EPICURE

Webinar series on profiling

From traces to optimization with NVIDIA Nsight Systems on heterogeneous HPC platforms

Laura Bellentani

Scientific Computing and Application Performance Team @ CINECA

Co-funded by the European Union

EuroHPC Joint Undertaking

This project has received funding from the European High Performance Computing Joint Undertaking under grant agreementNo.101139786. Views and opinions expressed are, however, those of the author(s) only and do not necessarily reflect those of the European Union orEuroHPC Joint Undertaking. Neither the European Union nor the granting authority can be held responsible for them.

Content

- Common bottlenecks of single-GPU programming
- Performance engineering workflow with NVIDIA NSight Systems tracing tool
- Asynchronous programming and multistream
- Awareness in MPI communications
- Examples from support activity (epicure project and beyond)

The presentation is based on the OpenACC offload model but everything we cover about Nsight Systems applies regardless of the underlying offload model

EPICURE

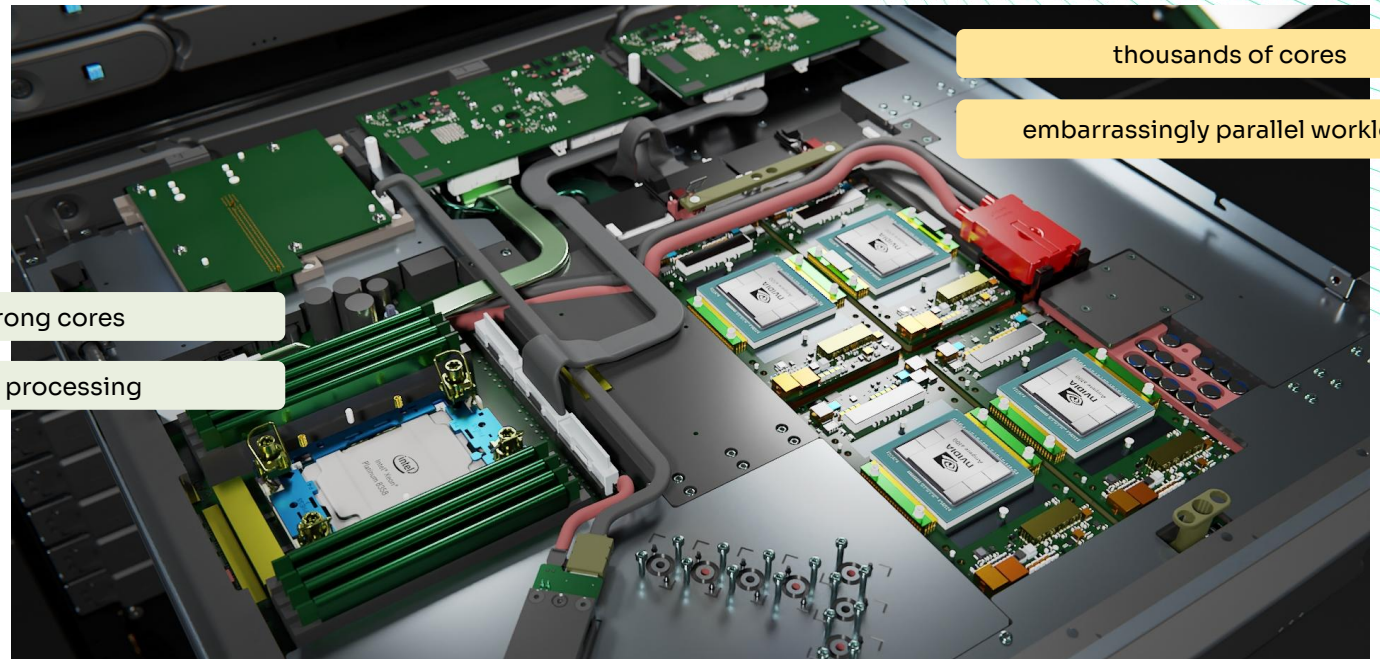
Unlocking European-level HPC Support

Common Bottlenecks of Single-GPU Programming

Heterogeneity in HPC

several strong cores

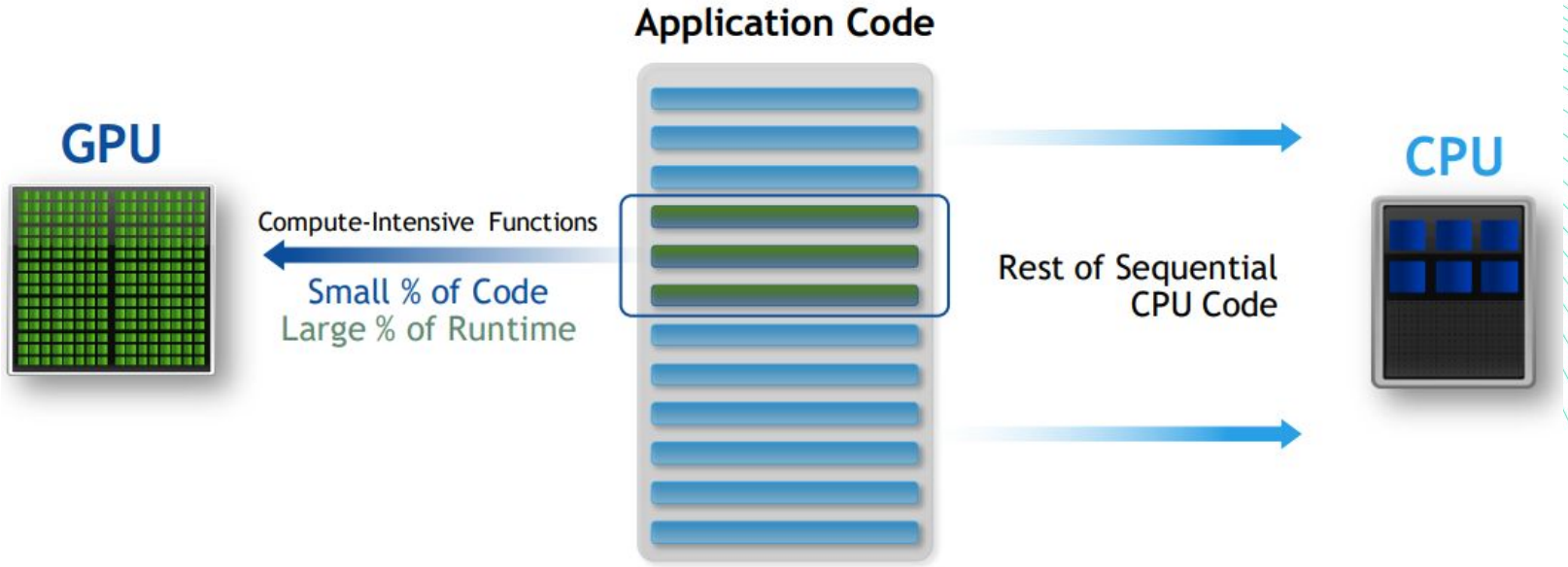
specialized processing



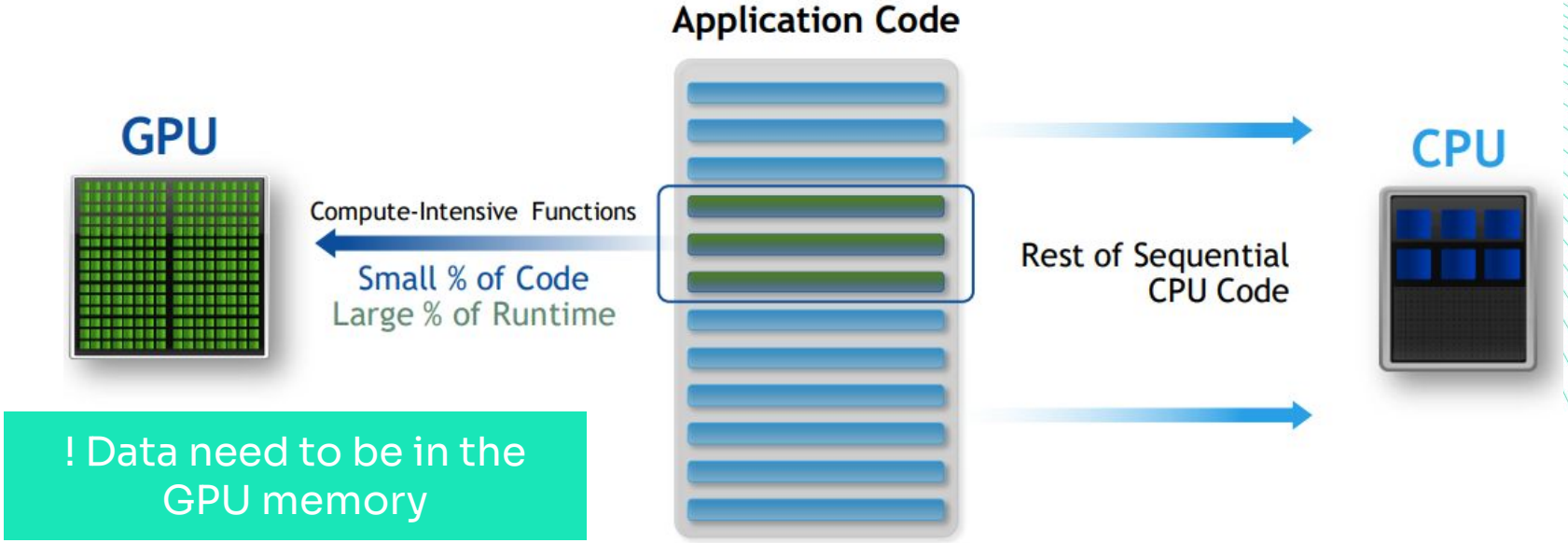
thousands of cores

embarrassingly parallel workloads

Heterogeneity in HPC

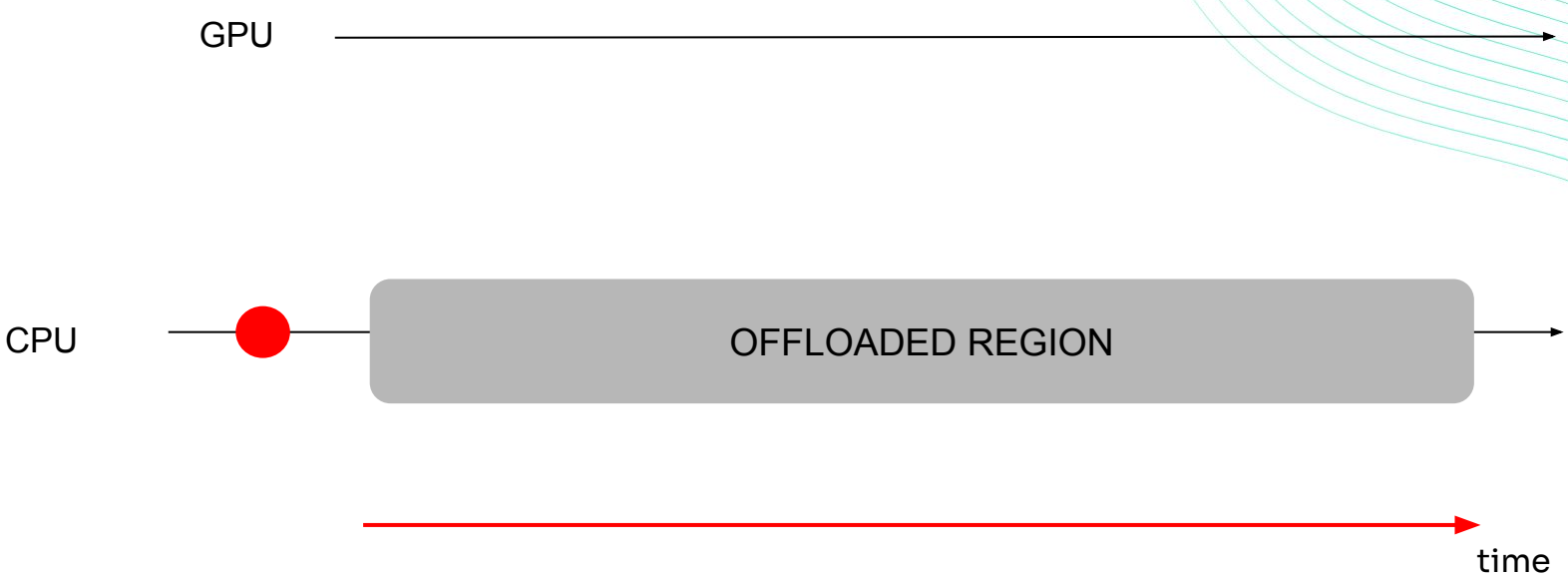


Heterogeneity in HPC

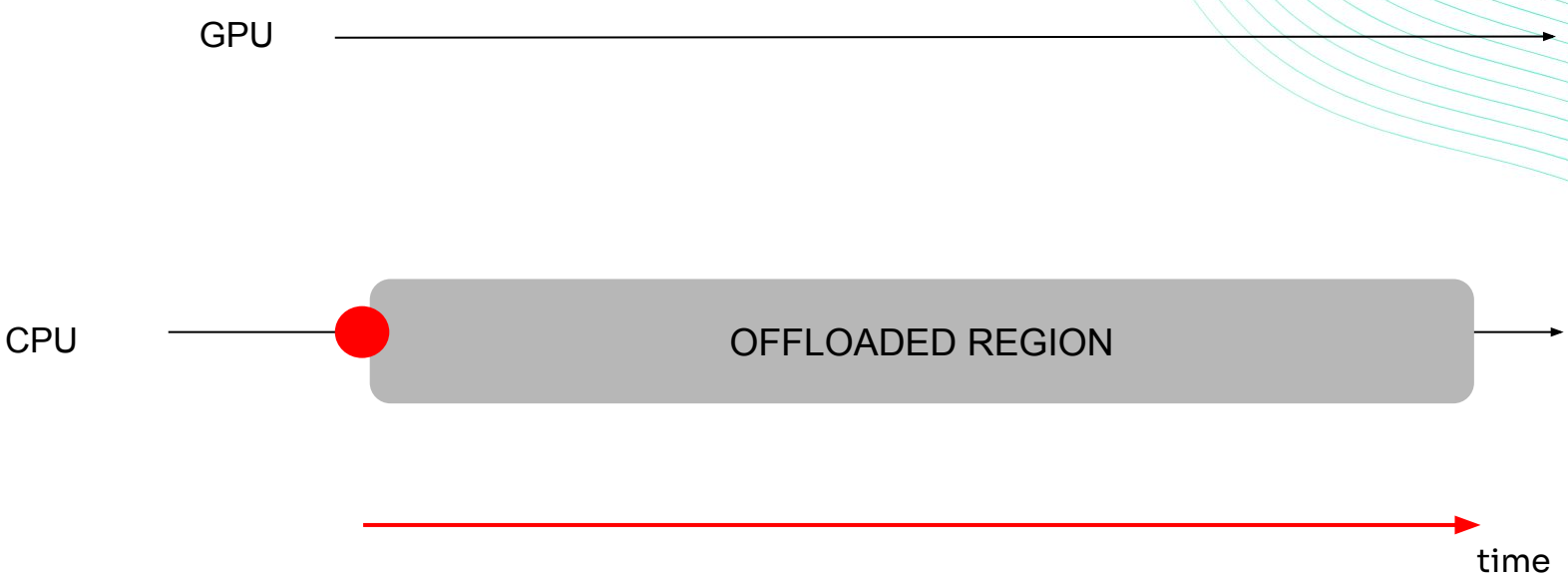


! Data need to be in the GPU memory

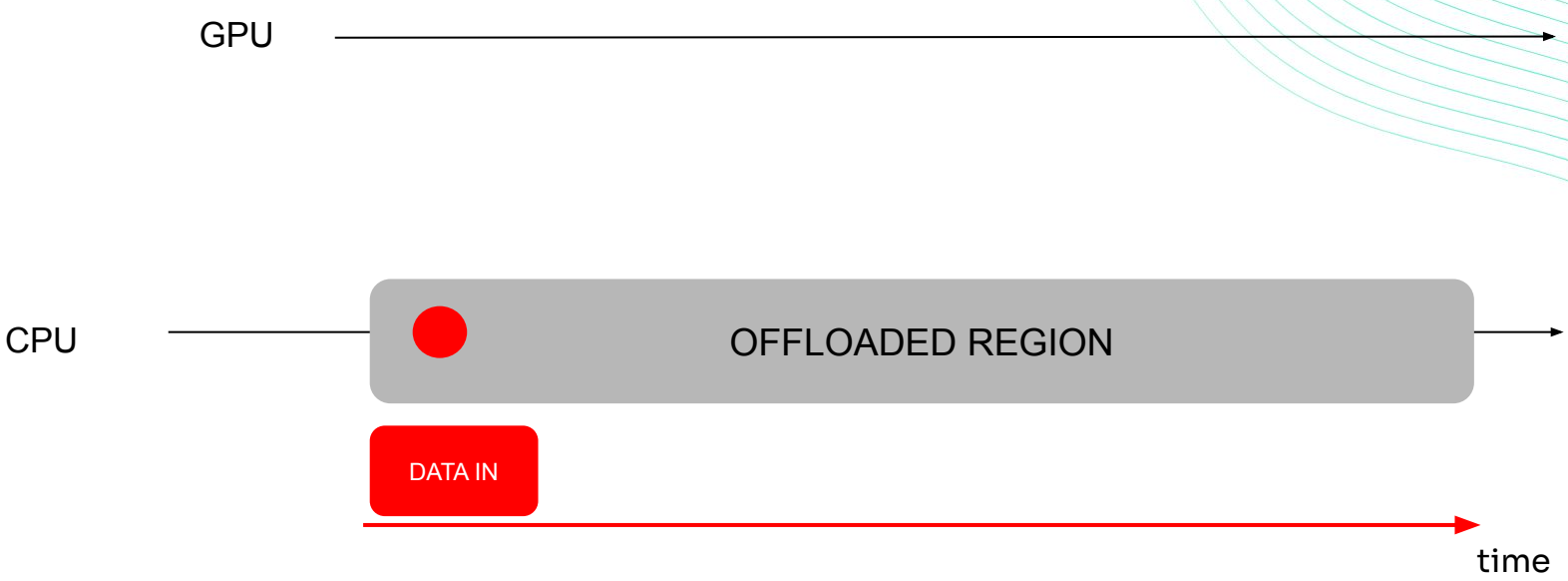
Heterogeneity in HPC



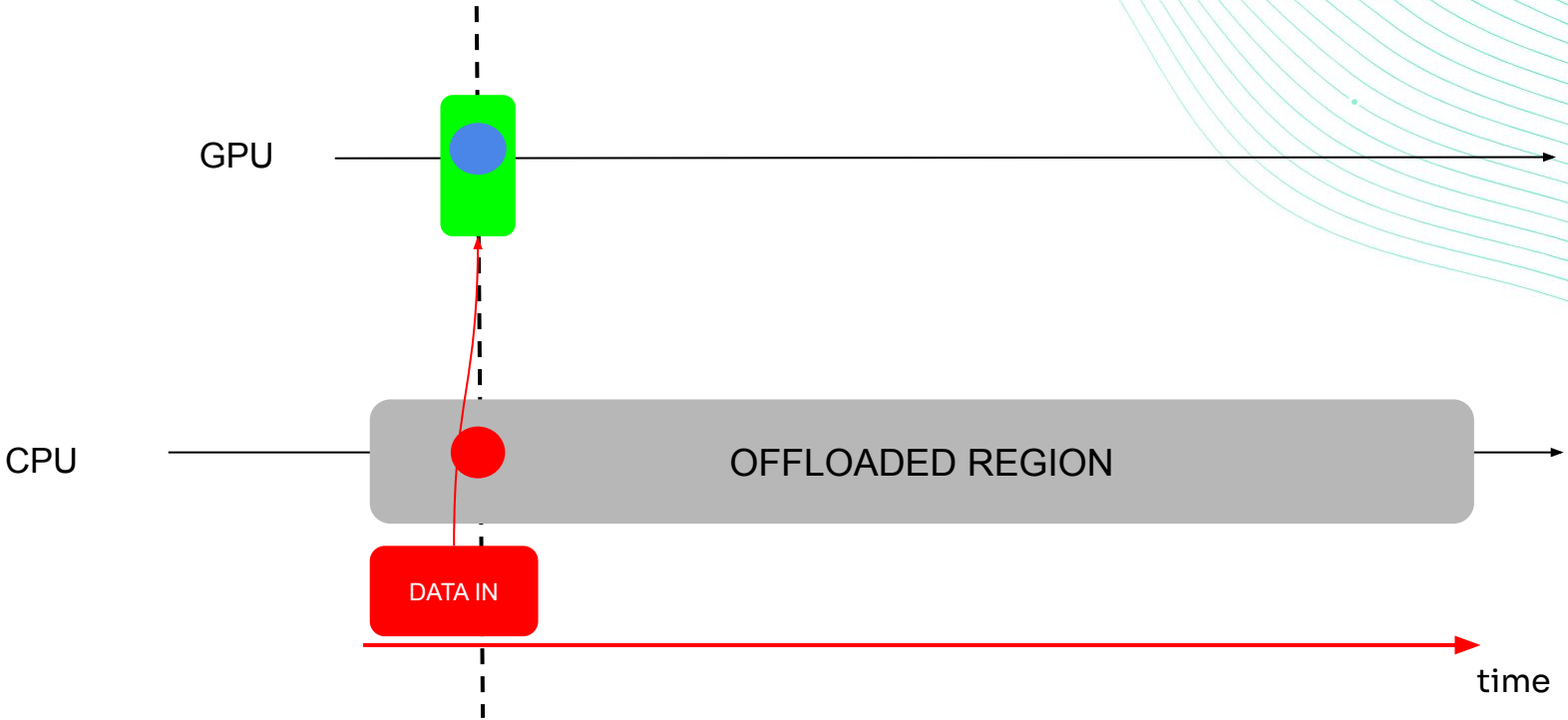
Heterogeneity in HPC



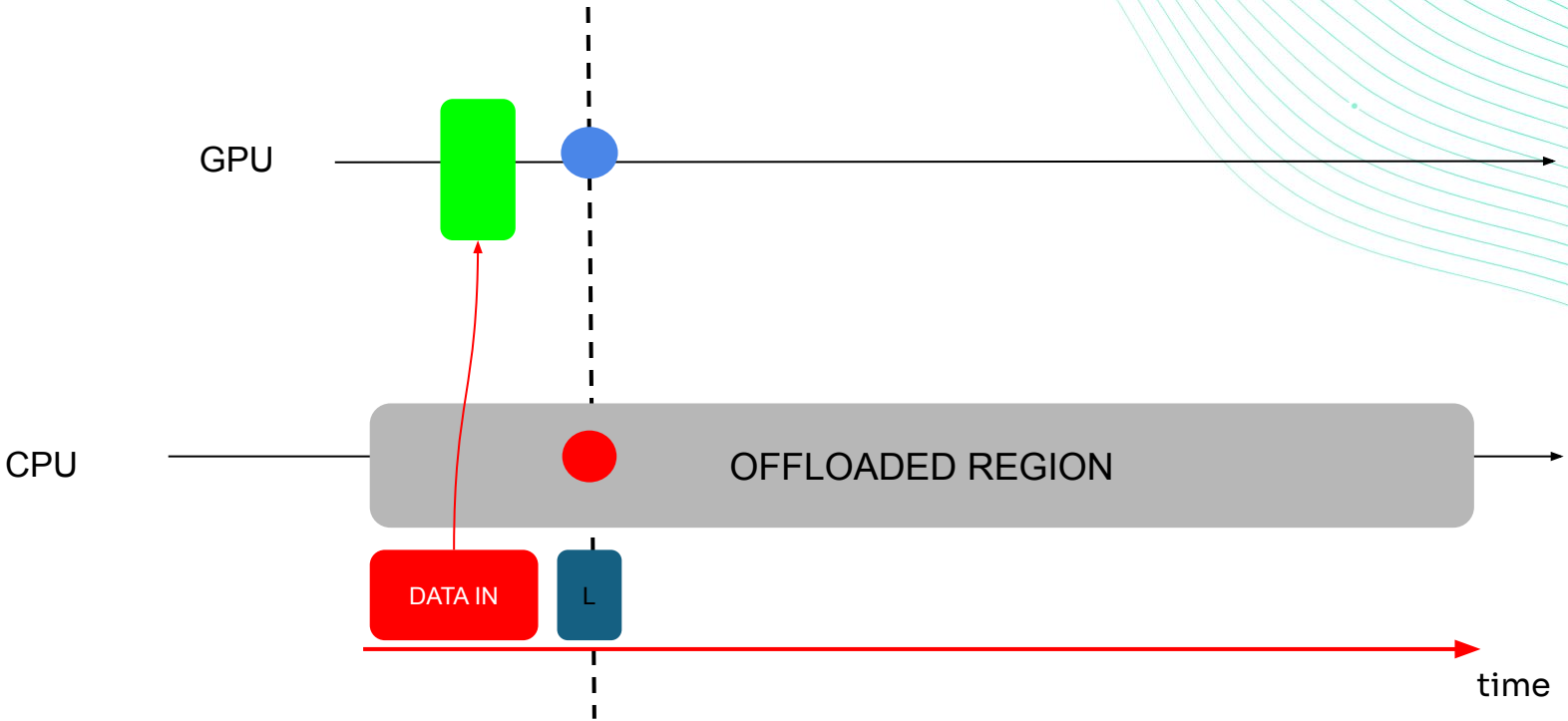
Heterogeneity in HPC



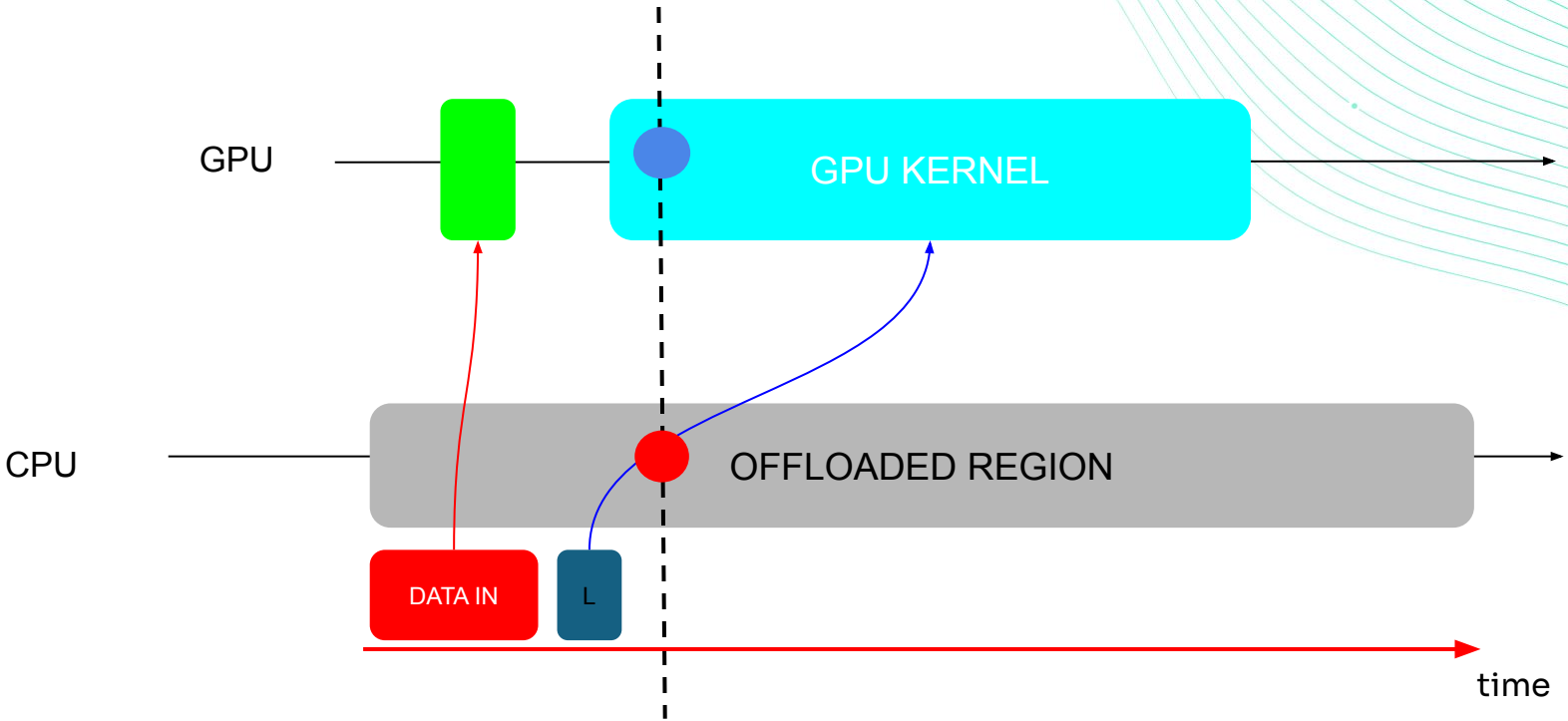
Heterogeneity in HPC



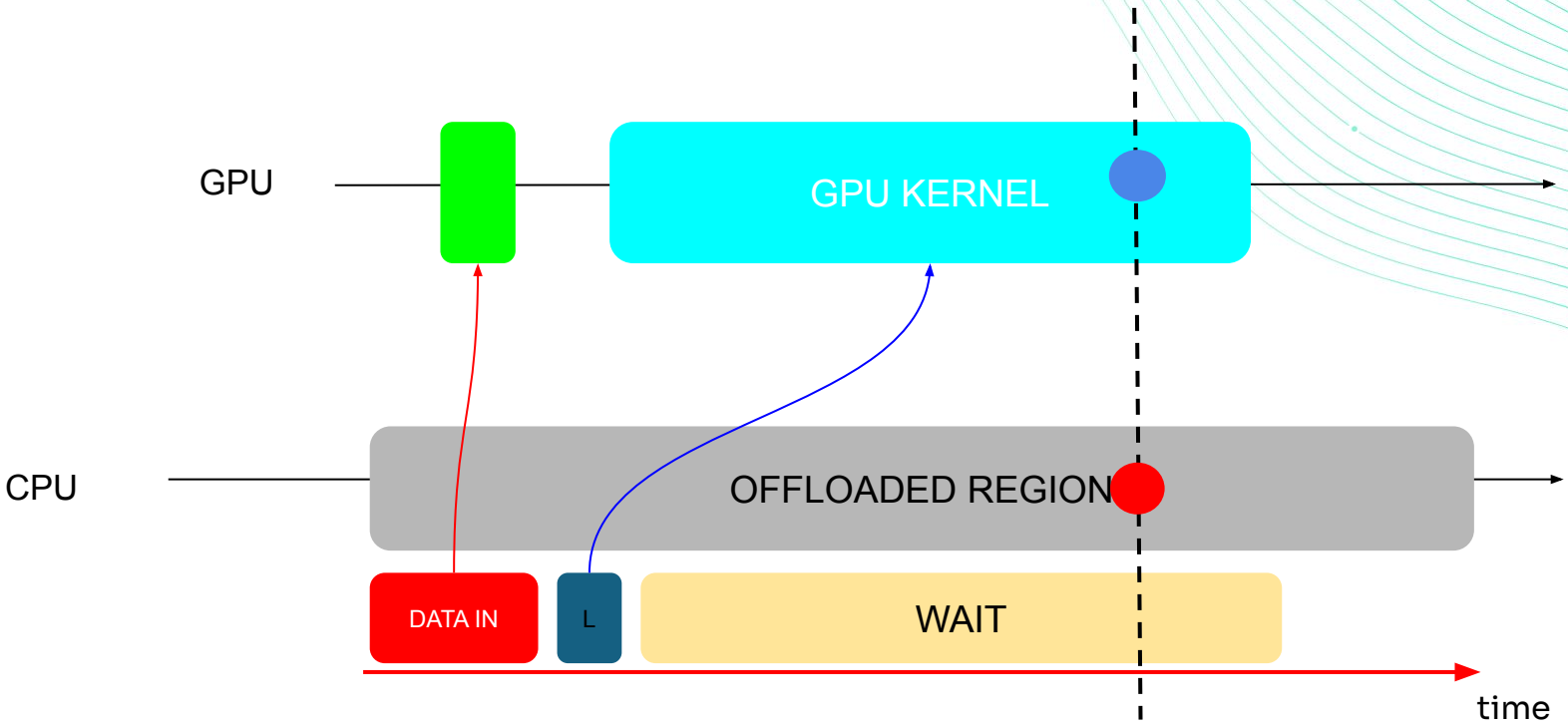
Heterogeneity in HPC



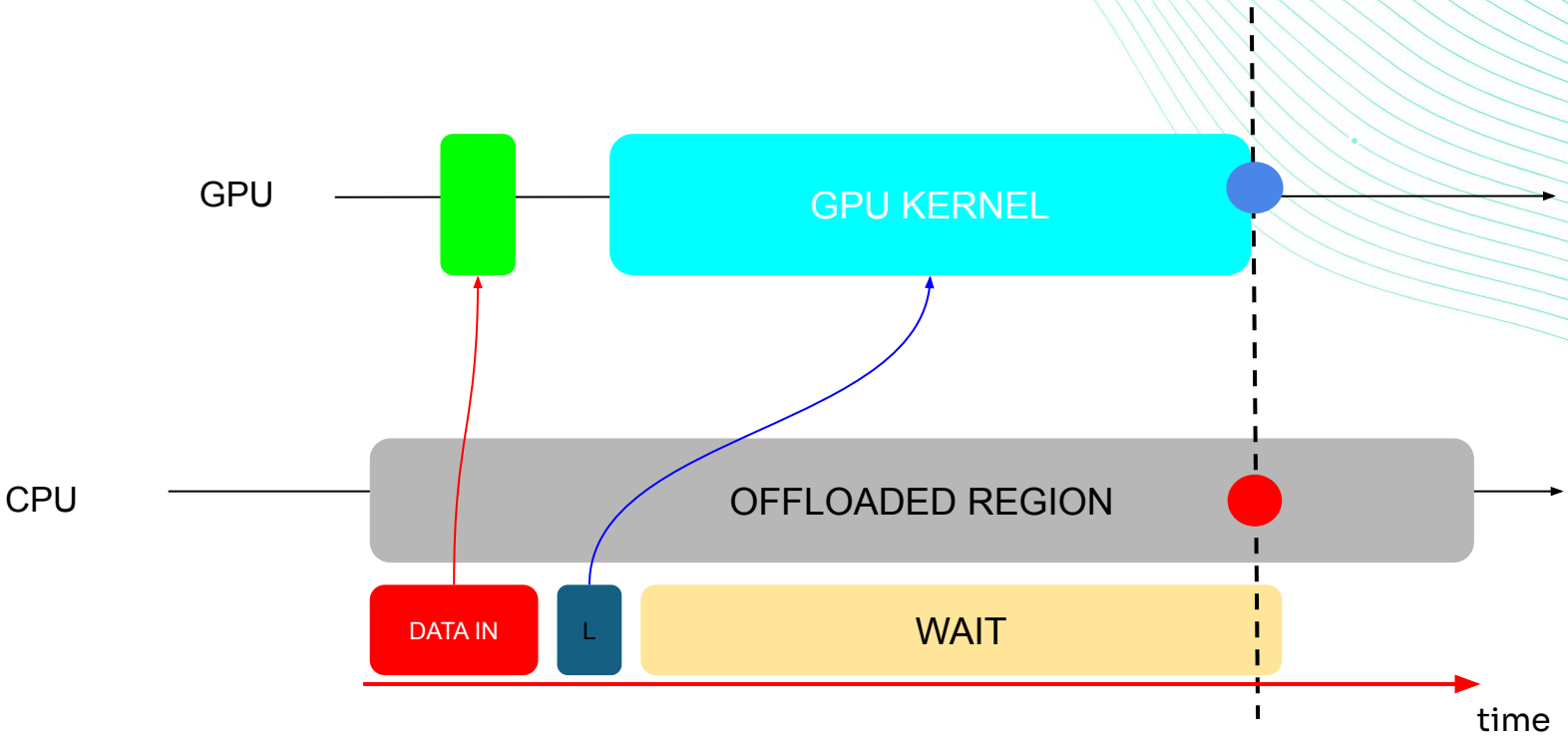
Heterogeneity in HPC



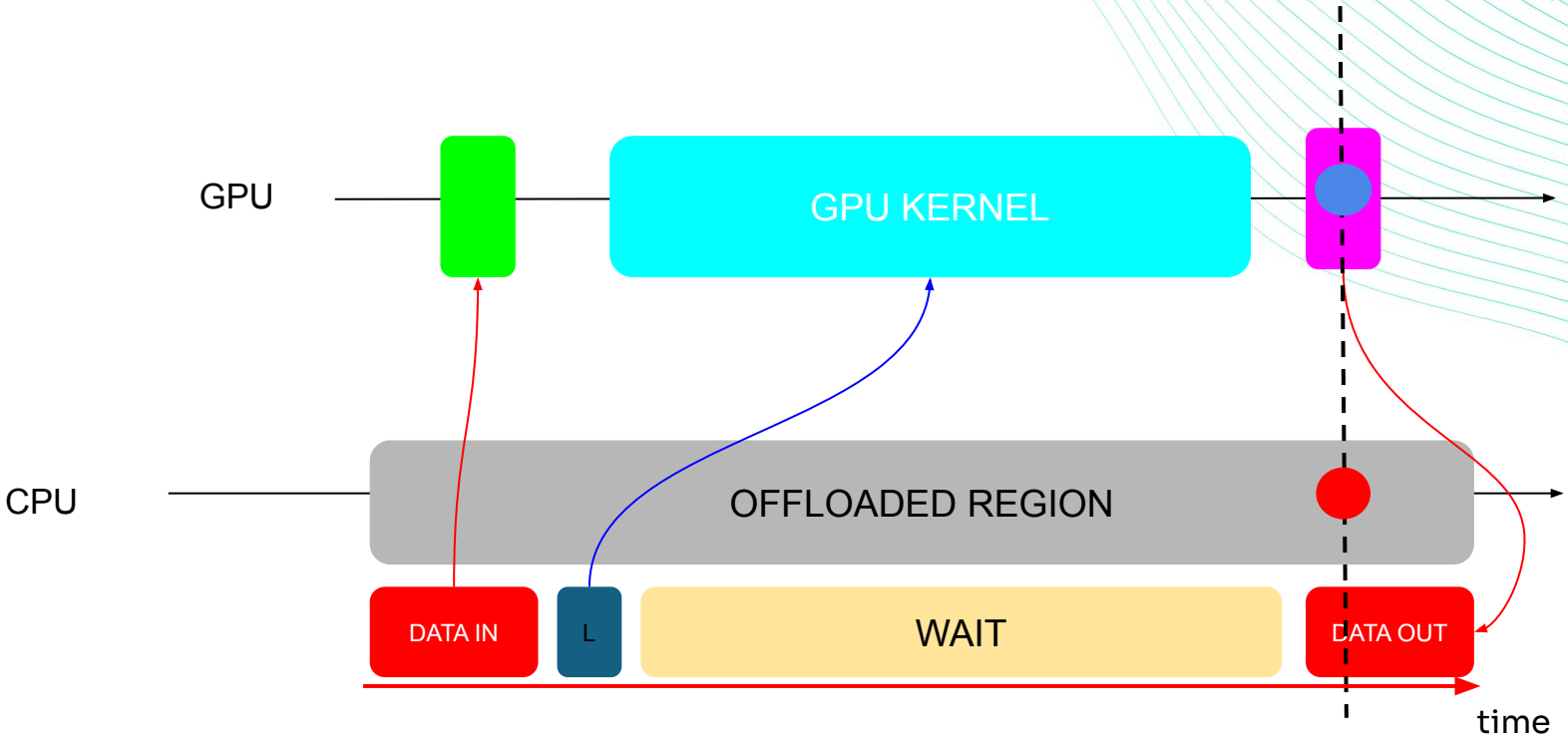
Heterogeneity in HPC



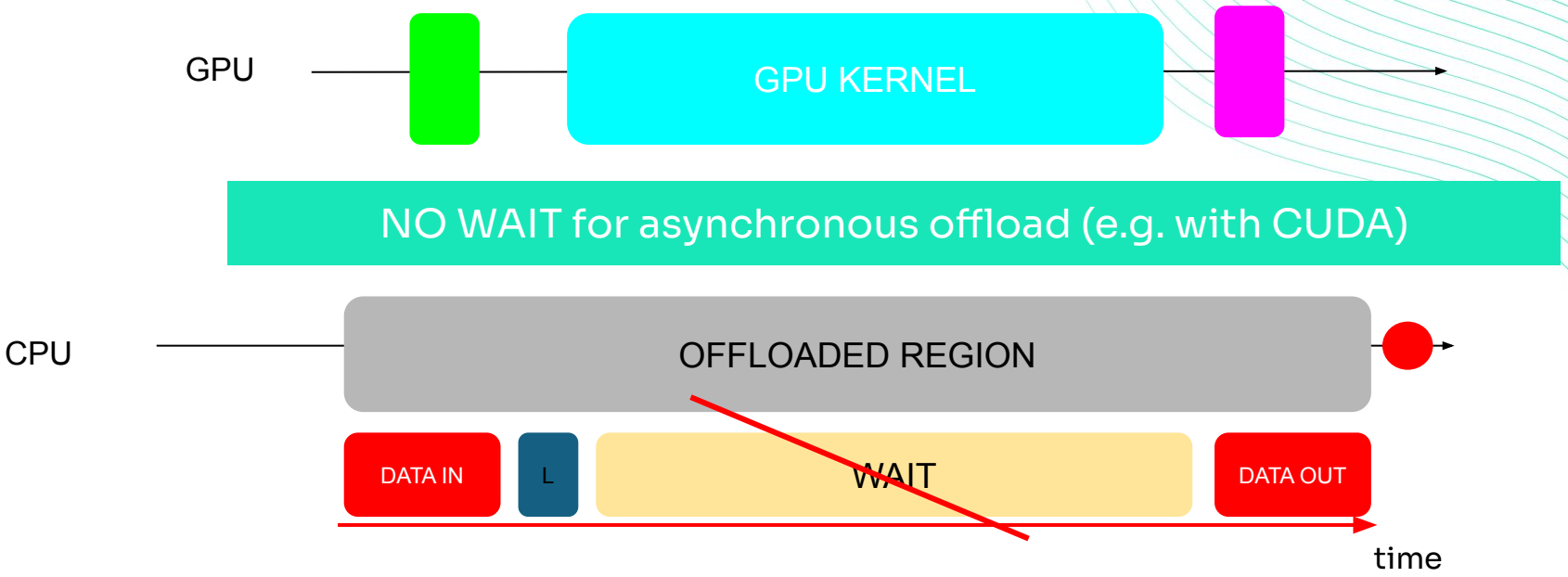
Heterogeneity in HPC



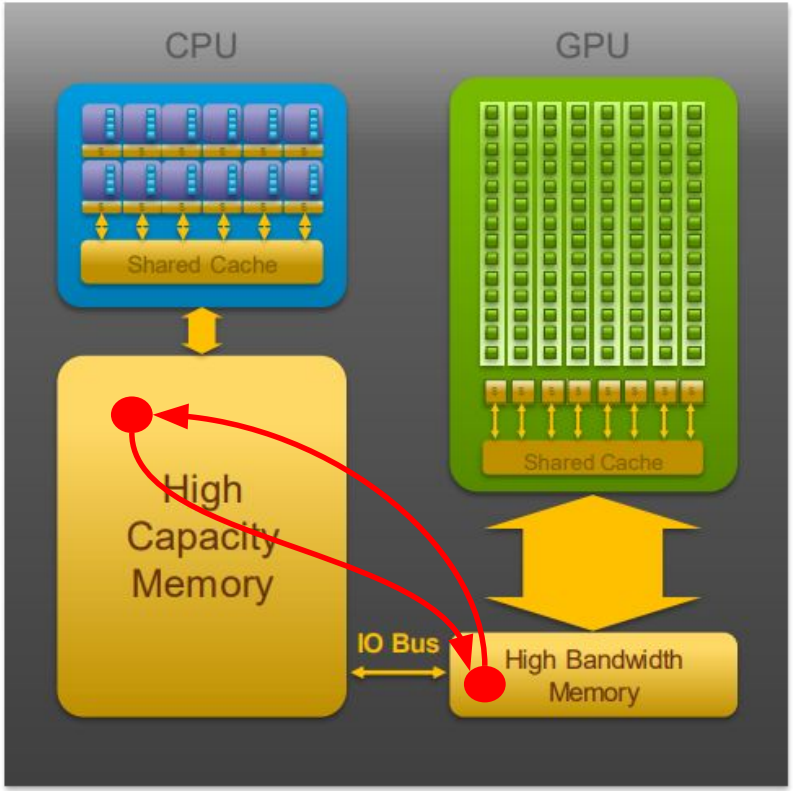
Heterogeneity in HPC



Heterogeneity in HPC



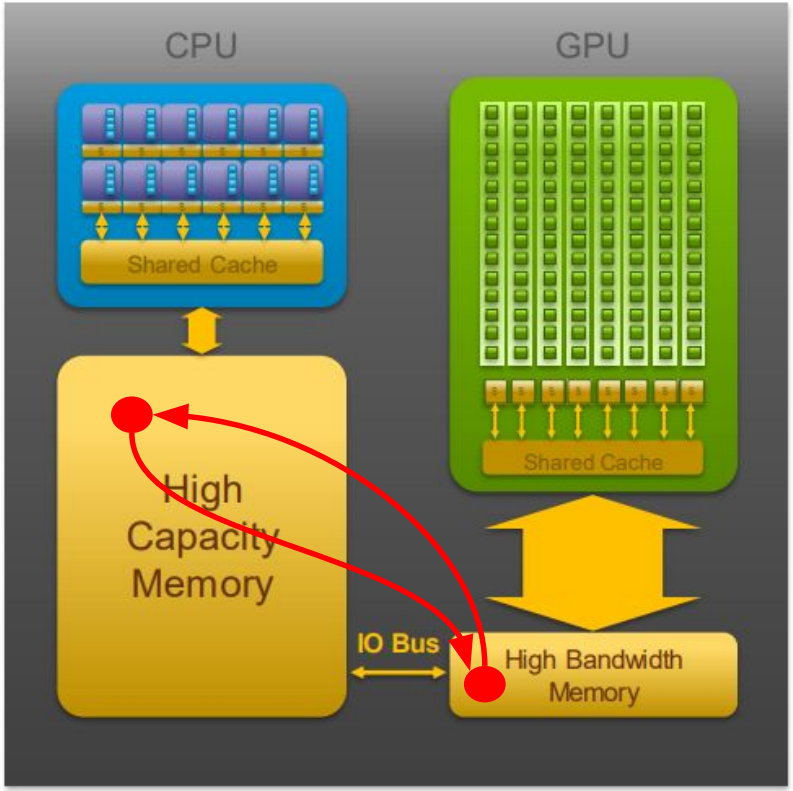
Data movements



- The program starts on the CPU
- GPUs and CPUs have separated memories
- GPUs need data in their own memory
- IO bus is slow compared to GPU BW

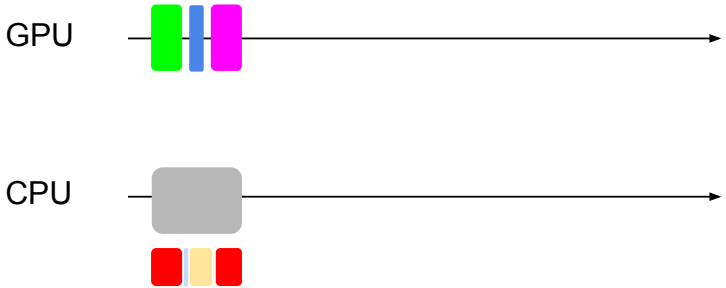
MOVING DATA IS EXPENSIVE

Data movements

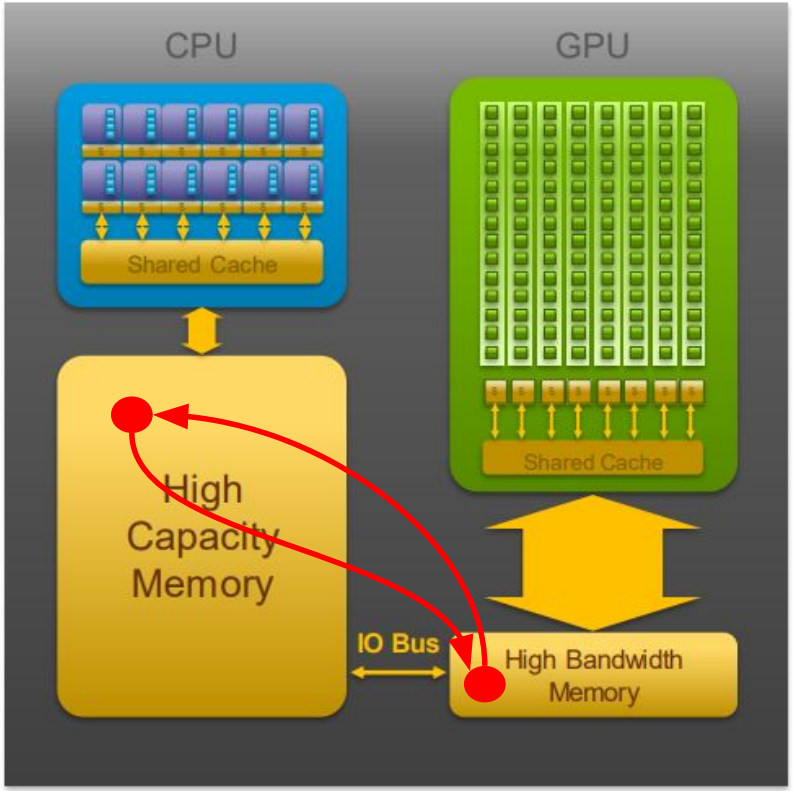


- The program starts on the CPU
- GPUs and CPUs have separated memories
- GPUs need data in their own memory
- IO bus is slow compared to GPU BW

MOVING DATA IS EXPENSIVE

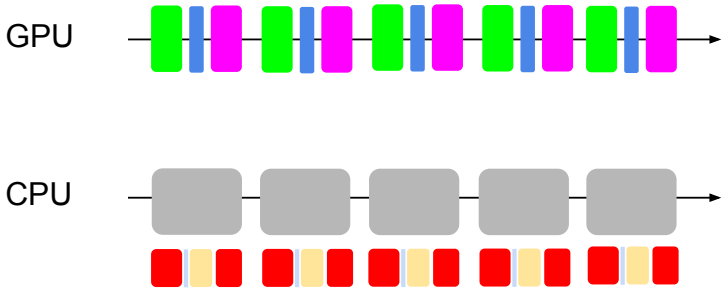


Data movements

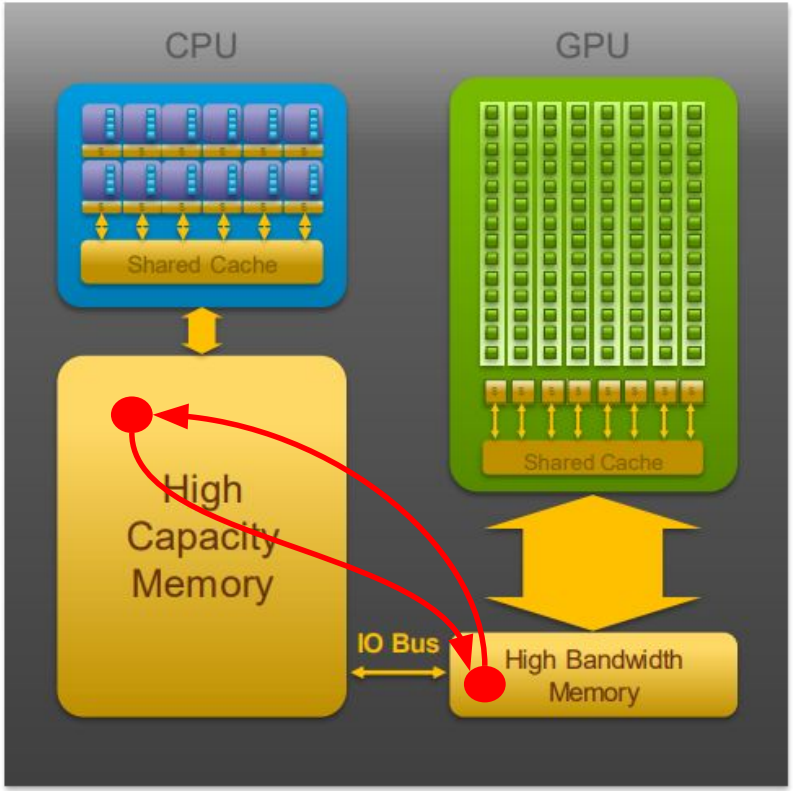


- The program starts on the CPU
- GPUs and CPUs have separated memories
- GPUs need data in their own memory
- IO bus is slow compared to GPU BW

MOVING DATA IS EXPENSIVE



Data movements

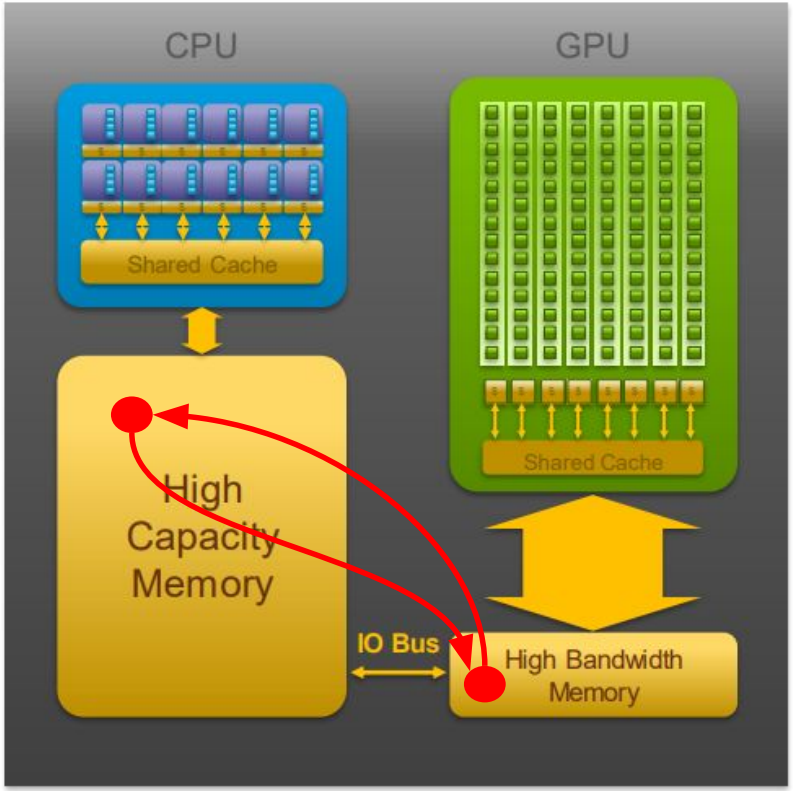


- The program starts on the CPU
- GPUs and CPUs have separated memories
- GPUs need data in their own memory
- IO bus is slow compared to GPU BW

MOVING DATA IS EXPENSIVE

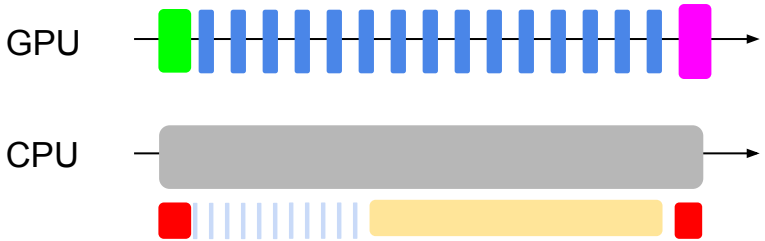
IMPROVE DATA LOCALITY

Data movements

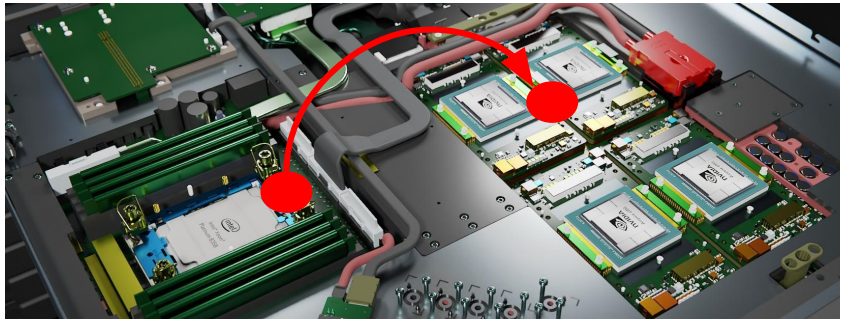


- The program starts on the CPU
- GPUs and CPUs have separated memories
- GPUs need data in their own memory
- IO bus is slow compared to GPU BW

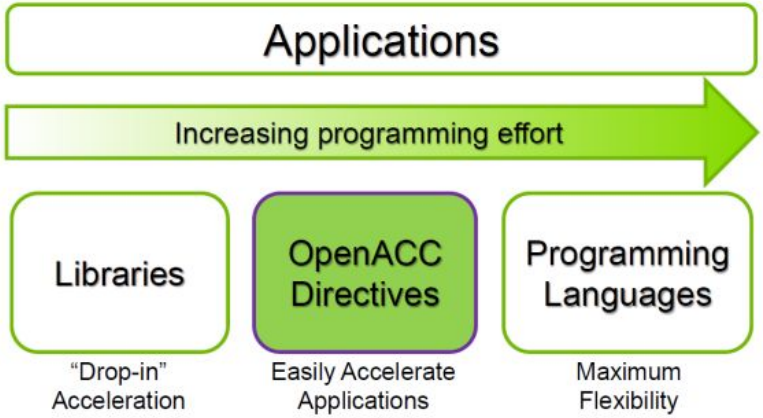
MOVING DATA IS EXPENSIVE

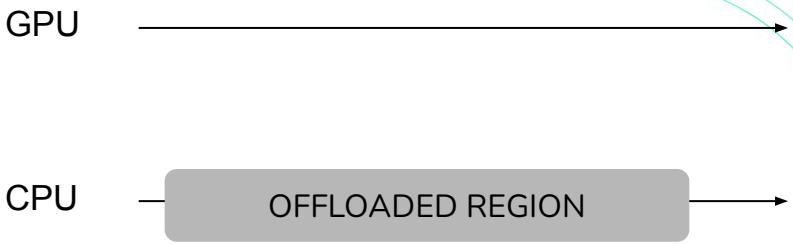


Kernel launch time

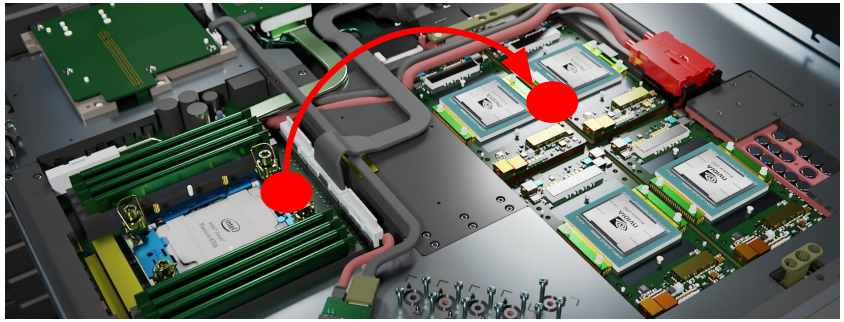


- The program starts on the CPU
- Many ways to offload kernels to GPUs (OpenACC, CUDA, CUDAlibraries…)
- There is always a latency between the launch of the kernel and its execution on the GPU (kernel launch time)

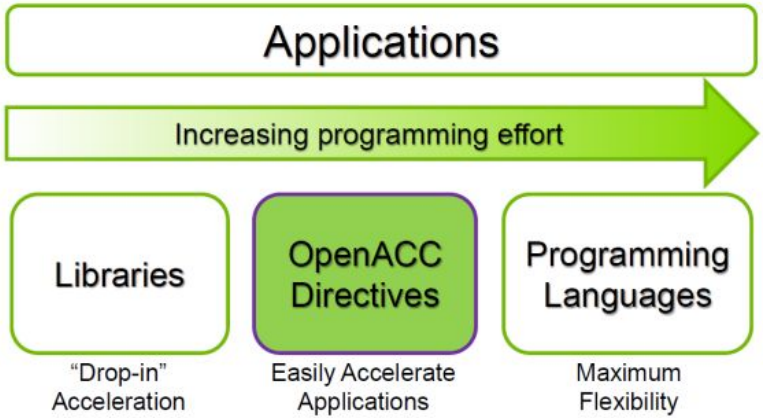


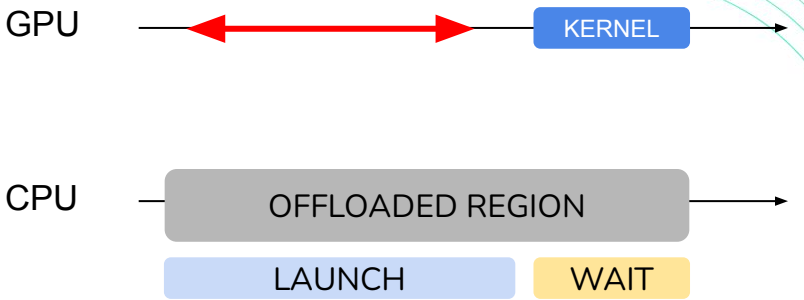


Kernel launch time

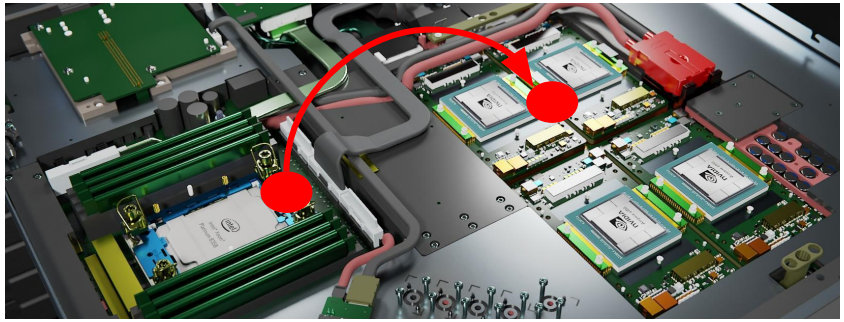


- The program starts on the CPU
- Many ways to offload kernels to GPUs (OpenACC, CUDA, CUDAlibraries…)
- There is always a latency between the launch of the kernel and its execution on the GPU (kernel launch time)

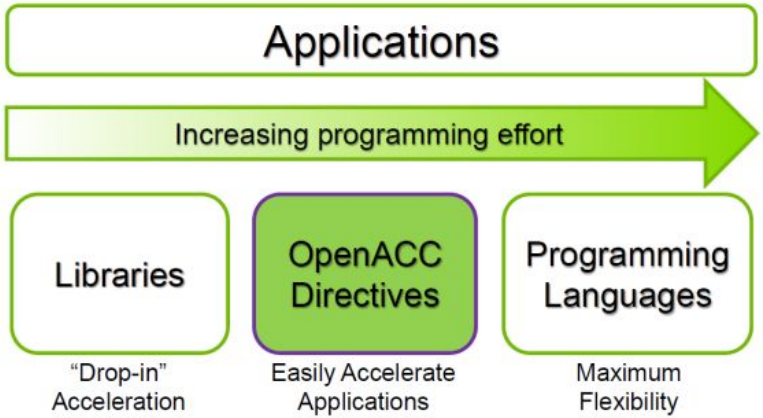


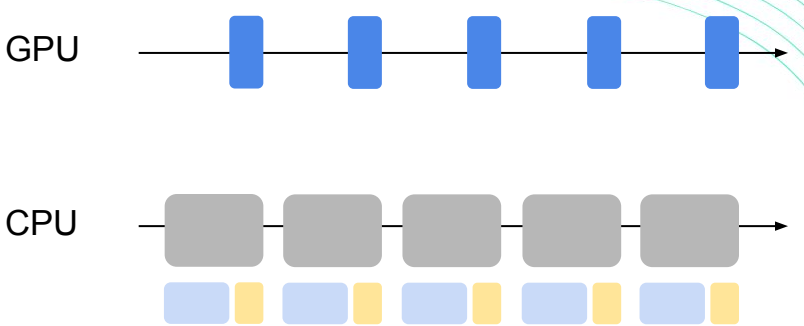


Kernel launch time

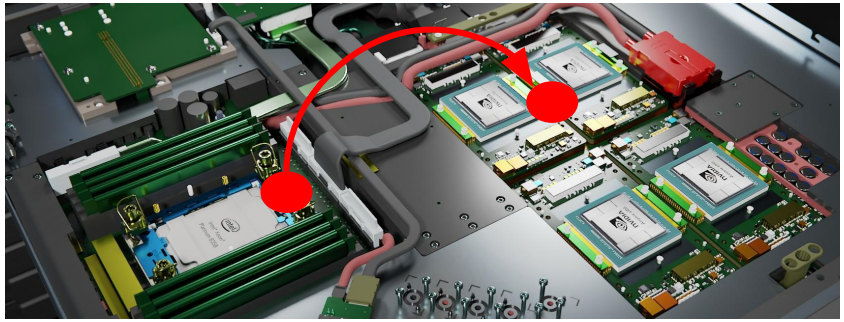


- The program starts on the CPU
- Many ways to offload kernels to GPUs (OpenACC, CUDA, CUDAlibraries…)
- There is always a latency between the launch of the kernel and its execution on the GPU (kernel launch time)

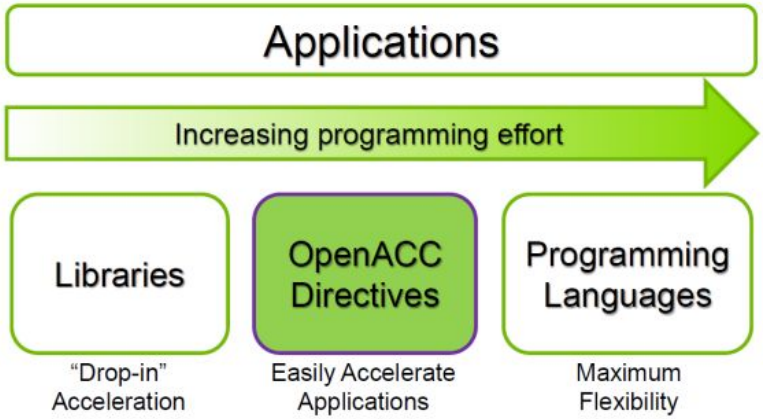


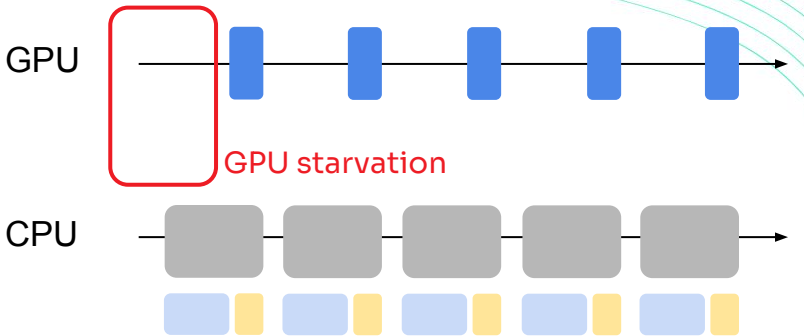


Kernel launch time

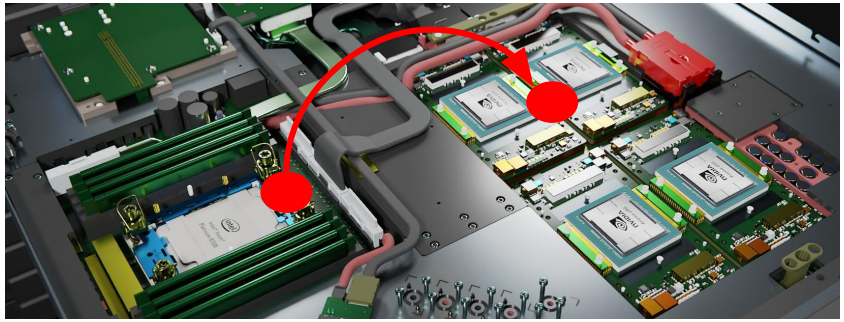


- The program starts on the CPU
- Many ways to offload kernels to GPUs (OpenACC, CUDA, CUDAlibraries…)
- There is always a latency between the launch of the kernel and its execution on the GPU (kernel launch time)

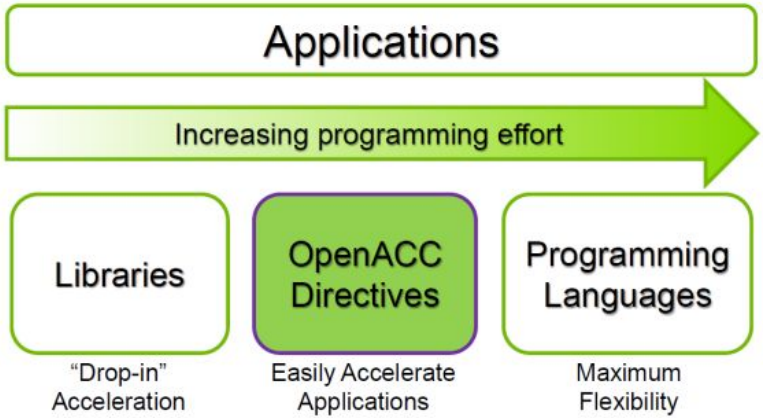




Kernel launch time



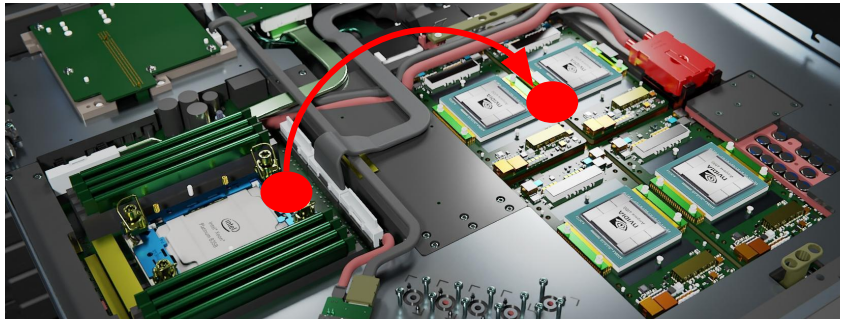
- The program starts on the CPU
- Many ways to offload kernels to GPUs (OpenACC, CUDA, CUDAlibraries…)
- There is always a latency between the launch of the kernel and its execution on the GPU (kernel launch time)



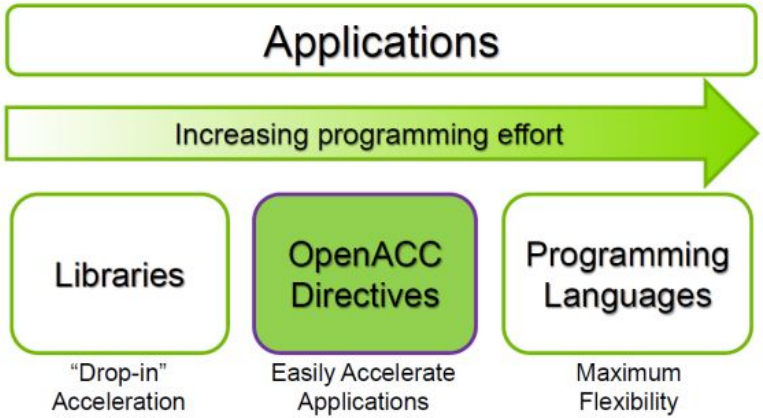
KERNEL LAUNCH ADDS OVERHEAD

EXPOSE AS MUCH PARALLELISM AS POSSIBLE

Kernel launch time



- The program starts on the CPU
- Many ways to offload kernels to GPUs (OpenACC, CUDA, CUDAlibraries...)
- There is always a latency between the launch of the kernel and its execution on the GPU (kernel launch time)



EPICURE

Unlocking European-level HPC Support

NSight Systems
Tracing GPUs and CPUs

Why NSight Systems

🔭Holistic view
Visualize CPU and GPU interconnected in one integrated timeline

🎯Nested bottlenecks
Identify nested bottlenecks as memory stalls, latencies and synchronizations

🌐Distributed computing
Native support to distributed applications on multi-GPU clusters

Why NSight Systems

🔭Holistic view
Visualize CPU and GPU interconnected in one integrated timeline

🎯Nested bottlenecks
Identify nested bottlenecks as memory stalls, latencies and synchronizations

🌐Distributed computing
Native support to distributed applications on multi-GPU clusters

- CUDA APIs
- OpenACC regions
- MPI & OpenMP instrumentation
- GPU kernel instrumentation
- CPU sampling
- Network metrics

Why NSight Systems

🔭Holistic view
Visualize CPU and GPU interconnected in one integrated timeline

🎯Nested bottlenecks
Identify nested bottlenecks as memory stalls, latencies and synchronizations

🌐Distributed computing
Native support to distributed applications on multi-GPU clusters

SAMPLING

INSTRUMENTATION

TRACES

SUMMARIES

CUDA APIs

OpenACC regions

MPI & OpenMP instrumentation

GPU kernel instrumentation

CPU sampling

Network metrics

Why NSight Systems

🔭Holistic view
Visualize CPU and GPU interconnected in one integrated timeline

🎯Nested bottlenecks
Identify nested bottlenecks as memory stalls, latencies and synchronizations

🌐Distributed computing
Native support to distributed applications on multi-GPU clusters

SAMPLING

INSTRUMENTATION

TRACES

SUMMARIES

CUDA APIs

OpenACC regions

MPI & OpenMP instrumentation

GPU kernel instrumentation

CPU sampling

Network metrics

Available via the NVIDIA hpc-sdk compiler suite

NVIDIA Profiling and Tracing Tool

nsys [command_switch][optional command_switch_options][application] [optional application_options]

Command switches: `nsys --help`

```
The most commonly used nsys commands are:
      profile       Run an application and capture its profile into a nsys-rep file.
      launch        Launch an application ready to be profiled.
      start         Start a profiling session.
      stop          Stop a profiling session and capture its profile into a nsys-rep file.
      cancel        Cancel a profiling session and discard any collected data.
      service       Launch the Nsight Systems data service.
      stats         Generate statistics from an existing nsys-rep or SQLite file.
      status        Provide current status of CLI or the collection environment.
      shutdown      Disconnect launched processes from the profiler and shutdown the profiler.
      sessions list List active sessions.
      export        Export nsys-rep file into another format.
      analyze       Identify optimization opportunities in a nsys-rep or SQLITE file.
      recipe        Run a recipe for multi-node analysis.
      nvprof        Translate nvprof switches to nsys switches and execute collection.

Use 'nsys --help <command>' for more information about a specific command.
```

NVIDIA Profiling and Tracing Tool

`nsys [command_switch][optional command_switch_options][application] [optional application_options]`

Command switches: `nsys --help`

```
-t, --trace=

   Possible values are 'cuda', 'cuda-hw', 'nvtx', 'cublas', 'cublas-verbose',
   'cuDNN', 'cuDNN-verbose', 'cusolver', 'cusolver-verbose', 'cusparse',
   'cusparse-verbose', 'mpi', 'oshmem', 'ucx', 'osrt', 'cudnn', 'opengl',
   'opengl-annotations', 'openacc', 'openmp', 'nvvideo', 'vulkan',
   'vulkan-annotations', 'python-gil', 'gds'(experimental) or 'none'.
   Select the API(s) to trace. Multiple APIs can be selected, separated by commas only
   (no spaces).
   If '<api>-annotations' is selected, the corresponding API will also be traced.
   If 'none' is selected, no APIs are traced.
   Default is 'cuda,nvtx,osrt,opengl'. Application scope.
```

NVIDIA Profiling and Tracing Tool

```
nsys [command_switch][optional command_switch_options][application] [optional application_options]
```

Command switches: `nsys --help`

- `--trace`
- `--backtrace`
- `--cuda-memory-usage`
- `--delay, --duration`
- `--nic-metrics`
- `--python-sampling`

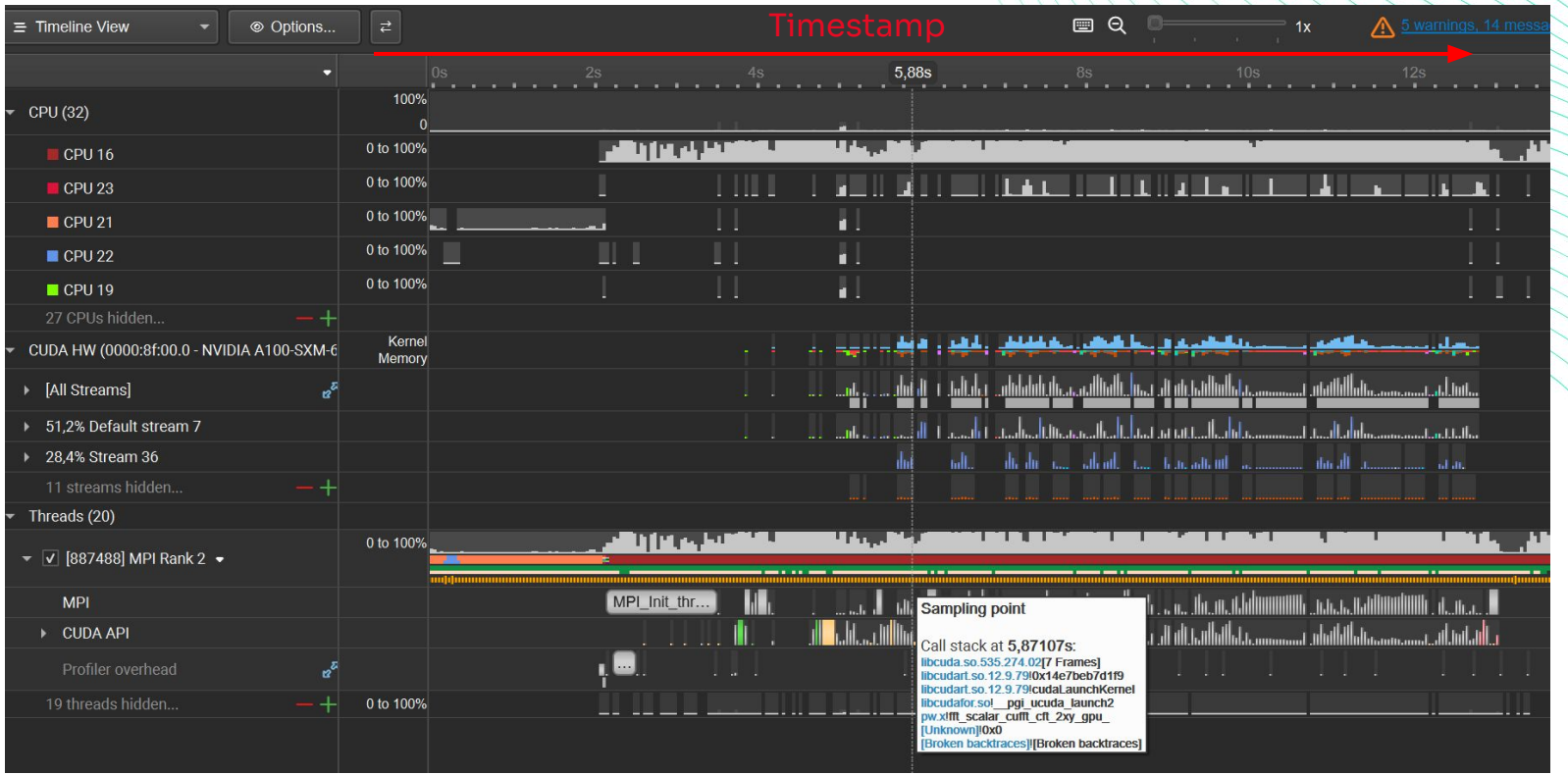
Supports Fortran, C/C++ and also Python codes

Good practices

- Short runs: 5-10 minutes, depending on
  - the number of events observed
  - the number of processes involved
- Focus on a subset of events relevant for your use case
- Apply filters to focus the measurement on the relevant part

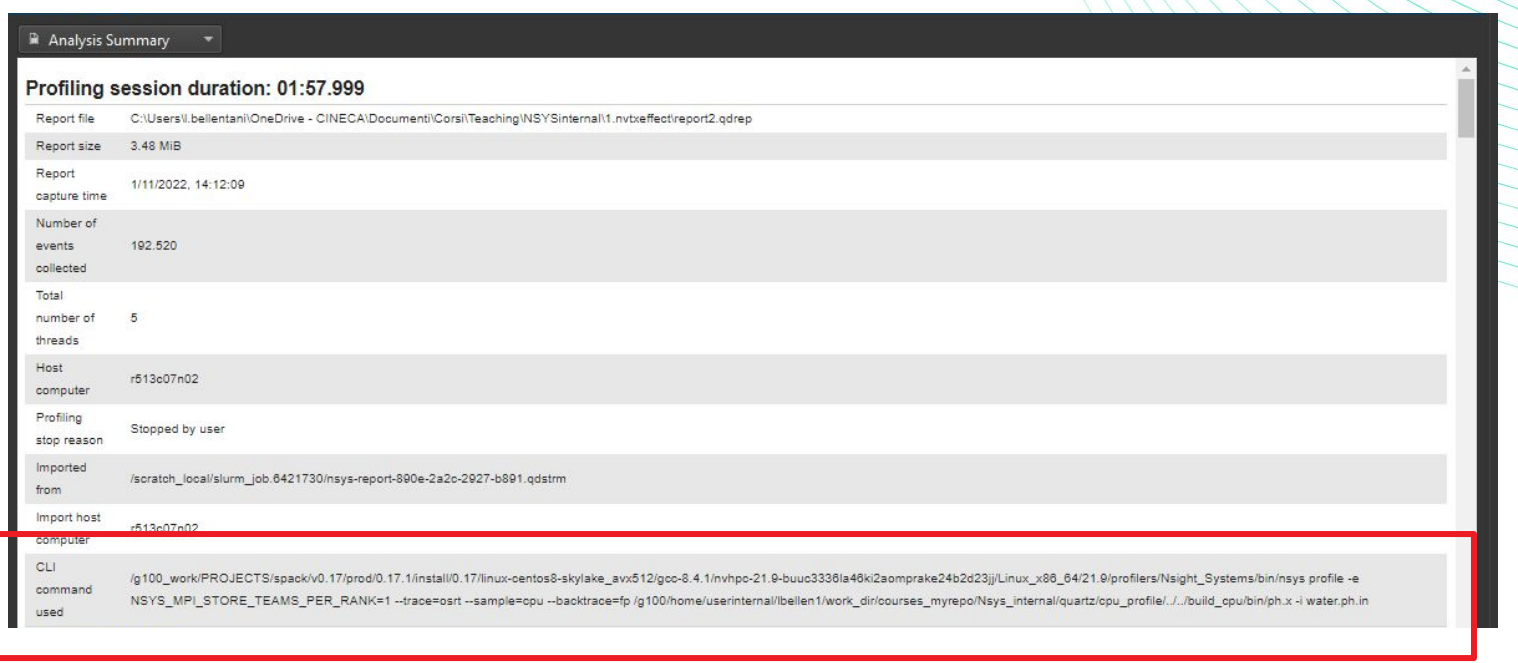
The demos will use simple toy codes offloaded with OpenACC in Fortran, but the same analysis applies to any programming model in C/C++ and also Python

Timeline view



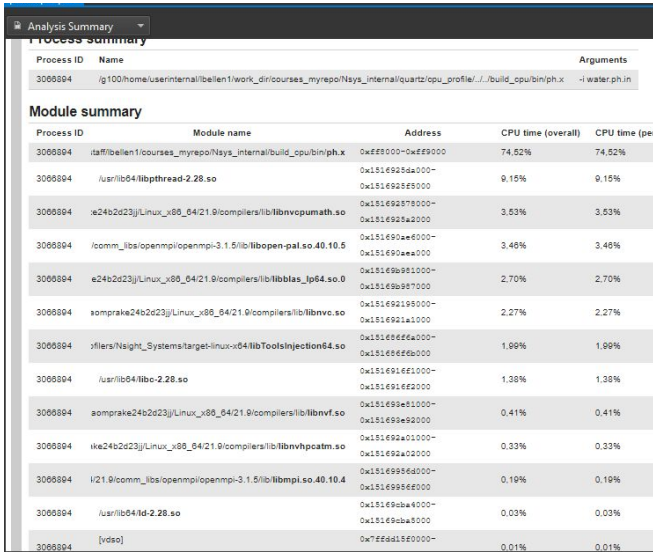
event location

Analysis Summary

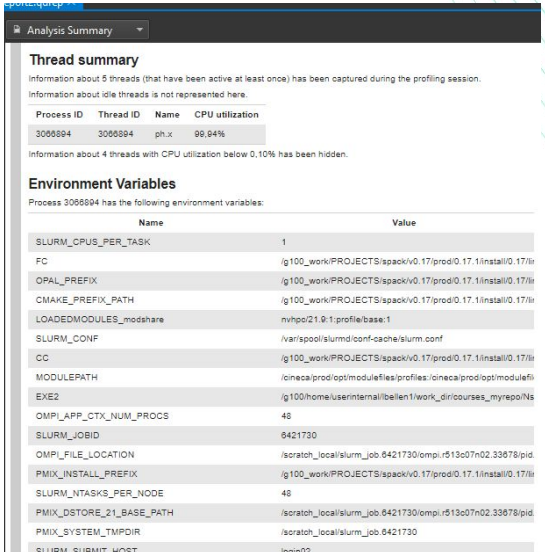


recover the command line

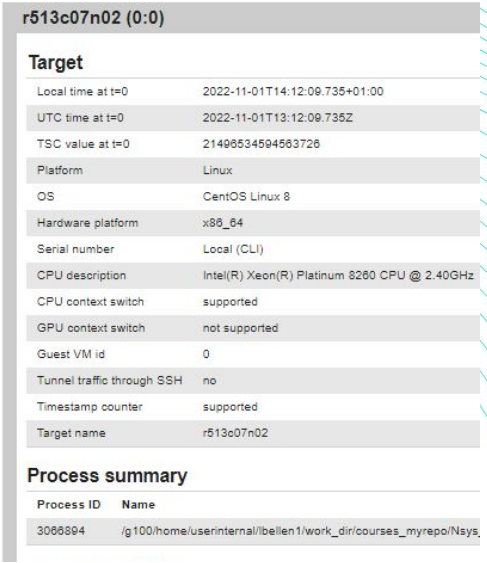
Analysis Summary



time in different modules

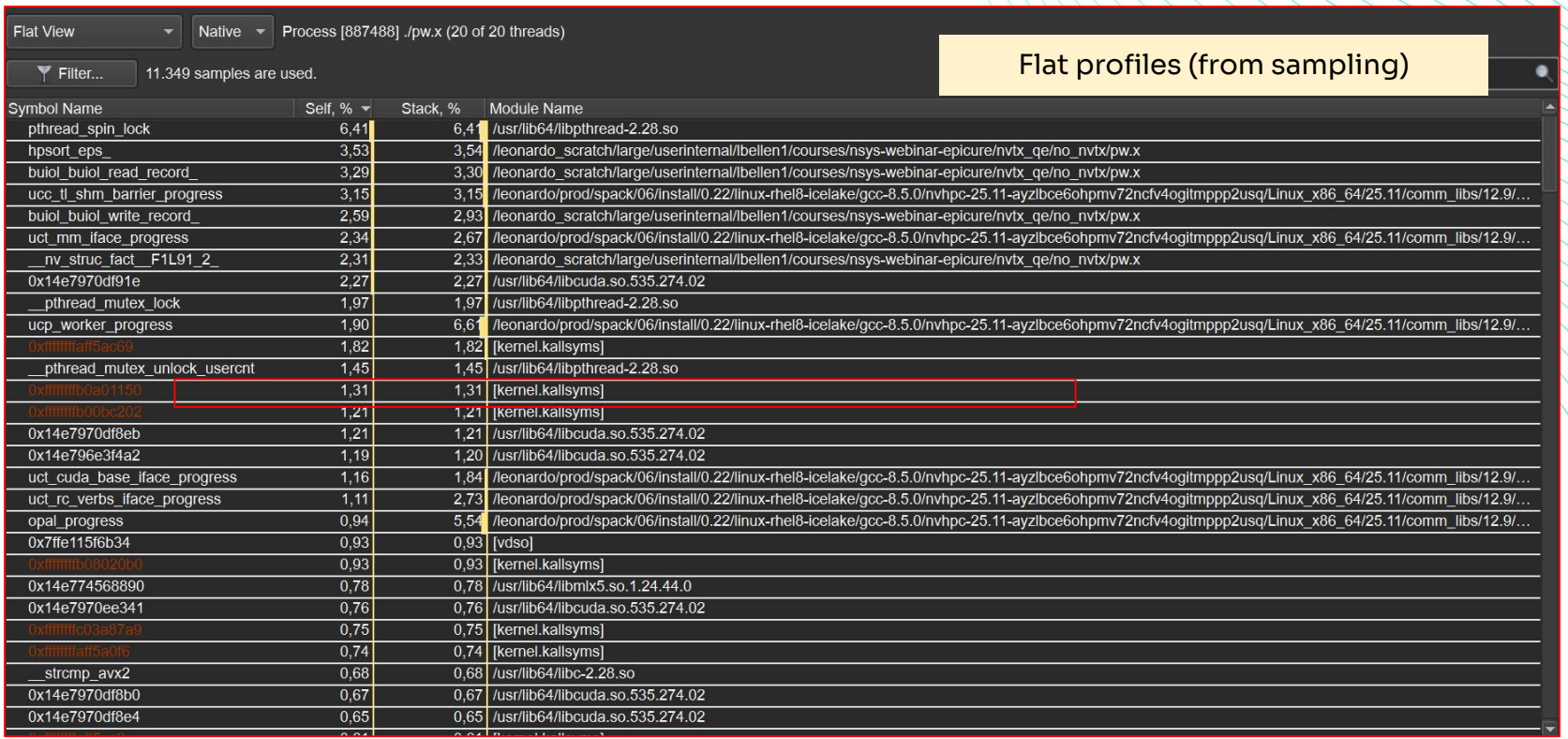


env vars at runtime



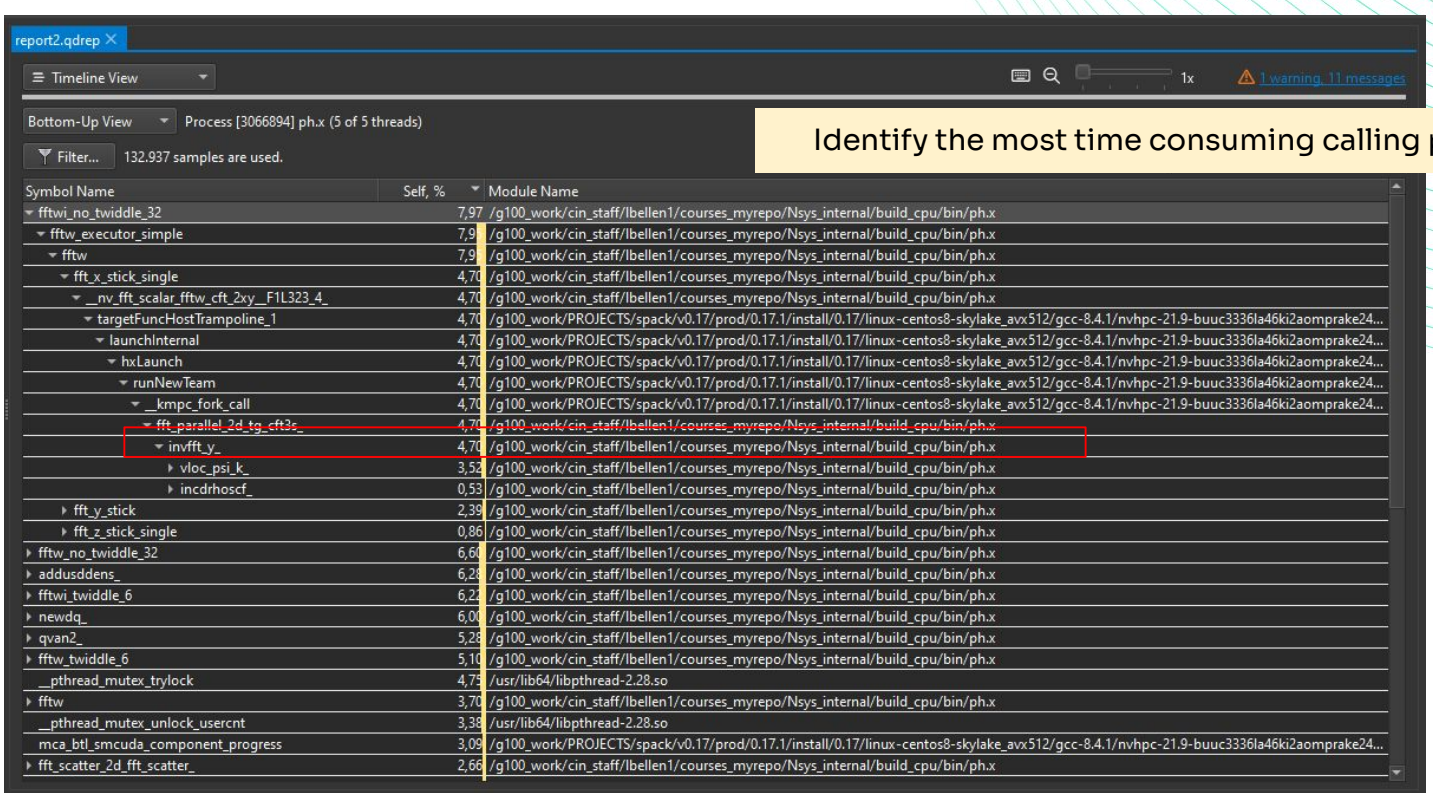
info on underlying hardware

Flat view



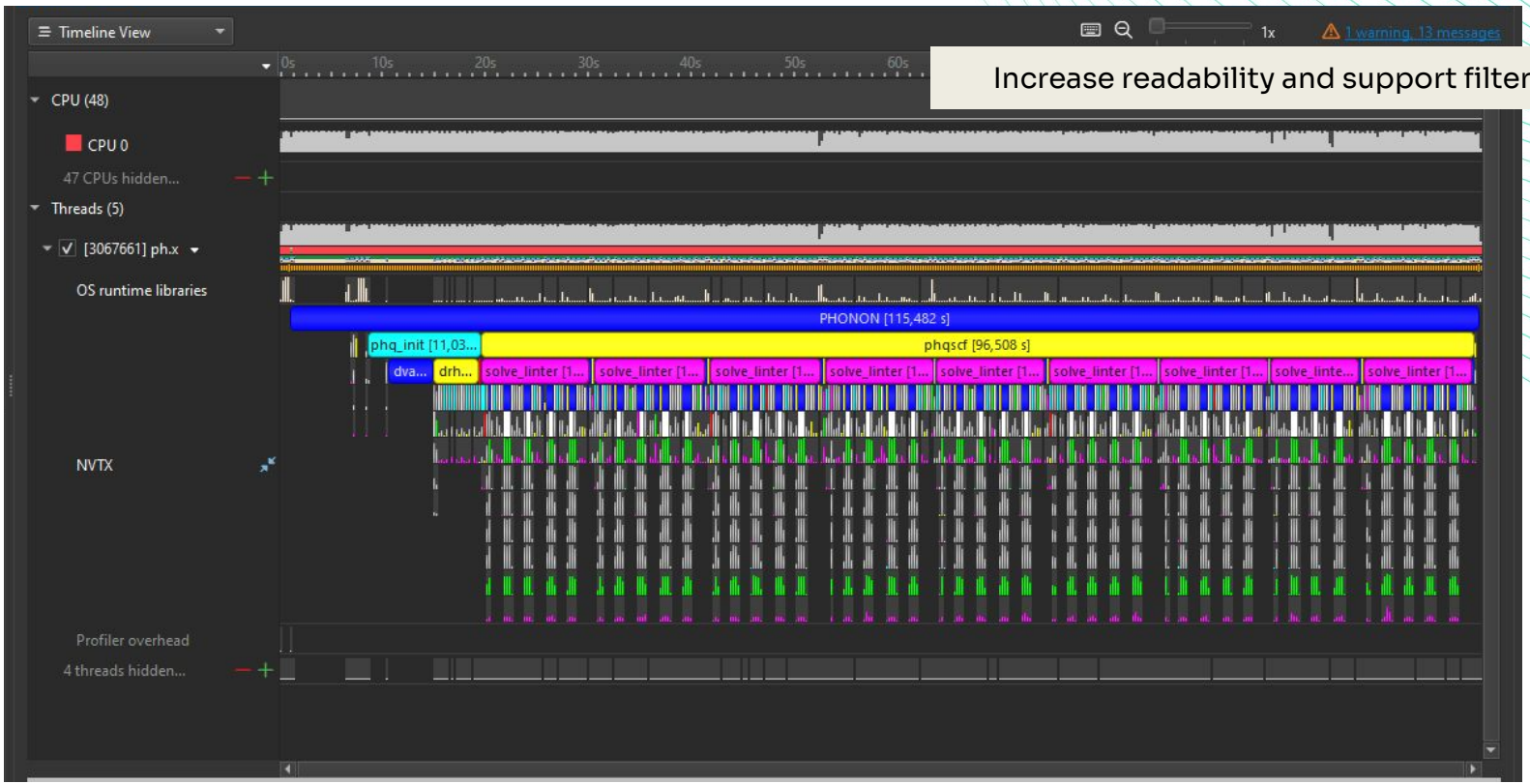
Symbol Name	Self, %	Stack, %	Module Name
pthread_spin_lock	6,41	6,41	/usr/lib64/libpthread-2.28.so
hpsort_eps_	3,53	3,54	/leonardo_scratch/large/userinternal/lbellen1/courses/nsys-webinar-epicure/nvtx_qe/no_nvtx/pw.x
buiol_buiol_read_record_	3,29	3,30	/leonardo_scratch/large/userinternal/lbellen1/courses/nsys-webinar-epicure/nvtx_qe/no_nvtx/pw.x
ucc_tl_shm_barrier_progress	3,15	3,15	/leonardo/prod/spack/06/install/0.22/linux-rhel8-icelake/gcc-8.5.0/nvhpc-25.11-ayzlbce6ohpmv72ncfv4ogitmppp2usq/Linux_x86_64/25.11/comm_libs/12.9/…
buiol_buiol_write_record_	2,59	2,93	/leonardo_scratch/large/userinternal/lbellen1/courses/nsys-webinar-epicure/nvtx_qe/no_nvtx/pw.x
uct_mm_iface_progress	2,34	2,67	/leonardo/prod/spack/06/install/0.22/linux-rhel8-icelake/gcc-8.5.0/nvhpc-25.11-ayzlbce6ohpmv72ncfv4ogitmppp2usq/Linux_x86_64/25.11/comm_libs/12.9/…
__nv_struc_fact__F1L91_2_	2,31	2,33	/leonardo_scratch/large/userinternal/lbellen1/courses/nsys-webinar-epicure/nvtx_qe/no_nvtx/pw.x
0x14e7970df91e	2,27	2,27	/usr/lib64/libcuda.so.535.274.02
__pthread_mutex_lock	1,97	1,97	/usr/lib64/libpthread-2.28.so
ucp_worker_progress	1,90	6,61	/leonardo/prod/spack/06/install/0.22/linux-rhel8-icelake/gcc-8.5.0/nvhpc-25.11-ayzlbce6ohpmv72ncfv4ogitmppp2usq/Linux_x86_64/25.11/comm_libs/12.9/…
0xffffffffaff5ac69	1,82	1,82	[kernel.kallsyms]
__pthread_mutex_unlock_usercnt	1,45	1,45	/usr/lib64/libpthread-2.28.so
0xffffffffb0a01150	1,31	1,31	[kernel.kallsyms]
0xffffffffb00bc202	1,21	1,21	[kernel.kallsyms]
0x14e7970df8eb	1,21	1,21	/usr/lib64/libcuda.so.535.274.02
0x14e796e3f4a2	1,19	1,20	/usr/lib64/libcuda.so.535.274.02
uct_cuda_base_iface_progress	1,16	1,84	/leonardo/prod/spack/06/install/0.22/linux-rhel8-icelake/gcc-8.5.0/nvhpc-25.11-ayzlbce6ohpmv72ncfv4ogitmppp2usq/Linux_x86_64/25.11/comm_libs/12.9/…
uct_rc_verbs_iface_progress	1,11	2,73	/leonardo/prod/spack/06/install/0.22/linux-rhel8-icelake/gcc-8.5.0/nvhpc-25.11-ayzlbce6ohpmv72ncfv4ogitmppp2usq/Linux_x86_64/25.11/comm_libs/12.9/…
opal_progress	0,94	5,54	/leonardo/prod/spack/06/install/0.22/linux-rhel8-icelake/gcc-8.5.0/nvhpc-25.11-ayzlbce6ohpmv72ncfv4ogitmppp2usq/Linux_x86_64/25.11/comm_libs/12.9/…
0x7ffe115f6b34	0,93	0,93	[vdso]
0xffffffffb08020b0	0,93	0,93	[kernel.kallsyms]
0x14e774568890	0,78	0,78	/usr/lib64/libmlx5.so.1.24.44.0
0x14e7970ee341	0,76	0,76	/usr/lib64/libcuda.so.535.274.02
0xffffffffc03a87a9	0,75	0,75	[kernel.kallsyms]
0xffffffffaff5a0f6	0,74	0,74	[kernel.kallsyms]
__strcmp_avx2	0,68	0,68	/usr/lib64/libc-2.28.so
0x14e7970df8b0	0,67	0,67	/usr/lib64/libcuda.so.535.274.02
0x14e7970df8e4	0,65	0,65	/usr/lib64/libcuda.so.535.274.02

Top-Down & Bottom-Up Profiles



Identify the most time consuming calling path

nvTools Library for Annotations (NVTX)



Increase readability and support filtering

CFD toy code

cfd.f90

```
do iter = 1, numiter

   call jacobistep_acc(psitmp, psi, m, n)

! Compute current error value if required

   if (checkerr .or. iter == numiter) then
      error = deltasq(psitmp, psi, m, n)
      error = sqrt(error)
      error = error / bnorm
   end if

! Quit early if we have reached required tolerance

   if (checkerr) then
      if (error .lt. tolerance) then
         write(*,*) 'CONVERGED iteration ', iter, ': terminating'
         exit
      end if
   end if

! Copy back
   do j=1,m
      do i=1,n
        psi(i,j) = psitmp(i,j)
      end do
   end do

! End iterative Jacobi loop

   if (mod(iter,printfreq) == 0) then

      if (.not. checkerr) then
         write(*,*) 'completed iteration ', iter
      else
         write(*,*) 'completed iteration ', iter, ', error = ', error
      end if

   end if

end do
```

jacobi.f90

```
module jacobi

  implicit none

contains

subroutine jacobistep_acc(psinew, psi, m, n)
  integer :: m, n
  integer :: i, j

  double precision, dimension(0:m+1, 0:n+1) :: psinew, psi

   do j = 1, n
      do i = 1, m
          psinew(i, j) = 0.25d0*(psi(i+1, j) + psi(i-1, j) + &
                                 psi(i, j+1) + psi(i, j-1)     )
      end do
  end do

end subroutine jacobistep_acc

double precision function deltasq(new, old, m, n)

  integer :: m, n, i, j
  double precision, dimension(0:m+1, 0:n+1) :: new, old
  double precision :: deltasq

  integer :: ierr

   deltasq = 0.d0
     do j = 1, n
      do i = 1, m
      deltasq = deltasq + (new(i,j)-old(i,j))**2
     end do
    end do

end function deltasq

end module jacobi
```

nvTools Library for Annotations (NVTX)

NVTX library (nvToolsExt.h) can be used to enhance trace readability:

- C-based API to annotate events and code ranges, to be visualized in the Nsight System timeline
- Limited overhead when the tool is not attached to the application

NVTX RANGES : annotates time span of code regions

C/C++ programs

- add `#include nvToolsExt.h`
- include the library at compile time
- range starters
  - `ASCII   LABEL : nvtxRangePush`**`A`**
  - `UNICODE LABEL : nvtxRangePush`**`W`**
  - `EVENT         : nvtxRangePush`**`Ex`**
- range stopper
  - `nvtxRangePop`

```
nvtxEventAttributes_t eventAttrib = {0};

//Declare version and size
eventAttrib.version = NVTX_VERSION;
eventAttrib.size = NVTX_EVENT_ATTRIB_STRUCT_SIZE;

// Message type and message
// // ASCII
eventAttrib.messageType = NVTX_MESSAGE_TYPE_ASCII;
eventAttrib.message.ascii = __FUNCTION__ ":ascii";
// //UNICODE


// Color type and color
eventAttrib.colorType = NVTX_COLOR_ARGB;
eventAttrib.color = COLOR_YELLOW;
```

nvTools Library for Annotations (NVTX)

NVTX library (nvToolsExt.h) can be used to enhance trace readability:

- C-based API to annotate events and code ranges, to be visualized in the Nsight System timeline
- Limited overhead when the tool is not attached to the application

NVTX RANGES : annotates time span of code regions

C/C++ programs

- add `#include nvToolsExt.h`
- include the library at compile time
- range starters
  - `ASCII   LABEL : nvtxRangePush`**`A`**
  - `UNICODE LABEL : nvtxRangePush`**`W`**
  - `EVENT         : nvtxRangePush`**`Ex`**
- range stopper
  - `nvtxRangePop`

```
#include “nvToolsExt.h”

// for message-only events
nvtxRangePushA("range_name");
[... code here ...]
nvtxRangePop();

nvtxRangePushW("range_name");
[... code here ...]
nvtxRangePop();

// for structured events
nvtxRangePushEx(&eventAttrib);
[... code here ...]
nvtxRangePop();
```

nvTools Library for Annotations (NVTX)

NVTX library (nvToolsExt.h) can be used to enhance trace readability:

- C-based API to annotate events and code ranges, to be visualized in the Nsight System timeline
- Limited overhead when the tool is not attached to the application

NVTX RANGES : annotates time span of code regions

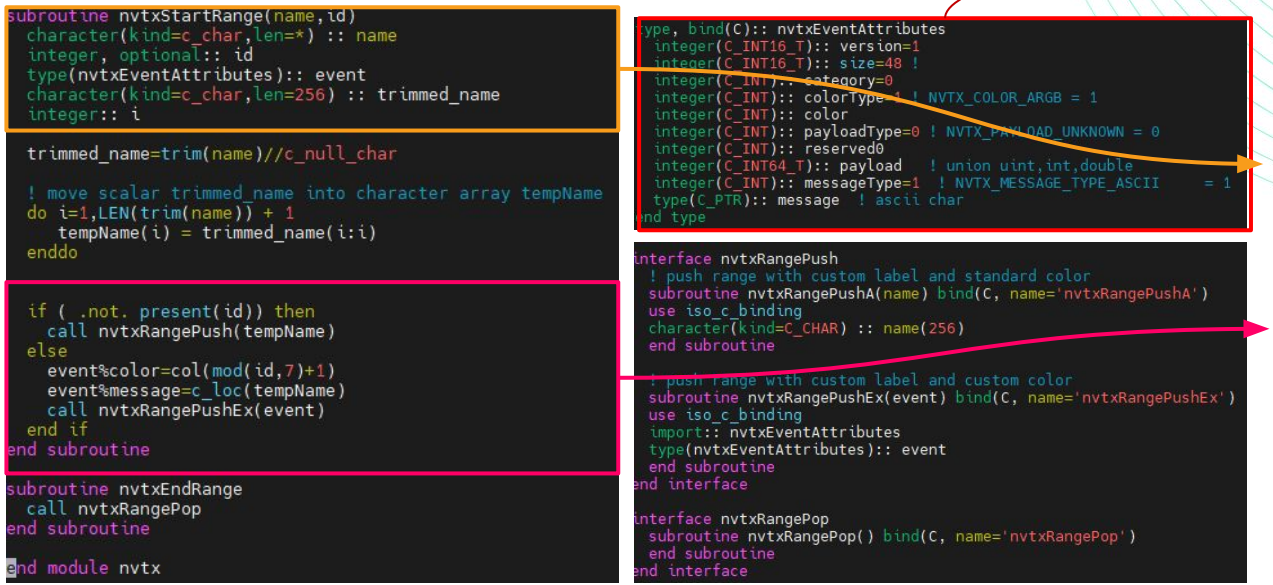
Fortran programs

- add `use nvtx`
- compile with `-cudalib=nvtx`
- range starters
  - `nvtxStartRange("name",[id])`
    - `id → colored (*Ex)`
    - `no id → uncolored (*A)`
- range stopper
  - `nvtxEndRange`

```
use nvtx

// for message-only events
call nvtxStartRange("range_name")
[... code here ...]
call nvtxEndRange()

// for structured events
call nvtxStartRange("range_name",0);
[... code here ...]
call nvtxEndRange;
```

nvTools Library for Annotations (NVTX)

Fortran module Wrapper in nvhpcsdk compiler suite



```
subroutine nvtxStartRange(name,id)
  character(kind=c_char,len=*) :: name
  integer, optional:: id
  type(nvtxEventAttributes):: event
  character(kind=c_char,len=256) :: trimmed_name
  integer:: i

  trimmed_name=trim(name)//c_null_char

  ! move scalar trimmed_name into character array tempName
  do i=1,LEN(trim(name)) + 1
     tempName(i) = trimmed_name(i:i)
  enddo

  if ( .not. present(id)) then
    call nvtxRangePush(tempName)
  else
    event%color=col(mod(id,7)+1)
    event%message=c_loc(tempName)
    call nvtxRangePushEx(event)
  end if
end subroutine

subroutine nvtxEndRange
  call nvtxRangePop
end subroutine

end module nvtx
```

```
type, bind(C):: nvtxEventAttributes
  integer(C_INT16_T):: version=1
  integer(C_INT16_T):: size=48 !
  integer(C_INT):: category=0
  integer(C_INT):: colorType=1 ! NVTX_COLOR_ARGB = 1
  integer(C_INT):: color
  integer(C_INT):: payloadType=0 ! NVTX_PAYLOAD_UNKNOWN = 0
  integer(C_INT):: reserved0
  integer(C_INT64_T):: payload   ! union uint,int,double
  integer(C_INT):: messageType=1  ! NVTX_MESSAGE_TYPE_ASCII     = 1
  type(C_PTR):: message  ! ascii char
end type
```

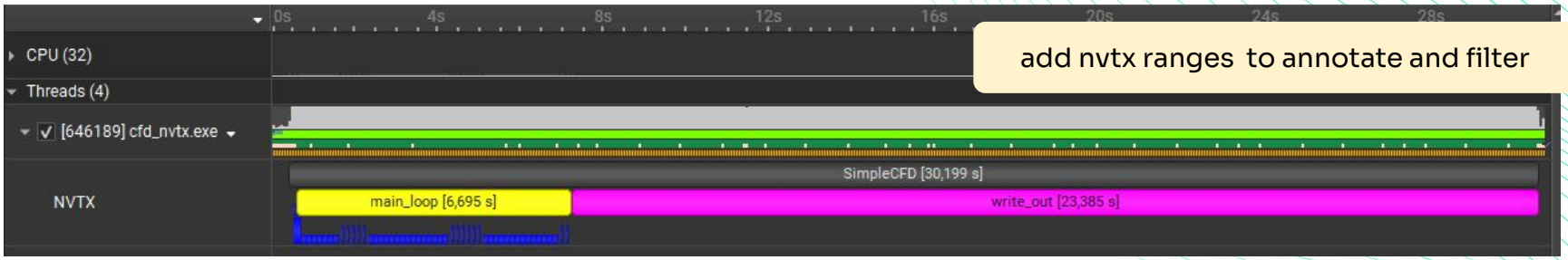
```
interface nvtxRangePush
  ! push range with custom label and standard color
  subroutine nvtxRangePushA(name) bind(C, name='nvtxRangePushA')
  use iso_c_binding
  character(kind=C_CHAR) :: name(256)
  end subroutine

  ! push range with custom label and custom color
  subroutine nvtxRangePushEx(event) bind(C, name='nvtxRangePushEx')
  use iso_c_binding
  import:: nvtxEventAttributes
  type(nvtxEventAttributes):: event
  end subroutine
end interface

interface nvtxRangePop
  subroutine nvtxRangePop() bind(C, name='nvtxRangePop')
  end subroutine
end interface
```

Defines a derived type for the event

Fills color and message attribute according to the presence or not of the optional ID parameter

Resolves the calling routine in *Ex or *A API according to input

A Performance Engineering Workflow



add nvtx ranges to annotate and filter

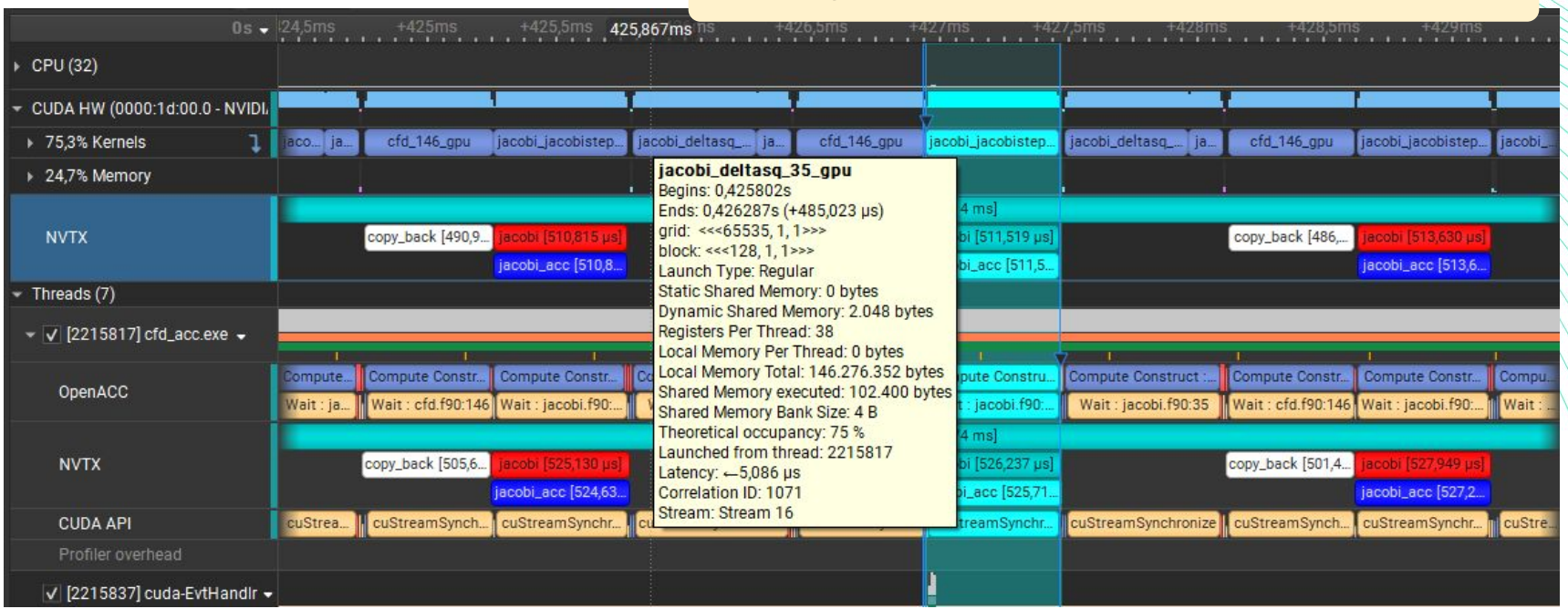
```
nsys profile --trace=nvtx  --capture-range=nvtx --nvtx-capture='main_loop' --env-var=NSYS_NVTX_PROFILER_REGISTER_ONLY=0
```

A Performance Engineering Workflow

check for expensive data movements and GPU starvation

A Performance Engineering Workflow

analyze how kernels are mapped on GPU hardware



Summaries (nsys stats)

```
nsys stats --help-reports
```

```
The following built-in reports are available:

  cuda_api_gpu_sum[:nvtx-name][:base|:mangled] -- CUDA Summary (API/Kernels/MemOps)
  cuda_api_sum -- CUDA API Summary
  cuda_api_trace -- CUDA API Trace
  cuda_gpu_kern_gb_sum[:nvtx-name][:base|:mangled] -- CUDA GPU Kernel/Grid/Block Summary
  cuda_gpu_kern_sum[:nvtx-name][:base|:mangled] -- CUDA GPU Kernel Summary
  cuda_gpu_mem_size_sum -- CUDA GPU MemOps Summary (by Size)
  cuda_gpu_mem_time_sum -- CUDA GPU MemOps Summary (by Time)
  cuda_gpu_sum[:nvtx-name][:base|:mangled] -- CUDA GPU Summary (Kernels/MemOps)
  cuda_gpu_trace[:nvtx-name][:base|:mangled] -- CUDA GPU Trace
  cuda_kern_exec_sum[:nvtx-name][:base|:mangled] -- CUDA Kernel Launch & Exec Time Summary
  cuda_kern_exec_trace[:nvtx-name][:base|:mangled] -- CUDA Kernel Launch & Exec Time Trace
  dx11_pix_sum -- DX11 PIX Range Summary
  dx12_gpu_marker_sum -- DX12 GPU Command List PIX Ranges Summary
  dx12_pix_sum -- DX12 PIX Range Summary
  mpi_event_sum -- MPI Event Summary
  mpi_event_trace -- MPI Event Trace
  mpi_msg_size_sum -- MPI Message Size Summary
  network_congestion[:ticks_threshold=<ticks_per_ms>] -- Network Devices Congestion
  nvtx_gpu_proj_sum -- NVTX GPU Projection Summary
  nvtx_gpu_proj_trace -- NVTX GPU Projection Trace
  nvtx_kern_sum[:base|:mangled] -- NVTX Range Kernel Summary
  nvtx_pushpop_sum -- NVTX Push/Pop Range Summary
  nvtx_pushpop_trace -- NVTX Push/Pop Range Trace
  nvtx_startend_sum -- NVTX Start/End Range Summary
  nvtx_sum -- NVTX Range Summary
  nvvideo_api_sum -- NvVideo API Summary
  openacc_sum -- OpenACC Summary
  opengl_khr_gpu_range_sum -- OpenGL KHR_debug GPU Range Summary
  opengl_khr_range_sum -- OpenGL KHR_debug Range Summary
  openmp_sum -- OpenMP Summary
```

Summaries (nsys stats)

```
nsys stats -r <summary-name> report.nsys-rep
```

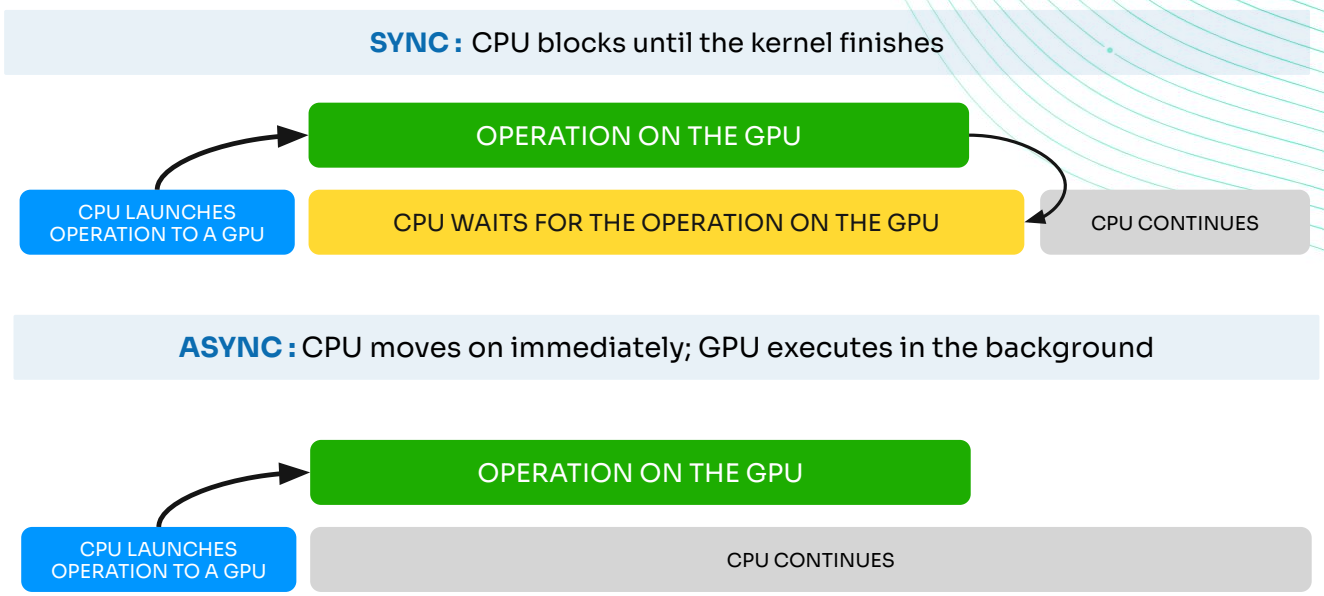
```
nsys stats -r openaccsum report_acc_data1.nsys-rep
Generating SQLite file report_acc_data1.sqlite from report_acc_data1.nsys-rep
Exporting 63118 events: [=================================================100%]
Processing [report_acc_data1.sqlite] with
[/leonardo/prod/spack/03/install/0.19/linux-rhel8-icelake/gcc-8.5.0/nvhpc-23.1-x5lw6edfmfuot2ipna3wseallzl4oolm/Linux_x86_64/23.1/profilers/Nsight_Systems/host-linux-x64/reports/openaccsum.py]...

** OpenACC Summary (openaccsum):

Time(%)  Total Time (ns)  Num Calls    Avg (ns)      Med (ns)     Min (ns)    Max (ns)    StdDev (ns)               Name
-------  ---------------  ---------  ------------  ------------  ----------  ----------  ------------  ------------------------------
   27.9    2,422,224,252         50  48,444,485.0  45,729,730.5  45,576,626  176,454,147  18,476,396.7  Exit Data@jacobi.f90:14
   26.4    2,292,084,948         50  45,841,699.0  45,729,495.0  45,514,774   47,135,248     366,299.5  Exit Data@cfd.f90:144
   20.5    1,779,791,765         50  35,595,835.3  35,474,408.0  35,253,494   36,959,563     390,115.3  Enter Data@cfd.f90:144
   19.8    1,719,850,039         50  34,397,000.8  33,854,316.0  33,666,671   55,305,149   3,046,996.4  Enter Data@jacobi.f90:14
    1.4      123,216,371         50   2,464,327.4   2,328,897.0   2,309,384    8,864,364     923,773.8  Wait@jacobi.f90:21
    1.3      115,610,293         50   2,312,205.9   2,307,225.0   2,283,833    2,386,689      21,258.0  Wait@cfd.f90:150
    0.7       58,974,491        100     589,744.9     584,813.0     497,833      684,812      88,260.9  Wait@cfd.f90:144
    0.7       58,777,669        100     587,776.7     513,375.0      48,488      683,116     100,939.1  Wait@jacobi.f90:14
    0.3       27,516,377          1  27,516,377.0  27,516,377.0  27,516,377   27,516,377           0.0  Device Init@jacobi.f90:14
    0.3       26,569,915         50     531,398.3     530,716.0     523,898      548,907       3,876.9  Compute Construct@jacobi.f90:14
    0.3       26,165,316         50     523,306.3     522,872.0     519,415      530,353       2,344.2  Compute Construct@cfd.f90:144
    0.1        8,457,818      1,000       8,457.8       7,139.5       5,890       72,950       3,862.6  Enqueue Upload@cfd.f90:144
```

EPICURE

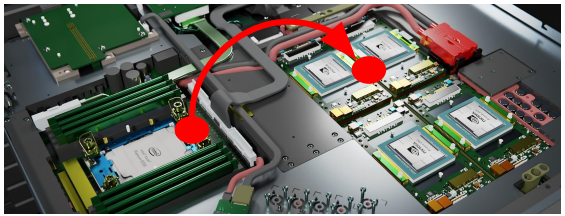
Unlocking European-level HPC Support

Asynchronous programming and multistream

Asynchronous Programming

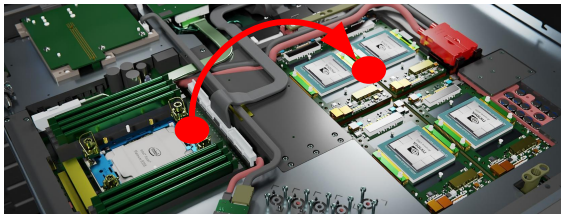


Asynchronous Programming



```
for (i = 0; i<n; i++)
    a[i] = 1

for (i = 0; i<n; i++)
    b[i] = 1

for (i = 0; i<n; i++)
    c[i] = a[i] + b[i]
```

Asynchronous Programming

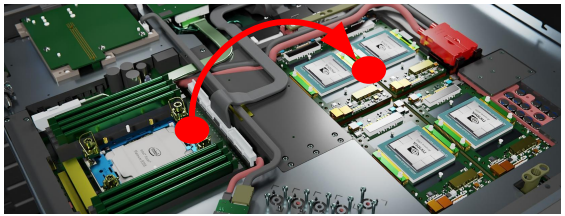


```
for (i = 0; i<n; i++)
    a[i] = 1
```

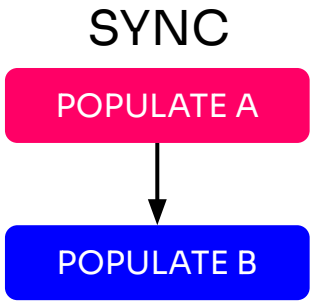
SYNC

POPULATE A

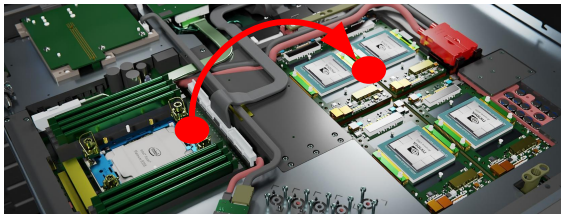
Asynchronous Programming



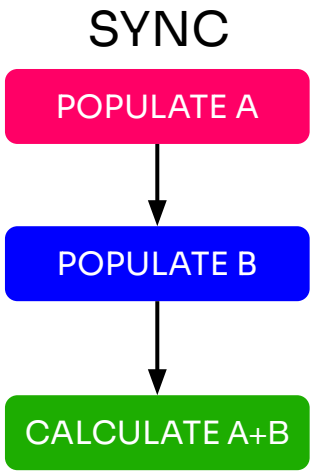
```
for (i = 0; i<n; i++)
    a[i] = 1

for (i = 0; i<n; i++)
    b[i] = 1
```



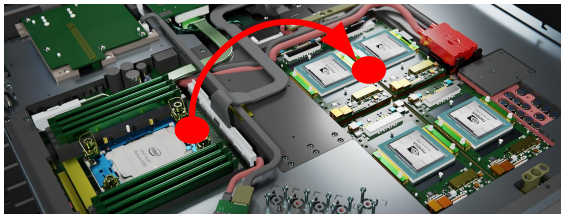
Asynchronous Programming



```
for (i = 0; i<n; i++)
    a[i] = 1

for (i = 0; i<n; i++)
    b[i] = 1

for (i = 0; i<n; i++)
    c[i] = a[i] + b[i]
```

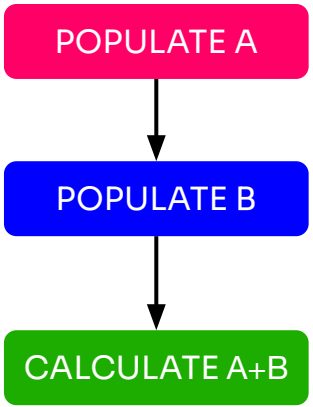


Asynchronous Programming



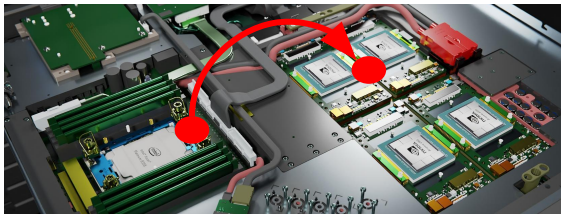
```
for (i = 0; i<n; i++)
    a[i] = 1

for (i = 0; i<n; i++)
    b[i] = 1

for (i = 0; i<n; i++)
    c[i] = a[i] + b[i]
```

SYNC

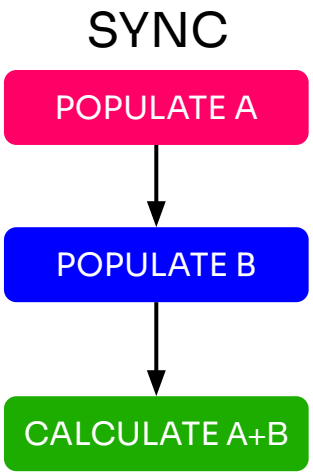


ASYNC

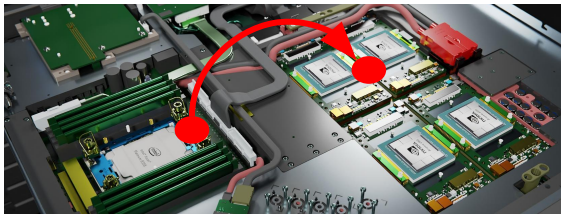
Asynchronous Programming



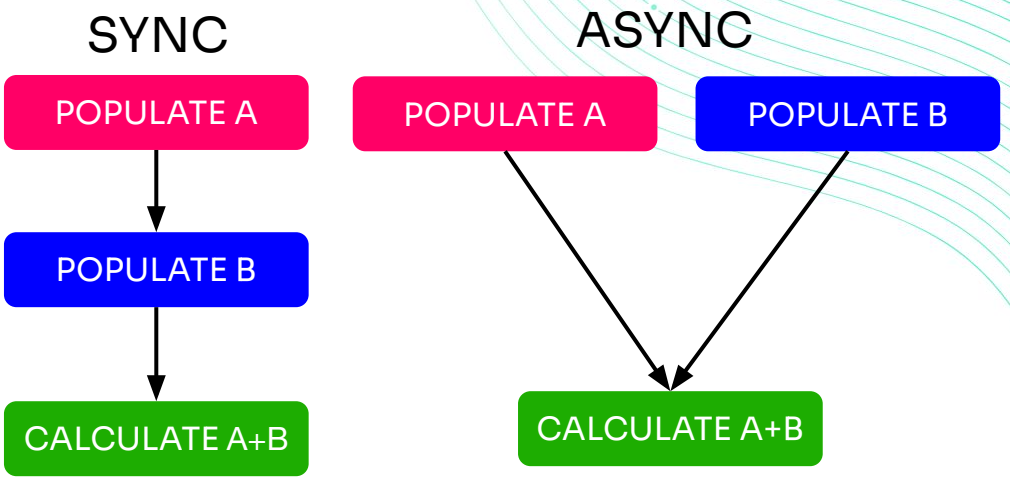
```
for (i = 0; i<n; i++)
    a[i] = 1

for (i = 0; i<n; i++)
    b[i] = 1
```



Asynchronous Programming



```
for (i = 0; i<n; i++)
    a[i] = 1

for (i = 0; i<n; i++)
    b[i] = 1

for (i = 0; i<n; i++)
    c[i] = a[i] + b[i]
```



Asynchronous Programming

queue 3

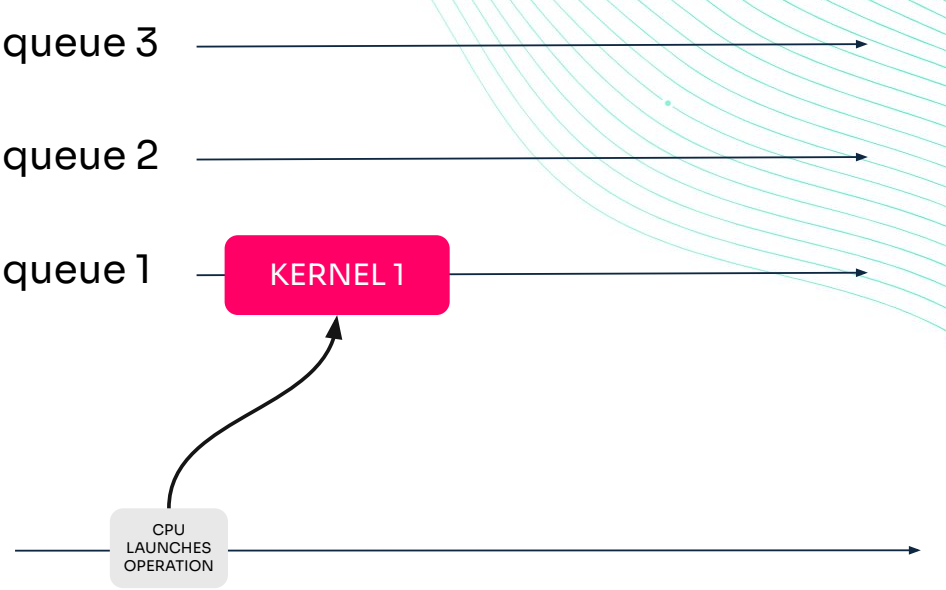
queue 2

queue 1

Asynchronous Programming

```
#pragma acc parallel loop async(1)
for (i = 0; i<n; i++)
 a[i] = 1
```

KERNEL IS COMPUTED ON STREAM 1

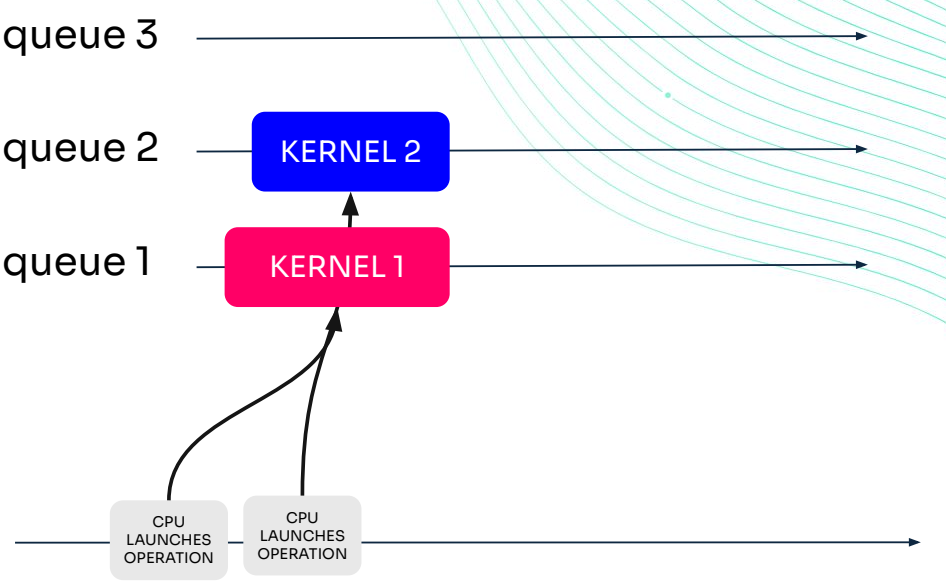


Asynchronous Programming

```
#pragma acc parallel loop async(1)
for (i = 0; i<n; i++)
 a[i] = 1
#pragma acc parallel loop async(2)
for (i = 0; i<n; i++)
 b[i] = 1
```

KERNEL IS COMPUTED ON STREAM 1

KERNEL IS COMPUTED ON STREAM 2



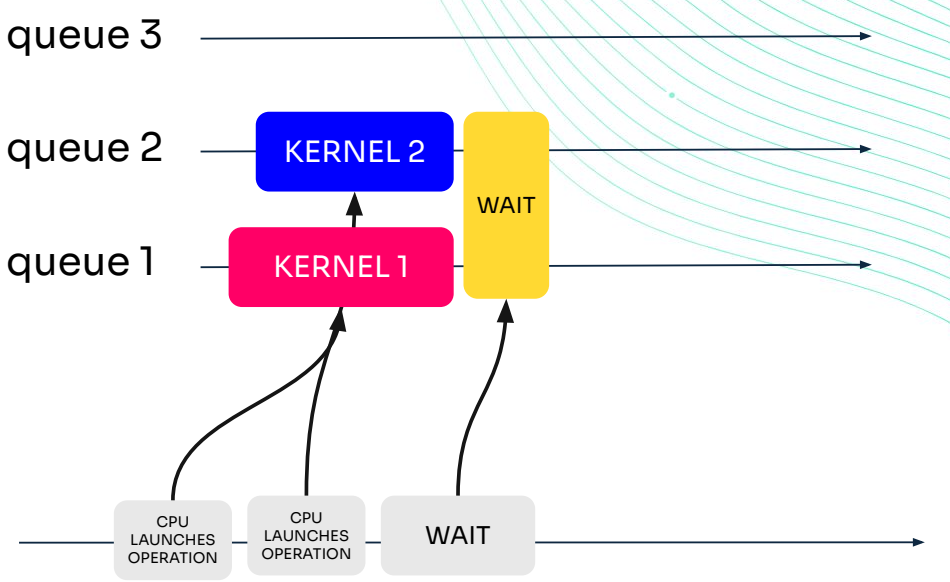
Asynchronous Programming

```
#pragma acc parallel loop async(1)
for (i = 0; i<n; i++)
 a[i] = 1
#pragma acc parallel loop async(2)
for (i = 0; i<n; i++)
 b[i] = 1
#pragma acc wait(1) async(2)
```

KERNEL IS COMPUTED ON STREAM 1

KERNEL IS COMPUTED ON STREAM 2

WAIT FOR STREAM 1 ON STREAM 2



Asynchronous Programming

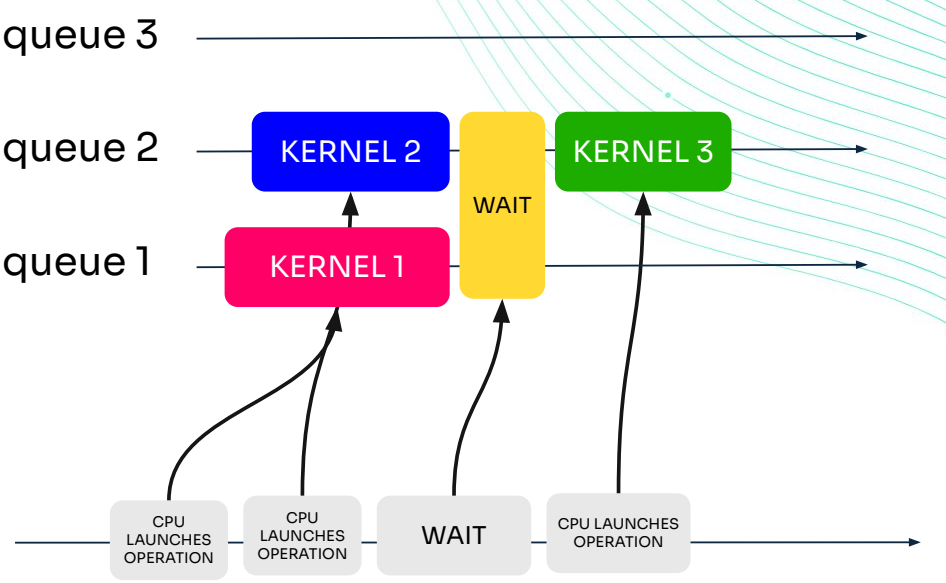
```
#pragma acc parallel loop async(1)
for (i = 0; i<n; i++)
 a[i] = 1
#pragma acc parallel loop async(2)
for (i = 0; i<n; i++)
 b[i] = 1
#pragma acc wait(1) async(2)
#pragma acc loop async(2)
for (i = 0; i<n; i++)
c[i] = a[i] + b[i]
```

KERNEL IS COMPUTED ON STREAM 1

KERNEL IS COMPUTED ON STREAM 2

WAIT FOR STREAM 1 ON STREAM 2

SUM ON STREAM 2



Asynchronous Programming

```
#pragma acc parallel loop async(1)
for (i = 0; i<n; i++)
 a[i] = 1
#pragma acc parallel loop async(2)
for (i = 0; i<n; i++)
 b[i] = 1
#pragma acc wait(1) async(2)
#pragma acc loop async(2)
for (i = 0; i<n; i++)
c[i] = a[i] + b[i]
#pragma acc update self[c[0:N]] async(2)
#pragma acc wait
```

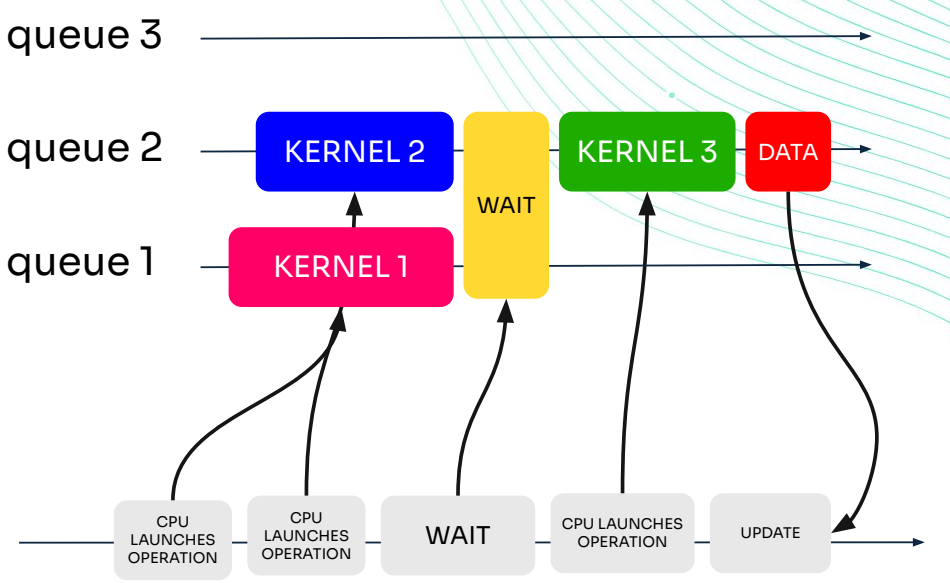
KERNEL IS COMPUTED ON STREAM 1

KERNEL IS COMPUTED ON STREAM 2

WAIT FOR STREAM 1 ON STREAM 2

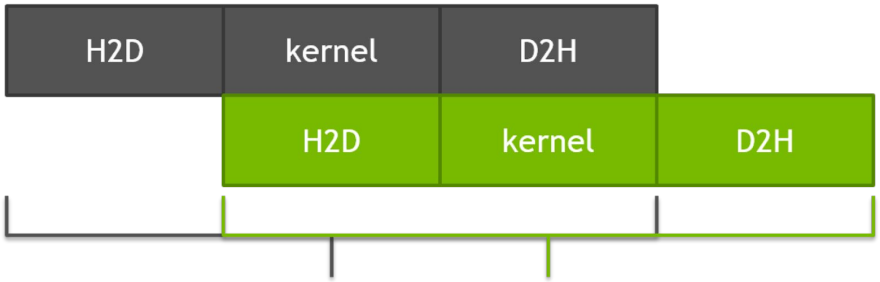
SUM ON STREAM 2

UPDATE FROM STREAM 2 AND SYNC



Asynchronous Programming

Two Independent Operations Serialized



Overlapping Copying and Computation

NOTE: In real applications, your boxes will not be so evenly sized.

Mandelbrot Example

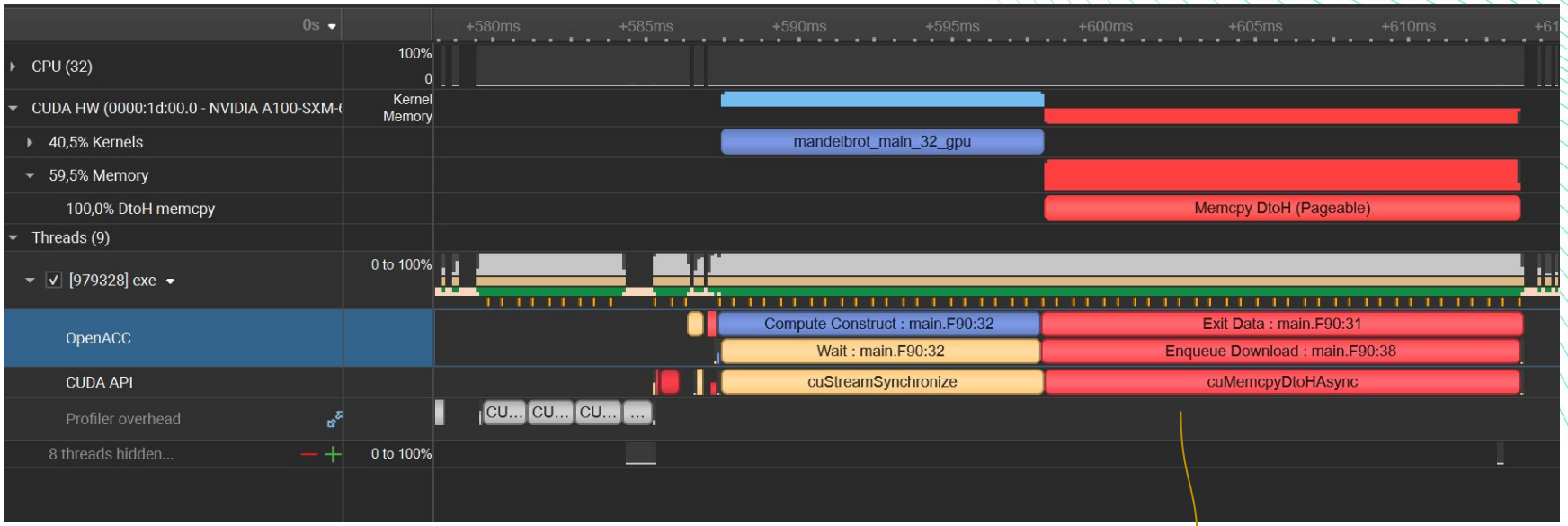
```
!$acc data copyout(image(1:HEIGHT, 1:WIDTH))
!$acc parallel loop
do iy=1,WIDTH
   do ix=1,HEIGHT
      image(ix,iy) = min(max(int(mandelbrot(ix-1,iy-1)),0),MAXCOLORS)
   enddo
  enddo
enddo
!$acc end data
```

- Data movements are manually managed
- Launches a single GPU kernel

```
!$acc routine seq

...

do while(((x*x+y*y).lt.4.0d0).and.(i.lt.MAX_ITERS))
   xtemp=x*x - y*y + x0
   y=2*x*y + y0
   x=xtemp
   i = i+1
enddo
mandelbrot =  dble(MAXCOLORS)*i/MAX_ITERS
```

- Each worker executes the routine sequentially

Mandelbrot Example



hide latency of D2H data movement

Mandelbrot Example

```
!$acc data create(image(1:HEIGHT, 1:WIDTH))
!$acc parallel loop
do iy=1,WIDTH
   do ix=1,HEIGHT
      image(ix,iy) = min(max(int(mandelbrot(ix-1,iy-1)),0),MAXCOLORS)
   enddo
  enddo
enddo
!$acc end data
```

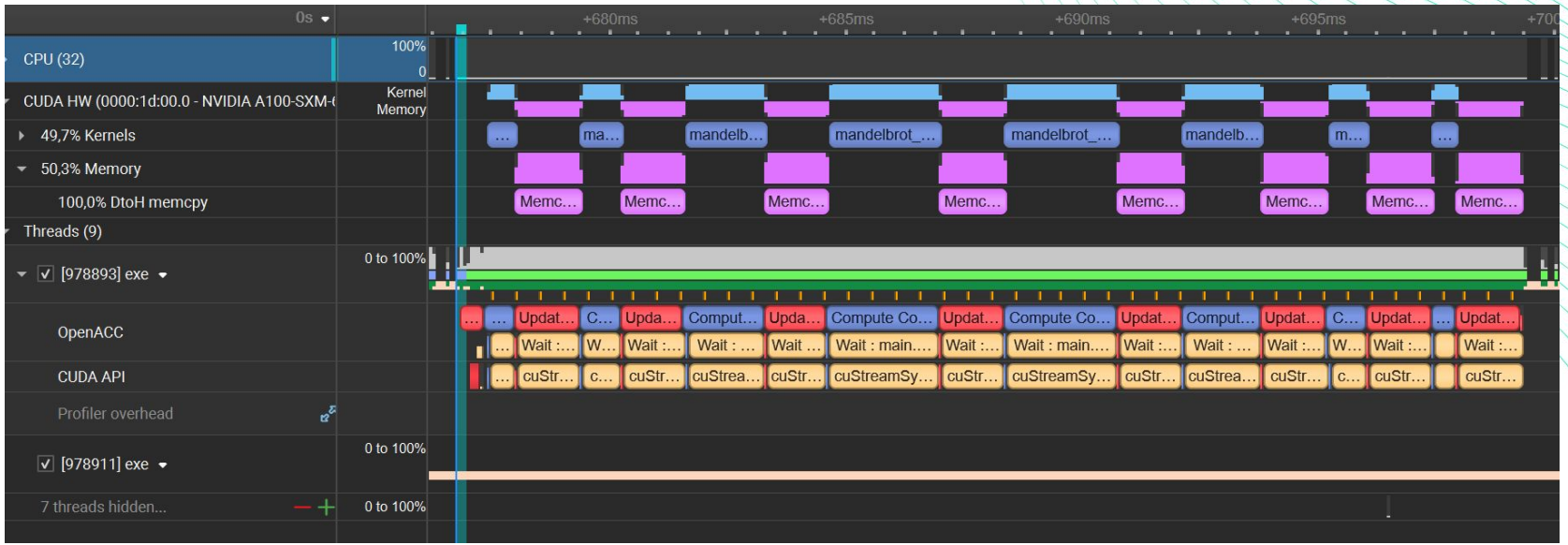
```
!$acc data create(image(1:HEIGHT, 1:WIDTH))
do block=0,(num_blocks-1)
  starty = block  * (WIDTH/NUM_BLOCKS) + 1
  endy   = min(starty + (WIDTH/NUM_BLOCKS), WIDTH)
  !$acc parallel loop
  do iy=starty,endy
    do ix=1,HEIGHT
      image(ix,iy) = min(max(int(mandelbrot(ix-1,iy-1)),0),MAXCOLORS)
    enddo
  enddo
  !$acc update self(image(:,starty:endy))
enddo
!$acc end data
```

- Data movements are manually managed
- Launch a single kernel on the GPU

split kernel computation and data movements in chunks

- Divide one dimension (width) into blocks
- Launch one GPU kernel per block
- Update only the block chunk of the GPU buffer on the host

Mandelbrot Example



Mandelbrot Example

```
!$acc data create(image(1:HEIGHT, 1:WIDTH))
!$acc parallel loop
do iy=1,WIDTH
   do ix=1,HEIGHT
      image(ix,iy) = min(max(int(mandelbrot(ix-1,iy-1)),0),MAXCOLORS)
   enddo
  enddo
enddo
!$acc end data
```

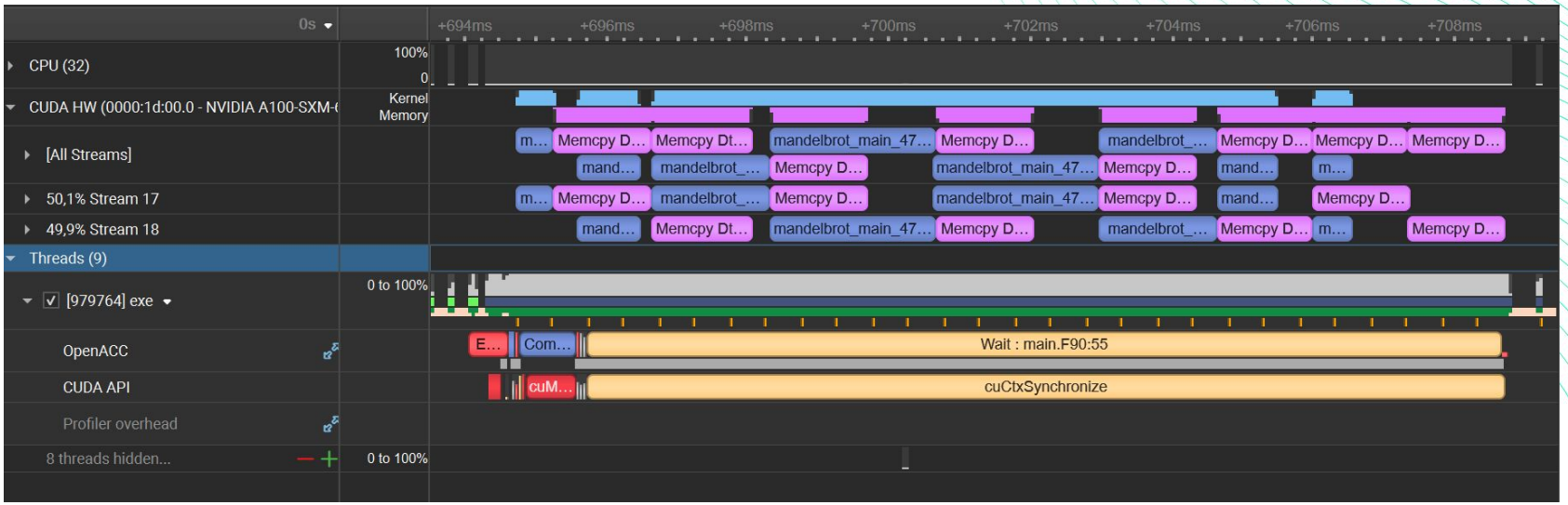
- Data movements are manually managed
- Launch a single kernel on the GPU

different streams based on the parity of the block index

```
!$acc data create(image(1:HEIGHT, 1:WIDTH))
do block=0,(num_blocks-1)
  starty = block  * (WIDTH/NUM_BLOCKS) + 1
  endy   = min(starty + (WIDTH/NUM_BLOCKS), WIDTH)
  !$acc parallel loop async(mod(block,2))
  do iy=starty,endy
    do ix=1,HEIGHT
      image(ix,iy) = min(max(int(mandelbrot(ix-1,iy-1)),0),MAXCOLORS)
    enddo
  enddo
  !$acc update self(image(:,starty:endy)) async(mod(block,2))
enddo
!$acc end data
```

- Divide one dimension (width) into blocks
- Launch one GPU kernel per block **asynchronously** on stream number depending on the block index
- Update **asynchronously** only the block chunk of the GPU buffer from the same stream of the GPU kernel

Mandelbrot Example

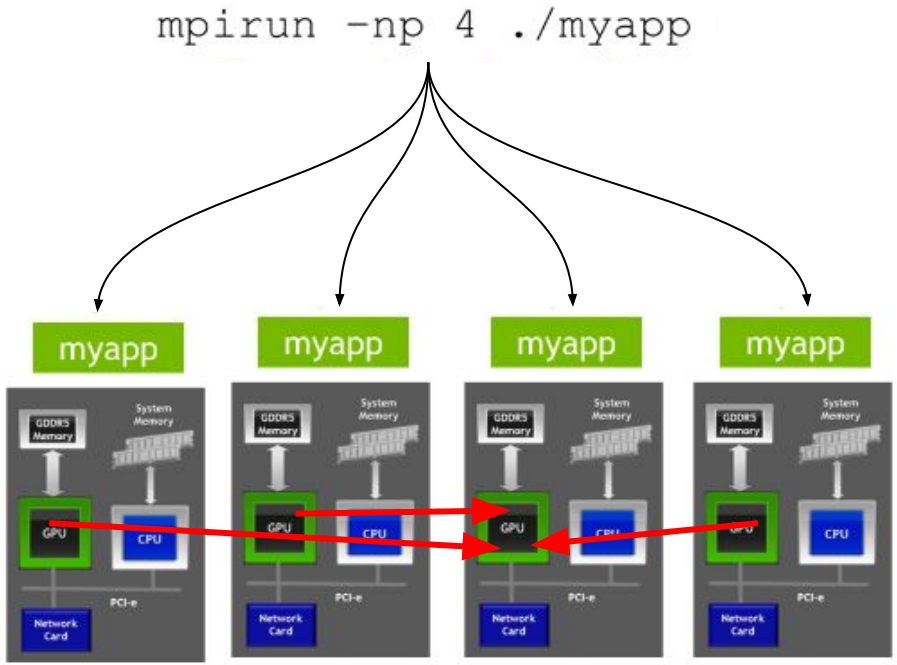


EPICURE

Unlocking European-level HPC Support

Awareness in MPI communications

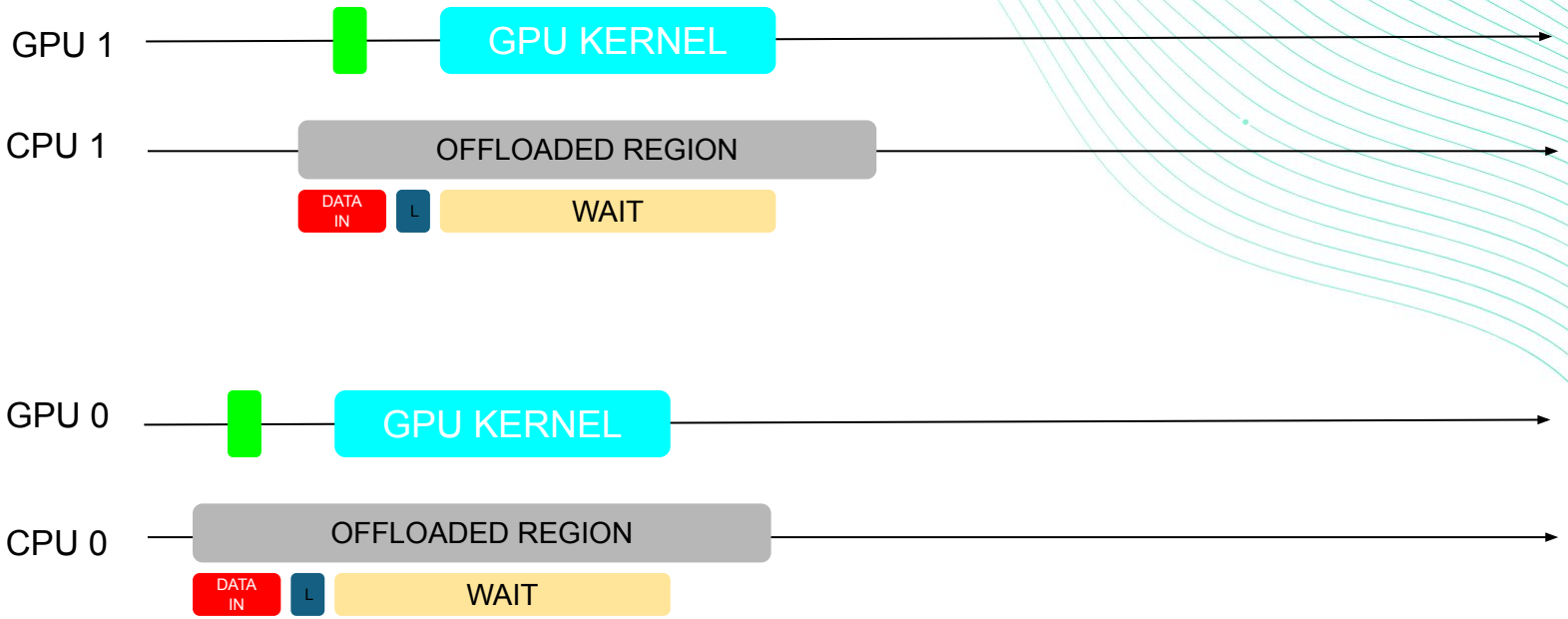
Multi-GPU Execution with MPI



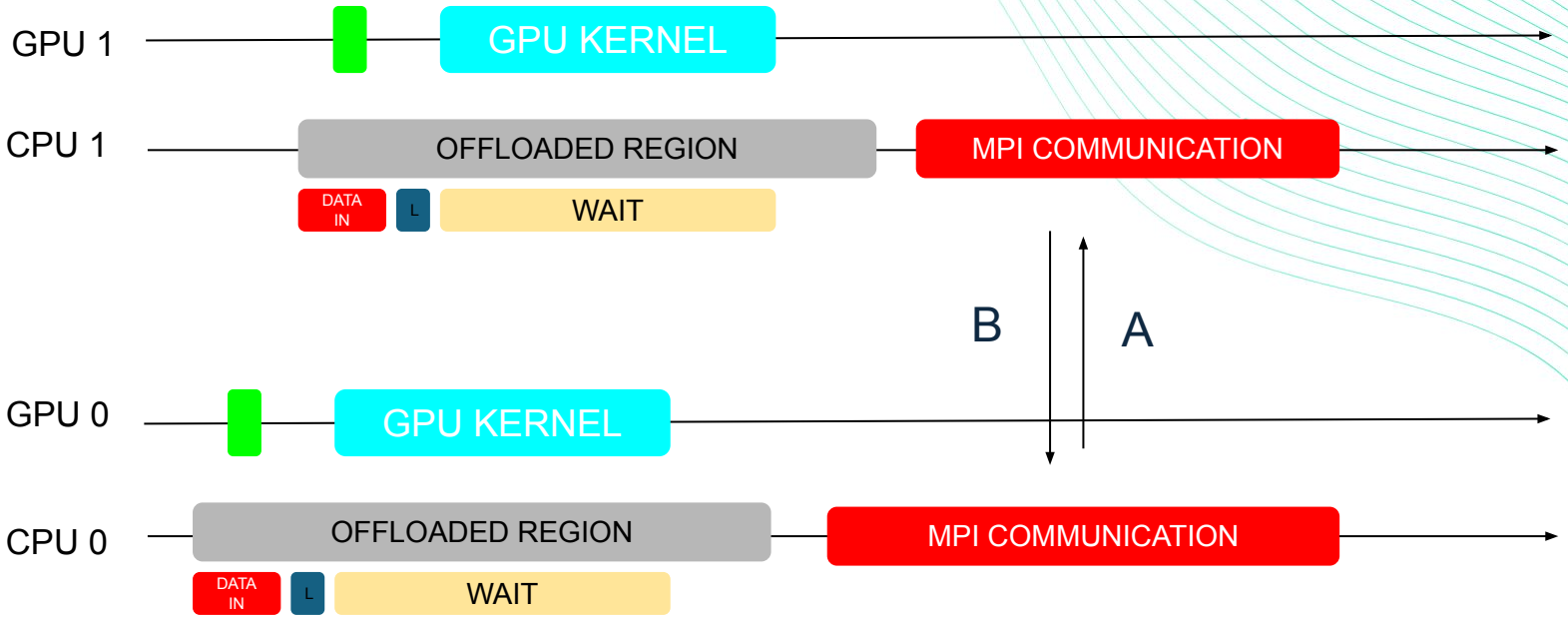
- initialize MPI
- bind MPI task to one GPU
- how to handle communications?

```
USE openacc

# initialize MPI
call mpi_comm_rank(comm, rank, ierr)
call mpi_comm_size(comm, size, ierr)

device = acc_get_device_type()
ndev = acc_get_num_devices( device )
mygpu = mod(rank, ndev)
call acc_set_device_num(mygpu, device)
```

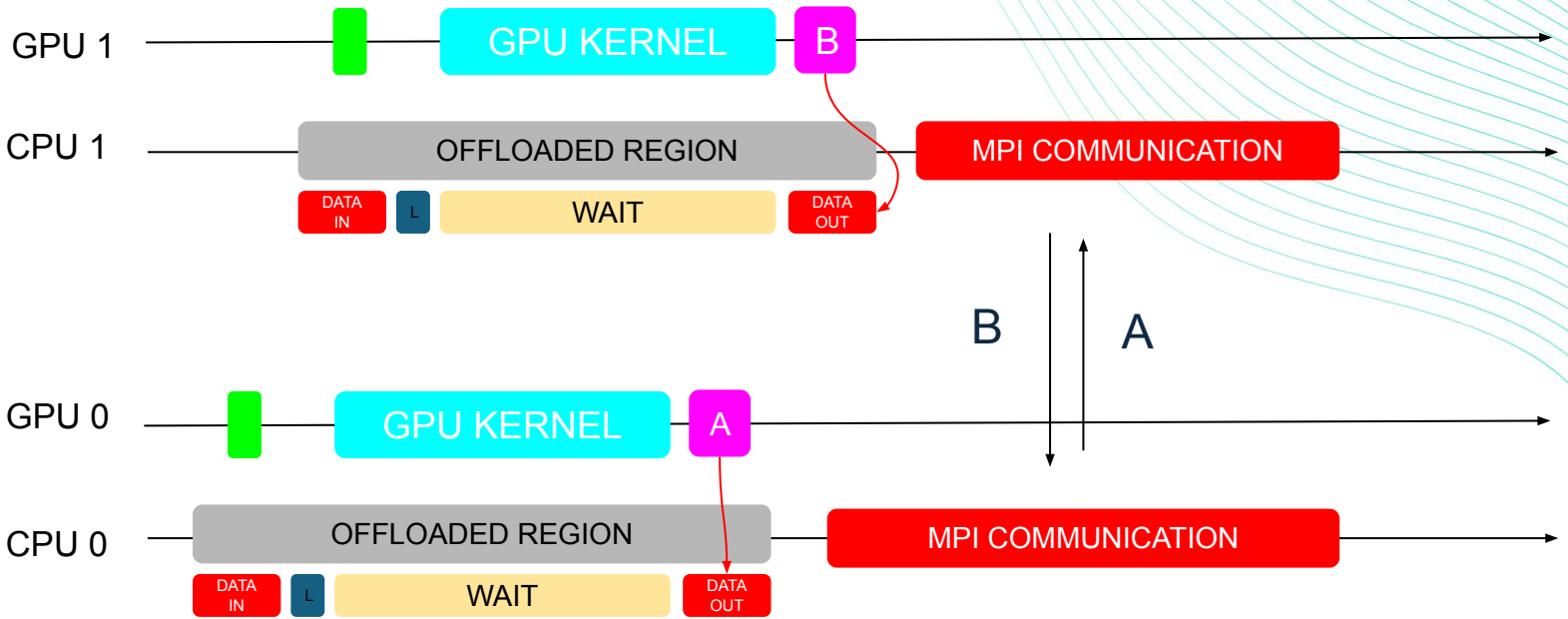
GPU to GPU communications



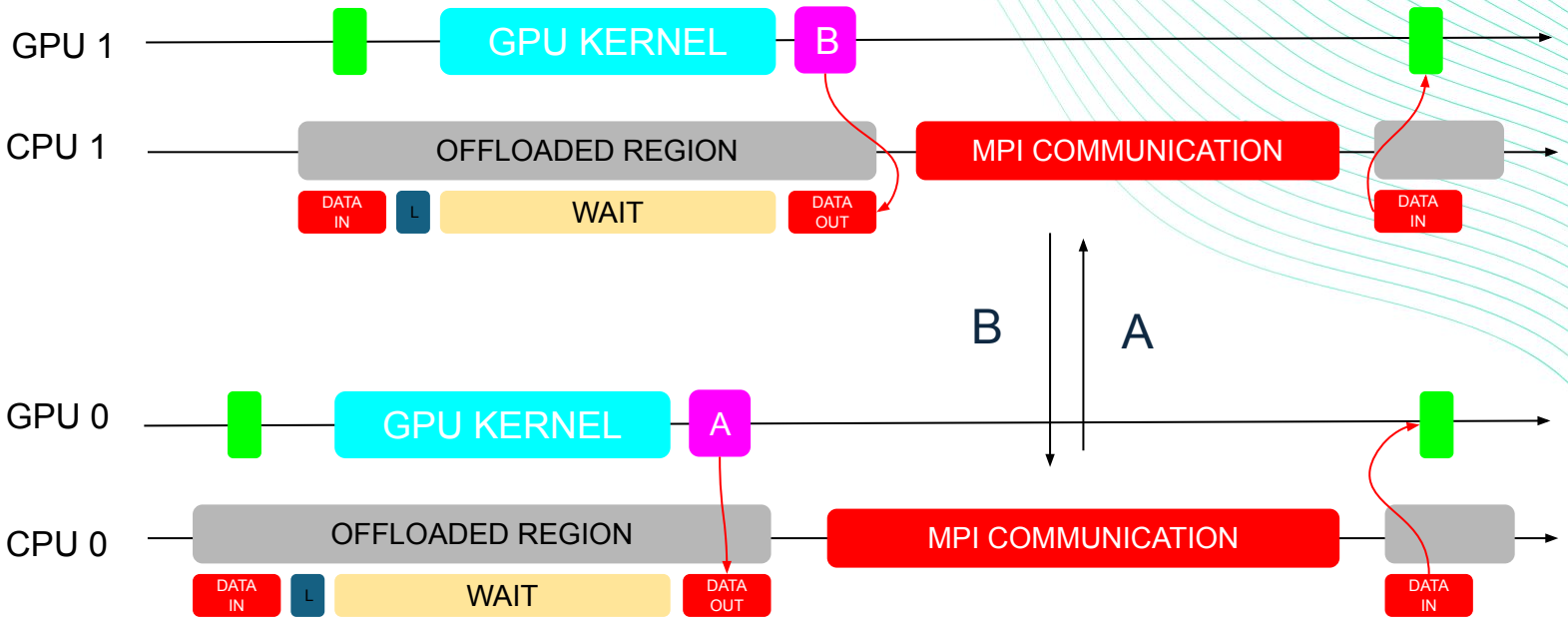
GPU to GPU communications



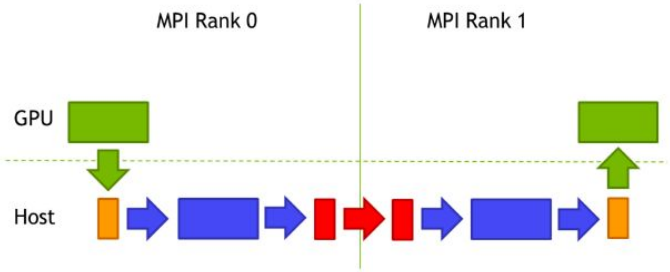
GPU to GPU communications

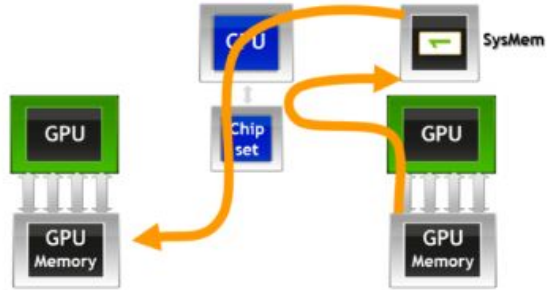


GPU to GPU communications



GPU Awareness in MPI



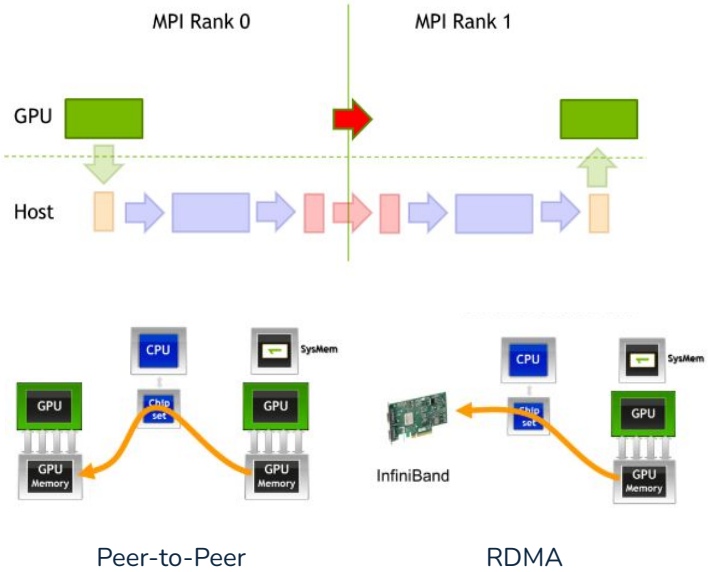


DATA STAGING THROUGH THE CPU

- Data is on the gpu for computations
- Data updated on the host before communication
- MPI send/recv the buffer on the host
- Data updated on the device after communication

```
[x computed on the device...]
!$acc update host(x)
call mpi_sendrecv(x, ...)
!$acc update device(x)
[x computed on the device...]
```

GPU Awareness in MPI



GPUDIRECT P2P & RDMA

Technology providing direct communications between GPU memories

- Data is on the gpu
- MPI send/recv the buffer on the GPU
- Computation can continue without update

```
[x computed on the device...]

!$acc host_data use_device(x)

call mpi_sendrecv(x, ...)

!$acc end host_data

[x computed on the device...]
```

MPI instrumentation with nsys

one report per GPU

```
mpirun nsys profile <command-switches> <executable> <executable arguments>
```

single-node report

```
nsys profile <command-switches> mpirun <executable> <executable arguments>
```

wrappers to instrument specific ranks

```
#!/bin/bash
# Use $PMI_RANK for MPICH and $SLURM_PROCID with srun.
if [ $OMPI_COMM_WORLD_RANK -eq 0 ]; then
    nsys profile -e NSYS_MPI_STORE_TEAMS_PER_RANK=1 -t mpi "$@"
else "$@"
fi
```

MPI instrumentation with nsys

Instrumentation of MPI APIs (only OpenMPI and MPICH)

```
mpirun nsys profile --trace=mpi <executable> <executable arguments>
```

- time spent in MPI routines
- size of the data transferred by MPI calls
- CUDA backend if GPU aware

Throughput (GB/s) of the network card interfaces for multi-node communications

```
mpirun nsys profile --nic-metrics=true <executable> <executable arguments>
```

MPI instrumentation with nsys

```
#main


do iter = 1, numiter
       ! Compute the new psi based on the old one
       call jacobistep(psitmp, psi, m, ln)
       if (checkerr .or. iter == numiter) then
              localerror = deltasq(psitmp, psi, m, ln)
              call mpi_allreduce(localerror, error, 1, MPI_DOUBLE_PRECISION, MPI_SUM, comm, ierr)
              error = sqrt(error)
              error = error / bnorm
       end if
       !  Copy back
      psi(1:m,1:ln) = psitmp(1:m,1:ln)

       ! Do a boundary swap
       call haloswap(psi, m, ln, comm, rank, size)
       !  Quit early if we have reached required tolerance
       if (checkerr) then
              if (error .lt. tolerance) then
                     if (rank == 0) write(*,*) 'CONVERGED iteration ', iter, ': terminating'
                     exit
              end if
       end if

       ! End iterative Jacobi loop
       if (mod(iter,printfreq) == 0) then
              if (rank == 0) then
                     if (.not. checkerr) write(*,*) 'completed iteration ', iter
              if (checkerr) write(*,*) 'completed iteration ', iter, ', error = ', error
              end if
       end if
  end do
```

domain in y direction divided among MPI tasks

communications to exchange boundaries and compute error

MPI instrumentation with nsys

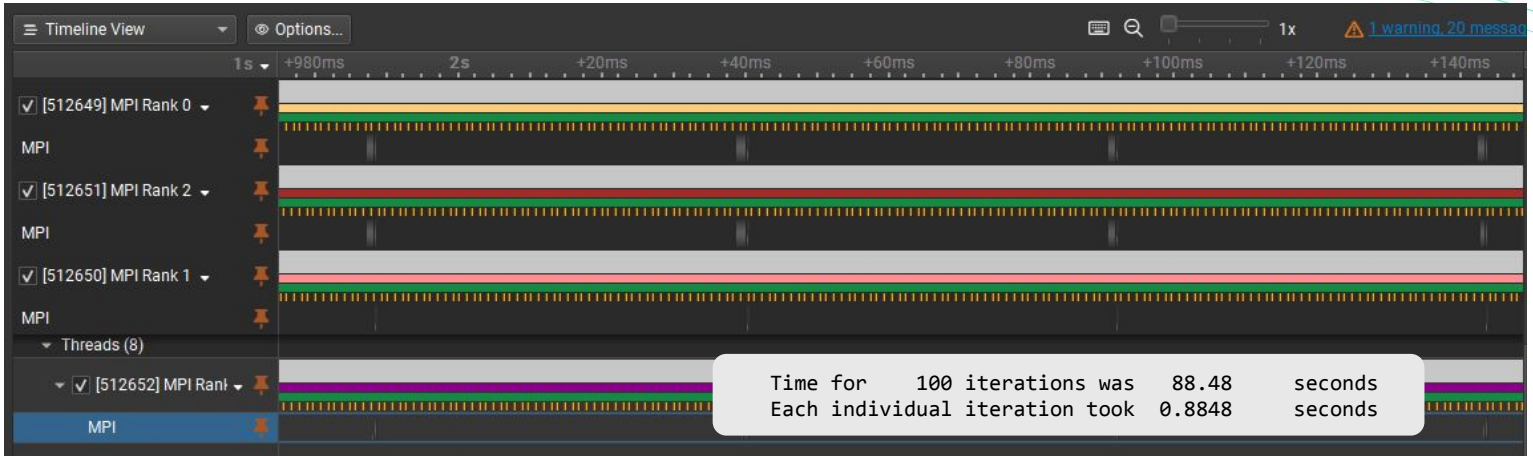
```
# halowasp

!  Send upper boundaries and receive lower ones

if (rank == 0) then
       call mpi_send(x(1,n), m, MPI_DOUBLE_PRECISION, rank+1, tag, comm, ierr)
else if (rank == size-1) then
       call mpi_recv(x(1,0), m, MPI_DOUBLE_PRECISION, rank-1, tag, comm, status, ierr)
else
       call mpi_sendrecv(x(1,n), m, MPI_DOUBLE_PRECISION, rank+1, tag, &
                         x(1,0), m, MPI_DOUBLE_PRECISION, rank-1, tag, &
                         comm, status, ierr)
end if

!  Send lower boundaries and receive upper ones

if (rank == 0) then
       call mpi_recv(x(1,n+1), m, MPI_DOUBLE_PRECISION, rank+1, tag, comm, status, ierr)
else if (rank == size-1) then
       call mpi_send(x(1,1), m, MPI_DOUBLE_PRECISION, rank-1, tag, comm, ierr)
else
       call mpi_sendrecv(x(1,1)   , m, MPI_DOUBLE_PRECISION, rank-1, tag, &
                        x(1,n+1), m, MPI_DOUBLE_PRECISION, rank+1, tag, &
                        comm, status, ierr)
end if
```

domain in y direction divided among MPI tasks

communications to exchange boundaries and compute error

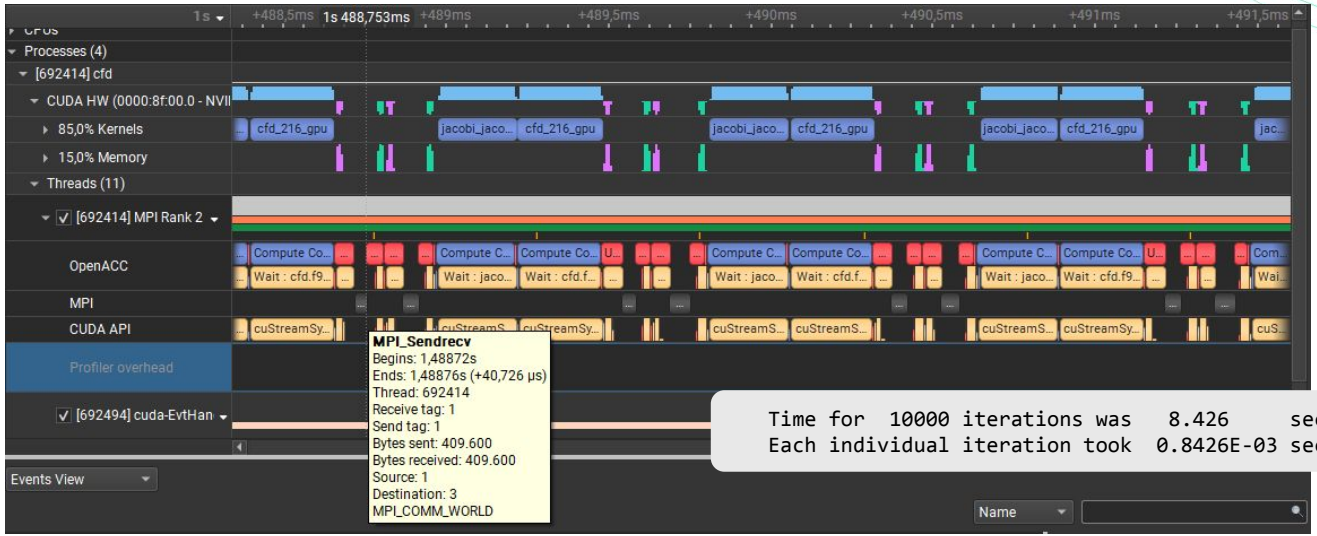
MPI instrumentation with nsys

Time (%)	Total Time (ns)	Instances	Avg (ns)	Med (ns)	Min (ns)	Max (ns)	StdDev (ns)	Style	Range
59.7	1,512,779,804	4	378,194,951.0	378,204,289.5	377,974,204	378,397,021	174,485.4	PushPop	MPI:MPI_Init
26.6	674,914,579	4	168,728,644.8	168,400,483.5	167,938,227	170,175,385	1,038,388.2	PushPop	MPI:MPI_Finalize
7.3	184,815,341	202	914,927.4	856,547.0	61,273	2,114,288	375,638.3	PushPop	MPI:MPI_Send
5.6	142,568,229	404	352,891.7	69,365.0	53,262	2,103,662	495,764.6	PushPop	MPI:MPI_Sendrecv
0.5	13,437,169	202	66,520.6	64,643.0	52,041	421,099	36,175.2	PushPop	MPI:MPI_Recv
0.2	5,499,485	16	343,717.8	1,693.0	771	1,509,796	610,382.6	PushPop	MPI:MPI_Bcast
0.1	1,883,257	8	235,407.1	47,928.0	24,836	782,075	329,251.4	PushPop	MPI:MPI_Allreduce
0.0	134,833	8	16,854.1	17,512.5	6,375	31,953	8,867.8	PushPop	MPI:MPI_Barrier



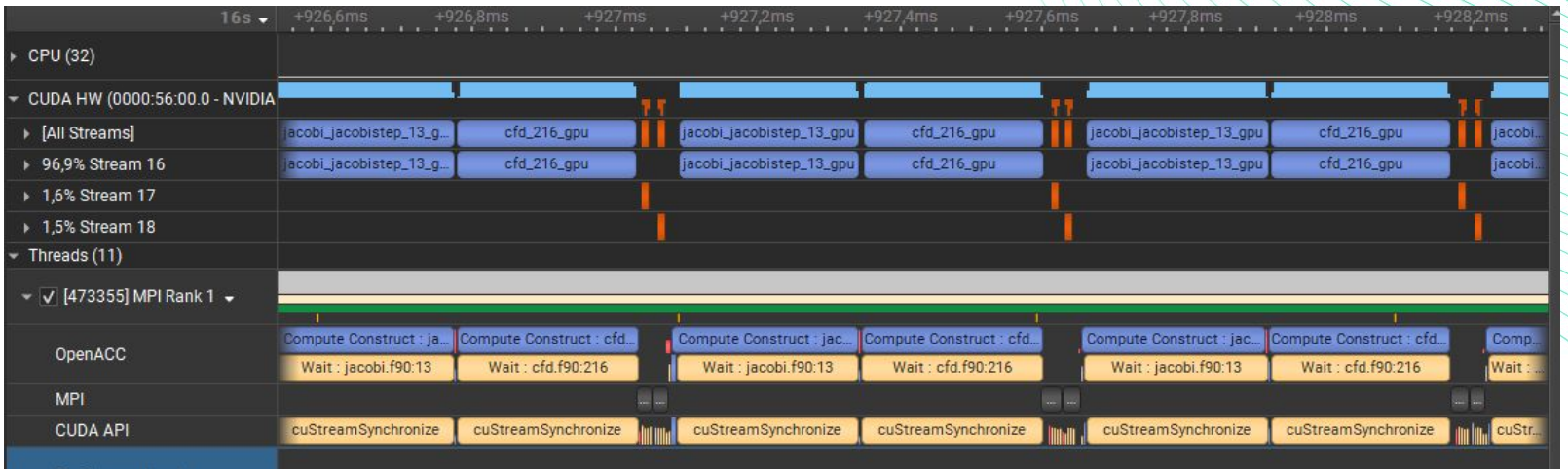
MPI instrumentation with nsys

```
** GPU MemOps Summary (by Time) (gpumemtimesum):

Time (%)  Total Time (ns)  Count   Avg (ns)  Med (ns)  Min (ns)  Max (ns)  StdDev (ns)      Operation
--------  ---------------  ------  --------  --------  --------  --------  -----------  ------------------
    51.5      384,769,082  20,014  19,225.0  18,880.0     2,400   711,137     14,956.5  [CUDA memcpy HtoD]
    48.5      362,883,764  20,013  18,132.4  17,823.0     1,567   642,753     13,660.6  [CUDA memcpy DtoH]
     0.0            1,440       1   1,440.0   1,440.0     1,440     1,440          0.0  [CUDA memset]
```




```
 Time for  10000 iterations was   8.426     seconds
 Each individual iteration took  0.8426E-03 seconds
```

MPI instrumentation with nsys




```
 ** GPU MemOps Summary (by Time) (gpumemtimesum):

Time (%)  Total Time (ns)   Count   Avg (ns)   Med (ns)   Min (ns)  Max (ns)  StdDev (ns)      Operation
--------  ---------------  -------  ---------  ---------  --------  --------  -----------  ------------------
    99.2    1,593,972,986  200,002    7,969.8    7,328.0     7,135    30,624      2,919.6  [CUDA memcpy PtoP]
     0.4        6,680,713       12  556,726.1  673,377.0     2,432   687,201    260,475.2  [CUDA memcpy HtoD]
     0.4        6,289,396       11  571,763.3  639,394.0     2,528   641,506    191,637.5  [CUDA memcpy DtoH]
     0.0            1,888        1    1,888.0    1,888.0     1,888     1,888          0.0  [CUDA memset]
```

EPICURE

Unlocking European-level HPC Support

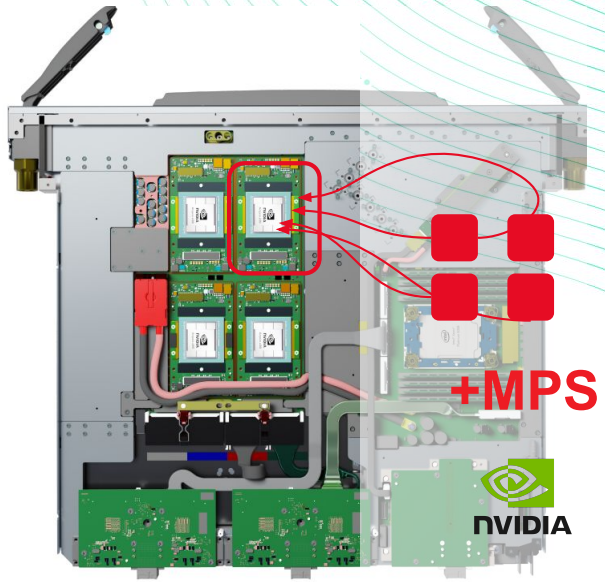
Production use cases

Multistream with NVIDIA MPS

Target application:

- Low GPU utilization in 1:1 MPI-GPU
- Not exhausting GPU available memory
- if MPI: MPI does not distribute memory but computations → kernel size is not affected by MPI distribution

Multi process service (MPS)
optimizes resource usage of multiple MPI tasks
CUDA kernels executed concurrently on the same GPU



→ use the same amount of resources
→ improve GPU utilization

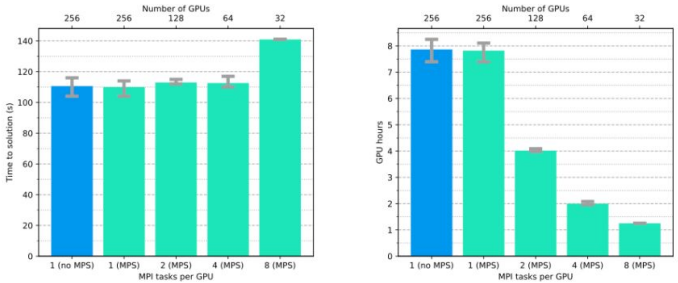
Multistream with NVIDIA MPS

EPICURE white paper explores GPU utilisation improvements with NVIDIA MPS

The EPICURE team published a new white paper explaining that scientific applications can significantly improve resource efficiency without modifying their source code.

Developed by Laura Bellentani (CINECA), Michele Casula (Sorbonne Université) and Tommaso Gorni (CINECA), the study explores the use of NVIDIA Multi-Process Service (MPS) to optimise Quantum Monte Carlo simulations performed with the TurboRVB code on the Leonardo supercomputer. The results showed that the same simulations could be completed using up to four times fewer GPU-hours while maintaining the same time-to-solution.

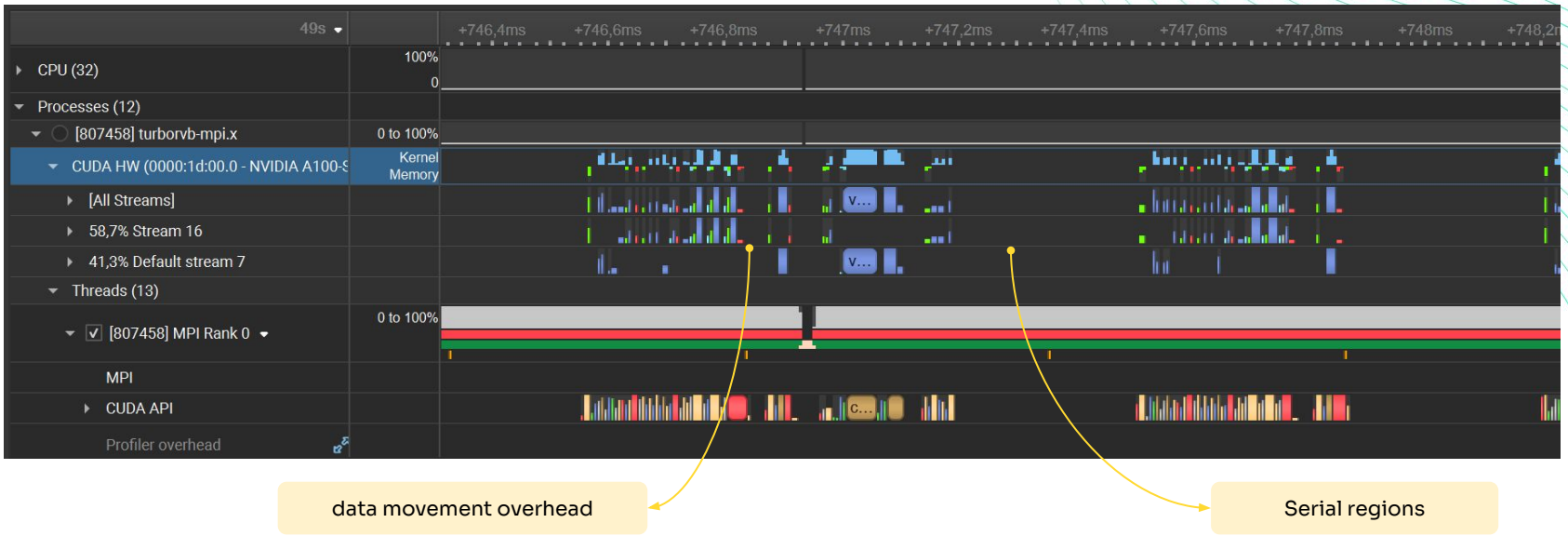
According to Tommaso Gorni, this type of optimisation has implications that go well beyond performance alone: *"saving 4× GPU-hours means saving 4× energy and 4× money, so it's fundamental from both sustainability and computational capability points of view. This way, a researcher can perform 4× more simulations with the same budget."*



https://epicure-hpc.eu/2026/06/15/white-paper-gpu-utilisation-nvidia/

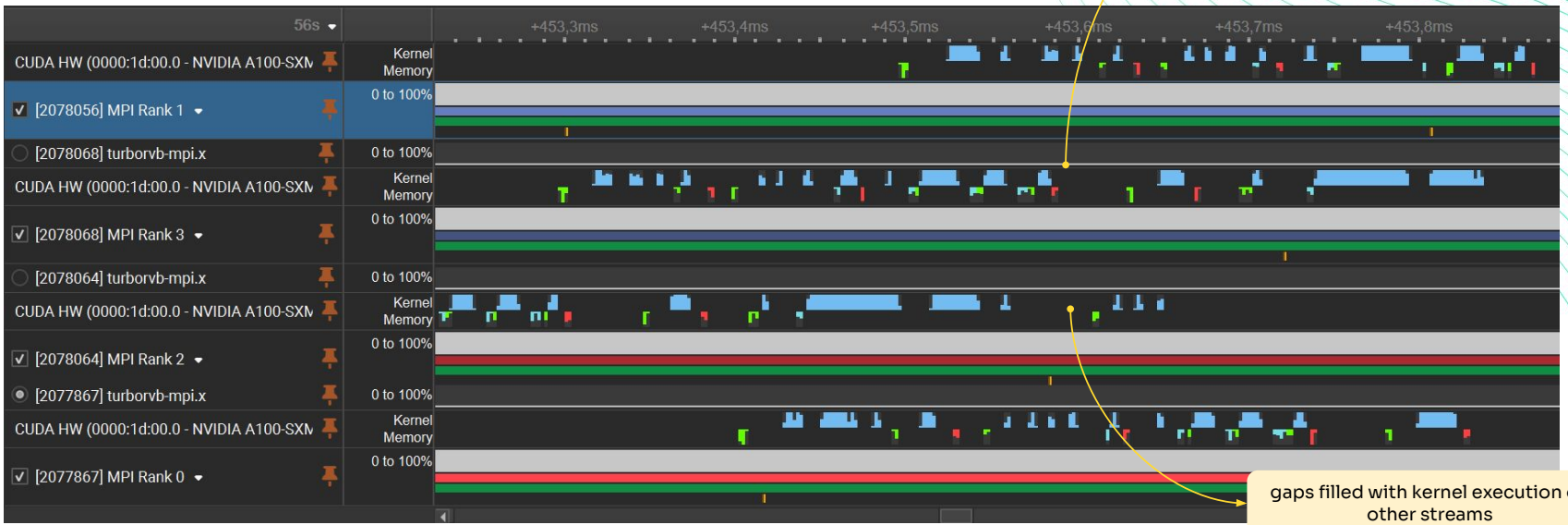
Multistream with NVIDIA MPS

1 MPI/GPU (64 nodes) : limited GPU utilization



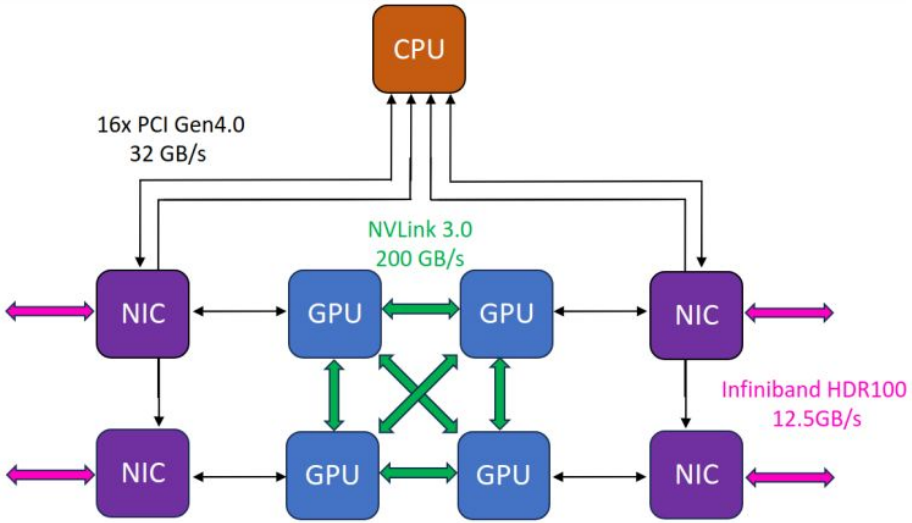
Multistream with NVIDIA MPS

4 MPI/GPU (16 nodes) : improved GPU utilization



NIC utilization

`nsys profile` **`--nic-metrics=true`**



Booster blade intra-node communication pattern (logic view)

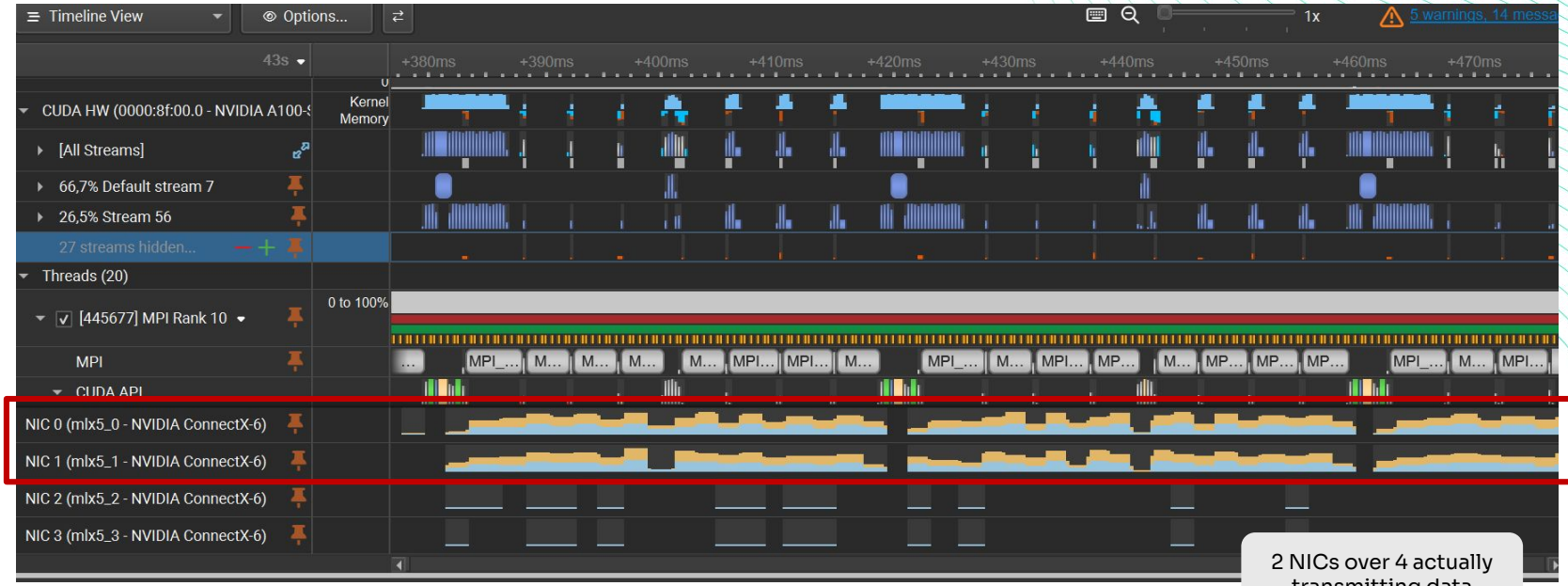
Target application:

- communication-intensive pattern for data distribution
- GPU-aware and non-blocking communications

MPI_Isend + MPI_Irecv + MPI_Waitall

The following results were originally collected in the context of supporting applications under a separate project. They are included here as they form part of the knowledge base that EPICURE AST leverages when onboarding new applications on Leonardo.

NIC utilization



2 NICs over 4 actually transmitting data

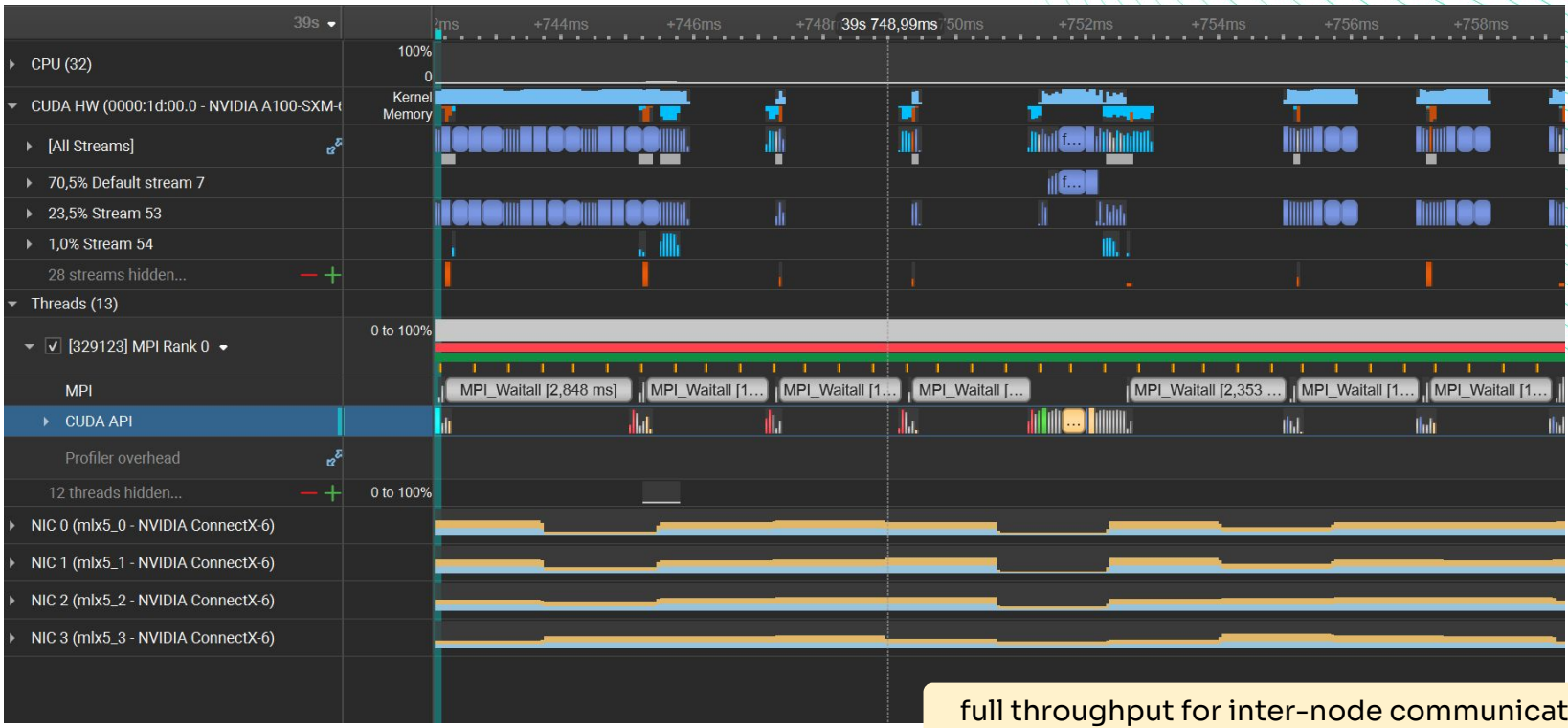
NIC utilization

Rendez-vous protocol → `UCX_MAX_RNDV_RAILS=2, UCX_NET_DEVICES=all`

Fix: bind MPI task to the NIC

```
#!/bin/bash
case $(( ${OMPI_COMM_WORLD_LOCAL_RANK} )) in
0) export UCX_NET_DEVICES=mlx5_0:1 CUDA_VISIBLE_DEVICES=0  ;;
1) export UCX_NET_DEVICES=mlx5_1:1 CUDA_VISIBLE_DEVICES=1  ;;
2) export UCX_NET_DEVICES=mlx5_2:1 CUDA_VISIBLE_DEVICES=2  ;;
3) export UCX_NET_DEVICES=mlx5_3:1 CUDA_VISIBLE_DEVICES=3  ;;
esac
echo Launching on $UCX_NET_DEVICES
$*
```

NIC utilization



full throughput for inter-node communications

NIC utilization

```
nsys stats -r mpi_event_sum
```

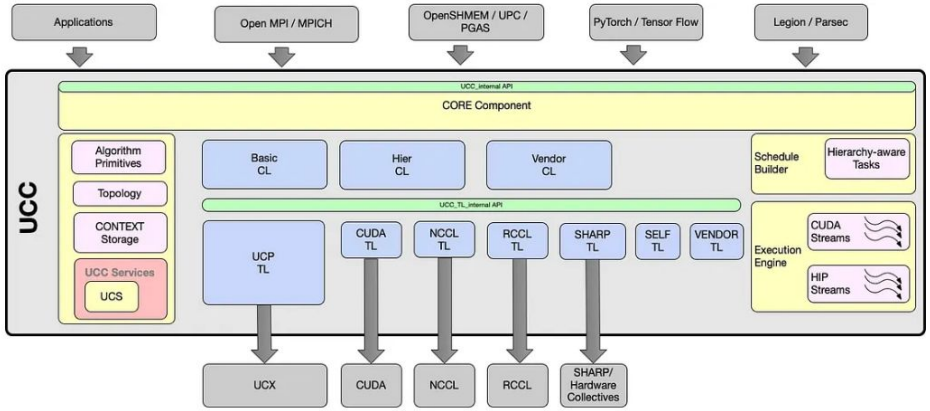
Default

```
** MPI Event Summary (mpi_event_sum):

 Time (%)  Total Time (ns)  Instances     Avg (ns)         Med (ns)         Min (ns)        Max (ns)     StdDev (ns)     Source         Name
 --------  ---------------  ---------  -------------  ---------------  -------------  --------------  ------------  -----------  ---------------
     69.0   23,433,959,722      5,362    4,370,376.7      3,937,090.5        418,487     123,581,023   3,949,169.7  start-wait   MPI_Waitall
     10.5    3,580,174,694        626    5,719,128.9         11,632.0          1,524     468,161,129  39,764,833.4  collectives  MPI_Allreduce
      7.2    2,435,833,106      5,379      452,841.3          4,147.0            247     682,529,404  15,590,393.0  collectives  MPI_Bcast
      5.9    1,993,431,901          1  1,993,431,901.0  1,993,431,901.0  1,993,431,901   1,993,431,901           0.0  other        MPI_Init_thread
      5.2    1,755,508,151      5,942      295,440.6          6,544.0            193     480,782,191   7,437,545.2  collectives  MPI_Barrier
      1.3      453,081,673          1    453,081,673.0    453,081,673.0    453,081,673     453,081,673           0.0  other        MPI_Finalize
      0.7      245,640,104    123,326        1,991.8            617.0            432      43,329,046     153,977.1  p2p          MPI_Isend
      0.1       43,186,821    123,326          350.2            187.0            140       4,273,718      16,991.0  p2p          MPI_Irecv
```

Binding

```
** MPI Event Summary (mpi_event_sum):

 Time (%)  Total Time (ns)  Instances     Avg (ns)         Med (ns)         Min (ns)        Max (ns)     StdDev (ns)     Source         Name
 --------  ---------------  ---------  -------------  ---------------  -------------  --------------  ------------  -----------  ---------------
     61.1   13,121,919,578      5,362    2,447,206.2      1,827,417.5        413,932     120,621,124   3,865,652.7  start-wait   MPI_Waitall
     16.0    3,432,553,685          1  3,432,553,685.0  3,432,553,685.0  3,432,553,685   3,432,553,685           0.0  other        MPI_Init_thread
     13.4    2,881,436,619        626    4,602,933.9          9,128.5          1,511     662,282,788  36,935,883.3  collectives  MPI_Allreduce
      6.7    1,440,369,487      5,942      242,404.8          6,122.0            147     452,484,132   6,956,530.1  collectives  MPI_Barrier
      1.2      258,374,472      5,379       48,033.9          2,542.0            151      55,312,528   1,249,257.1  collectives  MPI_Bcast
      0.8      175,197,118    123,326        1,420.6            523.0            331      10,709,824      67,904.6  p2p          MPI_Isend
      0.6      134,263,544          1    134,263,544.0    134,263,544.0    134,263,544     134,263,544           0.0  other        MPI_Finalize
      0.2       44,092,613    123,326          357.5            168.0            139       4,597,488      13,898.6  p2p          MPI_Irecv
```

GPU awareness in MPI calls



The components of UCC (source: UCC)

Target application:

- application with a CUDA-aware MPI implementation for collectives
- using latest UCC (v1.5+) for collective communications
- MPI buffer using custom MPI datatypes

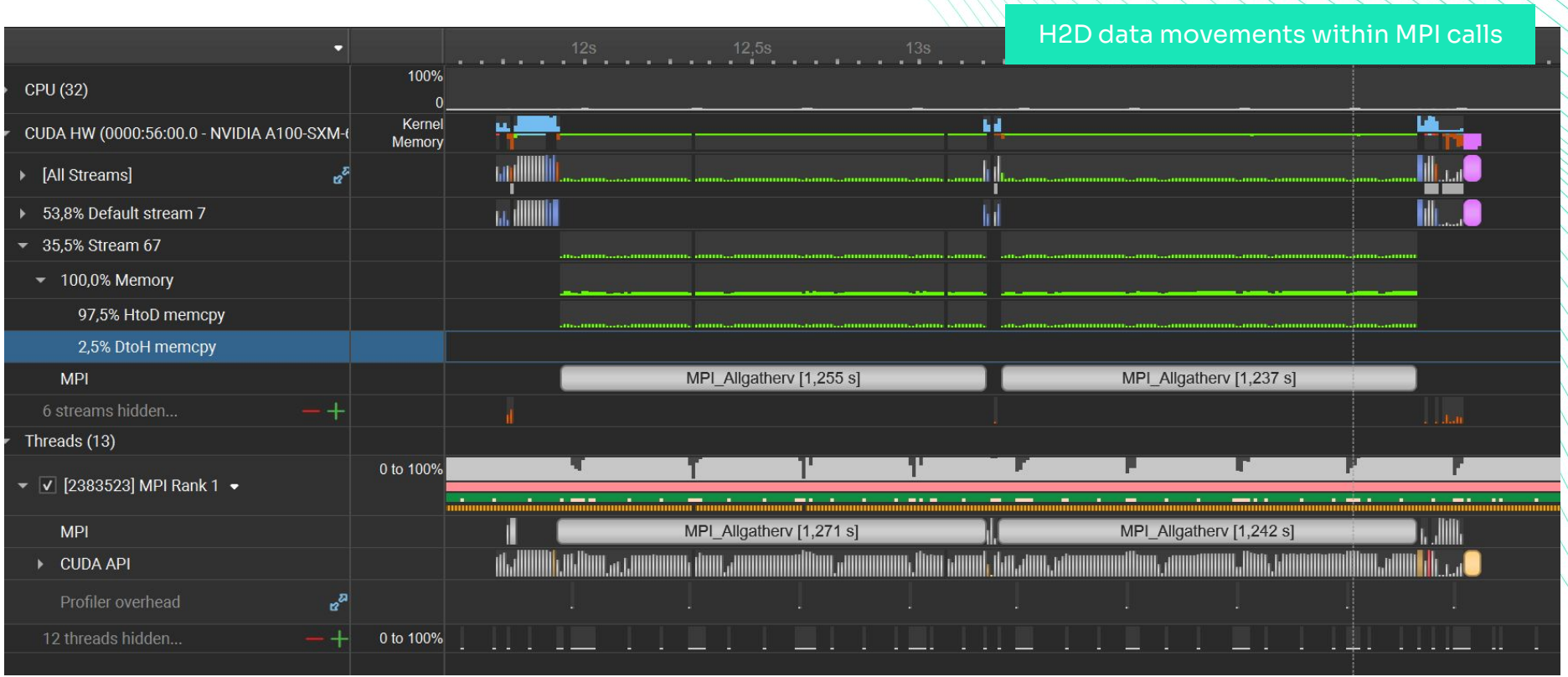
```
CALL mp_type_create_column_section( hpsi(1,1), 0, lda*npol, lda*npol, column_type )
…
!$acc host_data use_device(hpsi)
CALL mp_allgather( hpsi, column_type, recv_counts, displs, inter_bgrp_comm)
!$acc end host_data
```

creates a custom MPI datatype

GPU-aware by design

wrapper to MPI_Allgatherv

GPU awareness in MPI calls



nvhpc/v25_11 – UCC/v1_6

GPU awareness in MPI calls

export UCC_LOG_LEVEL=TRACE

```
[1783344467.163048] [lrdn3202:35538:0] ucc_coll_score_map.c:225  UCC  INFO  Allgatherv:
[1783344467.163048] [lrdn3202:35538:0] ucc_coll_score_map.c:225  UCC  INFO      Host: {0..4095}:TL_UCP:10=ucc_tl_ucp_allgatherv_knomial_init
{4K..inf}:TL_UCP:10=ucc_tl_ucp_allgatherv_ring_init
[1783344467.163048] [lrdn3202:35538:0] ucc_coll_score_map.c:225  UCC  INFO      Cuda: {0..4095}:TL_UCP:10=ucc_tl_ucp_allgatherv_knomial_init
{4K..inf}:TL_UCP:10=ucc_tl_ucp_allgatherv_ring_init
[1783344467.163048] [lrdn3202:35538:0] ucc_coll_score_map.c:225  UCC  INFO      CudaManaged: {0..4095}:TL_UCP:10=ucc_tl_ucp_allgatherv_knomial_init
{4K..inf}:TL_UCP:10=ucc_tl_ucp_allgatherv_ring_init
```

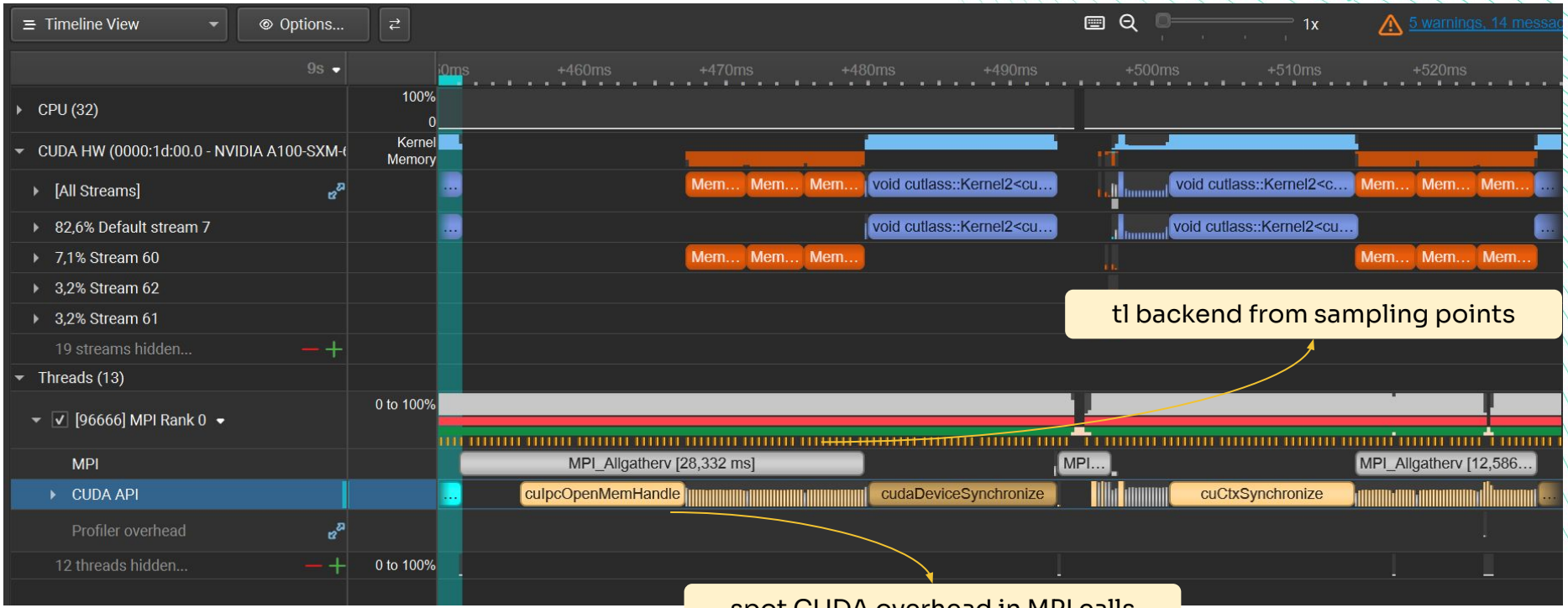
--mca coll_ucc_verbose 4

```
[lrdn1130.leonardo.local:1580810] ../../../../../ompi/mca/coll/ucc/coll_ucc_allgatherv.c:74 - mca_coll_ucc_allgatherv() running ucc allgatherv
[lrdn1130.leonardo.local:1580810] ../../../../../ompi/mca/coll/ucc/coll_ucc_allgatherv.c:32 - mca_coll_ucc_allgatherv_init() ompi_datatype is not supported: dtype =
[lrdn1130.leonardo.local:1580810] ../../../../../ompi/mca/coll/ucc/coll_ucc_allgatherv.c:83 - mca_coll_ucc_allgatherv() running fallback allgatherv
[lrdn1130.leonardo.local:1580808] ../../../../../ompi/mca/coll/ucc/coll_ucc_allgatherv.c:74 - mca_coll_ucc_allgatherv() running ucc allgatherv
[lrdn1130.leonardo.local:1580808] ../../../../../ompi/mca/coll/ucc/coll_ucc_allgatherv.c:32 - mca_coll_ucc_allgatherv_init() ompi_datatype is not supported: dtype =
[lrdn1130.leonardo.local:1580808] ../../../../../ompi/mca/coll/ucc/coll_ucc_allgatherv.c:83 - mca_coll_ucc_allgatherv() running fallback allgatherv
[lrdn1130.leonardo.local:1580809] ../../../../../ompi/mca/coll/ucc/coll_ucc_allgatherv.c:74 - mca_coll_ucc_allgatherv() running ucc allgatherv
[lrdn1130.leonardo.local:1580809] ../../../../../ompi/mca/coll/ucc/coll_ucc_allgatherv.c:32 - mca_coll_ucc_allgatherv_init() ompi_datatype is not supported: dtype =
[lrdn1130.leonardo.local:1580809] ../../../../../ompi/mca/coll/ucc/coll_ucc_allgatherv.c:83 - mca_coll_ucc_allgatherv() running fallback allgatherv
[lrdn1130.leonardo.local:1580807] ../../../../../ompi/mca/coll/ucc/coll_ucc_allgatherv.c:74 - mca_coll_ucc_allgatherv() running ucc allgatherv
[lrdn1130.leonardo.local:1580807] ../../../../../ompi/mca/coll/ucc/coll_ucc_allgatherv.c:32 - mca_coll_ucc_allgatherv_init() ompi_datatype is not supported: dtype =
[lrdn1130.leonardo.local:1580807] ../../../../../ompi/mca/coll/ucc/coll_ucc_allgatherv.c:83 - mca_coll_ucc_allgatherv() running fallback allgatherv
```

MPI custom datatype not supported in aware collectives

```
-    CALL mp_allgather( hpsi, column_type, recv_counts, displs, inter_bgrp_comm)
+    CALL MPI_ALLGATHERV(MPI_IN_PLACE, 0, MPI_DOUBLE_COMPLEX, hpsi, recv_counts*lda*npol, displs*lda*npol, MPI_DOUBLE_COMPLEX,
inter_bgrp_comm, ierr)
```

GPU awareness in MPI calls



GPU awareness in MPI calls

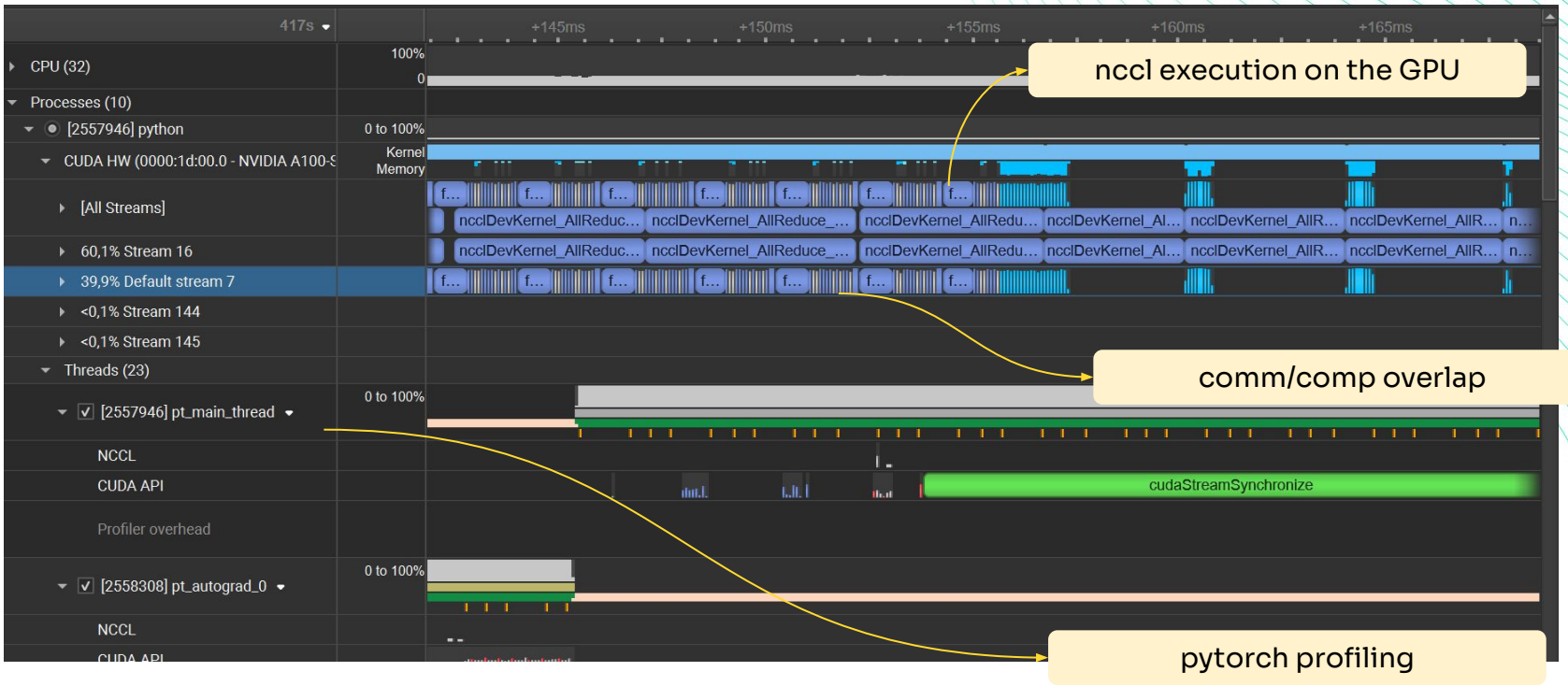
MPI custom datatype (MPI time: 28+ s)

```
** MPI Event Summary (mpi_event_sum):

Time (%)  Total Time (ns)  Instances     Avg (ns)        Med (ns)       Min (ns)       Max (ns)      StdDev (ns)     Source         Name
--------  ---------------  ---------  --------------  --------------  ------------  -------------  -------------  -----------  ---------------
    78.7   24,307,798,636         94    258,593,602.5    26,892,332.0       113,421  1,307,471,307  424,817,695.4  collectives  MPI_Allgatherv
     7.7    2,376,934,716      1,102      2,156,928.1           905.0           156    125,462,732    9,978,562.8  collectives  MPI_Barrier
     5.4    1,675,213,511      2,079        805,778.5       199,143.0         1,526    191,202,838    6,528,171.9  collectives  MPI_Allreduce
     4.5    1,403,379,077          1  1,403,379,077.0  1,403,379,077.0  1,403,379,077  1,403,379,077            0.0  other        MPI_Init_thread
     2.7      822,282,988        102      8,061,597.9       3,142,518.0     2,574,271     52,105,746   10,583,596.1  collectives  MPI_Gatherv
     0.5      166,215,711      1,205        137,938.3           1,084.0           138    101,064,299    2,959,293.5  collectives  MPI_Bcast
     0.4      125,636,435          1    125,636,435.0   125,636,435.0   125,636,435    125,636,435            0.0  other        MPI_Finalize
```

MPI double complex (MPI time: 4+ s)

```
** MPI Event Summary (mpi_event_sum):

Time (%)  Total Time (ns)  Instances     Avg (ns)        Med (ns)       Min (ns)       Max (ns)      StdDev (ns)     Source         Name
--------  ---------------  ---------  --------------  --------------  ------------  -------------  -------------  -----------  ---------------
    32.0    1,848,635,976      2,079        889,194.8       190,070.0         1,733    181,422,790    6,517,119.0  collectives  MPI_Allreduce
    24.4    1,412,217,701          1  1,412,217,701.0  1,412,217,701.0  1,412,217,701  1,412,217,701            0.0  other        MPI_Init_thread
    21.8    1,262,397,673        102     12,376,447.8       7,999,604.5     5,833,828     43,710,466    8,996,143.0  collectives  MPI_Gatherv
    12.8      738,858,125         94      7,860,192.8         433,010.0        86,592    113,886,690   17,203,492.9  collectives  MPI_Allgatherv
     4.4      252,339,743      1,102        228,983.4             518.0           148     96,454,543    3,336,380.3  collectives  MPI_Barrier
     2.5      144,477,986      1,299        111,222.5         1,312.0           137     72,266,678    2,091,901.8  collectives  MPI_Bcast
     2.1      120,348,919          1    120,348,919.0   120,348,919.0   120,348,919    120,348,919            0.0  other        MPI_Finalize
```

NCCL and Python profiling



Takeaways

- Nsight Systems is the first step for understanding where time goes, CPU, GPU, or network
- Async programming and multistream hide kernel launch overhead and data movement, and the efficiency can be verified on the timeline
- Do not give GPU-awareness in MPI for granted; nsys can be used to check if your MPI calls are really exploiting GPUdirect technology
- Use NVTX ranges to profile short, targeted runs, not the whole application

NVIDIA NSight Systems user-guide:
https://docs.nvidia.com/nsight-systems/UserGuide/index.html

EPICURE
Unlocking European-level HPC Support

Thank you!

Follow us

pmo-epicure@postit.csc.fi

Co-funded by the European Union

EuroHPC Joint Undertaking

This project has received funding from the European High Performance Computing Joint Undertaking under grant agreementNo.101139786. Views and opinions expressed are, however, those of the author(s) only and do not necessarily reflect those of the European Union orEuroHPC Joint Undertaking. Neither the European Union nor the granting authority can be held responsible for them.