**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**

Expected Speedup as a Function of Offload Intensity

![Graph showing speedup as a function of offload intensity](image)

**UMT Performance Scaling with Respect to Problem Size**

![Graph showing performance scaling](image)

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 GPU
  real(adiq), pointer :: STime (:,:) ! Needed on GPU
  real(adiq), pointer :: STotal (:,:) ! Not needed on GPU
end type ZoneData

type, public :: ZoneData
  integer :: nCorner ! Scalar needed on GPU
  real(adiq), pointer :: STime (:,:) ! Needed on GPU
end type ZoneData

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

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

**Pinched Memory**

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

Bandwidth Comparison of Data Transfer from the GPU

<table>
<thead>
<tr>
<th>Bandwidth (GB/s)</th>
<th>CUDA-Pinned</th>
<th>CUDA-Pageable</th>
<th>OpenMP-Pageable</th>
</tr>
</thead>
<tbody>
<tr>
<td>30.8</td>
<td>30.8</td>
<td>8.7</td>
<td>7.7</td>
</tr>
</tbody>
</table>

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 connection 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.

**References**


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