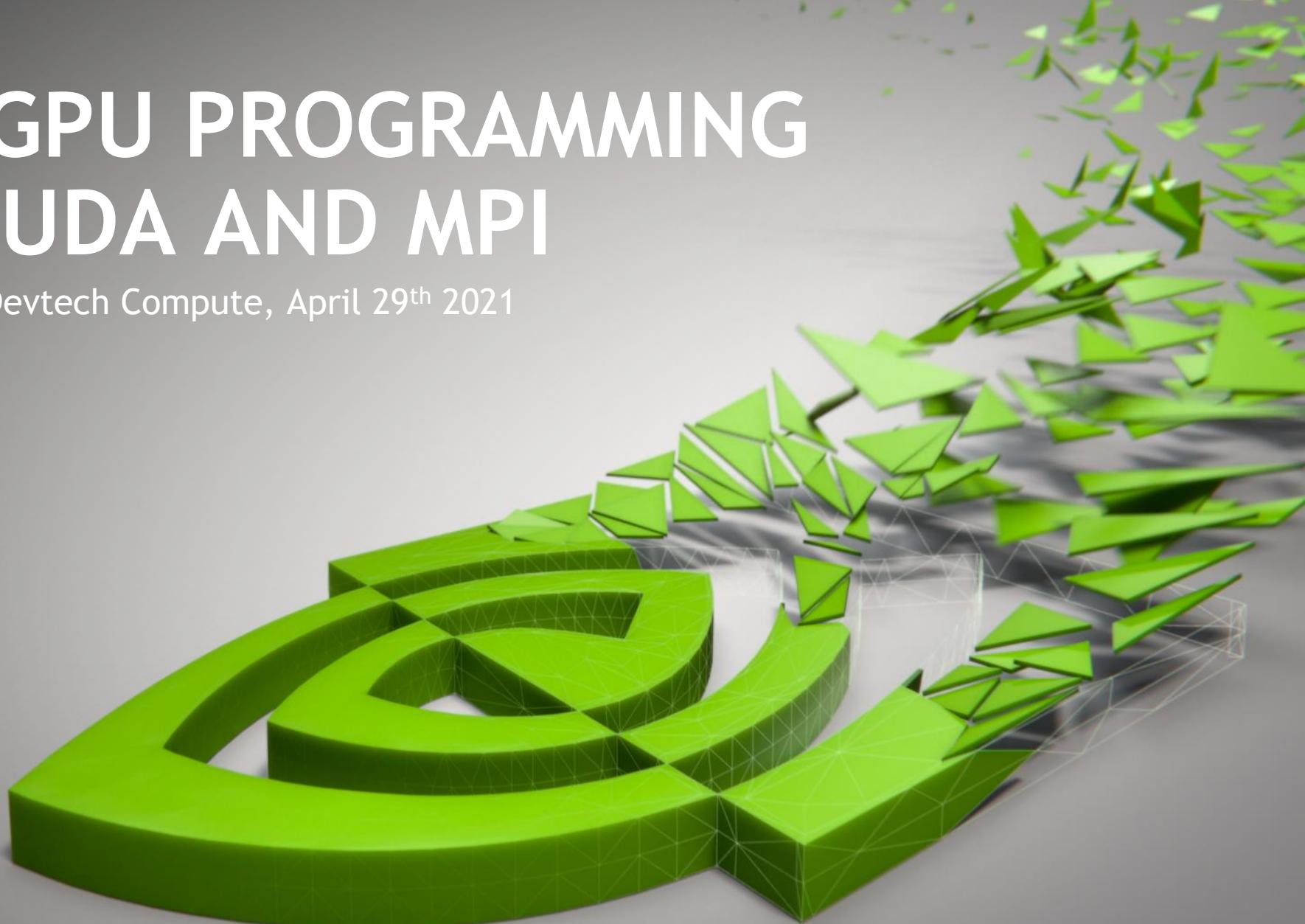
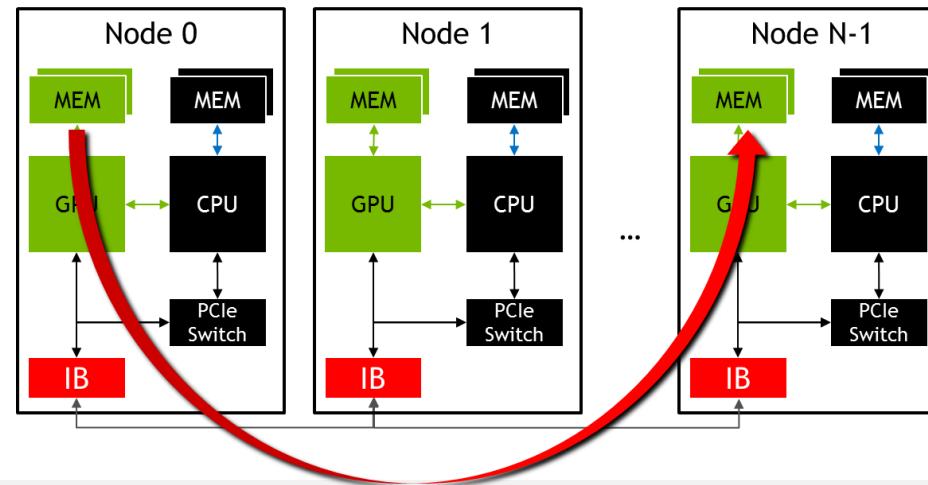


MULTI GPU PROGRAMMING WITH CUDA AND MPI

Jiri Kraus, Senior Devtech Compute, April 29th 2021



MPI+CUDA



```
//MPI rank 0  
MPI_Send(sbuf_d, size, MPI_DOUBLE, n-1, tag, MPI_COMM_WORLD);  
  
//MPI rank n-1  
MPI_Recv(rbuf_d, size, MPI_DOUBLE, 0, tag, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
```

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 Platform MPI, Cray MPT, ...

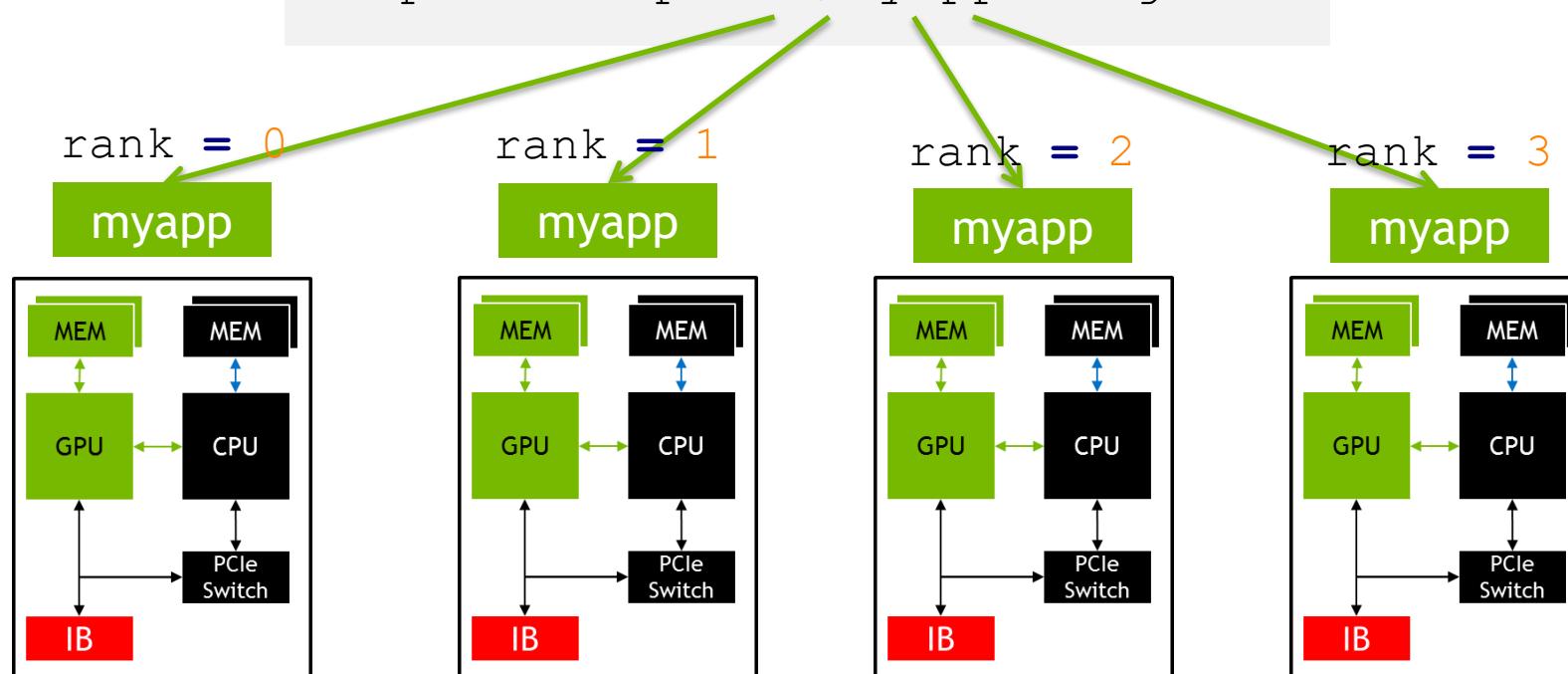
MPI - SKELETON

```
#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

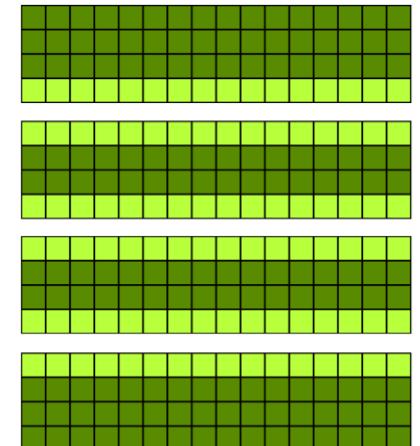
Solves the 2D-Laplace Equation on a rectangle

$$\Delta u(x, y) = 0 \quad \forall (x, y) \in \Omega \setminus \delta\Omega$$

Dirichlet boundary conditions (constant values on boundaries) on left and right boundary

Periodic boundary conditions on top and bottom boundary

Domain decomposition with stripes



Horizontal Stripes

EXAMPLE: JACOBI SOLVER

Single GPU

While not converged

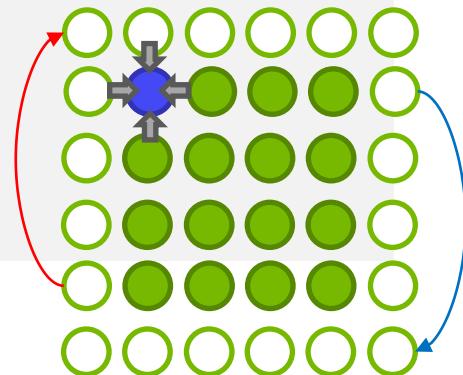
Do Jacobi step:

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

Apply periodic boundary conditions

swap a_new and a

Next iteration



EXAMPLE: JACOBI SOLVER

Multi GPU

While not converged

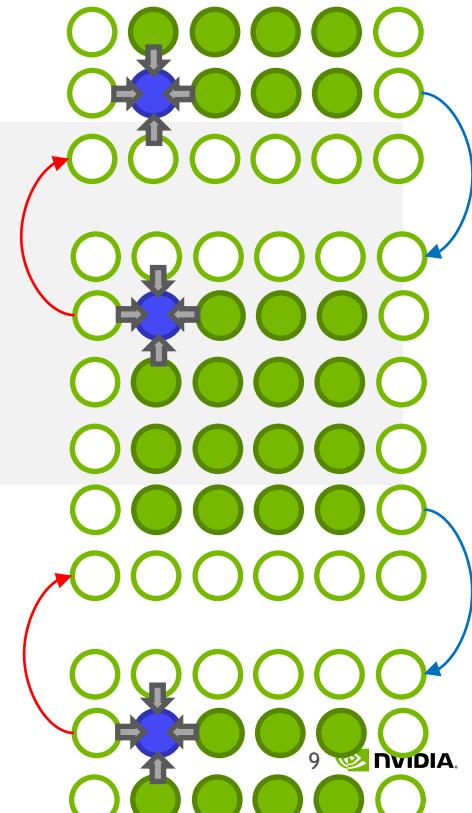
Do Jacobi step:

```
for (int iy = iy_start; iy < iy_end; ++iy)  
  
    for (int ix = 1; ix < (nx-1); ++ix)  
  
        a_new[iy*nx+ix] = 0.25f*((a[ iy    *nx + ix+1]+a[ iy    *nx + ix-1]  
                                +a[ (iy+1)*nx + ix] +a[ (iy-1)*nx + ix]));
```

Apply periodic boundary conditions and exchange halo with 2 neighbors

swap a_new and a

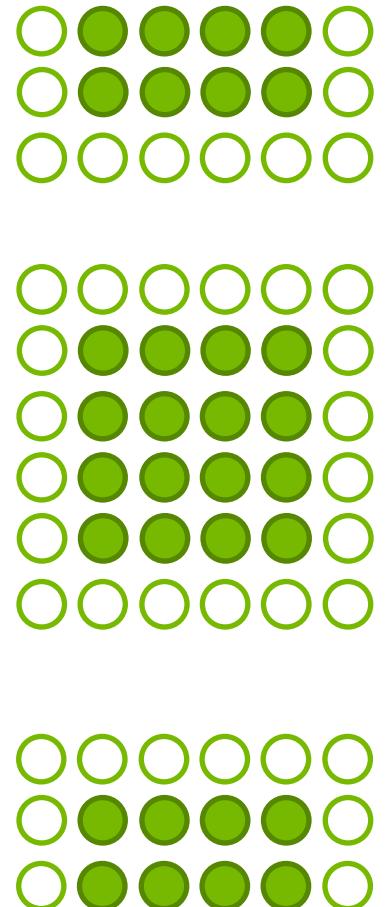
Next iteration



EXAMPLE JACOBI

Top/Bottom Halo

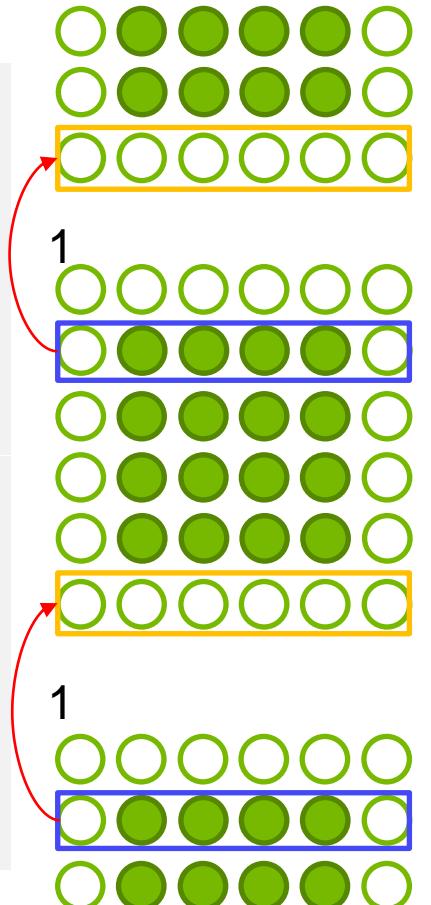
```
MPI_Sendrecv(a_new+iy_start*nx, nx, MPI_DOUBLE, top, 0,  
             a_new+iy_end*nx,     nx, MPI_DOUBLE, bottom, 0,  
             MPI_COMM_WORLD, MPI_STATUS_IGNORE);
```



EXAMPLE JACOBI

Top/Bottom Halo

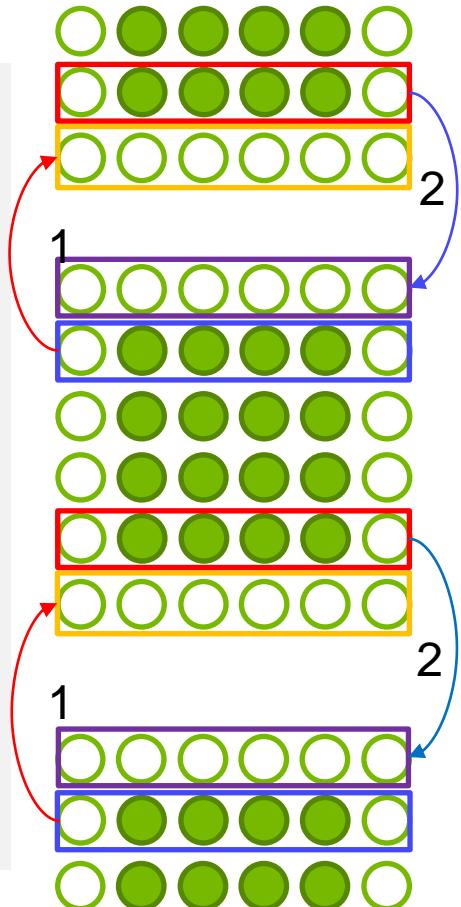
```
MPI_Sendrecv(a new+iy start*nx, nx, MPI_DOUBLE, top, 0,  
a new+iy end*nx, nx, MPI_DOUBLE, bottom, 0,  
MPI_COMM_WORLD, MPI_STATUS_IGNORE);
```



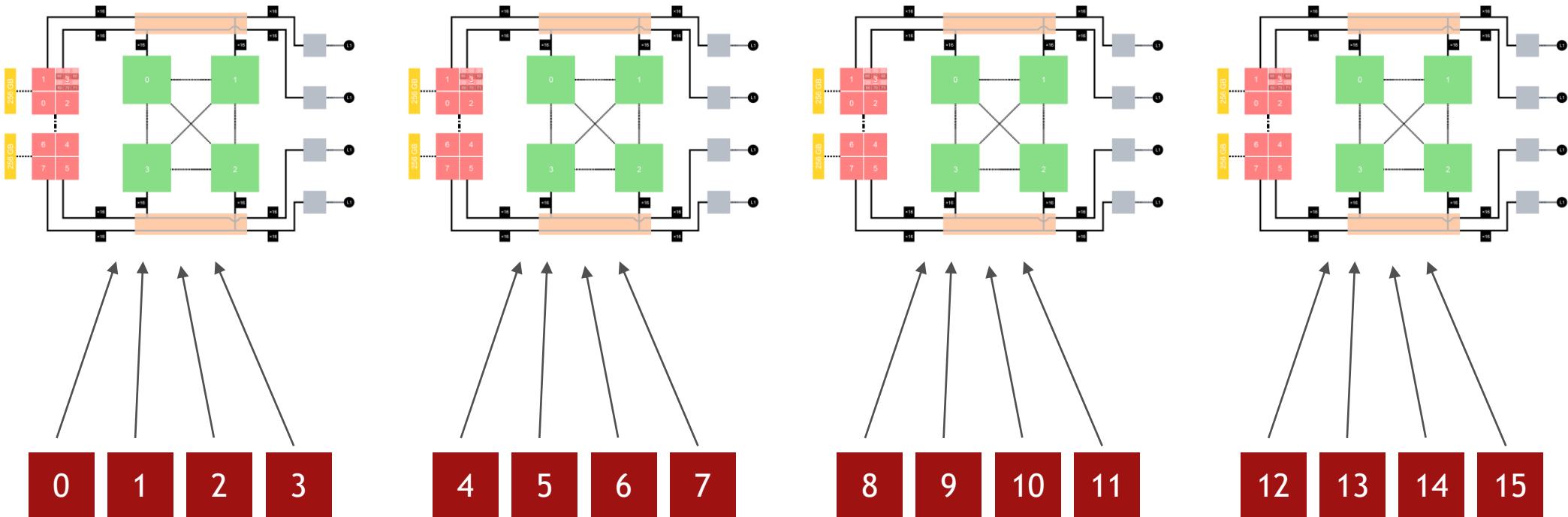
EXAMPLE JACOBI

Top/Bottom Halo

```
MPI_Sendrecv(a new+iy start*nx, nx, MPI_DOUBLE, top, 0,  
a new+iy end*nx, nx, MPI_DOUBLE, bottom, 0,  
MPI_COMM_WORLD, MPI_STATUS_IGNORE);  
  
MPI_Sendrecv(a new+(iy end-1)*nx, nx, MPI_DOUBLE, bottom, 1,  
a new+(iy start-1)*nx, nx, MPI_DOUBLE, top, 1,  
MPI_COMM_WORLD, MPI_STATUS_IGNORE);
```



HANDLING MULTI GPU NODES

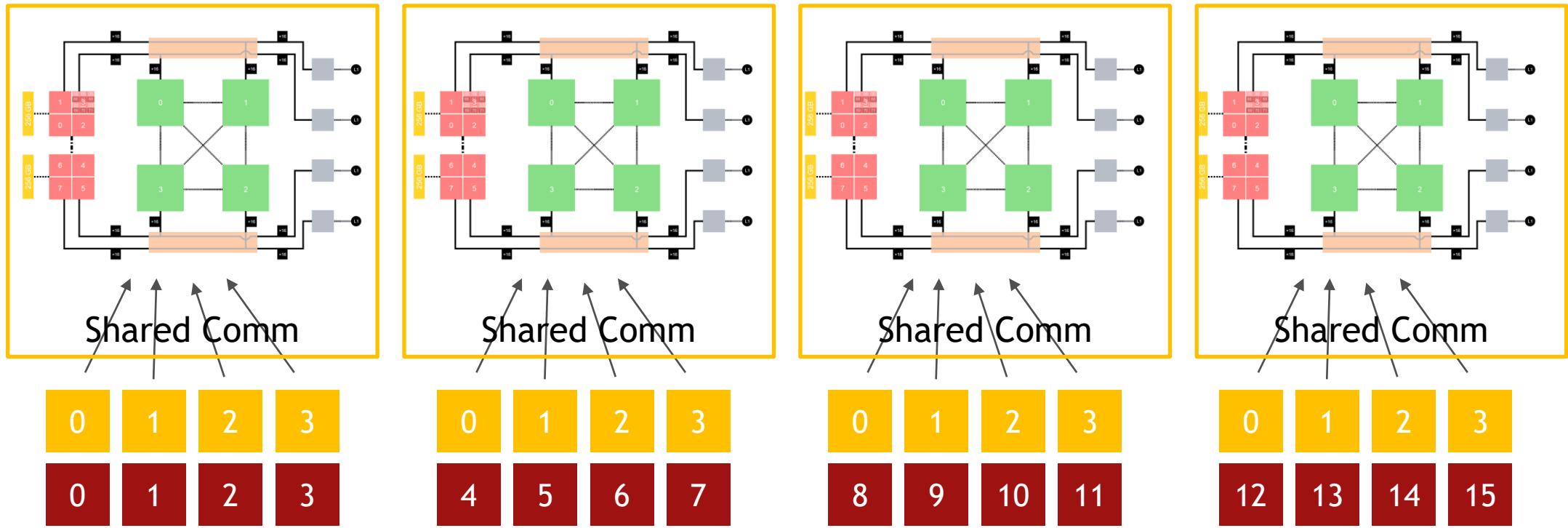


HANDLING MULTI GPU NODES

How to determine the local rank? - MPI-3

```
MPI_Comm local_comm;  
  
MPI_Info info;  
  
MPI_Info_create(&info);  
  
MPI_Comm_split_type(MPI_COMM_WORLD, MPI_COMM_TYPE_SHARED, rank, info, &local_comm);  
  
int local_rank = -1;  
  
MPI_Comm_rank(local_comm, &local_rank);  
  
MPI_Comm_free(&local_comm);  
  
MPI_Info_free(&info);  
  
cudaSetDevice(local_rank);
```

HANDLING MULTI GPU NODES



HANDLING MULTI GPU NODES

GPU-affinity

Use local rank:

```
int local_rank = //determine local rank  
int num_devs = 0;  
cudaGetDeviceCount(&num_devs);  
cudaSetDevice(local_rank%num_devs);
```

Needed if
resource manager
handles GPU
affinity

PROFILING OF MPI+CUDA APPS

PROFILING MPI+CUDA APPLICATIONS

Using Nsight Systems

Embed MPI rank in output filename and trace MPI (PMI-2)

```
srun -n $np nsys profile --trace=mpi,cuda,nvtx profile.%q{PMI_RANK} \
./application
```

Slurm (PMI-2): PMI_RANK
Slurm (PMIX): PMIX_RANK
OpenMPI: OMPI_COMM_WORLD_RANK
MVAPICH2: MV2_COMM_WORLD_RANK

PROFILING MPI+CUDA APPLICATIONS

Using Nsight Systems

The screenshot shows two terminal windows on a Linux system. The left window displays the command-line interface for running a MPI+CUDA application and generating a profile report. The right window shows the output of the application execution and the processing of the generated reports.

```
[kraus1@jwlogin23 workspace]$ make profile
srun --partition booster --gres=gpu:4 --time 0:10:00 --pty -n 4 nsys profile --trace=mpi,cuda,nvtx -o jacobi.%q{PMI_RANK} ./jacobi -niter 10
srun: job 3602419 queued and waiting for resources
srun: job 3602419 has been allocated resources
Warning: LBR backtrace method is not supported on this platform. DWARF backtrace method will be used.
WARNING: The command line includes a target application therefore the CPU context-switch scope has been set to process-tree.
Warning: LBR backtrace method is not supported on this platform. DWARF ba
Warning: The command line includes a target application therefore the CPU
Warning: LBR backtrace method is not supported on this platform. DWARF ba
Warning: The command line includes a target application therefore the CPU
Warning: LBR backtrace method is not supported on this platform. DWARF ba
Warning: The command line includes a target application therefore the CPU
Collecting data...
Collecting data...
Collecting data...
Collecting data...
Single GPU jacobi relaxation: 10 iterations on 8192 x 8192 mesh
  0, 22.626007
Jacobi relaxation: 10 iterations on 8192 x 8192 mesh
  0, 22.626051
Num GPUs: 4.
8192x8192: 1 GPU:  0.0842 s, 4 GPUs:  0.0591 s, speedup:  1.42, effi
Processing events...
Processing events...
Processing events...
Processing events...
|
```

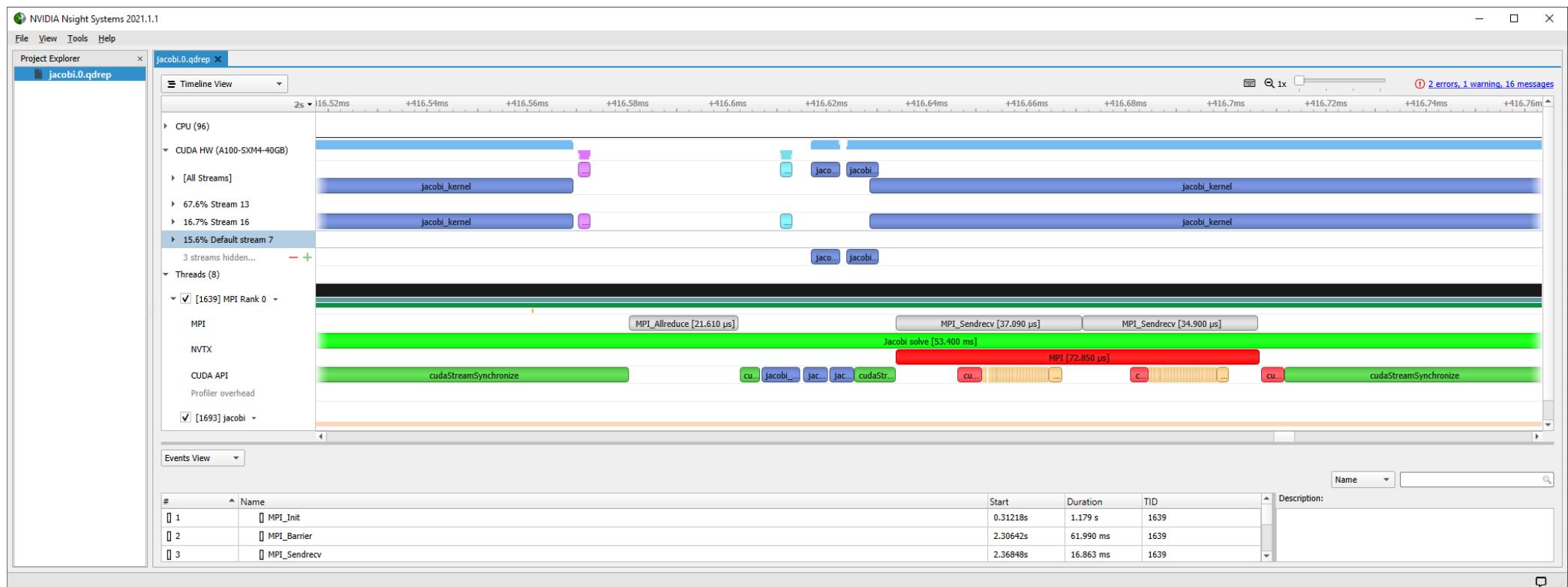
```
kraus1@jwlogin23 workspace]$ 
Saving temporary "/tmp/nsys-report-10c7-1bbb-3c85-299d.qdstrm" file to disk...
Creating final output files...
Creating final output files...
Saving temporary "/tmp/nsys-report-1342-dc76-d02f-993d.qdstrm" file to disk...
Creating final output files...
Creating final output files...
Saving temporary "/tmp/nsys-report-e423-4dfc-e629-fd9e.qdstrm" file to disk...
Creating final output files...
Creating final output files...
Processing [=====100%]
Processing [=====100%]
Processing [=====100%]
Processing [=====100%]
Saved report file to "/tmp/nsys-report-10c7-1bbb-3c85-299d.qdrep"
Saved report file to "/tmp/nsys-report-e423-4dfc-e629-fd9e.qdrep"
Saved report file to "/tmp/nsys-report-1342-dc76-d02f-993d.qdrep"
Saved report file to "/tmp/nsys-report-9c08-0209-133b-cea1.qdrep"
Report file moved to "/p/home/jusers/kraus1/juwels/workspace/cuda/5-Multi_GPU_MPI/exercises/solution3/jacobi.3.qdrep"
Report file moved to "/p/home/jusers/kraus1/juwels/workspace/cuda/5-Multi_GPU_MPI/exercises/solution3/jacobi.2.qdrep"

Report file moved to "/p/home/jusers/kraus1/juwels/workspace/cuda/5-Multi_GPU_MPI/exercises/solution3/jacobi.0.qdrep"
Report file moved to "/p/home/jusers/kraus1/juwels/workspace/cuda/5-Multi_GPU_MPI/exercises/solution3/jacobi.1.qdrep"

[kraus1@jwlogin23 workspace]$ ls *.qdrep
jacobi.0.qdrep jacobi.1.qdrep jacobi.2.qdrep jacobi.3.qdrep
[kraus1@jwlogin23 workspace]$ |
```

PROFILING MPI+CUDA APPLICATIONS

Using Nsight Systems



PROFILING MPI+CUDA APPLICATIONS

Third Party Tools

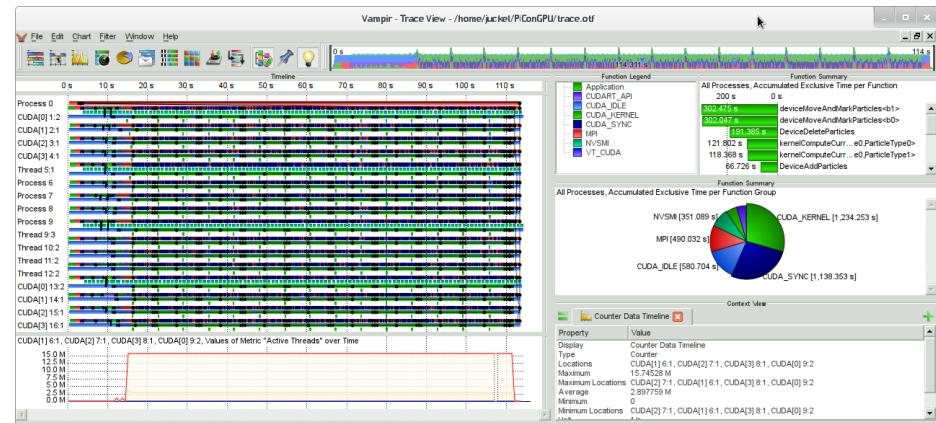
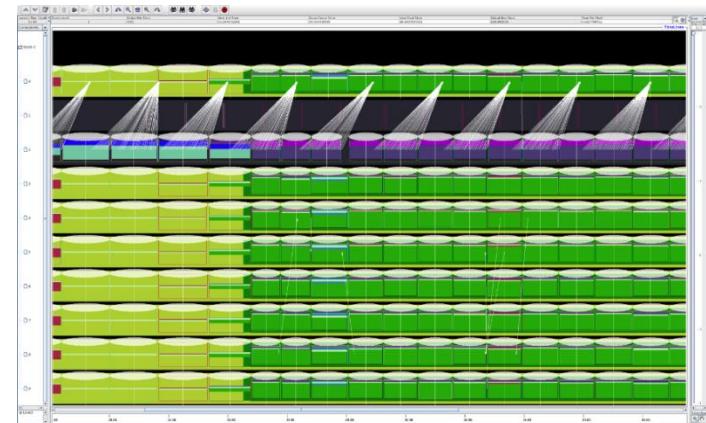
Multiple parallel profiling tools are CUDA-aware

Score-P

Vampir

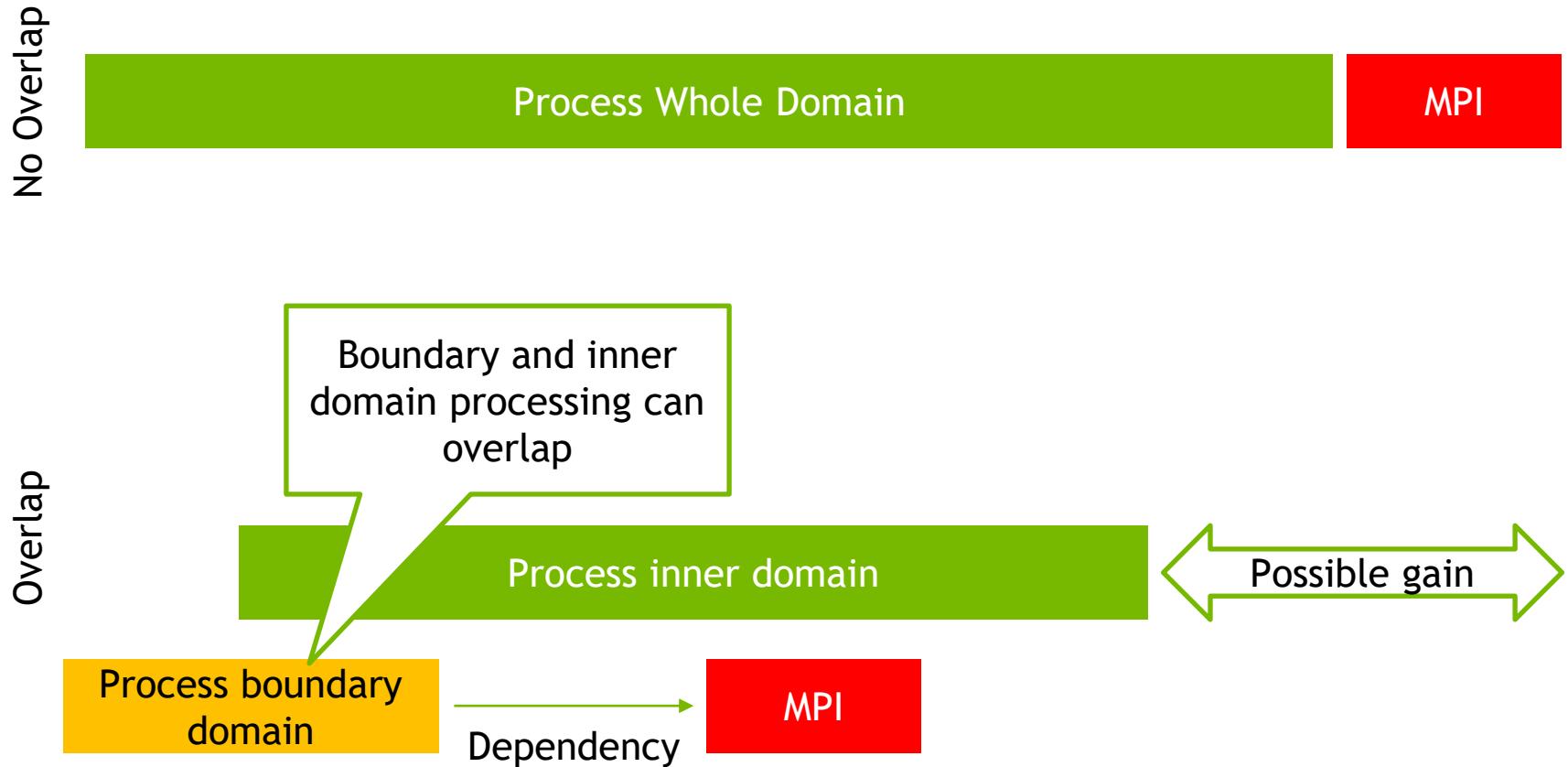
Tau

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



OVERLAPPING MPI AND COMPUTATION

COMMUNICATION + COMPUTATION OVERLAP



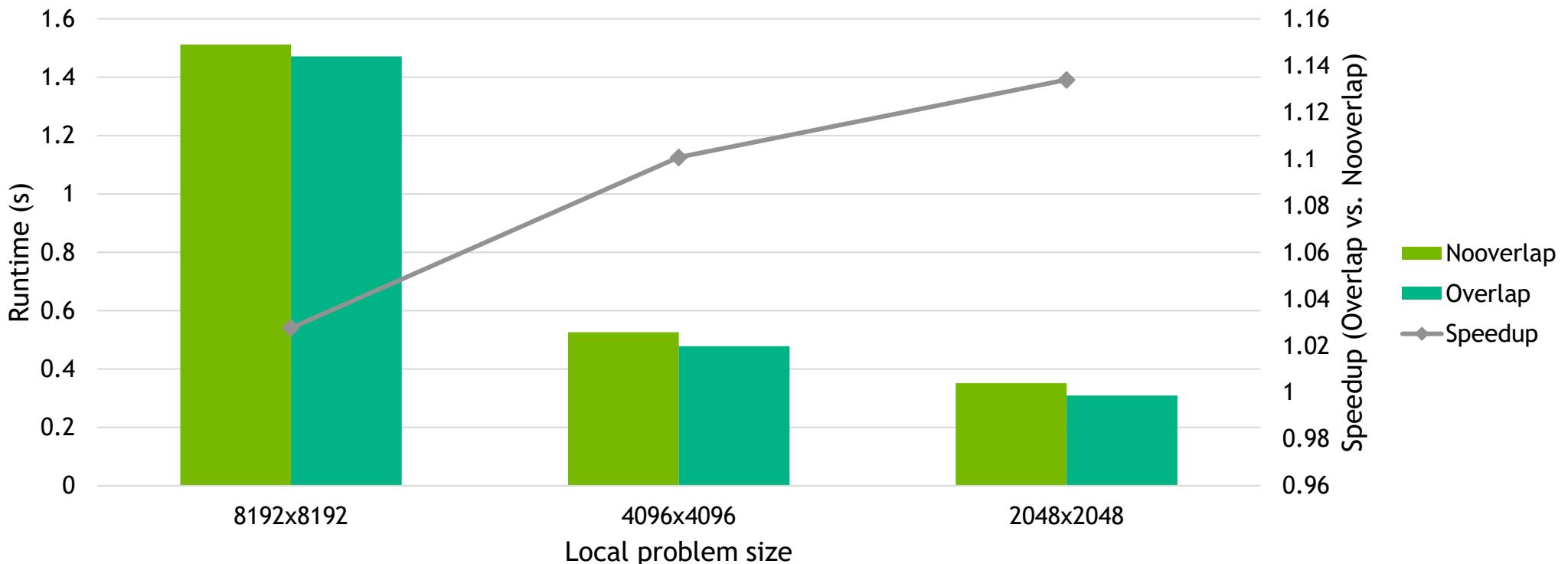
COMMUNICATION + COMPUTATION OVERLAP

Asynchronous execution with CUDA streams

```
launch_jacobi_kernel( a_new, a, l2_norm_m, iy_start, (iy_start+1), nx, halo_stream );
launch_jacobi_kernel( a_new, a, l2_norm_m, (iy_end-1), iy_end, nx, halo_stream );
launch_jacobi_kernel( a_new, a, l2_norm_m, (iy_start+1), (iy_end-1), nx, compute_stream );
int top = rank > 0 ? rank - 1 : (size-1); int bottom = (rank+1)%size;
//Apply periodic boundary conditions
cudaStreamSynchronize( halo_stream );
MPI_Sendrecv( a_new+iy_start*nx, nx, MPI_REAL_TYPE, top , 0,
              a_new+(iy_end*nx), nx, MPI_REAL_TYPE, bottom, 0,
              MPI_COMM_WORLD, MPI_STATUS_IGNORE );
MPI_Sendrecv( a_new+(iy_end-1)*nx, nx, MPI_REAL_TYPE, bottom, 0,
              a_new+(iy_start-1)*nx, nx, MPI_REAL_TYPE, top, 0,
              MPI_COMM_WORLD, MPI_STATUS_IGNORE );
cudaStreamSynchronize( compute_stream );
```

COMMUNICATION + COMPUTATION OVERLAP

ParaStationMPI/5.4.7-1 with UCX 1.8.1 - 4 A100 40GB



HANDS-ON

HANDS-ON MENU

4 tasks to choose from

Task 0: Using MPI

Task 1: Handle GPU Affinity

Task 2: Apply Domain Decomposition

Task 3: Overlap MPI and Compute

TASK 0: USING MPI

task0

Determine rank (`MPI_Comm_rank`) and size (`MPI_Comm_size`)

Add `MPI_Barrier` to ensure correct timing

```
Num GPUs: 1.  
8192x8192: 1 GPU: 5.4254 s, 1 GPUs: 5.3043 s, speedup:
```

```
Num GPUs: 1.  
8192x8192: 1 GPU: 5.4472 s, 1 GPUs: 5.3302 s, speedup:
```

```
Num GPUs: 1.  
8192x8192: 1 GPU: 5.4287 s, 1 GPUs: 5.3236 s, speedup:
```

```
Num GPUs: 1.  
8192x8192: 1 GPU: 5.4554 s, 1 GPUs: 5.3211 s, speedup:
```

Look for TODOs

Make Targets:

run: run jacobi with \$NP procs.
jacobi: build jacobi bin (default)
sanitize: run with compute-sanitizer
profile: profile with Nsight Systems

Solution is in solution3

<https://www.open-mpi.org/doc/current/>

TASK 1: HANDLING GPU AFFINITY

task1

Run with CUDA_VISIBLE_DEVICES=0,1,2,3

Handle GPU affinity with MPI_COMM_TYPE_SHARED

Run and report the performance

Look for TODOs

```
900, 0.122885
```

```
Num GPUs: 4.
```

```
8192x8192: 1 GPU: 25.1564 s, 4 GPUs: 24.4987 s, speedup: 1.03
```

Make Targets:

run: run jacobi with \$NP procs.
jacobi: build jacobi bin (default)
sanitize: run with compute-sanitizer
profile: profile with Nsight Systems

Solution is in solution1

<https://www.open-mpi.org/doc/current/>

TASK 2: APPLY DOMAIN DECOMPOSITION

task2

Calculate first (`iy_start`) and last (`iy_end`) row to be processed by each rank

Use `MPI_Sendrecv` to handle halo updates and periodic boundary conditions

Use `MPI_Allreduce` to calculate global L2 norm

Look for TODOs

```
900, 0.122885
Num GPUs: 4.
8192x8192: 1 GPU: 5.4283 s, 4 GPUs: 5.3205 s, speedup: 1.02
```

Make Targets:

run: run jacobi with \$NP procs.
jacobi: build jacobi bin (default)
sanitize: run with compute-sanitizer
profile: profile with Nsight Systems

Solution is in solution2

<https://www.open-mpi.org/doc/current/>

TASK 3: OVERLAP MPI AND COMPUTE

task3

Use `cudaStreamCreate` to create halo processing stream

Split jacobi step in top boundary, bottom boundary and bulk part

Launch top and bottom boundary part in halo processing stream

Look for TODOs

```
900, 0.122891
Num GPUs: 4.
8192x8192: 1 GPU: 5.4247 s, 4 GPUs: 1.4978 s, speedup: 3.62
```

Make Targets:

run: run jacobi with \$NP procs.
jacobi: build jacobi bin (default)
sanitize: run with compute-sanitizer
profile: profile with Nsight Systems

Solution is in solution3

<https://www.open-mpi.org/doc/current/>

SOLUTIONS

TASK 0: USING MPI

Solution

```
int main(int argc, char * argv[]) {
    int rank = 0;
    int size = 1;
    MPI_CALL( MPI_Init(&argc,&argv) );
    MPI_CALL( MPI_Comm_rank(MPI_COMM_WORLD,&rank) );
    MPI_CALL( MPI_Comm_size(MPI_COMM_WORLD,&size) );
    //...
    MPI_CALL( MPI_Barrier(MPI_COMM_WORLD) );
    double start = MPI_Wtime();
    while ( l2_norm > tol && iter < iter_max )
    //...
    MPI_CALL( MPI_Finalize() );
    return result_correct == 1 ? 0 : 1; }
```

TASK 1: HANDLING GPU AFFINITY

Solution

```
int dev_id = -1;
MPI_Comm local_comm;
MPI_Info info;
MPI_CALL( MPI_Info_create(&info) );
MPI_CALL( MPI_Comm_split_type(MPI_COMM_WORLD, MPI_COMM_TYPE_SHARED,
                           rank, info, &local_comm) );
MPI_CALL( MPI_Comm_rank(local_comm,&dev_id) );
MPI_CALL( MPI_Comm_free(&local_comm) );
MPI_CALL( MPI_Info_free(&info) );
int num_devs = 0;
CUDA_RT_CALL( cudaGetDeviceCount( &num_devs ) );
dev_id = dev_id % num_devs;
CUDA_RT_CALL( cudaSetDevice( dev_id ) );
```

TASK 2: APPLY DOMAIN DECOMPOSITION

Solution I

```
// Ensure correctness if ny%size != 0
int chunk_size = std::ceil( (1.0*ny)/size );
int iy_start = rank*chunk_size;
int iy_end = iy_start+chunk_size;
// Do not process boundaries
iy_start = std::max( iy_start, 1 );
iy_end = std::min( iy_end, ny - 1 );
```

TASK 2: APPLY DOMAIN DECOMPOSITION

Solution II

```
//Apply periodic boundary conditions
CUDA_RT_CALL( cudaStreamSynchronize( compute_stream ) );
PUSH_RANGE("MPI", 5)
MPI_CALL( MPI_Sendrecv( a_new+iy_start*nx, nx, MPI_REAL_TYPE, top , 0,
                        a_new+(iy_end*nx), nx, MPI_REAL_TYPE, bottom, 0,
                        MPI_COMM_WORLD, MPI_STATUS_IGNORE ) );
MPI_CALL( MPI_Sendrecv( a_new+(iy_end-1)*nx, nx, MPI_REAL_TYPE, bottom, 0,
                        a_new+(iy_start-1)*nx, nx, MPI_REAL_TYPE, top , 0,
                        MPI_COMM_WORLD, MPI_STATUS_IGNORE ) );
POP_RANGE
```

TASK 2: APPLY DOMAIN DECOMPOSITION

Solution III

```
CUDA_RT_CALL( cudaStreamSynchronize( compute_stream ) );
MPI_CALL( MPI_Allreduce( l2_norm_m, &l2_norm, 1, MPI_REAL_TYPE,
                         MPI_SUM, MPI_COMM_WORLD ) );
l2_norm = std::sqrt( l2_norm );
```

TASK 3: OVERLAP MPI AND COMPUTE

Solution

```
launch_jacobi_kernel( a_new, a, l2_norm_d,
                      iy_start, (iy_start+1), nx, halo_stream );
launch_jacobi_kernel( a_new, a, l2_norm_d,
                      (iy_end-1), iy_end, nx, halo_stream );
launch_jacobi_kernel( a_new, a, l2_norm_d, (iy_start+1),
                      (iy_end-1), nx, compute_stream );
int top = rank > 0 ? rank - 1 : (size-1); int bottom = (rank+1)%size;
//Apply periodic boundary conditions
CUDA_RT_CALL( cudaStreamSynchronize( halo_stream ) );
PUSH_RANGE("MPI",5)
MPI_CALL( MPI_Sendrecv( a_new+iy_start*nx, ...
```

TASK 3: OVERLAP MPI AND COMPUTE

Solution

