# Exploring the Ultimate Regime of Turbulent Rayleigh–Bénard Convection Through Unprecedented Spectral-Element Simulations


Niclas Jansson
KTH Royal Institute of Technology
Stockholm, Sweden
njansson@kth.se

Martin Karp
KTH Royal Institute of Technology
Stockholm, Sweden
makarp@kth.se

Adalberto Perez
KTH Royal Institute of Technology
Stockholm, Sweden
adperez@kth.se

Timofey Mukha
KTH Royal Institute of Technology
Stockholm, Sweden
tmu@kth.se

Yi Ju
Max Planck Computing and Data Facility
Garching, Germany
yju@rzg.mpg.de

Jiahui Liu
KTH Royal Institute of Technology
Stockholm, Sweden
jiahuil@kth.se

Szilárd Páll
KTH Royal Institute of Technology
Stockholm, Sweden
pszilard@kth.se

Erwin Laure
Max Planck Computing and Data Facility
Garching, Germany
erwin.laure@mpcdf.mpg.de

Tino Weinkauf
KTH Royal Institute of Technology
Stockholm, Sweden
weinkauf@kth.se

Jörg Schumacher
Technische Universität Ilmenau
Ilmenau, Germany
joerg.schumacher@tu-ilmenau.de

Philipp Schlatter*
Friedrich-Alexander-Universität (FAU) Erlangen–Nürnberg
Erlangen, Germany
philipp.schlatter@fau.de

Stefano Markidis
KTH Royal Institute of Technology
Stockholm, Sweden
markidis@kth.se



## ABSTRACT

We detail our developments in the high-fidelity spectral-element code Neko that are essential for unprecedented large-scale direct numerical simulations of fully developed turbulence. Major innovations are modular multi-backend design enabling performance portability across a wide range of GPUs and CPUs, a GPU-optimized preconditioner with task overlapping for the pressure-Poisson equation and in-situ data compression. We carry out initial runs of Rayleigh–Bénard Convection (RBC) at extreme scale on the LUMI and Leonardo supercomputers. We show how Neko is able to strongly scale to 16,384 GPUs and obtain results that are not possible without careful consideration and optimization of the entire simulation workflow. These developments in Neko will help resolving the long-standing question regarding the **ultimate regime** in RBC.


*Also with KTH Royal Institute of Technology.




*SC '23, November 12–17, 2023, Denver, CO, USA*



ACM ISBN 979-8-4007-0109-2/23/11.
https://doi.org/10.1145/3581784.3627039

**ACM Reference Format:**
Niclas Jansson, Martin Karp, Adalberto Perez, Timofey Mukha, Yi Ju, Jiahui Liu, Szilárd Páll, Erwin Laure, Tino Weinkauf, Jörg Schumacher, Philipp Schlatter, and Stefano Markidis. 2023. Exploring the Ultimate Regime of Turbulent Rayleigh–Bénard Convection Through Unprecedented Spectral-Element Simulations. In *The International Conference for High Performance Computing, Networking, Storage and Analysis (SC '23), November 12–17, 2023, Denver, CO, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3581784.3627039


## 1 JUSTIFICATION FOR ACM GORDON BELL PRIZE

We present a workflow to enable the resolution of a long-standing open issue in turbulence regarding the **ultimate regime** in Rayleigh–Bénard convection. Scaling and proof of concept on thousands of Nvidia/AMD GPUs puts answering this question within reach of modern computational science with regards to time-to-solution, storage requirements, and pre/post-processing.

## 2 PERFORMANCE ATTRIBUTES

| Performance attributes | Our submission |
|---|---|
| Category of achievement | Scalability, time-to-solution |
| Type of method used | Explicit/Implicit |
| Results reported | Whole application |
| Precision reported | Double-precision |
| System scale | Full-scale system |
| Measurements mechanism | Timers |

## 3 OVERVIEW OF THE PROBLEM

In fluid dynamics, the study of turbulent thermal convection is an area of both fundamental and applied importance, with ubiquitous applications to diverse phenomena in nature and technology. In the latter category, one can name thermal convection in chip cooling devices, heat exchangers in power plants, and energy-efficient indoor ventilation. Important natural phenomena include convection in the Earth's atmosphere, oceans and the interior of its mantle, as well as astronomical applications, such as convection in the Sun and other stars [24]. The canonical Rayleigh–Bénard convection (RBC), as illustrated in Fig. 1, is commonly used to study these turbulent flows, and it can be assessed in a controlled manner, yet has enough complexity to contain the key features of turbulence and convective heat transfer. As RBC is integral to understanding the transfer mechanisms between heat and momentum, this problem has been studied for decades. In particular, as early as 1962 Robert Kraichnan made several predictions that are yet to be verified [17]. As heat transfer is typically measured by the Nusselt number $Nu$, and depends strongly on the driving force of the convection given by the Rayleigh number $Ra$, the question is in exactly which way $Nu$ depends on $Ra$ in the limit of large $Ra$. Kraichnan conjectured that, as the boundary layers forming on the heated plates undergo laminar–turbulent transition, the dependency foregoes the classical scaling of $Nu \sim Ra^{1/3}$ and asymptotically converges towards the so-called **ultimate regime**, characterized by the scaling $Nu \sim Ra^{1/2}$. However, more than 60 years later, there is still no conclusive evidence whether and in case when the ultimate regime may be reached [7, 19, 24]. Examining the scaling at high enough $Ra$ would, therefore, end this open debate in turbulence research and be a major breakthrough in our understanding of some of the core geo- and astrophysical flows that occur at various scales in the universe.

In order to understand RBC, the study of flow in cylindrical cells has gained a lot of attention in recent years, both experimentally and computationally. One of the main issues that arise in experiments, however, is that at high enough $Ra$ they show an extreme sensitivity to the exact experimental conditions [19], including homogeneity of the boundaries, temperatures and other disturbances. Consequently, Direct Numerical Simulation (DNS) of RBC in slender cells has been the focus of recent research [9]. These simulations are among the largest DNS ever carried out on conventional CPUs, and the produced data volume quickly becomes unmanageable. Still, the largest $Ra$ simulated is $Ra = 10^{15}$, which is just about when the transition to the ultimate regime is expected to occur. In addition, the focus was on a slender cell with aspect ratio 1:10, meaning that the walls of the cell may delay the transition to fully turbulent boundary layers at lower $Ra$. In order to carry out simulations at higher $Ra$ and for larger aspect ratios, the issues of computational time and data management must be resolved via a suitable simulation workflow. The development of such a workflow is a major contribution in this work, where we accommodate modern computer architectures, reduce the data volume of the output and leverage in-situ techniques to be able to investigate the true nature of the ultimate regime.

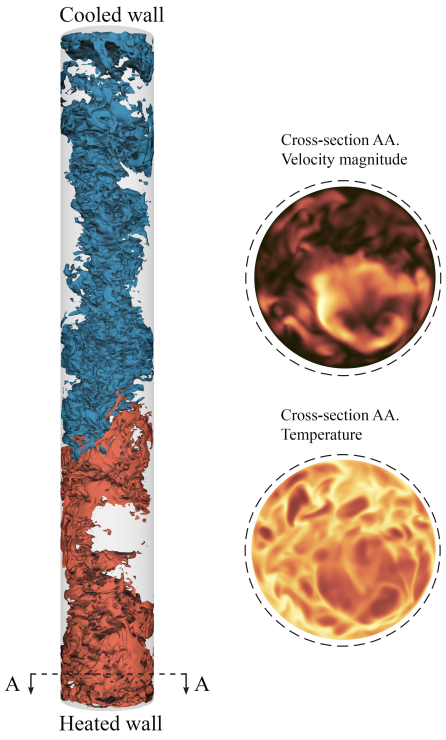


**Figure 1: Visualization of RBC at moderate $Ra = 10^{11}$ to illustrate the canonical physical problem. The flow in a cylindrical container is heated from below and cooled from the top, thus driving convective turbulence. Red and blue illustrate warm and cold fluid, respectively. We also show cross-section AA close to the heated bottom wall. The upper cross-section shows the velocity magnitude and the bottom one the temperature field.**

## 4 CURRENT STATE OF THE ART

### 4.1 Governing Equations

The incompressible Navier–Stokes equations are a highly nonlinear set of coupled partial differential equations. Commonly referred to as one of the most important unsolved problems of modern science, solutions for turbulent flows can only be obtained via numerical approximation, i.e. DNS, at a computational cost, which is in many practical cases beyond the capabilities of modern supercomputers. In this work, we consider the Navier–Stokes equations coupled with a scalar temperature field under the Boussinesq approximation:

$$\begin{aligned} \nabla \cdot \mathbf{u} &= 0, \\ \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} &= -\nabla p + \sqrt{\frac{Pr}{Ra}}\nabla^2 \mathbf{u} + T\mathbf{e}_z \\ \frac{\partial T}{\partial t} + (\mathbf{u} \cdot \nabla)T &= \frac{1}{\sqrt{RaPr}}\nabla^2 T, \end{aligned} \tag{1}$$

where $T$ is the temperature field, $\mathbf{u}$, $p$ the instantaneous velocity and pressure, and $Pr$ the Prandtl number, with $Pr = 1$ in our considered case. The quantities are non-dimensionalised by the cylinder height

$H$, the free-fall velocity $U_f$ and the temperature difference between the top and bottom plate $\Delta T$, see e.g. [19] for more details. The reason for the large computational cost is that turbulent flow is inherently multiscale, which necessitates resolving vortical motions of size ranging from $\sim H$ down to the Kolmogorov length scale $\eta$, at which they are dissipated. Typically, $H/\eta \sim Ra^{3/8}$, and as such, the number of grid points quickly becomes unmanageable even for moderately turbulent flows. Furthermore, the numerical method used to discretize the system (1) has to exhibit minimal dissipative and dispersive errors to accurately capture and track the small scales in time. In addition, the numerics need to be able to handle complex curved geometries, and allow for efficient implementation on modern hardware.

### 4.2 Numerical Methods

For high-fidelity turbulence simulations, various types of high-order discretization methods serve as the primary tools due to their favorable properties [23]. In particular, finite difference and Fourier methods have been considered for canonical cases of RBC [16]. However, both suffer from limitations with regards to the geometry of the computational domain, making them unsuitable as candidates for a general-purpose flow simulation framework, including cylindrical RBC.

The spectral-element method (SEM) provides an alternative. This approach falls into the category of high-order finite element methods, thus allowing the computational mesh to have a complex topology, while at the same time retaining the beneficial properties of high-order discretization, and data locality. In particular, the SEM can be formulated in a matrix-free fashion, leading to a high operational intensity and only unit-depth communication between elements. The SEM has been used for simulating a large variety of laminar, transitional and turbulent flows and has, due to the method's design, shown exemplary scaling. In particular, it has been used to carry out the above-mentioned RBC simulation at $Ra = 10^{15}$, leading to high-fidelity predictions of the Nusselt number as well as other instantaneous and integral quantities of interest. Those simulations used the full size of the Blue-Gene Supercomputer at the Argonne Leadership Supercomputing Facility and the spectral-element code Nek5000 [9]. Nek5000 has been shown to scale up to millions of CPU cores and is currently the main workhorse for spectral-element simulations on CPU machines.

However, most machines, as we approach exascale, are incorporating GPU accelerators as their main source of computational power. As such, the SEM has been carried over to GPUs through dedicated efforts such as the present approach, in the code Neko [11]. A more direct development of Nek5000 to GPUs is NekRS [3] relying on part of the original code structure. What distinguishes Neko is that it incorporates a new codebase in modern Fortran, and does not rely on the original solver Nek5000. This leads to granular control of memory allocation, modularity and extensibility via object-oriented design, which includes adding support for new compute architectures. However, the basic solution approach of Nek5000, Neko and NekRS is very similar, and shares some of the routines.

### 4.3 Direct numerical simulation of RBC

As mentioned, the highest $Ra$ that has been simulated to date is $10^{15}$ in a slender cell with aspect ratio 1:10 [9]. It is, however, conjectured that the aspect ratio plays a role in the transition to the ultimate regime [1], and experiments with larger aspect ratios might be necessary to investigate the onset of the ultimate regime, even though the flow already at $Ra = 10^{15}$ with aspect ratio 1:10 exhibits all characteristics of a fully turbulent flow. In preparation to settle this debate, we are in the process of carrying out a series of runs of RBC at high $Ra$ and different aspect ratios, considerably pushing forward the state-of-the-art with regards to RBC simulation.

## 5 INNOVATIONS REALIZED

In this work, we leveraged the capabilities of modern computer architectures for the simulation workflow of Rayleigh–Bénard convection, using a number of innovations both in pre-/post-processing, and simulation efficiency.

### 5.1 Performance Portability

To address the exascale computing challenge and to enable performant high-fidelity fluid-dynamics simulations across various platforms, we have developed Neko, a portable framework for high-order spectral-element-based simulations, focusing on incompressible flow simulations. The framework is implemented in Fortran 2008 and adopts a modern object-oriented approach, allowing for multi-tier abstractions of the solver stack and facilitating various hardware backends[10, 11, 14, 15]. Using Fortran as the language of choice instead of recently more popular languages, such as C++ or Python, might at first seem like an odd choice, particularly for developing a new code. However, Neko has its roots in the spectral-element code Nek5000 [22] from UChicago/ANL, introduced in the mid-nineties, tracing its origins to MIT's older NEKTON 2.0. Furthermore, research groups at KTH have extensively used the scalable Nek5000 and also further developed its Fortran 77 codebase, thus leaving a non-negligible trace of more than thirty years of verified and validated Fortran code, which, if rewritten into, e.g., C++ would have to go through a very expensive and time-consuming revalidation and reverification process before it could be used in production. Therefore, by using modern Fortran, already validated Fortran 77 kernels can directly be integrated into Neko, with only a minimal revalidation process.

Neko integrates in time the incompressible Navier–Stokes equations, ensuring single-core/accelerator efficiency via fast tensor product operator evaluations. For high-order methods, assembling either the local element matrix or the full stiffness matrix is prohibitively expensive. Therefore, a key to achieving good performance in spectral-element methods is to consider a matrix-free formulation, where one always works with the unassembled matrix on a per-element basis. Gather–scatter operations ensure the continuity of functions on the element level, operating on both intra-node and inter-node element data. Currently, Neko uses MPI for inter-node parallelism and parallel I/O for production runs, but one-sided communication options such as Coarray Fortran based gather-scatter kernels are under development.

When designing a flexible and maintainable framework for computational science, a major issue is finding the right level of abstraction. Too many levels might degrade performance, while too few results in a code base with many specialized kernels at a high maintenance cost. The weak form of the equation used in the spectral-element method allows Neko to recast equations in the form of the abstract problem to keep the abstractions at the top level and reduce the amount of platform-dependent kernels to a minimum. In Neko, this is realized using abstract Fortran types, with deferred implementations of required procedures. For example, to allow for different formulations of a simulation's governing equations, Neko provides an abstract type, defining the abstract problem's matrix-vector product. The type comes with a deferred procedure "compute" that would return the action of multiplying the stiffness matrix of a given equation with a vector. In a typical object-oriented fashion, whenever a routine needs a matrix-vector product, it is always expressed as a call to "compute" on the abstract base type and never on the actual concrete implementation. Abstract types are all defined at the top level in the solver stack during initialization and represent large, compute-intensive kernels, thus reducing overhead costs associated with the abstraction layer. Furthermore, this abstraction also accommodates the possibility of providing tuned matrix-vector products for specific hardware, only providing a particular implementation of "compute" without having to modify the entire solver stack.

However, regardless of the abstraction, modern Fortran does not provide a built-in method for interfacing with accelerators. Despite a popular choice to rely on vendor-specific solutions, domain-specific languages (DSL) or directives-based approaches when porting Fortran codes to accelerators, due to portability issues and reduced performance, a decision has been made to not use these solutions (e.g. CUDA Fortran, OpenACC). Instead, Neko uses a device abstraction layer to manage device memory, data transfer and kernel launches from Fortran. Behind this interface, Neko calls the native accelerator implementation written in, e.g., CUDA, HIP or OpenCL, with all data fully GPU-resident throughout a simulation. The interface also allows for vendor-specific optimizations, with auto-tuning of key kernels for sustained performance while keeping most of the solver written in a hardware-neutral yet performant way. Furthermore, if present, device-aware MPI is also exploited to minimize necessary data movement between host and device in communication kernels. Device-initiated communication, e.g. SHMEM is also under development.

## 5.2 Compression and In-Situ Data Analysis

The study of physical flow phenomena such as RBC through Computational Fluid Dynamics transcends the computation step alone and is a workflow. Pre-processing steps such as meshing are always required, and post-processing via modal decomposition such as Proper Orthogonal Decomposition [8] or the calculation of flow statistics is desirable. Thus, data management is a relevant issue, especially in view of increased efforts to produce suitable databases for training of data-driven models [2] and related methods.

To correctly study the flow dynamics in any post-processing technique, it is necessary to sample the instantaneous flow frequently, and for a long enough period, such that both the slow and fast dynamics are captured with sufficient accuracy and resolution. Since each flow sample for extensive turbulence simulations poses considerable storage requirements, post-processing the data generated becomes a bottleneck.

A solution to this limitation is to utilize the computational infrastructure available to the solver at run-time to facilitate the subsequent handling of the data by performing in-situ data analysis and transformation directly on the compute nodes. Several strategies of in-situ analysis were initially studied in Nek5000 on CPUs [12] but are now expanded to GPUs with the Neko framework. The most basic and perhaps crucial data transformation is data compression, which directly addresses the storage bottleneck. The performance of lossless compressors is typically limited by the Shannon entropy [25], i.e., the information content or variance of the data. Given that data produced by turbulence has a naturally high entropy due to the chaotic nature of the velocity fields in that regime, lossless compression alone is typically not sufficient to efficiently reduce the storage requirements. For this reason, the compression algorithm implemented in Neko relies on a lossy step [21] specifically designed for spectral-element data: It decreases the variance in the data set by performing an $L^2$ projection of the data set $u(x)$ to an orthogonal polynomial basis $\phi_i(x)$ such that:

$$u(x) = \sum_{i=0}^{N} \hat{u}_i \phi_i(x). \tag{2}$$

The $\hat{u_i}$ are the coefficients of the modal representation of $u(x)$, and exhibit a much lower variance for a correct choice of basis and can be further truncated to reduce the storage requirements. In Neko, we transform the field, truncate it and encode it through a lossless compression algorithm synchronously at run time to finally compress the data.

As a supporting tool, we choose ADIOS2 [6] to manage I/O operations during data compression, since it is an efficient instrument to control the information produced by the multiple MPI processes active during the simulation. An added benefit of this tool is that it naturally includes engines for performing asynchronous in-situ tasks. This means that while the main simulation is running on the GPUs, the data can be easily streamed to a data processing routine, running on the mostly unused CPUs of the compute nodes to post-process the data online, taking advantage of the fully allocated compute power. This is, in fact, the approach that we follow to perform streaming Proper Orthogonal Decomposition in parallel [18, 26], using a data processor written in Python. This approach is a promising addition, as the integration to Python allows for the potential to use large simulations to directly incorporate data-driven model training while avoiding storing the large amounts of data typically needed for this purpose, with a low impact on the simulation performance.

## 5.3 Pressure Preconditioner

Efficient preconditioning for the pressure is essential in incompressible fluid dynamics, as the Poisson equation that arises as a consequence of incompressibility is the main source of stiffness when computing a solution. We use a two-level additive overlapping Schwarz mutligrid method [4, 5] combined with a coarse grid

solver to precondition the linear system,

$$M_0^{-1} = R_0^T A_0^{-1} R_0 + \sum_{k=1}^{K} R_k^T \tilde{A}_k^{-1} R_k, \quad (3)$$

for a general $k$-level formulation, where $R_k$ and $R_k^T$ are the restriction and prolongation operations to move between different grid levels. The coarse grid problem $A_0$, on linear elements, is solved for using an approximate Krylov solver, a preconditioned Conjugate Gradient method, with a fixed number of iterations ($\approx$ 10) and an element-wise block Jacobi preconditioner. This has proven to be both an effective and scalable preconditioner on both CPUs and GPUs [14]. However, in terms of computational efficiency, the coarse grid's smaller problem size will reduce the preconditioners ability to fully utilize the GPU.

It is possible to decouple the coarse grid solve (first part of the right-hand side in (3)) from the rest of the multigrid solver which allows computing the two parts in parallel. One potential approach is to solve the coarse grid problem on the CPU and better utilize the available resources. However, transferring data to and from the GPU memory quickly becomes the limiting factor for different configurations, in particular for architectures where the interconnect is directly connected to the GPUs, since the CPU then has to transfer data back and forth (several times) to the GPU to perform communication inside each Krylov iteration due to e.g. inner products and reductions. The remaining parts of the preconditioner smoother, restriction and prolognation requires less communication. Solving for $A_k^{-1}$ in the right part of (3) is performed with an element wise (local) fast diagonalization method. Albeit better suited for GPUs, neither of these kernels will fully saturate the GPU as well. To increase GPU utilization, amortize communication costs, accelerate the critical path, and thus improve strong scalability, we have developed a new parallelization method of the additive Schwarz preconditioner. The idea with our new formulation is to exploit the available task-parallelism and launch the left and the right part of (3) in parallel on the device. We accomplish this by launching the independent work in parallel from different threads in an OpenMP parallel region. Tasks are launched in separate streams to allow overlap and increase GPU utilization, as illustrated in Fig. 2. The work in coarse grid solve is dominated by kernel launch latency (which throttles GPU execution) and small device kernels (typically too small to keep the GPU utilization high). Launching GPU work in parallel allows hiding this launch latency. It also exposes device-side concurrency enabling overlap of kernels and data movement on different streams, in particular the typically short coarse-solve kernels with other larger kernels. To maximize kernel overlap and ensure progress on both streams, we use stream priorities and assign higher priority to the stream where the coarse-solve work is launched. This is necessary on NVIDIA GPUs to allow small coarse-solve kernels to progress even in the presence of already executing larger kernels. This is not a concern on AMD GPUs, which can schedule concurrent kernels for parallel execution regardless of priorities. In addition, this approach also allows hiding some of the latency overheads incurred by host-initiated GPU-aware communication inherent to current MPI implementations which lack tight integration with accelerator programming models and require waiting on the host for data to be ready on GPU prior to communication. Stream-aware MPI approaches like that proposed by Namashivayam et al. [20] would integrate well with our approach and we expect these to further improve efficiency of our additive Schwarz preconditioner parallelization[1].

## 6 HOW PERFORMANCE WAS MEASURED

We evaluated the performance of Neko on a RBC case of unprecedented size in a cylinder with an aspect ratio of 1:10. In preparation for even higher $Ra$ runs, we considered the simulation at $Ra = 10^{15}$ as our benchmarking case. We then measured the performance of the entire application and assessed the scalability of Neko.

For our simulations at $Ra = 10^{15}$, we use a significantly larger mesh than for the DNS by Iyer et al. [9]. Our mesh is composed of 108M elements and polynomial degree 7, corresponding to 37B unique grid points and more than 148B degrees of freedom. The mesh is designed carefully to get an adequate refinement in the near-wall regions on both side walls and the bottom and top walls, while still capturing all relevant dynamics in the center of the cylinder, which will be necessary as we reach even higher $Ra$.

For our simulation with Neko, we utilize the splitting scheme as proposed in [13] in order to decouple the velocity and pressure solve at each time step. For the discretization in time, we utilize a mixed implicit-explicit scheme, combining an extrapolation scheme and a backwards difference scheme, both of order 3. We perform dealiasing (overintegration) according to the 3/2-rule. While the pressure is solved through a hybrid-Schwarz multigrid preconditioner combined with GMRES, as described in the previous section, the velocity and temperature field use a block-Jacobi preconditioner and conjugate gradient iterative solver.

As previously discussed, the spatial discretization in Neko is based on the spectral-element method, obtaining a high-order convergence and accurate results. The key component of the scalability in Neko is due to the so-called gather-scatter operation, performing the communication along element boundaries and enabling a fast evaluation of differential operators in a matrix-free fashion. In Neko, the gather-scatter is executed in native Fortran with a gather-scatter backend fully aware of the topology of the mesh. This lets the gather-scatter operation be carried out in two phases, one for the local and one for the shared elements between different MPI ranks. This approach has been successful to scale on both conventional CPUs and GPUs, but also more novel (and exotic) architectures such as vector processors [11].

Furthermore, on accelerators, Neko uses backends with heavily optimized native GPU kernels for the computation, and distributed the work along the GPUs with one MPI rank per logical GPU (one rank per Graphics Compute Die (GCD) on AMD devices and one rank per Nvidia device). For all experiments, both the Fortran and accelerator codes were compiled with full optimization (`-O3`) and only double precision floating point numbers were used throughout.

### 6.1 Strong Scalability

Strong-scaling results were obtained by measuring the average time per time-step for a given number of MPI ranks, with the average taken over 250 time-steps, with initial transient iterations

[1]The stream-aware MPI proposed in [20] was not available on the HPE Cray systems used in this work.

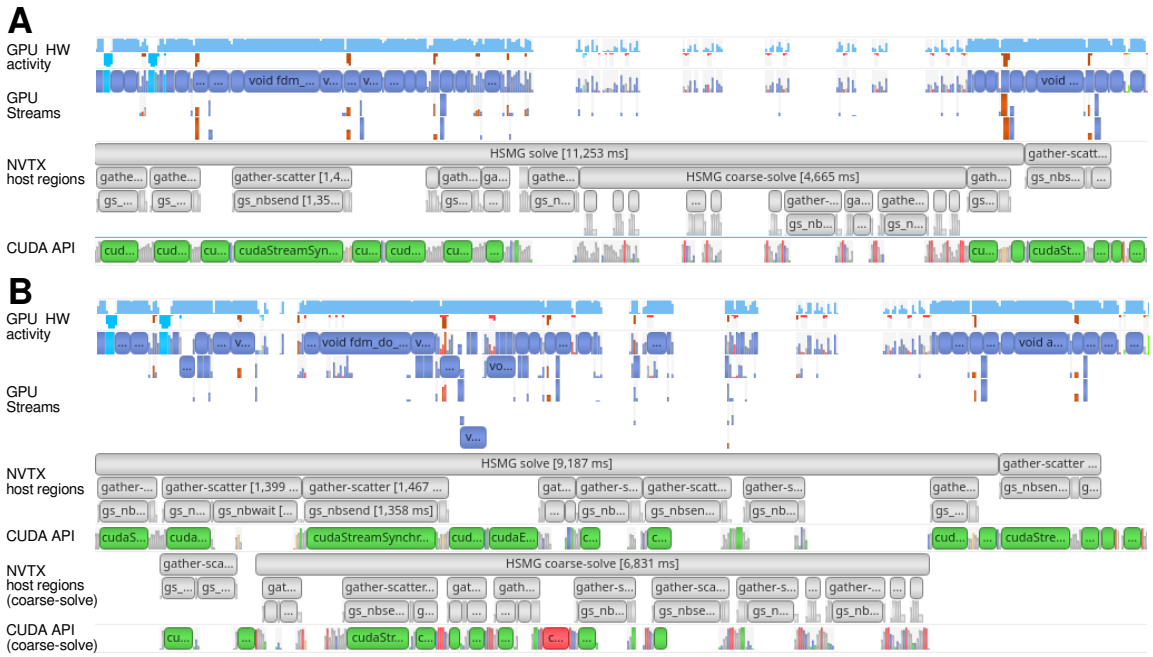


**Figure 2: Trace timeline view of the serial (A) and task-parallel (B) version of the additive Schwarz preconditioner executing on an NVIDIA A100 GPU as part of a single-node 4-GPU run of a small test case representative of the strong-scaling regime of typical production workloads. The task-parallel execution shows: improved GPU utilizaton (fewer gaps "GPU HW activity"), overlap of the coarse-solve GPU kernels with other work, and overlap of host ("CUDA API") activities across the two OpenMP threads scheduling work and MPI communication as well as with GPU execution. For this small test-case running four NVLink connected A100 GPUs the approximate wall-time reduction in the Schwarz preconditioner phase over 50 time-steps is 20%.**

removed. Measurements were performed using `MPI_Wtime` timings around relevant code regions, with global synchronisation points. For all our experiments, we use the same 108M element mesh, while increasing the number of MPI ranks until we almost fill the entire machine.

## 6.2 Data Compression

For the specific case of compression performance, we used an RBC case at $Ra = 10^{11}$. The compression ratio was measured as the relative difference between the original and compressed data size of an instantaneous flow sample. The reconstruction error was measured using a weighted $L^2$ norm of the error between the reconstructed and original field, which amounts to the calculation of the Root Mean Squared error, accounting for the nonuniform nature of the mesh.

# 7 PERFORMANCE RESULTS

The performance measurements were carried out on two of the European High-Performance Computing Joint Undertaking (EuroHPC JU) pre-exascale supercomputers LUMI and Leonardo. LUMI, a 309.10 PFlop/s HPE Cray EX, is located at CSC in Finland and is ranked as the world's third fastest supercomputer on the November 2022 Top500 list[2]. Leonardo, a 174.70 PFlop/s Atos BullSequana XH2000, is hosted by CINECA in Italy and is ranked number four on the November 2022 Top500 list. Performance is measured on both machines since they have two different computer architectures; Leonardo is primarily powered by Nvidia A100 GPUs and LUMI is utilizing the AMD MI250X; a detailed description of the experimental platforms is given in table 1. Experiments were performed between March–April 2023 on LUMI and during April 2023 on Leonardo during its early, pre-production state.

**Table 1: Hardware and software details for our experimental platforms. Bandwidth and performance is per GPU.**

| System | LUMI | Leonardo |
|---|---|---|
| Computing device | AMD MI250X | Nvidia A100 |
| Peak TFlop FP64/s | 47.9 (95.7 Matrix) | 9.7 (19.5 Tensor) |
| Peak BW/s | 3300 | 1550 |
| No. devices | 10240 | 13824 |
| Interconnect | HPE Slingshot 11<br>200 GbE NICs (4x200 Gb/s) | Nvidia HDR<br>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 |

[2]https://www.top500.org/lists/top500/list/2022/11/

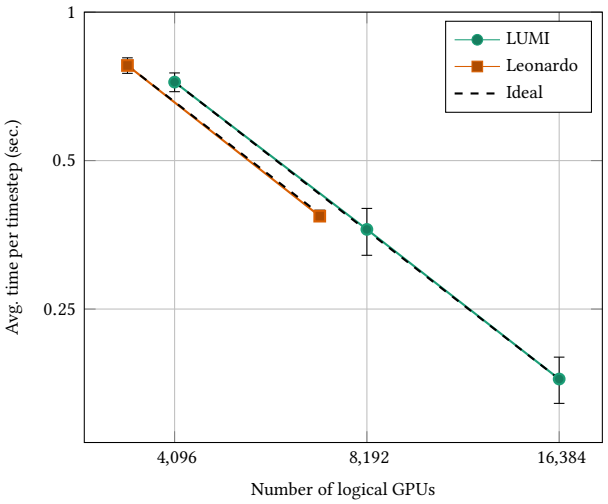


**Figure 3: Strong scaling of Neko for the RBC case, with performance measured in average time per time-step. Logical GPUs refers to one MI250X Graphics Compute Die (GCD) on LUMI and one A100 GPU on Leonardo. Perfect strong scaling is illustrated as black dashed lines for both systems, and the 99% confidence intervals is illustrated as error bars.**

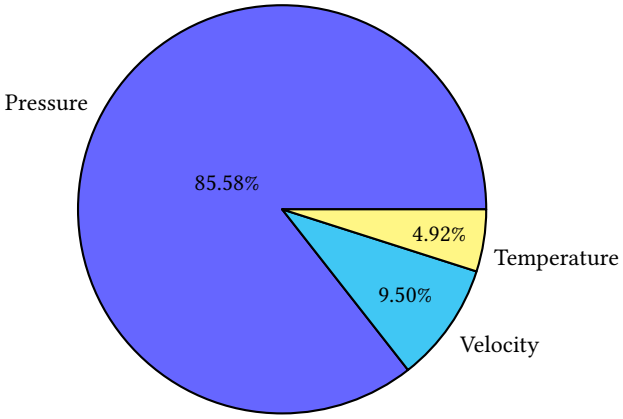


**Figure 4: Wall-time distribution of one time-step in the 16,384 GCD simulation on LUMI, where Pressure, Velocity and Temperature incorporate all components necessary to advance each quantity in time, e.g., generating right-hand sides, initial guesses and solving the equations.**

## 7.1 Strong Scaling

Strong scalability tests were performed for the outlined RBC case on both machines. On LUMI we used 4096, 8192 and 16384 GCDs, corresponding to 20%,40% and 80% of the full machine capacity, and on Leonardo 3456 and 6912 GPUs corresponding to 25% and 50% of the full machine capacity. We would again like to stress that we were given access to Leonardo very early in its pre-production state. Thus there was not enough time to obtain results for more than half of the machine before the nomination deadline.

The results for both machines, presented in Fig. 3, show that Neko achieves close to perfect parallel efficiency for the studied turbulent Rayleigh–Bénard convection case on both LUMI and Leonardo and their different GPUs, demonstrating the performance portability of the code. Furthermore, the results also show that Neko obtains close to perfect parallel efficiency with less than 7000 elements per logical GPU, significantly reducing the smallest required problem size for strong scalability limits compared to previously reported results [14]. The main reason for the improvements is the new overlapped pressure preconditioner (see Section 5.3). With pressure constituting more than 85% of the time for computing a time-step, as illustrated in Fig. 4, the new overlapped formulation addresses one of the major performance/scalability bottlenecks in Neko and similar spectral-element codes, significantly improving upon state of the art.

## 7.2 Data Compression

Fig. 5 shows the result of applying the compression scheme to a stream-wise velocity field in RBC, for which the method produced a 97% of data reduction with a relative error of 2.5%. Even at this level of compression, no visual difference can be observed, and thus the main characteristics of the flow remain unaffected. The reason that such a reduction of data is possible while conserving quality is that while all energetic scales of turbulence are required to solve the dynamics of the problem at run-time, some of them can be removed strategically for post-processing. Neko removes this information while respecting the error bounds specified by the user before executing the simulation. We note that the compression level always depends on the post-processing task to be performed on the data by the user, but we have found that conservative compression levels of 85–90% allow for high-fidelity results.

# 8 IMPLICATIONS

This work has several implications, both regarding the HPC aspects of said work and for the scientific community studying turbulence and Rayleigh–Bénard convection.

## 8.1 Insight into Rayleigh–Bénard Convection

As discussed in the Introduction, turbulence research has identified a number of relevant questions inherent to fundamental turbulent convective flows. As the control of the flow parameters in experimental studies is very challenging and at times even impossible due to unavoidable inaccuracies, physicists and engineers alike resort to numerical simulations as the tool for assessing flow observables under exactly controlled conditions.

This work considers the high-resolution simulation of turbulent flows at extreme parameters, under complex conditions yet in canonical geometries. In the case at hand, Rayleigh–Bénard convection is considered, at high Rayleigh numbers. This is challenging from a computational point of view as the viscosity in the system is low, and the flow is mainly driven by convection and inertia. Small excited scales, yet slow overall dynamics are the typical consequence in these situations, which directly leads to immense computational requirements.

Therefore, the development of sustainable workflows tailored for largest-scale simulations on modern hardware are essential to

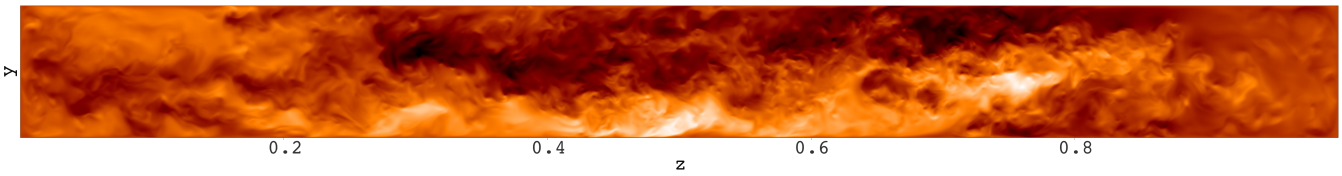


**Figure 5: Two-dimensional slice of a compressed stream-wise velocity field at $Ra = 10^{11}$. The data size was reduced by 97% with a relative error between the reconstructed and original fields of 2.5%. Only the compressed data set is shown as no appreciable differences exist for this visualization. In the picture, $z = 0$ represents the lower (hot) wall, and $z = 1$ the upper (cold) wall.**

address fundamental questions in turbulence. For RBC, the scaling of the heat transfer (measured as the Nusselt number) as a function of *Ra* is heavily debated in the limit of very high *Ra*. Our ambition is to use the workflow described in this work to settle this debate. What is however needed for that is the ability to perform large simulations (on tens of billions degrees of freedom) for a sufficiently long time, to collect statistics and modal data during the simulation lifetime, and to store selected instantaneous data.

The main aspect is the question of whether there is an ultimate regime in RBC, and when it could be reached. For this, one needs to consider a variety of aspect ratios of the RBC and sequences of Rayleigh numbers; all with careful validation of the setup, resolution and simulation development and duration. The consideration of the ultimate regime is very timely. The recent literature points out two completely diverse propositions: first that the ultimate domain might have already been reached [19], or that the ultimate regime is just within reach [1], or that it has not been reached [9]. This can only be settled through a numerical simulation made possible through the developments we have detailed in this work.

Direct implications of these results will allow for more detailed understanding of convective processes in e.g. the atmosphere and stars, eventually leading to refined turbulence models and low-order representations. These in turn will make engineering predictions more accurate which relate, for instance, to climate simulation.

## 8.2 HPC Impact

Detailed simulations of turbulence quickly reach unmanageable amounts of data. In this work we have illustrated how we can leverage modern computer architectures, high-order methods, and compression to overcome several of the obstacles for these large simulations. By adopting similar techniques in other codes and workflows, we anticipate that the productivity and scale of other simulations can increase as well. In particular, through in-situ data processing, we foresee that completely new ways of analyzing and processing data from large simulations will be made possible. This will in turn enable use to obtain new knowledge into complex phenomena across a large number of computational domains.

From our efforts, it is also clear that heterogeneous architectures with high-bandwidth accelerator devices such as GPUs offer a performance unmatched by previous architectures. This is only true however for codes and methods such as the spectral-element method that are able to utilize high-throughput, high-bandwidth architectures efficiently. Moving away from formulations requiring sparse matrix multiplications and instead looking towards for example matrix-free methods is crucial in order to efficiently utilize these upcoming architectures with a high memory bandwidth. In the current architectural transition of HPC toward dense accelerator-based machines with complex intra- and inter-node topologies and varying level of integration, it is critical to keep revising algorithms and parallelization techniques and adapt them to the fast changing landscape of HPC architectures. The GPU-resident parallelization and expressing more of the available concurrency in the application, here in the form of task-decomposed overlapped pressure-conditioner, have been key ingredient to achieve the good strong scalability on the LUMI and Leonardo pre-exascale machines. This algorithmic approach is expected to remain highly beneficial on future generations of HPC architectures. At the same time, integration of CPU and GPU resources on-package, which allows fine-grained task scheduling across both without expensive data movement, will likely require revisiting and re-tuning these algorithms.

## ACKNOWLEDGMENTS

This work has received funding from the European High Performance Computing Joint Undertaking (EuroHPC-JU) under grant agreements no. 101093393 (CEEC) and no. 101092621 (EXCELLERAT P2). This work has also received funding from the European Union's Horizon 2020 research and innovation program under grant agreements no. 955606 (DEEP-SEA) and no. 956748 (ADMIRE). The EuroHPC-JU receives support from the European Union's Horizon 2020 research and innovation programme and Germany, France, Spain, Italy, Slovenia, Greece, Belgium, Sweden, United Kingdom, Switzerland, and Denmark. Financial support was also provided by the Swedish e-Science Research Centre Exascale Simulation Software Initiative (SESSI) and the Swedish Research Council under grant agreement no. 2019-04723. Some of the computations were enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS) and the Swedish National Infrastructure for Computing (SNIC) at PDC Centre for High Performance Computing partially funded by the Swedish Research Council through grant agreement no. 2022-06725. We also acknowledge NAISS and SNIC for awarding this project access to the LUMI supercomputer, owned by the EuroHPC-JU, hosted by CSC (Finland) and the LUMI consortium through a LUMI Sweden XLarge call. We are grateful to CINECA and the EuroHPC-JU for the availability of high performance computing resources in the form of the Leonardo supercomputer hosted by CINECA (Italy).

# REFERENCES


[1] Guenter Ahlers, Eberhard Bodenschatz, Robert Hartmann, Xiaozhou He, Detlef Lohse, Philipp Reiter, Richard JAM Stevens, Roberto Verzicco, Marcel Wedi, Stephan Weiss, et al. 2022. Aspect ratio dependence of heat transfer in a cylindrical Rayleigh-Bénard cell. *Physical Review Letters* 128, 8 (2022), 084501.

[2] Steven L. Brunton, Bernd R. Noack, and Petros Koumoutsakos. 2020. Machine Learning for Fluid Mechanics. *Annual Review of Fluid Mechanics* 52, 1 (2020), 477–508. https://doi.org/10.1146/annurev-fluid-010719-060214

[3] Paul Fischer, Stefan Kerkemeier, Misun Min, Yu-Hsiang Lan, Malachi Phillips, Thilina Rathnayake, Elia Merzari, Ananias Tomboulides, Ali Karakus, Noel Chalmers, et al. 2022. NekRS, a GPU-accelerated spectral element Navier–Stokes solver. *Parallel Comput.* 114 (2022), 102982.

[4] Paul F Fischer. 1997. An Overlapping Schwarz Method for Spectral Element Solution of the Incompressible Navier–Stokes Equations. *J. Comput. Phys.* 133, 1 (1997), 84–101.

[5] Paul F. Fischer and James W. Lottes. 2005. Hybrid Schwarz-Multigrid Methods for the Spectral Element Method: Extensions to Navier-Stokes. In *Domain Decomposition Methods in Science and Engineering*. Springer Berlin Heidelberg, 35–49.

[6] William F Godoy, Norbert Podhorszki, Ruonan Wang, Chuck Atkins, Greg Eisenhauer, Junmin Gu, Philip Davis, Jong Choi, Kai Germaschewski, Kevin Huck, et al. 2020. ADIOS 2: The adaptable input output system. A framework for high-performance data management. *SoftwareX* 12 (2020), 100561.

[7] Siegfried Grossmann and Detlef Lohse. 2000. Scaling in thermal convection: a unifying theory. *Journal of Fluid Mechanics* 407 (2000), 27–56.

[8] Philip Holmes, John L. Lumley, and Gal Berkooz. 1996. *Proper orthogonal decomposition.* Cambridge University Press, 86–128. https://doi.org/10.1017/CBO9780511622700.004

[9] Kartik P Iyer, Janet D Scheel, Jörg Schumacher, and Katepalli R Sreenivasan. 2020. Classical 1/3 scaling of convection holds up to $Ra = 10^{15}$. *Proceedings of the National Academy of Sciences* 117, 14 (2020), 7594–7598.

[10] Niclas Jansson. 2021. Spectral element simulations on the NEC SX-Aurora TSUBASA. In *The International Conference on High Performance Computing in Asia-Pacific Region (HPC Asia 2021)*. ACM, Virtual Event, Republic of Korea, 32–39.

[11] Niclas Jansson, Martin Karp, Artur Podobas, Stefano Markidis, and Philipp Schlatter. 2021. Neko: A modern, portable, and scalable framework for high-fidelity computational fluid dynamics. *arXiv preprint 2107.01243* (2021).

[12] Yi Ju, Adalberto Perez, Stefano Markidis, Philipp Schlatter, and Erwin Laure. 2022. Understanding the Impact of Synchronous, Asynchronous, and Hybrid In-Situ Techniques in Computational Fluid Dynamics Applications. In *2022 IEEE 18th International Conference on e-Science (e-Science)*. 295–305. https://doi.org/10.1109/eScience55777.2022.00043

[13] George Em Karniadakis, Moshe Israeli, and Steven A Orszag. 1991. High-order splitting methods for the incompressible Navier-Stokes equations. *J. Comput. Phys.* 97, 2 (1991), 414–443.

[14] Martin Karp, Daniele Massaro, Niclas Jansson, Alistair Hart, Jacob Wahlgren, Philipp Schlatter, and Stefano Markidis. 2023. Large-Scale direct numerical simulations of turbulence using GPUs and modern Fortran. *The International Journal of High Performance Computing Applications* 37, 5 (2023), 487–502. https://doi.org/10.1177/10943420231158616

[15] Martin Karp, Artur Podobas, Tobias Kenter, Niclas Jansson, Christian Plessl, Philipp Schlatter, and Stefano Markidis. 2022. A High-Fidelity Flow Solver for Unstructured Meshes on Field-Programmable Gate Arrays: Design, Evaluation, and Future Challenges. In *International Conference on High Performance Computing in Asia-Pacific Region* (Virtual Event, Japan) *(HPC Asia 2022)*. ACM, Virtual Event, Japan, 125–136.

[16] Gijs L Kooij, Mikhail A Botchev, Edo MA Frederix, Bernard J Geurts, Susanne Horn, Detlef Lohse, Erwin P van der Poel, Olga Shishkina, Richard JAM Stevens, and Roberto Verzicco. 2018. Comparison of computational codes for direct numerical simulations of turbulent Rayleigh–Bénard convection. *Computers & Fluids* 166 (2018), 1–8.

[17] Robert H Kraichnan. 1962. Turbulent thermal convection at arbitrary Prandtl number. *The Physics of Fluids* 5, 11 (1962), 1374–1389.

[18] Faming Liang, Runmin Shi, and Qianxing Mo. 2016. A Split-and-Merge Approach for Singular Value Decomposition of Large-Scale Matrices. *Statistics and its interface* 9, 4 (2016), 453–459. https://doi.org/10.4310/SII.2016.v9.n4.a5

[19] Erik Lindborg. 2023. Scaling in Rayleigh–Bénard convection. *Journal of Fluid Mechanics* 956 (2023), A34.

[20] Naveen Namashivayam, Krishna Kandalla, Trey White, Nick Radcliffe, Larry Kaplan, and Mark Pagel. 2022. Exploring GPU Stream-Aware Message Passing using Triggered Operations. *CoRR* abs/2208.04817 (2022). https://doi.org/10.48550/arXiv.2208.04817 arXiv:2208.04817

[21] Evelyn Otero, Ricardo Vinuesa, Oana Marin, Erwin Laure, and Philipp Schlatter. 2018. Lossy Data Compression Effects on Wall-bounded Turbulence: Bounds on Data Reduction. *Flow, Turbulence and Combustion* 101, 2 (01 Sep 2018), 365–387. https://doi.org/10.1007/s10494-018-9923-5

[22] James W. Lottes Paul F. Fischer and Stefan G. Kerkemeier. 2008. nek5000 Web page. http://nek5000.mcs.anl.gov.

[23] Saleh Rezaeiravesh, Ricardo Vinuesa, and Philipp Schlatter. 2021. On numerical uncertainties in scale-resolving simulations of canonical wall turbulence. *Computers & Fluids* 227 (2021), 105024.

[24] Jörg Schumacher and Katepalli R Sreenivasan. 2020. Colloquium: Unusual dynamics of convection in the Sun. *Reviews of Modern Physics* 92, 4 (2020), 041001.

[25] C. E. Shannon. 1948. A mathematical theory of communication. *The Bell System Technical Journal* 27, 4 (1948), 623–656. https://doi.org/10.1002/j.1538-7305.1948.tb00917.x

[26] Zhu Wang, Brian McBee, and Traian Iliescu. 2016. Approximate partitioned method of snapshots for POD. *J. Comput. Appl. Math.* 307 (2016), 374–384. https://doi.org/10.1016/j.cam.2015.11.023 1st Annual Meeting of SIAM Central States Section, April 11–12, 2015.