ATPESC 21

Intel® VTune Profiler and Intel® Advisor Hands on in Intel® DevCloud

Kevin O’Leary
Agenda

1. Intel® DevCloud Setup
   Information on starting a GPU node in DevCloud.

2. Workload Description
   Overview of MandelbrotOMP sample and changes.

3. Intel VTune Profiler Server Setup
   Brief explanation on setting up Intel VTune Profiler Server in the DevCloud node.

4. Demo
   Running the sample in the DevCloud with Intel Advisor and Intel VTune Profiler.
Log into an Intel® DevCloud GPU node and configure the MandelbrotOMP sample.

Run Intel Advisor: Offload Advisor to estimate performance on Gen9 GT2 GPU.

Run Intel Advisor: GPU Roofline on offloaded implementation to visualize GPU performance.

Run Intel VTune Profiler: GPU Hotspots for deeper insights into GPU kernels and device metrics.
DevCloud Setup

1. Log into DevCloud via ssh
2. Start interactive gpu node:
   
   
   $ qsub -I -l
   nodes=1:gpu:ppn=2

3. Create MandelbrotOMP sample:
   
   https://github.com/oneapi-src/oneAPI-samples

4. Start Intel VTune Profiler Server in second ssh terminal

Intel DevCloud provides a free environment for testing the latest Intel CPUs and GPUs. Intel oneAPI toolkits are already installed and set up for use.

To create a DevCloud account, follow these steps:

This sample runs one or all of four algorithms for generating a Mandelbrot image. Each algorithm has an increasing level of optimization, from a serial implementation to using OpenMP for parallelization and simd vectorization.

Github link: https://github.com/oneapi-src/oneAPI-samples/tree/master/DirectProgramming/C%2B%2B/CombinationalLogic/MandelbrotOMP
To help demonstrate the capabilities of Intel Offload Advisor, we added a fifth function to use OpenMP offload to a GPU target:

- **src/mandelbrot.cpp**
  - **Copy the** `omp_mandelbrot (..)` **function and rename to** `offload_mandelbrot (..)`
  - **Change** `#pragma omp parallel for` **schedule to**:
    - `#pragma omp target teams distribute \`  
      `parallel for simd collapse(2) \`  
      `map(from:output[0:width*height])`  
      `map(to:height,width,xstep,ystep,max_depth)`

- **src/mandelbrot.hpp**
  - **Copy the** `omp_mandelbrot (..)` **function and rename to** `offload_Mandelbrot (..)`
Add a fifth option to enable the new offload_mandelbrot function

- src/main.cpp
  - Change the `max_depth` from 100 to 5000
  - Add variable `offload_time` to
    - `double serial_time, omp_simd_time, omp_parallel_time, omp_both_time;`
  - Add section for `offload_mandelbrot` under `printf("\nRunning all tests\n")`
  - Add case 5 with `offload_Mandelbrot` to switch (option)
  - Not using PERF_NUM
MandelbrotOMP Makefile

- Change options to use OpenMP offload capability
  - Change compiler from icpc to icpx
  - Remove qopenmp from CFLAGS and LIBFLAGS and add: -fiopenmp -fopenmp-targets=spir64
  - Add -g -D__INTEL_COMPILER to CFLAGS
Intel® VTune Profiler Server Setup

- Follow the instructions in the online Intel VTune Profiler Performance Analysis Cookbook:

- After setting up the ssh terminal for the DevCloud GPU node, open a new terminal and run:
  - $ ssh -L 127.0.0.1:55001:127.0.0.1:55001 devcloud
  - $ ssh -L 127.0.0.1:55001:127.0.0.1:55001 <node>
  - $ vtune-backend --web-port=55001 --enable-server-profiling

- Copy the URL provided into the browser to start the Intel VTune Profiler GUI
Demo

- Running the sample in the DevCloud with Intel Advisor and Intel VTune Profiler.
Demo Steps

- Example screenshots and commands from the demo follow
Log into Intel® DevCloud GPU Node

- Follow the instructions on slide 4 to open a ssh terminal for an interactive GPU node on DevCloud. This node uses Intel processor codenamed Coffee Lake and has an integrated Gen9 GT2 GPU.
  
  - `qsub -l nodes=1:gpu:ppn=2`
Run the following Intel Advisor CLI commands on the parallel OpenMP implementation of MandelbrotOMP (option 3) to estimate the performance benefits of offloading to a Gen9 GT2 GPU:

- `advisor --collect=survey --project-dir=./parallel_mandel --stackwalk-mode=online --static-instruction-mix -- /home/uxxxxx/MandelbrotOMP/release/Mandelbrot 3`

- `advisor --collect=tripcounts --project-dir=./parallel_mandel --flop --target-device=gen9_gt2 -- /home/uxxxxx/MandelbrotOMP/release/Mandelbrot 3`

- `advisor --collect=projection --project-dir=./parallel_mandel --config=gen9_gt2 --no-assume-dependencies`
Package Results and copy to local system

- Package the Intel Advisor project on the DevCloud node and copy to your local system with Advisor 2021.3 installed:

  - advisor --snapshot --project-dir=./parallel_mandel --pack --cache-sources --cache-binaries -- ./parallel_mandel_snapshot
This report shows that a speed up of 1.9x can be gained by offloading the loops.

The loop is expected to run for 538.2ms on the GPU.
Explore different GPU Configurations

Reconfigure GPU settings to a hypothetical new GPU

Then save custom config to scalers.toml

Rerun projection

advisor --collect=projection --project-dir=./parallel_mandel --custom-config=scalers.toml --no-assume-dependencies
View how we are running compared to system max

- Use Intel® Advisor CLI to generate a GPU Roofline report on the offload implementation (option 5):
  - advisor --collect=survey --project-dir=./offload_mandel --profile-gpu -- /home/uxxxxx/MandelbrotOMP/release/Mandelbrot 5
  - advisor --collect=tripcounts --project-dir=./offload_mandel --flop --profile-gpu -- /home/uxxxxx/MandelbrotOMP/release/Mandelbrot 5
  - advisor -report=roofline -gpu -project-dir=./offload_mandel --report-output=./gpu_roofline.html

- Create a snapshot for download to the local GUI:
  - advisor --snapshot --project-dir=./offload_mandel --pack --cache-sources --cache-binaries -- ./offload_mandel_snapshot
The overall elapsed time of 4.67s is much higher in the offloaded version than the parallel CPU implementation (1.49s). But the compute task has a speed-up:

- From 1.03s in `parallel_mandelbrot` to 0.72s in `offload_Mandelbrot`. Not quite hitting the estimate of 538.2ms.
- Nearly 4s is spent on the CPU.
The offload task appears to be bounded by the DP Vector Add Peak. Otherwise, it appears to make good use of the GPU.

- EU Array is 99.2% active, and the threading occupancy is almost 100%
- There is an unknown task consuming 3.951s of CPU time with 100% idle GPU time.
Intel VTune Profiler: GPU Hotspots command-line

- Running gpu-hotspots on the command-line
- `vtune --collect gpu-hotspots ./Mandelbrot 5`

- Generating a report
  
  Elapsed Time: 4.386s

- GPU Time: 0.682s
- EU Array Stalled/Idle: 0.8%
- GPU L3 Bandwidth Bound: 0.3%
- Hottest GPU Computing Tasks Bound by GPU L3 Bandwidth
  
  Computing Task  Total Time
  
  ------------

- Sampler Busy: 0.0%
- Hottest GPU Computing Tasks with High Sampler Usage

- FPU Utilization: 96.3%

Copy result directory to local system
Once the Intel VTune Profiler is running with the `vtune-backend` command, open the URL in the browser for the GUI.

- Set the application to `/home/uxxxxxxxx/MandelbrotOMP/release/Mandelbrot` and set the application parameter to 5.
- Run the GPU Compute/Media Hotspots analysis type
The Summary tab shows that although only a small percentage of the overall elapsed time is spent on the GPU, the offload task performs well on the GPU.

The Graphics tab doesn't indicate any major problems. Under the Platform sub-tab, there is an OpenMP task called `zeModuleCreate` that runs for about 3.5s. That explains the high CPU utilization time.
Summary

- You can use Advisor and VTune GUI & CLI to run the collection and to generate the reports.
- Advisor and VTune provides several analysis types to profile GPU workload.
- Each analysis type provides specific insights
Legal Disclaimer & Optimization Notice

- 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. For more complete information visit www.intel.com/benchmarks.

- INFORMATION IN THIS DOCUMENT IS PROVIDED “AS IS”. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

- Copyright © 2018, Intel Corporation. All rights reserved. Intel, Pentium, Xeon, Xeon Phi, Core, VTune, Cilk, and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804
Backup
Set up system for GPU analysis

- To collect GPU hardware metrics on Linux, you need
  - run the collection as root
  or
  - set `/proc/sys/dev/i915/perf_stream_paranoid` to 0
  - have read/write access to `/dev/dri/card*` and `/dev/dri/renderD*` files

- Optional: To collect information about DMA packets on Linux, you need
  - enable `CONFIG_DRM_I915_LOW_LEVEL_TRACEPOINTS` option for i915 kernel module
  - have read/write access to debugFS

VTune Profiler documentation: Set Up System for GPU Analysis
<table>
<thead>
<tr>
<th>Compiler Switches for Performance Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>-gline-tables-only</strong>&lt;br&gt;-fdebug-info-for-profiling**</td>
</tr>
<tr>
<td><strong>-debug offload</strong></td>
</tr>
<tr>
<td><strong>-parallel-source-info=2</strong></td>
</tr>
</tbody>
</table>

VTune Profiler documentation:
- Compiler Switches for Performance Analysis on Linux® Targets
- Debug Information for Linux® Application Binaries