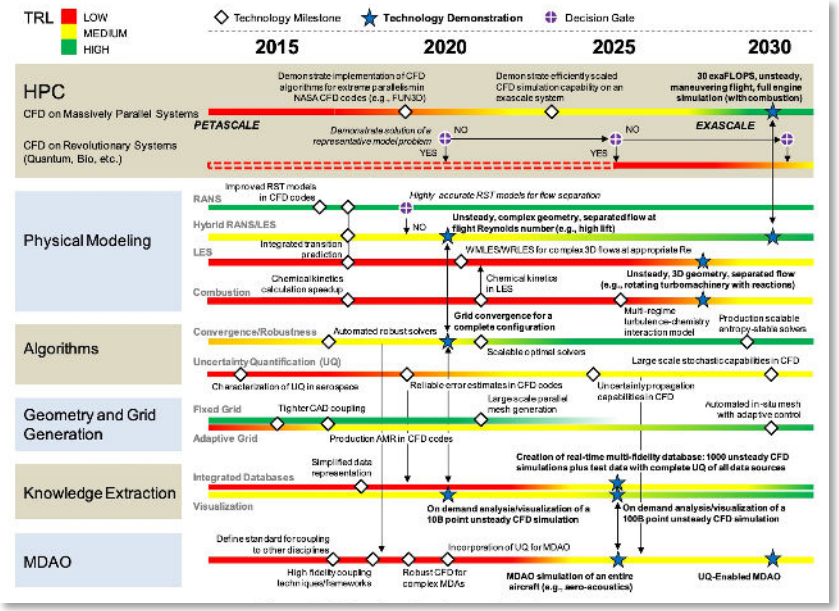
NEKO: A Modern, Portable, and Scalable Framework for High-Fidelity Computational Fluid Dynamics

Niclas Jansson, PDC Center for High Performance Computing

Introduction

About 10% of the energy use in the world is spent overcoming turbulent friction



No upper limit in fluid dynamics to the size of the systems to be studied via simulations

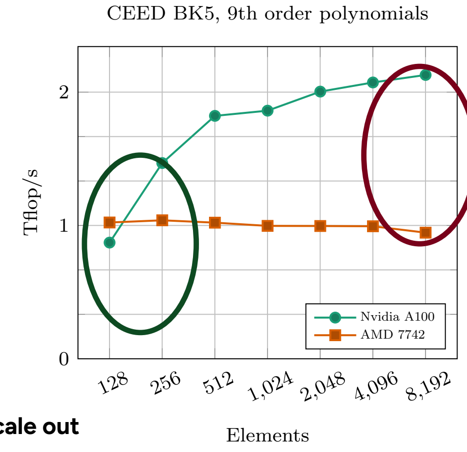
Computational Fluid Dynamics is one of the areas with a clear need and **great potential to reach exascale**

Introduction

- Exascale will require either **unreasonably large problem** sizes or **significantly improved efficiency** of current methods
  - Finite-Volume LES of a 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 379 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**

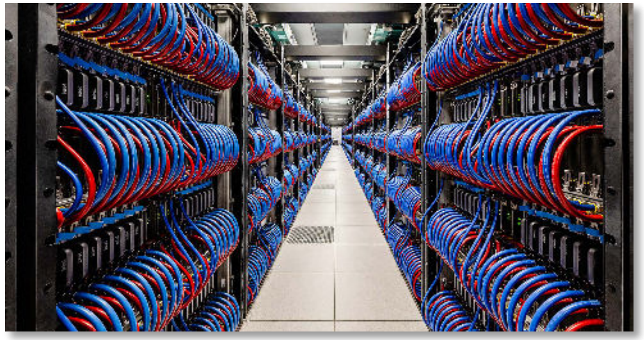


Accelerators works best with a lot of data!

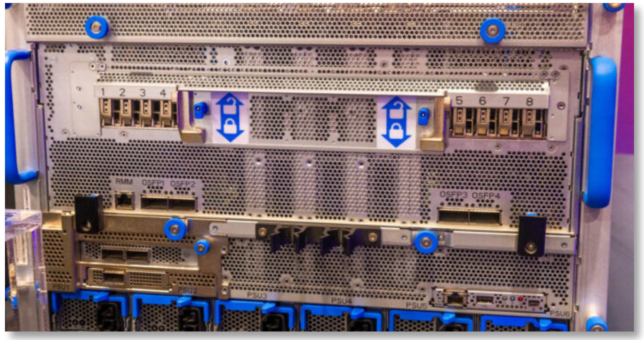
…but we rather scale out our problems…

Top500 List – November 2023

#1 Frontier: 1194 PFlop/s, AMD MI250X



#2 Aurora: 585 PFlop/s, Intel PVC



#3 Eagle: 561 PFlop/s, NVIDIA H100

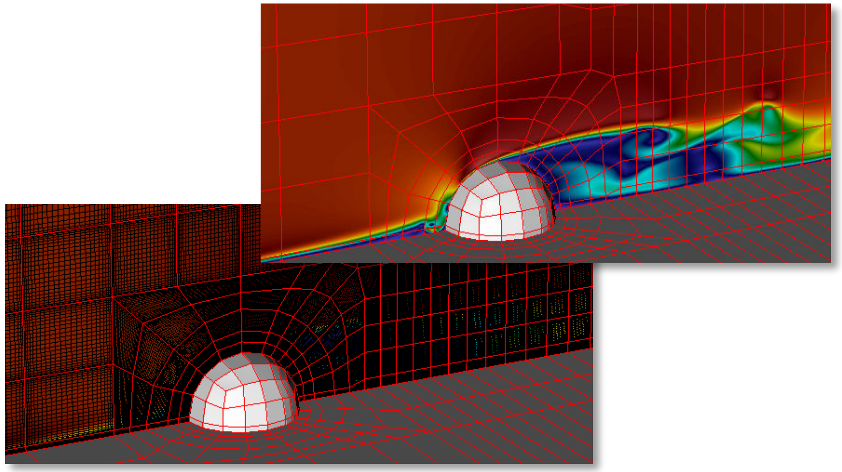
#4 Fugaku: 442 PFlop/s, Fujitsu A64FX

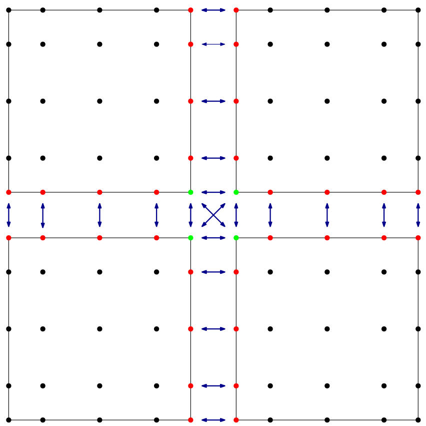
#5 LUMI: 378 PFlop/s, AMD MI250X

#6 Leonardo: 239 PFlop/s, NVIDIA A100

Spectral Elements

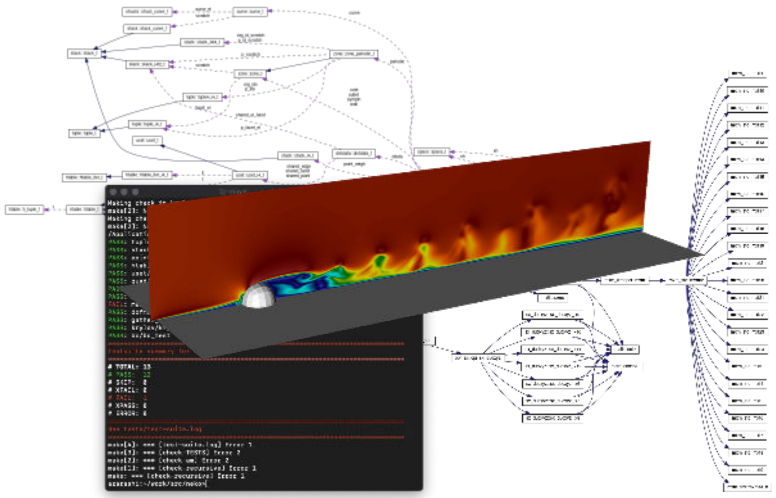
- Finite Elements with high-order basis functions
  - N-th order Legendre-Lagrange polynomials $l_i(\xi)$
  - Gauss-Lobatto-Legendre quadrature points ξ_i
  - Fast tensor product formulation
    - $u^e(\xi,\eta,\gamma) = \sum_{i,j,k}^{N} u_{i,j,k}^e l_i(\xi) l_j(\eta) l_k(\gamma)$
  - High-order at low 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 $A_L = \mathrm{diag}\{A^1, A^2, \ldots, A^E\}$ 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 and Q^T formed, only the action QQ^T is used
  - Matrix-vector product $w = Au \Rightarrow w_L = QQ^T A_L u_L$





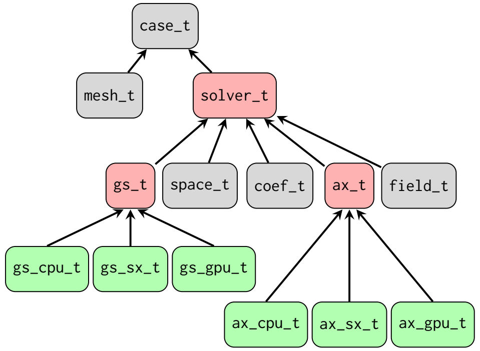
Portable Spectral Element Framework NEKO

- High-order spectral element flow solver
  - Incompressible Navier-Stokes equations
  - Matrix-free formulation, **small tensor products**
  - **Gather-scatter** operationst 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
```

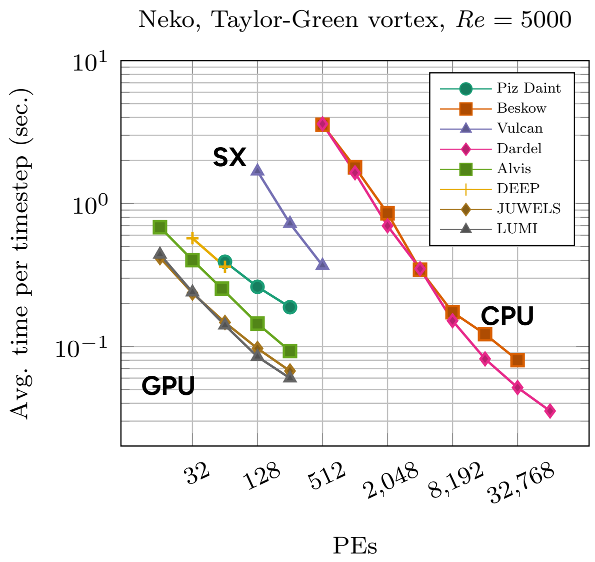


- Various hardware-backends
  - CPUs, GPUs down to exotic vector processors and FPGAs
    - **Device abstraction layer** for accelerators (CUDA/HIP/OpenCL)
  - Modern software engineering (pFUnit, ReFrame, **Spack**)

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

ExtremeFLOW/neko

www.neko.cfd



Device Abstraction Layer

How to interface Fortran with accelerators?

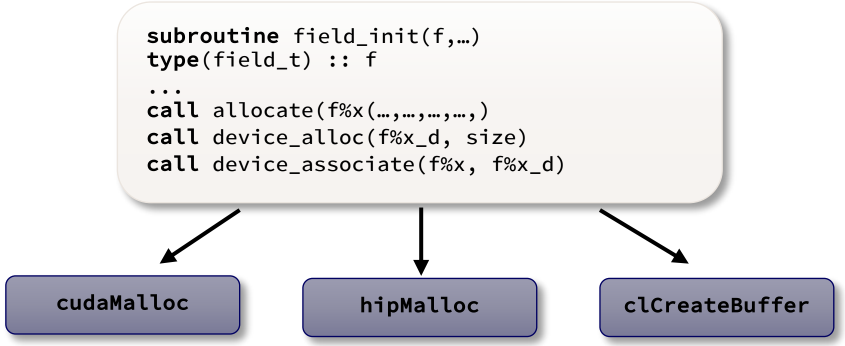
- Native CUDA/HIP/OpenCL implementation via C-interfaces
- Device pointers in each derived type

```
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
```

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

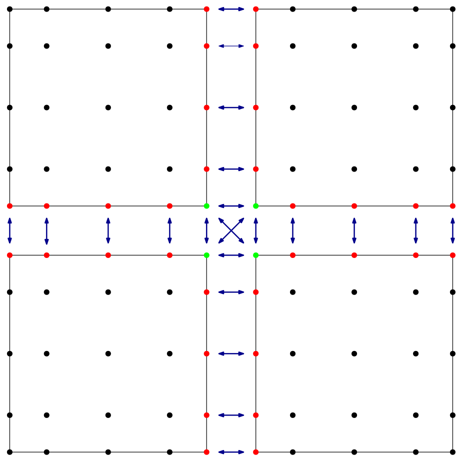
```
!> Enum @a hipError_t
enum, bind(c)
   enumerator :: hipSuccess = 0
   ...
end enum

!> 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
```

- Abstraction layer hiding memory management
- Hash table associating x with x_d
- Kernels invoked from the object hierarchy via C interfaces (Ax, vector ops)
  - **Wrapper functions** for each supported accelerator backend
  - **Templated** (CUDA/HIP) or **pre-processor macros** (OpenCL) for runtime parameters

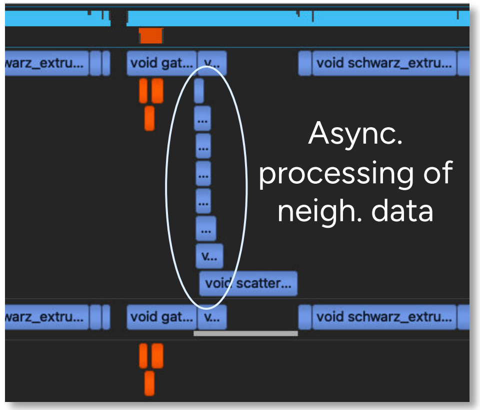


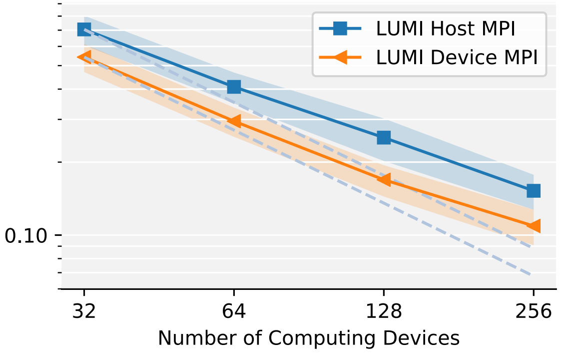
- **Auto/runtime tuning** based on polynomial order

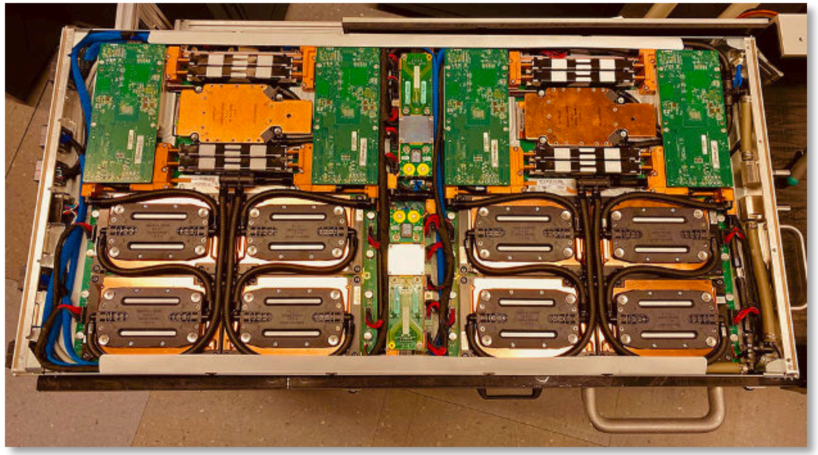
Gather-Scatter



- Uses indirect addressing and are (mostly) non-injective
- Topology aware optimisations
  - Facets (single neighbour), red points
    - Injective, **vectorizable** (always operating on **sorted** tuples)
  - Non facets (arbitrary number of neighbours), green points
    - **Cannot** be made injective, **not vectorizable** (small amount)
- Multiple levels of overlapping communication and computation
  - Overlapping with **non-blocking MPI** (device aware)
  - **Asynchronous** GPU kernels (neighbours in streams)
  - **Auto/runtime** tuning of all combinations



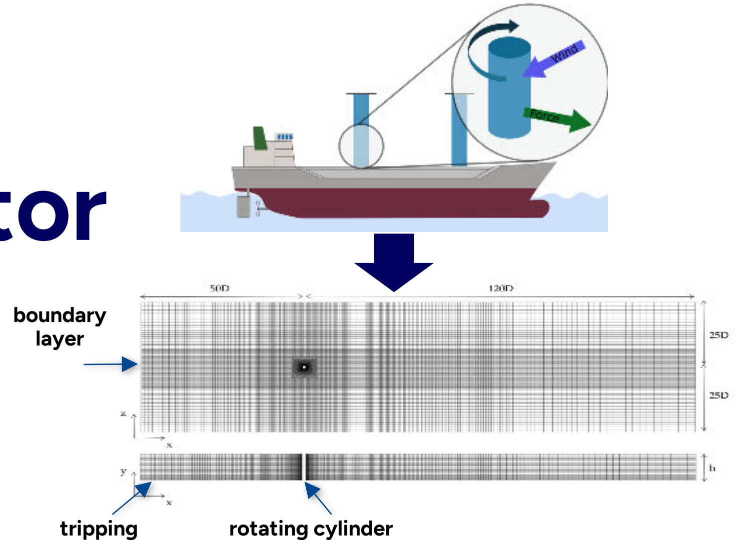


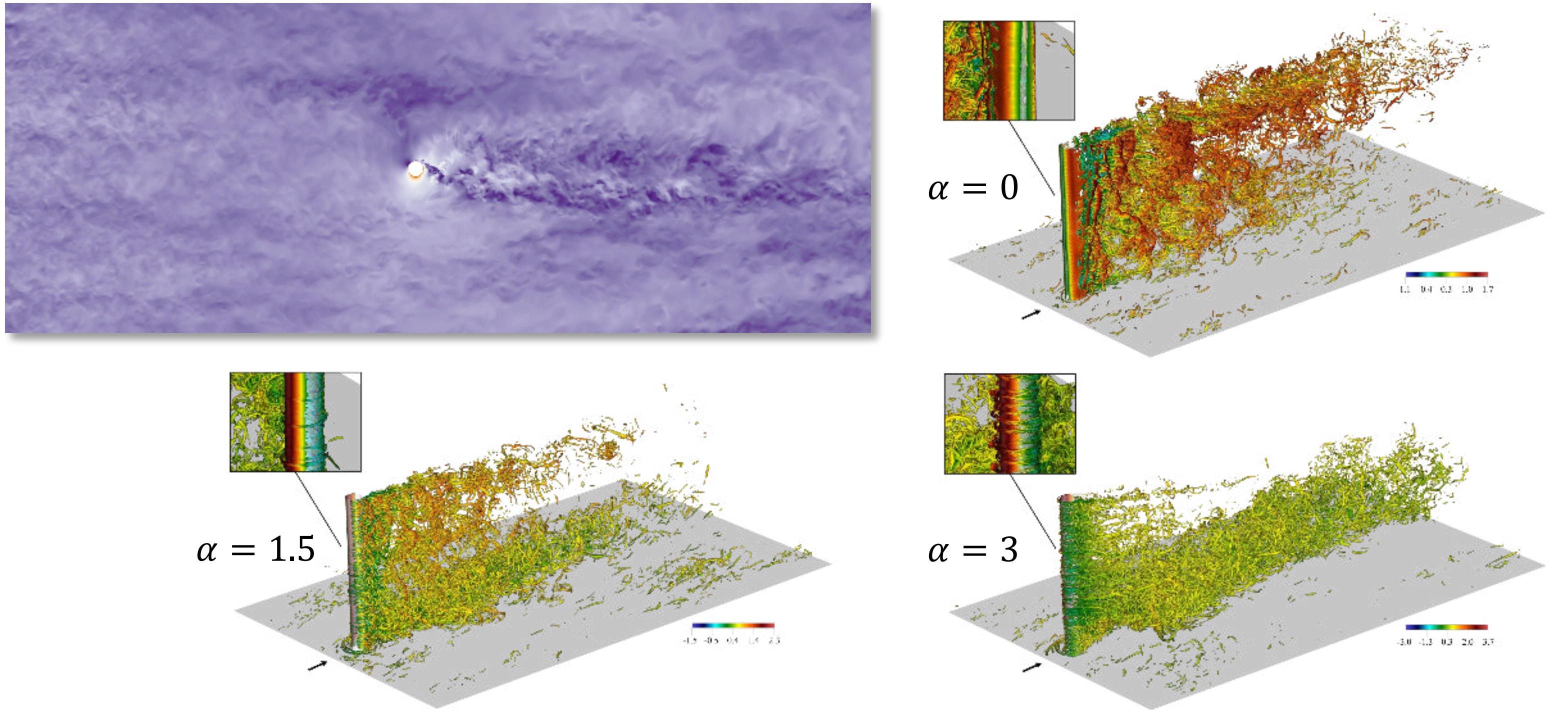


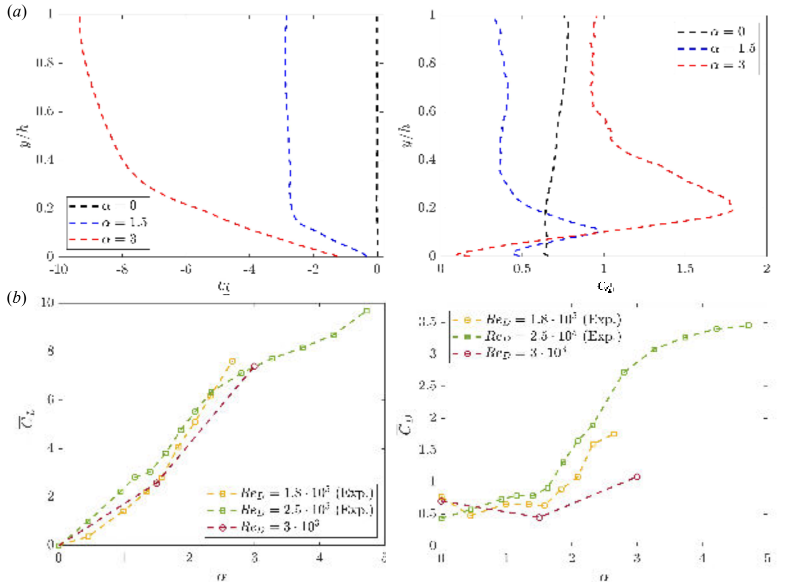
![KTH logo]

High-fidelity simulation of Flettner rotor

- DNS of the flow around a Flettner rotor at $Re_D = 3000$ in a turbulent boundary layer, for three different spinning ratios α
- **Less than two days** on Dardel-G (> two weeks on Dardel-C…)







Experimental data by Bordogna et al. , 2019.

D. Massaro et al., Direct numerical simulation of the turbulent flow around a Flettner rotor, *in review*, 2024.

Turbulent thermal convection

- Applications in nature and technology
  - From chip cooling, heat exchanges in power plants, to heat convection in the Earth's mantle and the sun.
- **Rayleigh-Bénard convection**: Canonical turbulent convection with fundamental open question: **Is there an ultimate regime,** i.e. anomalous scaling of Nusselt number (heat transfer) and Rayleigh number (buoyancy)?
  - Long-standing open issue in turbulence (Kraichnan 1962)
  - Difficult to conduct controlled experiments at high Rayleigh numbers $Ra > 10^{15}$
- Challenges with direct numerical simulations
  - **Large computational cost** due to resolution needs: $(H/\eta)^3 \sim Ra^{9/8}$
  - Numerical method with **minimal dissipative and dispersive errors** to capture and track small scales in time
  - Produces **unmanageable volumes of data**
  - **Long integration** times for steady state statistics
  - **Efficient implementation** on modern hardware

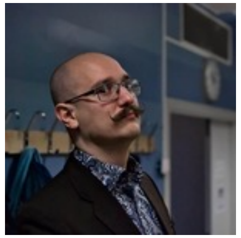
Illustration of the canonical problem at $Ra = 10^{13}$, iso-surfaces of temperature

KTH VETENSKAP OCH KONST

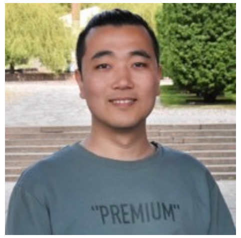
Team


Martin Karp


Adalberto Perez


Timofey Mukha


Yi Ju


Jiahui Liu


Szilárd Páll


Erwin Laure


Tino Weinkauf


Jörg Schumacher


Philipp Schlatter


Stefano Markidis

MAX PLANCK COMPUTING & DATA FACILITY MPCDF

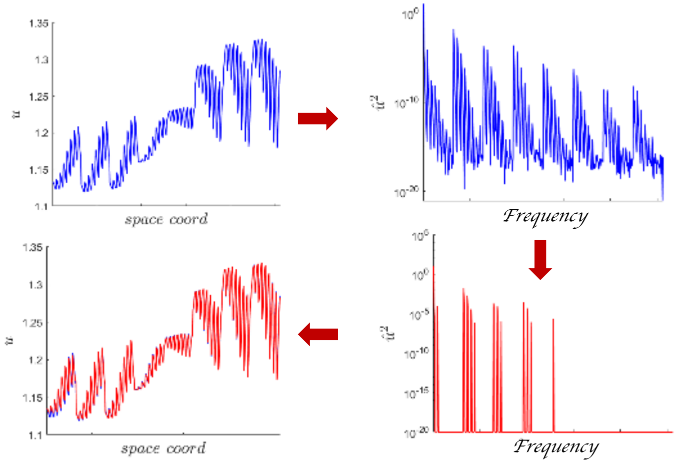
FAU Friedrich-Alexander-Universität Erlangen-Nürnberg

TECHNISCHE UNIVERSITÄT ILMENAU

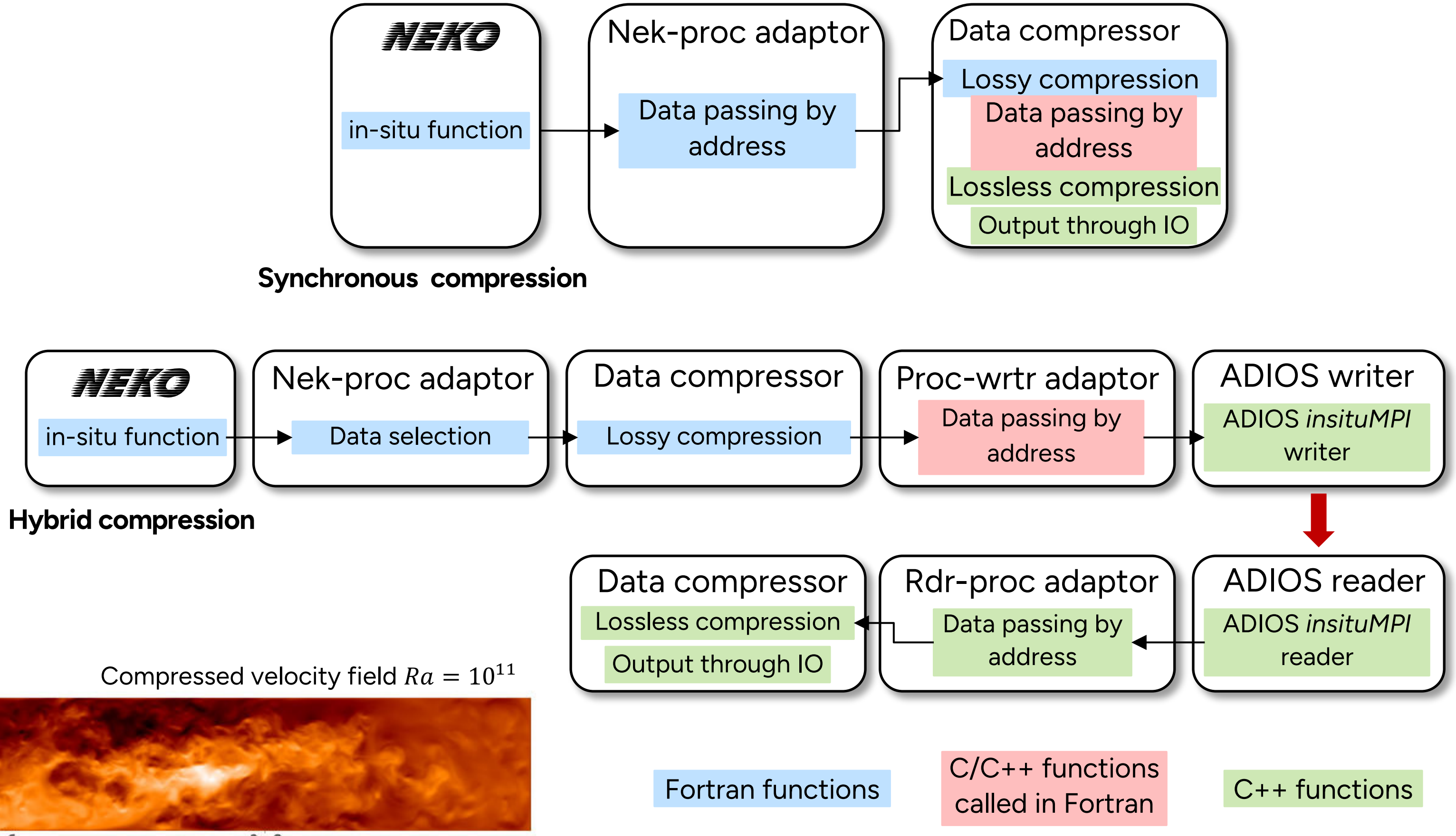
KTH VETENSKAP OCH KONST

Synchronous and Hybrid Data Compression

- **Lossy compression, physics-based method:**
  discard data not associated with the most energetic flow motions[1]

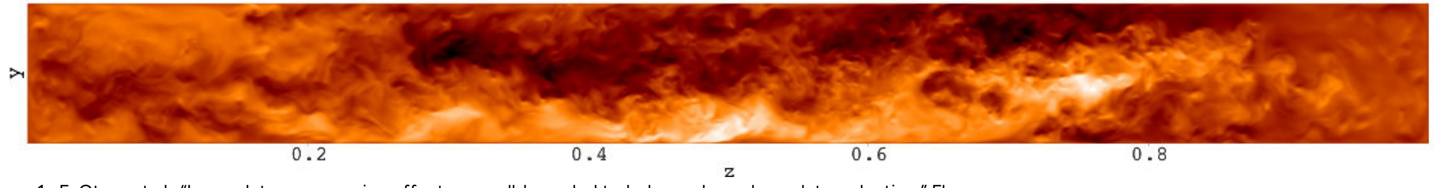


In-situ approach[2]



- **Lossless compression:**
  ADIOS2 operator with runtime configuration
- **97% data reduction** with a **relative error of 2.5%**

Compressed velocity field $Ra = 10^{11}$



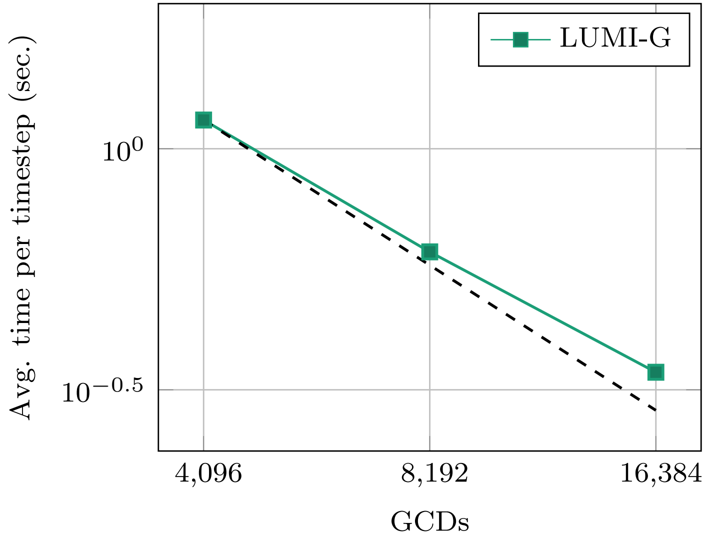
1: E. Otero et al., "Lossy data compression effects on wall-bounded turbulence: bounds on data reduction," Flow, Turbulence and Combustion, vol. 101, no. 2, pp. 365– 387, 2018.

2: Y. Ju et al., "In-Situ Techniques on GPU-Accelerated Data-Intensive Applications," eScience, 2023.

Performance Baseline

- Full machine runs towards the end of the LUMI-G pilot phase
- 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

Cylinder Re 50k, 113M el., 7th order poly.



Numerical Method $P_N - P_N$

- Time integration is performed using an implicit-explicit scheme (BDFk/EXTk)

$$\sum_{j=0}^{k} \frac{b_j}{dt} u^{n-j} = -\nabla p^n + \frac{1}{Re} \nabla^2 u^n + \sum_{j=1}^{k} a_j \left(u^{n-j} \cdot \nabla u^{n-j} + f^n \right)$$

with b_k and a_k coefficients of the implicit-explicit scheme, solving at time-step n

$$\Delta p^n = \sum_{j=1}^{k} a_j \left(u^{n-j} \cdot \nabla u^{n-j} + f^n \right)$$

$$\frac{1}{Re} \Delta u^n - \frac{b_0}{dt} u^n = \nabla p^n + \sum_{j=1}^{k} \left(\frac{b_j}{dt} u^{n-j} + a_j \left(u^{n-j} \cdot \nabla u^{n-j} + f^n \right) \right)$$

- Three velocity solves using CG with block Jacobi preconditioner (**fast**)
- One Pressure solve using GMRES with an additive overlapping Schwarz preconditioner (**expensive**)

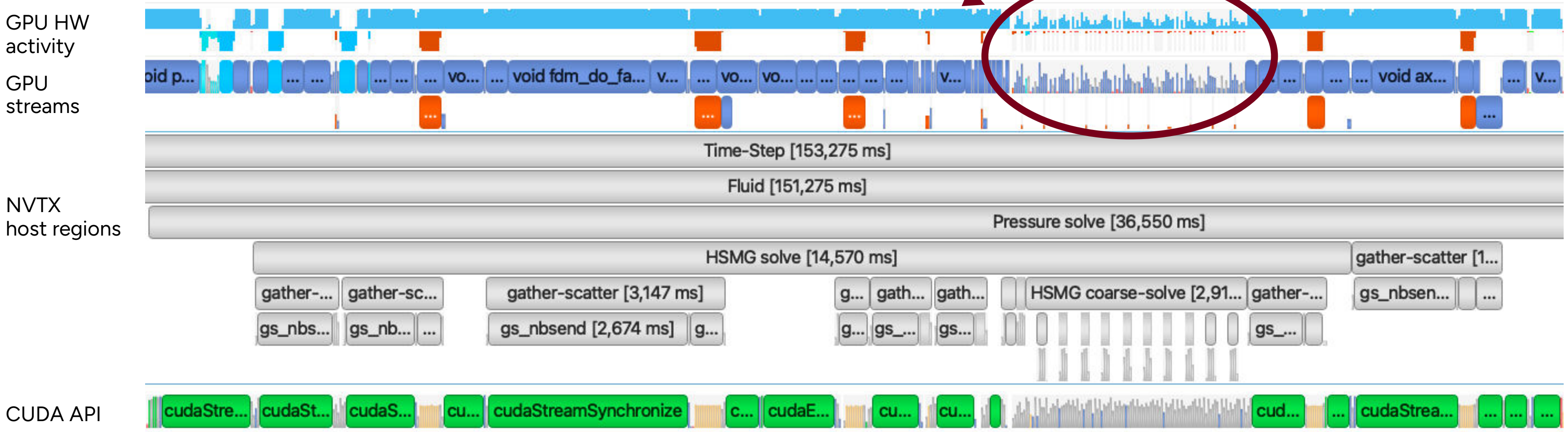
$M_0^{-1} = \underbrace{R_0^T A_0^{-1} R_0}_{\text{Coarse grid (linear elements)}} + \sum_{k=1}^{K} R_k^T \tilde{A}_k^{-1} R_k$, key is to have a **scalable coarse grid solver**

1. G.E. Karniadakis, M. Israeli, S.A. Orszag, High-order splitting methods for the incompressible Navier-Stokes equations, J. Comput Phys, 1991

Additive Schwarz Preconditioner on GPUs

- Coarse grid solved using an approximate Krylov solver
  - Preconditioned Pipelined Conjugate Gradient with a low, maximum iteration limit
- Low computational efficiency on GPUs
  - A_0 is on linear elements, too little data to keep the GPU busy.
  - Many small kernels, dominated by kernel launch latency

$$M_0^{-1} = R_0^T A_0^{-1} R_0 + \sum_{k=1}^{K} R_k^T \tilde{A}_k^{-1} R_k$$



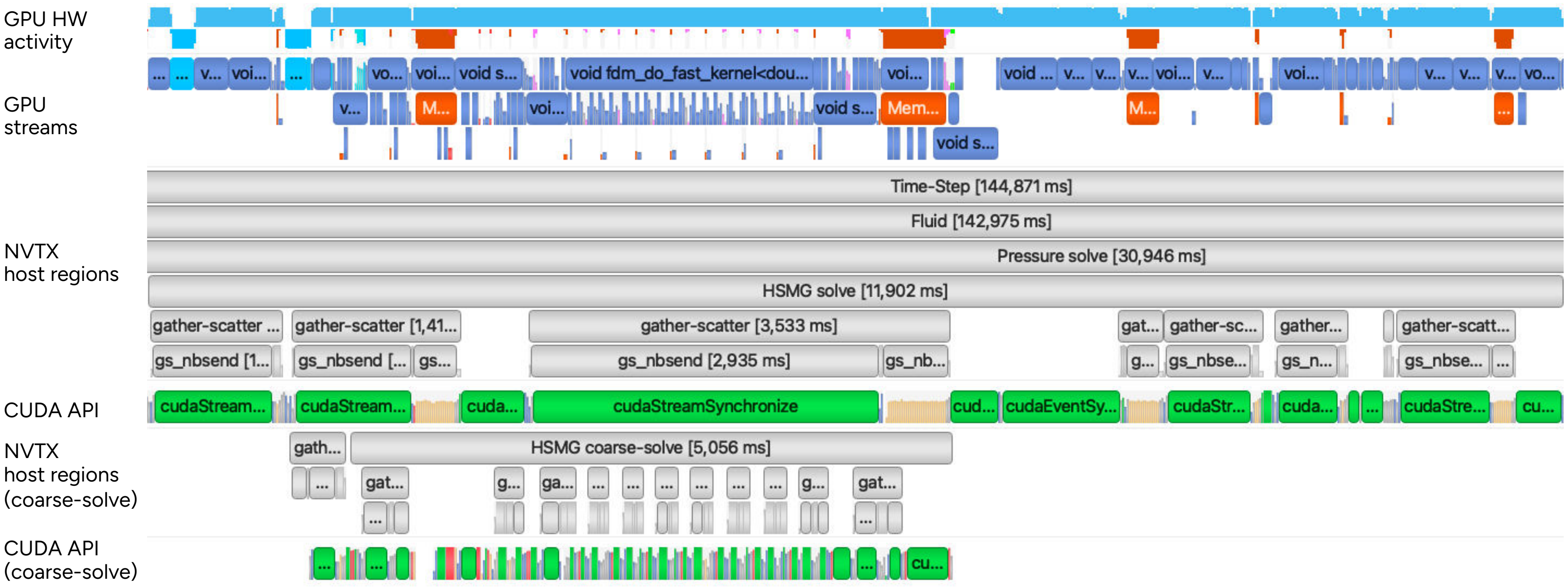
Task-decomposed Overlapped Preconditioner

- Exploit available **task-parallelism**
  - Launch the left and right part of M_0^{-1} in parallel on the device
  - Launch independent work in parallel from **different threads** in an OpenMP region
  - Launch tasks in **separate streams** to allow overlap and increase GPU utilization
  - Maximise kernel overlap using **stream priority** to ensure progress in both stream

Thread 0 | Thread 1

$$M_0^{-1} = R_0^T A_0^{-1} R_0 + \sum_{k=1}^{K} R_k^T \tilde{A}_k^{-1} R_k$$

Stream 1 | Stream 2



Performance Results

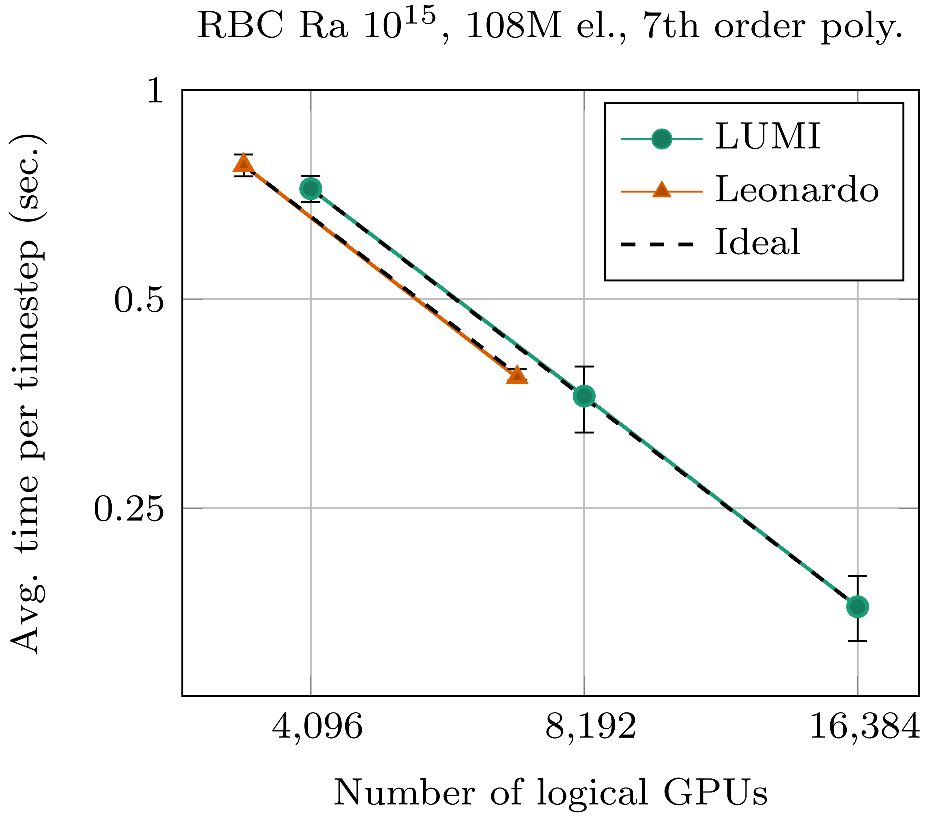
- Performance measurements on two of the EuroHPC-JU pre-exascale supercomputers **LUMI** and **Leonardo**
- Experiments were performed between
  - March–April 2023 on LUMI
  - April 2023 on Leonardo (pre-production)
- RBC in a cylinder with aspect ratio 1:10
  - $Ra = 10^{15}$
  - 108M elements, 7^{th} order polynomials
  - 37B unique grid points and more than 148B degrees of freedom
- Strong Scalability
  - Average time per timestep (after transient)
- One MPI rank per logical GPU
  - One rank per GCD (AMD)
  - One rank per device (Nvidia)

System	LUMI	Leonardo
Computing device	AMD MI250X	Nvidia A100 (custom)
Peak Tflop FP64/s	47.9 (95.7 Matrix)	11.2 (22.4)
Peak BW/s	3300	1640
No. devices	10240	13824
Interconnect	HPE Slingshot 11 200 GbE NICs (4x200 Gb/s)	Nvidia HDR 2x(2x100 Gb/s)
MPI	Cray MPICH 8.1.18	OpenMPI 4.1.4
Compiler	CCE 14.0.2	GCC 8.5.0
GPU Driver	5.16.9.22.20	520.61.05
CUDA/ROCm	ROCm 5.2.3	CUDA 11.8

Performance Results

- Close to perfect parallel efficiency on both LUMI and Leonardo
- Close to perfect parallel efficiency with less than 7000 elements per logical GPU
- Significantly reducing the smallest required problem size for strong scalability limits
- Improvements mainly due to the new overlapped pressure preconditioner

ACM Gordon Bell Prize Finalist 2023



99% confidence intervals is illustrated as error bars

N. Jansson et al., Exploring the Ultimate Regime of Turbulent Rayleigh-Bénard Convection Through Unprecedented Spectral-Element Simulations, SC '23: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 2023.

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 languages and programming models
- Modern software engineering approaches to ensure portability
  - Verification & validation
- Task-decomposed overlapped pressure preconditioner
  - Expressing more of the available concurrency of the application
  - Key ingredient to achieve good strong scalability on LUMI and Leonardo

www.neko.cfd

KTH

VETENSKAP
OCH KONST