Explore GPU Acceleration in the Intel® DevCloud

Alberto Villarreal, Software Apps Engineer, Intel Corporation
Anant Sinha, Software Apps Engineer, Intel Corporation
Our Speakers

Anant Sinha (Speaker)
Software Applications Engineer
Anant’s current work is focused on working with developers to let their deep learning and machine learning applications take advantage of Intel architectures. Anant joined Intel in 2018. He holds a Bachelors in Computer Science from BITS Pilani, Masters of Engineering in Computer Science from Cornell University and Masters of Science in Computer Science from University of California, Riverside.

Alberto Villarreal (Speaker)
Software Applications Engineer
Alberto works on performance analysis and optimization on current/future Intel architectures. His current work is focused on working with developers to let their applications take advantage of parallel architectures. Alberto joined Intel in 2016. He holds an MSc in Mathematics and Computer Science and an MSc in Geophysics (Colorado School of Mines, USA).

Meghana Rao (Host)
Artificial Intelligence Developer Evangelist
Meghana works closely with universities and developers in evangelizing Intel’s AI portfolio and solutions, helping them understand Machine Learning and Deep Learning concepts. She has a Bachelor’s degree in Computer Science and Engineering and a Master’s degree in Engineering and Technology Management.
Notices and Disclaimers

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.

All product plans and roadmaps are subject to change without notice.

Intel technologies may require enabled hardware, software or service activation.

Results have been estimated or simulated.

No product or component can be absolutely secure.

Intel does not control or audit third-party data. You should consult other sources to evaluate accuracy.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries. Other names and brands may be claimed as the property of others. © Intel Corporation.
Agenda

▪ oneAPI Overview
▪ Using the Intel® DevCloud to try oneAPI
▪ Overview of Intel® Advisor
▪ Demo
▪ Overview of Intel® VTune™ Profiler
▪ Demo
▪ Summary
Programming Challenges for Multiple Architectures

Growth in specialized workloads

Variety of data-centric hardware required

Separate programming models and toolchains for each architecture

Software development complexity limits freedom of architectural choice
Introducing oneAPI

Cross-architecture programming that delivers freedom of choice

Based on industry standards and open specifications

Exposes and exploits cutting-edge performance features of latest hardware

Compatible with existing high-performance languages and programming models including C++, OpenMP, Fortran, and MPI
oneAPI Industry Initiative

Alternative to Single-vendor Solution

A standards-based cross-architecture language, DPC++, based on C++ and SYCL

Powerful APIs designed for acceleration of key domain-specific functions

Low-level hardware interface to provide a hardware abstraction layer to vendors

Open standard to promote community and industry support

Enables code reuse across architectures and vendors

The productive, smart path to freedom for accelerated computing from the economic and technical burdens of proprietary programming models

Visit oneapi.com for more details
**Data Parallel C++ (DPC++)**

**DPC++ = ISO C++ and Khronos SYCL and community extensions**

- Based on modern C++
  - C++ productivity benefits and familiar constructs
- DPC++ is an extension to SYCL, leveraging additional features like:
  - Unified shared memory (USM)
  - ND-range subgroups
  - Ordered queue
DPC++ Extends SYCL 1.2.1

- **Enhance Productivity**
  - Simple things should be simple to express
  - Reduce verbosity and programmer burden

- **Enhance Performance**
  - Give programmers control over program execution
  - Enable hardware-specific features

- **DPC++**: Fast-moving open collaboration feeding into the SYCL standard
  - Open source implementation with goal of upstream LLVM
  - DPC++ extensions aim to become core SYCL or Khronos extensions
Complete DPC++ Program

- Single source
  - Host code and heterogeneous accelerator kernels can be mixed in same source files

- Familiar C++
  - Library constructs add functionality, such as:

<table>
<thead>
<tr>
<th>Construct</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queue</td>
<td>Work targeting</td>
</tr>
<tr>
<td>Buffer</td>
<td>Data management</td>
</tr>
<tr>
<td>parallel_for</td>
<td>Parallelism</td>
</tr>
</tbody>
</table>

```c++
#include <CL/sycl.hpp>
#include <iostream>
constexpr int num = 16;
using namespace cl::sycl;

int main() {
    auto R = range<1>{num};
    buffer<int> A{R};
    queue{}.submit([&](handler& h) {
        auto out = A.get_access<access::mode::write>(h);
        h.parallel_for(R, [=](id<1> idx) {
            out[idx] = idx[0];
        });
    });

    auto result = A.get_access<access::mode::read>(h);
    for (int i = 0; i < num; ++i)
        std::cout << result[i] << "\n";
    return 0;
}
```
Defining Work to Run on an Accelerator Device

- Kernels can be specified in multiple ways:
  - C++ Lambdas
  - Often thin wrapper calling a function
  - Functors
  - Interop

- Kernels are submitted to queues:
  - `parallel_for`
  - `single_task`
  - `parallel_for_work_group`

```cpp
#include <CL/sycl.hpp>
#include <iostream>
constexpr int num=16;
using namespace cl::sycl;

int main() {
  auto R = range<1>{ num };
  buffer<int> A{ R };
  queue{}.submit([&](handler& h) {
    auto out = A.get_access<access::mode::write>(h);
    h.parallel_for(R, [=](id<1> idx) {
      out[idx] = idx[0];
    });
  });

  auto result = A.get_access<access::mode::read>();
  for (int i=0; i<num; ++i)
    std::cout << result[i] << "\n";
  return 0;
}```
Kernel-based Model

- Kernel
  - Code that executes on an accelerator, typically many times/instances per kernel invocation (across an ND-range)

- Kernels clearly identifiable in code
  - Small number of classes can define a kernel (e.g. parallel_for)

- Developer specifies where kernels will run
  - Varying levels of control
Execution Model in Kernels

- Data Parallelism is expressed using ND-Ranges
  - Total Work = # Work-groups × # Work-items per Work-group
  - Bottom-up, hierarchical single program multiple data (SPMD) model
Intel® DevCloud for oneAPI

- A cloud-based sandbox to develop, test, and run workloads across a range of Intel® CPUs, GPUs, and FPGAs using Intel® oneAPI software
- For customers focused on data-centric workloads across a range of Intel® CPUs, GPUs, and FPGAs

No downloads | No hardware acquisition | No installation | No set-up & configuration

Get up & running in seconds!

software.intel.com/devcloud/oneapi
Getting Started
Intel® DevCloud for oneAPI

- Sign up for an account and get access to the DevCloud environment immediately
- Learn how to use Intel® oneAPI toolkits, compilers, performance libraries, and analysis tools PLUS Data Parallel C++ (DPC++)
- Test code and workloads on current and emerging Intel® hardware and software

Included Toolkits
- Intel® oneAPI Base Toolkit
- Intel® oneAPI HPC Toolkit
- Intel® AI Analytics Toolkit
- Intel® oneAPI Rendering Toolkit
- Intel® oneAPI DL Framework Developer Toolkit
- Intel® Distribution of OpenVINO™ Toolkit
- + more

Hardware
- CPU: Intel® Xeon® Scalable processors: 6128 & 8256
- GPU: Intel® Xeon® E-2176 P630 processors (with Intel® Graphics Technology/GPU)
- FPGA: Intel® Arria® 10 FPGA, Intel® Stratix® 10 FPGA

Featured Beta Tools & Libraries
- Intel® oneAPI DPC++/C++ Compiler
- Intel oneAPI DPC++ Library
- Intel® C++ & Intel® Fortran Compilers
- Intel® oneAPI Math Kernel Library
- Intel® oneAPI Data Analytics Library
- Intel® Distribution for Python
- Intel® VTune™ Profiler & Intel® Advisor
- Intel® FPGA Add-on for oneAPI Base Toolkit
- + many more...
How It Works

Intel® DevCloud for oneAPI

How It Works

Development Environment
- 220 GB of file storage
- 192 GB of RAM
- Up to 24 hours of continuous workload execution times
- Free 120-day access; account extensions upon request

Quick How-to Resources
- Videos
- Developer guides

https://devcloud.intel.com/oneapi/
Intel Analysis Tools for GPU Compute Analysis

**Intel® Advisor**

- **Offload Advisor**
  - Identify high-impact opportunities to offload
  - Detect bottlenecks and key bounding factors
  - Get your code ready even before you have the hardware by modeling performance, headroom and bottlenecks

- **Roofline Analysis**
  - See performance headroom against hardware limitations
  - Determine performance optimization strategy by identifying bottlenecks and which optimizations will payoff the most
  - Visualize optimization progress

- **Flow Graph Analyzer**
  - Visualize your CPU/GPU code and get recommendations for the CPU device

**Intel® VTune™ Profiler**

- **Offload Performance Tuning**
  - Explore code execution on various CPU and GPU cores on your platform
  - Correlate CPU and GPU activity
  - Identify whether your application is GPU- or CPU-bound

- **GPU Compute/Media Hotspots**
  - Analyze the most time-consuming GPU kernels; characterize GPU usage based on GPU hardware metrics
  - GPU code performance at the source-line level and kernel assembly level
Intel® Advisor
Using Intel® Advisor to Increase Performance

- Use Offload Advisor to find kernels to offload
- Optimize your kernels

1. Intel® Advisor – Offload Advisor
2. Choose
   - Kernel Style: Data Parallel C++
   - Directive Style: OpenMP
3. Intel® Advisor – GPU Roofline and Flow Graph analyzer
4. Optimized Code

- Existing CUDA code
- New Code
- Existing C++
- Existing Fortran or C
- Existing OpenCL™ Applications
- Intel® DPC++ Compatibility Tool
Tools to Maximize Heterogeneous Hardware Utilization

- **Offload Advisor**: Design offload strategy and model performance on GPU.
- **Roofline Analysis**: Optimize your application for memory and compute.
- **Vectorization Optimization**: Enable more vector parallelism and improve its efficiency.
- **Thread Prototyping**: Model, tune, and test multiple threading designs.
- **Build Heterogeneous Algorithms**: Create and analyze data flow and dependency computation graphs.

Learn More: software.intel.com/advisor
Starting from a optimized binary (running on CPU):

- Helps define which sections of the code should run on a given accelerator
- Provides performance projection on accelerators
Will Offload Increase Performance?

What is workload bounded by

Good Candidates to offload

Bad Candidates

Offloads bounded by

Top offloaded

Top non offloaded

Technical Webinar – Explore GPU Acceleration in the Intel® DevCloud
Compare Acceleration on Different GPUs

Gen9 – Not profitable to offload kernel

Gen11 – 1.6x speedup

Not profitable: Computation Time is high despite the full use of Target Device capabilities
What is a Roofline Chart?

A Roofline Chart plots application performance against hardware limitations

- Where are the bottlenecks?
- How much performance is being left on the table?
- What are the next steps?

Chart on right shows a CPU Roofline chart

Roofline first proposed by University of California at Berkeley: *Roofline: An Insightful Visual Performance Model for Multicore Architectures*, 2009

Cache-aware variant proposed by University of Lisbon: *Cache-Aware Roofline Model: Upgrading the Loft*, 2013
Find Effective Optimization Strategies

Intel® Advisor - GPU Roofline

Configure levels to display

Shows performance headroom for each loop

Likely bottlenecks

Suggests optimization next steps

Technical Webinar – Explore GPU Acceleration in the Intel® DevCloud
DEMO
Intel® VTune™ Profiler
Two GPU Analysis Types

- **Is the offload efficient?** - GPU Offload
  - Find inefficiencies in offload
  - Identify if you are CPU- or GPU-bound
  - Find the kernel to optimize first
  - Correlate CPU and GPU activity
  - Analyze DMA packet execution

- **Is the GPU kernel efficient?** - GPU Compute/Media Hotspots
  - Identify what limits the performance of the kernel
  - GPU source/instruction level profiling
  - Find memory latency or inefficient kernel algorithms
GPU Offload

- Simply following the sections on the Summary page
- Tuning methodology on top of HW metrics

**GPU Usage**: 0.6% 
Use this section to understand whether the GPU was utilized properly and which of the engines were utilized. Identify the amount of gaps in the GPU utilization that potentially could be loaded with some work. This metric is calculated for the engines that had at least one piece of work scheduled to them.

**GPU Engine / Packet Type** | **GPU Time** | **Usage (%)**
--- | --- | ---
Render and GPGPU | 1.146s | 0.6%
Unknown | 0.888s | 0.5%
GHAL3D | 0.249s | 0.1%
OpenCL | 0.009s | 0.0%

**EU Array Stalled/Idle**: 94.4% of Elapsed time
Analyze the average value of EU Array Stalled/Idle metric and identify why EUs were waiting for resources instead of doing computations. This metric is critical for compute-bound applications. Explore typical reasons for this kind of inefficiency listed below.

1. **GPU L3 Bandwidth Bound**: 0.5% of peak value
2. **DRAM Bandwidth Bound**: 0.0% of Elapsed time
3. **Occupancy**: 25.8% of peak value
   - Identify too large or too small computing tasks with low occupancy that make the EU array idle while waiting for the scheduler. Note that frequent SLM accesses and barriers may affect the maximum possible occupancy.

4. **Hottest GPU Computing Tasks with Low Occupancy**
5. **Sampler Busy**: 40.6% of peak value
See Profile Data on Your DPC++ Source

```c_cpp
184
185    
186    
187    T c = {}; 
188    for (size_t k = 0; k < w; k++)
189    {
190        c += ra[i][k] * rb[k][j];
191    }
192
```

Performance metrics at the DPC++ statement level

GPU assembly available for compute kernels
Timeline Correlate GPU and CPU Activity

- Identify too much or too little kernel activity
- Correlate GPU activity with kernels and threads

<table>
<thead>
<tr>
<th>Thread</th>
<th>matrix_multiply (TID: 8559)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GPU Execution Units</strong></td>
<td></td>
</tr>
<tr>
<td><strong>GPU Computing Threads</strong></td>
<td></td>
</tr>
<tr>
<td><strong>GPU EU Instructions</strong></td>
<td></td>
</tr>
<tr>
<td><strong>GPU Memory Access</strong></td>
<td>16257.376</td>
</tr>
<tr>
<td><strong>L3 &lt;-&gt; GT1 Total Band., 8697.855</strong></td>
<td></td>
</tr>
<tr>
<td><strong>GPU Texture Sampler</strong></td>
<td></td>
</tr>
<tr>
<td><strong>GPU L3 Cache Bandwidth an</strong>...</td>
<td></td>
</tr>
<tr>
<td><strong>GPU Shared Local Memory ...</strong></td>
<td></td>
</tr>
<tr>
<td><strong>GPU Utilization</strong></td>
<td></td>
</tr>
</tbody>
</table>

Scale Markers:
- VSync
- Running
- GPU Engines Usage (Render and GPGPU)
- User Tasks
- Computing Task
- GPU Execution Units
- EU Arrays (Active, Idle, Stalled)
- GPU Computing Threads (Computing Threads Start)
- GPU EU Instructions
- FPU Utilization

VTune is a trademark of Intel Corporation or its subsidiaries.
GPU Compute/Media Hotspots
Tune Inefficient Kernel Algorithms

- Analyze GPU Kernel Execution
  - Find memory latency or inefficient kernel algorithms
  - See the hotspot on the OpenCL™ or DPC++ source & assembly code

- Analyze DMA packet execution
  - Packet Queue Depth histogram
  - Packet Duration histogram

- GPU-side call stacks
Summary and Call-to-Action

- Get Started – test code & workloads using the Intel® DevCloud for oneAPI
- Diverse workloads are driving the need for heterogeneous compute architectures
- oneAPI unifies & simplifies programming of heterogeneous architectures delivering developer productivity & performance
- oneAPI is an open industry initiative & an Intel reference product
- oneAPI is interoperable with existing node & cluster programming model

software.intel.com/devcloud/oneapi
Get the Most from Your Code Today with Intel® Tech.Decoded

Visit TechDecoded.intel.io to learn how to put key optimization strategies into practice with Intel development tools.

- **Big Picture Videos**
Discover Intel's vision for key development areas.

- **Essential Webinars**
Gain strategies, practices and tools to optimize application and solution performance.

- **Quick Hit How-To Videos**
Learn how to do specific programming tasks using Intel® tools.

**TOPICS:**

- Visual Computing
- Code Modernization
- Systems & IoT
- Data Science
- Data Center & Cloud
More Resources

- Intel oneAPI
- Intel® oneAPI DPC++ Compiler\(^{(Beta)}\)
- Intel® Advisor\(^{(Beta)}\)
  - Product page – overview, features, FAQs, support...
  - User Guide
    - Offload Advisor User Guide
  - Cookbooks – Recipes to common problems
    - Design and Optimize Offload – Optimize GPU usage with Offload Advisor and GPU Roofline
- Intel® VTune™ Profiler\(^{(Beta)}\) – performance profiler
  - VTune cookbooks recipe to avoid frequent DRAM accesses