新手看招：Linux系统启动时间的极限优化 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/284/2021_2022__E6_96_B0_E6_89_8B_E7_9C_8B_E6_c103_284479.htm （1）首先是对Linux启动过程的跟踪和分析，生成详细的启动时间报告。 较为简单可行的方式是通过PrintkTime功能为启动过程的所有内核信息增加时间戳，便于汇总分析。PrintkTime最早为CELF所提供的一个内核补丁，在后来的Kernel 2.6.11版本中正式纳入标准内核。所以大家可能在新版本的内核中直接启用该功能。如果你的Linux内核因为某些原因不能更新为2.6.11之后的版本，那么可以参考CELF提供的方法修改或直接下载它们提供的补丁：http://tree.celinuxforum.org/CelfPubWiki/PrintkTimes 开启PrintkTime功能的方法很简单，只需在内核启动参数中增加“ time ”即可。当然，你也可以选择在编译内核时直接指定“ Kernel hacking ”中的“ Show timing information on printks ”来强制每次启动均为内核信息增加时间戳。这一种方式还有另一个好处：你可以得到内核在解析启动参数前所有信息的时间。因此，我选择后一种方式。 当完成上述配置后，重新启动Linux，然后通过以下命令将内核启动信息输出到文件：dmesg -s 131072 > ktime 然后利用一个脚本“ show_delta ”（位于Linux源码的scripts文件夹下）将上述输出的文件转换为时间增量显示格式： /usr/src/linux-x.xx.xx/scripts/show_delta ktime > dtime 这样，你就得到了一份关于Linux启动时间消耗的详细报告。 （2）然后，我们就来通过这份报告，找出启动中相对耗时的过程。 必须明确一点：报告中的时间增量和内核信息之间没有必然的对应关系，真正的时间消耗必须从内核源

码入手分析。 这一点对于稍微熟悉编程的朋友来说都不难理解，因为时间增量只是两次调用printk之间的时间差值。通常来说，内核启动过程中在完成一些耗时的任务，如创建hash索引、probe硬件设备等操作后会通过printk将结果打印出来，这种情况下，时间增量往往反映的是信息对应过程的耗时；但有些时候，内核是在调用printk输出信息后才开始相应的过程，那么报告中内核信息相应过程的时间消耗对应的是其下一行的时间增量；还有一些时候，时间消耗在了两次内核信息输出之间的某个不确定的时段，这样时间增量可能就完全无法通过内核信息反应出来了。 所以，为了准确判断真正的时间消耗，我们需要结合内核源码进行分析。必要的时候，例如上述第三种情形下，还得自己在源码中插入printk打印，以进一步确定实际的时间消耗过程。 以下是我上次裁减后Linux内核的启动分析： 内核启动总时间： 6.188s 关键的耗时部分： 1) 0.652s - Timer,IRQ,Cache,Mem Pages等核心部分的初始化 2) 0.611s - 内核与RTC时钟同步 3) 0.328s - 计算Calibrating Delay（4个CPU核心的总消耗） 4) 0.144s - 校准APIC时钟 5) 0.312s - 校准Migration Cost 6) 3.520s - Intel E1000网卡初始化 下面，将针对上述各部分进行逐一分析和化解。