Funded by the European Union under the
HORIZON-EUROHPC-JU-2023-INCO-06 programme.
Grant Agreement No. 101196247

Deliverable D5.1 –ICON-CLM Deployment

WP5 –Weather & Climate

Under Review

Funded by
the European Union

Document Information

Deliverable Number	D5.1
Deliverable Title	ICON-CLM Deployment
Due Date	2026-01-31 (PM12)
Deliverable Lead	CMCC
Authors	Manuel Stocchi, Francesca Mele, Italo Epicoco
Keywords	ICON-CLM, deployment
WP	WP5
Nature	Report
Dissemination Level	Public
Final Version Date	2026-01-26
Reviewed by	David Guibert (BULL) Antoine Morvan (BULL) Nishtha Srivastava (GUF) Andrey Alekseenko (KTH)
MGT Board Approval	2026-01-30

Document History

Partner	Date	Comments	Version
CMCC	2026-01-02	First draft	0.1
CMCC	2026-01-26	Final version	0.2

Executive Summary

This document describes the procedures, configurations, and technical steps performed to correctly install, configure, and run the ICON-CLM climate model on the PARAM Rudra supercomputer. The deliverable specifically documents the work carried out within Task 5.8 of the GANANA project, which focuses on porting the CMCC configuration of the ICON-CLM regional climate model, including its urban parameterization, onto the high-performance computing (HPC) infrastructure provided by C-DAC. This activity represents a fundamental prerequisite for enabling stable, efficient, and reproducible high-resolution climate simulations specifically designed for urban environments. By detailing the deployment process on PARAM Rudra, this deliverable provides a technical reference for the successful execution of ICON-CLM in an HPC context and establishes the foundation for subsequent model tuning and long-term urban climate simulations planned in the following project tasks.

Table of Contents

1 Introduction

Climate models are physical simulations that aim to predict the evolution of the Earth's climate, either globally or over a specific region, within a particular timeframe. The inherent complexity of these models requires simulations to be performed using high spatial resolution discretizations in order to better predict extreme events, particularly in regional and local-scale applications. As such, high-performance computing (HPC) has proven to be fundamental for running climate models that achieve reliable results in a feasible computational time.

One of the objectives of Work Package 5, "Weather and climate", within the GANANA project, an EU-India initiative aimed at fostering research collaboration in scientific HPC, is the exchange of expertise in regional and local climate modeling. In particular, simulations of climate evolutions will be run for the region of the Indian city of Pune (Maharashtra, Western India), using a domain resolution of approximately 2 km. These simulations will be run using the ICON-CLM model, which has been previously adopted by the CMCC Foundation in its research activities. These simulations will be run on PARAM Rudra, a supercomputing facility designed and deployed by the Centre for Development of Advanced Computing (C-DAC) primarily for the Inter-University Acceleration Centre (IUAC) in Delhi, which hosts the facility.

This deliverable documents the deployment of the ICON-CLM model on the PARAM-Rudra HPC system, providing the technical foundation for subsequent modeling and simulation activities within the project.

2 Overview of the ICON atmospheric model

The ICOsahedral Nonhydrostatic (ICON) model is a unified global numerical weather prediction (NWP) and climate modeling framework [1] developed through a collaboration involving the German Meteorological Service (DWD), the Max Planck Institute for Meteorology (MPI-M), the German Climate Computing Center (DKRZ), the Karlsruhe Institute of Technology (KIT), and the Swiss Center for Climate System Modeling (C2SM). It was designed with the goal of providing a next-generation modeling system to succeed COSMO, the numerical weather prediction and climate model previously used by the aforementioned institutions. In particular, ICON was developed to support high spatial resolutions (up to the order of kilometers), satisfy the requirements of exact local mass conservation and mass-consistent transport, be scalable on highly parallelized HPC architectures and enable static mesh refinement.

2.1 Spatial discretization

ICON is built on an unstructured triangular grid based on a uniform icosahedral horizontal discretization, often referred to as an icosahedral-triangular Arakawa C grid. The grid generation begins with a spherical icosahedron composed of 20 equilateral

triangles. It is refined through a multi-step subdivision process, starting with edge partitioning and followed by recursive bisection, where each triangle is divided into four smaller triangles. To improve numerical stability and accuracy, the grid undergoes optimization using a spring-based relaxation method that adjusts vertex positions until equilibrium is reached. The resulting domain consists primarily of hexagonal cells, except for 12 pentagonal cells corresponding to the original icosahedron vertices, each connected to five neighboring cells. This process is illustrated in Fig. 1.

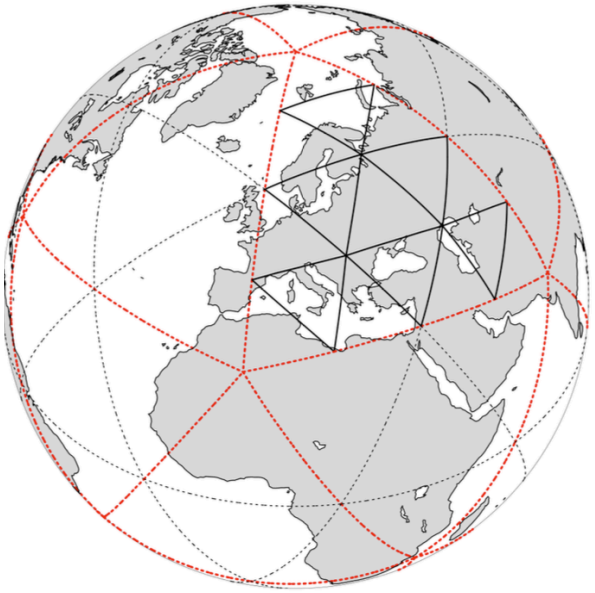

Figure 1: Illustration showing how the horizontal triangular grid is constructed. An original spherical icosahedron (in red) is iteratively split into equilateral triangles. These triangles can then be grouped to shape a hexagon grid, with the exception of points located on the vertices of the original icosahedron, which draw a pentagon. Illustration taken from Prill et al., (2025) [5].

In the vertical dimension, ICON employs a height-based coordinate system that follows the terrain. The model uses a Lorenz-type staggering, where prognostic variables such as horizontal velocity, virtual potential temperature, and density are defined at the full levels located at the centers of the vertical grid boxes. Vertical velocity, on the other hand, is defined at the half levels corresponding to the top and bottom faces of each vertical layer. Two main coordinate systems are available: the terrain-following Hybrid Gal-Chen coordinate and the Smooth Level Vertical (SLEVE) coordinate, which is the default and recommended option. SLEVE is designed to gradually reduce the influence of smaller-scale terrain features with height, allowing terrain-following layers near the surface to transition smoothly into levels of constant height at higher altitudes.

2.2 The physics of ICON

The dynamical core of ICON is based on a system of partial differential equations (PDEs) describing the transport of a multicomponent fluid composed of dry air, water vapor, cloud water, rain water, cloud ice, and solid precipitating particles (e.g., snow or hail). This formulation follows the approach proposed by Gassmann and Herzog (2008) [7]

and consists in a continuity equation for the fluid, mass balance equations for each component of the fluid, an energy balance equation used to predict the Exner pressure, and a vector-invariant equation for the momentum balance. The key feature of this equation set is the use of a non-hydrostatic formulation, which ensures better results at high resolutions.

The momentum balance equations employ the two-dimensional Lamb transformation, a vector identity that converts non-linear momentum advection terms into a vector-invariant form. This formulation grants that no gradients of vectors appear in the equation, thus simplifying the implementation on arbitrary frames of coordinates. The system of PDEs is closed with an equation of state for relating Exner pressure, potential temperature and fluid density and a set of boundary conditions for vertical momentum and diffusion fluxes. As it is typical of atmospheric models, grid-scale thermodynamic variables (i.e., fluid density, Exner pressure and potential temperature) are described as the sum of a horizontally homogeneous, constant in time, dry and hydrostatically balanced reference value and a fluctuation to this reference value.

2.3 Solution strategy

This theoretical model is implemented using three building blocks, designed to simulate processes occurring at different time scales: the dynamical core, tracer transport, and physics parametrization.

The dynamical core numerically solves a system of PDEs for the evolution of horizontal and vertical momentum, fluid density and Exner pressure, taking into account only the terms that can be resolved within the scale of the spatial grid. The tracer advection computes the redistribution of tracer mass fraction among the domain through the tracer continuity equation. This separation allows solving the problem with a time-splitting strategy: the dynamical core, which needs to work with short timesteps in order to ensure numerical stability, is updated using short timesteps, while the tracer advection is updated using longer timesteps (by default it is five times longer). The physics parametrization evaluates the effects of processes occurring at smaller spatial scales than the ones that are resolved by the discretized grid. These processes can be divided into fast physics processes and slow physics processes. The former include saturation adjustment, land-surface interaction, turbulent diffusion, and cloud microphysics, and are evaluated with the same timestep as tracer advection, while the latter include radiation, cumulus convection, non-orographic gravity drag, sub-grid scale orographic drag, and cloud cover, and are evaluated at longer timesteps. The adoption of this time-splitting strategy allows to strongly reduce the number of operations which are necessary for each timestep, strongly improving performance.

The dynamical core solves its set of equations using a two-time-level predictor–corrector scheme. While most terms are treated explicitly, vertical sound-wave propagation is handled implicitly to ensure numerical stability. Tracer

transport is addressed through mass-conserving finite volume methods, specifically Flux Form Semi-Lagrangian (FFSL) schemes. These methods combine Eulerian flux calculations with semi-Lagrangian trajectory estimates to compute mass fluxes accurately. Horizontal transport relies on reconstructing flux areas via backward trajectories and estimating subgrid tracer distributions using polynomial reconstructions, followed by numerical integration. Vertical transport employs high-order finite volume schemes such as the Piecewise Parabolic Method (PPM) and the Parabolic Spline Method (PSM), both designed to preserve mass and continuity. Additional filtering techniques are applied to maintain monotonicity and positive definiteness.

2.4 ICON-CLM

Over time, several declinations of ICON have been released, thus creating a framework of models that differ in the simulated physics (climate models, weather forecast models, ocean models) and in the scale that is studied, going from global, global with nested grids over specific areas, regional models, and regional models with nested grids. An overview of these implementations can be seen in Fig. 2.

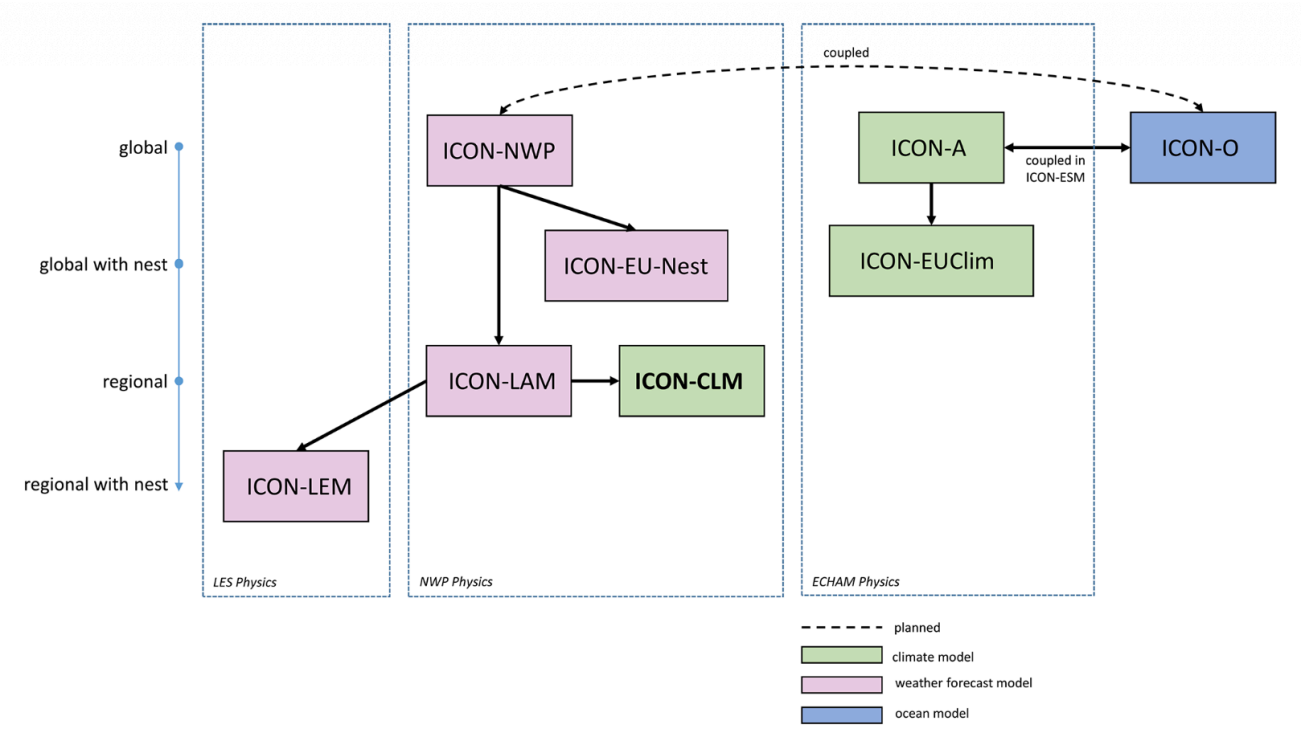

Figure 2: Schematic overview of the ICON modeling framework. Taken from Pham et al., (2021) [2]. Acronyms in the figure from top to bottom, left to right mean: ICON for Numerical Weather Prediction (ICON_NWP), ICON Atmosphere (ICON-A), ICON Ocean (ICON-O), ICON with Nested Domain over Europe (ICON-EU-Nest), ICON Climate model for Europe (ICON-EUClim), ICON in Limited Area Mode (ICON-LAM), ICON Climate Limited Area mode (ICON-CLM), and ICON Large Eddy Mode (ICON-LEM).

The Climate Limited-area Mode implementation (ICON-CLM) [2] is the ICON implementation for climate modeling on regional scales, and stems directly from the regional NWP model ICON-LAM (ICON Limited Area Mode). It was developed by the CLM community, a consortium of researchers and institutions working on regional climate models, with the goal to replace COSMO-CLM, the climate limited area mode of COSMO. With respect to its predecessor, ICON-CLM offers better performance for air temperature, slightly improved total cloud cover results, a reduction in warm temperature bias, and improved representation of diurnal cycles and wind speeds [2,3,4].

The core formulation remains consistent with the numerical weather prediction setup, but several modifications are introduced to accommodate long-term integrations and boundary-driven processes. The model relies on external forcing data to define initial and lateral boundary conditions, which are typically derived from global climate models or reanalysis datasets (such as the ERA5 or ERA-Interim dataset produced by the ECMWF, the European Centre for Medium-range Weather Forecast). Boundary fields are updated at discrete intervals with linear interpolation applied between updates to ensure temporal continuity. At the upper boundary, a relaxation technique is employed above a specified altitude to maintain consistency with large-scale atmospheric conditions and to suppress spurious wave reflections. For extended simulations, additional time-dependent forcings, including sea surface temperature, sea-ice cover, and greenhouse gas concentrations, are incorporated to capture evolving climate signals.

The terrestrial and urban interface in ICON is represented by the Land-Soil Model TERRA and the urban canopy parameterization TERRA_URB (TU). TERRA simulates the exchange of energy and water fluxes between land and atmosphere, providing the lower boundary condition for atmospheric processes. It uses a multi-layer soil structure to resolve temperature and moisture dynamics, and accounts for vertical transport of water and heat, including processes such as freezing, melting, runoff, and bare-soil evaporation. Vegetation effects are modeled through evapotranspiration and root water uptake, while snow processes are represented by a simplified prognostic scheme.

Urban environments are parameterized through TU, which captures key physical processes influencing the Urban Heat Island (UHI) effect. TU employs a bulk approach to represent urban morphology and thermal properties. It modifies surface energy and hydrological fluxes over impervious areas, and incorporates anthropogenic heat contributions. It also accounts for heat storage within urban materials, which absorb solar radiation during the day and release heat at night, contributing to elevated nighttime temperatures.

ICON applies a tiled land-surface approach to represent subgrid heterogeneity, ensuring consistent treatment of different surface types within each grid cell. This

framework allows realistic simulation of land-atmosphere interactions across both natural and urban landscapes.

2.5 Parallelization

A considerable portion of ICON, in particular the most computationally-intensive parts, are written in Fortran90 and can run in parallel on both distributed-memory architectures using MPI and on shared-memory architectures using OpenMP. This allows users to run ICON with both parallelization paradigms (i.e., with hybrid parallelization) in order to exploit at best the different levels of concurrent execution offered by modern HPC machines.
Additionally, ICON supports GPU acceleration. Some components of ICON-Atmosphere (ICON-A) have been ported to GPU using OpenACC, thus allowing users to reach even greater speedups for suitable problems. However, as the 2025 ICON user guide reports [5], GPU support is not operational yet, and it is recommended to consider the GPU version of ICON as experimental.

2.6 The SPICE runtime environment

SPICE, the Starter Package for ICON-CLM Experiments, provides a structured runtime environment for managing long-term regional climate simulations with ICON in Climate Limited-area Mode. Its design builds on the COSMO-CLM infrastructure, ensuring continuity in workflow and output organization to facilitate user transition and compatibility with existing post-processing tools.

The core of SPICE is the subchain, a main routine that orchestrates the simulation process through a sequence of interdependent steps, typically executed in monthly restart cycles. The workflow begins with data preparation, where global forcing datasets—such as initial conditions and lateral, upper, and lower boundary fields—are collected and verified for completeness and consistency. This step ensures that all required external inputs, including time-dependent fields like sea surface temperature and greenhouse gas concentrations, are available for the simulation period.

Next, the data conversion and interpolation phase preprocesses these inputs and maps them onto the ICON-CLM grid. This involves horizontal and vertical interpolation of boundary conditions, transformation of file formats, and generation of model-ready input fields. Accurate preprocessing is critical for maintaining consistency between the driving data and the regional model domain.

The third stage, model execution, manages the ICON-CLM job submission and runtime control. SPICE handles job scheduling, resource allocation, and restart management, ensuring that simulations proceed seamlessly across multiple cycles without user intervention. Upon completion of each cycle, the workflow transitions to archiving, where

output files are compressed, organized, and annotated with metadata compliant with international netCDF metadata conventions (CF conventions).

Finally, the post-processing phase generates derived products such as time series of selected variables and reformats outputs for downstream applications. These processed files are prepared for tools like CCLM2CMOR, enabling conversion to CMIP-compliant formats and publication through ESGF (Earth System Grid Federation) nodes. This stage ensures that ICON-CLM results can be integrated into broader climate research frameworks.

To support validation and reproducibility, SPICE incorporates the Climatological Test Suite (CTS) for automated multi-year test runs and ETOOLS for systematic evaluation against observational datasets. CTS facilitates consistency checks across extended simulations, while ETOOLS produces diagnostic metrics and standardized visualizations, streamlining quality control.

Through this structured workflow—spanning data preparation, preprocessing, execution, archiving, and post-processing—SPICE provides a comprehensive environment for managing ICON-CLM experiments efficiently and reliably, ensuring scientific rigor and interoperability in regional climate modeling.

3 Target HPC system description

3.1 System architecture

The target HPC system for ICON-CLM simulations is the PARAM Rudra supercomputing facility at the Inter-University Acceleration Center (IUAC) in Delhi. PARAM Rudra is based on a heterogeneous and hybrid configuration of Intel Xeon Gold 6240R Cascade Lake processors and NVIDIA Ampere A100 GPU cards. It was designed by the HPC Technologies groups of the Centre for Development of Advanced Computing (C-DAC). The system is composed of 10 login nodes, 2 visualization nodes, 8 management nodes, and 599 compute nodes, with a reported peak performance of 3.1 PFLOPS.

The 599 compute nodes are divided into 473 CPU-only nodes, 30 GPU nodes, 32 GPU-ready (GPUr) nodes, and 64 high memory (HM) CPU-only nodes. Each CPU node hosts two 2nd generation Intel Xeon G-6240R, 2.4GHz processors with 48 cores, a 192 GB DDR4 2933 MHz memory, and a 800GB SSD for local storage. GPU nodes have the same characteristics as CPU-only nodes, but have the capability of being upgraded to host GPUs in the future. GPU compute nodes have the same characteristics as CPU nodes, but additionally host two Nvidia A100 PCIe GPUs with 6912 cores and 80 GB HBM2e memory each. HM nodes have the same characteristics as CPU-only nodes, but have 768GB DDR4 2933 MHz memory.

Storage is based on the Lustre parallel file system, with a total usable capacity of 4.4 PiB and a throughput of 100GB/s.

Computing nodes are interconnected with an InfiniBand HDR 100 Gbps high-bandwidth, low-latency network. A schematic representation of the architecture of PARAM Rudra is shown in Fig.3.

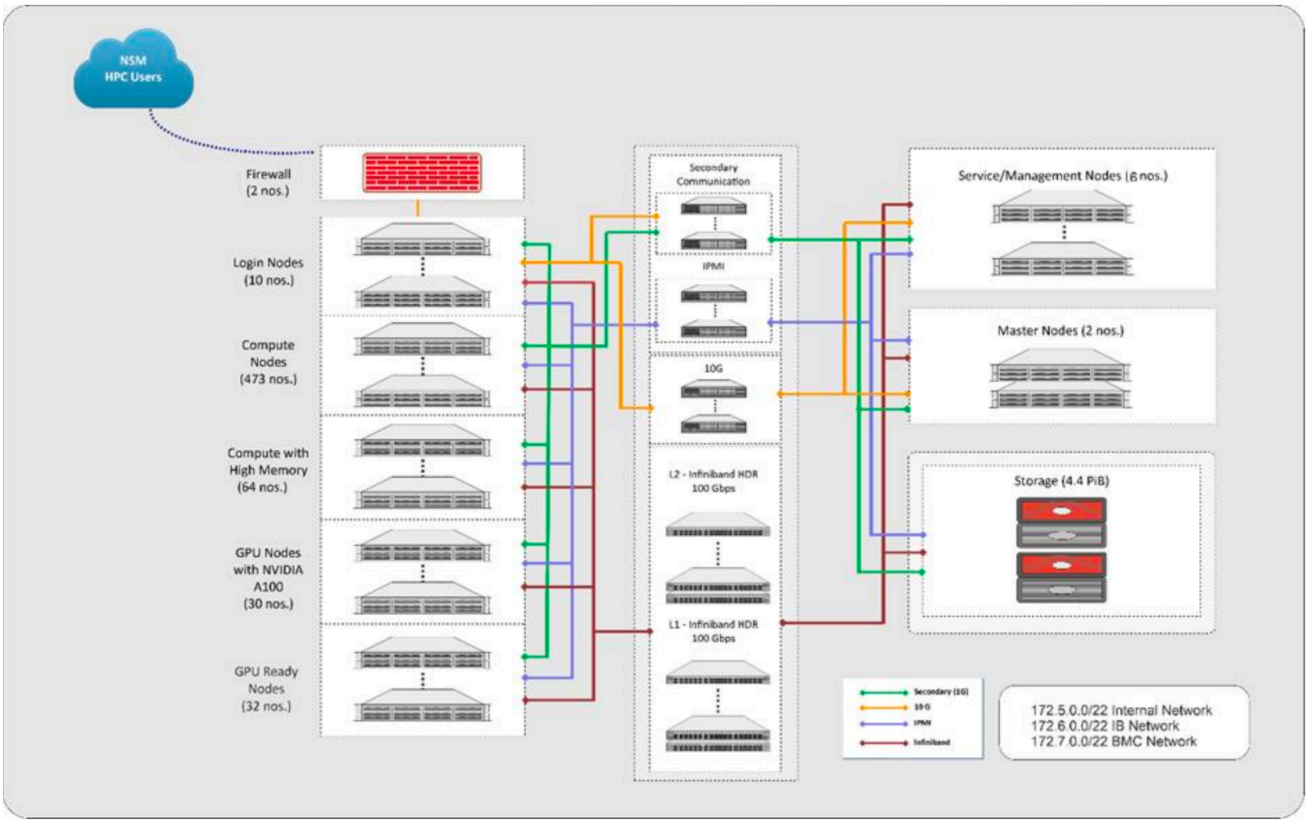

Figure 3: Overview of the PARAM Rudra HPC cluster architecture.

3.2 Software stack

The operating system on PARAM Rudra is Linux - Alma 8.7. The software stack is typical for standard HPC clusters, comprising functionalities for system management, resource management, application development, and performance monitoring. We will discuss only the software components that are crucial for the deployment of ICON-CLM. Both Intel oneAPI and GNU compilers are available, as well as CUDA and OpenACC toolkits for developing GPU-accelerated applications. Widely used libraries, such as NetCDF and HDF5, math libraries, Python libraries, and GNU scientific libraries, are present. The available MPI implementations are Intel MPI, MVAPICH2, and OpenMPI. Resource management and job scheduling is based on SLURM. Software packages are managed via the Spack package manager and Lmod/Environment Modules. The performance analysis tools Intel VTune Profiler and Intel Advisor are available as modules and can be initialized using the following commands:

```
Shell
module load apps/intel-advisor
module load apps/vtune
```

3.3 Spack package manager

PARAM Rudra extensively uses Spack. Spack is a source-based package manager designed to address the complexities of software deployment in HPC environments. Spack enables users to load applications or specific package versions, along with all their dependencies, in a flexible and controlled manner.
Unlike binary-based managers that distribute pre-built software with fixed configurations, Spack automates the compilation of packages from source code. This fundamental design choice allows for a high degree of control over the resulting binaries, ensuring that software is optimized for the specific microarchitecture and technical requirements of a given project.

Spack uses a flexible "spec" (specification) syntax that enables the explicit definition of build parameters. Users can specify the exact version of a compiler to be used for an entire dependency tree, ensuring consistency across the software stack. Additionally, Spack exposes libraries configuration options, called variants, allowing users to activate or deactivate specific functionalities.

Beyond individual package configuration, the tool manages the interdependencies between libraries by allowing multiple, differently configured versions of the same software to coexist on a single system. This is managed through a process called concretization, where Spack resolves a user's high-level requirements into a complete, viable DAG (Directed Acyclic Graph) of dependencies. By utilizing environments, these configurations are captured in metadata files that record the exact state of the software stack. This provides a mechanism for technical reproducibility, allowing a specific build environment to be moved between different clusters while maintaining identical library linkages and compiler settings.

4 Deployment workflow

4.1 Source code acquisition and versioning

The software version used in this deployment was ICON-2025.04 [7], which (at the time of writing) is the second latest publicly available release of ICON, dated April 2025. It was downloaded from the official Gitlab public repository using the command:

```
Shell
git clone --recursive -b icon-2025.04-public
https://gitlab.dkrz.de/icon/icon-model.git
```

The `--recursive` flag ensures that all external modules are downloaded alongside the core repository. These modules provide essential functionalities, including the evaluation of specific atmospheric phenomena (e.g., ART, jsbach), data interfaces handling (e.g., CDI, tixi) or performance and communication tools (e.g., kokkos, yaxt). A complete list of the external modules used by ICON-2025.04 and their purposes is provided in table 1.

MODULE	PURPOSE
ART	Treatment of atmospheric composition and interactions with clouds and radiation
CDI	C and Fortran interface for accessing climate model data
ComIn	Management of data exchange and simulation events between the ICON and 3rd party modules
ecRad	Atmospheric radiation scheme by ECMWF
fortran-support	A collection of general fortran-supporting modules used in ICON
iconmath	A collection of modules for enabling math operations
jsbach	Land and biosphere management
mtime	Model time management
ppm	Partitioning and Parallelization Module
probtest	Python scripts for model testing
RTE+RRTMGP	Computation of radiative fluxes in planetary atmospheres
tixi	XML interface library
yac	Coupling library for Earth System models
yaxt	Communication tool on top of MPI

Table 1: List of ICON external components

4.2 Build configuration and compilation

Although an upstream package for ICON exists in Spack, the configuration and compilation of ICON-CLM were performed using an ICON configuration and compilation bash script optimized for the Intel oneAPI environment with Intel MPI, using the package from the official repository. This approach allows full control over the configuration and compilation process and ensures alignment with official releases, without relying on a third-party system.

The objective was to obtain a highly optimized build of ICON-CLM targeting CPU-only nodes and parallelized with MPI. For this deployment, the Pure MPI execution model was selected to leverage established performance baselines and internal expertise regarding ICON's scalability. By utilizing a process-based parallelization strategy, we ensured a highly stable environment for initial benchmarking. Evaluation of the Hybrid MPI+OpenMP configuration is planned as a subsequent phase to investigate potential memory optimizations on high-core-count nodes.

ICON is configured via Autoconf and compiled using CMake and the Intel oneAPI toolchain to ensure seamless integration of its various modular components. After loading the appropriate Spack modules and setting the required environment variables, compiler flags, and MPI settings, the build process generates optimized binaries by invoking the corresponding make targets for the ICON atmosphere model.

This approach ensures consistency with established CMCC deployment practices, while allowing site-specific adaptations required by the PARAM Rudra system architecture.
The following Bash script snippet shows the configuration procedure adopted for this deployment:

```shell
if [ ${SIMULATION_MODE} = "CLM" ]; then
"${ICON_DIR}/configure" \
CMAKE="${CMAKE}/bin/cmake" \
--prefix="${INSTALL_DIR}" \
--enable-parallel-netcdf \
--enable-intel-consistency \
--enable-waves \
--enable-comin \
--enable-coupling \
--enable-yaxt \
--enable-mpi \
--enable-jsbach \
--enable-quincy \
--enable-mpi-checks=no \
--enable-grib2 \
--enable-vectorized-lrtm \
--enable-ecrad \
--enable-art \
--enable-art-gpl \
--enable-acm-license \
2>&1 | tee configure.log
```

The flags used in the configure command allow users to customize the ICON configuration according to specific requirements. The following section briefly describes these flags, grouped as in the ICON configuration script.

Model features

These flags determine which components of the model have to be installed. These components are in addition to the ones that are enabled by default for modeling atmosphere, large eddies, upper-atmosphere and ocean.

`--enable-waves`	enable the ocean surface wave component of ICON
`--enable-coupling`	enable the coupling
`--enable-comin`	enable the ICON community interface (COMIN)
`--enable-jsbach`	enable the land component of ICON (ICON-Land)
`--enable-quincy`	enable the QUINCY biogeochemistry model in the land component
`--enable-ecrad`	enable usage of the ECMWF radiation scheme (ecRad)
`--enable-art`	enable the aerosols and reactive trace component ART
`--enable-art-gpl`	enable GPL-licensed code parts of the ART component
`--enable-acm-license`	enable code parts that require accepting the ACM Software License Agreement

Infrastructural features

These flags are used to control the deployment on the hosting infrastructure.

`--enable-parallel-netcdf`	enable usage of the parallel features of NetCDF
`--enable-mpi`	enable MPI support
`--enable-yaxt`	enable the YAXT data exchange
`--enable-mpi-checks=no`	disable configure-time checks of MPI library for known defects
`--enable-grib2`	enable GRIB2 I/O

Optimization features

This flag is used to improve the model performances.

`--enable-vectorized-lrtm`	enable the parallelization-invariant version of LRTM

As PARAM Rudra is equipped with Intel CPUs, the Intel oneAPI toolkit was used for configuration and compilation to ensure tested compatibility with Intel hardware and to take advantage of its available optimization features. Compilation was performed using the Intel OneAPI MPI 2021.12.1 library, which provides MPI wrapper compilers for Intel toolchains, in particular `mpiifx` for Fortran and `mpiicx` for C.

Both C and Fortran codes were compiled setting FCFLAGS, CFLAGS, CXXFLAGS, FFLAGS to `-O3`, which enables the highest level of optimization, and `-xHost`, which activates architecture-specific optimization. OpenMP support was explicitly disabled using the `-qno-openmp` flag. Numerical accuracy was preserved by enabling the `-fp-model precise` option, while the `-ftz` flag was used to flush denormal floating-point values to zero in order to improve computational efficiency. In addition, the Fortran compiler was configured with the `-align array64byte` flag to align arrays on 64-byte boundaries, improving vectorization and memory access efficiency. Together, these compilation options aim to maximize performance while maintaining numerical precision. As previously mentioned, the build process was managed using the CMake build system.

4.3 Environment modules and dependencies

ICON-CLM depends on several external libraries for its build and execution. Library management on PARAM Rudra is handled using Spack, which allows multiple versions of the same library to coexist, as well as multiple builds of the same version compiled with different compilers. In this way, users have the possibility to load the specific libraries required for a given application while minimizing compatibility issues. In addition to system-wide library installations available on PARAM Rudra, users can also leverage Spack to install additional libraries locally in their user directories as needed.

For this deployment, an environment was used in which all libraries were compiled with the Intel oneAPI 2024.0.0 compilers. This choice was made because many of the required libraries were already compiled with this version on the system, allowing us to maximize reuse of existing installations. Using a consistent compiler version across the entire software stack ensures fully compatible and machine-optimized implementations.

The complete list of libraries used for building ICON-CLM is presented in Table 2. These libraries are categorized into system-wide installations and locally installed libraries. Local Spack installations in the user folder were necessary when a system-wide version compiled with Intel OneAPI 2024.0.0 compilers was not already available on the system.

LIBRARY	INSTALLATION	DEPENDENCIES	SPACK SPEC
Intel OneAPI compilers	System wide	-	@2024.0.0+envmod
Intel OneAPI mkl	System wide	-	@2024.0.0~cluster+envmods~gfortran~ilp64+shared
Intel OneAPI tbb	System wide	-	@2022.3.0+envmods
Intel OneAPI MPI	System wide	-	@2021.12.1~classic-names+envmods~external-libfabric~generic-names~ilp64
Cmake	System wide	-	@3.27.9~doc+ncurses+ownlibs~qtgui
glibc	System wide	-	@2.28
gcc-runtime	System wide	-	@14.2.0
libxml2	System wide	-	@2.13.5~http+pic~python+shared
binutils	System wide	-	@2.45~debuginfod~gas~gprofng~headers~interwork~ld~libiberty~lto~nls~pgo+plugins
libaec	System wide	-	@1.0.6~ipo+shared
ncview	System wide	-	@2.1.9
python	System wide		@3.14.2+bz2+ctypes+dbm~debug~freethreading+libxml2+lzma~optimizations+pic+pyexpat+pythoncmd+readline+shared+sqlite3+ssl~tkinter+uuid+zlib+zstd
perl	User	-	@5.42.0+cpanm+opcode+open+shared+threads
nghttp2	User	-	@1.67.1
openssl	User	perl	@3.6.0~docs+shared
curl	User	nghttp2, openssl	@8.15.0~gssapi~ldap~libidn2~librtmp~libssh~libssh2+nghttp2 build_system=autotools
hdf5	User	-	@1.14.6~cxx+fortran+hl~ipo~java~map+mpi+shared~subfiling~szip~threadsafe+tools
parallel-netcdf	User	perl	@1.14.1~burstbuffer+cxx~examples+fortran+pic+shared
netcdf-c	User	parallel netcdf; hdf5	@4.9.3+blosc~byterange~dap~fsync~hdf4~jna~logging+mpi~nczarr_zip+optimize~parallel-netcdf+pic+shared+szip+zstd
netcdf-fortran	User	netcdf-c, hdf5, parallel-netcdf	@4.6.2~doc+pic+shared
netcdf-cxx	User	netcdf-c, hdf5, parallel-netcdf	@4.2+netcdf4
eccodes	User	-	@2.41.0+aec~fortran~ipo~memfs~netcdf~openmp~png~pt

			hreads+shared~tools

Table 2: Complete list of libraries loaded via Spack for the deployment of ICON-CLM on PARAM Rudra. The first two columns indicate the library name and version. The third column specifies whether the installation is system-wide or local in the user's directory. The fourth column lists dependencies, including only those on locally installed libraries. The fifth column details the enabled features for each library.

The Spack environment file used for the installation, named `install_icon`, is shown in Fig. 4.

```
None
spack:
  specs:
  - /jbzlcvq #- intel-oneapi-compilers@2024.0.0
  - intel-oneapi-mpi@2021.12.1
  - cmake@3.27.9
  - glibc@2.28
  - gcc-runtime@13.2.0
  - intel-oneapi-mkl@2024.0.0
  - libaec@1.0.6
  - perl
  - nghttp2
  - openssl
  - curl
  - netcdf-cxx
  - hdf5 +fortran +hl
  - parallel-netcdf
  - netcdf-c
  - netcdf-fortran
  - eccodes

  compilers:
  - compiler:
      spec: oneapi@2024.0.0
      paths:
                                                                                cc:
/home/apps/SPACK/spack/opt/spack/linux-almalinux8-cascadelake/gcc-13.2.0/intel-onea
pi-compilers-2024.0.0-jtvke3n7wjspfb5g67oiuph7pzlgov7t/bin/icx
                                                                              cxx:
/home/apps/SPACK/spack/opt/spack/linux-almalinux8-cascadelake/gcc-13.2.0/intel-onea
pi-compilers-2024.0.0-jtvke3n7wjspfb5g67oiuph7pzlgov7t/bin/icpx
                                                                              f77:
/home/apps/SPACK/spack/opt/spack/linux-almalinux8-cascadelake/gcc-13.2.0/intel-onea
pi-compilers-2024.0.0-jtvke3n7wjspfb5g67oiuph7pzlgov7t/bin/ifx
                                                                               fc:
/home/apps/SPACK/spack/opt/spack/linux-almalinux8-cascadelake/gcc-13.2.0/intel-onea
pi-compilers-2024.0.0-jtvke3n7wjspfb5g67oiuph7pzlgov7t/bin/ifx
      operating_system: almalinux8
      target: cascadelake
      modules: []
      environment: {}
      extra_rpaths: []
 packages:
    all:
```

```
      require:
      - "%oneapi@2024.0.0"
    bison:
      require: "%gcc@13.2.0"
    flex:
      require: "%gcc@13.2.0"
    m4:
      require: "%gcc@13.2.0"
    pkgconf:
      require: "%gcc@13.2.0"
    zstd:
      require: "%gcc@13.2.0"
    zlib-ng:
      require: "%oneapi@2024.0.0"
    snappy:
      require: "%gcc@13.2.0"
    lz4:
      require: "%gcc@13.2.0"
    bzip2:
      require: "%gcc@13.2.0"
    c-blosc:
      require: "%gcc@13.2.0"
    libaec:
      require: "%oneapi@2024.0.0"
    perl:
      require: "%gcc@13.2.0"
    hdf5:
      require: "%oneapi@2024.0.0"
      variants: +fortran +hl +mpi
    netcdf-c:
      require: "%oneapi@2024.0.0"
    # Ensure netcdf-fortran also uses the same
    netcdf-fortran:
      require: "%oneapi@2024.0.0"
    intel-oneapi-mpi:
      externals:
      - spec: intel-oneapi-mpi@2021.12.1%oneapi@2024.0.0
        prefix: /home/mstocchi/spack/opt/spack/linux-cascadelake/intel-oneapi-mpi
-2021.12.1-exhu62otavz6buc3urntv7stl75hpvrf/mpi/latest
      buildable: false
  view:
    default:
      root: .spack-env/view
      link: all
      exclude:
      - intel-oneapi-mpi
  concretizer:
    unify: when_possible
```

Figure 4: Spack environment file “install_icon”

4.4 The configuration, build and installation process

As mentioned earlier, the entire configuration, build, and installation process was managed using a dedicated configuration script. In the initial steps of the process (see Fig. 5) the build type and simulation mode are determined, along with system-specific information such as the locations of key libraries, including MPI, Perl, and the NetCDF family. A compiler test is performed in order to ensure that the required binaries and libraries are correctly detected in these directories.

```
Shell
[INFO] Build type: RELEASE (default)
[INFO] Simulation mode: CLM
[INFO] Loading environment modules...
[INFO] Configuring environment variables...
[INFO] Environment variables configured:
HDF5ROOT=/home/fmele/spack/opt/spack/linux-cascadelake/hdf5-1.14.6-3xsrkwlecdwwbtmo
h4zmkujpsgpha6cm
NETCDROOTC=/home/fmele/spack/opt/spack/linux-cascadelake/netcdf-c-4.9.3-molhustzbey
k4hbw5ux24zjiytxvhc7
NETCDROOTF=/home/fmele/spack/opt/spack/linux-cascadelake/netcdf-fortran-4.6.2-pfuxh
zgs4qbdetk672rx5f3gzq5utbps
ECCODESROOT=/home/fmele/spack/opt/spack/linux-cascadelake/eccodes-2.41.0-yeqhsuobjg
vr4w4i3zy3sbazsl3y4tqs
MPIROOT=/home/apps/SPACK/spack/opt/spack/linux-almalinux8-cascadelake/oneapi-2024.0
.0/intel-oneapi-mpi-2021.12.1-cebrslruvi5m5a2ujgmchlszvrewdkf/mpi/2021.12
PERL=/bin/perl
[INFO] Configuring compilers...
[INFO] Build type: RELEASE (optimization enabled)
[INFO] Preparing build directory...
[SUCCESS] Build directory: /home/fmele/icon_inst/icon-model/build_CLM_release
[SUCCESS] Install directory: /home/fmele/icon_inst/icon-model/install_CLM_release
[INFO] Testing Fortran compiler...
Compiling test with FC=mpiifx...
[SUCCESS] Compiler is working correctly
Compiler test OK
```

Figure 5: Output snippet illustrating the Initial steps of the ICON-CLM installation procedure.

After this initial check, ICON and all the required external libraries are configured (Fig. 6).

```
Shell
[INFO] Configuring ICON...
configure: loop exchange is enabled because no GPU support is requested: disable
the loop exchange if required (--disable-loop-exchange)
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking for the fully qualified domain name of the host... login07
checking whether the Fortran compiler works... yes
```

```
checking for Fortran compiler default output file name... a.out
checking for suffix of executables...
checking for suffix of object files... o
checking whether we are using the GNU Fortran compiler... no
checking whether mpiifx accepts -g... yes
```

Figure 6: Output snippet showing the configuration checks during the ICON installation process.

Once the configuration is complete, ICON and the selected external packages are built using make, which compiles the source code, links the necessary libraries, and generates the optimized ICON-CLM binaries ready for execution on PARAM Rudra (Fig. 7). Because several external packages were enabled and multiple optimization options were applied, the build process took approximately one hour.

Shell

```
[SUCCESS] Configuration completed successfully!
[INFO] Compiling ICON...
[INFO] Using 1 parallel processes (out of 1 available)
make[1]: Entering directory '/home/fmele/icon_inst/icon-model/build_CLM_release'
DEPGEN c_binding.d
DEPGEN support/util_uuid.c.d
DEPGEN support/util_multifile_restart.c.d
DSL4JSB externals/jsbach/src/turbulence/mo_turb_memory_class.pp-jsb.f90
DEPGEN externals/jsbach/src/turbulence/mo_turb_memory_class.pp-jsb.f90.d
DSL4JSB externals/jsbach/src/turbulence/mo_turb_interface.pp-jsb.f90
DEPGEN externals/jsbach/src/turbulence/mo_turb_interface.pp-jsb.f90.d
DSL4JSB externals/jsbach/src/turbulence/mo_turb_init.pp-jsb.f90
DEPGEN externals/jsbach/src/turbulence/mo_turb_init.pp-jsb.f90.d
DSL4JSB externals/jsbach/src/turbulence/mo_turb_constants.pp-jsb.f90
DEPGEN externals/jsbach/src/turbulence/mo_turb_constants.pp-jsb.f90.d
```

Figure 7: Screenshot showing the ICON compilation process.

Once the build procedure is complete, an ICON executable is generated. Additional final checks are then performed in order to verify the correctness and completeness of the installation process, as well as to provide detailed information about the installation (Fig. 8).

Shell

```
Installation completed successfully!
[INFO] Verifying installation... [SUCCESS]
ICON executable found:
 /home/fmele/icon_inst/icon-model/install_CLM_release/bin/icon
-rwxr-xr-x       1       fmele       fmele       110M       Dec       24       23:51
/home/fmele/icon_inst/icon-model/install_CLM_release/bin/icon
```

```
[INFO] Files installed in /home/fmele/icon_inst/icon-model/install_CLM_release:
   /home/fmele/icon_inst/icon-model/install_CLM_release/bin/icon   [INFO]   Creating
environment script... [SUCCESS]
Environment script created:
 /home/fmele/icon_inst/icon-model/install_CLM_release/env_icon.sh

=====================================
INSTALLATION SUMMARY
=====================================
Build type: release
Effective mode: release
Simulation mode: CLM
Source directory: /home/fmele/icon_inst/icon-model
Build directory: /home/fmele/icon_inst/icon-model/build_CLM_release
Install directory: /home/fmele/icon_inst/icon-model/install_CLM_release

ICON Executable: /home/fmele/icon_inst/icon-model/install_CLM_release/bin/icon
Environment script:
 /home/fmele/icon_inst/icon-model/install_CLM_release/env_icon.sh

To use ICON:
source              /home/fmele/icon_inst/icon-model/install_CLM_release/env_icon.sh
/home/fmele/icon_inst/icon-model/install_CLM_release/bin/icon --help

Log files:
    Configure: /home/fmele/icon_inst/icon-model/build_CLM_release/configure.log
    Make: /home/fmele/icon_inst/icon-model/build_CLM_release/make.log
    Install: /home/fmele/icon_inst/icon-model/build_CLM_release/install.log

=====================================
[SUCCESS] INSTALLATION COMPLETED SUCCESSFULLY!
```

Figure 8: Screenshot showing the ICON installation summary

4.5 Configuration of SPICE

The SPICE runtime suite (Starter Package for ICON-CLM Experiments, v2.3) was downloaded from the Zenodo repository using `wget`, and extracted using `tar` with the following commands

Shell

```
wget   "https://zenodo.org/records/10047046/files/spice-v2.3.tar.gz?download=1"   -O
spice-v2.3.tar.gz
tar -xvf spice-v2.3.tar.gz
```

SPICE requires additional dependencies beyond those of ICON, in particular the post-processing libraries Climate Data Operators (CDO) and netCDF Operators (NCO). It also relies on the R programming language for statistical and visualization routines, as well as the Python libraries Jinja2, which is used for managing template files, and six,

which ensures smooth interoperability between Python 2 and Python 3 scripts. These additional dependencies and libraries were installed on PARAM Rudra using Spack creating another environment which we called “spice_env”, defined by the configuration file shown in Fig. 9:

```
None
spack:
  specs:
  - cmake
  - cdo %gcc@13.2.0
  - nco %gcc@13.2.0
  - ncview %gcc@13.2.0
  - netcdf-fortran %gcc@13.2.0
  - netcdf-c %gcc@13.2.0
  - r
  - python
  - py-jinja2
  - py-six
  - py-pip
  view: true
  concretizer:
    unify: true
```

Figure 9: file for the “icon_env” Spack environment

The configuration of SPICE was performed following the installation guide, which involves setting up these dependencies in the user environment and compiling Fortran programs that, along with Python and shell scripts, implement the workflow for data preparation, model execution, archiving, and post-processing. This process ensures that SPICE can manage ICON-CLM simulations efficiently, including handling restart cycles, interpolating forcing data onto the model grid, and generating CMIP-compliant output files for subsequent analysis.

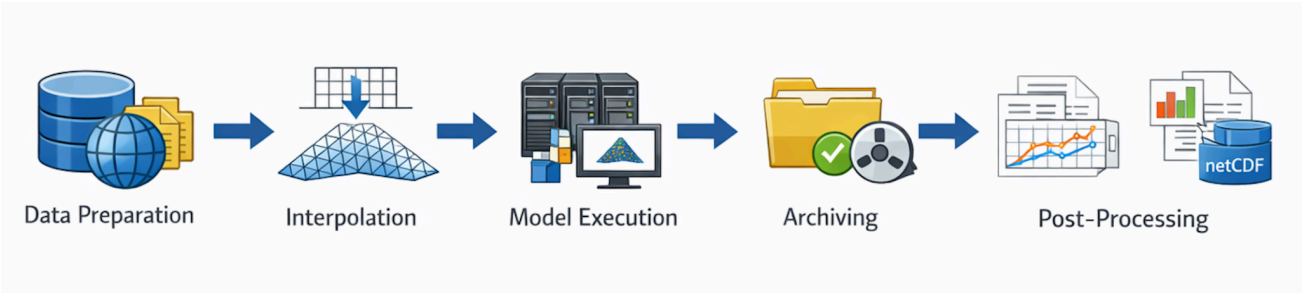

Figure 10: SPICE Workflow Overview

5 Testing

After successfully building ICON and SPICE, the next step was to test the installation. The original plan was to use the example cases provided in the SPICE installation tutorial, which involves a CLM simulation over a European domain with 50 km grid resolution and

ERA-Interim reanalysis data as boundary conditions. This test is designed for execution on the DKRZ computing systems, where the required reanalysis data are readily accessible. Due to the unavailability of these data, a simplified test was conducted instead.

The objective of this test was to verify that the ICON binary was correctly compiled and properly linked to MPI and the required external libraries. This test case was performed using the `icon_Zurich_R19B9_beo_v3_DOM01.nc` domain, which is centered on the city of Zurich (CH) and included in the test cases of the ART external module. Consequently, it is downloaded along with the ICON source code. The file can be found at the following path, relative to the root ICON folder:

```
None
icon-model/externals/art/runctrl_examples/oem_testcase_files/grid/icon_Zurich_R19B9
_beo_v3_DOM01.nc
```

The simulation was configured using an *`icon_master.namelist`* file (Fig. 11), which contains all the necessary parameters and settings for executing the run, and a `NAMELIST_atm` file (Fig. 12), specifying the model parameters.

```
None
&master_nml
 lrestart               = .false.
/
&master_model_nml
 model_type             = 1
 model_name             = "ATMO"
 model_namelist_filename = "NAMELIST_atm"
 model_min_rank         = 0
 model_max_rank         = 0
 model_inc_rank         = 1
/
&time_nml
 ini_datetime_string    = "2024-01-01T00:00:00Z"
 end_datetime_string    = "2024-01-01T00:05:00Z"
/
```

Figure 11: *icon_master.namelist* file

```
None
&parallel_nml
 nproma         = 8
/
&run_nml
 num_lev        = 50
 dtime          = 10.0
```

```
 ltestcase      = .true.
 iforcing       = 0
 msg_level      = 15
/
&nh_testcase_nml
 nh_test_name   = 'aes_bubble'
/
&grid_nml
 dynamics_grid_filename = "grid.nc"
/
```

Figure 12: *namelist_atm* file

The job was submitted using a SLURM script (Fig. 13), after activating the spack environment "*spice_env*".

Shell

```
#!/bin/bash
#SBATCH --job-name=icon_test
#SBATCH --nodes=1
#SBATCH --ntasks=2
#SBATCH --time=00:05:00
#SBATCH --account=nsmext
#SBATCH --output=icon_output_%j.log
#SBATCH --error=icon_error_%j.log

set -e
export LD_LIBRARY_PATH=$(spack location -e \
install_icon)/.spack-env/view/lib:$LD_LIBRARY_PATH
mpirun  -n 2 /home/mstocchi/icon-model/install_CLM_release/bin/icon
```

Figure 13: *SLURM script used to submit the job*

The run completed successfully. Although this minimal simulation does not provide relevant scientific results, inspection of the initial lines of the log files confirmed that the ICON-CLM deployment was successful. In particular, MPI processes were correctly initialized and all required external libraries were properly linked, validating the system setup for future, more complex simulations (Fig. 14).

Shell

```
ICON MPI interface runtime information:
 mo_mpi::start_mpi Used MPI version: 3.1
 mo_mpi::start_mpi ICON: Globally run on 2 mpi processes.

 ICON runs on 2 mpi processes.
```

```
executable: /home/mstocchi/icon-model/install_CLM_release/bin/icon
date: 20250126
time: 142625
user: mstocchi (mstocchi)
host: rpcn453 (Linux 4.18.0-477.27.2.el8_8.x86_64 x86_64)
version: 2025.04
revision: icon-2025.04-public-0-g1be2ca66ea0de149971d2e77e88a9f11c764bd22
repository: https://gitlab.dkrz.de/icon/icon-model.git
local branch: unknown
model components:
  JSBACH: jsbach-4.20p9-290-g766282ee5592ce0d52c88aeca43f38fa54976f4e
  ART: icon-2025.04-public-0-g1be2ca66ea0de149971d2e77e88a9f11c764bd22
application libraries:
  ECRAD: ecrad-safeguard-09666303-3-g4b57dab3be1db195afd8325736296f1bfca6cd63
  RTE-RRTMGP: v1.7-17-g11a1825f6b01314e440f3538f77a0a48f3cfdaa6
infrastructure and support libraries:
  ICONMATH: 1.2.0-0-g11f6c57a46b32644d3c872e9cdb18d0ad27911db
  FORTRAN-SUPPORT: 2.1.0-0-g02329904ff8f05bb85ce99312770eceaa8809e62
  TIXI:
    version: 2.2.2
    revision: icon-2025.04-public-0-g1be2ca66ea0de149971d2e77e88a9f11c764bd22
  YAC:
    version: 3.6.2
    revision: v3.6.2_p2-0-ge34f6c402bd3dbf088b9457c2fdcc6678b9a761c
  MTIME: 1.3.0-1-g04f02ccbd765104a1570355c5d3fdfcfdad11c4c
  CDI:
    version: 2.5.2
    revision: cdi-2.5.2-1-gf4f90b94c69716eed898d5e3c76a3e5d2f843095
other libraries:
  ECCODES: 2.41.0
  NetCDF-C: 4.9.3
  MPI: Intel(R) MPI Library 2021.12 for Linux* OS...
compilers:
   Fortran: Intel 2024.0.0 (Intel(R) Fortran Compiler for applications running on
Intel(R) 64, Version 2024.0.0 Build 20231017)
  C: Intel 2024.0.0 (GNU 4.2.1)
```

Figure 14: *ICON-CLM test log file*

6 Conclusions and remarks

This deliverable documents the work carried out for the deployment of ICON-CLM on the PARAM-Rudra supercomputer at IUAC. In particular, an MPI-parallelized version of the model was installed, together with the SPICE runtime environment. As documented, the ICON and SPICE binaries have been installed in user mode for the time being. Thanks to the collaboration with the C-DAC team and the support of the PARAM-Rudra system, the next step will be to share the installation experience with them and to provide, in the near future, a system-wide installation of libraries and applications that will be easily accessible to all users.

All required external libraries, including the full NetCDF dependency chain (HDF5, NetCDF-C, NetCDF-Fortran, Parallel-NetCDF), were built using Intel oneAPI compilers with MPI support via Spack package manager, available on the C-DAC machine. This approach ensured library consistency and optimized performance on PARAM Rudra's Intel CPUs.

A minimal test case was executed to verify that the ICON installation works correctly, that the MPI processes initialized properly, and that all the external libraries were correctly linked.

The next step will focus on simulations over regional domains, starting with a domain centered on the city of Pune. Meteorological data will be transferred to PARAM Rudra to provide boundary conditions. These tests will also evaluate model scalability and help optimize resource allocation for larger simulations.

7 References

[1] **Zängl**, G., **Reinert**, D., **Rípodas**, P., and **Baldauf**, M. (2015). *The ICON (ICOsahedral Nonhydrostatic) modelling framework of DWD and MPI-M: Description of the nonhydrostatic dynamical core*. Quarterly Journal of the Royal Meteorological Society, Vol. 141, No. 687, pp. 563–579. https://doi.org/10.1002/qj.2378

[2] **Pham**, T. et al., (2021). *ICON in Climate Limited-area Mode (ICON release version 2.6.1): a new regional climate model*. Geoscientific Model Development, 14, 985–1005. https://doi.org/10.5194/gmd-14-985-2021

[3] **Campanale**, A. et al., (2025). *Investigating urban heat islands over Rome and Milan during a summary period through the TERRA_URB parametrization of the ICON model*, Urban Clim. 60, 102335, https://doi.org/10.1016/j.uclim.2025.102335

[4] **Loprieno**, L.L. et al., (2026). *Comparative study of the COSMO and ICOn models performances over Italy: key insights from the impactful year 2023*. Atm. Res, 327, 108339, https://doi.org/10.1016/j.atmosres.2025.108339

[5] **Prill**, F. et al., (2025): *ICON Tutorial 2025: Working with the ICON model. Zenodo.* https://doi.org/10.5676/DWD_pub/nwv/icon_tutorial2025

[6] **Beate**, G. et al., (2025). *SPICE (Starter Package for ICON-CLM Experiments)*. Zenodo. https://doi.org/10.5281/zenodo.10047046

[7] **ICON partnership** (DWD; MPI-M; DKRZ; KIT; C2SM) (2025). ICON release 2025.04. World Data Center for Climate (WDCC) at DKRZ. DOI: https://doi.org/10.35089/wdcc/iconrelease2025.04

[8] **Gassmann** A. and **Herzog** H.-J. (2008). *Towards a consistent numerical compressible non-hydrostatic model using generalized Hamiltonian tools*, Q. J. R. Meteorol. Soc. 134: 1597–1613 DOI: https://doi.org/10.1002/qj.297