BLUE WATERS

SUSTAINED PETASCALE COMPUTING

May 3, 2013

# OpenACC Accelerator Directives

NCSA


GREAT LAKES CONSORTIUM
FOR PETASCALE COMPUTATION

CRAY

# OpenACC is ...

An API
- Inspired by OpenMP
- Implemented by Cray, PGI, CAPS
- Includes functions to query device(s)

Evolving
- Plan to integrate into OpenMP
- Support of the 1.0 specification has not resulted in portable code (more later)

# How can I get started with OpenACC?

OpenACC.org
- Quick reference guide (OpenMP programmers)
- Specifications: 1.0 and 2.0 draft
- Classes

OpenACC GPU Prog. Workshop
- Joint workshop with PSC, Xsede, Nvidia
- Targeted PGI implementation

# OpenACC on Blue Waters

Cray

- module load PrgEnv-cray craype-accel-nvidia35
- Caution: OpenMP is also enabled by default
- Directly generates ptx assembly for Nvidia accel.

PGI

- module load PrgEnv-pgi cudatoolkit
- Generates CUDA intermediate

Blue Waters OpenACC compiler table

# Support for directives varies: Cray

```
61  #pragma acc kernels loop
62          for( int j = 1; j < n-1; j++)
63          {
64  #pragma acc loop gang(16) vector(32)
65              for( int i = 1; i < m-1; i++ )
66              {
67                  Anew[j][i] = 0.25 * ( A[j][i+1] + A[j][i-1]
68                                      + A[j-1][i] + A[j+1][i]);
69              }
70          }
```

```
arnoldg@jyc1:~/openacc/wkshp> make laplace2d
cc -Gp -h acc,noomp,msgs -fpic -dynamic   -c -o laplace2d.o
laplace2d.c
WARNING: Ignoring gang clause on acc_loop at main:65
WARNING: Ignoring gang clause on acc_loop at main:77
```

# Support for directives varies: PGI

```
39  !$acc data copyin(sendbuf) copyout(recvbuf)
40
41  !$acc host_data use_device(sendbuf,recvbuf)
42        call MPI_ALLTOALL (sendbuf,n,mpi_complex,recvbuf,n,mpi_complex,comm_col,ierr)
43  !$acc end host_data
44
45  !$acc end data
```

```
arnoldg@h2ologin1:~/buaria> ftn -acc test.f90
PGF90-S-0155-A data clause for a variable appears within another region with a data clause
for the same variable sendbuf (test.f90: 41)
PGF90-S-0155-A data clause for a variable appears within another region with a data clause
for the same variable recvbuf (test.f90: 41)
  0 inform,   0 warnings,   2 severes, 0 fatal for MAIN
```

# Runtime differences using Cray's examples and PGI compiler: [code from : man openacc.examples ]

## PGI runtime incorrect

```
47            !!$ Compute a checksum
48            !$acc parallel copyin(a)
49                    total = 0
50            !$acc loop reduction(+:total)
51                    DO j = 1,M
52                            total = total + a(j)
53                    ENDDO
54            !$acc end loop
55            !$acc end parallel
```

## PGI runtime valid

```
47            !!$ Compute a checksum
48                    total = 0
49            !$acc kernels loop copyin(a) reduction(+:total)
50                    DO j = 1,M
51                            total = total + a(j)
52                    ENDDO
53
54            !$acc end kernels loop
```

# Tuning differences: Cray and PGI

```
  145  !$acc parallel num_gangs(1) vector_length(3072)
  146  !!$acc kernels
  147  !!data copy(part),copyin(fxy),create(nn,mm,dxp,dyp,np,mp,dx,dy,vx,vy)
  148        do 10 j = 1, nop
  149  c find interpolation weights
  150        nn = part(1,j)
  151        mm = part(2,j)
  152        dxp = part(1,j) - real(nn)
  153        dyp = part(2,j) - real(mm)
  154        nn = nn + 1
  155        mm = mm + 1
  156        amx = 1.0 - dxp
  157        mp = mm + 1
  158        amy = 1.0 - dyp
  159        np = nn + 1
  160  c find acceleration
  161        dx = dyp*(dxp*fxy(1,np,mp) + amx*fxy(1,nn,mp)) + amy*(dxp*fxy(1,np
  162       1,mm) + amx*fxy(1,nn,mm))
```

```
arnoldg@jyc1:~/Mori/pic2.0-acc-f> ftn -h acc -c push2.f
!$acc parallel num_gangs(1) vector_length(3072)
ftn-7271 crayftn: WARNING GPUSH2L, File = push2.f, Line = 145
  Unsupported OpenACC vector_length expression: Converting 3072 to
1024.
```

# OpenACC performance tools

Cray
- Perftools support
- Accelerator counters
- A multi-step process (for now)
- CRAY_ACC_DEBUG=1|2|3

PGI
- Profiling via PGI_ACC_TIME=1
- Tracing via PGI_ACC_NOTIFY=1|3

See the Blue Waters documentation

# OpenACC pitfalls

Beware of silent failure modes

- Omitting craype-accel-nvidia35 or cudatoolkit
- -g flag breaks the Cray OpenACC runtime environment
  - CRAY_ACC_ERROR ...
- CRAY_CUDA_PROXY=1 ( sharing the Accelerator in a node )
  - If code fits within the Accelerator memory, results are fine
  - CUDA_ERROR_OUT_OF_MEMORY
  - Incorrect results but no CUDA_ errors

OpenMP and OpenACC should not be nested within your code at this time

BLUE WATERS

SUSTAINED PETASCALE COMPUTING

May 3, 2013

CRAY_CUDA_PROXY
MPICH_RDMA_ENABLED_CUDA
MPICH_G2G_PIPELINE




GREAT LAKES CONSORTIUM
FOR PETASCALE COMPUTATION


# export CRAY_CUDA_PROXY=[1|0]

From the man pages [ man aprun ]:

Enables execution in simultaneous contexts for GPU-equipped nodes ( Hyper Q) when set to 1 or on. The default is 1. Debugging is only supported with the CUDA proxy disabled. To disable CUDA proxy, set to 0 or off

```
module unload cray-mpich2
module load cray-mpich2/5.6.4
export LD_LIBRARY_PATH=$CRAY_LD_LIBRARY_PATH:$LD_LIBRARY_PATH
```

# Comparison with a serial OpenACC sample code

```
time aprun -n 2 -N 1 ./fpic2_acc
 Initial Field, Kinetic and Total Energies:
 0.0000000E+00 0.1677870E+08 0.1677870E+08
 Initial Field, Kinetic and Total Energies:
 0.0000000E+00 0.1677870E+08 0.1677870E+08
…
real 0m41.581s
user 0m0.136s
sys  0m0.040s
```

2 GPUs

Sharing 1 GPU

```
time aprun -n 2 -N 2 ./fpic2_acc
 Initial Field, Kinetic and Total Energies:
 0.0000000E+00 0.1677870E+08 0.1677870E+08
 Initial Field, Kinetic and Total Energies:
 0.0000000E+00 0.1677870E+08 0.1677870E+08
…
real 0m53.325s
user 0m0.136s
sys  0m0.036s
```

# CRAY_CUDA_PROXY=0, error message

```
time aprun -n 2 -N 2 ./fpic2_acc
call to cuCtxCreate returned error 101: Invalid device
CUDA driver version: 5000
[NID 00080] 2013-05-09 10:30:50 Apid 170090: initiated application termination
Application 170090 exit codes: 1
Application 170090 resources: utime ~4s, stime ~0s, Rss ~270056, inblocks ~1659,
outblocks ~4288

real 0m4.868s
user 0m0.120s
sys  0m0.048s
```

# MPICH_RDMA_ENABLED_CUDA

## Module load cray-mpich2/5.6.4 or later

See also: GPUDirect

From the man pages [man mpi]:

MPICH_RDMA_ENABLED_CUDA

If set, allows the MPI application to pass GPU pointers directly to point-to-point and collective communication functions. Currently, if the send or receive buffer for a point-to-point or collective communication is on the GPU, the network transfer and the transfer between the host CPU and the GPU are pipelined to improve performance. Future implementations may use an RDMA-based approach to write/read data directly to/from the GPU, bypassing the host CPU.

Default: not set

# MPICH_G2G_PIPELINE

MPICH_G2G_PIPELINE

If nonzero, the device-host and network transfers will be overlapped to pipeline GPU-to-GPU transfers. Setting MPICH_G2G_PIPELINE to N will allow N GPU-to-GPU messages to be efficiently in-flight at any one time. If MPICH_G2G_PIPELINE is nonzero but MPICH_RDMA_ENABLED_CUDA is disabled, MPICH_G2G_PIPELINE will be turned off. If MPICH_RDMA_ENABLED_CUDA is enabled but MPICH_G2G_PIPELINE is 0, the default value is set to 16. Pipelining is never used on Aries networks for messages with sizes >= 8 KB and < 128 KB.

Default: not set

# MPICH_RDMA_ENABLED_CUDA , MPICH_G2G_PIPELINE (latency)

```
arnoldg@jyc1:~/osu-micro-benchmarks-4.0.1/mpi/collective>
> export MPICH_G2G_PIPELINE=1
> aprun -n 32 -N 1 ./osu_alltoall -d openacc | tail -4
# Size       Avg Latency(us)
262144                15232.80
524288                23825.47
1048576               39943.72
```

Don't set MPICH_G2G_PIPELINE=1

set MPICH_G2G_PIPELINE=4 ( or greater, remember Cray defaults it to 16 if unset )

```
> export MPICH_G2G_PIPELINE=4
> aprun -n 32 -N 1 ./osu_alltoall -d openacc | tail -4
262144                10815.72
524288                17106.18
1048576               28805.91
```

# MPICH_RDMA_ENABLED_CUDA , MPICH_G2G_PIPELINE (bandwidth)

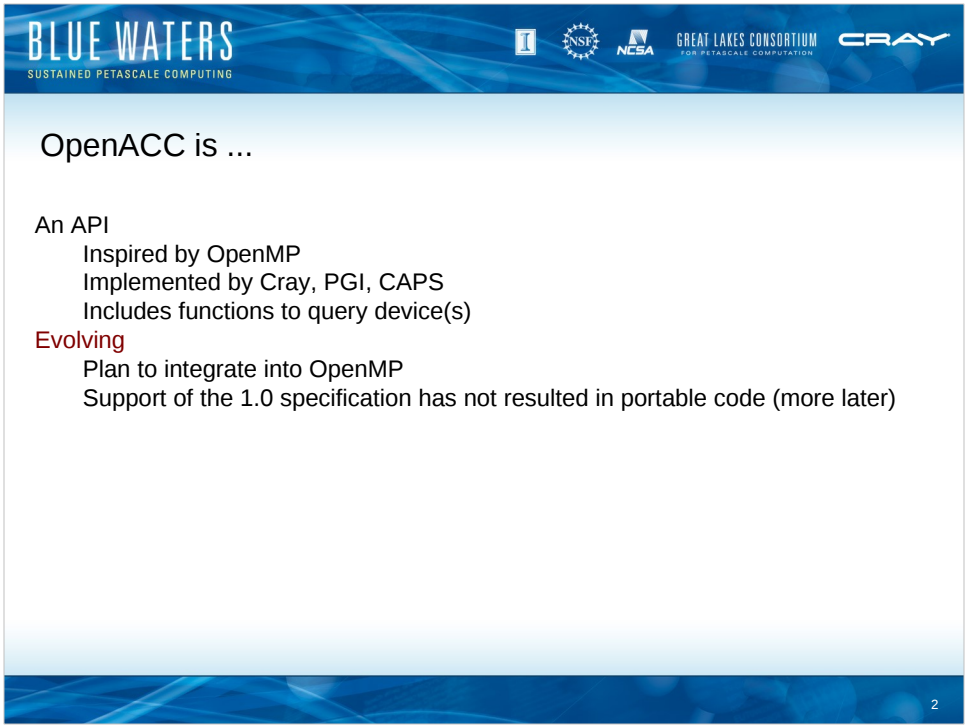
MPICH_G2G_PIPELINE=1

```
> aprun -n 2 -N 1 ./osu_bibw D D
# OSU MPI-OPENACC Bi-Directional Bandwidth Test v4.0.1
# Send Buffer on DEVICE (D) and Receive Buffer on DEVICE (D)
# Size     Bi-Bandwidth (MB/s)
131072                631.32
262144                931.28
524288               1227.69
1048576              1417.43
2097152              1676.00
4194304              1649.22
```

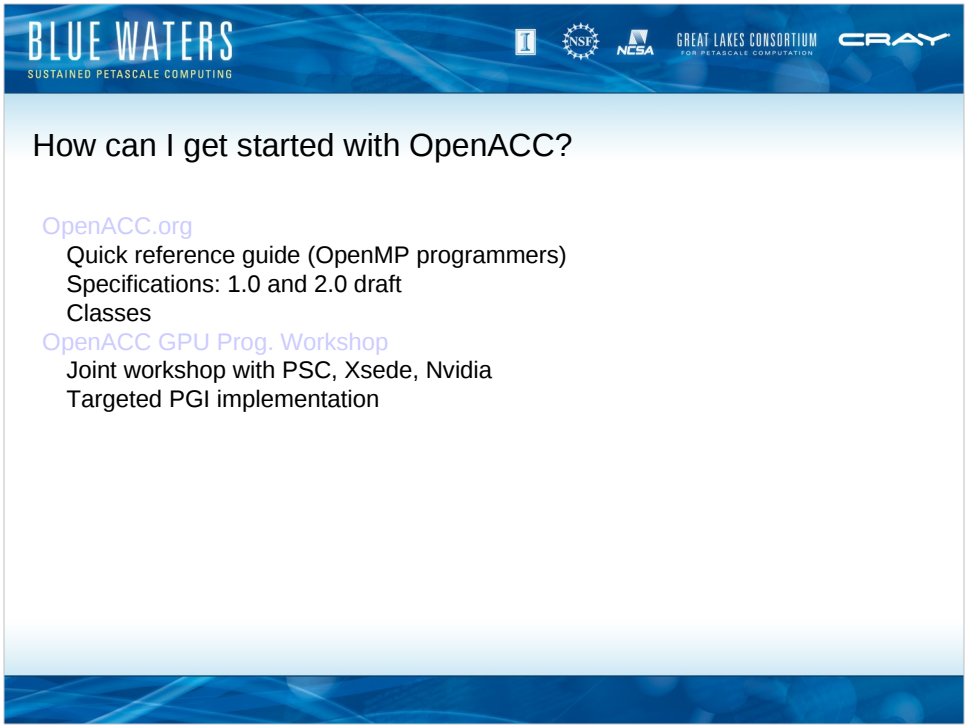
export MPICH_G2G_PIPELINE=4

```
mpi/pt2pt> aprun -n 2 -N 1 ./osu_bibw D D | tail -7
131072                639.54
262144                946.22
524288               1253.30
1048576              1433.01
2097152              1700.69
4194304              1875.10
```

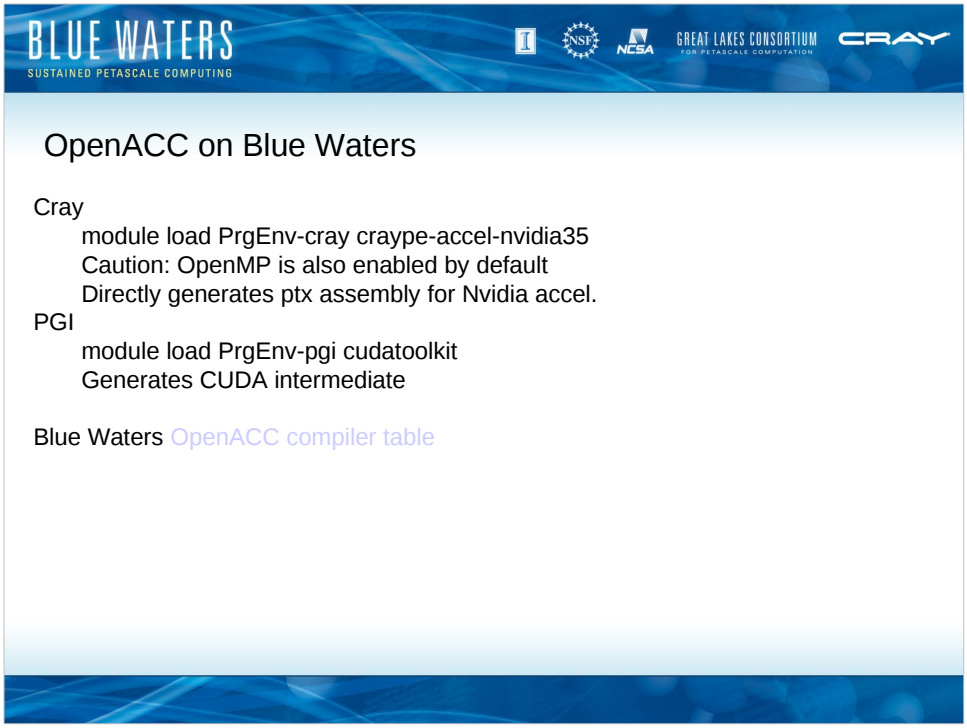

BLUE WATERS
SUSTAINED PETASCALE COMPUTING
May 3, 2013
OpenACC Accelerator Directives
NCSA
NSF
GREAT LAKES CONSORTIUM
FOR PETASCALE COMPUTATION
CRAY



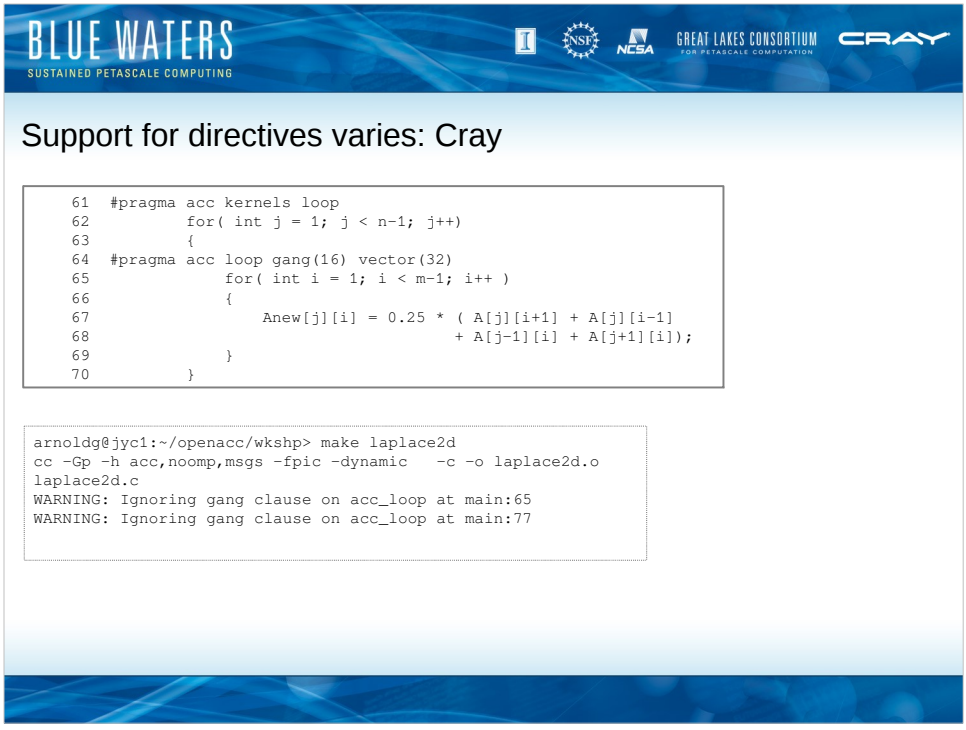

It's widely thought that OpenACC will be integrated into the OpenMP standard in 2013 or 2014. Intel's participation is an open question as they're currently pursuing their own extensions to OpenMP for XeonPHI support.



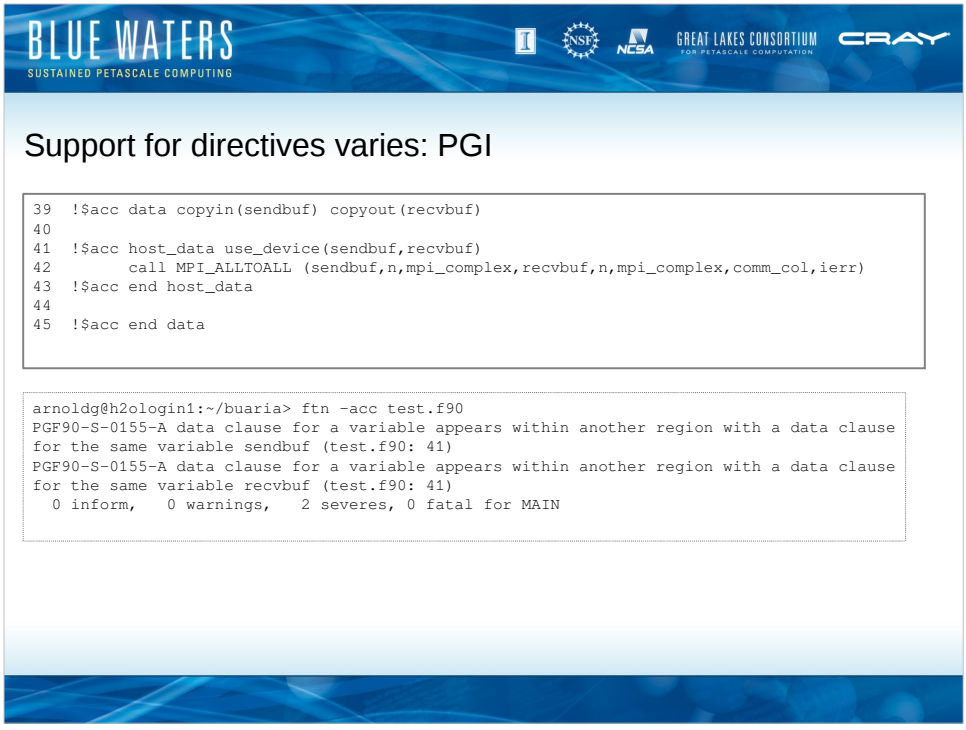

The OpenACC GPU Programming Workshop has been presented locally via HD Video as a virtual workshop. Presentation materials are available at the link.



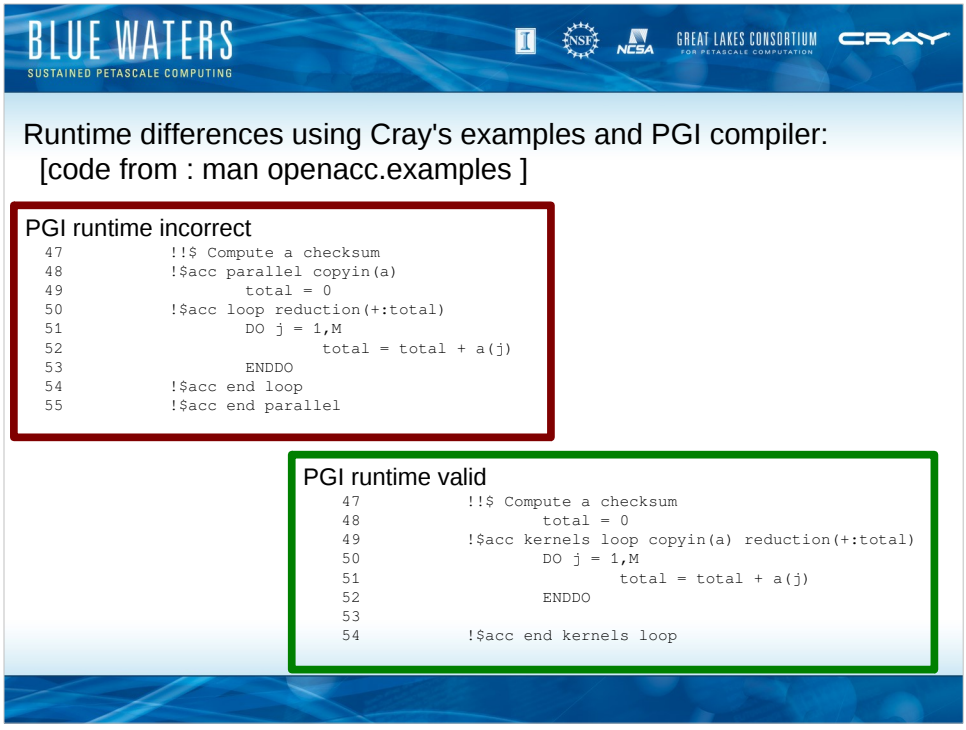

See the Blue Waters user guide and programming information for the compiler table and OpenACC discussion.



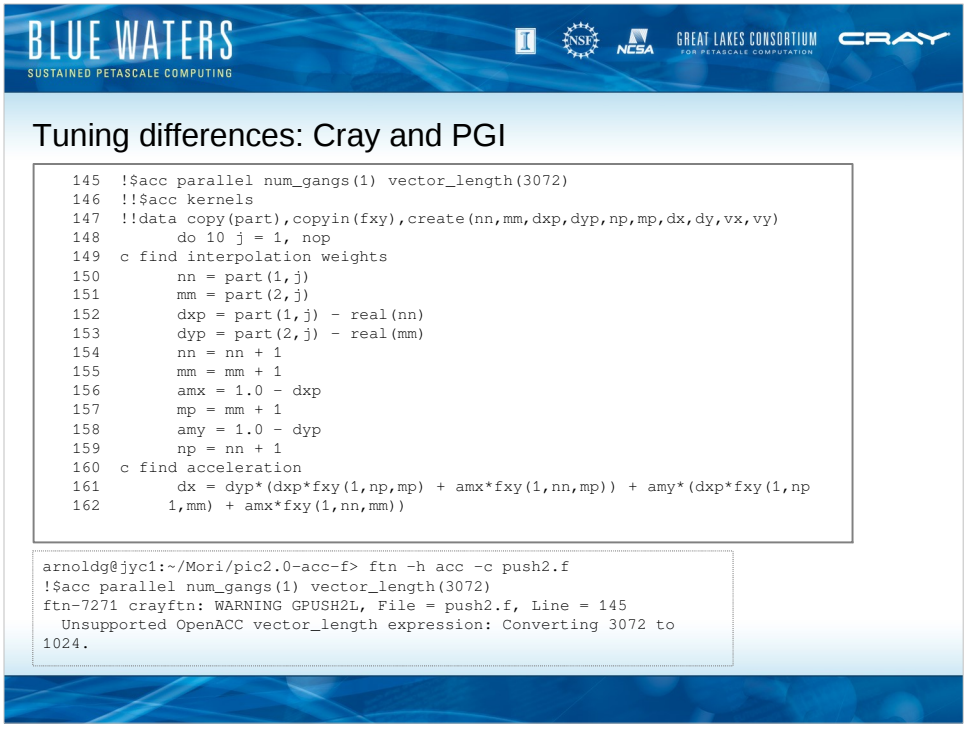

This loop is from the laplace2d.c example code from the OpenACC GPU Programming Workshop and it works with the PGI compiler without warnings or errors.



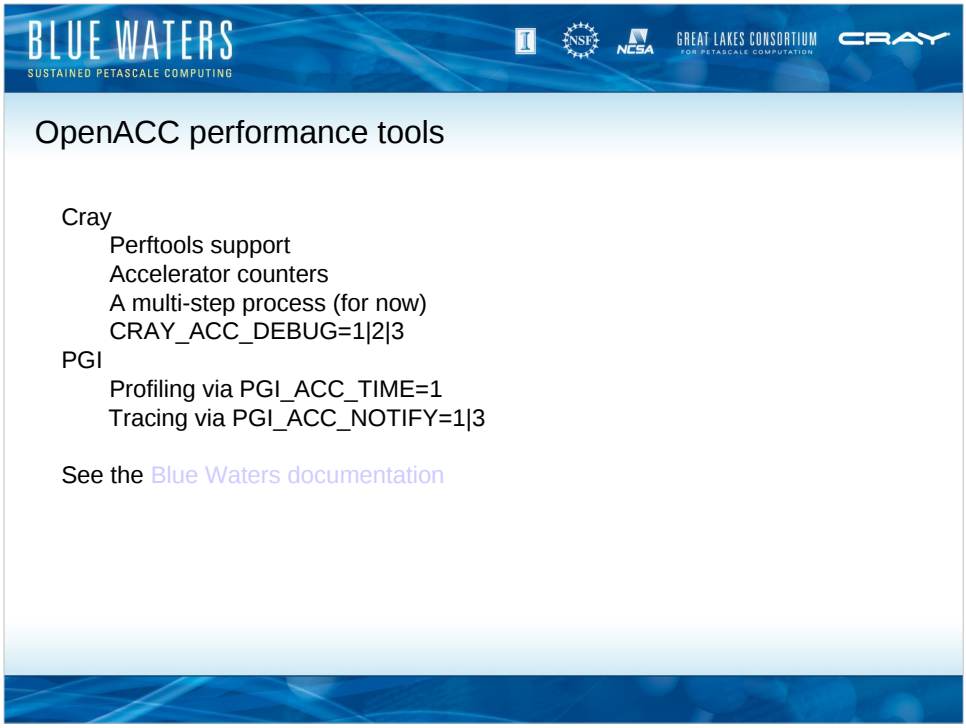

This code fragment was from a team on Blue Waters that is using the Cray compiler with some advanced specification-1.0 features to try to improve memory transfer performance between host and accelerator. The host_data directive is a hint to the compiler to use the address of the data on the accelerator when possible and streamline the use of memory bandwidth.



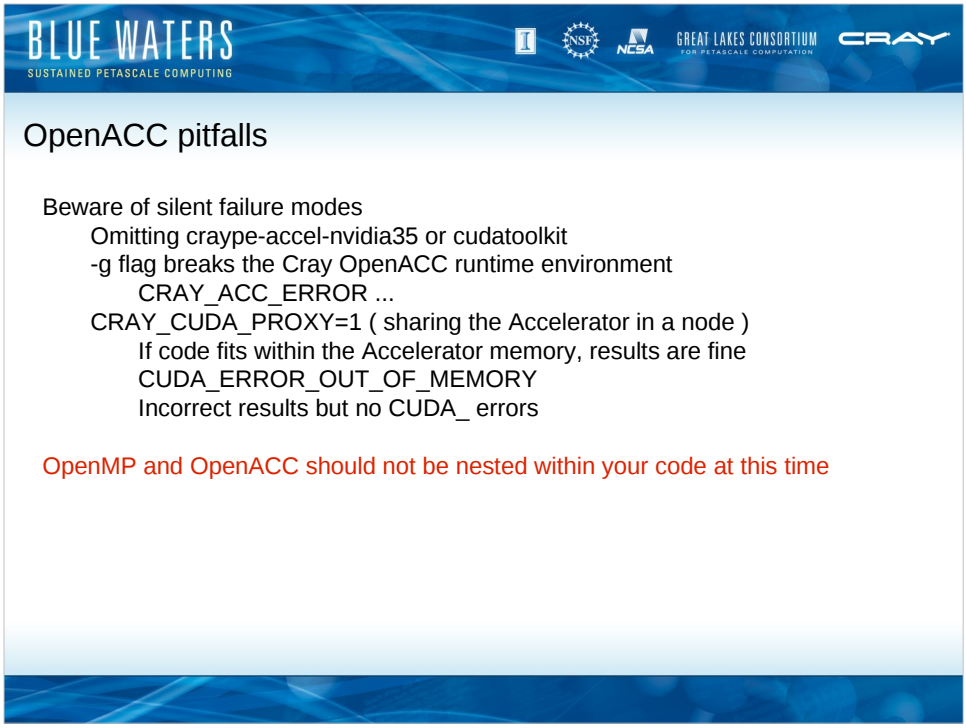

The code snippet here if rom example 4 of the openacc.examples Cray manual page. With PGI it compiled without warnings or errors but the checksum is invalid at runtime when using the parallel directive. The kernels directive yields correct results.




This code is a tuning exercise with a serial kernel for one of the Blue Waters science teams. The PGI compiler showed good speedup with the directive at line 45 (better than the alternative directives of 46-47). The Cray compiler does not accept that directive as written and performance was reduced in this case for the Cray version of the code.




Cray provides access to the Accelerator hw counters, but you can only get 1 set of counters per aprun invocation. PGI profiling is quick and easy to use. It's slightly more intuitive than the CRAY_ACC_DEBUG options.

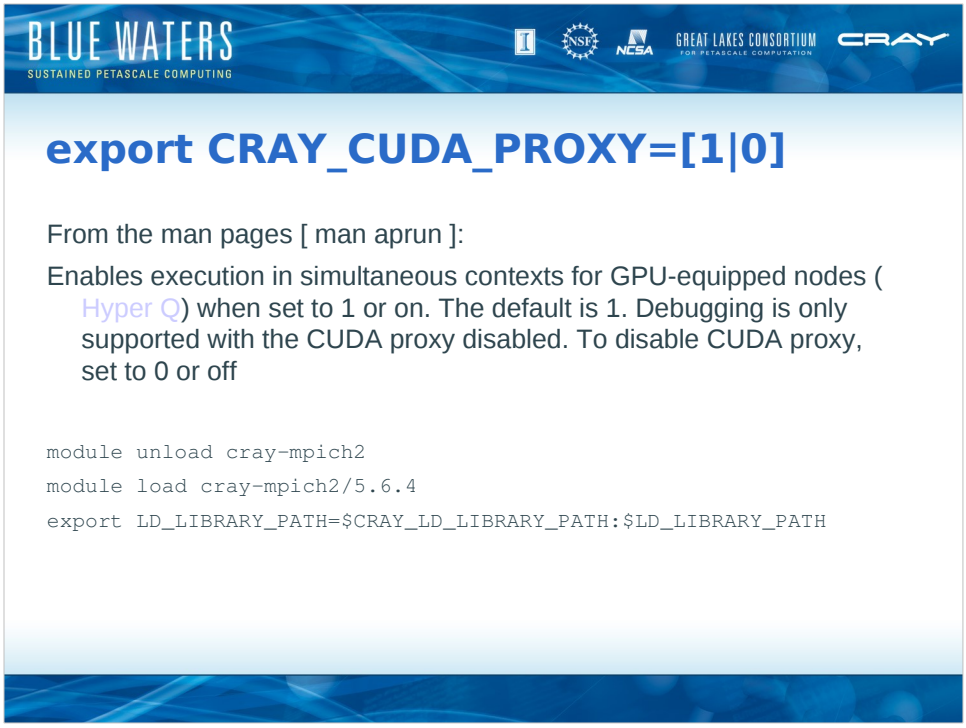



Both programming environments are subject to a variety of silent failures at compile or runtime. Error handling for the OpenACC programming environment is still somewhat immature.

Cray does not allow any nesting of OpenACC within OpenMP regions. PGI allows it but care must be taken with the API to manage threads sharing the GPU. NCSA does not recommend this programming practice at the current time.

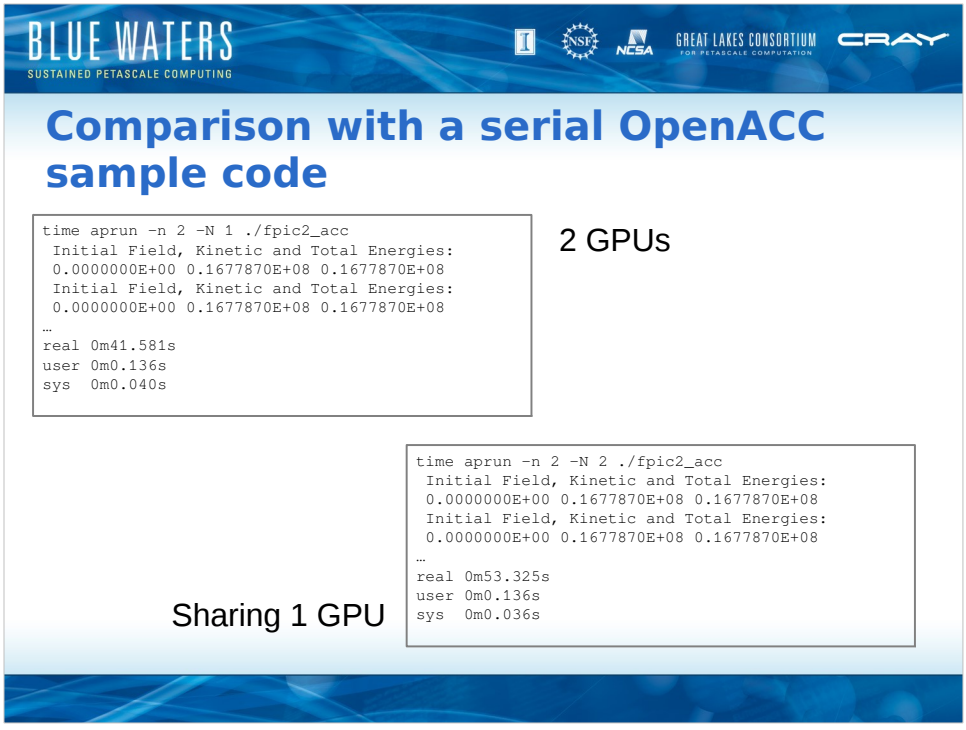
It's ok to use OpenMP and OpenACC in the same code if they target separate loops or sections of code.


This section covers GPU Hyper-q virtualization (CRAY_CUDA_PROXY) and work Cray is doing toward RDMA support.

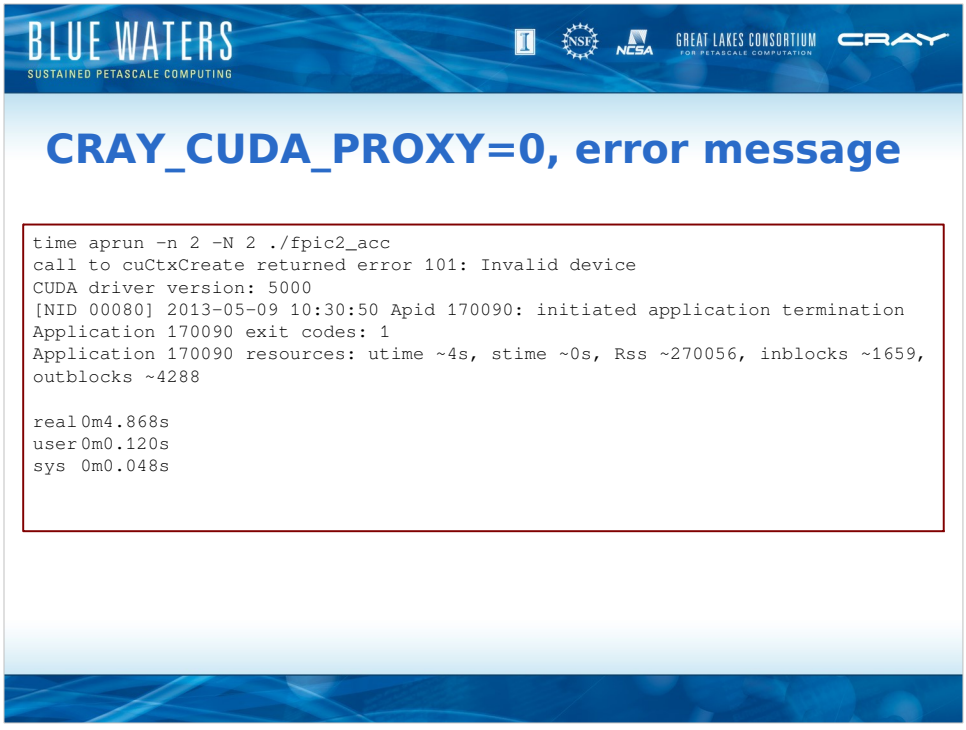



Note the debug requirement for CRAY_CUDA_PROXY set disabled. At the current software revision, CRAY_CUDA_PROXY is not defaulting to enabled as suggested in the manual page. It's best to manually set it if you plan to share a GPU with MPI ranks or OpenMP threads on a host.

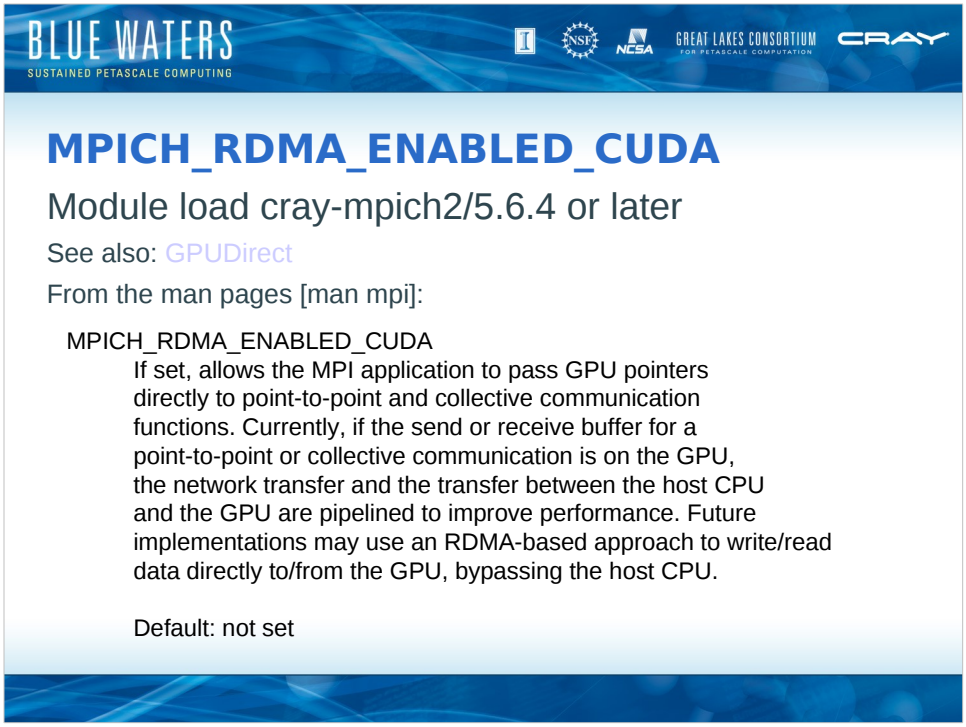
The module noted for mpich2 was used with MPICH_RDMA_ENABLED_CUDA as it's a newer feature for cray-mpich2.




The sample PRAC kernel shown is serial, but running 2 copies of it highlights the use of CRAY_CUDA_PROXY=1.

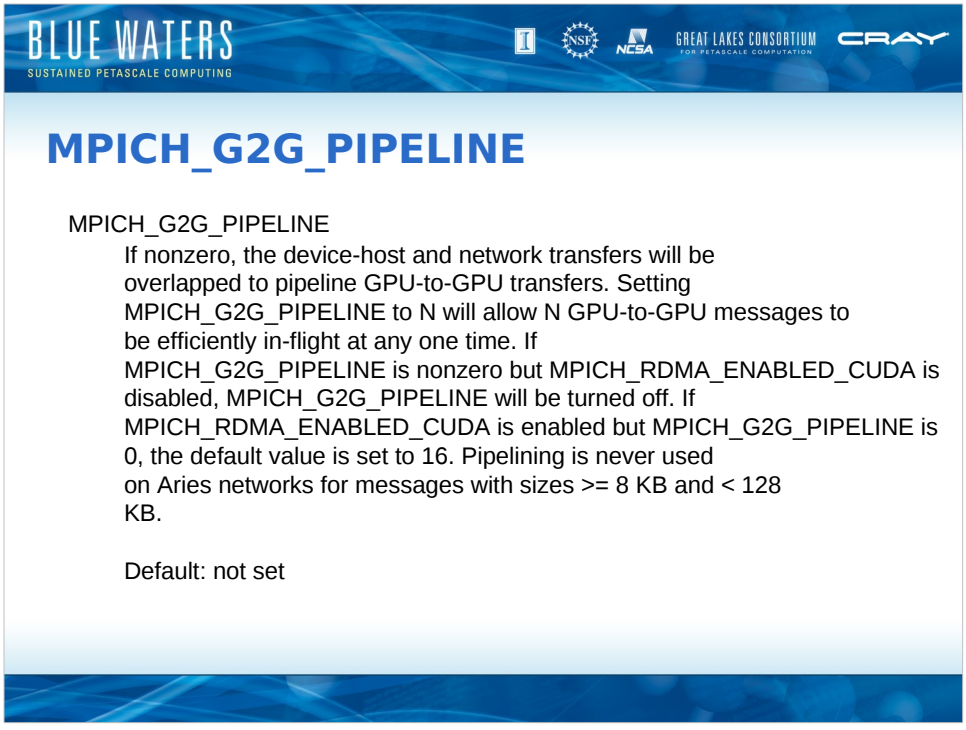



With hyper-q disabled, applications will fail if they're expecting multiple GPU contexts per node.

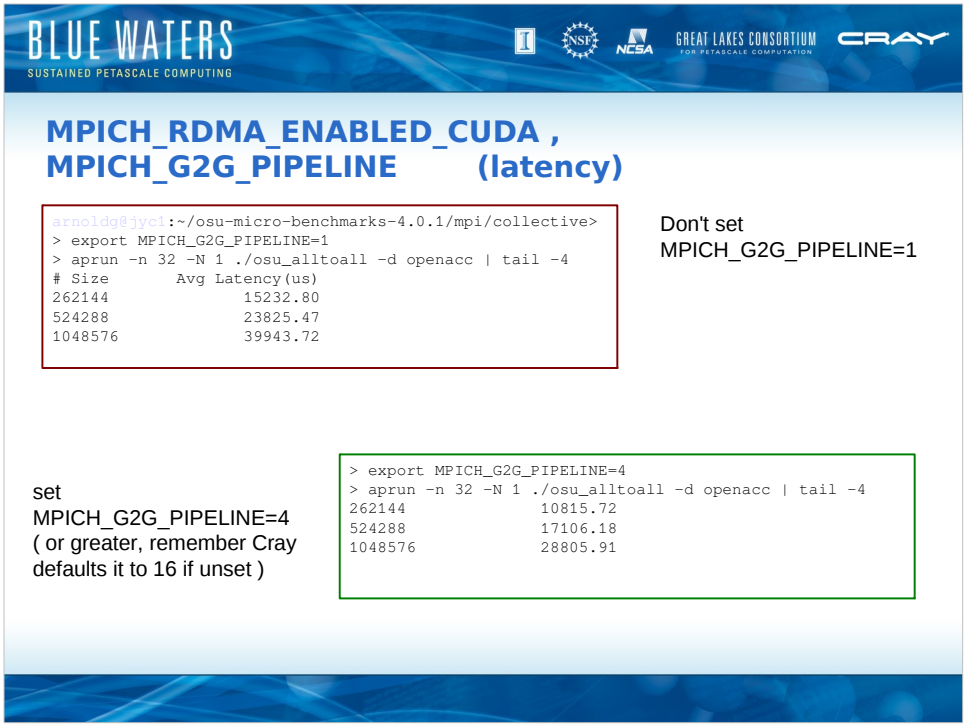



Cray is working toward pure RDMA from GPU to GPU over the Gemini network, but that functionality is not fully implemented. In the meantime, they've optimized the memory transfers via pipelining and they support the API (placing GPU buffers directly into MPI calls).

Using MPICH_RDMA_ENABLED_CUDA implies changing your code (or using code that's already been changed from a cluster where this is supported). See also https://developer.nvidia.com/gpudirect .

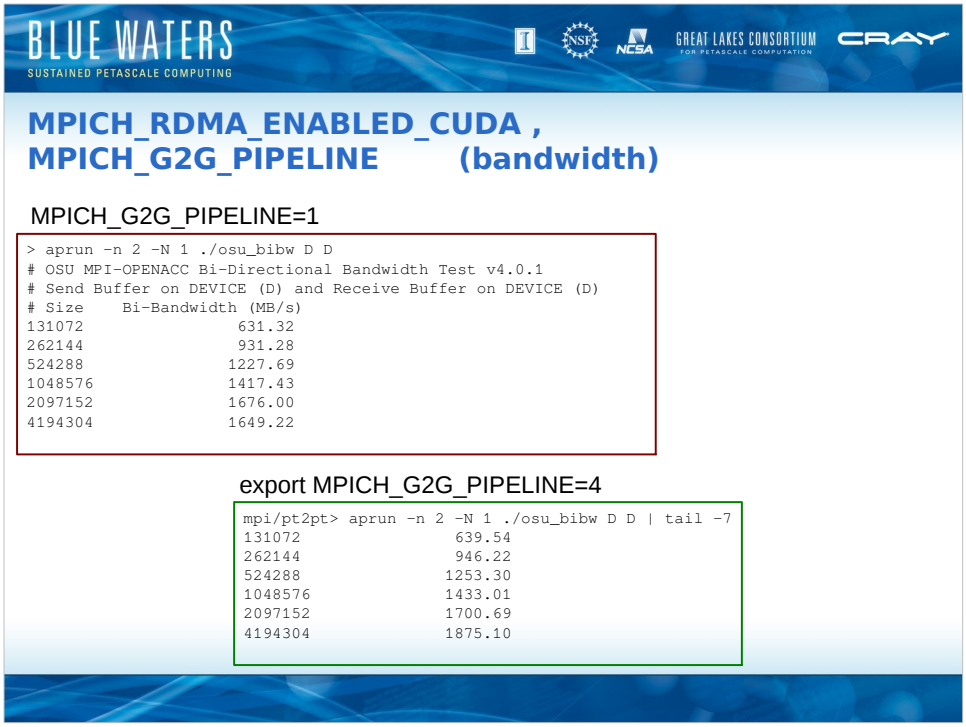



This environment variable is available to assist with tuning MPICH_RDMA_ENABLED_CUDA. It should be unset, or set to something > 1 .




The OSU micro benchmarks were built with PrgEnv-cray, craype-accel-nvidia35, and setting configure to cross-compile (--host=cray). Some minor hacking of the resultant Makefiles was also needed (removing -g, removing an unresolved malloc replacement).

The status of the PGI compiler support for addressing GPU buffers directly in MPI routines has not been investigated. I've seen at least one case where it was not supported and the compiler threw an error so for these examples I stuck with PrgEnv-cray.




The bi-directional bandwidth test doesn't show quite the improvement as the alltoall latency, but then again it's limited to only 2 ranks.