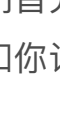


18 | 怎么减少 App 电量消耗?

戴铭 2019-04-20





00:00

讲述：冯永奇 大小：7.54M

08:13

你好，我是戴铭。

手机设备电量有限，App 开发时如不注意电量的的消耗，当用户发现你的 App 是耗电大户时，就会毫不犹豫地将其抛弃。所以，每次开发完，我们都需要去检查自己的 App 有没有耗电的问题。

耗电的原因有千万种，如果每次遇到耗电过多的问题，我们都从头查找一番的话，那必然会效率低下。

就比如说，测试同学过来跟你说“某个页面的前一个版本还挺好的，这个版本的耗电怎么多了那么多”，那么你首先想到可能就是这个页面有没有开启定位，网络请求是不是频繁，亦或是定时任务时间是不是间隔过小。接下来，你会去查找耗电问题到底是怎么引起的。你去翻代码的时候却发现，这个页面的相关功能在好几个版本中都没改过了。

那么，到底是什么原因使得这一个版本的耗电量突然增加呢？不如就使用排除法吧，你把功能一个个都注释掉，却发现耗电量还是没有减少。这时，你应该怎么办呢？接下来，我就在今天的文章里面和你详细分享一下这个问题的解法吧。

我们首先需要明确的是，只有获取到电量，才能够发现电量问题。所以，我就先从如何获取电量和你讲起。

如何获取电量？

在 iOS 中，IOKit framework 是专门用于跟硬件或内核服务通信的。所以，我们可以通过 IOKit framework 来获取硬件信息，进而获取到电量消耗信息。在使用 IOKit framework 时，你需要：

- 首先，把 IOPowerSources.h、IOPSKeys.h 和 IOKit 这三个文件导入到工程中；
- 然后，把 batteryMonitoringEnabled 置为 true；
- 最后，通过如下代码获取 1% 精度的电量信息。

复制代码

```
1 #import "IOPSKeys.h"
2 #import "IOPowerSources.h"
3
4 -(double) getBatteryLevel{
5     // 返回电量信息
6     CFTypeRef blob = IOPSCopyPowerSourcesInfo();
7     // 返回电量句柄列表数据
8     CFArrayRef sources = IOPSCopyPowerSourcesList(blob);
9     CFDictionaryRef pSource = NULL;
10    const void *psValue;
11    // 返回数组大小
12    int numOfSources = CFArrayGetCount(sources);
13    // 计算大小出错处理
14    if (!pSource) {
15        NSLog(@"Error in CFArrayGetCount");
16        return -1.0f;
17    }
18
19    // 计算所剩电量
20    for (int i=0; i<numOfSources; i++) {
21        // 返回电源可读信息的字典
22        pSource = IOPSCopyPowerSourceDescription(blob, CFArrayGetValueAtIndex(sources, i));
23        if (!pSource) {
24            NSLog(@"Error in IOPSCopyPowerSourceDescription");
25            return -1.0f;
26        }
27        psValue = (CFStringRef) CFDictionaryGetValue(pSource, CFSTR(kIOPSPNameKey));
28
29        int curCapacity = 0;
30        int maxCapacity = 0;
31        double percentage;
32
33        psValue = CFDictionaryGetValue(pSource, CFSTR(kIOPSPCurrentCapacityKey));
34        CFNumberGetValue((CFNumberRef)psValue, kCFNumberSInt32Type, &curCapacity);
35
36        psValue = CFDictionaryGetValue(pSource, CFSTR(kIOPSPMaxCapacityKey));
37        CFNumberGetValue((CFNumberRef)psValue, kCFNumberSInt32Type, &maxCapacity);
38
39        percentage = ((double) curCapacity / (double) maxCapacity * 100.0f);
40        NSLog(@"curCapacity: %d / maxCapacity: %d , percentage: %.1f ", curCapacity, maxCapacity, percentage);
41        return percentage;
42    }
43    return -1.0f;
44 }
```

说完耗电量的获取方法，我们再继续看如何解决电量问题。

如何诊断电量问题？

回到最开始的问题，当你用排除法将所有功能注释掉后，如果还有问题，那么这个耗电一定是由其他线程引起的。创建这个耗电线程的地方可能是在其他地方，比如是由第三方库引起，或者是公司其他团队开发的二方库。

所以，你需要逆向地去思考这个问题。这里，你不妨回顾一下，我们在第 12 篇文章“[iOS 崩溃千奇百怪，如何全面监控](#)”中是怎么定位问题的。

也就是说，我们还是先反过来看看出现电量问题的期间，哪个线程是有问题的。通过下面的这段代码，你就可以获取到所有线程的信息：

复制代码

```
1 thread_act_array_t threads;
2 mach_msg_type_number_t threadCount = 0;
3 const task_t thisTask = mach_task_self();
4 kern_return_t kr = task_threads(thisTask, &threads, &threadCount);
5
```

从上面代码可以看出，通过 task_threads 函数，我们就能够得到所有的线程信息数组 threads，以及线程总数 threadCount。threads 数组里的线程信息结构体 thread_basic_info 里有一个记录 CPU 使用百分比的字段 cpu_usage。thread_basic_info 结构体的代码如下：

复制代码

```
1 struct thread_basic_info {
2     time_value_t      user_time;      /* user 运行的时间 */
3     time_value_t      system_time;    /* system 运行的时间 */
4     integer_t          cpu_usage;      /* CPU 使用百分比 */
5     policy_t           policy;        /* 有效的计划策略 */
6     integer_t          run_state;      /* run state (see below) */
7     integer_t          flags;          /* various flags (see below) */
8     integer_t          suspend_count; /* suspend count for thread */
9     integer_t          sleep_time;    /* 休眠时间 */
10 };
11
```

有了这个 cpu_usage 字段，你就可以通过遍历所有线程，去查看是哪个线程的 CPU 使用百分比过高了。如果某个线程的 CPU 使用率长时间都比较高的话，比如超过了 90%，就能够推断出它是有问题的。这时，将其方法堆栈记录下来，你就可以知道到底是哪段代码让你 App 的电量消耗多了。

通过这种方法，你就可以快速定位到问题，有针对性地进行代码优化。多线程 CPU 使用率检查的完整代码如下：

复制代码

```
1 // 轮询检查多个线程 CPU 情况
2 + (void)updateCPU {
3     thread_act_array_t threads;
4     mach_msg_type_number_t threadCount = 0;
5     const task_t thisTask = mach_task_self();
6     kern_return_t kr = task_threads(thisTask, &threads, &threadCount);
7     if (kr != KERN_SUCCESS) {
8         return;
9     }
10    for (int i = 0; i < threadCount; i++) {
11        thread_info_data_t threadInfo;
12        thread_basic_info_t threadBaseInfo;
13        mach_msg_type_number_t threadBaseInfo = THREAD_INFO_MAX;
14        if (thread_info((thread_act_t)threads[i], THREAD_BASIC_INFO, (thread_info_data_t)&threadBaseInfo = (thread_base_info_t)threadInfo;
15        if (!!(threadBaseInfo->flags & TH_FLAGS_IDLE)) {
16            integer_t cpuUsage = threadBaseInfo->cpu_usage / 10;
17            if (cpuUsage > 90) {
18                //cup 消耗大于 90 时打印和记录堆栈
19                NSString *reStr = smStackOfThread(threads[i]);
20                // 记录数据库中
21                [[[[SMLogDB sharedInstance] increaseWithStackString:reStr] sut
22                NSLog(@"CPU usage overload thread stack: %@",reStr);
23            }
24        }
25    }
26 }
27 }
28 }
29
```

优化电量

现在我们已经知道了在线上碰到电量问题时，应该如何解决，但是电量的不合理消耗也可能来自其他方面。CPU 是耗电的大头，引起 CPU 耗电的单个问题可以通过监控来解决，但点滴汇聚终成大海，每一个不合理的小的电量消耗，最终都可能会造成大的电量浪费。所以，我们在平时的开发工作中，时刻关注对耗电量的优化也非常重要。

对 CPU 的使用要精打细算，要避免让 CPU 做多余的事情。对于大量数据的复杂计算，应该把数据传到服务器去处理，如果必须要在 App 内处理复杂数据计算，可以通过 GCD 的 dispatch_block_create_with_qos_class 方法指定队列的 Qos 为 QOS_CLASS_UTILITY，将计算工作放到这个队列的 block 里。在 QOS_CLASS_UTILITY 这种 Qos 模式下，系统针对大量数据的计算，以及复杂数据处理专门做了电量优化。

接下来，我们再看看除了 CPU 会影响耗电，对电量影响较大的因素还有哪些呢？

除了 CPU，I/O 操作也是耗电大户。任何的 I/O 操作，都会破坏掉低功耗状态。那么，针对 I/O 操作要怎么优化呢？

业内的普遍做法是，将碎片化的数据磁盘存储操作延后，先在内存中聚合，然后再进行磁盘存储。碎片化的数据进行聚合，在内存中进行存储的机制，可以使用系统自带的 NSCache 来完成。

NSCache 是线程安全的，NSCache 会在到达预设缓存空间值时清理缓存，这时会触发 cache:willEvictObject: 方法的回调，在这个回调里就可以对数据进行 I/O 操作，达到将聚合的数据 I/O 延后的目的。I/O 操作的次数减少了，对电量的消耗也就减少了。

SDWebImage 图片加载框架，在图片的读取缓存处理时没有直接使用 I/O，而是使用了 NSCache。使用 NSCache 的相关代码如下：

复制代码

```
1 - (UIImage *)imageFromMemoryCacheForKey:(NSString *)key {
2     return [self.memCache objectForKey:key];
3 }
4
5 - (UIImage *)imageFromDiskCacheForKey:(NSString *)key {
6     // 检查 NSCache 里是否有
7     UIImage *image = [self imageFromMemoryCacheForKey:key];
8     if (image) {
9         return image;
10    }
11    // 从磁盘读取
12    UIImage *diskImage = [self diskImageForKey:key];
13    if (diskImage && self.shouldCacheImagesInMemory) {
14        NSUInteger cost = SDCacheCostForImage(diskImage);
15        [self.memCache setObject:diskImage forKey:key cost:cost];
16    }
17    return diskImage;
18 }
19
```

可以看出，SDWebImage 将获取的图片数据都放到了 NSCache 里，利用 NSCache 缓存策略进行图片缓存内存的管理。每次读取图片时，会检查 NSCache 是否已经存在图片数据：如果有，就直接从 NSCache 里读取；如果没有，才会通过 I/O 读取磁盘缓存图片。

使用了 NSCache 内存缓存能够有效减少 I/O 操作，你在写类似功能时也可以采用这样的思路，让你的 App 更省电。

CPU 和 I/O 这两大耗电问题都解决后，还有什么要注意的呢？ 这里还有两份关于 App 电量消耗的资料，你可以对照你的 App 来查看。

苹果公司专门维护了一个电量优化指南“[Energy Efficiency Guide for iOS Apps](#)”，分别从 CPU、设备唤醒、网络、图形、动画、视频、定位、加速度计、陀螺仪、磁力计、蓝牙等多方面因素提出了电量优化方面的建议。所以，当使用了苹果公司的电量优化指南里提到的功能时，严格按照指南里的最佳实践去做就能够保证这些功能不会引起不合理的电量消耗。

同时，苹果公司在 2017 年 WWDC 的 Session 238 也分享了一个关于如何编写节能 App 的主题“[Writing Energy Efficient Apps](#)”。

小结


今天我跟你分享了如何通过获取线程信息里的 cpu_usage 字段来判断耗电线程，进而得到当前线程执行方法堆栈，从而精准、快速地定位到引起耗电的具体方法。我曾经用这个方法解决了几起难以定位的耗电问题，这些问题都出在二方库上。通过获取到的方法堆栈，我就有了充足的证据去推动其他团队进行电量优化。

除此之外，我还跟你介绍了如何在平时开发中关注电量的问题。在我看来，减少 App 耗电也是开发者的天职，不然如何向我们可爱的用户交代呢。

课后小作业

请你使用我今天分享的耗电检查方法，检查一下你的 App，看看哪个方法最耗电。


感谢你的收听，欢迎你在评论区给我留言分享你的观点，也欢迎把它分享给更多的朋友一起阅读。




iOS 开发高手课

从原理到实战，带你解决 80% 的开发难题

戴铭
前滴滴应用技术专家

新版升级：点击  请朋友读，20位好友免费读，邀请订阅更有 **现金奖励**。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。



吴开



由作者筛选后的优质留言将会公开显示，欢迎踊跃留言。

Command + Enter 发表 0/2000字 提交留言

精选留言(14)


泽七

获取电量为为什么不用 [[UIDevice currentDevice] batteryLevel] 呢？

2019-04-20   27



包罗万象

iokit已经无法导入了，请问老师这种情况你是怎么处理的？

2019-05-10   5



cp_kong

IOKit Frameworks 目前在iOS项目中无法导入了，要自己新建一个mac项目，然后从那个项目导入，再拷到iOS项目中

2019-05-15  1  3



Geek_f0b40f

IOKit Frameworks 是Mac 的框架？参考的这个吗？ <https://blog.csdn.net/uxyheaven/article/details/38167509>

2019-04-23   1

Alexander



获取电量的for循环直接return了。

2019-06-12  

月落泉

smStackOfThread这个方法哪有啊



作者回复: <https://github.com/ming1016/GCDFetchFeed>
GCDFetchFeed/GCDFetchFeed/GCDFetchFeed/Library/SMLogMonitor/SMCallStack.m

2019-05-10  

do.利军

Stack of thread: 771:
CPU used: 34.0 percent
user time: 542025 second
libsystem_kernel.dylib 0x111e6617a mach_msg_trap + 10
libsystem_kernel.dylib 0x111e696a2 thread_get_state + 421
Stock 0x102ef07e smStackOfThread + 878
Stock 0x102f0fea8 +[SMCPUMonitor updateCPU] + 296
Stock 0x1038d097f -[AppDelegate updateCPUInfo] + 47
Foundation 0x10bd2a135 __NSFireTimer + 83
CoreFoundation 0x10a8613e4 __CFRunLoopIs_CALLING_OUT_TO_A_TIMER_CALLBACK_FUNCTION__ + 20
CoreFoundation 0x10a860ff2 __CFRunLoopDoTimer + 1026
CoreFoundation 0x10a86085a __CFRunLoopDoTimers + 266
CoreFoundation 0x10a85aefc __CFRunLoopRun + 2220
CoreFoundation 0x10a85a302 CFRunLoopRunSpecific + 626
GraphicsServices 0x1127d52fe GSEventRunModal + 65
UIKitCore 0x11b5dfba2 UIApplicationMain + 140
Stock 0x103ff37d0 main + 112
libdyld.dylib 0x111b48541 start + 1



我检测到的信息如上，+后面的数字是什么意思呢？数值越大，表示占用越多？

2019-04-30  

刘松野『怠惰的编程师』



QOS_CLASS_UTILITY指定block的Qos，是不是和设置队列优先级效果一样？

作者回复: 是的

2019-04-30  

孤独的码者

iOS需要强制导入IOKit框架吗

2019-04-28  



Calabash_Boy

获取电量方法里，为什么循环里有个return呢？那就没必要写循环了吧

2019-04-25  

赫小僧

请教个问题，dispatch_block_create_with_qos_class 这种方式创建出的队列进行复杂计算的时候对电量有优化，相关信息可以去哪查看呢？

2019-04-23  

孙启超

请问老师一个问题，下面这段代码如何在断点时插入到有问题的方法中：

```
thread_act_array_t threads;
mach_msg_type_number_t threadCount = 0;
const task_t thisTask = mach_task_self();
kern_return_t kr = task_threads(thisTask, &threads, &threadCount);
```

2019-04-22  