Introduction to my journey towards Intel #oneAPI HPC toolkit:

Hi, I am an AI Application developer, currently working in TCS as Tech Lead and Prinicpal Researcher in my domain across various applications. I do regularly work on different projects and applications which requires the usage of HPC in terms of multithreading, memory optimization and so on. Therefore, while looking for something on the similar lines, I stumbled across Intel oneAPI toolkit and specifically went through Intel OneAPI HPC toolkit and it's integration for various HPC based computational complexities. So I will share my learning via this blog about oneAPI HPC toolkit.

Introduction to oneAPI HPC toolkit:

High-performance computing (HPC) is at the core of AI, machine learning, and deep learning applications. The Intel® oneAPI HPC Toolkit (HPC Kit) delivers what developers need to build, analyze, optimize, and scale HPC applications with the latest techniques in vectorization, multithreading, multi-node parallelization, and memory optimization.

This toolkit is an add-on to the Intel® oneAPI Base Toolkit, which is required for full functionality. It also includes access to the Intel® Distribution for Python*, the Intel® oneAPI DPC++/C++ Compiler, powerful data-centric libraries, and advanced analysis tools.

For details about all the Intel oneAPI toolkits, please visit
https://www.intel.com/content/www/us/en/developer/tools/oneapi/toolkits.html#gs.3npy6n

Features :

Build
Simplify implementation of HPC applications on CPUs and accelerators with Intel's industry-leading compiler technology and libraries.

Analyze
Quickly gauge how your application is performing, how resource use impacts your code, and where it can be optimized to ensure faster cross-architecture performance.

Scale
Deploy applications and solutions across shared memory and distributed memory (such as clusters) computing systems using the included standards-driven MPI library and benchmarks, MPI analyzer, cluster tuning tools, and cluster health-checking tools.

Usage and Deployment:

- Any user can download and install the HPC toolkit, for running it please check the specifications of the system requirements and Hardware from the link : https://www.intel.com/content/www/us/en/developer/articles/system-requirements/intel-oneapi-hpc-toolkit-system-requirements.html

- After Downloading the toolkit, follow the below links to configure your system and run your first sample.
    - For Linux: https://www.intel.com/content/www/us/en/develop/documentation/get-started-with-intel-oneapi-hpc-linux/top.html
    - For Windows : https://www.intel.com/content/www/us/en/develop/documentation/get-started-with-intel-oneapi-hpc-linux/top.html
    - For macOS : https://www.intel.com/content/www/us/en/develop/documentation/get-started-with-intel-oneapi-base-hpc-macos/top.html
    - Using Containers: https://www.intel.com/content/www/us/en/develop/documentation/get-started-with-intel-oneapi-hpc-linux/top/using-containers.html

- Once you have done the setup according to your environment, you can run the Samples program or run your own workloads
    - One can refer code samples in Github for reference: https://github.com/oneapi-src/oneAPI-samples

Run an application:

Now, the best part is after download and installation, let's run an application to test our system. For running, we can run in our system after ofcourse checking the system requirements and proper installation setup OR we can run it Intel Dev Cloud, with an Intel® DevCloud account, you get 120 days of access to the latest Intel® hardware—CPUs, GPUs, FPGAs—and Intel oneAPI tools and frameworks. No software downloads. No configuration steps. No installations. In order to use Intel Dev Cloud, I stumbled upon a very well detailed blog about how to connect with Intel Dev Cloud, please refer the link for reference: https://medium.com/@kazithaque22/intel-dev-cloud-access-via-ssh-for-oneapi-f5d8b8516a16

Now, for today's application sample running, I have selected my favourite problem statement which is Mandelbrot fractal pattern.

If you want to know more about Madelbrot pattern and equation, you can refer the link https://www.britannica.com/science/Mandelbrot-set

Mandelbrot is a SYCL* application that generates a fractal image by initializing a matrix of 512 x 512, where the computation at each point (pixel) is entirely independent of the computation at other points. The sample includes both parallel and serial calculation of the set, allowing for a direct comparison of results. The parallel implementation can demonstrate the use of Unified Shared Memory (USM) or buffers. You can modify parameters such as the number of rows, columns, and iterations to evaluate the difference in performance and load between USM and buffers. This is further described at the end of this document in the "Running the Sample" section.

The code will attempt to execute on an available GPU and fallback to the system's CPU if a compatible GPU is not detected. The compilation device is displayed in the output along with elapsed time to render the Mandelbrot image. This helps compare different offload implementations based on the complexity of the computation.

Running the sample in the DevCloud

1. Open a terminal on your Linux system.
2. Log in to DevCloud.

    Code:

    ssh devcloud

3. Download the samples.

   Code:

   git clone https://github.com/oneapi-src/oneAPI-samples.git

4. Change directories to the ISO3DFD OpenMP Offload sample directory.

   Code:

   cd~/oneAPI-samples/DirectProgramming/DPC++/CombinationalLogic/mandelbrot

Build and run the sample in batch mode

The following describes the process of submitting build and run jobs to PBS. A job is a script that is submitted to PBS through the qsub utility. By default, the qsub utility does not inherit the current environment variables or your current working directory. For this reason, it is necessary to submit jobs as scripts that handle the setup of the environment variables. In order to address the working directory issue, you can either use absolute paths or pass the -d <dir> option to qsub to set the working directory.

Create the Job Scripts

1. Create a build.sh script with your preferred text editor:

   nano build.sh

2. Add this text into the build.sh file:
   source /opt/intel/inteloneapi/setvars.sh > /dev/null 2>&1
   mkdir build
   cd build
   cmake ..
   make

3. Save and close the build.sh file.

4. Create a run.sh script with with your preferred text editor:
   nano run.sh

5. Add this text into the run.sh file:
   source /opt/intel/inteloneapi/setvars.sh > /dev/null 2>&1
   cd build
   make run

Build and run

Jobs submitted in batch mode are placed in a queue waiting for the necessary resources (compute nodes) to become available. The jobs will be executed on a first come basis on the first available node(s) having the requested property or label.

1. Build the sample on a gpu node.

   Code:
   qsub -l nodes=1:gpu:ppn=2 -d . build.sh

2. In order to inspect the job progress, use the qstat utility.

   watch -n 1 qstat -n -1

   Note: The watch -n 1 command is used to run qstat -n -1 and display its results every second.

3. When the build job completes, there will be a build.sh.oXXXXXX file in the directory. After the build job completes, run the sample on a gpu node:

   qsub -l nodes=1:gpu:ppn=2 -d . run.sh

4. When a job terminates, a couple of files are written to the disk:

   <script_name>.sh.e6732, which is the job stderr

   <script_name>.sh.o6756, which is the job stdout

5. Inspect the output of the sample.

   cat run.sh.oXXXX

   You should see output similar to this:

```
Platform Name: Intel(R) OpenCL HD Graphics
  Platform Version: OpenCL 2.1
       Device Name: Intel(R) Gen9 HD Graphics NEO
    Max Work Group: 256
 Max Compute Units: 24

Parallel Mandelbrot set using buffers.
Rendered image output to file: mandelbrot.png (output too large to display in text)
       Serial time: 0.0430331s
     Parallel time: 0.00224131s
Successfully computed Mandelbrot set.
```

6. Remove the stdout and stderr files and clean-up the project files.

   rm build.sh.*; rm run.sh.*; make clean

7. Disconnect from the Intel DevCloud.

   exit

So we can run several sample programs are available for you to try, many of which can be compiled and run in a similar fashion to this sample. Experiment with running the various samples on different kinds of compute nodes or adjust their source code to experiment with different workloads.

Conclusion:

Also Intel #oneAPI HPC toolkit can be run in integration with Cloud Plactofrm like GCP which is really interesting and can be deployed using Creating Intel Select Solution HPC clusters

Some testimonials about the toolkit :

“Intel's oneAPI toolkit has demonstrated powerful performance and good compatibility in GeoEast software applications, and has provided us with important help in the further exploration of heterogeneous computing.” – BGP Inc.

#oneAPI