OPENACC DATA MANAGEMENT

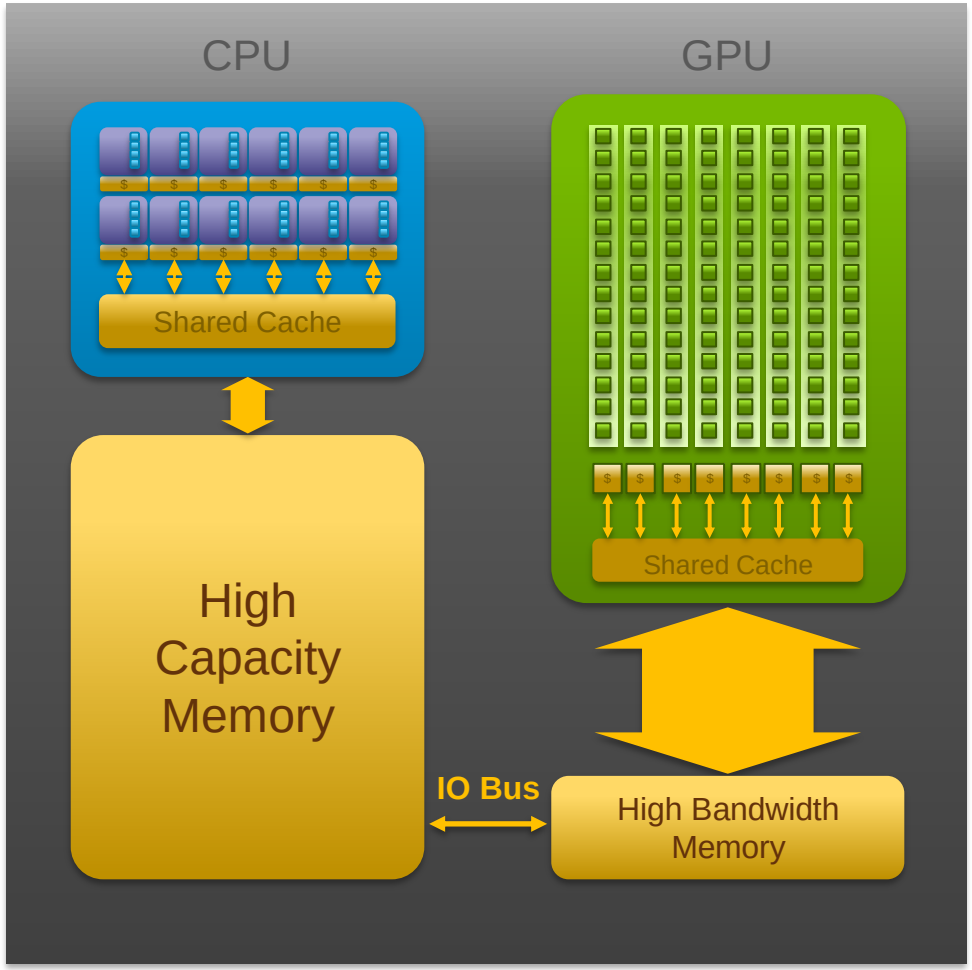
OpenACC
More Science, Less Programming

NVIDIA

CPU AND GPU MEMORIES

CPU + GPU
Physical Diagram

- CPU memory is larger, GPU memory has more bandwidth
- CPU and GPU memory are usually separate, connected by an I/O bus (traditionally PCI-e)
- Any data transferred between the CPU and GPU will be handled by the I/O Bus
- The I/O Bus is relatively slow compared to memory bandwidth
- The GPU cannot perform computation until the data is within its memory



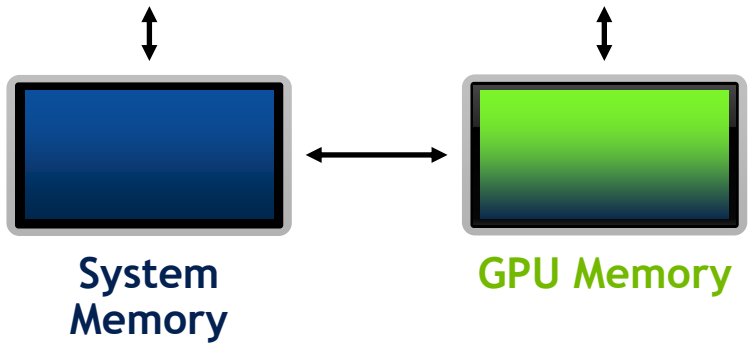
CUDA UNIFIED MEMORY

CUDA UNIFIED MEMORY

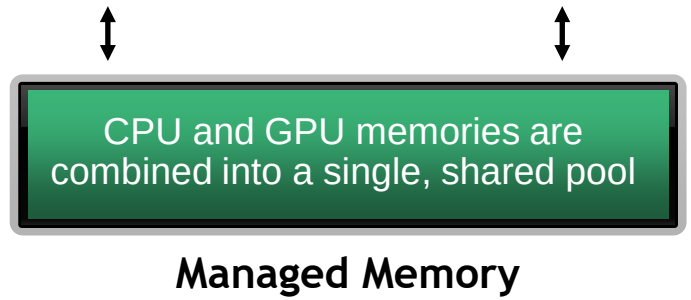
Simplified Developer Effort

Commonly referred to as
"managed memory."

Without Managed Memory



With Managed Memory



OpenACC More Science, Less Programming

NVIDIA

CUDA MANAGED MEMORY

Usefulness

- Handling explicit data transfers between the host and device (CPU and GPU) can be difficult
- The NVIDIA HPC compiler can utilize CUDA Managed Memory to defer data management
- This allows the developer to concentrate on parallelism and think about data movement as an optimization

```
$      nvc –fast –acc –ta=tesla:managed –Minfo=accel main.c
```

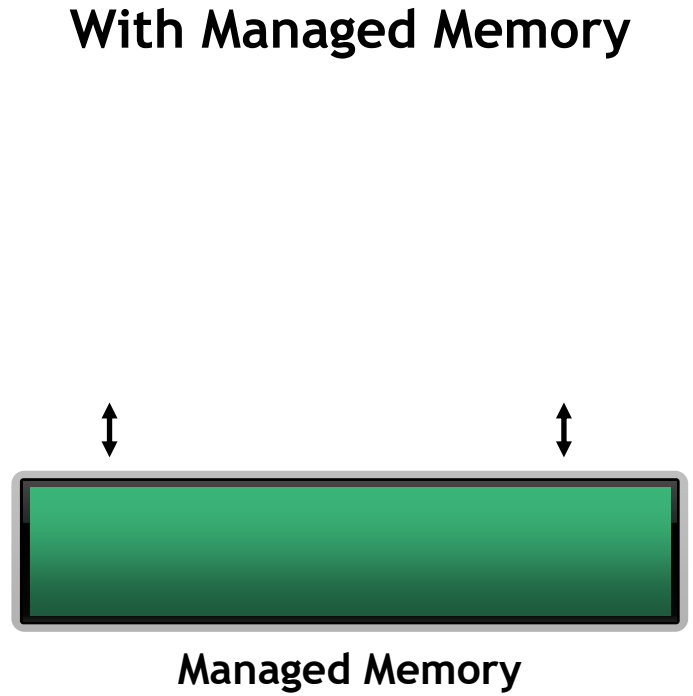
```
$      nvfortran –fast –acc –ta=tesla:managed –Minfo=accel main.f90
```

OpenACC More Science, Less Programming NVIDIA

MANAGED MEMORY

Limitations

- The programmer will almost always be able to get better performance by manually handling data transfers
- Memory allocation/deallocation takes longer with managed memory
- Cannot transfer data asynchronously
- Currently only available on NVIDIA GPUs with NVIDIA HPC SDK.



DATA SHAPING

DATA CLAUSES

`copy( list )` **Allocates memory on GPU and copies data from host to GPU when entering region and copies data to the host when exiting region.**

Principal use: For many important data structures in your code, this is a logical default to input, modify and return the data.

`copyin( list )` **Allocates memory on GPU and copies data from host to GPU when entering region.**

Principal use: Think of this like an array that you would use as just an input to a subroutine.

`copyout( list )` **Allocates memory on GPU and copies data to the host when exiting region.**

Principal use: A result that isn't overwriting the input data structure.

`create( list )` **Allocates memory on GPU but does not copy.**

Principal use: Temporary arrays.

OpenACC More Science, Less Programming NVIDIA

ARRAY SHAPING

- Sometimes the compiler needs help understanding the *shape* of an array
- The first number is the start index of the array
- In C/C++, the second number is how much data is to be transferred
- In Fortran, the second number is the ending index

```
copy(array[starting_index:length])
```

C/C++

```
copy(array(starting_index:ending_index))
```

Fortran

OpenACC More Science, Less Programming NVIDIA

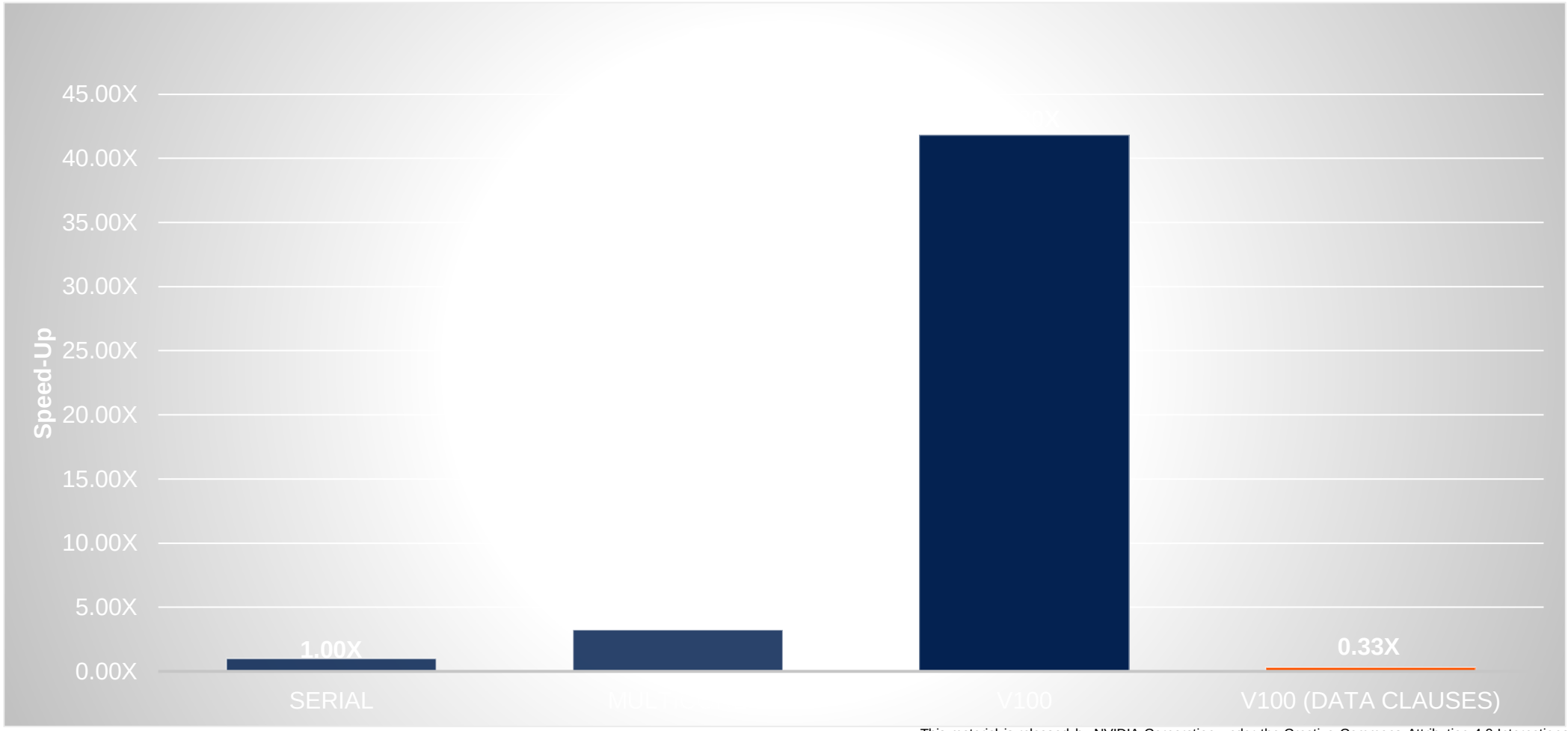
OPTIMIZED DATA MOVEMENT

```
while ( err > tol && iter < iter_max ) {
  err=0.0;

#pragma acc parallel loop reduction(max:err) copyin(A[0:n*m]) copy(Anew[0:n*m])
  for( int j = 1; j < n-1; j++) {
    for(int i = 1; i < m-1; i++) {

      Anew[j][i] = 0.25 * (A[j][i+1] + A[j][i-1] +
                           A[j-1][i] + A[j+1][i]);

      err = max(err, abs(Anew[j][i] - A[j][i]));
    }
  }

#pragma acc parallel loop copyin(Anew[0:n*m]) copyout(A[0:n*m])
  for( int j = 1; j < n-1; j++) {
    for( int i = 1; i < m-1; i++ ) {
      A[j][i] = Anew[j][i];
    }
  }
  iter++;
}
```

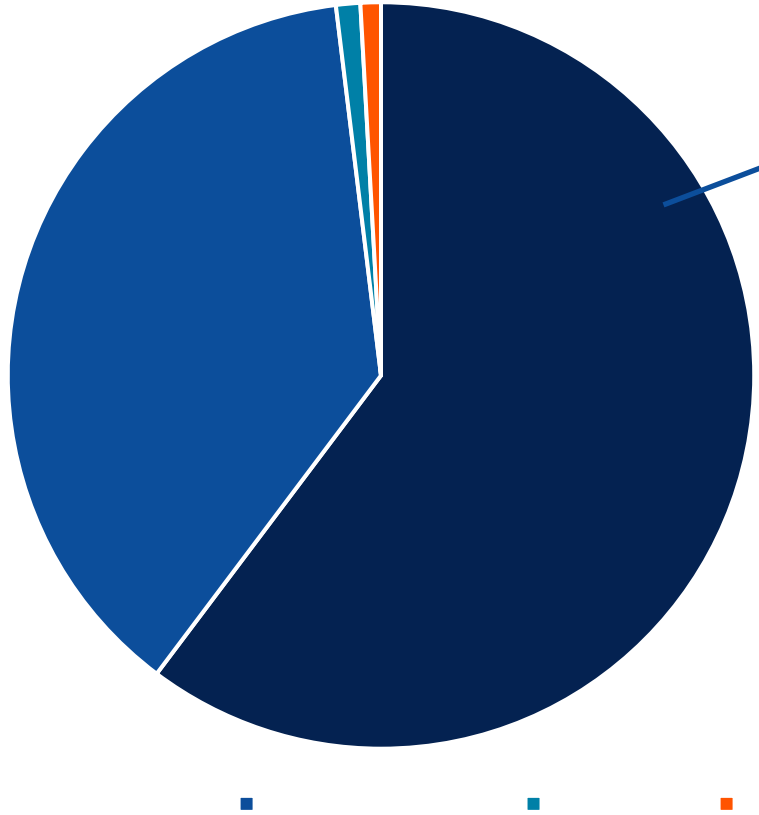
Data clauses provide necessary “shape” to the arrays.

OpenACC More Science, Less Programming NVIDIA

OPENACC ~~SPEED-UP~~ SLOWDOWN



RUNTIME BREAKDOWN



Nearly all of our time is spent moving data to/from the GPU

OPTIMIZED DATA MOVEMENT

```
while ( err > tol && iter < iter_max ) {
  err=0.0;

#pragma acc parallel loop reduction(max:err) copyin(A[0:n*m]) copy(Anew[0:n*m])
  for( int j = 1; j < n-1; j++) {
    for(int i = 1; i < m-1; i++) {

      Anew[j][i] = 0.25 * (A[j][i+1] + A[j][i-1] +
                           A[j-1][i] + A[j+1][i]);

      err = max(err, abs(Anew[j][i] - A[j][i]));
    }
  }

#pragma acc parallel loop copyin(Anew[0:n*m]) copyout(A[0:n*m])
  for( int j = 1; j < n-1; j++) {
    for( int i = 1; i < m-1; i++ ) {
      A[j][i] = Anew[j][i];
    }
  }
  iter++;
}
```

Currently we're copying to/from the GPU for each loop, can we reuse it?

OpenACC More Science, Less Programming NVIDIA

OPTIMIZE DATA MOVEMENT

OPENACC DATA DIRECTIVE

Definition

- The data directive defines a lifetime for data on the device beyond individual loops
- During the region data is essentially “owned by” the accelerator
- Data clauses express shape and data movement for the region

```
#pragma acc data clauses
{

 < Sequential and/or Parallel code >

}
```

```
!$acc data clauses

 < Sequential and/or Parallel code >

!$acc end data
```

OpenACC More Science, Less Programming NVIDIA

OPTIMIZED DATA MOVEMENT

```
#pragma acc data copy(A[:n*m]) copyin(Anew[:n*m])
while ( err > tol && iter < iter_max ) {
  err=0.0;

#pragma acc parallel loop reduction(max:err) copyin(A[0:n*m])
  for( int j = 1; j < n-1; j++) {
    for(int i = 1; i < m-1; i++) {

      Anew[j][i] = 0.25 * (A[j][i+1] + A[j][i-1] +
                           A[j-1][i] + A[j+1][i]);

      err = max(err, abs(Anew[j][i] - A[j][i]));
    }
  }

#pragma acc parallel loop copyin(Anew[0:n*m]) copyout(A[0:n*m])
  for( int j = 1; j < n-1; j++) {
    for( int i = 1; i < m-1; i++ ) {
      A[j][i] = Anew[j][i];
    }
  }
  iter++;
```

Copy A to/from the accelerator only when needed.

Copy initial condition of Anew, but not final value

REBUILD THE CODE

```
nvc -fast -ta=tesla -Minfo=accel laplace2d_uvm.c
main:
     60, Generating copy(A[:m*n])
         Generating copyin(Anew[:m*n])
     64, Accelerator kernel generated
         Generating Tesla code
         64, Generating reduction(max:error)
         65, #pragma acc loop gang /* blockIdx.x */
         67, #pragma acc loop vector(128) /* threadIdx.x */
     67, Loop is parallelizable
     75, Accelerator kernel generated
         Generating Tesla code
         76, #pragma acc loop gang /* blockIdx.x */
         78, #pragma acc loop vector(128) /* threadIdx.x */
     78, Loop is parallelizable
```

Now data movement only happens at our data region.

WHAT WE'VE LEARNED SO FAR

- CUDA Unified (Managed) Memory is a powerful porting tool
- GPU programming without managed memory often requires data shaping
- Moving data at each loop is often inefficient
- The OpenACC Data region can decouple data movement and computation

DATA SYNCHRONIZATION

OPENACC UPDATE DIRECTIVE

update: Explicitly transfers data between the host and the device

Useful when you want to synchronize data in the middle of a data region

Clauses:

self: makes host data agree with device data

device: makes device data agree with host data

```
#pragma acc update self(x[0:count])
#pragma acc update device(x[0:count])
```

C/C++

```
!$acc update self(x(1:end_index))
!$acc update device(x(1:end_index))
```

Fortran

OPENACC RESOURCES

Guides ● Talks ● Tutorials ● Videos ● Books ● Spec ● Code Samples ● Teaching Materials ● Events ● Success Stories ● Courses ● Slack ● Stack Overflow

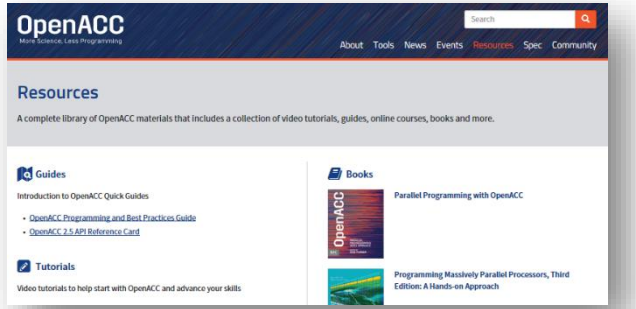
FREE Compilers

NVIDIA HPC SDK

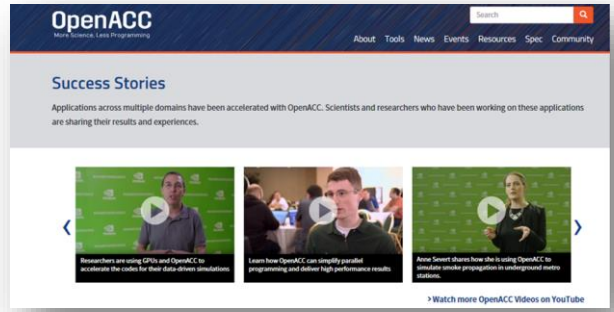
slack

https://www.openacc.org/community

Resources

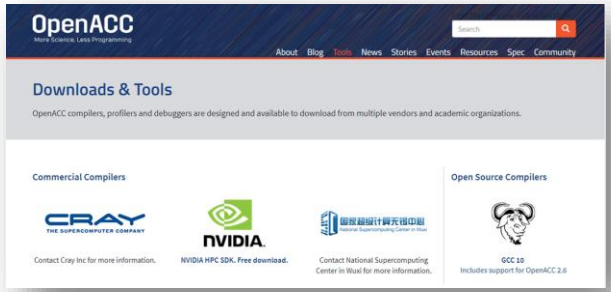


Success Stories



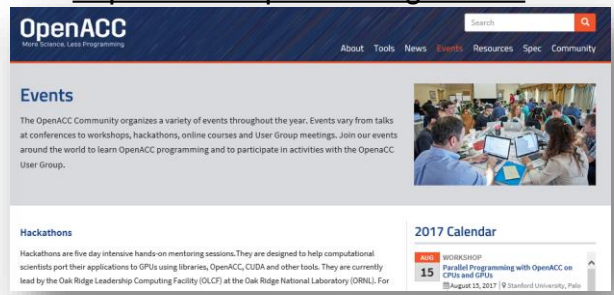
Compilers and Tools

https://www.openacc.org/tools



Events

https://www.openacc.org/events



OpenACC More Science, Less Programming NVIDIA