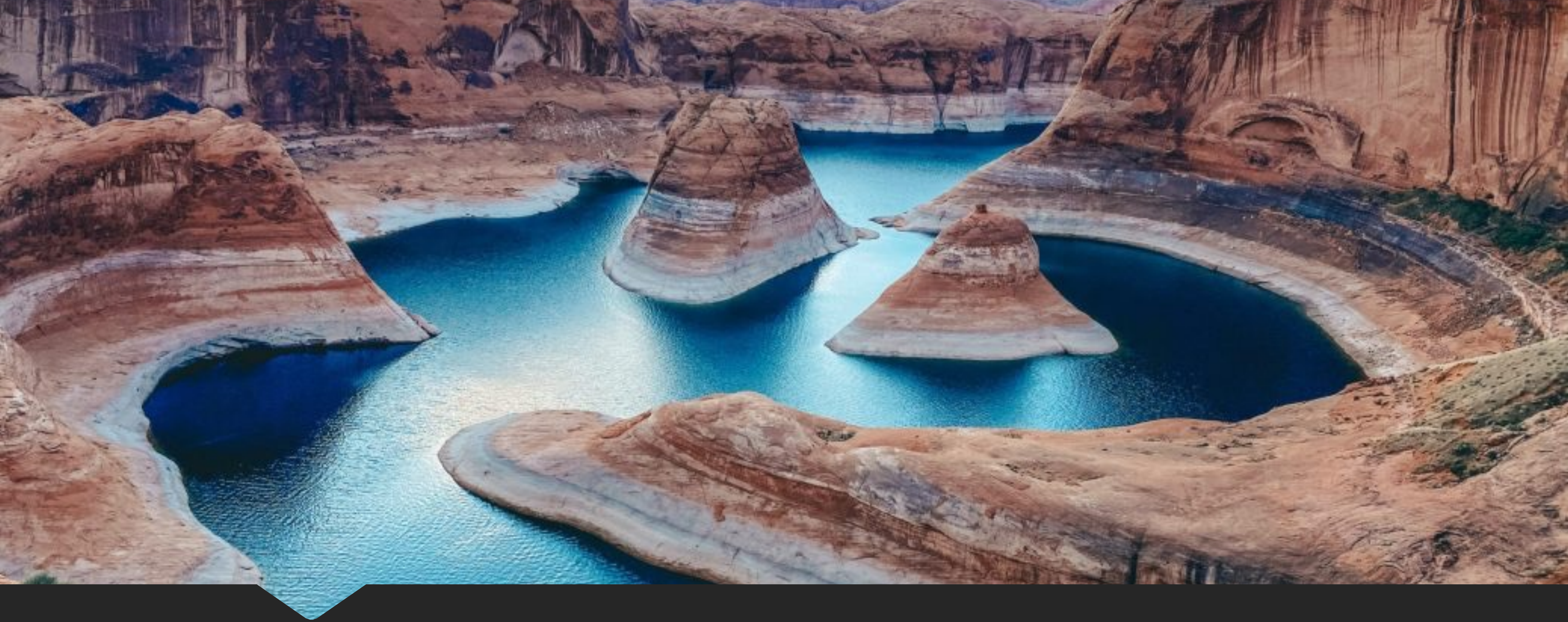
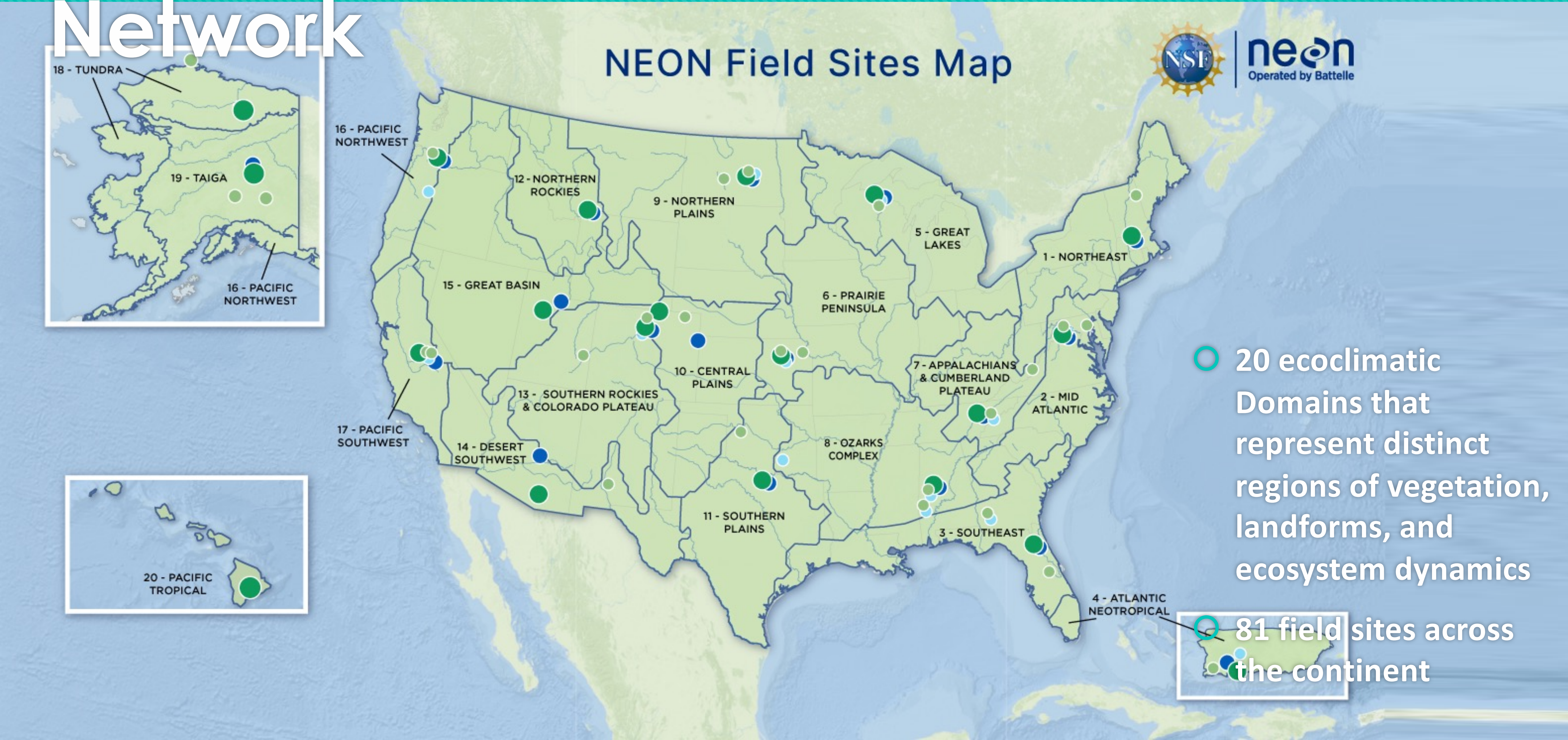


SYCL Support for Continental-Scale Ecological Observations: Scalable and Portable Blending of Massive Image Mosaics

Dr. Steve Petruzza, Dr. Valerio Pascucci

The University of Utah & Lawrence Livermore National Laboratory Intel oneAPI Center of Excellence
Utah State University

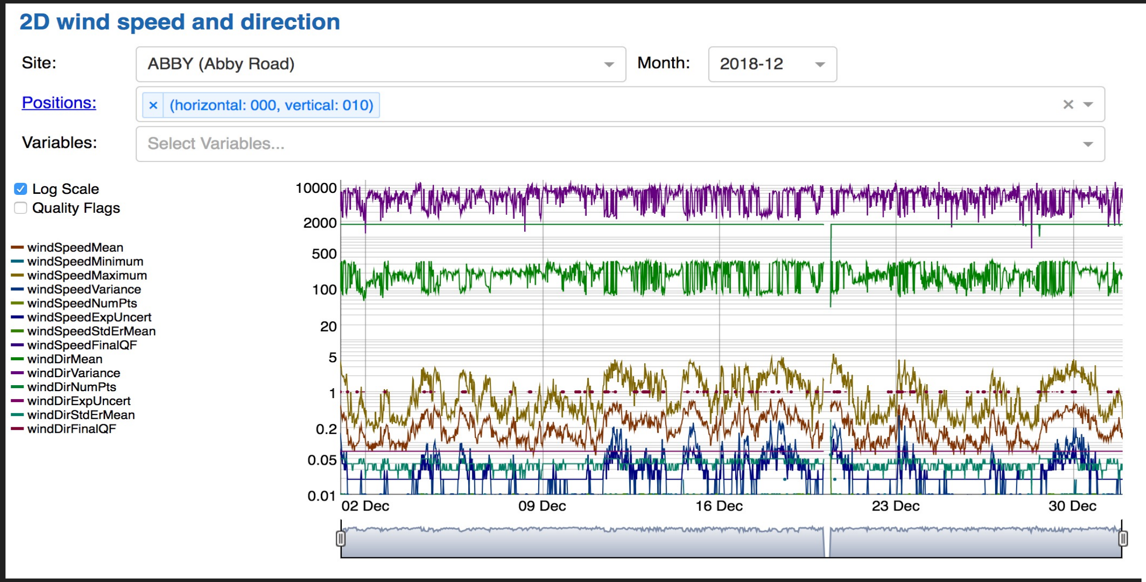
National Ecological Observatory Network

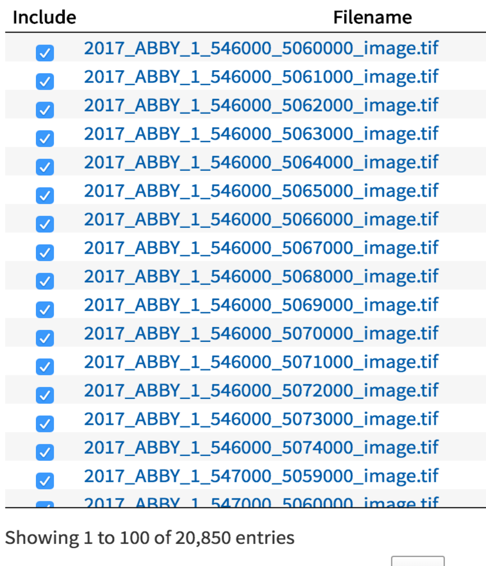


- 20 ecoclimatic Domains that represent distinct regions of vegetation, landforms, and ecosystem dynamics
- 81 field sites across the continent

NEON data

- NEON has a large amount of data that is shared with the community through their **data portal**
- For some data, such as sensor measurements, the portal provides an **interactive** navigation system
- For others, like **Airborne Observation Platforms data**, there is a long list of image files…
- There is a need to present all AOP data interactively, where the users can preview, navigate, and select/access/download the data they need

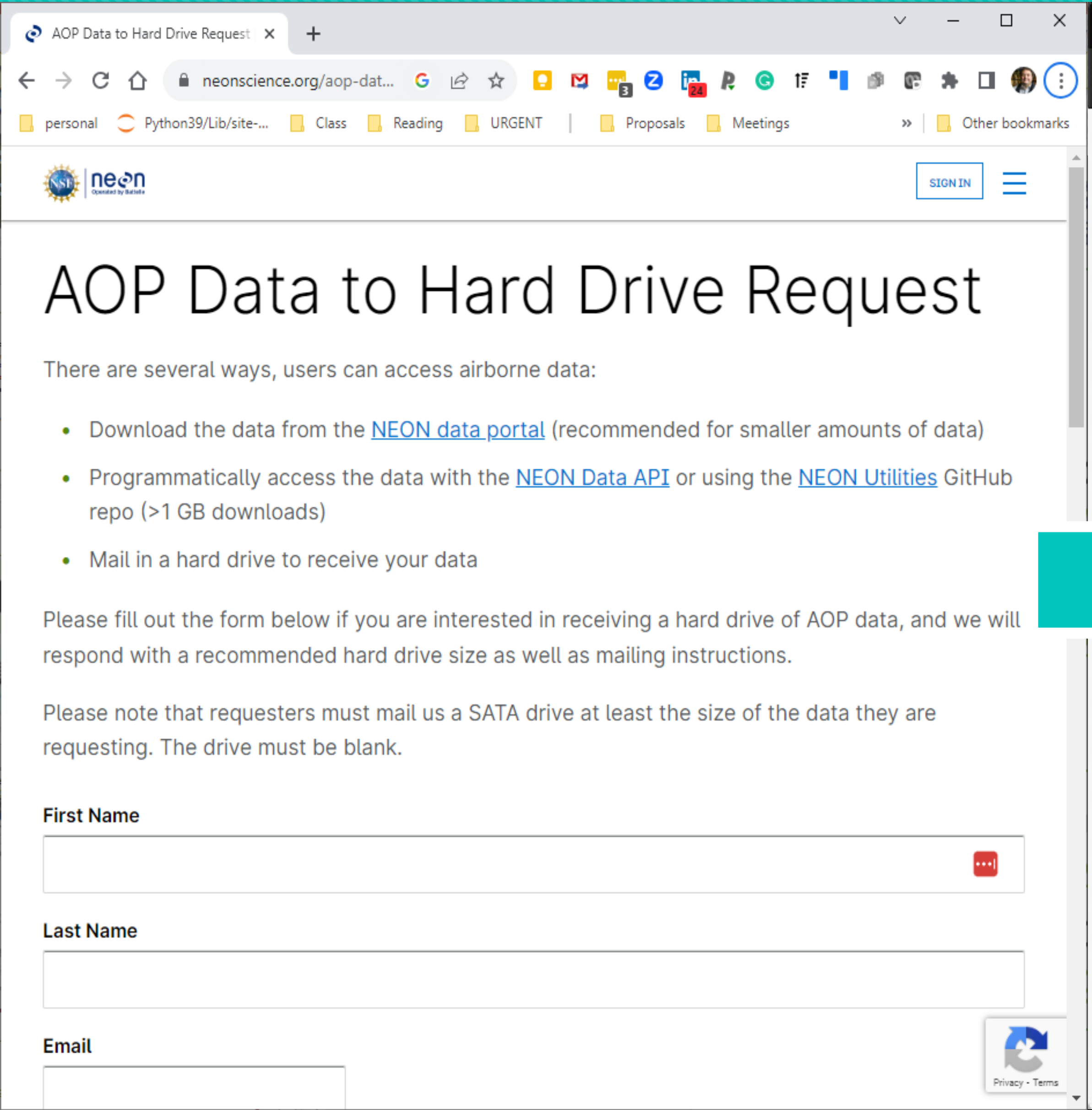




NEON Airborne Operational Platform Data

- NEON – National Ecological Observatory Network
- 59 AOP sites
- 1-2 datasets per site/year
- Dataset size range from 20 to 300GB
- Total data collection 100s TBs

Access to Airborne Remote Sensing Data


AOP Data to Hard Drive Request

neonscience.org/aop-dat...

personal Python39/Lib/site-... Class Reading URGENT Proposals Meetings Other bookmarks

SIGN IN

AOP Data to Hard Drive Request

There are several ways, users can access airborne data:

- Download the data from the NEON data portal (recommended for smaller amounts of data)
- Programmatically access the data with the NEON Data API or using the NEON Utilities GitHub repo (>1 GB downloads)
- Mail in a hard drive to receive your data

Please fill out the form below if you are interested in receiving a hard drive of AOP data, and we will respond with a recommended hard drive size as well as mailing instructions.

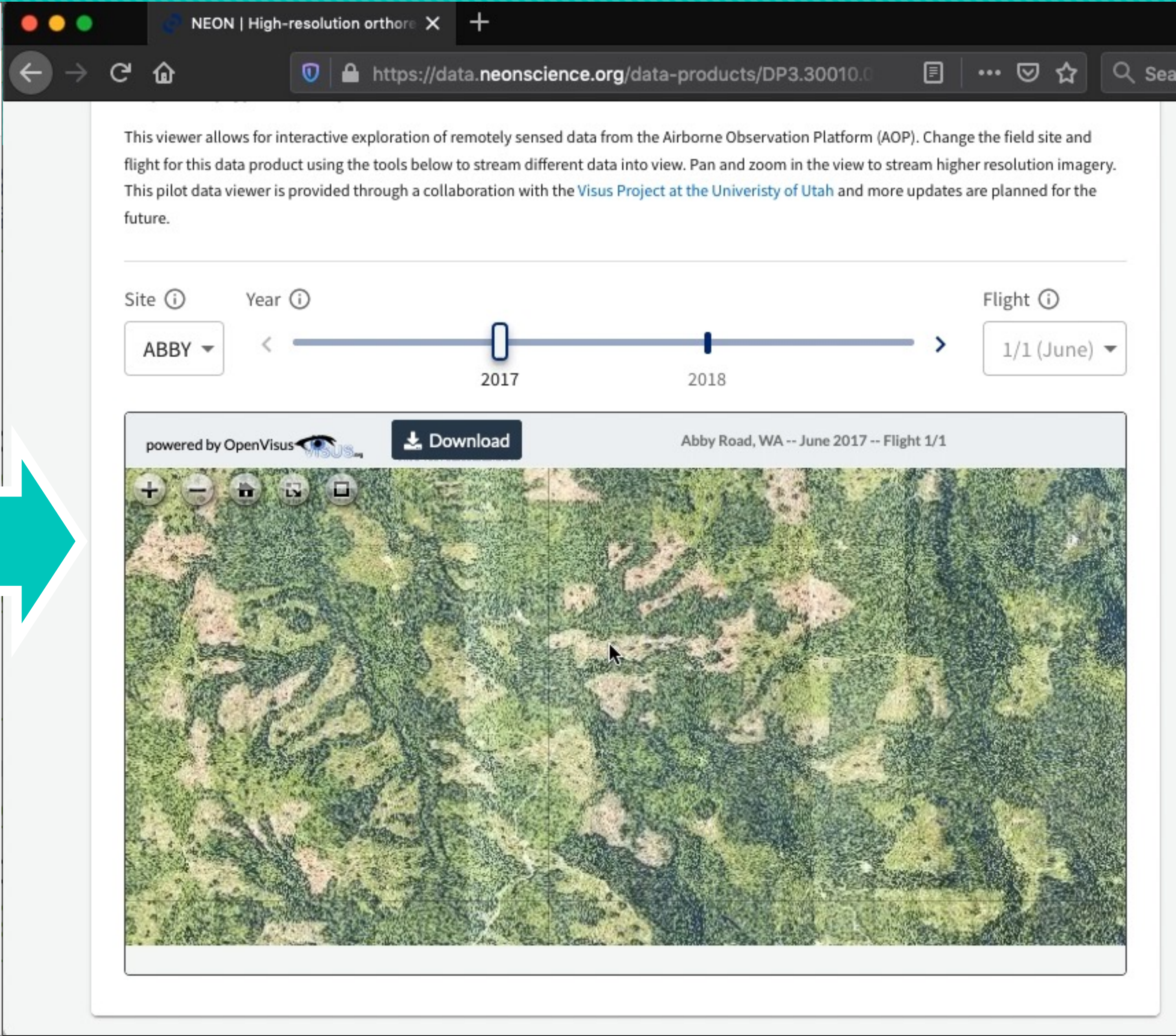
Please note that requesters must mail us a SATA drive at least the size of the data they are requesting. The drive must be blank.

First Name

Last Name

Email

Privacy - Terms


NEON | High-resolution orthore

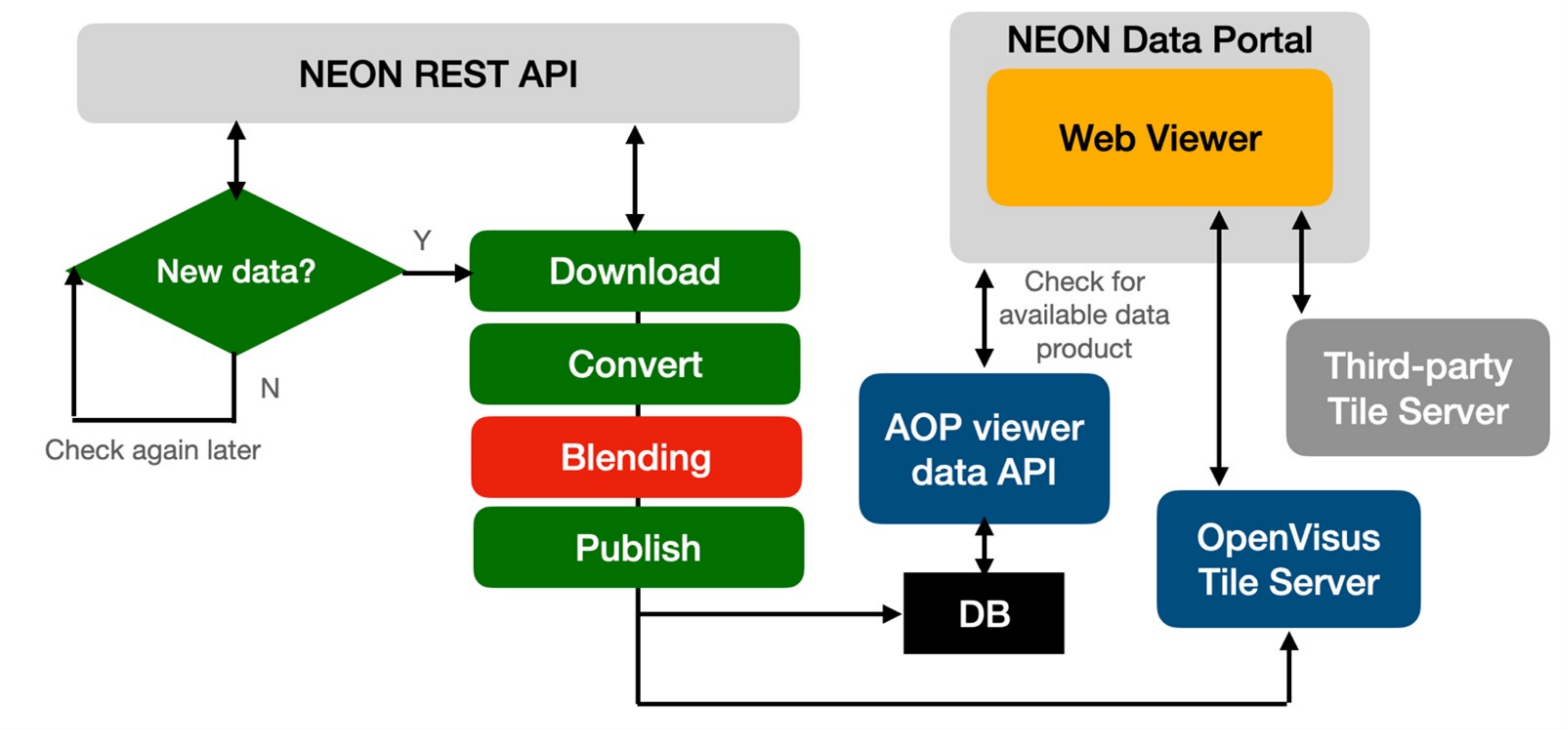
https://data.neonscience.org/data-products/DP3.30010.0

This viewer allows for interactive exploration of remotely sensed data from the Airborne Observation Platform (AOP). Change the field site and flight for this data product using the tools below to stream different data into view. Pan and zoom in the view to stream higher resolution imagery. This pilot data viewer is provided through a collaboration with the Visus Project at the Univeristy of Utah and more updates are planned for the future.

Site: ABBY | Year: 2017, 2018 | Flight: 1/1 (June)

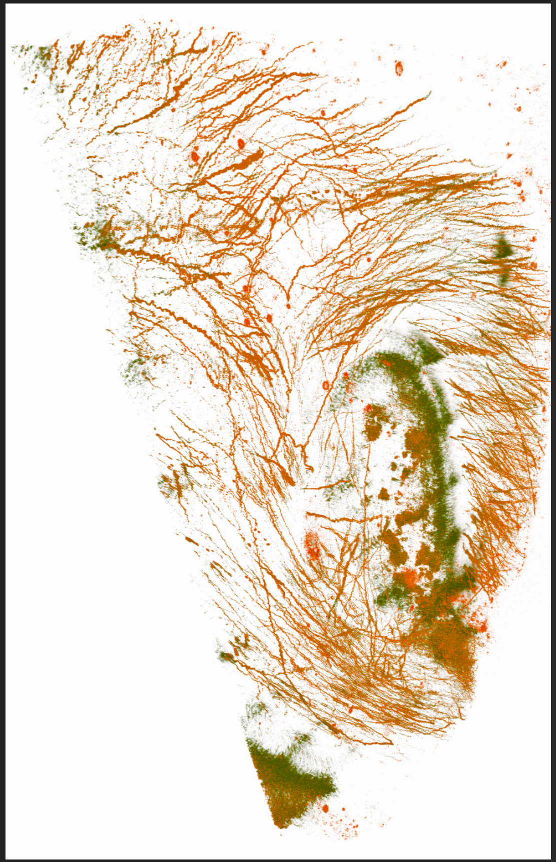
powered by OpenVisus | Download | Abby Road, WA -- June 2017 -- Flight 1/1

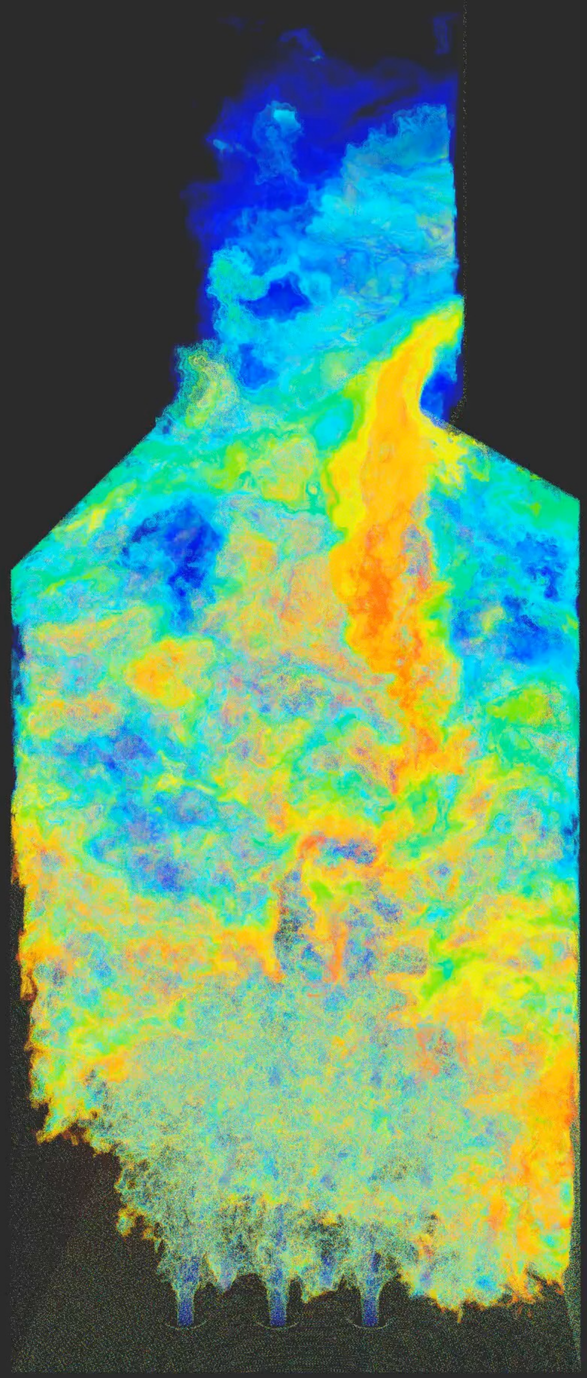
Data analysis and visualization pipeline



OpenVisus streaming visualization powered by Intel OSPRay

- Large scale combustion simulations
- Neuroscience data (600GB microscopy scan of Macaque monkey visual cortex)
- OpenVisus data streaming for large scale simulation
- High quality rendering at high framerate on local workstations using Intel OSPRay





Gradient domain image processing

Gradient domain image processing

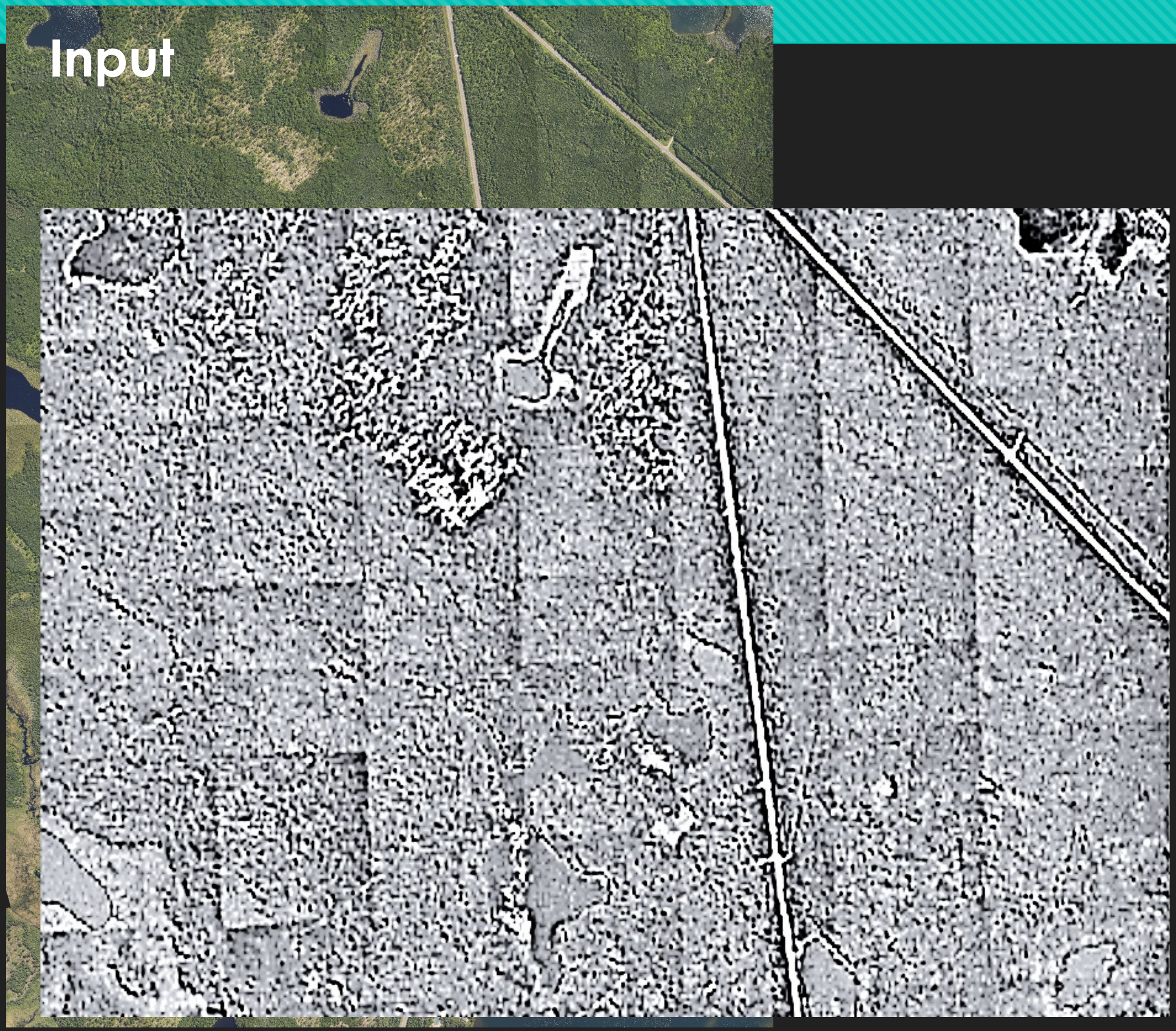
- images processed in the gradient space (not pixel)
- find the closest smooth image to the guiding gradient with a minimal least squared error
- equivalent to solving a 2D Poisson equation
  - direct solutions can be fast using FFT, but with low accuracy
  - or by discretizing the equations into a large linear system that can be solved iteratively (e.g., Conjugate Gradient)
- Requires to build and solve the system in memory

Challenges with NEON AOP data

Remove major artifacts that prevent a scientific investigation

Challenges with NEON AOP data

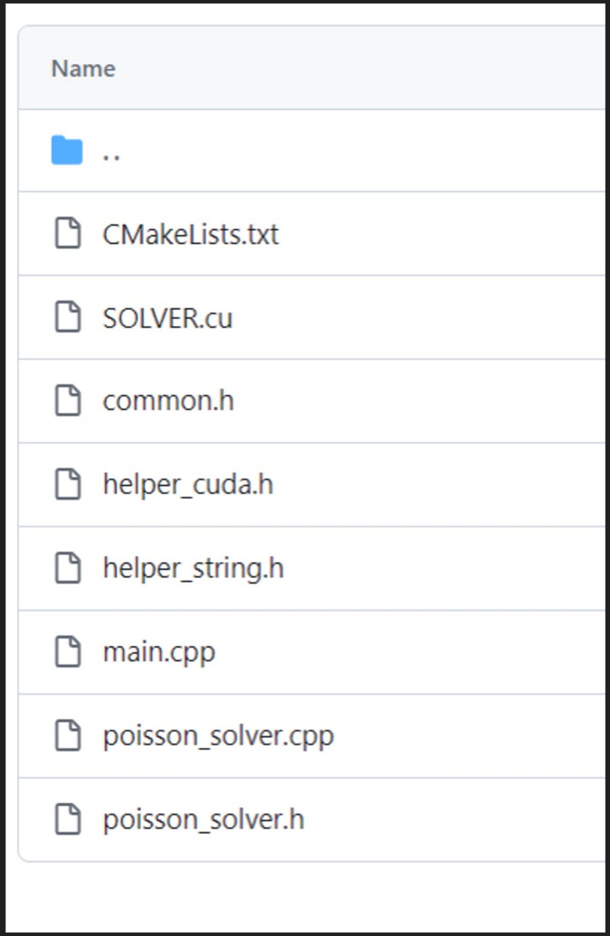
Remove major artifacts that prevent a scientific investigation

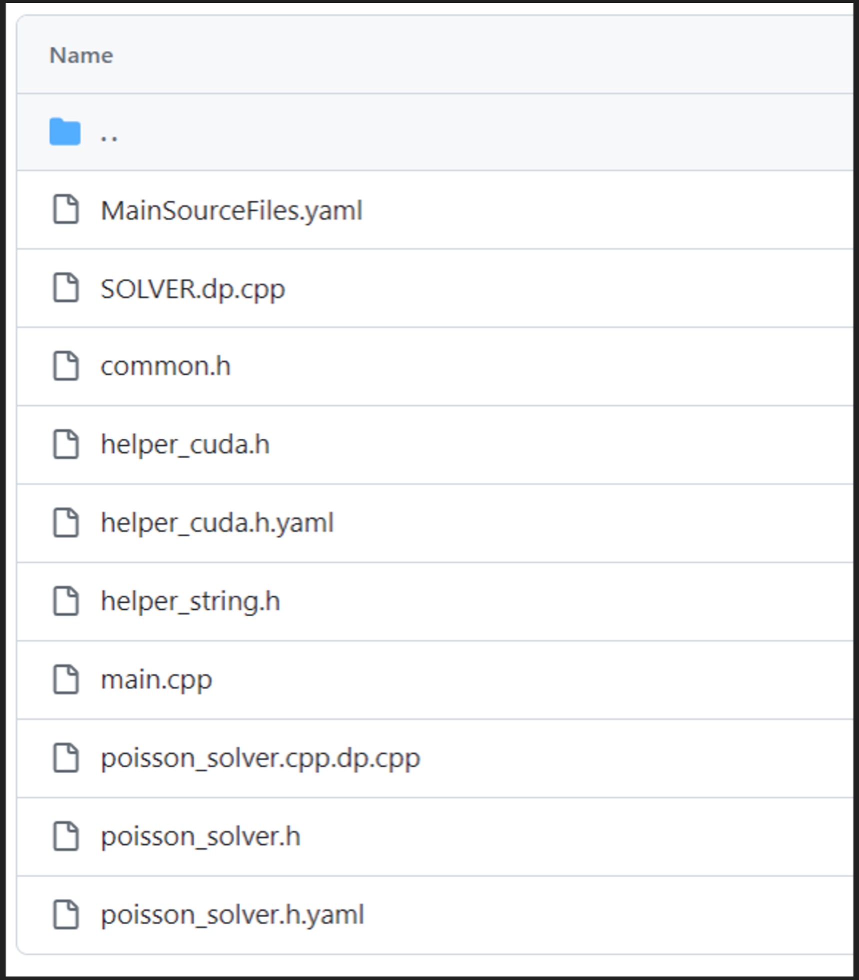


Challenges with NEON AOP data

Remove major artifacts that prevent a scientific investigation

Code conversion from CUDA to SYCL





Code conversion from CUDA to SYCL

poisson_solver / cg-solver / src / SOLVER.cu

spetruzza more testing clean up

Code Blame Executable File · 269 lines (210 loc) · 8.47 KB

```
#include <cuda.h>
#include <cuda_runtime.h>
#include <cuda_runtime_api.h>
#include "helper_cuda.h"
#include <stdio.h>
#include <cublas.h>
#include <cusparse.h>
#include <time.h>
#include "common.h"
#include <iostream>

#ifndef clamp
#define clamp(value,a,b)  (((value)<(a))?(a):(((value)>(b))?(b):(value)))
#endif

__global__ void mult_noshared(int dimx, int dimy, const float3 *x_old, float3 *x, uchar3 *map_data){
        int  i = blockIdx.x * blockDim.x + threadIdx.x;
        int  j = blockIdx.y * blockDim.y + threadIdx.y;
        int idx = j*dimx+i;

        if(i < dimx && j < dimy && map_data[idx].z != 255){
                        int x0 = idx-1;
                        int x1 = idx+1;
                        int y0 = idx-dimx;
                        int y1 = idx+dimx;

                        float3 temp;
```

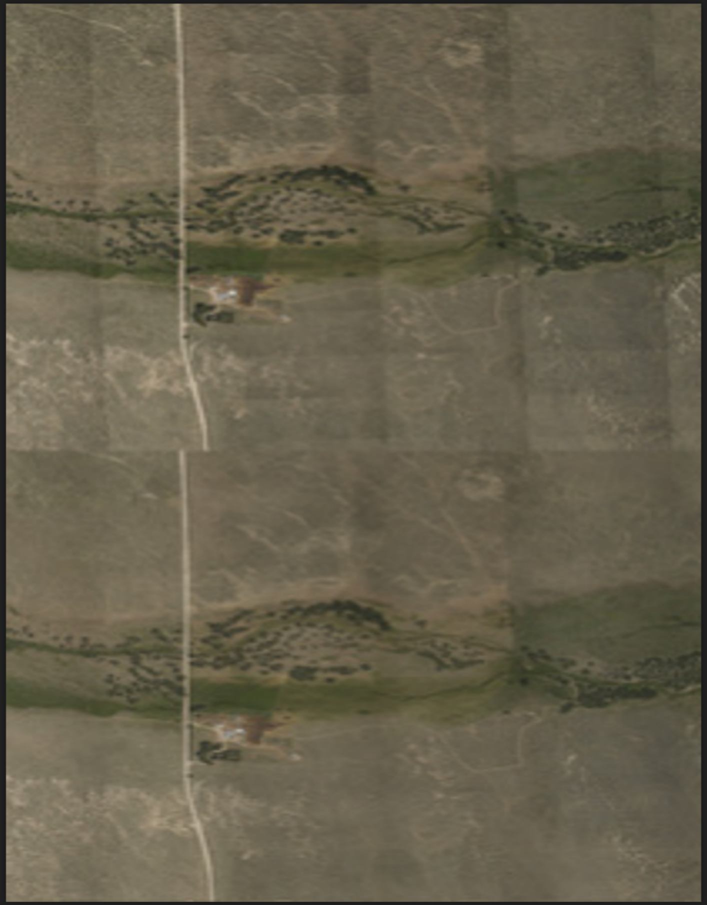
poisson_solver / cg-solver / sycl_converted / SOLVER.dp.cpp

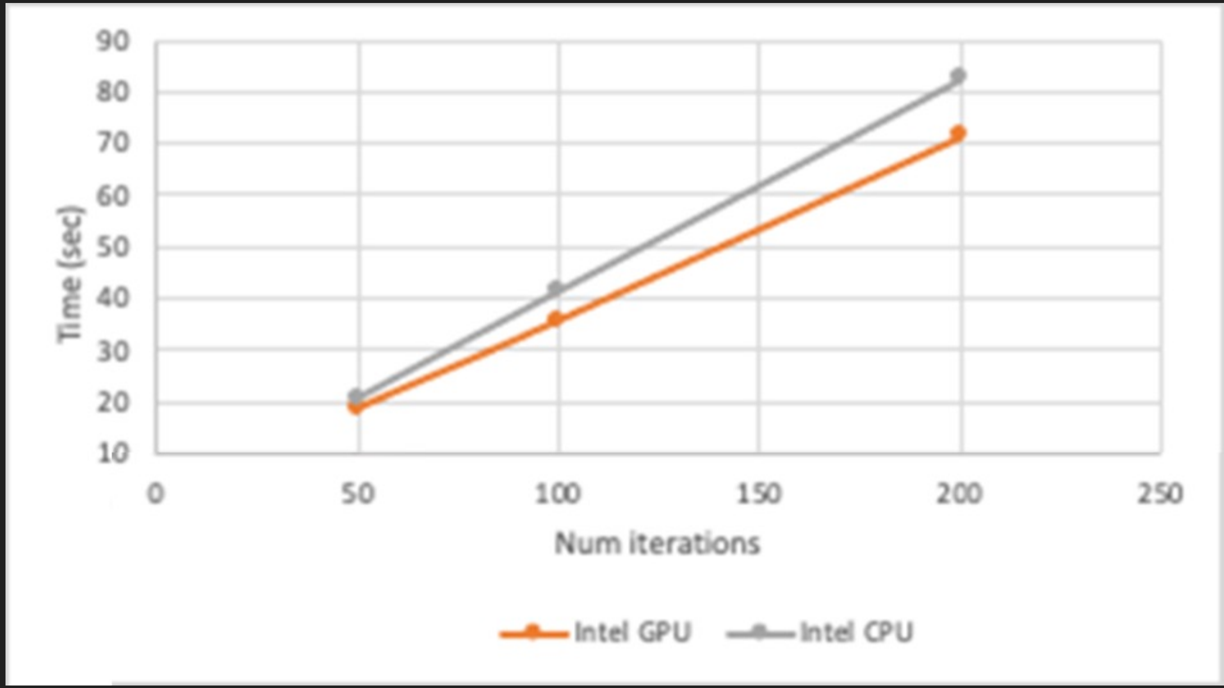
spetruzza add converted sycl code

Code Blame 617 lines (561 loc) · 29.9 KB

```
#include <sycl/sycl.hpp>
#include <dpct/dpct.hpp>
#include "helper_cuda.h"
#include <stdio.h>
#include <dpct/blas_utils.hpp>

#include <dpct/sparse_utils.hpp>

#include <time.h>
#include "common.h"
#include <iostream>

#ifndef clamp
#define clamp(value,a,b)  (((value)<(a))?(a):(((value)>(b))?(b):(value)))
#endif

void mult_noshared(int dimx, int dimy, const sycl::float3 *x_old,
                   sycl::float3 *x, sycl::uchar3 *map_data,
                   const sycl::nd_item<3> &item_ct1) {
        int i = item_ct1.get_group(2) * item_ct1.get_local_range(2) +
                item_ct1.get_local_id(2);
        int j = item_ct1.get_group(1) * item_ct1.get_local_range(1) +
                item_ct1.get_local_id(1);
        int idx = j*dimx+i;

        if (i < dimx && j < dimy && map_data[idx].z() != 255) {
                        int x0 = idx-1;
```

Experimental study: multiple solver iterations





Prior Intel® DevCloud for oneAPI

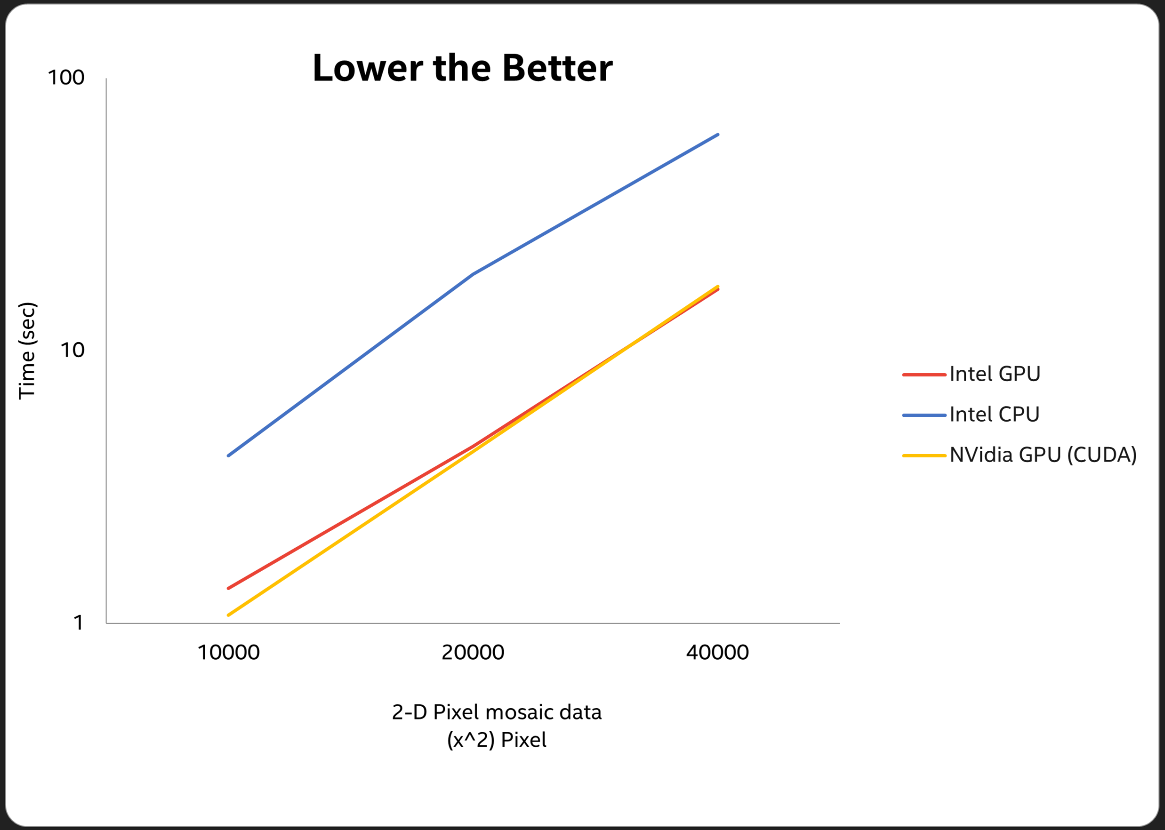
Intel CPU: 11th Gen Intel® Core™ i9-11900KB @ 3.30GHz

Intel GPU: Intel® Xeon® E-2176 P630 processors (2018)

Experimental Study: Increasing problem size on different computing platforms

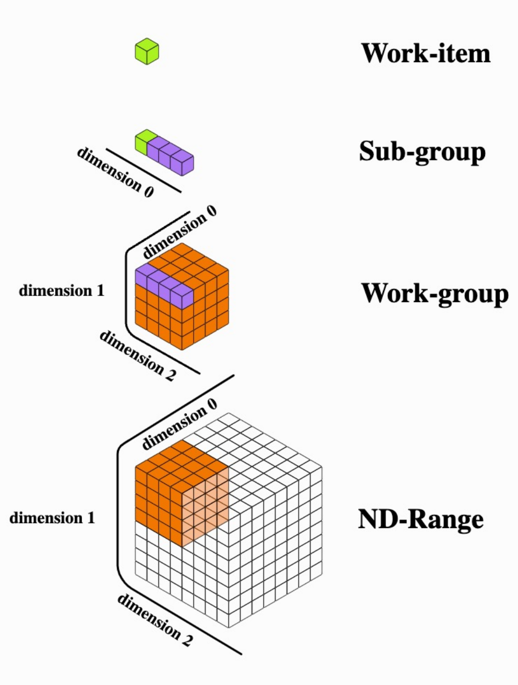
Data:

- Results_v1
- Results_v2



- CPU Intel® Xeon® Platinum 8480
- GPU Intel® Data Center GPU Max 1100
- GPU Nvidia RTX A6000

Execution Model



CUDA	SYCL
thread	work-item
warp	sub-group
block	work-group
grid	ND-range

Original CUDA Code	Migrated SYCL Code

Original CUDA Code:

```
__global__ void foo() {
  int a = threadIdx.x;
}

int main() {
  dim3 size_1(100, 200, 300);
  dim3 size_2(5, 10, 20);

  foo<<<size_1, size_2>>>();
}
```

Migrated SYCL Code:

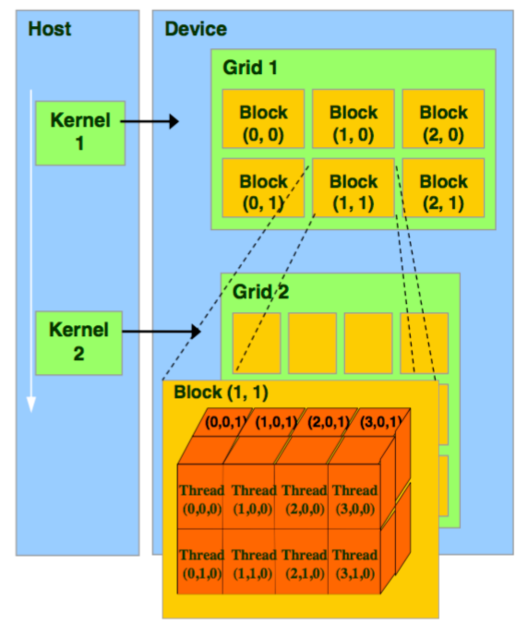
```
void foo(sycl::nd_item<3> item) {
  int a = item.get_local_id(2);
}

int main() {
  sycl::queue q;
  sycl::range<3> size_1(300, 200, 100);
  sycl::range<3> size_2(20, 10, 5);

  q.parallel_for(
    sycl::nd_range<3>(size_1 * size_2, size_2),
    [=](sycl::nd_item<3> item)
[[sycl::reqd_sub_group_size(32)]] {
      foo(item);
    });
}
```

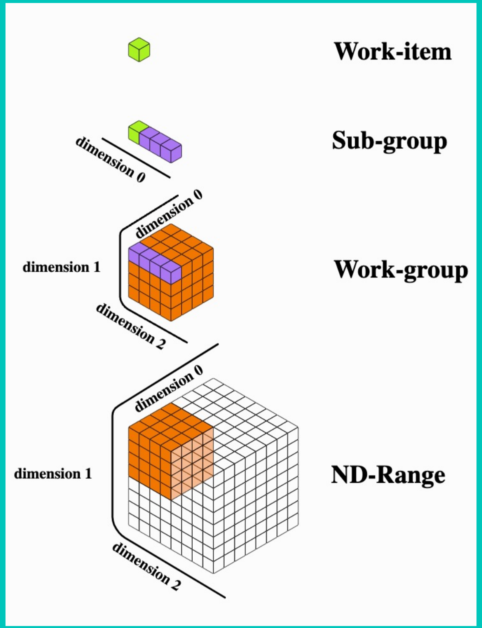
Migration CUDA->SYCL

Kernel functions

```
float_to_char<<<numBlocks_mult, threadsPerBlock>>>(dimx, dimy, _d_xvec, _d_sol_data);
```

CUDA






```
q.parallel_for(
    sycl::nd_range<3>(numBlocks_mult * threadsPerBlock, threadsPerBlock),
    [=](sycl::nd_item<3> item_ct1) {
      float_to_char(dimx, dimy, _d_xvec, _d_sol_data, item_ct1);
    });
```

SYCL

Memory Model

Original CUDA Code	Migrated SYCL Code

<pre>__global__ void foo() {
  __shared__ int shm[16];
  shm[0] = 2;
}
 	
int main() {
  foo<<<1, 1>>>();
}</pre>
	<pre>void foo(int *shm) {
	 shm[0] = 2;
	}
	
	int main() {
	  sycl::queue q;
	  q.submit([&](sycl::handler &cgh) {
	
	    sycl::local_accessor<int>
	shm_acc(sycl::range<1>(16), cgh);
	    cgh.parallel_for(
	        sycl::nd_range<3>(sycl::range<3>(1, 1,
	1), sycl::range<3>(1, 1, 1)), [=]
	(sycl::nd_item<3> item_ct1) {
	          foo(shm_acc.get_pointer());
	        });
	  });</pre>

Shared memory

Global, constant and unified memory

Original CUDA Code	Migrated SYCL Code

<pre>void foo() {
  int *mem1, *mem2;
 	
  cudaMalloc(&mem1, 10);
  cudaMallocManaged(&mem2, 10);
}</pre>
	<pre>void foo() {
	  sycl::queue q;
	  int *mem1, *mem2;
	
	  mem1 = sycl::malloc_device<int>(10, q);
	  mem2 = sycl::malloc_shared<int>(10, q);
	}</pre>

Migrate CUDA->SYCL

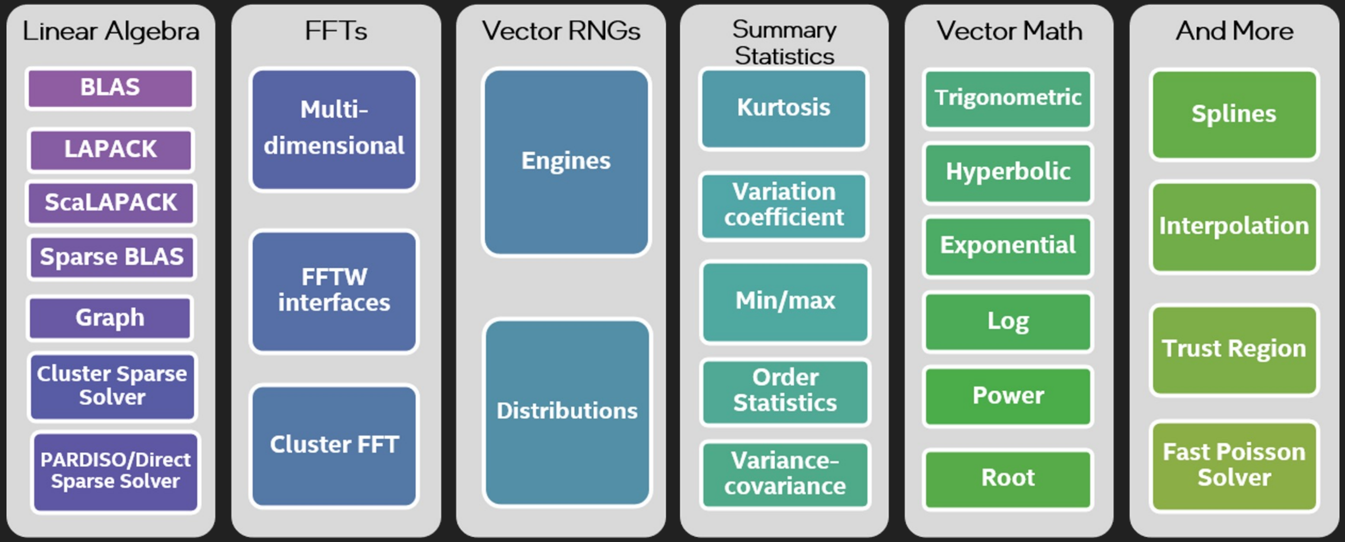
CUBLAS -> Intel OneMKL (Math Kernel Library)

```
rho = cublasSdot(N,(float *)_d_p,1,(float *)_d_p,1);
```

CUDA

```
float *res_temp_ptr_ct1 =
    sycl::malloc_shared<float>(1, dpct::get_default_queue());
rho = *res_temp_ptr_ct1;
oneapi::mkl::blas::column_major::dot(q, N,(float *)_d_p,1,(float *)_d_p,1, &rho).wait();
```

Intel OneMKL



Conclusions

- **SYCL enabled portability of software for use in the Cloud on different computing devices**
- **The Intel OneAPI compatibility tools allowed easy transition from vendor specific implementation of image blending algorithm to SYCL**
- **Experimental results demonstrated great performance portability**
- **Next Steps: SYCL implementation of ZFP compression library**

Reference: "Interactive Visualization and Portable Image Blending of Massive Aerial Image Mosaics". Steve Petruzza, Brian Summa, Amy Gooch, Christine Laney, Tristan Goulden, John Schreiner, Steven Callahan, Valerio Pascucci.
In press at *International Workshop on Big Data Analytics for Climate Change, IEEE International Conference on Big Data, 2023*