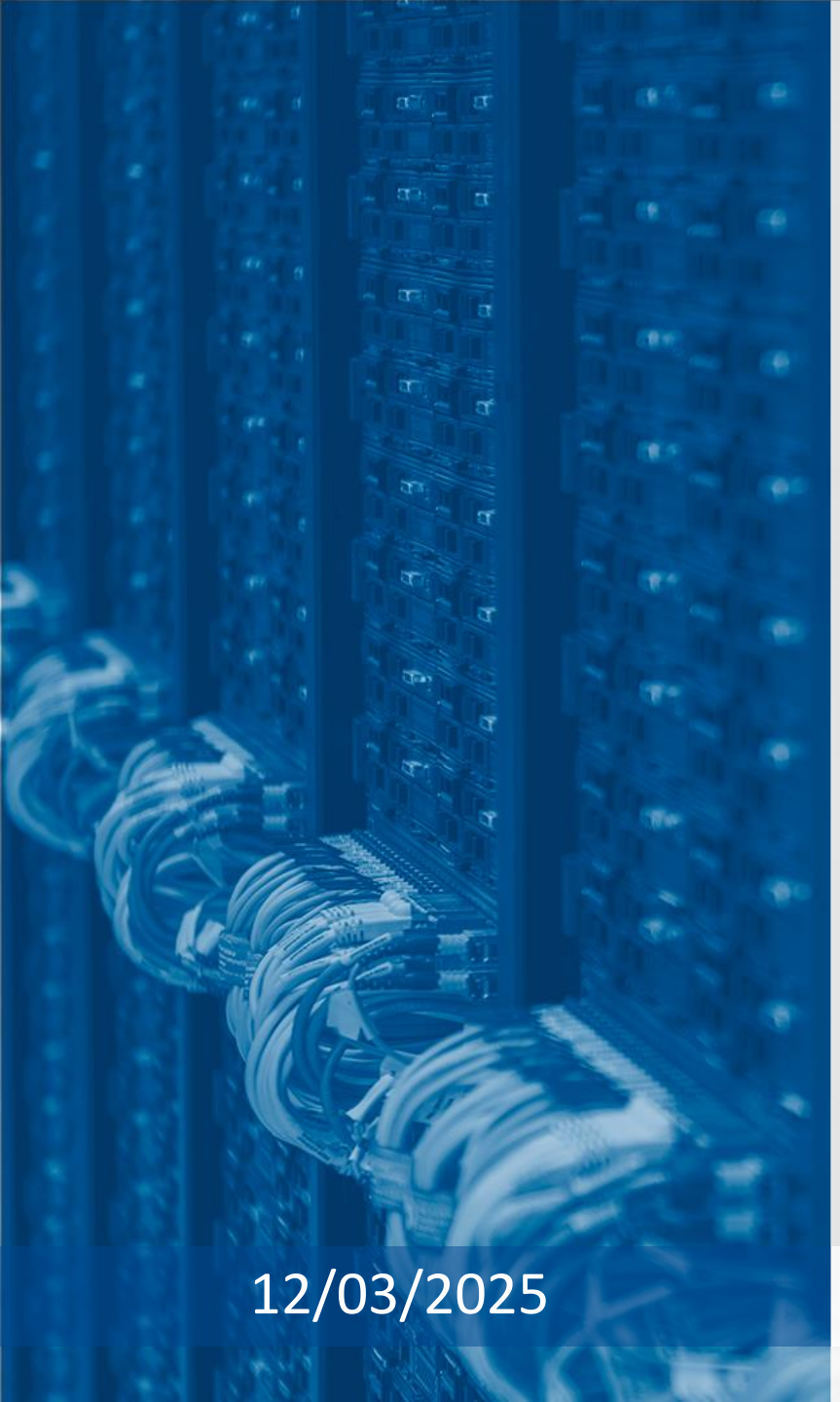


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Leveraging SmartNICs in HPC

Sergio Iserte and Antonio J. Peña

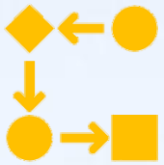
12/03/2025

Academic Salon on High-Performance Ethernet: Host Networking and Monitoring - Munich (Germany)

AccelCom
Accelerators & Communications for HPC

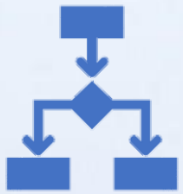
SmartNICs in HPC

Overlapping computational stages



Pre/post-processing

Asynchronous or non-dependent tasks



Computation/ Communication

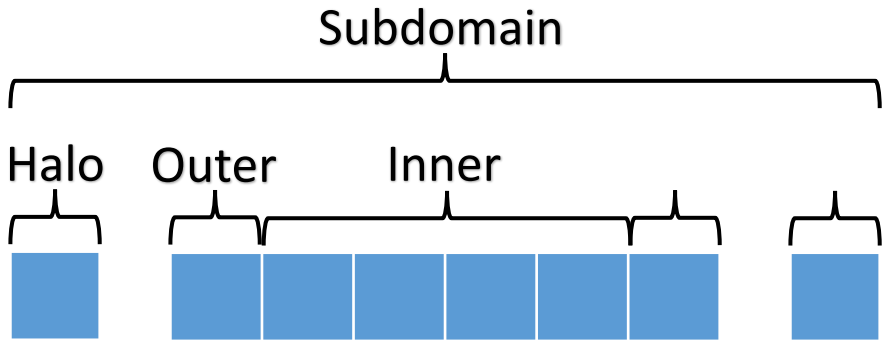
Adaptive mesh refinement

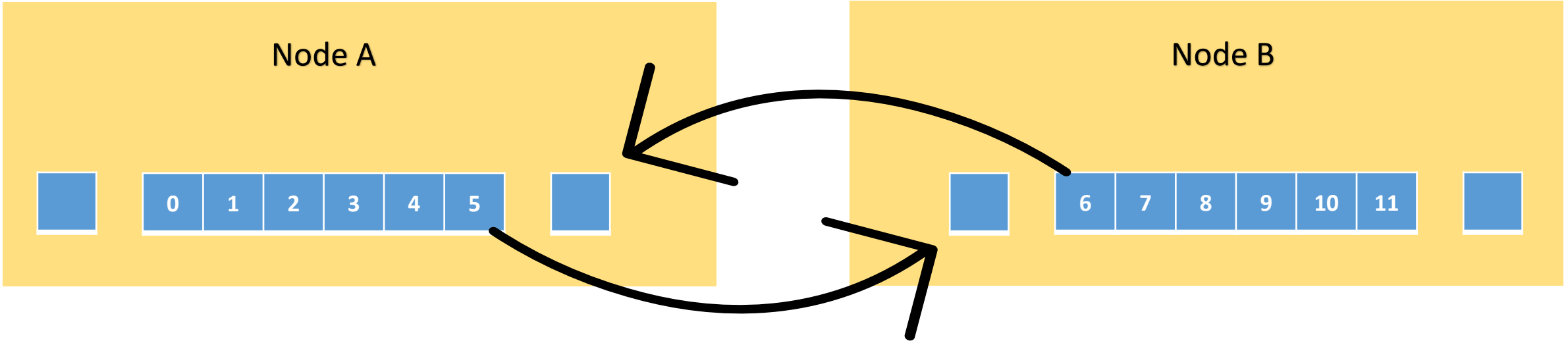
Neighbor building

Halo exchange

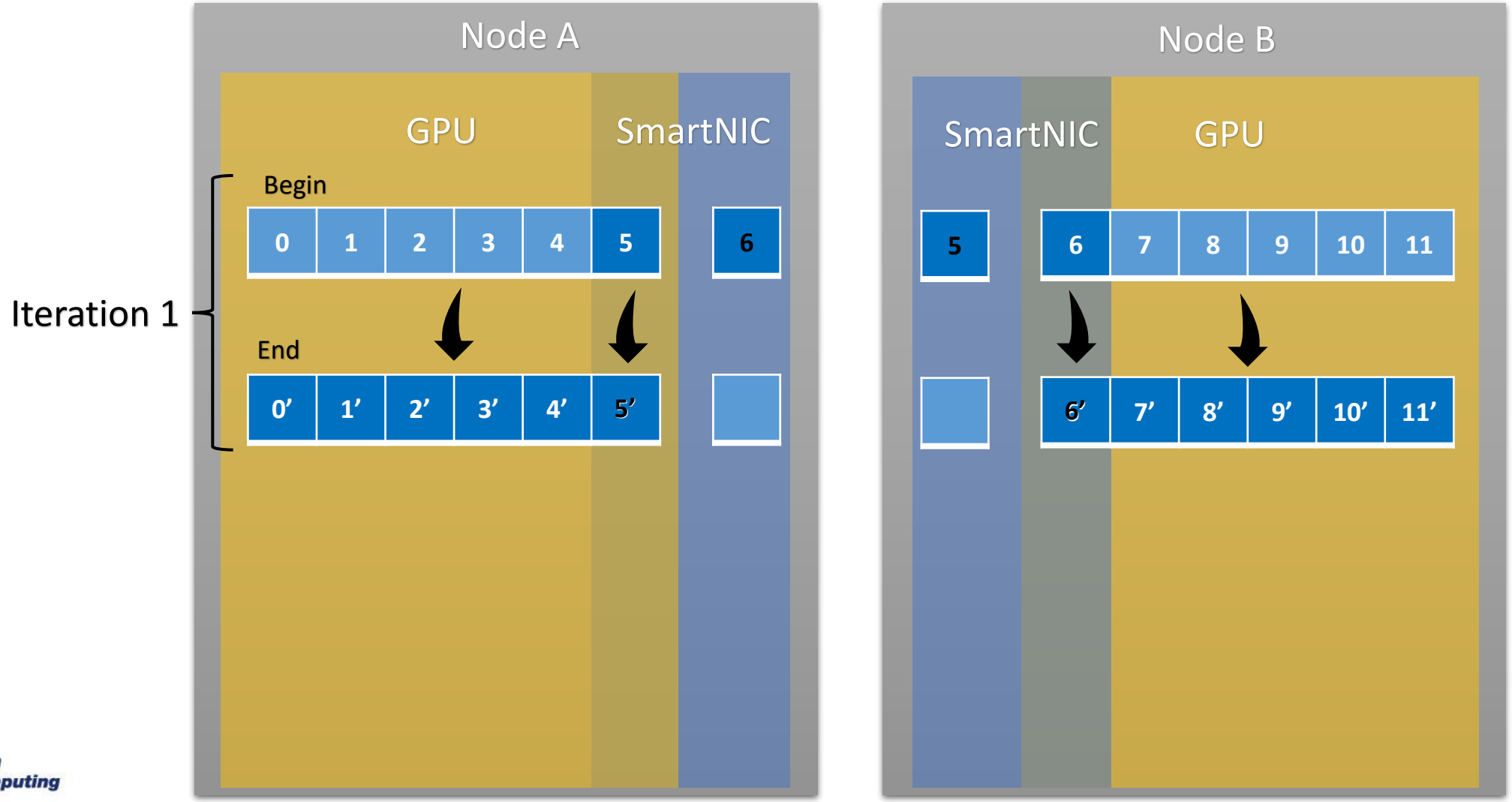
Halo Exchange Offloading

SIZE = 8
THICKNESS = 1

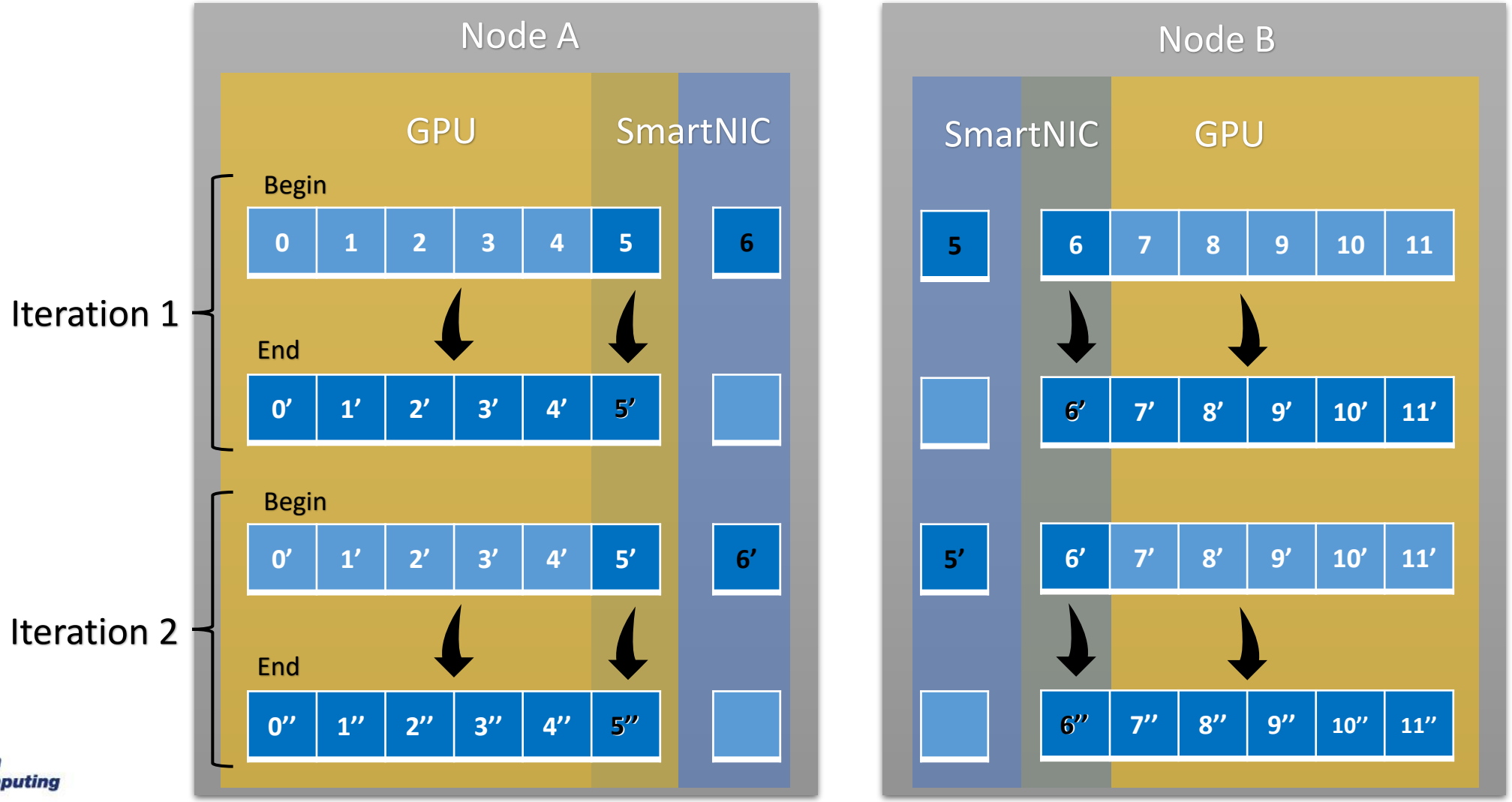




Halo Exchange Offloading



Halo Exchange Offloading



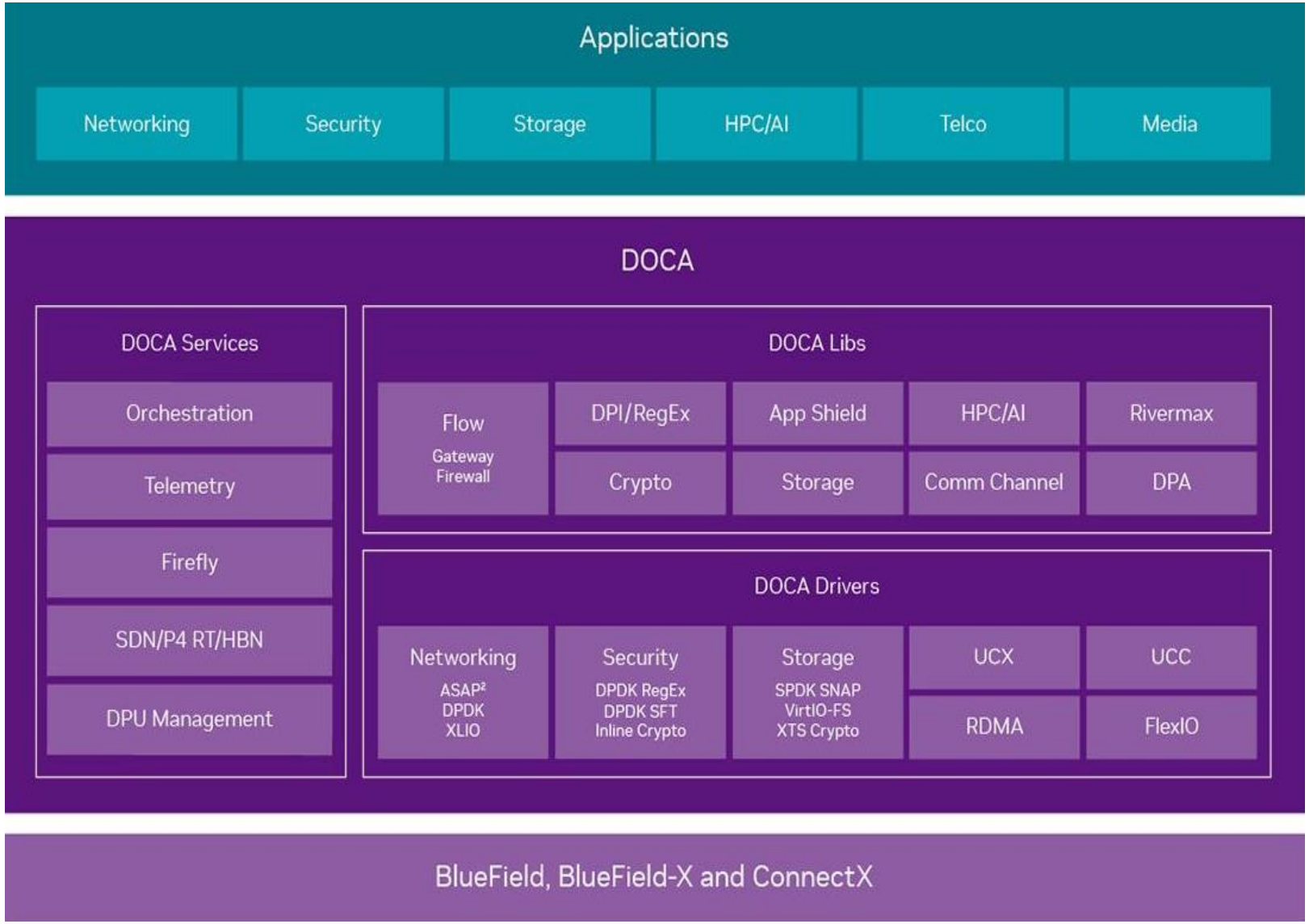
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

How to Code it?

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

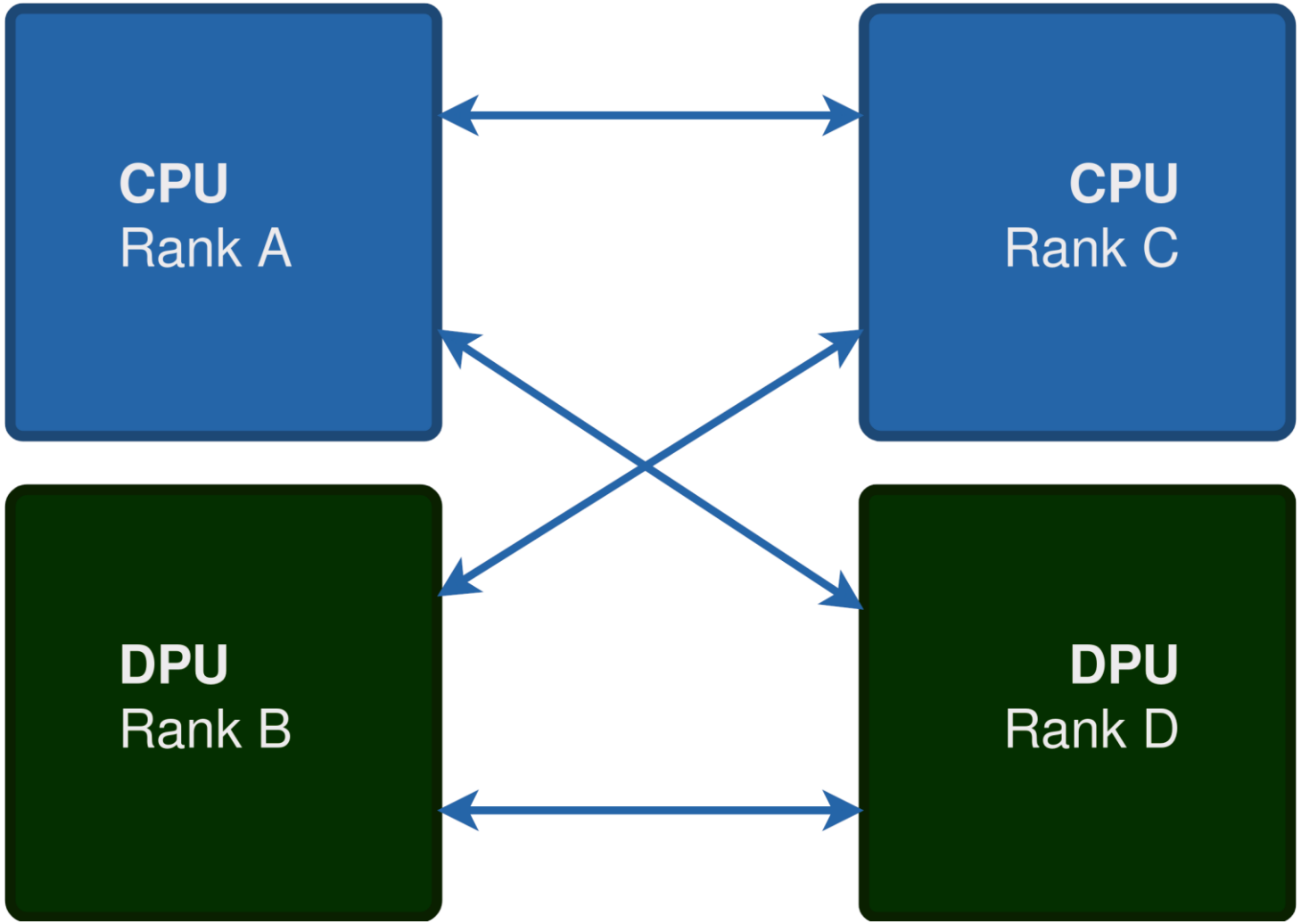
Low Level API

- For instance, in Nvidia BlueField DPUs, the DOCA SDK.
- DOCA is to DPUs what CUDA is to GPUs.
- Too low level for domain scientists.



MPMD Execution Model

- Multiple Program Multiple Data, for instance, in MPI.
- Poor productivity because the processors are widely different.

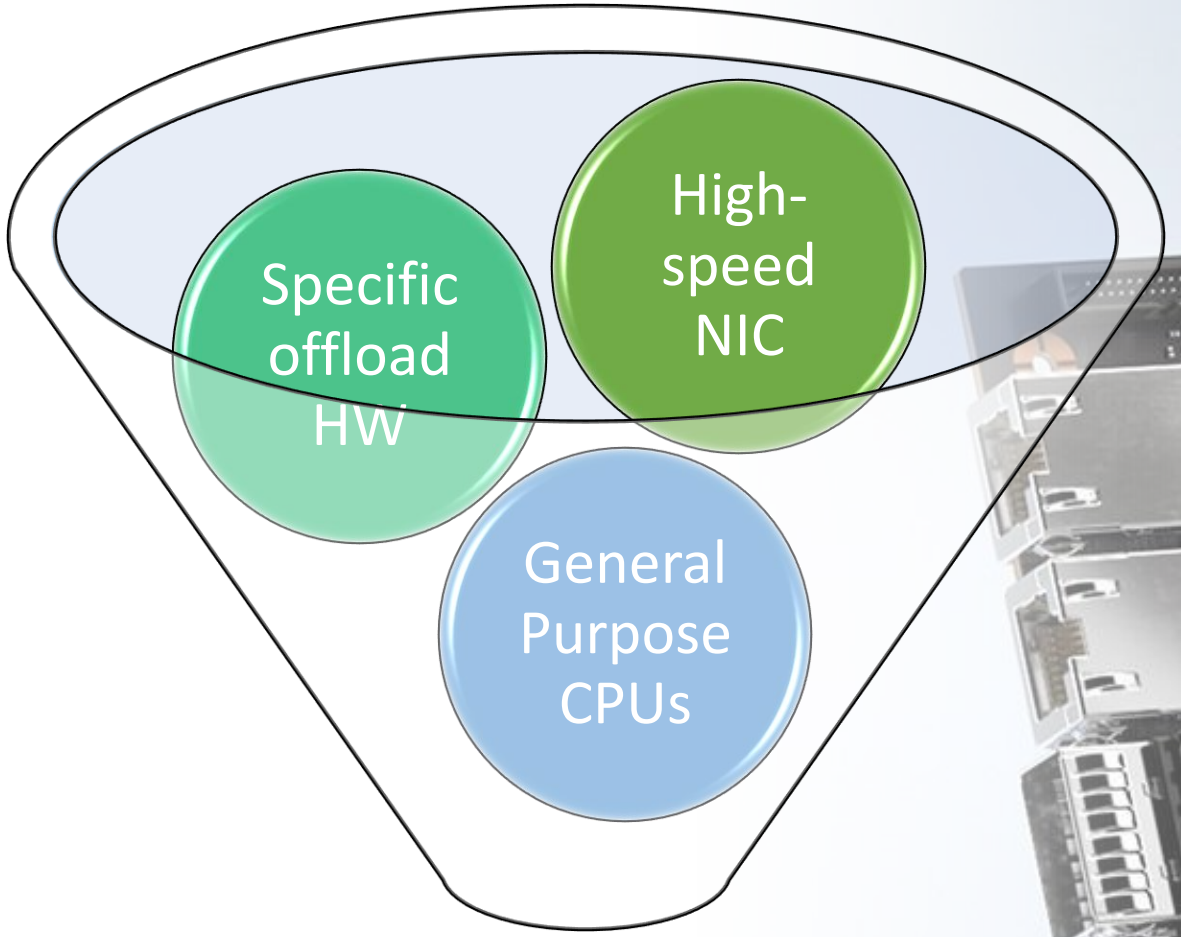


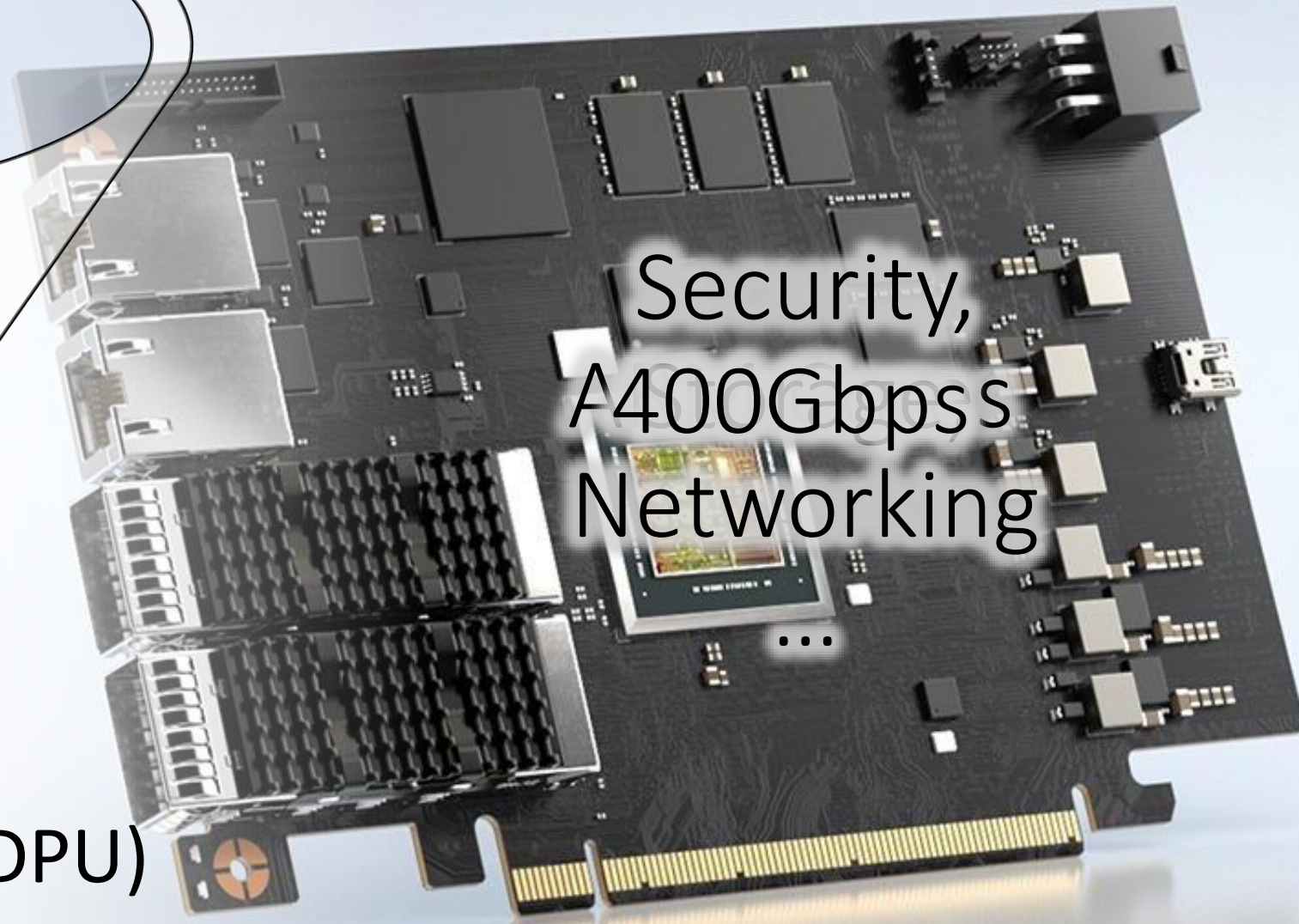
```
void main( int argc, char* argv[] ) {
    // Data initialization
    MPI_Init( &argc, &argv );
    MPI_Comm_rank( MPI_COMM_WORLD, &dpurank );
    if ( dpurank )  // DPU rank receives data
        MPI_Recv( &buf, SIZE, MPI_INT, 0, … );
    else            // Host rank sends data
        MPI_Send( &buf, SIZE, MPI_INT, dpurank, … );
    for( int t = 1; t <= TIMESTEPS; t++ ) {
        if ( dpurank ); // Offloaded compute
        else;           // Host compute
    }
    if ( dpurank )      // DPU rank send data
        MPI_Send( &buf, SIZE, MPI_INT, 0, … );
    else            // Host rank receive data
        MPI_Recv( &buf, SIZE, MPI_INT, dpurank, … );
    MPI_Finalize();
}
```

```
void main( int argc, char* argv[] ) {
    // Data initialization
    #pragma omp target
                data map(tofrom:buf[0:SIZE]))
    {
        for( int t = 1; t <= TIMESTEPS; t++ ) {
            #pragma omp target
            {
                // Offloaded compute
            }
                // Host compute
        }
    }
}
```

OpenMP DPU Offloading Support

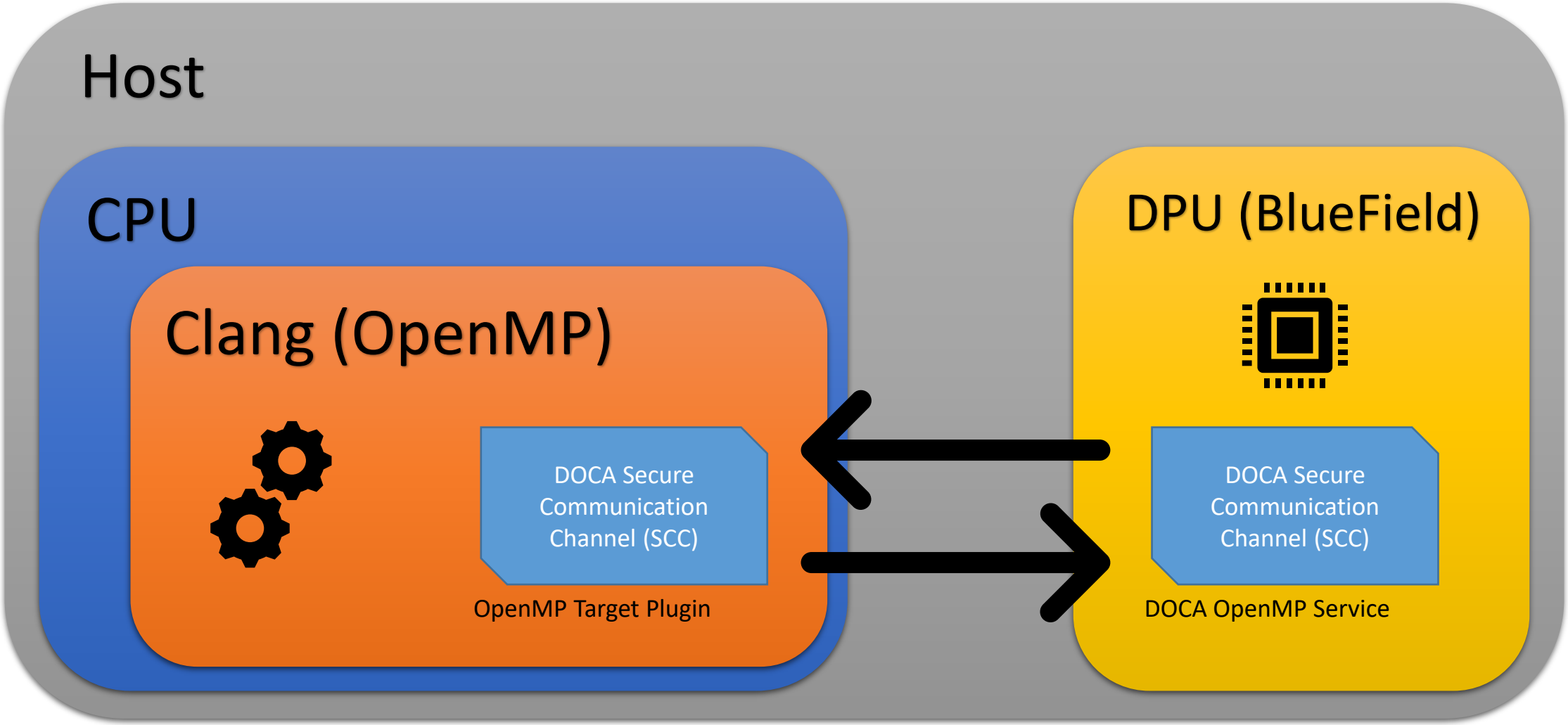
NVIDIA BlueField





Data Processing Unit (DPU)

ODOS Architecture

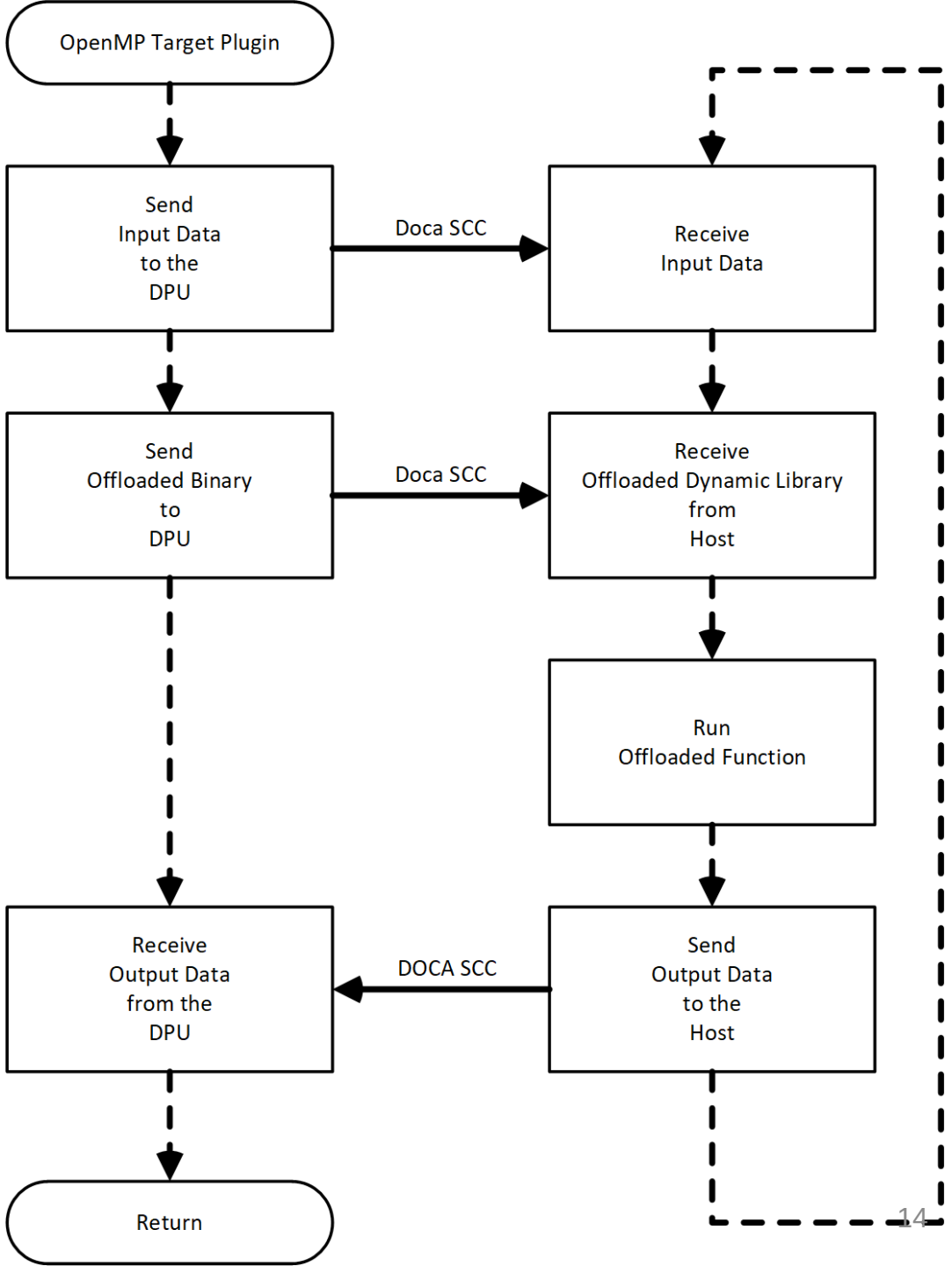


M. Usman, S. Iserte, R. Ferrer, and A. Peña, "DPU Offloading Programming with the OpenMP API" *LLVM-HPC23 held in conjunction with SC*, Nov. 2023. https://doi.org/10.1145/3624062.3624165

ODOS Flow

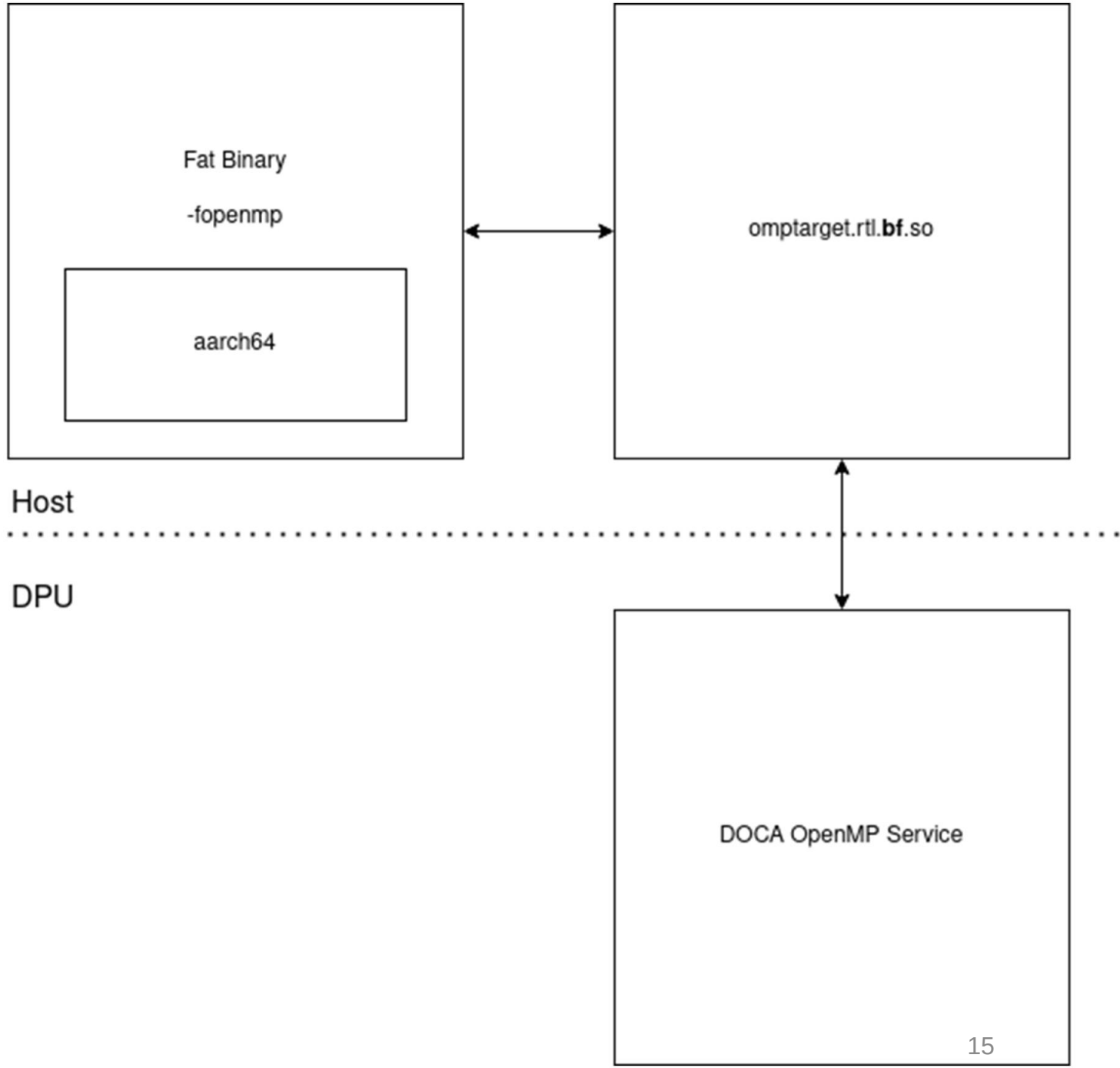
Using DPUs as an accelerator instead of a co-host.

DPU usage becomes very friendly to the HPC user.



ODOS Modules

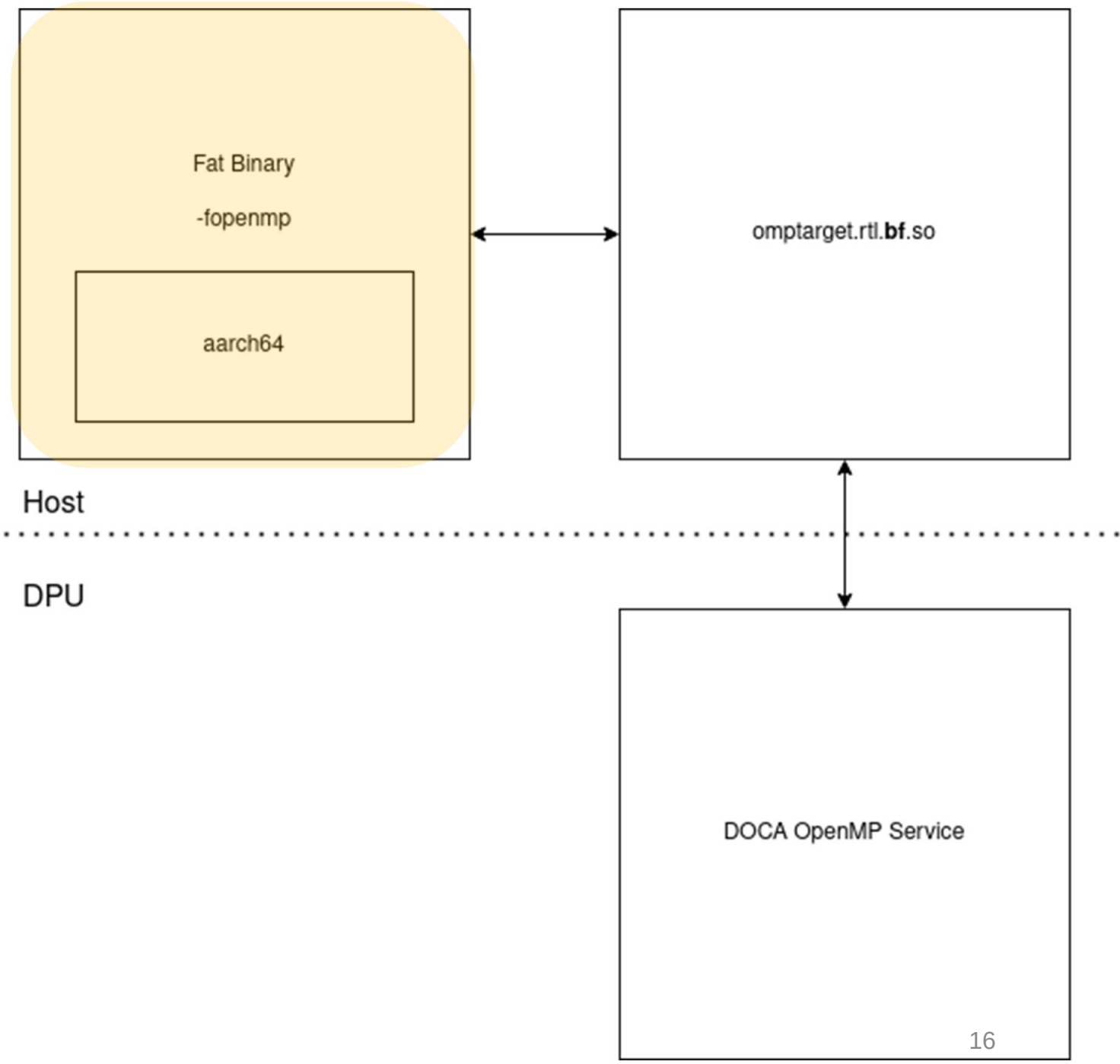
- Fat Binary Generation
- OpenMP Runtime (host)
- OpenMP Service (device)



Fat Binary Generation

Cross-compilation

- I.e.: x86-64 & aarch64
- Default Linux GNU GCC Compiler

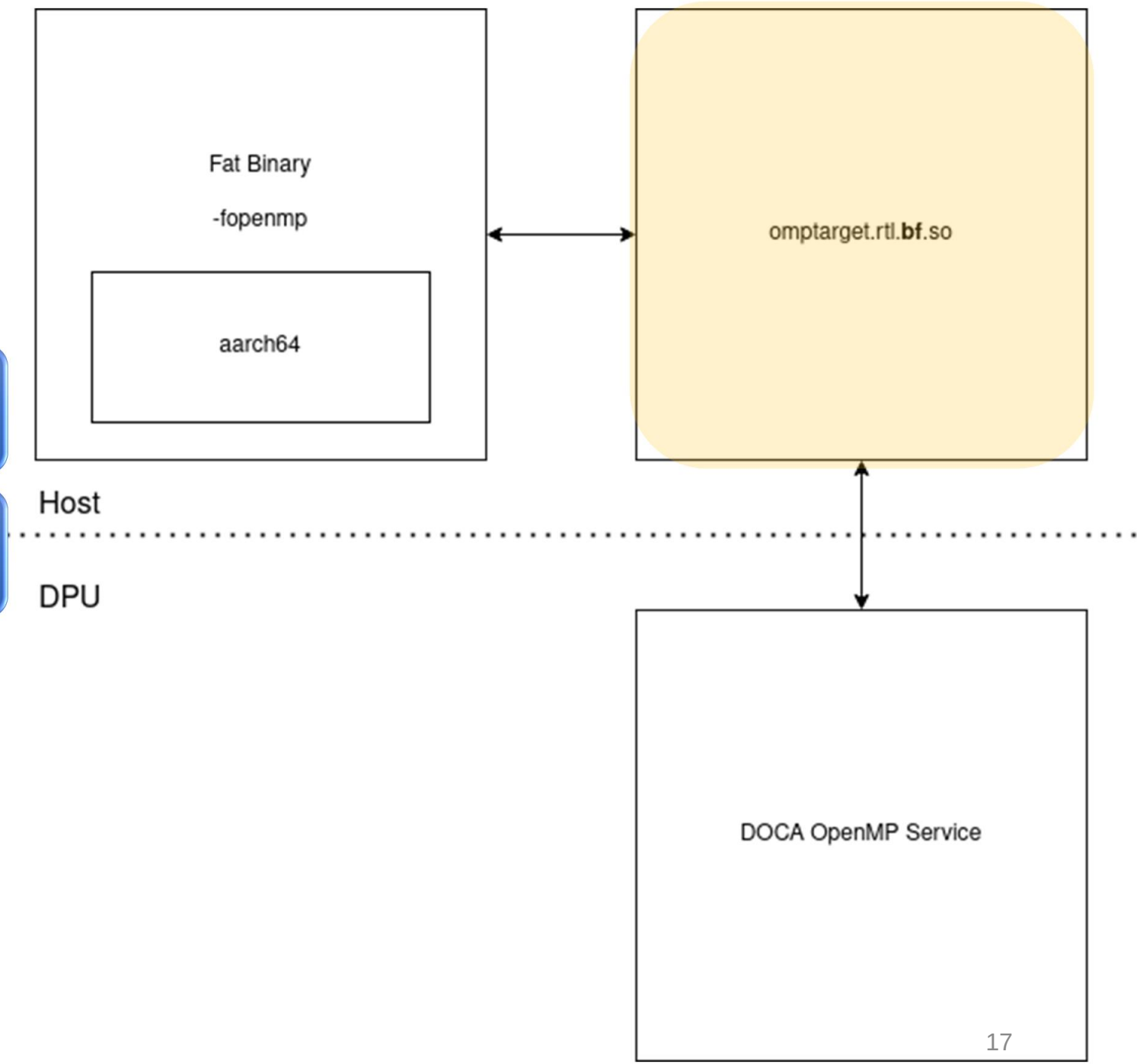


Target Plugin

Communication with the DPU using DOCA SCC

Send commands to

- Load binary.
- Allocate and exchange data.
- Execute target blocks.



DOCA OpenMP Service

Receive commands using DOCA SCC

Load binary

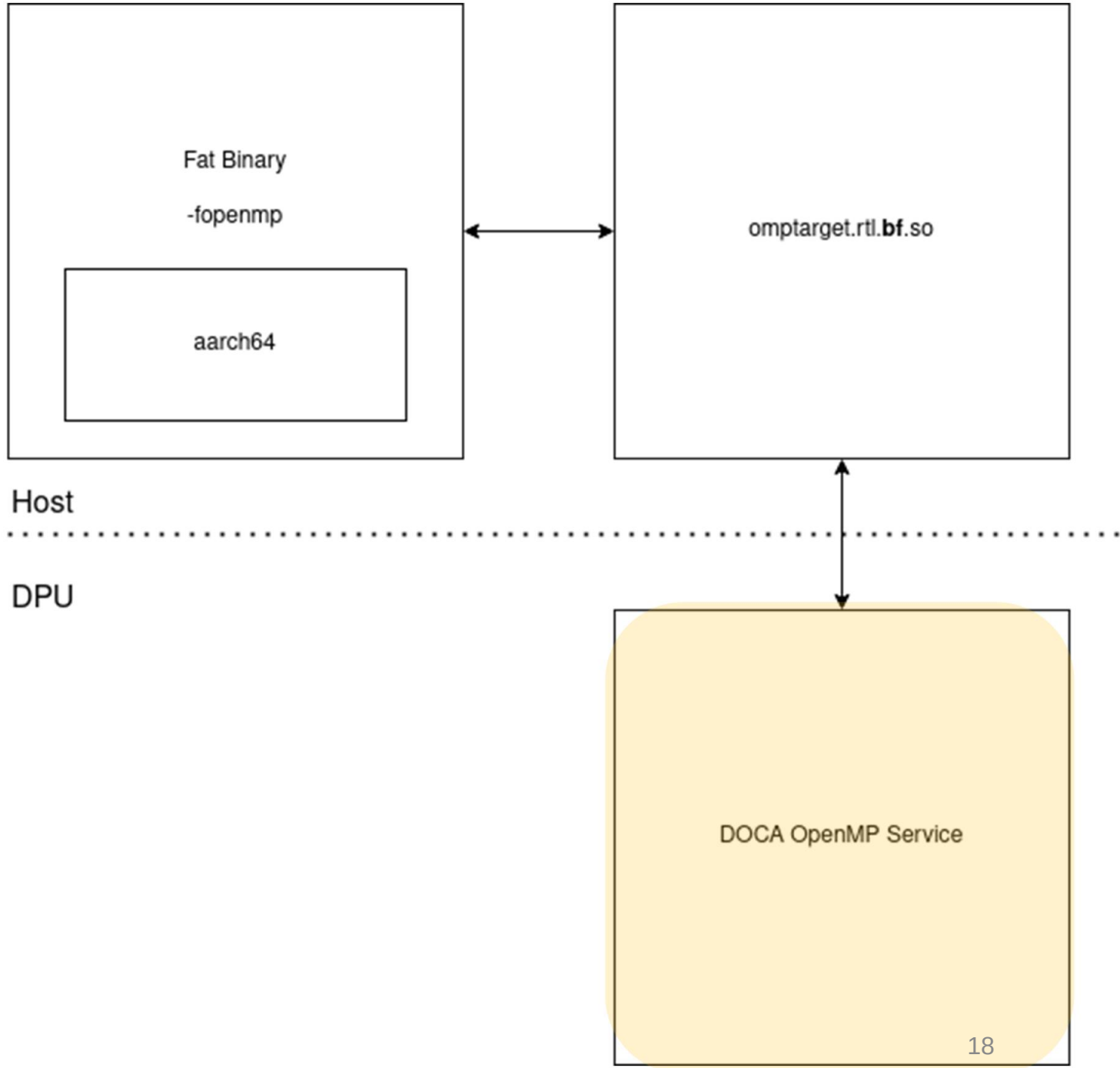
- Save image as a shared object
- Open the image [dlopen]
- Send back handles of symbols to host [dlsym]

Data operations

- Allocate/Delete [malloc/free]
- Send/Retrieve [DOCA SCC]

Target block execution

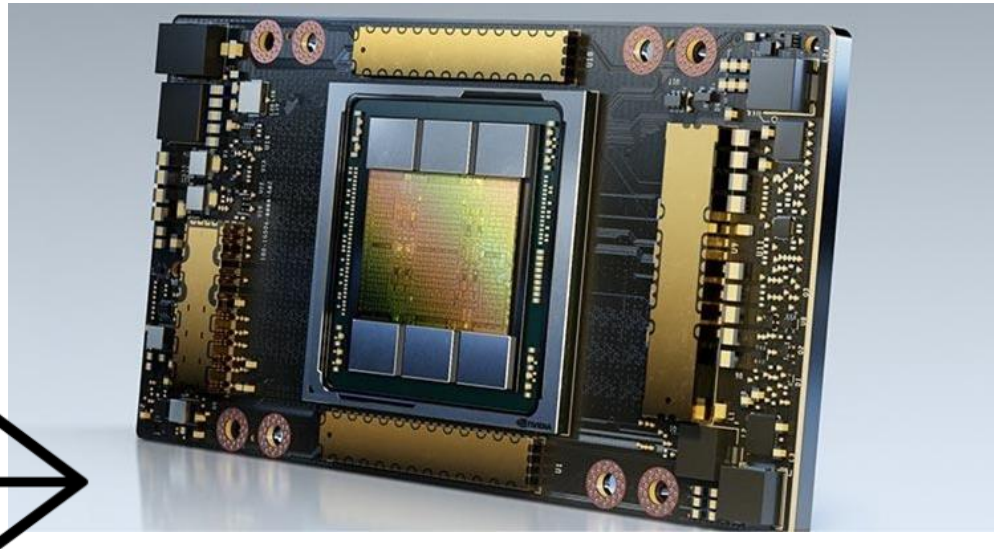
- Run target handle [ffi_prep_cif, ffi_call]



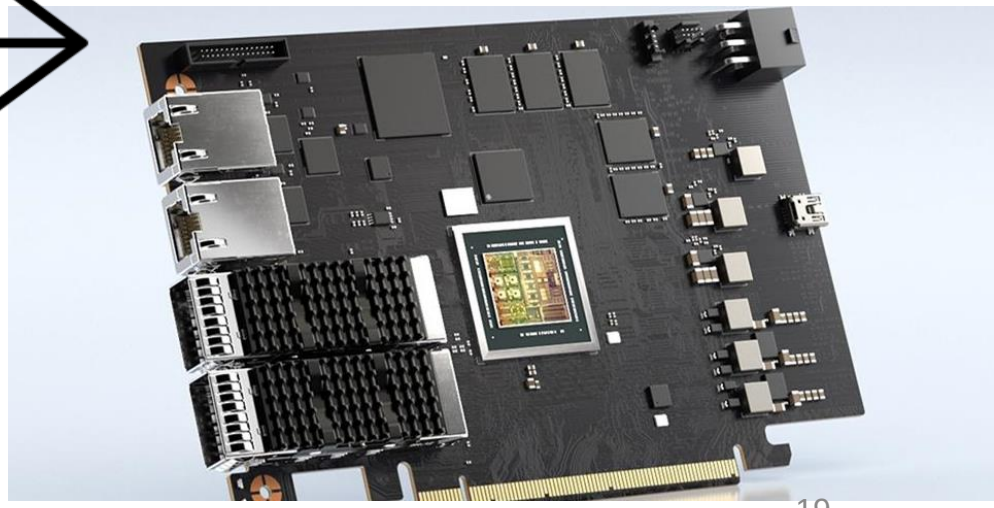
DPU Offload Programming with Standard OpenMP

```
int main ()
{
        /**************/
        /* exec in host */
        /**************/

        #pragma omp target device (  )
        {
                /***************/
                /* exec in target */
                /***************/
        }
}
```

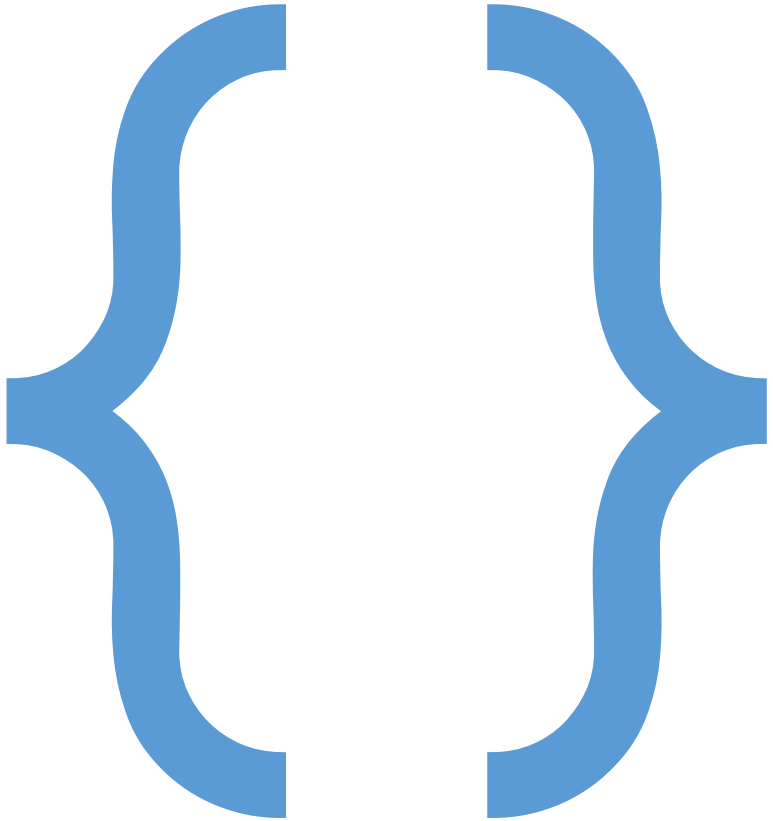
GPU



DPU

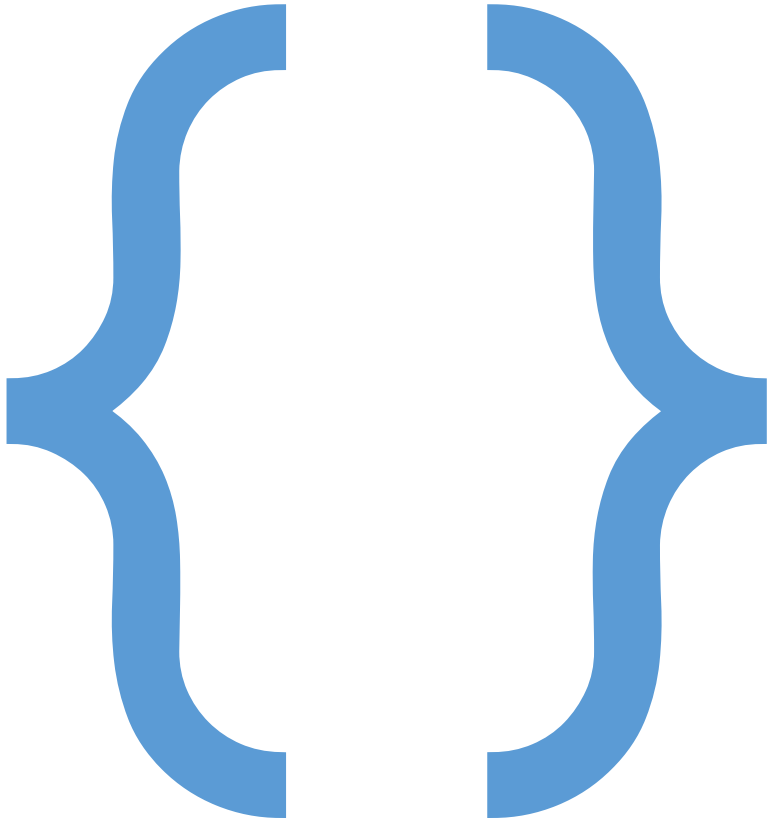


Basic Sintax



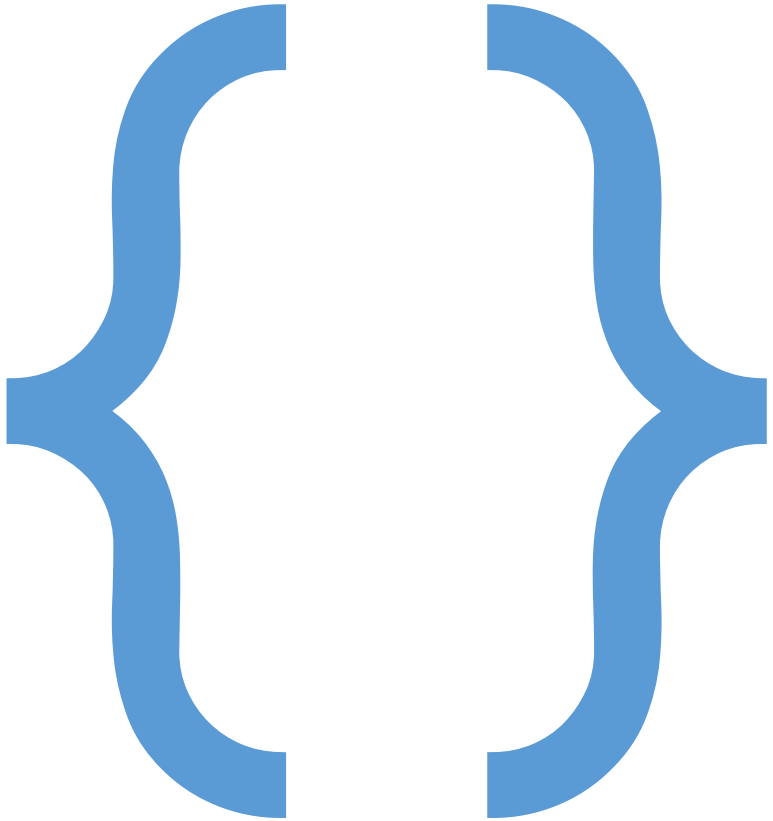
```
// exec in host

#pragma omp target
{
            // exec in target
}
```

Basic Sintax



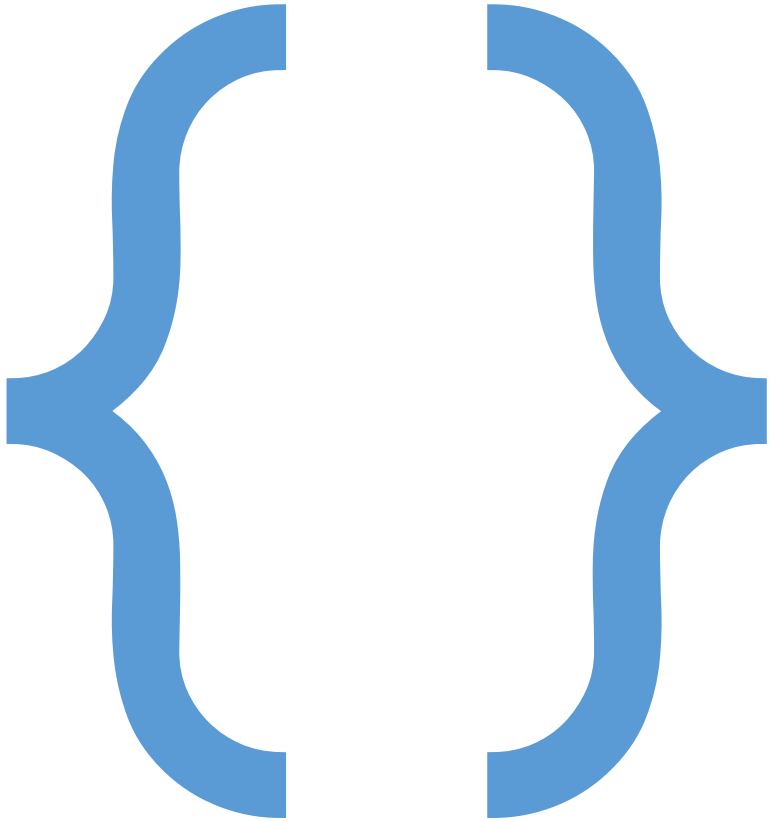
```
$ llvm-omp-device-info

Device (4):
   CUDA Driver Version:                10020
   CUDA Device Number:                 0
   Device Name:                        Tesla V100-SXM2-16GB
   Global Memory Size:                 16911433728 bytes
   Number of Multiprocessors:          80
   Concurrent Copy and Execution:      Yes
   Total Constant Memory:              65536 bytes
   Max Shared Memory per Block:        49152 bytes
   Registers per Block:                65536
   ...
   Compute Capabilities:               70
```

Basic Sintax



```
$ llvm-omp-device-info

Device (7):
    BlueField DPU device
        iface      :  ib0
        ib dev     :  mlx5_2
        doca id    :  2
        pci addr   :  42:00:0
        comm channel:
            max msg size             :   4080
            max send    queue size   :   8192
            max receive queue size   :   8192
...
```

Basic Sintax



```
// exec in host

#define __DPUID__ 7
#pragma omp target device(__DPUID__)
{
        // exec in target
}
```

How to Use it?

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

```c
#include <omp.h>
#include <stdio.h>

int main()
{
#pragma omp target
#pragma omp parallel
        puts("Hey! OpenMP even work in DPU...");

        return 0;
}
```

Host

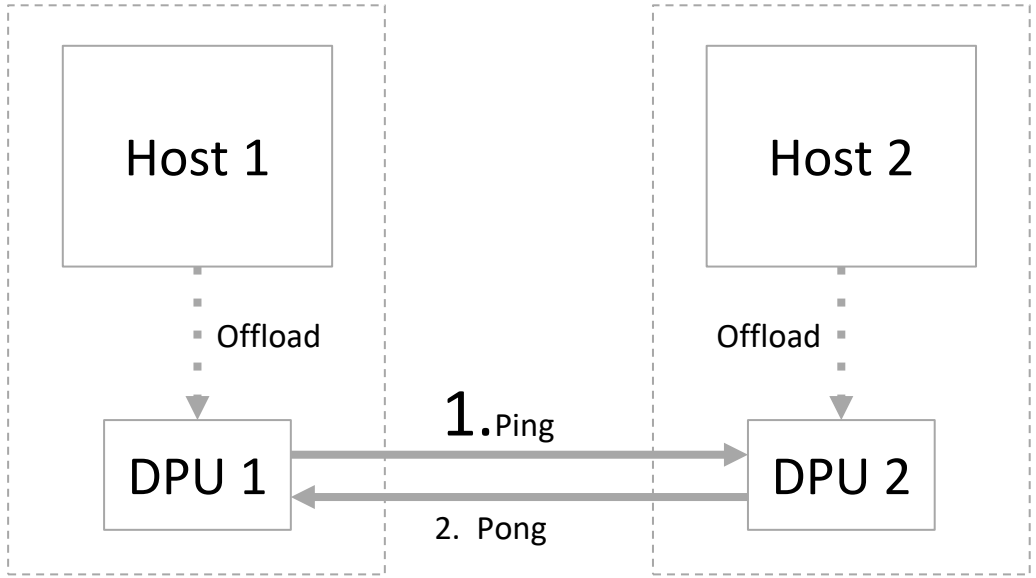
```
uthmanhere@thor013:~/omp_exp/labs_sc23/task_b/build$
```

DPU

```
uthmanhere@thorbf3a013:~$
```

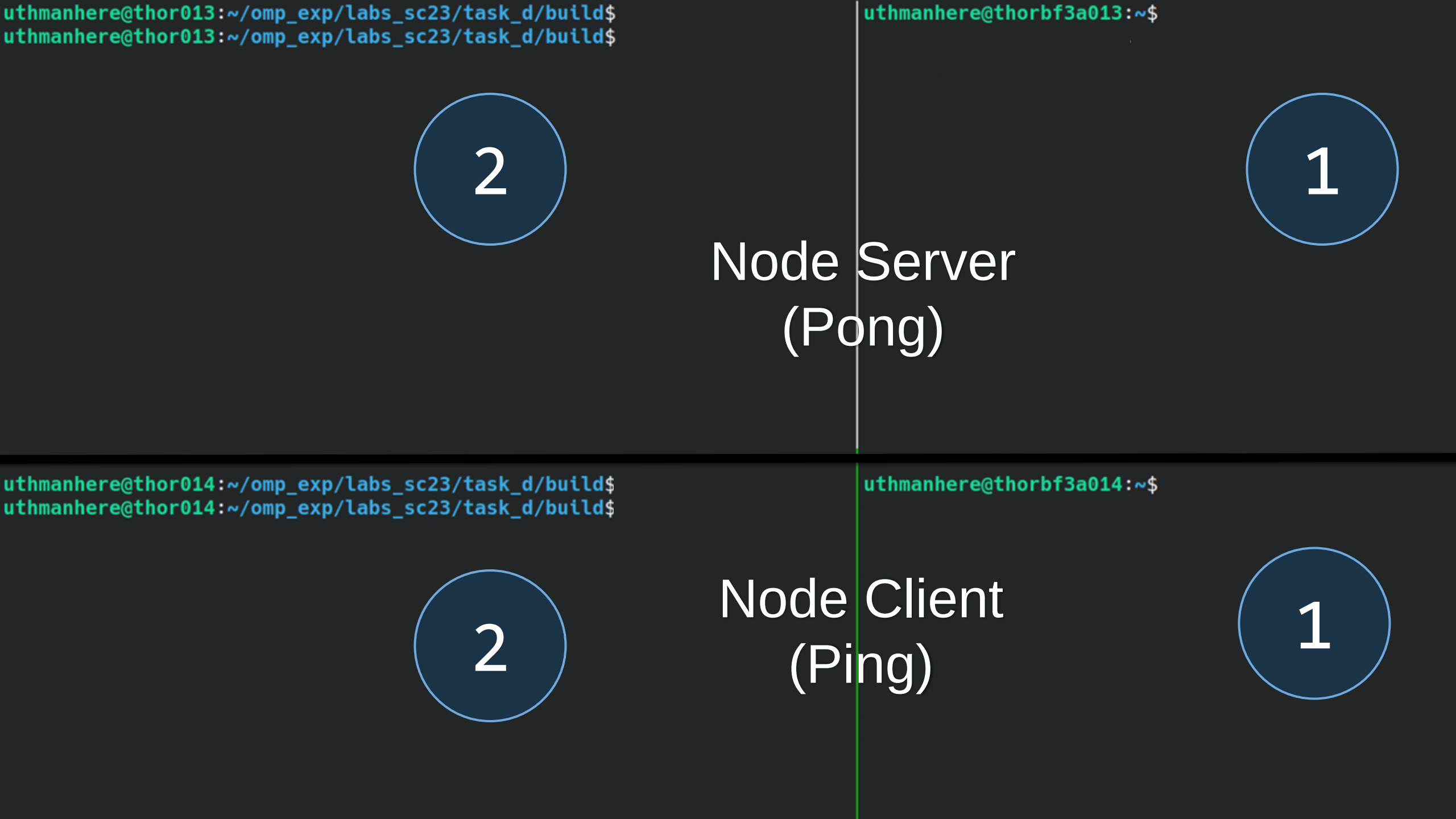
Asynchronous TCP I/O in DPU using OpenMP (1/2)

```
#pragma omp target nowait
{
        // initialize tcp sockets, listen, bind, connect, and accept.
        fd = tcp_init(mode);

        // compute and communicate asynchronously on DPU
        ping_pong(fd, mode);

        close(fd);
}
#pragma omp taskwait
```



Asynchronous TCP I/O in DPU using OpenMP (2/2)

```
void ping_pong(int fd, char mode)
{
        int i, m;
        m = 0;

        for (i = 0; i < _ITERATIONS_; ++i) {
                if (mode == 'c') {
                        printf("to   server > %02d\n", m);
                        send(fd, &m, sizeof(m), 0);
                        recv(fd, &m, sizeof(m), 0);
                        ++m;
                } else {
                        recv(fd, &m, sizeof(m), 0);
                        ++m;
                        printf("to   client > %02d\n", m);
                        send(fd, &m, sizeof(m), 0);
                }
        }

}
```


uthmanhere@thor013:~/omp_exp/labs_sc23/task_d/build$
uthmanhere@thor013:~/omp_exp/labs_sc23/task_d/build$
2
uthmanhere@thorbf3a013:~$
1
Node Server
(Pong)
uthmanhere@thor014:~/omp_exp/labs_sc23/task_d/build$
uthmanhere@thor014:~/omp_exp/labs_sc23/task_d/build$
2
uthmanhere@thorbf3a014:~$
1
Node Client
(Ping)

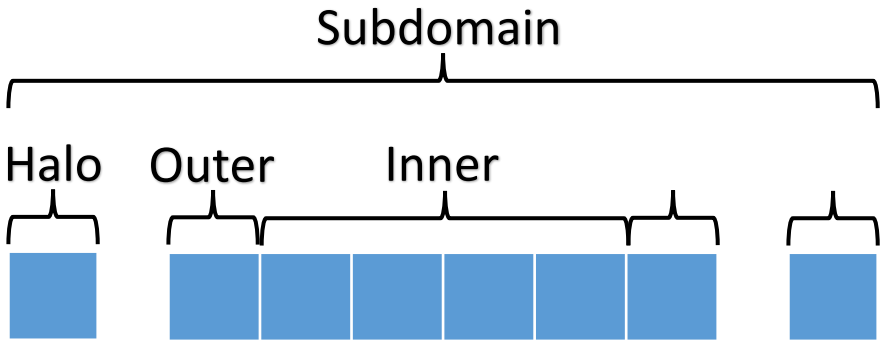
ODOS MPI

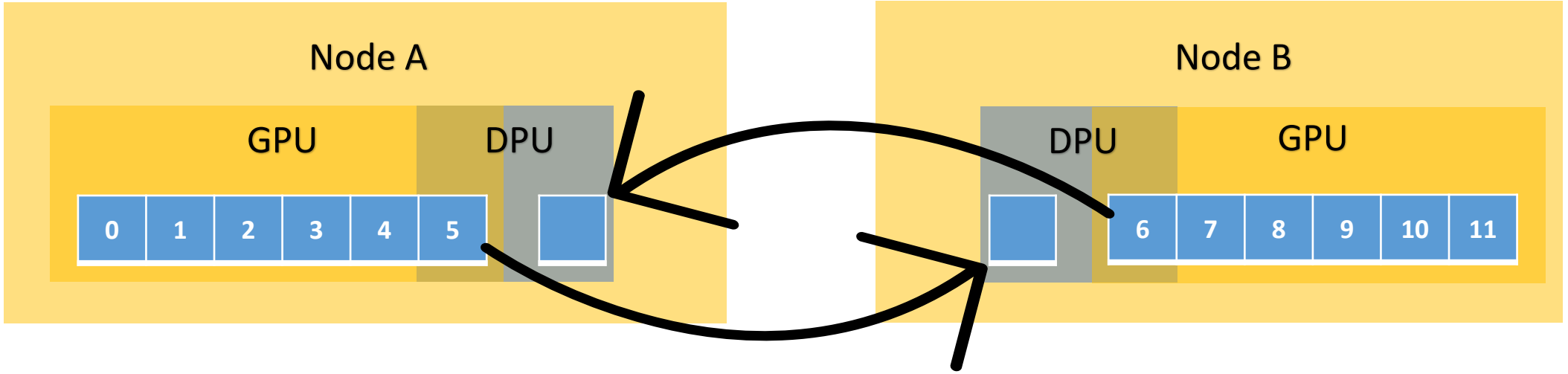
```
#pragma omp target
{
  if ( rank == 0 ) {
   MPI_Send( &buf, 1, MPI_INT, dst, 0, MPI_COMM_WORLD );
  } else {
   MPI_Recv( &recv_data, 1, MPI_INT, src, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE );
  }
}

#pragma omp target
{
      MPI_Alltoall(send_buffer, 1, MPI_INT, recv_buffer, 1, MPI_INT, MPI_COMM_WORLD);
}
```

Halo Exchange Offloading

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Halo Exchange Offloading





```
int curr_subdomain[SIZE + THICKNESS * 2];
int next_subdomain[SIZE + THICKNESS * 2];
#pragma omp target data map(to: curr_subdomain[THICKNESS:SIZE-THICKNESS])
                        map(from: next_subdomain[THICKNESS:SIZE-THICKNESS])
                        device (GPU_ID) nowait
{
    #pragma omp target teams num_teams(2)
    {
        #pragma omp parallel
        {
            next_subdomain[THICKNESS] = compute_inner(curr_subdomain[THICKNESS:SIZE-THICKNESS])
        }
    }
}
#pragma omp target data map(to: curr_subdomain[0:THICKNESS*2], curr_subdomain[SIZE-THICKNESS*2:SIZE]
                        map(from: next_subdomain[0:THICKNESS*2], next_subdomain[SIZE-THICKNESS*2:SIZE])
                        device (DPU_ID) nowait
{
    #pragma omp parallel
    #pragma omp single
    {
        #pragma omp task
        {
            next_subdomain[THICKNESS] = compute_outer(curr_subdomain[0:THICKNESS*2], left);
        }
        #pragma omp task
        {
            next_subdomain[SIZE-THICKNESS*2] = compute_outer(curr_subdomain[SIZE-THICKNESS*2:SIZE], right);
        }
    }
}
#pragma omp barrier
curr_subdomain = copy_domain(next_subdomain);
```

GPU

DPU

DPU Offloaded Function

```
int* compute_outer(int* data, int rank_id)
{
    int halo[THICKNESS], outer_domain[THICKNESS], tag_id = 0;
    MPI_Request request;
    MPI_Status status;

    MPI_Irecv(halo, THICKNESS, MPI_INT, rank_id, tag_id, MPI_COMM_WORLD, &request);
    outer_domain = compute(data);
    MPI_Send(outer_domain, THICKNESS, MPI_INT, rank_id, tag_id, MPI_COMM_WORLD);
    MPI_Wait(&request, &status);
    return outer_domain;
}
```

Wrapping up

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Conclusions

SmartNICs in HPC are highly promising:

Additional co-processor closer to the network.

Applications can be accelerated by:

Overlapping and offloading communication/computation stages.

Easy coding is crucial:

To extend SmartNICs usage among HPC users (OpenMP, Kokkos, SYCL...).

Collaboration Opportunities

- Extend ODOS to other SmartNICs or programming models.
- Explore other uses of SmartNICs within HPC.
- Implementation of SmartNIC-enabled scientific codes.

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

ODOS team:

- Muhammad Usman
- Mariano Benito
- Sergio Iserte
- Antonio J. Peña

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

See you around! 👋

FGCS SI on HPC Heterogeneous (Sub) Systems

Deadline May 31st, 2025

7th Marenostrum Hackathon, Barcelona (Spain)

October (TBD), 2025

sergio.iserte@bsc.es

AccelCom
Accelerators & Communications for HPC