iOS启动时间优化

分享人：洪辉

iOS启动时间优化

应用启动时间，直接影响用户对一款应用的判断和使用体验。

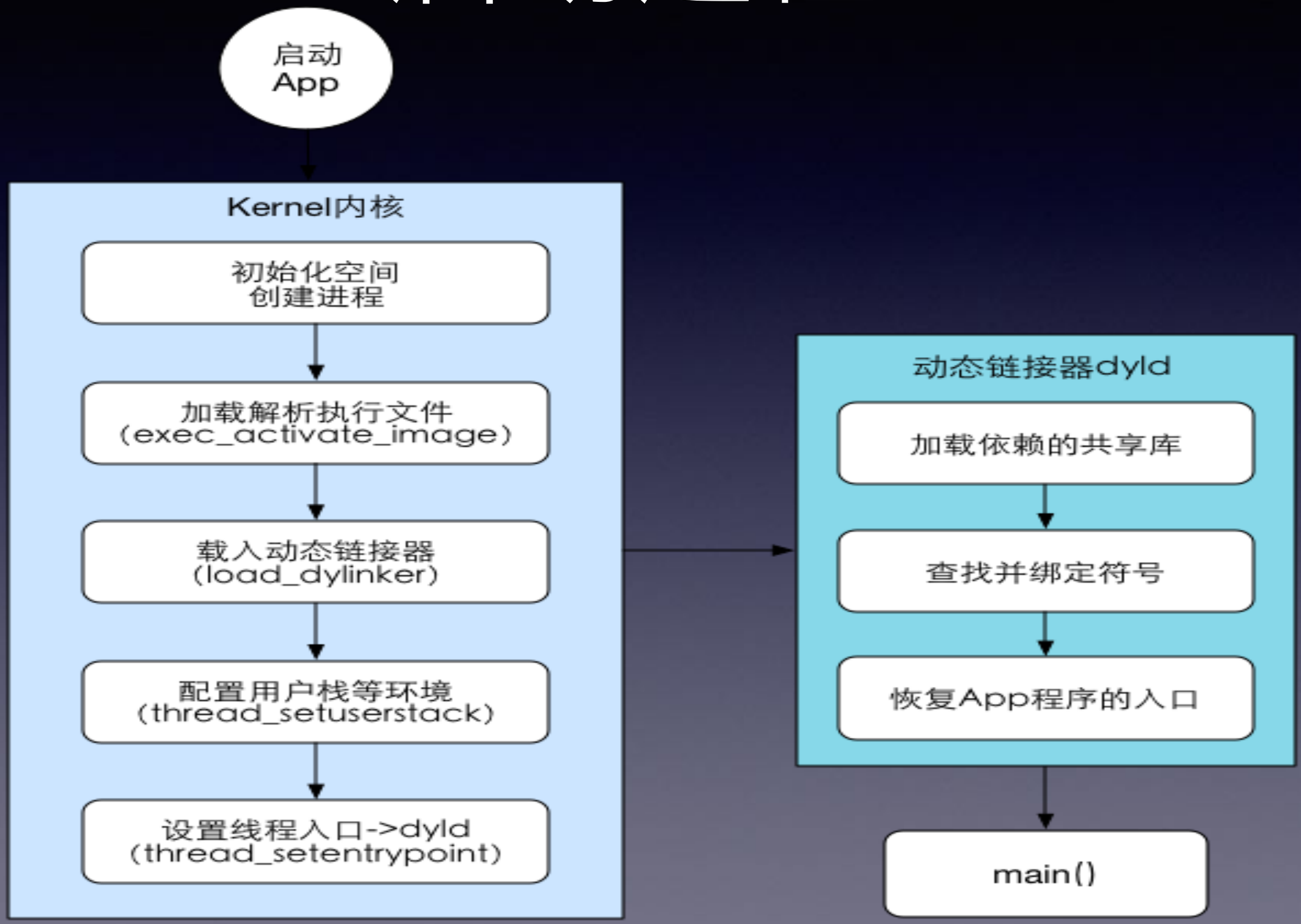
1、启动过程

2、启动时间标准

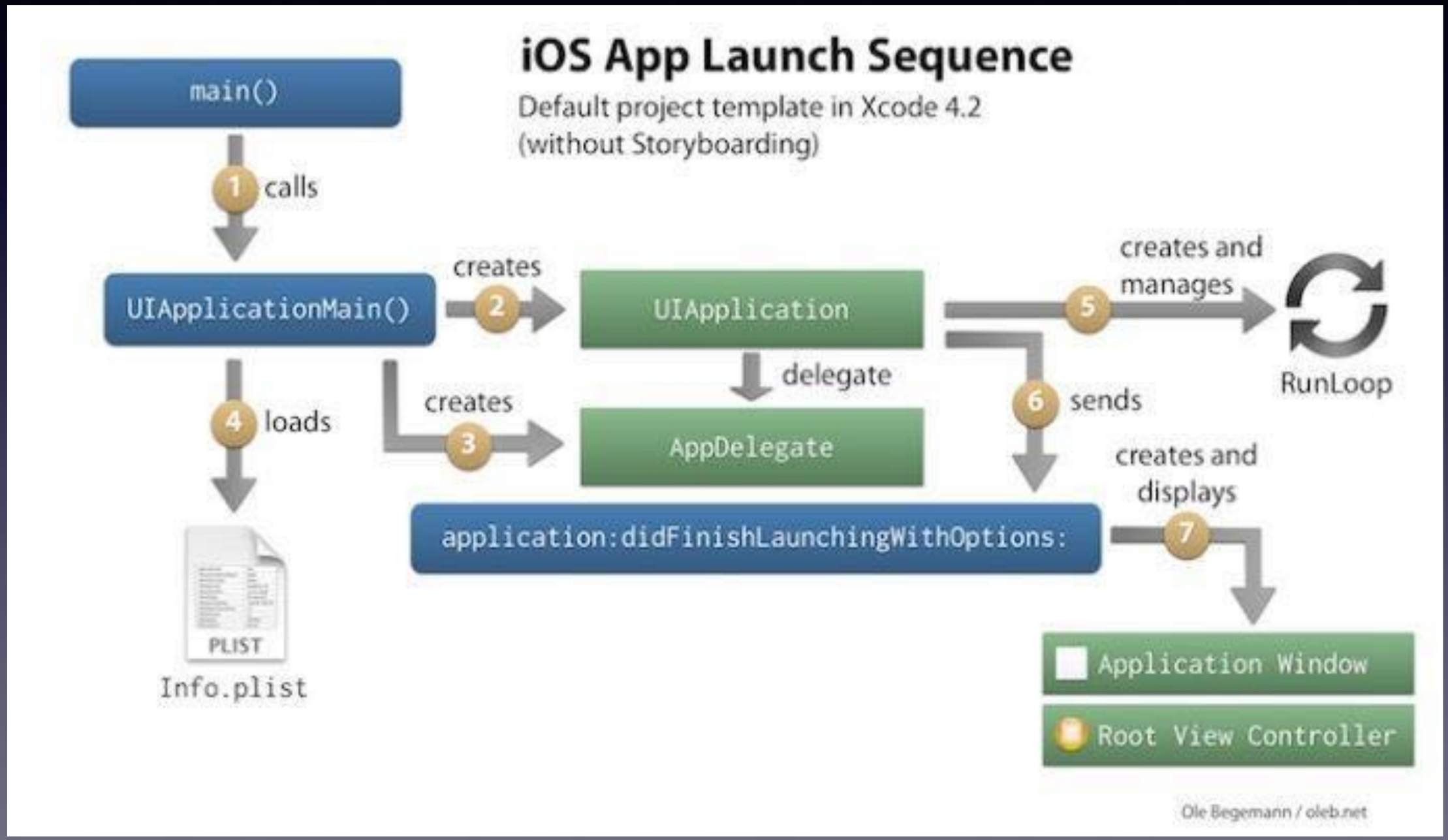
3、如何测量启动时间

4、如何优化

启动过程


启动
App
Kernel内核
初始化空间
创建进程
加载解析执行文件
(exec_activate_image)
载入动态链接器
(load_dylinker)
配置用户栈等环境
(thread_setuserstack)
设置线程入口->dyld
(thread_setentrypoint)
动态链接器dyld
加载依赖的共享库
查找并绑定符号
恢复App程序的入口
main()

启动过程


iOS App Launch Sequence
Default project template in Xcode 4.2
(without Storyboarding)
main()
1 calls
UIApplicationMain()
creates
2
UIApplication
creates and
manages
5
RunLoop
delegate
4 loads
creates
3
AppDelegate
6 sends
creates and
displays
7
application:didFinishLaunchingWithOptions:
PLIST
Info.plist
Application Window
Root View Controller
Ole Begemann / oleb.net

启动时间标准

1、不同设备上 App 启动速度不一样

2、启动时间最好控制在 400ms

3、启动时间一旦超过 20s，系统会认为发生了死循环并杀掉 App 进程

4、启动时间最好以 App 所支持的最低配置设备为准

启动时间标准

App启动分类

1、热启动

App 和数据已经在内存中，如杀死App再次启动

2、冷启动

App不在内核缓冲存储器中，如重启设备，再次打开App

冷启动耗时才是我们需要测量的重要数据

如何测量启动时间

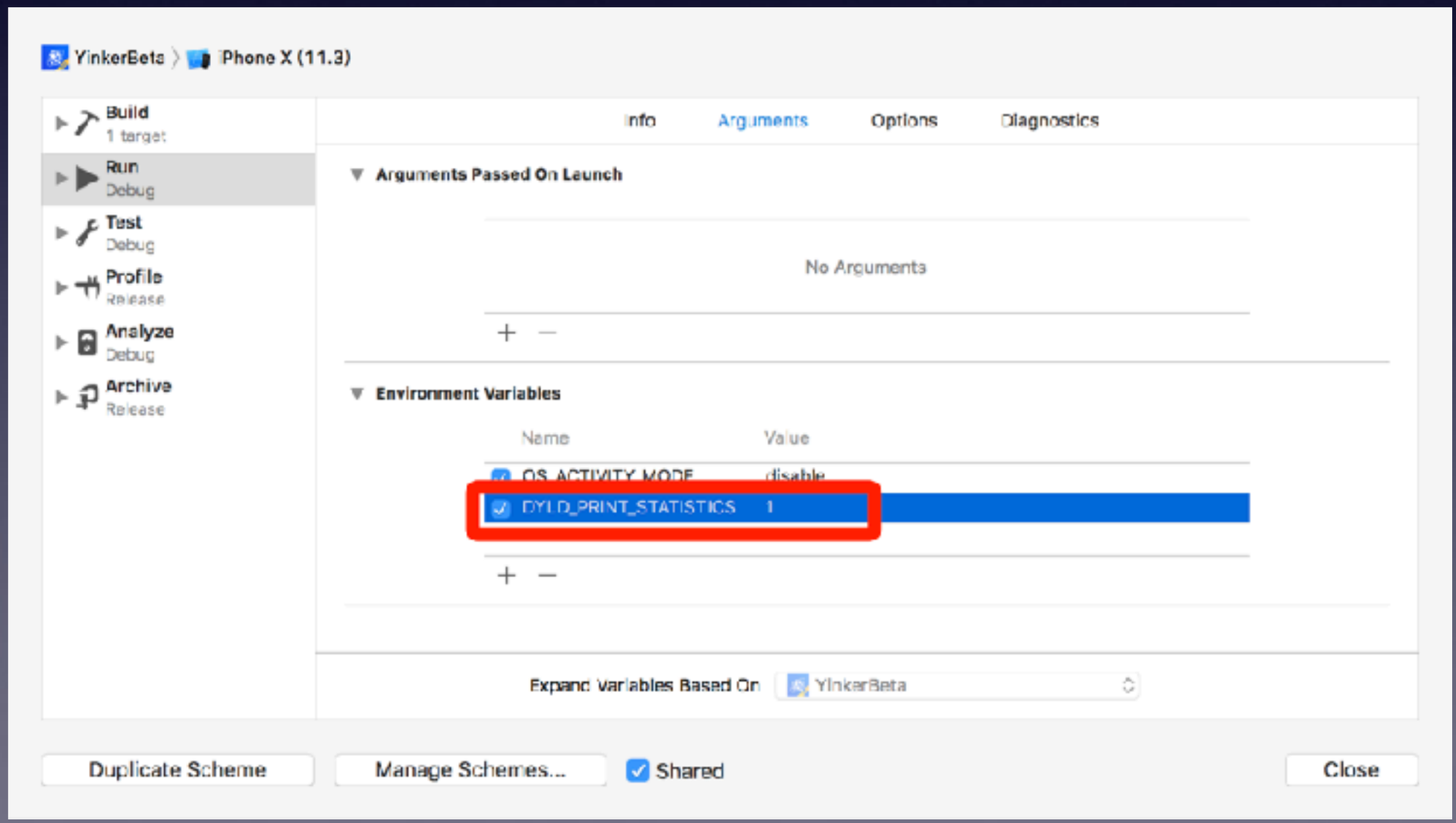
1、t(App总启动时间) = t1(pre-main加载时间) + t2(main之后的加载时间)。

2、t1 = 自身App可执行文件的加载， 系统dylib(动态链接库)加载；

3、t2 = main方法执行之后到AppDelegate类中的-(BOOL)Application:(UIApplication *)Application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions方法执行结束前这段时间

pre-main加载时间

在 Xcode 中 Edit scheme -> Run -> Auguments 将环境变量 DYLD_PRINT_STATISTICS 设为 1



pre-main加载时间

```
Total pre-main time: 1.0 seconds (100.0%)
         dylib loading time: 154.46 milliseconds (14.7%)
        rebase/binding time: 632.29 milliseconds (60.4%)
            ObjC setup time: 155.73 milliseconds (14.9%)
           initializer time: 102.53 milliseconds (9.8%)
       slowest intializers :
           libSystem.dylib :   4.76 milliseconds (0.4%)
          MediaServices :  23.23 milliseconds (2.2%)
             YinkerBeta :  94.55 milliseconds (9.0%)
```

pre-main加载时间

1、load dylibs。这一阶段dyld会分析应用依赖的dylib，找到其mach-o文件，打开和读取这些文件并验证其有效性，接着会找到代码签名注册到内核，最后对dylib的每一个segment调用mmap()。

一般情况下，iOS应用会加载100-400个dylibs，其中大部分是系统库，这部分dylib的加载系统已经做了优化。

pre-main加载时间

2、Rebase/Bind。在dylib的加载过程中，系统为了安全考虑，引入了ASLR（Address Space Layout Randomization）技术和代码签名。由于ASLR的存在，镜像（Image，包括可执行文件、dylib和bundle）会在随机的地址上加载，和之前指针指向的地址（preferred_address）会有一个偏差（slide），dyld需要修正这个偏差，来指向正确的地址。

Rebase做的是将镜像读入内存，修正镜像内部的指针，性能消耗主要在IO。Bind做的是查询符号表，设置指向镜像外部的指针，性能消耗主要在CPU计算。

pre-main加载时间

3、Objc setup。大部分ObjC初始化工作已经在Rebase/Bind阶段做完了，这一步dyld会注册所有声明过的ObjC类，将分类插入到类的方法列表里，再检查每个selector的唯一性。

pre-main加载时间

4、Initializers。这一阶段，dyld开始运行程序的初始化函数，调用每个Objc类和分类的+load方法，调用C/C++ 中的构造器函数（用attribute((constructor))修饰的函数），和创建非基本类型的C++静态全局变量。Initializers阶段执行完后，dyld开始调用main()函数。

main过程加载时间

这一阶段主要是didFinishLaunchingWithOptions方法到首页的viewDidAppear消耗的时间。

在didFinishLaunchingWithOptions方法里，我们会创建应用的window，指定其rootViewController，调用window的makeKeyAndVisible方法让其可见。我们会初始化一些三方库，设置系统UI风格，检查是否需要显示引导页、是否需要登录、是否有新版本等

如何优化

1、pre-main过程优化

2、main过程优化

pre-main过程优化

(1) 合并dylib，减少dylib数量；使用静态库

(2) 减少ObjC类（class）、方法（selector）、分类（category）的数量

(3) 减少C++虚函数的的数量（创建虚函数表有开销）

(4) 使用Swift structs（内部做了优化，符号数量更少）

(5) 少在类的+load方法里做事情，尽量把这些事情推迟到+initiailize

(6) 减少构造器函数个数，在构造器函数里少做些事情

(7) 减少C++静态全局变量的个数

main过程优化

1、梳理各个三方库，找到可以延迟加载的库，做延迟加载处理。

2、梳理业务逻辑，把可以延迟执行的逻辑，做延迟执行处理。比如检查新版本等逻辑。

3、避免复杂/多余的计算。

4、避免在首页控制器的viewDidLoad和viewWillAppear做太多事情

5、首页控制器用纯代码方式来构建。

Find 〉Text 〉Containing

UIAccessibilityElement

In Project　　Ignoring Case

11 results in 11 files

- UDTTTAttributedLabel.m Yinker
  - + (void)load {
- UITabBarItem+CYLTabBarControllerExtention.m Yinker
  - + (void)load {
- SASwizzler.m Yinker
  - + (void)load {
- UIActionSheet+AutoTrack.m Yinker
  - //+ (void)load {
- UIAlertView+AutoTrack.m Yinker
  - //+ (void)load {
- UICollectionView+AutoTrack.m Yinker
  - //+ (void)load {
- UIImage+AutoTrack.m Yinker
  - //+ (void)load {
- UITabBar+AutoTrack.m Yinker
  - + (void)load {
- UITableView+AutoTrack.m Yinker
  - //+ (void)load {
- NSDate+DateTools.m Yinker
  - + (void)load {
- TTTAttributedLabel.m Yinker
  - + (void)load {

Yinker 〉Yinker 〉YKBaseFrameWork 〉YKBaseUI 〉TTTAttributedLabel.m 〉TTTAttributedLabel

```
357 #if __IPHONE_OS_VERSION_MAX_ALLOWED >= 70000
358
359 #ifndef kCFCoreFoundationVersionNumber_iOS_7_0
360 #define kCFCoreFoundationVersionNumber_iOS_7_0 847.2
361 #endif

+ (void)load {
    static dispatch_once_t onceToken;
    dispatch_once(&onceToken, ^{
        if (kCFCoreFoundationVersionNumber < kCFCoreFoundationVersionNumber_iOS_7_0) {
            Class class = [self class];
            Class superclass = class_getSuperclass(class);

            NSArray *strings = @[
                                 NSStringFromSelector(@selector(isAccessibilityElement)),
                                 NSStringFromSelector(@selector(accessibilityElementCount)),
                                 NSStringFromSelector(@selector(accessibilityElementAtIndex:)),
                                 NSStringFromSelector(@selector(indexOfAccessibilityElement:)),
                                 ];

            for (NSString *string in strings) {
                SEL selector = NSSelectorFromString(string);
                IMP superImplementation = class_getMethodImplementation(superclass, selector);
                Method method = class_getInstanceMethod(class, selector);
                const char *types = method_getTypeEncoding(method);
                class_replaceMethod(class, selector, superImplementation, types);
            }
        }
    });
}
#endif
```

优化前

```
2018-04-23 10:43:33.982256+0800 YinkerBeta[6457:1931285]
didFinishLaunchingWithOptions start time 546144213982.212
2018-04-23 10:43:34.282441+0800 YinkerBeta[6457:1931285]
didFinishLaunchingWithOptions return time 546144214282.41

2018-04-23 10:43:34.297740+0800 YinkerBeta[6457:1931285]
YKHomeViewController didload start time 546144214297.73.
2018-04-23 10:43:34.454468+0800 YinkerBeta[6457:1931285]
YKHomeViewController didappear end time 546144214454.457
```

didFinishLaunchingWithOptions (return time - start time) = 300.2ms

(HomeDidAppear - didFinishLaunchingWithOptions start time) = 472.2ms

```objc
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    //clear UIWebview Cache
    [[NSURLCache sharedURLCache] removeAllCachedResponses];
    [[NSURLCache sharedURLCache] setDiskCapacity:0];
    [[NSURLCache sharedURLCache] setMemoryCapacity:0];

    //进入App事件
    [YKUMMgr event:@"start_app"];
    //清除角标
    [[UIApplication sharedApplication] setApplicationIconBadgeNumber:0];

    self.window = [[UIWindow alloc]init];
    self.window.frame = [UIScreen mainScreen].bounds;
    self.window.backgroundColor = [UIColor whiteColor];

    //初始化根视图
    YKTabBarControllerConfig *tabBarControllerConfig = [[YKTabBarControllerConfig alloc] init];
    self.tabBarControllerConfig = tabBarControllerConfig;
    self.window.rootViewController = tabBarControllerConfig.tabBarController;
    [self.window makeKeyAndVisible];

    //显示引导页
    [YKAppLaunchFlowManager startAppLaunchFlow];
    //友盟配置
    [[YKUMMgr sharedYKUMMgr] configUmengSettings];
    //键盘配置
    [IQKeyBoardCustomManager enableKeyBoard];

    //OneAPM
    [OneAPM startWithApplicationToken:[YKAppInfo oneAPMAppToken]];
    // Udesk
    [YKUdeskManager configUdeskUserInfo];
    //埋点
    [YKDataBuryPointManager configDataBuryPoint];
    [YKAllDataBuryPointService reportUserPushOpenedEveryDay];

    // 注册极光推送
    [YKJPRemoteNotificationMgr registerForRemoteNotification:application didFinishLaunchingWithOptions:launchOptions];
    return YES;
}
```

优化后

```objc
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    NSLog(@"didFinishLaunchingWithOptions start time %@", @(CFAbsoluteTimeGetCurrent()*1000));
    self.window = [[UIWindow alloc]init];
    self.window.frame = [UIScreen mainScreen].bounds;
    self.window.backgroundColor = [UIColor whiteColor];

    //初始化跟视图
    YKTabBarControllerConfig *tabBarControllerConfig = [[YKTabBarControllerConfig alloc] init];
    self.tabBarControllerConfig = tabBarControllerConfig;
    self.window.rootViewController = tabBarControllerConfig.tabBarController;
    [self.window makeKeyAndVisible];

    //友盟配置
    [[YKUMMgr sharedYKUMMgr] configUmengSettings];
    //进入App事件
    [YKUMMgr event:@"start_app"];

    //OneAPM
    [OneAPM startWithApplicationToken:[YKAppInfo oneAPMAppToken]];
    // 埋点
    [YKDataBuryPointManager configDataBuryPoint];

    // 注册极光推送
    [YKJPRemoteNotificationMgr registerForRemoteNotification:application didFinishLaunchingWithOptions:launchOptions];
    NSLog(@"didFinishLaunchingWithOptions return time %@", @(CFAbsoluteTimeGetCurrent()*1000));

    return YES;
```

```
2018-04-23 11:37:12.820361+0800 YinkerBeta[6485:1941960]
didFinishLaunchingWithOptions start time 546147432820.312
2018-04-23 11:37:13.073278+0800 YinkerBeta[6485:1941960]
didFinishLaunchingWithOptions return time 546147433073.24


2018-04-23 11:37:13.089106+0800 YinkerBeta[6485:1941960]
YKHomeViewController didload start time 546147433089.0959
2018-04-23 11:37:13.266187+0800 YinkerBeta[6485:1941960]
YKHomeViewController didappear end time 546147433266.1771
```

didFinishLaunchingWithOptions (return time - start time) = 252.9ms
(HomeDidAppear - didFinishLaunchingWithOptions start time) = 445.8ms

小结

1、删除不用的，多余的(库，文件，类，方法)

2、合并重复的(库，类等)

3、延迟加载

参考资料

http://yulingtianxia.com/blog/2016/02/27/TFSHelper/

http://www.cocoachina.com/ios/20170716/19876.html

https://segmentfault.com/a/1190000007769327

https://techblog.toutiao.com/2017/01/17/iosspeed/

https://developer.apple.com/videos/play/wwdc2016/406/

That’s All, thanks!