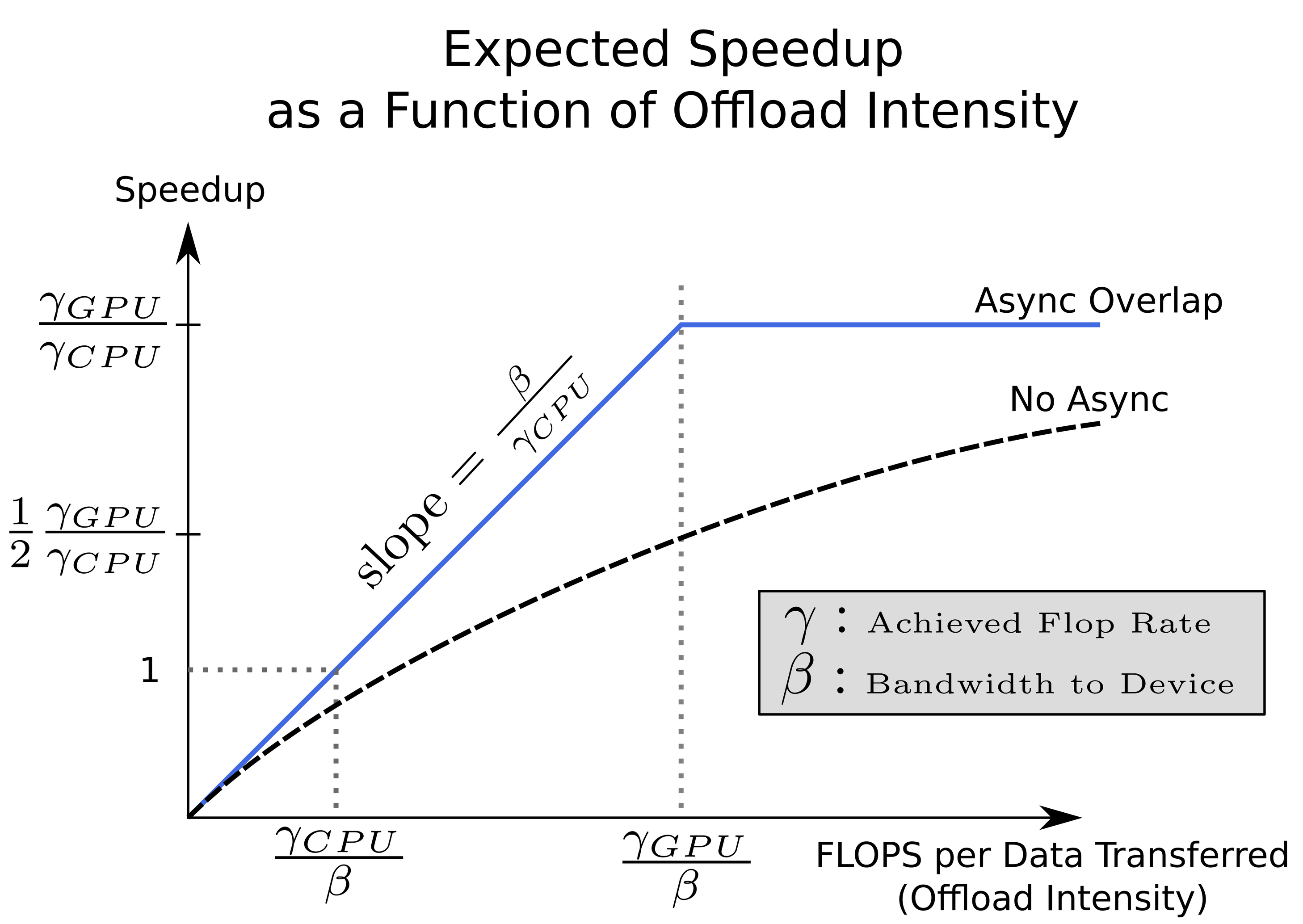
OpenMP 4 Asynchronous Data Movement with the XL Fortran Compiler

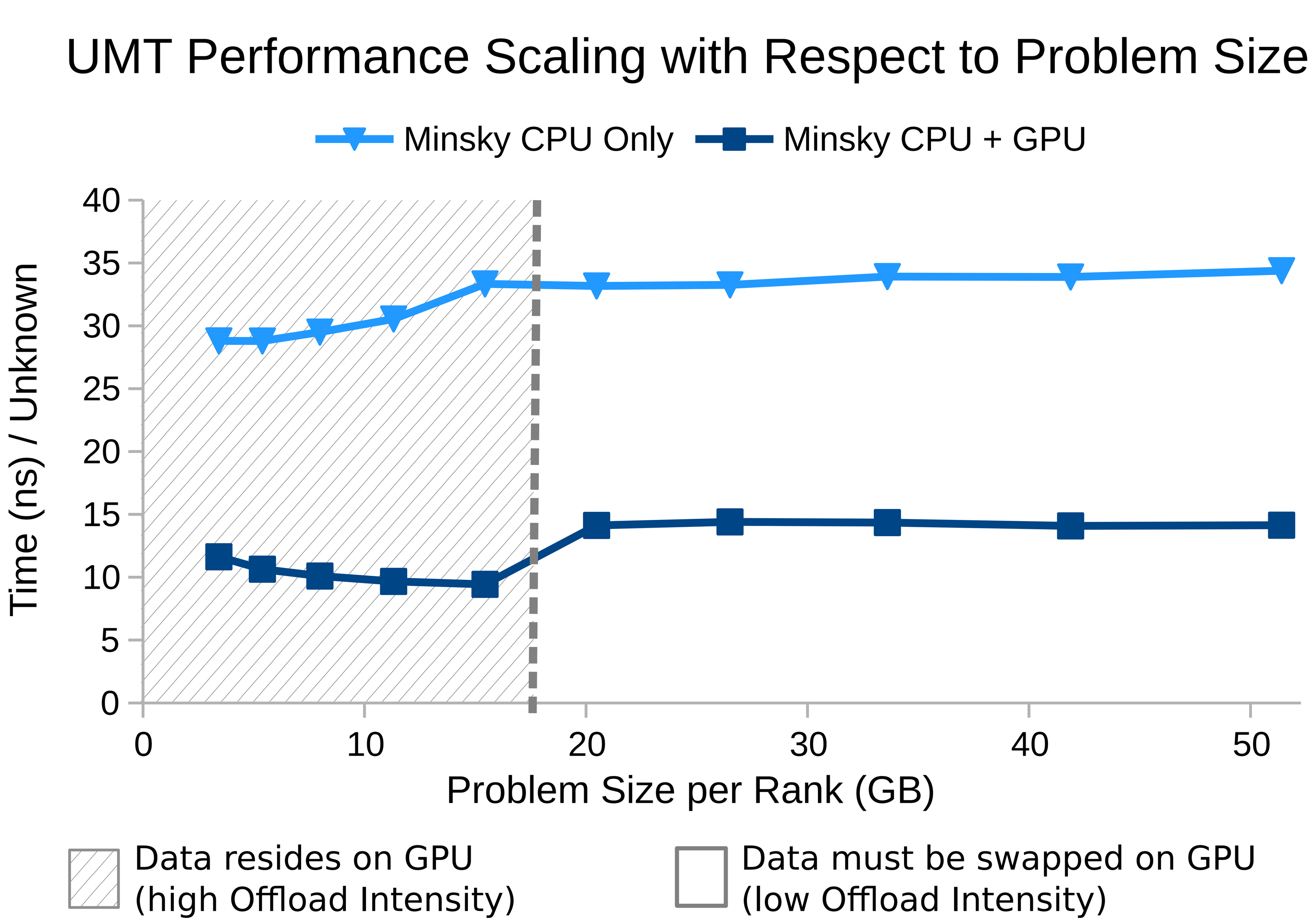
David Appelhans, dappelh@us.ibm.com   IBM Research

Introduction

- Teton proxy application (UMT, 50K lines) has been ported to the GPU using CUDA Fortran.
- Because of the fast NVLINK connection to system memory and the implementation of asynchronous data movement, performance is only reduced slightly when the data no longer fits in GPU memory [1].
- **We would like to achieve the same with OpenMP 4 offloading.**

Motivation





Minimal loss of performance in CUDA implementation above because 90% of compute is overlapped by data movement.

Challenges

1. Mapping deeply nested objects.
2. Fast data movement requires PINNED attribute.
3. Async movement and execution.

How to Map Deep Objects

Need to map deeply nested data types. A simplified example of mapping 2 level data is

```fortran
type, public :: ZoneData

  integer              :: nCorner        ! Scalar needed on GP
  real(adqt), pointer  :: STotal (:,:)   ! Not needed on GPU
  real(adqt), pointer  :: STime (:,:,:)  ! Needed on GPU

end type ZoneData

type(ZoneData), allocatable :: ZData(:)

allocate(ZData(nzones))
```

Map the base object (ZData), then map the desired members.

```fortran
!$omp target enterdata map(to:ZData)
do zone=1,nzones
   !$omp target enterdata map(to:ZData(zone)%Stime)
enddo

!$omp target
do zone = 1, nzones
   do c =1, ZData(zone)%nCorner
      do i=1,32
         ZData(zone)%STime(i,c,1) = i*2.0
      enddo
   enddo
enddo
!$omp end target

do zone =1,nzones
   !$omp target exitdata map(from:ZData(zone)%STime)
enddo

!$omp target exitdata map(release:ZData)
```

- We can safely work with %STime in the target region.
- If we tried to access %STotal, which we did not map, we would crash the GPU silently.

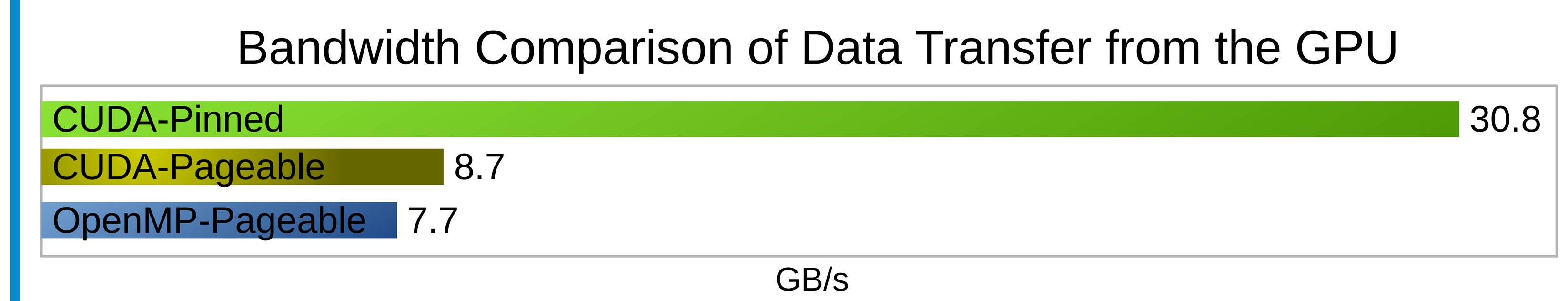
References

[1] D. Appelhans and B. Walkup. Leveraging nvlink and asynchronous data transfer to scale beyond the memory capacity of gpus. *ScalA Workshop Proceedings*, In Preparation 2017.

* Work displayed in this poster was done as part of the CORAL CoE contract.

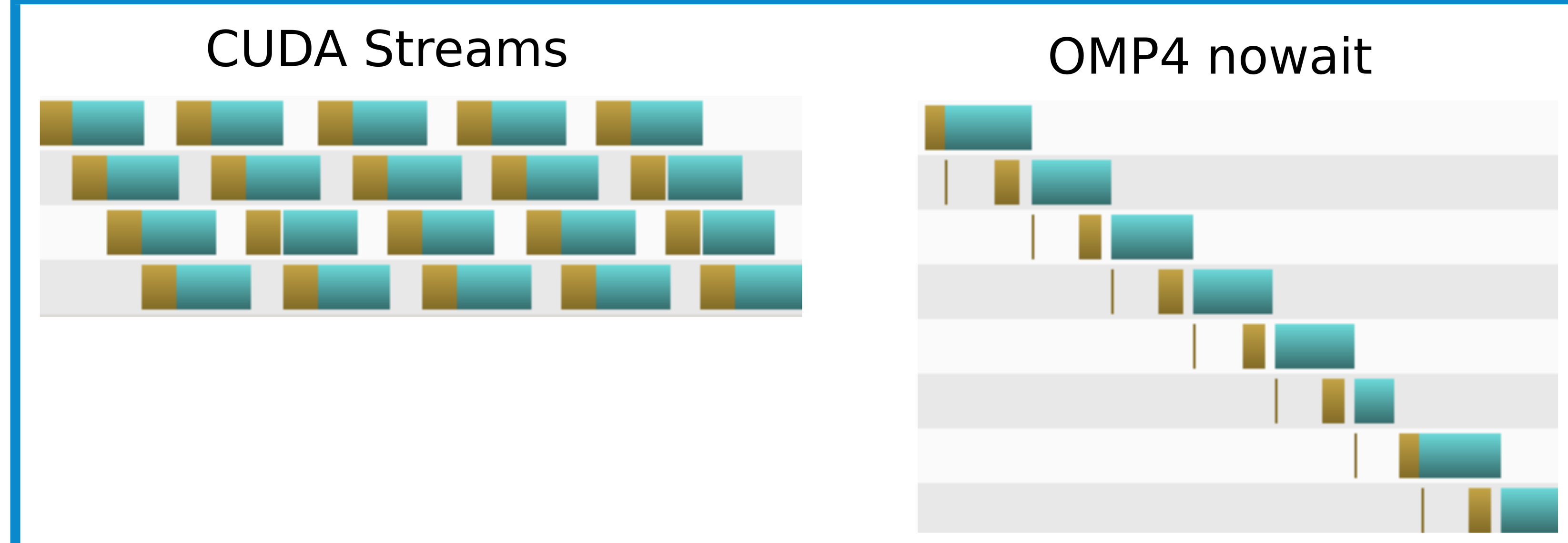
Pinned Memory

```fortran
real(kind=8), allocatable, pinned ::   A (:,:)
```



Pinned is not a supported concept with OpenMP, but strongly affects performance on current hardware.

Async



- Overlap now happening in OpenMP 4 runtime, albeit preliminary status.
- Ongoing efforts in XL compiler and runtime to improve the overlap. Functionality need identified and under development.
- OpenMP interoperability with CUDA streams would be very helpful for incrementally porting codes and achieving high performance.

Conclusions

- Asynchronous data movement combined with the NVLINK CPU to GPU connnection allows a new class of problems to be accelerated on GPUs.
- Achieving this with OpenMP 4 requires overlapping data mapping with execution of a target region. Depends and nowait clauses are expected to help with this, but require OpenMP only approach.
- OpenMP 4 interoperability with CUDA streams would help usability of OpenMP.
- Portability does not have to be an OpenMP 4 only approach.
- If 9 out of 10 kernels are ported using a directive approach, writing one kernel using a hardware specific language is realistic to maintain, and a viable path to achieving high performance.