原 深入浅出 - Android系统移植与平台开发（七） - 初识HAL

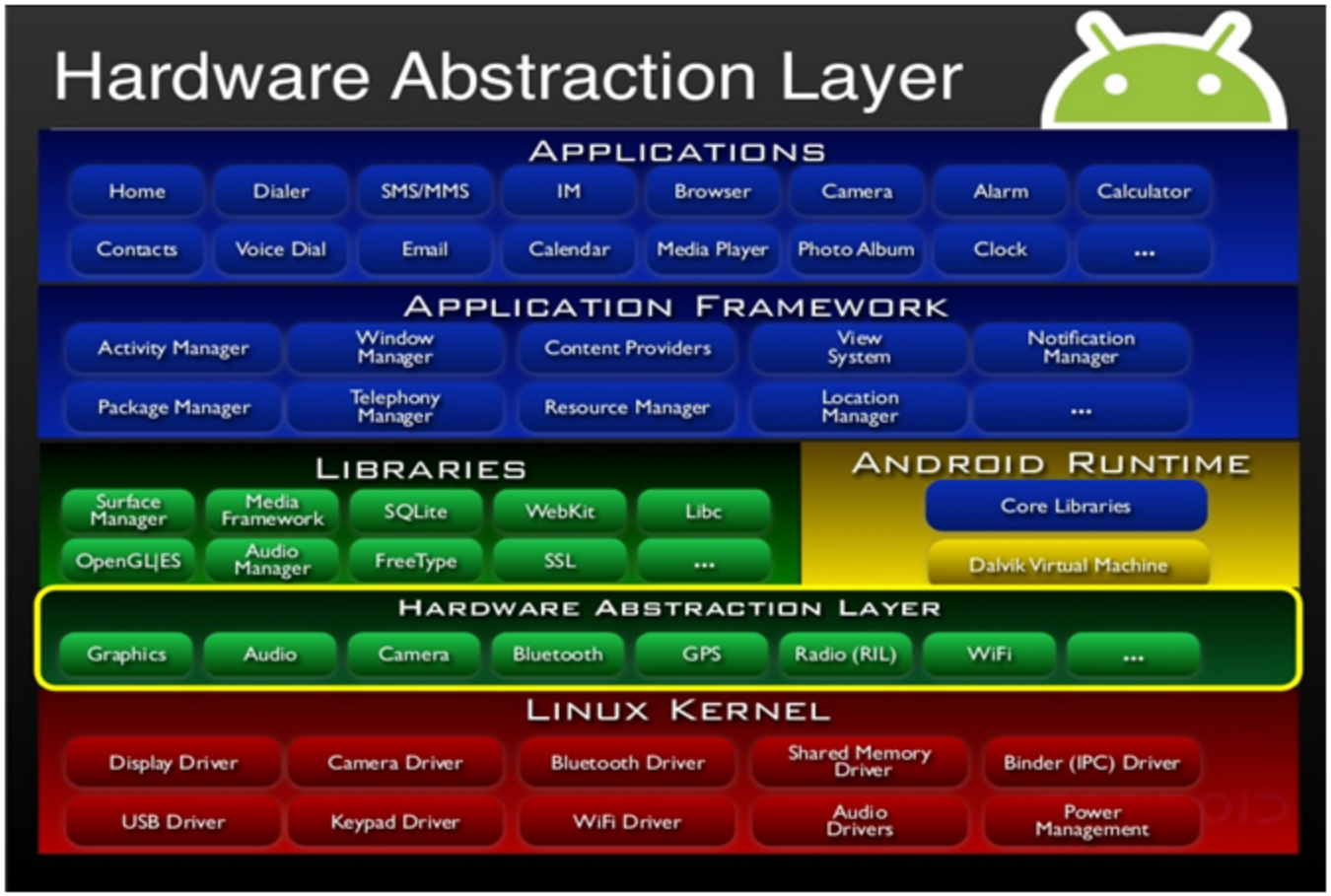
标签： android 平台 module 硬件驱动 框架 java

2012-10-14 13:35 14452人阅读 评论(6) 收藏 举报

分类： Android移植（59）

目录(?) [+]

1. HAL的module与stub



HAL（Hardware AbstractLayer）硬件抽象层是Google开发的Android系统里上层应用对底层硬件操作屏蔽一个软件层次，说白了，就是上层的应用不用

关心底层硬件具体如何工作的，只要向上层提供一个统一的接口即可，这种设计思想广泛的存在于当前的软件架构设计里。

严格来讲，Android系统里完全可以没有HAL硬件抽象层，上层应用层可以通过API调用到底层硬件，但是Android自出现开始一直打着开源的旗号，而一些硬件厂商由于商业因素，不希望自己的核心代码开源出来，而只是提供二进制代码。另外，Android系统里使用的一些硬件设备接口可能不是使用的Linux Kernel的统一接口，并且还有GPL版权的原因，所以Google不利己，在Android的架构里提出了HAL的概念，这个HAL其实就是硬件独立的意思，Android系统不依赖于某一个具体的硬件驱动，而是依赖于HAL代码，这样，第三方厂商可以将自己不开源的代码封装在HAL层，仅提供二进制。

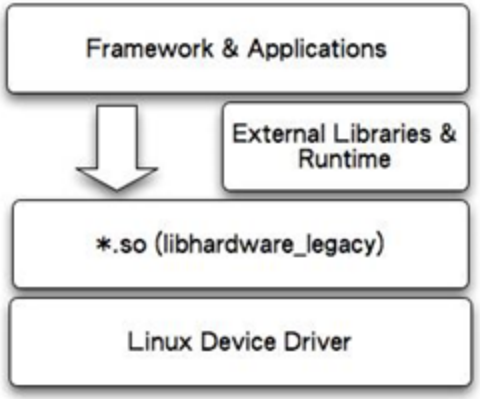
HAL的架构分为两种：

Ø 旧的架构module

Ø 新的架构modulestub

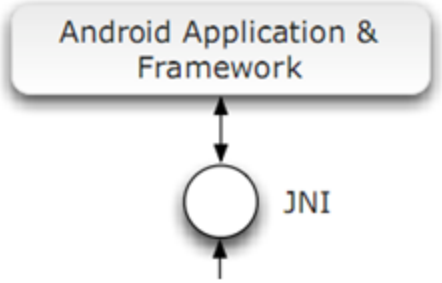
1.1 module架构

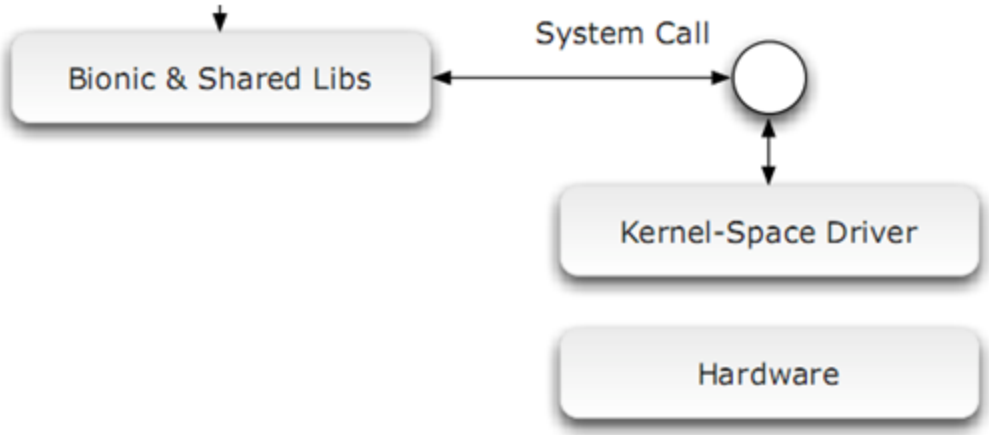
旧的架构比较好理解，Android用户应用程序或框架层代码由Java实现，Java运行在Dalvik虚拟机中，没有办法直接访问底层硬件，只能通过调用so本地库代码实现，在so本地库代码里有对底层硬件操作代码，如下图所示：



也就是说，应用层或框架层Java代码，通过JNI技术调用C或C++写的so库代码，在so库代码中调用底层驱动，实现上层应用的提出的硬件请求操作。实现硬件操作的so库为：module。

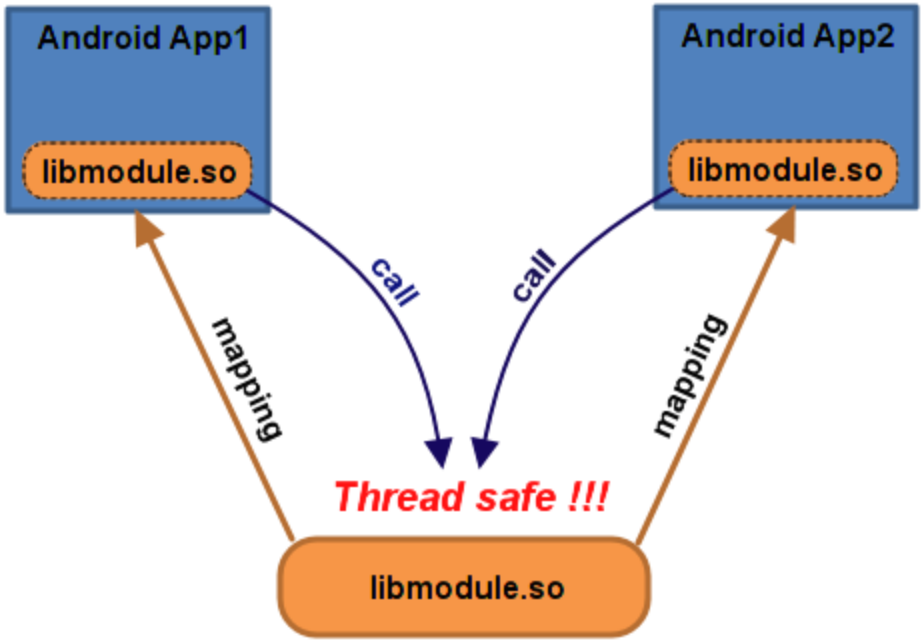
其实现流程如下图：



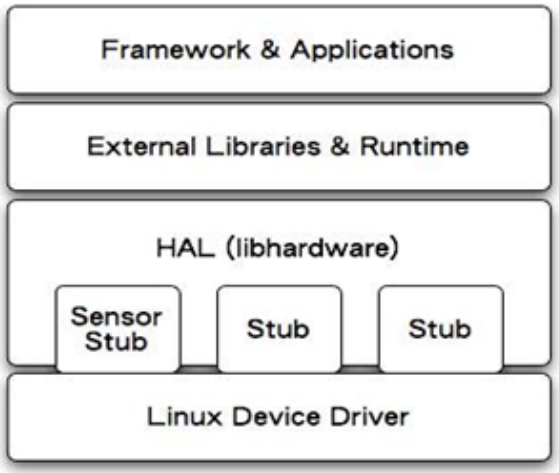


由此可见，Java代码要访问硬件效率其实挺低的，没有C代码效率高，但是Android系统在软件框架和硬件处理器上都在减少和C代码执行效率的差距，据国外测试的结果来看，基本上能达到C代码效率的95%左右。

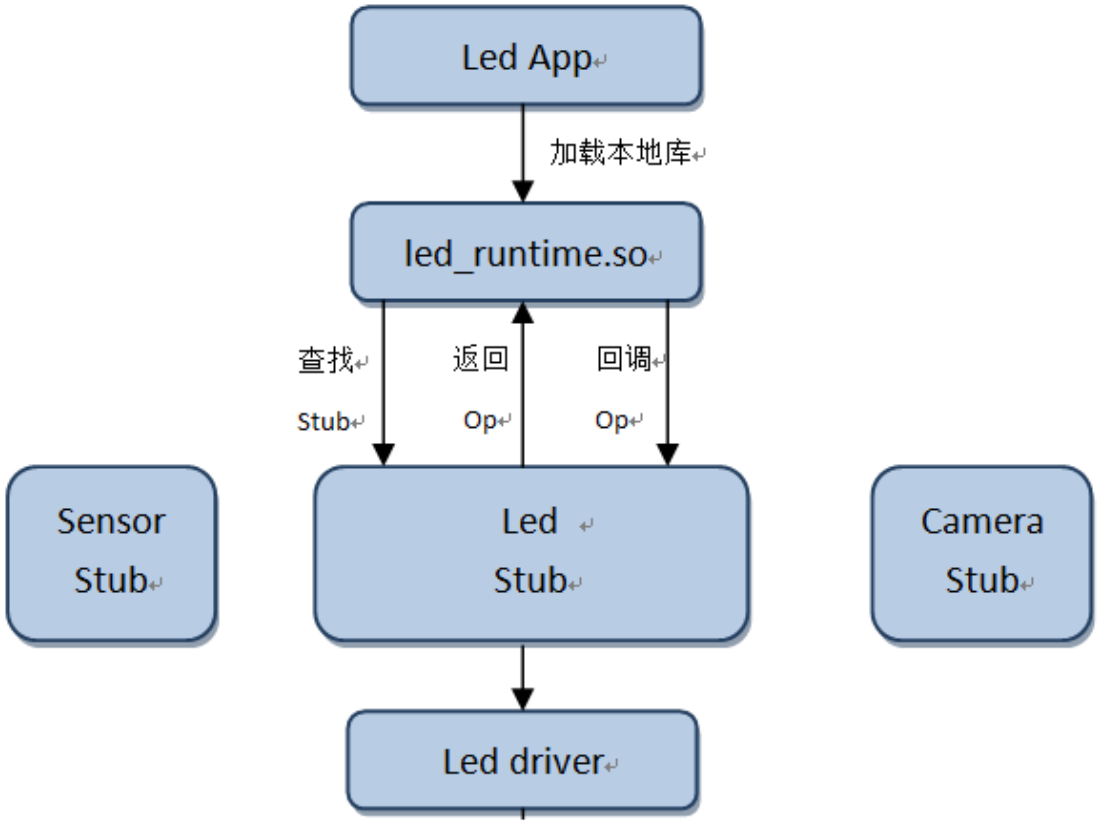
这种设计架构虽然满足了Java应用访问硬件的需要，但是，使得我们的代码上下层次间的耦合太高，用户程序或框架代码必须要去加载module库，如果底层硬件有变化，moudle要重新编译，上层也要做相应的变化，另外，如果多个应用程序同时访问硬件，都去加载module，同一module被多个进程映射多次，会有代码的重入问题。因此，Google又提出了新的HAL架构。

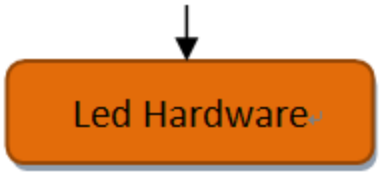


1.2 新的HAL架构



新的架构使用的是module stub方式。Stub是存根或桩的意思，其实说白了，就是指一个对象代表的意思。由上面的架构可知，上层应用层或框架层代码加载so库代码，so库代码我们称为module，在HAL层注册了每个硬件对象的存根stub，当上层需要访问硬件的时候，就从当前注册的硬件对象stub里查找，找到之后stub会向上层module提供该硬件对象的operations interface（操作接口），该操作接口就保存在了module中，上层应用或框架再通过这个module操作接口来访问硬件。如下图，以Led为例的示意图：





Led App为Android 应用程序，Led App里的Java代码不能操作硬件，将硬件操作工作交给本地module库 led_runtime.so，它从当前系统中查找Led Stub，查找到之后，Led Stub将硬件驱动操作返回给module，Led App操作硬件时，通过保存在module中的操作接口间接访问底层硬件。

问题来了：

Ø 麻烦，觉得比module方式复杂

Ø 硬件对象怎样注册为stub？

Ø 上层如何查找硬件对象的stub?

“麻烦”是确定的，但是Google这么聪明的公司不可能是光制造麻烦的公司，肯定是考虑到其它的优越性才使用这种方式。

1.3Module架构与Stub构架对比

在Module架构中，本地代码由so库实现，上层直接将so库映射进进程空间，会有代码重入及设备多次打开的问题。新的Stub框架虽然也要加载module库，但是这个module已经不包含操作底层硬件驱动的功能了，它里面保存的只是底层Stub提供的操作接口，底层Stub扮演了“接口提供者”的角色，当Stub第一次被使用时加载到内存，后续再使用时仅返回硬件对象操作接口，不会存在设备多次打开问题，并且由于多进程访问时返回的只是函数指针，代码没有重入问题。