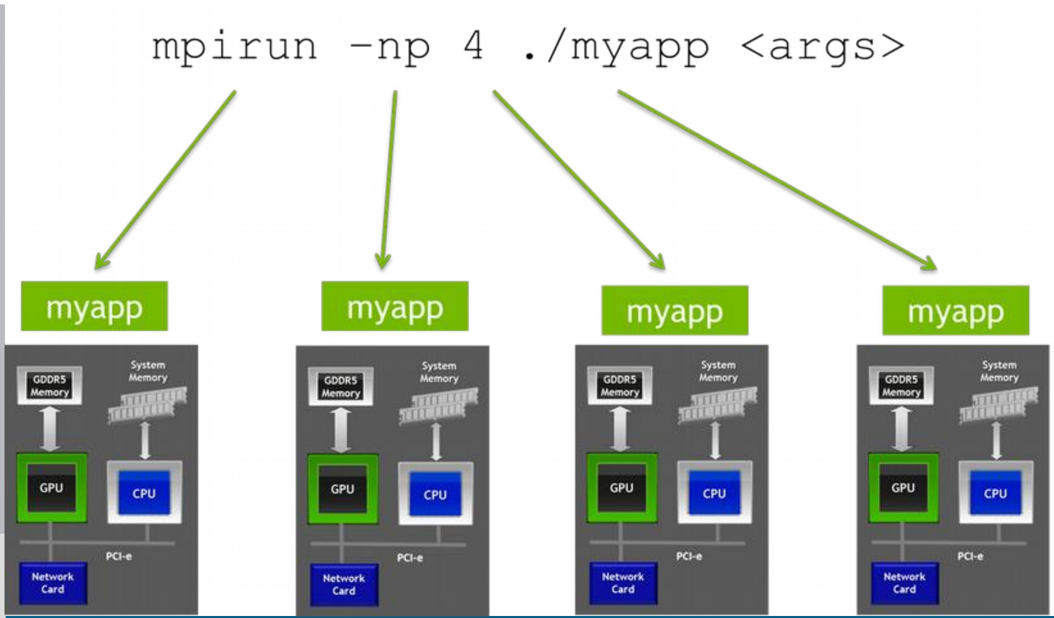
Mitglied der Helmholtz-Gemeinschaft



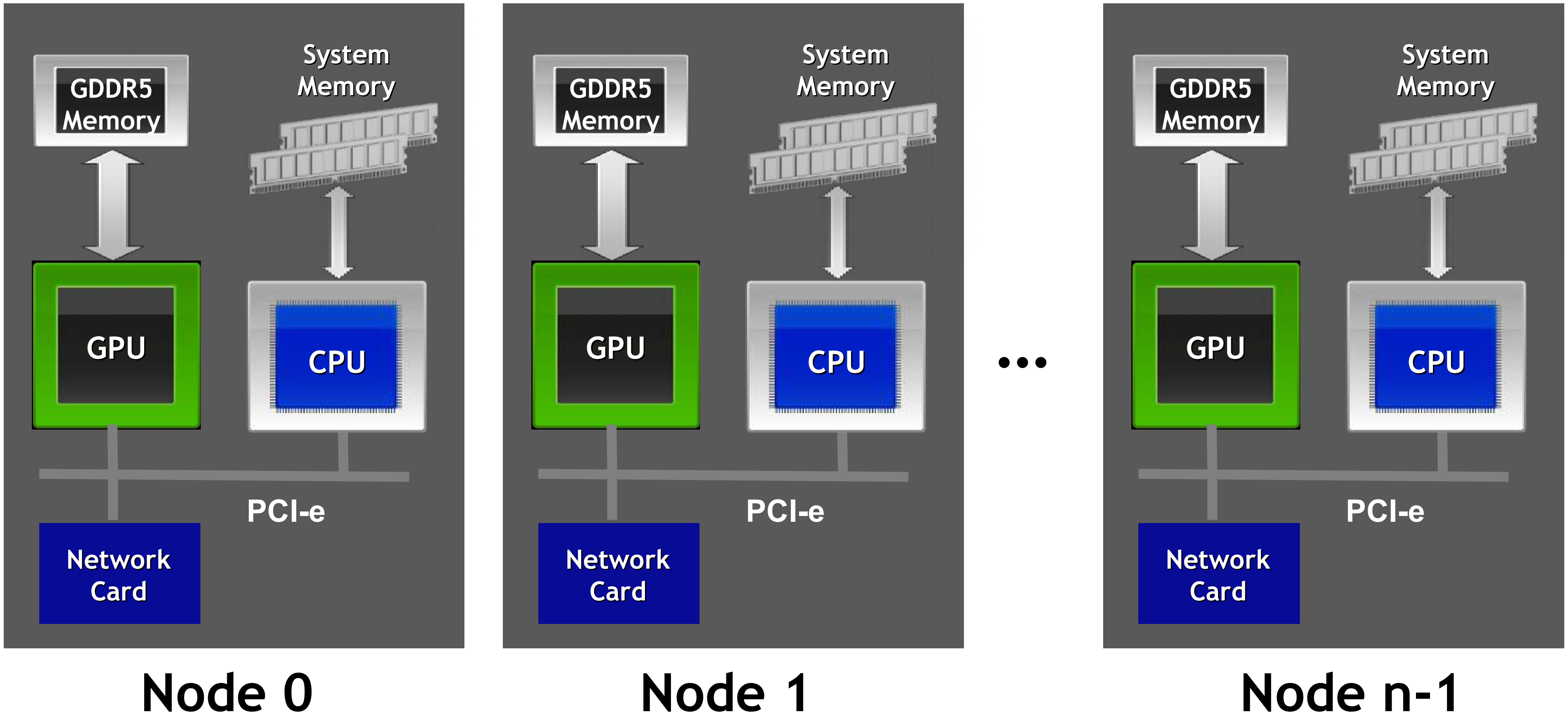
Introduction to CUDA-aware MPI

Presenter: Jan Meinke (JSC)

Jiri Kraus (NVIDIA)

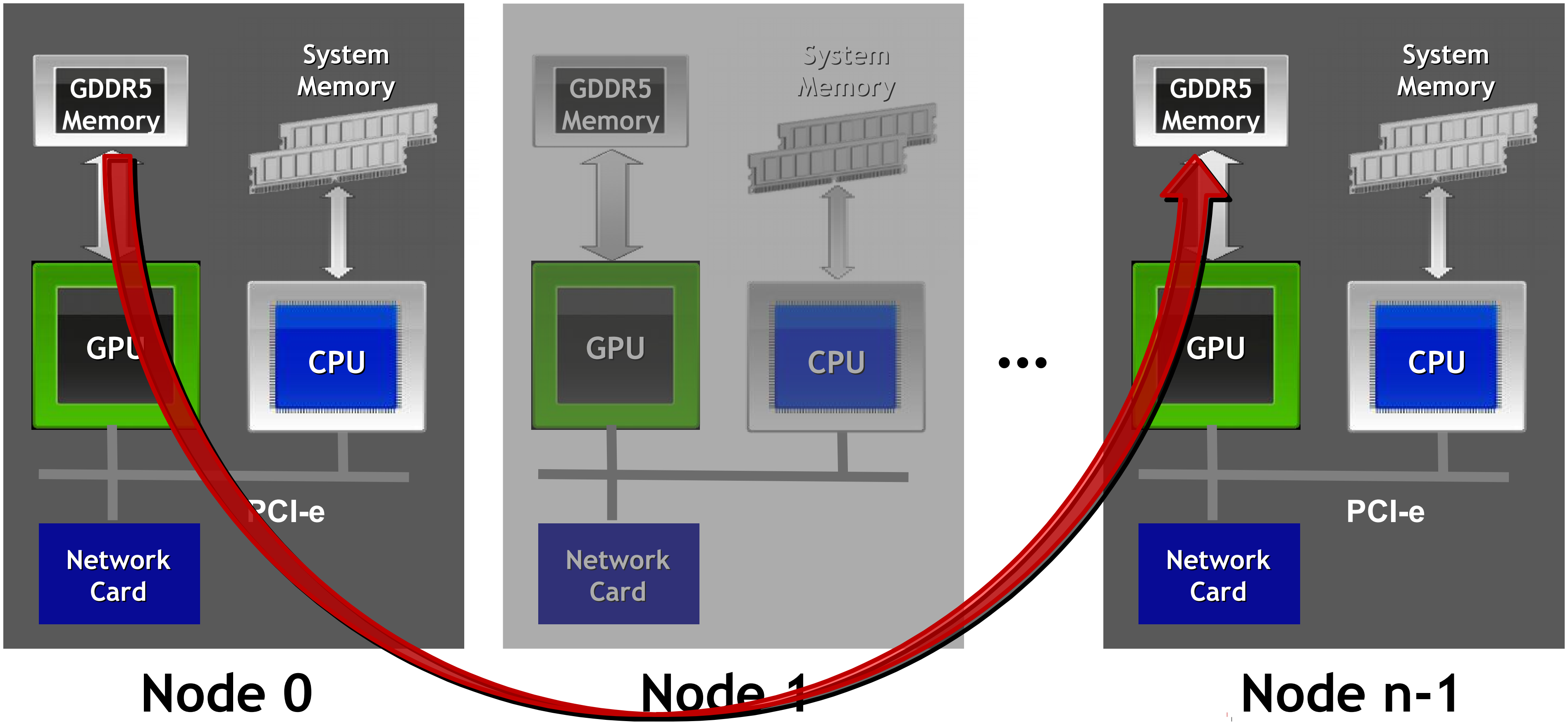
GPU Programming@Jülich Supercomputing Centre | Jülich | 25– 27 April 2016

MPI + CUDA



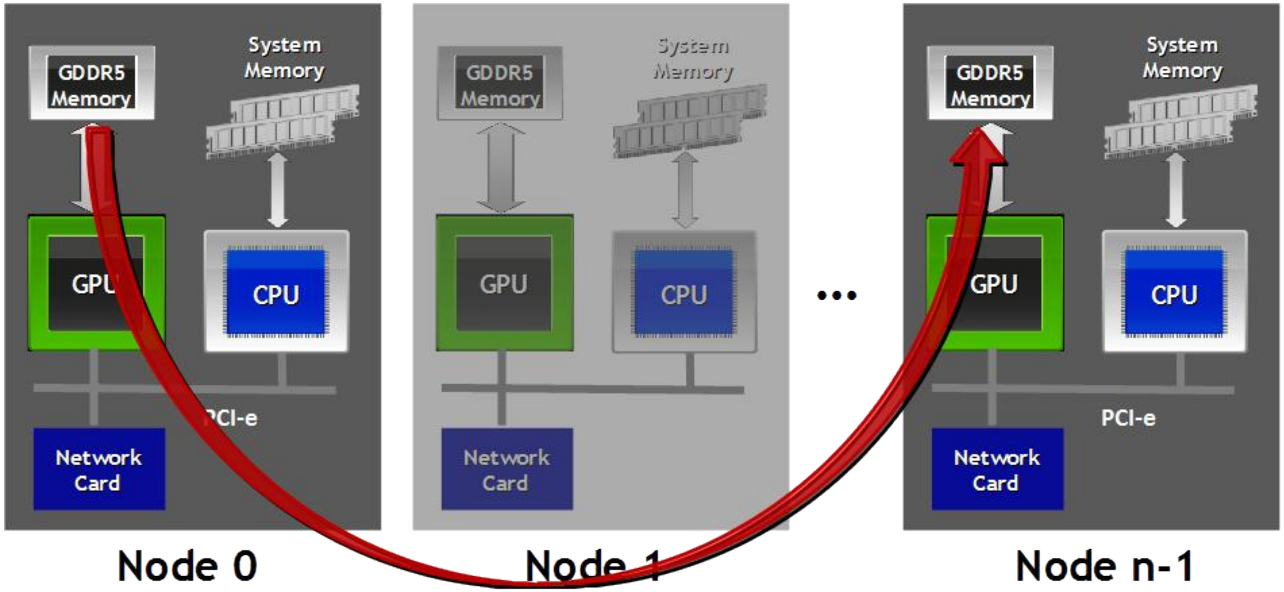
GDDR5 Memory
System Memory
GPU
CPU
PCI-e
Network Card
Node 0
GDDR5 Memory
System Memory
GPU
CPU
PCI-e
Network Card
Node 1
…
GDDR5 Memory
System Memory
GPU
CPU
PCI-e
Network Card
Node n-1

MPI + CUDA



GDDR5 Memory
System Memory
GPU
CPU
PCI-e
Network Card
GDDR5 Memory
System Memory
GPU
CPU
Network Card
GDDR5 Memory
System Memory
GPU
CPU
PCI-e
Network Card
Node 0
Node 1
Node n-1

MPI + CUDA




```
//MPI rank 0
MPI_Send(s_buf_d,size,MPI_CHAR,n-1,tag,MPI_COMM_WORLD);

//MPI rank n-1
MPI_Recv(r_buf_d,size,MPI_CHAR,0,tag,MPI_COMM_WORLD,&stat);
```

Outline

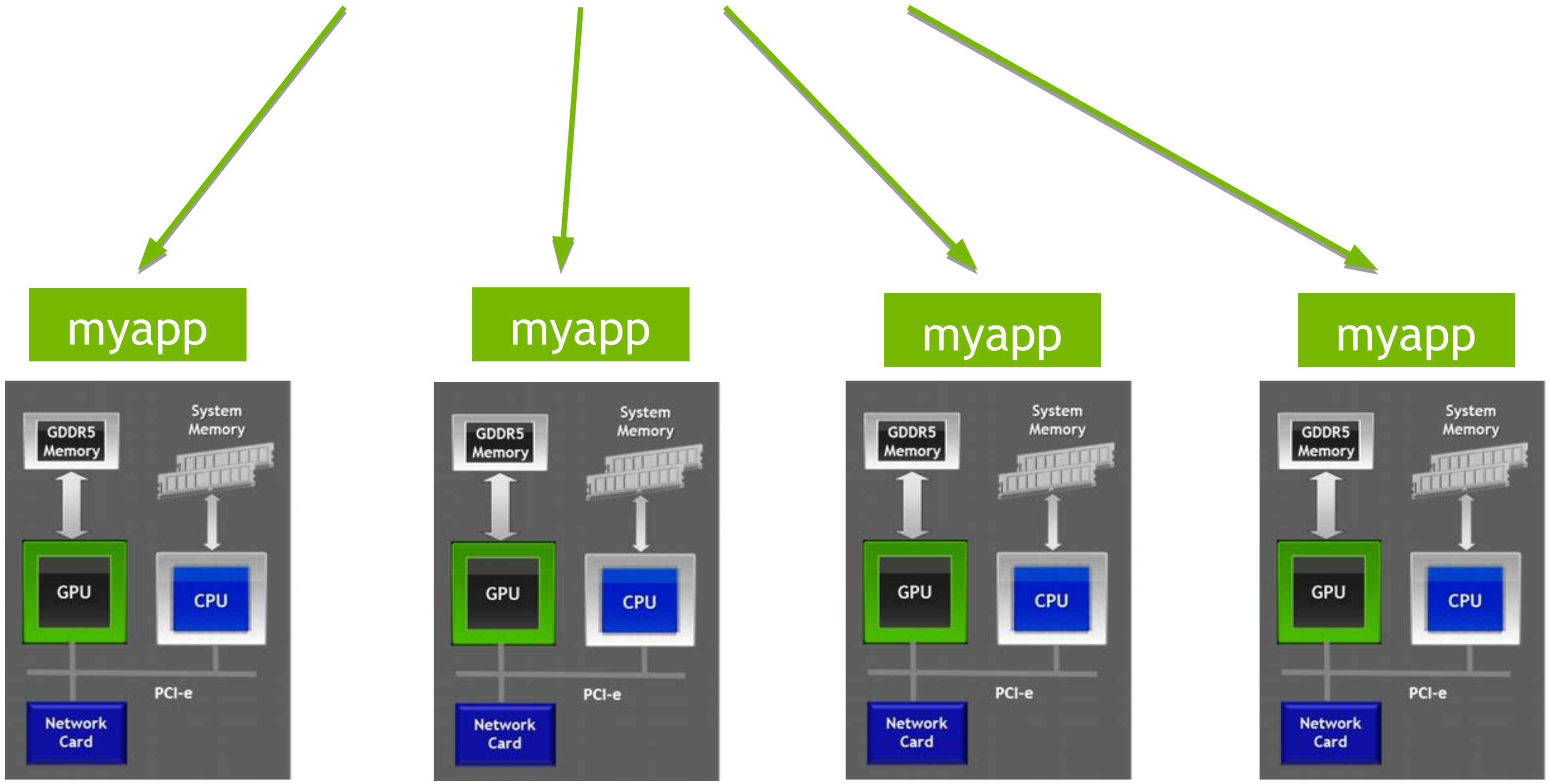
- Short Introduction to MPI
- Unified Virtual Addressing and GPUDirect
- How CUDA-aware MPI works
- Hands on

Message Passing Interface - MPI

- Standard to exchange data between processes via messages
    - Defines API to exchange messages
        - Pt. 2 Pt.: e.g. MPI_Send, MPI_Recv
        - Collectives, e.g. MPI_Reduce
- Multiple implementations (open source and commercial)
    - Binding for C/C++, Fortran, Python, ...

MPI – How to launch a MPI program

```
srun -n 4 ./myapp <args>
```

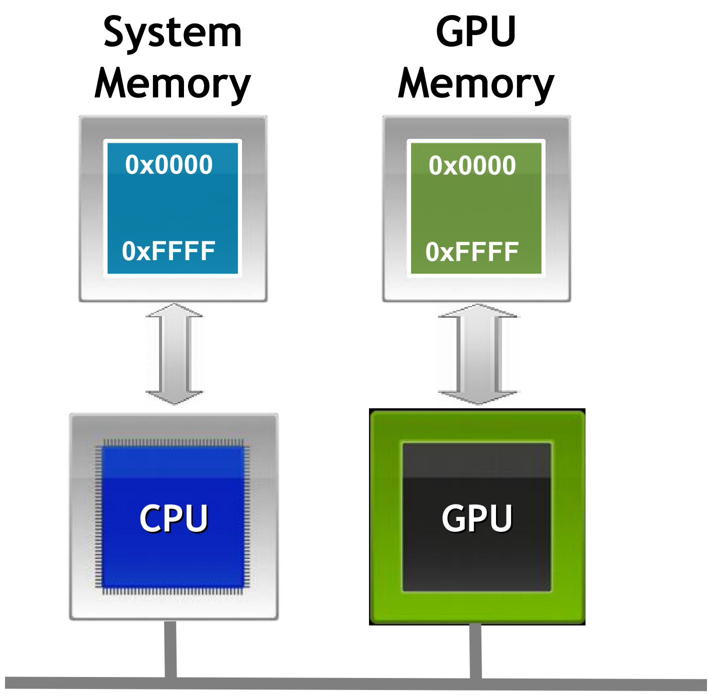


MPI – A minimal program

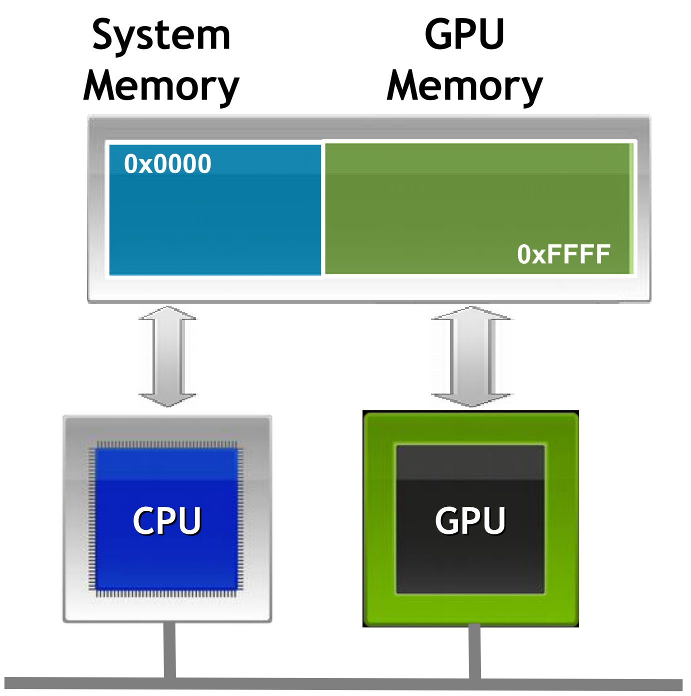
```
#include <mpi.h>
int main(int argc, char *argv[]) {
    int myrank;
    /* Initialize the MPI library */
    MPI_Init(&argc,&argv);
    /* Determine the calling process rank */
    MPI_Comm_rank(MPI_COMM_WORLD,&myrank);
    /* Call MPI routines like MPI_Send, MPI_Recv, ... */
    /* Shutdown MPI library */
    MPI_Finalize();
    return 0;
}
```

Unified Virtual Addressing

No UVA: Multiple Memory Spaces



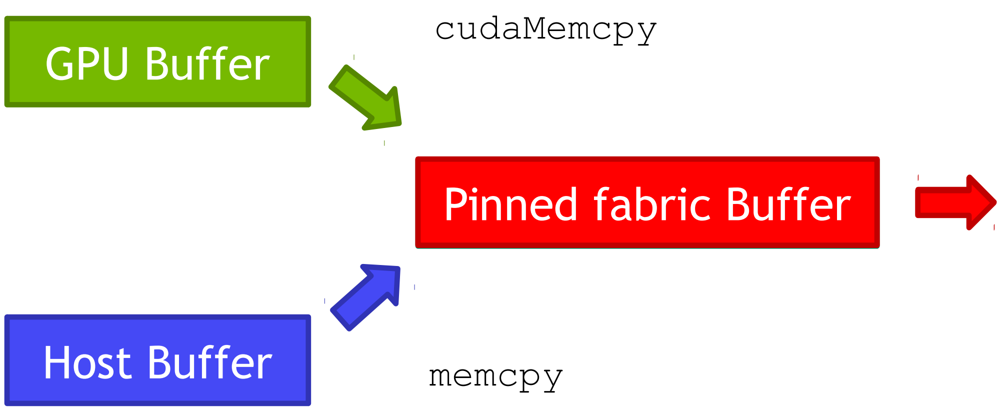
UVA : Single Address Space

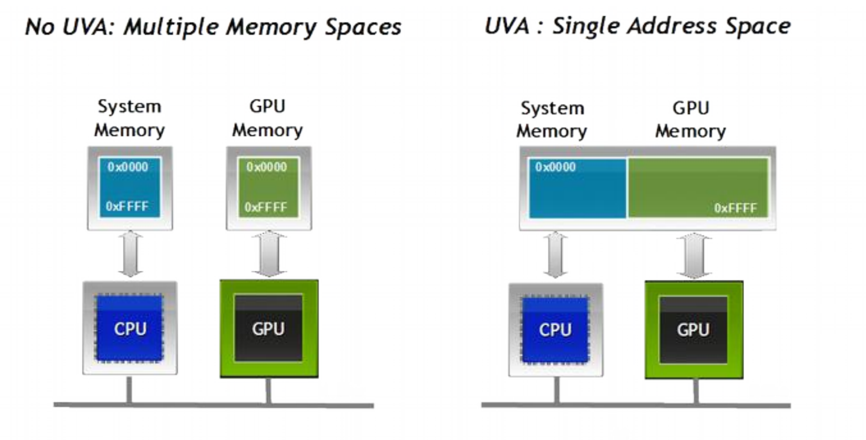


NVIDIA

JÜLICH FORSCHUNGSZENTRUM

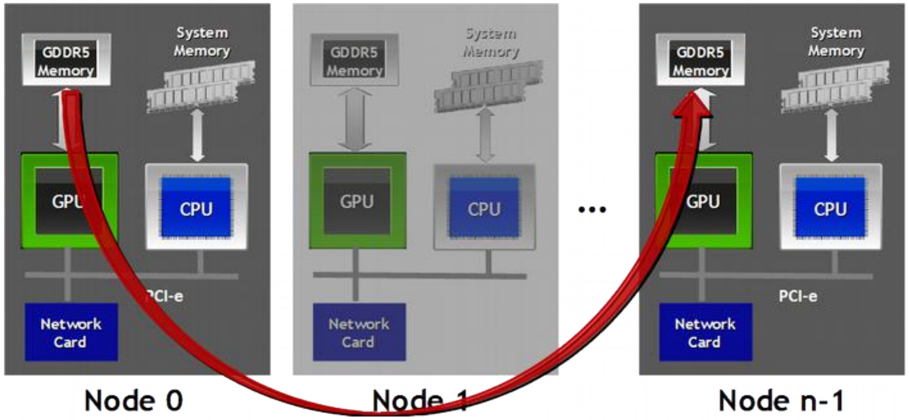
Unified Virtual Addressing





- One address space for all CPU and GPU memory
  - Determine physical memory location from a pointer value
  - Enable libraries to simplify their interfaces (e.g. MPI and cudaMemcpy)
- Supported on devices with compute capability 2.0 for
  - 64-bit applications on Linux and on Windows also TCC mode

MPI+CUDA



With UVA and CUDA-aware MPI

```
//MPI rank 0
MPI_Send(s_buf_d,size,…);

//MPI rank n-1
MPI_Recv(r_buf_d,size,…);
```

No UVA and regular MPI

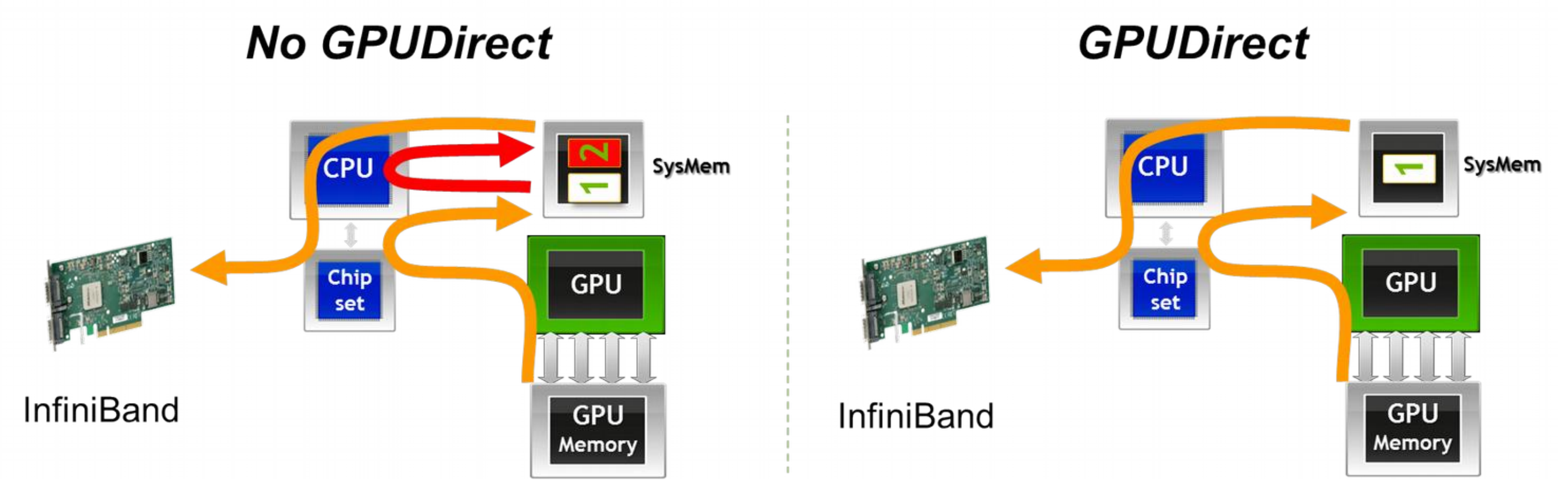
```
//MPI rank 0
cudaMemcpy(s_buf_h,s_buf_d,size,…);
MPI_Send(s_buf_h,size,…);

//MPI rank n-1
MPI_Recv(r_buf_h,size,…);
cudaMemcpy(r_buf_d,r_buf_h,size,…);
```

CUDA-aware MPI makes MPI+CUDA easier.

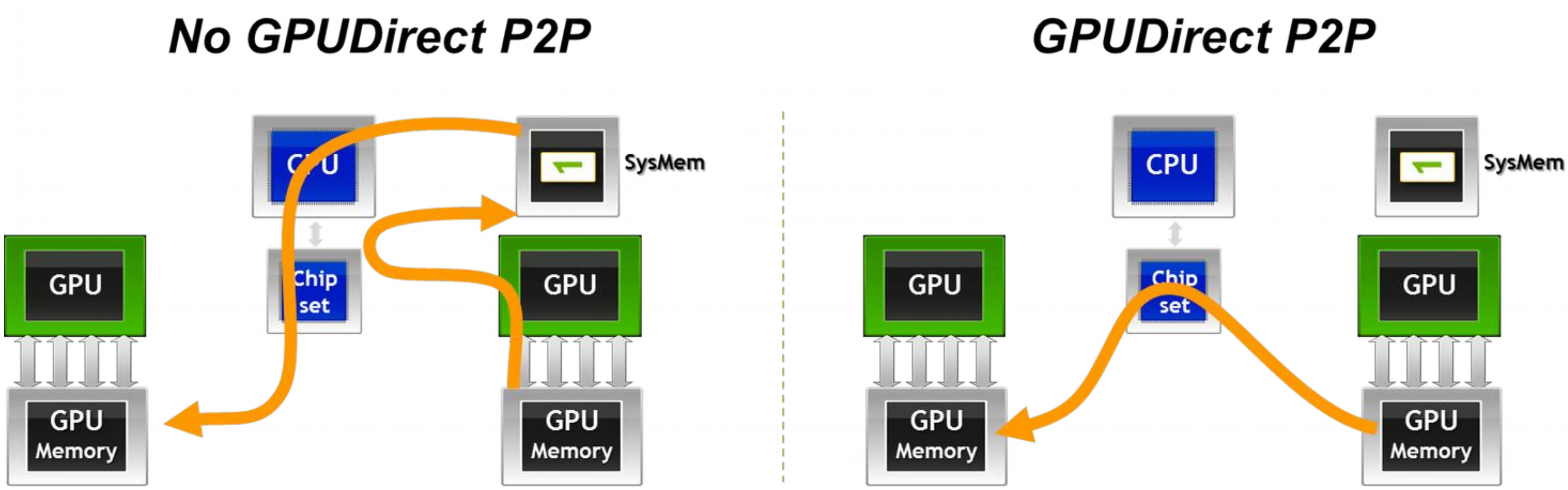
NVIDIA GPUDirect™

Accelerated communication with network & storage devices

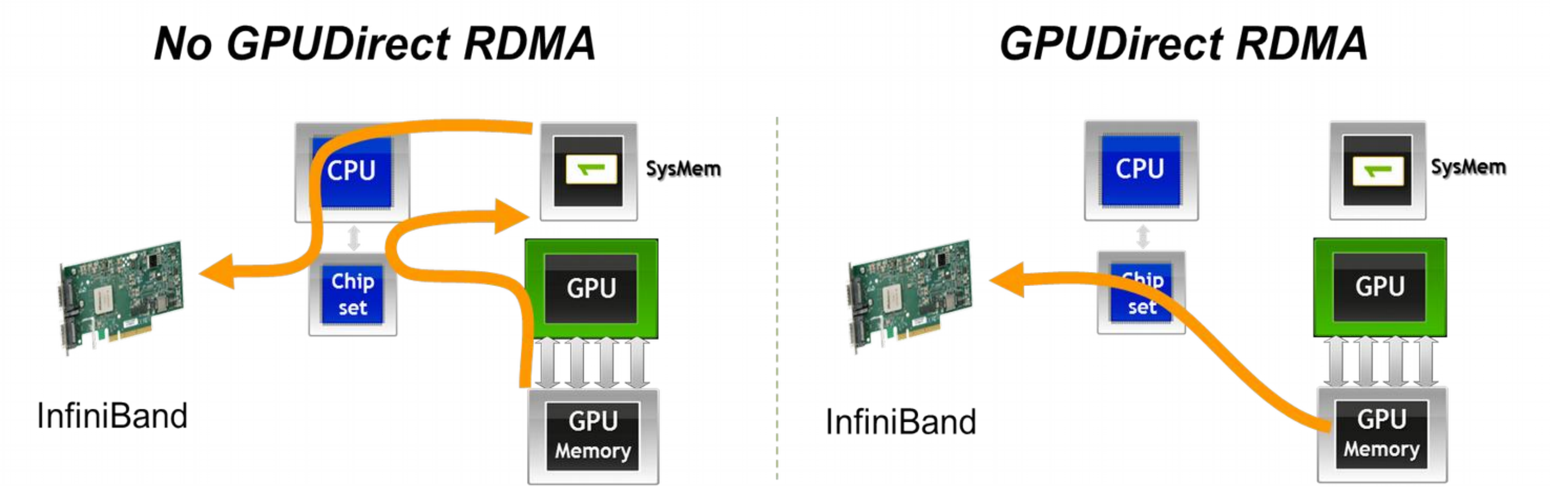


NVIDIA GPUDirect™

Peer to Peer Transfers



NVIDIA GPUDirect^{TM}
Support for RDMA



CUDA-Aware MPI

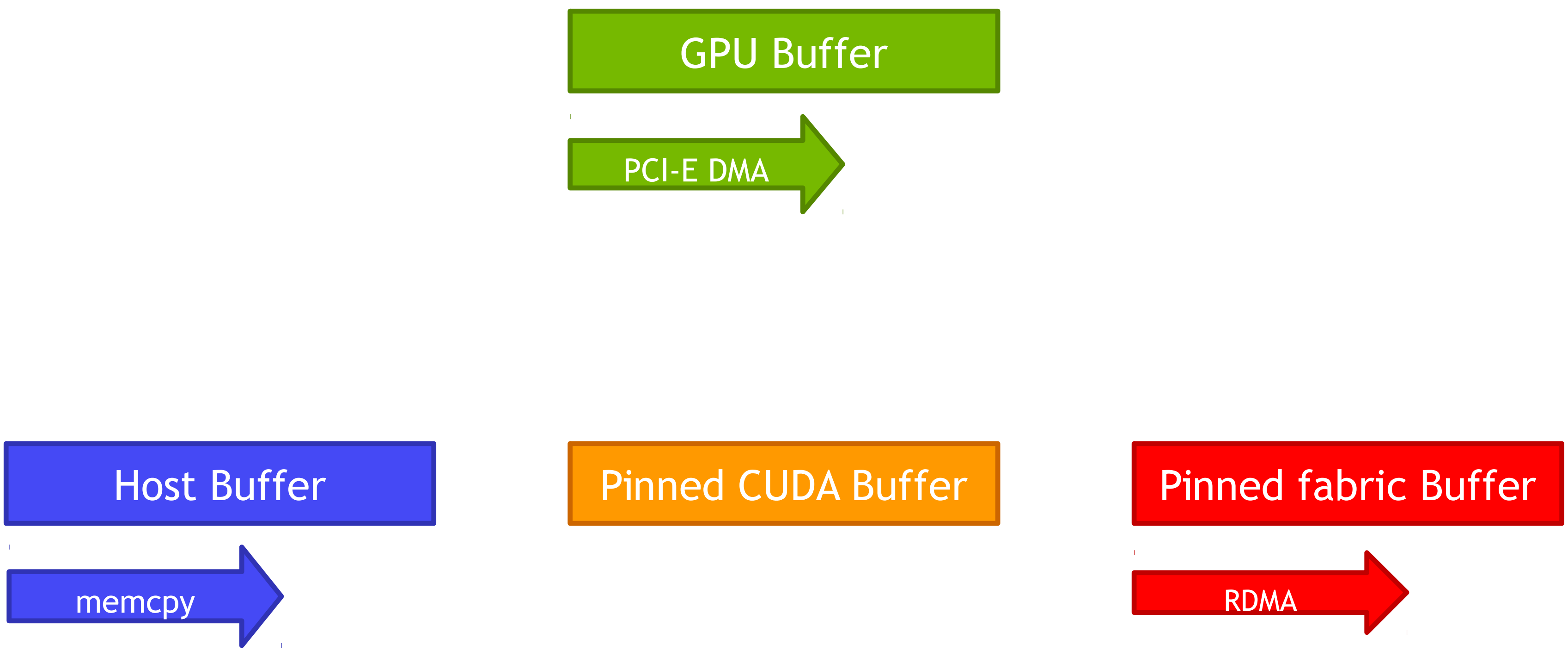
Example:

MPI Rank 0 MPI_Send from GPU Buffer

MPI Rank 1 MPI_Recv to GPU Buffer

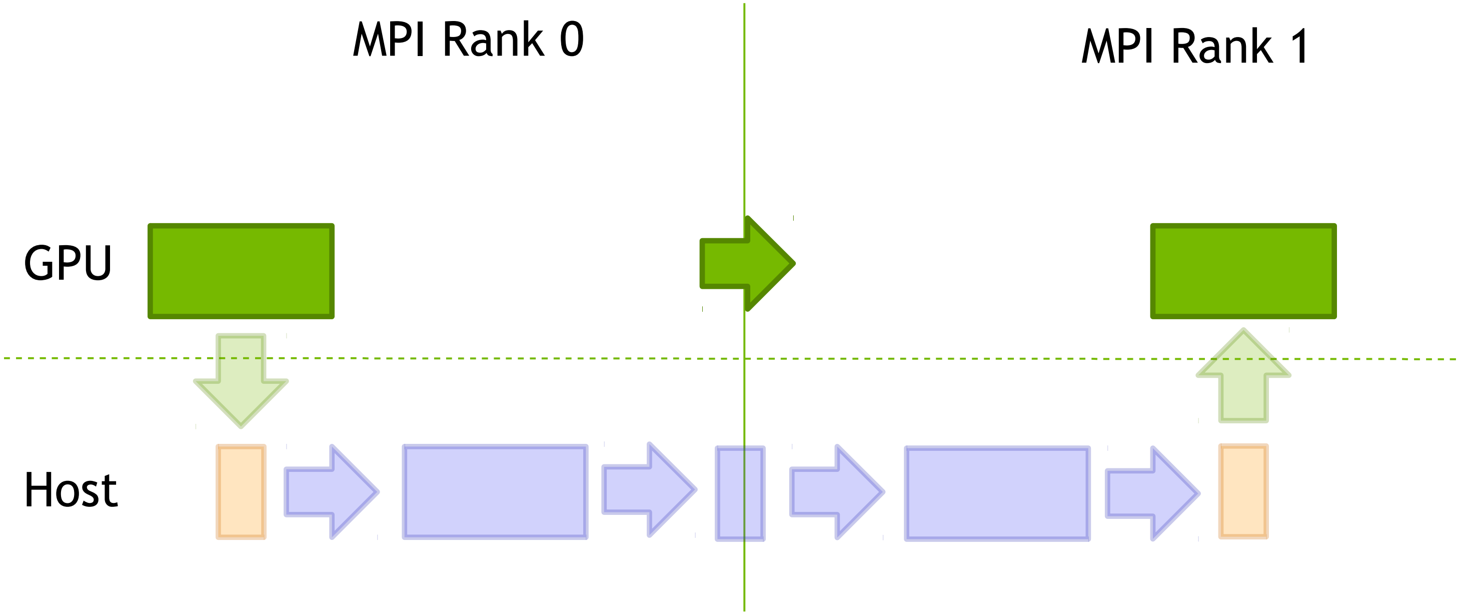
- Show how CUDA+MPI works in principle
  - Depending on the MPI implementation, message size, system setup, … situation might be different
- Two GPUs in one node

CUDA-Aware MPI



CUDA-Aware MPI GPU to local GPU

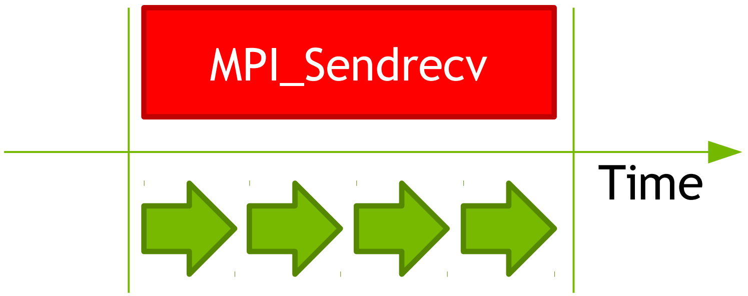
GPUDirect Peer to Peer Transfers



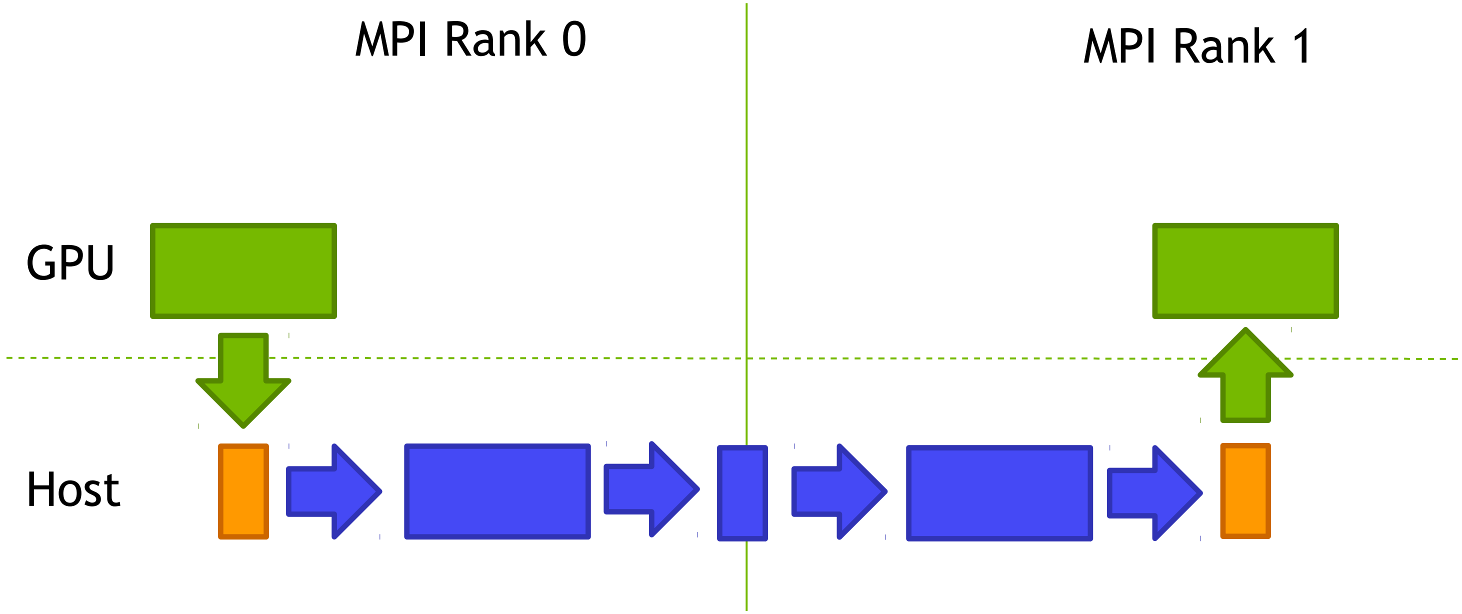

```
MPI_Send(s_buf_d,size,MPI_CHAR,1,tag,MPI_COMM_WORLD);

MPI_Recv(r_buf_d,size,MPI_CHAR,0,tag,MPI_COMM_WORLD,&stat);
```

CUDA-Aware MPI GPU to local GPU
GPUDirect Peer to Peer Transfers



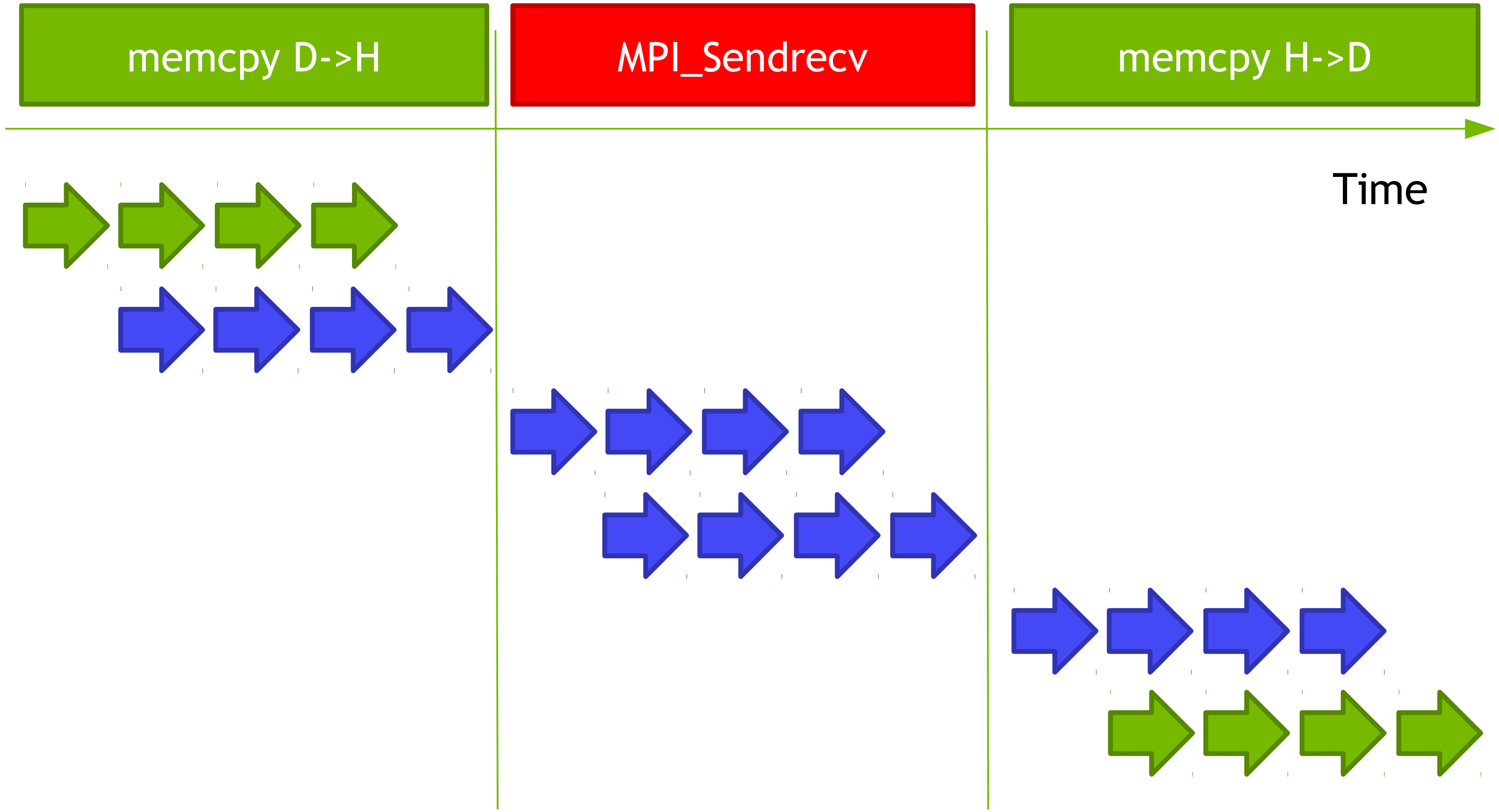
Regular MPI GPU to local GPU



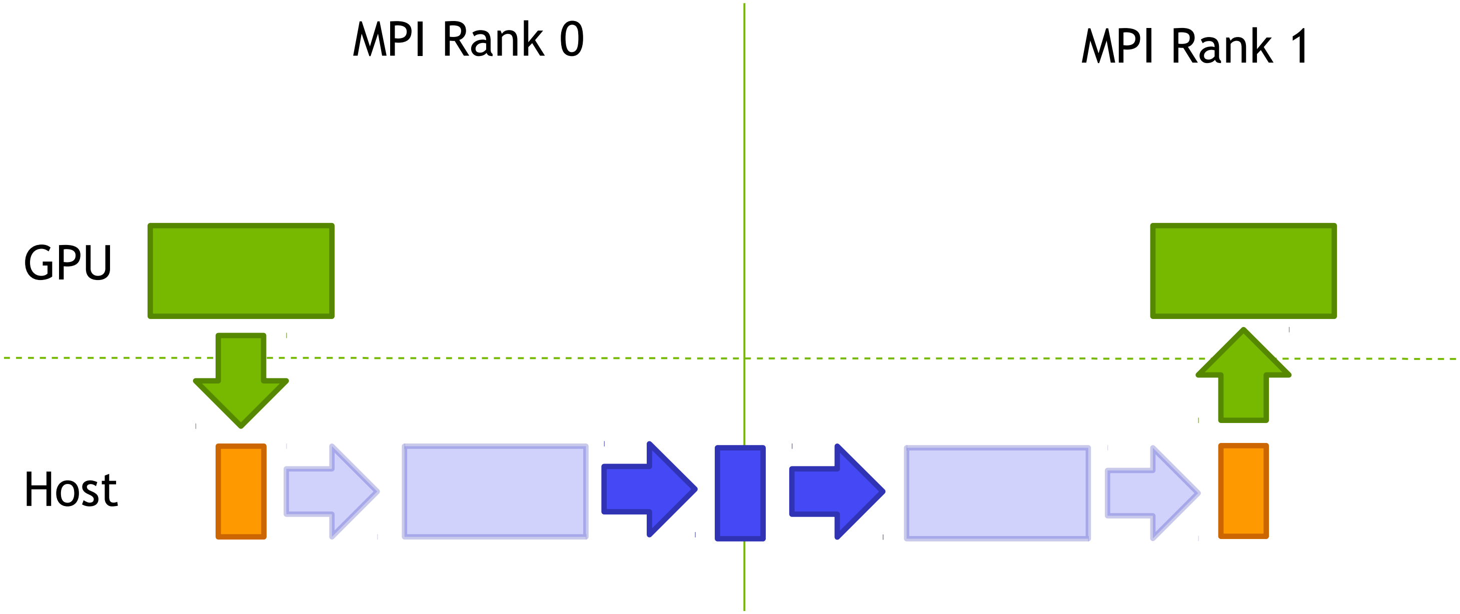

```
cudaMemcpy(s_buf_h,s_buf_d,size,cudaMemcpyDeviceToHost);
MPI_Send(s_buf_h,size,MPI_CHAR,1,tag,MPI_COMM_WORLD);

MPI_Recv(r_buf_h,size,MPI_CHAR,0,tag,MPI_COMM_WORLD,&stat);
cudaMemcpy(r_buf_d,r_buf_h,size,cudaMemcpyHostToDevice);
```

Regular MPI GPU to local GPU



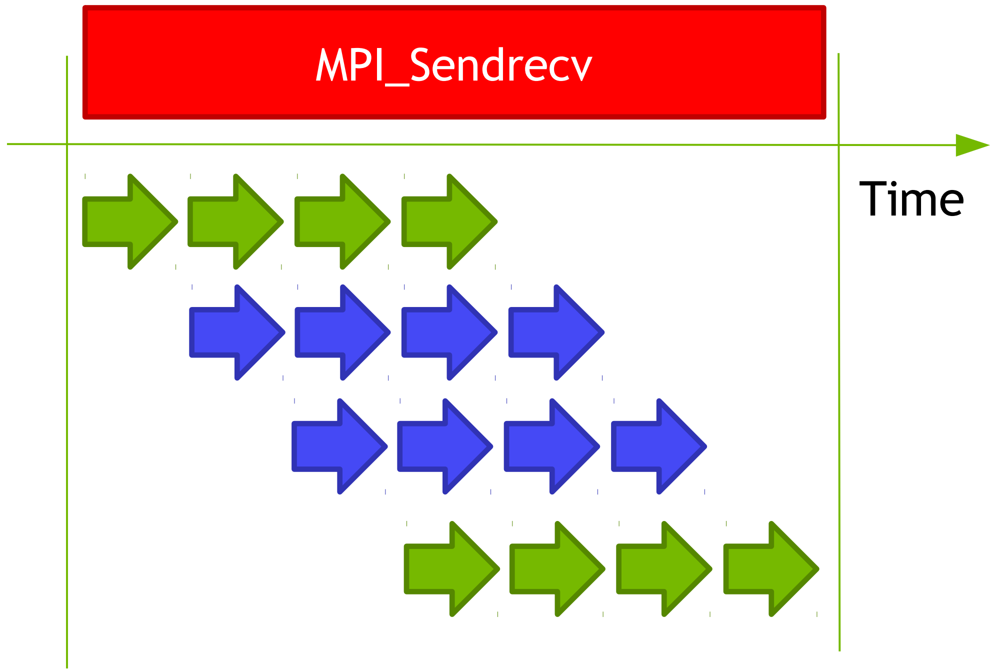
CUDA-aware MPI GPU to local GPU
without GPUDirect



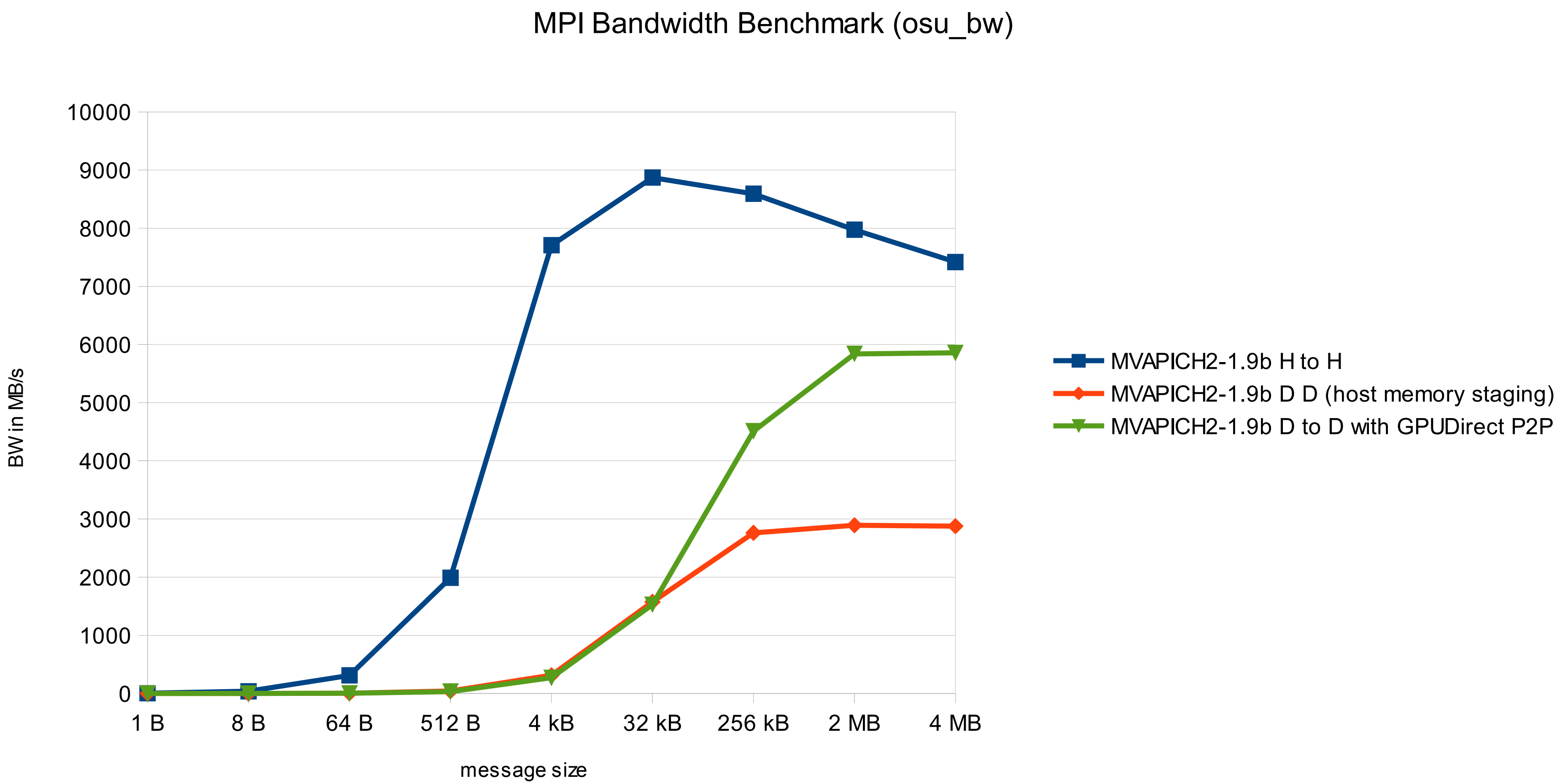

```
MPI_Send(s_buf_d,size,MPI_CHAR,1,tag,MPI_COMM_WORLD);

MPI_Recv(r_buf_d,size,MPI_CHAR,0,tag,MPI_COMM_WORLD,&stat);
```

CUDA-aware MPI GPU to local GPU
without GPUDirect



Performance Results intra Node



Latency (1 byte) 23.68 µs 17.59 µs 0.26 µs

CUDA-Aware MPI Implementations
Integrated Support for GPU Computing

- MVAPICH2 1.8/1.9/2.0

  http://mvapich.cse.ohio-state.edu/overview/mvapich2/
- OpenMPI 1.7(beta)/1.8

  http://www.open-mpi.org
- CRAY MPI (MPT 5.6.2)
- IBM Platform MPI (8.3)

CUDA-Aware Caveats

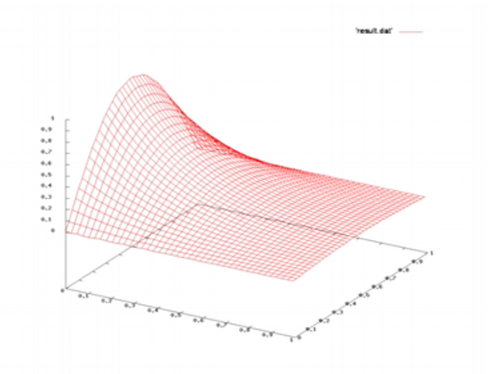
- cudaSetDevice needs to be called before MPI_Init
  - No longer necessary for latest releases of MVAPICH2 and OpenMPI
- MPI Environment vars. can be used to set GPU affinity
  - **Parastation: MPI_LOCALRANKID** (JURECA)
  - MVAPICH2: MV2_COMM_WORLD_LOCAL_RANK
  - OpenMPI: OMPI_COMM_WORLD_LOCAL_RANK
- **MV2_USE_CUDA** needs to be set for MVAPICH (done by module)
- MPICH_RDMA_ENABLED_CUDA for MPT on Cray
- PMPI_GPU_AWARE for Platform MPI
- Lib needs to be build with CUDA-awarenes enabled

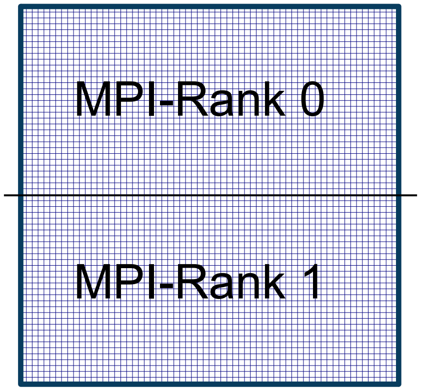
Conclusions

- Use CUDA-aware MPI when possible
- Depending on CUDA version, hardware setup, … a CUDA-aware MPI gives you
  - Ease of programming
  - Pipelined data transfer which automatically provides optimizations when available
    - Overlap CUDA copy and RDMA transfer
  - Utilization of the best GPUDirect technology available

Hands-on Example: Jacobi

- Solves the 2D-Poission equation on a square
  - Dirichlet boundary conditions
- 1D domain decompostion with two domains





Hands-on Example: Jacobi

While not converged

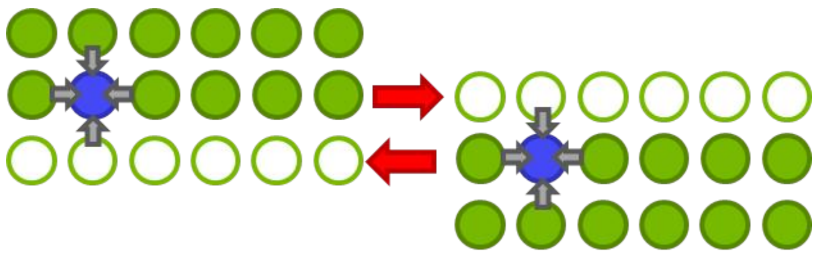
do Jacobi step:

```
for (int i=1; i < n-1; i++) for (int j=1; j < m-1; j++)
    u_new[i][j] = 0.0f - 0.25f*(u[i-1][j] + u[i+1][j]
                                + u[i][j-1] + u[i][j+1])
```

exchange halo

copy `u_new` and `u`

next iteration



Task: Modify the provided MPI+CUDA Jacobi to utilize CUDA-aware MPI

- Follow TODOs in
  - `CUDA-aware_MPI/exercises/tasks/jacobi_cuda.c`
  - Initialize CUDA before MPI_Init (call cudaSetDevice)
  - Pass device pointers directly to MPI
- Solution in

  `CUDA-aware_MPI/exercises/solutions/jacobi_cuda.c`
- Slides are in

  `CUDA-aware_MPI/slides/CUDA-aware_MPI.pdf`

Cheat Sheet

- Build Environment

```
source setup.sh
```

- Execution Instructions

```
srun -n 2 ./jacobi_mpi+cuda
```

- Slides

```
CUDA-aware_MPI/slides/CUDA-aware_MPI.pdf
```

- Introduction to CUDA-aware MPI

```
http://developer.nvidia.com/content/introduction-cuda-aware-mpi
```