# 第四部分-做一个初步的性能评估

**原著：Apple Inc.**
**翻译：Gavin**

因此，你有一些代码，你想看看它是否有性能问题。你从哪里开始呢?并不是所有的问题都能立即显现出来。您可能会注意到一个操作需要几秒钟才能执行，但是您可能不会注意到一个操作消耗了太多CPU周期或分配了太多内存。这就是苹果的性能工具发挥作用的地方。它们可以帮助您查看程序中容易被忽略的方面。

下面几节将简要概述如何在开始分析程序时使用一些关键工具。这些工具可以很好地识别潜在的问题，并且可以提供大量的性能数据。但是，请记住，可能还有其他工具提供与问题相关的更具体的信息。使用其他工具运行应用程序可以帮助您确定某个特定区域是否存在问题。

**重要提示**:性能工具可以帮助您调查性能问题。确保在分析期间尽可能多地收集数据。性能分析有点像一门艺术，需要仔细考虑所有可用的数据来发现真正的问题。

有关性能工具的更多信息，包括从哪里获得它们，请参见性能工具。

## 1、使用top工具

top工具是识别过程中潜在问题区域的重要工具。这个工具会显示一个周期性的系统使用统计数据集。使用top并理解它的输出是识别潜在性能问题的好方法。

top工具定期显示CPU使用率、内存使用率(各种类别)、资源使用率(如线程和端口)和分页事件的更新统计信息。在默认模式下，top显示所有系统进程的CPU和内存利用率。您可以使用这些信息来查看您的程序使用了多少内存，使用了多少CPU时间。空闲程序不应该使用任何CPU时间，而活动程序应该根据任务的复杂性消耗相应的CPU时间。

**注意**:如果您希望跟踪CPU使用情况和其他统计数据，请使用工具中的活动监视器模板。活动监视图形的性能趋势，提供内存使用、CPU使用和其他数据的实时显示。

清单4-1显示了来自top的典型统计输出。对于应用程序开发人员，您应该最感兴趣的统计数据是CPU使用率、驻留的私有内存使用(RPRVT)和pagein/pageout速率。这些值告诉您应用程序资源使用的一些关键信息。高CPU使用率可能意味着应用程序的任务没有得到适当的调优。增加内存使用和页面输入/页面输出速率可能表明需要减少应用程序的内存占用。

**Listing 4-1** Typical output of top

```
Processes:  36 total, 2 running, 34 sleeping... 81 threads
Load Avg:  0.24, 0.27, 0.23     CPU usage:  12.5% user, 87.5% sys, 0.0% idle
SharedLibs: num =   77, resident = 10.6M code, 1.11M data, 4.75M LinkEdit
MemRegions: num = 1207, resident = 16.4M + 4.94M private, 22.2M shared
PhysMem:  16.0M wired, 25.8M active, 48.9M inactive, 90.7M used, 37.2M free
VM:  476M + 39.8M   6494(6494) pageins, 0(0) pageouts

  PID COMMAND      %CPU   TIME      #TH #PRTS #MREGS RPRVT  RSHRD  RSIZE  VSIZE
  318 top          0.0%  0:00.36   1    23    13   172K   232K   380K  1.31M
  316 zsh          0.0%  0:00.08   1    18    12   168K   516K   628K  1.67M
  315 Terminal     0.0%  0:02.25   4   112    50  1.32M  3.55M  4.88M  31.7M
  314 CPU Monito   0.0%  0:02.08   1    63    35   896K  1.34M  2.14M  27.9M
  313 Clock        0.0%  0:01.51   1    57    38  1.02M  2.01M  2.69M  29.0M
  312 Dock         0.0%  0:03.72   2    77    78  2.18M  2.28M  3.64M  30.0M
  311 Finder       0.0%  0:07.68   4    86   171  7.96M  9.15M  15.1M  52.1M
  308 pbs          0.0%  0:01.37   4    76    40   928K   684K  1.77M  15.4M
  285 loginwindow  0.0%  0:07.19   2    70    58  1.64M  1.93M  3.45M  29.6M
  282 cron         0.0%  0:00.00   1    11    14    88K   228K   116K  1.50M
  245 sshd         0.0%  0:02.48   1    10    15   176K   312K   356K  1.41M
  222 SecuritySe   0.0%  0:00.14   2    21    24   476K   828K  1.29M  3.95M
  209 automount    0.0%  0:00.03   2    13    20   336K   748K   324K  4.36M
  200 nfsiod       0.0%  0:00.00   1    10    12     4K   224K    52K  1.22M
  199 nfsiod       0.0%  0:00.00   1    10    12     4K   224K    52K  1.2
[...]
```

在其头区域，top显示系统全局状态的统计信息。此信息包括平均负载;工艺及螺纹总数;以及总内存，分为各种类型，如私有的、共享的、有线的和免费的。它还包括关于系统框架的全局信息。在定期的时候，top更新这些统计数据来解释最近的系统活动。

表4-1描述了使用-w参数在CPU和内存使用模式中出现的柱状数据。有关top如何报告信息的详细信息，请参阅top手册页面

**Table 4-1** Output from top using the -w option

| Column | Description |
|---|---|
| PID | The BSD process ID. |
| COMMAND | The name of the executable or application package. (Note that Code Fragment Manager applications are named after the native process that launches them, LaunchCFMApp.) |
| %CPU | The percentage of CPU cycles consumed during the interval on behalf of this process (both kernel and user space). |
| TIME | The amount of CPU time (*minute*:*seconds.hundredths*) consumed by this process since it was launched. |
| #TH | The number of threads owned by this process. |
| #PRTS (delta) | The number of Mach port objects owned by this process. (To display the delta value relative to the value first displayed when top was launched, use the -w parameter.) |
| #MREG | The number of memory regions. |
| VPRVT | The private address space currently allocated. (This value is displayed only with the -w parameter.) |
| RPRVT (delta) | The total amount of resident private memory. (To display the delta value relative to the previous sample, use the -w parameter when running top.) |
| RSHRD (delta) | The resident shared memory. (To display the delta value relative to the previous sample, use the -w parameter when running top.) |
| RSIZE (delta) | The total resident memory as real pages that this process currently has associated with it. Some may be shared by other processes. (To display the delta value relative to the previous sample, use the -w parameter when running top.) |
| VSIZE (delta) | The total address space currently allocated, including shared memory. (To display the delta value relative to the previous sample, use the -w parameter when running top.)<br>This value is mostly irrelevant for OS X processes. Every application has a large virtual size because of the shared region used to hold framework and library code. |

RPRVT数据(用于驻留的私有页面)可以很好地度量应用程序使用了多少实际内存。
RSHRD列(用于驻留共享页面)显示与其他进程共享的所有共享映射文件或内存对象的驻留页面。

**注意**:top工具不提供映射到进程的共享库中的页面数量的单独计数。
顶部工具报告“共享内存”类别中windows的内存使用情况，因为窗口缓冲区与窗口服务器共享。

表4-2显示了在事件计数模式中显示的列，该模式在命令行中使用-e、-d或-a选项启用。您可以使用这些选项来获得对应用程序特定行为的额外了解。例如，您可以将页面错误的数量与应用程序正在使用的内存数量相关联，以确定应用程序的内存占用是否太大。

**Table 4-2** Output from top using the -d option

| Column | Description |
|---|---|
| PID | The BSD process ID. |
| COMMAND | The name of the executable or application package. (Note that Code Fragment Manager applications are named after the native process that launches them, LaunchCFMApp.) |
| %CPU | The percentage of CPU cycles consumed during the interval on behalf of this process (both kernel and user space). |
| TIME | The amount of CPU time consumed by this process (*minute:seconds.hundredths*) since it was launched. |
| FAULTS | The total number of page faults. |
| PAGEINS | The number of page-ins, requests for pages from a pager (each page-in represents a 4 kilobyte I/O operation). |
| COW_FAULTS | The number of faults that caused a page to be copied (generally caused by copy-on-write faults). |
| MSGS_SENT | The number of Mach messages sent by the process. |
| MSGS_RCVD | The number of Mach messages received by the process. |
| BSDSYSCALL | The number of BSD system calls made by the process. |
| MACHSYSCALL | The number of Mach system calls made by the process. |
| CSWITCH | The number of context switches to the process (the number of times the process has been given time to run by the kernel's scheduler). |

# 2、使用Instruments

工具是一种非常强大的工具，可以用来收集性能数据并分析应用程序的总体行为。工具可以显示不同类型的性能信息，如内存和CPU使用量，并将它们并排显示出来。以这种方式查看信息可以更容易地识别趋势和看似不同的度量之间的关系。

当您第一次启动Instruments时，要求您为您的文档选择一个启动模板(图4-1)并选择要配置的应用程序。每个模板都预先配置了一个或多个用于为特定情况收集数据的工具。例如，泄漏模板包括分配工具和泄漏工具，让您看到分配的内存块的总数和那些被认为是泄漏的内存块的子集。您可以随时向文档添加更多的工具，但是模板提供的公共配置通常足以完成典型的任务。

**Figure 4-1** Choosing an Instruments template

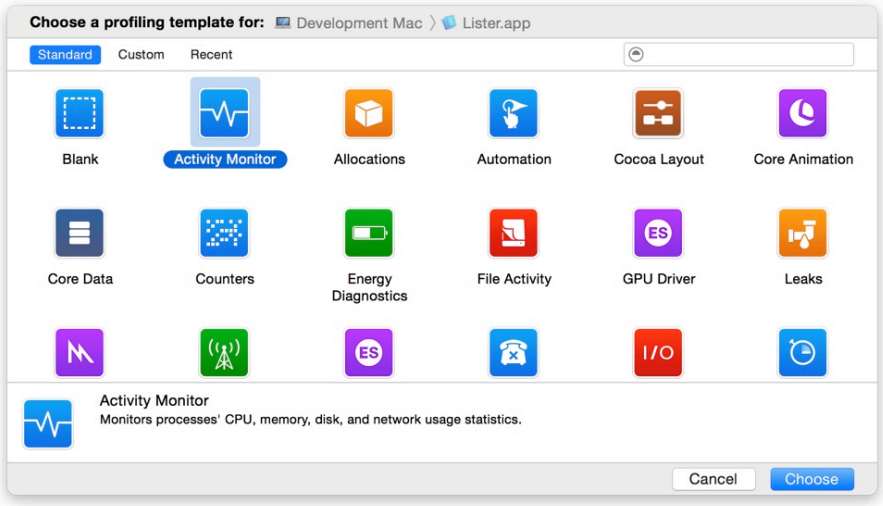
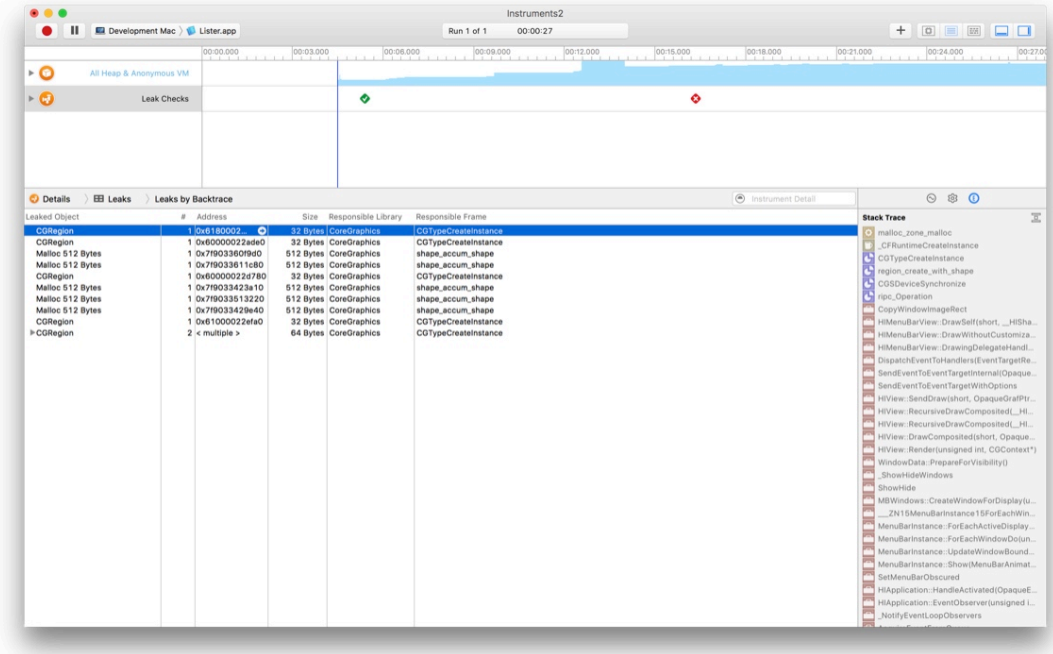


图4-2显示了泄漏模板在记录运行后收集的数据示例。时间轴和细节窗格是可配置的和可过滤的，因此您可以查看对您最感兴趣的数据。

**Figure 4-2** Examining the recorded data

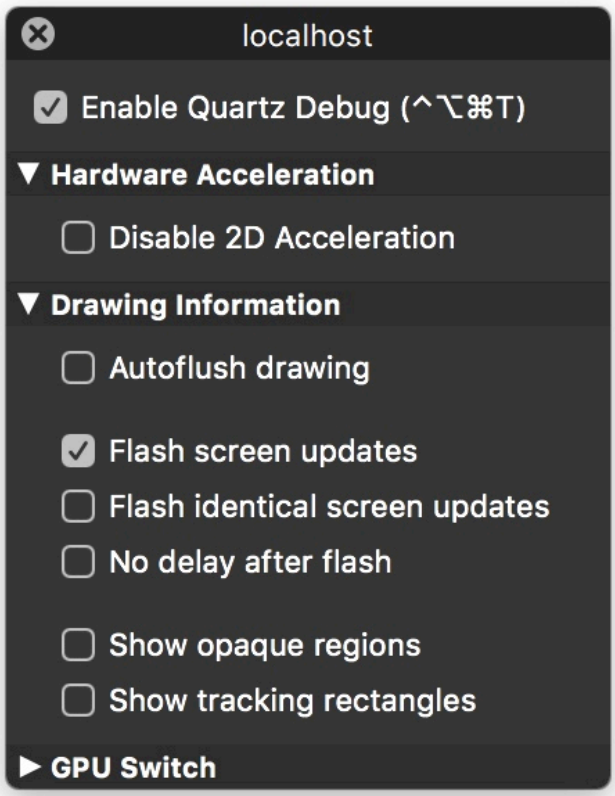



有关如何使用Instruments的详细信息以及关于可以收集的性能数据类型的信息，请参阅Instruments用户指南。

## 3、使用Quartz调试

Quartz Debug是确定绘图代码的效率的重要工具。该工具从程序的绘图调用中收集信息，以找出程序正在绘图的位置，以及是否在不必要地重新绘制内容。图4-3显示了Quartz Debug选项窗口。

**Figure 4-3** Quartz Debug options




启用了Flash屏幕更新选项后，Quartz Debug会直观地显示代码所在的位置。它在重新绘制操作即将发生的区域上放置一个黄色矩形，然后在重新绘制内容之前短暂地暂停。这种闪烁的黄色图案可以指出你所绘制的图片的位置。例如，如果您只更新自定义视图的一小部分，您可能不希望被迫重新绘制整个视图。或者，如果您看到一个系统控件连续多次被重绘，它可能会指出在更改其属性之前需要隐藏该控件。

“Flash完全相同的屏幕更新”选项会在结果与当前内容相同的任何区域显示红色。您可以使用此选项来检测代码中的冗余绘图操作。