iOS 十种线程锁

有毒的程序猿 关注
3 2017.11.11 19:46:03 字数 387 阅读 6,881

39赞
赞赏
多好文

前言

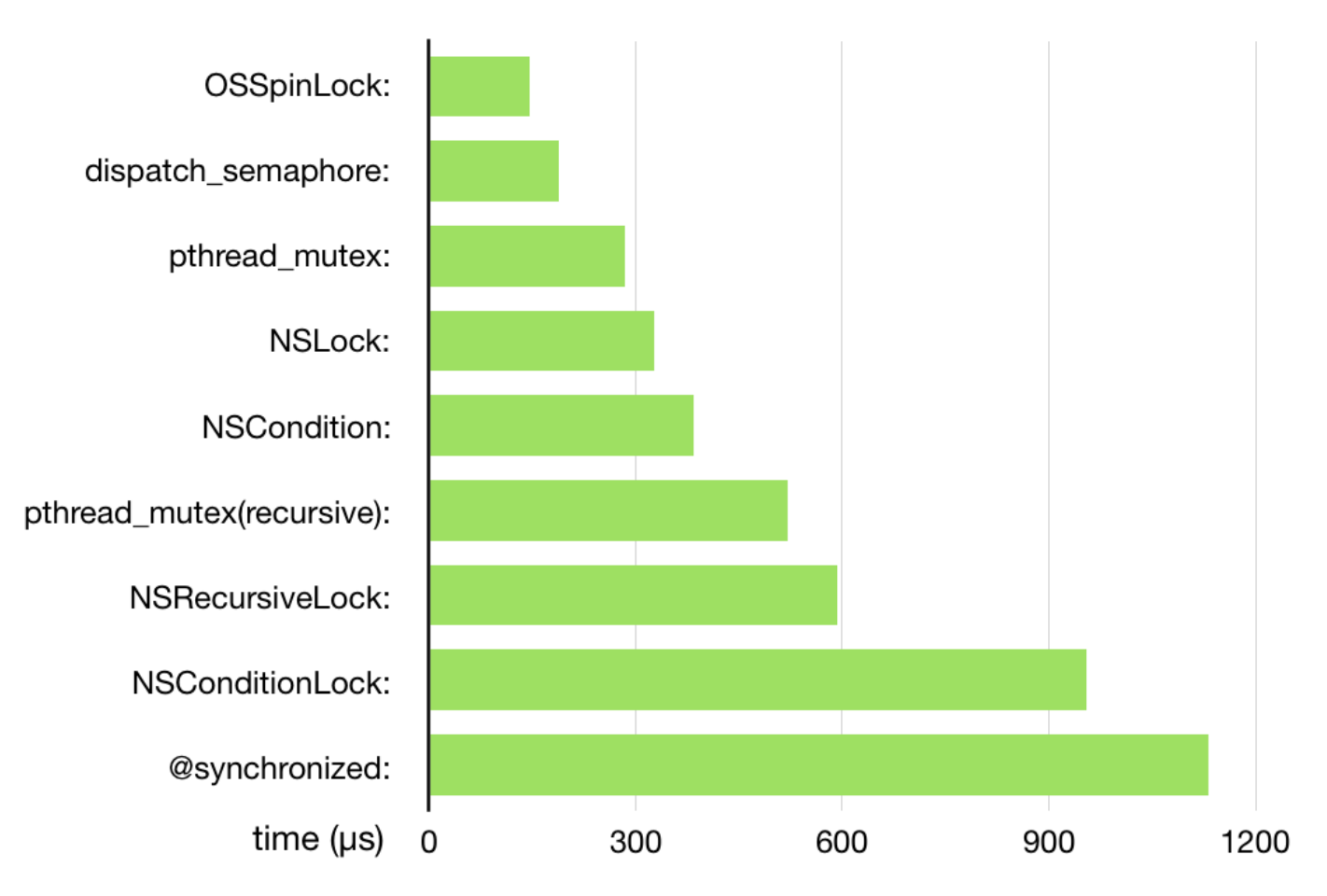
> 我觉得打游戏屏蔽脏话挺有必要的, 我走中单, 打野一直拿我蓝, 我开大打他路过交惩戒抢, 我气的不行, 就骂他, 结果打出来的都是"李白你是不是 * * * ?"、"抢 * * *的蓝", 我觉得没气势, 没办法, 我打了个"李白你个大坏蛋", 他就再也没抢过了, 还掩护我打蓝.

iOS 十种线程锁.jpeg

锁 是什么意思?

- `我们在使用多线程的时候多个线程可能会访问同一块资源，这样就很容易引发数据错乱和数据安全等问题，这时候就需要我们保证每次只有一个线程访问这一块资源，锁 应运而生。`
- `这里顺便提一下,上锁的两种方式trylock和lock使用场景:`

```
当前线程锁失败，也可以继续其它任务，用 trylock 合适
当前线程只有锁成功后，才会做一些有意义的工作，那就 lock，没必要轮询 trylock
注:以下大部分锁都会提供trylock接口,不再作解释
```



感谢yy大神.png

<准备操作>

```
测试代码
#define RHTICK   NSDate *startTime = [NSDate date];
#define RHTOCK   NSLog(@"==========Time: %f", -[startTime timeIntervalSinceNow]);
NSUInteger count = 1000*10000;//执行一千万次
RHTICK
for(int i=0; i<count; i++) {
加锁
解锁
}
RHTOCK
注:测试中执行时间会波动,所以我取的平均值.
```

一、OSSpinLock (自旋锁)

测试中效率最高的锁, 不过经 `YYKit` 作者确认, `OSSpinLock` 已经不再线程安全, `OSSpinLock` 有潜在的优先级反转问题.不再安全的 OSSpinLock;

- `0.097348s`

```
需要导入头文件
#import <libkern/OSAtomic.h>
// 初始化
 OSSpinLock spinLock = OS_SPINLOCK_INIT;
// 加锁
OSSpinLockLock(&spinLock);
// 解锁
OSSpinLockUnlock(&spinLock);
// 尝试加锁，可以加锁则立即加锁并返回 YES,反之返回 NO
OSSpinLockTry(&spinLock)
/*
注:苹果爸爸已经在iOS10.0以后废弃了这种锁机制,使用os_unfair_lock 替换,
顾名思义能够保证不同优先级的线程申请锁的时候不会发生优先级反转问题.
*/
```

二、os_unfair_lock(互斥锁)

- `0.171789s`

```
需要导入头文件
#import <os/lock.h>
// 初始化
 os_unfair_lock unfair_lock = OS_UNFAIR_LOCK_INIT;
// 加锁
os_unfair_lock_lock(&unfair_lock);
// 解锁
os_unfair_lock_unlock(&unfair_lock);
// 尝试加锁，可以加锁则立即加锁并返回 YES,反之返回 NO
os_unfair_lock_trylock(&unfair_lock);
/*
注:解决不同优先级的线程申请锁的时候不会发生优先级反转问题.
不过相对于 OSSpinLock , os_unfair_lock性能方面减弱了许多.
*/
```

三、dispatch_semaphore (信号量)

- `0.155043s`

```
// 初始化
dispatch_semaphore_t semaphore_t = dispatch_semaphore_create(1);
// 加锁
dispatch_semaphore_wait(semaphore_t,DISPATCH_TIME_FOREVER);
// 解锁
dispatch_semaphore_signal(semaphore_t);
/*
注: dispatch_semaphore  其他两个功能
1.还可以起到阻塞线程的作用.
2.可以实现定时器功能,这里不做过多介绍.
*/
```

四、pthread_mutex(互斥锁)

- `0.262592s`

```
需要导入头文件
#import <pthread/pthread.h>
// 初始化(两种)
1.普通初始化
pthread_mutex_t mutex_t;
pthread_mutex_init(&mutex_t, NULL);
2.宏初始化
pthread_mutex_t mutex =PTHREAD_MUTEX_INITIALIZER;
// 加锁
pthread_mutex_lock(&mutex_t);
// 解锁
pthread_mutex_unlock(&mutex_t);
// 尝试加锁，可以加锁时返回的是 0，否则返回一个错误
pthread_mutex_trylock(& mutex_t)
```

五、NSLock(互斥锁、对象锁)

- `0.283196s`

```
// 初始化
NSLock *_lock = [[NSLock alloc]init];
// 加锁
[_lock lock];
// 解锁
[_lock unlock];
// 尝试加锁，可以加锁则立即加锁并返回 YES,反之返回 NO
[_lock tryLock];
```

六、NSCondition(条件锁、对象锁)

- `0.293046s`

```
// 初始化
NSCondition *_condition= [[NSCondition alloc]init];
// 加锁
[_condition lock];
// 解锁
[_condition unlock];
/*
其他功能接口
wait 进入等待状态
waitUntilDate:让一个线程等待一定的时间
signal 唤醒一个等待的线程
broadcast 唤醒所有等待的线程
注: 所测时间波动太大, 有时候会快于 NSLock, 我取得中间值.
*/
```

七、NSConditionLock(条件锁、对象锁)

- `0.950285s`

```
// 初始化
NSConditionLock *_conditionLock = [[NSConditionLock alloc]init];
// 加锁
[_conditionLock lock];
// 解锁
[_conditionLock unlock];
// 尝试加锁，可以加锁则立即加锁并返回 YES,反之返回 NO
[_conditionLock tryLock];
/*
其他功能接口
- (instancetype)initWithCondition:(NSInteger)condition NS_DESIGNATED_INITIALIZER; //初
- (void)lockWhenCondition:(NSInteger)condition;//条件成立触发锁
- (BOOL)tryLockWhenCondition:(NSInteger)condition;//尝试条件成立触发锁
- (void)unlockWithCondition:(NSInteger)condition;//条件成立解锁
- (BOOL)lockBeforeDate:(NSDate *)limit;//触发锁 在等待时间之内
- (BOOL)lockWhenCondition:(NSInteger)condition beforeDate:(NSDate *)limit;//触发锁 条件
*/
```

八、NSRecursiveLock(递归锁、对象锁)

- `0.473536s`

```
// 初始化
NSRecursiveLock *_recursiveLock = [[NSRecursiveLock alloc]init];
// 加锁
[_recursiveLock lock];
// 解锁
[_recursiveLock unlock];
// 尝试加锁，可以加锁则立即加锁并返回 YES,反之返回 NO
[_recursiveLock tryLock];
/*
注: 递归锁可以被同一线程多次请求，而不会引起死锁。
即在同一线程中在未解锁之前还可以上锁, 执行锁中的代码。
这主要是用在循环或递归操作中。
- (BOOL)lockBeforeDate:(NSDate *)limit;//触发锁 在等待时间之内
*/
```

九、@synchronized()递归锁

- `1.101924s`

```
// 初始化
@synchronized(锁对象){
}
底层封装的pthread_mutex的PTHREAD_MUTEX_RECURSIVE 模式,
锁对象来表示是否为同一把锁

```

更多关于@synchronized;

十、pthread_mutex(recursive)(递归锁)

- `0.372398s`

```
// 初始化
pthread_mutex_t mutex_t;
pthread_mutexattr_t attr;
pthread_mutexattr_init(&attr); //初始化attr并且给它赋予默认pthread_mutexattr_settype(&attr
pthread_mutex_init(&mutex_t, &attr);
pthread_mutexattr_destroy(&attr); //销毁一个属性对象, 在重新进行初始化之前该结构不能重新使用
// 加锁
pthread_mutex_lock(&mutex_t);
// 解锁
pthread_mutex_unlock(&mutex_t);
/*
注: 递归锁可以被同一线程多次请求，而不会引起死锁。
即在同一线程中在未解锁之前还可以上锁, 执行锁中的代码。
这主要是用在循环或递归操作中。
*/
```

性能总结

```
OSSpinLock                          0.097348s
dispatch_semaphore                  0.155043s
os_unfair_lock                      0.171789s
pthread_mutex                       0.262592s
NSLock                              0.283196s
pthread_mutex(recursive)            0.372398s
NSRecursiveLock                     0.473536s
NSConditionLock                     0.950285s
@synchronized                       1.101924s
注:建议正常锁功能用 pthread_mutex ,os_unfair_lock (适配低版本)
```

锁的注解

1、自旋锁

```
OSSpinLock 就是典型的自旋锁
自旋锁的特点是在没有获取到锁时既锁已经被添加,还没有被解开时.
OSSpinLock处于忙等状态,一直占用CPU资源,类似如下伪代码:
while(锁没解开);

关于优先级反转问题
由于线程调度,每条线程的分配时间权重不一样,当权重小的线程先进入OSSpinLock优先加锁,
当权重大的线程再来访问,就阻塞在这,可能权重大的线程会一直分配到cpu所以一直会进来,
但是因为有锁,只能等待,权重小的线程得不到cpu资源分配,所以不会解锁,造成一定程度的死锁.

```

2、互斥锁

```
os_unfair_lock 、pthread_mutex是典型的互斥锁,在没有获取到锁时既锁已经被添加,还没有被解开时.
它们都会让当前线程进入休眠状态既不占用CPU资源,但是为什么,互斥锁比自旋锁的效率低呢,
是因为休眠,以及唤醒休眠,比忙等更加消耗CPU资源.

NSLock 封装的pthread_mutex的PTHREAD_MUTEX_NORMAL 模式
NSRecursiveLock 封装的pthread_mutex的PTHREAD_MUTEX_RECURSIVE 模式
```

3、条件锁

```
在一定条件下,让其等待休眠,并放开锁,等接收到信号或者广播,会从新唤起线程,并重新加锁.
pthread_cond_wait(&_cond, &_mutex);
 // 信号
    pthread_cond_signal(&_cond);
 // 广播
    pthread_cond_broadcast(&_cond);

像NSCondition封装了pthread_mutex的以上几个函数

NSConditionLock封装了NSCondition

```

4、递归锁

```
递归锁的主要意思是,同一条线程可以加多把锁.什么意思呢,就是相同的线程访问一段代码,
如果是加锁的可以继续加锁,继续往下走,不同线程来访问这段代码时,发现有锁要等待所有锁解开之后才可以继续往下

NSRecursiveLock 封装的pthread_mutex  的PTHREAD_MUTEX_RECURSIVE模式
```

39人点赞 > IOS编程 ···

有毒的程序猿 有心天下，却输一笔多情，我的山水落在你的眉间，你，肯入... 关注
总资产216 共写了3.4W字 获得154个赞 共61个粉丝

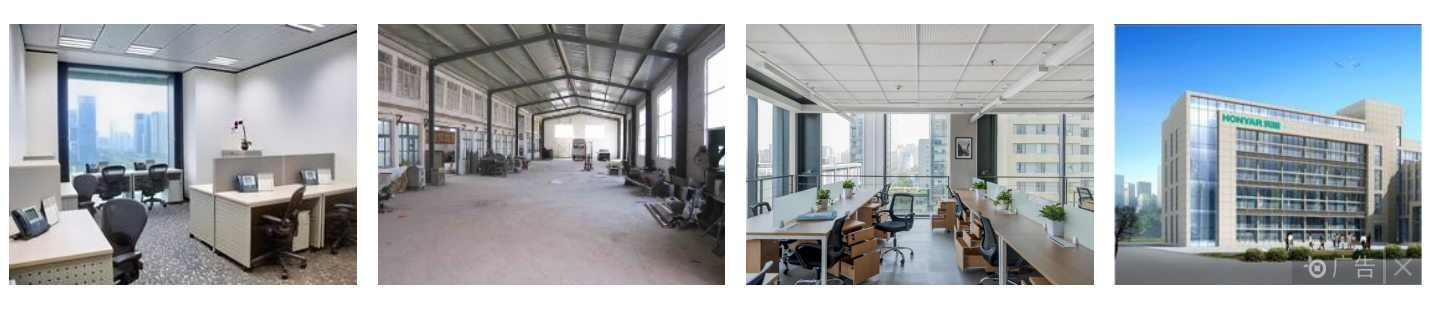
被以下专题收入，发现更多相似内容

+收入我的专题 ios常用功能 lock 多线程 ios-多线程 系统层知识 多线程 iOS开发 展开更多

推荐阅读 更多精彩内容 >

[iOS] 谈谈iOS多线程的锁
前言 iOS开发中由于各种第三方库的高度封装，对锁的使用很少，刚好之前面试中被问到的关于并发编程锁的问题，都是一知...
喵渣渣 阅读 2,855 评论 0 赞 32

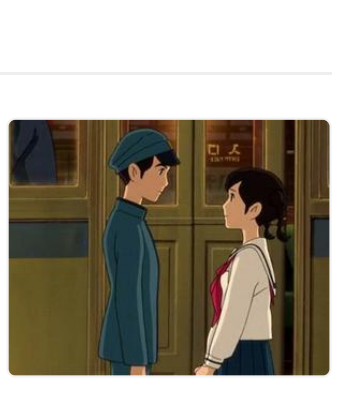
iOS - 关于线程同步
引用自多线程编程指南应用程序里面多个线程的存在引发了多个执行线程安全访问资源的潜在问题。两个线程同时修改同一资源有...
Mitchell 阅读 1,398 评论 1 赞 7



2017-05-04
今天北京沙尘＋雾霾，好久没有这种天气，还以为污染已远离我们，错觉了。在外游玩的群友多吸两口新鲜空气吧。 天气不好，...
王悦yue 阅读 62 评论 0 赞 4

流行性"心学病"
《知行合一王阳明，1472-1529》 三颗星 第一次看《王阳明知行合一》是因为身边的创业者渐已不谈佛家道家开始热...
Monstercarrie 阅读 118 评论 1 赞 2

嘿！老同学，听说你要结婚啦
上个星期收到老同学奇子的结婚邀请函，我吃了一惊，奇子是我大学同学，一个宿舍住了好几年。惊闻他下个月就要结婚了，我不...
Super安浅浅 阅读 661 评论 26 赞 21



推荐阅读
ReentrantLock源码分析
阅读 314
Java锁
阅读 228
Java锁全总结
阅读 146
漫画：从JVM锁扯到Redis分布式锁
阅读 122
Java面试题系列（四）——并发
阅读 599