Oracle Solaris Studio 12.2 DLight 教程

2010 年 9 月

简介

DLight 是一种基于 Oracle Solaris 动态跟踪 (DTrace) 技术的、面向 C/C++ 开发人员的交互式图形观察工具。您可以在 Solaris 平台上使用 DLight 以同步方式分析多个 DTrace 脚本中的数据，从而追踪到应用程序中的运行时问题的根本原因。

您必须在运行 DLight 的系统上具有超级用户权限。如果启动 DLight 时未以超级用户身份登录，当您第一次运行程序时会提示您输入超级用户口令。

DLight 包含五个针对 C、C++ 和 Fortran 程序的配置工具：

- 线程微观状态
- CPU 使用情况
- 内存使用情况
- 线程使用情况
- I/O 使用情况

DLight 还包含三个针对 AMP（Apache、MySQL、PHP）栈中进程的配置工具：

- Apache 监视器
- MySQL 监视器
- PHP 监视器

当您运行可执行目标或附加到目标进程时，每个工具都会在 "Run Monitor"（运行监视器）窗口中的一个图形中显示使用情况信息。每个图形都包含一个按钮，单击该按钮可获取更多信息。窗口底部的指示器控件可用于控制图形的视图。

生成样例应用程序

在已安装的 Oracle Solaris Studio 12.2 中的 `examples/dlight/ProfilingDemo` 目录下存在 `ProfilingDemo` 应用程序。

1. 将 ProfilingDemo 目录的内容复制到您自己的专用工作区：

   ```
   cp -r installation_directory/examples/dlight/ProfilingDemo ~/ProfilingDemo
   ```

2. 生成自己的程序副本：

   ```
   cd ~/ProfilingDemo
   make
   ```

   该程序是使用 -g 选项生成的，该选项会生成调试信息。（如果编译时未使用此选项，DLight 可能会收集并显示该程序的运行时数据，但用于显示函数源代码的功能将不起作用。）

启动 DLight

如果尚未运行 DLight，请通过键入以下内容将其启动：

installation_directory`/bin/dlight`

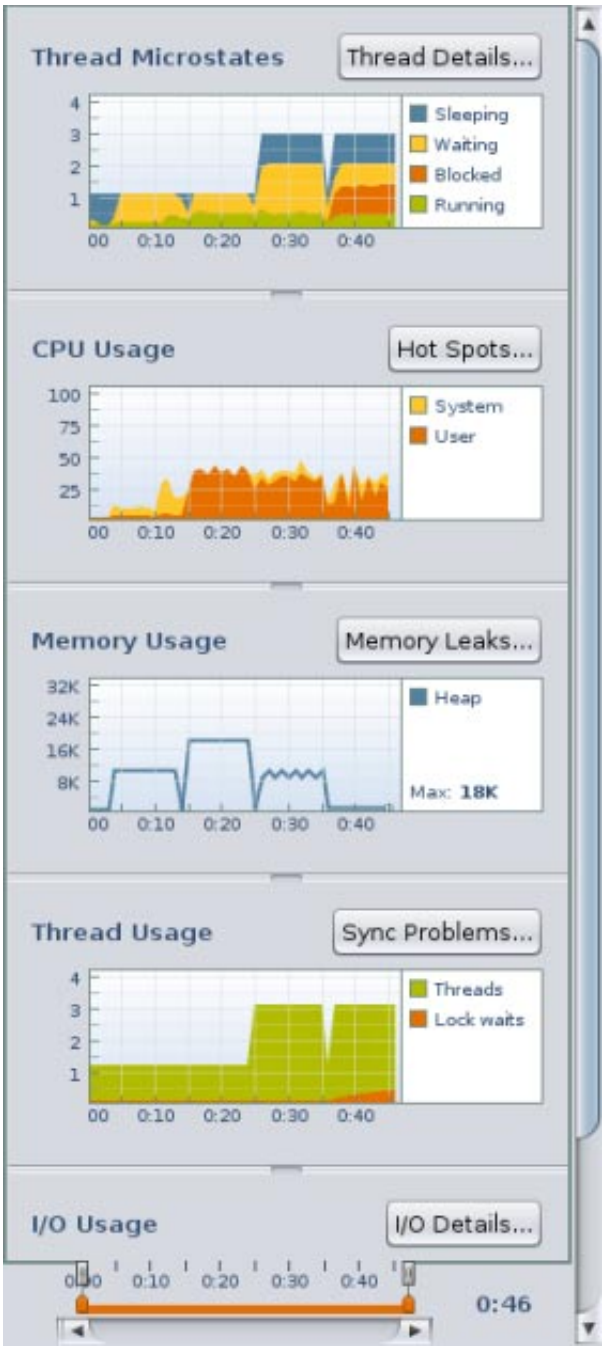
运行样例应用程序

1. 在 DLight 中，单击 "New DLight Target"（新建 DLight 目标）按钮  。在目标对话框中：
   a. 单击 "Executable Target"（可执行目标）选项卡。
   b. 在 "Run"（运行）字段中键入 `profilingdemo` 可执行文件的路径名，或者单击 "Browse"（浏览）以使用 "Open"（打开）对话框浏览到 `profilingdemo` 可执行文件。
   c. 样例应用程序不使用任何参数，因此请将 "Arguments"（参数）字段保留为空白。
   d. 仅当您想要跟踪可执行文件的某个子进程时才使用 "Trace"（跟踪）字段，因此请将该字段也保留为空白。
   e. 单击 "Run"（运行）。
   f. 如果启动 DLight 时未以超级用户身份登录，系统会提示您输入超级用户口令。
2. `ProfilingDemo` 应用程序开始执行，且 "Run Monitor"（运行监视器）窗口开始显示动态图形。在 "Output"（输出）窗口中，`ProfilingDemo` 程序会告知您正在执行的操作，这样您可以将输出与图形中显示的数据相匹配。
3. 每次程序请求该操作时请按 Enter 键，直到完成执行。

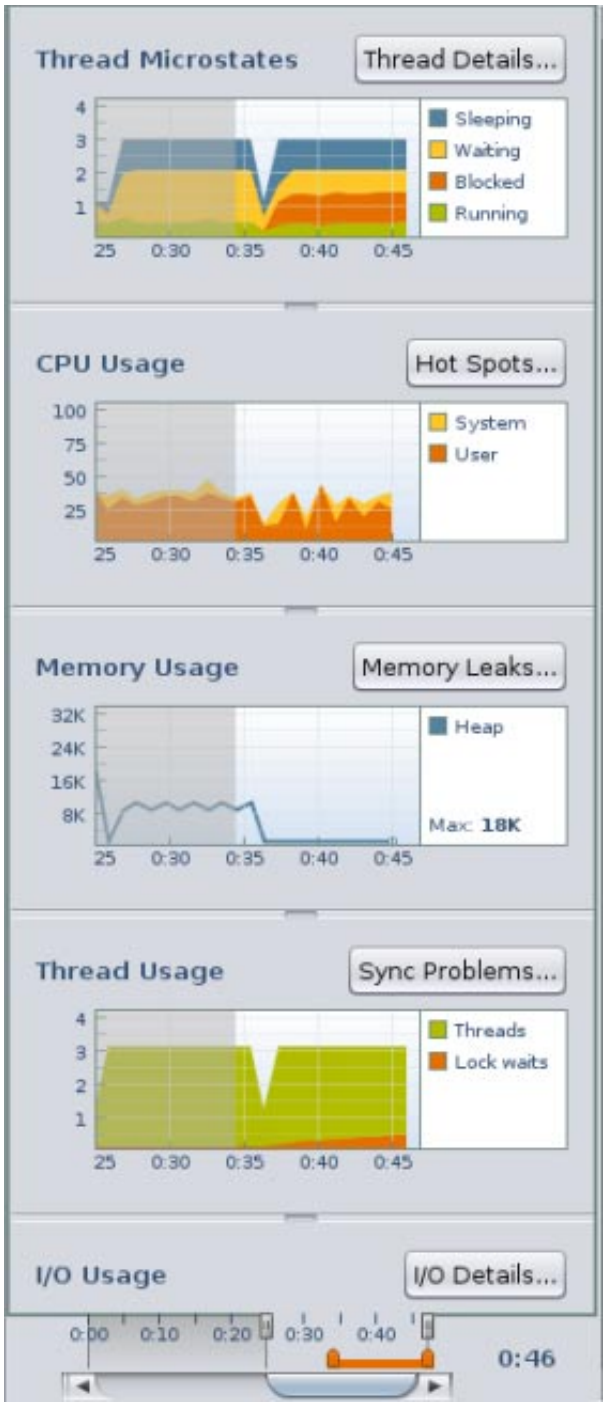
使用指示器控件

1. 在 "Run Monitor"（运行监视器）窗口的底部，可以看到用于控制图形视图的滑块："View"（视图）滑块、"Details"（详细信息）滑块和 "Time"（时间）滑块。将鼠标光标置于滑块端点的上方可获取有关滑块的信息。

2. 在 "Time"（时间）滑块上单击并按住鼠标光标，然后将滑块拖到左侧可看到运行的开始处。所有图形都会一起滑动，从而可以在给定时间看到每个方面（CPU、内存、线程、I/O）所发生的情况，并看到它们之间的关系。
3. 从左向右拖动 "Time"（时间）滑块可看到完整的运行。
4. 将鼠标光标移动到 "View"（视图）滑块，该控件覆盖了时间单元。使用 "View"（视图）滑块控件可以选择在图形中显示运行时的哪一部分。
5. 单击并拖动 "View"（视图）滑块的左手柄（起点），一直拖到运行的开始处。这些图形现在会立即显示整个运行。效果与尽可能缩小很相似。请注意，如果选择完整运行时，"Time"（时间）滑块将不起作用。您现在已经看到了所有数据，因此没有必要再进行滚动。



6. 现在让我们近距离地观察一下。向右拖动 "View"（视图）滑块的起点。当拖动手柄时，这些图形会放大，以聚焦到靠近运行结束处的区域。请注意，可以再次使用 "Time"（时间）滑块在运行时中来回滚动。
7. 将鼠标光标置于橙色 "Details"（详细信息）滑块的端点上方可获取关于如何使用该滑块的说明。使用 "Details"（详细信息）滑块控件可以选择运行时的某一部分进行详细检查。
8. 将 "Details"（详细信息）滑块的起点拖动到比 "View"（视图）滑块的起点稍晚的一个时间点。请注意，在起点前面的区域中，图形是灰显的，从而为起点和终点之间的图形提供了一种突出显示效果。



9. 单击图形的任意一个详细信息按钮（"Thread Details"（线程详细信息）、"Hot Spots"（热点）、"Memory Leaks"（内存泄漏）、"Sync Problems"（同步问题）或 "I/O Details"（I/O 详细信息））时，详细信息选项卡中会显示突出显示区域的数据。
10. 将 "View"（视图）滑块的起点拖回到运行的开始处，这样可以看到所有数据。

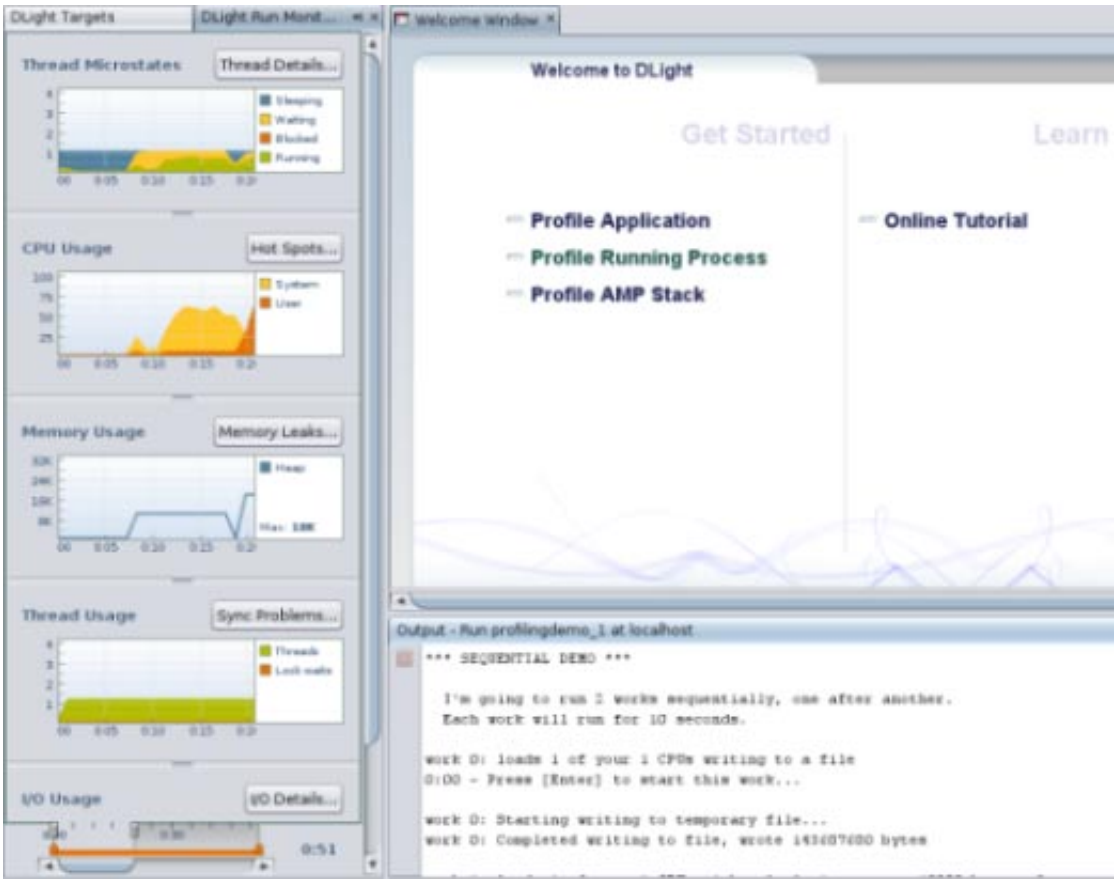
了解线程微观状态

在程序运行期间，随着程序线程进入各种执行状态，"Thread Microstates"（线程微观状态）图会显示程序线程的概览。Solaris 微观状态计数功能使用 DTrace 工具来提供进入和退出十种不同的执行状态时有关每个线程的状态的细粒度信息：

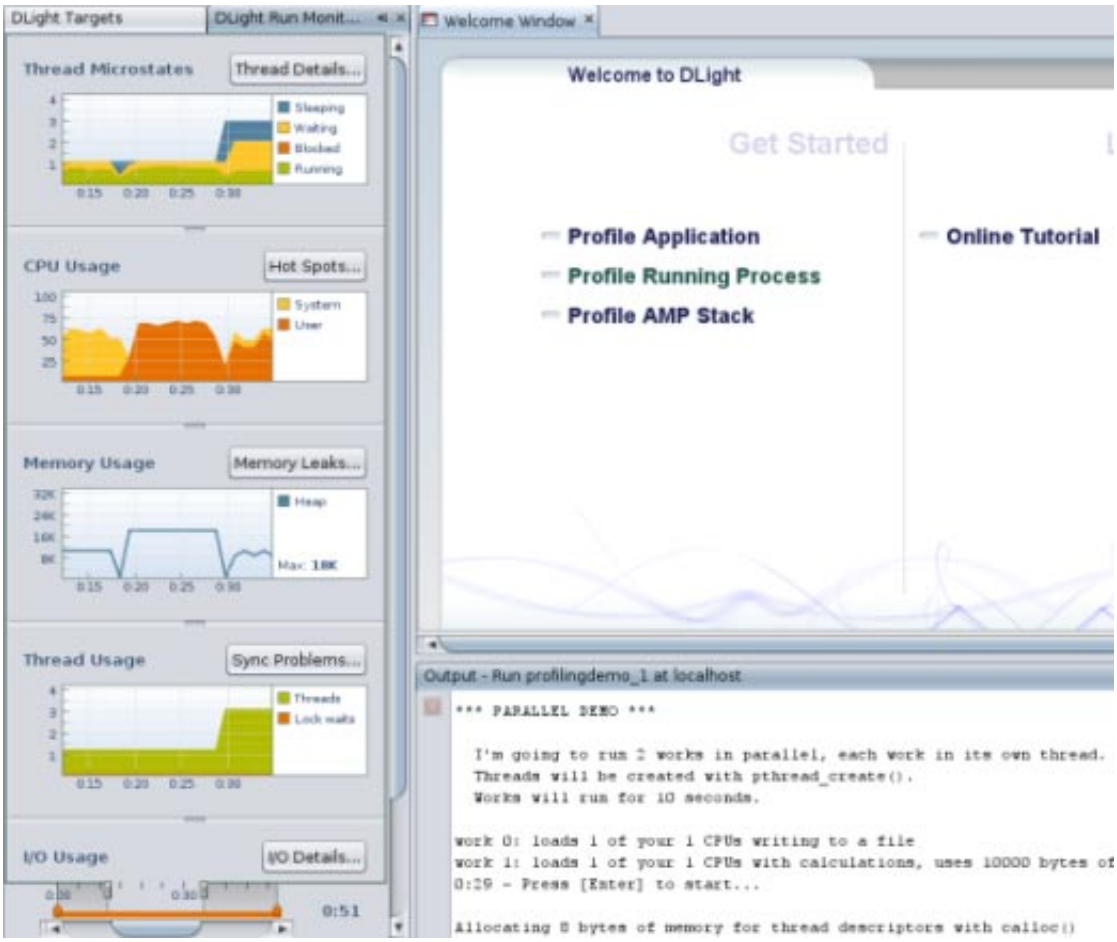
User Running（用户运行）	进程在用户模式下花费的时间百分比
System Running（系统运行）	进程在系统模式下花费的时间百分比
Other running（其他运行）	进程处理系统陷阱等所花费的时间百分比
Text page fault（文本缺页）	进程处理文本缺页所花费的时间百分比
Data page fault（数据缺页）	进程处理数据缺页所花费的时间百分比
Blocked（阻塞）	进程等待用户锁定所花费的时间百分比
Sleeping（休眠）	进程休眠所花费的时间百分比
Waiting（等待）	进程等待 CPU 所花费的时间百分比

"Thread Microstates"（线程微观状态）工具以图形方式显示在程序运行期间创建的所有线程的状态概括信息。仅显示四种状态："Sleeping"（休眠）、"Waiting"（等待）、"Blocked"（阻塞）和"Running"（运行）。这些状态提供了十个可能的微观状态的简化或摘要视图，并提供了程序中正在运行的所有线程的状态概述。例如，在"Running"（运行）状态中花费的时间代表正在运行的状态的所有类型：在用户模式下运行、在系统调用下运行、在缺页中运行以及在陷阱中运行。

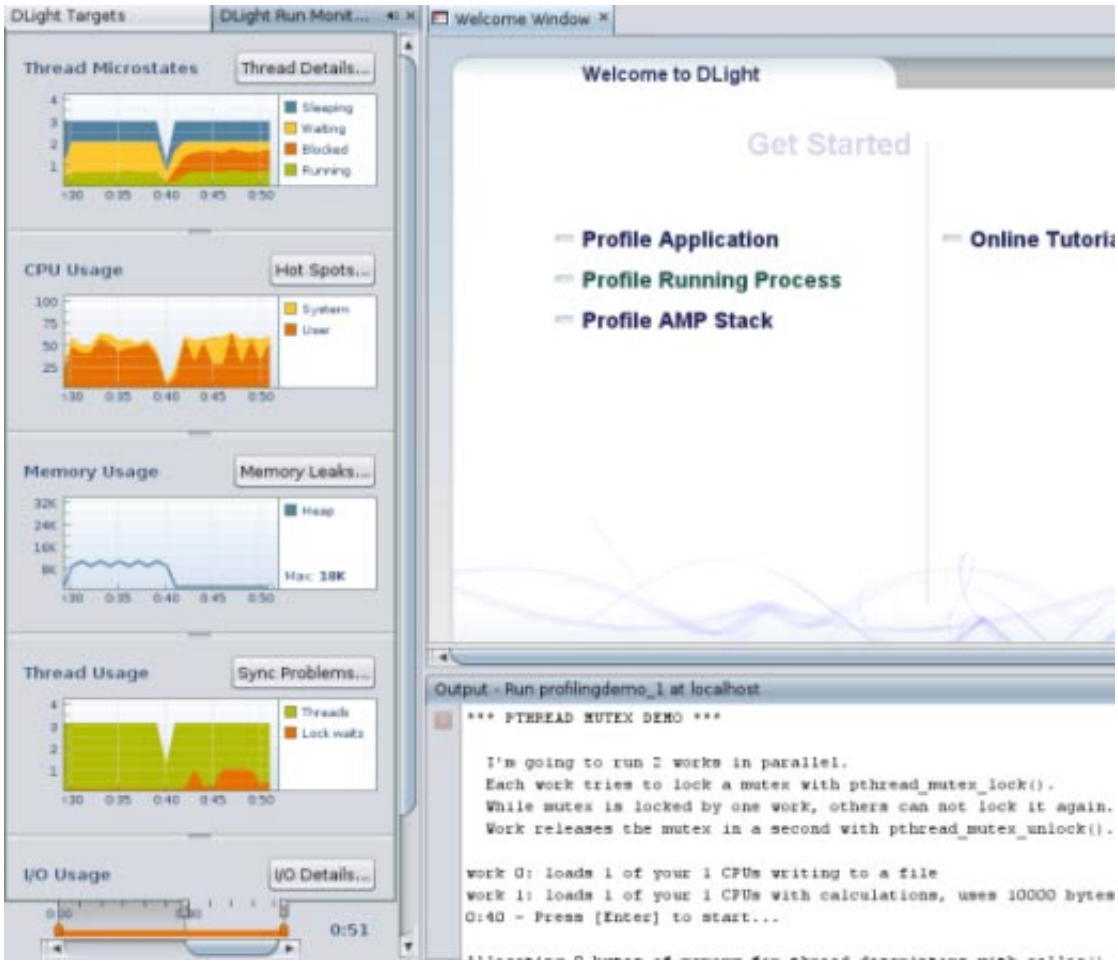
1. 将"View"（视图）滑块的左手柄向左移动，直到图形显示大约 20 秒的运行时，如下所示。此图像显示了在单一线程中相继运行两个任务时 `SEQUENTIAL DEMO` 部分期间的程序运行的开始处。线程休眠所处的点与程序等待用户按 Enter 键所处的点相对应。



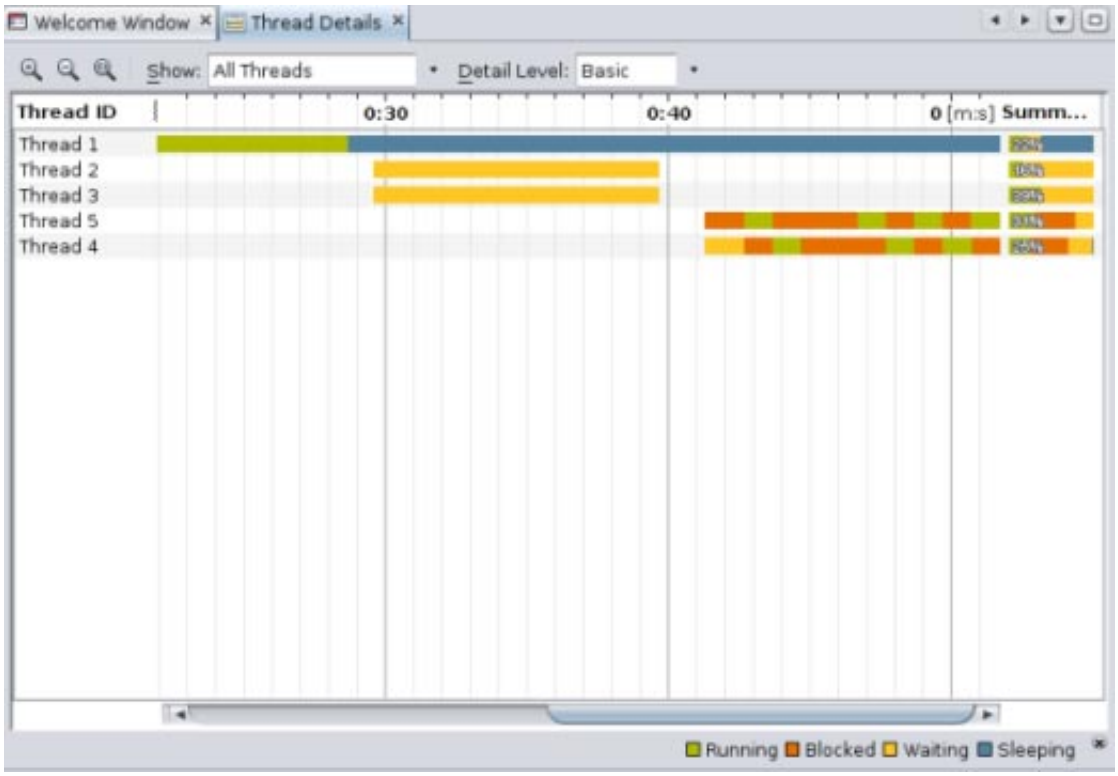
2. 单击 "Time"（时间）滑块并一直向右拖动，直到在 "Thread Microstates"（线程微观状态）工具中显示的线程数跳到三为止。
3. 当程序进入 `PARALLEL DEMO` 部分时，线程数会跳到三。主线程会启动两个其他线程以并行方式运行两个任务，每个任务都在各自的线程中运行。您可以看到在 "Waiting"（等待）状态（黄色）和 "Sleeping"（休眠）状态（蓝色）下花费了相当多的时间，而在 "Running"（运行）状态（绿色）下花费的时间则要少一些。在 `PARALLEL DEMO` 部分期间，不会在 "Blocked"（阻塞）状态（橙色）下花费任何时间，因为程序的这一部分不会执行任何线程同步策略，例如会阻塞线程的互斥锁。



4. 将 "Time"（时间）滑块一直拖到运行时的结束处。请注意橙色的 "Blocked"（阻塞）微观状态显示在程序进入 PTHREAD MUTEX DEMO 部分的点处，在该部分中，每个线程都使用互斥锁来防止其他线程在某些点处受到干扰。每个线程仅在获取互斥锁之后才可以主动运行。如果某个线程拥有互斥锁，则其他线程在尝试访问代码的锁定部分时会被阻止。当线程对同一数据进行重叠访问时，使用互斥锁可防止线程出现数据争用情况。

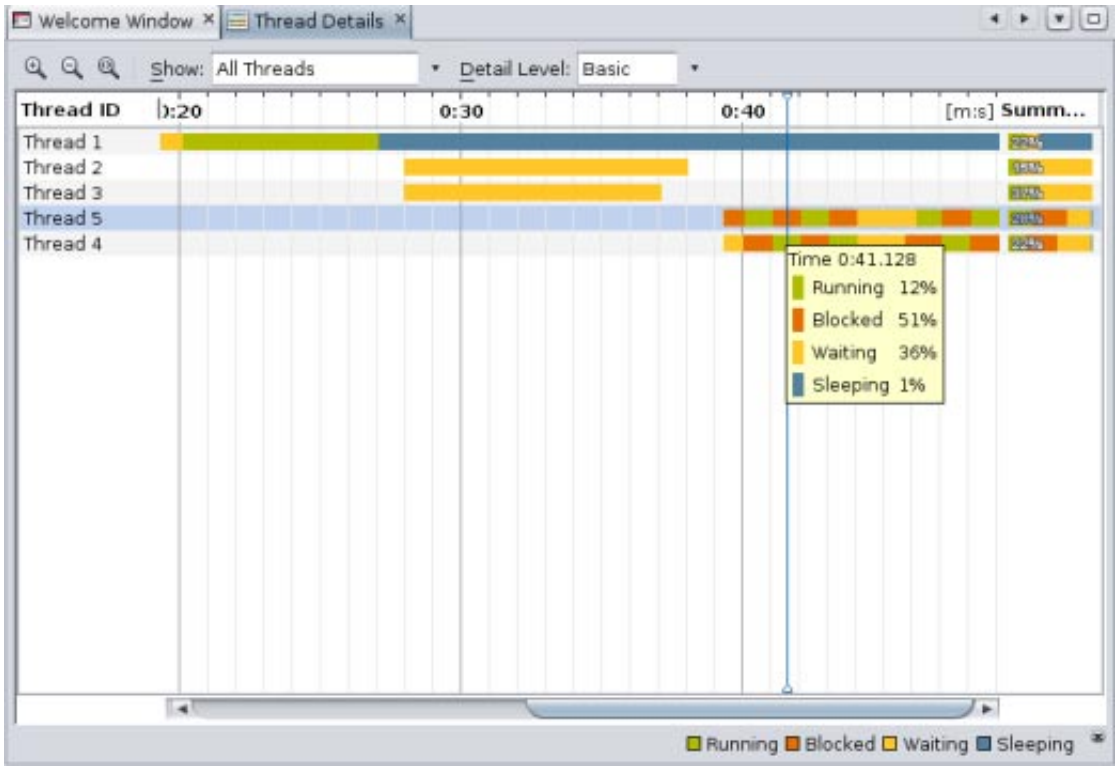


5. 单击 "Thread Details"（线程详细信息）按钮以显示有关线程微观状态的详细信息。"Thread Details"（线程详细信息）窗口将打开，显示在程序中运行的所有线程的图形时间线表示以及详细的状态信息。

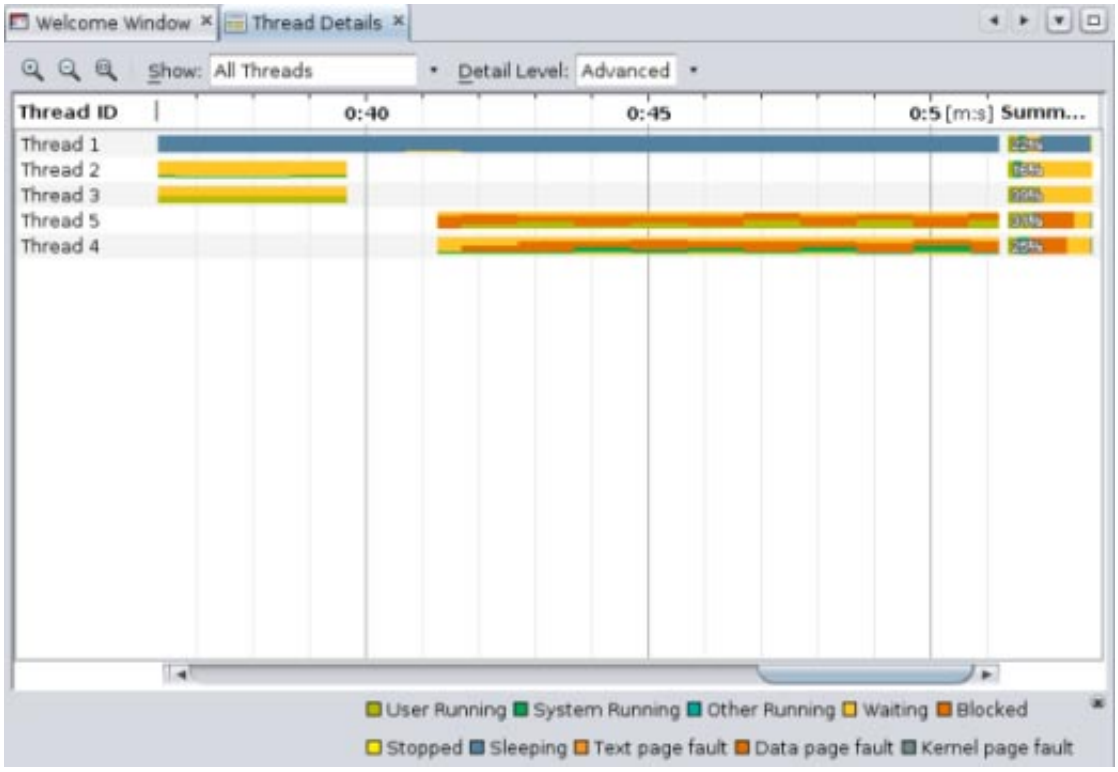


"Thread Details"（线程详细信息）窗口显示在程序的完整运行时期间每个线程的状态转换。

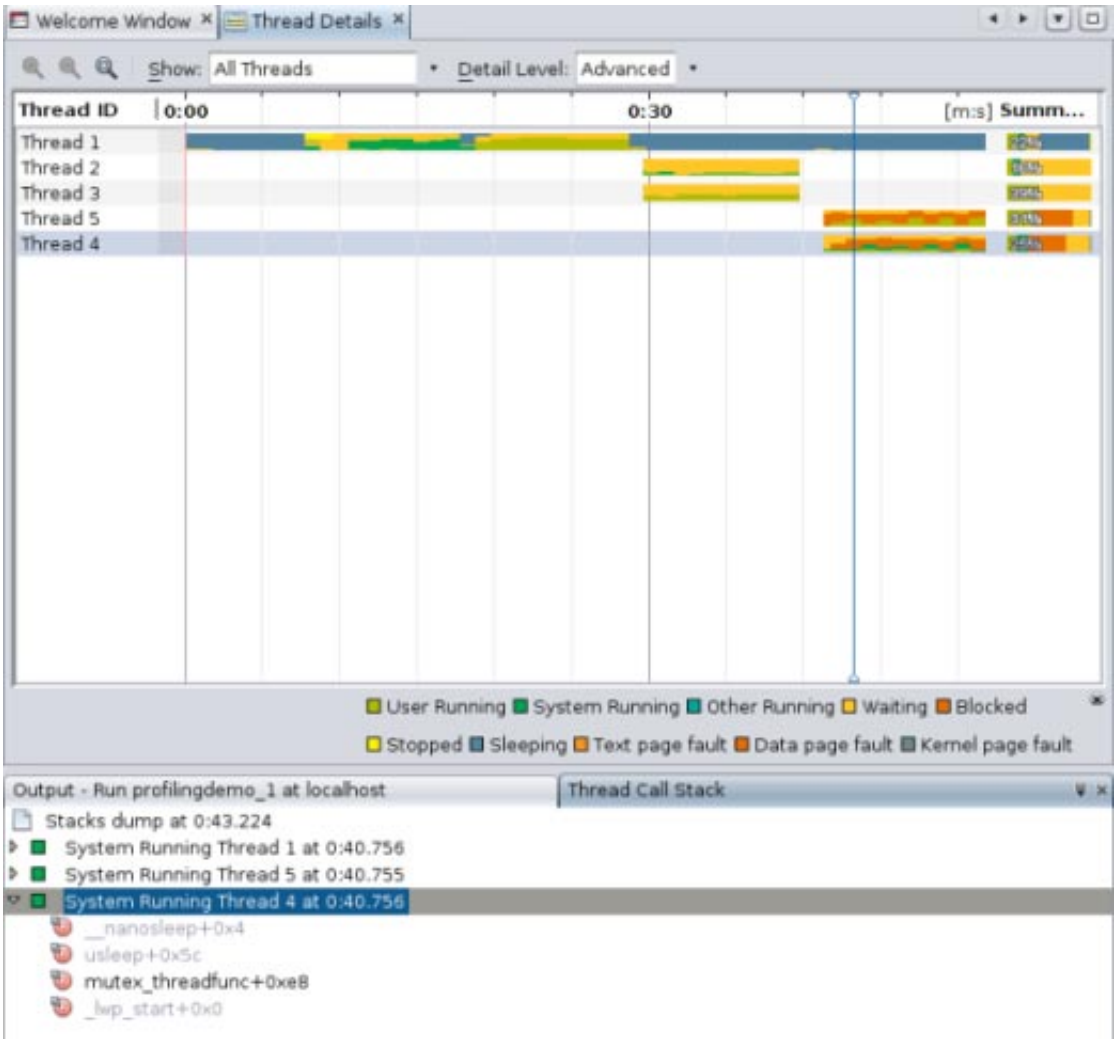
6. 将光标放置到线程的其中一个颜色区域上。将出现一个弹出窗口，显示有关在该特定时刻该线程将要发生的变化的详细信息。"Details"（详细信息）包括数据的收集时间以及该时刻在每个线程状态下花费的时间百分比。将光标放置到窗口右侧的 "Summary"（摘要）区域上方时，将出现一个弹出窗口，显示线程的完整运行在每个状态下的百分比。



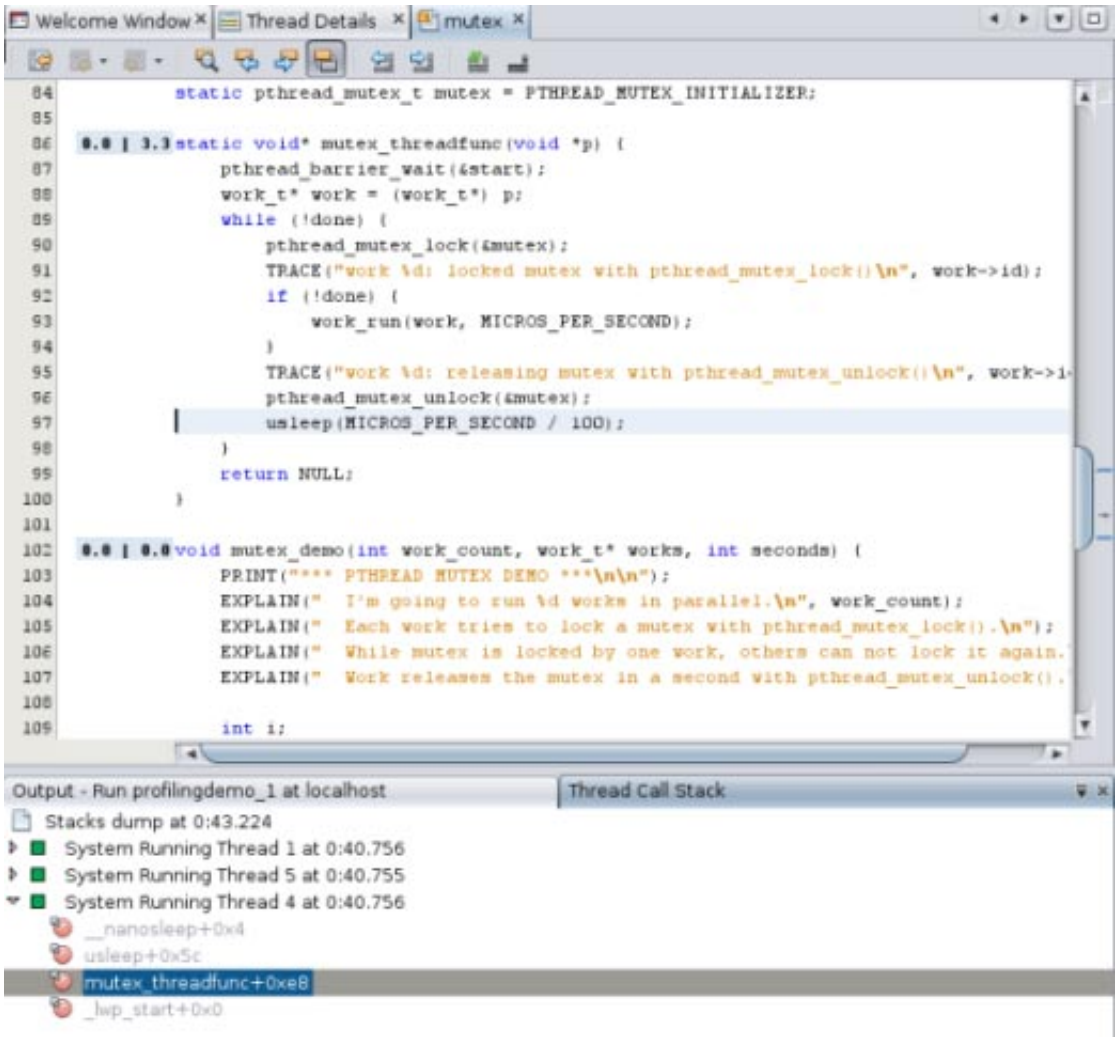
7. 尝试使用窗口的控件来更改显示的内容：
   - 缺省情况下，该窗口会显示所有线程。单击 "Show"（显示）下拉式列表右侧的向下箭头。您可以选择 "Live Threads only"（仅实时线程）仅显示尚未终止的线程，或者选择 "Finished Threads only"（仅已完成的线程）仅显示在程序运行期间已终止的线程。
   - 缺省情况下，该窗口仅显示四种概括的执行状态。单击 "Detail Level"（详细信息级）下拉式列表右侧的向下箭头。您可以选择 "Moderate"（中等）显示有关这些状态的更多详细信息，或者选择 "Advanced"（高级）显示十种微观状态。
   - 单击某个单独的线程，请注意该线程将突出显示。如果按住 Shift 键单击另一个线程，会选中某个范围的线程。要选择不相邻的多个线程，请在选择线程之前按 Ctrl 键。如果感兴趣的线程已突出显示，请右键单击并选择 "Show Only Selected Threads"（仅显示选定线程）。要再次看到所有线程，请从 "Show"（显示）下拉式列表中选择 "All Threads"（所有线程）。
8. 通过单击 "Zoom In"（放大）按钮，放大线程图形以便更仔细地查看。  。此图像显示了放大后的窗口，且 "Detail Level"（详细信息级别）设置为 "Advanced"（高级）。



9. 单击 "Zoom Out"（缩小）按钮  可返回到前一个缩放级别。
10. 单击 "Show Complete Run"（显示完整运行）按钮  会立即在 "Thread Details"（线程详细信息）窗口中显示完整运行。（可以再次单击该按钮  返回到线程详细信息的滚动视图。）
11. 单击线程 4 上的第二个橙色矩形。"Thread Call Stack"（线程调用栈）选项卡将打开，显示线程在此刻的调用栈。您可以展开该栈中的节点查看该线程中发生的调用，或者右键单击顶部节点并选择 "Expand All"（全部展开）查看所有线程中的调用。



12. 双击 `mutex_threadfunc` 函数可打开调用该函数的源文件。（您可以在未灰显的栈中显示任意函数的调用源文件。）

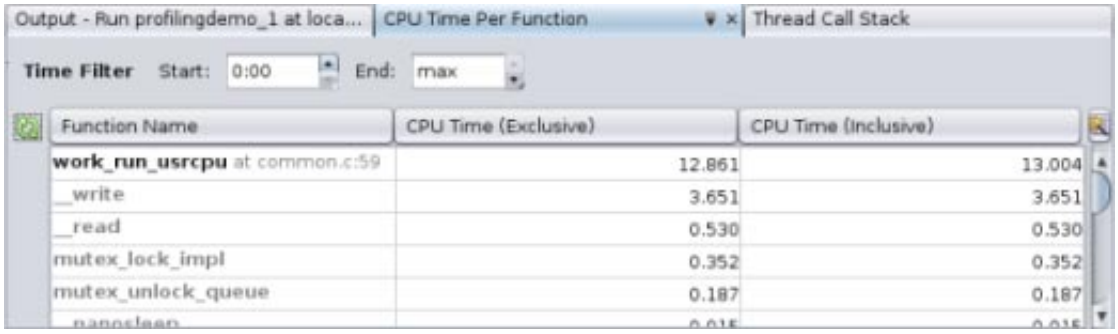


13. 单击 "Thread Details"（线程详细信息）选项卡返回到 "Thread Details"（线程详细信息）窗口。单击某个线程。您可以使用鼠标或键盘沿着线程的时间线导航。
    - 使用鼠标时，右键单击线程并使用 "Navigate"（导航）菜单向左或向右移动焦点，在时间线上确定一个点并更新 "Thread Call Stack"（线程调用栈）选项卡的内容，或者将焦点切换到 "Thread Call Stack"（线程调用栈）选项卡。
    - 要使用快捷键导航线程，请按下列键：
        - Ctrl + 向左方向键和 Ctrl + 向右方向键，用于在线程时间线上向左和向右滚动
        - Ctrl + 向下方向键，用于选择时间线上的某个点，此操作将更新 "Thread Call Stack"（线程调用栈）
        - Alt + 向下方向键，用于在 "Thread Call Stack"（线程调用栈）窗口上定位输入焦点
    - 在 "Thread Call Stack"（线程调用栈）窗口中，可以使用方向键和 Enter 键打开与函数相关联的源文件

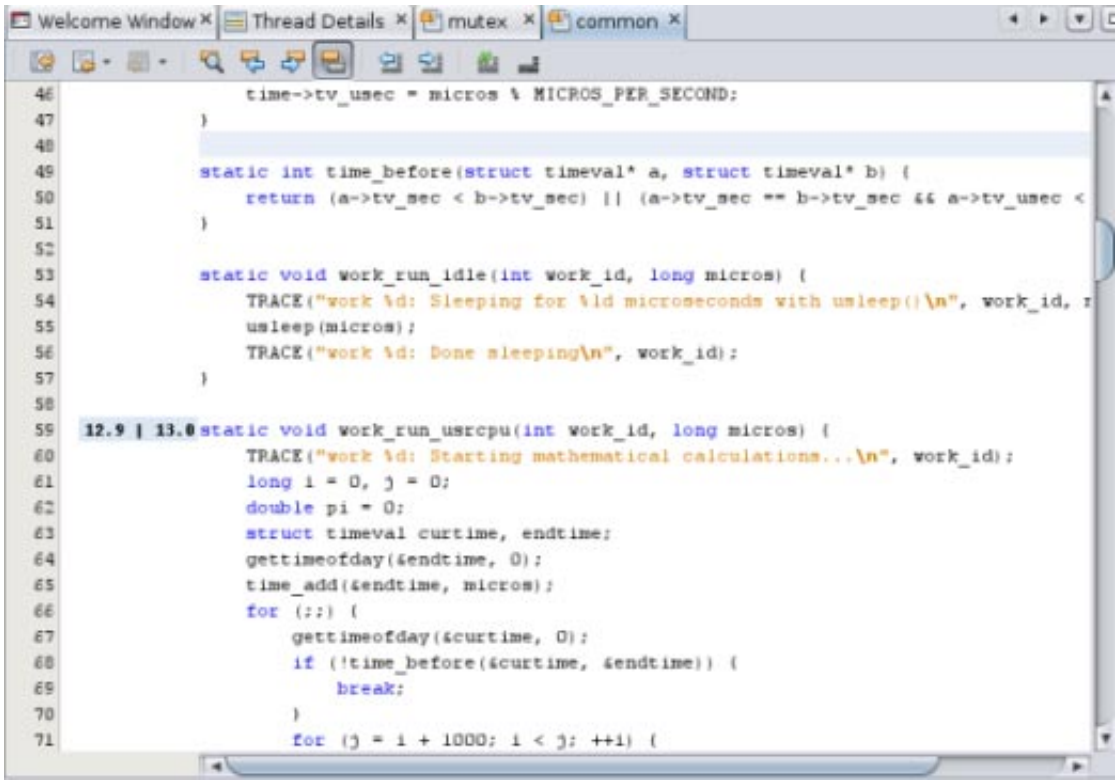
了解 CPU 使用情况

"CPU Usage"（CPU 使用情况）图显示了应用程序在其运行期间所使用的 CPU 总时间的百分比。

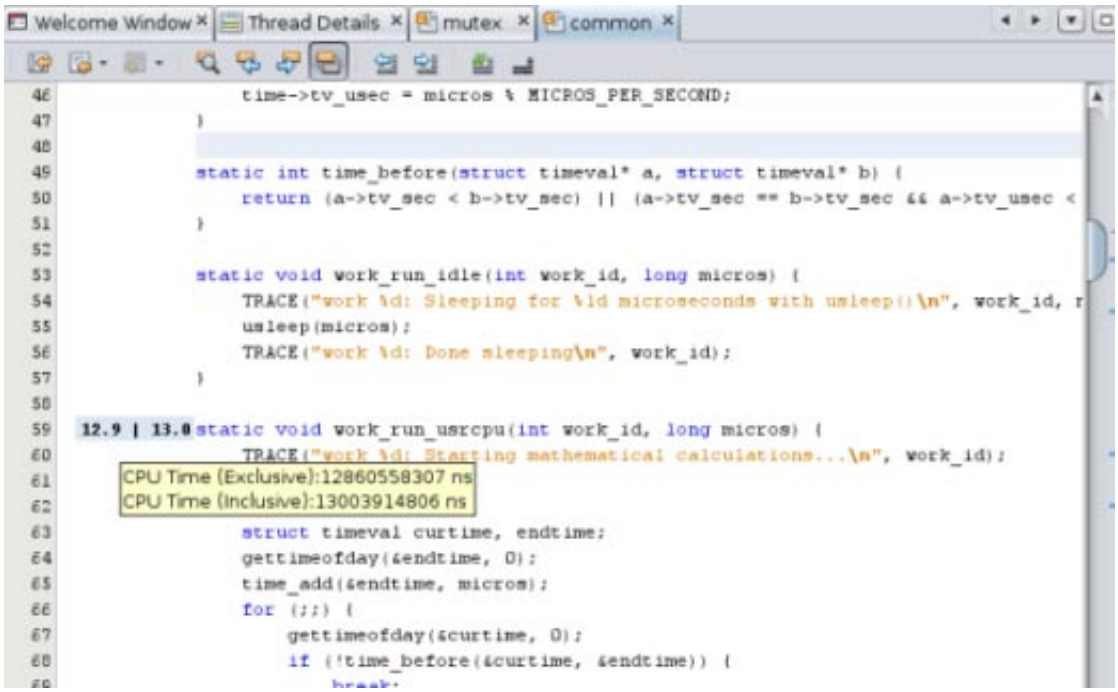
1. 单击 "Hot Spots"（热点）按钮可显示有关 CPU 时间的详细信息。"CPU Time Per Function"（每个函数的 CPU 时间）选项卡会打开，显示程序的各个函数以及每个函数所使用的 CPU 时间。这些函数按所使用的 CPU 时间的顺序列出，使用最长时间的函数列在第一个。如果程序仍在运行，最先显示的时间就是在单击按钮那一刻所消耗的时间量。



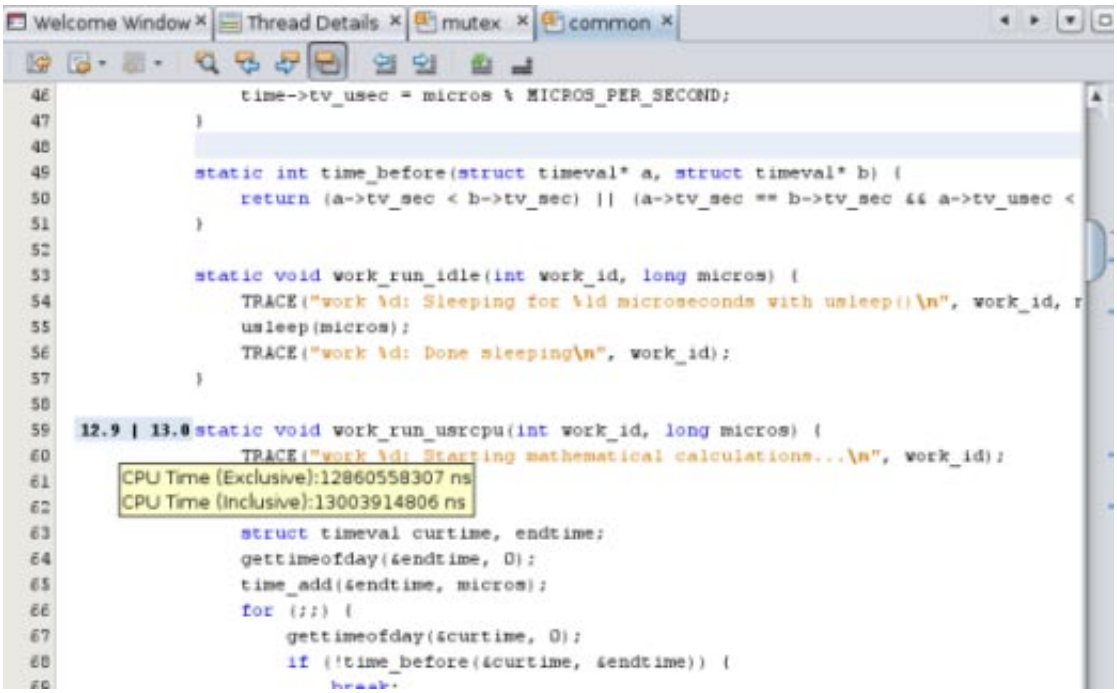
2. 单击 "Function Name"（函数名称）列的标题可按字母顺序对函数进行排序。
3. 单击 "CPU Time (Exclusive)"（CPU 时间（独占））列可按各个函数所使用时间的顺序对函数进行排序。
4. 请注意两列 "CPU Time"（CPU 时间）之间的差异。"CPU Time (Inclusive)"（CPU 时间（包含））显示从进入函数的时间起直到函数退出的时间为止所花费的 CPU 总时间，包括所列函数调用的所有其他函数的时间。"CPU Time (Exclusive)"（CPU 时间（独占））仅显示特定函数所使用的时间，而不包括它所调用的任何函数。
5. 单击 "CPU Time (Inclusive)"（CPU 时间（包含））列标题可将最消耗时间的函数放回到顶部。请注意，`work_run_usrcpu` 函数使用的 "CPU Time (Inclusive)"（CPU 时间（包含））仅略多于 "CPU Time (Exclusive)"（CPU 时间（独占）），这意味着其 CPU 时间中实际上只有很少一部分是由它所调用的其他函数使用的，而 `work_run_usrcpu` 函数本身使用了大部分时间。
6. 某些函数以粗体显示。您可以显示这些函数的源文件。双击 `work_run_usrcpu` 函数。`common.c` 文件将打开，光标停留在第 59 行的 `work_run_usrcpu` 函数上。该行的左边界中会显示一些数字。



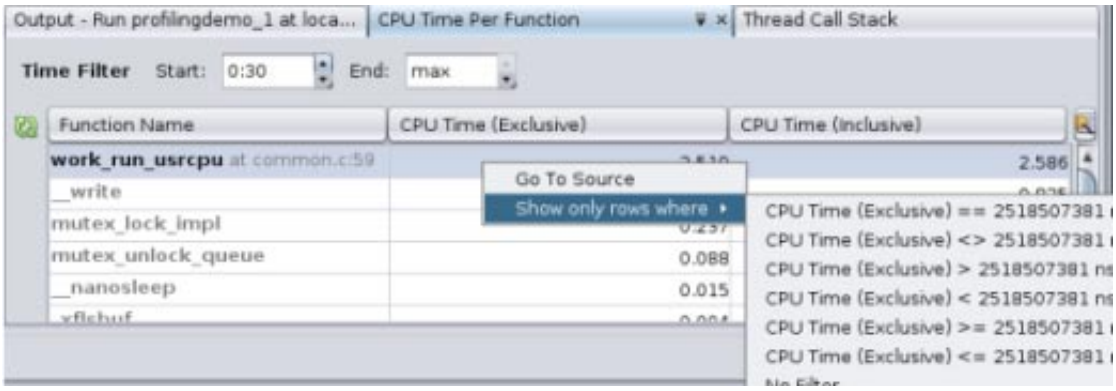
7. 将鼠标光标放置到左边界中的数字上方。这些数字与 "CPU Time Per Function"（每个函数的 CPU 时间）选项卡中所显示的函数的 CPU 独占时间和 CPU 包含时间的度量相同。这些度量为了节省空间而进行了四舍五入，但将光标置于其上方时会显示未舍入的值。`common.c` 源文件中也会显示 CPU 消耗行的度量（例如在 `work_run_usrcpu` 函数内执行计算的循环）。



8. 通过键入时间并按 Enter 键，或者使用箭头滚动秒中，将 "CPU Time Per Function"（每个函数的 CPU 时间）选项卡中的时间过滤器开始时间更改为 0:30。"Run Monitor"（运行监视器）窗口中的图形将随之变化，如同移动 "Details"（详细信息）滑块上的手柄时一样。如果拖动手柄，将更新 "CPU Time Per Function"（每个函数的 CPU 时间）选项卡中的 "Time Filter"（时间过滤器）设置以进行匹配。更为重要的是，为该选项卡中的函数显示的数据会进行更新以反映过滤器，因此仅会显示在该时间段内使用的 CPU 时间。



9. 您还可以过滤符合某个特定度量的数据。右键单击 `work_run_usrcpu` 的 "CPU Time (Exclusive)"（CPU 时间（独占））度量。选择 "Show only rows where"（仅选择符合以下条件的行）> "CPU Time (Exclusive) == the metric shown for `work_run_usrcpu`"（CPU 时间（独占）== work_run_usrcpu 显示的度量）。所有其他行都会过滤掉，仅显示 CPU 独占时间等于该度量的行。

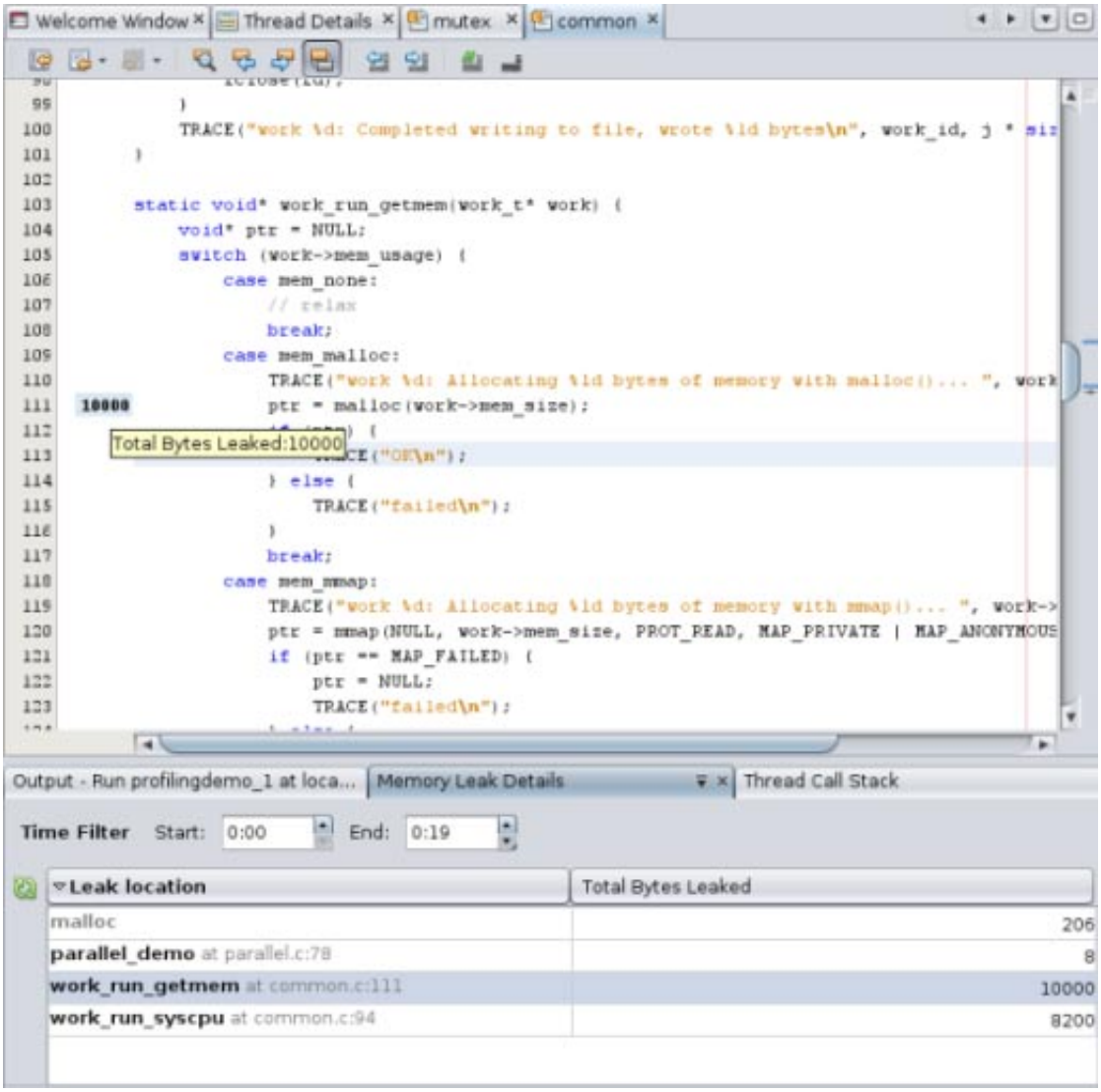


了解内存使用情况

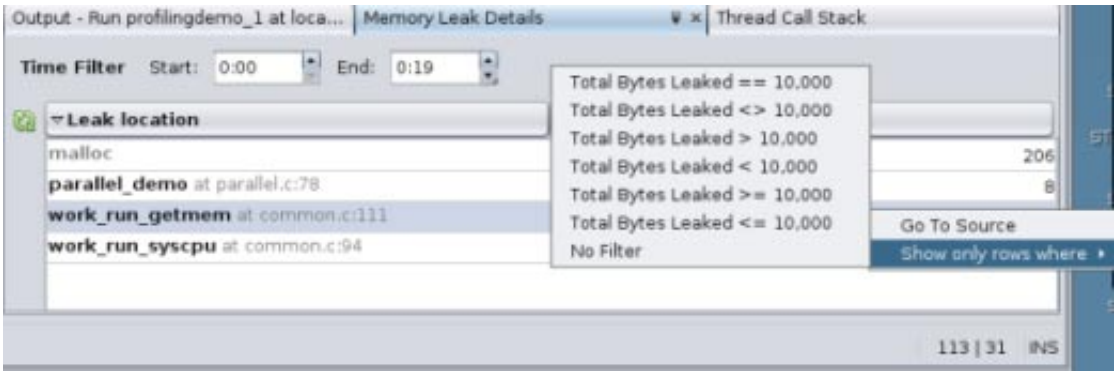
"Memory Usage"（内存使用情况）工具显示程序的内存堆是如何随其运行时改变的。您可以使用该工具来识别内存泄漏，内存泄漏是指程序中不再需要的内存无法释放的点。内存泄漏可能会导致程序中的内存消耗增加。如果某个存在内存泄漏的程序运行足够长的时间，它最终可能会用尽可用内存。

1. 在 "Run Monitor"（运行监视器）中向左和向右滑动 "Time"（时间）滑块可查看内存堆是如何随时间而增加和减少的。在程序的运行中有四个使用量高峰。前两个发生在 `SEQUENTIAL DEMO` 期间，第三个发生在 `PARALLEL DEMO` 期间，第四个发生在 `PTHREAD MUTEX DEMO` 期间。

2. 单击 "Memory Leaks"（内存泄漏）按钮可显示 "Memory Leak Details"（内存泄漏详细信息）选项卡，该选项卡显示有关哪些函数表现出内存泄漏的详细信息。该选项卡中仅会列出产生内存泄漏的函数。如果单击该按钮时程序正在运行，显示的泄漏位置就是单击按钮那一刻存在的位置。随着时间的推移，可能会出现更多泄漏，因此请单击 "Refresh"（刷新）按钮  更新列表。如果直到运行结束仍未检测到任何内存泄漏，"Memory Leak Details"（内存泄漏详细信息）选项卡会指示未找到内存泄漏。
3. 您可以通过更改开始时间和结束时间或者通过使用 "Run Monitor"（运行监视器）窗口中的 "Details"（详细信息）滑块来过滤数据。
4. 在此运行中，`ProfilingDemo` 程序显示了一个与 `work_run_getmem` 函数相关联的内存泄漏。双击 `work_run_getmem` 函数，`common.c` 文件将打开，且光标位于函数中发生内存泄漏的行上。
5. 在左边界中会显示内存泄漏度量。如同 "CPU Usage"（CPU 使用情况）度量一样，将鼠标移动到内存泄漏度量上也会显示详细信息。



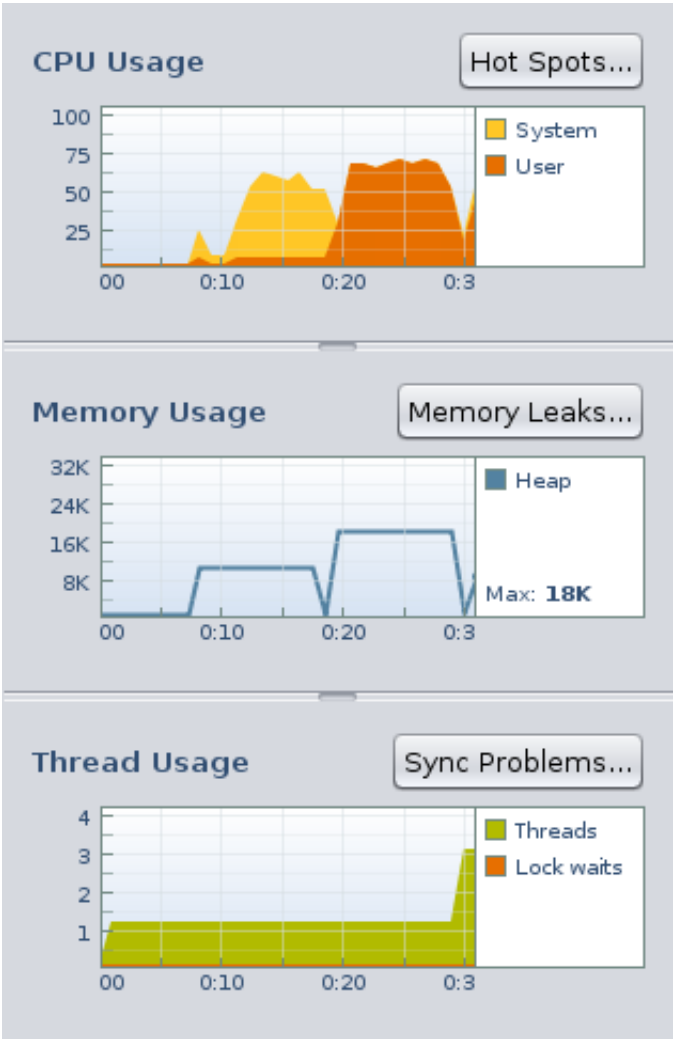
6. 与 "CPU Time Per Function"（每个函数的 CPU 时间）选项卡一样，您可以在 "Memory Leak Details"（内存泄漏详细信息）选项卡中右键单击度量，然后选择一个用于过滤数据的标准。



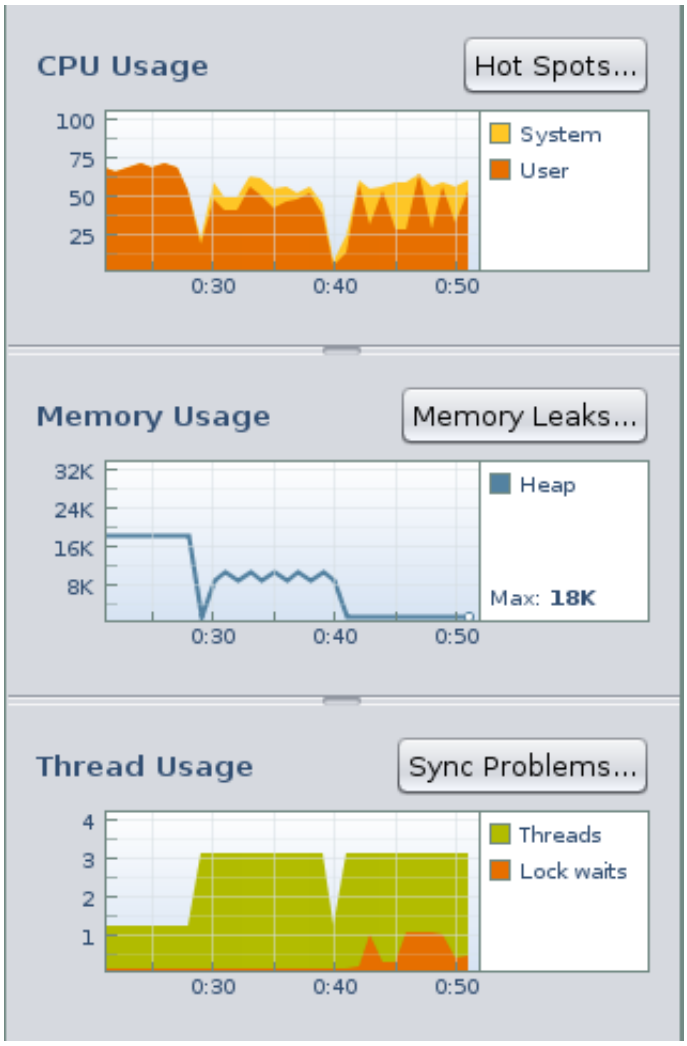
了解线程使用情况

"Thread Usage"（线程使用情况）工具显示程序所使用的线程数量，以及线程为了继续其任务必须等待获取锁定的所有时刻。此数据对于多线程应用程序很有用，这些应用程序为了避免昂贵的等待时间必须执行线程同步。

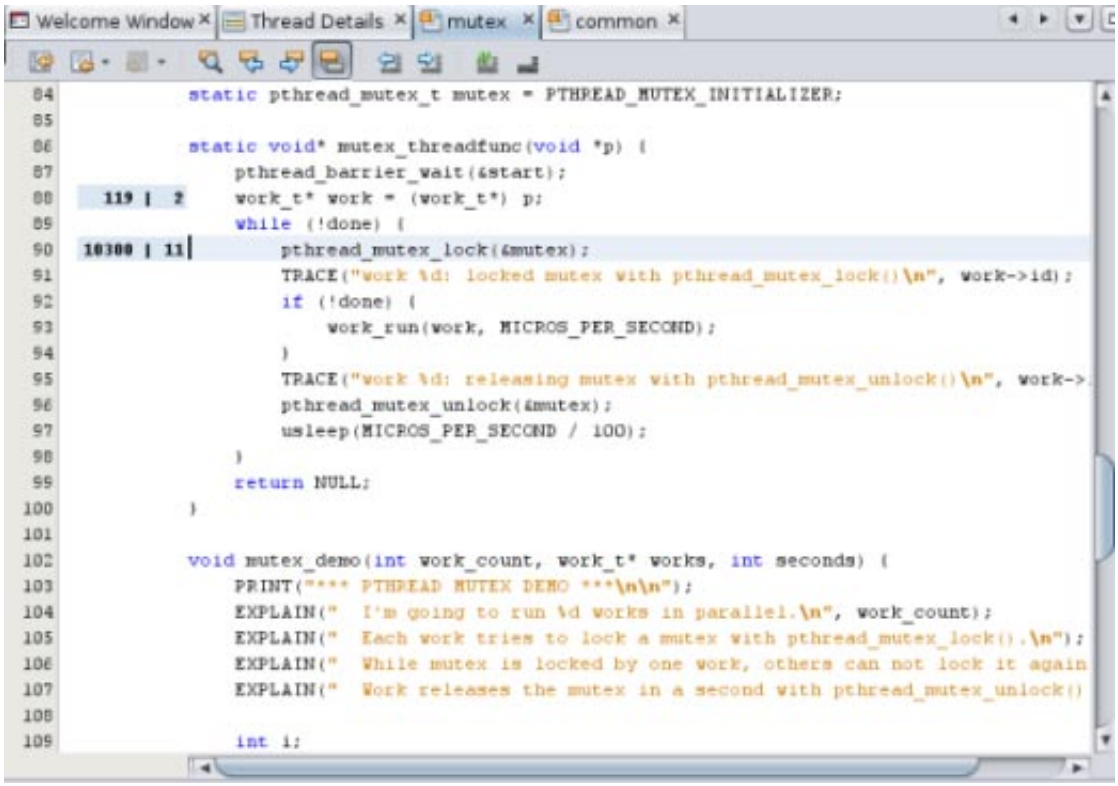
1. 将 "Time"（时间）滑块滑动到运行的开始处，并注意（如同在 "Thread Microstates"（线程微观状态）图中一样），在程序的 `SEQUENTIAL DEMO` 部分期间线程数量为一，但随着程序进入 `PARALLEL DEMO` 部分，线程数量会增加到三。
2. 移动 "View"（视图）滑块的端点手柄，这样您便可以看到从运行的开始处到另外两个线程启动之前的数据。
3. 查看同一时间段的 "CPU Usage"（CPU 使用情况）图和 "Memory Usage"（内存使用情况）图，并注意是单个线程正在执行某个使用 CPU 时间和内存的活动。此时间段与程序的 `SEQUENTIAL DEMO` 部分相对应，在该部分中主线程写入到文件，然后按顺序执行某些计算。在程序等待用户按 Enter 键时，CPU 和内存使用量都会降低，而线程数仍然是一个。



4. 将 "Time"（时间）滑块向右滑动，这样您便可以看到线程数增加到三时的两个点。线程数的第一次增加与程序运行的 `PARALLEL DEMO` 部分相对应，在该部分中主线程又启动了两个其他线程，以并行方式执行写入到文件和执行计算的工作。请注意，在此部分期间内存使用量和 CPU 使用量有一点高，但完成这两个任务的时间比 `SEQUENTIAL DEMO` 部分中少得多。



5. 请注意在 PARALLEL DEMO 线程完成之后线程数又恢复为一个，而主线程会等待用户按 Enter 键。
6. 在程序的 PTHREAD MUTEX DEMO 部分运行时，线程计数会再次增加到三个。请注意，在线程计数增加到三个之后不久，会出现一个锁定等待，以橙色显示。PTHREAD MUTEX DE O 使用互斥锁 防止多个线程对某些函数进行重叠访问，这导致线程需要等待获取锁定。
7. 单击 "Sync Problems"（同步问题）按钮可显示有关线程锁定的详细信息。"Thread Synchronization Details"（线程同步详细信息）选项卡将打开，并列出需要等待获取互斥锁的函数。同时还显示函数等待所花费的毫秒数以及函数必须等待某个锁定的次数的度量。
8. 如果在程序正在运行时单击了 "Sync Problems"（同步问题）按钮，可能需要单击 "Refresh"（刷新）按钮  使用最新的线程锁定更新显示。
9. 单击 "Wait Time"（等待时间）列的标题可按等待所花费的时间的顺序对函数进行排序。
10. 单击 "Lock Waits"（锁定等待次数）列的标题可按线程在函数中等待的次数对函数进行排序。
11. 双击 mutex_threadfunc 函数，该函数的锁定等待次数最多。mutex.c 源文件将打开，且光标位于调用 pthread_mutex_lock 函数的行上。此函数负责在读取或写入内存位置之前锁定该内存位置，而且必须等待，直到没有任何其他线程在该内存位置上有锁定为止。

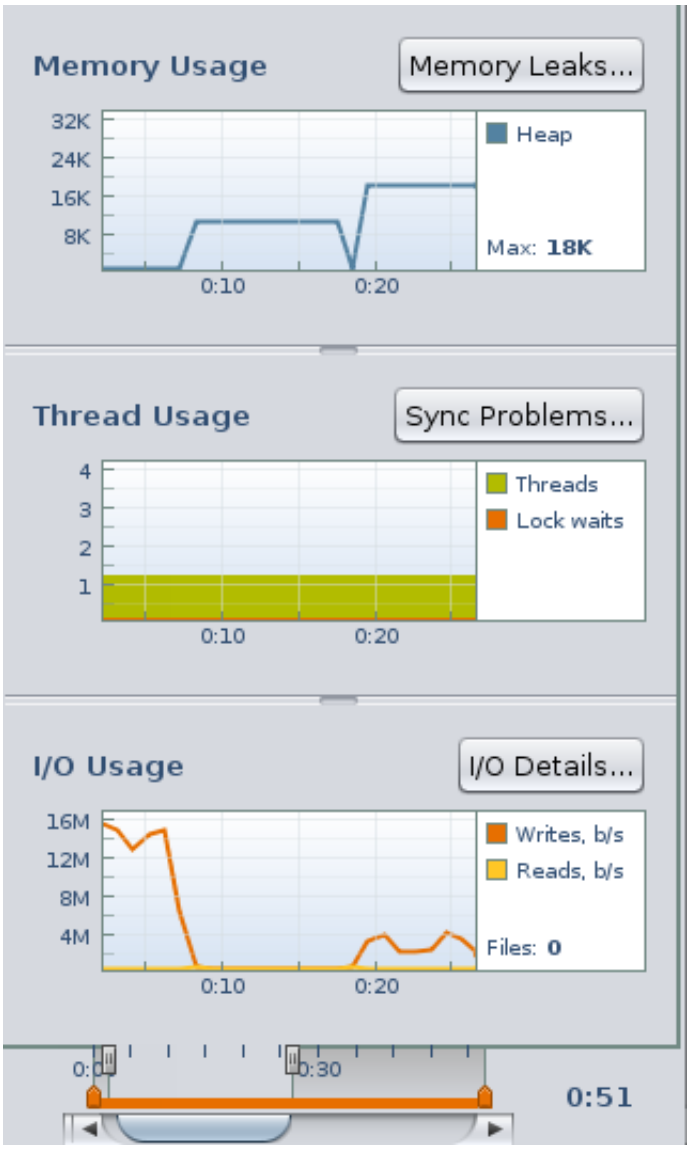


12. 在源文件的左列边界中显示了 "Wait Time"（等待时间）和 "Lock Waits"（锁定等待次数）的度量。将鼠标光标放置到这些度量的上方可查看详细信息，这些详细信息与 "Thread Synchronization Details"（线程同步详细信息）选项卡中显示的内容相符。
13. 右键单击这些度量，并取消选定 "Show Profiler Metrics"（显示配置程序度量）。这些度量将不会再显示在任何配置工具的程序编辑器中。
14. 选择 "View"（视图）> "Show Profiler Metrics"（显示配置程序度量）可再次显示这些度量。

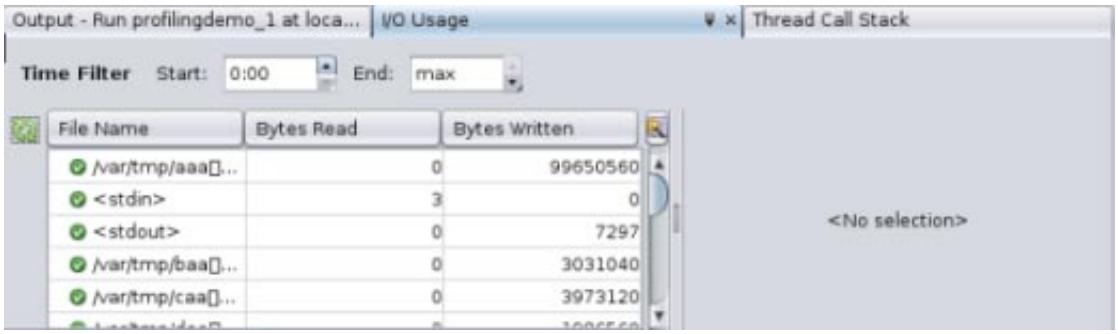
了解 I/O 使用情况

"I/O Usage"（I/O 使用情况）工具显示运行期间程序的读取和写入活动的概述。

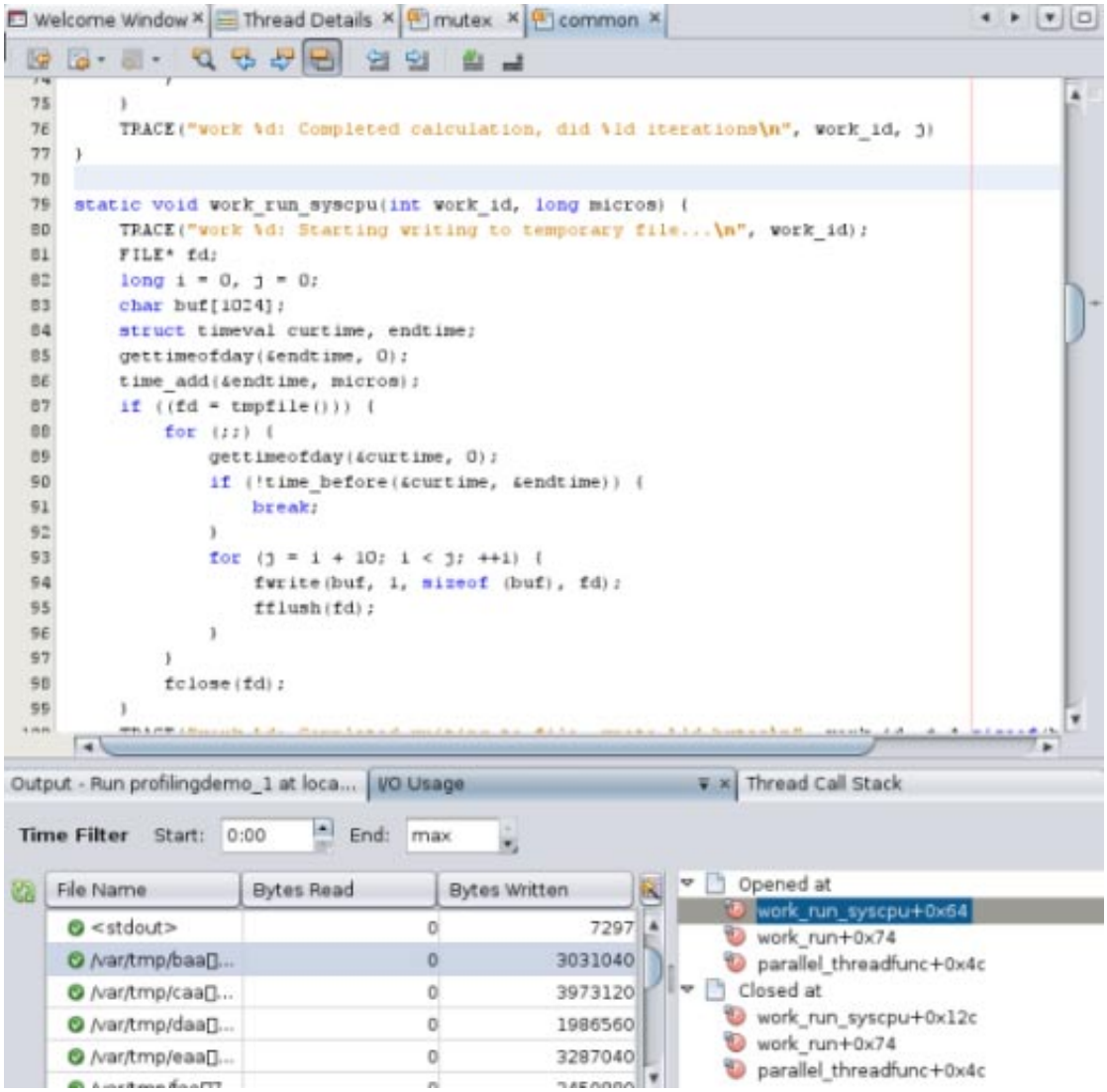
1. 下面的图像显示了在 `SEQUENTIAL DEMO` 部分期间（在此期间两个任务在单一线程中相继运行）运行开始处的 I/O 使用情况。在前几秒钟，程序启动，然后等待用户按 Enter 键。用户按 Enter 键时，程序会将字符写入到某个临时文件中。此活动在该图形中反映为显示写入的字节数的橙色线。



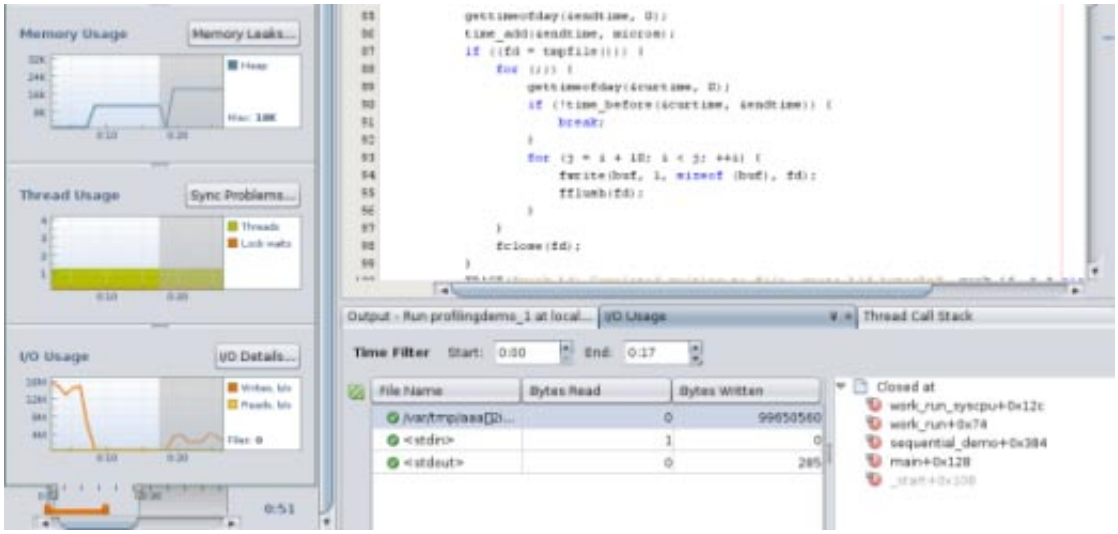
2. 请注意以下事项：
   - 橙色线显示了最后一秒写入的字节数。"Profiling Demo"（配置演示）程序会自己报告到"Output"（输出）窗口中：在 `SEQUENTIAL DEMO` 的这次运行期间，共写入 7998464 个字节（大约 76.3M）。如果将橙色线上显示的所有数据点加起来，总数会接近 76.3M。
   - 在此阶段，"CPU Usage"（CPU 使用情况）工具中显示的系统时间值很高，因为程序利用系统调用来生成数据并将其写入到磁盘。
   - "Memory Usage"（内存使用情况）工具显示了一个已分配的稳定的 8K 字节内存堆。
3. 单击 "I/O Details"（I/O 详细信息）按钮。"I/O Usage"（I/O 使用情况）详细信息选项卡将打开，并显示标准输入、标准输出以及程序读取自和写入到的临时文件。带有复选标记的文件已关闭。标有黄色图标的文件仍处于打开状态，以供执行读取和写入操作。



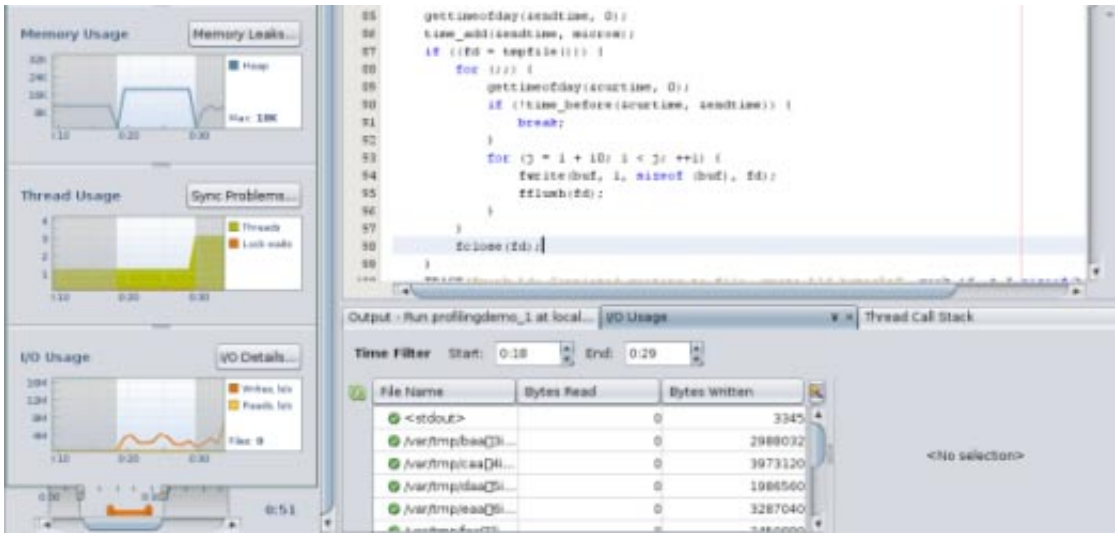
4. 请注意，此程序只需要偶尔按 Enter 键进行输入，因此 "Bytes Read"（读取的字节数）列不是很有用。您可以通过单击列标题右侧的 "Change Visible Columns"（更改可见列）按钮关闭 "Bytes Read"（读取的字节数）列。在 "Change Visible Columns"（更改可见列）对话框中，单击该复选框以取消选定 "Bytes Read"（读取的字节数），然后单击 "OK"（确定）。
5. 假定您需要有关此程序如何使用所有这些临时文件的更多详细信息。单击 "I/O Usage"（I/O 使用情况）详细信息选项卡中的 `/var/tmp/baa[]...` 文件可查看打开和关闭该文件的函数。会在面板中文件列表的右侧列出这些函数。
6. 双击函数列表中的 `work_run_syscpu` 函数可打开该函数的源文件。



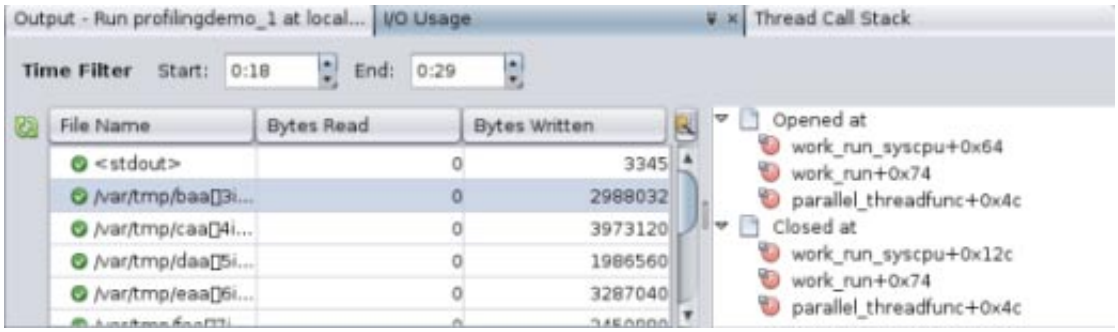
7. 与 "CPU Time Per Function"（每个函数的 CPU 时间）选项卡和 "Thread Synchronization Details"（线程同步详细信息）选项卡中一样，在 "I/O Usage"（I/O 使用情况）详细信息选项卡中，您可以指定一个查看时间间隔。在 "Run Monitor"（运行监视器）窗口中的 "I/O Usage"（I/O 使用情况）图中，写入活动在离运行的开始处（此时程序的 `SEQUENTIAL DEMO` 部分开始写入到某个临时文件）非常近的位置启动。
8. 通过在 "End"（结束）字段中键入运行结束的时间（在下图中为 0:17），分离运行的 `SEQUENTIAL DEMO` 部分。数据是在 "Run Monitor"（运行监视器）窗口和 "I/O Usage"（I/O 使用情况）详细信息选项卡中过滤的，因此您可以将焦点置于 `SEQUENTIAL DEMO` 阶段的输入和输出上。



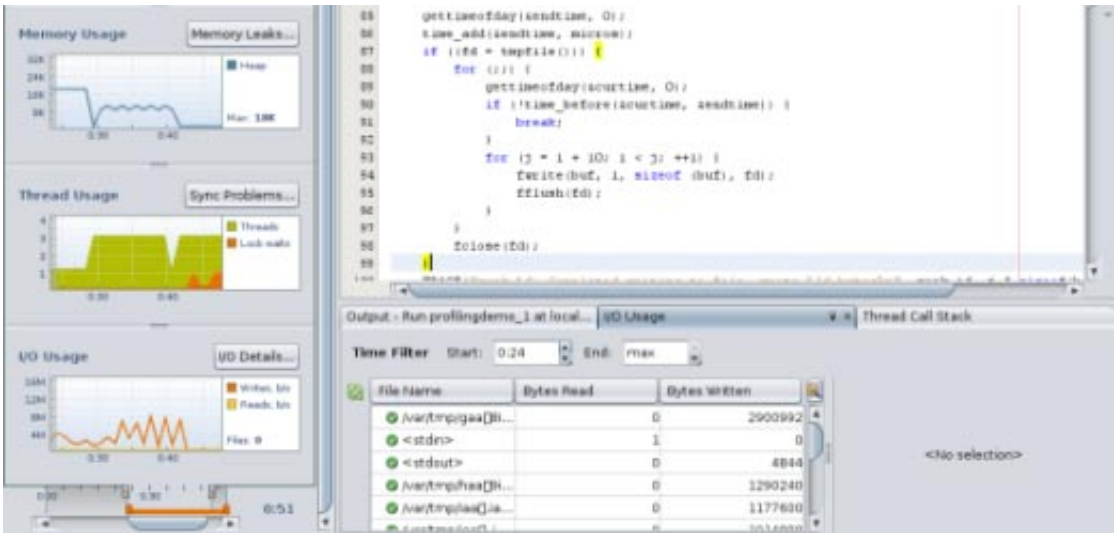
9. 请注意，在顺序演示期间，"I/O Usage"（I/O 使用情况）详细信息选项卡中会显示一个临时文件。一个单一线程将打开该文件，写入到该文件，然后关闭该文件。单击该文件可查看访问该文件的函数，然后双击某个函数可查看其源代码。
10. 在 "Run Monitor"（运行监视器）窗口中，向右拖动 "Time"（时间）滑块，直到看到写入活动在暂停之后再次开始。写入中的暂停发生在顺序演示的第二个任务期间，该任务是一个计算任务，在该任务期间不会发生任何磁盘写入。再次开始的写入活动显示程序正在进入 `PARALLEL DEMO` 部分，在该部分中用户按 Enter 键以启动任务，而且在单独的线程中同时向磁盘和计算写入。
11. 通过键入并行演示活动在 "Start"（开始）字段中开始的时间和在 "End"（结束）字段中结束的时间，可分离并行演示活动。对于此图像中显示的运行，开始时间是 0:18，结束时间是 0:29。



12. 请注意，几个临时文件会在运行的 PARALLEL DEMO 部分期间显示在 "I/O Usage"（I/O 使用情况）详细信息选项卡中。只有一个线程会写入到这些文件，因为每一秒钟它都会切换到一个新文件。正在计算的任务不会写入到磁盘。
13. 单击其中一个文件，会看到 parallel_threadfunc 函数正在打开和关闭文件。



14. 单击 "Details"（详细信息）滑块的右手柄并将其拖动到运行的结束处。在下图中，您可以看到，当程序进入 PTHREAD MUTEX DEMO 部分且用户按 Enter 键之后，I/O 活动会再次发生改变。通过将并行演示的开始时间更改为结束时间（在此示例中为 0:24），为运行的此部分过滤数据。



15. "I/O Usage"（I/O 使用情况）详细信息选项卡显示，在运行的 PTHREAD MUTEX DEMO 部分期间有多个文件处于打开状态。一个线程正在写入到某个文件，且每一秒钟它都会切换它写入到的文件，如同在 PARALLEL DEMO 部分中一样。但是，由于使用了互斥锁，有时写入线程会被计算线程阻止，无法继续写入文件。

821-2516

Oracle Corporation 500 Oracle Parkway, Redwood City, CA 94065 U.S.A.