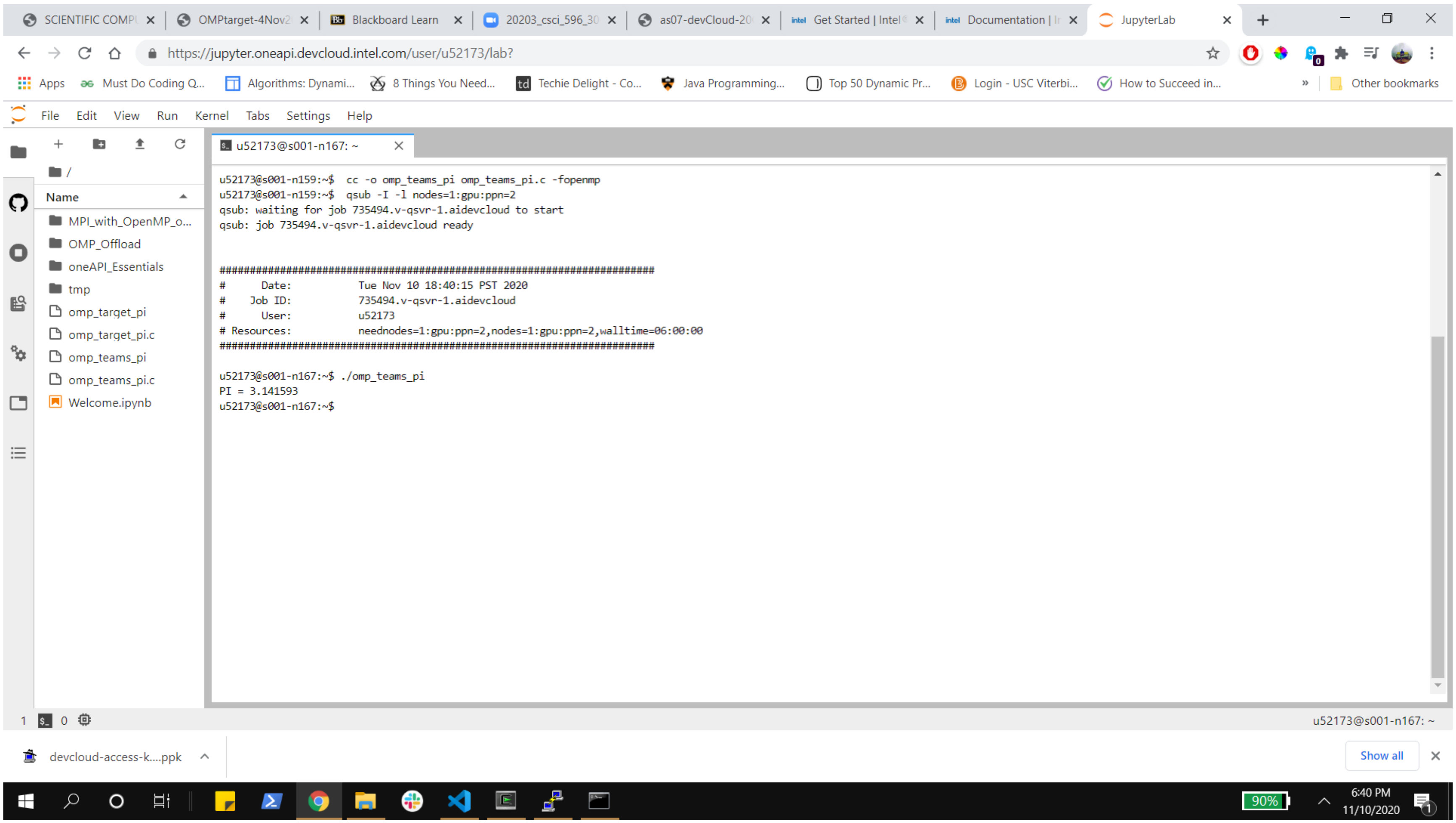
On Assignment 7

Q. **How can a Windows user transfer files to devCloud?**

A. **While the devCloud instruction recommends WinSCP, various other approaches have been taken by your classmates:**

> *Via* discovery@usc.edu

> *Via* devCloud Jupyter notebook terminal



On Assignment 7

Q. **Why hands-on training on devCloud?**

A. **To stay ahead: Open programming revolution has just started for heterogeneous parallel computing, but new languages (OpenMP target & DPC++) are not yet available on most systems like discovery; on devCloud, you have four months of free access to those languages + valuable accelerator-migration tools like Advisor.**

See http://cacs.usc.edu/education/cs596/s_exploregpuaccelerationintheinteldevcloud1604520259872.pdf

Top offloaded

Location	Speed Up	Bounded By	Data Transfer
[loop in iso_3dfdompparallel@52 at iso-3dfd_parallel.cc:53]	8.94x CPU 13.75s GPU 1.54s	LLC_BW	<0.01MB

Good Candidates to offload

On Assignment 7

Follow easy Jupyter notebook tutorials to learn more:
https://devcloud.intel.com/oneapi/get_started/

Intel® oneAPI HPC Toolkit
Deliver fast C++, Fortran, OpenMP*, and MPI applications that scale.

Hide Details

The toolkit includes:

- Intel® C++ Compiler
- Intel® Cluster Checker
- Intel® Fortan Compiler
- Intel® Inspector
- Intel® MPI Library
- Intel® Trace Analyzer and Collector

Resources:

Getting Started Guide
Deliver fast C++, Fortran, OpenMP, and MPI applications that scale.
View Guide

Developer Training Modules
Learn more about the HPC Toolkit with Jupyter Notebook* Training Modules.
View Training Modules

Sample Applications
Test out real-world solutions on hosted Intel® hardware with these sample applications.
Try Sample Applications

Module 0
Introduction to JupyterLab* and Notebooks.
Learn to use Jupyter notebooks to modify and run code as part of learning exercises.
Try it in Jupyter

Module 1
Introduction to DPC++
- Articulate how oneAPI can help to solve the challenges of programming in a heterogeneous world.
- Use oneAPI solutions to enable your workflows.
- Understand the DPC++ language and programming model.
- Become familiar with using Jupyter notebooks for training throughout the course.

Try it in Jupyter

Module 2
DPC++ Program Structure
- Articulate the SYCL* fundamental classes.
- Use device selection to offload kernel workloads.
- Decide when to use basic parallel kernels and ND Range Kernels.
- Create a host accessor.
- Build a sample DPC++ application through hands-on lab exercises.

Try it in Jupyter

Module 3
DPC++ Unified Shared Memory
- Use new DPC++ features like Unified Shared Memory (USM) to simplify programming.
- Understand implicit and explicit ways of moving memory using USM.
- Solve data dependency between kernel tasks in an optimal way.

Try it in Jupyter

Module 4
DPC++ Sub-Groups
- Understand advantages of using Sub-groups in DPC++.
- Take advantage of Sub-group collectives in ND-Range kernel implementation.
- Use Sub-group Shuffle operations to avoid explicit memory operations.

Try it in Jupyter

Module 5
Demonstration of Intel® Advisor
- See how Offload Advisor[1] identifies and ranks parallelization opportunities for offload.
- Run Offload Advisor using command line syntax.
- Use performance models and analyze generated reports.

Offload Advisor is a feature of Intel Advisor installed as part of the Intel(R) oneAPI Base Toolkit.

Try it in Jupyter

Module 5
Demonstration of Intel® Advisor
- See how Offload Advisor[1] identifies and ranks parallelization opportunities for offload.
- Run Offload Advisor using command line syntax.
- Use performance models and analyze generated reports.

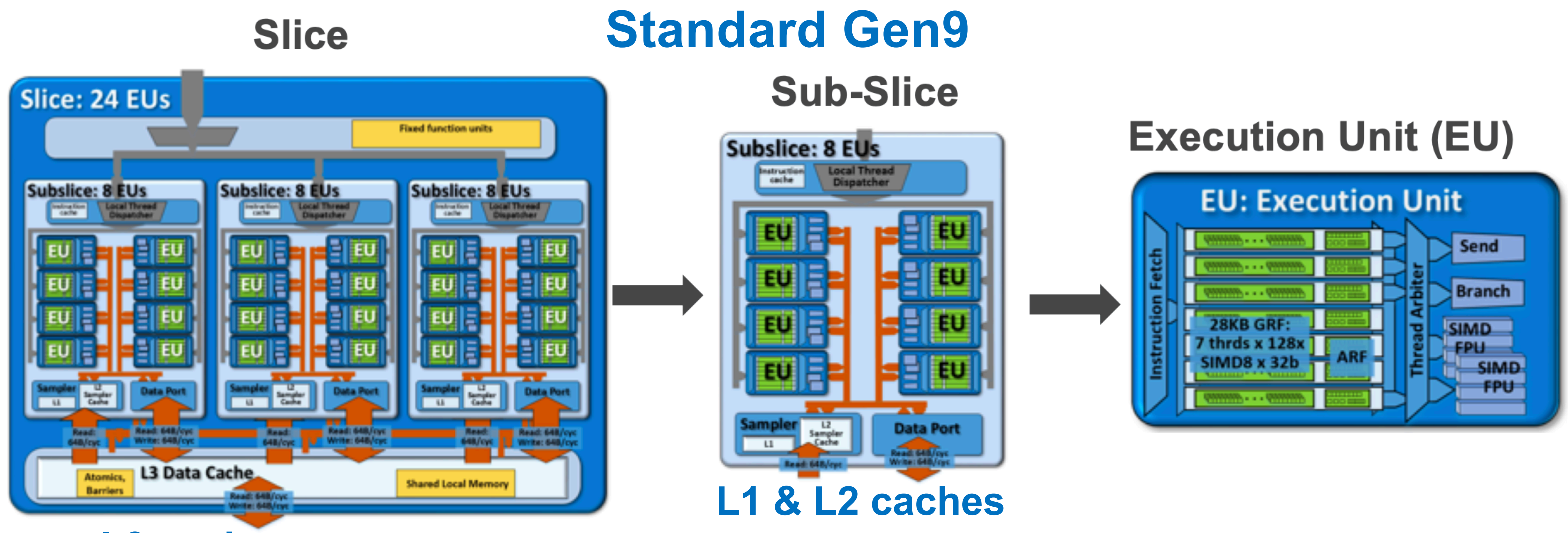
Offload Advisor is a feature of Intel Advisor installed as part of the Intel(R) oneAPI Base Toolkit.

Try it in Jupyter

On Assignment 7

Q. **Why is my** omp_teams_pi.c **slower than** omp_target_pi.c**?**

A. **Number of teams & threads were not chosen specific to the Gen9 GPU on devCloud; while providing additional control over data locality, overhead of** `teams` **construct needs be considerer as well.**



Gen9 (GT4)

Characteristics	Value	Notes
Clock Freq.	1.15 GHz	
Slices	3	
EUs	72	3 slice * 3 sub-slices * 8 EUs
Hardware Threads	504	72 EUs * 7 threads
Concurrent Kernel Instances	16,128	504 threads * SIMD-32
L3 Data Cache Size	1.5 MB	3 slices * 0.5 MB/slice
Max Shared Local Memory	576 KB	3 slice * 3 sub-slices * 64 KB/sub-slice
Last Level Cache Size	8 MB	
eDRAM size	128 MB	
32b float FLOPS	1152 FLOPS/cycle	72 EUs * 2 FPUs * SIMD-4 * (MUL + ADD)
64b float FLOPS	288 FLOPS/cycle	72 EUs * 1 FPU * SIMD-2 * (MUL + ADD)
32b integer IOPS	576 IOPS/cycle	72 EUs * 2 FPUs * SIMD-4