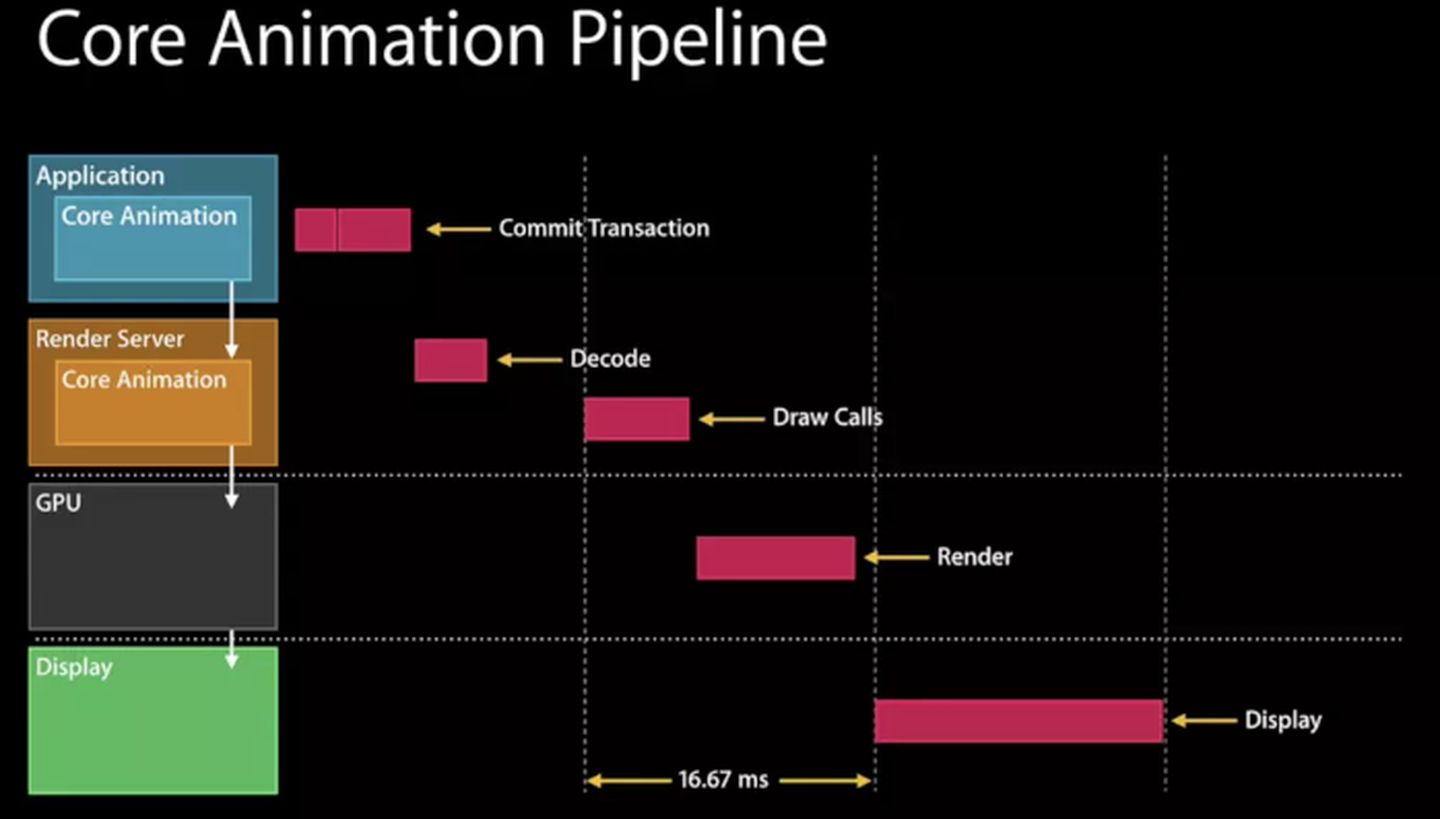
离屏渲染小节

### **iOS渲染架构**

在Application这一层中主要是CPU在操作，而到了Render Server这一层，CoreAnimation会将具体操作转换成发送给GPU的draw calls（以前是call OpenGL ES，现在慢慢转到了Metal），显然CPU和GPU双方同处于一个流水线中，协作完成整个渲染工作。

### **离屏渲染的定义**

如果要在显示屏上显示内容，我们至少需要一块与屏幕像素数据量一样大的frame buffer，作为像素数据存储区域，而这也是GPU存储渲染结果的地方。如果有时因为面临一些限制，无法把渲染结果直接写入frame buffer，而是先暂存在另外的内存区域，之后再写入frame buffer，那么这个过程被称之为离屏渲染。

### **GPU离屏渲染**

在上面的渲染流水线示意图中我们可以看到，主要的渲染操作都是由CoreAnimation的Render Server模块，通过调用显卡驱动所提供的OpenGL/Metal接口来执行的。通常对于每一层layer，Render Server会遵循“[画家算法](https://link.zhihu.com/?target=https://en.wikipedia.org/wiki/Painter's_algorithm" \t "/Users/51talk/Documents\\x/_blank)”，按次序输出到frame buffer，后一层覆盖前一层，就能得到最终的显示结果（值得一提的是，与一般桌面架构不同，在iOS中，设备主存和GPU的显存[共享物理内存](https://link.zhihu.com/?target=https://apple.stackexchange.com/questions/54977/how-much-gpu-memory-do-iphones-and-ipads-have" \t "/Users/51talk/Documents\\x/_blank)，这样可以省去一些数据传输开销）。

**对于每一层layer，要么能找到一种通过单次遍历就能完成渲染的算法，要么就不得不另开一块内存，借助这个临时中转区域来完成一些更复杂的、多次的修改/剪裁操作**。

如：

* 将一个layer的内容裁剪成圆角，可能不存在一次遍历就能完成的方法
* 容器的子layer因为父容器有圆角，那么也会需要被裁剪，而这时它们还在渲染队列中排队，尚未被组合到一块画布上，自然也无法统一裁剪

此时我们就不得不开辟一块独立于frame buffer的空白内存，先把容器以及其所有子layer依次画好，然后把四个角“剪”成圆形，再把结果画到frame buffer中。这就是GPU的离屏渲染。

### **常见离屏渲染场景分析**

cornerRadius+clipsToBounds

shouldRasterize（光栅化）  
masks（遮罩）  
shadows（阴影）  
edge antialiasing（抗锯齿）  
group opacity（不透明）  
复杂形状设置圆角等  
渐变

### **GPU离屏渲染的性能影响**

GPU的操作是高度流水线化的。本来所有计算工作都在有条不紊地正在向frame buffer输出，此时突然收到指令，需要输出到另一块内存，那么流水线中正在进行的一切都不得不被丢弃，切换到只能服务于我们当前的“切圆角”操作。等到完成以后再次清空，再回到向frame buffer输出的正常流程。

（光栅化实践）在tableView或者collectionView中，滚动的每一帧变化都会触发每个cell的重新绘制，因此一旦存在离屏渲染，上面提到的上下文切换就会每秒发生60次，并且很可能每一帧有几十张的图片要求这么做。

CALayer为这个方案提供了对应的解法：***shouldRasterize***。一旦被设置为true，Render Server就会强制把layer的渲染结果（包括其子layer，以及圆角、阴影、group opacity等等）保存在一块内存中，这样一来在下一帧仍然可以被复用，而不会再次触发离屏渲染。旨在于**降低性能损失，但总是至少会触发一次离屏渲染**。

渲染性能的调优，其实始终是在做一件事：**平衡CPU和GPU的负载，让他们尽量做各自最擅长的工作**。

处理方法

* 应用AsyncDisplayKit(Texture)作为主要渲染框架，对于文字和图片的异步渲染操作交由框架来处理
* 对于图片的圆角，统一采用“precomposite”的策略，也就是不经由容器来做剪切，而是预先使用CoreGraphics为图片裁剪圆角
* 对于视频的圆角，由于实时剪切非常消耗性能，我们会创建四个白色弧形的layer盖住四个角，从视觉上制造圆角的效果
* 对于view的圆形边框，如果没有backgroundColor，可以放心使用cornerRadius来做
* 对于所有的阴影，使用shadowPath来规避离屏渲染
* 对于特殊形状的view，使用layer mask并打开shouldRasterize（光栅化）来对渲染结果进行缓存
* 对于模糊效果，不采用系统提供的UIVisualEffect，而是另外实现模糊效果（CIGaussianBlur），并手动管理渲染结果

**<https://www.objc.io/issues/3-views/moving-pixels-onto-the-screen/>  屏幕显示原理**

**<https://www.jianshu.com/p/f9256987152b>**。**iOS**中三种切圆角的几种方法比较和图片优化