**Assignment**:

Some of the most common problems in today's computer science domain - such as artificial intelligence, simulation, and modeling - involve matrix multiplication. The algorithm is a triply-nested loop with a multiply and an add operation for each iteration. It is computationally intensive and it also accesses a lot of memory. We provided a simple C++ code that performs matrix multiplication. Identify regions to offload to the GPU, visualize performance of GPU kernels, and identify bottlenecks in your application using the Offload Advisor and the GPU Roofline analysis features of the Intel® Advisor Beta .

Please follow tasks below step by step:

Task 1: Compile and run the code and observe the primary results

Task 2: Collect application performance metrics with collect.py

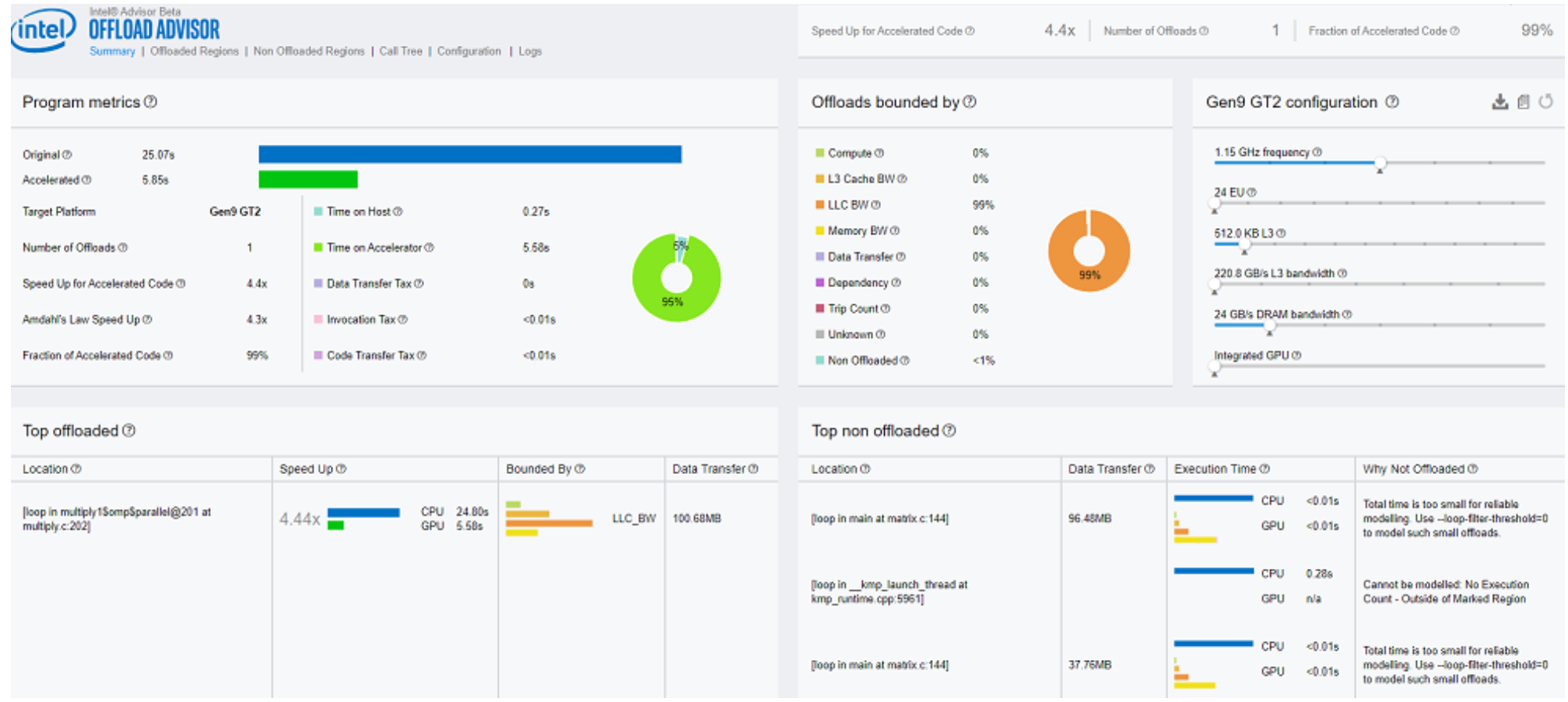
Task 3: Project your application performance on the GPU with analyze.py

Task 4: Open the generated report.html file in a web browser to see the performance projection results

In the Summary section of the report, note the following:

* The original CPU execution time, the predicted execution time on the GPU accelerator, the number of offloaded regions, and the speedup in the Program metrics pane.
* What the offloads are bounded by. In our case, the offloads are 99% bounded by the last-level cache (LLC) bandwidth.
* Exact source lines of the Top Offloaded code regions that will benefit from offloading to the GPU. In our case, there is only one code region recommended for offload.
* Exact source lines of the Top Non-Offloaded code regions that are not recommended for offload for various reasons. In our case, the time spent in the loops is too small to be modeled accurately and one of loops is outside of the code region marked for offloading.

The output should look like below:



Task 5: Rewrite the Matrix Multiply Code in Data Parallel C++

To rewrite the matrix multiply code, perform the following high-level steps, as shown in the code below:

* Select a device.
* Declare a device queue.
* Declare buffers to hold the matrix.
* Submit a job to the device queue.
* Execute the matrix multiply in parallel.

Task 6: Run a GPU Roofline Analysis

* Enable GPU profiling: *export ADVIXE\_EXPERIMENTAL=gpu-profiling*
* Run the Survey analysis with the --enable-gpu-profiling option
* advixe-cl –-collect=survey --enable-gpu-profiling --project-dir=<my\_project\_directory> --search-dir src:r=<my\_source\_directory> -- /home/test/matrix [app\_parameters]
* Run the Trip Count and FLOP analysis with --enable-gpu-profiling :
* advixe-cl –-collect=tripcounts --stacks --flop --enable-gpu-profiling --project-dir=<my\_project\_directory> --search-dir src:r=<my\_source\_directory> -- /home/test/matrix [app\_parameters]
* Generate a GPU Roofline report:
* advixe-cl --report=roofline --gpu --project-dir=<my\_project\_directory> --report-output=roofline.html

Open the generated roofline.html in a web browser to visualize GPU performance. The output should look like this:

