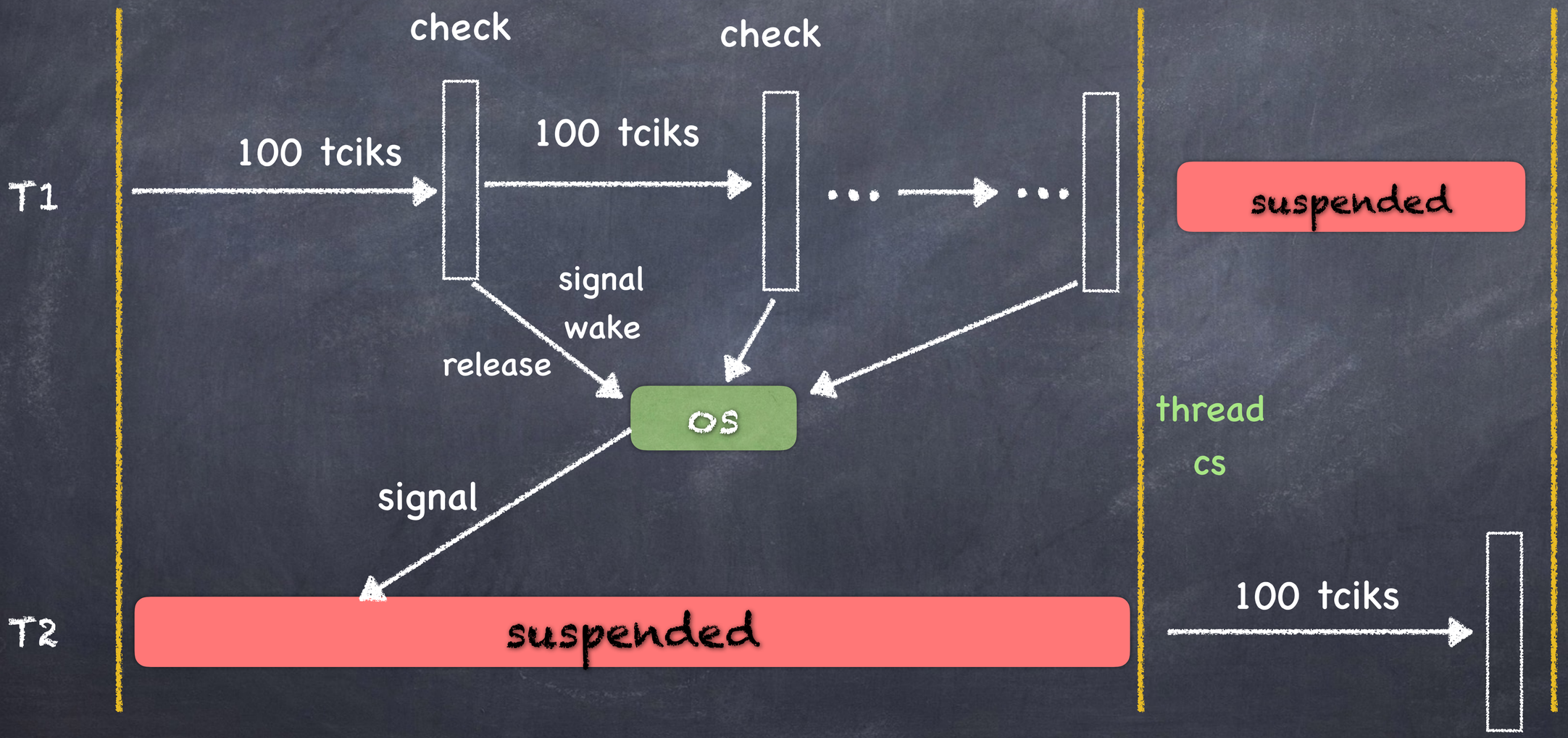
没拿到锁

重新尝试

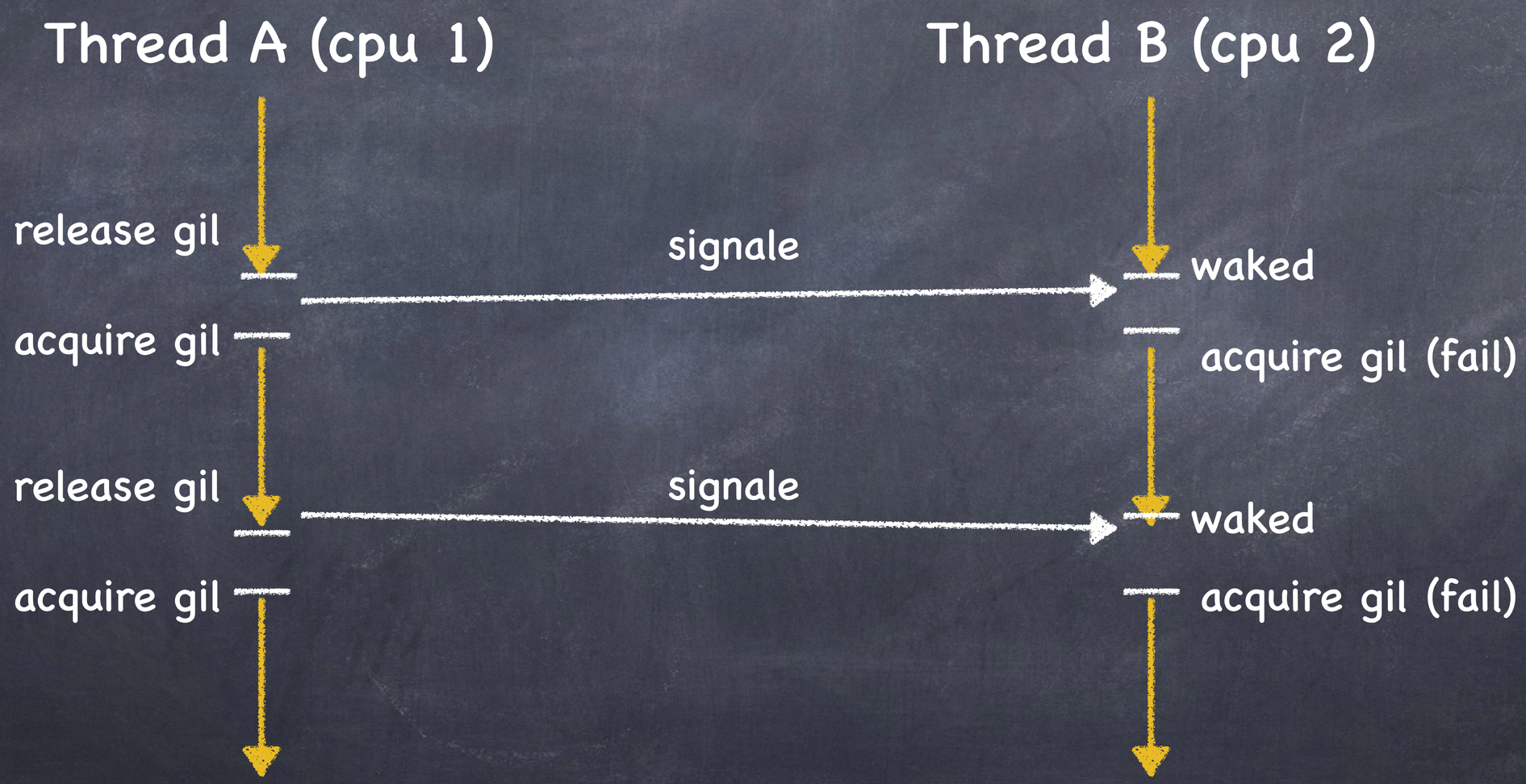
break to think

- python如何保证list\dict的操作原子性
- 某个线程一直高比率拿到锁？
- t1时间片用完了，kernel把t2调度起来，但gil还在t1手里？
- 多线程都在一个cpu core上？

不公平竞争

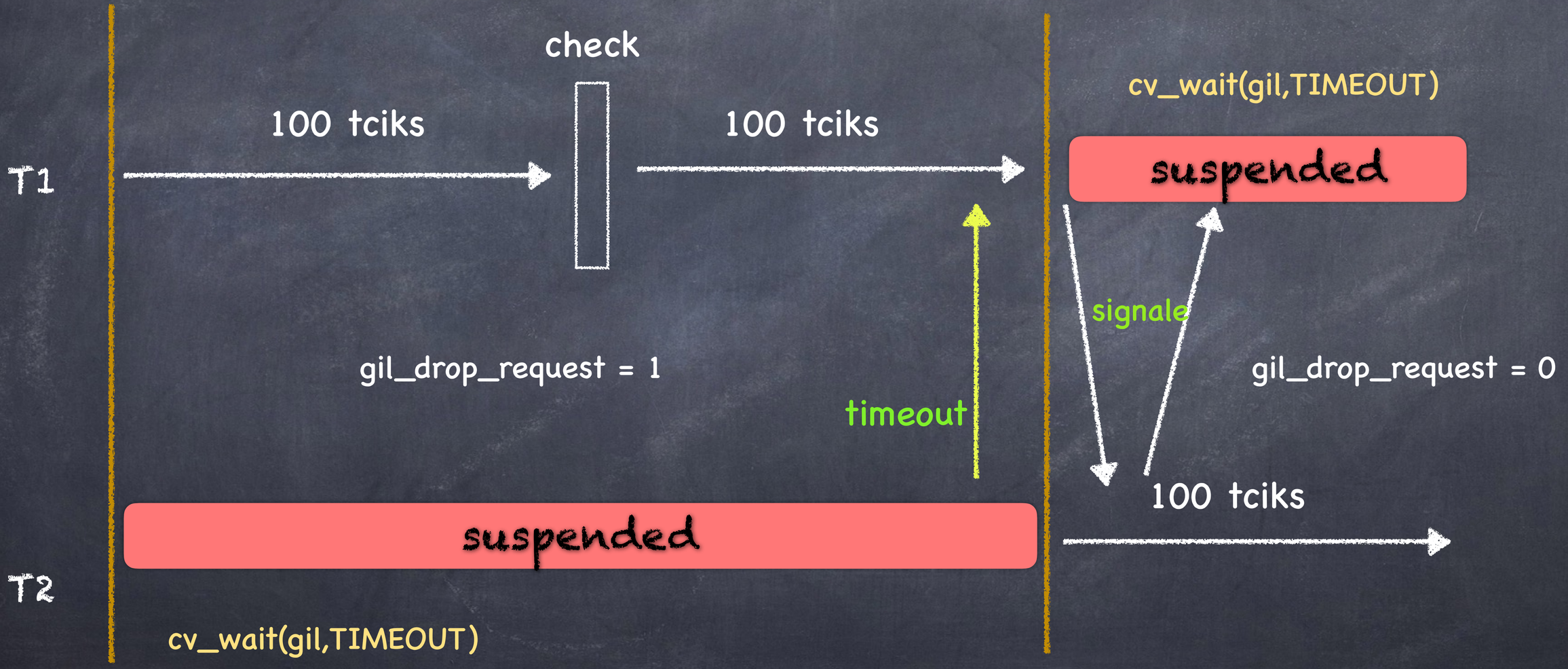

check
check
100 tciks
100 tciks
T1
signal
wake
release
os
suspended
thread
cs
signal
T2
suspended
100 tciks

不公平竞争


Thread A (cpu 1)
Thread B (cpu 2)
release gil
signale
waked
acquire gil
acquire gil (fail)
release gil
signale
waked
acquire gil
acquire gil (fail)

多核

new gil


check
100 tciks
100 tciks
T1
cv_wait(gil,TIMEOUT)
suspended
signale
gil_drop_request = 1
gil_drop_request = 0
timeout
100 tciks
suspended
T2
cv_wait(gil,TIMEOUT)

那么threading场景

- 由于有gil全局锁，python多线程的意义？
  - IO 密集
  - CPU 密集
- python的线程调度策略
  - 再次声明，解释器没有thread调度器

特例，Signal

- 当信号到达的时，解释器会按照每个tick都要check. 直到main thread处理了signal.

绕过gil

- multiprocessing
  - python gil 存在于线程之间
- ctypes
  - 调用c函数之前，会释放gil
- more … …

kill gil

- python能否去掉gil？
- 如何实现python底层去gil？
- 去掉之后又会出现什么？
- 总结：值不值，成本？

“Q & A”

-峰云就她了