

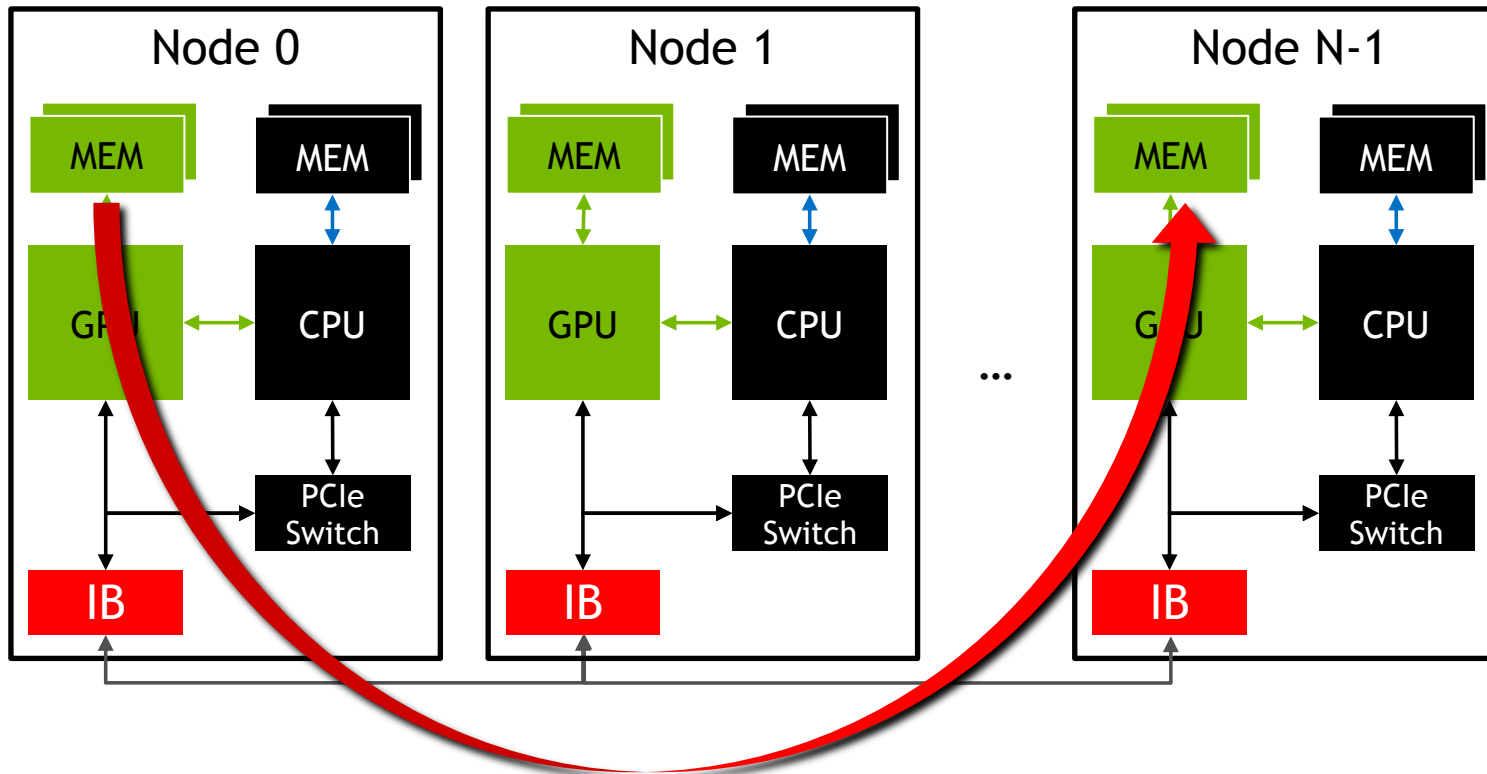


LECTURE ON MULTI-GPU PROGRAMMING

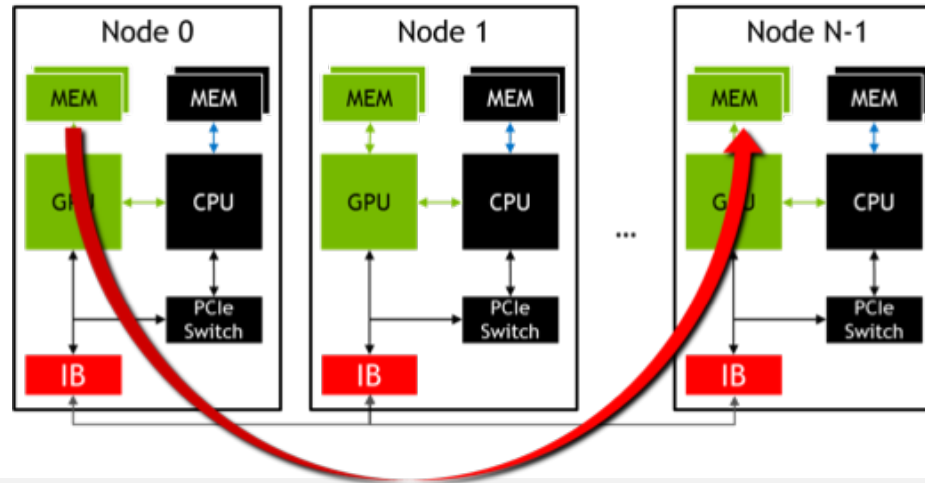
Mathias Wagner, November 12th 2018

MULTI GPU PROGRAMMING

With MPI and OpenACC



MPI+OPENACC



```
//MPI rank 0
#pragma acc host_data use_device( sbuf )
MPI_Send(sbuf, size, MPI_DOUBLE, n-1, tag, MPI_COMM_WORLD);

//MPI rank n-1
#pragma acc host_data use_device( rbuf )
MPI_Recv(rbuf, size, MPI_DOUBLE, 0, tag, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
```

An abstract network diagram with green nodes and lines on a dark background. The nodes are represented by small green circles, some of which are larger and more prominent. They are connected by thin, light green lines that crisscross the frame, creating a complex web of connections. The background is a deep black, with some faint, larger green circles that appear to be out of focus or part of the network's structure.

USING MPI FOR INTER GPU COMMUNICATION

MESSAGE PASSING INTERFACE - MPI

Standard to exchange data between processes via messages

Defines API to exchanges messages

Point to Point: e.g. `MPI_Send`, `MPI_Recv`

Collectives: e.g. `MPI_Reduce`

Multiple implementations (open source and commercial)

Bindings for C/C++, Fortran, Python, ...

E.g. MPICH, OpenMPI, MVAPICH, IBM Spectrum MPI, Cray MPT, ...

MPI - SKELETON

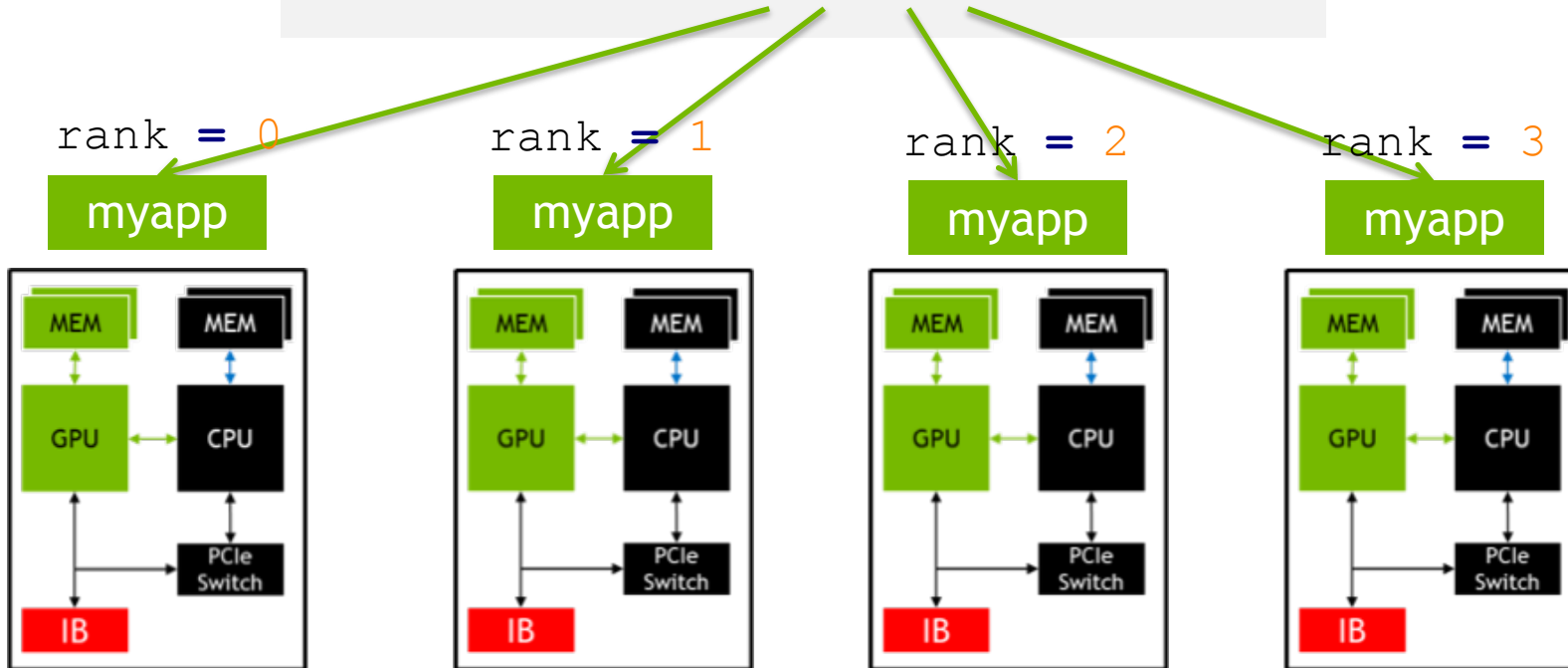
```
#include <mpi.h>

int main(int argc, char *argv[]) {
    int rank, size;
    /* Initialize the MPI library */
    MPI_Init(&argc, &argv);
    /* Determine the calling process rank and total number of ranks */
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    /* Call MPI routines like MPI_Send, MPI_Recv, ... */
    ...
    /* Shutdown MPI library */
    MPI_Finalize();
    return 0;
}
```

MPI

Compiling and Launching

```
$ mpicc -o myapp myapp.c  
$ mpirun -np 4 ./myapp <args>
```



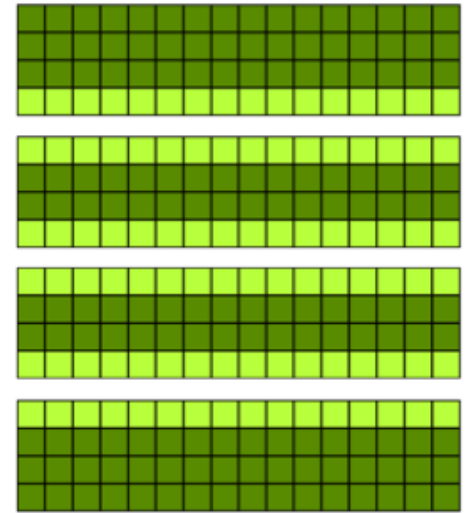
EXAMPLE: JACOBI SOLVER

Solves the 2D-Poisson Equation on a rectangle

$$\Delta u(x, y) = e^{-10*(x^2+y^2)} \quad \forall (x, y) \in \Omega \setminus \delta\Omega$$

Periodic boundary conditions

Domain decomposition with stripes



Horizontal Stripes

EXAMPLE: JACOBI SOLVER

Single GPU

While not converged

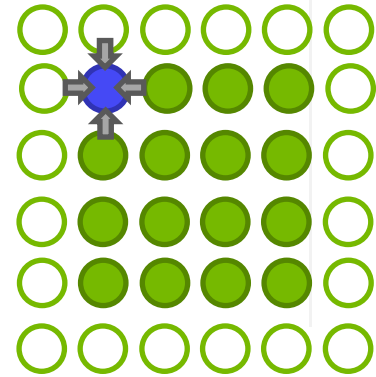
Do Jacobi step:

```
for (int iy = 1; iy < ny-1; ++iy)
    for (int ix = 1; ix < nx-1; ++ix)
        Anew[iy*nx+ix] = -0.25f * (rhs[iy*nx+ix] - (A[iy*nx+(ix-1)] + A[iy*nx+(ix+1)]
                                                    + A[(iy-1)*nx+ix] + A[(iy+1)*nx+ix]));
```

Copy `Anew` to `A`

Apply periodic boundary conditions

Next iteration



EXAMPLE: JACOBI SOLVER

Multi GPU

While not converged

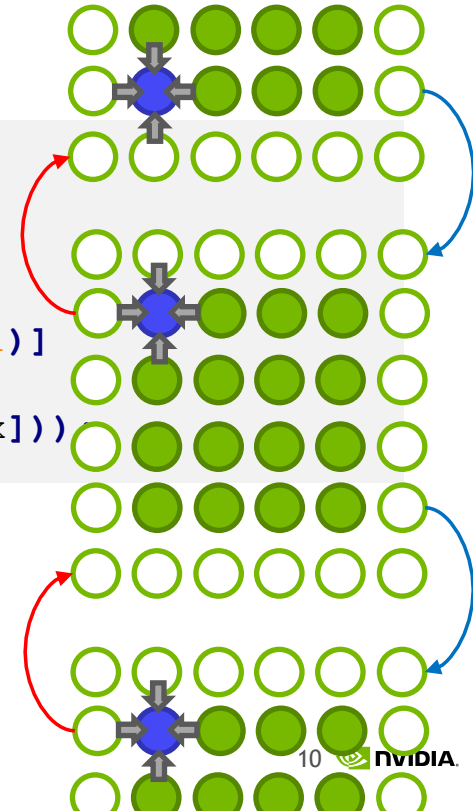
Do Jacobi step:

```
for (int iy = iy_start; iy < iy_end; ++iy)
    for (int ix = 1; ix < NX-1; ++ix)
        Anew[iy*nx+ix] = -0.25f * (rhs[iy*nx+ix] - (A[iy*nx+(ix-1)] + A[iy*nx+(ix+1)]
                                                    + A[(iy-1)*nx+ix] + A[(iy+1)*nx+ix]))
```

Copy `Anew` to `A`

Apply periodic boundary conditions and exchange halo with 2 neighbors

Next iteration

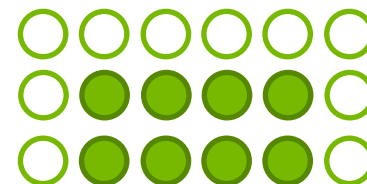
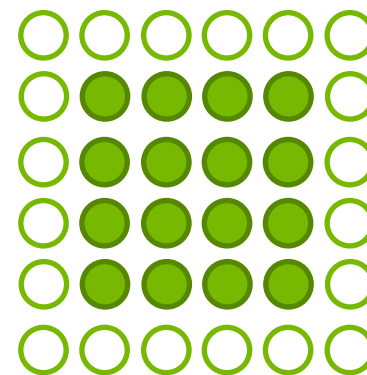
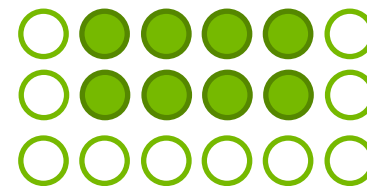


EXAMPLE JACOBI

Top/Bottom Halo

```
#pragma acc host_data use_device ( A )  
{
```

```
}
```



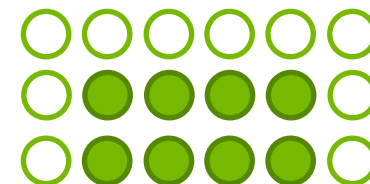
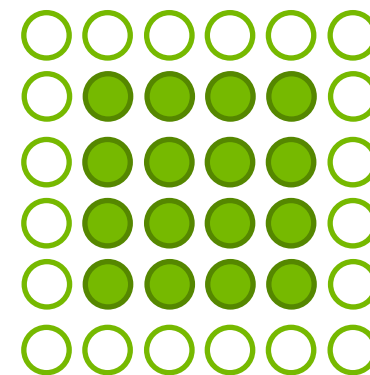
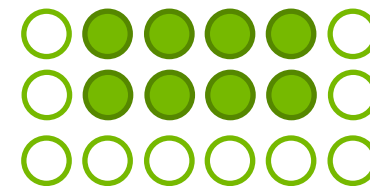
EXAMPLE JACOBI

Top/Bottom Halo

```
#pragma acc host_data use_device ( A )
{

MPI_Sendrecv(A+iy_start*nx+1, nx-2, MPI_DOUBLE, top, 0,
             A+iy_end*nx+1, nx-2, MPI_DOUBLE, bottom, 0,
             MPI_COMM_WORLD, MPI_STATUS_IGNORE);

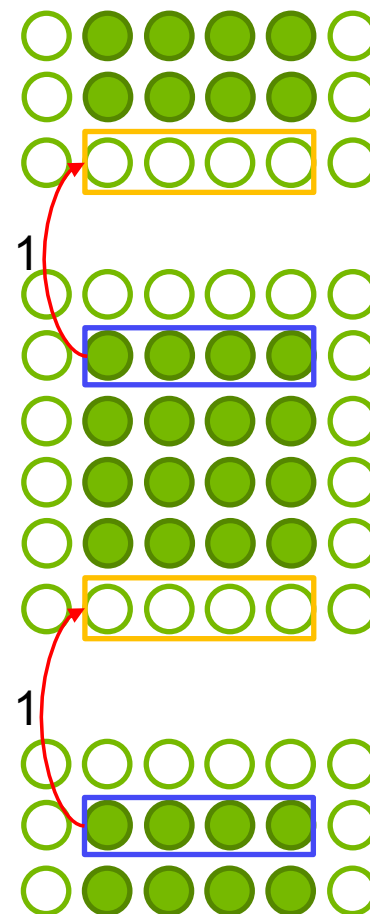
}
```



EXAMPLE JACOBI

Top/Bottom Halo

```
#pragma acc host_data use_device ( A )  
{  
MPI_Sendrecv(A+iy_start*nx+1, nx-2, MPI_DOUBLE, top, 0,  
             A+iy_end*nx+1, nx-2, MPI_DOUBLE, bottom, 0,  
             MPI_COMM_WORLD, MPI_STATUS_IGNORE);  
  
}
```

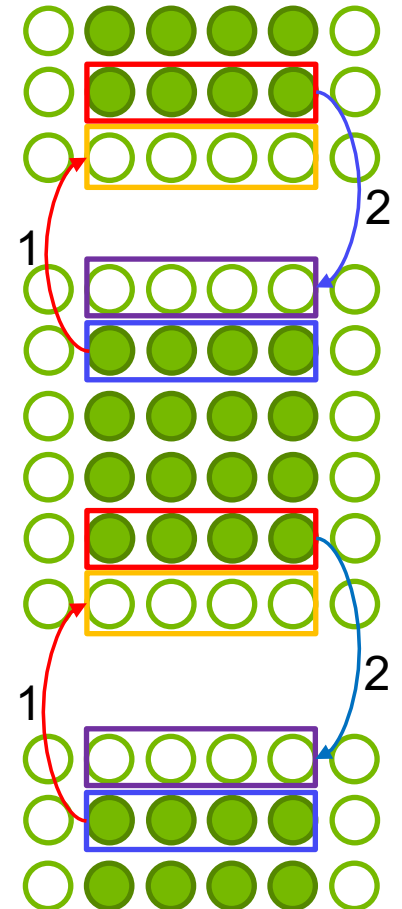


EXAMPLE JACOBI

Top/Bottom Halo

```
#pragma acc host_data use_device ( A )
{
    MPI_Sendrecv(A+iy_start*nx+1, nx-2, MPI_DOUBLE, top, 0,
                 A+iy_end*nx+1, nx-2, MPI_DOUBLE, bottom, 0,
                 MPI_COMM_WORLD, MPI_STATUS_IGNORE);

    MPI_Sendrecv(A+(iy_end-1)*nx+1, nx-2, MPI_DOUBLE, bottom, 1,
                 A+(iy_start-1)*nx+1, nx-2, MPI_DOUBLE, top, 1,
                 MPI_COMM_WORLD, MPI_STATUS_IGNORE);
}
```



HANDLING MULTI GPU NODES

GPU-affinity

```
#pragma acc set device_num( devicenum )

// Or using the API:

#if _OPENACC

acc_device_t device_type = acc_get_device_type();

int ngpus=acc_get_num_devices(device_type);

int devicenum=rank%ngpus;

acc_set_device_num(devicenum,device_type);

#endif /*_OPENACC*/
```

Alternative (OpenMPI / Spectrum MPI):

```
int devicenum = atoi(getenv("OMPI_COMM_WORLD_LOCAL_RANK"));
```

Alternative (MVAPICH2):

```
int devicenum = atoi(getenv("MV2_COMM_WORLD_LOCAL_RANK"));
```

An abstract network diagram with several glowing green nodes connected by thin, intersecting green lines. The background is dark blue/black with some blurred light spots.

PROFILING OF MPI+OPENACC APPS

PROFILING MPI+OPENACC APPLICATIONS

Using pgprof

Embed MPI rank in output filename, process name, and context name

```
mpirun -np $np pgprof --output-profile profile.%q{OMPI_COMM_WORLD_RANK}.nvvp
```

OpenMPI / Spectrum MPI: OMPI_COMM_WORLD_RANK

MVAPICH2: MV2_COMM_WORLD_RANK

Additionally highlight MPI activity in the timeline

```
mpirun -np $np pgprof --annotate-mpi openmpi --output-profile ...
```

OpenMPI / Spectrum MPI : --annotate-mpi openmpi

MVAPICH2: --annotate-mpi mpich

PROFILING MPI+OPENACC APPLICATIONS

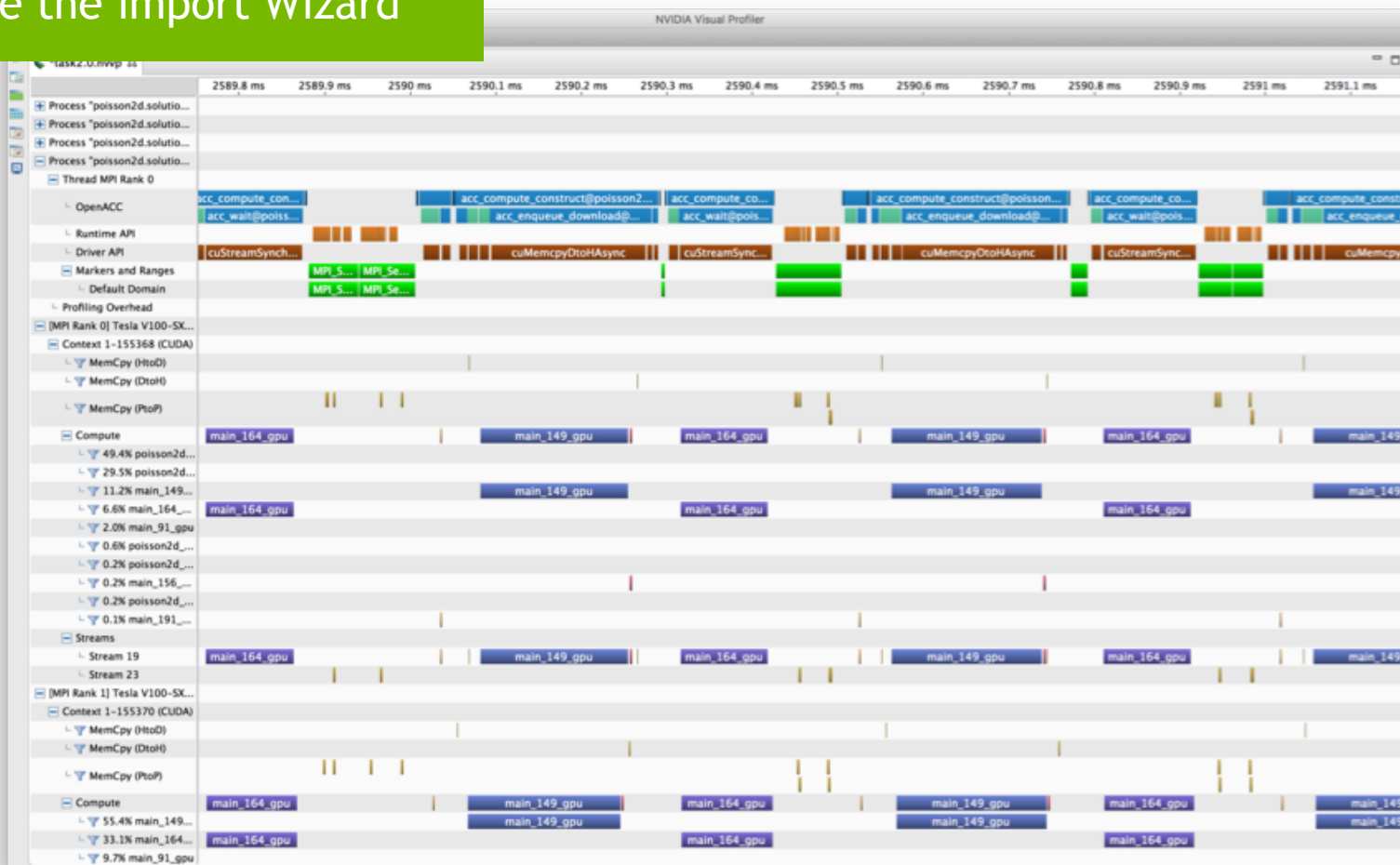
Using pgprof

```
ssh 5. ssh
bash-4.2$ jsrun --smpiargs "-gpu" -n1 -g ALL_GPUS -c ALL_CPUS -a4 pgprof --annotate-mpi openmpi --cpu-pro
filing off -o /gpfs/wolf/gen110/proj-shared/mathiasw/task2.%q{OMPI_COMM_WORLD_RANK}.nvvp ./poisson2d.solu
tion
==155052== PGPROF is profiling process 155052, command: ./poisson2d.solution
==155054== PGPROF is profiling process 155054, command: ./poisson2d.solution
==155055== PGPROF is profiling process 155055, command: ./poisson2d.solution
==155053== PGPROF is profiling process 155053, command: ./poisson2d.solution
==155052== Generated result file: /gpfs/wolf/gen110/proj-shared/mathiasw/task2.3.nvvp
==155054== Generated result file: /gpfs/wolf/gen110/proj-shared/mathiasw/task2.2.nvvp
==155053== Generated result file: /gpfs/wolf/gen110/proj-shared/mathiasw/task2.1.nvvp
==155055== Generated result file: /gpfs/wolf/gen110/proj-shared/mathiasw/task2.0.nvvp
Jacobi relaxation Calculation: 4096 x 4096 mesh
Calculate reference solution and time serial execution.
  0, 0.250000
 100, 0.249940
 200, 0.249880
 300, 0.249821
 400, 0.249761
 500, 0.249702
 600, 0.249642
 700, 0.249583
 800, 0.249524
 900, 0.249464
Parallel execution.
  0, 0.250000
 100, 0.249940
 200, 0.249880
 300, 0.249821
 400, 0.249761
 500, 0.249702
 600, 0.249642
 700, 0.249583
 800, 0.249524
 900, 0.249464
Num GPUs: 4.
4096x4096: 1 GPU: 1.3687 s, 4 GPUs: 0.5355 s, speedup: 2.56, efficiency: 63.90%
MPI time: 0.0882 s, inter GPU BW: 1.38 GiB/s
bash-4.2$
```

PROFILING MPI+OPENACC APPLICATIONS

Using pgprof

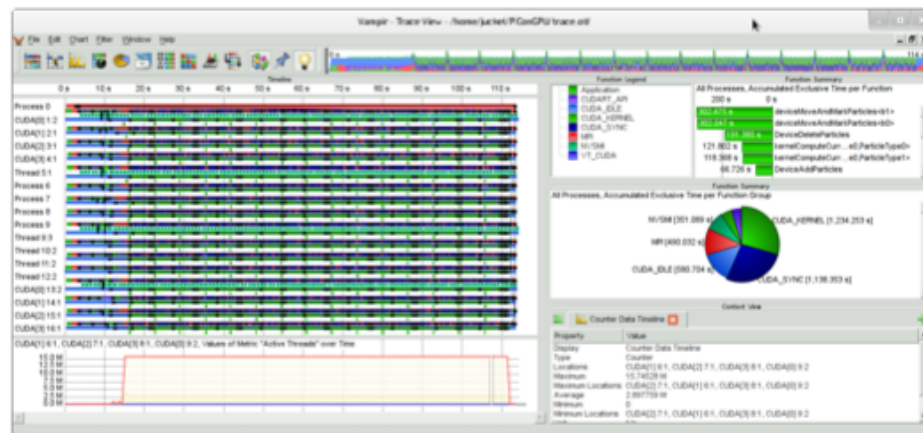
Use the import Wizard



Third Party Tools

Vampir

These tools are good for discovering MPI issues as well as basic OpenACC performance inhibitors.

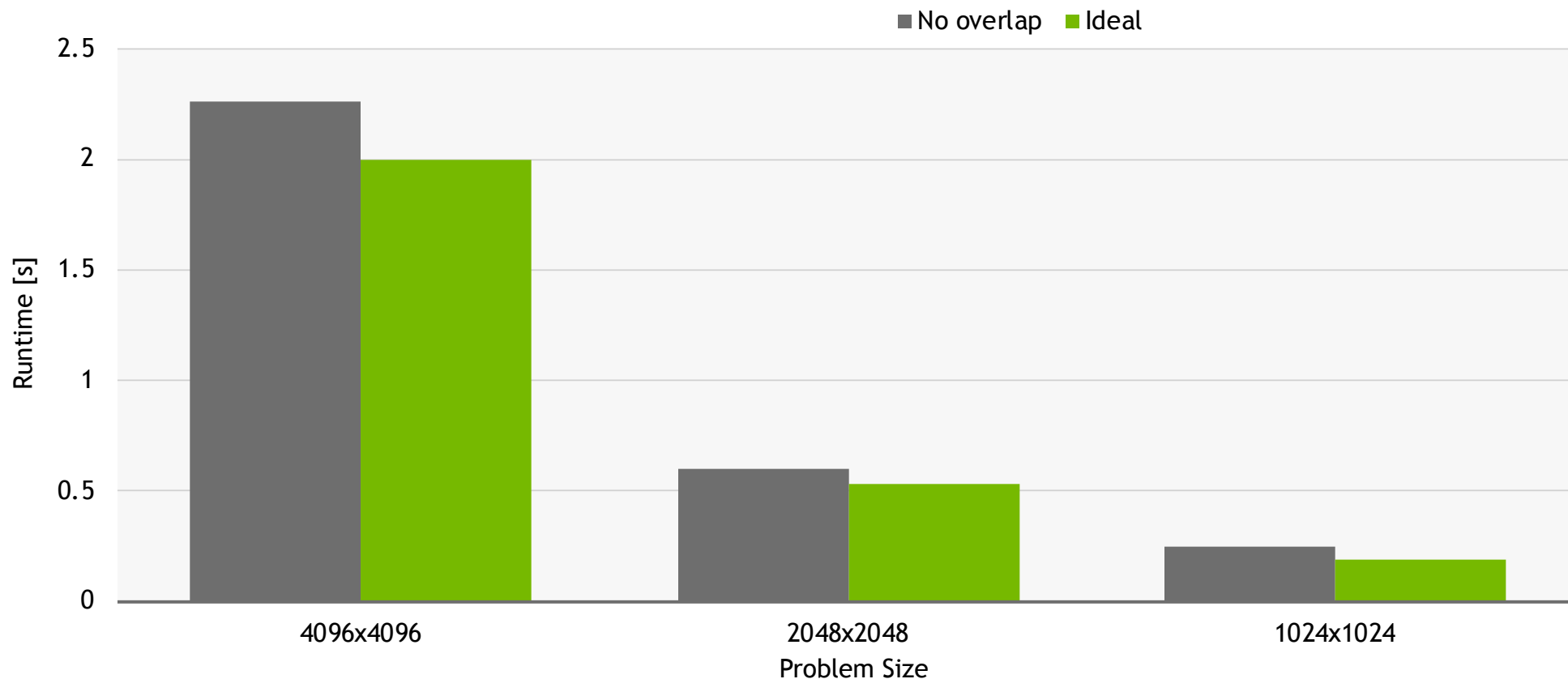


An abstract network diagram with a dark blue background. It features numerous small, bright green circular nodes connected by thin, light green lines. The lines crisscross the frame, creating a complex web of connections. Some nodes are more prominent than others, and the overall effect is one of a dynamic, interconnected system.

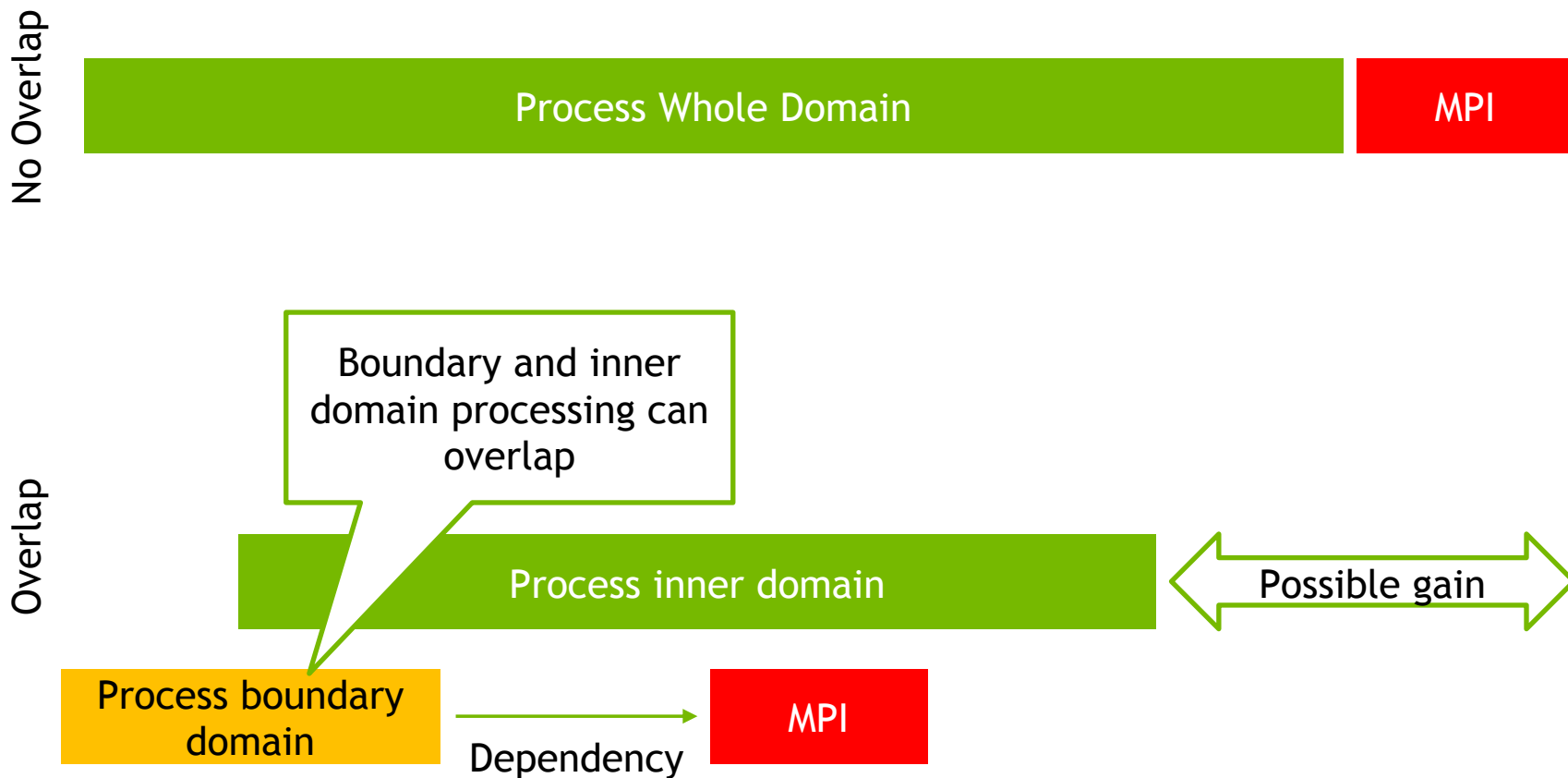
OVERLAPPING COMMUNICATION AND COMPUTATION

COMMUNICATION + COMPUTATION OVERLAP

OpenMPI 2.1.2 - 2 Quadro GV100



COMMUNICATION + COMPUTATION OVERLAP



COMMUNICATION + COMPUTATION OVERLAP

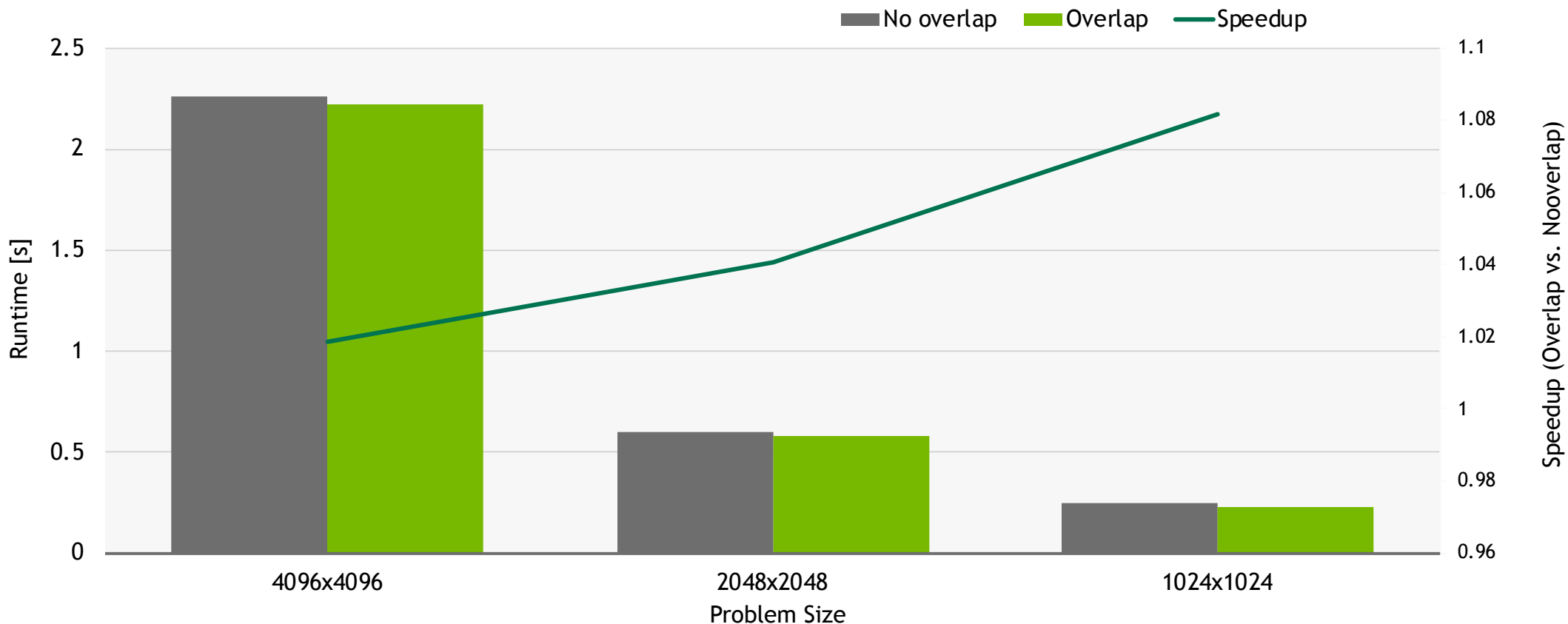
OpenACC with Async Queues

```
#pragma acc parallel loop present(A,Anew)
for ( ... ) //Process boundary

#pragma acc parallel loop present(A,Anew) async
for ( ... ) //Process inner domain
#pragma acc host_data use_device ( A ) {
    MPI_Sendrecv(A+iy_start*nx+1, nx-2, MPI_DOUBLE, top, 0,
                 A+iy_end*nx+1, nx-2, MPI_DOUBLE, bottom, 0,
                 MPI_COMM_WORLD, MPI_STATUS_IGNORE);
    MPI_Sendrecv(A+(iy_end-1)*nx+1, nx-2, MPI_DOUBLE, bottom, 1,
                 A+(iy_start-1)*nx+1, nx-2, MPI_DOUBLE, top, 1,
                 MPI_COMM_WORLD, MPI_STATUS_IGNORE);
}
#pragma acc wait //wait for iteration to finish
```

COMMUNICATION + COMPUTATION OVERLAP

OpenMPI 2.1.2 - 2 Quadro GV100



An abstract network diagram with a dark background. It features several bright green nodes connected by thin, light green lines, creating a complex web of connections. The nodes are scattered across the frame, with some appearing as small dots and others as larger, more prominent circles. The lines vary in thickness and brightness, suggesting different types of connections or data flow.

MPI AND UNIFIED MEMORY

MPI AND UNIFIED MEMORY

CAVEAT

Using Unified Memory with a *non Unified Memory-aware* MPI might break in some cases, e.g. when registering memory for RDMA, or even worse silently produce wrong results.



Use a Unified Memory-aware MPI with Unified Memory and MPI

Unified Memory-aware: CUDA-aware MPI with support for Unified Memory

MPI AND UNIFIED MEMORY

Current Status

Available Unified Memory-aware MPI implementations

- OpenMPI (since 1.8.5)
- MVAPICH2-GDR (since 2.2b)
- Spectrum MPI

Currently treat all Unified Memory as Device Memory

Good performance if all buffers used in MPI are touched mainly on the GPU.

MPI AND UNIFIED MEMORY

Without Unified Memory-aware MPI

Only use non Unified Memory Buffers for MPI: cudaMalloc, cudaMallocHost or malloc

Application managed non Unified Memory Buffers also allow to work around current missing cases in Unified Memory-aware MPI Implementations.

