Scalable High-Fidelity Simulation of Turbulence With NEKO Using Accelerators

Niclas Jansson[1], Martin Karp, Jacob Wahlgren, Stefano Markidis, Philipp Schlatter

[1]PDC Center for High Performance Computing
KTH Royal Institute of Technology

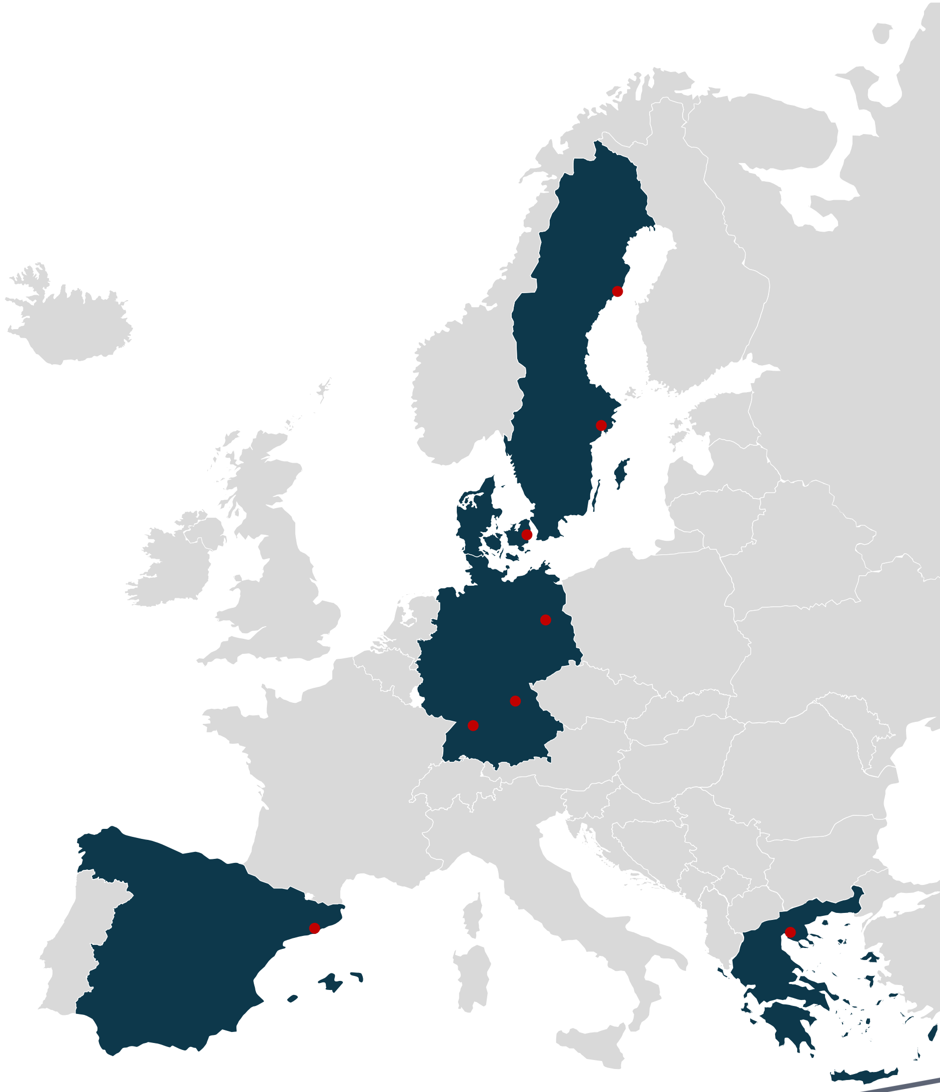
CEEC
Centre of Excellence in Exascale CFD

CEEC

Centre of Excellence in Exascale CFD

The main goal of CEEC is to address the extreme-scale computing challenge to enable the use of accurate and cost-efficient high fidelity computational fluid dynamics (CFD) simulations at exascale

- Implement **exascale-ready workflows** for addressing grand challenge scientific problems
- Develop **new or improved algorithms** that can efficiently exploit exascale systems.
- Significantly improve **energy efficiency** of simulations
- Demonstrate workflows on **lighthouse cases** relevant for both academia and industry



KTH VETENSKAP OCH KONST

Universität Stuttgart

FAU Friedrich-Alexander-Universität Erlangen-Nürnberg

DTU

BAM

BSC Barcelona Supercomputing Center Centro Nacional de Supercomputación

UMEÅ UNIVERSITET

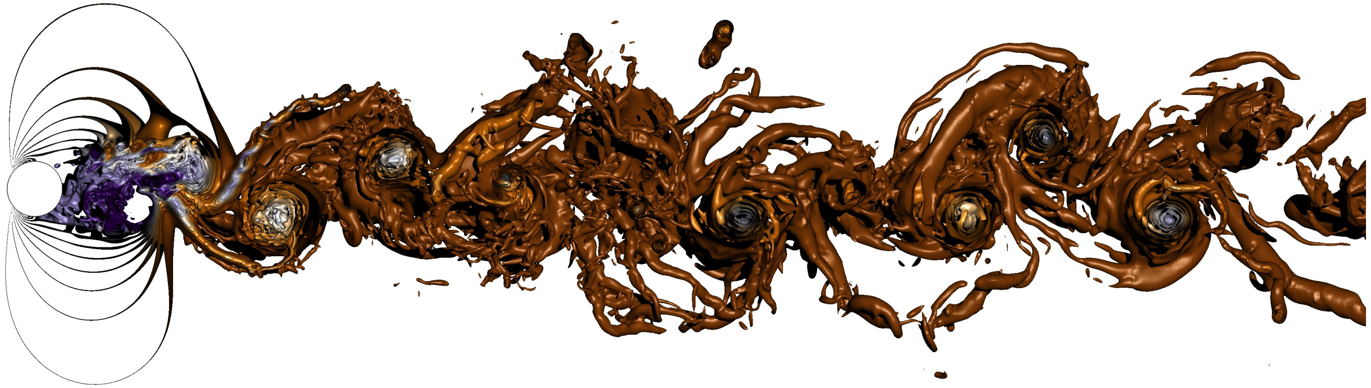
ceec-coe.eu

Introduction

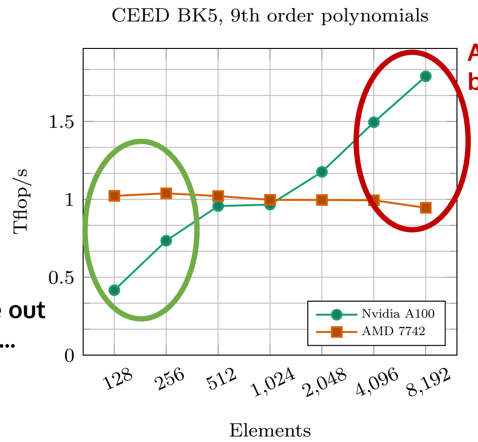
- Exascale will require either **unreasonable large problem** sizes or **significantly improved efficiency** of current methods
  - Finite-Volyme, LES of full car on the entire K computer (京) required **more than 100 billion grid points** to run efficiently
  - What problem size is needed to fill the 309 PFlop/s LUMI...
- High-order methods
  - Attractive numerical properties, **small dispersion** errors and more "accuracy" per degree of freedom
  - Better suited to take advantage of **modern hardware** (accelerators)

京: 82944 nodes, 663552 Cores, **10 PFlop/s**

Dardel: 56 nodes, 448 MI250X GCDs, ≈**10 PFlop/s**



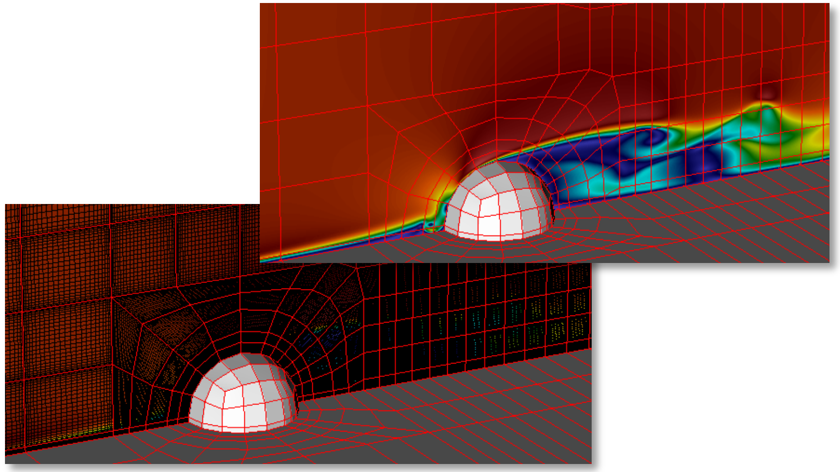
...but we rather scale out our problems in CFD...

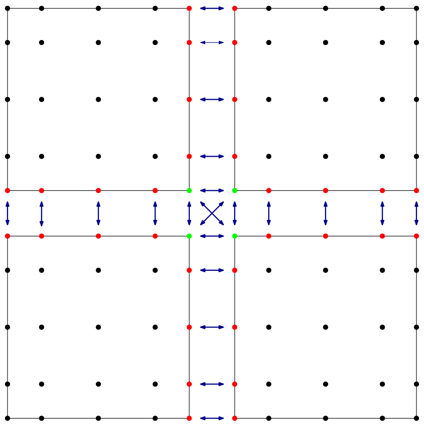


Accelerators works best with a lot of data!

Spectral Elements

- Finite Elements with high-order basis functions
  - N-th order Legendre-Lagrange polynomials $l_i(\xi)$
  - Gauss-Lobatto-Legendre quadrature points ξ_i
  - Fast tensor product formulation
    - $u^e(\xi,\eta,\gamma) = \sum_{i,j,k}^{N} u_{i,j,k}^e \, l_i(\xi) l_j(\eta) l_k(\gamma)$
  - High-order at low cost! (**Level 3 BLAS!**)
- Too expensive to assemble matrices
  - Element stiffness matrices $A_{i,j}^k$ with $\boldsymbol{O(N^6)}$ **non-zeros**
- Matrix free formulation, key to achieve good performance in SEM
  - Unassembled matrix $A_L = \mathrm{diag}\{A^1, A^2, \dots, A^E\}$ and functions $u_L = \{u^e\}_{e=1}^E$
  - Operation count is **only** $\boldsymbol{O(N^4)}$ **not** $\boldsymbol{O(N^6)}$
  - Boolean gather/scatter matrix Q^T and Q
    - Ensure continuity of functions on the element level $u = Q^T u_L$ and $u_L = Qu$
- Q nor Q^T formed, only the action QQ^T is used
  - Matrix-vector product $w = Au \Rightarrow w_L = QQ^T A_L u_L$

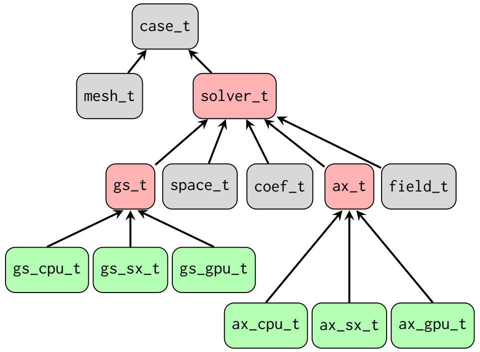


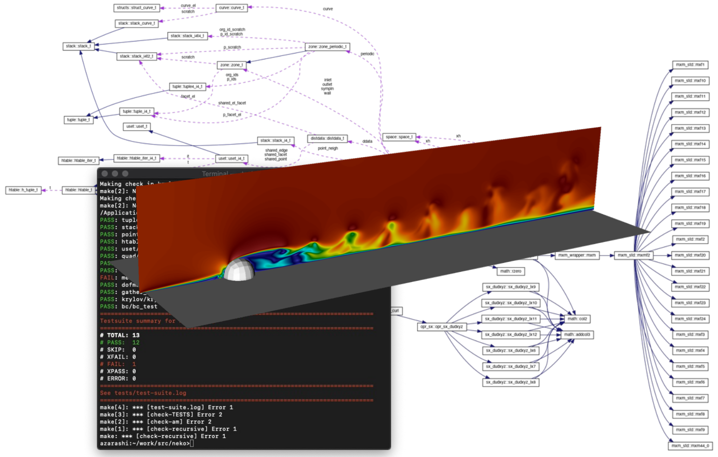


Portable Spectral Element Framework NEKO

- High-order spectral element flow solver
  - Incompressible Navier-Stokes equations
  - Matrix-free formulation, **small tensor products**
  - **Gather-scatter** operations between elements
- Modern **object-oriented** approach (Fortran 2008)

```
! Base type for a matrix-vector product providing Ax
type, abstract :: ax_t
 contains
   procedure(ax_compute), nopass, deferred :: compute
end type ax_t

! Abstract interface for computing Ax
abstract interface
   subroutine ax_compute(w, u, coef, msh, Xh)
     implicit none
     type(space_t), intent(inout) :: Xh
     type(mesh_t),  intent(inout) :: msh
     type(coef_t),  intent(inout) :: coef
     real(kind=dp), intent(inout) :: w(:,:,:,:)
     real(kind=dp), intent(inout) :: u(:,:,:,:)
   end subroutine ax_compute
end interface
```

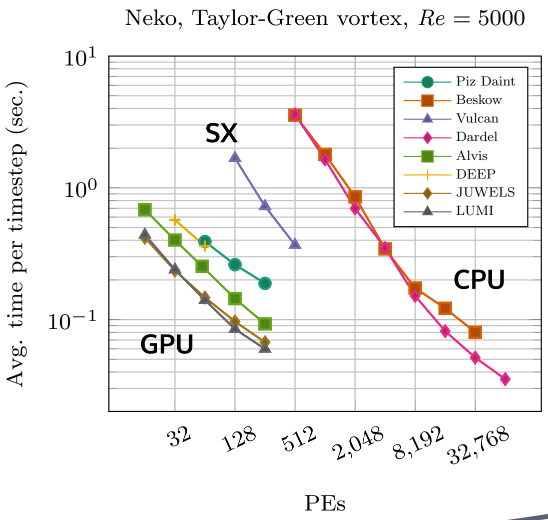




- Various hardware-backends
  - CPUs, GPUs down to exotic vector processors and FPGAs
    - **Device abstraction layer** for accelerators (CUDA/HIP/OpenCL)
- Modern Software Engineering (pFUnit, ReFrame, **Spack**)

```
> spack install neko+cuda
```

ExtremeFLOW/neko



Device Abstraction Layer

How to interface Fortran with accelerators?

Method	Portability	Performance	Programmability
OpenACC	☹️	😐	🙂 / 😬
OpenMP	😐	😐	🙂 / 😬
CUDA Fortran	😵	😎	😎
Native	😉	😎	😎

- Device pointers in each derived type

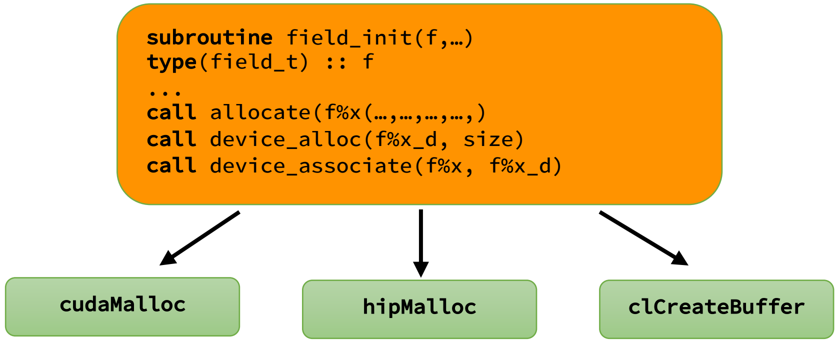
```fortran
type field_t
   real(kind=rp), allocatable :: x(:,:,:,:) !< Field data
   type(space_t), pointer :: Xh     !< Function space
   type(mesh_t), pointer :: msh     !< Mesh
   type(dofmap_t), pointer :: dof   !< Dofmap
   type(c_ptr) :: x_d = C_NULL_PTR !< Device pointer
end type field_t
```

- Abstraction layer hiding memory mgmt.
- Hash table associating x with x_d
- Kernels invoked from the object hierarchy via C interfaces (Ax, vector ops)

```
src/
|
|-- math
  `-- bcknd
        |-- cpu
        |-- device
        |    |-- cuda
        |    |-- hip
        |    `-- opencl
        |-- sx
        `-- xsmm
```

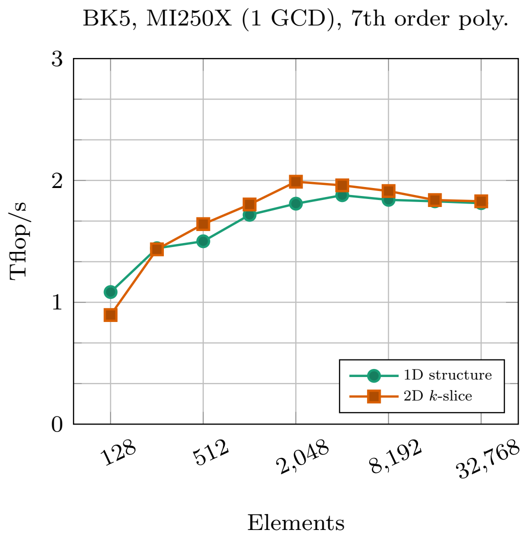
1. Allocate
2. Copy
3. Associate

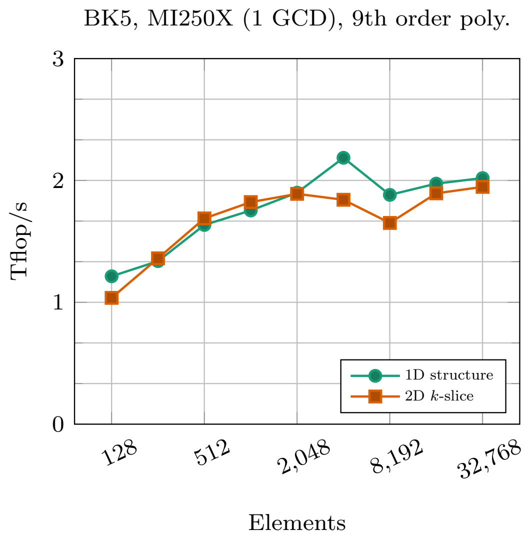
```fortran
!> Enum @a hipError_t
enum, bind(c)
   enumerator :: hipSuccess = 0
   ...
end enum

!> Enum @a hipMemcpyKind
enum, bind(c)
   enumerator :: hipMemcpyHostToHost = 0
   enumerator :: hipMemcpyHostToDevice = 1
   ...
end enum

interface
   integer (c_int) function hipMalloc(ptr_d, s) &
        bind(c, name='hipMalloc')
     use, intrinsic :: iso_c_binding
     implicit none
     type(c_ptr) :: ptr_d
     integer(c_size_t), value :: s
   end function hipMalloc
end interface
```

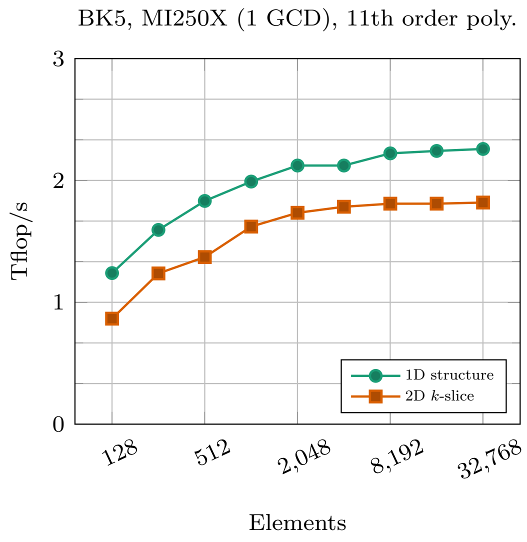


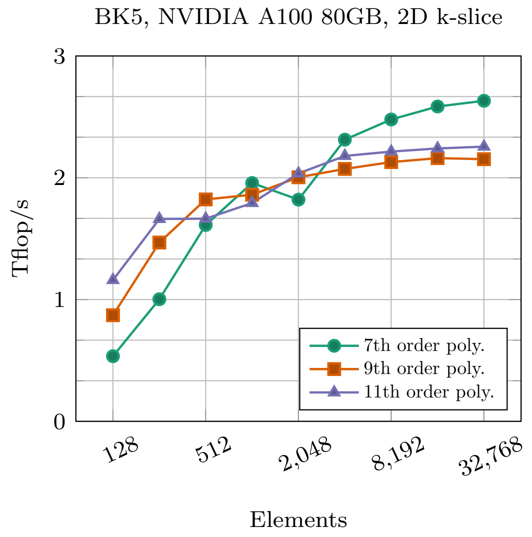
Device Kernel Optimisations

- Neko provides **wrapper functions** for each supported accelerator backend
  - **Templated** (CUDA/HIP) or **pre-processor macros** (OpenCL) for runtime parameters
- Shape and mapping of thread blocks to elements
  - **Significant affect on performance**
  - Very different performance on Nvidia and AMD
- **Auto/runtime tuning** based on polyomial order







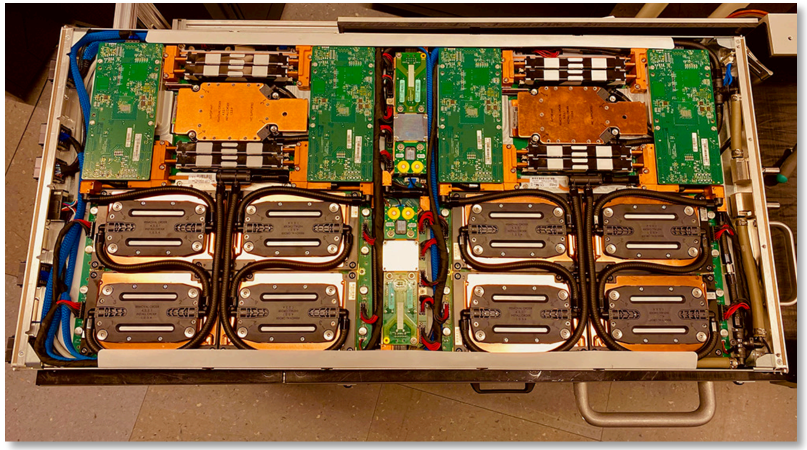


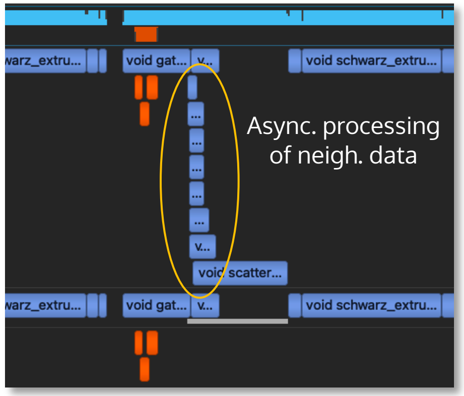
Gather-Scatter – Accelerators

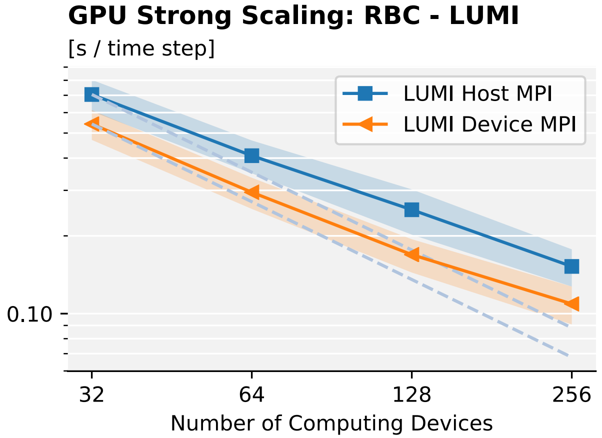
CEEC

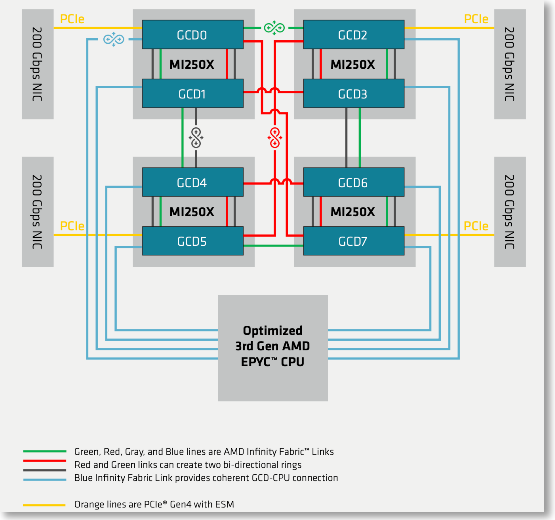
Recent supercomputers (**LUMI/Dardel**) have the NICs **connected directly** to the accelerators

- Nodes have a **complex memory hierarchy**
  - **Avoid memory transfers** at all costs!
- Multiple levels of overlapping communication and computation
  - Overlapping with **non-blocking MPI** (device aware)
  - **Asynchronous** pack/unpack GPU kernels (One stream/neighbour)
  - Auto/runtime tuning of all combinations





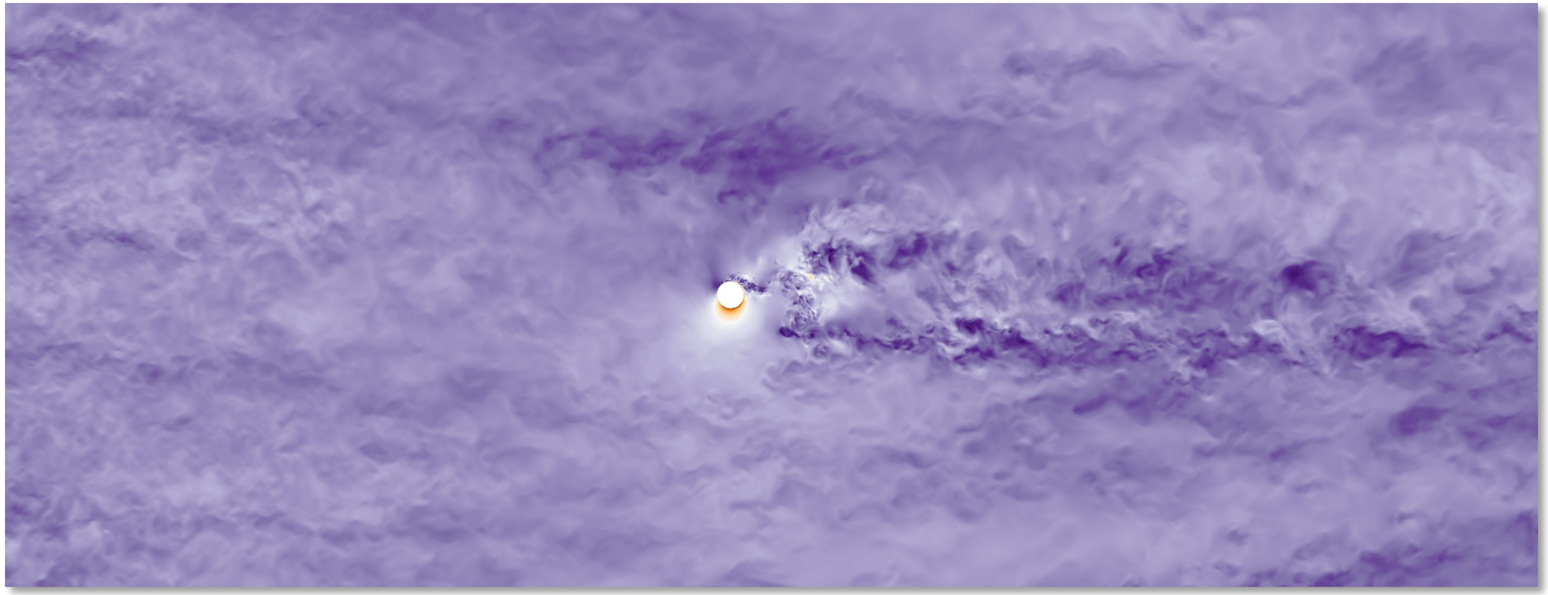


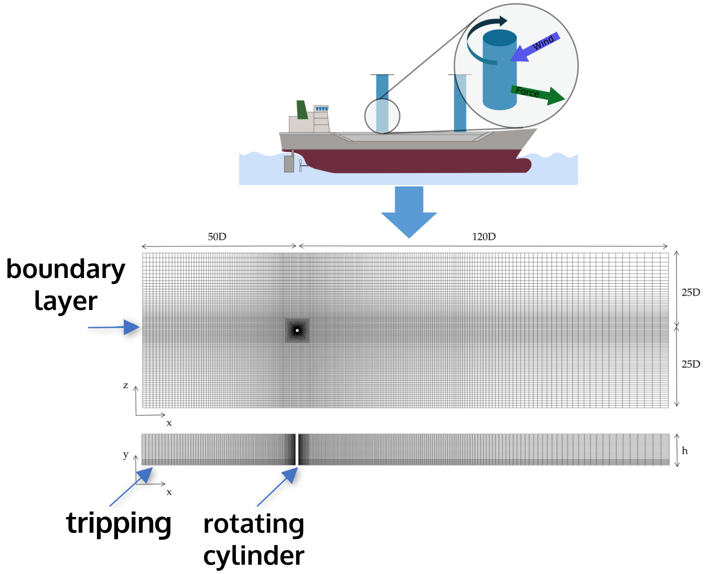


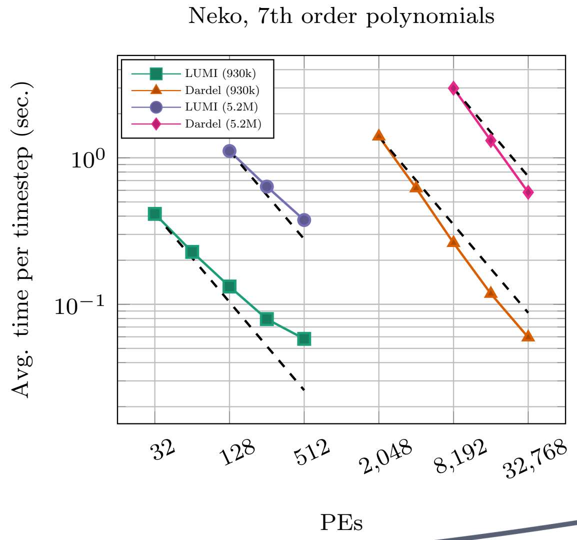
AMD CDNA2 Architecture white paper

Large-scale DNS of turbulence with applications in sustainable shipping

- DNS of the flow around a Flettner rotor at $Re = 30{,}000$ in a turbulent boundary layer
  - Two different meshes, 930k and 5.2M elements
  - 7th order polynomials (8 GLL points)
- Less than **two days** on LUMI (> two weeks on Dardel…)
  - Spanwise $C_l = 7.464$, experimental data measured 7~8

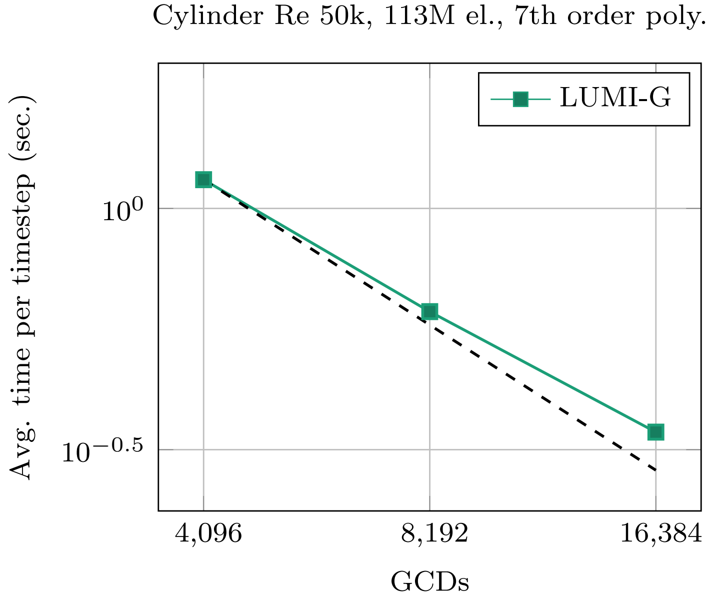






LUMI Hero Run

- Twelve-hour of exclusive access to the world's third-fastest supercomputer
- DNS of flow past a circular cylinder at Re = 50,000
  - 113M elements
  - 7th order polynomials (8 GLL points)
- Simulation restarted from prebaked low-order runs
  - Restart checkpoint: 453GB
  - Extrapolated to 7th order polynomials
  - Computed solution (snapshot): 1.5TB
- Preliminary results
  - Achieved close to 80% parallel efficiency
  - Using 20%, 40% and 80% of the entire machine



Summary

- High-order methods are **essential** on current HPC machines
  - More suitable for current hardware and improved accuracy for "free"
- The heterogenous HPC landscape is a **nightmare**
  - Find a suitable level of **abstraction**
  - Use the best tools, **mix langauges and programming models**
- Modern software engineering approaches to **ensure (performance) portability**
  - **Automate testing** across various architectures and programmig models
  - Deployment: **Spack,** verification & validation: **Reframe**

Jansson, N., Karp, M., Podobas, A., Markidis, S. and Schlatter, P., *Neko: A modern, portable, and scalable framework for high-fidelity computational fluid dynamics*. arXiv preprint arXiv:2107.01243, 2021.

Jansson, N., *Spectral Element Simulations on the NEC SX-Aurora TSUBASA*. In proc. HPCAsia 2021, 2021.

Karp, M., Podobas, A., Kenter, T., Jansson, N., Plessl, C., Schlatter, P. and Markidis, S., *A high-fidelity flow solver for unstructured meshes on field-programmable gate arrays: Design, evaluation, and future challenges*. In proc. HPCAsia 2022, 2022.

Karp, M., Jansson, N., Podobas, A., Schlatter, P., and Markidis, S., *Reducing Communication in the Conjugate Gradient Method: A Case Study on High-Order Finite Elements*. In proc. PASC 2022, 2022.

Karp, M., Massaro, D., Jansson, N., Hart, A., Wahlgren, J., Schlatter, P., and Markidis, S., *Large-Scale Direct Numerical Simulations of Turbulence Using GPUs and Modern Fortran*. International Journal of High Performance Computing Applications, Online First, 2023.

Funded by the European Union. This work has received funding from the European High Performance Computing Joint Undertaking (JU) and Sweden, Germany, Spain, Greece, and Denmark under grant agreement No 101093393.

Co-funded by the European Union

EuroHPC Joint Undertaking

Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European High Performance Computing Joint Undertaking (JU) and Sweden, Germany, Spain, Greece, and Denmark. Neither the European Union nor the granting authority can be held responsible for them.

CEEC

Centre of Excellence in Exascale CFD

Thank you for your attention!

CEEC

Centre of Excellence in Exascale CFD