Battery performance mali

Battery performance mali

Thisblog assumesthat the reader has familiarity with graphics terminology, in particular relating to tile-based rendering GPU architectures. Some useful quick-start guides on these topics can be found here:

Once you have followed the Quick Start Guide to set up your application and install the gator daemon to the target, it is time to select some data sources and start profiling. Connect to your device and bring up the Counter Selection dialog. In the Counter Selection dialog select the appropriate template for your device from the drop-down menu.

This will automatically select all of the data sources necessary to render the template's visualization. Click save, and then capture a trace of your application. Once the initial data analysis has completed the default Timeline visualization will be presented:

This will change the Timeline to display a pre-defined visualization, designed by our in-house performance analysis team. This will order the charts in a more methodical sequence, and make use of mathematical expressions to combine multiple raw counters to derive more readable metrics such as percentage utilization of a functional unit.

The initial view that the Timeline presents gives us 1 second per on-screen sample, which is too coarse for debugging graphics content as we are most interested in viewing how well we are processing frames which are typically between 16-32 milliseconds in length. The first step in the analysis is therefore to zoom in the view until single frames become distinct.

In the application shown in the sample we have added instrumentation to the source code to generate a Streamline marker annotation whenever the application callseglSwapBuffers(). These are visible as the red ticks on the time track above the charts.

In our example above we can see that the CPUs are all going completely idle for a significant proportion of the frame, so we are not CPU bound. We can also see that the GPU is active all of the time, so the GPU is highly likely to be the processor limiting this application's performance.

In terms of breaking down the GPU workload further we can see that the fragment shading queue is the one active all of the time, with the non-fragment queue used for all geometry and compute processing going idle for most of the frame. You would therefore look to optimize fragment workload for this application if you wanted to improve performance.

The following sections in this tutorial work through each of the charts in the template, and explain what they mean and what changes they could imply for an application developer looking to improve performance.

TheCPU Activitycharts show the per-CPU utilization, computed as a percentage of time the CPU was active, split by processor type if you have big.LITTLE clustering present. This is based off OS scheduling event data. TheCPU Cycleschart shows the number of cycles that each CPU was active, measured using the CPU performance monitoring unit (PMU). By considering both of these together we can assess the overall application software load; a high utilization and a high CPU cycle count indicate that the CPU is both very busy and running at a high clock frequency.

The process view at the bottom of the Timeline tab shows the application thread activity, allowing an identification of which threads are causing the measured load. Selecting one or more threads from the list will filter the CPU related charts so that only the load from the selected threads is shown. When a thread-level filter is active the chart title background changes to a blue-tinted color to indicate that not all of the measured load is currently visible.

If the application is not hitting its performance target and has a single CPU thread which is active all of the time then it is likely to be CPU bound. Improvements to frame time will require software optimizations to reduce the cost of this thread's workload. Streamline provides native software profiling via program counter sampling, in addition to the performance counter views. Software profiling is beyond the scope of this tutorial, so please refer to the Streamline User Guide for more information.