Profiling OpenCL™ Applications with System Analyzer and Platform Analyzer
Introduction

Intel® SDK for OpenCL™ Applications is a powerful software environment for OpenCL™ application development. System Analyzer and Platform Analyzer enable you to profile OpenCL code in visual computing applications. These are key components for profiling the OpenCL API and associated metrics support.

System analyzer collects and displays hardware and software metrics data from your application in real time. Platform Analyzer targets offline performance analysis of applications that use a graphics processing unit (GPU) for rendering, video processing, and computations. Specifically, Platform Analyzer collects real-time profiling data during the application run. Beyond tracing important OpenCL APIs, it captures important metrics for Intel® HD Graphics OpenCL device, such as utilization of execution units or memory traffic. Platform Analyzer then visualizes the execution profile of your application.

This article is an overview of the OpenCL support provided in System Analyzer and Platform Analyzer on the Windows* OS. Notice that more fine-grain methods of profiling exist, specifically with the Intel® VTune™ Amplifier XE. Still, both System and Platform Analyzers are great starting points for understanding the general platform (CPU and GPU) utilization via user-level OpenCL APIs.
What’s New

System Analyzer and Platform Analyzer profile GPU cores when running your application, which together with capturing general CPU activity, enables you to correlate activities on both devices. And you can also identify which device (and API) mostly bounds your application.

Improvements to OpenCL API support are numerous and include:

**Support for profiling OpenCL applications.** Previously, only CPU metrics were available for OpenCL applications, such as console-based applications.

**Tracing the application startup.** This feature is particularly useful when you need to measure the initialization costs or analyze very short applications.

**Improved support for metrics.** Now you can explore the performance of your application per selected GPU metrics over time in a much more accurate way (with less than 1ms granularity).

More importantly, the metrics are no longer tied to the frames. The fine-grained metrics are now available for all analysis types:

- In the **System Analyzer**’s real-time view, in both the Heads-up Display (HUD) and the standalone version:

  ![Fine-grained metrics in the System Analyzer](image)

  **Figure 1. Fine-grained metrics in the System Analyzer.**

- In the **Platform Analyzer**, where you can zoom in to the specific time region or frame and inspect it in detail offline. Refer to the “API Level Analysis with Platform Analyzer” section below.
For details on the metrics, refer to the Online help.

Computing queue for OpenCL kernels in the Platform Analyzer timeline view. The Platform Analyzer view has been extended to show information on the OpenCL device queues (see Figure 2).

![Figure 2. OpenCL device queue (named “Intel® HD Graphics 4000”, just as the device). This computing queue displays the stacked view of queued kernels (marked in blue) and execution path (marked in red). Notice the kernel names in the stacked view.](image)

Now you can better understand application flow over time with the following options:

- Match the GPU software queue (and associated DMA packets queue/execution) and the OpenCL device queues with OpenCL API commands issued by the host (CPU).
- Explore what contribution Microsoft DirectX* calls make to the same GPU software queue and the resulting GPU load (via DMA packets execution), and how these calls interfere with DMA packets that originate from the OpenCL API.
- Analyze video processing tasks from the Intel® Media SDK that manifest themselves in a separate DMA packets queue (mostly served with dedicated hardware-assisted execution, thus minimizing contention for GPU resources with rendering and computing tasks) and understand how Intel Media SDK tasks and OpenCL/DirectX execution overlap.
- For all three types of activities above, inspect overall device utilization with fine-grained GPU metrics as discussed above.
Refer to the [API Level Analysis with Platform Analyzer](#) section for the example trace.
Configuring an Analysis Profile

To configure an analysis profile:


2. Right-click the Monitor icon in the system tray and select Profiles... in the context menu to open the Profiles dialog:

![Profiles dialog box]

**Figure 3. Tracing tab in the Profiles dialog box.**

New options are now available on the Tracing tab of the Profiles dialog box:

- **Capture Application Startup** checkbox enables you to start tracing immediately upon application startup, which is useful for short applications that can be hard to trace via the regular Ctrl+Shift+T shortcut.

  Note: You can specify the trace duration with the spin box. To specify various types of events used to trigger the tracing, switch to the Trigger tab.

- **View fine-grained system-wide GPU metrics** checkbox enables collecting accurate data on the hardware counters.

- **Collect DMA Packet Type** checkbox enables collection of DMA packets, which is the key approach in correlating the user-level API calls and the resulting device load.
Running Platform Analysis

To start profiling your application:

1. Right-click the Monitor icon in the system tray and choose the Analyze Application... in the context menu.
2. Specify an application path and options in the dialog box, and click Run:

![Figure 4. Left: Intel Graphics Monitor context menu. Right: Analyze Application dialog.](image)

Now your application runs in the instrumented mode and you can capture the trace with Ctrl+Shift+T. Refer to the Online help on how to specify metrics of interest, how to conditionally capture the trace, etc. Note: If your application is very short in execution, consider using the new Capture Application Startup configuration option (see the previous section).
API Level Analysis with Platform Analyzer

Once you have generated a trace of your application, the next step is inspecting the timeline by opening the trace in the Platform Analyzer. It offers very handy ways to pinpoint the hotspots in execution and correlate them to the API calls.

In general, the particular analysis depends on the identified areas for improvement. For example, you may see that DirectX calls dominate the GPU execution path. In this case, use Frame Analyzer because it relies on the frame capture file (generated similarly to the trace capture, but with the Ctrl+Shift+C shortcut). The frame capture helps you understand exactly what is happening within your application on a frame-by-frame basis.

For the rest of the document we focus on features related to the OpenCL API. Specifically, we use the OpenCL and Intel Media SDK Interoperability code sample that exploits Intel® Media SDK for initial video decoding, processes the decoded video frame with the OpenCL API, and finally displays the resulting image on the screen with a DirectX API. So all three APIs are utilized in one sample!

Below is an example trace, viewed in the Platform Analyzer:

Figure 5. Example trace viewed in the Platform Analyzer. Note how the execution path (marked in red) of the OpenCL device queue (in blue) correlates to the DMA packets queue (in black). Hovering over any packet in
the DMA queue highlights its path through the queue to the actual execution by the GPU.

As you can see in Figure 5, the application has two kernels, Mouse and Process, in the OpenCL queue. Both follow the execution path, where the commands from the queue are executed back-to-back. Since Mouse has only a work-item (hover over an object to get the pop-up hint with kernel execution parameters), it is executed so fast, that you would need to zoom to spot it on the execution path.

In turn, the OpenCL queue execution path expedites kernels to the driver, where DMA packets of different types get multiplexed in the single DMA queue. This Render and GPGPU queue serves both graphics-originated (tagged “GHAL”) and compute-originated (tagged “OpenCL”) packets. Note that video transcoding tasks pass through a dedicated Video Codec queue, which enables the Intel Media SDK commands to run on the GPU in parallel in the majority of cases.

Unlike the OpenCL device queue where different colors are assigned to different kernels (matching the colors in the OpenCL execution path), the DMA queues have just two colors: light green for the packets still stacked in the DMA queue, and yellow for the packets currently being processed by the GPU. The DMA packets with DirectX “Present” calls are marked with cross-hatching (XXXXXX), and the color scheme is the same: green for queued DMA packets, yellow for packets being executed.

For more details on the Platform Analyzer and its GUI, refer to the Online help.
As we discussed in the previous section, the application optimization is typically started with a user-space analysis, for example, API-level tracing with assistance from the Platform Analyzer to sanitize a general application flow (previous section) and check that overall GPU utilization is ok.

After this phase of API-level analysis, you can focus on the most expensive OpenCL kernels with the help of specific metrics. The metrics appear on the same timeline:

![Figure 6](image.png)

**Figure 6.** Example of fine-grained metrics (charts at the bottom of the screenshot) in the Platform Analyzer. Notice that metrics appear on the same timeline. Also notice the resolution of the metrics (time scale on top). Hover over any specific point on the chart to get a popup hint with the exact value.

The Platform Analyzer supports the same types of OpenCL kernel metrics as the previous release:

- CPU-specific metrics, such as core utilization
- Intel HD Graphics execution units (EUs) metrics, of which GPU EUs active/idle/stalled are the most important
• Memory metrics, such as GPU memory reads/writes
• Power metrics for CPUs, Intel HD Graphics device, and the whole package

Unless your algorithm is memory-bound, the execution units (EUs) are likely to gate the performance of your application. The EU metrics can provide information on these bottlenecks. The goal is to maximize the utilization of EUs with useful computations. Refer to the OpenCL Applications Optimization Guide for tips and tricks.

The following information briefly describes the EU-related metrics:

• **GPU EUs Active** represents the percentage of time when the GPU execution units (EUs) were actively executing. **GPU EUs Idle** is the percentage of time when the GPU execution units (EUs) are neither actively executing instructions nor stalled (below).

• **GPU EUs Stalled** metric represents the percentage of time when the GPU execution units (EUs) were stalled. An EU becomes stalled when all of its threads are waiting for results from fixed function units, for example, requesting data from the Data Port or Sampler.

If GPU EUs Stalled is quite high, this might indicate inefficient memory bandwidth usage (for example, suboptimal data access granularity or cache thrashing, so that the GPU waits for data to arrive). See the OpenCL™ Optimization Guide for theoretical memory performance and hints on saturating the bandwidth.

Finally, if the number of workgroups in the flight is insufficient, EU utilization might be really low (GPU EUs Idle will be high). Too low a value for the local size provided to the clEnqueueNDRange call can also result in units being idle. Again, refer to the OpenCL™ Optimization Guide for details.

Also try the OpenCL™ and Intel® Media SDK Interoperability code sample, which enables you to pause/resume Intel Media SDK decoding and OpenCL code processing with a simple GUI. You can experiment with it to understand what effect the different sample pipeline stages have on the metrics.
Using Instrumentation and Tracing Technology (Intel® ITT) APIs for Custom Instrumentation of the CPU Code

Since the primary target for Platform Analyzer is GPU efficiency, it does not offer many insights into CPU code, beyond tracing the recognized OpenCL or DirectX API calls. For example, it does not provide hotspots for your general C/C++ code (unlike Intel® VTune™ Amplifier XE that provides much deeper analysis and source-level hotspots view). Still, you can check the overall CPU core utilization with the Platform Analyzer.

You can also annotate any CPU code with ITT API to explore how execution flow of the particular code region appears on the timeline with respect to the rest of activities. This sort of user instrumentation works well for both the Platform Analyzer and VTune Amplifier XE.
Summary

This paper covered the following key points:

- Platform Analyzer provides a powerful way to analyze OpenCL applications with support for the OpenCL device queues.
- You can inspect the overall GPU utilization and see the breakdown for each API, including the OpenCL API.
- You can explore a set of accurate metrics that cover the host CPU and the Intel HD Graphics OpenCL device.

Refer to the Online help for more information.
Notices

By using this document, in addition to any agreements you have with Intel, you accept the terms set forth below.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: http://www.intel.com/design/literature.htm

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark* and MobileMark*, are measured using specific computer systems, components, software, operations, and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Any software source code reprinted in this document is furnished under a software license and may only be used or copied in accordance with the terms of that license.
Summary

Intel, Core, VTune, and the Intel logo are trademarks of Intel Corporation in the US and/or other countries.

OpenCL and the OpenCL logo are trademarks of Apple Inc and are used by permission by Khronos.

Copyright © 2014 Intel Corporation. All rights reserved.

*Other names and brands may be claimed as the property of others.