[4系统初始化 1](#_Toc105334701)

[FLEX-SC 1](#_Toc105334702)

[计算机启动 2](#_Toc105334703)

[内核代码加载与运行 3](#_Toc105334704)

# 4系统初始化

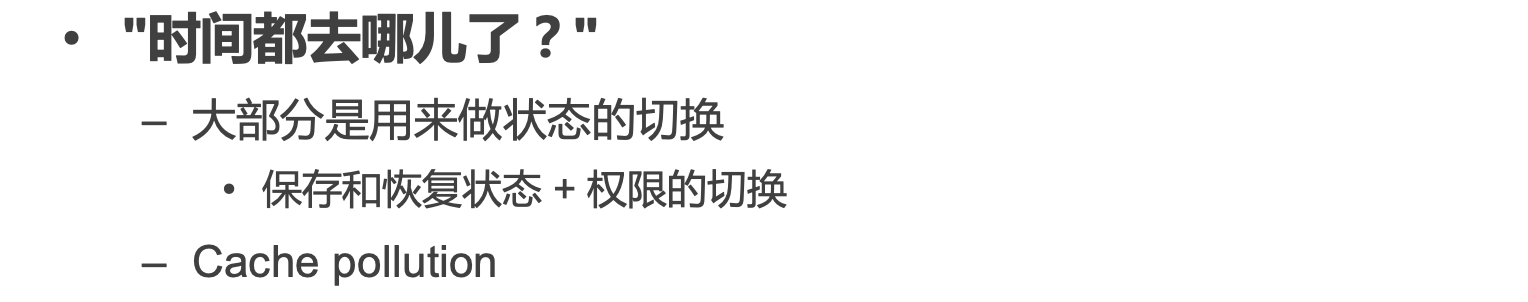
除了vDSO，还有什么办法能让系统调用的开销小一点呢？

## FLEX-SC

**> Flexible System Call Scheduling with Exception-Less System Calls**

> OSDI' 10

motivation: 大部分时间都是用来做状态的切换



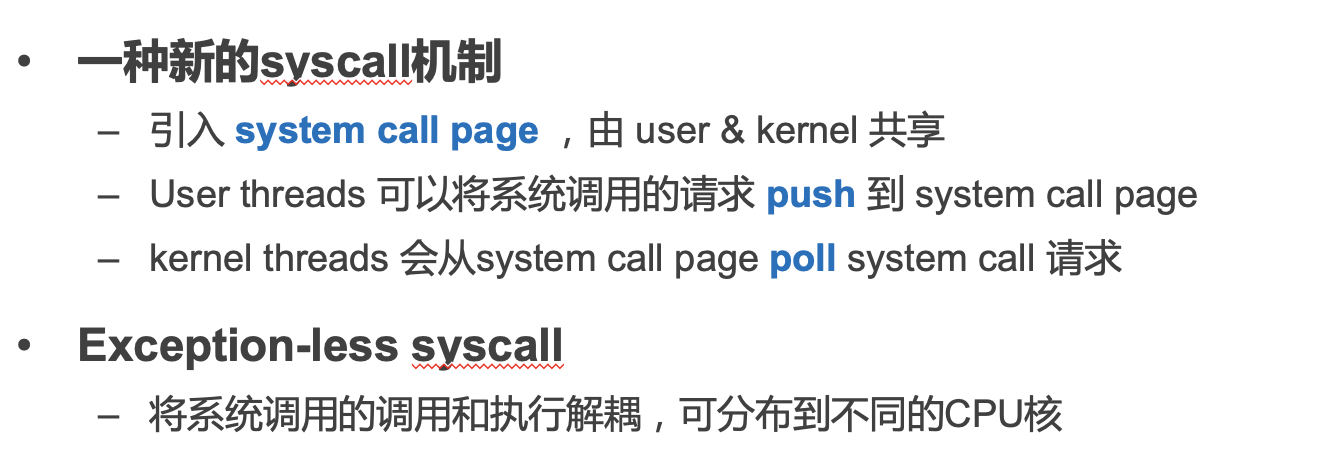
1. 保存和回复状态+权限切换

2. Cache pollution

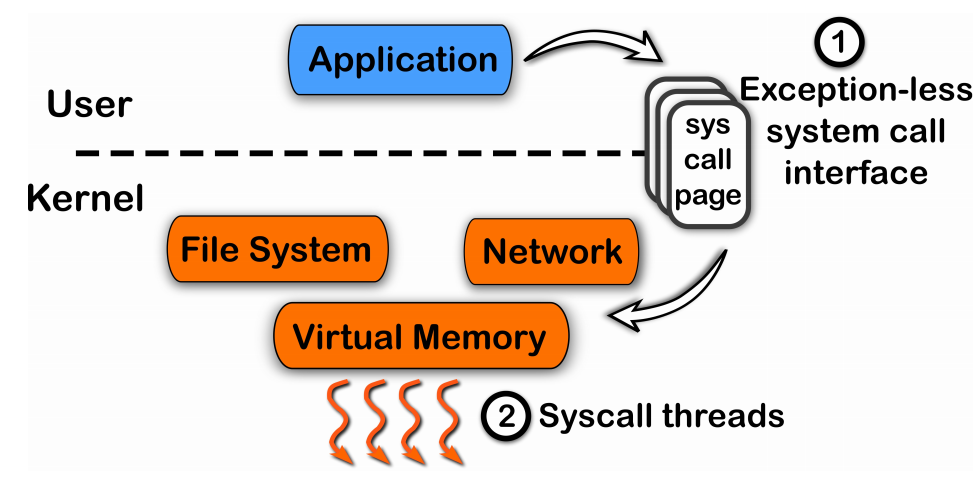
> 用户态的数据在cache里，切换到内核态，用户的数据就会被操作系统的数据占据

Q：有没有可能在不切换状态的情况下实现系统调用？

A：Flexible System Call

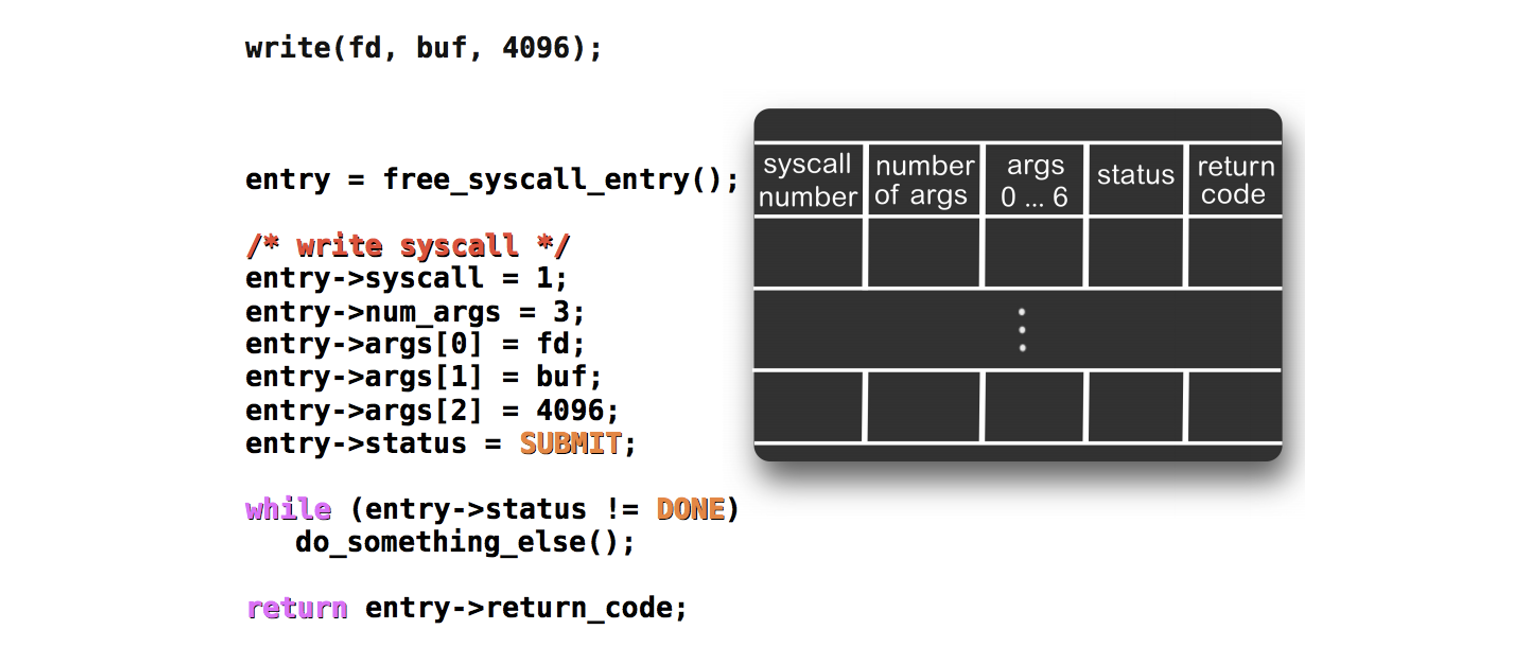


syscall变成异步了



应用把请求写到syscall page里，再由kernel执行syscall thread——可以做一些并行。

具体的过程：



从syscall page里面拿一个entry出来。

kernel做，然后做完之后把status写成一个done。

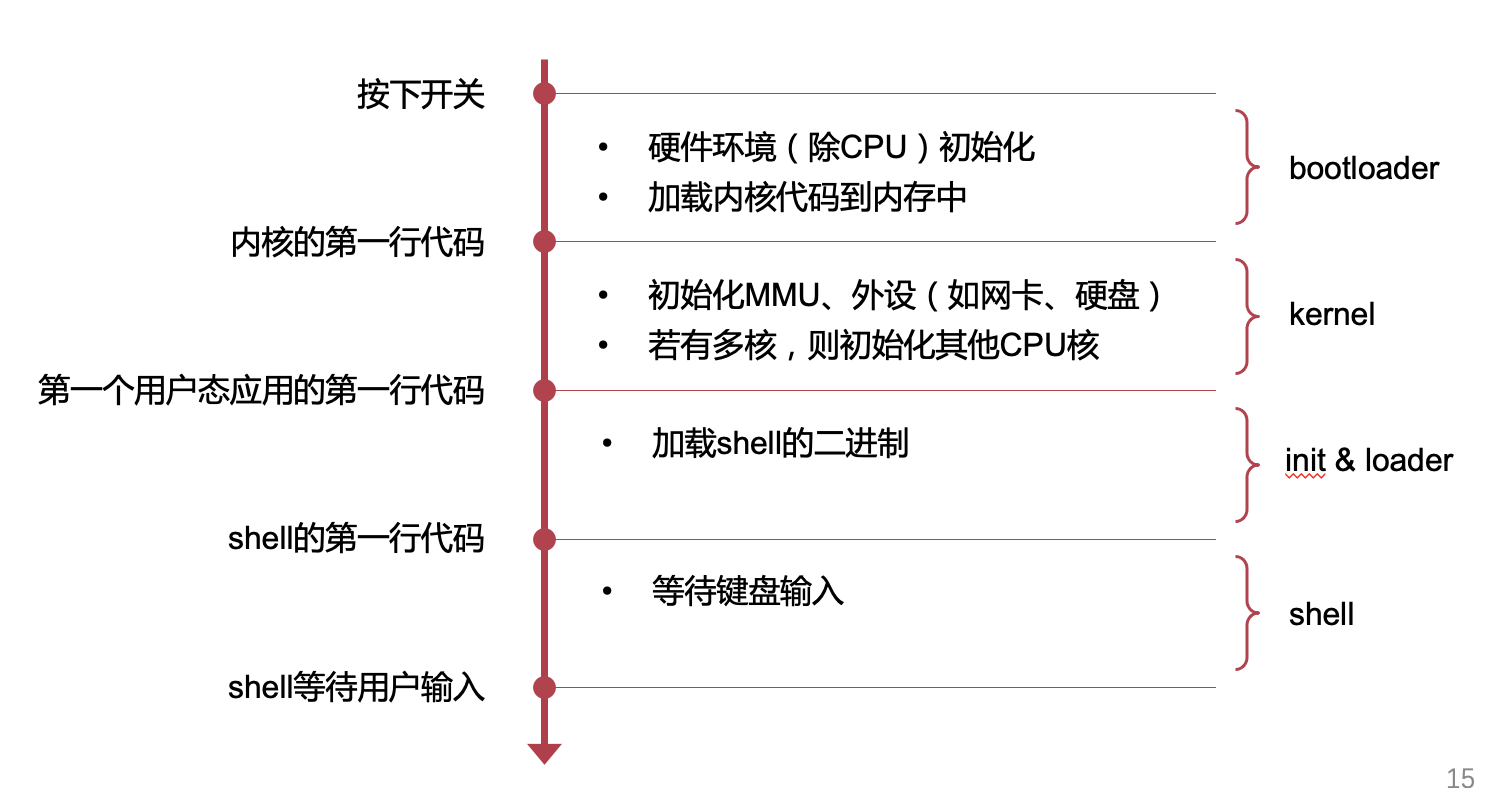
**没有切换到kernel的过程，节省了大量开销。**

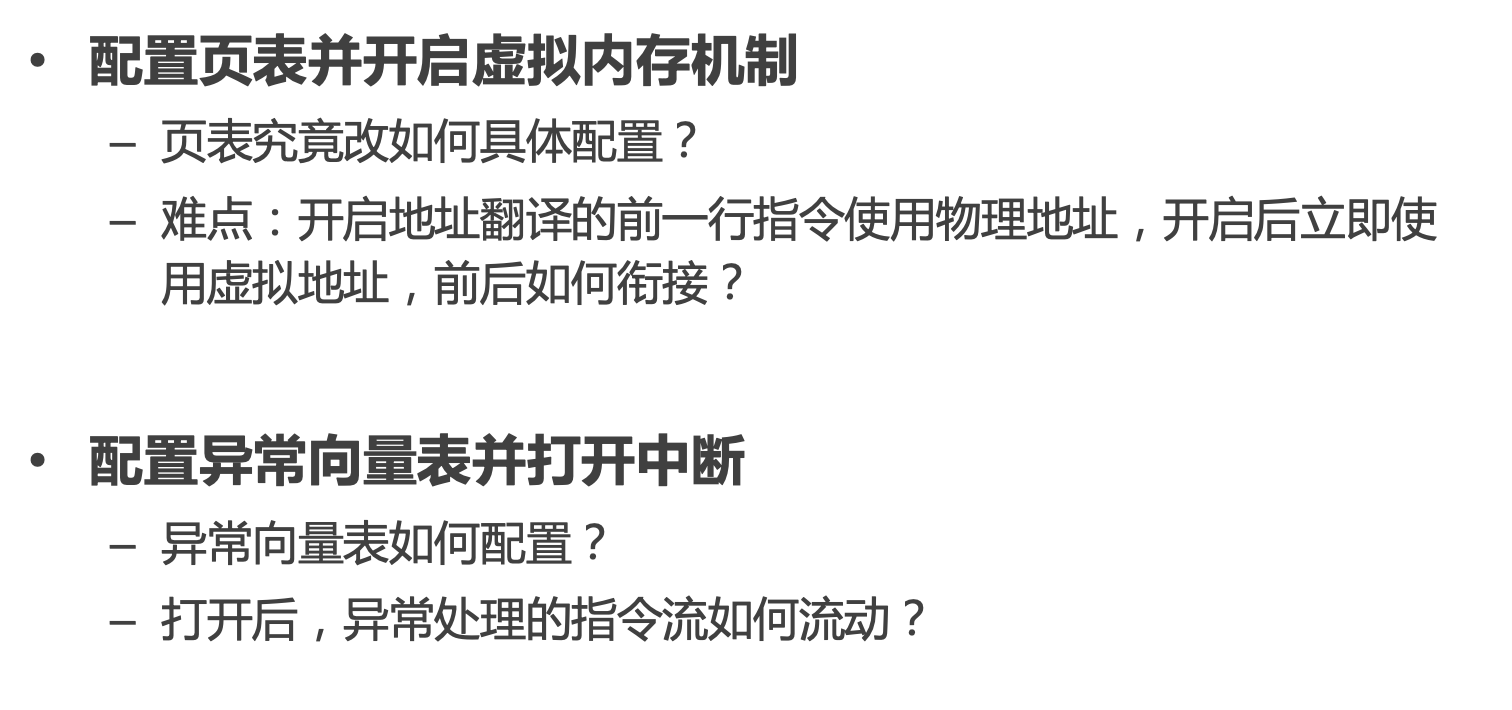
Q：FlesSC适合什么场景？

A：适合不那么紧急的事，重要且紧急的事直接调用syscall。

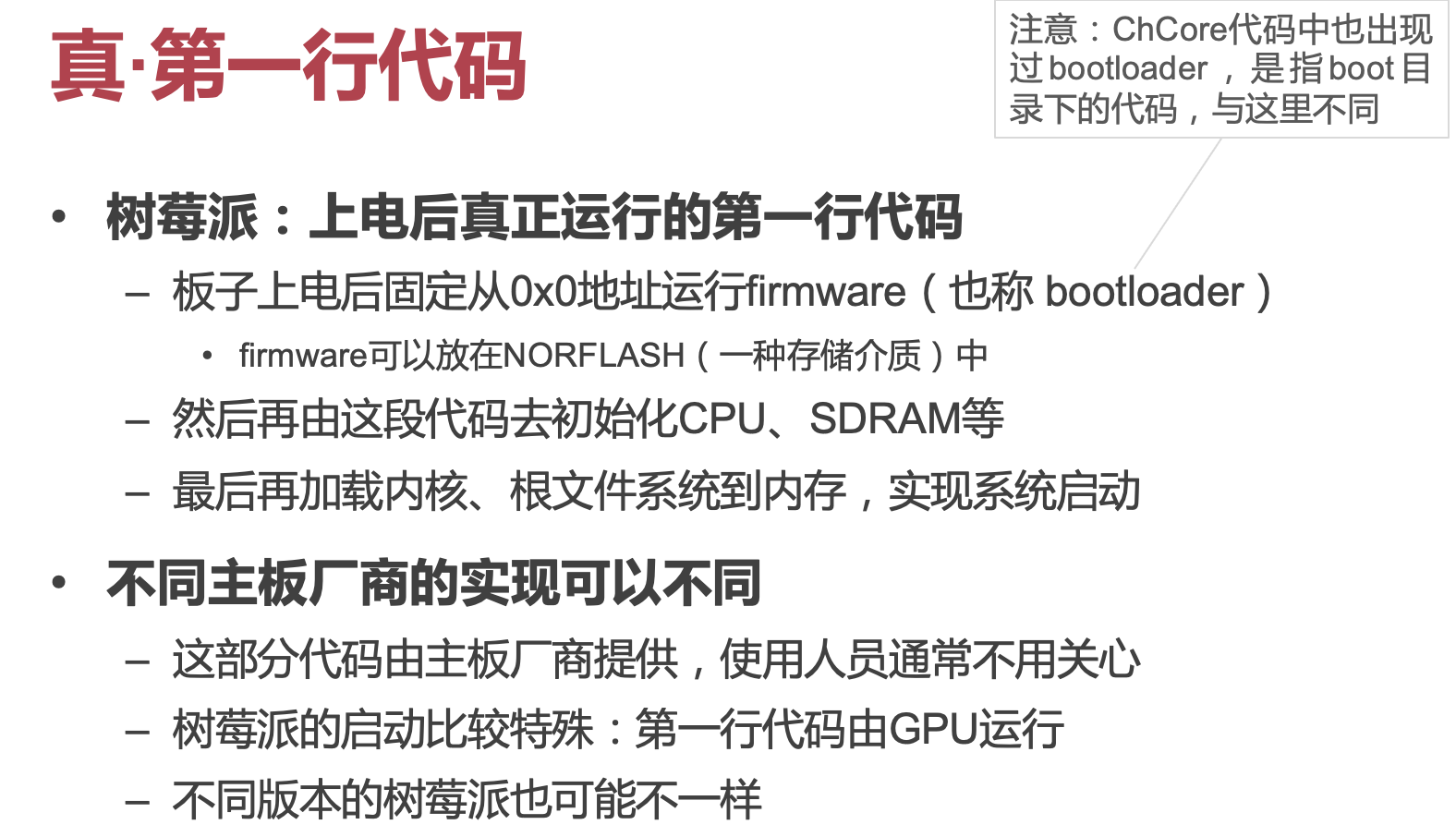
假设：应用有很多任务要做（且可以区分）比较适合，应用任务少且任务分解不了。

## 计算机启动

内核启动2个主要任务

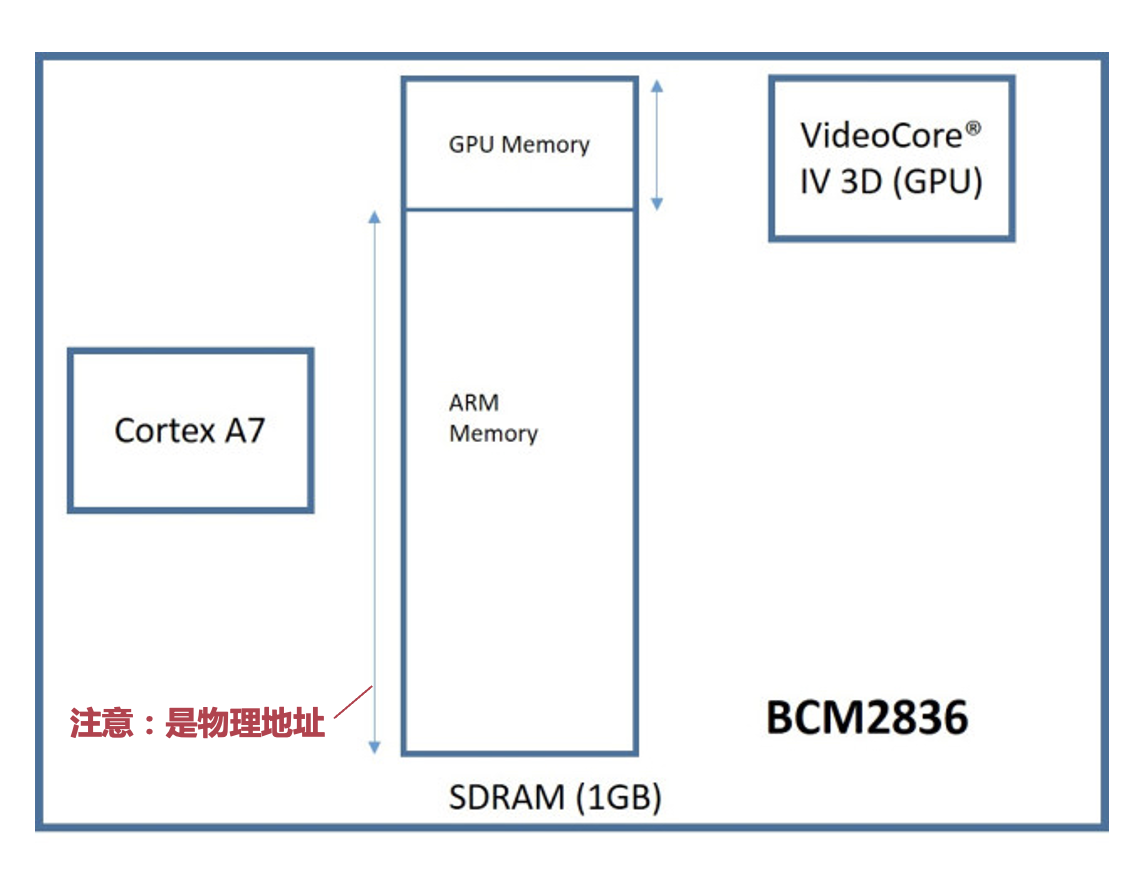


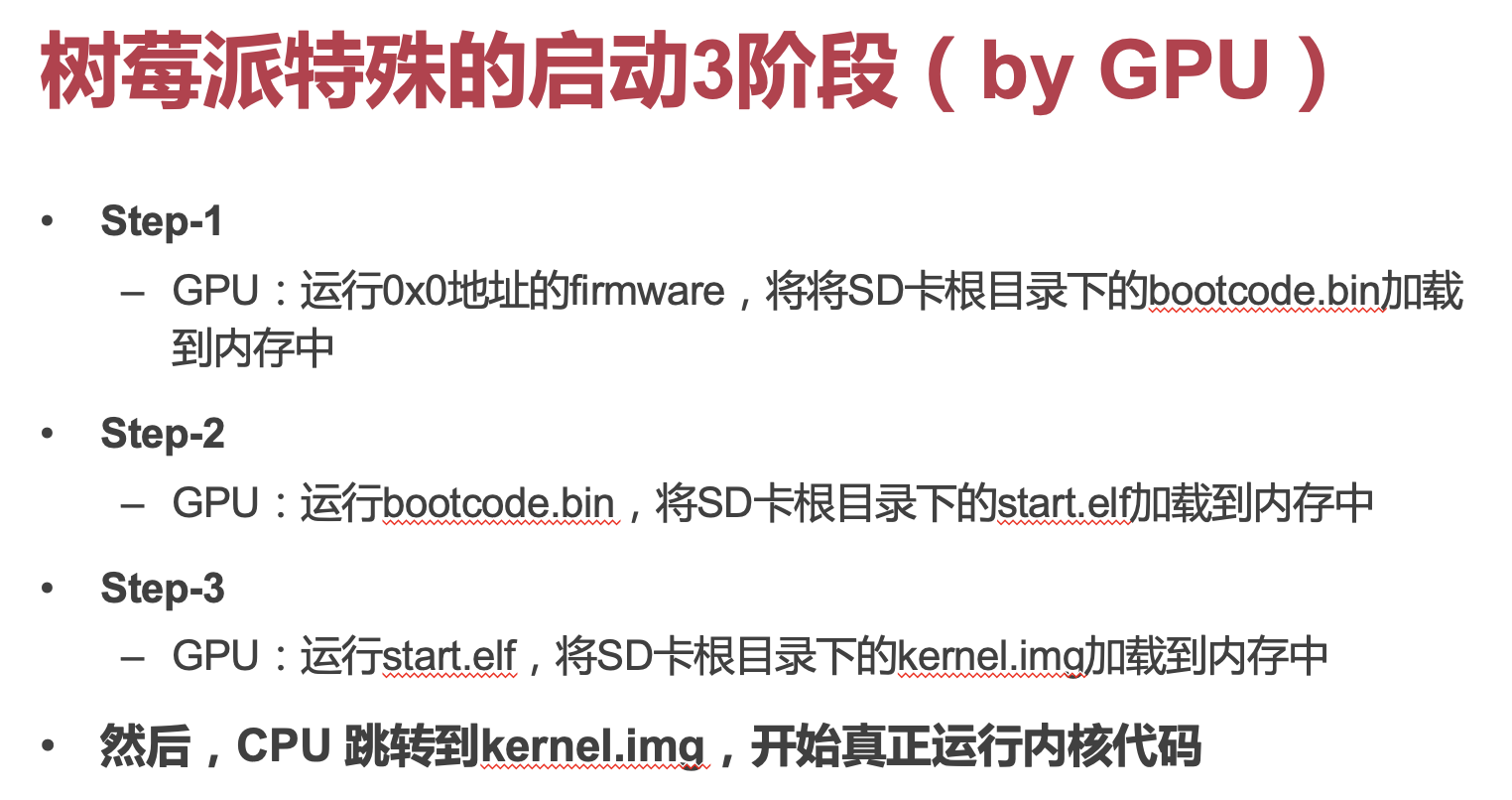
### 内核代码加载与运行



Q：why use GPU？

A：GPU是它生产的。





定义入口函数位置——CPU从预定义的RAM地址读取第一行代码，由硬件厂商决定。

boot和kernel目录：

boot目录，编译后放在.init段：.init：仅仅是做初始化，运行起来就可以释放这段内存。

kernel目录：编译后放在.text段，高地址范围。

设备内存：给外设一块物理地址空间。